Universidad del Bío-Bío, Chile

Facultad de Ciencias Empresariales

Departamento de Sistemas de Información

Evaluación de modelos predictivos basados en Deep Learning sobre Estimación de Recursos Minerales

Tesis presentada por Carlos Olmos de Aguilera Díaz
para obtener el grado de Magíster en Ciencias de la Computación
Dirigida por
Dr. Pedro Campos Soto
Dra. Nathalie Risso Sepúlveda

2021

Agradecimientos

A mis cercanos que me sostuvieron y profesores que me guiaron.

El desarrollo de esta Tesis fue apoyado por los proyectos DIUBB 2030248 IF/R y 194810 GI/VC.

Resumen

Uno de los procesos que tiene un mayor impacto en una operación minera es la estimación de concentración de minerales. En esta tesis se propone el uso de modelos de Deep Learning (DL) para la estimación de concentración de cobre empleando datos de una mina en Chile. Por un lado, estos datos incluyen información de un modelo de planificación de largo plazo o Long-Term Mine Planning Model (LTMP), compuesto de estimaciones de la concentración de minerales, información que se emplea para definir los flujos de caja que determinan la viabilidad económica en un proyecto minero, y por otro lado, información de un modelo de planificación de corto plazo o Short-Term Mine Planning Model (STMP), construido en base a la porción de roca extraída y que se considera como la concentración y distribución de referencia. Los modelos LTMP y STMP pueden ser contrastados para determinar discrepancias. Con el propósito de reducir la diferencia entre ambos se propone emplear los modelos Feedforward Neural Network (FNN), Convolutional Neural Network (CNN) y Recurrent Neural Network (RNN). Para CNN se evalúa la variación CNN 1D y para RNN la variación Long Short-Term Memory (LSTM), que han probado ser eficientes en conjuntos de datos en los que el orden es relevante y se espera consideren el orden espacial de la concentración mineral. En principio se cuenta con un conjunto de datos de 49,295,441 instancias, que es preprocesado mediante selección de variables, tratamiento de datos ausentes, generación de variables indicadoras y normalización, lo que produce un conjunto de datos de 732,870 instancias. También se propone que, para la estimación de concentración mineral de un nodo central, se consideren los atributos de nodos vecinos, lo que añade información de contexto para la estimación. La aplicación de esta metodología genera un conjunto de datos de 545,768 instancias. Se realizan experimentos empleando el método K-fold Cross-validation (CV) para optimización de hiperparámetros. Los hiperparámetros resultantes son empleados para realizar una prueba final. El desempeño se mide en base a las métricas MAE, MSE y R^2. En base a las métricas se comparan los modelos LTMP y STMP para calcular un baseline. Las estimaciones de los modelos DL también se comparan con STMP. En las pruebas finales, para la métrica MSE, el modelo FNN logra una mejora del 21 %, CNN logra una mejora del 37 % y RNN una mejora del 39 % respecto al baseline.

Keywords — Estimación de Concentración de Minerales, Aprendizaje Profundo, Reconciliación, Aprendizaje Automático, Redes Neuronales Recurrentes.

Abstract

Mineral resource estimation is one of the processes with greatest impact in mining. In this thesis Deep Learning (DL) models are proposed for Copper grade estimation for a mine in Chile. The proposed method involves a Long-Term Mine Planning Model (LTMP) that contains ore grade estimations along with geological properties, and a Short-Term Mine Planning Model (STMP) based on the portion of extracted rock. These models can be contrasted to determine discrepancies. In order to reduce the difference between them, Feedforward Neural Network (FNN), Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN) models are proposed. For CNN the variation CNN 1D is implemented and for RNN the variation Long Short-Term Memory (LSTM), which have proven to be efficient in datasets where the order is relevant and it is expected that they allow considering spatial order of mineral concentration in the estimations. Initially, a data set of 49,295,441 instances is preprocessed by variable selection, missing data treatment, indicator variable generation and normalization. This treatment produces a dataset of 732,870 instances. The use of central nodes and attributes of neighboring nodes is also proposed, which adds contextual information for the estimations. The application of this methodology generates a data set of 545,768 instances used for CNN and RNN. Experiments are performed using the K-fold Cross-validation (CV) method for hyperparameter optimization. The resulting hyperparameters are used to perform a final test. Performance is measured based on the MAE, MSE and R^2 metrics. The models LTMP and STMP are compared to calculate a baseline. DL models estimates are compared to STMP. In the final tests, for MSE, FNN achieves an improvement of 21 %, CNN achieves an improvement of 37 % and RNN an improvement of 39 % over baseline.

Keywords — Mineral Resource Estimation, Deep Learning, Reconciliation, Machine Learning, Recurrent Neural Network

Índice general

Índice de figuras

Índice de tablas

Lista de acrónimos

CNN Convolutional Neural Network.

CV K-fold Cross-validation.

DL Deep Learning.

DT Decision Tree.

FNN Feedforward Neural Network.

GA Genetic Algorithm.

GP Gaussian Process Regression.

GRNN General Regression Neural Network.

IK Indicator Kriging.

KNN K-Nearest Neighbors.

LK Lognormal Kriging.

LS-SVM Weighted Least Squares Support Vector Machine.

LSTM Long Short-Term Memory.

LTMP Long-Term Mine Planning Model.

MAE Mean Absolute Error.

ME Mean Error.

MIK Median Indicator Kriging.

ML Machine Learning.

MSE Mean Squared Error.

NN Neural Network.

OK Ordinary Kriging.

PSO Particle Swarm Optimization.

R^2 Coefficient of determination.

RBF Radial Basis Function.

RBNN Radial Basis Neural Network.

ReLU Rectified Linear Unit.

RF Random Forest.

RK Residual Kriging.

RNN Recurrent Neural Network.

SK Simple Kriging.

STMP Short-Term Mine Planning Model.

SVM Support Vector Machine.

WKNN Weighted K-Nearest Neighbors.

Nomenclatura

Esta sección describe la notación utilizada a lo largo de esta tesis.

Números y vectores

a	Un valor escalar
$\boldsymbol{a}$	Un vector
$\boldsymbol{A}$	Una matriz

Conjuntos

$\mathbb{A}$	Un conjunto
$\{0, 1\}$	Un conjunto con los números 0 y 1
$\{0, 1, ..., n\}$	Un conjunto con todos los números enteros entre el 0 y un valor n

Índices

a_i	Elemento i de un vector $\boldsymbol{a}$
$A_{i,j}$	Elemento i, j de una matriz $\boldsymbol{A}$

Conjunto de datos

$\mathbb{X}$	Un conjunto de datos de instancias
$x^{(i)}$	La i-ésima instancia (entrada) de un conjunto de datos
$y^{(i)}$	La etiqueta asociada con $x^{(i)}$ para Aprendizaje Supervisado
$\hat{y}^{(i)}$	La etiqueta estimada asociada con $x^{(i)}$
$x_j^{(i)}$	Elemento j de una instancia $x^{(i)}$

Capítulo 1

Introducción

La minería es una actividad de gran importancia en Chile, con un aporte al Producto Interno Bruto (PIB) nacional del 9.4 % en 2019 y un promedio de un 11.06 % en los últimos 10 años, por lo que representa un agente macroeconómico de dinamización de la economía y la producción chilena. Chile es el mayor productor de cobre en el mundo, en 2019 tuvo una participación del 28.4 % en la producción internacional, lo que equivale a 5,822 mil toneladas y un aporte del 8.4 % del PIB. En Chile también se producen otros minerales como el molibdeno (18.4 % de producción mundial, unas 53,541 toneladas), yodo (67.9 % de la producción mundial, que equivale a 20,826 toneladas), litio (26.1 % de la producción mundial, equivalente a 20,931 toneladas), entre otros (Servicio Nacional de Geología y Minería, 2019).

Para concretar la producción de un mineral se organizan y planifican las operaciones de una mina en base a estimaciones de concentración del o los minerales de interés. Se realizan estudios de largo plazo para el depósito completo, actualizados cada 1 a 3 años, como también, estudios de corto plazo, para una semana o día a día, para decisiones relacionadas con el control de la concentración del mineral y para planificación detallada (Rossi and Deutsch, 2014). Estos estudios, permiten modelar la situación de la mina y son empleados para orquestar las operaciones que culminan con la producción del o los minerales. Por lo tanto, aportan información empleada para definir expectativas de insumos requeridos, maquinaria, horas hombre y flujos de caja.

En la práctica, un 10 % de error en la estimación de concentración no es poco común, y es generalmente considerado como aceptable en la operación de una mina dentro de un periodo de un año. En algunos casos, al comparar la producción versus la concentración estimada pueden presentarse errores de hasta ±50 % a 80 %. Para una buena operación, los costos de producción representan al menos entre el 50 % y 70 % de los ingresos (ventas) de una mina y una disminución del 10 % en la concentración real (respecto de su estimación) se puede traducir en una disminución de entre un 20 % a un 40 % del excedente de producción (Dominy et al., 2002).

Estas inconsistencias entre las concentraciones estimadas y reales pueden generar una pérdida contable (disminución de activos), que en la práctica dependiendo del nivel proporcional de los cargos de amortización y depreciación, pueden tener como consecuencia

que un proyecto no sea viable financieramente, contrario a la apreciación inicial.

Por lo tanto, para que la planificación y operación de una mina sea exitosa es necesario contar con, entre otras cosas, estimaciones de calidad. Estas estimaciones están limitadas por la calidad de los datos de los que se alimentan y por la capacidad de los métodos de estimación.

En este sentido, buscando mejorar los resultados de métodos geoestadísticos tradicionalmente empleados para este propósito, se han realizado algunos trabajos que buscan estudiar diferentes modelos de Machine Learning (ML) como una alternativa interesante para la estimación de concentración de mineral, tales como Redes Neuronales, Máquinas de Vectores de Soporte y Árboles de Decisión. Esto, debido a la capacidad de modelar relaciones no lineales y a que no dependen de suposiciones respecto del problema (Samanta et al., 2005). Sin embargo, pocos de estos trabajos emplean modelos de Deep Learning, por lo que para esta tesis se propone la evaluación de arquitecturas actuales de este tipo, específicamente, los modelos Feedforward Neural Network (FNN), Convolutional Neural Network (CNN) y Recurrent Neural Network (RNN), debido a su estructura que ha demostrado una capacidad para modelar dependencias espaciales (Chen and Zhang, 2020).

1.1. Propuesta de tesis

1.1.1. Problema

La estimación de recursos minerales es un problema estadístico que involucra determinar la concentración porcentual del mineral de interés en áreas donde no se han extraído muestras. Esto se consigue mediante el uso de información puntual y georeferenciada en un espacio tridimensional, es decir, datos que pueden ser localizados mediante coordenadas horizontales (profundidad y anchura) y su altura.

Este proceso requiere de un conjunto $\mathbb{R}$ de datos que corresponden a muestras (usualmente de perforaciones) $r^{(1)}, r^{(2)}, r^{(3)}, \ldots, r^{(d)}$ recogidas de ubicaciones específicas y dispersas de un depósito, donde d representa la cantidad total de muestras o instancias en el conjunto de datos. Cada muestra $r^{(k)}$, con $k = 1, \ldots, d$, es un vector compuesto por coordenadas espaciales y características de la roca $\{c_1, c_2, c_3 \ldots, c_p\}$ como, por ejemplo, concentraciones de minerales, densidad de la roca, etcétera.

En base a las muestras del conjunto $\mathbb{R}$ y mediante métodos usualmente geoestadísticos, se realizan estimaciones (para todo el volumen en estudio) $l^{(1)}, l^{(2)}, l^{(3)}, \ldots, l^{(m)}$, donde $m \gg d$, que componen el conjunto $\mathbb{L}$, que corresponde al Long-Term Mine Planning Model (LTMP). Cada estimación $l^{(j)}$, con $j = 1, \ldots, m$, es un vector que incluye estimaciones para las variables $\{c_1, c_2, c_3 \ldots, c_p\}$ en las muestras en adición con otras propiedades geológicas $\{g_1, g_2, g_3 \ldots, g_q\}$ estimadas para la roca, como el tipo de alteración mineral, el tipo de ambiente geológico, etcétera, por lo que cada vector $l^{(j)}$ es conformado por características $\{c_1, c_2, c_3 \ldots, c_p, g_1, g_2, g_3 \ldots, g_q\}$.

Por otro lado, durante operaciones de extracción de roca, se recogen nuevas muestras para las características $\{c_1, c_2, c_3 \ldots, c_p\}$ que se emplean para construir un nuevo conjunto $\mathbb{S}$, que corresponde al Short-Term Mine Planning Model (STMP), que está conformado por

muestras $s^{(1)}, s^{(2)}, s^{(3)}, \ldots, s^{(n)}$, donde $n \gg d$ y $n < m$. Ambos conjuntos $\mathbb{L}$ y $\mathbb{S}$ comparten el mismo sistema de georeferencia y ambos tienen las características $\{c_1, c_2, c_3 \ldots, c_p\}$, a pesar de que $\mathbb{S}$ tiene información de una porción del volumen en estudio.

Por lo tanto, para aquellos puntos en el sistema compartido de georeferencia en los que se dispone de valores para las características $\{c_1, c_2, c_3 \ldots, c_p\}$, es posible realizar una comparación y establecer el error ε de las estimaciones en el conjunto $\mathbb{L}$ en contraste con los valores en $\mathbb{S}$.

En esta tesis se propone la implementación y evaluación de modelos de Deep Learning (DL) en un enfoque de aprendizaje supervisado para la estimación de concentración mineral. Esto es posible tras la selección de vectores del conjunto $\mathbb{L}$ en puntos para los que también se disponga información de concentración mineral en los vectores del conjunto $\mathbb{S}$. De este modo, un vector l_i se emplearía como vector de entrada X en un modelo Deep Learning (DL) para generar una estimación de concentración de mineral y la concentración en el vector s_i sería una guía para ajustar dicho modelo. Las estimaciones generadas por los modelos DL puede ser contrastadas con los valores en $\mathbb{S}$ para establecer el error ε'.

Finalmente, la diferencia entre los errores ε y ε' permitirá establecer el aporte de los modelos DL y determinar si efectivamente se logra disminución de error en la estimación de concentración mineral.

1.1.2. Hipótesis

La literatura muestra que se requieren métodos capaces de construir relaciones lineales y no lineales que modelen la distribución de propiedades en extensiones de roca en un depósito, considerando su distribución y dependencia espacial, para una adecuada estimación de concentración de minerales. Considerando las capacidades que tienen los modelos de Deep Learning en tareas de estimación complejas, esta propuesta de tesis se basa en la siguiente hipótesis:

> El uso de modelos basados en Deep Learning permite mejorar resultados de métodos tradicionales en el problema de estimación de concentración de recursos minerales.

1.1.3. Objetivos

Objetivo General

Evaluar el desempeño de modelos basados en Deep Learning sobre datos de Modelos de Recursos Minerales para determinar mejoras en la estimación de concentración de minerales.

Objetivos específicos

- OE1: Seleccionar modelos basados en Deep Learning que permitan realizar estimaciones de concentración de recursos minerales.

- OE2: Implementar modelos seleccionados, considerando posibles adaptaciones y modificaciones.
- OE3: Evaluar resultados de los modelos implementados mediante métricas de error y pruebas estadísticas.

1.1.4. Alcance de la Investigación

Esta tesis se desarrolla con el apoyo de profesionales del área de geología, pertenecientes a una consultora profesional con sede en Santiago, lo que provee acceso a datos reales de una mina en Chile. Esta mina es de cielo abierto y de ella se extrae principalmente Cobre, pero también se dispone de información de los minerales Arsénico y Molibdeno. Es importante destacar que información detallada respecto de los datos y de su origen no pueden ser revelados producto de cláusulas de confidencialidad. Por lo tanto, este trabajo se enfoca en el uso y evaluación de modelos de Deep Learning sobre datos ya recopilados.

Esta tesis contempla el procesamiento y adaptación de conjuntos de datos, la implementación de metodologías de entrenamiento y validación, la implementación de modelos y arquitecturas de Redes Neuronales, así como la selección adecuada de hiperparámetros. Los modelos empleados corresponden a implementaciones ya disponibles, extraídas de bibliotecas de código gratuitas y de libre acceso. No se incluye la implementación de ningún tipo de método de estimación geoestadístico y no se considera la combinación o ensamble de modelos DL.

1.1.5. Metodología de trabajo

La metodología de trabajo considera las siguientes actividades:

- OE1
  - Revisión bibliográfica: revisión exhaustiva respecto al uso de Deep Learning en la estimación de concentración mineral, y arquitecturas compatibles con dicha tarea.
  - Estudio y selección de soluciones: se estudian arquitecturas y enfoques de Deep Learning, identificando candidatos para implementación.
- OE2
  - Implementación: Implementación de modelos y arquitecturas de Deep Learning seleccionadas.
  - Experimentación preliminar y adaptaciones: ejecución de experimentos preliminares para ajustar hiperparámetros y realizar adaptaciones a las implementaciones realizadas.
- OE3

- Diseño de experimentos: Definición de métricas y metodología de evaluación, y preparación de datos para experimentación empleando datos reales de un depósito mineral localizado en Chile.
- Evaluación de resultados: ejecución de experimentos, tabulación e interpretación de resultados de los experimentos.

Para la implementación de esta tesis se ha seleccionado el lenguaje programación Python[1], y el uso de bibliotecas de código como Pandas[2] para el tratamiento de los datos, ScikitLearn[3] para su procesamiento, Matplotlib[4] para visualización de datos, Keras[5] y TensorFlow[6] para la implementación de modelos de DL. Dentro del desarrollo de la implementación, porciones de código fueron probados empleando Jupyter Notebooks[7].

Tabla 1.1: Versiones de bibliotecas de código por servidor

Librería	Versión en servidor	
	Postgrado	MLVC
Python	3.6.9	3.7.3
Pandas	1.1.3	1.0.1
ScikitLearn	0.23.2	0.22.1
Matplotlib	3.3.2	3.1.3
Keras	2.4.3	2.3.1
TensorFlow	2.3.1	2.1.0
Numpy	1.18.5	1.18.1
Scipy	1.5.2	1.4.1

Para los experimentos se dispone de dos servidores en la Universidad del Bío-Bío, el servidor de Posgrado de la Facultad de Ciencias Empresariales compuesto de 2 procesadores Intel Xeon Gold 5118 CPU @ 2.30 GHz de 12 núcleos y 24 hilos cada uno junto con 64 GB RAM; y el servidor del grupo de investigación Machine Learning y Visión por Computador (MLVC) que dispone de un procesador Intel Xeon E5-2603 v4 CPU @ 1.70 GHz y 40 GB de RAM. Las versiones para cada servidor de las librerías antes mencionadas están disponibles en la Tabla 1.1

1.2. Organización de Tesis

El resto de esta tesis se compone por el capítulo 2 de conceptos preliminares, donde se abordan conceptos relacionados a la estimación de concentración de recursos minerales,

[1] https://www.python.org/
[2] https://www.pandas.pydata.org/
[3] https://www.scikit-learn.org/stable/
[4] https://www.matplotlib.org/
[5] https://www.keras.io/
[6] https://www.tensorflow.org/
[7] https://www.jupyter.org/

detalles del funcionamiento de algunos métodos geoestadístico, se brinda una introducción sobre Machine Learning explicando conceptos relevantes en el área, para continuar con una explicación de los modelos de Deep Learning a evaluar. En el capítulo 3 se presentan los resultados de una revisión de la literatura en la que se seleccionan una serie de trabajos relacionados que permiten construir el estado del arte. En el capítulo 4 se brinda información sobre el conjunto de datos empleado, el preprocesamiento realizado, se explica la metodología empleada para la selección de nodos vecinos, los modelos implementados y las librerías de código empleadas. Luego, en el capítulo 5, se explica el diseño de los experimentos, la distribución de los datos disponibles para la validación y evaluación de las capacidades de los modelos, se presentan los resultados por cada experimento y finalmente se presenta una discusión sobre los resultados obtenidos. La tesis finaliza con el capítulo 6 donde se presentan las conclusiones del trabajo desarrollado y se discuten posibles trabajos futuros.

Capítulo 2

Conceptos Preliminares

2.1. Estimación de recursos minerales

La producción de un mineral está determinada por el ciclo de vida de una mina y abarca una serie de actividades específicas. Estas se pueden dividir en cuatro etapas principales: Prospección y Exploración, Desarrollo, Extracción, y Remediación.

La etapa de Prospección y Exploración comienza con la búsqueda de una zona para analizar su potencial mineralógico. Esta etapa busca desarrollar un informe de factibilidad económica y técnica. Esto se logra mediante la recopilación de datos obtenidos a partir de muestras de roca, su análisis y la identificación de recursos geológicos. Luego, en la etapa de Desarrollo de una mina, se trabaja en la planificación y logística de las operaciones mineras, se obtienen los permisos de extracción y se realizan las construcciones de infraestructura básica.

En la etapa de Extracción el mineral se obtiene mediante técnicas como la tronadura, que consiste en la fragmentación de la roca en su estado natural para reducirla a un tamaño adecuado mediante explosivos para su transporte y posterior procesamiento. La extracción se realiza por áreas, en un orden específico de acuerdo con la planificación. Esta es establecida en forma anual, mensual e incluso diariamente y es actualizada continuamente. Una vez llegado al final de la vida útil de una mina, en su cierre, corresponde la Remediación o remodelación de la tierra a un estado más natural.

En la etapa de Prospección y Exploración es importante establecer la factibilidad de la extracción y producción de mineral, lo que depende de las posibilidades de que los minerales puedan ser explotados con beneficio económico considerando los medios técnicos disponibles. Esto se determina en base a la información de la distribución, continuidad, y extensión de las propiedades en la roca, así como también el total de toneladas (Jalloh et al., 2016). Esto se logra mediante la información obtenida de las muestras de roca que se extraen a una cierta profundidad y distancia específicas, buscando cubrir parte importante del volumen de estudio. Cada muestra de roca tiene asociadas coordenadas espaciales lo que permite la construcción de modelos virtuales tridimensionales, los que usualmente son construidos mediante métodos geoestadísticos para la interpolación de valores en todo el volumen de interés.

El trabajo desarrollado en esta tesis considera un Modelo de Bloques como representación virtual tridimensional. Un Modelo de Bloques está compuesto por nodos, donde cada nodo simboliza una porción del volumen en estudio y está delimitado por una altura, ancho y largo (Ej: 15 metros de alto, 20 metros de ancho y 20 metros de largo - 6,000m^3). En la Figura 2.1 se aprecian los nodos representados como cubos. Cada nodo tiene una coordenada en un espacio tridimensional y atributos/variables geológicas. Los atributos en una roca pueden ser su densidad, concentración de un mineral (varios minerales pueden estar presentes), tipo de alteración en los minerales, tipo de litología, etcétera. Esta estructura en los datos permite, por ejemplo, identificar zonas de mayor concentración mineral y su distribución, la cual en la Figura 2.1 se representa con un color rojo más intenso.

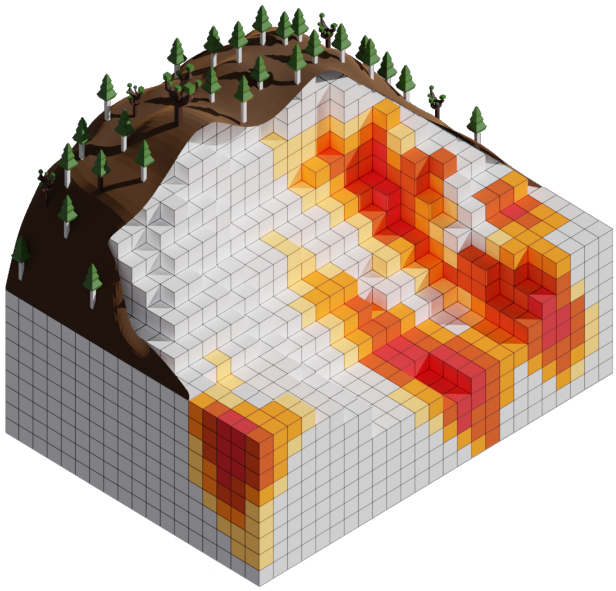


Figura 2.1: Ejemplo de representación de modelo de bloques. Basado en (Recmin, 2019)

LTMP y STMP son modelos de bloques. LTMP está compuesto por estimaciones (generadas principalmente por métodos geoestadísticos) en base a los datos extraídos de muestras puntuales. En el inicio del ciclo de vida de una mina LTMP juega un rol fundamental, ya que permite establecer, por ejemplo, el total de toneladas que se extraerán de un mineral y en qué periodos se realizará su producción lo que, a su vez, permite establecer los flujos de caja y finalmente la factibilidad económica del proyecto. Por otro lado, STMP también está compuesto por estimaciones en base a una enorme cantidad de muestras que se obtienen de la roca a medida que esta se extrae. Se considera como la realidad y permite realizar un seguimiento en el cumplimiento de las metas de producción. STMP crece en función del avance en la explotación de la mina y es contrastado con LTMP, este proceso se denomina Reconciliación, y permite evaluar la calidad de las estimaciones en LTMP y corregirlas (Parker, 2012).

Las estimaciones de concentración de minerales pueden ser validadas de forma gráfica

y estadística. La revisión gráfica implica visualización tridimensional de secciones y planos respecto a un atributo. La revisión estadística se realiza de forma global, para todo el depósito, o de forma local para volúmenes discretos, verificando la consistencia del modelo.

Se dice que se produce dilución cuando concentraciones de minerales pasan desapercibidas producto de diferencias en volúmenes empleados para realizar los análisis; esto ocurre, por ejemplo, cuando ciertos trazos de mineral de alta concentración corresponden a una parte menor del volumen de un bloque. Algo similar ocurre cuando existe una presencia heterogénea de minerales. Por esta razón es necesario validar los promedios globales y los promedios de los datos desagrupados, así como también, verificar que las relaciones espaciales y estadísticas entre las variables modeladas correspondan con las relaciones de los datos originales. Otras formas de validar estimaciones incluyen (1) la comparación con resultados de otros métodos, pero deben considerarse las diferencias en las propiedades de cada uno, dado que, por ejemplo, algunos métodos obtienen mejores estimaciones globales mientras que otros mejores estimaciones locales; (2) la comparación con estimaciones previas y datos históricos conocidos de producción; así, la capacidad del modelo para estimar operaciones futuras se puede fundamentar en su capacidad para estimar aquellos datos históricos.

2.1.1. Métodos Geoestadísticos

En la revisión de la literatura realizada en esta tesis se identifica al método Ordinary Kriging (OK) como el método en geoestadística más empleado para realizar estimaciones de concentración mineral (Afeni et al., 2020). Este método permite estimar la concentración mineral en un punto dentro del volumen en estudio de un depósito, en base a la concentración de puntos cercanos. Esto se basa en la suposición de que puntos más próximos geográficamente tienen valores más similares.

De acuerdo con (Shortridge, 2019) Ordinary Kriging es un método que emplea una suma ponderada tal como se observa en la Ecuación 2.1 para estimar la concentración mineral $\hat{Y}_0$ en un punto p_0 para el que no se conoce la concentración. Este requiere las concentraciones conocidas y_i de un grupo determinado de n puntos cercanos p_i. A cada concentración conocida se le asocia un peso w_i que es calculado en base a la ubicación de un punto p_i respecto de los demás puntos y respecto del punto p_0.

$$\hat{Y}_0 = \sum_{i=1}^{n} w_i y_i = [w_1 \cdots w_n] \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} = \boldsymbol{wy} \tag{2.1}$$

La formulación de OK define una variable aleatoria $Y(p_i) = Y_i$ y un estimador lineal $\hat{Y}$, de modo que, $\hat{Y}(p_0) = \hat{Y}_0$. De esta manera, se busca encontrar el estimador $\hat{Y}_0 = \boldsymbol{wy}$ de manera que se cumplan las siguientes propiedades:

1. $E(\hat{Y}_0) = E(Y_0)$ (imparcialidad). Lo que se satisface si $\sum_{i=1}^{n} w_i = 1$ y la media μ es estacionaria ($E(Y_i) = \mu, \forall i$) y desconocida.

2. La varianza de la predicción $\sigma_\epsilon^2 = E\left[Y_0 - \hat{Y}_0\right]^2 = \text{Var}(Y_0 - \hat{Y}_0)$ es minimizada.

El desarrollo de σ_ϵ^2 mediante el uso de propiedades de la varianza se presenta en la Ecuación 2.2. Donde $\text{Cov}(Y_i, Y_j)$, $\forall i, j$ $(j = 1, ..., n)$ es la matriz de covarianza que resulta del cálculo de las covarianzas entre cada uno de los puntos cercanos a p_0. Mientras que $\text{Cov}(Y_i, Y_0) \forall i$ es un vector de covarianzas que resulta del cálculo de las covarianzas entre cada punto cercano y el punto p_0. La función de covarianza usa las coordenadas de las instancias y de este modo, se incorpora información de cuán similar se espera que los valores del vecindario sean del valor a estimar.

$$\text{Var}(Y_0 - \hat{Y}_0) = \text{Var}(Y_0) + \sum_{i=1}^{n}\sum_{j=1}^{n} w_i w_j \text{Cov}(Y_i, Y_j) - 2\sum_{i=1}^{n} w_i \text{Cov}(Y_i, Y_0) \tag{2.2}$$

Para minimizar σ_ϵ^2 de manera que $\sum_{i=1}^{n} w_i = 1$ se emplea el método de multiplicadores de Lagrange. Esto implica llevar las ecuaciones a una forma Lagrangiana (Ecuaciones 2.3 y 2.4) que se resuelve mediante derivadas parciales y que tiene como resultado un sistema de ecuaciones lineales denominado Ecuaciones Kriging (Ecuación 2.5).

$$L = \text{Var}(Y_0 - \hat{Y}_0) + 2\lambda\left(\sum_{i=1}^{n} w_i - 1\right) \tag{2.3}$$

$$L = \text{Var}(Y_0) + \sum_{i=1}^{n}\sum_{j=1}^{n} w_i w_j \text{Cov}(Y_i, Y_j) - 2\sum_{i=1}^{n} w_i \text{Cov}(Y_i, Y_0) + 2\lambda\left(\sum_{i=1}^{n} w_i - 1\right) \tag{2.4}$$

$$\left[\begin{array}{c} w_1 \\ \vdots \\ w_n \\ \hline \lambda \end{array}\right] = \left[\begin{array}{ccc|c} C_{11} & \cdots & C_{1n} & 1 \\ \vdots & & \vdots & \vdots \\ C_{n1} & \cdots & C_{nn} & 1 \\ \hline 1 & \cdots & 1 & 0 \end{array}\right]^{-1} \left[\begin{array}{c} C_{10} \\ \vdots \\ C_{n0} \\ \hline 1 \end{array}\right] \tag{2.5}$$

Donde λ es un multiplicador de Lagrange que aparece debido a la restricción de imparcialidad. En el lado derecho en la Ecuación 2.5 están las covarianzas entre los puntos cercanos (izquierda) y las covarianzas entre los puntos cercanos y el punto a predecir (derecha). La resolución del sistema de ecuaciones permite obtener los valores de los pesos $\boldsymbol{w}$.

Inspirados en el método Ordinary Kriging o mediante la modificación de algunos de sus aspectos, se han desarrollado diversos métodos para adaptarse a propiedades en los datos disponibles y/o para mejorar la calidad de las estimaciones. Uno de estos métodos es Simple Kriging (SK), una forma más sencilla de OK, método que no considera la restricción de imparcialidad y además, supone una media μ estacionaria, pero conocida $E(Y_i) = E(Y_0) = \mu$, usualmente calculada a partir de la media de todo el conjunto de datos (Samanta et al., 2004). Estas modificaciones resultan en un sistema de ecuaciones

lineales más sencillo (Ecuación 2.6) y se altera la ecuación del cálculo del valor a estimar (Ecuación 2.7) (Shortridge, 2019).

$$\begin{bmatrix} w_1 \\ \vdots \\ w_n \end{bmatrix} = \begin{bmatrix} C_{11} & \cdots & C_{1n} \\ \vdots & & \vdots \\ C_{n1} & \cdots & C_{nn} \end{bmatrix}^{-1} \begin{bmatrix} C_{10} \\ \vdots \\ C_{n0} \end{bmatrix} \tag{2.6}$$

$$\hat{Y}_0 = \sum_{i=1}^{n} w_i y_i + \left(1 - \sum_{i=1}^{n} w_i\right) \mu \tag{2.7}$$

El método Ordinary Kriging y variaciones similares garantizan estimaciones locales precisas, pero mantiene un inconveniente conocido como efecto suavizante, donde el método sobrestima pequeños valores y subestima grandes valores. Además, en el caso de Simple Kriging (SK), las estimaciones suelen ser más cercanas a la media que se asume conocida (Yamamoto, 2005). Otro inconveniente es que el método Kriging no tiene en consideración la continuidad espacial del mineral. Esto resulta del promedio ponderado entre concentraciones cercanas.

Otras variaciones Kriging son Indicator Kriging (IK), que usa una función indicatriz que define un umbral que transforma los datos en 0 y 1 (Jafrasteh et al., 2018). Este método tiene una variación denominada Median Indicator Kriging (MIK) en el cual se asume que la continuidad espacial de los indicadores en varios límites se puede aproximar mediante una sola función estructural (Badel et al., 2011). Este método consume menos tiempo de ejecución, y es recomendable solo bajo ciertas condiciones en los datos; también existe el método Lognormal Kriging (LK), una variación de OK donde los datos se trabajan con una distribución log-normal (Samanta et al., 2002); y Residual Kriging (RK) también conocido como Regression-Kriging donde OK es aplicado sobre los residuales de un modelo de regresión (Chatterjee et al., 2007).

Los métodos Kriging no son los únicos métodos que han sido empleados para la estimación de concentración de minerales. Desde 1993 (Wu and Zhou, 1993) se ha estudiado el uso de modelos basados en ML, un área en desarrollo para la que se continúan produciendo avances y progresos.

2.2. Machine Learning

Machine Learning (ML) es un subcampo de la Inteligencia Artificial, que busca dotar a las máquinas de la capacidad de aprender de manera automática, por esto el término se traduce como Aprendizaje Automático. El aprendizaje en este contexto es un proceso en el que un sistema mejora su desempeño para alguna tarea en base a la experiencia, mediante la capacidad de detectar de forma automática patrones significativos que permitan ajustar su comportamiento. La extracción de información de conjuntos de datos se consigue mediante la optimización de un criterio de desempeño, lo que depende del ajuste de parámetros que dan forma a un modelo. El modelo por desarrollar puede ser predictivo, para realizar

estimaciones en el futuro, o descriptivo, para obtener conocimiento de los datos, o ambos (Alpaydin, 2014).

Los modelos de ML se alimenta de experiencia pasada contenida en conjuntos de datos. Los conjuntos de datos suelen ser un registro de instancias de un fenómeno en estudio, donde cada instancia está compuesta por variables relevantes para dicho fenómeno. Estas variables son denominadas atributos (del inglés, features) y las instancias son denominadas vectores de atributos.

Por lo general, los conjuntos de datos deben ser preparados antes de ser procesados por un modelo de ML, lo que implica realizar ciertas modificaciones en el conjunto de datos. Las modificaciones pueden variar en base a la sensibilidad de los modelos, algunos no toleran valores ausentes, son muy sensibles a valores atípicos (outliers) y/o requieren que los valores estén normalizados. En algunos casos, se suele realizar exclusión o selección de atributos y/o generación de nuevas variables a partir de las existentes, esto dice relación con la ingeniería de atributos, y es una diferencia importante respecto al preprocesamiento de los datos requerido para la implementación de modelos DL, enfoque descrito en la sección 2.3. Las modificaciones realizadas tienen como fin mejorar el desempeño de los modelos (Huang et al., 2015). Como resultado de este proceso se obtiene, para cada instancia, un vector $\boldsymbol{x}$ con las variables que serán procesadas por el modelo de ML, es decir, los datos de entrada para éste.

Para algunos fenómenos se dispone de un atributo objetivo o etiqueta, que suele ser denotado por y. Este atributo es aquel que se busca estimar en base al resto de atributos. En este escenario en que se dispone de un vector de atributos y una etiqueta, es posible implementar lo que se denomina enfoque supervisado. Esto permite emplear un modelo de ML que reciba un i-ésimo vector de entrada $\boldsymbol{x}_i$ y genere como salida una estimación $\hat{y}_i$ que es comparada con la etiqueta y_i. Esto es parte de lo que se denomina entrenamiento, donde en base a esta comparación se mide el error en la estimación y se ajusta el modelo, lo que se realiza de forma iterativa. De este modo, cuando el modelo generado logra captar de manera correcta las relaciones en los datos y es capaz de realizar estimaciones para para las instancias en el conjunto de datos, se espera que pueda realizar estimaciones correctas en escenarios futuros en los que no está disponible la etiqueta.

Otro enfoque denominado aprendizaje no supervisado, no requiere de valores etiqueta; este enfoque sólo emplea los vectores de entrada para construir modelos de probabilidades o para agrupar vectores de características similares en base a sus atributos, lo que permite extraer información y adquirir un mejor entendimiento sobre los datos.

Dependiendo del tipo de estimación a generar, se pueden distinguir (1) modelos de regresión, que suelen tener solo un valor de salida generado por el modelo, siendo este un valor numérico continuo y las etiquetas corresponden a valores numéricos, y (2) modelos de clasificación, donde el objetivo del modelo es determinar a qué clase corresponde un determinado vector de entrada; para lograr esto se suele disponer de un modelo con tantas salidas como clases y de todas las salidas aquella que resulta con un mayor valor indica a qué clase pertenece un vector de entrada. Entre los principales modelos de ML se pueden mencionar Neural Network (NN), Support Vector Machine (SVM) y Decision Tree (DT). Estos modelos, orientados a la clasificación, pueden ser modificados para realizar regresión.

Los modelos de SVM se basan en la teoría de aprendizaje estadístico (Vapnik, 2006). La idea base de una SVM es buscar un hiperplano separador con el máximo margen entre subconjuntos de instancias de distinto tipo o clase (etiqueta). Esto se logra mediante una función denominada kernel para transformar los datos de entrada a una dimensionalidad mayor (espacio n-dimensional) (Hsu et al., 2008). Por otro lado, los modelos DT son estructurados por una serie de restricciones o condiciones que están organizadas jerárquicamente y son sucesivamente aplicadas desde un punto inicial denominado raíz, avanzando en bifurcaciones, hasta un punto final denominado hoja. La representación gráfica es similar a la forma de un árbol invertido (Rodriguez-Galiano et al., 2015). Por último, NN es un modelo que se organiza en capas, cada una compuesta por unidades de cómputo, que realizan transformaciones sucesivas al vector de entrada. NN corresponden a uno de los métodos más empleados en la literatura revisada y son la antesala para analizar el enfoque de DL, por ello se describen estos modelos en la sección 2.3.1.

Uno de los principales problemas en ML es la dualidad del entrenamiento (u optimización) y la generalización. En el entrenamiento se ajusta el modelo para tener el mejor desempeño posible en el conjunto de datos de entrenamiento, mientras que la generalización se refiere a que tan bien el modelo, ya entrenado, se desempeña sobre instancias de datos que no ha visto antes (Chollet, 2017). La capacidad de generalización permite diferenciar un modelo que memoriza los datos de entrenamiento, de uno que aprende los patrones o relaciones en ellos y, el aprendizaje de los patrones en los datos permite realizar estimaciones precisas.

El Sobreajuste (Overfitting) es un problema de memorización, donde el modelo se ajusta demasiado al conjunto empleado para el entrenamiento y por lo tanto empeora su desempeño en otro conjunto de datos. A mayor overfitting menor capacidad de generalización. Se conoce como método de regularización a las técnicas en las que se realiza alguna modificación en un algoritmo de aprendizaje para mejorar la capacidad de generalización.

En el aprendizaje supervisado existen distintos métodos que permiten validar el modelo generado y su capacidad de generalización. Uno de estos métodos se denomina Holdout, y consiste en emplear un conjunto de datos que dispone de etiquetas y dividirlo en dos subconjuntos excluyentes: el primer subconjunto es empleado para ajustar los parámetros del modelo, proceso que se denomina entrenamiento; el segundo es empleado para comparar las estimaciones del modelo con las etiquetas, proceso denominado prueba. Adicionalmente, se suele separar un tercer subconjunto que es empleado para ajustar manualmente parámetros del modelo que no son entrenables, denominados hiperparámetros, lo que se realiza de forma sucesiva, previo a la prueba final del modelo. Los resultados sobre el conjunto de prueba se consideran una aproximación a la capacidad de generalización del modelo, es decir, de su rendimiento esperado sobre casos nuevos, de allí la importancia de "ocultar" una parte de los datos durante el proceso de entrenamiento. Otros métodos más elaborados son k-Fold Cross Validation, Leave-One-Out Cross-Validation y Stratified k-Fold Cross Validation, que dividen los datos de diferentes maneras, pero mantienen la idea base de entrenamiento y prueba con subconjuntos excluyentes (Arlot and Celisse, 2010).

El desempeño de los modelos se mide de manera concreta mediante métricas. En el caso de aprendizaje supervisado las etiquetas se comparan con las estimaciones del modelo. Esta

comparación difiere según se realice regresión o clasificación. Las métricas más empleadas para regresión son Mean Absolute Error (MAE) y Mean Squared Error (MSE), descritas en las ecuaciones 2.8 y 2.9, respectivamente. Donde I representa la cantidad de instancias para las que se realizan estimaciones.

$$MAE = \frac{1}{I}\sum_{i=1}^{I}|y_i - \hat{y}_i| \tag{2.8}$$

$$MSE = \frac{1}{I}\sum_{i=1}^{I}(y_i - \hat{y}_i)^2 \tag{2.9}$$

En el caso de clasificación, algunas de las métricas más empleadas son Accuracy, Precision y Recall, descritas en las ecuaciones 2.10, 2.11 y 2.12. Estas métricas se basan en un conteo, en un escenario donde existe la clase positivo y negativo, de modo que, para una entrada determinada, si el modelo estima que pertenece a la clase positivo y acierta, se denomina verdadero positivo (VP); de lo contrario, si el modelo estima positivo y falla, se denomina falso positivo (FP); si el modelo estima que pertenece a la clase negativo y acierta, se denomina verdadero negativo (VN); y de lo contrario si estima que pertenece a la clase negativo y falla, se denomina falso negativo (FN).

$$Accuracy = \frac{VP + VN}{VP + VN + FP + FN} \tag{2.10}$$

$$Precision = \frac{VP}{VP + FP} \tag{2.11}$$

$$Recall = \frac{VP}{VP + FN} \tag{2.12}$$

Cada métrica aporta información de la capacidad de un modelo desde puntos de vista diferentes, por lo que se complementan. Mientras Accuracy es el número de predicciones correctas realizadas por el modelo sobre todas las estimaciones realizadas, la precisión es intuitivamente la capacidad del clasificador de no etiquetar como positiva una instancia que es negativa y Recall, también llamada sensibilidad, es intuitivamente la capacidad del clasificador para encontrar todas las instancias positivas. También, debido a las características de cada métrica, adquieren mayor o menor relevancia en base al problema que se busque resolver.

Otro aspecto relevante es la comparación del desempeño entre modelos, en ciertas disciplinas la comparación entre modelos se realiza mediante los puntajes obtenidos en las métricas de desempeño, lo que se puede complementar con la aplicación de test estadísticos como Wilcoxon (García-Floriano et al., 2018).

2.3. Deep Learning

Los modelos de Machine Learning (ML) funcionan bastante bien en una variedad de problemas importantes. Sin embargo, estos modelos no han tenido éxito en resolver problemas centrales de Inteligencia Artificial, como reconocimiento de voz o reconocimiento de objetos. El desarrollo del Deep Learning (DL) como subcampo del ML, ha sido motivado en parte por la falta de capacidad de modelos tradicionales para alcanzar una buena generalización en dichas tareas de Inteligencia Artificial. Esto ocurre debido a que el desarrollo de modelos con capacidad de generalización se vuelve más difícil cuando se trabaja con datos de alta dimensión y como los mecanismos utilizados para lograr la generalización en ML tradicional son insuficientes para aprender funciones complejas en espacios de alta dimensión, estos espacios también suelen imponer altos costos computacionales. DL se diseñó para superar estos y otros obstáculos (Goodfellow et al., 2016).

