UNIVERSIDAD POLITÉCNICA SALESIANA

SEDE CUENCA

CARRERA DE INGENIERÍA ELECTRÓNICA

Trabajo de titulación previo
a la obtención del título
de Ingeniero Electrónico

PROYECTO TÉCNICO CON ENFOQUE INVESTIGATIVO:

"DESARROLLO DE UN SISTEMA PROTOTIPO DE GENERACIÓN DE ROSTROS A PARTIR DE SEÑALES DE VOZ UTILIZANDO REDES GENERATIVAS ADVERSARIAS"

AUTORES:

EDISON ALEXANDER ZUMBA NARVÁEZ

FERNANDO PATRICIO ZUMBA NARVÁEZ

TUTOR:

ING. CHRISTIAN RAÚL SALAMEA PALACIOS, Ph.D.

CUENCA - ECUADOR

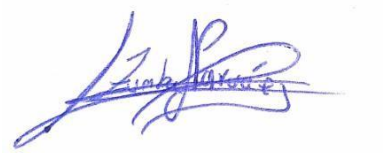
2022

CESIÓN DE DERECHOS DE AUTOR

Nosotros, Edison Alexander Zumba Narváez con documento de identificación N° 0705725364 y Fernando Patricio Zumba Narváez con documento de identificación N° 0705725372, manifestamos nuestra voluntad y cedemos a la UniversidadPolitécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de quesomos autores del trabajo de titulación: **"DESARROLLO DE UN SISTEMA PROTOTIPO DE GENERACIÓN DE ROSTROS A PARTIR DE SEÑÁLES DE VOZ UTILIZANDO REDES GENERATIVAS ADVERSARIAS"**, mismo que ha sido desarrollado para optar por el título de: *Ingeniero Electrónico*, en la Universidad Politécnica Salesiana, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En aplicación a lo determinado en la Ley de Propiedad Intelectual, en nuestra condición de autores nos reservamos los derechos morales de la obra antes citada. En concordancia, suscribimos este documento en el momento que hacemos entrega del trabajo final en formato digital a la Biblioteca de la Universidad Politécnica Salesiana.

Cuenca, enero de 2022.



Edison Alexander Zumba Narváez

C.I. 0705725364

Fernando Patricio Zumba Narváez

C.I. 0705725372

CERTIFICACIÓN

Yo, declaro que bajo mi tutoría fue desarrollado el trabajo de titulación: **"DESARROLLO DE UN SISTEMA PROTOTIPO DE GENERACIÓN DE ROSTROS A PARTIR DE SEÑALES DE VOZ UTILIZANDO REDES GENERATIVAS ADVERSARIAS"**, realizado por Edison Alexander Zumba Narváez y Fernando Patricio Zumba Narváez, obteniendo el *Proyecto Técnico con enfoque investigativo* que cumple con todos los requisitos estipulados por la Universidad Politécnica Salesiana.

Cuenca, enero de 2022.

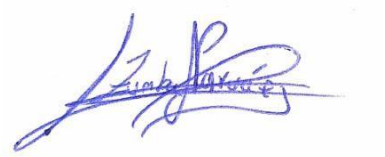
Ing. Christian Raúl Salamea Palacios, Ph.D.

C.I. 0102537180

DECLARATORIA DE RESPONSABILIDAD

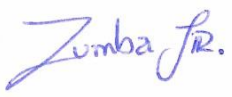
Nosotros, Edison Alexander Zumba Narváez con documento de identificación N° 0705725364 y Fernando Patricio Zumba Narváez con documento de identificación N° 0705725372, autores del trabajo de titulación: **"DESARROLLO DE UN SISTEMA PROTOTIPO DE GENERACIÓN DE ROSTROS A PARTIR DE SEÑALES DE VOZ UTILIZANDO REDES GENERATIVAS ADVERSARIAS",** certificamos que el total, contenido del *Proyecto Técnico con enfoque investigativo,* es de nuestra exclusiva responsabilidad y autoría.

Cuenca, enero de 2022.



Edison Alexander Zumba Narváez

C.I. 0705725364



Fernando Patricio Zumba Narváez

C.I. 0705725372

ÍNDICE

AGRADECIMIENTOS

El agradecimiento de este trabajo va dirigido primeramente a mis padres por todo el esfuerzo y sacrifico que han hecho por mí todos estos años dándome ese apoyo incondicional estando todos los días pendiente de mi apoyándome en cada meta que estoy cumpliendo y que estaré por cumplir. También quiero agradecer al Dr. Christian Salamea que gracias a su conocimiento y de su ayuda se pudo concluir con éxitos este trabajo. Finalmente quiero agradecer a todas mis amistades que me brindaron ese apoyo moral dándome buenos consejos y sobre todo poder compartir con ellos esta etapa de mi vida.

Edison Zumba Narvaéz.

Agradezco a mis padres y familia por haberme apoyado y confiado en mí en todo el momento de mi vida universitaria, son mi principal motivación. A mis amigos que siempre supieron brindarme su amistad y darme buenos consejos. Agradezco también, al Dr. Christian Salamea quien con sus conocimientos, paciencia y amistad supo brindarme la orientación y motivación para realizar este proyecto.

Fernando Zumba Narváez.

.

DEDICATORIA

Este trabajo va dedicado a mis padres Patricio y Ximena, a mi hermana Andrea, por todo el apoyo que me han dado durante cada etapa de mi vida siendo mi pilar fundamental en cada paso que doy y a todos mis amigos que estuvieron siempre y me brindaron todo su apoyo moral antes y durante la carrera.

Edison Zumba Narváez.

Este proyecto se lo dedico a mis padres Patricio y Ximena, me brindaron todo su amor, confianza y sobre todo comprensión durante toda mi formación académica. También va dedicado a mi hermano Edison por estos años de estudio universitario juntos, a mi hermana Andrea quien siempre estuvo pendiente de nosotros dos. Para Germán un gran amigo que forma parte de nuestra familia.

Fernando Zumba Narváez.

GLOSARIO

ASR: Sistema automático de reconocimiento.
AI: Inteligencia Artificial - Artificial Intelligence.
GAN: Red Generativa Adversaria - Adversary Generative Network
ASR: Reconocimiento Automático de Hablantes – Automatic Speaker Recognition
FC: Fully connected. Neuronas de una capa que están conectadas todas con las neuronas de la siguiente capa.
FSR: Reconocimiento Forense de Hablantes – Forensic Speaker Recognitio
DL: Aprendizaje Profundo - Deep Learning.
CPU: Unidad Central de Procesamiento - Central Processing Unit
GPU: Unidad de procesamiento gráfico - Graphics Processing Unit
TPU: Unidad de procesamiento de tensorial - Processing Unit
Vlog: Videos donde se explica o narra temas variados, principalmente se encuentran en la plataforma YouTube

RESUMEN

Actualmente, el desarrollo de tecnologías de aprendizaje automático y profundo han mejorado bastante y su uso en aplicaciones en las industrias, investigación, domótica, comercio, y en varios temas de detección de objetos en videos o en imágenes es aprovechado cada vez más. Una tecnología en concreto llamada red generativa adversaria (GAN) es la que usan en la mayoría de los casos para la reconstrucción o generación de imágenes, audio o señales. Esta característica es aprovechada en este trabajo para poder generar imágenes a partir de un audio de voz para crear un sistema que pueda servir en el campo de estudio de la acústica forense. La acústica forense es un tema de seguridad donde se trabajan con audios para identificar a un criminal con solo grabaciones de audio o de videos, para así clasificar audios y comparar con bases de datos que tienen las investigaciones criminalísticas. Existe la posibilidad de que el dueño de la voz no aparezca en evidencias de fotos o en videos donde no aparece, entonces complica a las investigaciones y no pueden identificarlo.

En este trabajo se propone una red generativa adversaria que genera la imagen de un rostro mediante un fragmento de audio de la voz del hablante. Se crea una base de datos donde contiene imágenes de rostros y el audio de personas que aparecen en videos de la plataforma YouTube, para entrenar los modelos que se proponen. Los videos seleccionados para la base de datos poseen una calidad de video y audio alta. El objetivo es generar imágenes de rostros con solo el audio de alguien que nunca se entrenó la red, es decir, generar un rostro parecido al dueño de la voz. Los resultados obtenidos de la experimentación de las modificaciones son valores cercanos a la métrica PSNR la que indica que resultados pueden considerarse como semejantes a un rostro humano. Adicional a ello, se diseña un Autoencoder con el mismo objetivo de la GAN, permitiendo comparar la eficiencia del sistema prototipo propuesto.

INTRODUCCIÓN

Las redes generativas adversarias (GANs) son redes desarrolladas para el aprendizaje profundo donde su campo se alinea a la reconstrucción o generaciones de imágenes, audio o señales de cualquier tipo. Las GANs enfocadas a la generación de imágenes aporta un campo muy importante en el mundo actual, un caso que se puede asociar es al campo de la seguridad como puede ser con la acústica forense. La acústica forense es un campo donde se tiene a científicos y lingüistas trabajando para mejorar técnicas criminalísticas del tratamiento con audios que pertenezcan principalmente a personas. Estos audios son tratados por técnicas donde se realiza la clasificación para identificar personas durante una investigación de un delito. Con el audio ya tratado proceden a verificar con base de datos de imágenes o videos para intentar identificar al acusado, extorsionador por llamadas, videos donde su rostro no se ve. Hay veces donde en la base de datos de imágenes o videos no existe ningún rastro del supuesto delincuente y no se pueda resolver el caso. En estos casos se puede el delincuente usa como protección el anonimato y siga causando problemas a la sociedad.

Con la mejora de tecnologías del aprendizaje profundo como un ejemplo es la GAN, se puede emplear este tipo de tecnología para entrenar una red de este tipo y poder generar imágenes de rostros de personas con solo un fragmento de audio. Se puede intentar conseguir por lo menos algún audio vía redes sociales, llamadas telefónicas o videos y audio que se pueden conseguir con teléfonos inteligentes, tablets, Ipads, Alexa o dispositivo inteligente que existe en la actualidad para extraer el audio donde se escucha hablar a la persona que quiere cometer un delito. Cabe aclarar que este proyecto está limitado a audios de voz descargados de videos de la plataforma Youtube donde el video aparece una persona a la vez, pero aumentando la base de datos se conseguiría aprender más características y poder generar rostros más semejantes a una persona que nunca se ha visto y solo se tenga como evidencia un audio.

ANTECEDENTES DEL PROBLEMA DE ESTUDIO

En la actualidad se ha incrementado el número de delitos telefónicos a nivel mundial, en los cuales mediante un buen desarrollo de ingeniería social obtienen los datos necesarios de sus víctimas y proceden con su extorsión o estafa. En el trabajo [1] indica que países como Rusia en el 2018 registran un incremento de denuncias de transacciones monetarias fraudulentas que se hacen mediante llamadas telefónicas. Para resolver este problema ellos plantearon usar sistemas inteligentes los cuales toman los audios de las llamadas telefónicas e identifiquen si el audio contiene características vocales que presentan en la mayoría de los delitos de este tipo, los resultados obtenidos de la detección fueron de 43.82% de eficiencia de este sistema, este resultado es por la razón de que los audios usados para el entrenamiento fueron audios de base de datos públicas y que no contienen audios de extorsiones o acosos telefónicos.

Pasando a analizar los datos en Latinoamérica, tenemos a México como el décimo país registrado en 2020 en tener ataques cibernéticos, en donde se encuentra las llamadas telefónicas vía aplicaciones de mensajería instantánea o redes sociales, las cuales son ejecutadas para fraude, robo de identidad, robo de datos, entre otros [1]. En Ecuador en el año 2019 la Fiscalía presenta un boletín a la ciudadanía indicando que el número de estafas incrementa y la mayoría se lo comete mediante hojas volantes, llamadas telefónicas, mensajes de texto, redes sociales y mensajerías instantáneas como WhatsApp [2]. El laboratorio de Criminalística y Ciencias Forenses en la ciudad de Quito se encarga de hacer investigaciones de diferente índole donde también se encuentra el análisis de audio y video, cuyo proceso es de aumentar su base de datos para hacer las comparaciones e identificar las características biométricas y poder identificar al delincuente [3].

La acústica forense dentro de la investigación criminalística busca la comprobación e identificación de los delitos cometidos por personas que realizan actos criminales como la de secuestro, robo, extorsión, evasión de impuestos como también actividades terroristas, en donde los oyentes tienen la capacidad de extraer información de un audio de voz de una persona lo cual según [4] facilita características físicas (sexo, edad, etc.), características psicológicas (estado de ánimo y actitudes) y características sociales (educación o estatus social). Todo este procedimiento se lo realiza luego de que hayan sucedido los hechos criminales mencionados anteriormente. En Ecuador existen casos de extorsión vía telefónica los cuales en su mayoría quedan sin resolver, debido a que el país no cuenta con un sistema que reconozca al extorsionador y no se pueda dar con el paradero o algún indicio de quién y cómo es esa persona. Existen ciertos casos en los que, si bien se concreta el paradero del extorsionador, se requiere un tiempo para buscar y analizar evidencias o patrones por parte de las instituciones de seguridad en el Ecuador.

