UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA

IMPLEMENTACIÓN DE SELECTIVIDAD DE SENSORES EN PROGRAMA CAPAZ DE PREDECIR EPICENTRO DE EVENTOS VOLCÁNICOS DEL VOLCÁN CHILLÁN

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL ELÉCTRICO

NICOLÁS IGNACIO DE CELIS CARMINE

PROFESOR GUÍA:
NÉSTOR BECERRA YOMA

MIEMBROS DE LA COMISIÓN:
ANDRÉS CABA RUTTE
JORGE WUTH SEPÚLVEDA

SANTIAGO DE CHILE
2022

RESUMEN DE LA MEMORIA PARA OPTAR
AL TÍTULO DE INGENIERO CIVIL ELÉCTRICO
POR: NICOLÁS DE CELIS CARMINE
FECHA: 2022
PROF. GUÍA: NÉSTOR BECERRA YOMA

IMPLEMENTACIÓN DE SELECTIVIDAD DE SENSORES EN PROGRAMA CAPAZ DE PREDECIR EPICENTRO DE EVENTOS VOLCÁNICOS DEL VOLCÁN CHILLÁN

La localización de eventos sísmicos de origen volcánico han sido una tarea desafiante en la literatura, ya que presenta una dificultad mayor que la de sismos de origen tectónico. Los eventos sísmicos volcánicos son útiles para determinar las orientaciones de estrés de los volcanes, así como su estructura interna y predicción de su comportamiento. Los métodos de localización semi-automáticos utilizados hoy en día presentan múltiples falencias, es por esto que este proceso aún es altamente dependiente del monitoreo de expertos.

Este trabajo consiste en mejorar el desempeño de un programa creado previamente por el autor, el cual corresponde de un modelo de *Deep Learning* entrenado con el objetivo de localizar automáticamente el epicentro de eventos sísmicos volcánicos del Volcán Chillán. Esto se logra utilizando la red de estaciones de monitoreo del Observatorio Volcanológico de los Andes Sur (OVDAS).

El modelo logra localizar los epicentros sísmicos utilizando la técnica de *End-to-End*, lo que le permite pasar por alto la etapa de detección de las ondas internas (P y S) del evento sísmico. Esto se realizó investigando las capacidades de las *Long-Short Term Memory*, que corresponde a un tipo especial de redes recurrente especializada en el aprendizaje de secuencias largas.

Finalmente se presentan y analizan los resultados obtenidos, efectuando una comparación entre las distintas arquitecturas de Deep Learning aplicadas, así como el efecto de la implementación una variable de incertidumbre. Además, se comparan también los resultados con el modelo creado previamente y los métodos automáticos que se utilizan hoy en día. El principal aporte del trabajo es la superación del modelo anterior (52,15 % vs. 48,5 % de acierto) utilizando la base de datos completa proporcionada, sin desechar los eventos problemáticos.

Dedicado a mi madre Carmen Gloria: gracias a ella soy lo que soy.
Y a mi padre Hernán: por su apoyo incondicional en todas mis decisiones.

Tabla de Contenido

Índice de Tablas

Índice de Ilustraciones

Capítulo 1

Introducción

Chile es país que posee la segunda cadena volcánica más grande y de mayor actividad en el mundo después de Indonesia. En Chile hay más de dos mil volcanes, de los cuales 500 de ellos se consideran geológicamente activos y 60 de ellos poseen registro eruptivo [23].

El estudio de la sismología volcánica tiene como principales propósitos conocer los patrones de actividad sísmica que permiten establecer adecuadamente la probabilidad de una erupción. Para los sismos volcánicos, los mecanismos de la fuente presentan mayores complejidades debido a que implican la dinámica adicional de gases, fluidos y sólidos en la generación de vibraciones, sumada a la gran complejidad de las estructuras volcánicas.

Esta realidad llevó al Servicio Nacional de Geología y Minería (Sernageomin), al Gobierno Regional de La Araucanía, Conaf y la Dirección Regional de Arquitectura a crear el Observatorio Volcanológico de los Andes Sur (OVDAS) [21], para así monitorear esta red de volcanes. El volcán Chillán es el cuarto volcán más activo de Chile, donde los principales peligros asociados a este son los lahares, flujos de detritos y coladas de lava, siendo la generación de lahares el mayor de todos para la población aledaña al volcán. Además, en corto plazo podrían ocurrir explosiones capaces de generar flujos calientes de ceniza. No obstante, este tipo de actividad sumada a explosiones como las ocurridas entre 2003 y 2004 constituyen un riesgo para la gran cantidad de turistas que visita el volcán [39].

La localización de los eventos sísmicos, tanto tectónicos como volcánicos, es una actividad que realizada manualmente logra la mayor precisión pero requiere mucho tiempo, ya que se necesita un modelo de velocidad de terreno y una clara detección de las ondas P y S. Hoy en día, la mayor parte de los instrumentos de las estaciones sismológicas digitalizan y almacenan datos y junto a que el monitoreo sísmico se ha crecido fuertemente en todo el planeta, genera una necesidad de analizar una cantidad enorme de datos. Por otro lado, los métodos semiautomáticos existentes hoy en día poseen evidentes falencias, por lo que los resultados siguen siendo monitoreados por expertos [51], por esta razón es necesaria una solución eficiente y confiable para que el tiempo empleado en localizar los sismos sea utilizado para analizar los posibles peligros que estos conllevan en tiempo real.

Para afrontar este problema, se ha desarrollado un sistema basado en Deep Learning, mediante la metodología *end-to-end*, es decir, el programa realiza todo el procesamiento y

análisis de información desde la detección de las ondas P y S, hasta la entrega de una coordenada correspondiente al epicentro del sismo. Esto ocurre sin la necesidad de proporcionarle un modelo de velocidad de terreno ni la geografía de la zona, lo importante es disponer de datos de la mejor calidad posible. Este sistema de Deep Learning utiliza una arquitectura de red neuronal que utiliza celdas Long-Short Term Memory (LSTM). La información que utiliza la red corresponde a las señales entregadas por el sismógrafo de cada estación como entrada y la localización de estos realizada por un experto, como salida deseada.

En cuanto a la información disponible para el entrenamiento del sistema, el OVDAS junto con la Universidad de la Frontera (UFRO) han proporcionado tres bases de datos, correspondientes a los períodos de 2015 al 2017 la primera, 2019 la segunda y la tercera sismos registrados en el 2020. Las tres bases de datos suman un total de 1.884 eventos. Cada evento contiene la señal en el tiempo registrada por el sismógrafo de cada estación y el respectivo epicentro. Para ingresar a la red neuronal, las señales en el tiempo son preprocesadas, aplicando una ventana de Hamming con un cierto traslape, para luego calcular la Fast Fourier Transform (FFT) a estas ventanas de tiempo.

Lo resultados del sistema desarrollado previamente, si bien son superiores a los métodos semiautomáticos comúnmente utilizados por los geólogos, contiene el gran problema de la escasa cantidad de estaciones consideradas para la localización, ya que se utilizó la cantidad mínima de 3 sensores.

1.1. Objetivos

1.1.1. Objetivo General

En base al sistema de localización desarrollado mediante Deep Learning, el objetivo general del trabajo es mejorar el modelo de Deep Learning actual, es decir, lograr que el sistema de localización de sismos volcánicos de este trabajo supere el porcentaje de aciertos del creado previamente, el cuál logra localizar un 48,5 % de los eventos con menos de $1km$ de error.

1.1.2. Objetivos Específicos

El programa de este trabajo debe ser capaz de utilizar el total de la información disponible de forma beneficiosa para el aprendizaje del sistema. Esto se refiere a que el programa anterior considera sólo 3 de las 10 estaciones disponibles, por lo que se plantea utilizar todas las estaciones de monitoreo.

La creación e implementación de un parámetro de incertidumbre que sea capaz de asignar un sesgo a los eventos que influyan negativamente en el aprendizaje del modelo.

1.2. Estructura

La organización del informe presenta la siguiente estructura:

1. En el Capítulo 2 se detallan los conceptos teóricos del problema (secciones de la 2.1 hasta 2.6) y muestra el estado del arte con trabajos relacionados en la literatura (2.7).
2. En el Capítulo 3 se muestra la metodología e implementación del sistema propuesto, donde se describe la base de datos (3.1), cálculos y variables creadas (3.4), el preprocesamiento (3.5) y la arquitectura utilizada 3.6, entre otros.
3. El Capítulo 4 presenta los resultados del trabajo, realizando un análisis y comparación con el trabajo previo mencionado.
4. Las conclusiones del trabajo se exponen en el Capítulo 5, donde se mencionan las ventajas del método y se proponen trabajos futuros para mejorar el desempeño del modelo.

Capítulo 2

Marco Teórico y Estado del Arte

2.1. Movimiento sísmico

Un movimiento sísmico es un movimiento vibratorio producido por la pérdida de estabilidad de masas de corteza los cuales pueden agruparse, tomando en cuenta su origen, como tectónicos, volcánicos y de colapso. Estos últimos son producidos principalmente por el derrumbamiento de techos de cavernas o minas y sólo son percibidos en áreas reducidas. Los sismos llamados tectónicos son aquellos producidos por rupturas de grandes dimensiones en la zona de contacto entre placas tectónicas (sismos interplaca) o bien en zonas internas de éstas (sismos intraplaca) [55].

Estos movimientos se propagan de forma concéntrica y tridimensional-mente a partir del punto de ruptura o derrumbamiento, este lugar se denomina hipocentro. Cuando las ondas procedentes del hipocentro llegan a la superficie se convierten en bidimensionales y se propagan desde el punto llamado epicentro [46].

Desde el hipocentro se producen las ondas internas, las cuales se subdividen en dos tipos: las ondas Primaria (P) y Secundaria (S). Las ondas P son las primeras que se registran en los sismógrafos y son de tipo longitudinal, por lo que las partículas rocosas vibran en la dirección del avance de la onda. Las ondas S son más lentas y de tipo transversal, es decir, la vibración se produce de forma perpendicular al avance de la onda.

2.1.1. Localización de eventos sísmicos

Existen múltiples métodos para la localización de eventos sísmicos, entre los métodos de localización clásicos están están los métodos gráficos, iterativos, de búsqueda en rejilla y de correlación.

El método de localización por excelencia es el método gráfico de círculos o triangulación, este puede ser implementado si se conocen las distancias de epicentro hasta al menos tres estaciones. Las distancias hasta cada estación pueden ser determinadas, en principio, usando

la diferencia de tiempos de llegada de las ondas P y S junto con un modelo de propagación [58]. Teniendo el modelo de propagación fijo en función de la zona geográfica, la correcta obtención de los inicios de las ondas P y S es la tarea que deben realizar los expertos a la hora de detectar nuevos eventos. El método más preciso es la inspección visual de los registros por parte de expertos, quienes aprovechan la información global de cada ola, las pistas locales para la detección de fases y la comparación de registros de otras estaciones, actividad que requiere mucho tiempo.

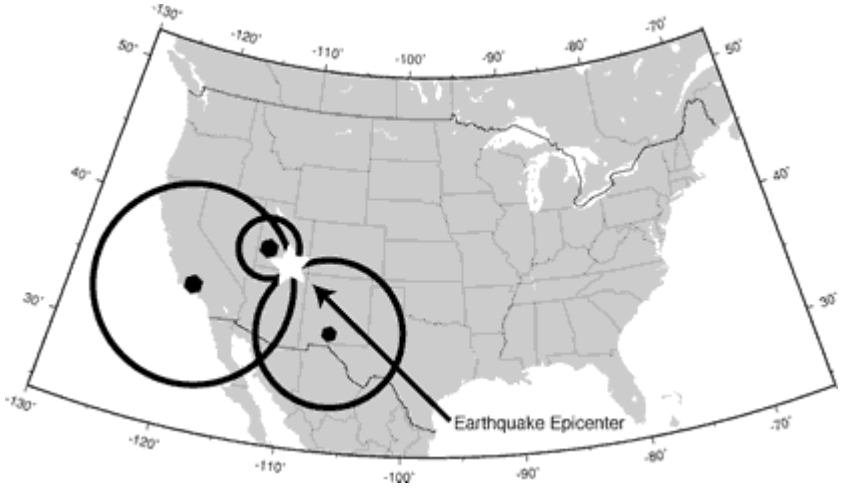

Figura 2.1: Se ejemplifica un caso de localización de un evento sísmico mediante triangulación.

2.1.2. Movimientos sísmicos de origen volcánico

Los eventos volcano-tectónicos son los generados principalmente por tensiones que surgen durante la migración de magma en la corteza terrestre. Estos terremotos pueden ser causador por cizallamiento o fractura por tracción durante el movimiento del magma desde la superficie profunda de la Tierra a través de conductos y diques. Los terremotos de tipo VT ocurren dentro y alrededor del edificio volcánico y reflejan la interacción de dos procesos geológicos generales: la migración de magma a la superficie de la tierra y la actividad tectónica de la corteza [10]. Se ha encontrado en estudios que los eventos de tipo VT es generalmente el precursor sísmico más temprano reportado de erupciones en volcanes, ya que la mayor parte de la información contenida en las formas de onda de estos eventos se puede extraer mediante un análisis simple de los datos registrados por las redes de monitoreo sísmico [61] [49].

En [36] se explican el estudio de los eventos volcano-tectónicos (VT), los cuales son los utilizados en este trabajo. Los eventos de alta frecuencia (también llamados eventos volcano-tectónicos) son útiles en los volcanes para determinar las orientaciones del estrés mediante el estudio de los mecanismos focales y la inversión del tensor de estrés.

Los problemas de la ubicación de los terremotos y de la determinación de la estructura de velocidades son interdependientes. Por lo tanto, se han realizado inversiones conjuntas para ubicaciones y estructura de velocidad. Está claro que los modelos precisos de velocidad y ubicación de los terremotos son fundamentales para la sismología y afectan la interpretación de toda la información derivada.

Las causas de los eventos VT, es decir, las fuentes de las tensiones, se han atribuido a las fuerzas tectónicas regionales, la carga gravitacional, los efectos de la presión de los poros y las fuerzas de hidrofracturamiento, térmicas y volumétricas asociadas con la intrusión, extracción, enfriamiento del magma o alguna combinación de cualquiera o todos estos.

2.1.3. Localización de eventos sísmicos volcánicos

Por lo general, la localización de los eventos sísmicos volcánicos genera un desafío mucho mayor que los de origen tectónico, esto debido a que cuando los terremotos ocurren en un volcán, la poca profundidad, la alta proximidad a la fuente, la menor magnitud y la compleja trayectoria de propagación, entre otros, dificulta en gran medida la detección de la onda interna S [16]. Por otro lado, según [17], "la estructura dentro de un volcán es, probablemente, el tema más complejo que los sismólogos hayan encontrado en la Tierra. Es extremadamente heterogéneo, anisotrópico y absorbente con interfaces y topografía irregulares que incluyen grietas de todos los tamaños y orientaciones. Los proceso de la fuente también son mucho más complicados que los terremotos tectónicos habituales, porque involucran una dinámica adicional del magma de gas y fluido en la generación de señales sísmicas".

2.2. Procesamiento de señales

Este trabajo se centra en un procesamiento frecuencial y temporal de señales, donde se consideran aspectos en el dominio del tiempo de la señales para aspectos de preprocesamiento y luego se analizan los datos en el dominio de la frecuencia. El dominio de la frecuencia se refiere al análisis matemático de funciones o señales con respecto a la frecuencia. Esto permite la representación del comportamiento cualitativo de un sistema, así como las características de la forma en que el sistema responde a cambios en el ancho de banda, ganancia, desplazamiento de fase, armónicos, etc [13]. Además, este dominio simplifica el estudio de las bandas dominantes de frecuencia lo que permite descartar o filtrar componentes de ruido.

2.2.1. Transformada discreta de Fourier (DFT)

La transformada discreta de Fourier es un algoritmo basado en la función de la transformada de Fourier que se aplica para secuencias de datos discretos. Las secuencias o señales se pueden descomponer como una suma de funciones sinusoidales, esto permite dividir la señal de entrada que se distribuye en el tiempo a un número de frecuencias de cierta longitud, amplitud y fases, donde todas esas frecuencias juntas forman la señal original, es decir, se cambia el dominio de la señal de temporal a frecuencial.

La DFT esta definida por:

$$X_k = \sum_{n=0}^{N-1} x_n \mathrm{e}^{-\frac{2\pi \mathrm{i}}{N} kn} \qquad k = 0, ..., N-1 \tag{2.1}$$

donde X_k es el k-ésimo coeficiente de la DFT y x_n denota la n-ésima muestra de la serie de tiempo que consiste de N muestras, además i corresponde a la unidad imaginaria ($i = \sqrt{-1}$) [11].

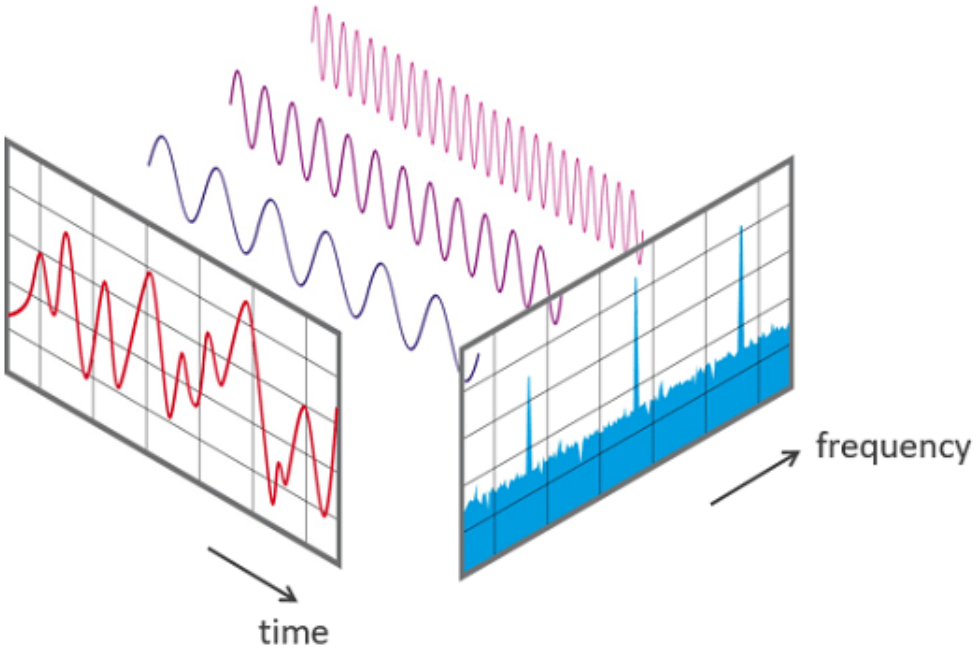

Figura 2.2: Cambio en el dominio de la señal de temporal a frecuencial [59].

2.2.2. Transformada rápida de Fourier (FFT)

La transformada rápida de Fourier (FFT por sus siglas en inglés) es el *algoritmo* para el cálculo de los coeficientes de Fourier. Este algoritmo ha producido importantes cambios en las técnicas computacionales utilizadas en el análisis espectral digital, simulación de filtros, entre otros. Este método calcula de manera eficiente la transformada discreta de Fourier (DFT) de una muestra de datos discretos, en otras palabras, la FFT es una implementación de la DFT que produce casi los mismos resultados pero calculada increíblemente más rápido y eficiente, lo que reduce la carga computacional significativamente. Además, este algoritmo reduce substancialmente los errores de redondeo asociados a estos cálculos. De hecho, tanto el tiempo de cálculo como los errores de redondeo se reducen en un factor de $(\log_2 N)/N$, con N el número de muestras de la serie de tiempo [11].

2.2.3. Transformada de Fourier en tiempo reducido (STFT)

La representación de la transformada de Fourier tiene varias limitaciones prácticas y conceptuales porque representa, para cada frecuencia ω, las características globales (en tiempo) de la señal. En consecuencia, la transformada de Fourier tiene la limitación práctica de que se debe conocer toda la señal para conocer su transformada. Además, la transformada de

Fourier no proporciona una representación adecuada para sistemas lineales variables en el tiempo [45].

Este problema afecta fuertemente para sistemas de reconocimiento automático de voz (ASR) o en el contexto de este trabajo de localización de una fuente emisora, ya que es necesario tener un análisis tanto temporal como frecuencial de las señales. Es por esto que la STFT se considera un método que descompone la señal no estacionaria en muchos segmentos pequeños, que se puede suponer que son localmente estacionarios, y aplica la FFT convencional a estos segmentos [52], proporcionando la representación temporal-frecuencial necesaria.

