UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

PREDICCIÓN DEL REGISTRO SÓNICO MEDIANTE UNA RED NEURONAL DE APRENDIZAJE PROFUNDO

TESIS

Que para obtener el título de

INGENIERO GEOFÍSICO

P R E S E N T A

JORGE ALEJANDRO VÁZQUEZ AYALA

DIRECTOR DE TESIS

DR. JOSÉ CARLOS ORTIZ ALEMÁN

Ciudad Universitaria, Cd. Mx., 2024

Resumen

El registro sónico es una herramienta importante que permite medir las propiedades mecánicas de las rocas, identificar rasgos estructurales, calibrar datos sísmicos y monitorear la integridad del pozo. A pesar de su importancia, frecuentemente se encuentra incompleto debido a cuestiones de tiempo, costo, fallas en la herramienta o datos medidos no confiables. Existen diversos métodos para generar registros sintéticos, los cuales regularmente están basados en relaciones empíricas o métodos estadísticos. En esta investigación, se aplicó una metodología de inteligencia artificial que consistió en entrenar una red neuronal profunda con datos reales provenientes de un campo petrolero en México para recrear el registro sónico a partir de su relación con otros registros geofísicos. Se entrenaron tres modelos distintos que utilizan diferentes registros como entrada para predecir el sónico. Los modelos se validaron con datos de los mismos pozos a los que el algoritmo no tuvo acceso durante el entrenamiento. Se midió el desempeño de los modelos con las métricas RMSE y MAPE, obteniendo valores satisfactorios. Además, se generaron registros sintéticos para pozos del campo donde inicialmente no se tomaron datos del sónico, demostrando la utilidad del método.

Abstract

The sonic log is crucial for measuring the mechanical properties of rocks, identifying structural characteristics, calibrating seismic data, and monitoring well wall integrity. Despite its importance, it's often incomplete due to time and/or cost constraints, tool malfunctions, or unreliable data. Numerous methods exist for generating synthetic logs, commonly based on empirical relations or statistical analysis. In this research, an artificial intelligence method was applied. A deep neural network was trained with real data from an oil field in Mexico, aiming to complete the sonic log through its relation with other geophysical well logs. Three models were trained using different logs as input. The models were validated with data from the same well that wasn't used during training. The performance of the models was measured with the metrics RMSE and MAPE, yielding satisfactory results. Finally, synthetic well logs were generated for some wells in the same field, demonstrating the method's usefulness.

Dedicatoria

A mi Madre, Padre y Hermano, a quienes debo absolutamente todo lo que soy. Aquellos que siempre me han provisto de amor y apoyo incondicional, y a quienes amo de regreso con todo mi corazón. Jamás les podré agradecer lo suficiente.

Al *Team Geofísica*: Alejandro Flores, Cristobal Cifuentes, Erick Villanueva, Nagibe Maroun y Rodrigo Barranco, por acompañarme siempre durante la carrera, en nuestras aventuras y en todo lo que hemos vivido. Mis amigos más sinceros de la vida.

Cristóbal, la persona que más me entiende. Mi amigo del alma y alguien en quien tengo toda la confianza del mundo.

Erick, quien me enseñó cosas que trascienden la carrera. Nunca olvidaré sus lecciones, sus frases folclóricas y nuestras aventuras.

Nagibe, la persona más inteligente que conozco. Siempre nos apoyamos, tanto académica como personalmente, y compartimos muchos momentos juntos.

Poblano, con una capacidad inigualable para la geofísica, responsabilidad y compromiso. Mi confidente cuando me sentía perdido

Rodrigo, mi amigo, inspiración y alguien a quien admiré durante toda la carrera, y seguiré admirando siempre.

Agradecimientos

Quiero expresar mi más sincero agradecimiento a todas las personas que, de alguna forma, hicieron posible la realización de este trabajo, tanto en el ámbito académico como personal.

Agradezco a mi asesor, el Dr. José Carlos Ortiz Alemán, por su apoyo durante esta investigación. Su conocimiento y sus sugerencias fueron imprescindibles para completar este trabajo.

Extiendo mi agradecimiento a mis sinodales, el Dr. Jaime Urrutia Fucugauchi, el M.I. Héctor Ricardo Castrejón Pineda, el M.C. Julián Zapotitla Román y el Ing. Mauricio Buendía Millán, por sus enseñanzas y valiosos consejos a lo largo de mi desarrollo profesional.

También agradezco a la Universidad Nacional Autónoma de México, a la Facultad de Ingeniería y al Instituto de Geofísica, por proporcionarme los recursos necesarios para completar mi formación académica y llevar a cabo esta investigación.

A las personas especiales en mi vida César, Elena, Erika, Fernanda, Gilberto, Gustavo, Leonardo, Luis, Néstor, Paula, Sebastián y Yusuri por ser una parte imprescindible de tantos momentos felices.

A mis profesores Orestes, Pablo, Gerardo, Fernando, Félix, Luis Javier, Edgar, Yukihiro, Sergio, Víctor y Nallely, por guiarme en todos los aspectos y ayudarme a florecer como geofísico.

Al Lic. Carlos Beteta, por abogar por mí.

Finalmente, a las familias Velázquez y Maroun, por acogerme y recibirme cálidamente en todo momento.

Índice general

Capítulo 1

Introducción

Los registros de pozos son herramientas esenciales para la evaluación y explotación de recursos en las ciencias de la Tierra. Su uso principal se encuentra en la exploración petrolera, donde permiten medir *in situ* parámetros petrofísicos cruciales para identificar y evaluar reservorios de hidrocarburos, delimitar litologías y correlacionar con datos obtenidos mediante métodos geofísicos superficiales, como la sísmica de reflexión.

Los registros sónicos, al proporcionar información detallada sobre las propiedades mecánicas y petrofísicas de las rocas, son fundamentales para la evaluación de yacimientos y la planificación de operaciones de producción. Sin embargo, el registro sónico a menudo está incompleto o ausente debido a restricciones de costo o tiempo, problemas en la estructura del pozo, fallas en la herramienta o la presencia de datos ruidosos o no confiables [36, 28]. Esto puede comprometer la precisión de los modelos geológicos.

Tradicionalmente, para generar un registro sónico sintético se han utilizado métodos geoestadísticos y empíricos [36, 38, 34, 17], que incluyen la correlación de múltiples pozos [4, 5] e incluso métodos que integran datos sísmicos [29, 33]. Aunque estos enfoques son efectivos, ninguno de ellos puede capturar completamente las relaciones no lineales entre el registro sónico y otros datos causadas por heterogeneidades y condiciones complejas en el subsuelo [57].

En los últimos años, se han comenzado a emplear métodos basados en inteligencia artificial para generar registros sintéticos. La mayoría de estos métodos utilizan Redes Neuronales Artificiales [53, 41, 28, 57], pero varían en cuanto a la arquitectura, el tipo de red neuronal empleada, los datos de entrada y su tratamiento. La ventaja principal de este nuevo enfoque es que las redes neuronales pueden manejar grandes volúmenes de datos y extraer características no lineales, lo cual es ideal para la tarea de generar un registro sónico sintético. En este contexto, se utilizará el término *predicción* para referirse al cálculo del registro sintético, ya que es un término más alineado con la terminología de la literatura de inteligencia artificial.

El objetivo de esta investigación es predecir el registro sónico a partir de otros registros geofísicos de pozo relacionados físicamente mediante un modelo de aprendizaje profundo. Se utilizaron datos reales de un campo petrolero en Tabasco, México. El programa se codificó en Python, empleando las bibliotecas Lasio para manipular datos de registros de pozo y TensorFlow para definir la arquitectura del modelo y llevar a cabo el entrenamiento.

Esta tesis se estructura de la siguiente manera: Los capítulos dos y tres presentan los antecedentes teóricos sobre los registros geofísicos de pozo y la inteligencia artificial, respectivamente. El capítulo cuatro detalla la metodología seguida para obtener la predicción, desde el preprocesamiento de los datos hasta el entrenamiento del modelo. El capítulo cinco muestra la convergencia del modelo durante el entrenamiento, su desempeño en los conjuntos de datos de entrenamiento, validación y prueba, así como los registros sónicos predichos para estos conjuntos y su aplicación en pozos del campo que carecen completamente de este registro. Finalmente, el capítulo seis discute y concluye sobre la metodología utilizada y los resultados obtenidos en esta investigación.

Capítulo 2

Registros Geofísicos de Pozo

Como se mencionó en la introducción, el objetivo de esta tesis es obtener el registro sónico a partir de otros registros de pozo relacionados. En este capítulo, se revisa la teoría general de los registros geofísicos de pozos. Se abordan las propiedades físicas de las rocas medidas con cada tipo de registro, el funcionamiento básico de las herramientas de medición y los parámetros que los relacionan entre sí. A continuación, se presentan algunas definiciones básicas.

Un registro geofísico de pozo se define como la medición de las propiedades físicas de la formación de roca atravesada por un pozo [6]. Estas mediciones se realizan con herramientas denominadas sondas, que pueden medir de manera pasiva o activa después de ejercer alguna influencia sobre la formación [11].

2.1. Registro Sónico

El registro sónico mide el tiempo que tarda una onda acústica en recorrer una longitud definida de la formación [52]. Básicamente, la herramienta (Figura 2.1 (a)), consiste en un transductor que emite una onda de sonido y dos receptores separados que miden la diferencia en el tiempo que tarda en llegar la onda a cada uno de ellos. Esta diferencia, conocida como tiempo de tránsito, es proporcional a la velocidad del sonido en la formación y a la distancia entre los receptores [15, 47]. La velocidad del sonido en la formación depende de factores como las propiedades elásticas de la matriz de roca, la porosidad y el contenido de fluidos [52].

Si hay derrumbes en la formación o la herramienta se inclina, las mediciones del tiempo de tránsito son erróneas porque hay diferente cantidad de lodo de perforación en el transmisor y el receptor. Una variante de la herramienta corrige estos efectos al incluir un segundo transmisor y otro par de receptores en la parte opuesta de la sonda [30]. A esta variante se le llama registro sónico compensado (Figura 2.1 (b)).

La influencia de la matriz de roca en la velocidad del sonido está determinada por los minerales que la componen y se define por su densidad y sus parámetros de elasticidad. Es posible distinguir los tipos de roca más comunes debido a que

el tiempo de tránsito en ellas es bien conocido. Además, dado que la velocidad de propagación de las ondas sónicas en los fluidos es menor, también se puede relacionar la velocidad con la porosidad y los fluidos del espacio poroso. En general, una mayor porosidad se asocia con una menor velocidad. Para una porosidad constante, la velocidad en gases es menor que en aceites, y la velocidad en aceites es menor que en el agua. En el caso del agua de la formación, una mayor salinidad aumenta la velocidad del sonido [47].

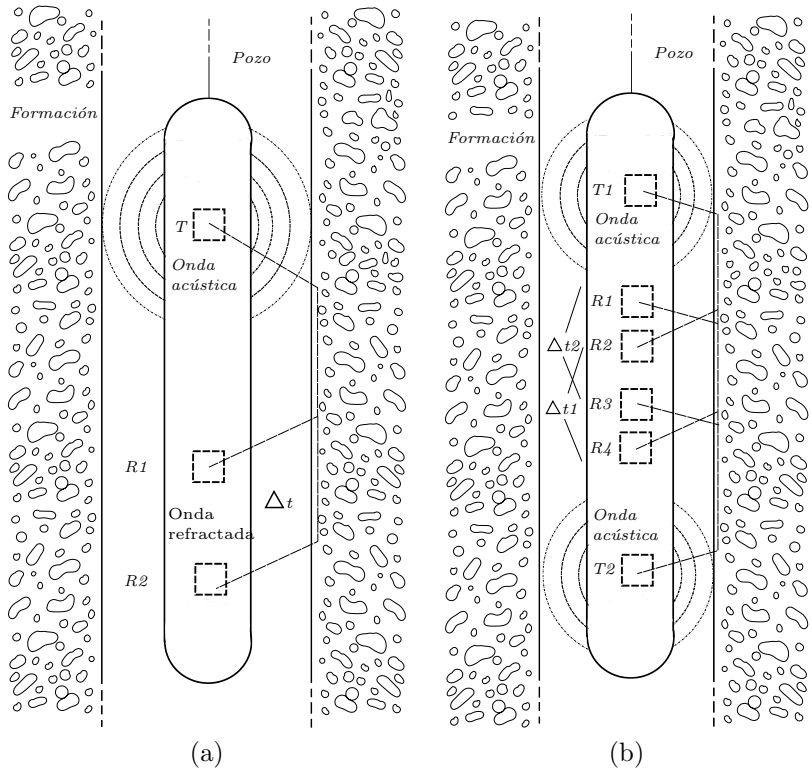


Figura 2.1: Esquemas de la sonda del registro sónico (a) y del registro sónico compensado (b). Adaptado de [11].

2.2. Registro de Resistividad

La resistividad es la propiedad que tienen los materiales para resistir el flujo de corriente eléctrica [30]. A diferencia de la resistencia eléctrica, la resistividad es una propiedad intrínseca de los materiales que no depende de sus dimensiones ni de su geometría. Esto resulta útil para identificar el tipo de material asociado con los valores de resistividad medidos.

Existen múltiples variantes de herramientas para medir la resistividad, pero todas siguen un sistema básico. Este sistema consiste en un emisor que induce un campo

eléctrico o electromagnético y un receptor que mide la respuesta de la formación a esta influencia [47] (Figura 2.2).

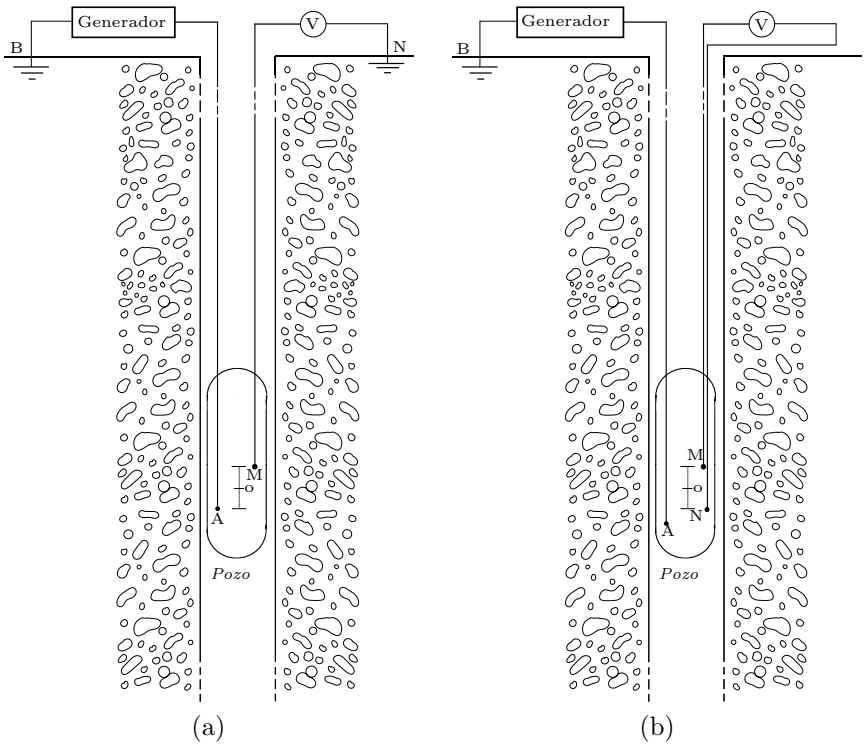


Figura 2.2: Esquemas de la sonda del registro de resistividad normal (a) y lateral (b). Adaptado de [47].

Las herramientas de resistividad no enfocadas, como la normal y lateral, presentan limitaciones al medir la resistividad en capas delgadas. Estas herramientas son altamente susceptibles a los efectos del lodo de perforación y de la pared del pozo, lo que dificulta la identificación precisa de los límites entre las capas [47]. Para mitigar estas deficiencias, se desarrollaron las herramientas de inducción. En este caso, el emisor es una bobina que genera un campo electromagnético en la formación; la componente vertical de dicho campo produce una corriente eléctrica que es proporcional a la conductividad de la formación, siendo esta el inverso de la resistividad. Esta corriente, a su vez, induce un campo electromagnético secundario, el cual genera una señal eléctrica que es detectada por la bobina receptora [30] (Figura 2.3 (a)). Las herramientas de inducción más sofisticadas tienen un arreglo de bobinas receptoras que permiten medir la conductividad a diferentes profundidades de investigación.

La resistividad de la formación depende de los minerales que componen la matriz, del contenido de arcillas, la porosidad, la cantidad y tipo de fluidos que llenan el espacio poroso y de la temperatura. En formaciones con baja porosidad y poco

contenido de arcillas, la resistividad está influenciada principalmente por la litología [44]. Generalmente, la resistividad de las rocas es alta, por lo que el flujo de corriente eléctrica en la formación ocurre principalmente a través de sales disueltas en el agua contenida en el espacio poroso de las rocas. La resistividad es inversamente proporcional a la salinidad del agua y aumenta significativamente si algún hidrocarburo ocupa el espacio poroso [11].

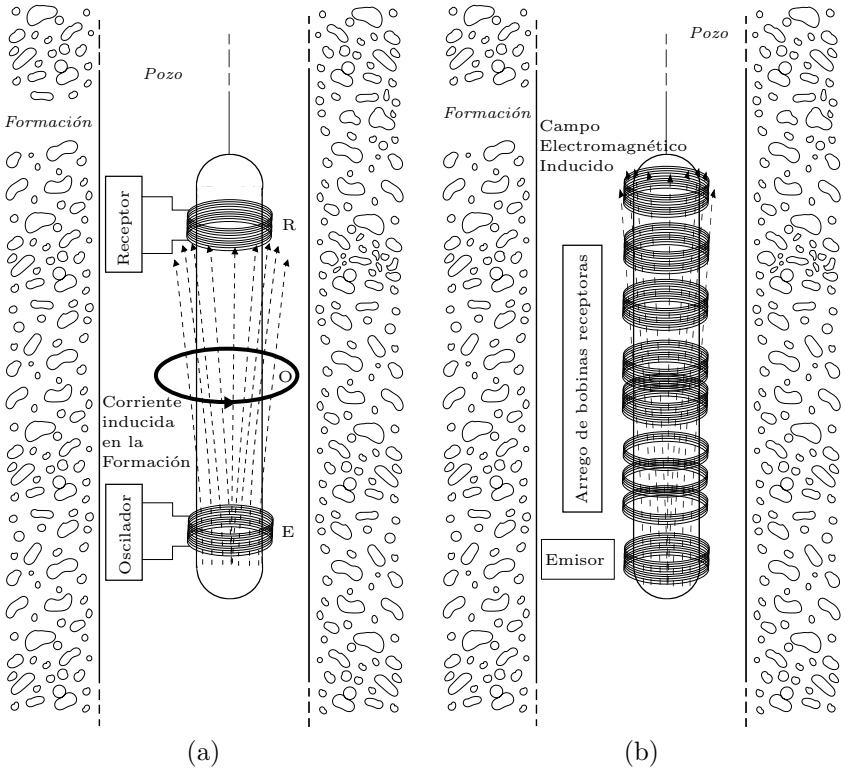


Figura 2.3: Esquemas de sondas del registro de resistividad de inducción básico (a) y en arreglo (b). Adaptado de [47, 11].

2.3. Registro de Rayos Gamma

Los rayos gamma son un tipo de radiación que proporciona valiosa información sobre las formaciones geológicas, ya que pueden penetrarlas y devolver una señal medible. En la Tierra, solo tres isótopos son responsables de la emisión de rayos gamma por parte de las formaciones geológicas: el potasio, el torio y el uranio. El potasio es la principal fuente de radiación, ya que está presente en varios grupos mineralógicos comunes, especialmente en las arcillas [11].

El registro de rayos gamma mide la radiación natural de las formaciones geológicas y puede utilizarse para diferenciar entre litologías. Las arcillas generan una respuesta

radioactiva alta, mientras que las litologías más limpias presentan una radiación significativamente más baja [44].

2.4. Registro de Densidad

El registro de densidad consiste en exponer la formación a una fuente artificial de radiación de energía media y medir el flujo de rayos gamma resultante de esta interacción [13]. El principio de medición se basa en el efecto Compton y el efecto fotoeléctrico (Figura 2.4). La fuente emite fotones o rayos gamma que colisionan con electrones del material objetivo. La energía inicial se divide entre la dispersión del fotón con energía reducida y la eyección del electrón en otra dirección, conocido como efecto Compton. Cuando el fotón transfiere toda su energía a un electrón y desaparece, se produce el efecto fotoeléctrico. Este fenómeno depende de la energía del fotón y del número atómico del material objetivo, siendo más frecuente cuando la energía es baja y el número atómico es alto [47].

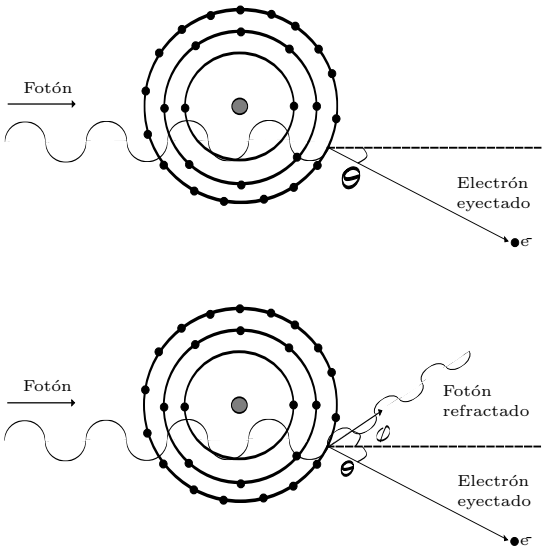


Figura 2.4: Esquema del efecto fotoélectrico (superior) y el efecto Compton (inferior). Adaptado de [47].

La cantidad de radiación que regresa a la formación está relacionada con su densidad de electrones, que a su vez está directamente vinculada con su densidad. Una formación densa tiene una alta densidad de electrones y atenúa significativamente la radiación, mientras que formaciones menos densas atenuarán la radiación en menor medida [13].

Este registro mide la densidad aparente, lo que significa que no mide solo la densidad de la litología, sino que también se ve afectado por la porosidad y por la densidad

del fluido que llena los poros.

2.5. Registro de Neutrones

Una forma de medir la porosidad de una formación es bombardeándola con neutrones de alta energía. Estos neutrones pierden velocidad al colisionar con los núcleos de los átomos que componen la formación. La pérdida de energía en estas colisiones es máxima cuando los núcleos tienen una masa similar a la de los neutrones incidentes, siendo mayor con los átomos de hidrógeno, debido a que su masa atómica es muy cercana a la de un neutrón [47]. Cuando los neutrones alcanzan su estado de mínima energía, se les denomina neutrones térmicos. Los núcleos absorben estos neutrones y emiten un rayo gamma. Dependiendo del tipo de herramienta utilizada, se mide la cantidad de neutrones térmicos, epitermales o los rayos gamma emitidos por la absorción de neutrones [14].

Este registro es especialmente sensible a la presencia de hidrógeno. Dado que los minerales que forman las rocas no contienen grandes cantidades de hidrógeno, su presencia se concentra en los fluidos dentro del espacio poroso. Por esta razón, el registro de neutrones se asocia con la porosidad de la formación. Generalmente, la herramienta devuelve unidades de porosidad con referencia a una caliza sin porosidad [2]. La Figura 2.5 se muestra un esquema del funcionamiento de esta herramienta.

