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

DISEÑO DE DISTRIBUCIONES A PRIORI DE REDES NEURONALES BAYESIANAS BASADAS EN LA SOLUCIÓN DE SU VERSIÓN DETERMINISTA.

TESIS PARA OPTAR AL GRADO DE
MAGÍSTER EN CIENCIAS DE LA INGENIERÍA, MENCIÓN MATEMÁTICAS APLICADAS

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL MATEMÁTICO

DAVID JOSE MOLINA PINOS

PROFESOR GUÍA:
FELIPE TOBAR HENRÍQUEZ

MIEMBROS DE LA COMISIÓN:
JOAQUÍN FONTBONA TORRES
ALVARO MACÍAS ARAYA

Este trabajo ha sido parcialmente financiado por Fondecyt Regular N° 1210606
y CMM ANID BASAL FB210005

SANTIAGO DE CHILE
2024

RESUMEN DE LA TESIS PARA OPTAR
AL GRADO DE MAGÍSTER EN CIENCIAS DE LA INGENIERÍA,
MENCIÓN MATEMÁTICAS APLICADAS
RESUMEN DE LA MEMORIA PARA OPTAR
AL TÍTULO DE INGENIERO CIVIL MATEMÁTICO
POR: DAVID JOSE MOLINA PINOS
FECHA: 2023
PROF. GUÍA: FELIPE TOBAR HENRÍQUEZ

DISEÑO DE DISTRIBUCIONES A PRIORI DE REDES NEURONALES BAYESIANAS BASADAS EN LA SOLUCIÓN DE SU VERSIÓN DETERMINISTA

Las redes neuronales son modelos matemáticos utilizados en el aprendizaje profundo que a través de las variables que representan a los datos, han demostrado su capacidad para resolver una amplia variedad de problemas. Sin embargo, también poseen desventajas. Uno de los principales problemas asociados con estas redes es su susceptibilidad a la sobre confianza. En otras palabras, a menudo tienen la tendencia a proporcionar resultados con una alta probabilidad, incluso si esos resultados son incorrectos. Lo cual en aplicaciones de alto riesgo, como conducción autónoma, pueden existir graves consecuencias.

Si se incorpora una perspectiva probabilística en estos modelos, lo cual se conoce como redes neuronales bayesianas, se puede obtener una medida de incertidumbre en las predicciones, lo que indica cuánto confía el modelo en esas predicciones. Si bien suena prometedor, hasta principios de 2020, no se han reportado desarrollos de modelos de aprendizaje profundo bayesiano en la industria, a pesar de los avances en este campo, esto se debe principalmente a que tienen menor precisión que la versión determinista de tales modelos. En este tipo de modelos se debe definir una distribución a priori de los parámetros. Una de las posibles causas del problema señalado para estos modelos es la mala elección del prior, pues comúnmente se utiliza simplemente una distribución normal centrada en cero. Por lo que el principal objetivo de esta investigación, es poder encontrar una distribución a priori, que tenga un buen rendimiento, y que además sirva para poder estimar la incertidumbre de las predicciones.

Se proponen priors basados en la solución de la forma determinista de la red, y se comparan con algunos de los priors estudiados hasta la fecha. Los resultados muestran que a partir de uno de los priors propuestos, se supera el rendimiento de la red determinista, pero que sin embargo en algunos casos, se obtiene una peor calibración. También se muestra una aplicación en salud acerca de cómo poder aplicar la propuesta realizada en un conjunto de datos de distrofias musculares, y es posible ver cómo el cálculo de la incertidumbre es coherente con lo que quiere lograr.

Se concluye que los resultados obtenidos son relevantes, pues se muestra que uno de los priors propuestos logra superar en ciertas métricas a la versión determinista, independiente de la arquitectura y conjunto de datos utilizado. Por último se propone seguir estudiando tal distribución, en problemas más complejos, con diferentes arquitecturas y diferentes datos, de manera de poder realizar una conclusión más completa, además de utilizar otras técnicas de inferencia.

A mis padres.

Agradecimientos

Quisiera agradecer a mis papás, Jheny y Gabriel por todo el apoyo que me brindaron en mi paso por la universidad, en donde siempre estuvieron presentes y no permitieron que nada me faltara, logrando que mi prioridad sean mis estudios, y así enfocarme totalmente en ellos. No me queda más que agradecerles por todo el sacrificio que han hecho durante estos años para que pueda culminar mis estudios de la mejor manera, muchas gracias.

Agradecer también a mi profesor guía Felipe Tobar, por acompañarme durante todo el proceso de Tesis, con su amplia experiencia y dándose el tiempo de reunirse semanalmente conmigo a conversar sobre los diferentes avances, agradecerle también por darme la libertad de investigar sobre un tema de mi interés, y por darme la confianza necesaria para poder realizar esta Tesis. A los profesores integrantes de la comisión Joaquín Fontbona y Alvaro Macías por su interés en participar en este trabajo. Agradecer también a mi profesor de la educación media Luis Muñoz, por motivarme a seguir aprendiendo y por mostrarme de lo que era capaz, muchas gracias por su desinteresada ayuda. Muchas gracias, sin duda jugaron un papel importante en mi desarrollo profesional.

Agradecer a mis amigos de la universidad Cristian, Nicolás, Arie, Felipe, Max, Vicky, y Ricardo por su ayuda durante todos estos años y por pasar muy buenos momentos fuera del estudio, gracias a todos. Agradecer también a mi polola Paula, por apoyarme durante todo este proceso y por preocuparse de que todo salga bien, muchas gracias.

Agradecer también al resto de mi familia, que a pesar de la distancia física, siempre estuvieron presentes con sus palabras de apoyo.

Muchas gracias a todos, no me queda duda que sin ustedes este trabajo no hubiera sido posible.

Tabla de Contenido

1. Introducción **1**

2. Marco teórico **3**

2.1. Aprendizaje profundo 3

2.1.1. Perceptrón 3

2.1.2. Perceptrón multicapa 4

2.1.3. Redes neuronales convolucionales 5

2.1.4. Redes neuronales recurrentes 8

2.1.5. Entrenamiento de una red neuronal 10

2.2. Inferencia bayesiana 11

2.2.1. Aproximación de la distribución posterior 12

2.3. Estimación de una densidad de probabilidad 15

2.3.1. Flujos normalizadores 16

3. Antecedentes **18**

3.1. Redes neuronales bayesianas 18

3.2. Problema 20

3.3. Estado del arte 21

4. Metodología **25**

4.1. Datos y modelos 25

4.1.1. Datos 25

4.1.2. Modelos 26
4.2. Propuestas y contribuciones 27
4.2.1. Propuesta 1 27
4.2.2. Propuesta 2 29
4.2.3. Contribuciones 29
4.3. Experimentos 30
4.3.1. Implementación propuesta 1 30
4.3.2. Evaluación de los priors 30
4.3.3. Estudio de las muestras de la posterior 32
4.3.4. Aplicación en salud 32

5. Análisis y resultados **33**
5.1. Implementación propuesta 1 33
5.1.1. Inspección de los parámetros 33
5.1.2. Máxima verosimilitud para la definición de la distribución a priori . . 34
5.1.3. Flujos normalizadores para la construcción de distribuciones a priori . 36
5.2. Evaluación de los priors 36
5.2.1. Elección de hiperparámetros 36
5.2.2. Evolución de los modelos 38
5.2.3. Estudio del cold posterior effect 39
5.3. Estudio de las muestras de la posterior 42
5.4. Aplicación en salud 43

6. Conclusión **45**

Bibliografía **51**

Anexo A. **52**
A.1. Aplicación en salud 52
A.2. Inspección de la posterior. 52

Índice de Tablas

4.1. Priors a considerar. $\mathcal{N}$ y $\mathcal{L}$ corresponden a la distribución Normal y de Laplace respectivamente. 31

5.1. Resultados de estimadores de máxima verosimilitud para el MLP y la CNN, caso Gaussiano. 35

5.2. Resultados de estimadores de máxima verosimilitud para el MLP y la CNN, caso Laplace. 35

5.3. Resultados para el MLP bayesiano. Batch corresponde al tamaño del batch, Logll corresponde a la log verosimilitud, y Ece al expectation calibration error. 37

5.4. Resultados para la CNN bayesiana. Batch corresponde al tamaño del batch, Logll corresponde a la log verosimilitud, y Ece al error esperado de calibración. 38

Índice de Ilustraciones

2.1. Gráficos de funciones de activación. Elaboración propia. 5

2.2. The CNN caption 7

2.3. Diagrama de una RNN. Elaboración propia. 9

2.4. Diagrama de una RNN con dos capas. Elaboración propia. 9

10figure.caption.9

4.1. Ejemplos de MNIST. Elaboración propia. 25

4.2. Ejemplos de FashionMNIST. Elaboración propia. 26

4.3. Ejemplos de CIFAR-10. Elaboración propia. 26

5.1. Histograma de los parámetros ya entrenados del perceptrón multicapa, separados por capa, en MNIST. Elaboración propia. 34

