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

DETECCIÓN Y LOCALIZACIÓN DE FUGAS EN REDES DE DISTRIBUCIÓN DE AGUA POTABLE EN UNA GRAN CIUDAD DE CHILE MEDIANTE UN ALGORITMO DE CLASIFICACIÓN DE REDES NEURONALES

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL

JAVIER IGNACIO LOBOS CONSTANZO

PROFESOR GUÍA:
YARKO NIÑO CAMPOS

MIEMBROS DE LA COMISIÓN:
YOLANDA ALBERTO HERNÁNDEZ
NITZA MIRANDA VALLEJOS

Powered@NLHPC: Esta investigación/tesis fue parcialmente apoyada
por la infraestructura de supercómputo del NLHPC (ECM-02)

SANTIAGO DE CHILE
2022

RESUMEN DE LA MEMORIA PARA OPTAR
AL TÍTULO DE INGENIERO CIVIL
POR: **JAVIER IGNACIO LOBOS CONSTANZO**
FECHA: 2022
PROF. GUÍA: YARKO NIÑO CAMPOS

DETECCIÓN Y LOCALIZACIÓN DE FUGAS EN REDES DE DISTRIBUCIÓN DE AGUA POTABLE EN UNA GRAN CIUDAD DE CHILE MEDIANTE UN ALGORITMO DE CLASIFICACIÓN DE REDES NEURONALES

Con el fin de enfrentar la crisis hídrica presente en Chile, surge la necesidad de optimizar el uso del recurso, minimizando las pérdidas en sus distintos usos y aplicaciones. Entre ellos, las fugas en las redes de distribución de agua potable pueden significar pérdidas de hasta un 50 % del agua para abastecimiento, en función de la antigüedad de la red. Identificar las fugas presentes en la red tiene un alto costo debido a que la mayoría no se manifiestan a simple vista, requiriendo el uso de equipo especializado para su detección. Este trabajo de título tiene como principal objetivo la aplicación de un algoritmo de aprendizaje automático de redes neuronales, que permita la detección y localización de fugas en la red de distribución de agua potable en una comuna de una gran ciudad de Chile. El trabajo implica desarrollar un modelo hidráulico de la red a estudiar en el software EPANET, con lo cual se generarán conjuntos de datos de entrenamiento y validación, correspondientes a las presiones de la red para distintos estados de operación y ubicación de fugas, lo que permitirá el ajuste del clasificador para la detección de fugas en la red. Los resultados del proceso de entrenamiento y validación muestran que las redes neuronales tienen un buen comportamiento para la detección de fugas en la red de estudio, donde se observa un *F1-Score* de 99 %, 96 % y 97 % para los 3 casos de estudio propuestos que complejizan el problema progresivamente. También, mediante el análisis de las curvas de entrenamiento de los distintos modelos, se descarta la existencia de un sobreajuste a los datos de entrenamiento y se determina la arquitectura óptima de redes neuronales para cada caso de estudio. Finalmente se valida el algoritmo de redes neuronales para la detección de fugas en una red de estudio y se determina que supone una mejor opción frente a la maquina de vector de apoyo propuesta por Garate(2020), ya que presenta mayores rendimientos y menores tiempos de cómputo para el entrenamiento de los modelos determinados.

"A mi madre."

Agradecimientos

En primer lugar y con mucho orgullo agradezco a mi madre, aquella persona que siempre me ha brindado su apoyo y cariño incondicionalmente. Que cuyos incontables esfuerzos, muchas veces no gratificados como deberían, me han hecho llegar hasta este punto. Estos 2 últimos años nos afectaron de manera terrible y nunca terminaré de agradecer todos los sacrificios que has hecho por el porvenir mio y de mis hermanas. Espero poder regalarte el viaje a La Toscana que siempre has querido.

También quiero agradecer a mi padre, que pese a nuestras diferencias y quizás de manera mas silenciosa siempre ha estado ahí para mi. A mi hermana pequeña, que pese a su corta edad, me ha enseñado a valorar aquellas cosas que antes creía que no tenían importancia.

Agradezco también a mi grupo de amigos del liceo, por todo el tiempo que compartimos a diario y que nos ayuda a todos a vivir una vida mas alegre. A los amigos que hice en la facultad y en especial a esas 3 personas que me han acompañando desde el primer año en plan común hasta ahora en la mención de hidráulica, con quienes sufrimos, nos ayudamos y disfrutamos durante este proceso, se que no siempre fui el mejor amigo, sin embargo son de las personas que mas valoro en la vida, siempre han sido los mejores y siempre les voy a desear lo mejor.

Finalmente agradezco a mi profesor guía Yarko, por su buena voluntad y ayuda durante este trabajo. A los miembros del grupo CAPTA por usar sus conocimientos para ayudar a mejorar la sociedad en la que vivimos y por la oportunidad de realizar este trabajo.

Tabla de Contenido

Índice de Tablas

Índice de Ilustraciones

Capítulo 1

Introducción

1.1. Motivación

La escasez hídrica es una problemática que adquiere cada vez mayor importancia a nivel global. En Chile, actualmente, el agua es un recurso escaso fundamental para el desarrollo de la sociedad del país (Costa Cordella, 2016). Ante tal escenario, la necesidad de generar soluciones sustentables y optimizar al máximo el uso del recurso son vitales para asegurar el consumo de agua de la sociedad en la actualidad y a futuro.

Las redes de distribución de agua potable abastecen a la población del vital recurso, por lo cual constituyen parte de la infraestructura crítica para el funcionamiento de ciudades y territorios urbanos. Rojek y Studzinski (2019) plantean que un manejo óptimo de las redes de agua potable debe cumplir tres objetivos: (i) asegurar la calidad del agua; (ii) asegurar presiones adecuadas en los puntos de abastecimiento, y (iii) eliminar las pérdidas de agua en el sistema. Este último es particularmente relevante ante el escenario de escasez hídrica y corresponde al enfoque de este trabajo de título.

En el año 2018, el porcentaje de aguas no facturadas a nivel nacional alcanzó el 33,4 % del agua producida, correspondiente a 1787 millones de metros cúbicos (SISS, 2020). Es decir, un importante volumen de agua se pierde en las redes. Con el fin de disminuir tales pérdidas, se busca implementar una metodología para la detección de fugas en las redes, las cuales se manifiestan y permanecen desapercibidas por extensos periodos de tiempo, lo que genera las principales pérdidas de la red. Además, las pérdidas pueden aumentar hasta un 50 % del total de agua producida, dependiendo de la antigüedad de la red (Kemba et al., 2017).

Una tecnología para detectar fugas corresponde a la propuesta por Chan et al. (2019), quienes definen metodologías con enfoque basado en datos (*data-driven aproach*), que consisten en recolectar y procesar datos del funcionamiento de la red estudiada para identificar patrones que permitan predecir o clasificar las fugas presentes en la red. Este trabajo de título se basa en los estudios de Rojek y Studzinski (2019), Hu et al. (2021) y Fan et al. (2021), los cuales aplican un enfoque basado en datos para identificar fugas en un modelo hidráulico de la correspondiente red estudiada, mediante el algoritmo de clasificación denominado redes neuronales. Los autores concluyen que la precisión del algoritmo permite detectar fugas en las redes, por lo tanto, se pretende estudiar si la metodología puede ser aplicada en una red

de una gran ciudad de Chile.

Gárate (2020) validó un algoritmo de clasificación supervisada conocido como máquina de Vector de Apoyo (SVM) para la detección de fugas en parte de una red de agua potable de la misma gran ciudad de Chile. En este contexto, el presente trabajo de título evalúa la aplicabilidad de un algoritmo diferente en la red estudiada por Gárate (2020) con el fin de definir el óptimo para la detección de fugas.

1.2. Objetivos

1.2.1. General

Detectar y localizar fugas en la red de distribución de agua potable de una gran ciudad de Chile mediante inteligencia artificial, aplicando un algoritmo de redes neuronales.

1.2.2. Específicos

- Generar y calibrar un modelo hidráulico en el software de código abierto EPANET de la red de distribución de agua potable a estudiar.
- Implementar un clasificador de redes neuronales para permitir la detección de fugas en la red modelada mediante aprendizaje automático.
- Evaluar el desempeño del clasificador propuesto y comparar su efectividad con metodologías similares.

1.3. Estructura del trabajo

En el segundo capítulo se presentan los antecedentes de la literatura relevantes para el estudio. El tercer capítulo se detalla la metodología llevada a cabo. En el cuarto capítulo presenta la red de agua potable a estudiar y sus características. El quinto capítulo muestran los criterios utilizados para la generación de los datos de entrada mediante simulaciones hidráulicas del sistema y los resultados de la aplicación del algoritmo. Finalmente, en el séptimo capítulo se presentan las conclusiones, analizando el desempeño del algoritmo y su aplicabilidad para la solución de problemas reales.

Capítulo 2

Estado del arte

2.1. Motivación

Las redes de distribución de agua potable son aquellas que permiten abastecer a la población del recurso desde la planta de tratamiento. Jowitt y Xu (1990) plantean que las fugas que se generan en estas redes añaden costos innecesarios producto de la necesidad de una mayor presión de impulsión y un aumento de la demanda de la red producto de las fugas.

Gárate (2020) aplicó un algoritmo de clasificación supervisada de máquina de vector de apoyo (SVM) para la detección de fugas en parte de la red de distribución de una comuna de Santiago de Chile. El autor validó la implementación de algoritmos de clasificación supervisada para tal objetivo obteniendo aproximadamente un 77 % de precisión en la detección. Se busca implementar un algoritmo de clasificación de aprendizaje profundo (*deep learning*) en la red estudiada por Gárate (2020) y verificar si el cambio de algoritmo permite obtener mayores precisiones en la detección de fugas en la red.

2.2. Pérdidas en redes de distribución de agua potable

Las aguas no facturadas (ANF) son aquellas que son producidas por el sistema de agua potable pero no se facturan (SISS, 2020). Es decir, es el volumen de agua que es producida en la planta de tratamiento, pero no llega a los usuarios. González (2017) definió el esquema de la tabla 2.1 para clasificar las ANF:

Tabla 2.1: Flujos en redes de distribución de agua potable (González, 2017).

Consumo autorizado	Consumo facturado	Consumo facturado medido	Agua facturada
		Consumo facturado no medido	
	Consumo no facturado	Consumo no facturado medido	Agua no facturada
		Consumo no facturado no medido	
Pérdidas de agua	Pérdidas aparentes	Consumo no autorizado	
		Inexactitud de medidores y manejo de datos	
	Pérdidas reales	Fugas en tuberías	
		Fugas y reboses en estanques de almacenamiento	
		Fugas en las conexiones de servicio	

Adedeji et al. (2018) concluyeron que la mayor amenaza para las redes de distribución de agua potable, respecto a las pérdidas de agua, son las fugas en las tuberías de la red, lo que según la Tabla 2.1 corresponde a las pérdidas reales de la red, donde diversos autores cuantifican estas pérdidas en un rango entre el 70 % (Baader et al., 2011) y 90 % de las pérdidas reales de la red (Adedeji et al., 2018).

Una fuga puede definirse como una descarga no contemplada de agua dentro de la red. Baader et al. (2011) plantean que dentro de las redes pueden producirse distintos tipos de fugas (orificios, grietas, falla en las uniones, entre otros), ante lo cual, los autores las clasifican de acuerdo con su ubicación en el sistema y tamaño.

2.2.1. Clasificación de fugas según ubicación

- **En tuberías de transporte y distribución:** pueden ocurrir en las tuberías, uniones y válvulas, generan flujos de mediana a alta tasa y se presentan en intervalos de tiempo de corto a mediana duración.
- **En sistemas de arranque:** los sistemas de arranque se definen como el tramo entre la matriz de distribución y el medidor de agua potable ubicado en la propiedad del usuario, son considerados por los autores como puntos débiles de la red producto de la alta tasa de falla que tienen sus uniones y accesorios. Presentan una alta dificultad de detección producto del bajo flujo que presentan estas tuberías, por lo cual las fugas permanecen en intervalos de tiempo prolongados.
- **En estanque de almacenamiento:** causadas por defectos en sensores de nivel y permeabilidad de paredes del estanque, son de fácil detección con un alto costo de reparación.

2.2.2. Clasificación de fugas según tamaño y permanencia

- **Fugas reportables o visibles:** se manifiestan en las principales tuberías de la red de distribución producto de fallas en las tuberías o conexiones. Dependiendo del tipo de suelo, tamaño de fuga y presión de la tubería, el agua perdida emerge a la superficie y permite identificar la fuga.
- **Fugas no reportadas o no visibles:** se definen como fugas cuyo caudal supera los 250 l/h a 50 m de presión, pero, debido a condiciones desfavorables, no se manifiestan en la superficie.
- **Fugas de fondo:** se definen como fugas con caudales menores a los 250 l/h a 50 m de presión, los autores plantean que estas pequeñas fugas no pueden ser detectadas con instrumentos de detección acústica de fugas y asumen que gran parte de estas permanecen sin reparación hasta que la pieza falle y sea eventualmente reemplazada, sin embargo, Xu et al. (2019) plantean que existen metodologías que permiten la detección de pequeñas fugas con sensores acústicos.

Finalmente, en la Figura 2.1 se presenta un esquema de la clasificación de fugas donde es posible identificar que las mayores pérdidas se generan por las fugas de menor caudal

que permanecen en el tiempo. Baader et al. (2011) concluyeron que las fugas de fondo son la componente principal de las pérdidas reales en la red, por lo cual, al considerar conjuntamente las conclusiones de los autores con Adedeji et al. (2018), es posible concluir que la principal causa de las pérdidas de agua en toda la red de distribución de agua potable se presenta debido a fugas de fondo.

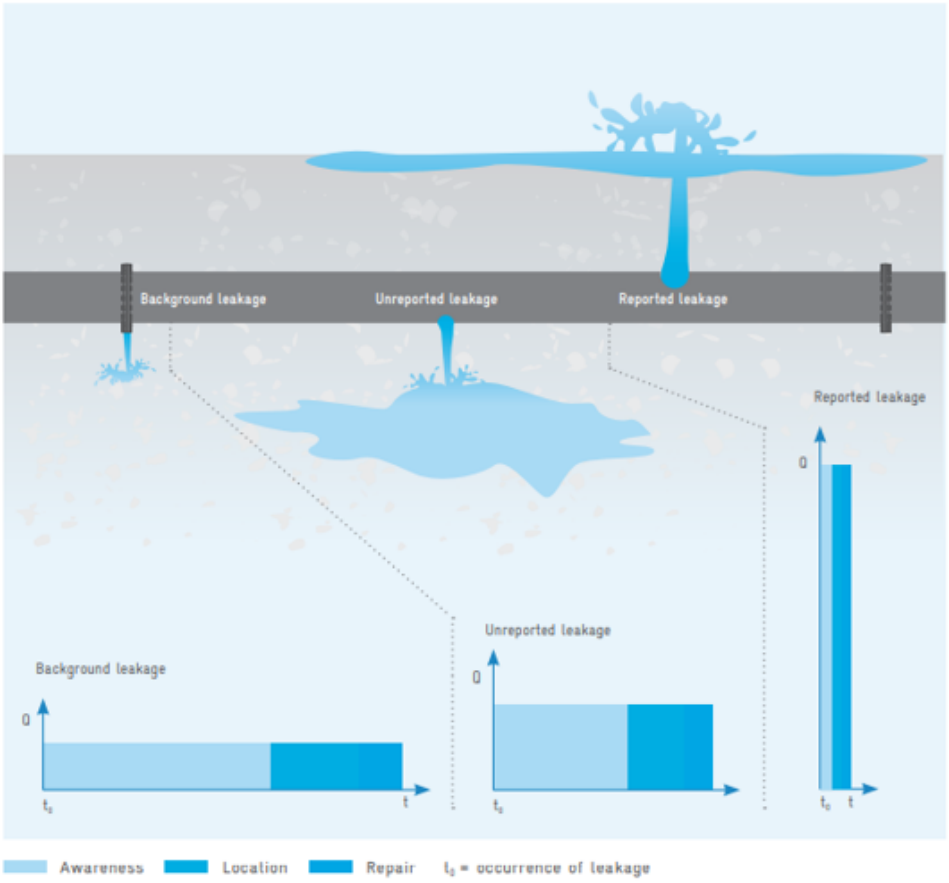

Figura 2.1: Esquema de fugas en red de agua potable (Klingel et al.,2018).

2.3. Hidráulica de fugas de agua potable

Las fugas en redes de agua potable pueden manifestarse en múltiples formas, entre las más comunes existen los orificios, roturas longitudinales y circunferenciales. Baader et al. (2011) modelan las fugas como un emisor, que se representa según la siguiente ecuación:

$$q = c \cdot h^{\alpha} \tag{2.1}$$

Donde:
q: caudal de fuga [gpm].
h: presión de la tubería [mca].
c: coeficiente de descarga [gpm/mca$^{0.5}$].
α: exponente de fuga [-].

La dependencia del caudal de fuga con la presión de la tubería se manifiesta mediante el exponente de fuga α el cual depende de una serie de factores:

- El tamaño y forma de la fuga.
- La capacidad de la fuga de cambiar de tamaño frente a cambios de presión, la cual depende de la forma de la fuga y la elasticidad de la tubería.
- El tipo de suelo que rodea la tubería.
- El régimen del flujo (laminar o turbulento).

Diversos estudios han demostrado que el valor de α típicamente se encuentra entre 0,5 y 2,79, siendo el valor promedio cercano a 1. Se recomienda el uso de un valor de α de 1,5 para tuberías flexibles (PVC, PE) y un valor de 0,5 para tuberías rígidas (AC, Hormigón, Hierro). No obstante, se recomienda utilizar un valor de 1,0 para redes de gran tamaño (Thornton et al., 2005).

Si bien el modelo de emisor parece simple para modelar un fenómeno que depende de una gran cantidad de factores, Walski et al. (2006) estudia experimentalmente fugas de distintos tamaños y formas para variados tipos de suelo y materialidad de tubería utilizando este modelo. Sus resultados indican que la ecuación representa de buena manera el fenómeno de fugas para la mayor parte de escenarios que se dan en la realidad de una red de agua potable.

2.4. Detección de fugas en redes de distribución de agua potable

De lo expuesto en las secciones anteriores, dado que la mayor parte de agua perdida en la red es producto de fugas, se desprende que la detección y localización de estas es fundamental para el manejo óptimo del recurso hídrico en las redes.

Chen et al. (2019) revisaron las distintas tecnologías disponibles para la detección y localización de fugas, diferenciando dos tipos de sistemas: activos y pasivos. Los sistemas activos son aquellos que infieren la ubicación de las fugas en la red con información existente que se mide a través de diferentes sensores existen dispuestos permanentemente en la red y registros históricos. Los sistemas pasivos son aquellos que utilizan diferentes tipos de sensores y metodologías para detectar las fugas en terreno.

Xu et al. (2019) afirmaron que existen metodologías para detectar fugas de fondo, las cuales representan las mayores pérdidas de la red, sin embargo, los autores plantearon que el alcance de aplicación de estas tecnologías es limitado (15 a 100 metros dependiendo de la materialidad de la tubería), lo que vuelve inviable evaluar el estado completo de una red que tiene cientos de kilómetros producto de la gran cantidad de mediciones a realizar.

Chen et al. (2019) definieron tres tipos de sistemas activos:

1. Basado en señales transitorias, el cual consiste en identificar las fugas en base a la respuesta que tiene la tubería ante ondas de presión que pueden ser generadas o modeladas.

2. Basado en modelos, el cual consiste en simular el comportamiento hidráulico de la red en base a funciones matemáticas. La ubicación de las fugas puede inferirse al simular el comportamiento de la red en condiciones de operación conocidas y comparar las presiones calculadas en la red con mediciones reales.

3. Basado en datos, consiste en la recopilación de datos para realizar análisis estadísticos que permitan generar modelos de predicción o clasificación de fugas. Una gran ventaja es que no se necesita un conocimiento avanzado respecto al sistema, ya que los patrones que permiten identificar las fugas se reconocen a través de los datos y mediciones. Por otro lado, su principal desventaja es el requerimiento de una gran cantidad de información para generar modelos consistentes, además que la información recopilada puede presentar anomalías en las mediciones que afectan el desempeño de la detección.