2.6. Relaciones Empíricas

A lo largo de este capítulo se han destacado las propiedades de las rocas que influyen en las mediciones de los diferentes registros y las relaciones que las vinculan entre sí. Existen relaciones empíricas que conectan los registros o algún parámetro petrofísico derivado de ellos de manera cuantitativa. Algunas de estas relaciones aplicables al registro sónico se detallan a continuación:

- En 1953, Faust encontró una relación entre la velocidad de la onda compresional y la resistividad profunda: $V_p = a(RZ)^{\frac{1}{6}}$, donde V_p es la velocidad de onda, a es una constante, R es la resistividad y Z es la profundidad. [17]

  Otra versión de esta relación substituye la resistividad por el factor de formación: $V_p = a(FZ)^{\frac{1}{6}}$, para $F = \frac{R_t}{R_w}$, donde R_t es la resistividad de la formación y R_w es la resistividad del agua de la formación. [18]

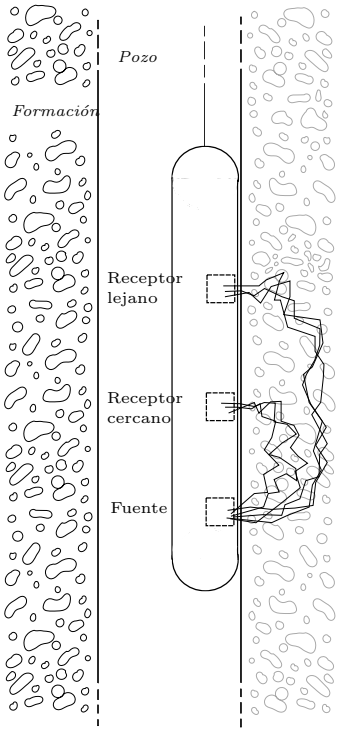


Figura 2.5: Esquema de la sonda del registro de neutrones. Adaptado de [11].

- Existe una relación para el tiempo de tránsito que, además de incluir la resistividad, incorpora el registro de rayos gamma para tener en cuenta la presencia de arcillas:

  $\Delta T = \Delta T_F f_R + (\Delta T_{sh} f_{sh} + \Delta T_{sd}(1 - f_{sh}))(1 - f_R)$,

  donde $f_R = \sqrt{0.81 R_w}/\sqrt{R}$, $f_{sh} = 0.33(2^{2G} - 1)$, $G = (GR - GR_{sd})/(GR_{sh} - GR_{sd})$.

  Aquí, ΔT es el tiempo de tránsito, f_R es la porosidad fraccional, f_{sh} es el volumen de arcilla fraccional, y los sufijos sd y sh se refieren a arena y arcilla, respectivamente. [19]

- La ley de Birch establece una relación lineal entre la densidad y la velocidad de la onda compresional: $V_p = a + \rho$, donde ρ es la densidad y a es una constante. [42]

- La relación de Gardner expresa la densidad en función de la velocidad de la onda compresional de la siguiente manera: $\rho = aV_p^m$, donde a y m son constantes. [43, 3]

  Se puede tratar esta relación como lineal aplicando logaritmos: $log(\rho) = log(a) + mlog(V_p)$.

- La ecuación de Lindseth también vincula la densidad con la velocidad de la onda compresional: $V_p = \frac{a}{1-b\rho}$, donde a y b son constantes. [42]

Las constantes de las relaciones mencionadas dependen de la litología de la formación o se calculan a partir de los datos disponibles de un grupo de pozos. Es importante destacar que estas expresiones, aunque útiles, son simplificaciones y no capturan toda la complejidad de las interacciones entre los registros.

Capítulo 3

Inteligencia Artificial

En este capítulo se presenta la teoría de inteligencia artificial necesaria para entender la metodología utilizada en esta investigación. Se explican los fundamentos del aprendizaje automático, cómo aprenden los algoritmos, algunos métodos de optimización, la necesidad del aprendizaje profundo y los tipos de redes neuronales que pertenecen a este campo.

3.1. Aprendizaje Automático

El Aprendizaje Automático, o Machine Learning (ML), es una rama de la inteligencia artificial cuyo objetivo es desarrollar algoritmos capaces de aprender a resolver tareas. En el campo de la computación, se dice que un programa aprende si su desempeño en la resolución de una tarea mejora conforme adquiere experiencia al resolverla [35]. Por lo tanto, los componentes esenciales de un programa de aprendizaje automático son la tarea a resolver, un conjunto de ejemplos que actúan como experiencia, y una métrica para medir su desempeño.

3.1.1. Aprendizaje

Los algoritmos de ML se pueden clasificar en dos tipos según la forma en que se proporcionan los datos de ejemplo para el aprendizaje: supervisado o no supervisado [16].

El aprendizaje supervisado abarca aquellos problemas en los que el conjunto de ejemplos contiene tanto las características (entradas del programa) como sus etiquetas correspondientes (resultados esperados). Es decir, este tipo de aprendizaje busca estimar la probabilidad condicional de una etiqueta dadas ciertas características [56]. Algunas de las tareas más comunes en este tipo de aprendizaje incluyen regresión, clasificación, procesamiento del lenguaje natural, búsqueda, sistemas de recomendación y aprendizaje secuencial. Predecir el registro sónico a partir de otros registros es un ejemplo de un problema de regresión.

En Machine Learning, un problema de regresión consiste en predecir un valor numéri-

co dado un conjunto de entradas. El programa aprenderá una función f: $R^n \rightarrow R$ [16], cuyo objetivo será retornar valores muy cercanos a los de las etiquetas.

El aprendizaje no supervisado, por otro lado, utiliza un conjunto de ejemplos que solo contiene características. Los algoritmos de este tipo intentan extraer información de los datos sin necesidad de intervención humana. Ejemplos de algoritmos no supervisados incluyen el Análisis de Componentes Principales y los algoritmos de agrupamiento, como k-means.

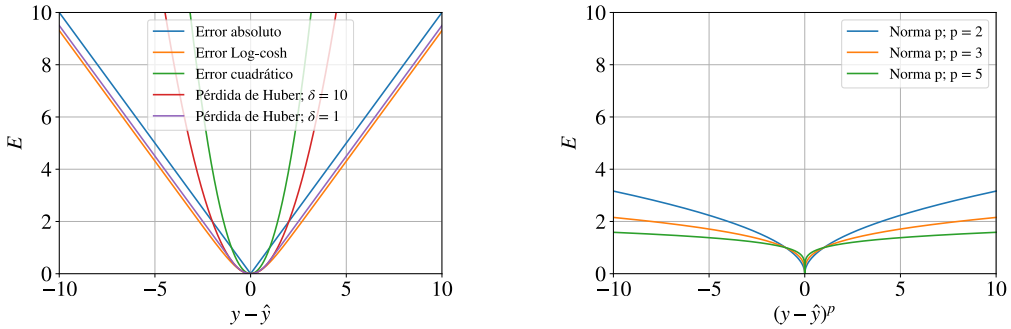
3.1.2. Métricas del desempeño

Dependiendo del tipo de problema que se desee resolver, las formas de medir el desempeño del programa variarán. Estas medidas de rendimiento se conocen como funciones de error, costo o pérdida. En un problema de regresión, donde las salidas del programa son valores continuos, el objetivo es minimizar la diferencia aritmética entre estos valores predichos y las etiquetas correspondientes [37].

A continuación, se presentan las funciones de costo más comunes para problemas de regresión, donde y representa los valores reales, $\hat{y}$ los valores predichos, y N el número de muestras:

- Error absoluto: $E(y, \hat{y}) = \sum_{i=1}^{N} |y_i - \hat{y}_i|$.
- Error Log-cosh: $E(y, \hat{y}) = \sum_{i=1}^{N} log(cosh(y_i - \hat{y}_i))$.
- Error cuadrático: $E(y, \hat{y}) = \sum_{i=1}^{N} (y_i - \hat{y}_i)^2$.
- Norma Lp: $E(y, \hat{y}) = \sqrt[p]{\sum_{i=1}^{N} |(y_i - \hat{y}_i)|^p}$.
- Pérdida de Huber: $E_\delta(y, \hat{y}) = \begin{cases} \frac{1}{2}(y_i - \hat{y}_i)^2, & \text{si } |y_i - \hat{y}_i| \leq \delta \\ \delta|y_i - \hat{y}_i| - \frac{1}{2}\delta^2, & \text{en cualquier otro caso} \end{cases}$

En la Figura 3.1 se muestran las gráficas de las distintas funciones de error mencionadas. La función Log-cosh está diseñada para asemejarse al error absoluto pero con la ventaja de que está es dos veces diferenciable. La función de pérdida de Huber es un caso similar donde, dependiendo del valor de δ, esta se aproxima al error absoluto o al error cuadrático. En general, se elige la función de costo dependiendo de la medida en que se quiera penalizar las diferencias grandes entre las predicciones y los datos reales.



(a) (b)

Figura 3.1: Gráficas del comportamiento de las diferentes funciones de error. (a)presenta el error absoluto, cuadrático y las funciones similares, y (b) muestra la normal L con distintos valores de p.

3.1.3. Redes Neuronales Artificiales

Las redes neuronales artificiales (ANNs) están ligeramente inspiradas en los sistemas neuronales naturales. Se utiliza el término "ligeramente"porque las ANNs no capturan todas las complejidades de los sistemas biológicos, y para optimizar los procesos computacionales, estos algoritmos no reflejan exactamente los procesos biológicos [35].

Las ANNs, tal como las conocemos hoy, comenzaron con Rosenblatt [45, 21], quien desarrolló un sistema neuronal hipotético llamado perceptrón. Los perceptrones están compuestos por tres tipos de unidades: sensoriales (S), asociativas (A) y de respuesta (R). Estas unidades reciben señales, ya sean externas o generadas internamente, y producen una señal de respuesta en función de la entrada.

En un perceptrón simple, las conexiones se establecen únicamente entre las unidades S y las unidades A, y entre las unidades A y una sola unidad R. Dependiendo del estímulo, las unidades S se excitarán o inhibirán, produciendo una salida binaria. Las unidades A combinan estas salidas mediante una suma ponderada, donde cada unidad A tiene un peso ω diferente y un mismo umbral θ para todas. El resultado y_i de estas unidades se suma ponderadamente de acuerdo a un segundo conjunto de pesos a y pasa a la unidad R, que compara este valor con cero. Si el resultado es mayor o igual a cero, la salida del perceptrón será 1; de lo contrario, será 0 [21].

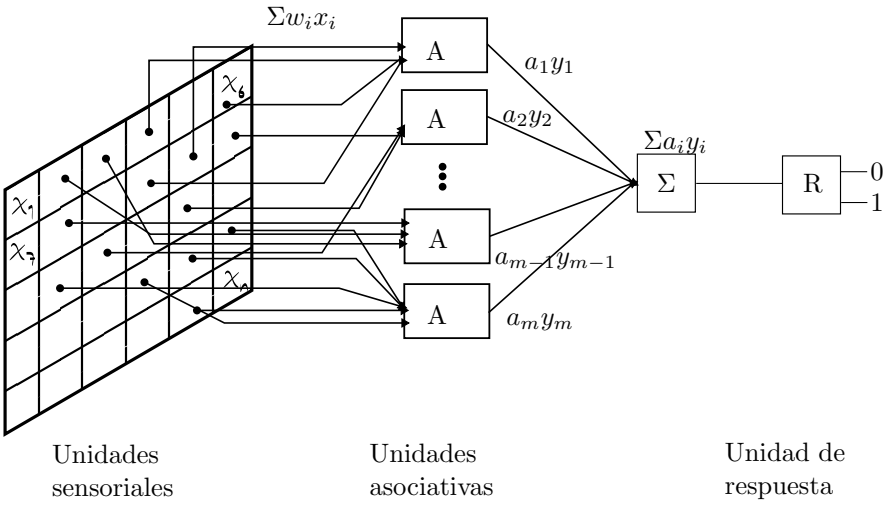


Figura 3.2: Esquema de la estructura del perceptrón de Rosenblatt. Adaptado de [21].

$$y = \begin{cases} 1 & \text{si } \sum_{i=1}^{N} (\omega_i x_i) \geq \theta \\ 0 & \text{si } \sum_{i=1}^{N} (\omega_i x_i) < \theta \end{cases}.$$

$$R = \begin{cases} 1 & \text{si } \sum_{i=1}^{M} (a_i y_i) \geq 0 \\ 0 & \text{si } \sum_{i=1}^{M} (a_i y_i) < 0 \end{cases}.$$

Actualmente, la nomenclatura de las unidades del perceptrón ha sido reemplazada, refiriéndose a estas simplemente como unidades de entrada, ocultas y de salida.

En el esquema descrito anteriormente, la respuesta del perceptrón está definida por una función llamada escalón unitario, que es discontinua y, por lo tanto, no es diferenciable. Esta falta de diferenciabilidad presenta problemas para los algoritmos de entrenamiento más avanzados, que se basan en el cálculo del gradiente. Para abordar esta limitación, se han desarrollado otras funciones que cumplen un rol similar al del escalón unitario pero sin esta carencia. Estas funciones se conocen como funciones de activación.

Una función de activación es una transformación escalar a escalar, $g : R \to R$, que es diferenciable casi en todo su dominio [9]. Las funciones de activación determinan cómo responden las neuronas artificiales a los estímulos. Incluir este tipo de funciones en las redes neuronales es esencial para permitir que estas redes modelen relaciones no lineales entre las entradas y las salidas. Sin funciones de activación, las redes neuronales no serían más que regresiones lineales complejas.

Las funciones de activación más comunes (Figura 3.3), junto con sus derivadas, son [9, 26]:

- Funcion Sigmoide:

$$\sigma(x) = \frac{1}{1+e^{-x}}.$$

$$\sigma'(x) = \frac{e^{-x}}{(1+e^{-x})^2}.$$

- Tangente hiperbólica:

$$tanh(x) = \frac{sinh(x)}{cosh(x)} = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}.$$

$$tanh'(x) = \frac{4e^{-2x}}{(1+e^{-2x})^2}.$$

- Unidades lineales rectificadas (ReLU):

$$ReLU(x) = \begin{cases} x & \text{si } x \geq 0 \\ 0 & \text{si } x < 0 \end{cases}.$$

$$ReLU'(x) = \begin{cases} 1 & \text{si } x \geq 0 \\ 0 & \text{si } x < 0 \end{cases}.$$

- Leaky ReLU:

$$LReLU(x) = \begin{cases} x & \text{si } x \geq 0 \\ 0.01x & \text{si } x < 0 \end{cases}.$$

$$LReLU'(x) = \begin{cases} 1 & \text{si } x \geq 0 \\ 0.01 & \text{si } x < 0 \end{cases}.$$

- Unidad lineal exponencial (ELU):

$$ELU(x) = \begin{cases} x & \text{si } x \geq 0 \\ \alpha(e^{x}-1) & \text{si } x < 0 \end{cases}.$$

$$ELU'(x) = \begin{cases} 1 & \text{si } x \geq 0 \\ \alpha e^{x} & \text{si } x < 0 \end{cases}.$$

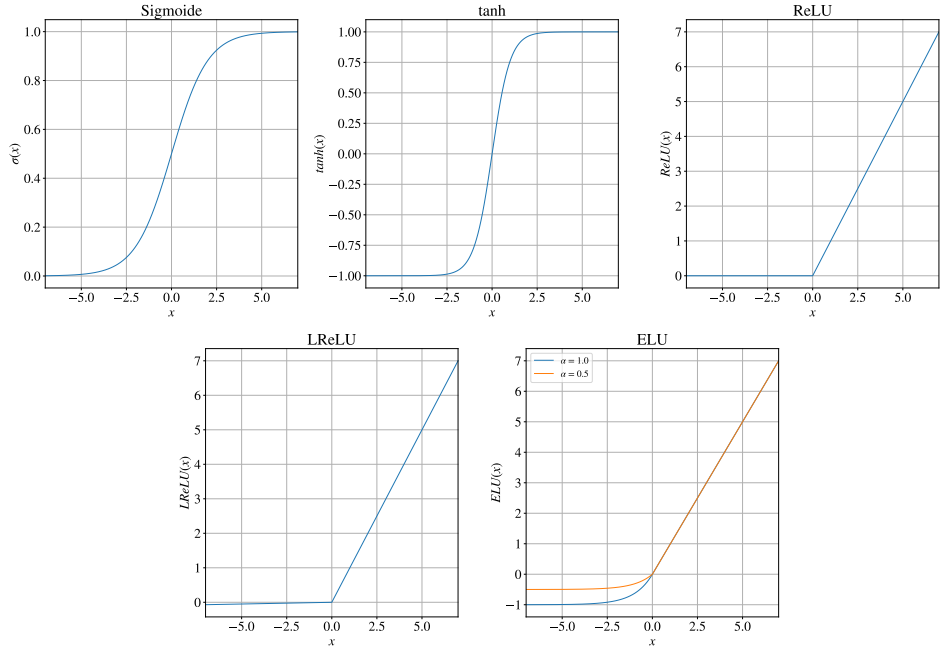


Figura 3.3: Gráficas de las funciones de activación más comunes.

3.1.4. Entrenamiento

Cualquiera de las unidades mencionadas recibe como entrada la suma ponderada de las salidas de las unidades anteriores a las que está conectada. Los pesos en estas sumas son los parámetros entrenables de la red. El proceso de entrenamiento consiste en actualizar estos pesos ω a medida que el programa gana experiencia, con el objetivo de minimizar la función de error. Los algoritmos fundamentales para este proceso son Descenso del Gradiente (Gradient Descent) y Propagación hacia Atrás (Backpropagation).

Descenso del gradiente

Los errores $E(y, \hat{y})$ mencionados en la sección (3.1.2) son funciones escalares respecto a las predicciones y las etiquetas. Si se calcula el error para cada ejemplo de forma individual, la etiqueta y será constante, por lo que el error dependerá únicamente de las predicciones $\hat{y}$. A su vez, $\hat{y}$ depende de los pesos ω que conectan las unidades de la red, por lo que el error se puede expresar en términos de los pesos $E(\omega)$.

El gradiente de una función escalar representa la dirección de máximo ascenso; por lo tanto, el gradiente negativo apunta hacia la dirección de máximo descenso. El algoritmo de Descenso del Gradiente busca una combinación de ω que minimice el

error. Esto se logra calculando el gradiente de la función de error con respecto a ω y actualizando sus valores en la dirección de máximo descenso (Figura 3.4). El gradiente está compuesto por las derivadas parciales de $E(\omega)$ respecto a cada peso ω_i:

$$\nabla E(\omega) = \left[\frac{\partial E(\omega)}{\partial \omega_0}, \frac{\partial E(\omega)}{\partial \omega_1}, \frac{\partial E(\omega)}{\partial \omega_2}, ..., \frac{\partial E(\omega)}{\partial \omega_n}\right]. \tag{3.1}$$

La actualización de ω se realiza mediante la siguiente relación [35]:

$$\omega^{(k)} = \omega^{(k-1)} + \Delta\omega, \tag{3.2}$$

donde el superíndice (k) representa el número de iteración y $\Delta\omega$ se calcula de la siguiente manera:

$$\Delta\omega = -\eta \nabla E(\omega), \tag{3.3}$$

donde η es una constante positiva llamada tasa de aprendizaje. Esta relación expresada para cada componente de ω queda como:

$$\Delta\omega_i = -\eta \frac{\partial E(\omega)}{\partial \omega_i}, \tag{3.4}$$

$$\omega_i^{(k)} = \omega_i^{(k-1)} + \Delta\omega_i. \tag{3.5}$$

Este proceso se repite para el error en cada ejemplo y se promedian sus cambios $\Delta\omega_i$, así formando los ajustes que se le harán al final de cada iteración. Esto puede llegar a ser tardado porque para actualizar ω una sola vez se deberá recorrer todo el conjunto de ejemplos. Existen variantes de este algoritmo que tratan de lidiar con este problema.

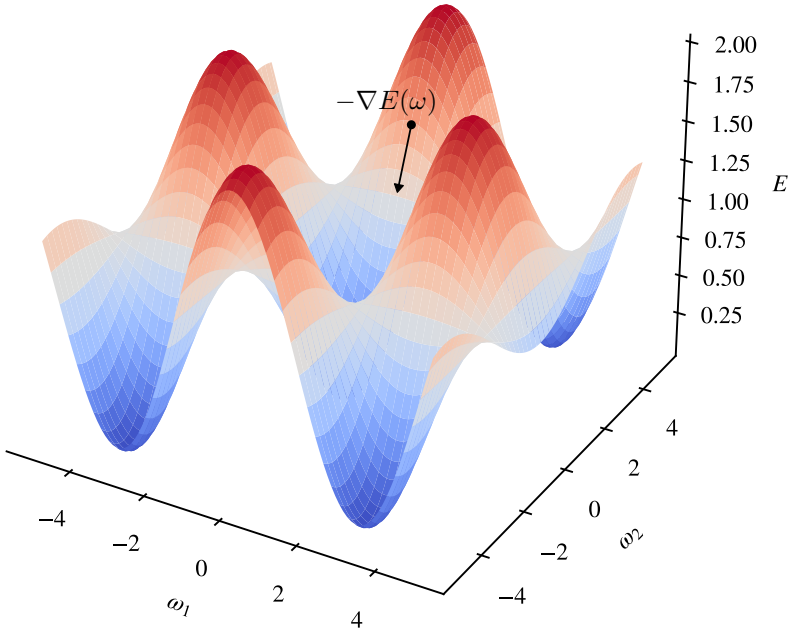


Figura 3.4: Ejemplo de una función de costo con la dirección de máximo descenso en uno de sus puntos.

Descenso del gradiente estocástico

El Descenso del Gradiente Estocástico (SGD) actualiza ω después de cada ejemplo, lo que hace que el algoritmo converja rápidamente a un mínimo de la función de error, aunque el valor del error puede fluctuar considerablemente. Esta fluctuación puede complicar la convergencia a un mínimo exacto, aunque se puede mitigar utilizando una tasa de aprendizaje que decrece a medida que avanza el algoritmo [46]. Otra versión de este algoritmo utiliza un subconjunto aleatorio del conjunto de ejemplos para cada actualización de los parámetros [22]. Estos subconjuntos se conocen como Mini-Batches.

Retropropagación

El algoritmo de Retropropagación o Backpropagation se utiliza para calcular el gradiente de la función de error con respecto a los pesos de una red neuronal de múltiples capas [1, 55, 27]. Para el desarrollo de las ecuaciones, se considera una red neuronal con L capas, donde todas las unidades de cada capa están conectadas a todas las unidades de la capa siguiente. La notación que se empleará es la siguiente:

a_i^l, es la unidad i de la capa l.

w_{ji}^l, es el peso que conecta la unidad i de la capa $l-1$ con la unidad j de la capa l.

n_l, es el número de unidades de la capa l.

z_j^l, es la suma ponderada de las unidades de la capa $l-1$ y sus respectivos pesos hacia la unidad j de la capa l.

$g(z)$, es alguna función de activación.

Se comienza el desarrollo definiendo z_j^l y a_j^l:

$$z_j^l = \sum_{i=0}^{n_l} (\omega_{ji}^l a_i^{l-1}), \tag{3.6}$$

$$a_j^l = g(z_j^l). \tag{3.7}$$

El objetivo del algoritmo es obtener $\frac{\partial E}{\partial \omega_{ji}^l}$. Se utilizará la regla de la cadena para desglosar esta derivada parcial en nuevas derivadas parciales que se puedan calcular. Esto es:

$$\frac{\partial E}{\partial \omega_{ji}^L} = \frac{\partial E}{\partial a_j^L} \frac{\partial a_j^L}{\partial z_j^L} \frac{\partial z_j^L}{\partial \omega_{ji}^L}, \tag{3.8}$$

donde $\frac{\partial E}{\partial a_j^L}$ es la derivada de la función de error, $\frac{\partial a_j^L}{\partial z_j^L}$ es la derivada de la no linealidad y $\frac{\partial z_j^L}{\partial \omega_{ji}^L}$ es la derivada de la suma ponderada. La ecuación (3.8) corresponde a la derivada del error para la última capa. A partir de este resultado se obtendrán las derivadas con respecto a los pesos de la capa anterior de la siguiente forma.

