ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

APRENDIZAJE NO SUPERVISADO PARA EL ANÁLISIS DE SEÑALES SÍSMICAS DEL VOLCÁN COTOPAXI

AUMENTO DE DATOS PARA SEÑALES SÍSMICAS USANDO TÉCNICAS DE APRENDIZAJE NO SUPERVISADO

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

ELVIS ALEXANDER TOSCANO QUINGALUISA

elvis.toscano@epn.edu.ec

DIRECTOR: VIVIANA CRISTINA PÁRRAGA VILLAMAR

viviana.parraga@epn.edu.ec

DMQ, febrero 2024

CERTIFICACIONES

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

ELVIS ALEXANDER TOSCANO QUINGALUISA

Certifico que el presente trabajo de integración curricular fue desarrollado por ELVIS ALEXANDER TOSCANO QUINGALUISA, bajo mi supervisión.

VIVIANA CRISTINA PÁRRAGA VILLAMAR
DIRECTOR

DECLARACIÓN DE AUTORÍA

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

ELVIS ALEXANDER TOSCANO QUINGALUISA

VIVIANA CRISTINA PÁRRAGA VILLAMAR

DEDICATORIA

Ha sido un proceso lleno de emociones, con caídas, errores, aciertos, experiencias buenas y malas. Ha sido un camino lleno de piedras, carente de la certeza de saber a dónde me llevará el siguiente paso. Ha sido un viaje del cual no me he bajado gracias a personas que me han acompañado, desde el inicio, a mitad del camino y en la recta final. Personas que muchas veces sin saberlo han hecho que mi proceso haya sido más llevadero.

A mi familia, en especial a mi mamá Flor Quingaluisa, a mi hermana Luz Toscano y a mi hermano Luis Toscano, por ser mi mayor fuente de inspiración, por su amor incondicional y por su constante apoyo en este camino hacia la realización de mis sueños.

A mis estimados maestros, Ana Zambrano, Ana Rodríguez y Pablo Hidalgo, por su dedicación y enseñanzas que han dejado una huella imborrable en mi formación académica y personal.

A mi tutora de tesis, Viviana Párraga, por su orientación, paciencia y guía durante todo este proceso, ayudándome a alcanzar mis metas.

A mis amigos, Cristian Lema, Michelle Chiluisa, Bryan Calvopiña y Alex Ramos, quienes han sido mi sostén emocional, mi fuente de alegría y motivación. Y en especial a Alex Ramos, por su apoyo incondicional en los momentos más difíciles, demostrándome que la verdadera amistad trasciende cualquier obstáculo.

A mi amiga Gysel, gracias por estar presente en los momentos clave. Tu apoyo constante ha sido esencial para mí en esta etapa final. Tu presencia ha sido un gran respaldo que me ha ayudado a mantenerme enfocado y motivado.

Finalmente, a todas aquellas personas que, aunque no estén mencionadas, han dejado su huella en mi camino, les agradezco de corazón. Cada gesto, cada palabra de aliento y cada momento compartido ha contribuido de alguna manera a mi crecimiento y desarrollo. Su influencia no ha pasado desapercibida y siempre llevaré conmigo el agradecimiento por su contribución a mi camino.

A todos ustedes, mi eterna gratitud.

AGRADECIMIENTO

Ha llegado el momento de expresar mi más sincero agradecimiento a todas aquellas personas que contribuyeron de manera significativa en la realización de este trabajo de tesis. Sus aportes, apoyo y aliento fueron fundamentales para culminar este importante proyecto académico.

En primer lugar, quiero agradecer a mi familia, en especial a mi mamá Flor Quingaluisa, a mi hermana Luz Toscano y a mi hermano Luis Toscano. Su amor incondicional, comprensión y constante respaldo fueron el motor que me impulsó a seguir adelante en cada paso de este camino académico.

Agradezco también a mis estimados maestros, Ana Zambrano, Ana Rodríguez y Pablo Hidalgo, cuya dedicación y enseñanzas dejaron una huella imborrable en mi formación. Su orientación y conocimientos fueron fundamentales para el desarrollo de este trabajo de investigación.

Agradezco de manera especial a mi tutora de tesis, Viviana Párraga, por su valiosa orientación, paciencia y guía a lo largo de todo el proceso. Sus sugerencias y retroalimentación contribuyeron de manera significativa a la calidad y rigurosidad de este trabajo.

Asimismo, quiero expresar mi profundo agradecimiento a mis amigos, Cristian Lema, Michelle Chiluisa, Bryan Calvopiña y Alex Ramos. Su compañía, palabras de aliento y apoyo incondicional fueron un pilar fundamental durante los momentos más desafiantes de este proceso. En especial, agradezco a Alex Ramos por estar siempre presente, brindándome su apoyo incondicional y demostrándome que la verdadera amistad trasciende cualquier obstáculo.

A todas estas personas, mi más sincero agradecimiento por formar parte de este viaje académico y por hacer que este proceso haya sido más llevadero y enriquecedor. Su contribución y apoyo han sido invaluable y quedarán grabados en mi memoria para siempre.

ÍNDICE DE CONTENIDO

RESUMEN

El presente proyecto se ha enfocado en el procesamiento y aumento de datos de señales sísmicas del volcán Cotopaxi. Se ha descrito de manera detallada el proceso empleado, el cual ha empezado desde la obtención de la base de datos original hasta la generación de datos sintéticos y su correspondiente análisis. Se ha empleado una metodología que abarca el cambio de formato de datos de .mat a .json, la estructuración y ordenamiento de la información, el tratamiento de datos y el entrenamiento mediante el modelo NICE para el aumento de datos.

Los resultados han sido favorables en la modificación y reestructuración de la base de datos, así como la implementación exitosa del modelo NICE para el aumento de datos en el conjunto de datos VC1. Sin embargo, se ha observado que el desempeño no ha sido óptimo en el conjunto de datos BREF, lo que ha dado cabida a sugerencias para futuras mejoras.

Finalmente, se ha obtenido un resultado deseable del modelo NICE en el aumento de datos para el conjunto de datos VC1. A pesar de que el desempeño del modelo no ha sido óptimo para el conjunto de datos BREF, se ha hecho énfasis en la importancia de explorar técnicas adicionales para su mejora. En general, el estudio ha proporcionado una metodología para el procesamiento y aumento de datos de señales sísmicas.

PALABRAS CLAVE: Señales sísmicas, Aumento de datos, Modelo NICE.

ABSTRACT

The present project has focused on the processing and augmentation of seismic signal data from the Cotopaxi volcano. It has been described in detail the process used, which has started from obtaining the original database to the generation of synthetic data and its corresponding analysis. A methodology has been used that includes the change of data format from .mat to .json, the structuring and ordering of the information, data processing and training using the NICE model for data augmentation.

The results have been favorable in the modification and restructuring of the database, as well as the successful implementation of the NICE model for data augmentation on the VC1 dataset. However, it has been observed that the performance has not been optimal in the BREF dataset, which has given room for suggestions for future improvements.

Finally, a desirable result of the NICE model in data augmentation has been obtained for the VC1 dataset. Although the performance of the model has not been optimal for the BREF dataset, the importance of exploring additional techniques for improvement has been emphasized. Overall, the study has provided a methodology for processing and augmentation of seismic signal data.

KEYWORDS: Seismic signals, Data augmentation, NICE model.

1 INTRODUCCIÓN

Una erupción volcánica desencadena efectos adversos para las personas que viven en sus cercanías como caída de ceniza, lahares, entre otros. Así, las consecuencias negativas de una erupción hacen imprescindible el monitoreo de volcanes [1]. En el Ecuador, es el Instituto Geofísico de la Escuela Politécnica Nacional (IG-EPN) el responsable de este monitoreo mediante sismómetros que capturan señales sísmicas provenientes de los volcanes. Estas señales sísmicas son procesadas para entender el estado de un volcán activo. Particularmente, categorizar los tipos de eventos sísmicos contenidos en dichas señales es crucial para encontrar posibles precursores de una erupción. Entre estos eventos se tiene, a los eventos de período largo (Long Period, LP) y a los volcano tectónicos (Volcano Tectonic, VT) [1].

Sin embargo, el "etiquetado" (i.e., categorización) manual de eventos sísmicos es una tarea que consume un tiempo considerable de recursos humanos altamente calificados (i.e., vulcanólogos). Además, la enorme cantidad de datos capturada diariamente por los sistemas de monitoreo hace inviable su procesamiento manual [2]. Frente a esto, en la literatura se ha propuesto el uso de aprendizaje automático (machine learning) para entrenar modelos que clasifiquen automáticamente los eventos sísmicos [3]. La mayor parte de trabajos se han enfocado en el aprendizaje supervisado donde el modelo es entrenado a partir de bases de datos etiquetadas. Una base de datos etiquetada hace referencia a datos que contienen tanto un espacio de características del evento sísmico (es decir, características relevantes del evento obtenidas, por ejemplo, de sus estadísticas) como la categoría o "etiqueta" (por ejemplo, evento tipo VT, LP, etc.) a la que pertenece el evento [3]. La principal desventaja de estos enfoques es la necesidad de una base de datos etiquetada manualmente por expertos.

Por otro lado, una base de datos no etiquetada contiene los datos de los eventos sin sus etiquetas o categorías. Los modelos de aprendizaje no supervisado usan bases de datos no etiquetadas para descubrir estructuras ocultas a partir de los eventos sísmicos que podrían ayudar a caracterizar mejor la dinámica del volcán [4]. Independientemente del tipo de aprendizaje (supervisado o no supervisado), el espacio de características de los eventos sísmicos es crucial para su clasificación y/o caracterización. Por ejemplo, muchos expertos han diseñado cuidadosamente varios espacios de características basados en las estadísticas del evento en tiempo, frecuencia u otro dominio [1]. Sin embargo, estos espacios suelen estar sesgados al conocimiento en mayor o menor grado de cierto dominio por parte del experto. Por este motivo, aprender de manera automática espacios de características aliviaría este sesgo, lo que se traduciría en modelos de clasificación de eventos sísmicos más robustos. De hecho, en otras áreas como la visión computacional ya se ha evidenciado que espacios de características aprendidos automáticamente desembocan en mejores modelos de clasificación. Considerando estos antecedentes y dado que las bases de datos no etiquetadas suelen ser

más abundantes porque no requieren etiquetado manual de expertos. Este proyecto tiene como objetivo utilizar técnicas de aprendizaje no supervisado para el aumento de datos.

1.1 OBJETIVO GENERAL

Aumentar datos de señales sísmicas del volcán Cotopaxi utilizando técnicas de aprendizaje no supervisado para introducir datos no observados.

1.2 OBJETIVOS ESPECÍFICOS

1. Adecuar la base de datos de eventos sísmicos del Volcán Cotopaxi.
2. Describir técnicas de aumento de datos basadas en aprendizaje no supervisado.
3. Implementar una técnica de aumento de datos basada en aprendizaje no supervisado.
4. Evaluar la técnica de aumento de datos implementada.

1.3 ALCANCE

El presente proyecto abarca el aumento de datos para señales sísmicas del volcán Cotopaxi utilizando aprendizaje no supervisado. Para llevar a cabo esta tarea, se han utilizado las bases de datos de eventos sísmicos del volcán Cotopaxi proporcionadas por el IG-EPN que están disponibles en el repositorio público ESeismic. Este repositorio contiene bases de datos de eventos sísmicos etiquetados por expertos, siendo estas últimas las que se han utilizado para el aumento de datos debido a que no incluyen la categorización de los eventos.

Para el desarrollo del proyecto se incluye una fase de diseño, implementación, pruebas y análisis de resultados. En la fase de diseño se realiza la adecuación de la base de datos para que pueda ser utilizada el posterior análisis. Además, se ha realizado una investigación acerca de la técnica o modelo que se ha empleado en el aumento de datos. En la fase de implementación se ha implementado la técnica o modelo de aumento de datos elegida en la fase de diseño. Finalmente, en la fase de pruebas y análisis de resultados se ha realizado una evaluación de la técnica o modelo de aumento de datos implementada en la fase anterior, para verificar su eficacia y precisión. Se ha llevado a cabo un análisis de los resultados obtenidos y se ha redactado un documento escrito que incluya los detalles del proceso seguido, los resultados obtenidos y las conclusiones del proyecto.

La fase de adecuación de datos se la ha realizado con la ayuda de un lenguaje de programación. Dependiendo del lenguaje de programación empleado, se ha procedido a utilizar el formato de datos que más facilidades tenga. Debido a que inicialmente se tiene la extensión del dataset en .mat, se ha cambiado a un formato conveniente.

Antes de ingresar a la fase de implementación, se ha elegido una técnica o modelo de aprendizaje no supervisado para el aumento de datos. Para la elección de la técnica a utilizarse, se ha tomado como criterio la lectura de artículos relacionados al presente tema. Una vez seleccionada la técnica se ha puesto en marcha su implementación mediante el lenguaje de programación elegido.

1.4 MARCO TEÓRICO

1.4.1 TIPOS DE EVENTOS SÍSMICOS VOLCÁNICOS

El mundo ha sufrido un impacto considerable debido a las erupciones volcánicas. A escala mundial se tiene registros de 7670 fallecimientos por causa directa o indirecta de erupciones volcánicas en el intervalo de 1986 a 2019 [1]. Es importante resaltar que las ciudades próximas a volcanes activos presentan un riesgo mayor. Debido a esta problemática, el monitoreo constante de volcanes es crucial para reducir el impacto de esta amenaza natural [1].

Existen varios tipos de eventos sísmicos que se pueden monitorear en un volcán. Cada uno de los eventos sísmicos son categorizados dependiendo de características específicas. Los eventos sísmicos se han clasificado en: volcano tectónicos, de largo periodo, híbridos, tremor volcánico y explosiones. Los registros de eventos sísmicos ayudan a obtener un mejor entendimiento de la actividad volcánica [5].

