UNIVERSIDAD NACIONAL DE INGENIERÍA

FACULTAD DE INGENIERÍA CIVIL

TESIS

ESTIMACIÓN DE PARÁMETROS HIPOCENTRALES CON BASE EN REDES NEURONALES GRÁFICAS

PARA OBTENER EL TÍTULO PROFESIONAL DE INGENIERO CIVIL

ELABORADO POR:

ANGEL OLIVER MATOS CHUQUIURI

ASESOR:

Dra. DIANA LUCÍA CALDERÓN CAHUANA

LIMA - PERÚ

2023

Matos Chuquiuri, Angel Oliver
amatosc@uni.pe
+51975528376 -2504536

DEDICATORIA

A mi mamá y papá por su infinito amor.

A mis hermanos, juntos somos el mejor equipo.

AGRADECIMIENTOS

Agradezco al proyecto “Mejoramiento y Ampliación de los Servicios del Sistema Nacional de Ciencia, Tecnología e Innovación Tecnológica” [contrato 038-2019] el cual es liderado por el Consejo Nacional de Ciencia, Tecnología e Innovación Tecnológica (Concytec) a través del Fondo Nacional de Desarrollo Científico, Tecnológico y de Innovación Tecnológica (Fondecyt) y el Banco Mundial, por financiar la presente tesis.

Agradezco enormemente a mis asesores Dra. Diana Calderon, Dr. Luis Moya y Dr. Carlos Gonzáles por su gran predisposición, continuo apoyo, paciencia y dedicación en la realización de mi tesis.

Agradezco al Prof. Shunichi Koshimura y al Dr. Erick Mas del International Research Institute of Disaster Science (IRIDeS) de la Universidad de Tohoku por permitirme usar su infraestructura en la realización de mi tesis. De igual manera, agradecer al Dr. Bruno Adriano por sus aportes y recomendaciones sobre mi tesis.

Agradecimiento especial a mi familia por su incondicional apoyo y comprensión en cada uno de mis proyectos, así como en esta tesis.

ÍNDICE

RESUMEN **3**

ABSTRACT **4**

PRÓLOGO **5**

LISTA DE TABLAS **6**

LISTA DE FIGURAS **7**

LISTA DE SÍMBOLOS Y SIGLAS **8**

CAPÍTULO I: INTRODUCCIÓN **10**

1.1 ANTECEDENTES 11

1.2 DESCRIPCIÓN DEL PROBLEMA DE INVESTIGACIÓN 13

1.3 OBJETIVOS DEL ESTUDIO 15

1.3.1 Objetivo General 15

1.3.2 Objetivos Específicos 15

1.4 HIPÓTESIS DEL ESTUDIO 15

1.4.1 Hipótesis General 15

1.4.2 Hipótesis Específicas 16

1.5 METODOLOGÍA 16

CAPÍTULO II: FUNDAMENTO TEÓRICO **17**

2.1 ONDAS SÍSMICAS 17

2.2 PARÁMETROS HIPOCENTRALES 19

2.2.1 Localización de la fuente sísmica 19

2.2.2 Magnitud del evento 21

2.3 APRENDIZAJE AUTOMÁTICO 23

2.3.1 Aprendizaje Supervisado 24

2.3.2 Algoritmos de regresión 25

2.4 REDES NEURONALES 25

2.4.1 ¿Cómo aprende una red neuronal? 26

2.4.2 Aprendizaje Profundo 27

2.4.3 Tipos de redes neuronales 28

2.4.4 Aplicaciones en la sismología 30

2.5 TEORÍA DE GRAFOS 32

2.5.1 Grafos 32
2.5.2 Matriz de Adyacencia 32
2.5.3 Tipos de Grafos 33
2.5.4 Feature Information 33
2.5.5 Tipos de problemas en grafos 34
2.6 REDES NEURONALES GRÁFICAS 35
2.6.1 *Message Passing* 35
2.6.2 *Graph Convolutional Networks* 37
2.6.3 *Graph Pooling* 38
2.6.4 *Differential Pooling* 38

CAPÍTULO III: CASO DE ESTUDIO **39**
3.1 BASE DE DATOS SÍSMICA 39
3.2 PREPROCESAMIENTO DE LA DATA SÍSMICA 39
3.3 ARQUITECTURA 44
3.4 ENTRENAMIENTO Y EVALUACIÓN 47

CAPÍTULO IV: DISCUSIÓN DE RESULTADOS **54**
4.1 ESTIMACIÓN DE LA MAGNITUD 54
4.2 ESTIMACIÓN DE LAS COORDENADAS EPICENTRALES 55
4.3 ESTIMACIÓN DE LA PROFUNDIDAD HIPOCENTRAL 56

CONCLUSIONES **58**

RECOMENDACIONES **60**

REFERENCIAS BIBLIOGRÁFICAS **61**

ANEXOS **65**

RESUMEN

El presente trabajo de investigación expone una metodología alternativa para la estimación de los parámetros hipocentrales (latitud, longitud, profundidad y magnitud) de una fuente sísmica, utilizando las formas de onda registradas en las estaciones de una red sísmica. Este novedoso método basado en Inteligencia Artificial permite obtener los parámetros de la fuente de manera automática y precisa, tomando la información bruta de los sismogramas y sin la intervención de un especialista.

En este trabajo se utilizó el novedoso marco de las Redes Neuronales Gráficas que, a diferencia de otros algoritmos de Aprendizaje Profundo, puede trabajar sobre datos no estructurados o grafos. Por lo tanto, se calibró un modelo basado en Redes Neuronales Gráficas que aprovecha explícitamente la estructura compleja de una red sísmica para la tarea de estimación de parámetros hipocentrales de un terremoto. En otras palabras, el modelo recibe los sismogramas registrados en cada estación de un terremoto organizados en un grafo como dato de entrada, donde finalmente el modelo devuelve los parámetros hipocentrales del terremoto.

El modelo se entrenó con eventos recuperados de la Red Sísmica del Sur de California, debido a la disponibilidad y accesibilidad de la información. Una ventaja importante de representar la información en un grafo es que es posible almacenar los sismogramas en las 3 direcciones y la información geográfica de las estaciones en los nodos, asegurando información que puede ser determinante para el aprendizaje del modelo.

Los experimentos muestran que usando una arquitectura simple basada en Redes Neuronales Gráficas, podemos obtener algunas mejoras con respecto a los resultados de estudios recientes basados en Aprendizaje Profundo. Para aplicaciones reales, los resultados pueden ser suficientemente precisos y rápidos para reportes sísmicos preliminares.

ABSTRACT

This research work exposes an alternative methodology for estimating the hypocentral parameters (latitude, longitude, depth and magnitude) of a seismic source, using the waveforms recorded in the stations of a seismic network. This novel method based on Artificial Intelligence allows obtaining the parameters of the source automatically and precisely, taking the raw information from the seismograms and without the intervention of a specialist.

In this work, the novel framework of Graph Neural Networks was used, which, unlike other Deep Learning algorithms, can work on unstructured data or graphs. Therefore, a Graph Neural Network model was calibrated that explicitly leverages the complex structure of a seismic network for the task of estimating hypocentral parameters of an earthquake. In other words, the model receives the seismograms recorded at each station of an earthquake organized in a graph as input data, where finally the model returns the hypocentral parameters of the earthquake.

The model was trained with events retrieved from the Southern California Seismic Network, due to the availability and accessibility of the information. An important advantage of representing the information in a graph is that it is possible to store the seismograms in the 3 directions and the geographical information of the stations in the nodes, ensuring information that can be decisive for learning the model.

The experiments show that using a simple architecture based on Graph Neural Networks, we can obtain some improvements with respect to the results of recent studies based on Deep Learning. For real applications, the results can be accurate and fast enough for preliminary seismic reports.

PRÓLOGO

El presente trabajo de investigación explora una metodología distinta a la tradicional en la estimación de los parámetros hipocentrales relacionados a un evento sísmico. Para este fin, se ha empleado un proceso con base en la inteligencia artificial, específicamente en Redes Neuronales Gráficas.

El trabajo evidencia que se pueden encontrar soluciones novedosas desde el campo de la Inteligencia artificial y aprendizaje automático en el campo de la Sismología y aplicaciones en la Ingeniería Civil. Estas soluciones, que provienen de buscar y analizar patrones en miles de datos, pueden asistir al hombre y automatizar procesos que pueden ser muy complejos o que pueden tomar mucho tiempo.

La presente metodología, permite caracterizar una fuente sísmica con tan solo recolectar los sismogramas provenientes de las estaciones de una red sísmica. Estos se entregan a un modelo entrenado, que finalmente calculará los parámetros hipocentrales de un evento sísmico (terremoto).

Parte del contenido de la presente tesis ha sido aceptada y expuesta en dos Congresos Internacionales, el SSA 2022 Annual Meeting en Washington y AOGS 2022 Annual Meeting llevado a cabo de manera virtual; en ambos casos se utilizaron datos de redes sísmicas de Estados Unidos, que a comparación de la peruana tiene una mayor densidad de estaciones, para la estimación de los parámetros hipocentrales.

Con este trabajo, se espera motivar nuevas investigaciones sobre metodologías alternativas que usen Inteligencia Artificial en el campo de la Ingeniería Civil, y que puedan ser aplicadas para lograr estimaciones de parámetros técnicos de importancia.

LISTA DE TABLAS

Tabla N° 3.1 Error absoluto medio (MAE) para cada parámetro hipocentral 50

LISTA DE FIGURAS

Figura N° 2.1 Ondas de cuerpo 17
Figura N° 2.2 Ondas superficiales 18
Figura N° 2.3 Diagrama de Wadati 20
Figura N° 2.4 Método de los círculos 20
Figura N° 2.5 Determinación de la magnitud de Richter 22
Figura N° 2.6 Saturación de ondas de cuerpo y superficie 23
Figura N° 2.7 Aprendizaje Supervisado 25
Figura N° 2.8 Regresión 26
Figura N° 2.9 Red Neuronal Biológica 26
Figura N° 2.10 Red Neuronal Artificial 27
Figura N° 2.11 Convolución de un filtro 3x3 sobre una imágen 2D 29
Figura N° 2.12 Red Convolucional para la clasificación de imágenes . . . 30
Figura N° 2.13 Grafo 32
Figura N° 2.14 *Directed Graph* y *Undirected Graph* 33
Figura N° 2.15 *Weighted Graph* y su matriz de adyacencia 33
Figura N° 2.16 *Graph Regression* 34
Figura N° 2.17 *Differential Pooling* 38
Figura N° 3.1 Área de estudio 40
Figura N° 3.2 Grafo de 50 nodos y sus enlaces 42
Figura N° 3.3 Distribución de los parámetros hipocentrales 43
Figura N° 3.4 *Data Augmentation* 43
Figura N° 3.5 *Graph Augmentation* 44
Figura N° 3.6 Arquitectura del modelo 45
Figura N° 3.7 Código de la Arquitectura 46
Figura N° 3.8 Selección del modelo 48
Figura N° 3.9 *Loop* de entrenamiento 49
Figura N° 3.10 Historia del error 50
Figura N° 3.11 Curva de densidad del error de predicción 51
Figura N° 3.12 Valor predicho vs Valor del catálogo 52
Figura N° 3.13 Distancia residual epicentral 53
Figura N° 4.1 Fallas en el área de estudio 55
Figura N° 4.2 Curva de densidad del error en las coordenadas epicentrales 56

LISTA DE SÍMBOLOS Y SIGLAS

SÍMBOLOS

σ	:	Función de activación no lineal
b	:	Sesgo o *bias*
G	:	Grafo o *graph*
V	:	Nodos o *nodes or vertices*
E	:	Enlaces o *edges*
A	:	Matriz de adyacencia o *adjacency matrix*
u, v	:	Nodos u y v
$\mathcal{N}(u)$	:	Vecindario del nodo u
x_u	:	*Node Features* del nodo u
X	:	*Feature Matrix* de todo el grafo
h_u	:	*Hidden embedding* del nodo u
H	:	*Hidden embedding* de todo el grafo
z_u	:	*Node embedding* del nodo u
Z	:	*Node embedding* de todo el grafo
W	:	Matriz de pesos
$m_{\mathcal{N}(u)}$	:	Mensaje de los nodos del vecindario del nodo u

SIGLAS

ML	:	*Machine Learning*
MLP	:	*MultiLayer Perceptron*
CNN	:	*Convolutional Neural Network*
RNN	:	*Recurrent Neural Network*
GNN	:	*Graph Neural Network*
GCN	:	*Graph Convolutional Network*
MAE	:	*Mean Absolute Error*
SCSN	:	*South California Seismic Network*

CAPÍTULO I: INTRODUCCIÓN

La realidad del Perú es que es un país sísmico y nos encontramos a la espera de un sismo de gran magnitud. Entonces, nos conviene estar preparados ante eventos como terremotos y tsunamis.

La caracterización de la fuente sísmica es una tarea principal en la sismología que implica la estimación de los parámetros hipocentrales (coordenadas epicentrales, profundidad hipocentral y magnitud del evento sísmico). Particularmente, para propósitos de alerta temprana, respuesta de emergencia y difusión oportuna de información, es necesario estimar los parámetros hipocentrales de la fuente sísmica, preferiblemente de manera rápida y sin la intervención de un analista. Una herramienta computacional que satisface estas necesidades es *Machine Learning*, lo que la convierte en un potencial candidato para abordar el desafío de la caracterización rápida de fuentes sísmicas. Varios investigadores han aplicado estos algoritmos al análisis de señales sísmicas entre los cuales destacan las redes neuronales debido a la capacidad de trabajar sobre registros sísmicos con poco o ningún preprocesamiento.

Las redes neuronales convolucionales (CNN, por sus siglas en inglés) son las más empleadas en la tarea de determinar los parámetros hipocentrales de la fuente sísmica. Los modelos basados en CNN trabajan bien con arreglos (como las imágenes) de datos. Sin embargo, si se quiere representar la data de una red sísmica como un arreglo de sismogramas, debemos predefinir un orden en las estaciones y un tamaño constante del arreglo. Además, dichos modelos no hacen uso de la información geográfica de las estaciones de la red sísmica, por lo que existe un potencial no utilizado, que puede significar una mejora en el rendimiento de los modelos.

Otro enfoque representó la red sísmica como un grafo sin enlaces, de esta manera, el modelo fue capaz de incorporar la información espacial de las estaciones. Del análisis realizado, se concluyó que la incorporación de las coordenadas geográficas de las estaciones fue determinante para obtener mejores predicciones en la determinación del epicentro del terremoto que un modelo que prescinde de la información espacial. Este método, al ser un grafo sin enlaces, considera que no es importante compartir la información entre sus nodos o estaciones. Entonces, queda la pregunta si el agregar enlaces puede ser beneficioso para mejorar el rendimiento del modelo. Es decir, confirmar si al modelar la red sísmica como un grafo se aprovecha mejor la estructura compleja y características propias de la misma. Afortunadamente, avances importantes para el análisis de grafos han ocurrido en los últimos años, por ejemplo, *Graph Neural Networks*.

Debido a los puntos tratados, se hace necesario continuar con la investigación y por lo tanto, en esta tesis se propone emplear *Graph Neural Networks* para la estimación de parámetros hipocentrales de un evento sísmico.

La presente tesis, se divide en cinco capítulos, resumidos a continuación: El Capítulo I, Introducción, presenta los antecedentes, explica la problemática, realiza la justificación e hipotesis, define los objetivos y establece la metodología de trabajo a seguir. El Capítulo II, Fundamento Teórico, aborda todos aquellos temas que son necesarios definir para comprender ampliamente lo desarrollado en la tesis, como conceptos, algoritmos, metodologías, entre otros. El Capítulo III, presenta paso a paso el desarrollo del modelo de *Graph Neural Networks* para la estimación de parámetros hipocentrales aplicado a un caso de estudio, desde el preprocesamiento de la información sísmica, pasando por la configuración de su arquitectura, hasta la evaluación del modelo final. Finalmente el Capítulo IV, presenta la discusión acerca de los resultados obtenidos, priorizando las conclusiones y recomendaciones más relevantes del trabajo.

1.1 ANTECEDENTES

En los últimos años, el aumento de la disponibilidad de grandes cantidades de información y poder computacional ha hecho posible encontrar soluciones alternativas y novedosas a problemas complejos, inspiradas en procesos naturales, dentro del área de la Inteligencia Artificial, tales como, algoritmos de *Machine Learning*. Dichos algoritmos se aplicaron en múltiples áreas del conocimiento, entre las cuales se encuentra la sismología.

Alrededor del mundo, la aplicación de modelos basados en *Machine Learning* en la solución de problemas en el campo de la sismología se realiza a partir de la década de los 90 (Wang y Teng, 1995; Zhao y Takano, 1999). Los modelos implementados obtuvieron resultados aceptables, empleando principalmente redes neuronales artificiales para tareas como la detección de eventos (Wang y Teng, 1995) y para el picado de arribos de ondas de cuerpo (Zhao y Takano, 1999). Actualmente, el aumento en la instrumentación sísmica, y en consecuencia, una mayor cantidad de data sísmica, promueve la investigación de estos métodos, ya que es bien sabido que el desempeño de los modelos mejora si se incrementa la cantidad de data.

Recientemente, se ha intentado aplicar *Machine Learning* a la caracterización de fuentes sísmicas (Kriegerowski et al., 2019; Kuang et al., 2021; Lomax et al., 2019; Mousavi y Beroza, 2020; Perol et al., 2018; Shen y Shen, 2021; Zhang et al., 2021). En el enfoque ConvNetQuake de Perol et al. (2018), se utilizó una red neuronal convolucional para detectar sismos (distinguir entre ruido y terremoto) y para determinar la región a la que corresponde cada terremoto detectado. Este estudio fue

ampliado posteriormente por Lomax et al. (2019) en la ubicación de telesismos a nivel global. Mousavi y Beroza (2020) diseñaron una red neuronal convolucional recurrente para estimar las magnitudes de los terremotos. Cabe destacar que estos estudios se centraron en el análisis de formas de onda provenientes de una sola estación, lo que va en contra de la intuición común de que se requieren por lo menos tres estaciones sísmicas para triangular y ubicar una fuente sísmica.

Por otro lado, un enfoque multiestación tomaría como dato de entrada, todas las formas de onda registradas por la red sísmica por evento sísmico. Este hecho resulta ser negativo ya que disminuye la cantidad de muestras disponibles para entrenamiento. Además, si el sismo es pequeño podría no ser detectado por todas las estaciones de la red, hecho que podría ser perjudicial o beneficioso dependiendo del enfoque que se le dé. Otro punto importante a tomar en cuenta sería definir como concatenar todas las formas de onda de la red sísmica en un solo dato representativo. Lo ideal sería que la red sísmica tenga una estructura euclideana, es decir, que las estaciones estén uniformemente distanciadas la una de la otra, lo cual permitiría organizar la data en, por ejemplo, un arreglo 2D como una imagen. Desafortunadamente, la mayoría de las redes sísmicas no están dispuestas en una estructura regular, por lo que la geometría de la red debe aprenderse implícitamente. Kriegerowski et al. (2019) definió un arreglo de estaciones de tamaño fijo y ordenó las estaciones en orden alfabético de acuerdo al nombre de sus estaciones. Por otro lado, Zhang et al. (2021) propuso un arreglo de estaciones de tamaño fijo formado por las 12 estaciones más cercanas al área de estudio, en la tarea de detectar y estimar los parámetros hipocentrales de la fuente sísmica. De manera similar, la red MagNet de Kuang et al. (2021) dio estimaciones de magnitud teniendo como dato de entrada un arreglo de tamaño fijo conformado y ordenado tomando las estaciones más cercanas al evento detectado. Por último, la red ArrayConvNet de Shen y Shen (2021) logró detectar eventos teniendo como dato de entrada un arreglo de estaciones, ordenando las estaciones de acuerdo al tiempo de la amplitud máxima en la dirección vertical. ArrayConvNet tambíen dio estimaciones de localización 4D (latitud, longitud, profundidad y tiempo de origen) para lo cual, simplemente ordenó las estaciones en orden alfabético. Es importante mencionar que todos estos métodos usaron redes convolucionales en sus arquitecturas. En cada investigación, se menciona la importancia de las redes convolucionales a la hora de extraer características importantes de los sismogramas y su eficiencia computacional en dicha tarea. Por otro lado, ninguno de los métodos incluyó la información espacial de las estaciones y red sísmica. A pesar de todas las dificultades mencionadas a la hora de usar el enfoque multiestación, estos modelos arrojaron estimaciones buenas en la localización de hipocentros, sin embargo, sigue siendo una pregunta abierta si se podrían lograr mejores resultados cuando se toma en cuenta la naturaleza no euclidiana de la red sísmica. Afortunadamente, se ha avanzado mucho en el campo

de las *Graph Neural Networks* (GNNs) (Scarselli et al., 2009), proporcionando un marco robusto para analizar datos no euclidianos utilizando herramientas de *Deep Learning* existentes.