Gárate (2020) revisó las metodologías planteadas por Chen et al. (2019) con el fin de identificar aquellas que pueden ser aplicadas en la red a estudiar, y aplicó una metodología de sistema activo basado en datos: clasificación de fugas mediante una máquina de vector de apoyo, *support vector machine* (SVM), el cual es un algoritmo de aprendizaje de máquinas (*machine learning*). El autor basó su elección en el estudio de Kemba et al. (2017), quienes implementaron una SVM para la detección de fugas en una red de tuberías para una comunidad de 140 viviendas. Los autores argumentan que una SVM es un algoritmo efectivo para realizar tareas de clasificación binaria debido a que son poco sensibles a la falta de generalidad en los patrones. Adicionalmente, afirmaron que con una tasa de muestreo adecuada la SVM es más efectiva que los algoritmos de aprendizaje profundo (*deep learning*).

Rojek y Studzinski (2019) plantearon que la calibración del modelo hidráulico es fundamental para que los algoritmos de clasificación sean confiables y puedan ser utilizados para la detección de fugas en redes reales. Gárate (2020) indicó que pese a no obtener suficiente información, obtuvo una calibración promedio del 73 %, lo que según el autor es aceptable para la investigación. No obstante, en sus resultados presentó zonas en la red con calibraciones inferiores al 50 %, con un mínimo de 31,25 % para las zonas más críticas. Estos resultados permiten concluir que el promedio no es una medida adecuada para evaluar la calibración total del modelo, dado que pese a obtener una buena calibración promedio del modelo, la existencia de zonas de baja calibración puede generar un sesgo importante en los resultados de la modelación. Un indicador adecuado podría ser un promedio ponderado en base al área de la zona calibrada o el largo de tuberías en ésta.

Finalmente se concluye que para la aplicación de cualquier tipo de algoritmo en la red de estudio se deberá mejorar la calibración de la red respecto al estudio de Gárate (2020), con el fin de que el algoritmo permita detectar fugas para la realidad de la red.

2.4.1. *Machine learning y deep learning*

Chollet (2017) define los algoritmos de aprendizaje de máquinas (ML, por sus siglas en inglés) como un subcampo de la inteligencia artificial. El funcionamiento de los algoritmos se

basa en aplicar transformaciones a los datos de entrada para identificar los posibles patrones o características presentes. Algunas transformaciones utilizadas son cambios de coordenadas, proyecciones lineales, operadores no lineales, entre otros. Los algoritmos de *machine learning* son bastante utilizados para resolver problemas de predicción y clasificación de datos.

Las aplicaciones del ML pueden dividirse en 3 tipos de aprendizaje: supervisado, no supervisado y por refuerzo. El aprendizaje supervisado consiste en utilizar un conjunto de datos de entrada y de resultados esperados para encontrar relaciones funcionales que permitan predecir el resultado esperado con los datos de entrada. La detección de fugas en tuberías consiste en un problema de clasificación supervisada en el cual se tiene como entrada las presiones en los nodos de la red y como resultado esperado la ubicación de las fugas en la red.

Los algoritmos de *machine learning* tienen un aprendizaje netamente de los datos que se le entrega, sin la necesidad de otro tipo de conocimiento externo, no obstante tienen limitaciones. Entre las principales limitaciones de estos algoritmos están el requerimiento de grandes conjuntos de datos de entrenamientos, los cuales pueden implicar grandes costos (Royal Society of Great Britain, 2017), para el problema de detección de fugas el acceso a datos de entrenamiento es limitado, dado que depende del número de sensores ubicados en la redes de agua potable y la disponibilidad de información por parte de las empresas sanitarias.

El aprendizaje profundo (*deep learning*) es un subcampo del aprendizaje automático (*machine learning*) el cual basa su funcionamiento en generar representaciones sucesivas de los datos de entrada para generar el aprendizaje, las cuales se denominan capas. En general, la representación de las capas de aprendizaje se modela mediante redes neuronales, las cuales codifican la información de entrada a través de múltiples capas de neuronas que aplican una serie de transformaciones para obtener el aprendizaje obtenido.

En la Figura 2.2, se ilustra el funcionamiento de una red neuronal para la detección de dígitos en imágenes.

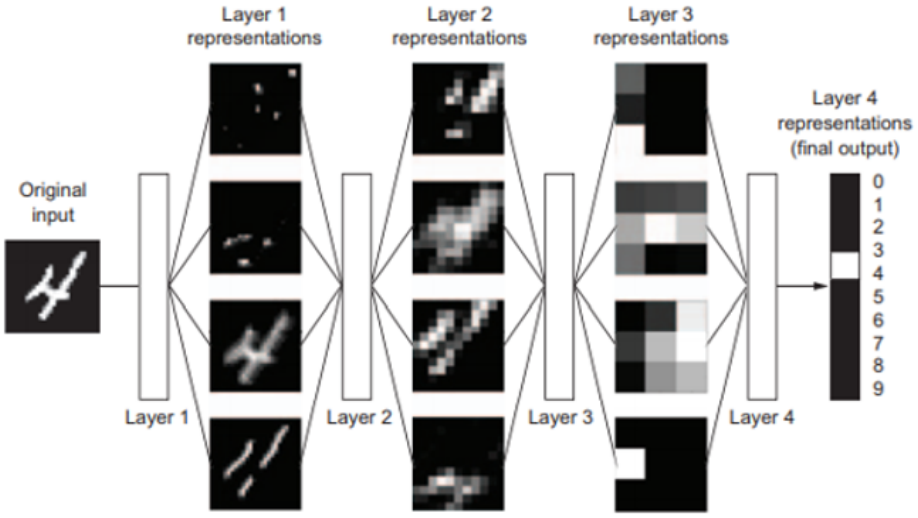

Figura 2.2: Red neuronal para la clasificación de dígitos (Chollet 2017).

Las redes neuronales artificiales (ANN, por sus siglas en inglés) son técnicas que basan su funcionamiento en la estructura del cerebro humano y sus neuronas, intentando reproducir su forma de aprendizaje. Es decir, se da en base a la información que recibe la red, la cual es capaz de generar una salida con esta información y reestructurarse en base al conocimiento adquirido en el proceso de aprendizaje. Las redes neuronales constituyen una herramienta muy efectiva para resolver problemas de clasificación (Jeatrakul y Wong, 2009).

Existen múltiples ventajas de usar ANN para problemas de clasificación:

- Puede adaptarse a los datos sin la necesidad de asumir previamente una función sobre estos.
- Las ANN son un aproximador universal de funciones, es decir, dada una función $f(x)$, existe una ANN que se aproxima a ésta con una precisión arbitraria (Gupta y Chandwani, 2012). Esta propiedad es de particular interés dado que el problema de clasificación consiste en encontrar la función $f(x)$ tal que $f(x) = y$, con x los datos de entrada al modelo e y la salida de la red.
- Las ANN son un modelo no lineal que puede adaptarse a cualquier tipo de problema.

Las neuronas de la red se organizan en tres tipos de capas: de entrada, las cuales reciben la información y contienen un número de neuronas igual al numero de entradas del sistema; las capas ocultas, que modifican la dimensionalidad del problema; y las capas de salida, donde se entrega la respuesta de la red (Figura 2.3).

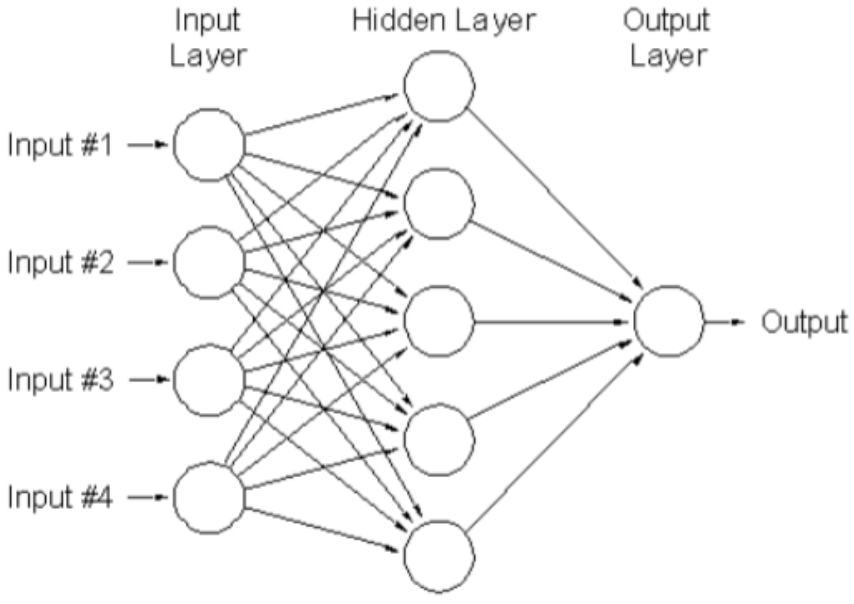

Figura 2.3: Estructura ANN (Jeatrakul y Wong, 2009).

En la Figura 2.4 se observa una neurona de la red, cada una está conectada con las neuronas de la capa siguiente y anterior. La conexión está dada mediante una ponderación lineal de la información de las neuronas de la capa anterior, las cuales tienen un peso asociado

W_{ij}. La respuesta esta dada por la aplicación de una función de activación $f(S_j)$ que varia dependiendo del algoritmo de entrenamiento y la arquitectura de la red utilizada (Dawson y Wilby, 2001).

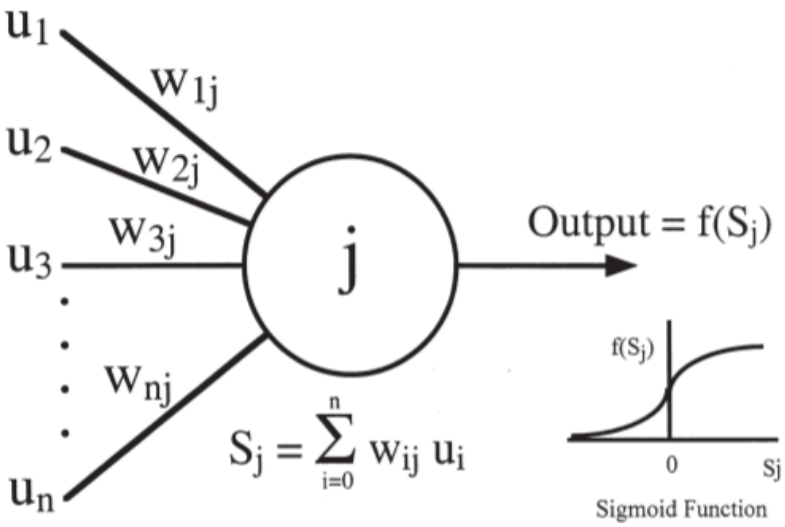

Figura 2.4: Neurona y aplicación de una función de activación f(S_j) (Dawson y Wilby, 2001).

Con respecto a las arquitectura de las redes, el perceptrón multicapa (MLP, por sus siglas en inglés) es una de las arquitecturas más utilizadas. Consiste en una organización 2D de las neuronas, como el que se observa en la Figura 2.2. El aprendizaje de estas redes consiste en reajustar las conexiones que tienen las neuronas mediante la variación de sus pesos (W_{ij} en la Figura 2.4). El algoritmo de entrenamiento más utilizado y efectivo corresponde al algoritmo de retropopagación (del inglés *back-propagation*) (Ramchoun et al., 2016). Esta arquitectura se ha utilizado para la detección de fugas por Rojek y Studzinski (2019).

Otra arquitectura ampliamente utilizada son las redes neuronales convolucionales (CNN, por sus siglas en inglés), las cuales son ampliamente utilizadas para detección de imágenes, de igual manera, se han aplicado para la detección de fugas en los estudios de Fan et al. (2021) y Zhou et al. (2019). Esta arquitectura contiene tres tipos de capa: capas de convolución, capas de agrupación y capas de conexión completa.

Como se observa en la Figura 2.5, las CNN, a diferencia del MLP, puede presentar arquitecturas en tres dimensiones, donde los datos de entrada corresponden a matrices. Las CNN presentan dos procesos claves, la convolución y el sub-muestreo. La convolución consiste en aplicar una transformación a un subconjunto de la matriz mediante un producto escalar con una matriz denominada filtro o *kernel*.

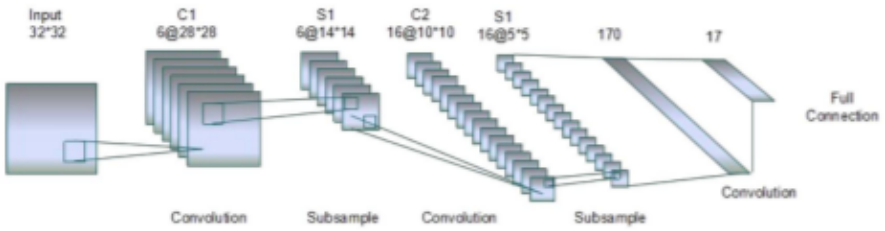

Figura 2.5: Arquitectura CNN (Agha et al., 2018).

La combinación de múltiples filtros generan un conjunto de nuevas matrices que se ubican en una capa de convolución observada en la Figura 2.6. La aplicación de este conjunto de filtros permite obtener múltiples características de los datos de entrada en las primeras capas de la red (Zhou et al., 2019).

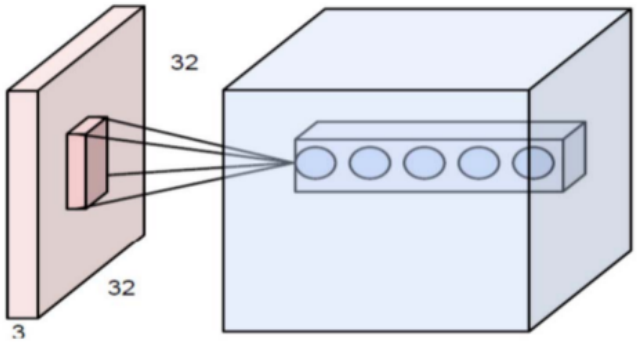

Figura 2.6: Convolución con múltiples filtros (Agha et al., 2018).

Otro proceso clave para el algoritmo de CNN es el sub-muestreo o *sub-sampling*, que consiste en disminuir en una medida aún mayor la cantidad de conexiones en la red, empleando un algoritmo de agrupación máxima o *max-pooling*. Éste consiste en incorporar las características más relevantes en las capas de convolución a una nueva capa, denominada de agrupación. La agrupación se realiza mediante un proceso similar a la convolución, donde se escoge una serie de subconjuntos de los datos para cada filtro y se obtiene el mayor valor, generando nuevas matrices con las características más importantes de cada filtro. Esta técnica se utiliza con el fin de reducir las conexiones existentes en la red neuronal, que para esta arquitectura son significativamente mayores a las ANN convencionales, permitiendo aumentar la eficiencia con la que aprende la red (Albawi et al., 2017).

Finalmente, al comparar la efectividad de las ANN con CNN se considera el estudio de Kareem et al., (2021). Los autores mostraron que, si bien la estructura CNN puede reconocer de forma más directa y eficaz las características de los datos, su aplicación para resolver problemas de clasificación de datos resulta excesiva dada la gran cantidad de información y capacidad de cómputo requerida en comparación a las ANN, que de igual forma logran aprender las características de los datos de forma indirecta. En esta misma línea, los auto-

res concluyeron que para este tipo de problemas las ANN tienen un mejor rendimiento y eficiencia, por lo cual se escoge la arquitectura de MLP como algoritmo de clasificación.

2.4.2. Problema de clasificación

El problema de clasificación en inteligencia artificial consiste en predecir a qué clase pertenece una entidad dado un conjunto de parámetros de entrada que posee. Dependiendo del comportamiento de las clases se consideran dos tipos de problemas de clasificación.

- **Clasificación multi-clase:** cada entidad se clasifica según una única clase del conjunto disponible.
- **Clasificación multi-etiqueta:** cada entidad puede clasificarse a una o más clases simultáneamente.

En el caso del problema de detección de fugas, la entidad a clasificar corresponde al estado de la red de agua potable, el cual se cuantifica mediante las presiones registradas por los sensores, los cuales son los parámetros de entrada para el modelo. Las clases están definidas por las tuberías de la red que presentan fugas.

El problema de clasificación multi-clase es el más estudiado en la literatura, donde se considera que en cada escenario sólo se manifiesta una fuga simultáneamente en el sistema. Sin embargo, producto del tamaño y antigüedad de las redes, este escenario es poco probable, dado que múltiples fugas pueden desarrollarse simultáneamente en la red, lo cual indica que el problema de clasificación multi-etiqueta es más adecuado para representar el estado de la red.

2.4.3. Funciones de activación

Cada neurona de la red tiene una función de activación que decide la cadencia con la cual se activan, además, otorga la propiedad de no linealidad a las ANN (Lau y Lim, 2019).

Actualmente la función de activación en las capas ocultas de las ANN por defecto es la función ReLU (unidad lineal rectificada), debido a su simplicidad y eficiencia en el entrenamiento de la red (Lau y Lim, 2019). La función ReLU tiene las siguientes ventajas respecto a otras funciones utilizadas (Albawi et al., 2017):

- Tiene definiciones simples tanto en su función como en su gradiente.

$$\mathrm{ReLU}(x) = \max(0, x) \tag{2.2}$$

$$\frac{d\mathrm{Relu}(x)}{dx} = \{1 \text{ si } x > 0;\ 0 \text{ en otro caso}\} \tag{2.3}$$

- Como se observa en la Figura 2.7, a diferencia de otras funciones de activación como, *sigmoid* o *tanh*, la función ReLU tiene un gradiente constante o nulo, lo que favorece el entrenamiento de los modelos al evitar una patología típica conocida como "Problema

de desvanecimiento de gradiente", en la cual producto de la regla de la cadena y de los bajos valores de las derivadas parciales de las funciones de activación dificulta el aprendizaje de la red, debido a que la actualización de los pesos de las conexiones neuronales dependen directamente de dichas derivadas, que al presentar un valor extremadamente pequeño mantienen constante las conexiones de la red en el entrenamiento y suprimen su aprendizaje.

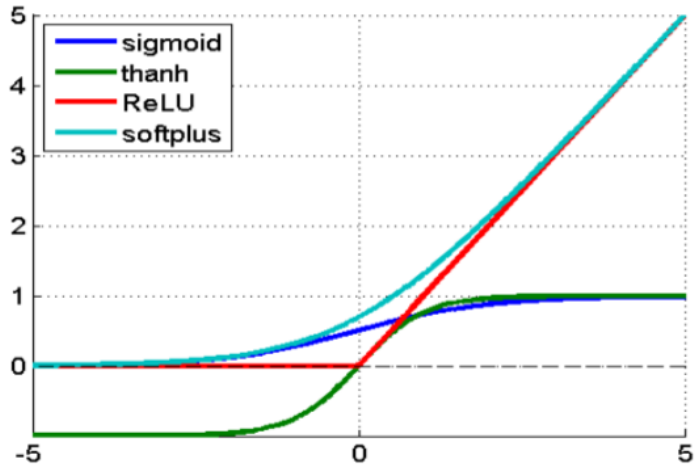

Figura 2.7: Funciones de activación típicas (Albawi et al., 2017).

2.5. Metodologías consideradas para la detección de fugas

En primer lugar, se considera el trabajo de Rojek y Studzinski (2019), quienes aplican un algoritmo basado en redes neuronales para la detección de fugas de fondo en diversas redes municipales de Polonia. La red estudiada por los autores contiene 514.7 km de tuberías, una red de mayor tamaño que la red estudiada por Gárate (2020) de 260 km, y presenta, según los autores, una pérdida del 30 %, muy similar a lo presentado por SISS (2020) para las redes en Chile.

Los autores generaron en base a la información disponible de las tuberías (longitud, diámetro, materialidad y ubicación) y de la operación de la red (régimen de caudales y presiones) un modelo hidráulico que es utilizado para la creación de los datos de entrada para el algoritmo de aprendizaje automático. Para calibrar el modelo los autores utilizan un algoritmo heurístico de optimización multi-criterio, el cual busca minimizar la diferencia de caudal y presión de los valores obtenidos en el modelo y los valores obtenidos por los sensores en la red. Para la generación de datos se realizaron 736 simulaciones de 37 fugas en nodos aleatorios de la red para caudales de 15 y 35 l/s los que se dividieron en una proporción de 75:15:15 en datos de entrenamiento, prueba y validación. Comparan la efectividad de dos tipos de redes neuronales, perceptrón multicapa y redes de Kohonen, las cuales combinan técnicas

tradicionales de redes neuronales con algoritmos de *clustering*. Los autores concluyen que para los problemas de detección de fugas la red perceptrón multicapa resulta ser el algoritmo más adecuado con una precisión del 86 %, la cual presenta mayor precisión que el algoritmo de SVM presentado por Gárate (2020).

