Universidad Nacional Autónoma de México

INSTITUTO DE INVESTIGACIONES EN MATEMÁTICAS APLICADAS Y EN SISTEMAS
posgrado en ciencia e ingeniería de la computación

Redes neuronales profundas para el reconocimiento del habla con datos audiovisuales

T E S I N A

que para optar por el grado de:

Especialista en Cómputo de Alto Rendimiento

P R E S E N T A :

Fidel López Saca

TUTOR

Dr. Paul Erick Méndez Monroy

Ciudad Universitaria, Ciudad de México, Junio 2022.

Gracias

A mi mamá y hermano por la confianza, paciencia, apoyo y ánimo.

A todos mis amigos por su valiosa amistad a pesar de la distancia.

A las personas que han compartido una parte de su vida y experiencia.

Agradecimientos

A través de estas líneas expreso mi profundo agradecimiento a mi asesor el Dr. Paul Erick Méndez Monroy por su valioso tiempo, ayuda, paciencia, consejos, comentarios, sugerencias y críticas semanales durante el desarrollo de este proyecto. A los miembros del jurado el Dr. Oscar Alejandro Esquivel Flores y Dr. Jorge Luis Pérez González por sus valiosos comentarios y sugerencias en la revisión de este trabajo.

A todos los profesores y miembros de la Especialidad en Cómputo de Alto rendimiento por el tiempo invertido en la enseñanza. Al coordinador el Dr. Javier Gómez Castellanos y a la asistente de procesos Maria de Lourdes González Lora, que siempre han estado al pendiente y proporcionando el apoyo necesario para poder continuar y finalizar de forma satisfactoria la especialidad.

A la Universidad Nacional Autónoma de México por darme la oportunidad de cumplir uno más de mis sueños.

Resumen

En los últimos años las redes neuronales profundas han sido populares para el reconocimiento de patrones, los datos principalmente son extraídos de imágenes, audio y texto. Varios trabajos han demostrado que son una herramienta prometedora en diferentes campos como: la clasificación de imágenes, detección de objetos y segmentación. También se han utilizado en otros campos como el reconocimiento del habla usando información obtenida de audio y/o video, a este tipo de información se le conoce como datos audiovisuales. Por ejemplo, las imágenes (extraídas de los videos) pueden ser usadas para realizar una lectura de labios de una persona para saber lo que dice al hablar sin necesidad de que se cuente con el audio, o al contrario, se pueden utilizar las señales de audio para este reconocimiento o incluso se pueden utilizar ambos tipos de datos.

En este trabajo se reporta el proceso para poder utilizar datos audiovisuales para reconocer las palabras que un persona dice al momento de hablar. Se utilizaron dos conjuntos de datos audiovisuales GRID y LRS2, el primero puede ser descargado de forma libre, en el segundo es necesario solicitar un usuario y contraseña para poder tener acceso. Se hizo un acondicionamiento de la información para poder utilizar los datos como entrada a las redes. Del video se hace un pre-procesamiento de los frames para la detección del rostro de una persona, luego se extraen características de la boca, estas características sirven para relacionarlos con las palabras dichas en el video al momento de entrenar los modelos. La contribución se concentra principalmente en el diseño de una red neuronal para el reconocimiento de palabras, esta arquitectura tiene como base la red LipNet, la principal modificación es el tamaño de los kernel en las diferentes capas de convolución, se sustituyó la capa de max pooling por average pooling, entre otras aportaciones descritas en este trabajo. Al final del proyecto se reportaron los resultados como los errores WER y CER obtenidas en los entrenamientos y pruebas. Para procesar el audio se entrenó por separado con una red especializada para este tipo de señales. El Hardware es importante para disminuir los tiempos de entrenamiento, se utilizó una GPU RTX 3080 Ti para procesar los datos de forma paralela.

Índice general

Lista de acrónimos

CNN *Convolutional Neural Network*
AVSR *Audio-Visual Speech Recognition*
STFT *Short-Time Fourier Transform*
RNN *Recurrent Neural Network*
LSTM *Long Short-Term Memory*
GRU *Gated Recurrent Unit*
CTC *Connectionist Temporal Classification*
WER *Word Error Rate*
CER *Character Error Rate*
GPU *Graphics Processing Unit*

Índice de figuras

1 Introducción

El sentido de la vista puede ser imitado extrayendo información de los píxeles que forman las imágenes como: bordes, texturas, colores, entre otros. Sin embargo, las señales que se utilizan para la visión son diferentes al del sonido, olfato, gusto y tacto, por lo que se procesa de forma diferente, en cambio, a una persona le sirve el conjunto de datos para tomar decisiones en diferentes eventos. Por ejemplo, los autos autónomos tienen que reconocer en tiempo real su entorno, aunque, el entorno está en constante cambio con nuevas escenas, ruidos, eventos, etc., sin embargo, si se pueden conocer las acciones del conductor, se pueden inferir los accidentes. Entonces, con base en las acciones realizadas por los humanos, se pueden reconocer sus actividades a realizar, utilizando por ejemplo, la visión artificial (existen conjuntos de datos especializados en este tema [2]). Por otra parte, el sonido es importante para el humano, entre muchas cosas nos sirve para comunicarnos mediante el habla. Aunque durante años de investigación no se han desarrollado equipos capaces de comprender y generar fragmentos de lenguaje natural, hay métodos y algoritmos que son útiles para tratar de emular el habla, como: las redes neuronales convolucionales, las redes de memoria de largo plazo a corto plazo, entre otros [3]. En [1], han demostrado que trabajar con las características visuales combinadas con el audio puede mejorar el reconocimiento, realizaron comparaciones entre dos modelos "*Connectionist Temporal Classification* (CTC) *loss*" y "*sequence-to-sequence loss*", utilizaron el conjunto de datos LRS2-BBC con un *Word Error Rate* (WER) de 48.3 %.

El reconocimiento de voz con datos audiovisuales es útil para poder interactuar entre una persona y una computadora, tiene aplicaciones prácticas en el mundo real, por ejemplo: decirle comandos a un teléfono celular esto puede ser en un entorno ruidoso, resolver el problema del habla simultánea de múltiples hablantes, transcribir películas para personas con problemas de audición, entre otros [4]. También, se pueden analizar los labios, esta es la tarea de decodificar texto a partir del movimiento de la boca de un hablante [5], esta habilidad es aprendida por personas con discapacidad auditiva y es utilizada para interpretar el habla sin escuchar el sonido. Emular la capacidad humana para comprender un mensaje hablado, no solo al escuchar, sino también visualizando a la persona que habla es un desafío. *Deep Learning* [6] o aprendizaje profundo es uno de los métodos de aprendizaje que tiene algoritmos para el análisis masivo de datos, se puede utilizar para resolver diferentes problemas relacionados con imágenes, texto, audio, entre otros, actualmente está siendo utilizado para el reconocimiento del habla. En está área están las redes neuronales convolucionales o *Convolutional Neural Network* (CNN) [7] son utilizadas principalmente para

imágenes, vídeo y audio, también están las redes neuronales recurrentes o *Recurrent Neural Network* (RNN), utilizadas para el análisis de texto [6] con aprendizaje no supervisado.

El Hardware utilizado para el análisis de datos es importante, ya que, al entrenar los modelos pueden tardar días, semanas o meses dependiendo de la cantidad de información y se puede reducir a horas o minutos utilizando una unidad de procesamiento gráfico o *Graphics Processing Unit* (GPU), este y otros dispositivos son útiles al momento de estar analizando grandes cantidades de datos. También influye el Framework y bibliotecas utilizadas para el desarrollo de las redes, como son PyTorch [8] y TensorFlow [9] útiles para la construcción de redes neuronales.

Planteamiento del problema

El reconocimiento del habla de forma automática es un problema abierto, ya que existen diferentes cuestiones como el pre-procesado de datos para ingresarlos como entrada a los modelos de redes neuronales, otra dificultad es la cantidad de datos a utilizar para entrenamiento y prueba. Cada conjunto de datos tiene un formato diferente, una estructura de oraciones diferente, entre otros detalles. Por otro lado, el diseño de una red puede ser simple o complejo depende del tipo de información a analizar, esto implica nuevas formas y métodos a experimentar en diferentes capas como por ejemplo agregar las redes GRU o Transformers para realizar el análisis datos en un tiempo t, otra es experimentar en que capas funcionan mejor y con sus parámetros de entrenamiento. Estos y otros problemas son a los que se enfrenta este proyecto.

1.1. Objetivos

A continuación se describe los objetivos generales y específicos.

Objetivo general

Analizar, diseñar y construir redes neuronales para el procesamiento de datos audiovisuales para reconocer las palabras que una persona dice al momento de hablar.

Objetivos específicos

Los objetivos particulares son los siguientes:

- Analizar los conjuntos de datos audiovisuales usando redes neuronales agregando en las capas intermedias las redes GRU y Transformers.
- Procesar de forma paralela y concurrente los datos audiovisuales en CPUs y GPUs.
- Realizar pruebas de rendimiento en una GPU.

En este trabajo se describe el análisis, desarrollo, entrenamiento y pruebas de redes neuronales profundas para el reconocimiento del habla con datos audiovisuales, desde la selección de un conjunto de datos hasta las pruebas de rendimiento. Se hizo el análisis de la estructura de diferentes redes neuronales como AlexNet, ResNet y LipNet. La propuesta se enfoca en agregar a las capas intermedias diferentes tamaños de filtros principalmente en las capas de convolución así como en las capas de pooling. Para realizar los entrenamientos se utilizaron los conjuntos de datos audiovisuales GRID y LRS2, las cuales pueden ser descargadas desde su página principal (en las siguientes secciones se describe cada una). Cabe destacar que se hizo un acondicionamiento de la información, se reclasificaron los videos y los archivos de texto, de los videos se crearon los archivos de audio, entre otras tareas, esto antes de entrenar las modelos. De los videos se hace un pre-procesamiento para detectar el rostro, después se extrae la boca para guardarlos como imágenes que servirán como datos de entrada a la red. Por otro lado se utilizó la Transformada de Fourier Corta para poder procesar las señales de los audios y poder realizar un análisis de la información destacando las redes GRU y Transformer. Para la codificación se ha utilizado información de diferentes fuentes, en algunas se han modificado el código, en otros casos se ha creado desde cero principalmente en el pre-procesamiento de datos.

Para llevar a cabo este proyecto se han utilizado los conocimientos de la Especialidad en Cómputo de Alto Rendimiento (CAR) como: la programación orientada a objetos, la programación paralela, el uso de la GPU para el cómputo de alto rendimiento, el análisis de la información para procesar los datos de video y audio de forma paralela. También, el aprendizaje del sistema operativo Linux para la configuración e instalación de bibliotecas, las redes neuronales convolucionales, redes neuronales recurrentes, entre otros temas importantes, estas impartidas en los diferentes cursos de CAR.