A continuación se presentan los modelos DL empleados en esta tesis. Se inicia con una explicación del funcionamiento de las Redes Neuronales para introducir conceptos relacionados con DL. Luego, se presentan conceptos relacionados con las Redes Neuronales Recurrentes o Recurrent Neural Network (RNN) y Redes Neuronales Convolucionales o Convolutional Neural Network (CNN), donde se explica parte de su funcionamiento y se presentan las variaciones implementadas Long Short-Term Memory (LSTM) y CNN de una dimensión.

2.3.1. Feedforward Neural Networks

El modelo de Neural Network (NN) está compuesto por unidades de cómputo, usualmente denominadas neuronas artificiales. Dentro del modelo, las neuronas son organizadas en capas y cada capa puede tener un número diferente de neuronas. En general, todas las neuronas dentro de una misma capa realizan su cómputo en paralelo. Existen variados tipos de NN que se diferencian en base al tipo de neurona que compone la red, en base a la forma en la que fluye la información dentro de la red, etcétera. Una red bastante representativa es la Feedforward Neural Network (FNN) que recibe una entrada $\boldsymbol{x}$ que brinda información inicial y produce una salida $\hat{y}$, la información fluye desde la primera capa, avanzando a través de la red de capa en capa hasta llegar a la capa de salida (Goodfellow et al., 2016). Dentro de los modelos FNN existe un modelo denominado Densely-Connected Neural Network (o Fully-Connected Neural Network) donde cada neurona en una capa tiene conexiones con cada una de las neuronas en la siguiente capa, este modelo es muy representativo y se suele omitir la diferencia, siendo mencionado como FNN.

Todo modelo FNN tiene una capa de entrada, capas ocultas y una capa de salida. La capa de entrada es la primera capa del modelo y como su nombre lo indica es la capa por donde ingresa la información. El tamaño de la capa de entrada, depende del tamaño del vector de entrada. La capa de salida es la última capa que, en problemas de regresión, suele estar compuesta por solo una neurona mientras que en problemas de clasificación se emplean tantas neuronas como clases a identificar. Las capas ocultas son todas las capas entre las capas de entrada y de salida. Estas partes de un modelo FNN están señaladas en

la Figura 2.2, en donde se presenta un modelo de tres capas ocultas.

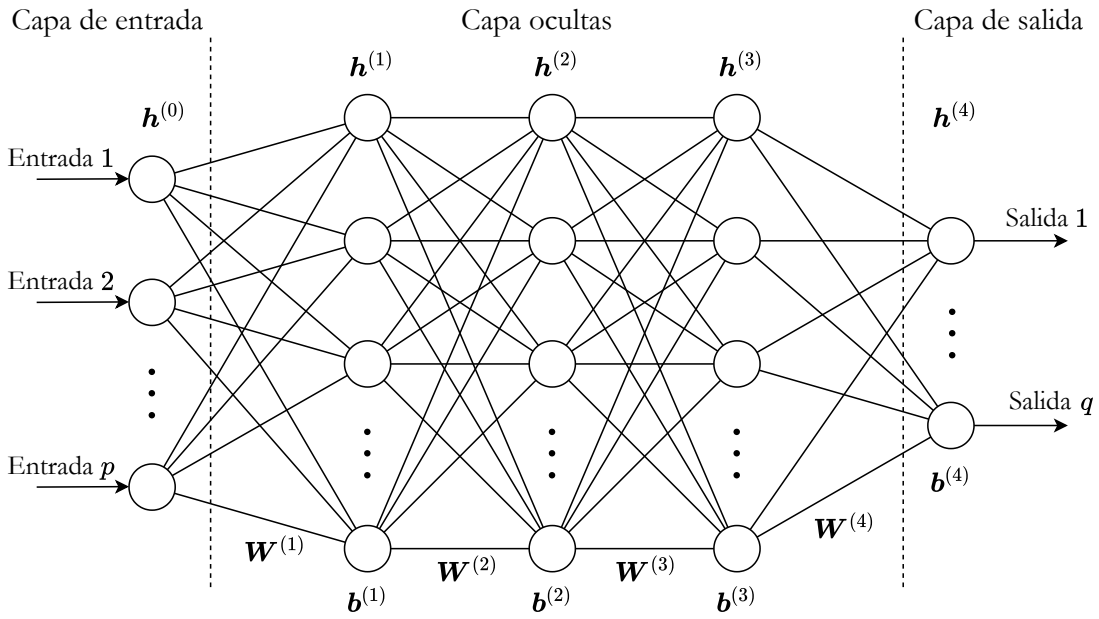


Figura 2.2: Ejemplo de Feedforward Neural Network

Las capas ocultas y la capa de salida están compuestas por neuronas. Cada neurona recibe una serie de valores de la capa anterior, procesan estos valores y generan un valor de salida. Los valores generados por cada neurona en una capa son transmitidos a cada una de las neuronas en la siguiente capa. En general, una neurona artificial asocia un peso a cada conexión con las neuronas de una capa anterior. Estos pesos son empleados para realizar una suma ponderada con los valores de entrada. A esta suma ponderada se agrega un valor denominado sesgo y el resultado es entregado como argumento a una función, denominada función de activación.

En la Figura 2.3 se presenta una neurona artificial, donde cada valor que ingresa está representado por x_p, donde $p = 1, \ldots, P$, con sus respectivos pesos representados por w_p asociados a cada conexión. Una vez ingresan valores a la neurona se realiza una suma ponderada, a la que se suma el sesgo, representado por b. El sesgo y los pesos son los parámetros que se deben ajustar en el proceso de entrenamiento para mejorar el desempeño de una Red Neuronal. El sesgo es un valor que permite ajustar el resultado a obtener de la función de activación. La función de activación, representada por f, actúa como filtro y condiciona la salida en base a la suma ponderada y el sesgo.

Si bien se puede realizar un análisis de la arquitectura de una red al nivel de una neurona, los cálculos se realizan por capas, y se suele definir en términos de matrices y vectores, que es más cercano a la forma en que se trabajan los datos en la implementación de los modelos. Debido a esto, se puede definir un modelo FNN como un conjunto $\mathbb{H}$ de

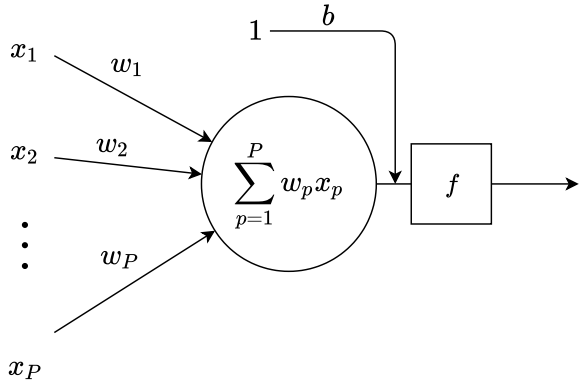


Figura 2.3: Ejemplo de neurona artificial.

capas $\boldsymbol{h}^{(d)}$ con $d = 1, \ldots, D$, donde la primera capa en el modelo es representada por $\boldsymbol{h}^{(0)}$. Los pesos, que a nivel de una neurona podrían ser representados por un vector, en una capa son representados por una matriz y los sesgos en una capa son agrupados en un vector.

De acuerdo con la ecuación 2.13 una capa $\boldsymbol{h}^{(d)}$ tiene una matriz de pesos $\boldsymbol{W}^{(d)}$ y un vector $\boldsymbol{b}^{(d)}$ asociados. Cada elemento en la matriz de pesos está asociado a una conexión entre una capa oculta $\boldsymbol{h}^{(d)}$ y la capa anterior. Las dimensiones de una matriz de pesos son $n \times m$ donde n es la cantidad de neuronas en la capa oculta $\boldsymbol{h}^{(d)}$ y m es la cantidad de neuronas en la capa anterior. Por otro lado, los sesgos son representados por un vector $\boldsymbol{b}^{(d)}$ de n elementos, donde cada valor de sesgo está asociado a su respectiva neurona en la capa oculta $\boldsymbol{h}^{(d)}$.

$$\boldsymbol{W}^{(d)} = \begin{bmatrix} W_{1,1}^{(d)} & W_{1,2}^{(d)} & \ldots & W_{1,m}^{(d)} \\ W_{2,1}^{(d)} & W_{2,2}^{(d)} & \ldots & W_{2,m}^{(d)} \\ \vdots & \vdots & \ddots & \vdots \\ W_{n,1}^{(d)} & W_{n,2}^{(d)} & \ldots & W_{n,m}^{(d)} \end{bmatrix}; \quad \boldsymbol{b}^{(d)} = \begin{bmatrix} b_1^{(d)} \\ b_2^{(d)} \\ \vdots \\ b_n^{(d)} \end{bmatrix}; \quad \boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_p \end{bmatrix} \tag{2.13}$$

Entonces las operaciones realizadas por cada neurona en la d-ésima capa pueden ser representadas mediante la Ecuación 2.14. Esta ecuación considera los elementos que componen una neurona artificial antes mencionados, una función de activación, pesos, sesgos y una entrada. En la ecuación, el vector de entrada para una capa $\boldsymbol{h}^{(d)}$ corresponde a la salida generada por la capa anterior $\boldsymbol{h}^{(d-1)}$, lo que captura el funcionamiento de una FNN donde la información fluye de capa en capa. En el caso de la primera capa oculta $\boldsymbol{h}^{(1)}$ el elemento $\boldsymbol{h}^{(d-1)}$ en la ecuación correspondería al vector de entrada $\boldsymbol{x}$.

$$\boldsymbol{h}^{(d)} = f^{(d)}\left(\boldsymbol{W}^{(d)}\boldsymbol{h}^{(d-1)} + \boldsymbol{b}^{(d)}\right) \tag{2.14}$$

La función de activación tiene un rol importante en la capacidad que tiene una Red Neuronal. Sin una función de activación la capa descrita en la Ecuación 2.14 constaría de un

producto punto (producto entre $\boldsymbol{W}$ y $\boldsymbol{x}$) y una adición (adición de sesgo), dos operaciones lineales. La capa solo sería capaz de aprender transformaciones lineales de los valores de entrada. Por lo tanto, una función no lineal o de activación es necesaria para ampliar la capacidad de un modelo, pues brinda la habilidad de capturar relaciones no lineales. Rectified Linear Unit (ReLU) (Ecuación 2.15) es un ejemplo de función de activación.

$$ReLu(z) = max(0, z) \tag{2.15}$$

La salida de la última capa $\boldsymbol{h}^{(D)}$ para un modelo de D capas corresponde a una estimación $\hat{y}$. En el caso de una regresión, esta salida correspondería a un valor escalar, mientras que en una tarea de clasificación, correspondería a un vector compuesto por un valor asociado a cada clase. En la Ecuación 2.16 se puede ver el desarrollo completo de la última capa, que en un inicio involucra la capa anterior $D-1$ (penúltima capa), que a su vez requiere de la capa anterior $D-2$, lo que se repite hasta llegar a la primera capa que contempla el vector de entrada $\boldsymbol{x}$. En esencia la última capa es el resultado de la ejecución iterativa de funciones que realizan transformaciones en los datos.

$$\begin{aligned}
\hat{y} &= \boldsymbol{h}^{(D)} \\
&= f^{(D)}\left(\boldsymbol{W}^{(D)}\boldsymbol{h}^{(D-1)} + \boldsymbol{b}^{(D)}\right) \\
&= f^{(D)}\left(\boldsymbol{W}^{(D)}\left[f^{(D-1)}(\boldsymbol{W}^{(D-1)}\boldsymbol{h}^{(d-2)} + \boldsymbol{b}^{(D-1)})\right] + \boldsymbol{b}^{(D)}\right) \\
&= f^{(D)}\left(\boldsymbol{W}^{(D)}\left[f^{(D-1)}(\boldsymbol{W}^{(D-1)}\ldots[\boldsymbol{W}^{(1)}\boldsymbol{x} + \boldsymbol{b}^{(1)}]\ldots + \boldsymbol{b}^{(d-2)})\right] + \boldsymbol{b}^{(D-1)}\right)
\end{aligned} \tag{2.16}$$

En el entrenamiento de una NN, se busca encontrar el conjunto de valores para sus parámetros (pesos y sesgos), de modo que, para cada vector de entrada, la salida producida por la red sea igual (o muy parecida) a la salida esperada. Un algoritmo ampliamente empleado para la búsqueda de estos valores es Back-propagation (Rumelhart et al., 1986). Este algoritmo establece que mediante la definición de una función de error y su optimización es posible calcular el ajuste de sus parámetros.

La función de error se suele denominar "función de costo" y su optimización se realiza mediante el cálculo del gradiente. Los valores que genera la función de costo dependen de los valores que tengan los parámetros de la red y el gradiente permite determinar cómo deben ajustarse los parámetros para minimizar la función de costo. En la Figura 2.4 se muestra un ejemplo para un modelo que depende solo de un parámetro y, en base a los valores que adopta este parámetro, se forma una curva de los valores que genera la función de costo. Dado un valor específico en los parámetros, el cálculo del gradiente permite obtener la dirección en la que los parámetros deben ser ajustados. Repetir el proceso del cálculo de gradiente permite moverse en la curva hasta llegar a un costo mínimo. Esto se puede observar en la Figura 2.4.

En la práctica, una red tendrá una enorme cantidad de parámetros, y cada parámetro, puede adquirir una variedad de valores. Debido a esto, cada parámetro puede considerarse una dimensión. Si un parámetro permite formar una curva en dos dimensiones (Figura 2.4),

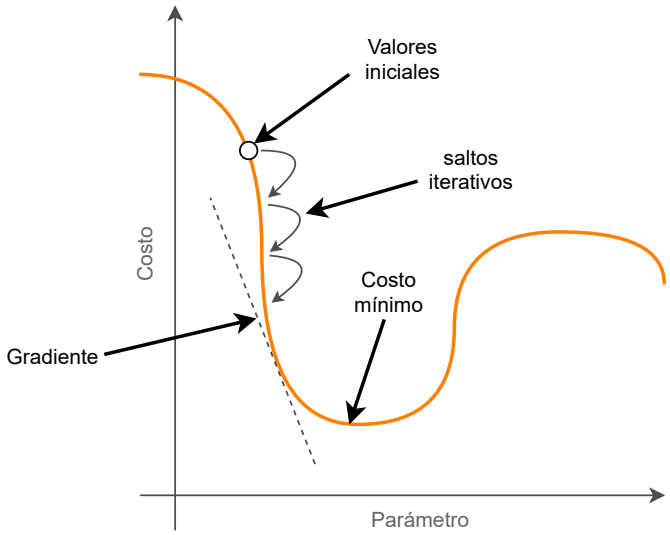


Figura 2.4: Descenso del gradiente para un parámetro. Basado en (Chollet, 2017)

dos parámetros generarían un espacio tridimensional y así sucesivamente. Cada conjunto de parámetros constituirá un espacio distinto y los valores que tengan los parámetros en un momento determinado, corresponderán a un punto en este espacio. El descenso del gradiente permite navegar en este espacio de soluciones, esto en busca de los valores adecuados para los parámetros de la red, que generen un costo mínimo.

La función de costo J compara la salida esperada y (etiqueta), con la salida de la red $\boldsymbol{h}^{(D)}$, como se muestra en la Ecuación 2.17.

$$J = Costo(y, \hat{y}) \tag{2.17}$$

La función de costo varía según el tipo de problema y suelen ser métricas de desempeño como MAE o MSE, tal como se muestra en la Ecuación 2.18, que presenta un ejemplo empleando la función MSE, donde I es el tamaño del conjunto de instancias, $\hat{y}_i$ la salida generada para la instancia i y y_i la etiqueta correspondiente.

$$J = MSE = \frac{1}{I}\sum_{i=1}^{I}(y_i - \hat{y}_i)^2 \tag{2.18}$$

El cálculo del gradiente de la función de costo (∇J) se puede realizar por separado para cada instancia i y luego calcular un promedio, tal como se muestra en la Ecuación 2.19. Para el ejemplo de función de costo en la Ecuación 2.18 su forma J_i para una instancia i se presenta en la Ecuación 2.20, donde el índices i ahora está implícito. Este cambio se realiza en base a la regla de derivada de sumas[1].

[1]La regla de suma en derivadas establece que la derivada de la suma de dos funciones diferenciables es igual a la suma de la derivación de cada función por separado: $\frac{d(u+v)}{dx} = \frac{du}{dx} + \frac{dv}{dx}$

$$\nabla J = \frac{1}{I}\sum_{i=1}^{I} \nabla J_i \tag{2.19}$$

$$J_i = (y - \hat{y})^2 \tag{2.20}$$

El cálculo del gradiente ∇J_i se descompone en derivadas parciales respecto a cada parámetro en la red y para su implementación se pueden agrupar por capas, tal como se muestra en la Ecuación 2.21.

$$\nabla J_i = \left[\frac{\partial J_i}{\partial W^{(D)}}, \frac{\partial J_i}{\partial b^{(D)}}, \ldots, \frac{\partial J_i}{\partial W^{(d)}}, \frac{\partial J_i}{\partial b^{(d)}}, \ldots, \frac{\partial J_i}{\partial W^{(1)}}, \frac{\partial J_i}{\partial b^{(1)}}\right] \tag{2.21}$$

El detalle de cada derivada parcial en una d-ésima capa se observa en la Ecuación 2.22. Cada $W^{(d)}_{j,k}$ es un peso, donde j es el índice de las neuronas de la d-ésima capa y k es el índice de las neuronas de la capa anterior. Cada $b^{(d)}_j$ corresponde a los sesgos, donde j es el índice de las neuronas de la d-ésima capa. Para efectos prácticos los cálculos se manipulan a nivel de capa en torno a vectores y matrices.

$$\frac{\partial J_i}{\partial W^{(d)}_{jk}} = \begin{bmatrix} \frac{\partial J_i}{\partial W^{(d)}_{1,1}} & \frac{\partial J_i}{\partial W^{(d)}_{1,2}} & \cdots & \frac{\partial J_i}{\partial W^{(d)}_{1,m}} \\ \frac{\partial J_i}{\partial W^{(d)}_{2,1}} & \frac{\partial J_i}{\partial W^{(d)}_{2,2}} & \cdots & \frac{\partial J_i}{\partial W^{(d)}_{2,m}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial J_i}{\partial W^{(d)}_{n,1}} & \frac{\partial J_i}{\partial W^{(d)}_{n,2}} & \cdots & \frac{\partial J_i}{\partial W^{(d)}_{n,m}} \end{bmatrix}; \quad \frac{\partial J_i}{\partial b^{(d)}_j} = \begin{bmatrix} \frac{\partial J_i}{\partial b^{(d)}_1} \\ \frac{\partial J_i}{\partial b^{(d)}_2} \\ \vdots \\ \frac{\partial J_i}{\partial b^{(d)}_n} \end{bmatrix} \tag{2.22}$$

Resolver cualquiera de las derivadas parciales requiere de la regla de la cadena[2] en derivadas. Si se considera el gradiente de la función de costo para los pesos de la última capa D resulta en la Ecuación 2.23, esto dado que la función de costo J incluye $h(D)$, que a su vez incluye $W^{(D)}$.

$$\frac{\partial J}{\partial \boldsymbol{W}^{(D)}} = \frac{\partial J}{\partial \boldsymbol{h}^{(D)}} \frac{\partial \boldsymbol{h}^{(D)}}{\partial \boldsymbol{W}^{(D)}} \tag{2.23}$$

De la Ecuación 2.23 todos los términos son conocidos y derivables, pero es conveniente redefinir $h^{(d)}$ para los cálculos posteriores, de modo que:

$$\boldsymbol{h}^{(d)} = f^{(d)}\left(\boldsymbol{z}^{(d)}\right) \tag{2.24}$$

$$\boldsymbol{z}^{(d)} = \boldsymbol{W}^{(d)}\boldsymbol{h}^{(d-1)} + \boldsymbol{b}^{(d)} \tag{2.25}$$

De esta forma, la ecuación 2.23 se puede desarrollar más y resulta en:

[2]La regla de la cadena en derivadas establece que dadas dos funciones diferenciables $v = f(u)$ y $u = g(x)$ la derivada de v respecto a x será igual a $\frac{dv}{dx} = \frac{dv}{du}\frac{du}{dx}$

$$\frac{\partial J}{\partial \boldsymbol{W}^{(D)}} = \frac{\partial J}{\partial \boldsymbol{h}^{(D)}} \frac{\partial \boldsymbol{h}^{(D)}}{\partial \boldsymbol{z}^{(D)}} \frac{\partial \boldsymbol{z}^{(D)}}{\partial \boldsymbol{W}^{(D)}} \tag{2.26}$$

Calcular las derivadas parciales en la Ecuación 2.26 es más cómodo, ya que, el término $\frac{\partial \boldsymbol{h}^{(D)}}{\partial \boldsymbol{z}^{(D)}}$ es la derivada de las funciones de activación y su cálculo se puede realizar por separado. Si se continua con la derivada parcial en la capa $D-1$ (aplicando de forma reiterada la regla de la cadena) resulta la Ecuación 2.27.

$$\frac{\partial J}{\partial \boldsymbol{W}^{(D-1)}} = \frac{\partial J}{\partial \boldsymbol{h}^{(D)}} \frac{\partial \boldsymbol{h}^{(D)}}{\partial \boldsymbol{z}^{(D)}} \frac{\partial \boldsymbol{z}^{(D)}}{\partial \boldsymbol{h}^{(D-1)}} \frac{\partial \boldsymbol{h}^{(D-1)}}{\partial \boldsymbol{z}^{(D-1)}} \frac{\partial \boldsymbol{z}^{(D-1)}}{\partial \boldsymbol{W}^{(D-1)}} \tag{2.27}$$

Si se comparan las Ecuaciones 2.26 y 2.27, se observa que las primeras dos derivadas parciales aparecen en ambas. Estos términos corresponden al error de la última capa D que se transfiere a la capa anterior $D-1$. Este error se suele representar con $\delta^{(d)}$ y para la última capa corresponde a la Ecuación 2.28

$$\delta^{(D)} = \frac{\partial J}{\partial \boldsymbol{h}^{(D)}} \frac{\partial \boldsymbol{h}^{(D)}}{\partial \boldsymbol{z}^{(D)}} \tag{2.28}$$

Este error se transfiere de capa en capa, por lo que se va acumulando. El error que se transfiere de la capa $D-1$ a la capa $D-2$ incluiría $\delta^{(D)}$ como se muestra en la siguiente Ecuación

$$\delta^{(D-1)} = \delta^{(D)} \frac{\partial \boldsymbol{z}^{(D)}}{\partial \boldsymbol{h}^{(D-1)}} \frac{\partial \boldsymbol{h}^{(D-1)}}{\partial \boldsymbol{z}^{(D-1)}} \tag{2.29}$$

De esta manera el error en la estimación producida por la red se propaga hacia atrás (Back-propagation) desde la última capa hasta la primera, lo que permite asociar el error en la estimación producida por la última capa con la contribución que realiza cada neurona desde el inicio.

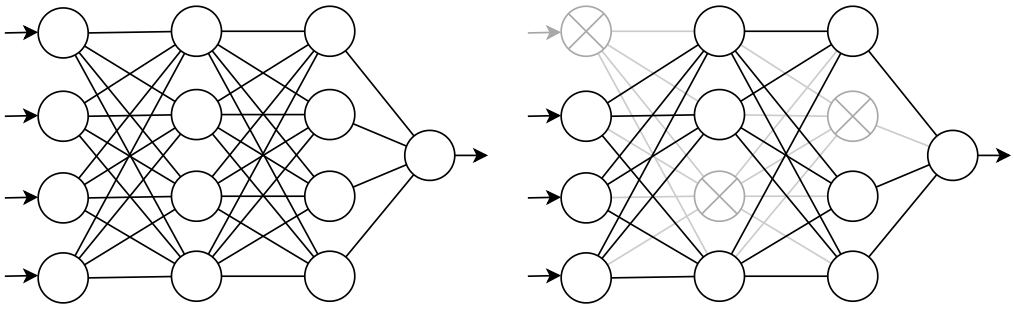
Una vez calculado el promedio del gradiente para todas las instancias, este se invierte, ya que el objetivo es minimizar la función de costo, tal como se muestra en la Ecuación 2.30, donde $\boldsymbol{\theta}$ representa todos los parámetros en la red, η es un factor que determina la contribución del ajuste calculado, este factor se denomina ratio de aprendizaje (del inglés, learning rate) y $\boldsymbol{\theta}^{'}$ los nuevos valores para los parámetros.

$$\boldsymbol{\theta}^{'} = \boldsymbol{\theta} - \eta \nabla J(\boldsymbol{\theta}) \tag{2.30}$$

Las Ecuaciones 2.19 y 2.30 implican que el ajuste de los parámetros se realiza en base al promedio del ajuste calculado para todo el conjunto de datos. Esta es una forma de realizar el descenso del gradiente que se denomina Batch Gradient Descent. Existe otra variación denominada Stochastic Gradient Descent (SGD) que consiste en calcular el ajuste para una instancia e inmediatamente ajustar los pesos, lo que se repite por cada instancia. Esta variación tiene un mejor desempeño (Ruder, 2016). Además, existen otras formas de optimizar el descenso del gradiente, estos métodos son descritos en la sección 4.3

Las Redes Neuronales también son propensas al sobreajuste y por ello se han propuesto

técnicas de regularización. Una de estas técnicas se denomina Dropout (Srivastava et al., 2014), que durante el proceso de entrenamiento ignora un porcentaje de neuronas y reduce a cero los valores de salida de estas neuronas en una capa. Las neuronas por ignorar se seleccionan de manera aleatoria. Un ejemplo sencillo de la aplicación de dropout sobre la capa de entrada y las capas ocultas en una Red Neuronal aparece en la Figura 2.5. El ignorar de forma aleatoria distintas neuronas en el proceso de entrenamiento, ralentiza el proceso de aprendizaje, pero impide que la estimación dependa de unas pocas neuronas.



(a) Red Neuronal sin dropout (b) Red Neuronal tras de aplicar dropout

Figura 2.5: Ejemplo de cambio tras el uso de dropout. Basado en (Srivastava et al., 2014)

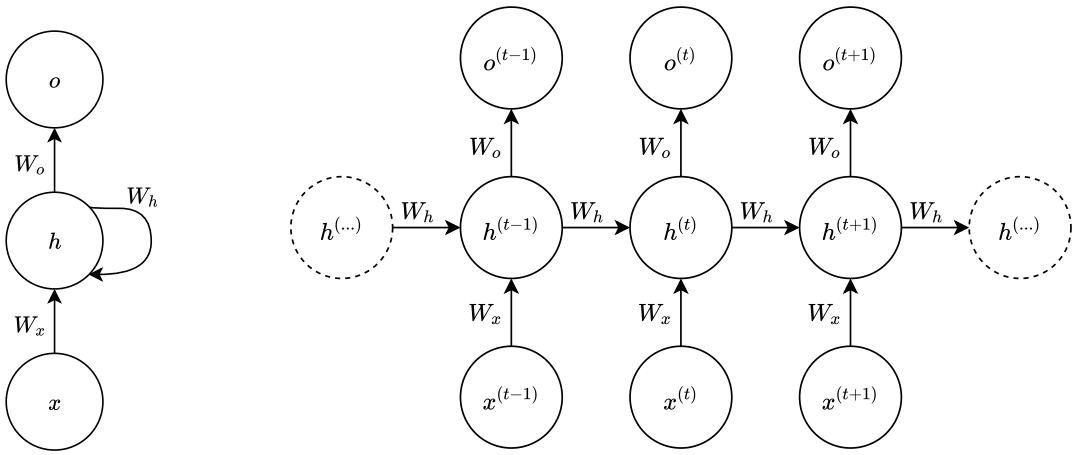
Muchos de los elementos que componen una red neuronal como el número de capas, el número de neuronas en cada capa, la cantidad de neuronas de salida, las funciones de activación por cada capa, etc. pueden ser definidos en base al problema a resolver y es esta versatilidad, entre otros factores, lo que permite que este modelo tenga resultados sobresalientes en diversas áreas del conocimiento.

2.4. Redes Neuronales Recurrentes

Como se describió antes, una FNN recibe vectores de entrada $\boldsymbol{x}_i$ y emplea pesos para generar una estimación $\hat{y}_i$ y mediante un proceso de entrenamiento se busca ajustar los pesos para que las estimaciones se acerquen al valor real y_i. En este esquema la Neural Network modela la relación entre el vector de entrada y la salida esperada, asumiendo que existe independencia entre vectores de entrada. Esto no es un inconveniente en la solución de diversos problemas, pero comienza a afectar la eficiencia del modelo en cuanto el orden en los datos resulta más relevante. Esto ocurre en la interpretación de texto, donde el orden en las palabras puede cambiar el significado de una oración, o en la estimación del clima, información que está sujeta a las estaciones del año o la hora del día, etcétera. En esta tesis los datos mantienen un orden espacial en la distribución de concentración de minerales y un modelo que sea capaz de capturar esta relación espacial debería tener el potencial de ofrecer una solución más eficiente.

Las Recurrent Neural Networks (RNN) son un tipo de NN que tienen la habilidad para que información persista en una secuencia de entradas. Esto es posible debido a que opera en forma iterativa, tal como se ve en la Figura 2.6a, donde se emplea el vector de entrada

para producir una salida, pero también mantiene una conexión de retroalimentación. Si desplegamos la forma compacta presentada en la Figura 2.6a resulta la Figura 2.6b, donde se puede apreciar que en diferentes iteraciones $t-1$, t y $t+1$ la Red Neuronal recibe un vector de entrada distinto y genera una salida distinta. Además, en cada iteración se recibe y transmite información de una iteración a otra, lo que se denomina estado h (Goodfellow et al., 2016).



(a) Versión resumida (b) Versión extendida

Figura 2.6: Gráfico computacional de red neuronal recurrente. Basado en (Goodfellow et al., 2016)

En la Figura 2.6a se aprecia un grafo que resume el flujo de información en una RNN, en esta red, la estructura puede separarse en tres partes, cada una parametrizada por un conjunto distinto de pesos: las conexiones entre unidades de entrada y unidades ocultas son parametrizadas por la matriz de pesos $\boldsymbol{W}_x$; las conexiones recurrentes entre unidades ocultas son parametrizadas por la matriz de pesos $\boldsymbol{W}_h$; y las conexiones entre unidades ocultas y la capa de salida son parametrizadas por $\boldsymbol{W}_o$.

Si nos ubicamos en una iteración t el estado $h^{(t)}$ será el resultado de la transformación de los datos de entrada ($\boldsymbol{W}_x\boldsymbol{x}^{(t)}$) más el estado de la iteración anterior $h^{(t-1)}$ ponderado por la matriz $\boldsymbol{W}_h$ más el sesgo $\boldsymbol{b}_h$, lo que corresponde a la Ecuación 2.31, donde f es una función de activación.

$$h^{(t)} = f\left(\boldsymbol{W}_x\boldsymbol{x}^{(t)} + \boldsymbol{W}_h h^{(t-1)} + \boldsymbol{b}_h\right) \tag{2.31}$$

El estado $h^{(t)}$ obtenido será usado en la iteración actual, donde es ponderado con los pesos $\boldsymbol{W}_o$ para generar la salida $o^{(t)}$ tal como se muestra en la Ecuación 2.32, donde $\boldsymbol{b}_o$ también son sesgos.

$$o^{(t)} = \boldsymbol{W}_o h^{(t)} + \boldsymbol{b}_o \tag{2.32}$$

La salida $o^{(t)}$ permite obtener una estimación $\hat{y}^{(t)}$, como se muestra en la Ecuación 2.33, donde $f_{\hat{y}}$ es otra función de activación. El estado $h^{(t)}$ también será empleado en la siguiente iteración. Finalmente, la repetición de este proceso, no solo permite que un estado anterior afecte al siguiente estado y a su vez una estimación, sino que además, los primeros estados mantienen una influencia en las últimas estimaciones, algo similar a un efecto dominó.

$$\hat{y}^{(t)} = f_{\hat{y}}(o^{(t)}) \tag{2.33}$$

Existen tres patrones principales de las RNN: (1) una RNN que genera una salida en cada tiempo t y que tiene conexiones recurrentes entre capas ocultas (caso que se abordó en párrafos anteriores y cuya representación gráfica se incluye en la Figura 2.6b); (2) una RNN que genera una salida en cada tiempo t y que solo tiene conexiones recurrentes desde la salida $o^{(t)}$ hasta las capas ocultas en un tiempo $t+1$; y (3) una RNN con conexiones recurrentes entre capas ocultas que lee una secuencia de vectores de entrada y solo tiene una salida (Goodfellow et al., 2016).

2.4.1. Long Short-Term Memory

En la estimación de recursos minerales las conexiones recurrentes y la transmisión de un estado entre iteraciones, puede permitir captar relaciones espaciales y la continuidad entre distintos puntos/nodos en el volumen en estudio. Long Short-Term Memory (LSTM) es un tipo de RNN que aumenta la capacidad de mantener estados generados en iteraciones anteriores, dado que una RNN estándar tiene bajo desempeño cuando existe una brecha de información relevante entre alguna iteración anterior y la iteración actual (Bengio et al., 1994). Por lo que un modelo LSTM resulta promisorio para construir un modelo que tenga en cuenta la dependencia espacial.

El aprendizaje en un contexto donde existe una brecha de información relevante entre iteraciones, se vuelve complejo principalmente debido a que el gradiente en el proceso de back-propagation (necesario para la adecuada actualización de parámetros) se puede volver muy pequeño (problema denominado vanishing gradient) o muy grande (problema denominado exploding gradients). Para enfrentar este problema (Hochreiter and Schmidhuber, 1997) proponen el modelo LSTM que incluye una serie de aspectos adicionales al enfoque RNN mencionado en la sección anterior.

Si realizamos un acercamiento a la Figura 2.6b resaltando los flujos de información en una RNN y su funcionamiento interno resulta la Figura 2.7. Cada flujo se corresponde con la Ecuación 2.31, en el centro la función de activación se alimenta de un vector de entrada más el estado generado en la iteración anterior, para generar un nuevo estado y una salida.

Por otro lado, el algoritmo LSTM presenta una estructura más compleja al incluir una compuerta de entrada $i^{(t)}$, una compuerta de salida $o^{(t)}$ y un estado interno $E^{(t)}$ separado del estado de salida $h^{(t)}$ en cada iteración. La compuerta de entrada permite proteger el contenido en memoria de perturbaciones generadas por entradas irrelevantes. De la misma manera, la compuerta de salida protege de perturbaciones generadas por información irrelevante ya presente en memoria. El estado interno ya no tiene una relación tan cercana con la salida generada en cada iteración, pasando de una iteración a otra

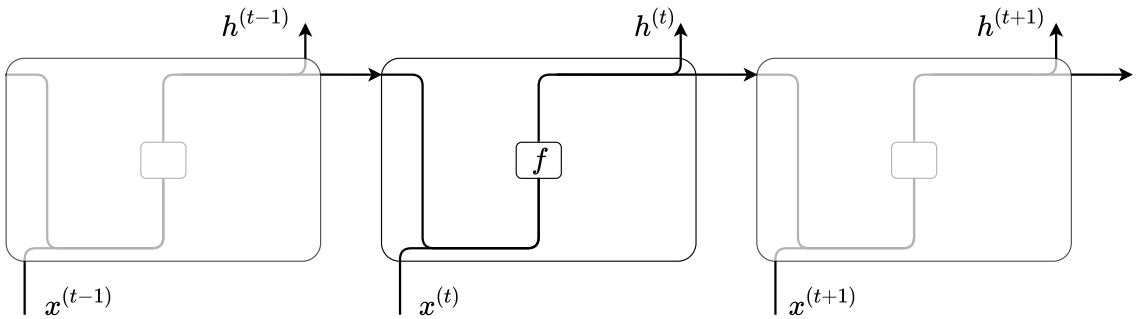


Figura 2.7: Célula RNN. Basado en (Olah, 2015)

sufriendo leves cambios. Con todo lo mencionado un diagrama de los flujos de información y su funcionamiento interno resultaría en la Figura 2.8

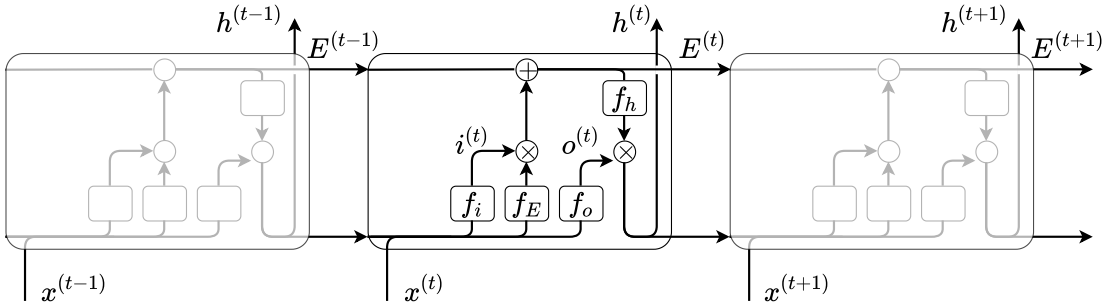


Figura 2.8: Célula original LSTM

El primer paso en el flujo de datos es la generación de valores candidatos que serán sumados al estado interno proveniente de la iteración anterior $E^{(t-1)}$. Estos valores candidatos $\tilde{E}^{(t)}$ se generan mediante la Ecuación 2.34, donde f_E es una función de activación como tanh, $\boldsymbol{W}_E$ son pesos y $\boldsymbol{b}_E$ son los sesgos.

$$\tilde{E}^{(t)} = f_E\left(\boldsymbol{W}_E[\boldsymbol{x}^{(t)}, h^{(t-1)}] + \boldsymbol{b}_E\right) \tag{2.34}$$

El segundo paso es seleccionar qué nueva información será incluida en el estado interno mediante la compuerta de entrada, una capa definida por la Ecuación 2.35 donde f_i es una función de activación como la función sigmoide, $\boldsymbol{W}_i$ son pesos y $\boldsymbol{b}_i$ son los sesgos. Los valores generados por una función sigmoide se encuentran entre 0 y 1, por lo que al multiplicar el resultado de la compuerta de entrada por los candidatos, un 0 implicaría el descarte de un valor y un 1 implicaría conservarlo sin cambios.

$$i^{(t)} = f_i\left(\boldsymbol{W}_i[\boldsymbol{x}^{(t)}, h^{(t-1)}] + \boldsymbol{b}_i\right) \tag{2.35}$$

El siguiente paso es actualizar el estado de acuerdo con la Ecuación 2.36 al sumar los valores candidatos una vez que han sido filtrados mediante la compuerta de entrada.

$$E^{(t)} = E^{(t-1)} + i^{(t)}\tilde{E}^{(t)} \tag{2.36}$$

El estado de salida, definido en la Ecuación 2.38, es generado una vez que el nuevo estado $E^{(t)}$ es sometido a la función de activación f_h, como la función tanh para que los valores estén entre -1 y 1. El resultado de esta transformación es filtrado por la compuerta de salida, capa definida por la Ecuación 2.37, donde f_o es una función de activación como la función sigmoide, $\boldsymbol{W}_o$ son pesos y $\boldsymbol{b}_o$ son los sesgos.

$$o^{(t)} = f_o\left(\boldsymbol{W}_o[\boldsymbol{x}^{(t)}, h^{(t-1)}] + \boldsymbol{b}_o\right) \tag{2.37}$$

$$h^{(t)} = o^{(t)} f_h\left(E^{(t)}\right) \tag{2.38}$$

En el trabajo de (Gers et al., 1999) se identifica un problema relacionado con el tratamiento de la memoria en una LSTM, donde determina que los mismos factores que le permiten aumentar la capacidad de mantener información entre varias iteraciones, producen una acumulación de información que desemboca en una saturación de memoria. Por esto, proponen la inclusión de una compuerta de olvido que permita filtrar los valores en el estado interno procedentes de una iteración anterior. La compuerta de olvido, definida en la Ecuación 2.37, comparte la misma estructura que las otras compuertas, donde f_σ es una función de activación como la función sigmoide, $\boldsymbol{W}_\sigma$ son pesos y $\boldsymbol{b}_\sigma$ son los sesgos.

$$\sigma^{(t)} = f_\sigma\left(\boldsymbol{W}_\sigma[\boldsymbol{x}^{(t)}, h^{(t-1)}] + \boldsymbol{b}_\sigma\right) \tag{2.39}$$

Este cambio en la ecuación del estado interno resulta en la Ecuación 2.40. Mientras que el diagrama con los flujos de información resulta en la Figura 2.9. Esta versión del algoritmo LSTM es la que se emplea en el desarrollo e implementación de algunos de los experimentos presentados en esta tesis.

$$E^{(t)} = \sigma^{(t)}E^{(t-1)} + i^{(t)}\tilde{E}^{(t)} \tag{2.40}$$

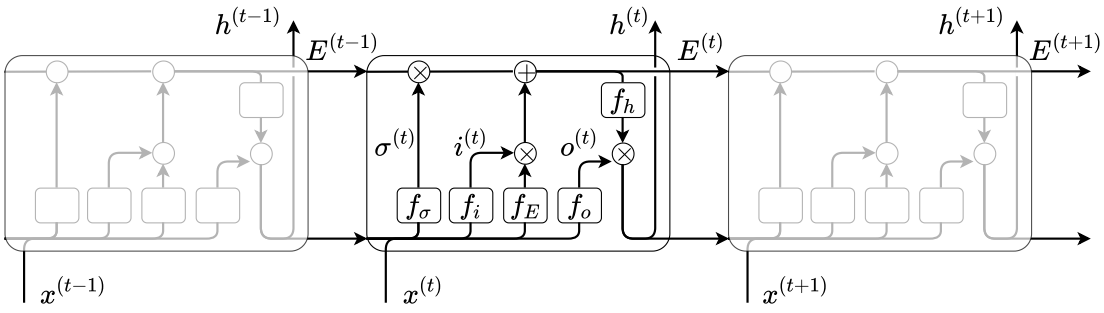


Figura 2.9: Célula LSTM con compuerta para olvido. Basado en (Olah, 2015)

2.5. Redes Neuronales Convolucionales

Otra de las principales arquitecturas en DL son las Redes Neuronales Convolucionales (Convolutional Neural Networks -convnets o CNN), empleadas principalmente para procesar datos que tienen una topología de dos dimensiones (2D) similar a una cuadrícula como las imágenes. Una CNN emplea una operación matemática llamada convolución, un tipo de operación lineal que consiste en aplicar una función que se denomina Kernel sobre datos de entrada, de modo que, la salida sea de menor tamaño y con ciertas características realzadas (Goodfellow et al., 2016).

En la Figura 2.10 se presenta un ejemplo de la operación de convolución, donde se aplica un kernel (2x2) sobre una imagen de entrada (3x4), donde el valor asociado a cada píxel es representado por letras. La convolución resulta del producto punto entre el kernel y una porción de la imagen. Los cuadrados con flechas indican cómo se forma el elemento superior izquierdo de la imagen de salida tras aplicar el kernel sobre la parte superior izquierda de la imagen de entrada.