En primer lugar, los eventos volcano tectónicos (VT) son de alta frecuencia y similares a los eventos tectónicos. En general, estos sismos indican aperturas de grietas o ruptura de rocas. Este tipo de eventos suele preceder a la actividad eruptiva [5].

En segundo lugar, los eventos de largo periodo (LP) son de baja frecuencia y se relacionan al movimiento de fluidos como gases o magma. La duración de este tipo de eventos se encuentra en el intervalo de los segundos hasta más de un minuto. El análisis de estos eventos son complejos de analizar debido a la dificultad de determinar su inicio [5].

En tercer lugar, los eventos híbridos (HB) comparten características similares entre VT y LP. Este tipo de eventos indican movimientos de fluidos (Eventos híbridos) y fracturas (Eventos de largo periodo) [5].

En cuarto lugar, el evento tremor volcánico es una señal sísmica que persiste en intervalos de tiempo que varían desde minutos hasta horas. Esta señal sísmica está asociada a la salida de vapor, ceniza y gases. El tremor volcánico se puede clasificar en: tremor de alta frecuencia, tremor armónico y tremor de baja frecuencia. Cada tipo de tremor volcánico está asociado a la frecuencia de la señal sísmica. Dependiendo de la frecuencia del tremor volcánico se puede reflejar diferentes procesos y condiciones dentro del volcán [5].

Finalmente, las explosiones son señales sísmicas típicas en el proceso de una erupción. Las explosiones y los tremores son los dos tipos de señales sísmicas que se identifican cuando existe un proceso eruptivo [5].

1.4.2 REPOSITORIO DE SEÑALES SÍSMICAS DEL VOLCÁN COTOPAXI

MicSigV1 junto a SeisBenchV1 son conjuntos de datos de señales sísmicas en Ecuador. SeisBenchV1 es un conjunto de datos derivado de MicSigV1. Ambos, MicSigV1 y SeisBenchV1 conforman el primer repositorio de señales sísmicas volcánicas del Ecuador abierto al público. Este repositorio tiene como meta llegar a la comunidad científica [1].

El repositorio de señales sísmicas del volcán Cotopaxi ha sido alimentado mediante la recolección de datos de las estaciones VC1 y BREF. Los datos se recogieron en distintos periodos que comprenden desde el año 2012 al 2019. Durante el periodo de recolección de datos, se han recolectado 1187 eventos sísmicos de las estaciones VC1 y BREF. Los eventos sísmicos que se han observado son: VT (volcano tectónicos), LP (largo periodo), ICE (terremoto de hielo), HB (híbrido) y REG (regional). Cada uno de los eventos sísmicos han sido etiquetados por expertos en el área. A continuación, se puede visualizar la descripción de las variables del conjunto de datos del volcán Cotopaxi [1].

Variable	Descripción
Network	Código de red ecuatoriano
	Valor: EC
Station	Código de la estación
	Valores: VC1, BREF
SampleRate	Frecuencia de muestreo en Hz
	Valores: 100 (VC1) y 50 (BREF)
Component	Hacha de sismómetro
	Valor: SHZ (eje vertical)
Year	Año
Month	Mes
Type	Etiqueta del evento
	Valores: VT, LP, ICE, HB, REG
Duration	Duración en segundos
StartPoint	Inicio del evento en muestras con respecto a la variable Data
EndPoint	Fin del evento en muestras con respecto a la variable Data
Data	Señal extendida 10 segundos antes y después de detectarse el evento

Figura 1 Descripción de variables del conjunto de datos sísmicos del volcán Cotopaxi [1].

1.4.3 PREPROCESAMIENTO DE DATOS

El tratamiento y preprocesamiento de las señales sísmicas volcánicas en el presente Trabajo de Integración Curricular, implica el manejo de algunos formatos de archivos, así como de lenguajes de programación y varias librerías. A continuación, se explican conceptos asociados al manejo de la información obtenida en el repositorio.

1.4.3.1. Formatos (.MAT y .JSON)

Un archivo en cuya extensión es .MAT es un archivo contenedor de datos, utilizado por el programa MATLAB. Este tipo de archivo puede incluir datos tales como funciones, variables, matrices [6].

Un archivo JSON es un archivo de texto simple, que contiene datos estructurados para guardar e intercambiar información, cuya lectura es sencilla tanto para sistemas informáticos como para personas. Originalmente proviene de JavaScript, sin embargo, dado su versatilidad y simplicidad, este formato se ha ido popularizando, por lo que es compatible con la mayoría de lenguajes de programación y se ha vuelto un formato predominante para el intercambio de datos [7].

1.4.3.2. Lenguajes de programación (Matlab y Python)

MATLAB es un lenguaje de programación multiparadigma, que posee su propio IDE (*Integrated Development Environment*, Entorno de Desarrollo Integrado) y una amplia colección de librerías. Inicialmente se dio a conocer como un lenguaje de programación basado en matrices, por lo que de allí viene su nombre, mediante las siglas “Matrix Laboratory”. MATLAB se caracteriza por su facilidad de uso y por disponer de una gran cantidad de funciones propias que se utilizan ampliamente en el área de las ciencias y la ingeniería [8].

Python es un lenguaje de programación interpretado de alto nivel multiparadigma, dinámico y multiplataforma [9]. Fue creado por Guido van Rossum y lanzado en 1991. En la actualidad, Python se emplea extensamente en aplicaciones web, ciencia de datos, desarrollo de software y Machine Learning (Aprendizaje Automático) [10], [11].

Por otro lado, es importante conceptualizar una herramienta muy útil para el desarrollo del presente Trabajo de Integración Curricular, la cual se denomina Google Colab.

Este producto de Google Research, consiste en un servicio alojado de recursos computacionales que permite a los usuarios escribir y ejecutar código de Python directamente desde el navegador web, sin requerir de ningún tipo de instalación, además, su uso es gratuito [12]. Esta herramienta es ideal para aplicaciones de análisis de datos, educación y aprendizaje

automático, razón por la cual, representa una herramienta fundamental para el desarrollo del presente proyecto.

1.4.3.3. Librerías (Python)

A continuación, se describen las librerías de Python más importantes, las cuales serán utilizadas para el desarrollo del presente trabajo.

- **Numpy:** es una librería de Python que se especializa en análisis de datos y cálculos numéricos. Esta librería es capaz de manejar un gran volumen de datos de manera eficiente, ya que los procesa rápidamente [13].
- **Pandas:** es una librería de Python especializada en el análisis y manipulación de datos estructurados como tablas, matrices o series temporales. Esta librería tiene un amplio uso en el campo del Aprendizaje Automático [14].
- **Json:** La librería json de Python, es un paquete que permite trabajar con datos en formato JSON [15].

1.4.4 APRENDIZAJE AUTOMÁTICO

El Aprendizaje Automático (Machine Learning) se describe como una rama de la inteligencia artificial (IA) que habilita a las máquinas para aprender de manera automática a partir de datos y experiencias previas, permitiéndoles reconocer patrones y realizar predicciones con mínima intervención humana [16].

Los métodos de aprendizaje automático posibilitan que las computadoras funcionen de manera autónoma sin necesidad explícita de programación. Los algoritmos de aprendizaje automático son moldeados utilizando un conjunto de datos de entrenamiento para construir un modelo. Cuando se presenta nuevo conjunto de datos al algoritmo de aprendizaje automático previamente entrenado, utiliza el modelo creado para hacer una predicción [16].

Debido a que los algoritmos de aprendizaje automático pueden ser entrenados de diferentes maneras, el aprendizaje automático se clasifica principalmente en las siguientes categorías [16]:

- Aprendizaje supervisado
- Aprendizaje no supervisado
- Aprendizaje semisupervisado
- Aprendizaje de refuerzo

Con el aprendizaje supervisado, la computadora utiliza un conjunto de datos etiquetados para aprender a realizar una tarea humana, siendo este el modelo menos complejo al intentar imitar el proceso de aprendizaje humano [17].

Por otro lado, con el aprendizaje no supervisado, la computadora trabaja con datos no etiquetados y extrae información o patrones previamente desconocidos. Existen diversas formas en que los algoritmos de aprendizaje automático llevan a cabo esta tarea, entre las cuales se incluyen [17]:

1. **Agrupación en clúster**, donde la computadora identifica puntos de datos similares dentro de un conjunto de datos y los agrupa en "clústeres".
2. **Estimación de densidad**, en la cual la computadora descubre patrones al observar la distribución de los datos.
3. **Detección de anomalías**, donde la computadora identifica puntos de datos significativamente diferentes del resto en un conjunto de datos.
4. **Análisis de componente principal (PCA)**, mediante el cual, la computadora analiza y resume un conjunto de datos para utilizarlo en la realización de predicciones precisas.

1.4.5 RED NEURONAL

A una red neuronal (RN[1]) también se le conoce con el nombre de red neuronal artificial (RNA[2]). Una red neuronal se ha constituido mediante capas que están interconectadas mediante nodos. Los nodos son pequeñas unidades cuya función ha sido realizar operaciones matemáticas con la finalidad de descubrir patrones en un conjunto de datos. Las RN han sido diseñadas inspirándose en el funcionamiento de las neuronas [18].

Dentro de la teoría de las RN se ha presentado los siguientes términos: neurona, entrada, Red Neuronal Profunda (RNP[3]), pesos, bias y función de activación. La neurona se ha encargado de recibir valores ponderados, realizar cálculos matemáticos y generar una salida, debido a esto, es considerada el bloque fundamental de una RN. La entrada hace referencia a los valores que se han transmitido a las neuronas para su procesamiento. A la Red Neuronal Profunda se la ha considerado como una RNA, con la diferencia que ha presentado diversas capas ocultas entre la entrada y salida. Una Red Neuronal Profunda es usada para facilitar que patrones complejos sean reconocidos. Los pesos son de suma importancia para el aprendizaje y detección de patrones debido a que representan el grado de importancia de las conexiones existentes entre neuronas. Otro elemento importante es el Bias el cual ha sido de gran ayuda

[1] **RN:** Red Neuronal
[2] **RNA:** Red Neuronal Artificial
[3] **RNP:** Red Neuronal Profunda

para modular la activación de un nodo, además de facilitar respuestas retardadas o acelerads. Finalmente se tiene a la función de activación, cuyo propósito ha sido permitir que la RN aprenda patrones intrincados. La función de activación mejora la capacidad de la red para procesar datos complejos gracias a su propiedad de no linealidad [18].

Es necesario saber que dentro las RN, los pesos y biases se han utilizado como parámetros que se pueden entrenar. Esto ha permitido que la red pueda aprender patrones mediante el ajuste de los pesos y biases con la finalidad de la mejora del rendimiento y la optimización de predicciones [18].

1.4.6 FLUJO DE NORMALIZACIÓN, GAN Y VAE

El flujo de normalización es una técnica utilizada para conocer la distribución de probabilidad de conjuntos de datos extensos. Las redes generativas antagónicas (GAN[4]) y los autoencoders variacionales (VAE[5]) son otras técnicas utilizadas. Estas técnicas se emplean en el aprendizaje de distribuciones de datos complejos. Sin embargo, los flujos de normalización abordan limitaciones que presentan las GAN y VAE. Algunas limitaciones son: resultados borrosos y de baja calidad al emplear VAE y el colapso de modo al utilizar GAN. Debido a que los flujos de normalización utilizan funciones reversibles, superan las limitaciones que tienen los VAE y GAN [19].

Los flujos de normalización hacen uso de funciones biyectivas, es decir, funciones cuyas inversas pueden ser obtenidas de manera analítica. Un ejemplo simple de función biyectiva es la función identidad $f(x) = x$. Las funciones biyectivas se caracterizan por ser reversibles. Una función es reversible si existe una única salida para cada entrada de una función. Según este criterio, las funciones cuadráticas como por ejemplo $f(x) = x^2$ no son candidatas para realizar un flujo de normalización [19].

Debido al uso de funciones reversibles, el aumento de datos es una tarea que los flujos de normalización cumplen con precisión y confiabilidad. Sin embargo, debido a que se debe mantener la biyectividad[6] de los modelos de flujo, es posible que se llegue a espacios latentes de altas dimensiones, lo que puede aumentar la dificultad de interpretación. También se debe considerar que las muestras generadas por modelos basados en flujo, en comparación con

[4] **GAN:** Redes Generativas Antagónicas.
[5] **VAE:** Autoencoder Variacional.
[6] **Biyectividad:** En matemáticas, la biyectividad se refiere a una conexión entre conjuntos donde cada elemento del primer conjunto está vinculado de manera precisa con uno y solo uno del segundo conjunto, y viceversa. En pocas palabras, esto implica que no hay elementos repetidos ni excluidos en la relación entre ambos conjuntos [20].

GAN y VAE, no son muy buenas. Aunque los modelos basados en flujo sean una buena opción para el aumento de datos, se debe tener en cuenta sus limitaciones en la calidad de las muestras [19].