5.2. Histograma de los parámetros ya entrenados de la CNN, separados por capa, en MNIST. Elaboración propia. 34

5.3. Histograma de los parámetros ya entrenados de la CNN y la función de densidad creada a través de flujos normalizadores. Elaboración propia. 36

5.4. Evolución del rendimiento para cada prior para el MLP en FashionMNIST. Logll corresponde a la log verosimilitud y Ece corresponde al error de calibración esperado. Elaboración propia. 39

5.5. Cold posterior effect para cada prior, para el MLP en todos los conjuntos de datos considerados. Elaboración propia. 40

5.6. Cold posterior effect para cada prior, para la CNN en todos los conjuntos de datos considerados. Elaboración propia. 41

5.7. Histograma de las muestras de la posterior, para el MLP en los datos MNIST. Utilizando el prior Normal y NF. Elaboración propia. 42

5.8. Histograma de las muestras de la posterior, para el MLP en los datos MNIST, utilizando el prior Determinista. Elaboración propia. 42

5.9. Matriz de confusión para las predicciones en el conjunto de prueba. Elaboración propia 43

5.10. Disminución de la dimensionalidad, usando t-SNE y UMAP por separado. Elaboración propia. 44

A.1. Valores nulos por atributo. Elaboración propia. 52

A.2. Histograma de los parámetros ya entrenados separados por capa en Fashion-MNIST. Elaboración propia. 53

A.3. Histograma de los parámetros ya entrenados separados por capa en CIFAR-10. Elaboración propia. 53

A.4. Histograma de los parámetros ya entrenados separados por capa en Fashion-MNIST. Elaboración propia. 54

A.5. Histograma de los parámetros ya entrenados separados por capa en CIFAR-10. Elaboración propia. 54

Capítulo 1

Introducción

En la última década, el aprendizaje profundo ha experimentado un rápido avance debido al progreso en tecnología de hardware, como el procesamiento de gráficos (GPU), y la disponibilidad de grandes conjuntos de datos [44]. Los modelos que se utilizan en esta área, han sido aplicados en una gran cantidad de campos, sin embargo tienen una desventaja, que consiste básicamente en que tienden a dar resultados con alta probabilidad para predicciones incorrectas.

Una manera de poder enfrentar este problema es a través de la versión probabilística de este tipo de modelos, también conocidos como redes neuronales bayesianas, en donde los parámetros son variables aleatorias, y para poder realizar predicciones se utiliza la distribución condicional de los parámetros dado los datos. Para este tipo de modelos también existe un problema, que se debe a que su rendimiento es inferior a su versión determinista (esto para el comúnmente utilizado prior Gaussiano isotropico), una de las posibles causas es la mala elección del prior. Se han propuesto distribuciones a priori que superan a la versión determinista de la red [14], pero que lo hacen para arquitecturas o conjuntos de datos particulares. Es sumamente relevante encontrar una buena distribución a priori, pues permitiría tener una versión de las redes neuronales, capaces de estimar la incertidumbre, así como también tener predicciones precisas, lo cual ayudaría a aplicar este tipo de modelos en situaciones riesgosas.

El objetivo de esta investigación es poder encontrar una distribución a priori, que mejore el rendimiento con respecto a la versión de determinista de la red, poder evaluar estas distribuciones a través de diferentes arquitecturas y conjunto de datos, así como también realizar un caso práctico en donde la estimación de la incertidumbre sea importante, como por ejemplo, en diagnósticos médicos.

Para lograr lo anterior, se proponen distribuciones a priori basadas en la solución de la red determinista, y se utiliza un perceptrón multicapa, y una red neuronal convolucional para evaluar estas distribuciones en tres conjuntos de datos.

La estructura de la Tesis es la siguiente:

1. Introducción.
2. Marco Teórico: Para lograr una mejor comprensión del trabajo realizado, se explica

qué es el aprendizaje profundo, los principales modelos, como es que estos modelos aprenden de los datos, también se realiza un resumen de inferencia bayesiana, y cómo es posible estimar densidades de probabilidad.

3. Antecedentes: Acá se explican las redes neuronales bayesianas, y cómo poder calcular la incertidumbre de las predicciones, también se habla acerca del problema que se quiere resolver, y del estado del arte de tal problema.

4. Metodología: En este Capítulo se detalla cuáles son las propuestas, cuales son las contribuciones de la investigación, y también cómo son los experimentos que se llevarán a cabo.

5. Análisis y Resultados: Se exponen los resultados de los experimentos descritos en la metodología, y se realiza un análisis detallado para cada uno de los experimentos.

6. Conclusión.

Capítulo 2

Marco teórico

Durante todo este Capítulo se considerará lo siguiente, se tendrá $N > 0 \in \mathbb{N}$ datos, denotados como $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$ con $x_i \in \mathbb{R}^Q$, $y_i \in \mathbb{R}^D$ para todo $i \in \{1, \ldots, N\}$, con Q, $D \in \mathbb{N}$ la dimensión de entrada y de salida respectivamente.

2.1. Aprendizaje profundo

El aprendizaje profundo, es un subcampo del aprendizaje automático que utiliza redes neuronales artificiales para aprender de las variables que representan a los datos [31] y a partir de lo anterior poder resolver una amplia cantidad de problemas, como lo son el reconocimiento de imágenes [34], reconocimiento de voz [41], detección de fraudes [46], así como también ser una herramienta útil para los sistemas de recomendación [60]. Han llegado a abarcar diferentes áreas como la astronomía [37], deportes [5], medicina [54] y ciberseguridad [2], por dar algunos ejemplos.

2.1.1. Perceptrón

El perceptrón es un modelo probabilístico propuesto en 1958 por el psicólogo Frank Rosenblatt [47], el cual incorpora aspectos biológicos del cerebro [23]. Este modelo corresponde a la versión más básica de una red neuronal y matemáticamente se define de la siguiente manera:

Definición 2.1 (Perceptrón) *El perceptrón es una función $f : \mathbb{R}^Q \to \mathbb{R}$ tal que, para $x \in \mathbb{R}^Q$:*

$$f(x) = \begin{cases} 1, & si\ w^T x + b \geq 0 \\ 0, & si\ w^T x + b < 0. \end{cases} \tag{2.1}$$

Si bien este modelo es útil en algunas aplicaciones, su capacidad de es limitada debido a su arquitectura, ya que funciona solo para problemas de clasificación binaria. Desde entonces,

se han desarrollado nuevas arquitecturas de redes neuronales más complejas, que permiten abordar tareas más complejas [8].

2.1.2. Perceptrón multicapa

El perceptrón multicapa también fue introducido por Frank Rosenblatt en [47] y puede ser entendido como una red de perceptrones conectados que forman múltiples capas, formalmente se definen de la siguiente manera:

Definición 2.2 (Perceptrón multicapa) *Es una función f tal que $f : \mathbb{R}^Q \to \mathbb{R}^D$. La cual está definida recursivamente de la siguiente manera:*

$$h_0 = x \tag{2.2}$$
$$h_i = g_i\left(W_i h_{i-1} + b_i\right) \tag{2.3}$$
$$h_M = W_M h_{M-1} + b_M \tag{2.4}$$
$$f(x) = h_M. \tag{2.5}$$

Considerando $M > 1$, $K_0 = Q$, $K_i \in \mathbb{N}$ para $i \in \{1, \ldots, M-1\}$ y $K_M = D$. Se tiene que $W_i \in \mathbb{R}^{K_i \times K_{i-1}}, b_i \in \mathbb{R}^{K_i}$. h_0 es conocido como capa de entrada, para $i \in \{1, \ldots, M-1\}$, h_i es conocido como capa oculta, y h_M es conocido como capa de salida. Se abrevia como MLP por sus siglas en inglés (multilayer perceptron). Por último, para cada $i \in \{1, \ldots, M-1\}$, g_i es una función de activación, las cuales tienen como objetivo que las redes neuronales sean funciones no lineales.

