Universidad Zaragoza

Modelización e Investigación Matemática, Estadística y Computación

Curso 2020-2021

Trabajo Fin de Máster

Implementación de diferentes técnicas de Deep Generative Models para detección de eventos en Sensores Acústicos Distribuidos

---

Antonio Almudévar Atienza

Director

Alfonso Ortega Giménez

Ponente

Mario Pérez Riera

ABSTRACT

In this work, four solutions to detect events in signals from Distributed Acoustic Sensors (DAS) [1] are presented. These solutions are based on Deep Learning techniques. DAS is a novel technology and it can be used in a wide variety of applications, which motivates this work [2]. Signals from DAS are noisy and applying processing techniques to them becomes necessary. Thus, some alternatives are proposed to solve this problem. Since anomaly detection is a well-studied field, the problem is addressed by considering events as anomalies.

Specifically, Deep Learning techniques have been largely used to solve anomaly detection problems due to their performance and flexibility. The following text presents four alternatives to process signals from DAS sensors based on concepts such as convolutional [4] and recurrent layers [5], Autoencoders [6], Variational Autoencoders [7] or Generative Adversarial Networks [8]. The detection performance of these approaches are compared in terms of area under ROC curve. [9].

The following text is divided into five chapters. The first one is the introduction and presents the problem and sets main goals. In the second one, theoretical preliminaries necessary to develop the work are explained. The third chapter deals with work methodology. In the fourth one, work scenario, experiments and their performance are presented. Finally, the last chapter evaluates the work and suggests future research.

RESUMEN

En este trabajo se presentan cuatro soluciones que permiten detectar eventos en señales procedentes de sistemas de sensado acústico distribuido (DAS) [1] haciendo uso de técnicas de aprendizaje profundo. La idea del DAS es innovadora y tiene variedad de aplicaciones, algo que motiva el desarrollo de trabajos como el presente [2]. Las señales procedentes de estos sensores son ruidosas y requieren un tratamiento para detectar eventos. Por lo tanto, se proponen varias alternativas para llevar a cabo este tratamiento y es la continuación del trabajo desarrollado en [3]. La idea principal es abordar el problema desde una perspectiva de detección de anomalías, el cual es un problema ampliamente estudiado y para el que existen multitud de soluciones según el escenario.

En concreto, en los últimos años, se han utilizado técnicas de aprendizaje profundo para resolver el problema de la detección de anomalías debido al buen rendimiento y flexibilidad que ofrecen. En este texto se dan cuatro propuestas basadas en conceptos como capas convolucionales [4] y recurrentes [5], Autoencoders [6], Variational Autoencoders [7] o Generative Adversarial Networks [8] para procesar las señales procedentes de sensores DAS. El rendimiento de estas propuestas se compara en términos de área bajo la curva ROC [9].

El texto se divide en cinco capítulos. El primero es una introducción y sirve para presentar el problema y establecer objetivos. En el segundo se presentan los conceptos teóricos que permiten desarrollar el trabajo. En el tercero se explica la metodología de trabajo y las cuatro propuestas desarrolladas. En el cuarto se presentan el escenario de trabajo, las pruebas realizadas y los resultados obtenidos. Por último, el quinto sirve para evaluar todo el trabajo desarrollado y proponer líneas futuras de investigación.

AGRADECIMIENTOS

Me gustaría empezar dando las gracias a mis padres por apoyarme, confiar en mí y acompañarme en este camino. También quiero dar las gracias a todos mis familiares y amigos con los que siempre puedo contar y que me ayudan a seguir adelante.

Por otro lado, me gustaría agradecer a Alfonso su ayuda, sus consejos y el tiempo dedicado para que esta trabajo salga adelante. Igualmente, quiero reconocer a Mario su labor como enlace con la docencia del máster.

Por último, quiero dar las gracias a Aragon Photonics por confiar en mí y apostar por la investigación y el desarrollo.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1. INTRODUCCIÓN

En este capítulo se plantea el problema, la situación antes de comenzar este proyecto, así como algunas aplicaciones que tienen los sistemas de sensado acústico distribuido.

1.1. Motivación del proyecto

En los últimos años se han presentado técnicas que emplean fenómenos físicos que se dan en la propagación de ondas electromagnéticas, en particular de la luz, para detectar eventos de tipo mecánico en el entorno físico de una fibra óptica. Estas técnicas se basan principalmente en el impacto que un evento de tipo mecánico tiene en el fenómeno de la dispersión Rayleigh, algo que se explicará un poco más en detalle en el siguiente capítulo. Los sistemas que desarrollan estas técnicas se conocen como DAS (distributed acoustic sensor) [1], ya que emplean la fibra óptica como sensor distribuido para captar eventos en sus proximidades haciendo uso de transmisores y receptores ópticos.

El porqué de la relevancia de un sistema capaz de detectar eventos en el entorno de la fibra óptica es la multitud de aplicaciones que puede tener, desde seguridad [10] hasta control de tráfico. Desde el punto de vista de la seguridad, este sistema podría suponer un método para detectar un vehículo circulando por una zona prohibida o a una persona picando en un terreno bajo el cual hay una fibra óptica. De manera similar, no es difícil darse cuenta de que, teniendo una medida que contiene información sobre la evolución de un cuerpo en tiempo y distancia, se puede calcular la velocidad del mismo. También se puede detectar una acumulación de vehículos en una zona concreta, lo que puede ser un indicativo de que se ha producido un accidente. También, esta tecnología se ha utilizado para detectar actividad sísmica [11, 12].

Y todo esto se puede conseguir con un sistema invisible a los ojos del usuario, abarcando grandes extensiones de terreno de hasta 50 km y de manera independiente de la orografía. Otra ventaja fundamental es el aprovechamiento de lo que se conoce como fibra oscura, esto es, fibra óptica que ha sido desplegada y a la que no se la ha dado ningún uso. La razón de ser de esta fibra oscura es precisamente facilitar la implementación de técnicas que en un principio no habían sido planteadas sin suponer esto un gran coste económico ni la necesidad de solicitar gran cantidad de permisos burocráticos [13].

1.2. Introducción a la problemática

Este trabajo, en cambio, no tiene como objetivo estudiar y desarrollar sistemas basados en estos fenómenos, sino trabajar con las señales capturadas por receptores que

funcionan gracias a esta tecnología y tratar de encontrar en ellas los cambios que un evento produce en el comportamiento habitual de la señal óptica propagada dentro de la fibra. Este proyecto está en una fase cuya meta es detectar si hay un evento o no, es decir, se trata de una tarea de detección de anomalías y no de su clasificación.

La señal bidimensional (tiempo y distancia) procedente de los receptores se muestra ruidosa y sin ningún tratamiento resulta difícil de interpretar. En el trabajo previo a este [3] se propone una solución basada en los conceptos más elementales del aprendizaje profundo con resultados satisfactorios. En él se propone una red neuronal que es capaz por ella misma, sin la ayuda de un humano, de detectar un evento anómalo en las inmediaciones de la fibra a partir de la señal proporcionada por los receptores acústicos.

Sin embargo, el trabajo previo es sólo un primer paso para estudiar en profundidad esta problemática y aquí se avanza un poco más proponiendo tres nuevos sistemas capaces de mejorar el rendimiento. Estos sistemas están enmarcados también en el ámbito del aprendizaje profundo, pero añaden complejidad y nuevos conceptos que permiten mejorar el tratamiento de las señales.

1.3. Objetivos

Cuando se va a realizar un trabajo como este, es fundamental establecer unos objetivos claros para poder valorar de una manera mensurable la consecución de ellos. En este caso, establecer unos objetivos no es complicado, ya que el trabajo tiene un fin claro, pero sí que es cierto que se pueden establecer varios puntos que miden el avance del trabajo realizado. Los objetivos principales podrían resumirse en los siguientes puntos:

- **Adquirir conocimientos sobre aprendizaje profundo y modelos generativos.** Si bien el trabajo previo utiliza técnicas de aprendizaje profundo, para seguir avanzando en el proyecto, se requiere subir un peldaño conceptual y para ello es necesario estudiar y comprender nuevos conceptos.
- **Implementar y analizar la existencia de mejoras al introducir capas LSTM.** Debido al hecho de que una de las dimensiones de las capturas del DAS es el tiempo, analizar la evolución de las señales a lo largo del mismo puede proporcionar mejoras en el rendimiento.
- **Implementar y analizar la utilización de Variational Autoencoders (VAE) como solución al problema.** Introducir modelos estadísticos puede mejorar el rendimiento del sistema en algunos aspectos.
- **Implementar y analizar las prestaciones ofrecidas por las Generative Adversarial Networks (GAN) como solución al problema.** En estos últimos años, ha incrementado la popularidad de estos modelos, por lo que han surgido aproximaciones basadas en ellos que pueden ayudar a la resolución de este problema.

- **Comparar los distintos sistemas mediante una medida de bondad.** Dado que se proponen distintos sistemas y se tienen varios parámetros de diseño, conviene comparar formalmente cómo éstos afectan a los sistemas, así como los sistemas entre sí.

2. MARCO TEÓRICO

En este capítulo se desarrollan los conceptos teóricos necesarios para comprender la resolución del problema. Además, la nomenclatura introducida es la misma que se utiliza en el siguiente capítulo para describir los sistemas.

2.1. Sensores acústicos distribuidos

Aunque este no es un trabajo centrado en el campo de la óptica y su objetivo es procesar y tratar de sacar el máximo partido de información proporcionada por sistemas que aprovechan características de la luz dentro de una fibra, a continuación se propone un repaso de los fenómenos físicos que permiten desarrollar sistemas como este.

La dispersión Rayleigh es la dispersión de una onda electromagnética por partículas de tamaño mucho menor a la longitud de onda de la radiación [14]. Aunque este es un fenómeno no deseable en la transmisión de información a través de una fibra óptica, puede tener aplicaciones, como es el caso [15].

Por otro lado, la reflectometría óptica coherente en dominio temporal (C-OTDR) es una técnica de medición utilizada para determinar algunas características de un medio de transmisión, como la fibra óptica, mediante la observación de las ondas reflejadas [16, 17]. Esto lo lleva a cabo utilizando la dispersión Rayleigh, que permite detectar señales de frecuencias acústicas a través de grandes distancias.

Se han presentado en los últimos años sistemas basados en C-OTDR que utilizan la fibra óptica como medio sensor para detectar vibraciones de forma distribuida (DAS) en el entorno físico próximo a la fibra. Éstos tienen la capacidad de detectar eventos con una muy buena precisión.

En cuanto a la implementación de estos sistemas, habitualmente se emplea un láser coherente, cuya señal es amplificada e inyectada en una fibra óptica. La potencia óptica retroesparcida por la fibra sigue un patrón de interferencias por la suma coherente de todos los frentes de onda generados por difusión Rayleigh. Si la retrodifusión se mantiene constante en fase, la situación de interferencia producida por la suma coherente de la señal retroesparcida permanece constante. Sin embargo, la modificación de uno o varios elementos por una perturbación de tipo mecánica se traduce en un cambio de fase de la señal retroesparcida en ese punto concreto, lo que da lugar a una variación en la situación de interferencia. Debido a este fenómeno la fibra se convierte en un sensor distribuido que permite detectar perturbaciones mecánicas en una posición concreta de la fibra.

Habitualmente, en cuanto al rango de funcionamiento de estos sistemas, se habla de unos 40-50 km, aunque esto depende evidentemente de la configuración óptica y de cuál

es el evento que produce la perturbación mecánica. El porqué de esta limitación en distancia es simplemente que, por ejemplo, para una fibra monomodo y una longitud de onda de operación de 1550 nm se tiene una atenuación típica de 0.2 dB/km, por lo que para una longitud de fibra de 50 km en la que la luz ha de realizar un recorrido de ida y vuelta se tiene una atenuación de la señal de 20 dB y esto se traduce en un empeoramiento de la relación señal a ruido.

Aquí es donde entra el sistema desarrollado en este trabajo, que supone una novedad en cuanto a la forma de tratar de encontrar estas perturbaciones mecánicas, y está basado en conceptos propios del campo de la detección anomalías, es decir, se ha considerado que estas perturbaciones mecánicas son eventos anómalos que se pretende detectar.

2.2. Redes neuronales

Los modelos de regresión lineal y logística y sirven para modelar o clasificar variables multidimensionales de manera más o menos acertadas según la procedencia de dichas variables. El problema es que la realidad es, en su mayoría, no lineal, por lo que emplear los modelos mencionados no es la mejor forma de ajustarla. Por este motivo, surgen alternativas cuyo objetivo es el mismo, pero permiten introducir términos no lineales. Una de las opciones más populares para dicha tarea son las redes neuronales o artificial neural networks (ANN). Las tareas que estas pueden realizar se clasifican en dos grandes grupos:

- **Clasificación:** Convierte la entrada X a una salida discreta o categórica Y.
- **Regresión:** Convierte la entrada X a una salida continua Y.

Las redes neuronales, al igual que otros modelos de regresión o clasificación, tienen parámetros que han de ser establecidos según el valor de un conjunto de datos de entrada. El proceso de determinar el valor de dichos parámetros se conoce como **fase de entrenamiento** y el conjunto de datos que se utiliza para ello se conoce como datos de entrenamiento. Por otro lado, una vez se ha llevado a cabo el entrenamiento, las redes neuronales están preparadas para modelar datos similares a aquellos con los que ha sido entrenada. El proceso de modelar estos datos se conoce como **fase de inferencia**.

Aunque el surgimiento de la multitud de aplicaciones para las se utilizan redes neuronales es algo ocurrido a lo largo de las últimas dos décadas, la realidad es que muchos de los algoritmos que se emplean hoy en día vieron la luz mucho antes, siendo en 1943 cuando surge el primer concepto que se puede relacionar con las actuales ANN [19].

2.2.1. La neurona

Una red neuronal esta formada por un conjunto de neuronas. Por lo tanto, cabe comenzar la explicación definiendo qué se entiende por neurona en el sentido matemático.

Aunque, como se explicará más adelante, hay distintos tipos de neuronas y formas de interconectarlas, una neurona, esencialmente es una función $T : \mathbb{R}^n \to \mathbb{R}$ definida por:

$$T(\boldsymbol{x}) = f\left(b + \sum_{i=1}^{n} w_i \cdot x_i\right) = f(b + \boldsymbol{w}^T \cdot \boldsymbol{x}) \tag{2.1}$$

donde $x_i \in \mathbb{R}$ son las entradas a la neurona, $w_i \in \mathbb{R}$ son los pesos de la neurona, $b \in \mathbb{R}$ es el sesgo de la red y $f : \mathbb{R} \to \mathbb{R}$ es la función de activación.

2.2.2. La función de activación

Como se ha mencionado ya en la sección anterior, las funciones de activación son funciones que van de $\mathbb{R}$ en $\mathbb{R}$ y en ellas reside la esencia de las redes neuronales, ya que introduce el comportamiento no lineal a la neurona, lo cual permite modelar con acierto la realidad.

Sigmoide

Esta función de activación convierte cualquier valor de $\mathbb{R}$ al intervalo $(0, 1)$ y se define de la siguiente forma:

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

Es especialmente utilizada como función de activación en la última capa de una red cuando se realiza una tarea de clasificación.

Tangente hiperbólica

Otra función similar a la anterior que también es utilizada como activación es la tangente hiperbólica [20] y lleva cualquier valor de $\mathbb{R}$ al rango $(-1, 1)$. Se define como:

$$\tanh x = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{2.3}$$

Se desaconseja utilizar la sigmoide y tangente hiperbólica como funciones de activación en capas intermedias, ya que pueden originar el problema del desvanecimiento de gradiente (vanishing gradient). Este problema sucede cuando la derivada de la función de activación es casi nula para un conjunto muy grande de valores y supone un problema porque esto puede ralentizar y empeorar el entrenamiento de la red. En el algoritmo de entrenamiento, como se explica en 2.2.5, se utiliza el gradiente de la función de coste la cual depende de las funciones de activación y hay que tratar de que el menor número de elementos de ésta se anulen.

ReLU

Para evitar el problema del gradiente descendente, se propone utilizar la funcón ReLU (Rectified Linear Unit)[21], que se define como:

$$ReLU(x) = \text{máx}(0, x) \tag{2.4}$$

Esta función de activación es muy utilizada en capas intermedias de la red, entre otras cosas, por ser ligera desde un punto de vista computacional. El problema que tiene es que para entradas negativas, su salida es cero y sucede a menudo que algunas neuronas tienen un bias negativo grande y la entrada a la función de activación es negativa siempre, por lo que se dice que la neurona muere.

LeakyReLU

Para evitar el problema de las neuronas muertas, se proponen alternativas a ReLU [22]. Entre ellas está la función LeakyReLU, que se define como:

$$LeakyReLU(x) = \begin{cases} a \cdot x & \text{si} \quad x < 0 \\ x & \text{si} \quad x \geq 0 \end{cases} \tag{2.5}$$

donde, a menudo, $a = 0{,}01$.

2.2.3. Fase de inferencia

La idea de la neurona previamente dada es interesante, pero es simplemente una composición de dos funciones, una lineal y otra que no lo es y que toma como entrada un vector. El poder de estos sistemas viene de la combinación de un número bastante alto de estas unidades. Para comprender esto mejor, es común utilizar un diagrama como el que aparece en la figura 2.1. Cada uno de los conjuntos de neuronas que tienen como entrada un mismo vector se llama capa. La expresión matemática que expresa la salida de una capa con respecto a su entrada es análoga a la de una única neurona. Así, dada una red neuronal con L capas, la salida de la capa $l \in \{1, 2, ..., L\}$ de dicha red es $\boldsymbol{o}^l$, de forma que:

$$\boldsymbol{o}^l = T(\boldsymbol{x}^l) = F_l\left(\boldsymbol{b}^l + W^l \cdot \boldsymbol{x}^l\right) \tag{2.6}$$

donde:

- $\boldsymbol{x}^l = (x_i^l) \in \mathbb{R}^{n_l}$ donde x_i^l es la entrada i de la capa l
- $W^l = (w_{ij}^l) \in \mathbb{R}^{n_l \times m_l}$ donde w_{ij}^l es el peso i de la neurona j de la capa l
- $\boldsymbol{b}^l = (b_j^l) \in \mathbb{R}^{m_l}$ donde b_j^l el bias de la neurona j de la capa l
- $F_l : \mathbb{R}^{m_l} \to \mathbb{R}^{m_l}$ es la función de activación de la capa l

- $\boldsymbol{o}^l = (o_j^l) \in \mathbb{R}^{m_l}$ donde o_j^l es la salida de la neurona j de la capa l
- n_l y m_l son el número de entradas y el número de neuronas de la capa l, respectivamente.

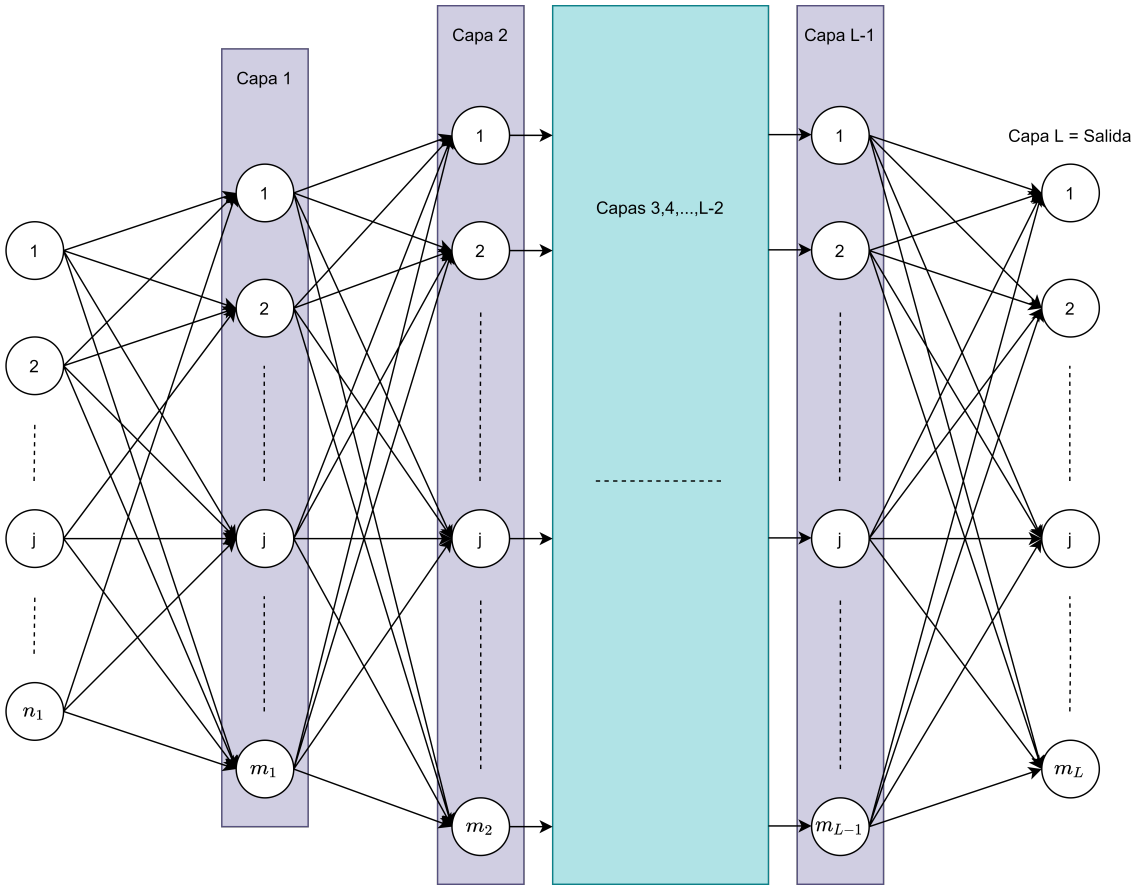


Fig. 2.1. Diagrama de una red neuronal con L capas

Aquí se deben comentar varias cosas. La primera es que la salida de la capa l es la entrada de la capa $l+1$, es decir, $y_l = x_{l+1}$. Por lo tanto, $n_{l+1} = m_l$. Así, m_l es uno de los parámetros de diseño de una red neuronal. Por otro lado, la entrada a una red $\boldsymbol{x}$ se corresponde con la entrada a la primera capa y, por tanto, $\boldsymbol{x}^1 = \boldsymbol{x}$. Equivalentemente, la salida de la red $\boldsymbol{y}$ se corresponde con la salida de la última capa, es decir, $\boldsymbol{y} = \boldsymbol{o}^L$. En concreto, para tareas de clasificación binaria, la salida debe ser un escalar, esto es, $m_L = 1$.