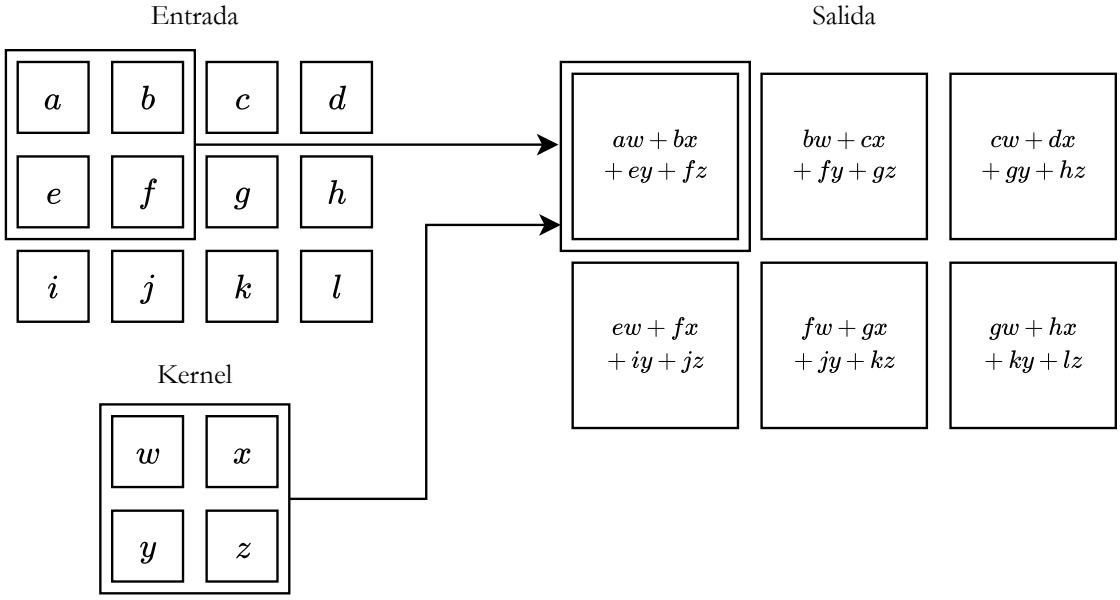


Figura 2.10: Ejemplo de operación convolución. Donde kernel se restringe a posiciones dentro de la imagen de entrada. Basado en (Goodfellow et al., 2016)

En la implementación de una CNN la entrada y salida se denominan "feature map". En la Figura 2.11 se presenta un ejemplo del funcionamiento de una capa convolucional, con un "feature map" de tres "canales" ya que, corresponde a una imagen en formato RGB (rojo, verde y azul) y para cada color hay un canal, que es una matriz del mismo tamaño que la imagen original con valores asociados a un color. El feature map de salida está determinado por las dimensiones de la imagen de entrada y por la cantidad de "filtros". Cada filtro está

constituido por un kernel distinto y codifica aspectos específicos de los datos de entrada. En la figura se están empleando 4 filtros, cada flujo indicado por las flechas representa un filtro distinto. En primera instancia se aplica la operación convolución en la imagen de entrada por cada color de forma independiente y luego se suman elemento por elemento para generar una única matriz denominada "response map" (Dumoulin and Visin, 2018). La profundidad o canales del feature map de salida dependerá de la cantidad de filtros.

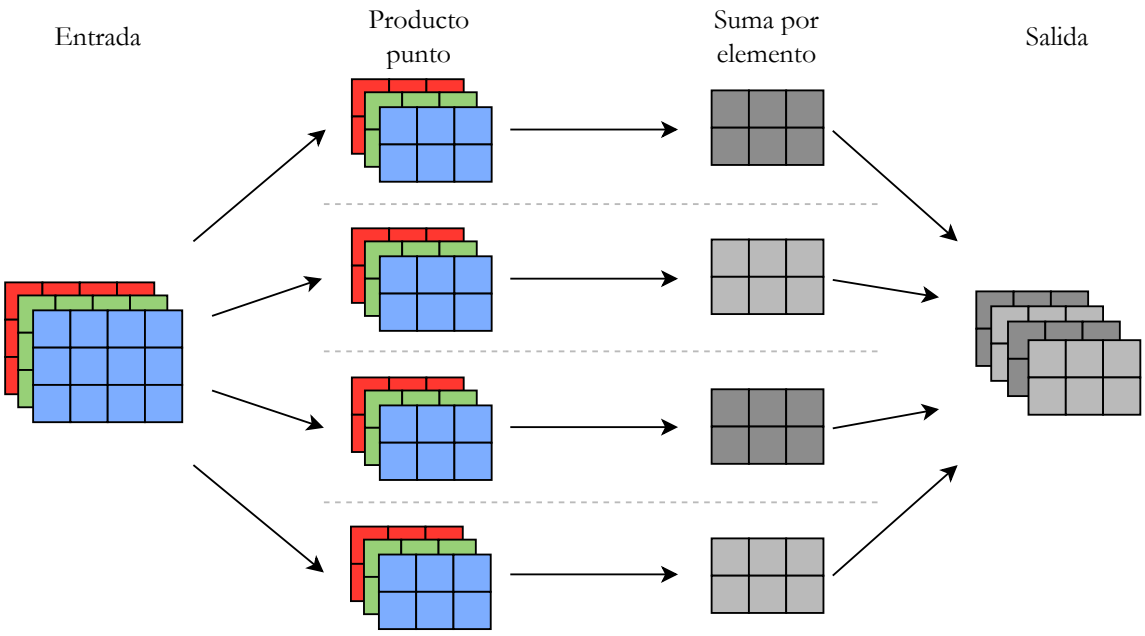
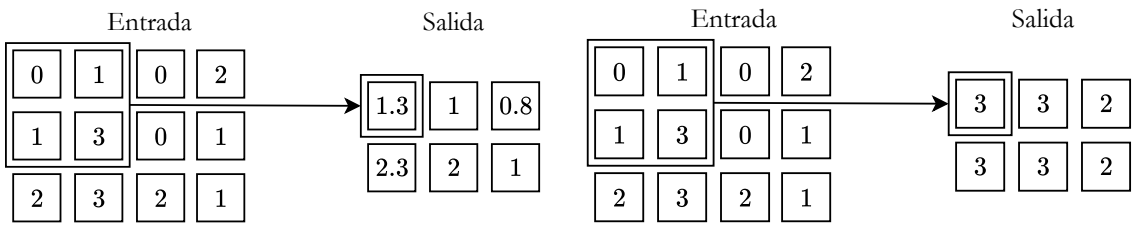


Figura 2.11: Funcionamiento de una capa de convolución. Basado en (Dumoulin and Visin, 2018)

En la práctica, una capa convolucional es implementada junto a otros tipos de capas, como las capas Pooling, que realizan operaciones para reducir una imagen de entrada. Estas operaciones no realizan un producto punto en base a un kernel, pero emplean una ventana tipo kernel sobre la que se selecciona el valor mayor (Max Pooling) o se genera un promedio (Average Pooling). Estas capas buscan reducir un feature map de forma drástica. En la Figura 2.12a se presenta un ejemplo donde se emplea una ventana 2x2 para realizar la operación Average Pooling en la imagen de entrada de 3x4. En la Figura 2.12b se presenta un ejemplo de la operación Max Pooling. En ambos ejemplos los cuadrados con flechas indican cómo se forma el elemento superior izquierdo de la imagen de salida tras aplicar la operación correspondiente. La aplicación reiterada de capas convolucionales en combinación con otros tipos de capas como las capas Pooling, permite extraer características de los datos, permitiendo representaciones modulares de forma eficiente.



(a) Average Pooling (b) Max Pooling

Figura 2.12: Ejemplo de operaciones Pooling

2.5.1. Redes Convolucionales 1D

El modelo CNN tiene una forma de procesar los datos que no está limitada a imágenes. Las operaciones que se realizan sobre los datos de dos dimensiones (2D) han sido adaptadas para poder trabajar con datos de 1D, en especial cuando se trata de secuencias de datos donde el orden es relevante, como palabras en una oración, datos de vibraciones en un dispositivo industrial, o cambios en dispositivos electrónicos como un acelerómetro (Goodfellow et al., 2016). El tiempo, espacio u orden es tratado en los datos como una dimensión, tal como el ancho o largo en una imagen. En el caso del problema presentado en esta tesis en donde se cuenta con nodos en un espacio tridimensional, un modelo como CNN tiene el potencial para tener en consideración relaciones espaciales que determinan la concentración mineral. De acuerdo con los resultados de la revisión de la literatura realizada por (Kiranyaz et al., 2020), recientemente se han publicados trabajos principalmente en las áreas de Reconocimiento Automático de Voz (Automatic Speech Recognition - ASR), Monitoreo de electrocardiogramas, Detección de Daño Estructural en base a vibraciones, entre otros, pero ningún trabajo se relaciona con la estimación de concentración mineral o algún problema en donde la relación espacial sea un aspecto relevante.

(Abdel-Hamid et al., 2014) es sino el primer trabajo que plantea el uso de una Red Convolucional para datos de una dimensión, específicamente para el ASR. Aquel trabajo define la operación de convolución para datos de una dimensión de acuerdo con la Ecuación 2.41, donde Q_j es el "response map" de salida que resulta de la aplicación de un filtro j ($j = 1, \ldots, J$) mediante la operación convolución sobre un "feature map" de entrada compuesto por canales O_i, con $i = 1, \ldots, A$ y f es una función de activación. A es la profundidad, donde cada i es un atributo. El símbolo $*$ representa la operación convolución entre O_i y el kernel $w_{i,j}$ que es un vector 1D, a diferencia del kernel en una operación 2D donde el kernel es una matriz.

$$Q_j = f\left(\sum_{i=1}^{A} O_i * w_{i,j}\right) \tag{2.41}$$

En la Figura 2.13 se representa un ejemplo de operación de convolución 1D, donde las instancias están ordenadas en base a su temporalidad, primero las más recientes y al

final las más antiguas. Cada columna es una instancia y cada fila un atributo o variable distinto (O_i). Se define una ventana de tamaño 5, lo que significa que se recorrerá la entrada considerando de a 5 instancias (columnas) para aplicar la operación convolución. Esto es similar a lo que se explicó en la sección anterior para una convolución 2D, donde se extraen partes 2D de una imagen a las que se les aplica una transformación idéntica. De la misma manera, para la operación de convolución 1D, la cantidad de atributos en la salida está determinada por la cantidad de filtros J a emplear (Chollet, 2017). Otras operaciones mencionadas en la sección anterior como Max Pooling y Average Pooling también pueden ser empleadas para una entrada de instancias ordenadas. Todas estas operaciones aplicadas sobre un grupo de instancias, permite resaltar características en los datos manteniendo su orden en consideración.

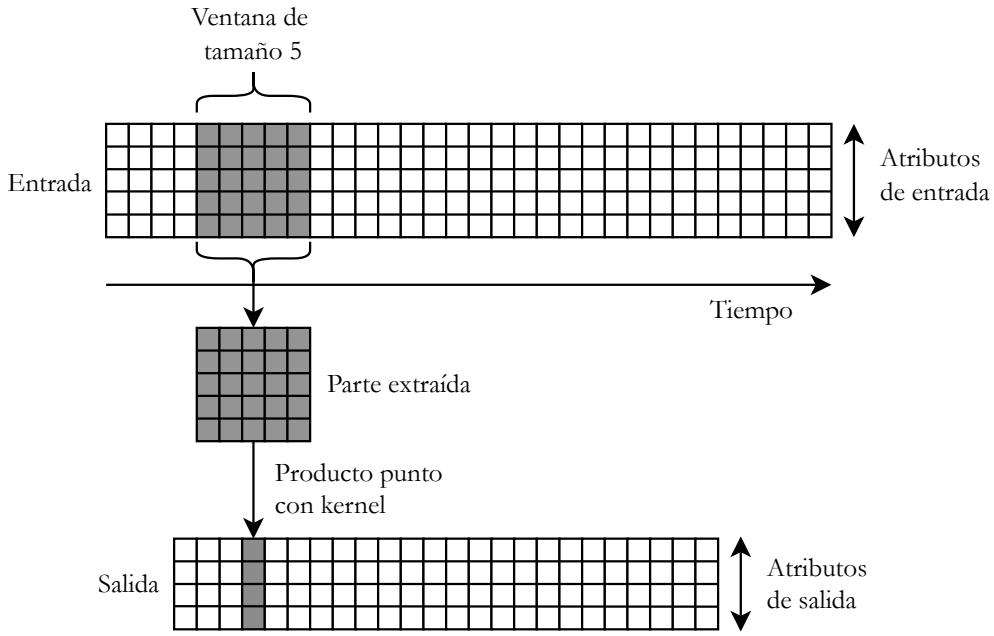


Figura 2.13: Funcionamiento de una capa de convolución 1D. Basado en (Chollet, 2017)

El detalle de las operaciones que se realizan para cada valor en la Ecuación 2.41 se presenta en la Ecuación 2.42. Donde $q_{j,m}$ es el m-ésimo elemento generado para un filtro j, que se calcula considerando todos los valores de los atributos de una cantidad V (ventana) de instancias del feature map de entrada. Cada $o_{i,m}$ es el valor del i-ésimo atributo en la m-ésima instancia.

$$q_{j,m} = f\left(\sum_{i=1}^{A}\sum_{n=1}^{V} o_{i,n+m-1} w_{i,j,n} + b_j\right) \tag{2.42}$$

Se puede notar que la definición de las Ecuaciones 2.41 y 2.42 no dependen de la cantidad de instancias a considerar en el feature map de entrada. Para comprender mejor

el funcionamiento de estas ecuaciones, específicamente la Ecuación 2.42, se considera un feature map de K instancias y una ventana $V = 3$. Definir un valor para la ventana de instancias permite desarrollar la Ecuación 2.42 que resulta en la siguiente ecuación:

$$q_{j,m} = f(\sum_{i=1}^{A} o_{i,1+m-1} w_{i,j,1} + o_{i,2+m-1} w_{i,j,2} + o_{i,3+m-1} w_{i,j,3} + b_j) \tag{2.43}$$

En la Ecuación 2.43 se puede notar que independiente del tamaño de la ventana se consideran todos los atributos i. Además, las instancias a considerar en la ventana dependerán del valor de m, un aumento o disminución de este valor desplaza la ventana. Un ejemplo para el cálculo del elemento $q_{1,1}$ ($j = 1$ y $m = 1$) del feature map de salida se presenta en la Figura 2.14, donde los elementos involucrados están coloreados de morado. A la derecha, en la parte superior de la figura está el feature map de entrada y en la parte inferior el feature map de salida. A la izquierda están agrupados todos los pesos, donde cada columna agrupa los kernels para un filtro y en rojo están marcados los elementos en un kernel (como se observa es un vector, definido como $w_{i,j}$ en Ecuación 2.41) que son multiplicados por los elementos para el atributo $i = 1$ dentro de la ventana de instancias. Ejemplos del cálculo de otros elementos del feature map de salida están disponibles en el Anexo A.

Hay que considerar que dado una cantidad K de instancias en el feature map de entrada y una ventana $V = 3$ el feature map de salida tendrá $K - 2$ columnas. Esto se produce dado que para el ancho de la ventana esta cabe $K-2$ veces, esto se puede evitar agregando márgenes y así mantener una salida con la misma cantidad de columnas.

Cada modelo DL tiene distintas propiedades y formas para procesar los datos, el modelo FNN toma un vector de entrada y realiza transformaciones sucesivas en los datos; CNN puede procesar un grupo de datos y de forma sucesiva realiza una síntesis extrayendo características relevantes; y RNN también trata con grupos de datos y tiene la capacidad de conservar un estado que permite procesar nodos de forma individual mientras mantiene información que persiste entre nodos y que puede brindar la habilidad de modelar relaciones espaciales. Para la problemática que se define en esta tesis, en la que se propone trabajar con estimaciones de concentración mineral a partir de estimaciones en los modelos LTMP y STMP es adecuado emplear los modelos DL debido a que (1) el modelo LTMP incluye variables de distinto tipo, distinta unidad de medida y distinta escala, aspectos con los que los modelos de DL pueden lidiar, (2) tienen la capacidad de modelar relaciones lineales, no lineales y en el caso de los modelos CNN y RNN consideran el orden en los datos, lo que es importante en la distribución y continuidad de las propiedades de la roca. (3) Y por último, respecto al escenario en que se estima la concentración mineral a partir de muestras del yacimiento, en la problemática que se presenta, se dispone de una mayor cantidad de datos y es que la cantidad de datos empleados para la implementación de modelos DL tiene un impacto en su desempeño.

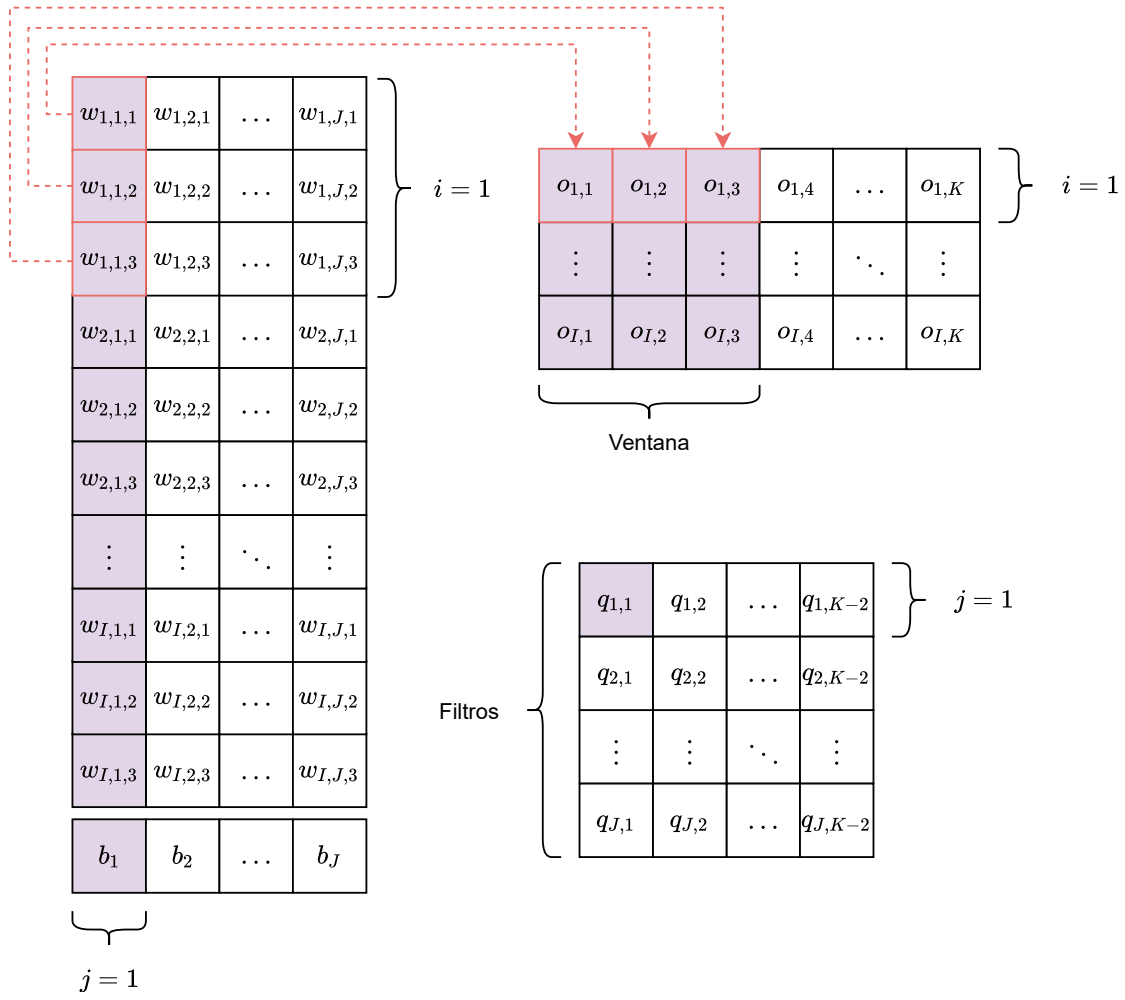


Figura 2.14: Ejemplo del cálculo de $q_{1,1}$ para una ventana $V = 3$

Capítulo 3

Estado del Arte

Es importante conocer las soluciones que han sido propuestas por parte de la academia para mejorar la estimación de concentración mineral y así poder identificar aspectos que pueden resultar útiles o en los que se puede profundizar. Una revisión del estado del arte permite conocer los avances actuales en la materia y evita repetir experimentos, propiciando el desarrollo de alternativas nuevas y más eficientes.

Esta tesis contempla la evaluación de ciertos modelos de DL para mejorar estimaciones generadas por métodos geoestadísticos. En las búsquedas iniciales no se encontraron trabajos que abordaran el mismo problema. Por ello, se amplía el alcance de la búsqueda a trabajos que apliquen métodos de ML para la estimación de concentración mineral donde las instancias en los datos contengan variables asociadas a una coordenada en un espacio tridimensional.

3.1. Proceso de revisión

La búsqueda de trabajos relacionados comienza con el desarrollo de un string de búsqueda inicial, que luego es mejorado en base a términos extraídos de documentos encontrados. Este proceso iterativo permitió generar el siguiente string de búsqueda: "*(Ore grade estimation OR Geochemical mapping OR Geochemical patterns OR Geostatistics OR Kriging OR Geochemistry OR Geologic Resource) AND (Deep learning OR Machine learning OR Neural Network OR Recurrent Neural Network OR Convolutional Neural Network)*"

La definición de las fuentes de literatura, corresponden a bases de datos electrónicas. Estas son ScienceDirect[1] , IEEExplore[2] , Springer[3] , Google Schoolar[4] y arXiv[5] . En base a los términos clave se diseñaron vectores de búsqueda para cada fuente, lo que permitió mejorar la calidad y cantidad de resultados.

[1] https://www.sciencedirect.com/
[2] https://ieeexplore.ieee.org/
[3] https://link.springer.com/
[4] https://scholar.google.com/
[5] https://arxiv.org/

Como criterio de inclusión se consideraron aquellos trabajos que se refieren a la definición y/o aplicación de modelos de ML en la estimación de concentración de mineral en un depósito. Como criterio de exclusión, se descartaron aquellos trabajos que (1) no aplican modelos de ML a la estimación de características en un depósito mineral, (2) consideran enfoques de análisis de imágenes, (3) emplean datos sísmicos y/o análisis de series de tiempo, (4) emplean datos de explosivos para tronadura y (5) no consideran estimación de concentración de mineral. Tras la aplicación de los criterios descritos se seleccionan 29 artículos, resumidos en la Tabla 3.1.

Muchos de los trabajos fueron descartados por abordar procesos de la etapa de Prospección y Exploración previas a la estimación mineral, trabajos que buscaban determinar el potencial de áreas en un mapa de escala regional para contener depósitos minerales con propósitos de explotación.

3.2. Resultados

3.2.1. Estimación mediante Redes Neuronales

De los trabajos seleccionados (Wu and Zhou, 1993) es el más antiguo en donde se propone un método de Inteligencia Artificial, específicamente una Red Neuronal, como solución alternativa para la estimación de recursos minerales frente a los métodos empleados en ese tiempo (razonamiento geométrico y técnicas geoestadísticas). También, se menciona que técnicas de geoestadística como OK son limitadas debido a suposiciones respecto a los datos. Se asume, por ejemplo, que existe una relación en la concentración de mineral entre puntos, determinada por su proximidad. Esto requiere que para el cálculo de la concentración en un punto se determine el radio dentro del que se consideran puntos cercanos, los que tendrán mayor o menor ponderación en función de su ubicación. Otras suposiciones están relacionadas con los efectos que tienen distintos ambientes geológicos en la distribución de minerales. Es interesante notar, que documentos más recientes acuden al mismo argumento (Goswami et al., 2017), (Zhang et al., 2013), (Samanta et al., 2005). (Wu and Zhou, 1993) en su trabajo concluye que las estimaciones de la Red Neuronal generan resultados razonablemente precisos comparados con métodos convencionales en aquel momento.

(Samanta et al., 2002) es el segundo documento más antiguo, en que se emplea una red neuronal, este trabajo comienza a incorporar otros aspectos que son relevantes en un esquema de implementación de modelos de ML. Para lidiar con datos sesgados en la separación de datos en entrenamiento y prueba, realiza una división arbitraria de los datos en grupos en base a rangos de concentración mineral, para posteriormente emplear un algoritmo genético para realizar la división de los datos en subconjuntos de entrenamiento, prueba y validación. En esta misma línea los autores desarrollan otro trabajo (Samanta et al., 2004), en que implementan el método de Mapas Autoorganizados (Kohonen Network[6]) para generar los conjuntos de entrenamiento, prueba y validación. Una vez realizada la división en

[6]Kohonen Network es un tipo de Red Neuronal que es entrenada usando aprendizaje no supervisado para producir una representación discreta del espacio de las muestras de entrada, denominado mapa.

Tabla 3.1: Resumen de trabajos relacionados

ref.	Mineral	Método/modelo	Métrica
(Wu and Zhou, 1993)	Cobre	FNN, OK	MSE, MAE
(Samanta et al., 2002)	Oro	FNN, OK, LK, SK, GA	ME, MAE, MSE, R^2, NMSE
(Samanta et al., 2004)	Oro	FNN, OK, LK, SK	ME, MAE, MSE, R^2
(Samanta et al., 2005)	Oro	FNN, Adaboost	bias, MAE, MSE, R^2
(Samanta et al., 2006)	Oro, Bauxita, Hierro	FNN	MSE
(Dutta et al., 2006)	Sílice, Alúmina	FNN, OK, OK+FNN, Ensamble, GA	MSE and R^2
(Chatterjee et al., 2007)	Óxido de Calcio, Alúmina, Óxido de hierro, Sílice	GRNN, OK, GRNN+RK, GA	ME, MAE, MSE,R^2
(Mahmoudabadi et al., 2009)	Hierro	FNN, OK, GA	MSE
(Tutmez, 2009)	Lignito (carbón)	FNN, OK, RBNN, Fuzzy c-means clustering	Std, Variance account for (VAF) and RMSE
(Dutta et al., 2010)	Oro	FNN, OK, SVM	ME, MAE, RMSE y R^2
(Li et al., 2010)	Cobre	Wavelet NN	MSE
(Chatterjee et al., 2010)	Plomo y Zinc	FNN, OK, SVM, Ensamble, GA	ME, MAE, MSE, R^2
(Samanta, 2010)	Oro	FNN, OK, RBNN	MAE, MSE and R^2
(Badel et al., 2011)	Hierro	FNN, MIK	MSE, R
(Gholamnejad et al., 2012)	Hierro	FNN	MSE
(Tahmasebi and Hezarkhani, 2012)	Cobre	FNN, GA, Lógica difusa	MSE
(Zhang et al., 2013)	Plata y Cobre	FNN, OK, SVM, KNN, PSO,	ME, MAE, RMSE,R^2
(Li et al., 2013)	Cobre	FNN, SVM, PSO	RMSE
(Jalloh et al., 2016)	Rutilo	FNN, OK	MSE, R^2, Variance
(Goswami et al., 2016)	Cobre	OK, SVM, GRNN	ME, MAE, RMSE y R^2
(Goswami et al., 2017)	Hierro	FNN, OK, GRNN	SE, MAE, MSE, RMSE, R and R^2
(Caté et al., 2017)	Oro	KNN, naïve Bayes, SVM, DT, RF	Precision, Recall, f1-score
(Hari et al., 2018)	Níquel y Cobalto	FNN, OK	R
(Jafrasteh and Fathianpour, 2017)	Hierro	RBNN	MSE
(Singh et al., 2018)	Hierro	OK, RNN	R^2, MSE
(Jafrasteh et al., 2018)	Cobre	FNN, OK, RF, GP, IK	MSE, NMSE
(Schnitzler et al., 2019)	Sodio	RF	R^2
(Iglesias et al., 2020)	Wolframio y Estaño	FNN, SVM, DT	R^2, Precision, Recall, Confusion Matrix
(Afeni et al., 2020)	Hierro	FNN, OK	MAE, RMSE, MSE, R and R^2

Feedforward Neural Network (FNN), Ordinary Kriging (OK), Median Indicator Kriging (MIK), Indicator Kriging (IK), Lognormal Kriging (LK), Simple Kriging (SK), Residual Kriging (RK), Neural Network (NN), General Regression Neural Network (GRNN), Radial Basis Neural Network (RBNN), Support Vector Machine (SVM), Mean Absolute Error (MAE), Mean Squared Error (MSE), Coefficient of determination (R^2), Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Gaussian Process Regression (GP), K-Nearest Neighbors (KNN)

base a concentración se entrena una Red Neuronal y los métodos OK, Lognormal Kriging (LK) y SK. En (Samanta et al., 2002) y (Samanta et al., 2004) las Redes Neuronales obtiene mejores resultados por una leve diferencia. En (Samanta et al., 2005) realizan un estudio comparativo entre el desempeño de la combinación de varias Redes Neuronales mediante el algoritmo Adaboost en comparación con el desempeño de sólo una Red Neuronal, en la estimación de concentración de oro. Los resultados indican que el conjunto de Redes Neuronales no es superior, y ambos casos obtienen un desempeño bajo. Este análisis se realiza con datos del Ártico y muestras a grandes intervalos debido a las condiciones del terreno y factores económicos. En (Samanta et al., 2006) concentran los esfuerzos en la comparación de algoritmos de entrenamiento y optimización de hiperparámetros para una Red Neuronal.

En (Dutta et al., 2006) se propone por primera vez una combinación entre métodos geoestadísticos y ML, mediante la combinación de las estimaciones o salidas. La combinación de OK y FNN resulta superior solo en la estimación de uno de los minerales considerados en la investigación. Por otro lado, (Mahmoudabadi et al., 2009) investiga la optimización de una FNN mediante un algoritmo genético. De acuerdo con sus resultados concluyen que su Red Neuronal genera mejores estimaciones que el método OK. En (Badel et al., 2011) se establece una comparación entre los métodos Median Indicator Kriging (MIK) (Variación del método Indicator Kriging (IK)) y Redes Neuronales, siendo el primer método el que obtiene mejores resultados. En (Gholamnejad et al., 2012) solo se implementa una Red Neuronal sin contrastar los resultados con otro métodos, pero en base a las estimaciones concluyen que resulta una alternativa válida, ya que, requiere menor conocimiento y tiempo para llevar a cabo su implementación. En (Tahmasebi and Hezarkhani, 2012) los investigadores implementan una Red Neuronal en combinación con lógica difusa, evaluando estos métodos por separado para luego combinarlos considerando algoritmos genéticos para optimización de hiperparámetros. La combinación de lógica difusa, Red Neuronal y un algoritmo genético genera los mejores resultados en estimación de concentración de cobre. En (Jalloh et al., 2016) los investigadores se concentran en la integración de modelos de ML con métodos geológicos para la generación de modelos de bloques, donde obtienen mejores resultados tras emplear una Red Neuronal, por sobre los resultados del método OK en la estimación de Rutilo (óxido de titanio). (Hari et al., 2018) desarrolla una etapa de clasificación para identificar la presencia o ausencia de pequeños cúmulos de mineral en el fondo del océano y posteriormente realizar un proceso de regresión para estimar la concentración del mineral. En este trabajo se utiliza un conjunto de datos reducido para el cual se obtiene un buen desempeño para una FNN por sobre el método OK. (Afeni et al., 2020) implementa FNN y OK, donde el último método es el que genera mejores resultados para la estimación de Hierro.

3.2.2. Estimación mediante otros modelos de Machine Learning

Algunos de los trabajos seleccionados consideran otros modelos ML además de las Redes Neuronales. En (Dutta et al., 2010) los investigadores emplean una metodología muy similar a (Samanta et al., 2002) y (Samanta et al., 2004) en la división de los datos para

entrenamiento y prueba. Comparan el desempeño de FNN, SVM y OK para estimar concentración de oro. SVM obtuvo un mejor desempeño, sin embargo, éste fue marginal. Los autores del trabajo presentado en (Chatterjee et al., 2010) integran un algoritmo genético para optimizar los pesos en ensamble de modelos para realizar una suma ponderada de sus salidas. En los resultados, se compara el ensamble de salida ponderada, sin ponderar, la mejor Red Neuronal y el método OK. El ensamble de métodos mediante la suma ponderada de sus salidas se desempeña mejor sólo para uno de los minerales en estudio mientras que para el otro no presenta una mejora significativa. (Zhang et al., 2013) analiza los métodos Weighted Least Squares Support Vector Machine (LS-SVM), OK y una Red Neuronal. Adicionalmente, emplean Weighted K-Nearest Neighbors (WKNN), un algoritmo de clasificación para la interpolación de valores ausentes, Particle Swarm Optimization (PSO) para la división del conjunto de datos en entrenamiento y prueba, en base a los resultados concluyen que LS-SVM alcanza un rendimiento superior. En (Li et al., 2013) los investigadores se enfocan en evaluar el algoritmo PSO, considerando algunas de sus variaciones para ajustar parámetros del modelo SVM y los resultados son contrastados con una Red neuronal. El mejor puntaje es obtenido por la variación Self-Learning de PSO en estimación de cobre. En (Caté et al., 2017) los investigadores evalúan los métodos: K-Nearest Neighbors (KNN), el cual usa puntos vecinos etiquetados en un espacio Euclidiano formado por características de entrada para predecir clases; el método Naïve Bayes, el cual se basa en el teorema de Bayes para evaluar la probabilidad de que un evento (clase) ocurra dado el valor de datos de entrada; SVM, DT y Random Forest (RF). De acuerdo con los resultados, RF obtuvo mejores resultados en la clasificación de presencia de mineral de oro. En (Jafrasteh et al., 2018) evalúan los modelos FNN, RF y Gaussian Process Regression (GP) en contraste con métodos geoestadísticos OK e IK. Los resultados demuestran que GP con una estandarización simétrica de las entradas de ubicación espacial producen las predicciones más precisas. En (Schnitzler et al., 2019) se concentran en el uso de RF en la estimación de sodio, evaluando el desempeño en función de cambios en la cantidad de instancias y variables empleadas, demostrando que es un método que alcanza un buen desempeño mediante el análisis de las estimaciones, pero sin compararlas con otros métodos. (Iglesias et al., 2020) buscan el método adecuado para estimación de wolframio y estaño, para ello evalúan el uso de DT, FNN, SVM. Según sus resultados, DT obtiene un mejor desempeño.

3.2.3. Estimación mediante variantes de Redes Neuronales

En (Chatterjee et al., 2007) se evalúa el desempeño del modelo General Regression Neural Network (GRNN), un tipo de red neuronal que emplea como función de activación una función de base radial o Radial Basis Function (RBF). En dicho trabajo concluyen que una combinación de la salida de GRNN combinado con Residual Kriging (RK) (OK aplicado en residuales) supera significativamente el desempeño de OK y levemente el desempeño del modelo GRNN. (Tutmez, 2009) emplea un método de clusterización difuso, en base a los clústeres generados se extraen datos para generar conjuntos de entrenamiento y definir el número de reglas difusas que son combinadas con un modelo Radial Basis Neural Network

(RBNN). (Li et al., 2010) implementa el modelo Wavelet Neural Network, denominado así debido al uso de la función de activación wavelet. Este trabajo es el primero que propone de forma explícita la incorporación de información de puntos vecinos como parte del vector de entrada, donde se reportan buenos resultados, sin embargo, no se contrastan resultados con otros métodos para el mismo conjunto de datos. En (Samanta, 2010) contrastan los métodos OK, FNN, RBNN para la estimación de Oro, donde concluyen que este último método alcanza un mejor desempeño. En (Goswami et al., 2016) los investigadores comparan el método de OK, SVM y GRNN, donde es este último método el que obtiene el mejor resultado en la estimación de concentración de Hierro. El mismo equipo (Goswami et al., 2016) continuó trabajando con el mismo conjunto de datos para luego publicar otro trabajo (Goswami et al., 2017), donde se compara el desempeño de los modelos GRNN, FNN y OK, siendo el primero el que mejor se desempeña. (Jafrasteh and Fathianpour, 2017) evalúan redes neuronales con distintas funciones de activación, siendo estas Radial Basis, Local Linear Radial Basis, Local Linear Radial Basis con Gaussian, Local Linear Radial Basis con skewed Gaussian. Del trabajo concluyen que Local Linear Radial Basis Neural Network con skewed Gaussian tiene un mejor desempeño. De los trabajos seleccionados (Singh et al., 2018) es el único que emplea RNN para la estimación de concentración mineral, sin embargo carece de información explícita respecto del vector de entrada y los criterios empleados para generarlos. Este trabajo contrasta los resultados con el método OK con el que mantiene resultados muy similares respecto a las métricas, pero en términos visuales, se aprecia una mayor continuidad de la concentración en las estimaciones del modelo RNN.

3.3. Síntesis

El mineral para el que más trabajos se han desarrollado es el Hierro, seguido por Cobre y Oro, luego Aluminio, Sílice y finalmente otros minerales para los que solo existe un trabajo, como Wolframio, Estaño, Sodio, Cobalto, Níquel, Rutilo, Plata, Zinc, Plomo y Calcio. Hay que mencionar que no existen trabajos en los que se realicen estimaciones de Molibdeno y/o Arsénico. Mientras que las métricas de desempeño más empleadas son MSE, MAE, Coefficient of determination (R^2) y Mean Error (ME), en ese orden.

Ninguno de los trabajos incluidos propone una metodología para la correcta implementación de modelos de ML. La falta de una metodología unificadora dificulta la comparación de trabajos y sus resultados. Algunos de los trabajos no mencionan haber utilizado algún método de preprocesamiento de datos, de los trabajos que mencionan algún tipo de preprocesamiento la mayoría emplea algún tipo de normalización, siendo el rango [-1,1] el más frecuente y se destaca el uso de normalización simétrica de coordenadas con el propósito de mantener las proporciones (Jafrasteh et al., 2018). Otro de los métodos de preprocesamiento señalados son el Análisis de Componentes Principales (PCA) para reducción de dimensionalidad (Hari et al., 2018), (Iglesias et al., 2020); la exclusión de instancias con valores ausentes (Zhang et al., 2013); y el tratamiento de variables categóricas mediante la generación de variables indicadoras (Chatterjee et al., 2010).

A pesar de que algunos de los trabajos realizan optimización de hiperparámetros me-

diante algún método/algoritmo (Chatterjee et al., 2010), (Samanta, 2010), (Zhang et al., 2013), (Tahmasebi and Hezarkhani, 2012), la mayoría identifica hiperparámetros mediante pruebas reiteradas, de forma arbitraria o empleando conocimiento previo. Respecto a la metodología de validación, Holdout es el método más empleado, el segundo método más utilizado es K-fold Cross-validation (CV) (Zhang et al., 2013), (Caté et al., 2017), (Jafrasteh et al., 2018), (Iglesias et al., 2020). Hay que mencionar que algunos trabajos consideran la generación de grupos/clústeres de datos de los cuales se extraen datos para luego implementar algún método de validación.

Se puede notar que existen trabajos aplicados sobre distintos minerales y que consideran diversos métodos de ML. Algunos de los trabajos realizan implementaciones más sencillas de los modelos, mientras que otros exploran implementaciones más complejas, considerando métodos de optimización de hiperparámetros (Tahmasebi and Hezarkhani, 2012), (Li et al., 2013) y de ensamble de modelos (Samanta et al., 2005), (Caté et al., 2017), (Schnitzler et al., 2019). Algunos trabajos se enfocan en explorar con más profundidad solo un modelo (Li et al., 2013), (Schnitzler et al., 2019), mientras que otros buscan identificar el modelo adecuado (Dutta et al., 2010), (Goswami et al., 2016), (Iglesias et al., 2020), y no todos los trabajos contrastan los resultados con métodos tradicionales como Kriging.

De los trabajos revisados, 18 contrastan sus resultados con algún método geoestadístico. En los trabajos (Badel et al., 2011) y (Afeni et al., 2020), los métodos MIK y OK obtienen mejores resultados que un método de ML. Ambos trabajos emplean el modelo FNN y declaran que las arquitecturas e hiperparámetros fueron seleccionados mediante pruebas reiteradas. En los trabajos (Samanta et al., 2004), (Dutta et al., 2006), (Jalloh et al., 2016), (Hari et al., 2018) y (Singh et al., 2018) se obtienen resultados similares entre OK y FNN.

(Dutta et al., 2010), (Chatterjee et al., 2010), (Zhang et al., 2013), (Goswami et al., 2016) implementan SVM en contraste con una Red Neuronal y OK. En todos, los métodos de ML obtienen resultados que superan los resultados del método OK en su implementación, pero solo en (Zhang et al., 2013) una variación del método SVM obtiene mejores resultados, por sobre una red neuronal. (Chatterjee et al., 2007), (Samanta, 2010), (Goswami et al., 2017) emplean algún tipo de RBNN, superando el resultado de otros métodos geoestadísticos y de ML. (Samanta et al., 2002), (Dutta et al., 2010) y (Goswami et al., 2016) emplean una Red Neuronal con función de activación Gaussiana, superando el resultado OK. En (Jafrasteh et al., 2018) mediante el método GP supera métodos geoestadísticos y de ML. Hay que mencionar que (Mahmoudabadi et al., 2009), (Tutmez, 2009) y (Chatterjee et al., 2010) logran superar métodos geoestadísticos empleando un algoritmo genético para encontrar los valores óptimos para la inicialización de un FNN, empleando lógica difusa en combinación de Redes Neuronales y ensamble de Redes Neuronales, respectivamente.

De los trabajos cuyos resultados no se comparan con algún método geoestadístico (Wu and Zhou, 1993) y (Gholamnejad et al., 2012) implementan FNN, (Li et al., 2010) implementa Redes Neuronales con función de activación wavelet, (Tahmasebi and Hezarkhani, 2012) implementa una combinación de Redes Neuronales con lógica difusa, (Samanta et al., 2005) implementa Redes Neuronales con función gaussiana, (Jafrasteh and Fathianpour, 2017) implementa RBNN, (Iglesias et al., 2020) emplea DT, (Schnitzler et al., 2019) utiliza RF y (Li et al., 2013) variaciones de SVM . Todos estos trabajos concluyen que se

alcanza un desempeño razonablemente preciso mediante el análisis de propiedades de las estimaciones.

Sólo tres de los trabajos incluidos en la revisión no incorporan alguna forma de Red Neuronal. Sólo un trabajo (Singh et al., 2018), emplea algún modelo considerado como DL (Recurrent Neural Network). El resto de las implementaciones de redes neuronales consisten en arquitecturas de una capa y un número reducido de neuronas, lo que se justifica con el tamaño reducido de los conjuntos de datos. Sin embargo, en general los trabajos no comparten el esquema de pruebas que realizaron para diseñar las arquitecturas, lo que cobra relevancia al momento de considerar las conclusiones de algunos de los trabajos, que afirman que su método supera a una red neuronal. Por último, hay que mencionar que ninguno de los trabajos implementa alguna forma de CNN.

Los trabajos que se han desarrollado a la fecha muestran que los modelos de ML son una herramienta poderosa, capaces de alcanzar mejores resultados en la estimación de concentración mineral que métodos geoestadísticos. Una desventaja que tiene esta área resulta de la naturaleza de los conjuntos de datos, los que suelen ser privados. Como consecuencia diferentes investigadores no pueden probar los métodos que proponen sobre un mismo conjunto de datos y así reportar alguna mejora. Por otro lado, si bien se ha probado un gran espectro de modelos de ML, estos no son suficientes como para observar una tendencia. Esto se vuelve más complejo si consideramos que en cada trabajo se evalúan distintos modelos para distintos minerales, siendo incluso que algunos no incluyen resultados de métodos geoestadísticos, que son los que predominan en la industria.

Capítulo 4

Implementación

Para llevar a cabo el entrenamiento y la evaluación de modelos de ML/DL es necesario realizar un análisis y preparación de los conjuntos de datos disponibles. En este capítulo se describen el preprocesamiento realizado, los conjuntos de datos generados, la metodología para la inclusión de nodos vecinos y las herramientas empleadas para la implementación de los modelos de Deep Learning.

4.1. Conjunto de datos

El punto de partida para la experimentación es la exploración y tratamiento de un conjunto de datos de 49,295,441 de instancias al que se hace referencia como conjunto de datos original. Este conjunto de datos corresponde a un modelo de bloques, donde cada nodo tiene 15 metros de alto, 20 metros de ancho y 20 metros de largo, lo que equivale a un volumen de $6{,}000m^3$. Cada nodo en el conjunto de datos original tiene 25 variables, algunas del Long-Term Mine Planning Model (LTMP) y otras del Short-Term Mine Planning Model (STMP). Todas las variables están descritas en la Tabla 4.1.