Aspecto	GAN	VAE	Flujo de normalización
Concepto	Genera muestras de datos desde ruido.	Aprende representaciones latentes y genera datos.	Transforma datos entre distribuciones complejas y simples.
Estabilidad en el entrenamiento	Inestable, propenso a colapsos de modo y gradientes que desaparecen.	Más estable, pero puede producir resultados borrosos.	Muy estable, evita problemas comunes de entrenamiento.
Claridad en la Inferencia	No tiene una evaluación exacta de la distribución de probabilidad.	Inferencia más simple pero puede resultar borrosa.	Evaluación exacta e inferencia de la distribución de probabilidad.
Expresividad	Puede producir muestras de alta calidad pero puede sufrir de colapso de modos.	Capaz de generar muestras diversas pero puede tener limitaciones.	Ofrece modelos de variación local poderosos sin ruido.
Convergencia	Requiere ajuste cuidadoso de hiperparámetros y puede ser desafiante converger.	Relativamente más fácil de converger pero aún requiere optimización.	Mucho más fácil de converger, evitando desafíos comunes.
Calidad de Muestras	Conocido por producir muestras de alta calidad.	Puede producir muestras decentes pero pueden carecer de nitidez.	Las muestras pueden no ser tan buenas como otros métodos.
Interpretabilidad	Interpretación desafiante debido a falta de preservación de volumen.	Interpretación posible pero puede no preservar bien el volumen.	Interpretación desafiante debido a alta dimensionalidad.
Aplicaciones	Ampliamente utilizado en generación de imágenes, transferencia de estilo y aumento de datos.	Aplicado en varios dominios incluyendo generación de imágenes y aprendizaje de representaciones.	Adecuado para estimación de densidad, detección de valores atípicos y bioinformática.

Figura 2 Tabla comparativa entre las técnicas de flujo de normalización, GAN y VAE **[19]**

En conclusión, se ha observado que de acuerdo a la Figura 2, los flujos de normalización han ofrecido una alternativa estable y precisa para la modelización de datos complejos. Se ha destacado por la estabilidad en el proceso de entrenamiento y la capacidad de proporcionar una inferencia exacta. Se ha sugerido la utilización de flujos de normalización cuando se requiere una modelización estable, precisa de distribuciones complejas [19].

1.4.7 FUNDAMENTOS DEL MODELO NICE

NICE (Estimación de Componentes Independientes No lineales) es un modelo que mediante el empleo de flujos de normalización permite resolver tareas como: estimación de densidad y generación de datos. Debido a que NICE emplea capas de acoplamiento aditivo, se le denomina un modelo generativo basado en flujos. Las capas de acoplamiento aditivo que utiliza

el modelo NICE son funciones reversibles cuya finalidad es mapear la distribución de probabilidad de muestras reales a una distribución previa [20].

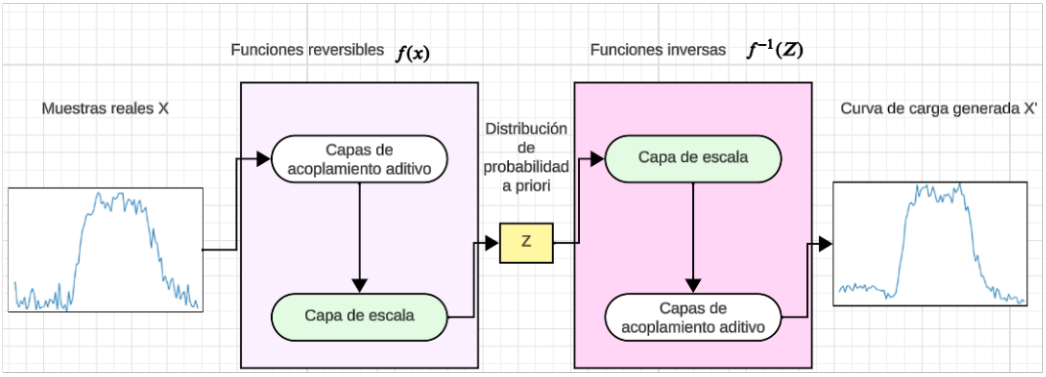


Figura 3 Ejemplo de arquitectura del modelo NICE [21].

1.4.8 PÉRDIDAS DE ENTRENAMIENTO Y VALIDACIÓN

En este apartado se ha dado a conocer dos tipos de métricas que son importantes al momento de evaluar un modelo de aprendizaje automático los cuales son: pérdidas de entrenamiento y validación. Para entender sobre estas métricas, se ha empezado definiendo los conceptos y explorando sus aplicaciones [22].

Para manejar conjuntos de datos extensos se ha aprovechado de las redes neuronales artificiales para discernir patrones. Las redes neuronales artificiales son algoritmos que han permitido a las computadoras aprender de los conjuntos de datos. La manera en la que aprenden de los datos ha sido imitando conexiones neuronales, inspirándose en los organismos biológicos.

Se sabe que las redes neuronales artificiales se han estructurado por nodos y pesos interconectados. El funcionamiento ha sido descrito como señales de entrada que pasan a través de las neuronas, se activan a través de una función y después se ponderan para finalmente tener una señal de salida como producto de este proceso. Entonces, las redes neuronales artificiales tienen un rendimiento el cual es evaluado mediante una métrica llamada pérdida. Los errores de un modelo basado en redes neuronales son cuantificados mediante la métrica conocida como pérdida [22].

Pérdida de entrenamiento

La presente métrica ha sido utilizada para medir la conformidad del modelo con los datos de entrenamiento al evaluar los errores dentro del conjunto de entrenamiento. Para calcular esta

métrica se ha procedido a sumar los errores en ejemplos individuales dentro de un conjunto de entrenamiento. Si se desea realizar la medición, tener en cuenta que se lo debe hacer después de cada lote [22].

Pérdida de validación

A diferencia de la pérdida de entrenamiento, consiste en la evaluación del rendimiento del modelo en un conjunto de validación que esté apartado, diferente del conjunto de datos de entrenamiento. De igual forma que la pérdida por entrenamiento se ha calculado a través de la suma de errores, en ejemplos dentro del conjunto de validación. Con la finalidad de determinar si se ha requerido ajustes en el modelo, la pérdida de validación se evalúa después de cada época [22].

Consecuencias de la pérdida de entrenamiento y validación

Con la finalidad de diagnosticar el rendimiento de un modelo se ha visualizado las pérdidas de entrenamiento y validación. Una vez que se ha visualizado las pérdidas se ha podido diagnosticar tres escenarios diferentes [22]:

1. **Sub-ajuste:** El presente escenario se ha desencadenado cuando el modelo ha capturado de forma insuficiente los patrones en el conjunto de entrenamiento. Esto implica que la pérdida de entrenamiento sea más baja que la pérdida de validación. Entonces, se puede catalogar este escenario como poco deseable. Algunas alternativas para resolver este resultado poco deseable son: un entrenamiento adicional o una ampliación del conjunto de datos. Observar en Figura 4.
2. **Sobreajuste:** Se ha presentado cuando el modelo ha sido incapaz de generalizar datos nuevos. Las consecuencias de este escenario han sido: alta pérdida de validación y baja pérdida de entrenamiento. Una metodología para contrarrestar el sobreajuste ha sido la detención del entrenamiento al momento de la estabilización del entrenamiento. Observar en Figura 5.
3. **Ajuste Bueno:** Este es el mejor escenario posible debido a que el ajuste del modelo ha sido el óptimo. Se ha logrado un equilibrio entre el sobreajuste y el subajuste. Observar en Figura 6.

Gráficas de pérdidas de entrenamiento y validación

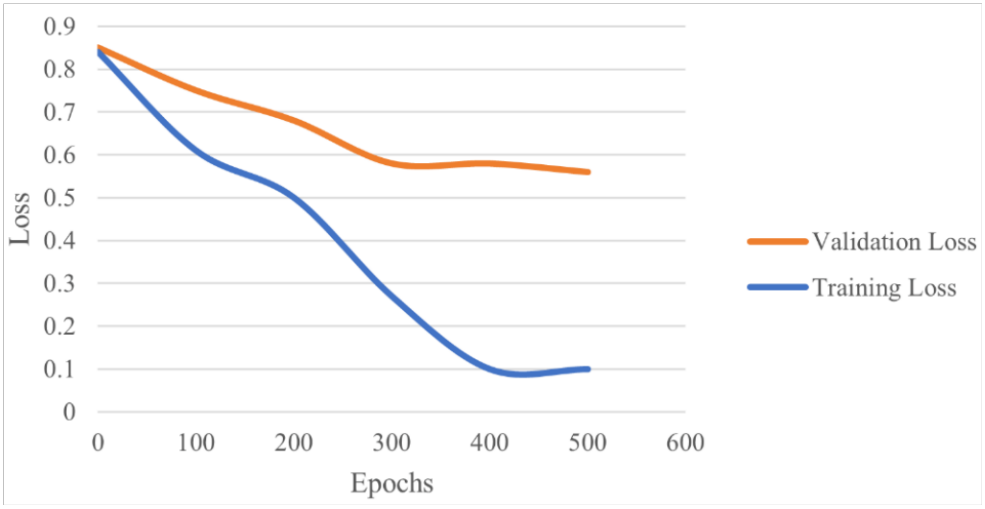


Figura 4 Subajuste

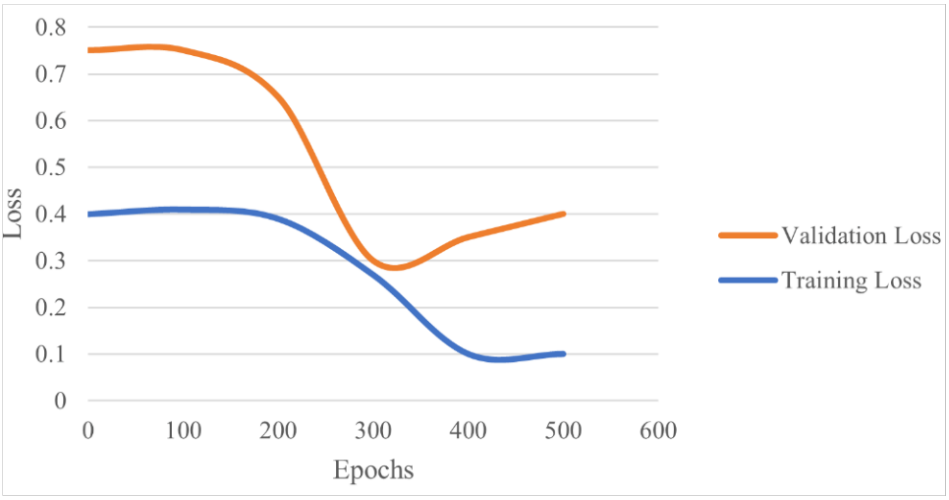


Figura 5 Sobreajuste

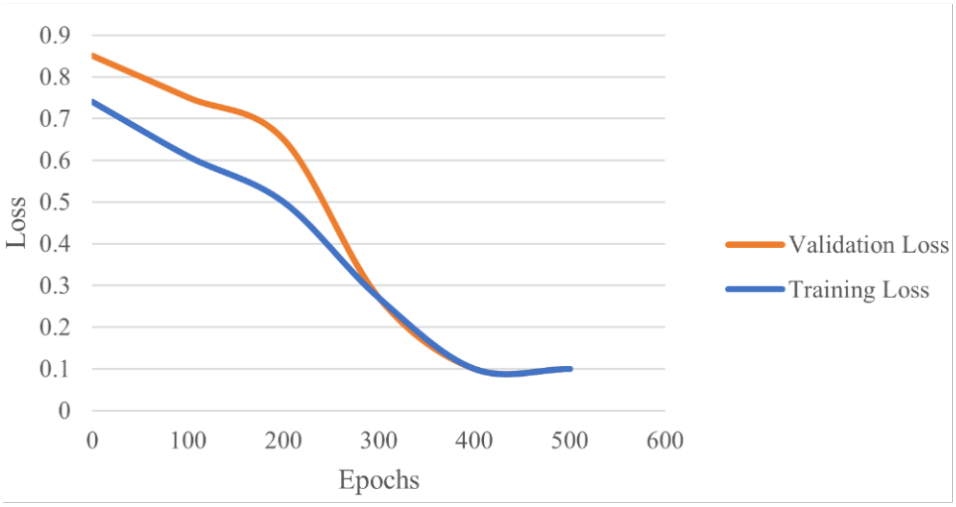


Figura 6 Buen ajuste

2 METODOLOGÍA

El presente capítulo detalla el procedimiento que se ha llevado a cabo para realizar el aumento de datos de señales de eventos sísmicos del volcán Cotopaxi. En el primer paso se obtuvo el conjunto de datos llamado MicSigV1 en formato mat. En el segundo paso se cambió de formato mat a json con la finalidad de usar Python como lenguaje de programación. En el tercer paso se ordenó y estructuró la información para facilitar su utilización. En el cuarto paso se extrajo la información relevante para realizar el aumento de datos. En el quinto paso se alimentó la información relevante obtenida en el modelo NICE. En el sexto paso se obtienen las métricas del modelo. Finalmente, se obtiene el aumento de datos de las señales de eventos sísmicos del volcán Cotopaxi.

Para detallar cada uno de los pasos presentes en el aumento de datos se utilizó el Modelo y Notación de Procesos de Negocio (BPMN). BPMN es un modelo estándar empleado para comprender y comunicar cualquier proceso haciendo uso de la notación gráfica. Debido a la notación gráfica empleada, se facilita la comprensión de los diferentes procesos realizados.