En el capítulo 2 se describe el marco teórico, con el objetivo de ayudar al lector a comprender los siguientes capítulos. En el capítulo 3 se describen los antecedentes, describiendo algunas de las redes presentes en la literatura para el reconocimiento del habla a través del análisis de video o audio. En el capítulo 4 se describe la metodología y en el capítulo 5 el desarrollo experimental. En el capítulo 6 se hace el análisis de resultados y finalmente en el capítulo 7 las conclusiones.

2 Marco teórico

El reconocimiento de voz audio-visual o *Audio-Visual Speech Recognition* (AVSR) se refiere a la transcripción automática de expresiones habladas utilizando grabaciones de voz y análisis del rostro de una persona de forma simultánea [10]. Emula la capacidad del humano para comprender un mensaje hablado, no solo al escuchar, sino también, visualizando a la persona que habla. En esta área se puede analizar por separado o en conjunto la información por un lado el video y por otro el audio, con el procesamiento de video se pueden analizar solo los labios o lectura de labios, esta es una habilidad que aprenden específicamente las personas con discapacidad auditiva que se utiliza para comprender o interpretar el habla sin escuchar el audio [4]. Otra forma de describirlo es que la lectura de labios es la tarea de decodificar texto a partir del movimiento de la boca de un hablante [5]. Para el procesamiento de este tipo de información, se utilizaron las redes neuronales convolucionales, GRU y Transformer, con métricas de evaluación específicas en está área como son WER y CER. A continuación se describen estos conceptos que son útiles en la construcción de la red.

2.1. *Convolutional Neural Network*

La red neuronal convolucional o *Convolutional Neural Network* (CNN) es una red neuronal, pero a diferencia de las redes tradicionales tiene entre sus capas a la convolución, la cual es una operación entre un filtro y la imagen de entrada, el filtro es utilizado para extraer características de la imagen. La CNN se puede dividir en dos partes una es la extracción de características y la otra es la clasificación, en la extracción de características se encuentran la operación de convolución, pooling, ReLU, entre otras [7], en la parte de clasificación están las capas totalmente conectadas.

Las CNNs como LeNet [11] de Yann LeCun o DenseNet [12] de Huang, han sido utilizadas para clasificación, detección de objetos y segmentación [7], se han registrado resultados superiores a los algoritmos tradicionales como SIFT [13] y HOG [14] utilizados para extraer características generadas a mano. Algunas de estas redes se han utilizado para la clasificación de sonido como SoundNet [15].

La convolución en 2D la definen en [16] como sigue:

$$[\mathbf{conv}(\mathbf{x}, \mathbf{w})]_{c'ij} = \sum_{c=1}^{C} \sum_{i'=1}^{k_w} \sum_{j'=1}^{k_h} W_{c'ci'j'} X_{c,i+i',j+j'}, \tag{2.1}$$

sin sesgo y con un paso de uno de C canales a C' canales, x son los valores de entrada y w son los pesos. También en el mismo escrito definen a la convolución espacio temporal como sigue:

$$[\mathbf{stconv}(\mathbf{x},\mathbf{w})]_{c'tij} = \sum_{c=1}^{C}\sum_{t'=1}^{k_t}\sum_{i'=1}^{k_w}\sum_{j'=1}^{k_h} W_{c'ct'i'j'} X_{c,t+t',i+i',j+j'}, \tag{2.2}$$

donde se puede procesar lo información de video a través del tiempo haciendo varias convoluciones.

Además de la convolución se pueden utilizar las redes GRU para poder analizar la información audio-visual en un tiempo t extrayendo datos de video o audio como se explica a continuación.

2.2. GRU

La unidad recurrente cerrada o *Gated Recurrent Unit* (GRU) es un tipo de red neuronal recurrente, tiene mejoras como por ejemplo agregar celdas y puertas (cells and gates) para propagar la información en más pasos de tiempo y aprender a controlar este flujo de información, es parecida a la red neuronal de memoria a corto-largo plazo o *Long Short-Term Memory* (LSTM) [5]. Puede ser utilizada desde PyTorch llamando a la clase torch.nn.GRU(*args, **kwargs) [17] está definida como sigue:

$$\mathbf{r_t} = \sigma(W_{ir}x_t + b_i r + W_{hr}h_{(t-1)} + b_{hr}), \tag{2.3}$$
$$\mathbf{z_t} = \sigma(W_{iz}x_t + b_i z + W_{hz}h_{(t-1)} + b_{hz}), \tag{2.4}$$
$$\mathbf{n_t} = tanh(W_{in}x_t + b_i n + rt * (W_{hn}h_{(t-1)} + b_{hn})), \tag{2.5}$$
$$\mathbf{h_t} = (1 - z_t) * n_t + z_t * h_{(t-1)}, \tag{2.6}$$

donde h_t es el estado oculto en el tiempo t, x_t es la entrada en el tiempo t, $h_{(t-1)}$ es el estado oculto inicial en el tiempo $(t-1)$ o el estado inicial en el tiempo 0, y r_t, z_t, n_t son el reinicio, actualización y nueva, respectivamente. σ es la función sigmoide y $*$ es el producto Hadamard.

También existe otra tipo de red útil para el análisis de audio, como se describe en la siguiente sección.

2.3. *Transformer*

En el trabajo [18] describen un nuevo modelo al que llaman Transformer, tiene un mejor rendimiento que las redes neuronales convolucionales y recurrentes para la traducción de lenguas por ejemplo de Inglés a Alemán, requieren menos tiempo de entrenamiento que las recurrentes y son paralelizables. La principal característica es que prescinde de la recurrencia y la convolución, solo se basa en mecanismos de

atención. Otro trabajo que le da continuidad a este modelo se describe en [19], donde describen a BERT (Bidirectional Encoder Representations from Transformers), está diseñado para entrenar representaciones bidireccionales a partir de texto sin etiquetar, PyTorch se basa en estos modelos para crear la función torch.nn.Transformer usado principalmente para las traducciones.

Otro tema importante son las métricas de evaluación de una red neuronal, porque se puede analizar si la red aprende o deja de aprender lo cual puede implicar continuar entrenando o cambiar de parámetros en caso de que sea necesario.

2.4. Métricas de evaluación

Las métricas para evaluar las redes neuronales juegan un papel importante para un clasificador, principalmente se utilizan en entrenamiento y pruebas. De una matriz de confusión se toma lo siguiente: TP son los verdaderos positivos, estos son los elementos que se clasifican de forma correcta, al contrario se tiene a TN que son los verdaderos negativos, por otro lado, también se toman en cuenta a FN que son los falsos negativos y finalmente FP son falsos positivos estos son los mal clasificados [20].

Al entrenar las redes neuronales convolucionales usualmente se utilizan las siguientes métricas [20]:

- Exactitud (Accuracy): calcula la proporción de clases predichas correctas con respecto al número total de muestras evaluadas. Calcula la cantidad de predicciones positivas que fueron correctas.

$$\textbf{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.7}$$

- Sensibilidad (Sensitivity/Recall): calcula la proporción de clases positivos que se clasifican correctamente.

$$\textbf{Sensitivity} = \frac{TP}{TP + FN} \tag{2.8}$$

- Especificidad (Specificity): calcula la proporción de clases negativas que se clasifican correctamente.

$$\textbf{Specificity} = \frac{TN}{FP + TN} \tag{2.9}$$

- Precisión (Precision): calcula el porcentaje de casos positivos detectados.

$$\textbf{Precision} = \frac{TP}{TP + FP} \tag{2.10}$$

- F1-Score: calcula el promedio armónico entre la "Sensibilidad" y "Precisión".

$$\textbf{F1}_{\textbf{score}} = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{2.11}$$

Sin embargo, para medir el rendimiento del modelo con datos audiovisuales se utilizan las siguientes tasas de error.

2.4.1. WER y CER

Para la evaluación del modelo las métricas de validación que se utilizaron son *Character Error Rate* (CER) [21] y *Word Error Rate* (WER) [22], ya que es necesario comparar el texto decodificado con el texto original para evaluar las clasificaciones erróneas. Se obtienen de la siguiente forma:

$$\mathbf{CER} = \frac{C_S + C_D + C_I}{C_N}, \tag{2.12}$$

$$\mathbf{WER} = \frac{W_S + W_D + W_I}{W_N}, \tag{2.13}$$

donde C son caracteres, W son palabras. S es el número de caracteres sustituidos por clasificaciones incorrectas, D es el número de eliminaciones que no deberían estar presentes para los caracteres decodificados, I es el número de caracteres insertados para caracteres no seleccionados y N es el número total de caracteres correctas.

Al entrenar una red se necesita Hardware con alto poder de cómputo, en este proyecto se utilizó un servidor con una GPU NVIDIA GeForce RTX 3080 Ti con 12GB. Las ventajas de utilizar este hardware se describen en la siguiente sección.

2.5. GPUs

Las unidades de procesamiento gráfico o *Graphics Processing Unit* (GPU) [23] han sido utilizados para poder entrenar las redes neuronales profundas, debido a su poder de cómputo y su bajo costo (entre otras razones). En la investigación son útiles cuando se están haciendo experimentos, ya que al utilizarlos se pueden disminuir los tiempos de entrenamiento y pruebas, por ejemplo de días a unas cuantas horas, claro dependiendo de diferentes factores como el hardware y el software con que se cuente. Existen diferentes empresas que desarrollan este tipo de hardware pero para utilizarlos se necesita de software especializado, por ejemplo NVIDIA lanzó CUDA (Compute United Device Architecture) para poder utilizar sus diferentes GPUs como: GeForce, Quadro y Tesla, RTX, entre otros, también se tiene que instalar los drivers y por otro lado se pueden utilizar bibliotecas que simplifican el acceso a este tipo de hardware.

3 Estado del arte

En el trabajo de Harry McGurk y John MacDonald de 1976 [24] describen la influencia de la visión sobre la forma de reconocer el habla. Actualmente existen trabajos que continúan en este camino, analizando los videos, los audios o ambos. En las siguientes secciones se describen los trabajos relacionados y la influencia al presente proyecto.

3.1. Reconocimiento de palabras utilizando imágenes

Principalmente la lectura de labios tiene como objetivo reconocer palabras u oraciones con base en el movimiento de labios sin importar que el audio no esté disponible, LipNet [5] es un buen ejemplo con respecto a este tipo de reconocimiento, LipNet es una arquitectura de red neuronal que mapea secuencias de fotogramas de diferentes videos a texto, haciendo uso de convoluciones espacio-temporales, una red recurrente y *Connectionist Temporal Classification* (CTC). LipNet logró una precisión del 95.2 % utilizando el conjunto de imágenes GRID [25], superando a las personas expertas en reconocer la lectura de labios.

