Generación de señales sintéticas de eventos sismo-volcánicos del volcán Cotopaxi a través de un modelo de red neuronal adversario generativo condicional

Galarza Cruz, Christian Ramiro y Vega Vergara, Josselyn Michelle

Departamento de Eléctrica, Electrónica y Telecomunicaciones

Carrera de Ingeniería en Electrónica y Telecomunicaciones

Trabajo de titulación previo a la obtención del título de Ingeniero en Electrónica y Telecomunicaciones

Ing. Lara Cueva, Román Alcides PhD.

15 de julio de 2022

COPYLEAKS

TESIS_GALARZA_VEGA_FINAL.pdf

Scanned on: 15:12 July 13, 2022 UTC

Firmado electrónicamente por:
ROMAN
ALCIDES LARA

1.3%	8	22037
Overall Similarity Score	Results Found	Total Words in Text

Identical Words	87
Words with Minor Changes	1
Paraphrased Words	190
Omitted Words	78

Departamento de Eléctrica, Electrónica y Telecomunicaciones

Carrera de Ingeniería en Electrónica y Telecomunicaciones

Certificación

Certifico que el trabajo de titulación: "**Generación de señales sintéticas de eventos sismo-volcánicos del volcán Cotopaxi a través de un modelo de red neuronal adversario generativo condicional**" fue realizado por los señores **Galarza Cruz, Christian Ramiro** y **Vega Vergara, Josselyn Michelle**; el mismo que cumple con los requisitos legales, teóricos, científicos, técnicos y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPE, además fue revisado y analizado en su totalidad por la herramienta de prevención y/o verificación de similitud de contenidos; razón por la cual me permito acreditar y autorizar para que se lo sustente públicamente.

Sangolquí, 15 de julio de 2022

Firmado electrónicamente por:
ROMAN ALCIDES LARA

Ing. Lara Cueva, Román Alcides PhD.

C.I.: 1713988218

Departamento de Eléctrica, Electrónica y Telecomunicaciones

Carrera de Ingeniería en Electrónica y Telecomunicaciones

Responsabilidad de Autoría

Nosotros Galarza Cruz, Christian Ramiro y Vega Vergara, Josselyn Michelle, con cédulas de ciudadanía nº 1719865907 y 1721549598, declaramos que el contenido, ideas y criterios del trabajo de titulación **: Título**: **"Generación de señales sintéticas de eventos sismo-volcánicos del volcán Cotopaxi a través de un modelo de red neuronal adversario generativo condicional"** es de nuestra autoría y responsabilidad, cumpliendo con los requisitos legales, teóricos, científicos, técnicos, y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPE, respetando los derechos intelectuales de terceros y referenciando las citas bibliográficas.

Sangolquí, 15 julio del 2022

Galarza Cruz, Christian Ramiro

C.C.: 1719865907

Vega Vergara, Josselyn Michelle

C.C.:1721549598

Departamento de Eléctrica, Electrónica y Telecomunicaciones

Carrera de Ingeniería en Electrónica y Telecomunicaciones

Autorización de Publicación

Nosotros Galarza Cruz, Christian Ramiro y Vega Vergara, Josselyn Michelle, con cédulas de ciudadanía nº 1719865907 y 1721549598, autorizamos a la Universidad de las Fuerzas Armadas ESPE publicar el trabajo de titulación: **Título: "Generación de señales sintéticas de eventos sismo-volcánicos del volcán Cotopaxi a través de un modelo de red neuronal adversario generativo condicional"** en el Repositorio Institucional, cuyo contenido, ideas y criterios son de nuestra responsabilidad.

Sangolquí, 15 julio del 2022

.....	.....
Galarza Cruz, Christian Ramiro	**Vega Vergara, Josselyn Michelle**
C.C.: 1719865907	C.C.:1721549598

Dedicatoria

A mis queridos padres, Cinthya Cruz y César Galarza, que han sido y serán el motor que impulsa mis sueños y esperanzas, estuvieron siempre a mi lado y brindaron su incondicional apoyo en todo este proceso educativo. Todo lo que soy hoy es gracias a ustedes.

Christian Ramiro Galarza Cruz

Este trabajo de investigación está dedicado a mi familia. Un sentimiento especial de gratitud a mis queridos padres, Leonidas Vega y María Agusta Vergara, por su constante apoyo y por creer en mi capacidad para lograr tanto. A mis abuelitos, mi hermana y mi tía por todas sus oraciones y bendiciones; además su genuino amor y ayuda.

Josselyn Michelle Vega Vergara

Agradecimientos

Primeramente, doy gracias a mi familia, ya que suponen los cimientos de mi desarrollo. Todos y cada uno de ustedes: mi madre Cinthya, mi padre César y mi hermano Jorge, sus bendiciones y enseñanzas a lo largo de mi vida me han protegido y me han llevado por el camino del bien. Estuvieron presentes durante todo mi proceso de estudio del pregrado hasta finalizar con el presente trabajo de investigación. Los quiero mucho.

Del mismo modo, agradezco a todas las personas que me acompañaron durante mis estudios. Mi infinito agradecimiento a ti Karen por siempre confiar en mí, por acompañarme y apoyarme siempre. A todos mis amigos de la Universidad, en especial a mi grupo de amigos "Los Puñales", por compartir invaluables recuerdos, risas y tristezas juntos. A ti Josselyn, por ser mi amiga y compañera en el desarrollo del trabajo de titulación, juntos ejecutamos y finalizamos este proyecto de investigación exitosamente.

Agradezco a la Universidad de las Fuerzas Armadas ESPE, cuna de líderes, por permitirme forjar mis conocimientos dentro de sus aulas y laboratorios. Recordaré durante toda mi vida todas las experiencias que viví dentro de sus instalaciones. Un especial agradecimiento a nuestro mentor, Ing. Román Lara, PhD., por brindar sus conocimientos y enseñanzas para que se pueda ejecutar un trabajo de investigación de calidad.

Finalmente, agradezco a quien empiece a leer esta investigación, por permitir que mi conocimiento incurra dentro de su repertorio de información mental.

Christian Ramiro Galarza Cruz

Quiero transmitir mi más sincero agradecimiento a mi familia, mi madre María Augusta, mi padre Leonidas y mi hermana Karla, por su apoyo incondicional a lo largo de toda mi vida y durante el proceso del pregrado. Gracias a su amor incondicional y a sus oraciones, he tenido la oportunidad de completar este trabajo de investigación.

Me gustaría aprovechar esta ocasión para expresar mi más sincero agradecimiento Arnaldo por darme siempre tu apoyo, a todos mis queridos amigos especialmente a mi compañero Christian Galarza, con el cual nos hemos animado mutuamente durante todo el proceso de la ejecución del trabajo de investigación.

A la Universidad de las Fuerzas Armadas ESPE, por permitirme culminar en su totalidad mis estudios, a los ingenieros de la carrera de Electrónica y Telecomunicaciones por darnos todos los conocimientos los cuales nos permitieron desarrollar el trabajo presentado.

Finalmente, agradezco al Ing. Román Lara, PhD. por ser nuestro mentor, brindar sus conocimientos y enseñanzas. Su entusiasmo y disposición a ofrecer retroalimentación hicieron que la realización de esta investigación fuera una experiencia placentera.

Josselyn Michelle Vega Vergara

Índice de Contenidos

Índice de Tablas

Índice de Figuras

Resumen

Hoy en día se vive una creciente era de la información y de los datos, que se recolectan a través de una infinidad de medios. Estos datos tienen un peso muy importante en distintos campos de investigación, como en el estudio de los desastres naturales, como los eventos sismo-volcánicos, que tiene como objetivo el precautelar vidas a través de alertas tempranas eficientes. El etiquetado de los eventos sismo-volcánicos; como el Volcano-Tectónico (VT) y Largo Periodo (LP), es un verdadero problema. El etiquetado visual no es una opción viable por lo ineficiente y costoso que puede llegar a ser.

Este trabajo de investigación propone diseñar un modelo de Red Neuronal Adversario Generativo Condicional (CGAN, del inglés *Conditional Generative Adversarial Network*), el cual genera señales sintéticas de eventos sismo-volcánicos de tipo LP y VT, el trabajo consta de tres etapas: pre-procesamiento se acondicionan las señales sismo-volcánicas LP y VT. En la CGAN se establece dos modelos, el Generador y Discriminador, con diferentes parámetros para el entrenamiento y la obtención de respuestas en frecuencia sintéticas con características de eventos sismo-volcánico LP y VT; En el pos-procesamiento, se tienen señales sismo volcánicas sintéticas en el tiempo LP y VT. Las señales sintéticas son evaluadas por: un software de detección y clasificación de eventos sismo-volcánicos, además de la evaluación visual de las señales a través de los expertos del Instituto Geofísico de la Escuela Politécnica Nacional.

La exactitud de la clasificación se mantiene estable entre un 96.5% y un 98.5% tras mezclar señales sísmicas sintéticas y reales. El modelo propuesto proporciona una estructura eficaz para generar señales sismo-volcánicas sintéticas de alta calidad.

Palabras Clave: volcano-tectónico, largo periodo, redes neuronales adversarias generativas condicionales, señales sintéticas

Abstract

Today we live in a growing era of information and data, which are collected through a myriad of means. These data have a very important weight in different fields of research, as in the study of natural disasters, such as seismic-volcanic events, which aims to protect lives through efficient early warnings. The labeling of seismo-volcanic events, such as Volcano-Tectonic (VT) and Long Period (LP), is a real problem. Visual labeling is not a viable option because of how inefficient and costly it can be.

This research work proposes to design a Conditional Generative Adversarial Neural Network (CGAN) model, which generates synthetic signals of LP and VT seismo-volcanic events, the work consists of three stages: pre-processing, LP and VT seismo-volcanic signals are conditioned. In the CGAN two models are established, the Generator and Discriminator, with different parameters for training and obtaining synthetic frequency responses with characteristics of LP and VT seismo-volcanic events; in the post-processing, synthetic seismo-volcanic signals are obtained in LP and VT time. The synthetic signals are evaluated by a software of detection and classification of seismo-volcanic events, in addition to the visual evaluation of the signals through the experts of the Geophysical Institute of the National Polytechnic School.

The classification accuracy remains stable between 96.5% and 98.5% after mixing synthetic and real seismic signals. The proposed model provides an effective framework for generating high quality synthetic seismo-volcanic signals.

Keywords: volcano-tectonic, long-period, conditional generative adversarial neural networks, synthetic signals

Capítulo I

Introducción

Antecedentes

La monitorización de la actividad volcánica es fundamental para mitigar las posibles consecuencias de las erupciones volcánicas, especialmente en áreas densamente pobladas cerca de volcanes activos; como en el cantón el Chaco ubicado en las faldas del volcán Reventador, la ciudad de Baños cerca del volcán Tungurahua, la provincia de Riobamba cerca del volcán Sangay, y en el cantón Latacunga donde se encuentra el volcán Cotopaxi. A medida que avanza la investigación sobre vulcanología en Ecuador, la cantidad de volcanes listados como "potencialmente activos" puede aumentar (Toulkeridis, 2013).

El volcán Cotopaxi se caracteriza por ser un gran estratovolcán de 5.897 msnm de elevación, se lo considera como uno de los volcanes más activos y altos de todo el mundo, se encuentra ubicado en el Parque Nacional Cotopaxi a 60 km del sureste de la ciudad de Quito capital del Ecuador. Se lo identifica por ser uno de los 452 volcanes que componen el cinturón de fuego del Pacífico; aproximadamente el 80% de los terremotos más grandes del mundo ocurren en esta línea (Secretaría Nacional de Gestión de Riesgos, 2015).

El volcán Cotopaxi es uno de los más peligrosos del mundo debido a la creación de corrientes de lodo conocidas como lahares, por sus episodios eruptivos, el Instituto Geofísico de la Escuela Politécnica Nacional (IGEPN) realiza una monitorización continua las 24 horas del día, los 365 días del año para recolectar datos sobre los múltiples cambios que exhibe el volcán Cotopaxi desde 1983. Una red de 59 estaciones equipadas con detectores de lahares, deformaciones por movimiento de magma, medidores de emisión de gases, detectores de terremotos, pluviómetros, cámaras y termopares dan

seguimiento a la actividad sísmica y volcánica del volcán Cotopaxi (IGEPN, 2020). Los datos adquiridos por las estaciones se guardan y evalúan para detectar si el volcán Cotopaxi inicia un proceso eruptivo que pueda alterar el equilibrio del ecosistema ecuatoriano. Los datos obtenidos son analizados por geólogos o vulcanólogos expertos del IGEPN, los cuales se encargan de realizar el etiquetado de forma visual de los eventos registrados, y se procede a guardar en una base de datos.

Dado que cada señal debe interpretarse por separado, la tarea de etiquetar las actividades sísmico-volcánicas requiere una cantidad considerable de capital humano y tiempo, lo que puede desencadenar una supresión de información si existe una alta actividad sísmica en el volcán Cotopaxi. Por lo tanto, es fundamental que la información recopilada sea analizada y categorizada automáticamente, lo cual minimiza el tiempo requerido para el etiquetado correcto de las actividades sísmico-volcánicas.

En los últimos años, se han desarrollado una variedad de métodos basados en el aprendizaje automático para abordar el problema de la clasificación de eventos sismo-volcánicos, que incluyen: bosque aleatorio (Rodgers et al., 2016) (N. Perez et al., 2020), árboles de decisión (Lara-Cueva et al., 2016), máquina de vectores de soporte y redes neuronales (Lara-Cueva et al., 2016) (N. Perez et al., 2020), especialmente modelos de perceptrones multicapa (Curilem et al., 2009)(N. Perez et al., 2020). Los métodos automáticos y efectivos de predicción de terremotos se han vuelto cada vez más importantes, ya que la cantidad de datos sísmicos ha aumentado considerablemente. Las arquitecturas de redes neuronales profundas han sido efectivas para detectar objetos e identificar patrones a través del uso de técnicas de aprendizaje automático. Debido a la creciente capacidad informática hoy en día, las redes neuronales convolucionales se han mostrado prometedoras en la visión por computadora, el procesamiento de imágenes y señales, entre otros campos de investigación.

Justificación

Las erupciones volcánicas desencadenan una secuencia de eventos físicos que, además de presentar varios riesgos para la civilización y producir variaciones en el equilibrio del ecosistema, atraen la atención de la comunidad científica encargada del estudio de la sismología. El análisis de la sismicidad de una región volcánica se encuentra en una monitorización continua, esto permite recopilar conocimientos sobre las características y patrones de actividad de la región de manera comprensible. Los volcanes producen una gran cantidad de señales sísmicas, las cuales son difíciles de dar seguimiento a través de diferentes factores. Existen distintos tipos de eventos sismo-volcánicos como lo son los eventos Volcano-Tectónico (VT), Largo Periodo (LP), Híbridos, Regionales y *Icequakes*. Además de los eventos mencionados existen otros que no se producen con frecuencia (por ejemplo, los de otra naturaleza, como los rayos) y que, además, no se manifiestan en todos los volcanes (Malfante et al., 2018)

Los conjuntos de datos masivos de señales volcánicas y sísmicas ahora están disponibles para la comunidad gracias a un aumento en el número de observatorios sísmicos, una mayor autonomía de las estaciones de registro y a la disponibilidad de nuevos sensores. Adicionalmente, la creciente popularidad de la implementación de equipos a través del Internet de las Cosas (IoT, del inglés *Internet of Things*) facilita la conectividad entre estaciones de monitorización y centrales de análisis de datos. Como resultado se tiene un gran número de señales que requieren ser estudiadas y clasificadas. Debido a una inspección y clasificación visual que tradicionalmente se realiza en los centros de datos se provoca que a la larga este método no llegase a ser óptimo y factible. Adicionalmente, dado que cada señal debe interpretarse por separado, la tarea de etiquetar las actividades sismo-volcánicas requiere una cantidad considerable de capital humano y tiempo.

La gran mayoría de los observatorios volcánicos aún procesan sus datos visualmente ya que se enfatiza la importancia de los modelos en pleno funcionamiento. La tarea es particularmente compleja y algunas señales son extremadamente difíciles de analizar debido a que los expertos cambian su comprensión y análisis de las señales a lo largo del tiempo.

Se han considerado nuevos enfoques, como el uso de *Machine Learning* para la clasificación de las señales obtenidas. Lo anterior ha demostrado ser extremadamente efectivo en una amplia gama de campos, incluido el reconocimiento de imágenes, voz, robótica, análisis de datos entre otros (Hastie et al., 2009). Sin embargo, las bases de datos de señales sismo-volcánicas etiquetadas liberadas al público y utilizadas para alimentar los modelos de clasificación son escasas debido a la falta de procesamiento y etiquetado de las mismas.

Por consiguiente y a través de la búsqueda de alternativas en la clasificación de las señales que se han realizado en el volcán Cotopaxi, el proyecto se enfoca en la generación de señales sintéticas de eventos sismo-volcánicos a través de un modelo de red neuronal adversario generativo condicional (CGAN, del inglés *Conditional Generative Adversarial Network*).

La CGAN comprende una arquitectura para entrenar modelos generativos basados en el aprendizaje profundo, esta técnica es capaz de aprender de muestras por medio de un conjunto de entrenamientos, con ellos es capaz de producir nuevas muestras de datos donde tienen la misma configuración estadística que las muestras utilizadas para el entrenamiento (Mirza & Osindero, 2014). Con la generación de señales sintéticas se busca enriquecer las bases de datos actuales con señales confiables para que se presten en beneficio del estudio de la sismología en el Ecuador a través del IGEPN y de sus colaboradores.

Alcance del Proyecto

En el presente proyecto se desarrolla una aplicación en el software MATLAB donde se realiza el análisis, tratamiento y generación de señales sintéticas a través del entrenamiento de un modelo CGAN. Como punto de partida se realiza un estudio sobre las señales sismo-volcánicas procedentes del volcán Cotopaxi para después analizar el estado del arte de la generación de señales sintéticas de una dimensión en una CGAN.

Para entrenar el modelo se emplea la respuesta en frecuencia de los eventos sismo-volcánicos LP y VT pertenecientes al *dataset* MicSigV1 provisto por el IGEPN. Una vez entrenado, el modelo CGAN puede generar formas de onda sísmicas sintéticas con la etiqueta del tipo de evento respectiva. Las señales resultantes se generan a través de la implementación del modelo entrenado en una interfaz gráfica creada en MATLAB por su facilidad y versatilidad en el uso de datos.

Finalmente se realiza un protocolo de pruebas mediante distintos métodos. El primero, a través de un detector en el que se evalúa y clasifica el tipo de evento detectado. El segundo, mediante un autoencoder y el tercero mediante la categorización visual del evento a través de la inspección realizada por expertos del IGEPN.

Objetivos

Objetivo General

Generar señales sintéticas de eventos sismo-volcánicos a través de un modelo de red neuronal generativo adversario condicional (CGAN).

Objetivos Específicos

- Preprocesar las señales de las bases de datos de eventos sismo-volcánicos proporcionados por el IGEPN.
- Implementar una arquitectura de un modelo de red neuronal basada en un generador y discriminador.

- Monitorizar el proceso de entrenamiento de la CGAN, de forma que se pueda identificar y corregir posibles fallas de convergencia.
- Generar una base de datos de señales sintéticas de eventos sismo-volcánicos que se encuentren etiquetados.
- Evaluar la base de datos sintética con herramientas de software y con los expertos del IGEPN.

Trabajos Relacionados

Se han realizado una serie de trabajos de investigación los cuales se encuentran relacionados al uso de la GAN, para producir voz sintetizada a partir de la entrada de texto como lo realizan en Yang et al. (2017), por medio de la GAN el autor determinó si la entrada es de habla natural o tiene alguna condición específica, con ello se obtuvo resultados donde la arquitectura genero habla más natural, las cuales satisfacían a la percepción humana a comparación de los métodos convencionales que se han utilizado.

En el proyecto denominado WaveGAN, los autores utilizan una arquitectura de red generativa contradictoria capaz de sintetizar audio (Donahue et al., 2018). La estructura de la red es extremadamente similar a la llamada Red Adversarial Generativa Convolucional Profunda, en la cual se utiliza capas convolucionales tanto en el generador como en el discriminador, se puede observar una amplia variedad de dominios de audio, como el habla humana, los sonidos de los pájaros y la música, con resultados bastante convincentes. Uno de los dominios más difíciles de sintetizar es sin duda el habla humana, uno de los aspectos importantes es que la voz humana es para el audio lo que los rostros humanos son para las imágenes. Es realmente difícil para una máquina aprender a generar estos dominios mientras los mantiene indistinguibles de los reales.

En sismología, los investigadores también han aplicado GAN a varios problemas existentes. En Li et al. (2018), se entrenó una GAN para mitigar los problemas de falsas

alertas tempranas de terremotos. Se usaron 700,000 formas de onda recolectadas en el sur de California y Japón, se entrenó a una GAN para aprender las propiedades de las ondas P del terremoto de primera llegada, con el fin de determinar si las formas de onda son generadas por terremotos o fuentes de ruido locales. Los investigadores demostraron que un discriminador de aprendizaje automático profesional puede detectar el 99,2 por ciento de los terremotos y el 98,4 por ciento del ruido de fondo. Este discriminador aumento en gran medida la solidez de los sistemas de alerta temprana al reducir el número de alertas falsas.

El proyecto de investigación denominado *Seismic Waveform Synthesis Using Generative Adversarial Networks* (SeismoGen) (Wang et al., 2020), desarrolla un modelo CGAN, que produce series de tiempo sísmicas sintéticas. Aunque la GAN se han utilizado en tareas de aumento de datos en el pasado (Perez & Wang, 2017), no se conoce ninguna aplicación de datos sintéticos producidos por GAN para problemas de aumento de datos para series de tiempo 1D o tareas de detección de eventos sísmicos. En el proyecto SeismoGen verifican visual y cuantitativamente la precisión de los eventos sísmicos sintéticos, y obtienen muestras sísmicas sintéticas de alta calidad, las señales generadas mejoraron significativamente los modelos para la detección y clasificación de terremotos en el volcán de Oklahoma.

Mientras que, con el fin de generar la respuesta de frecuencia de magnitud de eventos sismo-volcánicos LP y VT del volcán Cotopaxi, en el artículo ESeismic-GAN utilizan el modelo de Redes adversarias generativas convolucionales profundas (DCGAN, del inglés *Deep Convolutional Generative Adversarial Networks*). Los autores se enfocaron en crear respuestas de frecuencia de magnitud, luego combinar estas respuestas de magnitud creadas con respuestas de fase genuinas y obtener nuevas señales volcánicas con la transformada inversa rápida de Fourier. Se utilizó 2 tipos de evaluaciones, la inspección visual y la distancia de Frechet, con los dos métodos al

comparar las señales reales con las sintéticas se concluyó que las señales sismo-volcánicas sintéticas generadas LP tienen mejores resultados que las VT, donde logró demostrar que las GAN se pueden emplear en el campo volcánico (Grijalva et al., 2020).

Organización del Trabajo

El presenta trabajo consta de cinco capítulos, en donde se explica cada una de las etapas a realizar para el presente proyecto. El primer capítulo proporciona una introducción concisa al estudio, así como justificación e importancia. También describe el alcance del proyecto, los objetivos y los trabajos previos relevantes para el tema de investigación.

En el segundo capítulo describe el marco teórico de la investigación, se cubre los conceptos clave que deben entenderse para comprender el mecanismo que se aplicaría. En este capítulo se ubica el levantamiento del estado de arte tanto de la teoría del aprendizaje no supervisado como de la aplicación de modelos generativos en distintos campos de investigación, además del análisis de la generación de señales sintéticas temporales de una dimensión a través de una CGAN.

En el tercer capítulo se presenta el tratamiento de las señales sísmico-volcánicas pertenecientes a la base de datos MicSigV1 del IGEPN, la metodología utilizada para la implementación del modelo CGAN, además se describe el entrenamiento realizado a la CGAN con la base de datos de eventos sismo-volcánicos y se monitorizan de forma que se pueda corregir posibles fallas de convergencia.

Finalmente, en el quinto capítulo se presentan la base de datos de señales sintéticas de eventos sismo-volcánicos generada con la CGAN y el análisis de similitud de las señales de eventos sismo-volcánicos generadas con las señales originales del Volcán Cotopaxi, además se realiza una discusión de los hallazgos obtenidos en este trabajo de investigación, se detallan las conclusiones y propuestas obtenidas y se recomiendan estudios adicionales que complementen este trabajo de investigación.

Capítulo II

Marco Teórico

Eventos sísmicos en volcanes

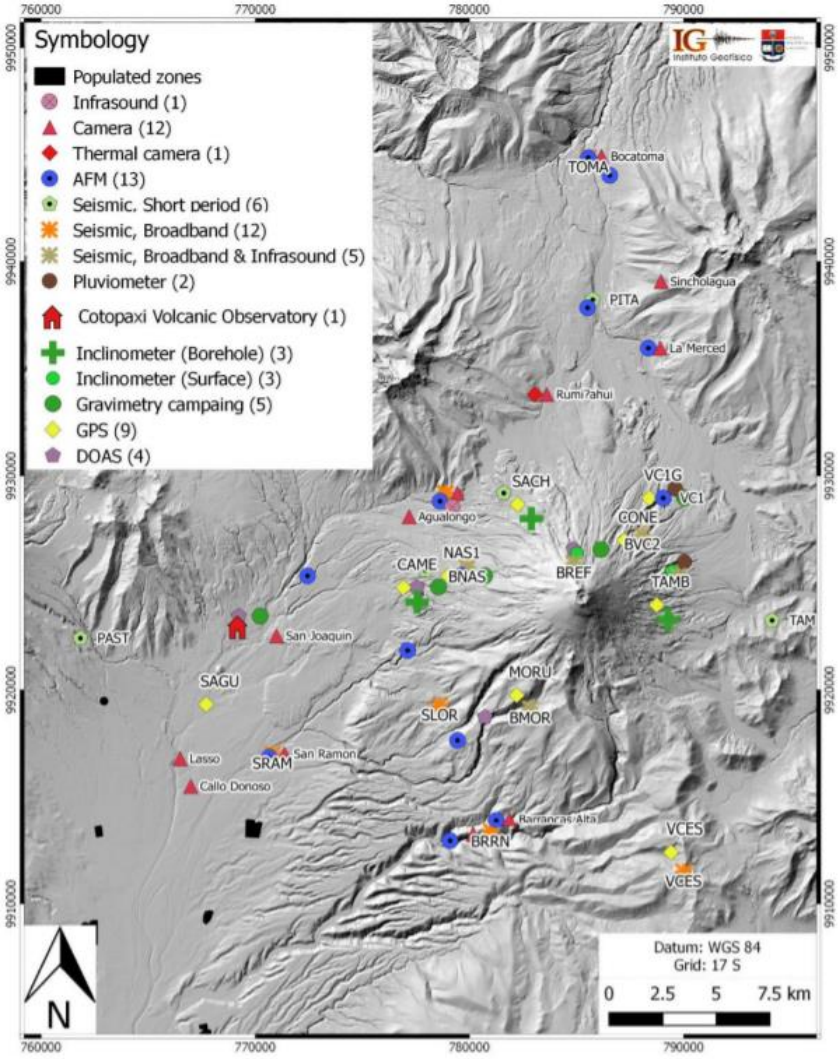
El Volcán Cotopaxi

Ecuador es un país con características geológicas y geodinámicas extremadamente distintas, una de las cuales es un vulcanismo muy activo que ha tenido implicaciones a largo plazo para las comunidades que viven cerca del volcán. Uno de los volcanes más activos y potencialmente destructivos del país es el volcán Cotopaxi, el cual está ubicado en la provincia del mismo nombre (IGEPN, 2020). Desde 1977, este volcán ha mostrado una actividad eruptiva significativa y es uno de los volcanes mejor monitorizados del país por el IGEPN. La presencia de ciertas estaciones geofísicas en este volcán indica la existencia de una base de datos grande y creciente que ha capturado muchos períodos de actividad en este volcán.

En el volcán Cotopaxi se encuentran 16 estaciones sismológicas, las cuales se clasifican en dos grupos, cinco estaciones se las denomina sismológicas de periodo corto (SP, del inglés *Short-Period*), poseen sensores con un rango de frecuencia de respuesta de 1-50 Hz, las 11 estaciones faltantes son las sismológicas de banda ancha (BB, del inglés *Broadband*), con un rango de frecuencia de respuesta de 0.1-25 Hz (R. A. Lara-Cueva, Benítez, et al., 2016). Las diferentes estaciones de las puede observar en la Figura 1.

Figura 1

Sistema de monitorización de actividad sísmica del Volcán Cotopaxi



Nota. Estaciones de monitorización ubicadas en el volcán Cotopaxi. Adaptado de *Volcán Cotopaxi - Instituto Geofísico - EPN* (https://www.igepn.edu.ec/).

Este sistema de monitorización está operativo las 24 horas del día, los siete días de la semana. El sistema registra sismogramas en archivos de 12000 segundos que se digitalizan con un convertidor de analógico a digital de 12 bits a una frecuencia de muestreo de 100 Hz (SP) y 50 Hz (BB), respectivamente.

Sismicidad. La sismicidad es una de las características de los volcanes estrictamente reguladas, debido a que la reactivación de un volcán o el inicio de una fase eruptiva incluye el movimiento o emisión de material volcánico como lo son el magma o los gases, este fenómeno genera vibraciones en la estructura volcánica que pueden detectarse mediante sismómetros, por un incremento en el número de tales ocurrencias implica un aumento de la actividad volcánica, esta monitorización se realiza de forma continua. El creciente volumen de datos volcánicos de varios tipos ha llevado al desarrollo de una serie de enfoques destinados a predecir erupciones volcánicas (McNutt, 1996). Los cambios en las características de sismicidad pueden convertirse en antecedentes, es decir, indicaciones previas de una nueva actividad eruptiva; el estudio y manejo de estos factores ha hecho factible pronosticar erupciones volcánicas con cierto éxito en situaciones particulares (Chouet, 1996). En este contexto, se menciona que la sismicidad es una técnica útil para determinar los patrones de actividad sísmica que permiten determinar la probabilidad de una erupción de manera oportuna.

Eventos sismo-volcánicos

Una variedad de movimientos sísmicos denominados eventos sismo-volcánicos ocurren dentro de un volcán y pueden proporcionar información sobre la actividad del volcán si se registran y evalúan adecuadamente. Aunque todavía no es posible predecir la erupción de un volcán con certeza, es fundamental comprender la historia eruptiva del volcán y realizar una monitorización continua de los eventos sismo-volcánicos, con el fin de mejorar los sistemas de alerta temprana del volcán en tiempo real y así reducir el impacto de un fenómeno eruptivo (Fournier D’albe, 1979).

Por estas razones, se requieren enfoques efectivos y nuevas tecnologías para monitorizar, mitigar y prevenir los peligros asociados con la actividad volcánica. Muchos enfoques actuales se basan en el análisis manual de los diversos parámetros obtenidos mediante el análisis y estudio de los volcanes; sin embargo, debido a la gran variedad y

cantidad de datos obtenidos por los sensores ubicados en toda la estructura del volcán, el análisis visual ya no es una opción viable, lo que nos lleva a buscar un método automatizado que permita una monitorización a mayor escala. Los vulcanólogos y / o geólogos expertos pueden comprender la actividad interna del volcán mediante la evaluación de los eventos sísmicos recopilados. Esto se logra al escanear datos sísmicos en busca de señales de actividad volcánica que puedan estar conectadas con un cierto tipo de actividad volcánica (Nugroho & Winarko, 2011).

Tipos de Eventos

Según la forma, frecuencia de las señales sísmicas y como su origen potencial, los eventos sismo-volcánicos se han clasificado principalmente en las siguientes categorías:

- Eventos Volcano-Tectónico (VT),
- Eventos Largo Periodo (LP),
- Tremores volcánicos (TRE, del inglés *Tremor*),
- Eventos Híbridos (HYB, del inglés *Hybrid*) comparten algunas características con eventos tipo VT y LP,
- Rupturas de glaciar (ICE, del inglés *Icequakes*), se presentan en volcanes con glaciares,
- Eventos de muy largo período (VLP, del inglés V*ery Long Period Events*) que son comparables a los LP, pero con frecuencias características incluso más bajas que los LP típicos,
- Eventos regionales, son eventos sismo-volcánicos los cuales ocurren lejos del volcán, pero son lo suficientemente grande para ser detectados (Sigurdsson et al., 2015).

Los eventos LP están asociados con la resonancia de grietas causada por deformaciones en la superficie del volcán. Las formas de onda de estos eventos tienen una forma cónica con un espectro restringido confinado en bandas de frecuencia entre 0.5 y 5 Hz, estos eventos tienen una duración típica la cual se encuentra entre 5 y 40 segundos (Bean et al., 2014).