Para una señal discreta en el tiempo x_n, la STFT se define como:

$$X_w(nL, \omega) = \sum_{m=-\infty}^{\infty} x(m)w(nL - m)\mathrm{e}^{-j\omega m} \tag{2.2}$$

donde el subíndice w en $X_w(nL, \omega)$ denota el análisis de la ventana $w(n)$. El parámetro L es un entero que denota la separación de tiempo entre secciones adyacentes. Este parámetro es independiente del tiempo y es seleccionado tal que garantice un grado de superposición de tiempo entre secciones adyacentes. Para un valor fijo de n, $X_w(nL, \omega)$ representa la transformada de Fourier con respecto a la muestra m de la ventana $f_n(m) = x(m)w(nL - \omega)$. La interpretación de la ventana deslizante considera que $X_w(nL, \omega)$ se genera al desplazar la ventana de análisis a través de la señal. Después de cada desplazamiento de L muestras, la ventana se multiplica por la señal y la transformada de Fourier se aplica al producto [40].

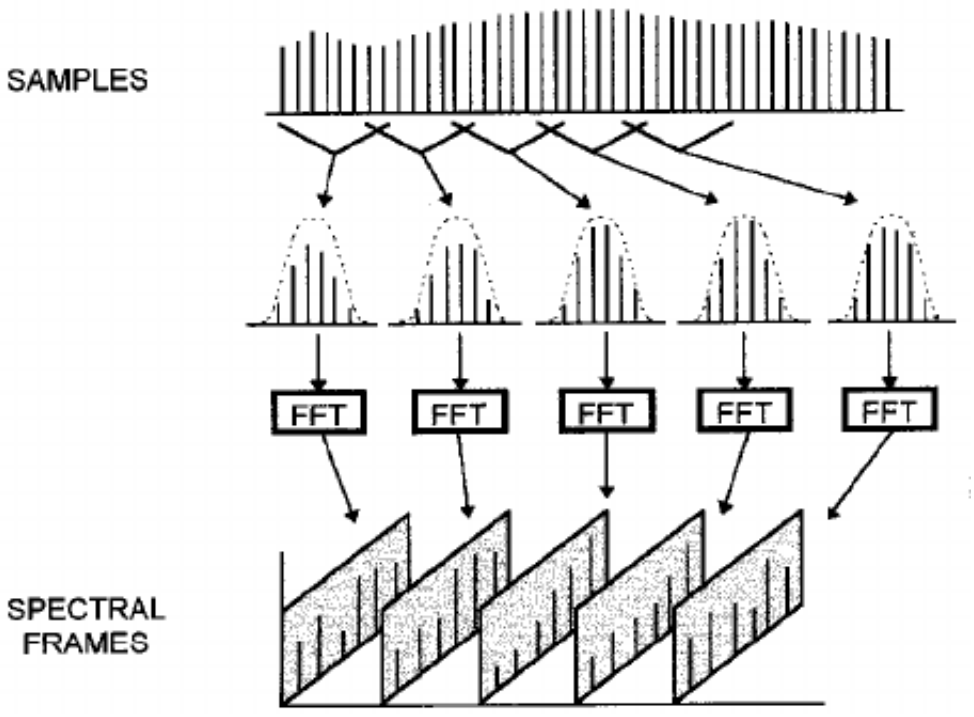

Figura 2.3: Aplicación de la STFT a una señal [15].

2.2.4. Ventana de Hamming

En el procesamiento de señales, una función de ventana es una función matemática cuyo valor es cero fuera de algún intervalo elegido. Estas representan una herramienta estadística que permite a los usuarios ver una pequeña región de una señal con la menor cantidad de fuga de cualquier otra parte de la señal. Las aplicaciones de las funciones de ventana incluyen análisis espectral, diseño de filtros de respuesta de impulso finito (FIR) y *beamforming*.

La ventana de Hamming se ha caracterizado por tener una buena resolución de frecuencia, una reducción de la fuga espectral y un rendimiento de ruido aceptable. Esta ventana se utiliza en cálculos para suavizar los datos antes de aplicar el análisis de Fourier [6].

La ventana de Hamming se define con la ecuación 2.3 y su función toma la forma representada en la Figura 2.4.

$$v(n) = a_0 - a_1 \cos \frac{2\pi n}{N-1} \tag{2.3}$$

donde $a_0 = 0{,}53836$ y $a_1 = 0{,}46164$.

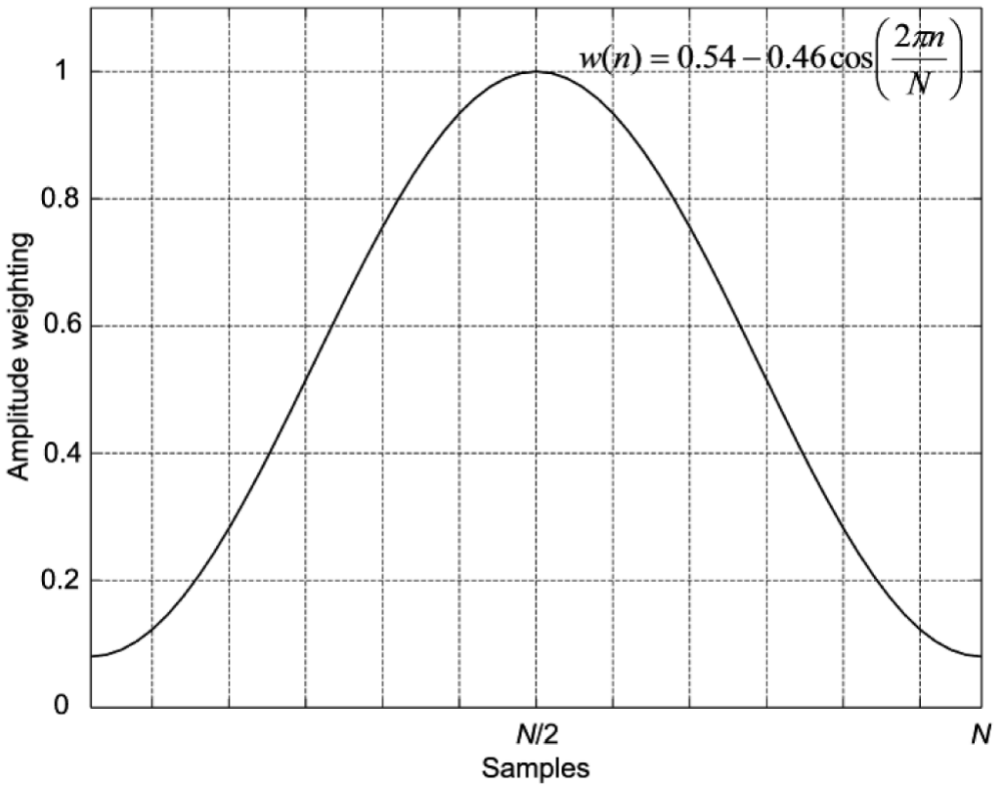

Figura 2.4: Función de la ventana de Hamming [6]

2.2.5. Signal-to-noise ratio (SNR)

La relación señal-ruido (SNR) se refiere a la relación entre la potencia de la señal y la potencia del ruido contenida en una grabación. Esta relación parametriza el rendimiento de los sistemas de procesamiento de señal óptimos cuando el ruido es Gaussiano [24].

Dada una señal $s(t)$ y un ruido $N(t)$, en primer lugar es necesario calcular la potencia promedio de la señal, definida en la ecuación 2.4.

$$P_s = \frac{1}{t_f - t_\mathrm{i}} \int_{t_\mathrm{i}}^{t_f} s^2(t)\mathrm{d}t \tag{2.4}$$

Donde t_i y t_f representan los puntos de inicio y final de la sección de interés de la señal.

Con la potencia de la señal y la potencia del ruido es posible calcular el SNR que se define de la siguiente forma:

$$SNR = \frac{P_s}{P_N} \tag{2.5}$$

Esta relación se suele expresar también en decibelios de la forma:

$$SNR(\mathrm{d}B) = 10\ log_{10} \frac{P_s}{P_N} \tag{2.6}$$

Los ingenieros usualmente consideran un SNR de 2 (3 dB) como el límite entre un SNR alto y bajo.

2.3. Inteligencia Artificial (IA)

La inteligencia artificial es la ciencia e ingeniería de la fabricación de máquinas inteligentes [35], e busca que las máquinas puedan imitar comportamientos inteligentes, los cuales pueden ser muy diversos, como conducir, reconocer voces, reconocer imágenes, ganar juegos, entre otras. Hoy en día tenemos cada vez más ejemplos de como ciertas áreas logran alcanzar un rendimiento incluso mayor al humano.

Hablando más certeramente, un resultado en inteligencia artificial consiste en el aislamiento de un problema particular de procesamiento de información, la formulación de una teoría computacional para él la construcción de un algoritmo que lo implemente y una demostración práctica de que el algoritmo es exitoso. Lo importante aquí, y es lo que hace posible el progreso, es que una vez que se ha establecido una teoría computacional para un problema en particular, nunca se tiene que volver a hacer y, en este sentido, un resultado en inteligencia artificial se comporta como un resultado en matemáticas o cualquiera de las ciencias naturales duras [34].

Por otro lado, la inteligencia artificial de la que se habla no es una inteligencia igual a la de los humanos, sino que una inteligencia localizada, capaz de resolver uno o un conjunto muy acotado de problemas, es por esto que nacen las siguientes clasificaciones de inteligencia artificial:

- La inteligencia artificial débil se refiere a aquellos sistemas computacionales que pueden cumplir con un conjunto muy limitado de tareas.
- La inteligencia artificial fuerte es cuando las IAs son capaces de aplicarse a una gran variedad de problemas y dominios diferentes. Al día de hoy, este grupo se mantiene como una aspiración, todas nuestras IAs se clasifican como débiles.

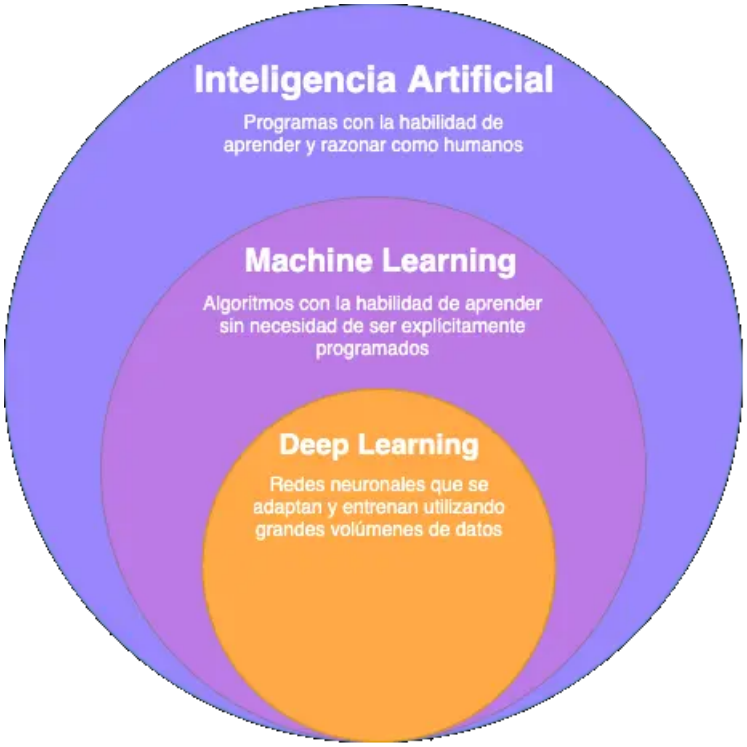

Figura 2.5: Conjuntos de Inteligencia Artificial, Machine Learning y Deep Learning [22].

2.3.1. Machine Learning

El machine learning es un subconjunto de la inteligencia artificial, el cual construye un modelo matemático basado en datos e muestra, conocidos como "datos de entrenamiento", con el fin de hacer predicciones o decisiones sin estar programado explícitamente para realizar la tarea. Además, es una disciplina enfocada en dos preguntas interrelacionadas: ¿Cómo se pueden construir sistemas informáticos que mejoren automáticamente a través de la experiencia? y ¿Cuáles son las leyes teórico-estadístico-información-computacional fundamentales que gobiernan todos los sistemas de aprendizaje, incluidos los ordenadores, seres humanos y las organizaciones?. El estudio del machine learning es importante tanto para abordar estas

cuestiones científicas y de ingeniería fundamentales y para el software informático altamente práctico que ha producido y distribuido en muchas aplicaciones [25].

Todas las capacidades pueden ser imitadas programando tareas específicas que los sistemas deban realizar, es por esto que se remarca la diferencia entre programar un sistema para que pueda realizar una tarea en específico a programarla para que aprenda a realizarla. Este cambio de paradigma es lo que hace interesante el *Machine Learning* y por ello es común confundir la inteligencia artificial con el Machine Learning, cuando en realidad el Machine Learning es sólo una parte de la inteligencia artificial [48].

TensorFlow

TensorFlow es una biblioteca de código abierto diseñada para el calculo numérico y machine learning a gran escala. Creada por el equipo de Google Brain, TensorFlow agrupa una gran cantidad de modelos y algoritmos de machine learning y deep learning, también conocidos como redes neuronales. Estas librerías utilizan Python para proporcionar una API (Application Programming Interface) de front-end (visible para el usuario permitiendo una interacción directa) para la construcción de aplicaciones mientras que ejecuta esas aplicaciones en C++ de alto rendimiento [1] [62].

Keras

Keras es una API de redes neuronales de alto nivel desarrollada con el enfoque de permitir una experimentación rápida. Keras proporciona una interfaz Python para redes neuronales artificiales, la cuál actúa como una interfaz para las librerías de TensorFlow. Esta API prioriza la experiencia del desarrollador, por lo que se dice que es una API diseñada para seres humanos, no máquinas. Para eso, ofrece APIs consistentes y simples. Por otro lado, Keras tiene un soporte integrado para redes convolucionales, redes recurrentes y cualquier combinación de ambas [9].

2.3.2. Deep Learning

El deep learning es un conjunto del machine learning que simula la estructura jerárquica del cerebro humano, procesando datos de un nivel inferior a un nivel superior y componiendo gradualmente conceptos cada vez más semánticos. En los últimos años, Google, Microsoft, IBM y Baidu han invertido una gran cantidad de recursos en la investigación y desarrollo del deep learning [64], esto ha logrado mejorar drásticamente el estado del arte en el reconocimiento de voz, reconocimiento de objetos visuales, detección de objetos y muchos otros dominios, como el descubrimiento de fármacos y genómica [29]. El aprendizaje profundo es quizás el progreso más exitoso realizado por la comunidad de aprendizaje automático en los últimos 10 años.

El deep learning o aprendizaje profundo, permite a los modelos computacionales aprender representaciones de datos con múltiples niveles de abstracción. Este enfoque es especialmente

adecuado para las tareas complejas en las que no todos los aspectos de los objetos pueden clasificarse de antemano. El propio sistema se encarga de buscar los diferenciadores adecuados; en cada capa, se analizan las nuevas entradas en busca de otras características, que el sistema utiliza para decidir como clasificar las entradas. Este proceso capacita a una máquina a enfrentarse a contextos más complejos, donde depende de la interconexión con otros sistemas y donde la volumetría de datos le obliga a tener en cuenta muchas variables de las que debe aprender.

2.4. Redes Neuronales

2.4.1. Red Neuronal Artificial

Son redes distribuidas y paralelas de procesadores elementales simples (neuronas), que generan modelos capaces de adquirir conocimiento del ambiente a través de un proceso de aprendizaje y almacenamiento de conocimiento adquirido en las conexiones sinápticas (pesos). Una neurona artificial es vista como una unidad procesadora de información con cuatro componentes básicos: entradas $x = [x_1, ..., x_m]$ pesos que ponderan las entradas $w = [w_{k1}, ..., w_{km}]$, sumador o combinador lineal con un sesgo b_k y una función de activación no-lineal $\varphi(\cdot)$ que limita la amplitud de la salida [60]. La ecuación de una neurona se formula como:

$$s_k = \sum_{j=0}^{m} w_{kj} x_j \tag{2.7}$$

$$v_k = s_k + b_k \tag{2.8}$$

$$y_k = \varphi(v_k) \tag{2.9}$$

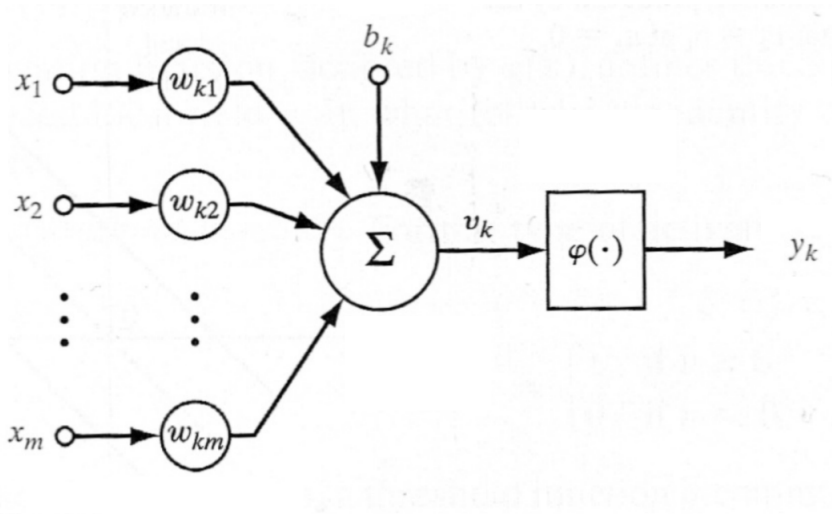

Figura 2.6: Representación gráfica del modelo de neurona artificial.

Ahora, es posible crear una red de neuronas con distintas arquitecturas y funciones, creando un sistema de procesadores elementales interconectados. Estas neuronas son generalmente

organizadas en capas o niveles, pudiendo agregar tantas capas como se desee y en donde las salidas de una capa están conectadas a las entradas de la siguiente [47], este tipo de arquitectura se le llama Multi Layer Perceptron (MLP).

El aprendizaje de la red neuronal se puede definir como el proceso en que se produce el ajuste de los parámetros libres (sesgo b y pesos w), esto en base a una determinada estimulación por el entorno que rodea a la red. Al construir la red, se comienza de un modelo de neurona y una determinada arquitectura de red, donde los pesos iniciales se establecen como nulos o aleatorios. Existen dos tipos principales de aprendizaje de la red: Supervisado y No Supervisado, en este trabajo se enfocará en los modelos de aprendizaje supervisado.

En el aprendizaje supervisado se presentan dos vectores, uno de entrada y otro de salida deseada. La salida computada por la red se compara con la salida deseada, y los pesos de la red se modifican en el sentido de reducir el error cometido. Este proceso se repite iterativamente hasta que la diferencia entre la salida computada y deseada sea aceptablemente pequeña [47], para así proveer un mapeo parametrizado no-lineal entre los vectores de entrada y salida deseada. Los polinomios proporcionan tales mapeos, y siempre que tengamos un número suficientemente grande de términos en el polinomio, es posible aproximar cualquier función razonable con precisión arbitraria [5].

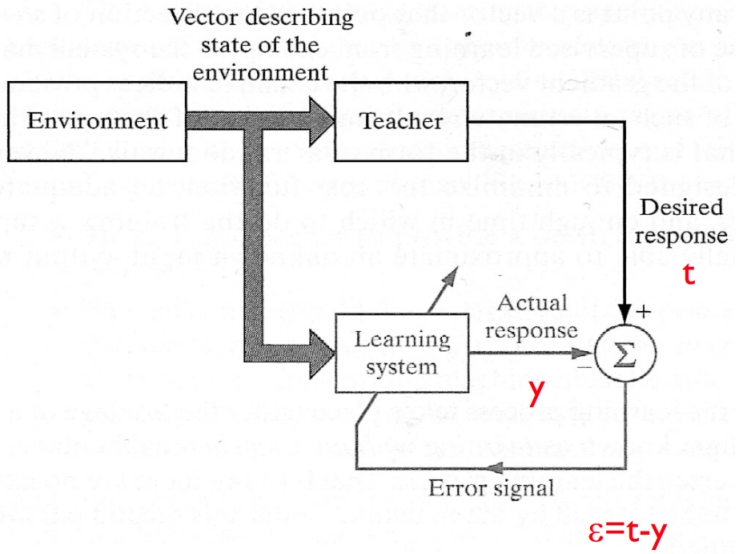

Figura 2.7: Diagrama de bloques del modelo de aprendizaje supervisado [60].