En otro trabajo [1] realizaron diferentes entrenamientos y pruebas donde la red con mejor rendimiento fue TM-seq2seq, que logra un WER del 48.3 % con el conjunto de datos LRS2 [26], logra una mejora de más del 22 % en comparación con el anterior 70.4 %.

Para este proyecto se hace uso *Connectionist Temporal Classification* (CTC) útil para calcular el error en una serie temporal continua. LipNet se toma como base para realizar el análisis de las imágenes (frames) obtenidas de los videos.

3.2. Reconocimiento de palabras con datos audiovisuales

En el trabajo [1] describen el reconocimiento de frases y oraciones analizando los rostros de personas que están hablando, esto independiente de que contenga audio o no. Comparan dos modelos de lectura de labios: *Connectionist Temporal Classification* (CTC) y la otra con pérdida de secuencia-secuencia utilizando una arquitectura *transformer self-attention* [1] en la figura 3.1 se muestra el modelo propuesto. Los

resultados que obtuvieron muestran que los movimientos de la boca proporcionan información importante en el reconocimiento del habla cuando la señal de audio tiene ruido. También describen que hay un mejor rendimiento cuando la señal de audio es limpia. Realizaron experimentos audiovisuales, solo con audio, y otro agregando ruido a las oraciones originales, el reconocimiento de voz en donde hay ruido es un problema abierto y desafiante. La combinación de las dos formas proporciona una mejora significativa, ya que la tasa de error de palabras se reduce significativamente, hasta en un 30 %. En particular, los modelos audiovisuales funcionan mucho mejor que los de video o solo de audio en presencia de un ruido fuerte en el ambiente [1].

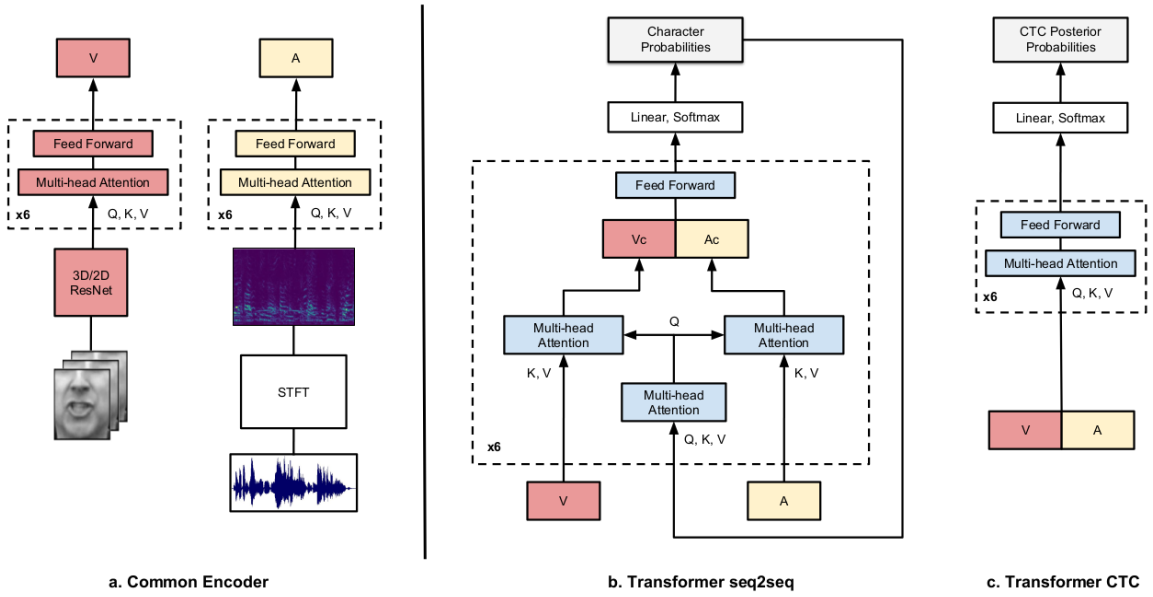

Figura 3.1: Modelos Transformer y CTC [1].

En [4] transforman la señal de audio a una imagen utilizando la transformada de Fourier corta o *Short-Time Fourier Transform* (STFT), la imagen resultado es la entrada a una red neuronal convolucional (CNN). En este trabajo proponen un algoritmo para el reconocimiento automático de voz utilizando datos audiovisuales, combinando la STFT con una CNN en 3D, para extraer las características del video. Las características audiovisuales se concatenan en un solo vector de características así se obtienen resultados de las secuencias de video. Las características extraídas se ingresan en una nueva capa GRU [5] y se entrenan usando CTC [4], las probabilidades de caracteres son las salidas de la red.

Existen otros trabajos que se han desarrollado para reconocer lo que dice una persona con base en el análisis de los labios y el audio como se describe a continuación. En [3] se hace una revisión de los avances en el área de reconocimiento de voz con datos audiovisuales, describen conjuntos de datos utilizados generalmente, modelos y formas en que se pueden transformar las datos para poder ser utilizados en las redes neuronales profundas. Gao et al. [27] separan el audio de diferentes sonidos que hay

de fondo, la ventaja de esto es que no solo analizan los labios sino las expresiones de las personas que están hablando [28].

Para realizar el procesamiento de señales de audio se puede hacer uso STFT y las redes GRU para este trabajo.

3.3. Audio y vídeo no sincronizados

En el trabajo [1] evaluaron el desempeño de los modelos audiovisuales cuando las entradas de audio y video no están alineadas temporalmente. Dado que el audio y el video se han sincronizado en el conjunto de datos, cambiaron sintéticamente los frames de video para lograr un efecto de desincronización. Entonces pre-entrenaron un modelo para calibrarlo y el efecto de desincronización prácticamente desaparezca. El modelo TM-seq2seq funciona mejor para la lectura de labios en términos de WER, cuando no se analiza el audio. Para tareas solo de audio o audiovisuales, los dos métodos funcionan de manera similar.

Los modelos TM-CTC y TM-seq2seq tienen una arquitectura más compleja y son más difíciles de entrenar, ya que el modelo audiovisual completo tarda aproximadamente 8 días en completar el entrenamiento para los conjuntos de datos GRID y LRS2, en una sola GPU GeForce Titan X con 12 GB de memoria. El modelo audiovisual TM-CTC entrena más rápido, es decir, en aproximadamente 5 días en el mismo hardware.

Pre-entrenar un modelo, no con las oraciones completas si no con algunas palabras, es útil para entrenar en un menor tiempo y también para que la red se adapte a la sincronización de la información obtenida de audio y video, se intentará implementar al final de este proyecto.

3.4. Pre-procesamiento de datos

En [29] y [1] se describe el proceso para el pre-procesamiento de datos audiovisuales, para poder ingresar la información a las redes neuronales. En la figura 3.2 se muestra el proceso para poder realizar el entrenamiento. Del video se obtienen los frames o imágenes, después se detecta el rostro por algún método de detección, del rostro se obtienen las características faciales de cuanto una persona habla, después se sincroniza con el audio para el análisis de la información en un tiempo t, también se hace un análisis de las oraciones para poder verificar que el algoritmo aprenda de forma correcta, finalmente se entrena el modelo con la información audiovisual.

Se adopta esta metodología para el pre-procesamiento de los conjuntos de datos, principalmente la forma en que se detecta el rostro y parte de la boca.

3.5. Conjunto de datos

Existen diferentes conjuntos de datos que pueden ser utilizados para trabajar con datos audiovisuales [30], por ejemplo:

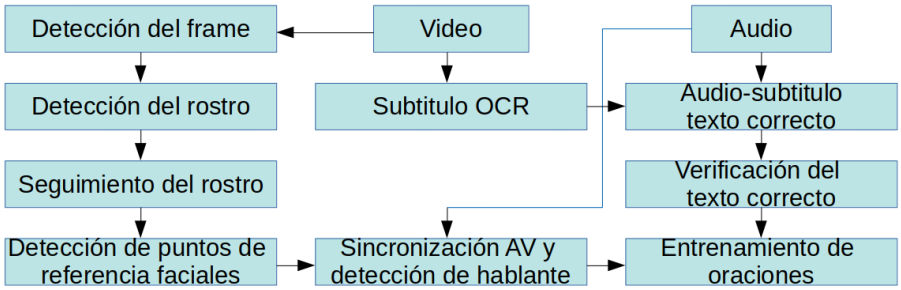

Figura 3.2: Pre-procesamiento de datos.

- GRID audiovisual sentence corpus [25], el corpus consta de grabaciones (faciales) de audio y video de 1,000 oraciones pronunciadas por cada uno de los 34 hablantes (18 hombres, 16 mujeres).
- The Oxford-BBC Lip Reading Sentences 2 (LRS2) Dataset [26], formado por la recopilación y el pre-procesamiento de miles de videos de la televisión británica.
- The Oxford-BBC Lip Reading in the Wild (LRW) Dataset [31].
- Lip Reading Sentences 3 (LRS3) Dataset [32].
- Modality [33], consta de más de 30 horas de grabaciones.

De estos conjuntos de datos audio-visuales se toma a GRID y LRS2, ya que existe una amplia documentación de donde son utilizadas para realizar entrenamientos y pruebas.

3.6. Entrenamiento y pruebas

Para realizar los entrenamientos y pruebas de las redes se necesitan computadoras o servidores con gran poder de cómputo, pero una vez entrenadas se pueden implementar en dispositivos más pequeños como las NVIDIA Jetson Nano e incluso en celulares. El proceso de entrenamiento se muestra en la figura 3.3. Usualmente los conjuntos de datos se dividen en entrenamiento y prueba, en otras ocasiones se divide en un subconjunto más, esta es la validación, principalmente para realizar un entrenamiento y afinar la red. Cuando se tiene la red entrenada se puede implementar en dispositivos con menor capacidad de cómputo.

Usualmente los conjuntos de datos ya están separadas en entrenamiento y pruebas, pueden estar separadas en diferentes rutas o con archivos como un CSV o texto plano. Entonces esta información se utiliza para entrenar, hacer pruebas y finalmente comparar con otros trabajos realizados por otros autores.

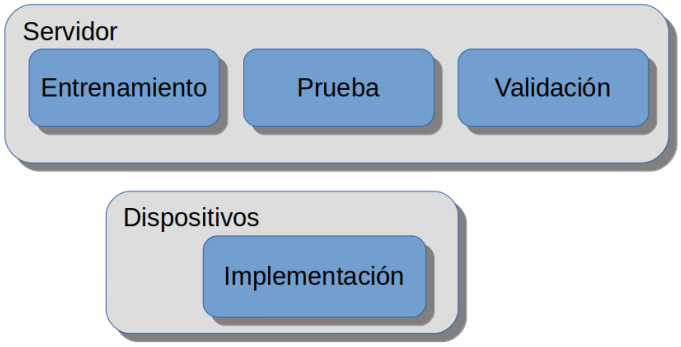

Figura 3.3: Entrenamiento e implementación.