Otro estudio considerado es el de Hu et al. (2021), en el cual se utiliza un algoritmo modificado de redes neuronales convolucionales para la detección de fugas en un modelo hidráulico, de una red de distribución ficticia. Los autores realizaron 632 simulaciones ubicando 4 fugas de 38 l/s aleatoriamente en los 49 nodos disponibles en la red. Los autores compararon su metodología propuesta con otros algoritmos de inteligencia artificial, en los que se incluye la SVM y concluyeron que su algoritmo presenta una mejora de un 78 % respecto a SVM.

Zhou et al. (2019) estudian un problema similar con un algoritmo de redes neuronales convolucionales para fugas de 4,6 l/s a 22,9 l/s y obtienen precisiones de aproximadamente 80 %. Un resultado importante de este estudio da a conocer que las fugas pequeñas tienen un bajo impacto en el estado de la red el cual tiende a ser confundido por variaciones en la demanda de la red, los autores plantean que la clave para mejorar la capacidad de detección de fugas pequeñas es la calibración del modelo hidráulico, lo que reafirma la necesidad de mejorar las calibraciones obtenidas por Gárate (2020) en su estudio.

Hu et al. (2021) utilizaron una arquitectura de ANN para la detección de fugas en un modelo hidráulico de 432 nodos donde se generan fugas de 0,05 m de diámetro. Entre las conclusiones más relevantes de este estudio se destaca la validación del algoritmo de redes neuronales para la detección de fugas en redes de mayor tamaño y que la detección en las áreas de monitoreo llegan a precesiones de hasta 100 %, la cual disminuye a medida que las tuberías son más lejanas a estas áreas, lo que revela una dependencia de la precisión del algoritmo con el nivel de sensorización del sistema.

Finalmente, para la calibración del modelo se consideran los estudios de Sanz y Pérez (2017) y Meier y Barkdoll (2000) que validan el uso de un algoritmo de calibración denominado algoritmo genético (GA), que basa su funcionamiento en la selección natural, trabajan con una población de individuos, los cuales representan una solución factible del problema. A cada individuo de la población se le asigna una puntuación relacionada a la adaptación de la solución, a medida que la puntuación de un individuo aumenta, la probabilidad que tiene de reproducirse con otro individuo aumenta. Al producirse el cruce, se genera un nuevo individuo con una combinación de la información genética (parámetros de la solución) de ambos padres, generando una nueva población. El algoritmo se repite iterativamente hasta que la población converge a una solución óptima del problema.

2.6. Conclusiones

De lo revisado en la literatura se concluye que, para obtener resultados consistentes y aplicables en la realidad, en primer lugar será necesario mejorar la calibración de la red estudiada respecto a lo planteado por Gárate (2020), para lo cual una metodología adecuada es el algoritmo genético validado por Sanz y Pérez (2017) para redes de gran tamaño.

Adicionalmente, se propone la aplicación de algoritmos de aprendizaje profundo como una alternativa al algoritmo de SVM, en específico se considera la arquitectura de perceptrón multicapa debido a que, según lo revisado en la literatura, esta arquitectura presenta mejor comportamiento en problemas de clasificación con datos acotados como la detección de fugas. También, se considera como una alternativa confiable debido a que sus propiedades le permiten adaptarse a cualquier tipo de problema, en especial en aquellos con marcada no linealidad.

Finalmente, de los estudios revisados, se concluye que el algoritmo logra precisiones mayores al 80 % para la detección de fugas en modelos hidráulicos de redes de menor tamaño y se ha validado como algoritmo efectivo para la resolución del problema estudiado.

Capítulo 3

Metodología

3.1. Formulación del modelo hidráulico

El modelo hidráulico a generar corresponde a la red estudiada por Gárate (2020) para la detección de fugas. En su estudio, el autor recompila la información de materialidad de tuberías, dimensiones, estimación de demandas del sector y la ubicación de los elementos de la red (válvulas, estanques y sus características) para generar un modelo hidráulico en el software *OpenFlows WaterGems*. En este estudio se generó un modelo hidráulico con los antecedentes recopilados por el autor, pero utilizando el softwate EPANET.

EPANET es un software de distribución libre ampliamente utilizado para la modelación hidráulica de redes de distribución de agua potable, desarrollada por la Agencia de Protección Ambiental (EPA, por sus siglas en inglés) de Estados Unidos. La elección de este software radica en su versatilidad, accesibilidad y a su compatibilidad para su uso directo en diferentes lenguajes de programación como *Python*, lo que permite generar los datos de entrada y desarrollar los algoritmos de detección en un mismo entorno de programación. Su elección también radica en que, dado el punto anterior, el software otorga la oportunidad de la utilización de la infraestructura del Laboratorio Nacional de Computación de Alto Rendimiento (NLHPC, por sus siglas en inglés) de la Universidad de Chile para el cálculo de las simulaciones realizadas.

La formulación del modelo planteado se realizó utilizando una herramienta presente en el software *OpenFlows WaterGems*, la cual permite exportar las características de la red a un archivo con extensión .INP, compatible con EPANET. Debido a que existen elementos como válvulas y bombas que funcionan de distintas maneras en ambos softwares, se realizó un proceso de validación del modelo reformulado, que consiste en ejecutar ambos modelos con iguales parámetros de entrada, verificando que las presiones y caudales calculados en las tuberías y nodos del sistema presenten el mismo valor.

El software EPANET fue empleado mediante el lenguaje de programación *Python*, junto con la herramienta *Water Network Tool for Resilience* (WNTR), un paquete de funciones diseñado para analizar la resiliencia de las redes de agua potable. El paquete esta basado en EPANET y permite construir y modificar redes en formato .INP y utilizar una serie de simuladores para calcular el estado de la red.

3.2. Generación de datos de entrada

En esta etapa se generaron los datos necesarios para el aprendizaje del clasificador, que consisten en los valores de las presiones detectadas en la entrada de las válvulas reductoras de presión (PRV) presentes en el sistema. Para asegurar una correcta clasificación se generaron escenarios con y sin fugas en el sistema. La generación de una fuga para una tubería de la red sigue el siguiente procedimiento:

1. Se escoge la tubería a fallar.
2. Se genera un nuevo nodo para el sistema ubicado en el punto medio de la tubería.
3. En el nuevo nodo generado se asigna una demanda correspondiente al caudal de la fuga, el cual puede ser constante o variable.

En el caso de fugas de caudal variable, se utilizó el modelo presentado en la ecuación 2.1, considerando un valor de α igual a 0,5 debido a que el 86 % de las tuberías de la red en análisis corresponden a materiales rígidos. El valor del coeficiente de descarga se calibró para alcanzar los valores de caudal deseado para la fuga.

Una vez definida las ubicaciones y caudales de las fugas impuestas en el sistema, entre cada simulación, se modificó la demanda base de cada nodo del sistema por un valor aleatorio que sigue una distribución normal $N(1; 0, 1)$. Es decir, se ponderó según un valor aleatorio en un rango del +/- 10 % su valor base, esto con el fin de otorgar variabilidad a los datos de entrada al modelo de clasificación. Adicionalmente, se evaluó el impacto que tiene el desempeño del clasificador al aumentar esta variación a una distribución normal $N(1; 0, 3)$.

Con la red ajustada para representar el estado deseado, se calculó el modelo con el paquete WNTR y el simulador EPANET. El modelo se calculó con un enfoque basado en demandas (DD), en el cual las presiones del sistema dependen de las demandas de los nodos, las cuales se suponen conocidas y satisfechas en su totalidad en las ecuaciones de balance de masa y pérdida de carga. El modelo se calculó en un periodo extendido de 24 horas.

Una vez calculado el modelo se recopilaron los datos necesarios para el entrenamiento de la red neuronal en 2 matrices. La matriz de la ecuación 3.1 corresponde a los datos de entrada del modelo, donde las columnas corresponden a cada uno de los 24 puntos de medición de presión en la red y las filas representan las mediciones de cada simulación. La matriz de la ecuación 3.2 corresponde a la matriz de salida y representa el estado de la red mediante un conjunto de vectores binarios donde cada elemento representa una tubería en el sistema que puede o no tener fuga.

$$X = \begin{pmatrix} PRV_1 & PRV_2 & \cdots & PRV_{24} \\ a & b & \cdots & c \\ \vdots & \vdots & \ddots & \vdots \\ d & e & \cdots & f \end{pmatrix} \tag{3.1}$$

$$Y = \begin{pmatrix} T_0 & T_2 & \cdots & T_n \\ 0 & 0 & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 0 & \cdots & 0 \end{pmatrix} \tag{3.2}$$

3.3. Desarrollo del modelo de clasificación

La implementación de las redes neuronales de clasificación se desarrolló mediante la librería *Tensorflow* en *Python*. *Tensorflow* es una biblioteca de aprendizaje automático desarrollado por *Google* con un amplio abanico de herramientas para implementar modelos de aprendizaje profundo.

Debido a que no existe una regla general para determinar el número apropiado capas y neuronas ocultas, la arquitectura de la red debe determinarse mediante un método de prueba y error hasta encontrar la red que permita representar cada problema. Con respecto a la capa de entrada, 24 neuronas son utilizadas para todos los problemas, donde, a cada sensor de presión en la red le corresponde una neurona diferente. En cuanto a las neuronas en la capa de salida, su cantidad corresponde al número de clases del problema, es decir, al número de tuberías que pueden presentar fugas, más una clase que corresponde al escenario sin fugas.

Acerca de las funciones de activación, para las neuronas de las capas de entrada y ocultas se utilizó la función ReLU por las ventajas que supone respecto otras funciones típicamente utilizadas. En cuanto a la capa de salida, la función de activación depende del tipo de problema a resolver:

- Para el problema de clasificación multiclase (una sola fuga se manifiesta en el sistema) se considera la función *Softmax*, como se observa en la Figura 3.1. Esta función permite asignarle una probabilidad de predicción normalizada a cada una de las clases de la red, las cuales, como regla, deben sumar 1, lo que permite que la salida de la red sea única, escogiendo aquella clase con mayor probabilidad (Nabiyev et al., 2021).
- Para el problema de clasificación multietiqueta (más de una fuga se manifiesta de manera simultánea) se considera la función de activación *Sigmoid* (Ecuación 3.3) para las capas de salida, que, a diferencia de la función *Softmax*, calcula la probabilidad de cada clase de manera independiente, lo que permite que existan clases clasificadas simultáneamente.

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{3.3}$$

Como se observa en la Figura 3.2, la función *Softmax* se adecua a problemas de clasificación multiclase y la función *Sigmoid* a problemas multietiqueta.

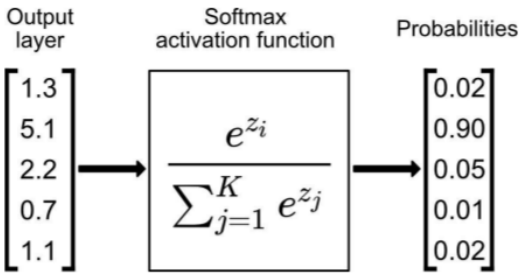

Figura 3.1: Comportamiento de la función *Softmax* en una capa de salida (Nabiyev y Malekzadeh, 2021).

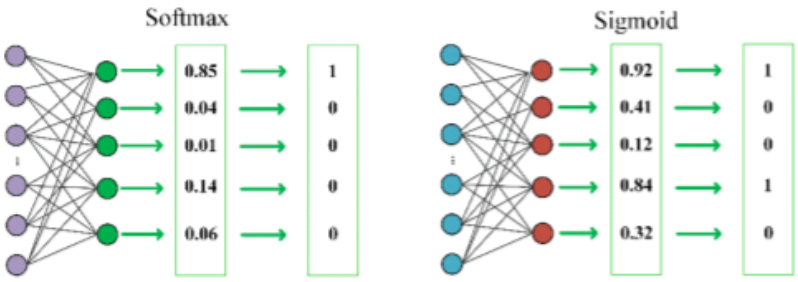

Figura 3.2: Diferencias entre *Softmax* y *Sigmoid* en la capa de salida de una ANN. Adaptado de (Chen et al., 2019).

3.4. Entrenamiento y validación

Para el entrenamiento de los modelos se consideró el algoritmo de *backpropagation*. Las fases del algoritmo son las siguientes:

- Se utiliza un subconjunto de los datos de entrenamiento como entrada para la red neuronal, lo que genera un conjunto de salidas.
- El conjunto de salidas es evaluado con las salidas esperadas mediante una función de pérdida, asignando un error, lo que es conocido como *feed-forwarding*.
- El error calculado es propagado desde las neuronas de salida. En la Figura 3.3 se presenta la retropropagación de una neurona a su capa predecesora, donde $h(x)$ corresponde a la salida de la red neuronal, que permite calcular $f(x)$, correspondiente a la función de pérdida y Delta al error correspondiente para cada neurona.
- Para la neurona de salida de la red (NS1), Delta_0 corresponde a $f(Z_0)$, es decir, la pérdida total de la red corresponde a las neuronas de salida.

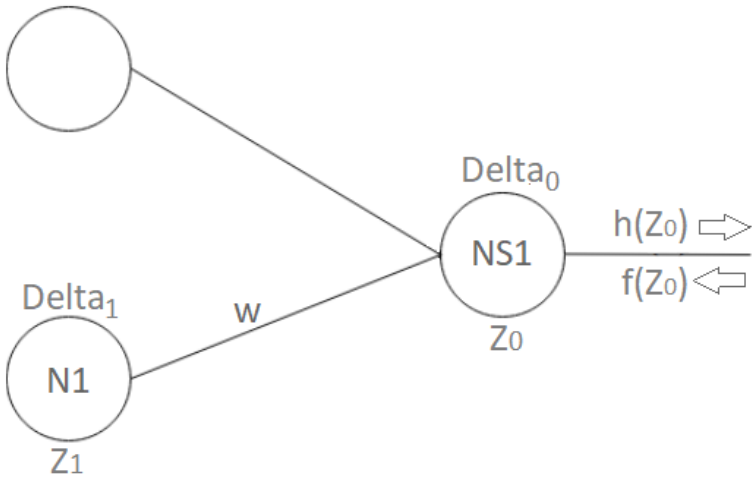

Figura 3.3: Retropropagación de error en la red neuronal.

- La propagación del error a las neuronas en las capas predecesoras se calcula en base al gradiente de la función *f(x)* según la regla de la cadena, con Delta_1:

$$\text{Delta}_1 = W \cdot f'(Z_1) \cdot \text{Delta}_0 \tag{3.4}$$

- Una vez calculado el error correspondiente a cada neurona se actualizan los pesos de las conexiones de la red mediante la técnica del descenso de gradiente, con lo que los pesos actualizados de la red se calculan como:

$$W_{\text{actualizado}} = W - \alpha \cdot Z_i \cdot \text{Delta}_i \tag{3.5}$$

  Con α correspondiente a la razón de aprendizaje del algoritmo, lo que puede interpretarse como la sensibilidad que tiene la red a cambiar su estructura.

- Finalmente, se actualizan los pesos de toda la red y se procesan nuevos subconjuntos de entrenamiento recursivamente, hasta llegar al nivel de aprendizaje deseado.

Como funciones de pérdida se considera la entropía cruzada binaria para el problema de clasificación multiclase y la función de entropía cruzada categórica para el problema de clasificación multietiqueta, debido a que son ampliamente utilizadas con las funciones de activación *Sigmoid* y *Softmax*, respectivamente (Nabiyev et al., 2021).

Para la validación de los modelos se consideraron las métricas indicadas en la tabla 3.1:

Tabla 3.1: Métricas consideradas para la validación de los modelos.

Métrica	Descripción	Expresión
Accuracy	Métrica que mide la cantidad de aciertos del modelo en base a la cantidad de verdaderos positivos (TP) y verdaderos negativos (TN) por sobre el total considerando los falsos positivos (FP) y falsos negativos (FN). Es una de las métricas más utilizadas para evaluar modelos de inteligencia artificial. Su valor se encuentra entre 0 y 1.	$\frac{TP+TN}{TN+TP+FP+FN}$
Recall	Métrica que mide la cantidad de elementos que el modelo es capaz de identificar. En el caso de la detección de fugas, la métrica indica cuantas de las fugas detecta el modelo. Su valor se encuentra entre 0 y 1.	$\frac{TP}{TP+FN}$
Precision	Métrica que mide el porcentaje de acierto del modelo. En el caso de la detección de fugas, la métrica indica cuantas de las fugas detectadas por el modelo son efectivamente fugas. Su valor se encuentra entre 0 y 1.	$\frac{TP+TN}{TN+TP+FP+FN}$
F1-Score	Corresponde al promedio armónico de *Precision* y *Recall*. Consiste en una métrica útil para evaluar de forma general el modelo. Su valor se encuentra entre 0 y 1.	$\frac{2 \cdot Precision \cdot Recall}{Precision+Recall}$

Capítulo 4

Modelo hidráulico red de estudio

4.1. Descripción de red de agua potable e información disponible

La Figura 4.1 presenta la red de distribución estudiada en el trabajo, que abarca a cuatro de las 32 comunas presentes en la gran ciudad. La red fue seleccionada producto de la amplia variedad de información disponible y con el fin de mantener la continuidad del estudio de Gárate (2020) y comparar las metodologías.

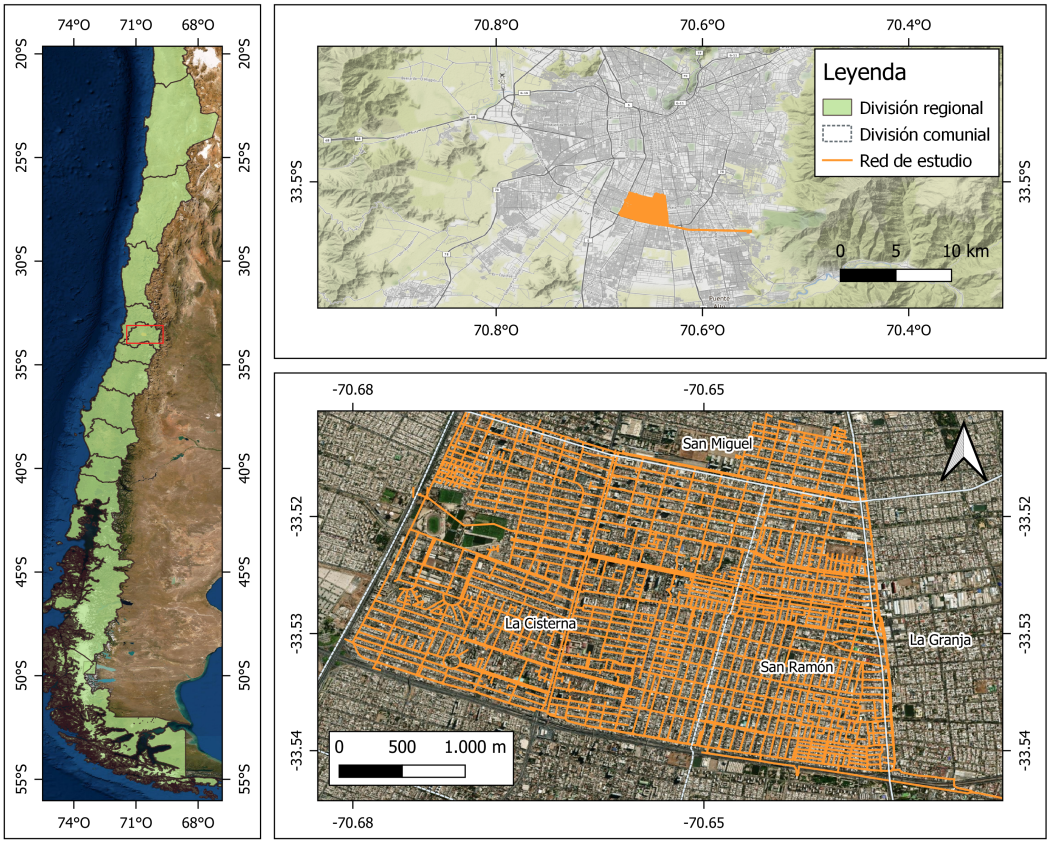

Figura 4.1: Ubicación de la red de distribución en estudio.

La información correspondiente a la disposición, materialidad y operación de la red fue recopilada e implementada en un modelo hidráulico por Gárate (2020) en el software *WaterGems*. El modelo contempla la representación de la red en un periodo extendido de 24 horas para el mes de agosto de 2020.