En el deep learning se tienen los conceptos de *loss* (pérdidas) y optimizadores, los cuales se definirán a continuación:

Función de pérdidas (*loss function*)

Las funciones de pérdidas nos dicen que tan mal se está desempeñando el sistema en un determinado momento. Este es un parámetro necesario para entrenar una red neuronal ya que cuantifica la diferencia entre la salida deseada y la salida generada por la red. De esta función podemos derivar los gradientes que se utilizan para actualizar los pesos [54]. Lo

que se busca realizar es asumir la pérdida y tratar de minimizarla, lo que lleva a un mejor desempeño del modelo.

Para el caso de este trabajo, se utilizan funciones de pérdidas usadas en configuraciones de regresión, donde los resultados esperados y pronosticados son valores de números reales. De este tipo de funciones de pérdidas se destacan dos principales: El error cuadrático medio (*mean squared error*, MSE) y el error absoluto medio (*mean absolute error*, MAE), las cuales se definen en las ecuaciones 2.10 y 2.11.

$$MSE = \frac{1}{n}\sum_{\mathrm{i}=1}^{n}(Y_{\mathrm{i}} - \hat{Y}_{\mathrm{i}})^2 \tag{2.10}$$

$$MAE = \frac{1}{n}\sum_{\mathrm{i}=1}^{n}|Y_{\mathrm{i}} - \hat{Y}_{\mathrm{i}}| \tag{2.11}$$

En cuanto al desempeño de ambas funciones, el MSE penaliza más los errores más grandes y es un estimador consistente de la varianza del error. Por otro lado, el MAE penaliza la distancia euclidiana en el error y no es tan exigente con los llamados *outliers* como lo es el MSE, por lo que es más apropiado si la varianza de las muestras es muy baja o no existe [8].

Función de optimización

Las funciones de optimización son algoritmos o métodos usados para cambiar los atributos de las redes neuronales, tal como los pesos y la tasa de aprendizaje con el objetivo de reducir las pérdidas [14]. Se entrará en mayor detalles con el optimizador *ADAM*, dado que es el utilizado en este trabajo.

El optimizador Adam (no es un acrónimo pero está derivado de *Adaptative Moment Estimation*) es un algoritmo de descenso de gradiente estocástico basado en la estimación de momentos de primer orden (la media del gradiente) y de segundo orden (el gradiente al cuadrado de los elementos) utilizando una media móvil exponencial y corrige su sesgo. La actualización de pesos final es proporcional a la tasa de aprendizaje multiplicada por el momento de primer orden dividido por la raíz cuadrada del momento de segundo orden [27]. Este optimizador tiene la ventaja de ser un método demasiado rápido que converge rápidamente, sin embargo, es un método costoso computacionalmente. En general, Adam es considerado uno de los mejores (si no el mejor) optimizadores para una basta cantidad de tareas, mostrando un desempeño superior a otros algoritmos como Gradiend Descent (GD), Stochastic Gradient Descent (SGD), Adaptative Gradient (AdaGrad) y AdaDelta.

2.4.2. Redes Neuronales Recurrentes (RNN)

Las redes neuronales recurrentes son redes neuronales que evolucionan durante una serie de iteraciones desde un estado inicial a un estado normalmente estacionario, momento en el que se consulta la salida de la red, la cual será usada para el preprocesamiento temporal

posterior; esto se consigue implementando algunas neuronas que reciben como entrada la salida de una de las capas y transmite su salida en una de las capas de un nivel posterior a ella [48].

Las redes *feed-forward* son denominadas generalmente como redes estáticas, ya que están organizadas en capas que se conectan de forma unidireccional y producen una única salida para un conjunto de entrada, es decir, el estado de una red es independiente del estado anterior. Por otro lado, las redes recurrentes son sistemas dinámicos, donde el cálculo de una entrada, en un determinado paso, depende del paso anterior y en algunos casos del paso futuro [12].

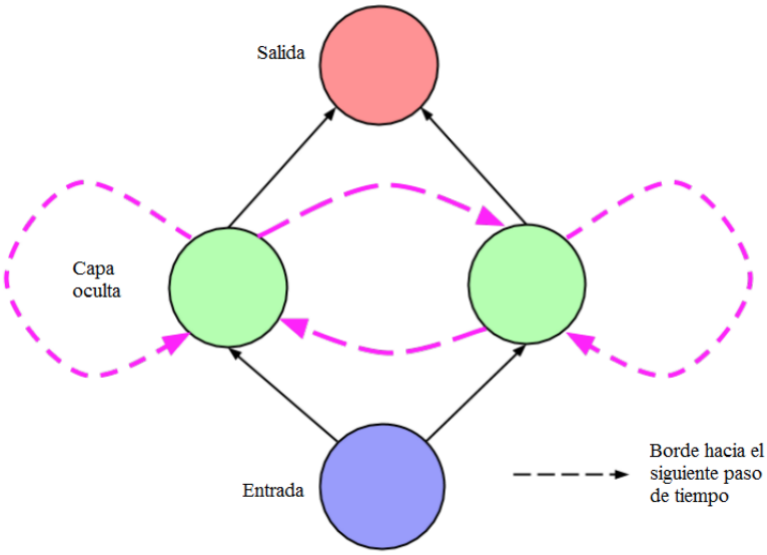

Figura 2.8: Red neuronal recurrente simple [20]

.

Las redes neuronales recurrentes (RNN) son capaces de traducir secuencias de entrada a secuencias de salida [29], esto hace de las RNN adecuadas para resolver problemas con información de tipo secuencias, como señales dinámicas complejas. Para secuencias muy largas, el gradiente puede desaparecer o explotar al calcular las derivadas del error, lo que impone restricciones a la hora de capturar información a largo plazo [53].

En cuanto al entrenamiento de una red recurrente, una red neuronal recurrente genérica, con input u_t y estado x_t para el tiempo t, está dada por:

$$x_t = W_{rec}\sigma(x_{t-1} + W_{\text{in}}u_t + b) \tag{2.12}$$

donde los parámetros del modelo están dados por la matriz de pesos recurrente W_{rec}, el bias b y la matriz de pesos de entrada W_{in}; x_0 es proporcionado por el usuario, puesto como cero o aprendido, y σ es una función punto a punto. Se tiene un costo $\mathcal{E} = \sum_{1\leq t\leq T}\mathcal{E}_t$ que mide el rendimiento de la red en alguna tarea determinada, donde $\mathcal{E}_t = \mathcal{L}(x_t)$.

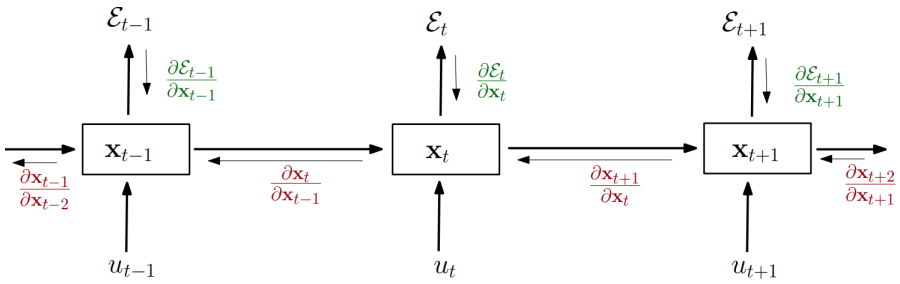

Figura 2.9: Proceso de una red recurrente. Se denota el estado oculto como x_t, la entrada de la red por u_t en el tiempo t y por $\mathcal{E}_t$ el error obtenido de la salida del tiempo t[43]

.

2.4.3. Desvanecimiento del gradiente

El problema de desvanecimiento del gradiente (en inglés *Vanishing Gradient Problem (VGP)*) en una RNN ocurre cuando se realiza *backpropagation* y métodos de aprendizaje basados en reducción de gradiente, el cual le impide a la red aprender correctamente dependencias con mas de 10 pasos de tiempo [57]. Este problema también tiene un caso inverso llamado explosión del gradiente, que ocurre cuando se utilizan funciones de activación cuyas derivadas pueden tomar valores más grandes.

En [43] se explica donde ocurre el desvanecimiento de gradiente y explica el entrenamiento de una red recurrente, donde se representa el entrenamiento de una RNN en la Figura 2.9 y se analiza el *backpropagation through time (BPTT)* de forma que se resalte mejor el problema del vanishing gradient:

$$\frac{\partial \mathcal{E}}{\partial \theta} = \sum_{1 \leq t \leq T} \frac{\partial \mathcal{E}_t}{\partial \theta} \tag{2.13}$$

$$\frac{\partial \mathcal{E}_t}{\partial \theta} = \sum_{1 \leq k \leq t} \left(\frac{\partial \mathcal{E}}{\partial x_t} \frac{\partial x_t}{\partial x_k} \frac{\partial^+ x_k}{\partial \theta}\right) \tag{2.14}$$

$$\frac{\partial x_t}{\partial x_k} = \prod_{t \geq \mathrm{i} \geq k} \frac{\partial x_\mathrm{i}}{\partial x_{\mathrm{i}-1}} = \prod_{t \geq \mathrm{i} \geq k} W_{rec}^T \mathrm{diag}(\sigma'(x_{\mathrm{i}-1})) \tag{2.15}$$

Estas ecuaciones fueron obtenidas escribiendo los gradientes de forma de una suma de productos. Cualquier componente de gradiente $\frac{\partial \mathcal{E}}{\partial \theta}$ es también una suma (ecuación 2.14), cuyos términos los llamamos contribuciones temporales o *componentes temporales*. Se puede ver que cada *componente temporal* $\frac{\partial \mathcal{E}}{\partial x_t} \frac{\partial x_t}{\partial x_k} \frac{\partial^+ x_k}{\partial \theta}$ mide como θ en el paso k afecta el costo en el paso $t > k$. Los factores $\frac{\partial x_t}{\partial x_k}$ (ecuación 2.15) transporta el error .[en] el tiempo"desde el paso t, hacia atrás al paso k.

Con lo anterior, se podría distinguir entre contribuciones a largo plazo y corto plazo, donde largo plazo se refiere a componentes para los que $k \ll t$ y corto plazo a todo lo demás.

En base a esto, se le llama explosión del gradiente cuando los componentes de largo plazo crecen exponencialmente más que los de corto plazo. Por otro lado, el *vanishing gradient* se refiere al comportamiento opuesto, que es cuando los componentes de largo plazo convergen rápidamente hacia cero, lo que hace imposible para el modelo aprender la correlación entre eventos temporalmente distantes.

2.5. Long-Short Term Memory (LSTM)

Las redes Long-Short Term Memory (LSTM) son un tipo de redes neuronales recurrentes. Estas nacen como solución al problema de desvanecimiento de gradiente presentado en las RNNs tradicionales, donde las LSTM son capaces de aprender información a largo plazo [20], recordando mejor períodos mayores de tiempo. La estructura interna de las LSTM les permite agregar o eliminar información dentro del estado de la celda de memoria, regulada mediante el uso de puertas que permiten que la información fluya a lo largo del tiempo, de esta forma permite resolver tareas más desafiantes que las RNNs tradicionales.

La arquitectura de las celdas LSTM se compone de tres puertas o *gates*:

- **Puerta de entrada (*Input gate*)**: Es la responsable de agregar nueva información a la celda, quien selecciona la información más importante y pondera según relevancia.
- **Puerta del olvido (*Forget gate*)**: Esta puerta elimina la información innecesaria o irrelevante para realizar la tarea, este paso es esencial para optimizar el rendimiento de la red.
- **Puerta de salida (*Output gate*)**: Esta puerta procede a calcular el nuevo estado oculto de la celda.

Además, también incluye una celda de memoria que se utiliza para almacenar valores en intervalos de tiempo (estado de la celda) que modula la salida. Las puertas regulan como se actualizan estos valores a lo largo del tiempo. En la Figura 2.10 se muestra gráficamente una celda LSTM donde c_t corresponde a la memoria de la celda, i_t es la puerta de entrada, la compuerta de olvido f_t y la puerta de salida o_t. Se representa la función sigmoide como σ y el estado oculto corresponde a h_t, el cual se obtiene determinando que tan vieja o nueva es la información que la celda debería ignorar, esto se refleja en las ecuaciones 2.16 a la 2.21.

$$\mathrm{i}_t = \sigma(W_{\mathrm{i}} Z_t + U_{\mathrm{i}} h_{t-1}) \tag{2.16}$$

$$f_t = \sigma(W_f Z_t + U_f h_{t-1}) \tag{2.17}$$

$$o_t = \sigma(W_o Z_t + U_o h_{t-1}) \tag{2.18}$$

$$\tilde{c}_t = \tanh(W_c Z_t + U_c h_{t-1}) \tag{2.19}$$

$$C_t = f_t \odot c_{t-1} + \mathrm{i}_t \odot \tilde{c}_t \tag{2.20}$$

$$h_t = o_t \odot \tanh(C_t) \tag{2.21}$$

donde Z_t, h_t, C_t son la entrada exógena, el estado oculto o salida y el estado de la celda en el frame t, respectivamente; i_t, f_t y o_t son los vectores de la puerta de entrada, del olvido y de salida en el frame t respectivamente. En cuanto a los pesos, se tiene que W_i, W_f, W_o y W_c son las matrices de pesos de las puertas correspondientes a la entrada Z_t; U_i, U_f, U_o y U_c son las matrices de pesos correspondientes a la salida del frame anterior.

Para una mejor comprensión del funcionamiento de la celda LSTM, se discutirá el cálculo de h_t, C_t en el frame t. La información de la red hasta el frame t esta contenida en h_{t-1} y C_{t-1}. La información en el estado de la celda se puede entender como un vector de pesos que modula la estimación de la salida h_t, que a su vez es la entrada endógena en el frame $t+1$. En este sentido, .ºlvidar.º .ªgregarïnformación se modela como ponderación con valores menores a 1 o sumando valores mayores a 0, respectivamente. Para lograr esta dinámica, h_{t-1} y Z_t se procesan para generar vectores que se operan punto a punto con el estado de celda C_{t-1}. La puerta de olvido genera un vector con valores menores a 1, de acuerdo con la función de activación sigmoidea en 2.17, que pondera C_{t-1} mediante el operador de producto puntual como se puede ver en 2.20. La puerta de entrada genera un vector que se combina con un estado de celda candidato obtenido en 2.19 utilizando el operado punto a punto. El resultado se suma a $f_t \odot c_{t-1}$ para determinar un nuevo estado de celda, es decir, C_t. Finalmente, la puerta de salida estima el siguiente estado oculto, en primer lugar o_t se estima de acuerdo con 2.18. Entonces, h_t se estima como en 2.21 para decidir qué información debe llevar el estado oculto. El nuevo estado de celda actualizado y el nuevo estado oculto se transfieren al siguiente paso de tiempo.

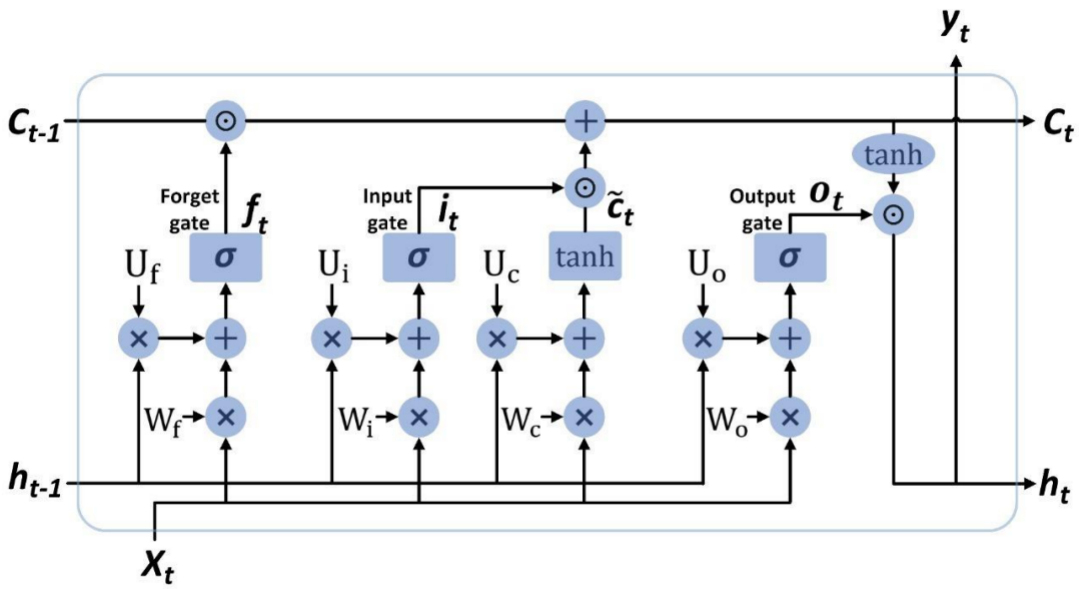

Figura 2.10: Representación gráfica de una celda LSTM.

2.5.1. Aplicaciones de la LSTM

La LSTM tiene una amplia variedad de tareas en Deep Learning, donde se destaca principalmente en labores de predicción de secuencias y análisis temporal de señales, esto enfocado

en largos períodos de tiempo. La gran capacidad de análisis secuencial de las LSTM hacen de ella una de las mejores opciones en la literatura para problemas como el de predicción de texto, traducción automática de textos, predicción de stock, reconocimiento de voz, entre otros.

2.6. End-to-end

El End-to-End (E2E) es una técnica que se refiere al entrenamiento de un sistema complejo sin especificar los modelos intermedios presentes en los diseños tradicionales. En el Deep Learning, se suele utilizar alguna variante de las redes neuronales artificiales profundas, donde es importante tanto la arquitectura de la red como el algoritmo de entrenamiento o aprendizaje. El uso de esta técnica se ha incrementado en el último tiempo, ya que su impacto ha sido excelente, en muchos ámbitos como el procesamiento voz [48] o modelos de regresión.

Los esquemas basado en E2E se están convirtiendo en un enfoque prometedor que intenta ofrecer buenos resultados sin modelos a priori [31] [56]. Estos han proporcionado resultados muy competitivos en el campo del reconocimiento automático de voz (ASR) utilizando sólo información de entrada-salida [4] [30] [65].

Para este trabajo, el E2E se ve reflejado en la ventaja que ofrece dada la capacidad de estimar la posición de los epicentros utilizando sólo las señales registradas por los sismógrafos, evitando la necesidad de crear modelos de velocidad de propagación de señales, picado de las ondas P y S de la señal sísmica y múltiples otros factores.

2.7. Estado del Arte

2.7.1. Localización de eventos con picado automático de ondas P y S

La localización de eventos sísmicos volcánicos con la selección automática de ondas P y S no se ha abordado en la literatura de manera exhaustiva. En vulcanología, los eventos sísmicos son generados por la actividad volcánica, como es el caso de los eventos de tipo volcano-tectónicos (VT), donde es posible distinguir la llegada de las ondas P y S. Con los tiempos de llegada de las ondas P y S es posible calcular el tiempo en que la onda tardó en llegar desde su origen hasta el sismógrafo que lo detectó para luego aplicar el método de triangulación explicado previamente, sin embargo, muchos sistemas no consideran los cambios de velocidad provocador por los distintos materiales que componen el medio de propagación, por lo que el error puede ser de varios kilómetros. Los modelos más actuales incluyen modelos de propagación del terreno, por lo que la distancia estimada al epicentro ya no es una circunferencia centrada en el sismógrafo [32]. En el caso de los volcanes, debido a la baja energía que transmiten las ondas, estas se disipan más rápido y por lo tanto se necesitan sensores en las cercanías para detectar las señales, por lo que se necesita una mayor precisión y se utilizan modelos de velocidad con más capas.

Para eventos sísmicos, la selección automática de fase se ha abordado en numerosas publicaciones utilizando detectores STA/LTA, curtosis y asimetría [50][3]. En los últimos años, gran parte de la investigación sobre detectores automáticos se ha centrado en el *deep learning* debido a los excelentes resultados obtenidos [66]. Se han utilizados arquitecturas de red complejas para seleccionar automáticamente los tiempos de llegada de las ondas P y S. El gran problema de estas redes es que dependen fuertemente de la base de datos específica a la que se aplican y requieren una buena relación señal a ruido (SNR), además de fases claras en los sismogramas y pocos de ellos se aplican a eventos de tipo VT. Esto puede deberse a que, a diferencia de la sismología tectónica, cuando la fuente del sismo ocurre en un volcán, la poca profundidad, la proximidad de la fuente, la menor magnitud y la compleja trayectoria de propagación, entre otros factores, hacen que la detección de la onda S sea más difícil [16].

