Implementación de un sistema clasificador de microterremotos del volcán Cotopaxi basado en técnicas de Deep Learning

Iglesias Giler, Iván Alejandro y Rosero Arciniega, Luis Alejandro

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.

10 de julio de 2023

Informe de originalidad

NOMBRE DEL CURSO
QuasiGraduados

NOMBRE DEL ALUMNO
IVAN ALEJANDRO IGLESIAS GILER

NOMBRE DEL ARCHIVO
IVAN ALEJANDRO IGLESIAS GILER - Tesis Revisión Originalidad

SE HA CREADO EL INFORME
10 jul 2023

Resumen

Fragmentos marcados	4	0,3 %
Fragmentos citados o entrecomillados	1	0,1 %

Coincidencias de la Web

mathworks.com	2	0,2 %
alertamundialinfo.com	1	0,1 %
researchgate.net	1	0,1 %
keytrends.ai	1	0,1 %

1 de 5 fragmentos

Fragmento del alumno MARCADO

El Cotopaxi, situado **en la Cordillera Oriental**, se ubica **a 35 km al noreste de Latacunga ya 45 km al sureste de Quito**. Reconocido como uno de los volcanes más peligrosos del...

Mejor coincidencia en la Web

El volcán **Cotopaxi** se ubica **en la Cordillera Oriental** (Real), **a** una distancia de **35 km al noreste de Latacunga y a 45 km al sureste de Quito**.

Actividad del volcán Cotopaxi pone en alerta a las autoridades de ... https://alertamundialinfo.com/2023/02/28/actividad-del-volcan-cotopaxi-pone-en-alerta-a-las-autoridades-de-ecuador/

2 de 5 fragmentos

Fragmento del alumno MARCADO

Un autoencoder es un tipo de red neuronal que se utiliza para el aprendizaje de características y **la** reducción de **la** dimensionalidad. El autoencoder consta de dos componentes principales: un...

Mejor coincidencia en la Web

Departamento de Eléctrica, Electrónica y Telecomunicaciones

Carrera de Ingeniería en Electrónica y Telecomunicaciones

Certificación

Certifico que el trabajo de titulación: **"Implementación de un sistema clasificador de microterremotos del volcán Cotopaxi basado en técnicas de Deep Learning"** fue realizado por los señores **Iglesias Giler, Iván Alejandro y Rosero Arciniega, Luis Alejandro;** el mismo que cumple con los requisitos legales, teóricos, 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, 10 de julio de 2023

ROMAN ALCIDES LARA CUEVA

Ing. Román Alcides Lara Cueva, PhD.

C.C: 1713988218

ESPE
UNIVERSIDAD DE LAS FUERZAS ARMADAS
INNOVACIÓN PARA LA EXCELENCIA

Departamento de Eléctrica, Electrónica y Telecomunicaciones

Carrera de Ingeniería en Electrónica y Telecomunicaciones

Responsabilidad de Autoría

Nosotros, **Iglesias Giler, Iván Alejandro y Rosero Arciniega, Luis Alejandro,** con cédulas de ciudadanía n° 1724086838 y 1726056409, declaramos que el contenido, ideas y criterios del trabajo de titulación: **Implementación de un sistema clasificador de microterremotos del volcán Cotopaxi basado en técnicas de Deep Learning** 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, 10 de julio de 2023

Iglesias Giler, Iván Alejandro

C.C: 1724086838

Rosero Arciniega, Luis Alejandro

C.C: 1726056409

ESPE
UNIVERSIDAD DE LAS FUERZAS ARMADAS
INNOVACIÓN PARA LA EXCELENCIA

Departamento de Eléctrica, Electrónica y Telecomunicaciones

Carrera de Ingeniería en Electrónica y Telecomunicaciones

Autorización de Publicación

Nosotros, **Iglesias Giler, Iván Alejandro y Rosero Arciniega, Luis Alejandro,** con cédulas de ciudadanía n° 1724086838 y 1726056409, autorizamos a la Universidad de las Fuerzas Armadas – ESPE publicar el trabajo de titulación: **Implementación de un sistema clasificador de microterremotos del volcán Cotopaxi basado en técnicas de Deep Learning** en el Repositorio Institucional, cuyo contenido, ideas y criterios son de nuestra responsabilidad.

Sangolquí, 10 de julio de 2023

Iglesias Giler, Iván Alejandro

C.C: 1724086838

Rosero Arciniega, Luis Alejandro

C.C: 1726056409

Dedicatoria

El presente trabajo de titulación está dedicado a mi abuelita, Judith Puebla, quien en vida me dio la motivación, cariño, comprensión y confianza para persistir en mis estudios y proyectos personales, que siempre tuvo las palabras justas y necesarias que me ayudaron a superar las adversidades que se presentaron en este arduo camino. De igual manera a mis padres, Iván y Janneth, quienes me apoyaron de manera emocional y económica de principio a fin durante mi carrera universitaria. Así como a mis hermanos, Sebastián y Mateo, a quienes con mi ejemplo espero inspirar y motivar en su día a día.

Iván Alejandro Iglesias Giler

El presente trabajo de titulación le dedico a Dios, por nunca dejarme solo y estar siempre a mi lado en todo momento. De igual manera a mis padres, Luis y Gloria, quienes me han dado el apoyo durante toda mi vida y nunca dudaron de mí, y a mis hermanos, Allison y Mateo, que siempre confiaron en mí y estuvieron a mi lado. Este logro es por ustedes.

Luis Alejandro Rosero Arciniega

Agradecimiento

En primer lugar, agradezco a Dios por entregarme la sabiduría, fortaleza, pero sobre todo salud para poder afrontar y superar todos los retos que se presentaron.

A mi madre Janneth, quien con su amor me enseñó la responsabilidad para conseguir lo que anhelo, el respeto y consideración para con los demás, pero sobre todo me forjó y me hizo la persona quien soy. Por estar siempre pendiente de mí bienestar y felicidad, por quererme como nadie más lo hará y motivarme siempre a superarme como persona y profesional.

A mi padre Iván, quien supo enseñarme que el conocimiento y el trabajo duro nos permiten llegar tan lejos como nos lo propongamos. Por ayudarme en momentos de duda y encontrar motivación en la necesidad.

A mi abuelita Judith, quien con su sabiduría supo guiar mi senda. Por escucharme y motivarme siempre.

A mi tutor Ing. Román Alcides Lara Cueva, PhD., por habernos brindado su tiempo, paciencia y confianza para la realización del presente trabajo de titulación.

Al Departamento de Eléctrica, Electrónica y Telecomunicaciones, el cual, mediante sus docentes y colaboradores, me brindó la oportunidad de crecer a nivel personal y profesional.

A mi camarada Alejandro Rosero, por permitirme compartir con él conocimientos y experiencias, por brindarme su amistad y apoyo en todo momento. Fue un gusto realizar contigo el presente trabajo de titulación.

A mis amigos y compañeros, con quienes superamos dificultades, pero sobre todo forjamos lazos de amistad, ustedes también son parte de este logro.

Iván Alejandro Iglesias Giler

Agradecimiento

Ante todo, doy gracias a Dios por permitirme vivir, por darme una madre, un padre, una hermana, un hermano, familia y amigos. Gracias Dios por toda la humildad, sabiduría, fortaleza, paciencia, convicción, para seguir adelante cada día de mi vida.

A mis padres, Luis y Gloria, quienes me han demostrado que el amor de padres es lo más puro y verdadero, que en la vida nada es fácil pero tampoco imposible. Por estar siempre conmigo apoyándome en cada paso que he dado, por sus sabios consejos, ánimos para seguir adelante y no rendirme hasta lograr el objetivo.

A mis hermanos, Allison y Mateo, quienes han sido mi motivo para no rendirme y ser un ejemplo para ellos, por su cariño y compañía que me dieron durante toda mi vida.

A mi gran amigo Iván Iglesias, que lo considero como un hermano más, agradezco por confiar en mí y desarrollar este presente trabajo.

A mis amigos que me dejó la vida universitaria, muchas gracias por todas las experiencias vividas dentro y fuera de las aulas, hicieron que este proceso sea un poco más agradable y llevadero.

Luis Alejandro Rosero Arciniega

Índice de Contenido

Índice de Tablas

Índice de Figuras

Resumen

La integración de las tecnologías de inteligencia artificial (AI, del inglés *Artificial Intelligence*) y aprendizaje profundo (DL, del inglés *Deep Learning*) tiene una importancia significativa en el monitoreo de riesgos para la seguridad pública de Ecuador. Con los riesgos potenciales asociados a las erupciones volcánicas, la adopción de estas tecnologías tiene el potencial de revolucionar los sistemas de monitoreo y permiten el modelado predictivo para la evaluación efectiva del comportamiento volcánico. El presente estudio se centra en la clasificación de microsismos volcánicos de Largo Período (LP, del inglés *Long Period*) y Volcano Tectónico (VT, del inglés *Volcano Tectonic*) como indicadores de actividad volcánica utilizando algoritmos DL: Autoencoder Apilado (SA, del inglés *Stacked Autoencoder*) y Red Neuronal Profunda (DNN, del inglés *Deep Neural Network*), mediante aprendizaje supervisado. El análisis incorpora métricas de rendimiento como la exactitud, la precisión, la sensibilidad, la especificidad y la tasa de error balanceada (BER, del inglés *Balanced Error Rate*).

El estudio utiliza para el entrenamiento y las pruebas una base de datos original, una base de datos sintética y una base de datos mixta que combina datos originales y sintéticos. Además, los microsismos volcánicos se enriquecen al agregar los coeficientes de dos transformadas Wavelet discretas, Daubechies y Symlets, lo que da lugar a otras dos bases de datos mixtas, con un total de cinco conjuntos de datos. La investigación evalúa los algoritmos SA y DNN en la clasificación de microsismos volcánicos, teniendo en cuenta la posible influencia de las transformadas Wavelet. El estudio revela resultados superiores en la clasificación de microsismos volcánicos LP y VT del Cotopaxi a través de pruebas realizadas con los algoritmos SA y DNN, que consiguen mejorar el objetivo del IGEPN de obtener un valor de BER de 0.01.

Palabras clave: clasificación, Cotopaxi, aprendizaje profundo, aprendizaje supervisado, microsismos volcánicos.

Abstract

The integration of Artificial Intelligence (AI) and Deep Learning (DL) technologies is of significant importance in risk monitoring for public safety in Ecuador. With the potential risks associated with volcanic eruptions, the adoption of these technologies has the potential to revolutionize monitoring systems and enable predictive modeling for effective assessment of volcanic behavior. The present study focuses on the classification of Long Period (LP) and Volcano Tectonic (VT) volcanic micro-earthquakes as indicators of volcanic activity using Stacked Autoencoder (SA) and Deep Neural Network (DNN) DL algorithms through supervised learning. The analysis incorporates performance metrics such as accuracy, precision, sensitivity, specificity and Balanced Error Rate (BER).

The study uses for training and testing an original database, a synthetic database and a mixed database combining original and synthetic data. In addition, the volcanic microseisms are enriched by adding the coefficients of two discrete Wavelet transforms, Daubechies and Symlets, resulting in two other mixed databases, with a total of five datasets. The research evaluates the SA and DNN algorithms in volcanic microseismic classification, taking into account the possible influence of Wavelet transforms. The study reveals superior results in the classification of Cotopaxi LP and VT volcanic microearthquakes through tests performed with the SA and DNN algorithms, which manage to improve the IGEPN's goal of obtaining a BER value of 0.01.

Keywords: classification, Cotopaxi, deep learning, supervised learning, volcanic microseisms.

Capítulo I

Introducción

La vulcanología, el estudio científico de los volcanes y la actividad volcánica, es de gran importancia para la sociedad debido a su papel en la comprensión y mitigación de los riesgos volcánicos. Los vulcanólogos pueden reconocer e interpretar microsismos volcánicos relacionados con el movimiento del magma, las emisiones de gases y la deformación del terreno (Sparks, 1997), mediante el análisis de señales y datos procedentes de diversos instrumentos de vigilancia, como sismómetros y sensores de gases. Los microsismos volcánicos se procesan mediante técnicas de análisis, para extraer información valiosa sobre la actividad volcánica. La aplicación del análisis de señales en vulcanología permite a los investigadores caracterizar los procesos volcánicos, prever erupciones y alertar a tiempo a las comunidades en riesgo (Ibáñez & Carmona, 2000).

Durante los últimos años el campo de la Inteligencia Artificial (AI, del inglés *Artificial Intelligence*) ha tenido una importante evolución, y ha obtenido logros sin precedentes con el desarrollo de herramientas que permiten generar contenido de lenguaje natural, imágenes, música e incluso videos mediante modelos computacionales y matemáticos que utilizan la predicción y clasificación como eje principal. En este contexto, los científicos de varias áreas de investigación se encuentran con la necesidad de incorporar estos nuevos desarrollos a sus proyectos para facilitar la realización de tareas repetitivas o que pueden sesgarse por el factor humano.

En el campo de las telecomunicaciones, se pueden utilizar modelos de Aprendizaje Automático (ML, del inglés *Machine Learning*) para el procesamiento y análisis de señales eléctricas o electrónicas de un determinado sistema. En los trabajos de (Carrera et al., 2020; R. Lara et al., 2020; R. Lara-Cueva et al., 2020), se han utilizado los modelos antes mencionados

para el procesamiento de señales producidas por microsismos volcánicos, donde se obtuvieron buenos resultados para el reconocimiento de microsismos volcánicos obtenidos del Cotopaxi. Sin embargo, en últimas investigaciones realizadas con la utilización de modelos de Aprendizaje Profundo (DL, del inglés *Deep Learning*), que requieren una mayor cantidad de datos para su entrenamiento, se obtuvieron mejores resultados en el reconocimiento de microsismos volcánicos (F. Lara et al., 2021), esto explora su clasificación como trabajos futuros. Es por ello que se han desarrollado y experimentado con diversos algoritmos de DL que puedan adaptarse a aplicaciones de análisis de señales, debido a que con estos modelos se mejoran de manera exponencial los resultados en tareas de clasificación.

En el ámbito del grupo de investigación universitaria, se llevaron a cabo diversos trabajos relacionados con el análisis de microsismos volcánicos utilizando técnicas de ML tradicional. Estos enfoques abarcaron desde la detección de microsismos volcánicos hasta una temprana clasificación, también aprovechan las propiedades distintivas de los microsismos volcánicos. Estos enfoques demostraron su eficacia al proporcionar una detección confiable y ayudar en la comprensión de la actividad volcánica. Sin embargo, en la búsqueda de mejorar aún más los resultados y explorar nuevas posibilidades, también se han realizado investigaciones que emplean técnicas de DL, las cuales se caracterizan por su capacidad de procesar grandes volúmenes de datos y capturar patrones complejos en los microsismos volcánicos. Estos avances prometedores en el campo del análisis de microsismos volcánicos respaldan la necesidad de continuar con la exploración y comparación de diferentes enfoques de DL para mejorar la precisión y eficiencia en la clasificación de los microsismos volcánicos.

En este contexto, el propósito del presente trabajo es desarrollar algoritmos y técnicas de DL para clasificar microsismos volcánicos del Cotopaxi con el objetivo de en un futuro aportar a la creación de un sistema de alerta temprana ante una posible erupción del volcán, de

tal manera que se logre con ello mitigar las pérdidas humanas y materiales que conllevaría dicho evento, para lo cual se requiere entrenar dos modelos de DL, basados en las técnicas de una Red Neuronal Profunda (DNN, del inglés *Deep Neural Network*) y el Autoencoder Apilado (SA, del inglés *Stacked Autoencoder*).

Antecedentes

Ecuador es un país que se encuentra atravesado por la cordillera de los Andes, debido a ello posee varios volcanes cuya actividad sísmica es supervisada debido a que una próxima erupción es un hecho inevitable (Mothes, 1998). Cada volcán tiene un comportamiento particular, el volcán Cotopaxi, por ejemplo, presenta los siguientes microsismos volcánicos: Largo Período (LP, del inglés *Long Period*), Tremor (TRE, del inglés *Tremor*), Híbrido (HB, del inglés *Hybrid*), *icequake* (ICE, del inglés *Ice Quake*) y Volcano Tectónico (VT, del inglés *Volcano Tectonic*) (R. Lara-Cueva et al., 2015). El monitoreo de los microsismos volcánicos del Cotopaxi, principalmente de LP y VT, pueden proporcionar pistas de su comportamiento con el objetivo de reconocer el aumento de actividad volcánica que podría preceder a una erupción y minimizar sus consecuencias (Chouet et al., 1994) (Chouet, 1996). VT son microsismos volcánicos que tienen lugar en un entorno volcánico con una duración de tiempo variable que suele ser inferior a 30 s y un contenido espectral amplio que suele estar por encima de 5 Hz, mientras que los microsismos volcánicos LP, también conocidos como eventos de baja frecuencia, muestran una falta de fases distinguibles y una duración de tiempo típica inferior a 90 s, con contenido espectral limitado a bandas de frecuencia estrechas entre 2 y 5 Hz (Ibáñez & Carmona, 2000). En este contexto, el uso de sistemas de reconocimiento automático basados en los clasificadores de aprendizaje automático pueden ayudar a los expertos a mejorar la labor de detección y clasificación de los microsismos volcánicos, que requiere mucho tiempo (R. Lara-Cueva et al., 2020). Los modelos basados en el DL han superado a los enfoques clásicos basados en ML en varias tareas de clasificación (Minaee et al., 2022). Una

DNN puede analizar la correlación entre las características de alta dimensión de los datos, que consta de una capa de entrada, una capa de salida y varias capas ocultas. Las capas adyacentes están conectadas unas con otras de forma completa (Ma et al., 2019). Un *autoencoder* es un tipo de red neuronal que se utiliza para aprender una representación comprimida de los datos de entrada, que luego puede utilizarse para reconstruir los datos originales con una pérdida mínima de información (Ju et al., 2015). El modelo de SA es un algoritmo de aprendizaje de características que utiliza dos o más *autoencoders* como bloques de construcción para crear una estructura entrenamiento de redes profundas más compleja (Lv et al., 2014).

Justificación e Importancia

El Cotopaxi, situado en la Cordillera Oriental, se ubica a 35 km al noreste de Latacunga ya 45 km al sureste de Quito. Reconocido como uno de los volcanes más peligrosos del mundo, se gana esta reputación debido a sus frecuentes erupciones, estilo eruptivo único, terreno accidentado, cubierta de hielo y la gran cantidad de poblaciones potencialmente afectadas. El peligro que representa el Cotopaxi se deriva de la posibilidad de que sus erupciones desencadenen inmensos lahares (flujos de lodo y escombros). El peligro potencial que representan estos lahares es inmenso, ya que fluirían por los valles cercanos, poniendo en riesgo las áreas densamente pobladas. Comprende el Valle Interandino entre Mulaló y Latacunga, así como un tramo del Valle de los Chillos. Se ha aproximado que más de 300.000 individuos habitan actualmente regiones que son susceptibles a lahares en caso de erupciones volcánicas repetidas. Las repercusiones de una erupción del Cotopaxi se extienden más allá de las inmediaciones, ya que una porción sustancial de las regiones de la Sierra y la Costa de Ecuador podría verse afectada por la caída de ceniza generalizada. Es por ello que el Instituto Geofísico de la Escuela Politécnica Nacional (IGEPN) realiza un monitoreo constante del volcán Cotopaxi las 24 horas del día y 365 días del año mediante una red de sismógrafos

instalada en el volcán, al analizar los sucesos que presenta, principalmente de los microsismos volcánicos LP y VT. Actualmente los expertos del IGEPN realizan una clasificación y etiquetado visual de todos los microsismos detectados y son almacenados en una base de datos, ocupación que requiere de mucho tiempo y paciencia, en caso de que exista un aumento de actividad sísmica en el volcán Cotopaxi, puede llevar a los técnicos a realizar clasificaciones con errores, debidos a la presión por la gran cantidad de microsismos volcánicos surgidos. El IGEPN requiere de un sistema que le permita la clasificación de los microsismos volcánicos en menor tiempo de lo que actualmente tardan los profesionales, que además mitigue el error sistemático cometido, es por ello que se evidencia la importancia de contar con un sistema clasificador de microsismos volcánicos automatizado.

Con el surgimiento de algoritmos mejorados en el campo de ML para varios sectores de la vida científica y profesional a nivel global, el sector de las telecomunicaciones es uno de los más beneficiados especialmente en el área del análisis de señales. Las señales sísmicas presentan una oportunidad en el desarrollo de sistemas automatizados para la detección de desastres naturales, poder así mejorar la prevención y seguridad ante situaciones emergentes de los ciudadanos en países como Ecuador, donde se encuentra el volcán Cotopaxi. La mayor presencia de microsismos volcánicos LP y VT pueden proporcionar pistas del comportamiento del volcán con el objetivo de reconocer el aumento de la actividad volcánica que podría preceder a una erupción y con ello lanzar una alerta temprana a la población y minimizar las consecuencias en caso de desastre.

Debido a que la precisión requerida para un sistema clasificador debe ser muy elevada por todo lo que conlleva el desarrollo de un sistema de alerta temprana, la mejor estrategia es utilizar algoritmos para redes neuronales convolucionales de DL, técnicas que requieren una gran cantidad de datos para su correcto funcionamiento y aplicación. Las bases de datos que

posee el IGEPN de microsismos volcánicos del Cotopaxi no contienen suficientes datos por lo que es necesaria la generación de una base de datos sintética para el aprendizaje de los algoritmos antes mencionados.

En base a todos los hechos mencionados y considerando que en Ecuador es necesario un sistema que permita reconocer a tiempo las señales que prevengan una posible erupción del volcán Cotopaxi, se quiere implementar un sistema clasificador de microsismos volcánicos mediante técnicas de DL de elevada confiablidad. Lo que se propone es comparar la precisión en la clasificación de microsismos volcánicos LP y VT de dos algoritmos de DL: DNN y SA, para determinar cuál optimiza de mejor manera el uso en la clasificación de microsismos volcánicos del Cotopaxi, y finalmente probar si se puede extrapolar a otros volcanes.

Alcance

El presente proyecto de investigación desarrolló un aplicativo que realiza la clasificación de los microsismos volcánicos. El alcance del presente proyecto de investigación es estudiar la efectividad de dos modelos de DL: SA y DNN, para la clasificación de microsismos volcánicos etiquetados como LP o VT del Cotopaxi. Los microsismos volcánicos originales se encontraban en el dominio temporal, por lo que fueron procesados mediante la transformada rápida de *Fourier* (FFT, del inglés *Fast Fourier Transform*), con la estimación de la densidad espectral de potencia de *Welch* para convertirlos al dominio de la frecuencia, resultando en señales con 257 muestras, y luego fueron normalizadas entre 0 y 1 en su amplitud. Las señales se dividieron en un conjunto de entrenamiento que comprende el 70.00% de los datos y un conjunto de prueba que comprende el 30.00% de los datos. También se procesaron los microsismos volcánicos originales en el dominio del tiempo con las transformadas *Wavelet* de *Daubechies* y *Symlets*, y sus resultantes se trataron con la FFT para pasarlas al dominio frecuencial, se normalizan sus amplitudes entre 0 y 1, y fueron sumadas a las señales originales procesadas, por lo que

finalmente como resultado se obtuvieron señales con 514 muestras (por un lado 257 muestras FFT más 257 muestras FFT de *Daubechies*, y por otro 257 muestras FFT más 257 muestras FFT de *Symlets*), y fueron nuevamente divididas en un conjunto de entrenamiento y otro de prueba.

Los modelos de SA y DNN se entrenaron por medio de utilizar cinco tipos diferentes de bases de datos de microsismos volcánicos del Cotopaxi procesados como se indicó. La primera comprende únicamente microsismos reales, obtenidos directamente de los sensores en el volcán, que contiene 1044 señales LP y 101 VT, la segunda tiene microsismos sintéticos, generados mediante una Red Adversaria Generativa Condicional (CGAN, del inglés *Conditional Generative Adversarial Network*) del trabajo de investigación previo de (Galarza Cruz & Vega Vergara, 2022), de la que se obtuvieron 5000 microsismos volcánicos LP y 5000 del tipo VT. La tercera contiene una mezcla aleatoria de reales y sintéticas, para tener finalmente 5000 del tipo LP y 5000 microsismos volcánicos VT. La cuarta y quinta bases de datos siguieron el procedimiento de la base de datos mixta para los microsismos volcánicos procesados con la transformada *Wavelet* de *Daubechies* y *Symlets* respectivamente.

Por último, se desarrolló una aplicación en MATLAB® al utilizar la herramienta *AppDesigner*, que muestra los mejores modelos en función de los resultados obtenidos tras el entrenamiento con las cinco bases de datos, se muestra en una pestaña el mejor resultado para los microsismos volcánicos procesados originales, y en otra pestaña aparece la solución óptima para los que fueron tratados además con *Wavelet*. En cada pestaña se muestra el Gráfico de la Matriz de Confusión y las métricas de rendimiento: exactitud, precisión, especificidad, sensibilidad y la tasa de error equilibrada (BER, del inglés *Balanced Error Rate*) para el mejor modelo de SA y de DNN en cada pestaña. La manera de validar los modelos es probándolos con los microsismos volcánicos para prueba del Cotopaxi y algunos del volcán

Llaima. A través de este proyecto, se busca proporcionar información valiosa sobre la eficacia de los modelos de DL para la clasificación de microsismos volcánicos de los tipos LP y VT, además de ayudar a una mejor comprensión del comportamiento del volcán Cotopaxi.

Objetivos

Objetivo General

- Implementar un sistema clasificador de microterremotos del volcán Cotopaxi mediante técnicas de *Deep Learning* (DL).

Objetivos Específicos

- Identificar las bases de datos de Eventos Sismovolcánicos (ESV) reales y sintéticos generada mediante *Data Augmentation* (DA) disponibles para trabajar con técnicas de DL.
- Desarrollar algoritmos basados en DL para clasificar eventos Largo Período (LP) y Volcano Tectónico (VT) a través de *Autoencoder* y *Deep Neural Network* (DNN).
- Evaluar el desempeño de los clasificadores mediante el análisis de las métricas de rendimiento con base de datos real, sintética y mixta.
- Comparar los resultados entre la clasificación de *Autoencoder* y DNN.

Trabajos relacionados

La clasificación de microsismos volcánicos es de gran relevancia para la ciencia, por lo que se han desarrollado algunos trabajos en donde utilizan varios métodos para realizar dicha tarea, el objetivo de los estudios realizados es mejorar el rendimiento de las métricas para los sistemas de clasificación.

En el estudio de (Jaramillo et al., 2014) se propuso un método basado en estimadores espectrales clásicos y de máxima entropía que permiten la caracterización de microsismos volcánicos, se basa en el análisis espectral autorregresivo con el cual definen las

características espectrales de los microsismos volcánicos, como resultado este método permite detectar los microsismos volcánicos que se producen en tiempo real.

Dentro de las investigaciones desarrolladas en (R. A. Lara-Cueva, Benítez, et al., 2016) se presentó un sistema de detección de los microsismos volcánicos del Cotopaxi, para la extracción de sus características utilizaron dos técnicas, el filtro y la incrustación, aplicando algoritmos de clasificación como *k-Nearest Neighbours* (k-NN) y *Decision Trees* (DT). La implementación de una ventana de segmentación de 15 s, una matriz F con 50 características y G con 25 datos dieron los mejores resultados, arrojando un 99.00% de precisión y sensibilidad en la etapa de detección.

Un sistema automático en tiempo real de detección y clasificación de microsismos volcánicos se presentó en (R. A. Lara-Cueva, Benitez, et al., 2016), para la etapa de detección utilizaron ventanas no solapadas con duración de 15 segundos y en la etapa de clasificación de los microsismos volcánicos tuvieron una duración de 1 minuto, cada señal detectada de un microsismo volcánico fue sometida a la extracción de características temporales, estadísticas, espectrales y de dominio de escala, para identificar los microsismos volcánicos LP y VT, donde se utilizaron dos técnicas de selección de características: *wrapper* y *embedder*, las cuales fueron usadas dentro de los algoritmos de clasificación: máquina de vectores de apoyo (SVM, del inglés *Support Vector Machine*) y DT. El mejor resultado con 99.00% de precisión en la fase de detección y 97.00% en la fase de clasificación con respecto a la precisión y sensibilidad se logró con el clasificador SVM.

En el artículo de (R. Lara-Cueva, Paillacho-Salazar, et al., 2017) se desarrolló un sistema de detección basado en clasificación supervisada de los microsismos volcánicos y no volcánicos, poseen varias características las cuales fueron extraídas por medio de la FFT y *Wavelet*, para la clasificación de microsismos volcánicos utilizaron el algoritmo de DT al

emplear técnicas embebidas, como resultado obtuvieron una exactitud del 99.00% y 98.00% en precisión.

Otro trabajo relacionado se presenta en (R. Lara-Cueva, Benitez, et al., 2017), el cual es un sistema automático basado en algoritmos de aprendizaje automático para el reconocimiento de microsismos volcánicos y no volcánicos como relámpagos y ruido de fondo (BN, del inglés *Back Noise*). En la etapa de detección utilizan el algoritmo de DT y para la etapa de clasificación usan SVM en su modalidad de multiclase. Teniendo como resultado en la etapa de detección un 98.00% de precisión en la identificación de eventos BN y en la etapa de clasificación un 90.00% de precisión, en donde el kernel lineal fue el parámetro óptimo para maximizar el rendimiento de la clasificación.

En el estudio de (Venegas et al., 2019) se presentó un algoritmo integral basado en 5 conocidos clasificadores de ML que presentan el mejor rendimiento, el algoritmo desarrollado se enfoca en clasificar microsismos volcánicos LP y VT. La metodología utilizada es la combinación de modelos de ML para determinar cuál es el más indicado en la tarea de clasificación de microsismos volcánicos LP y VT. Los resultados mostraron que el clasificador de bosque aleatorio con 100 predictores basados en árboles fue el mejor para la clasificación.

En el trabajo de (Duque et al., 2020) se analizó el uso de seis métodos diferentes basados en *clustering* con la finalidad de clasificar microsismos volcánicos LP y VT, y también para encontrar señales superpuestas no volcánicas. Las pruebas realizadas muestran que el método BIRCH con k=2 es el mejor modelo para la clasificación de los eventos puros LP y VT con una puntuación de precisión de 81.00% y 88.00%. El método de agrupamiento espectral con k=3 también mostro buenos resultados, donde fue capaz de clasificar los microsismos con y sin presencia de señales superpuestas, que da como resultado una puntuación de precisión equilibrada de 50.00%.