Derivando las unidades de la capa l con respecto a las unidades de la capa anterior $l-1$ y aplicando la regla de la cadena:

$$\frac{\partial a_j^l}{\partial a_i^{l-1}} = \frac{\partial a_j^l}{\partial z_i^l} \frac{\partial z_j^l}{\partial a_i^{l-1}}. \tag{3.9}$$

Substituyendo (3.6) en (3.9),

$$\frac{\partial a_j^l}{\partial a_i^{l-1}} = \frac{\partial a_j^l}{\partial z_i^l}\frac{\partial}{\partial a_i^{l-1}}\left(\sum_{i=0}^{n_l}(\omega_{ji}^l a_i^{l-1})\right) = \frac{\partial a_j^l}{\partial z_i^l}\omega_{ji}^l. \tag{3.10}$$

Utilizando la regla de la cadena para obtener la derivada en términos de ω_{ik}^{l-1}:

$$\frac{\partial a_j^l}{\partial \omega_{ik}^{l-1}} = \frac{\partial a_j^l}{\partial a_i^{l-1}}\frac{\partial a_i^{l-1}}{\partial z_i^{l-1}}\frac{\partial z_i^{l-1}}{\partial \omega_{ik}^{l-1}}. \tag{3.11}$$

Ahora, se pone la ecuación anterior en términos del error:

$$\frac{\partial E}{\partial \omega_{ik}^{l-1}} = \frac{\partial E}{\partial a_j^l}\frac{\partial a_j^l}{\partial \omega_{ik}^{l-1}} = \frac{\partial E}{\partial a_j^l}\frac{\partial a_j^l}{\partial a_i^{l-1}}\frac{\partial a_i^{l-1}}{\partial z_i^{l-1}}\frac{\partial z_i^{l-1}}{\partial \omega_{ik}^{l-1}}, \tag{3.12}$$

y substituyendo (3.10) en (3.12):

$$\frac{\partial E}{\partial \omega_{ik}^{l-1}} = \omega_{ji}^l\frac{\partial a_j^l}{\partial z_i^l}\frac{\partial E}{\partial a_j^l}\frac{\partial a_i^{l-1}}{\partial z_i^{l-1}}\frac{\partial z_i^{l-1}}{\partial \omega_{ik}^{l-1}}. \tag{3.13}$$

Con esta ecuación se tiene la derivada parcial buscada con respecto a los pesos de la capa $l-1$. Se nota que los pesos ω_{ji}^l son conocidos y las derivadas $\frac{\partial a_i^{l-1}}{\partial z_i^{l-1}}$ y $\frac{\partial z_i^{l-1}}{\partial \omega_{ik}^{l-1}}$ se calculan fácilmente. El factor $\frac{\partial E}{\partial a_j^l}$ es parte de la ecuación (3.6), por lo que es un valor que se obtuvo para la capa posterior y en esta capa ya es conocido.

Finalmente, ω_{ik}^{l-1} influencía a todas las unidades de la siguiente capa, por lo que su error será la suma de los errores asociados a cada una de estas unidades, entonces, se reescribe (3.11) como:

$$\frac{\partial E}{\partial \omega_{ik}^{l-1}} = \sum_{j=0}^{n_l}\left(\omega_{ji}^l\frac{\partial E}{\partial a_j^l}\frac{\partial a_i^{l-1}}{\partial z_i^{l-1}}\frac{\partial z_i^{l-1}}{\partial \omega_{ik}^{l-1}}\right). \tag{3.14}$$

Esta última ecuación se aplica desde la penúltima capa hasta la primera en orden descendente. De ahí el nombre del algoritmo *Retropropagación*.

Algoritmos de optimización

En el SGD, los parámetros ω se actualizan en una dirección calculada a partir del subconjunto de datos con el que se está trabajando en ese momento. Esta dirección no necesariamente es la óptima para todo el conjunto de datos, por lo que los pesos de la red pueden actualizarse en una dirección que se aleje del mínimo global. Una manera de mitigar este problema es añadir un factor adicional llamado momento a la actualización de parámetros [31, 12]. El momento reduce el impacto de los cambios pequeños en el gradiente en la actualización de ω. Esta variante del algoritmo se conoce como Descenso del Gradiente Estocástico con Momento (SGDM). El momento m se calcula de la siguiente manera:

$$m^{(k)} = \beta m^{(k-1)} + (1 - \beta)\nabla E(\omega), \tag{3.15}$$

$$\omega^{(k)} = \omega^{(k-1)} - m^{(k)}, \tag{3.16}$$

donde k y $k - 1$ representan las iteraciones actual y anterior, respectivamente, y β es un parámetro que controla la influencia del momento previo.

Por otro lado, el funcionamiento correcto del SGD convencional depende fuertemente de elegir una tasa de aprendizaje adecuada. Si esta tasa es demasiado pequeña, la convergencia del algoritmo será lenta y existe el riesgo de que converja a un mínimo local. En cambio, si la tasa es demasiado grande, es posible que la solución diverja del mínimo global. Dado que la tasa de aprendizaje es la misma para todos los parámetros, es difícil seleccionar una que sea óptima simultáneamente para todos, ya que la magnitud de sus gradientes puede variar significativamente.

El algoritmo Gradiente Adaptativo (AdaGrad) [10] aborda este problema escalando la tasa de aprendizaje de cada peso según la suma acumulada de sus gradientes anteriores. De esta forma, los gradientes de mayor magnitud tienen tasas de aprendizaje menores, y los gradientes menores tienen tasas de aprendizaje mayores. Su expresión matemática es la siguiente:

$$s^{(k)} = s^{(k-1)} + \nabla E(\omega)^2, \tag{3.17}$$

$$\omega^{(k)} = \omega^{(k-1)} - \frac{\eta}{\sqrt{s^{(k)}} + \epsilon} \nabla E(\omega). \tag{3.18}$$

AdaGrad tiene la desventaja de que la tasa de aprendizaje decae constantemente a medida que el algoritmo avanza, lo que puede ralentizar la convergencia, especialmente cuando la función de costo no es convexa.

La propagación RMS (**RMSprop**) [20] es una variante del gradiente adaptativo que escala la tasa de aprendizaje utilizando un factor s similar al momento, en lugar de la suma acumulada de la magnitud de las derivadas. Este factor da mayor importancia a los gradientes de las iteraciones recientes y su decaimiento no es constante. El factor s está definido como:

$$s^{(k)} = \beta s^{(k-1)} + (1 - \beta) \nabla E(\omega)^2. \tag{3.19}$$

La regla de actualización es la misma que en el gradiente adaptativo.

Adam [23] es un algoritmo que combina las ideas de AdaGrad y RMSprop. Calcula tasas de cambio adaptativas para cada parámetro basadas en el primer y segundo momento del gradiente. Estos se calculan de la siguiente manera:

$$m^{(k)} = \beta_1 m^{(k-1)} + (1 - \beta_1) \nabla E(\omega), \tag{3.20}$$

$$v^{(k)} = \beta_2 v^{(k-1)} + (1 - \beta_2) \nabla E(\omega)^2, \tag{3.21}$$

donde β_1 y β_2 son constantes cercanas a 1.

En las primeras iteraciones, los momentos están sesgados hacia valores pequeños debido a la multiplicación por $(1-\beta_1)$ y $(1-\beta_2)$. Estos sesgos se corrigen dividiendo entre $(1-\beta_1^k)$ y $(1-\beta_2^k)$, respectivamente. De esta forma, en las primeras iteraciones se reduce este efecto y, a medida que el algoritmo avanza, los divisores tienden a la unidad:

$$\hat{m}^{(k)} = \frac{m^{(k)}}{1 - \beta_1^k}, \tag{3.22}$$

$$\hat{v}^{(k)} = \frac{v^{(k)}}{1 - \beta_2^k}. \tag{3.23}$$

La regla de actualización de parámetros queda como:

$$\omega^{(k)} = \omega^{(k-1)} - \eta \frac{\hat{m}^{(k)}}{\sqrt{\hat{v}^{(k)}} + \epsilon}, \tag{3.24}$$

donde ϵ es un número muy pequeño para evitar la división entre cero.

3.2. Aprendizaje profundo

El aprendizaje profundo es una rama del aprendizaje automático que utiliza redes neuronales con múltiples capas para resolver problemas complejos. La principal diferencia entre el aprendizaje automático convencional y el aprendizaje profundo radica en la cantidad y el tipo de capas utilizadas. Los modelos de aprendizaje automático suelen estar compuestos por una o dos capas, mientras que en el aprendizaje profundo se emplean hasta cientos o incluso miles de capas, que pueden ser de diferentes tipos. Estas capas están interconectadas desde el inicio hasta el final del modelo, de ahí el término "profundo"[56].

En las siguientes secciones se describirán los tipos fundamentales de redes neuronales utilizadas en el aprendizaje profundo.

3.2.1. Red Neuronal Feedforward

El término Feedforward proviene de las palabras en inglés "Feed"(alimentar) y "Forward"(hacia adelante). Este tipo de redes neuronales recibe su nombre porque la información fluye de capa en capa desde la entrada x hasta la salida, sin conexiones de retroalimentación hacia capas anteriores [16]. Su arquitectura es la misma descrita en la sección (3.1.4), donde se explicaron los algoritmos de descenso del gradiente y de propagación hacia atrás. Cuando todas las neuronas de cada capa están conectadas a todas las neuronas de la capa siguiente, la red se denomina Red

Neuronal Completamente Conectada (FCNN), Figura 3.5.

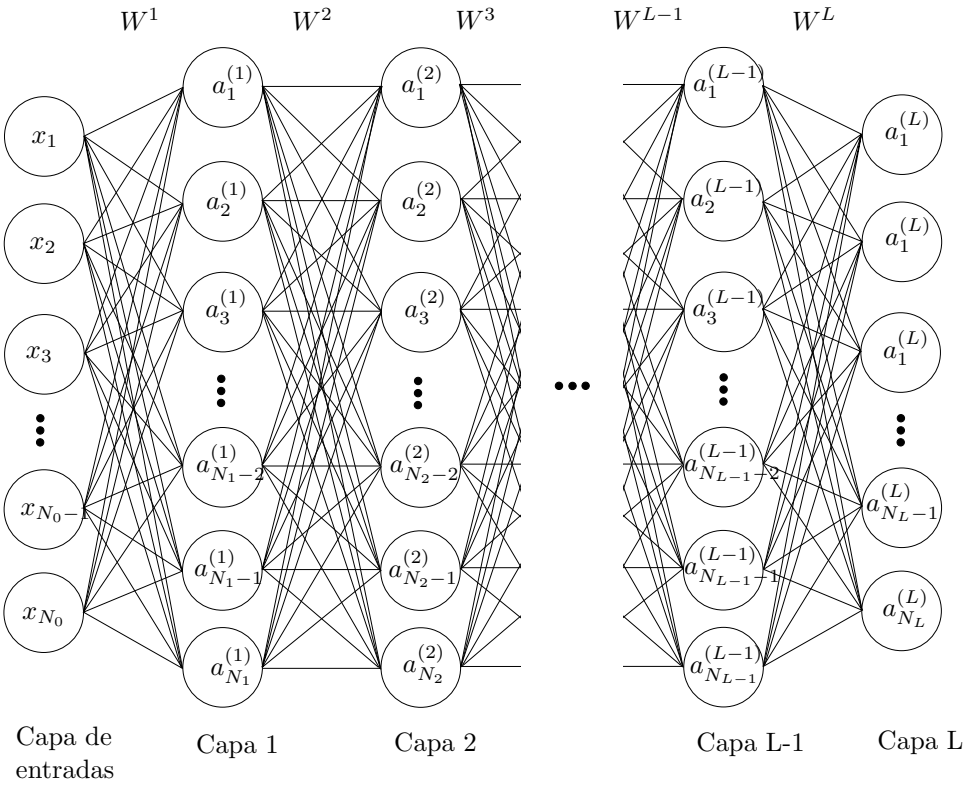


Figura 3.5: Esquema de la estructura de una Red Neuronal Completamente Conectada.

Las conexiones entre las neuronas de una capa y la siguiente se representan mediante una multiplicación de matriz por vector. En este contexto, el vector a^{l-1} representa las salidas de la capa anterior, y la matriz W está definida por los pesos w_{ji}, que conectan la neurona i de la capa actual con la neurona j de la capa siguiente.

$$W = \begin{bmatrix} w_{11} & w_{12} & w_{13} & ... & w_{1N} \\ w_{21} & w_{22} & w_{23} & ... & w_{2N} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ w_{M1} & w_{M2} & w_{M3} & ... & w_{MN} \end{bmatrix}; a = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_N \end{bmatrix}.$$

Entonces a^l se expresa como:

$$a^l = g(W^l \times a^{l-1}). \tag{3.25}$$

Donde g es una función de activación. Generalmente, a cada neurona se le suma un término b_j, llamado sesgo, que actúa de manera independiente a los valores de la

capa anterior. Para simplificar la notación, este término se puede representar como un peso adicional w_{M0} y un valor $a_0 = 1$, omitiendo así el sesgo en las fórmulas. Esta forma matricial se utilizará en adelante para mantener compactas las operaciones entre las capas.

3.2.2. Red Neuronal Convolucional

Una red neuronal convolucional (CNN) es una arquitectura en la que al menos una de sus capas utiliza la operación de convolución en lugar de unidades interconectadas [22]. Este tipo de redes está diseñado para trabajar con datos estructurados en forma de matrices, tales como series temporales unidimensionales (1-D), imágenes digitales bidimensionales (2-D) e incluso videos tridimensionales (3-D) [56]. Además, las CNN pueden manejar múltiples canales en las entradas, como las diferentes bandas de color (rojo, verde y azul) en una imagen.

La convolución está definida de la siguiente manera:

$$f(x) * g(x) = \int_{-\infty}^{\infty} f(t)g(x-t)dt, \tag{3.26}$$

y en su forma discreta:

$$f[n] * g[n] = \sum_{i=-\infty}^{\infty} f[i]g[n-i]. \tag{3.27}$$

Estas ecuaciones se pueden extender a múltiples dimensiones, por ejemplo la convolución bidimensional discreta toma la siguiente forma:

$$f[n,m] * g[n,m] = \sum_{j=-\infty}^{\infty} \sum_{i=-\infty}^{\infty} f[i,j]g[n-i,m-j]. \tag{3.28}$$

El caso más común de uso para las redes neuronales convolucionales es el procesamiento de imágenes 2D con múltiples canales. Una capa convolucional consta de una entrada $X[i,j,k]$ y un conjunto de kernels, o filtros, de menor tamaño que esta $W[a,b,k]$. En esta notación, i y j recorren las dimensiones espaciales de la imagen, y k representa el canal. Para W, a y b recorren las dimensiones del filtro. Los filtros

se convolucionan con la entrada, y los resultados $Z[i,j]$ de cada convolución pasan por una función de activación no lineal, produciendo así la salida de la capa (Figura 3.6).

La operación de la convolución para esta red neuronal queda entonces:

$$Z[i,j] = \sum_{a=-n_a/2}^{n_a/2} \sum_{b=-n_b/2}^{n_b/2} \sum_{k=1}^{n_k} W[a,b]X[i+a, j+b], \tag{3.29}$$

donde n_a y n_b son el tamaño del filtro, y n_k es el número de canales. Esta operación se centra en el dato $[i,j]$ y se aplica padding si el filtro cae en los bordes de la imagen. La salida de la capa tendrá un número de canales igual al número de filtros aplicados a la entrada. El filtro W puede recorrer los datos de entrada con un paso mayor a uno, este hiperparámetro es conocido como stride.

Es común añadir una capa adicional llamada pooling después de las unidades no lineales. Esta capa realiza un resumen estadístico en un punto con respecto a sus vecinos [16]. La operación de pooling recorre la entrada por ventanas y retorna un único valor para cada posición que cubre la ventana. Esto no solo reduce las dimensiones del resultado, sino que también proporciona invarianza a la traslación, lo que significa que se da más importancia a la presencia de una característica que a su posición exacta o rotación [22]. Generalmente, se añade algún otro tipo de red neuronal al final de las capas convolucionales para ejecutar la tarea con base en las características aprendidas (Figura 3.6).

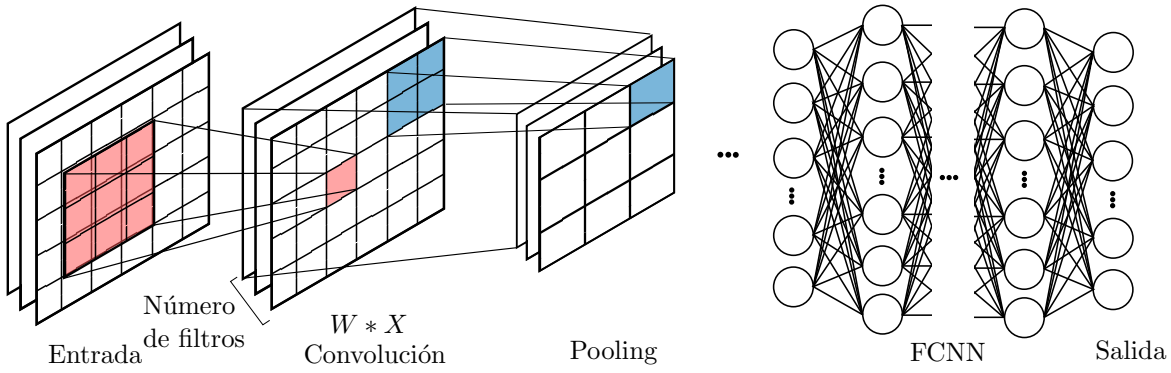


Figura 3.6: Esquema de la estructura de una Red Neuronal Convolucional. Adaptado de [50].

Los parámetros entrenables en esta arquitectura son los pesos que definen cada uno de los filtros. Al igual que en las redes neuronales Feedforward, los pesos de

los kernels se inicializan aleatoriamente y se actualizan iterativamente mediante el cálculo del gradiente para minimizar la función de error [56].

3.2.3. Red Neuronal Recurrente

Las redes neuronales recurrentes (RNNs) se aplican en problemas donde los datos tienen forma de secuencia discreta. Estas redes tienen la característica de incorporar la información de los pasos anteriores para calcular los pasos siguientes, un proceso conocido como recurrencia [22]. A diferencia de las redes neuronales anteriores, que asumían que cada entrada se muestreaba independientemente a partir de la misma distribución de probabilidad, las RNNs consideran que los datos dentro de una secuencia son dependientes entre sí [56].

Las RNNs están compuestas por una entrada en forma de secuencia x_t, una secuencia de salidas yt, y un estado oculto ht en cada paso del tiempo [39]. Estos estados ocultos están conectados mediante matrices de pesos U_{hx}, W_{hh} y V_{yh} (Figura 3.7).

Estas unidades se expresan matemáticamente como:

$$h_t = g_h(W_{hh}^T h_{t-1} + U_{hx}^T x_t), \tag{3.30}$$

$$y_t = g_y(V_{yh}^T h_t). \tag{3.31}$$

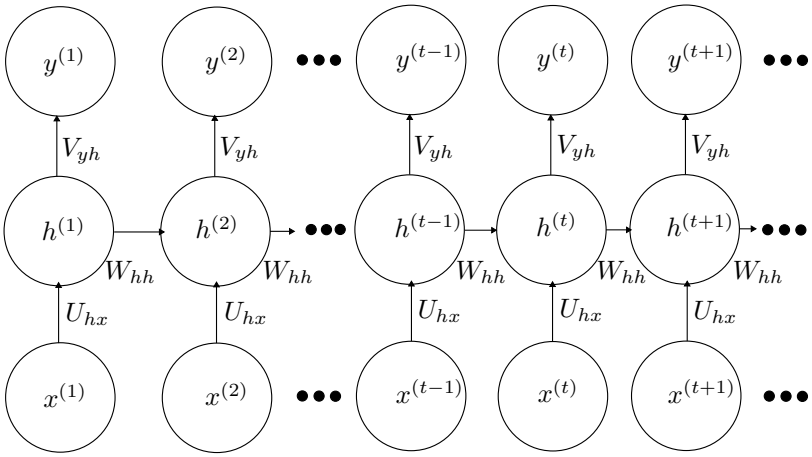


Figura 3.7: Esquema de la estructura de una Red Neuronal Recurrente. Adaptado de [16].

El cálculo de cada estado oculto h_t toma en cuenta la entrada x_t correspondiente para ese paso y el estado oculto del paso anterior h_{t-1}. La salida y_t está en función del valor de h_t. Tanto h_t como y_t pasan por una función de activación no lineal. Dependiendo de la tarea que se desee resolver con la red, se puede optar por obtener la salida y_t en cada paso de tiempo, o bien, una salida final después de procesar toda la secuencia [56].

Los parámetros entrenables para esta arquitectura son los pesos U_{hx}, W_{hh} y V_{yh}. Estas matrices de pesos se comparten entre todos los pasos t. Por ejemplo, todas las entradas $x_0, ..., x_t$ se multiplicarán por la misma matriz U_{hx}. Los pesos se manejan en forma de matrices para facilitar su operación sobre los estados. En cada paso del tiempo, la entrada x_t puede tener K componentes y puede haber MM estados ocultos h_t. Así, la matriz U_{hx} será de tamaño $M \times K$, la matriz W_{hh} de $M \times M$ y V_{yh} de $K \times M$.

Para calcular el gradiente de la función de error con respecto a los pesos, se utiliza una variante del algoritmo de retropropagación llamada **Back Propagation Through Time** (BPTT). Su desarrollo matemático se presenta a continuación:

y_t es la salida de la última capa.

$z_t = W_{yh}^T h_t$.

$1 \leq t \leq T$, es el rango de t.

El error se calcula a partir de la capa de salidas y_t de la misma manera que se hace en las capas de una red Feed Forward. El error total será la suma del error de y_t en cada paso del tiempo:

$$\frac{\partial E}{\partial V_{yh}} = \sum_{t=1}^{T} \frac{\partial E_t}{\partial V_{yh}}, \tag{3.32}$$

$$\frac{\partial E_t}{\partial V_{yh}} = \frac{\partial E_t}{\partial y_t} \frac{\partial y_t}{\partial z_t} \frac{\partial z_t}{\partial V_{yh}}, \tag{3.33}$$

$$\frac{\partial E}{\partial V_{yh}} = \sum_{t=1}^{T} \left(\frac{\partial E_t}{\partial y_t} \frac{\partial y_t}{\partial z_t} \frac{\partial z_t}{\partial V_{yh}} \right), \tag{3.34}$$

donde $\frac{\partial E_t}{\partial y_t}$ es la derivada de la función de costo, $\frac{\partial y_t}{\partial z_t}$ es la derivada de la no linealidad y $\frac{\partial z_t}{\partial V_{yh}}$ es la derivada del producto de la matriz de pesos por los estados ocultos en ese paso del tiempo con respecto a la matriz V_{yh}.

Para las otras dos matrices se tiene:

$$\frac{\partial E_t}{\partial W_{hh}} = \sum_{t=1}^{T} \left(\frac{\partial E_t}{\partial y_t} \frac{\partial y_t}{\partial z_t} \frac{\partial z_t}{\partial h_t} \frac{\partial h_t}{\partial W_{hh}} \right), \tag{3.35}$$

$$\frac{\partial E_t}{\partial U_{hx}} = \sum_{t=1}^{T} \left(\frac{\partial E_t}{\partial y_t} \frac{\partial y_t}{\partial z_t} \frac{\partial z_t}{\partial h_t} \frac{\partial h_t}{\partial U_{hx}} \right). \tag{3.36}$$

Las derivadas parciales $\frac{\partial h_t}{\partial W_{hh}}$ y $\frac{\partial h_t}{\partial U_{hx}}$ tienen dependencia con los pasos en el tiempo anteriores. Tomando como ejemplo el primer caso:

$$\frac{\partial h_t}{\partial W_{hh}} \rightarrow \frac{\partial h_t}{\partial W_{hh}} + \frac{\partial h_t}{\partial h_{t-1}} \frac{\partial h_{t-1}}{\partial W_{hh}} + \frac{\partial h_t}{\partial h_{t-1}} \frac{\partial h_{t-1}}{\partial h_{t-2}} \frac{\partial h_{t-2}}{\partial W_{hh}} + ... \tag{3.37}$$

Esto se reescribe como:

$$\frac{\partial h_t}{\partial W_{hh}} \rightarrow \sum_{r=1}^{T} \frac{\partial h_t}{\partial h_r} \frac{\partial h_r}{\partial W_{hh}}. \tag{3.38}$$

Finalmente, se substituye (3.38) en (3.36),

$$\frac{\partial E_t}{\partial W_{hh}} = \sum_{t=1}^{T} \left(\frac{\partial E_t}{\partial y_t} \frac{\partial y_t}{\partial z_t} \frac{\partial z_t}{\partial h_t} \sum_{t=1}^{T} \left(\frac{\partial h_t}{\partial h_r} \frac{\partial h_r}{\partial W_{hh}} \right) \right), \tag{3.39}$$

y el caso análogo para $\frac{\partial h_t}{\partial U_{hx}}$:

$$\frac{\partial E_t}{\partial U_{hx}} = \sum_{t=1}^{T} \left(\frac{\partial E_t}{\partial y_t} \frac{\partial y_t}{\partial z_t} \frac{\partial z_t}{\partial h_t} \sum_{t=1}^{T} \left(\frac{\partial h_t}{\partial h_r} \frac{\partial h_r}{\partial U_{hx}} \right) \right). \tag{3.40}$$

RNNs Bidireccionales

En las RNNs convencionales, la salida y_t considera únicamente la información x_t correspondiente al paso actual y a los valores anteriores $x_1, x_2, ..., x_{t-1}$. Para casos en los que y_t depende de toda la secuencia, se introducen las redes neuronales recurrentes bidireccionales [16]. En estas redes, se añade el cálculo del estado oculto posterior $\overleftarrow{h_t}$, obteniendo así una representación de la secuencia en ambas direcciones:

$$\overrightarrow{h_t} = g_h(W^T_{hh_f} h_{t-1} + U^T_{hx_f} x_t), \tag{3.41}$$

$$\overleftarrow{h_t} = g_h(W^T_{hh_b} h_{t+1} + U^T_{hx_b} x_t), \tag{3.42}$$

$$h_t = [\overrightarrow{h_t}; \overleftarrow{h_t}]. \tag{3.43}$$

La salida y_t se obtiene de manera similar a las RNNs convencionales, usando esta representación concatenada del estado oculto.