2.1 PROCESO PARA LA OBTENCIÓN DE LA BASE DE DATOS

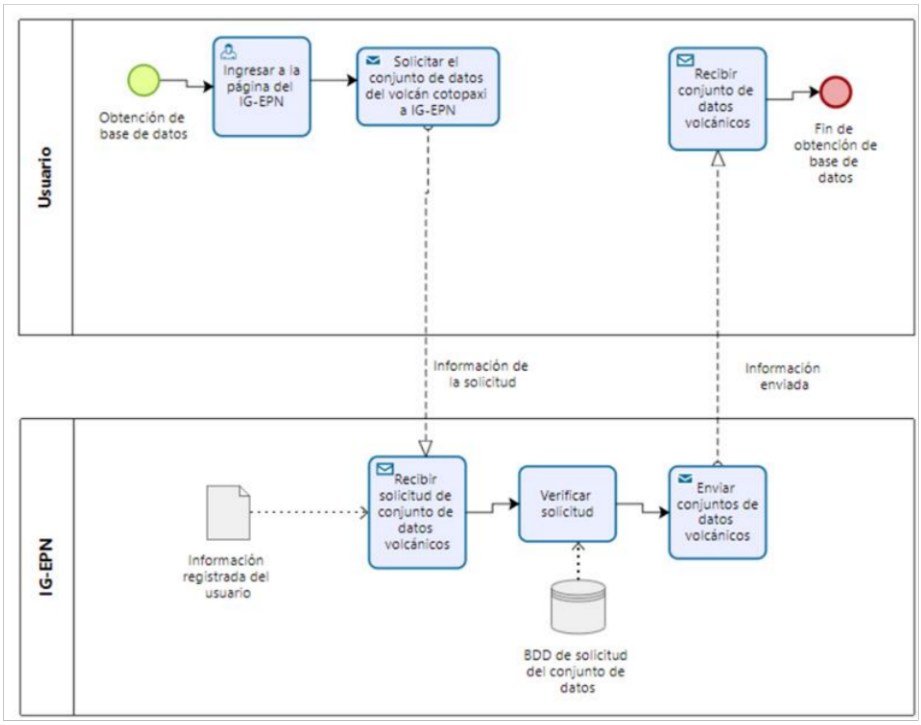


Figura 7 Proceso de obtención de la base de datos.

En el procedimiento a seguir de la Figura 7 se observa la relación existente entre el usuario y el Instituto Geofísico de la Escuela Politécnica Nacional.

En el proceso de obtención de la base de datos de las señales sísmicas del volcán Cotopaxi ha sido crucial la interacción con el Instituto Geofísico. Debido a que el Instituto Geofísico dispone de los repositorios de datos de señales sísmicas, ha sido necesario solicitarles esta información. A través de la página web del IG-EPN se ha realizado la solicitud del repositorio de señales sísmicas del volcán Cotopaxi (ESeismic) [23]. Finalmente se ha obtenido la base de datos sísmicas del volcán Cotopaxi en formato mat. Una vez finalizada la fase de obtención de datos se ha procedido con el tratamiento del conjunto de datos obtenido.

2.2 PROCESO PARA EL CAMBIO DE FORMATO DE DATOS

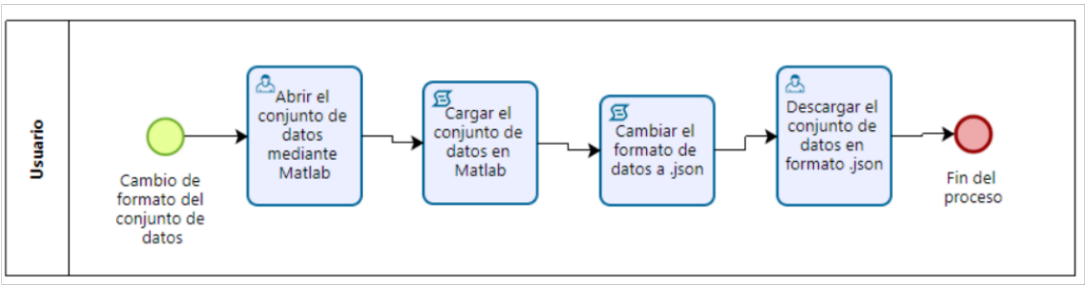


Figura 8 Proceso de cambio de formato del conjunto de datos.

En el procedimiento a seguir de la figura 8 se ha desarrollado el proceso para cambiar el formato de datos de mat a json mediante Matlab en línea. El proceso ha empezado desde la apertura del conjunto de datos mediante Matlab. Dentro del entorno de Matlab se ha desarrollado un script que ha permitido la carga del conjunto de datos. Después, se ha cambiado el formato de datos a json empleando otro script. Finalmente, el usuario ha descargado el conjunto de datos en el formato deseado para su posterior análisis.

2.3 PROCESO DE ESTRUCTURACIÓN Y ORDENAMIENTO

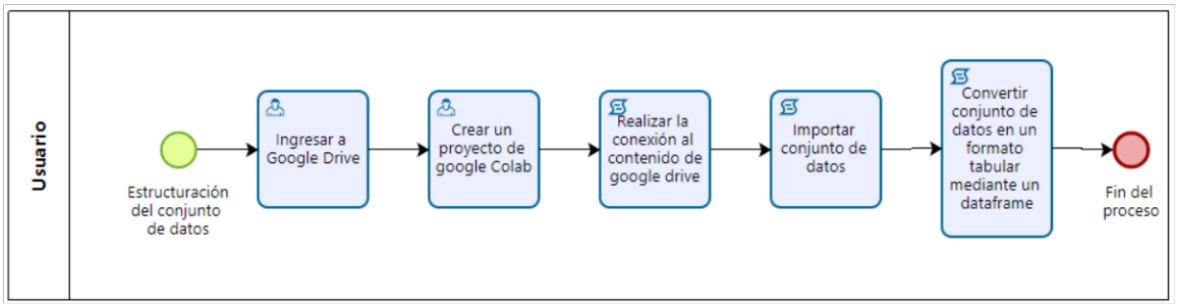


Figura 9 Proceso de estructuración del conjunto de datos.

En el procedimiento a seguir de la Figura 9 se ha desarrollado el proceso para estructurar el conjunto de datos. Como tarea de usuario se ha ingresado a Google Drive. Después, el usuario ha creado un proyecto de Google Colab. A continuación, se ha realizado la conexión al

contenido de Google Drive desde el entorno de Google Colab mediante un script. A través de un script se ha importado el conjunto de datos que en el proceso anterior se obtuvo por parte del usuario. Finalmente, se ha convertido el conjunto de datos en un formato de tabla, con la finalidad de visualizar de mejor manera el conjunto de datos, empleando un script.

2.4 PROCESO PARA EL TRATAMIENTO DE DATOS

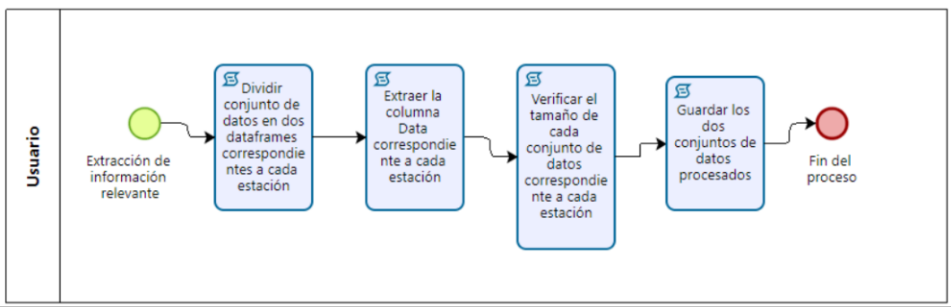


Figura 10 Proceso de extracción de información relevante.

En el procedimiento a seguir de la figura 10 se ha desarrollado el proceso para la extracción de información relevante. Se ha dividido el conjunto de datos en dos tablas que corresponden a cada estación, mediante un script. Las estaciones son: VC1 y BREF. Se ha extraído la columna Data de cada tabla (VC1 y BREF) a través de un script. Después, se ha verificado el tamaño de cada conjunto de datos. Para la verificación del tamaño se ha comprobado si las dimensiones de cada conjunto de datos son las deseadas. Entonces, se ha examinado cada conjunto de datos para evaluar el número de filas y número de columnas presentes. Lo que se ha requerido previo al proceso de entrenamiento de los conjuntos de datos, es mantener las dimensiones compatibles de los conjuntos de datos correspondientes a VC1 y BREF.

2.5 PROCESO DE ENTRENAMIENTO

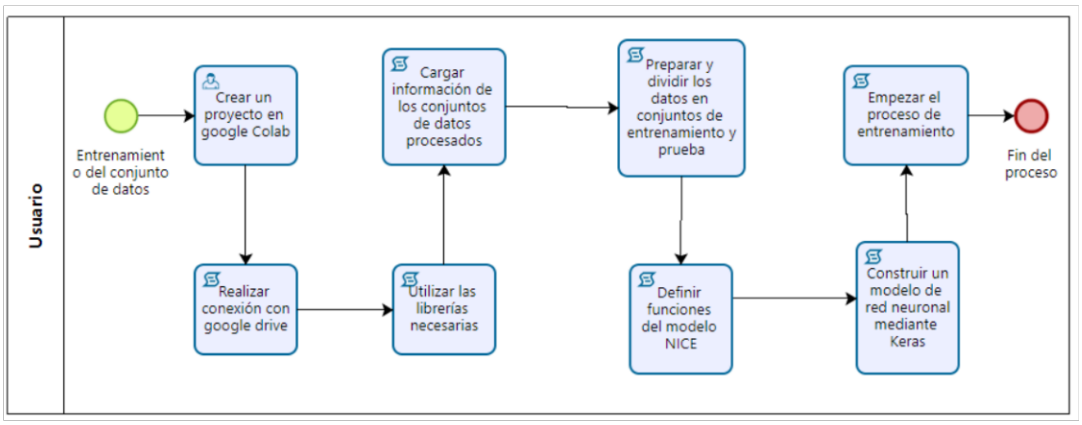


Figura 11 Proceso de entrenamiento del conjunto de datos.

En el procedimiento a seguir de la figura 11 se ha desarrollado el proceso de entrenamiento del conjunto de datos procesado, correspondiente a VC1 y BREF. Por parte del usuario se ha creado un segundo proyecto en Google Colab. A continuación, se ha realizado la conexión con Google Drive mediante un script. Después, empleando un script se ha definido las librerías necesarias para el entrenamiento de los conjuntos de datos. Una vez que se ha definido las librerías, se ha cargado la información de los conjuntos de datos empleando un script. Se ha preparado y dividido los datos en conjuntos de entrenamiento y prueba. Después, se ha definido las funciones del modelo NICE a través de otro script. Se ha construido un modelo de red neuronal para cada conjunto de datos previo al proceso de entrenamiento. Finalmente, se ha empezado el proceso de entrenamiento.

2.6 PROCESO DE OBTENCIÓN DE MÉTRICAS

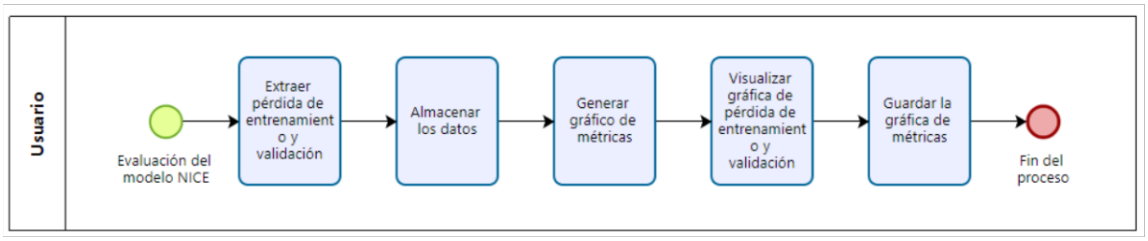


Figura 12 Proceso de evaluación del modelo NICE.

En el procedimiento a seguir de la figura 12 se ha desarrollado el proceso de evaluación del modelo NICE, correspondiente a VC1 y BREF. Tras completar el proceso de entrenamiento, se ha empleado las pérdidas de entrenamiento y validación mediante un script, seguido por el almacenamiento de los datos para su posterior visualización. Se ha generado las gráficas con las siguientes métricas: pérdida de entrenamiento y pérdida de validación. Luego, mediante un script, se ha creado un gráfico que muestra las curvas de las métricas planteadas. Finalmente, se ha guardado la gráfica resultante.

2.7 PROCESO PARA EL AUMENTO DE DATOS

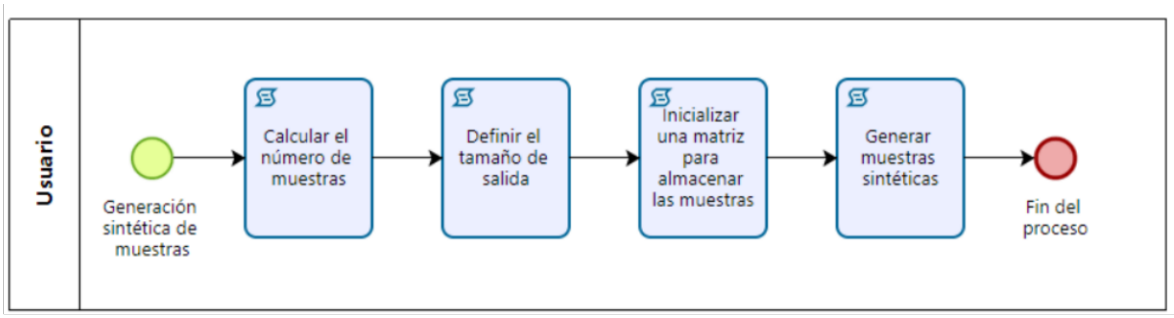


Figura 13 Proceso de generación sintética de muestras.

En el procedimiento a seguir de la figura 13 se ha desarrollado el proceso de aumento de datos del modelo NICE, correspondiente a VC1 y BREF. Se ha calculado el número de muestras mediante un script. Después se ha definido el tamaño de la salida en la generación de muestras. Se ha inicializado una matriz para almacenar las muestras que serán generadas. Finalmente, se han generado las muestras sintéticas.

3 RESULTADOS

Esta sección presenta los resultados obtenidos desde la obtención de la base de datos hasta la generación de datos sintéticos de los conjuntos de datos que corresponden a “VC1” y “BREF”, las cuales representan a estaciones de la red sismológica desplegadas en el volcán Cotopaxi. Dentro de los resultados se ha realizado un análisis de las siguientes métricas: pérdida de entrenamiento y pérdida de validación.