Un estudio sobre las principales bandas de frecuencias que permiten maximizar las métricas de rendimiento de un clasificador automatizado para microsismos volcánicos VT y LP se presentó en (R. Lara-Cueva et al., 2020), para la elección de las características se usaron métodos envolventes e incrustados, entre los que se seleccionaron las principales frecuencias que maximizan el BER de los algoritmos de clasificación. Se usó *bootstrapping* para estimar un intervalo de confianza para las frecuencias de los microsismos y una diferencia de umbral con amplitud de 3 dB con la finalidad de que no se solapen las frecuencias características de los microsismos volcánicos LP y VT. El clasificador DT muestra un total de 10 frecuencias clave y el clasificador SVM con 39 frecuencias agrupadas en tres bandas de frecuencias principales, para distinguir los microsismos volcánicos LP de los VT. Se obtuvo los mejores resultados con el método *Welch* en DT y con los clasificadores SVM con el método *Multitapper*.

En (F. Lara et al., 2021) se creó un sistema de identificación de microsismos volcánicos del Cotopaxi mediante técnicas de DL. Las etapas de detección y clasificación emplearon redes neuronales convolucionales y espectrogramas. Las redes neuronales fueron entrenadas usando una base de datos de 1187 microsismos, utilizando aprendizaje por transferencia. Los hallazgos demuestran que el sistema logró una precisión del 99,00% en la etapa de detección y del 97,00% en la etapa de clasificación.

Un nuevo enfoque basado en una combinación adecuada de técnicas de morfología matemática y criterios de similitud para clasificar microsismos volcánicos VT y LP se mostró en (Perez et al., 2022), en donde se explora el dominio del microsismo volcánico con el fin de obtener un nuevo espacio de características, el cual está basado en el mapa de bordes del patrón de microsismos volcánicos. Los resultados determinaron que en el rendimiento de clasificación la norma *L2* fue estadísticamente superior en microsismos volcánicos con señales

superpuestas y similar con aquellos sin señales superpuestas, en donde se obtuvo puntuaciones medias de precisión de 93.34% y 96.88%.

Capítulo II

Marco Teórico

Inteligencia Artificial

La IA es un campo de la informática que se enfoca en la creación de programas y mecanismos que muestren comportamientos inteligentes similares a los seres humanos. Estos sistemas pueden analizar grandes cantidades de datos, identificar patrones y tendencias para formular predicciones de forma automática y precisa. Se define como la rama de la ciencia computacional que investiga y desarrolla programas que presuponen procesos mentales inteligentes (Porcelli, 2020).

El uso de modelos de aprendizaje computacional basados en las redes neuronales de la biología humana ha empoderado a la inteligencia artificial (IA) para crear sistemas inteligentes que agilizan la toma de decisiones a través del procesamiento rápido de datos. Este dominio abarca una amplia gama de áreas, incluido el reconocimiento de voz, la visión artificial, el procesamiento del lenguaje natural, la adquisición de conocimientos, la robótica avanzada y busca sistemas que posean la capacidad de percibir, razonar, participar y adquirir conocimientos (Márquez Jairo, 2020).

Aprendizaje Automático

El ML es una disciplina de la IA que utiliza algoritmos para identificar patrones, realizar predicciones, aprender de los datos y tomar decisiones. Para su estudio se utilizan algoritmos como DT, *clustering* basado en instancias, regresión, redes neuronales, bayesianos y redes adversarias generativas (*GAN*, del inglés *Generative Adversarial Networks*), que se ejecutan en la ciencia de datos y requieren diversos cálculos matemáticos debido a la complejidad de la información. La principal ventaja del ML es su capacidad para identificar patrones y tomar

decisiones en cuestión de semanas, con una reducción significativa de costos en comparación con otras técnicas estándar que pueden llevar años (Márquez Jairo, 2020).

Aprendizaje Profundo

El DL es una rama del ML que utiliza algoritmos correlacionales para clasificar datos. Se basa en redes neuronales y utiliza una jerarquización de la información para segmentar patrones categorizados por niveles, similar al proceso de aprendizaje humano. A medida que se escalan niveles más complejos de datos, se produce un aprendizaje progresivo (Márquez Jairo, 2020).

Figura 1

Modelo de relación entre DL, ML e IA

Aprendizaje Supervisado

El análisis predictivo suele emplear algoritmos de aprendizaje supervisado, que requieren la interacción humana para etiquetar los datos con precisión. Con el aprendizaje supervisado, normalmente se procesan conjuntos de datos de entrada y salida, además se enseña al modelo con ejemplos. El modelo aprende la relación entre los datos de entrada y de salida y puede hacer predicciones con nuevos conjuntos de datos. El objetivo es construir un modelo que haga predicciones basadas en pruebas en presencia de incertidumbre. Los

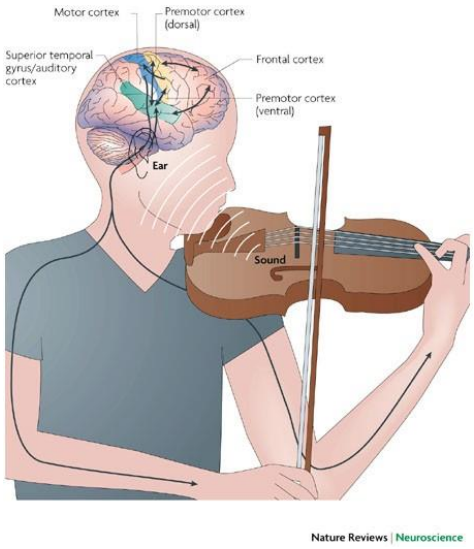
métodos de aprendizaje supervisado pueden tratar problemas de regresión y clasificación. En la regresión, se estima una relación matemática entre una o más variables y una variable continua. La relación puede entonces predecir la variable desconocida a partir de las conocidas. La regresión puede utilizarse para determinar la posición y la velocidad de un coche mediante *GPS*. La clasificación se utiliza cuando hay una variable discreta desconocida. Suele utilizarse para determinar qué muestra concreta pertenece a un conjunto predefinido de clases. Algunos ejemplos de clasificación son la detección de *spam* en el correo electrónico o la identificación de caras en una imagen (Uddin et al., 2019).

Entrenamiento de modelos de aprendizaje profundo

El DL es parte del ML, que a su vez forma parte del campo de la IA. Las redes neuronales son la base constitutiva de DL, dado que obtienen su inspiración en las neuronas humanas que para aprender una nueva habilidad realizan conexiones en el sistema nervioso donde antes no existían, por ejemplo, al aprender a tocar un instrumento musical se realizan conexiones entre las neuronas del cerebro con las células neuromusculares. Esta inspiración conllevo a la creación del perceptrón, que es el modelo matemático del funcionamiento de la neurona humana.

Figura 2

Ejemplo de las interacciones que tienen las neuronas al tocar un instrumento musical



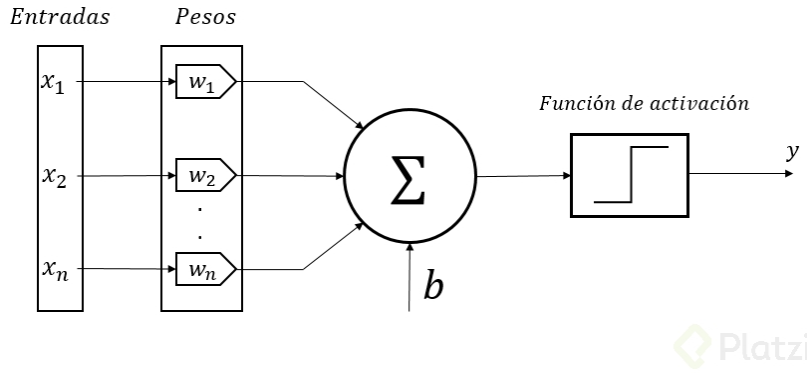
Nota. El gráfico muestra las interacciones auditivo-motoras en la percepción y producción musical. Tomado de (Zatorre et al., 2007).

El modelo matemático del perceptrón representa una sola neurona en el aprendizaje profundo, cada neurona del perceptrón tiene varias entradas, que son las etiquetas de identificación en el aprendizaje supervisado, y una salida con las cuales se va a comunicar con otras neuronas. Cada entrada se multiplica por un peso, que son las características que le quiero dar a mi modelo de neurona para que pueda aprender cierta habilidad. El proceso que lleva a cabo la neurona para el aprendizaje supervisado de forma matemática, para obtener el valor de la salida, es por tanto la suma de todas las entradas multiplicadas por cada uno de los pesos que finalmente llevan a una salida. Finalmente se suma un factor adicional que se conoce como *bias* o sesgo que va a resultar en que la salida resultante varíe de manera proporcional en porcentaje a ese valor, ajustando los resultados del modelo a lo que se desea obtener (Goodfellow et al., 2016).

Sin embargo, son necesarias muchas neuronas relacionadas en un algoritmo para un correcto aprendizaje, como ocurre en el cerebro humano, por lo que en la resolución de problemas es necesario juntar muchos perceptrones para la resolución de problemas específicos, y es allí cuando surge un problema. Dado que cada perceptrón se modela matemáticamente como una regresión lineal, al ser sumados en un modelo para su interacción dará como resultado que converjan en una sola regresión lineal, lo que no es óptimo dado que no es la interacción que requiere el aprendizaje. Para solucionar este problema existe la función de activación, que no es más que agregarle componentes de no linealidad al perceptrón, ello se logra pasando la regresión lineal por una función no lineal, lo que hará variar los resultados en la salida de cada perceptrón y permitirá que muchas neuronas interactúen entre sí, sin que converjan a una sola neurona (Goodfellow et al., 2016).

Figura 3

Modelo del perceptrón



Nota. Explicación del modelo del perceptrón. Tomado de (Espitia Corredor & Platzi, 2017).

Los modelos de DL aprenden en base a un proceso de entrenamiento de la red neuronal. El entrenamiento busca minimizar el valor de la función de pérdida en función de la variación de las características propias del modelo, es decir los pesos y el *bias*. Los ajustes se realizan en función de los resultados obtenidos por el modelo mediante un algoritmo de

optimización, en las redes neuronales de DL el que es utilizado por defecto se conoce como BP. La función de pérdida más utilizada en las redes neuronales de DL es conocida como Entropía Binaria Cruzada y funciona comparando las salidas del modelo, que se denotan como Clase Predicha, con las etiquetas proporcionadas por el supervisor del aprendizaje de la red, que se denominan como Clase Verdadera.

Aprendizaje basado en el Gradiente

Crear y entrenar una red neuronal de DL es similar a entrenar cualquier otro modelo de ML utilizando el descenso de gradiente. Sin embargo, las redes neuronales difieren de los modelos lineales en que su no linealidad hace que la mayoría de las funciones de pérdida interesantes no sean convexas. Como resultado, las redes neuronales se entrenan normalmente utilizando optimizadores iterativos basados en el gradiente, cuyo objetivo es minimizar la función de coste en lugar de resolver ecuaciones lineales o algoritmos de optimización convexa que garantizan la convergencia global (Goodfellow et al., 2016).

La convergencia de la optimización convexa está garantizada independientemente de los parámetros iniciales. Sin embargo, cuando se trata de funciones de pérdida no convexas y la aplicación del descenso de gradiente estocástico, la convergencia no está asegurada y depende en gran medida de los valores de los parámetros iniciales. En el caso de las redes neuronales de aprendizaje profundo, es de suma importancia inicializar todos los pesos con valores aleatorios pequeños. En cuanto al sesgo, puede inicializarse en cero o en pequeños valores positivos (Goodfellow et al., 2016).

Algoritmo de Retropropagación

Cuando se utiliza una red neuronal de DL para generar una salida $\hat{y}$ basada en una entrada x, el flujo de información dentro de la red avanza hacia delante, se origina en la capa de entrada y atraviesa las unidades ocultas en cada capa subsiguiente, hasta llegar a la capa

de salida. Este proceso se conoce como propagación directa. Durante el entrenamiento, la propagación directa continúa hasta que se produce un costo escalar $J(\theta)$. El algoritmo de retropropagación (BP, del inglés *Backpropagation*) en ese sentido habilitar el flujo de información de costos a la inversa a través de la red es crucial para el cálculo del gradiente (Goodfellow et al., 2016).

Aunque calcular una expresión analítica para el gradiente es sencillo, evaluarlo numéricamente puede ser costoso desde el punto de vista computacional. El algoritmo de BP proporciona un procedimiento económico y sencillo para calcular el gradiente. Es importante tener en cuenta que la propagación hacia atrás se refiere solo al método para calcular el gradiente y no al algoritmo de aprendizaje completo para redes neuronales multicapa. De hecho, se utiliza un algoritmo diferente, como el descenso de gradiente estocástico, para realizar el aprendizaje utilizando este gradiente (Goodfellow et al., 2016).

Es importante comprender la distinción entre propagación directa, propagación inversa y el algoritmo de aprendizaje cuando se trabaja con redes neuronales. El algoritmo de BP permite el cálculo eficiente del gradiente durante el entrenamiento, pero es solo un componente del algoritmo de aprendizaje general (Goodfellow et al., 2016).

Autoencoder Apilado

Un *autoencoder* es un tipo de red neuronal que se utiliza para el aprendizaje de características y la reducción de la dimensionalidad. El *autoencoder* consta de dos componentes principales: un codificador y decodificador. El codificador utiliza los datos de entrada y los comprime en una representación de dimensiones inferiores, conocida como codificación o representación de características. A continuación, el decodificador recibe la codificación y la utiliza para reconstruir los datos lo más fielmente posible a los originales (Goodfellow et al., 2016). Durante el entrenamiento, el *autoencoder* minimiza el error de

reconstrucción entre los datos originales y los reconstruidos, este proceso obliga al *autoencoder* a aprender una representación compacta de los datos que capte las características más importantes, mientras que se descarta, al mismo tiempo, el ruido o la información redundante. La codificación aprendida puede utilizarse entonces como representación de características para otros modelos de ML (Yang et al., 2017).

Entonces el SA es conocido como una red neuronal que consta de dos o más capas de *autoencoders* apiladas juntas, unas sobre otras o una detrás de otras. La primera capa toma los datos de entrada y aprende una representación comprimida de los mismos. Las capas siguientes toman la representación comprimida de la capa anterior y aprenden una representación aún más comprimida. La última capa del modelo reconstruye los datos de entrada originales a partir de la representación comprimida (Zhou et al., 2019). Los *autoencoders* apilados suelen utilizarse para el aprendizaje supervisado y pueden aprender características a partir de datos de entrada brutos etiquetados. Se han utilizado en diversas aplicaciones, como el reconocimiento de imágenes, el procesamiento del lenguaje natural y la detección de anomalías (Yu et al., 2022).

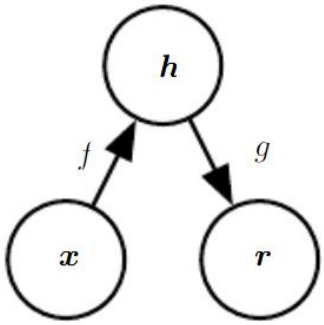
Entrenamiento del SA

Los *autoencoders* son un tipo de redes neuronales de DL, y su entrenamiento se consigue normalmente aplicando el descenso de gradiente mediante BP. Además de BP, los *autoencoders* también pueden entrenarse mediante recirculación, un algoritmo de aprendizaje que compara las activaciones de la red en la entrada original con las activaciones en la entrada reconstruida. Aunque se cree que la recirculación es biológicamente más plausible que la propagación hacia atrás, no se utiliza habitualmente en aplicaciones de DL. Consta de una función codificadora y una decodificadora que produce una reconstrucción, con una capa oculta que sirve de código para representar la entrada. Sin embargo, si la red aprende a copiar

perfectamente la entrada, no resulta especialmente útil. Por ello, los *autoencoders* suelen diseñarse para que prioricen qué aspectos de la entrada copiar, aprendiendo a menudo propiedades útiles de los datos en el proceso (Goodfellow et al., 2016). Esta arquitectura se muestra en la Figura 4.

Figura 4

Arquitectura de un autoencoder



Nota. La estructura general de un *autoencoder*, que asigna una entrada x a una salida (llamada reconstrucción) r a través de una representación interna o código h. El *autoencoder* tiene dos componentes: el codificador f (que asigna x a h) y el decodificador g (que asigna h a r). Tomado de (Goodfellow et al., 2016).

Para evitar la copia perfecta, los *autoencoders* suelen tener restricciones que les permiten copiar sólo aproximadamente, y sólo la entrada que se parece a los datos de entrenamiento, lo que puede resultar especialmente útil en aplicaciones de generación y clasificación (Goodfellow et al., 2016).

Red Neuronal Profunda

La arquitectura de una DNN se compone de múltiples capas de bloques de construcción elementales, estas capas pueden ser de intercalación no lineal o subredes más complejas. Cada capa utiliza la salida de la capa anterior como entrada, para aprender gradualmente una

representación jerárquica. El aprendizaje de estas representaciones jerárquicas se logra mediante el algoritmo de BP, que ajusta los parámetros de la red para minimizar el error de la salida esperada con respecto a la salida real. Las DNN han logrado avances significativos en diferentes áreas, como el procesamiento de señales de imagen, vídeo, habla y audio, permitiendo la extracción de características intrincadas y abstractas que antes eran difíciles de obtener (Low et al., 2020).

Se pueden identificar tres tipos principales de redes neuronales profundas: *Multilayer Perceptrons* (MLP), *Convolutional Neural Networks* (CNN) y *Recurrent Neural Networks* (RNN). MLP es la forma más básica de DNN, donde cada neurona está conectada a las neuronas de las capas vecinas, y se utiliza para clasificaciones sencillas. CNN incluye capas convolucionales que pueden extraer características visuales, mientras que RNN tiene conexiones de retroalimentación y puede procesar datos secuenciales. Las CNN se utilizan para el reconocimiento y procesamiento visual, mientras que las RNN pueden reconocer o generar datos secuenciales. La combinación de CNN y RNN permite el reconocimiento de entidades visuales variables en el tiempo, como el reconocimiento de gestos y acciones y el subtitulado de imágenes (Shin & Yoo, 2020).

Entrenamiento de DNN

Las redes neuronales profundas son un tipo de redes neuronales *feedforward* que se utilizan habitualmente en el aprendizaje profundo. Estos modelos pretenden aproximar una función f^*, por ejemplo, un clasificador que mapea una entrada x a una categoría y. Una red feedforward define un mapeo $y = f(x;\theta)$ y aprende el valor de los parámetros θ que conducen a la mejor aproximación de la función. El nombre "*feedforward*" proviene del hecho de que la información fluye desde la entrada x a través de los cálculos intermedios utilizados para definir f y, finalmente, a la salida y. No hay conexiones de retroalimentación en las que las salidas del

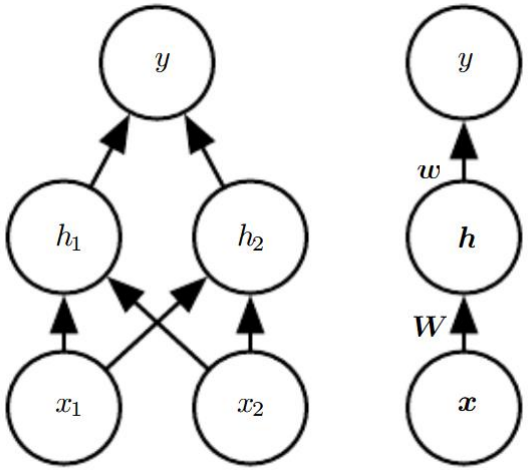
modelo se retroalimenten a sí mismas. Por ejemplo, una cadena de tres funciones $f^{(1)}$, $f^{(2)}$ y $f^{(3)}$ podría formar $f(x) = f^{(3)}\left(f^{(2)}\left(f^{(1)}(x)\right)\right)$. Aquí, $f^{(1)}$ se denomina primera capa, $f^{(2)}$ segunda capa, y así sucesivamente. La longitud total de la cadena da la profundidad del modelo. La capa de salida es la última capa de una red *feedforward* (Goodfellow et al., 2016).

Durante el entrenamiento de la red neuronal, $f(x)$ se hace coincidir con $f^*(x)$ utilizando ejemplos aproximados y ruidosos de $f^*(x)$ evaluados en diferentes puntos de entrenamiento. Cada ejemplo x va acompañado de una etiqueta $y \approx f^*(x)$. Los ejemplos de entrenamiento especifican directamente lo que debe hacer la capa de salida en cada punto x; debe producir un valor que se acerque a y. El comportamiento de las otras capas no está especificado explícitamente por los datos de entrenamiento. El algoritmo de aprendizaje debe decidir cómo utilizar esas capas para producir la salida deseada. Como los datos de entrenamiento no muestran la salida deseada para cada una de estas capas, se denominan capas ocultas. El algoritmo de aprendizaje debe decidir cómo utilizar estas capas para implementar de la mejor manera posible una aproximación de f^* (Goodfellow et al., 2016).

El término "profundo" deriva del hecho de que las redes neuronales tienen múltiples capas ocultas. Estas capas son importantes porque permiten a la red aprender representaciones cada vez más complejas de la entrada. Cuantas más capas tenga la red, más características abstractas podrá aprender. Sin embargo, el proceso de entrenamiento puede ser costoso desde el punto de vista computacional (Goodfellow et al., 2016).

Figura 5

Arquitecturas simples de redes neuronales profundas



Nota. Un ejemplo de red *feedforward*, dibujada en dos estilos diferentes. Ambos estilos tienen solamente una capa oculta. (Izquierda) En este estilo, cada unidad es un nodo del grafo. Este estilo es explícito e inequívoco, pero para redes más grandes que este ejemplo, puede consumir muchos recursos. (Derecha) En este estilo, existe un nodo en el gráfico para cada vector completo que representa las activaciones de una capa, mucho más compacto. Tomado de (Goodfellow et al., 2016).

Aprendizaje No Supervisado

El aprendizaje no supervisado es un método que tiene como objetivo descubrir patrones ocultos o estructuras inherentes dentro de conjuntos de datos, y es capaz de descubrir esos esquemas sin necesidad de intervención humana. Estos algoritmos pueden analizar datos no etiquetados y encontrar patrones e información relevantes. Los métodos de aprendizaje no supervisado se utilizan habitualmente en problemas que requieren agrupamiento y asociación. El *clustering* consiste en agrupar datos similares para analizar cada grupo y encontrar patrones. Por ejemplo, los algoritmos de agrupación pueden utilizarse para identificar grupos de clientes

en función de su historial de compras en línea y enviar anuncios específicos a cada grupo. Los métodos de asociación, por su parte, se utilizan para identificar artículos que se observan con frecuencia y que suelen comprarse juntos. Las tiendas en línea utilizan métodos de asociación para sugerir productos adicionales que pueden interesar a sus usuarios basándose en el contenido de los carritos de compra(Alloghani et al., 2020).

Aprendizaje por Refuerzo

El aprendizaje por refuerzo es un método de ensayo y error que proporciona información a las máquinas en función de sus acciones y experiencias. Este método también se conoce como aprendizaje por error y consiste en asignar valores positivos a los resultados deseables y valores negativos a los indeseables. El objetivo es encontrar soluciones óptimas, en las que el sistema aprenda a evitar los resultados negativos y a esforzarse por conseguir los positivos. El aprendizaje por refuerzo tiene aplicaciones prácticas en la creación de sistemas inteligentes para videojuegos, robótica y automatización industrial (Mills et al., 2022).

Data Augmentation

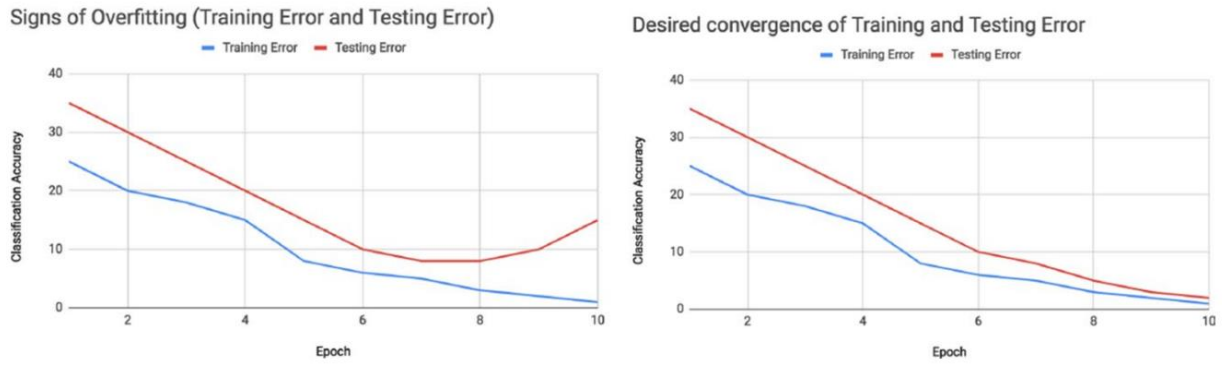
En los últimos años los algoritmos de DL han incursionado de manera exitosa en la generación de conjuntos de datos, en el paradigma del aprendizaje supervisado con mayor frecuencia, donde se han utilizado sobre todo GAN para la creación de nuevos conjuntos de datos con un parecido muy amplio a los originales. El éxito de los modelos DL durante la fase de entrenamiento depende en gran medida de la cantidad de datos a los que se puede acceder para el proceso, con ello se evita el sobreajuste, fenómeno que ocurre cuando un algoritmo para un modelo aprende una función con una elevada varianza, y como resultado modela sin errores los datos de entrenamiento. Por ello, se requiere incrementar los datos disponibles, cuando esa cantidad no es suficiente para garantizar buenos resultados del modelo (Shorten & Khoshgoftaar, 2019).

DA ha demostrado ser una técnica muy eficaz para el aumento de los datos disponibles, ya que ahorra tiempo y recursos necesarios para la recopilación de datos, y mejora el rendimiento general del modelo. Es crucial mitigar el problema del sobreajuste, ya que puede afectar negativamente a la capacidad de generalización de un modelo. Sobreajuste es un fenómeno que se caracteriza por la diferencia de rendimiento de un modelo cuando se evalúa con datos de entrenamiento frente a datos de prueba. Los modelos con poca capacidad de generalización tienen datos de entrenamiento sobre ajustados, lo que reduce su precisión y su rendimiento (Wong et al., 2016).

A manera de detectar el sobreajuste, se puede representar gráficamente la precisión del entrenamiento y la validación en cada época durante el entrenamiento. A continuación, en la Figura 6 se muestra la afectación del sobreajuste, que confirma que el aumento de las épocas de entrenamiento puede hacer que el modelo haga *overfitting* del conjunto de datos para entrenamiento y obtenga malos resultados en el conjunto de pruebas. Por lo tanto, es esencial encontrar un equilibrio entre el entrenamiento del modelo y la prevención del sobreajuste para lograr una mejor generalización y rendimiento general.

Figura 6

Verificación del sobreajuste entre los conjuntos de datos para entrenamiento y prueba



Nota. El gráfico de la izquierda ilustra un punto de inflexión en el que el error de validación empieza a aumentar a medida que disminuye la tasa de entrenamiento. Por el contrario, el gráfico de la derecha muestra un modelo bien entrenado con una relación deseable entre los errores de entrenamiento y de prueba. Tomado de (Shorten & Khoshgoftaar, 2019).

En la Figura 6, el gráfico de la izquierda indica que el modelo se ha ajustado en exceso a los datos de entrenamiento, lo que provoca un rendimiento deficiente cuando se prueba con el conjunto de validación. El gráfico de la derecha muestra en cambio que el modelo se ha entrenado de forma óptima y puede generalizar bien a los nuevos datos. Estos resultados ponen de relieve la importancia de evitar el sobreajuste y lograr un equilibrio entre el entrenamiento del modelo y la optimización de su rendimiento tanto en el conjunto de entrenamiento como en el de pruebas.

DL con MATLAB®

El software *MATLAB*® es una plataforma popular para aplicaciones de DL. Proporciona una interfaz fácil de usar para construir y entrenar varios modelos de DL, incluidos SA y DNN. La plataforma incluye varias características útiles, como funciones integradas, algoritmos y librerías, que permiten a los desarrolladores implementar técnicas de DL de forma eficaz. La *Deep Learning Toolbox* de *MATLAB*® es un conjunto completo de herramientas para construir y entrenar modelos de DL. Esta herramienta incluye modelos y arquitecturas pre entrenados, que pueden personalizarse fácilmente para aplicaciones específicas. Además, proporciona una serie de herramientas de visualización para supervisar y evaluar el rendimiento del modelo, lo que puede ayudar en el proceso de depuración y optimización (Mathworks, 2023).

En *MATLAB*®, crear un modelo de SA es muy sencillo. El primer paso es crear un *autoencoder* con una capa oculta y entrenarlo al hacer uso de los datos de entrada, a continuación, la salida de la capa del codificador se utiliza como entrada para otro *autoencoder*

con una capa oculta más pequeña y el proceso se repite hasta alcanzar el número de capas deseado (Mathworks, 2022a). Del mismo modo, construir una DNN en *MATLAB*® implica crear y configurar diferentes tipos de capas, como capas convolucionales, capas totalmente conectadas y capas de agrupación, y entrenar la red utilizando los datos de entrada y salida. El software proporciona funciones para visualizar la arquitectura de la red y supervisar el progreso del entrenamiento (Mathworks, 2022b).

Transformada Wavelet Discreta

Las transformadas de *Wavelet* son funciones matemáticas utilizadas para analizar y procesar señales, como las de audio, imagen y vídeo. Se utilizan para descomponer las señales en sus componentes de frecuencia, lo que revela detalles que no son visibles en las transformadas de *Fourier* tradicionales. Al utilizar la función $\psi(t)$ de la *wavelet* madre, es posible producir un conjunto de *wavelets*. El análisis *wavelet* se clasifica a grandes rasgos en dos tipos: transformada *wavelet* continua (CWT, del inglés *Continuous Wavelet Transform*) y transformada *wavelet* discreta (DWT, del inglés *Discrete Wavelet Transform*). (Lidong et al., 2015).