3.7. NVIDIA Nsight

NVIDIA proporciona software para realizar el análisis de rendimiento de dispositivos como GPU y CPU útiles para cuando se programa código especializado en alto rendimiento, un ejemplo es NVIDIA Nsight System y el otro es Nsight Compute. Nsight es un entorno de desarrollo y análisis para la optimización del uso de código en diferentes dispositivos. Existen otros como:

- Nsight Graphics: para depurar/optimizar cargas de trabajo de gráficos.
- Nsight Deep Learning Designer: aquí se pueden diseñar redes neuronales.
- Nsight Eclipse: IDE para codificar.

El análisis de rendimiento del código y del hardware se puede realizar desde un servidor a diferentes dispositivos, también se pueden analizar otros servidores como se muestra en la figura 3.4.

3.7.1. Nsight System

Nsight System fue desarrollado para analizar el rendimiento de aplicaciones en un sistema, tiene las siguientes características:

- Da seguimiento a las actividades del sistema de CPU multi-núcleo y multi-GPU.
- Localiza opciones para optimizar (lugares donde la CPU y GPU no están siendo utilizados).
- Muestra uso y estado de subprocesos en CPU-GPU.
- Análisis de sobrecarga.
- Uso de memoria.

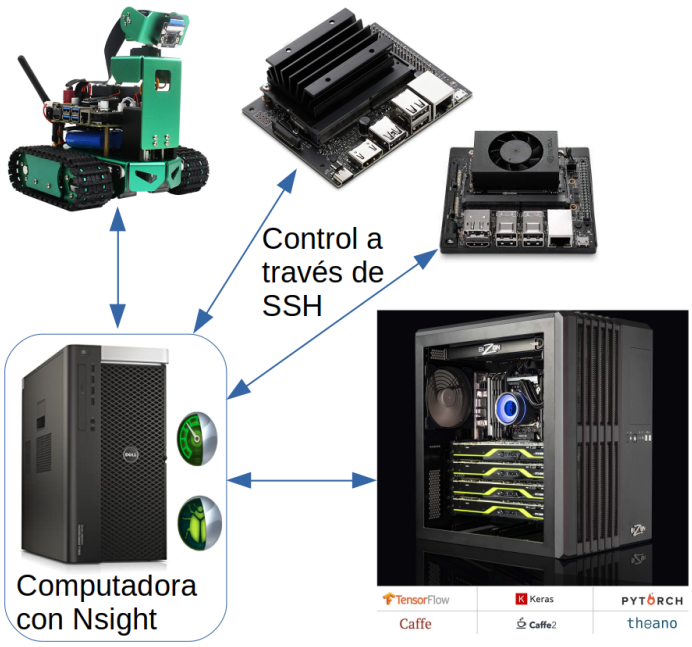

Figura 3.4: Comunicación host y dispositivos.

- Funciona en: contenedores, máquinas virtuales y sistemas con limitaciones de acceso.
- Modos interactivos/programables.
- Se puede analizar: CUDA, OpenACC, OpenGL, cuDNN, cuBLAS, entre otros.
- Soporte para Linux y Windows.

3.7.2. Nsight Compute

Permite visualizar el rendimiento con base en el análisis del código usando GPUs. Es útil después de un análisis inicial con Nsight Systems, una vez identificado lugares críticos para lograr un alto rendimiento. Este software tiene las siguiente características:

- Se puede analizar y depurar la programación con CUDA.
- Se puede visualizar el rendimiento de GPU.
- Se puede identificar cuellos de botella.
- Se puede comparar los resultados, para poder optimizar.
- Instalación: Linux, Windows, MacOSX.
- GPU: Pascal (GP10x), Volta, Turing.

4 Metodología

En las siguientes secciones se describen: la arquitectura red propuesta, los conjuntos de datos audiovisuales que se utilizaron, software y hardware necesarios para llevar a cabo el proyecto. Principalmente se analiza la arquitectura de red neuronal paso a paso para analizar las secuencias imágenes de los diferentes videos.

4.1. Metodología propuesta

Se hace un diseño de una arquitectura de red neuronal utilizando solo las imágenes. Como primer paso se obtienen los frames de los videos, se detecta el rostro de donde se extraen los labios, la boca y la nariz. Después se pasan las imágenes por una red neuronal convolucional, se agrega GRU en las capas intermedias, luego se hace la clasificación, así la red produce el texto esperado. En la figura 4.1 se muestra un diagrama con la metodología propuesta.

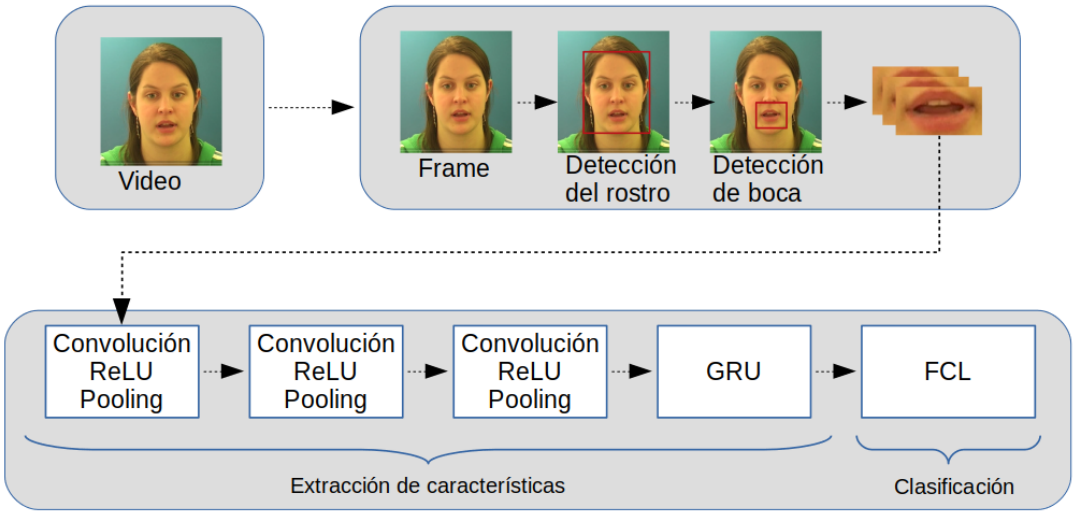

Figura 4.1: Metodología para el reconocimiento de palabras cuando una persona está hablando.

El diseño de la red neuronal convolucional se describe como sigue:

- Convolución: la primera capa se compone de una convolución entre la imagen de entrada de 128×64 con un kernel de 7×7 para obtener en un inicio un mayor número de características.
- ReLU: la función de activación ReLU se utiliza porque es simple y rápida por lo que en distintos artículos lo han documentado.
- Dropout: se utiliza dropout para evitar que el modelo se adapte a las imágenes que se tienen para entrenamiento y se generalice en las pruebas.
- Max Pooling: se obtienen las mejores características de la convolución.
- Convolución: de nuevo se repite la convolución pero ahora con un kernel de 5×5.
- ReLU: se repite la función de activación.
- Dropout: de nuevo dropout.
- Max Pooling: se repite maxpooling.
- Convolución: ahora se cambia a un kernel de 3×3.
- ReLU: se repite la función de activación.
- Dropout: de nuevo dropout.
- Max Pooling: se repite maxpooling.
- GRU: se ha documentado que GRU obtiene mejor rendimiento para el procesamiento de lenguaje natural y con oraciones pequeñas.
- Dropout.
- GRU: se repite una GRU con el proceso parecido a LipNet.
- Dropout.
- FC: Finalmente una capa totalmente conectada.

Al realizar las pruebas se utiliza la función torch.nn.CTCLoss para sumar las probabilidades de las posibles oraciones del objetivo.

Para realizar el análisis de señales se utiliza el modelo descrito en [1] utilizando parte del código descrito en [34] disponible en github para el conjunto de datos de LRS2. El modelo utilizado se llama AudioNet, este tiene diferentes capas basándose en la arquitectura Transformer. Tiene 6 capas de codificación y otras 6 de decodificación. Para poder utilizar el audio se utiliza la transformada de Fourier corta (STFT), de las señales ingresadas, que se transforman linealmente en un solo vector de características de una dimensión de 512 dando 25 vectores por segundo. Entonces calcula las probabilidades de obtener un conjunto de caracteres en cada paso de tiempo.

4.2. Conjunto de datos utilizados

Los conjuntos de datos que se utilizaron son las siguientes:

4.2.1. GRID

GRID [25] es un dataset que contiene información audiovisual. En este conjunto de datos se encuentran videos, audios y textos de personas hablando. Principalmente consta de grabaciones (faciales) de audio (formato wav) y video (formato mpg) de 1,000 oraciones pronunciadas por cada uno de los 34 personas (18 hombres, 16 mujeres), la persona 21 no tiene completa la información, por lo tanto, solo se utilizaron 33 datos audiovisuales. El dataset está disponible gratuitamente para uso de investigación. En [16] se describen las divisiones de datos, dos personas masculinas (1 y 2) y dos femeninas (20 y 22) se utilizaron para la evaluación (3,971 videos). El resto de videos se utilizaron para el entrenamiento (2,8775 videos). Los videos duran 3 segundos con una velocidad de 25 fotogramas por segundo. Al recortar la boca se obtiene un tamaño de 160×80 pixeles en RGB.

En la figura 4.2 se muestran ejemplos de la información obtenida del conjunto de datos GRID.

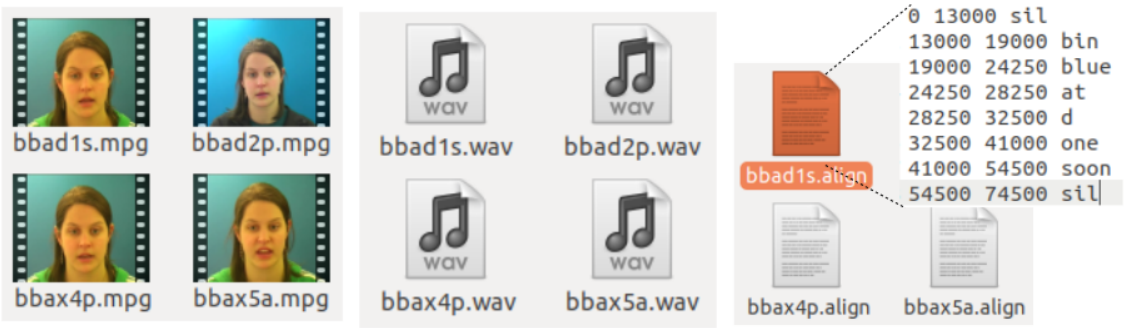

Figura 4.2: Videos en formato mpg, audios en formato wav y texto con diferentes palabras, información obtenida del conjunto de datos GRID.