Siguiendo esa línea, van den Ende y Ampuero (2020) propusieron un método para incorporar la información espacial de las estaciones de una red sísmica en la tarea de caracterización de la fuente sísmica, utilizando un enfoque de *Graph Neural Networks*, aplicado a la tarea de caracterización de la fuente sísmica. En la incorporación de la ubicación geográfica de las estaciones en el proceso de aprendizaje, se encontró que el modelo de *Deep Learning* logró un rendimiento superior en la predicción de los parámetros hipocentrales en comparación con un modelo que prescinde de la disposición de la red sísmica. Este representa la red sísmica como un grafo sin enlaces, donde las estaciones son los nodos y cada nodo incluye las formas de onda detectadas. De esta manera, se aprovecha información geográfica que ciertamente se conoce a priori y se usa en metodologías clásicas, sin embargo, al ser un grafo sin nodos, se considera que compartir información entre nodos no será un factor que favorezca el análisis, por lo que se sigue sin emplear todo el potencial de las *Graph Neural Networks* disponible en la literatura actual, por ejemplo, el concepto de *message passing* (Gilmer et al., 2017) presente en modelos sobresalientes como en *Graph Convolutional Networks* (Kipf y Welling, 2017), mecanismo que trata de aprovechar las características de cada nodo, así como la estructura del grafo.

Con base en las investigaciones antes mencionadas, se concluye que el uso de *Graph Neural Networks* para la estimación de parámetros hipocentrales se convierte en una atractiva opción para la automatización en la obtención de predicciones más precisos y rápidos para toda red sísmica de un país altamente sísmico.

1.2 DESCRIPCIÓN DEL PROBLEMA DE INVESTIGACIÓN

El Perú está comprendido entre una de las regiones de más alta actividad sísmica que existe en la Tierra, por lo tanto, está expuesto a este peligro, que trae consigo la pérdida de vidas humanas y pérdidas materiales (Alva y Castillo, 1993).

Actualmente, el Instituto Geofísico del Perú (IGP), a través del Centro Sismológico Nacional (CENSIS), dispone de estaciones sísmicas que se usan para determinar los parámetros hipocentrales de la fuente sísmica, y es la encargada de publicar los reportes sísmicos (Tavera, 2019). Sin embargo, esta información se conoce a posteriori, cuando el evento sísmico ya concluyó y los daños ya ocurrieron. Es por este motivo que Perú se encuentra en la implementación de su propio sistema de alerta temprana denominado Sistema de Alerta Sísmica Peruano (SASPe) que garantice una alerta de emergencia oportuna ante sismos. No obstante, este aún

no está disponible y las chances de tener un gran sismo en las costas del Perú se hace cada vez más grande.

En general, el mundo viene experimentando con estos sistemas que, por medio del estudio de la corteza terrestre, buscan mitigar el impacto que puedan tener los sismos, sin embargo, ha ocurrido que las estimaciones no han sido las adecuadas. Como ejemplo de gran relevancia, podemos mencionar el caso del terremoto y posterior tsunami de Tohoku en 2011, donde una subestimación del tamaño de las olas ocasionó una confiada y lenta evacuación de la población que provocó la muerte de más de 20000 personas (Cyranoski, 2011). Un caso más reciente ocurrió en Indonesia. En Setiembre del 2018, un terremoto de 7.5 Mw provocó un tsunami que golpeó la ciudad de Palu dejando por lo menos 420 muertos. La Agencia Indonesia para la Meteorología, Climatología y Geofísica (BMKG, por sus siglas en indonesio) que en un comienzo había emitido una alerta de tsunami, minutos después fue levantada, aunque el tsunami finalmente si llegó a las costas de Palu, inclusive con olas muchos más altas de lo estimado inicialmente por la BMKG. La pérdida de las comunicaciones y los mareográfos en Palu, los cuales envían señales de alerta de tsunami, dañados por el terremoto previo son algunas de las causas que no permitieron alertar a la población que se vio sorprendida por el fuerte y rápido oleaje (The Guardian, 2018). Estudios recientes sugieren que el terremoto de supercorte (*supershear earthquake* en inglés, donde la velocidad de ruptura a lo largo de la falla supera a las ondas de corte S) pudo haber causado el tsunami con olas más altas de lo esperado. Teniendo en cuenta que el terremoto fue de falla de desgarre lo cual se caracteriza por movimientos horizontales (causado por esfuerzos de corte) y por ende no genera grandes movimientos verticales que provocan los tsunamis (Bao et al., 2019). Finalmente, este año en Perú podemos mencionar el derrame de petróleo en Ventanilla causado por el oleaje anómalo que a su vez fue producto de la erupción de un volcán submarino en Tonga y que las autoridades no pudieron advertir ni prevenir. Dicho evento resultó en un desastre ecológico y personas damnificadas al verse dañado su principal recurso para subsistir (BBC News Mundo, 2022).

Con respecto a los métodos para caracterizar la fuente sísmica que usan *Machine Learning*, se había mencionado la popularidad de modelos que incluyen redes convolucionales en su arquitectura. Estos modelos han tenido resultados notables debido a la eficiencia de las redes convolucionales de extraer las mejores características de arreglos de datos. Sin embargo, el modelo exigía que los arreglos cuenten con un orden en las estaciones predefinido y un tamaño constante del arreglo. Por otra parte, en el trabajo de van den Ende y Ampuero (2020) se representó la red sísmica como un grafo sin enlaces, y vio mejorías en la localización debido a la utilización de las coordenadas geográficas de las estaciones. No obstante, al ser

un grafo sin enlaces, considera que no es importante compartir la información entre sus nodos o estaciones. Entonces, queda la pregunta si al agregar enlaces y compartir información sísmica entre estaciones puede ser beneficioso para mejorar el rendimiento del modelo.

Teniendo en cuenta todo lo anterior, se cree necesario continuar con la investigación en cuanto al empleo de técnicas alternativas como *Machine Learning* y en particular *Graph Neural Networks* para la estimación rápida y precisa de parámetros hipocentrales de un evento sísmico, lo cual a futuro permitirá la mejora de la respuesta ante eventos sísmicos.

1.3 OBJETIVOS DEL ESTUDIO

1.3.1 Objetivo General

Proponer un modelo con base en *Graph Neural Networks* para el análisis de señales sísmicas registradas en una red sísmica, con el fin de brindar una estimación de los parámetros hipocentrales de la fuente sísmica.

1.3.2 Objetivos Específicos

- Revisión bibliográfica de técnicas de *Machine Learning* y *Graph Neural Networks*.
- Preprocesar los registros (de sensores de banda ancha, como velocímetros) provenientes de la red sísmica, tales que sirvan como dato de entrada para el modelo.
- Calibrar un modelo empleando *Graph Neural Networks* que permita estimar la localización y la magnitud de los eventos sísmicos en el área de estudio.
- Analizar los resultados obtenidos del modelo de *Graph Neural Networks* propuesto para la estimación de los parámetros hipocentrales.

1.4 HIPÓTESIS DEL ESTUDIO

1.4.1 Hipótesis General

Teniendo en cuenta que es posible estimar la magnitud y localización de un evento sísmico usando el modelo de *Graph Neural Networks*, se plantea que dichas predicciones son precisas y confiables, lo cual confirma la importancia de analizar la red sísmica como un grafo en la estimación automática de los parámetros hipocentrales de un evento sísmico.

1.4.2 Hipótesis Específicas

- Las herramientas de análisis actuales con base en GNN son suficientes para ser empleadas en áreas como la sismología.
- El modelo de GNN será capaz de predecir con precisión y rapidez los parámetros hipocentrales.
- Los resultados obtenidos de la evaluación confirmarán una mejora en la precisión de los resultados gracias al mejor aprovechamiento de la estructura compleja de la red sísmica.

1.5 METODOLOGÍA

La metodología seguida para el desarrollo de la presente tesis es la siguiente:

- Recolección de información: Se buscó información referida al tema de estimación de parámetros hipocentrales de un evento sísmico y del empleo de algoritmos de *Machine Learning* relacionados a este tema. Se usaron artículos científicos encontrados en la red, revistas científicas y diversos libros referentes a los temas de interés.
- Recopilación de una base de datos sísmica: Búsqueda de información sísmica de alguna red sísmica, en este caso, se empleó parte de la Red Sísmica del Sur de California (SCSN, por sus siglas en inglés)
- Preprocesamiento y procesamiento de los datos: Con la información obtenida se procedió a preprocesar la data para que sirva como dato de entrada para el entrenamiento y evaluación del modelo. El proceso incluyó el uso del lenguaje de programación Python, así como librerías de uso libre para preprocesamiento y procesamiento de datos, diseño de la arquitectura, y el entrenamiento y evaluación del modelo propuesto. Además, se utilizó el poder computacional de computadoras alojadas en la nube como Google Colaboratory y el importante aporte de la computadora del International Research Institute of Disaster Science (IRIDeS) de la Universidad de Tohoku para el entrenamiento de los modelos, entre otros recursos.
- Análisis de resultados: Se discutirán los resultados del modelo propuesto, las razones detrás de los resultados y las mejoras a futuros estudios.

CAPÍTULO II: FUNDAMENTO TEÓRICO

En este capítulo, se revisarán los principales conceptos necesarios para comprender lo desarrollado en la tesis.

2.1 ONDAS SÍSMICAS

Existen múltiples fases que van sucediendo durante el viaje de la onda sísmica a través de los múltiples materiales en el subsuelo. Las dos fases principales, también conocidas como ondas de cuerpo son las ondas P y las ondas S, Figura N° 2.1. Las ondas P o de compresión corresponden a aquellas que viajan en la misma dirección del movimiento. Dichas ondas P se desplazan a través de todos los medios sean estos sólidos o líquidos. Por otra parte, las ondas S o de corte corresponden a aquellas que viajan de manera perpendicular a la dirección de movimiento o tangentes al frente de onda. Las ondas S solamente se desplazan a través de medios sólidos.

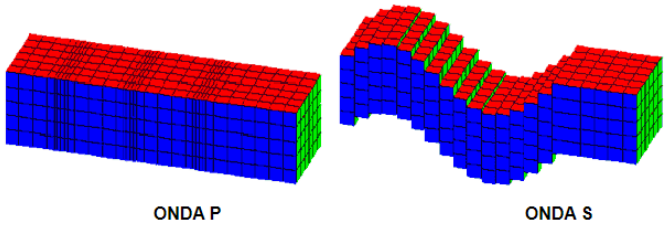


FIGURA N° 2.1: Ondas de cuerpo. Recuperado de udc.es

Estas dos fases principales P y S, tienen diferencias notables que permiten obtener de manera muy rápida características particulares del sismo. En primer lugar, las ondas P viajan mucho más rápido que las ondas S, hecho que permite observar en los sismogramas una separación entre los dos arribos, que se va haciendo más grande en la medida en que la distancia del sismómetro al sitio donde se originó el sismo se hace también más grande. Resulta evidente que la diferencia de tiempo entre los dos arribos, al ser directamente proporcional a la distancia al sitio donde se produjo el movimiento, permite estimar, de manera preliminar, la distancia epicentral.

La otra diferencia importante entre las ondas P y S, consiste en que el movimiento generado por la onda P, es generalmente, mucho menor al generado por la onda S, siendo ésta última la responsable de la mayor parte de los daños ocasionados, no solamente por su mayor magnitud, sino porque debido a que el frente de onda emerge desde el subsuelo de manera casi perpendicular a la superficie, la onda S se mueve paralela a la superficie, de tal manera que cualquier estructura sobre la superficie se ve sometida a fuerzas horizontales aplicadas en su base ejerciendo

un efecto de péndulo que puede llegar a ser amplificado en caso que la frecuencia natural de la mencionada estructura coincida aproximadamente con la frecuencia principal de la onda sísmica.

Existe un tercer tipo de ondas, llamadas superficiales debido a que solo se propagan por las capas más superficiales de la Tierra, decreciendo su amplitud con la profundidad. Estas ondas superficiales tienen una velocidad menor a las ondas de cuerpo, razón por la cual son registradas con posterioridad al arribo de estas. En ocasiones, debido a que las ondas superficiales tienen una mayor duración y una amplitud mayor, pueden llegar a ser incluso más destructivas que las fases iniciales de las ondas de cuerpo. Dentro de este tipo de ondas se pueden diferenciar dos modalidades, denominadas ondas Rayleigh y ondas Love en honor a los científicos que demostraron teóricamente su existencia, Figura N° 2.2.

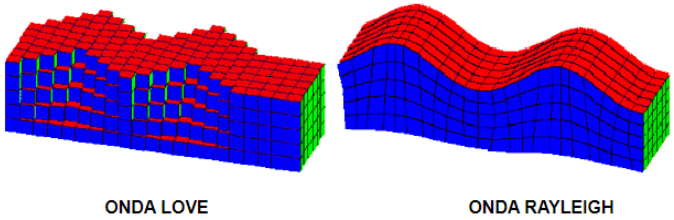


FIGURA N° 2.2: Ondas superficiales. Recuperado de udc.es

Las ondas Rayleigh se forman en la superficie de la Tierra y hacen que las partículas se desplacen según una trayectoria elíptica retrógrada. En cambio, las ondas Love se originan en la interfase de dos medios con propiedades mecánicas diferentes; en este caso tienen un patrón local de movimiento perpendicular a la dirección de propagación y paralelo a la superficie. Debido a que la disminución de la amplitud con la distancia a la fuente es lenta, estas ondas son responsables de daños apreciables en las áreas más cercanas al epicentro.

Una diferencia importante entre las ondas de cuerpo y las ondas superficiales corresponde a la propiedad conocida cómo dispersión. Las ondas de corte viajan a una velocidad más o menos constante, la cual depende casi que exclusivamente de los materiales y no de la longitud de onda. Sin embargo, este comportamiento no es igual para las ondas superficiales. Entre mayor sea la longitud de onda mayor será la penetración que tengan estas en la profundidad y debido a que la velocidad de desplazamiento de las ondas se incrementa con la profundidad tendremos entonces que las ondas superficiales de mayor penetración viajan a una velocidad mayor que aquellas que tienen longitudes de onda menores y, por lo tanto, se desplazan por capas más superficiales a una menor velocidad. El análisis de este fenómeno de dispersión es empleado también en el conocimiento del interior de la Tierra.

2.2 PARÁMETROS HIPOCENTRALES

Los parámetros que caracterizan a la fuente sísmica se llaman parámetros hipocentrales. Tenemos a la latitud y longitud del epicentro, profundidad focal, magnitud y tiempo de origen. La localización de la fuente sísmica y su magnitud son los parámetros de interés en la presente investigación, por lo que se definirán a continuación.

2.2.1 Localización de la fuente sísmica

En sismología es fundamental conocer con precisión la ubicación del origen del sismo, dado que permite, no solo la generación de alertas y estimación de los probables daños que puedan ser causados, sino que permite estudiar la estructura interna de la Tierra. En general, el sitio en profundidad donde se generó la energía es conocido como hipocentro y su proyección sobre la superficie terrestre es conocida como epicentro.

Existen muchas maneras de realizar una localización del hipocentro. La forma más precisa de localización requiere del análisis de los registros sísmicos en varias estaciones alrededor del evento, al menos 3, identificando en cada uno de ellos los arribos principales que, generalmente, corresponden a los tiempos de llegada de las fases P y S, así como también de un modelo de la estructura interna de la Tierra, que contiene la geometría y las velocidades de desplazamiento de las ondas de cada una de las capas o rasgos geológicos del área. Dado que, el conocimiento de dicho modelo está basado principalmente en modelos de velocidad preliminares, este resulta ser, generalmente, aproximado.

Los modelos físico-matemáticos de la propagación de la energía sísmica basados en la ecuación de onda permiten obtener los tiempos de viaje a partir de una fuente conocida, lo que se conoce cómo problema directo. De otro lado, la localización de la fuente a partir de los tiempos de llegada de las diferentes fases y el análisis de los registros sísmicos es conocida como problema inverso.

Cuando se cuenta con la información de varias estaciones alrededor de la fuente la localización del evento puede llevarse a cabo para sismos locales mediante la determinación de las diferencias de tiempo de llegada de las fases P y S.

El primer paso consiste en la determinación del tiempo de origen del evento. Esto se hace mediante el diagrama de Wadati, Figura N° 2.3, que consiste en realizar una regresión lineal entre el tiempo de llegada de la onda P y las diferencias de tiempo entre la onda P la onda S, en toda las estaciones en las cuales se registró el sismo. Con base en esta regresión, se calcula el tiempo para el cual la diferencia entre las ondas P y S sea igual a cero, condición que sólo se cumple en la fuente,

de donde dichas fases salen acopladas, y, por lo tanto, ese tiempo corresponderá al tiempo en que se produjo el evento.

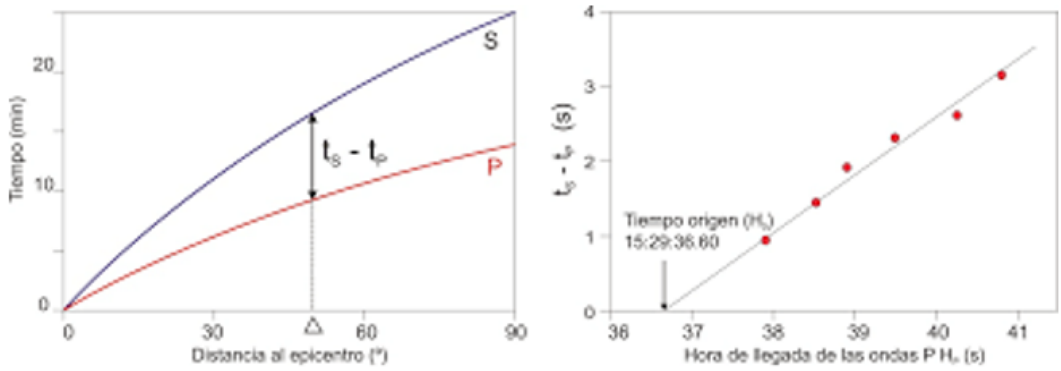


FIGURA N° 2.3: Diagrama de Wadati. Adaptado de revistareduca.es.

Una vez conocido el tiempo en que se produjo el evento se puede calcular la distancia a cada una de las estaciones considerando una velocidad constante de propagación de onda. A continuación, se puede trazar un círculo con dicho radio y centro en cada estación, estableciendo una zona de intersección de dichos círculos, que corresponde al lugar donde se originó el evento. Este procedimiento es bastante aproximado y es conocido como el método de los círculos, Figura N° 2.4.

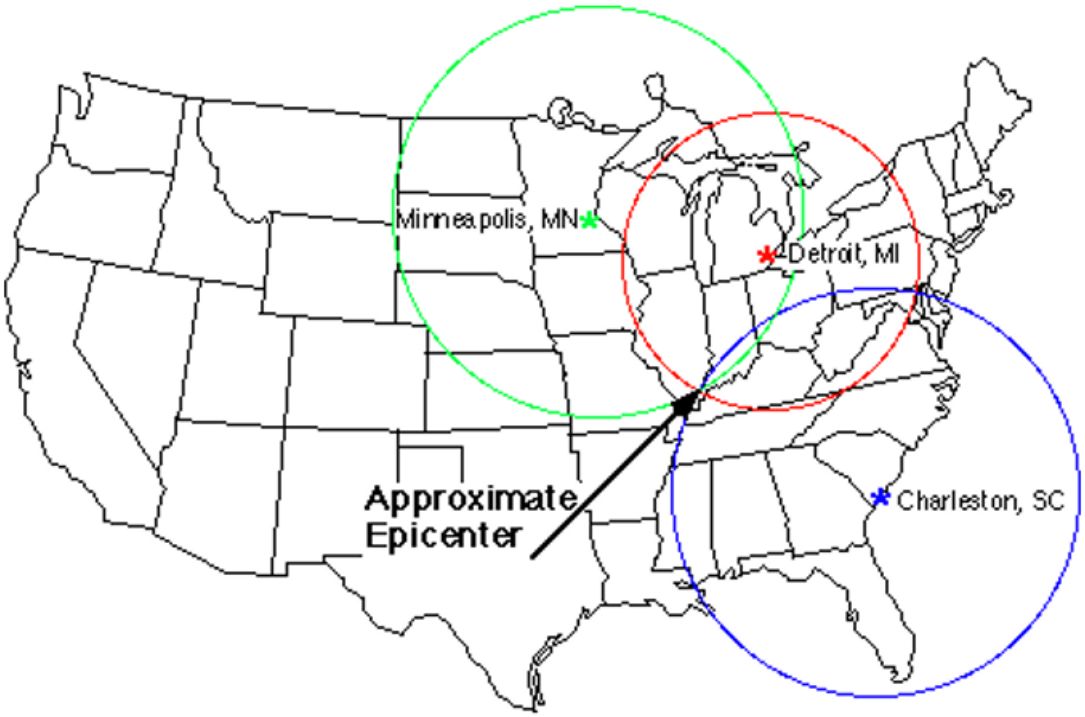


FIGURA N° 2.4: Método de los círculos. Adaptado de IRIS.