La técnica de CWT se enfrenta a un problema de redundancia de datos cuando maneja grandes cantidades de datos, mientras que el enfoque de DWT supera este problema al utilizar *wavelets* ortogonales. Por lo tanto, en este estudio se utilizaron las *wavelets* madre ortogonales de *Daubechies* y *Symlets*. El enfoque de la DWT también evita el solapamiento de las componentes y reduce la influencia del ruido, por lo que es el método más adecuado y más utilizado cuando se emplean técnicas de DL (D. Zhang, 2019).

La DWT utiliza la función *wavelet* y la función de escalado conocidas como filtro pasa alto y filtro pasa bajo, respectivamente. La DWT en forma matemática se define por (Hadi & Tombul, 2018):

$$\mathcal{W}f(j,k) = \int_{-\infty}^{+\infty} f(t)\psi_{j,k}^{*}(t)dt \quad (1)$$

$$\psi_{j,k}(t) = a_0^{-j/2}\psi\left(a_0^{-j}t - b_0 k\right) \quad (2)$$

donde a_0 y b_0 son las constantes, j es el nivel de descomposición y k es el factor de traslación temporal. La DWT frecuentemente utilizada es la DWT diádica que se obtiene al sintonizar los valores $a_0 = 2$ y $b_0 = 1$ y que dirige a la función de la Ecuación 2 (Daubechies, 1992):

$$\psi_{j,k}(t) = 2^{-j/2}\psi\left(2^{-j}t - k\right) \quad (3)$$

En primer lugar, al utilizar una DWT diádica, la señal pasa por los filtros, lo que da lugar a dos tipos de conjuntos de coeficientes: aproximación, producida por el filtro de paso bajo, y detalle, producido por el filtro de paso alto. En cada nivel subsiguiente, denominado nivel *i*, se producen un coeficiente de aproximación y un número *i* de coeficientes de detalle. En el presente estudio, hemos considerado el sexto nivel de descomposición como el nivel óptimo basados en el trabajo de (Saltos Torres, 2014).

Modelo Daubechies

Las *wavelets* de *Daubechies* son una familia de DWT que deben su nombre a la matemática belga Ingrid Daubechies. Son *wavelets* de soporte compacto con fase extrema y el mayor número de momentos de fuga para una anchura de soporte dada. Los filtros de escala asociados son filtros de fase mínima. Estas envolventes son un conjunto de funciones que pueden utilizarse para descomponer una señal en distintos componentes de frecuencia. Este tipo de señales tienen la ventaja de poder ofrecer una reconstrucción exacta de la señal original, a diferencia de otros tipos de *wavelets* (Daubechies, 1992).

Modelo Symlets

El modelo *Symlets* es una mejora de la aproximación de *Daubechies*, que aborda el inconveniente de la asimetría aproximada presente en la segunda. Son *wavelets* de soporte compacto con fase extrema y el mayor número de momentos de fuga para una anchura de

soporte dada. Los filtros de escala asociados son filtros de fase mínima. La base de la *wavelet Symlets* tiene mejor regularidad que la *Daubechies*, lo que puede reducir la distorsión de fase en el análisis y la reconstrucción de señales no estacionarias, como los microsismos volcánicos (Khushaba, 2021).

Densidad Espectral de Potencia

La densidad espectral de potencia (PSD, del inglés *Power Spectral Density*) representa la potencia contenida en una señal por unidad de frecuencia. Es una herramienta fundamental en el tratamiento y análisis de señales, ya que permite a los investigadores examinar el contenido frecuencial y la distribución de energía de una señal. En telecomunicaciones, la PSD se utiliza para evaluar la eficiencia y el rendimiento de los sistemas de comunicación. Al analizar la PSD de una señal, los investigadores pueden determinar la relación señal/ruido, identificar posibles fuentes de interferencia, así como optimizar el diseño y la asignación de canales de comunicación. La reciente investigación de (Ali et al., 2021) proporciona un análisis exhaustivo de la PSD en sistemas de telecomunicaciones, discute sus aplicaciones y su impacto en los numerosos esquemas de conversión digital que se han introducido en el campo de la comunicación digital.

En el análisis geofísico, la PSD se emplea para estudiar señales sísmicas y caracterizar el comportamiento del subsuelo terrestre. Mediante el análisis de la PSD de los datos sísmicos, los investigadores pueden identificar y localizar eventos, determinar la magnitud y el contenido de frecuencia de los movimientos telúricos y obtener información sobre las estructuras geológicas y las propiedades de la corteza terrestre. El trabajo de (Z. Zhang et al., 2021) se presenta un análisis detallado de la PSD, discute sus aplicaciones en la monitorización de terremotos, la detección de fallas y la obtención de imágenes del subsuelo. Los autores

demuestran cómo la PSD puede ayudar a comprender la dinámica de las ondas sísmicas y proporcionar información valiosa para la evaluación de peligros.

Métricas de Evaluación

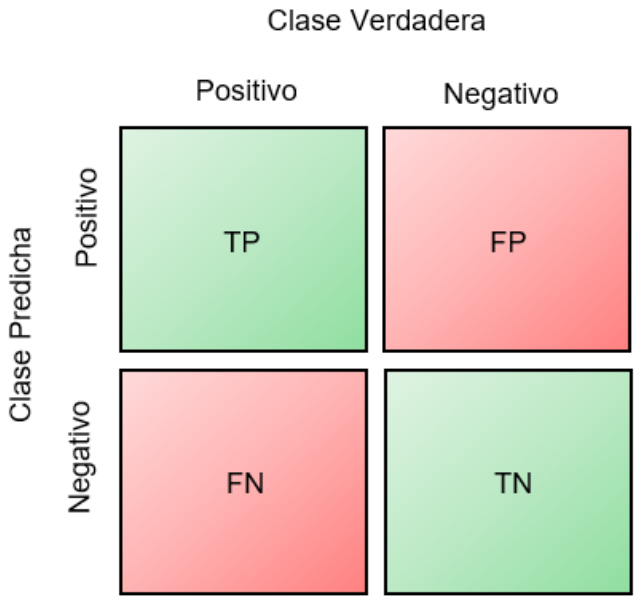
Tras la implementación de algoritmos de DL, la evaluación del modelo entrenado es esencial y, por lo tanto, las métricas de evaluación se vuelven imprescindibles. La matriz de confusión, la precisión, la sensibilidad, la especificidad y el BER son las métricas de evaluación utilizadas con más frecuencia en DL.

Matriz de Confusión

Se trata de una de las métricas más claras e ilustrativas, que compara la predicción del modelo con el resultado real. Se utiliza principalmente en algoritmos que generan dos o más clases como salida. La Figura 7 muestra cómo se construye una matriz de confusión con dos clases (Vakili et al., 2020).

Figura 7

Matriz de confusión, modelo con dos clases



Nota. Verdadero positivo (TP, del inglés *True Positive*), verdadero negativo (TN, del inglés *True Negative*), falso negativo (FN, del inglés *False Negative*), falso positivo (FP, del inglés *False positive*).

A partir de la matriz de confusión se extraen las siguientes métricas de evaluación:

Exactitud

Es una medida que muestra la proporción de puntos de datos clasificados correctamente sobre el número total de observaciones. Esencialmente, mide la capacidad de un modelo para hacer predicciones precisas para un conjunto determinado de datos de prueba.

$$Exactitud = \frac{TP + TN}{TP + TN + FP + FN}$$

Precisión

Es una medida que indica el número de casos TP de todas las predicciones correctas realizadas por el modelo. Esta métrica es especialmente valiosa cuando la preocupación se centra más en los FP que en los falsos negativos.

$$Precisión = \frac{TP}{TP + TN}$$

Especificidad

Se refiere a la capacidad de un modelo entrenado para predecir con exactitud el número de casos TN.

$$Especificidad = \frac{TN}{TN + FP}$$

Sensibilidad

Es una métrica que representa la capacidad de un modelo entrenado para predecir correctamente los casos TP.

$$Sensibilidad = \frac{TP}{TP + FN}$$

BER

La tasa de error equilibrada, mide el error promedio dentro de cada clase positiva.

$$BER = 1 - \left(\frac{Sensibilidad + Especificidad}{2}\right)$$

Capítulo III

Materiales y Métodos

Conjunto de datos para Clasificación

Los grandes conjuntos de datos son esenciales para entrenar modelos de DL, como señalan (Le Quy et al., 2022) en su estudio sobre el sesgo de los conjuntos de datos. Descubrieron que los conjuntos de datos más grandes conducen a modelos más robustos, particularmente cuando los datos son diversos. Un conjunto de datos diverso puede ayudar a prevenir el sesgo en el modelo al proporcionar suficientes ejemplos de diferentes subgrupos y variaciones del dominio del problema. Además, un conjunto de datos más amplio permite cubrir mejor el espacio de entrada, lo que mejora el rendimiento de la generalización.

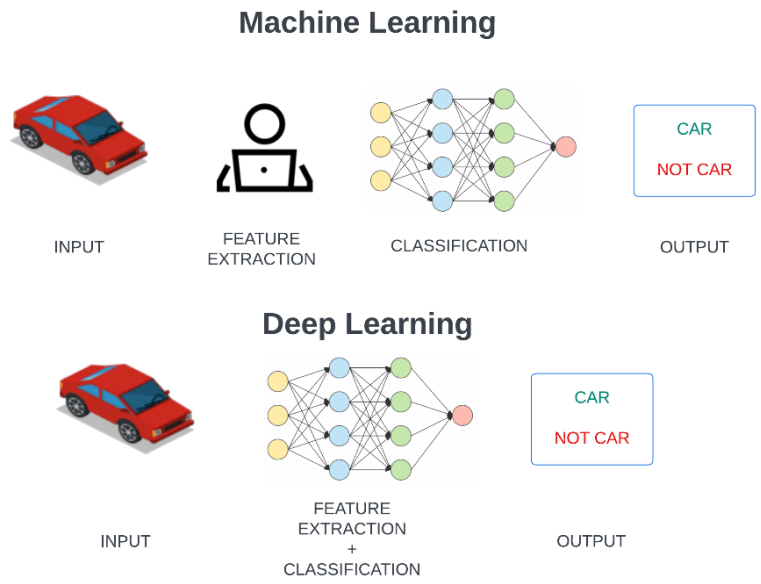
Al seleccionar un conjunto de datos para un problema de clasificación con DL, la elección de las etiquetas es crucial. Según (Su et al., 2012), un etiquetado de mala calidad puede afectar negativamente al rendimiento del modelo y conducir a una interpretación errónea de las predicciones del modelo. Por lo tanto, es esencial garantizar que el proceso de etiquetado sea coherente, preciso y representativo del dominio del problema. Es por ello que la elección de la base de datos proporcionada por el IGEPN es de suma importancia dado que ya se encuentran etiquetadas todos los microsismos volcánicos producidos por el Cotopaxi.

Los autores también destacan la importancia de las técnicas de DA, para incrementar el tamaño y la diversidad del conjunto de datos, sobre todo cuando los datos disponibles son limitados. Debe tener suficientes ejemplos que sean representativos de las diferentes clases, etiquetas consistentes y precisas, y reflejar la complejidad del problema, con ello, los modelos de DL pueden lograr una alta precisión en la clasificación del set de datos.

Comparativa entre ML y DL

Figura 8

Diferencias entre ML y DL



El ML, un subconjunto de la IA, se centra en algoritmos que permiten a las computadoras aprender de los datos y hacer predicciones o tomar decisiones sin programación explícita. El ML ofrece varias ventajas, como su versatilidad en varios dominios y la capacidad de manejar tareas complejas. Sin embargo, el DL, un subconjunto del ML, se ocupa específicamente de las redes neuronales artificiales inspiradas en la estructura y el funcionamiento del cerebro humano. El DL destaca en el procesamiento de grandes cantidades de datos y en la extracción automática de características jerárquicas, lo que da lugar a predicciones y clasificaciones tanto más precisas como sofisticadas.

La potencia de DL reside en su capacidad para aprender patrones y representaciones complejas, lo que puede resultar especialmente ventajoso en proyectos cruciales en los que la precisión y el rendimiento son primordiales. Con sus arquitecturas neuronales profundas y sus

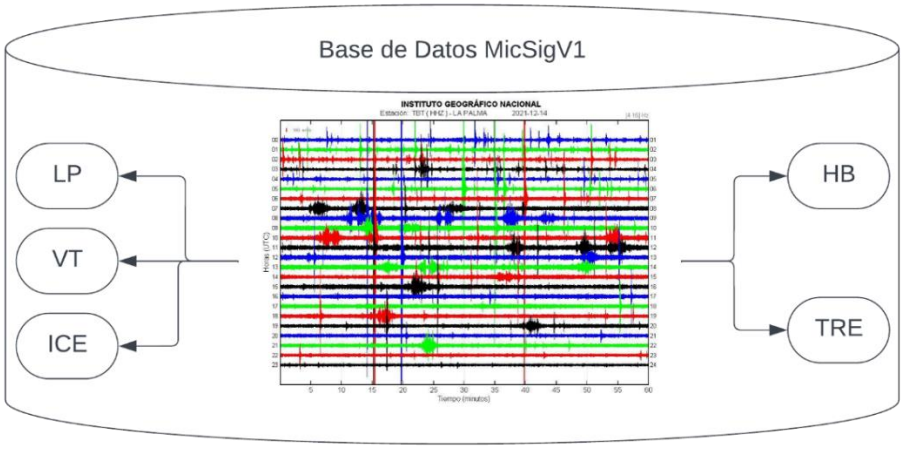
avanzadas capacidades de aprendizaje, el DL suele superar a los métodos tradicionales de ML, lo que lo convierte en la mejor opción para abordar tareas críticas y difíciles en campos como el reconocimiento y análisis de señales, el procesamiento del lenguaje natural y los sistemas autónomos.

Conjunto de datos de microsismos volcánicos

En el desarrollo del proyecto de investigación se utilizó la versión inicial del conjunto de datos de la base de datos denominada MicSigV1, que proporciona una serie de microsismos volcánicos en forma de señales discretas. Estas señales están representadas por matrices de puntos que contienen información relevante, como marcas de tiempo para el punto inicial y final, anotaciones del evento y su etiqueta de clase según la clasificación realizada entre los años 2012 y 2015 por los expertos del IGEPN (Pérez et al., 2020). El conjunto de datos MicSigV1, como se aprecia en la Figura 9, consta de un total de 1187 microsismos volcánicos, de los cuales 1044 corresponden a microsismos volcánicos LP, 101 son del tipo VT, 27 son TRE, 8 HB y 7 ICE.

Figura 9

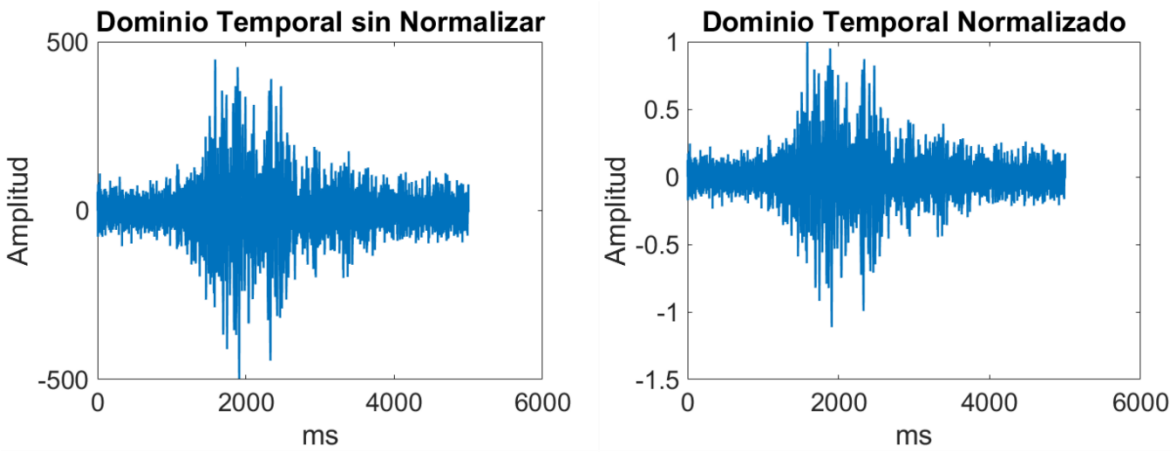
Base de Datos MicSigV1



En la Figura 10 se muestra un microsismo volcánico del Cotopaxi en el dominio del tiempo, en el literal a) se puede observar sin normalizar, y en b) se aprecia después de la normalización. Antes de proceder a la clasificación, los microsismos volcánicos sin procesar se sometieron a un desplazamiento para eliminar los efectos instrumentales y posteriormente se filtraron utilizando un filtro de respuesta al impulso finito (FIR, del inglés *Finite Impulse Response*) pasa banda, no causal, de fase cero, orden de 256 y la frecuencia de corte inferior definida en 0.7 Hz y la superior en 50 Hz. El propósito de este filtrado es principalmente eliminar un pico predominante relacionado con los microsismos marinos presentes a 0.2 Hz, para preservar así el contenido espectral principal de los microsismos volcánicos que son de interés de los investigadores (Pérez et al., 2020). El filtro digital no causal de fase cero procesa la señal de entrada tanto en el sentido directo como en el inverso. Es importante destacar que el filtro utilizado tiene un alto orden (lo que implica una banda de transición pequeña) y no afectan a la información relevante de los microsismos volcánicos, por lo que es seguro emplear un filtro no causal. Además, la fase lineal de los filtros FIR no introduce distorsión en la fase (Rousselet, 2012).

Figura 10

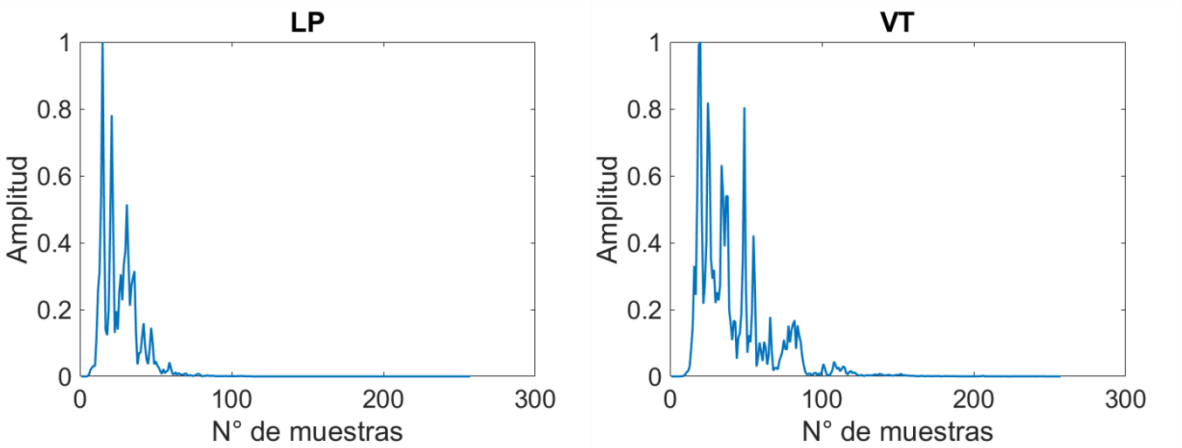
Representación microsismo volcánico del Cotopaxi en el dominio temporal



A los microsismos volcánicos en el dominio del tiempo se los procesa con 512 puntos de la FFT, lo que permitió realizar una transformación al dominio de la frecuencia. Este proceso generó una representación frecuencial compuesta por 257 muestras de cada uno de los microsismos volcánicos. La obtención de la FFT aporta información esencial sobre la distribución de la energía en la señal y facilita el análisis y clasificación posterior de los microsismos volcánicos. Una representación de un LP y VT en el dominio de la frecuencia normalizados, se muestran en la Figura 11.

Figura 11

Representación de microsismos volcánicos del Cotopaxi en el dominio frecuencial



Obtención del Conjunto de datos sin Wavelet

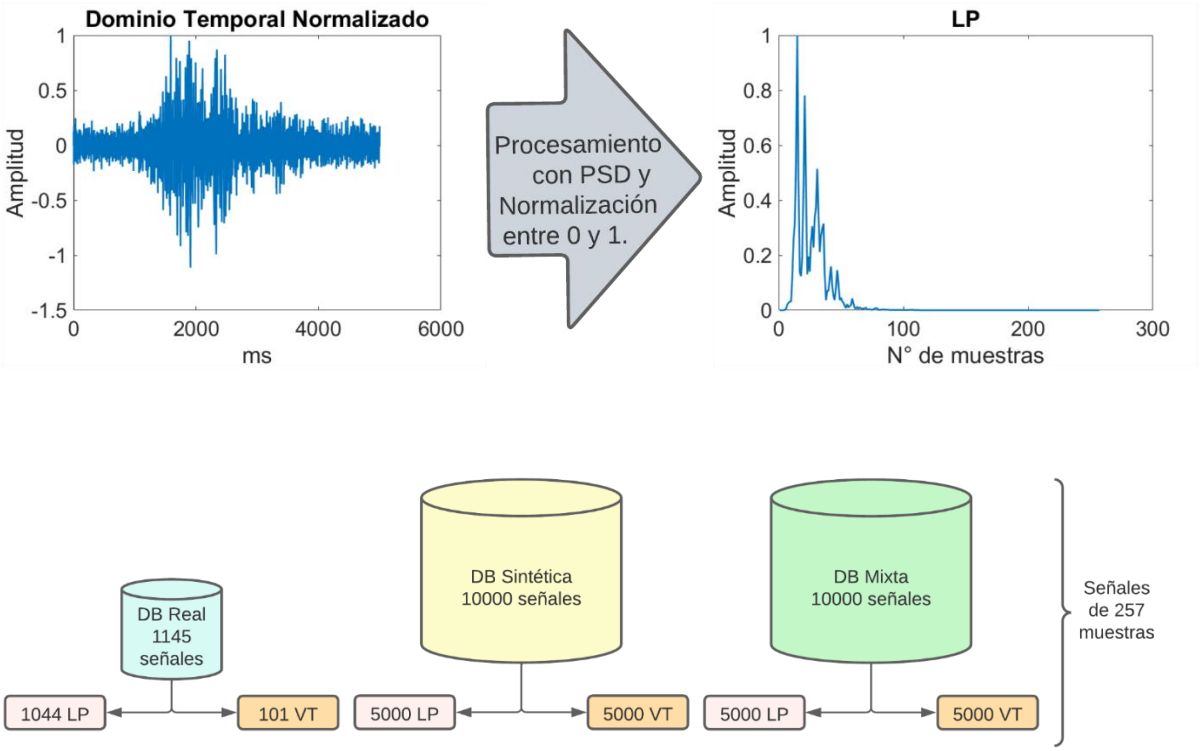
En el *set* de datos antes descrito, se encontró registro de 1044 microsismos volcánicos del tipo LP y únicamente 101 del tipo VT, finalmente se obtuvo una base de datos real de 1145 microsismos volcánicos. Dado la limitante del tipo VT encontrados en el *data set* del IGEPN, se optó por realizar DA mediante CGAN basado en el trabajo de (Galarza Cruz & Vega Vergara, 2022), para obtener un conjunto de datos más amplio que asegure el correcto entrenamiento y prueba de los modelos de DL. Se crearon 5000 microsismos volcánicos sintéticos LP y 5000

VT, dando como resultado una base de datos ‘Sintética’ con 10000 microsismos volcánicos en total, suficiente para los propósitos del presente trabajo.

Para asegurar la variedad y evitar el sesgo que pueda producirse con las señales sintéticas generadas se creó otra data *set* con microsismos volcánicos de la base de datos ‘Real’ y ‘Sintética’ mezclados de manera aleatoria dando un total de 10000 microsismos volcánicos en una base de datos mixta. Los tres *sets* de datos se utilizaron con los modelos de SA y DNN.

Figura 12

Representación del conjunto de datos sin Wavelet



Obtención del Conjunto de datos con Wavelet

Una vez obtenidas las bases de datos anteriores, se procedió a la aplicación de la transformada *Wavelet* a todos y cada uno de los microsismos volcánicos del tipo LP y VT presentes en la base de datos mixta. Para obtener los parámetros óptimos debe emplearse una *wavelet* madre de tipo 10, conocida como 'DB10' en el caso del modelo de *Daubechies* y 'SYM10' para el modelo de *Symlets*. Se obtuvieron los coeficientes de detalle de niveles 2, 3, 4 y 5, además del coeficiente de aproximación de nivel 6, posterior a ello se procedió a unificar los coeficientes conseguidos en un solo vector, tanto para *Daubechies* como para *Symlets* (Saltos Torres, 2014).

La aplicación de la FFT a una transformada Wavelet proporciona información valiosa sobre el contenido de frecuencia y la distribución de potencia de la señal a distintas escalas o niveles de la descomposición Wavelet. El significado físico de aplicar la FFT a una transformada Wavelet reside en la comprensión de las características frecuenciales de la señal a diferentes escalas o niveles. Nos permite identificar características importantes, como periodicidades, oscilaciones o resonancias, además, puede ayudar a detectar anomalías o cambios en el contenido frecuencial de la señal, lo que puede resultar especialmente útil en aplicaciones como el análisis de señales sísmicas o la identificación de patrones específicos en una señal.

Se procesa cada vector con los coeficientes elegidos de *Wavelet*, tanto del modelo *Daubechies* como *Symlets*, por cada uno de los microsismos volcánicos se extrajo la FFT, se logró cambiar al dominio de la frecuencia cada uno de ellos con 257 muestras. Las Figuras 13 y 14 muestran la FFT de un vector de coeficientes de *Wavelet* discreta de microsismos volcánicos en el dominio de la frecuencia, para los modelos *Daubechies* y *Symlets* respectivamente. Luego, se procedió a concatenar los vectores de coeficientes *Wavelet*

aplicados FFT con su correspondiente microsismo volcánico en la base de datos mixta, se obtuvo nuevos conjuntos de datos con microsismos volcánicos procesados y normalizados con 514 muestras cada uno. Las Figuras 15 y 16 muestran esta concatenación para los modelos *Daubechies* y *Symlets* respectivamente.