Algunos ejemplos de funciones de activación son las siguientes:

1. Una función de activación que se caracteriza por su facilidad de implementación y buen rendimiento [59], es la función ReLU: $\mathbb{R} \to \mathbb{R}$ (por sus siglas en inglés, *rectified linear unit*) [40]. Para $x \in \mathbb{R}$:

$$\mathrm{ReLU}(x) = \max(0, x). \tag{2.6}$$

2. La función de activación sigmoid, denotada como σ transforma valores reales al intervalo $(0, 1)$. Para $x \in \mathbb{R}$:

$$\sigma(x) = \frac{1}{1 + \exp(-x)}. \tag{2.7}$$

3. La tangente hiperbólica, denotada como tanh, transforma valores reales al intervalo $(-1, 1)$. Para $x \in \mathbb{R}$:

$$\tanh(x) = \frac{1 - \exp(-2x)}{1 + \exp(-2x)}. \tag{2.8}$$

En la Figura 2.1 se puede ver una ilustración de cómo son estas funciones. En caso de aplicar una función de activación a un vector, esta simplemente se aplica componente a componente,

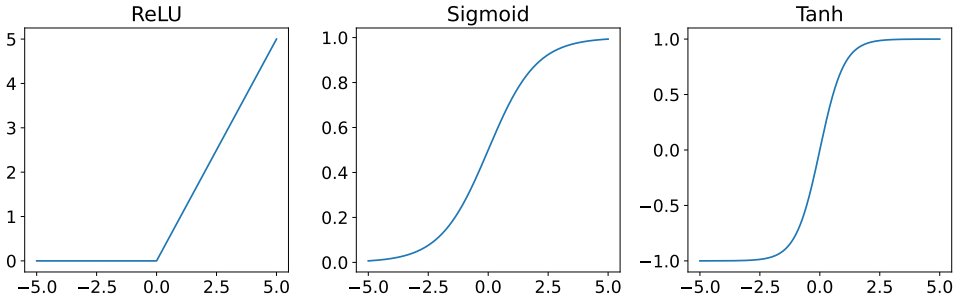

Figura 2.1: Gráficos de funciones de activación. Elaboración propia.

es decir, para g una función de activación y $x = (x_i)_{i=1}^{Q} \in \mathbb{R}^Q$ con $Q \in \mathbb{N}$ se adoptará la siguiente notación,

$$g(x) = (g(x_1), g(x_2), \ldots, g(x_n)) \,. \tag{2.9}$$

Un teorema relevante de esta arquitectura es que incluso para solo una capa oculta, es posible aproximar cualquier función vectorial definida en un compacto.

Antes de enunciar el teorema, es necesario introducir la siguiente notación: $C(X, Y)$ corresponde al conjunto de las funciones continuas que van desde X a Y.

Teorema 2.3 *(Aproximación universal [45]) Sea $\varphi \in C(\mathbb{R}, \mathbb{R})$. Se tiene que φ no es polinomial si y solo si para todo $n \in \mathbb{N}$, $m \in \mathbb{N}$, compacto $K \subseteq \mathbb{R}^n$, $f \in C(K, \mathbb{R}^m)$, $\varepsilon > 0$, existe $k \in \mathbb{N}$, $W \in \mathbb{R}^{k \times n}$, $b \in \mathbb{R}^k$, $C \in \mathbb{R}^{m \times k}$ tal que,*

$$\sup_{x \in K} \|f(x) - g(x)\| < \varepsilon,$$

con

$$g(x) = C\left(\varphi\left(Wx + b\right)\right).$$

Si bien, a partir de un MLP con una capa oculta, es posible aproximar cualquier función, cuando se trata de problemas asociados a la clasificación de imágenes, las MLP tienen una limitación, ya que tratan cada píxel como una característica independiente, sin tener en cuenta la estructura espacial de los datos.

2.1.3. Redes neuronales convolucionales

Las redes neuronales convolucionales (CNN) son un tipo de arquitectura hecha para trabajar con imágenes, de hecho su diseño está basado en la percepción visual [19].

Este tipo de redes forman una parte relevante dentro de los avances del aprendizaje profundo, si bien se utilizó en los años 90 para el reconocimiento de dígitos [32], sus amplias aplicaciones actuales se deben a trabajos más recientes [10], cuando una CNN ganó la competencia reconocimiento visual ImageNet [29].

La principal transformación que compone a las CNNs es la denominada convolución. Se ilustrará con un ejemplo tal operación para luego definirla formalmente.

Sea M una imagen (que simplemente se entiende como una matriz) de 100x100, tal que

$$M_{ij} = \begin{cases} 0, & j <= 50, \\ 1, & j > 50. \end{cases}$$

Es decir, la mitad blanca y la otra mitad negra (M_{ij} representa la intensidad del color negro en el píxel de la posición (i, j)), sea además $K = \begin{pmatrix} -1/3 & 0 & 1/3 \\ -1/3 & 0 & 1/3 \\ -1/3 & 0 & 1/3 \end{pmatrix}$.
Es posible transformar la imagen M utilizando K a través de la siguiente transformación:

$$S(i, j) = \sum_{n=0}^{2} \sum_{m=0}^{2} M(i + m, j + n)K(n, m).$$

Obteniendo el siguiente resultado:

$$S_{ij} = \begin{cases} 1 & j \in \{49, 50\} \\ 0 & \sim \end{cases}$$

Es decir una imagen con una línea negra en la mitad.

De alguna manera se puede decir que a través de K se pudo encontrar una característica de la imagen M correspondiente a una línea vertical. Se podría probar con diferentes tipos de matrices K y encontrar otras características, como bordes o lineas horizontales, pero también es posible considerar a K como un parámetro y que el mismo modelo aprenda, mediante técnicas de optimización, cuáles son las características. La matriz K anterior es conocido como kernel o filtro.

Para una imagen M de dos dimensiones y un kernel K de dos dimensiones también, la convolución en su formal general tiene la siguiente forma:

$$S(i, j) = (M * K)(i, j) = \sum_{m} \sum_{n} M(i + m, j + n)K(i, j) \tag{2.10}$$

Es posible notar que a través de la convolución el tamaño de la imagen disminuye, esto se puede solucionar agregándole un borde de ceros a la imagen. Esto es lo que se conoce como padding. Un padding $= k$ con $k \in \mathbb{N}$ corresponde a agregarle k bordes de ceros a la imagen.

Adicionalmente, cuando se realiza la convolución, se empieza a través del borde superior izquierdo recorriendo toda la imagen, en el ejemplo anterior se aplicó la convolución considerando que el kernel se movía o bien una dirección a la derecha o bien una dirección hacia abajo, sin embargo, ya sea por eficiencia computacional o por querer disminuir el tamaño de la imagen, el kernel se podría desplazar más de una vez en cada dirección, esto es lo que se conoce como stride, que usualmente corresponde a dos números (s_1, s_2) donde s_1, s_2 son enteros positivos que significan la cantidad de píxeles que se desplaza el kernel horizontal y verticalmente respectivamente.

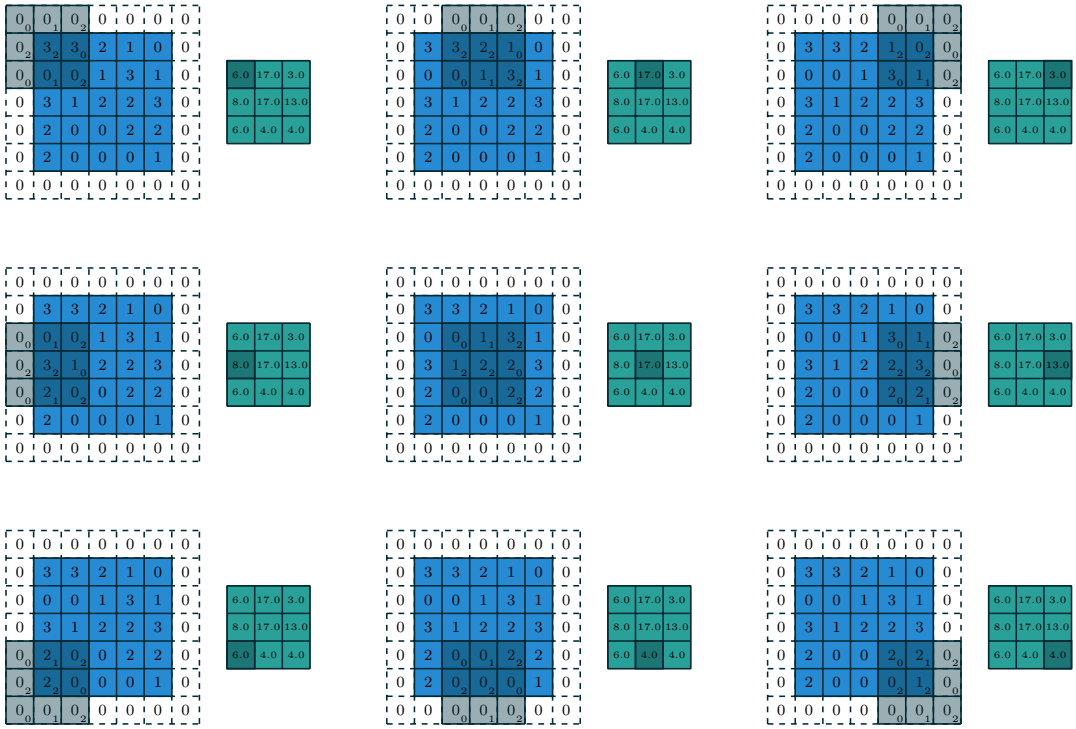

Figura 2.2: Ejemplo de una convolución con stride=$(2, 2)$ y padding $= 1$. La matriz azul corresponde a la imágen, la matriz gris al kernel y la matriz verde corresponde al resultado de la convolución.[1]

En la Figura 2.2 se puede ver una ilustración de la convolución con un stride=$(2, 2)$, un padding$= 1$, y un kernel $K = \begin{pmatrix} 0 & 1 & 2 \\ 2 & 2 & 0 \\ 0 & 1 & 2 \end{pmatrix}$.