Existen en la literatura métodos para localizar la fuente que son más precisos, sin

embargo, requieren del análisis humano, dicho procesamiento no permite la inmediatez de la información. Por lo tanto, dichos métodos no son adecuados para alerta temprana, donde la velocidad de procesamiento es la que prima.

Calculada la localización de la fuente con un método u otro, estos son reportados al público por las instituciones competentes. Muchas veces el primer reporte no es del todo preciso. Hoy en día, varios estudios se hicieron para mejorar la precisión, en los que se conoce como relocalización, y se basa principalmente en registros históricos.

Estudios recientes demuestran que es posible emplear la información histórica de eventos para realizar la localización en tiempo real mediante la aplicación de algoritmos de *Machine Learning* para conseguir un estimativo de los parámetros de manera rápida y con la mayor precisión posible.

2.2.2 Magnitud del evento

El concepto de magnitud fue introducido inicialmente con el fin de tener una medición instrumental objetiva de la energía liberada en un evento sísmico (Richter, 1935). Mientras que el concepto de intensidad está basado en los daños producidos por el evento y la percepción humana, los cuales dependen de la distancia a la fuente y condiciones locales, la magnitud emplea mediciones instrumentales de los desplazamientos registrados ajustados para la distancia epicentral y la profundidad de la fuente, empleando instrumentos con características estándar. Sin embargo, por tratarse de un parámetro empírico no está relacionado directamente con ningún parámetro físico de la fuente.

El concepto original de la magnitud Richter (M_L), fue basado en la medición de la máxima amplitud registrada en los equipos Wood-Anderson estándar de corto periodo de la red sismológica del sur de California para la clasificación de sismos locales en esa región. La determinación de la magnitud de Richter se puede ver de manera gráfica en la Figura N° 2.5, donde la magnitud solo depende de la amplitud máxima y de la diferencia de arribos de las fases P y S. Así mismo, se desarrollaron otras fórmulas empíricas inspiradas en Richter. Por ejemplo, de acuerdo a la fórmula empírica de Lilie, para el cálculo de esta magnitud se emplea la amplitud del mayor arribo (A, en micrómetros), el cual generalmente corresponde a la onda S, corregida con base en la distancia entre la fuente y el receptor (Δ, en kilómetros), calculada a partir de la diferencia de los tiempos de arribo de las ondas P y S.

$$M_L = \log A - 2.48 + 2.76 \log \Delta$$

También se formularon otras maneras de calcular la magnitud: Magnitud para ondas

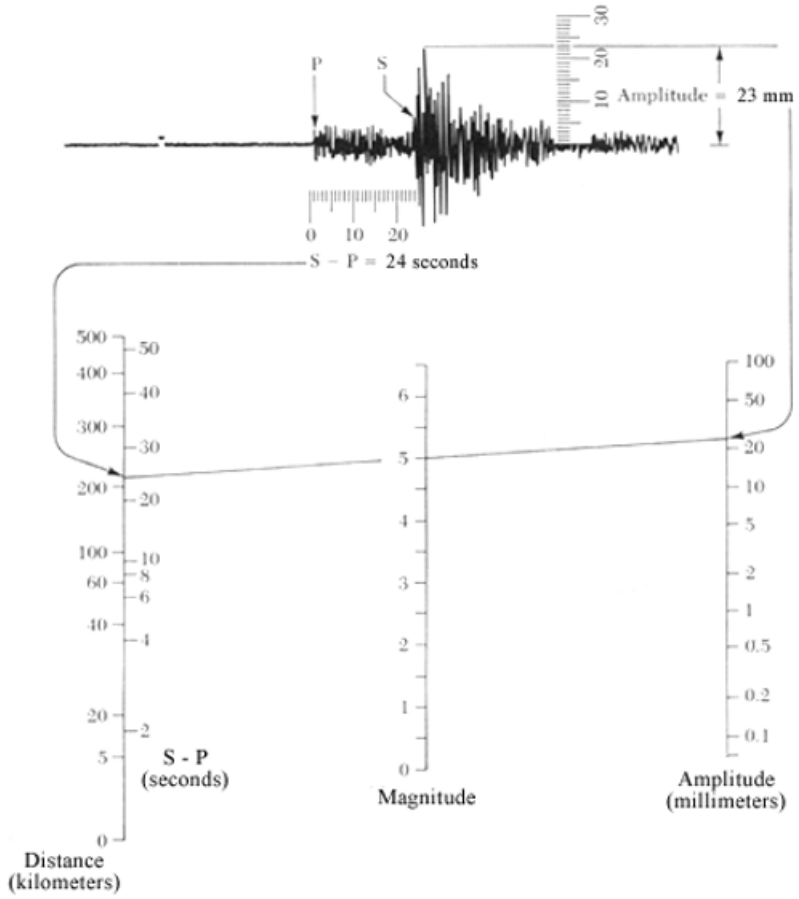


FIGURA N° 2.5: Determinación de la magnitud Richter. Adaptado de IRIS.

superficiales (Ms) y para ondas de cuerpo (mb). Estas mediciones como la medición de Richter son empíricas y no reflejan correctamente el tamaño de los sismos.

La noción de energía liberada en un evento sísmico está definida mediante una cantidad escalar denominada momento sísmico (M_0), dado en unidades de energía. El momento sísmico (M_0) se calcula mediante el producto de la rigidez de la roca (μ), el área involucrada en el movimiento (S) y su desplazamiento promedio ($\langle d \rangle$), concepto que refleja de una mejor manera el tamaño del evento.

$$M_0 = \mu S \langle d \rangle$$

Asociada a este momento sísmico se definió la magnitud de momento sísmico (M_w), la cual refleja la energía asociada con el evento de tal manera que puede ser considerada como la medida de magnitud más apropiada para la caracterización de un sismo. Sin embargo, esta magnitud sólo puede ser calculada después de conocer con cierto grado de precisión la geometría de la superficie donde se produjo el movimiento, lo cual requiere estudios posteriores detallados, razón por

la cual no es factible su uso en alertas tempranas.

$$M_w = \log \frac{M_0}{1.5} - 10.73$$

A pesar de las limitaciones que presentan las mediciones de magnitud empíricas para magnitudes grandes, la posibilidad de poder obtener rápidamente sus valores hace que estas magnitudes sean de gran utilidad para el estudio y caracterización de los eventos. Sin embargo, es importante tener en cuenta que subestiman la magnitud para eventos con magnitudes de momento mayores a 6. Este fenómeno es conocido como saturación y se muestra en la Figura N° 2.6.

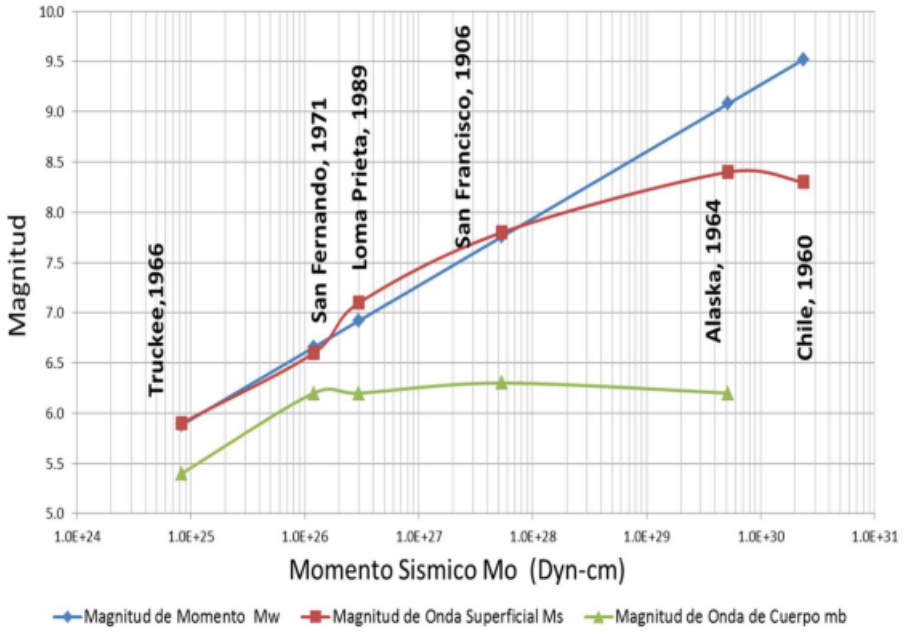


FIGURA N° 2.6: Saturación de ondas de cuerpo y superfici. Adaptado de Ochoa Gutiérrez (2016).

2.3 APRENDIZAJE AUTOMÁTICO

Aprendizaje Automático (o *Machine Learning*) es la ciencia (y el arte) de programar computadoras para que puedan aprender de los datos. (Géron, 2019)
Machine Learning es una aplicación de Inteligencia Artificial que proporciona a las computadoras la capacidad de aprender y mejorar automáticamente a partir de la experiencia sin estar programados explícitamente en la solución de la tarea que uno desea resolver. Se centra en el uso de algoritmos y desarrollo de modelos que pueden acceder a los datos y utilizarlos para aprender por si mismos.

Machine Learning ha resultado ser una importante herramienta al servicio del hombre, la cual destaca en tareas como:

- Simplificar código de problemas de automatización para los que las soluciones existentes requieren muchos ajustes manuales o estan sujetas a muchas reglas.
- Encontrar soluciones a problemas complejos para los que no existe una buena solución utilizando un enfoque tradicional.
- Adaptación a datos cambiantes o a nuevos datos incluso en tiempo real.
- Obtener conocimientos sobre problemas complejos y sacar conclusiones de grandes cantidades de datos.

Existen muchas maneras de categorizar los algoritmos de *Machine Learning*, la más común es aquella que las clasifica según la participación del hombre en el proceso de aprendizaje, estas son: Aprendizaje Supervisado, Aprendizaje No Supervisado, Aprendizaje Semi-Supervisado y Aprendizaje Reforzado. En la presente investigación se usará un enfoque de Aprendizaje Supervisado.

2.3.1 Aprendizaje Supervisado

En este enfoque, la data de entrenamiento incluye también las soluciones esperadas o etiquetas. Desde un punto de vista matemático, se conoce la variable de entrada X y la variable de salida Y, lo que se plantea es usar un algoritmo para aprender la función f que mejor relaciona a X con Y:

$$f(X) = Y$$

El fin es que esa función f sea capaz de predecir Y cuando se le entrege al modelo un X nunca ante visto.

Se le conoce como Aprendizaje Supervisado debido a que el proceso de aprendizaje de un algoritmo a partir de la data de entrenamiento se puede considerar como un maestro que supervisa el proceso de aprendizaje. Se conocen las respuestas correctas, el algoritmo hace predicciones iterativas sobre la data de entrenamiento y es corregido por el maestro. El aprendizaje se detiene cuando el algoritmo alcanza un nivel aceptable de rendimiento.

En la Figura N° 2.7, las variables de entrada X son las imágenes de frutas y las variables de salida Y son los números que representan cada categoría de frutas. Para el entrenamiento, el algoritmo se apoya de las imágenes con sus etiquetas. Una vez obtenido el modelo, este se evalúa con imágenes nuevas sin etiquetas, y se espera que se categoricen correctamente.

Ahora, existen una gran cantidad de problemas del tipo supervisado, los cuales pueden agruparse en dos principales grupos:

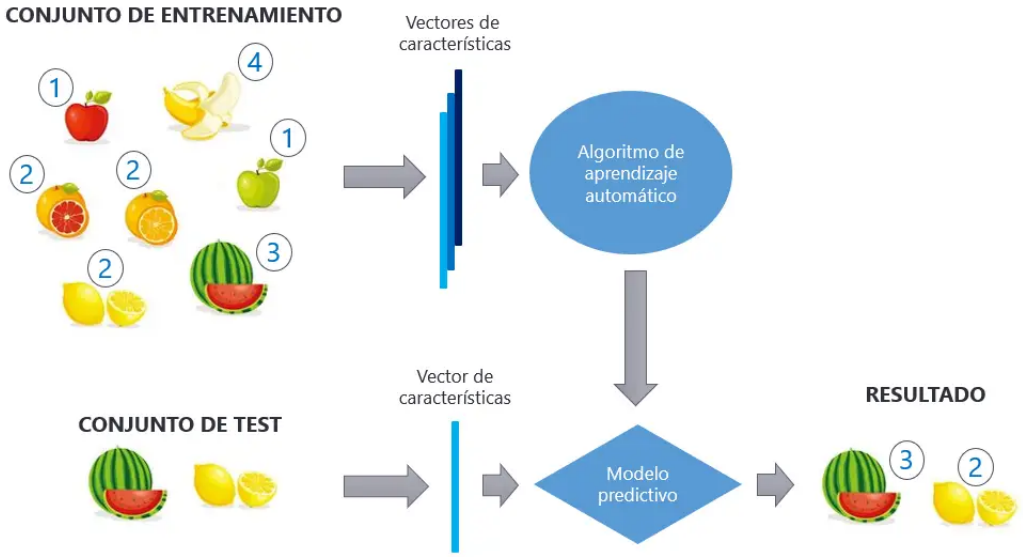


FIGURA N° 2.7: Aprendizaje Supervisado. Recuperado de Calvo (2019)

- Problemas de Clasificación
- Problemas de Regresión

Esta investigación, trata de resolver un problema de regresión.

2.3.2 Algoritmos de regresión

Mientras que un algoritmo de clasificación busca predecir una clase o valor cualitativo, un algoritmo de regresión lo que busca es predecir un valor numérico objetivo, Figura N° 2.8, como el precio de un automóvil, dado un conjunto de características o *features* (kilometraje, edad, marca, etc.) llamadas predictores. Este tipo de tarea se llama regresión. Por ejemplo, para entrenar el sistema, debe darle muchos ejemplos de automóviles, incluidos sus predictores y sus etiquetas (es decir, sus precios).

2.4 REDES NEURONALES

Las redes neuronales (o *neural networks*, en inglés) son modelos simples del funcionamiento del sistema nervioso. Las unidades básicas son las neuronas, que generalmente se organizan en capas, como se muestra en la Figura N° 2.9.

Una red neuronal es un modelo simplificado que emula el modo en que el cerebro humano procesa la información, como se aprecia en la Figura N° 2.10.

FIGURA N° 2.8: Regresión. Adaptado de Géron (2019)

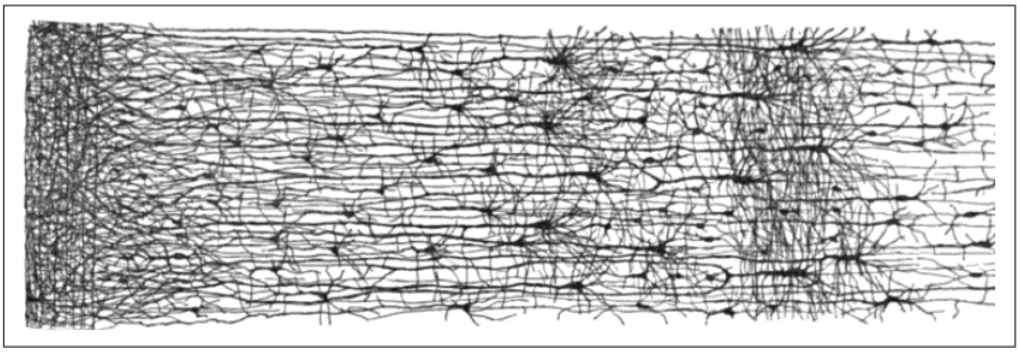


FIGURA N° 2.9: Red Neuronal Biológica. Adaptado de Cajal (1899)

2.4.1 ¿Cómo aprende una red neuronal?

Las unidades de procesamiento o neuronas se organizan en capas. Normalmente, hay tres partes en una red neuronal: Una capa de entrada, con unidades que representan los campos de entrada; una o varias capas ocultas; y una capa de salida, con una unidad o unidades que representan las valores a predecir. Las neuronas se conectan con fuerzas de conexión variables (o pesos). Los datos de entrada se presentan en la primera capa, y los valores se propagan neurona por neurona y capa por capa. Al final, se envía un resultado desde la capa de salida.

Al principio, todas los pesos son aleatorios y las predicciones de la red son, posiblemente, absurdas. La red aprende a través del entrenamiento. Continuamente se presentan a la red ejemplos para los que se conoce el resultado, entonces, los resultados obtenidos del modelo se comparan con los resultados conocidos. La información procedente de esta comparación se pasa hacia atrás a través de la red, *backpropagation*, cambiando las ponderaciones gradualmente, a través de la búsqueda de un mínimo en la función costo. A medida que progresa el entrenamiento, la red se va haciendo cada vez más precisa en la replicación de resultados conocidos. Una vez entrenada, la red se puede aplicar a casos futuros en los que se

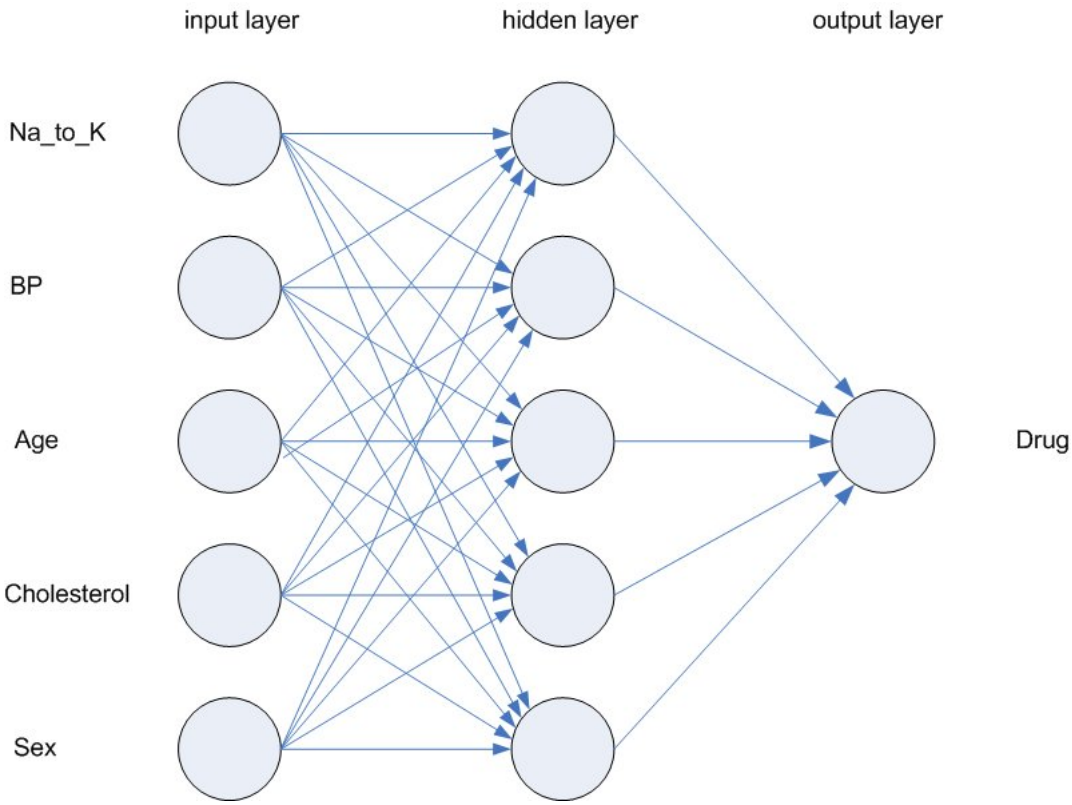


FIGURA N° 2.10: Red Neuronal Artificial. Adaptado de IBM (2020)

desconoce el resultado.