Existen trabajos en los cuales se generan rostros con diferentes técnicas, pero trabajos o estudios enfocados con inteligencia artificial (AI) a resolver este problema es muy poco investigado o tratado. Los trabajos relacionados a la generación de rostros por medio de técnicas de inteligencias artificial usan Redes Generativas Adversarias (GANs) y los modelos que se usan son el VGG-Face, SEGAN, BEGAN [5]–[8]. Además, las bases de datos que usaron fueron: AVSpeech, CelebA y Voxceleb, base de datos que contienen videos, imágenes de personas famosas mas no de personas que pudieren haber cometido actos criminales como, por ejemplo, extorsiones realizadas por vía telefónica.

JUSTIFICACIÓN (IMPORTANCIA Y ALCANCES)

Basado en todo lo expuesto en los párrafos anteriores, este trabajo está enfocado a desarrollar un sistema prototipo inteligente de generación del rostro de una persona en base de una señal de voz, que permita, establecer una línea base de investigación en el área de la "Acústica Forense" y exponer los primeros resultados de un estudio que involucrará rostros de Youtubers más conocidos en Internet. El trabajo está restringido a señales de voz obtenidas de la plataforma de YouTube a partir de bases de datos diseñadas para el efecto. Para trabajos futuros se espera poder extender las evaluaciones para pruebas con audio de cualquier persona, esto permitirá que este proyecto se pueda aplicar en cualquier sistema integral de seguridad.

OBJETIVOS

OBJETIVO GENERAL

- Desarrollar un sistema prototipo de generación de rostros a partir de señales de voz utilizando Redes Generativas Adversarias

OBJETIVOS ESPECÍFICOS

- Crear una base de datos de imágenes de rostros de personas con correspondencia de audio a partir de bases de datos ya existente en la red para el entrenamiento de una red generativa adversaria.
- Diseñar una red generativa adversaria para la construcción de un rostro a partir de una señal de ruido como entrada obteniendo una configuración de base eficiente en la generación de rostros.
- Definir la métrica más adecuada para evaluar el entrenamiento y funcionamiento de la red en base a la distancia entre puntos en una imagen.
- Adaptar la configuración de la red introduciendo como vector de entrada al embedding neuronal de la voz para la generación de rostros.
- Evaluar la eficiencia del sistema final comparándolo con la configuración de un autoencoder.

CAPÍTULO 1

1. FUNDAMENTACIÓN TEÓRICA O ESTADO DEL ARTE

En este punto se realiza una descripción de los sistemas utilizados actualmente de investigación en acústica forense, las redes neuronales basadas en la implementación de reconstrucción de rostros humanos, y el entorno donde se estará trabajando para la implementación del sistema de generación de imágenes.

1.1. SISTEMAS DE ACÚSTICA FORENSE

Los sistemas actuales de identificación (ASR) de criminales que cometen secuestro, robos, extorciones, evasión de impuestos como también actividades terroristas, son sistemas de tipo físico en donde científicos y lingüistas forenses desarrollan sistemas basado en algoritmos para el reconocimiento forense de hablantes que actualmente se implementan en muchas empresas, aplicaciones de comercio electrónico, análisis forense y aplicación de la ley [9]. Para los juicios penales existen sistemas que se mejoran cada vez para verificar si un audio puede ser admitido y sea válido, pero aún sigue siendo un sistema que aún no tiene un grado de efectividad bueno y por tal motivo estos audios admitidos ingresan y se prueban con el resto de evidencia para que el Juez evalúe y pueda dictaminar un juicio. Los audios aun no sirven como una fuerte evidencia debido a que pueden ser modificados o alterados [10].

1.2. SISTEMAS DE GENERACIÓN DE IMÁGENES CON APRENDIZAJE PROFUNDO CON SEÑALES DE AUDIO

El audio de voz es una señal biométrica que contiene información sobre la identidad, género y estado emocional del hablante [6]. Existen varios trabajos enfocados a la generación de imágenes de rostros utilizando aprendizaje profundo. Donde indican diferentes modos de entrenar un modelo, algunos trabajos toman como entrada videos de Dataset Públicos donde estos son ingresados segmentos de audio del video y el objetivo es reproducir una imagen del rostro del hablante y son condicionadas con los videos de entrada. Otro trabajo donde crean una base de datos

de imágenes de rostros de Youtubers, para el entrenamiento del modelo que proponen que mediante un segmento de audio ingrese a la entrada de una red generadora de una GAN y al mismo tiempo este condicionada la red discriminadora con el audio para que la generación de imágenes genere buenos resultados. En ambos casos se entrenan los modelos con aprendizaje auto-supervisado[6], [11]. En [8] proponen un modelo novedoso el cual plantea reconstruir la imagen de un rostro humano mediante su voz. Es trabajo que responde a la pregunta ¿Podemos imaginar cómo es la identidad de una persona (persona desconocida) con el mayor número de detalles de su rostro con solo tener un audio de voz? Entonces, propusieron la creación de una GAN con un marco computacional simple. Aunque sus resultados son muy buenos, presentan un problema. Al momento de entrenar con videos de base de datos de videos públicas en el caso de [6] que crea su propia base de datos basados en videos de Youtube, son miles de horas de personas que hablan en videos y sus movimientos son varios, poseen vestimenta que cubre su rostro, estilos de cabello que tapan su rostro, bigote, gafas, forma de orejas, uso de maquillaje, accesorios como pircings faciales, todo esto mencionado hace que sus resultados varíen en sus métricas al momento de evaluar para generar un rostro. El sistema más utilizado y que trabajan estos proyectos para la generación de rostros son las redes generativas adversarias (GAN).

1.3. REDES GENERATIVAS ADVERSARIAS

La idea básica de cómo trabaja una GAN es explicando con la siguiente analogía. Se puede comparar con un juego entre dos jugadores, el primer jugador se lo conoce como **generador** y el segundo como **discriminador**. El discriminador examina ejemplos o muestras para determinar si son verdaderas o falsas. El discriminador "aprende" usando técnicas de aprendizaje supervisado, dividiendo las entradas en dos clases: verdaderas o falsas. En cambio, el generador es entrenado para que genere las muestras hasta que el discriminador sea engañado y los clasifique como verdaderos [12]. Las GANs pertenecen a la familia de modelos de aprendizaje profundo. En términos técnicos, la red generadora tiene como trabajo generar muestras a partir de distribuciones tales que sean suficientemente cercanas a las muestras reales con el único objetivo de engañar a la red discriminadora. El trabajo de la red discriminadora es distinguir las muestras generadas (falsificaciones) de las muestras auténticas. En la Figura 1 se muestra la arquitectura de un modelo GAN [13].

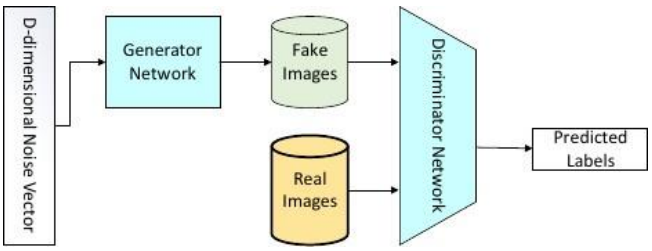


Figura 1 Esquema básico de una GAN.
Fuente: [13]

1.3.1. CAPAS CONVOLUCIONALES

En [14] explica que las capas convolucionales son frecuentemente usadas en problemas de visión computacional, donde la computadora puede trabajar con una imagen y extrae características y patrones mediante "filtros" o "kernels" de la matriz que representa a la imagen. Se usan también en procesamiento de video. Trabaja en dos tipos llamados: Upsampling y Downsampling. La diferencia entre estos dos es la siguiente: Downsampling intenta comprimir la entrada y Upsampling intenta expandir.

- **Capa convolucional**

La capa convolucional su trabajo es reducir la entrada, ya que el resultado de la aplicación es menos filas y menos columnas de la entrada original. Para controlar el grado de comprensión expansión de la entrada, se pueden aplicar otras herramientas como padding, strides y dilation.

- Para incrementar las dimensiones de la salida se usa por lo general un relleno (padding). Al aplicar esto su resultado es rellenar con ceros los bordes, al hacer esto no afecta al producto punto, pero da más espacio para que el filtro se deslice por la imagen. Ver Figura 2.

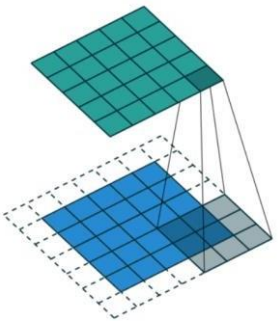


Figura 2 Representación de aplicación de padding con un valor de 1. Los bordes punteados son los espacios rellenados con 0.
Fuente: [14]

- Los pasos (strides) controlan el número de unidades que el filtro se desliza a la vez. Esta función viene por defecto con valor 1, pero se puede aumentar el valor para comprimir aún más la salida. Ver Figura 3.

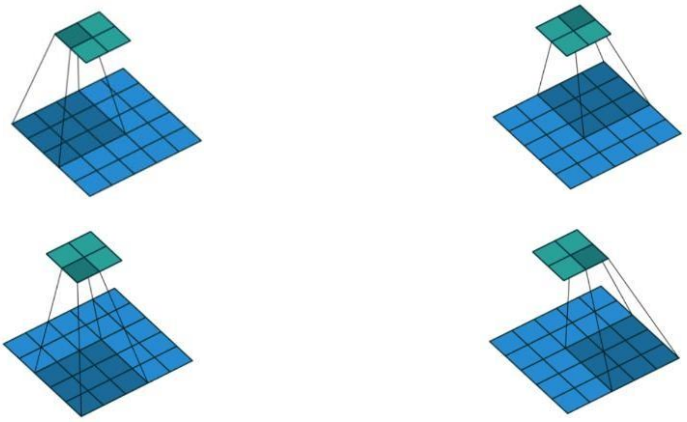


Figura 3 Representación de aplicación de stride de 2x2. La secuencia es de izquierda a derecha y de arriba hacia abajo.
Fuente:[14]

- **Capa convolucional transpuesta**

La capa convolucional transpuesta a diferencia que la capa convolucional, trabaja al revés, es decir, aumenta la entrada. Este tipo de funciones se usan generalmente en Autoencoders y GANs o algún tipo de red que requiera

reconstrucción de imágenes. Al momento de aplicar esta función lo que hace es que las dimensiones de entrada y salida cambien. Es decir, que la salida será mayor que la entrada. En la Figura 4 se puede observar la aplicación de esta función. Se pueden aplicar las mismas funciones que la convolucional normal como es padding, stride y dilatation solamente que función “al revés”.

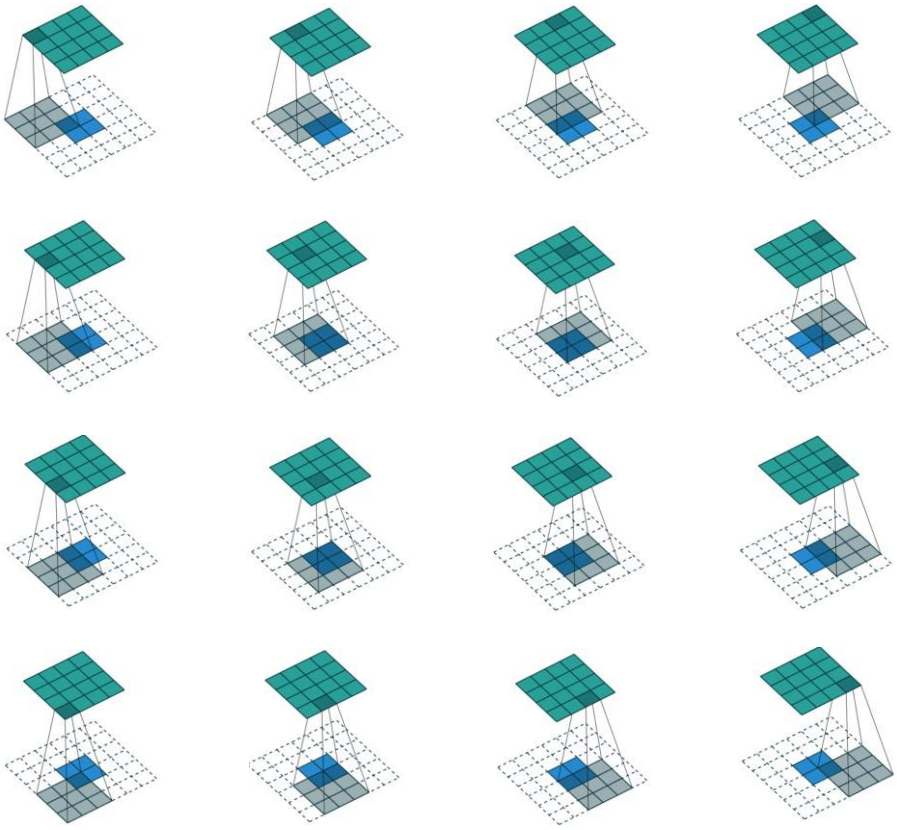


Figura 4 Representación de aplicación de una capa convolucional transpuesta. La secuencia es de izquierda a derecha y de arriba hacia abajo.
Fuente:[14]

- Cuando se agrega un padding a la capa convolucional transpuesta, se remueve el padding de la entrada y la salida resultante es más pequeña.

