ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

GENERACIÓN DE SEÑALES VOLCÁNICAS ARTIFICIALES DEL TIPO VT (VOLCANO-TECTONIC) Y LP (LONG PERIOD) USANDO UN MODELO CGAN (CONDITIONAL GENERATIVE ADVERSARIAL NETWORK)

TRABAJO DE INTEGRACIÓN CURRICULAR PRESENTADO COMO REQUISITO PARA LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN TECNOLOGÍAS DE LA INFORMACIÓN

CHRISTIAN MAURICIO YÉPEZ ESCALANTE
christian.yepez@epn.edu.ec

DIRECTOR: Ph.D. FELIPE LEONEL GRIJALVA ARÉVALO
felipe.grijalva@epn.edu.ec

Quito, septiembre 2022

CERTIFICACIONES

Yo, CHRISTIAN MAURICIO YÉPEZ ESCALANTE declaro que el trabajo de integración curricular aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

CHRISTIAN MAURICIO YÉPEZ ESCALANTE

Certifico que el presente trabajo de integración curricular fue desarrollado por CHRISTIAN MAURICIO YÉPEZ ESCALANTE, bajo mi supervisión.

Ph.D. FELIPE LEONEL GRIJALVA ARÉVALO
DIRECTOR

DECLARACIÓN DE AUTORÍA

A través de la presente declaración, afirmamos que el trabajo de integración curricular aquí descrito, así como el (los) producto(s) resultante(s) del mismo, son públicos y estarán a disposición de la comunidad a través del repositorio institucional de la Escuela Politécnica Nacional; sin embargo, la titularidad de los derechos patrimoniales nos corresponde a los autores que hemos contribuido en el desarrollo del presente trabajo; observando para el efecto las disposiciones establecidas por el órgano competente en propiedad intelectual, la normativa interna y demás normas.

CHRISTIAN MAURICIO YÉPEZ ESCALANTE

Ph.D. FELIPE LEONEL GRIJALVA ARÉVALO

DEDICATORIA

Dedico este trabajo con mucho cariño a mis padres José y Cecibel y a mi hermano Alex, gracias por el apoyo que me han brindado toda mi vida.

Christian Yépez

AGRADECIMIENTOS

Agradezco a todas las personas que a lo largo de la carrera han mostrado apoyo, consejos y ayuda. Gracias a estas personas he alcanzado esta meta.

A mis padres por todo el apoyo, paciencia y dedicación. Gracias a ellos por convertirme enseñarme sus valores y principios, gracias a ellos soy la persona que ahora soy.

A mi hermano por la ayuda que me ha dado y el soporte que me ha brindado.

A mis amigos Rodrigo, Daniel, Abi, Mayelin, Jenni, Lili los cuales siempre me han brindado sus consejos y su amistad incondicional.

A mi tutor el doctor Felipe Grijalva el cual me ha guiado en la realización de este proyecto, gracias por la paciencia y el apoyo.

A todos los profesores que he tenido durante mi vida que me han transmitido su conocimiento.

Finalmente, quiero agradecer a la Escuela Politécnica Nacional por darme la oportunidad de estudiar en sus aulas donde conocí genta maravillosa y tuve experiencias increíbles.

ÍNDICE DE CONTENIDO

RESUMEN

Ecuador es un país que cuenta con muchos volcanes activos, entre ellos el Cotopaxi, cuya erupción podría ocasionar daños irreparables como la pérdida de vidas humanas, así como también problemas ambientales, económicos y sociales entre otros; por ello el estudio de señales sísmicas volcánicas para la prevención es de suma importancia, siendo una limitante la cantidad de datos originales que se tienen.

El presente trabajo utiliza un modelo cGAN para generar nuevas señales sísmicas volcánicas del tipo VT y LP. Este modelo fue desarrollado en la librería Pytorch para Python y se entrenó con la base de datos pública del IGEPN. Las señales generadas fueron comparadas con las señales originales mediante la métrica FID, llegando a tener un valor aceptable para este tipo de señales y siendo estas similares a las originales tanto en el dominio del tiempo como en el de la frecuencia.

PALABRAS CLAVE: Aprendizaje automático, Redes Generativas, GAN, cGAN, Señales Sísmicas Volcánicas.

ABSTRACT

Ecuador is a country that has many active volcanoes, including Cotopaxi, whose eruption could cause irreparable damage such as loss of human life, as well as environmental, economic and social problems, among others; For this reason, the study of volcanic seismic signals for prevention is of the utmost importance, but the amount of original data available is a limitation.

The present work uses a cGAN model to generate new volcanic seismic signals of the type VT and LP. This model was developed in the Pytorch library for Python and was trained with the IGEPN public database. The generated signals were compared with the original signals using the FID metric, reaching an acceptable value for this type of signals and being similar to the original ones both in the time domain and in the frequency domain.

KEYWORDS: Machine Learning, Generative Networks, GAN, cGAN, Volcanic Seismic Signals.

1. INTRODUCCIÓN

El estudio de las señales percibidas de la actividad sísmica y micro-sísmica de un volcán es de gran importancia, ya que, a través del análisis de éstas, se podría detectar y predecir a tiempo posibles desastres naturales. En estas señales se reflejan la actividad interna del volcán, como lo son el movimiento de fluidos y la interacción de estos con roca sólida [1]. Debido a la gran cantidad de datos que se extraen del monitoreo de los volcanes un análisis clásico ya no es efectivo y por eso se requieren de métodos que puedan analizar o procesar gran cantidad de datos a la vez para sacar todo el provecho que se pueda de estos.

En el país el Instituto Geofísico de la Escuela Politécnica Nacional (IGEPN) es el encargado de la monitorización y recolección de datos micro-sísmicos de los distintos volcanes del Ecuador.

Uno de los volcanes al cual el IGEPN monitoriza con más ahínco es el volcán Cotopaxi debido a que presenta actualmente actividad y en el supuesto caso de una erupción causaría en primer lugar lahares debido a que el volcán tiene una cobertura glaciar. Estos lahares podrían presentar un riesgo a poblaciones aledañas. Además, debido a la ubicación geográfica del volcán en el centro del país podría ocasionar problemas socioeconómicos en caso de erupción [2, 3].

Los sensores utilizados por el IGEPN en la recolección de los datos del volcán Cotopaxi registran varios tipos de eventos micro-sísmicos. Los que son más comunes e importantes para su estudio son los siguientes:

- Sismo VT (Volcano-Tectonic)
- Sismo LP (Long Period)

El flujo de datos recolectado por los sensores que monitorean el volcán Cotopaxi del IGEPN son almacenados y procesados de la siguiente forma. Primero se realiza la detección de un evento sísmico de una forma semiautomática [4]. Luego la clasificación, es decir determinar qué tipo de evento microsísmico se trata, de los eventos detectados, lo realiza manualmente un experto en vulcanología [5]. El análisis y clasificación de las señales obtenidas de los eventos micro-sísmicos en la actualidad lo realizan los especialistas vulcanólogos de manera manual o semiautomática lo cual demanda mucho tiempo. Además, las señales obtenidas de los sismógrafos deben ser procesadas con antelación para su estudio, incrementando aún más el tiempo.

Esta actividad toma una cantidad considerable de tiempo, además de que requiere un experto en vulcanología para analizar los datos, lo cual limita el procesamiento y estudio de las señales o la detección temprana de posibles desastres naturales [2].

Por esta razón la necesidad de algoritmos de Aprendizaje de Máquina que permitan automatizar todo este proceso de clasificación, y por ello conseguir una predicción más rápida de cualquier desastre natural.

Es posible con las arquitecturas de aprendizaje de máquina disponibles crear un buen clasificador de estas señales micro-sísmicas pero para ello se necesitan gran cantidad de datos para entrenar estas arquitecturas, al menos en el orden de las mil observaciones [6]. El problema consiste en que en este caso no se dispone de una base de datos tan amplia ya que a pesar de que se tiene una gran cantidad de datos sin procesar, los eventos que son de interés no ocurren con frecuencia y por ello se genera un desbalance en los datos de entrada que afectan a un sistema de clasificación que se intente hacer con estos datos.

En este contexto, el problema de tener una base de datos con pocos eventos micro-sísmicos de interés se puede solucionar mediante la generación de las señales artificiales. Para ello se implementará una arquitectura de aprendizaje de máquina llamada red generativa adversaria condicional (cGAN) [7] a la cual se la entrena con las pocas señales reales que disponemos y puede ser capaz tanto de generar eventos micro-sísmicos VT cómo LP.

Generar nuevos eventos permitirá crear nuevas arquitecturas de aprendizaje de máquina que ayudará al IGEPN a la detección, clasificación y estudio de los distintos eventos micro-sísmicos volcánicos. Con lo cual los distintos expertos del IGEPN podrían dedicarse a otras investigaciones y dejar el procesamiento manual de estas señales. Finalmente, estos algoritmos podrían ayudar, a expertos en vulcanología, a la detección temprana de erupciones volcánicas y ejecutar un plan de contingencia oportuno. En caso de persistir el problema los investigadores vulcanológicos deberán seguir clasificando las señales micro-sísmicas vulcanológicas manualmente y de esta manera no podrían dedicarse a tiempo completo un estudio más detallado de las señales ya clasificadas.

1.1. Objetivo general

Generar señales volcánicas artificiales del tipo VT (Volcano-Tectonic) y LP (Long Period) usando un modelo cGAN (Conditional Generative Adversarial Network).

1.2. Objetivos específicos

- Analizar los modelos de aprendizaje automático relacionadas con la generación de señales.
- Analizar las características de las señales micro-sísmicas volcánicas y sus tipos.
- Entrenar una arquitectura cGAN (Red generativa adversaria condicional) para generar señales micro-sísmicas a partir de una base de datos del volcán Cotopaxi.
- Evaluar la calidad de las señales generadas a través de un algoritmo evaluador conocido como Fréchet Distance (FD) que compare las señales generadas con las reales.

1.3. Alcance

Se diseñará y entregará una arquitectura conocida como cGAN(Red generativa adversaria condicional), esta arquitectura, mostrada en la figura 1.1, de redes neuronales está conformada por 2 redes compitiendo entre ellas en un juego de suma cero. La primera red llamada red generadora se encarga de generar una señal nueva. La segunda red neuronal se llama red discriminadora y se encarga de determinar si una señal de entrada es un ejemplo real o si esta es generada por la red generadora. Finalmente, a cada una de estas redes se le ingresa una etiqueta externa que contiene información del tipo de señal micro-sísmica y distinguirá entre los dos eventos (VT y LP). De esta manera una única arquitectura generará ambos tipos de eventos. [7]

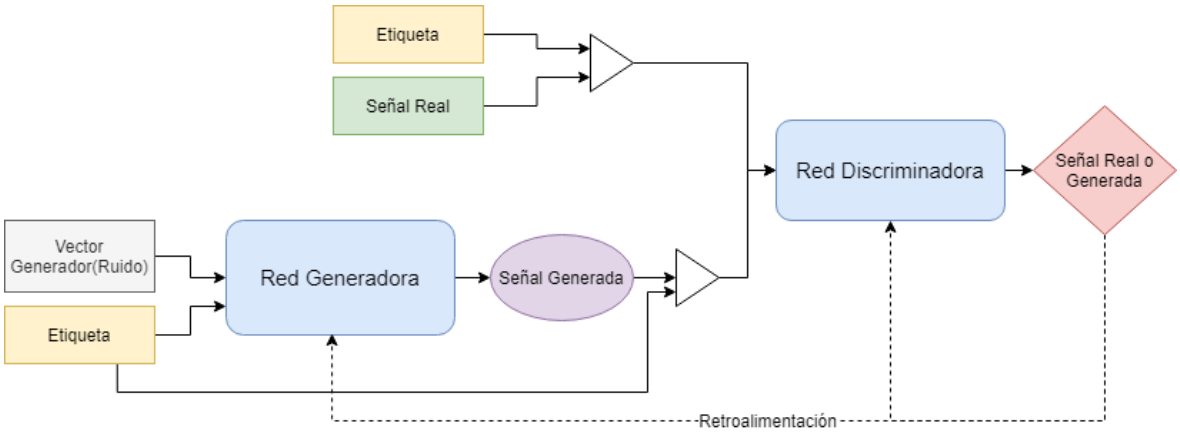

Figura 1.1: Arquitectura de un modelo cGAN.

El entrenamiento de esta arquitectura consiste en pasar por la red discriminadora alternadamente tanto señales reales como generadas, intentando la red generadora engañar a

la discriminadora. Al tener esta 'competencia' el modelo convergerá a un punto en el que pueda generar señales sísmicas de las mismas características de una real del conjunto de datos del entrenamiento.