Con respecto a F_l, se ha comentado que es la función de activación de la capa l, pero, tal y como se ha explicado en 2.2.2, éstas van de $\mathbb{R}$ en $\mathbb{R}$ y se definen para una neurona y no para una capa. Sin embargo, habitualmente se aplica la misma función de activación para todas las neuronas de una misma capa y se emplea la generalización $F_l(a_1, a_2, ..., a_n) = (f_l(a_1), f_l(a_2), ..., f_l(a_n))$ donde $f_l : \mathbb{R} \to \mathbb{R}$ sí se corresponde con la descripción dada en la sección 2.2.2. Así, a F_l se le conoce como función de activación de la capa l.

En este punto, aún no se ha comentado nada acerca de w_{ij}^l y b_j^l, los pesos y bias de la red, respectivamente. Éstos reciben el nombre de parámetros de la red y se ha de establecer

su valor. A diferencia de un modelo de regresión lineal, no hay una fórmula cerrada que permita calcular los parámetros del modelo óptimos para un conjunto de datos debido a la gran complejidad de estos modelos. Sin embargo, hay métodos que permiten buscar valores adecuados de w_{ij}^l y b_j^l para la casuística concreta de un problema. Este proceso se conoce como fase de entrenamiento.

2.2.4. Funciones de coste

Las redes neuronales son modelos de regresión o clasificación y, al igual que sucede con los modelos de regresión lineal y logística, es necesario definir métricas que reflejen la bondad de ajuste del modelo. Para ello, se definen las funciones de coste, que, de alguna forma, relacionan la salida de la red (y) y la salida objetivo (y^t). A continuación, se describen algunas de de las funciones de coste más populares. Éstas suelen representar de forma acertada el funcionamiento de la red, pero se pueden definir funciones de coste ad hoc según el problema y sus objetivos como, de hecho, se realiza en varios de las alternativas presentadas en este trabajo.

Error cuadrático medio

Esta función de coste es empleada en tareas de regresión. Para el caso en que se trabaja con imágenes, se define el MSE para dos matrices $Y = (y_{ij})$ e $Y^t = (y_{ij}^t)$ de tamaño $N \times M$, de la siguiente forma:

$$MSE(Y, Y^t) = \frac{1}{MN} \sum_{i=1}^{M} \sum_{j=1}^{N} (y_{ij} - y_{ij}^t)^2 \tag{2.7}$$

Entropía de Shannon y entropía cruzada

Dada una variable aleatoria X que puede tomar los valores $x_1, x_2, ..., x_n$ con probabilidades $p(x_1), p(x_2), ..., p(x_n)$, respectivamente, la entropía de Shannon [23] de X se define como:

$$H(p) = -\sum_{i=1}^{n} p(x_i) \log p(x_i) \tag{2.8}$$

Esta medida refleja el nivel de información, sorpresa o incertidumbre inherente a los posibles valores que puede tomar una variable aleatoria. Cuando el logaritmo se toma con base 2, indica el mínimo número medio de bits necesarios para codificar los posibles sucesos $x_1, x_2, ..., x_n$.

La entropía cruzada se utiliza para comparar dos distribuciones de probabilidad p y q y parte de la definición de entropía de Shannon. Así, siendo $p(x_i)$ y $q(x_i)$ las probabilidades de que suceda el suceso x_i en las distribuciones p y q, respectivamente, la entropía cruzada

entre p y q se define como

$$H(p, q) = -\sum_{i=1}^{n} p(x_i) \log q(x_i) \tag{2.9}$$

Si el logaritmo tiene base 2, entonces se puede interpretar como el número medio de bits necesarios para codificar datos procedentes de p modelándolos a través de q.

En el caso de clasificación, binaria este concepto es muy utilizado como función de coste. En esta situación, la salida objetivo es $y^t \in \{0, 1\}$. Además, como se explica en 2.2.2, la última capa de la red tiene como activación la función sigmoide o alguna similar, por lo que $\boldsymbol{y} \in (0, 1)$. Así, si $n = 2$ y se establece $p(x_0) = y^t$, $p(x_1) = 1 - y^t$, $q(x_0) = y$, $q(x_1) = 1 - \boldsymbol{y}$, queda que:

$$H(y^t, y) = -y^t \log(y) - (1 - y^t) \log(1 - y) \tag{2.10}$$

Divergencia Kullback–Leibler

La entropía cruzada permite comparar dos distribuciones de probabilidad. En concreto, si se considera p fija, $H(p, q)$ aumentará conforme q discrepe de p. Sin embargo, tiene una fuerte dependencia con el valor de $H(p)$. De esta forma, habrá casos en los que p y q sean similares, pero $H(p)$ sea elevado y, por tanto $H(p, q)$ también lo sea. Esto significa que $H(p, q)$ no es una buena medida en términos absolutos para analizar como de buena es la aproximación de p vía q. Aquí surge un nuevo concepto llamado divergencia Kullback–Leibler o divergencia KL (D_{KL}) [24] y que se define a continuación:

$$D_{KL}(p||q) = H(p, q) - H(p) = \sum_{i=1}^{n} p(x_i) \log \frac{p(x_i)}{q(x_i)} \tag{2.11}$$

Por lo tanto, $D_{KL}(p||q)$ se puede interpretar como el número medio de bits adicionales requeridos para codificar datos procedentes de $p(x)$ utilizando la distribución $q(x)$ en lugar de utilizar $p(x)$.

El concepto de divergencia KL se puede extender a dos distribuciones continuas p y q de la forma:

$$D_{KL}(p||q) = \int_{-\infty}^{\infty} p(x) \log \frac{p(x)}{q(x)} dx \tag{2.12}$$

donde p y q son funciones de densidad de probabilidad. Aunque aún no se han presentado, hay modelos basados en redes neuronales que tratan de estimar parámetros de una distribución de probabilidad, en cuyo caso, esta medida es especialmente interesante para ser utilizada como función de coste. Algunas ideas importantes de la divergencia KL son:

- $D_{KL}(p||q) \geq 0$
- $D_{KL}(p||q) = 0 \leftrightarrow p = q$
- En general, $D_{KL}(p||q) \neq D_{KL}(q||p)$, por lo que no es una distancia

2.2.5. Fase de entrenamiento - Métodos de gradiente descendente

Estas medidas del error dependen de la entrada a la red X y de los parámetros w_{ij}^k y b_j^k, ya que la salida de la red también depende de éstos, así como de la salida objetivo Y^t. La fase de entrenamiento de una red consiste entonces en tratar de encontrar los valores de cada w_{ij}^k y b_j^k que minimizan la función de coste para las entradas X y salidas objetivo Y_t dadas durante esta fase. Así pues, éstas se pueden ver como campos escalares cuyas variables son los pesos w_{ij}^k y los sesgos b_j^k de la red. Debido a ser funciones con tantas variables y una expresión analítica difícilmente tratable, la búsqueda del mínimo global de forma analítica es inabordable, en particular cuando el tamaño de la red aumenta. Sin embargo, existen algoritmos iterativos que permiten acercarse al mínimo global de una función con un número muy elevado de variables. Así pues, entrenar una red neuronal consiste en ejecutar estos algoritmos para encontrar un valor cercano al mínimo global de la función de coste. El conjunto de algoritmos que tienen este objetivo reciben el nombre de algoritmos de gradiente descendente.

Dado un campo escalar $f : \mathbb{R}^n \to \mathbb{R}$ y un punto $\boldsymbol{\theta}$ de su dominio, se define $\nabla f(\boldsymbol{\theta}) : \mathbb{R}^n \to \mathbb{R}^n$ como el gradiente de f en θ e indica la dirección en la cual campo de f aumenta más rápidamente en el entorno de $\boldsymbol{\theta}$. Equivalentemente, f decrece más rápidamente en el entorno de $\boldsymbol{\theta}$ en la dirección $-\nabla f(\boldsymbol{\theta})$. De esta forma, si:

$$\boldsymbol{\theta}_{n+1} = \boldsymbol{\theta}_n - \alpha \cdot \nabla f(\boldsymbol{\theta}_n) \tag{2.13}$$

para $\alpha > 0$, se cumple que $f(\boldsymbol{\theta}_{n+1}) \leq f(\boldsymbol{\theta}_n)$. Por lo tanto, si se parte de un punto cualquiera $\boldsymbol{\theta}_0$, cabe esperar que repitiendo el proceso de la expresión 2.13 un número de veces, la secuencia $\boldsymbol{\theta}_0, \boldsymbol{\theta}_1, \boldsymbol{\theta}_2 ...$ se acerque a un mínimo local de la función f. El factor α recibe el nombre de learning rate (tasa de aprendizaje) ya que regula cuánto se modifica el parámetro $\boldsymbol{\theta}$ entre iteraciones consecutivas. Así, los algoritmos de gradiente descendente se basan en actualizar el parámetro $\boldsymbol{\theta}$ según ideas similares a 2.13.

El problema de tratar este asunto desde un punto de vista iterativo en lugar de analítico es que el objetivo ideal es encontrar el mínimo global de la función de coste, pero es algo común que una vez se encuentra un mínimo local, el algoritmo quede atrapado en él y no encuentre otro mínimo menor. Esto se entiende bien en la figura 2.2 en la que se muestra un caso trivial de una función de coste que depende de una única variable. El learning rate es fundamental para la convergencia del algoritmo, ya que si es muy elevado el algoritmo no covergerá y dará pasos demasiado grandes, mientras que si es muy pequeño quedará atrapado en mínimos locales.

En 1847 ya se propuso una primera solución que aún a día de hoy se sigue utilizando salvo escasas modificaciones [25]. Uno de los algoritmos más utilizados para dicha tarea es Adam, que es el que se ha utilizado para entrenar todas las redes del presente trabajo. En 2.14 se dan únicamente las ecuaciones de actualización de los pesos, ya que es suficiente para dar una idea básica de cuál es el objetivo del algoritmo y su funcionamiento. No obstante, en [26] se puede encontrar el desarrollo del algoritmo más en detalle.

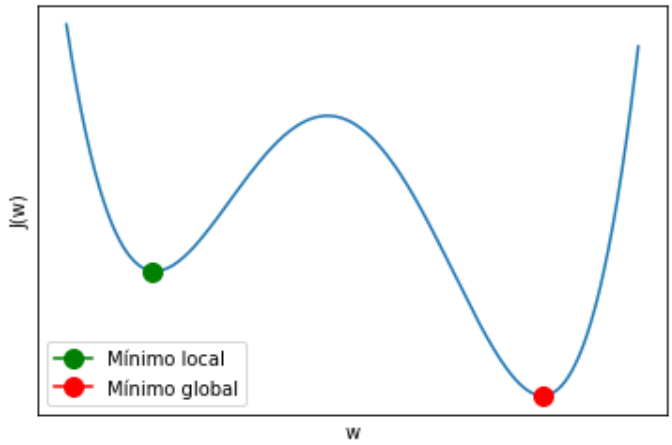


Fig. 2.2. Ejemplo de función de coste de una variable con un mínimo local y un mínimo global

Para explicar este algoritmo, se consideran dos entradas x_n e y_n y un objeto $\boldsymbol{\theta}$. Este proceso aplicado al entrenamiento de redes neuronales trata, por tanto, de buscar mínimos de una función de coste multivariable $J(x_n, y_n, \boldsymbol{\theta}_{n-1})$ donde x_n e y_n serán la entrada y salida de la red, respectivamente y $\boldsymbol{\theta}_{n-1}$ contendrá los parámetros de la red w_{ij}^k y b_j^k en la iteración $n-1$ para $n \geq 1$. Los parámetros de la red se actualizarán en cada iteración. El el caso de las redes neuronales, los valores en $\boldsymbol{\theta}_0$ son inicializados de forma aleatoria y con valores cercanos a cero. El algoritmo, además de x_n e y_n, toma cuatro parámetros de entrada, que son β_1, β_2, α (learning rate) y ϵ.

$$
\begin{aligned}
m_0 &= v_0 = 0 \\
\nabla_n &= \frac{\partial J(x_n, y_n, \boldsymbol{\theta}_{n-1})}{\partial \boldsymbol{\theta}_{n-1}} \\
m_n &= \beta_1 m_{n-1} + (1-\beta_1)\nabla_n = (1-\beta_1)\sum_{i=1}^{n} \beta_1^{n-i} \nabla_i \\
v_n &= \beta_2 m_{n-1} + (1-\beta_2)\nabla_n = (1-\beta_2)\sum_{i=1}^{n} \beta_2^{n-i} \nabla_i \\
\hat{m}_n &= \frac{m_n}{1-\beta_1^n} \\
\hat{v}_n &= \frac{v_n}{1-\beta_2^n} \\
\theta_n &= \boldsymbol{\theta}_{n-1} - \alpha \frac{\hat{m}_n}{\sqrt{\hat{v}_n} + \epsilon}
\end{aligned}
\tag{2.14}
$$

Dados los conjuntos de datos para entrenamiento X e Y, x_n e y_n se escogen seleccionando lotes de tamaño habitualmente fijo de X e Y, respectivamente. Cada uno de estos lotes recibe el nombre de batch y el tamaño de batch hace referencia a la longitud de dichos lotes.

Además, es común que los datos X e Y, se utilicen varias veces para el proceso del entrenamiento debido, principalmente, a que emplearlos una vez suele ser insuficiente para la convergencia del algoritmo. La i-ésima vez que se utilizan los conjuntos X e Y

recibe el nombre de epoch *i* y el número de veces que se usan recibe el nombre de número de epochs.

2.2.6. Normalización de las entradas

El valor del gradiente de una función depende evidentemente de la magnitud de las entradas de dicha función. En este caso, $\nabla_n = \frac{\partial J(x_n, y_n, \theta_{n-1})}{\partial \theta_{n-1}}$ depende de x_n, por lo que su magnitud es fundamental para una convergencia eficiente del algoritmo [27]. Dado que normalizar es un concepto poco concreto, aparecen aquí dos términos, que son la estandarización y escalado [28].

Primero, en cuanto a la estandarización, consiste en manipular los datos de forma que sigan una distribución de probabilidad concreta, habitualmente con media cero y varianza uno. Este concepto no se ha empleado en este trabajo, ya que podría afectar a las estadísticas de los datos anómalos.

Por otro lado, el escalado consiste en convertir todos los datos a un rango de valores como, por ejemplo, de cero a uno. Este procedimiento sí se ha empleado en este trabajo, en concreto el escalado MinMax, que transforma el conjunto de datos X a Y según la expresión:

$$Y = \frac{X - \text{mín}(X)}{\text{máx}(X) - \text{mín}(X)} \tag{2.15}$$

2.3. Tipos de capas

Como ya se comentaba en la sección 2.2.1, no hay una única definición de neuronas ni una única forma de conectarlas. La dada hasta el momento es la más elemental y sencilla de comprender y que sirve para explicar conceptos tales como las fases de entrenamiento e inferencia, o las funciones de coste. Sin embargo, en multitud de aplicaciones, se requieren un tipo de prestaciones que la definición de neurona vista hasta ahora no es capaz de ofrecer. Por este motivo, a continuación se explican algunas de las capas más utilizadas y que han servido para desarrollar este trabajo.

2.3.1. Capas fully-connected

En las secciones 2.2.1 y 2.2.3 se ha dado una definición de neurona y se ha explicado una forma de conectarlas entre sí. A esta forma de conectar este tipo de neuronas se le conoce como fully-connected debido a que todas las salidas de una capa sirven de entrada para la siguiente.

2.3.2. Capas convolucionales

Las redes neuronales están inspiradas en los mecanismos que los humanos tienen para adquirir nuevos conocimientos, y uno de ellos es la vista. Por esta razón, se emplean redes neuronales que aprovechan las características de las imágenes o, alternativamente, señales de dos dimensiones, que es precisamente el caso de este proyecto [29, 4].

Para empezar, a continuación se define la correlación cruzada ($*$) entre dos señales bidimensionales $X[m,n],\ m \in \{-p, -p+1, ..., 0, ..., p-1, p\},\ n \in \{-q, -q+1, ..., 0, ..., q-1, q\}$ y $h[i,j]$ $i \in \{-u, -u+1, ..., 0, ..., u-1, u\},\ j \in \{-v, -v+1, ..., 0, ..., v-1, v\}$, ya que es la operación que se lleva a cabo en las capas convolucionales.

$$(h * X)[k,l] = \sum_{i=-u}^{u} \sum_{j=-v}^{v} h[i,j] \cdot X[k+i, l+j] \tag{2.16}$$

Una capa convolucional l toma como entrada un conjunto de matrices $\boldsymbol{X}^l = (X^l_m)$ y obtiene a la salida otro conjunto de matrices $\boldsymbol{O}^l = (O^l_k)$. Para ello, se realiza lo siguiente

$$\boldsymbol{O}^l = \boldsymbol{F}^l\left(\boldsymbol{b}^l + \boldsymbol{W}^l \star \boldsymbol{X}^l\right) \rightarrow O^l_k = F^l_k\left(b^l_k + \sum_{m=1}^{L_{l-1}} W^l_{km} * X^l_m\right),\ k = 1, ..., L_l \tag{2.17}$$

donde:

- $\boldsymbol{X}^l = (X^l_m) = (x^l_{mij}) \in \mathbb{R}^{L_{l-1} \times H_{l-1} \times W_{l-1}}$ donde X^l_m es la entrada m a la capa l
- $\boldsymbol{W^l} = (W^l_{km}) = (w^l_{kmij}) \in \mathbb{R}^{L_l \times L_{l-1} \times f_l \times f_l}$ donde W^l_{km} es el filtro k para la entrada m de la capa l
- $\boldsymbol{b}^l = (b^l_k) \in \mathbb{R}^{L_l}$ donde b^l_k es el bias k de la capa l
- $\boldsymbol{F}^l = (F^l) : \mathbb{R}^{L_l \times H_l \times W_l} \rightarrow \mathbb{R}^{H_l \times W_l}$ donde F^l es la función de activación de la capa l.
- $\boldsymbol{O}^l = (O^l_m) = (o^l_{mij}) \in \mathbb{R}^{L_l \times H_l \times W_l}$ donde O^l_m es la salida m de la capa l
- H_l y W_l son el número de filas y de columnas de las matrices de entrada de la capa l
- L_l es el número de filtros de la capa l, recibe el nombre de número de canales de la capa L y es un parámetro de diseño
- f_l es el tamaño de filtro de la capa l y es un parámetro de diseño

En las capas convolucionales es común utilizar un factor de diezmado, así como añadir ceros alrededor de la entrada antes de realizar la correlación cruzada para evitar el efecto borde, lo cual se conoce como padding. Con esto, si se sustituye la expresión 2.16 en 2.17,

se llega a lo siguiente:

$$
\begin{aligned}
o^l_{kij} = f_l\left(b^l_k + \sum_{m=1}^{L_{l-1}}\sum_{n_1=1}^{f_l}\sum_{n_2=1}^{f_l} w^l_{k,m,n_1,n_2}\cdot x^l_{m,D^l(i-1)+n_1,D^l(j-1)+n_2}\right),\\
k = 1,...,L_l;\ i = 1,...,H_l;\ j = 1,...,W_l,\\
W_l = \left\lfloor \frac{W_{l-1}+2P^l-f_l}{D^l}\right\rfloor + 1,\ H_l = \left\lfloor \frac{H_{l-1}+2P^l-f_l}{D^l}\right\rfloor + 1
\end{aligned}
\tag{2.18}
$$

donde D^l y P^l son el factor de diezmado y el tamaño de padding de la capa l, respectivamente y son también parámetros de diseño.

2.3.3. Capas recurrentes y LSTM

En ciertas ocasiones, puede resultar de utilidad tener en cuenta la relación, si es que existe, entre entradas consecutivas. El caso más obvio es el de un vídeo en el que las entradas a las redes pueden ser los fotogramas, que no son independientes entre sí, e interesa analizar precisamente esa dependencia entre ellos. Es por esto que surge un concepto llamado Recurrent Neural Network (RNN) [5], que hace referencia a un tipo de capa que sufre modificaciones según las últimas entradas. Se dice que son capas con memoria.