LSTM y GRU

Para abordar la limitación de las RNNs en cuanto a la retención de dependencias a largo plazo, se han desarrollado arquitecturas que reemplazan la unidad recurrente simple. Las dos unidades más populares de este tipo son las Long Short-Term Memory (LSTM) y las Gated Recurrent Units (GRU).

Las unidades Long Short-Term Memory (LSTM) [Hochreiter,] (Figura 3.8) incluye un estado adicional c_t que actúa como una memoria de largo plazo. Su estructura interna consta de cuatro componentes clave, operando de la siguiente manera [8]:

1. Compuerta de Olvido: Se introduce el estado oculto de la unidad anterior h_{t-1} y la entrada actual x_t a una función sigmoide para decidir qué información se debe descartar o mantener de la memoria c_{t-1}.
2. Compuerta de Entrada: Mediante otra función sigmoide, se decide qué información se integrará a la memoria c_{t-1}. Esta información se normaliza con una función *tanh* antes de sumarse a la memoria.

3. Estado de Memoria: La nueva memoria c_t se calcula combinando la memoria anterior con la información nueva ponderada por la puerta de entrada.

4. Compuerta de Salida: La salida y_t se calcula como la normalización del estado de memoria c_t mediante una función $tanh$, y se combina con la entrada actual y el estado oculto a través de una función sigmoide.

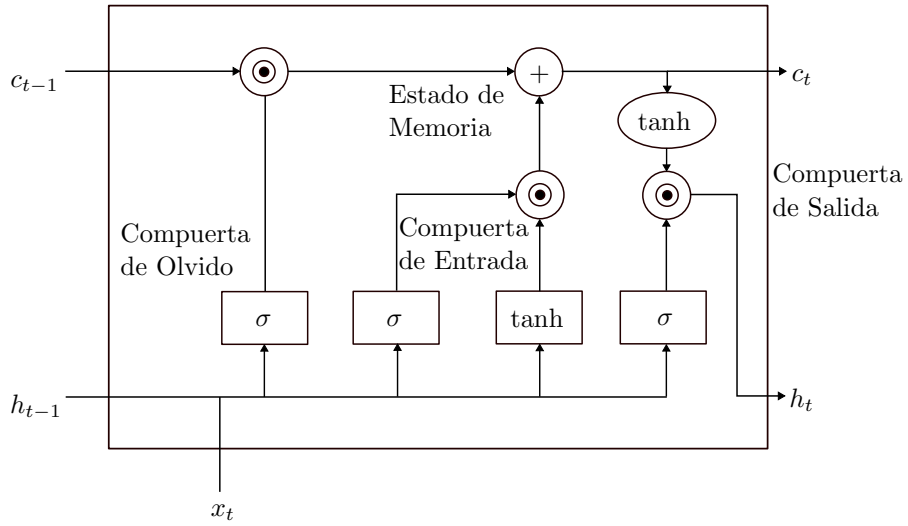


Figura 3.8: Esquema de la estructura de una unidad LTSM. Adaptado de [56].

Las Gated Recurrent Units (GRU) [Cho, 2014] (Figura 3.9) presentan una estructura similar a las LSTM pero sin el estado de memoria adicional c_t. La GRU utiliza dos compuertas principales [49]:

1. Compuerta de Reinicio: Controla la influencia del estado oculto anterior h_{t-1} en la entrada actualx_t.

2. Compuerta de Actualización: Determina la cantidad en la que se utiliza la entrada x_t para calcular el estado oculto.

3. Estado Provisional: El estado oculto provisional $\hat{h}_t$ se calcula con una función tanh en función de la entrada x_t y el estado anterior h_{t-1}, ponderado por la compuerta de reinicio.

4. Estado Actual: El estado oculto h_t que pasa a la siguiente unidad es una combinación del estado oculto anterior h_{t-1} y el estado provisional $\hat{h}_t$, ponderado por la compuerta de actualización.

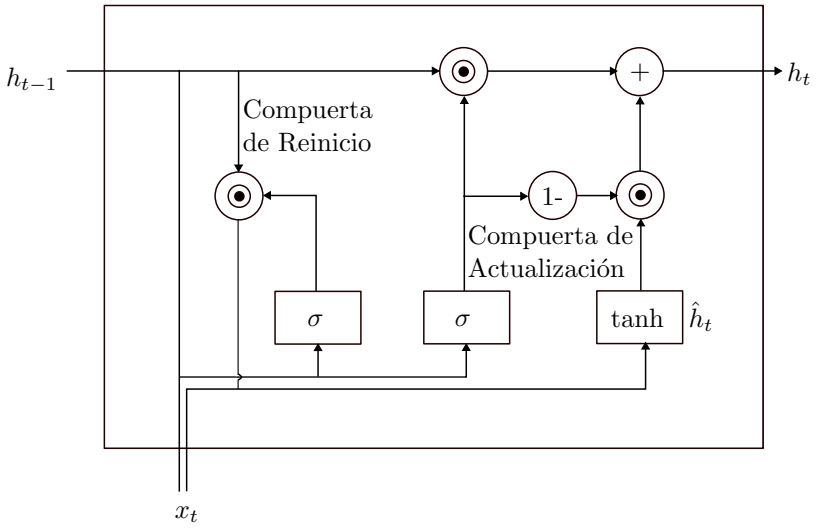


Figura 3.9: Esquema de la estructura de una GRU. Adaptado de [56].

Las RNNs descritas no se consideran profundas directamente debido a la falta de capas ocultas en su estructura. Para convertirlas en redes profundas, se pueden apilar múltiples capas recurrentes unas sobre otras [56]. En este enfoque, los estados ocultos de una capa se conectan a los estados ocultos de la capa siguiente, con el último estado oculto conectado a la salida de manera convencional. Alternativamente, se pueden usar múltiples estados h_t dentro de cada unidad, interconectados con los estados h_{t+1} de la siguiente unidad, siguiendo una arquitectura similar a una red neuronal Feedforward [40].

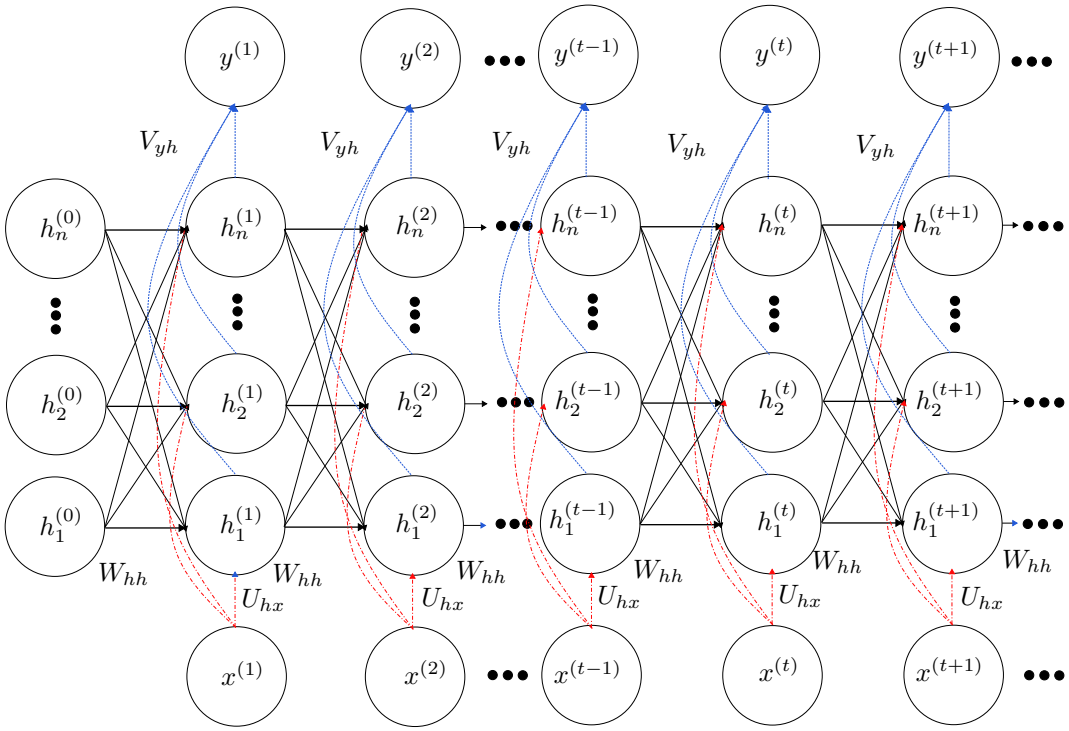


Figura 3.10: Esquema de la estructura de una Red Neuronal Recurrente Profunda. Adaptado de [40].

Capítulo 4

Metodología

El propósito de este capítulo es presentar la secuencia de métodos seguida para llegar a la predicción del registro sónico. Se describirá el análisis de los datos, el tratamiento al que estuvieron sujetos, cómo se integran a la red neuronal y los detalles sobre esta.

Como se mencionó en la introducción los métodos de Inteligencia Artificial para la predicción de registros geofísicos han dado buenos resultados y se han vuelto populares en los últimos años [53, 41, 28, 57]. Los más sencillos de estos usan Redes Neuronales Completamente Conectadas, mientras que los más exitosos hace combinaciones entre estas con Redes Neuronales Convolucionales y Redes Neuronales Recurrentes.

4.1. Datos

Los datos con los que se trabajó en este estudio fueron los registros de pozo pertenecientes a un campo petrolero en Tabasco, México. Los registros disponibles varían de un pozo a otro, y estos incluyen al caliper, rayos gamma, potencial espontáneo, resistividad, porosidad de neutrones, densidad y el registro sónico. Las mediciones se tomaron cada 15 centímetros. Los datos de todos los pozos fueron previamente corregidos por el proveedor, de manera que se puede trabajar con ellos directamente.

Dado que el objetivo de la red neuronal es predecir el registro sónico, solo se consideraron los pozos que contaban con este registro. De todos los pozos proporcionados, únicamente seis tienen el registro sónico. Cuatro de estos presentan registros en común, además del sónico, los cuales son los registros de resistividad, rayos gamma, porosidad de neutrones y densidad. Por lo que para el entrenamiento se cuenta estos pozos, nombrados por confidencialidad A, B, C y D, y se presentan en la Figura 4.1.

4.1.1. Dependencia entre registros

En el capítulo dos se habló de las propiedades físicas de la formación que asocian al registro sónico con los demás registros, así como relaciones empíricas que los unen matemáticamente. Se puede corroborar la dependencia entre el registro sónico y los

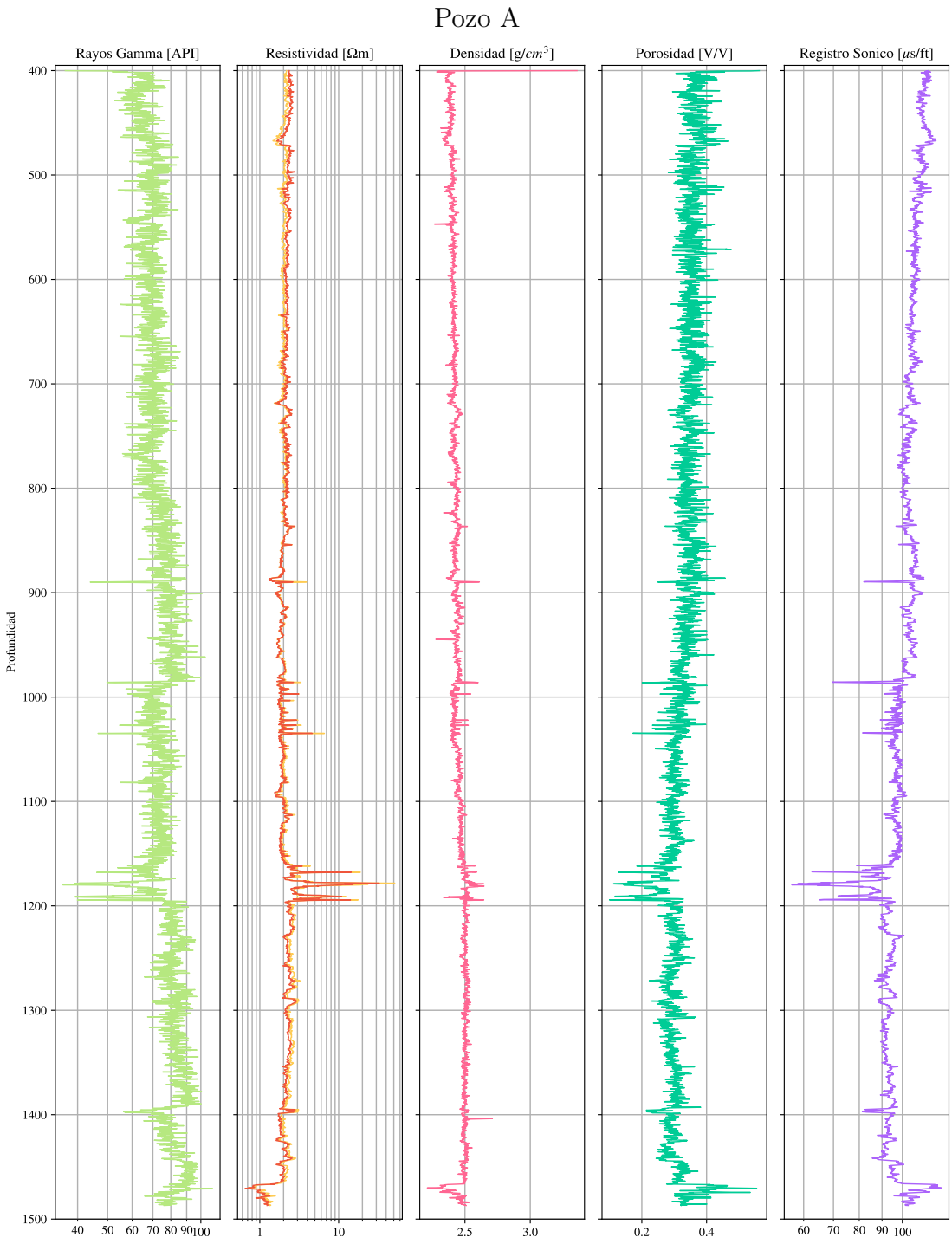


Figura 4.1: Pozos A - D. Se muestran únicamente los registros relevantes al entrenamiento.

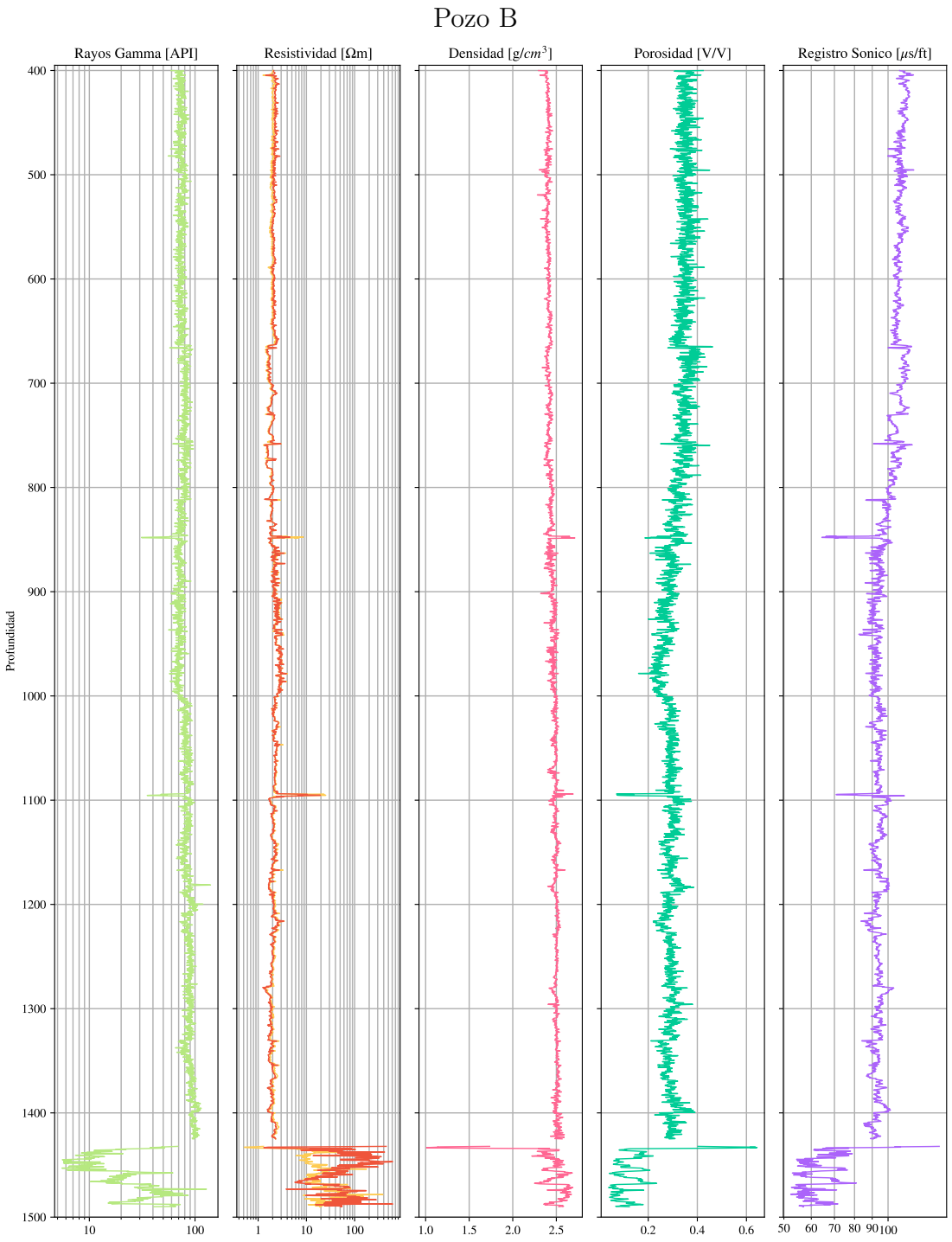


Figura 4.1: Pozos A - D. Se muestran únicamente los registros relevantes al entrenamiento (continuación).

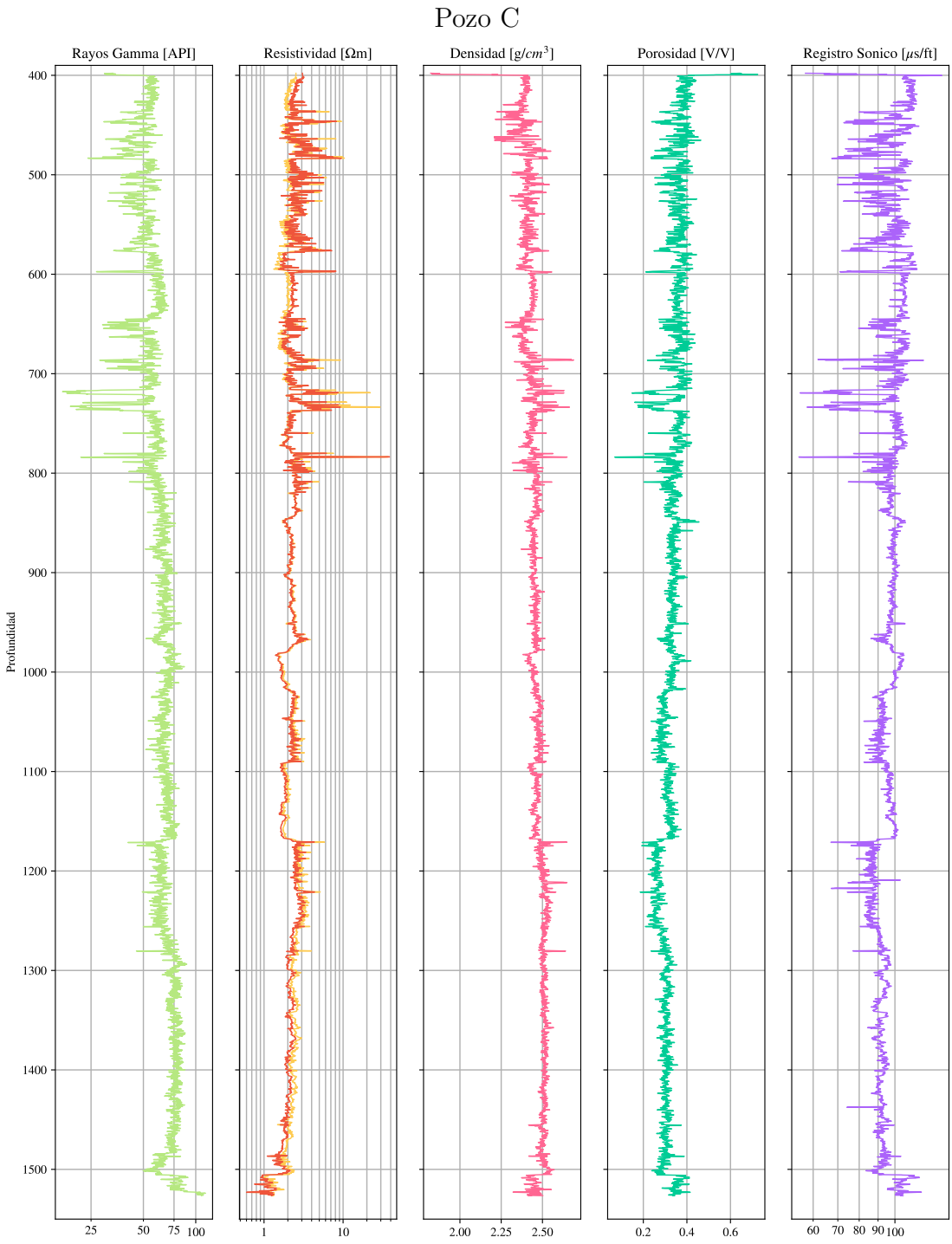


Figura 4.1: Pozos A - D. Se muestran únicamente los registros relevantes al entrenamiento (continuación).