La arquitectura cGAN será desarrollada en el lenguaje de programación Python con la biblioteca Pytorch. Esta red generará nuevos ejemplos de eventos micro-sísmicos volcánicos del tipo VT (Volcano Tectonic) y LP (Long Period) y se entrenará con una base de datos pública del volcán Cotopaxi disponible en el trabajo "ESeismic: Towards an Ecuadorian volcano seismic repository" [8].

Al construir y evaluar el modelo se lo hará con los siguientes elementos:

- **Base de datos:** Para entrenar estos modelos se utilizará la base de datos pública proporcionada por el IGEPN la cual fue recopilada y procesada en el trabajo "ESeismic: Towards an Ecuadorian volcano seismic repository" [8]. La base consta de 4 tipos de eventos, pero solo se tomará en cuenta los que corresponden a eventos VT y LP.

- **Entorno y lenguaje de programación:** El lenguaje de programación que se utilizara en el desarrollo del proyecto es Python, en particular se utilizará la biblioteca de software libre Pytorch. La cual está diseñada para ayudar a crear arquitecturas de aprendizaje automático [9].

  En lo referente al entorno de programación, debido al alto uso computacional que supone entrenar y optimizar una arquitectura cGAN se utilizará la herramienta Google Colaborative que permite disponer de un alto poder computacional en una máquina virtual para ejecutar código de Python. Esta herramienta tiene un plan sin costo que cubre las necesidades de este proyecto.

- **Modelo de aprendizaje de máquina a usar:** El modelo que se empleará en el presente proyecto es el de cGAN(Red generativa adversaria condicional). Este modelo se compone de dos redes: una generadora que creará nuevas señales, y una discriminadora que determinará si las señales son reales o generadas. Al entrenarlas estas compiten y se mejoran entre sí hasta que la red generadora sea capaz de generar señales similares a las reales. Además, a ambas redes se les dará una entrada adicional externa conocida como etiqueta que servirá para determinar qué tipo de señal deberá generar [7].

- **Características de las señales y evaluación del modelo:** Las señales generadas serán similares a las reales tanto en el dominio del tiempo como en el de la frecuencia,

debido a que hay información importante en cada uno de estos dominios. Para ello se utilizará un algoritmo evaluador conocido como Fréchet Distance (FD) que compara la calidad de las señales generadas en relación con las señales originales [10].

Este trabajo tendrá como producto final demostrable el modelo de la arquitectura cGAN entrenada para generar las señales. Además, se entregará el script en Python de la implementación del modelo. Cabe mencionar que este trabajo se concentra en el entrenamiento del modelo cGAN por lo que no contempla el desarrollo de una interfaz gráfica de usuario.

1.4. Marco teórico

1.4.1. Sismología Volcánica

El estudio de señales micro-sisimicas volcánicas es de vital importancia ya que del análisis de estas es posible entender el comportamiento interno de un volcán y de esta manera estar atento a posibles erupciones volcánicas que pueden prevenir desastres naturales [2, 3]. El proceso eruptivo de un volcán siempre viene acompañado por estas señales.

Estos sismos tienen una magnitud debajo de la que un ser humano es capaz de sentir y por ello es necesario el uso de sismógrafos para detectarlos. El análisis de estas señales debe ser hecho con la mayor brevedad posible ya que con ello se tiene una mejor respuesta a cualquier evento o desastre natural.

A estas señales se le atribuye a cada una un evento interno asociado en el volcán estos pueden ser rompimientos de roca volcánica, movimiento de fluidos volcánicos, etc. Las señales de estos eventos asociados siempre son similares ya sea en su frecuencia típica, sus picos o sus duraciones y en base a estos atributos a las señales micro-sismicas volcánicas se las clasifica de la siguiente manera:

- Eventos de periodo largo (*Long-Period*, LP)
- Terremotos volcano-tectónicos (*Volcano-Tectonic*, VT)
- Eventos híbridos (*Hybrid*, HB)
- Terremotos de hielo (*Ice Quake*, ICE)
- Eventos regionales (*Regional*, REG)

Esta clasificación puede ser variada de acuerdo con la bibliografía seleccionada, siendo la mencionada la mas comun, ya que la teoría de la sismología volcánica es una ciencia que crese en base a las observaciones de eventos y a la experiencia acumulada de los expertos.

En el presente trabajo nos centramos en la generación de los eventos más comunes y relevantes para el estudio de la sismología volcánica, estos son VT(Volcano-Tectonic) y LP (Long Period).

1.4.1.1. Microsismo volcánico del tipo VT(Volcano-Tectonic)

Estos eventos se relacionan con la rotura de la roca como respuesta del cambio de la tensión del magma producida por el movimiento de este. Esta rotura se caracteriza por señales de frecuencias entre 5 y 15 Hz con picos de energía entre los 6 y 8 Hz. Generalmente ocurren varios eventos de estos en un corto lapso de tiempo [11].

1.4.1.2. Microsismo volcánico del tipo LP(Long Period)

Los eventos LP son atribuidos a una resonancia de grietas o conductos llenos de líquido producidas por el cambio de presión del fluido. Estos al estar relacionados con la presión del magma del volcán son un buen indicador de una posible erupción. Las frecuencias de estos eventos son de hasta 5 Hz y su energía se concentra entre los 1.2 y 2.5 Hz que puede durar hasta 20 segundos. Este evento es más común que el evento VT [11].

1.4.1.3. Volcan Cotopaxi

El Cotopaxi es un volcán activo que se ubica sobre la Cordillera Real de los Andes del Ecuador (Latitud 0°40’ Sur; Longitud 78°26’ Oeste; 5 897 msnm), a 45 km al sureste de Quito [12]. Este es uno de los volcanes más vigilados del Ecuador. Esto es debido a que ya ha presentado en su historia 5 procesos eruptivos [13] y este al ser un glaciar presenta además el riesgo de la formación de lahares [2]. Este volcán es monitoreado por el Instituto Geofísico de la Escuela Politécnica Nacional (IGEPN) [12].

El IGEPN tiene instalados diferentes sensores en las cercanías del volcán como se puede ver en la figura 1.2. Este gran monitoreo es de vital importancia ya que puede darse un fuerte evento eruptivo y un sistema de alertas tempranas puede ser crucial para minimizar los posibles daños.

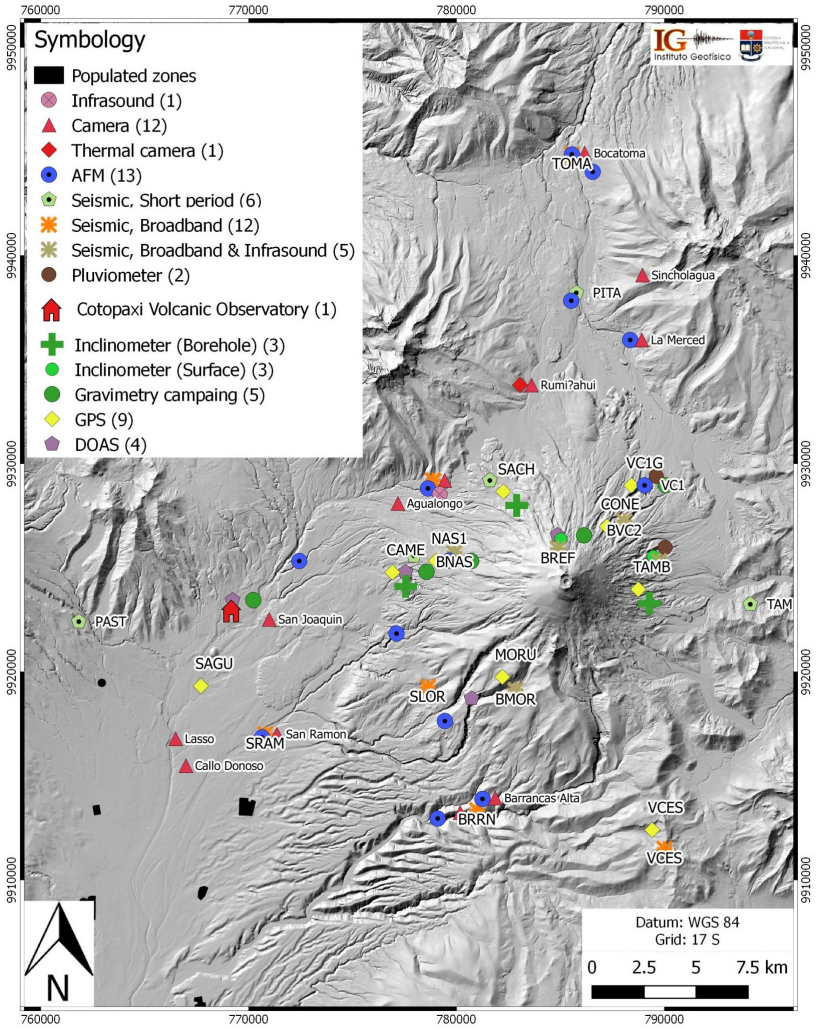

Figura 1.2: Red de Monitoreo del volcán Cotopaxi. Fuente [14]

1.4.2. Aprendizaje automático

El aprendizaje automático es una rama de la Inteligencia artificial que tiene como objetivo que la maquina aprenda las características y patrones de un conjunto de datos que se le da como entrada sin estas ser programadas explícitamente. A diferencia de la IA tradicional este no pretende en base a algoritmos emular el comportamiento inteligente, sino a través del procesamiento de los datos el computador encuentre la función que relacione la entrada y salida.

El aprendizaje automático es usado principalmente para, una vez entrenado el modelo, en base a una entrada de datos no vista anteriormente por el modelo predecir la salida. Entre sus principales aplicaciones se encuentran diagnósticos médicos, análisis de mercado para los diferentes sectores de actividad, clasificación de secuencias de ADN, reconocimiento del habla y del lenguaje escrito, etc.

El aprendizaje automático a grandes rasgos se lo puede clasificar en aprendizaje supervisado y aprendizaje no supervisado.

1.4.2.1. Aprendizaje automático supervisado

El aprendizaje automático supervisado es usado en el caso de que se tenga claramente una pareja de datos una correspondiente a la entrada de datos al modelo, y la salida esperada. El aprendizaje o entrenamiento del modelo es de la siguiente manera: al modelo se le ingresan los datos de entrada y este produce unos datos de salida, estos datos se los compara con la salida esperada y se penaliza al modelo en caso de haber dado una salida incorrecta, a través de muchas iteraciones el modelo será capaz de dar una salida muy cercana a la salida real esperada y es cuando el modelo se lo considerara entrenado.

En este caso si al modelo se le da una entrada X se espera que este nos dé una salida $Y = f(X)$ donde f seria la función que el modelo tiene que aprender a través de los datos entregados. Generalmente estos modelos se los usa para realizar problemas de regresión o de clasificación. Por ejemplo, se puede entrenar un modelo capas de clasificar imágenes de distintos animales, en este caso los datos de entrada X serán la foto de los animales las cuales estarían previamente etiquetadas con su correspondiente especie que correspondería a la salida Y esperada.

1.4.2.2. Aprendizaje automático no supervisado

En el aprendizaje no supervisado al algoritmo se lo entrena con un conjunto de datos no etiquetado. En cambio, estos datos tienen una gran cantidad de información en la estructura de los mismos. En este caso el computador deberá descubrir y representar esta estructura. Generalmente en este apartado entran problemas de clasificación y asociación, por ejemplo, agrupar distintos tipos de señales en base a sus características.

1.4.3. Aprendizaje profundo

El aprendizaje profundo basa su funcionamiento en las conocidas como redes neuronales para su funcionamiento, estas redes neuronales tienen como objetivo emular el comportamiento del cerebro conectando gran cantidad de neuronas o perceptrones. En los últimos años debido a él gran aumento de la capacidad de procesamiento en las computadoras, en especial en el área de GPU, este campo ha tenido una gran acogida en temas como como la visión por ordenador, el reconocimiento de voz, la traducción automática y la robótica. Una de las limitantes de estos nuevos modelos es que para entrenarlos estos necesitan una gran cantidad de datos.

1.4.3.1. Neurona Artificial

Una neurona artificial es la unidad de procesamiento de la red neuronal. Se compone de varias entradas de datos a las cuales se les realiza una suma ponderada para luego pasarlas a una función de activación y finalmente este valor correspondería a su salida [15]. Los pesos de cada una de estas entradas corresponden a los parámetros que se actualizarán durante el entrenamiento y de los cuales depende el funcionamiento del modelo. Una representación de una neurona artificial se la puede ver en la figura 1.3.