Antes de presentar los tipos de Redes Neuronales, conviene primero conocer el concepto de *Deep Learning*

2.4.2 Aprendizaje Profundo

Aprendizaje Profundo (o *Deep Learning*) es un subcampo de *Machine Learning* basado en redes neuronales artificiales con *representation learning* (conjunto de técnicas que permite a un sistema descubrir automáticamente las representaciones necesarias para la detección o clasificación de características a partir de datos sin procesar). Esto reemplaza hacer feature engineering (escoger las características más relevantes de la data) de manera manual y permite que sea la máquina quien escoja, apoyada de una cantidad enorme de muestras.

Para conseguir ese nivel de abtracción, el modelo de redes neuronales agrega varias capas ocultas a su arquitectura. Cada capa extrae características de un nivel cada vez más alto hasta llegar a la respuesta final.

En otras palabras, la muestra ingresa a la red y obtiene características simples de la misma. A medida que se profundiza más en la red, se forman características más complejas a partir de combinar las simples, hasta llegar a la predicción final. Es por

este motivo que se le conoce por Aprendizaje Pofundo o *Deep Learning*.

2.4.3 Tipos de redes neuronales

Los 3 tipos más importantes de redes neuronales son: Perceptrón multicapa, redes neuronales convolucionales, redes neuronales recurrentes.

- Perceptrón Multicapa: Un perceptrón multicapa o multilayer perceptron (MLP, por sus siglas en inglés) es la clase más simple de red neuronal.

  Consiste en al menos 3 capas: Capa de entrada,capa o capas ocultas y capa de salida. A excepción de los nodos de entrada, cada nodo es una neurona que usa una función de activación no lineal. MLP ultiliza una técnica de aprendizaje supervisada llamada *backpropagation* para el entrenamiento. Son las funciones de activación no lineal lo que hace a MLP una herramienta poderosa, ya que, puede distinguir data que no es linealmente separable. Además, le agrega complejidad para modelar la frecuencia de los potenciales de acción, o funciones de activación, de las neuronas biológicas.

  Las funciones de activación no lineales más conocidas son la tangente hiperbólica y la función sigmoide.

$$y(x) = \tanh(x) \tag{2.1}$$

$$y(x) = (1 + e^{-x})^{-1} \tag{2.2}$$

  En la actualidad, los estudios realizados en el campo del *deep learning* reconocen a la unidad lineal rectificada o rectified linear unit (ReLU, por sus siglas en inglés) como la función de activación más usada debido a la posibilidad de superar las inestabilidades numéricas a las que se llegan al usar la función sigmoide.

$$\text{ReLU}(x) = \begin{cases} x, & \text{if } x \geq 0 \\ 0, & \text{if } x < 0 \end{cases} \tag{2.3}$$

- Redes Neuronales Convolucionales: También llamadas Convolutional Neural Networks (CNN, por sus siglas en inglés). Este tipo de red es una variación de un perceptrón multicapa, sin embargo, debido a que su aplicación es realizada en matrices bidimensionales, son muy efectivas para tareas de visión artificial, como en la clasificación y segmentación de imágenes, entre otras aplicaciones.

Las redes neuronales convolucionales consisten en múltiples capas de filtros convolucionales de una o más dimensiones. Después de cada capa, por lo general se añade una función de activación no-lineal. Y si se gusta, se puede agregar una capa que reduzca la dimensionalidad, *Max-Pooling Layer*

Como redes de clasificación, al principio se encuentra la fase de extracción de características, compuesta de neuronas convolucionales y de reducción de muestreo.

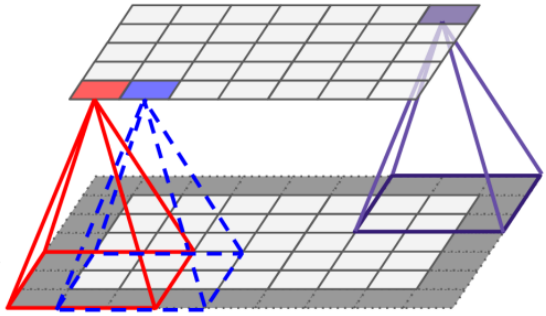


FIGURA N° 2.11: Convolución de un filtro 3x3 sobre una imágen 2D. Adaptado de Géron (2019)

En la fase de extracción de características, la imágen 2D pasa a través de una serie de filtros. Estos filtros realizan una operación a lo largo y ancho de toda la imagen, como una convolución, Figura N° 2.11. El resultado de dicha operación se le conoce como mapa de características o *feature map* y se calcula como:

$$Y_j = \sigma\left(\sum_i K_{ij} \circledast Y_i + b_j\right) \tag{2.4}$$

Donde:

- Y_j : Neurona de la capa j formadas por la combinación lineal de las salidas de las neuronas de la capa anterior
- Y_i : Neurona de la capa anterior
- K_{ij} : Filtro
- b_j : Sesgo o Bias (en inglés)
- σ : Función de activación no lineal

El operador de convolución tiene el efecto de filtrar la imagen de entrada con un filtro o *kernel* previamente entrenado. Esto transforma los datos de tal manera que ciertas características (determinadas por la forma del filtro) se vuelven más dominantes en la imagen de salida al tener estas un valor numérico más alto asignados a los pixeles que las representan. Estos filtros tienen ha-

bilidades de procesamiento de imágenes específicas, como por ejemplo la detección de bordes que se puede realizar con filtros que resaltan la gradiente en una dirección en particular. Sin embargo, los filtros que son entrenados por una red neuronal convolucional generalmente son más complejos para poder extraer otras características más abstractas y no triviales.

Por ejemplo, una CNN contiene varias capas ocultas especializadas y con una jerarquía, esto quiere decir que las primeras capas pueden detectar líneas, curvas y se van especializando hasta llegar a capas más profundas que reconocen formas complejas como un rostro o la silueta de un animal.

Al final de la red se encuentran neuronas de perceptron sencillas para realizar la clasificación final sobre las características extraídas. Un ejemplo típico de red convolucional se puede observar en la Figura N° 2.12

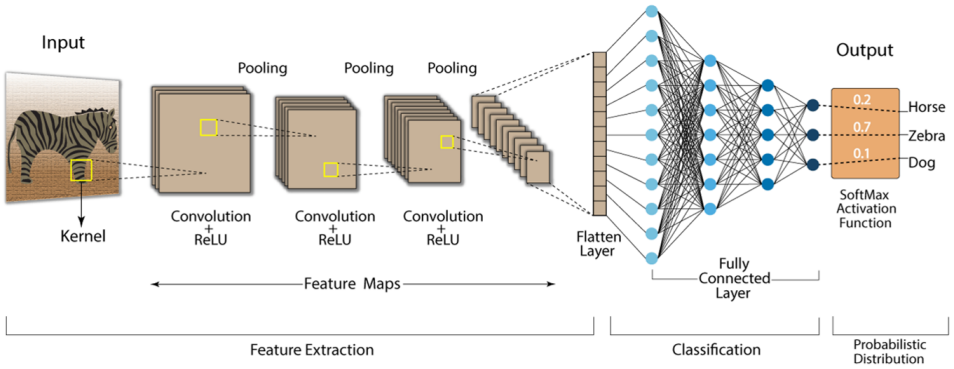


FIGURA N° 2.12: Red Convolucional para la clasificación de imágenes. Recuperado de Swapna (2021)

- Redes Recurrentes: También llamadas Recurrent Neural Networks (RNN, por sus siglas en inglés). Son una clase de redes para analizar datos de series temporales permitiendo tratar la dimensión de tiempo. Derivados de las redes neuronales, las redes recurrentes pueden usar su estado interno (memoria) para procesar secuencias de entradas de longitud variable. Esto los hace aplicables a tareas como el procesamiento del lenguaje natural o el reconocimiento de voz.

2.4.4 Aplicaciones en la sismología

Sin duda la sismología no ha sido sido indiferente al reciente éxito de *Machine Learning*, por lo que se ha intentado aplicar para la solución de tareas típicas del campo, tales como: Inversión de curvas de dispersión, detección de sismos, en la determinación de los parámetros hipocentrales de un evento sísmico, entre otras tareas. A continuación se resumen algunas aplicaciones:

- Inversión de Curvas de Dispersión: Yablokov et al. (2021) describen un nuevo algoritmo para la inversión de perfiles de velocidad de ondas transversales unidimensionales a partir de curvas de dispersión del modo fundamental de las ondas superficiales de Rayleigh. Se propone una red neuronal artificial para resolver el problema inversión. El modelo recibe curvas de dispersión y calcula las velocidades y espesores de las capas de suelo.

- Detección de Sismos: Perol et al. (2018) demostraron que un algoritmo de red neuronal convolucional podría usarse para la detección y la ubicación de sismos. Ross et al. (2018) utilizaron el algoritmo de CNN para identificar ondas P, ondas S y ruido en datos continuos. El rendimiento es mucho mejor que con los algoritmos tradicionales de activación y detección y, si se aplica a redes de alerta temprana en tiempo real, reducirá sustancialmente las falsas alertas.

  Zhu y Beroza (2019) diseñaron una arquitectura, llamada PhaseNet, que tomaba las 3 componentes de un sismograma y devolvía la distribución de probabilidad de obtener una onda P, onda S y ruido. PhaseNet estaba compuesto por capas convolucionales y deconvolucionales para tratar con series de tiempo unidimensionales.

  Actualmente, existen herramientas de *Machine Learning* que se usan para la detección y asociación de ondas de fase, como es el caso de easyQuake (Walter et al., 2020). EasyQuake da la opción de usar el *Generalized Phase Detection* (GPD) *picker* (Ross et al., 2018) o el EQTransformer *picker* (Mousavi et al., 2020) para detección de fase; en cuanto a la asociación de fase o diferenciar entre onda P y S, se sigue usando PhasePApy (Chen y Holland, 2016). EasyQuake viene siendo empleado por la *Oklahoma Geological Survey* (OGS) para completar sus catálogos sísmicos, siendo la primera implementación en el mundo real de *Machine Learning* en sismología. Cabe mencionar que por razones de respeto al público, siempre se revisan los resultados por un analista previo a su paso al catálogo de la *United States Geological Survey* (USGS) y a su posterior publicación.

- Estimación de Parámetros Hipocentrales: Así también, se ha intentado aplicar *Machine Learning* a la caracterización de fuentes sísmicas (Kriegerowski et al., 2019; Kuang et al., 2021; Lomax et al., 2019; Mousavi y Beroza, 2020; Perol et al., 2018; Shen y Shen, 2021; Zhang et al., 2021). En la arquitectura ConvNetQuake de Perol et al. (2018), se utilizó una red neuronal convolucional para detectar sismos (distinguir entre ruido y terremoto) y para determinar la localización regional de cada terremoto detectado. Este estudio fue ampliado posteriormente por Lomax et al. (2019) para la detección de sismos a nivel global.

Mousavi y Beroza (2020) diseñaron una red neuronal convolucional recurrente para estimar las magnitudes de los terremotos. De igual forma, la arquitectura MagNet de Kuang et al. (2021) dió estimaciones de magnitud pero usando un enfoque multiestación.

Siguiendo con el enfoque multiestación, Zhang et al. (2021) propuso un modelo capaz de detectar y estimar los parámetros hipocentrales de la fuente sísmica, logrando buenos resultados con tan sólo 4 segundos de ventana de sismograma llegada la onda P. Por último, la arquitectura ArrayConvNet de Shen y Shen (2021) se entrenó por separado para detectar y para localizar 4D (latitud, longitud, profundidad y tiempo de origen) la fuente sísmica.

van den Ende y Ampuero (2020) propusieron un método para incorporar la geometría de una red sísmica en arquitecturas de *Deep Learning* utilizando un enfoque de *Graph Neural Networks* (Scarselli et al., 2009), aplicado a la tarea de caracterización de la fuente sísmica. Al incorporar la ubicación de las estaciones en el modelo, se logró un rendimiento superior en la predicción de los parámetros hipocentrales del evento sísmico en comparación con un modelo que prescinde de la disposición de la red sísmica.

2.5 TEORÍA DE GRAFOS

2.5.1 Grafos

Los grafos son un lenguaje general para describir y analizar entidades con relaciones e/o interacciones, Figura N° 2.13. Formalmente, un grafo $G = (V, E)$ está definido por un conjunto de nodos o *nodes* V y un conjunto de aristas o *edges* E entre estos nodos. Denotamos un enlace que va del nodo $u \in V$ al nodo $v \in V$ como $(u, v) \in E$.

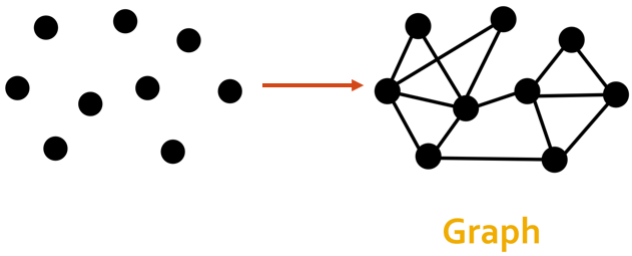


FIGURA N° 2.13: Grafo. Adaptado de Hamilton (2020)

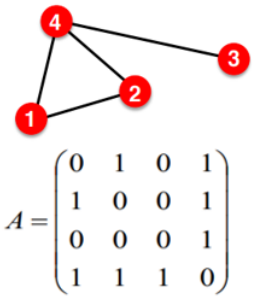
2.5.2 Matriz de Adyacencia

Una forma conveniente de representar grafos es mediante una matriz de adyacencia $A \in \mathbb{R}^{|V|x|V|}$. Para representar un grafo con una matriz de adyacencia, ordenamos los nodos en el grafo de modo que cada nodo indexe una fila y columna en

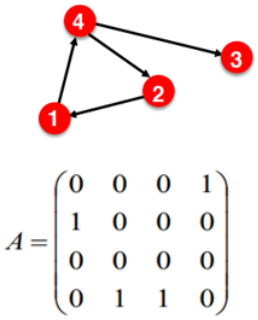
particular en la matriz de adyacencia. Entonces podemos representar la presencia de aristas como entradas en esta matriz: $A[u,v] = 1$ si $(u,v) \in E$ y $A[u,v] = 0$, en caso contrario.

2.5.3 Tipos de Grafos

Si el grafo es *Undirected Graph*, entonces A será una matriz simétrica, pero si el grafo es *Directed Graph* (es decir, la dirección del borde importa), entonces A no será necesariamente simétrico. En la Figura N° 2.14 se muestra un ejemplo de cada uno con sus respectivas matrices de adjacencia.



(a) *Undirected Graph* y su matriz de adyacencia. Adaptado de Hamilton (2020)



(b) *Directed Graph* y su matriz de adyacencia. Adaptado de Hamilton (2020)

FIGURA N° 2.14: *Directed Graph* y *Undirected Graph*

Algunos grafos también pueden tener *weighted edges*, donde las entradas en la matriz de adyacencia son valores reales arbitrarios en lugar de $0, 1$, tal y como se muestra en Figura N° 2.15. Por ejemplo, un *weighted edge* en un gráfico de interacción proteína-proteína podría indicar la fuerza de la asociación entre dos proteínas.

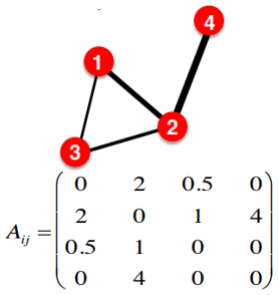


FIGURA N° 2.15: *Weighted Graph* y su matriz de adyacencia. Adaptado de Hamilton (2020)

2.5.4 Feature Information

En muchos casos también tenemos información de atributos o características asociadas con un grafo. Muy a menudo, estos son atributos a nivel de nodo o *node*

features que representamos usando una matriz de valor real $X \in \mathbb{R}^{|V|xm}$, donde asumimos que el orden de los nodos es consistente con el orden en la matriz de adyacencia. Por ejemplo, una imagen de perfil, edad, género, intereses y ubicación asociados con un usuario en una red social.

En casos excepcionales, también consideraremos grafos que tienen características asociadas a sus aristas o *edge features*. Por ejemplo, para saber que tipo de conexión o que tan fuerte es la relación entre dos usuarios en una red social.

En algunos casos, incluso asociamos características de valor real a grafos completos o *graph features*. Por ejemplo, en una red sísmica, podemos asociar a cada evento sísmico con sus parámetros hipocentrales.

2.5.5 Tipos de problemas en grafos

Las tareas más importantes que surgen al trabajar con grafos se pueden dividir en 3 categorías principales:

- Node Classification: Predecir la propiedad de un nodo. Por ejemplo, categorizar usuarios o ítems en una página web de ventas.
- Edge Prediction: Predecir si faltan enlaces entre dos nodos. Por ejemplo, cuando Facebook te sugiere una amistad.
- Graph Classification or Regression: La tarea aquí es clasificar un grafo completo en categorías o, en el caso de regresión, relacionar un grafo completo con un número o números reales. Por ejemplo, en el caso de clasificación, predecir las propiedades de una molécula, si es o no tóxica; y en el caso de regresión, predecir el valor de solubilidad de una molécula, como se muestra en la Figura N° 2.16.

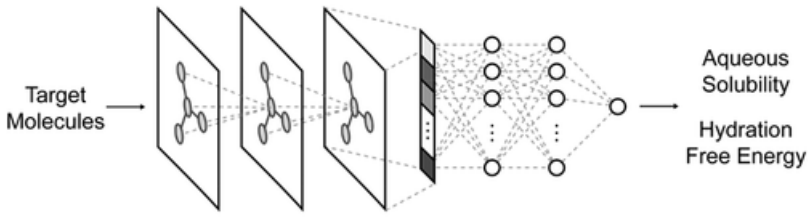


FIGURA N° 2.16: *Graph Regression*. Adaptado de Cho y Choi (2019)

La idea clásica para resolver este problema se basa en aplicar un par de *graph convolutions*, para luego aplicar una capa de *graph pooling*, esto provocará que se forme un ”supernodo”que represente a todo el grafo, finalmente este ”supernodo”se pasa por una MLP que nos dará las predicciones finales. Estos nuevos conceptos se explicarán en el siguiente apartado.

Esta investigación plantea resolver un problema de *Graph Regression*.

2.6 REDES NEURONALES GRÁFICAS

En esta parte, se presenta a las Redes Neuronales Gráficas o *Graph Neural Networks* (GNN), que es un marco general para definir redes neuronales profundas en grafos. La idea clave es que se quiere generar representaciones de nodos que realmente dependan de la estructura del grafo, así como de cualquier información de características que podamos tener.

El principal desafío en el desarrollo de *encoders* complejos para datos estructurados en grafos es que nuestra caja de herramientas de *Deep Learning* habitual no se aplica. Por ejemplo, las redes neuronales convolucionales están bien definidas solo sobre entradas estructuradas en cuadrícula (por ejemplo, imágenes), mientras que las redes neuronales recurrentes están bien definidas solo sobre secuencias (por ejemplo, texto). Para definir una red neuronal profunda sobre grafos, necesitamos definir un nuevo tipo de arquitectura de *Deep Learning*.

2.6.1 *Message Passing*

Una característica de un GNN es que utiliza una forma de transmisión de mensajes neuronales o *neural message passing* (Gilmer et al., 2017) en la que los mensajes vectoriales se intercambian entre nodos y se actualizan mediante redes neuronales.

Entonces, podemos tomar un grafo de entrada $G = (V, E)$, junto con un conjunto de *node features* $X \in \mathbb{R}^{dx|V|}$, y usar esta información para generar *node embeddings* $z_u,\ \forall u \in V$.

Durante cada iteración de *message passing* en un GNN, una *hidden embedding* $h_u^{(k)}$ correspondiente a cada nodo $u \in V$ se actualiza de acuerdo con la información agregada de la vecindad de u, $\mathcal{N}(u)$. Esta actualización de *message passing* se puede expresar de la siguiente manera:

$$h_u^{(k+1)} = \text{UPDATE}\left(h_u^{(k)},\ \text{AGGREGATE}\left(\left\{h_v^{(k)},\ \forall v \in \mathcal{N}(u)\right\}\right)\right) \tag{2.5}$$

$$= \text{UPDATE}\left(h_u^{(k)},\ m_{\mathcal{N}(u)}^{k}\right) \tag{2.6}$$

donde UPDATE y AGGREGATE son funciones diferenciables arbitrarias (es decir, redes neuronales) y $m_{\mathcal{N}(u)}$ es el "*message*"que se agrega de la vecindad de u, $\mathcal{N}(u)$. Usamos superíndices para distinguir los *embbedings* y funciones en diferentes iteraciones de *message passing* o diferentes "*layers*"de la GNN.