La Tabla 4.1 presenta la distribución de las materialidades para las tuberías de la red. Se observa una alta predominancia de materiales rígidos (a excepción del polietileno de alta densidad y policloruro de vinilo) que corresponde aproximadamente un 86 % de las tuberías de la red. De esta forma, para el modelo de fuga planteado en la Ecuación 2.1, se prefirió un valor de α igual a 0,5 correspondiente a fugas en materiales rígidos por sobre el valor de α igual a 1 para redes de gran tamaño, que supone una alta heterogeneidad en los materiales de las tuberías en la red.

Tabla 4.1: Materialidad tuberías de la red.

Materialidad	**Longitud [m]**
Asbesto Cemento	182.715,9
Acero	5.653,9
Concreto	1.093
Fierro fundido	4.507,1
Hormigón	7.588,4
Polietileno de alta densidad	28.266
Policloruro de vinilo	2.342,7
Total	232.167

Entre los elementos que condicionan la operación de la red, en la Tabla 4.2 se identificaron estanques de alimentación, válvulas reductoras de presión (PRV) y válvulas de control de flujo (FCV).

Tabla 4.2: Válvulas presentes en el sistema.

Elemento		
PRV	**n°**	24
FCV	**n°**	69

Adicionalmente, en la Figura 4.2 se dispone de mediciones telemétricas de la presión de entrada y salida para cada PRV del sistema, con un paso de tiempo de 15 minutos entre mediciones, las cuales permiten monitorear y cuantificar el comportamiento real de la red.

Finalmente, se considera la información respecto a patrones de consumo en nodos domiciliarios y con edificios de mayor tamaño estimada por Gárate (2020), calculadas en base al caudal de salida de los estanques de almacenamiento disponibles. Las estimaciones consideran ocho patrones de consumo que diferencian las cuatro semanas del mes de agosto separadas en patrones de semana (lunes-viernes) y fin de semana (sábado-domingo).

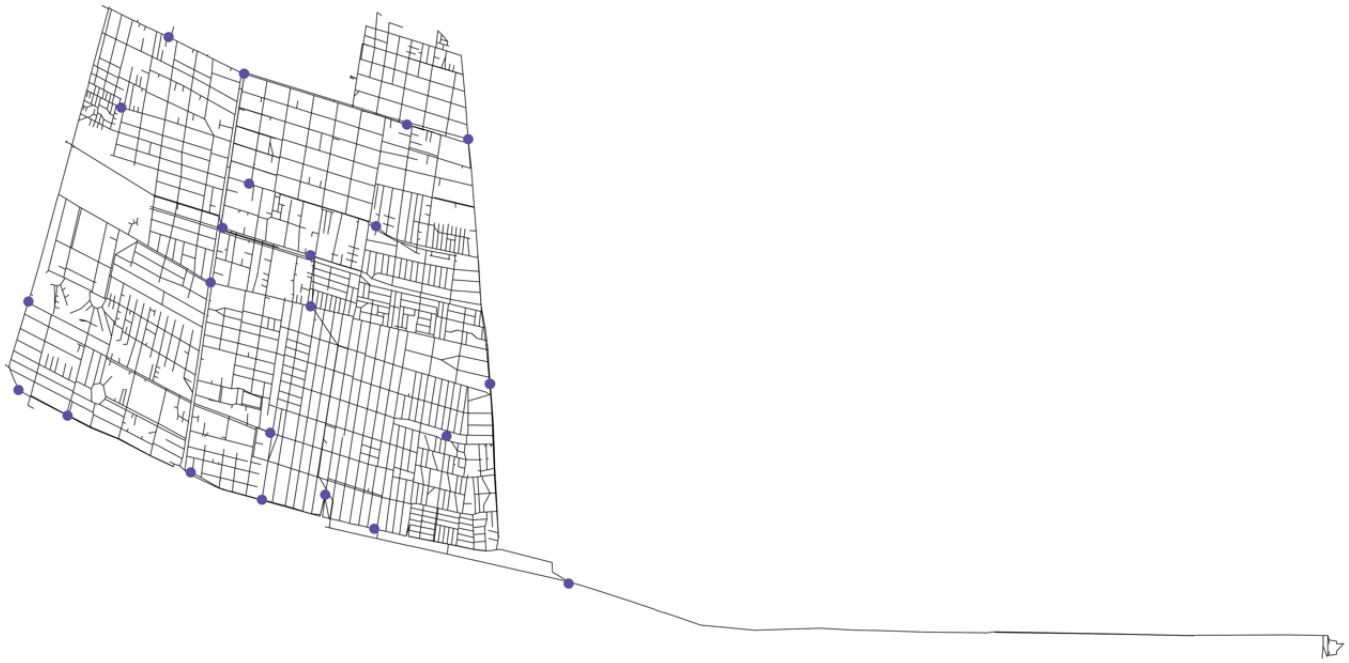

Figura 4.2: Puntos de medición telemétrica de presión (representados en azul), la presión es medida inmediatamente aguas arriba y aguas abajo de la válvula reductora de presión.

4.2. Planteamiento del modelo hidráulico en EPANET

Se utilizó la herramienta presente en el software *WaterGems* para exportar el modelo hidráulico a un formato compatible con EPANET. Para validar que el modelo es consistente al cambiar de formato, se realizó una simulación con los mismos parámetros de entrada para ambos modelos y se compararon las presiones en los nodos del sistema. En la Figura 4.3 se presentan las presiones calculadas en ambos modelos para el nodo 11.459.

Como se observa en la Figura 4.3, las presiones en el nodo presentado difieren significativamente, lo que implica que el modelo no se ha representado correctamente al exportar la información de *WaterGems*. La inconsistencia entre ambos modelos se produce porque existen funcionalidades de *WaterGems* que EPANET no posee de manera directa. En particular, se observa que EPANET no permite asignar patrones de funcionamiento a las válvulas reguladoras de presión en el sistema.

Además, en la Figura 4.4 se observa el patrón de funcionamiento de la PRV 33.036, que representa como varía la consigna de la válvula durante el día. La consigna se define como la presión de salida a la que esta configurada la válvula, mientras tanto el patrón esta definido según la razón entre la consigna que presenta la válvula en un tiempo dado y una consigna base definida.

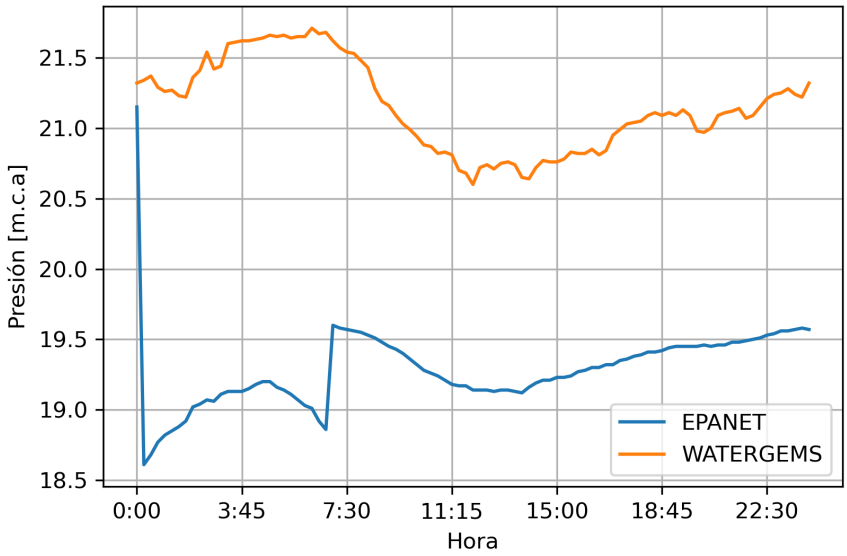

Figura 4.3: Presiones calculadas en el nodo 11.459.

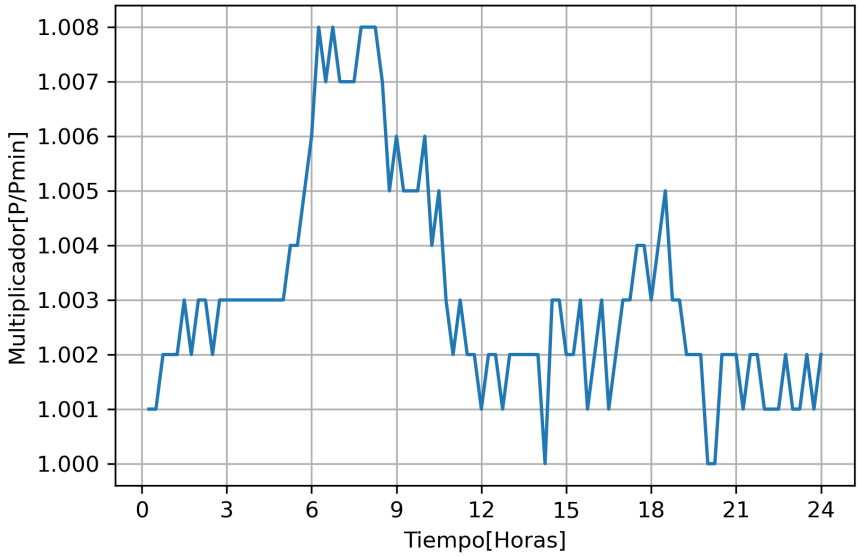

Figura 4.4: Patrón de funcionamiento de la PRV 33.036.

En EPANET la configuración de las válvulas funciona mediante reglas, imposibilitando el uso de patrones como en *WaterGems*. Una regla (Código 4.3) se define como una instrucción que ejecuta un elemento del sistema para una cierta condición.

Código 4.1: Regla en EPANET para cambiar la consigna de la válvula 31 a las 12 a.m.

```
1 RULE 1 #Numero de regla
2 IF SYSTEM CLOCKTIME = 12:00 AM #Condición de regla (En particular, condición
    ↪ horaria)
3 THEN VALVE 31 SETTING IS 32.27 #Resolución de regla (En particular se modifica la
    ↪ consigna de la valvula)
```

En base a lo planteado anteriormente, es necesario descomponer el patrón de cada una de las PRV del sistema en un conjunto de reglas que permitan replicar la operación del modelo en *WaterGems*, lo cual se realiza mediante un programa desarrollado en *Python* (Código 4.2), que genera un archivo de texto con las reglas en el formato presentado en el Código 4.3.

Código 4.2: Programa conversión de patrones.

```
import numpy as np
import pandas as pd
data=pd.read_csv("patronesss.CSV",sep=";") #Se lee los archivos con los patrones del sistema
valores=[35129,35131,35134,35136,35139,35141,35144,35146,35148,35150,35154,35156,
35158,35160,35162,35164,35166,35168,35170,35172,35174,35176,35381,35383]#Identificación de patrones de interés.
dataframe=pd.DataFrame([])
valvulas=[31,33,34,37,33036,33059,33065,33068,33074,33083,33091,33124,
33130,33872,33875,33878,33881,33882,33885,33889,33898,34668,35352,35377] #identificación de válvulas de interés.

for j in valores:
    aa=data.loc[data["ID"]==j,:]
    datos=np.array([])
    for i in range(aa.shape[0]): #Se leen y guardan los patrones de las valvulas deseadas
        datos=np.append(datos,[aa.iloc[i,1::]])
    dataframe=dataframe.append(pd.DataFrame(datos).T)
dataframe=dataframe.set_index([valvulas])

horas=["12","01","02","03","04","05","06","07","08","09","10","11"]
minutos=["00","15","30","45"]
mañana=["AM","PM"]#Se construye un set de horarios de la forma HH:MM AM/PM
horarios=np.array([])
for z in mañana:
    for i in horas:
        for j in minutos:
            horarios=np.append(horarios,i+":"+j+" "+z)
dataframe=dataframe.set_axis(horarios, axis=1, inplace=False)

f=open("reglas.txt","w")#Se genera un archivo de texto para guardar las reglas
indice=1
for i in dataframe.columns:#Se construyen las reglas del patron en formato EPANET
    f.write("RULE "+str(indice)+"\n")
    indice=indice+1
    f.write("IF SYSTEM CLOCKTIME = "+i+str("\n"))
    for j in dataframe.index.values:
        if j==31:
            f.write("THEN VALVE "+str(j)+" SETTING IS "+ str(dataframe.loc[j,i])+str("\n") )
        else:
            f.write("AND VALVE "+str(j)+" SETTING IS "+ str(dataframe.loc[j,i])+str("\n"))
    f.write("\n")

f.close()#Se guarda el archivo con las reglas
```

Código 4.3: Conversión de patrones.

```
RULE 1
IF SYSTEM CLOCKTIME = 12:00 AM
THEN VALVE 31 SETTING IS 32.27382
AND VALVE 33036 SETTING IS 23.126160000000002
AND VALVE 33059 SETTING IS 23.1438
AND VALVE 33074 SETTING IS 28.15911
AND VALVE 33124 SETTING IS 23.2704
AND VALVE 33130 SETTING IS 26.52185
AND VALVE 33882 SETTING IS 30.068479999999997
AND VALVE 33885 SETTING IS 29.09907
AND VALVE 33889 SETTING IS 30.5321
AND VALVE 33898 SETTING IS 23.91883

RULE 2
IF SYSTEM CLOCKTIME = 12:15 AM
THEN VALVE 31 SETTING IS 32.5811511
AND VALVE 33036 SETTING IS 23.1049264
AND VALVE 33059 SETTING IS 23.1149837
AND VALVE 33074 SETTING IS 28.4407011
.
.
```

Una vez construidas las reglas, se aplican al modelo y se realizan nuevamente las simulaciones. Se observa que la diferencia entre ambas curvas se ha reducido significativamente, con ambos modelos presentando el mismo comportamiento de presiones para la simulación dada, permitiendo validar el uso del modelo reconstruido para generar los datos de entrada para el algoritmo de clasificación (Figura 4.5).

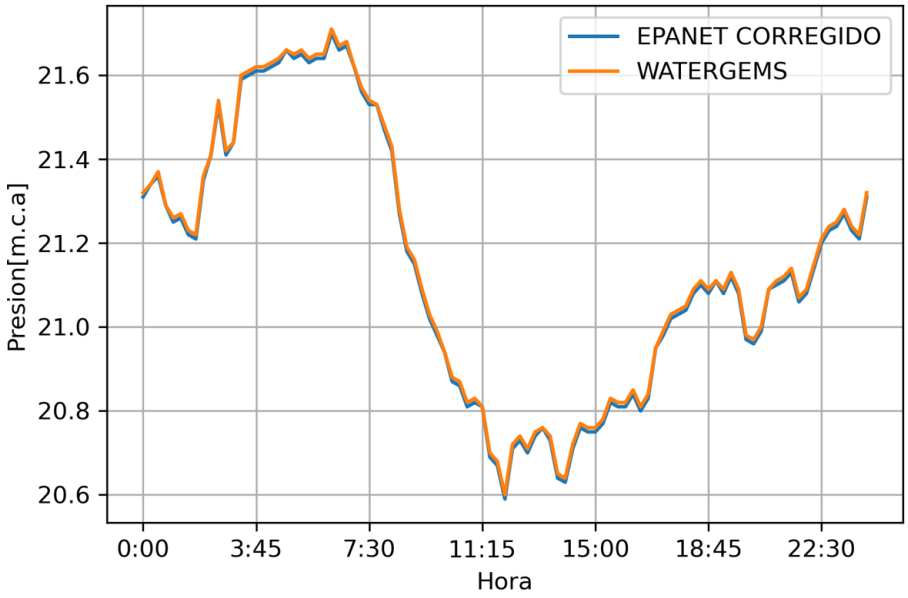

Figura 4.5: Presiones recalculadas en el nodo 11.459.

Capítulo 5

Resultados

5.1. Caso de estudio A: Conjunto acotado de tuberías

Los datos del primer caso de estudio propuesto se generan con los siguientes criterios de simulación:

- Se considera un conjunto aleatorio de cinco tuberías del sistema.
- Para cada simulación, una tubería presenta una fuga de caudal constante de 70 l/s.
- En cada simulación, la demanda base de todos los nodos se pondera por un valor aleatorio con una distribución normal $N(0; 0.1)$.
- Para cada tubería se realizan 20 simulaciones de fuga y 20 sin fugas.

Con los datos generados se obtiene un total de 7.889 escenarios del estado de la red, con lo cual se construyen las matrices de las ecuaciones 3.1 y 3.2 presentadas en el Capítulo 3, considerando un total de 5.522 escenarios para el entrenamiento de la red neuronal y un total de seis clases, que corresponden a las cinco tuberías escogidas, más una correspondiente al escenario sin fugas.

Con respecto a la arquitectura de la red, dado que no existe una regla general para determinar el número adecuado de neuronas y capas, se experimentó con diferentes tipos de redes y se evaluó su desempeño para escoger la mejor arquitectura, considerando tres capas ocultas para cada red. En la Tabla 5.1 se presentan algunas de las redes estudiadas.

Tabla 5.1: Arquitecturas de red estudiadas, se presentan el número de neuronas correspondiente a cada capa oculta de la red, en cada red la capa de entrada presenta 24 neuronas de entrada correspondiente al numero de sensores disponibles y seis neuronas de salida correspondiente al numero de clases.

Red neuronal	Capa 1	Capa 2	Capa 3	*F1-Score* (min, max, promedio)
A	4	6	6	0.84 / 0.99 / 0.96
B	**10**	**6**	**10**	**0.99 / 1.00 / 1.00**
C	20	30	20	0.98 / 0.99 / 1.00
D	15	5	7	0.93 / 1.00 / 0.98
E	7	20	10	0.89 / 1.00 / 0.97

Se escoge la arquitectura de la red B debido a que presenta el mejor desempeño y a que presenta un bajo número de conexiones, lo que disminuye el tiempo de cómputo del entrenamiento de la red.

En la Figura 5.1 se presenta el comportamiento de la red en su fase de entrenamiento con sus datos de prueba y validación, donde es posible notar que el modelo tiene un buen ajuste, y no se observa sobreajuste o subajuste (del inglés *overfitting* y *underfitting*; Figura 5.2) en el modelo, con un error de entrenamiento bajo y una baja diferencia entre los errores de entrenamiento y validación.

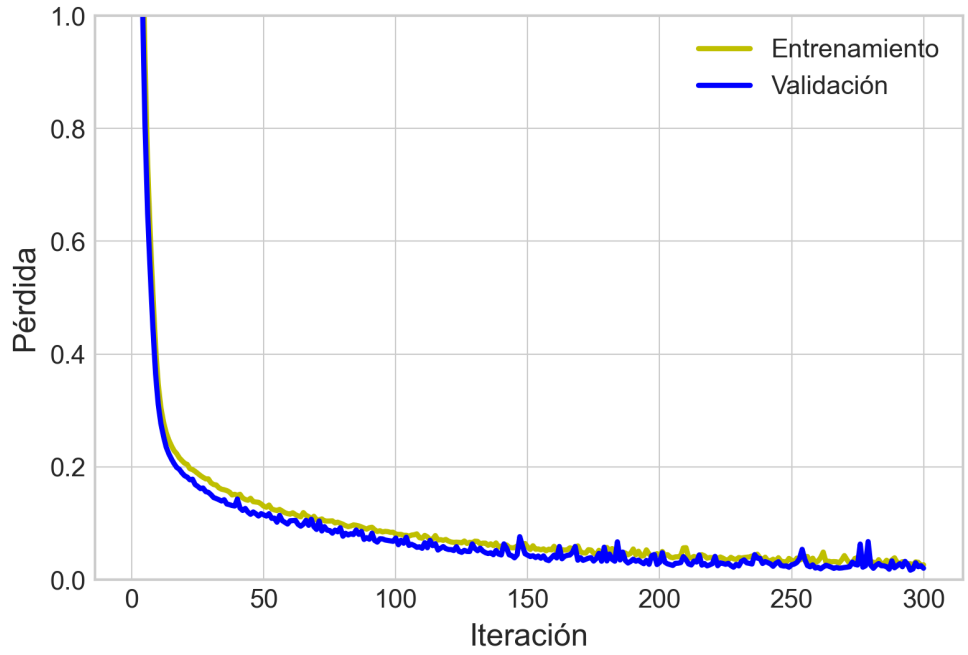

Figura 5.1: Curva de entrenamiento Red B. La pérdida se calculó con la función de entropía cruzada categórica y se consideró un 30 % de los datos como validación para cada iteración.