Luego de realizar la convolución, por lo general se aplica una función de activación (ReLU por ejemplo) y finalmente una transformación denominada pooling la cual se aplica localmente en la imagen, moviéndose de acuerdo a su stride y retornando un único valor. La convolución con su determinado kernel, padding y stride sumado al pooling es lo que se conoce como capa convolucional. Una red convolucional por lo tanto, es una concanetación de estas capas convolucionales que usualmente termina con una red MLP.

Cabe señalar que a diferencia de la convolución, el pooling no tiene parámetros. Los más clásicos son el max-pooling y el average pooling, que calculan el valor máximo y promedio respectivamente de los píxeles de forma local en la imagen.

Observación En general se aplica más de un filtro por convolución, con el objetivo de extraer más características, esto da como resultado varias matrices de salida las cuales finalmente se concatenan para que posteriormente sean el input de la siguiente capa.

[1]Tomada de: `https://github.com/vdumoulin/conv_arithmetic`

Si bien las redes neuronales convolucionales han demostrado ser efectivas en el procesamiento de imágenes, su capacidad para modelar datos secuenciales (que se detallarán de mejor manera a continuación) es limitada.

2.1.4. Redes neuronales recurrentes

Hasta el momento, nos hemos enfocado datos de entrada de dimensión fija, como lo son los datos tabulares o imágenes. Sin embargo, existen otro tipo de datos que no pueden ser ocupados por las arquitecturas descritas anteriormente, los cuales son los datos secuenciales, que corresponde simplemente a una secuencia de datos representados a través de una tupla [49]. Un ejemplo puede ser la reseña de una película, $(x^{(1)}, x^{(2)}, \ldots, x^{(T)})$ donde cada x es una palabra. Las redes neuronales recurrentes (RNN) [48], se encargan justamente de procesar este tipo de datos. La principal idea de esta arquitectura es que pueda recibir una secuencia de largo arbitrario como input e ir incorporando la información de manera secuencial. Existen varios tipos de problemas donde es necesario ocupar esta arquitectura, por nombrar algunos:

- Series de tiempo.
- Traducción: Dado un texto en un idioma (dato secuencial) traducirlo a otro idioma.
- Reconocimiento de voz.
- Análisis de sentimientos. Dado un texto, identificar si tiene un sentimiento positivo o negativo.

Definición 2.4 (Red neuronal recurrente) *La RNN es una función f tal que para $x = (x_t)_{t=1}^{T}$, $x_t \in \mathbb{R}^Q$:*

$$h_0 = 0 \tag{2.11}$$

$$h_t = g_t\left(Ux_t + b_U + Vh_{t-1} + b_V\right),\ t \in \{1, \ldots, T\} \tag{2.12}$$

$$f(x) = Wh_T + c. \tag{2.13}$$

Con $U \in \mathbb{R}^{M\times Q}$, $V \in \mathbb{R}^{M\times M}$, $b_U, b_V \in \mathbb{R}^M$, $W \in \mathbb{R}^{D\times M}$, $c \in \mathbb{R}^D$ parámetros, $M \in \mathbb{N}$ la dimensión de las variables h_t, $0 \leq t \leq T$. y g_t función de activación para cada $t \in \{1, \ldots, T\}$.

En la Figura 2.3 se puede ver un diagrama que ilustra la definición de la red.

Observación Debido a su practicidad, la tangente hiperbólica es la función de activación que más se utiliza en las RNNs [9].

Es natural pensar que debido a su poca cantidad de parámetros, esta red no puede ser lo suficientemente flexible, sin embargo, es posible seguir agregándole capas como se ilustra en la Figura 2.4.

Si bien este tipo de redes sirven para procesar datos secuenciales, tenían el conocido problema del desvanecimiento del gradiente [27]. Para resolver este problema se creó una RNN más sofisticada denominada *long short-term memory* (LSTM) que resolvía tal problema [18].

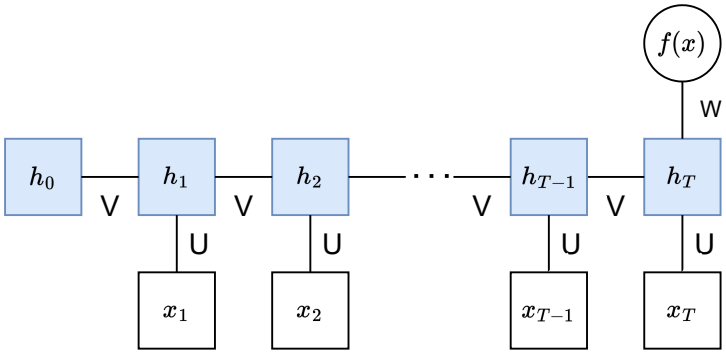

Figura 2.3: Diagrama de una RNN. Elaboración propia.

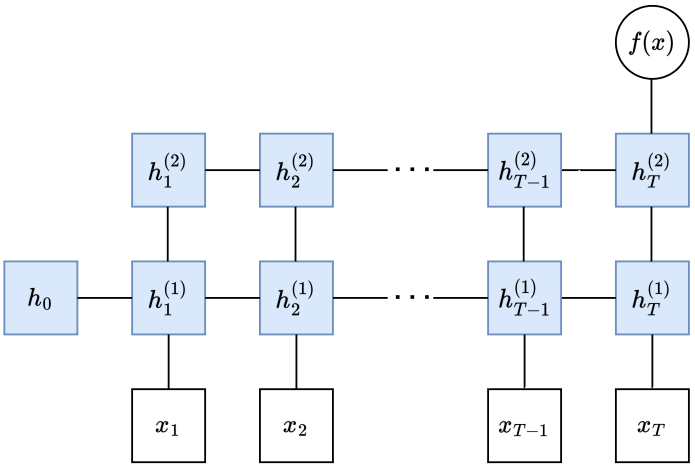

Figura 2.4: Diagrama de una RNN con dos capas. Elaboración propia.

Definición 2.5 (Long Short Term Memory) *Para* $x = (x_t)_{t=1}^T$,

$$i_t = \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{t-1} + b_{hi}) \tag{2.14}$$
$$f_t = \sigma(W_{if}x_t + b_{if} + W_{hf}h_{t-1} + b_{hf}) \tag{2.15}$$
$$g_t = tanh(W_{ig}x_t + b_{ig} + W_{hg}h_{t-1} + b_{hg}) \tag{2.16}$$
$$o_t = \sigma(W_{io}x_t + b_{io} + W_{ho}h_{t-1} + b_{ho}) \tag{2.17}$$
$$c_t = f_t \odot c_{t-1} + i_t \odot g_t \tag{2.18}$$
$$h_t = o_t \odot tanh(c_t) \tag{2.19}$$

h_t es conocido como hidden state, c_t como cell state, i_t, f_t, g_t, o_t como, input, forget, cell y output gates. σ es la función sigmoide y $\odot$ es el producto de Hadamard. Finalmente el output de esta red es $(h_t)_{t=1}^T$.

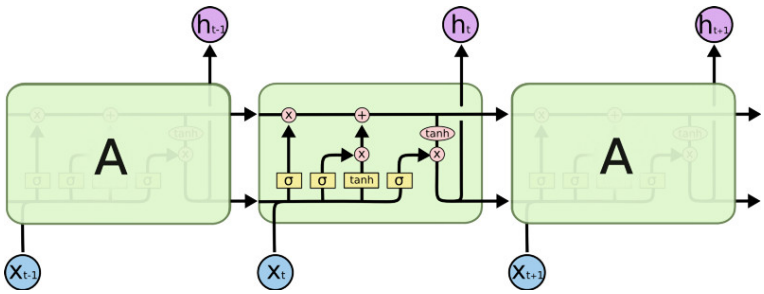

Figura 2.5: Diagrama de una LSTM.[2]

2.1.5. Entrenamiento de una red neuronal

Hasta el momento solo se han definido las redes neuronales, por lo que a continuación se explicará cómo es que este tipo de algoritmos aprenden a resolver diferentes tipos de problemas, dentro de los cuales se encuentra por ejemplo un problema de regresión, el cual se utilizará como ejemplo para poder explicar cómo es el entrenamiento.