Los eventos VT están relacionados con cambios de tensión causados por el flujo de magma y fallas creadas por la roca. Las fases P y S de las señales VT tienen una gran amplitud y su rango de frecuencia es de 5 a 15 Hz.

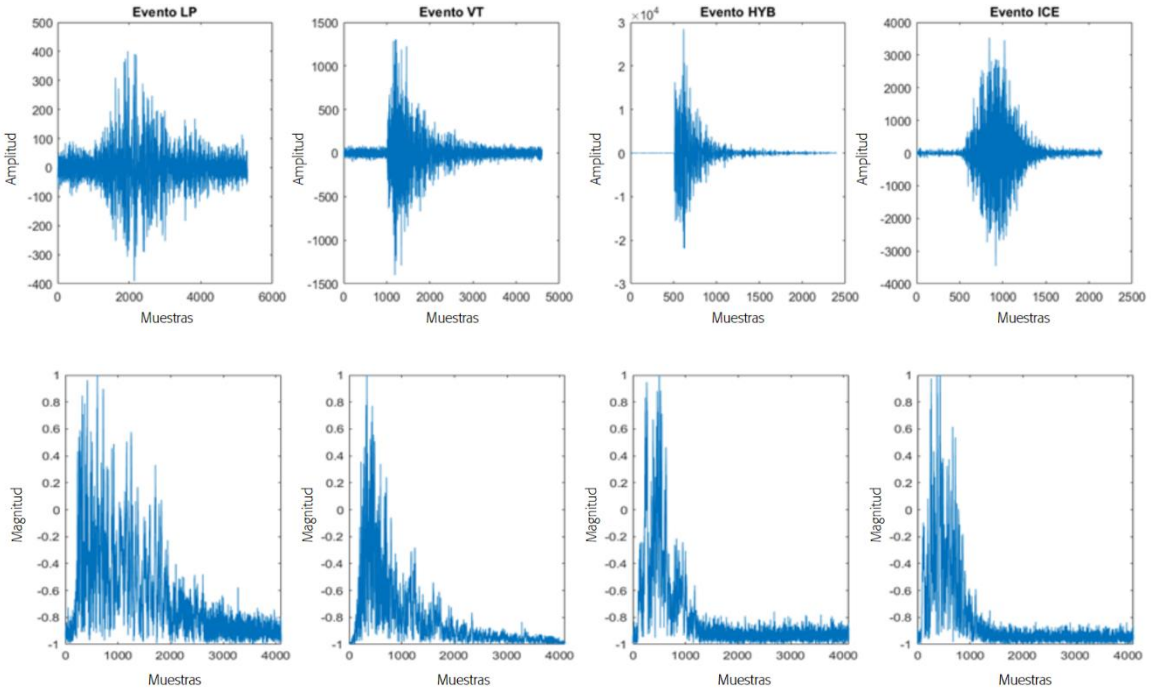
Los eventos TRE se caracterizan por ser señales frecuentes en volcanes, las cuales tienen una amplitud constante y se producen por un largo periodo de tiempo. La duración de estas señales puede variar desde unos pocos minutos hasta muchos días, lo que las convierte en una ocurrencia relativamente complicada. Además, si la frecuencia de un TRE es superior a 6 Hz, puede ser que se trate de un deslizamiento de tierra o una avalancha, pero si la frecuencia es inferior a 0,5 Hz, se considera que la señal se deba a la desgasificación lo cual está relacionada con la actividad propia del volcán (Sigurdsson et al., 2015).

A veces, un evento volcanotectónico (VT) desencadena un evento de período largo (LP), y viceversa. Una señal sísmica que presenta una mezcla de ambos tipos se denomina híbrida (HYB). Este evento híbrido es un evento VT que desencadenó un evento LP. Los eventos con rupturas de glaciar (ICE) se presentan en volcanes con glaciares,

La forma de onda de una señal sismo-volcánica obtenida del volcán Cotopaxi se observa en la Figura 2. Se proporcionan cuatro observaciones, una para cada uno de los cuatro eventos: LP, VT, HYB y ICE respectivamente.

Figura 2

Formas de onda de eventos sismo-volcánicos capturados por el sistema de monitorización del volcán Cotopaxi



Aprendizaje de Máquina

En el año de 1959, Samuel (1959) definió al aprendizaje de máquina como “Un campo de estudio en el que se le da a las computadoras la habilidad de aprender sin estar algo explícitamente programado”. En las últimas décadas, el aprendizaje de máquina se ha convertido en uno de los principales enfoques de estudio de las tecnologías de la información y computación. Un algoritmo de aprendizaje de máquina posee la habilidad de aprender a través de un conjunto de datos.

Mitchell (1997) da una definición más formal de lo que es el aprendizaje de máquina en el que: “Un programa tiene la capacidad de aprender dada una experiencia E, con respecto a una serie de tareas T y una medida de rendimiento P, si su rendimiento en las tareas (denotado con P) mejora con la experiencia E”. Cada una de

estas variables es esencial para el funcionamiento estructural de un algoritmo de aprendizaje de máquina.

Tipos de Tareas

Una tarea T forma parte del resultado de haber aprendido en un algoritmo. Por ejemplo, si se desea que un robot tenga la destreza de hablar, entonces el habla es la tarea en sí. Si se usa la programación tradicional, se puede lograr que un robot pueda hablar, sin embargo, se le puede dar la habilidad de aprender esa tarea a través de aprendizaje de máquina. Estas tareas permiten que un sistema de aprendizaje de máquina pueda procesar un *example*, donde este es un conjunto de características que han sido previamente medidas y procesadas de algún objeto u evento del cual se desee realizar un algoritmo de aprendizaje de máquina. Un *example* puede ser una serie de características que define a un grupo de imágenes (Goodfellow et al., 2016).

Clasificación

La clasificación es empleada para predecir la pertenencia de una categoría a un grupo de instancias de datos de entrada. Si bien es muy empleada en varios campos de investigación sufre de ciertas falencias, una de ellas es la de manejar un conjunto de datos con datos faltantes o insuficientes (Soofi & Awan, 2017). Específicamente, a un algoritmo de aprendizaje de máquina se le solicita especificar las k categorías pertenecientes a un dato de entrada. Matemáticamente, un algoritmo produce una función f tal que $f: \mathbb{R}^n \rightarrow \{1, \ldots, k\}$. Si $y = f(x)$, el modelo asigna a una entrada descrita por el vector x una categoría identificada por el resultado numérico que produce y (Goodfellow et al., 2016). Hay un sinfín de aplicaciones entre las cuales se destacan el reconocimiento visual de objetos o reconocimiento de señales, como puede ser la detección de eventos sismo-volcánicos (Altamirano, 2021).

Regresión

En la regresión, a un algoritmo de aprendizaje de máquina se le solicita predecir un valor numérico dada una entrada. Se define matemáticamente como la salida de una función f tal que $f: \mathbb{R}^n \rightarrow \mathbb{R}$. Se observa que tiene una similitud con la clasificación, sin embargo, el tipo de salida es diferente (Goodfellow et al., 2016). Existen algunos ejemplos de regresión; como es el caso de la predicción de enfermedades críticas del tipo cardiovasculares, diabetes e hipertensión (Nusinovici et al., 2020).

Síntesis y muestreo

En este tipo de tareas, los algoritmos de aprendizaje de máquina generan nuevos valores similares a los datos de entrada. A estos sistemas se los entrena con una gran cantidad de datos con el fin de que se puedan recopilar características similares entre sí para que sea útil en la generación de nueva información. Es popular el uso de redes GAN en este tipo de tarea (Goodfellow et al., 2020).

Rendimiento de un algoritmo de aprendizaje de máquina

Es importante establecer una medida cuantitativa que permita definir el porcentaje de rendimiento que posee un modelo de aprendizaje de máquina, a esto se lo conoce como tasa de error o precisión. Puede ser difícil en algunos casos el establecer o definir si un modelo realiza un proceso adecuado, por lo que en una gran mayoría de casos no es técnicamente posible obtener un error del cero por ciento. Varios modelos pueden ser afectados por diversos factores, entre los que está la cantidad de datos o incluso la potencia computacional que se disponga. En algunos casos es necesaria la intervención humana para definir si un resultado es válido o no (Goodfellow et al., 2016).

Tipos de Aprendizaje

Los algoritmos de aprendizaje automático también pueden clasificarse como supervisados y no supervisados, que influyen como una retroalimentación en el tipo de experiencia que se les permite tener durante el proceso de aprendizaje (Russell, 2021).

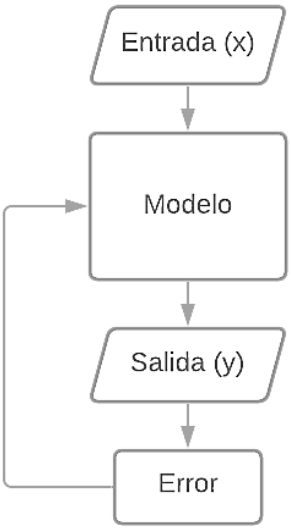
Aprendizaje Supervisado

En el aprendizaje supervisado, un sistema observa tanto la entrada como la salida y aprende una función que es asignada desde la entrada a la salida. Un ejemplo simple, que lo define Russell (2021), es una serie de imágenes de cámaras que funcionan como entradas, cada una de ellas acompañada de una salida que diga "autobús" o "peatón", etc. A esta salida se la denomina etiqueta. Este sistema o modelo aprende una función que, ante una nueva imagen, predice la etiqueta adecuada.

Dentro de este tipo de experiencia se ubica en su mayoría modelos clasificadores y regresiones. Obsérvese en la Figura 3 una representación del funcionamiento de un algoritmo con aprendizaje supervisado.

Figura 3

Diagrama de flujo de un modelo de aprendizaje supervisado



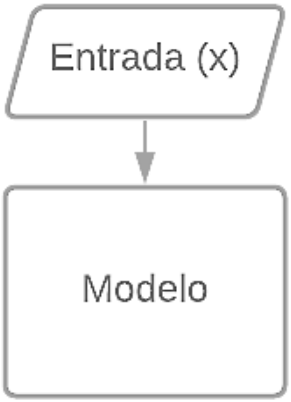
Aprendizaje No Supervisado

En el aprendizaje no supervisado, un modelo aprende patrones en la entrada sin ninguna retroalimentación explícita, es decir, no le es entregada ningún tipo de información al algoritmo ya que el mismo debe ser capaz de encontrar una función que esté acorde a la entrada del sistema. El aprendizaje no supervisado se vence a sí mismo ya que descubre patrones ocultos en los datos.

Según Murphy (2012) este es un problema mucho menos definido, ya que no se tiene conocimiento del tipo de patrones que hay que buscar, y no hay una métrica de error obvias que utilizar (a diferencia del aprendizaje supervisado, en el que se puede comparar una predicción de salida para una entrada dada). Dentro de este tipo de experiencia se ubican los modelos generativos adversarios. En la Figura 4 se representa el diagrama de funcionamiento de un algoritmo de aprendizaje no supervisado. Cabe aclarar que esta representación es generalizada, ya que puede variar con respecto al problema que se pretende solucionar, como un modelo generativo o un clúster.

Figura 4

Diagrama de flujo de un modelo de aprendizaje no supervisado



Aprendizaje Profundo

El aprendizaje profundo (DL, del inglés *Deep Learning*) se basa en un conjunto de algoritmos que intentan modelar abstracciones de alto nivel en los datos. Estos algoritmos desarrollan una arquitectura jerárquica de aprendizaje y representación de datos en capas. Esta arquitectura de aprendizaje jerárquico emula el proceso de aprendizaje profundo y por capas de las áreas sensoriales primarias del neocórtex del cerebro humano, que extrae automáticamente características y abstracciones de los datos subyacentes.

Según Goodfellow et al. (2016) el aprendizaje profundo tiene la facultad de extraer información de alto nivel y abstracto desde datos en bruto. Por ejemplo, información como el acento de un idioma, las formas geométricas que identifican un objeto en una imagen, entre otras, solo podrían ser identificadas a través de una comprensión humana de los datos. En este sentido, este paradigma soluciona este problema central, permitiéndole a una computadora construir conceptos complejos a partir de conceptos más simples. Todo esto es posible a través de las capas de las redes neuronales artificiales. En su forma simple, una red neuronal artificial posee tres capas (entrada, oculta y salida) por lo que una red neuronal profunda estaría compuesta por una capa de entrada, una de salida y varias capas ocultas de por medio, de forma que esto sustenta el concepto de aprendizaje profundo.

Red neuronal artificial

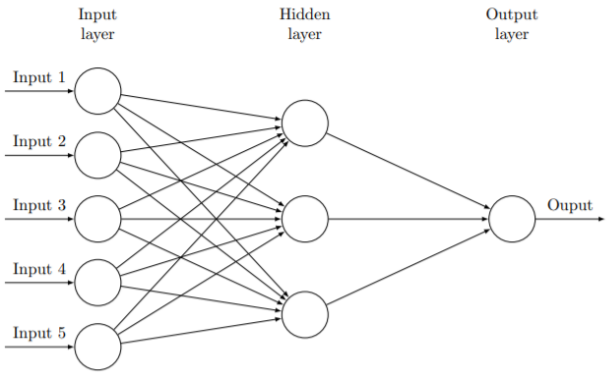
Las redes neuronales artificiales (ANN, del inglés *Artificial Neural Networks)* son algoritmos de aprendizaje automático los cuales se introdujeron como modelos simplificados del funcionamiento del cerebro humano y, por tanto, intentan imitarlo (Parsons, 2005). Una ANN es una función que traduce sus entradas en salidas, donde las entradas numerosas variables como imágenes, números, vectores o matrices de datos y la salida, en la mayoría de las situaciones, una probabilidad.

ANN es una agrupación de capas como se puede observar en la Figura 5. Las capas se utilizan para aumentar la capacidad de complejidad de la ANN; con más capas, la red es capaz de aprender más niveles de abstracción, como se sugiere en (Zeiler & Fergus, 2014). Una capa está formada por neuronas, donde todas las neuronas de la capa i están vinculadas a todas las neuronas de la capa $i + 1$, estas conexiones se denominan pesos. Perceptrón es el nombre que se le da a una neurona que tiene pesos de entrada. Según Parsons (2005**)**, un solo perceptrón fue una de las primeras redes neuronales artificiales.

En la Figura 5, se puede observar que una ANN tiene tres tipos diferentes de capas: la capa de entrada, es la que almacena los datos de entrada de la red, y la capa de salida, donde se realiza la salida de predicción de la red y las capas ocultas, encargadas de la detección de estructuras complicadas en los datos.

Figura 5

Red neuronal artificial multicapa

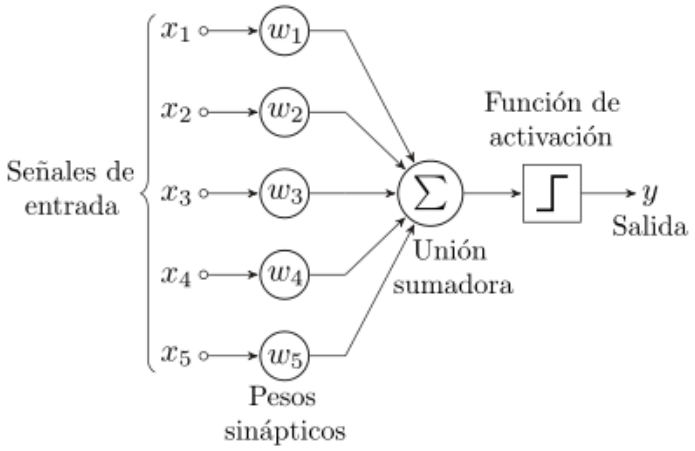


Nota. Capas de una red neuronal. Adaptado de Deep *Learning* (p. 324), por Goodfellow et al., 2016.

Las ANN están formadas por múltiples capas de perceptrones que están conectados en una cascada de retroalimentación. En la Figura 6 se observa modelo de perceptrón, un nodo recibe señales (múltiples) en su lado izquierdo.

Figura 6

Diagrama de nodo (modelo de perceptrón)



Nota. Modelo de un perceptrón Adaptado de The *Knowledge Engineering Review* (p. 432), por Parsons, S. 2005.

A continuación, las señales se suman y el valor resultante se transmite a la función de activación (más información sobre las funciones de activación se presentan en la sección 2.9.3). La salida se muestra en el lado derecho. La señal fluye a través de las conexiones formadas por las líneas que conectan los nodos. Matemáticamente, la figura anterior se puede traducir como:

$$y = f\left(a + \sum_{i=1}^{N} x_i\, w_i\right),$$

donde f es la función de activación y a es conocido como el *bias*. El *bias* tiene la propiedad única de no tener entrada y una salida, siempre es igual a uno. Además, se puede vincular un máximo de un *bias* a un nodo con un peso desde el sesgo hasta la entrada.

Tipos de ANN

Las redes de retroalimentación, redes neuronales recurrentes y las redes neuronales convolucionales (CNN, del inglés *Convolutional Neural Networks*) son ejemplos de métodos o arquitecturas ANN.

Redes de retroalimentación. Los nodos de una capa se encuentran conectados solo a los nodos de la siguiente capa (Gupta, 2017). La información fluye directamente desde los nodos de entrada a los nodos de salida sin conexiones de retroalimentación en las que las salidas del modelo se retroalimentan a sí mismo.

Redes neuronales recurrentes. Es similar a una red neuronal de retroalimentación, pero se diferencia en que tiene al menos un circuito de retroalimentación. Estas conexiones de retroalimentación devuelven las salidas de ciertos nodos de la red a las capas o nodos de entrada para ejecutar cálculos repetidos (Gupta, 2017).

Redes Neuronales Convolucionales

Las CNN son un tipo de red neuronal multicapa y están inspirados en la percepción visual de las cosas. Dos neuropsicólogos Hubel & Wiesel (1968) experimentaron y descubrieron que las neuronas dentro del cerebro de un gato son organizadas a través de capas. Estas capas aprenden a reconocer patrones visuales a través de la extracción de características y luego combinándolas para una posterior representación de esas características en un nivel superior, este uno de los principios básicos del aprendizaje profundo.

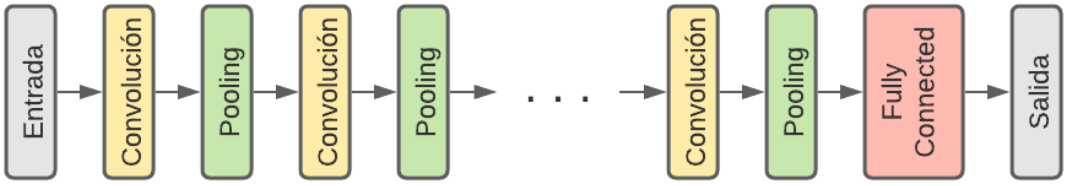
Una CNN posee una arquitectura del tipo *feed-forward*, que según Ghosh et al. (2020) puede aprender características muy abstractas de los objetos, especialmente los datos espaciales. Un modelo puede aprender varias características de los datos de entrada (por ejemplo, una imagen) con múltiples niveles de abstracción. Las capas

iniciales aprenden y extraen las características de alto nivel (con menor abstracción) y las capas más profundas aprenden y extraen las características de bajo nivel (con mayor abstracción). En la Figura 7 se muestra un diagrama de bloques del funcionamiento de manera conceptual de un modelo CNN (Sultana et al., 2018).

Un modelo típico de CNN está compuesto de uno o varios bloques de convolución, cada uno seguido de una capa de *sub-sampling* o *pooling*, para posteriormente emplear funciones de activación y tener la capa totalmente conectada o *fully connected* hasta concluir con el resultado final. En la siguiente sección se conoce a detalle cada una de las capas dentro de un modelo CNN.

Figura 7

Diagrama de bloques de un modelo típico de CNN



Nota. Bloques para modelo CNN. Adaptado de "*Advancements in image classification using convolutional neural network*" (p. 123), por Sultana et al., 2018, *2018 Fourth International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN).*

Capa Convolucional

Las CNN tienen como componente principal e indispensable la capa convolucional. La misma es una operación matemática conocida como convolución, en la que se opera la entrada con una serie de filtros (conocido comúnmente en el área de investigación como *kernel*). Matemáticamente, el valor que proviene de la entrada o de una capa anterior se define con una matriz $M(i,j)$ que puede simbolizar inicialmente

una señal o una imagen, $S(i,j)$ el resultado final de la convolución y K es un *kernel*. Cada elemento del *kernel* está dentro de los rangos $-a \leq m \leq a$ y $-b \leq n \leq b$. La convolución está denotada por la siguiente fórmula (Goodfellow et al., 2016):

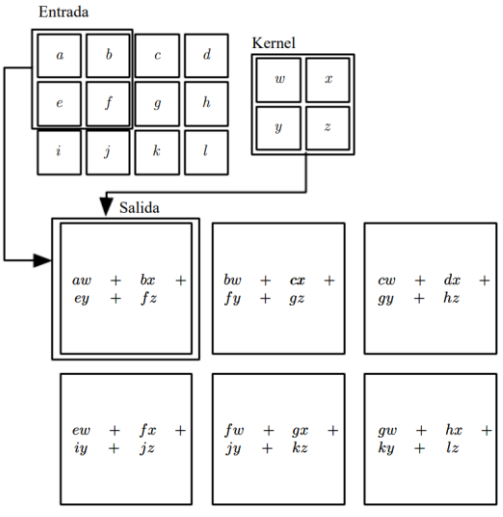
$$S(i,j) = (K * M)(i,j) = \sum_{m=-a}^{a} \sum_{n=-b}^{b} M(i-m, j-n)\, K(m,n)\,.$$

Un *kernel* se define como una matriz de valores discretos, a estos valores se los conoce como “pesos”. Inicialmente durante el proceso de aprendizaje todos estos pesos son asignados con valores aleatorios que se renuevan conforme el modelo se actualice.

A través de la actualización de pesos se extraen características importantes de una imagen o señal. Un ejemplo claro puede ser la detección de bordes o de figuras en una imagen, así como la detección de picos en una señal temporal. En esencia, el principio de la convolución consiste en arrastrar y mover el *kernel* de convolución sobre la matriz. En cada posición se obtiene la convolución entre el *kernel* y la matriz, como se puede observar en la Figura 8.

Figura 8

Ejemplo de una convolución en dos dimensiones.



Nota. Al ser en dos dimensiones este ejemplo funcionaría para un modelo de CNN de una imagen. Adaptado de *Deep Learning* (p. 334), por Goodfellow et al., 2016.

A continuación, el *kernel* se desplaza un número *s* de píxeles, donde s simboliza un *stride*. Usualmente también se suele añadir un relleno con ceros, que es un margen de tamaño p que contiene valores de cero alrededores de la matriz para controlar el tamaño de la salida. Si se asume que se aplicaran C_0 *kernel*, de tamaño $k \times l$, a una matriz de entrada de tamaño $W_i \times H_i \times C_i$ (donde W es el ancho, H el alto y C la cantidad de canales) la salida es finalmente de tamaño $W_o \times H_o \times C_o$ donde C_0 es la cantidad de *kernel* propuestos en cada capa de convolución. Estos valores se pueden calcular a través de las siguientes fórmulas:

$$W_0 = \frac{W_i - k + 2p}{s} + 1 \qquad , \qquad H_0 = \frac{H_i - l + 2p}{s} + 1 \, .$$

Es importante el uso de las fórmulas previas, ya que se va a calcular el tamaño de salida de cada capa con el objetivo de diseñar sucesivamente los siguientes. Según Ghosh et al. (2020) existe una serie de ventajas a la hora de aplicar capas convolucionales en modelos de DL, entre las que están: la conectividad dispersa, dado que el cálculo de los pesos entre las capas convolucionales es eficiente a nivel de memoria por el pequeño número de pesos que existe; y la compartición de pesos en el que no existen valores de pesos dedicados entre capas por lo que todos estos pesos aprenden de forma continua, donde se reduce drásticamente el tiempo de entrenamiento.

Capa de Convolución Transpuesta

Una capa convolucional transpuesta, se le conoce como capa de convolución fraccional, es común en las CNN, se utiliza para el muestreo ascendente del modelo del generador o para producir un mapa de características de salida con una dimensión espacial mayor que el mapa de características de entrada. El *padding* y *stride* definen la capa convolucional transpuesta de la misma manera que construyen la capa convolucional normal (Dumoulin et al., 2018).

En la capa convolucional transpuesta cabe la posibilidad de añadir entre las filas y columnas de la entrada una cantidad de números de ceros denominado z (inserción de ceros) con el fin de aumentar el tamaño de la muestra de entrada $(2 * i - 1) \times (2 * i - 1)$, además se calcula el *padding* p', mismo que se puede observar en la Figura 9:

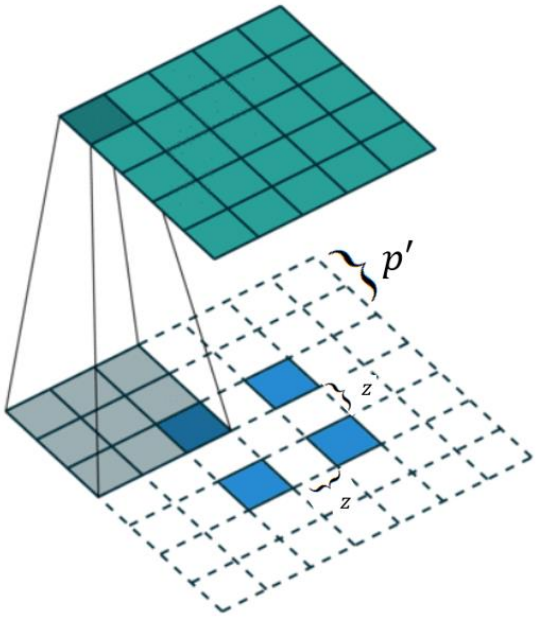
$$z = s - 1 \,,$$

$$p' = k - p - 1 \,.$$

Una vez obtenidos los nuevos valores de inserción de ceros y *padding*, el *stride* en toda la convolución transpuesta tiene el valor de 1.

Figura 9

Convolucional transpuesta con stride de 2



Nota. Ejemplo de convolución transpuesta con Stride igual a 2. Adaptado de *A guide to convolution arithmetic for deep learning* (p. 450) por (Dumoulin et al., 2018).

Se pueden obtener los anchos y altos de la salida por medio de las siguientes ecuaciones:

$$W_0 = (W_i - 1) \times s + k - 2p \,,$$

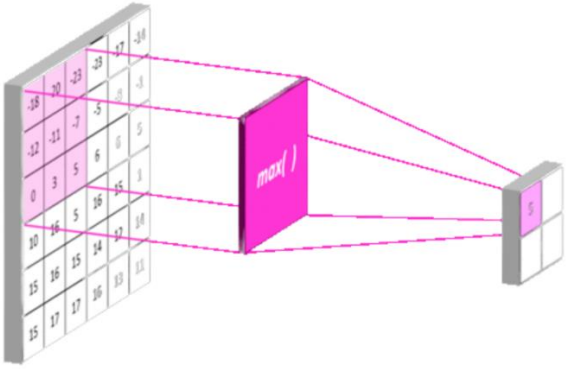
$$H_0 = (H_i - 1) \times s + k - 2p \,,$$

Capa Pooling

Otro componente básico importante en muchos diseños de CNN es la capa *pooling*, también conocido como *subsampling*. Se utiliza generalmente para reducir la resolución espacial de los mapas de características producidos por la capa convolucional en las direcciones de ancho y alto. Se encuentran varios tipos de operador como: *max pooling*, el cual es el más empleado y este escoge el valor máximo de la ventana; el *average pooling*, que escoge el valor promedio de la ventana; y el L^2-*norm pooling* que suma los cuadrados de la ventana. Es posible escoger el tamaño, el *stride* y el tipo de operador. En la Figura 10 se muestra el uso de esta capa en una matriz, nótese que se emplea uno del tipo *max pooling* por lo que escogería el valor máximo en cada iteración. Según Goodfellow et al. (2016) la capa de *pooling* ayuda a que la representación sea aproximadamente invariante a pequeñas diversificaciones procedentes de la entrada, con lo cual mejora el proceso de entrenamiento. Esta capa se suele emplear a continuación de la capa de convolución en un modelo de CNN común.

Figura 10

Capa Pooling del tipo Max Pooling



Nota. Demostración de capa *Max Pooling*. Adaptado de Convolutional Neural Networks – Basics, por R. Robinson, 2017, GitHub Pages (https://mlnotebook.github.io/post/CNN1/). CC BY 2.0

Funciones de Activación

Las ANN implican una transformación no lineal de representaciones por medio de los diferentes niveles. La función de activación es responsable de esta no linealidad. En una CNN, se ubican funciones de activación no lineales después de cada capa de aprendizaje. La función escalonada es la función de activación más antigua conocida. Esta función fue definida por primera vez por Rosenblatt (1957), se utilizó en el perceptrón. La función produce una salida binaria, con uno si la entrada es mayor que un umbral específico y cero en caso contrario, como se muestra en la expresión a continuación:

$$f(x) = \begin{cases} 1, & si\ x \geq C \\ 0, & si\ x < C \end{cases}.$$

La función escalonada ya no se la utiliza como función de activación debido a que no favorece al aprendizaje de la ANN. Existen varias funciones de activación, las cuales son de uso común como la Sigmoide, Tangente Hiperbólico, Unidad lineal rectificada (ReLU del inglés, *Rectified Linear Unit*), Leaky-ReLU (LReLU) y Paramétrico-ReLU (P-ReLU), en la Tabla 1 se pueden observar las definiciones y Rangos para cada una de las funciones mencionadas.

Tabla 1

Funciones de activación no lineal de uso común

Función de Activación	Definición	Rangos
Sigmoid	$\sigma(x) = \frac{1}{1+e^{-x}}$	$(0,1)$
Tanh	$tanh(x) = \frac{e^{x}+e^{-x}}{e^{x}+e^{-x}}$	$(-1,1)$
ReLU	$f(x) = \begin{cases} x, & si\ x \geq 0 \\ 0, & otros\ casos \end{cases}$	$[0,\infty)$
L- ReLU	$f(x) = \begin{cases} x, & si\ x \geq 0 \\ \alpha x, & otros\ casos \end{cases}$	$(-\infty,\infty)$
P- ReLU	$f(x) = \begin{cases} x, & si\ x \geq 0 \\ \omega x, & otros\ casos \end{cases}$	$(-\infty,\infty)$

Nota. α en L- ReLU es un tiempo constante generalmente muy pequeño, mientras que ω P- ReLU, es un parámetro del cual se puede aprender (Yu, 2018).

Los problemas que se presentan en las funciones de activación aparecen si su derivada es menor a uno, también denominado *vanishing gradient* y si la derivada es mayor que uno denominado, *exploding gradient.* Estos dos problemas se pueden evitar con una función de activación cuya derivada, sea constante e igual a uno. Una función que tiene una derivada constante es la función lineal, la función de activación a la cual se usa en la práctica y comparte esta idea es la función de activación ReLU (Nair & Hinton, 2010), la cual viene dada por la fórmula $f(x) = \max\{0, x\}$, además esta función es muy utilizada ya que requiere una carga cálculo mínima en comparación con otras funciones de activación.

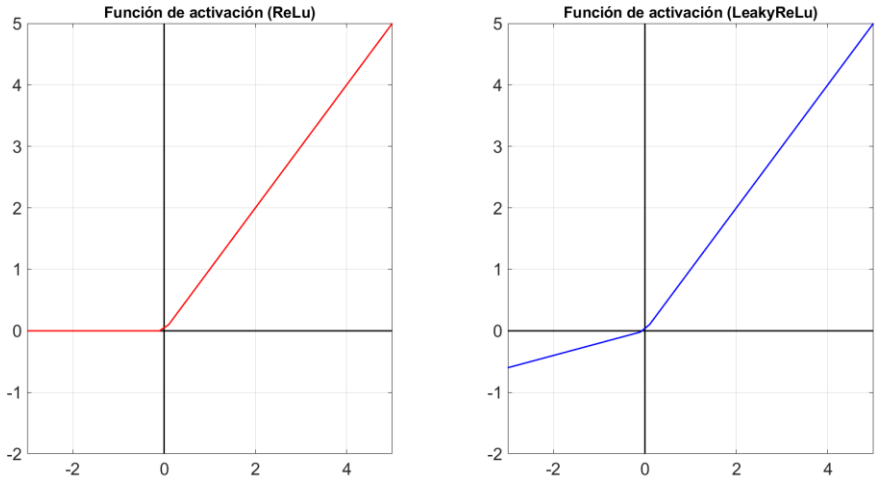
ReLU. Es una función de activación continua con derivada constante. La derivada para entradas positivas es 1 y para entradas negativas es 0. Sin embargo, esta derivada cero puede resultar en un problema conocido como *Dying ReLU*. Al tener una derivada cero, puede significar que los pesos usados no se ajustan durante el

entrenamiento. Las neuronas dejan de responder a los cambios en la entrada si los pesos no cambian. *Dying ReLU* se presenta por gradiente de cero, donde como consecuencia algunas neuronas se extinguirán y no responderán más, lo que hace que una parte sustancial de la red no se vuelva activar. Este problema se lo puede resolver con bastante facilidad, al utilizar una variante denominada Leaky-ReLU.