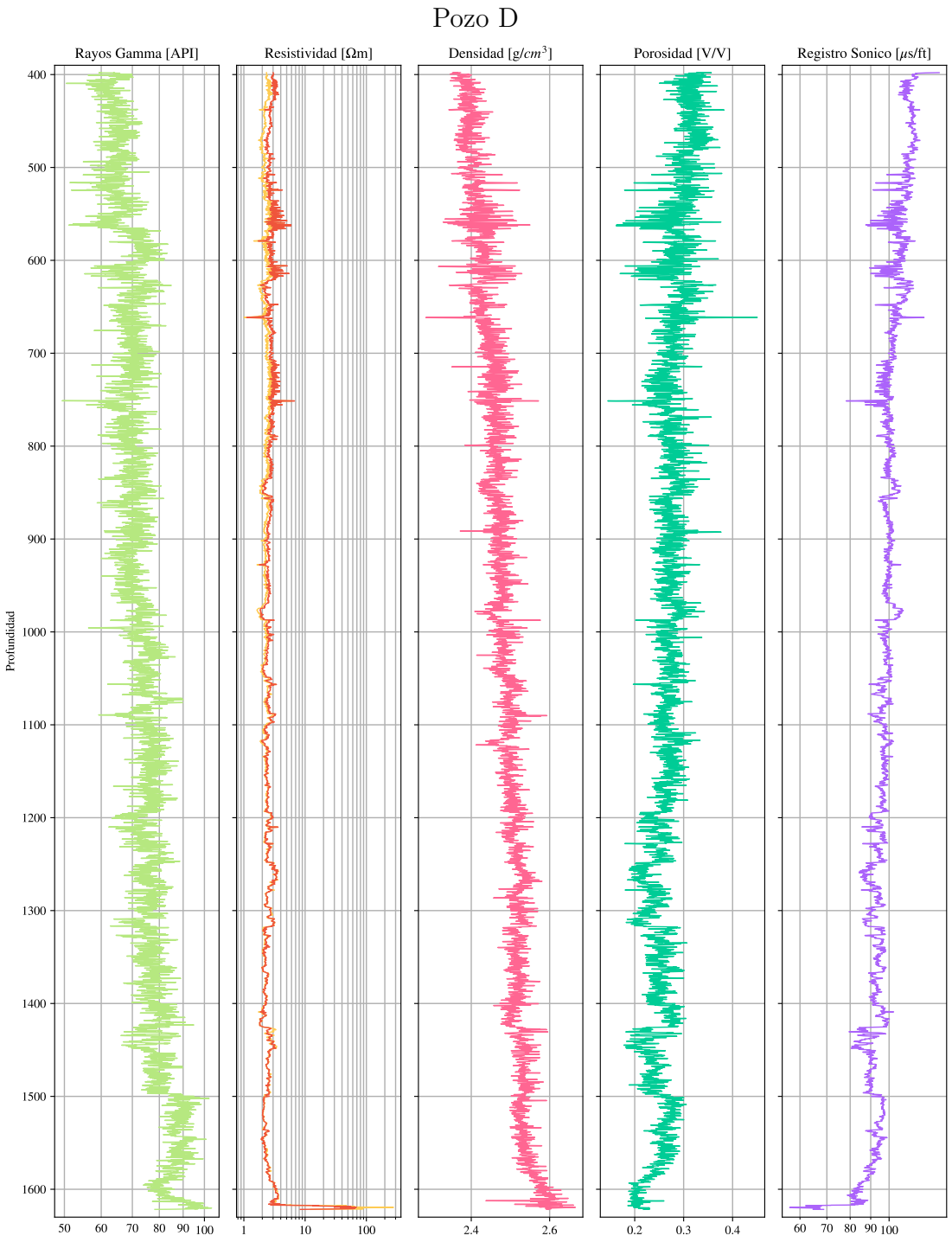


Figura 4.1: Pozos A - D. Se muestran únicamente los registros relevantes al entrenamiento (continuación).

otros registros directamente con los datos reales. Esta se ve reflejada en los diagramas de dispersión que se muestran en la Figura 4.2 donde se notan las dependencias no lineales que presentan los registros.

4.2. Tratamiento de datos

Antes de introducir los datos a la red neuronal se le hizo un tratamiento adicional al procesamiento geofísico previo. En los registros de pozos se debe tener cuidado removiendo valores atípicos, ya que aunque algunos datos estén estadísticamente alejados de la media, pueden ser producto de efectos geológicos de interés y no de un error de los instrumentos. El criterio que se siguió para remover o conservar a los datos aparentemente atípicos fue monitorear si el comportamiento atípico es congruente para todos los registros.

Otro proceso esencial previo al entrenamiento es el escalamiento de los datos. Debido a que los datos de entrada tienen diferentes unidades y rangos, es necesario escalarlos para que todos contribuyan de manera equitativa al entrenamiento de la red neuronal. Comúnmente se hace una estandarización, una normalización o una transformación.

En este caso, se optó por una transformación logarítmica. Esto debido a que hay datos muy alejados de la media en los registros de resistividad y esta transformación reduce el sesgo de las distribuciones. Para mantener la coherencia de las unidades de los datos, se aplicó la misma transformación al resto de los registros, 4.3.

Generalmente, cuando se ocupa una red neuronal, los datos de entrada se tienen que introducir en el mismo orden y deben tener las mismas dimensiones. Los registros tienen dimensiones distintas y si se toman completos la secuencia de datos sería demasiado larga. Una secuencia de datos muy larga puede causar problemas en la estabilidad del entrenamiento y consumir más recursos computacionales. Debido a esto se dividieron los pozos en ventanas en profundidad de 200 datos con un traslape de cuatro datos entre ellas, donde los últimos cuatro datos de una ventana son los primeros de la siguiente. Finalmente, para evitar que el entrenamiento esté sesgado por el orden en que se introducen los datos, las ventanas se reacomodaron aleatoriamente.

4.2.1. Datos de entrenamiento, validación y prueba

Para evaluar la capacidad de generalización del modelo entrenado, se dividieron los datos en tres subconjuntos: entrenamiento, validación y prueba.

El conjunto de entrenamiento, que representa el 70 % de los datos, se utiliza para ajustar los parámetros de la red neuronal durante el proceso de entrenamiento. Es decir, a partir de este conjunto, la red .ªprende.ª establecer las relaciones entre las variables de entrada y salida.

El conjunto de validación, que representa el 15 % de los datos, se emplea para supervisar que las predicciones de los modelos concuerden con los datos reales. Es decir, este no interviene en el ajuste de los parámetros de la red, pero nos da información sobre su desempeño en datos sobre los cuales no se entrena directamente. Se busca optimizar el costo de este conjunto de datos, por lo que los hiperparámetros de la red se modifican en función de este.

El conjunto de prueba, que representa el 15 % restante de los datos, se reserva para una evaluación final del modelo. Estos datos no se utilizan ni en el entrenamiento ni en la validación, por lo que proporcionan una estimación imparcial del desempeño del modelo en datos completamente nuevos.

La selección de los tres grupos de datos fue aleatoria, en la Figura 4.5 se presenta el muestreo de los datos originales, mientras que en la 4.4 se presentan las distribuciones de cada conjunto. La proporción de los datos que cae dentro de cada conjunto se considerando la cantidad limitada de pozos disponibles, por lo que se buscó un balance entre tener una buena cantidad de datos de entrenamiento y tener suficientes datos de validación y prueba para que el modelo sea confiable.

4.3. Arquitectura de la red neuronal

La red neuronal propuesta consta de tres capas convolucionales, seguidas de dos capas recurrentes bidireccionales con unidades GRU y finalmente con una FCNN (Figura 4.6). La implementación de estas se hizo con la paquetería Tensorflow de Keras.

Las capas convolucionales tienen la función de extraer las características más importantes de los registros de entrada. Estas pueden operar sobre todas los registros

simultáneamente, similar a los canales de una imagen, lo cuál reduce el número de parámetros de la red y acelera el proceso de entrenamiento. Al apilar múltiples capas se busca que el algoritmo extraiga las relaciones complejas que guardan los registros. Los hiperparámetros utilizados para estas capas fueron 32, 64 y 64 kernels, con longitud en profundidad de 3, 7 y 14 datos respectivamente. La función de activación elegida fue ReLU y para mantener las dimensiones de los datos de entrada se usó un *paso* de 1, *padding* en los bordes y se omitieron las capas de *pooling*.

Las redes neuronales recurrentes se usaron para aprender las tendencias que tienen las características extraídas de las capas convolucionales con respecto a las demás, ya sean anteriores y posteriores. En estas se utilizaron Unidades GRU bidireccionales con 32 nodos en la primera capa y 64 la segunda. Su función de activación fue *tanh*.

Las últimas capas sirven para afinar los detalles de la regresión, tomando los resultados de las capas recurrentes y ajustándolos al registro sónico observado. Estas se diseñaron con 32, 64 y 1 nodo, respectivamente, y su función de activación fue *tanh*.

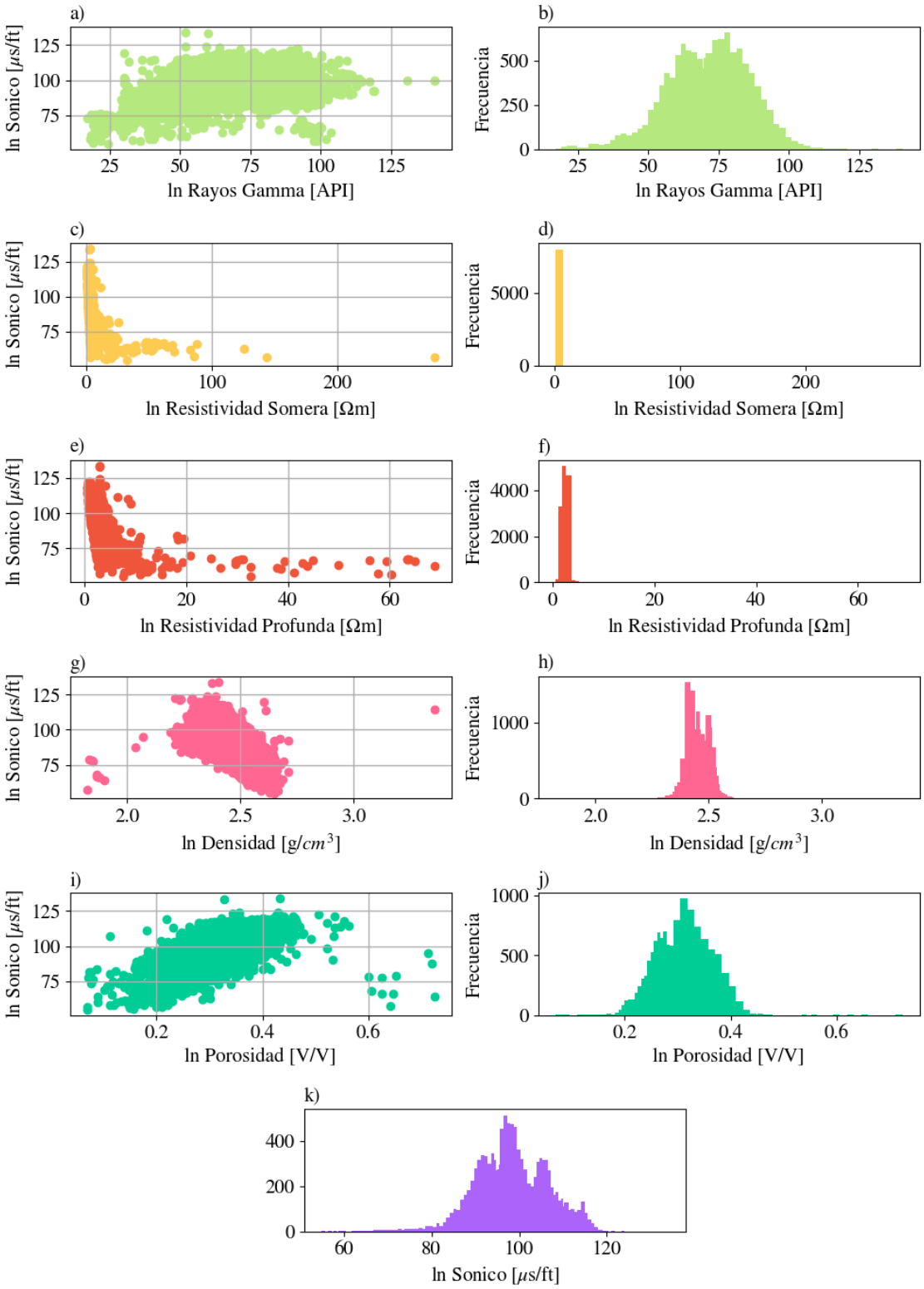


Figura 4.2: Diagramas de dispersión e histogramas de los registros de pozo.

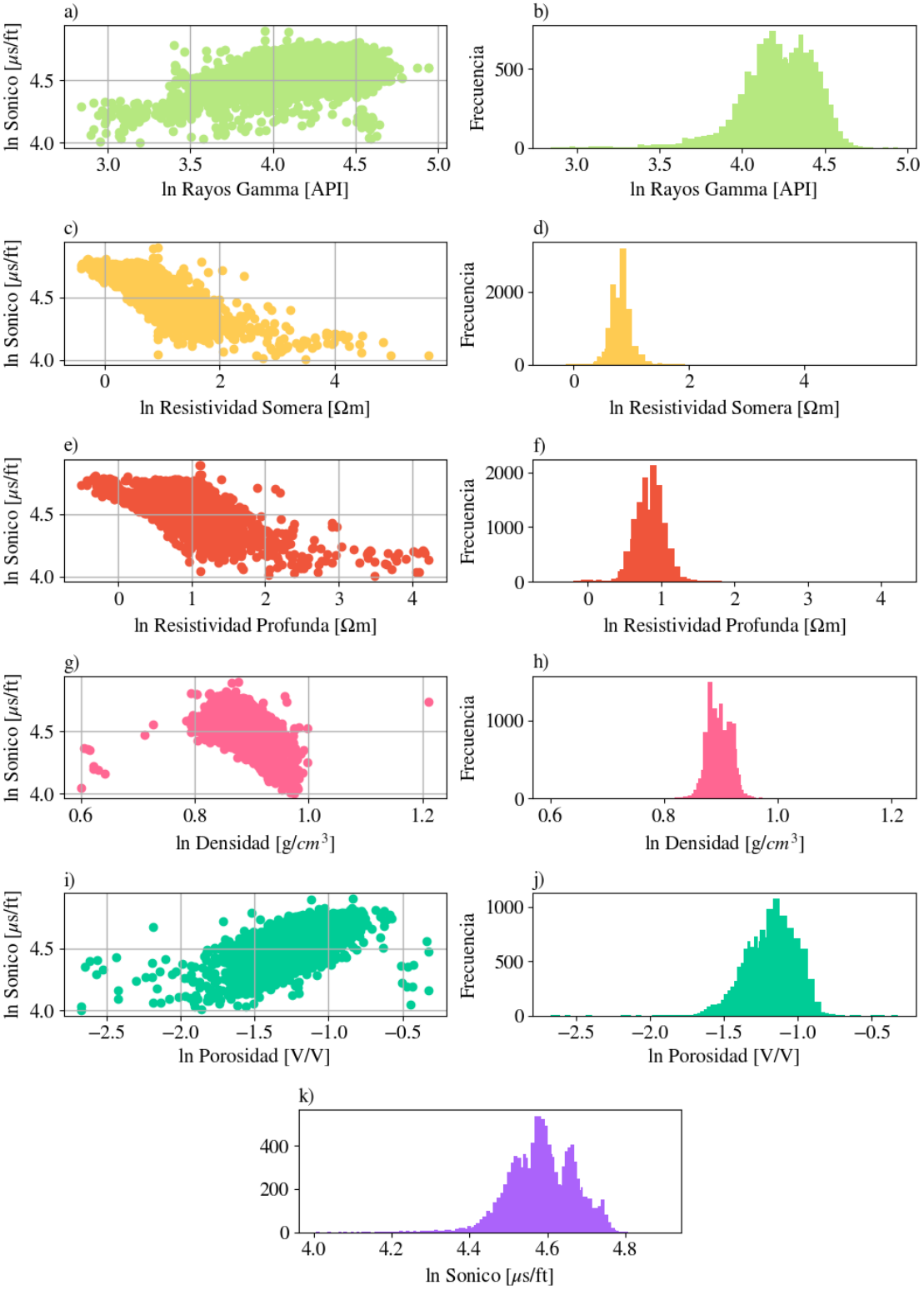


Figura 4.3: Diagramas de dispersión e histogramas de los datos de entrenamiento después de la transformación logarítmica.

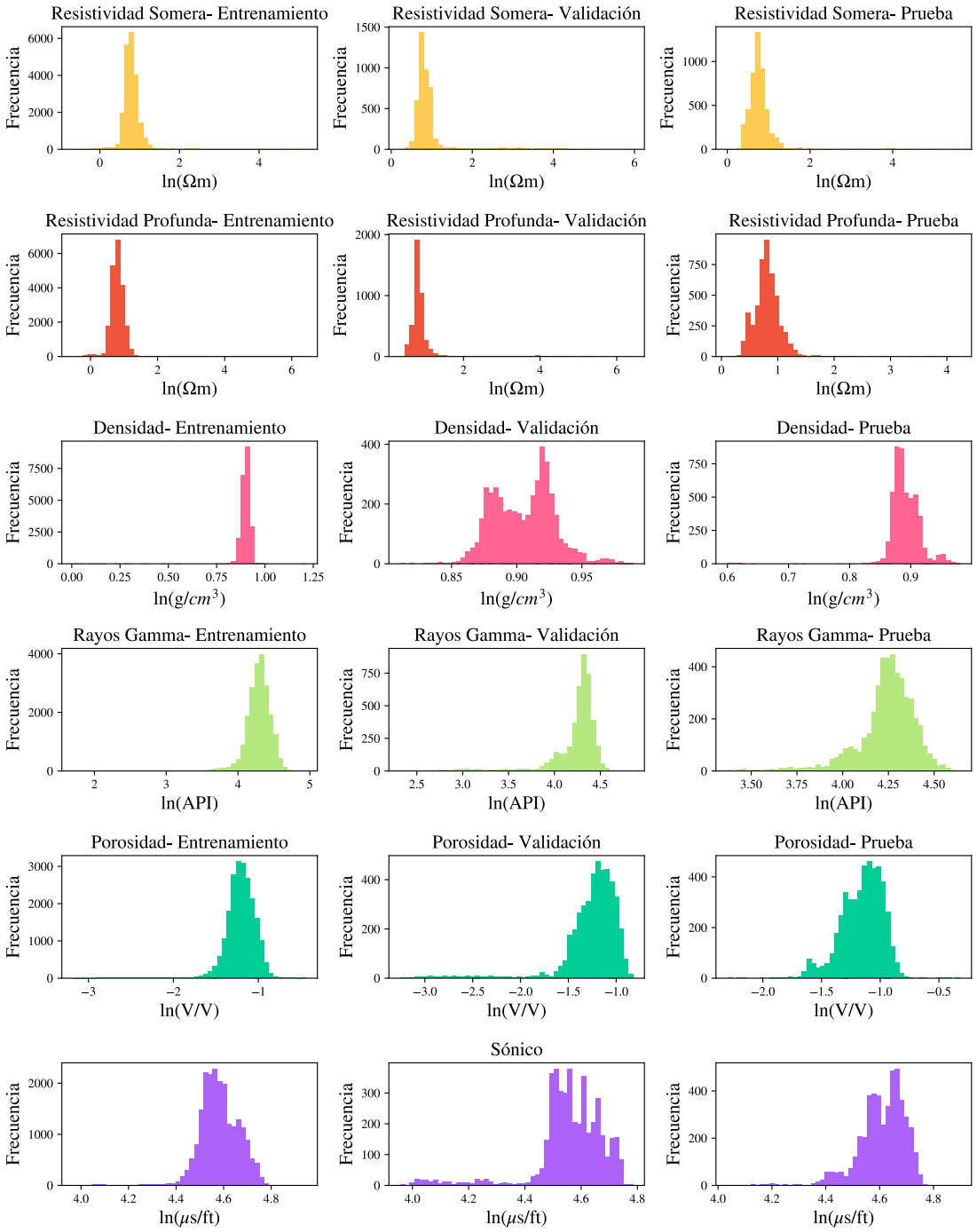


Figura 4.4: Histrogramas de los conjuntos de datos de entrenamiento, validación y prueba.

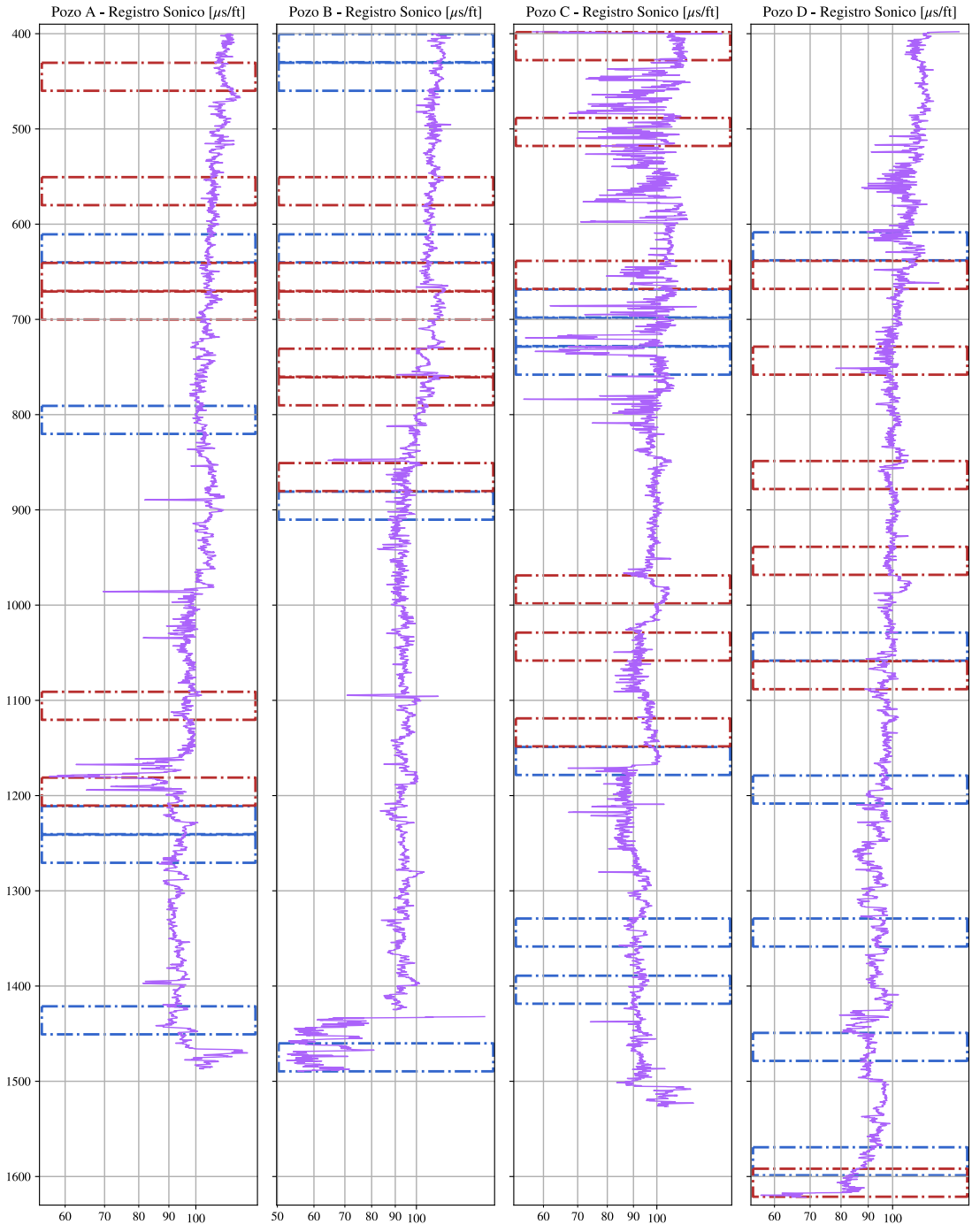


Figura 4.5: Registro sónico de los pozos A - D. Los rectángulos corresponden a los intervalos de validación (azul) y de prueba (marrón).