Existen diferentes formas de modelar las RNN, pero a continuación se da la más extendida. Aparece un concepto que se conoce como estado oculto (hidden state) y que es un vector $\boldsymbol{h}_t \in \mathbb{R}^L$ y que va cambiando con el tiempo según el valor de la entrada en el instante t, $\boldsymbol{x}_t \in \mathbb{R}^n$. Este vector introduce precisamente el concepto de memoria permitiendo que la salida presente dependa de entradas anteriores. Por lo tanto, la salida de la capa, $\boldsymbol{y}_t \in \mathbb{R}^m$ depende de $\boldsymbol{h}_t$ y, por tanto de $\boldsymbol{x}_t$. A continuación se dan las expresiones de actualización de $\boldsymbol{h}_t$ y de $\boldsymbol{y}_t$:

$$
\begin{aligned}
\boldsymbol{h}_t &= f_h\left(W_h\cdot\boldsymbol{h}_{t-1} + W_x\cdot\boldsymbol{x}_t + b_h\right)\\
\boldsymbol{y}_t &= f_t\left(W_y\cdot\boldsymbol{h}_t + b_y\right)
\end{aligned}
\tag{2.19}
$$

donde $W_h \in \mathbb{R}^{L\times L}$, $W_x \in \mathbb{R}^{L\times n}$, $W_y \in \mathbb{R}^{m\times L}$, $b_h \in \mathbb{R}^L$ y $b_y \in \mathbb{R}^m$. La longitud L del estado oculto es un parámetro de diseño y las longitudes de la entrada y salida de la capa depende del problema concreto. En la figura 2.3 se da un esquema en el que se muestra la capa recurrente desplegada en el tiempo, lo que permite entender mejor como se modifica el estado oculto

Esta formulación permite calcular $\boldsymbol{y}_t$ en función de entradas anteriores $\boldsymbol{x}_t$. El problema de este tipo de redes es que, en general, se dice que no tienen memoria a largo plazo, esto es, las entradas de instantes muy anteriores apenas tienen influencia en la salida del instante actual. Para ver esto, se considera $J_i(\theta)$ la función de coste en el instante i, es

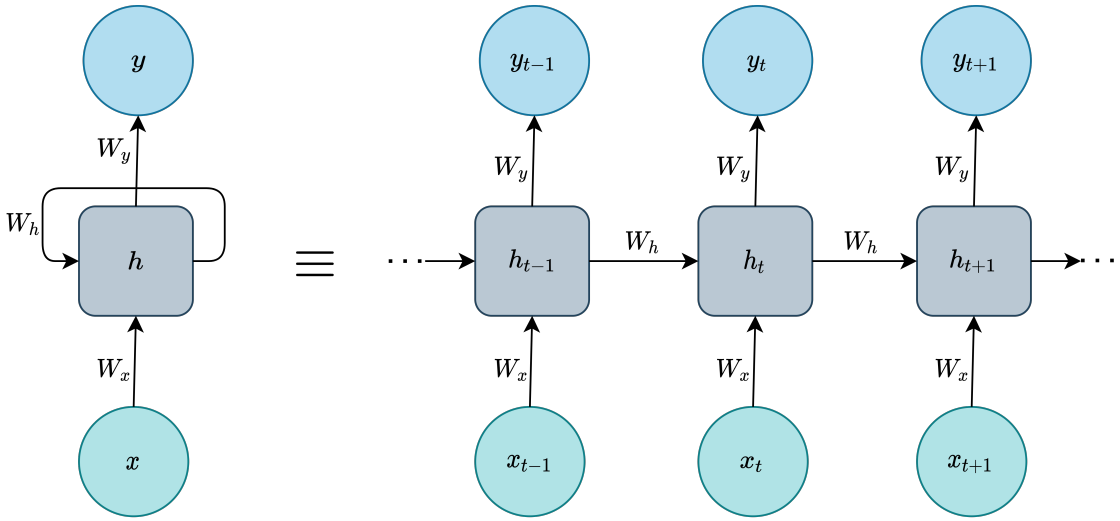


Fig. 2.3. Esquema de capa recurrente

obvio que depende de $\boldsymbol{h}_j$ con $j < i$. Así, para ver la influencia de $\boldsymbol{h}_j$ sobre J_i, se tiene que:

$$\begin{aligned}\frac{\partial J_i(\theta)}{\partial \boldsymbol{h}_j} &= \frac{\partial J_i(\theta)}{\partial \boldsymbol{h}_i}\prod_{j<t\leq i}\frac{\partial \boldsymbol{h}_t}{\partial \boldsymbol{h}_{t-1}} = \frac{\partial J_i(\theta)}{\partial \boldsymbol{h}_i}\prod_{j<t\leq i} diag\left(\sigma_h'\left(W_h\cdot\boldsymbol{h}_{t-1}+W_x\cdot\boldsymbol{x}_t+b_h\right)\right)W_h \\ &= \frac{\partial J_i(\theta)}{\partial \boldsymbol{h}_i}W_h^{i-j}\prod_{j<t\leq i} diag\left(\sigma_h'\left(W_h\cdot\boldsymbol{h}_{t-1}+W_x\cdot\boldsymbol{x}_t+b_h\right)\right)\rightarrow \\ \left\|\frac{\partial J_i(\theta)}{\partial \boldsymbol{h}_j}\right\| &\leq \left\|\frac{\partial J_i(\theta)}{\partial \boldsymbol{h}_i}\right\|\|W_h\|^{i-j}\prod_{j<t\leq i}\left\|diag\left(\sigma_h'\left(W_h\cdot\boldsymbol{h}_{t-1}+W_x\cdot\boldsymbol{x}_t+b_h\right)\right)\right\| \end{aligned} \tag{2.20}$$

En [30] se demuestra que sea γ tal que $\sigma_h(x) < \gamma$, si el radio espectral de W_h es menor que $\frac{1}{\gamma}$, entonces $\left\|\frac{\partial J_i(\theta)}{\partial \boldsymbol{h}_j}\right\|$ tiende a cero (vanishing gradient) cuando $i-j$ es grande. En cambio, cuando el radio espectral de W_h es mayor que $\frac{1}{\gamma}$, el gradiente crece de forma exponencial (exploding gradient).

Para solucionar los problemas anteriores, se proponen alternativas que sí permiten tener en cuenta entradas de instantes bastante anteriores para calcular el actual. Se dice que este tipo de capas tienen memoria a largo plazo. En concreto, la solución más popular se conoce como capas Long-Short Term Memory (LSTM) [31]. La que a continuación se da realmente no es la formulación más común de capas LSTM, sino que es una modificación llamada Peephole LSTM [32] y es la que sirve de paso para formular las capas LSTM convolucionales, que son las realmente utilizadas en este trabajo.

$$\begin{aligned}\boldsymbol{i}_t &= \sigma\left(W_{xi}\boldsymbol{x}_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i\right)\\ \boldsymbol{f}_t &= \sigma\left(W_{xf}\boldsymbol{x}_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f\right)\\ \boldsymbol{c}_t &= \boldsymbol{f}_t c_{t-1} + \boldsymbol{i}_t\tanh\left(W_{xc}\boldsymbol{x}_t + W_{hc}h_{t-1} + b_c\right)\\ \boldsymbol{o}_t &= \sigma\left(W_{xo}\boldsymbol{x}_t + W_{ho}h_{t-1} + W_{co}c_{t-1} + b_o\right)\\ \boldsymbol{h}_t &= \boldsymbol{o}_t\tanh(\boldsymbol{c}_t)\end{aligned} \tag{2.21}$$

donde:

- $\boldsymbol{x}_t \in \mathbb{R}^n$ es la entrada en el instante t
- $\boldsymbol{f}_t \in \mathbb{R}^m$ es la puerta de olvido en el instante t
- $\boldsymbol{i}_t \in \mathbb{R}^m$ es la puerta de entrada en el instante t
- $\boldsymbol{o}_t \in \mathbb{R}^m$ es la puerta de salida en el instante t
- $\boldsymbol{c}_t \in \mathbb{R}^m$ es la celda de memoria en el instante t
- $\boldsymbol{h}_t \in \mathbb{R}^m$ es el estado oculto en el instante t, que es la salida

La mayor innovación de las capas LSTM es el vector $\boldsymbol{c}_t$, que es la celda de memoria (memory cell), y actúa básicamente como un acumulador de información pasada. Por cada nueva entrada $\boldsymbol{x}_t$, la celda de memoria se actualiza añadiendo la información que índica la puerta de entrada $\boldsymbol{i}_t$ (input gate) y eliminando la que indica la puerta de olvido $\boldsymbol{f}_t$ (forget gate). La puerta de salida (output gate) $\boldsymbol{o}_t$ señala qué información de la celda de memoria se propaga a la salida $\boldsymbol{h}_t$. En la figura 2.4 se incluye un esquema de una capa Peephole LSTM.

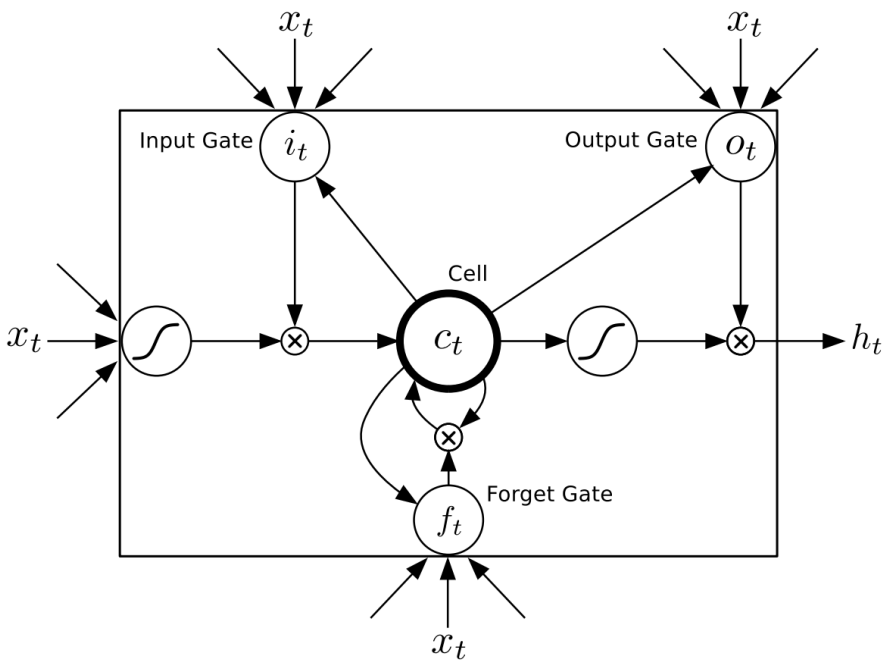


Fig. 2.4. Esquema capa Peephole LSTM. Recuperada de [33]

2.3.4. Capas LSTM Convolucionales

Hasta ahora se han explicado, por un lado las capas convolucionales, que permiten tener en cuenta la correlación espacial, y, por otro las capas LSTM, que permiten tener en cuenta la correlación temporal. Puede haber situaciones, como en el procesado de vídeos, donde interesa tener en cuenta las correlaciones en ambos sentidos. Por esto, surge el

concepto de capas LSTM convolucionales [34], que tiene una definición muy similar a la anterior dada de capas LSTM, pero toma como entradas matrices en lugar de vectores y se utiliza la correlación cruzada para calcular la salida y las variables internas de la capa. Así:

$$\begin{aligned} I_t &= \sigma\left(W_{xi} \star X_t + W_{hi} \star H_{t-1} + W_{ci} \odot C_{t-1} + b_i\right) \\ F_t &= \sigma\left(W_{xf} \star X_t + W_{hf} \star H_{t-1} + W_{cf} \odot C_{t-1} + b_f\right) \\ C_t &= F_t \odot C_{t-1} + I_t \odot \tanh\left(W_{xc} \star X_t + W_{hc} \star H_{t-1} + b_c\right) \\ O_t &= \sigma\left(W_{xo} \star X_t + W_{ho} \star H_{t-1} + W_{co} \odot C_{t-1} + b_o\right) \\ H_t &= O_t \odot \tanh\left(C_t\right) \end{aligned} \tag{2.22}$$

Aunque no se hace un estudio detallado de las dimensiones de las distintas variables debido a su extensión, se ha de tener en cuenta lo siguiente: X_t, I_t, F_t, C_t, O_t y H_t son tridimensionales, los pesos W_{xi}, W_{hi}, W_{ci}, W_{xf}, W_{hf}, W_{cf}, W_{xc}, W_{hc}, W_{xo}, W_{ho} y W_{co} tienen cuatro dimensiones, y los bias b_i, b_f, b_c y b_o son vectores. Esto da una idea de la complejidad computacional que requieren estas capas.

Las capas LSTM y LSTM convolucionales están basadas en la misma idea y la principal diferencia reside en que las primeras determinan el valor de las puertas, la celda de memoria y la salida basándose en capas fully connected, mientras que las segundas determinan estos valores basándose en capas convolucionales.

2.3.5. Batch Normalization

En la fase de entrenamiento, se busca minimizar una función de coste mediante un algoritmo iterativo. Este algoritmo iterativo, conforme la red va aumentando en número de capas, requiere de mucho tiempo para converger. Uno de los motivos por los que esto sucede es que las entradas a las capas difieren considerablemente entre iteraciones debido a que los parámetros de capas previas se modifican durante el entrenamiento. Este fenómeno se conoce como covariate shift [35] y consiste en que la distribución de las entradas a las capas cambia entre iteraciones. Por lo tanto, se propone un tipo de capa que permite normalizar las entradas a la capa siguiente. Este concepto recibe el nombre de Batch Normalization [36] porque, durante el entrenamiento, para normalizar las entradas, se calculan la media y desviación típica de todas las entradas de un lote o batch y se normalizan con respecto a éstos. Este tipo de capa tienen una peculiaridad y es que modifican su comportamiento entre las fases de entrenamiento e inferencia.

Fase de entrenamiento

Durante el entrenamiento, sea $B = \{x_1, x_2, ..., x_m\}$ el conjunto de entradas en el batch B, donde m es el tamaño de batch el conjunto de salidas $B' = \{y_1, y_2, ..., y_m\}$ se obtiene de

la siguiente forma:

$$\begin{aligned} \mu_B &= \frac{1}{m}\sum_{i=1}^{m} x_i \\ \sigma_B^2 &= \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_B)^2 \\ \hat{x}_i &= \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} \\ y_i &= \gamma\hat{x}_i + \beta,\ i = 1, 2, ..., m \end{aligned} \tag{2.23}$$

donde ϵ es un real positivo cercano a cero y los parámetros γ y β son determinados de la misma forma que el resto de parámetros durante el entrenamiento.

Fase de inferencia

Durante la fase de inferencia, la información ya no entra a las capas de la red mediante lotes, sino que entra individualmente. El problema de esto es que, si la forma de determinar la salida y en función de la entrada x en la fase de inferencia es igual a la descrita, la media de la entrada es ella misma. Por eso, el comportamiento de estas capas es distinto en entrenamiento que en inferencia. Para ello, se calculan también durante la fase de entrenamiento dos valores μ y σ de forma que por cada lote cambia su valor:

$$\begin{aligned} \mu &= \alpha \cdot \mu + (1 - \alpha) \cdot \mu_B \\ \sigma &= \alpha \cdot \sigma + (1 - \alpha) \cdot \sigma_B \end{aligned} \tag{2.24}$$

donde α es un parámetro de diseño. Así, los valores μ y σ se actualizan durante entrenamiento con cada batch y, una vez éste termina, estos valores quedan fijos y sirven para calcular la salida y de la capa según la entrada x, ahora sí, de forma similar al entrenamiento:

$$\begin{aligned} \hat{x} &= \frac{x - \mu}{\sqrt{\sigma^2 + \epsilon}} \\ y &= \gamma\hat{x} + \beta,\ i = 1, 2, ..., m \end{aligned} \tag{2.25}$$

Batch normalization y capas convolucionales

Si la capa de Batch Normalization está tras una capa convolucional, el cálculo de la salida en función de la entrada se modifica [37]. Esto se debe a que, ahora para el batch B la entrada a la capa es $\boldsymbol{X} = (x_{icjk}) \in \mathbb{R}^{m \times L \times H \times W}$, donde m es el tamaño de batch, L es el número de canales de la capa convolucional previa y $H \times W$ es el tamaño de las matrices de salida. De esta forma, la salida para cada canal $c \in \{1, 2, ..., L\}$, $Y_c = (y_{icjk})$ se define

como

$$\begin{aligned}
\mu_{B,c} &= \frac{1}{mHW}\sum_{i=1}^{m}\sum_{j=1}^{H}\sum_{k=1}^{W} x_{icjk} \\
\sigma^2_{B,c} &= \frac{1}{mHW}\sum_{i=1}^{m}\sum_{j=1}^{H}\sum_{k=1}^{W}(x_{icjk}-\mu_{B,c})^2 \\
\hat{x}_{icjk} &= \frac{x_{icjk}-\mu_{B,c}}{\sqrt{\sigma^2_{B,c}+\epsilon}} \\
y_{icjk} &= \gamma_c \hat{x}_{icjk} + \beta_c
\end{aligned} \tag{2.26}$$

es decir, para cada canal, el valor medio y varianza son escalares.

2.4. Autoencoders

Un autoencoder es una estructura típica de red neuronal que tiene dos partes principales: el codificador y el decodificador, y cuyo objetivo es que la entrada al codificador y la salida del decodificador sean lo más parecidas posible. La salida del codificador es la entrada del decodificador y, aunque, en general, no se pueden modelar estos elementos como funciones matemáticas invertibles, se podría asemejar al concepto de funciones inversas. La salida del codificador se llama código y es habitualmente de menor tamaño que la entrada.

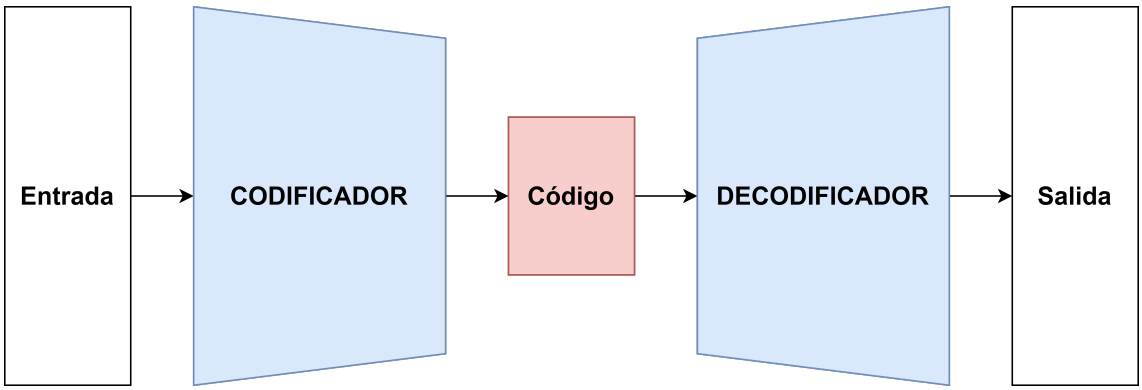


Fig. 2.5. Esquema básico de un Autoencoder

El concepto de codificar y decodificar es empleado en multitud de aplicaciones informáticas cuando interesa disminuir el tamaño que un fichero ocupa en memoria, o cuando se pretende que la transmisión de información entre dispositivos sea lo más eficiente posible. Tradicionalmente, se han investigado formas de codificar la información de manera que se extrajeran las características más relevantes del tipo de archivo en cuestión. Habitualmente, en este proceso de codificación y decodificación se admiten pérdidas, esto es, la entrada al codificador y la salida del decodificador pueden no ser exactamente iguales, pero sí comparten las propiedades más importantes. Un ejemplo de esto es el formato de

imagen JPEG [38], el cual se aprovecha de la poca resolución que tiene el ojo humano para cambios bruscos de texturas y color para reducir el tamaño de las imágenes. En la figura 2.6 se comparan una misma imagen en dos formatos, PNG sin pérdidas y JPEG con pérdidas y el tamaño que ocupan cada uno de los ficheros. A pesar de haber reducido su tamaño a menos de la cuarta parte, es necesario prestar mucha atención para percibir la diferencia.

(a) Imagen PNG (462 KB) (b) Imagen JPEG (104 KB)

Fig. 2.6. Una misma imagen en formato PNG (sin pérdidas) y JPEG (con pérdidas)

Un autoencoder, por lo tanto, realiza el tradicional proceso de codificar y decodificar, pero su punto fuerte reside en que no es necesario que un humano le dicte cuál es el proceso a realizar sino que, mediante el procedimiento tradicional que se emplea para entrenar redes neuronales, la propia red es capaz de extraer las características más relevantes de las señales con las que habitualmente se va a trabajar. Así pues, una red con esta estructura se puede ver como un sistema capaz de extraer características relevantes de un conjunto de datos por sí sola, lo cual es fundamental en infinidad de aplicaciones como la presente. De hecho, y muy en relación a lo anterior, se han utilizado para comprensión con pérdidas en imágenes dando unos resultados notables y comparables con el formato JPEG [39].

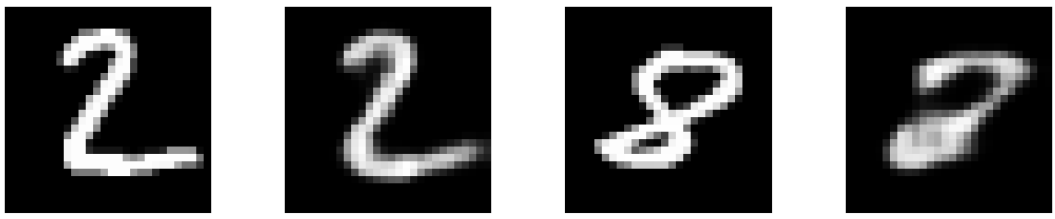
2.4.1. Autoencoder y detección de anomalías

La detección de anomalías es un campo ampliamente estudiado y para el que se han desarrollado un gran número de métodos para conseguir objetivos relacionados con la detección y clasificación de eventos anómalos [40, 41]. Habitualmente, se hace una división entre detección de anomalías mediante un método supervisado y no supervisado. El primer caso es útil y eficaz cuando se tienen los datos etiquetados, ya que el método consiste básicamente en ser capaces de encontrar qué distribución de probabilidad describen los datos normales y los anómalos. Por otro lado, la detección de anomalías para el caso no

supervisado trae consigo una serie de dificultades añadidas con respecto al caso anterior. Primero, no se tienen los datos etiquetados, lo que significa que no se posee información a priori de si un suceso concreto es anómalo o no. Además, no se pueden extraer características de un tipo de suceso, pues no se sabe cuáles de los datos se corresponden con ese tipo de sucesos.

Para este último caso, el cual es más relevante para este trabajo, se han propuesto soluciones basadas en algoritmos iterativos como K-means [42]. Como alternativa, surgen ideas basadas en el uso de redes neuronales, en particular de autoencoders [43, 44]. La idea fundamental que hay detrás es sencilla e intuitiva. Como se ha explicado en la sección 2.4, el objetivo es que la entrada y salida de éstos sea lo más parecida posible tras haber pasado por una versión comprimida de los datos. Para esto, las redes se entrenan con unos datos para que extraigan las características más relevantes de éstos. Así, si tras haber entrenado la red para codificar y decodificar un tipo de información en concreto, trata de codificar otro tipo de datos el rendimiento será claramente peor. Esto último es precisamente el principio en el que se basa el empleo de autoencoders como método de detección de anomalías, en entrenar las redes con datos normales de manera que éstos los reconstruya con un error mínimo, pero para los datos anómalos, en cambio, obtenga un error de una magnitud superior.

Para ayudar a explicar esto, se ha creado un autoencoder sencillo y se ha hecho uso de la base de datos MNIST [45]. Este autoencoder ha sido entrenado para codificar y decodificar imágenes que contienen un número dos manuscrito de distintas maneras. Posteriormente, se le han pasado como entrada dos imágenes, una que contiene un dos y otra que con un ocho, que haría las veces de anomalía. En la figura 2.7 se muestran los resultados. Se ve claramente cómo la imagen que contiene el número dos ha sido bien reconstruida, mientras que la que contiene el número ocho no lo ha sido tanto e incluso puede dar la impresión de que ha reconstruido bien la parte común que tienen el dos y el ocho.