Leaky-ReLU. Es una función de activación cuya derivada nunca es cero, es decir todas las neuronas tienen un valor, esto en ocasiones puede ser un problema debido a que se provoca una escasez de activación, en una red grande se podría decir que todas las neuronas tendrán normalmente un valor de activación diferente a cero (Maas et al., 2013). La salida de la red se describe como neuronas / activaciones, por lo que la activación es densa y muy costosa, sin embargo, en redes profundas el aporte es positivo. Ambas funciones de activación se pueden observar en la Figura 11.

Figura 11

Funciones de activación ReLU y Leaky- ReLU



(a) (b)

Nota. a) Función ReLU, b) Función Leaky-ReLU

Seleccionar la función de activación adecuada es una tarea difícil, donde cada función de activación tiene su propio conjunto de ventajas e inconvenientes. Sin

embargo, hay algo que decir sobre la selección de la función de activación adecuada. Para redes pequeñas ya que no existen gran cantidad de capas ocultas se utiliza Tangente Hiperbólico, para las redes neuronales profundas la función de activación ReLU se usa principalmente, ya que esta función no tiene el problema del gradiente inestable, además tiene ventajas las cuales aceleran procesos dentro de la red.

Funciones de pérdida

Una función de pérdida es la última capa presente en cada arquitectura CNN del tipo clasificación. Esta capa es responsable de la predicción del error generada por el modelo en sí a través del entrenamiento. El error indica a la red cuan alejada está su predicción de ser real, por lo que este error debe ser optimizado y mejorado durante todo el proceso de aprendizaje. De manera general, la función de pérdida toma en consideración dos parámetros para calcular el error que son: la predicción del modelo (la estimación en sí); y la salida real, conocida como etiqueta (en inglés *label*). Existe un sinfín de funciones utilizadas para estimar el error de un conjunto de pesos de una red neuronal, sin embargo, hay que considerar el escenario de entrenamiento para apuntar y escoger la función que más se adecúe a las necesidades propias del modelo.

Según Goodfellow et al. (2016), se prefiere una función de pérdida donde el espacio de las posibles soluciones pueda ser mapeado en un paisaje suave en el que un algoritmo de optimización pueda navegar sin problemas entre iteraciones para actualizar los pesos de una red neuronal y buscar el máximo ajuste del modelo. Para tal caso existe un marco para encontrar las mejores estimaciones estadísticas de parámetros a partir de los datos de entrenamiento, conocida como la estimación de máxima verosimilitud o (MLE, del inglés *máximum likelihood estimation*). Se tiene un conjunto de datos de entrenamiento con una o más variables de entrada y se necesita un modelo para estimar los parámetros de peso del modelo que mejor mapeen los ejemplos de las entradas a la variable de salida o de destino. Dada la entrada, el

modelo trata de realizar predicciones que coincidan con la distribución de datos de la variable objetivo. En la estimación de máxima verosimilitud, una función de pérdida estima si la medida de distribución de las predicciones realizadas por un modelo coincide con la distribución de las variables objetivo en los datos de entrenamiento. Un beneficio de usar la máxima verosimilitud como marco para estimar los parámetros del modelo (pesos) para las redes neuronales y en el aprendizaje automático en general es que a medida que aumenta la cantidad de ejemplos en el conjunto de datos de entrenamiento, la estimación de los parámetros del modelo mejora.

Se tiene en consideración lo que es la máxima verosimilitud, un error entre dos distribuciones puede ser medida al emplear una función de pérdida del tipo entropía cruzada binaria (BCE, del inglés *binary cross entropy*). En un problema de clasificación binaria se tendría dos clases en el que se puede predecir la probabilidad de que un ejemplo pertenezca a una clase. En tal caso, una probabilidad denotada por p pueda tener una salida tal que $p \in \{0, 1\}$. El BCE es el promedio negativo del logaritmo de las probabilidades predichas de forma correcta. Dicho de forma matemática, el BCE se define a través de la siguiente fórmula de forma muy general:

$$L(p) = -\frac{1}{N}\sum_{i=1}^{N} \log p_i \, .$$

La función de pérdida BCE, donde se considera la probabilidad de que se tengan las dos clases (cero o uno), se define de la siguiente forma:

$$L(p, y) = -\frac{1}{N}\sum_{i=1}^{N} [y_i \log p_i + (1 - y_i) \log(1 - p_i)] \, ,$$

donde: y_i es la etiqueta, p_i es la predicción de 1 y (1-p_i) la probabilidad de la clase 0. El signo negativo al inicio de la ecuación tiene como objetivo el evitar que la pérdida sea negativa de forma que se normalice entre 0 y 1 al igual que la obtención del logaritmo en ese rango, y da como resultado un valor menor a cero. Cabe añadir que esta función

de pérdida necesita valores de probabilidad en el rango de uno a cero por lo que se limita esos valores a través de una función de activación del tipo sigmoide de forma que se interprete el valor de salida como una probabilidad. Esta propiedad es conocida, según Goodfellow et al. (2016), como regresión logística.

Proceso de aprendizaje

En la sección anterior se describen los componentes que forman parte de una modelo de red neuronal convolucional. En esta sección se describen pasos claves para entrenar de forma óptima un modelo. Se describe sobre: el preprocesamiento de los datos; la inicialización de los pesos de las capas convolucionales; la regularización de una CNN; y finalmente, sobre el algoritmo de optimizador de entrenamiento.

Preprocesamiento de los datos

Es crucial el preprocesamiento y tratamiento de los datos previo al entrenamiento. Como primer punto se tiene que el conjunto de datos de entrenamiento debe ser limpio, es decir, que los datos puedan ser entrenados en un formato uniforme y que a su vez estos mantengan una gran cantidad de características (Ghosh et al., 2020). Este es el primer paso antes de desarrollar el modelo en sí ya que, en el caso de una CNN, se requiere establecer un tamaño en la capa de entrada (donde se ingresan los datos entrenables del *dataset*) por lo que se debe considerar este requerimiento para que se preparen los datos en un formato uniforme.

Eliminación de la media (X'). En un conjunto de datos, se resta la media $\bar{x}$ a cada punto, y se tiene como resultado un conjunto de datos donde la tendencia lineal es eliminada. Matemáticamente se describe a través de la siguiente fórmula:

$$X' = X - \bar{x}\,,$$

donde la media $\bar{x}$ es:

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i\,.$$

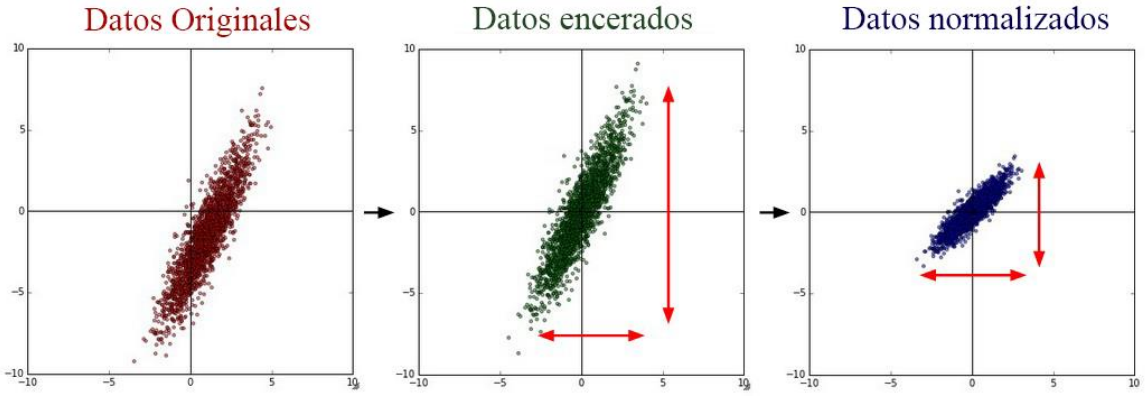
Normalización (X'')**.** De todo el conjunto de datos se divide cada dimensión por su respectiva desviación estándar. Matemáticamente se puede expresar de la siguiente forma:

$$X'' = \frac{X'}{\sqrt{\frac{\sum_{i=1}^{N}(x_i - x)^2}{N-1}}}.$$

En la Figura 12 se muestra un preprocesamiento de datos de un conjunto de datos. Nótese cada uno de los procedimientos comunes para poseer información uniforme. Cabe añadir que de forma común se preprocesan el conjunto de datos de las formas antes mencionadas por lo que este procedimiento varía por el tipo de datos que se maneja. En el caso de señales temporales se maneja usualmente una normalización de cero a uno y una eliminación del componente continuo (Grijalva et al., 2020).

Figura 12

Ejemplo del preprocesamiento de datos



Nota. Procesamiento de datos originales, encerados y normalizados. Adaptado de CS231n: Convolutional Neural Networks for Visual Recognition, por Li et al., 2021, Stanford Course (https://cs231n.github.io/neural-networks-2/). CC BY 2.0

.

Inicialización de parámetros

Una CNN posee un sinfín de parámetros o pesos en sus capas convolucionales y, mientras más profunda sea la red más parámetros tiene. En principio, los pesos deben ser inicializados de forma correcta al inicio del entrenamiento ya que de eso depende que tan rápido puede ser entrenado un modelo (Ghosh et al., 2020). Una de las formas más sencillas de inicializar los pesos es estableciéndolos con valor cero sin embargo se pueden entrar en ambigüedades de entrenamiento ya que todas las capas se inician con valores iguales por lo que no hay disparidad entre las neuronas, por consecuencia es posible que no se aprendan varias características de los conjuntos de datos. En general se inicializan los parámetros de forma aleatoria con matrices rellenas de valores aleatorios. Uno de los métodos más populares para inicializar de forma aleatoria es: inicialización aleatoria gaussiana, inicialización aleatoria uniforme, y finalmente la inicialización aleatoria ortogonal.

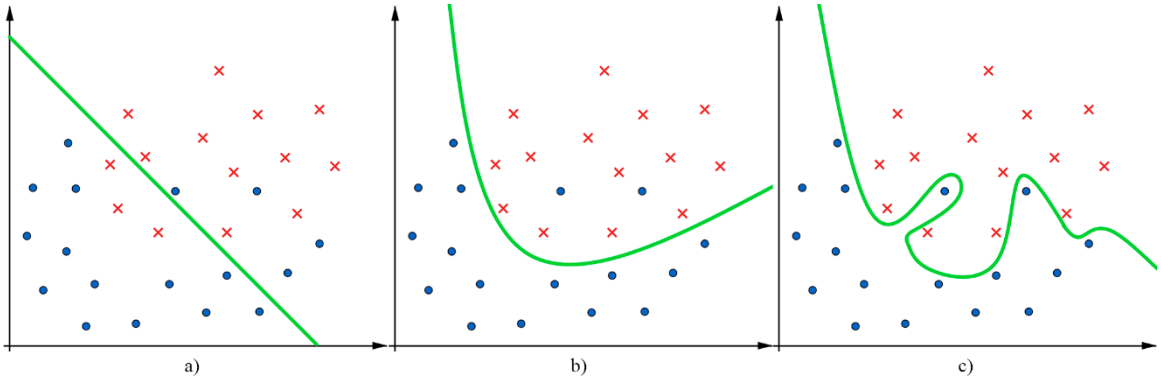
Regularización de una CNN

El principal reto de los algoritmos de aprendizaje radica en adaptarse adecuadamente a una entrada nueva o no vista previamente dentro del conjunto de los datos de entrenamiento, la capacidad de lograrlo se lo llama generalización. Uno de los problemas más comunes a la hora de hacer un modelo de red neuronal es el de evitar un sobreajuste. Esta anomalía se suscita si el modelo se desempeña de forma correcta en el entrenamiento, sin embargo, no se pueden predecir los datos con precisión. A esta convergencia de aprendizaje, si no se logra una buena generalización se genera un sobreajuste (Ghosh et al., 2020). En la Figura 13 se muestran distintos casos de aprendizaje en una clasificación binaria. En la primera imagen a) la clasificación es demasiado simple para poder establecer y denotar una varianza en el conjunto de datos, en la segunda imagen b) se observa un ajuste apropiado por el que sería normal que ciertos valores sean descartados dentro del modelo, y finalmente en la tercera

imagen c) nótese que existe un ajuste en el que se sobre aprende el modelo, es decir, que es demasiado bueno para ser verdad.

Figura 13

Ejemplos de aprendizaje en una clasificación binaria



Nota. a) Ajuste Insuficiente, b) Ajuste Apropiado y c) Sobreajuste. Adaptado de Ghosh et al., 2020.

Existen numerosas formas para evadir un sobreajuste en el entrenamiento de un modelo. Una de las ideas más innovadoras es la de *Batch Normalization* propuesta por Ioffe & Szegedy (2015). El desplazamiento interno de covariables puede explicarse como el cambio en la distribución de las activaciones en cada capa debido a la continua actualización de los pesos durante el entrenamiento. Este desplazamiento puede llegar a ser muy alto de forma que tarde más en converger, así aumenta el tiempo de entrenamiento y reduce la eficiencia en general. Esta problemática se soluciona con el método de *Batch Normalization***,** donde la normalización forme parte de la arquitectura del modelo como una capa dentro de una arquitectura CNN. Eso asegura que las activaciones de salida de una red sigan una distribución gaussiana unitaria y normaliza la salida en cada capa a través de la resta de la media y división de la desviación estándar.

Selección de un optimizador

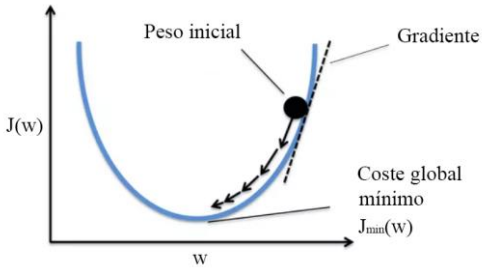
En la parte final del diseño de un modelo de aprendizaje de redes neuronales en general se encuentra la selección de un optimizador de aprendizaje, es decir, de un algoritmo de aprendizaje. El objetivo primordial del algoritmo es la de minimizar la diferencia que existe entre una predicción y la realidad. Esta diferencia se define como el error o función de pérdida basada en parámetros como pesos, entre otros. Los métodos basados en el gradiente son los más empleados dentro del campo de las CNN (X. S. Yang, 2019). Estos métodos calculan la gradiente, es decir, la derivada parcial de la función de pérdida en función a los pesos, donde se denota a través de la siguiente fórmula, donde W son los pesos y $J(W)$ la función de pérdida, en una k iteración:

$$W^{(k+1)} = W^{(k)} - \frac{\partial}{\partial W^{(k)}} J(W)\,.$$

A través de todo el cálculo de gradientes se busca el valor mínimo de $J_{min}(W)$ en cada iteración se logra un mejor resultado de entrenamiento y se generan los pesos de la red convolucional que corresponden a un aprendizaje óptimo y satisfactorio. De manera gráfica, el cálculo de la gradiente en cada iteración puede observarse en la Figura 14.

Figura 14

Descripción gráfica del cálculo de gradiente de una función de pérdida



Nota. La inicialización de los parámetros, como el peso, se da a través de distintas formas explicadas en la Sección 0. Adaptado de *Introduction to Algorithms for Data Mining and Machine Learning*. Elsevier Science & Technology por Yang, X. S. 2019.

Uno de los mejores algoritmos de aprendizaje es la Estimación Adaptativa de Momentos (ADAM, del inglés *Adaptive Moment Estimation*) (Kingma & Ba, 2014). El algoritmo ADAM es sencillo de implementar, eficiente desde el punto de vista computacional, ya que requiere poca memoria y es apropiado para situaciones con grandes conjuntos de datos y parámetros. ADAM combina dos enfoques de descenso de gradiente estocástico, Gradientes Adaptativos (AdaGrad, del inglés *Adaptive Gradients*) y Propagación de Raíz Cuadrada (RMSP, del inglés *Root Mean Square Propagation*). En lugar de utilizar todo el conjunto de datos para calcular el gradiente real, este algoritmo de optimización utiliza un subconjunto de datos seleccionados al azar para crear una aproximación estocástica (Goodfellow et al., 2016).

Su nombre deriva de la estimación adaptativa de momentos y la razón por la que se llama así es porque ADAM utiliza las estimaciones del primer y segundo momento del gradiente para adaptar la tasa de aprendizaje de cada peso de la red neuronal. El momento m_n de una variable aleatoria se define como el valor esperado de esa variable elevado a la potencia de n. Matemáticamente se expresa como:

$$m_n = E[X^n]\,,$$

donde; m es el momento y X la variable aleatoria. Cabe añadir que el gradiente de una función de pérdida puede ser considerada una variable aleatoria ya que siempre es evaluada a través de un conjunto de datos inicialmente aleatorios. El primer momento es la media y el segundo momento es la varianza no central (es decir, que no es restada la media durante el cálculo de la varianza). Para el cálculo de momentos, ADAM utiliza las medias móviles exponenciales, calculadas en el gradiente evaluado en un mini *batch* actual, y se expresan a través de las siguientes fórmulas:

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t\,,$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2\,,$$

donde m_t es la media móvil del gradiente y v_t el gradiente al cuadrado. El gradiente en el *mini batch* actual está denotado por g y los valores de beta β son parámetros introducidos propios del algoritmo, que oscilan entre 0.9 y 0.999. Se considera la corrección de sesgo para el primer $\widehat{m_t}$ y segundo $\widehat{v_t}$ estimador de momento (Kingma & Ba, 2014) se tienen las siguientes fórmulas:

$$\widehat{m_t} = \frac{m_t}{1-\beta_1^t},$$

$$\widehat{v_t} = \frac{v_t}{1-\beta_2^t}.$$

El escalamiento de las medias móviles puede ser empleada para cada parámetro de aprendizaje (pesos en CNN), denota la siguiente ecuación:

$$w_t = w_{t-1} - \alpha \frac{\widehat{m_t}}{\sqrt{\widehat{v_t} + \epsilon}},$$

donde: *w* son los pesos del modelo y α el tamaño del *stride*. También se necesita especificar otras variables para el algoritmo, como la tasa de aprendizaje en distintos programas de software para desarrollo de modelos de aprendizaje de redes neuronales.

Modelos Generativos

Los modelos generativos son métodos basados en redes neuronales profundas los cuales, sin supervisión, aprenden a aproximarse a una distribución de datos. En el modelado generativo, para un entrenamiento x se extraen muestras de una distribución de datos empírica denominada $p_{data(x)}$ como datos de entrenamiento, un modelo generativo profundo aprende explícitamente una distribución estimada $p_{model(x)}$, la cual se aproxime al $p_{data(x)}$ lo más posible (Goodfellow et al., 2014).

Muchas técnicas de modelado generativo se basan en la estimación de densidad; sin embargo, algunos modelos generativos omiten todo el problema de desarrollar una función de densidad manejable y, en cambio, aprenden solo un mecanismo de generación de muestras manejable. Estos se conocen como modelos

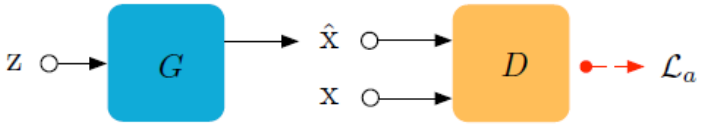
generativos implícitos. Esta categoría incluye la GAN. Se utilizaron GAN para construir un modelo generativo implícito profundo que podría producir muestras verdaderas a partir de la distribución del modelo en un solo paso de generación, con el fin de eliminar el requisito del proceso de generación incremental o la naturaleza aproximada del muestreo de cadenas de Markov (Goodfellow, 2017).

Redes Adversarias Generativas

Goodfellow et al. (2014) propuso un nuevo modelo GAN como proceso de confrontación para estimar modelos generativos. La Figura 15 muestra la estructura básica de GAN, que consta de dos redes: un generador (G) y un discriminador (D). El generador aprende a crear muestras realistas y estas muestras son utilizadas como ejemplos de entrenamiento negativos para el discriminador.

Figura 15

Esquema Original de una GAN



Nota. Adaptado de *An Introduction to Image Synthesis with Generative Adversarial Nets* (p. 17), por Huang et al., 2018.

El discriminador, por otro lado, aprende a discriminar entre muestras falsas y reales producidas por el generador y así penaliza al generador por crear muestras poco realistas. Durante la fase de entrenamiento, el generador comienza a crear muestras poco realistas de las cuales el discriminador aprende rápidamente a diferenciar.

Específicamente, G toma como entrada un vector de ruido z muestreado de una distribución p_z, y genera muestras o datos falsos $\hat{x} = G(z)$, los cuales pueden verse como una muestra de la distribución generativa aprendida p_G. D discrimina entre los

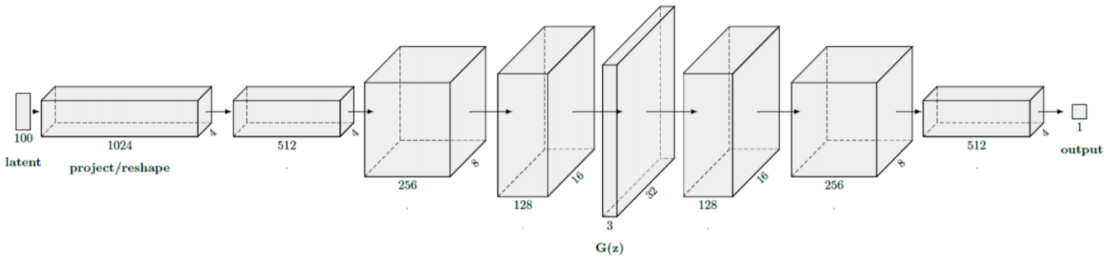
datos falso $\hat{x}$ y un valor x muestreados de la distribución de datos reales $p_{data}(x)$ (Huang et al., 2018).

En la arquitectura interna de una GAN de manera general, Ham et al. (2020) sugiere seguir la idea de las GAN Convolucionales Profundas basadas en una parte de la arquitectura de un Autocodificador Variacional (VAE, del inglés *Variational Autoencoder*), en el que se emplean capas convolucionales para el *downsampling* en el discriminador y para el *upsampling* en el generador se pueden emplear convoluciones que permitan aumentar la información por lo que es una buena opción el uso de las convoluciones transpuestas. La función de activación *Leaky ReLU* se utiliza en las capas ocultas del decodificador variacional y la red del generador, y la función de activación *ReLU* se utiliza en las capas ocultas del codificador variacional y el discriminador. Adicional se emplea la función de activación *Tanh* para la capa de salida del generador y la función de activación *Sigmoide* para la capa de salida del discriminador, por lo que es la responsable de calcular la probabilidad de que la entrada dada sea real o generada. También se emplea el *Batch Normalization* entre todas las capas, excepto la capa de salida del generador y la capa de entrada del codificador variacional y el discriminador.

En la Figura 16 se muestra esta arquitectura donde, además de los componentes mencionados previamente se tiene la capa de *Project & Reshape* en el que su función es la de implementar una capa totalmente conectada (Dense, del inglés *fully connected layer*) en el que se convierte el ruido (vector latente denotado por z) en un vector de mayor dimensión.

Figura 16

Arquitectura básica de una GAN



Nota. Adaptado de Unbalanced GANs: Pre-training the Generator of Generative Adversarial Network using Variational Autoencoder, por Ham et al., 2020.

Obsérvese también que es la arquitectura de una imagen en el que: las capas que existen entre el *Project & Reshape* y la capa de 3x32x32 son el *upsampling* de los datos; mientras que desde ese punto a la salida está compuesto por el *downsampling* de los datos, así arroja un único dato de probabilidad.

El método básico para entrenar el generador y el discriminador es crear un juego "mínimo-máximo" de dos jugadores, en el que el generador G intenta originar datos realistas para engañar al discriminador, mientras tanto el discriminador D intenta distinguir entre datos reales y sintéticos, de acuerdo con la siguiente función:

$$\min_{G} \max_{D} \mathcal{L}_a(G,D) = \mathbb{E}_{x\sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1-D(G(z)))]\,,$$

donde: $\mathcal{L}_a(G,D)$ se denomina la pérdida adversarial para G y D, $D(x)$ es la probabilidad de que x sea real. Al momento de realizar el entrenamiento, los parámetros de un modelo en la ecuación previa se actualizan mientras que los parámetros del otro son completamente fijos. Si se tiene un generador fijo, el discriminador óptimo es:

$$D^*(x) = \frac{p_{data}(x)}{p_{data}(x) + p_G(x)}.$$

Con lo que se puede definir que el generador es óptimo si $p_{data}(x) = p_G(x)$, $D(x)$ produce 0.5 para toda muestra extraída de x es decir, al generador se lo puede

considerar como óptimo si el discriminador se confunde al máximo y no puede distinguir las muestras falsas de las reales

Cabe añadir que, el momento en el que el discriminador se encuentra mejor entrenado que el generador se puede llegar a producir un rechazo de muestras por parte de G con una confianza cercana a uno, esto provoca que $1 - D(G(z))$ se sature y no puede proporcionar un gradiente suficiente para actualizar el generador (Goodfellow et al., 2014). Para un mejor entrenamiento del generador, es una práctica común maximizar $\log D(x)$.

Entrenamiento

El entrenamiento del generador y del discriminador está basado en el puntaje de clasificación entregado por la capa final del discriminador, indica que tan verídica es la información generada (falsa). En este sentido, y al considerar la revisión de las funciones de pérdida de redes neuronal previamente, se emplea una función del tipo BCE ya que se toma en cuenta dos casos, en el que el dato ingresado puede ser falso o verdadero donde existen dos probabilidades. Se considera como referencia la ecuación de la función BCE, se puede reescribir de la siguiente manera con los casos actuales:

$$L(y, \hat{y}) = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log \hat{y_i} + (1 - y_i)\log(1 - \hat{y_i})]\,,$$

donde: N es el número de *batch* donde se promedia en definitiva todo el *batch*, $\hat{y_i}$ es la predicción del modelo del discriminador $D(x)$ y y_i es la etiqueta verídica independiente mente que sea real o falsa, por lo que $y \in \{0,1\}$ se puede observar el diagrama de bloques del discriminador mostrado en la Figura 17. En este sentido, la predicción del discriminador puede denotarse de las siguientes dos formas:

$$\begin{cases} D(x) \approx 1 & X_{real} \\ D(x) \approx 0 & X_{falso} \end{cases}.$$

Si el valor es más cercano a 1 el discriminador predice que la entrada es una muestra de datos verdadera, mientras que una probabilidad cercana indicaría que la entrada es falsa. Por lo tanto, las muestras de entrenamiento X vendrían a ser X_{real} y X_{falso}, como se puede observar en la Figura 17. En el caso del discriminador, existen dos principales objetivos: el primero es el de maximizar la probabilidad de que los datos verídicos, denotados por $D(X_{real})$ sean cercanos a 1; mientras que el segundo es la minimización de la probabilidad de que los datos falsos, denotado por $D(X_{falso})$ sea cercano a 0, donde X_{falso} es la salida del generador denominado $G(z)$.

Figura 17

Diagrama del discriminador

Para el primer objetivo, se considera la ecuación de pérdida de BCE para el caso de una GAN, si se maximiza la probabilidad $D(X_{real})$: la etiqueta verdadera tiene que ser $y = 1$; y la salida predicha por el discriminador $\hat{y}$ debe ser real, tal que $\hat{y} = D(X_{real})$, al reescribir la ecuación BCE se tiene:

$$D_{pérdida_{real}} = -\log D(X_{real})\,.$$

Para el segundo objetivo, de igual forma se considera la ecuación de pérdida de BCE para el caso de una GAN, si se minimiza la probabilidad $D(X_{falsa})$: la etiqueta verdadera tiene que ser $y = 0$, y la salida predicha por el discriminador $\hat{y}$ debe ser falsa, tal que $\hat{y} = D(X_{falsa})$, donde $X_{falsa} = G(Z)$, donde $G(z)$ es la salida del generador en términos del vector ruido z. Se reescribe la ecuación del BCE se tiene:

$$D_{pérdida_{falsa}} = -\log\Big(1 - D\big(G(z)\big)\Big).$$

Por lo que finalmente la función de pérdida del discriminador es la suma de los dos objetivos,

$$D_{pérdida} = D_{pérdida_{real}} + D_{pérdida_{falsa}},$$

$$D_{pérdida} = -\log D(X_{real}) - \log\Big(1 - D\big(G(z)\big)\Big).$$

En el caso del generador existe un único objetivo, el de maximizar la probabilidad de que $D(G(z))$ sea cercano a uno, de forma que se encamine a la probabilidad de que los datos generados (falsos) se parezcan bastante a los datos del conjunto. Para este objetivo, la etiqueta verdadera tiene que ser $y = 1$, y la salida predicha por el discriminador $\hat{y}$ debe ser $\hat{y} = D(G(Z))$ que, como se mencionó, $G(z)$ es la salida del generador en términos del vector ruido z. Se reescribe la ecuación del BCE se tiene:

$$G_{loss} = -\log D\big(G(z)\big).$$

Con estas funciones de pérdida se busca, a través de distintos algoritmos, el aprendizaje y actualización de los parámetros (pesos).

Fallas de entrenamiento

Entrenar a una GAN en la práctica puede ser difícil y no tan confiable. Hay tres fallas más típicas que ocurren a lo largo del proceso de entrenamiento, según la evidencia empírica (Dumoulin et al., 2018).

En la no convergencia, el discriminador y el generador no pueden alcanzar el equilibrio. Se crean diferentes muestras como parte del proceso de formación, pero su calidad no aumenta significativamente.

En el colapso de modo el generador solo produce una mínima cantidad de muestras similares y en algunos casos suele producir muestras únicas, allí resulta denominado colapso completo.

En la rápida convergencia de la pérdida, el discriminador óptimo no es capaz de generar suficiente gradiente para mantener actualizado el generador. El entrenamiento para GAN inestable se puede mejorar al utilizar estructuras cuidadosamente planificadas o métodos de entrenamiento heurísticos (Creswell et al., 2017).

En el *One-sided label smoothing* se reemplaza los valores de salida 0 y 1 del discriminador, por valores suavizados como 0.9 y 0.1. Esto se puede considerar como un regularizador para disminuir la confianza del discriminador, lo que lo alienta a proporcionar gradientes fuertes para actualizar el generador. Agregar ruido a las muestras reales y falsas antes de introducirlas en el discriminador es un enfoque similar.

En la experiencia de repetición el discriminador se actualiza y utiliza las muestras generadas previamente guardadas en memoria en lugar de las muestras creadas recientemente. Esto mejora significativamente la convergencia del aprendizaje adversario.

En la discriminación por *mini-batch* se agrega una entrada adicional, la cual codifica la distancia entre una muestra determinada en un mini-batch. El discriminador puede detectar fácilmente si el generador proporciona las mismas salidas, es decir es capaz de distinguir las muestras generadas de las muestras reales, lo que está destinado a evitar y aliviar el problema del colapso de modo.

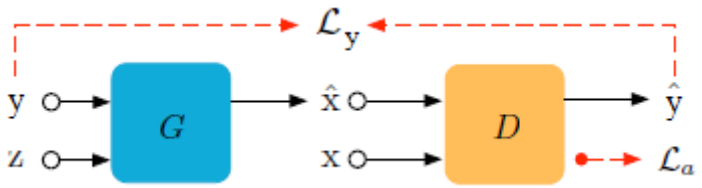
El promedio histórico es un método el cual permite regularizar los parámetros de la red, por medio de la obtención de un promedio el cual se lo obtiene mediante el proceso de entrenamiento y penaliza a todos los parámetros de la red al momento de que se desvíen del promedio obtenido. Esta técnica ayuda a resolver el problema de la no convergencia.

Redes Adversarias Generativas Condicionales

Al utilizar una GAN original, no se tienen control completo sobre las muestras generadas, debido a que la salida depende solo del ruido aleatorio, para mejorar los procesos de la GAN, Mirza & Osindero (2014), introdujeron a las Redes Adversarias Generativas Condicionales CGAN, definidas como la extensión de una GAN tradicional, donde se agrega una entrada condicional y a un ruido aleatorio z para que la muestra generada esté definida por $G(y, c)$. La información y puede ser una etiqueta de clases categóricas, imágenes y textos; con esto la CGAN intenta aprender un modelo generativo condicional implícito. En la Figura 18 se muestra el diagrama de bloques de un modelo CGAN.