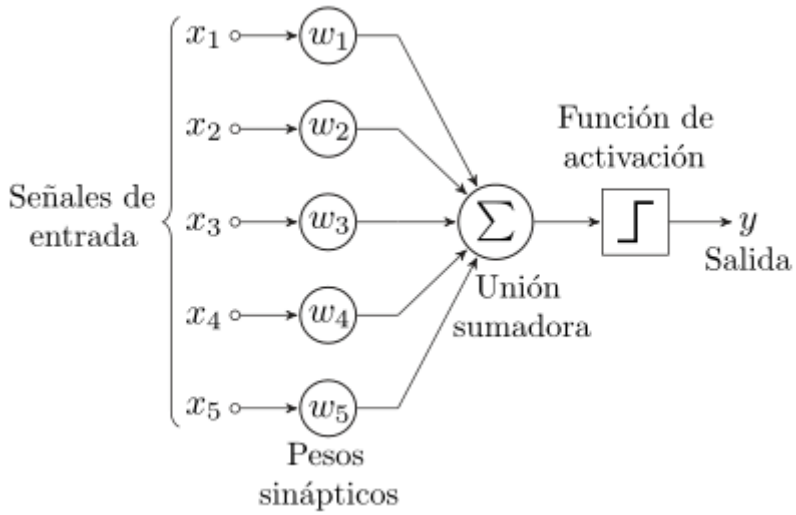

Figura 1.3: Neurona artificial

Con la función de activación se busca limitar el rango de salida de la red neuronal en cierto rango, además de que para que esta sea usada en una red estas funciones deberían ser no lineales ya que en caso de serlo toda la red podría ser resumida en una sola neurona. En la figura 1.4 se pueden ver las principales funciones de activación utilizadas.

1.4.3.2. Redes neuronales

Una red neuronal es un conjunto de neuronas conectadas entre si de manera que esta tenga varias capas. Estas son el componente fundamental de cualquier modelo de Aprendizaje profundo. También son llamadas Perceptrón multicapa. Incrementar el numero de capas de una red podría ayudar a mejorar el rendimiento de esta, así como también el número de sus parámetros entrañables. Además, una red más grande podría aumentar significativamente el costo computacional del entrenamiento y podría generar problemas como el overfiting en el cual se diría que una red esta sobreentrenada [15].

Una red neuronal tiene 3 tipos de capas siendo estas las de entrada que correspondería a

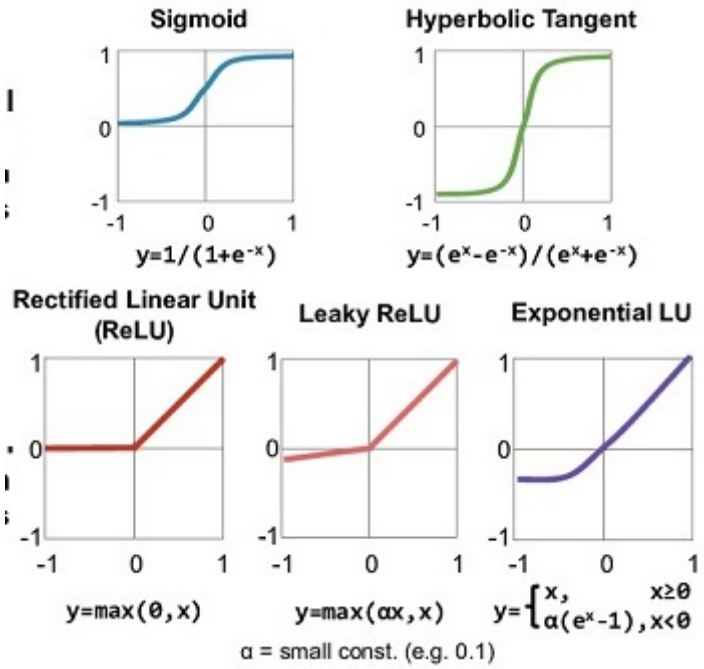

Figura 1.4: Funciones de activación

los datos ingresados, las capas ocultas las cuales realizan el procesamiento de los datos y la capa de salida que corresponde a el resultado obtenido por la red, este modelo se lo aprecia en la figura 1.5.

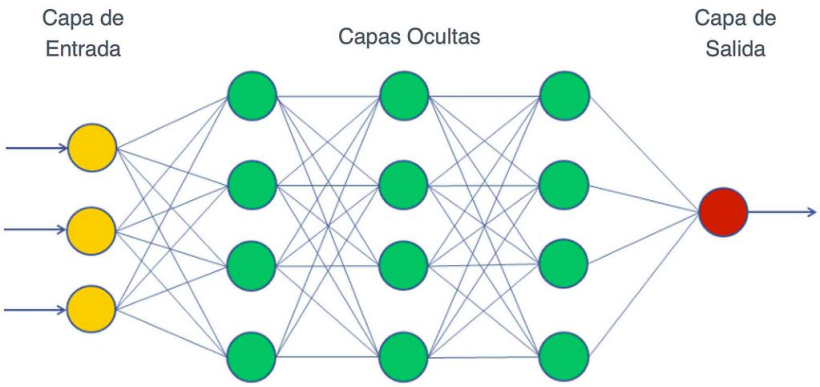

Figura 1.5: Estructura de una red neuronal

1.4.3.3. Redes neuronales convolucionales

Una red neuronal convolucional es un tipo de red que intenta imitar el funcionamiento abstracto de cerebro mediante la introducción de convoluciones en sus capas. En esta cada capa procesa cada parte de los datos introducidos abstrayendo de estos una característica de manera que mientras más nos adentramos en una de estas redes menos abstracta será la característica que pueda detectar. Cada característica estará representada por un filtro

que será el elemento con el cual se operará para la convolución de la salida de la capa anterior o de la entrada de datos. Este filtro es el que deberá entrenarse para poder captar estas características independientemente de la ubicación de los datos [16].

Una convolución es una operación lineal entre la entrada a la capa y el filtro correspondiente que se muestra en la figura 1.6. Este filtro recorrerá el conjunto de datos desplazando a través de esta a través de un parámetro de nuestra elección. Además de esta operación el conjunto de datos se agrandará o reducirá utilizando varias algoritmos con el objetivo de adaptar estas salidas de las convoluciones a la arquitectura de la red.

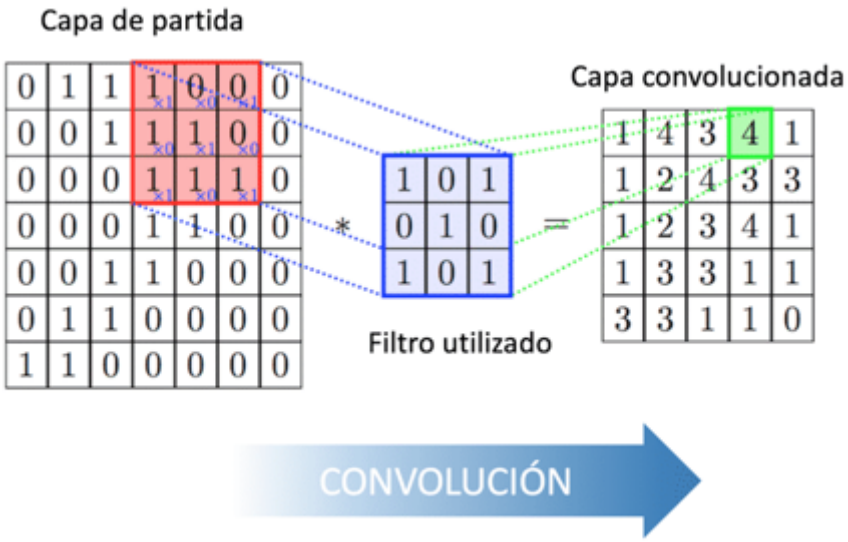

Figura 1.6: Funcionamiento de una convolución

Una red neuronal convolucional puede incluir capas tradicionales en su arquitectura para mejora el desempeño de esta. Normalmente la aplicación principal de estas es en el área de la visión artificial y en general el procesamiento de imágenes.

1.4.4. Modelos Generativos

Un modelo generativo tiene como objetivo a partir de ruido generar una distribución de datos coherente. Estos buscan captar la distribución y características de los datos que le son dados en el proceso de entrenamiento para luego generar datos nuevos no vistos durante el entrenamiento pero similares a estos. Dos arquitecturas profundas son los principales pilares para desarrollar modelos generativos: Variational Autoencoders (VAEs) y Generative Adversarial Networks (GANs).

1.4.4.1. Red Generativa Adversaria (GAN)

Una red antagónica generativa (GAN) es un modelo de aprendizaje automático en el que dos redes neuronales compiten entre sí para entre las 2 lograr el objetivo de generar nuevos conjuntos de datos. Las GAN generalmente se ejecutan sin supervisión y utilizan un marco de juego cooperativo de suma cero para aprender.

La GAN se componen de 2 redes la primara es el generador la cual se encarga de generar el nuevo conjunto de datos y la segunda es el discriminador la cual intenta determinar si los datos que pasan a través de ella son creados por el generador o si en cambio son datos reales. Estas 2 redes compiten entre si hasta que el generador pueda crear datos nuevos muy parecidos a los reales. Las redes que componen la GAN están compitiendo entre ellas en un juego de suma cero [17].

Es importante mencionar que el generador nunca ve los datos de entrenamientos reales únicamente recibe información de que si lo que esta genero engaño o no al discriminador [18]. La arquitectura de la red GAN se la puede ver en la figura 1.7.

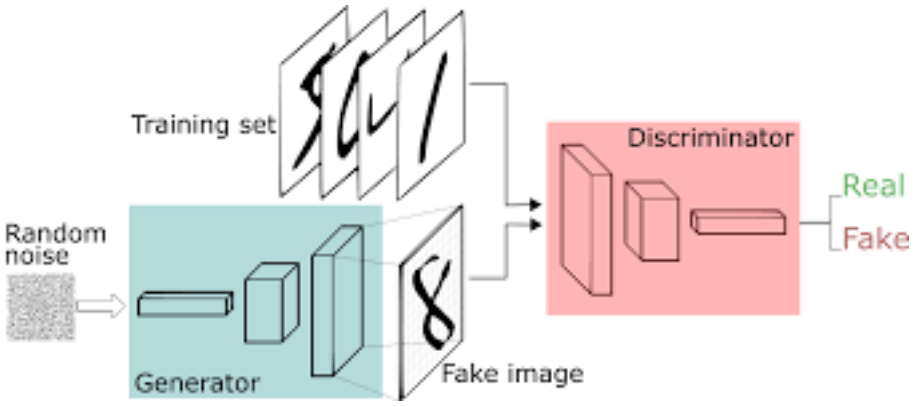

Figura 1.7: Modelo GAN

1.4.4.2. Red Generativa Adversaria condicional (cGAN)

Una cGAN es una red similar a la GAN con la diferencia que, a cada red, el discriminador y el generador se les alimenta con un parámetro adicional llamado etiquetas que representa una clase, el objetivo es que a través de este parámetro se pueda controlar la generación y generar datos de distintas clases [7].

Esta es el modelo elegido para el presente trabajo debido a que se desea generar señales volcánicas eligiendo su clase, es decir si esta es VT o LP. La arquitectura de la red cGAN se la puede ver en la figura 1.1.

1.4.5. Herramientas utilizadas

Para el presente proyecto se utilizaron principalmente las siguientes herramientas, lenguajes de programación o bibliotecas:

- Matlab es una plataforma de programación que se enfoca principalmente en el trabajo con matrices y el procesamiento de datos, debido a esto este software fue elegido para implementar la primera parte de filtrado de la base de datos.
- Python es un lenguaje de programación interpretado multiparadigma que se enfatiza en la legibilidad del código, fue creado en 1991 y tiene ya 3 versiones, se eligió este lenguaje para el presente proyecto debido a que tiene varias bibliotecas para el procesamiento de señales, así como para el aprendizaje automático.
- Pytorch es un framework de aprendizaje automático libre para Python mantenido por Meta (Facebook) tiene un gran número de funcionalidades como la diferenciación automática y el manejo de tensores, además de tener implementadas las principales funciones de coste y optimización utilizadas en redes GAN y cGAN razón por la cual se lo eligió para el presente proyecto.
- Jupyter Notebook es un entorno informático para la web en las cuales se pueden crear cuadernos de trabajo en los que se ejecute código escrito en Python de manera estructurada y pudiendo alternarlo con texto plano, debido a la facilidad de ejecutar código y visualizar los resultados de este en tiempo real se eligió esta plataforma para el desarrollo del script principal del proyecto.
- Google Colaborative es una plataforma de Google en la que se dispone de una capacidad de cómputo necesaria para la ejecución del código del proyecto, en esta plataforma es posible ejecutar un cuaderno de Jupyter de manera gratuita y hacer uso de la capacidad de procesamiento ya sea de CPU o GPU.

2. METODOLOGÍA

En la presente sección se describen los procedimientos para generar las señales volcánicas. Inicialmente, se realizó un pre-procesamiento de señales en el dominio del tiempo (Sección 2.2), esto con el objetivo de estandarizar las señales de manera que tengan una misma longitud en tiempo y un mismo rango en amplitud, además, descartar las señales más ruidosas que pueden interferir en el entrenamiento de la red neuronal.