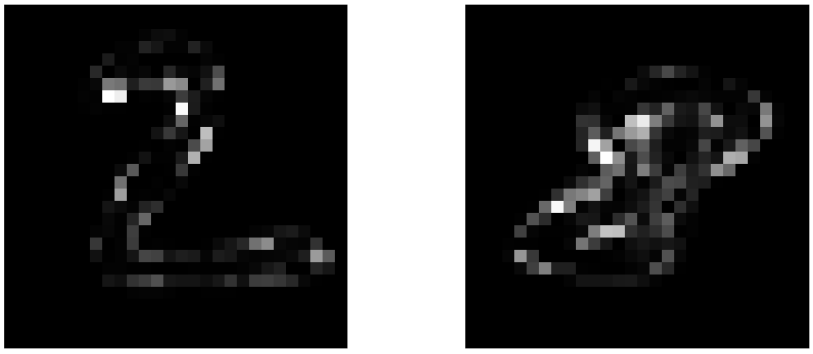


(a) 2 original (b) 2 decodificado (c) 8 original (d) 8 decodificado

Fig. 2.7. Comparación reconstrucción de un número 2 y un número 8 con un autoencoder entrenado para reconstruir imágenes con números 2

De hecho, en la figura 2.8 se muestra el error cuadrático instantáneo para la reconstrucción del dos y el ocho. Se ve que para el caso del dos las zonas más problemática son los bordes, pero que en la parte correspondiente al interior del número el error es

mínimo. En cambio, en el ocho los bordes tienen un error alto, pero es que además en alguna zona de la parte interior del número el error sigue siendo alto y, de hecho, en las zonas que es bajo es por su similitud con el número 2. Además, el error cuadrático medio para ambos casos, para el dos el valor es 0.0141, mientras que para el ocho es 0.0348, hecho que permitiría detectar el ocho como una anomalía fácilmente. A estas medidas que permiten discerninir entre datos normales y anómales se les conoce como marcadores de anomalías.



(a) ISE para el número 2 (b) ISE para el número 8

Fig. 2.8. Error cuadrático instantáneo de la reconstrucción de un número 2 y un número 8 con un autoencoder entrenado para reconstruir imágenes con números 2

2.5. Variational Autoencoders

Los autoencoders permiten obtener una versión comprimida de la entrada a partir de la cual se puede recuperar una copia que es parecida a la entrada. Sin embargo, estos sistemas no permiten generar nuevos datos parecidos a los originales, lo cual es interesante para muchas aplicaciones. Para ello, se proponen soluciones como los Variational Autoencoders (VAE) [7, 46], que son modelos que permiten generar datos nuevos procedentes de la misma distribución de probabilidad de los datos originales. Los modelos que, como estos, son capaces de generar datos nuevos artificiales reciben el nombre de modelos generativos.

El problema es que, dado un conjunto de datos $\boldsymbol{x} = (x^i)$, estimar la distribución de probabilidad de $p(x^i)$ es un problema intratable debido a que los datos son, en general, de alta dimensionalidad, ya que, es común, por ejemplo, utilizar estos sistemas para trabajar con imágenes. A su vez, si se piensa, por ejemplo, en imágenes de rostros humanos, uno puede darse de que hay una serie reducida de rasgos comunes que todos ellos tienen y que son los que verdaderamente deben prevalecer al crear nuevas imágenes de rostros.

Todo esto unido hace pensar en la posibilidad de modelar en términos de probabilidad

un espacio de menor dimensión que el original y que contenga la información esencial. Éste recibe el nombre de espacio latente (latent space) porque en él prevalecen ocultas y escondidas las principales características de los elementos del espacio original al que pertenecen las x^i. Por otro lado, hacen falta sistemas que permitan pasar del espacio original al reducido, y viceversa, lo cual lleva a pensar en estructuras con cierta similaridad a los autoencoders.

A partir de ahora, z es la variable correspondiente a los datos del espacio latente, que siguen una distribución $p(z)$. Igualmente, $p(z|x^i)$ es la distribución que siguen los valores z que han sido generados a partir de las entradas originales x^i. Finalmente, $p(x^i|z)$ es la distribución de probabilidad que siguen los datos del espacio original generados a partir del espacio latente.

Relacionando lo anterior con el concepto de autoencoder, se puede pensar que el conocimiento de $p(z|x^i)$ permite obtener z a partir de x^i, lo cual recuerda al concepto de codificador. Por otro lado, $p(x^i|z)$ permite obtener la reconstrucción a partir de un vector del espacio reducido, lo cual es similar a la idea del decodificador. Finalmente, el espacio latente se puede corresponder con el del código. Así, $p(z)$ explica qué distribución de probabilidad siguen los elementos del espacio latente. Es precisamente en esto último donde reside la mayor diferencia entre los autoencoders y los VAE, ya que estos últimos no son una solución determinista que da el valor del código para una entrada, sino que modelan qué distribución sigue el espacio latente y, por tanto, son capaces de generar nuevos elementos dentro de éste. Estos nuevos elementos, a través del decodificador se traducen en nuevos datos similares a los originales.

A continuación se da un desarrollo matemático que justifica la explicación previa. Para empezar, es muy complejo estimar $p(z|x^i)$ y $p(x^i|z)$. Debido a esto, el procedimiento sigue con aproximar dichas funciones con familias de distribuciones ya conocidas y estimar los parámetros que mejor realizan el ajuste. Así, se escoge $p_\phi(z|x^i)$ como aproximación de $p(z|x^i)$ y $q_\theta(x^i|z)$ como aproximación de $p(x^i|z)$, donde ϕ y θ son los parámetros de los que depende la aproximación. Las capas de la red neuronal contendrán precisamente los valores de ϕ y θ. Habitualmente, se elige que:

- $p(z) \sim N(0, I)$ es la distribución a priori
- $p_\phi(z|x^i) \sim N(\mu_z(x^i, \phi), \sigma_z(x^i, \phi))$ es la distribución a posteriori
- $q_\theta(x^i|z) \sim N(\mu_x(z, \theta), \sigma_x(z, \theta))$ es la verosimilitud

Por lo tanto, se va a configurar el codificador para que tenga dos salidas, que son μ_z y σ_z. Igualmente, el decodificador tendrá dos salidas: μ_x y σ_x.

Para lograr que estas aproximaciones sean acertadas, se emplea el concepto de divergencia Kullback–Leibler. Precisamente, la función de coste viene determinada en los VAE por este concepto y parte del hecho de que $p_\phi(z|x^i)$ y $p(z|x^i)$ deben ser similares y, por lo tanto, se ha de minimizar el valor de $D_{KL}(p_\phi(z|x^i)||p(z|x^i))$. Esta expresión se desarrolla a

continuación a partir de [47]:

$$D_{KL}(p_\phi(z|x^i)||p(z|x^i)) = \int_{-\infty}^{\infty} p_\phi(z|x^i) \log \frac{p_\phi(z|x^i)}{p(z|x^i)} dz$$

$$= \int_{-\infty}^{\infty} p_\phi(z|x^i) \log \frac{p_\phi(z|x^i)p(x^i)}{q_\theta(x^i|z)p(z)} dz = \int_{-\infty}^{\infty} p_\phi(z|x^i) \left(\log p(x^i) - \log \frac{q_\theta(x^i|z)p(z)}{p_\phi(z|x^i)} \right) dz$$

$$= \log p(x^i) \int_{-\infty}^{\infty} p_\phi(z|x) dz - \int_{-\infty}^{\infty} p_\phi(z|x^i) \log \frac{q_\theta(x^i|z)p(z)}{p_\phi(z|x^i)} dz$$

$$= \log p(x^i) - \int_{-\infty}^{\infty} p_\phi(z|x^i) \log \frac{p(z)}{p_\phi(z|x^i)} dz - \int_{-\infty}^{\infty} p_\phi(z|x^i) \log q_\theta(x^i|z) dz$$

$$= \log p(x^i) + D_{KL}\left(p_\phi(z|x^i)||p(z)\right) - E_{z\sim p_\phi(z|x^i)}\left[\log q_\theta(x^i|z)\right] \tag{2.27}$$

y se llega a que

$$\log p(x^i) - D_{KL}(p_\phi(z|x^i)||p(z|x^i)) = E_{z\sim p_\phi(z|x^i)}\left[\log q_\theta(x^i|z)\right] - D_{KL}\left(p_\phi(z|x^i)||p(z)\right) \tag{2.28}$$

La divergencia Kullback–Leibler es siempre positiva, por lo que se tiene que:

$$\log p(x^i) \geq E_{z\sim p_\phi(z|x^i)}\left[\log q_\theta(x^i|z)\right] - D_{KL}\left(p_\phi(z|x^i)||p(z)\right) \tag{2.29}$$

por lo que el término derecho es una cota inferior para $\log p(x^i)$ para todos los x^i, la cual se conoce como ELBO (Evidence Lower Bound).

En el lado izquierdo de la expresión 2.28, el término $\log p(x^i)$ no depende de ϕ y θ, por lo que a la hora de minimizar $D_{KL}\left(p_\phi(z|x^i)||p(z|x^i)\right)$ no es necesario tenerlo en cuenta. Queda así que la función de coste a minimizar para un VAE es:

$$L(\phi,\theta) = -\text{ELBO} = -E_{z\sim p_\phi(z|x^i)}\left[\log q_\theta(x^i|z)\right] + D_{KL}\left(p_\phi(z|x^i)||p(z)\right) \tag{2.30}$$

El primer término es menos el logaritmo de la verosimilitud (negative log likelihood) y, por lo tanto, al minimizarlo, se pretende maximizar la verosimilitud de x^i dado un $z \sim p_\phi(z|x^i)$. Como se ha establecido que $q_\theta(x^i|z) \sim N(\mu^x, \sigma^x)$, donde se impone que $(\sigma^x)_{ij} = 0$ si $i \neq j$, queda:

$$-\log q_\theta(x^i|z) = \frac{1}{2}\sum_j \left(\frac{x^i_j - (\mu_x)_j}{(\sigma_x)_j} \right)^2 + \log\left(\sqrt{2\pi}(\sigma_x)_j \right) \tag{2.31}$$

Este término recibe el nombre de coste de reconstrucción, ya que impone precisamente que la diferencia entre la entrada a la red (x^i) y la salida de ésta asociada a la media (μ^x) sea baja. De hecho, es común imponer que el decoder tenga una sola salida, quedando este término de la siguiente forma:

$$-\log q_\theta(x^i|z) = \frac{1}{2}\sum_j \left(x^i_j - (\mu_x)_j \right)^2 + \log\left(\sqrt{2\pi} \right) = c_1 \cdot MSE\left(x^i, (\mu_x)_j\right) + c_2 \tag{2.32}$$

Donde c_1 y c_2 son costantes que no dependen de los parámetros de la red, por lo que no afectan a la optimización de la red.

Y en el segundo término, al haberse impuesto que $p(z) \sim N(0, I)$, tal y como se demuestra en [7], queda que:

$$D_{KL}\left(p_\phi(z|x^i)||p(z)\right) = -\frac{1}{2}\sum_j (\sigma_x)_j^2 + (\mu_x)_j^2 - 1 - \log(\sigma_x)_j^2 \tag{2.33}$$

Este término recibe el nombre de coste variacional, ya que impone que el espacio latente generado por los datos de entrada siga una distribución cercana a $N(0, I)$.

2.5.1. Reparametrization Trick

Utilizar partes no deterministas dentro de la red tiene múltiples ventajas, pero es necesario introducir algún cambio para que el correcto funcionamiento no se vea alterado. En el algoritmo de Backward Propagation se ha de calcular el gradiente de la función de coste.

Sin embargo, el elemento del espacio latente z dado una entrada x^i se ha de elegir aleatoriamente de forma que $z \sim p_\phi(z|x^i) = N(\mu_z, \sigma_z)$. El hecho de que z se elija de forma aleatoria impide calcular el gradiente con respecto a ϕ y θ de la función de coste, por lo que se ha de buscar una forma que permita calcularlo según entradas únicamente deterministas.

Aquí se introduce lo que se conoce el Parametrization Trick. Éste consiste en que en lugar de tomar la entrada al codificador como una muestra aleatoria $z \sim N(\mu_z, \sigma_z)$ se escoge de la siguiente forma:

$$z = \mu_z + \sigma_z \odot \epsilon \tag{2.34}$$

donde $\epsilon \sim N(0, 1)$. De esta forma el término aleatorio viene dado por ϵ, que no depende de los parámetros de la red, por lo que no genera problemas en el cálculo del gradiente. La figura 2.9 ayuda a reforzar la comprensión de la idea.

2.6. Generative Adversarial Networks

En la sección anterior se han presentado los VAE, los cuales surgen con la intención de generar nuevos datos, es decir, son modelos generativos. Hay multitud de ideas que comparten este objetivo y una de las que más se ha popularizado estos últimos años son las Generative Adversarial Networks (GAN) [8].

Estos modelos proponen entrenar dos redes de forma simultánea para que compitan entre ellas. La primera de ellas recibe el nombre de Generador (G) y su objetivo es generar datos artificiales similares a los reales partiendo de ruido perteneciente a un espacio distinto al de los datos, en general. La segunda recibe el nombre de Discriminador (D) y su objetivo es discernir entre qué datos son reales y qué datos son artificialmente generados por G.

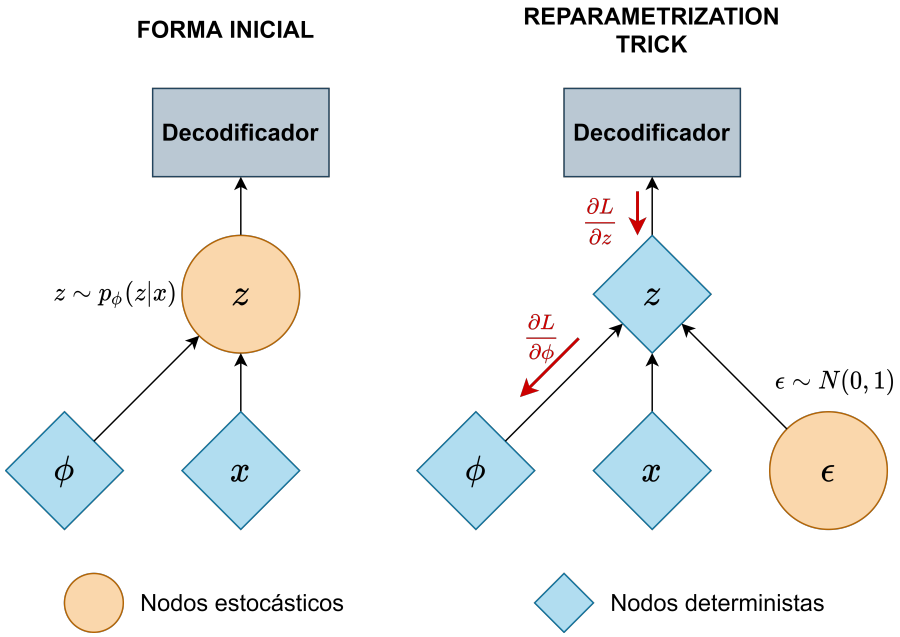


Fig. 2.9. Intuitivamente, en la forma propuesta inicialmente, no es posible calcular el gradiente, ya que hay un paso intermedio entre los parámetros ϕ y la salida que es estocástico. En cambio, con el reparametrization trick, este problema queda solucionado.

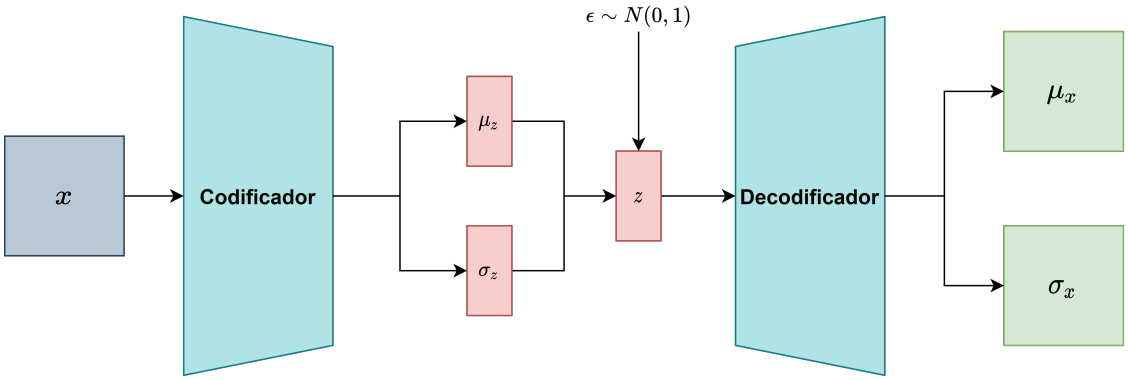


Fig. 2.10. Esquema básico de un Variational Autoencoder

Así el objetivo del generador es estimar la distribución p_g sobre los datos $\boldsymbol{x} \sim p_{data}(\boldsymbol{x})$. Para ello, se define la distribución del ruido que sirve de entrada a G como $p_z(\boldsymbol{z})$. Así, la red $G(\boldsymbol{z}, \theta_g)$ convierte el ruido $\boldsymbol{z}$ al espacio de los datos. Por otro lado, $D(\boldsymbol{x}, \theta_d)$ toma los datos reales o artificiales para dar a su salida un escalar que, habitualmente, estará entre 0 y 1, lo cual lleva a pensar en una función sigmoide o similar para la última capa. De hecho, $D(\boldsymbol{x})$ se puede interpretar como la probabilidad de que $\boldsymbol{x}$ provenga de los datos reales. En la figura 2.11 se muestra un esquema de una GAN.

En cuanto al entrenamiento, se entrena D para maximizar el valor de $D(\boldsymbol{x})$ cuando $\boldsymbol{x}$ proviene de datos reales y, simultáneamente, se entrena G para minimizar $1 - D\left(G\left(\boldsymbol{z}\right)\right)$. Así, para optimizar se plantea el siguiente problema Minimax:

$$\min_G \max_D V(D,G) = \min_G \max_D \mathbb{E}_{x \sim p_data(\boldsymbol{x})} \left[\log\left(D(\boldsymbol{x})\right)\right] + \mathbb{E}_{z \sim p_z(\boldsymbol{z})} \left[\log\left(1 - D(G\left(\boldsymbol{x}\right))\right)\right] \quad (2.35)$$

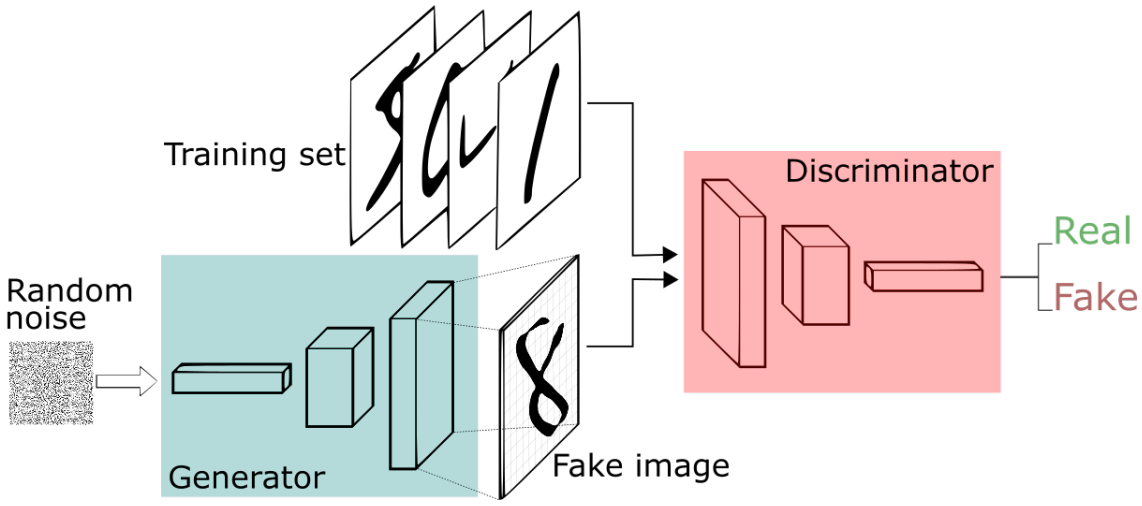


Fig. 2.11. Esquema una Generative Adversarial Network. Recuperada de [48]

De esta forma, las funciones de coste del generador (L_G) y del discriminador (L_D) a minimizar quedan de la siguiente forma:

$$L_D = -L_G = -\mathbb{E}_{x\sim p_{data}}\left[\log\left(D(\boldsymbol{x})\right)\right] - \mathbb{E}_{z\sim p_z(\boldsymbol{z})}\left[\log\left(1 - D(G\left(\boldsymbol{z}\right))\right)\right] \tag{2.36}$$

Aunque se puede utilizar la expresión 2.36 para optimizar las dos redes, según [8], es recomendable redefinir la función de coste del generador por la siguiente:

$$L_G = -\mathbb{E}_{z\sim p_z(\boldsymbol{z})}\left[\log\left(D(G\left(\boldsymbol{z}\right))\right)\right] \tag{2.37}$$

El motivo por el que se realiza esto es para evitar el problema del desvanecimiento de gradiente. Éste surge aquí si se emplea la función de coste L_G definida en 2.36, ya que si el discriminador consigue minimizar el término $-\mathbb{E}_{z\sim p_z(\boldsymbol{z})}\left[\log\left(1 - D(G\left(\boldsymbol{z}\right))\right)\right]$, entonces, para modificar los parámetros θ_G se tiene que el gradiente de la función de coste L_G es casi nulo. La expresión 2.37 significa que se entrena G para maximizar $D(G(\boldsymbol{z}))$. El nombre que reciben las GAN con estas modificaciones es Non-Saturating GAN.

2.7. Teoría de decision Bayesiana

El fin último de este proyecto es desarrollar un sistema que decida si hay o no una anomalía en un lugar y en un instante concreto. Por esto, en esta sección se dan unas pinceladas básicas de algunas ideas que conviene conocer sobre teoría de la decisión.

La teoría de la decisión Bayesiana [49] toma como punto de partida el Teorema de Bayes, el cual se enuncia a continuación. Sea $H_1, H_2, ..., H_n$ un conjunto de sucesos mutuamente excluyentes y exhaustivos, y tales que la probabilidad de cada uno de ellos es distinta de cero y sea B un suceso cualquiera del que se conocen las probabilidades condicionales $P(B|H_i)$. Entonces, la probabilidad $P(H_i|B)$ viene dada por la expresión:

$$P(H_i|B) = \frac{P(B|H_i)P(H_i)}{P(B)} \tag{2.38}$$

donde:

- $P(H_i)$ son las probabilidades a priori,
- $P(B|H_i)$ es la probabilidad de B en la hipótesis H_i (likelihood de H_i),
- $P(H_i|B)$ son las probabilidades a posteriori.