4.2.2. LRS2

El conjunto de datos de Oxford-BBC Lip Reading Sentences 2 (LRS2) consta de miles de oraciones de diferentes personas de la televisión de la BBC. Cada oración tiene aproximadamente 100 caracteres de longitud. Los conjuntos de entrenamiento, validación y prueba se dividen por la fecha de emisión. Para obtener este conjunto de datos se necesita pedir acceso con un usuario y una contraseña, el proceso se describe en [26].

Las estadísticas del LRS2 se muestran en la tabla 4.1.

La estructura de archivos se muestran en la figura 4.3 con el ejemplo de oración en la figura 4.4.

Tabla 4.1: Conjuntos de datos LRS2.

Datos	Fechas	Expresiones	Palabras	Vocabulario
Pre-Entrenamiento	11/2010 – 06/2016	96,318	2,064,118	41,427
Entrenamiento	11/2010 – 06/2016	45,839	329,180	17,660
Validación	06/2016 – 09/2016	1,082	7,866	1,984
Pruebas	09/2016 – 03/2017	1,243	6,663	1,698

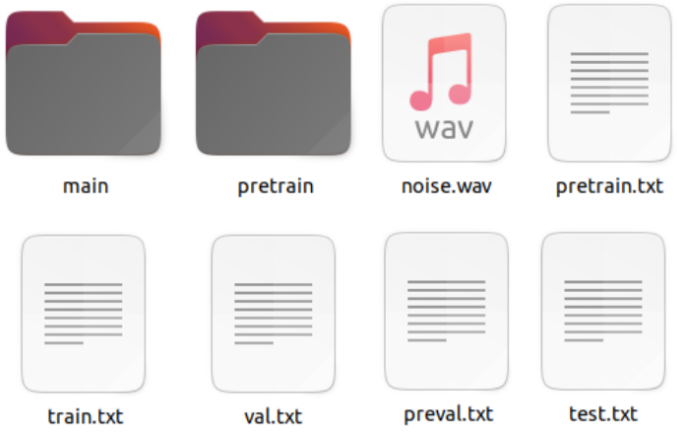

Figura 4.3: Información del conjunto LRS2.

Como se puede ver GRID y LRS2 manejan datos audiovisuales, pero, en diferentes formatos, con diferentes tipos de archivos y estructuras, solo esta parte lleva tiempo para la codificación del pre-procesamiento de datos para que puedan ser ingresados a una red. GRID es el conjunto de datos que fue utilizado para el reporte de resultados de este proyecto ya que se entrenó y validó completo, sin embargo LRS2 fue utilizado para hacer una comparación en lo forma de pre-procesamiento de datos y por el momento fue utilizado para entrenar una red usando solo el audio.

4.3. Estrategia para el entrenamiento

Para poder entrenar el modelo necesitamos aprovechar la cantidad de datos disponibles así como el hardware con que se cuenta. Para realizar el entrenamiento se dividen en las siguientes etapas:

- Se hace un entrenamiento solo con las imágenes.

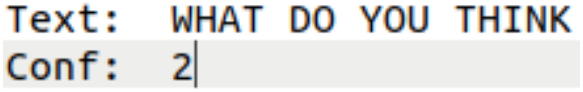

Figura 4.4: Oración en inglés.

- Se hace un entrenamiento solo con el audio.
- Finalmente se entrenan en conjunto.

4.4. Infraestructura utilizada

Para poder llevar a cabo este trabajo de investigación se utilizó la siguiente infraestructura:

4.4.1. Software

Las principales bibliotecas instaladas son las siguientes:

- CUDA 11.6
- NVIDIA cuDNN
- PyTorch 1.12
- OpenCV 4
- Python 3.7
- NVIDIA Nsight Systems 2021.3.2

Las anteriores no son las únicas bibliotecas, también se utilizaron: tensorboardX, numpy, matplotlib, entre otras.

4.4.2. Hardware

Para realizar los entrenamientos se utilizó un servidor con las siguientes características:

- Procesador Intel Core i9-10850K: 10 núcleos, con 20 hilos.
- Memoria RAM de 32 GB.
- GPU NVIDIA GeForce RTX 3080 Ti GDDR6X 12GB.
- Sistema operativo: Ubuntu server 20.04.
- Almacenamiento SSD 1TB NVMe PCIe.

5 Desarrollo experimental

En las siguientes secciones se describen diferentes procesos que se han tenido que realizar para la parte experimental del proyecto. Se describe el pre-procesamiento de datos para el reconocimiento del rostro y la boca de una persona, el entrenamiento del modelo propuesto, el uso y el análisis de rendimiento de la GPU. El código completo del proyecto se encuentra en github en http://github.com/fidelcc/proyectoFinalCAR.

5.1. Uso de la GPU

En este proyecto se utilizó una GPU NVIDIA RTX 3080 Ti para realizar los entrenamientos y pruebas, para poder utilizarla se instaló CUDA con los drivers necesarios. La GPU tiene 12 GB de memoria RAM con 8.6 de capacidad de cómputo. Con el siguiente código se puede verificar si el programa puede utilizar CUDA y la GPU con PyTorch.

```
import torch

if torch.cuda.is_available():
    device_id = torch.cuda.current_device()
    gpu_properties = torch.cuda.get_device_properties(device_id)
    print("%d GPU(s) disponible(s)." % torch.cuda.device_count())
    print("IdGPU %d, (%s)," % (device_id, gpu_properties.name))
    print("Capacidad de computo %d.%d con %.1fGb de memoria.\n" %
    (gpu_properties.major, gpu_properties.minor,
    gpu_properties.total_memory / 1e9))
```

Cuando se entrena una red es conveniente verificar si efectivamente está trabajando la GPU o solo la CPU, así como saber como se utiliza la memoria, entre otras características importantes. En la figura 5.1 se puede ver el rendimiento del servidor al entrenar la red. Para visualizar el comportamiento de la GPU desde la terminal se puede teclear “watch -n 0.5 nvidia-smi”.

5.2. Pre-procesamiento de datos

Los conjuntos de datos que se utilizaron contienen videos, audios y archivos de texto. Entonces primero se tuvo que realizar un análisis para poder trabajar cada uno de los tipos de datos, por lo tanto se hizo un pre-procesamiento de los datos audiovisuales.

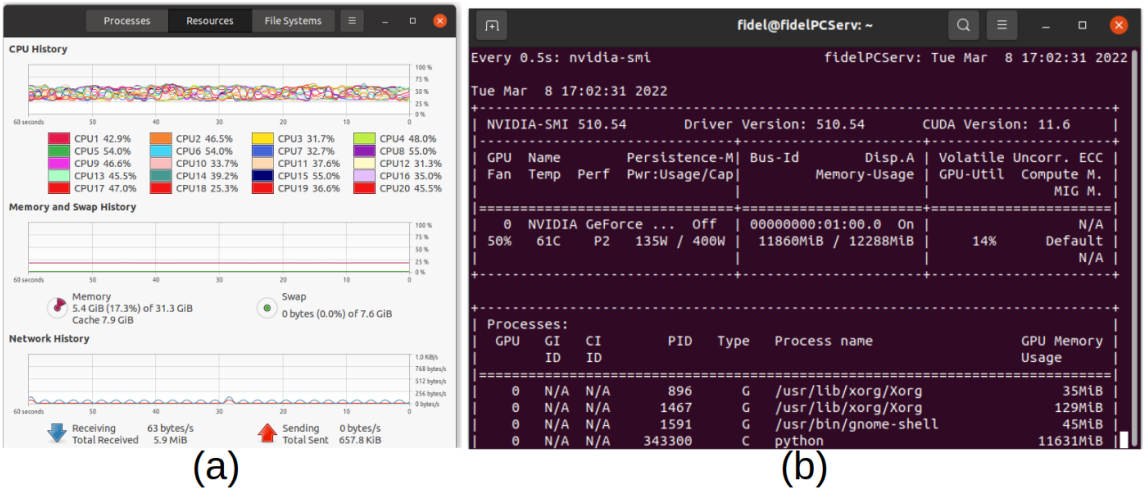

Figura 5.1: Información de procesos, en (a) se puede ver el uso de los cores en la CPU, en (b) se ve el uso de la memoria en la GPU.

5.2.1. Reconocimiento de rostro y boca

Para utilizar los conjuntos de datos primero se tuvo que hacer un pre-procesamiento de los videos. Del video se obtienen los frames y se detecta en donde se encuentra el rostro, después se extrae la boca, cada extracción se guarda como imagen en las carpetas correspondientes, estas imágenes son la información que se utilizan como datos de entrada a la red. Otro método es unir los frames y guardarlos en una imagen como se describe más adelante.

Para el reconocimiento del rostro existen diferentes métodos como son: utilizar archivos haarcascade (haarcascade_frontalface_default.xml) ya generados que contienen valores útiles para detectar el rostro, es rápido pero tiene problemas ya que no siempre detecta el rostro en los frames, otro método es utilizar una biblioteca como facenet_pytorch el cual contiene a la red MTCNN, esté método es más tardado que harcascade sin embargo encuentra los rostros en los videos de los dos dataset utilizados. Se utilizó la segunda opción, pero al codificar se agregó código para utilizar el batch completo de videos de cada carpeta, así, puede ser utilizada la GPU, haciendo más rápido el pre-procesamiento.

Las principales líneas de codificación son las siguientes donde primero se importa MTCNN para el reconocimiento del rostro:

```
from facenet_pytorch import MTCNN
```

Se configura para que sea utilizada la GPU:

```
detector = MTCNN(select_largest=False, post_process=False, device='cuda:0')
```

Se lee el video y se obtienen los frames:

```
cap = cv2.VideoCapture(video)
...
_, frame = cap.retrieve()
```

Se reconocen los rostros y se gurdan las imágenes:

```
faces = detector(frames)
...
for i, frame_faces in enumerate(faces):
  c, w, h = frame_faces.shape
  frame_faces = frame_faces.permute(1, 2, 0).int().numpy()[int(h/2):h, 0:w ]
  ...
  plt.savefig(path_write + '%d.png' % countFrame, bbox_inches='tight',
                  pad_inches = 0)
...
```

Para el pre-procesamiento del conjunto de datos GRID se procesaron los videos, se analizó cada frame detectando el rostro y después la boca como se muestra en la figura 5.2. Se utilizó el código anterior para poder realizar este proceso.

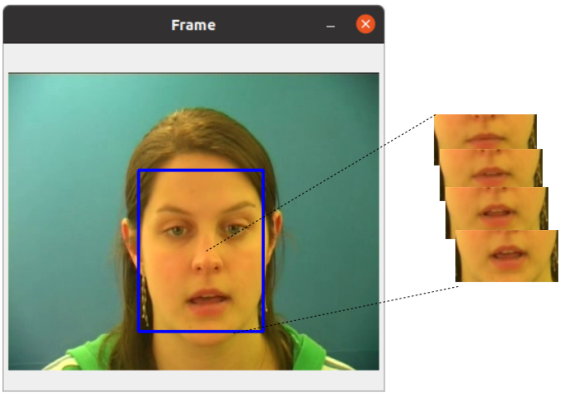

Figura 5.2: Pre-procesamiento con el conjunto de datos GRID.