Figura 18

Esquema general de una CGAN



Nota. Adaptado de *An Introduction to Image Synthesis with Generative Adversarial Nets* (p. 22), por (Huang et al., 2018).

Formalmente una CGAN se diferencia de una GAN al incluir la variable condicional y de la siguiente manera:

$$\min_G \max_D \mathcal{L}_a(G, D) = \mathbb{E}_{x\sim p_{data}(x)}[\log D(x|y)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z|y)))] .$$

Por lo tanto, $G: z \times y \rightarrow \hat{\boldsymbol{x}}$ y $D: x \times y \rightarrow [0,1]$, el discriminador se entrena para ser capaz de maximizar el etiquetado correcto, mientras que el generador se entrena para minimizar $\log(1 - D(G(z|y)))$.

En comparación con las GAN originales, las GAN condicionales controlan directamente sobre las muestras generadas en función de las restricciones

condicionales aplicadas. Las CGAN son mayormente utilizadas para aplicaciones en la que los datos siguen una secuencia para la generación de muestras que pueden ser complicadas. En definitiva, las CGAN no son estrictamente algoritmos de aprendizaje no supervisado ya que se requiere de datos etiquetados como entrada como una capa adicional, por lo que es importante considerar un correcto preprocesamiento de datos inicialmente etiquetados por lo que este modelo sería del tipo híbrido semi supervisado. La arquitectura puede ser implementada de manera similar a la GAN (Figura 16) con la diferencia de agregar a la entrada tanto del generador como del discriminador la etiqueta correspondiente.

Capítulo III

Materiales y Metodología

En el presente capítulo se describen los siguientes apartados: Primero, los materiales y software utilizados para la realización y entrenamiento del modelo de CGAN; Segundo, describir el tipo de metodología la cual se llevó a cabo para el desarrollo del modelo CGAN, se presenta la estructura utilizada en los modelos de red neuronal (generador y discriminador) así como su proceso de entrenamiento. Se detalla el preprocesamiento, el modelo CGAN y finalmente el posprocesamiento.

Materiales

¨Para el presente proyecto se emplea el software MATLAB® en su versión R2020a y se hace énfasis en la utilización de los paquetes *Deep Learning Toolbox* y *Signal Processing Toolbox*. El entrenamiento de algoritmos de DL requiere de cálculos y procesamientos intensivos por lo cual, para lograr un entrenamiento eficiente en tiempo, fue necesario el uso de computadoras con tarjeta gráfica dedicada que aceleren los procesos de entrenamiento. MATLAB® acelera este procedimiento con un soporte para las tarjetas gráficas NVIDIA® que posean CUDA®. En este sentido, las características de las computadoras utilizadas para este proyecto son las mostradas en la Tabla 2.

Tabla 2

Características de las computadoras utilizadas en el proyecto

MSI® GL63	DELL® G3 3579
	
- Intel® Core® I7 8th Gen (6 core, 12 threads, 2.2GHz - Turbo 4GHz) - **Nvidia® GeForce® GTX 1060 (6GB VRAM)** - 16 GB RAM - 1TB SSD	- Intel® Core® I7 8th Gen (6 core, 12 threads, 2.2GHz - Turbo 4GHz) - **Nvidia® GeForce® GTX 1050 Ti (4GB VRAM)** - 16 GB RAM - 1TB HDD

Nota. En la tabla se puede observar las características de dos computadoras, utilizadas para el proyecto de investigación.

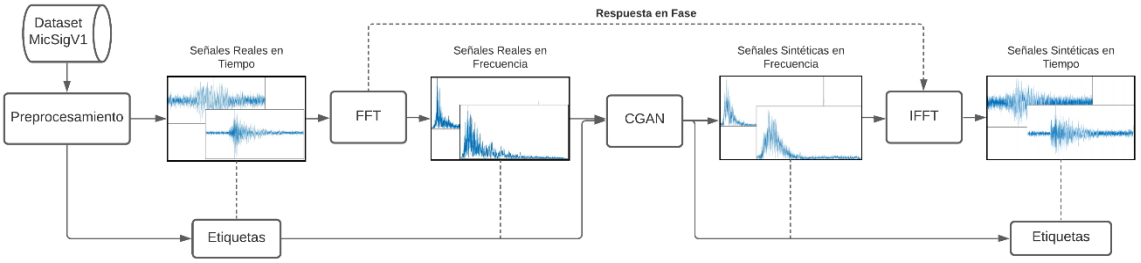
Metodología

La metodología del proyecto es del tipo experimental y sigue el diagrama de bloques expuesto en la Figura 19. En principio es fundamental disponer con una base de datos que alimente el modelo de red neuronal, en el caso de estudio particular se emplean las señales de los eventos sismo-volcánicos disponibles en la base de datos MicSigV1. El conjunto de datos necesita de preprocesamiento y depende del tipo de dato o estructura que requiera el modelo de red neuronal. En este caso es fundamental el uso del dominio de la frecuencia de los eventos sismo-volcánicos a través de la Transformada Rápida de Fourier (FFT, del inglés *Fast Fourier Transform*), ya que las señales sísmicas brindadas por el IGEPN se encuentran en el dominio temporal y procesar en el dominio del tiempo es particularmente difícil debido a la composición extremadamente compleja de la forma de onda sísmica (Grijalva et al., 2020). Estos valores más las etiquetas, alimentan al modelo CGAN que se compone de dos submodelos internos: el generador y el discriminador. El modelo generador es responsable de generar nuevos datos plausibles que, idealmente, no se pueden

distinguir de los datos reales. El modelo discriminador es responsable de clasificar los datos y verificarla como real o falsa. El discriminador penaliza al generador si produce resultados inverosímiles. El modelo CGAN a su salida genera señales sintéticas en el dominio de la frecuencia y, para que éstas sean aprovechables es fundamental volverlas a transformar al dominio temporal a través de la Transformada Inversa Rápida de Fourier (IFFT, del inglés *Inverse Fast Fourier Transform*), se realiza un posprocesamiento final y como salida de todo el proceso se tienen las señales sintéticas etiquetadas.

Figura 19

Diagrama de bloques del proyecto



Base de Datos

Para el desarrollo del presenta proyecto, se emplea un conjunto de datos de señales de eventos sismo-volcánicos discretos del Volcán Cotopaxi, obtenido del repositorio ESeismic en cortesía el IGEPN, denominado MicSigV1 (Pérez et al., 2020).

El conjunto de datos MicSigV1 fue obtenido de tres estaciones BREF, VC1 y BVC2, la base de datos contiene un número total de 1187 registros sísmicos discretos detectados por el algoritmo STA/LTA y etiquetados manera visual por los especialistas del IGEPN. De esos 1187 registros sísmicos, se dividen en los eventos descritos a continuación:

- 1044 LP
- 101 VT

- 27 REG
- 8 HB
- 7 ICEQUAKES

Las señales de la base de datos MicSigV1 se encuentran muestreadas a una frecuencia de 50 Hz y 100 Hz. Cabe añadir que estos eventos se encuentran almacenados en el formato $.mat$ de MATLAB. En el proceso de generación de la base de datos, todas las señales fueron sometidas a una serie de pasos de procesamiento que incluían la eliminación de efectos instrumentales y la aplicación de un filtro no causal FIR de paso alto y una frecuencia de corte de 1 Hz con el objetivo de preservar el contenido espectral para su posterior análisis (Pérez et al., 2020).

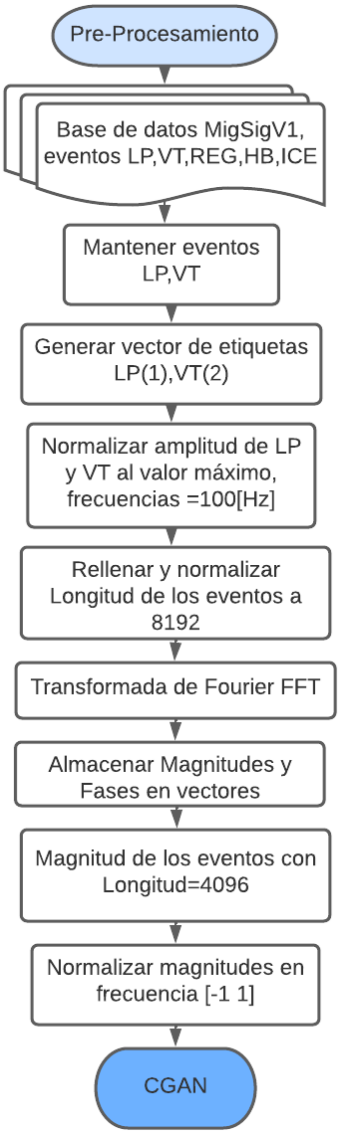
Preprocesamiento

La base de datos de forma general consta de cinco tipos de eventos sismo-volcánicos con distintas características por cada evento como son, entre los más significativos: la frecuencia de muestreo, el año en el que fue identificado, el tipo de evento, la duración, un punto de inicio, un punto final del evento, y el evento guardado en un formato del tipo *double*. De los cinco tipos de eventos sismo-volcánicos que se encuentran en la base de datos, se utilizaron únicamente los eventos LP y VT puesto que su nivel de ocurrencia es alto en comparación con el resto de los eventos. Para el entrenamiento del modelo se cuenta con un total de 1044 eventos LP y 101 VT muestreadas a distintas frecuencias, entre 50 Hz y 100 Hz.

El preprocesamiento se puede observar en la Figura 20, del conjunto de datos seleccionados se proceden a seleccionar los valores que alimentan al modelo los cuales son: el tipo de evento (etiquetas del modelo) y el evento en sí (la señal). Como primer punto se separan en un vector las etiquetas en un formato apropiado de lectura para el modelo de aprendizaje que en este caso es en números por lo que se mapea a los eventos LP con el valor 1 y a los eventos VT con el valor 2.

Figura 20

Diagrama de bloques del preprocesamiento



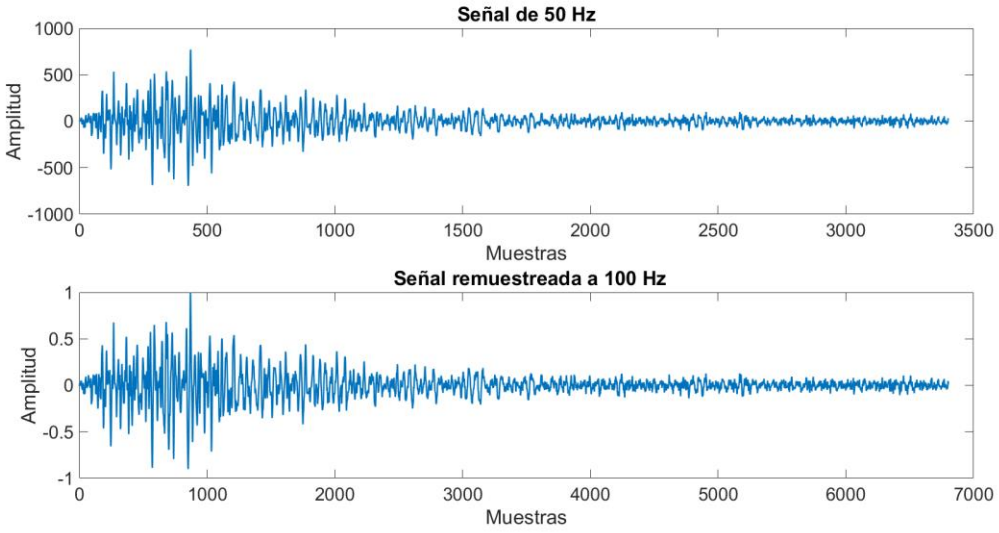
Como segundo punto, las señales se remuestrean a una frecuencia de 100 Hz para todo el conjunto de datos a través del uso de un filtro de anti-solapamiento polifásico disponible a través del software en el que se desarrolla el proyecto (MATLAB, 2020). Adicionalmente se realiza una normalización del tipo máximo a estos vectores iniciales a través del uso de la siguiente ecuación:

$$X' = \frac{X}{|X_{max}|},$$

donde X' es el vector normalizado, X el vector por normalizar y X_{max} el valor máximo del vector. Normalizar los vectores en esta instancia ayuda a tener las amplitudes balanceadas antes de los siguientes pasos. Obsérvese en la Figura 21 un evento sísmico original del tipo LP en el que se realiza el remuestreo de frecuencia desde los 50 Hz hasta los 100 Hz (nótese en la cantidad de muestras), así como la normalización de amplitud a su punto máximo.

Figura 21

Remuestreo a 100 Hz y normalización de la amplitud



Posteriormente las señales deben pasar por una normalización en la cantidad de muestras que cada uno tiene, debido a que el modelo de red neuronal se alimenta de valores con el mismo tamaño de muestras. En promedio, los eventos tienen una duración de 35 segundos en el que la desviación estándar es de cerca de 15 segundos. Si se suman ambas cantidades se tiene un valor de 50 segundos. Al tener una frecuencia de muestreo de 100 Hz se tiene un valor referencial de 5000 muestras. De preferencia el modelo de red neuronal debe emplear en sus entradas valores equivalentes a una potencia de dos (esto debido a la manipulación que se hace de la

información entre capas) por lo que el valor más cercano al valor referencial obtenido es de 8192.

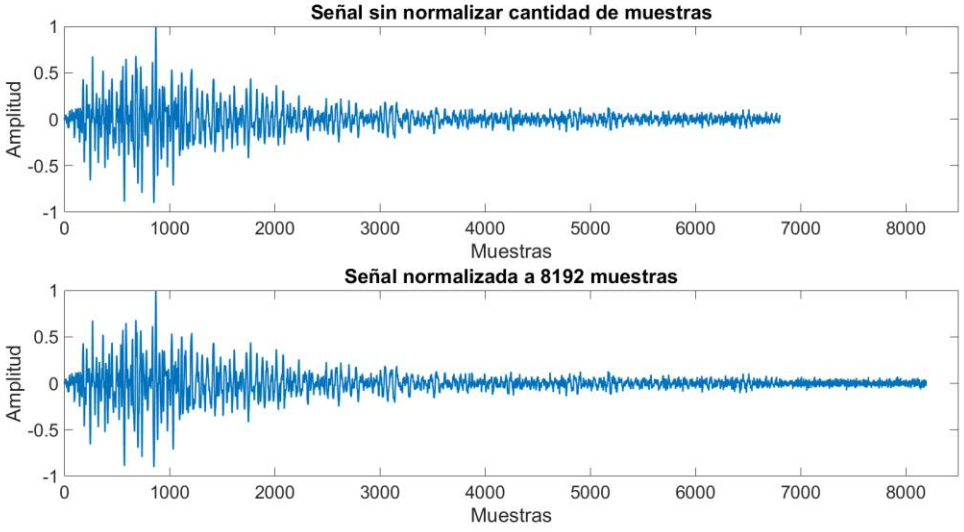
El valor de muestras necesarias obtenido es ideal puesto que, además de cumplir con las estadísticas propias de las señales, posee una resolución alta que le permite al modelo aprender más características de los eventos. Para equiparar todos los eventos a la misma cantidad de muestras se emplea un rellenado de información a las señales con ruido aditivo gaussiano blanco (AWGN, del inglés *Aditive White Gaussian Noise*) con una potencia de aproximadamente -3dBm (potencia promedio de ruido en las señales normalizadas que se generaron). Matemáticamente se expresa de la siguiente forma:

$$X' = X + n_{awgn}$$

donde, X' es el vector añadido ruido, X el vector sin ruido y n_{wgn} el vector de ruido gaussiano blanco. El proceso se observa en la Figura 22.

Figura 22

Normalización en la cantidad de muestras



Con el preprocesamiento de las señales temporales listo, se procede a sacar las magnitudes de las señales a través de la FFT. La transformada se la hace de punto a

punto hasta la longitud de la señal $N = 8192$. Al generarse el vector de frecuencia, por la simetría hermitiana generada a través de la FFT en señales reales, se tiene un vector donde el resultado de la frecuencia ocurre dos veces. Por tal motivo el vector de frecuencia útil es de tamaño $N/2 = 4096$ debido a que los valores restantes son el reflejo de esta. Adicional a esto, se guardan las fases reales (obtenidas a través de la FFT) para posteriormente utilizarlas en la reconstrucción de las señales sintéticas.

Los vectores de respuesta en frecuencia originales pasan por una normalización de amplitud del tipo *Min-Max* (*del inglés Minimum and Max Normalization*) que es dividir cada valor de la señal por el valor absoluto de su punto máximo, descrita matemáticamente en la siguiente ecuación:

$$X' = a + \frac{(X - X_{min})(b - a)}{X_{max} - X_{min}},$$

donde X' es el vector normalizado, a es el punto mínimo deseado, b el punto máximo deseado, X_{min} el valor mínimo del vector y X_{max} el valor máximo del vector. La normalización debe darse entre los valores -1 y 1, por lo que:

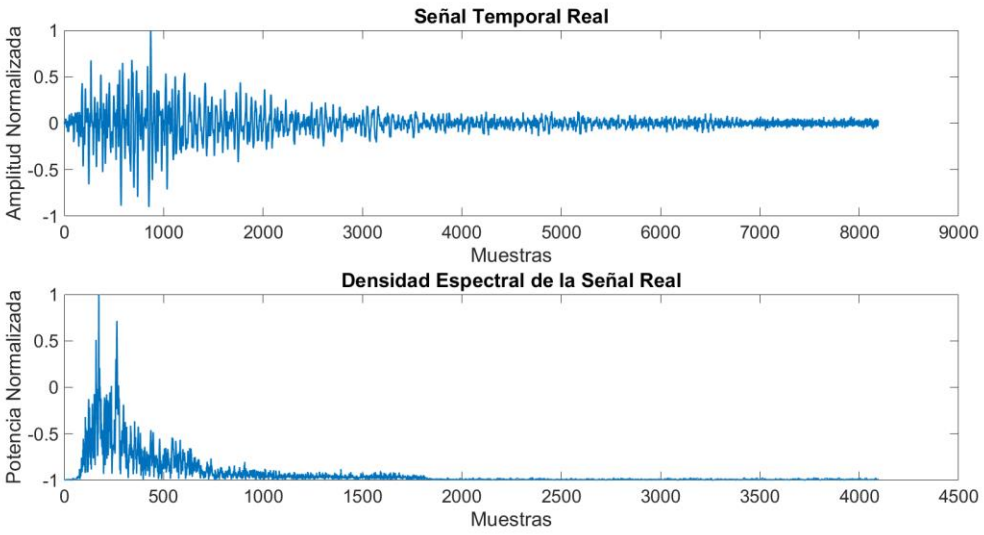
$$a = -1\,,$$

$$b = 1\,.$$

La justificación en el uso de estos valores de a y b radica en la aplicación de un activador del tipo tangente hiperbólico dentro del modelo del generador ya que el dominio de esta función está entre -1 a 1. Un ejemplo de este procedimiento puede observarse en la Figura 23, en el que se muestra la FFT de una señal temporal real de un evento LP. Con este proceso final, se tiene como entrada a la red neuronal: la matriz de las magnitudes en frecuencia y las etiquetas originales. Las fases originales se emplean en la reconstrucción de las señales en la fase de posprocesamiento final.

Figura 23

FFT de la Señal Temporal Real

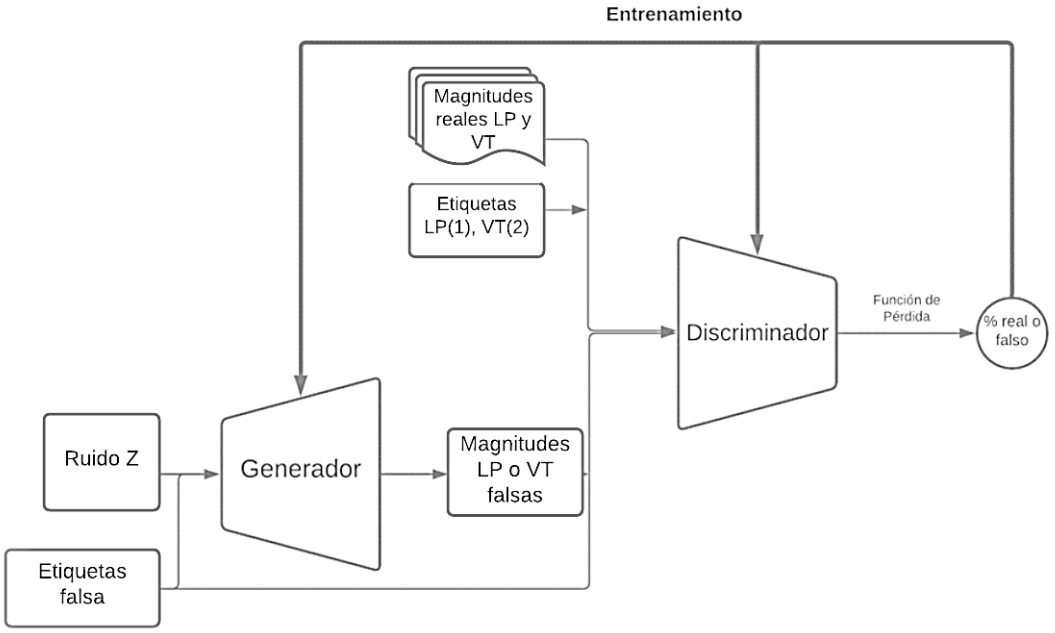


CGAN

El vector de magnitudes en frecuencia y las etiquetas de los eventos LP y VT ingresan como entradas al sistema. Como se puede observar en la Figura 24, inicialmente el primer modelo es el Generativo y este no depende directamente de los datos ingresados ya que en el proceso de aprendizaje se retroalimenta de la salida del modelo.

Figura 24

Diagrama de bloques general de la CGAN



En primera instancia, en el modelo del generador ingresaron dos variables aleatorias las cuales son un vector ruido *Z* y un vector de etiquetas generadas falsas como se puede observar en la Figura 24. A la salida del generador se obtiene una respuesta en frecuencia generada la cual puede ser LP o VT el cual depende de la etiqueta falsa ubicada por el modelo, esta etiqueta durante el entrenamiento se vuelva más precisa gracias a la puntuación y retroalimentacion recibida por el modelo del discriminador.

Las variables falsas (respuestas en frecuencia y etiquetas) ingresan al modelo Discriminador en donde tambien ingresan el conjunto de magnitudes en frecuencia y etiquetas reales previamente obtenidas en el preprocesamiento.

El modelo del discriminador es capaz de procesar y generar una función de pérdida, la cual indica la veracidad del resultado del modelo del generador, como se presenta en la ecuación siguiente:

$$D_{pérdida} = -\log D(X_{real}) - \log\Big(1 - D\big(G(z)\big)\Big)\,,$$

Donde: $D(X_{real})$ de denomina datos veridicos y $D\big(G(z)\big)$ los datos falsos.

La retroalimentación de aprendizaje hacia el generador y discriminador se la hace a través del cálculo de gradientes de las funciones de pérdida. Estos gradientes se emplean en la actualización de los pesos entrenables del *kernel* por medio de la función de pérdida del generador

$$G_{loss} = -\log D\big(G(z)\big)\,.$$

Generador. En la Tabla 3, se lista la arquitectura completa del generador del modelo CGAN. Se denomina d al tamaño del filtro el cual es igual a 64.

Tabla 3

Arquitectura del Generador

Operacion	*Kernel Size*	*Salida*
Entrada z	100	
projectandReshape	100	(4x1x1024)
Etiquetas	(1x1x1)	
Concat (proj + etiquetas)	(4x1x1024) + (1x1x1)	(4x1x1025)
Trans Conv1D (Stride=1)	(8*d,kernel_size=13)	(16x1x512)
batchNormalization		
ReLU		
Trans Conv1D (Stride=4)	(4*d,kernel_size=6)	(64x1x256)
batchNormalization		
ReLU		
Trans Conv1D (Stride=4)	(2*d,kernel_size=6)	(256x1x128)
batchNormalization		
ReLU		
Trans Conv1D (Stride=4)	(1*d,kernel_size=6)	$(1024x1x64)$
batchNormalization		
ReLU		
Trans Conv1D (Stride=4)	(1, kernel_size=6)	(4096x1x1)
batchNormalization		
Tanh		

Nota. En la tabla se presenta todas las capas utilizadas para el entrenamiento del modelo Generador.

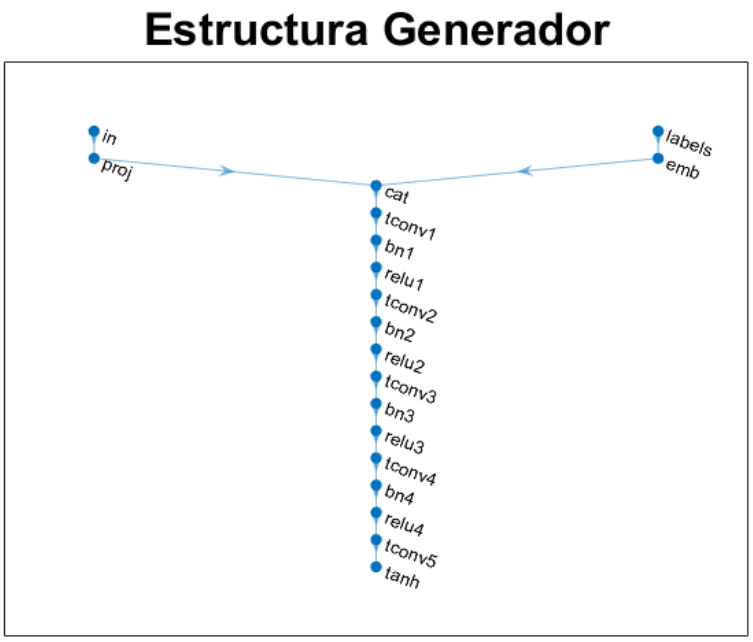
El generador es un modelo totalmente convolucional transpuesto, en el cual ingresa un vector de ruido y de etiquetas para generar una magnitud de frecuencia sintética de tamaño $(4096 \times 1 \times 1)$. Como se puede observar en la Figura 25 al generador ingresa un vector de ruido de números aleatorios normalmente distribuidos de longitud 100 en conjunto con el vector de etiquetas. Los dos elementos, antes de ingresar al generador, se convierten en una entrada del tipo imagen con la función *imageInputLayer*.

Al vector ruido se le realiza una proyección y remodelación con la función *projectAndReshape* la cual utiliza una capa completamente enlazada para aumentar la escala de la entrada y remodelar la salida al tamaño requerido, esta función forma parte del *Deep Learning Toolbox* de MATLAB.

Al generador ingresa una sola capa, por lo que el ruido remodelado y las etiquetas se concatenan en una sola, esto se puede observar en la Figura 25.

Figura 25

Estructura del Generador



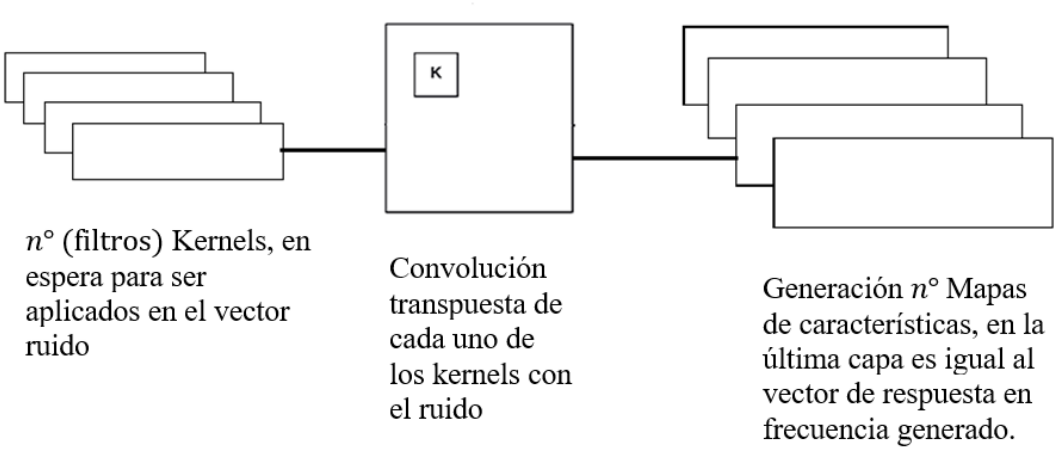
El modelo del generador propuesto se divide en cinco bloques, cada uno de ellos consta de: capa de convolución transpuesta, capa de *Batch Normalization,* una función de activación ReLU y en la capa de salida se utiliza una función de activación Tanh. Estos se explican a detalle a en la Figura 25.

La capa *Batch Normalization* se ubica en la mitad de cada bloque y estandariza todas las activaciones realizadas en el bloque anterior, con el fin de tener una media de cero y una varianza unitaria (Ioffe & Szegedy, 2015).

Con la capa *Batch Normalization* se logró acelerar el proceso de entrenamiento. Para el modelo de generador diseñado en la capa de salida no se utiliza esta capa *Batch Normalization* donde se consideró el modelo presentado en Radford et al. (2016).

Figura 26

Filtros y Kernel en el Generador.



El tamaño del filtro en la capa más profunda es mayor con un total de 512, el filtro es grande debido a que tiene que identificar una mayor cantidad de características en cada iteración realizada como se observa en Figura 26, mientras recorre todas las capas el tamaño del filtro disminuye en potencia de dos hasta llegar a uno donde ya se encuentra la respuesta en frecuencia generada, recomendación de Donahue et al. (2018).

Los anchos iniciales se plantearon con un máximo de la longitud de la magnitud de la señal, para los cinco bloques del modelo se estableció los saltos entre cada bloque a una potencia de dos, en el modelo del generador comenzó, desde el ancho 4 hasta llegar al ancho 4096 que es la salida igual al tamaño de longitud de la respuesta frecuencial.

Para la obtención de los tamaños de *kernel* utilizados en cada capa convolucional transpuesta del generador se emplea la fórmula detallada a continuación

$$k = W_o - s(W_i - 1) + 2p,$$

donde: W_o es el ancho final, W_i es el ancho inicial, s el stride y p es el padding. Para la segunda capa se tiene el siguiente tamaño de *kernel*. Donde se utilizan los valores,

$$k = 64 - 4(16 - 1) + 2\,,$$

$$k = 6\,.$$

Con la fórmula presentada se obtienen todos los valores del *kernel* que se observan en la Tabla 4, para cada bloque del modelo generador presentado en la Tabla 3.

Tabla 4

Valores kernel, capas modelo generador

Bloque	W_i	W_o	s	Tamaño del *Kernel*
1	4	16	1	13
2	16	64	4	6
3	64	256	4	6
4	256	1024	4	6
5	1024	4096	4	6

Nota. En la tabla se presenta todos los valores establecidos para obtener el tamaño del *kernel* en el modelo del generador.

La función de activación ReLU permite superar el problema del gradiente de fuga y favorece las activaciones dispersas (Radford et al., 2016). La función de activación Tanh es utilizada en la última capa del generador y garantiza que los valores generados estén mapeados entre $[-1\ 1]$, igual a la normalización realizada en el preprocesamiento a la respuesta en frecuencia, además se considera esta opción como se puede observar en el modelo generado en C.

Discriminador. En la Tabla 5 se lista la arquitectura completa del discriminador del modelo CGAN. Se denomina d al tamaño del filtro igual a 64. El discriminador es un modelo totalmente convolucional, en el cual ingresa el vector de magnitudes frecuenciales de los eventos LP, VT y el vector de etiquetas antes almacenadas en el preprocesamiento donde se distingue entre los eventos LP con un valor de 1 y los VT con el valor de 2, en la salida del discriminador se obtiene un vector etiqueta (1x1x1), el cual es una puntuación de probabilidad entre 0 y 1.