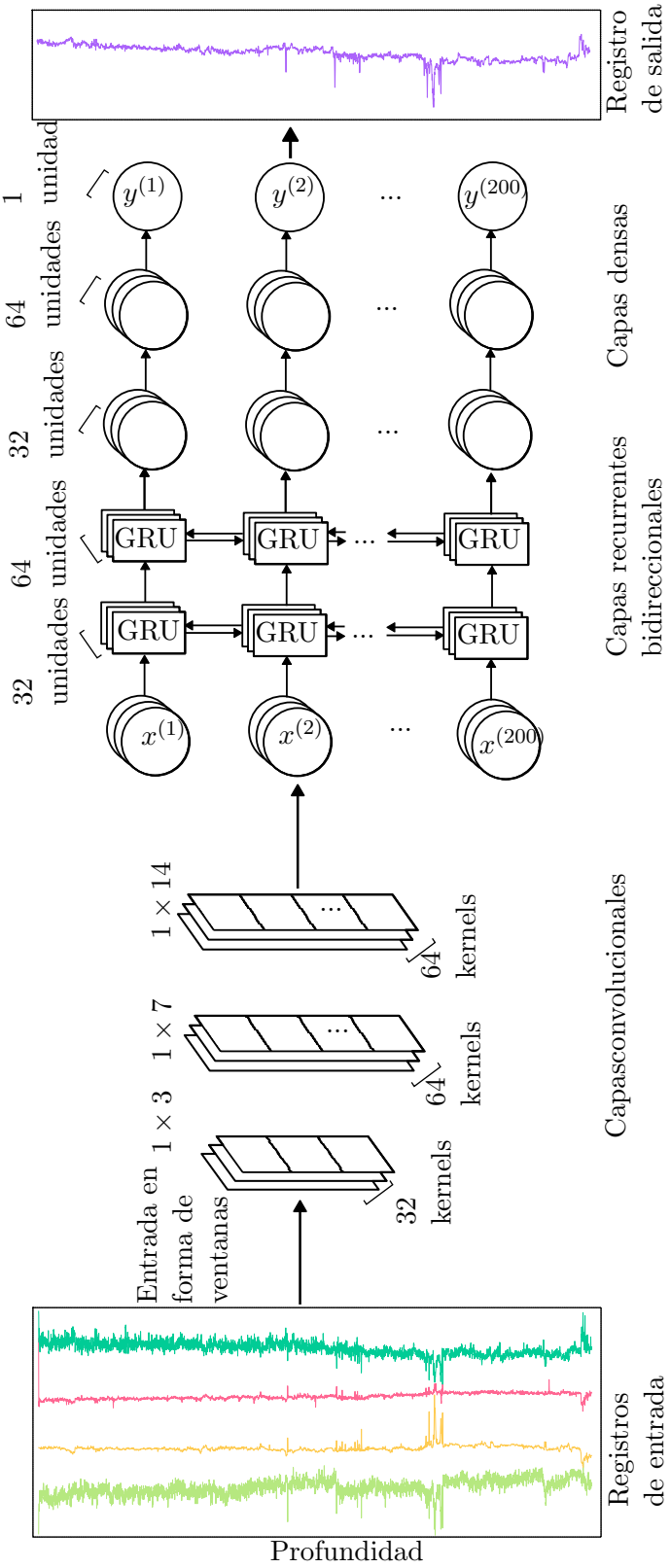


Figura 4.6: Arquitectura propuesta de la red neuronal para predecir el registro sonico.

Capítulo 5

Resultados

En este capítulo se presentan las predicciones del registro sónico para los datos de prueba descritos en la sección anterior. Además, se incluyen las curvas de costo en función de las épocas para ilustrar la convergencia durante el entrenamiento, así como algunos registros sónicos calculados para pozos del mismo campo petrolero en los que no se midió este registro.

Las predicciones se realizaron utilizando los registros de resistividad, rayos gamma, densidad y porosidad. Se entrenaron tres modelos diferentes, cada uno utilizando distintas combinaciones de estos registros para abordar situaciones en las que no se dispone de todos los datos.

5.1. Convergencia

Las curvas de costo contra épocas durante el entrenamiento se muestran en la Figura 5.1. En los tres casos, se encontró un modelo con el costo mínimo. Los costos de validación para los modelos seleccionados fueron: 8.5072×10^{-4}, 8.2446×10^{-4} y 1.4827×10^{-3} para los modelos 5R, 4R y 2R, respectivamente, mientras que los costos de entrenamiento fueron: 5.9188×10^{-4}, 5.8828×10^{-4} y 7.6999×10^{-4}.

5.2. Datos de prueba

Las Figuras 5.2 - 5.5 muestran las predicciones del registro sónico para los intervalos de prueba de los pozos utilizados en el entrenamiento. Estas figuras incluyen los registros de entrada, los resultados de los tres modelos y los datos originales. La similitud entre las predicciones y los datos reales se cuantificó utilizando la raíz del error cuadrático medio (RMSE) y el error absoluto medio porcentual (MAPE), cuyos valores se presentan en la Tabla 5.1. Esta tabla muestra los errores para las secciones de entrenamiento, validación y prueba. Además, las mismas métricas se calcularon únicamente para los intervalos de prueba, como se muestra en la Tabla 5.2.

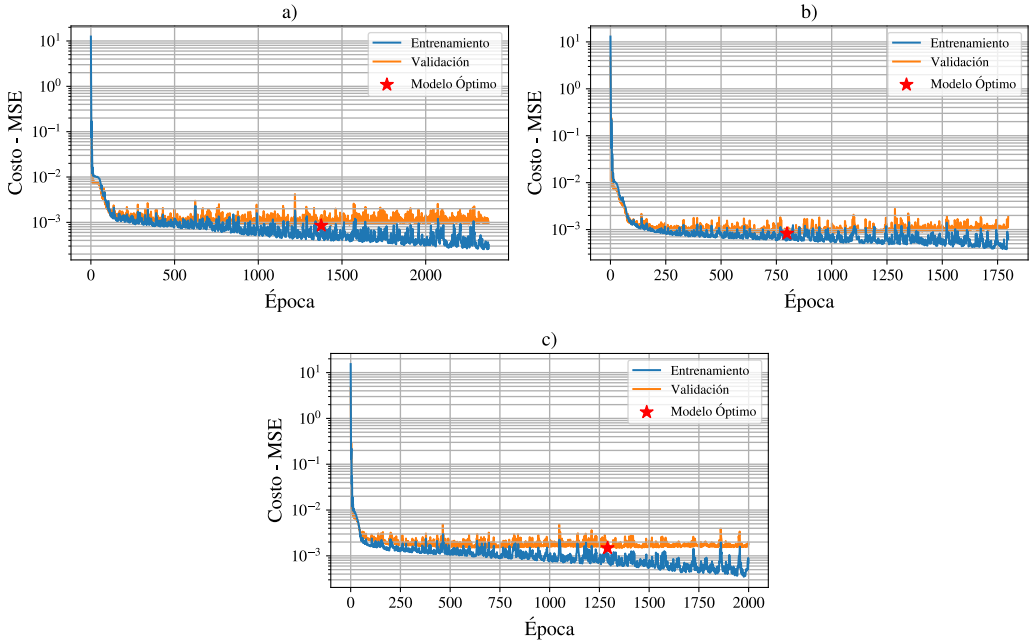


Figura 5.1: Curvas costo contra época de los modelos a)5R, b)4R y c)2R.

	Modelo 5R		Modelo 4R		Modelo 2R	
Nombre del Pozo	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE
A	1.2949	1.68 %	1.3632	1.86 %	1.3577	1.84 %
B	1.2170	1.55 %	1.3071	1.79 %	1.3757	1.97 %
C	1.4304	2.18 %	1.4834	2.34 %	1.6289	2.81 %
D	1.1862	1.44 %	1.1970	1.47 %	1.3701	1.93 %

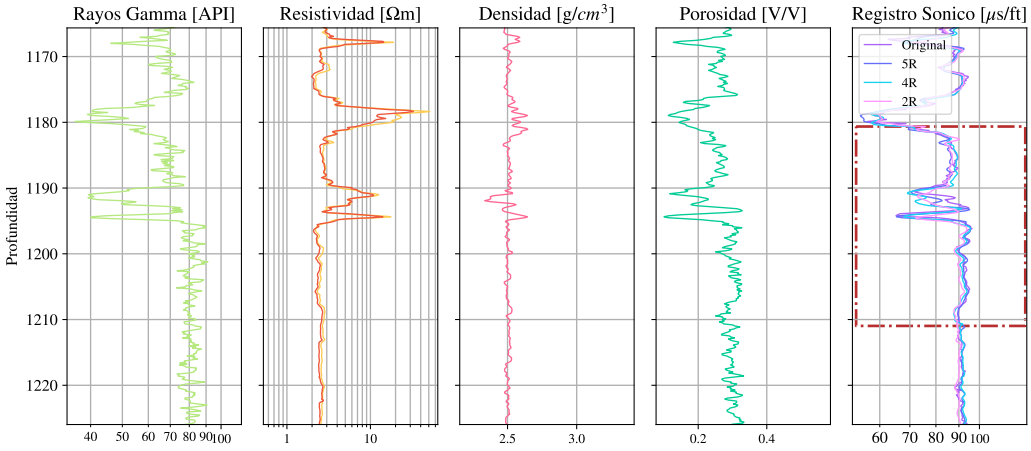
Tabla 5.1: Comparación entre las predicciones de los diferentes modelos para los datos de entrenamiento, validación y de prueba.

	Modelo 5R		Modelo 4R		Modelo 2R	
Nombre del Pozo	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE
A - Prueba	1.4267	2.03 %	1.4212	2.02 %	1.6251	2.61 %
B - Prueba	1.3483	1.79 %	1.4744	2.14 %	1.4693	2.18 %
C - Prueba	1.6672	2.98 %	1.7134	3.12 %	1.7995	3.44 %
D - Prueba	1.2911	1.84 %	1.2108	1.62 %	1.7004	3.02 %

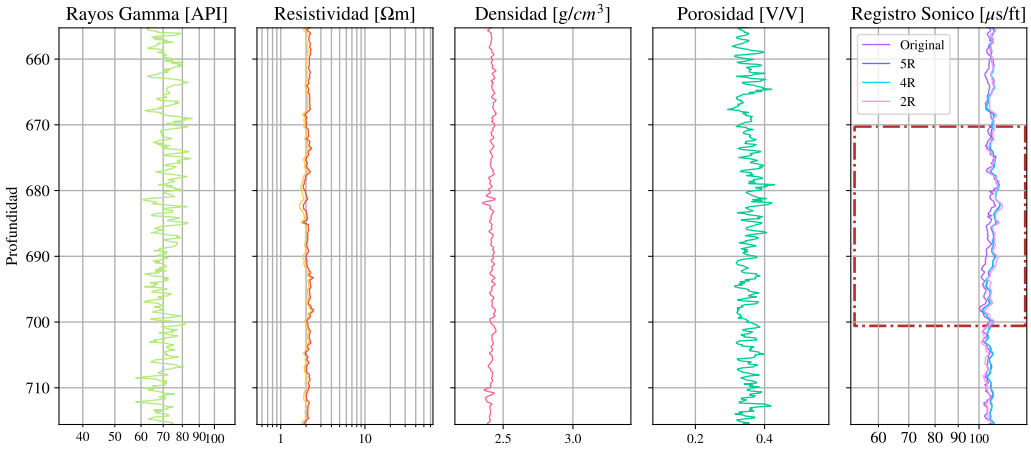
Tabla 5.2: Comparación entre las predicciones de los diferentes modelos únicamente para los datos de prueba.

5.3. Predicción en pozos

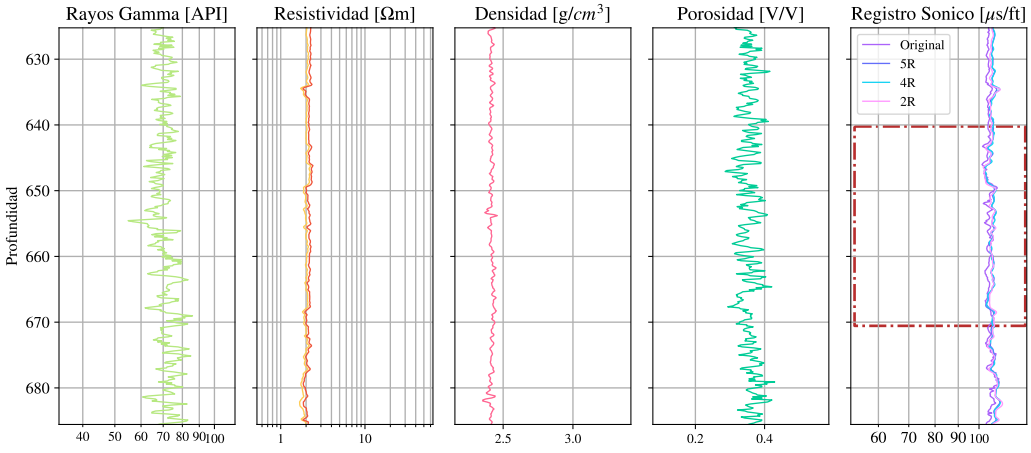
Los modelos entrenados se aplicaron para predecir el registro sónico en tres pozos adicionales dentro del mismo campo petrolero donde se llevó a cabo el entrenamiento, en los que el registro sónico no fue medido durante la adquisición de datos. La selección del modelo a utilizar se basó en los registros disponibles.



(a) Intervalo 1

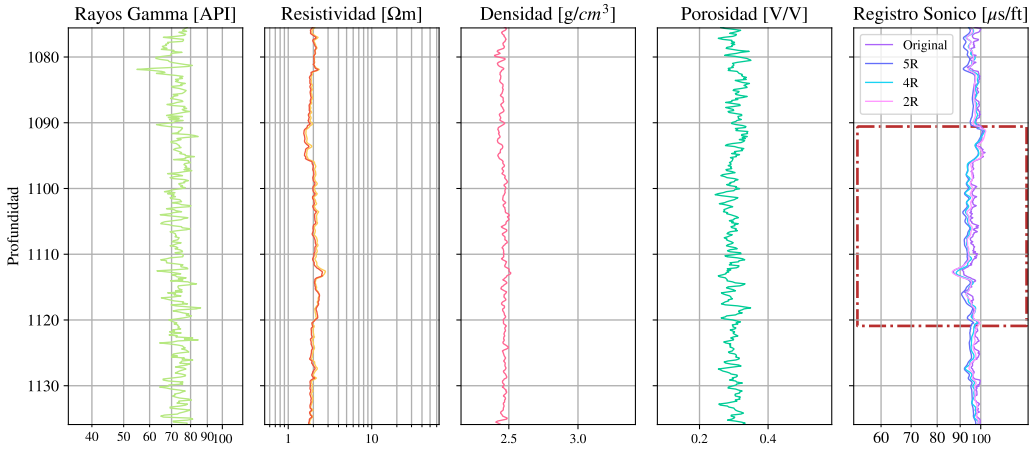


(b) Intervalo 2

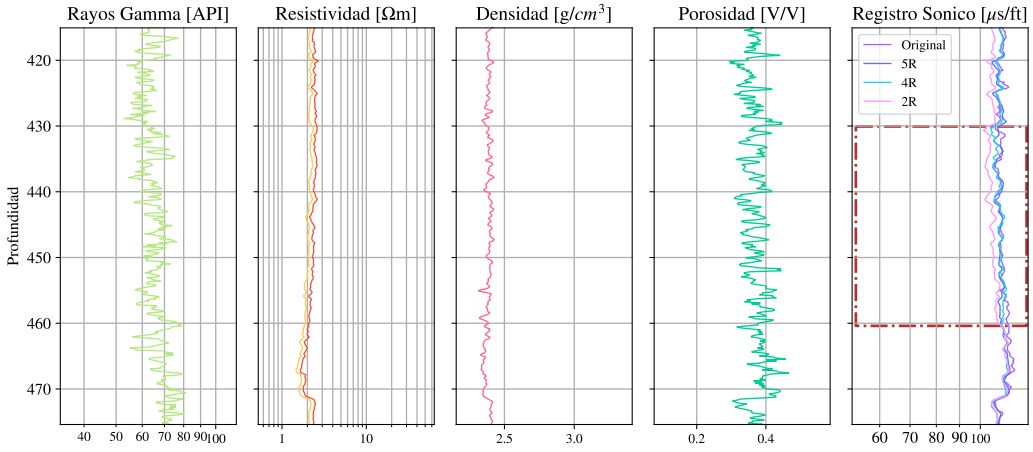


(c) Intervalo 3

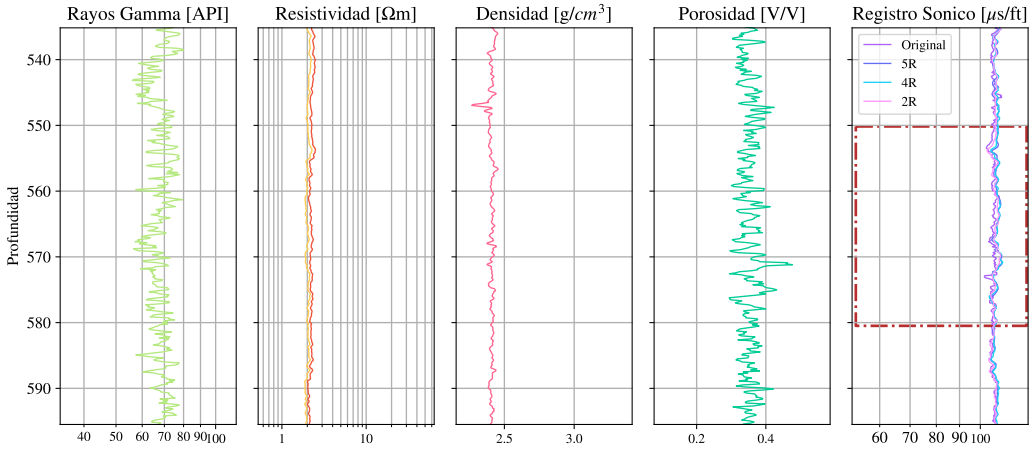
Figura 5.2: Predicciones del registro sónico en el Pozo A. La comparación del registro sónico original y los predichos se encuentra en el carril 5. Los rectángulos marrones corresponden a los intervalos de prueba.



(d) Intervalo 4

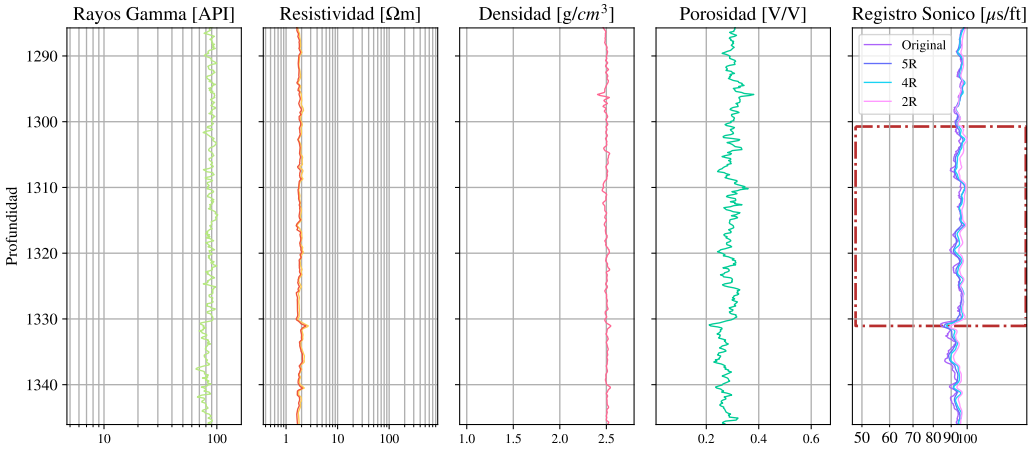


(e) Intervalo 5

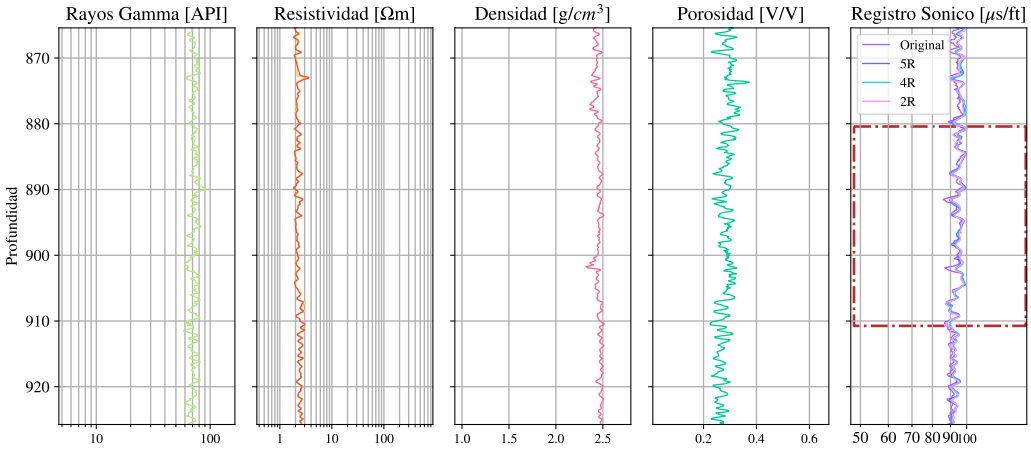


(f) Intervalo 6

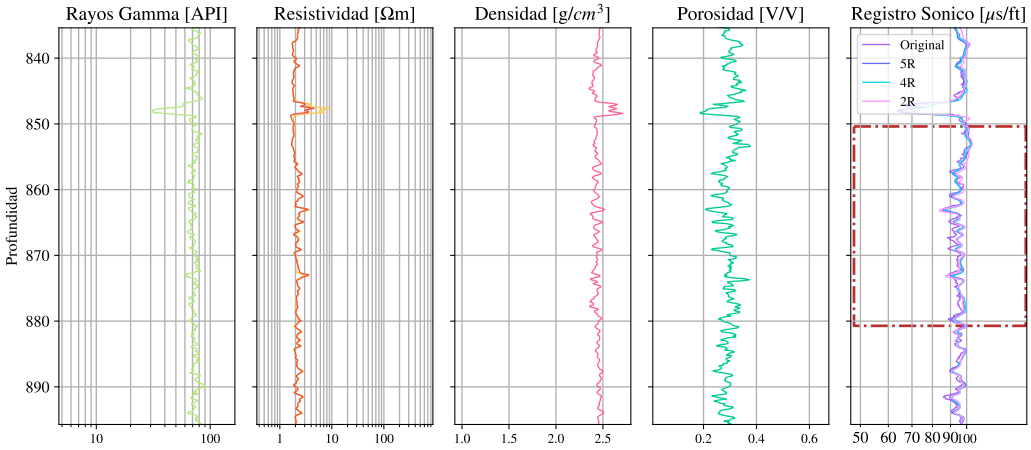
Figura 5.2: Predicciones del registro sónico en el Pozo A. La comparación del registro sónico original y los predichos se encuentra en el carril 5. Los rectángulos marrones corresponden a los intervalos de prueba (continuación).