Figura 13

Representación de la transformada Wavelet Daubechies de microsismos volcánicos del Cotopaxi en el dominio frecuencial

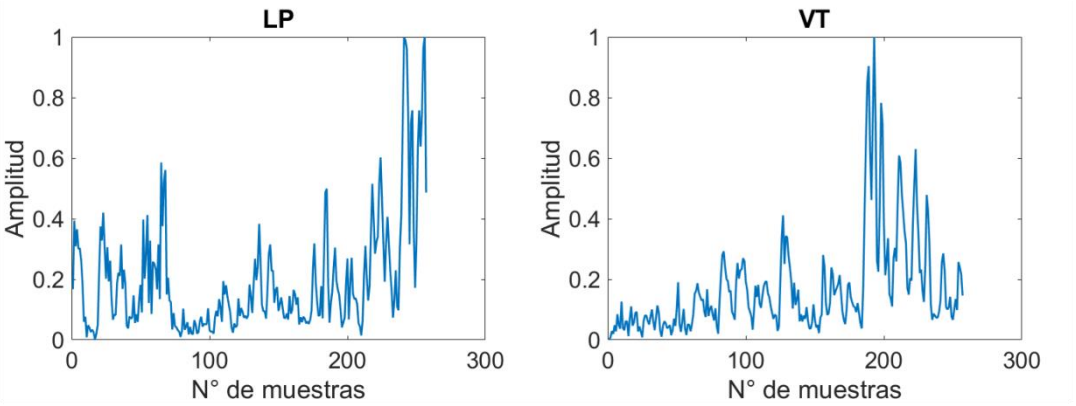


Figura 14

Representación de la transformada Wavelet Symlets de microsismos volcánicos del Cotopaxi en el dominio frecuencial

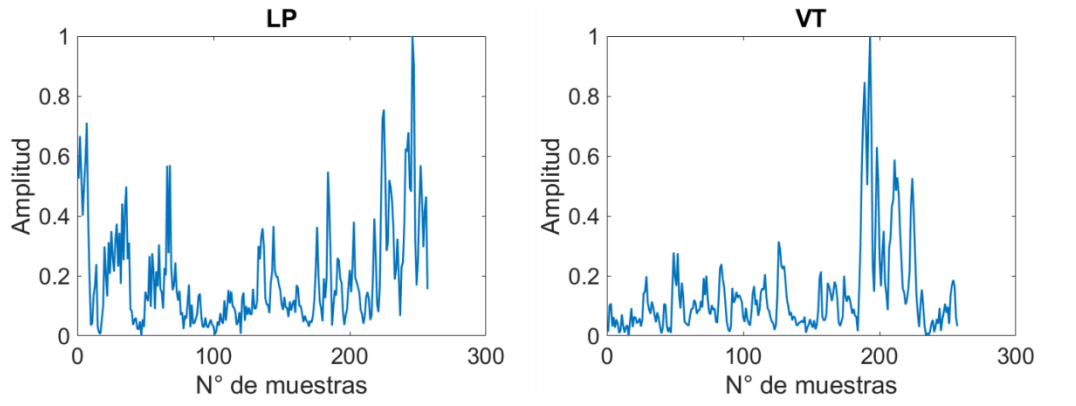


Figura 15

Representación de la FFT concatenada con la transformada Wavelet Daubechies de microsismos volcánicos del Cotopaxi en el dominio frecuencial

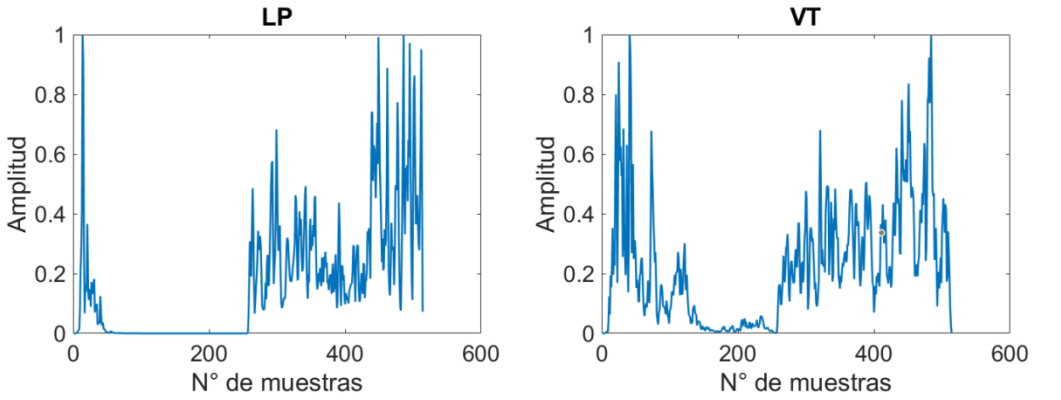
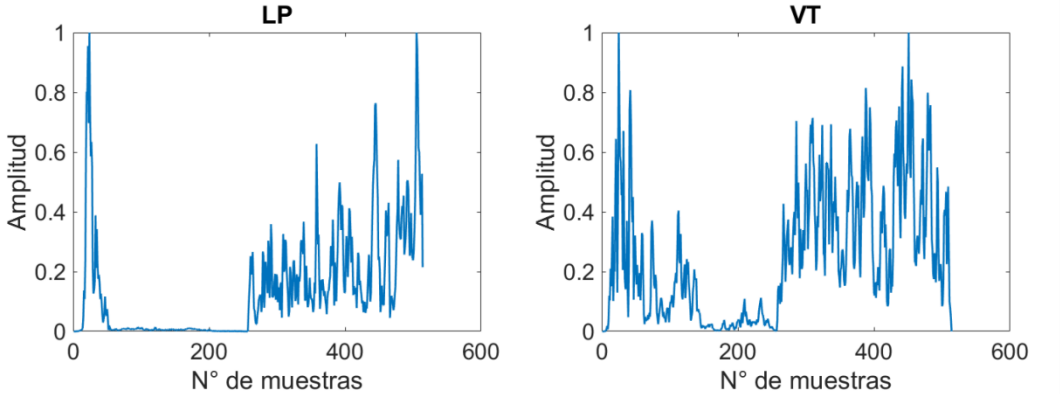


Figura 16

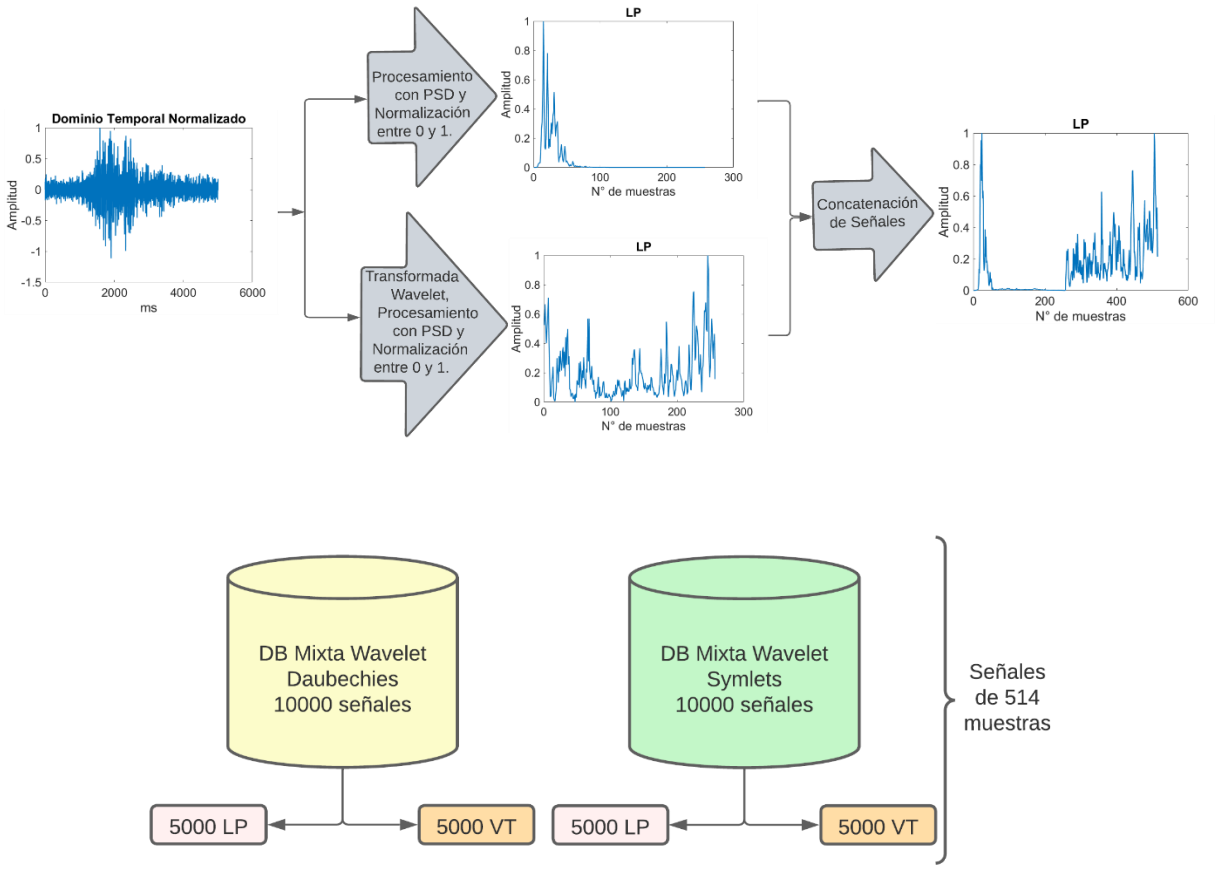
Representación de la FFT concatenada con la transformada Wavelet Symlets de microsismos volcánicos del Cotopaxi en el dominio frecuencial



Finalmente, se obtuvo dos bases de datos mixtas, correspondientes a los modelos de *Daubechies* y *Symlets* respectivamente, las cuales fueron nombras como DB ‘Mixta *Wavelet Daubechies*’ y DB ‘Mixta *Wavelet Symlets*’ que se utilizaron con los modelos de SA y DNN.

Figura *17*

Representación del conjunto de datos para entrenamiento y prueba con Wavelet



Diseño e Implementación

Para determinar el porcentaje de la cantidad de señales de las bases de datos con las que se trabaja para el entrenamiento y el *test* de los modelos de DL, se realiza varias pruebas variando diferentes rangos, los cuales se variaron de la siguiente forma:

- 80.00% para entrenamiento y 20.00% para prueba.
- 70.00% para entrenamiento y 30.00% para prueba.
- 60.00% para entrenamiento y 40.00% para prueba.
- 50.00% para entrenamiento y 50.00% para prueba.

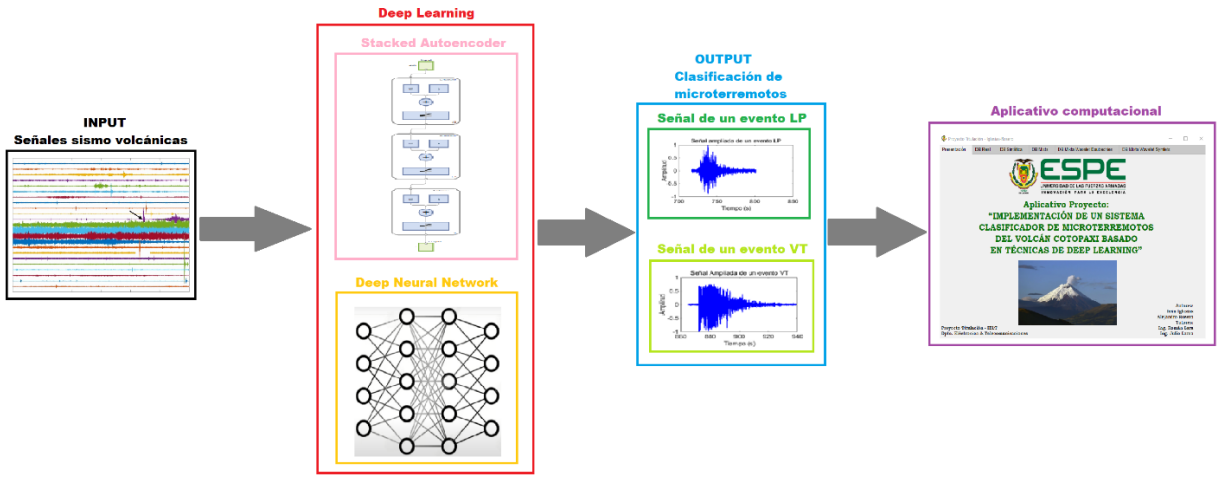
- 40.00% para entrenamiento y 60.00% para prueba.
- 30.00% para entrenamiento y 70.00% para prueba.
- 20.00% para entrenamiento y 80.00% para prueba.

Mediante prueba error se determinó que el mejor rango para trabajar con modelos de DL es 70.00% para entrenamiento y 30.00% para prueba, de igual manera en el grupo de investigación en investigaciones previas han utilizado estos porcentajes para el entrenamiento de ML y DL.

En la Figura 18 se muestra el diagrama general del sistema para la clasificación de microsismos volcánicos del Cotopaxi, donde se tiene como bloque de entrada los microsismos volcánicos de las bases de datos mencionadas en el capítulo anterior, posteriormente se tiene el bloque de los modelos de Deep Learning que son entrenados para la clasificación de eventos LP y VT, a continuación se presenta el bloque de la clasificación de los microsismos volcánicos por los modelos de SA y DNN, finalmente los resultados de la clasificación y métricas de rendimiento se muestran en el aplicativo computacional.

Figura *18*

Diagrama general del sistema

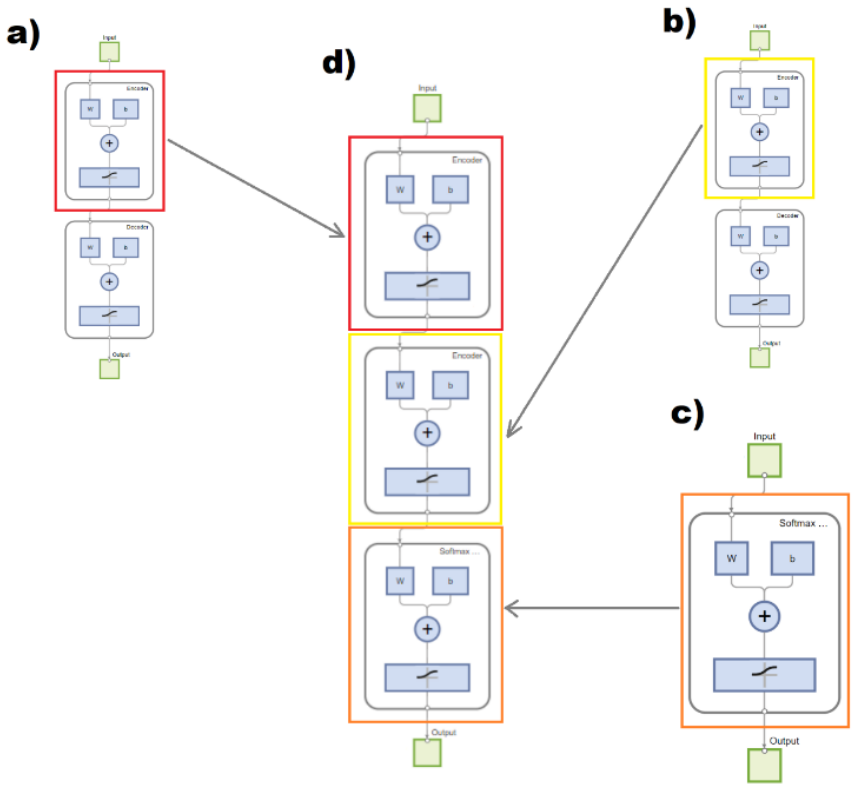


Algoritmo de SA

Para la programación del algoritmo en *MATLAB*® se crea un archivo, el cual nos permite escribir todo el código necesario para desarrollar el modelo de DL. Se utiliza la versión R2022a de *MATLAB*® debido a que contine funciones propias para programar el algoritmo del SA. El diseño se basa en la creación de dos *autoencoder* y una capa *softmax* de clasificación de los microsismos volcánicos LP y VT, posteriormente se apila los codificadores de cada *autoencoder* junto a la capa *softmax* para tener como resultado final un SA que permita clasificar los microsismos volcánicos LP y VT del volcán Cotopaxi. En la Figura 19 se muestra los bloques del diseño implementado para el SA.

Figura *19*

Diagrama del SA



Nota. a) Primer *Autoencoder*, b) Segundo *Autoencoder*, c) Capa *Softmax* de Clasificación, d) *SA*.

Cargar bases de datos y selección de rango de datos para SA

El primer paso de la programación del algoritmo es cargar la base de datos que se utiliza para el entrenamiento y validación del SA, esta base de datos puede ser cualquier de las que están explicadas en el capítulo anterior. También se escoge el porcentaje de las señales que contiene la base de datos, en este diseño se utiliza el 70.00% de las señales para entrenar y el 30.00% sobrante para validar el funcionamiento del SA.

Código 1

Cargar base de datos y selección de rango de datos.

```
load dataTrain;
X=X';Y=Y';

X1=X(:,[1:3500,5001:8500]);
Y1=Y(:,[1:3500,5001:8500]);

X2=X(:,[3501:5000,8501:10000]);
Y2=Y(:,[3501:5000,8501:10000]);
```

Nota. El código muestra que se carga una base de datos que contiene en la variable X los microsismos volcánicos del volcán Cotopaxi y en Y las etiquetas de los mismos. Se transponen los valores de cada variable para poder trabajar con ellas posteriormente en el código. En X1 se guarda el 70.00% de las señales LP y VT, Y1 contiene el setenta por ciento de las etiquetas de las señales escogidas en X1, X2 guarda el 30.00% de las señales restantes LP y VT, Y2 contiene el treinta por ciento de las etiquetas restantes de las señales guardadas en X2.

Primer Autoencoder

Seguidamente, se programa el primer *autoencoder* el cual tiene la finalidad de comprimir las mejores características de los datos de entrada que pertenecen a la base de datos escogida. Se utiliza una función propia de *MATLAB*® para la implementación de este *autoencoder* que se muestra en el Código 2.

Código 2

Programación primer autoencoder.

```
autoenc1 = trainAutoencoder(X1, a, 'MaxEpochs', b, 'L2WeightRegularization', c,
'SparsityRegularization', d, 'SparsityProportion', e, 'ScaleData', false);
feat1 = encode(autoenc1, X1);
```

Nota. En el código mostrado se indica cómo se configura la función `trainAutoencoder`, que permite la creación del primer *autoencoder*. Adicionalmente, se utiliza la función `encode`, la cual extrae las características codificadas del autoenc1 entrenado y se almacena en la variable `feat1`.

Tabla 1

Parámetros de la función trainAutoencoder Primer Autoencoder.

Parámetro	Valor	Descripción
X1	-	Datos de entrenamiento, especificados como una matriz de muestras de entrenamiento.
Neuronas	a	Es el número de neuronas de la capa oculta. Debe ser menor de la mitad de las muestras de las señales de la base de datos con la que se entrena el *autoencoder*.
MaxEpochs	b	Número máximo de épocas o iteraciones de entrenamiento. Se escoge valores aleatorios entre [1-1000].
L2WeightRegularization	c	Es el coeficiente para el regularizador de peso L2 en la función de coste. Se escoge valores aleatorios en el rango de [0.005, 0.01].
SparsityRegularization	d	Coeficiente que controla el impacto del regularizador de dispersión en la función de coste. Se escoge valores aleatorios en el rango de [0, 10].
SparsityProportion	e	La proporción de dispersión es un parámetro del regularizador de dispersión, controla la dispersión de la salida de la capa oculta. Los valores deben estar en el rango de [0,1].
ScaleData	false	Indicador para escalar los datos de entrada.

Nota. La tabla muestra los parámetros escogidos para el diseño del primer *autoencoder*.

Segundo Autoencoder

Para la programación de este *autoencoder* se usa como datos de entrada las características codificadas del primer *autoencoder* entrenado, de igual manera se utiliza la función `trainAutoencoder` para implementarlo, en el Código 3 se indica lo mencionado.

Código 3

Programación segundo autoencoder

```
autoenc2 = trainAutoencoder(feat1, f, 'MaxEpochs', g,'L2WeightRegularization',
h,'SparsityRegularization', i, 'SparsityProportion', j, 'ScaleData', false);

feat2 = encode(autoenc2, feat1);
```

Nota. Se indica la configuración de la función `trainAutoencoder` para la creación del segundo *autoencoder*. También se utiliza la función `encode`, la cual extrae las características codificadas del *autoenc2* entrenado y se guarda en la variable feat2.

Tabla 2

Parámetros de la función trainAutoencoder Segundo Autoencoder.

Parámetro	Valor	Descripción
feat1	-	Datos de entrenamiento, en este diseño son las características codificadas del primer *autoencoder*.
Neuronas	f	Es el número de neuronas de la capa oculta. Debe ser la mitad del número de neuronas escogidas en el primer *autoencoder*.
MaxEpochs	g	Número máximo de épocas o iteraciones de entrenamiento. Debe ser la mitad del número de épocas escogidas en primer *autoencoder*.
L2WeightRegularization	h	Es el coeficiente para el regularizador de peso L2 en la función de coste. Debe ser la mitad del número escogido en el primer *autoencoder*.
SparsityRegularization	i	Coeficiente que controla el impacto del regularizador de dispersión en la función de coste. Debe ser el mismo número escogido en el primer *autoencoder*.
SparsityProportion	j	La proporción de dispersión es un parámetro del regularizador de dispersión, controla la dispersión de la salida de la capa oculta. Debe ser el número de *SparsityProportion* escogido en el primer *autoencoder* menos 0.05.
ScaleData	false	Indicador para re escalar los datos de entrada.

Nota. La tabla muestra los parámetros escogidos para el diseño del segundo *autoencoder*.

Capa *softmax* para clasificación y SA

Posteriormente, se programa la capa de clasificación *softmax* teniendo como datos de entrada las características codificadas del segundo *autoencoder*. Después se apila los

codificadores del primer y segundo *autoencoder* para ser unidos a la capa de clasificación *softmax* y finalmente formar el SA.

Código 4

Programación de la capa de clasificación softmax y SA.

```
softnet = trainSoftmaxLayer(feat2, Y1, 'MaxEpochs', b);
stackednet = stack(autoenc1, autoenc2, softnet);
stackednet = train(stackednet, X1, Y1);
```

Nota. La función `trainSoftmaxLayer` permite crear una capa de clasificación, para ser entrenada necesita de una matriz de datos de entrada, matriz de clases o etiquetas y el número de épocas necesarias. `Stack` es una función que apila los codificadores de los *autoencoders* en conjunto con la capa *softmax*. Finalmente, la función `train` entrena el SA utilizando el rango de señales escogidas (X1) con sus respectivas etiquetas (Y1).

Tabla 3

Parámetros de la función trainSoftmaxLayer.

Parámetro	Valor	Descripción
feat2	-	Datos de entrenamiento, para este diseño son las características codificadas del segundo *autoencoder*.
Y1	-	Es la matriz de clases o etiquetas de los datos de entrada.
MaxEpochs	b	Número máximo de épocas o iteraciones de entrenamiento. Debe ser el mismo número de épocas escogidas del primer *autoencoder*.

Nota. La tabla muestra los parámetros escogidos para el diseño de la capa *softmax* y su configuración se muestra en el Código 4.

Validación del SA

Finalmente, el SA entrenado se prueba con las señales escogidas para la validación, en el Código 5 se muestra lo mencionado.

Código 5

Programación de la prueba del SA y Matriz de confusión.

```
SA = stackednet(X2);
plotconfusion(Y2, SA);
```

Nota. Se prueba el SA entrenado para clasificar las señales almacenadas en X2 y se muestra la matriz de confusión con respecto a Y2 que contiene las etiquetas de los microsismos volcánicos de X2.

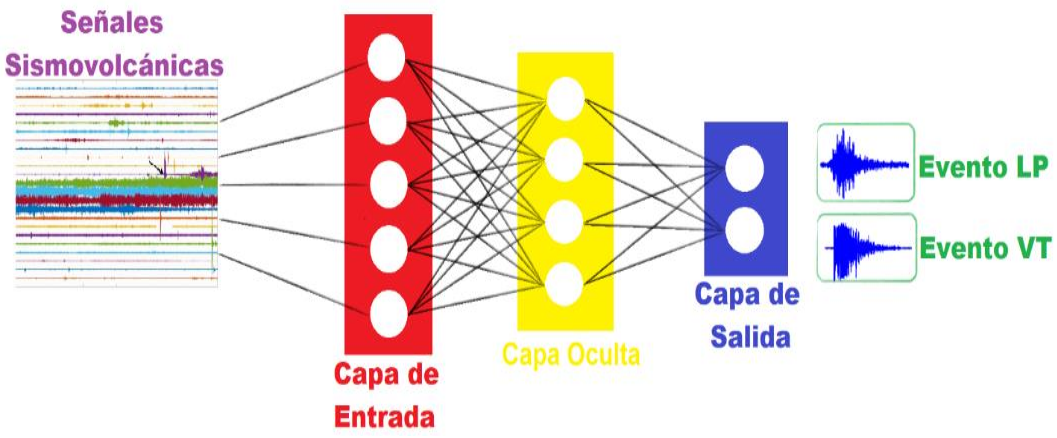
Mediante ensayo-error se realiza varias pruebas con las diferentes bases de datos que fueron explicadas en el anterior capítulo, la primera prueba utiliza la base datos 'Real' de los microsismos volcánicos del Cotopaxi, en donde se varía los cinco parámetros que se utilizan en la función `trainAutoencoder`, estos fueron variados de manera aleatoria en diferentes rangos, el mismo procedimiento se realizó para las demás bases de datos.

Algoritmo de la Deep Neural Network

La programación de este algoritmo se desarrolla en un archivo en *MATLAB*®, en donde se utiliza funciones propias del software que permiten la creación de una DNN. En la Figura 20, se aprecia los bloques que conforman este modelo de DL, el cual está formado por una capa de entrada que recibe todas las características de las señales que contiene la base de datos con la que se va entrenar, después se tiene una capa oculta y finalmente una capa de salida que se encargar de la clasificación de los microsismos volcánicos LP y VT del Cotopaxi.

Figura 20

Diagrama de DNN



Cargar bases de datos y selección de rango de datos para DNN

De la misma forma que se realizó con el SA se carga la base de datos con la que se va trabajar, manteniendo los mismos porcentajes de señales para el entrenamiento y validación de la DNN, como se muestra en el Código 6.

Código 6

Cargar base de datos

```
load dataTrain;
XTrain=X;
YTrain=categorical(Y);

X1=XTrain([1:3500,5001:8500],:);
Y1=YTrain([1:3500,5001:8500],:);

X2=XTrain([3501:5000,8501:10000],:);
Y2=YTrain([3501:5000,8501:10000],:);
```

Nota. En la variable `XTrain` se almacena todos los microsismos volcánicos de la base de datos y en `YTrain` se guarda una matriz categórica de las etiquetas de los microsismos volcánicos de la base de datos. Se almacena el 70.00% de los microsismos volcánicos y sus etiquetas en las

variables X1 y Y1, lo cual sirve para el entrenamiento, el 30.00% sobrante de los microsismos volcánicos y sus etiquetas se guarda en X2 y Y2 para la validación de la DNN entrenada.

Sintaxis de la construcción de la Deep Neural Network

Primero tenemos que definir las diferentes capas de nuestra DNN, comenzando por la capa de entrada que recibe los datos que contienen los microsismos volcánicos de la base de datos escogida, posteriormente se crea la capa oculta con sus diferentes funciones para aprender y finalmente se tiene la capa de salida que es la encargada de la clasificación los microsismos volcánicos LP y VT, lo mencionado está en el Código 7.

Código 7

Programación de las capas de la Deep Neural Network.

```
numFeatures=257;
numClasses=2;

layers = [
featureInputLayer(numFeatures,'Normalization', 'none')
fullyConnectedLayer(50)
batchNormalizationLayer
reluLayer
fullyConnectedLayer(numClasses)
softmaxLayer
classificationLayer];
```

Nota. Se crea una variable `numFeatures` indicando el número de muestras que se tiene por cada microsismo volcánico de la base de datos, la variable `numClasses` indica el número de clases para la clasificación de microsismos volcánicos LP y VT. También se muestra la configuración de las capas que permiten crear la DNN.

Tabla 4

Funciones de las distintas capas de la DNN.

Capa	Funciones	Descripción
Capa de entrada	*featureInputLayer(numFeatures, 'Normalization', 'none')*	Es una capa de entrada de características, ingresa datos de características a una red y aplica la normalización de datos, en este diseño no se normaliza los datos de entrada.
Capa oculta	*fullyConnectedLayer(x)*	Es una capa completamente conectada que combina las características de la capa de entrada y como tamaño de salida de la capa se escogió 50.
	batchNormalizationLayer	Se normaliza las activaciones y los gradientes, haciendo que el entrenamiento de la red sea un problema de optimización más fácil y acelera el entrenamiento de la red, también reduce la sensibilidad a la inicialización de la red.
	reluLayer	Una capa *ReLU* realiza una operación de umbral para cada elemento de la entrada, donde cualquier valor menor que cero se establece en cero.
Capa de salida	*fullyConnectedLayer(numClasses)*	Es la última capa totalmente conectada que combina las características para clasificar los microsismos volcánicos. Se escoge como salida 2 tipos de clasificación.
	softmaxLayer	La función de activación *softmax* normaliza la salida de la capa completamente conectada. La salida de la capa *softmax* consiste en números positivos que se suman a uno, que luego se pueden utilizar como probabilidades de clasificación por la capa de clasificación.
	classificationLayer	La capa final es la capa de clasificación. Esta capa utiliza las probabilidades devueltas por la función de activación *softmax*, para asignar a cada entrada de las clases mutuamente excluyentes y calcular la pérdida.

Nota. La tabla indica todas las funciones utilizadas para la creación de las capas de la DNN y su configuración se muestra en el Código 7.

Opciones de entrenamiento de la DNN

Después de haber creado las capas de la DNN se programa los parámetros de la función `trainingOptions` con los que se va entrenar el modelo de DL, en el Código 8 se muestra como esta función permite elegir parámetros como el optimizador Adam, el valor del mini lote para el entrenamiento, luego se mezcla los datos en cada época y finalmente se muestra el progreso del entrenamiento de aprendizaje de la DNN.

Código 8

Programación de la función trainingOptions.

```
miniBatchSize = 16;

options = trainingOptions('adam', ...
'MiniBatchSize',miniBatchSize, ...
'Shuffle','every-epoch', ...
'Plots','training-progress', ...
'Verbose',false);
```

Nota. Se crea una variable `miniBatchSize` con el valor del tamaño del mini lote de entrenamiento que se utiliza. Después se asigna los diferentes parámetros para la función `trainingOptions`.

Tabla 5

Parámetros de la función trainingOptions.

Parámetro	Valor	Descripción
adam	-	Optimizador Adam, mantiene una media móvil a nivel de elemento tanto de los gradientes de parámetros como de sus valores cuadrados.
MiniBatchSize	*miniBatchSize*	Tamaño de mini lote que desea usar para cada iteración de entrenamiento, Un mini lote es un subconjunto del conjunto de entrenamiento que se usa para evaluar el gradiente de la función de pérdida y actualizar los pesos. En este diseño el tamaño de es de 16.
Shuffle	*every-epoch*	Opción para cambiar el orden de los datos, *'every-epoch'*: cambiar el orden de los datos de entrenamiento antes de cada época de entrenamiento y cambiar el orden de los datos de validación antes de cada validación de la red.
Plots	*training-progress*	Crea una figura y muestra las métricas de entrenamiento en cada iteración. La gráfica muestra información sobre la pérdida y precisión de los mini lotes, también indica la pérdida y precisión de validación, así como información adicional sobre el progreso del entrenamiento de la DNN.
Verbose	*false*	Indicador para mostrar información sobre el progreso del entrenamiento en la ventana de comandos, especificado como 1 (true) o 0 (false).

Nota. La tabla indica los parámetros escogidos para el diseño de la DNN y su configuración se muestra en el Código 8.

A continuación, se muestra la Figura 21, la cual nos indica todas las características que tiene el optimizador Adam escogido para el entrenamiento de la DNN.

Figura 21

Características del Optimizador Adam en MATLAB®

```
options =

  TrainingOptionsADAM with properties:

           GradientDecayFactor: 0.9000
       SquaredGradientDecayFactor: 0.9990
                         Epsilon: 1.0000e-08
                InitialLearnRate: 1.0000e-03
               LearnRateSchedule: 'none'
               LearnRateDropFactor: 0.1000
               LearnRateDropPeriod: 10
                  L2Regularization: 1.0000e-04
          GradientThresholdMethod: 'l2norm'
                GradientThreshold: Inf
                        MaxEpochs: 30
                    MiniBatchSize: 16
                          Verbose: 0
                  VerboseFrequency: 50
                    ValidationData: []
                ValidationFrequency: 50
                 ValidationPatience: Inf
                          Shuffle: 'every-epoch'
                   CheckpointPath: ''
                CheckpointFrequency: 1
          CheckpointFrequencyUnit: 'epoch'
             ExecutionEnvironment: 'auto'
                       WorkerLoad: []
                        OutputFcn: []
                             Plots: 'training-progress'
                   SequenceLength: 'longest'
              SequencePaddingValue: 0
           SequencePaddingDirection: 'right'
               DispatchInBackground: 0
            ResetInputNormalization: 1
        BatchNormalizationStatistics: 'population'
                      OutputNetwork: 'last-iteration'
```

Nota. Las características que se muestran son por defecto que tiene *MATLAB®* al usar este tipo de optimizador. Los únicos valores que se cambiaron por defecto son el `MiniBatchSize`, `Shuffle` y `Plots`.

Entrenamiento y validación de la DNN.

Finalmente se programa el entrenamiento de la DNN, como se muestra en el Código 9 para entrenar se utiliza la función `trainNetwork`, la misma que tiene como parámetros de entrada las matrices de los microsismos volcánicos y sus etiquetas correspondientes, también se agrega las capas y opciones de entrenamiento que se explicaron anteriormente. Seguidamente se prueba la DNN entrenada clasificando los microsismos volcánicos escogidos para la validación, los resultados se muestran en una matriz de confusión comparando con las etiquetas de dichas señales.