La métrica de pérdida de entrenamiento se la ha empleado para evaluar el ajuste del modelo a los datos del entrenamiento midiendo su error en el conjunto de datos correspondiente. Por otro lado, la métrica de la pérdida de validación se ha evaluado para observar que tan bien el modelo ha podido generalizar datos que no se han visto. En conclusión, le pérdida de entrenamiento se ha empleado para evaluar el rendimiento del modelo en los datos de entrenamiento, mientras que la pérdida de validación se ha utilizado para verificar la capacidad de generalizar datos nuevos.

Finalmente, se ha realizado el proceso de generación de datos sintéticos. En este apartado se ha mostrado mediante tablas la generación de datos nuevos. También se ha realizado un análisis entre la generación de datos correspondientes a VC1 y BREF.

3.1 CAMBIO DE FORMATO DE DATOS

Los resultados que se han obtenido en el cambio de formato de datos son:

1. Se ha cargado la base de datos “MicSigV1_with_spectrogramsTemplate”.

Código 1 Base de datos cargada en formato mat

```
%Cargo base de datos .mat
data=load("MicSigV1_with_spectrogramsTemplate.mat")
```

2. Se ha cambiado el formato a json y se ha guardado el nuevo conjunto de datos con el nombre datos.

Código 2 Cambio de formato de base de datos a json.

```
%Convertir datos a formato json

jsonData = jsonencode(data);
%guardar datos en .json
fid = fopen('MicSigV1_with_spectrogramsTemplate.json', 'w');
fwrite(fid, jsonData, 'char');
%%Guardado de datos en mi sistema de archivos local para posterior descarga
% Abre un archivo para escritura en modo texto
fid = fopen('datos.json', 'w');
% Escribe los datos JSON en el archivo
fprintf(fid, '%s', jsonData);
```

3. Finalmente se ha descargado el nuevo conjunto de datos llamado “datos.json” para su posterior procesamiento.

3.2 ORDENAMIENTO Y ESTRUCTURACIÓN DE LA BASE DE DATOS

Los resultados que se han obtenido en el ordenamiento y estructuración de la base de datos son:

1. Se ha habilitado la carga y descarga de archivos en el entorno de Google Colab y se ha permitido al usuario el acceso a archivos almacenados en Google drive desde el entorno de Google Colab.
2. Se ha importado bibliotecas de Python utilizadas para: trabajar con datos tipo json, escribir y leer datos en formato .mat, manipular y analizar datos estructurados y trabajar con matrices y arreglos multidimensionales.
3. Se ha cargado un archivo de tipo json llamado “data” mediante Google Drive. Después se ha extraído una columna específica llamada MicSigV1. Finalmente, se ha convertido el conjunto de datos en un DataFrame empleando la biblioteca pandas.

Código 3 Visualización del conjunto de datos

```
df=pd.read_json('/content/drive/MyDrive/data.json')
bn=pd.DataFrame(df.MicSigV1.values.tolist())
pd.DataFrame.from_records(bn).head()
```

	0	1	2	3	4	5	6	7	8	9	...
0	Network	Station	SampleRate	Component	Year	Month	Type	Duration	StartPoint	EndPoint	...
1	[[EC], [VC1], 100, [SHZ], 2012, 1, [VT], 23, 1...	[[EC], [VC1], 100, [SHZ], 2012, 1, [LP], 47, 1...	[[EC], [VC1], 100, [SHZ], 2012, 1, [LP], 56, 1...	[[EC], [VC1], 100, [SHZ], 2012, 1, [VT], 49, 1...	[[EC], [VC1], 100, [SHZ], 2012, 1, [VT], 31, 1...	[[EC], [VC1], 100, [SHZ], 2012, 1, [LP], 29, 1...	[[EC], [VC1], 100, [SHZ], 2012, 1, [LP], 33, 1...	[[EC], [VC1], 100, [SHZ], 2012, 1, [LP], 42, 1...	[[EC], [VC1], 100, [SHZ], 2012, 1, [LP], 24, 1...	[[EC], [VC1], 100, [SHZ], 2012, 1, [LP], 30, 1...	...
2	None	None	None	None	None	None	None	None	None	None	...

3 rows × 1187 columns

Figura 14 Visualización del conjunto de datos sin ordenar.

4. Se ha procedido a estructurar y ordenar la tabla. Mediante el uso de librerías de Python se ha realizado varias operaciones sobre el conjunto de datos. Finalmente se ha obtenido la tabla ordenada como se puede apreciar en la Figura 11.

	Network	Station	SampleRate	Component	Year	Month	Type	Duration	StartPoint	EndPoint	Data	SpecImageResize	signals_t_trimmed_resampled_meanremoved_normalized	image_name
0	[EC]	[VC1]	100	[SHZ]	2012	1	[VT]	23	1001	3301	[23.35552995609319, -26.21249638722304, 7.4787...	[[43, 43, 44, 45, 46, 46, 47, 49, 50, 50, 50, ...	[0.01185078369979397, 0.01181335287391387, -0....	VC1-VT-0001.png
1	[EC]	[VC1]	100	[SHZ]	2012	1	[LP]	47	1001	5701	[-9.39456702711197, 5.797439374378023, -23.901...	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...	[0.0137374020391823, -0.09066534913637994, -0....	VC1-LP-0002.png
2	[EC]	[VC1]	100	[SHZ]	2012	1	[LP]	56	1001	6601	[-3.915706523024277, 0.4205326687697789, -21.3...	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...	[0.004317625342252428, 0.03917574996119901, 0....	VC1-LP-0003.png
3	[EC]	[VC1]	100	[SHZ]	2012	1	[VT]	49	1001	5901	[-2.568625465852772, 23.91770870655175, -10.74...	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...	[0.006520690350132239, 0.0205035440649939, 0.0...	VC1-VT-0004.png
4	[EC]	[VC1]	100	[SHZ]	2012	1	[VT]	31	1001	4101	[55.05791051352128, 41.33929940327658, -19.424...	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...	[0.011373826473140 58, 0.0069589409478376 97, -0...	VC1-VT-0005.png
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1182	[EC]	[BREF]	50	[BHZ]	2019	3	[LP]	23	501	1651	[-5.580699845211789, -3.212906366210063, 34.67...	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...	[0.0263011655048726 1, 0.0246870372367498 7, 0.0...	BREF-LP-1183.png
1183	[EC]	[BREF]	50	[BHZ]	2019	3	[LP]	28	501	1901	[56.59814278772568, 35.85003980602864, 7.89771...	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...	[0.004162638650843283, -0.00123336382466905, -...	BREF-LP-1184.png
1184	[EC]	[BREF]	50	[BHZ]	2019	3	[LP]	46	501	2801	[2.387041324869946, -64.67153614317736, -62.47...	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...	[0.0029086026182451 89, -0.0004809937169418903,...	BREF-LP-1185.png
1185	[EC]	[BREF]	50	[BHZ]	2019	3	[LP]	32	501	2101	[-22.84205399680515, -27.18911304867139, -27.6...	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...	[0.022423584812535 66, -0.0116464090569898 8, 0....	BREF-LP-1186.png
1186	[EC]	[BREF]	50	[BHZ]	2019	3	[LP]	28	501	1901	[7.089124578261767, 13.85918934305973, 12.2714...	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...	[0.021711666655923 2, 0.0118419302426565 1, 0.00...	BREF-LP-1187.png

1187 rows × 14 columns

Figura 15 Visualización del conjunto de datos ordenado

```
Data columns (total 14 columns):
 #   Column                                        Non-Null Count  Dtype
---  ------                                        --------------  -----
 0   Network                                       1187 non-null   object
 1   Station                                       1187 non-null   object
 2   SampleRate                                    1187 non-null   int64
 3   Component                                     1187 non-null   object
 4   Year                                          1187 non-null   int64
 5   Month                                         1187 non-null   int64
 6   Type                                          1187 non-null   object
 7   Duration                                      1187 non-null   int64
 8   StartPoint                                    1187 non-null   int64
 9   EndPoint                                      1187 non-null   int64
 10  Data                                          1187 non-null   object
 11  SpecImageResize                               1187 non-null   object
 12  signals_t_trimmed_resampled_meanremoved_normalized  1187 non-null   object
 13  image_name                                    1187 non-null   object
dtypes: int64(6), object(8)
memory usage: 130.0+ KB
None
```

Figura 16 Información general del conjunto de datos ordenado

5. Una vez que se ha ordenado la información del conjunto de datos correspondiente, se ha procedido a verificar la integridad de los datos. Una forma de analizar la integridad de los datos ha sido la verificación de valores faltantes. Se ha utilizado el método “isna()” junto a “sum()” para contar el número de valores nulos de cada columna. Lo que se ha buscado es verificar que no existan valores nulos en el conjunto de datos. A continuación, se ha visualizado que no existen valores nulos.

```
Network                                                  0
Station                                                  0
SampleRate                                               0
Component                                                0
Year                                                     0
Month                                                    0
Type                                                     0
Duration                                                 0
StartPoint                                               0
EndPoint                                                 0
Data                                                     0
SpecImageResize                                          0
signals_t_trimmed_resampled_meanremoved_normalized       0
image_name                                               0
dtype: int64
```

Figura 17 Número de valores nulos en cada columna del conjunto de datos

3.3 EXTRACCIÓN DE INFORMACIÓN RELEVANTE

Los resultados que se han obtenido en la extracción de información relevante son:

1. Se ha procedido a dividir el Dataframe observado en la ilustración 13 por estaciones. El conjunto de datos consta de dos estaciones que son “VC1” y “BREF”.
2. Como primer paso se ha procedido a obtener el conjunto de datos correspondiente a la información de la estación VC1. Para esto, se ha llevado a cabo la identificación de los registros que corresponden a la estación VC1. Una vez identificados los registros, se ha separado del conjunto de datos original. Finalmente, se ha procedido a visualizar el nuevo conjunto de datos obtenido que consta solamente de la estación VC1.

Código 4 Código para obtener un dataframe correspondiente a la estación VC1

```
df_vc1 = df.head(481)#Identificación de los registros correspondientes a VC1
display(df_vc1)#Visualización del nuevo conjunto de datos de la estación VC1
```

	Network	Station	SampleRate	Component	Year	Month	Type	Duration	StartPoint	EndPoint	Data	SpecImageResize	signals_t_trimmed_resampled_meanremoved_normalized	image_name
0	[EC]	[VC1]	100	[SHZ]	2012	1	[VT]	23	1001	3301	[23.35552995609319, -26.21249638722304, 7.4787...	[[43, 43, 44, 45, 46, 46, 47, 49, 50, 50, 50, ...	[0.01185078369979397, 0.01181335287391387, -0....	VC1-VT-0001.png
1	[EC]	[VC1]	100	[SHZ]	2012	1	[LP]	47	1001	5701	[-9.39456702711197, 5.797439374378023, -23.901...	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...	[0.0137374020391823, -0.0906653491363799 4, -0....	VC1-LP-0002.png
2	[EC]	[VC1]	100	[SHZ]	2012	1	[LP]	56	1001	6601	[-3.915706523024277, 0.4205326687697789, -21.3...	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...	[0.004317625342252428, 0.03917574996119901, 0....	VC1-LP-0003.png
3	[EC]	[VC1]	100	[SHZ]	2012	1	[VT]	49	1001	5901	[-2.568625465852772, 23.91770870655175, -10.74...	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...	[0.006520690350132239, 0.0205035440649939, 0.0...	VC1-VT-0004.png
4	[EC]	[VC1]	100	[SHZ]	2012	1	[VT]	31	1001	4101	[55.05791051352128, 41.33929940327658, -19.424...	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...	[0.0113738264731405 8, 0.006958940947837697, -0...	VC1-VT-0005.png
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
476	[EC]	[VC1]	100	[SHZ]	2014	1	[LP]	40	1001	5001	[-34.83302294297383, 10.17518748846738, 4.2373...	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...	[0.0229387330198906 3, 0.05381989018017178, 0.0...	VC1-LP-0477.png
477	[EC]	[VC1]	100	[SHZ]	2014	1	[LP]	44	1001	5401	[-44.49859116121449, 3.340577950940697, -22.29...	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...	[-0.013996361639052 35, -0.01041119895154159, 0...	VC1-LP-0478.png
478	[EC]	[VC1]	100	[SHZ]	2014	1	[LP]	45	1001	5501	[-19.09053644391337, -24.08748167250992, -8.49...	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...	[0.012342048733924 82, 0.0306256070209868 6, 0.0...	VC1-LP-0479.png
479	[EC]	[VC1]	100	[SHZ]	2014	1	[LP]	46	1001	5601	[-12.90267466968753, -22.22163760197435, 0.314...	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...	[-0.025937864318491 23, -0.029603968078447 88, -...	VC1-LP-0480.png
480	[EC]	[VC1]	100	[SHZ]	2014	1	[LP]	56	1001	6601	[-10.99456974364485, -22.5335529422007, -29.50...	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...	[0.008040939936245 111, -0.007406914437518573, ...	VC1-LP-0481.png

481 rows × 14 columns

Figura 18 Visualización del conjunto de datos correspondiente a VC1

```
Data columns (total 14 columns):
 #   Column                                          Non-Null Count  Dtype
---  ------                                          --------------  -----
 0   Network                                         481 non-null    object
 1   Station                                         481 non-null    object
 2   SampleRate                                      481 non-null    int64
 3   Component                                       481 non-null    object
 4   Year                                            481 non-null    int64
 5   Month                                           481 non-null    int64
 6   Type                                            481 non-null    object
 7   Duration                                        481 non-null    int64
 8   StartPoint                                      481 non-null    int64
 9   EndPoint                                        481 non-null    int64
 10  Data                                            481 non-null    object
 11  SpecImageResize                                 481 non-null    object
 12  signals_t_trimmed_resampled_meanremoved_normalized  481 non-null  object
 13  image_name                                      481 non-null    object
dtypes: int64(6), object(8)
memory usage: 52.7+ KB
None
```

Figura 19 Información general correspondiente al conjunto de datos de la estación VC1

3. Como segundo paso se ha procedido a obtener el conjunto de datos correspondiente a la información de la estación BREF. Para esto, se ha llevado a cabo la identificación de los registros que corresponden a la estación BREF. Una vez identificados los registros, se ha separado del conjunto de datos original. Finalmente, se ha procedido a visualizar el nuevo conjunto de datos obtenido que consta solamente de la estación BREF.