2.7.2. Localización de eventos volcánicos basados en machine learning

Hasta donde sabemos, el problema de localización de eventos volcánicos con machine learning no se ha abordado en la literatura. Por otro lado, existen métodos de localización sísmica tectónica, que si bien es un problema similar, como se ha explicado previamente, la complejidad aumenta para el caso volcánico. En [44] se presenta un método de detección y localización de eventos sísmicos a través de redes neuronales convolucionales (CNN) procesando las señales en el tiempo usando estaciones con tres canales: componente vertical (Z), componente horizontal (R) y transversal (T). La localización se aborda como un problema de clasificación dividiendo el área de estudio en 6 grupos de aproximadamente $100[km^2]$ cada uno. Logran una correcta identificación de clúster en el 74,5 % de los casos. Otro modelo de detección y localización se muestra en [38], donde se utiliza una red de 76 estaciones de monitoreo en California, EE.UU. En este modelo, se crean imágenes 3D de la señal sísmica utilizando el SLF (función de probabilidad espacial) de un subconjunto de los pares de estaciones disponibles para crear una ventana de tiempo seleccionada, luego es posible extraer la ubicación del sismo del máximo del SLF. La separación promedio entre estaciones es $162[km]$. De un total de 2522 señales en las que se detecta actividad sísmica, en 1192 casos es posible localizar eventos.

En [28], se propuso un método de localización de eventos sísmicos tectónicos a través de CNN utilizando señales de múltiples estaciones, donde la arquitectura propuesta comienza con capas 1D que procesan la información en el tiempo para cada canal y luego una capa 2D que combina la información de las distintas estaciones. Obtiene un error de predicción inferior a $200[m]$ en el 86 % de los casos, con un área de estudio de aproximadamente $8[km^2]$.

2.7.3. Modelos de entrada de información a redes neuronales

El problema de la información con dimensión variable ha sido recurrente en el mundo del Deep Learning, ya que los modelos requieren de dimensiones predefinidas como *input* a las redes neuronales. Este problema ocurre generalmente en proyectos relacionados al procesamiento de señales como reconocimiento de voz, electrocardiogramas, señales sísmicas, etc., ya

que las señales raramente tienen igual cantidad de muestras. En [26] se aborda el problema de detección de enfermedades mediante el análisis de electrocardiogramas de largo variable, que son analizados por una red neuronal convolucional (CNN). Para solucionarlo, se hacen dos métodos principales, uno involucra un preprocesamiento y extracción de características y el otro introduce las señales sin cambios, considerando un largo fijo para las señales y repitiendo las de menor largo hasta completar la dimensión requerida Por ejemplo, se define el largo de entrada en 90 segundos, entonces una señal de 10 segundos sería repetida 9 veces para obtener la dimensión adecuada para la red. Este método obtiene resultados superiores al de extracción de características, llegando a una puntuación F1 promedio de 0,83.

En [18] se propone una Multi-Variable Long-Short Term Memory (MV-LSTM), que corresponde a una red neuronal recurrente con celdas LSTM, pero equipada con tensores en los estados ocultos que permiten aprender representaciones específicas a las variables exógenas de la serie de tiempo, lo que da lugar a un nivel de atención tanto temporal como variable. Los experimentos presentados son evaluados en base a la atención obtenida a las variables exógenas, ya que la LSTM es de por sí buena interpretando la información temporal. Las series de tiempo exógenas utilizadas incluyen variables como temperatura, presión, dirección del viento, etc. Se obtienen resultados que demuestran un desempeño de predicción comparable de MV-LSTM con respecto a los modelos basados en *encoder-decoder*.

2.7.4. Ponderación y propagación de la incertidumbre de los datos en redes neuronales

La propagación de la incertidumbre de los datos es una técnica ampliamente utilizada en la literatura, ya que su capacidad de proporcionar información adicional de la calidad de datos o desempeño de algoritmos ha llamado en los últimos años el interés de sus aplicaciones en el mundo del Deep Learning.

En [33] se propone un método para la estimación de la incertidumbre basada en redes Bayesianas y muestreo de Monte-Carlo, se logra modelar completamente diferentes fuentes de incertidumbre de predicción, así como también la incorporación de datos previos (ruido del sensor). Para computar la incertidumbre de los datos, se propaga el ruido del sensor por la red mediante el denominado Filtrado de Densidad Asumida (o sus siglas en inglés ADF - *Assumed Density Filtering*). Para éste método, se cambian las activaciones de cada capa, incluyendo de input y output, por distribuciones de probabilidad. En resumen, el ADF modifica el paso hacia adelante de una red neuronal para generar no solo predicciones de salida, sino también sus respectivas incertidumbres de datos. Para ello, ADF propaga la incertidumbre de entrada que, en un escenario robótico, corresponde a las características de ruido del sensor.

En [42] se propone la ponderación y propagación de la incertidumbre en métodos de DNN-HMM (Deep Neural Network - Hidden Markov Model) para el reconocimiento de voz. Esto se realiza para la cancelación de ruido y se define una variable que pondera la incertidumbre considerando la definición de la varianza de la incertidumbre en la cancelación de ruido en un filtro pasa banda de [63]. Esta ponderación de la incertidumbre se propaga por la Deep Neural Network como una variable aleatoria. Este enfoque logra resultados que logran una sustancial reducción del WER (*Word-Error-Rate*) con entrenamiento limpio.

Capítulo 3

Metodología e Implementación

En este capitulo se presentarán todos los aspectos que constituyen la metodología e implementación del trabajo. En la Figura 3.1 se presenta un esquema con las principales partes de la metodología. El proceso de obtención de la base de datos, parámetros a utilizar y comparación de resultados se detallan en profundidad. Pero en primer lugar, es necesario mencionar el criterio de acierto, barrido de hiperparámetros y una breve descripción del problema a superar.

Para lograr un modelo capaz de localizar los epicentros con el menor error posible, se definió en primer lugar una localización *acertada*, el OVDAS impuso que una predicción con menos de 1 km de error es considerada acertada o aceptable. En base a esto, el porcentaje de predicciones del modelo con un error menor a 1 km se definió como *porcentaje de acierto* y en base a este porcentaje se guardan o desechan configuraciones del modelo.

Una vez definido el criterio a mejorar, se procede a realizar un barrido de hiperparámetros con el fin de entrenar múltiples modelos con todas las configuraciones posibles, para así lograr reconocer una tendencia del comportamiento de este y encontrar la estructura con el mayor porcentaje de acierto posible.

Además, también es necesario comparar los resultados de los modelos con el creado en la etapa anterior de este trabajo, donde debido al problema de la intermitencia de registros mencionada, sólo se utilizaron 3 estaciones de monitoreo que estaban presentes en toda la base de datos, es por esto que se debe superar el porcentaje de acierto de la etapa anterior, pero utilizando la DB completa.

3.1. Base de datos

El trabajo realizado tuvo el apoyo de la Universidad de la Frontera, la cuál proporcionó las base de datos (DB) que consiste de 1884 eventos sismológicos pertenecientes al Volcán Chillán. Estos eventos son de tipo Volcano-Tectónicos (VT) obtenidos de una red de estaciones de monitoreo perteneciente al Observatorio Volcanológico de los Andes Sur (OVDAS). De estos eventos se registra la señal sismológica en el eje vertical registrada por cada sismógrafo a una

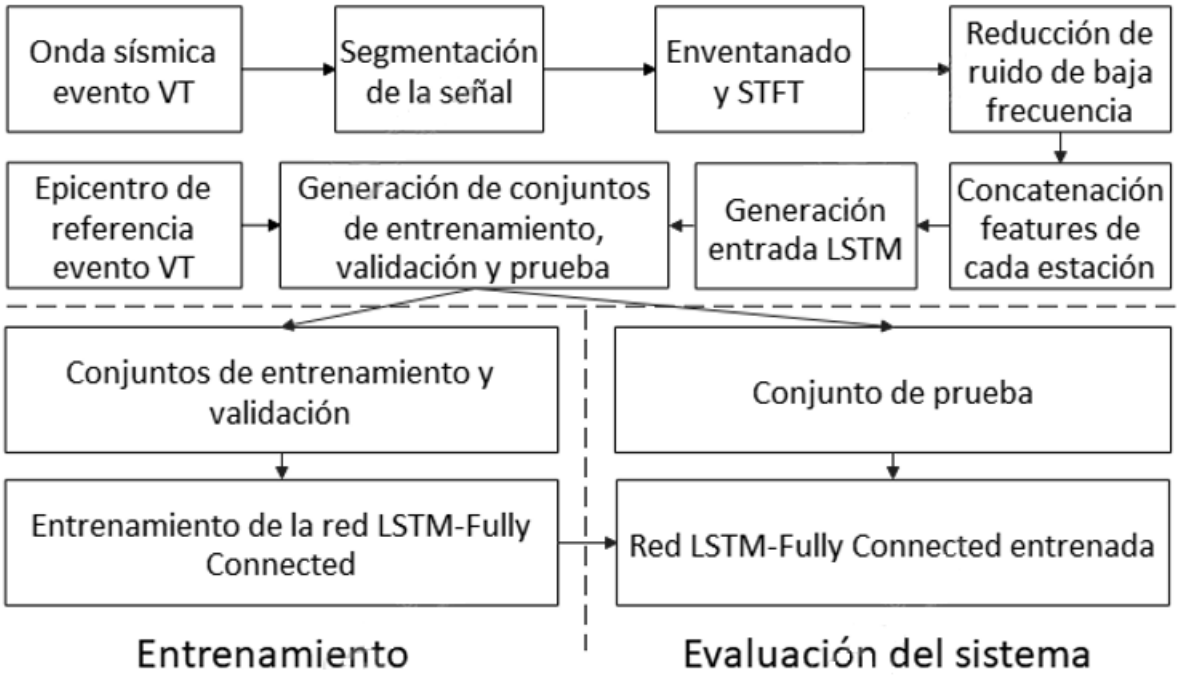

Figura 3.1: Pipeline que explica el proceso para entrenar la red LSTM.

frecuencia de $100Hz$, hora de inicio y término del evento y su respectiva ubicación.

La base de datos se divide en 3 periodos, el primero corresponde a eventos pertenecientes a los años 2015 al 2017 y consta de 953 eventos, el segundo consiste de registros del año 2019 con 241 eventos y el tercer periodo contiene registros del año 2020 con un total de 690 eventos.

En cuanto a las estaciones de monitoreo, estas fueron aumentando con el pasar del tiempo, donde la primera base de datos (2015 a 2017) contiene los registros de sólo 5 sismógrafos, la base de datos del 2019 contiene 7 sismógrafos y la del 2020 consta de las señales de 10 estaciones. Es importante destacar que no todas las estaciones están activas para todos los eventos, se requiere que para un determinado sismo estén al menos 3 sismógrafos recibiendo la señal para así localizar el epicentro. En la Figura 3.2 se muestra la ubicación de las estaciones de monitoreo sobre el Volcán Chillán.

En cuanto a la ubicación de los eventos proporcionada por el OVDAS, en primer lugar se realiza una identificación visual de la onda P y, de ser identificable, S, lo cual se le llama *picado* de las ondas internas del sismo. Luego para la localización se utiliza el software HYPO71, al cual se le ingresan los picados de las señales junto a un modelo de velocidad de las señales en el terreno. Este software entrega las localizaciones consideradas como *target* o *ground truth* de este trabajo.

La distribución de los sismos se mantiene en el comportamiento usual de los volcanes, donde presentan un *enjambre* de eventos en el área más cercana al volcán, sin embargo existe un porcentaje de eventos más lejanos. El área de estudio para la localización de los sismos queda definida por un rectángulo de 31.2 km de ancho (longitud) correspondiente y 33.3 km de alto (latitud), lo que genera un área de 1038.96 km^2, esto se puede observar en la Figura

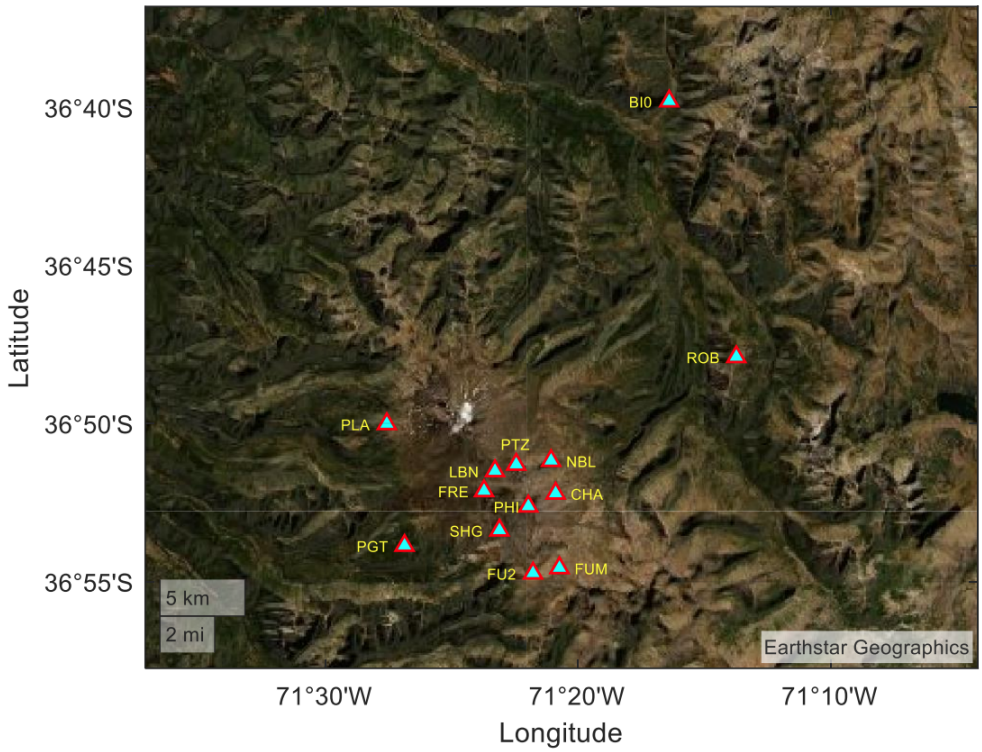

Figura 3.2: Ubicación de las estaciones de monitoreo, las estaciones presentes en la base de datos son: FRE, FU2, LBN, NBL, PLA, ROB, SHG, BIO, CHA y CHS.

3.3 que muestra los eventos del primer período de la base de datos.

3.1.1. Problema de intermitencia de registros

En la base de datos, si bien se tienen hasta 10 estaciones de monitoreo, no todos los eventos cuentan con los registros de todos los sismógrafos, como se observa en la Figura 3.4, donde los sensores *FU2*, *PLA* y *ROB* no están activos durante ese evento por lo que no contienen registros, por otro lado, las estaciones *LBN*, *CHA* y *CHS* no existen en la base de datos del 2015 por lo que rellena su registro con ceros. Esto genera un problema en el entrenamiento de la red, ya que una entrada variable de señales sísmicas no es posible por rigurosidades de las dimensiones de entrada de la red neuronal.

Si bien la idea del trabajo es utilizar la base de datos completa, es decir, los registros de todas las estaciones de monitoreo, resultó ser forzosamente necesario desechar la estación BIO debido a que está presente en sólo el 10 % de los eventos, siento un sensor muy perjudicial al aprendizaje de la red neuronal.

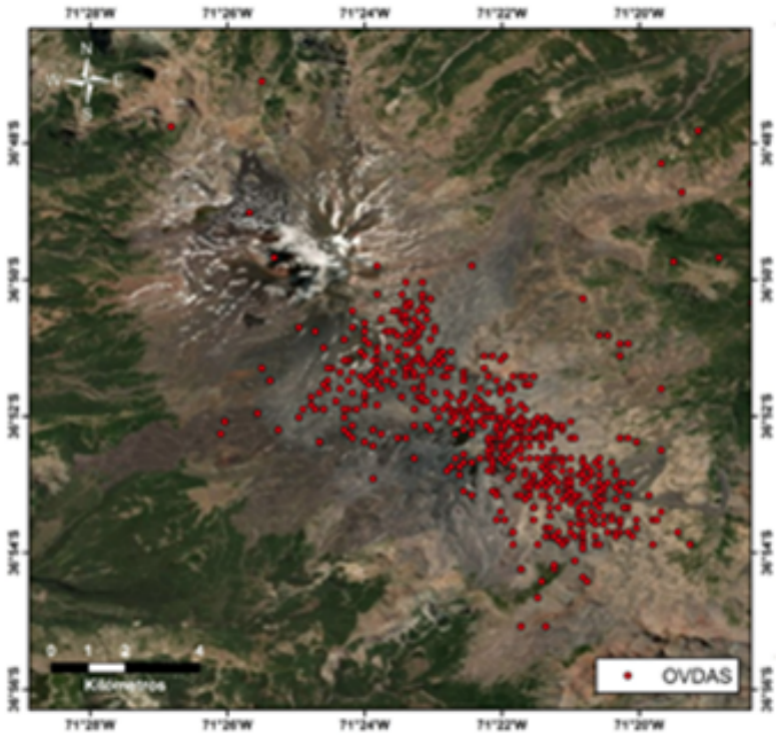

Figura 3.3: Representación geográfica de eventos de la base de datos del Volcán Chillán.

3.2. Elección modelo

Las redes neuronales recurrentes se destacan por su habilidad de aprender secuencias o variantes a lo largo del tiempo [37]. Por esta razón, en un problema de localización sísmica donde es esencial estimar el tiempo de llegada de las ondas P y S, es de esperar que una red recurrente tenga la capacidad de resolver el problema. Sin embargo, como se menciona anteriormente, las redes neuronales ordinarias conllevan una dificultad de entrenar debido al problema del desvanecimiento del gradiente. Según [57], este problema evita que las RNNs aprendan más de 10 pasos. Con tal de superar este desafío, como se menciona en la sección 2.5, la LSTM se creó con el objetivo de preservar la información importante por una mayor cantidad de pasos, al mismo tiempo que se descarta la información irrelevante en una secuencia de tiempo. Al hacerlo, en caso de entrenarse una LSTM para localizar los epicentros de eventos sísmicos, esta debería ser capaz de registrar los tiempos de llegada de las ondas P y S. Por el contrario, esta debería ignorar cualquier componente de la señal que no sea relevante para la tarea, como por ejemplo el ruido.