Las *embbedings* iniciales en $k = 0$ se establecen en las características de entrada

para todos los nodos, es decir, $h_u^{(0)} = x_u,\ \forall u \in V$. Después de ejecutar K iteraciones de *message passing* de la GNN, podemos usar la salida de la capa final para definir los *embbedings* para cada nodo, es decir,

$$z_u = h_u^{(K)},\ \forall u \in V \tag{2.7}$$

La intuición básica detrás del marco de *message passing* de una GNN es sencilla: En cada iteración, cada nodo agrega información de su vecindario local y, a medida que avanzan estas iteraciones, el *embbeding* de cada nodo contiene más y más ïnformación"de pasos más lejanos del grafo.

Pero, ¿qué tipo de información codifican realmente estos *node embbedings*? Generalmente, esta información se presenta en dos formas. Por un lado, hay información estructural sobre el grafo. Por ejemplo, después de k iteraciones de *message passing* de GNN, el *embbeding* $h_u^{(k)}$ del nodo u podría codificar información sobre todos los nodos en la vecindad de k-saltos de u.

Además de la información estructural, el otro tipo de información clave capturada por los *node embbedings* de GNN se basa en características. Después de k iteraciones de *message passing* de GNN, los *embbedings* para cada nodo también codifican información sobre todas las características en su vecindario de k-saltos.

Ahora, podemos comenzar por definir el modelo de GNN más básico, que es una simplificación de los modelos GNN originales propuestos por Scarselli et al. (2009). El *message passing* del GNN básico, se define como

$$h_u^{(k)} = \sigma\left(W_{\text{self}}^{(k)} h_u^{(k-1)} + W_{\text{neigh}}^{(k)} \sum_{v \in \mathcal{N}(u).} h_v^{(k-1)} + b^{(k)}\right) \tag{2.8}$$

donde $W_{\text{self}}^{(k)}$, $W_{\text{neigh}}^{(k)} \in \mathbb{R}^{d^{(k)} x d^{(k-1)}}$ son matrices con parámetros entrenables y σ denota una no linealidad *element-wise* o elemento por elemento de la matriz. Con respecto al *bias* $b^{(k)} \in \mathbb{R}^{d^{(k)}}$ es frecuentemente omitido por simplicidad en la notación, sin embargo, incluirla en el algoritmo puede ser importante para obtener un gran rendimiento. Entonces, la función AGGREGATE es una simple suma de los *node features* de los nodos del vecindario y la función UPDATE es una red neuronal.

Algunas GNN puede ser expresadas como operaciones entre matrices a nivel del grafo completo, por ejemplo, en el caso de esta GNN básica:

$$H^{(k)} = \sigma\left(AH^{(k-1)} W_{\text{neigh}}^{(k)} + H^{(k-1)} W_{\text{self}}^{(k)}\right) \tag{2.9}$$

donde $H^{(k)} \in \mathbb{R}^{|V|xd}$ denota la matriz de los *node features* de cada nodo en la capa t de la GNN (cada nodo corresponde a una fila en la matriz), A es la matriz de adyacencia del grafo completo; notar que se omitió el *bias*.

A manera de simplificación, se suele añadir *self-loops* y omitir la explícita función UPDATE. Entonces, definimos *message passing* como:

$$h_u^{(k)} = \text{AGGREGATE}\left(\left\{h_v^{(k-1)},\ \forall v \in \mathcal{N}(u) \cup \{u\}\right\}\right) \tag{2.10}$$

,

donde ahora la función AGGREGATE se toma sobre el conjunto $\mathcal{N}(u) \cup \{u\}$, es decir, los vecinos del nodo, así como el propio nodo.

Y si lo queremos ver de manera global, sería asi:

$$H^{(k)} = \sigma\left((A+I)\,H^{(k-1)}W^{(k)}\right) \tag{2.11}$$

Así mismo, tomar la función AGGREGATE como una simple suma puede conllevar a resultados inestables y dificultades en la optimización, debido mayormente a la sensibilidad al grado de los nodos. Es por esta razón que se prefiere normalizar, y una manera simple es tomar el promedio:

$$m_{\mathcal{N}(u)} = \frac{\sum\limits_{v \in \mathcal{N}(u)} h_v}{|\mathcal{N}(u)|} \tag{2.12}$$

2.6.2 *Graph Convolutional Networks*

Los investigadores han encontrado maneras eficientes de normalizar, como es el caso de la *symmetric normalization* empleada por Kipf y Welling (2017):

$$m_{\mathcal{N}(u)} = \sum_{v \in \mathcal{N}(u)} \frac{h_v}{\sqrt{|\mathcal{N}(u)||\mathcal{N}(v)|}} \tag{2.13}$$

Es así que se forma uno de los modelos de GNN más populares, Graph Convolutional Networks (GCN), que incluye *self-loops* y la *symmetric normalization*:

$$h_u^{(k)} = \sigma\left(W^{(k)} \sum_{v \in \mathcal{N}(u) \cup \{u\{} \frac{h_v}{\sqrt{|\mathcal{N}(u)||\mathcal{N}(v)|}}\right) \tag{2.14}$$

Este enfoque fue descrito por primera vez por Kipf y Welling (2017) y ha demostrado ser una de las arquitecturas GNN de referencia más populares y efectivas.

2.6.3 *Graph Pooling*

En los problemas de clasificación o regresión de grafos. Se emplea una o varias Graph Pooling Layers para generar un ”supernodo”que represente al grafo completo. Esto se logra haciendo operaciones entre los nodos que no dependan del orden o la cantidad de nodos del grafo. Por ejemplo, operaciones como la suma, el promedio y el máximo.

$$h_G = pool\left(h_1^k, h_1^k, ..., h_n^k\right) \tag{2.15}$$

2.6.4 *Differential Pooling*

Ying et al. (2018) nos propone una forma de reducir el grafo de una manera más escalonada, Figura N° 2.17. De acuerdo a los resultados de su investigación, aplicar este tipo de *Graph Pooling* mejora el rendimiento de los modelos en tareas de *Graph Classification* o *Graph Regression*.

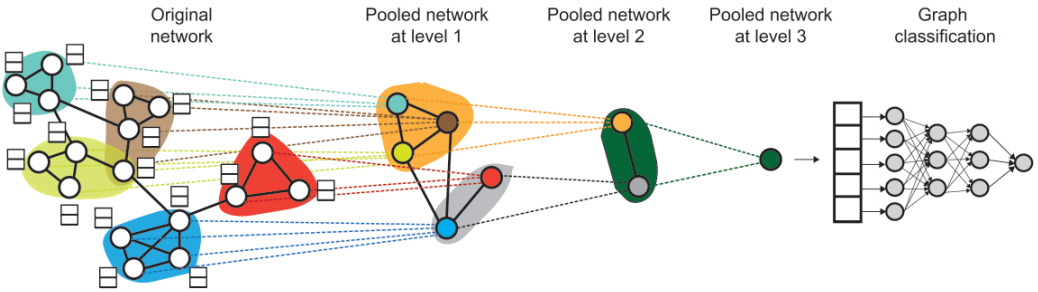


FIGURA N° 2.17: *Differential Pooling*. Adaptado de Ying et al. (2018)

CAPÍTULO III: CASO DE ESTUDIO

Para la calibración de un modelo de GNN que estime los parámetros hipocentrales a partir de los tiempo historia de la red sísmica, el primer paso es conseguir la base de datos que servirá para el entrenamiento del modelo. Luego, esa base de datos pasará por un preprocesamiento, dejándolo listo para ser ingresado al futuro modelo. Definiremos la arquitectura del modelo. Finalmente, se entrenará el modelo y se evaluarán los resultados. Todo el proceso se realizará en el lenguaje de programación Python.

3.1 BASE DE DATOS SÍSMICA

El primer paso es obtener una base de datos, en este caso, se usó el procedimiento realizado por van den Ende y Ampuero (2020). Esto se hizo para tener una línea base con la cual comparar los resultados del presente modelo.

Se usó la librería Obspy (Beyreuther et al., 2010) para descargar el inventario de estaciones sísmicas de banda ancha y el catálogo sísmico de *Southern California Seismic Network* (Hutton, Woessner, y Hauksson, 2010).

El área de estudio elegida fue en California. Precisamente, se definió una región cuadrada desde 32° hasta 36° grados en latitud, y desde -120° a -116° grados en longitud, tal y como se muestra en la Figura N° 3.1.

Se descargó las 3 direcciones (Norte-Sur, Este-Oeste y vertical) de los sismogramas registrados en el área mencionada, y que hayan ocurrido desde el año 2000 hasta el año 2015. Además, se escogió solo sismos con magnitud mayor o igual a 3; no se limitaron los sismos con respecto a su profundidad.

De esta manera, quedaron disponibles para su uso 1386 eventos sísmicos y 187 estaciones sismológicas.

3.2 PREPROCESAMIENTO DE LA DATA SÍSMICA

La base de datos pasó por un preprocesamiento. Siguiendo la metodología de van den Ende y Ampuero (2020), primero se quitó la respuesta del instrumento, luego se filtró de acuerdo a una pasa banda de 0.1 - 8 Hz. Luego, se interpoló las formas de onda en un tiempo base común de 1<t<101 segundos después del tiempo de origen del evento sísmico sobre 2048 muestras igualmente espaciadas, simulando así una frecuencia de muestreo de 20 Hz aproximadamente.

En el trabajo de van den Ende y Ampuero (2020) se explica que tomando una ventana de tiempo de 100 segundos debería asegurar que las ondas P de todos los eventos lleguen a ser registrados hasta por la más lejana estación sismológica

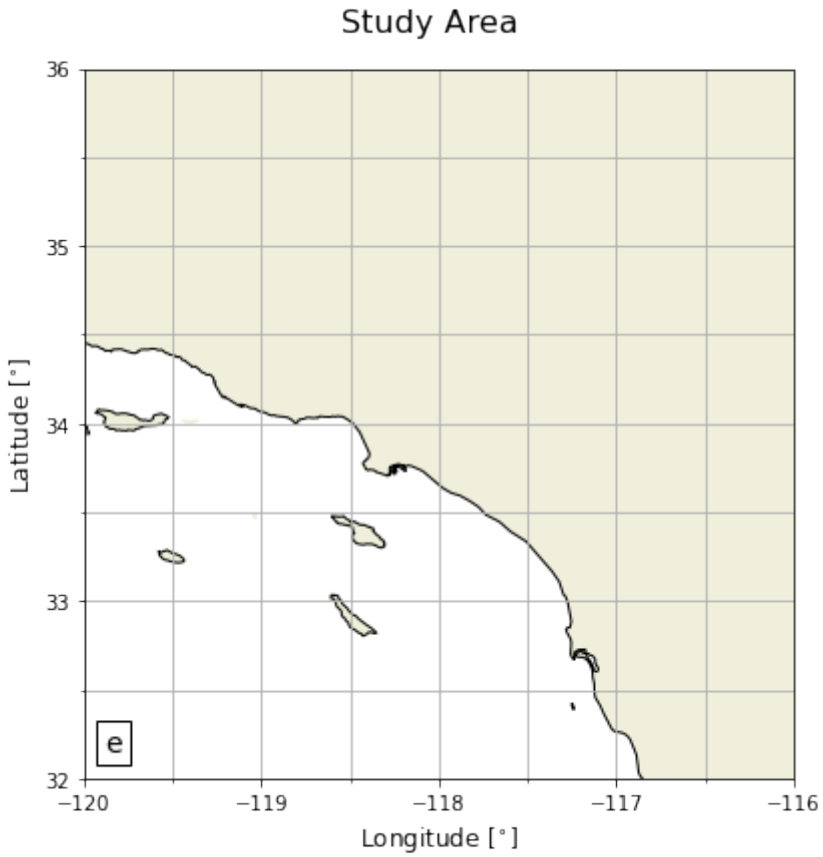


FIGURA N° 3.1: Área de estudio. Adaptado de van den Ende y Ampuero (2020)

dentro del área de estudio. A su vez, indica que el rango de frecuencias elegido asegura quedarnos con frecuencias que no se terminarán diluyendo.

Por otra parte, se normalizó los parámetros hipocentrales de los eventos sísmicos. La latitud y longitud se normalizó en un rango de +-1 empleando la latitud/longitud máxima/mínima de nuestra área de estudio. Así mismo, para normalizar la profundidad hipocentral, se tomó un mínimo de 0 km y un máximo de 30 km; mientras que, para la magnitud, se tomó un mínimo de 3 y un máximo de 6. La normalización es una buena práctica en el preprocesamiento de toda base de datos en Aprendizaje Automático, que apoya la estabilidad numérica del algoritmo.

El siguiente preprocesamiento a la base de datos viene propuesto por la presente investigación. Se dispuso de solo 50 estaciones para todo evento grabado. Estas estaciones son las que más eventos han registrado, en otras palabras, las que más formas de onda han recuperado. Esta medida se consideró debido a la cantidad grande de estaciones de las 187 estaciones originales, que no estuvieron operacionales durante el periodo de tiempo seleccionado (2000-2015), es decir, no grabaron ninguna señal.

Lo anterior implica que para cada evento, se tendrán exactamente 3 sismogramas (en cada dirección) por cada una de las 50 estaciones seleccionadas. En caso de que alguna de las estaciones seleccionadas no haya registrado algún evento, este se completa con un sismograma de ceros. Cabe mencionar, que además de los sismogramas, también se cuenta con la localización de cada una de las 50 estaciones seleccionadas, latitud y longitud normalizada de igual manera que los epicentros.

La idea de tener la información de esta manera es poder representarla en forma de grafos. Un evento sísmico es representado como un grafo. Se tendrá 4 nodos por cada estación: El nodo principal que encapsula la información espacial (latitud y longitud) de la estación y los 3 nodos virtuales que guardan los 3 sismogramas (cada uno con una dirección). Los nodos principales (estaciones) tiene enlaces con otras estaciones. Los enlaces fueron hechos análogamente al ”mesh”típico en elementos finitos, sin embargo, los enlaces superiores a 0.5° de distancia fueron descartados.

La idea detrás de esto es que los enlaces representen una relación de cercanía. A diferencia de van den Ende y Ampuero (2020), en esta investigación se considera que compartir información entre estaciones cercanas significará un beneficio para el mejor entendimiento de la estructura compleja de la red, por consiguiente, mejores predicciones del modelo. Los nodos virtuales (sismogramas) solo están conectados a su respectiva estación.

De esta manera se definen los grafos (eventos sísmicos), por ejemplo, en la Figura N° 3.2 se puede apreciar un grafo típico.

Esta investigación plantea darle un enfoque de aprendizaje supervisado, los grafos mencionados vendrían a ser nuestras muestras. Las etiquetas serán los parámetros hipocentrales de cada evento, es decir, latitud, longitud, profundidad y magnitud del evento. Ya que se van a predecir números reales trabajando sobre grafos completos, se convierte en un problema de *Graph Regression*.

Terminado el preprocesamiento, tenemos 1372 muestras cada uno con su respectiva etiqueta. Se dividirá la base de datos en 3 partes en la proporción de 90-5-5. El 90 % de la base de datos se destinó para el entrenamiento del modelo, 5 % para el set de validación, que sirve básicamente para optimizar los hiperparámetros del modelo y seleccionar el mejor modelo. El 5 % restante se destinó para el set de evaluación que sirve para evaluar el error de las predicciones en data nunca antes vista.

Por lo tanto, hay 1234 muestras para el set de entrenamiento, 69 para el set de validación y 69 para el set de evaluación. La distribución de las muestras se puede ver en la Figura N° 3.3.

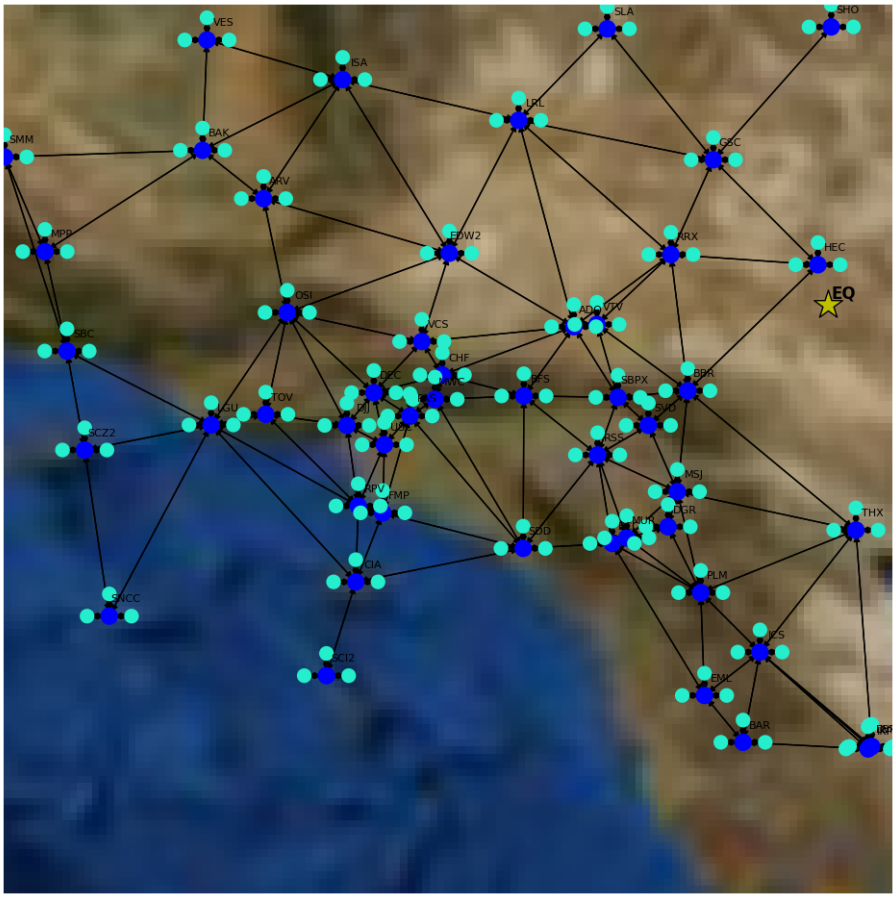


FIGURA N° 3.2: Grafo de 50 nodos y sus enlaces.

Ciertamente, 1234 muestras son muy pocas para entrenar un modelo basado en redes neuronales.

Una técnica frecuentemente usada en *Machine Learning* es conocida como *Data Augmentation*. Esta técnica consiste en generar data artificial a partir de pequeños cambios a las muestras originales. Dichos cambios o funciones aplicadas no deben alterar de manera brusca la integridad de las muestras originales, para así recrear una muestra artificial bastante representativa y real. Generalmente, esta técnica ayuda a evitar *overfitting*.

Un ejemplo sencillo con el cual se puede entender este concepto surge al trabajar con imágenes, modelos típicos de *Deep Learning* usualmente requieren de miles a millones de muestras para su entrenamiento, entonces es muy frecuente aplicar *Data Augmentation*. En el problema de clasificación de imágenes, se suelen hacer pequeñas rotaciones, traslaciones, cambiar la orientación o hasta jugar con la profundidad de los canales RGB de las imágenes originales.

En la Figura N° 3.4, se aprecia 6 imágenes nuevas a partir de una imagen original de un gatito . Dichas muestras nuevas han sido producto de algunas operaciones

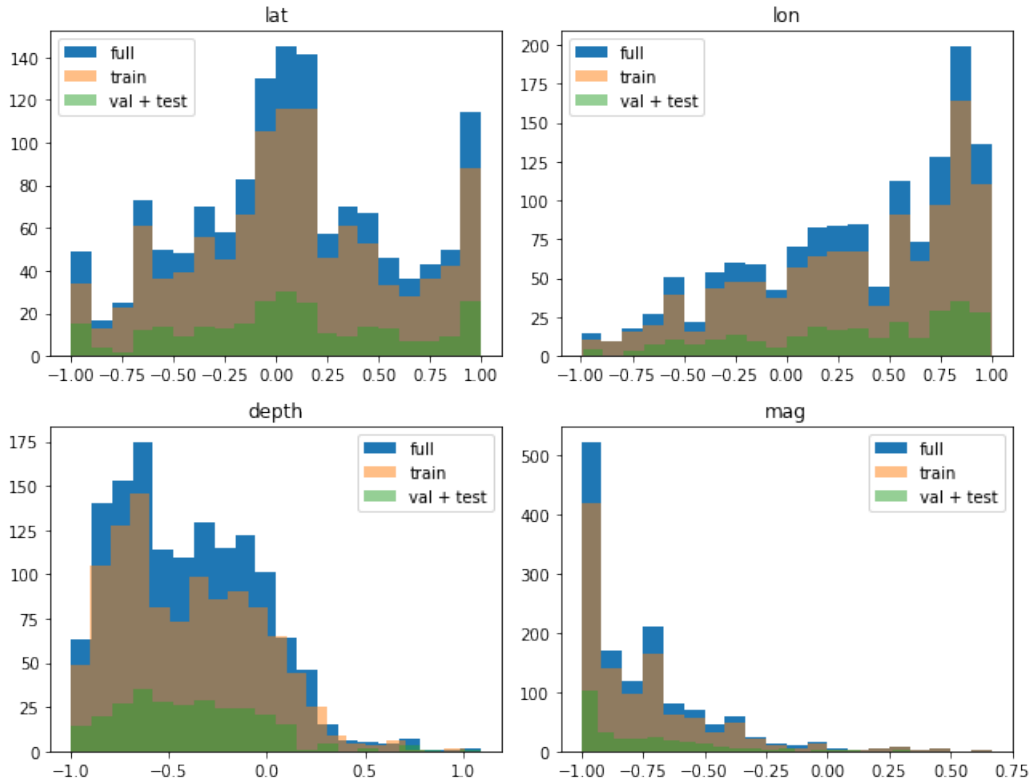


FIGURA N° 3.3: Distribución de los parámetros hipocentrales. Adaptado de van den Ende y Ampuero (2020)

matemáticas hechas sobre la original que, sin embargo, siguen representando claramente a la imagen de un gatito.