(a) Intervalo 1

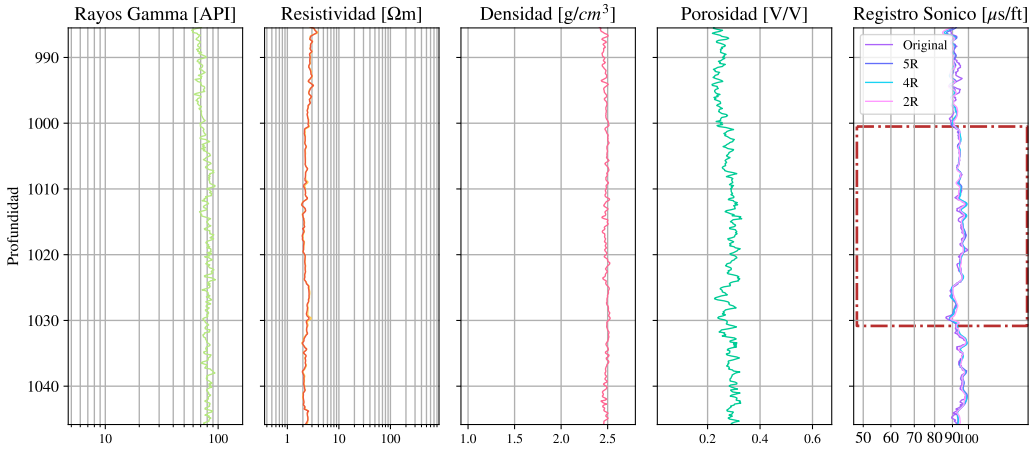


(b) Intervalo 2

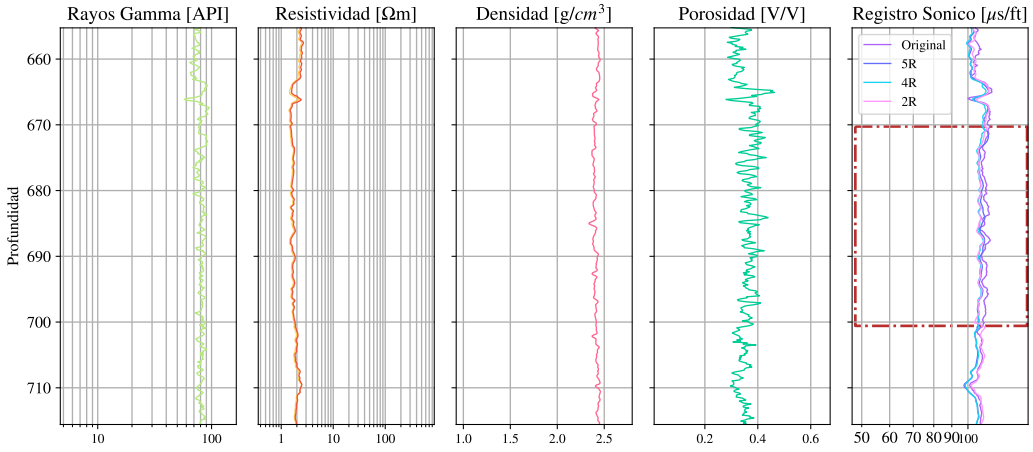


(c) Intervalo 3

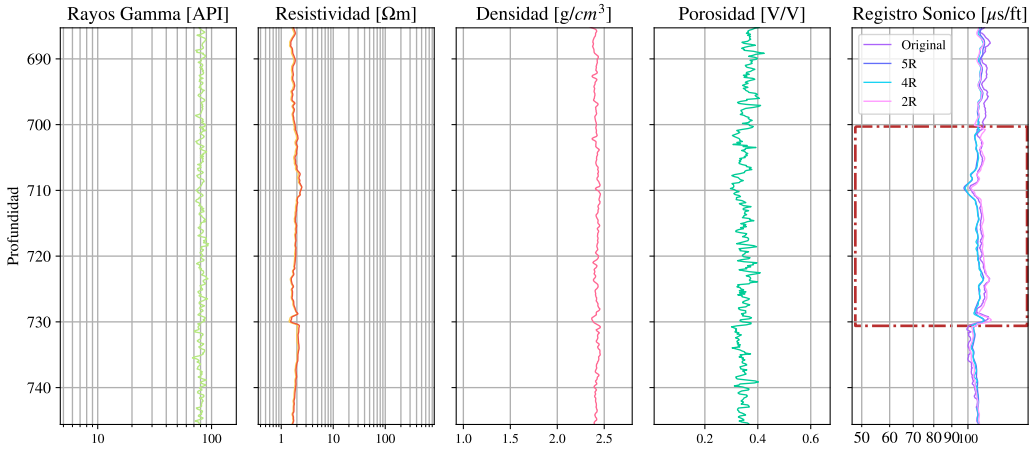
Figura 5.3: Predicciones del registro sónico en el Pozo B. La comparación del registro sónico original y los predichos se encuentra en el carril 5. Los rectángulos marrones corresponden a los intervalos de prueba.



(d) Intervalo 4

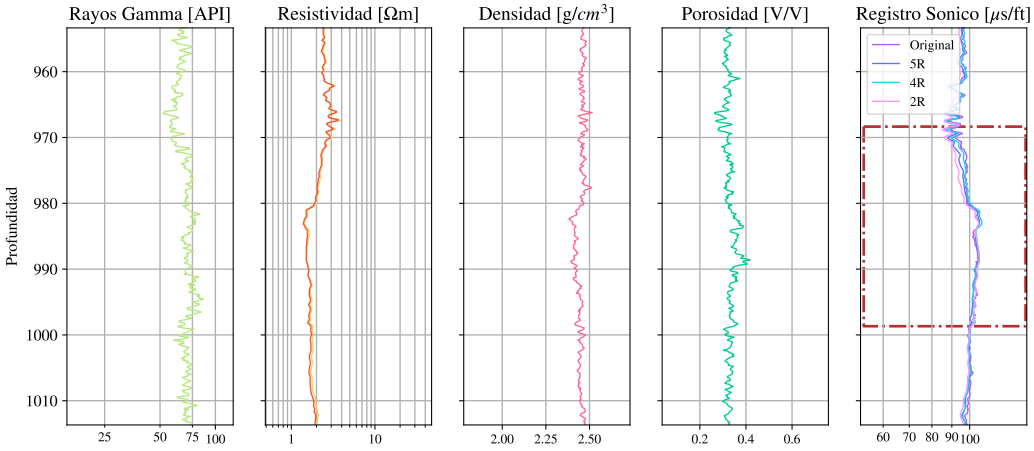


(e) Intervalo 5

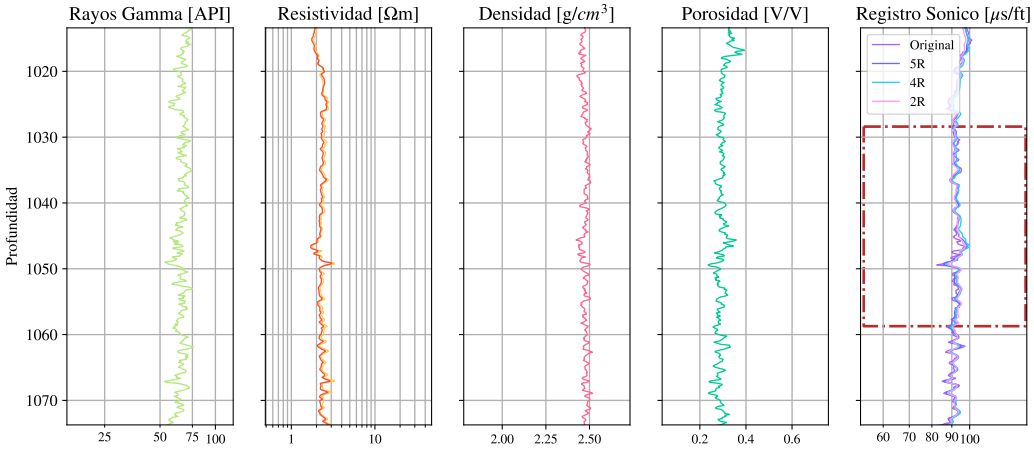


(f) Intervalo 6

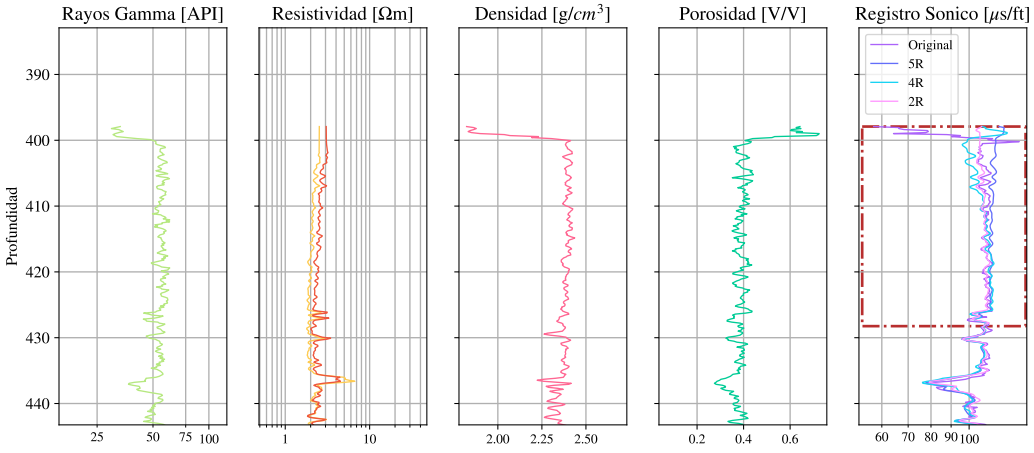
Figura 5.3: Predicciones del registro sónico en el Pozo B. La comparación del registro sónico original y los predichos se encuentra en el carril 5. Los rectángulos marrones corresponden a los intervalos de prueba (continuación).



(a) Intervalo 1

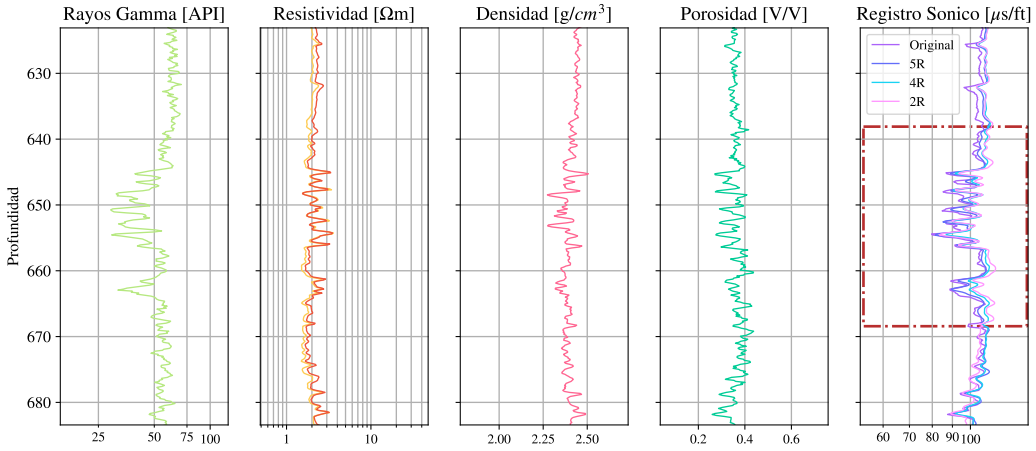


(b) Intervalo 2

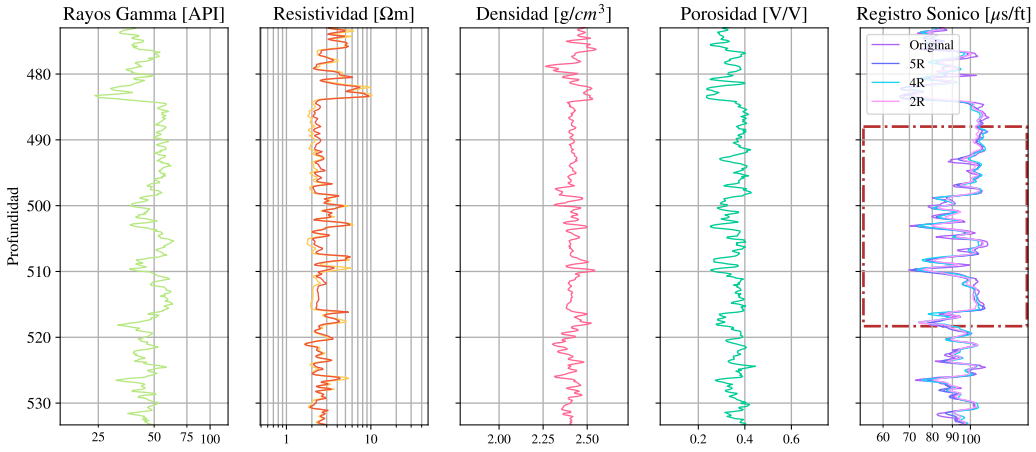


(c) Intervalo 3

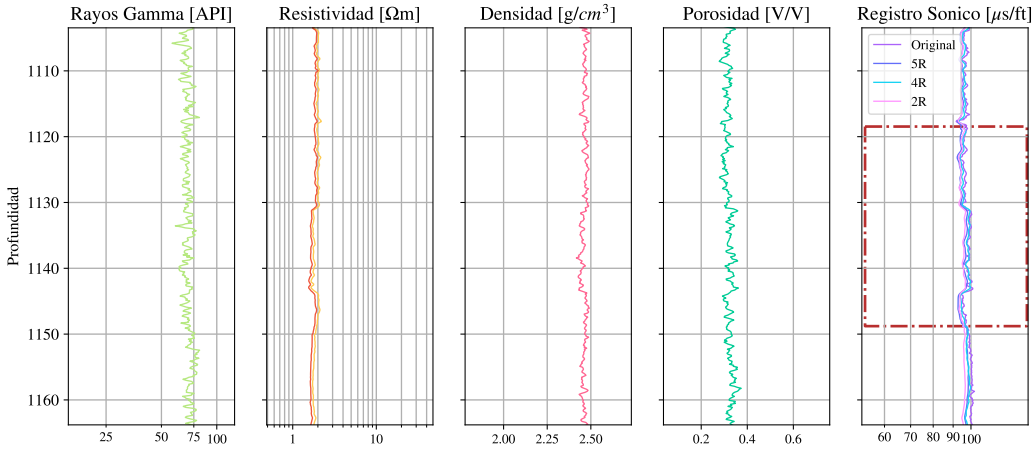
Figura 5.4: Predicciones del registro sónico en el Pozo C. La comparación del registro sónico original y los predichos se encuentra en el carril 5. Los rectángulos marrones corresponden a los intervalos de prueba.



(d) Intervalo 4

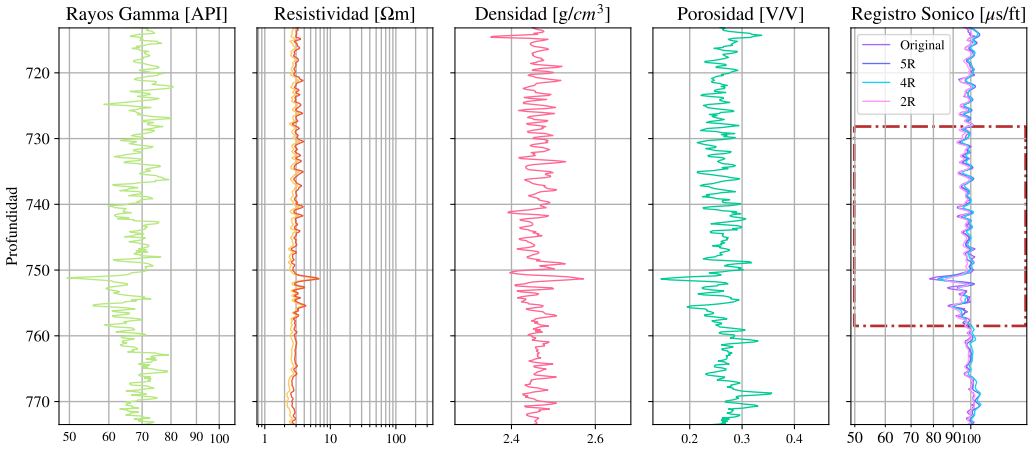


(e) Intervalo 5

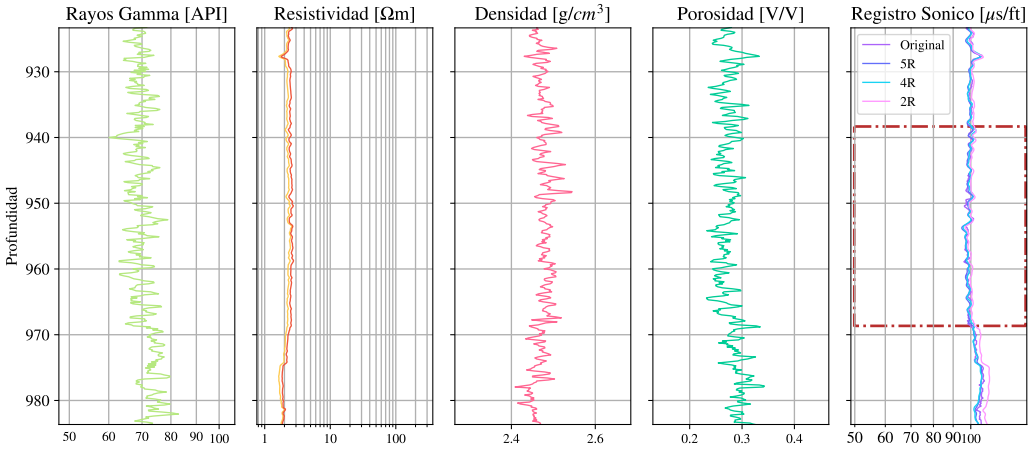


(f) Intervalo 6

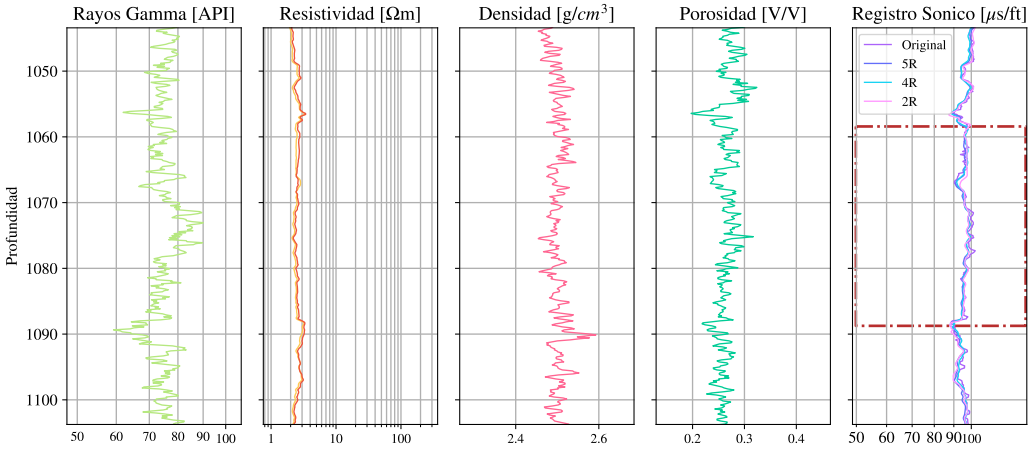
Figura 5.4: Predicciones del registro sónico en el Pozo C. La comparación del registro sónico original y los predichos se encuentra en el carril 5. Los rectángulos marrones corresponden a los intervalos de prueba (continuación).



(a) Intervalo 1

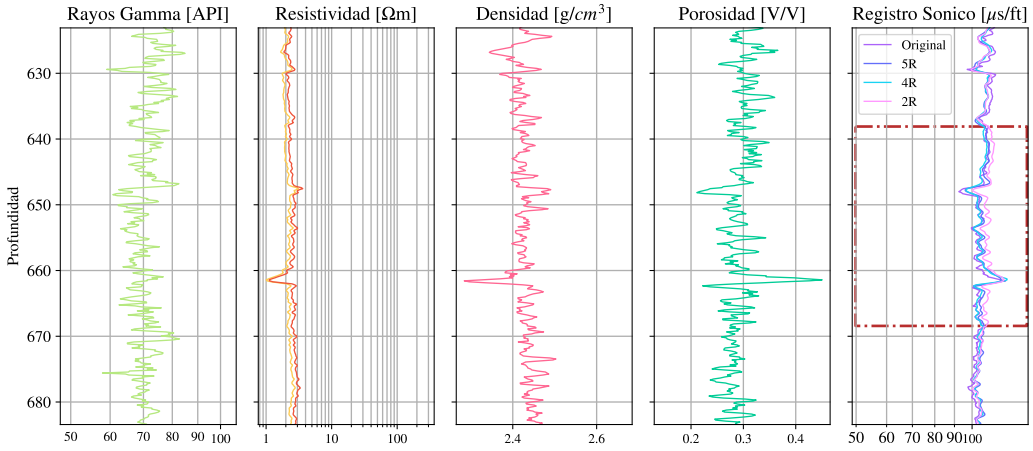


(b) Intervalo 2

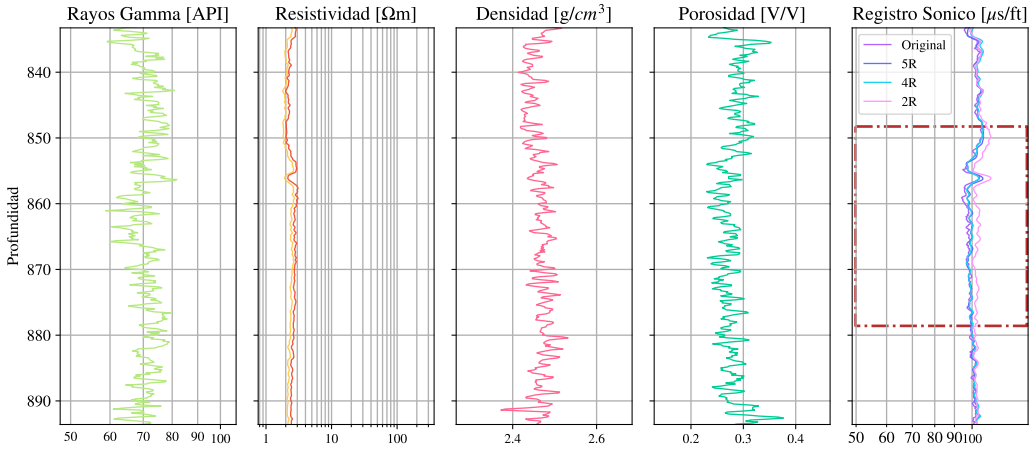


(c) Intervalo 3

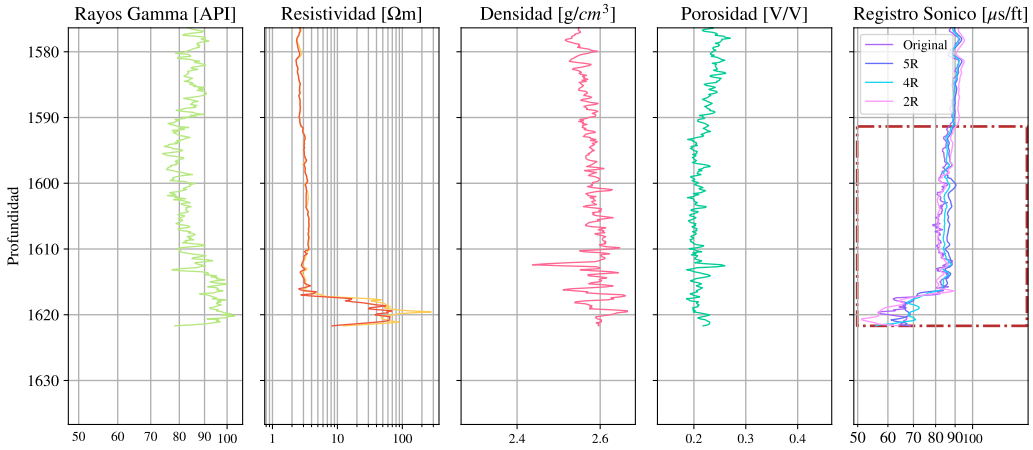
Figura 5.5: Predicciones del registro sónico en el Pozo D. La comparación del registro sónico original y los predichos se encuentra en el carril 5. Los rectángulos marrones corresponden a los intervalos de prueba.



(d) Intervalo 4



(e) Intervalo 5



(f) Intervalo 6.

Figura 5.5: Predicciones del registro sónico en el Pozo D. La comparación del registro sónico original y los predichos se encuentra en el carril 5. Los rectángulos marrones corresponden a los intervalos de prueba (continuación).

En el primer pozo, se emplearon los tres modelos debido a que se disponía de todos los registros utilizados durante el entrenamiento. En los dos pozos siguientes, solo se contaba con los registros de resistividad y rayos gamma, por lo que se utilizó el modelo 2R. Los registros sónicos generados para estos pozos se muestran en las Figuras 5.6 y 5.7, junto con los registros de entrada correspondientes.