3.3. Hiperparámetros

Los hiperparámetros son un pilar fundamental en el correcto funcionamiento del modelo de Deep Learning. Es necesario explorar una basta cantidad de configuraciones para cada tipo de arquitectura indagada, ya que en ciertos casos, hasta el más mínimo cambio en una variable conlleva resultados totalmente distintos, es por esto que en cada investigación

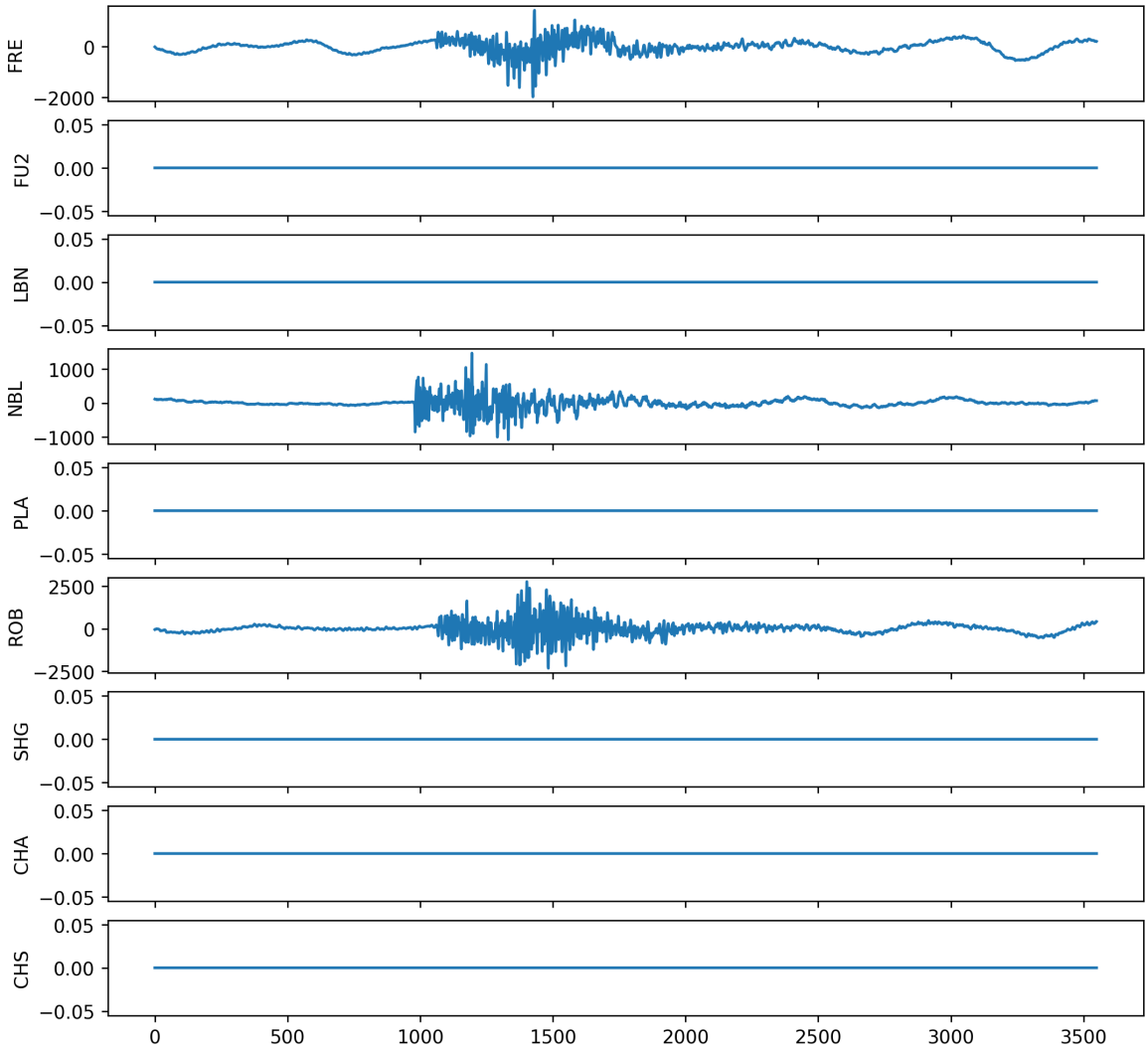

Figura 3.4: Señales completas en el tiempo de los 9 sismógrafos correspondiente al evento 1 de la base de datos del 2015.

relacionada al área de Deep Learning, es imperiosa la etapa de barrido de hiperparámetros.

Cada configuración marca una tendencia del comportamiento del modelo, por lo que cada resultado puede abrir a un nuevo barrido de hiperparámetros necesario, donde es importante no ser redundante en las pruebas realizadas. Para este trabajo, las configuraciones intentadas consistieron tanto de hiperparámetros como de parámetros y funciones en el preprocesamiento. En la Tabla 3.1 se muestran las variables que fueron configuradas en la búsqueda del modelo óptimo.

Para todos los modelos se utilizó el optimizador *ADAM*, ya que en un principio se realizaron pruebas con distintos optimizadores, donde *ADAM* resultó ser ampliamente el optimizador que mejor se desempeñó para esta tarea. En cuanto a la función de pérdidas o *loss function*, la métrica impuesta por el OVDAS de localizaciones a menos de un kilómetro de error no es directamente aplicable como función de pérdidas para el entrenamiento de la red por lo que se probaron las dos métricas por excelencia de modelos regresivos: el *Mean*

Tabla 3.1: Hiperparámetros de la LSTM y variables de preprocesamiento en las que se exploró durante la creación del modelo.

Hiperparámetros LSTM	**Variables preprocesamiento**
Tasa de Aprendizaje	Largo de ventanas
Cantidad de nodos LSTM	Traslape de ventanas
Cantidad de nodos Fully Connected	Bandas de frecuencia eliminadas
Porcentaje de Dropout	Tiempo inicial de la señal a considerar
Cantidad de capas LSTM	Tiempo final de la señal a considerar
Cantidad de capas Fully Connected	Tipos de normalización
Tamaño del batch	
Función de activación ultima capa FC	

Squared Error (MSE) y el *Mean Absolute Error* (MAE). En esta función ocurre algo similar que con el optimizador, la *loss function* MAE se coronó como la opción en todas las pruebas. Por último, se utilizó la función de activación de tangente hiperbólica *tanh()* que trae por defecto la celda LSTM de Keras.

3.4. Variable para propagar la incertidumbre

Un aspecto importante del trabajo es la creación e implementación de una parámetro que pueda propagar la incertidumbre de cada sensor por la red neuronal. Por lo general, los métodos que permiten propagar la incertidumbre por las redes neuronales son utilizando redes neuronales prealimentadas (*feed-forward*) como se muestra en los casos mencionados en la Sección 2.7.4. Esto generaría la necesidad de cambiar la arquitectura del modelo, lo cuál no está dentro de las posibilidades debido a que la arquitectura de red recurrente con celdas LSTM es la base del trabajo. Sin embargo, siguiendo la línea de ponderar la incertidumbre, se crea una variable que considera el SNR de cada señal y la cantidad de estaciones activas de cada evento. Esto debido a que, en primer lugar, el SNR es un parámetro fundamental para el análisis de las señales, donde un SNR muy bajo implica una observación de menor calidad de los componentes de la señal. Por otro lado, en la localización de eventos sísmicos es directo que una mayor cantidad de estaciones de monitoreo permiten una localización más precisa del epicentro, es por esto que un evento con, por ejemplo, 9 estaciones activas debe ser más confiable que una con sólo 3, por lo que el entrenamiento debe generar más énfasis en casos con más sensores activos.

3.4.1. Cálculo SNR

El marco teórico se define el SNR o *signal-to-noise ratio*, sin embargo, para calcular el SNR de las señales sísmicas de las base de datos no se conoce la señal limpia ni el ruido por separado. Es por esto que se calcula el SNR utilizando la ecuación 3.1, ya que utiliza la potencia promedio de la señal completa (señal más ruido) y la potencia promedio del ruido por sí solo.

$$SNR = \frac{\overline{S_X}}{\overline{S_N}} = \frac{\overline{S_{(N+X)}} - \overline{S_N}}{\overline{S_N}} \tag{3.1}$$

donde S_X corresponde a la potencia de la señal limpia (sin ruido), S_N es la potencia del ruido y $S_{(N+X)}$ la señal completa (señal más ruido).

Ahora, para distinguir entre la señal completa y el ruido, se considera ruido a la señal comprendida entre los segundos 0 a 8 de las señales (ver Figura 3.5), ya que durante ese período todo el registro corresponde a ruido natural del ambiente, que se apodará $N(x)$. Por otro lado, la señal comprendida entre 0 y 15 segundos corresponde a la señal completa que contiene tanto a la señal sísmica como el ruido, apodada $R(x)$. Se procede a calcular la potencia promedio de ambas señales usando las ecuaciones 3.2 y 3.3.

$$\overline{S_N} = \sum_{x=x_\text{i}}^{x_r} \frac{N(x)^2}{x_r} \tag{3.2}$$

$$\overline{S_{(N+X)}} = \sum_{x=x_\text{i}}^{x_f} \frac{R(x)^2}{x_f} \tag{3.3}$$

Donde x_i denota la muestra de inicio de la señal, x_r la muestra en que el considerado ruido termina y x_f la muestra en la que termina la señal.

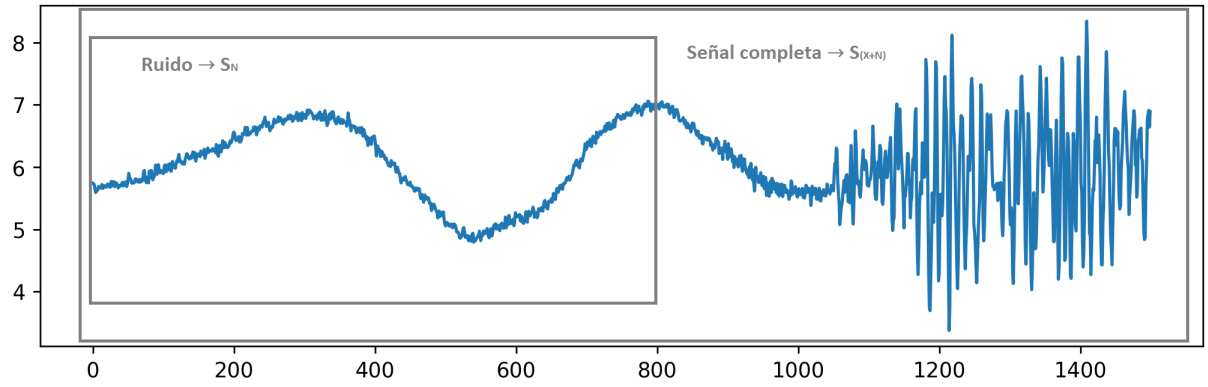

Figura 3.5: Señal sísmica entre 0 y 8 segundos, donde se representa las secciones consideradas como ruido y como señal completa.

3.4.2. Definición de la variable de incertidumbre

Se propone una variable de incertidumbre con valores entre 0 y 1, la cuál será distinta para cada sensor en cada evento. En primer lugar, se define que las estaciones inactivas tienen un valor de 0 en su confiabilidad directamente. En cuanto al SNR, este toma valores entre $-\infty$ y $+\infty$, el cual se transforma a valores entre 0 y 1 distinguiendo entre distintos segmentos de valores. La transformación aplicada al SNR se muestra en la ecuación 3.4.

Tabla 3.2: Relación establecida entre el número de estaciones activas con un valor entre 0 y 1.

Número de sensores activos	**Valor asignado para la variable (NE)**
3	0.2
4	0.33
5	0.5
6	0.6
7	0.75
8	0.85
9	1

$$\overline{SNR} = \begin{cases} 1 & \text{si SNR} \geq 600 \\ \frac{SNR}{3000} + 0{,}8 & \text{si SNR} \in [10, 600) \\ \frac{SNR}{50} + 0{,}6 & \text{si SNR} \in [1, 10) \\ \frac{SNR}{10} + 0{,}5 & \text{si SNR} \in [0, 1) \\ \frac{1+SNR}{10} + 0{,}4 & \text{si SNR} \in [-1, 0) \\ \frac{100+SNR}{250} & \text{si SNR} \in [-100, -1) \\ 0 & \text{si SNR} \leq -100 \end{cases} \tag{3.4}$$

Por otro lado, en la Tabla 3.2 se muestra relación definida para la cantidad de sensores activos en cada evento, a este parámetro se apodará NE

Con ambos parámetros definidos, el valor final de la variable de incertidumbre se calcula promediando ambos valores, donde además, se establece directamente como 1 en el caso de que los 9 sensores estén activos. En la ecuación 3.5 se muestra la función para calcular esta variable.

$$\overline{VAR} = \begin{cases} 1 & \text{si NE} = 1 \\ \frac{\overline{SNR}+NE}{2} & \text{si NE} \neq 1 \end{cases} \tag{3.5}$$

3.5. Preprocesamiento

3.5.1. Preprocesamiento señales sísmicas

El OVDAS entrega la señal en el tiempo registrada por cada sismógrafo, la cuál se observa en la Figura 3.6, como se mencionó anteriormente, las señales se analizarán en el dominio de la frecuencia aplicando la *short-time Fourier transform* a las señales temporales para así lograr una representación en frecuencia sin perder información temporal de los registros.

Todos los eventos sísmicos tienen duraciones distintas, lo que deriva en registros sísmicos de largos variables, es por esto que es necesario recortar las señales a una sección de tiempo de interés. Las señales están todas alineadas en base a la estación de referencia *FRE*, en la cuál se sitúa el arribo de la onda P en el segundo 10. En base a esto, dada el área de estudio, se

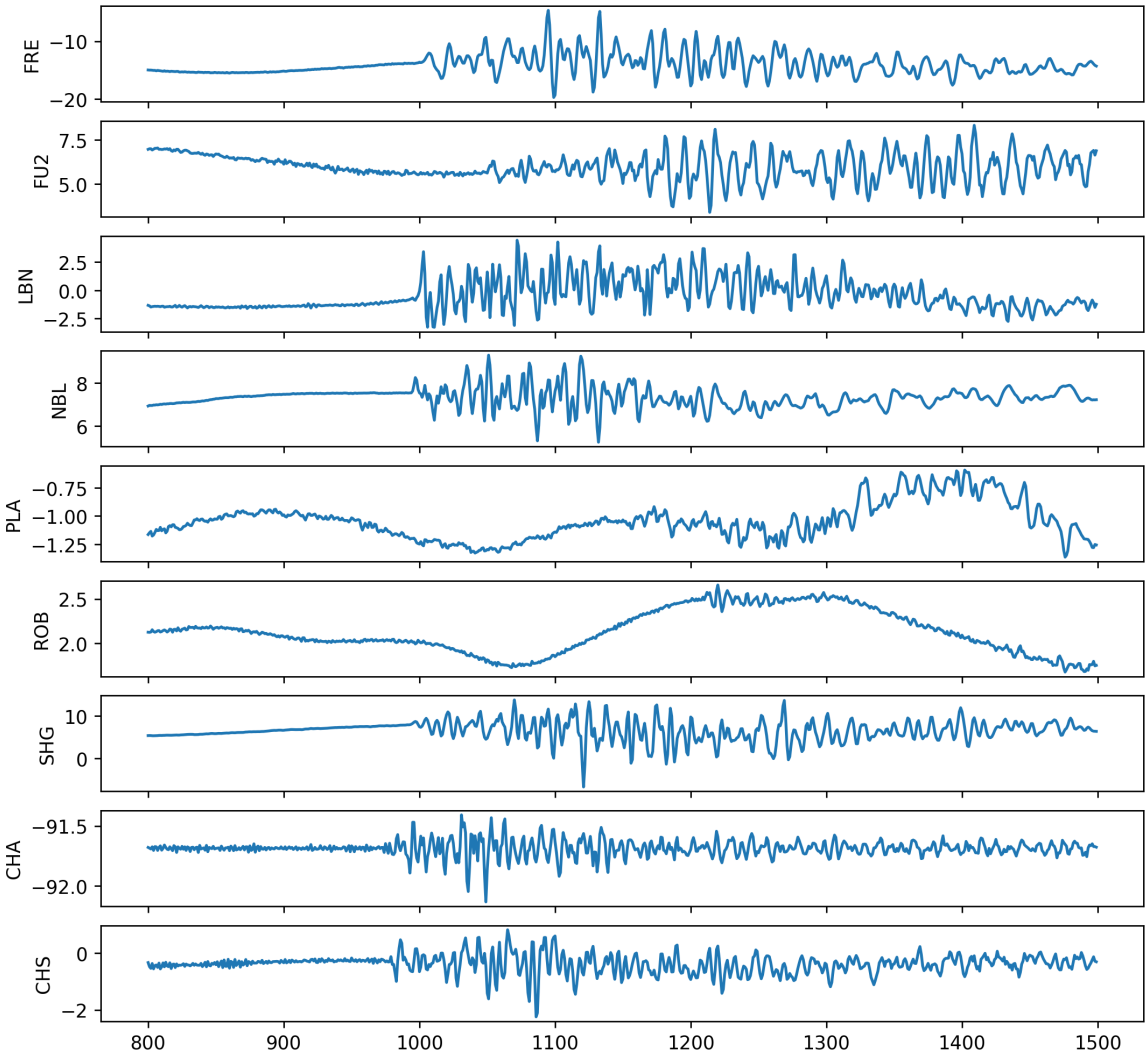

Figura 3.6: Señal en el tiempo de los 9 sismógrafos correspondiente al evento 100 de la base de datos del 2020.

considera que la sección de interés de las señales (arribo ondas P y S) quedan bien definidos entre los segundos 8 y 15 de las señales, es decir, 2 segundos antes del arribo de la onda P en la referencia y 5 segundos después de esta, esto se refleja en la Figura 3.6 donde la señal se grafica desde la muestra 800 hasta la 1500, que a una frecuencia de muestreo de $100Hz$ corresponde a los segundos 8 al 15.

En cuanto a la aplicación de la STFT, dado que las señales sísmicas son registradas a una frecuencia de $100Hz$, la transformada de Fourier se realiza con 64 puntos (nfft), que corresponde a la potencia de 2 mayor y más cercana a la mitad de la frecuencia de muestreo. Para aplicar la STFT se utilizó la ventana de Hamming, donde el traslape y el largo de las ventanas son variables que se exploraron en la búsqueda de parámetros óptimos. De la STFT, la matriz obtenida de cada sensor de cada evento es una de dimensiones $n\_frames \times n\_bines$, donde n_frames corresponde a la cantidad de ventanas obtenidas, la cual se obtiene de la ecuación 3.6, donde N corresponde a la cantidad de muestras de la señal, w es el tamaño de muestras de cada ventana de Hamming y sp es el porcentaje de traslape (*shift percentage*)

de las ventanas. Por otro lado, *n_bines* son la cantidad de bandas de frecuencia calculados por la transformada de Fourier, que corresponde a la mitad de los puntos FFT más uno, $n\_bines = \text{nfft}/2 + 1$. Sin embargo, se elimina una cantidad de bandas de las primeras bandas de frecuencia para eliminar el ruido de baja frecuencia, cantidad que llamaremos *bines_elim*, que como se observa en la Figura 3.6, las señales de las estaciones ROB y SHG presentan un alto ruido de baja frecuencia. En la Figura 3.7 se muestra gráficamente el proceso de enventanado y cálculo de la transformada de Fourier.

$$n\_frames = \frac{N}{w * sp} - 1 \tag{3.6}$$

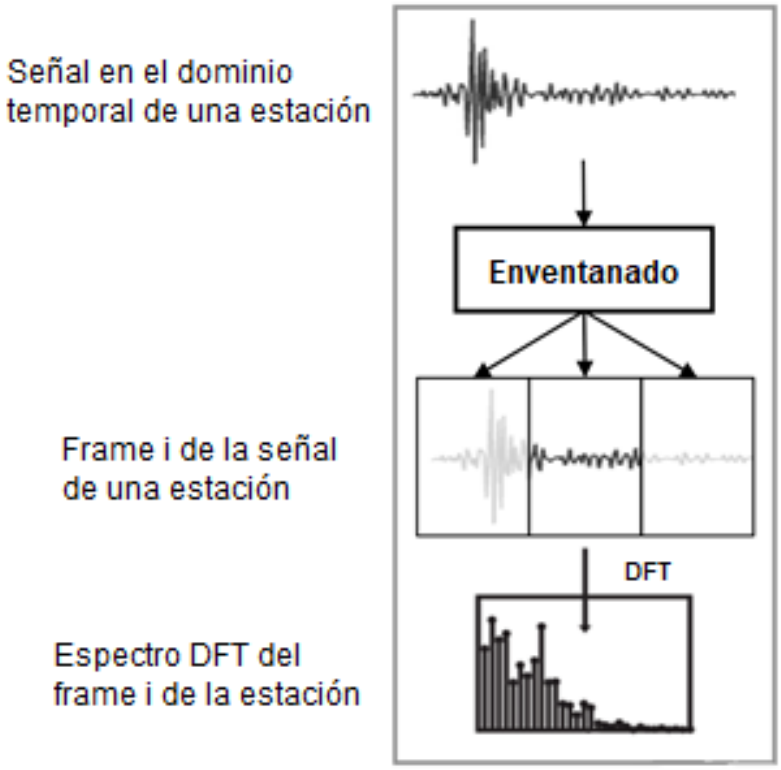

Figura 3.7: Enventanado y cálculo de la transformada de Fourier para las señales en el tiempo.

Luego de la aplicación de la STFT, se procede a normalizar sus valores, los cuales son complejos, por lo que en primer lugar se les calcula el módulo para pasarlos a valores reales ($\mathbb{R}$)y al módulo de la STFT se le calcula el logaritmo natural.