Código 5 Código para obtener un dataframe correspondiente a la estación BREF

```
df_bref = df.iloc[481:1187]#Identificacion de los registros correspondientes a BREF
display(df_bref)#Visualización del nuevo conjunto de datos de la estación BREF
```

	Network	Station	SampleRate	Component	Year	Month	Type	Duration	StartPoint	EndPoint	Data	SpecImageResize	signals_t_trimmed_resampled_meanremoved_normalized	image_name
481	[EC]	[BREF]	50	[BHZ]	2018	1	[LP]	19	501	1451	[-3.277797624252992, 24.95103546320621, -2.354...	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...	[-0.01181770410911597, -0.05593226299908337, -...	BREF-LP-0482.png
482	[EC]	[BREF]	50	[BHZ]	2018	1	[LP]	37	501	2350	[10.915903251610 01, 45.93796770601318, 26.7352...	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...	[0.004698957904464334, 0.008145070498134151, 0...	BREF-LP-0483.png
483	[EC]	[BREF]	50	[BHZ]	2018	1	[LP]	32	501	2101	[-18.82262519640063, -4.377671730037062, 23.47...	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...	[0.001889917932097893, -0.002100262250834148, ...	BREF-LP-0484.png
484	[EC]	[BREF]	50	[BHZ]	2018	1	[LP]	128	501	6901	[30.5755827043704, -5.677898601730965, -25.303...	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...	[0.01390862459655947, 0.04021041069016312, 0.0...	BREF-LP-0485.png
485	[EC]	[BREF]	50	[BHZ]	2018	1	[LP]	32	501	2099	[57.49351516939352, 37.61314450013, -6.3374877...	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...	[0.00817882489565203, 0.008692624409318972, 0....	BREF-LP-0486.png
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1182	[EC]	[BREF]	50	[BHZ]	2019	3	[LP]	23	501	1651	[-5.580699845211789, -3.212906366210063, 34.67...	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...	[0.026301165504872 61, 0.0246870372367 4987, 0.0...	BREF-LP-1183.png
1183	[EC]	[BREF]	50	[BHZ]	2019	3	[LP]	28	501	1901	[56.59814278772568, 35.85003980602864, 7.89771...	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...	[0.004162638650843283, -0.00123336382466905, -...	BREF-LP-1184.png
1184	[EC]	[BREF]	50	[BHZ]	2019	3	[LP]	46	501	2801	[2.387041324869946, -64.67153614317736, -62.47...	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...	[0.002908602618245189, -0.0004809937169418903,...	BREF-LP-1185.png
1185	[EC]	[BREF]	50	[BHZ]	2019	3	[LP]	32	501	2101	[-22.84205399680515, -27.189113048671 39, -27.6...	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...	[0.022423584812535 66, -0.011646409056989 68, 0....	BREF-LP-1186.png
1186	[EC]	[BREF]	50	[BHZ]	2019	3	[LP]	28	501	1901	[7.089124578261767, 13.85918934305973, 12.2714...	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...	[0.021711666655923 2, 0.011841930242656 51, 0.00...	BREF-LP-1187.png

706 rows × 14 columns

Figura 20 Visualización del conjunto de datos correspondiente a BREF

```
Data columns (total 14 columns):
 #   Column                                                Non-Null Count  Dtype
---  ------                                                --------------  -----
 0   Network                                               706 non-null    object
 1   Station                                               706 non-null    object
 2   SampleRate                                            706 non-null    int64
 3   Component                                             706 non-null    object
 4   Year                                                  706 non-null    int64
 5   Month                                                 706 non-null    int64
 6   Type                                                  706 non-null    object
 7   Duration                                              706 non-null    int64
 8   StartPoint                                            706 non-null    int64
 9   EndPoint                                              706 non-null    int64
 10  Data                                                  706 non-null    object
 11  SpecImageResize                                       706 non-null    object
 12  signals_t_trimmed_resampled_meanremoved_normalized    706 non-null    object
 13  image_name                                            706 non-null    object
dtypes: int64(6), object(8)
memory usage: 77.3+ KB
None
```

Figura 21 Información general correspondiente al conjunto de datos de la estación BREF

4. A continuación, se ha extraído la columna Data correspondiente a cada estación. Para ambos casos se ha seguido un proceso similar. Como primer paso se ha definido los nombres de las columnas a eliminar. Como segundo paso se ha creado una variable para cada conjunto de datos que guarda información del respectivo conjunto de datos. Finalmente, se ha visualizado el nuevo conjunto de datos con la información deseada tanto para VC1 como para BREF.

	Data
481	[-3.277797624252992, 24.95103546320621, -2.354...
482	[10.91590325161001, 45.93796770601318, 26.7352...
483	[-18.82262519640063, -4.377671730037062, 23.47...
484	[30.5755827043704, -5.677898601730985, -25.303...
485	[57.49351516939352, 37.61314450013, -6.3374877...
...	...
1182	[-5.580699845211789, -3.212906366210063, 34.67...
1183	[56.59814278772568, 35.85003980602864, 7.89771...
1184	[2.387041324869946, -64.67153614317736, -62.47...
1185	[-22.84205399680515, -27.18911304867139, -27.6...
1186	[7.089124578261767, 13.85918934305973, 12.2714...

706 rows × 1 columns

Figura 22 Visualización de la columna Data correspondiente a la estación BREF

	Data
0	[23.35552995609319, -26.21249638722304, 7.4787...
1	[-9.39456702711197, 5.797439374378023, -23.901...
2	[-3.915706523024277, 0.4205326687697789, -21.3...
3	[-2.568625465852772, 23.91770870655175, -10.74...
4	[55.05791051352128, 41.33929940327658, -19.424...
...	...
476	[-34.83302294297383, 10.17518748846738, 4.2373...
477	[-44.49859116121449, 3.340577950940697, -22.29...
478	[-19.09053644391337, -24.08748167250992, -8.49...
479	[-12.90267466988753, -22.22163760197435, 0.314...
480	[-10.99456974364485, -22.5335529422007, -29.50...

481 rows × 1 columns

Figura 23 Visualización de la columna Data correspondiente a la estación VC1

5. Como siguiente paso se ha igualado la longitud de cada lista de la columna Data de los conjuntos de datos correspondientes a VC1 y BREF. Para igualar la longitud de todas las listas se ha rellenado cada lista con ceros hasta igualar a la lista con mayor longitud. Una vez que se ha obtenido la misma longitud en todas las listas, se ha procedido a dividir en columnas cada uno de los conjuntos de datos para su posterior análisis. A continuación, se detallará el proceso para cada conjunto de datos.
6. Se ha ajustado la longitud de las listas del conjunto de datos VC1 de acuerdo con la longitud máxima de las listas de la estación BREF. Las listas a las que se han aumentado su longitud han sido completadas con ceros. Después se ha convertido las listas en columnas separadas. A continuación, se ha eliminado una columna con la finalidad de tener un número par de columnas. Finalmente, se ha impreso los resultados obtenidos

Código 6 Ajuste de longitud de listas y conversión de listas a columnas del conjunto de datos VC1

```
# Longitud deseada
desired_length = 40501
# Ajustar la longitud de todas las listas a la longitud deseada, llenando con
ceros
df_vc4['Data'] = df_vc4['Data'].apply(lambda x: x + [0] * (desired_length -
len(x)))
# Convertir las listas en columnas separadas
df_expanded_vc2 = pd.DataFrame(df_vc4['Data'].to_list(), columns=[f'Data_{i}'
for i in range(1, len(df_vc4['Data'].iloc[0])+1)])
print("Convertir las listas en columnas separadas")
display(df_expanded_vc2)
#Eliminar una columna para tener un número par de columnas
df_expanded_vc2=df_expanded_vc2.drop(df_expanded_vc2.columns[-1], axis=1)
display(df_expanded_vc2)
```

7. Se ha ajustado la longitud de las listas del conjunto de datos BREF de acuerdo con la longitud máxima de sus listas. Las listas a las que se han aumentado su longitud fueron completadas con ceros. Después se ha convertido las listas en columnas separadas. A continuación, se ha eliminado una columna con la finalidad de tener un número par de columnas. Finalmente, se ha impreso los resultados obtenidos.

Código 7 Ajuste de longitud de listas y conversión de listas a columnas del conjunto de datos BREF

```
# Encontrar la longitud máxima en la columna 'Data'
max_length_bref3 = df_bref3['Data'].apply(len).max()
print(max_length_bref3)
# Aumentar la longitud de todas las listas a la longitud máxima, llenando con
ceros
df_bref3['Data'] = df_bref3['Data'].apply(lambda x: x + [0] *
(max_length_bref3 - len(x)))
# Convertir las listas en columnas separadas
df_expanded_bref3 = pd.DataFrame(df_bref3['Data'].to_list(),
columns=[f'Data_{i}' for i in range(1, len(df_bref3['Data'].iloc[0])+1)])
#Eliminar una columna para tener un número par de columnas (40500)
df_expanded_bref4=df_expanded_bref3.drop(df_expanded_bref3.columns[-1],
axis=1)
```

8. Finalmente, se ha procedido a guardar los conjuntos de datos correspondientes a VC1 y BREF que fueron procesados.

Código 8 Guardado de los conjuntos de datos procesados correspondientes a las estaciones VC1 y BREF

```
#Guardado de conjunto de datos procesado correspondiente a la estación VC1
df_expanded_vc2.to_json('df_vc2_json_separated.json', orient='records',
lines=True)
#Guardado del conjunto de datos procesado correspondiente a BREF
df_expanded_bref4.to_json('df_bref3_json_separated.json', orient='records',
lines=True)
```

3.4 ENTRENAMIENTO DEL CONJUNTO DE DATOS

Los resultados que se han obtenido en el entrenamiento del conjunto de datos son:

1. Se ha realizado una conexión con Google Drive de la misma manera como se lo ha realizado en “Código 3”.
2. Con la finalidad de construir y entrenar los conjuntos de datos obtenidos en “Código 12”, se ha utilizado Keras y TensorFlow. Keras ha sido utilizado para entrenar y construir modelos de redes neuronales. Con la finalidad de implementar un entorno de trabajo de aprendizaje automático, se ha hecho uso de la biblioteca Tensor Flow el cual ha sido desarrollado por Google [24]. En conclusión, se ha realizado las importaciones adecuadas para el manejo de datos, construcción de modelos de redes neuronales y entrenamiento en el entorno de Google Colab.
3. Se ha cargado la información de los conjuntos de datos procesados. Los datos cargados se almacenaron en dos variables las cuales son: df_vc2_json_separated y df_bref3_json_separated correspondientes a las estaciones VC1 y BREF.
4. Se ha preparado y dividido los datos en conjuntos de entrenamiento y prueba. Se ha utilizado el 80% de los datos para entrenamiento y el restante 20% para validación en cada conjunto de datos correspondiente a VC1 y BREF. Después, se ha concatenado los dos conjuntos de datos correspondientes a cada estación y se mezclan los datos de manera aleatoria. Finalmente, se han dividido los conjuntos de datos en entrenamiento y prueba en una proporción de 70% y 30%.

Código 9 División del conjunto de datos en entrenamiento y prueba

```
# Separar en entrenamiento y prueba con el 80% de los datos para entrenamiento
y el 20% para validación
x0_train =
df_vc2_json_separated[:int(np.rint(df_vc2_json_separated.shape[0]*0.8))]
x0_test =
df_vc2_json_separated[int(np.rint(df_vc2_json_separated.shape[0]*0.8)):]
print('Entrenamiento: ', len(x0_train))
print('Prueba: ', len(x0_test))
#Estacion 2
print("estacion 2", df_bref3_json_separated.shape)
# Separar en entrenamiento y prueba con el 80% de los datos para entrenamiento
y el 20% para validación
x1_train =
df_bref3_json_separated[:int(np.rint(df_bref3_json_separated.shape[0]*0.8))]
x1_test =
df_bref3_json_separated[int(np.rint(df_bref3_json_separated.shape[0]*0.8)):]
# Todas las estaciones concatenadas
total = np.vstack((df_vc2_json_separated, df_bref3_json_separated))
np.random.shuffle(total)
X_train = total[:int(np.rint(total.shape[0]*0.7))]
X_test = total[int(np.rint(total.shape[0]*0.7)):]
print("estaciones concatenadas", total.shape)
```

```
estaciones concatenadas (1187, 40500)
```

Figura 24 Dimensión de las estaciones concatenadas

5. Se ha definido las funciones del modelo NICE. Se ha establecido seis funciones en el modelo NICE las cuales se han empleado para la construcción del modelo de redes neuronales mediante. La primera función creada ha sido Shuffle que se ha utilizado para mezclar la dimensión de entrada de dos maneras las cuales son: inversión directa y aleatoria. La segunda función creada ha sido ConcatVector y se ha empleada para fusionar dos partes. La tercera función creada ha sido AddCouple la cual ha sido empleada para implementar una capa de acoplamiento aditivo. La cuarta función creada ha sido build_basic_model cuya finalidad ha sido construir un modelo base para la capa de acoplamiento aditivo. La quinta función creada ha sido SplitVector y se ha empleado para dividir la entrada en dos partes, fusionando las particiones. Finalmente, la última función creada ha sido Scale cuya finalidad ha sido implementar una capa de escala. Esto se muestra en el ANEXO I
6. Se ha creado una arquitectura de red neuronal empleando las funciones definidas anteriormente. En la presente arquitectura se han instanciado las siguientes capas:

Shuffle, SplitVector, AddCouple, ConcatVector, Scale. Se ha definido la dimensión de entrada que corresponde a 40500. Se han construido dos modelos utilizados en las capas de acoplamiento aditivo. Se ha definido un tensor de entrada que tiene la forma de la dimensión original. Se ha añadido ruido negativo en la entrada. La capa Shuffle ha sido utilizada para mezclar las dimensiones de entrada. Mediante la capa SplitVector se ha dividido el tensor que ha sido mezclado en dos partes. A través de un modelo básico, se ingresa una parte que ha sido dividida y se obtiene una salida. Después, se ha aplicado la capa de acoplamiento aditivo entre las dos partes. Nuevamente, se ha concatenado las dos partes mediante la capa ConcatVector.