- Cuando se agrega la función stride, produce el mismo efecto explicado en la capa convolucional normal, pero en este caso afecta a la entrada y no a la salida.

1.3.2. FUNCIÓN DE ACTIVACIÓN RELU Y LRELU

La función de activación ReLU es la función comúnmente utilizada en estos últimos años como activación en capas ocultas, los resultados indican que ReLU conduce a un mayor y consistente gradiente con el que ayuda al aprendizaje basado en gradientes [15]. En las salidas de todas las capas de la red generadora de una GAN se usa este tipo de activación, exceptuando la capa de salida [16]. La grafica de la función se la puede observar en la Figura 5.

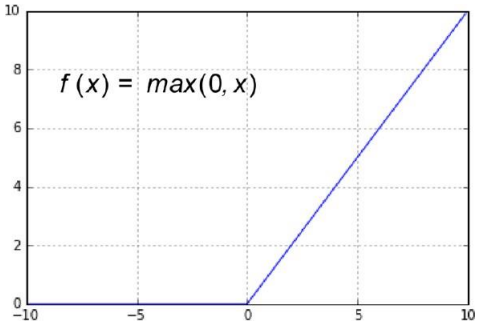


Figura 5 Función de activación ReLU.
Fuente: [15]

La función de activación Leaky ReLU (LReLU) es una mejora de ReLU, esta intenta eliminar el problema de la "neurona moribunda" que aparece en ReLU con una pendiente pequeña positiva que no conduce a un problema de gradiente cero. Entonces, durante la retro-propagación, las neuronas en la región negativa también se toman en consideración [17]. Su expresión matemática es la ecuación (1) y su grafica se observa en la Figura 6. Este tipo de activación es usado en una red discriminadora de una GAN en todas sus capas [16].

$$f(x) = \begin{cases} \alpha x & para\ x < 0 \\ x & para\ x \geq 0 \end{cases};\ \alpha\ es\ una\ constante\ pequeña \qquad (1)$$

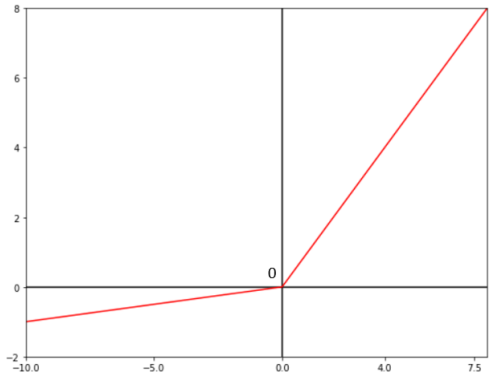


Figura 6 Función de activación LReLU.
Fuente:[17]

1.3.3. FUNCIÓN DE ACTIVACIÓN TANH

La función tangente hiperbólica es la función usada para las salidas de las capas ocultas de una red neuronal [15]. En el caso de una GAN, específicamente para la salida de la red generadora se utiliza esta activación [16].

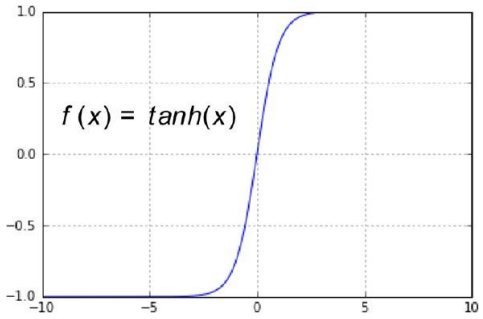


Figura 7 Función de activación tangente hiperbólica.
Fuente: [15]

1.3.4. FUNCIÓN DE ACTIVACIÓN SIGMOID

La función sigmoid se puede usar en la capa de salida de una red neural junto con la función de entropía cruzada binaria (binary cross-entropy) para la clasificación de problemas binarios. La salida de esta función es modelada como una distribución de Bernoulli [15]. En la Figura 8 se puede observar la gráfica de la función sigmoid.

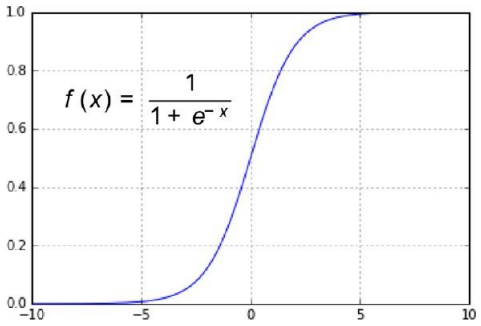


Figura 8 Función Sigmoid.
Fuente:[15]

1.4. AUTOENCODERS

Un autoencoder es una red neuronal que esta entrenada intencionalmente para copiar los datos de su entrada a su salida. Internamente, tiene una capa oculta $\boldsymbol{h}$ que describe una codificación usada para representar los datos de entrada. La red está estructurada por dos partes: un **encoder** cuya función es $h = f(x)$ y un **decoder**que produce una reconstrucción $r = g(h)$. Si un autoencoder tiene un aprendizaje de $g(f(x)) = x$, entonces no es un modelo muy útil. El objetivo de un autoencoder es no aprender a copiar perfectamente un dato de entrada. Por lo general, los autoencoders están restringidos a solo hacer una copia aproximada y solo hacer a los datos de entrada que se parecen o tienen semejanza a los datos de entrenamiento. Como el modelo se ve obligado a priorizar que aspectos de la entrada deben de ser copiados, a menudo aprenden propiedades útiles de los datos [18].

Un autoencoder además del encoder y decoder poseen dos componentes adicionales. Se los puede observar en la Figura 9.

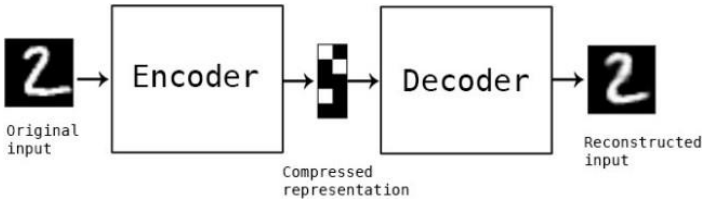


Figura 9 Representación de un autoencoder.
Fuente:[19]

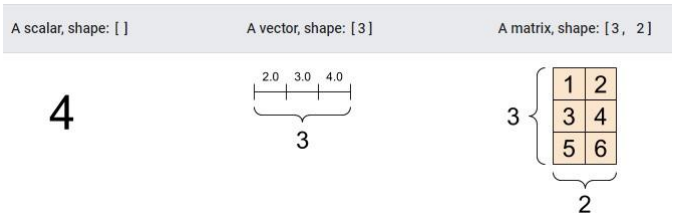
Donde,

- **Encoder:** Es la red que aprender a reducir las dimensiones de la entrada y los comprime en una representación codificada.
- **Cuello de botella (Bottleneck):** Es la capa que contiene a la representación codificada de los datos de entrada. Esta es la dimensión más baja de los datos de entrada.
- **Decoder:** Es la red que aprende a reconstruir los datos a partir de la representación codificada.
- **Perdida de reconstrucción:** Este es el método que mide el rendimiento del decodificador y la semejanza de la salida con la entrada [19].

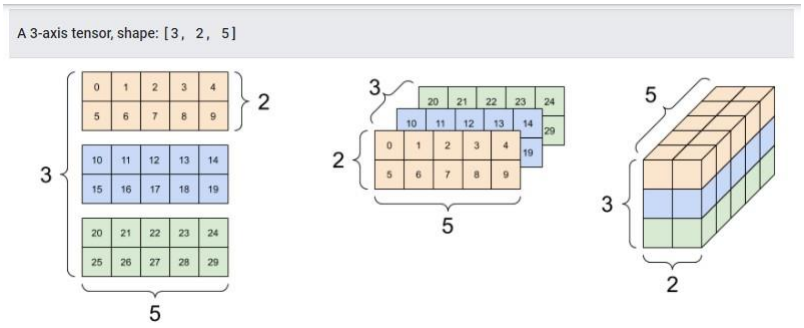
1.5. TENSORFLOW

Para explicar lo que es TensorFlow, primero se describe que es un tensor y que es Flow.

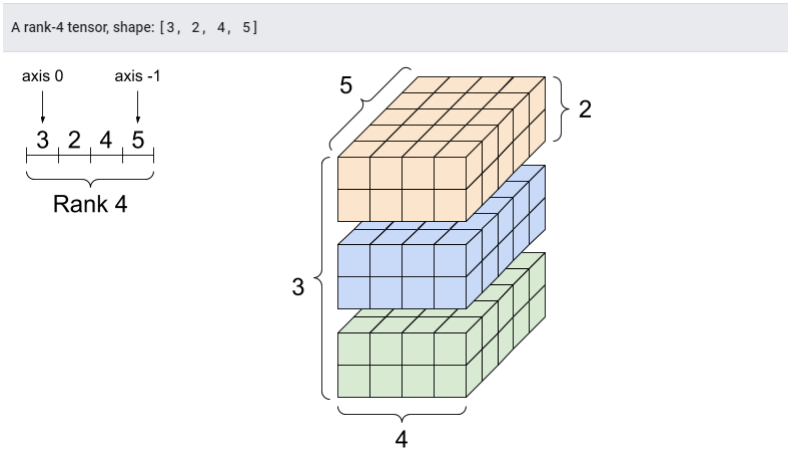
Un tensor es un arreglo multidimensional que representa tipos de datos, ver Figura 10. Los valores que contiene un tensor son datos idénticos de una forma conocida. Esta forma es la dimensionalidad del arreglo. En Machine Learning funciona con abundantes datos en formatos complejos, los tensores proporcionan una gran solución para trabajar con este tipo de datos de una manera más fácil. Unavez que los datos estén representados como tensores, se pueden aplicar operacionespara poder obtener resultados. El flujo (Flow) son las múltiples operaciones que se realizan con los tensores, esta librería está desarrollada para trabajar eficientemente con GPU o TPU [20].



a) Representación de un tensor escalar, tensor vector y tensor matriz.



b) Representación de un tensor de 3 ejes. Hay muchas formas de visualizar un tensor de más de dos ejes.



c) Representación de un tensor de 4 ejes.

Figura 10 Representaciones de tensores en varias dimensiones.
Fuente:[21]

TensorFlow permite que las operaciones (cálculos matemáticos) que se realizan con los tensores se describan como gráficos de flujo de datos de estado. Describir los modelos de Machine Learning o Deep Learning ayuda al rendimiento computacional durante el entrenamiento. En la Figura 11 se indica una estructura simple de un modelo de gráficos de flujo, la suma de dos tensores A y B (Add) y luego este resultado es multiplicado (MatMul) por un tercer tensor C [22].

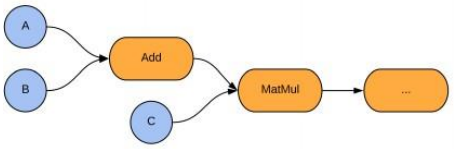


Figura 11 Ejemplo simple de un modelo computacional de TensorFlow.
Fuente:[22]

1.6. GOOGLE COLABORATY

Google Colaboratory más conocido como Colab, es un proyecto que tiene el objetivo de difundir la educación e investigación en Machine Learning. Brinda la ejecución de código Python con librerías más utilizadas preinstaladas para Machine Learning como TensorFlow, Matplotlib, Keras, etc. Todo esto ejecutado en una máquina virtual en la nube que se desactiva después de un periodo de tiempo y todos los datos y configuraciones se pierden. Pero, el código escrito se guarda en el almacenamiento de Google Drive. Además, Colab permite habilitar una GPU de una manera muy fácil, esto se hace para acelerar el proceso de ejecución de entrenamientos de los modelos. El código de Python se escribe sobre los conocidos cuadernos de Jupyter (Notebook Jupyter) donde trabajan con celdas, el cual es una buena herramienta para editar código, documentar el código, insertar imágenes, visualizar datos, escritura en LaTeX, y crear tablas. Es la herramienta más usada para compartir y replicar trabajos científicos desde los experimentos y resultados que son presentados en un mismo documento [23].

CAPÍTULO 2

2. MARCO METODOLÓGICO

En este capítulo se presenta el proceso y las técnicas que se aplicaron para crear la base de datos, el diseño y desarrollo de la arquitectura GAN junto con las fases de entrenamiento, evaluación y pruebas, además el desarrollo de un autoencoder para evaluar la GAN.

2.1. BASE DE DATOS

Para el desarrollo del sistema prototipo de generación de rostros mediante señales de voz, fue necesario construir una base de datos propia. En resumen, la base de datos, para este proyecto, contiene una clasificación de rostros con sus respectivas correspondencias de voz para el entrenamiento de la GAN donde a cada imagen le corresponde un audio de un segundo. Se obtiene a partir de la selección de videos de YouTube divididos por género: 5 Youtubers masculinos y 4 Youtubers femeninos. Se construyó está base de datos debido a que en Internet no hay una base de datos relacionada o parecida a la que requiere este trabajo.