Para el caso de este trabajo existen únicamente dos sucesos ya que el fin no es categorizar las anomalías sino que es exclusivamente detectarlas. Entonces, el suceso H_1 se corresponde con la ausencia de eventos anómalos y el suceso H_2 con la presencia de éstos.

Bien, antes de realizar este sistema de detección de anomalías se podrían haber estimado unas probabilidades a priori de normalidad y de anomalía, esto es, $P(H_1)$ y $P(H_2)$, respectivamente donde, por la propia definición de anomalía, es claro que $P(H_1) > P(H_2)$. Bajo este supuesto y sin el conocimiento de información adicional, lo que la teoría de la decisión dice es que el suceso ocurrido es siempre H_1. Esto significa pensar que nunca hay anomalías, una hipótesis falsa. Pero en caso de poseer información adicional, se puede utilizar el Teorema de Bayes.

Para continuar, surge un nuevo concepto llamado función de coste de error. Lo que ésta nos indica es cómo de costosa es una acción en relación al suceso ocurrido. Esto es, dadas las acciones $\alpha_1, \alpha_2, ..., \alpha_r$ y los sucesos $H_1, H_2, ..., H_n$, entonces $\lambda(\alpha_i|H_j) = \lambda_{ij}$ indica el coste de realizar la acción α_i bajo la ocurrencia del suceso H_j. A partir de esto, se define el coste esperado $R(\alpha_i|B)$ de la acción α_i bajo la ocurrencia del suceso B según la expresión:

$$R(\alpha_i|B) = \sum_{j=1}^{n} \lambda_{ij} P(H_j|B) \tag{2.39}$$

Con esta definición se elige la acción que minimiza este coste esperado, esto es, $\alpha*$ según la expresión:

$$\alpha* = \arg\min_{\alpha_i} R(\alpha_i|B) \tag{2.40}$$

Esto último, aplicado a situaciones con dos sucesos, como es el caso, si además se consideran también dos acciones α_1 y α_2, que se corresponden con decidir el suceso 1 o el 2, respectivamente, se transforma en una expresión cerrada y un umbral que permite decidir si ha ocurrido un suceso u otro.

$$\begin{aligned} R(\alpha_1|B) &= \lambda_{11} P(H_1|B) + \lambda_{12} P(H_2|B) \\ R(\alpha_2|B) &= \lambda_{21} P(H_1|B) + \lambda_{22} P(H_2|B) \end{aligned} \tag{2.41}$$

Por lo tanto, y según lo indicado en 2.40, a continuación se desarrolla la expresión que determina bajo qué condición se ha de elegir la acción α_1, es decir, interpretar que ha

ocurrido el suceso H_1:

$$\begin{gathered} R(\alpha_1|B) < R(\alpha_2|B) \leftrightarrow (\lambda_{21} - \lambda_{11})P(H_1|B) > (\lambda_{12} - \lambda_{22})P(H_2|B) \leftrightarrow \\ (\lambda_{12} - \lambda_{22})P(B|H_1)P(H1) > (\lambda_{21} - \lambda_{22})P(B|H_2)P(H2) \leftrightarrow \\ \frac{P(B|H_1)}{P(B|H_2)} > \frac{\lambda_{12} - \lambda_{22}}{\lambda_{21} - \lambda_{11}} \frac{P(H_2)}{P(H_1)} \end{gathered} \tag{2.42}$$

La fuerza de esta expresión es que establece un umbral que es independiente de la observación B. El término $\frac{P(B|H_1)}{P(B|H_2)}$ recibe el nombre de ratio de verosimilitud (likelihood ratio), por lo que la solución del problema se reduce a decidir H_1 si el ratio de verosimilitud es mayor que un umbral independiente de la observación B y H_2 en caso contrario.

A la hora de calificar el rendimiento de un sistema se emplean los términos que se explican en la tabla 2.1 y que están íntimamente relacionados con las funciones de coste λ_{ij}.

Concepto	Suceso ocurrido	Suceso detectado	Función de coste relacionada
Verdadero negativo	H_1	H_1	λ_{11}
Falso negativo	H_1	H_2	λ_{12}
Falso positivo	H_2	H_1	λ_{21}
Verdadero positivo	H_2	H_2	λ_{22}

TABLA 2.1. CONCEPTOS PARA MEDIR RENDIMIENTO DE UN DETECTOR

Y a partir de estos conceptos, se describen otros dos que sirven para valorar rápidamente el rendimiento de un sistema de detección. Éstos son la probabilidad de detección (P_d), que indica la probabilidad de detectar una anomalía cuando la hay, y la probabilidad de falsa alarma(P_{fa}), que hace referencia a detectar una anomalía cuando no la hay. A continuación se dan sus definiciones en función de los conceptos de la tabla 2.1.

$$\begin{aligned} P_d &= \frac{VP}{VP + FN} \\ P_{fa} &= \frac{FP}{FP + VN} \end{aligned} \tag{2.43}$$

Partiendo de los dos términos anteriores, surge el concepto de curva ROC (Receiver operating characteristic) [9], que es una representación que tiene en su eje de abcisas el valor de P_{fa} y en su eje de ordenadas en valor de P_d. Estas curvas son especialmente interesantes porque permiten analizar de manera rápida el compromiso entre las probabilidades de detección y falsa alarma. En la figura 2.12 se muestran unas curvas ROC que corresponden a tres sistemas A, B y C con distintas prestaciones. El A es el que peor rendimiento proporciona resultando no informativo y es el peor caso posible de curva ROC. El sistema B es uno realista y en el que se ve bien el compromiso entre probabilidades de falsa alarma y detección. Por último, el sistema C da el rendimiento perfecto, ya que se

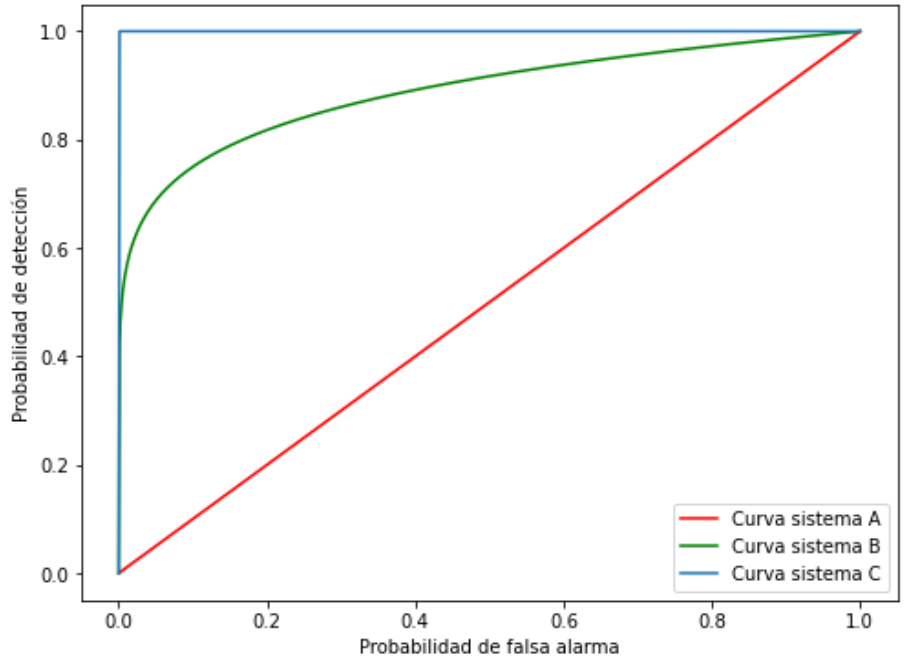


Fig. 2.12. Curvas ROC que describen el rendimiento de tres sistemas con distintas prestaciones

puede conseguir una probabilidad de detección de 1 y una de falsa alarma de 0 de forma simultánea.

Por último, surge otro concepto, que es el área bajo la curva ROC (AUC), que proporciona una medición agregada del rendimiento en todos los umbrales de clasificación posible. Este valor oscila de 0 a 1. Un modelo en el que el 100 % de las predicciones son correctas tendrá un AUC de 1 y uno en el que el 100 % son incorrectas tendrá un AUC de 0.

3. METODOLOGÍA

Una vez explicados los conceptos teóricos necesarios para entender este trabajo, en este capítulo se presentan la forma de procesar la información procedente de los DAS.

3.1. Preprocesado de la información

La información procedente del sensor DAS es una matriz de gran tamaño, ya que contiene información en tiempo y distancia. Así, si $S = (s_{ij}) \in \mathbb{R}^{L_t \times L_d}$ es la salida del sensor, s_{ij} contiene información sobre el instante i en la posición j, y L_t y L_d son el número de muestras temporales y espaciales, respectivamente. Para utilizar esta matriz como entrada a los sistemas diseñados, es necesario aplicar unos cambios sobre ella, que se explican a continuación.

3.1.1. Diezmado

Habitualmente, la frecuencia de muestreo con la que trabajan los sensores se traduce en una cantidad de datos demasiado alta que puede ralentizar su procesado. Es por este motivo, que se propone aplicar un factor de diezmado temporal D_t, lo cual significa disminuir la frecuencia de muestreo en un factor D_t. Este se va a considerar uno de los parámetros de diseño del algoritmo. Entonces, sea $S = (s_{ij}) \in \mathbb{R}^{L_t \times L_d}$ la matriz original, tras el diezmado queda S^D, que se define como:

$$S^D = (s^D_{ij}) \in \mathbb{R}^{L'_t \times L_d},\ s^D_{ij} = s_{(i-1)D_t+1,j} \tag{3.1}$$

donde $L'_t = \left\lceil \frac{L_t}{D_t} \right\rceil$ es el número de filas tras el diezmado temporal.

En cuanto a la resolución espacial, se ha decidido no aplicar un factor de diezmado sobre ella, ya que no es demasiado elevada, aunque el procedimiento para hacerlo sería análogo.

3.1.2. Patching

Una técnica bastante utilizada en el procesado de señales y que es empleada a lo largo de este trabajo es el patching (parcheado). Su idea es sencilla, pero conviene definir algunos parámetros. Para empezar, hay dos factores, N y M que hacen referencia a la longitud y desplazamiento de parche, respectivamente. Dados estos factores y un vector $\boldsymbol{x}$ de longitud L, se obtiene una matriz cuyas filas contienen cada una de las partes en los que

se divide dicho vector, las cuales se llaman parches. Esto es, sea $\boldsymbol{x} = (x_1, x_2, ..., x_L) \in \mathbb{R}^L$, la función patching g_P se define como:

$$g_P : \mathbb{R}^L \to \mathbb{R}^{W \times N},\ g_P(\boldsymbol{x}) = (a_{ij}) \in \mathbb{R}^{W \times N},\ a_{ij} = x_{M(i-1)+j} \tag{3.2}$$

donde $W = \left\lceil \frac{L-N}{M} + 1 \right\rceil$ y es el número de parches.

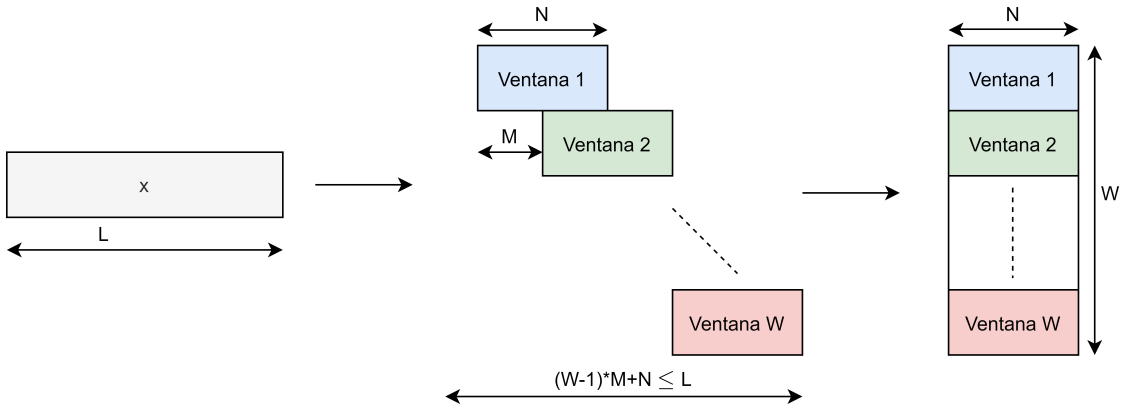


Fig. 3.1. Proceso de parcheado

Este concepto se puede extrapolar a señales bidimensionales. Concretamente, para el caso de este problema, se definen cuatro parámetros N_t y N_d, que son la longitud de parche en las dimensiones temporales y espaciales, respectivamente, y M_t y M_d, que son el desplazamiento de parche en las dimensiones temporales y espaciales, respectivamente. Así, dada la señal diezmada por un factor D_t, $S^D = (s^D_{ij})$, la salida del patching es $S^P \in \mathbb{R}^{W_t \times W_d \times N_t \times N_d}$ y se define como:

$$S^P = (s^P_{i_t,i_d,j_t,j_d}) \in \mathbb{R}^{W_t \times W_d \times N_t \times N_d},\ s^P_{i_t,i_d,j_t,j_d} = s^D_{M_t(i_t-1)+j_t, M_d(i_d-1)+j_d} \tag{3.3}$$

donde $W_t = \left\lceil \frac{L'_t - N_t}{M_t} + 1 \right\rceil$ y $W_d = \left\lceil \frac{L_d - N_d}{M_d} + 1 \right\rceil$.

Finalmente, a la salida, se le aplica un concepto que se conoce como flattening y que pasa de una matriz de matrices a un vector de matrices S^F, que se define como:

$$S^F = (s^F_{kij}) \in \mathbb{R}^{W_t \cdot W_d \times N_t \times N_d},\ s^F_{kij} = s^P_{k_t,k_d,i,j} \tag{3.4}$$

donde $k_d = \left\lceil \frac{k}{W_t} \right\rceil$ y $k_t = k - (k_d - 1) \cdot W_t$.

3.1.3. Escalado

Por último, se aplica el escalado MinMax a cada una de los parches. De esta forma, la entrada a la red X, queda:

$$X = (X_k) \in \mathbb{R}^{W_t \cdot W_d \times N_t \times N_d},\ X_k = \frac{S^F_k - \text{mín}(S^F_k)}{\text{máx}(S^F_k) - \text{mín}(S^F_k)} \tag{3.5}$$

Así, las entradas a la red van a ser las matrices $X_k \in \mathbb{R}^{N_t \times N_d},\ k \in \{1, 2, ..., W_t \cdot W_d\}$.

3.2. Modelos de aprendizaje profundo evaluados

En este punto, se cuenta con un conjunto de matrices X_k de tamaño $N_t \times N_d$ que deben ser procesadas mediante un sistema que sea capaz de generar marcadores de anomalías. Para ello, todos las propuestas se basan en el mismo concepto y es comparar de alguna forma la entrada y salida de sistemas de aprendizaje profundo. Como se ha explicado en la sección 2.4.1, los autoencoders han sido ampliamente utilizados en el campo de detección de anomalías y, consecuentemente, los VAE como posible mejora a los anteriores. Igualmente, se propone utilizar redes GAN para detección de anomalías con ánimo de abordar el problema desde una perspectiva distinta.

3.2.1. Propuesta 1 - Autoencoder con capas convolucionales

De las cuatro alternativas presentadas en este trabajo, esta es la más elemental y fue desarrollada en [3]. Este primer sistema está basado en el concepto de autoencoder y hace uso de capas convolucionales para sacar un máximo partido a la correlación espacio-temporal dentro de los parches. En la figura 3.2 se muestra un esquema de la red siguiendo la nomenclatura de la sección 2.3.2. Se han empleado tres capas para el codificador y tres para del decodificador y el autoencoder es simétrico por simplicidad. La capa 7 no tiene función de activación y su principal misión es que la salida de la red sea una única matriz. Como función de coste se utiliza el error cuadrático medio entre las matrices X_k y $\hat{X}_k$, que son las entradas y salidas de la red, respectivamente.

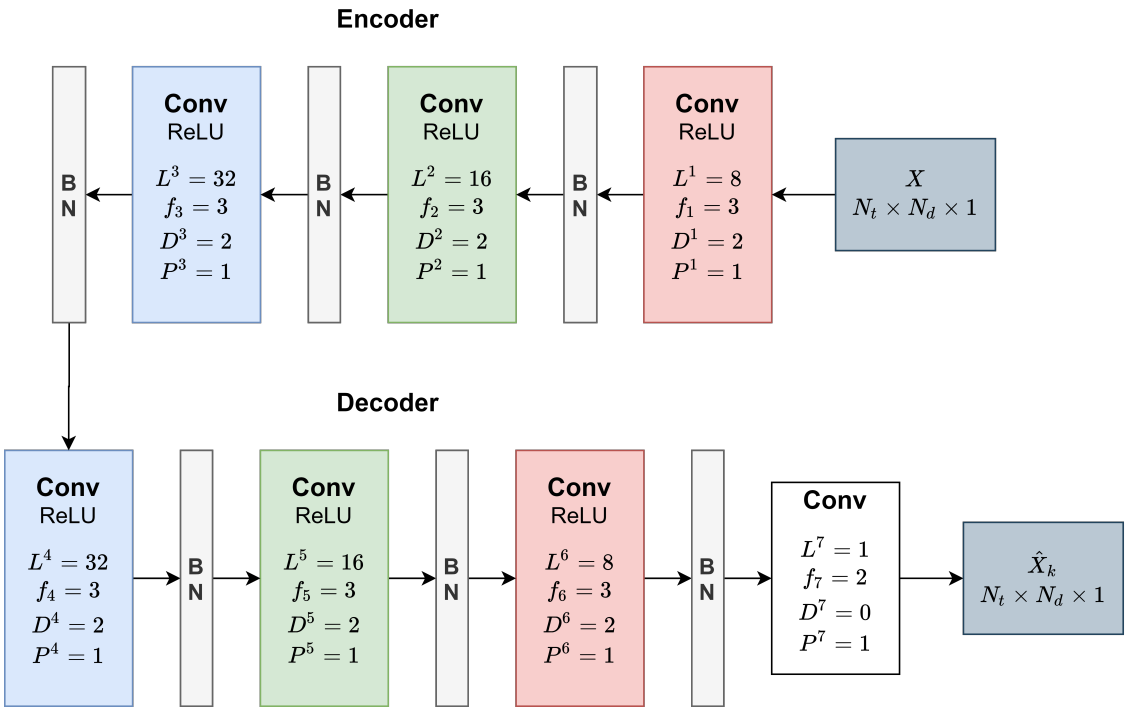


Fig. 3.2. Estructura de la red de la propuesta 1 - Autoencoder con capas convolucionales

La red se entrena con datos libres de anomalías, de forma que aprenda a codificar y decodificar únicamente datos normales, como se explica en 2.4.1. Una vez la red ha sido entrenada, al haberse utilizado el error cuadrático medio como función de coste, se propone utilizar el error cuadrático instantáneo entre X_k y $\hat{X}_k$ como marcador de anomalía. Por lo tanto, se considerará que hay una anomalía en un punto si el error cuadrático correspondiente a ese punto es elevado o, equivalentemente, si la probabilidad de que haya un error cuadrático medio tal es reducida.

$$A_k^{SE} = (a_{kij}) \in \mathbb{R}^{N_t \times N_d},\ a_{kij} = \|x_{kij} - \hat{x}_{kij}\|_2 \tag{3.6}$$

Tras obtener todas las matrices A_k se realiza un proceso inverso al patching, obteniendo de nuevo un a matriz de tamaño $L_t' \times L_d$ que tendrá valores altos en zonas donde existe una anomalía y valores bajos en zonas de normalidad.

3.2.2. Propuesta 2 - Autoencoder con capas LSTM convolucionales

El sistema anterior permite tener en cuenta la correlación espacio-temporal dentro de cada ventana, pero cada entrada la trata como independiente, cosa que se traduce en una pérdida de información. El conjunto de matrices que constituyen la entrada a los sistemas de procesado están ordenados temporalmente, cosa que se puede aprovechar utilizando capas convolucionales LSTM.

Cuando se implementan las capas LSTM y, equivalentemente, las LSTM convolucionales, las entradas se pasan en grupos de longitud fija N_g, que es la verdadera memoria que tienen las capas. Por cada nuevo grupo de capas, el estado oculto y la celda de memoria se resetean. Por lo tanto, las entradas X_k a estos sistemas tendrán tamaño $N_g \times N_t \times N_d$. En este caso, se va a tomar que $N_g = 10$ siempre, pero podría ser un parámetro de diseño cuya influencia en el rendimiento se analizara. En la figura 3.3 se incluye un esquema de este sistema.

La idea es similar al anterior, pero se incluyen algunas capas convolucionales LSTM que permiten incluir cierta memoria y no analizar los parches consecutivos de forma independiente. Además, las capas 1, 2, 6, 7 y 8, es decir, las convolucionales actúan sobre cada una de las N_g matrices de cada grupo individualmente.

Este sistema sigue el mismo principio que el anterior y se utiliza el error cuadrático medio como función de coste. Además, el marcador de anomalía es de nuevo el error cuadrático instantáneo entre entrada y salida de la red.