7. En otras palabras, se ha implementado un modelo de red neuronal con funciones personalizadas que ejecutan las siguientes operaciones: mezcla, división, acoplamiento aditivo y escala.
8. Se ha empezado el proceso de entrenamiento. Se ha entrenado el modelo empleando los datos de entrenamiento y validación especificados. El entrenamiento es detenido en caso de que el conjunto de validación no mejore.

Código 10 Proceso de entrenamiento del conjunto de datos VC1

```
checkpoint_filepath = '/content/df_vc2_json_separated.json'
model_checkpoint_callback = tf.keras.callbacks.ModelCheckpoint(
    filepath=checkpoint_filepath,
    save_weights_only=True,
    monitor='val_loss',
    mode='max',
    save_best_only=True)

early_stop_callback = tf.keras.callbacks.EarlyStopping(monitor='val_loss',
patience=10)

history0 = encoder0.fit(x0_train,
            x0_train,
            batch_size=128,
            epochs=100,
            validation_data=(x0_test, x0_test),
callbacks=[model_checkpoint_callback, early_stop_callback])
```

```
Epoch 90/100
4/4 [==============================] - 18s 5s/step - loss: 58340056.0000 - val_loss: 59544916.0000
Epoch 91/100
4/4 [==============================] - 16s 4s/step - loss: 57539316.0000 - val_loss: 59893196.0000
Epoch 92/100
4/4 [==============================] - 16s 3s/step - loss: 57415620.0000 - val_loss: 59051844.0000
Epoch 93/100
4/4 [==============================] - 19s 4s/step - loss: 56753124.0000 - val_loss: 58819380.0000
Epoch 94/100
4/4 [==============================] - 20s 6s/step - loss: 56351400.0000 - val_loss: 58736148.0000
Epoch 95/100
4/4 [==============================] - 18s 5s/step - loss: 56126520.0000 - val_loss: 58622796.0000
Epoch 96/100
4/4 [==============================] - 27s 8s/step - loss: 55862592.0000 - val_loss: 58281772.0000
Epoch 97/100
4/4 [==============================] - 19s 5s/step - loss: 57006308.0000 - val_loss: 58012812.0000
Epoch 98/100
4/4 [==============================] - 18s 4s/step - loss: 55288328.0000 - val_loss: 58425204.0000
Epoch 99/100
4/4 [==============================] - 15s 3s/step - loss: 55472456.0000 - val_loss: 58068788.0000
Epoch 100/100
4/4 [==============================] - 19s 4s/step - loss: 55637824.0000 - val_loss: 57740716.0000
```

Figura 25 Proceso de entrenamiento del conjunto de datos VC1

Código 11 Proceso de entrenamiento del conjunto de datos BREF

```
checkpoint_filepath = '/content/df_bref3_json_separated.json'

model_checkpoint_callback = tf.keras.callbacks.ModelCheckpoint(
    filepath=checkpoint_filepath,
    save_weights_only=True,
    monitor='val_loss',
    mode='max',
    save_best_only=True)

early_stop_callback = tf.keras.callbacks.EarlyStopping(monitor='val_loss',
patience=10)

history1 = encoder1.fit(x1_train,
            x1_train,
            batch_size=128,
            epochs=100,
            validation_data=(x1_test, x1_test),
callbacks=[model_checkpoint_callback, early_stop_callback])
```

```
Epoch 90/100
5/5 [==============================] - 12s 3s/step - loss: 1905743104.0000 - val_loss: 164742758400.0000
Epoch 91/100
5/5 [==============================] - 10s 2s/step - loss: 1900261632.0000 - val_loss: 163991126016.0000
Epoch 92/100
5/5 [==============================] - 11s 2s/step - loss: 1914565632.0000 - val_loss: 163253731328.0000
Epoch 93/100
5/5 [==============================] - 11s 2s/step - loss: 1903734784.0000 - val_loss: 162597421056.0000
Epoch 94/100
5/5 [==============================] - 12s 3s/step - loss: 1877231616.0000 - val_loss: 161861107712.0000
Epoch 95/100
5/5 [==============================] - 11s 2s/step - loss: 1860662528.0000 - val_loss: 161115111424.0000
Epoch 96/100
5/5 [==============================] - 12s 3s/step - loss: 1842819712.0000 - val_loss: 160444743680.0000
Epoch 97/100
5/5 [==============================] - 10s 2s/step - loss: 1838999168.0000 - val_loss: 159786598400.0000
Epoch 98/100
5/5 [==============================] - 13s 3s/step - loss: 1832212096.0000 - val_loss: 159200460800.0000
Epoch 99/100
5/5 [==============================] - 9s 2s/step - loss: 1819522944.0000 - val_loss: 158619959296.0000
Epoch 100/100
5/5 [==============================] - 14s 3s/step - loss: 1813937792.0000 - val_loss: 158013751296.0000
```

Figura 26 Proceso de entrenamiento del conjunto de datos BREF

3.5 EVALUACIÓN DEL MODELO NICE

La Figura 24 indica las pérdidas de entrenamiento y validación que se obtuvieron al alimentar el modelo NICE con el conjunto de datos correspondiente a la estación VC1.

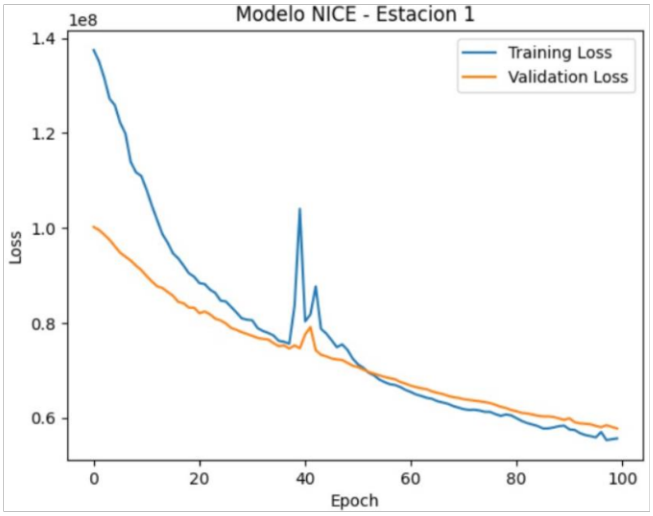


Figura 27 Modelo NICE de la estación VC1

Se ha realizado la evaluación del modelo NICE que ha sido alimentado por el conjunto de datos correspondiente a la estación VC1. Para la presente evaluación se ha tomado en cuenta las siguientes métricas: pérdida de entrenamiento y pérdida de validación. En la Figura 24 se ha podido observar la curva de pérdida de entrenamiento (color azul) y la curva de pérdida de

validación (color naranja). En el presente escenario se ha podido observar que la pérdida de entrenamiento y la pérdida de validación disminuyen y se han logrado estabilizar. En otras palabras, el modelo NICE para la estación VC1 indica un ajuste óptimo [22].

La Figura 25 indica las pérdidas de entrenamiento y validación que se obtuvieron al alimentar el modelo NICE con el conjunto de datos correspondiente a la estación BREF.

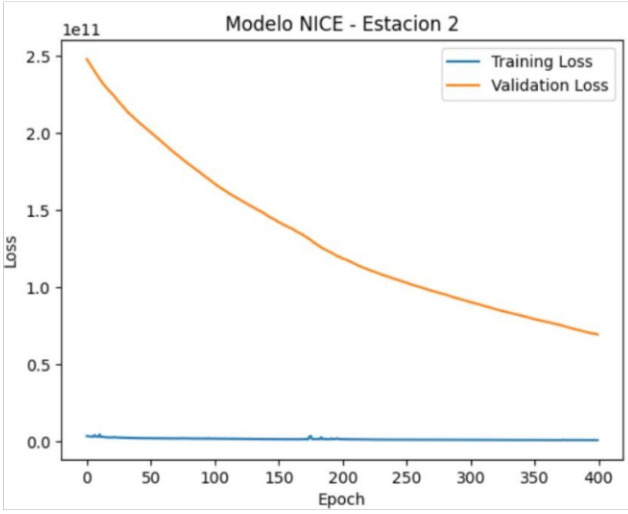


Figura 28 Modelo NICE de la estación BREF

Se ha realizado la evaluación del modelo NICE que ha sido alimentado por el conjunto de datos correspondiente a la estación BREF. Para la presente evaluación se ha tomado en cuenta las siguientes métricas: pérdida de entrenamiento y pérdida de validación. En la Figura 25 se ha podido observar la curva de pérdida de entrenamiento (color azul) y la curva de pérdida de validación (color naranja). En el presente escenario se ha podido observar que la pérdida de entrenamiento y la pérdida de validación no logran estabilizarse. En esta ocasión la pérdida de validación ha sido mayor que la pérdida de entrenamiento. No se ha logrado modelar con precisión los datos de entrenamiento por lo cual se han generado errores considerables. Este error puede indicar que se ha necesitado más entrenamiento para reducir las pérdidas. De manera alternativa para mejorar el modelo se pudo haber aumentado los datos de entrenamiento. En conclusión, no existe un buen ajuste en el modelo NICE para la estación BREF.

3.6 GENERACIÓN DE DATOS SINTÉTICOS

Los resultados que se han obtenido al seguir el procedimiento de la figura son:

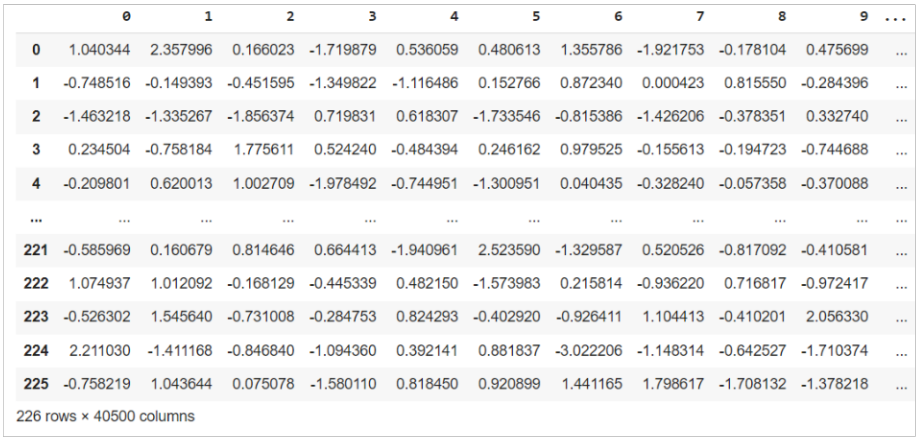
El “decoder0” ha sido el decodificador correspondiente al modelo “encoder0”. Con la finalidad de reconstruir la entrada original a partir de su representación codificada se ha utilizado el modelo “encoder0”. Esto ha sido esencial para la reconstrucción de datos mediante la utilización del autoencoder completo. De la misma manera, se ha definido el modelo “decoder1” que corresponde al conjunto de datos BREF. De manera similar, el “decoder1” ha sido el decodificador correspondiente al modelo “encoder1”. En conclusión, se necesitan de ambos modelos, “decoder” y “encoder” para empezar el proceso de generación sintética de muestras.

Finalmente, se ha podido generar muestras mediante el modelo “decoder0” y “decoder1”. La generación sintética de muestras se la ha logrado calculando el 40% del tamaño del conjunto de entrenamiento. El tamaño de salida se ha definido como el tamaño del número de columnas del conjunto de datos original. El modelo “decoder0” y “decoder1” se ha empleado para predecir la salida correspondiente al vector de muestra y se ha almacenado en la matriz “samples_x1”. Como último paso, las muestras generadas se han guardado en un archivo empleando la función “np.savetxt()”.