Código 9

Programación del entrenamiento y validación de la Deep Neural Network.

```
net = trainNetwork(X1, Y1, layers, options);

YPred = classify(net, X2);
figure()
plotconfusion(YPred,Y2 )
```

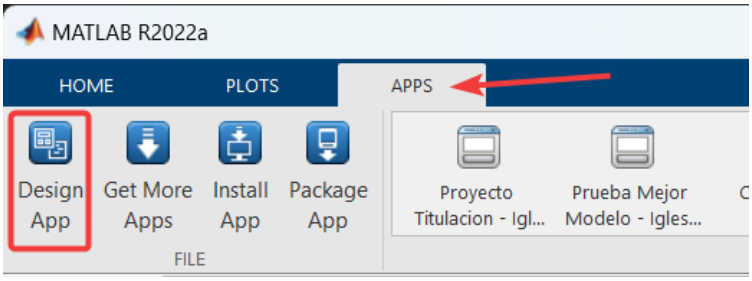
Nota. En la función `trainNetwork` se utiliza las siguientes variables: *X1* y *Y1* que son el 70.00% de los microsismos volcánicos y sus etiquetas respectivamente, *layers* tiene las capas diseñas de la DNN y `options` almacena las opciones con las que se entrena el modelo diseñado. La función `classify` permite que la DNN entrenada clasifique el 30.00% de los microsismos volcánicos en la variable X2 y los resultados de la clasificación se muestran en la matriz de confusión.

Diseño del Aplicativo Computacional

Para la elaboración del aplicativo computacional se utilizó la herramienta '*Design App*' de *MATLAB*® dentro de la pestaña '*APP*' en la interfaz principal como se muestra en la Figura 22 a continuación:

Figura 22

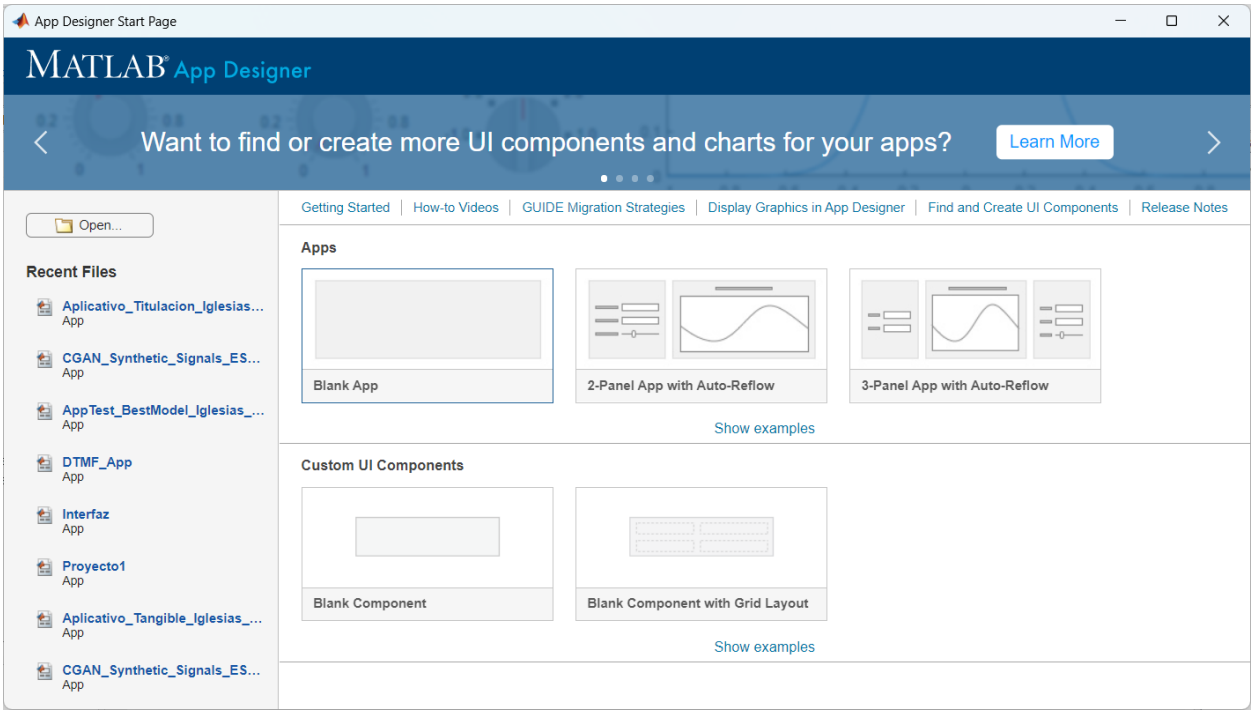
Herramienta 'Design App' en MATLAB®



Se despliega una nueva ventana llamada '*App Designer Start Page*', en la que se pueden abrir proyectos previos, archivos recientes si los hubiera, y se pueden crear tanto aplicaciones como componentes de *UI* nuevos, basados en plantillas o totalmente en blanco, se observa la ventana en la Figura 23.

Figura 23

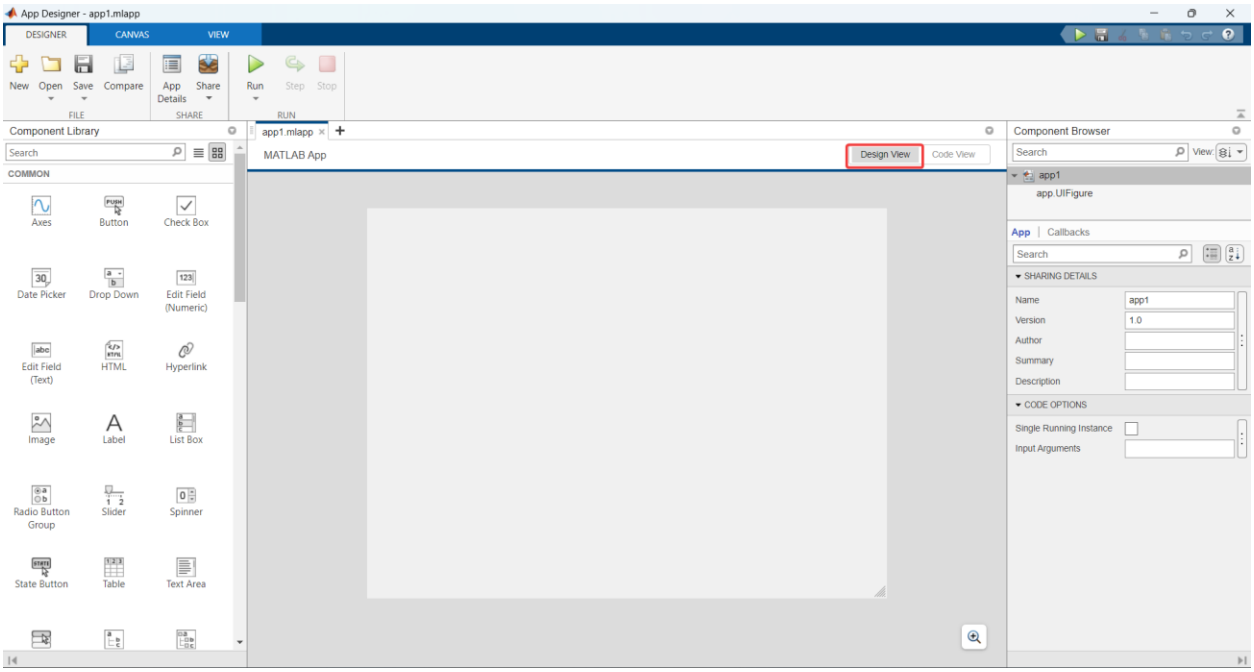
Ventana 'App Designer Start Page' de MATLAB®



Escogimos ‘*Blank App*’ y se procedió a elegir los componentes y programar su funcionamiento para la creación del aplicativo. En la ventana abierta al elegir crear una aplicación en blanco, se tienen dos partes principales de un nuevo aplicativo, la parte del diseño visual que se realiza en la pestaña ‘*Design View*’ remarcada en la Figura 24, donde se puede escoger entre una librería de componentes que se arrastran al recuadro central y se los manipula para que queden acorde a lo deseado por el usuario.

Figura 24

Pestaña ‘Design View’ de ‘App Designer’ en MATLAB®

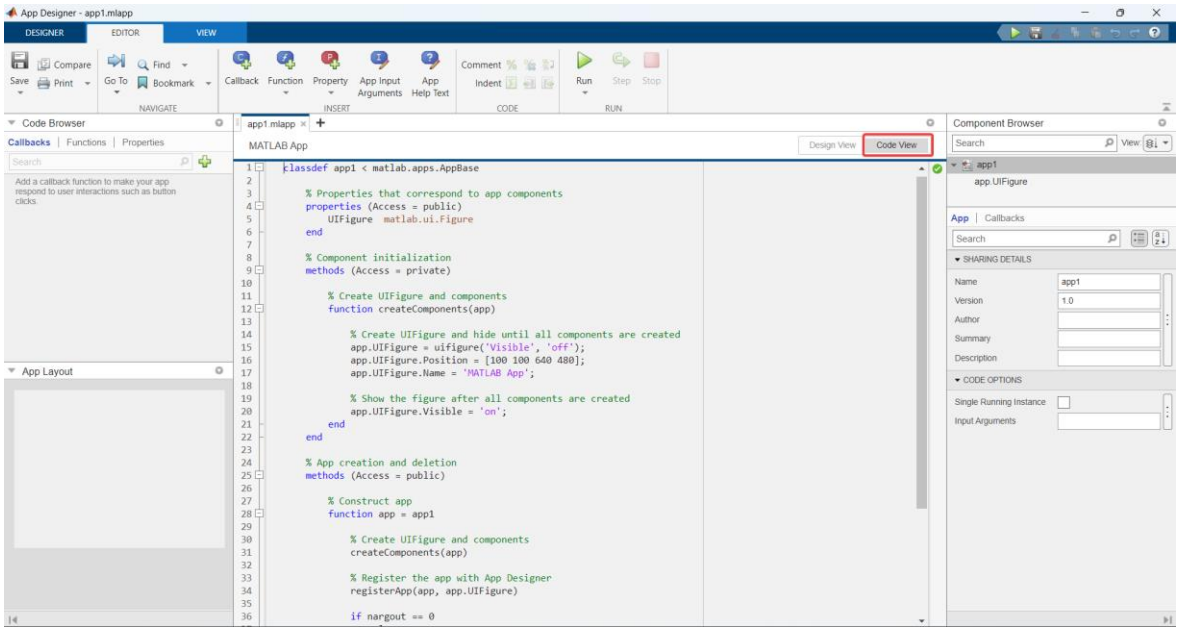


La parte del diseño funcional, el código de programación que permite al aplicativo realizar las acciones que desea el programador mediante el paradigma de la programación orientada a objetos, se puede observar en la Figura 25, donde está remarcada la pestaña de ‘*Code View*’. En esta vista del código se pueden programar las acciones por componente en la interfaz de usuario o mediante funciones con el paradigma Modelo-Vista-Controlador (MVC), donde se

programa una o varias funciones y luego se las llama en cada componente. En el diseño del aplicativo computacional se utilizó MVC para su programación y funcionamiento.

Figura 25

Pestaña 'Code View' de 'App Designer' en MATLAB®



El diseño visual fue realizado de la siguiente manera, primero se agrandó el área en la que se ubica la aplicación, llamado *canvas*. Después, se eligió una vista por pestañas, llamada '*Tab Group*', desde la biblioteca de componentes en la categoría '*Containers*', y se la igualó con el tamaño del *canvas* para que toda la aplicación se ubique dentro del grupo por pestañas. En el aplicativo se tienen 5 pestañas, la primera muestra una carátula de presentación para el proyecto, la segunda y tercera pestañas contienen las gráficas de las matrices de confusión, tanto para SA como DNN, además de una tabla que muestra las métricas de rendimiento para los modelos de SA y DNN escogidos, también existe un cuadro de texto que muestra la ruta de la base de datos elegida para su procesamiento, otro cuadro de texto que muestra el

significado de los símbolos en las matrices de confusión, además del botón que se presiona para elegir la base de datos y procesarla.

La segunda pestaña se centra en el procesamiento con modelos entrenados sin *Wavelet* y la tercera en los que se utilizó *Wavelet* para el entrenamiento. Las matrices de confusión se presentan en un componente '*Panel*', tanto para DNN como SA, de manera dinámica dependiendo de la base de datos procesada por el aplicativo, y en la parte inferior de la pestaña se muestran las métricas de rendimiento de SA y DNN en un componente de tabla llamado '*UITable*'. Finalmente, el componente entre los paneles de matrices de confusión y la tabla de las métricas de rendimiento tiene el nombre de '*PathTextArea*' y nos ayuda a explorar la computadora donde se encuentre el aplicativo y elegir archivos para ser procesados por él. La tercera pestaña tiene los mismos componentes que la segunda, sin embargo, se centra en el procesamiento de las bases de datos utilizando los modelos entrenados con *Wavelet*. En las Figuras 26 y 27 se muestran cómo se ven las pestañas 2 y 3 una vez diseñadas.

Figura 26

Diseño de la segunda pestaña del aplicativo computacional en MATLAB®

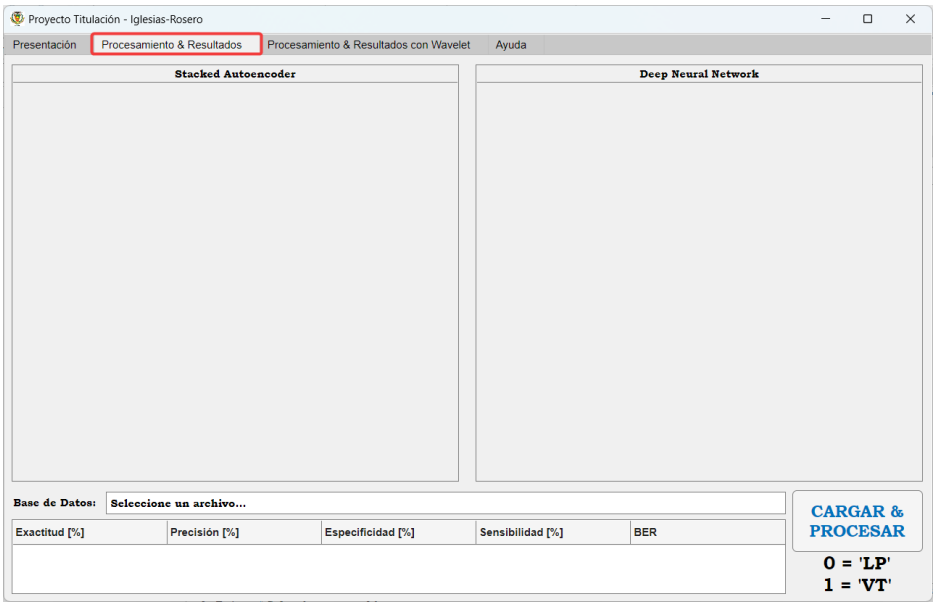
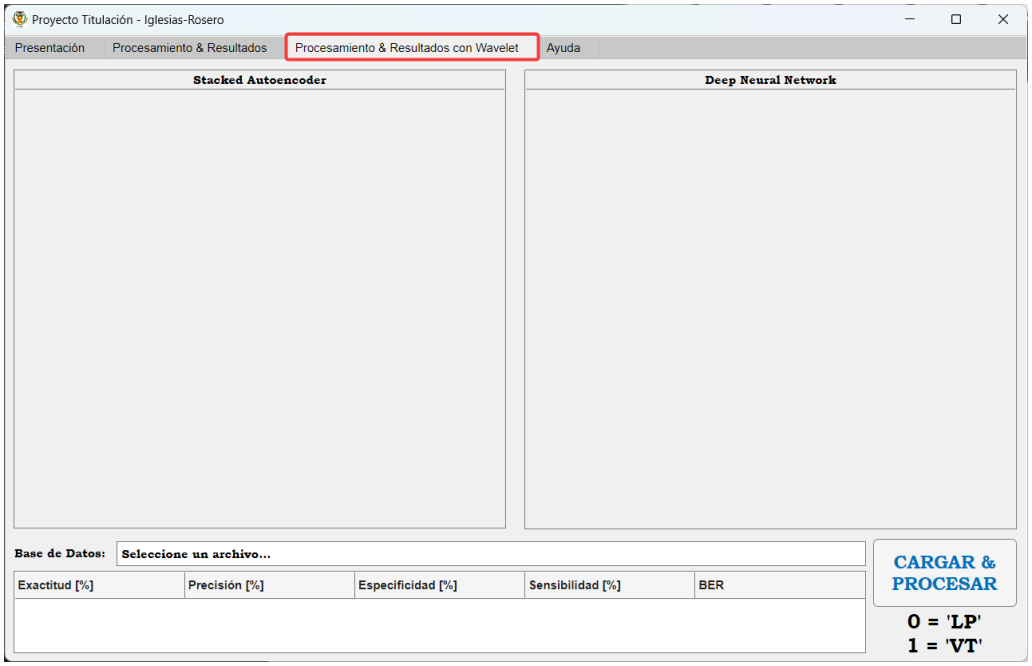


Figura 27

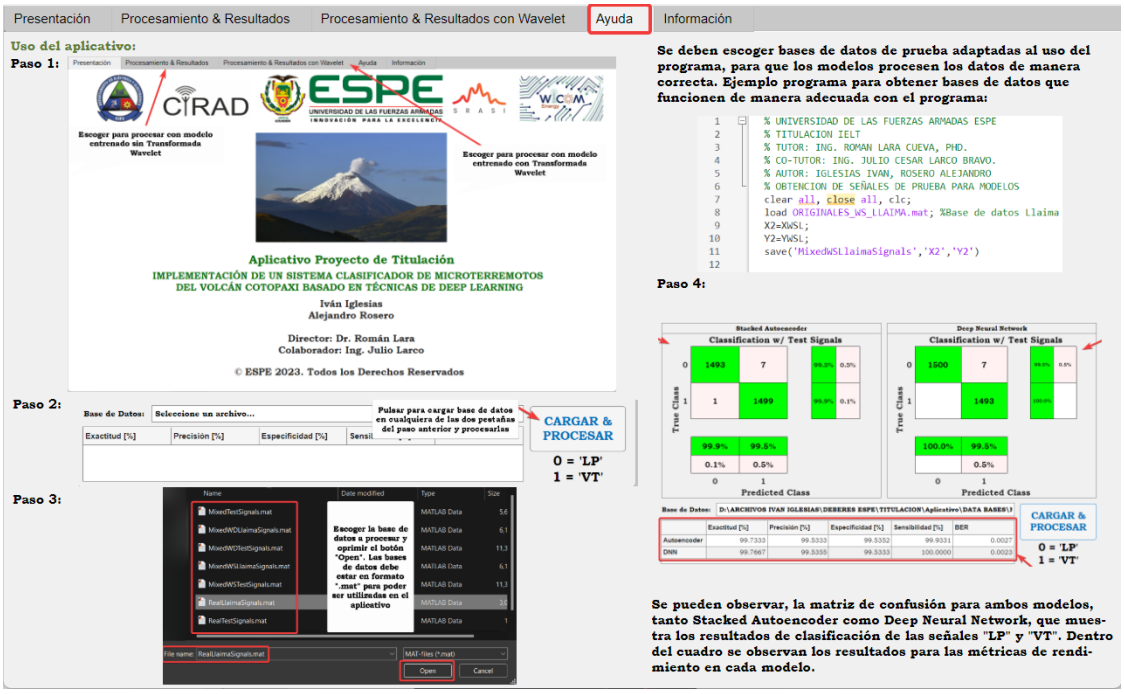
Diseño de la tercera pestaña del aplicativo computacional en MATLAB®



La cuarta pestaña es para ayuda del usuario del aplicativo, contiene instrucciones de como operar la aplicación y la preparación de las bases de datos que se pueden utilizar para procesar en las pestañas dos y tres mencionadas anteriormente. Simplemente se utilizan componentes '*Image*' para mostrar las imágenes correspondientes y '*Label*' para las instrucciones en pantalla. En la Figura 28 se muestra el diseño de la cuarta pestaña del aplicativo computacional.

Figura 28

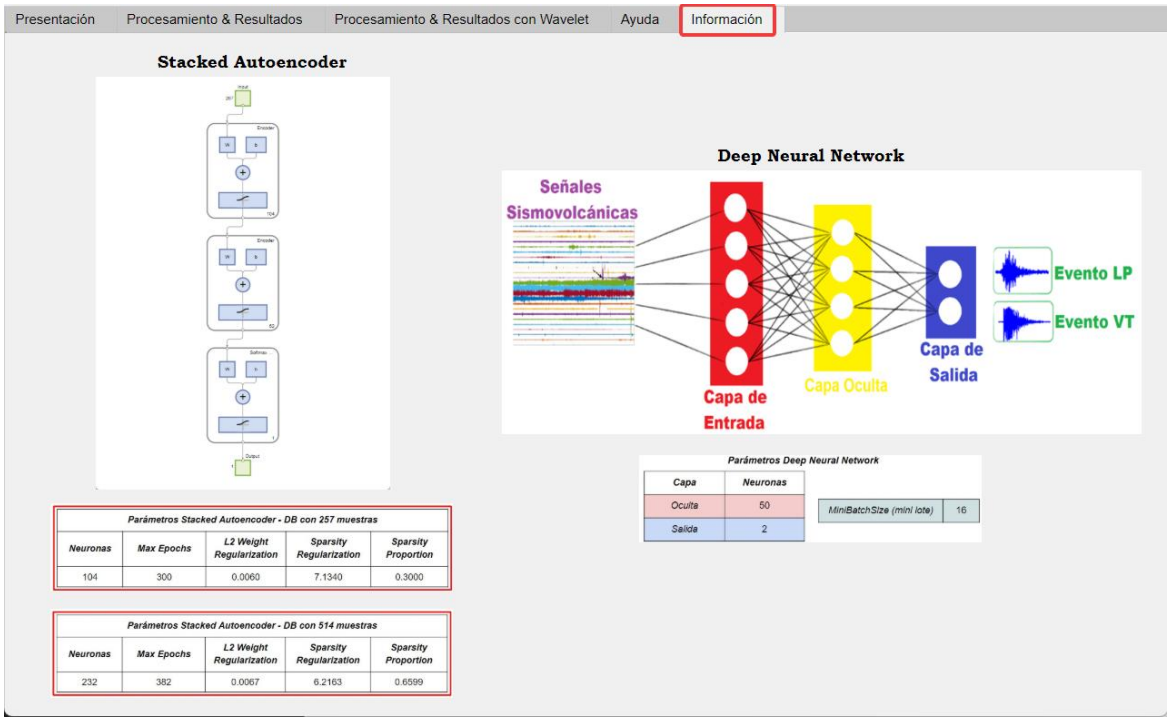
Diseño de la cuarta pestaña del aplicativo computacional en MATLAB®



La quinta y última pestaña contiene la información acerca de los algoritmos de DL desarrollados, para el SA se presenta su diagrama de bloque y los mejores parámetros utilizados para el entrenamiento de los modelos escogidos para el aplicativo, tanto con Wavelet como sin ella. También se muestran el diagrama de bloques que representa el funcionamiento de la DNN, así como sus mejores parámetros utilizados para los modelos que funcionan en el aplicativo, sin Wavelet y utilizándola. En la Figura 29 se muestra el diseño de la quinta pestaña del aplicativo computacional.

Figura 29

Diseño de la quinta pestaña del aplicativo computacional en MATLAB®



A continuación, se detalla el código utilizado en el diseño del aplicativo, que una vez obtenidos los modelos con los mejores resultados tanto de SA como DNN para datos con y sin *Wavelet*, se adjuntan al funcionamiento lógico en forma de modelos independientes para el procesamiento en la segunda y tercera pestañas de las bases de datos escogidas y preparadas como se indica en la pestaña '*Ayuda*'. Los modelos se extraen entrenados como archivos de *MATLAB®*, y se agregan a cada pestaña que realiza el procesamiento con y sin *Wavelet* respectivamente en forma de código, como si fuera un objeto que se adjunta a una clase, y así poder procesar y probar nuevos datos o nuevas bases de datos. Se utilizó el paradigma '*MVC*', representando '*M*' al modelo o datos que se van a procesar, '*C*' al controlador o lógica que se del funcionamiento correcto del aplicativo. Se programa la lógica una vez en forma de funciones y luego se llama desde las vistas para que ejecuten el código, esto beneficia a la

velocidad de la aplicación y a la depuración de errores por parte del programador. En los Códigos 10 y 11 se muestra un ejemplo del código utilizado en forma de función y una muestra de la llamada en las vistas mediante funciones respectivamente.

Código 10

Programación de una función para establecer la lógica de la segunda pestaña.

```
function cargarprocesar(app)
        try
            load MIXED_AUTOENCODER_MODEL.mat; % Carga el modelo elegido de SA
            [file,path] = uigetfile('*.mat'); % Recibe un archivo de base datos
que se encuentra en la computadora
            app.PathTextArea.Value = fullfile(path,file); % Agrega la ruta de
ubicación del archivo elegido
            load(fullfile(path,file)); % Carga el archivo segun la ruta elegida
            X2=X2';
            Y2=Y2';
            Ycat=categorical(Y2);
            panel1 = app.StackedAutoencoderPanel;
            tabla1 = app.UITable;
            y1 = stackednet(X2); % Prueba el autoencoder para clasificar los datos
de X2 (3000 señales)
            y2 = round(y1);
            ycate = categorical(y2);
            cm1 = confusionchart(Ycat, ycate,...
                'ColumnSummary','column-normalized', ...
                'RowSummary','row-normalized',...
                'DiagonalColor','g','OffDiagonalColor','r',...
                'Title','Classification w/ Test Signals',...
                'Parent',panel1,...
                'FontName','Bookman Old Style',FontSize=11);% Grafico de matriz
confusion
            matrix1=cm1.NormalizedValues;
            TP1=matrix1(1,1);
            TN1=matrix1(2,2);
            FP1=matrix1(1,2);
            FN1=matrix1(2,1);
            A1= ((TP1+TN1)/(TP1+TN1+FP1+FN1))*100; %Formula Exactitud
            P1= (TP1/(TP1+FP1))*100;            %Formula Precision
            S1= (TN1/(TN1+FP1))*100;            %Formula Especificidad
            R1= (TP1/(TP1+FN1))*100;            %Formula Sensibilidad
            BER1= 1-((R1+S1)/200);             %Formula BER
            Data1 = [A1 P1 S1 R1 BER1];
            set(tabla1,'Data',Data1);
```

```
            load MIXED_DNN_MODEL.mat; % Carga el modelo elegido de DNN
            load(fullfile(path,file)); % Carga el archivo segun la ruta elegida
            panel2 = app.DeepNeuralNetworkPanel;
            tabla2 = app.UITable;
            Y2=categorical(Y2);
            YPred = classify(net, X2);
            cm2 = confusionchart(YPred, Y2,...
                'ColumnSummary','column-normalized', ...
                'RowSummary','row-normalized',...
                'DiagonalColor','g','OffDiagonalColor','r',...
                'Title','Classification w/ Test Signals',...
                'Parent',panel2,...
                'FontName','Bookman Old Style',FontSize=11);% Grafico de matriz
confusion
            matrix2=cm2.NormalizedValues;
            TP2=matrix2(1,1);
            TN2=matrix2(2,2);
            FP2=matrix2(1,2);
            FN2=matrix2(2,1);
            A2= ((TP2+TN2)/(TP2+TN2+FP2+FN2))*100; %Formula Exactitud
            P2= (TP2/(TP2+FP2))*100;            %Formula Precision
            S2= (TN2/(TN2+FP2))*100;            %Formula Especificidad
            R2= (TP2/(TP2+FN2))*100;            %Formula Sensibilidad
            BER2= 1-((R2+S2)/200);             %Formula BER
            newData = [A2 P2 S2 R2 BER2];
            oldData = get(tabla2,'Data');
            Data2 = [oldData;newData];
            set(tabla2,'Data',Data2);
        catch
            msgbox("No se seleccionó ningún archivo o un archivo" + ...
                " correcto de Base de datos.")
            app.PathTextArea.Value = 'Seleccione un archivo...';
        end
    end
```

Nota. En la función presentada se utiliza código propio de *MATLAB*® para cargar archivos, cambiar de tipo de datos utilizados necesarios para el funcionamiento correcto del programa y lógica de programación general con operaciones para obtener los resultados esperados, todo ello se encuentra dentro de un `try-catch` utilizado para el manejo de excepciones y errores que mediante acciones del usuario puedan romper el aplicativo. Para obtener más información sobre el funcionamiento es recomendable leer los comentarios en las líneas más importantes que explican lo que hacen. Se utilizan una función similar para la tercera pestaña.

Código 11

Ejemplo de programación de la lógica llevada al componente mediante el llamado de funciones.

```
% Callbacks that handle component events
    methods (Access = private)

        % Button pushed function: CARGARPROCESARButton
        function CARGARPROCESARButtonPushed(app, event)
            cargarprocesar(app);
        end

        % Button pushed function: CARGARPROCESARButton_2
        function CARGARPROCESARButton_2Pushed(app, event)
            cargarprocesarwavelet(app);
        end
    end
```

Nota. En la función `CARGARPROCESARButtonPushed` se utiliza dos parámetros: `app` y `event`, que utiliza el paradigma de la programación orientada a objetos, llama a app que es el aplicativo en sí, con su lógica programada en funciones del controlador antes mencionado (mostrado en el Código 10) y se llama a un evento, que es lo que ocurre cuando el usuario realiza una acción con un componente en la interfaz, en este caso el evento es cuando se presiona un botón. Simplemente al utilizar MVC se llama a la lógica del controlador en cada componente, ahorrando así recursos y código además de optimizar la aplicación.

Luego de finalizado tanto el diseño del aplicativo y resuelto la lógica con programación, se procede a desplegar la aplicación como parte de la interfaz *MATLAB*®, generando un instalable que funciona con una versión de *MATLAB*® desde R2022a en adelante. El archivo generado se puede instalar desde la pestaña '*APP*' de la interfaz de usuario de *MATLAB*® e incluye todas las descripciones, archivos y dependencias que necesita para que funcione correctamente. En la Figura 25 se muestra los detalles del funcionamiento del aplicativo computacional desarrollado.