Sean $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$ datos, donde y_i corresponde al precio de una vivienda, y x_i a las características de tal vivienda. Y f_W una red neuronal (un MLP por ejemplo) con parámetros W, se busca entonces encontrar los parámetros W que representen de mejor manera la relación entre las características de la vivienda, con su precio. Para lo anterior, se plantea el siguiente problema de optimización:

$$\min_W \mathcal{L}(W, \mathcal{D}). \tag{2.20}$$

$\mathcal{L}$ se conoce como función de costo o de pérdida, para este problema de regresión es común utilizar el error cuadrático medio como función de costo:

$$\mathcal{L}_{\text{ECM}}(W, \mathcal{D}) = \frac{1}{N} \sum_{i=1}^{N} (f_W(x_i) - y_i)^2. \tag{2.21}$$

Para poder minimizar tales funciones de pérdida, se ocupan algoritmos basados en descenso del gradiente, como por ejemplo adam [25]. Este tipo de algoritmos son iterativos y buscan el óptimo a través de la dirección del gradiente, el tamaño del paso con el cual se mueven los parámetros corresponde al learning rate.
Por otro lado, para calcular el gradiente de la función de pérdida, se utiliza en cada iteración un subconjunto aleatorio de los datos denominado batch, utilizar todos los datos tiene la desventaja que en caso de que se tenga un gran volumen de datos sea costoso evaluar la función de pérdida, por otro lado, existen los métodos estocásticos que utilizan solo un dato por cada iteración, lo cual también es ineficiente en caso de que exista una gran cantidad de datos.

Es natural preguntarse qué función de pérdida ocupar para un problema de clasificación, donde el valor a predecir corresponde a un número entero en $\{1, \ldots, C\}$, con $C \in \mathbb{N}$ la cantidad de clases, en tal caso se utiliza una función de pérdida diferente al error cuadrático medio, para la cual es necesaria aplicar una transformación a la salida de la red neuronal de tal

[2]Tomada de: `http://colah.github.io/posts/2015-08-Understanding-LSTMs/`

manera que sea una distribución de probabilidad, es decir que simplemente las componentes del vector sumen 1, y estén entre 0 y 1, esta transformación es la denominada softmax:

$$\text{softmax}(z)_i = \frac{e^{z_i}}{\sum_{i=1}^{C} e^{z_i}}, \tag{2.22}$$

con $i \in \{1, \ldots, C\}$ y $z \in \mathbb{R}^C$.

De esta manera, la función de pérdida es la siguiente,

$$\mathcal{L}_{EC}(W, \mathcal{D}) = -\frac{1}{N} \sum_{i=1}^{N} \sum_{c=1}^{C} 1_{y_i=c} \log\, p(\text{softmax}(f_W(x_i)_c)), \tag{2.23}$$

también conocida como entropía cruzada.

2.2. Inferencia bayesiana

Sea nuevamente f_W una red neuronal, con W sus parámetros y $\mathcal{D} = (x_i, y_i)_{i=1}^N$ datos, que se asumirán independientes e idénticamente distribuidos. Como se vio en la sección anterior, a partir del problema de optimización planteado, se busca el valor de los parámetros W que minimicen la función de pérdida. Otra forma de abordar este problema es mediante la estadística bayesiana, en la que se considera una distribución de probabilidad sobre los parámetros W.

En este marco de trabajo, los parámetros del modelo, denotados usualmente como θ, son desconocidos y por lo tanto representados como una variable aleatoria. Un ejemplo de modelo puede ser justamente la red neuronal f_W.

Primeramente se define el prior (o distribución a priori), como el conocimiento que se tiene acerca de los parámetros antes de ver los datos, y se denota como $p(\theta)$.

También se define la verosimilitud, que corresponde a la siguiente distribución condicional,

$$y|x, \theta \sim p(y|x, \theta), \tag{2.24}$$

con x la variable dependiente, e y la variable independiente o variable a predecir.

Utilizando teorema de Bayes [1], es posible conocer la distribución de los parámetros θ luego de observar los datos, a partir de la siguiente expresión:

$$p(\theta|\mathcal{D}) = \frac{p(\mathcal{D}|\theta)p(\theta)}{p(\mathcal{D})}, \tag{2.25}$$

la cual es conocida como distribución posterior.

Notar que en este caso, no se tiene un valor puntual de los parámetros θ sino más bien una distribución de probabilidad. Una de las aplicaciones más interesantes de esto es que dado

un nuevo dato de entrada x^*, se tiene una distribución de probabilidad para la predicción, correspondiente a:

$$p(y^*|\mathcal{D}) = \int p(y^*|x^*, \theta)p(\theta|\mathcal{D})d\theta. \tag{2.26}$$

Esta integral se puede interpretar como promediar de acuerdo a su probabilidad ($p(\theta|\mathcal{D})$) todos los posibles modelos ($p(y^*|x^*, \theta)$). Esto es lo que se conoce como inferencia bayesiana.

Para modelos sencillos, como una regresión lineal, la expresión anterior tiene una forma cerrada (considerando un prior y una verosimilitud Gaussiana), sin embargo, si se consideran modelos más complejos, como una red neuronal, que debido a su no linealidad la distribución posterior es intratable [39]. En tales casos se utilizan técnicas que permitan aproximar la distribución posterior.

2.2.1. Aproximación de la distribución posterior

2.2.1.1. Aproximación de Laplace

La aproximación de Laplace [6] selecciona el parámetro que maximiza la función de densidad de la distribución posterior, también conocido como máximo a posteriori:

$$\begin{aligned} \theta_{\text{MAP}} &= \underset{\theta}{\operatorname{argmax}}\, p(\theta|\mathcal{D}) \\ &= \underset{\theta}{\operatorname{argmax}}\, p(\mathcal{D}|\theta)p(\theta). \end{aligned} \tag{2.27}$$

Luego se realiza una aproximación de Taylor de segundo orden al logaritmo de la posterior en torno a θ_{MAP},

$$\begin{aligned} \log p(\theta|\mathcal{D}) \approx \log p(\theta_{\text{MAP}}|\mathcal{D}) \\ + \frac{1}{2}(\theta - \theta_{\text{MAP}})^T(H + \tau I)(\theta - \theta_{\text{MAP}}). \end{aligned} \tag{2.28}$$

Con H el hessiano del $\log p(\theta|\mathcal{D})$. En caso de que $p(\theta) = \mathcal{N}(0, \tau^{-1}I)$, $\tau > 0$, se obtiene que la aproximación de la distribución posterior corresponde a una distribución Gaussiana centrada en θ_{MAP} y con una matriz de covarianzas igual a $(H + \tau I)^{-1}$.

La principal desventaja que tiene esta aproximación es que solo se considera el θ_{MAP} y un entorno simétrico de tal punto, y no considera otras regiones del soporte.

2.2.1.2. Inferencia Variacional

La inferencia variacional [22], [4] consiste en encontrar una distribución $q_\phi(\theta)$ que sea fácil de evaluar, esté parametrizada por ϕ y que sea "similar" a $p(\theta|\mathcal{D})$. Para realizar tal aproximación, se busca minimizar la divergencia de Kullback - Leibler [30]:

$$\text{KL}(q_\phi(\theta)||p(\theta|\mathcal{D})) = \int q_\phi(\theta)\log\left(\frac{q_\phi(\theta)}{p(\theta|\mathcal{D})}\right)d\theta. \tag{2.29}$$

Observación Lo anterior está definido solo cuando $q_\phi(\theta)$ es absolutamente continua con respecto a $p(\theta|\mathcal{D})$.

La ecuación (2.29) establece una forma no simétrica de medir qué tan similares son dos distribuciones (es cero en caso de que las distribuciones sean iguales, y es positiva en cualquier otro caso), por lo tanto, se busca ϕ tal que minimice la divergencia anterior. Utilizando el teorema de Bayes:

$$p(\theta|\mathcal{D}) = \frac{p(\mathcal{D}|\theta)p(\theta)}{p(\mathcal{D})} \propto p(\mathcal{D}|\theta)p(\theta),$$

se obtiene que:

$$\begin{aligned}\mathrm{KL}(q_\phi(\theta)||p(\theta|\mathcal{D})) &\propto \int q_\phi(\theta)\log\left(\frac{q_\phi(\theta)}{p(\mathcal{D}|\theta)p(\theta)}\right)d\theta \\ &= -\int q_\phi(\theta)\log\left(p(\mathcal{D}|\theta)\right)d\theta + \int q_\phi(\theta)\log\left(\frac{q_\phi(\theta)}{p(\theta)}\right)d\theta.\end{aligned}$$

Por lo que minimizar la divergencia KL es equivalente a minimizar:

$$\begin{aligned}\mathcal{L}(\phi) &:= -\int q_\phi(\theta)\log(p(\mathcal{D}|\theta))d\theta + \mathrm{KL}(q_\phi(\theta)||p(\theta)) \\ &= -\sum_{i=1}^{N}\int q_\phi(\theta)\log(p(y_i|x_i,\theta))d\theta + \mathrm{KL}(q_\phi(\theta)||p(\theta)).\end{aligned} \tag{2.30}$$

El principal problema de minimizar tal expresión, es que evaluar la sumatoria podría ser muy costoso para N grande, con N la cantidad de observaciones. Para resolverlo, se utiliza una técnica conocida como optimización a través de mini batches, que consiste en aproximar la ecuación anterior, mediante el siguiente estimador insesgado:

$$\hat{\mathcal{L}}(\phi) := -\frac{N}{M}\sum_{i\in S}\int q_\phi(\theta)\log\, p(y_i|x_i,\theta)d\theta + KL(q_\phi(\theta)||p(\theta)), \tag{2.31}$$

con $S \subseteq N$ un conjunto aleatorio de tamaño M, luego se puede ocupar algún algoritmo de optimización estocástica para resolver lo anterior, obteniendo así un óptimo ϕ^* que también será óptimo de $\mathcal{L}(\phi)$.