En el conjunto de datos cada nodo se diferencia por su ubicación en el espacio, lo que se registra en variables que definen coordenadas de profundidad, ancho y altura (X, Y y Z). La mayoría de las variables disponibles corresponden al LTMP, algunas estimadas mediante métodos geoestadísticos y otras generadas para organizar la extracción de roca. Solo las variables que brindan información sobre la concentración de los minerales Cobre, Arsénico y Molibdeno (cut_lp, as_lp y mot_lp), tienen su contraparte en el STMP (cut_pz, as_pz y mot_pz), y que además, son las únicas variables del STMP.

Al graficar cada nodo como un punto en un espacio tridimensional se genera la Figura 4.1, que corresponde a todo el volumen en estudio, un paralelepípedo de 1,710 metros de alto, 7,330 metros de largo y 5,830 metros de profundidad. Este volumen es una referencia y tiene dimensiones que superan el volumen de roca de interés, estando incluso por sobre la superficie del terreno para el que se busca generar estimaciones.

Tabla 4.1: Variables en conjunto de datos original

Variable	tipo	Descripción
X	numérica	coordenadas de la ubicación del nodo en el eje x
Y	numérica	coordenadas de la ubicación del nodo en el eje y
Z	numérica	coordenadas de la ubicación del nodo en el eje z
alter_lp	categórica	atributo de la roca. Tipo de alteración.
categ18	categórica	contiene las categorías que indican el nivel de confianza con el que la estimación debería coincidir con el valor real. 1, 2 y 3 corresponde a Medido, Indicado e Inferido, respectivamente. Confianza en orden descendente.
lito_lp	categórica	atributo de la roca. tipo de litología
mnzn_lp	categórica	atributo de la roca. tipo de zona mineral, tipo de mineral donde está el cobre, tipo de sulfuro y tipo de óxido
Modelo	categórica	separa ambientes geológicos. Pórfido versus Vetas. Con la misma información es más sencillo modelar ambientes pórfidos. Definen comportamientos del modelo
Periodo	categórica	variable que separa en volumen la temporalidad de la explotación del yacimiento
Zona_Explotada	categórica	variable que separa la zona explotada o rajo
as_lp	numérica	valores estimados del arsénico en el largo plazo
as_pz	numérica	valores estimados del arsénico en el corto plazo
cut_lp	numérica	valores estimados del cobre total en el largo plazo
cut_pz	numérica	valores estimados del cobre total en el corto plazo
mot_lp	numérica	variable que contiene los valores estimados del molibdeno en el largo plazo
mot_pz	numérica	variable que contiene los valores estimados del molibdeno en el corto plazo
ugcut_lp	categórica	variable que separa en unidades de volumen las características similares del comportamiento de la variable cut_lp
densidad	numérica	variable densidad de la roca
SN	numérica	número del bloque
Aux_Pit_Quinquenio	categórica	variable que contiene el volumen de explotación denominado quinquenio considerados a extraer a 5 años.
Aux_Pit_Recursos	categórica	variable que contiene el volumen de explotación, considerado como económico, denominado Recursos (todo el volumen que contiene mineral)
Aux_Pit_Reservas	categórica	variable que contiene el volumen de explotación, considerados para extraer durante la vida útil de la mina, denominado Reservas (volumen que contiene mineral económicamente explotable)
MG_Bo_2018	categórica	variable que contiene tipo de sulfuro denominada Bornita (3 volúmenes excluyentes)
MG_Cpy_2018	categórica	variable que contiene tipo de sulfuro denominada Calcopirita (3 volúmenes excluyentes)
Distance_cut	numérica	variable que contiene la isodistancia, a las muestras más cercana, vecino más cercano, desde las muestras. En general a mayor distancia de las muestras mayor incertidumbre en las variables as_lp, cut_lp y mot_lp.

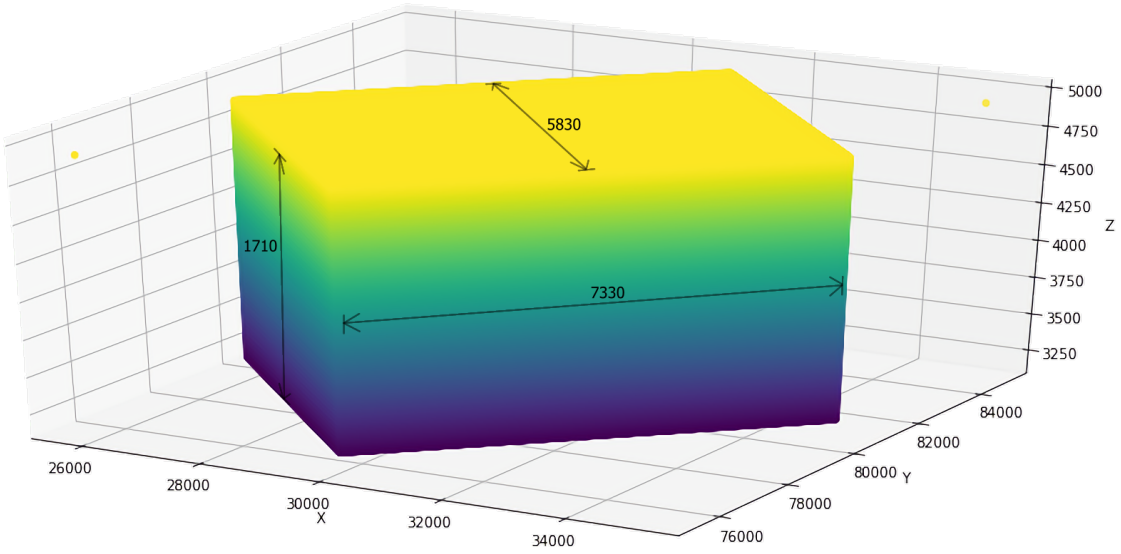


Figura 4.1: Cubo de datos original. La unidad de todas las cifras está en metros.

4.2. Preprocesamiento de datos

El primer objetivo en el preprocesamiento de datos es purgar el conjunto de datos original y preservar información que permita contrastar los modelos LTMP y STMP, lo que se logra a través de las variables de concentración mineral. Un análisis inicial de valores ausentes permite identificar que un 15.7 % de instancias presentaban valores ausentes en las variables de concentración mineral del LTMP, lo que indica que para un porcentaje de nodos no se realizaron estimaciones. Tras graficar solo los nodos para los que se realizaron estimaciones resulta la Figura 4.2, donde la parte superior puede interpretarse como la superficie del terreno y los nodos que no disponen de valores se pueden interpretar como "aire".

El análisis de valores ausentes también permitió identificar que gran parte (98.4 % aproximadamente) de los nodos no disponían de información en variables del STMP. Esto es debido a que solo una pequeña parte del volumen en estudio ha sido explotada. De acuerdo con el objetivo de esta tesis y para que sea posible la implementación de un enfoque de aprendizaje supervisado, se excluyen todas las instancias o nodos con valores ausentes. Por lo que resulta un conjunto de datos de 732,870 instancias donde cada nodo dispone de información para ambos modelos, STMP y LTMP. Al graficar estas instancias resulta la Figura 4.3, cuya forma de cono coincide con la forma del orificio generado en una mina de cielo abierto.

Se decide excluir las variables que brindan información sobre la planificación de la fase de explotación del yacimiento y además tienen un porcentaje de instancias con valores ausentes muy alto (Aux_Pit_Quinquenio, Aux_Pit_Recursos y Aux_Pit_Reservas). También se excluye una variable empleada para registrar un identificador para cada nodo

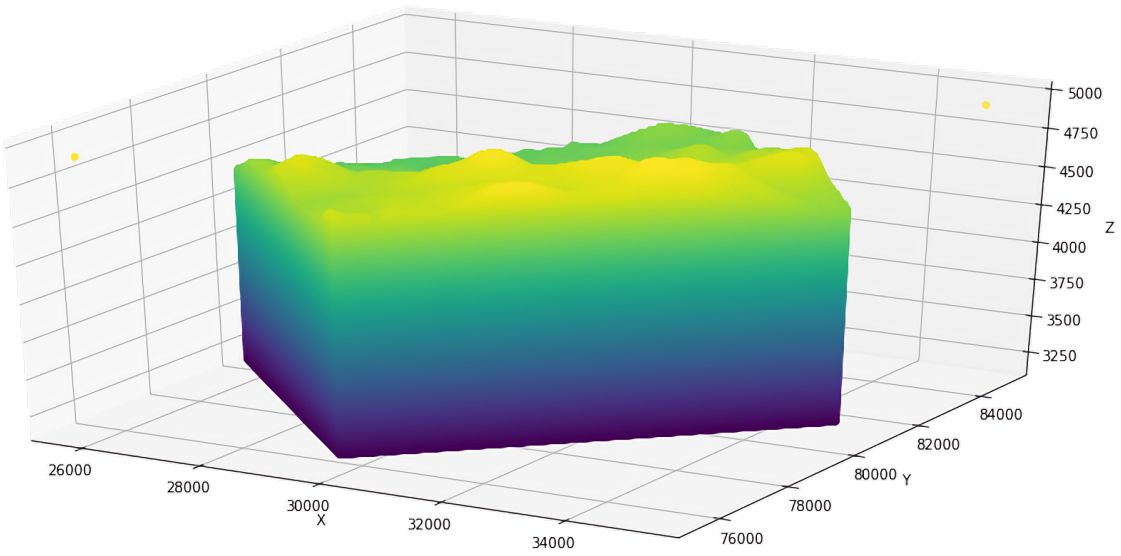


Figura 4.2: Conjunto de datos original sin valores ausentes en las variables del LTMP. La unidad de todas las cifras está en metros.

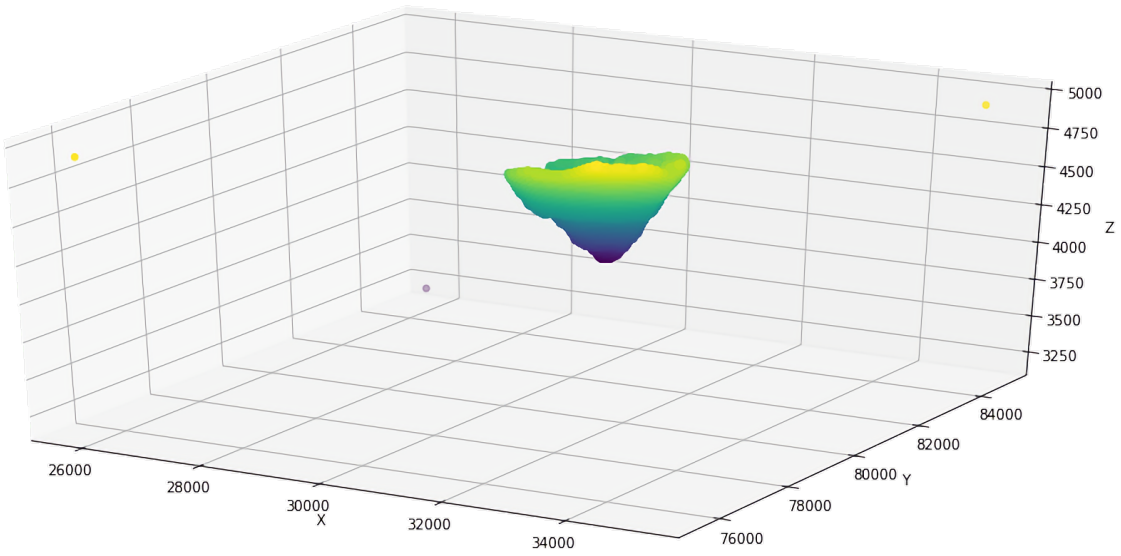


Figura 4.3: Sin valores ausentes en LTMP y STMP

(SN), una variable empleada para registrar el periodo de explotación, cuyos valores estaban corrompidos e inconsistentes (Periodo) y una variable que permite diferenciar instancias que han sido explotadas de las que no (Zona_Explotada).

Las variable de concentración de Cobre (cut_lp y cut_pz) registran el porcentaje de Cobre en un nodo, mientras que las variables de concentración de Arsénico y Molibdeno (as_lp, as_pz, mot_lp y mot_pz) registran las partículas por millón (ppm) en cada nodo. Para la implementación de los experimentos se decide hacer un enfoque en la concentración de cobre, lo que tiene como consecuencia directa la exclusión de las variables de concentración de arsénico (as_pz) y molibdeno (mot_pz) del STMP. Por lo tanto, la variable de concentración de cobre (cut_pz) es considerada como la etiqueta y el objetivo a estimar para los modelos.

Para el preprocesamiento de las variables categóricas se hace una distinción entre variables ordinales y nominales de la siguiente forma. Las variables que contienen información de los tipos de sulfuro Bornita y Calcopirita (MG_Bo_2018 y MG_Cpy_2018) son tratadas como variables cualitativas ordinales y las clases son reemplazadas por números arbitrarios del 1 al 4 manteniendo el orden informado por expertos del área minera. Por otro lado, las variables del tipo de alteración, tipo de litología, tipo de zona mineral y tipos de concentración de cobre (alter_lp, lito_lp, mnzn_lp, ugcut_lp, respectivamente) son tratadas como variables cualitativas nominales por lo que por cada clase en una de las variables se genera una variable indicadora (un 1 si para la instancia pertenece a la variable clase y un 0 de lo contrario). Esto aumenta la cantidad de variables por nodo a 43, sin contar la concentración de cobre del STMP. El aumento de variables es debido a que la variable tipo de alteración (alter_lp) tiene 7 clases, la variable de tipo de litología (lito_lp) tiene 9 clases, la variable de tipo de zona mineral (mnzn_lp) tiene 6 clases y tipos de concentración de cobre (ugcut_lp) 9 clases.

Todas las variables numéricas son normalizadas por separado, empleando el promedio y la desviación estándar de acuerdo con la Ecuación 4.1, donde la variable normalizada z es el resultado de la resta del promedio μ a cada valor en x, todo dividido por su desviación estándar σ. Sin embargo, normalizar las variables de ubicación espacial (variables X, Y y Z) por separado puede generar distorsión en las dimensiones del depósito, especialmente cuando la extensión espacial del depósito a lo largo de cada dimensión es diferente. Para preservar la distancias relativas, la misma escala es empleada para las tres dimensiones espaciales, por esto se normalizan estas variables en conjunto, de manera simétrica (Jafrasteh et al., 2018).

$$z = \frac{x - \mu}{\sigma} \tag{4.1}$$

Como resumen, se llevaron a cabo las siguientes tareas de preprocesamiento de los datos:

- Selección/exclusión de variables
- Exclusión de variables ausentes
- Tratamiento de variables cualitativas ordinales y nominales

- Normalización de variables numéricas, normalización simétrica
- Cambio de orden (orden aleatorio de instancias)

El conjunto de datos que resulta del preprocesamiento contiene 732,870 instancias que disponen de datos sobre la concentración de mineral para los modelos STMP y LTMP. Es un conjunto de datos que puede ser empleado para entrenar un modelo de ML en un enfoque supervisado, ya que se disponen de vectores de entrada del LTMP y de etiquetas del STMP. La concentración de cobre de STMP, que es la variable considerada como etiqueta en los experimentos (cut_pz), contiene valores entre 0 (y_{min}) y 35.25 (y_{max}), con un promedio de 0.43 ($\bar{y}$). El histograma de los valores de concentración de cobre, junto con un resumen de estadísticos, se muestra en la Figura 4.4. La distribución de los valores de concentración de cobre tiene un sesgo (skewness) positivo alto. El alto valor del coeficiente de variación ($\sigma/\bar{y} = 1.51$) indica la presencia de valores extremos en los datos. En la mayoría del volumen en estudio, el contenido de cobre es bastante bajo. Algunos lugares son ricos en cobre. La alta relación entre el máximo y la media ($y_{max}/\bar{y} = 82.02$) significa que para obtener predicciones confiables, es necesario modelar regiones de alta y baja concentración con alta precisión.

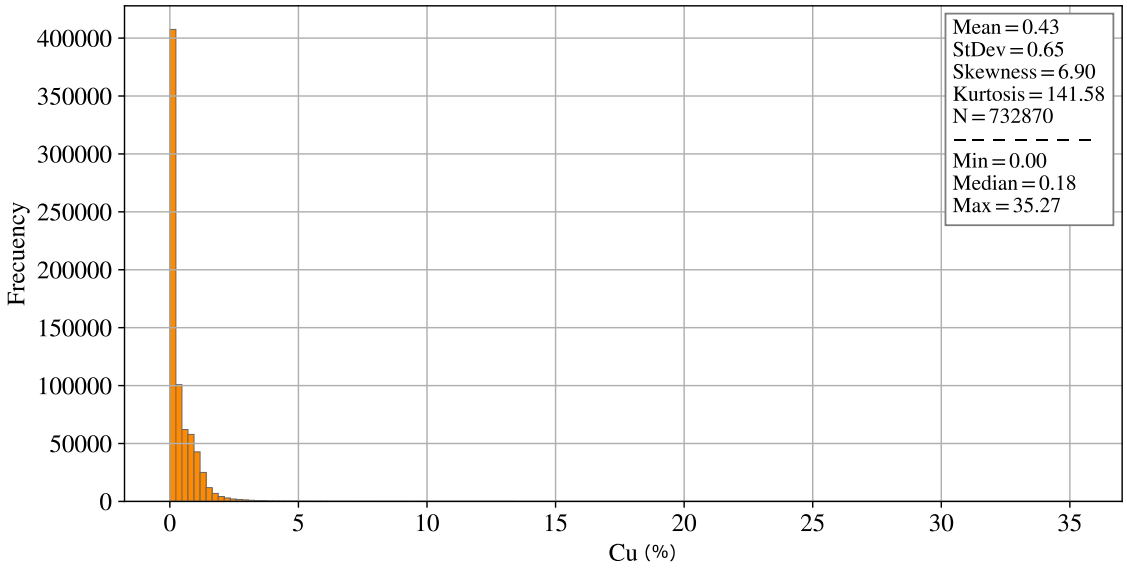


Figura 4.4: Histograma para la variable de concentración de cobre

La Figura 4.5 muestra estadísticos sólo considerando las concentraciones menores a 3, donde se mantiene un sesgo en la distribución y permite comprender con más claridad que en general el contenido de cobre es bastante bajo y el modelo debe ser capaz de distinguir aquellas zonas en particular en las que se encuentra una alta concentración mineral.

Otro aspecto relevante en el conjunto de datos es el tipo de mineral de cobre. En los datos se dispone de una variable que agrupa a nodos en zonas minerales (mnzn_lp),

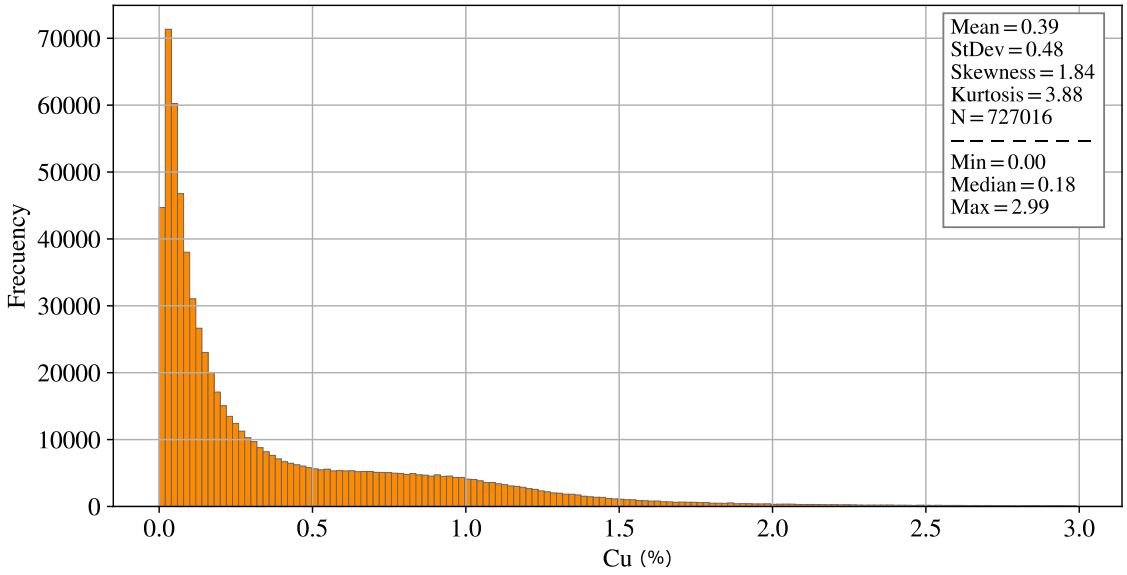


Figura 4.5: Histograma solo de instancias con concentración de cobre < 3.0

información que se complementa con las variables que permiten identificar la presencia de sulfuros denominados Bornita (MG_Bo_2018) y Calcopirita (MG_Cpy_2018). El análisis de estas variables indica que en solo una de las seis zonas minerales predomina mineral de Cobre tipo Sulfuro (Bornita y Calcopirita), lo que corresponde a un 19 % aprox. de las instancias en el conjunto de datos que resulta del preprocesamiento. El detalle de la cantidad de instancias por zona mineral en los datos está disponible en la Tabla 4.2, donde se aprecia que del total de instancias de mineral de Cobre tipo Sulfuro se concentran en una zona mineral.

Tabla 4.2: Número de instancias que contienen mineral de Cobre tipo Sulfuro u Óxido

Zona	instancias	Óxido	Sulfuro
A	6,898	6,898	0
B	216,361	21,6359	2
C	10,848	10,847	1
D	68,850	66,981	1,869
E	143,530	2,698	140,832
F	286,383	285,926	457
Total	732,870	589,709	143,161

4.2.1. Nodos colindantes

En el trabajo presentado por (Li et al., 2010), para la estimación de concentración mineral de un nodo, se considera información de nodos cercanos o vecinos. En base a dicha

propuesta, en esta tesis se emplea la fórmula de distancia euclidiana (Ecuación 4.2), que permite calcular la distancia entre dos puntos $\boldsymbol{p} = (p_1, \ldots, p_n)$ y $\boldsymbol{q} = (q_1, \ldots, q_n)$, en un espacio de n dimensiones, para determinar la vecindad de nodos cercanos. Esto permite identificar, para cada uno de los nodos, los nodos que están dentro de un radio determinado y, por lo tanto, son parte de su vecindad.

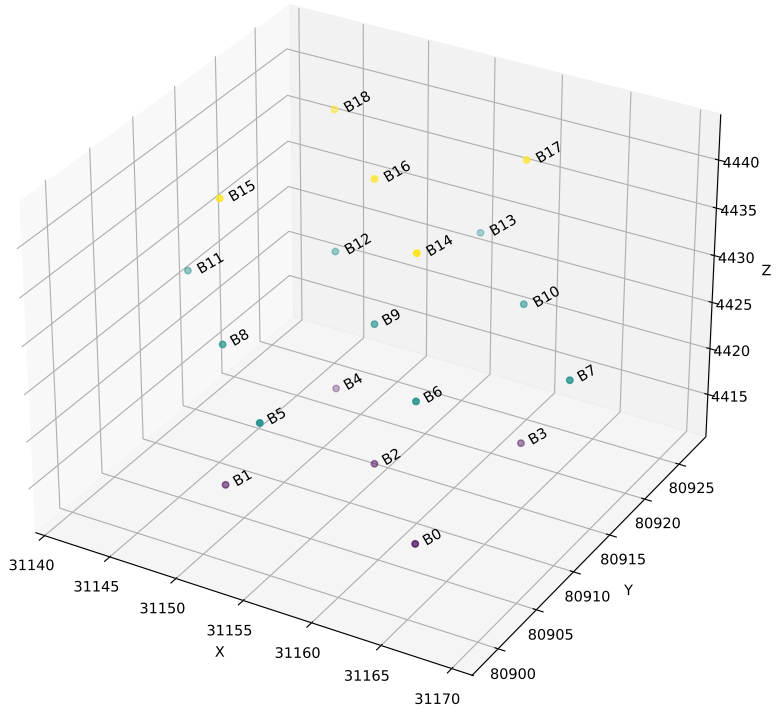
En la propuesta de (Li et al., 2010) se trabaja con datos en dos dimensiones, una diferencia a considerar respecto de esta tesis donde se trabaja con datos en un espacio tridimensional. Debido a esta característica de los datos, es deseable determinar un radio que genere una vecindad con nodos vecinos en la misma altura del nodo central, y también, nodos vecinos que estén a una altura superior e inferior.

$$d(\boldsymbol{p}, \boldsymbol{q}) = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2} \tag{4.2}$$

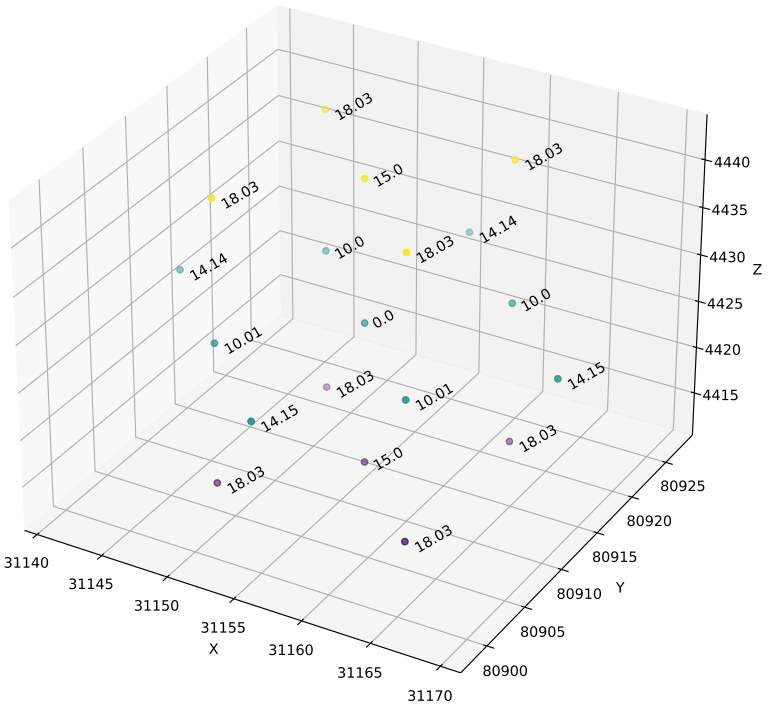
Considerando lo anterior, se determina un radio especifico que resulta en un total de 18 nodos vecinos. La configuración de cada nodo vecino para un nodo central (punto $B9$) se aprecia en la Figura 4.6a, donde se pueden observar tres niveles separados por altura, el nivel en el centro de color celeste, donde está el nodo central rodeado de 8 nodos; el nivel inferior de color morado, con 5 nodos; y el nivel superior de color amarillo, con 5 nodos. Esta configuración se ajusta a los valores discretos en las coordenadas, de manera que se pueda disponer de los nodos más próximos en la misma altura del nodo central en adición a nodos en una altura superior e inferior. En la figura 4.6b cada nodo es etiquetado con su distancia al nodo central, lo que refuerza la idea del uso de un radio que se ajuste a la realidad en los datos.

La aplicación de este proceso de inclusión de nodos vecinos en cada nodo redujo el número de instancias de 732,870 a 545,768 (un 25.53 % menos). Esta reducción se produce debido a que se descartan aquellos nodos que no disponen de 18 nodos vecinos en la configuración descrita en el párrafo anterior. En la Figura 4.7 se puede observar una representación del conjunto de datos de 732,870 instancias con 43 variables, junto a una representación del conjunto de datos de 545,768 instancias que contiene información del nodo central (B_9) y de los nodos vecinos, cada uno con 44 variables al incluir la distancia euclidiana. Disponer de atributos de nodos cercanos enriquece la información disponible en una instancia, pero como costo se reduce el número total de instancias disponibles para los procesos de entrenamiento y prueba de los modelos de DL. Esta nueva estructuración de los datos permite el uso de modelos Convolutional Neural Network (CNN) y Recurrent Neural Network (RNN) que requieren secuencias de instancias como entrada, tras brindar un nodo y su vecindad.

El histograma de los valores de concentración de cobre para este nuevo conjunto de datos, junto con un resumen de estadísticos, se muestra en la Figura 4.8. La distribución de los valores de concentración de cobre no muestra cambios. Se mantiene un sesgo positivo alto y aumentan levemente la mediana, el sesgo y la media, pero se mantienen los valores mínimos y máximo.

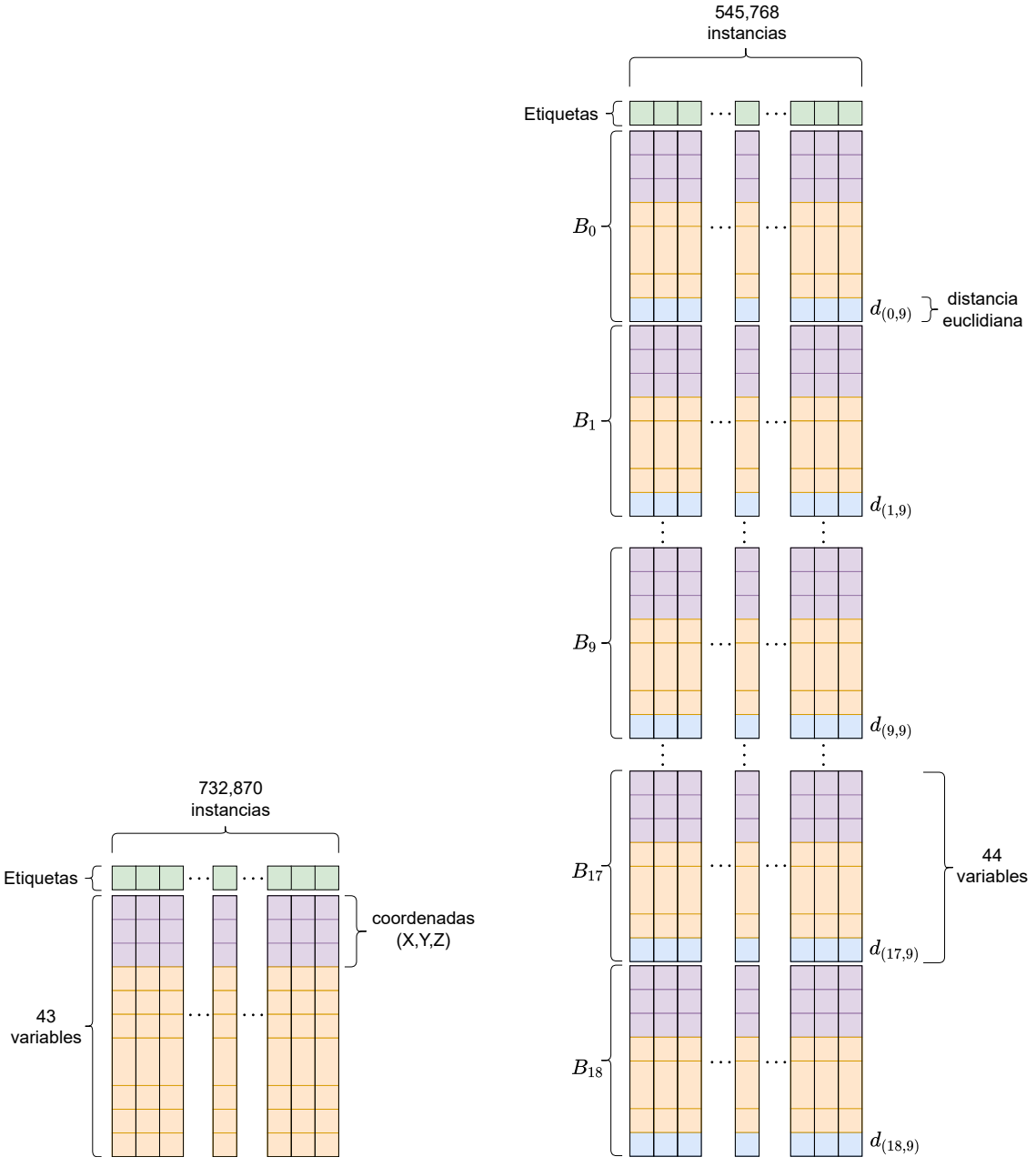


(a) Visualización del orden de los puntos B_i



(b) Distancia de cada punto B_i al punto B_9 en el centro de la Figura 4.6a

Figura 4.6: Visualizaciones de puntos vecinos



(a) 43 variables

(b) 19 nodos × 44 variables

Figura 4.7: Conjuntos de datos e información de su composición

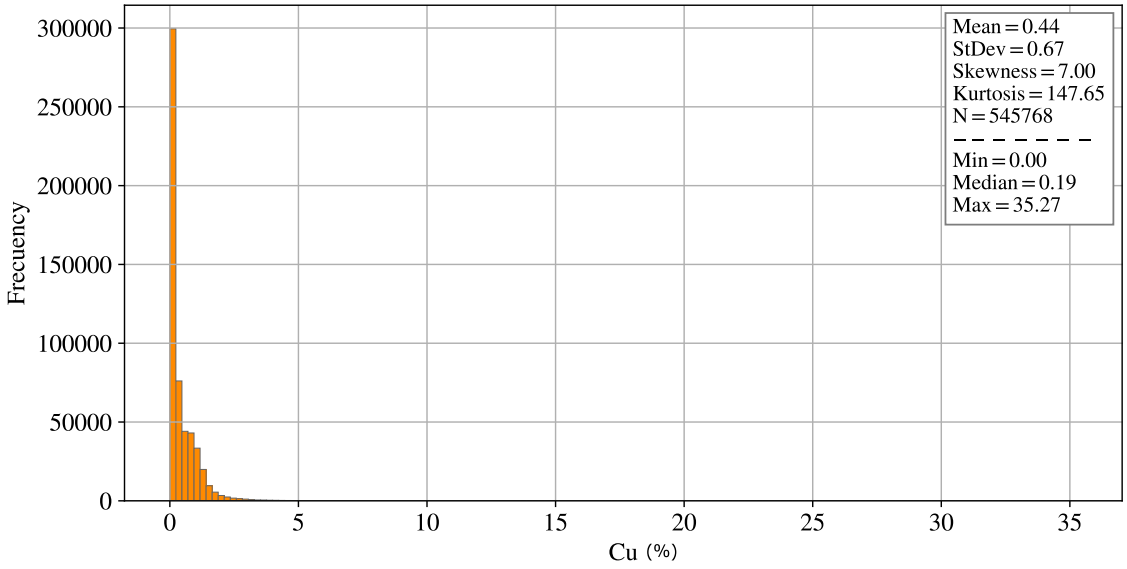


Figura 4.8: Histograma de concentración de cobre para conjunto de datos con nodos vecinos

4.3. Modelos implementados

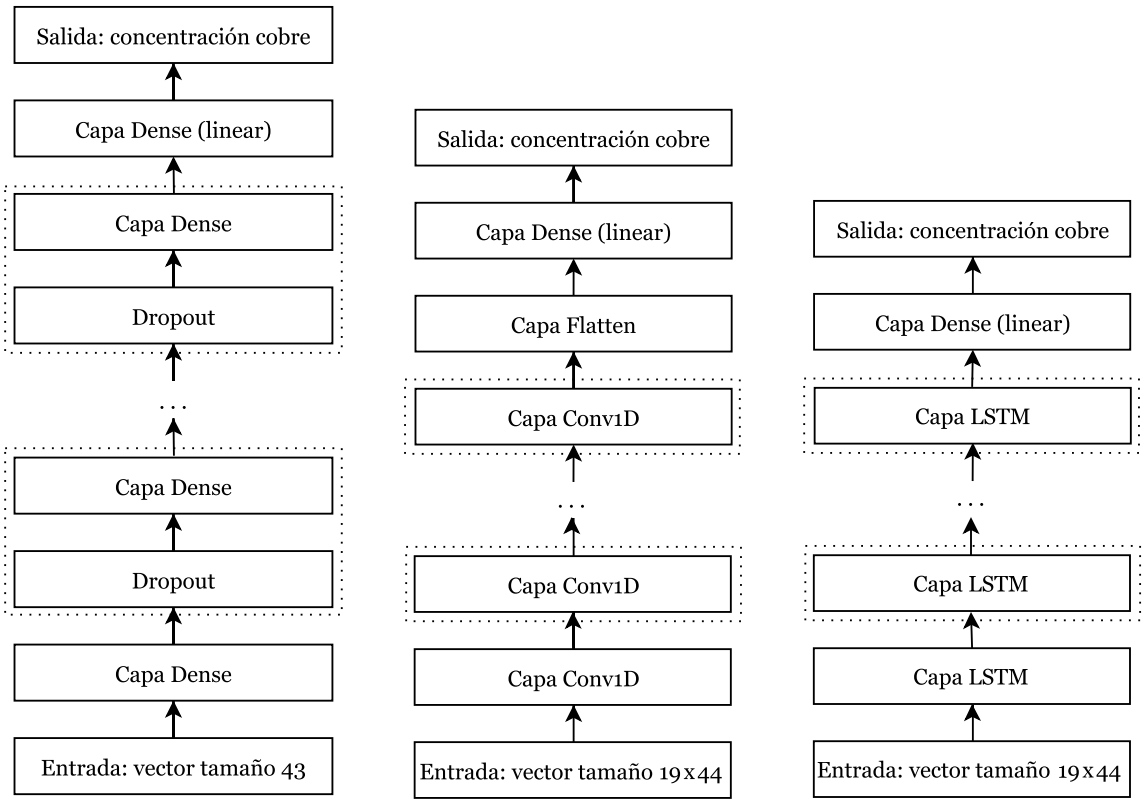
En esta tesis se consideran los modelos Feedforward Neural Network (FNN), Convolutional Neural Network (CNN) y Recurrent Neural Network (RNN). Estos modelos son implementados haciendo uso de la biblioteca de código Keras. Esta biblioteca tiene implementados diferentes algoritmos y componentes de modelos de ML/DL, que son empleados como bloques de construcción para diseñar modelos con variadas arquitecturas. Para cada capa, una clase[1] es implementada y al ser instanciada, deben indicarse los valores para los hiperparámetros disponibles en la implementación de la clase. Si no se indican valores explícitamente, se emplean valores por defecto.

Para el modelo FNN se emplean capas densamente conectadas (Densely-connected Neural Network), implementadas mediante la clase Dense[2], la que realiza la operación definida en la Ecuación 2.14. Todos los experimentos para este modelo se realizan con un conjunto de datos de 43 variables, por lo que cada modelo FNN tiene una capa de entrada o primera capa de tamaño 43 y dado que el objetivo es estimar la concentración mineral, una capa de salida de tamaño 1. Cada capa Dense puede ser configurada mediante varios hiperparámetros, pero en los experimentos realizados solo se consideran la cantidad de neuronas y la función de activación, el resto de hiperparámetros mantiene su valor por defecto. La cantidad de capas ocultas es un hiperparámetro que también se considera. Como se ve en la Figura 4.9a luego de la capa de entrada, se emplea una capa Dense, la

[1]Una clase es un concepto empleado en programación orientada a objetos para referirse a una plantilla de código. Mediante las clases se pueden crear objetos, proporcionando valores iniciales para definir un estado e implementaciones de comportamiento llamadas funciones o métodos.

[2]https://keras.io/api/layers/core_layers/dense

primera capa oculta. Posteriormente por cada capa oculta adicional se implementa una capa Dropout[3] seguida de una capa Dense.



(a) Capas en modelo FNN (b) Capas en modelo CNN (c) Capas en modelo RNN

Figura 4.9: Esquemas de capas en implementación de cada modelo

Para el modelo CNN de 1 dimensión se emplea la clase Conv1D[4]. Los hiperparámetros considerados para cada capa Conv1D son la cantidad de filtros, el tamaño de kernel y la función de activación. El resto de hiperparámetros mantiene su valor por defecto. En la Figura 4.9b se puede observar un ejemplo de las arquitecturas implementadas para CNN. Se emplea una capa de entrada que recibe un grupo de 19 nodos, cada uno con 44 variables. Estos datos corresponden al conjunto de datos que disponen de información del nodo central para el que se realiza la estimación en adición con información de nodos cercanos. La capa de salida genera solo un valor que corresponde a la concentración de mineral estimada. Cada capa oculta corresponde a una capa Conv1D. Previo a la capa de salida es necesario incluir una capa Flatten[5], seguida de una capa Dense. La capa Flatten aplana el vector de datos que reciba de entrada, por ejemplo, si recibiera el grupo de

[3]https://keras.io/api/layers/regularization_layers/dropout
[4]https://keras.io/api/layers/convolution_layers/convolution1d/
[5]https://keras.io/api/layers/reshaping_layers/flatten/

nodos (19×44) los concatenaría de forma secuencial generando un vector de 836 valores ($19 * 44 = 836$). Para la capa Dense se emplea la función linear (4.5), donde la salida es la misma suma ponderada calculada, esta capa permite sintetizar el vector proveniente de la capa Flatten y producir solo un valor.

Para el modelo RNN se emplea específicamente el modelo Long Short-Term Memory (LSTM), implementado mediante la clase LSTM[6]. Los hiperparámetros considerados para cada capa LSTM son la cantidad de neuronas o unidades que definen la salida de la capa, el valor de dropout que se aplica sobre el vector de entrada de la capa y dropout recurrente que se aplica al estado recurrente. El resto de hiperparámetros mantiene su valor por defecto. Para los modelos CNN y RNN se emplea el mismo conjunto de datos, por ello ambos tienen una capa de entrada 19×44. Cada capa oculta corresponde a una capa LSTM y para generar la salida se emplea una capa Dense que permite sintetizar el vector de salida de la última capa LSTM en solo un valor de salida.

4.4. Optimizadores

Para cada uno de los modelos se consideran los optimizadores SGD, RMSprop, Adam, Adadelta, Adagrad, Adamax y Nadam, disponibles en la API[7] de la biblioteca Keras. Cada uno de estos optimizadores tiene un impacto en el proceso de entrenamiento y posterior desempeño de un modelo. Entre todos los optimizadores, el método Stochastic Gradient Descent (SGD) es la base e inspiración para los demás optimizadores.

Como se muestra en la Figura 4.10 una función de costo puede tener varios valores mínimos locales, pero solo un mínimo global (Ruder, 2016). Dado que el descenso del gradiente avanza desde un punto en la curva y en dirección opuesta al gradiente tendera a un mínimo local. Dependiendo del valor inicial de los parámetros el descenso se puede detener en un mínimo local. Esto se puede evitar en parte con el ratio de aprendizaje y por lo mismo es un aspecto importante en los optimizadores.

Adagrad (Duchi et al., 2011) (Adaptative Gradient) es un algoritmo de optimización basado en gradiente que adapta el ratio de aprendizaje para cada parámetro en un modelo, desempeñando grandes actualizaciones para parámetros asociados a características poco frecuentes y pequeñas actualizaciones para parámetros asociados a características frecuentes. Esto produce un desempeño más robusto al tratar con conjuntos de datos sesgados, permitiendo que tipos de datos menos frecuentes puedan tener más incidencia en el ajuste de parámetros.

Adagrad acumula los gradientes o ajustes utilizados en cada iteración durante el proceso de entrenamiento, esta información es empleada para la reducción del ratio de aprendizaje en cada parámetro, sin embargo, esto resulta en un cambio agresivo en la reducción del ratio de aprendizaje. Adadelta (Zeiler, 2012) es una variación de Adagrad que busca evitar este inconveniente generando una ventana durante la que se consideran los gradientes, disminuyendo el ratio de aprendizaje de manera uniforme. RMSprop (Root Mean Square

[6] https://keras.io/api/layers/recurrent_layers/lstm
[7] https://keras.io/api/optimizers/

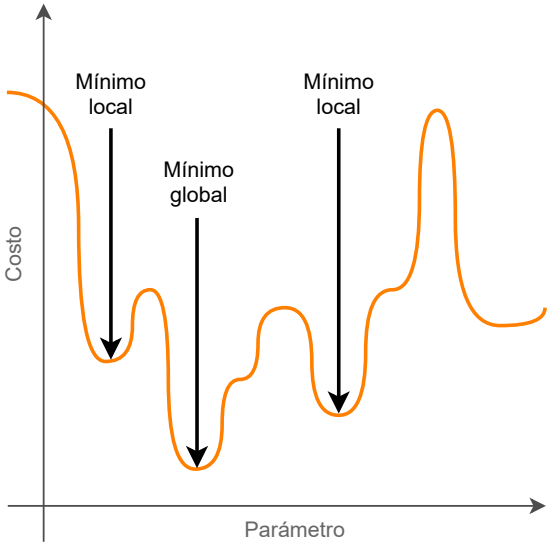


Figura 4.10: Mínimo local y mínimo global. Basado en (Chollet, 2017)

Propagation) funciona de la misma manera que Adadelta, pero ambos optimizadores fueron desarrollados de forma independiente (Ruder, 2016).