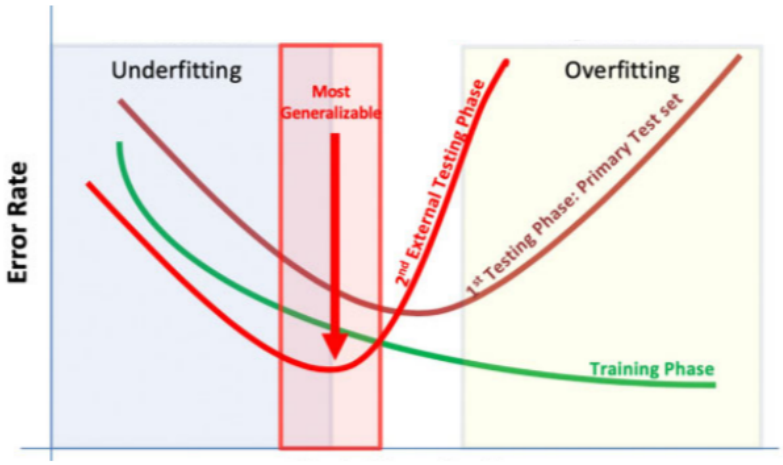

Figura 5.2: Patologías típicas de modelos de *machine learning*, *underfitting*, con baja diferencia de errores de entrenamiento y validación, pero con un alto error de entrenamiento. *Overfitting*, en el cual existe una alta diferencia entre los errores de entrenamiento y validación pero se observan bajos errores de entrenamiento.

Tabla 5.2: Desempeño del modelo propuesto para datos de prueba. Se presentan las métricas consideradas por clase y ponderadas según el numero de datos disponibles.

Clase	Precision	Recall	*F1-score*
0	1,000	1,000	1,000
18748	0,993	0,980	0,987
21030	1,000	0,994	0,997
21432	0,966	0,977	0,972
27972	0,980	1,000	0,990
30317	1,000	1,000	1,000
Accuracy	0,993	0,993	0,993
Promedio simple	0,990	0,992	0,991
Promedio ponderado	0,993	0,993	0,993

Se observa que el modelo presenta métricas similares para todas las clases con valores cercanos a 1. Si bien se observan valores altos de las métricas para cada clase del modelo y un buen ajuste a los datos, es posible inferir que el comportamiento se debe a que los datos utilizados en el modelo carecen de generalidad y existe una baja cantidad de clases a clasificar.

5.2. Caso de estudio B: Conjunto de datos ampliado

El segundo caso de estudio corresponde a una ampliación del conjunto de datos del caso de estudio A, con el fin de dar mayor generalidad a los datos de entrenamiento y generar un modelo que pueda ajustarse a la complejidad del problema. Los datos se generan con los siguientes criterios:

- Se considera un conjunto aleatorio de 100 tuberías del sistema.
- Para cada simulación, una tubería presenta una fuga de caudal variable modelada como un emisor (Ecuación 2.1). La fuga se modela para que la tubería considere el 24 % del caudal total del sistema como pérdidas de fuga.
- En cada simulación, la demanda base de todos los nodos se pondera por un valor aleatorio con una distribución normal de promedio 1 y desviación estándar de valor 0,3. Aumentando la variación de los datos de entrada al modelo.
- Para cada tubería se realizan 20 simulaciones de fuga y 20 simulaciones sin fugas.

Del conjunto de datos generados, se descartan aquellas tuberías de largo inferiores a 0,5 metros, debido a que corresponden a las tuberías que se generan artificialmente en el modelo para representar las válvulas existentes.

Se obtiene un total de 83.410 escenarios, que consideran 58.367 escenarios como datos de entrenamiento y 25.023 como datos de prueba, con un total de 87 clases, correspondientes a las tuberías seleccionadas del conjunto con una longitud mayor a 0,5 metros, más una clase de escenario sin fugas.

En la Figura 5.3, y teniendo en cuenta la Figura 5.2, se observa que la arquitectura de capas considerada en el caso de estudio anterior (Tabla 5.1, red B) presenta *underfitting* ante la nueva problemática, por lo cual, el modelo carece de la complejidad necesaria para representar el problema con los nuevos datos considerados, ante lo cual se considera una nueva arquitectura para la red. Dado que se requiere un modelo de mayor complejidad, se consideran redes de 3, 4 y 5 capas ocultas para este caso.

Además, en las Tablas 5.3, 5.4 y 5.5 se observa que al complejizar la red, el desempeño en la curva de entrenamiento mejora a lo presentado en la Figura 5.3 respecto a la red considerada en el caso A, reafirmando el *underfitting* que presentaba el modelo. Adicionalmente, se observa que la cantidad de conexiones entre neuronas de las redes con un buen comportamiento aumenta considerablemente respecto a los modelos presentados en el caso A.

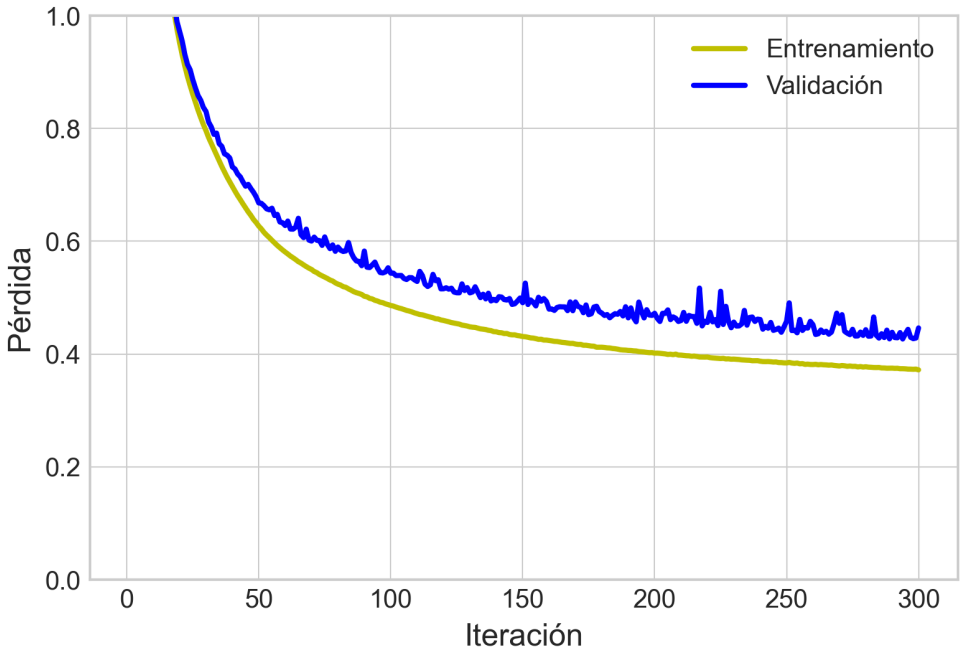

Figura 5.3: Curva de entrenamiento para red de caso de estudio A. La pérdida se calculó con la función de entropía cruzada categórica y se consideró un 20 % de los datos como validación para cada iteración.

Tabla 5.3: Arquitecturas de red estudiadas con tres capas ocultas, se presenta el número de neuronas correspondiente a cada capa oculta de la red, en cada red la capa de entrada presenta 24 neuronas de entrada correspondiente al número de sensores disponibles y 87 neuronas de salida correspondiente al número de clases. Como desempeño, se presentan las pérdidas de entrenamiento (Ent) y validación (Val) finales del modelo, donde una mayor precisión del algoritmo se ve reflejada con valores de pérdida cercanos a 0.

Red neuronal	Capa 1	Capa 2	Capa 3	Desempeño (Ent – Val)
A	30	50	20	0,154 – 0,175
B	50	70	30	0,112 – 0,162
C	15	30	40	0,191 – 0,199
D	40	30	60	0,135 – 0.164

Tabla 5.4: Arquitecturas de red estudiadas con cuatro capas ocultas.

Red	Capa 1	Capa 2	Capa 3	Capa 4	Desempeño (Ent – Val)
A	15	30	20	10	0,257 – 0,289
B	40	20	10	30	0,163 – 0,170
C	30	50	20	15	0,155 – 0,173
D	50	20	30	40	0,122 – 0,145
E	80	40	50	30	0,095 – 0,096

Tabla 5.5: Arquitecturas de red estudiadas con cinco capas ocultas.

Red	Capa 1	Capa 2	Capa 3	Capa 4	Capa 5	Desempeño (Ent – Val)
A	20	40	30	50	20	0,150 – 0,156
B	30	20	50	40	30	0,139 – 0,187
C	50	60	30	20	40	0,111 – 0,167
D	**80**	**40**	**50**	**30**	**30**	**0,092 – 0,096**

Con respecto a la arquitectura de la red, se observa que el mayor desempeño se obtiene al imponer una mayor cantidad de neuronas en la primera capa de la red, con el fin de extraer una gran cantidad de características en los datos y luego reducir el número de neuronas en las capas sucesivas para que las características más relevantes predominen en las capas sucesivas. Adicionalmente, en cuanto al número de capas del modelo, no se observa una correlación con el desempeño de las redes, siendo la distribución del número de neuronas en cada capa lo que condiciona principalmente el rendimiento de los modelos.

En la Figura 5.4 se presenta la curva de entrenamiento para el modelo escogido, correspondiente a la red neuronal D presentada en la Tabla 5.5. En la curva se aprecia que el desempeño del modelo mejora considerablemente de lo presentado en la Figura 5.3, sin embargo, se observan menores rendimientos que los presentados en el caso A.

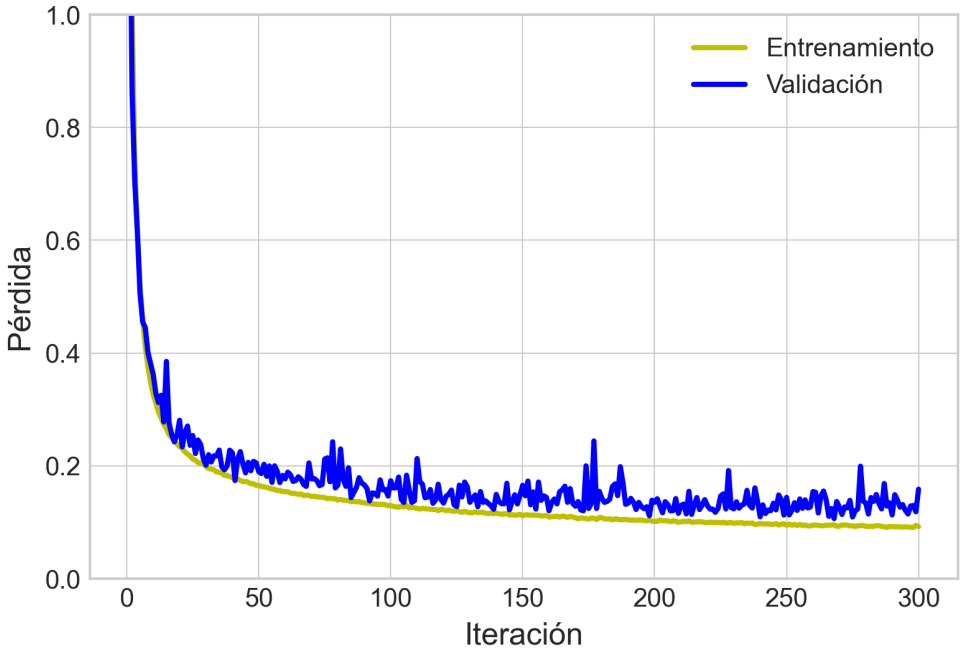

Figura 5.4: Curva de entrenamiento para red escogida caso B. La pérdida se calculó con la función de entropía cruzada categórica y se consideró un 20 % de los datos como validación para cada iteración.

La disminución del rendimiento del modelo respecto al caso de estudio A es un resultado esperado, dado que el conjunto de datos presenta una mayor variabilidad en sus escenarios debido a que las demandas en los nodos del sistema se ponderan por factores con una mayor desviación estándar, no obstante lo anterior el modelo continua presentando un buen desempeño en la detección de fugas del sistema.

En la Tabla 5.6, se presentan las 6 clases con menor desempeño del sistema. En anexo se presentan el resto de las clases. Se observa que en promedio el modelo presenta un buen desempeño, con un *F1-score* de un 95 %, lo que indica que no existe un sesgo del modelo hacia a predecir falsos negativos o falsos positivos y una buena tasa de detección de fugas. Las clases presentadas en la Tabla 5.6 tienen un desempeño significativamente inferior a lo obtenido en las demás clases del modelo. Para estas clases se observan 2 tipos de comportamiento:

- Clases con un alto *recall* y una baja *precision* (i): las métricas indican que en este caso, el modelo no logra detectar todos los escenarios con fuga de la tubería, sin embargo, cada vez que se detecta una fuga es con alta probabilidad una fuga.
- Clases con una alta *precision* y un bajo *recall* (ii): en este caso se observa que el modelo reconoce la mayoría de los escenarios de fuga para la tubería, sin embargo, tiende a detectar falsos escenarios de fuga.

Tabla 5.6: Desempeño del modelo propuesto para datos de prueba caso B. Se presentan las métricas consideradas para las clases con menor rendimiento, no obstante, las métricas promedio, correspondiente a las tres filas finales, consideran todas las clases consideradas.

Clase	*Precision*	*Recall*	*F1-score*
0	0,751	0,966	0,845
23053	0,942	0,643	0,764
23149	0,682	0,658	0,670
23473	0,259	0,783	0,390
31917	0,861	0,800	0,830
34913	0,714	0,735	0,725
Accuracy	0,956	0,956	0,956
Promedio simple	0,952	0,955	0,951
Promedio ponderado	0,962	0,956	0,957

Para el problema de detección de fugas, es preferible obtener clases con métricas como la del caso (i), dado que existe un costo asociado a verificar en terreno las fugas detectadas por el algoritmo, por lo cual, debe reducirse al máximo la cantidad de posibles falsas alarmas del sistema. También, dado que la fuga es un fenómeno físico permanente en la tubería, basta que el algoritmo la detecte en uno de los múltiples escenarios que se obtienen a lo largo del tiempo para localizar la fuga.

Para estudiar si existe alguna correlación entre el bajo desempeño del algoritmo en estas tuberías y la ubicación espacial de estas, en las Figuras 5.5, 5.6, 5.7 y 5.9 se presenta la ubicación de las clases presentadas en la Tabla 5.6.

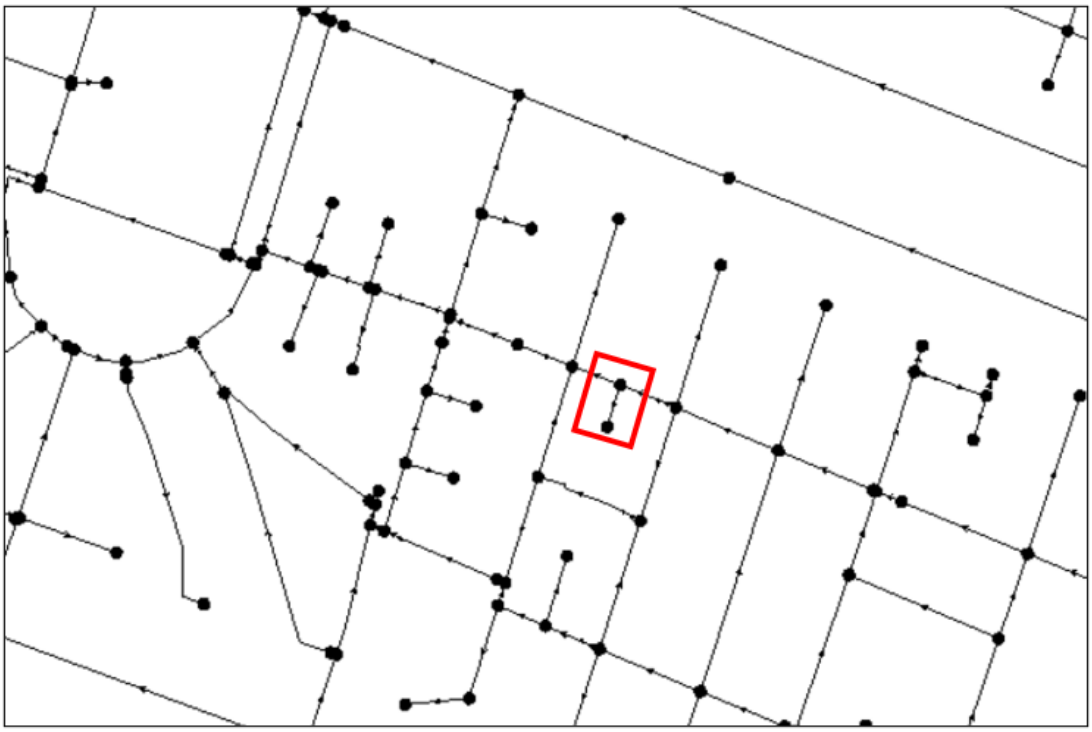

Figura 5.5: Ubicación de la tubería 23053 en el sistema. En rojo se presenta la tubería descrita, identificándose igualmente con circunferencias negras los nodos del sistema.

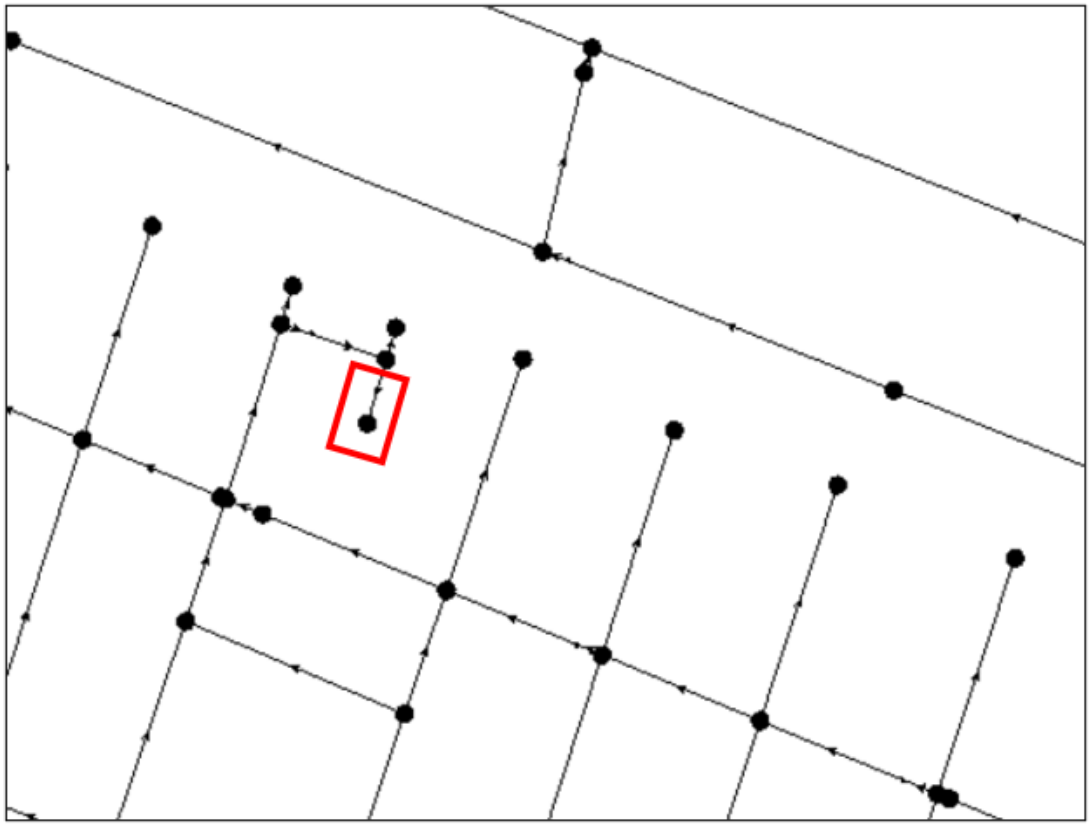

Figura 5.6: Ubicación de la tubería 23149 en el sistema.