Tabla 5

Arquitectura del Discriminador

Operacion	*Kernel Size*	*Salida*
Entrada x G(z)	(4096x1x1)	
Etiquetas	(1x1x1)	(4096x1x2)
Conv1D (Stride=4) *LReLU (α=0.2)*	(d,kernel_size=6)	(1024x1x64)
Conv1D (Stride=4) *LReLU (α=0.2)*	(2*d,kernel_size=6)	(256x1x128)
Conv1D (Stride=4) *LReLU (α=0.2)*	(4*d,kernel_size=6)	(64x1x256)
Conv1D (Stride=4) *LReLU (α=0.2)*	(8*d,kernel_size=6)	(16x1x512)
Conv1D (Stride=4) *LReLU (α=0.2)*	(16*d,kernel_size=6)	(4x1x1024)
Conv1D (Stride=1)	(1,kernel_size=4)	(1x1x1)

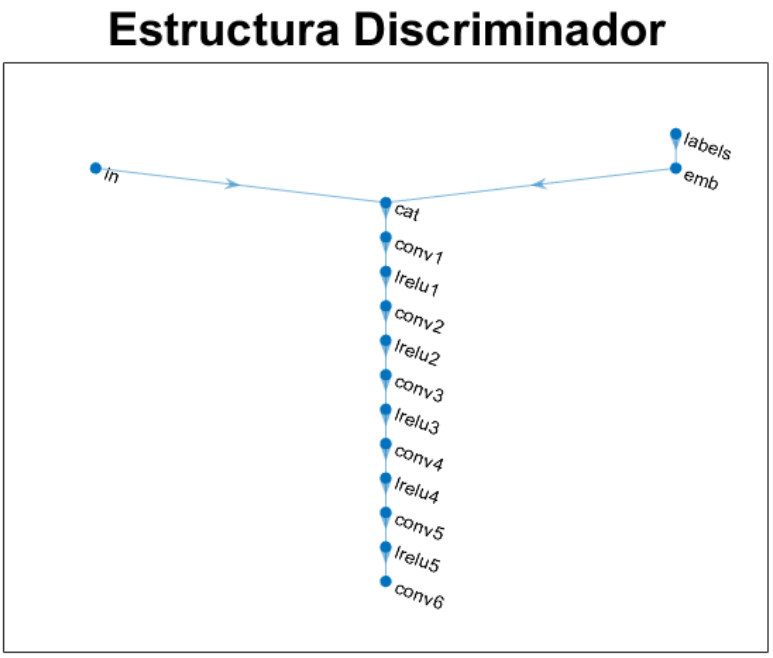
Nota. En la tabla se presenta todas las capas utilizadas para el entrenamiento del modelo Discriminador.

Como se puede observar en la Figura 27 al discriminador ingresa el vector de magnitudes frecuenciales de los eventos LP y VT al cual se le convierte en tipo imagen con la función *imageInputLayer* y el vector de etiquetas. Al generador ingresa una sola capa, por lo que el vector imagen de las magnitudes frecuenciales reales y las etiquetas se concatenan en una sola, esto se puede observar en la Figura 27.

El modelo del discriminador se divide en seis bloques, cada uno de ellos consta de: capa de convolución y la función de activación Leaky ReLU los cuales se explican a detalle a continuación. Los pesos del modelo discriminador se encuentran inicializados con valores aleatorios gaussianos muy pequeños y la pendiente de Leaky ReLU se inicializó con un valor de 0,2, este permite tener valores de gradientes distintos de cero al momento del entrenamiento y el paso de algunos valores negativos lo cual ayudó a evitar que el modelo CGAN se atasque (Radford et al., 2016). Los filtros para el discriminador inician con un valor de 64 en cada bloque, para después duplicarlos hasta llegar el quinto bloque y finalmente se queda con solo 1 filtro en la salida del modelo.

Figura 27

Estructura del Discriminador



Se inicia la primera capa con un vector con longitud igual a la longitud de las magnitudes en frecuencia 4096, se establecieron los saltos entre cada bloque de capa convolucional en potencia de dos, en el modelo del discriminador inicia desde el ancho 4096 hasta llegar al ancho 1, el cual da la puntuación de probabilidad del modelo.

Para la obtención de los tamaños de *kernel* utilizados en cada capa convolucional del discriminador se emplea la fórmula detallada a continuación:

$$k = W_i - s(W_o - 1) + 2p$$

Para la segunda capa se tiene el siguiente tamaño de *kernel*. Donde se utilizan los valores.

$$k = 1024 - 4(256 - 1) + 2$$

$$k = 6$$

Se utilizó la formula presentada y se obtienen todos los valores del *kernel* los cuales se observan en la Tabla 6 para cada bloque del modelo discriminador presentado en la Tabla 5

Tabla 6

Valores kernel, capas modelo generador

Bloque	W_i	W_o	s	Tamaño del Kernel
1	4096	1024	4	6
2	1024	256	4	6
3	256	64	4	6
4	64	16	4	6
5	16	4	4	6
6	4	1	1	4

Nota. En la tabla se presenta todos los valores establecidos para obtener el tamaño del *kernel* en el modelo del discriminador.

Etapa de Entrenamiento

El modelo CGAN se entrena a través de bucles anidados donde depende del número de *epoch* (del inglés, *epoch)* y el tamaño del lote (del inglés, *minibatchsize)*. *Epoch* es el ciclo completo de entrenamiento, el cual está compuesto de una cantidad de lotes e iteraciones las cuales se le calculan en relación con el número de datos existentes.

Se planifica que se va a probar con diferentes *epoch*, para identificar el número exacto el cual ayuda a el modelo CGAN a capturar características de las respuestas en frecuencias naturales ingresadas para el entrenamiento, además se sigue lo mencionado en Karras et al. (2020), si se encuentra limitado el número de datos con los cuales se van a entrenar a una red neuronal adversaria, al ubicar gran cantidad de *epoch* podría llegar a un sobre entrenamiento del modelo*,* por lo que no se obtienen buenos resultados a la salida del modelo del generador.

El entrenamiento de la CGAN se define en primera estancia la aleatorización el vector de respuestas en frecuencia con su respectivo vector de etiquetas, esta acción se

repite cada vez que el valor de la *epoch* aumenta en el entrenamiento. Para empezar en el proceso de entrenamiento se calcula los gradientes que alimenta al optimizador:

En primer lugar, se calculan las predicciones de los datos reales a través del entrenamiento del modelo del discriminador, se logra a partir de los dos vectores de entrada: vectores de eventos originales y sus etiquetas respectivas.

En segundo lugar, se calculan las predicciones para valores generados de la siguiente forma: se entrena el modelo del generador y se obtiene un vector sintético, posterior a ese paso se entrena al modelo del discriminador con el vector sintético obtenido, así como también las etiquetas empleadas.

Como tercer paso, se calculan las probabilidades de las predicciones realizadas en el primer y segundo paso a través del uso de la función de activación sigmoide (Radford et al., 2016).

Si el discriminador aprende a diferenciar entre las respuestas en frecuencia reales y las generadas demasiado rápido, el generador puede fallar en el entrenamiento. Para equilibrar mejor el aprendizaje del discriminador y del generador, se invierten aleatoriamente las etiquetas de una parte de las magnitudes en frecuencia reales.

El objetivo del generador es crear datos los cuales el discriminador clasifique como "reales" mientras que el objetivo del discriminador es no ser "engañado" por el generador. Este caso se cumple si se emplean las funciones de pérdida asociadas al BCE la cual se la define para modelar tanto el generador y el discriminador.

Pérdidas del discriminador. Se tiene contemplado la eficiencia del discriminador para distinguir las respuestas en frecuencia reales de las sintéticas que salen del generador. Para las predicciones reales se tienen una etiqueta como 1 y para las falsas una etiqueta de 0. Esta respuesta obtenida en la función de perdida se retroalimenta al generador.

Perdidas del generador. Se tiene contemplado el análisis del engaño que puede tener el modelo generador hacia el discriminador, igualmente tiene la retroalimentación de las puntuaciones del discriminador para actualizar los pesos del vector ruido, al final del entrenamiento se obtiene clasificación de 1 si la señal llegó a engañar al discriminador.

Finalmente, con las pérdidas se calcula por cada modelo los gradientes. Los parámetros tanto del discriminador como del generador se actualizan por cada iteración con los gradientes previamente calculados a través del optimizador de Adam, el cual se utilizó los parámetros que recomienda el creador (Kingma & Ba, 2014): la tasa de aprendizaje y los coeficientes beta uno y dos, que corresponden a los siguientes valores:

$$\alpha = 0.001$$

$$B_1 = 0.5$$

$$B_2 = 0.999$$

Posteriormente la herramienta Toolbox Deep Learning de Matlab, permite observar un puntaje el cual se monitoriza entre 0 a 1 donde se verifica si el generador y el discriminador, durante el tiempo de entrenamiento de la CGAN cumple con los objetivos establecidos.

La puntuación del generador viene dada por la media de la probabilidad de las respuestas en frecuencia generadas en el modelo, las puntuaciones del generador comienzan desde el valor de 1 y durante el entrenamiento este converge a 0.5 hasta que el generador mejore.

La puntuación del discriminador viene dada por la media de la probabilidad de la salida del discriminador, está compuesta de dos partes la primera es la probabilidad de salida del discriminador para la respuesta en frecuencia reales y la segunda es la probabilidad de respuesta en frecuencia sintéticas, las puntuaciones del discriminador

empiezan cerca de 0 y convergen a 0.5 por lo que dependen de la mejora del generador para engañar al discriminador.

Posprocesamiento final

Posterior al entrenamiento de la CGAN, el modelo genera dos vectores: las magnitudes frecuenciales sintéticas y sus etiquetas respectivas por tipo de evento. En la Figura 28 se puede observar el diagrama de bloques de todo este proceso final.

Los vectores de respuesta en frecuencia sintéticos deben pasar por una normalización min-max de 0 a 1 debido a que en un inicio la separación del tiempo en la magnitud frecuencial se la hizo en ese dominio de la amplitud. Por tal motivo se emplea la ecuación general min-max, con los valores de a y b como 0 y 1.

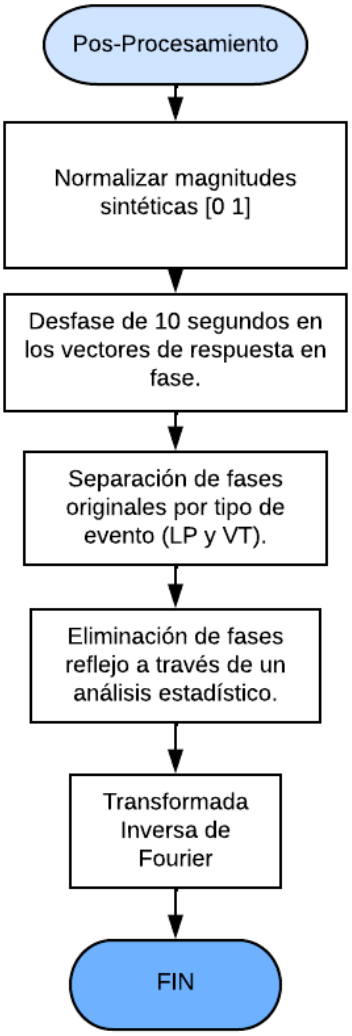
$$X' = a + \frac{(X - X_{min})(b - a)}{X_{max} - X_{min}},$$

$$X' = \frac{(X - X_{min})}{X_{max} - X_{min}},$$

donde X' es el vector normalizado, X_{min} el valor mínimo del vector y X_{max} el valor máximo del vector.

Figura 28

Diagrama de bloques del posprocesamiento



Como se comentó previamente, la reconstrucción de las señales se realiza con la magnitud y fase. De las fases originales previamente obtenidas de la base de datos de los eventos sísmicos, se realiza una separación de tipos de eventos (LP y VT) en dos vectores distintos. Al momento de reconstruir el vector de respuesta en frecuencia a una señal temporal con la fase original, las señales temporales sintéticas sufren una tendencia en desfasarse. Para corregir esta anomalía se opta por generar un vector de

respuesta en fase con un desfase de diez segundos, sin que se altere el vector de respuesta en frecuencia sintético generado por el modelo CGAN.

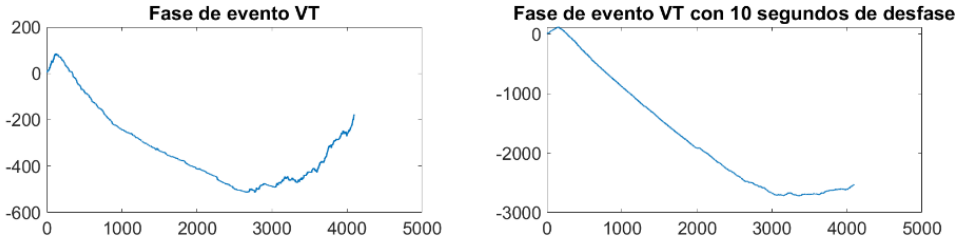
$$S' = concat[S(7000{:}8192), S(1{:}6999)]\,,$$

$$P' = unwrap\left(angle\left(FFT(S')\right)\right),$$

donde: S' es el vector de la señal desfasado diez segundos a través de una concatenación de vectores, los diez últimos segundos de datos pasan a ser los primeros y se concatenan con el resto de la señal. Las fases originales y las desfasadas en 10 segundos pueden observarse en la Figura 29.

Figura 29

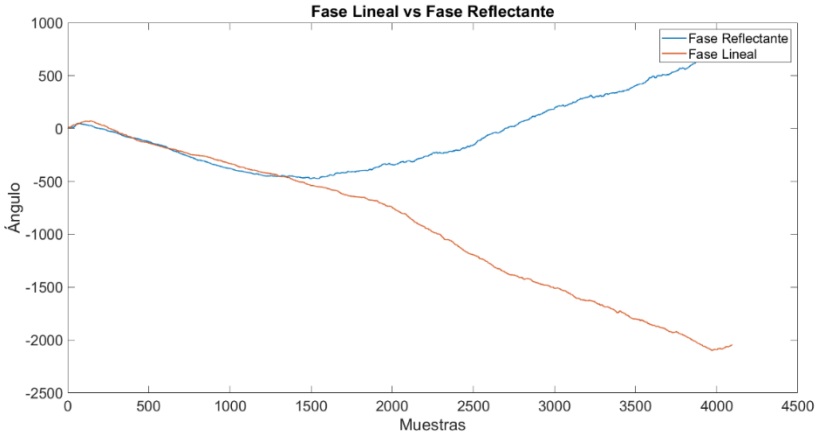
Respuesta en Fase de un evento VT y su desfase de diez segundos.



El próximo paso es la reconstrucción a una señal temporal del evento en cuestión a través de la IFFT. Para lograrlo, se emplean los vectores de respuesta en frecuencia generados y los vectores de respuesta en fase originales con su desfase de diez segundos. Se tienen dos tipos de fases: las lineales y reflectantes como se puede observar en la Figura 30.

Figura 30

Fases Lineal y Reflectante



Las fases reflejo causan una distorsión en la reconstrucción del evento temporal de la señal, a tal punto en que ambos tipos de eventos se vean muy similares pese a que su respuesta en frecuencia es distinta.

Si se emplean únicamente las fases lineales en la reconstrucción de los eventos, se logran encontrar claras diferencias en el dominio del tiempo del evento reconstruido sintéticamente. Por tal motivo se emplean las fases lineales de ambos tipos de eventos. Para descartar las fases reflejo se hace un análisis estadístico. En primer lugar, se obtiene la media de los últimos valores del vector de las fases, esto con el objetivo de no tener valores finales superiores al primer valor del vector de la fase, posteriormente se eliminan los valores que sobrepasen la media para lograr así que ningún valor final sobrepase al valor inicial. La media obtenida para eventos LP y VT se puede observar a continuación:

$$\bar{\bar{x}} = \frac{\sum_{i=1}^{N} X_i}{N}$$

$$\overline{\overline{x_{LP}}} = \frac{\sum_{i=1}^{N_{LP}} X_{i_{LP}}}{N_{LP}} = \frac{\sum_{i=1}^{N_{LP}} X_{i_{LP}}}{1031} = -1181$$

$$\overline{\overline{x_{VT}}} = \frac{\sum_{i=1}^{N_{VT}} X_{i_{VT}}}{N_{VT}} \frac{\sum_{i=1}^{N_{VT}} X_{i_{VT}}}{101} = -206$$

Los valores medios pueden observarse en la Figura 31 para eventos LP y en la Figura 32 para eventos VT.

Figura 31

Total de Fases Originales de Eventos LP

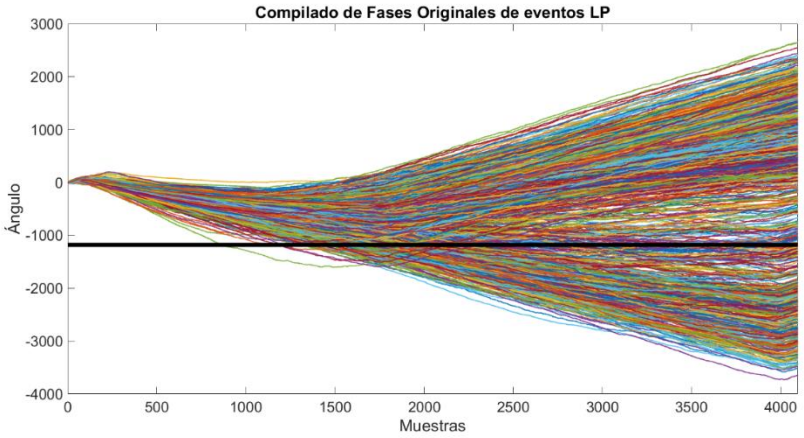
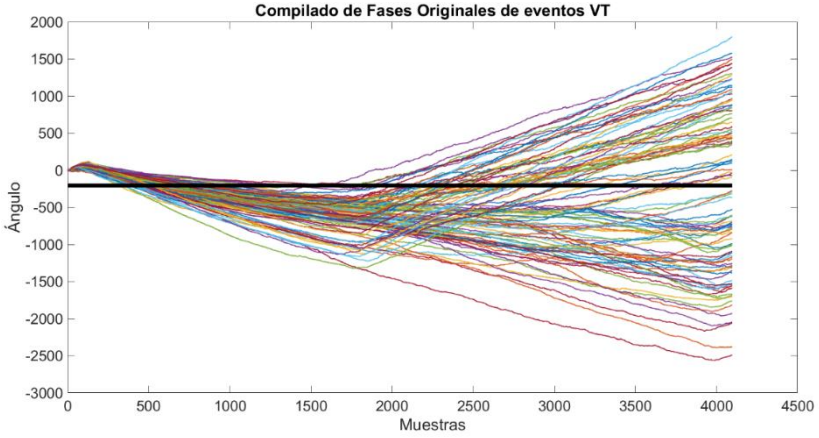


Figura 32

Total de Fases Originales de Eventos VT



Las fases lineales filtradas pueden observarse tanto en la Figura 33 para eventos LP como en la Figura 34 para eventos VT. Nótese la desaparición de las fases reflejo en ambos, y cómo ningún valor final es superior al valor inicial.

Figura 33

Fases Filtradas de Eventos LP

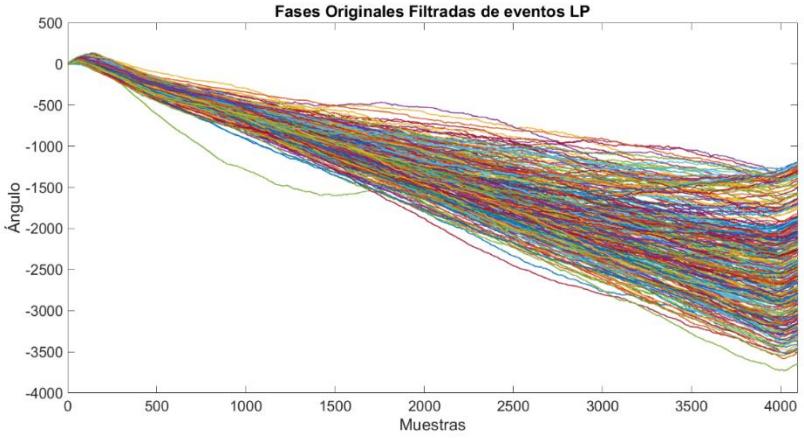
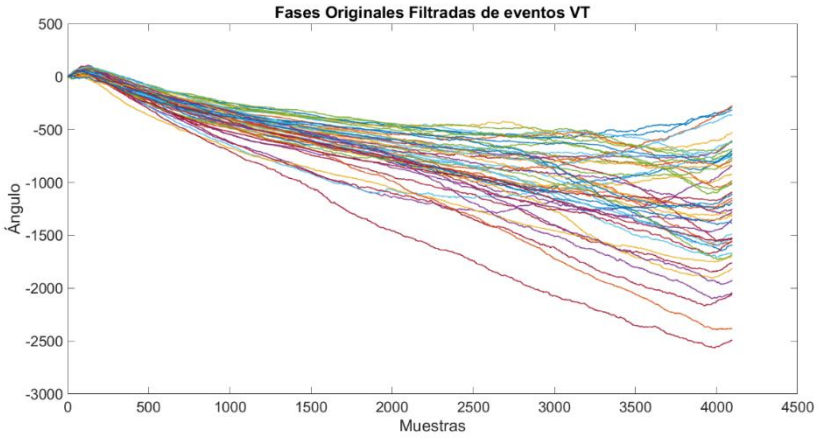


Figura 34

Fases Filtradas de Eventos VT



Con estos cambios en la fase, se realiza la IFFT con los vectores obtenidos.

Aplicación

Al realizar todo el proceso de entrenamiento dentro del software MATLAB® se aprovechó y desarrolló una interfaz gráfica dentro del programa debido a la versatilidad y facilidad de poder utilizar el modelo de red neuronal y vectores empleados durante todo el proceso de entrenamiento. En la Figura 35 se muestra la interfaz de la aplicación que tiene la finalidad de generar señales sintéticas en relación 50/50 (mitad por cada

tipo de evento LP y VT). Tiene distintas funciones como: posibilidad de agregar ruido a los extremos de la señal para aumentar su tamaño y simular un evento sismo-volcánico; definir la cantidad de muestras sintéticas (total) a generar; especificar la etiqueta y el directorio de los archivos a generar. Si bien las señales que se generan están en función de las muestras (4096 para los vectores de respuesta en frecuencia) los resultados presentados por la aplicación están en función de densidad espectral de potencia, con el método de Welch para 512 muestras. Este método utiliza estimaciones de la densidad espectral de potencia de diferentes segmentos de la serie temporal. Los periodos gramas modificados representan estimaciones aproximadamente no correlacionadas de la verdadera densidad espectral de potencia y el promedio reduce la variabilidad por lo que visualmente es más sencillo visualizar gráficamente la ubicación de la mayor cantidad de potencia en referencia a la frecuencia.

Figura 35

Pantalla principal de la interfaz gráfica del proyecto

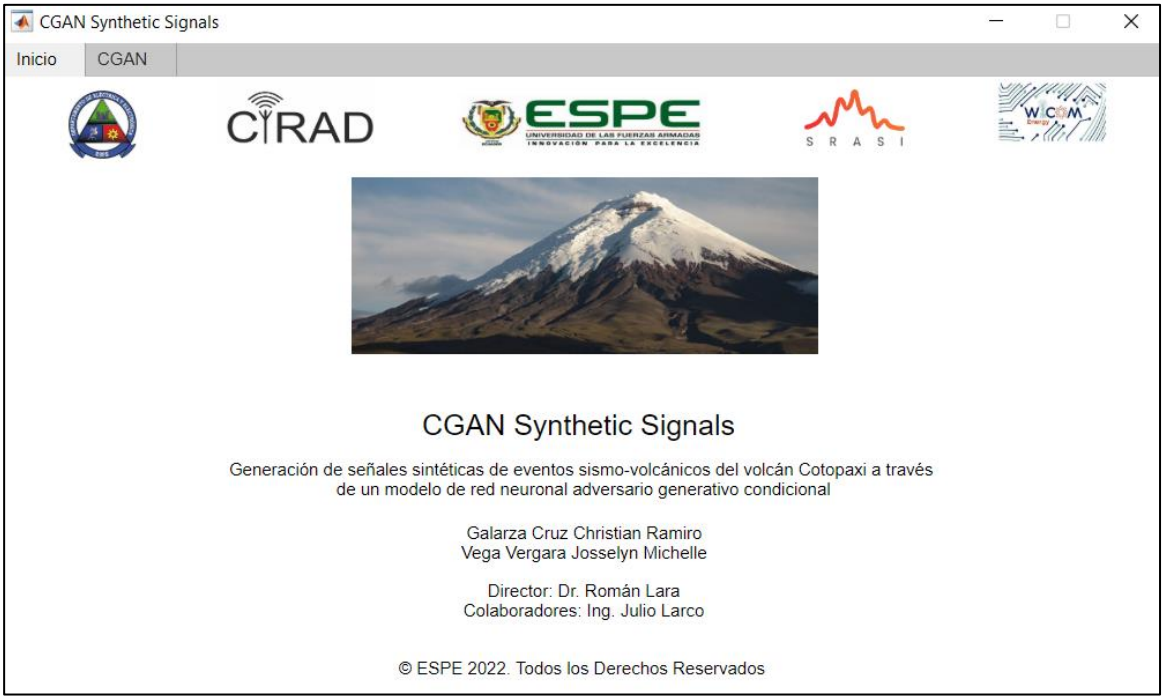


Figura 36

Aplicación del modelo CGAN

Evaluación

Para la evaluación de las señales LP y VT sintéticas generadas se utiliza la matriz de confusión y métricas estadísticas comunes en el área de *machine learning*, las cuales son: Exactitud (A_c del inglés, *accuracy*), Precisión (P_r), Sensibilidad (S_e), Especificidad (S_p) y Tasa de Error Balanceada (BER del inglés, *balance error rate)*.

Matriz de Confusión. Es una métrica que permite abordar el rendimiento de un método de clasificación. Se utiliza en problemas de clasificación tanto binaria como de multiclase, la matriz de confusión de la puede observar en la Tabla 7.

Tabla 7

Matriz de confusión

		Valores Actuales	
		Positivo	**Negativo**
Valores pronosticados	**Positivo**	Verdadero Positivo (VP)	Falso Positivo (FP)
	Negativo	Falso Negativo (FN)	Verdadero Negativo (VN)

Nota. En la tabla se presenta los parámetros de la matriz de confusión.

La matriz de confusión se compone de cuatro propiedades principales, que establecen la métrica de medición del clasificador: las tasas de Verdadero Positivo (VP), Falso Positivo (FP), Falso Negativo (FN) y Verdadero Negativo (VN). La tasa de VP es el número total de resultados o pronósticos correctos en los que la clase real fue positiva, la tasa de FP es el número total de resultados o pronósticos incorrectos en los que la clase real fue positiva., la tasa de VN es el número total de resultados o pronósticos correctos en los que la clase real era negativa y la tasa de FN es el número total de resultados o pronósticos incorrectos en los que la clase real era negativa (Tripathi et al., 2021).

Según estos valores, las medidas de rendimiento más utilizadas para la clasificación incluyen los valores A_c, P_r, S_e, S_p y BER. El cálculo de estos indicadores se basa en los valores de la matriz de confusión (Demir, 2022).

Las medidas de rendimiento son: la exactitud, que indica qué tan cerca está el resultado de una medición del valor verdadero, por lo que mientras más alto sea este valor se tiene que la clasificación fue correcta; la precisión indica el porcentaje de dispersión del conjunto de valores, por lo que se valida con qué frecuencia acierta el modelo; la sensibilidad indica la proporción de los casos positivos que fueron correctamente identificados; la especificidad indica los casos negativos que el algoritmo ha clasificado correctamente; y el BER indica el total de predicciones incorrectas en la

prueba. Se calculan las métricas de rendimiento respectivamente a través de las siguientes ecuaciones:

$$A_c(\%) = \frac{VP + VN}{VN + VP + FN + FP} \times 100\,,$$

$$P_r(\%) = \frac{VP}{VP + FP} \times 100\,,$$

$$S_e(\%) = \frac{VP}{VP + FN} \times 100\,,$$

$$S_p(\%) = \frac{VN}{VN + FP} \times 100\,,$$

$$BER = 1 - \frac{S_e + S_p}{2}.$$

Capítulo IV

Análisis de Resultados

En el presente capítulo se presenta el análisis de los resultados de acuerdo con la metodología propuesta en función de la prueba y error. Se identifican las distintas fallas observadas durante el entrenamiento del modelo CGAN hasta encontrar el mejor modelo con los mejores parámetros. Posteriormente se muestran resultados de las respuestas en frecuencia y señales de eventos LP y VT sintéticos generados con el modelo CGAN. Finalmente se muestra la evaluación de las señales LP y VT sintéticas con diferentes softwares, aplicativos de detección y clasificación de eventos sismo-volcánicos desarrollados por el Grupo de Investigación en Sistemas Inteligentes (WiCom-Energy) y el Centro de Investigación de Redes Ad Hoc (CIRAD), además de la evaluación visual realizada por los expertos del IGEPN.

Ajuste de los parámetros para un modelo CGAN óptimo

Durante los entrenamientos realizados al modelo CGAN, se observó que a través del ajuste de parámetros como las *epoch* y los *minibatch* se pueden resolver los problemas de fallas presentados y mejorar el resultado final.

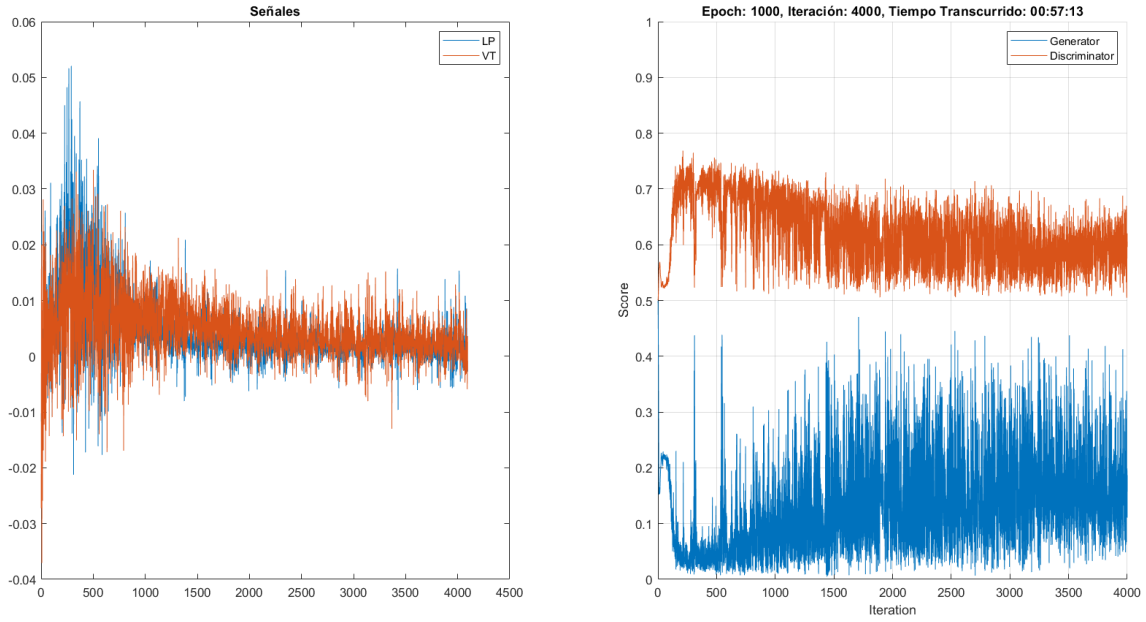
En principio se fijó el número de bloques, capas convolucionales y convolucionales transpuestas en los dos modelos del generador y discriminador. Esto conlleva a que el modelo al momento del entrenamiento no colapse y así se obtenga a la salida del discriminador valores distintos de cero, lo que al generador durante el entrenamiento le ayudaba a seguir un proceso de aprendizaje óptimo por recomendación de (Radford et al., 2016).

Adicionalmente, el ajuste del número de filtros también se mantuvo fijo debido a que es esencial a la hora de obtener mejores resultados durante el entrenamiento. Si las capas más profundas tienen una gran cantidad de filtros el modelo es capaz de buscar

características más ocultas de lo entrenado, sin embargo, una excesiva cantidad de filtros en esas capas puede llegar a ocasionar un efecto negativo debido a que la cantidad de características por aprender está directamente relacionada con un número máximo de filtros necesarios. Si se sobrepasa, se llega a tener un fallo del tipo colapso como se puede observar en la Figura 37.

Figura 37

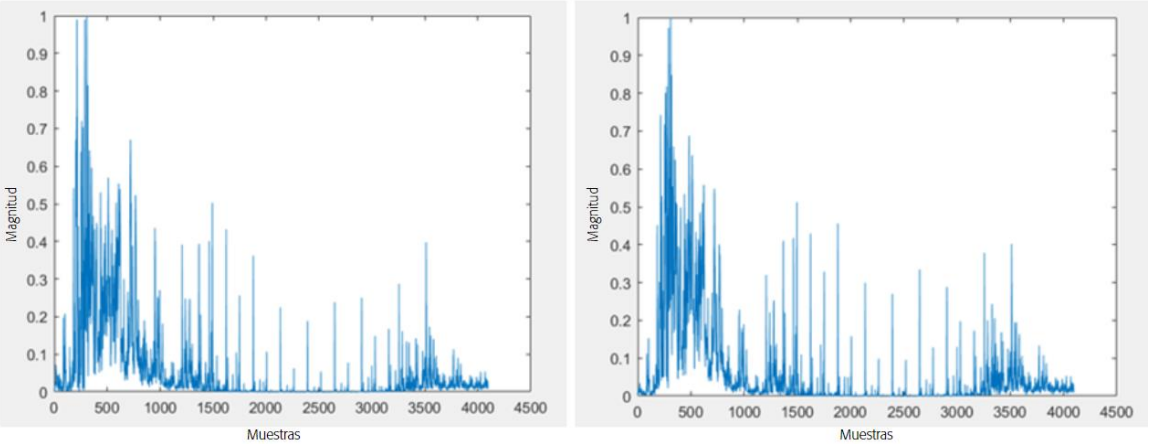
Modo colapso con exceso de filtros



En la Figura 38 se muestran los resultados obtenidos sobrepasa el número de filtros recomendados de los modelos Generador y Discriminador con un valor de 2056 filtros en sus capas más profundas. Adicionalmente, el tener gran número de filtros en la capa más profunda es ineficiente debido a que, por las características propias del hardware de la computadora empleada en el presente trabajo de investigación se llegó a tal punto de sobre esforzar los equipos y no culminar con todo el entrenamiento.