Luego se realiza una normalización *Z-score* a cada banda de frecuencia de cada sensor de cada evento, por separado, esto con el objetivo de que cada banda de frecuencia tenga un promedio de cero y una desviación estándar de uno, esto se muestra en la ecuación 3.7, donde se define una banda de frecuencia como $F = [f_0; ...; f_{\frac{nfft}{2}-1}]$.

$$Z_F = [z_0; ...; z_{\frac{nfft}{2}+1-bines\_elim}] \quad con \quad z_{\text{i}} = \frac{f_{\text{i}} - \mu_{\text{i}}}{\sigma_{\text{i}}} \tag{3.7}$$

Se procede a concatenar las STFT normalizadas de cada sensor para estructurar la matriz de entrada o *input* de la red neuronal, como se observa en la Figura 3.8 y tendrá dimensiones de:

$$\{n\_frames \times 9 * n\_bines\} = \left\{ \frac{N}{w * sp} - 1 \times 9 * (\frac{nfft}{2} + 1 - bines\_elim) \right\} \tag{3.8}$$

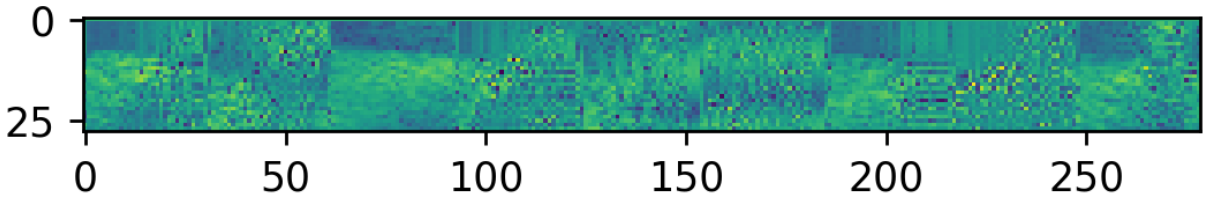

Figura 3.8: Resultado del preprocesamiento e input de la red neuronal, es un ejemplo con un porcentaje de traslape del 50 %, una ventana de Hamming de 500$[ms]$ y eliminando 2 bins de las frecuencias mas bajas.

Finalmente, los eventos de la base de datos completa son desordenados aleatoriamente para dividirlos en los conjuntos de *Train* (80 %), *Validation* (10 %) y *Test* (10 %).

3.5.2. Normalización coordenadas

Como se mencionó anteriormente, el área de estudio tiene aproximadamente 1038,96km^2, la cual comprende entre las latitudes $-37{,}824$ a $-36{,}723$ y longitudes desde $-71{,}155$ hasta $-71{,}507$. Estas coordenadas fueron normalizadas a valores entre 0 y 1, donde estas latitudes y longitudes fueron transformadas a distancias en el sistema internacional, despreciando la curvatura de la Tierra, ya que el área es lo suficientemente pequeña para no influir en los cálculos.

Para la normalización, se busca los 4 eventos más extremos de cada orientación *NO*, *NE*, *SO* y *SE* y se realiza una transformación lineal de las coordenadas en (lon, lat) a un (x, y) normalizado con $x, y \in [0, 1]$.

3.6. Arquitectura LSTM

La red LSTM se compone de capas, las cuales cada una de estas tiene un gran número de unidades o neuronas. El modelo desarrollado se probaron redes de entre 2 a 5 capas LSTM. Estas van seguidas de neuronales *Fully Connected* (FC) y una capa final de salida tipo FC de 2 neuronas, las cuales adoptan los valores de las coordenadas de estimación del epicentro del evento sísmico. Es importante mencionar también que se implementa un *Dropout* después de cada capa FC con el objetivo de aumentar la capacidad de generalización del modelo. La arquitectura de la red LSTM propuesta se observa en la Figura 3.9.

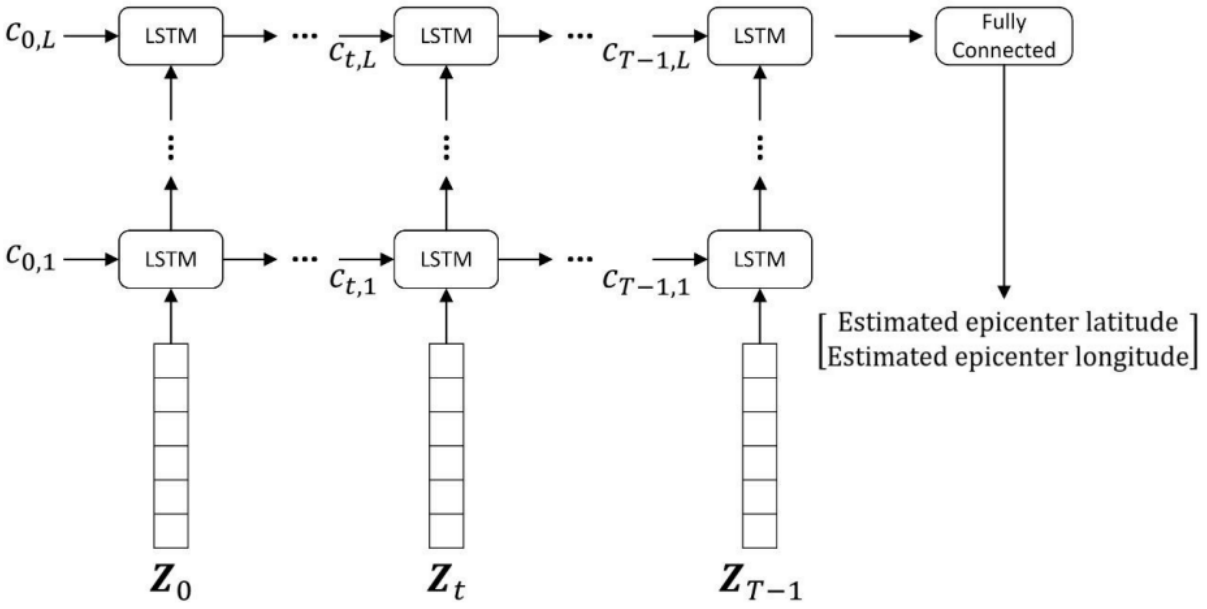

Figura 3.9: Representación gráfica de la arquitectura de la red LSTM implementada. $C_{t,l}$ corresponde al estado de la LSTM en el frame t de la capa l; Z_t es el vector de entrada correspondiente al frame t; T es la cantidad de frames de la señal de entrada y L es la cantidad de capas LSTM.

3.7. Posición variable de incertidumbre

Se hicieron dos pruebas para la utilización de la variable de incertidumbre, la primera consiste en agregar esta variable como una fila en la parte superior del input de la red neuronal (ver Figura 3.10).

La segunda prueba es multiplicar la variable calculada por los valores de la STFT de cada sensor, como se muestra en la Figura 3.11.

3.8. Modelo LSTM previo

Como se menciona anteriormente, este trabajo tiene una versión previa, donde la principal diferencia es la base de datos utilizada para el entrenamiento de la red. Dada la intermitencia de los registros de cada sensor, en el trabajo previo se optó por mantener sólo los registros de las 3 estaciones más presentes en toda la base de datos, por lo que la entrada a la red consiste en el cálculo de la STFT y posterior concatenación de sólo 3 señales sísmicas. Este modelo obtuvo un 48,5 % de acierto con un total de 1532 eventos.

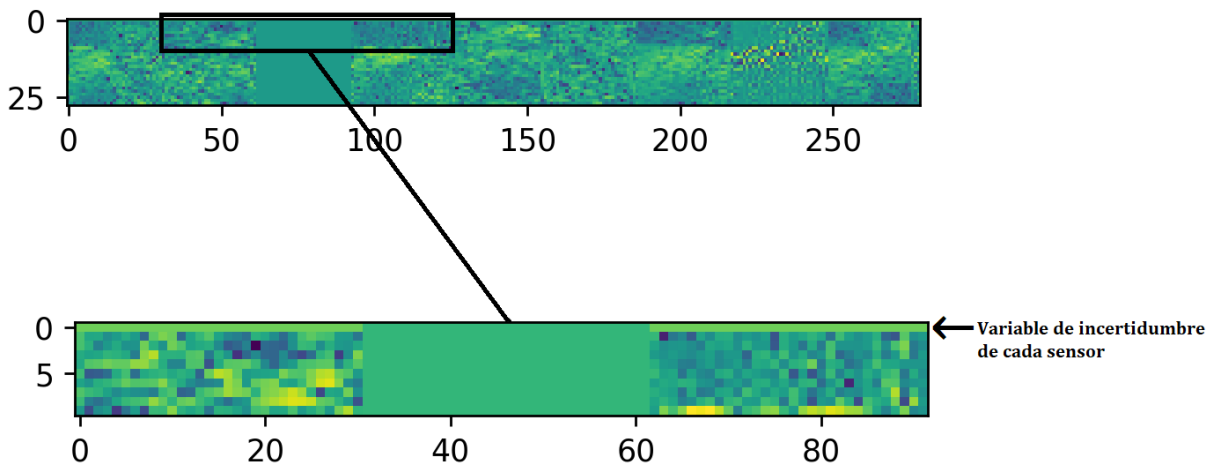

Figura 3.10: Ejemplo matriz de entrada a la red neuronal para el caso de posicionar la variable de incertidumbre como una fila al inicio de la matriz. Se realiza un zoom a las 10 primeras filas de los sensores 2, 3 y 4 para una mejor comprensión de la presencia de la variable. Se observa además que el sensor 3 está apagado para este evento sísmico.

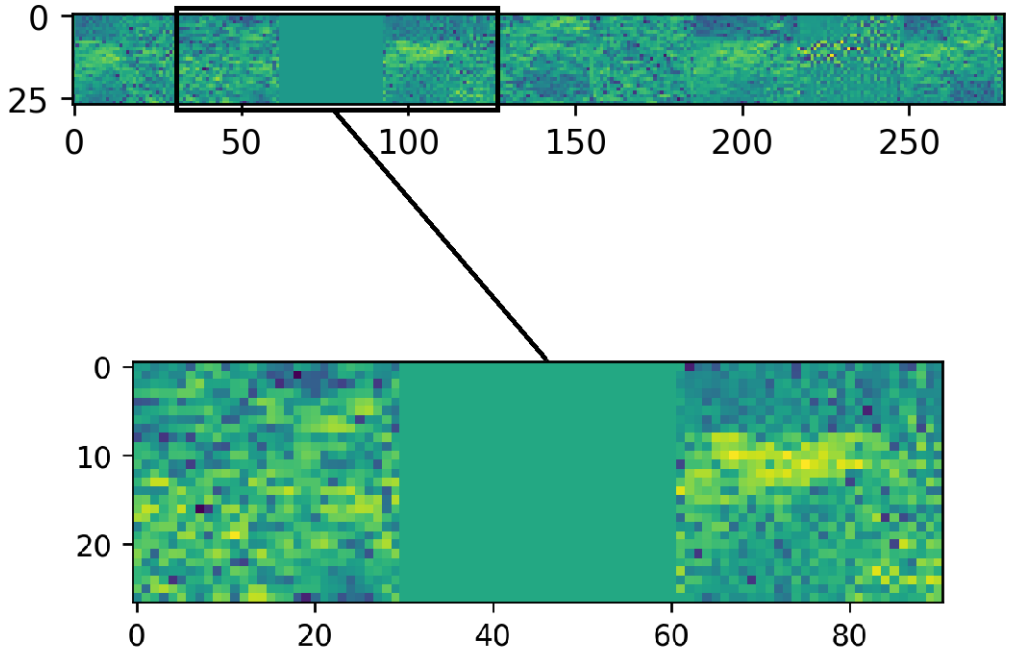

Figura 3.11: Ejemplo matriz de entrada a la red neuronal para el caso de multiplicar la variable de incertidumbre por el resultado de la STFT. Se realiza un zoom de los sensores 2, 3 y 4 para una mejor comprensión de la presencia de la variable. Se observa además que el sensor 3 está apagado para este evento sísmico.

Capítulo 4

Resultados

En este capítulo se presentarán los resultados obtenidos por el modelo descrito, donde se listarán los hiperparámetros estudiados, las distintas aplicaciones de la variable de incertidumbre, los resultados del modelo original y su comparación en desempeño con los modelos semi-automáticos utilizados por el OVDAS hoy en día.

4.1. Barrido de hiperparámetros

En la Tabla 3.1 se especifican todas las variables que se configurarán en la búsqueda del modelo que presente el mejor desempeño en el llamado porcentaje de acierto.

Para la tasa de aprendizaje o *learning rate* (LR), TensorFlow proporciona un learning rate por defecto de 0.001, por lo que se realizaron pruebas en torno a ese número, intentando valores entre 0.00001 y 0.01, donde el mejor resultado se obtuvo con 0.0008. Para la cantidad de nodos LSTM y Fully Connected, en la literatura se recomienda siempre utilizar potencias de 2 para estos hiperparámetros, en este caso se intentaron valores entre 32 a 1024 nodos en distintas combinaciones para ambos tipos de celdas. Para la LSTM tanto 256 como 512 nodos muestran un comportamiento bastante similar, por lo que se escoge 256 nodos LSTM debido que reduce significativamente el costo computacional del entrenamiento. Por otro lado, para las capas Fully Connected el mejor desempeño se obtuvo con 256 nodos.

En cuanto a la cantidad de capas LSTM y Fully Connected, se intentaron modelos con 1 a 17 capas LSTM y de 1 a 10 capas FC, sin considerar la capa de salida de 2 nodos. Los mejores resultados se obtuvieron con 5 capas LSTM y 4 capas FC. El dropout se incorpora entre cada capa FC, este se configuró asignándole valores entre 5 % y 15 %, variando también en cuantas de las capas se aplicó. El óptimo se encontró con un 10 % de dropout en la penúltima y ultima capa oculta FC.

El valor tamaño del batch de entrenamiento se varió entre 8 a 128 muestras, donde un tamaño de 32 muestras se coronó rápidamente como la mejor opción, lo cual es lo usual en la literatura. Por último, se realizaron pruebas con distintas funciones de activación para la capa FC de salida de 2 nodos, probando funciones como ReLU (*Rectified Linear Unit*), tanh

(tangente hiperbólica), sigmoidea y lineal que trae por defecto las capas FC de TensorFlow. En esta prueba, la función ReLU muestra un mejor desempeño que el resto de funciones. En la Tabla 4.1 se muestran todos los hiperparámetros mencionados.

Tabla 4.1: Hiperparámetros óptimos encontrados en las configuraciones estudiadas.

Hiperparámetro	**Valor óptimo encontrado**
Tasa de aprendizaje	0.0008
Cantidad de nodos LSTM	256
Cantidad de nodos Fully Connected	256
Porcentaje de Dropout	10 %
Cantidad de capas LSTM	5
Cantidad de capas Fully Connected	04
Tamaño del batch	32
Función de activación última capa FC	ReLU

4.2. Parámetros de preprocesamiento

Para el preprocesamiento de la base de datos, se configuraron las variables mostradas en la 3.1. En primer lugar, el largo de las ventanas de varió entre 0.05 segundos a ventanas de 2 segundos, eso a una tasa de muestreo de $100Hz$ son entre 50 a 2000 muestras, por la misma linea se intentaron traslapes de ventanas entre 0 % y 75 %. Todas las pruebas indicaron que una ventana de 0.5 segundos se desempeñó mejor y un traslape del 50 % resultó ser el óptimo aunque ligeramente por sobre 66 %. Estas dos variables proporcionaron el equilibrio del *trade-off* de resolución temporal y frecuencial.

Se eliminaron las dos bandas de frecuencia más bajas, resultado de pruebas de eliminar entre 0 y 5 bandas. La señal de interés a considerar se varió entre tiempos iniciales de los 0 a los 9 segundos y tiempos finales de 14 a los 25 segundos, donde tanto el análisis de las señales como las pruebas demostraron que el recorte óptimo de las señales es desde los 8 a los 15 segundos. Finalmente, las normalización utilizada en todo momento fue la llamada *Z-score* sin embargo, calcular el logaritmo natural del valor absoluto de la STFT antes de aplicar el *Z-score* mejoro sustancialmente el desempeño del modelo.

Tabla 4.2: Variables de preprocesamiento óptimas encontradas en las configuraciones estudiadas.

Variable de preprocesamiento	**Valor óptimo encontrado**
Largo de ventanas	0.5 s
Traslape de ventanas	50 %
Bandas de frecuencia eliminadas	2
Tiempo inicial de la señal a considerar	8 s
Tiempo final de la señal a considerar	15 s
Tipo de normalización	Z-score del logaritmo natural

4.3. Modelos LSTM creados

A continuación se mostrarán los resultados obtenidos por todos los modelos LSTM creados, los cuales serán comparados con el modelo antiguo mencionado en la Sección 3.8 y finalmente se muestra un gráfico que compara todos estos modelos.

4.3.1. LSTM - Variable de incertidumbre agregada en la primera fila del input

El desempeño de este modelo en cuanto al porcentaje de aciertos, se refleja en el gráfico de frecuencia acumulada de la Figura 4.1, el cual es de 51,08 % con un error promedio de 1,791km.

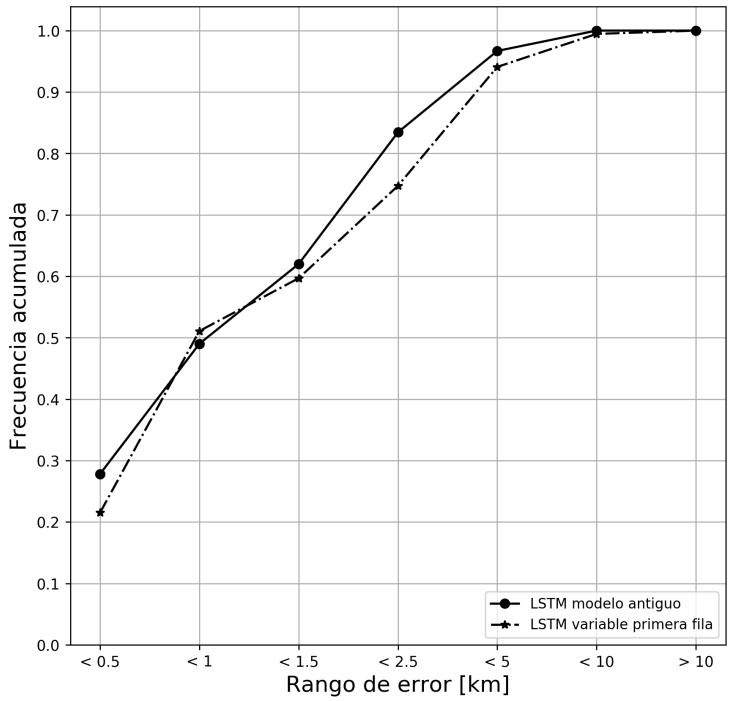

Figura 4.1: Frecuencia acumulada del error del modelo LSTM con la variable de incertidumbre agregada en la primera fila, se compara con el desempeño del modelo antiguo.

En la Figura 4.2 se muestra una representación gráfica (o geográfica) de las localizaciones del modelo, en coordenadas de latitud y longitud, por motivos de simplicidad visual, no se especifica una relación directa entre predicción y referencia.

Como se aprecia en la Figura 4.2, los eventos sísmicos forman un llamado *enjambre* de eventos, es por esto que se calcula un *centroide* de los eventos el cual denota la localización media de los eventos de entrenamiento, para así realizar una medición de la precisión del modelo considerando la distancia de un evento determinado al *centroide* de todos estos, la cual se observa en la Figura 4.3, que además muestra el porcentaje de eventos de entrenamientos a las determinadas distancias del centroide de referencia.

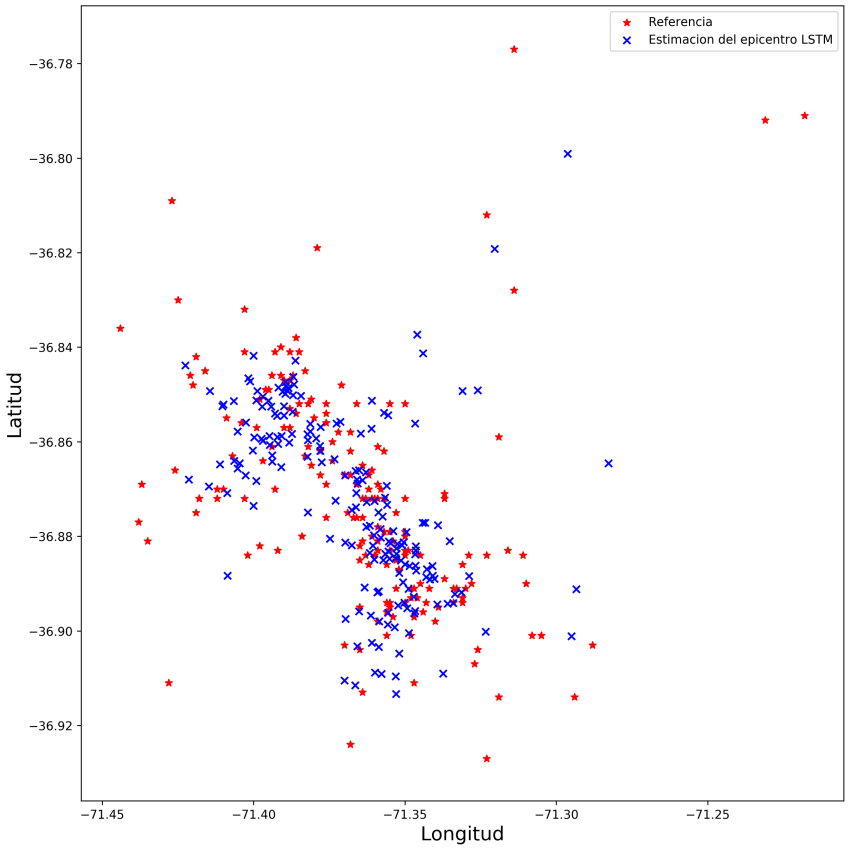

Figura 4.2: Localizaciones realizadas por el programa y las ubicaciones reales de los eventos sísmicos, modelo con la variable de incertidumbre agregada en la primera fila del input.