Posteriormente, se obtuvo el espectrograma de cada una de las imágenes que están en el dominio de la frecuencia, al realizar este procedimiento obtuvimos una imagen bidimensional de 129 x 33 con un solo canal que representa el espectro en magnitud de la señal original, a esta imagen resultante se la volvió a normalizar para tener una entrada estándar.

Con estas señales, se procedió a entrenar el modelo de cGAN (Sección 2.3 y y 2.4) el cual generaría espectrogramas de magnitud falsos, las cuales para su uso es necesario regresarlos al dominio del tiempo, lo que se logra mediante el uso de el algoritmo de Griffin-Lim (Sección 2.5) que nos permite reconstruir una señal a través de únicamente su espectrograma en magnitud, posteriormente los resultados fueron evaluados mediante el uso de FID (Sección 2.6).

Figura 2.1: Proceso de generación de señales volcánicas

Este proceso se resume en la Figura 2.1 y se describirá más ampliamente en las siguientes secciones.

2.1. Base de Datos

Para el presente proyecto se utilizo la base de datos de MicSigV1 disponible públicamente en el repositorio Eseismic [1] que se compone de eventos volcánicos sin procesar de dos estaciones (VC1 y BREF) [8]. Para lo cual primeramente se filtró las señales de tipo VT y LP de los detectores, esta base tiene varios tipos de señal como se muestra en la figura 2.2.

El proceso de generación de nuevas señales volcánicas VT y LP mediante la arquitectura cGAN empieza por el filtrado de las señales de esta base y adaptándola a un formato más

[1] disponible en https://www.igepn.edu.ec/eseismic_web_site/index.php

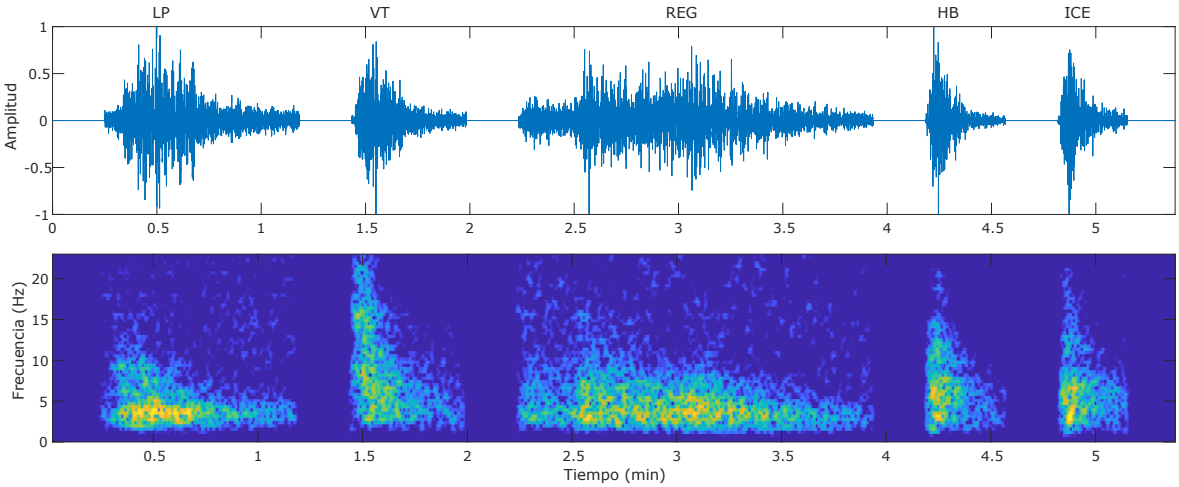

Figura 2.2: Eventos volcánicos registrados en el volcán Cotopaxi

usable por nuestro módulo de pre procesamiento. El filtrado se lo realizó en MATLAB.

2.2. Pre-procesamiento

En el pre-procesamiento el objetivo es que todas las señales tengan características comunes en el dominio del tiempo, para poder entrenar el modelo cGAN. En primer lugar se realizó una normalización en el número de muestras y en la frecuencia de muestreo de las diferentes señales. Por ello se normalizó las señales a 4096 muestras por señal con una frecuencia de muestreo de 50 Hz, esto se logró descartando señales en las que el evento sísmico tuvo una mayor duración, las señales cuyo evento sísmico fue menor se las relleno con ruido o se conservó el ruido propio de la señal original. Con ello la duración de todas las señales que se usaron para el entrenamiento de la cGAN fue de 81.92 segundos, teniendo el evento sísmico al inicio de la muestra y ruido al final de esta.

Posteriormente ya con unas señales de una longitud en tiempo estandarizadas, se procede a quitar el valor dc de las señales que corresponde al valor medio de estas, esto con el objetivo de estandarizar más las señales.

Luego se realizó un proceso para eliminar las señales ruidosas, calculando el SNR de la señal con la siguiente ecuación:

$$SNR = 10log_{10}(P_{senial}/P_{ruido}) \tag{2.1}$$

En donde se consideró que la potencia de la señal es la potencia del evento sísmico y la potencia del ruido fue calculado con el ruido integrado al final de la señal.

Se descartaron las señales que superaron un límite de señal de ruido de 10, depurando la base de datos para que no tenga señales muy ruidosas que impidan distinguir del ruido propio de la señal y que causen problemas al modelo cGAN al momento de aprender a generarlas, el valor límite del SNR se lo consiguió mediante la visualización de las señales filtradas y teniendo en cuenta que no se reduzca de una manera drástica el tamaño del dataset.

Finalmente, con el motivo de tener señales estandarizadas, se procedió a normalizar la amplitud de las muestras en el dominio del tiempo entre -1 y 1 se esta manera las señales de mucha potencia o de baja potencia no serían un problema al momento de la generación [19]. Debido a que las formas originales de la señal en el dominio del tiempo podrían dar problemas al momento de la generación, seguimos un enfoque similar al seguido por los trabajos de MelNet y MelGAN [20, 21] los cuales evitaron generar en el dominio del tiempo. En la figura 2.3 se puede ver cómo queda una señal antes y después del proceso de preprocesamiento.

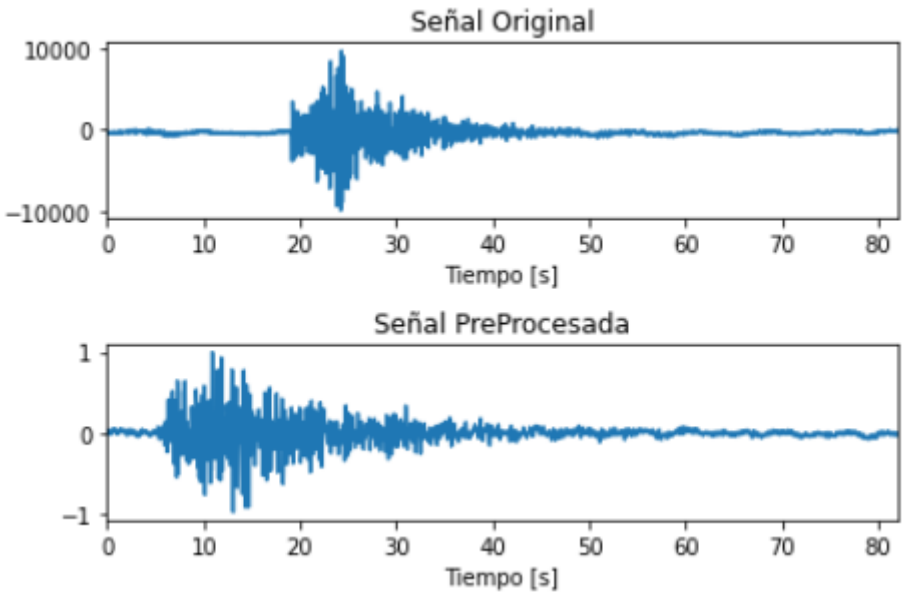

Figura 2.3: Señales antes y después proprocesamiento

El último paso del pre-procesamiento fue pasar las señales del dominio del tiempo al dominio de la frecuencia, esto se lo logró con la Transformada de Fourier en tiempo reducido o espectrograma, siendo la longitud de cada segmento de 256 muestras sin overlap entre los segmentos, de esta manera en la salida tendríamos un espectrograma con un tamaño de 129 * 33 del cual únicamente tomamos su magnitud. Esto procedimiento se lo realizó sobre todas las señales de la base de datos, que hayan pasado por el proceso de filtrado y pre-

procesamiento y los espectrogramas se los guardó en una nueva base de datos en formato h5 el cual nos ayudó a cargar y entrenar al modelo de una manera dinámica.

De esta manera a la red cGAN únicamente entraría una especie de “imagen” de una sola dimensión correspondiente al espectrograma de la nueva señal generada. El esquema del pre-procesamiento se lo resumen en la Figura 2.4.

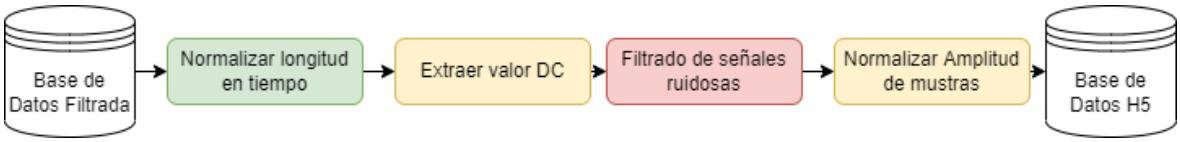

Figura 2.4: Preprocesamiento de las señales volcánicas

2.3. Diseño del Modelo

La arquitectura de la red neuronal usada para el presente trabajo se desarrolló con base al modelo *Deep Convolutional Generative Adversarial Networks* (DCGAN) [19] que es usada para la generación de imágenes, además de añadir un parámetro para condicionalmente controlar la generación que describe la red *Conditional Generative Adversarial Nets* (cGAN) [7]. A estos modelos se los combinó y modificó de tal manera que se puedan generar espectrogramas unidimensionales de un tamaño de 133 x 33, además el parámetro condicional de la red corresponda a la clase deseada de la señal volcánica que se desea generar, en este caso VT o LP.

El discriminador se lo formó con capas convolucionales bidimensionales. La reducción de la dimensionalidad se la realiza con el método sugerido por Radford et al. [19] en el cual en lugar de usar el enfoque tradicional de usar capas de max pooling, éste se logra mediante el uso de convoluciones escalonadas (strided convolutions) que se van reduciendo del tamaño de la imagen ingresada a la par que el número de filtros se aumenta progresivamente. De esta manera al final se tiene únicamente un vector de filtros de tamaño 1*1 que se lo pasa por una última capa que reduce el número de filtros a 1, el cual correspondería a la probabilidad de que la imagen pasada por este sea una imagen real. La arquitectura final del discriminador se lo puede ver en la figura 2.5.

Para las capas convolucionales, los pesos del *kernel* se inicializaron con media cero y una desviación estándar de $0{,}02$ con distribución normal $\mathcal{N}(0, 0.2)$ [22]. La capa de activación, ubicada después de cada capa convolucional a excepción de la última, fue una función