Sea Φ el conjunto del parámetro ϕ, se le llama familia variacional al conjunto $\{q_\phi :\ \phi \in \Phi\}$, es decir, al conjunto en donde se busca la distribución más similar a la distribución posterior. El problema con esta técnica, es que la aproximación al restringirse justamente a la familia variacional, existe una compensación entre escoger una familia variacional más amplia y poder resolver el problema variacional.

2.2.1.3. Métodos de muestreo

Esta técnica se enfoca en obtener muestras de la distribución posterior, y no buscar una distribución cercana a la posterior, por lo que no se restringe a una familia de distribuciones.

Uno de estos métodos que escala a grandes volúmenes de datos (algo usual en aprendizaje profundo) es SGLD [56] (Stochastic Gradient Langevin Dynamics). El algoritmo consiste en lo siguiente: en cada iteración t se define un batch de $n < N$ datos $X_t = \{x_{ti}\}_{i=1}^{n}$. Luego la actualización de los parámetros se realiza de la siguiente manera:

$$\theta_{t+1} = \theta_t - \frac{\varepsilon_t}{2}\left(\nabla \log\left(p(\theta_t)\right) + \frac{N}{n}\sum_{i=1}^{n}\nabla \log\left(p(x_{ti}|\theta_t)\right) + \right) + \eta_t, \tag{2.32}$$

tal que ε_t

$$\sum_{t\geq 1}\varepsilon_t = \infty, \quad \sum_{t\geq 1}\varepsilon_t^2 < \infty,$$

y $\eta_t \sim \mathcal{N}(0, \varepsilon_t)$.

Observación SGLD está motivado y derivado de una ecuación diferencial estocástica.

Teniendo, las aproximaciones dadas por inferencia variacional, la aproximación de Laplace, o bien las muestras dadas por SGLD, es natural preguntarse cómo utilizarlas para poder calcular la distribución de la predicción $p(y^*|\mathcal{D})$ definida en la ecuación (2.26).

Sea entonces $q(\theta)$ una aproximación de la distribución posterior, dada por la aproximación de Laplace o por inferencia variacional, de esta manera,

$$\begin{aligned} p(y^*|\mathcal{D}) &= \int p(y^*|x^*, \theta)p(\theta|\mathcal{D})d\theta \\ &\approx \int p(y^*|x^*, \theta)q(\theta)d\theta. \end{aligned} \tag{2.33}$$

Por lo general, la cantidad de parámetros de una red neuronal, es lo suficientemente grande para que calcular la integral de (2.33) sea computacionalmente costoso, sin embargo, es posible aproximarla utilizando la aproximación de Monte Carlo, es decir:

$$\int p(y^*|x^*, \theta)q(\theta)d\theta \approx \frac{1}{M}\sum_{i=1}^{M} p(y^*|x^*, \theta_i), \tag{2.34}$$

con $\theta_i \sim q(\theta)$, con $i \in \{1, \ldots, M\}$, y $M > 0$ la cantidad de muestras, lo cual es factible, pues como se comento, obtener muestras de $q(\theta)$ es fácil.

En el caso de SGLD, se aproxima la distribución de la predicción directamente a través de Monte Carlo, utilizando una aproximación empírica de $p(\theta|\mathcal{D})$, es decir,

$$\begin{aligned} p(y^*|\mathcal{D}) &= \int p(y^*|x^*, \theta)p(\theta|\mathcal{D})d\theta \\ &\approx \frac{1}{M}\sum_{i=1}^{M} p(y^*|x^*, \theta_i), \end{aligned} \tag{2.35}$$

en este caso $\theta_i \sim p(\theta|\mathcal{D})$, para $i \in \{1, \ldots, M\}$, y $M > 0$ la cantidad de muestras obtenidas por SGLD.

2.3. Estimación de una densidad de probabilidad

La función de densidad de probabilidad de una variable aleatoria es un concepto sumamente relevante en estadística [51]. Pues, para X una variable aleatoria que toma valores en los reales, y f_X su función de densidad de probabilidad, se tiene que:

$$\mathbb{P}(a \leq X \leq b) = \int_a^b f_X(x)dx. \tag{2.36}$$

Es decir, permite conocer los valores en los cuales se puede encontrar X bajo una cierta probabilidad.

Suponer ahora que se tienen ciertos datos $\mathcal{D} = \{x_i\}_{i=1}^N$, con $N > 0$, independientes y provenientes de una misma variable aleatoria desconocida, la idea entonces, es poder estimar cuál es la densidad de probabilidad de tal variable aleatoria.

Una de las técnicas que se utiliza es considerar una familia paramétrica de distribuciones, $\{f_\theta : \theta \in \Theta\}$ con Θ el espacio de los parámetros. Y asumir que los datos provienen de alguna distribución dentro de la familia paramétrica, para luego buscar dentro de tal familia, cuál es la función de densidad más cercana a la de la variable aleatoria desconocida a través del estimador de máxima verosimilitud.

La función de verosimilitud, para $\theta \in \Theta$, corresponde a:

$$\begin{aligned} L(\theta) &= f_\theta(x_1, \ldots, x_N) \\ &= \Pi_{i=1}^N f_\theta(x_i). \end{aligned} \tag{2.37}$$

La última igualdad viene dada por la condición de independencia, y que provienen de la misma distribución. Finalmente el estimador de máxima verosimilitud, denotado como θ_{MLE}, corresponde a:

$$\theta_{\text{MLE}} = \underset{\theta \in \Theta}{\text{argmax}}\, L(\theta). \tag{2.38}$$

Para el caso de que la familia paramétrica corresponda al de las distribuciones normales, el estimador de máxima verosimilitud es conocido.

Proposición 2.6 *Sean $\{x_i\}_{i=1}^N$ datos pertenecientes a los reales, independientes e idénticamente distribuidos, provenientes de una distribución normal con media μ y desviación estándar σ desconocidas. Luego, el estimador de máxima verosimilitud corresponde a:*

$$\mu_{MLE} = \frac{1}{N}\sum_{i=1}^N x_i, \qquad \sigma_{MLE} = \sqrt{\frac{1}{N}\sum_{i=1}^N (x_i - \mu_{MLE})^2}. \tag{2.39}$$

Utilizando tales hipótesis, es posible estimar la función de densidad de los datos. Cabe señalar que los datos no necesariamente distribuyen normalmente, sin embargo, se puede ocupar este mismo criterio, pero con otra familia de distribuciones, por ejemplo, la de las distribuciones de Laplace, donde también es conocido el estimador de máxima verosimilitud.

Proposición 2.7 *Sean $\{x_i\}_{i=1}^N$ datos pertenecientes a los reales, independientes e idénticamente distribuidos, provenientes de una distribución de Laplace con media μ y escala b desconocidas. Luego, el estimador de máxima verosimilitud corresponde a:*

$$\mu_{MLE} = mediana(\{x_1, \ldots, x_N\}), \qquad b_{MLE} = \frac{1}{N}\sum_{i=1}^{N} |x_i - \mu_{MLE}|. \tag{2.40}$$

Si bien, explorar a través de la familia de distribuciones de Laplace, da más opciones que solamente considerar las distribuciones normales, sigue siendo restrictivo, pues, no existe la posibilidad de acercarse de buena manera a distribuciones con más de una moda. Para construir estimaciones de funciones de densidad más complejas existen varias técnicas, una de ellas son los flujos normalizadores.

2.3.1. Flujos normalizadores

Sean $\{x_i\}_{i=1}^N$, N datos, de dimensión $Q > 0$. La idea principal de los flujos normalizadores [26], [38] es crear distribuciones sofisticadas, a través de distribuciones simples. Para realizar lo anterior, se considera una variable aleatoria $u \in \mathbb{R}^{\mathbb{Q}}$, tal que su distribución $p(u)$ sea fácil de obtener muestras, por ejemplo una distribución uniforme o una Gaussiana. Luego, para construir la distribución compleja, se considera una función $F : \mathbb{R}^Q \to \mathbb{R}^Q$ no lineal e invertible (denominada flujo).