4.3.2. LSTM - Variable de incertidumbre multiplicada por el resultado de la STFT

El modelo con la variable de incertidumbre multiplicada muestra los mejores resultados entre modelos con igualdad de condiciones, llegando a un 52,15 % de predicciones con menos de 1 km de error y un error promedio de 1,673km. Esto se ve reflejado en la Figura 4.4.

Luego una representación gráfica de estas predicciones se observan en la Figura 4.5.

Al igual que en el modelo anterior, se realiza el cálculo del porcentaje de acierto en base al centroide de referencia calculado, junto al porcentaje de eventos de entrenamiento en esas distancias, el cual se observa en la Figura 4.6.

4.3.3. LSTM - Sin variable de incertidumbre

Es necesario comparar igualmente el desempeño del modelo de ambas posiciones de la variable de incertidumbre con uno que no la utiliza. El caso sin esta variable obtuvo un 45,16 % de aciertos, su error acumulado se presenta en la Figura 4.7 y el error promedio de localizaciones es de 1,671km.

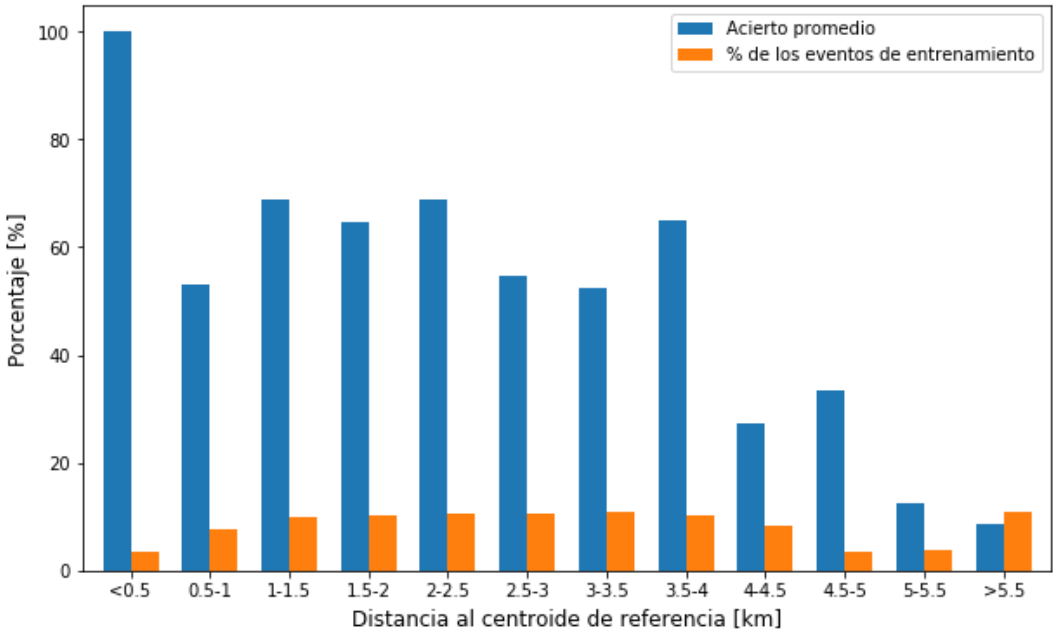

Figura 4.3: Acierto del modelo vs. la distancia del evento al centroide de referencia, modelo con la variable de incertidumbre agregada en la primera fila del input.

4.3.4. LSTM - Sin base de datos de los años 2015 a 2017 - Variable de incertidumbre multiplicada por el resultado de la STFT

El último modelo creado no considera la primera base de datos, correspondiente al período del 2015 al 2017, ya que esta es la que presenta la mayor intermitencia en la cantidad de sensores activados, además de contener sólo 6 estaciones de monitoreo como máximo. Por lo tanto, se realizó esta prueba con el objetivo de analizar el desempeño del modelo con una base de datos más robusta. Esta prueba se realizó utilizando la variable de incertidumbre multiplicada por la salida de la STFT.

Este modelo obtuvo resultados muy superiores al resto, logrando un 76,67 % de acierto y un error promedio de localizaciones de 1,052km. En la Figura 4.8 se muestra la frecuencia acumulada del error obtenido por este modelo comparado con el que multiplica la variable de incertidumbre por el resultado de la STFT.

4.3.5. Comparación de todos los modelos

La comparación del desempeño de todos los modelos se presenta en la Figura 4.9 y en la Tabla 4.3.

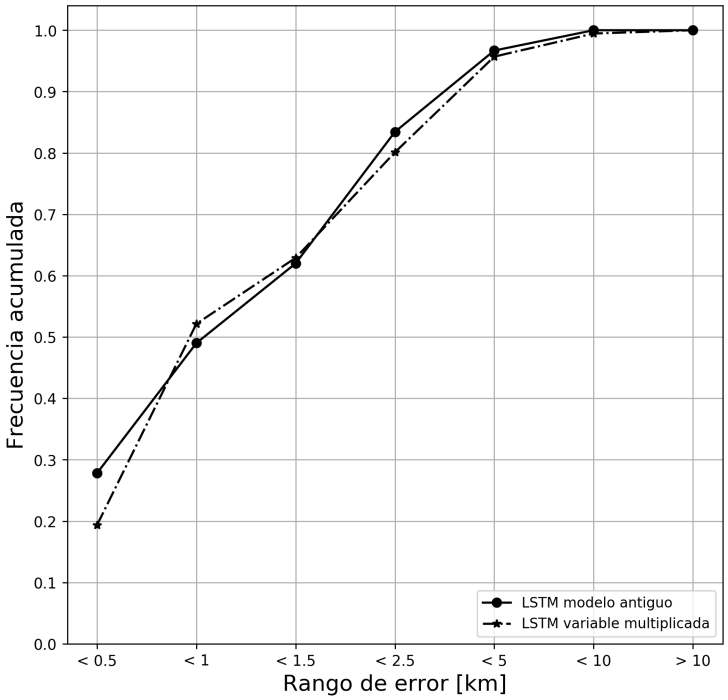

Figura 4.4: Frecuencia acumulada del error del modelo LSTM con la variable de incertidumbre multiplicada por el resultado de la STFT, se compara con el desempeño del modelo antiguo.

4.4. Discusión de resultados

4.4.1. Porcentaje de aciertos

Como puede verse en la Figura 4.9, dejando de lado el caso sin la base de datos 2015-2017, el modelo con el mejor desempeño es el que multiplica la variable de incertidumbre creada por el resultado de la STFT, logrando un 52,15 % de localizaciones con menos de $1km$ de error. Además, también logra el mejor resultado considerando el error promedio de las predicciones, que si bien no es la principal métrica a considerar, es una medida que refleja de gran forma la calidad de localizaciones que realiza el modelo.

La diferencia entre el modelo antiguo, que considera sólo 3 estaciones presentes en todos los eventos, con los implementados en este trabajo no contienen una gran diferencia, siendo apenas el mejor modelo actual apenas un 3,62 % superior. Sin embargo, la base de datos utilizada es completamente distinta, ya que se consideran todos los sensores disponibles agregando un *zero-padding* para las estaciones apagadas en un determinado evento, metodología que se descartó desde un principio en el modelo antiguo debido a que no superó el 30 % de aciertos. Esto logra un avance considerable en el aprovechamiento de la base de datos proporcionada por el OVDAS, ya que hasta ahora gran parte de esta se desperdiciaba por la gran fragmentación que presenta.

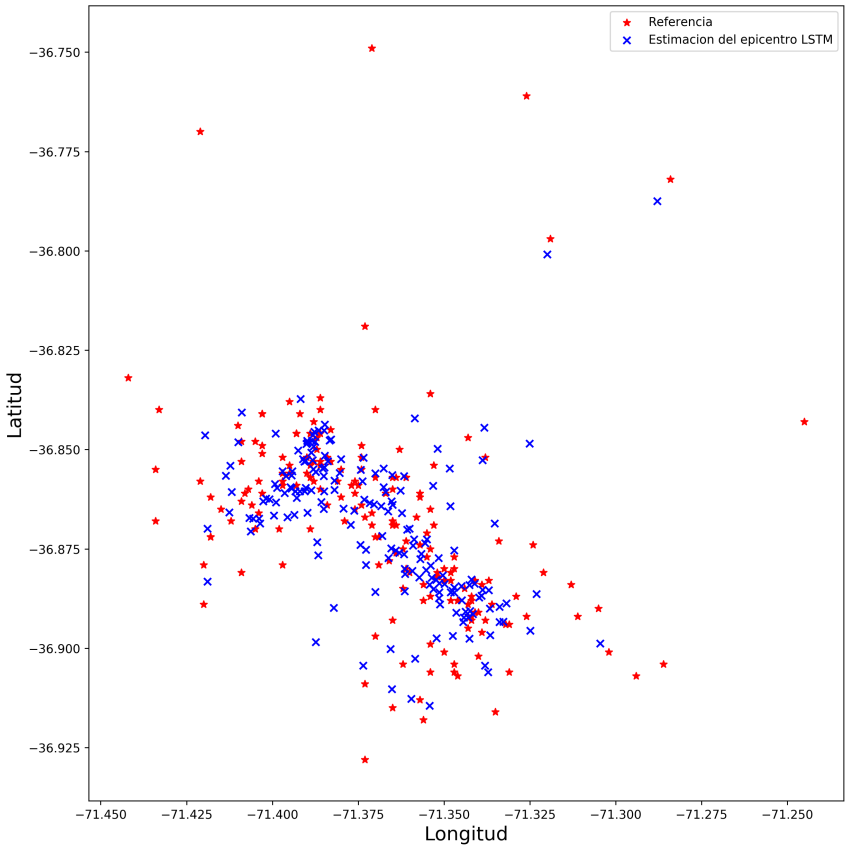

Figura 4.5: Localizaciones realizadas por el programa y las ubicaciones reales de los eventos sísmicos, modelo con la variable de incertidumbre multiplicada por el resultado de la STFT.

4.4.2. Desempeño a medida que aumenta la distancia al centroide

Como se observa en las Figuras 4.3 y 4.6, la precisión en la estimación del epicentro en ambos casos es de un 70 % promedio para eventos a menos de 2,5km del centroide, lo cual representa el aproximadamente el 40 % de los datos de entrenamiento, es decir, el 40 % de los datos con los que se entrena la red están en un área de 19,63km^2. Entre 4 y 5.5 km se tiene cerca de un 50 % de los datos de entrenamiento, lo que corresponde a un área de 75,40km^2, por lo que se tiene una densidad de eventos de entrenamiento mucho menor, sin embargo, el modelo no reduce su desempeño significativamente hasta superar los 4km de distancia, que es donde se reduce significativamente la densidad de eventos por kilómetro cuadrado. Cabe destacar que para cada entrenamiento se tienen conjuntos de entrenamiento distintos, ya que como se explica en la sección de Implementación, los conjuntos de *Train*, *Validation* y *Test* se generan distribuyendo aleatoriamente los eventos, es por esto que existen pequeñas diferencias en la densidad de eventos por kilómetro cuadrado de los dos modelos analizados. Estos resultados son consistentes con el hecho de que la capacidad de los métodos de aprendizaje automáticos dependen, en cierta medida, de la cantidad de ejemplos de entrenamiento.

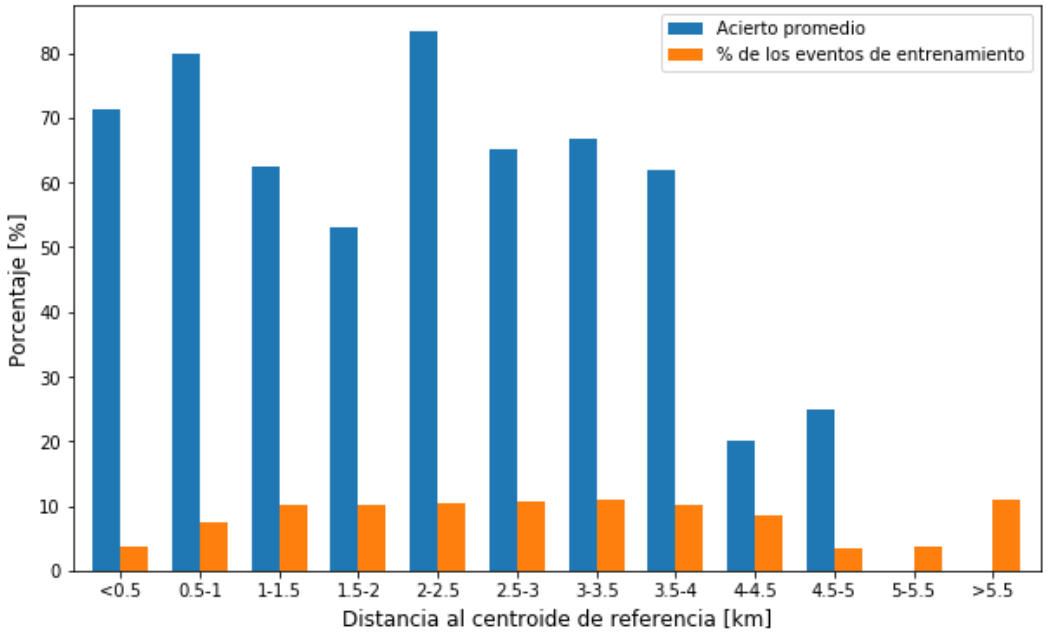

Figura 4.6: Acierto del modelo vs. la distancia del evento al centroide de referencia, modelo con la variable de incertidumbre multiplicada por el resultado de la STFT.

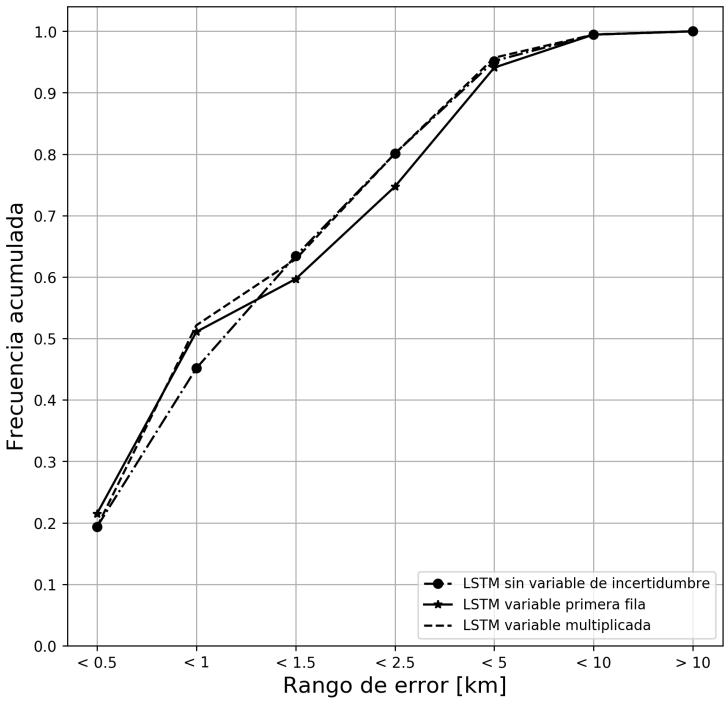

Figura 4.7: Frecuencia acumulada del error del modelo LSTM sin la variable de incertidumbre, se compara con el desempeño de los modelos que si utilizan esta variable.

4.4.3. Deep Learning vs. Picado automático+HYPOSAT

En la Figura 4.9 se observa la comparación de los modelos de Deep Learning con el desempeño de un sistema de picado automático de las ondas P y S más un software de localización

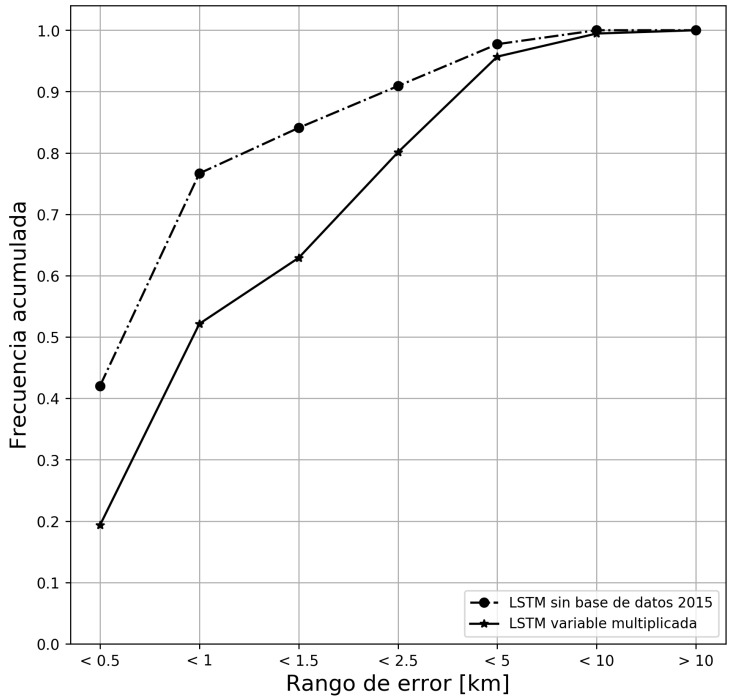

Figura 4.8: Frecuencia acumulada del error del modelo LSTM sin la base de datos 1, se compara con el desempeño del modelo que sí la incluye. En ambos casos se utiliza la variable de incertidumbre multiplicada por el resultado de la STFT.

llamada HYPOSAT (PhasePicker + HYPOSAT). Este sistema se utilizó con el modelo de propagación recientemente actualizado del volcán Chillán [7]. Como se puede apreciar en el gráfico, los modelos de Deep Learning logran una notoria superioridad en la calidad de localizaciones realizadas. De hecho, los resultados presentados del PhasePicker+HYPOSAT son en base a solo los casos donde el sistema entrega una localización, lo cual ocurre para un 17,64 % de los eventos.

El bajo porcentaje de acierto logrado por el PhasePicker+HYPOSAT se debe parcialmente a que el HYPOSAT está diseñado para localizar eventos sísmicos de origen tectónico, por lo que el área de estudio de este trabajo es muy pequeña. Este sistema fue diseñado para localizar distancias mucho mayores, lo que conlleva un margen de error aceptable mucho mayor, por lo que dada la métrica aplicada, es esperable un desempeño mucho peor por parte de este método automático. Además. el sistema de picado automático PhasePicker presenta evidentes errores en su detección de las ondas P y S, lo que deriva en una incapacidad del HYPOSAT de lograr localizar un epicentro.

4.4.4. Observación de la red neuronal y justificación MAE

En cuanto a la exploración de los hiperparámetros, el aprendizaje de la red se vio duramente afectado cuando se superaban las 6 capas LSTM, donde su aprendizaje llegaba hasta cierto punto sin lograr mejorar el modelo. Algo más radical ocurrió al implementar más de 10 capas LSTM, donde la red no aprendió nada, sin se capaz de realizar ninguna localización.