Para el conjunto de datos LRS2 se tuvo que extraer parte del rostro en el cual se puede analizar las expresiones de cuando una personas está hablando. Así se obtienen archivos para entrenamiento como se muestra en la figura 5.3. Este es otro método que también se puede utilizar, para realizar el entrenamiento con mayor rapidez y evitar estar leyendo cada imagen, las imágenes (ahora son arreglos multi-dimensionales) se guardan en un archivo (con extensión .npy).

Figura 5.3: Pre-procesamiento con el conjunto de datos LRS2.

Para obtener las características del audio y video se basó en el trabajo de [1] el resultado se muestra en la imagen 5.4.

Figura 5.4: Pre-procesamiento del video guardado como una imagen.

5.3. Entrenamiento

Con base en la arquitectura diseñada descrita en la metodología se desarrolla la estructura de la red como se muestra en las siguientes líneas:

```
LipImages(
  (conv1): Conv3d(3, 32, kernel_size=(3, 7, 7), stride=(1, 2, 2),
           padding=(1, 2, 2))
  (pool1): MaxPool3d(kernel_size=(1, 2, 2), stride=(1, 2, 2),
           padding=0, dilation=1, ceil_mode=False)
  (conv2): Conv3d(32, 48, kernel_size=(3, 5, 5), stride=(1, 1, 1),
           padding=(1, 2, 2))
  (pool2): MaxPool3d(kernel_size=(1, 2, 2), stride=(1, 2, 2),
           padding=0, dilation=1, ceil_mode=False)
  (conv3): Conv3d(48, 64, kernel_size=(3, 3, 3), stride=(1, 1, 1),
           padding=(1, 1, 1))
  (pool3): AvgPool3d(kernel_size=(1, 5, 5), stride=(1, 1, 1),
           padding=0)
  (gru1): GRU(2112, 256, bidirectional=True)
  (gru2): GRU(512, 256, bidirectional=True)
  (FC): Linear(in_features=512, out_features=28, bias=True)
  (relu): ReLU(inplace=True)
  (dropout): Dropout(p=0.5, inplace=False)
  (dropout3d): Dropout3d(p=0.5, inplace=False)
)
```

La red propuesta se basa principalmente en LipNet, también en redes como AlexNet. Se utilizó una red pre-entrenada para las primeras pruebas esto para ver si la red sigue aprendiendo a medida que sigue entrenando. Principalmente se modificaron los tamaños de los kernels para obtener mejores características al momento de realizar la convolución. Para entrenar la red se utilizaron los parámetros descritos en la tabla 5.1.

Tabla 5.1: Parámetros de entrenamiento.

Parámetros	**Valor**
Batch	20
Taza de aprendizaje	$2e-4$
Procesadores	20
Épocas	300

5.3.1. Entrenamiento con una red pre-entrenada GRID

Se realizó un primer entrenamiento utilizando una red pre-entrenada de LipNet con el conjunto de datos GRID, en las figuras 5.5 y 5.6 se muestra como fue el comportamiento en el proceso de aprendizaje. El tiempo de entrenamiento de 50 épocas fue de 6 horas con 24 minutos.

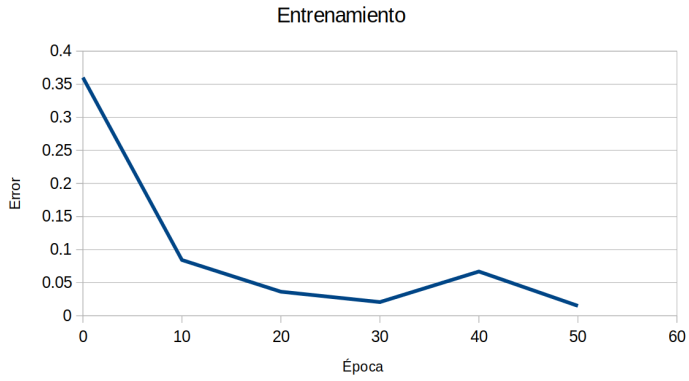

Figura 5.5: Disminución del error en el entrenamiento.

Se puede ver que la red continua aprendiendo pero llega un momento en que el error de pérdida ya no disminuye, aunque, el WER sigue disminuyendo, esto pareciera ser que es por que la red se adapta a las palabras.

5.3.2. Análisis de rendimiento de la GPU con NVIDIA Nsight

Al estar codificando y entrenando las redes neuronales es importante que se analice el uso de la GPU y la CPU en diferentes tiempos, así se puede visualizar cuándo y en qué parte del código no están siendo ocupadas, en la figura 5.7 se muestra el uso de la GPU en diferentes momentos. Se puede ver que prácticamente todo el tiempo está ocupada, sin embargo, este análisis no es suficiente ya que se necesita ejecutar NVIDIA Compute para ver las partes de código que se pueden optimizar, pero es una primer acercamiento a este tipo de software.

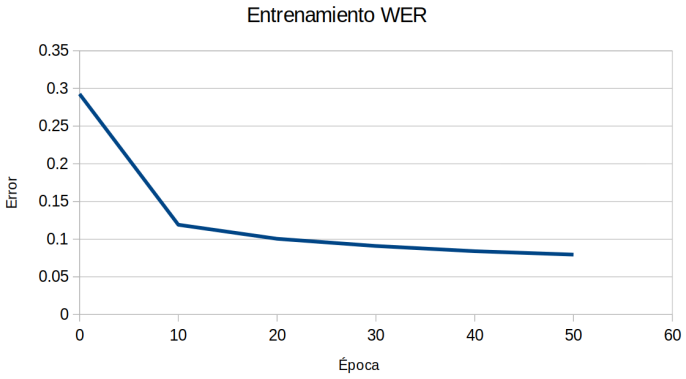

Figura 5.6: Error WER en entrenamiento.

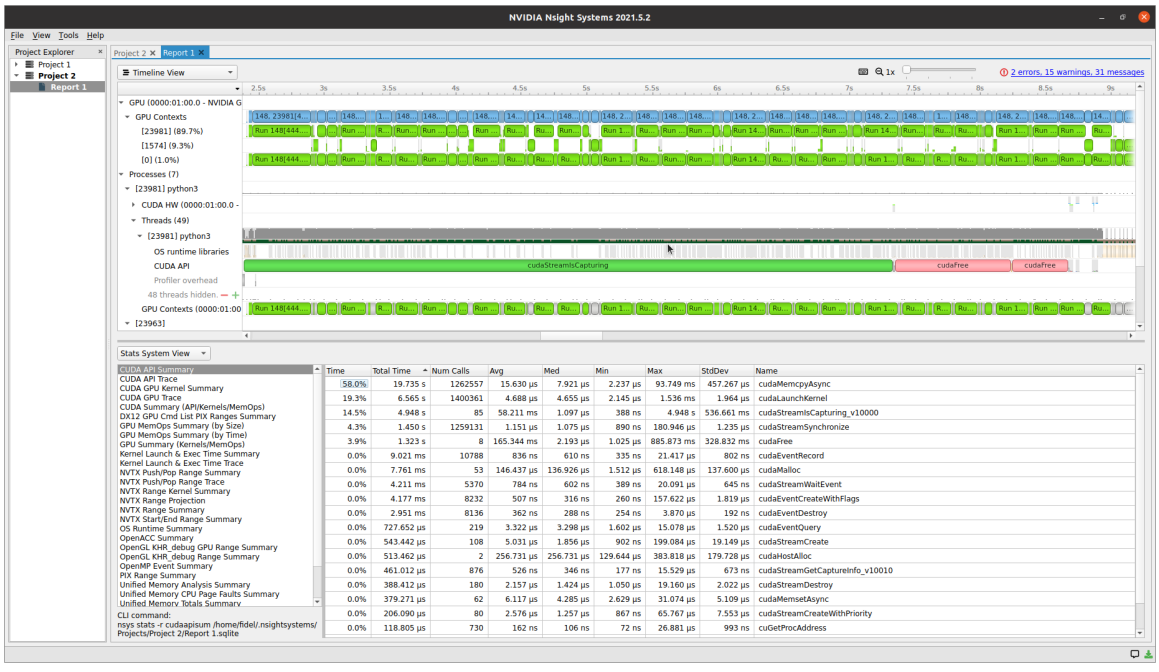

Figura 5.7: Análisis de rendimiento de la GPU.

6 Análisis de Resultados

En las siguientes secciones se hace el análisis de los resultado obtenidos. Primero se describen las pruebas realizadas con una red pre-entrenada, después los resultados con la red propuesta, finalmente se muestra el reconocimiento de oraciones con videos que la red no ha visto.

6.1. Pruebas con una red pre-entrenada

El primer entrenamiento se realizó con LipNet usando una red pre-entrenada, con el resultado de este entrenamiento se pueden inferir los posibles resultados que se pueden alcanzar entrenando una red desde cero. En las pruebas se obtuvo un error de aprendizaje, CER y WER como se muestran en las figuras 6.1, 6.2 y 6.3.

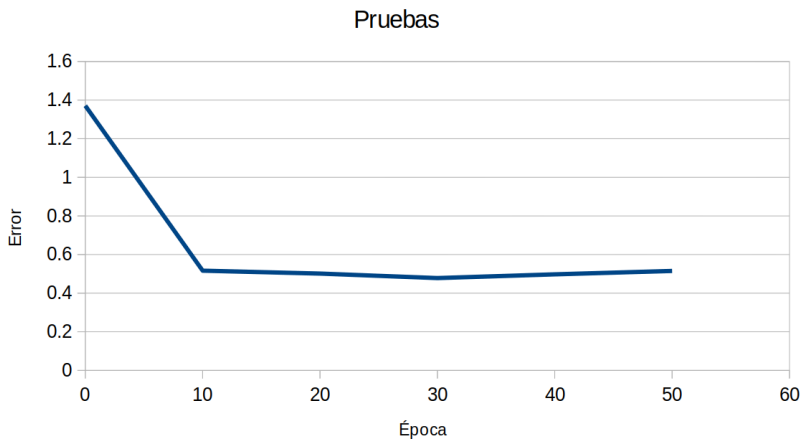

Figura 6.1: Disminución del error en pruebas con la red pre-entrenada.

Se entrenó la red con 50 épocas sin embargo pareciera que a medida que pasan las épocas el error de aprendizaje empieza a ser más grande con los datos de prueba, esto puede ser porque la red se entrena y se adapta a los datos de entrenamiento, a esto se le llama sobre-entrenamiento, entonces tenemos que tener cuidado cuando se entrene la red propuesta. El error en CER se comporta de forma similar al error de aprendizaje, aunque en el error en WER se visualiza poco este fenómeno.