SGD tiene problemas cuando la superficie del espacio de soluciones tiene hendiduras o valles como en la Figura 4.11, oscilando entre pendientes de mínimos locales en busca del mínimo global, lo que retrasa el proceso de entrenamiento y puede producir un estancamiento en mínimos locales. Esta situación suele agravarse a medida que el ratio de aprendizaje disminuye. Momentum es un método que ayuda a acelerar SGD en la dirección relevante y reduce las oscilaciones mediante la adición de aceleración y velocidad (Ruder, 2016). Esto permite que, durante el descenso del gradiente en una pendiente de la superficie del espacio de soluciones, se acumule velocidad manteniendo la dirección de descenso, disminuyendo el efecto que tendría un mínimo local, el que podría desviar la dirección a otro punto en el espacio de soluciones.

Adam (Adaptive Moment Estimation) (Kingma and Ba, 2015) combina la adaptación del ratio de aprendizaje para cada parámetro propuesta para los algoritmos RMSprop y Adadelta con el Momentum en la búsqueda por un mínimo global. Adamax (Kingma and Ba, 2015) es una variación de Adam, ambos métodos fueron propuestos en la misma publicación.

Nadam (Nesterov-accelerated Adaptive Moment Estimation) (Dozat, 2016) es un optimizador que resulta de la combinación entre el método Adam y Nesterov Accelerated Gradient (NAG) (Nesterov, 1983). Mientras que Momentum primero calcula el gradiente actual y luego da un gran salto en la dirección del gradiente acumulado actualizado, NAG primero hace un gran salto en la dirección del gradiente acumulado anterior, mide el gradiente y luego hace una corrección. Esta actualización anticipada evita que se avance demasiado rápido y da como resultado una mayor capacidad de respuesta (Ruder, 2016).

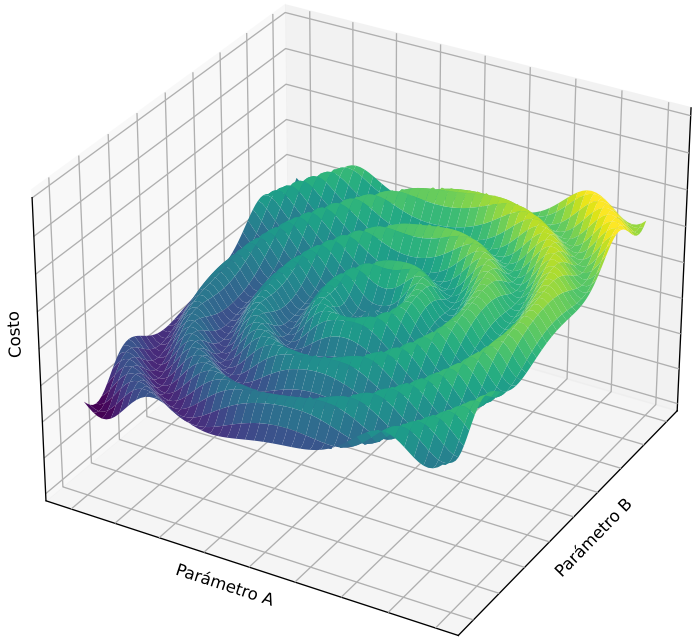


Figura 4.11: Espacio de soluciones para dos parámetros. Basado en (Ruder, 2016)

4.5. Funciones de activación

Respecto a las funciones de activación, existe una amplia variedad para ser empleadas en las capas de un modelo. La biblioteca Keras dispone de una API[8] de funciones de activación para facilitar su implementación. Las funciones de activación empleadas en los experimentos son softmax, softplus, softsign, ReLU, tanh, sigmoid, hard sigmoid y linear. ReLU es mencionada en la sección 2.3.1, es una de las funciones más empleadas en NN. La función más sencilla es la función lineal (Ecuación 4.3), que no realiza ningún cambio en un vector z.

$$linear(z) = z \tag{4.3}$$

La función de activación sigmoid (Ecuación 4.4) tiene un rango de valores que se encuentra entre 0 y 1. Con un valor de entrada igual a cero el valor de salida es 0.5, para un valor de entrada menor a -4 la salida generada tiende a 0 y para un valor de entrada mayor a 4 la salida tiende a 1. Por su rango de valores se puede emplear para regular el paso de valores, 0 impediría el paso de un valor, 1 permitirá el paso de un valor en su integridad y valores intermedios también permitirían el paso de un valor pero de forma parcial . El valor medio en el rango de valores generado por la función sigmoid no está centrado en 0 a diferencia de la función tangente hiperbólica (tanh - Ecuación 4.5) que genera valores entre -1 y 1.

[8] https://keras.io/api/layers/activations/

$$sigmoid(z) = \frac{1}{1 + e^{-z}} \tag{4.4}$$

$$tanh(z) = \frac{sinh(z)}{cosh(z)} = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}} \tag{4.5}$$

Hard sigmoid es un tipo de función que mantiene la forma de la función sigmoid, generando valores entre 0 y 1, pero de una forma más simple, reduciendo el tiempo de cómputo. La forma de hard sigmoid puede variar según la implementación, en la biblioteca de Tensorflow[9] está definida de acuerdo con la Ecuación 4.6

$$hard\ sigmoid(z) = \begin{cases} 0 & z < -2.5 \\ 0.2z + 0.5 & -2.5 \leq z \leq 2.5 \\ 1 & z > 2.5 \end{cases} \tag{4.6}$$

De acuerdo con la documentación de Keras, softmax convierte un vector z de n valores en un vector de probabilidades. Los valores del vector de probabilidades están entre 0 y 1, y el resultado de su suma debe ser 1. La aplicación de la función softmax puede dividirse en dos pasos, el primero consiste en la suma de todos los elementos del vector z, en la Ecuación 4.7 se emplea el índice j; El segundo paso, consiste en dividir cada valor por separado en z por la sumatoria del paso anterior, en la ecuación se emplea el índice i para diferenciar ambos pasos. Softmax se utiliza a menudo como activación para la última capa de una red de clasificación porque el resultado podría interpretarse como una distribución de probabilidad.

$$softmax(z) = \frac{e^{z_i}}{\sum_{j=1}^{n} e^{z_j}} \tag{4.7}$$

softplus está definida de acuerdo con la Ecuación 4.8. Es una aproximación "suave" de la función de activación ReLu.

$$softplus(z) = \log(e^{z} + 1) \tag{4.8}$$

softsign está definida de acuerdo con la Ecuación 4.9. Es una alternativa a la función de activación tanh, ya que también produce valores entre -1 y 1, pero el cálculo de operaciones involucradas requiere menor tiempo de cómputo.

$$softsign(z) = \frac{z}{|z| + 1} \tag{4.9}$$

[9] https://www.tensorflow.org/api_docs/python/tf/keras/activations/hard_sigmoid

Capítulo 5

Resultados y discusión

5.1. Diseño experimental

Se entrenan los modelos Feedforward Neural Network, Recurrent Neural Network y Convolutional Neural Network para la estimación de concentración mineral, implementados con la biblioteca de código Keras. La entrada de los datos corresponde a la información disponible del LTMP según lo descrito en la sección 4. La variable por estimar corresponde a la concentración de mineral de Cobre.

Las estimaciones generadas por los modelos DL son contrastadas con la concentración de mineral "real" disponible en variables del STMP, y de esta manera se calcula el error de los modelos DL. Mencionado esto, es también importante considerar que las variables de LTMP contienen estimaciones de la concentración mineral generadas mediante métodos geoestadísticos (OK), lo que se contrasta con la información del STMP produciendo un error que se considera como baseline. De este modo, es posible comparar el desempeño de los modelos DL con el desempeño de un método geoestadístico y establecer de forma clara una disminución en el error en la estimación de concentración mineral. Para medir este error, en este capítulo se presenta el valor de las métricas MAE, MSE y R^2 tanto para los modelos entrenados como para los valores considerados como baseline. Estas métricas son las más representativas en problemas de regresión en ML y además, de acuerdo con la revisión de la literatura (Sección 3.3), son las más empleadas en estimación de concentración mineral.

Las métricas MAE y MSE son un promedio, lo que les distingue es que MAE calcula el valor absoluto de la diferencia entre la salida de un modelo y el valor esperado, mientras que MSE calcula el cuadrado de esa diferencia. Esto hace que MAE sea una métrica más sencilla que genera un valor directo del error en las estimaciones, pero como promedio oculta información. En cambio, MSE al emplear un exponente amplifica errores más grandes, impidiendo que pasen desapercibidos al calcularse el promedio. Estas propiedades hacen que suelen ser empleadas en conjunto. Otra característica de las métricas MSE y MAE es que estas generan valores que solo aplican para un conjunto de valores esperados, específicamente para una unidad de medida e incluso una granularidad de valores. Esto hace que los puntajes para las métricas MAE y MSE sean específicos para un problema o conjunto de datos y que no sea sencillo interpretarlos. Por otro lado, la métrica R^2 permite evaluar

la relación lineal entre valores de referencia y su estimación. Permite cuantificar el grado de similitud entre la línea de una regresión lineal de un conjunto de datos y la línea de la función identidad ($f(x) = x$). Los valores generados por esta métrica se encuentran en un rango entre 0 y 1, donde 1 indicaría que no hay error en las estimaciones, es decir, cada una de las estimaciones coincide con el valor de referencia. Esta lógica aplica para cualquier problema y, por lo tanto, es más sencillo de interpretar, basta con estar familiarizado con el rango de valores que genera y su significado. Por esto R^2 se complementa con las métricas MAE y MSE.

Para la validación de los experimentos y sus resultados, es muy relevante la forma en la que se emplean los datos. Es imperante evitar ciertos errores como medir el desempeño de los modelos sobre los conjuntos empleados para entrenamiento. En esta tesis se emplea el esquema de la Figura 5.1. Del conjunto inicial de datos se reserva una fracción para realizar la prueba final, esta fracción de instancias sólo es usada al finalizar los experimentos. El conjunto de instancias restantes primero se emplea en realizar entrenamiento y selección de mejores valores para los hiperparámetros de cada modelo. El entrenamiento y evaluación se lleva a cabo empleando el método de validación K-fold Cross-validation (CV), con K=10 (Zhang et al., 2013), (Jafrasteh et al., 2018), (Caté et al., 2017), (Iglesias et al., 2020). El método CV consiste en dividir el conjunto de datos en K subconjuntos denominados folds, realizando K iteraciones. En cada iteración un fold distinto es empleado para prueba y el resto para entrenamiento. De este modo, en cada iteración se genera un modelo distinto que es evaluado y, por lo tanto, al finalizar todas las iteraciones, se genera un promedio del desempeño. La cantidad de folds empleada debe ser considerada en base a la cantidad de datos disponibles, a la capacidad de cómputo y del tiempo disponible. Un mayor número de folds resulta en una mayor cantidad de iteraciones y en consecuencia mayor tiempo de cómputo, sin embargo, el método CV con un K elevado genera resultados más robustos respecto a variaciones en la selección de datos de entrenamiento y prueba.

Para esta tesis se considera como un experimento el entrenamiento de un modelo y su evaluación para diferentes valores en un hiperparámetro específico mientras el resto de hiperparámetros no varía (*ceteris paribus*). Esto permite dilucidar el efecto que tiene cada hiperparámetro de forma individual. Para comenzar con los experimentos, una vez seleccionados los hiperparámetros a ser analizados, se desarrollan escenarios iniciales donde a los hiperparámetros se les asocian valores arbitrarios.

Para cada experimento se analiza y estudia solo un hiperparámetro variando el valor que puede adquirir y se presentan los resultados en términos de las métricas MAE, MSE y R^2. Los resultados son presentados principalmente en gráficas para facilitar su visualización y análisis. Para cada experimento las gráficas presentan resultados de cada métrica por separado. Adicionalmente, los resultados de los experimento para los modelos FNN, CNN y RNN, se presentan en tablas en los Anexos B, C y D, respectivamente.

Dado que en los experimentos de análisis de hiperparámetros se emplea el método CV, el valor que se presenta para cada métrica corresponde al promedio de las iteraciones. En las tablas disponibles en los anexos se presenta la desviación estándar (DE) de los valores considerados para generar cada promedio. En las gráficas, la desviación estándar se presenta como una línea vertical de color gris oscura.

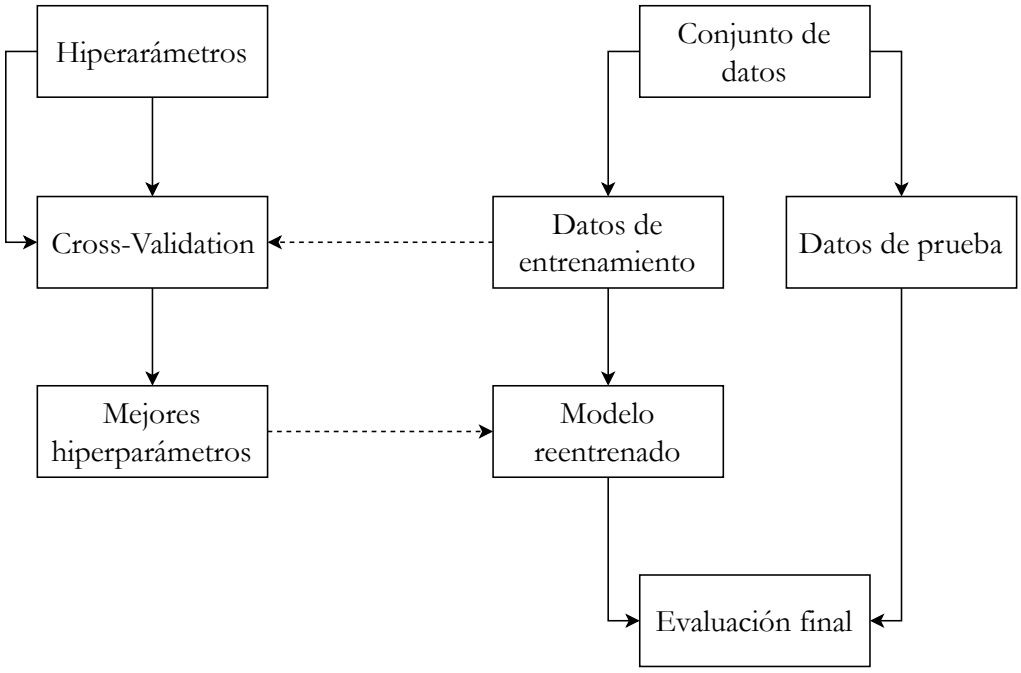


Figura 5.1: Metodología en implementación y distribución de datos

Como se menciona antes, se considera como baseline la comparación entre (1) las estimaciones generadas por métodos geoestadísticos incluida en las variables de concentración mineral del LTMP (cut_lp) y (2) las estimaciones disponibles en STMP (cut_pz). De acuerdo con el esquema empleado para la distribución de los datos (Figura 5.1), se dispone de un valor baseline para el conjunto empleado en el proceso de entrenamiento y para el conjunto de prueba, esto para cada modelo DL. Hay que considerar que el modelo FNN es evaluado con un conjunto de datos cuyo procesamiento difiere al procesamiento empleado en el conjunto de datos con el que se evalúan los modelos CNN y RNN. Por esto, se presentan valores distintos en las métricas del baseline, que es informado en la sección de cada modelo. Por último, el conjunto de entrenamiento es dividido en folds de acuerdo con el método CV como se indicaba antes, por lo que las métricas corresponden a su promedio y desviación estándar. Esto también aplica para el cálculo del baseline en cada métrica.

Todos los experimentos emplean early stop en la etapa de entrenamiento, es decir, se detiene el entrenamiento si no hay una mejora en la función de costo luego de 15 iteraciones, lo que se calcula en base al fold de validación. De los experimentos se extrae la información de la cantidad de épocas que serán empleados en la prueba final. Una vez realizados los experimentos para los modelos y determinados los valores para cada hiperparámetro se procede a entrenar cada modelo con el conjunto de datos de entrenamiento completo y se realiza la evaluación final con los datos de prueba ocultos hasta ese momento. De esta manera, se evita que los modelos seleccionados se ajusten a los datos de prueba.

Los experimentos para el CNN se realizan en un servidor del grupo de investigación

Machine Learning y Visión por Computador (MLVC), mientras que los experimentos de los modelos FNN y RNN se realizan en el servidor de Postgrado de la Facultad de Ciencias Empresariales. En la sección 1.1.5 se describen las especificaciones de ambos servidores.

5.2. Feedforward Neural Network

Para el modelo FNN los valores iniciales de los hiperparámetros considerados son:

- Neuronas: 20
- Capas ocultas: 1
- Función de activación: ReLu
- Dropout: 0.0
- Optimizador: RMSprop

El conjunto empleado para realizar los experimentos con este modelo contiene en total 732,870 instancias. 60,000 instancias son reservadas para una prueba final. Se implementa 10-Fold Cross Validation con las 672,870 instancias restantes, cada fold con 67,287 instancias. Cada iteración en el esquema CV implica el uso de 9 folds para entrenamiento (605,583 instancias) y un fold para validación (67,287 instancias).

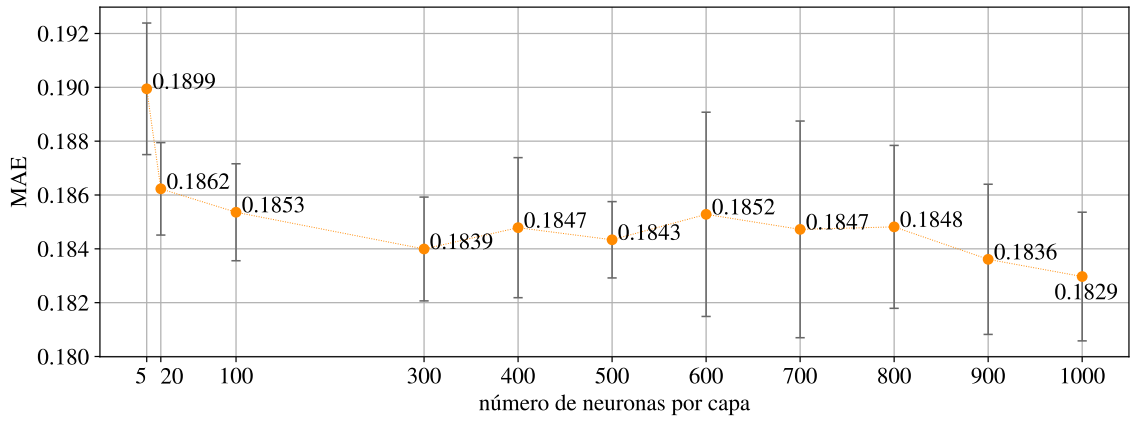
Para las instancias empleadas en validación, de donde se recoge el rendimiento de los modelos en los experimentos, el baseline obtiene 0.1951 en la métrica MAE (DE 0.0014), 0.2238 en métrica MSE (DE 0.0110) y 0.4669 en métrica R^2 (DE 0.0126).

5.2.1. Neuronas

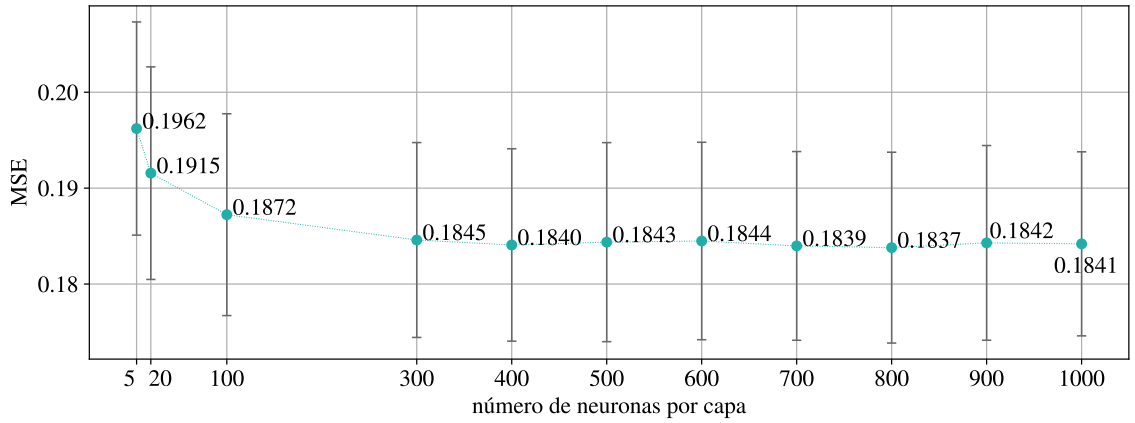
El primer experimento corresponde a la evaluación del rendimiento en función de la cantidad de neuronas, los resultados para las métricas MAE, MSE y R^2 están disponibles en la Figura 5.2. Se entrena el modelo con los hiperparámetros iniciales a excepción de la cantidad de neuronas por capa. Se entrena el modelo con 5, 20, 100, 300, 400, 500, 600, 700, 800, 900 y 1,000 neuronas por capa. De las gráficas de las métricas MSE y R^2 se puede observar una mejora en el rendimiento que converge en las 400 neuronas. También se aprecia un pequeño aumento en las 700 neuronas.

5.2.2. Capas

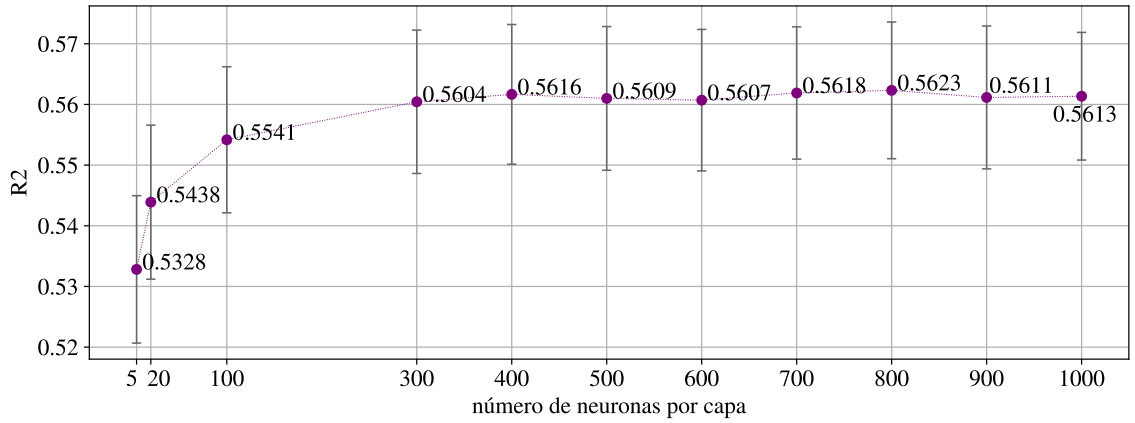
El segundo experimento corresponde a la evaluación del rendimiento en función de la cantidad de capas, se realizan pruebas con 1, 2, 3, 4 y 5 capas. Los resultados para las métricas MAE, MSE y R^2 están disponibles en la Figura 5.3. Para este experimento se emplean 700 neuronas por capa y el resto de hiperparámetros iniciales. Como se puede observar en las gráficas de las tres métricas, cuando el número de capas es 3, se obtiene un mejor desempeño.



(a) Métrica MAE

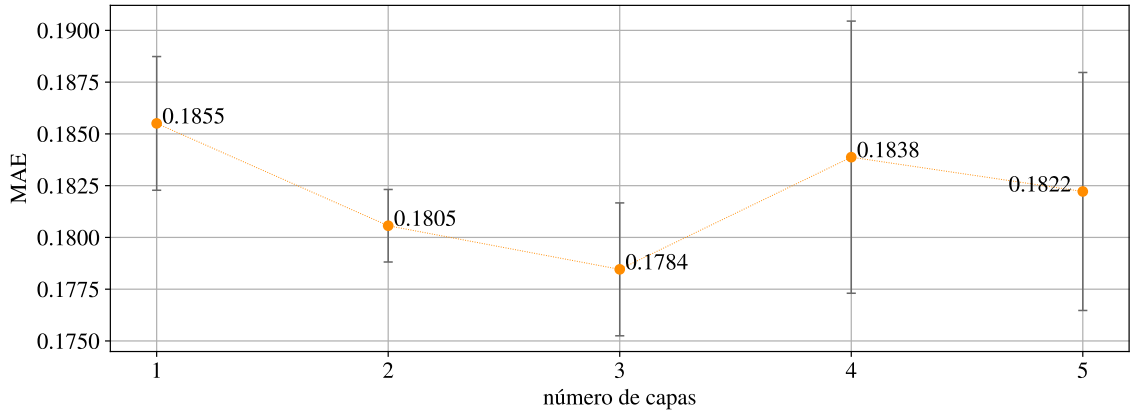


(b) Métrica MSE

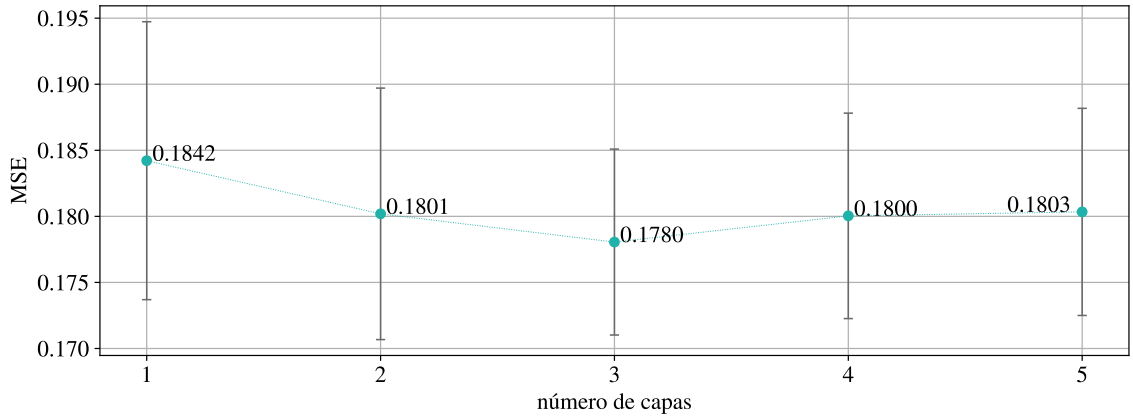


(c) Métrica R^2

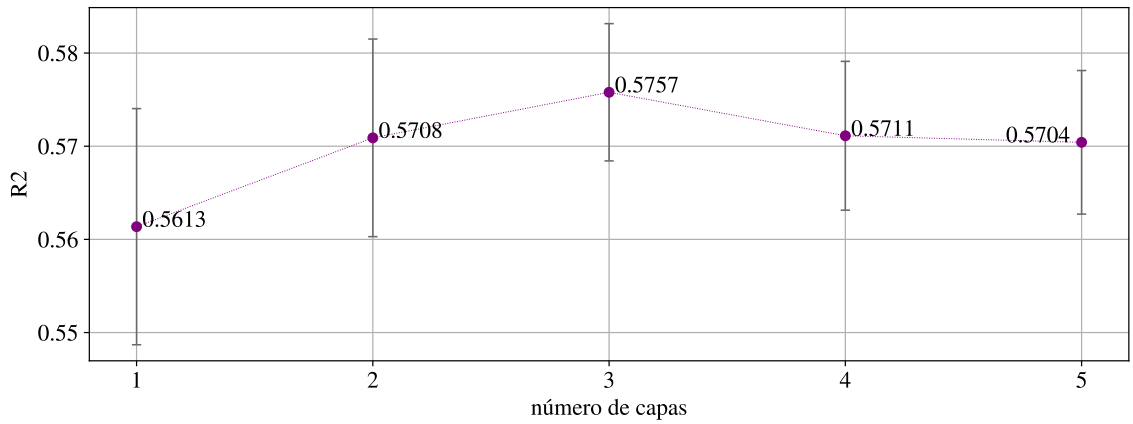
Figura 5.2: Gráficas de desempeño para diferente número de neuronas por capa en la FNN



(a) Métrica MAE



(b) Métrica MSE



(c) Métrica R^2

Figura 5.3: Gráficas de desempeño para diferente número de capas en la FNN

5.2.3. Función de activación

El tercer experimento corresponde a la evaluación del rendimiento respecto a la función de activación en las neuronas, los resultados para las métricas MAE, MSE y R^2 están disponibles en la Figura 5.4. Para este experimento se emplean 3 capas cada una con 700 neuronas y el resto de hiperparámetros mantiene los valores iniciales. Se prueban las funciones de activación softmax, softplus, softsign, relu, tanh, sigmoid, hard sigmoid y linear, descritas en la sección 4.5. Para la métrica MAE la función softplus es la que genera un mejor rendimiento, pero para las métricas MSE y R^2 es la función sigmoide.

5.2.4. Dropout

El cuarto experimento corresponde a la evaluación del rendimiento para diferentes valores en las capas dropout, los resultados para las métricas MAE, MSE y R^2 están disponibles en la Figura 5.5. Los valores para los que se realizan pruebas son 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8 y 0.9. Sólo cuando se emplea 0.1 para dropout se aprecia una mejora en las métricas MSE y R^2. Estas pruebas se realizan con 3 capas ocultas de 700 neuronas cada una y la función de activación sigmoide.

5.2.5. Optimizador

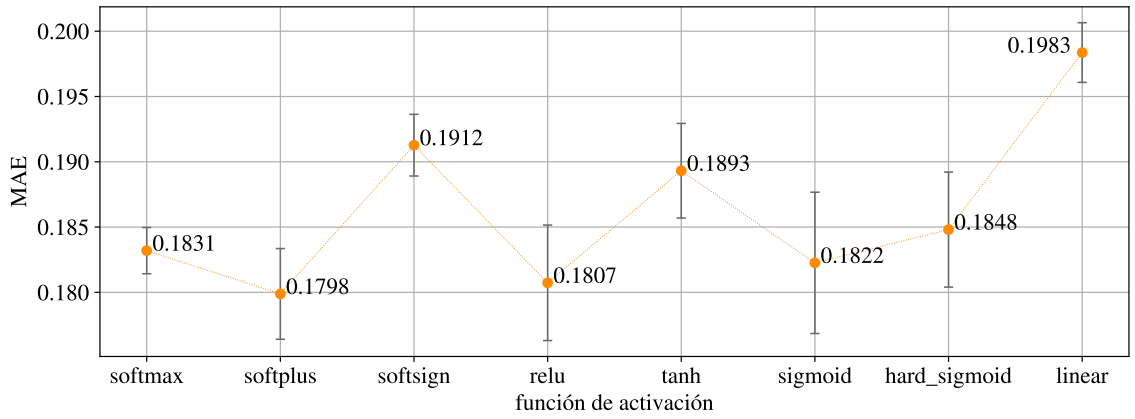
El quinto y último experimento para el modelo FNN corresponde a la evaluación del rendimiento para los algoritmos optimizadores SGD, RMSprop, Adagrad, Adadelta, Adam, Adamax y Nadam descritos en la sección 4.4. Los resultados para las métricas MAE, MSE y R^2 están disponibles en la Figura 5.6. Con 3 capas ocultas de 700 neuronas cada una, la función de activación sigmoide y 0.1 de dropout, el mejor optimizador resulta ser Nadam. Los resultados en este experimento son consistentes en las tres métricas.

5.3. Convolutional Neural Network

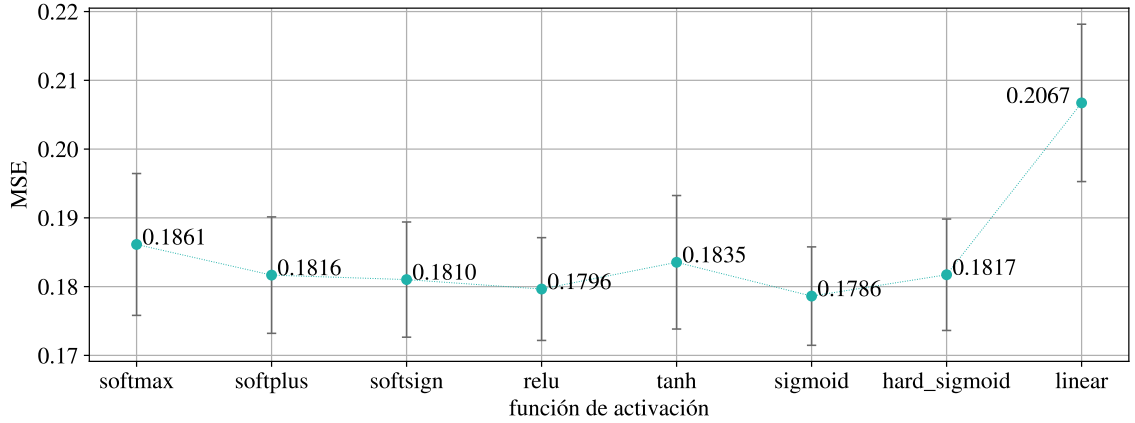
Para el modelo CNN los valores iniciales de los hiperparámetros considerados son:

- Filtros: 32
- Tamaño kernel: 5
- Capas: 1
- Función de activación: ReLu
- Optimizador: RMSprop

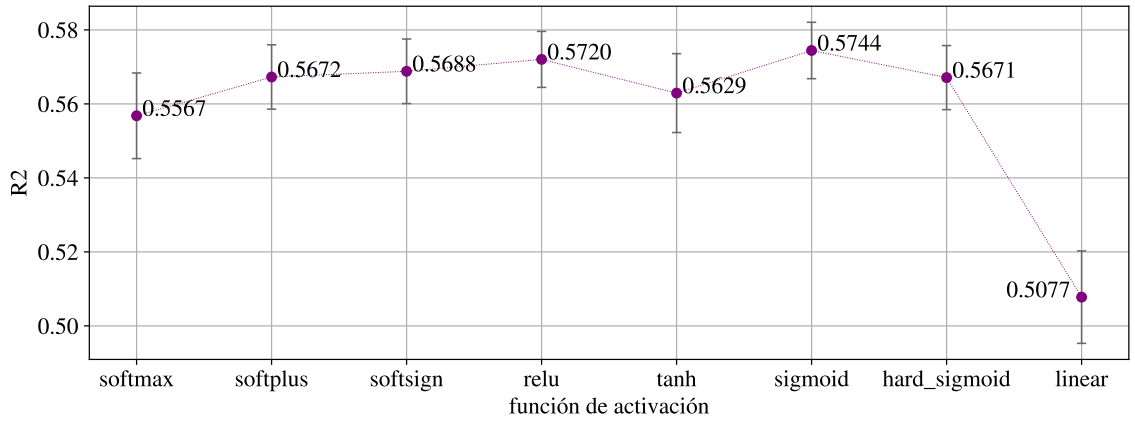
El conjunto empleado contiene en total 545,768 instancias. 45,000 instancias son reservadas para una prueba final. Se implementa 10-Fold Cross Validation con las 500,768 instancias restantes, cada fold con 50,076 instancias. Cada iteración en el esquema CV



(a) Métrica MAE

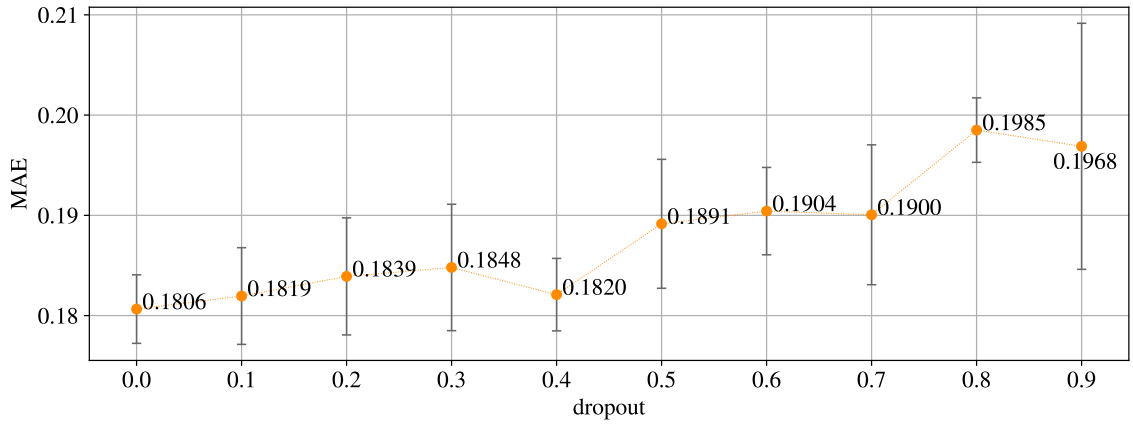


(b) Métrica MSE

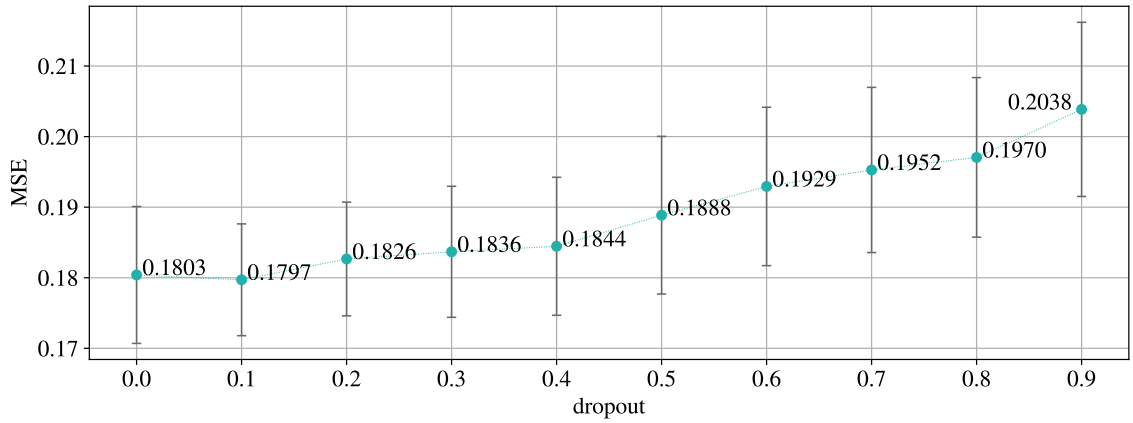


(c) Métrica R^2

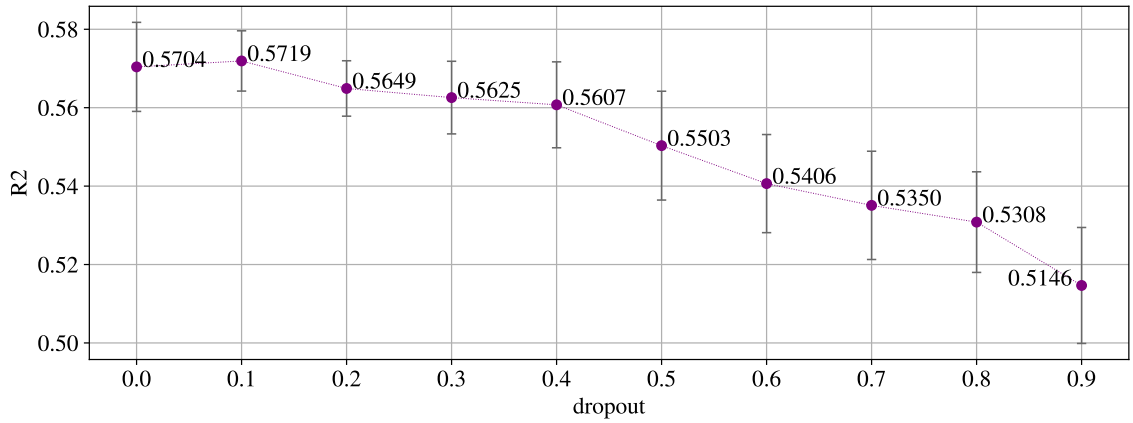
Figura 5.4: Gráficas de desempeño para diferentes funciones de activación en la FNN



(a) Métrica MAE

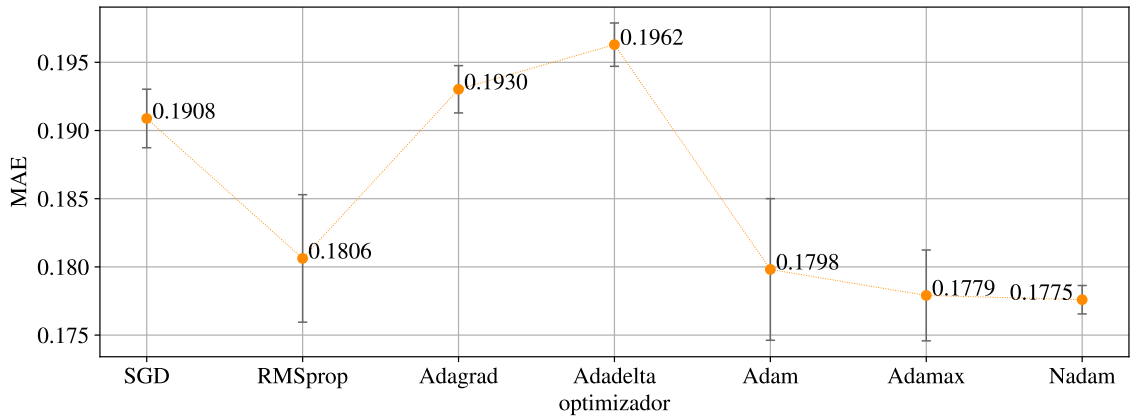


(b) Métrica MSE

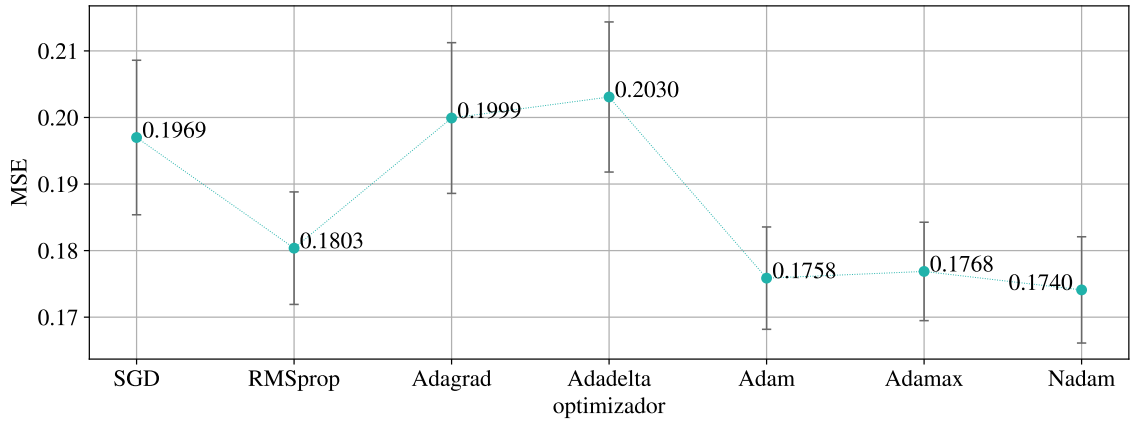


(c) Métrica R^2

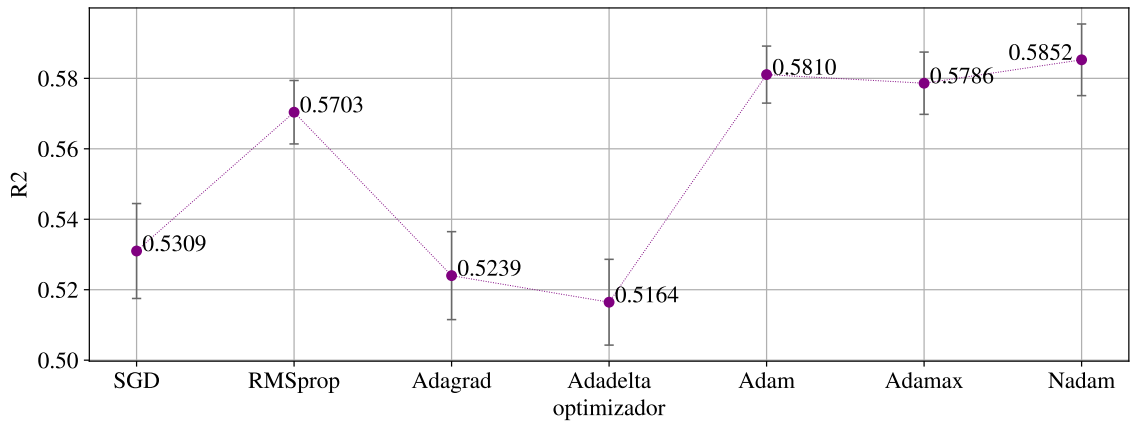
Figura 5.5: Gráficas de desempeño para diferentes valores de dropout en la FNN



(a) Métrica MAE



(b) Métrica MSE



(c) Métrica R^2

Figura 5.6: Gráficas de desempeño para diferentes optimizadores en la FNN

implica el uso de 9 folds para entrenamiento (450,684 instancias) y un fold para validación (50,076 instancias).