3.2.3. Propuesta 3 - Variational Autoencoder

Hasta el momento, se han presentado dos soluciones deterministas, pero, tal y como se ha explicado, existen algunas ventajas al utilizar modelos generativos cuyos parámetros

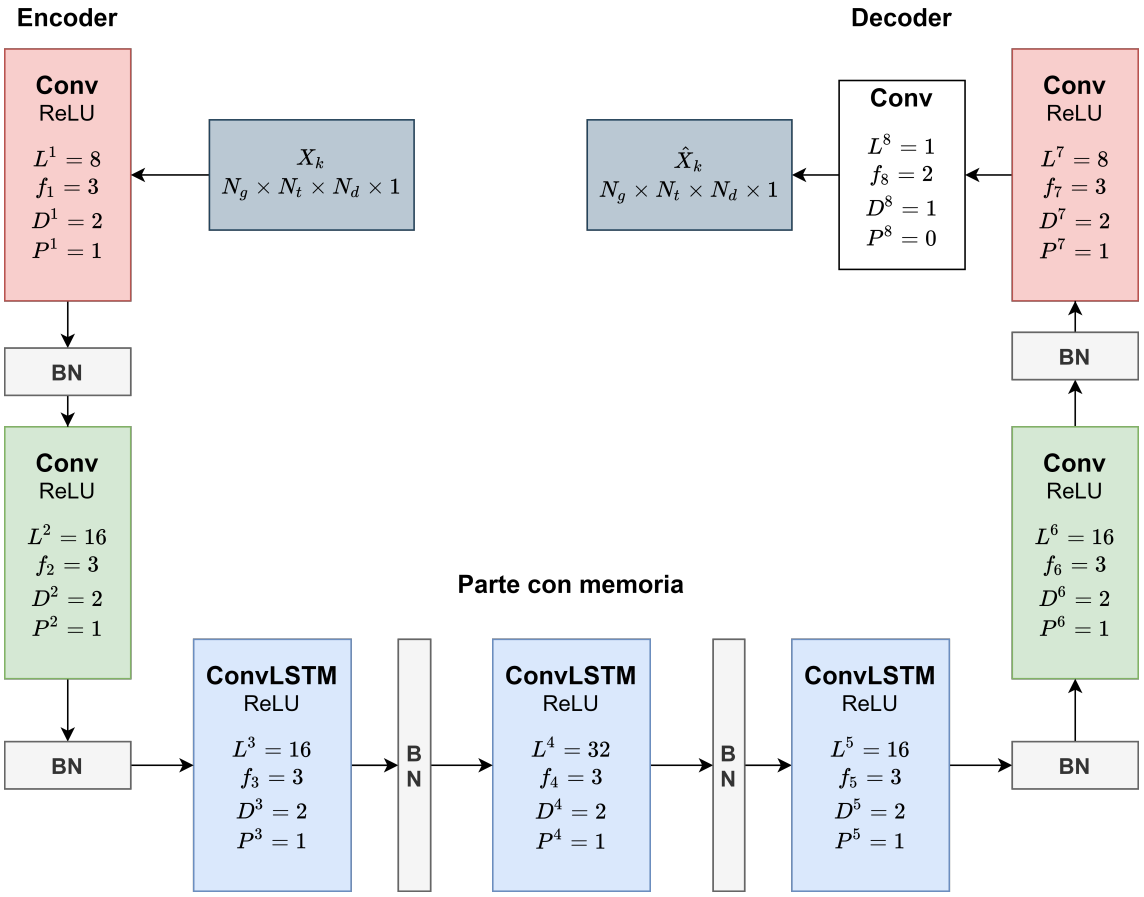


Fig. 3.3. Estructura de la red de la propuesta 2 - Autoencoder con capas LSTM convolucionales

sirven para modelar distribuciones de probabilidad. En este sistema se propone implementar una estructura similar a la del primero, pero introduciendo la idea del Variational Autoencoder. El esquema es el mismo que en el del primer caso, lo cual permite compararlos a igualdad de condiciones. Se muestra en la figura 3.4

La idea de entrenar la red sólo con datos normales prevalece, pero en este caso, la función de coste se modifica con respecto a los dos anteriores y es la que se desarrolla en la sección 2.5, esto es, la suma del error de reconstrucción entre entrada y salida de la red y de la divergencia KL entre $N(\mu_z, \sigma_z)$ y $N(0,1)$. Por lo tanto, se procura que entrada y salida sean similares y que el espacio latente siga una distribución cercana a una normal con media nula y varianza unidad. Debido a lo anterior, aquí se utilizan dos marcadores de anomalías, lo cual permite reducir a casi la mitad el tiempo de procesado y que están relacionados de nuevo con la función de coste.

El primer marcador es la divergencia KL entre $N(\mu_z, \sigma_z)$ y $N(0,1)$. A diferencia del error cuadrático instantáneo, éste da un valor para cada parche, en lugar de dar uno para cada elemento del mismo. Se define en 3.7. El segundo marcador es de nuevo el error cuadrático instantáneo, que sí da un valor para cada elemento del parche.

$$A_{KL} = \frac{1}{2}\sum_{j=1}^{64}\left((\mu_z)_j^2 + (\sigma_z)_j^2 - \log(\sigma_z)_j^2 - 1\right) \in \mathbb{R} \tag{3.7}$$

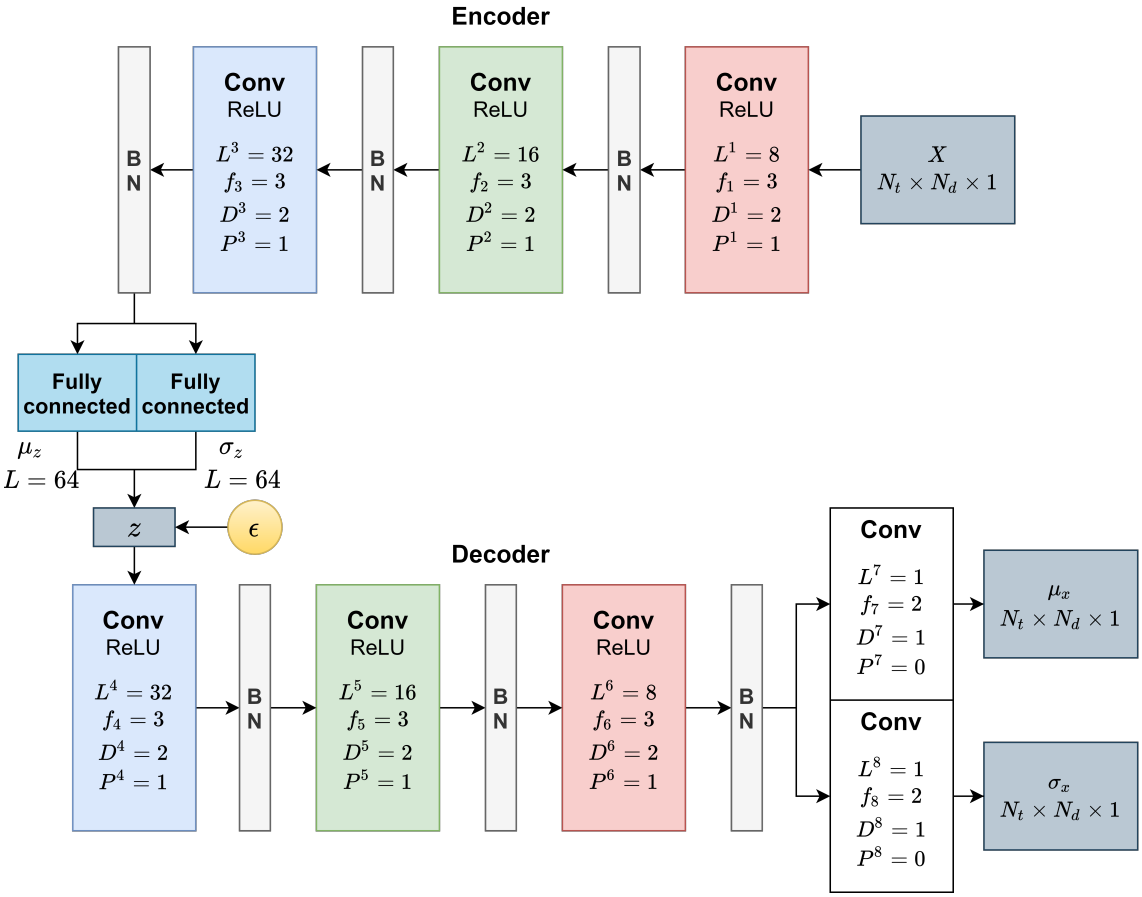


Fig. 3.4. Estructura de la red de la propuesta 3 - Variational Autoencoder

Aunque puede parecer que utilizar el primer marcador no aporta demasiado, ya que simplemente da información del parche completo y se pierde resolución, es la clave para reducir considerablemente la cantidad de procesado que se debe realizar. Esto se debe a que no es necesario utilizar la parte del decodificador para calcularlo. Por lo tanto, se propone utilizarlo como un filtro que decide qué parches son potencialmente anómalos o, equivalentemente, pueden contener elementos anómalos. De esta forma, sólo los parches que pasan este filtro serán decodificados y sólo para ellos se calculará el error cuadrático instantáneo. La idea es que, por la propia definición de anomalía, es muy improbable que las haya, por lo que casi ningún parche pasará el primer filtro impuesto y, por tanto, casi ningún parche pasará a través del decodificador, resultando esto en que para la mayoría de situaciones sólo se tratarán los datos en el codificador.

3.2.4. Propuesta 4 - GANomaly

Igual que sucede con los VAE, estos últimos años se han propuesto métodos para detectar anomalías utilizando GAN. La primera aproximación se desarrolla en [50] y recibe el nombre de AnoGAN. En [51] se propone otro método llamado EGBAD (Efficient GAN-Based Anomaly Detection). Uno de los que mejores resultados ha dado según la literatura [52] se desarrolla en [53] y recibe el nombre de GANomaly. Los autores de esta

idea proponen otra que recibe el nombre de skip-ganomaly [54]. Este último es el que se ha utilizado aquí.

El esquema de este sistema se muestra en la figura 3.5. Esta propuesta está compuesta por dos redes, una de ellas es el generador y tiene la estructura de un autoencoder cuyo objetivo es reconstruir las imágenes originales. Por otro lado, el discriminador toma las imágenes originales y las artificiales como entradas y su objetivo es ser capaz de distinguirlas e identificar como falsas las artificiales y como reales las originales. De esta forma el generador trata de reconstruir las imágenes lo mejor posible y el discriminador busca mecanismos para detectar imágenes no procedentes del conjunto de las originales. De nuevo, los datos con los que se entrena la red están libres de anomalías, por lo que tanto el generador como el discriminador aprenden a trabajar para este tipo de datos.

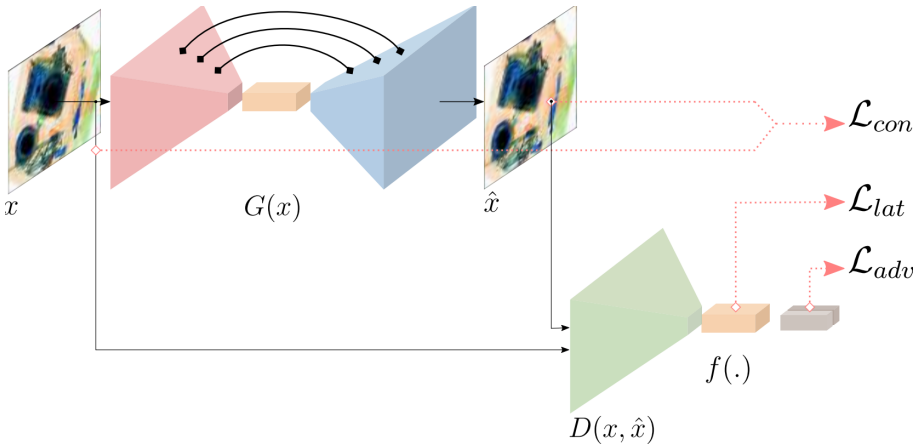


Fig. 3.5. Esquema del sistema Skip-ganomaly. Recuperada de [54]

Función de coste del discriminador

En este caso, la función de coste del discriminador se define como:

$$L_D = -\mathbb{E}_{x\sim p_{data}}\left[\log\left(D(\boldsymbol{x})\right)\right] - \mathbb{E}_{x\sim p_{data}}\left[\log\left(1 - D(G\left(\boldsymbol{x}\right))\right)\right] \tag{3.8}$$

Función de coste del generador

Por otro lado, la función de coste del generador, se descompone como la suma de dos términos. El primero de ellos incita al generador a aprender la información contextual de las entradas, es decir, trata de que entrada y salida sean similares píxel a píxel. Se define como el error medio entre entrada y salida y se conoce como pérdida contextual:

$$L_{con} = \mathbb{E}_{x\sim p_{data}}\|\boldsymbol{x} - G(\boldsymbol{x})\|_1 \tag{3.9}$$

El segundo término tiene como objetivo que las imágenes creadas por el generador y las originales sean indistinguibles. Para esto, en lugar de definir la función dada en 2.37, en

[55] se propone una alternativa que se ha probado empíricamente que mejora la convergencia de las dos redes. Esta función de coste recibe el nombre de feature matching y, en lugar de maximizar la salida del discriminador, trata de que la salida de una capa concreta sea lo más parecida posible para los datos originales y para los artificiales. Intuitivamente, esto implica que ambos tipos de imágenes tengan características elementales similares, ya que la salida de esa capa intermedia contiene la información más relevante de la entrada, debido a que la estructura del discriminador es la de un decodificador. En este caso, se propone que esta capa sea la previa a la salida del discriminador, que se corresponde con la versión más comprimida de los datos. Debido a esto, recibe el nombre de pérdida latente. Así, sea f la función que describe la salida de dicha capa, se define como:

$$L_{lat} = \mathbb{E}_{x \sim p_{data}} \| f(\boldsymbol{x}) - f(G(\boldsymbol{x})) \|_2 \tag{3.10}$$

Así, la función de coste se define como:

$$L_G = \lambda_{con} L_{con} + \lambda_{lat} L_{lat} \tag{3.11}$$

donde se ha establecido que $\lambda_{con} = 50$ y $\lambda_{lat} = 1$

Concretamente, la configuración en cuanto a número capas y filtros, funciones de activación... se muestra en la figura 3.6.

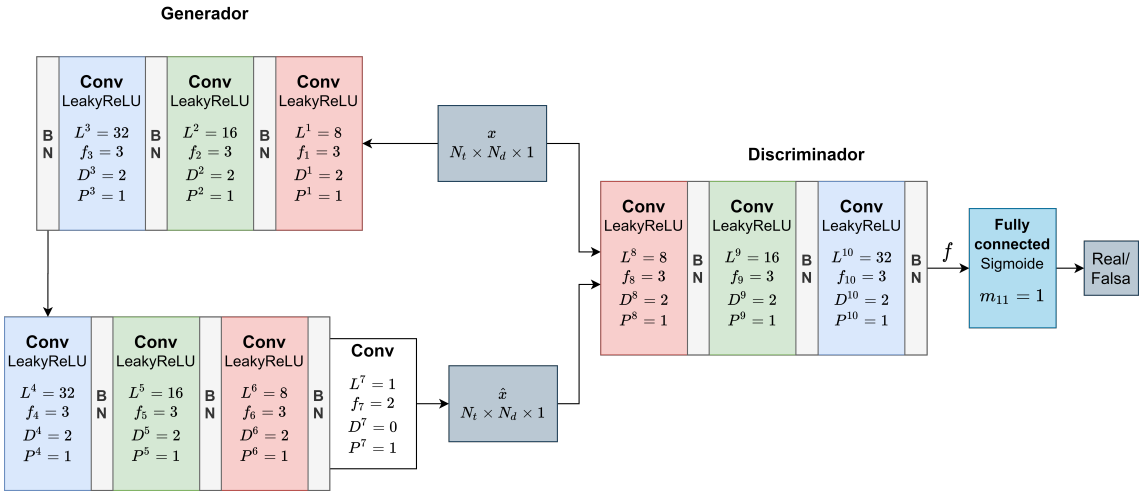


Fig. 3.6. Estructura de la red de la propuesta 4 - GAN

De nuevo, como se ha entrenado el generador para que las imágenes a entrada y salida sean lo más similares posible, aquí se emplea el error cuadrático instantáneo como marcador de anomalías. La hipótesis es que, al añadir la función de coste 3.10, esta reconstrucción será mejor que en el caso de la propuesta 1 y, por lo tanto, el sistema será capaz de detectar las anomalías de forma más diferenciada.

3.3. Estimación de parámetros estadísticos de los marcadores de anomalías

En este punto del desarrollo, se conoce cómo se definen y obtienen los marcadores de anomalías en los distintos sistemas. En esta sección se aborda el problema de estimar las distribuciones de probabilidad que siguen los distintos marcadores de anomalías en situaciones de normalidad. Este último paso permite construir un sistema cuya salida es la probabilidad de que el marcador de anomalías esté en rangos típicos, para que, cuando esta probabilidad sea baja, se considere que hay una anomalía. Para construirlo, se cuenta con capturas sin anomalías que sirven de entrada a la red en fase de inferencia una vez ésta ya ha sido entrenada. De esta forma, se tienen la entrada y salida de la red en casos de normalidad, por lo que es posible calcular los marcadores de anomalías en situaciones de normalidad e incidir la distribución que siguen.

Para justificar esto, se puede utilizar el resultado de 2.42. En este caso, se considera que H_1 y H_2 son las hipótesis que asumen normalidad y anomalía, respectivamente. Por otro lado, B se define como: "el marcador de anomalías toma un determinado valor". De esta forma se considerará que hay una anomalía si y sólo si:

$$P(B|H_1) \leq \frac{\lambda_{12} - \lambda_{22}}{\lambda_{21} - \lambda_{11}} \frac{P(H_2)}{P(H_1)} P(B|H_2) = \gamma \tag{3.12}$$

Por lo tanto, aquí se pretende calcular el término $P(B|H_1)$ esto es, la probabilidad de que un marcador de anomalías tome determinado valor en caso de normalidad. Para esto, se debe inferir la distribución de probabilidad que toman los marcadores de anomalías en el caso de normalidad.

Es común en estos casos tomar un único parámetro que sirva como umbral para decidir si hay o no una anomalía. El primer motivo para esto es que λ_{11}, λ_{12}, λ_{21} y λ_{22} son parámetros de diseño que permiten decidir el grado de penalización por fallos según si éstos vienen de falsos negativos o falsos positivos, respectivamente. Por otro, los valores de $P(H_1)$ y $P(H_2)$ son completamente dependientes de la aplicación y cambian según la situación o con el tiempo. Por último, el hecho de que haya pocas anomalías y puedan tener distintos orígenes, impide estimar la distribución de probabilidad de $B|H_2$. Como γ es un parámetro de diseño, para valorar el rendimiento del sistema, lo ideal es recurrir al concepto de AUC, que, intuitivamente, evalúa el sistema en todo el rango de posibles valores que γ puede tomar, es decir, $[0, 1]$.

Por otro lado, debido a que el comportamiento de la luz en la fibra varía con la distancia, se observa que la distribución que siguen estos marcadores también se modifica a lo largo de la misma. Por este motivo, resulta conveniente realizar una estimación diferente según las distancias. Aprovechando que se tiene un número de W_t de parches que corresponden a un mismo conjunto de distancias, la idea es modelar la distribución de probabilidad que siguen los elementos de los marcadores de anomalías asociados a parches correspondientes a un mismo conjunto de distancias, lo cual se observa en la figura

3.7. Así, la solución consiste en tomar todos los elementos de los W_t parches de $N_d \times N_t$ y estimar la distribución de probabilidad que siguen.

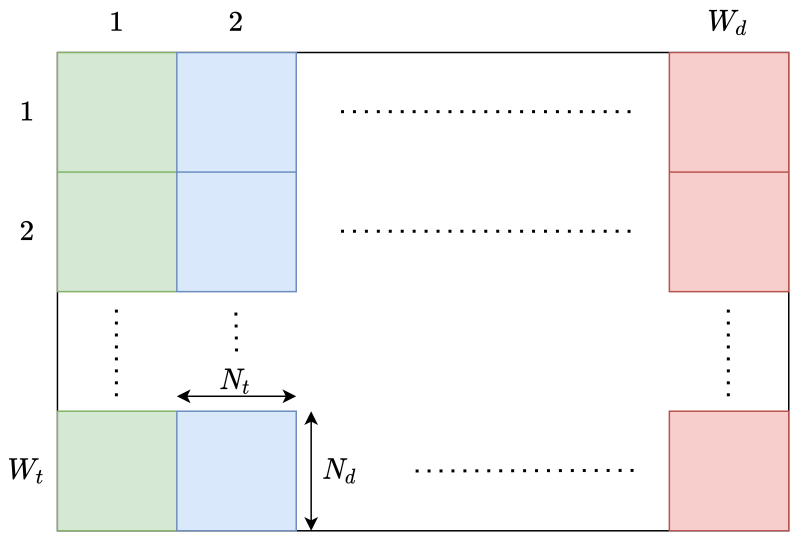


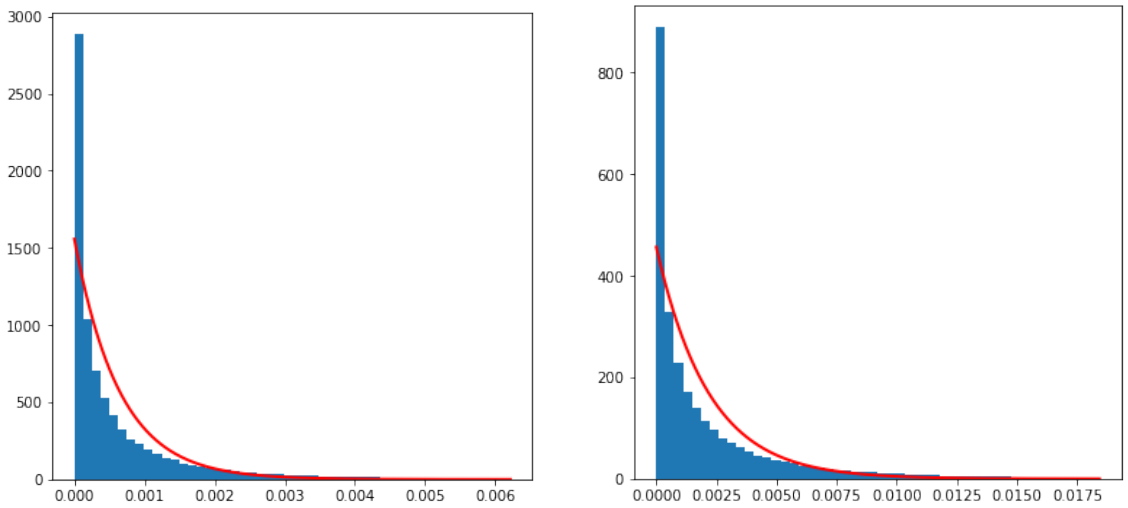
Fig. 3.7. Existen un número W_t de parches correspondientes a un mismo conjunto de distancias

En los distintos sistemas se proponen dos marcadores de anomalías, por lo que a continuación se analizan las distribuciones de probabilidad que se utilizan para modelar cada uno de ellos.