Figura 38

Vectores de respuesta en frecuencia del tipo LP generadas con un modelo colapsado



Por tal motivo, y de acuerdo a lo planteado en la metodología se procedió a fijar parámetros como el número de capas y filtros a los recomendados por Radford et al. (2016) y se variaron los parámetros como el número de iteraciones, *epoch* y *minibatch* en el proceso de entrenamiento para encontrar el modelo óptimo. Para identificar inicialmente el número de *epoch* en el entrenamiento del modelo CGAN se tomó como recomendación de Karras et al. (2020) el tener como referencia un número de *epoch* similar o cercana al número de datos totales. Se utilizó en el entrenamiento 1145 datos de señales sismo-volcánicas entre LP y VT por lo que se partió de este modo en un valor de 1000 *epoch* para variar el resto de los parámetros que son las iteraciones y el *minibatch*.

Como primer punto, se establecen cuantas iteraciones se requieren por cada *epoch*. En cada *epoch* se permuta aleatoriamente el orden de las señales que van a ser procesadas en el modelo. Se pueden establecer por ejemplo relaciones de 1:1, 1:2 y así sucesivamente entre iteraciones y *epoch*. Esa relación está dada por los *minibatch.*

Se fijó como punto de partida el uso de 1000 *epoch* para variar el número de iteraciones entre sí, adicionalmente se prueban distintas *epoch* para analizar el

comportamiento y considerar el uso de otros valores. En la Tabla 8 se muestra distintos *epoch* y *minibatch* utilizados hasta la búsqueda de los parámetros definitivos para un entrenamiento eficiente.

Tabla 8

Pruebas con diferentes parámetros de entrenamiento en el modelo CGAN

Epoch	Iteraciones	*Minibatch*	Diferencia de Score
1000	8000	140	0.62
1000	4000	256	0.25
1000	22000	50	0.55
2000	8000	256	0.25
3000	12000	256	0.30
3000	66000	50	0.40

Nota. En la tabla se presenta la variación de parámetros de entrenamiento como son las *epoch*, las iteraciones y los *minibatch*. Se calcula la diferencia de score de los modelos gráficamente.

Como se observa en la Tabla 8, se entrenó con 1000, 2000 y 3000 *epoch* y se varió la cantidad de *minibatch* con distintas combinaciones. Si se tiene una gran cantidad de iteraciones con un *minibatch* pequeño, como se puede observar en la Figura 39 con 1000 *epoch* y 22000 iteraciones, y en la Figura 40 con 3000 *epoch* y 66000 iteraciones, se genera durante el proceso de entrenamiento una falla documentada como modo colapso.

En este modo colapso, la puntuación entre el modelo del generador y discriminador fluctúa y varía entre sí, sin llegar a ser estable y denotar un camino fijo.

Figura 39

Primer error de modo colapso con un minibatch corto

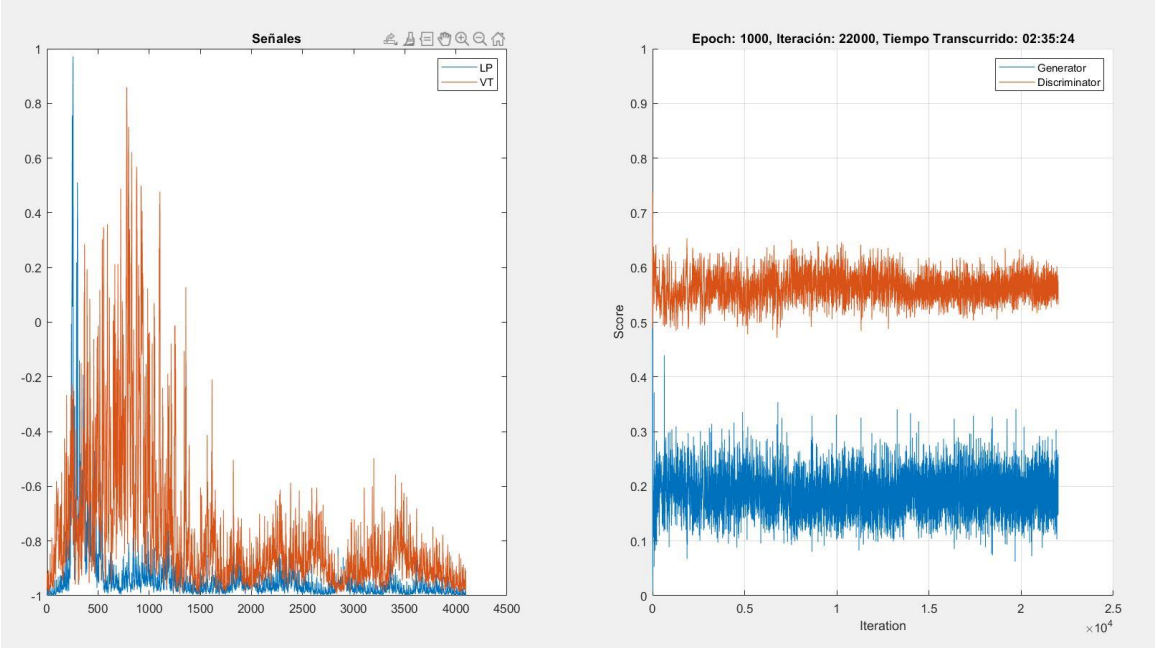
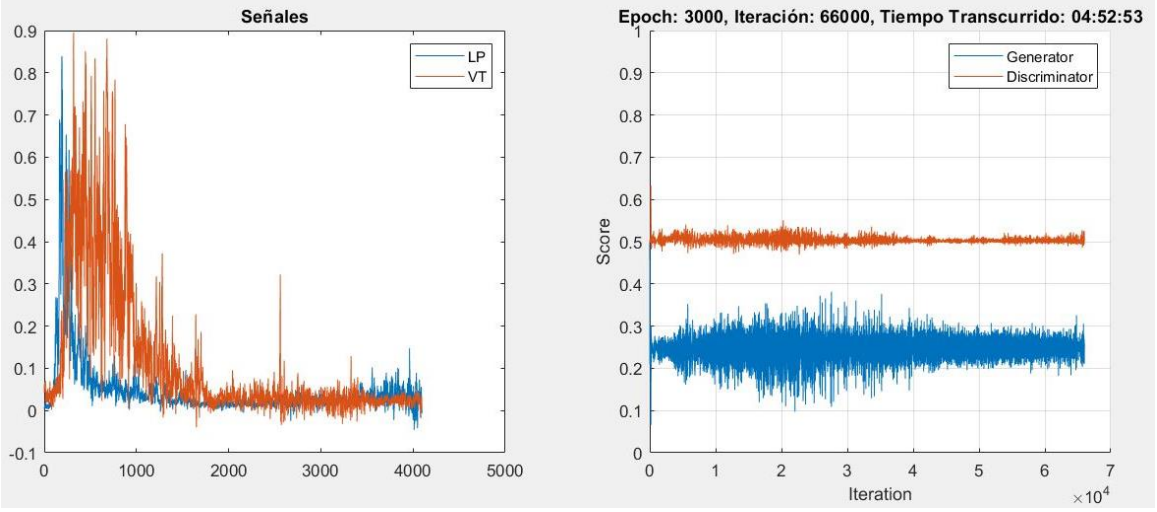


Figura 40

Segundo error de modo colapso con un minibatch corto

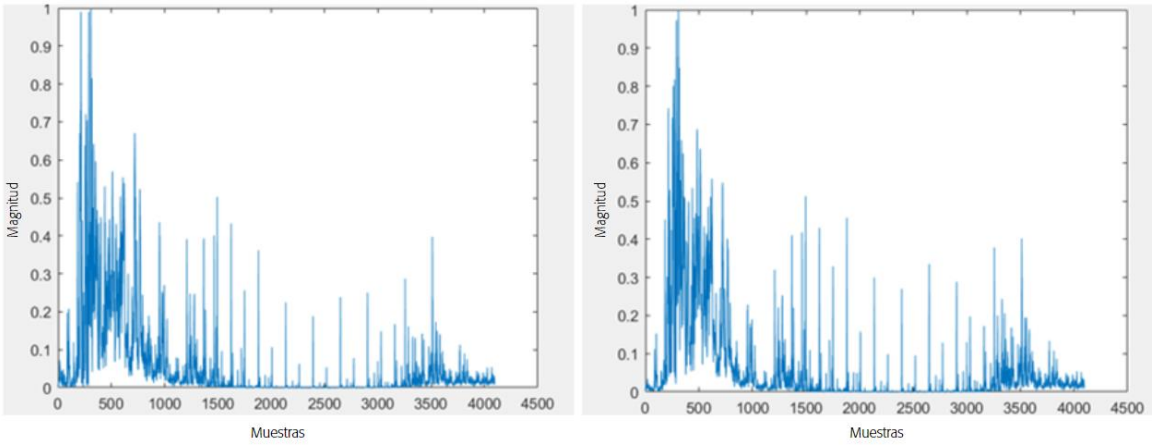


Las vectores de respuesta en frecuencia sintéticos formados a la salida del modelo generador son casi idénticos por cada tipo de evento como se puede observar

en la Figura 41, en el que el modelo genera vectores de respuesta en frecuencia de un evento LP sin tener diversidad o diferenciación entre sí.

Figura 41

Vectores de respuesta en frecuencia del tipo LP generadas con un modelo colapsado



Se observó que una relación menor de iteraciones y *epoch*, con un total de 256 *minibatch* indistintamente de la cantidad de *epoch* genera estabilidad en el proceso de entrenamiento de los modelos como se observa en la Figura 42, Figura 43 y Figura 44.

Figura 42

Caso 1: 256 Minibatch, 1000 Epoch, 4000 Iteraciones

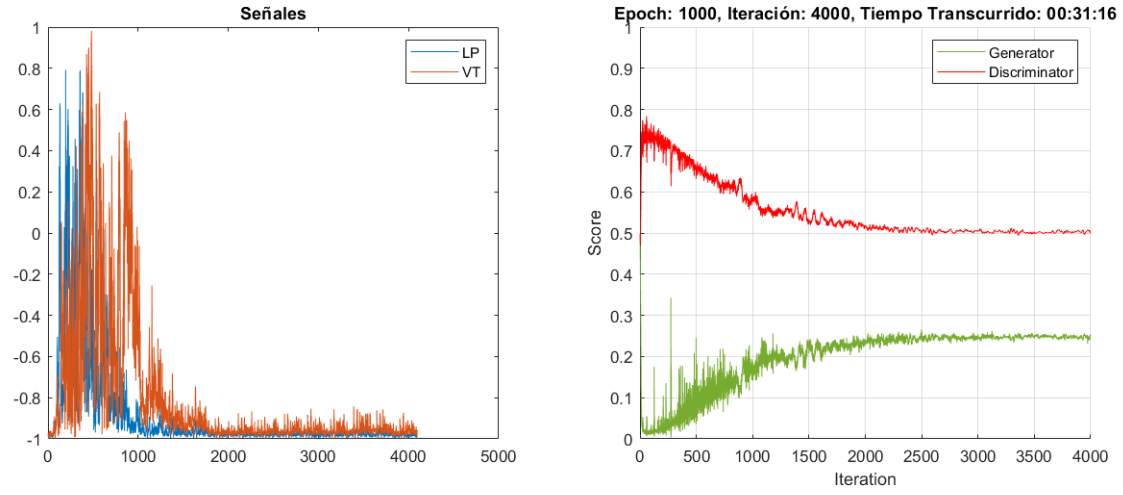


Figura 43

Caso 2: 256 Minibatch, 2000 Epoch, 8000 Iteraciones

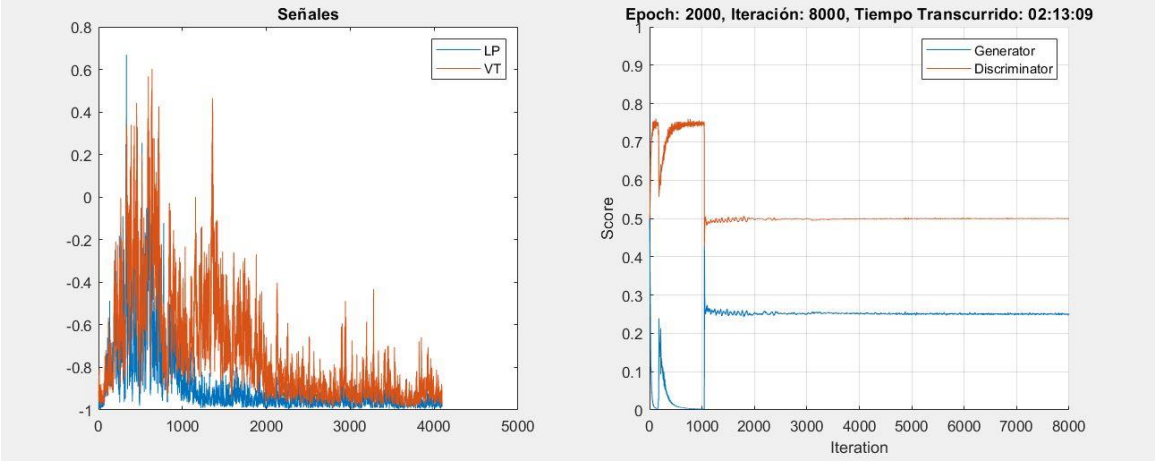
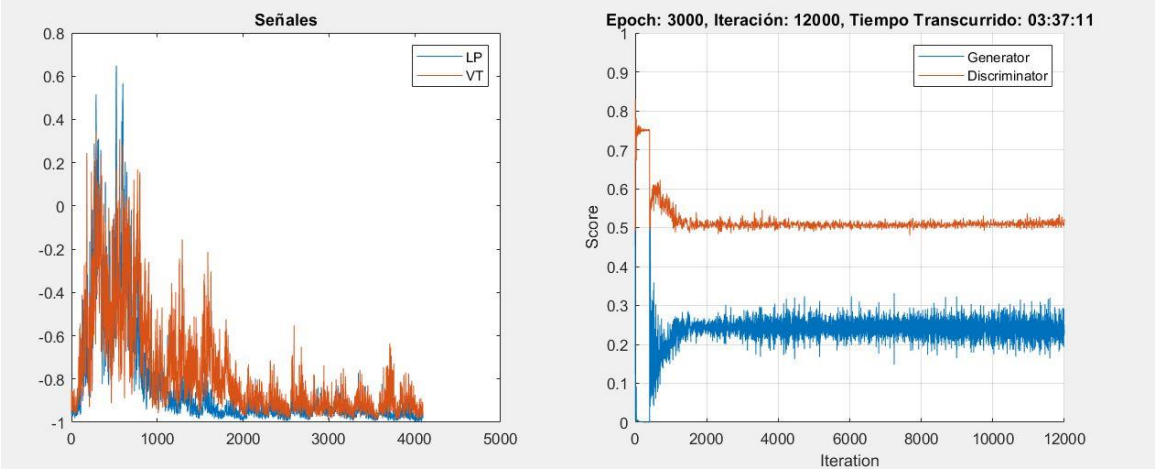


Figura 44

Caso 3: 256 Minibatch, 3000 Epoch, 12000 Iteraciones



Como se validan en las figuras previas se analizan tres casos con un punto en común: el uso de 256 como cantidad de *minibatch*. Se observa que existe una estabilidad con respecto a la puntuación de aprendizaje tanto del modelo generador como con el discriminador, donde se alcanzó valores más cercanos al 0 (estabilidad y entrenamiento perfecto) con valores de 0.25 y 0.30, se tiene como resultado un modelo que genera señales más similares a las originales. En el Caso 3 se observa que existe una estabilidad en la puntación sin embargo se comienza a desestabilizar conforme

pasan las iteraciones, esto se trata de un tipo de falla conocido como sobre entrenamiento y se puede validar el resultado en la forma de las señales. El mismo tipo de falla se puede encontrar en el Caso 2, la puntuación no llega a ser ruidosa sin embargo lleva demasiadas iteraciones lo cual lo vuelve estable y con ello ha ser ineficiente en tiempo de uso de recursos y con tendencia a ser sobre entrenada. El caso más ideal es el Caso 1, se llega a estabilizar en menor tiempo, la distancia entre las puntuaciones es la mejor conseguida con un valor de 0.25.

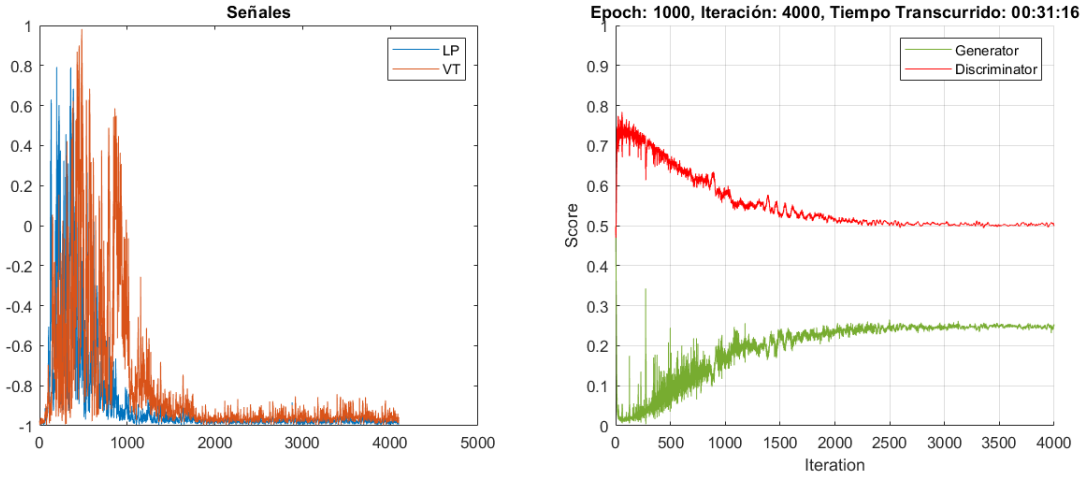
Por lo tanto, el modelo CGAN se ajustó a un total de 1000 *epoch* debido a la rápida convergencia del generador y discriminador con una diferencia de score de 0.25, igualmente se define que la cantidad de *epoch* es adecuada para el número de respuestas en frecuencia que se encontraban en la base de datos que ingresaban al modelo.

Resultados del modelo CGAN

En la Figura 45 se presentan las puntuaciones de entrenamiento del generador y el discriminador de la red CGAN final con un total de 1000 *epoch* con 4000 iteraciones. Se puede observar como en las iteraciones iniciales los dos modelos entrenan. El discriminador se encarga de evaluar y retroalimenta todas las puntuaciones al generador. Cerca de las 2500 iteraciones las puntuaciones de los dos modelos se estabilizan y se mantienen constantes durante todo el entrenamiento hasta su finalización, lo cual proporcionó estabilidad y convergencia.

Figura 45

Modelo Final Generado



Con los ajustes realizados al modelo CGAN, se obtuvieron los siguientes resultados para las respuestas en frecuencia LP y VT, las mismas que fueron creadas a partir del modelo del Generador. Para los eventos LP se puede observar magnitudes naturales en la Figura 46 y en la Figura 47 para las magnitudes sintéticas. Adicionalmente para eventos VT se observan magnitudes reales en la Figura 48 y las magnitudes sintéticas generadas con el modelo CGAN se observan en la Figura 49. En los dos casos se verifica que las magnitudes sintéticas generadas tienen gran cantidad de similitudes con las respuestas en frecuencia naturales con las cuales se entrena el modelo.

Figura 46

Magnitudes de la respuesta en frecuencia LP naturales

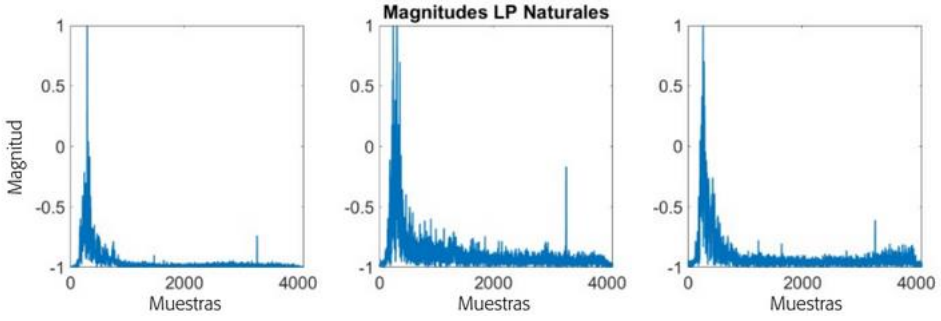


Figura 47

Magnitudes de la respuesta en frecuencia LP generadas en el modelo CGAN

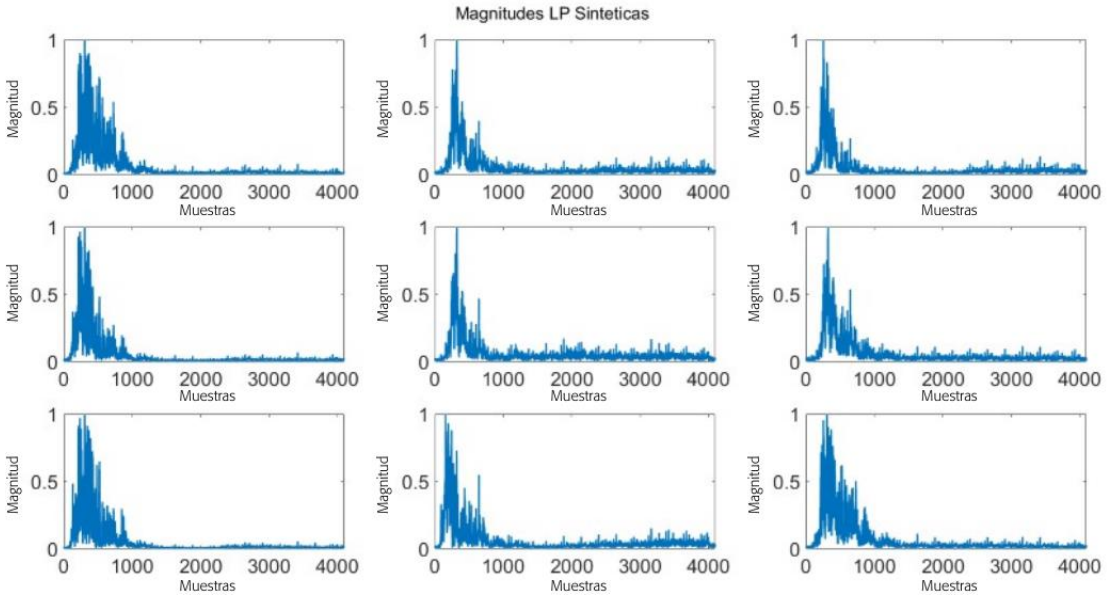


Figura 48

Magnitudes de la respuesta en frecuencia VT naturales

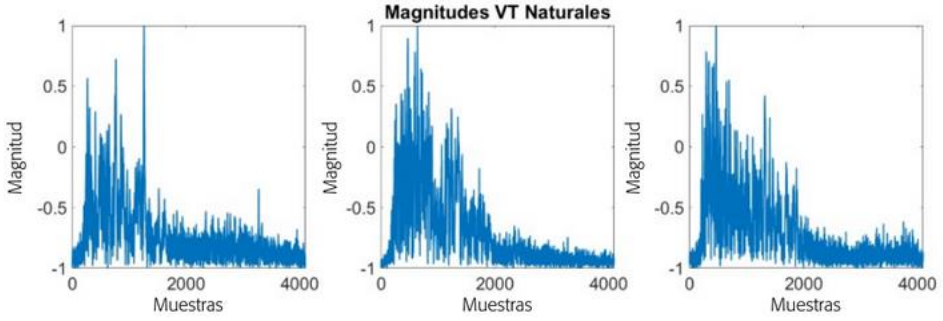
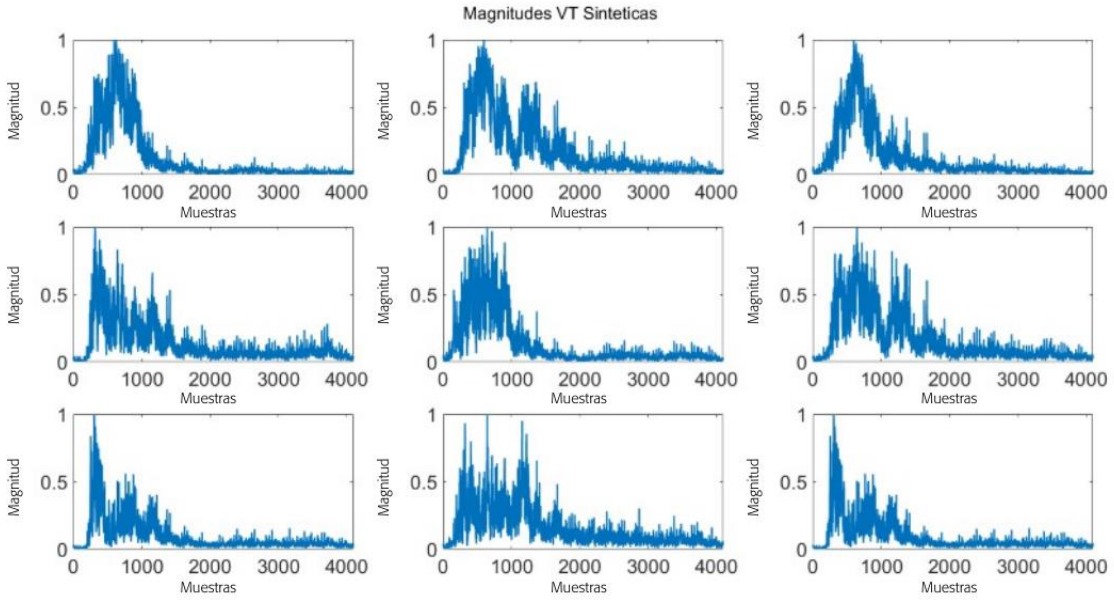


Figura 49

Magnitudes de la respuesta en frecuencia VT generadas en el modelo CGAN



Señales LP y VT en el tiempo

En la Figura 50 y Figura 51 se observa la magnitud, la fase y la señal temporal de los eventos LP y VT respectivamente. Nótese la utilización del uso de las fases naturales que tienen una forma lineal. Se muestran los vectores de respuesta en frecuencia con los 4096 puntos, resultante inicial del modelo CGAN.

Figura 50

Magnitud LP Sintético, fase LP natural y señal sintética sismo-volcánica de tipo LP.

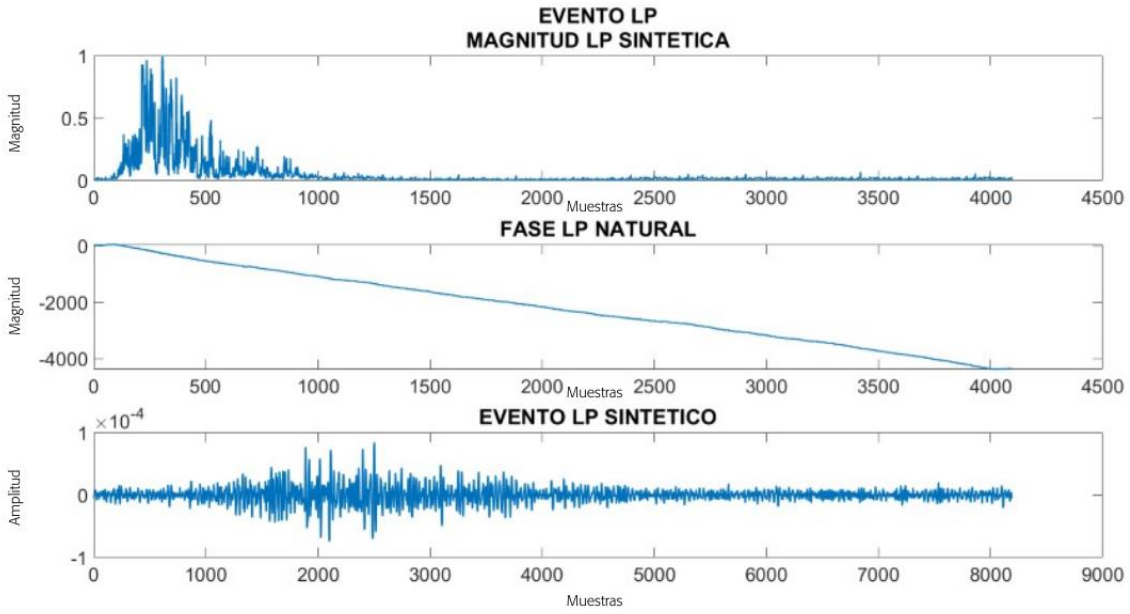
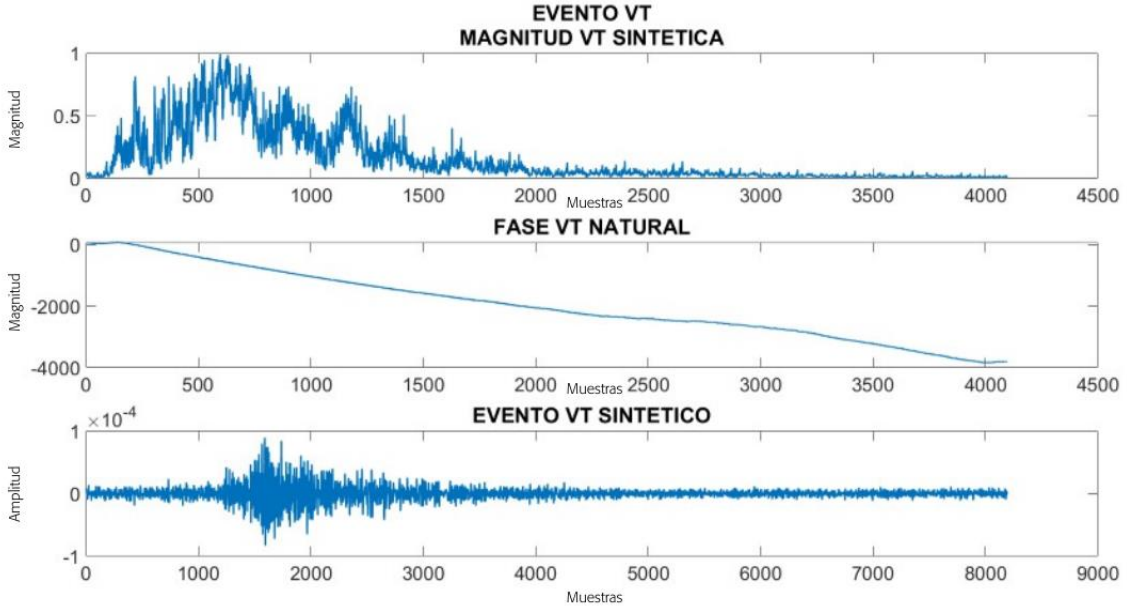


Figura 51

Magnitud VT Sintético, fase VT natural y señal sintética sismo-volcánica de tipo VT.



En las Figura 52,Figura 53,Figura 54, se presentan señal sismo-volcánica LP y en las Figura 55,Figura 56,Figura 57 se encuentran señal sismo volcánica VT sintéticas

en el tiempo y su PSD respectivamente generadas de la aplicación desarrollada. Como se comentó previamente, se muestra la densidad espectral de potencia de Welch de 512 puntos, en el que se visualiza claramente la concentración de potencia de los eventos LP y VT respectivamente.