La base de datos (Revisar APÉNDICE B para revisar requerimientos para crear entorno de trabajo) está construida con imágenes de rostros humanos sacados de fragmentos de **vlogs** de habla hispana. Para esto se optó por la búsqueda de videos dentro de la plataforma de YouTube de 9 Youtubers los cuales contienen información de cualquier ámbito (tutoriales, opiniones, etc.), pero tomando en consideración dos situaciones puntuales, la primera que cada persona en el video se encuentre en primer plano y la segunda consideración fue que el Youtuber aparezca en la medida de lo posible en casi todo el video sin tener transiciones de otras personas u otros eventos dentro del video, de tal manera que se facilite la detección de un mismo rostro para cada video. Por esta razón, se procedió a realizar una búsqueda minuciosa de videos que cumplan con los dos parámetros requeridos dentro de la plataforma de YouTube.

La base de datos está conformada por 10101 imágenes de rostros humanos con su respectiva correspondencia de audios, las cuales pertenecen a un 58% de rostros masculinos y el 42% de rostros femeninos, como se muestra en la Figura 12.

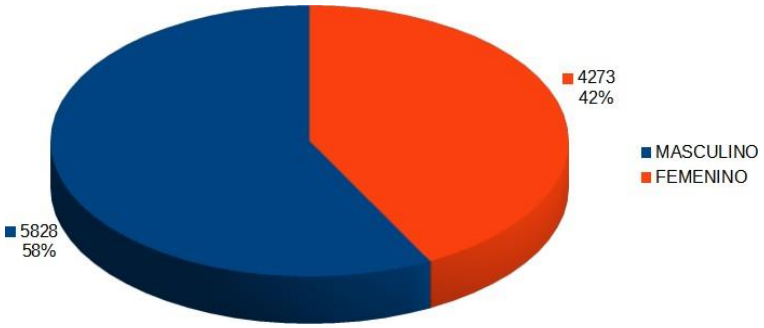


Figura 12 Estructura de la base de datos. Porcentajes de imágenes con su correspondiente audio. Fuente: Autores

En la Figura 13 se indica el número de imágenes y audio que contiene cada uno de los 9 Youtubers que a su vez está separada para datos de entrenamiento y de prueba. Para la división se consideró el mayor número de datos de cada Youtuber para el conjunto de entrenamiento y el menor para el conjunto de evaluación.

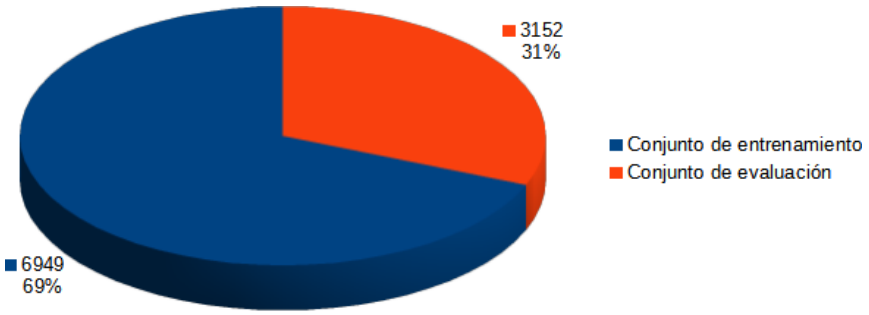


Figura 13 División de base de datos para conjunto de entrenamiento y conjunto de evaluación. Fuente: Autores

2.1.1. CONSTRUCCIÓN DE LA BASE DE DATOS

Luego de obtener los videos, se utilizan los algoritmos del trabajo [6] donde se hacen unas modificaciones para acoplar a este trabajo. Primero se extrae el audio de cada video. A continuación, se leen los videos con su respectivo audio y para cada video se fragmenta cada 4 segundos con la ayuda del modelo entrenado Haar Cascades [24] para la detección del rostro y al mismo tiempo se fragmenta el audio del mismo tamaño. El resultado de la aplicación del algoritmo original es el siguiente donde únicamente se modificaron las imágenes a blanco y negro.

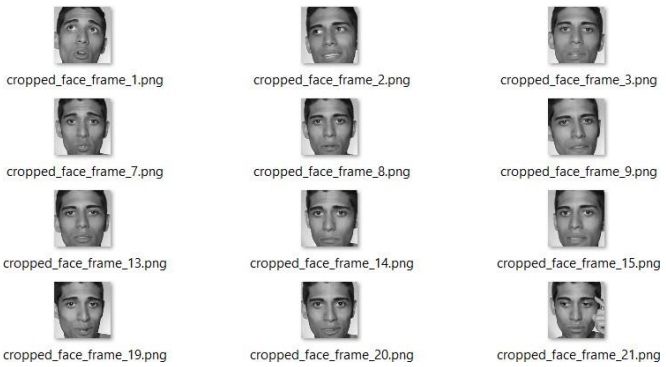


Figura 14 Resultado de detección de rostro de un video con algoritmo original.
Fuente: Autores

Luego de ajustar los parámetros y aplicando un par de condiciones en el algoritmo, se consigue detectar solo el rostro como tal, es decir, se detecta el rostro y se recorta parte de frente, mentón, orejas dando como resultado lo que puede observar en la Figura 15. Como punto final de aplicación de estos algoritmos se lleva a cabo la conversión del audio obtenido en el punto anterior a formato .wav. Las imágenes se pasan a formato jpg.

Figura 15 Resultado de detección de rostro de un video con algoritmo modificado.
Fuente: Autores

Lo que se consigue hasta este punto es la creación de una base de datos de números de imágenes, audio y distribución reflejados en la Figura 13. Todo el proceso realizado de estas actividades se indica en la Figura 16.

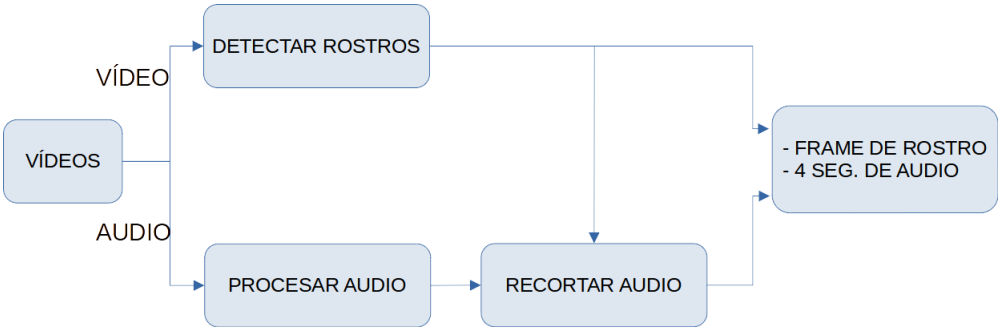


Figura 16 Esquema de construcción de base de datos.
Fuente: Autores

2.1.2. PREPROCESAMIENTO Y APLICACIÓN DE DATA AUGMENTATION DE LA BASE DE DATOS

Para usar esta base de datos en la GAN y Autoencoder propuestos en este trabajo, es importante preprocesar los datos que se tienen. Luego de una serie de pruebas de entrenamientos en Google Colaboratory, se llegaron a tener los siguientes

problemas: el tiempo de subida de la base de datos completa a la nube de Google, el tiempo de carga y lectura desde el almacenamiento de la nube y por último el tiempo que demora la red en entrenarse con los tamaños y pesos originales de las imágenes y audio de la base de datos creada. Por ello, se realiza lo siguiente:

- Normalización del tamaño de las imágenes:

El tamaño de las imágenes de la base de datos obtenida es de 232x232 píxeles y a color. Al comprobar que el tiempo computacional requerido para manejar las imágenes con ese tamaño, se consideró reducir las imágenes a un tamaño de 64x64 píxeles, dado que, con este tamaño, no se presenta distorsión alguna y la red se entrena adecuadamente. Además, se convirtieron las imágenes de color a blanco y negro y con un formato jpg, con esto se consigue una reducción de peso considerable.

- Normalización del tamaño de audio:

Antes de aplicar la técnica de Data Augmentation a los datos de entrenamiento, se procede a reducir el tamaño de los audios de evaluación. Lo que se tiene es que para una imagen de un rostro corresponde un audio de 4 segundos de tamaño en formato .wav. Entonces, se procede a recortar un segundo de audio de los cuatro que se tiene al inicio. Con esto realizado se tiene una nueva modificación que permite aplicar una forma de Data Augmentation al conjunto de entrenamiento.

- Aplicación de Data augmentation:

Luego de revisar los trabajos [25]–[27] donde se explican técnicas de data augmentation para aplicar a imágenes y audio, en el trabajo [28] se usa una GAN como técnica de aumento. Todos estos trabajos tienen técnicas eficientes para aplicar, la desventaja es que se aplican independientemente, tanto para las imágenes como el audio, es decir, no se pudo aplicar ninguna de estas técnicas a este trabajo. Esto debido al hecho de que el propósito de este trabajo es generar un rostro con el audio y este debe generarse lo más básico posible. No debe tener una rotación para ningún lado, siempre debe estar el rostro en una posición frontal. Por estos motivos lo que se procedió a aplicar como una técnica de data augmentation fue lo siguiente:

Tomando en cuenta que esto se aplica a los datos de entrenamiento. Entonces, se tiene que por cada imagen corresponde un audio de 4 segundos, se procede a repetir la imagen 3 veces más y se divide el audio en 4 de un segundo de tamaño. Para aumentar un poco más la base de datos se procede a repetir un audio de posición aleatoria para cada muestra de audio y se repite una imagen de rostro.

Como el rostro siempre va a ser el mismo para cada audio de 4 segundos originalmente, se consigue que ahora sean 5 audios de segundo de tamaño y 5 imágenes iguales para cada fotograma que se procesa en el algoritmo creado.

- Compresión de base de datos:

Una vez realizada las modificaciones mencionadas en los puntos anteriores, lo que se consiguió es tener más datos para el conjunto de entrenamiento y normalizar las imágenes y audio tanto para el conjunto de entrenamiento y de evaluación. Probando nuevamente en Google Colaboratory aún se encontraron problemas con el tiempo en la subida de la base de datos completa y la lectura y carga en la máquina virtual donde corre el código Python de la GAN. Lo que se resolvió con estos cambios fue la disminución de tiempo de entrenamiento de la GAN. La solución que se encontró fue comprimir la base de datos completa a un formato npy [29]. El formato facilita la subida y lectura de toda la base completa en la nube de Google, en la Figura 17 se observa el proceso de compresión de las imágenes y audio. En la Figura 18 se observa cómo está distribuida la base de datos final.

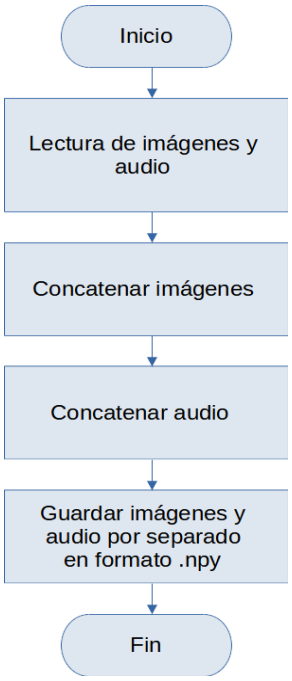


Figura 17 Proceso de comprensión de base de datos.
Fuente: Autores

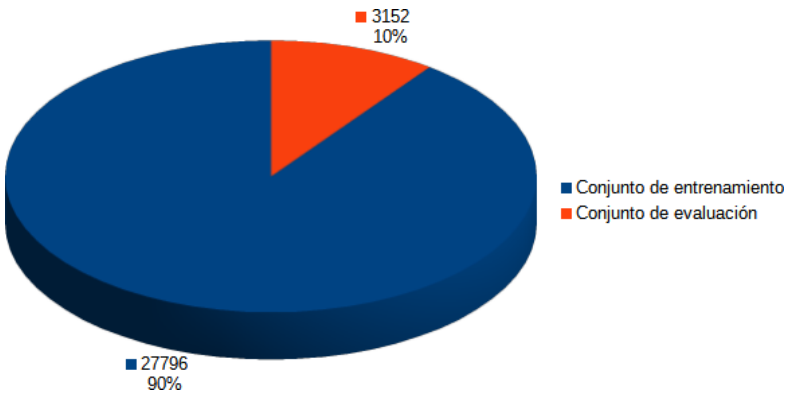


Figura 18 Base de datos final.
Fuente: Autores

2.2. RED GENERATIVA ADVERSARIA

Para este trabajo se seleccionó una red generativa adversaria (GAN), debido a su uso en su gran mayoría de trabajos para la generación de imágenes, textos, audios, además su característica de poseer dos redes donde siempre compiten entre ellas para así generar mejores resultados a la salida [6], [8], [30].

2.2.1. ARQUITECTURA DE LA RED

El diseño de la GAN de este trabajo se seleccionó luego de realizar varias pruebas de las combinaciones que se encontraron en el estado del arte. La configuración de la red se llama DCGAN considerada a partir de [8]. Para las configuraciones de las redes generadora y discriminadora se consideró la propuesta de [30] por el hecho que presentan buenos resultados en la generación de imágenes y su configuración es simple. Se debe de tener en cuenta que este trabajo se basa en crear un prototipo de sistema de generación de rostros los más sencillo posible, debido a esto, uno de los cambios iniciales para realizar pruebas fue de que todas las imágenes y audios tengan un menor tamaño y su velocidad de lectura sea rápida, entonces se optó por pasar las imágenes a blanco y negro y también que la generación de rostros sea en ese formato.