Se considera además que: $x = F(u)$, $u \sim p(u)$. Definiendo así $p_x(x)$ (la función de densidad de x), que a través de una función F lo suficientemente flexible se puede crear una distribución sofisticada.

Notar que muestrear de $p_x(x)$ es sencillo, pues basta muestrear de $p_u(u)$ y evaluar $x = F(u)$, con p_u la función de densidad de u Pero ahora, ¿Cómo se conoce $p_x(x)$? El teorema de cambio de variables da la respuesta de manera directa, pues dice que:

$$p_x(x) = p_u(u)|\text{det}J(F)(u)|^{-1}, \;\; u = F^{-1}(x). \tag{2.41}$$

Finalmente, la única restricción que queda, es computar de manera rápida $\text{det}J(F)(u)$. Para ello se han descubierto funciones F que permiten evaluar tal determinante de manera eficiente, que sean flexibles e invertibles, de manera de poder construir distribuciones más complejas. Una de las ventajas de los flujos normalizadores es que permite evaluar de manera exacta y eficiente la verosimilitud.

Se presentan a continuación algunos ejemplos de flujos normalizadores, para $Q = 1$:

- Biyección escalar afín: Para $\theta = (a, b) \in \mathbb{R}^2$, el flujo es $F_\theta(u) = au + b$. Este corresponde a uno de los ejemplos más simples, pues simplemente consiste en una transformación afín.
- Perturbaciones de orden superior: Es fácil darse cuenta que el flujo anterior es limitado, sin embargo, es posible hacerlo más flexible agregándole una perturbación de orden

superior [61], el flujo normalizador es:

$$F_\theta(u) = au + b + \frac{c}{1 + (du + e)^2}, \tag{2.42}$$

con $\theta = (a, b, c, d, e) \in \mathbb{R}^5$. Si $a > \frac{9}{8\sqrt{3}}cd$ y $d > 0$ el flujo es invertible.

- Splines: Estos flujos corresponden a funciones que son polinomios por trozos, o bien funciones racionales por trozos (division entre dos polinomios), existen $K + 1$ *knots* u_k, x_k que son simplemente puntos del plano cartesiano por donde pasa el flujo, es decir, que para cada $k \in \{1, \ldots, K + 1\}$, $h(u_k) = x_k$ y se define el flujo en (u_{k-1}, u_k) realizando una interpolación desde x_{k-1} a x_k a través de un polinomio o una función racional. A partir de esta idea se desarrolló un flujo conocido como *rational quadratic neural spline flow* [11] que permite construir flujos más flexibles, con respecto a los anteriormente mencionados, la principal característica de este flujo es que utiliza una red neuronal para su definición y tiene mejor rendimiento que las comúnmente usadas transformaciones afines sin la necesidad de utilizar métodos numéricos para la definición de su función inversa, es decir que existe una expresión conocida para tal función.

Capítulo 3

Antecedentes

Habiendo explicado las redes neuronales como tal, sus posibles aplicaciones, así como también cómo funciona la inferencia bayesiana, este Capítulo se centra en dar una explicación a las redes neuronales bayesianas (Sección 3.1), explicar cuál es el problema en este tipo de redes (Sección 3.2), y cuáles son los avances con respecto a tal problema (Sección 3.3).

3.1. Redes neuronales bayesianas

Las redes neuronales bayesianas (BNN por sus siglas en inglés) fueron propuestas por primera vez en 1987 por John Hopfield [7], luego en 1992 se estudió extensivamente cómo aplicar Monte Carlo e Inferencia Variacional en tales redes [35]. Este tipo de modelos son una versión probabilística de las redes neuronales. En particular, se puede entender como un modelo bayesiano parámetrico.

Considerando a $f : \mathbb{R}^Q \to \mathbb{R}^D$ una red neuronal con $W \in \mathbb{R}^P$ sus parámetros, $Q, D \in \mathbb{N}$ y $P \in \mathbb{N}$ la cantidad de parámetros. Usualmente se utiliza un prior Gaussiano isotropico, es decir,

$$W \sim \mathcal{N}(0, \sigma^2 I), \tag{3.1}$$

con $I \in \mathbb{R}^{P \times P}$ la matriz identidad y $\sigma^2 > 0$ un hiperpárametro.

Observación Cuando se utilice una distribución unidimensional para definir a una variable aleatoria multidimensional, significará que todas las componentes de la variable aleatoria siguen la misma distribución.

Por otro lado, para el problema de regresión, la verosimilitud que se utiliza es la siguiente,

$$p(y|W, x) = \mathcal{N}(f_W(x), \sigma^2), \tag{3.2}$$

con $f_W(x)$ la salida de la red neuronal, y $\sigma^2 > 0$.

Para un problema de clasificación, se utiliza una distribución categórica como verosimilitud, utilizando además la función softmax, es decir:

$$p(y|x, W) = \text{softmax}(f_W(x)). \tag{3.3}$$

Para entender las ventajas de la versión probabilística de las redes neuronales, es importante entender cuáles son las limitaciones de su versión determinista. Una de ellas es que se pueden engañar de manera sencilla, debido a que pueden presentar una alta confianza para imágenes irreconocibles [42], la manera de construir tales imágenes la realizan a través de algoritmos evolutivos [55] en donde se busca maximizar la probabilidad de que una imagen pertenezca a una clase perturbando, o añadiéndole ruido blanco a la imagen original.

La incorporación de una visión probabilística de este tipo de modelos, permite tener una estimación de la incertidumbre sobre las predicciones, es decir, una métrica que diga qué tan seguro está el modelo acerca de tales predicciones. Esto puede es sumamente útil en aplicaciones de alto riesgo, como lo son los vehículos autónomos, o diagnósticos médicos.

Otra ventaja de este tipo de redes es su aplicación en el área de aprendizaje activo [16], la cual se encarga de estudiar el problema en donde se presenta una determinada cantidad de datos etiquetados, y una de datos no etiquetados, estos últimos tienen un costo económico para poder darles una etiqueta, por ejemplo, financiar a un experto, por lo tanto, resulta necesario identificar cuáles son los datos que se deben etiquetar para proporcionar al modelo la mayor cantidad de información posible.

Como se comentó, una de las ventajas de darle una perspectiva probabilística a las redes neuronales, es poder calcular la incertidumbre. Esta incertidumbre es posible separarla en dos tipos:

- **Incertidumbre Epistémica:** Corresponde a la incertidumbre generada debido a una carencia en los datos, la cual es reducible recolectando más datos o mejorando el modelo.
- **Incertidumbre Aleatoria:** Corresponde a la incertidumbre generada por la aleatoriedad de los datos, la cual no es reducible mejorando el modelo.

Para el problema de clasificación donde el output es una distribución de probabilidad condicional con $C > 0$ posibles etiquetas. Es posible utilizar la teoría de la información para medir la incertidumbre a través del concepto de entropía de Shannon [50]. La entropía de Shannon mide la cantidad de incertidumbre o entropía en una variable aleatoria. Esto se puede hacer calculando la entropía de una distribución de probabilidad de la variable aleatoria, en particular, es posible ocuparla en la distribución de la predicción, es decir:

$$\mathbb{H}\left(y|x, \mathcal{D}\right) = -\sum_{c=1}^{C} \int p(y = c|x, W)p(W|\mathcal{D})dW \log\, p(y = c|x, \mathcal{D}). \tag{3.4}$$

Cuanto mayor sea la entropía, mayor será la incertidumbre asociada a la variable aleatoria, por ejemplo, en este caso, la entropía toma su valor máximo cuando la probabilidad para cada clase es la misma, mientras que toma el valor el valor mínimo de cero cuando existe

una clase con probabilidad 1. Por lo tanto, la entropía de Shannon es una medida útil para cuantificar la incertidumbre en una variable aleatoria, sin embargo, no logra distinguir entre incertidumbre epistémica y aleatoria.

Es posible medir la incertidumbre epistémica para un nuevo input x como la cantidad de información que ganaría tal modelo (o equivalentemente los parámetros) en caso de conocer el verdadero valor y asociado a ese input. La teoría de la información formaliza esta idea a través de la información mutua [50] entre el verdadero valor y y los parámetros del modelo:

$$I(y, w|x, \mathcal{D}) = H[y|x, \mathcal{D}] - \mathbb{E}_{p(W|\mathcal{D})} H[p(y|x, W)]. \tag{3.5}$$

En la práctica para un problema de clasificación, se utiliza la ecuación (3.4) y la identidad $p(y = c|x, \mathcal{D}) = \int p(y = c|x, W)p(W|\mathcal{D})dW$ para desarrollar la información mutua:

$$\begin{aligned} I(y, W|x, \mathcal{D}) = &- \sum_{c=1}^{C} \int p(y = c|x, W)p(W|\mathcal{D})dW \cdot \log \left(\int p(y = c|x, W)p(W|\mathcal{D})dW \right) \\ &+ \mathbb{E}_{p(W|\mathcal{D})} \left(- \sum_{c=1}^{C} p(y = c|x, W) \log\, p(y = c|x, W) \right), \end{aligned}$$

Finalmente se aproxima la información mutua considerando una aproximación de la distribución posterior con alguna de las técnicas que se comentaron en la Sección 2.2.1.