Figura 52

Señal sismo volcánica LP y su PSD

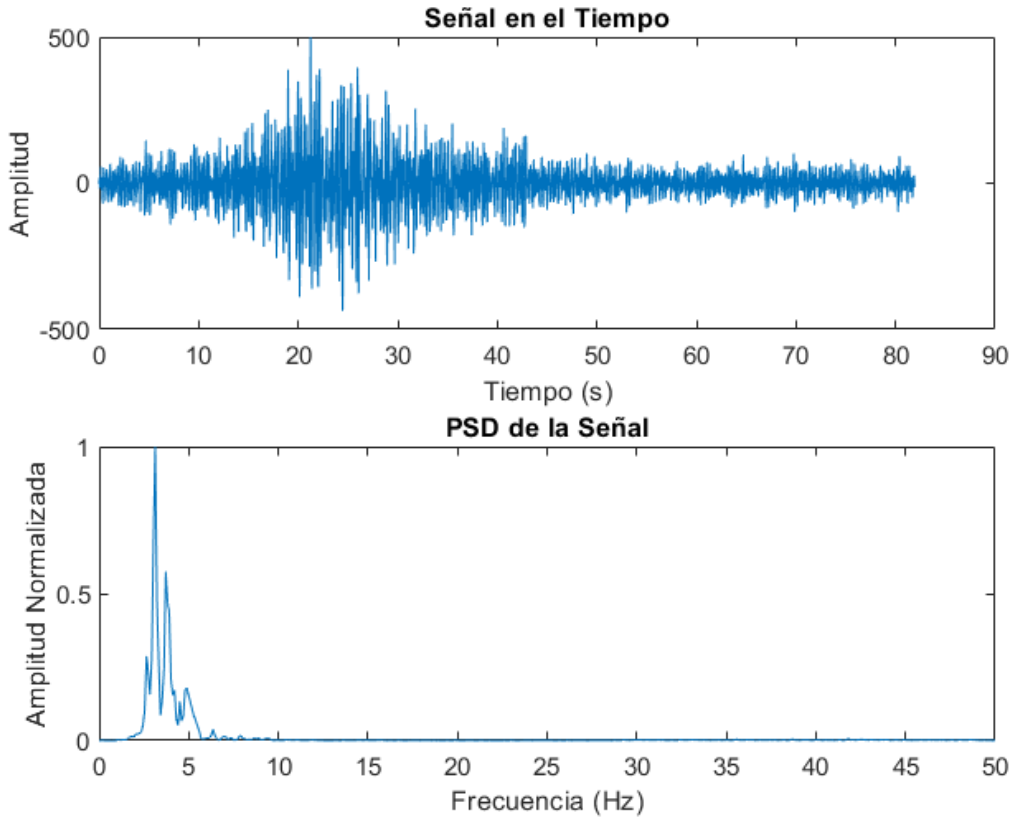


Figura 53

Señal sismo volcánica LP y su PSD

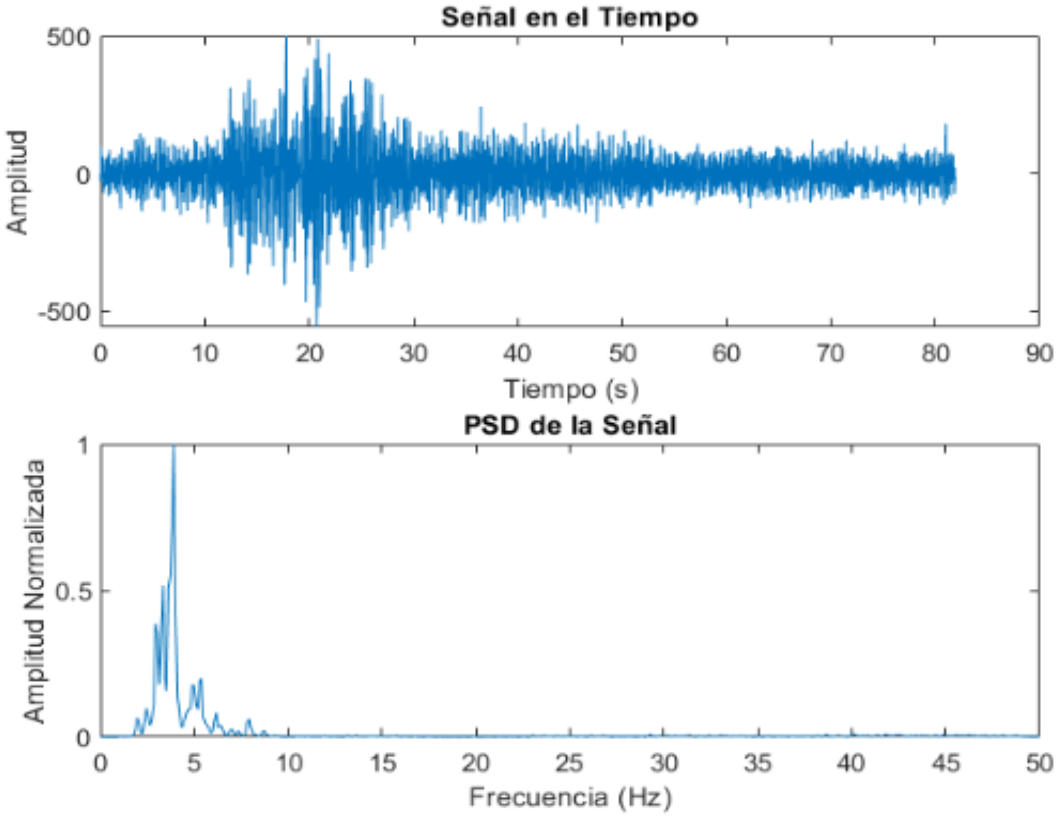


Figura 54

Señal sismo volcánica LP y su PSD

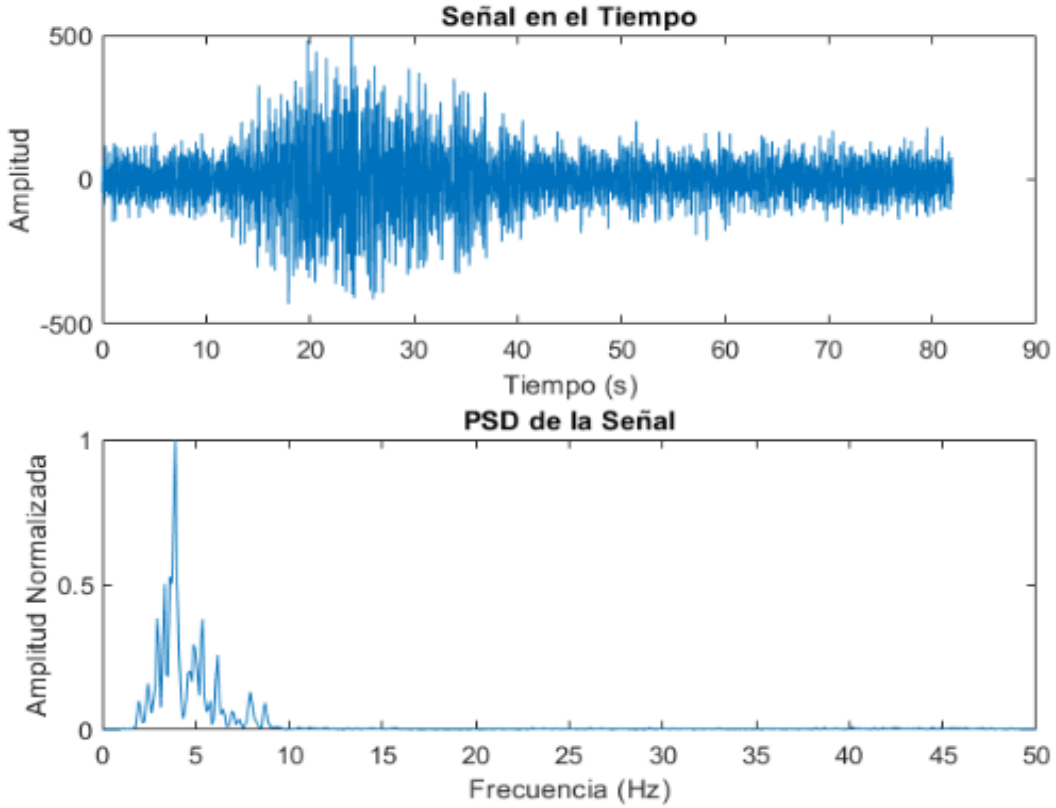


Figura 55

Señal sismo volcánica VT y su PSD

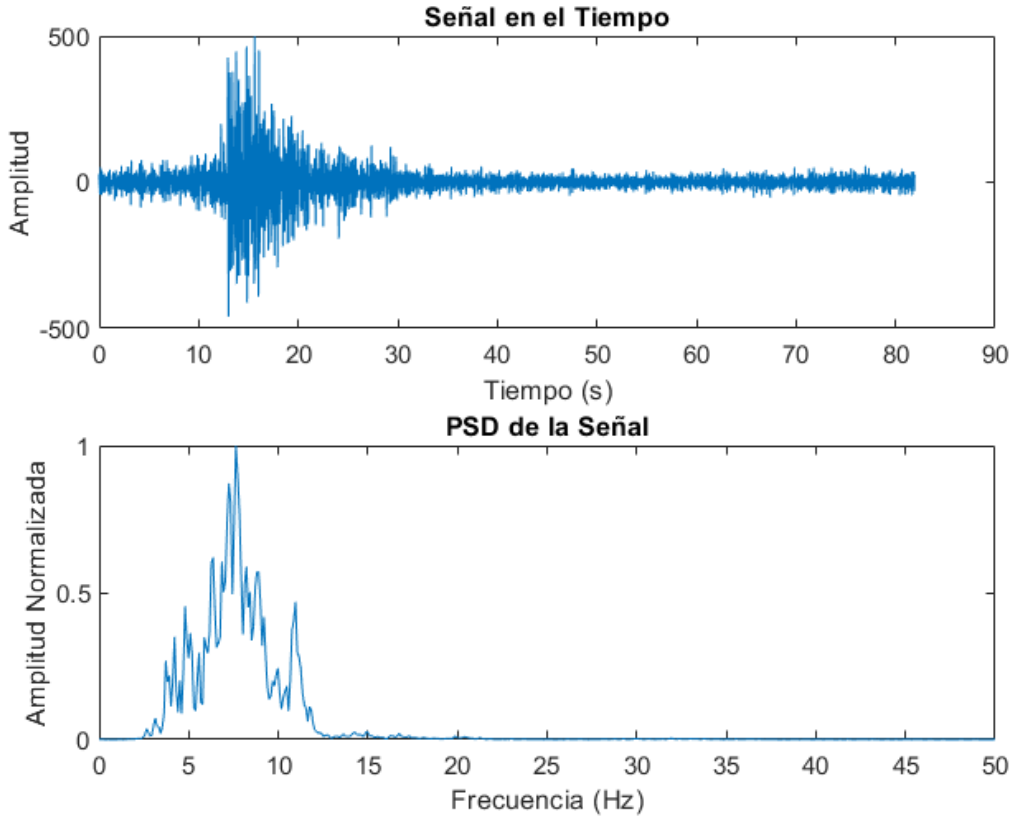


Figura 56

Señal sismo volcánica VT y su PSD

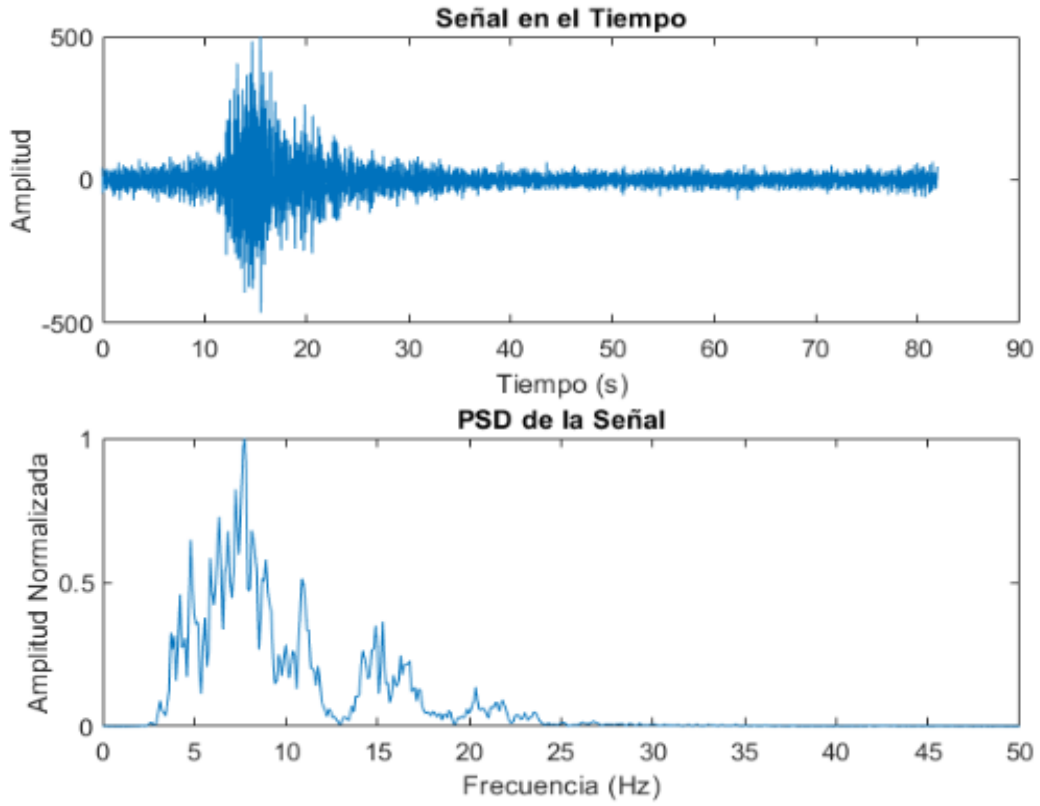
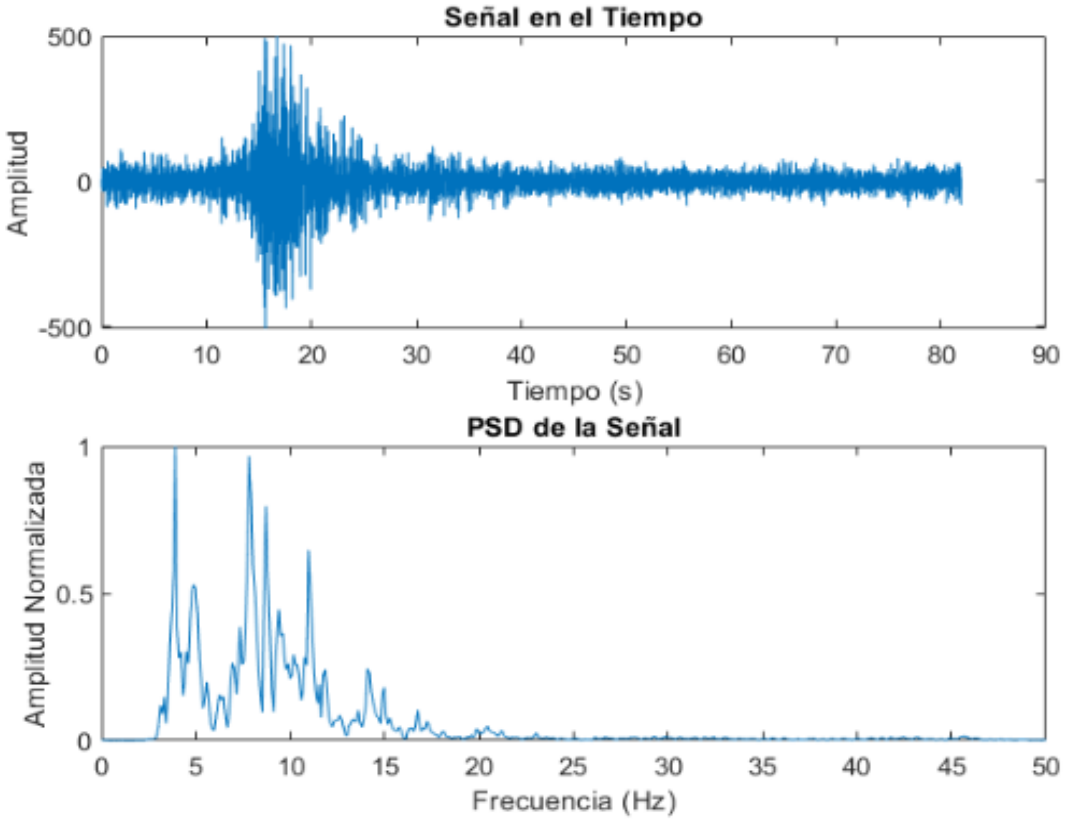


Figura 57

Señal sismo volcánica VT y su PSD



Evaluación de las señales sintéticas sismo-volcánicas LP y VT a través de distintas herramientas

Para la evaluación de las señales sintéticas generadas se emplearon dos herramientas de software en el que se detectan y clasifican señales sismo-volcánicas del volcán Cotopaxi, además se presenta la clasificación visual de los eventos a través de los expertos de la IGEPN por medio de una encuesta en línea. Para las tres herramientas que permiten clasificar las señales según el evento sismo-volcánico LP o VT se obtienen métricas de evaluación utilizadas en el área de inteligencia artificial (R. Lara-Cueva et al., 2016), como lo son la exactitud, precisión, sensibilidad, especificidad y BER.

Software de reconocimiento de eventos sismo-volcánicos

La primera aplicación utilizada es un sistema de reconocimiento de eventos sismo-volcánicos el cual se basa en redes neuronales convolucionales. El sistema tiene un 99% de éxito durante la detección y un 97% en la clasificación (Lara Mina, 2021). En la Figura 58, se puede observar la pantalla de inicio en donde se configura el número de señales, se cargan todas las rutas para el funcionamiento del sistema. En las Figura 59 y Figura 60 se presentan imágenes del software en funcionamiento donde clasifica señales de eventos sismo-volcánicos LP y VT respectivamente.

Figura 58

Pantalla de configuración software de reconocimiento de eventos sismo-volcánicos

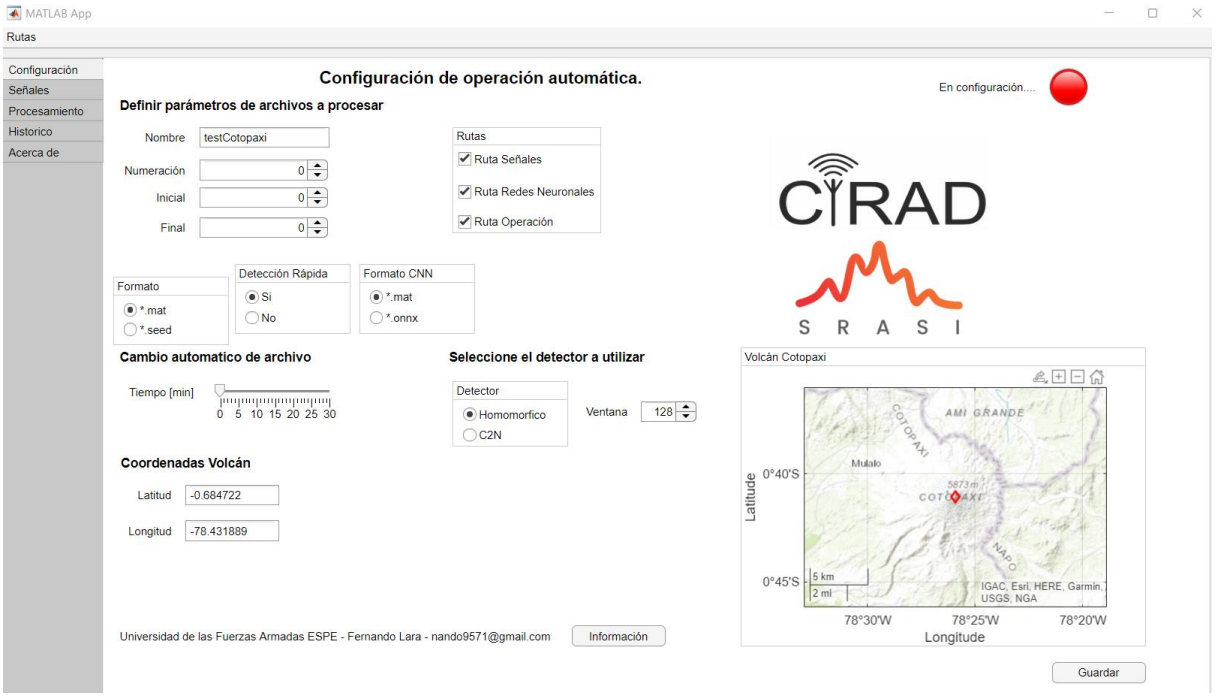


Figura 59

Señal clasificada como LP

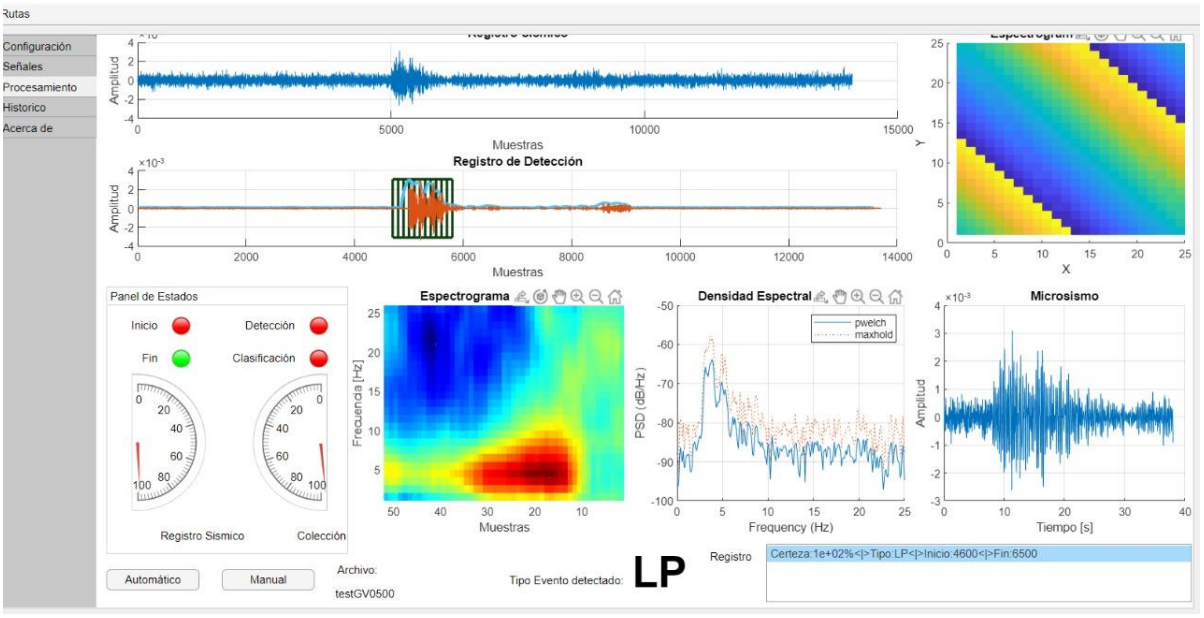
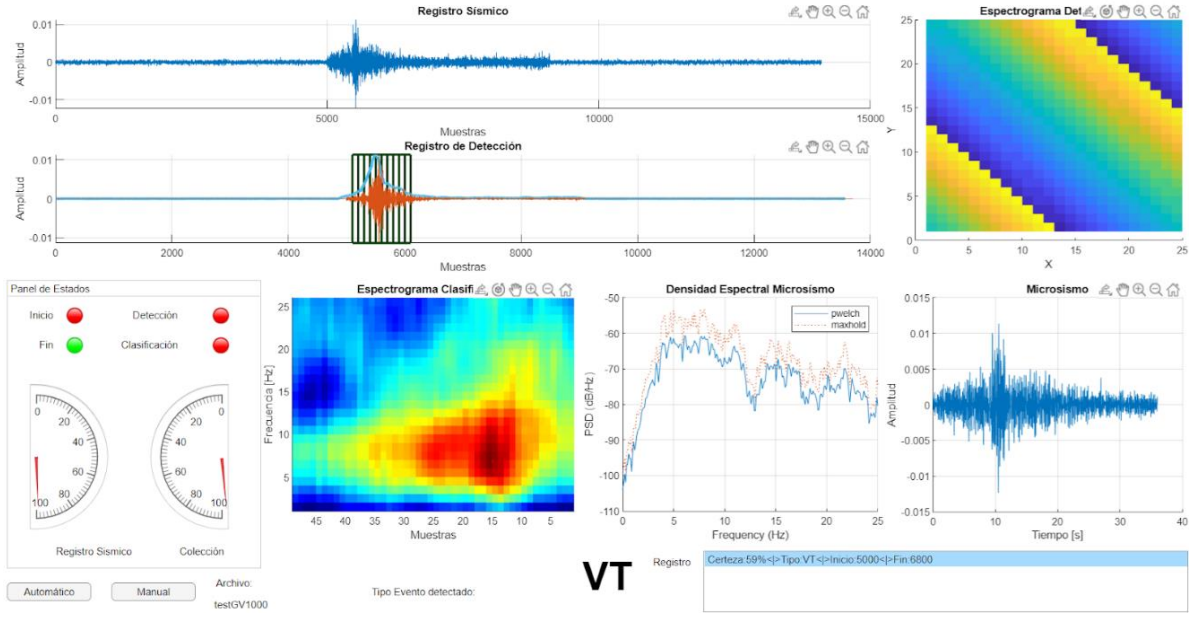


Figura 60

Señal clasificada como VT



En la herramienta de Lara Mina (2021) se ingresó 1000 señales sintéticas, las cuales se dividen equitativamente en 500 señales sismo-volcánicos del tipo LP y 500

señales sismo-volcánicos del tipo VT. Todas las señales ingresadas en la herramienta fueron detectadas y clasificadas. En la Figura 61 y Tabla 9 se presenta la matriz de confusión y las métricas de evaluación del sistema de reconocimiento de eventos sismo-volcánicos en el que se obtuvo una exactitud del 90.40% para la clasificación de los eventos LP y VT.

Tabla 9

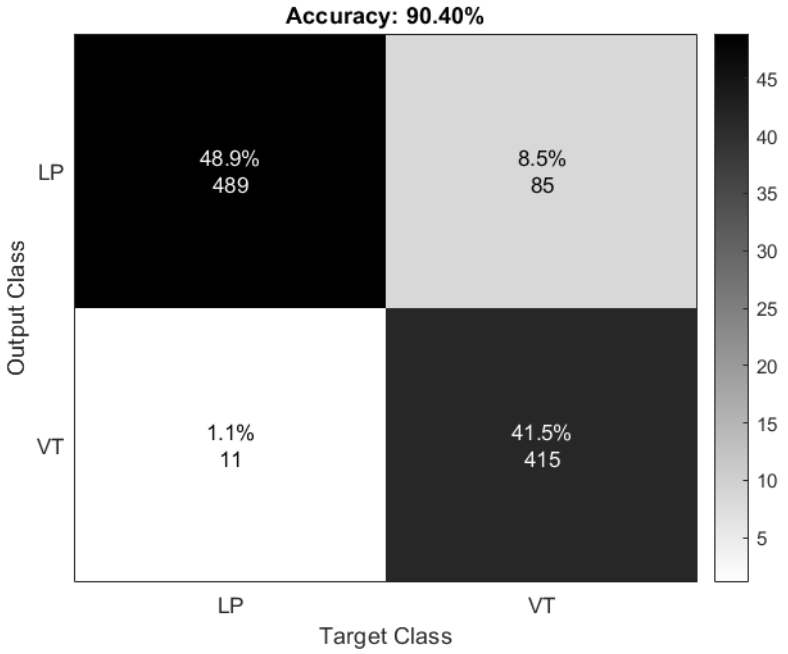
Métricas de evaluación sistema de reconocimiento de eventos sismo-volcánicos

Herramientas de clasificación	Exactitud	Precisión	Sensibilidad	Especificidad	BER
Sistema de Reconocimiento	90.40 %	85.19 %	97.80 %	83.00 %	0.10

Nota. En la tabla de presenta la Exactitud, Precisión, Sensibilidad, Especificidad, BER, del sistema de reconocimiento de eventos sismo-volcánicos, utilizado para evaluar las señales sismo-volcánicas LP y VT.

Figura 61

Matriz de confusión del sistema de reconocimiento



Clasificador de evento sismo-volcánicos con autoencoder

La segunda herramienta empleada es un autoencoder el cual consta de 257 características y tiene una exactitud del 98% para la clasificación de eventos sismo-volcánicos. De igual forma que en la anterior herramienta, se ingresaron 1000 señales sintéticas, las cuales se dividen en 500 señales sismo-volcánicas LP y 500 señales sismo-volcánicos VT. Se presentan los resultados en la Figura 62 a través de una matriz de confusión, se llegó a una exactitud del 97.50%. En la Tabla 10 se ubican todas las métricas de evaluación Autoencoder.

Figura 62

Matriz de confusión del autoencoder

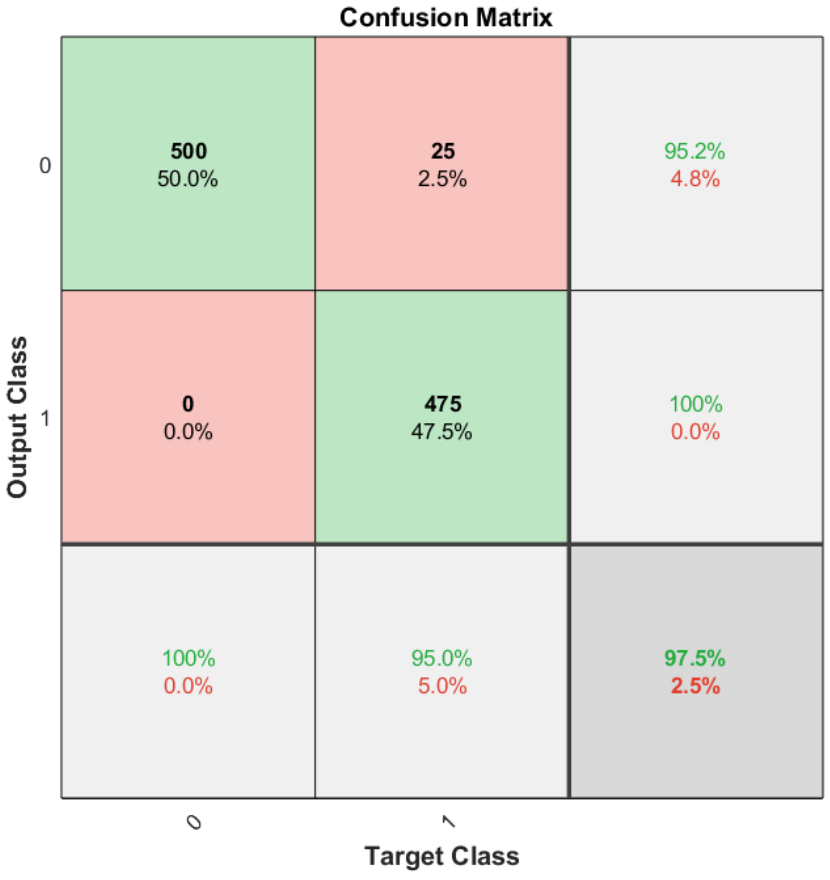


Tabla 10

Métricas de evaluación Autoencoder

Herramientas de clasificación	Exactitud	Precisión	Sensibilidad	Especificidad	BER
Autoencoder	97.50 %	95.20 %	100.0 %	95.00 %	0.03

Nota. En la tabla de presenta la Exactitud, Precisión, Sensibilidad, Especificidad, BER, del Autoencoder, utilizado para evaluar las señales sismo-volcánicas LP y VT.

Evaluación visual a través de los expertos del IGEPN

Para la evaluación visual de las señales, se realizó un formulario en la plataforma Google Forms, en el que se presentan un total de veinte señales de Eventos Sismo-volcánicos LP y VT de las cuales hay: cinco señales LP sintéticas, cinco señales VT sintéticas, cinco señales LP originales y cinco señales VT originales. Las señales se ubicaron en el formulario intercaladas aleatoriamente. La encuesta fue aplicada a nueve expertos del IGEPN y a un experto del Servicio Geológico Colombiano.

Con el conjunto de datos recopilado se realiza un análisis de dos variables: el tipo de evento (LP versus VT) y la naturaleza de origen de la señal (sintética versus real). El primer análisis permite identificar si los eventos sintéticos logran confundirse entre los eventos reales del mismo tipo, sea LP o VT. El segundo análisis permite identificar si el grupo que respondió la encuesta logra diferenciar una señal sintética de una real y viceversa independientemente del tipo de evento.

Análisis de resultados entre tipos de eventos LP y VT. En la Tabla 12 se tabulan los resultados de los diez expertos a través de la diferencia entre eventos LP y VT. Se obtiene una exactitud del 90.5% presentada como matriz de confusión en la .