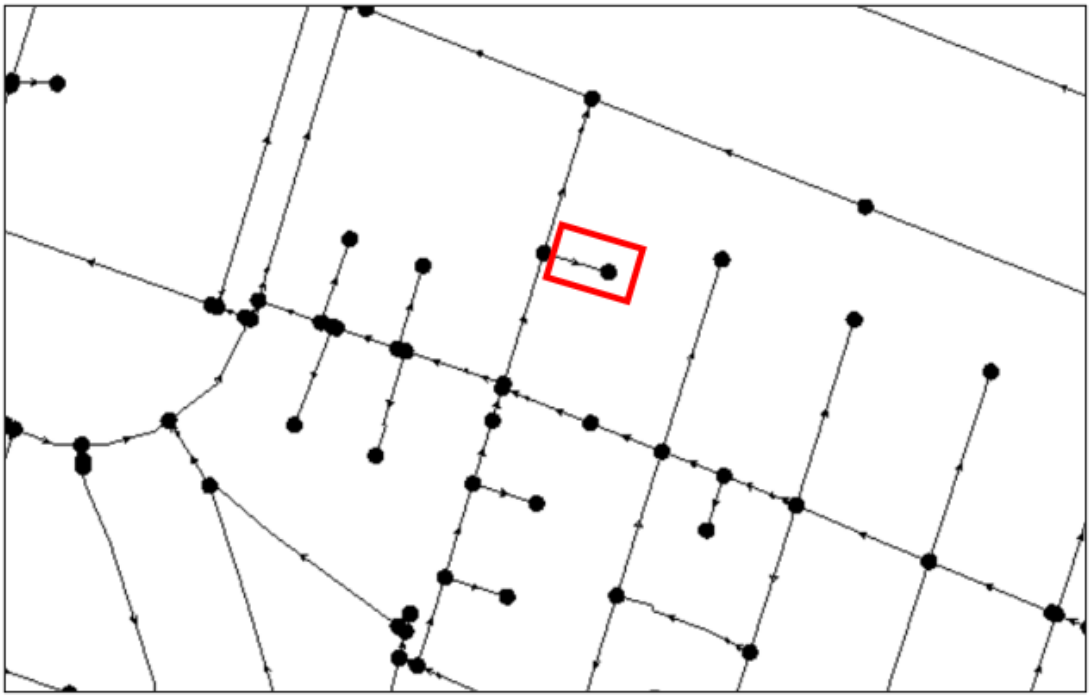

Figura 5.7: Ubicación de la tubería 23473 en el sistema.

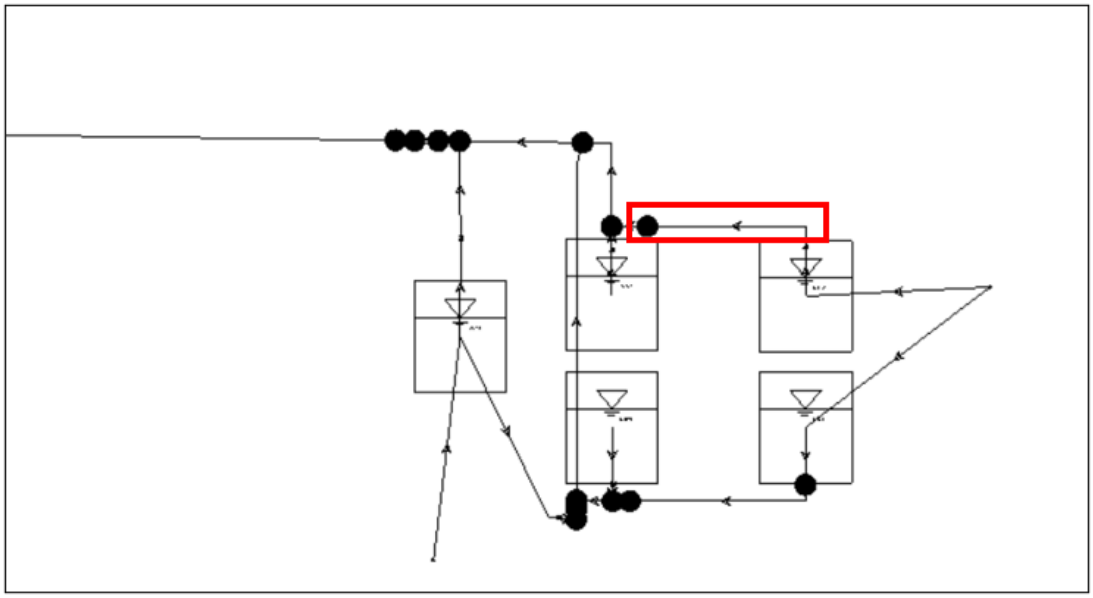

Figura 5.8: Ubicación de la tubería 31917 en el sistema. La ubicación de esta tubería corresponde al sistema de estanques que alimentan la red de distribución.

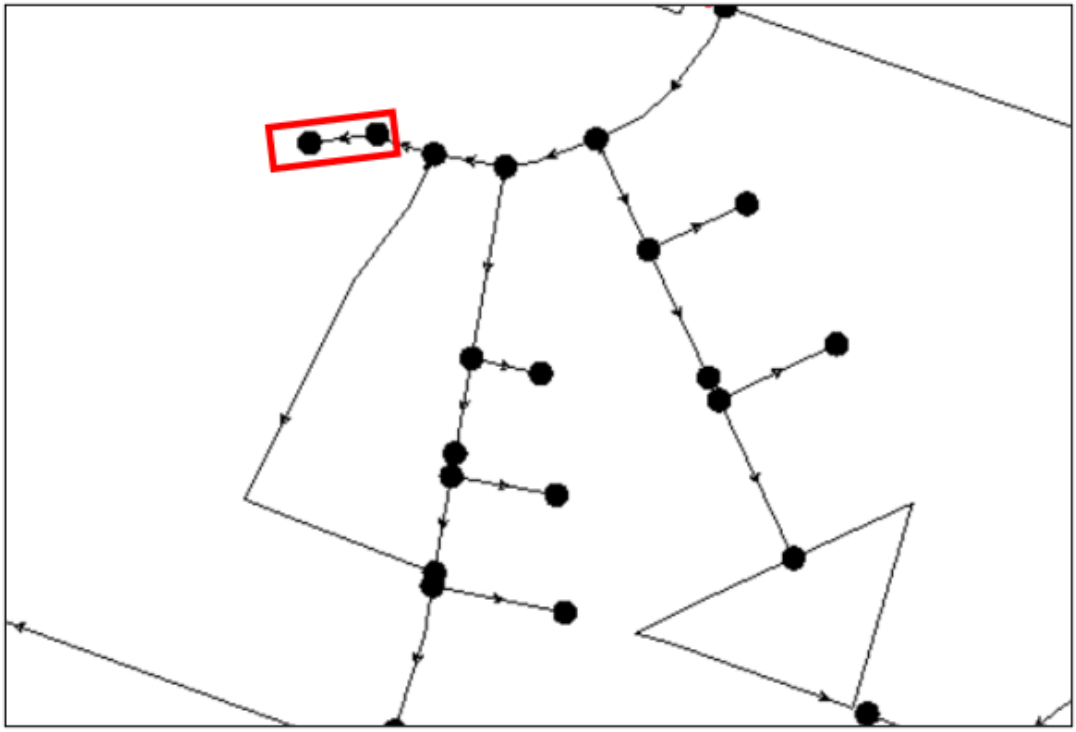

Figura 5.9: Ubicación de la tubería 34913 en el sistema.

Como se puede observar, las tuberías corresponden a tramos terminales de la red. Para estas tuberías es posible que el algoritmo no sea capaz de distinguir entre las variaciones de presión en el sistema producto de una fuga en la tubería y una variación de la demanda en su nodo adyacente, presentando dificultades para detectar la fuga de manera consistente en estas tuberías.

En el caso de la tubería presentada en la Figura 5.8, se tiene una tubería en el sistema de estanques de la red, el cual se encuentra a una distancia significativa de cualquier sensor de presión en la red, por lo cual el algoritmo presenta problemas para detectar fugas en las tuberías en esta zona dado que los cambios producidos por las fugas en este sector no son representadas en los sensores de presión del sistema.

Finalmente, en base a lo presentado con anterioridad se concluye que la ubicación espacial y la disponibilidad de sensores en el sistema están relacionados con el desempeño que tendrá el algoritmo para detectar fugas en las tuberías. En especifico, aquellas tuberías alejadas del sistema de sensores.

5.3. Caso de estudio C: Conjunto de datos ampliado con ocurrencia simultánea de fugas

El último caso de estudio corresponde a un conjunto de datos que busca representar de forma más realista el problema en comparación al caso anterior, considerando fugas que se desarrollan simultáneamente en diferentes tuberías y a caudales variables. Los datos se generan los siguientes criterios:

- Se considera un conjunto aleatorio de 100 tuberías del sistema.
- Para cada simulación, se considera un subconjunto de 1 a 5 tuberías que presentan falla. Al igual que en el caso de estudio anterior se considera el 24 % del caudal total del sistema como pérdidas por fugas, las que son repartidas entre el subconjunto de tuberías escogidas y modeladas como un emisor.
- Se mantiene el criterio del caso de estudio anterior respecto a la variación de demanda base en cada nodo del sistema.
- Se realizan 2000 simulaciones para escenarios de fuga y 20 simulaciones de escenarios sin fugas.

Los datos generados por las simulaciones generan un total de 2000 subconjuntos de 1 a 5 tuberías de los $9,1296 \times 10^9$ posibles, por lo cual, en este caso, la mayoría de escenarios que debe detectar el algoritmo no estará presente en el entrenamiento del modelo. Se obtiene un total de 83.410 escenarios de los cuales se consideran 61.013 como datos de entrenamiento y 26.149 como datos de prueba con un total de 91 clases.

Para evaluar el desempeño de la red se consideran las métricas de *precision* y *recall* en el entrenamiento, ya que la salida del modelo corresponde a un vector de 91 valores binarios de los cuales un número entre 0 y 5 valores pueden ser considerados verdaderos, provocando un desbalance entre positivos y negativos en el vector de salida. Al calcular la función de pérdida en cada iteración del entrenamiento se obtienen valores muy bajos producto de la predominancia que existe de valores negativos, lo que implica que, independientemente del valor obtenido en el vector de salida del modelo, este es muy similar al vector de validación si se consideran los 0 del vector. Ante esto se consideran las métricas nombradas anteriormente, dado que permiten evaluar el modelo en base a los valores positivos que se generan, restando significancia al desbalance producido.

Para determinar la arquitectura de la red se considera como punto de partida la red neuronal presentada en la Tabla 5.5 para el caso anterior.

En las Figuras 5.10 y 5.11 se observa que la métrica *precision* alcanza valores altos rápidamente en el entrenamiento en comparación a la métrica *recall*, esto quiere decir que el modelo puede ajustarse con una menor cantidad de datos para comenzar a detectar fugas en el sistema, sin embargo, requiere una mayor cantidad para ajustarse a detectar la totalidad de fugas.

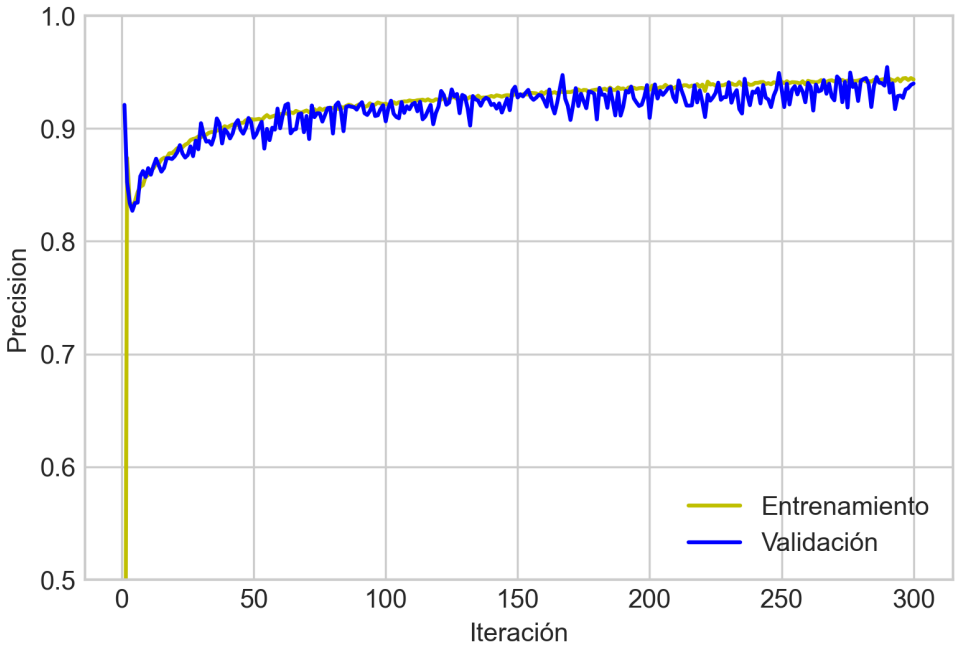

Figura 5.10: Métrica *precision* para el entrenamiento de la red. A medida que *precision* aumenta, la cantidad de fugas detectadas que son efectivamente fugas aumenta.

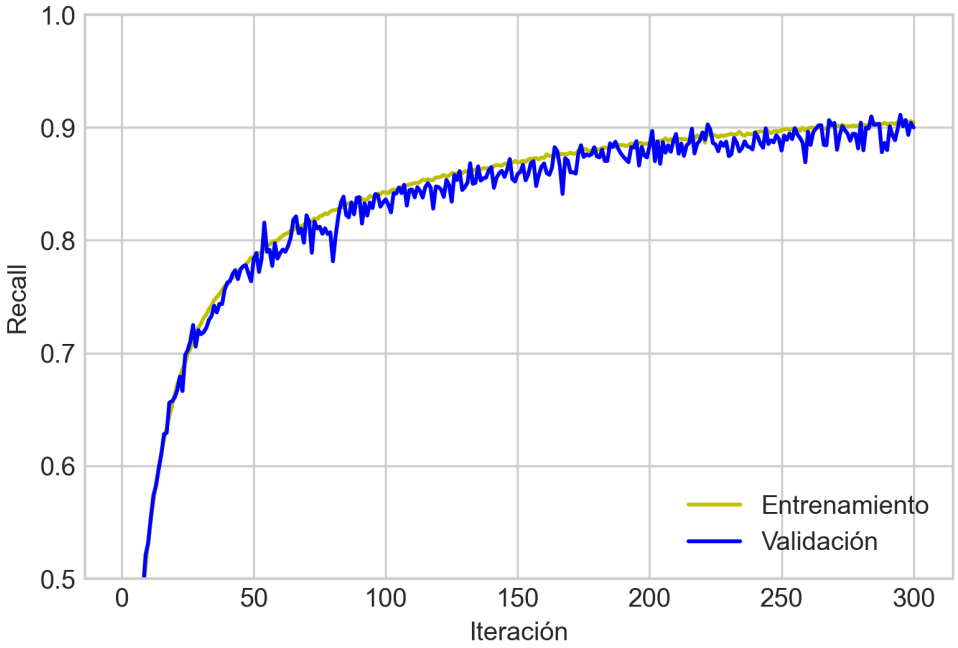

Figura 5.11: Métrica *recall* para el entrenamiento de la red, a mayor valor de *recall*, la cantidad de fugas detectadas del total de fugas aumenta.

En la Tabla 5.7 se observa que las métricas tienen valores similares para datos de entrenamiento y validación, lo que permite descartar un sobreajuste del modelo en los datos de entrenamiento. Adicionalmente, se observa un desempeño similar al de la red presentada en la Tabla 5.6, con una disminución del 1 % en *precision* y 5 % en *recall*. No obstante, se mantiene un buen desempeño con la arquitectura presentada. Se estudia aumentar el número de neuronas de la red para verificar si se obtiene un mayor desempeño.

Tabla 5.7: Desempeño del modelo propuesto para datos de prueba caso C. Se presentan las métricas *precision* y *recall* para la iteración del entrenamiento.

	Precision	*Recall*
Entrenamiento	0,943	0,904
Validación	0,939	0,900

En la Tabla 5.8 al aumentar el número de neuronas de la red manteniendo el número de capas ocultas se mejora considerablemente el desempeño del modelo, registrando un aumento del 5 % en *precision* y un 9 % en *recall*, ambas métricas alcanzando un valor del 99 %.

Tabla 5.8: Arquitecturas de red estudiadas caso C. Se consideran cinco capas ocultas para cada red. La capa de entrada contiene 24 neuronas correspondiente al número de sensores del sistema y la capa de salida contiene 91 neuronas correspondiente al número de clases a clasificar. Se considera la función ReLu para las capas ocultas y la función *Sigmoid* para la capa de salida. Como métrica de desempeño se considera *F1-Score*.

Red	Capa 1	Capa 2	Capa 3	Capa 4	Capa 5	Desempeño(Ent – Val)
A	80	60	80	50	70	0,975 – 0,968
B	30	20	50	40	30	0,964 – 0,967
C	80	55	40	60	30	0,964 – 0,967
D	50	30	80	100	65	0,981 – 0,978
E	**200**	**100**	**50**	**70**	**120**	**0,991 – 0,990**

En las Figuras 5.12 y 5.13 se presentan las curvas de entrenamiento para las métricas *recall* y *precision* del modelo. Al analizar las curvas se observa que los datos de entrenamiento y validación tienen comportamientos similares y presentan valores altos, lo que permite descartar un sobreajuste del modelo.

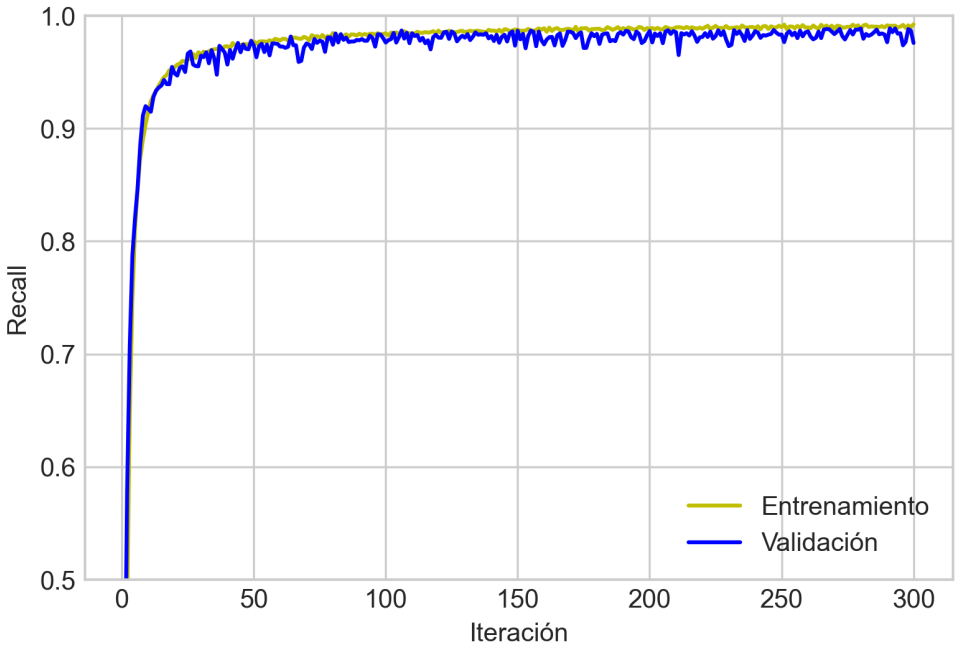

Figura 5.12: Curva de entrenamiento para red escogida. La métrica considerada es *recall* y se considera un 30 % de los datos como datos de validación.

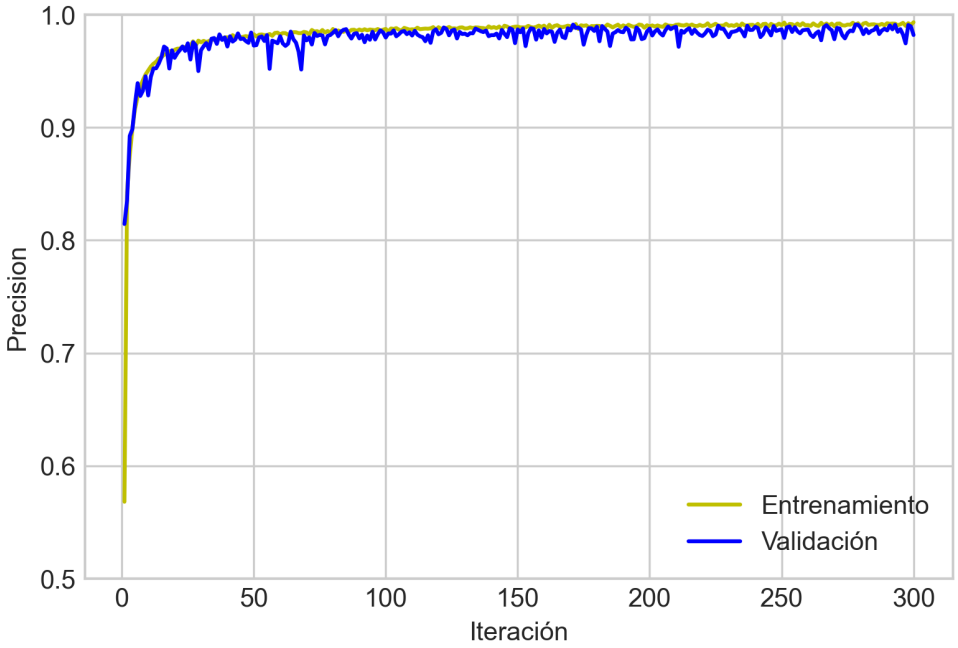

Figura 5.13: Curva de entrenamiento para red escogida. La métrica considerada es *precision*.