2.2.2. ARQUITECTURA DCGAN

- Arquitectura DCGAN con ruido como entrada

Es una arquitectura simple que se entrena con distintos tipos de dataset de imágenes que se encuentran en la web. La finalidad de esta red es generar imágenes realísticas ingresando como entrada un vector de ruido [30]. Su proceso de trabajo se ilustra en la Figura 19.

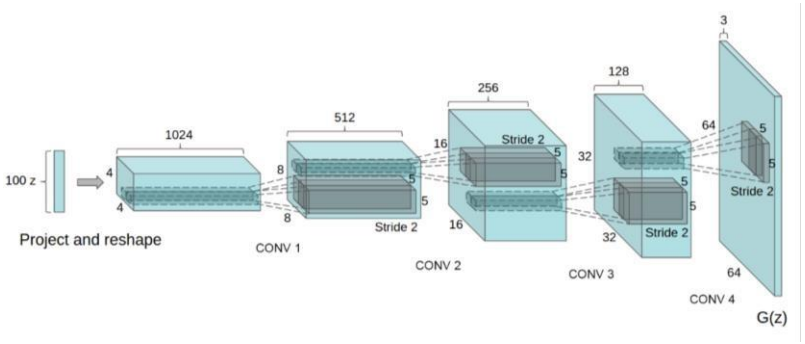


Figura 19 Esquema de GAN (red generadora) con ruido a la entrada.
Fuente:[30]

- Arquitectura DCGAN con audio como entrada

La estructura de esta red tiene un audio de un determinado tamaño que sirve como entrada a la red generadora y esta a su vez genera el rostro. Para ingresar los audios a la red generadora se utiliza un método para tratar al audio y esto se llama realizar un embedding del audio. A la salida de la red generadora se obtiene una imagen la cual pasará por redes discriminadoras, la una verifica si la imagen generada es real o no. La segunda red realiza una verificación de que tan idénticas son las imágenes generadas con las imágenes del conjunto de entrenamiento y evaluación [8]. En la Figura 20 se puede observar a detalle como es el proceso que realiza la red completa.

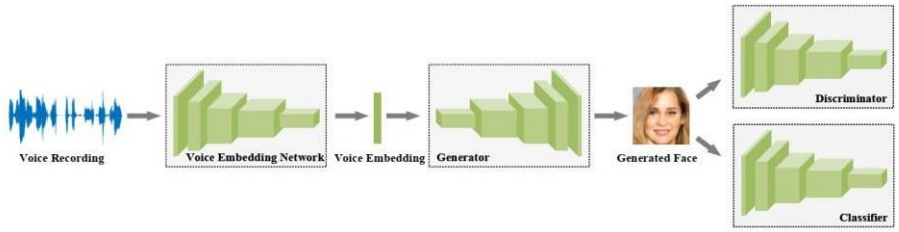


Figura 20 Esquema base del desarrollo de la GAN de este trabajo.
Fuente:[8]

Voice Embedding Network			Generator		
Layer	Act.	Output shape	Layer	Act.	Output shape
Input	-	$64 \times t_0$	Input	-	$64 \times 1 \times 1$
Conv $3_{/2,1}$	BN + ReLU	$256 \times t_1$	Deconv $4 \times 4_{/1,0}$	ReLU	$1024 \times 4 \times 4$
Conv $3_{/2,1}$	BN + ReLU	$384 \times t_2$	Deconv $3 \times 3_{/2,1}$	ReLU	$512 \times 8 \times 8$
Conv $3_{/2,1}$	BN + ReLU	$576 \times t_3$	Deconv $3 \times 3_{/2,1}$	ReLU	$256 \times 16 \times 16$
Conv $3_{/2,1}$	BN + ReLU	$864 \times t_4$	Deconv $3 \times 3_{/2,1}$	ReLU	$128 \times 32 \times 32$
Conv $3_{/2,1}$	BN + ReLU	$64 \times t_5$	Deconv $3 \times 3_{/2,1}$	ReLU	$64 \times 64 \times 64$
AvePool $1 \times t_5$	-	64×1	Deconv $1 \times 1_{/1,0}$	-	$3 \times 64 \times 64$
Discriminator			**Classifier**		
Layer	Act.	Output shape	Layer	Act.	Output shape
Input	-	$3 \times 64 \times 64$	Input	-	$3 \times 64 \times 64$
Conv $1 \times 1_{/1,0}$	LReLU	$32 \times 64 \times 64$	Conv $1 \times 1_{/1,0}$	LReLU	$32 \times 64 \times 64$
Conv $3 \times 3_{/2,1}$	LReLU	$64 \times 32 \times 32$	Conv $3 \times 3_{/2,1}$	LReLU	$64 \times 32 \times 32$
Conv $3 \times 3_{/2,1}$	LReLU	$128 \times 16 \times 16$	Conv $3 \times 3_{/2,1}$	LReLU	$128 \times 16 \times 16$
Conv $3 \times 3_{/2,1}$	LReLU	$256 \times 8 \times 8$	Conv $3 \times 3_{/2,1}$	LReLU	$256 \times 8 \times 8$
Conv $3 \times 3_{/2,1}$	LReLU	$512 \times 4 \times 4$	Conv $3 \times 3_{/2,1}$	LReLU	$512 \times 4 \times 4$
Conv $4 \times 4_{/1,0}$	LReLU	$64 \times 1 \times 1$	Conv $4 \times 4_{/1,0}$	LReLU	$64 \times 1 \times 1$
FC 64×1	Sigmoid	1	FC $64 \times k$	Softmax	k

Figura 21 Configuraciones GAN.
Fuente:[8]

2.2.3. ARQUITECTURAS PROPUESTAS

Basados en la configuración y estilo de entrenamiento del modelo DCGAN expuesto anteriormente, la primera estructura propuesta para el entrenamiento del sistema prototipo se observa en la Figura 22. El tamaño y característica del "vector ruido" es el mismo que se utiliza en el trabajo [30] , es decir, el vector contiene 100 valores aleatorios de una distribución normal.

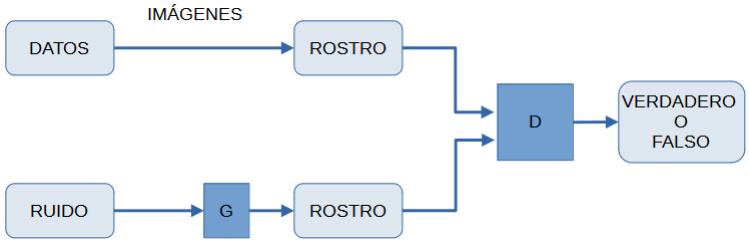


Figura 22 Esquema de la primera estructura de entrenamiento. G y D son las redes generadora y discriminadora.
Fuente: Autores

Como segunda red propuesta en este trabajo, se tiene al siguiente esquema que se observa en la Figura 23. Ahora, este modelo condiciona a su red generadora como entrada un audio de voz de un segundo de tamaño que corresponde a la imagen del rostro correspondiente.

Previo al ingreso de cada audio a la red generadora, se debe de aplicar un proceso mediante el cual se extraigan las características del audio y que pueda ingresar a la red. El proceso seleccionado para poder realizar dicho proceso es la extracción de los "Coeficientes Cepstrales en las Frecuencias de Mel" o "MFCC" con siglas en inglés. Este método es uno de los usados en trabajos relacionados al "Reconocimiento de voz" [31]–[33].

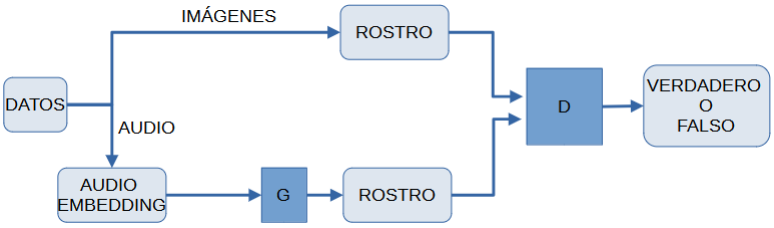


Figura 23 Esquema de la segunda estructura de red propuesta.
Fuente: Autores

En la Tabla 1 se indica el tipo de cada capa, el tamaño y número de filtros de cada capa tanto para la red generadora (capas deconvolucionales) como para la red discriminadora (capas convolucionales). Además, también se indican las funciones de activación usadas en cada capa. Esta arquitectura de la red es usada para los dos esquemas propuestos. Para entender las características de las capas, se

explica con el siguiente ejemplo: $Deconv\ 3x3_{/2,same}$, donde el kernel es de tamaño 3x3, 2 es el valor del paso (stride) y same corresponde a la configuración de padding.

Tabla 1 Arquitectura de Red Generativa Adversaria propuesta.
Fuente: Autores

Red Generadora				Red Discriminadora		
Capa	Activación	Forma de salida		Capa	Activación	Forma de salida
Input	ReLU	4x4x1024		Input	LReLU	64x64x64
Deconv $3x3_{/2,same}$	ReLU	8x8x512		Conv $3x3_{/2,same}$	LReLU	32x32x128
Deconv $3x3_{/2,same}$	ReLU	16x16x256		Conv $3x3_{/2,same}$	LReLU	16x16x256
Deconv $3x3_{/2,same}$	ReLU	32x32x128		Conv $3x3_{/2,same}$	LReLU	8x8x512
Deconv $3x3_{/2,same}$	ReLU	64x64x64		Conv $3x3_{/2,same}$	LReLU	4x4x1024
Deconv $1x1_{/1,same}$	Tanh	64x64x1		FC	Sigmoid	1

2.3. DESARROLLO DE LOS MODELOS

Los modelos de redes generativas adversarias propuestos fueron desarrollados completamente en lenguaje Python [34]. Este lenguaje es el más utilizado para desarrollo de modelos de Machine Learning y Deep Learning según el estado del arte revisado.

2.3.1. HARDWARE Y SOFTWARE

Para que el modelo funcione adecuadamente, es necesario tener instalado lo siguientes módulos (estos son los principales módulos usados para la manipulación de imágenes, audio, visualización de datos, creación y entrenamiento de modelo):

- Librosa: librería que procesa audio y música [35].
- Matplotlib: librería que crea visualizaciones estáticas, animadas e interactivas de datos [36].
- NumPy: librería que trabaja con arreglos n-dimensionales [37].
- OpenCV: librería de visión computacional, permite el procesamiento de imágenes [38].
- Pillow: librería que permite la manipulación de imágenes [39].
- TensorFlow: librería que permite el desarrollo y entrenamiento de modelos de Machine Learning [40].
- Python versión 3.7.12 y GPU (Google Colab) Tesla K80 [41].

2.3.2. PREPARACIÓN DE LA BASE DE DATOS

Los datos obtenidos luego de aplicar la Compresión de base de datos descrita en la sección 2.1.2 y que se indica en la Figura 18 son lo que se suben a la nube de

Google Drive [42], el cual permite almacenar todo tipo de datos (imágenes, audio, documentos, etc.). La cuenta permite almacenar 15GB (Revisar APÉNDICE B para más información de cuenta Drive). El peso total de la base de datos es aproximadamente 4 GB, así que es suficiente espacio para subir la base. Como la base de datos está en formato npy, el tiempo de subida a Drive es relativamente corto.

2.3.3. PROGRAMACIÓN DE LOS MODELOS

La programación de la GAN fue realizada con TensorFlow y Python 3.7 en un cuaderno de Jupyter que se compila en Google Colab (Revisar APÉNDICE B para información de Google Colab). Una vez cargada la base de datos en Google Drive, se crea el cuaderno. Para iniciar su uso, se debe de encender la máquina virtual como se indica en Figura 24 donde se habilita la GPU, esto se hace dirigiéndose a las configuraciones del tipo de ejecución en Google Colab que por defecto viene con CPU.

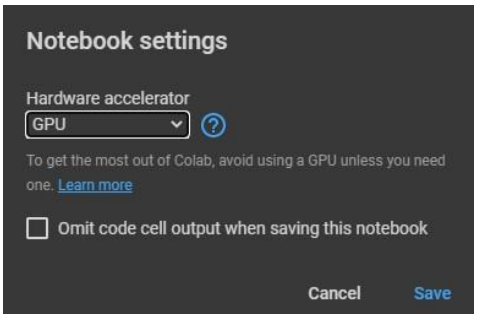


Figura 24 Encendido de máquina virtual con GPU. Primer paso que realizar.

Luego, se debe de cargar el almacenamiento de Drive en Colab para la posterior lectura de la base de datos. En la Figura 25, se indica las líneas de código que permite realizar dicha acción. Adicional a esto, se debe de dar un clic en el enlace en azul que brinda un código de autorización temporal y se debe de ingresar para tener el permiso para cargar el almacenamiento.