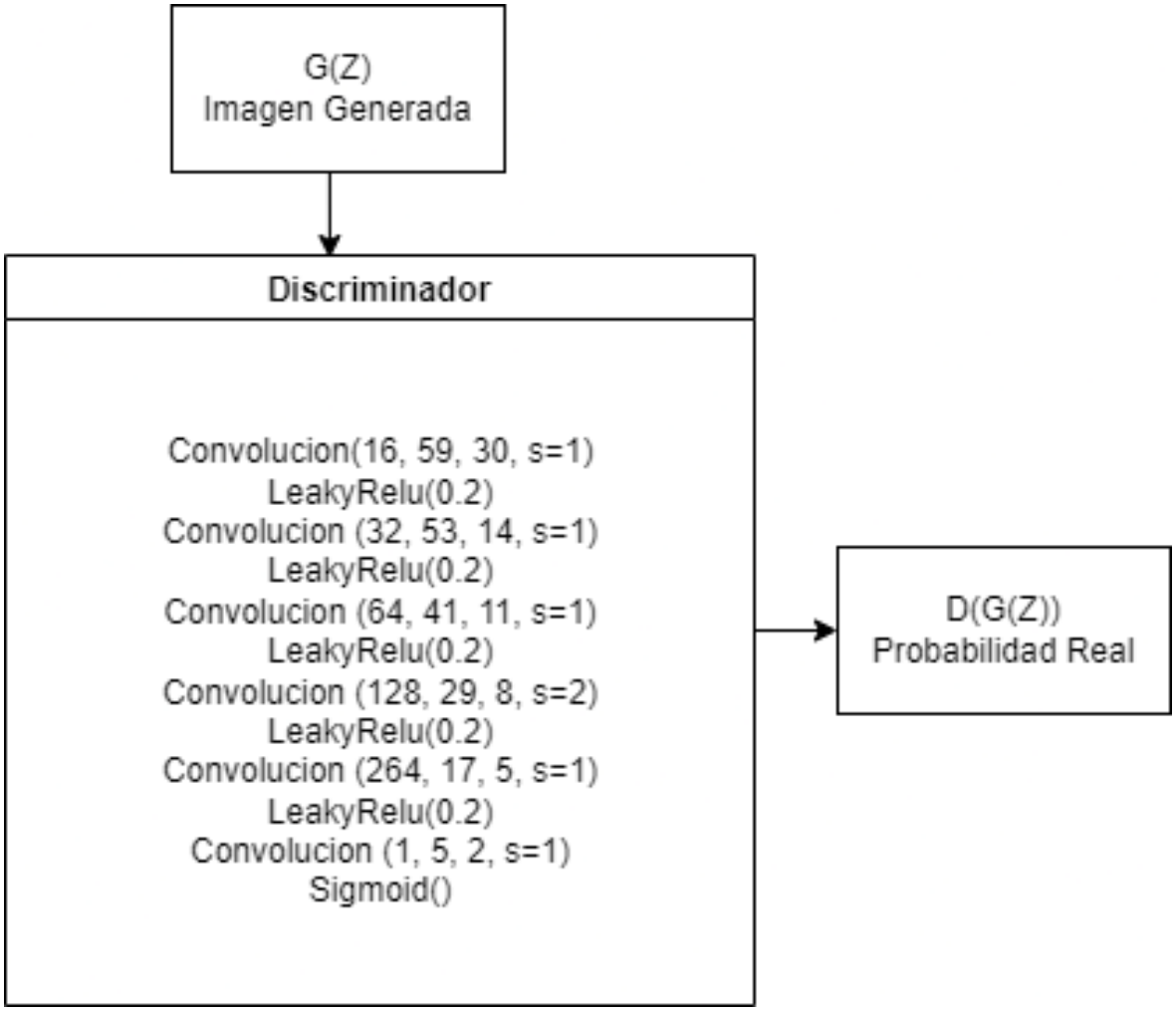

Figura 2.5: Arquitectura del discriminador. El discriminador toma una imagen de 133 x 29. Va aumentando el número de filtros y reduciendo la dimensionalidad de la imagen. Finalmente, la salida es un único valor correspondiente a la probabilidad de que la imagen generada sea real.

LeakyRelu con una pendiente negativa de 0.2 para permitir la activación de pequeños datos negativos [22].

El generador está formado por capas convolucionales transpuestas (transposed convolutions) [19]. Las cuales a partir de un vector de ruido se genera una pequeña imagen bidimensional con un número de filtros de 1024 para luego ir reduciendo progresivamente este número, a la vez que va aumentando el tamaño de la imagen generada hasta finalmente tener una imagen única de tamaño de 129*33. La arquitectura final del generador se lo puede ver en la figura 2.6.

De manera similar, al discriminador se pasa la salida de estas capas convolucionales transpuestas por una capa LeakyRelu. Para terminar, al final de la última capa del discriminador se lo pasa por una capa de activación tangente hiperbólica, con lo cual la salida del genera-

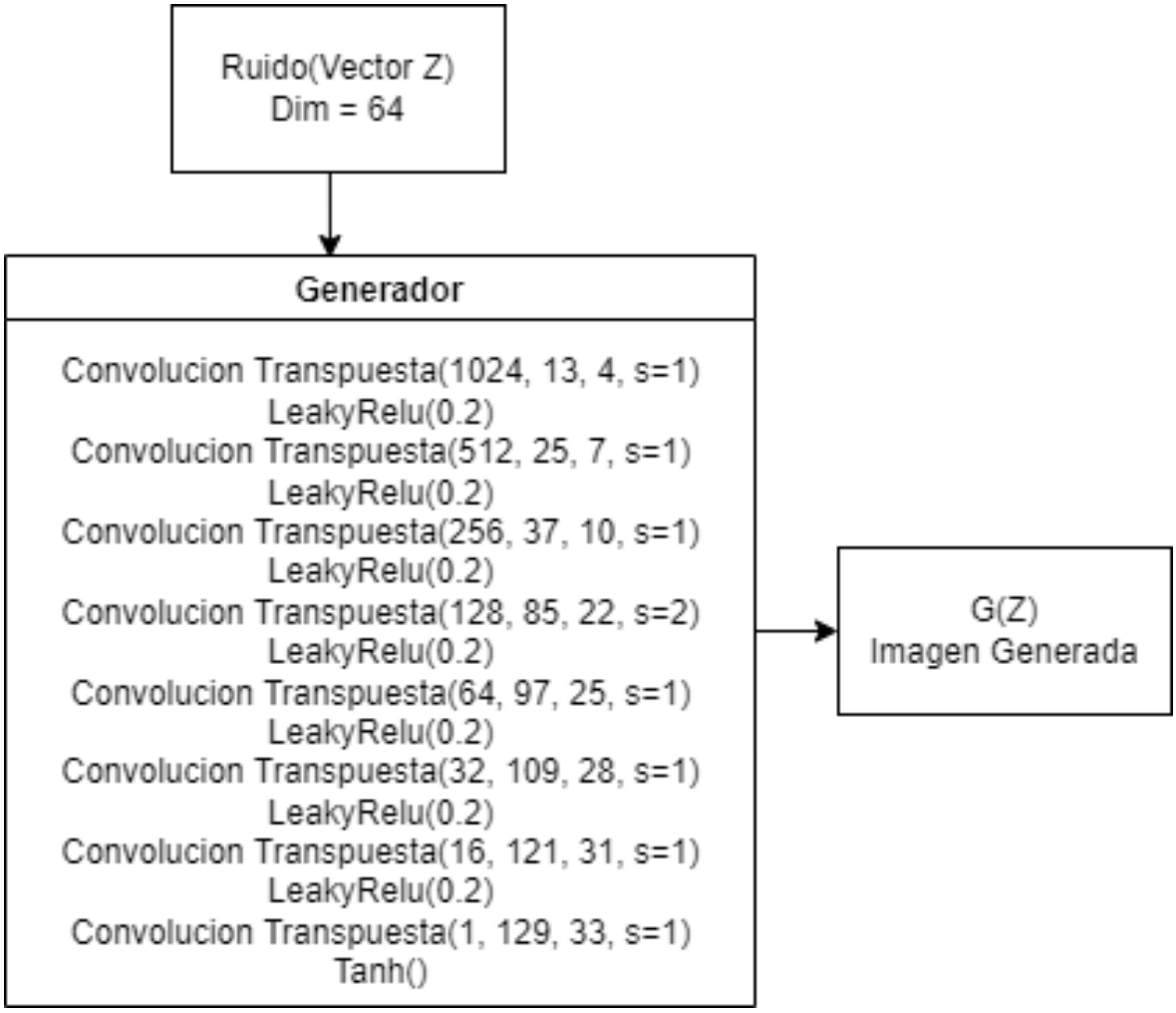

Figura 2.6: Arquitectura del generador. El generador toma un vector de ruido el cual va aumentando de dimensión y reduciendo el número de filtros, finalmente la salida es una imagen de 133*29 correspondiente a la imagen generada.

dor es una imagen de 129*33 con valores en el rango $[-1, 1]$. Los valores de inicialización y de activación de las capas LeakyReLu y capas convolucionales se lo realizó de la misma manera que en el discriminador.

Es importante mencionar que debido a que se estuvieron generando imágenes bidimensionales rectangulares (no cuadradas), el kernel de estos, se adaptó de tal manera que se pudo subir o bajar progresivamente la dimensionalidad de las imágenes obteniéndose finalmente la imagen rectangular.

El uso de capas de Batch Normalization (BN) es común en este tipo de arquitecturas generativas, en el presente trabajo, después de que se realizó varias pruebas, se descartó el uso de estas capas ya que se tenían problemas de convergencia en el modelo. Debido a esto se observó que se redujo el número de épocas necesarias para que el modelo converja, siendo

necesaria únicamente 15 épocas. Estos resultados son coherentes con los encontrados en otros trabajos que también han evitado usar BN en el uso de GAN y cGAN [23–25].

Además la red cGan crea mediante el uso de generadores aleatorios un vector **z** de 64 valores, este vector, junto a una etiqueta de tipo **t**, pasa al generador cuya salida será $G(\mathbf{z}, \mathbf{t})$. Este resultado que correspondería a una imagen correspondiente al espectrograma de una red sísmica se lo pasa al discriminador el cual, junto a nuevamente la etiqueta del tipo generara una salida $D(G(\mathbf{z}, \mathbf{t}), \mathbf{t})$ la cual corresponderá a la probabilidad de que la imagen $G(\mathbf{z}, \mathbf{t})$ sea una imagen similar a una real del tipo correspondiente.

Además, cabe mencionar que el entrenamiento del discriminador se realizó con mini-batches correspondientes tanto a imágenes de la data set real como de imágenes generadas con el modelo. El modelo alterna el entrenamiento del discriminador y del generador en 2 etapas, en las cuales cada uno es entrenado de forma independiente.

Finalmente, otra modificación introducida en la arquitectura típica de modelo cGAN es la de entrenar el generador 3 veces por cada vez que se entrena el discriminador, con esto se logra evitar el sobre entrenamiento del discriminador que aprende más rápido que el generador y por ello la red cGAN colapse en su entrenamiento. Con este mismo objetivo se introdujo un regularizador en el discriminador.

2.4. Entrenamiento

En el proceso de entrenamiento se utilizaron mini-batches de 12 muestras cada uno por un total de 15 épocas. La función de perdidas función utilizada es Binary Cross Entropy(BCE) [17] y el optimizador utilizado es ADAM [26]. Además, como ya se indicó para reducir la fuerza del discriminador, que puede colapsar la red debido a que este tiene un trabajo más sencillo, se utilizó un regularizador del tipo L2 [27, 28] en el discriminador con un parámetro de 0.001, este regularizador evitará que el discriminador aprenda de una manera rápida y le dará al generador para mejorar.

Igualmente, como se indicó anteriormente, para reducir la fuerza con la que el discriminador aprende debido a su tarea más sencilla, el generador se lo entrena 3 veces por cada vez que se entrene al discriminador, con estas modificaciones el generador es capaz de generar señales más reales antes que el discriminador colapse.

Si no se hicieran estas correcciones, el discriminador llegaría a distinguir imágenes reales y generadas muy rápido, sin dar tiempo para que el generador genere imágenes capaces

de confundir al discriminador y por ello colapsar el modelo ya que rápidamente este diferenciaría entre imágenes reales y generadas, por lo cual no daría lugar a un aprendizaje de la arquitectura.

El entrenamiento de la arquitectura implementada se lo realiza en 2 fases, cada una de las cuales entrena y realiza una actualización de los parámetros de únicamente una de las 2 partes del modelo. Estas 2 fases se repiten por el número de épocas totales.

De esta manera la primera fase del entrenamiento es para entrenar al Discriminador, para este se toman tanto señales reales como señales generadas cada una con su respectiva etiqueta del tipo de señales volcánicas ingresado. En esta etapa el discriminador espera poder distinguir correctamente si la señal es real o es generada y en función del acierto de este, se actualizarán sus parámetros mediante la retro-propagacion. En esta fase únicamente se usa el generador para generar las señales falsas, mas no se actualizarán ninguno de sus parámetros, por lo cual no se realiza ningún entrenamiento ni mejora del generador. Esta fase se la puede ver en la Figura 2.7.

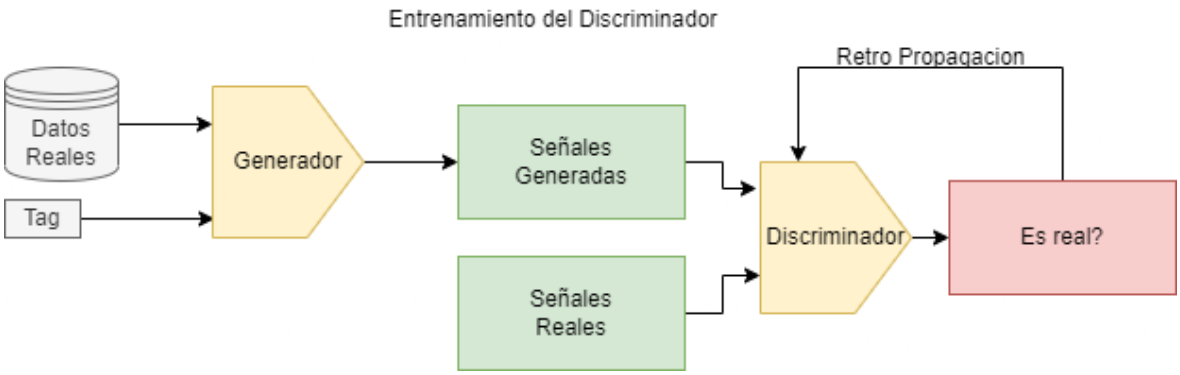

Figura 2.7: Entrenamiento del discriminador. El discriminador toma tanto señales reales como generadas para su entrenamiento. En función a su capacidad para detectar correctamente cada una de estas se actualizarán sus parámetros.