Capítulo IV

Pruebas y Resultados

En el presente capítulo se detallan los resultados que conllevaron el desarrollo, entrenamiento, pruebas y elección de los mejores modelos de DNN y SA como parte esencial de la clasificación de microsismos volcánicos de tipo LP y VT del Cotopaxi. A continuación, se redacta los ensayos realizados para SA, DNN y el aplicativo computacional tanto para los modelos que incluyen *Wavelet* como los que no.

Pruebas con SA

Como mejores parámetros se eligieron 20 resultados de los valores óptimos en las métricas de evaluación del SA, en la Tabla 6 se presenta lo mencionado.

Tabla 6

Mejores parámetros de la función trainAutoencoder para la DB 'Real'.

			Parámetros			Test Autoencoder con 30% de las señales de la base de datos 'Real'				
Pruebas	Neuronas	MaxEpochs	L2Weight Regularization	Sparsity Regularization	Sparsity Proportion	Exactitud [%]	Precisión [%]	Especificidad [%]	Sensibilidad [%]	BER
1	104	300	0.0060	7.1340	0.3000	95.00	93.33	93.55	96.55	0.0495
2	116	338	0.0051	5.9590	0.0571	100.00	100.00	100.00	100.00	0.0000
3	102	366	0.0050	8.8620	0.4584	96.67	93.33	93.75	100.00	0.0312
4	108	320	0.0068	8.8620	0.5417	96.67	96.67	96.67	96.67	0.0333
5	96	390	0.0086	8.5390	0.6492	95.00	93.33	93.55	96.55	0.0495
6	110	408	0.0090	8.2100	0.6816	96.67	96.67	96.67	96.67	0.0333
7	115	404	0.0079	8.5490	0.6353	96.67	93.33	93.75	100.00	0.0312
8	119	408	0.0075	8.4730	0.5226	96.67	93.33	93.75	100.00	0.0312
9	106	402	0.0087	8.5560	0.6871	96.67	96.67	96.67	96.67	0.0333
10	120	408	0.0077	6.7930	0.6648	96.67	96.67	96.67	96.67	0.0333
11	102	408	0.0089	9.3044	0.5876	95.00	93.33	93.55	96.55	0.0495
12	92	400	0.0087	4.1600	0.6900	96.67	93.33	93.75	100.00	0.0312
13	93	402	0.0081	2.7271	0.5910	95.00	93.33	93.55	96.55	0.0495
14	110	400	0.0071	6.8846	0.6673	95.00	93.33	93.55	96.55	0.0495
15	104	392	0.0071	1.4815	0.6749	96.67	93.33	93.75	100.00	0.0312
16	101	400	0.0078	2.5213	0.6801	95.00	93.33	93.55	96.55	0.0495
17	116	382	0.0067	6.2163	0.6599	96.67	93.33	93.75	100.00	0.0312
18	118	386	0.0071	2.0325	0.6427	96.67	93.33	93.75	100.00	0.0312
19	120	308	0.0089	8.0756	0.5456	95.00	93.33	93.55	96.55	0.0495
20	98	396	0.0080	4.3933	0.6307	95.00	93.33	93.55	96.55	0.0495

Nota. En la tabla se puede apreciar que los mejores parámetros escogidos son los de color amarillo, estos dieron los mejores resultados de las métricas de evaluación al entrenar el SA con la base de datos 'Real'.

Debido a la desigualdad entre los microsismos volcánicos LP y VT que tiene la base de datos 'Real' se realiza otra prueba conocida como *folds*, esta consiste en variar el rango para escoger 101 LP diferentes para cada prueba y así verificar que los parámetros escogidos de la

Tabla 6 son los adecuados para entrenar el SA. Se realizan 4 ensayos de *folds* y en todas las pruebas realizadas el valor del BER es similar, con lo mencionado anteriormente, los resultados se presentan en la Tabla 7.

***Tabla* 7**

Prueba de Folds con los mejores parámetros de la función trainAutoencoder para la base de datos 'Real'.

Parámetros						Test Autoencoder con 30% de las señales de la base de datos 'Real'				
Pruebas	Neuronas	MaxEpochs	L2Weight Regularization	Sparsity Regularization	Sparsity Proportion	Exactitud [%]	Precisión [%]	Especificidad [%]	Sensibilidad [%]	BER
1	116	338	0.0051	5.9590	0.0571	96.67	93.33	93.75	100.00	0.0312
2	116	338	0.0051	5.9590	0.0571	98.33	96.67	96.77	100.00	0.0161
3	116	338	0.0051	5.9590	0.0571	100.00	100.00	100.00	100.00	0.0000
4	116	338	0.0051	5.9590	0.0571	98.33	96.67	96.77	100.00	0.0161

Nota. Como se observa en la tabla al realizar las 4 pruebas de *folds* los resultados obtenidos de las métricas de evaluación son óptimos para el entrenamiento del SA.

Posteriormente se obtuvo la media de las métricas de rendimiento de las pruebas con *folds* ejecutadas, debido a que la media proporciona información sobre la tendencia central de las métricas de rendimiento. Calculando esta medida, se puede determinar el valor medio o típico en torno al cual se agrupan las métricas de rendimiento. Esta información ayuda a resumir los datos y a proporcionar un único valor representativo que refleje la tendencia general, en la Tabla 8 se muestra la media de los *folds* antes mencionados.

Tabla 8

Media de las métricas de rendimiento de las pruebas de folds.

	Métricas de Rendimiento				
	Exactitud [%]	Precisión [%]	Especificidad [%]	Sensibilidad [%]	BER
Media	98.33	96.67	96.82	100	0.0159

El mismo procedimiento mencionado anteriormente se utiliza para las demás bases de datos, la segunda prueba es con la base de datos ‘Sintética’, los resultados obtenidos se muestran en la Tabla 9.

Tabla 9

Mejores parámetros de la función trainAutoencoder para la base de datos ‘Sintética’.

	Parámetros					Test Autoencoder con 30% de las señales de la base de datos ‘Sintética’				
Pruebas	Neuronas	MaxEpochs	L2Weight Regularization	Sparsity Regularization	Sparsity Proportion	Exactitud [%]	Precisión [%]	Especificidad [%]	Sensibilidad [%]	BER
1	104	300	0.0060	7.1340	0.3000	99.97	100.00	100.00	99.93	0.0003
2	116	338	0.0051	5.9590	0.0571	99.97	100.00	100.00	99.93	0.0003
3	102	366	0.0050	8.8620	0.4584	99.93	100.00	100.00	99.87	0.0007
4	108	320	0.0068	8.8620	0.5417	99.97	100.00	100.00	99.93	0.0003
5	96	390	0.0086	8.5390	0.6492	99.97	100.00	100.00	99.93	0.0003
6	110	408	0.0090	8.2100	0.6816	99.97	100.00	100.00	99.93	0.0003
7	115	404	0.0079	8.5490	0.6353	99.97	100.00	100.00	99.93	0.0003
8	119	408	0.0075	8.4730	0.5226	99.97	100.00	100.00	99.93	0.0003
9	106	402	0.0087	8.5560	0.6871	99.97	100.00	100.00	99.93	0.0003
10	120	408	0.0077	6.7930	0.6648	99.97	100.00	100.00	99.93	0.0003
11	102	408	0.0089	9.3044	0.5876	99.97	100.00	100.00	99.93	0.0003
12	92	400	0.0087	4.1600	0.6900	99.97	100.00	100.00	99.93	0.0003
13	93	402	0.0081	2.7271	0.5910	99.97	100.00	100.00	99.93	0.0003
14	110	400	0.0071	6.8846	0.6673	99.97	100.00	100.00	99.93	0.0003
15	104	392	0.0071	1.4815	0.6749	99.97	100.00	100.00	99.93	0.0003
16	101	400	0.0078	2.5213	0.6801	99.97	100.00	100.00	99.93	0.0003
17	116	382	0.0067	6.2163	0.6599	99.97	100.00	100.00	99.93	0.0003
18	118	386	0.0071	2.0325	0.6427	99.97	100.00	100.00	99.93	0.0003
19	120	308	0.0089	8.0756	0.5456	99.97	100.00	100.00	99.93	0.0003
20	98	396	0.0080	4.3933	0.6307	99.97	100.00	100.00	99.93	0.0003

Nota. Como se observa en la tabla los resultados obtenidos de las métricas de evaluación son los mismo en 19 pruebas, por lo cual se escogió de manera objetiva los parámetros de la prueba 9, estos son usados para entrenar el SA con la base de datos ‘Sintética’.

La siguiente prueba para elegir los mejores parámetros que permitan entrenar el SA es con la base de datos ‘Mixta’, de igual manera que las anteriores pruebas se realiza el *test* y se obtiene de resultados lo que muestra la Tabla 10.

Tabla 10

Mejores parámetros de la función trainAutoencoder para la base de datos ‘Mixta’.

Parámetros						Test Autoencoder con 30% de las señales de la base de datos ‘Mixta’				
Pruebas	Neuronas	MaxEpochs	L2Weight Regularization	Sparsity Regularization	Sparsity Proportion	Exactitud [%]	Precisión [%]	Especificidad [%]	Sensibilidad [%]	BER
1	104	300	0.0060	7.1340	0.3000	99.73	99.53	99.54	99.93	0.0027
2	116	338	0.0051	5.9590	0.0571	99.47	99.33	99.34	99.60	0.0053
3	102	366	0.0050	8.8620	0.4584	99.60	99.47	99.47	99.73	0.0040
4	108	320	0.0068	8.8620	0.5417	99.57	99.47	99.47	99.67	0.0043
5	96	390	0.0087	8.5390	0.6492	99.63	99.73	99.73	99.53	0.0037
6	110	408	0.0090	8.2100	0.6816	99.63	99.67	99.67	99.60	0.0037
7	115	404	0.0079	8.5490	0.6353	99.37	99.33	99.33	99.40	0.0063
8	119	408	0.0075	8.4730	0.5226	99.47	99.27	99.27	99.67	0.0053
9	106	402	0.0087	8.5560	0.6871	99.67	99.60	99.60	99.73	0.0033
10	120	408	0.0077	6.7930	0.6648	99.43	99.33	99.34	99.53	0.0057
11	102	408	0.0089	9.3044	0.5876	99.63	99.60	99.60	99.67	0.0037
12	92	400	0.0087	4.1600	0.6900	99.70	99.67	99.67	99.73	0.0030
13	93	402	0.0081	2.7271	0.5910	99.60	99.47	99.47	99.73	0.0040
14	110	400	0.0071	6.8846	0.6673	99.63	99.67	99.67	99.60	0.0037
15	104	392	0.0071	1.4815	0.6749	99.47	99.40	99.40	99.53	0.0053
16	101	400	0.0078	2.5213	0.6801	99.57	99.40	99.40	99.73	0.0043
17	116	382	0.0067	6.2163	0.6599	99.47	99.33	99.34	99.60	0.0053
18	118	386	0.0071	2.0325	0.6427	99.63	99.60	99.60	99.67	0.0037
19	120	308	0.0089	8.0756	0.5456	99.60	99.60	99.60	99.60	0.0040
20	98	396	0.0080	4.3933	0.6307	99.70	99.60	99.60	99.80	0.0030

Nota. Los mejores parámetros escogidos para entrenar el SA con la base de datos ‘Mixta’ son los de la prueba 1.

Para las dos bases de datos faltantes se realiza el mismo procedimiento que se utilizó para las bases de datos ‘Real’, ‘Sintética’ y ‘Mixta’, en la Tabla 11 se muestra los mejores resultados de los parámetros de la función trainAutoencoder para la base de datos ‘Mixta Wavelet Daubechies’ y en la Tabla 12 están los resultados de la base de datos ‘Mixta Wavelet Symlets’.

Tabla 11

Mejores parámetros de la función trainAutoencoder para la base de datos 'Mixta Wavelet Daubechies'.

Parámetros						Test Autoencoder con 30% de las señales de la base de datos 'Mixta Wavelet Daubechies'				
Pruebas	Neuronas	MaxEpochs	L2Weight Regularization	Sparsity Regularization	Sparsity Proportion	Exactitud [%]	Precisión [%]	Especificidad [%]	Sensibilidad [%]	BER
1	208	300	0.0060	7.1340	0.3	99.57	99.73	99.73	99.40	0.0043
2	232	338	0.0051	5.9590	0.0571	99.23	99.67	99.66	98.81	0.0076
3	204	366	0.0050	8.8620	0.4584	99.53	99.73	99.73	99.34	0.0047
4	216	320	0.0068	8.8620	0.5417	99.60	99.80	99.80	99.40	0.0040
5	192	390	0.0087	8.5390	0.6492	99.53	99.73	99.73	99.34	0.0047
6	220	408	0.0090	8.2100	0.6816	99.50	99.67	99.67	99.34	0.0050
7	230	404	0.0079	8.5490	0.6353	99.43	99.67	99.67	99.20	0.0057
8	238	408	0.0075	8.4730	0.5226	99.57	99.80	99.80	99.34	0.0043
9	212	402	0.0087	8.5560	0.6871	99.53	99.80	99.80	99.27	0.0047
10	240	408	0.0077	6.7930	0.6648	99.53	99.73	99.73	99.34	0.0047
11	204	408	0.0089	9.3044	0.5876	99.53	99.80	99.80	99.27	0.0047
12	184	400	0.0087	4.1600	0.6900	99.57	99.80	99.80	99.34	0.0043
13	186	402	0.0081	2.7271	0.5910	99.53	99.73	99.73	99.34	0.0047
14	220	400	0.0071	6.8846	0.6673	99.50	99.67	99.67	99.34	0.0050
15	208	392	0.0071	1.4815	0.6749	99.53	99.80	99.80	99.27	0.0047
16	202	400	0.0078	2.5213	0.6801	99.50	99.73	99.73	99.27	0.0050
17	232	382	0.0067	6.2163	0.6599	99.47	99.67	99.67	99.27	0.0053
18	236	386	0.0071	2.0325	0.6427	99.53	99.73	99.73	99.34	0.0047
19	240	308	0.0089	8.0756	0.5456	99.53	99.80	99.80	99.27	0.0047
20	196	396	0.0080	4.3933	0.6307	99.53	99.73	99.73	99.34	0.0047

Nota. Los parámetros de la prueba 4 fueron los seleccionados como los mejores para entrenar el SA con la base de datos 'Mixta Wavelet Daubechies'.

Tabla 12

Mejores parámetros de la función trainAutoencoder para la base de datos 'Mixta Wavelet Symlets'.

Parámetros						Test Autoencoder con 30% de las señales de la base de datos 'Mixta Wavelet Symlets'				
Pruebas	Neuronas	MaxEpochs	L2Weight Regularization	Sparsity Regularization	Sparsity Proportion	Exactitud [%]	Precisión [%]	Especificidad [%]	Sensibilidad [%]	BER
1	208	300	0.0060	7.1340	0.3000	99.50	99.53	99.53	99.47	0.0050
2	232	338	0.0051	5.9590	0.0571	99.50	99.53	99.53	99.47	0.0050
3	204	366	0.0050	8.8620	0.4584	99.47	99.53	99.53	99.40	0.0053
4	216	320	0.0068	8.8620	0.5417	99.47	99.53	99.53	99.40	0.0053
5	192	390	0.0087	8.5390	0.6492	99.47	99.53	99.53	99.40	0.0053
6	220	408	0.0090	8.2100	0.6816	99.47	99.53	99.53	99.40	0.0053
7	230	404	0.0079	8.5490	0.6353	99.43	99.53	99.53	99.34	0.0057
8	238	408	0.0075	8.4730	0.5226	99.50	99.53	99.53	99.47	0.0050
9	212	402	0.0087	8.5560	0.6871	99.47	99.53	99.53	99.40	0.0053
10	240	408	0.0077	6.7930	0.6648	99.50	99.53	99.53	99.47	0.0050
11	204	408	0.0089	9.3044	0.5876	99.43	99.47	99.47	99.40	0.0057
12	184	400	0.0087	4.1600	0.6900	99.47	99.60	99.60	99.34	0.0053
13	186	402	0.0081	2.7271	0.5910	99.47	99.53	99.53	99.40	0.0053
14	220	400	0.0071	6.8846	0.6673	99.43	99.47	99.47	99.40	0.0057
15	208	392	0.0071	1.4815	0.6749	99.40	99.53	99.53	99.26	0.0060
16	202	400	0.0078	2.5213	0.6801	99.47	99.47	99.47	99.47	0.0053
17	232	382	0.0067	6.2163	0.6599	99.53	99.67	99.67	99.40	0.0047
18	236	386	0.0071	2.0325	0.6427	99.50	99.53	99.53	99.47	0.0050
19	240	308	0.0089	8.0756	0.5456	99.43	99.53	99.53	99.34	0.0057
20	196	396	0.0080	4.3933	0.6307	99.43	99.47	99.47	99.40	0.0057

Nota. El valor de BER más bajo pertenece a los parámetros de la prueba 17, por lo cual ellos son elegidos para entrenar el SA con la base de datos 'Mixta Wavelet Symlets'.

Después de realizadas todas las pruebas anteriores, se obtuvo un modelo entrenado del algoritmo de SA para cada tipo de bases de datos que se trabajó, cada modelo tiene los mejores parámetros escogidos que se puede apreciar en la Tabla 13. Finalmente se obtuvo un archivo de cada modelo que se exportó para utilizarse con el aplicativo.

Tabla 13

Mejores parámetros del modelo SA entrenado con las distintas bases de datos.

	Parámetros				
Base de Datos	Neuronas	MaxEpochs	L2Weight Regularization	Sparsity Regularization	Sparsity Proportion
Real	116	338	0.0051	5.9590	0.0571
Sintética	106	402	0.0087	8.5560	0.6871
Mixta	104	300	0.0060	7.1340	0.3000
Mixta Wavelet Daubechies	216	320	0.0068	8.8620	0.5417
Mixta Wavelet Symlets	232	382	0.0067	6.2163	0.6599

Nota. Cada modelo de SA entrenado con las distintas bases de datos tiene sus propios parámetros.

Deep Neural Network

Para esta técnica de DL se exporto el único modelo que se programó y fue explicado en el capítulo anterior, en donde se indicó todos los parámetros necesarios para cada función que se utiliza al crear el modelo de DNN, en la Tabla 14 se observa los parámetros para entrenar a la DNN con las distintas bases de datos.

Tabla 14

Parámetros del modelo DNN entrenado con las distintas bases de datos.

Parámetros			
Capa	Neuronas		
Oculta	50	MiniBatchSize (mini lote)	16
Salida	2		

Nota. Un único modelo de DNN entrenado con los parámetros anteriores se utiliza para todas las bases de datos mencionadas en el capítulo anterior.

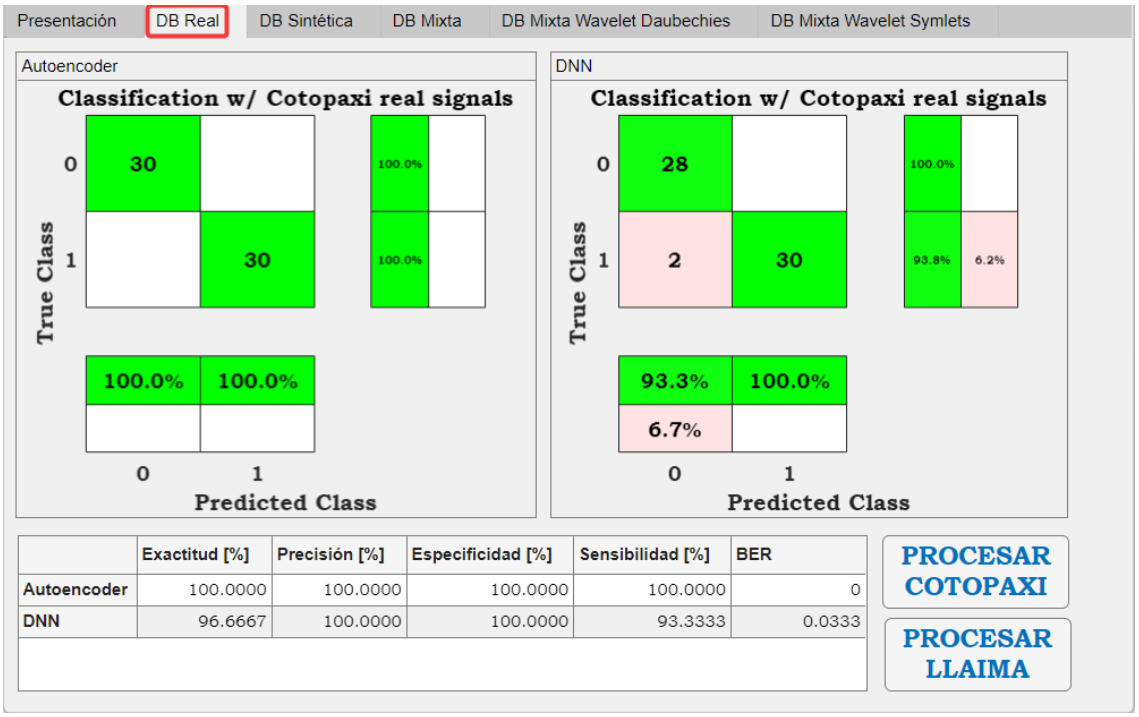
De igual manera que el SA se obtuvo un archivo del modelo de la DNN que se exportó para utilizarse en las pruebas del aplicativo desarrollado.

Elección del mejor modelo de cada algoritmo

Para la elección del modelo con mejores resultados para bases de datos que incluyen Wavelet y las que no, se procedió a probar los modelos con el 30.00% de los datos que no se utilizaron para el entrenamiento, o, dicho de otra manera, que se reservaron para probar los modelos. En el caso de los modelos entrenados sin *Wavelet*, a través de las bases de datos Real, Sintética y Mixta, estos son los resultados obtenidos para los datos de prueba de cada uno, mostrando la matriz de confusión además de las métricas de rendimiento tanto para SA y DNN:

Figura *30*

Resultados para los modelos entrenados con la base de datos Real



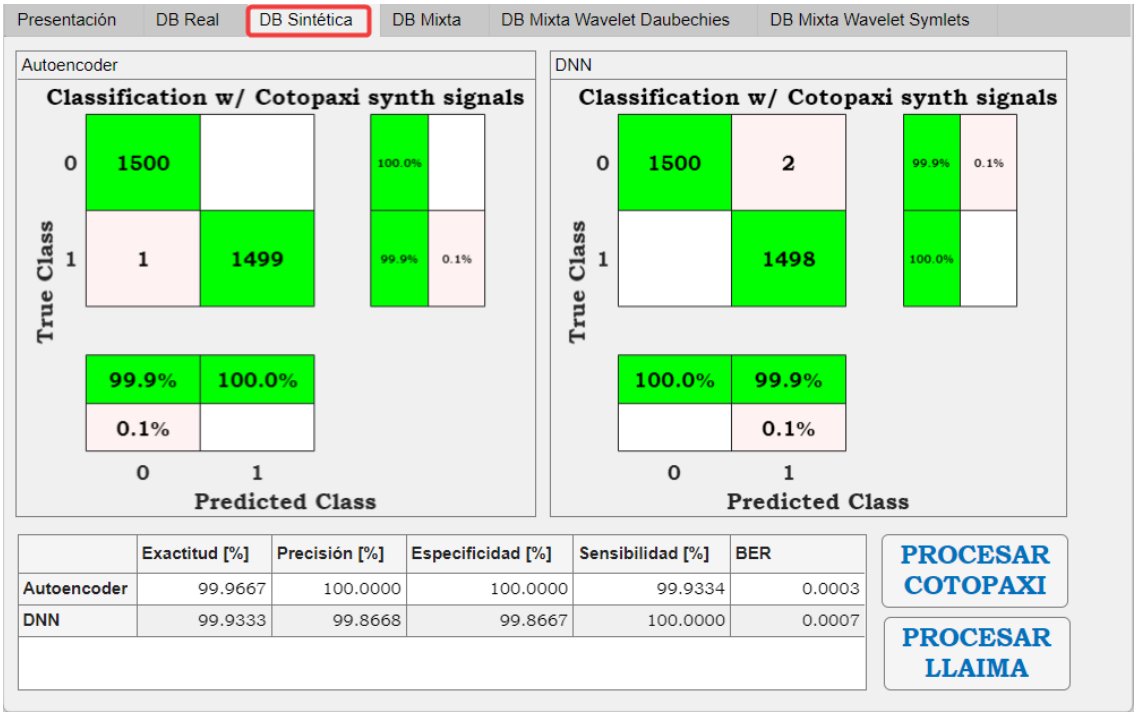
Nota. Los datos de *test* para esta prueba son 60 microsismos volcánicos en total, 30 del tipo LP, representados por ‘0’ y 30 microsismos volcánicos del tipo VT representados con el número ‘1’, esto por la limitación de VT en la base de datos ‘Real’ del volcán Cotopaxi. Se aprecia que los resultados son afectados por el número de datos para entrenamiento y prueba.

Como se observa en la Figura 30, en el caso del modelo de SA se aprecia un BER de 0.0000 y un 100.00% de exactitud, precisión, especificidad y sensibilidad. Estos parámetros indican que cada modelo de DL clasifica de manera perfecta, pese a contar con tan pocos datos para ese cometido. Sin embargo, la falta de datos para entrenamiento y prueba se aprecia mejor en los resultados del modelo de DNN, donde pese a fallar muy poco en la clasificación, donde se equivoca en apenas dos señales del tipo ‘LP’ como se aprecia en la matriz de confusión de la derecha, decrece considerablemente en su exactitud y sensibilidad lo

que afecta al BER que finalmente es de 0.0333, superior al 0.0100 que tiene como objetivo la investigación. Por su baja confiabilidad al contar con muy pocos datos, por su escaso rendimiento para el caso de DNN y por no cumplir con uno de los objetivos establecidos para el proyecto, estos modelos quedan descartados.

Figura 31

Resultados para los modelos entrenados con la base de datos Sintética



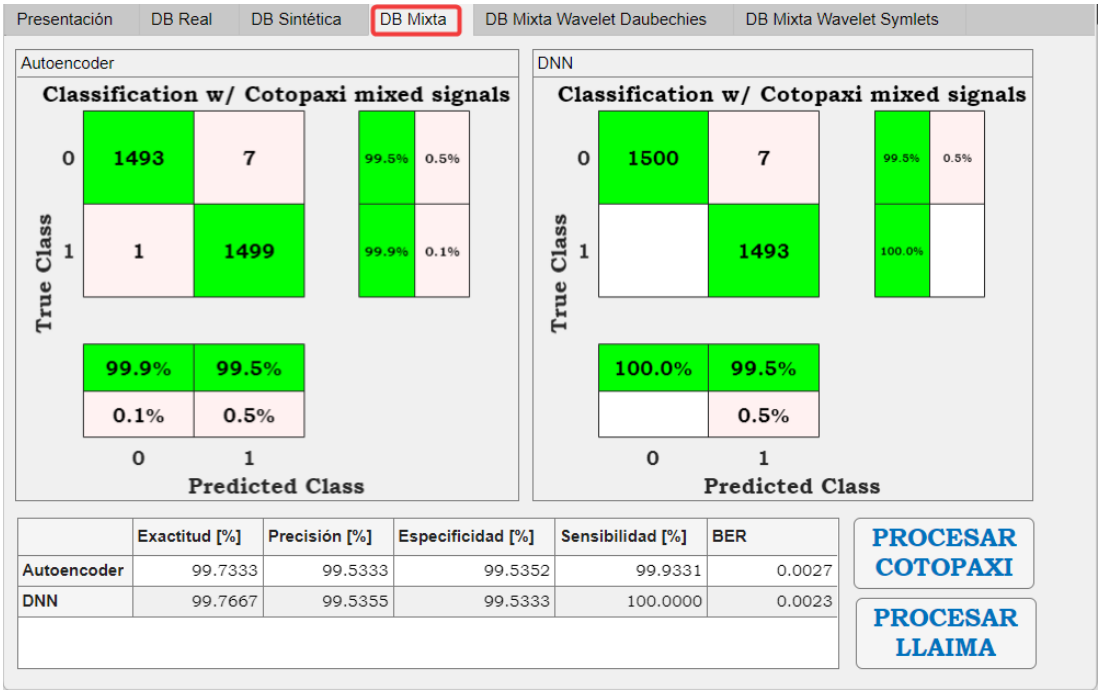
	Exactitud [%]	Precisión [%]	Especificidad [%]	Sensibilidad [%]	BER
Autoencoder	99.9667	100.0000	100.0000	99.9334	0.0003
DNN	99.9333	99.8668	99.8667	100.0000	0.0007

Nota. Los datos de *test* para esta prueba son 3000 microsismos volcánicos en total, 1500 del tipo LP, representados por '0' y 1500 microsismos volcánicos del tipo VT representados con el número '1', todos son microsismos volcánicos sintéticos del Cotopaxi generados mediante CGAN, basado en el trabajo de (Galarza Cruz & Vega Vergara, 2022). Se aprecia que los resultados son afectados por tratarse de microsismos volcánicos ideales, tanto para entrenamiento como para prueba.

Como se aprecia en la Figura 31, para el modelo de DNN se tiene un BER de 0.0007, una sensibilidad del 100.00%, especificidad y precisión prácticamente iguales con valores de 99.86% y 99.86% respectivamente, finalmente, una exactitud del 99.93%. En el caso del modelo de SA se observa un BER de 0.0003, 99.96% de exactitud, un 100.00% de precisión y especificidad, y 99.93% de sensibilidad. Estos parámetros indican que los modelos clasifican de manera casi ideal, con un buen número de datos para entrenamiento y prueba tanto para DNN como SA. No obstante, debido a que ambos modelos fueron entrenados con datos sintéticos generados mediante CGAN, que representan los microsismos volcánicos de manera ideal, se previó que existiera un sobreajuste a los datos de entrenamiento, y dados los resultados casi perfectos en la clasificación, ello se corrobora. Se descartan estos modelos, para evitar que exista así algún tipo de sesgo y sobreajuste de datos en futuros estudios que tomen como base al presente.