En la Tabla 5.9 presentan aquellas clases con un desempeño menor al 95 % de detección de fugas. Se observa que, en promedio, el modelo presenta un F1-Score de 98 %, con una alta tasa de detección de fugas. Adicionalmente, es posible notar que las clases con rendimientos más bajos no se alejan significativamente del promedio, manteniéndose para todas las tuberías del conjuntos de datos una tasa de detección de al menos un 90 %. También, es posible distinguir que para la mayoría de las clases presentada la métrica *recall* tiene un mayor valor que *precision*, lo que indica que para estas clases el modelo prioriza no dar falsas alarmas por sobre detectar las fugas para todos los escenarios dados. Este comportamiento se busca obtener en el modelo dada la inversión que significa corroborar la existencia de la fuga en terreno.

Tabla 5.9: Clases con desempeño inferior al 95 %. Se presentan las métricas consideradas para las clases con menor rendimiento, no obstante, las métricas promedio, correspondiente a las dos filas finales, consideran todas las clases consideradas.

Clase	*Precision*	*Recall*	F1-score
0	0,817	0,901	0,859
15860	0,901	1,000	0,949
18201	0,908	0,907	0,908
29271	0,926	0,976	0,945
30337	0,952	0,931	0,936
31779	0,914	0,975	0,943
Promedio simple	0,979	0,986	0,982
Promedio ponderado	0,981	0,986	0,983

A pesar de que se obtiene un buen desempeño para detectar todas las tuberías seleccionadas, al igual que en el caso anterior, en las figuras 5.14, 5.15, 5.16, 5.17 y 5.18 se presentan las ubicaciones de las tuberías en el sistema.

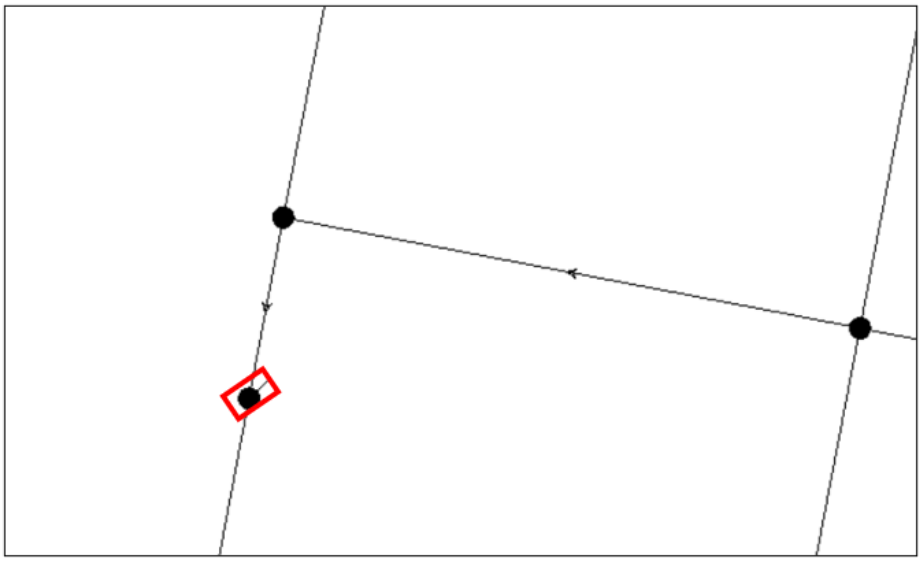

Figura 5.14: Ubicación de la tubería 15860 en el sistema. En rojo se presenta la tubería descrita, identificándose igualmente con circunferencias negras los nodos del sistema.

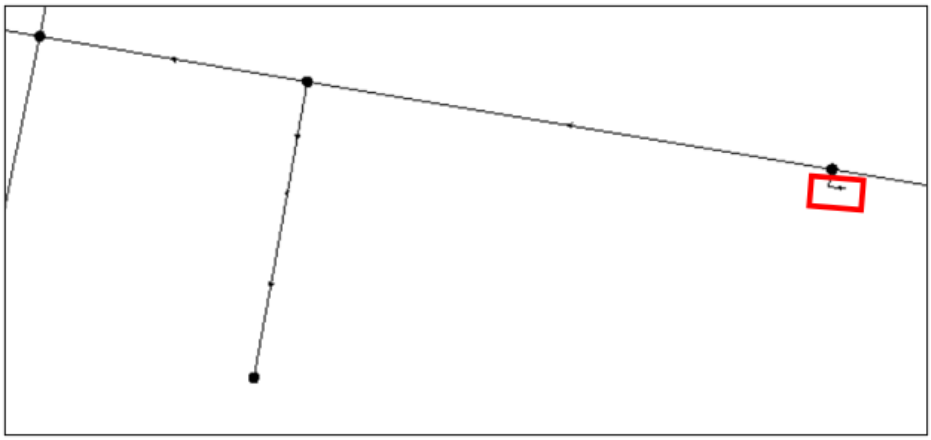

Figura 5.15: Ubicación de la tubería 18201 en el sistema.

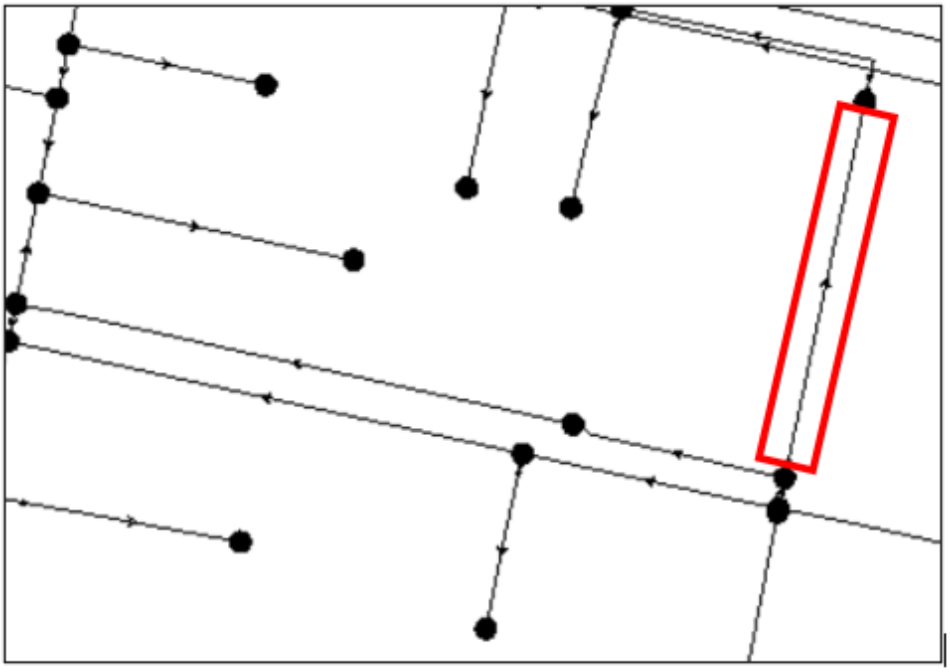

Figura 5.16: Ubicación de la tubería 29271 en el sistema.

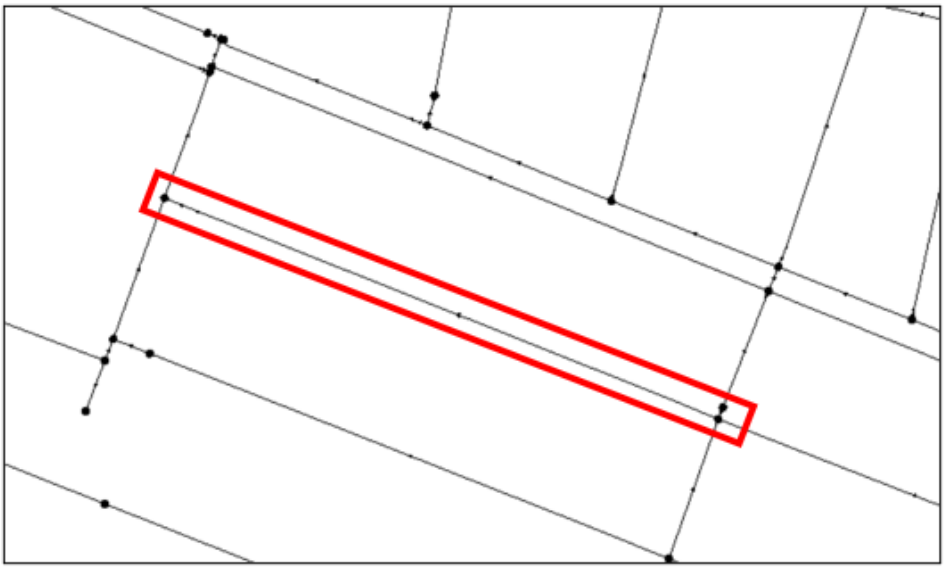

Figura 5.17: Ubicación de la tubería 30337 en el sistema.

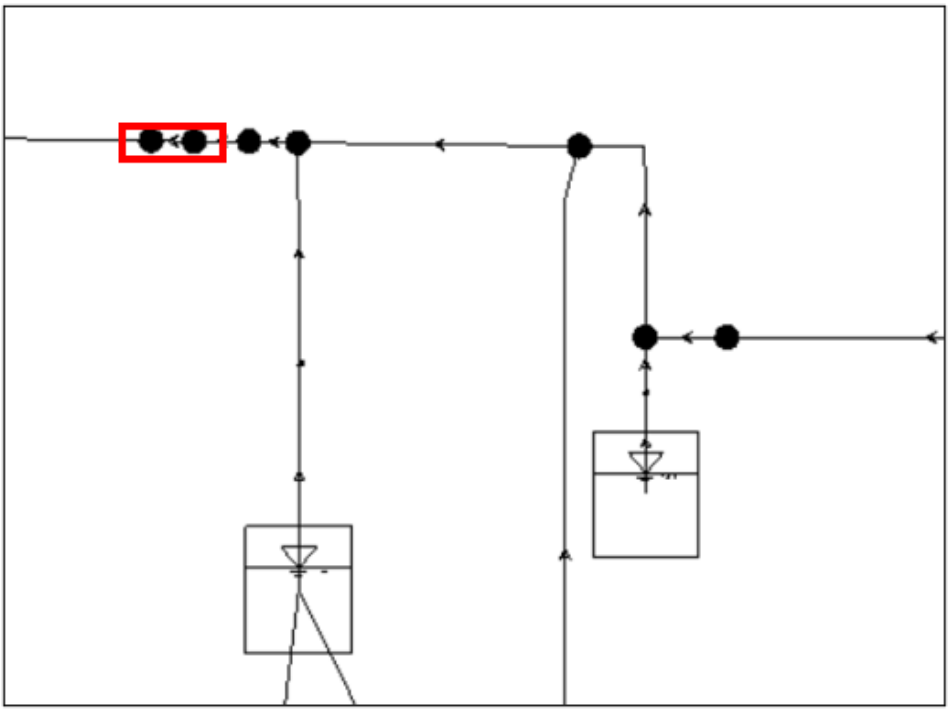

Figura 5.18: Ubicación de la tubería 31779 en el sistema. La ubicación de esta tubería corresponde al sistema de estanques de la red.

En las Figuras 5.14, 5.15 y 5.18 se observan tuberías similares a las identificadas en el caso de estudio B, correspondientes a tramos terminales y tuberías alejadas del sistema de sensores de la red, lo que permite reafirmar que el algoritmo presenta mayores dificultades para detectar fugas en este tipo de tuberías del sistema. En cuanto a las tuberías presentadas en las Figuras 5.16 y 5.17 no se observa un patrón atribuible al desempeño de las clases.

Finalmente, con respecto a la clase 0, es un resultado esperado que corresponda a la clase con el desempeño más bajo, debido a que, para cada error que cometa el algoritmo al identificar una fuga en cualquiera de las 90 clases restantes, ésta tiene una incidencia directa sobre el desempeño sobre la clase 0, al estar relacionada al escenario de ausencia de fugas.

5.4. Tiempo de cómputo del algoritmo

Otro aspecto relevante a considerar para la resolución del problema es el tiempo de cómputo necesario para entrenar las redes con los datos obtenidos. Si bien este parámetro no incide en el desempeño que tienen los modelos para detectar fugas, si son cruciales para evaluar la factibilidad de la metodología para resolver problemas reales.

Como se observa en la Tabla 5.10, el tiempo de entrenamiento de los modelos es bajo, donde el modelo de mayor complejidad presenta un tiempo de entrenamiento de 11 minutos 19 segundos en un equipo con un procesador de 4 núcleos de 2.3 GHz y 8 GB de memoria RAM, lo que podría considerarse un equipo de uso cotidiano.

Tabla 5.10: Tiempo de cómputo para casos de estudio. El tiempo considerado corresponde al tiempo de entrenamiento de cada modelo. La columna neuronas representa la totalidad de neuronas de la red excluyendo las capas de entrada y salida.

Red neuronal	Capas ocultas	Neuronas	Tiempo de cómputo [segundos]
Caso 1.B	3	26	25,84
Caso 2.D	5	230	441,24
Caso 3.E	5	540	679,32

5.5. Comparación de resultados

En esta sección se comparan algunos resultados obtenidos con el estudio de Gárate (2020) para la red de estudio. Algunos de los aspectos analizados son el tiempo de computo del algoritmo, la complejidad del problema resuelto y el desempeño de los modelos.

En primer lugar, con respecto a los tiempos de procesamiento, Gárate (2020) presentó 2 modelos de SVM como aquellos que presentan buenos resultados, los cuales denomina 'Segunda' y 'Quinta SVM', asociadas a los ordenamientos 1.2 y 2.2 de su estudio (en adelante, los modelos se denominaran según su ordenamiento asociado).

Los tiempos asociados a cada modelo pueden observarse en la Figura 5.19. En general, se observa que los modelos de redes neuronales tienen un menor tiempo de cómputo que las SVM, en especial con respecto al modelo del ordenamiento 2.2, donde la red 3.E presenta una disminución de tiempo del 84 % y un 46 % respecto al ordenamiento 1.2.

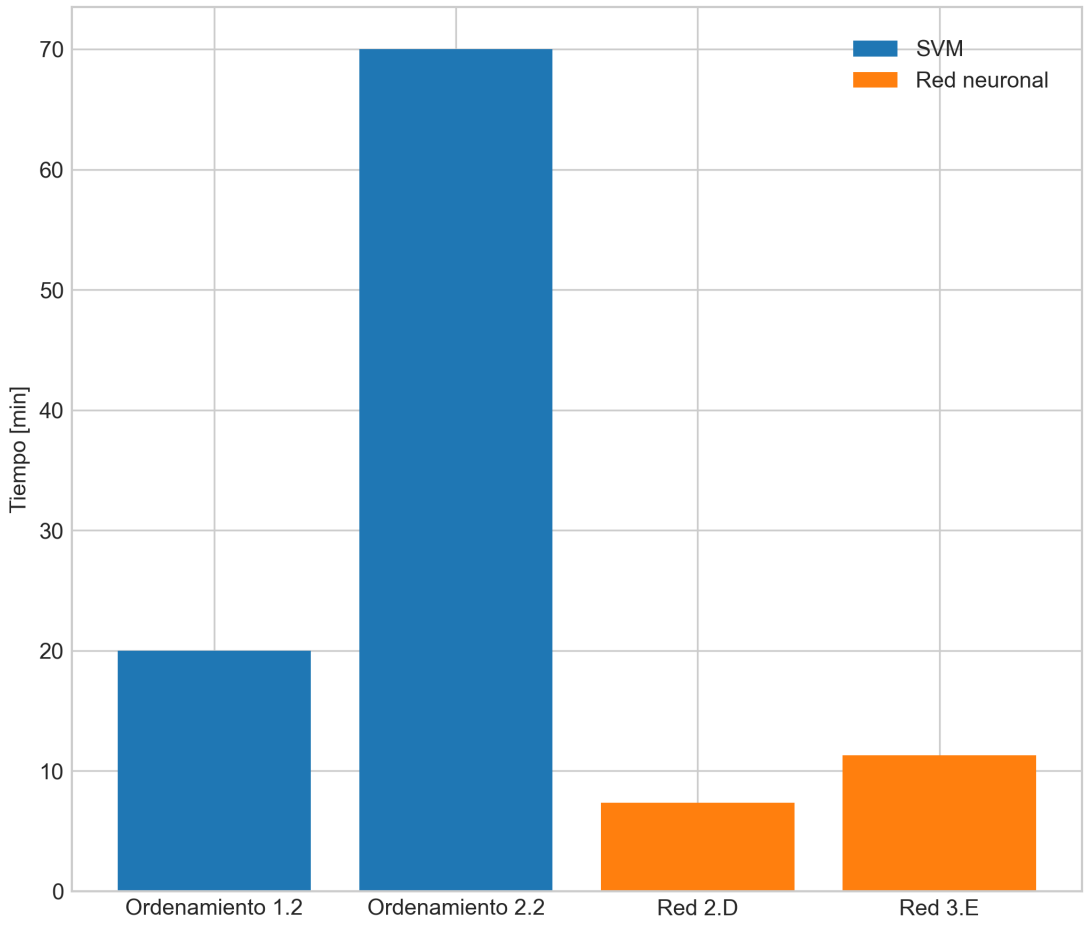

Figura 5.19: Tiempo de procesamiento modelos de SVM estudiados por Gárate (2020) y redes neuronales estudiadas. Los modelos de SVM considerados son aquellos que el autor define como óptimos para el problema de detección de fugas.

Otro aspecto a considerar para comparar ambas metodologías, es la complejidad del problema resuelto. El problema planteado por Gárate (2020) consiste en detectar posibles fugas en un conjunto de 16 nodos del sistema en diferentes horas del día. En este trabajo las principales modificaciones al problema planteado son las siguientes:

- Los modelos se plantearon con la finalidad de detectar la fuga en una tubería en específico del sistema por sobre asociar la fuga a un nodo cercano, lo cual entrega mayor información al usuario respecto al estado de la red.
- Se consideró un mayor número de clases, de 16 nodos del sistema a mas de 90 tuberías del sistema, lo que significa un aumento de aproximadamente 6 veces en el número de clases de salida.
- Se consideraron fugas de diferente caudal, las cuales pueden presentarse simultáneamente en distintas tuberías de la red y permanecen permanentemente en la tubería cuyo caudal dependerá de la presión de la tubería con fuga.

Considerando las modificaciones presentadas, es posible notar que, pese a resolver un pro-

blema significativamente más complejo, los modelos de redes neuronales presentan menores tiempos de procesamiento que los modelos de SVM.

Con Respecto al desempeño de los modelos, Garate (2020) concluyó que sus modelos con mayor desempeño presentan un sobreajuste a los datos de entrenamiento, por lo cual plantea re-entrenar sus modelos con una mayor tolerancia al error. En las Tablas 5.11 y 5.12 se presenta el desempeño presentado por Gárate (2020) para sus modelos de SVM.

Tabla 5.11: Desempeño **Segunda SVM** en base al hiper parámetro C del modelo. El modelo presentado en la segunda fila corresponde al modelo presentado por el autor al re-entrenar el modelo, al detectar un sobreajuste de los datos de entrenamiento. Adicionalmente, el autor recomienda utilizar un valor de C de 15000 o 20000, por lo cual el desempeño esperado estaría entre un 50 % y 90 %.