La segunda y última fase es dedicada al entrenamiento del Generador, en esta etapa todo el batch corresponde a señales generadas por el generador, las cuales pasan al discriminador para determinar la probabilidad de que estas sean reales. Debido a que en esta etapa el objetivo del generador es engañar al discriminador, el generador actualiza sus parámetros en función a que tanto estas imágenes actuales lograron efectivamente engañar al discriminador, el discriminador no actualizará sus parámetros en esta etapa. Esta fase se la puede ver en la Figura 2.8.

En cada una de las fases el objetivo es distinto, en la primera etapa el discriminador debe-

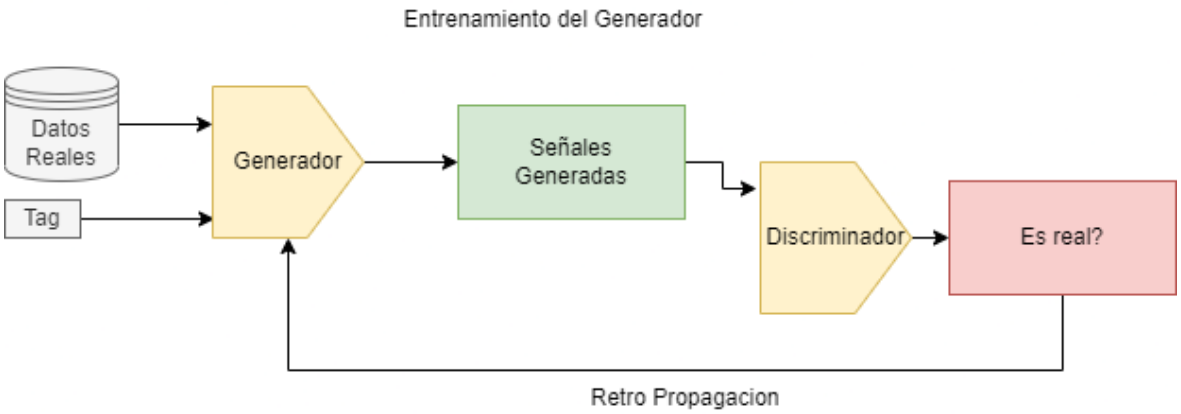

Figura 2.8: Entrenamiento del generador. El generador tomara únicamente imágenes generadas para su entrenamiento y actualizara sus parámetros de manera que le permita engañar al discriminador.

rá intentar mejorar en detectar imágenes reales o falsas correctamente, en la segunda el generador aprenderá a generar imágenes falsas que parezcan reales, este comportamiento adversario de la red finalmente logrará obtener señales reales aparentes desde el ruido introducido inicialmente.

En el trabajo se usa el enfoque TTUR (Two Time-scale Update Rule) [29]. Este enfoque establece diferentes tasas de aprendizaje para el generador y el discriminador para ajustar de mejor manera las necesidades del sistema y conseguir señales más realistas [29]. Por ello para el discriminador se establece una tasa de aprendizaje de 0.00009 y para el discriminador de 0.0002.

2.5. Recuperación de la señal

Una vez generada la imagen del espectrograma en el dominio de la frecuencia es necesaria pasarla al dominio del tiempo, el problema es que la red cGAN únicamente nos da una parte de espectrograma, el de magnitud, faltándonos el de fase para reconstruir correctamente la señal por ello se usará el algoritmo de Griffin-Lim que nos permite reconstruir la señal en tiempo desde únicamente el espectrograma de magnitud [30].

Este algoritmo se basa en la redundancia de la transformada de Fourier de tiempo corto (STFT) y a través de una fase inicial aleatoria reconstruirá una señal en el dominio del tiempo [30]. Es necesario que en el proceso de reconstrucción se elija cuidadosamente el número de iteraciones ya que un gran número de estas a pesar que mejora la señal que se quiere recuperar, también esta tiene un alto costo computacional. Por ello el número de

iteraciones elegidos para este proyecto fue de 1000. Este se lo eligió cogiendo señales de la base de datos y pasándolas a el dominio de la frecuencia posteriormente descartando su espectrograma de fase y pasándola nuevamente al dominio del tiempo mediante este algoritmo, finalmente se realizó una comparación visual de las señales original y recuperada de manera que estas eran muy similares. En la figura 2.9 se puede ver un ejemplo de estas señales siendo estas muy similares.

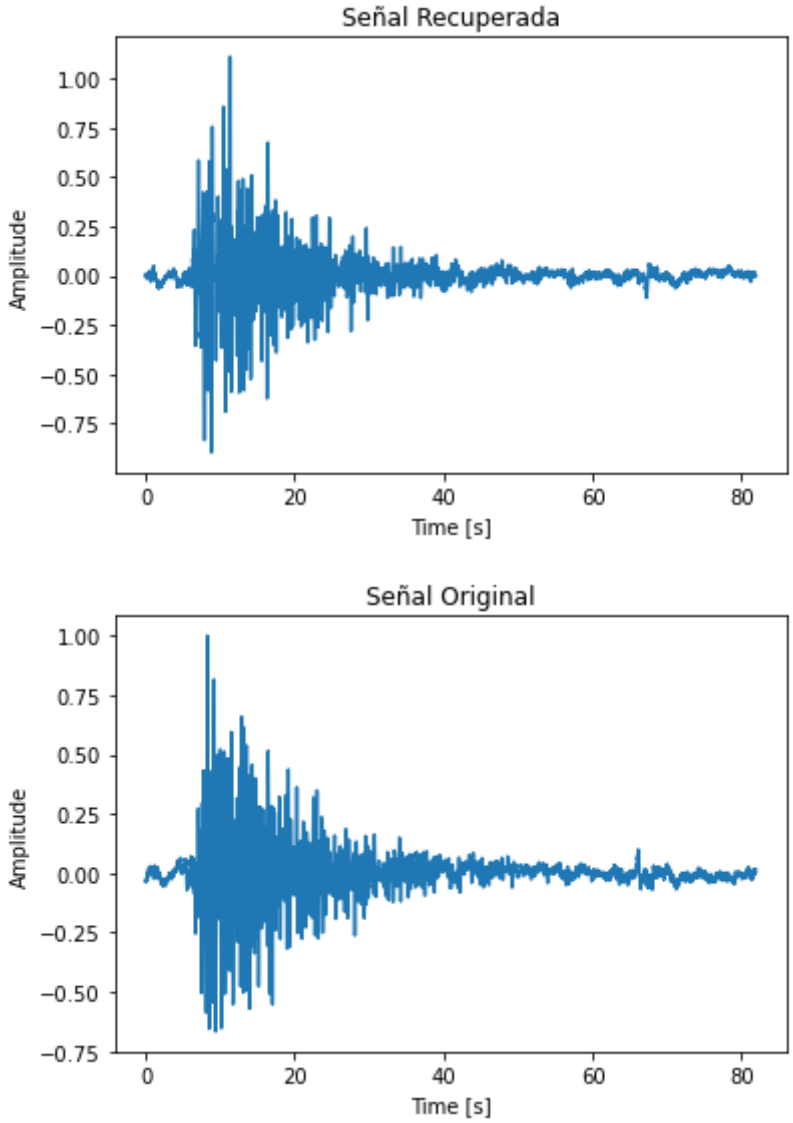

Figura 2.9: Señal en el dominio del tiempo original y señal reconstruida desde el algoritmo de Griffin-Lin

Con el uso de este algoritmo es posible obtener señales similares señales a las señales finales del preprocesamiento, pero siendo estas artificiales y generadas a partir del modelo diseñado.

2.6. Evaluacion

Al entrenar una red cGAN o cualquier tipo de red GAN debido a la naturaleza de esta, es decir 2 modelos compitiendo entre sí, el modelo se entrena hasta que haya un punto en el que se esté satisfecho con los resultados obtenidos y ambas redes (el discriminador y generador) alcanzan un equilibrio. Por lo cual estas no están optimizándose hacia la convergencia [31], esto conlleva a que no haya un medidor claro al momento de evaluar una GAN [10]. Además, debido a que un buen modelo implica llegar a este equilibrio el generador con cada pasada de mini-batch cambiará su comportamiento no solo mejorando sino también pudiendo desmejorar en la calidad de imágenes generadas.

Por ello el primer filtro que se implementó en el presente trabajo es el ver que las señales generadas sean similares visualmente a las señales originales [10] (tanto en el dominio del tiempo como en el de la frecuencia), una vez las señales pasaron este filtro se procede a evaluarlas con métodos más certeros. Este proceso ayudó a encontrar los parámetros finales del modelo propuesto.

Una vez obtenidos estos resultados previos se procedió a evaluar mediante el parámetro de la distancia *Fréchet Inception Distance* (FID) [29]. La cual es la métrica más común para evaluar este tipo de redes [10] cuya salida es una imagen o en este caso un espectrograma y utiliza las características obtenidas por un clasificador pre-entrenado de imágenes conocido como *Inception V3* [32]. A esta red se le pasan los espectrogramas generados para luego evaluarlos haciendo uso de la Distancia de Fréchet (FD).

La Distancia de Fréchet (FD) [29] es una medida objetiva que se utiliza para determinar el grado de similitud entre dos distribuciones de probabilidad, en nuestro caso, la distribución del conjunto de datos original y la distribución de datos generados. Mientras más bajos sean los resultados de esta más similares son los conjuntos de datos. La FD se calcula mediante:

$$d^2 = \|\mu_r - \mu_g\|^2 + \mathrm{Tr}\left(\mathbf{C}_r + \mathbf{C}_g - 2\sqrt{\mathbf{C}_r\mathbf{C}_g}\right), \tag{2.2}$$

donde, μ_r y μ_g son los valores medios de las magnitudes reales y generadas, respectivamente, $\mathbf{C}_r$ and $\mathbf{C}_g$ son las matrices de covarianza de las magnitudes reales y generadas, respectivamente.

La red *Inception V3* fue entrenada para clasificar imágenes de objetos naturales, sin embargo es posible comparar con esta misma espectrogramas como se lo realizó en [33]. Los espectrogramas son obtenidos directamente de la salida del generador y son procesados

por la red *Inception V3*, previamente entrenada para obtener los vectores de características y luego se calcula la FID (Fréchet Inception Distance). Con la cual se evaluará la misma.

De esta manera, se utilizó tanto la percepción humana para poder calibrar los parámetros finales de la red como la FID para realizar una evaluación más medible de esta, que compara lo similar que es un espectrograma real a uno generado y de esta manera nos da una medida que se puede utilizar para evaluar el rendimiento del modelo. Con la combinación de estos métodos se obtuvieron los mejores parámetros para que la red genere señales volcánicas realistas. Un diagrama de como se usa la métrica de evaluación FID se lo puede ver en la figura 2.10,

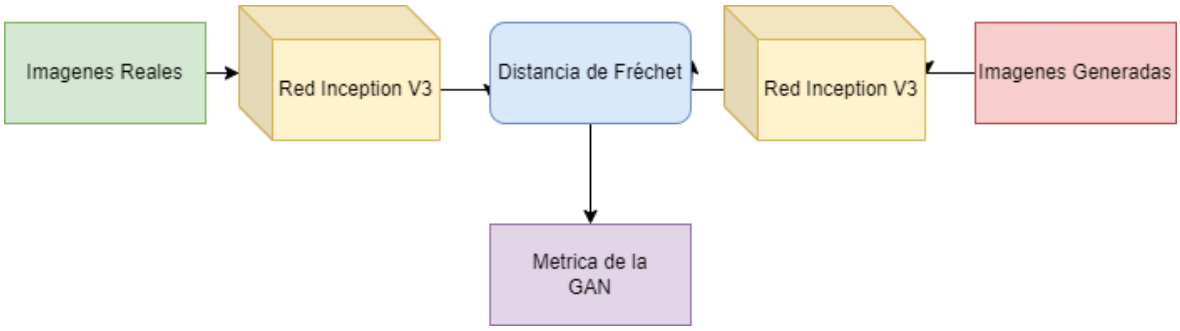

Figura 2.10: Diagrama del flujo del FID

3. RESULTADOS, CONCLUSIONES Y RECOMENDACIONES

3.1. Resultados

Debido a la naturaleza de las señales volcánicas generadas (señales sísmicas del tipo VT y LP) que son muy variables, el diseño de una red generativa que logre generar estas señales con precisión es de alta complejidad. El diseño propuesto por el presente trabajo pudo capturar las frecuencias típicas de estas señales en el dominio del tiempo en los espectrogramas como se puede apreciar en la Figura 3.1. y en la Figura 3.2 se puede apreciar las señales en el dominio del tiempo. Se observó que generar un espectrograma el cual se lo genera como una imagen fue un acierto ya que como fue observado por Ulyanov et al. [34] las capas convolucionales del modelo pueden captar los patrones de estas señales vulcanológicas.