Error cuadrático instantáneo

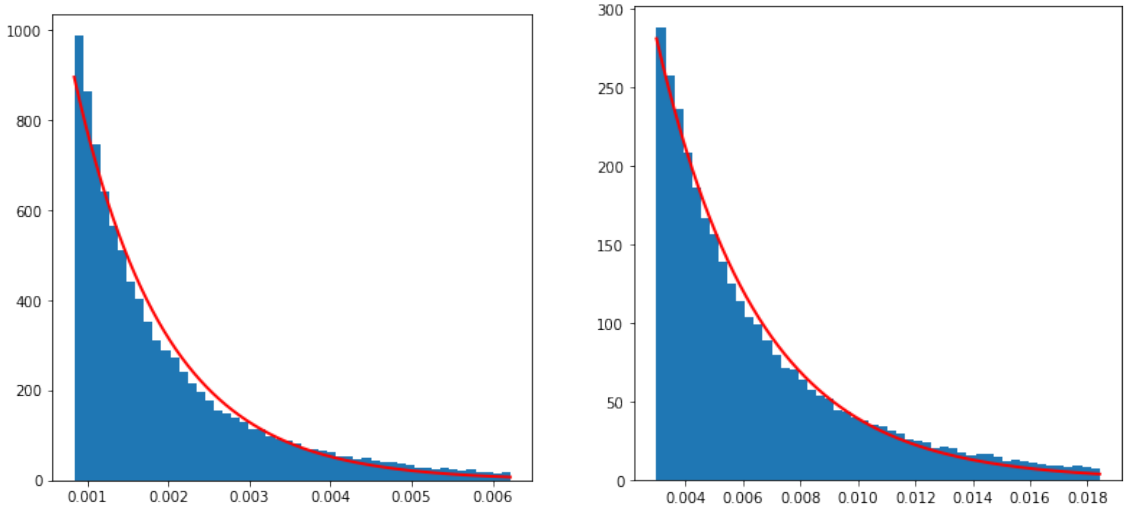
Este marcador es empleado en los cuatro sistemas y en todos se modela de la misma forma debido a que su comportamiento es similar. Para explicar esto, conviene visualizar histogramas de este marcador de anomalías para datos de normalidad según distancias. En la figura 3.8 se muestran histogramas correspondientes a dos conjuntos de distancias. Para empezar, con esta representación se puede apreciar la necesidad de modelar los marcadores de anomalías según conjuntos de distancias, ya que varían considerablemente según la posición de la fibra. Por otro lado, parece que una distribución exponencial puede modelar de forma acertada los datos, lo cual sería especialmente interesante debido a la sencillez de esta distribución. Así, la línea roja en ambas gráficas representa la función de distribución de probabilidad exponencial estimada para cada conjunto de datos mediante una estimación de máxima verosimilitud [56]. Esta función parece no adaptarse del todo bien a los datos. Sin embargo, cabe darse cuenta de que no es realmente necesario modelar los valores más bajos de los datos, ya que el valor que toma este marcador cuando los datos son anómalos es elevado. Por este motivo, se propone modelar sólo los datos superiores a un determinado percentil, el cual puede ser un parámetro de diseño. En la figura 3.9 se muestra el histograma de los datos de error cuadrático instantáneo superior al percentil 75 correspondientes a las mismas distancias y se puede apreciar que, en este caso, la distribución exponencial si se adapta mucho mejor a los datos. Tras realizar alguna prueba, se

ha decidido tomar el percentil 90 para todos los casos.



(a) 512 - 563.2 km (b) 30720 - 30771.2 km

Fig. 3.8. Histogramas de error cuadrático instantáneo según distancia



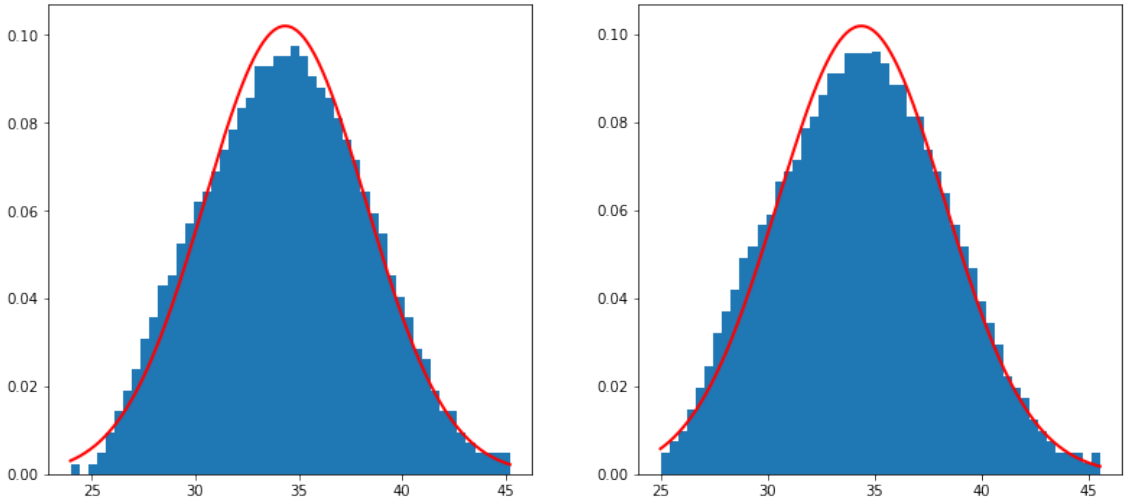
(a) 512 - 563.2 km (b) 30720 - 30771.2

Fig. 3.9. Histogramas de error cuadrático instantáneo según distancia para datos superiores al percentil 75

Divergencia KL

En la propuesta 3, se propone el VAE como medio para conseguir dos marcadores de anomalías. Uno de ellos es el error cuadrático instantáneo, cuyo modelado se ha explicado ya. El otro es la divergencia KL de cada parche. Para este marcador de anomalías, se ha seguido la misma metodología y se propone modelar la divergencia KL por zonas espaciales de la fibra. En la figura 3.10 se muestra un histograma de la divergencia KL en distintas zonas de la fibra. En este caso, una distribución normal ajusta los datos y,

como se aprecia en la figura, el comportamiento es muy similar en ambos tramos por lo que aquí no parece necesario el hecho de modelar de distinta forma. Sin embargo, también se propone estimar distintos parámetros para distintas distancias, ya que puede haber situaciones en la que sí sea conveniente.



(a) 512 - 563.2 km (b) 30720 - 30771.2 km

Fig. 3.10. Histogramas de error cuadrático instantáneo según distancia

3.4. Resumen conceptual de la forma de trabajo

Aunque se han presentado cuatro propuestas, el objetivo es el mismo en todas ellas y sólo cambia la forma de alcanzarlo. Por ese motivo, se introduce esta sección, que pretende resumir y clarificar las tres fases principales de trabajo del sistema:

Entrenamiento de los sistemas de Deep Learning. La primera fase en cualquiera de las propuestas consiste en tomar datos ausentes de anomalías procedentes del DAS, preprocesarlos y entrenar la red neuronal con ellos. La idea detrás de esto es que, cuando la red se entrene con datos libres de anomalías, la reconstrucción e inferencia de datos normales sea exitosa y que la de datos anómalos fracase. A partir de estas inferencias, se definen unos marcadores de anomalías, que tomarán distintos valores según si los datos son normales o anómalos.

Modelado probabilístico de los marcadores de anomalías en datos normales. Una vez la red ha sido entrenada, se toman de nuevo datos libres de anomalías, se preprocesan y se pasan como entrada a la red para que realice las predicciones. Una vez se han realizado dichas predicciones, es posible calcular los marcadores de anomalías. Por lo tanto, con una cantidad suficientemente grande de datos, se pueden estimar los parámetros de una

distribución de probabilidad que modele de forma acertada los marcadores de anomalías cuando la entrada al sistema son datos normales.

Fase de trabajo. Una vez la red ha sido entrenada y se han estimado los parámetros que modelan la distribución que siguen los marcadores de anomalías cuando las entradas son normales, se está en disposición de comenzar el trabajo. Para ello, se toman datos directamente del DAS sin información a priori sobre si son normales o anómalos y se preprocesan. Después se pasan como entrada a la red neuronal para que realice la tarea de inferencia. Una vez esta tarea se ha realizado, se calculan los marcadores de anomalías y se estima la probabilidad de que un marcador de anomalías tome ese valor bajo la hipótesis de normalidad según los parámetros estimados en la fase anterior. Por lo tanto, la salida del sistema es una probabilidad, la cual puede ser utilizada para establecer un umbral a partir del cual se considere que la entrada es anómala.

4. MARCO EXPERIMENTAL Y RESULTADOS

Tras explicar la metodología de trabajo que permite detectar anomalías desde señales procedentes de los DAS, en este capítulo se presentan el escenario en el que se han capturado las señales utilizadas para este trabajo, las distintas pruebas realizadas y los resultados obtenidos en éstas.

4.1. Captura de la información

Aunque la forma en que la información es capturada por el receptor acústico distribuido no es realmente parte de este trabajo, sí que conviene presentar el escenario en el que se tomaron las medidas con las que se ha trabajado, ya que esto permite familiarizarse con las posiciones en las que potencialmente están los eventos. En la figura 4.1 se incluye un esquema que permite entender bien el escenario en el que se tomaron las medidas.

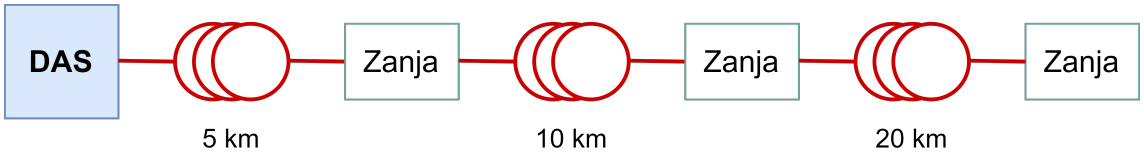


Fig. 4.1. Esquema del escenario de toma de medidas

La longitud de fibra con las que se ha trabajado es de 35.2 km, pero, realmente, su configuración es de bobinas de 5, 10 y 20 kilómetros enrolladas y una zanja, que es donde se han producido los eventos. En la figura 4.2 se puede ver el escenario donde se realizaron las pruebas.

Hay que adelantar aquí un aspecto y es que tal y como se muestra en la figura 4.1, el escenario donde se tomaron las medidas pretende simular uno final en el que hay una fibra de 35.2 km desplegada en el suelo. Este hecho genera un pequeño problema y es que, según se ha podido comprobar, el comportamiento de la luz dentro de la fibra no es el mismo en los carretes que en la zona de las zanjas. Por lo tanto, a la hora de que las redes generalicen su aprendizaje y teniendo en cuenta que la mayoría de las medidas con las que se ha trabajado son de zonas pertenecientes fibra en carretes, las zonas de zanja, simplemente por el hecho de estar en una zanja generarán un mayor error de predicción aún en casos en los que no hay eventos.

La señal que entrega el sensor acústico tiene una resolución espacial de 6.4 metros y una resolución temporal de un milisegundo o, equivalentemente, una frecuencia de muestreo de 1kHz. Por lo tanto, para una captura de un minuto, $L_t = 60000$ y $L_d = 5500$.

Fig. 4.2. Escenario físico donde se tomaron las medidas

Al estar todas las propuestas basadas en sistemas de Deep Learning, es necesario contar con información para entrenarlos. Para ello, se ha contado con los datos correspondientes a capturas de diez minutos sin anomalías a lo largo de los 35.2 km útiles de la fibra. Debido a que el comportamiento de la luz en la fibra se modifica con la distancia, esta información se ha clasificado por conjuntos de distancias y para cada una de las propuestas que a continuación se dan, se ha entrenado una misma estructura con información que corresponde a distintas distancias. La idea es que para cada propuesta que se da, haya un conjunto de sistemas que tengan la misma estructura, pero sus parámetros sean diferentes según en qué puntos de la fibra se especifica su trabajo. En concreto se ha decidido que:

- **Conjunto 1:** 0 - 16.0 km
- **Conjunto 2:** 16.0 - 25.6 km
- **Conjunto 3:** 25.6 - 32 km
- **Conjunto 4:** 32.0- 35.2 km

Aunque en esta ocasión se han dividido los datos en cuatro grupos únicamente debido a que el conjunto de datos para entrenar las redes es reducido, el número de grupos puede depender del escenario y de la cantidad de datos disponibles.

Una vez las redes han sido entrenadas, para estimar la distribución de probabilidad que siguen los marcadores de anomalías cuando no hay eventos, se ha contado con una captura de otros cinco minutos de datos en situación de normalidad.

Por otro lado, en la tabla 4.1 aparece información sobre las señales con eventos que se han utilizado para evaluar el sistema.

Etiqueta medida	Comentario	Duración total	Tipo de evento	Duración (s)
Entra	Oruga avanza	300	Avanza	106
			Silencio	194
Excavadora 0m	Distancia = 0 m	240	Silencio	60
			Excava	39
			Silencio	40
			Excava	50
			Silencio	51
Excavadora 10m	Distancia = 10 m	240	Silencio	60
			Excava	43
			Silencio	46
			Excava	45
			Silencio	46
Martillo Hidráulico	Iteración 1	120	Silencio	42
			Pico 0 m	11
			Silencio	29
			Pico 5 m	11
			Silencio	18
			Pico 10 m	9

TABLA 4.1. FICHAS DE LAS MEDIDAS CON EVENTOS UTILIZADAS PARA DESARROLLAR Y MEDIR EL RENDIMIENTO DE ESTE TRABAJO

4.2. Explicación de las pruebas realizadas

El banco de pruebas podría ser extenso debido al número de parámetros de diseño presentados, a la existencia de cuatro sistemas implementados o a posibles variaciones de las estructuras de aprendizaje profundo. Sin embargo, se han desarrollado dos tipos de comparaciones elementales que a continuación se explican.

Comparación 1: Influencia de los parámetros de preprocesado en el autoencoder

En esta prueba se analiza cómo los parámetros de preprocesado D_t, N_t, M_t, N_d y M_d influyen cuando el procesado se lleva a cabo a través de la propuesta 1. El objetivo fundamental de esta prueba es analizar cómo varía el rendimiento según los parámetros de preprocesado en la propuesta en la que se basan todas las demás. La hipótesis es que

los resultados que se obtienen para el sistema más elemental son extrapolables a todos los demás. Así, los posibles valores que toman los parámetros son los siguientes:

- $D_t \in \{10, 40, 100\}$
- $N_t \in \{32, 64, 128, 256\}$
- $M_t \in \{0{,}75 \cdot N_t, N_t\}$
- $N_d \in \{8, 16, 32\}$
- $M_d \in \{0{,}75 \cdot N_d, N_d\}$

Estas pruebas se realizan sobre la señal correspondiente a la oruga excavando directamente sobre el terreno bajo el que está enterrada la fibra óptica.

Comparación 2: Rendimiento de las propuestas bajo una misma elección de parámetros de diseño

Tras analizar la influencia de los parámetros de diseño del preprocesado en la propuesta 1, aquí se fijan unos valores para dichos parámetros y se comparan los resultados obtenidos en las cuatro propuestas distintas. En concreto, los valores que se fijan son $D_t = 10$, $N_t = 128$, $M_t = 128$, $N_d = 8$ y $M_d = 6$.

En este caso, se utilizan todas las capturas de la tabla 4.1 para analizar cómo influye cada tipo de medida en cada sistema. Por otro lado y debido a que es esperable que el rendimiento de estos sistemas empeoren de forma proporcional a la distancia a la que se encuentran del sensor, se dividen los resultados según tramos de la fibra. En concreto, se divide en tres trozos: de 0 a 12.8 km, de 12.8 a 25.6 km y de 25.6 a 35.2 km.

4.3. Resultados

Como medida para analizar objetivamente los resultados de las dos comparaciones previamente descritas, se propone utilizar el AUC. La ventaja de esta medida es que permite valorar el rendimiento de los sistemas independientemente del umbral de decisión.

4.3.1. Comparación 1

A continuación se incluyen tres tablas que dan los resultados de esta comparación. Estas tablas reflejan los valores de AUC obtenidos para los factores de diezmado $D_t = 10$, $D_t = 40$ y $D_t = 100$ y cada una de ellas muestra la variación de los resultados según el resto de los parámetros.

De lo anterior, se pueden extraer las siguientes conclusiones:

		$N_d = 8$		$N_d = 16$		$N_d = 32$	
		$M_d = 6$	$M_d = 8$	$M_d = 12$	$M_d = 16$	$M_d = 24$	$M_d = 32$
$N_t = 32$	$M_t = 24$	0.9658	0.9638	0.9537	0.9549	0.9449	0.9402
	$M_t = 32$	0.9684	0.9626	0.9502	0.9558	0.9433	0.9361
$N_t = 64$	$M_t = 48$	0.9712	0.9650	0.9469	0.9555	0.9473	0.9370
	$M_t = 64$	0.9713	0.9658	0.9498	0.9565	0.9463	0.9348
$N_t = 128$	$M_t = 96$	0.9686	0.9657	0.9462	0.9603	0.9451	0.9338
	$M_t = 128$	0.9716	0.9661	0.9464	0.9588	0.9394	0.9312
$N_t = 256$	$M_t = 192$	0.9695	0.9653	0.9469	0.9616	0.9421	0.9338
	$M_t = 256$	0.9729	0.9638	0.9478	0.9585	0.9449	0.9328

TABLA 4.2. COMPARACIÓN 1: $D_T = 10$

		$N_d = 8$		$N_d = 16$		$N_d = 32$	
		$M_d = 6$	$M_d = 8$	$M_d = 12$	$M_d = 16$	$M_d = 24$	$M_d = 32$
$N_t = 32$	$M_t = 24$	0.9703	0.9645	0.9452	0.9530	0.8532	0.8600
	$M_t = 32$	0.9725	0.9671	0.9506	0.9526	0.8493	0.8489
$N_t = 64$	$M_t = 48$	0.9740	0.9675	0.9457	0.9589	0.8423	0.8522
	$M_t = 64$	0.9753	0.9693	0.9401	0.9380	0.8391	0.8447
$N_t = 128$	$M_t = 96$	0.9748	0.9719	0.9394	0.9467	0.8280	0.8318
	$M_t = 128$	0.9755	0.9696	0.9485	0.9430	0.8387	0.8388
$N_t = 256$	$M_t = 192$	0.9789	0.9742	0.9482	0.9588	0.8544	0.8557
	$M_t = 256$	0.9786	0.9720	0.9402	0.9523	0.8507	0.8441

TABLA 4.3. COMPARACIÓN 1: $D_T = 40$

		$N_d = 8$		$N_d = 16$		$N_d = 32$	
		$M_d = 6$	$M_d = 8$	$M_d = 12$	$M_d = 16$	$M_d = 24$	$M_d = 32$
$N_t = 32$	$M_t = 24$	0.9686	0.9653	0.9388	0.9384	0.8362	0.8367
	$M_t = 32$	0.9682	0.9664	0.9391	0.9303	0.8235	0.8169
$N_t = 64$	$M_t = 48$	0.9748	0.9687	0.9313	0.9301	0.8047	0.7417
	$M_t = 64$	0.9708	0.9656	0.9291	0.9261	0.7561	0.7751
$N_t = 128$	$M_t = 96$	0.9772	0.9648	0.9253	0.9218	0.7601	0.7282
	$M_t = 128$	0.9677	0.9680	0.9247	0.9200	0.7503	0.7517
$N_t = 256$	$M_t = 192$	0.9780	0.9797	0.9505	0.9384	0.8325	0.8053
	$M_t = 256$	0.9737	0.9771	0.9419	0.9277	0.7777	0.7953

TABLA 4.4. COMPARACIÓN 1: $D_T = 100$

- Cuanto menor es D_t, menos afectan el resto de los parámetros al rendimiento.
- El valor de N_t parece no tener influencia en el rendimiento del sistema.
- Dado un N_t fijo, el valor de M_t apenas tiene influencia.
- En general, dado un N_d fijo, el rendimiento es mejor cuando M_d vale un 75 % de N_d que cuando vale un 100 %.
- Cuanto menor es N_d, mejor es el rendimiento.

4.3.2. Comparación 2

En esta comparación se muestran los valores de AUC obtenidos según las diferentes propuestas de este trabajo para unos parámetros de diseño fijos.

	0.0 - 12.8 km	12.8 - 25.6 km	25.6 - 35.2 km
Autoencoder	0.99956	0.99955	0.99090
Autoencoder-LSTM	0.99951	0.99972	0.98238
VAE - Error cuadrático	0.99941	0.99929	0.98660
VAE - Divergencia KL	0.98542	0.98460	0.94264
GAN	0.99961	0.99958	0.99084

TABLA 4.5. COMPARACIÓN 2: HIDRÁULICO 1

	0.0 - 12.8 km	12.8 - 25.6 km	25.6 - 35.2 km
Autoencoder	0.97944	0.97702	0.77745
Autoencoder-LSTM	0.97195	0.96253	0.81238
VAE - Error cuadrático	0.97633	0.97597	0.83115
VAE - Divergencia KL	0.90036	0.84349	0.77754
GAN	0.97831	0.97652	0.75772

TABLA 4.6. COMPARACIÓN 2: ENTRA

	0.0 - 12.8 km	12.8 - 25.6 km	25.6 - 35.2 km
Autoencoder	0.99774	0.98790	0.92927
Autoencoder-LSTM	0.99779	0.98639	0.94560
VAE - Error cuadrático	0.99837	0.98731	0.93946
VAE - Divergencia KL	0.94233	0.92294	0.86486
GAN	0.99771	0.98767	0.92230

TABLA 4.7. COMPARACIÓN 2: ORUGA 0 M

De esta comparación se pueden extraer las siguientes conclusiones:

	0.0 - 12.8 km	12.8 - 25.6 km	25.6 - 35.2 km
Autoencoder	0.99515	0.98710	0.88433
Autoencoder-LSTM	0.99621	0.98460	0.90141
VAE - Error cuadrático	0.99255	0.98159	0.87884
VAE - Divergencia KL	0.97873	0.91628	0.91960
GAN	0.99485	0.98746	0.89043

TABLA 4.8. COMPARACIÓN 2: ORUGA 10 M

- La utilización de capas LSTM convolucionales mejora el rendimiento con respecto a su no utilización en distancias más altas y para eventos más difícilmente detectables.
- El rendimiento del marcador de anomalía basado en el error cuadrático es similar para el autoencoder y para el VAE.
- El marcador de anomalías del VAE basado en la divergencia KL da unos resultados peores que el resto. Por este motivo, será conveniente utilizar un umbral alto de forma que apenas se pierdan anomalías a cambio de decodificar un mayor número de parches.
- La propuesta basada en las GAN no mejora el rendimiento con respecto al autoencoder en las pruebas realizadas. Por lo tanto, se desaconseja su uso debido a que, aunque la fase de inferencia tiene la misma complejidad comutacional, su entrenamiento es más costoso.