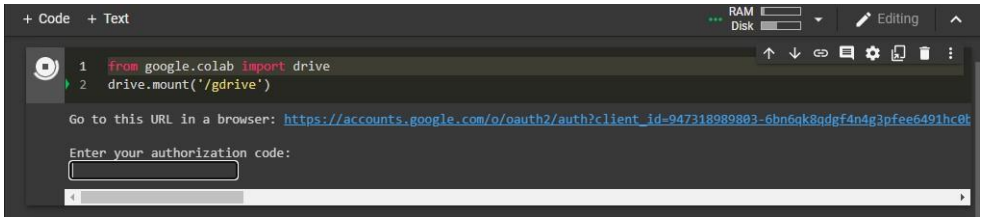


Figura 25 Líneas de código que permite la carga del almacenamiento de Drive en la máquina virtual de Colab.

Después de cargar el almacenamiento se debe de realizar la lectura y decodificación de la base de datos, permitiendo luego la manipulación de la base de datos para poder ingresar a la red para el entrenamiento y evaluación. En la Figura 26 se puede ver la secuencia del entrenamiento de la GAN con un ruido como entrada.

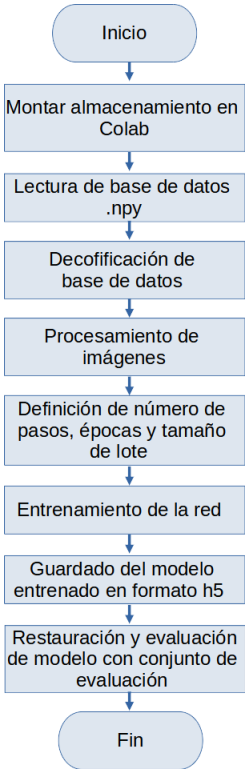


Figura 26 Algoritmo de entrenamiento del primer modelo.
Fuente: Autores

A continuación, se presenta el segundo algoritmo de entrenamiento de la red, tal como se indica el cambio que tiene esta red respecto a la anterior es en el condicionamiento que se le da a la entrada. La entrada de la red anterior recibe como entrada un vector de ruido de valores aleatorios de una distribución normal, en cambio, ahora recibe el audio correspondiente a la imagen del rostro. En la Figura 27 se puede observar el esquema del algoritmo de la segunda red.

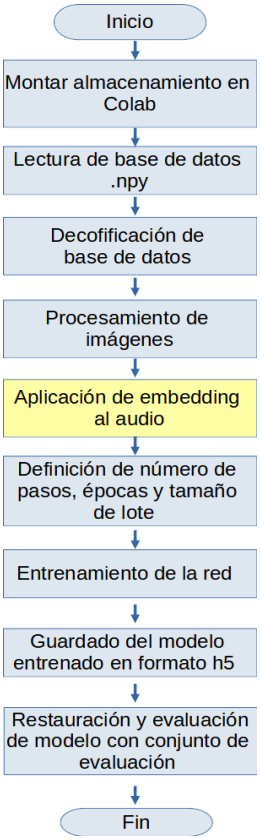


Figura 27 Algoritmo de entrenamiento del segundo modelo.
Fuente: Autores.

2.3.4. ENTRENAMIENTO DE LAS REDES

La base de datos contiene 30948 imágenes con su respectivo audio. Las imágenes son de tamaño 64x64 pixeles y el audio de un segundo. El conjunto de datos de entrenamiento está formado por 27796 de imágenes y audio, 3152 datos son para el conjunto de pruebas. En total son 9 Youtubers que forman esta base de datos, 6 son para el entrenamiento y 3 para las evaluaciones. Para los datos del

entrenamiento se aplica un aumento para tener más datos para el aprendizaje. Para el entrenamiento de ambas redes fueron desarrolladas y compiladas en Google Colabcon la GPU habilitada (Nvidia Tesla K80).

Para el entrenamiento de la primera red se divide en lotes para que el proceso de compilación sea más rápido. Entonces, lo que se define es un vector aleatorio con valores de una distribución normal para que la red empiece generando ruido como imagen hasta que aprenda a generar un rostro con las características de los 6 Youtubers y luego probar con el conjunto de evaluación y medir su eficiencia.

Para el proceso de entrenamiento de la segunda red propuesta, se toma los mismos pasos anteriores. Pero ahora, se modifica lo siguiente, el ruido es reemplazado por los audios correspondientes a las imágenes de los rostros. Previo al ingreso de la entrada de la red generadora, a los audios se le extraen los MFCCs variando el número de coeficientes en cada entrenamiento.

2.4. AUTOENCODER

Para probar la eficiencia de la red como paso adicional, se procede a configurar un autoencoder para poder ver si esta red puede generar mejores resultados o no que la GAN. Se desarrollaron dos autoencoders con el mismo objetivo de los entrenamientos de la GAN. Se configuró el primer autoencoder con imágenes como entrada (encoder) para que su salida (decoder) pueda generar las mismas imágenes. Para a segunda configuración se introduce la imagen concatenada con el audio correspondiente intentando que a la salida reproduzca la misma imagen.

A continuación, se indica en la Figura 28 como es el algoritmo de entrenamiento del primer autoencoder. Este autoencoder está basado en uno de los esquemas revisados de [43] donde este modelo es utilizado para reproducir imágenes para el análisis médico. La entrada está configurada para que ingrese imágenes a blanco y negro de tamaño 64x64 píxeles.

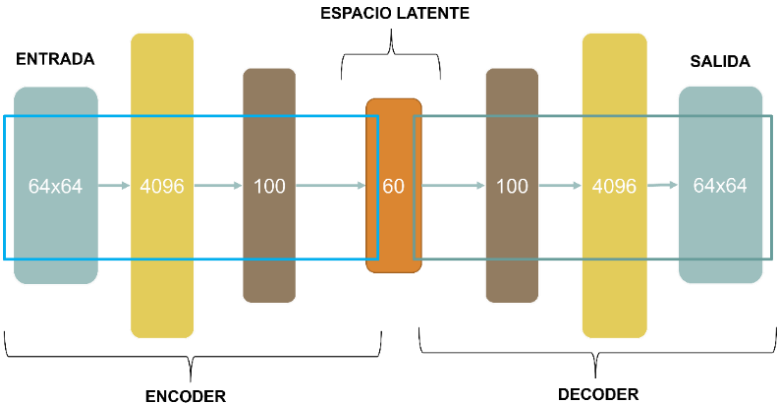


Figura 28 Esquema de Autoencoder con imágenes como entrada.
Fuente: Autores

El esquema del segundo autoencoder se observa en la Figura 29. Se realizaron adaptaciones del primer autoencoder para que este diseño pueda entrenarse con una entrada de imagen de 64x64 píxeles a blanco y negro al cual se concatena su audio correspondiente. El audio que se concatena a la red es el mismo audio que ingresa a la red generadora de la GAN, es decir, es un audio aplicado MFCC. Luego de concatenar la imagen con su audio de voz, se tiene un vector unidimensional de tamaño 4136 el cual luego de varias pruebas se aplica PCA para reducir su tamaño conservando las características más importantes de los datos originales [44] y pueda ingresar de esa manera y que se entrene hasta obtener a la salida la imagen del rostro.

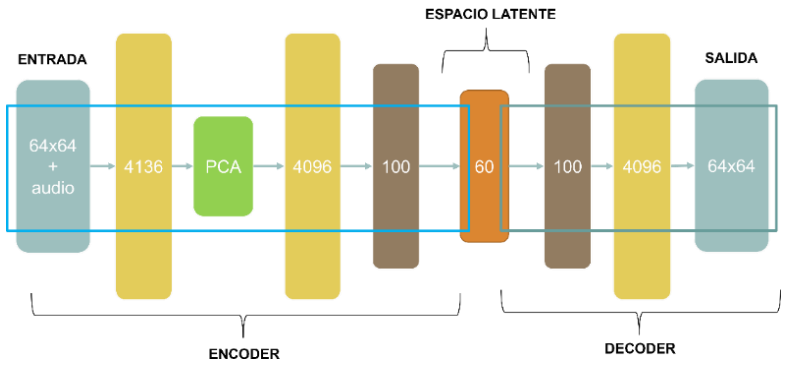


Figura 29 Esquema de Autoencoder con imágenes concatenado audio como entrada.
Fuente: Autores

2.5. EVALUACIÓN Y PRUEBAS DE MODELOS

La métrica que se usa para evaluar los resultados de las redes propuestas es el "PSNR" (Peak Signal to Noise Ratio). Esta métrica es usada en [45] donde sirve para evaluar las imágenes generadas por una GAN y mejorar su resolución.

- Proceso para calcular el PSNR

Primero se calcula MSE (Mean Squared Error). Con la ecuación (2) se toma la imagen 1 (P) y se restan todos sus píxeles con la imagen 2 (Q) y se eleva al cuadrado. M y N es el ancho y alto de la imagen respectivamente [46].

$$MSE = \frac{1}{MN} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} (P(i,j) - Q(i,j))^2 \quad (2)$$

Con el MSE calculado se procede a calcular el PSNR, donde W=256 [46].

$$PSNR = 10 \log_{10} \left(\frac{(W-1)^2}{MSE} \right) = 20 \log_{10} \left(\frac{W-1}{\sqrt{MSE}} \right) \quad (3)$$

En la Tabla 2 se pueden ver los valores de semejanza que tienen dos imágenes luego de aplicar PSNR.

Tabla 2 Métrica de PSNR.
Fuente: [46]

PSNR [dB]	**SEMEJANZA**
>30	Alta
[40 - 45]	Muy alta
[38 - 40]	Aceptable
< 30	deficiente

CAPÍTULO 3

3. IMPLEMENTACIÓN Y ANÁLISIS DE RESULTADOS

En este capítulo se presentan los resultados obtenidos de las redes prototipo desarrolladas en este trabajo. Se presenta los resultados de entrenamiento de cada red, así como su respectivo valor máximo de métrica de cada evaluación. Y se indica cual es la configuración con mejores resultados. Los resultados a presentar en este capítulo corresponden a los obtenidos del entrenamiento y también se comprobó su funcionamiento en una maquina local con GPU del grupo GIIRA. Los datos tienen una diferencia mínima con los resultados obtenidos de Google Colab como se indica en la Figura 32. Únicamente se mostrarán los resultados de las imágenes generadas con modelos entrenados con Google Colab.

3.1. RESULTADOS DEL ENTRENAMIENTO

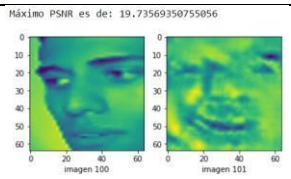
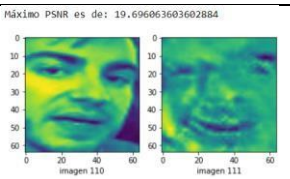
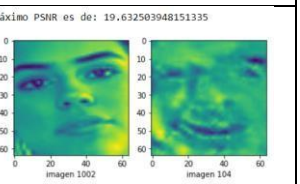
Los resultados que se presentan son obtenidos con el conjunto de evaluación, que está formado en base de la información tomada de 3 Youtubers. Los resultados de las tablas donde se indica Prueba 1, hacen referencia al Youtuber 1 y así sucesivamente. El resultado que se presenta en Prueba 1 tanto para la Tabla 3 como la Tabla 6, es la aplicación de la métrica PSNR a todo el conjunto de imágenes de evaluación con las imágenes generadas con el audio de Youtuber 1, y se selecciona el PSNR más alto. Con esto se puede verificar si la red genera una imagen parecida a la imagen del rostro del Youtuber 1 únicamente con las características aprendidas del conjunto de entrenamiento. En la sección 3.1.4 se analizan los resultados para indicar cual fue la configuración con la que se obtuvo los mejores resultados y ver la eficiencia que tiene el sistema prototipo versus el Autoencoder con la generaciónde imágenes dando un audio a la entrada en ambas redes. El número de épocas parael entrenamiento de cada GAN fue de 50, con un Learning Rate de 0.0001.

3.1.1. RESULTADO DE RED GENERADORA CON ENTRADA DE RUIDO

En la Tabla 3 se puede visualizar la generación de imágenes con un vector de ruido como entrada. Los valores calculados de PSNR son menores a 20. El

objetivo es sobrepasar el valor de 30 como se indica en la Tabla 2. El tiempo aproximado de entrenamiento fue de 2 horas y media.

Tabla 3 Resultados de red entrenada con ruido como entrada.
Fuente: Autores

PRUEBA 1	PRUEBA 2	PRUEBA 3
		

3.1.2. RESULTADO DE RED GENERADORA CON ENTRADA DE AUDIO

Los resultados que se visualizan en la Tabla 6 (Revisar APÉNDICE A), son resultados obtenidos luego de entrenar la red GAN con diferentes valores de números de coeficientes MFCC. El número de coeficientes utilizado varió en función de la complejidad computacional y del tiempo utilizada para la ejecución del entrenamiento y fueron los siguientes: 20, 30, 40, 50 y 60. El tiempo de entrenamiento con cada variación de número de coeficientes tuvo un tiempo aproximando de 4 horas por cada una. En total fueron 20 horas de entrenamiento con estas pruebas de la red.

3.1.3. RESULTADO DE AUTOENCODER

Los resultados obtenidos con el Autoencoder con entrada de imágenes tienen valores de PSNR muy buenos. Sus valores de PSNR están por encima de 45 que se considera un valor muy alto. Pero, solo verifica que este autoencoder es capaz de reproducir imágenes a la salida con imágenes solas como entrada. Basados en estos resultados se modifica esta red para poder concatenar a la entrada la imagen con su audio. Los resultados de esa configuración se indican en la Figura 31. Por el momento esta configuración nos sirve como un punto base.