FIGURA N° 3.4: Data Augmentation. (Recuperado de Nanonets.com).

Siguiendo la misma línea, se incrementará el número de muestras (grafos) del set de entrenamiento. Para esto, la operación que se aplicará a cada grafo será un

desfase hacia la derecha de tamaño N en el dominio del tiempo a todos los canales de todas las estaciones. N es un número entero entre 20 y 200, eso significa que las señales se mueven 1 segundo a 10 segundos a la derecha (frecuencia de muestreo 20 Hz).

Hacemos 20 desfases para cada muestra. En la Figura N° 3.5, se aprecia la muestra original, junto con las muestras aumentadas.

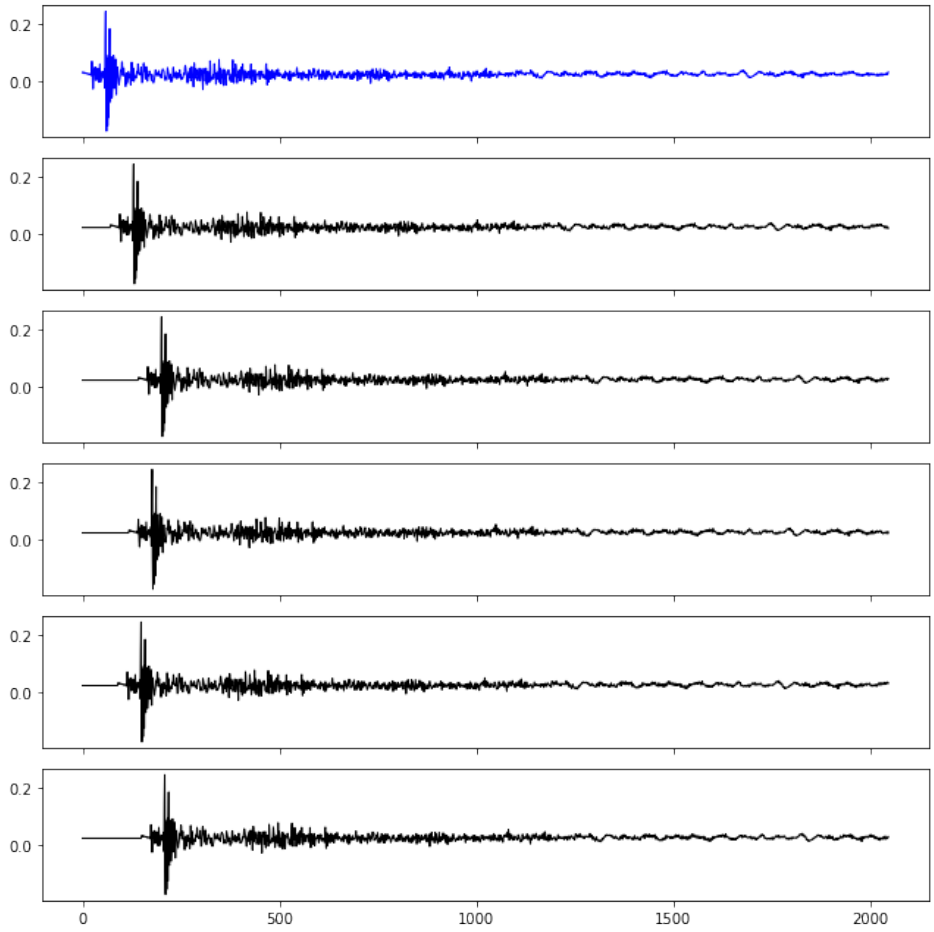


FIGURA N° 3.5: *Graph Augmentation*.

Finalmente, luego de aplicar *Data Aumentation*, se tendrán 24680 muestras en el set de entrenamiento.

3.3 ARQUITECTURA

El siguiente paso es definir la arquitectura del modelo.

La arquitectura se refiere a la estructura de nuestra red neuronal, es decir, que capas se usaron y la secuencia entre ellas.

Hay dos puntos importantes al momento de diseñar una arquitectura. La primera es asegurar que las operaciones entre tensores sean dimensionalmente correctas

de principio a fin. En este caso, el modelo recibe un *batch* de grafos en cada iteración, cada grafo está compuesto por una serie de tensores relacionados entre si tal y como fue explicado en la sección anterior. El modelo (arquitectura + pesos) debe ser capaz de procesar estos datos a través de sus capas para finalmente dar como salida un *batch* de tensores de 4 elementos (latitud, longitud, profundidad y magnitud).

El segundo punto es entender que a partir de una arquitectura podemos tener infinitos modelos. Cada modelo difiere de otro por los valores de sus parámetros o pesos. El mejor modelo usualmente es aquel que reduce el error al mínimo, en otras palabras, el que predice mejor. La manera de calcular los mejores pesos es a través del entrenamiento, usando *backpropagation*.

Teniendo en cuenta lo anterior, la Figura N° 3.6 muestra el esquema de la arquitectura.

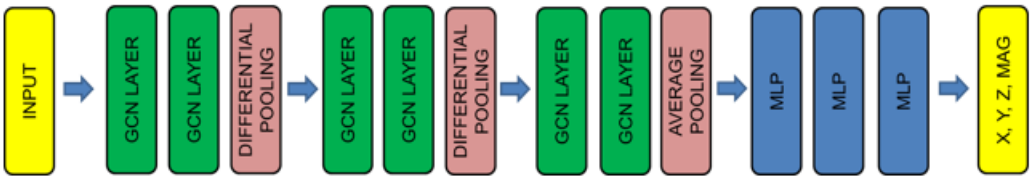


FIGURA N° 3.6: Arquitectura del modelo.

De manera más detallada, la arquitectura se estructura de la siguiente manera. Se considera por facilidad de explicación que se trabajará con *batches* de tamaño 1, es decir, se recibirá un solo grafo por iteración. Cabe mencionar que en realidad el *batch size* es superior a 1, y es un hiperparámetro del modelo que se optimiza con ayuda del set de validación.

Cada iteración se puede dividir en 4 partes: En la primera parte, se recibe un grafo, conformado por la matriz de *node features* X de dimensión 200×2048 (200, debido a las 50 estaciones principales más las 150 estaciones virtules; 2048 es la dimensión de cada sismograma) y su matriz de adyacencia A de dimensión 200×200. Esta pasa por 2 capas de convolución *Graph Convolutional* (Kipf y Welling, 2017), que no es más que un *message passing* con un tipo especial de normalización, logrando compartir información entre nodos de un mismo vecindario. Además, en cada convolución se reducirá la dimensión de los *node features* a la mitad (ejemplo, de 2048 a 1024). Por lo tanto, luego de dos convoluciones, la matriz de *node features* X tendrá dimensión 200×512 y su matriz de adyacencia A seguirá siendo de dimensión 200×200. Para terminar con la primera parte, se irá reduciendo la cantidad de nodos de manera escalonada usando *Differential Pooling* de Ying et al. (2018). Se reduce la cantidad de nodos a la cuarta parte, quedando la matriz de *node features* X de dimensión 50×512 y su matriz de adyacencia A seguirá siendo de dimensión 50×50.

La segunda parte, se repite el procedimiento de la primera parte, quedando la matriz de *node features* X de dimensión 13×128 y su matriz de adyacencia A seguirá siendo de dimensión 13×13.

En la tercera parte, se harán dos convoluciones, en la primera se reducirá la dimensión de la matriz de *node features* X a la mitad, y en la segunda se mantendrá la dimensión. Para la *pooling layer*, se reducirán los 13 nodos a 1 solo nodo usando el promedio en la segunda dimensión. Por lo tanto, la matriz de *node features* X será de dimensión 1×64. Este único nodo representa al grafo completo.

Finalmente, en la cuarta y última parte, el nodo o vector resultante del paso anterior es ideal para ser pasado por una MLP. Esta red cuenta con 3 capas simples (*Fully Conected Layers*), dando como salida un vector resultante Y de dimensión 1×4. El vector resultante representa la predicción del modelo, es decir, los 4 parámetros hipocentrales.

La arquitectura fue hecha en Python, particularmente, se usó la librería Pytorch y su extensión Pytorch Geometric. Pytorch ofrece una caja de herramientas para desarrollar proyectos de *Deep Learning* y su extensión para trabajar con grafos. En la Figura N° 3.7, podemos ver parte del código para definir la arquitectura, precisamente, las capas que la componen. El código de la arquitectura lo podemos encontrar en el Apéndice A. El código completo se encuentra en el repositorio GitHub de AngelMC.

```
28  class my_network(torch.nn.Module):
29
30      def __init__(self, num_nodes):
31          super(my_network, self).__init__()
32
33          self.conv1 = DenseGCNConv(2048, 1024)
34          self.conv2 = DenseGCNConv(1024, 512)
35          num_nodes = ceil(0.25 * num_nodes)
36          self.pool1 = DiffPool(512, num_nodes)
37
38          self.conv3 = DenseGCNConv(512, 256)
39          self.conv4 = DenseGCNConv(256, 128)
40          num_nodes = ceil(0.25 * num_nodes)
41          self.pool2 = DiffPool(128, num_nodes)
42
43          self.conv5 = DenseGCNConv(128, 64)
44          self.conv6 = DenseGCNConv(64, 64)
45
46          self.linear1 = Linear(64, 32)
47          self.linear2 = Linear(32, 16)
48          self.linear3 = Linear(16, 4)
```

FIGURA N° 3.7: Código de la Arquitectura.

Entonces, en cada iteración hacemos una predicción, la cual se compara con la etiqueta del evento y produce un error. El algoritmo intentará reducir el error, usará *backpropagation* y actualizará los pesos del modelo. En resumen, el modelo se entrenará. Este proceso se explicará con más detalle en la siguiente sección.

3.4 ENTRENAMIENTO Y EVALUACIÓN

Una vez definida la arquitectura, ya se puede comenzar con el entrenamiento y la evaluación del modelo.

El entrenamiento es un proceso iterativo donde el modelo aprende del set de entrenamiento. Al ser aprendizaje supervisado, tenemos etiquetas que guiarán al modelo en la dirección correcta. En un comienzo las predicciones serán aleatorias, ya que los pesos iniciales del modelo se inicializan aleatoriamente (en realidad, hay estudios que proponen inicializaciones especiales que aseguren la estabilidad numérica).

Con el paso de las iteraciones, los pesos toman valores óptimos y el error se reduce. En algún momento, el error no se reduce más, el modelo ha llegado a su máxima capacidad. Esto no es algo negativo si tenemos en cuenta el problema que queremos resolver y el nivel de precisión que este requiere. Por ejemplo, en el problema de clasificación de imágenes, no es lo mismo estar 80 % seguros de que cierta imagen es un gato o no, que estar 80 % seguros de haber detectado un tumor maligno o no en una imagen médica.

Entonces, terminado el entrenamiento, debemos evaluar el modelo. Para esto se ha reservado una parte de la base de datos que no ha sido tocada en el entrenamiento, llamada set de evaluación. Si se tiene suerte, el modelo hace predicciones tan buenas como en el entrenamiento. Esto quiere decir que el modelo no solamente ha memorizado, sino que realmente ha aprendido.

La pregunta es, ¿qué pasaría si no tenemos suerte o el modelo no cumple con el nivel de rendimiento requerido? No pasa nada, se pueden hacer un par de cosas:

Podemos realizar modificaciones en la arquitectura que aumenten la capacidad de nuestro modelo. Hay que tener cuidado con no excederse en la complejidad añadida, ya que podría mejorar el rendimiento en el set de entrenamiento, mas no en el set de evaluación. Esto se conoce como *overfitting* o sobreajuste, significa que el modelo solamente está memorizando el set de entrenamiento, pero al momento de mostrarle data nunca antes vista, su rendimiento se reduce. Lo contrario se llama *underfitting* y significa que el modelo no es lo suficientemente inteligente para aprender de los datos, por lo que tiene un mal rendimiento incluso en el set de entrenamiento. Lo más común para solucionar *underfitting* es aumentar la capacidad del modelo añadiendo más capas. En la Figura N° 3.8 se puede apreciar una

comparación de los casos que pueden darse a la hora de seleccionar un modelo.

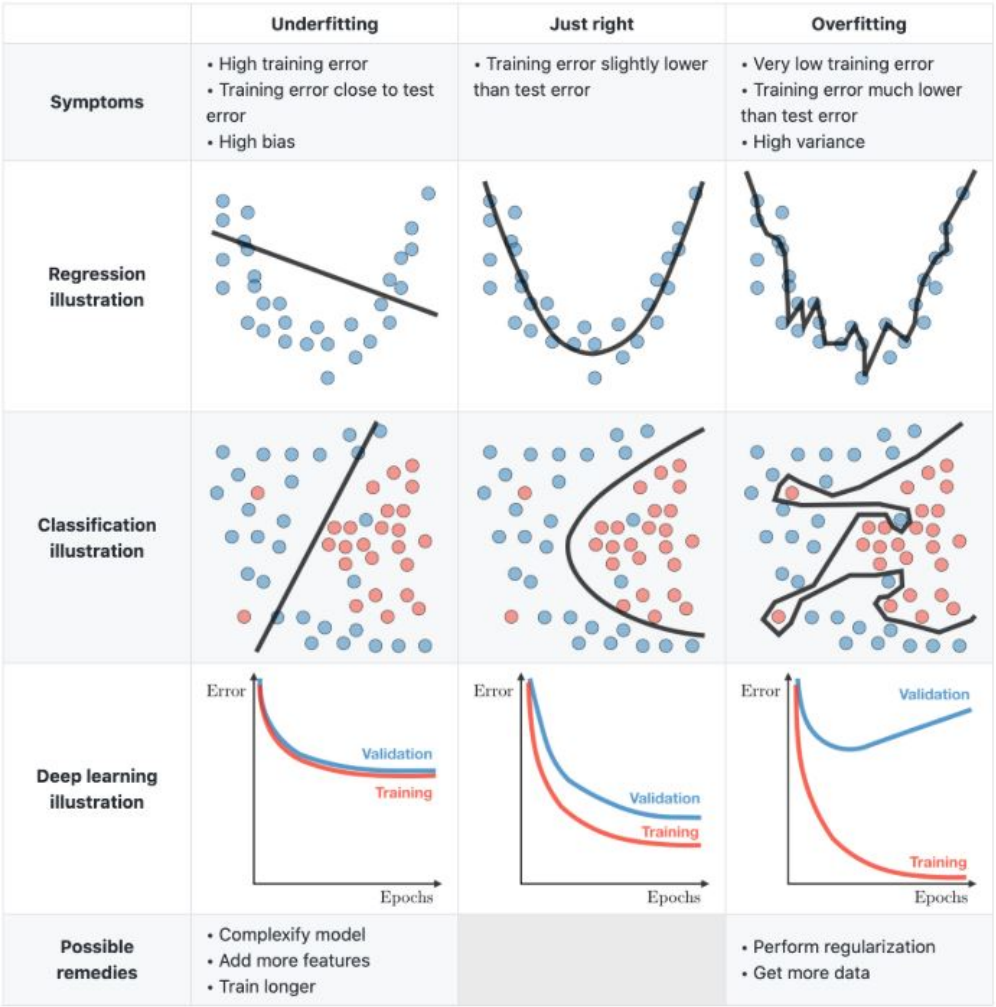


FIGURA N° 3.8: Selección del modelo. Comparación de *underfitting* vs buen ajuste vs *overfitting*. (Recuperado de Kaggle).

También se puede modificar los hiperparámetros del algoritmo. Son llamados así ya que estos parámetros especiales son elegidos por el investigador previamente al entrenamiento, usando su pericia, experiencia y hasta intuición.

Hechas las modificaciones, se puede volver a entrenar el modelo. Si todo sale bien, se tendrá un buen rendimiento en el set de entrenamiento. Ahora, al probar nuevamente con los datos reales o set de evaluación. De no conseguir los resultados deseados, se debe volver a empezar.

Pongámonos en la situación que finalmente las modificaciones surgieron efecto y se tiene un buen rendimiento tanto en el set de entrenamiento como en el set de evaluación. Entonces, se selecciona ese modelo, se lleva a producción y que trabaje en el mundo real. Lo más probable es que nos llevemos la desagradable

sorpresa que el modelo no rinde como lo hacía en su etapa de desarrollo. A pesar de que supuestamente usamos data real reservada para la evaluación.

Esto se explica por el hecho de haber optimizado los hiperparámetros y la arquitectura de acuerdo al rendimiento obtenido en el set de evaluación, lo que inconscientemente generó un sesgo o *bias*.

Finalmente, para evitar el efecto anterior, se acostumbra a reservar una parte de la base de datos para optimizar los hiperparámetros, conocida como set de validación. Al igual que el set de evaluación, esta porción no se usa para el entrenamiento.

Se aplicará los conocimientos aprendidos para el entrenamiento y evaluación.

El rendimiento del modelo se evalúa a través de una pérdida de error absoluto medio (o MAE, por sus siglas en inglés) entre las predicciones del modelo y las etiquetas, y el entrenamiento se realiza minimizando la pérdida utilizando el optimizador ADAM (Kingma y Ba, 2014). El *learning rate* será 0.00005 y el *batch size* será de 64 muestras. El entrenamiento procede durante 2500 épocas, momento en el que el rendimiento del modelo se ha saturado. Usando la computadora de IRIDES de la Universidad de Tohoku, la fase de entrenamiento duró aproximadamente 12 horas en total.

En la Figura N° 3.9, se observa el *loop* de entrenamiento. El entrenamiento y validación se hacen al mismo tiempo. El código completo puede encontrarse en el Apéndice A.

```
for epoch in range(epochs):
    loss = train()
    train_mse = test(train_loader)
    val_mse = test(val_loader)
    #scheduler.step()
    hist["loss"].append(loss)
    hist["Train_MSE"].append(train_mse)
    hist["Val_MSE"].append(val_mse)
    print(f'Epoch: {epoch+1:03d}, Loss: {loss:.4f}, Train MSE: {train_mse:.4f}, Val MSE: {val_mse:.4f}')
```

FIGURA N° 3.9: *Loop* de entrenamiento.

Finalizado el entrenamiento, en la Figura N° 3.10 vemos como el error fue disminuyendo con el paso de las épocas. La línea azul de entrenamiento no se encuentra tan lejos de la línea naranja de validación. Por lo que podemos inferir que el modelo aprendió.

Por último, se entrega el set de evaluación al modelo entrenado. En un problema de regresión, el éxito se logra si el error es mínimo, en nuestro caso, nuestra métrica principal es MAE. A continuación, se muestra la Tabla N° 3.1 con los resultados de MAE para cada parámetro, además, se convierten los resultados en grados a kilómetros (1° ≈111 km).

Para explorar más los resultados, se han planteado las siguientes ilustraciones que nos dan diversas perspectivas del rendimiento del modelo.