Sin embargo, las redes tipo GAN y cGAN tienen un problema, si el diseño no fue hecho de manera correcta, en este caso se denomina el modo colapso *(Mode Collapse)* el cual causa que el generador únicamente genere un tipo de imagen, la cual identificó como la imagen que logra engañar al discriminador [18], este problema y otros como generación de solo ruido o problemas en las que el generador no captaba finalmente la forma de la señal fueron encontrados en el desarrollo del proyecto, estos problemas se los identificó ya que a simple vista no corresponden a una señal volcánica como se lo puede ver en la figura 3.3.

La función de pérdida tanto del discriminador como del generador se estabiliza pasadas unas pocas iteraciones del entrenamiento y permanece relativamente constante en todo el entrenamiento como se puede apreciar en la figura 3.4. Esa constancia es debido a que constantemente el generador y el discriminador mejoran en sus respectivas funciones, conforme van mejorando cada vez al generador se le hace más difícil engañar al discriminador y a su vez al discriminador cada vez le llegan imágenes generadas más reales y por ello sus funciones de pérdida se mantienen similares y no se ve un claro indicativo de que la red ha mejorado su desempeño.

De manera similar en la figura 3.4 se aprecia la función de pérdida del discriminador tanto evaluada a los eventos reales, como a los generados, como se mencionó estas una vez estabilizadas tienen a cambiar poco y se mantienen relativamente estable. Cabe mencionar

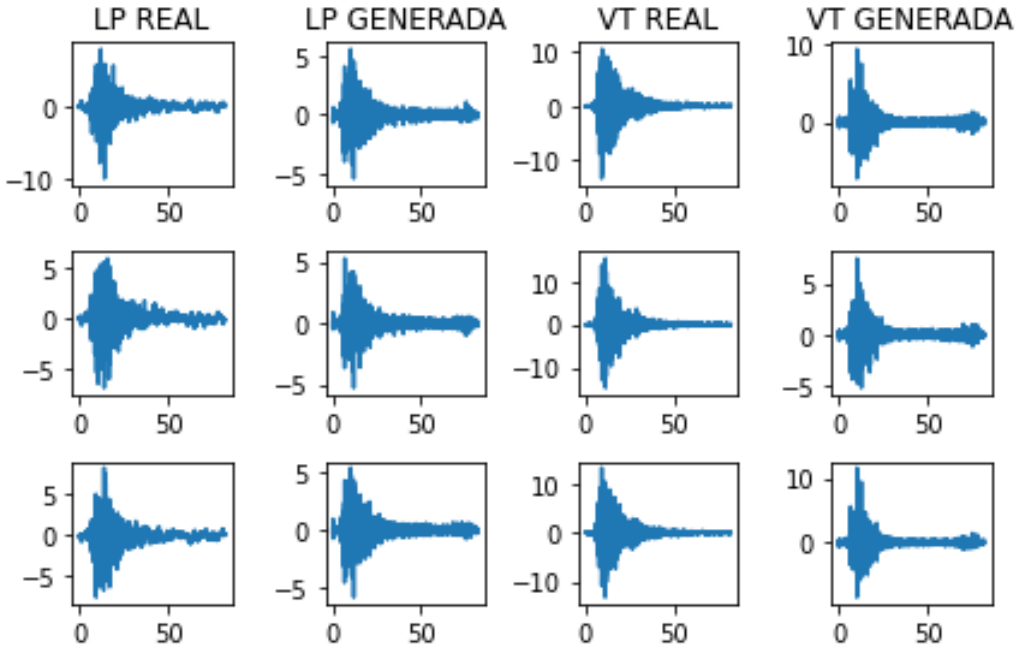

Figura 3.1: Comparación entre señales reales y generadas en el dominio del tiempo

que la función de perdida completa del discriminador es el promedio de estas funciones de perdida parciales y en función de este promedio se realiza la actualización de parámetros.

La figura 3.5 nos muestra la precisión del discriminador en identificar muestras reales y generadas correctamente, es decir el porcentaje de acierto, similarmente a como ocurre con la función de pérdida esta se estabiliza después de transcurridos unas cuantas épocas de entrenamiento y lo hace de tal manera que se ubican ambas precisiones en el centro es decir que identifica correctamente aproximadamente la mitad de las muestras ingresadas al discriminador. Esto nos indica que tanto el generador como el discriminador están mejorando ya que ninguno se ha vuelto tan fuerte que mueva esta función ya sea a todo acierto, que indicaría un discriminador muy fuerte, o a todas fallas, que indicaría un generador muy fuerte. Un ejemplo de este fallo se lo puede ver en la figura 3.6.

El modelo se entrenó por un total de 15 épocas ya que en estas el modelo fue capaz de generar señales que captan las características de las señales volcánicas tanto en el dominio del tiempo como en el de la frecuencia. Debido a la limitación del conjunto de datos aumentar mucho el número de épocas podría ser perjudicial [27, 35]. Por ello observando las medidas de precisión del discriminador se concluyó que entrenarlo durante mas épocas no sería útil ya que el modelo parece converger en las 15 épocas.

Este trabajo utiliza la métrica de la FID para evaluar el desempeño de la GAN y poder elegir

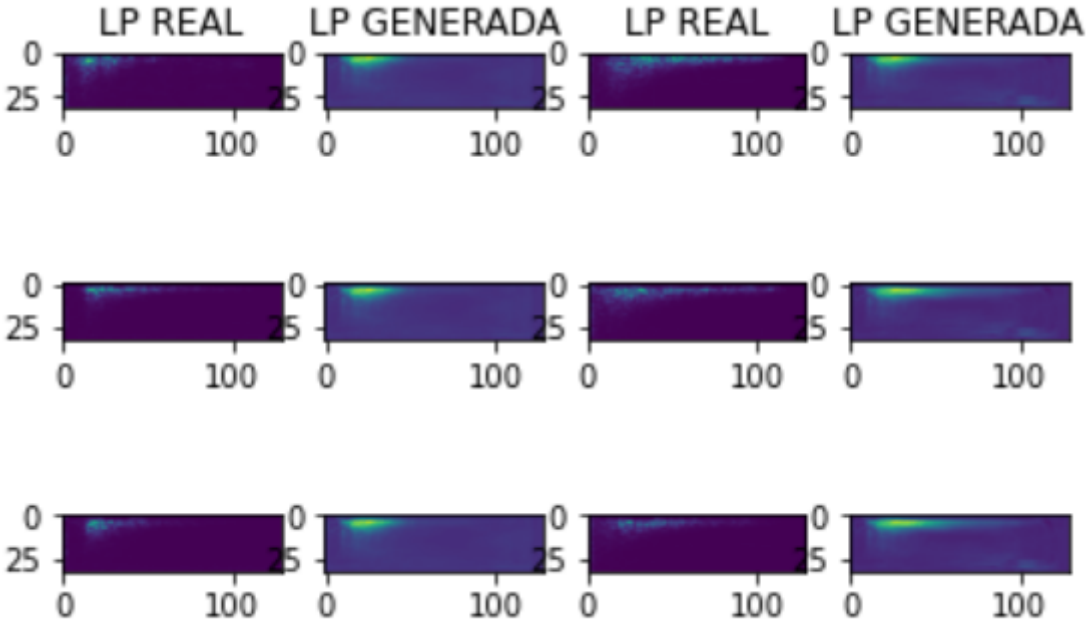

Figura 3.2: Comparación entre señales reales y generadas en el dominio de la frecuencia

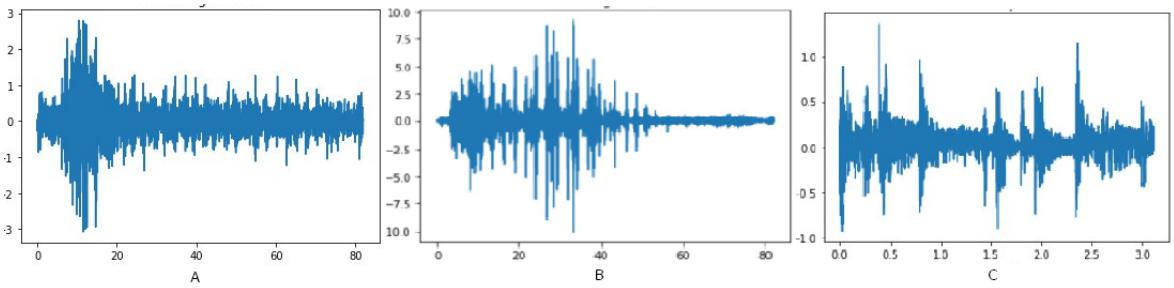

Figura 3.3: Señales generadas incorrectamente.a) Señal generada muy ruidosa b) Señal generada no corresponde visualmente c) Señal generada con el colapso del modelo.

sus parámetros correctamente. Esta métrica evalúa la GAN en el dominio de la frecuencia tomando directamente la salida de las imágenes de espectrogramas vulcanológicos de la salida del generador y las compara con los eventos reales de nuestra base de datos, de esta manera el índice obtenido nos da una comparación entre una señal real y una generada. Además, se comparará también las señales generadas con ruido para poder tener un punto de referencia. Finalmente se comparan las señales generadas entre sí para poder determinar si la parte condicional del modelo esta efectivamente generando eventos distintos en función a la etiqueta. Los resultados obtenidos se los detallan finalmente en la tabla 3.1.

Como se aprecia en la tabla 3.1 la red es capaz de generar de mejor manera los eventos del tipo LP que los eventos del tipo VT esto puede ser debido a que en la base de datos que se usó para el entrenamiento hay una mayor cantidad de eventos LP que de eventos VT.

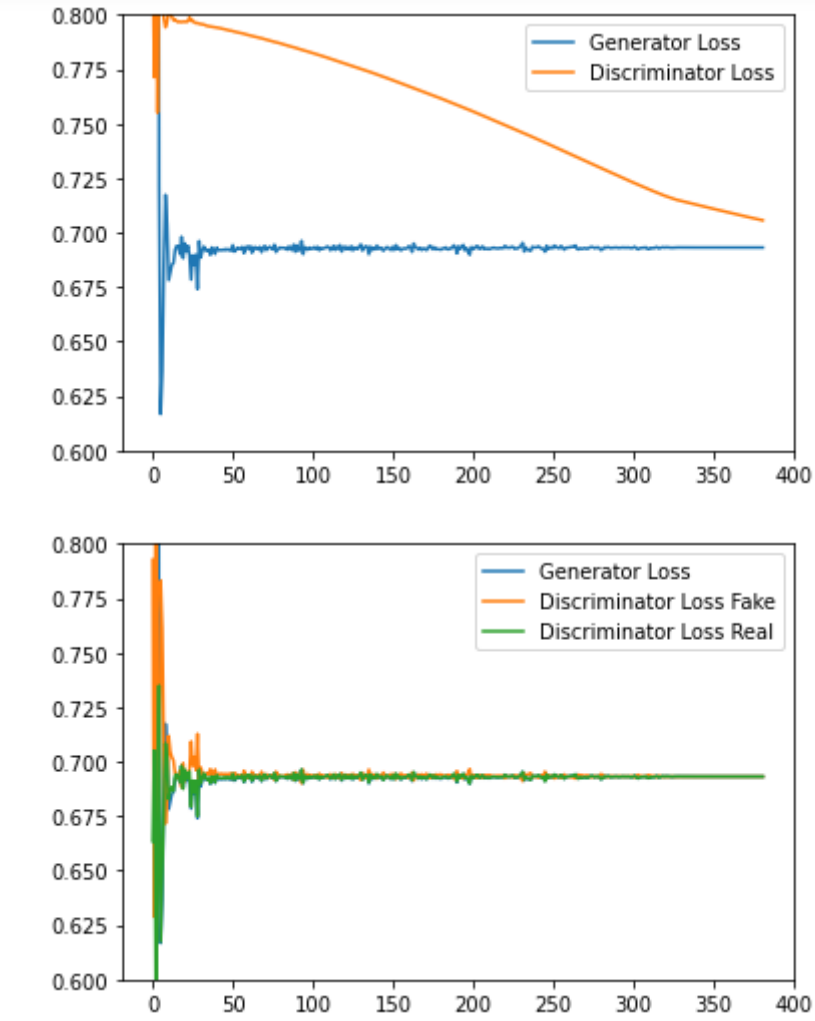

Figura 3.4: Función de pérdida del generador y discriminador

Finalmente hay que mencionar que la métrica usada de la FID es útil para comparar y encontrar similitud entre diferentes grupos de imágenes, sin embargo, la red Incepción V3 no está entrenada inicialmente con imágenes de estas características, por lo que no es posible tener una clara base numérica de comparación de los índices obtenidos. De esta manera los resultados de la tabla 3.1 que corresponde a la métrica FID están limitados por la baja cantidad de eventos para el entrenamiento y la disparidad de estos mismos, así como el método de evaluación usado que no es específica para el uso con espectrogramas.