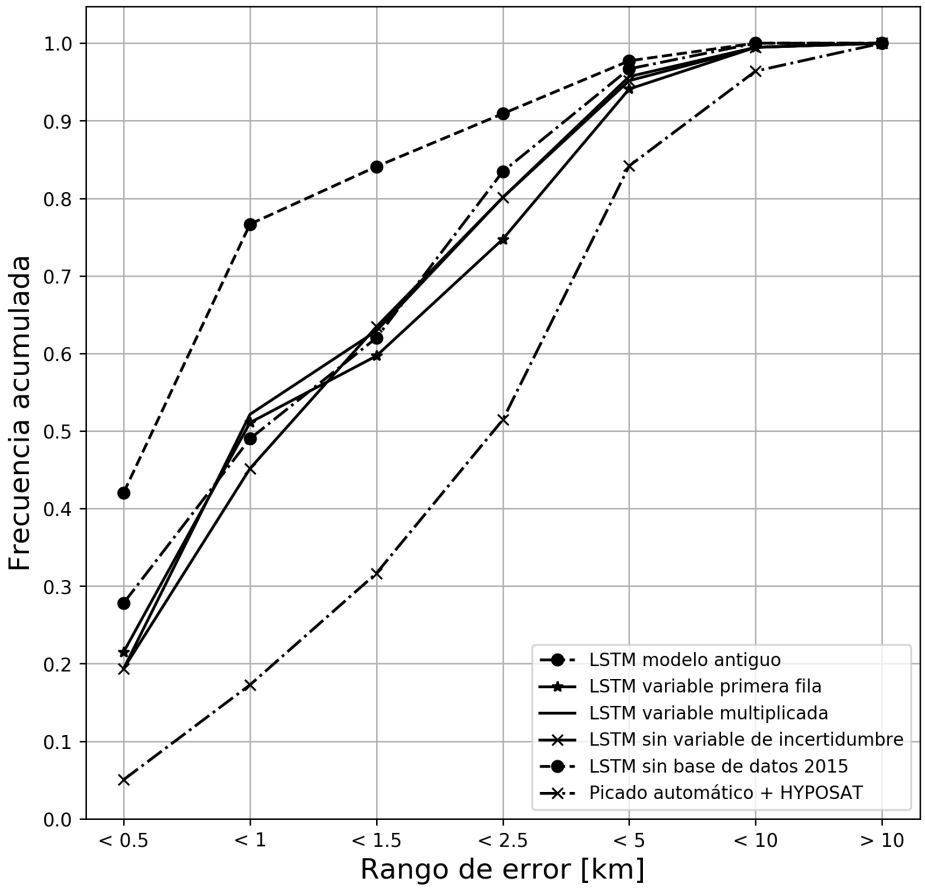

Figura 4.9: Comparación de la frecuencia acumulada del error de todos los modelos descritos. Incluye comparación con el método automático HYPOSAT.

En el marco teórico se menciona que mientras más capas tenga un modelo de Deep Learning, es capaz de lograr una mayor abstracción de los datos y así mejorar el desempeño de los modelos, sin embargo, lo anterior indica que una red muy compleja no es necesaria, ya que modelos con 3 a 5 capas LSTM logran un análisis óptimo para la relación de las señales de entrada con sus respectivas referencias.

Por otro lado, como se menciona en la Sección 3.2, la función de pérdidas utilizada es el *Mean Absolute Error* que cumple con el desempeño esperado según lo descrito en el marco teórico, ya que existe una gran concentración de eventos en un área pequeña llamada *enjambre* y los eventos fuera de este (*outliers*) consisten de una cantidad mucho menor. Debido a esto, el entrenamiento de la red presta una mayor atención a los eventos dentro del enjambre y tiende a no enfatizar tanto su aprendizaje en los llamados outliers. Esto se ve reflejado en el desempeño del modelo analizado anteriormente, donde el modelo muestra una clara mejora en su calidad de predicciones cuando la densidad de eventos de entrenamiento por kilómetro cuadrado es mayor.

Tabla 4.3: Resultados de todos los modelos implementados.

Modelo	**Porcentaje de acierto [%]**	**Error promedio [km]**	**Mejor error de predicción [km]**	**Peor error de predicción [km]**
Variable incertidumbre en la primera fila input	51.08	1.791	0.101	13.240
Variable incertidumbre multiplicada por el resultado de la STFT	52.15	1.673	0.052	11.523
Sin variable de incertidumbre	45.16	1.671	0.106	12.041
Sin base de datos 2015-2017	76.67	1.052	0.078	9.024
Modelo antiguo	48.50	1.762	-	-
Picado automático con HYPOSAT	17.27	3.108	-	-

4.4.5. Omisión base de datos 2015-2017

La base de datos correspondiente al período de los años 2015 a 2017 es sin dudas la que presenta la fragmentación más grande en cuanto a la presencia de los sensores, además de poseer la cantidad más baja de estaciones de monitoreo. Además, actualmente se tiene una red de más de 10 estaciones en el volcán, por lo tanto, un entrenamiento de la red realizado con una mayor cantidad de estaciones puede lograr ser más provechosa que incorporar menos estaciones y más fragmentación de estas.

Finalmente, se demostró en los resultados que tener más estaciones con mayor presencia de registros deriva en un desempeño muy superior, donde el *trade-off* de bajar la cantidad de eventos de entrenamiento pero mejorar la calidad de estos resulta positivamente para el aprendizaje del modelo. No se está diciendo que la base de datos 2015-2017 está obsoleta, si no que para el modelo diseñado actualmente podría ser considerada eludible.

4.4.6. Limitaciones del método

Este modelo no necesita de un picado de las ondas P y S, sin embargo, no es un localizador 100 % automático desde la detección hasta la localización, ya que necesita de una señal en la que se conoce a priori existe un evento sísmico en esta y que además el inicio de este esta centrado en base a una estación de referencia.

Si bien la normalización de las coordenadas de referencia ayuda al aprendizaje de la red LSTM, esta también lo limita a la zona geográfica acotada del Volcán Chillán, donde utilizar las coordenadas de [$Longitud, Latitud$] podría ampliar los limites geográficos del modelo. Sin embargo, este se diseñó para eventos volcánicos, los cuales difícilmente se escapan de esta

zona del propio volcán, por lo que podría llegar a ser redundante.

Por otro lado, el modelo aprende y predice sesgado en parte por la geografía del volcán, donde reconoce zonas más probables y estima los epicentros en base a esto. Es por esto que lo más probable es que un entrenamiento de la red LSTM utilizando señales del Volcán Chillán no se desempeñe con la misma precisión en cualquier otro volcán. Sin embargo, es posible que la arquitectura y los parámetros si sean generalizables para su aplicación en otros volcanes siempre y cuando sean entrenados con eventos propios del volcán a estudiar.

La cantidad de estaciones, variable que está fuera del alcance de este trabajo, es relativamente baja. Si bien 3 estaciones es el mínimo para lograr una localización, esta conlleva una incertidumbre mayor que una localización realizada con más sensores. En modelos de Deep Learning dedicados a la localización de eventos sísmicos tectónicos se suelen utilizar redes de más de 20 estaciones de monitoreo, por lo que los resultados obtenidos en este trabajo tienen un gran mérito en cuanto a su precisión vs. la cantidad de sensores.

Capítulo 5

Conclusiones

Se logró el objetivo principal del trabajo que consistió en mejorar la precisión del modelo creado con anterioridad, utilizando la base de datos completa, superando por un 3,65 % el porcentaje de acierto del mejor modelo que incluye las 3 bases de datos proporcionadas por el OVDAS. En esta misma línea, el segundo mejor modelo supera por un 2,58 % al primer modelo creado. Además, utilizando sólo las bases de datos del 2019 y 2020 se logró una configuración que supera por 24,52 % al mejor método que incluye las 3 bases de datos. Todos estos resultados utilizando la variable de incertidumbre definida.

Si bien no se conoce el proceso interno que utiliza la red de Deep Learning, se demostró que el modelo es capaz de aprovechar múltiples características de la información de entrenamiento, logrando ser capaz de resolver el problema considerablemente mejor que los métodos automáticos (*PhasePicker+HYPOSAT*) utilizados hoy en día por el OVDAS.

En cuanto a la variable de incertidumbre creada e implementada, si bien esta no se propagó por el entrenamiento de la red, se demostró que esta afectó positivamente el aprendizaje del modelo de Deep learning para ambas utilizaciones de esta implementadas. Este resultado se podía esperar debido a que esta variable incluye información que afecta tanto al análisis de las señales como el incremento de precisión que conlleva aumentar la cantidad de sensores para la localización.

Una de las grandes ventajas del método implementado es el no necesitar un picado de las ondas P y S, ya que la correcta detección de estas es en cierta medida una de las tareas más difíciles y que sigue siendo monitoreada por expertos.

Un punto de gran interés es que no existe estudio alguno en la literatura que permita comparar este estudio con otro modelo que utilice Deep Learning, ya que existen (no abundantemente) modelos que trabajan con sismos de origen tectónico, pero ninguno con eventos volcánicos. Es por esto que la única comparación o *baseline* disponible son el modelo creado previamente y los métodos automáticos que se utilizan hoy en día. Dicho esto, este trabajo crea una nueva línea de exploración sísmica volcánica dentro del mundo de la Inteligencia Artificial, que de seguir progresando y perfeccionándose, podría beneficiar enormemente a un país tan volcánicamente activo como lo es Chile.

Como trabajo futuro, la *Data Augmentation* podría generar una mejora sustancial en la precisión de la red LSTM, ya que como se menciona en 4.4.2, secciones con una mayor cantidad de eventos por kilómetro cuadrado tienen a obtener predicciones con un mayor porcentaje de aciertos. Es por esto que la *Data Augmentation* tiene el potencial de generar una distribución más homogénea de los eventos en el área geográfica de estudio.

Finalmente, también se propone explorar con mayor exhaustividad las capacidades del modelo utilizando sólo las bases de datos del 2019 y 2020, los excelentes resultados obtenidos ameritan un análisis más profundo de este. Además, se plantea evaluar la posibilidad de la incorporación de una nueva base de datos, como lo podría ser una que incorpore eventos registrados del 2021, para así analizar la estructura de las nuevas estaciones de monitoreo disponibles y su esperado impacto positivo en el aprendizaje de la red LSTM.

Bibliografía

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] R Aguilar. Red neuronal de topología flexible. In *VI Congreso Nacional de Ciencias de la Computación, La Paz, Bolivia*, 1999.

[3] Isaac Álvarez, Luz García, Sonia Mota, Guillermo Cortés, Carmen Benítez, and Ángel De la Torre. An automatic p-phase picking algorithm based on adaptive multiband processing. *IEEE Geoscience and remote sensing letters*, 10(6):1488–1492, 2013.

[4] Kartik Audhkhasi, Andrew Rosenberg, Abhinav Sethy, Bhuvana Ramabhadran, and Brian Kingsbury. End-to-end asr-free keyword search from speech. *IEEE Journal of Selected Topics in Signal Processing*, 11(8):1351–1359, 2017.

[5] Christopher M Bishop et al. *Neural networks for pattern recognition*. Oxford university press, 1995.

[6] Zoran S Bojkovic, Bojan M Bakmaz, and Miodrag R Bakmaz. Hamming window to the digital world. *Proceedings of the IEEE*, 105(6):1185–1190, 2017.

[7] C Cardona, F Gil-Cruz, L Franco-Marín, J San Martín, O Valderrama, J Lazo, C Cartes, S Morales, E Hernández, J Quijada, et al. Volcanic activity accompanying the emplacement of dacitic lava domes and effusion of lava flows at nevados de chillán volcanic complex–chilean andes (2012 to 2020). *Journal of Volcanology and Geothermal Research*, 420:107409, 2021.

[8] Menzie Chinn. Them's Fightin' Words: Futures, Mean Squared Error, Mean Absolute Error — Econbrowser, 07 2019.

[9] Francois Chollet et al. Keras, 2015.

[10] Bernard Chouet. Volcano seismology. *Pure and applied geophysics*, 160(3):739–788, 2003.

[11] William T Cochran, James W Cooley, David L Favin, Howard D Helms, Reginald A Kaenel, William W Lang, George C Maling, David E Nelson, Charles M Rader, and Peter D Welch. What is the fast fourier transform? *Proceedings of the IEEE*, 55(10):1664–1674, 1967.

[12] Isis Bonet Cruz, Sain Salazar Martínez, Abdel Rodríguez Abed, Ricardo Grau Ábalo, and Maria Matilde García Lorenzo. Redes neuronales recurrentes para el análisis de secuencias. *Revista Cubana de Ciencias Informáticas*, 1(4):48–57, 2007.

[13] DeepAI. Frequency Domain, 06 2020.

[14] Sanket Doshi. Various Optimization Algorithms For Training Neural Network, 12 2021.

[15] Rajmil Fischman. The phase vocoder: theory and practice. *Organised Sound*, 2:127 – 145, 08 1997.

[16] Luz García, Gerardo Alguacil, Manuel Titos, Ornella Cocina, Angel De la Torre, and Carmen Benítez. Automatic s-phase picking for volcano-tectonic earthquakes using spectral dissimilarity analysis. *IEEE Geoscience and Remote Sensing Letters*, 17(5):874–878, 2019.

[17] Paolo Gasparini, Roberto Scarpa, and Keiiti Aki. *Volcanic seismology*, volume 3. Springer Science & Business Media, 2012.

[18] Tian Guo, Tao Lin, and Yao Lu. An interpretable lstm neural network for autoregressive exogenous model. *arXiv preprint arXiv:1804.05251*, 2018.

[19] Francisco Herrera. Big data: Preprocesamiento y calidad de datos. *novática*, 237:17, 2016.

[20] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[21] Icarito. Chile país de volcanes. *www.icarito.com*, 2010.

[22] Infaimon. Ventajas del Machine Learning y Deep Learning para la evolución de la visión artificial, 11 2021.

[23] La Tercera Iván. Chile: El país con la segunda cadena volcánica más grande del mundo. *www.latercera.com*, 2008.

[24] Don H Johnson. Signal-to-noise ratio. *Scholarpedia*, 1(12):2088, 2006.

[25] Michael I Jordan and Tom M Mitchell. Machine learning: Trends, perspectives, and prospects. *Science*, 349(6245):255–260, 2015.

[26] Rishikesan Kamaleswaran, Ruhi Mahajan, and Oguz Akbilgic. A robust deep convolutional neural network for the classification of abnormal cardiac rhythm using single lead electrocardiograms of variable length. *Physiological measurement*, 39(3):035006, 2018.

[27] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[28] Marius Kriegerowski, Gesa M Petersen, Hannes Vasyura-Bathke, and Matthias Ohrnberger. A deep convolutional neural network for localization of clustered earthquakes based on multistation full waveforms. *Seismological Research Letters*, 90(2A):510–516, 2019.

[29] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[30] Bo Li, Shuo-yiin Chang, Tara N Sainath, Ruoming Pang, Yanzhang He, Trevor Strohman, and Yonghui Wu. Towards fast and accurate streaming end-to-end asr. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6069–6073. IEEE, 2020.

[31] Liu Liu, Rujing Wang, Chengjun Xie, Po Yang, Fangyuan Wang, Sud Sudirman, and Wancai Liu. Pestnet: An end-to-end deep learning approach for large-scale multi-class pest detection and classification. *IEEE Access*, 7:45301–45312, 2019.

[32] Cinna Lomnitz. A fast epicenter location program. *Bulletin of the Seismological Society of America*, 67(2):425–431, 1977.

[33] Antonio Loquercio, Mattia Segu, and Davide Scaramuzza. A general framework for uncertainty estimation in deep learning. *IEEE Robotics and Automation Letters*, 5(2):3153–3160, 2020.

[34] David Marr. Artificial intelligence—a personal view. *Artificial Intelligence*, 9(1):37–48, 1977.

[35] John McCarthy. What is artificial intelligence? 2007.

[36] Stephen R McNutt. Volcanic seismology. *Annu. Rev. Earth Planet. Sci.*, 32:461–491, 2005.

[37] Larry Medsker and Lakhmi C Jain. *Recurrent neural networks: design and applications*. CRC press, 1999.

[38] Stephen G Mosher and P Audet. Automatic detection and location of seismic events from time-delay projection mapping and neural network classification. *Journal of Geophysical Research: Solid Earth*, 125(10):e2020JB019426, 2020.

[39] J Naranjo, Jennifer S Gilbert, and RSJ Sparks. Geologia del complejo volcanico, nevados de chillan. *Revista Geologica de Chile*, 114:114, 2008.

[40] S Nawab, T Quatieri, and Jae Lim. Signal reconstruction from short-time fourier transform magnitude. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 31(4):986–998, 1983.

[41] Michael A Nielsen. *Neural networks and deep learning*, volume 25. Determination press San Francisco, CA, 2015.

[42] José Novoa, Josué Fredes, Víctor Poblete, and Néstor Becerra Yoma. Uncertainty weighting and propagation in dnn–hmm-based speech recognition. *Computer Speech & Language*, 47:30–46, 2018.

[43] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pages 1310–1318. PMLR, 2013.

[44] Thibaut Perol, Michaël Gharbi, and Marine Denolle. Convolutional neural network for earthquake detection and location. *Science Advances*, 4(2):e1700578, 2018.

[45] Michael Portnoff. Time-frequency representation of digital signals and systems based on short-time fourier analysis. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 28(1):55–69, 1980.

[46] Gobierno de España. Proyecto Biosfera. Ministerio de Educación, Deporte y Cultura. Movimientos sísmicos. *www.recursostic.educacion.es/*.

[47] Francisco Javier Gómez Quesada, Miguel Angel Fernández Graciani, María Teresa López Bonal, and María Alonso Díaz-Mata. Aprendizaje con redes neuronales artificiales. *Ensayos: Revista De La Facultad De Educación De Albacete*, (9):169–180, 1994.

[48] Wilber Roberto Ramos Lovón. End-to-end deep learning para el reconocimiento automático del habla. 2019.

[49] Diana C Roman and Katharine V Cashman. The origin of volcano-tectonic earthquake swarms. *Geology*, 34(6):457–460, 2006.

[50] Zachary E Ross and Yehuda Ben-Zion. Automatic picking of direct p, s seismic phases and fault zone head waves. *Geophysical Journal International*, 199(1):368–381, 2014.

[51] Juan Ignacio Sabbione. *Algoritmos matemáticos y computacionales para la detección automática de señales sísmicas*. PhD thesis, Universidad Nacional de La Plata, 2012.

[52] MS Safizadeh, AA Lakis, and M Thomas. Using short-time fourier transform in machinery diagnosis. *Proceedings of WSEAS (Brazil)*, pages 494–200, 2005.

[53] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

[54] Seb. An Introduction to Neural Network Loss Functions, 09 2021.

[55] SERNAGEOMIN. Actividad sísmica en volcanes. *www.sernageomin.cl*.

[56] Yangyang Shi, Mei-Yuh Hwang, and Xin Lei. End-to-end speech recognition using a high rank lstm-ctc based model. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7080–7084. IEEE, 2019.

[57] Ralf C Staudemeyer and Eric Rothstein Morris. Understanding lstm–a tutorial into long short-term memory recurrent neural networks. *arXiv preprint arXiv:1909.09586*, 2019.

[58] S. Stein and M Wysession. *An introduction to Seismology, Earthquakes and Earth Structure*. Oxford: Blackwell Science, 2003.

[59] Oleksii Trekhleb. Playing with Discrete Fourier Transform Algorithm in JavaScript, 02 2021.

[60] Pablo Estevez V. Redes neuronales artificiales. *Inteligencia Computacional, Facultad de Ciencias Físicas y Matemáticas, Universidad de Chile*, 2019.

[61] Randall White and Wendy McCausland. Volcano-tectonic earthquakes: A new tool for estimating intrusive volumes and forecasting eruptions. *Journal of Volcanology and Geothermal Research*, 309:139–155, 2016.

[62] Serdar Yegulalp. What is tensorflow? the machine learning library explained. *InfoWorld*.

[63] Néstor Becerra Yoma and Miguel Villar. Speaker verification in noise using a stochastic version of the weighted viterbi algorithm. *IEEE Transactions on Speech and Audio Processing*, 10(3):158–166, 2002.

[64] Kai Yu, Lei Jia, Yuqiang Chen, and Wei Xu. Deep learning: yesterday, today, and tomorrow. *Journal of computer Research and Development*, 50(9):1799, 2013.

[65] Mohammad Zeineldeen, Albert Zeyer, Ralf Schlüter, and Hermann Ney. Layer-normalized lstm for hybrid-hmm and end-to-end asr. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7679–7683. IEEE, 2020.

[66] Weiqiang Zhu and Gregory C Beroza. Phasenet: a deep-neural-network-based seismic arrival-time picking method. *Geophysical Journal International*, 216(1):261–273, 2019.