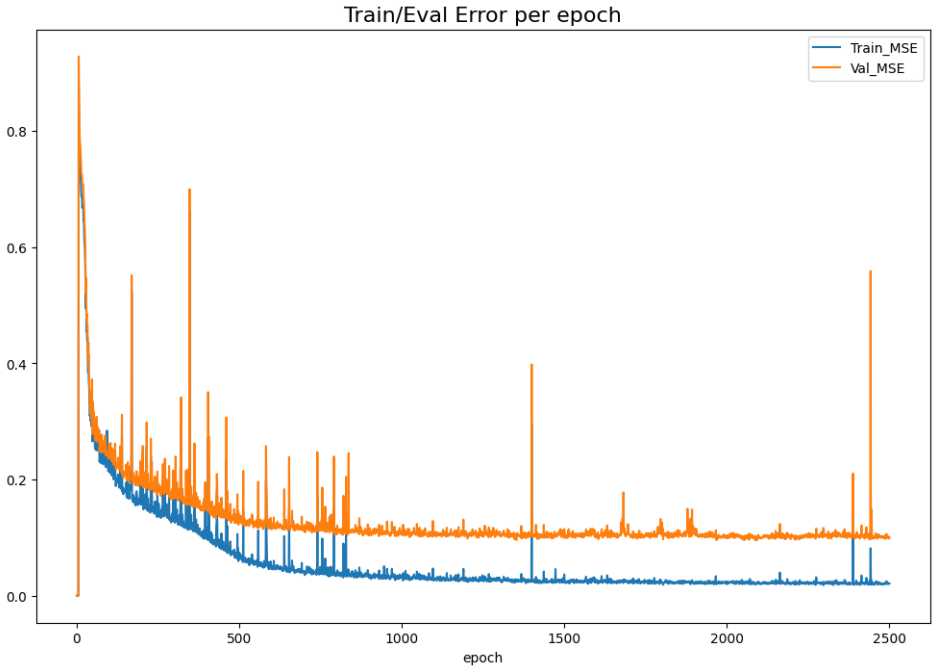


FIGURA N° 3.10: Historia del error.

TABLA N° 3.1: Error absoluto medio (MAE) para cada parámetro hipocentral.

Latitud	Longitud	Profundidad	Magnitud
0.155°	0.150°	2.56 km	0.130
17.18 km	16.60 km		

En la Figura N° 3.11, se muestra la curva de densidad de los errores de predicción. Esta figura muestra la probabilidad con la que se repite cierto error para los 4 parámetros hipocentrales. De la figura, podemos intuir que mientras más acumulación haya en el centro, es más probable que ocurra un error cercano a cero.

En la Figura N° 3.12, comparamos el valor predicho por el modelo con el valor real proveniente del catálogo de la SCSN. Los puntos acumulados cerca de la función identidad nos habla de un buen rendimiento del modelo.

Finalmente, en la Figura N° 3.13, en el área de estudio, dibujamos flechas. Cada flecha va desde el epicentro predicho por el modelo y apunta al epicentro registrado en el catálogo. Las flechas más pequeñas indican que el modelo fue capaz de predecir con gran exactitud el epicentro del evento sísmico.

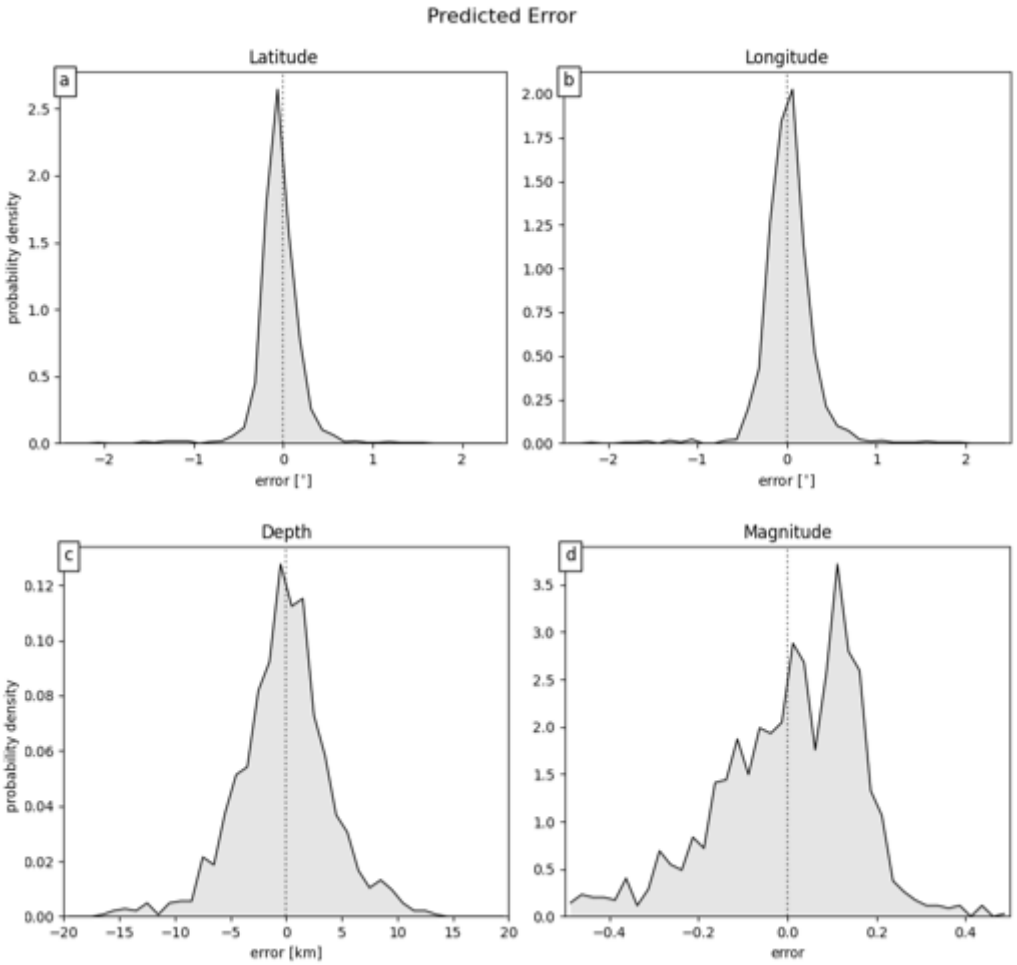


FIGURA N° 3.11: Curva de densidad del error de predicción.

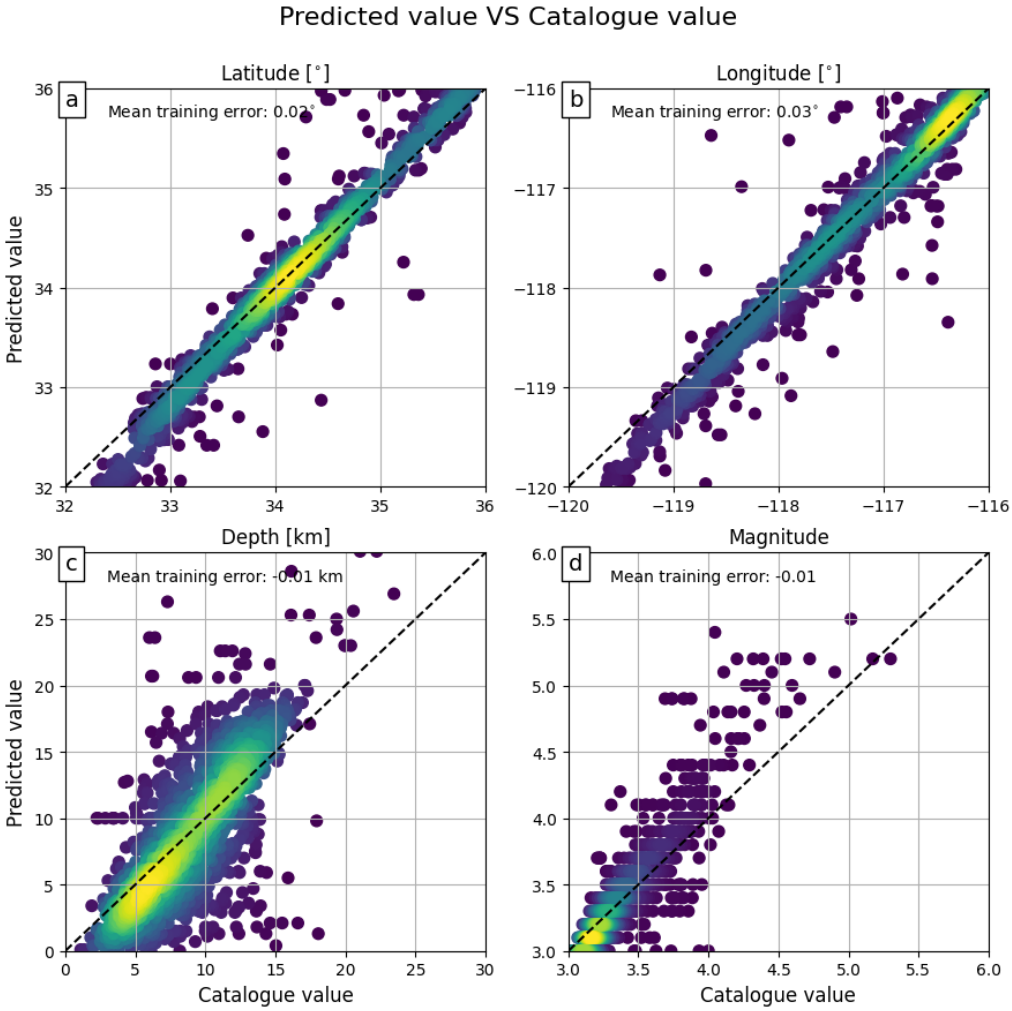


FIGURA N° 3.12: Valor predicho vs Valor del catálogo.

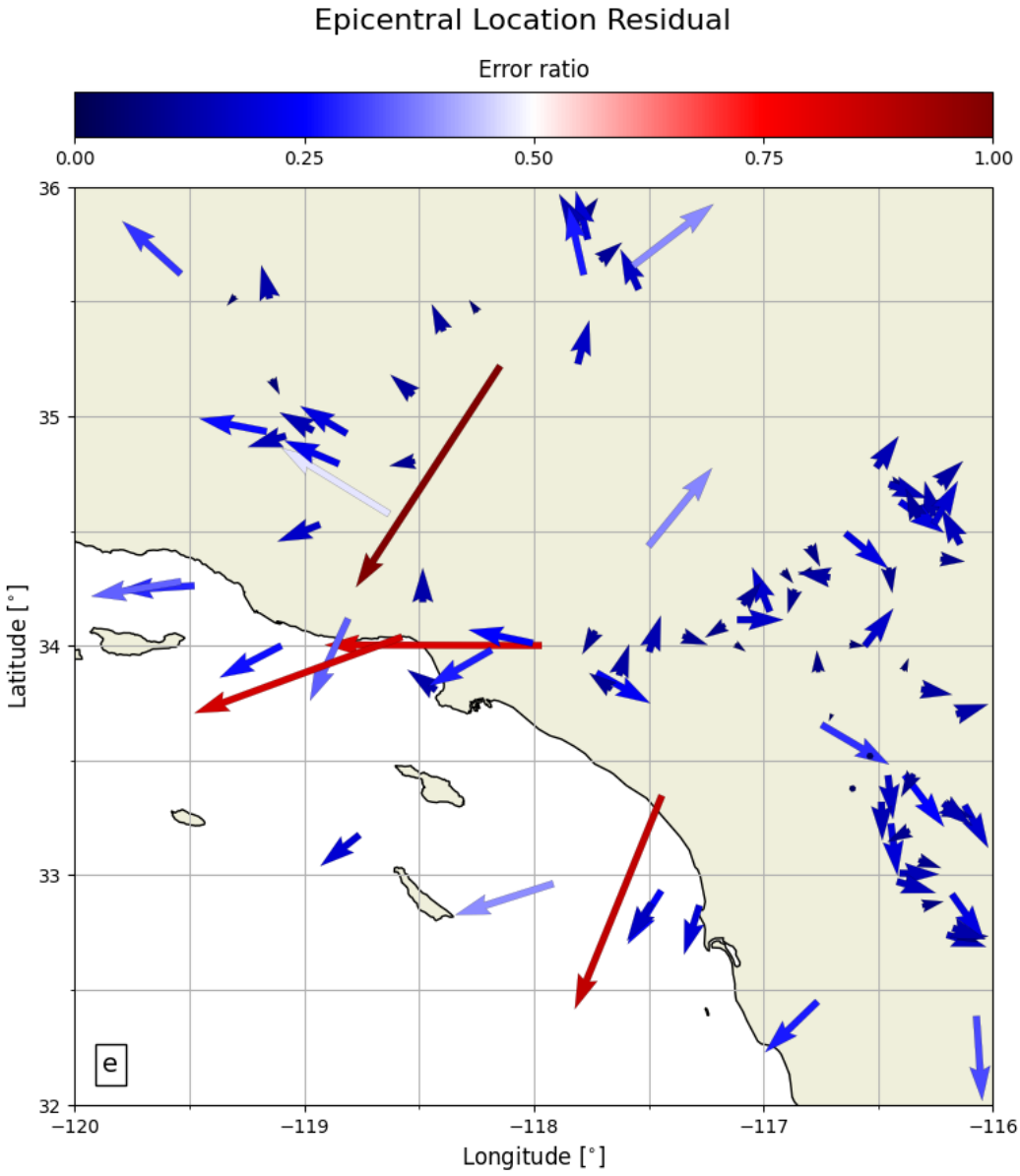


FIGURA N° 3.13: Distancia residual epicentral.

CAPÍTULO IV: DISCUSIÓN DE RESULTADOS

A continuación, se procederá a analizar los resultados de la evaluación del modelo para cada uno de los parámetros hipocentrales.

4.1 ESTIMACIÓN DE LA MAGNITUD

Con base en el modelo propuesto en esta investigación y la base de datos explicada en el Capítulo 3, se realizó la estimación de la magnitud.

La distribución estadística de las magnitudes de los eventos de la presente base de datos fue presentada en la Figura N° 3.3 del Capítulo 3. Esta distribución permite observar que las magnitudes de los eventos con los que se cuenta no tienen una distribución uniforme, teniéndose muchas magnitudes pequeñas y muy pocas de gran magnitud. Esta distribución desigual es una condición natural de esta variable, por lo que no es posible obtener un muestreo homogéneo en un periodo de tiempo dado.

Teniendo en cuenta esto, el modelo se entrenó considerando una base de datos desbalanceada, algo no recomendado en *Machine Learning*, sin embargo, es una cualidad que tiene que ser asumida y gestionada por el modelo mismo ya que es la manera en que esta variable se presenta en el mundo real.

La métrica para evaluar el rendimiento del modelo es el error absoluto medio (MAE, por sus siglas en inglés). En el set de evaluación, el valor de MAE es de 0.13 unidades de magnitud, lo cual resulta ser una muy buena precisión considerando los retos que enfrentó como el trabajar con una base de datos desbalanceada y sin la intervención de un sismólogo analista experto que selecciona las ondas principales.

Desde otra perspectiva, en la Figura N° 3.11, podemos interpretar que el error que se repite más veces está alrededor de cero con un pequeño sesgo hacia la derecha. Considerando que el error es calculado como el valor predicho menos el valor del catálogo, el modelo está sobreestimando la magnitud real de muchos de los eventos. Este fenómeno también puede ser observado en la Figura N° 3.12, donde para eventos de mayor magnitud, los puntos se ubican a la izquierda de la función identidad.

El modelo por su cuenta está tratando de manejar la data desbalanceada. El modelo no cuenta con mucha información de eventos con grandes magnitudes de la cual aprender, al compensar el error este tiende a sobreestimar.

4.2 ESTIMACIÓN DE LAS COORDENADAS EPICENTRALES

Con base en el modelo propuesto en esta investigación y la base de datos explicada en el Capítulo 3, se realizó la estimación de la latitud y longitud del epicentro del evento.

La distribución estadística de las latitudes y longitudes de los eventos de nuestra base de datos fue presentada en la Figura N° 3.3 del Capítulo 3. Esta distribución permite observar que gran parte de los eventos se aglomera en la parte centro y este del área de estudio. Esta distribución es natural debido a la presencia de fallas en esa área de California (Figura N° 4.1), por lo que no es posible obtener un muestreo homogéneo de los epicentros.

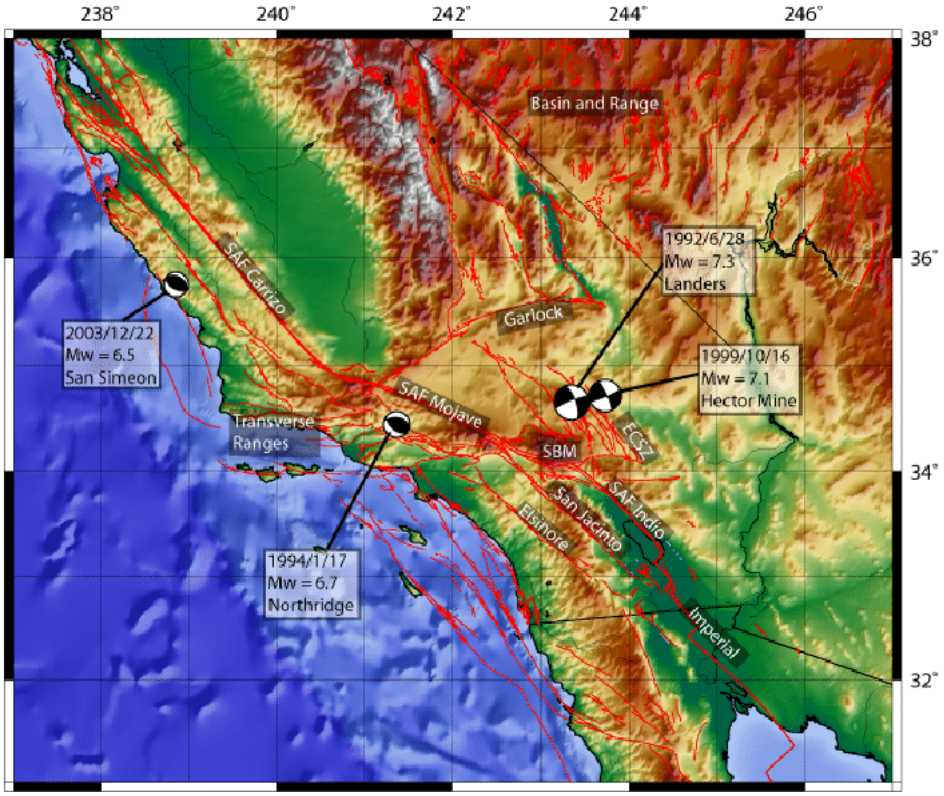


FIGURA N° 4.1: Fallas en el área de estudio. Recuperado de USGS.

Teniendo en cuenta esto, el modelo se entrenó considerando una base de datos desbalanceada, algo no recomendado en *Machine Learning*, sin embargo, es una cualidad que tiene que ser asumida y gestionada por el modelo mismo ya que es la manera en que estas variables se presentan en el mundo real.

En el set de evaluación, los valores de MAE son de 0.155° (17 km) y 0.15° (16.6 km) grados sexagesimales para la latitud y longitud respectivamente, lo cual resulta en una precisión aceptable para un primer reporte sísmico de manera rápida.

El valor de MAE para van den Ende y Ampuero (2020) es de 0.11° (13 km), lo cual

se encuentra cerca de los resultados de esta investigación. Si bien esta metodología emplea una muy simple arquitectura, el modelo fue capaz de obtener buenos resultados sin tanto ensayo y error.

En la Figura N° 4.2, se oberva la frecuencia en los errores para ambas variables. En ambos casos se muestran picos muy delgados en el cero. Esto confirma que el modelo pudo demostrar lo aprendido en data completamente nueva como el set de evaluación.

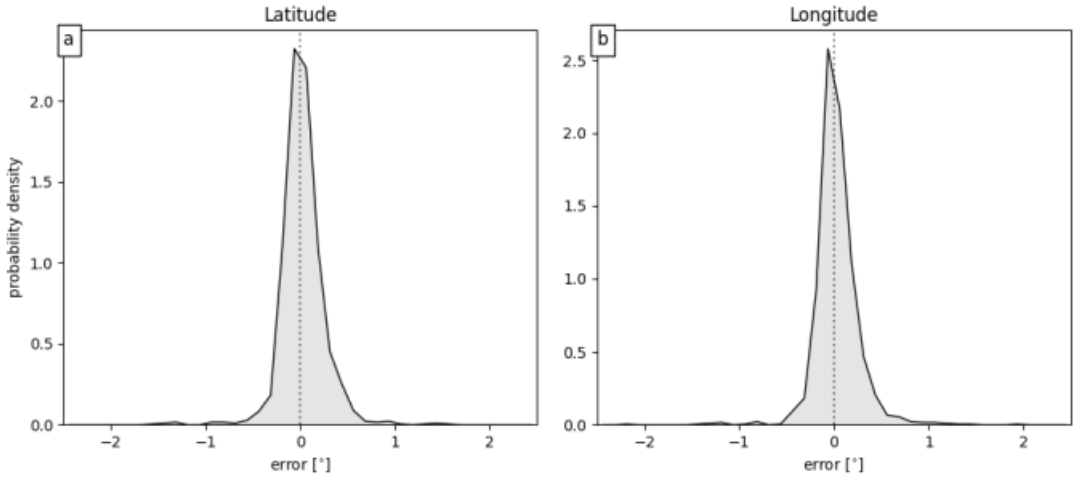


FIGURA N° 4.2: Curva de densidad del error en las coordenadas epicentrales.