Figura 32

Resultados para los modelos entrenados con la base de datos Mixta



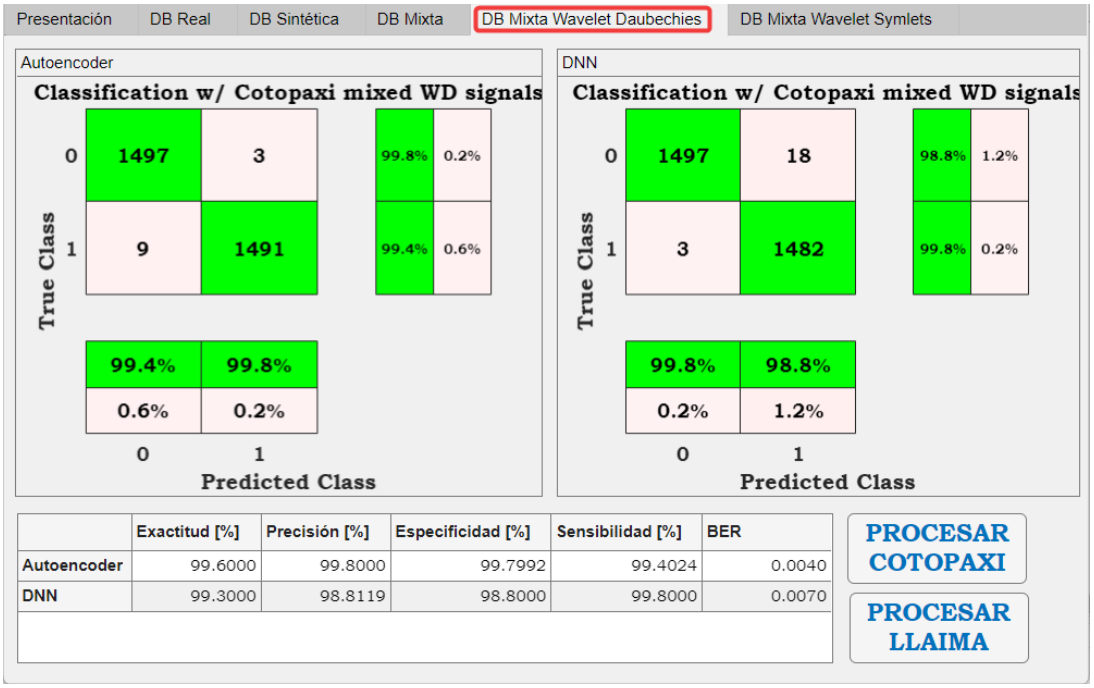
	Exactitud [%]	Precisión [%]	Especificidad [%]	Sensibilidad [%]	BER
Autoencoder	99.7333	99.5333	99.5352	99.9331	0.0027
DNN	99.7667	99.5355	99.5333	100.0000	0.0023

Nota. Los datos de *test* para esta prueba son 3000 microsismos volcánicos en total, 1500 del tipo LP, representados por ‘0’ y 1500 microsismos volcánicos del tipo VT representados con el número ‘1’, y los conforman microsismos volcánicos aleatoriamente ordenados entre reales y sintéticos del volcán Cotopaxi. Se aprecia que los resultados para las métricas de rendimiento y las matrices de confusión se ven mucho más concisos y cercanos entre sí para ambos modelos, además de tratarse de una excelente clasificación de microsismos volcánicos.

Como se aprecia en la Figura 32, para el modelo de SA se tiene una exactitud del 99.73%, una precisión de 99.53%, especificidad del 99.53%, sensibilidad de 99.93% y un BER de 0.0027. En el caso del modelo de DNN se observa una exactitud del 99.76%, una precisión de 99.53%, sensibilidad del 100.00%, especificidad de 99.53%, y un BER de 0.0023. Estos parámetros indican excelentes resultados en la clasificación de microsismos volcánicos para ambos modelos, con resultados que favorecen de forma ligera a DNN sobre SA. Estos modelos de DL son los que mejor representan y se ajustan a la realidad del volcán Cotopaxi, dado que cuentan con una cantidad de datos de entrenamiento suficientes para evitar el sobre ajuste para modelos de DL, como también con una cantidad de microsismos volcánicos reales aleatoria entre los datos de entrenamiento, así como en los datos de prueba. Dado que los excelentes resultados para clasificación de microsismos volcánicos obtenidos en estos modelos de SA y DNN son tan congruentes para cada una de las métricas de rendimiento, se optó por tomarlos como los mejores entre los obtenidos, es decir son los que se utilizarán en el aplicativo para clasificación de todo tipo de microsismos volcánicos procesados sin trsnsformada *Wavelet*.

Figura 33

Resultados para los modelos entrenados con una base de datos Mixta con Wavelet Daubechies



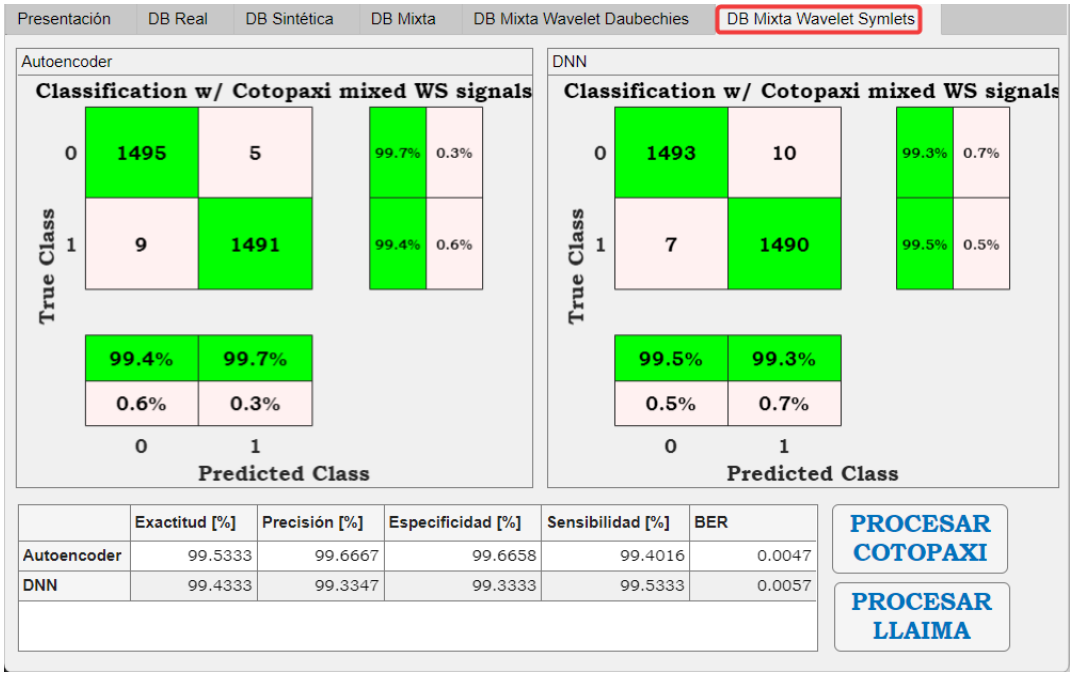
Nota. Los datos de *test* para esta prueba son 3000 microsismos volcánicos en total, 1500 del tipo LP, representados por '0' y 1500 microsismos volcánicos del tipo VT representados con el número '1', y la conforman microsismos volcánicos aleatoriamente ordenados entre reales y sintéticos del volcán Cotopaxi procesadas además con la transformada *Wavelet* de *Daubechies*. Se aprecia que los resultados para las métricas de rendimiento y las matrices de confusión se ven dispares entre ambos modelos de DL y con más errores para el modelo de DNN.

Como aclaración, para ambos modelos de DL que prueban datos procesados con *Wavelet*, se utilizó una base de datos Mixta con idénticas características a la anterior mencionada con la única diferencia de que está procesada además con un algoritmo de transformada *Wavelet*, siendo una transformada *Wavelet* de *Daubechies* la utilizada en la

Figura 33. Para el modelo de DNN se tiene una exactitud del 99.30%, una precisión de 98.81%, especificidad del 98.80%, sensibilidad de 99.80% y un BER de 0.0070. En el caso del modelo de SA se observa una exactitud del 99.60%, una precisión de 99.80%, especificidad de 99.79%, sensibilidad de 99.40% y un BER de 0.0040. Estos parámetros indican excelentes resultados en la clasificación de microsismos volcánicos para ambos modelos, con resultados que favorecen a SA sobre DNN. Sin embargo, debido a que ambos modelos de DL fueron entrenados con el mismo *set* de datos, no deberían variar de tal manera como lo hacen en los resultados, lo que podría indicar alguna inconsistencia con un modelo u otro en particular. Se descartan estos modelos, para evitar que exista así algún tipo de sesgo y sobreajuste de datos en futuros estudios que tomen como base al presente, evitando así que se obtengan resultados no válidos, si se prueba los modelos con señales procesadas con *Wavelet* distintas a las del volcán Cotopaxi.

Figura 34

Resultados para los modelos entrenados con una base de datos Mixta con Wavelet Symlets



Nota. Los datos de *test* para esta prueba son 3000 microsismos volcánicos en total, 1500 del tipo LP, representados por ‘0’ y 1500 microsismos volcánicos del tipo VT representados con el número ‘1’, y la conforman microsismos volcánicos aleatoriamente ordenados entre reales y sintéticos del volcán Cotopaxi procesados además con la transformada *Wavelet* de *Symlets*. Se aprecia que los resultados para las métricas de rendimiento y las matrices de confusión se ven mucho más concisos y cercanos entre sí para ambos modelos de DL, además de tratarse de una excelente clasificación de microsismos volcánicos.

Como se aprecia en la Figura 34, para el modelo de SA se tiene una exactitud del 99.53%, una precisión de 99.66%, especificidad del 99.67%, sensibilidad de 99.40% y un BER de 0.0047. En el caso del modelo de DNN se observa una exactitud del 99.43%, una precisión de 99.33%, sensibilidad del 99.53%, especificidad de 99.33%, y un BER de 0.0057. Estos parámetros indican excelentes resultados en la clasificación de señales para ambos modelos, con resultados que favorecen de forma ligera a SA sobre DNN. Estos modelos de DL son los que mejor representan y se ajustan a microsismos volcánicos del Cotopaxi procesados con transformada *Wavelet*, dado que cuentan con una cantidad de datos con *Wavelet* de entrenamiento suficientes para evitar el sobre ajuste para modelos de DL, como también con una cantidad de microsismos volcánicos reales aleatoria entre los datos de entrenamiento, así como en los datos de prueba. Dado que los excelentes resultados para clasificación de microsismos volcánicos obtenidos en estos modelos de SA y DNN son más congruentes entre sí con respecto al anterior para cada una de las métricas de rendimiento, se optó por tomarlos como los mejores entre los obtenidos, es decir son los que se utilizarán en el aplicativo para clasificación de todo tipo de microsismos volcánicos procesados con *Wavelet*.

Tabla 15

Mejores resultados de los modelos de DL con las distintas bases de datos.

Base de datos	Modelo DL	Exactitud [%]	Precisión [%]	Especificidad [%]	Sensibilidad [%]	BER
DB Real	SA	100.00	100.00	100.00	100.00	0.0000
DB Real	DNN	96.66	100.00	100.00	93.33	0.0333
DB Sintética	SA	99.96	100.00	100.00	99.93	0.0003
DB Sintética	DNN	99.93	99.86	99.86	100.00	0.0007
DB Mixta	SA	99.73	99.53	99.53	99.93	0.0027
DB Mixta	DNN	99.76	99.53	99.53	100.00	0.0023
DB Mixta Wavelet Daubechies	SA	99.60	99.80	99.79	99.40	0.0040
DB Mixta Wavelet Daubechies	DNN	99.30	98.81	98.80	99.80	0.0070
DB Mixta Wavelet Symlets	SA	99.53	99.66	99.66	99.40	0.0047
DB Mixta Wavelet Symlets	DNN	99.43	99.33	99.33	99.53	0.0057

Nota. Como se puede apreciar el modelo con menor BER pertenece al SA entrenado con la DB Real, la cual contiene 257 muestras por cada microsismo volcánico. Para las bases de datos que contienen 514 muestras por cada microsismo volcánico, el SA obtuvo el menor BER con un valor de 0.0040, este fue entrenado con la DB Mixta *Wavelet Daubechies*.

Resultados obtenidos con el aplicativo

Pruebas con bases de datos del volcán Cotopaxi

Para este tipo de pruebas se utilizó cinco tipos de bases de datos: Real, Sintética y Mixta cada una con 257 muestras por cada microsismo volcánico procesado en el dominio de la frecuencia, y dos bases de datos Mixtas con 514 muestras por cada microsismo volcánico agregadas las transformadas *Wavelet* discretas de *Daubechies* y *Symlets* respectivamente,

procesadas en el dominio de la frecuencia, junto a las características mencionadas en el capítulo anterior para cada una de ellas.

Datos de prueba DB Real

Figura 35

Resultados para los datos de prueba de la base de datos Real del volcán Cotopaxi

	Exactitud [%]	Precisión [%]	Especificidad [%]	Sensibilidad [%]	BER
Autoencoder	100.0000	100.0000	100.0000	100.0000	0
DNN	98.3333	96.7742	96.6667	100.0000	0.0167

Nota. Los datos de prueba constan de 60 microsismos volcánicos en total, 30 del tipo LP, representados por ‘0’ y 30 microsismos volcánicos del tipo VT representados con el número ‘1’, de la base de datos reales del volcán Cotopaxi.

En la Figura 35 se puede apreciar que el SA alcanza el 100.00% de exactitud, precisión, especificidad, sensibilidad y un BER de 0.0000, lo que nos indica que este modelo de DL está perfectamente entrenado para clasificar microsismos volcánicos del volcán Cotopaxi. Mientras

que la DNN se obtuvo varios valores de las métricas de rendimiento que son aceptables para la clasificación, sin embargo, en esta prueba realizada con la base de datos Real el mejor modelo para clasificar es el SA.

Datos de prueba DB Sintética

Figura 36

Resultados para los datos de prueba de la base de datos Sintética del volcán Cotopaxi

Nota. Los datos de prueba constan de 3000 microsismos volcánicos en total, 1500 del tipo LP, representados por ‘0’ y 1500 microsismos volcánicos del tipo VT representados con el número ‘1’, de la base de datos sintéticos del volcán Cotopaxi.

Al realizar la prueba con la DB Sintética se puede observar que los dos modelos de DL entrenados obtuvieron valores iguales en las métricas de rendimiento, esto se puede apreciar

en la Figura 36, lo que nos indica que tanto el SA como la DNN están muy bien entrenados para clasificar esta base de datos utilizada.

Datos de prueba DB Mixta

Figura 37

Resultados para los datos de prueba de la base de datos Mixta del volcán Cotopaxi

Nota. Los datos de prueba constan de 3000 microsismos volcánicos en total, 1500 del tipo LP, representados por ‘0’ y 1500 microsismos volcánicos del tipo VT representados con el número ‘1’, de la base de datos mixta del volcán Cotopaxi.

Como se observa en la Figura 37, los mejores resultados de las métricas de rendimiento se obtuvieron con la DNN, mientras que el SA presento resultados casi simulares con

pequeñas variaciones en el porcentaje de las métricas de rendimiento, por lo tanto, la DNN es el modelo indicado para clasificar los LP y VT de la base de datos Mixta.

Datos de prueba DB Mixta con Wavelet Daubechies

Figura 38

Resultados para los datos de prueba de la base de datos Mixta con Wavelet Daubechies del volcán Cotopaxi

	Exactitud [%]	Precisión [%]	Especificidad [%]	Sensibilidad [%]	BER
Autoencoder	99.8000	99.9333	99.9332	99.6676	0.0020
DNN	99.7333	99.6011	99.6000	99.8667	0.0027

Nota. Los datos de prueba constan de 3000 microsismos volcánicos en total, 1500 del tipo LP, representados por ‘0’ y 1500 microsismos volcánicos del tipo VT representados con el número ‘1’, de la base de datos mixta del volcán Cotopaxi con *Wavelet Daubechies*.

El mejor BER obtenido en esta prueba es de 0.0020, el cual pertenece al modelo de SA, este se presenta en la Figura 38, donde también se aprecia los valores de las otras métricas de

rendimiento tanto para la DNN y el SA. En esta ocasión el SA superó a la DNN y es el modelo indicado para utilizarlo en la clasificación de las señales pertenecientes a la base de datos Mixta con *Wavelet* Daubechies.

Datos de prueba DB Mixta con Wavelet Symlets

Figura 39

Resultados para los datos de prueba de la base de datos Mixta con Wavelet Symlets del volcán Cotopaxi

Nota. Los datos de prueba constan de 3000 microsismos volcánicos en total, 1500 del tipo LP, representados por ‘0’ y 1500 microsismos volcánicos del tipo VT representados con el número ‘1’, de la base de datos mixta del volcán Cotopaxi con *Wavelet Symlets*.

Al realizar la prueba de los modelos de DL entrenados con la DB Mixta con Wavelet Symlets se puede observar en la Figura 39, que el SA logro tener mejores resultados en las

métricas de rendimiento, donde su BER es menor con respecto al de la DNN, aseverando que el SA es el indicado para la clasificación de los eventos LP y VT con esta base de datos utilizada.

Tabla 16

Mejores resultados del Volcán Cotopaxi en el aplicativo de los modelos de DL con las distintas bases de datos.

Base de datos	Modelo DL	Exactitud [%]	Precisión [%]	Especificidad [%]	Sensibilidad [%]	BER
DB Real	SA	100.00	100.00	100.00	100.00	0.0000
DB Real	DNN	98.33	96.77	96.66	100.00	0.0167
DB Sintética	SA	100.00	100.00	100.00	100.00	0.0000
DB Sintética	DNN	100.00	100.00	100.00	100.00	0.0000
DB Mixta	SA	99.73	99.53	99.53	99.93	0.0027
DB Mixta	DNN	99.76	99.53	99.53	100.00	0.0023
DB Mixta Wavelet Daubechies	SA	99.80	99.93	99.93	99.66	0.0020
DB Mixta Wavelet Daubechies	DNN	99.73	99.60	99.60	99.86	0.0027
DB Mixta Wavelet Symlets	SA	99.53	99.66	99.66	99.40	0.0047
DB Mixta Wavelet Symlets	DNN	99.43	99.33	99.33	99.53	0.0057

Nota. El SA y DNN entrenados con la DB Sintética alcanzo el valor de 0 con respecto al BER, demostrando que estos modelos lograron la clasificación perfecta. Para el modelo SA entrenado con la DB Mixta *Wavelet Daubechies* se obtuvo un BER de 0.0020, el cual es un valor bajo con respecto al valor de 0.0100 que solicita el IGEPN.

Pruebas con base de datos del volcán Llaima

Estas pruebas se realizaron para saber si se pueden generalizar los modelos para otros datos de volcanes además del Cotopaxi. Se probó con la base de datos real del volcán Llaima,

además de sus respectivas bases de datos procesadas con *Daubechies* y *Symlets*. La base de datos del Llaima recibió el mismo tratamiento y procesamiento que la MicSigV1, obteniendo los siguientes resultados:

Figura 40

Representación microsismo volcánico del Llaima en el dominio temporal

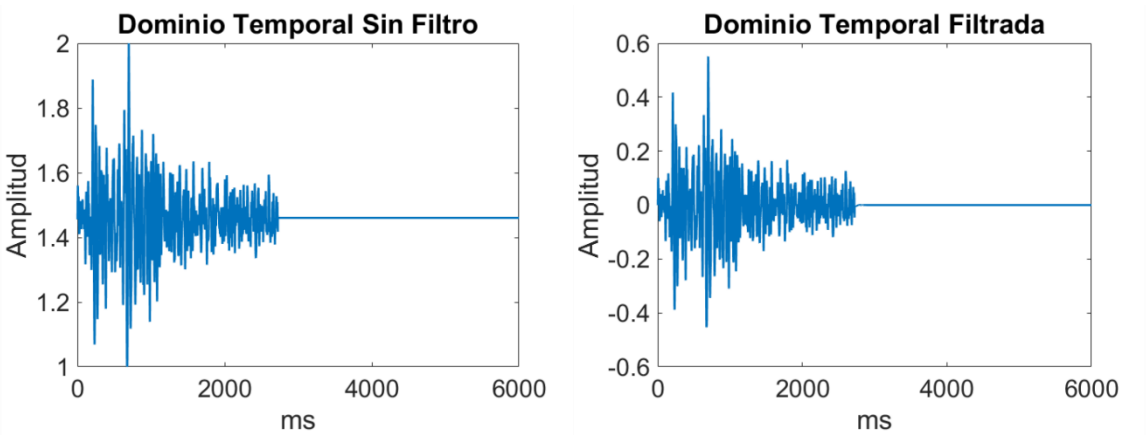


Figura 41

Representación microsismo volcánico LP del Llaima en el dominio frecuencial

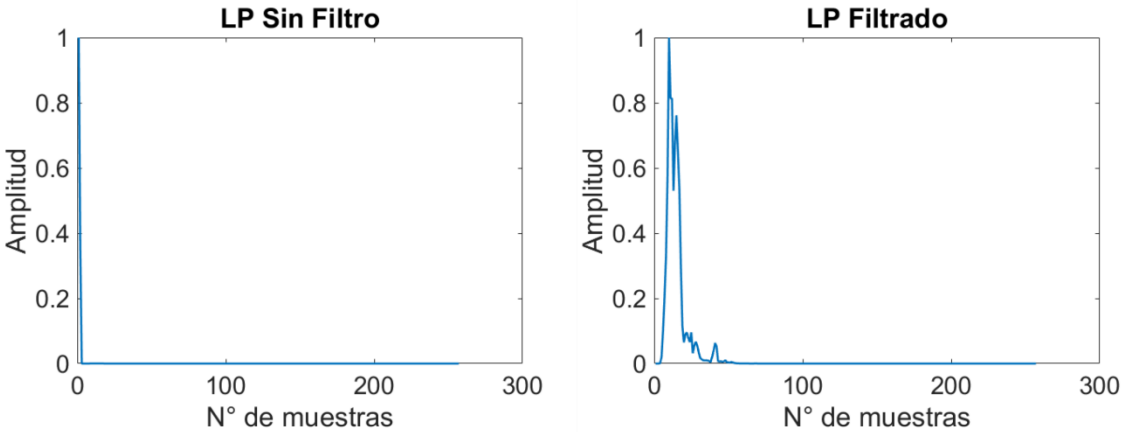


Figura 42

Representación microsismo volcánico VT del Llaima en el dominio frecuencial

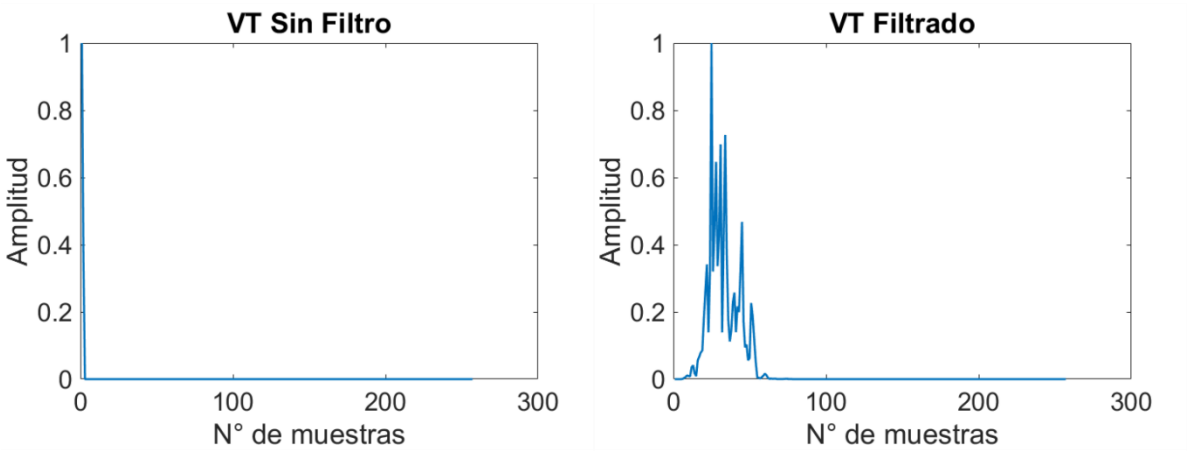


Figura 43

Representación de la transformada Wavelet Daubechies de microsismos volcánicos del Llaima en el dominio frecuencial

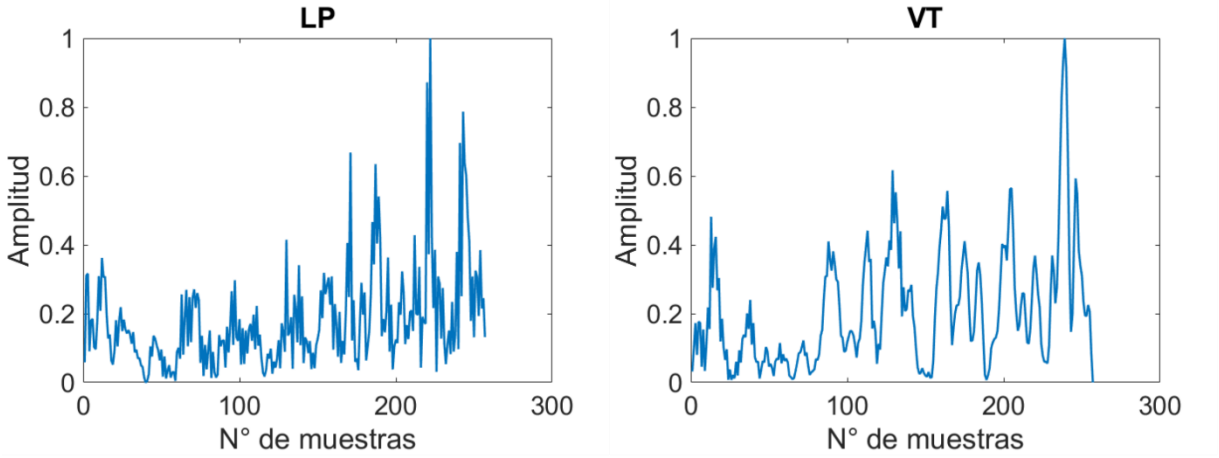


Figura 44

Representación de la transformada Wavelet Symlets de microsismos volcánicos del Llaima en el dominio frecuencial

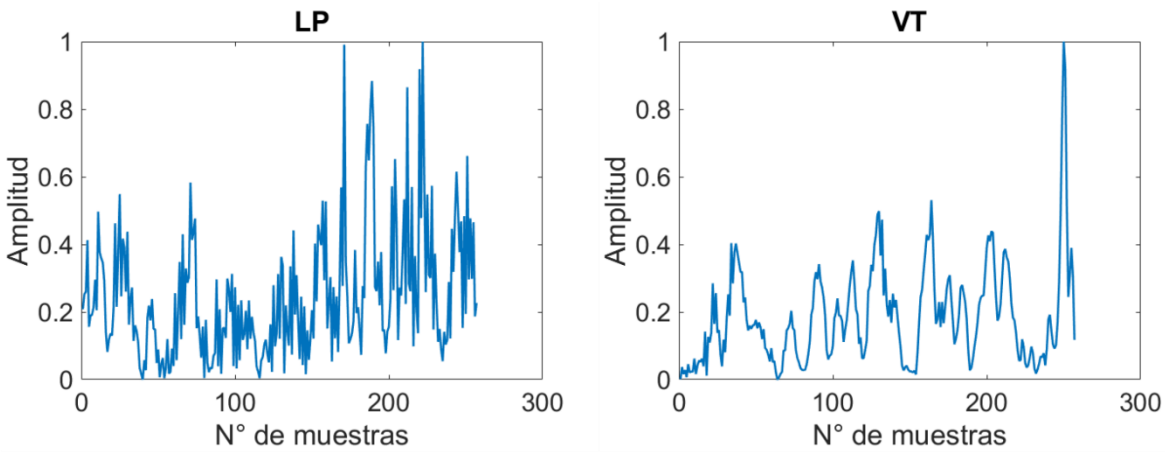


Figura 45

Representación de la FFT concatenada con la transformada Wavelet Daubechies de microsismos volcánicos del Llaima en el dominio frecuencial

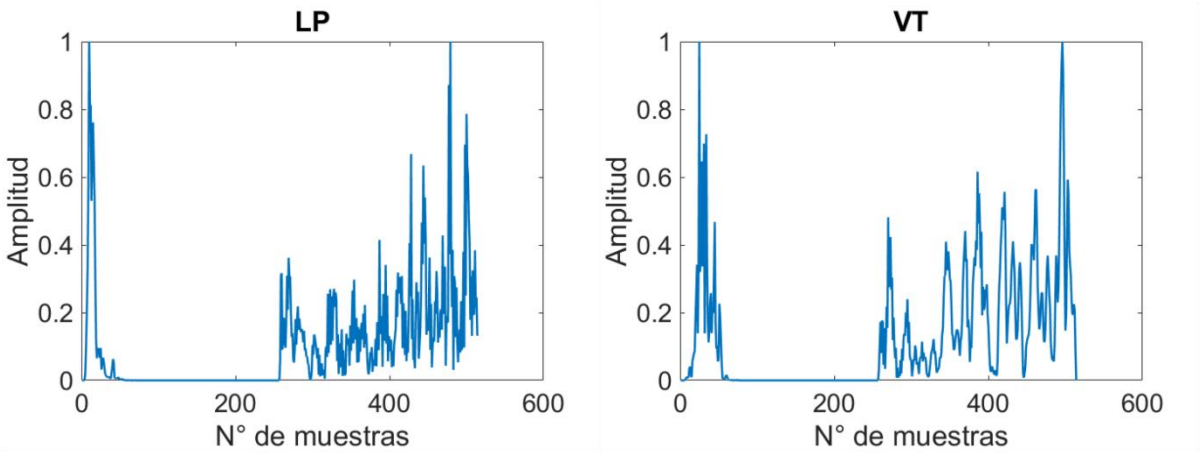
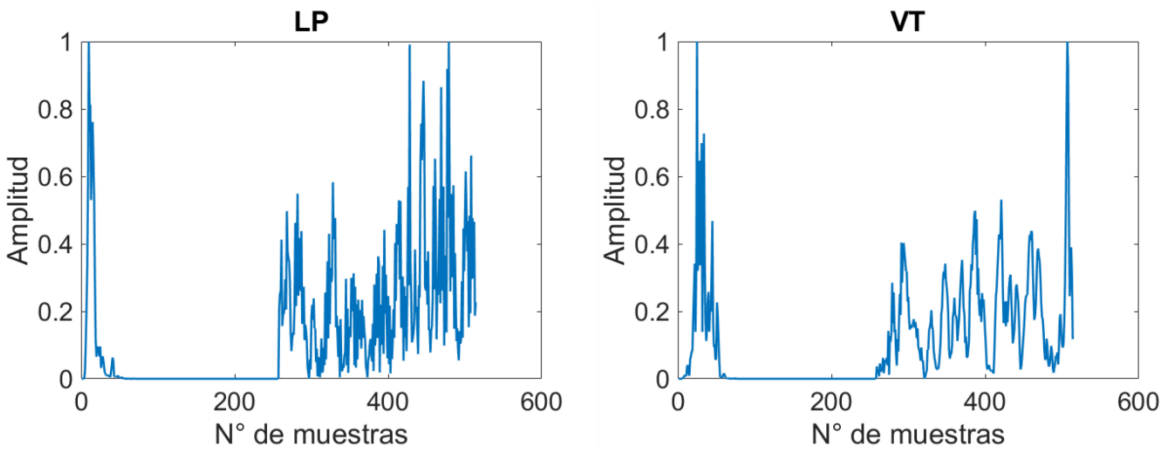