	0	1	2	3	4	5	6	7	8	9	...
0	-0.248094	-0.112463	1.129081	1.261176	-1.832870	-0.343904	-0.019166	-1.303408	-1.476400	-0.325139	...
1	0.920087	0.008031	1.794951	-1.270797	1.771874	-1.089795	-0.558050	-0.801293	0.702676	0.550753	...
2	0.213234	0.516995	-0.939163	0.290499	-1.387292	-1.302539	1.162453	-1.472552	1.716716	1.593246	...
3	-0.246854	-1.339018	0.042364	0.257353	-1.844632	0.140265	-0.119096	0.830752	2.303159	0.424121	...
4	-0.417618	-0.247514	0.467748	-0.454534	-0.062218	-1.322764	-0.328894	-0.364862	0.739355	0.923643	...
...	...	...	...	...	...	...	...	...	...	...	...
149	-0.815044	0.314395	-0.859128	0.394703	0.297817	-0.146131	-0.942831	0.796235	-0.009471	0.260958	...
150	1.616307	0.448237	1.680036	-0.639751	-2.205627	1.328933	1.556335	-0.122862	-0.387414	-1.110280	...
151	-0.934950	0.314206	-0.018194	-0.044323	-0.738323	2.137609	-1.001495	1.322368	-0.278105	2.522479	...
152	-0.698318	-0.995886	0.135842	0.023902	0.747603	-1.574994	-1.469681	-0.312556	-0.614367	0.410071	...
153	0.015397	-0.526072	0.181149	0.707287	0.492135	-0.631242	1.314426	-0.271827	-1.481225	-1.746443	...

154 rows × 40500 columns

Figura 29 Aumento de datos sintéticos de la estación VC1



	0	1	2	3	4	5	6	7	8	9	...
0	1.040344	2.357996	0.166023	-1.719879	0.536059	0.480613	1.355786	-1.921753	-0.178104	0.475699	...
1	-0.748516	-0.149393	-0.451595	-1.349822	-1.116486	0.152766	0.872340	0.000423	0.815550	-0.284396	...
2	-1.463218	-1.335267	-1.856374	0.719831	0.618307	-1.733546	-0.815386	-1.426206	-0.378351	0.332740	...
3	0.234504	-0.758184	1.775611	0.524240	-0.484394	0.246162	0.979525	-0.155613	-0.194723	-0.744688	...
4	-0.209801	0.620013	1.002709	-1.978492	-0.744951	-1.300951	0.040435	-0.328240	-0.057358	-0.370088	...
...	...	...	...	...	...	...	...	...	...	...	...
221	-0.585969	0.160679	0.814646	0.664413	-1.940961	2.523590	-1.329587	0.520526	-0.817092	-0.410581	...
222	1.074937	1.012092	-0.168129	-0.445339	0.482150	-1.573983	0.215814	-0.936220	0.716817	-0.972417	...
223	-0.526302	1.545640	-0.731008	-0.284753	0.824293	-0.402920	-0.926411	1.104413	-0.410201	2.056330	...
224	2.211030	-1.411168	-0.846840	-1.094360	0.392141	0.881837	-3.022206	-1.148314	-0.642527	-1.710374	...
225	-0.758219	1.043644	0.075078	-1.580110	0.818450	0.920899	1.441165	1.798617	-1.708132	-1.378218	...

226 rows × 40500 columns

Figura 30 Aumento de datos sintéticos de la estación BREF

Tabla 1 Tabla comparativa de aumento de datos y memoria utilizada de la estación VC1 y BREF

	Conjunto de datos de la estación 1 (VC1)	Conjunto de datos de la estación 2 (BREF)
Número de filas aumentadas	154	226
Número de columnas	40500	40500
Memoria utilizada para el aumento de datos	47.6 MB	69.8 MB

Se ha realizado un aumento de datos del 40% de la muestra original de cada conjunto de datos. Se puede observar que los dos conjuntos de datos aumentados no presentan el mismo número de datos aumentados. Esto se debe a que los conjuntos de datos correspondientes a VC1 y BREF son diferentes. Esto ha causado que los recursos computacionales requeridos para el aumento de datos tanto para el conjunto de datos VC1 y BREF sean diferentes. De esta manera se puede observar que se requiere más memoria para el aumento de datos de la estación 2 (

4 CONCLUSIONES Y RECOMENDACIONES

4.1 CONCLUSIONES

A pesar de la complejidad existente en el conjunto de datos referente a los eventos sísmicos del volcán Cotopaxi, se ha logrado la modificación del formato de la base de datos obtenida mediante un script realizado en Matlab. Este proceso ha permitido una comprensión clara acerca de la información presente en la base de datos. Se ha conseguido tener una visión más clara acerca del tipo de datos que maneja la base de datos de eventos sísmicos del volcán Cotopaxi. Además, como resultado se ha obtenido la base de datos en un formato compatible para el entorno Google Colab.

La finalidad de la adecuación de la base de datos de eventos sísmicos del volcán Cotopaxi no solamente ha sido la modificación del formato del conjunto de datos para que sea compatible con el entorno de trabajo de Google Colab, sino también la reestructuración y ordenamiento de la información. Debido a este procedimiento, se ha facilitado el procesamiento de los datos para un posterior análisis. Como resultado se ha obtenido una base de datos organizada.

A pesar de que existen varias técnicas que han sido utilizadas para el aumento de datos como, por ejemplo: modelo generativo antagónico (GAN), autoencoders variacional (VAE) y flujo de normalización, los flujos de normalización han representado una mejor opción para el aumento de datos. Debido a que el flujo de normalización ha superado la baja calidad de VAE y el colapso de modo de GAN, es una técnica preferida para emplearla en el aumento de datos.

Se ha implementado un modelo basado en flujos de normalización denominado modelo NICE para el aumento de datos. Se ha optado por un modelo que utiliza flujos de normalización debido a que el entrenamiento es más estable a comparación del modelo generativo antagónico y convergen con mayor facilidad a comparación del autoencoder variacional.

Al realizar el aumento de datos de las señales sísmicas del volcán Cotopaxi se pudo constatar que el modelo NICE tuvo un buen desempeño en el conjunto de datos correspondiente a la estación VC1. Sin embargo, el desempeño correspondiente al conjunto de datos BREF no ha sido óptimo, debido a que a diferencia del conjunto de datos VC1, las métricas de pérdida de entrenamiento y pérdida de validación no se han estabilizado.

4.2 RECOMENDACIONES

Es crucial entender el significado de la información de cualquier conjunto de datos, con la finalidad de tener un mejor nivel de interpretación de los datos. Se sugiere la implementación de una tabla que describa cada columna del conjunto de datos, debido a que la comprensión de los datos es importante para el procesamiento de los datos.

Con el fin de organizar y ordenar la base de datos obtenida para el presente proyecto, que se ha realizado en Google Colab, se recomienda hacer uso de las bibliotecas “numpy” y “pandas”. Debido a que numpy y pandas facilitan la manipulación de los datos, se han empleado para cargar, limpiar, filtrar y transformar conjuntos de datos. Además, ha sido posible la visualización de datos mediante DataFrames gracias a la biblioteca pandas. El proceso de preparación de los datos para su posterior análisis y modelado se han facilitado gracias al aprovechamiento de estas bibliotecas.

Tener en cuenta que al conocer las limitaciones de los modelos generativos antagónicos y los autoencoders variacionales frente a los flujos de normalización, se recomienda el uso de los flujos de normalización para contrarrestar los problemas de mala calidad en los resultados y el colapso de modo.

Es importante tener en cuenta que los flujos de normalización se basan en funciones reversibles. Se ha tomado ventaja de las funciones reversibles como base de la implementación del modelo NICE. De esta manera, se sugiere la utilización del modelo NICE para el aumento de datos debido a las ventajas de las funciones reversibles. Una de las ventajas de las funciones reversibles ha sido la capacidad de inversión.

Para abordar con el problema de desempeño del conjunto de datos correspondiente a la estación BREF se ha sugerido aumentar el número de épocas en el proceso de entrenamiento del conjunto de datos, tratando de evitar el sobre ajuste. Además, es recomendable la adquisición de más datos sísmicos para el conjunto BREF. Estas dos acciones podrían mejorar el desempeño de la implementación del modelo NICE en el conjunto de datos BREF.

5 REFERENCIAS BIBLIOGRÁFICAS

[1] N. Pérez, D. Benítez, F. Grijalva, R. Lara-Cueva, M. Ruiz, y J. Aguilar, «ESeismic: Towards an Ecuadorian volcano seismic repository», *Journal of Volcanology and Geothermal Research*, vol. 396, p. 106855, may 2020, doi: 10.1016/j.jvolgeores.2020.106855

[2] M. Malfante, M. Dalla Mura, J. Mars, J.-P. Metaxian, O. Macedo, y L. Inza, «Automatic Classification of Volcano Seismic Signatures», *Journal of Geophysical Research: Solid Earth*, vol. 123, sep. 2018, doi: 10.1029/2018JB015470

[3] R. A. Lara-Cueva, D. S. Benítez, E. V. Carrera, M. Ruiz, y J. L. Rojo-Álvarez, «Automatic Recognition of Long Period Events From Volcano Tectonic Earthquakes at Cotopaxi Volcano», *IEEE Transactions on Geoscience and Remote Sensing*, vol. 54, n.º 9, pp. 5247-5257, sep. 2016, doi: 10.1109/TGRS.2016.2559440

[4] A. Duque *et al.*, «Exploring the unsupervised classification of seismic events of Cotopaxi volcano», *Journal of Volcanology and Geothermal Research*, vol. 403, n.º 107009, oct. 2020, doi: 10.1016/j.jvolgeores.2020.107009. Disponible en: http://www.scopus.com/inward/record.url?scp=85089550238&partnerID=8YFLogxK. [Accedido: 19 de febrero de 2024]

[5] Viracucha E., de la Bastida J., «Sistema Informático para el Procesamiento y Análisis de Señales Sísmicas de Volcanes en el Ecuador.», *REVISTA EPN*, vol. VOL. 33, p. 7, ENERO 2014.

[6] «MAT File Extension - What is .mat and how to open? - ReviverSoft». Disponible en: https://www.reviversoft.com/en/file-extensions/mat. [Accedido: 21 de enero de 2024]

[7] F. G. de Zúñiga, «Archivo JSON: qué es y para qué sirve», *Blog de arsys.es*, 2 de agosto de 2023. Disponible en: https://www.arsys.es/blog/archivo-json-que-es-y-para-que-sirve. [Accedido: 22 de enero de 2024]

[8] «What is MATLAB?», *Simplilearn.com*, 23 de septiembre de 2022. Disponible en: https://www.simplilearn.com/tutorials/matlab-tutorial/what-is-matlab-introduction-for-beginners. [Accedido: 22 de enero de 2024]

[9] «What is Python? Executive Summary», *Python.org*. Disponible en: https://www.python.org/doc/essays/blurb/. [Accedido: 22 de enero de 2024]

[10] «Introduction to Python». Disponible en: https://www.w3schools.com/python/python_intro.asp. [Accedido: 22 de enero de 2024]

[11] «¿Qué es Python? - Explicación del lenguaje Python - AWS», *Amazon Web Services, Inc.* Disponible en: https://aws.amazon.com/es/what-is/python/. [Accedido: 22 de enero de 2024]

[12] «Google Colab». Disponible en: https://research.google.com/colaboratory/faq.html?authuser=3&hl=es. [Accedido: 23 de enero de 2024]

[13] A. S. Alberca, «La librería Numpy», *Aprende con Alf*. Disponible en: https://aprendeconalf.es/docencia/python/manual/numpy/. [Accedido: 23 de enero de 2024]

[14] «Pandas : La biblioteca de Python dedicada a la Data Science», *Formación en ciencia de datos | DataScientest.com*, 19 de diciembre de 2022. Disponible en: https://datascientest.com/es/pandas-python. [Accedido: 24 de enero de 2024]

[15] «Python JSON». Disponible en: https://www.w3schools.com/python/python_json.asp. [Accedido: 24 de enero de 2024]

[16] «What is machine learning? Understanding types & applications». Disponible en: https://www.spiceworks.com/tech/artificial-intelligence/articles/what-is-ml/. [Accedido: 25 de enero de 2024]

[17] «¿Qué es el aprendizaje automático? | Glosario». Disponible en: https://www.hpe.com/lamerica/es/what-is/machine-learning.html. [Accedido: 26 de enero de 2024]

[18] K. E. Koech, «The Basics of Neural Networks (Neural Network Series) — Part 1», *Medium*, 14 de mayo de 2022. Disponible en: https://towardsdatascience.com/the-

basics-of-neural-networks-neural-network-series-part-1-4419e343b2b. [Accedido: 21 de febrero de 2024]
[19] A. Omray, «Introduction to Normalizing Flows», *Medium*, 16 de julio de 2021. Disponible en: https://towardsdatascience.com/introduction-to-normalizing-flows-d002af262a4b. [Accedido: 12 de febrero de 2024]
[20] «Bijective Functions: Definition, Examples & Differences», *StudySmarter UK*. Disponible en: https://www.studysmarter.co.uk/explanations/math/pure-maths/bijective-functions/. [Accedido: 21 de febrero de 2024]
[21] L. Ge, W. Liao, S. Wang, B. Bak-Jensen, y J. R. Pillai, «Modeling Daily Load Profiles of Distribution Network for Scenario Generation Using Flow-Based Generative Network», *IEEE Access*, vol. 8, pp. 77587-77597, 2020, doi: 10.1109/ACCESS.2020.2989350
[22] H. Ullon, «hernanullon/SynteticLoadCurves». 28 de noviembre de 2022. Disponible en: https://github.com/hernanullon/SynteticLoadCurves. [Accedido: 7 de febrero de 2024]
[23] baeldung, «Training and Validation Loss in Deep Learning | Baeldung on Computer Science», 19 de febrero de 2022. Disponible en: https://www.baeldung.com/cs/training-validation-loss-deep-learning. [Accedido: 19 de febrero de 2024]
[24] «Descarga de Conjunto de Datos ESeismic - Instituto Geofísico - EPN». Disponible en: https://www.igepn.edu.ec/senales-sismicas/fomulario-eseismic. [Accedido: 16 de febrero de 2024]
[25] «Everything You Wanted To Know About TensorFlow», *Databricks*, 10 de julio de 2018. Disponible en: https://www.databricks.com/glossary/tensorflow-guide. [Accedido: 21 de febrero de 2024]