Para las instancias empleadas en validación, de donde se recoge el rendimiento de los modelos en los experimentos, el baseline obtiene 0.2032 en la métrica MAE (DE 0.0016), 0.2425 en métrica MSE (DE 0.0144) y 0.4617 en métrica R^2 (DE 0.0138).

5.3.1. Filtros

El primer experimento corresponde a la evaluación del rendimiento respecto a la cantidad de filtros por capa. Se realizan pruebas con 16, 32, 64, 128, 256 y 512 filtros. Los resultados para las métricas MAE, MSE y R^2 están disponibles en la Figura 5.7. Para las métricas MSE y R^2 se observa una convergencia en los 128 filtros, aunque para los 512 se presenta una leve mejora.

5.3.2. Tamaño kernel

El segundo experimento corresponde a la evaluación del rendimiento respecto al tamaño de kernel, los resultados se presentan en la Tabla 5.1. Se prueban kernels de tamaño 3 y 5 empleando 512 filtros. En las métricas MSE y R^2 se obtienen mejores resultados con un kernel tamaño 3.

Tabla 5.1: CNN: Resultados por tamaño de kernel

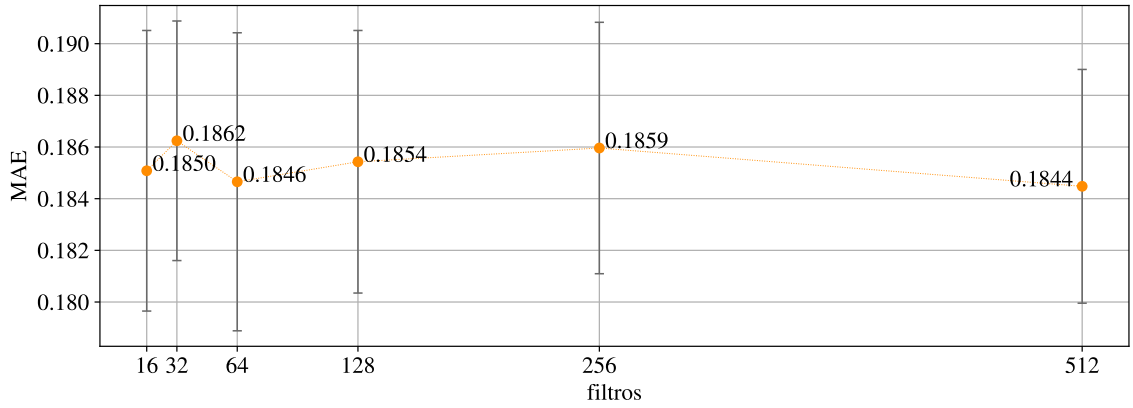
kernel	MAE (DE)	MSE (DE)	R^2 (DE)
3	0.1852 (0.0053)	**0.1743** (0.0133)	**0.6131** (0.0189)
5	**0.1844** (0.0045)	0.1759 (0.0142)	0.6097 (0.0208)

5.3.3. Capas

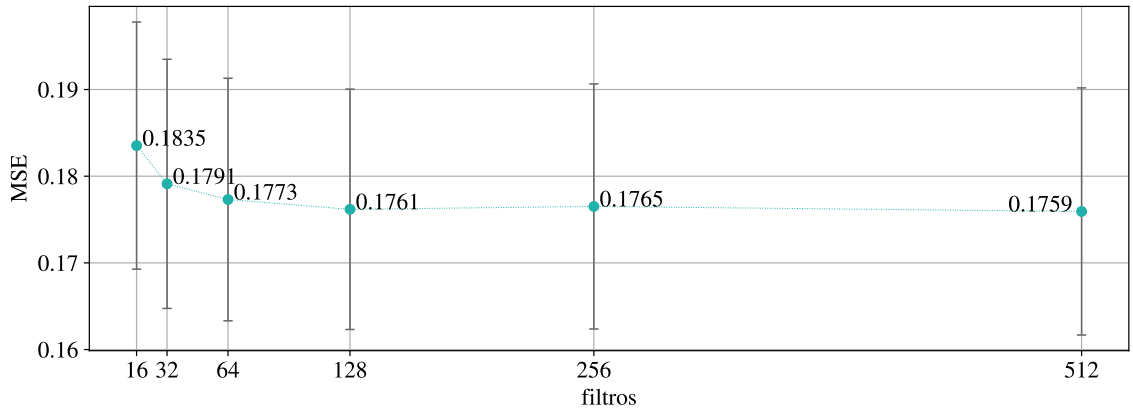
El tercer experimento corresponde a la evaluación del rendimiento respecto a la cantidad de capas, los resultados para las métricas MAE, MSE y R^2 están disponibles en la Figura 5.8. Los mejores resultados se obtienen con 3 capas. Cada capa con 512 filtros y un kernel de tamaño 3.

5.3.4. Función de activación

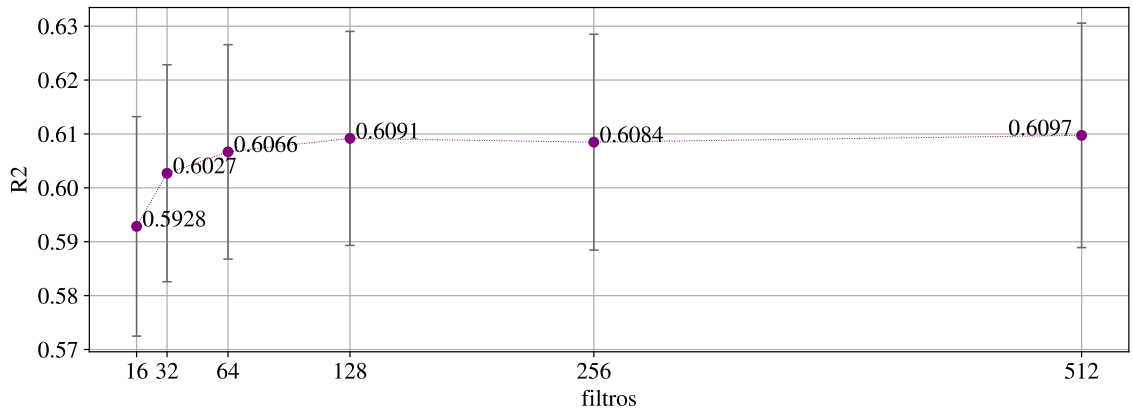
El cuarto experimento corresponde a la evaluación del rendimiento respecto a la función de activación asignada a cada capa, los resultados para las métricas MAE, MSE y R^2 están disponibles en la Figura 5.9. Empleando 2 capas, cada una con 512 filtros y kernel de tamaño 3, se prueban las funciones relu, linear(None), sigmoid, tanh, softmax, softplus, softsign y hard sigmoid. También se considera el no uso de función de activación (None). La función sigmoide obtiene los mejores resultados, superando levemente a la función de activación ReLu.



(a) Métrica MAE

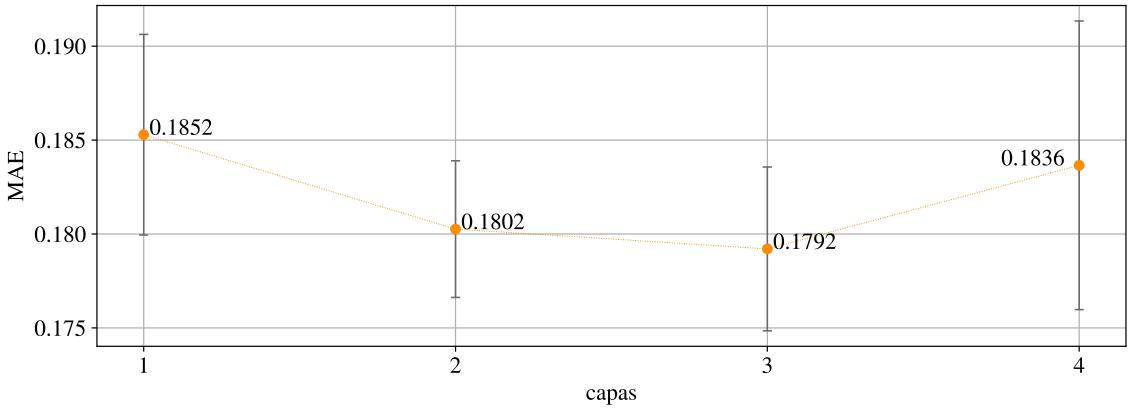


(b) Métrica MSE

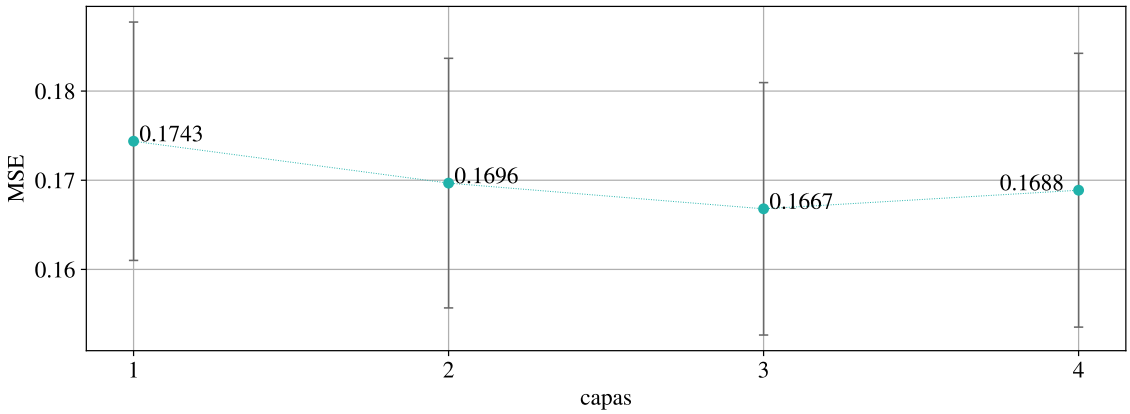


(c) Métrica R^2

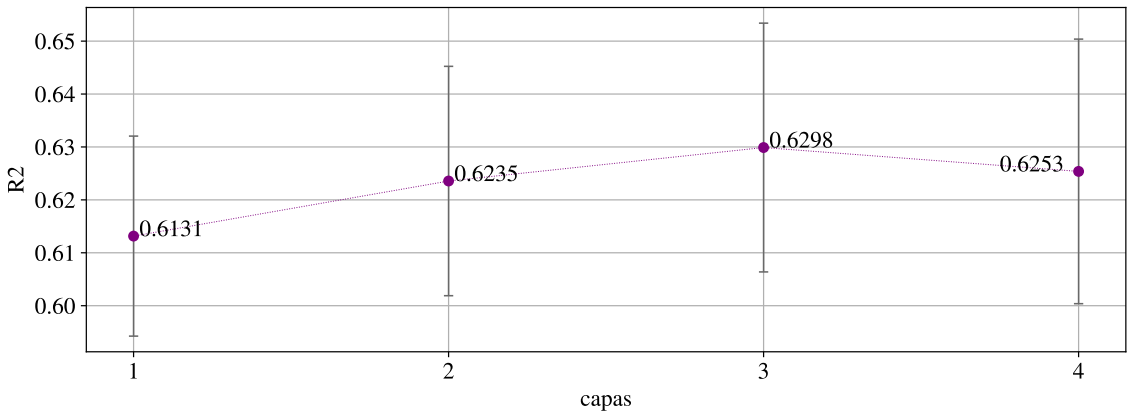
Figura 5.7: Gráficas de desempeño para diferentes cantidad de filtros en la CNN



(a) Métrica MAE

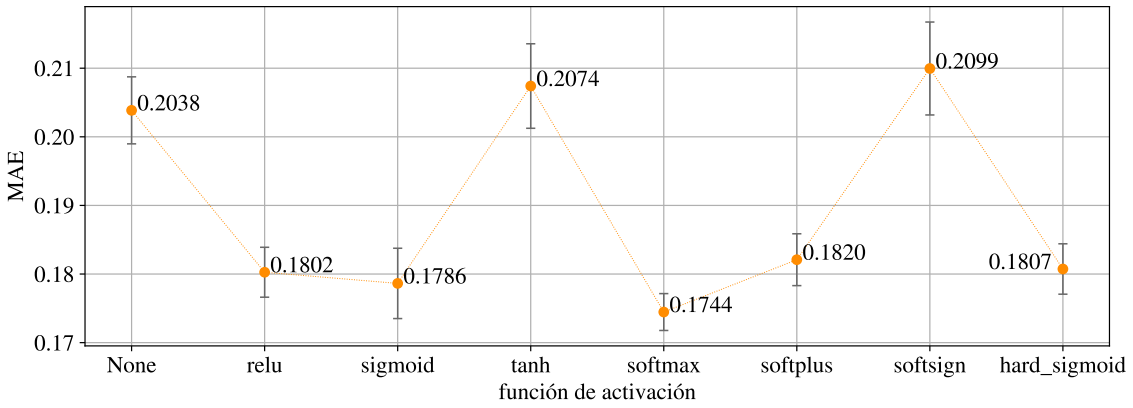


(b) Métrica MSE

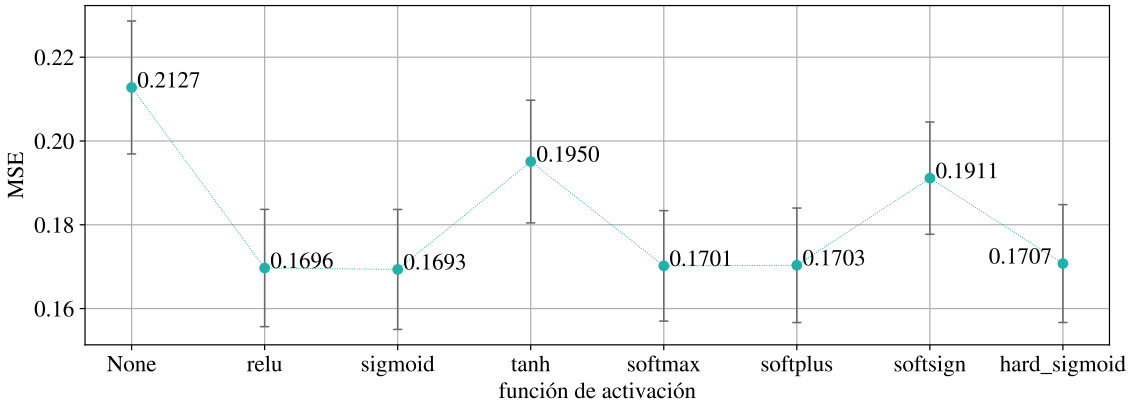


(c) Métrica R^2

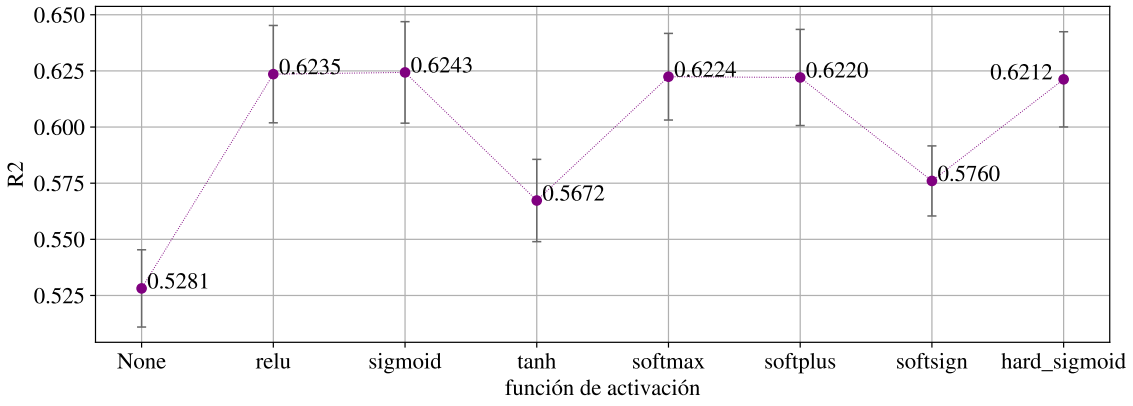
Figura 5.8: Gráficas de desempeño por cantidad de capas en la CNN



(a) Métrica MAE



(b) Métrica MSE



(c) Métrica R^2

Figura 5.9: Gráficas de desempeño por función de activación en la CNN

5.3.5. Optimizador

El quinto y último experimento para el modelo CNN corresponde a la evaluación del rendimiento para diferentes algoritmos de optimización, los resultados para las métricas MAE, MSE y R^2 están disponibles en la Figura 5.10. Nadam es el optimizador con el que se obtienen los mejores resultados en las tres métricas.

5.4. Recurrent Neural Network

Para el modelo RNN los valores iniciales de los hiperparámetros considerados son:

- Neuronas: 50
- Capas ocultas: 2
- Dropout: 0.0
- Dropout recurrente: 0.0
- Activación: tanh
- Activación recurrente: sigmoid
- Optimizador: RMSprop

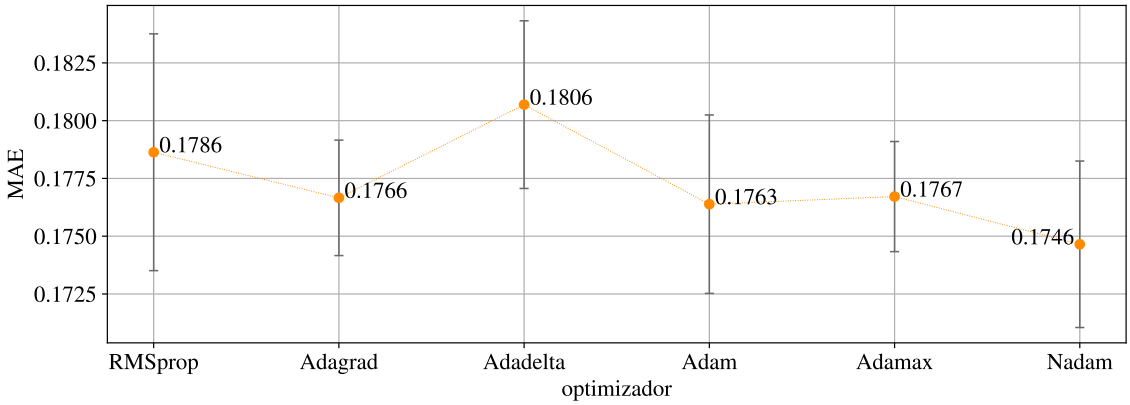
El conjunto empleado es el mismo utilizado para los experimentos con los modelos CNN, que contiene en total 545,768 instancias y donde 45,000 instancias son reservadas para una prueba final. Se implementa 10-Fold Cross Validation con las 500,768 instancias restantes, cada fold con 50,076 instancias. Cada iteración en el esquema CV implica el uso de 9 folds para entrenamiento (450,684 instancias) y un fold para validación (50,076 instancias).

Para las instancias empleadas en validación, de donde se recoge el rendimiento de los modelos en los experimentos, el baseline obtiene 0.2032 en la métrica MAE (DE 0.0016), 0.2425 en métrica MSE (DE 0.0144) y 0.4617 en métrica R^2 (DE 0.0138).

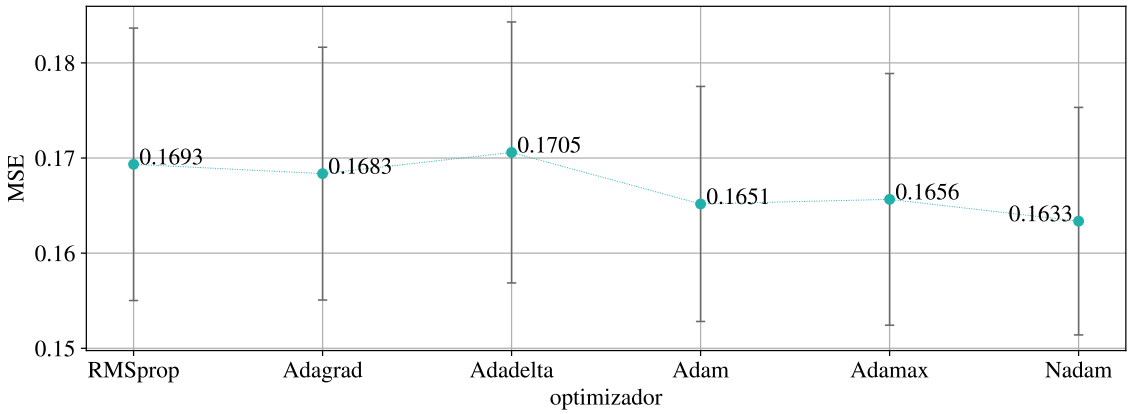
En los experimentos no se analiza el impacto de los hiperparámetros función de activación regular, función de activación recurrente y optimizadores, debido a su excesivo tiempo de cómputo, que excede el horizonte temporal de ejecución de esta tesis.

5.4.1. Neuronas

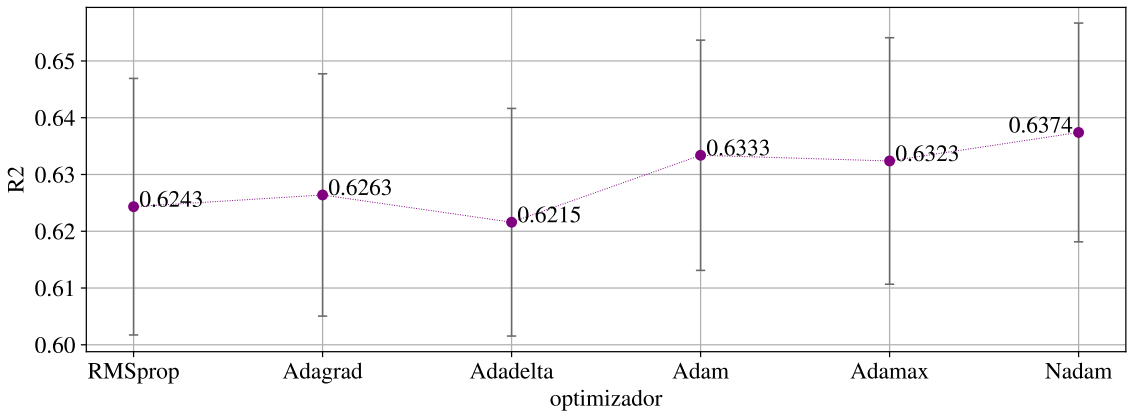
El primer experimento corresponde a la evaluación del rendimiento respecto a la cantidad de neuronas por capa, los resultados para las métricas MAE, MSE y R^2 están disponibles en la Figura 5.11. Se realizan pruebas con 5, 20, 50, 100, 300 y 500 neuronas. De acuerdo con los resultados en las tres métricas, el modelo alcanza el mejor desempeño con 500 neuronas por capa.



(a) Métrica MAE

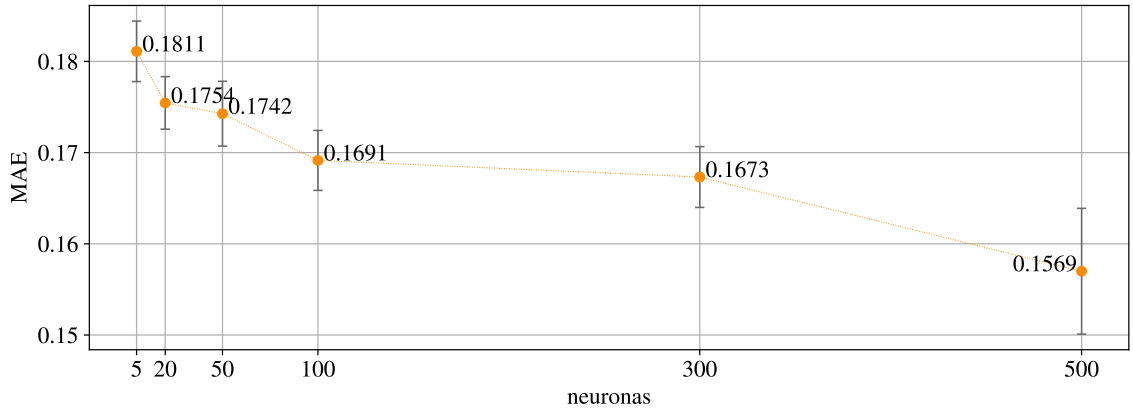


(b) Métrica MSE

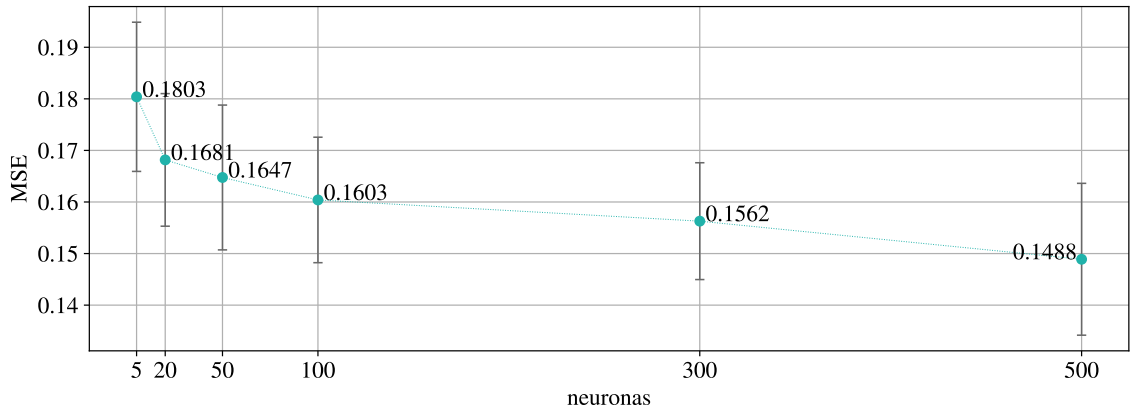


(c) Métrica R^2

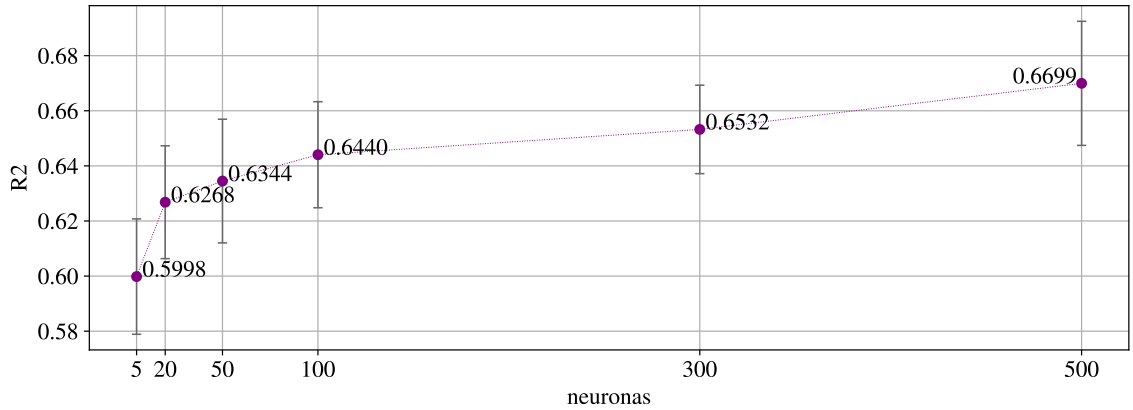
Figura 5.10: Gráficas de desempeño para diferentes optimizadores en la CNN



(a) Métrica MAE



(b) Métrica MSE



(c) Métrica R^2

Figura 5.11: Gráficas de desempeño para diferentes cantidades de neuronas en la RNN

5.4.2. Capas

El segundo experimento corresponde a la evaluación del rendimiento respecto a la cantidad de capas, los resultados para las métricas MAE, MSE y R^2 están disponibles en la Figura 5.12.

Las pruebas se realizan con 100 neuronas, a pesar de que con 500 neuronas se alcanzan los mejores resultados. Esto debido al excesivo tiempo de cómputo que se requiere con tal cantidad de neuronas por capa, de aproximadamente 24 horas por modelo entrenado, aumenta a 240 dado que se utiliza el método CV. En este esquema, se realizan pruebas con 1, 2, 3 y 4 capas. Se puede observar que los resultados con 2 capas superan levemente al resto.

5.4.3. Dropout recurrente

El tercer experimento corresponde a la evaluación del rendimiento para distintos valores de dropout recurrente en cada capa, los resultados para las métricas MAE, MSE y R^2 están disponibles en la Figura 5.13. Empleando 2 capas con 100 neuronas cada una, se prueban diferentes valores para dropout recurrente. En cuanto a las métricas MSE y R^2 se aprecia una convergencia en 0.4 con resultados que se mantienen hasta 0.7. Se opta por mantener un dropout recurrente de 0.4, ya que su aumento no reporta una mejora.

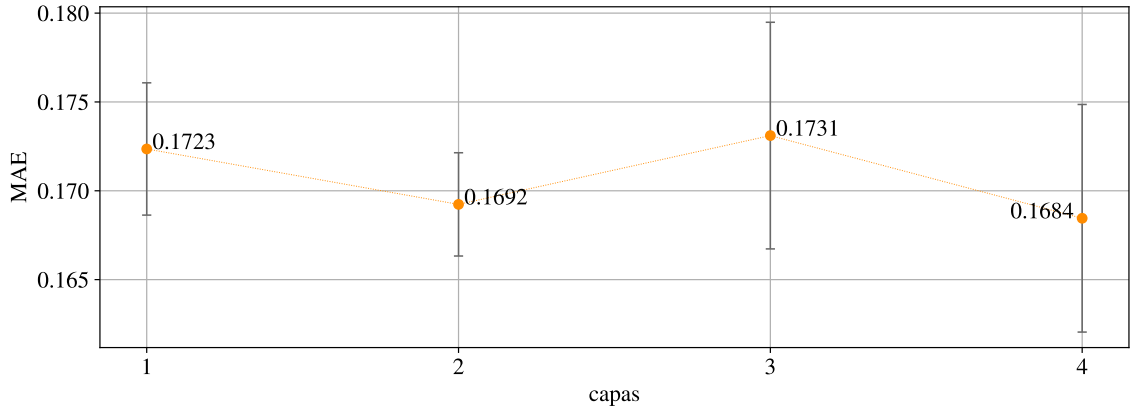
5.4.4. Dropout

El cuarto y último experimento corresponde a la evaluación del rendimiento para distintos valores de dropout regular en cada capa, los resultados para las métricas MAE, MSE y R^2 están disponibles en la Figura 5.14. Ya con 0.4 en dropout recurrente, el dropout convencional no genera mejoras en los resultados.

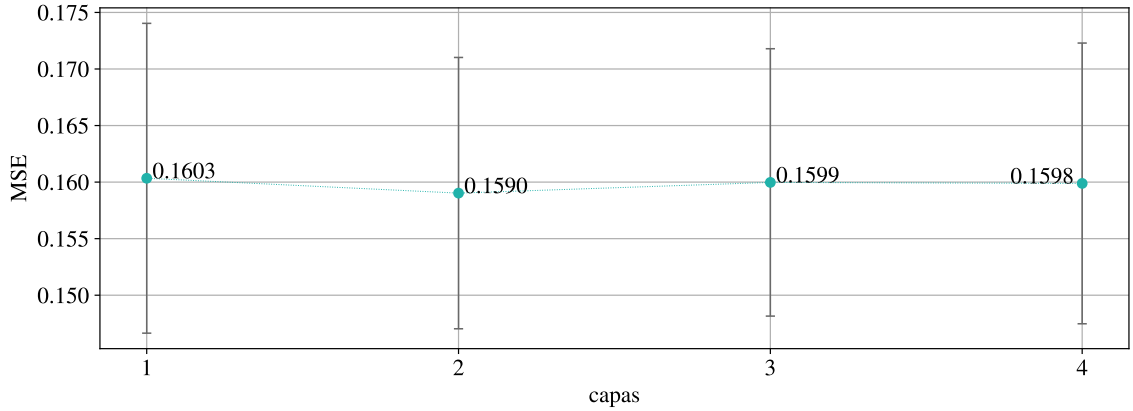
Para los experimentos de los hiperparámetros Dropout recurrente y Dropout regular se realiza una división en los datos por fold, generando un archivo de datos para cada fold. Esto permite cargar en memoria solo un fold para entrenamiento y evaluación. Este ajuste en el tratamiento de los datos fue necesario debido al alto uso de la memoria del servidor por otros usuarios y no por el alto consumo de memoria de los modelos.

5.5. Pruebas finales

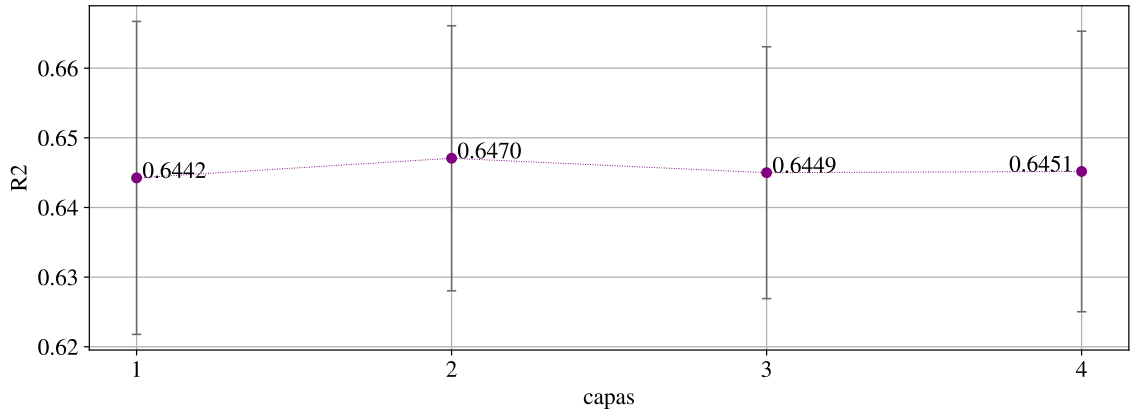
Del procesamiento de los datos se generó un conjunto de 732,870 instancias, de donde se reservaron 60,000 instancias para realizar la prueba final y el resto (672,870 instancias) se empleó para realizar los experimentos para el modelo FNN. Para la prueba final, un nuevo modelo FNN es entrenado empleando las 672,870 instancias y su capacidad puesta a prueba con las 60,000 instancias reservadas desde el inicio de los experimentos. Por otro lado, sobre el conjunto de datos de 732,870 instancias, se realiza un procesamiento para incluir nodos vecinos. De este tratamiento resultan 545,768 instancias, de las que 45,000 son reservadas para una prueba final y el resto de instancias (500,768) es empleada para realizar experimentos para los modelos CNN y RNN. Para la prueba final de los modelos



(a) Métrica MAE

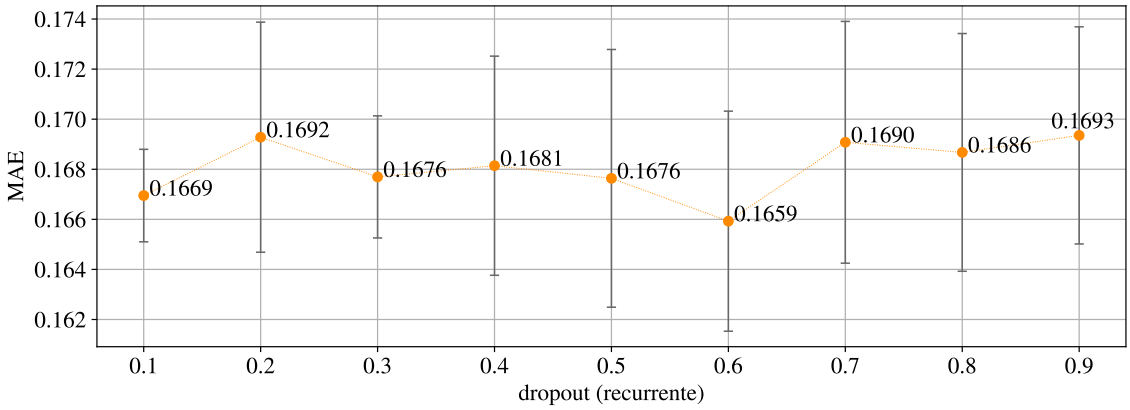


(b) Métrica MSE

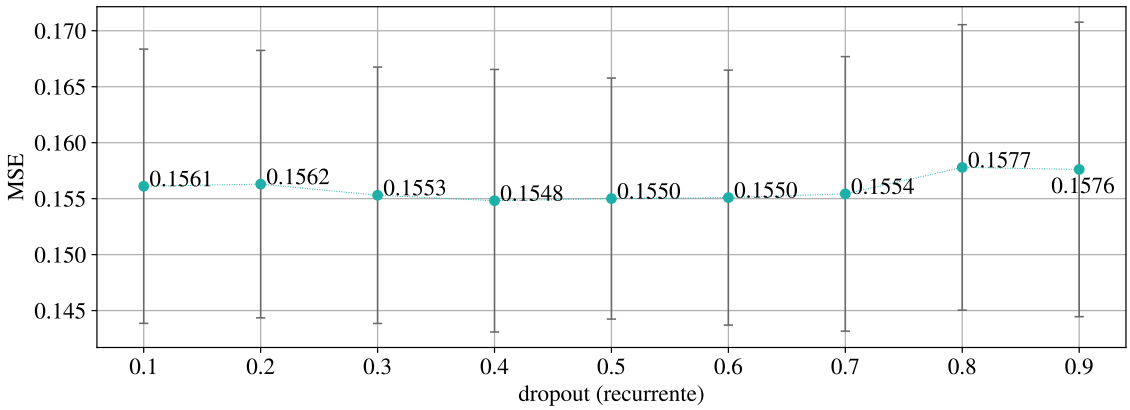


(c) Métrica R^2

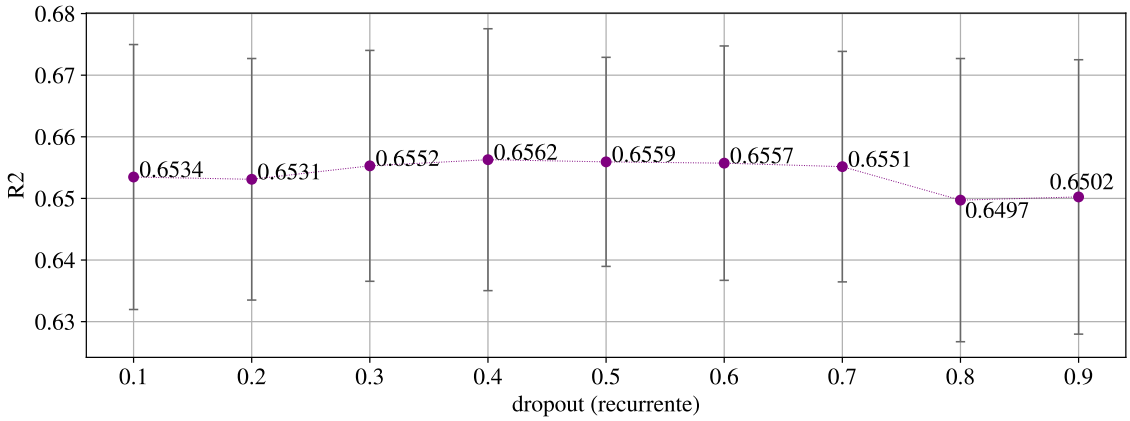
Figura 5.12: Gráficas de desempeño para diferente número de capas en la RNN



(a) Métrica MAE

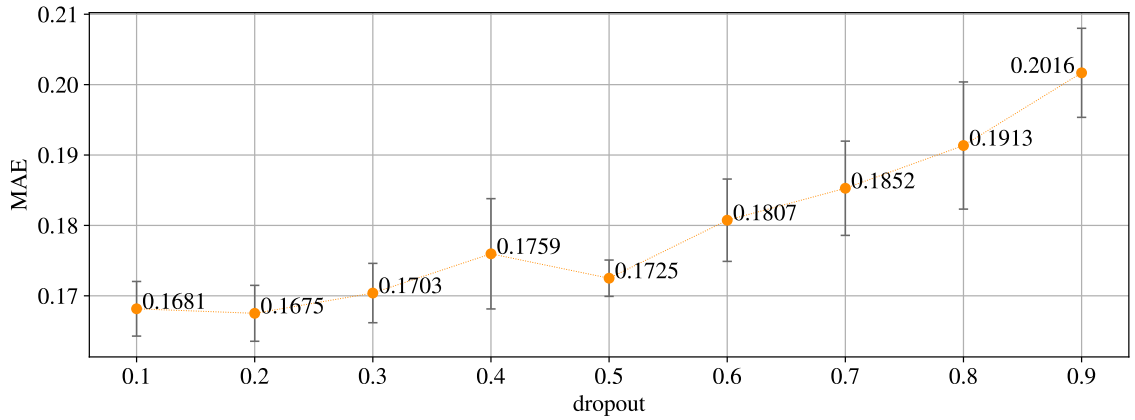


(b) Métrica MSE

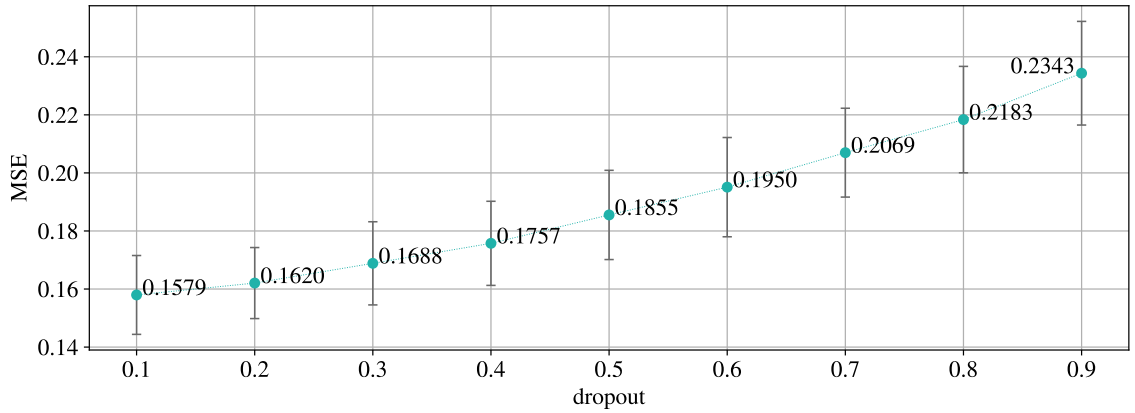


(c) Métrica R^2

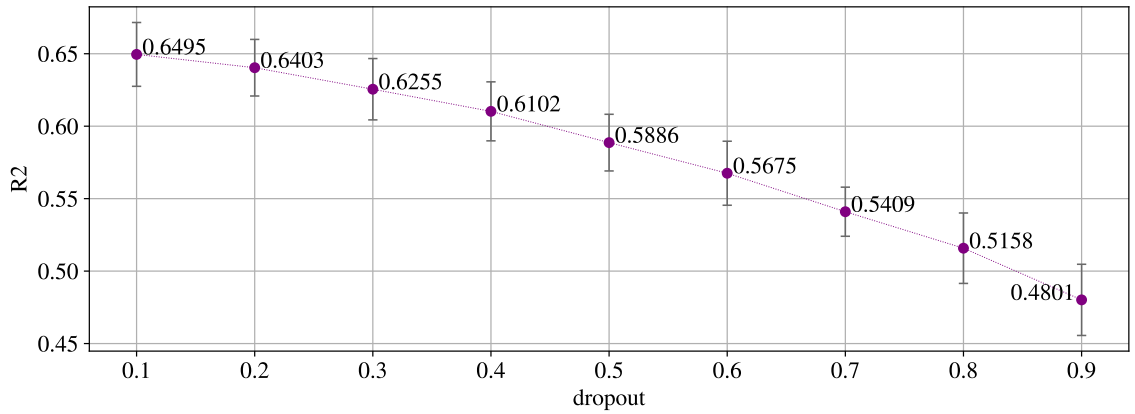
Figura 5.13: Gráficas de desempeño para diferente dropout recurrente en la RNN



(a) Métrica MAE



(b) Métrica MSE



(c) Métrica R^2

Figura 5.14: Gráficas de desempeño para diferente dropout en la RNN

CNN y RNN estos son entrenados sobre las 500,768 instancias y son evaluadas sobre las mismas 45,000 instancias. El desempeño de baseline para los conjuntos de prueba en las tres métricas empleadas está disponible en la Tabla 5.2.