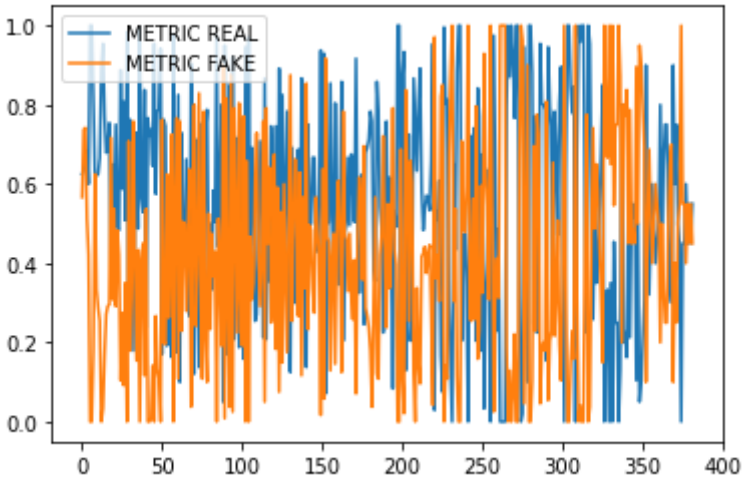

Figura 3.5: Función de precisión en el discriminador

Tabla 3.1: Tabla de resultados FID. Para las comparaciones te tomó en cuenta generar el mismo número de muestras que las muestras reales usadas.

Comparación	FID (Frecuencia)
LP Real - LP Generado	234.92
LP Real - Ruido	520.36
VT Real - VT Generado	307.46
VT Real - Ruido	503.47
VT Generado - LP Generado	85.53

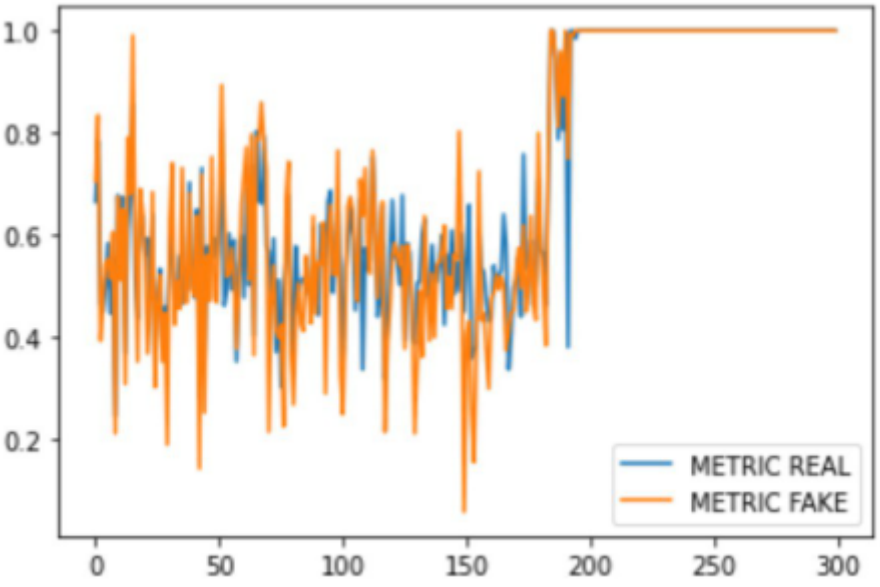

Figura 3.6: Función de precisión en el discriminador cuando existe un colapso del modelo

3.2. Conclusiones

- En el presente trabajo se logró generar señales sísmicas VT (Volcano-Tectonic) y LP (Long-Period) mediante el uso de una arquitectura de aprendizaje automático cGAN específicamente se logró generar los espectrogramas de éstos para luego mediante el algoritmo de GriffLim regresarlos al dominio del tiempo. Estas señales son similares a las originales tanto en el dominio de la frecuencia como en el dominio del tiempo

- Al momento de entrenar una red cGAN es importante tener en cuenta la fuerza de cada una de sus partes, para que puedan llegar a una convergencia y a generar lo deseado, para ello la modificación y diseño del modelo debe estar a la par con la experimentación y de esta manera llegar a la arquitectura deseada. En este contexto se evidenció que modificar la arquitectura del modelo original, adaptándolo para nuestras necesidades mejoró el desempeño del modelo y nos llevó a obtener mejores resultados.

- El modelo cGAN desarrollado es sensible en cuanto a sus hiperparametros debido a que leves variaciones en estos llevan a distintos resultados pudiendo incluso colapsar por estas pequeñas variaciones, por ello la elección cuidadosa de mano de la experimentación de estos parámetros es de suma importancia.

- Las señales generadas por estos modelos son importantes para una prevención en las actividades sísmicas, y el uso de los resultados de este proyecto puede ser usado para mejorar los sistemas de detección y prevención de actividad volcánica.

- Los parámetros medidos de la métrica FID usada para evaluar el modelo indican que el modelo genera correctamente señales del tipo volcánicas y que es posible elegir si se generan señales del tipo VT o del tipo LP. Esto debido a que la métrica FID al comparar las señales de VT y LP generadas nos da un valor no cercano a 0. Esto quiere decir que las señales VT y LP generadas no son muy similares entre si y existe una diferencia entre ellas.

- La cantidad de datos disponibles para el entrenamiento de la red cGAN es crucial ya que debido baja cantidad de elementos VT en comparación a la cantidad de elementos LP el modelo tiene mayor dificultad en generar estas señales lo cual se puede evidenciar en la medida FID de las mismas.

3.3. Recomendaciones

- Debido a la naturaleza de las señales sísmicas y que estas son parecidas a señales de sonido, en futuros trabajos se recomienda utilizar arquitecturas de aprendizaje automático más especializadas en la generación de señales acústicas como MelGAN.
- Ya que los datos de las señales sísmicas son datos secuenciales ligados entre ellos se recomienda hacer uso de tecnologías de redes que permitan generar este tipo de datos como las redes LSTM para generar señales volcánicas.
- Modificar la arquitectura presentada en este proyecto con el objetivo de admitir tanto el espectrograma de magnitud como el espectrograma de fase en el entrenamiento y generación, este cambio podría mejorar la generación de señales ya que en el espectrograma de fase podría tener información adicional que un modelo generativo podría ser capaz de abstraer y de esta manera mejorar los resultados en la generación.

4. REFERENCIAS BIBLIOGRÁFICAS

[1] M. Malfante, M. Dalla Mura, J.-P. Métaxian, J. I. Mars, O. Macedo, and A. Inza, "Machine learning for volcano-seismic signals: Challenges and perspectives," *IEEE Signal Processing Magazine*, vol. 35, no. 2, pp. 20–30, 2018.

[2] C. Macías, "Diseño de un sistema digital de monitoreo de lahares en tiempo real," *Repositorio Digital Institucional de la Escuela Politécnica Nacional*, 2012.

[3] R. Tilling, "The critical role of volcano monitoring in risk reduction," *Advances in Geosciences*, vol. 14, 01 2008.

[4] R. Lara, M. Rodriguez, and J. Larco, "A real-time microearthquakes-detector based on voice activity detection and endpoint detection: An approach to cotopaxi volcano," *Journal of Volcanology and Geothermal Research*, vol. 400, p. 106867, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0377027319305517

[5] E. Carrera, A. Pérez, and R. Lara-Cueva, "Automated systems for detecting volcano-seismic events using different labeling techniques," *International Conference on Applied Technologies*, vol. 1, pp. 133–144, 2019.

[6] G. Cortés, M. C. Benítez, L. García, I. Álvarez, and J. M. Ibanez, "A comparative study of dimensionality reduction algorithms applied to volcano-seismic signals," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 9, no. 1, pp. 253–263, 2016.

[7] M. Mirza and S. Osindero, "Conditional generative adversarial nets," 2014. [Online]. Available: https://arxiv.org/abs/1411.1784

[8] N. Pérez, D. Benítez, F. Grijalva, R. Lara-Cueva, M. Ruiz, and J. Aguilar, "Eseismic: Towards an ecuadorian volcano seismic repository," *Journal of Volcanology and Geothermal Research*, p. 106855, 2020.

[9] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle,

A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[10] A. Borji, "Pros and cons of gan evaluation measures," *Computer Vision and Image Understanding*, vol. 179, pp. 41–65, 2019.

[11] R. Trombley, *The Forecasting of Volcanic Eruptions*. iUniverse, 2006. [Online]. Available: https://books.google.com.ec/books?id=piyUqDucLosC

[12] "Cotopaxi," https://www.igepn.edu.ec/cotopaxi, 2020. [Online]. Available: https://www.igepn.edu.ec/cotopaxi

[13] D. Andrade, E. I. G. Escuela Politécnica Nacional (Quito, and I. de recherche pour le développement (France), *Los peligros volcánicos asociados con el Cotopaxi*, ser. Peligros volcánicos en el Ecuador. Corporación Editora Nacional, 2005. [Online]. Available: https://books.google.com.ec/books?id=pn1dAAAAMAAJ

[14] "Cotopaxi red de monitoreo instituto geofísico EPN," https://igepn.edu.ec/cotopaxi-red-de-monitoreo, 2020. [Online]. Available: https://igepn.edu.ec/cotopaxi-red-de-monitoreo

[15] S. Shalev-Shwartz and S. Ben-David, *Understanding Machine Learning: From Theory to Algorithms*, ser. Understanding Machine Learning: From Theory to Algorithms. Cambridge University Press, 2014.

[16] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*. MIT press Cambridge, 2016, vol. 1.

[17] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[18] I. Goodfellow, "Nips 2016 tutorial: Generative adversarial networks," 2016.

[19] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[20] S. Vasquez and M. Lewis, "Melnet: A generative model for audio in the frequency domain," 2019.

[21] K. Kumar, R. Kumar, T. de Boissiere, L. Gestin, W. Z. Teoh, J. Sotelo, A. de Brébisson, Y. Bengio, and A. C. Courville, "Melgan: Generative adversarial networks for conditional waveform synthesis," in *Advances in Neural Information Processing Systems*, 2019, pp. 14 910–14 921.

[22] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. icml*, vol. 30, no. 1, 2013, p. 3.

[23] C. Donahue, J. McAuley, and M. Puckette, "Adversarial audio synthesis," in *International Conference on Learning Representations*, 2018.

[24] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of gans for improved quality, stability, and variation," 2017.

[25] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2019, pp. 4401–4410.

[26] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014.

[27] T. Karras, M. Aittala, J. Hellsten, S. Laine, J. Lehtinen, and T. Aila, "Training generative adversarial networks with limited data," *arXiv preprint arXiv:2006.06676*, 2020.

[28] K. Xu, C. Li, J. Zhu, and B. Zhang, "Understanding and stabilizing gans' training dynamics using control theory," in *Proceedings of the International Conference on Machine Learning (ICML), Vienna*, 2020.

[29] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a local nash equilibrium," in *Advances in neural information processing systems*, 2017, pp. 6626–6637.

[30] Y. Wakabayashi and N. Ono, "Griffin-lim phase reconstruction using short-time fourier transform with zero-padded frame analysis," in *2019 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, 2019, pp. 1863–1867.

[31] V. Nagarajan and J. Z. Kolter, "Gradient descent gan optimization is locally stable," in *Advances in neural information processing systems*, 2017, pp. 5585–5595.

[32] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the inception architecture for computer vision,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826.

[33] J. Bai, B. Wang, C. Chen, J. Chen, and Z. Fu, “Inception-v3 based method of lifeclef 2019 bird recognition.” in *CLEF (Working Notes)*, 2019.

[34] D. Ulyanov, A. Vedaldi, and V. Lempitsky, “Deep image prior,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 9446–9454.

[35] V. Nagarajan, C. Raffel, and I. J. Goodfellow, “Theoretical insights into memorization in gans,” in *Neural Information Processing Systems Workshop*, 2018.

ANEXOS

Dado que los algoritmos de aprendizaje automático fueron entrenados en la plataforma Jupyter, se adjunta como anexo digital los scripts en extensión ".ipynb" así como el código fuente de extensión ".py" para cada etapa del trabajo y además la base de datos utilizada antes y después del proceso de pre-procesamiento. Adicionalmente se adjunta en formato H5 una base de datos de 10000 señales VT y 10000 señales LP generadas con el modelo implementado.

ANEXO A. Instrucciones para ejecutar el código fuente.

ORDEN DE EMPASTADO