Figura 46

Representación de la FFT concatenada con la transformada Wavelet Symlets de microsismos volcánicos del Llaima en el dominio frecuencial

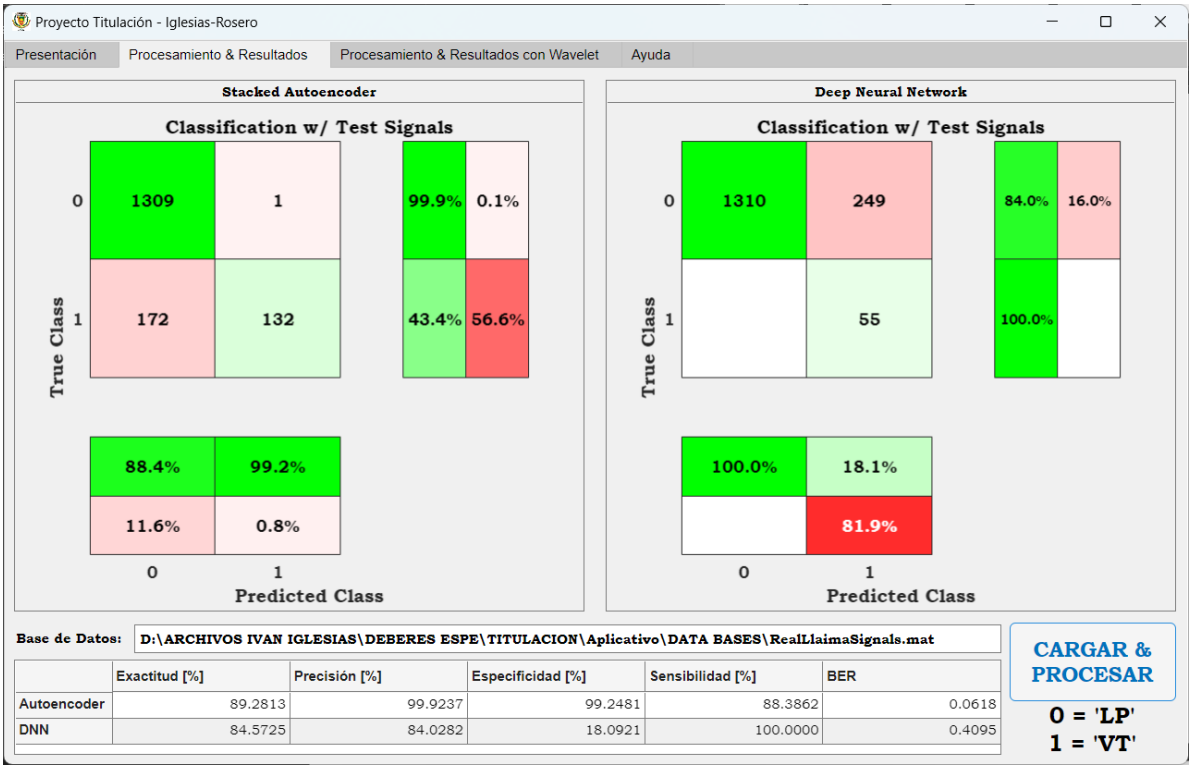


Datos de prueba DB Real

Como prueba adicional se utilizó los modelos de DL ya entrenados para probarlos con otra base de datos real del volcán Llaima, los resultados obtenidos de las métricas de rendimiento se muestran en la Figura 47, en donde el SA alcanzo las mejores métricas de rendimiento, por lo tanto, este modelo podría ser usado como base para caracterizarlo con este tipo de base datos.

Figura 47

Resultados para los datos de prueba de la base de datos Real del volcán Llaima



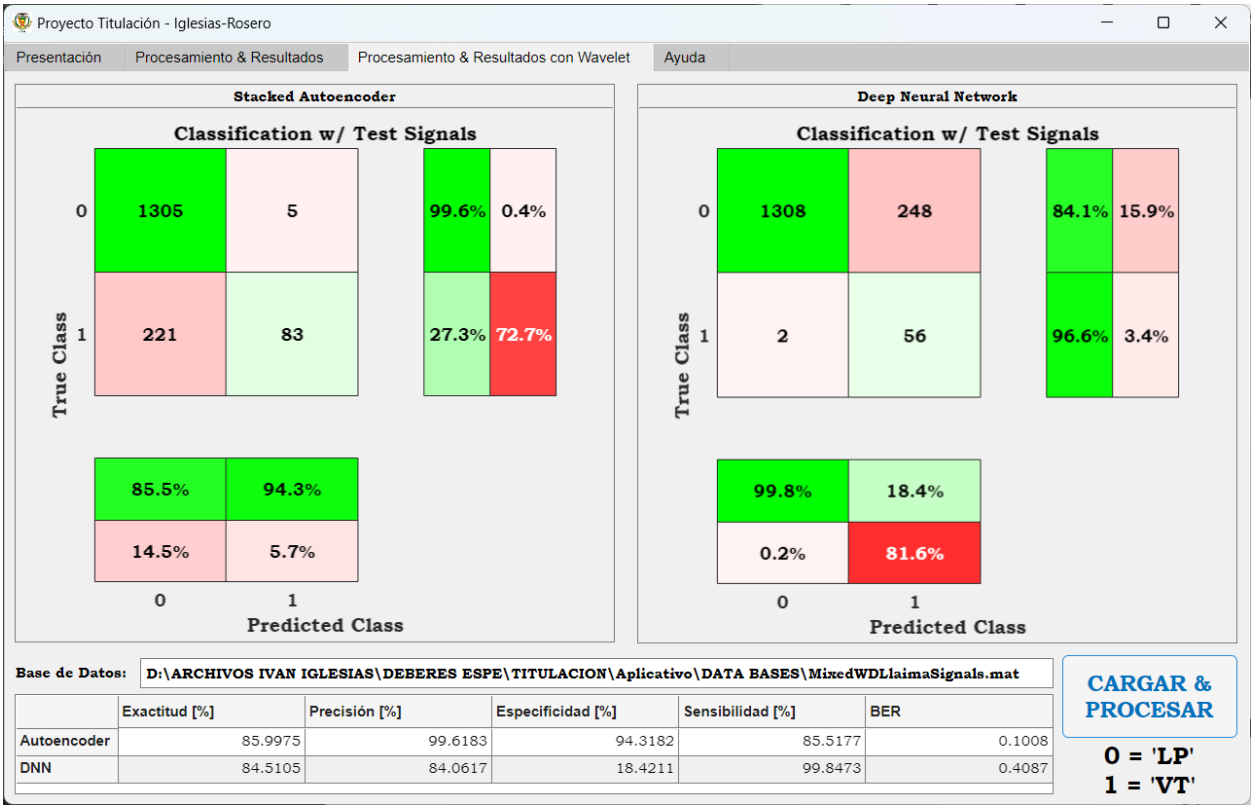
Nota. Los datos de prueba constan de 1614 microsismos volcánicos en total, 1310 del tipo LP, representados por ‘0’ y 304 microsismos volcánicos del tipo VT representados con el número ‘1’, de la base de datos real completa del volcán Llaima.

Datos de prueba DB Real con Wavelet Daubechies

En la Figura 48, se aprecia los resultados obtenidos de las métricas de rendimiento del SA y de la DNN al ser probados con la DB Real con Wavelet Daubechies, teniendo al SA como mejor modelo para clasificar los eventos LP y VT de esta base de datos.

Figura 48

Resultados para los datos de prueba de la base de datos Real con Wavelet Daubechies del volcán Llaima



	Exactitud [%]	Precisión [%]	Especificidad [%]	Sensibilidad [%]	BER
Autoencoder	85.9975	99.6183	94.3182	85.5177	0.1008
DNN	84.5105	84.0617	18.4211	99.8473	0.4087

Nota. Los datos de prueba constan de 1614 microsismos volcánicos en total, 1310 del tipo LP, representados por ‘0’ y 304 microsismos volcánicos del tipo VT representados con el número ‘1’, de la base de datos real completa del volcán Llaima, incluida *Wavelet Daubechies*.

Datos de prueba DB Real con Wavelet Symlets

Para esta prueba se utilizó una DB Real con Wavelet Symlets, donde el SA resalta con los mejores para las métricas de rendimiento, al igual que las anteriores pruebas realizadas con las bases de datos del volcán Llaima este modelo es el que mejor se acopla para la clasificación de los microsismos volcánicos.

Figura 49

Resultados para los datos de prueba de la base de datos Real con Wavelet Symlets del volcán Llaima

	Exactitud [%]	Precisión [%]	Especificidad [%]	Sensibilidad [%]	BER
Autoencoder	86.7410	99.5420	94.1176	86.2434	0.0982
DNN	85.0062	84.4961	21.0526	99.8473	0.3955

Nota. Los datos de prueba constan de 1614 microsismos volcánicos en total, 1310 del tipo LP, representados por ‘0’ y 304 microsismos volcánicos del tipo VT representados con el número ‘1’, de la base de datos real completa del volcán Llaima, incluida *Wavelet Symlets*.

Tabla 17

Mejores resultados del Volcán Llaima en el aplicativo de los modelos de DL con las distintas bases de datos.

Base de datos	Modelo DL	Exactitud [%]	Precisión [%]	Especificidad [%]	Sensibilidad [%]	BER
DB Real	SA	89.28	99.92	99.24	88.38	0.0618
DB Real	DNN	84.57	84.02	18.09	100.00	0.4095
DB Real Wavelet Daubechies	SA	85.99	99.61	94.31	85.51	0.1008
DB Real Wavelet Daubechies	DNN	84.51	84.06	18.42	99.84	0.4087
DB Real Wavelet Symlets	SA	86.74	99.54	94.11	86.24	0.0982
DB Real Wavelet Symlets	DNN	85.00	84.49	21.05	99.84	0.3955

Nota. El modelo que mejor se acopla a la DB del Volcán Llaima es el SA con el que se logró un BER de 0.0618, lo cual se puede tener como base para caracterizarlo a este tipo de base de datos.

Capítulo V

Conclusiones y Trabajos Futuros

El trabajo de investigación identificó y compiló con éxito una colección de bases de datos real, sintética y mixtas, mediante técnicas de DA. Estas bases de datos proporcionan recursos valiosos para aplicar técnicas de DL en el campo de la sismología volcánica. Al aprovechar estas bases de datos, los investigadores lograron desarrollar y mejorar modelos de DL para la clasificación de microsismos volcánicos. La disponibilidad de datos reales y aumentados permiten un entrenamiento y prueba más exhaustivos de los modelos de DL desarrollados, y facilitan una clasificación precisa y fiable de los microsismos volcánicos del Cotopaxi.

En el presente trabajo se desarrollaron con éxito algoritmos de DL, en concreto SA y DNN, para la clasificación de microsismos volcánicos de tipo LP y VT. Al aprovechar las capacidades de DL, estos algoritmos pueden clasificar y diferenciar con precisión entre eventos LP y VT, y contribuyen a mejorar la comprensión y el análisis de estos microsismos volcánicos. Los algoritmos basados en DL desarrollados proporcionan herramientas valiosas para la clasificación de microsismos volcánicos mediante el aprendizaje supervisado.

La presente investigación evaluó con éxito el rendimiento de los clasificadores mediante el análisis de las métricas de rendimiento de exactitud, precisión, especificidad, sensibilidad y BER para ambos algoritmos desarrollados de DL. El análisis de las métricas de rendimiento propuestas proporciona información valiosa sobre la eficacia y fiabilidad de los clasificadores a la hora de catalogar de manera adecuada los microsismos volcánicos de tipo LP y VT al emplear bases de datos real, sintética y mixtas para entrenamiento y prueba. La incorporación de datos tanto reales como sintéticos mejora la generalización y robustez de los clasificadores,

permiten modelos más precisos y adaptables para la clasificación de microsismos volcánicos del Cotopaxi.

Se comparó con éxito los resultados entre las técnicas de clasificación de SA y DNN, mediante la evaluación y el análisis de los resultados de ambos enfoques, se obtuvieron valiosos conocimientos sobre su rendimiento en la clasificación de microsismos volcánicos. Las mejores observaciones y pruebas del presente trabajo indican que SA demostró resultados de 99.80% de exactitud, 99.93% de precisión y especificidad, 99.66% en sensibilidad y un resultado de 0.0020 para el BER, superiores en comparación con DNN en la clasificación de microsismos volcánicos del Cotopaxi que obtuvo 99.73% de exactitud, 99.60% de precisión y especificidad, 99.86% en sensibilidad y 0.0027 como resultado del BER. El SA mostró resultados en las métricas de rendimiento superiores a los obtenidos por DNN en la correcta categorización de LP y VT, aunque ambos métodos superaron la expectativa de un BER de 0.01 solicitado por el IGEPN para la clasificación de microsismos volcánicos entre LP y VT. Además, se realizaron pruebas con bases de dato del volcán Llaima con resultados desfavorables para ambos algoritmos desarrollados para la clasificación de microsismos volcánicos LP y VT.

En conclusión, este trabajo de investigación implementó con éxito un sistema clasificador de microterremotos para el volcán Cotopaxi utilizando técnicas de DL. Los modelos DL desarrollados aprovecharon bases de datos real, sintética y mixtas para entrenar y probar los clasificadores. A través de la evaluación de métricas de rendimiento: exactitud, precisión, especificidad, sensibilidad y BER, se evaluó la eficacia y fiabilidad de los clasificadores. La comparación entre los algoritmos DL, reveló que SA demostró un rendimiento superior a la hora de clasificar con precisión los microsismos volcánicos LP y VT. El problema de clasificar estos dos tipos de eventos está solucionado, y tanto los modelos SA como DNN superaron las

expectativas establecidas por la IGEPN para una BER de 0.01. Estos resultados contribuyen al campo de la sismología volcánica, proporcionan una valiosa herramienta para mejorar la clasificación de los microsismos volcánicos del Cotopaxi.

Nuestro grupo de investigación se encuentra interesado en continuar esta línea de investigación, por lo cual se propone el diseño e implementación de un modelo multiclase de DL específicamente adaptado para la clasificación de los microsismos volcánicos de los volcanes Cotopaxi y Llaima. Este modelo DL dedicado debe considerar las características y patrones distintivos de cada volcán.

Existe gran interés también en explorar características y parámetros adicionales de las 84 características disponibles en la base de datos SeisBenchV1 a través del grupo de investigación. La ampliación del conjunto de características conducirá a modelos DL más robustos, que podrían proporcionar información valiosa para el proceso de clasificación.

La investigación futura del grupo se interesa en la creación de bases de datos sintética y mixtas específicas para el volcán Llaima, siguiendo los mismos pasos de tratamiento y preprocesamiento que los utilizados para los microsismos volcánicos del Cotopaxi. Al incorporar las transformadas Daubechies y Symlets y mantener la consistencia en el análisis, estas bases de datos podrán ser utilizadas en modelos DL, facilitan el desarrollo de modelos DL generalizados capaces de clasificar microsismos de diversos entornos volcánicos.

Estamos interesados como grupo en investigar el impacto de la fusión las bases de datos de los volcanes Cotopaxi y Llaima. Además, se debería considerar la posibilidad de trabajar con datos de PSD que contengan un mayor número de muestras tanto para los microsismos volcánicos del Cotopaxi como para los del Llaima. Aumentar el tamaño del conjunto de datos proporcionará una distribución de muestras más representativa y permitirá un entrenamiento y prueba más robustos de los algoritmos de DL. Finalmente, la incorporación de

técnicas de aprendizaje no supervisado, como algoritmos de agrupación o *clustering*, puede ayudar a descubrir patrones y estructuras ocultas en los datos para la clasificación de los microsismos volcánicos.

Referencias

Ali, T., Razwan, A. R., Baig, I., Irfan, M., & Draz, U. (2021). Efficient Shadow Encoding Scheme Towards Power Spectrum Density in Digital Network Communication. *Wireless Personal Communications*, *119*(4), 3179–3206. https://doi.org/10.1007/s11277-021-08393-4

Alloghani, M., Al-Jumeily, D., Mustafina, J., Hussain, A., & Aljaaf, A. J. (2020). *A Systematic Review on Supervised and Unsupervised Machine Learning Algorithms for Data Science* (pp. 3–21). https://doi.org/10.1007/978-3-030-22475-2_1

Carrera, E. V., Pérez, A., & Lara-Cueva, R. (2020). *Automated Systems for Detecting Volcano-Seismic Events Using Different Labeling Techniques* (pp. 133–144). https://doi.org/10.1007/978-3-030-42520-3_11

Chouet, B. A. (1996). Long-period volcano seismicity: its source and use in eruption forecasting. *Nature*, *380*(6572), 309–316.

Chouet, B. A., Page, R. A., Stephens, C. D., Lahr, J. C., & Power, J. A. (1994). Precursory swarms of long-period events at Redoubt Volcano (1989–1990), Alaska: their origin and use as a forecasting tool. *Journal of Volcanology and Geothermal Research*, *62*(1–4), 95–135.

Daubechies, I. (1992). *Ten Lectures on Wavelets* (1st edition). SIAM: Society for Industrial and Applied Mathematics.

Duque, A., González, K., Pérez, N., Benítez, D., Grijalva, F., Lara-Cueva, R., & Ruiz, M. (2020). Exploring the unsupervised classification of seismic events of Cotopaxi volcano. *Journal of Volcanology and Geothermal Research*, *403*, 107009. https://doi.org/10.1016/j.jvolgeores.2020.107009

Espitia Corredor, D. E., & Platzi. (2017). *Perceptron*. https://static.platzi.com/media/user_upload/Perceptron-5a202e5d-420a-4987-ba54-01628db3b70d.jpg

Galarza Cruz, C. R., & Vega Vergara, J. M. (2022). Generación de señales sintéticas de eventos sismo-volcánicos del volcán Cotopaxi a través de un modelo de red neuronal adversario generativo condicional. En *Tesis - Carrera de Ingeniería en Electrónica y Telecomunicaciones*. Universidad de las Fuerzas Armadas - ESPE. http://repositorio.espe.edu.ec/handle/21000/31542

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press. https://www.deeplearningbook.org

Hadi, S. J., & Tombul, M. (2018). Forecasting Daily Streamflow for Basins with Different Physical Characteristics through Data-Driven Methods. *Water Resources Management*, *32*(10), 3405–3422. https://doi.org/10.1007/s11269-018-1998-1

Ibáñez, J. M., & Carmona, E. (2000, agosto). *SISMICIDAD VOLCÁNICA*. https://www.researchgate.net/publication/242584653_SISMICIDAD_VOLCANICA.

Jaramillo, C., Leon, R., Lara-Cueva, R., Benitez, D. S., & Ruiz, M. (2014). A new structure for sequential detection and maximum entropy spectral estimator for characterization of volcanic seismic signals. *2014 IEEE Latin-America Conference on Communications (LATINCOM)*, 1–6. https://doi.org/10.1109/LATINCOM.2014.7041847

Ju, Y., Guo, J., & Liu, S. (2015). A Deep Learning Method Combined Sparse Autoencoder with SVM. *2015 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery*, 257–260. https://doi.org/10.1109/CyberC.2015.39

Khushaba, R. (2021, abril). *Wavelets-based Feature Extraction*. https://www.youtube.com/watch?v=fxfS0vSAsTA

Lara, F., Lara-Cueva, R., Larco, J. C., Carrera, E. V., & León, R. (2021). A deep learning approach for automatic recognition of seismo-volcanic events at the Cotopaxi volcano. *Journal of Volcanology and Geothermal Research*, *409*, 107142. https://doi.org/10.1016/j.jvolgeores.2020.107142

Lara, R., Rodriguez, M., & Larco, J. (2020). A real-time microearthquakes-detector based on voice activity detection and endpoint detection: An approach to Cotopaxi Volcano. *Journal of Volcanology and Geothermal Research*, *400*, 106867. https://doi.org/10.1016/j.jvolgeores.2020.106867

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. 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., Benitez, D. S., Paillacho, V., Villalva, M., & Rojo-Alvarez, J. L. (2017). On the use of multi-class support vector machines for classification of seismic signals at Cotopaxi volcano. *2017 IEEE International Autumn Meeting on Power, Electronics and Computing (ROPEC)*, 1–6. https://doi.org/10.1109/ROPEC.2017.8261613

Lara-Cueva, R., Bernal, P., Saltos, M. G., Benitez, D. S., & Rojo-Alvarez, J. L. (2015). Time and Frequency Feature Selection for Seismic Events from Cotopaxi Volcano. *2015 Asia-Pacific Conference on Computer Aided System Engineering*, 129–134. https://doi.org/10.1109/APCASE.2015.30

Lara-Cueva, R., Larco, J. C., Benítez, D. S., Pérez, N., Grijalva, F., & Ruiz, M. (2020). On finding possible frequencies for recognizing microearthquakes at Cotopaxi volcano: A machine learning based approach. *Journal of Volcanology and Geothermal Research*, *407*, 107092. https://doi.org/10.1016/j.jvolgeores.2020.107092

Lara-Cueva, R., Paillacho-Salazar, V., & Villalva-Chaluisa, M. (2017). Hacia un sistema de detección automática de señales del volcán Cotopaxi. *DYNA*, *84*(200), 176–184. https://doi.org/10.15446/dyna.v84n200.54573

Le Quy, T., Roy, A., Iosifidis, V., Zhang, W., & Ntoutsi, E. (2022). A survey on datasets for fairness-aware machine learning. *WIREs Data Mining and Knowledge Discovery*, *12*(3). https://doi.org/10.1002/widm.1452

Lidong, H., Wei, Z., Jun, W., & Zebin, S. (2015). Combination of contrast limited adaptive histogram equalisation and discrete wavelet transform for image enhancement. *IET Image Processing*, *9*(10), 908–915. https://doi.org/10.1049/iet-ipr.2015.0150

Low, C.-Y., Park, J., & Teoh, A. B.-J. (2020). Stacking-Based Deep Neural Network: Deep Analytic Network for Pattern Classification. *IEEE Transactions on Cybernetics*, *50*(12), 5021–5034. https://doi.org/10.1109/TCYB.2019.2908387

Lv, Y., Duan, Y., Kang, W., Li, Z., & Wang, F.-Y. (2014). Traffic Flow Prediction With Big Data: A Deep Learning Approach. *IEEE Transactions on Intelligent Transportation Systems*, 1–9. https://doi.org/10.1109/TITS.2014.2345663

Ma, C., Du, X., & Cao, L. (2019). Analysis of Multi-Types of Flow Features Based on Hybrid Neural Network for Improving Network Anomaly Detection. *IEEE Access*, *7*, 148363–148380. https://doi.org/10.1109/ACCESS.2019.2946708

Márquez Jairo. (2020). Artificial intelligence and Big Data as solutions to COVID-19. *Revista de Bioética y Derecho*, *50*, 315–331. http://scielo.isciii.es/scielo.php?script=sci_arttext&pid=S1886-58872020000300019&lng=es&tlng=pt

Mathworks. (2022a). *Classify data using trained deep learning neural network - MATLAB classify*. Classify data using trained deep learning neural network. https://www.mathworks.com/help/deeplearning/ref/seriesnetwork.classify.html?s_tid=doc_ta

Mathworks. (2022b). *Train Stacked Autoencoders for Image Classification - MATLAB & Simulink Example*. Train Stacked Autoencoders for Image Classification. https://www.mathworks.com/help/deeplearning/ug/train-stacked-autoencoders-for-image-classification.html?s_tid=srchtitle_stacked%20autoencoders_1

Mathworks. (2023). *MATLAB for Deep Learning*. https://www.mathworks.com/solutions/deep-learning.html

Mills, A. W., Goings, J. J., Beck, D., Yang, C., & Li, X. (2022). Exploring Potential Energy Surfaces Using Reinforcement Machine Learning. *Journal of Chemical Information and Modeling*, *62*(13), 3169–3179. https://doi.org/10.1021/acs.jcim.2c00373

Minaee, S., Kalchbrenner, N., Cambria, E., Nikzad, N., Chenaghlu, M., & Gao, J. (2022). Deep Learning--based Text Classification. *ACM Computing Surveys*, *54*(3), 1–40. https://doi.org/10.1145/3439726

Mothes, P. (Coord.). (1998). *Actividad volcánica y pueblos precolombinos en el Ecuador* (First). Ediciones Abya Yala. https://abyayala.org.ec/producto/actividad-volcanica-y-pueblos-precolombinos-en-el-ecuador/

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., Granda, F. S., Benitez, D., Grijalva, F., & Lara, R. (2022). Toward Real-Time Volcano Seismic Events' Classification: A New Approach Using Mathematical Morphology and Similarity Criteria. *IEEE Transactions on Geoscience and Remote Sensing*, *60*, 1–13. https://doi.org/10.1109/TGRS.2020.3048107

Porcelli, A. M. (2020). Inteligencia Artificial y la Robótica: sus dilemas sociales, éticos y jurídicos. *Derecho Global. Estudios sobre Derecho y Justicia*, *6*(16), 49–105. https://doi.org/10.32870/dgedj.v6i16.286

Rousselet, G. A. (2012). Does filtering preclude us from studying ERP time-courses? *Frontiers in psychology*, *3*, 131.

Saltos Torres, M. G. (2014). *Análisis de señales sísmicas del volcán Cotopaxi mediante las transformadas de WAVELET y Fourier* [Universidad de las Fuerzas Armadas - ESPE]. http://repositorio.espe.edu.ec/handle/21000/9642

Shin, D., & Yoo, H.-J. (2020). The Heterogeneous Deep Neural Network Processor With a Non-von Neumann Architecture. *Proceedings of the IEEE*, *108*(8), 1245–1260. https://doi.org/10.1109/JPROC.2019.2897076

Shorten, C., & Khoshgoftaar, T. M. (2019). A survey on Image Data Augmentation for Deep Learning. *Journal of Big Data*, *6*(1), 60. https://doi.org/10.1186/s40537-019-0197-0

Sparks, R. S. J. (1997). *Volcanic Plumes*.

Su, H., Deng, J., & Fei-Fei, L. (2012). *Crowdsourcing annotations for visual object detection*. 40–46.

Uddin, S., Khan, A., Hossain, M. E., & Moni, M. A. (2019). Comparing different supervised machine learning algorithms for disease prediction. *BMC Medical Informatics and Decision Making*, *19*(1), 281. https://doi.org/10.1186/s12911-019-1004-8

Vakili, M., Ghamsari, M., & Rezaei, M. (2020). *Performance Analysis and Comparison of Machine and Deep Learning Algorithms for IoT Data Classification*.

Venegas, P., Perez, N., Benitez, D. S., Lara-Cueva, R., & Ruiz, M. (2019). Building Machine Learning Models for Long-Period and Volcano-Tectonic Event Classification. *2019 IEEE CHILEAN Conference on Electrical, Electronics Engineering, Information and Communication Technologies (CHILECON)*, 1–6. https://doi.org/10.1109/CHILECON47746.2019.8987505

Wong, S. C., Gatt, A., Stamatescu, V., & McDonnell, M. D. (2016). Understanding Data Augmentation for Classification: When to Warp? *2016 International Conference on Digital Image Computing: Techniques and Applications (DICTA)*, 1–6. https://doi.org/10.1109/DICTA.2016.7797091

Yang, H.-F., Dillon, T. S., & Chen, Y.-P. P. (2017). Optimized Structure of the Traffic Flow Forecasting Model With a Deep Learning Approach. *IEEE Transactions on Neural Networks and Learning Systems*, *28*(10), 2371–2381. https://doi.org/10.1109/TNNLS.2016.2574840

Yu, M., Quan, T., Peng, Q., Yu, X., & Liu, L. (2022). A model-based collaborate filtering algorithm based on stacked AutoEncoder. *Neural Computing and Applications*, *34*(4), 2503–2511. https://doi.org/10.1007/s00521-021-05933-8

Zatorre, R. J., Chen, J. L., & Penhune, V. B. (2007). When the brain plays music: auditory–motor interactions in music perception and production. *Nature Reviews Neuroscience*, *8*(7), 547–558. https://doi.org/10.1038/nrn2152

Zhang, D. (2019). Wavelet Transform. En D. Zhang (Ed.), *Fundamentals of Image Data Mining: Analysis, Features, Classification and Retrieval* (pp. 35–44). Springer International Publishing. https://doi.org/10.1007/978-3-030-17989-2_3

Zhang, Z., Walter, F., McArdell, B. W., Haas, T., Wenner, M., Chmiel, M., & He, S. (2021). Analyzing Bulk Flow Characteristics of Debris Flows Using Their High Frequency Seismic Signature. *Journal of Geophysical Research: Solid Earth*, *126*(12). https://doi.org/10.1029/2021JB022755

Zhou, P., Han, J., Cheng, G., & Zhang, B. (2019). Learning Compact and Discriminative Stacked Autoencoder for Hyperspectral Image Classification. *IEEE Transactions on Geoscience and Remote Sensing*, *57*(7), 4823–4833. https://doi.org/10.1109/TGRS.2019.2893180