Tabla 5.2: Desempeño de baseline en conjuntos de prueba

Instancias	MAE	MSE	R^2	Modelos probados
60,000	0.1960	0.2419	0.4504	FNN
45,000	0.2023	0.2379	0.4664	CNN y RNN

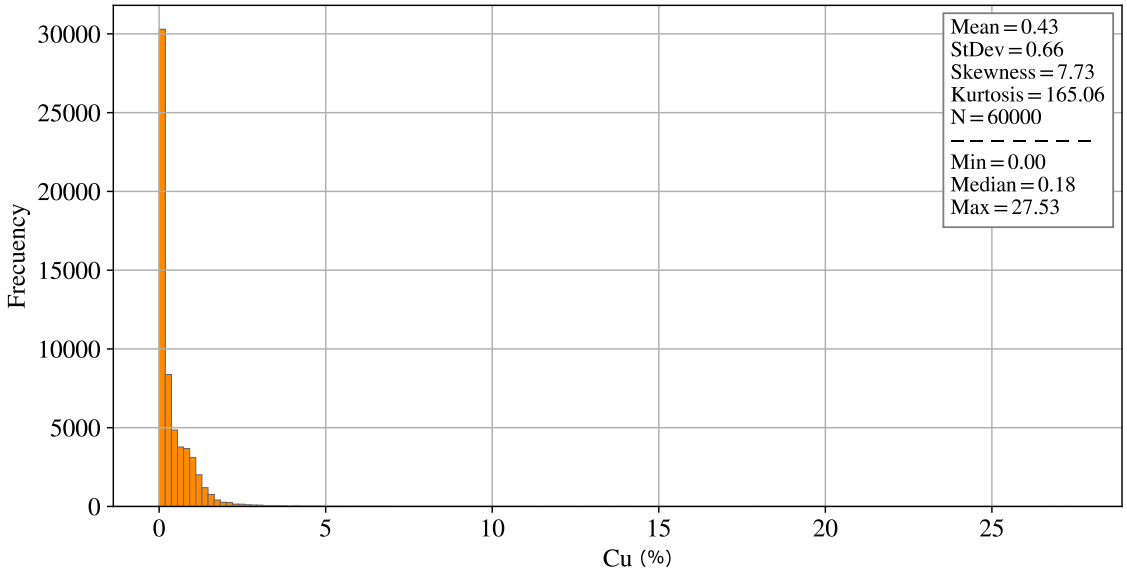
Para efectos de análisis del desempeño de los modelos, resulta directa la comparación entre los modelos RNN y CNN mediante las métricas MAE, MSE y R^2, ya que, comparten los mismos conjuntos de entrenamiento y prueba. En cambio, el modelo FNN es entrenado y puesto a prueba con otros conjuntos de datos. De manera preliminar, una buena comparación puede darse mediante el porcentaje de mejora en las métricas en comparación con el baseline del respectivo conjunto de datos de prueba. En las Figuras 5.15a y 5.15b, se presentan histogramas y un resumen de estadísticos de los conjuntos de prueba antes mencionados de 60,000 y 45,000 instancias. Se puede apreciar que ambos conjuntos de datos mantienen una alta similitud en los valores de cada estadístico. Ambos mantienen un sesgo positivo, un promedio que difiere en 0.01, una desviación estándar que difiere en 0.01, una mediana que difiere en 0.01, con el mismo valor mínimo y un valor máximo que difiere en 4.24. También es importante mencionar que ambos conjuntos mantienen similitud con la distribución y estadísticos presentados en la Figura 4.4.

La ubicación de cada nodo en el espacio tridimensional para los conjuntos de prueba es presentada en la Figura 5.16, un color más oscuro representa una menor altura (de acuerdo con el eje Z). En la Figura 5.16a se presenta el conjunto de prueba para el modelo FNN y en la Figura 5.16b se presenta el conjunto de prueba para los modelos CNN y RNN. En esta comparación visual, se puede apreciar que ambos conjuntos tienen una distribución espacial bastante similar y es apenas visible alguna diferencia dado que un conjunto es más pequeño.

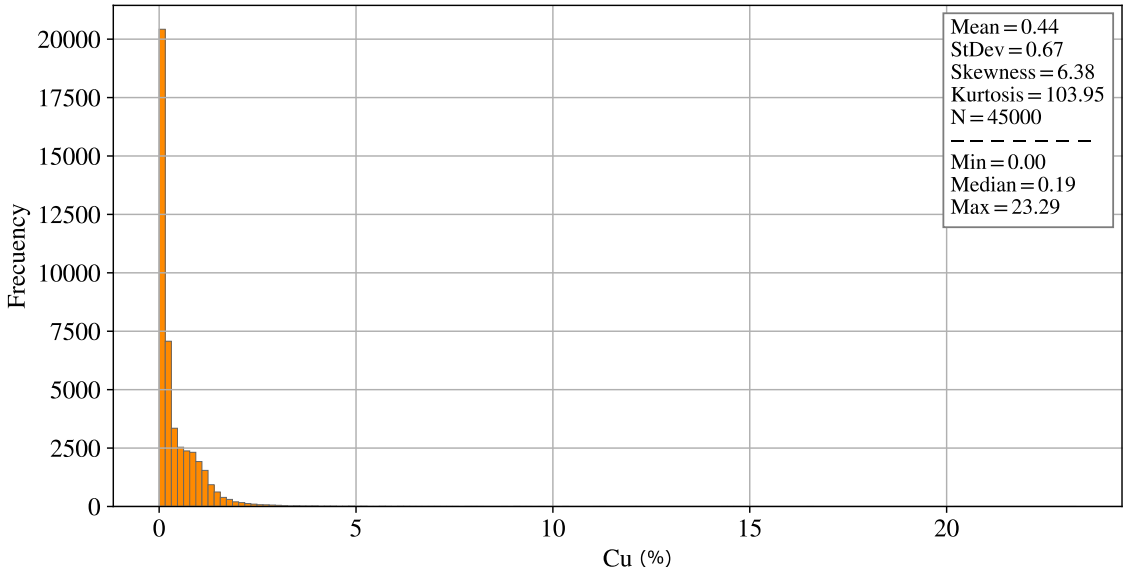
5.5.1. Prueba final FNN

Para la prueba final los hiperparámetros del modelo FNN son:

- Neuronas: 700
- Capas ocultas: 3
- Función de activación: sigmoid
- Dropout: 0.1
- Optimizador: Nadam

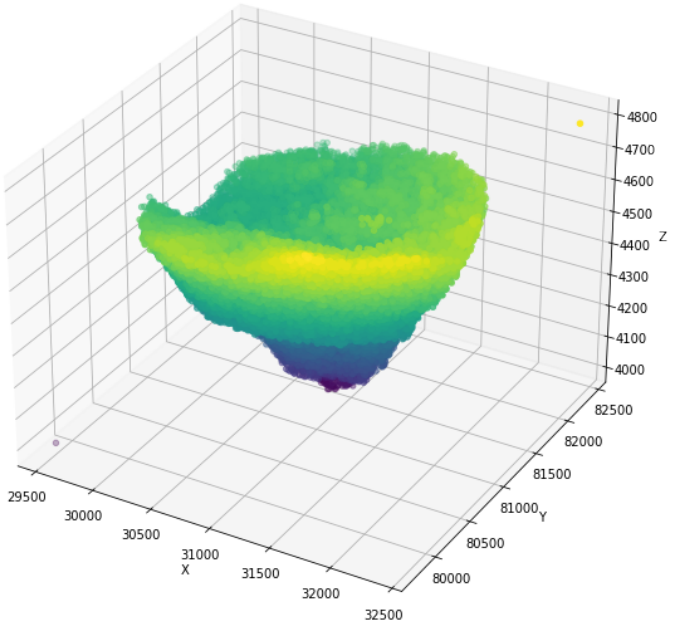


(a) Histograma de concentración de cobre de instancias de prueba

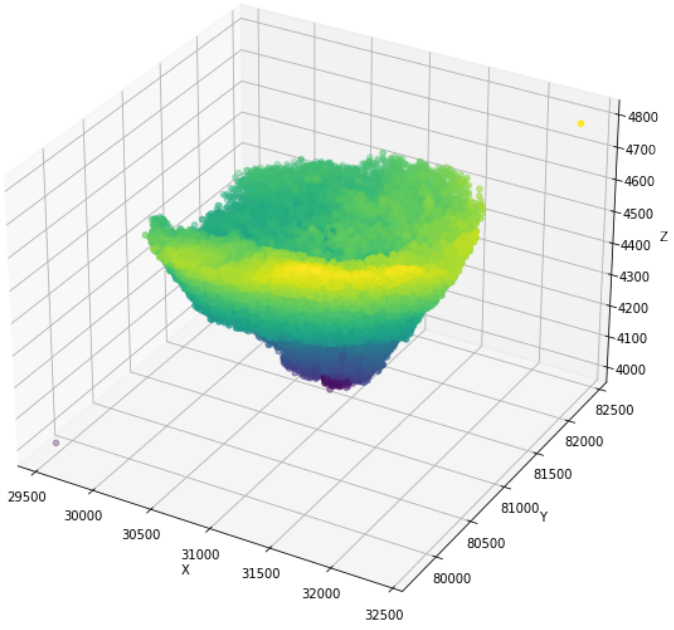


(b) Histograma de concentración de cobre de instancias de prueba para nodos con vecindad

Figura 5.15: Histogramas de concentración de cobre para conjunto de datos de prueba final



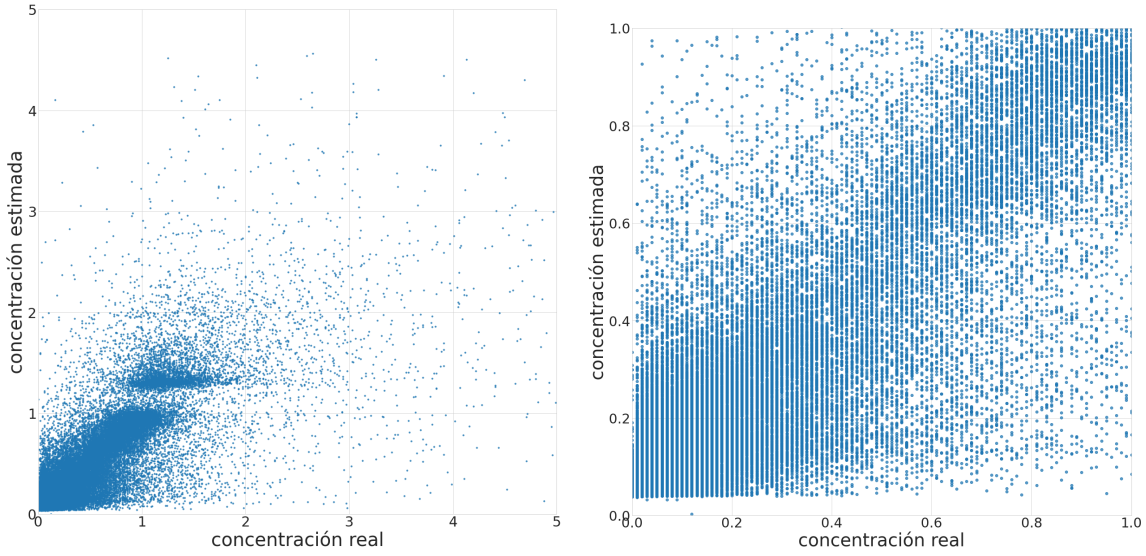
(a) Conjunto de prueba de 60,000 instancias



(b) Conjunto de datos de prueba de 45,000 instancias

Figura 5.16: Representación de ubicación espacial de nodos en conjuntos de prueba

Como se mencionó antes, se emplea el conjunto de datos de prueba de 60,000 instancias reservado desde el inicio de los experimentos. Para el conjunto de prueba el valor de las métricas para el baseline son 0.1960 para MAE, 0.2419 para MSE, 0.4504 para R^2 y 0.6816 para R. Mientras que para el modelo FNN el rendimiento es de 0.1763 para MAE, 0.1920 para MSE, 0.5638 para R^2 y 0.7574 para R. Un gráfico de tipo dispersión donde el eje horizontal considera el valor real y el eje vertical considera el valor estimado es presentado en la Figura 5.17, para comparar visualmente el desempeño. El entrenamiento y prueba requirió 1,827 segundos (30 minutos y 27 segundos) en el servidor de Posgrado de la Facultad. Para el entrenamiento se usaron 80 épocas, que corresponde al promedio de épocas límite en los que se conseguía una mejora en el entrenamiento en el experimento de optimizadores para el algoritmo Nadam. Las capas empleadas para la arquitectura que compone el modelo empleado en la prueba final y la forma de la salida que genera cada capa se presenta en la Tabla 5.3.



(a) Máxima concentración 5 (b) Máxima concentración 1

Figura 5.17: FNN: Gráficos de dispersión de prueba final

5.5.2. Prueba final CNN

Para la prueba final los valores para los hiperparámetros del modelo CNN son:

- Filtros: 512
- Tamaño kernel: 3
- Capas: 3
- Función de activación: Sigmoid

Tabla 5.3: FNN: Arquitectura empleada en prueba final

Capa	Forma de salida	Parámetros
Dense	(700)	30,800
Dropout	(700)	0
Dense	(700)	490,700
Dropout	(700)	0
Dense	(700)	490,700
Dense	(1)	701

La primera capa en la tabla es la capa de entrada que recibe un vector de forma (43). El total de parámetros a entrenar es 1,012,901

- Optimizador: Nadam

Se emplea un conjunto de datos de prueba de 45,000 instancias reservado desde el inicio de los experimentos. Para el conjunto de prueba el valor de las métricas para el baseline son 0.2023 para MAE, 0.2379 para MSE, 0.4664 para R^2 y 0.6920 para R. Mientras que para el modelo CNN el rendimiento es de 0.1670 para MAE, 0.1499 para MSE, 0.6637 para R^2 y 0.8126 para R. Un gráfico de tipo dispersión donde el eje horizontal considera el valor real y el eje vertical considera el valor estimado es presentado en la Figura 5.18, para comparar visualmente el desempeño. El entrenamiento y prueba requirió 20,239 segundos (5 horas, 37 minutos y 19 segundos) en el servidor de Posgrado de la Facultad. Para el entrenamiento se usaron 64 épocas, que corresponde al promedio de épocas límite en los que se conseguía una mejora en el entrenamiento en el experimento de optimizadores para el algoritmo Nadam. Las capas empleadas para la arquitectura que compone el modelo empleado en la prueba final y la forma de la salida que genera cada capa se presenta en la Tabla 5.4.

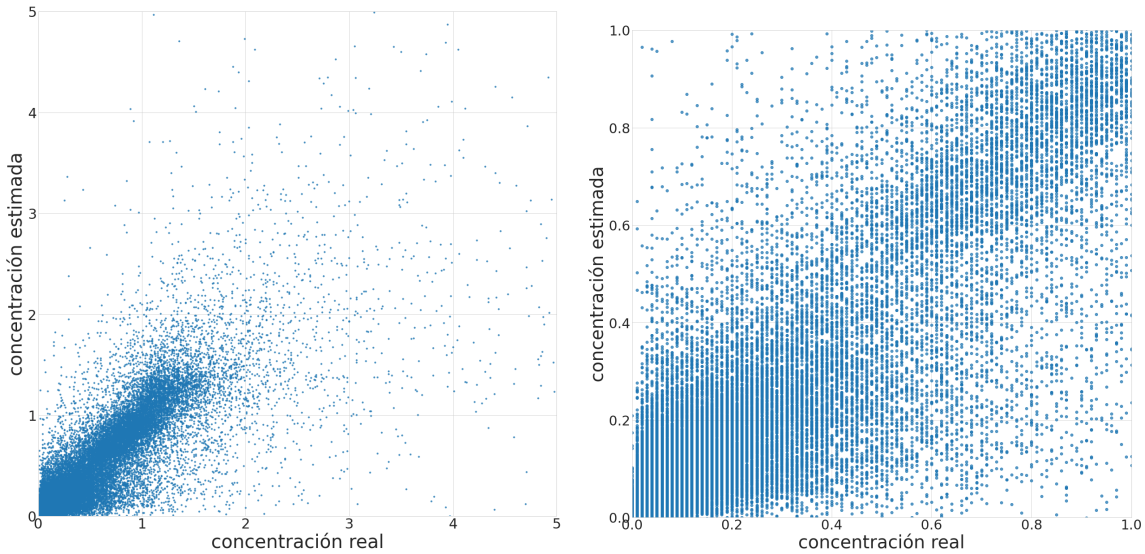
Tabla 5.4: CNN: Arquitectura empleada en prueba final

Capa	Forma de salida	Parámetros
Conv1D	(17, 512)	68,096
Conv1D	(15, 512)	786,944
Conv1D	(13, 512)	786,944
Flatten	(6,656)	0
Dense	(1)	6,657

La primera capa en la tabla es la capa de entrada que recibe un vector de forma (19,44). El total de parámetros a entrenar es 1,648,641

5.5.3. Prueba final RNN

El optimizador que se emplea en la prueba final es Nadam método que obtuvo los mejores resultados en los dos modelos anteriores (FNN y CNN). Las funciones de activación



(a) Máxima concentración 5

(b) Máxima concentración 1

Figura 5.18: CNN: Gráficos de dispersión de prueba final

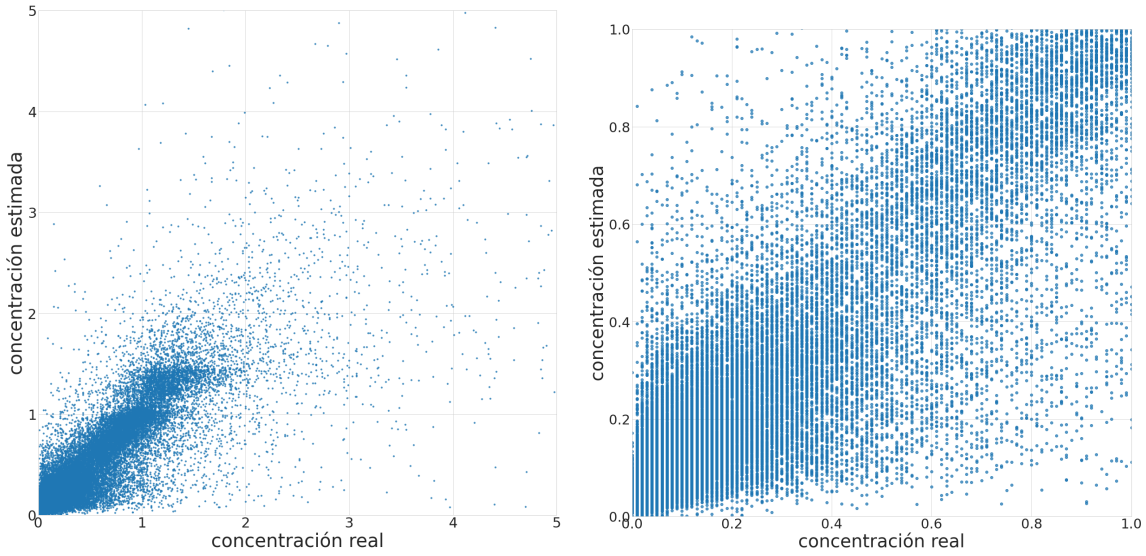
empleadas corresponden a las definidas por defecto en la librería Keras. Estas funciones de activación son ampliamente empleadas y de acuerdo con investigaciones empíricas son las que mejor resultados producen en capas LSTM. Como ejemplo, (Breuel, 2015), presenta un informe en donde se comparan diferentes valores para los hiperparámetros en una LSTM y concluye que las funciones de activación tanh y sigmoid tienen un mejor desempeño, lo que se determina para dos conjuntos de datos de números y caracteres dibujados a mano (MINST y UW3, respectivamente). También, (Jozefowicz et al., 2015) presenta una investigación en donde se implementan cerca de 10,000 arquitecturas RNN mediante algoritmos genéticos, donde se consideran las funciones de activación tanh(), sigmoid(), ReLu() y variaciones de Linear(). Las arquitecturas son puestas a prueba con tres conjuntos de datos, el primero de operaciones aritméticas, el segundo predicción de caracteres para la generación de archivos XML y una tarea de modelado de lenguaje a nivel de palabras. Las tres mejores arquitecturas resultan ser LSTM, con las funciones de activación tanh y sigmoid.

Entonces para la prueba final los valores para los hiperparámetros del modelo RNN son:

- Neuronas: 500
- Capas ocultas: 2
- Dropout: 0.0
- Dropout recurrente: 0.4
- Activación: tanh

- Activación recurrente: sigmoid
- Optimizador: Nadam

Se emplea un conjunto de datos de prueba de 45000 instancias reservado desde el inicio de los experimentos. Para el conjunto de prueba el valor de las métricas para el baseline son 0.2023 para MAE, 0.2379 para MSE, 0.4664 para R^2 y 0.6920 para R. Mientras que para el modelo RNN el rendimiento es de 0.1591 para MAE, 0.1442 para MSE, 0.6765 para R^2 y 0.8221 para R. Un gráfico de tipo dispersión donde el eje horizontal considera el valor real y el eje vertical considera el valor estimado es presentado en la Figura 5.19, para comparar visualmente el desempeño. El entrenamiento y prueba requirió 48,544 segundos (13 horas, 29 minutos y 4 segundos) en el servidor de Posgrado de la Facultad. Para el entrenamiento se usaron 55 épocas, que corresponde al promedio de épocas límite en los que se conseguía una mejora en el entrenamiento en el experimento de dropout con valor 0.4. Las capas empleadas para la arquitectura que compone el modelo empleado en la prueba final y la forma de la salida que genera cada capa se presenta en la Tabla 5.5.



(a) Máxima concentración 5 (b) Máxima concentración 1

Figura 5.19: CNN: Gráficos de dispersión de prueba final

5.5.4. Relevancia estadística

El desempeño de los modelos DL en el proceso de entrenamiento y prueba se mide al comparar cada predicción (y_{DL}) con el valor esperado (y_{test}). De la misma manera, el desempeño del método geoestadístico cuyas estimaciones son parte de LTMP (y_{LTMP}), se puede medir al compararlas con los datos del STMP (y_{test}).

Tabla 5.5: RNN: Arquitectura empleada en prueba final

Capa	Forma de salida	Parámetros
LSTM	(19, 500)	1,090,000
LSTM	(500)	2,002,000
Dense	(1)	501

La primera capa en la tabla es la capa de entrada que recibe un vector de forma (19,44). El total de parámetros a entrenar es 3,092,501.

Para determinar si existe una mejora entre los modelos DL y LTMP, se debe calcular el error de cada modelo. Al comparar el error que se genera, es posible determinar cuál modelo tiene mayor precisión. El error de ambos modelos para cada instancia puede ser definido por el residual absoluto. En el caso de los modelos DL corresponderá a $residual_{DL} = |y_{test} - y_{DL}|$ y las estimaciones en el LTMP será $residual_{LTPM} = |y_{test} - y_{LTPM}|$.

El test de Wilcoxon (Wilcoxon, 1945) es una prueba que permite probar la relevancia estadística de la diferencia entre dos conjuntos de datos. En este caso, la diferencia está dada por $residual_{LTPM} - residual_{DL}$, cuyo resultado tiene tres alternativas, primero, puede ser igual a cero, lo que implicaría que ambos modelos tienen el mismo error, segundo, el resultado es positivo, esto implica que es mayor el error en las estimaciones del LTMP y tercero, el resultado es negativo, lo que implica que es mayor el error en las estimaciones del modelo DL.

Para llevar a cabo la implementación del test Wilcoxon se utilizó una función de la librería Scipy[1]. Dicha función permite realizar una prueba bilateral (two-sided test), donde la hipótesis nula dice que la mediana de las diferencias es cero frente a la alternativa de que es diferente de cero y una prueba unilateral (one-sided test), donde la hipótesis nula establece que la mediana es negativa frente a la alternativa de que es positiva. En otras palabras, la prueba bilateral plantea la hipótesis nula de que no hay diferencia significativa frente a la alternativa de que si hay, mientras que la prueba unilateral plantea como hipótesis nula que predominan valores negativos frente a la alternativa de que predominan valores positivos.

Para cada modelo considerado en las pruebas finales antes descritas, se lleva a cabo la prueba bilateral y luego la prueba unilateral, el resumen de los resultados se encuentra en la Tabla 5.6. Como se puede observar, se dispone de una columna con el valor P de cada prueba, para cada modelo. Todos los valores P cumplen con nivel de significancia = 5 % (0.05), es decir cada valor cumple con $P < 0.05$, lo que permite rechazar cada hipótesis nula. Por lo tanto, para cada modelo, existe una diferencia significativa y además, predominan valores positivos, por lo que existe una disminución del error estadísticamente. significativa.

[1] https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.wilcoxon.html

Tabla 5.6: Resumen test estadístico

	Hipótesis nula	*P*-value	Conclusión
FNN	No hay diferencia	1.943e-84	Si hay diferencia
	Hay mas valores negativos	9.716e-85	Hay mas positivos
1D CNN	No hay diferencia	4.385e-251	Si hay diferencia
	Hay mas valores negativos	2.193e-251	Hay mas positivos
LSTM	No hay diferencia	0.0	Si hay diferencia
	Hay mas valores negativos	0.0	Hay mas positivos

5.6. Discusión

La metodología empleada muestra que los modelos de FNN, CNN y RNN efectivamente mejoran el desempeño de las estimaciones de concentración de cobre. Para el preprocesamiento de los datos se realiza una selección de variables, descartando variables que aportan información de la organización y el orden de explotación mineral y manteniendo variables de propiedades geológicas de la roca. Se contempla tratamiento de valores ausentes los que se encontraron en ambos modelos, STMP y LTMP, lo que reduce de manera drástica el conjunto de datos original de 49,295,441 instancias a 732,870. Se aplica un tratamiento de variables cualitativas lo que aumenta el tamaño del vector de entrada de 25 a 43 y se realiza una normalización de variables cuantitativas, siendo las variables de coordenadas normalizadas de forma simétrica. Además, se propone la inclusión de nodos vecinos que aporten mayor información en la estimación de concentración para un nodo central, generando un conjunto de datos de 545,768 instancias, esto permite la implementación de modelos que requieren como entrada grupos de datos, como los modelos CNN y RNN.

5.6.1. Sobre optimización de hiperparámetros

Para la evaluación de FNN se consideran la cantidad de neuronas, la cantidad de capas, función de activación en las capas ocultas, la inclusión de capas con dropout y diferentes optimizadores para el ajuste de los pesos. En el experimento con diferentes cantidades de neuronas se establece que el incremento en la cantidad de neuronas produce una mejora en el rendimiento. Sin embargo, la evolución en la mejora del rendimiento converge en cierto punto. En los resultados para diferente cantidad de capas ocultas, se logra mejorar el desempeño, pero pasado cierta cantidad de capas el desempeño comienza a empeorar. En el experimento de funciones de activación, ReLu obtiene buenos resultados, pero se ve levemente superada por el desempeño de la función de activación sigmoid. Es interesante observar que la función de activación con peor desempeño es la lineal. Hay que recordar que el uso de una función de activación no lineal permite al modelo capturar relaciones no lineales. El hecho de que una función de activación lineal obtenga los peores resultados,

brinda indicios de que efectivamente podrían existir relaciones no lineales y, por lo tanto, los modelos de DL son adecuados para resolver el problema propuesto en esta tesis. Por otro lado, la adición de dropout en toda la arquitectura no genera grandes cambios en el desempeño, en los experimentos el valor de Dropout se aumenta de forma gradual y se logra una mejora al considerar un 10 %, pero el aumento en este porcentaje se traduce rápidamente en un declive en el desempeño. Para el análisis de optimizadores se obtuvieron los resultados más consistentes entre métricas, donde Nadam obtuvo el mejor resultado. En general se observa consistencia en los resultados obtenidos por las métricas MSE y R^2, pero hay discrepancias en los resultados de la métrica MAE.

Para la evaluación del modelo CNN se consideraron la cantidad de filtros, tamaño de kernel, capas, función de activación en capas ocultas y optimizador. Los resultados de la cantidad de filtros tienen una evolución similar a la obtenida por cantidad de neuronas para el modelo FNN. Al aumentar la cantidad de filtros por capa, luego de cierto punto el desempeño comienza a converger. La evaluación del tamaño de kernel resultó de inmediato en un kernel pequeño con mejores resultados, esto indica que una ventana más pequeña, es decir, menos instancias sobre las que aplicar el producto punto, genera mejores resultados. En el experimento de capas se observa el mismo comportamiento que en el experimento con el modelo FNN, donde el desempeño aumenta progresivamente hasta la capa número tres y en la capa cuatro comienza a empeorar. Al igual que en el experimento de funciones de activación para el modelo FNN las funciones de activación Relu y sigmoid obtienen mejores resultados, siendo esta última la mejor de todas las funciones. Respecto a los optimizadores, el mejor resulta ser Nadam, al igual que en FNN.

Para la evaluación de RNN se considera la cantidad de neuronas por capa oculta, la cantidad de capas ocultas, el dropout recurrente, la cantidad de dropout regular, la función de activación regular y recurrente y el optimizador. Los tres últimos hiperparámetros no fueron evaluados mediante experimentos. En la evaluación de cantidad de neuronas, a más neuronas mejor desempeño, pero también más tiempo de cómputo. En los experimentos se llega a experimentar con 500 neuronas y al observar el gráfico de resultados, se observa que los resultados no convergen, por lo que se puede pronosticar que un aumento en la cantidad de neuronas aún podría mejorar el desempeño, sin embargo, se desiste debido al tiempo excesivo de cómputo. Respecto a la cantidad de capas, el aumento de estas no presenta mejores resultados, situación que también ocurre con el modelo FNN. De los experimentos con dropout recurrente se obtiene una mejora leve y con dropout regular no se obtiene ninguna mejora.

Para definir el cambio producido en el proceso de optimización de hiperparámetros, se puede comparar el mejor desempeño del primer experimento y el mejor desempeño del último experimento en términos porcentuales. Para el modelo FNN se logra una mejora en las métricas MAE, MSE y R^2 en un 3.9 %, 5.4 % y 4.2 %, esto se tradujo además, en estimaciones más estables, ya que la desviación estándar disminuyó 33.9 % en promedio. Para el modelo CNN se logra una mejora en las métricas MAE, MSE y R^2 en un 5.3 %, 7.2 %, 4.5 %, esto se tradujo además, en estimaciones más estables, ya que la desviación estándar disminuyó 14.6 % en promedio. Para el modelo RNN se realizan las comparaciones considerando 100 neuronas en vez de 500, en estos términos se logra una mejora en las

métricas MAE, MSE y R^2 en un 0.6 %, 3.4 % y 1.9 %, esto se tradujo en estimaciones menos estables, ya que la desviación estándar aumentó 13.8 % en promedio.

Cada experimento es consecutivo y en un orden arbitrario. El mejor valor para un hiperparámetro es empleado en el siguiente experimento. Al considerar esto, se puede establecer que los valores para los hiperparámetros que resultan de la experimentación, son consecuencia del orden en que se evaluaron los hiperparámetros. Por ejemplo, en el caso del modelo FNN, primero se evalúa el número de neuronas, donde resulta que 700 neuronas generan un mejor desempeño, lo que condiciona los resultados del experimento para el número de capas. Así también tiene efecto en los experimentos para los hiperparámetros posteriores.

5.6.2. Sobre pruebas finales

Una vez se establecen los mejores valores para los hiperparámetros considerados para cada modelo, se ponen a prueba en conjuntos de datos apartados para este propósito, donde se busca validar los hiperparámetros y comprobar la capacidad de generalización de los modelos. El desempeño que tiene cada modelo en su respectivo conjunto de prueba, el cambio porcentual entre el baseline de cada conjunto de prueba y el respectivo modelo, y el tiempo de cómputo (en segundos) está disponible en la Tabla 5.7. Se puede establecer que los tres modelos tienen la capacidad de reducir el error. Los resultados indican que los modelos se ajustan a los datos y son capaces de capturar las relaciones lineales, no lineales y espaciales. En términos porcentuales, el modelo RNN alcanza la mayor mejora y se puede establecer con claridad su superioridad respecto al modelo CNN. La comparación respecto al modelo FNN no es tan directa dado que no se entrenan y prueban con los mismos conjuntos, pero en principio también se puede concluir que RNN le supera, dado que logra un porcentaje de mejora superior. Además, se puede notar que para el proceso de entrenamiento y prueba, el modelo CNN requirió 20,239 segundos en 64 iteraciones (aproximadamente 316 segundos por iteración), mientras que el modelo RNN requirió 48,544 segundos en 55 iteraciones (aproximadamente 882.6 segundos por iteración), 2.4 veces más de tiempo.

Tabla 5.7: Desempeño de modelos en conjuntos de prueba

	MAE (% cambio)	MSE (% cambio)	R^2 (% cambio)	R (% cambio)	segundos
baseline (60,000)	0.1960	0.2419	0.4504	0.6816	-
FNN	0.1763 (-10.05)	0.192 (-20.63)	0.5638 (25.18)	0.7574 (11.12)	1,827
baseline (45,000)	0.2023	0.2379	0.4664	0.6920	-
CNN	0.1670 (-17.45)	0.1499 (-36.99)	0.6637 (42.30)	0.8126 (17.42)	20,239
RNN	**0.1591** (-21.35)	**0.1442** (-39.39)	**0.6765** (45.05)	0.8221 (18.80)	48,544

Es también importante considerar el cambio que se produce entre el mejor desempeño obtenido en la optimización de hiperparámetros y el desempeño obtenido en las pruebas, que para mayor claridad se puede establecer en términos porcentuales. Para el modelo

FNN se logra una mejora en las métricas MAE y R^2 en un 0.7 % y 3.7 %, pero un deterioro de un 10.3 % en la métrica MSE. Para el modelo CNN se logra una mejora en las métricas MAE y MSE en un 4.4 % y 8.2 %, pero un deterioro de un 4.1 % en la métrica R^2. Para el modelo RNN se logra una mejora en la métrica MSE de un 3.0 %, pero un deterioro en las métricas MAE y R^2 de un 1.4 % y 1.0 %. En general no se producen grandes cambios en el desempeño de los modelos, lo que permite validar los hiperparámetros seleccionados.

Para los modelos CNN y RNN se emplea información de un nodo central, para el que se estima la concentración mineral y además, se emplean los atributos de nodos vecinos que se encuentran dentro de un radio. Esta configuración en los datos implica la suposición de que existen relaciones entre la concentración de un nodo y los atributos de nodos cercanos. Suposición similar a la que realizan métodos geoestadísticos como Ordinary Kriging. Sin embargo, ambos casos se distinguen porque para los modelos CNN y RNN se emplea información no solo de la concentración, sino también de atributos en general, lo que brinda información del contexto en que se encuentra un nodo. La mejora producida por los modelos CNN y RNN es superior a la mejora producida por el modelo FNN lo que se puede atribuir a la incorporación de información del contexto de los nodos.

Si consideramos el funcionamiento del modelo RNN en su forma LSTM este tiene la capacidad de mantener una memoria interna, un estado, que se transfiere y transforma cuando se realiza el procesamiento de cada instancia individual. Esta forma de procesar los datos difiere respecto al método CNN de 1 dimensión, donde un grupo de instancias se somete a una operación que "sintetiza" los datos extrayendo diferentes características, generando una especie de resumen desde distintas perspectivas. Dicho de otra manera, en un grupo de instancias, el método LSTM realiza un análisis individual y el método CNN 1D un análisis global. Esta diferencia en el procesamiento de los datos puede ser la causa de la diferencia en el rendimiento.

Capítulo 6

Conclusiones y Trabajos Futuros

En esta tesis se busca establecer el alcance y aporte de modelos de DL en la reducción de error en la estimación de concentración de minerales, para lo cual se contó con un conjunto de datos reales procedentes de una mina de cielo abierto en Chile, que posee concentraciones de mineral de Cobre tipo Sulfuro y Óxido. Para lograrlo se propusieron tres objetivos que se resumen en (1) la selección de modelos DL, (2) su implementación y (3) su evaluación. Para el uso de los datos disponibles fue necesario realizar un preprocesamiento que incluyó la exclusión de variables que no aportan información de las propiedades de concentración mineral, la exclusión de instancias con valores ausentes, el tratamiento de variables categóricas ordinales y categóricas nominales y la normalización de las variables empleando promedio y desviación estándar. Para las variables de coordenadas espaciales se realizó una normalización simétrica para no distorsionar la unidad de medida y las proporciones en las dimensiones, el resto de las variables numéricas se normaliza de forma individual. Además, se trabajaron los datos para la generación de un nuevo conjunto de datos donde cada nodo dispusiera de información de nodos vecinos dentro de un radio determinado.

El objetivo específico 1 (OE1) señala: "Seleccionar modelos basados en Deep Learning que permitan realizar estimaciones de concentración de recursos minerales". Su completitud se verifica mediante los resultados del estudio y análisis de la literatura, presentados principalmente en la Sección 3, del estado del arte y la Sección 2, de conceptos preliminares. Construir el estado del arte requirió la revisión de diferentes trabajos de revistas científicas y permitió identificar modelos basados en Deep Learning en la estimación de concentración de recursos minerales (Feedforward Neural Network y Recurrent Neural Network). Mientras que la definición y descripción de los modelos requirió el estudio de otras fuentes, como libros del área. Esto extendió los algoritmos identificados y permitió seleccionar modelos con características similares a los propuestas en el estado del arte (Long Short-Term Memory Recurrent Neural Network y 1D Convolutional Neural Network).

La selección de modelos basados en Deep Learning se realizó mediante el estudio y revisión de literatura. En primera instancia no se encontraron trabajos que resolvieran la misma problemática, por lo que se consideraron trabajos que abordan una problemática similar que corresponde a la estimación de recursos minerales en base a muestras puntuales.

Se revisaron en detalle 29 trabajos que emplean algún modelo ML. Solo un trabajo (Singh et al., 2018), emplea algún modelo DL. El resto de las implementaciones de redes neuronales consisten en arquitecturas de una capa y un número reducido de neuronas, lo que se justifica con el tamaño reducido de los conjuntos de datos. Sin embargo, en general los trabajos no comparten el esquema de pruebas que realizaron para diseñar las arquitecturas, lo que cobra relevancia al momento de considerar las conclusiones de algunos de los trabajos, que afirman que su método supera a una red neuronal. Hay que mencionar que ninguno de los trabajos implementa alguna forma de CNN, método que es considerado en esta tesis en base a (Kiranyaz et al., 2020), (Chollet, 2017).

En general, de la revisión de literatura se puede concluir que (1) son escasos los trabajos que evalúan modelos DL. Esto es consistente con el tamaño reducido de los conjuntos de datos con los que se cuenta para la estimación de concentración de mineral en base a muestras. Sin embargo, con la problemática que se presenta en esta tesis, se cuenta con un buen volumen de datos de la extracción (STMP), por lo que la cantidad de datos disponibles incrementa de forma considerable, generando el escenario adecuado para la implementación de modelos DL y la explotación de sus capacidades. (2) En los trabajos no se encontró un framework o esquema, que facilite la comparación de los resultados de distintos trabajos y que permita llevar un seguimiento de los avances en el área. Es necesario establecer etapas claras en el preprocesamiento de datos y la validez de las técnicas, definir métricas para analizar los datos y facilitar la relación de sus propiedades con el potencial de diferentes modelos. Finalmente este esquema permitiría establecer con más claridad la forma adecuada en que métodos geoestadísticos pueden interactuar con modelos DL.

El objetivo específico 2 (OE2) señala: "Implementar modelos seleccionados, considerando posibles adaptaciones y modificaciones". Su completitud se verifica mediante los resultados de la preparación de los datos, presentados principalmente en la Sección 4 y los resultados del diseño e implementación de la metodología y los experimentos presentados principalmente en la Sección 5. Llevar a cabo la implementación de los modelos, requirió de la correcta preparación y formato de los datos para hacerlos compatibles con los modelos. Por otro lado, se debió llevar a cabo una metodología adecuada, que permitiera la realización de experimentos con el propósito de definir valores para los hiperparámetros de los diferentes modelos, considerando aspectos como sobrentrenamiento, generalización, validación, métodos de regularización, etc.

La implementación de los modelos Feedforward Neural Network, Convolutional Neural Network y Recurrent Neural Network se realiza empleando la biblioteca de código Keras, que dispone de APIs para facilitar la construcción de arquitecturas y la modificación de sus hiperparámetros. Para lograr la implementación de modelos DL se aplican técnicas de preprocesamiento a un conjunto original de 49,295,441 de instancias, del que resultan 732,870 instancias con información de los modelos LTMP (considerado como vector de entrada) y STMP (que contiene concentración mineral empleada como etiqueta). Este conjunto es suficiente para la experimentación con el modelo FNN. Para la implementación de los modelos CNN y RNN se preparan los datos para disponer de grupos de datos como entrada para los modelos. Estos grupos consisten en un nodo central y nodos vecinos dentro de un radio determinado. De este modo, los atributos de todos los nodos vecinos se emplean

como información adicional para la estimación de concentración mineral del nodo central. Solo 545,768 nodos de los 732,870 disponen de nodos vecinos en la misma configuración. Las técnicas para el tratamiento de los datos como la normalización simétrica o el uso de nodos vecinos están basadas en técnicas empleadas en trabajos seleccionados de la revisión de la literatura, lo que se menciona en las secciones 3.3 y 4.

El objetivo específico 3 (OE3) señala: "Evaluar resultados de los modelos implementados mediante métricas de error y pruebas estadísticas". Su completitud se verifica mediante los resultados presentados en la Sección 5, donde se señala el desempeño de los modelos mediante las métricas MAE, MSE y R2. En particular, en la Sección 5.5, se presentan resultados de las pruebas finales, donde se considera también la métrica R y se presentan diagramas de dispersión de las estimaciones versus valores esperados. En la Sección 5.5.4 se presentan los resultados de la prueba de relevancia estadística Wilcoxon. Finalmente, en las secciones 5.6 y 6 se presentan análisis del desempeño de los modelos.