Esto también puede ser observado desde la Figura N° 3.12, donde los puntos siempre están cerca de la función identidad.

Finalmente, en la Figura N° 3.13, se dibujan flechas en el área de estudio. Cada flecha va desde el epicentro predicho por el modelo y apunta al epicentro real del evento. Las flechas más pequeñas indican que el modelo fue capaz de predecir con gran exactitud el epicentro del evento sísmico y viceversa. Las flechas pertenecen a las predicciones en el set de evaluación. Debido a la mayor cantidad de datos en la zona central y este, las flechas más pequeñas se acumulan ahí. En la zona noroccidental, se observan flechas un poco más largas, la mayoría apuntando hacia afuera. Esto quiere decir que el modelo tiende a predecir dichos epicentros más al centro de lo que en realidad es, esto puede deberse a la base de datos no está igualmente distribuida en el área de estudio.

4.3 ESTIMACIÓN DE LA PROFUNDIDAD HIPOCENTRAL

Con base en el modelo propuesto en esta investigación y la base de datos explicada en el Capítulo 3, se realizó la estimación de la profundidad hipocentral.

La distribución estadística de las profundidades de los eventos de nuestra base de datos fue presentada en la Figura N° 3.3 del Capítulo 3. Esta distribución es

bimodal, presentando dos picos para profundidades pequeñas y medias. Esta distribución es natural por la geología en esa área de California (Figura N° 4.1), por lo que no es posible obtener un muestreo homogéneo de las profundidades.

Teniendo en cuenta esto, el modelo se entrenó considerando una base de datos desbalanceada. Esta una característica que tiene que ser asumida y gestionada por el modelo mismo dado que es la manera en que esta variable se presenta en el mundo real.

En el set de evaluación, el valor de MAE es de 2.56 km, lo cual resulta en una precisión aceptable para un primer reporte de manera rápida.

El valor de MAE para van den Ende y Ampuero (2020) es de 3.3 km. Eso significa que el presente modelo pudo mejorar un poco los resultados en cuanto a profundidad hipocentral.

En la Figura N° 3.11, se observa la frecuencia en los errores para ambas variables. Se muestra un pico delgado alrededor del cero. Esto confirma que el modelo pudo demostrar lo aprendido en data completamente nueva como el set de evaluación. Esto también puede ser observado desde la Figura N° 3.12, donde los puntos se encuentran relativamente cerca de la función identidad, aunque con cierta dispersión.

CONCLUSIONES

En el presente trabajo se ha empleado un modelo de Redes Neuronales (*Graph Neural Networks*) para estimar los parámetros hipocentrales de un evento sísmico a partir de registros de una red sísmica. Cabe resaltar que la estimación rápida y preliminar de dichos parámetros hipocentrales resulta útil en el proceso de generación de mapas de distribución de intensidades en zonas urbanas altamente sísmicas. Esto será de particular interés para el proyecto que viene desarrollando el Centro Peruano Japonés de Investigaciones Sísmica y Mitigación de Desastres (CISMID-FIC-UNI) con instituciones técnicas japonesas, en el cual se busca la pronta estimación de daños por sismo en Lima Metropolitana. A continuación, se presentan los principales hallazgos de este trabajo:

1. La literatura actual sobre *Graph Neural Networks* permite hacer predicciones a partir del análisis de grafos completos, por lo cual, es aplicable a la caracterización de la fuente sísmica, donde la información que entra tiene la forma de un grafo.

2. La base de datos fue obtenida de la Red Sísmica del Sur de California, recopilándose un total de 1372 eventos sísmicos divididos en 3 para el entrenamiento, validación y evaluación. Para el contexto de *Deep Learning*, esta cantidad es muy poca para el entrenamiento, por lo que en este trabajo se propuso una técnica especial de *Data Augmentation* con el fin de aumentar la cantidad de eventos disponibles.

3. El modelo fue probado para diferentes cantidades de estaciones (o nodos), siendo el modelo de 50 estaciones el que dio mejores resultados. Esta cantidad de estaciones es óptima y viable para la Red Sísmica de California.

4. El modelo propuesto dio predicciones con un error absoluto medio (MAE) de 0.13 en magnitud al momento de la evaluación, siendo este un resultado bastante bueno y equiparable con los resultados de otros estudios.

5. La data desbalanceada afecta las predicciones en magnitud, provocando sobreestimaciones a eventos de mayor magnitud, esto es debido a la poca cantidad de eventos de gran magnitud.

6. Los resultados obtenidos muestran una buena precisión en la ubicación del epicentro considerando lo simple de la arquitectura, la base de datos desbalanceada usada en este estudio y sin una rigurosa optimización de hiperparámetros del algoritmo. Los errores son de unos cuantos kilómetros (17.18 km en latitud y 16.6 km en longitud) en promedio. Valores que pueden ser suficientes para reportes sísmicos preliminares.

7. Las predicciones en profundidad, con un error absoluto medio (MAE) de 2.56 km, muestran una mejoría con respecto a estudios pasados.

8. Esta investigación proporciona epicentro y magnitud de un evento sísmico de manera rápida (100 segundos). Dicha información resulta útil en el proceso de generación de mapas de intensidad realizado por el CISMID-FIC-UNI.

9. La aplicación de esta metodología al Perú es materia de futuros estudios. Los principales retos son la poca cantidad de estaciones, la baja densidad de las redes sísmicas en Perú y la poca cantidad de eventos registrados que sirvan para entrenar un modelo en Perú. Estos retos podrían encararse con ideas como mejorar la arquitectura del modelo para que funcione bien usando menos estaciones, aumentar la base de datos de entrenamiento usando sismos sintéticos, o complementando nuestra base de datos con sismos de redes de otros países como Chile o Japón, considerando la similitud en el mecanismo de ruptura (subducción) de sus sismos.

10. De ser posible la obtención de datos sismográficos adecuadamente distribuidos en Lima y provincias cercanas, se propondrá la aplicación de la presente metodología al proyecto de elaboración de mapa de intensidades del CISMID-FIC-UNI.

RECOMENDACIONES

1. Con referencia en la base de datos de la red sísmica del sur de California utilizada en la presente tesis y pensando en su aplicación en Lima, se recomienda la instalación de un mayor número de estaciones acelerográficas en Lima Metropolitana y provincias cercanas, considerando una distribución más homogénea y evitando su concentración únicamente en la zona metropolitana.

2. En la búsqueda de la implementación de metodologías de este tipo en la Red Acelerográfica del CISMID (REDACIS), se sugiere realizar investigaciones que utilicen señales de aceleración en lugar de aquellas de velocidad, como las implementadas en esta tesis. Lo cual podría resultar en una adecuada estimación de los parámetros hipocentrales para eventos de campo cercano e intermedio, dada la naturaleza de los registros producto de instrumentos de aceleración para este tipo de eventos.

3. Revisar constantemente las novedades en las herramientas aplicables a *Graph Neural Networks* que puedan mejorar la arquitectura, en específico, en la tarea de predecir a partir de grafos enteros.

4. Revisar en la literatura los avances con respecto al preprocesamiento de la base de datos sísmica y técnicas nuevas de *Data Augmentation* para eventos sísmicos.

5. Buscar bases de datos lo más balanceada posible, es decir, eventos con magnitudes y ubicaciones lo más homogenea posible. Otra buena opción a estudiarse podría ser la generación de sismogramas sintéticos.

REFERENCIAS BIBLIOGRÁFICAS

Alva, J., y Castillo, J. (1993). Peligro Sísmico en el Perú. En *VII Congreso Nacional de Mecánica de Suelos e Ingeniería de Cimentaciones.* Lima, Perú.

Bao, H., Ampuero, J.-P., Meng, L., Fielding, E. J., Liang, C., Milliner, C. W., … Huang, H. (2019). Early and persistent supershear rupture of the 2018 magnitude 7.5 Palu earthquake. *Nature Geoscience*, *12*(3), 200–205.

BBC News Mundo. (2022). *Tonga: el grave derrame de petróleo en la costa de Perú a causa de la erupción del volcán en la nación insular.* Descargado de https://www.bbc.com/mundo/noticias-60064916

Beyreuther, M., Barsch, R., Krischer, L., Megies, T., Behr, Y., y Wassermann, J. (2010). ObsPy: A python toolbox for seismology. *Seismological Research Letters*, *81*(3), 530–533. doi: 10.1785/gssrl.81.3.530

Cajal, S. R. (1899). *Comparative Study of the Sensory Areas of the Human Cortex*. Clark University. Descargado de https://books.google.com.pe/books?id=2Dv-zWg89tsC

Calvo, D. (2019). *Aprendizaje Supervisado.* Descargado de https://www.diegocalvo.es/aprendizaje-supervisado/

Chen, C., y Holland, A. A. (2016). PhasePApy: A robust pure Python package for automatic identification of seismic phases. *Seismological Research Letters*, *87*(6), 1384–1396. doi: 10.1785/0220160019

Cho, H., y Choi, I. S. (2019). Deep Learning Algorithm of Graph Convolutional Network: A Case of Aqueous Solubility Problems. *Bulletin of the Korean Chemical Society*, *40*(6), 485–486.

Cyranoski, D. (2011). Japan's tsunami warning system retreats. *Nature*. Descargado de https://doi.org/10.1038/news.2011.477 doi: 10.1038/news.2011.477

Géron, A. (2019). *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*. O'Reilly Media.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., y Dahl, G. E. (2017). Neural message passing for quantum chemistry. *34th International Conference on Machine Learning, ICML 2017*, *3*, 2053–2070.

Hamilton, W. L. (2020). Graph representation learning. *Synthesis Lectures on Artifical Intelligence and Machine Learning*, *14*(3), 1–159.

Hutton, K., Woessner, J., y Hauksson, E. (2010). Earthquake monitoring in southern California for seventy-seven years (1932-2008). *Bulletin of the Seismological Society of America*, *100*(2), 423–446. doi: 10.1785/0120090130

IBM. (2020). *El modelo de redes neuronales.* Descargado de https://www.ibm.com/docs/es/spss-modeler/SaaS?topic=networks-neural-model

Kingma, D. P., y Ba, J. L. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 1–15.

Kipf, T. N., y Welling, M. (2017). Semi-supervised classification with graph convolutional networks. *5th International Conference on Learning Representations, ICLR 2017*, 1–14.

Kriegerowski, M., Petersen, G. M., Vasyura-Bathke, H., y Ohrnberger, M. (2019). A deep convolutional neural network for localization of clustered earthquakes based on multistation full waveforms. *Seismological Research Letters*, *90*(2 A), 510–516. doi: 10.1785/0220180320

Kuang, W., Yuan, C., y Zhang, J. (2021). Network-based earthquake magnitude determination via deep learning. *Seismological Research Letters*, *92*(4), 2245–2254. doi: 10.1785/0220200317

Lomax, A., Michelini, A., y Jozinović, D. (2019). An Investigation of Rapid Earthquake Characterization Using Single-Station Waveforms and a Convolutional Neural Network. *Seismological Research Letters*, *90*(2 A), 517–529. doi: 10.1785/0220180311

Mousavi, S. M., y Beroza, G. C. (2020). A Machine-Learning Approach for Earthquake Magnitude Estimation. *Geophysical Research Letters*, *47*(1), 1–7. doi: 10.1029/2019GL085976

Mousavi, S. M., Ellsworth, W. L., Zhu, W., Chuang, L. Y., y Beroza, G. C. (2020). Earthquake transformer—an attentive deep-learning model for simultaneous earthquake detection and phase picking. *Nature Communications*, *11*(1), 1–12. Descargado de https://doi.org/10.1038/s41467-020-17591-w doi: 10.1038/s41467-020-17591-w

Ochoa Gutiérrez, L. H. (2016). *Solución rápida y automática de parámetros hipocentrales para eventos sísmicos, mediante el empleo de técnicas de aprendizaje de máquina* (Tesis Doctoral no publicada). Universidad Nacional de Colombia.

Perol, T., Gharbi, M., y Denolle, M. A. (2018). Convolutional neural network for earthquake detection and location. *Science Advances*, *4*(2), 1–9. doi: 10 .1126/sciadv.1700578

Ross, Z. E., Meier, M. A., Hauksson, E., y Heaton, T. H. (2018). Generalized seismic phase detection with deep learning. *Bulletin of the Seismological Society of America*, *108*(5), 2894–2901. doi: 10.1785/0120180080

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., y Monfardini, G. (2009). The graph neural network model. *IEEE Transactions on Neural Networks*, *20*(1), 61–80. doi: 10.1109/TNN.2008.2005605

Shen, H., y Shen, Y. (2021). Array-based convolutional neural networks for automatic detection and 4d localization of earthquakes in hawai'i. *Seismological Research Letters*, *92*(5), 2961–2971. doi: 10.1785/0220200419

Swapna. (2021). *Convolutional Neural Networks.* Descargado de https://developersbreach.com/convolution-neural-network-deep-learning/

Tavera, H. (2019). *¿Cómo se elabora un reporte sísmico?* Descargado de https://www.gob.pe/institucion/igp/noticias/74057-como-se-elabora-un-reporte-sismico-por-el-dr-hernando-tavera

The Guardian. (2018). *Indonesia's geophysics agency under fire for lifting tsunami warning.* Descargado de https://www.theguardian.com/world/2018/sep/30/indonesias-geophysics-agency-under-fire-for-lifting-tsunami-warning

van den Ende, M. P., y Ampuero, J. P. (2020). Automated Seismic Source Characterization Using Deep Graph Neural Networks. *Geophysical Research Letters*, *47*(17), 1–11. doi: 10.1029/2020GL088690

Walter, J. I., Ogwari, P., Thiel, A., Ferrer, F., y Woelfel, I. (2020). easyQuake: Putting machine learning to work for your regional seismic network or local earthquake study. *Seismological Research Letters*, *92*(1), 555–563. doi: 10.1785/0220200226

Wang, J., y Teng, T.-L. (1995). Artificial neural network-based seismic detector. *Bulletin of the Seismological Society of America*, *85*(1), 308–319. doi: 10.1785/bssa0850010308

Yablokov, A. V., Serdyukov, A. S., Loginov, G. N., y Baranov, V. D. (2021). An artificial neural network approach for the inversion of surface wave dispersion curves. *Geophysical Prospecting*. doi: 10.1111/1365-2478.13107

Ying, R., Morris, C., Hamilton, W. L., You, J., Ren, X., y Leskovec, J. (2018). Hierarchical graph representation learning with differentiable pooling. *Advances in Neural Information Processing Systems*, *2018-Decem*, 4800–4810.

Zhang, X., Zhang, M., y Tian, X. (2021). Real-Time Earthquake Early Warning With Deep Learning: Application to the 2016 M 6.0 Central Apennines, Italy Earthquake. *Geophysical Research Letters*, *48*(5), 0–10. doi: 10.1029/2020GL089394

Zhao, Y., y Takano, K. (1999). An artificial neural network approach for broadband seismic phase picking. *Bulletin of the Seismological Society of America*, *89*(3), 670–680. doi: 10.1785/bssa0890030670

Zhu, W., y Beroza, G. C. (2019). PhaseNet: A deep-neural-network-based seismic arrival-time picking method. *Geophysical Journal International*, *216*(1), 261–273. doi: 10.1093/gji/ggy423

ANEXOS

ANEXO A: CÓDIGOS EN PYTHON

A continuación se presentan los códigos desarrollados en *Python*. En este sentido, el Código A.1 muestra la arquitectura del modelo usado en la presente investigación. El Código A.2 muestra el *loop* usado para el entrenamiento y la validación.

Arquitectura del modelo

```
# IMPORTACIÓN DE MODULOS
import torch
from torch.nn import Linear #, BatchNorm1d
from torch_geometric.nn import DenseGCNConv
from torch_geometric.nn import dense_diff_pool
import torch.nn.functional as F # relu, leaky_relu, tanh, elu
from math import ceil

# CLASE DIFFERENTIAL POOLING

class DiffPool(torch.nn.Module):

    def __init__(self, feature_size, output_dim):
        super(DiffPool, self).__init__()
        self.feature_size = feature_size
        self.output_dim = output_dim
        self.embed = DenseGCNConv(self.feature_size, self.feature_size)
        self.pool = DenseGCNConv(self.feature_size, self.output_dim)

    def forward(self, x, a, mask=None):
        z = self.embed(x, a)
        s = self.pool(x, a)
        #print("s =", s.shape)

        x_new, a_new, l1, e1 = dense_diff_pool(z, a, s, mask)
        #x_new = s.t() @ z
        #a_new = s.t() @ a @ s
        return x_new, a_new

# MI ARQUITECTURA

class my_network(torch.nn.Module):

    def __init__(self, num_nodes):
        super(my_network, self).__init__()

        self.conv1 = DenseGCNConv(2048, 1024)
        self.conv2 = DenseGCNConv(1024, 512)
        num_nodes = ceil(0.25 * num_nodes)
        self.pool1 = DiffPool(512, num_nodes)
```

```
41
42          self.conv3 = DenseGCNConv(512, 256)
43          self.conv4 = DenseGCNConv(256, 128)
44          num_nodes = ceil(0.25 * num_nodes)
45          self.pool2 = DiffPool(128, num_nodes)
46
47          self.conv5 = DenseGCNConv(128, 64)
48          self.conv6 = DenseGCNConv(64, 64)
49
50          self.linear1 = Linear(64, 32)
51          self.linear2 = Linear(32, 16)
52          self.linear3 = Linear(16, 4)
53
54      def forward(self, x, a, mask=None):
55
56          x = self.conv1(x, a)
57          x = F.leaky_relu(x)
58          x = self.conv2(x, a)
59          x = F.leaky_relu(x)
60          x, a = self.pool1(x, a)
61
62          x = self.conv3(x, a)
63          x = F.leaky_relu(x)
64          x = self.conv4(x, a)
65          x = F.leaky_relu(x)
66          x, a = self.pool2(x, a)
67
68          x = self.conv5(x, a)
69          x = F.leaky_relu(x)
70          x = self.conv6(x, a)
71          x = F.leaky_relu(x)
72          x = torch.mean(x, dim=1, keepdim=True)
73
74          y = self.linear1(x)
75          y = F.leaky_relu(y)
76          y = F.dropout(y, p=0.5, training=self.training)
77
78          y = self.linear2(y)
79          y = F.leaky_relu(y)
80          y = F.dropout(y, p=0.2, training=self.training)
81
82          y = self.linear3(y)
83          #y = F.leaky_relu(y)
84
85          return x, a, y
```

Código A.1: Arquitectura del modelo

Loop de entrenamiento y validación

```
# IMPORTACIÓN DE MODULOS

import numpy as np
import torch
import torch_geometric
from torch_geometric.data import Data
from torch_geometric.loader import DenseDataLoader
from torch.nn import MSELoss  #, L1Loss
from torch.optim import Adam
import torch_geometric.transforms as T
import random
from my_gnn import *
from utils import *

# INITIAL SETTINGS

print("Initial Settings...")
device = torch.device("cuda" if torch.cuda.is_available() else "cpu") #
    ARGS.device
criterion = MSELoss()
num_nodes = dataset[0].x.shape[0]
model = my_network(num_nodes=num_nodes)
if torch.cuda.device_count() > 1:
  print("Let's use", torch.cuda.device_count(), "GPUs!")
  # dim = 0 [30, xxx] -> [10, ...], [10, ...], [10, ...] on 3 GPUs
  model = torch.nn.DataParallel(model)
model.to(device)
lr = ARGS.lr
optimizer = Adam(model.parameters(), lr=lr)
epochs = ARGS.epochs

# LOOP DE ENTRENAMIENTO

hist = {"loss":[], "Train_MSE":[], "Val_MSE":[]}
for epoch in range(epochs):
    loss = train()
    train_mse = test(train_loader)
    val_mse = test(val_loader)
    #scheduler.step()
    hist["loss"].append(loss)
    hist["Train_MSE"].append(train_mse)
    hist["Val_MSE"].append(val_mse)
    print(f'Epoch: {epoch+1:03d}, Loss: {loss:.4f}, Train MSE: {
    train_mse:.4f}, Val MSE: {val_mse:.4f}')
```

Código A.2: Loop de entrenamiento y validación