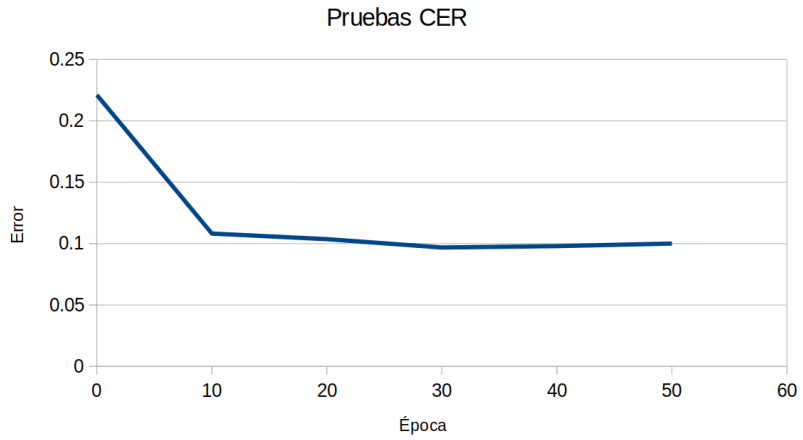

Figura 6.2: Disminución del error CER con la red pre-entrenada.

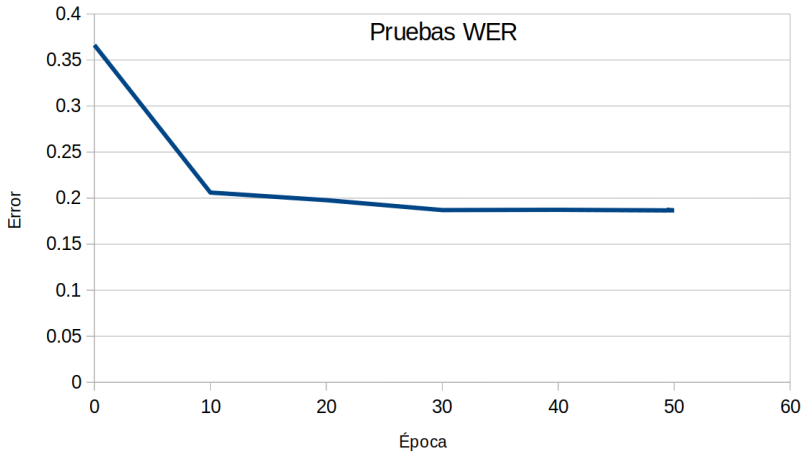

Figura 6.3: Disminución del error WER con la red pre-entrenada.

6.2. Resultados con la red propuesta

Al entrenar la red solo utilizando la CPU con los 20 cores en una época el tiempo fue de 77 minutos, entonces se puede inferir que con 300 épocas el tiempo previsto serían de 385 horas. En cambio utilizando la GPU al entrenar la red propuesta solo con imágenes el tiempo de entrenamiento por época fue alrededor de 6 minutos, el tiempo total fue de 31 horas con 300 épocas. Los resultados del error de aprendizaje se muestran en la figura 6.4, se puede ver que que en las primeras 50 épocas el error de aprendizaje disminuye rápido pero después es mucho más lento, después de 150 épocas prácticamente ya no disminuye el error, en la época 300 el error fue de 0.018.

En la figura 6.5 se puede ver el error en WER, este error sigue adaptándose hasta la época 300, podríamos inferir que seguirá disminuyendo pero ya no es tan rápido entonces tenemos que valorar el tiempo de entrenamiento con los resultados esperados.

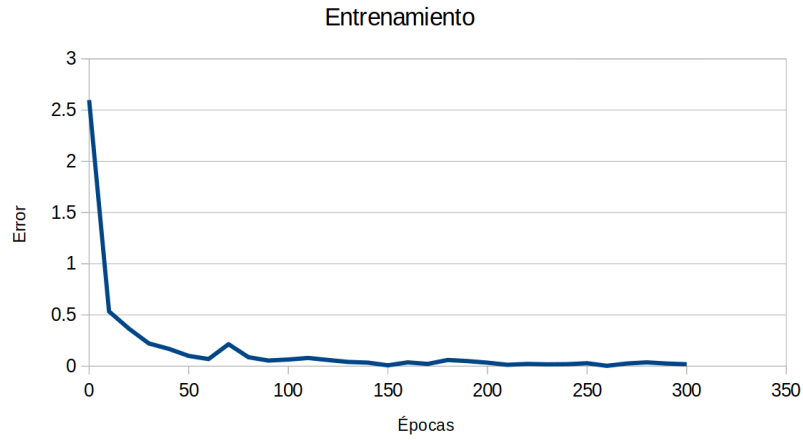

Figura 6.4: Error de aprendizaje en el entrenamiento.

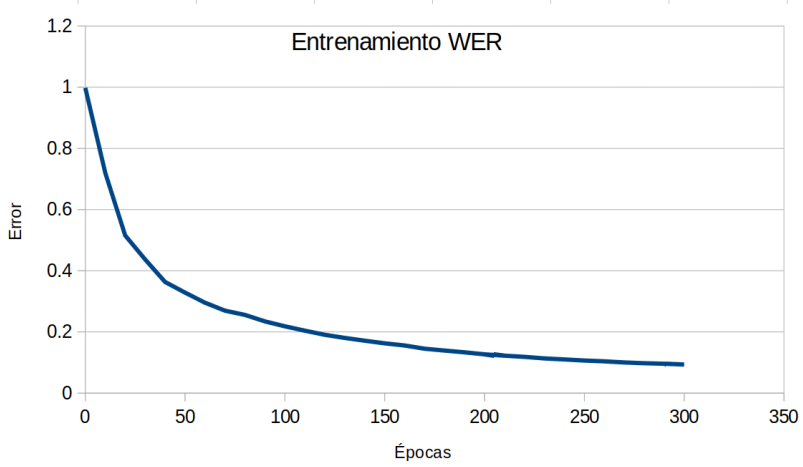

Figura 6.5: Error WER en entrenamiento.

En la figura 6.6 se muestra el error en CER con las datos de prueba. Se puede ver que se comportan parecido al entrenamiento, en las primeras 25 épocas disminuye el error pero después de eso disminuye muy poco e incluso en las últimas iteraciones ya no disminuye. Se puede ver que ya no importa continuar con el entrenamiento, al menos que cambiemos los parámetros de entrenamiento o incluso modificando la red.

El error en WER con los datos de prueba es parecido al error en CER analizado anteriormente como se muestra en la figura 6.7. Prácticamente en las últimas épocas ya no disminuye el error.

6.3. Validación de la información

En la figura 6.8 se muestra un ejemplo con las palabras correctas encontradas al tratar de reconocer las expresiones a partir de los movimientos de los labios con videos

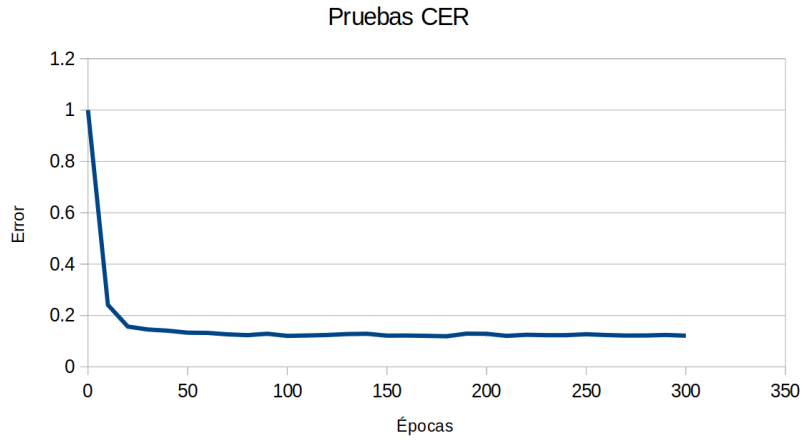

Figura 6.6: Error CER en pruebas.

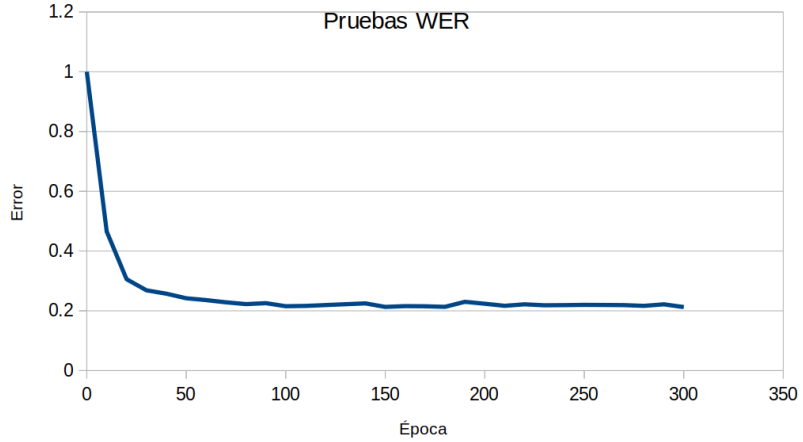

Figura 6.7: Error WER en pruebas.

no vistos previamente con los datos de prueba. Los resultados muestran que la red encuentra varias palabras de forma correcta, sin embargo, en las que no encuentra una igualdad exacta puede cambiar una letra por otra.

6.4. Resultados con audio GRID

También se realizaron entrenamientos utilizando solo señales de audio, los resultados del entrenamiento se muestran en la figura 6.9 con 100 épocas con un tiempo de 46 minutos, con un error de aprendizaje de 0.015 este error implica que la red aprendió la mayor cantidad de palabras. Sin embargo al realizar las pruebas con datos que no ha visto la red se obtuvo 0.14, el CER de entrenamiento fue de 0.005 el WER de 0.01. El CER en pruebas fue de 0.03 y el WER de 0.077. Se ha tenido una mejora significativa con respecto a usar solo las imágenes.

Predicción	Real
BIN RED AT G NINE PLEASE	BIN RED IN Z EIGHT PLEASE
LAY WHITE IN I ONE AGAIN	LAY WHITE IN L ONE AGAIN
PLACE RED AT H ONE AGAIN	PLACE RED AT X ONE AGAIN
PLACE BLUE AT I FIVE SOON	PLACE BLUE AT I FIVE SOON
PLACE BLUE IN K FOUR NOW	BIN GREEN AT N FOUR NOW
PLACE RED BY A SEVEN SOON	PLACE GREEN BY K SEVEN SOON
LAY GREEN WIT V NINE PLEASE	LAY GREEN WITH M EIGHT PLEASE
PLACE WHITE AT G FIVE NOW	PLACE WHITE WITH Q EIGHT NOW
BIN GREED WITH B SIX PLEASE	BIN GREEN WITH B SIX PLEASE
LAY GREEN WITH D THREE NOW	LAY GREEN WITH T THREE AGAIN

Figura 6.8: Análisis de labios con videos, las palabras en color rojo son las que no reconoce de forma correcta.