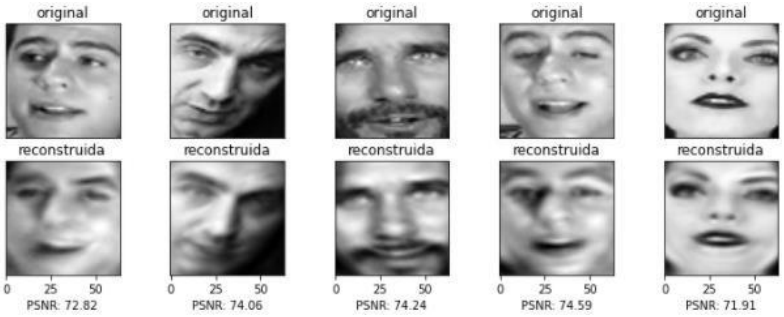


Figura 30 Resultados de Autoencoder con imágenes puras como entrada.
Fuente: Autores.

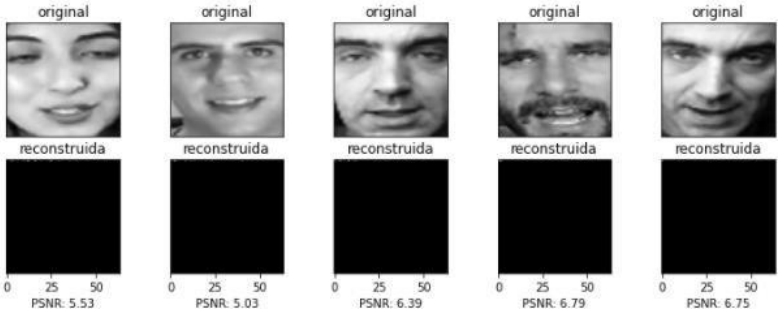


Figura 31 Resultados de Autoencoder con imágenes concatenado con audio como entrada.
Fuente: Autores.

3.1.4. COMPARACIÓN DE RESULTADOS

El valor de PSNR promedio de la GAN que recibe como entrada un vector de ruido tiene valores menores a 20 dB, por lo que se intenta aumentar el valor de PSNR para sobrepasar a 30 dB haciendo la modificación a introducir el audio de voz que corresponde a cada imagen de rostro. El objetivo es sobrepasar el valor de PSNR de 30 con esta nueva configuración.

La Figura 32 indica los valores de PSNR promedios calculados a partir de la aplicación de diferente número de coeficientes MFCC. Se presentan los resultandos tanto del entrenamiento en Google Colab y la PC del grupo GIIRA. Ambos tienes valores muy semejantes así con esto se comprueba que el entrenamiento en nube se obtiene resultados muy parecidos que con entrenamientos de manera local.

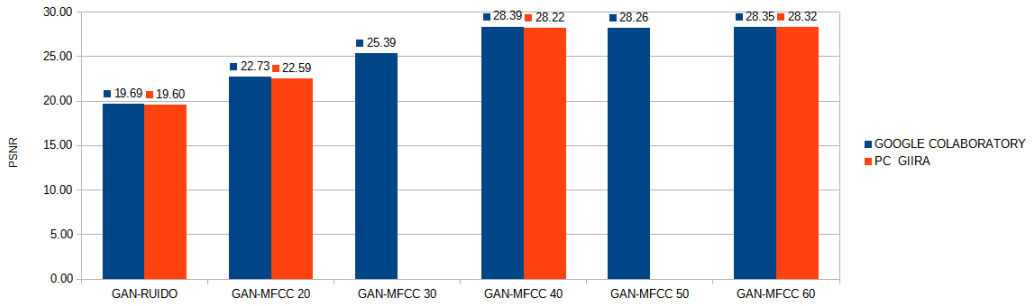


Figura 32 Resultados de las evaluaciones con la métrica PSNR.
Fuente: Autores

Entonces, de acuerdo con los valores de la Figura 32 se selecciona como mejor resultados a la GAN entrenada con 40 coeficientes MFCC con un PSNR de 28.39 dB. Las imágenes generadas se pueden revisar en la Tabla 6 (Revisar APÉNDICE A).

Adicional, se indica en el siguiente apartado la mejora que posee la red GAN con respecto al entrenamiento del Autoencoder con entrada de imágenesconcatenada con el audio de voz.

- **Mejora de generación de imágenes comparando el PSNR promedio de GAN con el del Autoencoder.**

Luego de aplicar PSNR al conjunto de evaluación se obtienen valores inferiores a 6 dB. Al calcular un PSNR promedio de todo el conjunto se tiene que su valor PSNR es 5.85 dB.

Se calcula la mejora entre el PSNR promedio de la GAN con 40 coeficientes y este último Autoencoder y se tiene una mejora de 79% de PSNR de la GAN con respecto al Autoencoder.

3.1.5. COMPARACIÓN DE LAS TÉCNICAS DE AUTOENCODER Y GANS PARA LA RECUPERACIÓN DE INFORMACIÓN CODIFICADA

Como se observa en la Tabla 4 los valores máximos obtenidos por cada red propuesta en este trabajo están en orden descendente.

Tabla 4 Resultados obtenidos de cada red entrenada.
Fuente: Autores

REDES	PSNR PROMEDIO [dB]
1. AUTOENCODER SIN AUDIO	73.80
2. GAN CON AUDIO	28.39
3. GAN CON RUIDO	19.69
4. AUTOENCODER CON AUDIO	5.85

La red 1 de la Tabla 4 tiene un valor de PSNR muy alto, es decir, genera imágenes muy parecidas a las imágenes de rostros que se ingresan como entrada, es una red que funciona bien. Esta red se omite en las comparaciones debido a que, esta red no posee una entrada configurada para que ingrese una imagen con su respectivo audio. A continuación, se presentan las mejoras que se tiene entre las 3 redes restantes de la Tabla 4.

Tabla 5 Comparación de redes.
Fuente: Autores

Comparación de redes	Mejora [%]
GAN CON AUDIO VS GAN CON RUIDO	30.64
GAN CON AUDIO VS AUTOENCODER CON AUDIO	79.40
GAN CON RUIDO VS AUTOENCODER CON AUDIO	70.30

En la Tabla 5 se observan las mejoras que tienen una red sobre las otras. La GAN con audio (GAN con MFCC de 40) es la red que mejores resultados obtuvo sobre las otras dos redes. Tiene mejores resultados que la GAN con ruido y que el Autoencoder con audio.

CAPÍTULO 4

4. CONCLUSIONES Y RECOMENDACIONES

4.1. CONCLUSIONES

Este trabajo consistió en dos partes, la primera parte fue la creación de una base de datos que este conformada por imágenes de rostros con su respectivo audio de voz. La segunda parte fue el desarrollo de configuraciones de redes generativas adversarias que generen imágenes de rostros a partir de un audio de voz. Los resultados fueron evaluados en dos partes, la primera parte por una métrica y la segunda por los resultados de la métrica aplicados a imágenes generadas por un Autoencoder que genera imágenes de rostros a partir de imágenes concatenadas con su audio de voz. Con esto descrito se procede a concluir que:

- Las modificaciones que se realizaron a la base de datos inicial creada para este proyecto ayudó con las velocidades de carga a la nube, de la lectura de imágenes y audio, procesamiento de estos y el entrenamiento de los modelos.
- El propósito de este trabajo es generar imágenes de rostros a partir de un audio de voz de un hablante donde se toma base una GAN, donde el rostro generado debe de poseer los rasgos más característicos del hablante. Es decir, se eliminó cabello, una porción del mentón y frente, orejas de las imágenes de la base de datos.
- Basados en una GAN para la generación de imágenes a partir de un vector de ruido, se configuró la red para que se puedan entrenar imágenes de rostros de personas donde los resultados obtenidos fueron de 19.68 dB que basados en la métrica de PSNR deben de sobrepasar el valor de 30 dB para ser considerado un resultado de una imagen de rostro.
- La GAN se modificó para que reciba el audio de voz en vez del vector de ruido, donde los resultados obtenidos fueron mejorados a un 28.39 dB. El valor del resultado del PSNR para considerar una imagen de un rostro en este caso debe de ser de 30 dB, pero en este caso los resultados son cercanos.
- La configuración de un Autoencoder para generar imágenes de rostros a partir de imágenes concatenadas con audio no tuvo resultados para nada buenos, su PNSR es de 5.85 dB donde comparando con la GAN, la GAN tiene resultados muy superiores a esta red.

4.2. RECOMENDACIONES

Se recomienda que:

- La generación de imágenes de rostros a partir de un audio de voz es un tema de complejidad, por el hecho que dé se deben de tomar en cuenta diferentes aspectos para generar un rostro, se debe de considerar la edad, rasgos físicos de una región o país, genero, color de piel, para esto se necesita una base de datos donde exista una clasificación de todo este tipo de consideraciones. Se podría optar un proyecto como este para un país en concreto, debido a que en el mundo existe un sinnúmero de varias razas, etc.
- El tamaño de 64x64 píxeles es un tamaño adecuado para entrenar una red para generar imágenes de rostros, pero se podría aumentar el tamaño sin problema, lo que sí se puede aumentar para obtener más mejoramiento de la red es aumentar el tamaño del audio, en este caso se usa el audio de un segundo de tiempo.
- Para reproducir un el proceso entrenamiento de esta red, es necesario el uso de una GPU, por el hecho de que minimiza los tiempos de entrenamiento de la red.
- En la Colab se debe procurar hacer pruebas iniciales con el entrenamiento de la red con el uso de CPU, debido a que en este tipo de ejecución Colab no pone un tiempo límite para su uso. Una vez que se tenga preparado todo y realizado configuraciones de la red de forma apropiada, habilitar la GPU para entrenar. El tiempo de uso de la GPU de Colab es de 12 horas, una vez agotado el tiempo de uso Colab te permite habilitar la GPU 12 o más horas después del último uso.

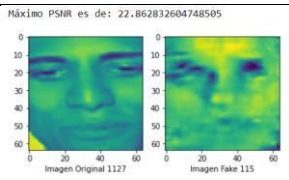
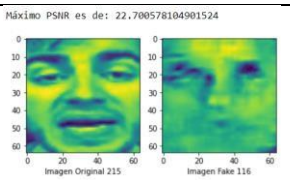
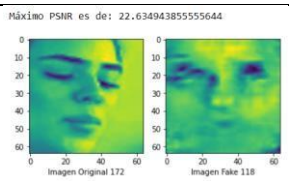
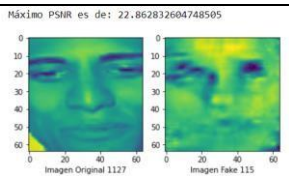
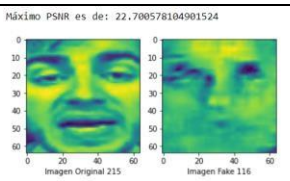
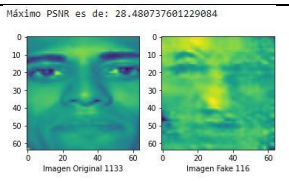
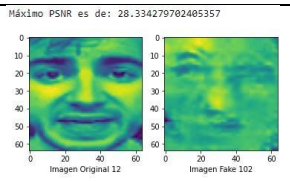
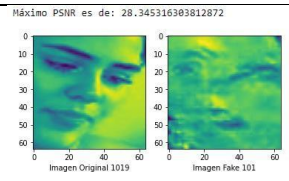
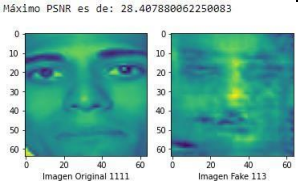
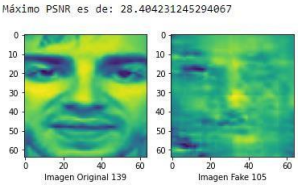
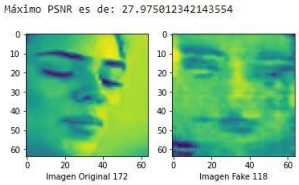
4.3. TRABAJOS FUTUROS

Este proyecto está abierto a seguir siendo desarrollado para su mejora para apoyar en la ayuda de forense acústica. Este tipo de proyecto si se mejora se puede a llegar a generar imágenes a partir de grabaciones de audio o video donde no se vea a la persona que hablar y poder generar un bosquejo simple de como es. Esto abre una opción de muchas ayudas para los investigadores de delitos de secuestros, extorciones o también puede ayudar en el caso de alguna sentencia donde al denunciado este siendo acusado con pruebas digitales falsas.