Hiper parámetro C	Tasa de detección de fugas
30000	90 %
10000	50 %

Tabla 5.12: Desempeño **Quinta SVM** en base al hiper parámetro C del modelo. El modelo presentado en la segunda fila corresponde al modelo presentado por el autor al re-entrenar el modelo, al detectar un sobreajuste de los datos de entrenamiento. Adicionalmente, el autor recomienda utilizar un valor de C de 5000, por lo cual el desempeño esperado estará entre un 65 % y 85 %.

Hiper parámetro C	Tasa de detección de fugas
30000	90 %
1000	65 %
10000	80 %

Como se observa, la información presentada por Gárate (2020) para evaluar el desempeño de los modelos es limitada, no obstante, al comparar los resultados con lo presentado en las Tablas 5.6 y 5.7 se aprecia una mejora significativa, con los modelos 2.D y 3.E presentando una tasa de aciertos del 95 % y 97 %, respectivamente para datos de prueba y descartando un sobre-ajuste hacia los datos de entrenamiento.

Finalmente, en términos generales y por las razones anteriormente expuestas, es posible afirmar que los modelos de redes neuronales presentados en este estudio se ajustan de mejor manera para la resolución del problema de detección de fugas en la red de estudio.

Capítulo 6

Conclusiones

Los resultados de este estudio indican que los modelos de redes neuronales presentan un buen desempeño (promedio de 97 % y valores mayores a 90 % para la totalidad de las clases en el caso de estudio de mayor complejidad), por lo cual, se valida su utilización para la detección de fugas en redes de gran tamaño.

Entre las ventajas del algoritmo se encuentra su bajo tiempo de cómputo y versatilidad, dado que los modelos pueden ajustarse fácilmente a un nuevo problema al cambiar su arquitectura, es decir, modificando su cantidad de neuronas y capas sin necesidad de modificar el ordenamiento de los datos de entrada o salida. Adicionalmente, el tiempo de cómputo del entrenamiento de los modelos es bajo (inferior a 15 minutos para las redes estudiadas), ejecutado en un ordenador de uso cotidiano.

Un aspecto que compromete la viabilidad de la metodología es el tiempo necesario para generar los datos de entrada del modelo, que en este estudio se realizó mediante simulaciones hidráulicas y constituyó un tiempo aproximado de 15 horas para un conjunto de 100 tuberías, lo que corresponde al 2 % de las tuberías de la red. Considerando que las simulaciones fueron realizadas en la infraestructura del NLHPC, cuyo hardware es significativamente superior a un equipo de uso cotidiano, se concluye que se requerirá de un tiempo significativamente superior para evaluar el estado completo de la red estudiada, que requiere generar simulaciones para cada tubería del sistema.

En cuanto a la calibración del modelo, se mantiene el modelo calibrado por Gárate (2020) y se desestima considerar otra metodología para calibrar el modelo debido a que producto de la insuficiencia de datos, y en particular de las demandas del sistema, cualquier metodología de calibración genera resultados artificiales al no conocer y tener un punto de comparación con el comportamiento real de la red. En esta misma línea, se mantiene la conclusión de Gárate (2020) respecto a la importancia que tiene una buena calibración del modelo si se busca aplicar la metodología con fines operacionales. En un modelo con una buena calibración, las presiones de la red deben coincidir con las observaciones que se tienen en terreno mientras que, simultáneamente puede representarse el estado operacional de las tuberías, mediante el coeficiente de rugosidad; y la demanda de los nodos de la red.

Una debilidad de la metodología es la cantidad de información requerida para entrenar

los modelos. En este trabajo se generaron mediante simulación hidráulica, sin embargo, los datos idealmente deberían ser medidos en terreno. En particular, la información que presenta mayor dificultad en ser obtenida es la de salida de los modelos, es decir, el historial de fugas que han presentado las tuberías del sistema debido a que, o bien no se buscan activamente fugas en terreno para las redes y sólo se reparan aquellas fugas criticas reportadas por los usuarios, o no se tiene un registro de los reportes/reparaciones de fugas en el sistema.

Otro aspecto a destacar es que los modelos, además de detectar fugas, permitieron determinar que existe un grupo de tuberías en el cual los sensores no cuantifican con claridad cambios en las tuberías respecto a escenarios con o sin fugas, en particular aquellas tuberías pertenecientes a tramos terminales o alejadas del sistema donde el algoritmo presenta menores tasas de detección de fugas, concluyendo que la información entregada por el sistema también podría ser utilizada para evaluar la sensorización del sistema.

Finalmente, a pesar de que en el caso de estudio C se presenta una problemática de mayor complejidad y más cercana a la realidad que en el caso de estudio B. Ambos enfoques pueden ser utilizados para resolver el problema de detección de fugas, presentando una buena tasa de detección.

Recomendaciones para la aplicabilidad de la metodología en casos reales

Como ya se mencionó anteriormente, la principal limitación de aplicar la metodología propuesta para resolver problemas reales es la falta de información en específico de fugas detectadas en terreno del sistema, la cual es escasa o inexistente. Para abordar esta problemática se recomienda considerar un sistema híbrido de entrenamiento para los modelos, con datos de simulación hidráulica y datos de terreno de la siguiente forma:

- En primer lugar, generar un modelo entrenado netamente con datos de simulación hidráulica, para generar las conexiones sinápticas en la red que contienen el conocimiento del comportamiento de las fugas en el sistema.
- Con el modelo entrenado para reconocer las características de las fugas, se recomienda re-entrenar el modelo con los datos de terreno disponible utilizando la red ya generada como punto de partida. Para aquello, se recomienda un entrenamiento con una baja tasa de aprendizaje y congelando las conexiones en las capas ocultas, permitiendo variar las conexiones sólo para las capas de entrada y salida del modelo, lo cual permite que la red mantenga el conocimiento aprendido de los datos de simulación e incorpore lentamente al modelo las discrepancias que puedan darse por diferencias entre el modelo hidráulico y el comportamiento real de la red.
- El proceso anterior debe repetirse constantemente en el tiempo con la nueva información que se genere en terreno con el fin de retro-alimentar el modelo y volverlo cada vez más robusto.

Para redes de tamaño extenso se recomienda aplicar el algoritmo en subconjuntos de

tuberías, como se realizó en el presente trabajo. El subconjunto puede definirse según una zona de interés a evaluar, criticidad de las tuberías o a un límite espacial. Esto permite reducir la cantidad de datos necesaria para el entrenamiento de los modelos y también reduce los costos al acotar la cantidad de tuberías de las cuales deben tenerse datos de terreno. En cuanto a la distribución de sensores, se recomienda una distribución espacial con la mayor homogeneidad espacial posible, con el fin de evitar tuberías aisladas del sistema de sensores, que tienen una menor tasa de detección por el algoritmo.

Finalmente, se destaca que el desempeño del algoritmo para problemáticas reales dependerá en gran medida de la calidad de la calibración del modelo hidráulico de la red para sus primeras fases de entrenamiento. A medida que se incorporan o se dispone de datos de terreno el, efecto de la calibración disminuye y aumenta la efectividad del algoritmo, debido a que el comportamiento real de la red se incorpora en el modelo a través de los datos. Es por esto que, si las empresas sanitarias no disponen de información suficiente, el uso de esta metodología debe considerarse como una solución de mediano o largo plazo para monitorear el estado de fugas en redes de agua potable.

Bibliografía

Adedeji, K. B., Hamam, Y., Abe, B. T., y Abu-Mahfouz, A. M. (2018). *Pressure Management Strategies for Water Loss Reduction in Large-Scale Water Piping Networks: A Review*. Springer Singapore. doi: 10.1007/978-981-10-7218-5_33

Albawi, S., Abed Mohammed, T., y ALZAWI, S. (2017, 08). Understanding of a convolutional neural network.. doi: 10.1109/ICEngTechnol.2017.8308186

Baader, J., Fallis, P., Hübschen, K., Klingel, P., Knobloch, A., Laures, C., ... Ziegler, D. (2011). *Guidelines for water loss reduction - a focus on pressure management*. Deutsche Gesellschaft für Internationale Zusammenarbeit (GIZ) GmbH.

Chan, T. K., Chin, C. S., y Zhong, X. (2019). Review of Current Technologies and Proposed Intelligent Methodologies for Water Distributed Network Leakage Detection. *IEEE Access*, *6*, 78846–78867. doi: 10.1109/ACCESS.2018.2885444

Chen, Z., Mauricio, A., Li, W., Gryllias, K., Chen, Z., Mauricio, A., ... Gryllias, K. (2019). Multi-label fault diagnosis based on Convolutional Neural Network and Cyclic Spectral Coherence To cite this version : HAL Id : hal-02188570 Multi-label fault diagnosis based on Convolutional Neural Network and Cyclic Spectral Coherence.

Chollet, F. (2017). *Deep learning with python* (1st ed.). USA: Manning Publications Co.

Costa Cordella, E. (2016). Diagnóstico Para Un Cambio: Los Dilemas De La Regulación De Las Aguas En Chile. *Revista chilena de derecho*, *43*(1), 335–354. doi: 10.4067/s0718-34372016000100014

Dawson, C. W., y Wilby, R. L. (2001). Hydrological modelling using artificial neural networks. *Progress in Physical Geography*, *25*(1), 80–108. doi: 10.1177/030913330102500104

Fan, X., Zhang, X., y Yu, X. . B. (2021). Machine learning model and strategy for fast and accurate detection of leaks in water supply network. *Journal of Infrastructure Preservation and Resilience*, *2*(1). doi: 10.1186/s43065-021-00021-6

Gárate. (2020). Detección y localización de fugas en parte de la red de distribucion de aguas de Santiago, utilizando una maquina de vector de apoyo (SVM). Memoria para optar al Título de Ingeniero Civil. Departamento de Ingeniería Civil, Universidad de Chile.

González. (2017). Diseño, aplicación y evaluación de una metodología de detección activa de pérdidas en redes de agua potable. Memoria para optar al Título de Ingeniero Civil. Departamento de Obras Civiles, Universidad Técnica Federico Santa María.

Gupta, N., y Chandwani, V. (2012, 05). Artificial neural networks as universal function approximators. *International Journal of Emerging Trends in Engineering and Development*, *4*, 456-464.

Hu, X., Han, Y., Yu, B., Geng, Z., y Fan, J. (2021). Novel leakage detection and water loss management of urban water supply network using multiscale neural networks. *Journal of Cleaner Production*, *278*, 123611. doi: 10.1016/j.jclepro.2020.123611

Jeatrakul, P., y Wong, K. W. (2009). Comparing the performance of different neural networks for binary classification problems. *2009 8th International Symposium on Natural*

Language Processing, SNLP '09, 111–115. doi: 10.1109/SNLP.2009.5340935

Jowitt, P. W., y Xu, C. (1990). Optimal Valve Control in Water-Distribution Networks. *Journal of Water Resources Planning and Management*, *116*(4), 455–472. doi: 10.1061/(asce)0733-9496(1990)116:4(455)

Kemba, J., Gideon, K., y Nyirenda, C. N. (2017). Leakage detection in Tsumeb east water distribution network using EPANET and support vector regression. *2017 IST-Africa Week Conference, IST-Africa 2017*(July). doi: 10.23919/ISTAFRICA.2017.8102401

Lau, M. M., y Lim, K. H. (2019). Review of adaptive activation function in deep neural network. *2018 IEEE EMBS Conference on Biomedical Engineering and Sciences, IECBES 2018 - Proceedings*, 686–690. doi: 10.1109/IECBES.2018.08626714

Meier, R. W., y Barkdoll, B. D. (2000). Sampling design for network model calibration using genetic algorithms. *Journal of Water Resources Planning and Management*, *126*(4), 245-250. doi: 10.1061/(ASCE)0733-9496(2000)126:4(245)

Ramchoun, H., Amine, M., Idrissi, J., Ghanou, Y., y Ettaouil, M. (2016). Multilayer Perceptron: Architecture Optimization and Training. *International Journal of Interactive Multimedia and Artificial Intelligence*, *4*(1), 26. doi: 10.9781/ijimai.2016.415

Rojek, I., y Studzinski, J. (2019). Detection and localization of water leaks in water nets supported by an ICT system with artificial intelligence methods as away forward for smart cities. *Sustainability (Switzerland)*, *11*(2). doi: 10.3390/su11020518

Royal Society of Great Britain. (2017). *Machine learning : the power and promise of computers that learn by example* (Vol. 66) (n.º January). Royal Society Open Science.

Sanz, G., y Pérez, R. (2017). Real-Time Monitoring and Operational Control of Drinking-Water Systems. , 437. doi: 10.1007/978-3-319-50751-4

SISS. (2020). Informe Gestión Sector Sanitario, SISS 2020.

Walski, T., Bezts, W., Posluszny, E. T., Weir, M., y Whitman, B. E. (2006). Modeling leakage reduction through pressure control. *Journal \ American Water Works Association*, *98*(4). doi: 10.1002/j.1551-8833.2006.tb07642.x

Xu, T., Chen, S., Guo, S., Huang, X., Li, J., y Zeng, Z. (2019). A small leakage detection approach for oil pipeline using an inner spherical ball. *Process Safety and Environmental Protection*, *124*, 279–289. doi: 10.1016/j.psep.2018.11.009

Zhou, X., Tang, Z., Xu, W., Meng, F., Chu, X., Xin, K., y Fu, G. (2019). Deep learning identifies accurate burst locations in water distribution networks. *Water Research*, *166*, 115058. doi: 10.1016/j.watres.2019.115058

Anexos

A.1. Tablas de desempeño modelos caso B y C

A continuación se presentan las tablas de desempeño completas para los modelos:

Tabla A.1: Desempeño del modelo propuesto para datos de prueba Caso B. Se presentan las métricas consideradas por clase y promedio en la fila final.

Clase	Precision	Recall	F1-Score
0	0,751	0,966	0,845
16405	1,000	0,959	0,979
16661	0,990	0,995	0,993
16782	0,993	0,898	0,943
17055	0,934	1,000	0,966
17438	1,000	0,962	0,981
17450	1,000	0,989	0,995
17571	0,979	0,815	0,890
18153	0,984	0,993	0,989
18284	0,861	0,972	0,913
18424	0,997	0,997	0,997
18632	0,956	0,984	0,970
19131	0,959	0,944	0,951
19170	0,914	0,933	0,923
19215	1,000	1,000	1,000
19446	1,000	0,987	0,994
20047	1,000	0,984	0,992
20049	0,997	1,000	0,998
20623	0,982	0,991	0,986
20736	0,908	0,846	0,876
21146	1,000	1,000	1,000
21191	1,000	0,983	0,991
21289	0,997	0,929	0,962
21449	1,000	0,930	0,964
21530	0,962	1,000	0,981
21655	1,000	1,000	1,000
21824	1,000	1,000	1,000
21975	0,997	1,000	0,999
22289	0,930	0,966	0,948
22599	0,993	0,903	0,946
22912	0,985	1,000	0,993
22934	0,996	0,991	0,994
23048	0,997	0,990	0,994
23053	0,942	0,643	0,764
23130	0,997	1,000	0,998
23148	0,899	0,984	0,940
23149	0,682	0,658	0,670
23171	1,000	1,000	1,000
23386	0,984	0,929	0,956
23459	1,000	1,000	1,000
23473	0,260	0,783	0,390
23600	1,000	0,976	0,988
23948	0,975	1,000	0,987
23988	0,912	0,965	0,938

Tabla A.2: Continuación Tabla A.1

Clase	Precision	Recall	F1-Score
24081	0,967	0,967	0,967
24082	0,967	0,957	0,962
24089	0,996	0,991	0,993
24553	0,997	1,000	0,999
24690	0,912	0,997	0,953
25898	0,956	0,985	0,970
25918	0,990	1,000	0,995
26032	0,993	0,996	0,994
26236	0,948	0,975	0,962
26260	0,974	0,960	0,967
26358	0,906	0,959	0,932
26421	1,000	1,000	1,000
27127	1,000	0,940	0,969
27403	0,968	0,909	0,938
27424	1,000	0,997	0,999
27664	0,983	0,996	0,989
27743	0,986	0,997	0,992
27748	0,940	0,887	0,913
28320	0,997	0,942	0,969
28489	0,996	0,981	0,989
28562	0,900	0,996	0,946
28617	1,000	0,993	0,997
29165	1,000	0,917	0,957
29376	1,000	0,990	0,995
29461	0,977	0,924	0,950
29474	0,983	1,000	0,991
29682	0,979	1,000	0,989
29687	0,795	0,891	0,840
29696	0,977	0,912	0,943
29698	0,988	0,968	0,978
29761	0,923	1,000	0,960
30283	0,919	0,978	0,948
30296	0,990	0,997	0,993
31917	0,861	0,798	0,828
33078	0,983	0,917	0,949
33093	0,997	1,000	0,999
33144	0,803	0,976	0,881
33146	1,000	1,000	1,000
33583	0,909	0,949	0,929
33585	0,952	0,888	0,919
34413	0,997	0,980	0,989
34913	0,714	0,735	0,724
35029	0,986	0,979	0,983
accuracy	0,956	0,956	0,956
macro avg	0,952	0,955	0,951

Tabla A.3: Desempeño del modelo propuesto para datos de prueba Caso C. Se presentan las métricas consideradas por clase, promedio simple y ponderado por numero de datos en la fila final.

Clase	Precision	Recall	F1-Score
0	0,818	0,905	0,859
15860	0,903	1,000	0,949
16715	0,998	0,986	0,992
16972	0,993	0,998	0,995
17004	0,996	1,000	0,998
17144	0,998	0,972	0,985
17145	0,988	1,000	0,994
17282	0,964	0,975	0,970
17299	0,985	0,999	0,992
17351	0,984	0,994	0,989
17820	0,992	0,981	0,986
17987	1,000	1,000	1,000
18201	0,908	0,907	0,908
18335	1,000	0,999	1,000
18700	1,000	0,999	0,999
18821	0,994	0,959	0,976
18827	0,999	0,992	0,995
19714	1,000	0,996	0,998
20128	0,980	0,997	0,988
20178	0,978	0,999	0,989
20261	0,994	1,000	0,997
20292	1,000	0,994	0,997
20324	0,970	0,999	0,984
20434	0,998	1,000	0,999
20500	0,998	0,991	0,994
21024	0,965	0,970	0,968
21297	0,979	0,996	0,987
21303	0,921	0,982	0,951
21451	0,990	0,997	0,994
21489	1,000	1,000	1,000
21561	0,983	0,993	0,988
21730	1,000	1,000	1,000
21814	0,968	0,952	0,960
21896	0,999	0,999	0,999
21942	0,983	0,988	0,986
22108	0,995	0,996	0,996
22212	0,971	0,999	0,985
22413	0,988	0,960	0,974
23175	0,988	0,992	0,990
23266	0,923	0,987	0,954
23284	1,000	0,996	0,998
23326	0,959	0,997	0,978
23540	0,972	0,971	0,972
24064	0,998	0,941	0,969

Tabla A.4: Continuación Tabla A.3

Clase	Precision	Recall	F1-Score
0	0,818	0,905	0,859
15860	0,903	1,000	0,949
16715	0,998	0,986	0,992
16972	0,993	0,998	0,995
17004	0,996	1,000	0,998
17144	0,998	0,972	0,985
17145	0,988	1,000	0,994
17282	0,964	0,975	0,970
17299	0,985	0,999	0,992
17351	0,984	0,994	0,989
17820	0,992	0,981	0,986
17987	1,000	1,000	1,000
18201	0,908	0,907	0,908
18335	1,000	0,999	1,000
18700	1,000	0,999	0,999
18821	0,994	0,959	0,976
18827	0,999	0,992	0,995
19714	1,000	0,996	0,998
20128	0,980	0,997	0,988
20178	0,978	0,999	0,989
20261	0,994	1,000	0,997
20292	1,000	0,994	0,997
20324	0,970	0,999	0,984
20434	0,998	1,000	0,999
20500	0,998	0,991	0,994
21024	0,965	0,970	0,968
21297	0,979	0,996	0,987
21303	0,921	0,982	0,951
21451	0,990	0,997	0,994
21489	1,000	1,000	1,000
21561	0,983	0,993	0,988
21730	1,000	1,000	1,000
21814	0,968	0,952	0,960
21896	0,999	0,999	0,999
21942	0,983	0,988	0,986
22108	0,995	0,996	0,996
22212	0,971	0,999	0,985
22413	0,988	0,960	0,974
23175	0,988	0,992	0,990
23266	0,923	0,987	0,954
23284	1,000	0,996	0,998
23326	0,959	0,997	0,978
23540	0,972	0,971	0,972
24064	0,998	0,941	0,969