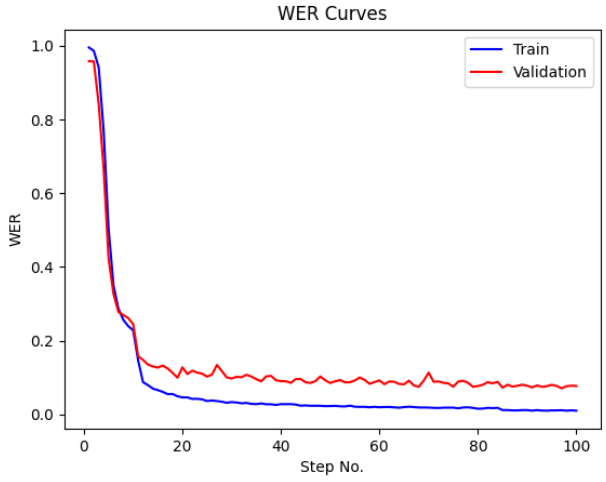

Figura 6.9: Resultados del entrenamiento y validación con audio con 100 épocas.

En la figura 6.10 se muestran ejemplos de oraciones que la red no había visto, sin embargo, los resultados muestran que la red tiene un rendimiento alto a comparación de utilizar las imágenes. Con estos resultados pareciera que la red no tiene problemas de aprender oraciones con un hablante, pero el problema sería cuando hay múltiples hablantes, este problema se podrá experimentar en futuros trabajos.

Como se puede ver en la figura 6.11 existen diferentes tipos de errores en cada red, sin embargo si las redes pudieran corregir el error de la otra, esto podría mejorar el reconocimiento de las oraciones en conjunto, este es un proyecto que se puede continuar trabajando para un futuro proyecto.

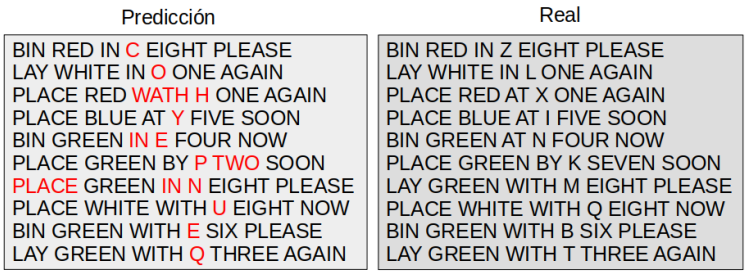

Figura 6.10: Análisis de audios, el color rojo es la letra que no corresponde a la real.

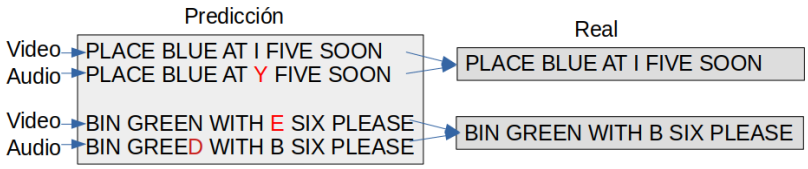

Figura 6.11: Errores en video y audio, al unirse los modelos podrían corregir los errores.

7 Conclusiones y trabajo futuro

7.1. Conclusiones

En este proyecto se ha descrito el proceso para el reconocimiento del habla, utilizando redes neuronales analizando los datos audiovisuales de dos conjuntos de datos (LRS2 y GRID). Se propuso un modelo de red neuronal para el análisis del habla de una persona usando solo las imágenes obtenidas de los videos también se entrenó una red solo con señales de audio. Se pudo comprobar que el modelo propuesto realiza el análisis de rostros de personas que están hablando y los transforma a texto, así se puede analizar lo que dice una persona. No se han utilizado las bases de datos completas, sin embargo con los datos utilizados han sido suficientes para reconocer las secuencias de voz de audio y video transformándolos a caracteres. El mejor modelo compite con lo propuesto en el estado de arte. Las señales ingresan limpias sin embargo se puede modificar para que exista ruido, las pruebas muestran que utilizando las señales de audio da mejor precisión que utilizando las imágenes, esto se ha probado solo con un hablante pero si se cuenta con múltiples hablantes y un ambiente con ruido puede ser un problema.

7.2. Trabajo futuro

Como trabajo futuro se propone entrenar las redes con las bases de datos completas, combinar las redes neuronales y agregar nuevos métodos de clasificación en las últimas capas de las redes. Se obtuvieron buenos resultados entrenando con datos de señales de audio solo con un hablante, sin embargo trabajar con múltiples hablantes puede ser un problema, por la cantidad de ruido en el ambiente, este es otro tema que se tiene pensado trabajar en un siguiente proyecto.

Bibliografía

[1] T. Afouras, J. S. Chung, A. W. Senior, O. Vinyals, and A. Zisserman, "Deep audio-visual speech recognition," *CoRR*, vol. abs/1809.02108, 2018.

[2] J. Chaquet, E. Carmona, and A. Fernández-Caballero, "A survey of video datasets for human action and activity recognition," *Computer Vision and Image Understanding*, vol. 117, pp. 633–659, 06 2013.

[3] L. Xia, G. Chen, X. Xu, J. Cui, and Y. Gao, "Audiovisual speech recognition: A review and forecast," *International Journal of Advanced Robotic Systems*, vol. 17, no. 6, p. 1729881420976082, 2020.

[4] C. Belhan, D. Fikirdanis, O. Cimen, P. Pasinli, Z. Akgun, Z. O. Yayci, and M. Turkan, "Audio-visual speech recognition using 3d convolutional neural networks," in *2021 Innovations in Intelligent Systems and Applications Conference (ASYU)*, pp. 1–5, 2021.

[5] Y. M. Assael, B. Shillingford, S. Whiteson, and N. de Freitas, "Lipnet: Sentence-level lipreading," *CoRR*, vol. abs/1611.01599, 2016.

[6] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–44, 05 2015.

[7] A. Khan, A. Sohail, U. Zahoora, and A. S. Qureshi, "A survey of the recent architectures of deep convolutional neural networks," *CoRR*, vol. abs/1901.06032, 2019.

[8] Pytorch, "Start locally." https://pytorch.org/, 2022. Accessed: 2022-03-07.

[9] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems." https://www.tensorflow.org/, 2015. Accessed: 2022-02-01.

[10] G. Sterpu, C. Saam, and N. Harte, "How to teach dnns to pay attention to the visual modality in speech recognition," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 1052–1064, 2020.

[11] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, pp. 2278–2324, Nov 1998.

[12] G. Huang, Z. Liu, and K. Q. Weinberger, "Densely connected convolutional networks," *CoRR*, vol. abs/1608.06993, 2016.

[13] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[14] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *international Conference on computer vision & Pattern Recognition (CVPR'05)*, vol. 1, pp. 886–893, IEEE Computer Society, 2005.

[15] Y. Aytar, C. Vondrick, and A. Torralba, "Soundnet: Learning sound representations from unlabeled video," *CoRR*, vol. abs/1610.09001, 2016.

[16] Y. M. Assael, B. Shillingford, S. Whiteson, and N. de Freitas, "Lipnet: End-to-end sentence-level lipreading," *GPU Technology Conference*, 2017.

[17] Torch, "Gru." https://pytorch.org/docs/stable/generated/torch.nn.GRU.html, 2022. Accessed: 2022-03-20.

[18] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *CoRR*, vol. abs/1706.03762, 2017.

[19] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," *CoRR*, vol. abs/1810.04805, 2018.

[20] L. Alzubaidi, J. Zhang, A. J. Humaidi, A. Al-dujaili, Y. Duan, O. Al-Shamma, J. Santamaría, M. A. Fadhel, M. Al-Amidie, and L. Farhan, "Review of deep learning: concepts, cnn architectures, challenges, applications, future directions," *Journal of Big Data*, vol. 8, 2021.

[21] S. Fenghour, D. Chen, K. Guo, and P. Xiao, "Lip reading sentences using deep learning with only visual cues," *IEEE Access*, vol. 8, pp. 215516–215530, 2020.

[22] S. Jeon, A. Elsharkawy, and M. S. Kim, "Lipreading architecture based on multiple convolutional neural networks for sentence-level visual speech recognition," *Sensors*, vol. 22, no. 1, 2022.

[23] Q. Huang, Z. Huang, P. Werstein, and M. Purvis, "Gpu as a general purpose computing resource," in *2008 Ninth International Conference on Parallel and Distributed Computing, Applications and Technologies*, pp. 151–158, 2008.

[24] H. McGurk and J. MacDonald, "Hearing lips and seeing voices," *Nature*, vol. 264, pp. 746–748, 1976.

[25] S. C. Jon Barker, Martin Cooke and X. Shao, "The GRID audiovisual sentence corpus." http://spandh.dcs.shef.ac.uk/gridcorpus/, 2013. Accessed: 2022-02-15.

[26] Visual Geometry Group, "The Oxford-BBC Lip Reading Sentences 2 (LRS2) Dataset." https://www.robots.ox.ac.uk/~vgg/data/lip_reading/lrs2.html, 2016. Accessed: 2022-02-04.

[27] R. Gao and K. Grauman, "Visualvoice: Audio-visual speech separation with cross-modal consistency," *CoRR*, vol. abs/2101.03149, 2021.

[28] R. Gao and K. Grauman, "Visualvoice: Audio-visual speech separation with cross-modal consistency." https://vision.cs.utexas.edu/projects/VisualVoice/, 2021. Accessed: 2022-02-03.

[29] J. S. Chung, A. W. Senior, O. Vinyals, and A. Zisserman, "Lip reading sentences in the wild," *CoRR*, vol. abs/1611.05358, 2016.

[30] J. Hong, M. Kim, S. J. Park, and Y. M. Ro, "Speech reconstruction with reminiscent sound via visual voice memory," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, pp. 1–1, 2021.

[31] Visual Geometry Group, "The Oxford-BBC Lip Reading in the Wild (LRW) Dataset." https://www.robots.ox.ac.uk/~vgg/data/lip_reading/lrw1.html, 2016. Accessed: 2022-02-04.

[32] Visual Geometry Group, "Lip Reading Sentences 3 (LRS3) Dataset." https://www.robots.ox.ac.uk/~vgg/data/lip_reading/lrs3.html, 2016. Accessed: 2022-02-04.

[33] Multimedia Systems Department of Gdansk University of Technology, "Modality corpus." http://modality-corpus.org/, 2022. Accessed: 2022-03-20.

[34] S. Shah, "Deep audio-visual speech recognition." https://github.com/lordmartian/deep_avsr. Accessed: 2022-03-03.