APÉNDICES

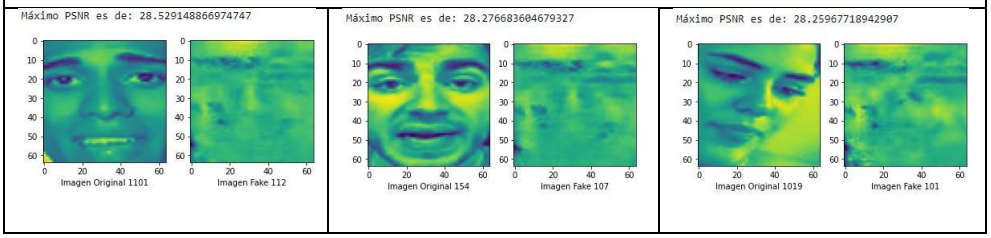
APÉNDICE A

Tabla 6 Resultados de PSNR de imágenes generadas con diferente número de coeficientes de MFCC. Fuente: Autores

PRUEBA 1	PRUEBA 2	PRUEBA 3
20 MFCCs		
		
30 MFCCs		
		
40 MFCCs		
		
50 MFCCs		
		

APÉNDICES

60 MFCCs



APÉNDICE B

Este trabajo para su desarrolla en dos partes:

1. **Creación de base de datos**

Para este paso, se desarrolla en una maquina física (poder de cómputo es opcional) por el hecho de que se necesita descargar videos, manipular, reproducir, visualizar los datos que se van obtenido en cada paso de la creación de base de datos. Estos procesos no son fáciles de realizar en la nube de Google.

Además, se requiere instalar la versión de Ubuntu 18.04 LTS donde los algoritmos basados en el trabajo [6] y las librerías de soporte para archivos multimedia que están en ese trabajo son compatibles con esa versión. Para este trabajo se hizo uso de una máquina virtual y sobre este correr Ubuntu.

Software utilizado para crear entorno de trabajo de este proceso:

- (Máquina Virtual) Virtual Box: https://www.virtualbox.org/
- (Sistema Operativo) Ubuntu 18: https://releases.ubuntu.com/18.04/
- (Lenguaje de programación) Python 2.7: https://www.python.org/download/releases/2.7
- (Editor de código) Visual Studio Code: https://code.visualstudio.com/
- (Librería para tratamiento de imágenes) OpenCV 3.4.3: https://docs.opencv.org/3.4.3/

2. **Entrenamiento de modelos**

Para este paso, si se requiere de poder de cómputo. Una solución viable fue usar una GPU de Google. Al momento de crear una cuenta de Google (Cuenta Gratis), tienes acceso a poderosas herramientas tanto de Apps de Oficina (Procesadores de texto, Hojas de cálculos, Editores de Presentaciones, etc.), espacio de almacenamiento de 15 GB, Editor de código Python y muchas otras.

Las dos últimas mencionadas son las utilizadas para este trabajo la cual sirve para alojamiento de la base de datos y el entrenamiento de la GAN y del Autoencoder en código Python.

- **Google Drive**

Google Drive es un espacio de almacenamiento en los servidores de Google, este brinda una capacidad de almacenamiento de 15 GB [42]. Para poder usar este producto solo debes de ingresar a tu cuenta Google y listo. Tiene herramientas gráficas que permite la subida y bajada de archivos, cambio de nombre, eliminación de documentos, papelera de reciclaje, creación de nuevas carpetas. Este último paso

es recomendado, para tener en una carpeta la Base de Datos y en otra el código de los modelos.

- **Google Colab**

Es una plataforma que corre en una máquina virtual en la nube de Google. Permite escribir código Python en cuaderno de Jupyter (Jupyter Notebook). Para crear un cuaderno se debe estar en Drive y con un clic derecho aparecen opciones donde una de esas es un cuaderno. Un cuaderno creado en este entorno permite desarrollar proyectos enfocados a Ciencia de Datos o Investigación en Inteligencia Artificial de una manera fácil, por el hecho de que en este entorno de trabajo no requiere ningún tipo de configuración previa. Por defecto trabaja con CPU para compilar código, pero presenta la opción de cambiar la ejecución de CPU a GPU de manera gratuita. La desventaja de usar GPU es que tienes un tiempo limitado de 12 horas para el uso de estas y cuando cumples este tiempo tienes otras 12 o incluso hasta más horas, las cuales no podrás hacer uso de la GPU. Tiene la opción de poder compartir código con otro colaborador y ejecutar código en tiempo real. Puede leer archivos de manera local o en la nube, soporta texto enriquecido y LaTeX. También, se puede instalar, actualizar y regresar versiones de librerías nativas o complementarias de Python. Ahora únicamente soporta versión de Python 3 en adelante [41].

Documentación de Colab: https://colab.research.google.com/notebooks/intro.ipynb

Nota: Para los entrenamientos de los modelos realizados en este trabajo se trabajó con la versión 2.3 de TensorFlow. Las versiones que soporta Colab son: Python 3.7.12 y TensorFlow 2.7.0 [41].

Código para regresar a la versión 2.3 de TensorFlow

- ¡pip install tensorflow==2.3

REFERENCIAS BIBLIOGRAFICAS

[1] D. Aksman *et al.*, “Fraud Detection: Using Artificial Intelligence to Identify Suspicious Persons Over the Phone An Interactive Qualifying Project Report: submitted to the Faculty of the In collaboration with.”

[2] “Fiscalía General del Estado | Fiscalía Informa.” https://www.fiscalia.gob.ec/fiscalia-informa/ (accessed Nov. 07, 2021).

[3] “Ministerio del Interior inauguró moderno Laboratorio de Criminalística y Ciencias Forenses ‘María Eugenia Carrera’, en Quito – Ministerio de Gobierno.” https://www.ministeriodegobierno.gob.ec/ministerio-del-interior-inauguro-moderno-laboratorio-de-criminalistica-y-ciencias-forenses-maria-eugenia-carrera-en-quito/ (accessed Nov. 07, 2021).

[4] B. Salmón Torralbo and others, “Un estudio de lingü\’\istica aplicada: el reconocimiento de voces en el ámbito forense,” 2019.

[5] T.-H. Oh *et al.*, “Speech2Face: Learning the Face Behind a Voice,” May 2019, [Online]. Available: http://arxiv.org/abs/1905.09773

[6] A. Duarte *et al.*, “Wav2Pix: Speech-conditioned Face Generation using Generative Adversarial Networks,” Mar. 2019, [Online]. Available: http://arxiv.org/abs/1903.10195

[7] J. Choe, S. Park, K. Kim, J. H. Park, D. Kim, and H. Shim, “Face Generation for Low-shot Learning using Generative Adversarial Networks.”

[8] Y. Wen, R. Singh, and B. Raj, “Face Reconstruction from Voice using Generative Adversarial Networks.” [Online]. Available: https://github.com/cmu-mlsp/reconstructing_faces_from_voices

[9] S. Singh, “Forensic and automatic speaker recognition system,” *International Journal of Electrical and Computer Engineering*, vol. 8, no. 5, pp. 2804–2811, Oct. 2018, doi: 10.11591/ijece.v8i5.pp.2804-2811.

[10] H. Fraser, “‘Enhancing’ forensic audio: false beliefs and their effect in criminal trials,” *Australian Journal of Forensic Sciences*, vol. 52, no. 2, pp. 165–177, Mar. 2020, doi: 10.1080/00450618.2018.1491115.

REFERENCIAS BIBLIOGRÁFICAS

[11] F. Roldan Snchez, “Speech-conditioned Face Generation with Deep Adversarial Networks,” 2018.

[12] I. Goodfellow, “NIPS 2016 Tutorial: Generative Adversarial Networks,” Dec. 2016, [Online]. Available: http://arxiv.org/abs/1701.00160

[13] S. Minaee, A. Abdolrashidi, H. Su, M. Bennamoun, and D. Zhang, “Biometrics Recognition Using Deep Learning: A Survey,” Nov. 2019, [Online]. Available: http://arxiv.org/abs/1912.00271

[14] “Convolutions: Transposed and Deconvolution | by Mars Xiang | Medium.” https://medium.com/@marsxiang/convolutions-transposed-and-deconvolution-6430c358a5b6 (accessed Nov. 30, 2021).

[15] N. Ketkar, *Deep Learning with Python*. Apress, 2017. doi: 10.1007/978-1-4842-2766-4.

[16] “Learning generative adversarial networks _ next-generation deep learning simplified (PDFDrive)”.

[17] “What Are Activation Functions in Deep Learning? | by David Ben Gurion | Nov, 2021 | Towards Data Science.” https://towardsdatascience.com/what-are-activation-functions-in-deep-learning-cc4f01e1cf5c (accessed Nov. 30, 2021).

[18] I. Goodfellow, Y. Bengio, and A. Courville, “Deep Learning.”

[19] “Auto-Encoder: What Is It? And What Is It Used For? (Part 1) | by Will Badr | Towards Data Science.” https://towardsdatascience.com/auto-encoder-what-is-it-and-what-is-it-used-for-part-1-3e5c6f017726 (accessed Nov. 30, 2021).

[20] “What is TensorFlow? Definition, Benefits, Tutorial | Analytics Vidhya.” https://medium.com/analytics-vidhya/what-is-tensorflow-tensorflow-explained-for-beginners-2021-6ae3c288eea4 (accessed Dec. 01, 2021).

[21] “Introduction to Tensors | TensorFlow Core.” https://www.tensorflow.org/guide/tensor (accessed Dec. 01, 2021).

[22] M. Broughton *et al.*, “TensorFlow Quantum: A Software Framework for Quantum Machine Learning,” Mar. 2020, [Online]. Available: http://arxiv.org/abs/2003.02989

REFERENCIAS BIBLIOGRÁFICAS

[23] T. Carneiro, R. V. M. da Nobrega, T. Nepomuceno, G. bin Bian, V. H. C. de Albuquerque, and P. P. R. Filho, "Performance Analysis of Google Colaboratory as a Tool for Accelerating Deep Learning Applications," *IEEE Access*, vol. 6, pp. 61677–61685, 2018, doi: 10.1109/ACCESS.2018.2874767.

[24] "OpenCV: Face Detection using Haar Cascades." https://docs.opencv.org/3.4.3/d7/d8b/tutorial_py_face_detection.html (accessed Nov. 07, 2021).

[25] J. Wang and L. Perez, "The Effectiveness of Data Augmentation in Image Classification using Deep Learning."

[26] D. S. Park *et al.*, "SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition," Apr. 2019, doi: 10.21437/Interspeech.2019-2680.

[27] T. Ko, V. Peddinti, D. Povey, and S. Khudanpur, "Audio Augmentation for Speech Recognition," 2015. [Online]. Available: http://www.isip.piconepress.com/

[28] A. Chatziagapi *et al.*, "Data augmentation using GANs for speech emotion recognition," in *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, 2019, vol. 2019-September, pp. 171–175. doi: 10.21437/Interspeech.2019-2561.

[29] D. Grado *et al.*, "Identificación de la Fuente de Adquisición de Ficheros Multimedia de Dispositivos Móviles mediante Deep Learning TRABAJO FIN DE GRADO," 2018.

[30] A. Radford, L. Metz, and S. Chintala, "UNSUPERVISED REPRESENTATION LEARNING WITH DEEP CONVOLUTIONAL GENERATIVE ADVERSARIAL NETWORKS."

[31] K. S. Ahmad, A. S. Thosar, J. H. Nirmal, and V. S. Pande, "A unique approach in text independent speaker recognition using MFCC feature sets and probabilistic neural network," Feb. 2015. doi: 10.1109/ICAPR.2015.7050669.

[32] A. Winursito, R. Hidayat, M. Nur, Y. Utomo, and A. Bejo, "Feature Data Reduction of MFCC Using PCA and SVD in Speech Recognition System; Feature Data Reduction of MFCC Using PCA and SVD in Speech Recognition System," 2018.

REFERENCIAS BIBLIOGRÁFICAS

[33] L. Juvela *et al.*, “Speech Waveform Synthesis from MFCC Sequences with Generative Adversarial Networks,” in *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, Sep. 2018, vol. 2018-April, pp. 5679–5683. doi: 10.1109/ICASSP.2018.8461852.

[34] “Welcome to Python.org.” https://www.python.org/ (accessed Nov. 24, 2021).

[35] “Librosa.” https://librosa.org/ (accessed Nov. 24, 2021).

[36] “Matplotlib — Visualization with Python.” https://matplotlib.org/ (accessed Nov. 24, 2021).

[37] “NumPy.” https://numpy.org/ (accessed Nov. 24, 2021).

[38] “Releases - OpenCV.” https://opencv.org/releases/ (accessed Nov. 24, 2021).

[39] “Python Pillow.” https://python-pillow.org/ (accessed Nov. 24, 2021).

[40] “TensorFlow.” https://www.tensorflow.org/ (accessed Nov. 24, 2021).

[41] “Welcome To Colaboratory - Colaboratory.” https://colab.research.google.com/ (accessed Nov. 24, 2021).

[42] “Cloud Storage for Work and Home - Google Drive.” https://www.google.com/drive/ (accessed Nov. 25, 2021).

[43] Y. Dai and G. Wang, “Analyzing Tongue Images Using a Conceptual Alignment Deep Autoencoder,” *IEEE Access*, vol. 6, pp. 5962–5972, Dec. 2017, doi: 10.1109/ACCESS.2017.2788849.

[44] K. Siwek and S. Osowski, “Autoencoder versus PCA in face recognition.”

[45] J. Gabriel and M. Cruz, “Diseño de una red generativa antagónica para el mejoramiento de la resolución de imágenes.”

[46] H. Caballero Hernández, “Cálculo de la dispersión de píxeles en imágenes RGB para Esteganografía con base en la Teoría Fractal,” 2020.