Figura 63. En la Tabla 11 se ubican todas las métricas de evaluación Clasificación Visual.

Tabla 11

Métricas de evaluación Clasificación Visual

Herramientas de clasificación	Exactitud	Precisión	Sensibilidad	Especificidad	BER
Clasificación Visual	90.50 %	87.85 %	94.00 %	87.00 %	0.10

Nota. En la tabla de presenta la Exactitud, Precisión, Sensibilidad, Especificidad, BER, del Clasificación Visual, utilizado para evaluar las señales sismo-volcánicas LP y VT.

Tabla 12

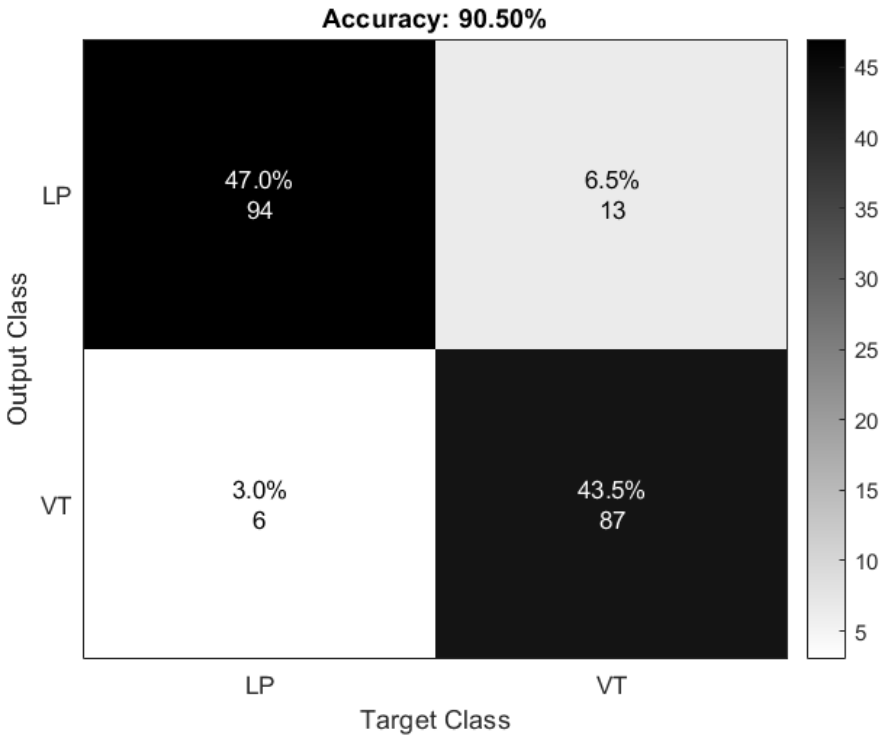
Resultados del formulario entre eventos LP y VT

Nº	Tipo de Señal	LP	VT
1	LP Sintético	100%	0%
2	LP Natural	100%	0%
3	VT Natural	0%	100%
4	LP Sintético	90%	10%
5	VT Sintético	10%	90%
6	VT Sintético	50%	50%
7	LP Natural	100%	0%
8	VT Natural	0%	100%
9	LP Sintético	90%	10%
10	VT Sintético	0%	100%
11	VT Natural	0%	100%
12	LP Natural	100%	0%
13	LP Sintético	90%	10%
14	LP Sintético	90%	10%
15	VT Sintético	0%	100%
16	VT Sintético	10%	90%
17	LP Sintético	100%	0%
18	VT Sintético	30%	70%
19	LP Sintético	80%	20%
20	VT Sintético	30%	70%

Nota. En la tabla se observa los resultados obtenidos del formulario resuelto por expertos del IGEPN.

Figura 63

Matriz de confusión entre eventos LP y VT



Análisis de resultados entre señales sintéticas y reales. En la Tabla 14 se tabulan los resultados de los trece expertos a través de la diferencia en la naturaleza de origen de los eventos, entre sintéticos y reales. Se tienen un total de 280 señales entre LP naturales, VT naturales, LP sintéticas y VT sintéticas.

Tabla 13

Métricas de evaluación Clasificación Visual

Herramientas de clasificación	Exactitud	Precisión	Sensibilidad	Especificidad	BER
Clasificación Visual	48.21 %	48.38 %	53.57 %	42.85 %	0.52

Nota. En la tabla de presenta la Exactitud, Precisión, Sensibilidad, Especificidad, BER, del Clasificación Visual, utilizado para evaluar las señales sismo-volcánicas LP y VT.

Se obtiene una exactitud del 48.21% presentada como matriz de confusión en la Figura 64. En la Tabla 13 Tabla 11 se ubican todas las métricas de evaluación Clasificación Visual.

Tabla 14

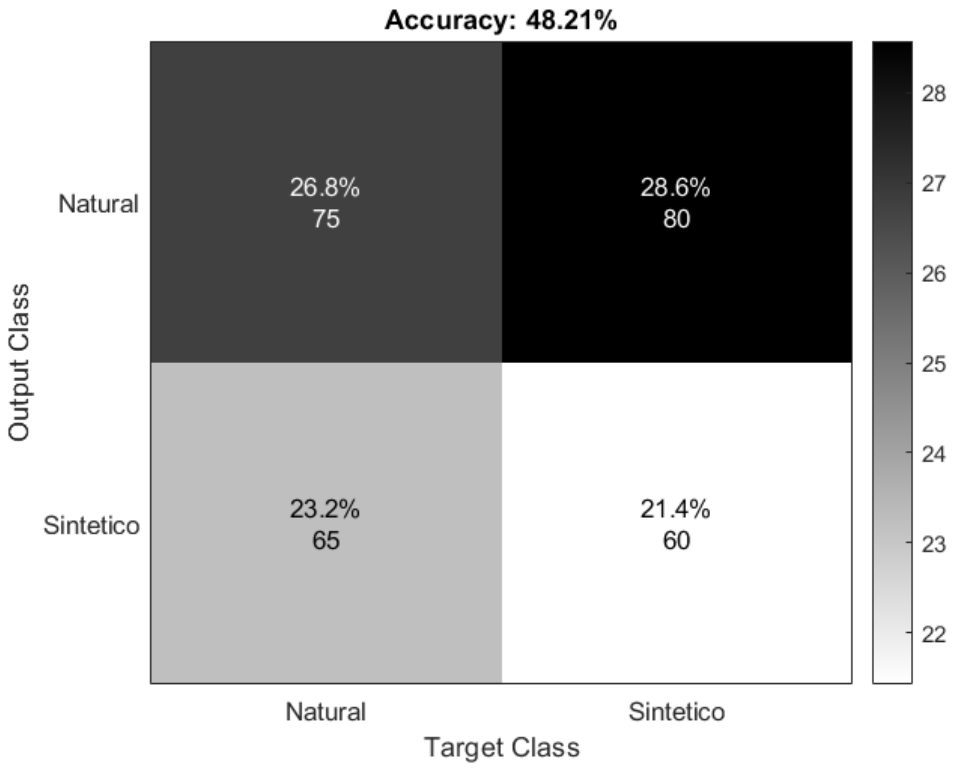
Resultados del formulario entre señales sintéticas y reales

Nº	Tipo de Señal	Sintético	Natural
1	LP Sintético	40%	60%
2	LP Natural	60%	40%
3	VT Natural	40%	60%
4	LP Sintético	50%	50%
5	VT Sintético	40%	60%
6	VT Sintético	60%	40%
7	LP Natural	10%	90%
8	VT Natural	30%	70%
9	LP Sintético	40%	60%
10	VT Sintético	10%	90%
11	VT Natural	60%	40%
12	LP Natural	80%	20%
13	LP Sintético	40%	60%
14	LP Sintético	30%	70%
15	VT Sintético	50%	50%
16	VT Sintético	70%	30%
17	LP Sintético	30%	70%
18	VT Sintético	40%	60%
19	LP Sintético	40%	60%
20	VT Sintético	60%	40%
21	VT Natural	70%	30%
22	LP Natural	60%	40%
23	VT Natural	40%	60%
24	LP Natural	10%	90%
25	VT Natural	30%	70%
26	LP Natural	70%	30%
27	VT Natural	20%	80%
28	LP Natural	70%	30%

Nota. En la tabla se observa los resultados obtenidos del formulario resuelto por expertos del IGEPN.

Figura 64

Matriz de confusión entre señales sintéticas y reales



Resultados de las herramientas empleadas

En la Tabla 15 se presentan para las tres herramientas de clasificación de señales empleadas, cinco parámetros de análisis: exactitud, precisión, sensibilidad, especificidad y BER. Se recopilan los resultados de clasificación entre tipos de eventos LP y VT.

Adicionalmente se realizan pruebas con el autoencoder, en el que se experimenta con diferente número de eventos sismo-volcánicos sintéticos. Se agrega el número de datos de eventos sintéticos del 20%, 40%, 60%, 80% y se disminuye el número datos reales, así de esta forma analizar la naturaleza de clasificación en el autoencoder, mientras más señales sintéticas aparecen.

Tabla 15

Resumen de resultados de las métricas de evaluación en las tres herramientas empleadas

Herramientas de clasificación	Exactitud	Precisión	Sensibilidad	Especificidad	BER
Sistema de Reconocimiento	90.40 %	85.19 %	97.80 %	83.00 %	0.10
Autoencoder	97.50 %	95.20 %	100.0 %	95.00 %	0.30
Clasificación Visual	90.50 %	87.85 %	94.00 %	87.00 %	0.10

Nota. En la tabla de presenta la Exactitud, Precisión, Sensibilidad, Especificidad, BER, de las 3 herramientas utilizadas para evaluar las señales sismo-volcánicas LP y VT.

Se utilizaron un total de 200 eventos sismo-volcánicos entre los cuales consta de 100 eventos LP y 100 eventos VT con la variación de los sintéticos en todos los casos. Esto se puede observar en la Tabla 16, se observa como la selección adecuada de eventos sismo-volcánicos sintéticos estabiliza la exactitud del autoencoder.

Tabla 16

Resultados de Autoencoder señales naturales, sintéticas y señales naturales aumentados con señales sintéticas

Casos	Exactitud	Precisión	Sensibilidad	Especificidad	BER
Naturales	98.5%	99.0%	98.0%	99.0%	0.02
Sintéticas	96.5%	100%	93.5%	100%	0.04
Real 80%+Sintéticas 20%	98.0%	99.0%	97.1%	99.0%	0.02
Real 60%+Sintéticas 40%	98.0%	100%	96.2%	100%	0.02
Real 40%+Sintéticas 60%	97.0%	100%	94.3%	100%	0.03
Real 20%+Sintéticas 80%	96.5%	100%	93.5%	100%	0.04

Nota. En la tabla de presenta la Exactitud, Precisión, Sensibilidad, Especificidad, BER, de la herramienta Autoencoder con señales sismo-volcánicas LP y VT naturales y sintéticas.

Capítulo V

Conclusiones y Trabajos Futuros

Conclusiones

Para el presente trabajo de titulación, se utilizó la base de datos MicSigV1 que fue proporcionada por el IGEPN, esta base de datos contiene un número total de 1187 registros de los cuales existen 1044 LP, 101 VT, 27 REG, 8 HB y 7 ICE de las cuales solamente se trabajaron con LP y VT ya que basado en la teoría de la sísmica, su incremento desmesurado en un sistema volcánico denota una posible erupción de este. Por ello, se preprocesaron solo los eventos LP y VT para lo cual: se realizó un remuestreo de las señales a 100 Hz, normalización de la amplitud al pico máximo de cada señal, obtención de la respuesta en frecuencia a través de la FFT, normalización de las magnitudes a través del método *min-max*, rellenado de las muestras con AWGN a un total de 8192 muestras y así se logra tener datos preparados y listos que forman parte del conjunto de base de datos que se pueda entrenar el modelo CGAN.

En la fase de diseño del modelo CGAN, se implementó un modelo generador capaz de aprender a crear respuestas en frecuencia realistas y un modelo discriminador que aprende a distinguir respuestas en frecuencia falsas. Se configuro para los dos modelos el tamaño de las capas, *kernel, stride*, la función de activación, cantidad de filtros. Se logra obtener un modelo CGAN definitivo el cual permitió obtener respuestas en frecuencia con las características de eventos sismo volcánicos de tipo LP y VT. La diversificación de los parámetros y características de los modelos durante el entrenamiento fue crucial para encontrar un modelo definitivo a través de la monitorización del proceso de entrenamiento de distintos modelos.

Durante la fase de entrenamiento en la monitorización de distintos modelos se validó que, al utilizar una mayor cantidad de filtros y capas CGAN, provocaba fallos de

entrenamiento propios de los modelos antagónicos adversarios. Por otro lado, se encontró que una menor cantidad de filtros, capas provocaba que las señales sintéticas resultantes salieran muy similares entre sí, tanto en los eventos LP como en los VT, por lo que el modelo general no encontraba tantas características de las señales de donde aprender. Pese a tener limitados datos con señales de una naturaleza compleja y haber obtenido distintos modelos fallidos, a través de la variación de los parámetros se identificó un balance en el aprendizaje al utilizar 1000 *epoch* y con 256 *minibatch*. Además, gracias a la utilización de GPU se obtuvo una aceleración y optimización del entrenamiento con un tiempo de treinta y un minutos.

Se diseñó una aplicación en el software MATLAB 2020 ®, la cual permite generar señales sintéticas de eventos sismo-volcánicos etiquetados. En ella se define la cantidad de señales a generar de las cuales la mitad son del tipo LP y la otra en tipo VT en una relación de 50/50. La base de datos se presenta en un formato $.mat$ que contiene: la etiqueta del evento, la frecuencia de muestreo, la procedencia de la señal sismo-volcánica y el vector de la señal temporal. La aplicación permite además agregar cierta cantidad de ruido AWGN en los extremos de los eventos sintéticos con la finalidad de facilitar el ingreso de las señales en detectores sismo-volcánicos ya que en los mismos debe existir una continuidad inicial y posterior del evento. Adicionalmente la aplicación tiene la función de graficar las señales de la base de datos creadas, en la misma se una gráfica del evento temporal y de la PSD de la señal.

En las herramientas de análisis se obtuvieron las siguientes exactitudes en la clasificación de eventos LP y VT: sistema de reconocimiento, con 90.4%; *autoencoder*, con 97.5% y clasificación visual, con 90.5%. La herramienta con el mejor resultado de exactitud es el *autoencoder*, por lo que se realizó una segunda prueba con la mezcla de cierta cantidad de señales sísmicas sintéticas y reales a través de la variación de sus cantidades, se logró una exactitud de entre un 96.5% y a un 98.5%. El modelo

propuesto proporciona una estructura eficaz para generar señales sismo-volcánicas sintéticas de alta calidad. Esta pequeña variación negativa de la exactitud al clasificar más señales sintéticas con el *autoencoder* se debe al tener un conjunto de datos con una naturaleza compleja y con pocos valores en el conjunto de datos de entrenamiento en eventos VT por lo que no se llegó a un aprendizaje totalmente profundo de las características, sin embargo, la exactitud es alta. En cuanto a eventos LP la variación fue casi nula en la exactitud y se concluye que fácilmente estas señales pueden pasar como si fuesen reales. Por lo tanto, se valida experimentalmente el modelo desarrollado como un enfoque prometedor para el aumento de datos de señales sismo-volcánicas de alta calidad.

Los resultados de las herramientas de clasificación y detección con valores de exactitud superiores al 90% y un BER inferior a 0.10 ratifican que el uso de una gran cantidad de señales sismo-volcánicas sintéticas etiquetadas que poseen características de eventos LP y VT sirven para clasificar y detectar eventos, lo que conlleva a alimentar las bases de datos actuales para estudios futuros basados en técnicas de *Deep Learning.*

En este proyecto de investigación se implementó un modelo CGAN que genera señales sismo-volcánicas sintéticas para dos tipos de eventos sísmicos: LP y VT. El estudio se basó en el método de generación de magnitudes de respuesta en frecuencia y que, en conjunto con las magnitudes de respuesta en fase de señales sismo-volcánicas reales, se obtuvieron nuevas y únicas señales sintéticas sismo-volcánicas de los dos tipos de eventos en el dominio del tiempo a través de la IFFT. Estas señales sintéticas fueron evaluadas por distintas herramientas de detección y clasificación de eventos sismo-volcánicos de tipo LP y VT donde se obtuvieron porcentajes de exactitud mayores al 90% y en precisión sobre el 99%.

Por otro lado, el entrenamiento de redes adversarias generativas con etiquetas CGAN supone una mejora significativa con respecto a la utilización de redes simples GAN. Un único modelo es el que discrimina y genera eventos LP y VT, optimiza recursos y tiempo durante el entrenamiento. La GAN elaborada por un grupo de investigación interno posee una exactitud del 46.4% en la clasificación visual entre señales reales y sintéticas frente a la CGAN elaborada en este trabajo de investigación que posee una exactitud del 48.21%. La CGAN beneficia al uso de conjuntos de datos limitados, como en esta investigación al tener poco más de 100 señales VT, ayudándole al modelo a discriminar de forma óptima durante el entrenamiento.

Trabajos Futuros

En las herramientas utilizadas para la clasificación de eventos sismo-volcánicos LP y VT, se tienen altos los valores de exactitud y precisión mayores de 90% en los dos casos, al tener una base de datos pequeña de 1145 señales para el entrenamiento, el modelo CGAN tuvo dificultades a la hora de conseguir todas las características de la respuesta en frecuencia en las señales de tipo VT. Por tal motivo, para mejorar los resultados, se recomienda entrenar al modelo CGAN no solo con la base de datos proporcionada por el IGEPN, sino con el uso de bases de datos de señales sintéticas como la que se presenta en B. Perez (2022). Con ello tener una base de datos con un mayor número señales sismo-volcánicas entre naturales y sintéticas, con ello entrenar la CGAN por más tiempo lo que conlleva a que el modelo sea capaz de extraer mayor número de características de las señales sismo-volcánicas VT y LP.

Se recomienda analizar el uso de las nuevas herramientas que posee el *Deep Learning Toolbox* de MATLAB ® en su versión más actual del año 2022. El entrenamiento de redes generativas de datos unidimensionales se presenta en esta

versión en el que se evita procesamientos internos donde se optimiza del código y de tiempos de entrenamiento de los modelos.

Nuestro grupo de trabajo está interesado en que se generen señales sintéticas de eventos LP, VT, HYB y ICEQUAKE con nuevos métodos de generación de elementos con redes generativas adversarias como son: Auxiliary Classifier Generative Adversarial Network, Progressive Growing Generative Adversarial Network, Big Generative Adversarial Network, Information Maximizing Generative Adversarial Network. Estos modelos poseen gran cantidad de características como explorar en mejorar las funciones de perdida además del estudio del espacio latente. Adicionalmente, con el modelo CGAN diseñado en este trabajo de investigación, se puede llegar a entrenar con bases de datos de diferentes volcanes, como el volcán Chileno Llaima, se debe considerar el preprocesamiento de las señales antes del ingreso al modelo, igualmente se varió el tiempo de entrenamiento de los parámetros de Adam. Por otro lado, se podría llegar a utilizar diferentes métodos de Deep Learning para la generación de fases sintéticas de eventos sismo-volcánicas de tipo VT y LP, ya que la base de datos del volcán Cotopaxi liberada al público es limitada.

Referencias

Altamirano, S. (2021). *Sistema de reconocimiento de microterremotos en tiempo real del volcán Cotopaxi aplicando aprendizaje supervisado* [Tesis de Pregrado, Universidad de las Fuerzas Armadas ESPE]. http://repositorio.espe.edu.ec/jspui/handle/21000/23743

Bean, C. J., De Barros, L., Lokmer, I., Métaxian, J.-P., O' Brien, G., & Murphy, S. (2014). Long-period seismicity in the shallow volcanic edifice formed from slow-rupture earthquakes. *Nature Geoscience*, *7*(1), 71–75. https://doi.org/10.1038/ngeo2027

Chouet, B. A. (1996). Long-period volcano seismicity: Its source and use in eruption forecasting. In *Nature* (Vol. 380, Issue 6572, pp. 309–316). Macmillan Magazines Ltd. https://doi.org/10.1038/380309a0

Creswell, A., White, T., Dumoulin, V., Arulkumaran, K., Sengupta, B., & Bharath, A. A. (2017). Generative Adversarial Networks: An Overview. *IEEE Signal Processing Magazine*, *35*(1), 53–65. https://doi.org/10.1109/msp.2017.2765202

Curilem, G., Vergara, J., Fuentealba, G., Acuña, G., & Chacón, M. (2009). Classification of seismic signals at Villarrica volcano (Chile) using neural networks and genetic algorithms. *Journal of Volcanology and Geothermal Research*, *180*(1), 1–8. https://doi.org/10.1016/j.jvolgeores.2008.12.002

Demir, F. (2022). 14 - Deep autoencoder-based automated brain tumor detection from MRI data. In V. Bajaj & G. R. Sinha (Eds.), *Artificial Intelligence-Based Brain-Computer Interface* (pp. 317–351). Academic Press. https://doi.org/https://doi.org/10.1016/B978-0-323-91197-9.00013-8

Donahue, C., McAuley, J., & Puckette, M. (2018). Adversarial Audio Synthesis. *ArXiv*. http://arxiv.org/abs/1802.04208

Dumoulin, V., Visin, F., & Box, G. E. P. (2018). *A guide to convolution arithmetic for deep*

learning.

Fournier D'albe, E. M. (1979). Objectives of volcanic monitoring and prediction. *Journal of the Geological Society*, *136*(3), 321–326. https://doi.org/10.1144/gsjgs.136.3.0321

Ghosh, A., Sufian, A., Sultana, F., Chakrabarti, A., & De, D. (2020). *Fundamental Concepts of Convolutional Neural Network* (pp. 519–567). https://doi.org/10.1007/978-3-030-32644-9_36

Goodfellow, I. (2017). *NIPS 2016 Tutorial: Generative Adversarial Networks*.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). *Generative Adversarial Nets*.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2020). Generative adversarial networks. *Communications of the ACM*, *63*(11), 139–144. https://doi.org/10.1145/3422622

Grijalva, F., Ramos, W., Pérez, N., Benítez, D., Lara-Cueva, R., & Ruiz, M. (2020). ESeismic-GAN: A Generative Model for Seismic Events from Cotopaxi Volcano. *Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, *En Revisió*.

Gupta, T. (2017). *Deep Learning: Feedforward Neural Network.* https://towardsdatascience.com/deep-learning-feedforward-neural-network-26a6705dbdc7

Ham, H., Jun, T. J., & Kim, D. (2020). *Unbalanced GANs: Pre-training the Generator of Generative Adversarial Network using Variational Autoencoder*. https://arxiv.org/abs/2002.02112v1

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction* (2nd ed.). Springer Publishing.

Huang, H., Yu, P. S., & Wang, C. (2018). *An Introduction to Image Synthesis with*

Generative Adversarial Nets. 17.

Hubel, D. H., & Wiesel, T. N. (1968). Receptive fields and functional architecture of monkey striate cortex. *The Journal of Physiology*, *195*(1), 215–243. https://doi.org/10.1113/jphysiol.1968.sp008455

IGEPN. (2020). *Volcán Cotopaxi - Instituto Geofísico - EPN*. https://www.igepn.edu.ec/

Ioffe, S., & Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *32nd International Conference on Machine Learning, ICML 2015*, *1*, 448–456. https://arxiv.org/abs/1502.03167v3

Karras, T., Aittala, M., Hellsten, J., Laine, S., Lehtinen, J., & Aila, T. (2020). *Training Generative Adversarial Networks with Limited Data*. arXiv. https://doi.org/10.48550/ARXIV.2006.06676

Kingma, D. P., & Ba, J. (2014). Adam: A Method for Stochastic Optimization. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*. https://arxiv.org/abs/1412.6980v9

Lara-Cueva, R. A., Benitez, D. S., Carrera, E. V., Ruiz, M., & Rojo-Alvarez, J. L. (2016). Automatic Recognition of Long Period Events from Volcano Tectonic Earthquakes at Cotopaxi Volcano. *IEEE Transactions on Geoscience and Remote Sensing*, *54*(9), 5247–5257. https://doi.org/10.1109/TGRS.2016.2559440

Lara-Cueva, R. A., Benítez, D. S., Carrera, E. V., Ruiz, M., & Rojo-Álvarez, J. L. (2016). Feature selection of seismic waveforms for long period event detection at Cotopaxi Volcano. *Journal of Volcanology and Geothermal Research*, *316*, 34–49. https://doi.org/10.1016/j.jvolgeores.2016.02.022

Lara-Cueva, R., Carrera, E. V., Morejon, J. F., & Benitez, D. (2016). Comparative analysis of automated classifiers applied to volcano event identification. *2016 IEEE Colombian Conference on Communications and Computing, COLCOM 2016 - Conference Proceedings*, 1–6. https://doi.org/10.1109/ColComCon.2016.7516377

Lara Mina, M. F. (2021). *Sistema de Reconocimiento Automático de Micro sismos Volcánicos basado en Redes Neuronales Convolucionales*. http://repositorio.espe.edu.ec/bitstream/21000/26461/1/T-ESPE-046542.pdf

Li, F.-F., Krishna, R., & Xu, D. (2021). *CS231n Convolutional Neural Networks for Visual Recognition*. Stanford Spring Course 2021. https://cs231n.github.io/neural-networks-2/

Li, Z., Meier, M. A., Hauksson, E., Zhan, Z., & Andrews, J. (2018). Machine Learning Seismic Wave Discrimination: Application to Earthquake Early Warning. *Geophysical Research Letters*, *45*(10), 4773–4779. https://doi.org/10.1029/2018GL077870

Maas, A. L., Hannun, A. Y., & Ng, A. Y. (2013). *Rectifier Nonlinearities Improve Neural Network Acoustic Models*.

Malfante, M., Dalla Mura, M., Métaxian, J.-P., Mars, J., Macedo, O., Inza, A., & DALLA MURA Member IEEE, M. (2018). *Machine Learning for Volcano-Seismic Signals: Challenges and Perspectives MACHINE LEARNING FOR VOLCANO-SEISMIC SIGNALS: CHALLENGES AND PERSPECTIVES 1 Machine Learning for Volcano-seismic Signals: Challenges and Perspectives*. *35*(2), 20–30. https://doi.org/10.1109/MSP.2017.2779166ï

MATLAB. (2020). *9.8.0.1323502 (R2020a)*. The MathWorks Inc.

McNutt, S. R. (1996). Seismic Monitoring and Eruption Forecasting of Volcanoes: A Review of the State-of-the-Art and Case Histories. In *Monitoring and Mitigation of Volcano Hazards* (pp. 99–146). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-80087-0_3

Mirza, M., & Osindero, S. (2014). Conditional Generative Adversarial Nets. *ArXiv*. https://arxiv.org/abs/1411.1784

Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill Education. https://books.google.com.ec/books?id=xOGAngEACAAJ

Murphy, K. (2012). *Machine learning : a probabilistic perspective* (1st ed.). MIT Press.

Nair, V., & Hinton, G. E. (2010). Rectified Linear Units Improve Restricted Boltzmann Machines. *Proceedings of the 27th International Conference on International Conference on Machine Learning*, 807–814.

Nugroho, A., & Winarko, E. (2011). Geographical information system and Web Service implementation for volcanic eruptions geologic disasters surveillance, monitoring, and mitigation system. *Proceedings of the 2011 International Conference on Electrical Engineering and Informatics, ICEEI 2011*. https://doi.org/10.1109/ICEEI.2011.6021766

Nusinovici, S., Tham, Y. C., Chak Yan, M. Y., Wei Ting, D. S., Li, J., Sabanayagam, C., Wong, T. Y., & Cheng, C. Y. (2020). Logistic regression was as good as machine learning for predicting major chronic diseases. *Journal of Clinical Epidemiology*, *122*, 56–69. https://doi.org/10.1016/j.jclinepi.2020.03.002

Parsons, S. (2005). Introduction to Machine Learning by Ethem Alpaydin. *The Knowledge Engineering Review*, *20*(4), 432–433. https://doi.org/10.1017/s0269888906220745

Perez, B. (2022). Generación de señales sintéticas de eventos sismo-volcánicos del volcán Cotopaxi mediante el método de Bootstrap. In *Repositorio ESPE*. UNIVERSIDAD DE LAS FUERZAS ARMADAS ESPE.

Pérez, N., Benítez, D., Grijalva, F., Lara-Cueva, R., Ruiz, M., & Aguilar, J. (2020). ESeismic: Towards an Ecuadorian volcano seismic repository. *Journal of Volcanology and Geothermal Research*, *396*, 106855. https://doi.org/10.1016/J.JVOLGEORES.2020.106855

Perez, N., Venegas, P., Benitez, D., Lara-Cueva, R., & Ruiz, M. (2020). A New volcanic seismic signal descriptor and its application to a data set from the cotopaxi volcano. *IEEE Transactions on Geoscience and Remote Sensing*, *58*(9), 6493–6503. https://doi.org/10.1109/TGRS.2020.2976896

Radford, A., Metz, L., & Chintala, S. (2016). Unsupervised representation learning with deep convolutional generative adversarial networks. *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*, 1–16.

Robinson, R. (2017). *Convolutional Neural Networks - Basics*. https://mlnotebook.github.io/post/CNN1/

Rodgers, M., Smith, P., Pyle, D., & Mather, T. (2016). Waveform classification and statistical analysis of seismic precursors to the July 2008 Vulcanian Eruption of Soufrière Hills Volcano, Montserrat. *EGUGA*, EPSC2016-8336. https://ui.adsabs.harvard.edu/abs/2016EGUGA..18.8336R/abstract

Rosenblatt, F. (1957). *The Perceptron, a Perceiving and Recognizing Automaton Project Para*. Cornell Aeronautical Laboratory. https://books.google.com.ec/books?id=P%5C_XGPgAACAAJ

Russell, S. (2021). *Artificial intelligence : a modern approach* (4th ed.). Pearson.

Samuel, A. L. (1959). Some Studies in Machine Learning Using the Game of Checkers. *IBM Journal of Research and Development*, *3*(3), 210–229. https://doi.org/10.1147/rd.33.0210

Secretaría Nacional de Gestión de Riesgos, X. (2015). *Plan de Contingencia Interinstitucional de la Secretaría de Gestión de Riesgo para enfrentar amenaza de erupción del volcán Cotopaxi*.

Sigurdsson, H., Houghton, B., McNutt, S., Rymer, H., & Stix, J. (2015). The Encyclopedia of Volcanoes. In A. Press (Ed.), *The Encyclopedia of Volcanoes* (Segunda Ed). Elsevier. https://doi.org/10.1016/c2015-0-00175-7

Soofi, A., & Awan, A. (2017). Classification Techniques in Machine Learning: Applications and Issues. *Journal of Basic & Applied Sciences*, *13*, 459–465. https://doi.org/10.6000/1927-5129.2017.13.76

Sultana, F., Sufian, A., & Dutta, P. (2018). Advancements in Image Classification using

Convolutional Neural Network. *2018 Fourth International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN)*. https://doi.org/10.1109/icrcicn.2018.8718718

Toulkeridis, T. (2013). *Volcanes Activos Ecuador | Request PDF* (T. Toulkeridis (ed.); 1st ed.). Imprenta y Offset Santa Rita.

Tripathi, A., Singh, A. K., Singh, K. K., Choudhary, P., & Vashist, P. C. (2021). Chapter 1 - Machine learning architecture and framework. In K. K. Singh, M. Elhoseny, A. Singh, & A. A. Elngar (Eds.), *Machine Learning and the Internet of Medical Things in Healthcare* (pp. 1–22). Academic Press. https://doi.org/https://doi.org/10.1016/B978-0-12-821229-5.00005-7

Wang, T., Trugman, D., & Lin, Y. (2020). *SeismoGen: Seismic Waveform Synthesis Using Generative Adversarial Networks*.

Yang, S., Xie, L., Chen, X., Lou, X., Zhu, X., Huang, D., & Li, H. (2017). Statistical parametric speech synthesis using generative adversarial networks under a multi-task learning framework. *2017 IEEE Automatic Speech Recognition and Understanding Workshop, ASRU 2017 - Proceedings*, *2018-Janua*, 685–691. https://doi.org/10.1109/ASRU.2017.8269003

Yang, X. S. (2019). *Introduction to Algorithms for Data Mining and Machine Learning*. Elsevier Science \& Technology. https://books.google.com.ec/books?id=QtKdDwAAQBAJ

Yu, S. (2018). *Adversarial Learning for Image-to-Image Generative Creativity Simiao Yu October 2018* (Issue October).

Zeiler, M. D., & Fergus, R. (2014). Visualizing and understanding convolutional networks. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *8689 LNCS*(PART 1), 818–833. https://doi.org/10.1007/978-3-319-10590-1_53