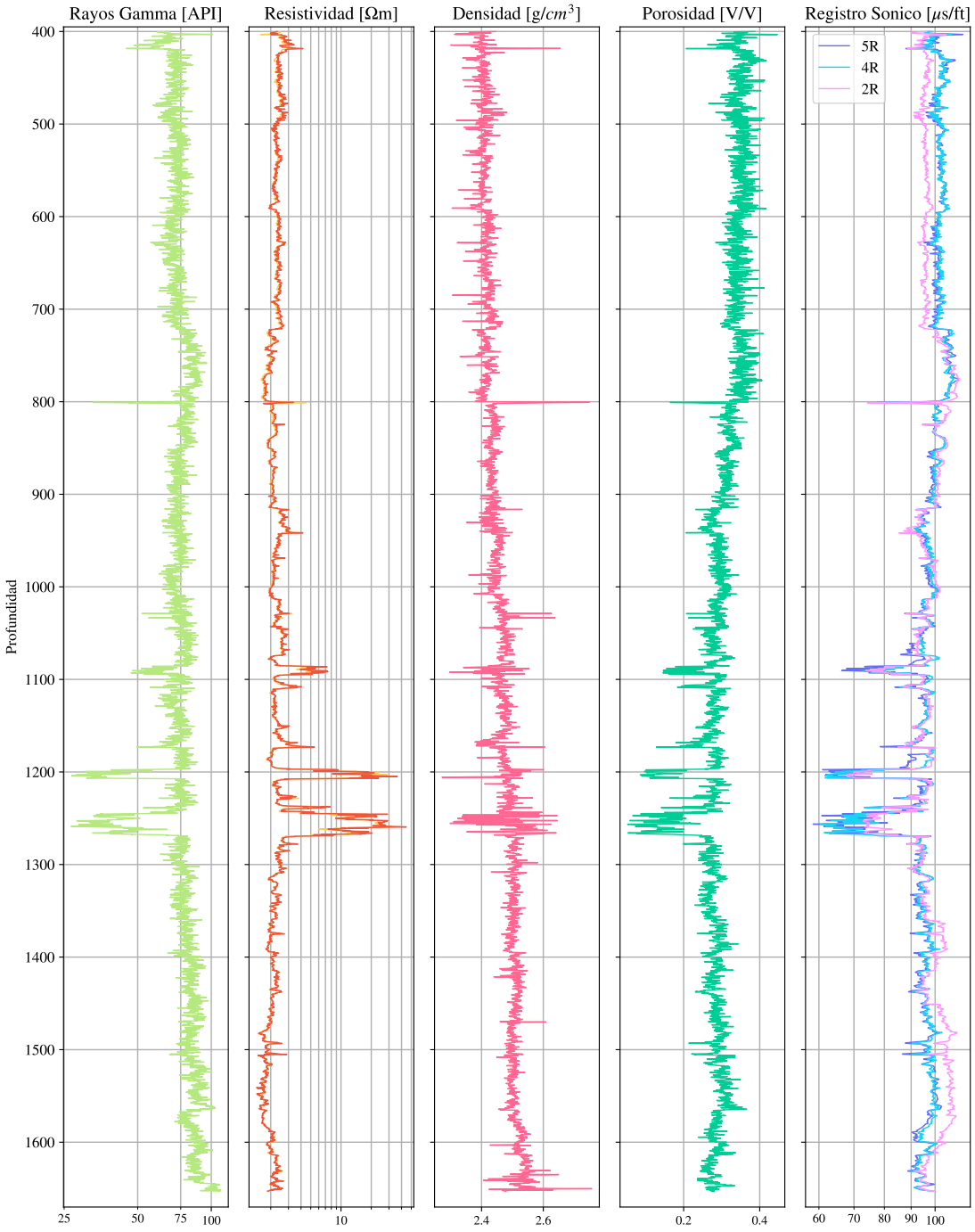
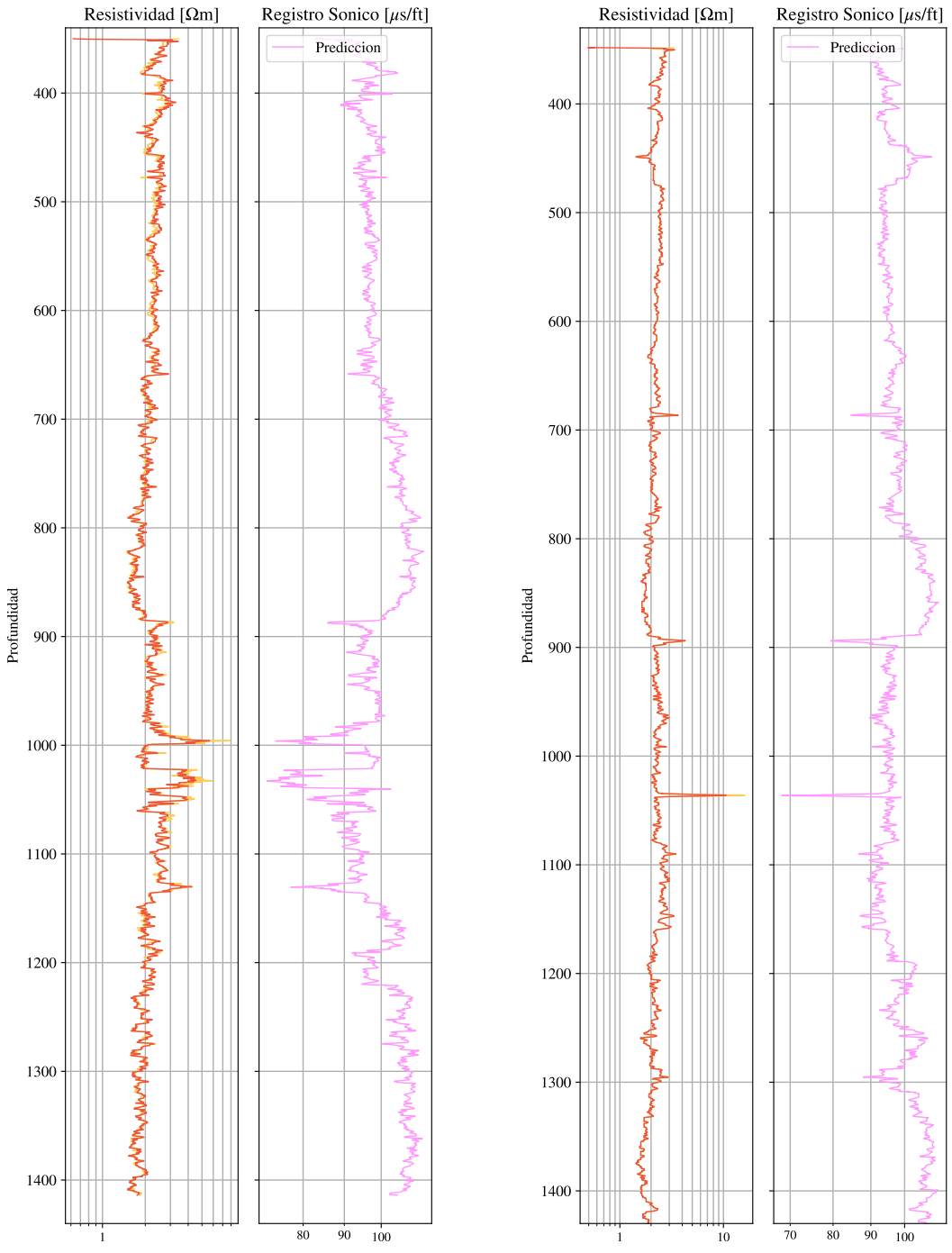


Figura 5.6: Predicción del registro sónico en el Pozo X con los modelos 5R, 4R y 2R.



(a) Pozo Y.

(b) Pozo Z.

Figura 5.7: Predicción del registro sónico en los Pozos Y y Z con el modelo 2R.

Capítulo 6

Discusión

En esta investigación, se empleó una red neuronal para predecir el registro sónico a partir de los registros de resistividad, rayos gamma, densidad y porosidad. El objetivo fue entrenar la red para aprender la relación entre el registro sónico y los otros registros en las secciones del pozo donde se contaba con datos completos, con el fin de completar el registro sónico en las áreas donde no se disponía de datos.

En este capítulo se reflexiona sobre los aspectos clave de la metodología y los resultados, incluyendo la selección y procesamiento de los datos, el entrenamiento de la red neuronal, los modelos obtenidos y su desempeño, así como una posible mejora para hacer la metodología más robusta.

6.1. Tratamiento de los datos

Antes de introducir los datos en la red neuronal, se aplicó una transformación logarítmica para manejar los valores atípicos considerados geológicamente relevantes. Esta transformación preservó las relaciones entre los datos y facilitó la estabilidad del entrenamiento. Los datos se dividieron aleatoriamente en conjuntos de entrenamiento, validación y prueba, con proporciones de 70 %, 15 % y 15 %, respectivamente. Esta división buscó equilibrar la cantidad de datos en cada conjunto, asegurando muestras representativas para cada etapa del proceso. Los histogramas de cada subconjunto muestran distribuciones similares, lo que indica que el muestreo aleatorio fue efectivo.

6.2. Convergencia

Las curvas de época contra costo (Figura 5.1) muestran el comportamiento esperado en el entrenamiento de redes neuronales. Inicialmente, el error para los datos de entrenamiento y validación es alto, pero disminuye con el aumento de épocas. El comportamiento errático de las curvas es característico del algoritmo de optimización Adam, por lo que se analizó conforme a su tendencia general.

En las primeras épocas, el costo de validación es menor que el de entrenamiento, indicando que la red necesita más entrenamiento. En épocas avanzadas, el costo de validación se estabiliza mientras que el de entrenamiento continúa disminuyendo, sugiriendo sobreentrenamiento. Para evitar este problema, se seleccionaron los modelos óptimos monitoreando el costo de validación, eligiendo aquellos con el costo de validación mínimo y un costo de entrenamiento menor pero cercano.

6.3. Métricas del Desempeño

La similitud entre el registro sónico real y la predicción se midió con el (Error Cuadrático Medio) y el MAPE (Error Absoluto Medio Porcentual) (Tablas 5.1 y 5.2). Estas métricas están normalizadas respecto al número de datos de cada registro, lo que permite una comparación objetiva a pesar de las diferencias en el número de muestras entre pozos. El RMSE tiene unidades de $[\mu s/ft]$ y proporciona una idea del tamaño del error en las unidades originales del registro. El MAPE, al ser una proporción, no está afectado por las unidades específicas y ofrece una visión sobre el error relativo.

6.4. Comparación entre modelos

El registro de rayos gamma mostró una dependencia menos clara con el registro sónico en comparación con otros registros (Figura 4.3). Esta falta de correlación se reflejó en las métricas de error, donde el modelo que incluía todos los registros (5R) y el modelo que excluía únicamente al registro de rayos gamma (4R) tuvieron resultados similares. Las métricas favorecieron al modelo 5R en los pozos B y C, mientras que el modelo 4R mostró mejor desempeño en el pozo D. En la predicción del Pozo X (Figura 5.6), ambos modelos produjeron resultados casi idénticos.

El modelo que utilizó solo el registro de resistividad (2R) mostró el peor desempeño, como era de esperar, debido a su naturaleza más austera. Sin embargo, su error en los intervalos de prueba se mantuvo dentro de un rango aceptable, y su predicción en el Pozo X mostró tendencias y rasgos generales similares a los de los otros modelos.

6.5. Propuesta de Mejora

Una posible mejora en la metodología sería incorporar el contexto geológico mediante la vectorización de la columna litológica. Esto permitiría usar información geológica numérica junto con los registros de pozo. La vectorización de la columna litológica podría asignar valores a las litologías basados en sus propiedades físicas, de manera que litologías similares se encuentren vectorialmente cercanas, análogo a cómo los modelos de inteligencia artificial para el lenguaje transforman palabras en vectores. Esto podría ayudar a la red neuronal a captar mejor las relaciones entre los datos geológicos y los registros de pozo.

Capítulo 7

Conclusión

El objetivo principal de esta tesis fue predecir el registro sónico en secciones de pozos donde no se contaba con datos de este a partir de su relación con otros registros de pozo, utilizando una red neuronal profunda.

Los resultados obtenidos, evaluados en intervalos de prueba y en secciones donde no se disponía del registro sónico original, demuestran la eficacia de este método para abordar la falta de datos en las adquisiciones. El desempeño de los tres modelos entrenados indica que el enfoque es robusto incluso cuando no se cuenta con todos los registros. Incluso el modelo que utiliza solo los registros de resistividad proporciona estimaciones útiles del registro sónico. Se pueden aplicar metodologías similares para predecir otros tipos de registros siempre que exista una dependencia entre ellos.

Dado que los registros predichos se asemejan bastante a los reales, estos modelos pueden ser útiles en diversas aplicaciones, ya sean en el campo de la exploración petrolera, en la geotécnica, geotermia, etc. De esta manera, estas industrias se benefician de métodos que permiten remediar errores en la adquisición de datos o reducir la necesidad de realizar mediciones en todos los pozos.

Para futuras investigaciones, se propone incorporar la columna litológica para mejorar la precisión de los registros sintéticos, especialmente en casos donde no se dispone de todos los registros. Además, esta incorporación podría permitir la aplicación de modelos entrenados en una región geográfica a otros pozos ubicados en regiones diferentes, ampliando así el alcance de los modelos desarrollados.

Bibliografía

[1] Amari, S. (1993). *Backpropagation and stochastic gradient descent method*, Neurocomputing, 5, 185-196. https://doi.org/10.1016/0925-2312(93)90006-O

[2] Asquith, G. & Krygowski, D. (2004). *Basic Well Log Analysis*. The American Association of Petroleum Geologists.

[3] Atat J. G., Uko, E. D., Tamunobereton-ari, I., & Eze, C. L. (2020). *The Constants of Density-Velocity Relation for Density Estimation in Tau Field, Niger Delta Basin*. IOSR Journal of Applied Physics, 12(1), 19-26. https://www.iosrjournals.org/iosr-jap/papers/Vol12-issue1/Series-1/D1201011926.pdf

[4] Bader, S., Wu, X., & Fomel, S. (2019). *Missing log data interpolation and semiautomatic seismic well ties using data matching techniques*. Interpretation, 7(2), T347-T361. https://doi.org/10.1190/INT-2018-0044.1

[5] Bader, S., Wu, X., & Fomel, S. (2018). *Missing well log estimation by multiple well-log correlation*. 80th EAGE Conference & Exhibition. https://doi.org/10.3997/2214-4609.201801234

[6] Bjørlykke, K. (2015). *Petroleum geoscience: From sedimentary environments to rock physics*. Springer. https://doi.org/10.1007/978-3-642-34132-8

[7] Brie, A., Endo, T., Hoyle, D., Codazzi, D., Esmersoy, C., & Hsu, K. (1998). *New Directions in Sonic Logging*. Oilfield Review, 10(1), 40-55.

[8] Chung, J., Gulchere, C., Cho, K., & Benigo, Y. (2014). *Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling*. arXiv preprint arXiv:1412.3555

[9] Ding, B., Qian, H., & Zhou, J. (2018). *Activation functions and their characteristics in deep neural networks*. En 2018 Chinese control and decision conference (CCDC) (pp. 1836-1841). IEEE. https://doi.org/10.1109/CCDC.2018.8407425

[10] Duchi, J., Hazan, E., & Singer, Y. (2011). *Adaptive subgradient methods for online learning and stochastic optimization*. Journal of machine learning research, 12(7). https://dl.acm.org/doi/10.5555/1953048.2021068

[11] Ellis, D. V. (2007). *Well logging for earth scientists*. Springer.

[12] Fu, J., Wang, B., Zhang, H., Zhang, Z., Chen, W., & Zheng, N. (2023). *When and why momentum accelerates sgd: An empirical study*. arXiv preprint arXiv:2306.09000. https://doi.org/10.48550/arXiv.2306.09000

[13] Glover, P. (2013). *The formation density log*. Petrophysics (pp. 13-1 a 13-20). University of Leeds. https://homepages.see.leeds.ac.uk/~earpwjg/PG_EN/CD%20Contents/GGL-66565%20Petrophysics%20English/Chapter%2013.PDF

[14] Glover, P. (2015). *The neutron log*. Petrophysics (pp. 15-1 a 15-20). University of Leeds. https://homepages.see.leeds.ac.uk/~earpwjg/PG_EN/CD%20Contents/GGL-66565%20Petrophysics%20English/Chapter%2015.PDF

[15] Glover, P. (2016). *Sonic log*. University of Leeds. https://homepages.see.leeds.ac.uk/~earpwjg/PG_EN/CD%20Contents/GGL-66565%20Petrophysics%20English/Chapter%2016.PDF

[16] Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press.

[17] Guntoro, T., Putri, I., & Bahri, A. S. (2013). *Petrophysical relationship to predict synthetic porosity log*. Search and Discovery Article, 41124. https://www.searchanddiscovery.com/documents/2013/41124guntoro/ndx_guntoro.pdf

[18] Hacikoylu, P., Dvorkin, J., & Mavko, G. (2006). *Resistivity-velocity transforms revisited*. The Leading Edge, 25(8), 1006-1009. https://doi.org/10.1190/1.2335159

[19] Halliburton. (s.f.). *Pseudo Sonic Log ProMax*. Halliburton. https://esd.halliburton.com/support/LSM/GGT/ProMAXSuite/ProMAX/5000/5000_8/Help/promax/pseudo.pdf

[20] Hinton, G., Srivastava, N., & Swersky, K. (2012). *Overview of mini-batch gradient descent*. In Neural Networks for Machine Learning (Lecture 6a). University of Toronto. http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf

[21] Kanal, L. N. (2003). *Perceptron*. Encyclopedia of Computer Science (pp. 1383-1385).

[22] Ketkar, N. (2017). *Deep learning with Python: A hands-on introduction*. Apress.

[23] Kingma, D. P., & Ba, J. L. (2015). *Adam: A method for stochastic optimization.* Proceedings of the 3rd International Conference for Learning Representations (ICLR). https://doi.org/10.48550/arXiv.1412.6980

[24] Kim, J., Lee, S., Park, H., & Choi, Y. (2022). *Synthetic shear sonic log generation utilizing hybrid machine learning techniques.* Journal of Geophysics and Engineering, 19(4), 123-134. https://doi.org/10.1016/j.aiig.2022.09.001

[25] Lai, J., Su, Y., Xiao, L., Zhao, F., Bai, T., Li, Y., Li, H., Huang, Y., Wang, G., & Qin, Z. (2024). *Application of geophysical well logs in solving geologic issues: Past, present and future prospect.* Geoscience Frontiers, 15(3), 101779. https://doi.org/10.1016/j.gsf.2024.101779

[26] Lau, M. M., & Lim, K. H. (2018). *Review of adaptive activation function in deep neural network.* En IEEE-EMBS Conference on Biomedical Engineering and Sciences (IECBES). (pp. 686-690) https://doi.org/10.1109/IECBES.2018.8626714

[27] LeCun, Y. (1988). *A theoretical framework for back-propagation.* En D. Touretzky, G. Hinton, & T. Sejnowski (Eds.) Proceedings of the 1988 connectionist models summer school (Vol. 1, pp. 21-28). Morgan Kaufmann. https://nyuscholars.nyu.edu/en/publications/a-theoretical-framework-for-back-propagation-2

[28] Li, Z., Xia, J., Liu, Z., Lei, G., Lee, K., & Ning, F. (2023). *Missing sonic logs generation for gas hydrate-bearing sediments via hybrid networks combining deep learning with rock physics modeling.* IEEE Transactions on Geoscience and Remote Sensing. https://doi.org/10.1109/TGRS.2023.3330869

[29] Lines, L. R., & Alam, M. (2012). *Synthetic seismograms, synthetic sonic logs, and synthetic core.* CREWES. https://www.crewes.org/Documents/ResearchReports/2012/CRR201259.pdf

[30] Liu, H. (2017). *Principles and applications of well logging.* Springer. https://doi.org/10.1007/978-3-662-54977-3

[31] Liu, Y., Gao, Y., & Yin, W. (2020). *An improved analysis of stochastic gradient descent with momentum.* In Proceedings of the 3rd International Conference for Learning Representations (ICLR). https://doi.org/10.48550/arXiv.2007.07989

[32] Luthi, S. M. (2001). *Geological well logs: Their use in reservoir modeling*. Springer. https://doi.org/10.1007/978-3-662-04627-2

[33] Maalouf, E., & Torres-Verdín, C. (2018). *Inversion-based method to mitigate noise in borehole sonic logs*. Geophysics, 83(2), D61-D71. https://doi.org/10.1190/geo2017-0334.1

[34] Makarian, E., Mirhashemi, M., Elyasi, A., Mansourian, D., Falahat, R., Radwan, A. E., El-Aal, A., Fan, C., & Li, H. (2023). *A novel directional-oriented method for predicting shear wave velocity through empirical rock physics relationship using geostatistics analysis*. Scientific Reports, 13, Article 47016. https://doi.org/10.1038/s41598-023-47016-9

[35] Mitchell, T. M. (1997). Machine learning. McGraw-Hill.

[36] Mirhashemi, M., Ranjineh Khojasteh, E., Shad Manaman, N., & Makarian, E. (2004). *Efficient sonic log estimations by geostatistics, empirical petrophysical relations, and their combination: Two case studies from Iranian hydrocarbon reservoirs*. Journal of Petroleum Science and Engineering, 45(3-4), 123-134. https://doi.org/10.1016/j.petrol.2022.110384

[37] Nie, F., Hu, Z., & Li, X. (2018). *An investigation for loss functions widely used in machine learning*. Communications in Information and Systems, 18(1), 37-52. https://doi.org/10.4310/CIS.2018.v18.n1.a2

[38] Ojala, I. (2009). *Using rock physics for constructing synthetic sonic logs*. Proceedings of the 3rd Canada-US Rock Mechanics Symposium. https://geogroup.utoronto.ca/wp-content/uploads/RockEng09/PDF/Session6/4016%20PAPER.pdf

[39] Pascanu, R., Gulcehre, C., Cho, K., & Bengio, Y. (2014). *How to construct deep recurrent neural networks*. En Proceedings of the 3rd International Conference for Learning Representations (ICLR). https://doi.org/10.48550/arXiv.1312.6026

[40] Pascanu, R. (2014). *On recurrent and deep neural networks*. Université de Montréal. https://www.iro.umontreal.ca/~lisa/pointeurs/RazvanPascanuDefenceSlide.pdf

[41] Pham, T., Tran, T., & Nguyen, H. (2020). *Missing well log prediction using convolutional long short-term memory network*. Geophysics, 85(4), WA159-WA170. https://doi.org/10.1190/geo2019-0044.1

[42] Potter, C. C., & Stewart, R. R. (1998). *Density predictions using Vp and Vs sonic logs*. CREWES Research Report, 10, 1-10. https://www.crewes.org/Documents/ResearchReports/1998/1998-10.pdf

[43] Quijada, M. F., & Stewart, R. R. (2007). *Density estimations using density-velocity relations and seismic inversion*. CREWES. https://www.crewes.org/Documents/ResearchReports/2007/2007-01.pdf

[44] Ran, Q., & Wang, Y. (2014). *Volcanic gas reservoir characterization*. Elsevier. https://doi.org/10.1016/C2013-0-19130-1

[45] Rosenblatt, F. (1958). *The perceptron: A probabilistic model for information storage and organization in the brain*. Psychological Review, 65(6), 386–408. https://doi.org/10.1037/h0042519

[46] Ruder, S. (2016). *An overview of gradient descent optimization algorithms*. arXiv. https://arxiv.org/abs/1609.04747

[47] Serra, O. (1984). *Fundamentals of well logging*. Elsevier.

[48] Sharma, S. (2020). *Activation functions in neural networks*. International Journal of Engineering and Applied Sciences and Technology. https://www.ijeast.com/papers/310-316,Tesma412,IJEAST.pdf

[49] Shen, G., Tan, Q., Zhang, H., Zeng, P., & Xu, J. (2018). *Deep learning with gated recurrent unit networks for financial sequence predictions*. Procedia Computer Science, 131, 895-903. https://doi.org/10.1016/j.procs.2018.04.204

[50] Tabian, I., Fu, H., & Sharif Khodaei, Z. (2019). *A convolutional neural network for impact detection and characterization of complex composite structures*. Sensors, 19(22), 4933. https://doi.org/10.3390/s19224933

[51] Tiab, D., & Donaldson, E. C. (2015). *Petrophysics: Theory and practice of measuring reservoir rock and fluid transport properties*. Elsevier.

[52] Tixier, M. P., Alger, R. P., & Doh, C. A. (1959). *Sonic logging*. Petroleum Transactions, AIME, 216(1), 106-114. https://doi.org/10.2118/1115-G

[53] Wang, J., Cao, J., Fu, J., & Xu, H. (2022). *Missing well logs prediction using deep learning integrated neural network with the self-attention mechanism*. Energy, 261. https://doi.org/10.1016/j.energy.2022.125270

[54] Wang, S., Yang, L., Chen, X., Chen, W., Saad, O. M., & Chen, Y. (2023). *Deep-learning missing well-log prediction via long short-term memory network with attention-period mechanism*. Geophysics, 88(1), D31–D48. https://doi.org/10.1190/geo2020-0749.1

[55] Wythoff, B. J. (1993). *Backpropagation neural networks: A tutorial*. Chemometrics and Intelligent Laboratory Systems, 18(1), 115-155. https://doi.org/10.1016/0169-7439(93)80012-7

[56] Zhang, A., Lipton, Z. C., Li, M., & Smola, A. J. (2023). *Dive into deep learning*. Cambridge University Press. https://d2l.ai/

[57] Zhang, D., Chen, Y., & Meng, J. (2018). *Synthetic well logs generation via Recurrent Neural Networks*. Petroleum Exploration and Development, 45(4), 629-639. https://doi.org/10.1016/j.petrol.2018.04.001