4.3.3. Mapa tiempo-distancia de una señal

Para ilustrar el proceso de forma más clara y comprender la mejora que la introducción de estos sistemas supone, a continuación se introducen dos figuras. La 4.3 muestra la señal proporcionada por el DAS y la 4.4 la salida del error cuadrático de una de estas propuestas. Como se puede apreciar, la mejora es evidente, ya que en la señal proporcionada por el DAS ni siquiera es posible distinguir los eventos.

Por otro lado, este es buen momento para explicar un problema surgido debido al escenario y que puede haber afectado al entrenamiento de las redes. Como se comenta en la sección 4.1, en el escenario de captura de datos, las zonas de zanja, aún en casos en los que no hay eventos, tienen un error de predicción mayor, puesto que el comportamiento de la luz de la fibra no es el mismo en los carretes que en la zanja. En la figura 4.5 se muestra un mapa tiempo-distancia para una medida de dos minutos en la que no se produce ningún evento. Como se puede ver, las partes correspondientes a la zanja tienen un error cuadrático superior. En este punto, se podría haber considerado dedicar redes a entrenarse exclusivamente en las zonas de zanjas, pero se ha contado con muy pocos datos ausentes de anomalías para entrenar las redes, por lo que esto no es realizable en esta

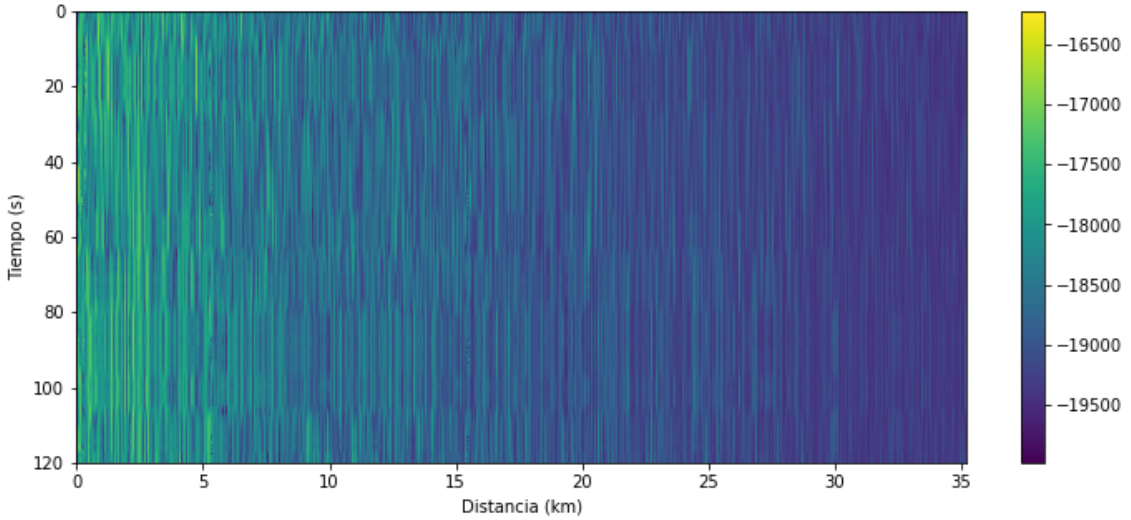


Fig. 4.3. Señal de martillo hidráulico procedente del DAS sin ningún tratamiento

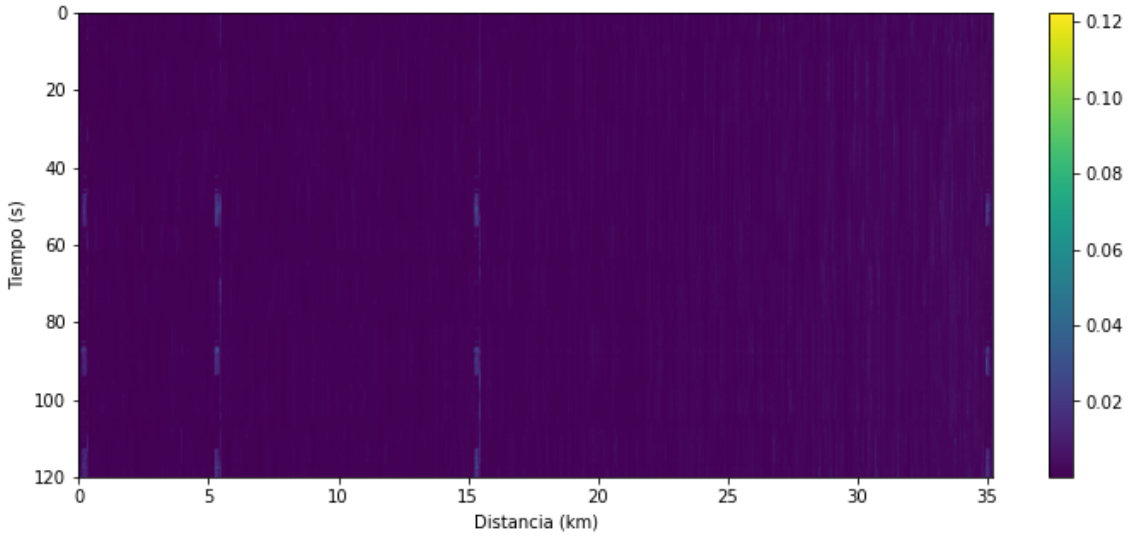


Fig. 4.4. Señal de martillo hidráulico tras el tratamiento

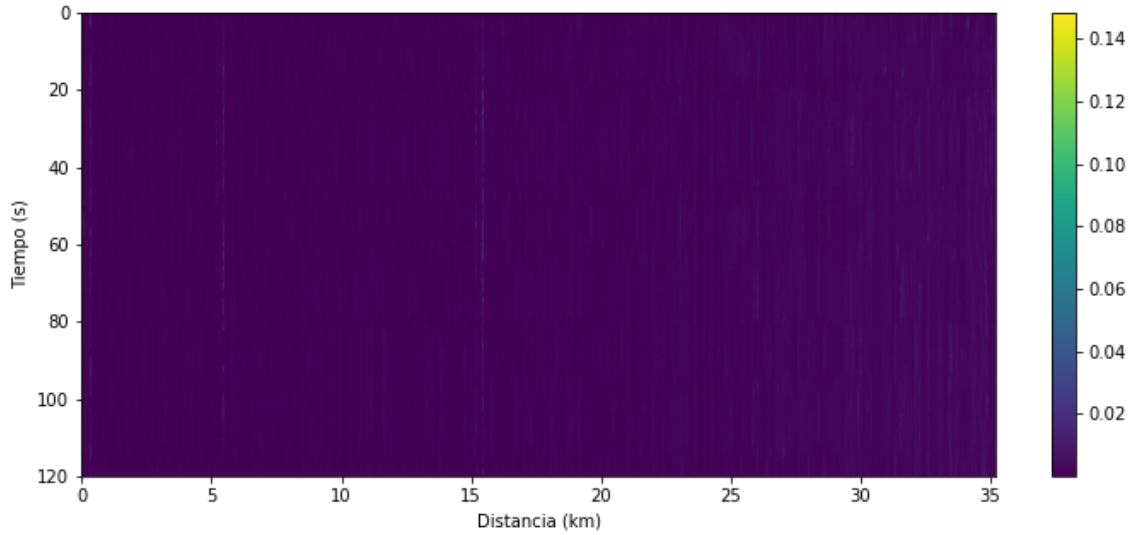


Fig. 4.5. Señal sin eventos anómalos tras el tratamiento

situación. Esto no ocurriría en un escenario en el que toda la fibra estuviera desplegada de forma homogénea.

5. CONCLUSIONES Y LÍNEAS FUTURAS

En este capítulo se recogen las conclusiones extraídas de este trabajo y se analiza el cumplimiento de los objetivos planteados inicialmente. También se comentan algunas de las ideas que no se han llegado a implementar por falta de tiempo y de un banco de medidas de mayor tamaño.

5.1. Conclusiones

En este trabajo se han planteado cuatro alternativas basadas en técnicas de aprendizaje profundo para detectar anomalías en señales bidimensionales procedentes de sensores DAS. La primera es un autoencoder con capas convolucionales, la segunda incluye a la anterior capas LSTM, la tercera utiliza la misma estructura que la primera introduciendo términos probablísticos con un VAE y la última propone utilizar el concepto de GAN para resolver el problema. La primera fue planteada en [3] y sirve como base para el desarrollo de las otras tres, que permiten mejorar el rendimiento del sistema, bien sea aumentando el AUC en determinadas circunstancias, o disminuyendo la cantidad de datos a procesar.

Por otro lado, se han estimado las distribuciones que siguen los distintos marcadores de anomalías, lo cual permite que la salida del sistema completo sea realmente la probabilidad de que un dato no sea anómalo. De esta forma, es sencillo establecer un umbral a partir del cual considerar que se ha producido un evento y que sea dependiente de la aplicación y situación.

Otro aspecto fundamental, que se consigue en parte gracias a haber empleado técnicas aprendizaje profundo, es el hecho de que el sistema descrito sería funcional en otros entornos. En este sentido, puede decirse que todas las propuestas descritas son flexibles y la idea de modelar de diferente forma las distintas distancias de la fibra permite ajustarse a casi cualquier entorno.

En cuanto al cumplimiento de los objetivos, el primero requería el aprendizaje de nuevas técnicas de Deep Learning, tales como capas recurrentes, VAE o GAN. Esto no es algo objetivamente mensurable, pero se han desarrollado sistemas basados en estas tecnologías, por lo que se puede considerar que los resultados son satisfactorios en este aspecto.

Los tres objetivos siguientes eran precisamente dar alternativas basadas en LSTM, VAE y GAN al sistema previamente planteado, lo cual se considera cumplido, ya que, no sólo se han implementado estas alternativas, sino que además mejoran el rendimiento o reducen el tiempo de procesado, justificando así la necesidad de añadir dificultad conceptual al diseño inicial.

Por último, se ha realizado un análisis de cómo afectan distintos parámetros de diseño al rendimiento. También se han comparado las distintas propuestas para analizar si su inclusión está justificada. Estos análisis son importantes de cara a optimizar el funcionamiento y los resultados del sistema de detección de anomalías.

La importancia de este trabajo reside en que se han dado cuatro soluciones funcionales, flexibles y robustas que son capaces de llevar a cabo una tarea de realce en las señales procedentes de los DAS que, a primera vista y sin ningún tipo de tratamiento, son ruidosas y parecen no proporcionar información.

5.2. Líneas futuras

El objetivo de esta sección no es tratar de dar ideas de aplicaciones del sistema descrito, ya que es algo que no es propio de este trabajo, sino que más bien es listar algunas ideas que por falta de tiempo o por estar fuera del objetivo inicial del trabajo no se han llegado a desarrollar.

- **Estudiar cómo afectan diferentes configuraciones de capas al rendimiento.** En este trabajo se han tomado unas estructuras en cuanto a número de capas, filtros de cada capa convolucional, tamaño del espacio latente... Sin embargo, aunque se ha probado que su funcionamiento es correcto, puede no ser el mejor. Por tanto, un trabajo pendiente es el de comparar formalmente y con medidas de rendimiento como el AUC distintas configuraciones.

- **Clasificación de eventos.** El desarrollo de este trabajo, debido al escenario y cantidad de información disponible, está enfocado en detectar anomalías de forma no supervisada. Sin embargo, para entornos en los que claramente haya un conjunto fijo de posibles tipos de eventos, el siguiente paso sería clasificar dichas anomalías.

- **Nuevos sistemas basados en GANs.** Estas redes están ganando popularidad estos últimos años y hay variedad de propuestas y nuevas posibilidades para afrontar el problema de detección de anomalías en estructuras basadas en ellas. Aunque aquí se ha propuesto una idea basada en estas redes, este es un campo más amplio y en el que conviene seguir trabajando.

- **Combinación entre propuestas.** La mejora más directa que se podría llevar a cabo es la introducción de capas LSTM en las propuestas 3 y 4. También, en la propuesta 4, el generador podría ser un VAE en lugar de un autoencoder estocástico, lo cual podría combinar las ventajas de ambos conceptos.

BIBLIOGRAFÍA

[1] J. Park y H. F. Taylor, "Fiber Optic Intrusion Sensor using Coherent Optical Time Domain Reflectometer," *Japanese Journal of Applied Physics*, vol. 42, n.º Part 1, No. 6A, pp. 3481-3482, 2003. DOI: 10.1143/jjap.42.3481. [En línea]. Disponible en: https://doi.org/10.1143%2Fjjap.42.3481.

[2] "¿Qué pasa en las vías del tren? Pregúntale a la fibra óptica," *Heraldo de Aragón*, 2017. [En línea]. Disponible en: https://www.heraldo.es/noticias/sociedad/2017/10/30/que-pasa-las-vias-del-tren-preguntale-fibra-optica-1204293-310.html.

[3] Antonio Almudévar, "Trabajo de fin de grado: Deep Generative Models para sensores acústicos distribuidos," Universidad de Zaragoza, 2020.

[4] A. Krizhevsky, I. Sutskever y G. E. Hinton, "Imagenet classification with deep convolutional neural networks," en *Advances in neural information processing systems*, 2012, pp. 1097-1105.

[5] D. E. Rumelhart, G. E. Hinton y R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, n.º 6088, pp. 533-536, 1986.

[6] Wikipedia contributors, *Autoencoder — Wikipedia, The Free Encyclopedia*, [Online; accessed 27-June-2021], 2021. [En línea]. Disponible en: https://en.wikipedia.org/w/index.php?title=Autoencoder&oldid=1029582652.

[7] D. P. Kingma y M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[8] I. J. Goodfellow et al., "Generative adversarial networks," *arXiv preprint arXiv:1406.2661*, 2014.

[9] Wikipedia contributors, *Receiver operating characteristic — Wikipedia, The Free Encyclopedia*, [Online; accessed 27-June-2021], 2021. [En línea]. Disponible en: https://en.wikipedia.org/w/index.php?title=Receiver_operating_characteristic&oldid=1029293991.

[10] A. Owen, G. Duckworth y J. Worsley, "OptaSense: Fibre optic distributed acoustic sensing for border monitoring," en *2012 European Intelligence and Security Informatics Conference*, IEEE, 2012, pp. 362-364.

[11] T. M. Daley et al., "Field testing of fiber-optic distributed acoustic sensing (DAS) for subsurface seismic monitoring," *The Leading Edge*, vol. 32, n.º 6, pp. 699-706, 2013.

[12] T. Parker, S. Shatalin y M. Farhadiroushan, "Distributed Acoustic Sensing–a new tool for seismic applications," *first break*, vol. 32, n.º 2, pp. 61-69, 2014.

[13] Wikipedia, *Fibra oscura — Wikipedia, La enciclopedia libre*, [Internet; descargado 24-junio-2020], 2019. [En línea]. Disponible en: https://es.wikipedia.org/w/index.php?title=Fibra_oscura&oldid=117345730.

[14] A. T. Young, "Rayleigh scattering," *Applied optics*, vol. 20, n.º 4, pp. 533-535, 1981.

[15] J. Mateo, M. n. Losada e I. Garcés, "Limitaciones de las fibras ópticas," en *Dispositivos y Sistemas de Transmisión Óptica*, 2017, cap. 2, pp. 40-70.

[16] T. Horiguchi y M. Tokuda, "Optical time domain reflectometer for single-mode fibers," *IEICE TRANSACTIONS (1976-1990)*, vol. 67, n.º 9, pp. 509-515, 1984.

[17] Sanjay, *Difference between OTDR and COTDR*. 2016. [En línea]. Disponible en: https://mapyourtech.com/entries/general/difference-between-otdr-and-cotdr-.

[18] J. P. Garbayo, D. Sanahuja, C. Heras, J. Subías y Í. Salinas, "Desarrollo y caracterización de un sensor acústico distribuido basado en la técnica de medida C-OTDR," *Jornada de Jóvenes Investigadores del I3A*, vol. 6, 2018.

[19] W. S. McCulloch y W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, vol. 5, n.º 4, pp. 115-133, 1943.

[20] B. L. Kalman y S. C. Kwasny, "Why tanh: choosing a sigmoidal function," en *[Proceedings 1992] IJCNN International Joint Conference on Neural Networks*, IEEE, vol. 4, 1992, pp. 578-581.

[21] V. Nair y G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," en *Icml*, 2010.

[22] B. Xu, N. Wang, T. Chen y M. Li, "Empirical evaluation of rectified activations in convolutional network," *arXiv preprint arXiv:1505.00853*, 2015.

[23] C. E. Shannon, "A mathematical theory of communication," *Bell system technical journal*, vol. 27, n.º 3, pp. 379-423, 1948.

[24] S. Kullback y R. A. Leibler, "On information and sufficiency," *The annals of mathematical statistics*, vol. 22, n.º 1, pp. 79-86, 1951.

[25] A. Cauchy, "Méthode générale pour la résolution des systemes d'équations simultanées," *Comp. Rend. Sci. Paris*, vol. 25, n.º 1847, pp. 536-538, 1847.

[26] D. P. Kingma y J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[27] J. Sola y J. Sevilla, "Importance of input data normalization for the application of neural networks to complex industrial problems," *IEEE Transactions on nuclear science*, vol. 44, n.º 3, pp. 1464-1468, 1997.

[28] M. Shanker, M. Y. Hu y M. S. Hung, "Effect of data standardization on neural network training," *Omega*, vol. 24, n.º 4, pp. 385-397, 1996.

[29] Y. LeCun, L. Bottou, Y. Bengio y P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, n.º 11, pp. 2278-2324, 1998.

[30] R. Pascanu, T. Mikolov e Y. Bengio, "On the difficulty of training recurrent neural networks," en *International conference on machine learning*, PMLR, 2013, pp. 1310-1318.

[31] S. Hochreiter y J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, n.º 8, pp. 1735-1780, 1997.

[32] F. A. Gers, J. Schmidhuber y F. Cummins, "Learning to forget: Continual prediction with LSTM," *Neural computation*, vol. 12, n.º 10, pp. 2451-2471, 2000.

[33] A. Graves, "Generating sequences with recurrent neural networks," *arXiv preprint arXiv:1308.0850*, 2013.

[34] X. Shi et al., "Convolutional LSTM network: A machine learning approach for precipitation nowcasting," *arXiv preprint arXiv:1506.04214*, 2015.

[35] H. Shimodaira, "Improving predictive inference under covariate shift by weighting the log-likelihood function," *Journal of statistical planning and inference*, vol. 90, n.º 2, pp. 227-244, 2000.

[36] S. Ioffe y C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," en *International conference on machine learning*, PMLR, 2015, pp. 448-456.

[37] *Batch Normalization Explained*, https://leimao.github.io/blog/Batch-Normalization/, Accessed: 2021-06-06.

[38] G. K. Wallace, "The JPEG still picture compression standard," *IEEE transactions on consumer electronics*, vol. 38, n.º 1, pp. xviii-xxxiv, 1992.

[39] L. Theis, W. Shi, A. Cunningham y F. Huszár, "Lossy image compression with compressive autoencoders," *arXiv preprint arXiv:1703.00395*, 2017.

[40] V. Chandola, A. Banerjee y V. Kumar, "Anomaly detection: A survey," *ACM computing surveys (CSUR)*, vol. 41, n.º 3, pp. 1-58, 2009.

[41] N. Görnitz, M. Kloft, K. Rieck y U. Brefeld, "Toward supervised anomaly detection," *Journal of Artificial Intelligence Research*, vol. 46, pp. 235-262, 2013.

[42] Wikipedia contributors, *K-means clustering — Wikipedia, The Free Encyclopedia*, [Online; accessed 19-June-2021], 2021. [En línea]. Disponible en: https://en.wikipedia.org/w/index.php?title=K-means_clustering&oldid=1021685301.

[43] M. Sakurada y T. Yairi, "Anomaly detection using autoencoders with nonlinear dimensionality reduction," en *Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis*, 2014, pp. 4-11.

[44] M. Schreyer, T. Sattarov, D. Borth, A. Dengel y B. Reimer, "Detection of anomalies in large scale accounting data using deep autoencoder networks," *arXiv preprint arXiv:1709.05254*, 2017.

[45] Y. LeCun y C. Cortes, "MNIST handwritten digit database," 2010. [En línea]. Disponible en: http://yann.lecun.com/exdb/mnist/.

[46] C. Doersch, "Tutorial on variational autoencoders," *arXiv preprint arXiv:1606.05908*, 2016.

[47] S. Odaibo, "Tutorial: Deriving the standard variational autoencoder (vae) loss function," *arXiv preprint arXiv:1907.08956*, 2019.

[48] *What are Generative Adversarial Networks?* https://2018.igem.org/Team:Vilnius-Lithuania-OG/Gan_Introduction, Accessed: 2021-06-07.

[49] *Chapter 4 Bayesian Decision Theory*, https://www.byclb.com/TR/Tutorials/neural_networks/ch4_1.htm, Accessed: 2021-06-04.

[50] T. Schlegl, P. Seeböck, S. M. Waldstein, U. Schmidt-Erfurth y G. Langs, "Unsupervised Anomaly Detection with Generative Adversarial Networks to Guide Marker Discovery," *CoRR*, vol. abs/1703.05921, 2017. arXiv: 1703.05921. [En línea]. Disponible en: http://arxiv.org/abs/1703.05921.

[51] H. Zenati, C. S. Foo, B. Lecouat, G. Manek y V. R. Chandrasekhar, "Efficient gan-based anomaly detection," *arXiv preprint arXiv:1802.06222*, 2018.

[52] F. Di Mattia, P. Galeone, M. De Simoni y E. Ghelfi, "A survey on gans for anomaly detection," *arXiv preprint arXiv:1906.11632*, 2019.

[53] S. Akcay, A. Atapour-Abarghouei y T. P. Breckon, "Ganomaly: Semi-supervised anomaly detection via adversarial training," en *Asian conference on computer vision*, Springer, 2018, pp. 622-637.

[54] S. Akçay, A. Atapour-Abarghouei y T. P. Breckon, "Skip-ganomaly: Skip connected and adversarially trained encoder-decoder anomaly detection," en *2019 International Joint Conference on Neural Networks (IJCNN)*, IEEE, 2019, pp. 1-8.

[55] T. Salimans et al., "Improved techniques for training gans," *arXiv preprint arXiv:1606.03498*, 2016.

[56] I. J. Myung, "Tutorial on maximum likelihood estimation," *Journal of mathematical Psychology*, vol. 47, n.º 1, pp. 90-100, 2003.