La evaluación de los modelos se logra en dos etapas: (1) Optimización de hiperparámetros mediante CV y (2) prueba final. Para la prueba final se reserva una porción de datos, empleada solo para este propósito. El resto de los datos es empleado para entrenamiento, pero antes de la prueba final, es empleado en la optimización de hiperparámetros. En la primera etapa para cada modelo se realizan experimentos solo para algunos de sus hiperparámetros. Para cada hiperparámetro se consideran diferentes valores y para cada valor se entrena un modelo y se evalúan mediante 10-fold CV. Para determinar si efectivamente se logra una reducción en el error de las estimaciones de concentración se utiliza como baseline la estimación generada por un método geoestadístico (Ordinary Kriging) disponible en el LTMP, el que se compara con las estimaciones en el STMP, mediante las métricas de desempeño. De la experimentación y análisis de hiperparámetros se obtiene una configuración final de hiperparámetros para cada modelo. En la prueba final se valida esta configuración y la capacidad de generalización de los modelos. Esto dado que la prueba final se realiza sobre conjuntos de datos distintos del que se emplea para el entrenamiento y optimización de hiperparámetros.

De los resultados obtenidos en las pruebas finales se puede establecer que todos los modelos evaluados tienen la capacidad de reducir el error en la estimación de concentración de cobre. De acuerdo con las pruebas, el modelo RNN, en su variante Long Short-Term Memory, genera mayores mejoras en cuanto al baseline, produciendo una mejora del 21.35 %, 39.39 % y 45.05 % en las métricas MAE, MSE y R^2, respectivamente. Le siguen los modelos Convolutional Neural Network 1D con una mejora de 17.45 %, 36.99 % y 42.30 %.y Feedforward Neural Network que produce una mejora del 10.05 %, 20.63 % y 25.18 % en las métricas MAE, MSE y R^2.

Los resultados presentados en esta tesis permiten validar la hipótesis que señala que: "El uso de modelos basados en Deep Learning permite mejorar resultados de métodos tradicionales en el problema de estimación de concentración de recursos minerales". Esto implica que estimaciones generadas por métodos geoestadísticos en combinación con modelos DL pueden obtener mejores estimaciones como lo muestra, por ejemplo, el uso del modelo RNN con una mejora del 45 % en la métrica R^2. De los resultados destacan los modelos CNN y RNN que para la estimación de concentración en un nodo consideran los

atributos de nodos vecinos como información de contexto. De acuerdo con la revisión de literatura realizada, el modelo CNN de una dimensión (1D) no habían sido empleado para la estimación de concentración mineral y en el caso del modelo RNN no había sido empleado en su versión LSTM. Este último modelo obtiene los mejores resultados superando el desempeño del modelo CNN 1D con una diferencia de 0.01 aproximadamente en cada métrica, pero requiere un mayor tiempo de cómputo, que en la prueba final resultó en 2.4 veces más tiempo.

6.1. Trabajos futuros

Para la evaluación se consideró un espectro limitado de los hiperparámetros que pueden ser alterados en las capas empleadas para la construcción de las arquitecturas en cada modelo, esto debido a tiempos excesivos de cómputo. Por lo que un trabajo futuro a considerar consistiría en ampliar los experimentos, considerando más hiperparámetros y valores en cada uno de los modelos.

Las métricas empleadas para registrar el desempeño de los modelos DL, en los diferentes experimentos llevados a cabo, permiten realizar una comparación para diferentes valores en los hiperparámetros. Sin embargo, las métricas no brindan información del aporte que realiza cada atributo del vector de entrada en el desempeño de un modelo. Este análisis proporciona un mejor entendimiento de los modelos y ayuda a dirigir mejor los esfuerzos en el tratamiento y obtención de los datos.

La combinación de los modelos es un aspecto que no se cubre en esta tesis. En el caso del modelo CNN y RNN se requiere una última capa Dense para que resulte solo un valor como salida. Donde la adición de más capas ocultas podría producir una mejora en el desempeño, en especial con el modelo CNN donde una capa Flatten genera un vector de salida de 6,656 valores, que solo es procesado con una capa Dense, así también, se considera un trabajo futuro realizar combinaciones entre capas LSTM y CNN 1D.

Los experimentos que se realizaron fueron para la estimación de concentración de cobre. La arquitectura e hiperparámetros empleadas en las pruebas finales y los resultados obtenidos son orientados a la estimación de concentración de cobre. Sin embargo, estos resultados no son directamente extrapolables a la estimación de Arsénico y Molibdeno, concentraciones minerales disponibles en los datos. Por lo que un trabajo futuro consistiría en establecer la capacidad de modelos en la estimación de estos minerales.

Del conjunto original de datos de 49,295,441 instancias se extraen aquellos nodos con información del LTMP y además del STMP. De esto resultan conjuntos de datos con los que se realizan los experimentos presentados. Un trabajo futuro consistiría en emplear modelos para realizar estimaciones en aquellos nodos que disponen de información para el LTMP y no del STMP. El análisis de estas estimaciones podría permitir establecer zonas con mayor tasa de error.

Otro aspecto relevante en esta tesis es la configuración empleada para los nodos vecinos y los criterios para llegar a esta. Para aprovechar la distribución de los nodos en el volumen en estudio, se consideran nodos a la misma altura del nodo central, más nodos en una altura superior y nodos en una altura inferior. Considerando esto, un trabajo futuro consistiría

en explorar otras configuraciones de nodos vecinos y el impacto en el desempeño de los modelos y el tiempo de ejecución requerido.

Finalmente, un trabajo que podría producir un gran impacto es el desarrollo de un framework para el análisis de los datos, su procesamiento y la correcta implementación y evaluación de modelos ML. Este tipo de trabajos se ha desarrollado en otras áreas (Pospieszny et al., 2018) y establece un estándar que facilita el seguimiento de avances y por lo tanto la propuesta de nuevas y más eficientes soluciones.

Apéndice A

Ejemplos Convolución 1D

En base a la descripción de la operación de convolución 1D descrita en la sección 2.5.1, se presentan ejemplos del cálculo de diferentes elementos de un feature map de salida.

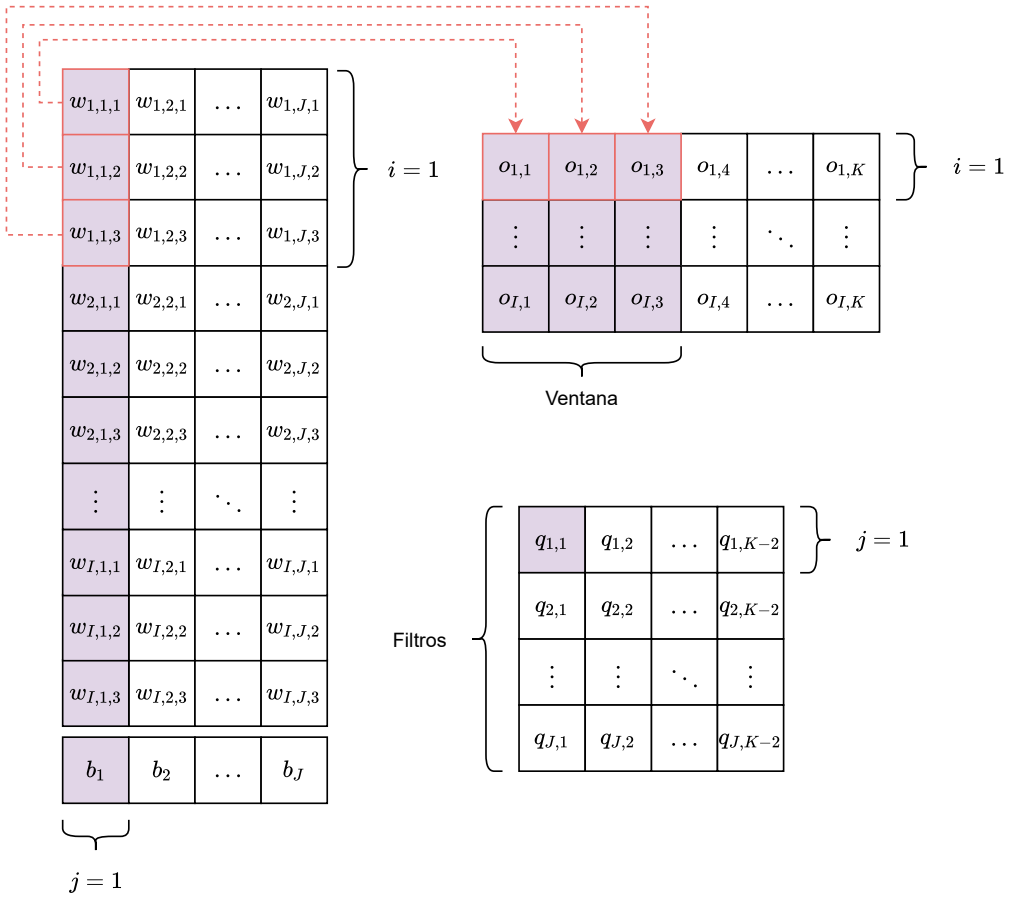


Figura A.1: Ejemplo del cálculo de $q_{1,1}$ con una ventana $V = 3$

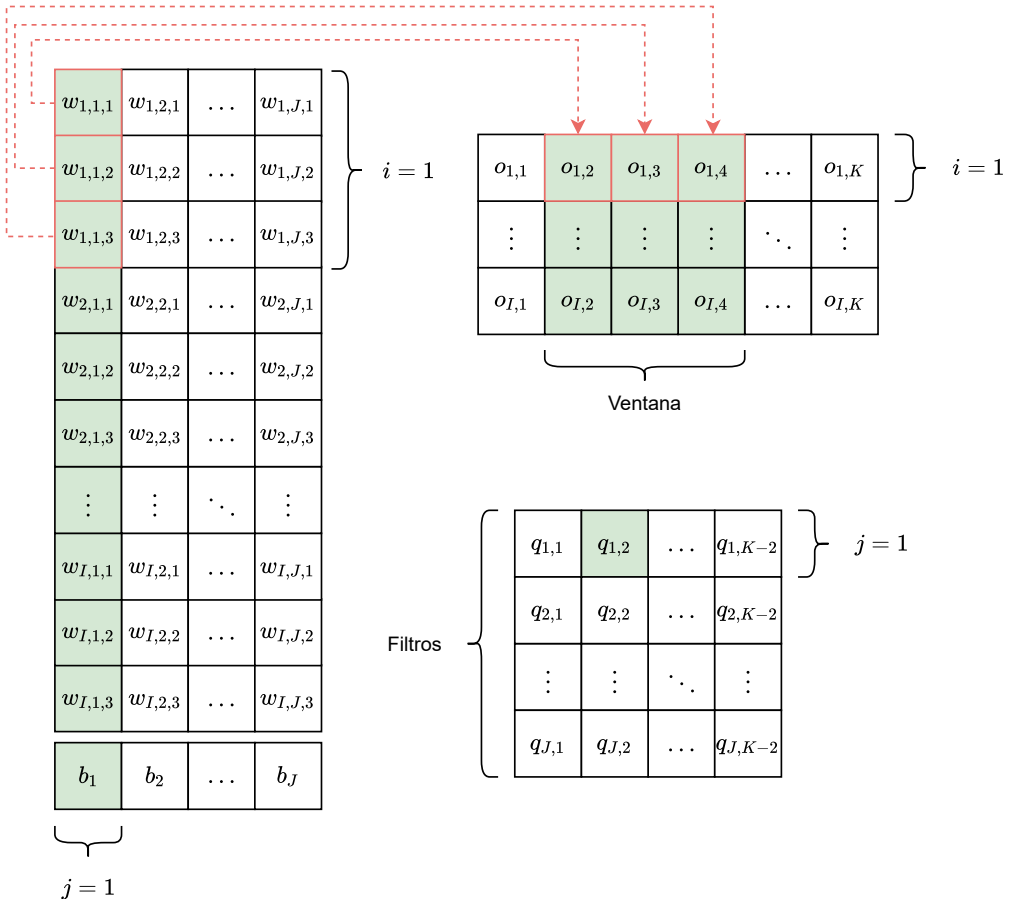


Figura A.2: Ejemplo del cálculo de $q_{1,2}$ ($j = 1$ y $m = 2$) para una ventana $V = 3$.

El cambio en el valor de m genera un desplazamiento de la ventana de instancias empleadas, lo que genera un nuevo valor en el feature map de salida en la misma profundidad y dado que no varía el valor de j, se aplica el mismo filtro que en la figura anterior.

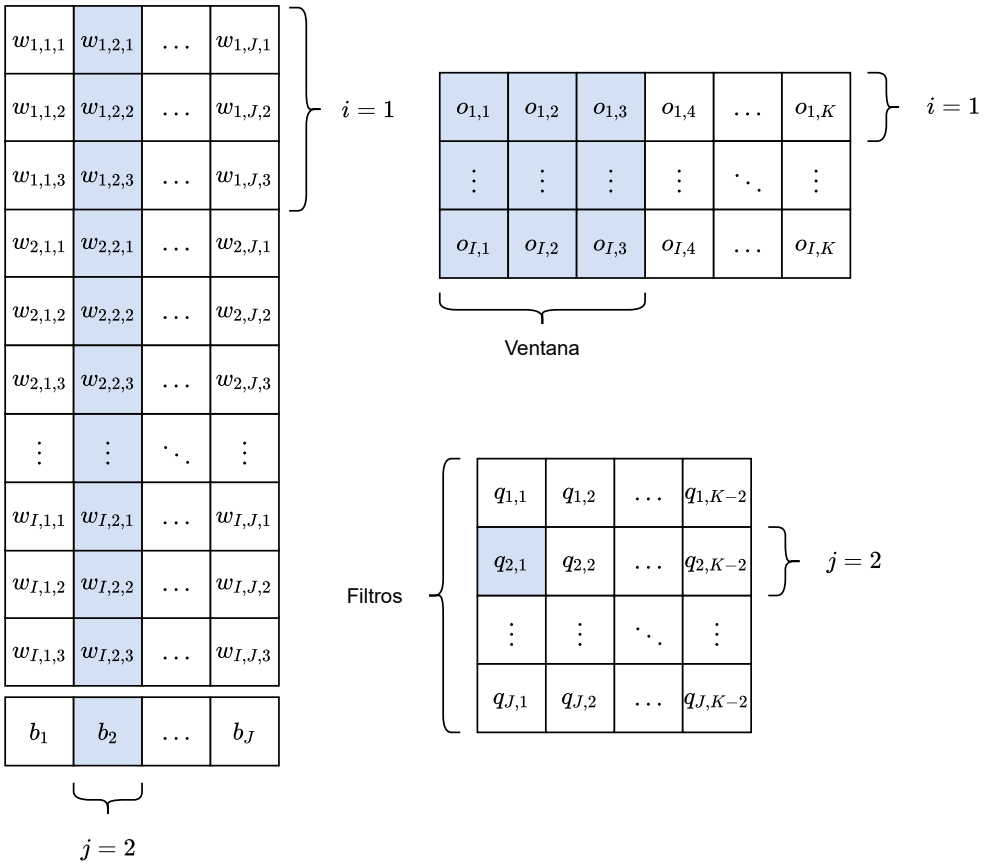


Figura A.3: Ejemplo del cálculo de $q_{2,1}$ ($j = 2$ y $m = 1$) con una ventana $V = 3$

Respecto de las figuras anteriores, un cambio en el valor de j implica el uso de otro filtro, que en el feature map de salida se traduce en el cálculo de un elemento distinto y aumenta la profundidad de la salida.

Apéndice B

Tablas de Resultados para modelo FNN

Tabla B.1: FNN: Resultados por cantidad de neuronas

Neuronas	MAE (DE)	MSE (DE)	R^2 (DE)
5	0.1899 (0.0024)	0.1962 (0.0111)	0.5328 (0.0121)
20	0.1862 (0.0017)	0.1915 (0.0110)	0.5438 (0.0126)
100	0.1853 (0.0018)	0.1872 (0.0105)	0.5541 (0.0120)
300	0.1839 (0.0019)	0.1845 (0.0101)	0.5604 (0.0118)
400	0.1847 (0.0026)	0.1840 (0.0100)	0.5616 (0.0115)
500	0.1843 (0.0014)	0.1843 (0.0103)	0.5609 (0.0118)
600	0.1852 (0.0037)	0.1844 (0.0102)	0.5607 (0.0116)
700	0.1847 (0.0040)	0.1839 (0.0098)	0.5618 (0.0109)
800	0.1848 (0.0030)	**0.1837** (0.0099)	**0.5623** (0.0112)
900	0.1836 (0.0027)	0.1842 (0.0101)	0.5611 (0.0117)
1000	**0.1829** (0.0023)	0.1841 (0.0095)	0.5613 (0.0105)

Tabla B.2: FNN: Resultados por cantidad de capas

Capas	MAE (DE)	MSE (DE)	R^2 (DE)
1	0.1855 (0.0032)	0.1842 (0.0105)	0.5613 (0.0126)
2	0.1805 (0.0017)	0.1801 (0.0095)	0.5708 (0.0106)
3	**0.1784** (0.0032)	**0.1780** (0.0070)	**0.5757** (0.0073)
4	0.1838 (0.0065)	0.1800 (0.0077)	0.5711 (0.0079)
5	0.1822 (0.0057)	0.1803 (0.0078)	0.5704 (0.0077)

Tabla B.3: FNN: Resultados por función de activación

Activación	MAE (DE)	MSE (DE)	R^2 (DE)
softmax	0.1831 (0.0017)	0.1861 (0.0103)	0.5567 (0.0115)
softplus	**0.1798** (0.0034)	0.1816 (0.0084)	0.5672 (0.0086)
softsign	0.1912 (0.0023)	0.1810 (0.0083)	0.5688 (0.0087)
relu	0.1807 (0.0044)	0.1796 (0.0074)	0.5720 (0.0075)
tanh	0.1893 (0.0036)	0.1835 (0.0097)	0.5629 (0.0106)
sigmoid	0.1822 (0.0054)	**0.1786** (0.0071)	**0.5744** (0.0076)
hard sigmoid	0.1848 (0.0044)	0.1817 (0.0081)	0.5671 (0.0086)
linear	0.1983 (0.0022)	0.2067 (0.0114)	0.5077 (0.0124)

Tabla B.4: FNN: Resultados por valor en dropout

Dropout	MAE (DE)	MSE (DE)	R^2 (DE)
0.0	**0.1806** (0.0034)	0.1803 (0.0097)	0.5704 (0.0113)
0.1	0.1819 (0.0048)	**0.1797** (0.0079)	**0.5719** (0.0076)
0.2	0.1839 (0.0058)	0.1826 (0.0080)	0.5649 (0.0070)
0.3	0.1848 (0.0063)	0.1836 (0.0092)	0.5625 (0.0092)
0.4	0.1820 (0.0036)	0.1844 (0.0097)	0.5607 (0.0109)
0.5	0.1891 (0.0064)	0.1888 (0.0111)	0.5503 (0.0138)
0.6	0.1904 (0.0043)	0.1929 (0.0112)	0.5406 (0.0125)
0.7	0.1900 (0.0069)	0.1952 (0.0117)	0.5350 (0.0138)
0.8	0.1985 (0.0032)	0.1970 (0.0113)	0.5308 (0.0128)
0.9	0.1968 (0.0122)	0.2038 (0.0123)	0.5146 (0.0147)

Tabla B.5: FNN: Resultados por optimizador

Optimizador	MAE (DE)	MSE (DE)	R^2 (DE)
SGD	0.1908 (0.0021)	0.1969 (0.0116)	0.5309 (0.0134)
RMSprop	0.1806 (0.0046)	0.1803 (0.0084)	0.5703 (0.0090)
Adagrad	0.1930 (0.0017)	0.1999 (0.0113)	0.5239 (0.0124)
Adadelta	0.1962 (0.0015)	0.2030 (0.0112)	0.5164 (0.0121)
Adam	0.1798 (0.0051)	0.1758 (0.0076)	0.5810 (0.0080)
Adamax	0.1779 (0.0033)	0.1768 (0.0073)	0.5786 (0.0088)
Nadam	**0.1775** (0.0010)	**0.1740** (0.0079)	**0.5852** (0.0101)

Apéndice C

Tablas de Resultados del modelo CNN

Tabla C.1: CNN: Resultados por cantidad de filtros

Filtros	MAE (DE)	MSE (DE)	R^2 (DE)
16	0.1850 (0.0054)	0.1835 (0.0142)	0.5928 (0.0203)
32	0.1862 (0.0046)	0.1791 (0.0143)	0.6027 (0.0201)
64	0.1846 (0.0057)	0.1773 (0.0139)	0.6066 (0.0198)
128	0.1854 (0.0050)	0.1761 (0.0138)	0.6091 (0.0198)
256	0.1859 (0.0048)	0.1765 (0.0141)	0.6084 (0.0200)
512	**0.1844** (0.0045)	0.1759 (0.0142)	0.6097 (0.0208)
1024	0.1870 (0.0042)	**0.1758** (0.0137)	**0.6099** (0.0193)

Tabla C.2: CNN: Resultados por tamaño de kernel

kernel	MAE (DE)	MSE (DE)	R^2 (DE)
3	0.1852 (0.0053)	**0.1743** (0.0133)	**0.6131** (0.0189)
5	**0.1844** (0.0045)	0.1759 (0.0142)	0.6097 (0.0208)

Tabla C.3: CNN: Resultados por cantidad de capas

capas	MAE (DE)	MSE (DE)	R^2 (DE)
1	0.1852 (0.0053)	0.1743 (0.0133)	0.6131 (0.0189)
2	0.1802 (0.0036)	0.1696 (0.0139)	0.6235 (0.0216)
3	**0.1792** (0.0043)	**0.1667** (0.0141)	**0.6298** (0.0235)
4	0.1836 (0.0076)	0.1688 (0.0153)	0.6253 (0.0250)

Tabla C.4: CNN: Resultados por función de activación

Activación	MAE (DE)	MSE (DE)	R^2 (DE)
None	0.2038 (0.0048)	0.2127 (0.0158)	0.5281 (0.0171)
relu	0.1802 (0.0036)	0.1696 (0.0139)	0.6235 (0.0216)
sigmoid	0.1786 (0.0051)	**0.1693** (0.0143)	**0.6243** (0.0225)
tanh	0.2074 (0.0061)	0.1950 (0.0146)	0.5672 (0.0183)
softmax	**0.1744** (0.0026)	0.1701 (0.0131)	0.6224 (0.0192)
softplus	0.1820 (0.0037)	0.1703 (0.0136)	0.6220 (0.0214)
softsign	0.2099 (0.0067)	0.1911 (0.0134)	0.5760 (0.0156)
hard sigmoid	0.1807 (0.0036)	0.1707 (0.0140)	0.6212 (0.0212)

Tabla C.5: CNN: Resultados por optimizador

Optimizador	MAE (DE)	MSE (DE)	R^2 (DE)
SGD	0.3774 (0.0938)	0.3986 (0.0955)	0.1102 (0.2207)
RMSprop	0.1786 (0.0051)	0.1693 (0.0143)	0.6243 (0.0225)
Adagrad	0.1766 (0.0024)	0.1683 (0.0132)	0.6263 (0.0213)
Adadelta	0.1806 (0.0036)	0.1705 (0.0137)	0.6215 (0.0200)
Adam	0.1763 (0.0038)	0.1651 (0.0123)	0.6333 (0.0202)
Adamax	0.1767 (0.0023)	0.1656 (0.0132)	0.6323 (0.0217)
Nadam	**0.1746** (0.0036)	**0.1633** (0.0119)	**0.6374** (0.0192)

Apéndice D

Tablas de Resultados del modelo RNN

Tabla D.1: RNN: Resultados por cantidad de neuronas

Neuronas	MAE (DE)	MSE (DE)	R^2 (DE)
5	0.1811 (0.0033)	0.1803 (0.0144)	0.5998 (0.0209)
20	0.1754 (0.0028)	0.1681 (0.0128)	0.6268 (0.0204)
50	0.1742 (0.0035)	0.1647 (0.0140)	0.6344 (0.0224)
100	0.1691 (0.0032)	0.1603 (0.0121)	0.6440 (0.0192)
300	0.1673 (0.0033)	0.1562 (0.0113)	0.6532 (0.0160)
500	**0.1569** (0.0068)	**0.1488** (0.0147)	**0.6699** (0.0225)

Tabla D.2: RNN: Resultados por cantidad de capas

Neuronas	MAE (DE)	MSE (DE)	R^2 (DE)
1	0.1723 (0.0037)	0.1603 (0.0136)	0.6442 (0.0224)
2	0.1692 (0.0029)	**0.1590** (0.0119)	**0.6470** (0.0190)
3	0.1731 (0.0063)	0.1599 (0.0118)	0.6449 (0.0180)
4	**0.1684** (0.0064)	0.1598 (0.0124)	0.6451 (0.0201)

Tabla D.3: RNN: Resultados por dropout recurrente

Dropout recurrente	MAE (DE)	MSE (DE)	R^2 (DE)
0.1	0.1669 (0.0018)	0.1561 (0.0122)	0.6534 (0.0214)
0.2	0.1692 (0.0045)	0.1562 (0.0119)	0.6531 (0.0195)
0.3	0.1676 (0.0024)	0.1553 (0.0114)	0.6552 (0.0187)
0.4	0.1681 (0.0043)	**0.1548** (0.0117)	**0.6562** (0.0212)
0.5	0.1676 (0.0051)	0.1550 (0.0107)	0.6559 (0.0169)
0.6	**0.1659** (0.0043)	0.1550 (0.0113)	0.6557 (0.0190)
0.7	0.1690 (0.0048)	0.1554 (0.0122)	0.6551 (0.0187)
0.8	0.1686 (0.0047)	0.1577 (0.0127)	0.6497 (0.0229)
0.9	0.1693 (0.0043)	0.1576 (0.0131)	0.6502 (0.0222)

Tabla D.4: RNN: Resultados por dropout

Dropout	MAE (DE)	MSE (DE)	R^2 (DE)
0.1	0.1681 (0.0038)	**0.1579** (0.0135)	**0.6495** (0.0220)
0.2	**0.1675** (0.0039)	0.1620 (0.0122)	0.6403 (0.0195)
0.3	0.1703 (0.0042)	0.1688 (0.0143)	0.6255 (0.0211)
0.4	0.1759 (0.0078)	0.1757 (0.0144)	0.6102 (0.0203)
0.5	0.1725 (0.0025)	0.1855 (0.0153)	0.5886 (0.0195)
0.6	0.1807 (0.0058)	0.1950 (0.0170)	0.5675 (0.0220)
0.7	0.1852 (0.0066)	0.2069 (0.0153)	0.5409 (0.0169)
0.8	0.1913 (0.0090)	0.2183 (0.0183)	0.5158 (0.0243)
0.9	0.2016 (0.0063)	0.2343 (0.0178)	0.4801 (0.0245)

Referencias

Ossama Abdel-Hamid, Abdel-rahman Mohamed, Hui Jiang, Li Deng, Gerald Penn, and Dong Yu. Convolutional neural networks for speech recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 22(10):1533–1545, 2014. doi: 10.1109/TASLP.2014.2339736.

Thomas Afeni, Abiodun Lawal, and Rasaq Adeyemi. Re-examination of itakpe iron ore deposit for reserve estimation using geostatistics and artificial neural network techniques. *Arabian Journal of Geosciences*, 13(14):657, Jul 2020. ISSN 1866-7538. doi: 10.1007/s12517-020-05644-9. URL https://doi.org/10.1007/s12517-020-05644-9.

Ethem Alpaydin. *Introduction to Machine Learning*. Adaptive Computation and Machine Learning. MIT Press, Cambridge, MA, 3 edition, 2014. ISBN 978-0-262-02818-9.

Sylvain Arlot and Alain Celisse. A survey of cross-validation procedures for model selection. *Statistics Surveys*, 4(0):40–79, 2010. ISSN 1935-7516. doi: 10.1214/09-ss054. URL http://dx.doi.org/10.1214/09-SS054.

Mehdi Badel, Saeed Angorani, and Masoud Shariat Panahi. The application of median indicator kriging and neural network in modeling mixed population in an iron ore deposit. *Computers & Geosciences*, 37(4):530 – 540, 2011. ISSN 0098-3004. doi: https://doi.org/10.1016/j.cageo.2010.07.009. URL http://www.sciencedirect.com/science/article/pii/S0098300410003006.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *Transactions on neural networks*, 5(2):157–166, 1994. doi: 10.1109/72.279181.

Thomas Breuel. Benchmarking of lstm networks, 2015. URL https://arxiv.org/abs/1508.02774.

Antoine Caté, Lorenzo Perozzi, Erwan Gloaguen, and Martin Blouin. Machine learning as a tool for geologists. *Leading Edge*, 36(3):215–219, 2017. ISSN 19383789. doi: 10.1190/tle36030215.1.

Snehamoy Chatterjee, Sukumar Bandopadhyay, Rajive Ganguli, A. Bhattacherjee, B. Samanta, and S. K. Pal. General regression neural network residual estimation for ore

grade prediction of limestone deposit. *Mining Technology*, 116(3):89–99, 2007. doi: 10.1179/174328607X228875. URL https://doi.org/10.1179/174328607X228875.

Snehamoy Chatterjee, Sukumar Bandopadhyay, and David Machuca. Ore grade prediction using a genetic algorithm and clustering based ensemble neural network model. *Mathematical Geosciences*, 42(3):309–326, Apr 2010. ISSN 1874-8953. doi: 10.1007/s11004-010-9264-y. URL https://doi.org/10.1007/s11004-010-9264-y.

Yuntian Chen and Dongxiao Zhang. Physics-constrained deep learning of geomechanical logs. *IEEE Transactions on Geoscience and Remote Sensing*, page 1–12, Feb 2020. ISSN 0196-2892. doi: 10.1109/tgrs.2020.2973171.

Francois Chollet. *Deep Learning with Python*. Manning Publications Co., USA, 1 edition, 2017. ISBN 1617294438.

Simon Dominy, Mark Noppé, and Alwyn Annels. Errors and uncertainty in mineral resource and ore reserve estimation: The importance of getting it right. *Exploration and Mining Geology*, 11(1–4):77–98, 2002. ISSN 09641823. doi: 10.2113/11.1-4.77.

Timothy Dozat. Incorporating nesterov momentum into adam. *ICLR Workshop*, 2016.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(61): 2121–2159, 2011. URL http://jmlr.org/papers/v12/duchi11a.html.

Vincent Dumoulin and Francesco Visin. A guide to convolution arithmetic for deep learning, 2018. URL https://arxiv.org/pdf/1603.07285.pdf.

Sinjita Dutta, Debasmita Misra, Rajive Ganguli, Biswajit Samanta, and Sukumar Bandopadhyay. A hybrid ensemble model of kriging and neural network for ore grade estimation. *International Journal of Mining, Reclamation and Environment*, 20 (1):33–45, 2006. doi: 10.1080/13895260500322236. URL https://doi.org/10.1080/13895260500322236.

Sridhar Dutta, Sukumar Bandopadhyay, Rajive Ganguli, and Debasmita Misra. Machine learning algorithms and their application to ore reserve estimation of sparse and imprecise data. *Journal of Intelligent Learning Systems and Applications*, 02(02):86–96, 2010. ISSN 2150-8402. doi: 10.4236/jilsa.2010.22012.

Andrés García-Floriano, Cuauhtémoc López-Martín, Cornelio Yáñez-Márquez, and Alain Abran. Support vector regression for predicting software enhancement effort. *Information and Software Technology*, 97(January):99–109, 2018. ISSN 09505849. doi: 10.1016/j.infsof.2018.01.003. URL https://doi.org/10.1016/j.infsof.2018.01.003.

Felix Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: continual prediction with lstm. In *1999 Ninth International Conference on Artificial Neural Networks ICANN 99. (Conf. Publ. No. 470)*, volume 2, pages 850–855 vol.2, 1999. doi: 10.1049/cp:19991218.

Javad Gholamnejad, Sara Kasmaee, Amirhosein Kohsary, and Ali Nezamolhosseini. Grade estimation of ore stockpiles by using artificial neural networks: case study on choghart iron mine in iran. *Mining and Mineral Engineering*, 4(1):17–25, 2012. URL http://www.inderscience.com/offer.php?id=47997.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. The MIT Press, 2016. ISBN 0262035618. URL http://www.deeplearningbook.org.

Agam Goswami, Manoj Mishra, and Dipti Patra. Adapting pattern recognition approach for uncertainty assessment in the geologic resource estimation for indian iron ore mines. In *2016 International Conference on Signal Processing, Communication, Power and Embedded System (SCOPES)*, page 1816–1821. IEEE, Oct 2016. ISBN 978-1-5090-4620-1. doi: 10.1109/SCOPES.2016.7955758. URL http://ieeexplore.ieee.org/document/7955758/.

Agam Goswami, Manoj Mishra, and Dipti Patra. Investigation of general regression neural network architecture for grade estimation of an indian iron ore deposit. *Arabian Journal of Geosciences*, 10(4):80, Feb 2017. ISSN 1866-7538. doi: 10.1007/s12517-017-2868-5. URL https://doi.org/10.1007/s12517-017-2868-5.

Vishnu Hari, Bharath Kalyan, Mandar Chitre, and Varadarajan Ganesan. Spatial modeling of deep-sea ferromanganese nodules with limited data using neural networks. *IEEE Journal of Oceanic Engineering*, 43(4):997–1014, 2018. doi: 10.1109/JOE.2017.2752757.

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

Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin. A practical guide to support vector classification. *BJU international*, 101(1):1396–400, 2008. ISSN 1464-410X. URL http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf.

Jianglin Huang, Yan Fu Li, and Min Xie. An empirical analysis of data preprocessing for machine learning-based software cost estimation. *Information and Software Technology*, 67:108–127, 2015. ISSN 09505849. doi: 10.1016/j.infsof.2015.07.004. URL http://dx.doi.org/10.1016/j.infsof.2015.07.004.

Carla Iglesias, Isabel Antunes, M. T.D. Albuquerque, J. Martínez, and J. Taboada. Predicting ore content throughout a machine learning procedure – an sn-w enrichment case study. *Journal of Geochemical Exploration*, 208:106405, 2020. ISSN 03756742. doi: 10.1016/j.gexplo.2019.106405. URL https://doi.org/10.1016/j.gexplo.2019.106405.

Bahram Jafrasteh and Nader Fathianpour. A hybrid simultaneous perturbation artificial bee colony and back-propagation algorithm for training a local linear radial basis neural network on ore grade estimation. *Neurocomputing*, 235:217 – 227, 2017. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2017.01.016. URL http://www.sciencedirect.com/science/article/pii/S0925231217300474.

Bahram Jafrasteh, Nader Fathianpour, and Alberto Suárez. Comparison of machine learning methods for copper ore grade estimation. *Computational Geosciences*, 22 (5):1371–1388, Oct 2018. ISSN 1573-1499. doi: 10.1007/s10596-018-9758-0. URL https://doi.org/10.1007/s10596-018-9758-0.

Abu Jalloh, Sasaki Kyuro, Yaguba Jalloh, and Abubakarr Barrie. Integrating artificial neural networks and geostatistics for optimum 3d geological block modeling in mineral reserve estimation: A case study. *International Journal of Mining Science and Technology*, 26(4):581–585, Jul 2016. ISSN 2095-2686. doi: 10.1016/J.IJMST.2016.05.008. URL https://www.sciencedirect.com/science/article/pii/S2095268616300167.

Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. An empirical exploration of recurrent network architectures. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, page 2342–2350. JMLR.org, 2015.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6980.

Serkan Kiranyaz, Onur Avci, Osama Abdeljaber, Turker Ince, Moncef Gabbouj, and Daniel Inman. 1d convolutional neural networks and applications: A survey. *Mechanical Systems and Signal Processing*, 151:107398, 2020. ISSN 0888-3270. doi: https://doi.org/10.1016/j.ymssp.2020.107398. URL https://www.sciencedirect.com/science/article/pii/S0888327020307846.

Xiao-li Li, Yu-ling Xie, Qian-jin Guo, and Li-hong Li. Adaptive ore grade estimation method for the mineral deposit evaluation. *Mathematical and Computer Modelling*, 52(11):1947 – 1956, 2010. ISSN 0895-7177. doi: https://doi.org/10.1016/j.mcm.2010.04.018. URL http://www.sciencedirect.com/science/article/pii/S0895717710002086. The BIC-TA 2009 Special Issue.

Xiao-li Li, Li-hong Li, Bao-lin Zhang, and Qian-jin Guo. Hybrid self-adaptive learning based particle swarm optimization and support vector regression model for grade estimation. *Neurocomputing*, 118:179–190, 2013. ISSN 09252312. doi: 10.1016/j.neucom.2013.03.002. URL http://dx.doi.org/10.1016/j.neucom.2013.03.002.

Hamid Mahmoudabadi, Mohammad Izadi, and Mohammad Bagher Menhaj. A hybrid method for grade estimation using genetic algorithm and neural networks. *Computational Geosciences*, 13(1):91–101, Mar 2009. ISSN 1573-1499. doi: 10.1007/s10596-008-9107-9. URL https://doi.org/10.1007/s10596-008-9107-9.

Yurii Nesterov. A method for unconstrained convex minimization problem with the rate of convergence $o(1/k^2)$. 1983.

Christopher Olah. Understanding lstm networks, 2015. URL http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

Harry Parker. Reconciliation principles for the mining industry. *Mining Technology*, 121 (3):160–176, 2012. doi: 10.1179/1743286312Y.0000000007. URL https://doi.org/10.1179/1743286312Y.0000000007.

Przemyslaw Pospieszny, Beata Czarnacka-Chrobot, and Andrzej Kobylinski. An effective approach for software project effort and duration estimation with machine learning algorithms. *Journal of Systems and Software*, 137:184–196, 2018. ISSN 0164-1212. doi: https://doi.org/10.1016/j.jss.2017.11.066. URL https://www.sciencedirect.com/science/article/pii/S0164121217302947.

Recmin. Recmin - conceptos generales, 2019. URL http://www.recmin.com/RecMinFree_Help/es/RecMin_web/conceptos.html.

Victor Rodriguez-Galiano, Manuel Sanchez-Castillo, Mario Chica-Olmo, and Mario Chica-Rivas. Machine learning predictive models for mineral prospectivity: An evaluation of neural networks, random forest, regression trees and support vector machines. *Ore Geology Reviews*, 71:804–818, Dec 2015. ISSN 0169-1368. doi: 10.1016/J.OREGEOREV.2015.01.001. URL https://www.sciencedirect.com/science/article/pii/S0169136815000037.

Mario Rossi and Clayton Deutsch. *Mineral resource estimation*. Springer Netherlands, Jan 2014. ISBN 9781402057175. doi: 10.1007/978-1-4020-5717-5.

Sebastian Ruder. An overview of gradient descent optimization algorithms. *CoRR*, abs/1609.04747, 2016. URL http://arxiv.org/abs/1609.04747.

David Rumelhart, Geoffrey Hinton, and Ronald Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

Biswajit Samanta. Radial basis function network for ore grade estimation. *Natural Resources Research*, 19(2):91–102, Jun 2010. ISSN 1573-8981. doi: 10.1007/s11053-010-9115-z. URL https://doi.org/10.1007/s11053-010-9115-z.

Biswajit Samanta, Sukumar Bandopadhyay, and Rajive Ganguli. Data segmentation and genetic algorithms for sparse data division in nome placer gold grade estimation using neural network and geostatistics. *Exploration and Mining Geology*, 11(1-4):69–76, 01 2002. ISSN 0964-1823. doi: 10.2113/11.1-4.69. URL https://doi.org/10.2113/11.1-4.69.

Biswajit Samanta, Sukumar Bandopadhyay, Rajive Ganguli, and Sridhar Dutta. Sparse data division using data segmentation and kohonen network for neural network and geostatistical ore grade modeling in nome offshore placer deposit. *Natural Resources Research*, 13(3):189–200, Sep 2004. ISSN 1573-8981. doi: 10.1023/B:NARR.0000046920.95725.1b. URL https://doi.org/10.1023/B:NARR.0000046920.95725.1b.

Biswajit Samanta, Sukumar Bandopadhyay, Rajive Ganguli, and Sinjita Dutta. A comparative study of the performance of single neural network vs. adaboost algorithm based

combination of multiple neural networks for mineral resource estimation. *Journal of The South African Institute of Mining and Metallurgy*, 105(4):237–246, 2005. ISSN 0038223X.

Biswajit Samanta, Sukumar. Bandopadhyay, and Rajive Ganguli. Comparative evaluation of neural network learning algorithms for ore grade estimation. *Mathematical Geology*, 38(2):175–197, Feb 2006. ISSN 1573-8868. doi: 10.1007/s11004-005-9010-z. URL https://doi.org/10.1007/s11004-005-9010-z.

Nathalie Schnitzler, Pierre-Simon Ross, and Erwan Gloaguen. Using machine learning to estimate a key missing geochemical variable in mining exploration: Application of the random forest algorithm to multi-sensor core logging data. *Journal of Geochemical Exploration*, 205:106344, Oct 2019. ISSN 03756742. doi: 10.1016/j.gexplo.2019.106344. URL https://www.sciencedirect.com/science/article/pii/S0375674218304047.

Servicio Nacional de Geología y Minería. *Anuario de la minería de Chile*, volume 1. 2019. URL https://www.sernageomin.cl/anuario-de-la-mineria-de-chile/.

Ashton Shortridge. Lecture notes in ordinary kriging. Spatial Data Analysis Course, Department of Geography, Michigan State University, Fall 2019. URL https://msu.edu/~ashton/classes/866/papers/gatrell_ordkrige.pdf.

Rahul Singh, D. Ray, and B. C. Sarkar. Recurrent neural network approach to mineral deposit modelling. In *Proceedings of the 4th IEEE International Conference on Recent Advances in Information Technology, RAIT 2018*, page 1–5. Institute of Electrical and Electronics Engineers Inc., Jun 2018. ISBN 9781538630396. doi: 10.1109/RAIT.2018.8389063.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014. URL http://jmlr.org/papers/v15/srivastava14a.html.

Pejman Tahmasebi and Ardeshir Hezarkhani. A hybrid neural networks-fuzzy logic-genetic algorithm for grade estimation. *Computers & Geosciences*, 42:18 – 27, 2012. ISSN 0098-3004. doi: https://doi.org/10.1016/j.cageo.2012.02.004. URL http://www.sciencedirect.com/science/article/pii/S0098300412000398.

Bülent Tutmez. Use of hybrid intelligent computing in mineral resources evaluation. *Applied Soft Computing*, 9(3):1023 – 1028, 2009. ISSN 1568-4946. doi: https://doi.org/10.1016/j.asoc.2009.02.001. URL http://www.sciencedirect.com/science/article/pii/S1568494609000222.

Vladimir Vapnik. The nature of statistical learning theory. *Technometrics*, 38(4):409, May 2006. ISSN 00401706. doi: 10.2307/1271324.

Frank Wilcoxon. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80–83, 1945. doi: 10.2307/3001968. URL https://www.jstor.org/stable/3001968.

Xiping Wu and Yingxin Zhou. Reserve estimation using neural network techniques. *Computers and Geosciences*, 19(4):567–575, 1993. ISSN 00983004. doi: 10.1016/0098-3004(93)90082-G.

Jorge Yamamoto. Correcting the smoothing effect of ordinary kriging estimates. *Mathematical Geology*, 37(1):69–94, 2005. ISSN 08828121. doi: 10.1007/s11004-005-8748-7.

Matthew Zeiler. ADADELTA: an adaptive learning rate method. *CoRR*, abs/1212.5701, 2012. URL http://arxiv.org/abs/1212.5701.

Xunan Zhang, Shiji Song, Jiabiao Li, and Cheng Wu. Robust ls-svm regression for ore grade estimation in a seafloor hydrothermal sulphide deposit. *Acta Oceanologica Sinica*, 32(8):16–25, Aug 2013. ISSN 1869-1099. doi: 10.1007/s13131-013-0337-x. URL https://doi.org/10.1007/s13131-013-0337-x.