3.2. Problema

A principios de 2020, Florian Wenzel y compañía [57] ofrecen una muy buena introducción al problema que se abordará. Comentan que a pesar de los avances en el área de aprendizaje profundo bayesiano, hasta principios del 2020, no han existido publicaciones de desarrollos de estos modelos en la industria, por lo que es necesario conocer la razón de esto. En ese mismo estudio se muestra que las predicciones realizadas por redes neuronales bayesianas a través de muestras dadas por MCMC tienen peor rendimiento que una red determinista. Adicionalmente estudian la distribución posterior temperada, que simplemente se define así para $T > 0$:

$$p_T(\theta|\mathcal{D}) = \frac{p(\theta|\mathcal{D})^{\frac{1}{T}}}{\int p(\theta|\mathcal{D})^{\frac{1}{T}} d\theta}. \tag{3.6}$$

Se encuentra que al considerar la distribución posterior temperada con $T < 10^{-1}$ se obtiene mejor rendimiento que con $T = 1$, e incluso mejor que la red determinista. Que tal distribución con $T \ll 1$ (comúnmente $T < 10^{-1}$) tenga mejor rendimiento que la distribución posterior original se conoce en la literatura como cold posterior effect.

Los experimentos que utilizaron para ejemplificar lo anterior corresponden a una ResNet-20 [17] en CIFAR-10 [28] y una CNN-LSTM en unos datos de reseñas de películas de IMDB, además la inferencia se realizó a través de SGLD, por último, el prior considerado fue una distribución $\mathcal{N}(0, \sigma I)$, con $\sigma > 0$, e I la matriz identidad.

El problema de la distribución posterior temperada es que para $T < 1$, significa considerar $\frac{1}{T}$ replicas de los datos originales, y un prior proporcional a $p(\theta)^{\frac{1}{T}}$, lo cual para el caso Gaussiano es equivalente a considerar a multiplicar la desviación estándar por T. Esto genera una evidencia fuerte a modelos individuales, de hecho, cuando $T \to 0$, la masa de la posterior se concentra en la solución dada por el MAP (máximo a posteriori), además $T = 1$ corresponde a la verdadera posterior, por lo que al tener un mejor rendimiento con $T < 1$, significa que puede existir un problema con la elección de la verosimilitud, el método de inferencia, o la definición del prior. De hecho una de las principales conclusiones de [57] fue que se necesitaba más estudio de los priors en redes neuronales bayesianas.

3.3. Estado del arte

Luego de tal paper, surgieron varios estudios con respecto a la elección del prior en redes neuronales bayesianas, por lo que esta sección se encarga de estudiar el estado del arte de este problema.

Vicent Fortuin y compañia ofrecen una buena primera aproximación a tal problema [14], tienen como objetivo estudiar diferentes opciones al tradicional prior considerado en la literatura, que corresponde simplemente al prior Gaussiano isotropico. Debido a la no interpretabilidad de los parámetros, la idea de este paper es entrenar la red determinista e inspirarse en la distribución de los parámetros ya entrenados, si bien cada dato no tiene una distribución, pues son valores puntuales, estudian el histograma de tales parámetros en cada capa, o bien la correlación entre los kernels de cada capa convolucional.

Muestran que al incorporar tal información en los priors, se mitiga el cold posterior effect para el caso de los MLP (ocupando una distribución Laplace o t-student como prior), sin embargo, aumenta en el caso de las CNN (al ocupar un prior Gaussiano con matriz de covarianza diferente a la identidad). Para mostrar lo anterior se utilizan arquitecturas como MLP, CNN, y ResNet, y MNIST [33], FashionMNIST [58] y CIFAR-10 [28] como conjunto de datos. Destacan que no se espera encontrar un prior universal que tenga mejor rendimiento en todos los datos y arquitecturas, además que el prior Gaussiano isotropico no es óptimo, en el sentido que tiene peor precisión que la versión determinista, por lo que vale la pena estudiar otras opciones.

Con respecto al cold posterior effect, no obtuvieron conclusiones claras, pues si bien se eliminó al utilizar una distribución de Laplace o t-student como prior para el MLP, en el caso de la CNN, aumentó tal efecto al utilizar el prior Gaussiano con matriz de covarianza diferente a la identidad, sin embargo, no se puede descartar que exista otro prior para tal arquitectura que no presente tal efecto. Cabe destacar que para realizar la inferencia utilizaron Stochastic Gradient Markov Chain Monte Carlo (SG-MCMC), con el objetivo de escalar a conjunto de datos más grandes. Comentan además que si la mala elección del prior es la causa del cold posterior effect, se puede considerar la contra reciproca de ese argumento, y de tal manera, en caso de no observar el cold posterior effect, esto significaría la elección de un mejor prior.

Por último, muestran un resumen de lo realizado hasta la fecha, donde se señala que no se había utilizado la idea de analizar los parámetros ya entrenados de forma determinista e

incorporar tal información en la creación de la distribución a priori. También se menciona que si bien se han propuesto otros priors, estos son utilizados para problemas específicos, se comenta además de la aproximación de la posterior a través de dropout [15], la cual consiste en una técnica de regularización, pero que sin embargo no está bien calibrada [12].

Es posible notar que tanto en [57] como en [14] se utilizan técnicas de inferencia que permitan escalar a grandes volúmenes de datos, y a partir de ahí estudiar la BNN respectiva. Es por ello que en [21] se estudió la distribución posterior de una BNN utilizando como técnica de inferencia Hamiltonian Monte Carlo (HMC) [3].

Las principales contribuciones de [21] fueron las siguientes:

- Las BNNs obtienen mejor rendimiento que su versión determinista. Considerando como métricas la precisión, la log verosimilitud y el error esperado de calibración. Y que a diferencia de los otros estudios, no es necesario considerar la posterior temperada para mejorar el rendimiento de la BNN.
- Otro punto relevante que muestran es que el rendimiento de la BNN es robusto a la elección de la escala del prior, y relativamente similar para un prior Gaussiano isotropico, mezcla de Gaussianas y priors logísticos. Por último señalan que las muestras obtenidas a través HMC son más cercanas a las obtenidas por SGLD (y deep ensembles (promedio de varios modelos diferentes basados en redes neuronales)) que las muestras dadas por inferencia variacional.

Los modelos utilizados fueron una ResNet-20 y una CNN-LSTM con los conjuntos de datos CIFAR-10 e IMDB respectivamente, exactamente la misma configuración que en [57].

Notar que este articulo es controversial, pues se obtienen resultados que difieren a los obtenidos por Vicent Fortuin en [14], sin embargo, existe una diferencia sustancial, la cual es la manera que se realiza la inferencia, pues en este caso se utiliza HMC. Sin embargo HMC no es una técnica práctica y para poder realizar los experimentos correspondientes se utilizaron 512 TPUs (unidades de procesamiento tensorial).

Por otro lado, Lorenzo Noci y Kevin Roth en [43] profundizan en el estudio del cold posterior effect estudiando sus posibles causas. Comentan que una de ellas es la mala especificación del prior.

Notar que normalmente en inferencia bayesiana, la cantidad de parámetros es pequeña en comparación con la cantidad de datos, por lo que el prior es rápidamente dominado por los datos. Sin embargo, en las redes neuronales bayesianas, no sucede necesariamente lo anterior, pues en estos modelos se acostumbra a tener una gran cantidad de parámetros cercana o mayor a la cantidad de datos [24], por lo que sigue teniendo influencia la elección del prior en la distribución posterior. Muestran además que el cold posterior effect está fuertemente vinculado con la elección del prior.

Otra de las posibles causas que mencionan son las siguientes:

- Método de inferencia: Concluyen que SG-MCMC es suficientemente preciso, o no es una causa del cold posterior effect.

- Mala elección de la verosimilitud: Concluyen que es suficiente, pero no necesario para que ocurra el cold posterior effect.

Utilizan el mismo método de inferencia que en [57], una ResNet-20 como modelo, CIFAR-10 y SVHN como conjunto de datos, y también se explora el modelo CNN-LSTM con el conjunto de datos IMDB, al igual que en [57].

En [13] Vincent Fortuin ofrece un estudio completo acerca de los priors en el área de aprendizaje profundo bayesiano. En particular detalla las siguientes distribuciones a priori:

- Gaussiano isotropico: Estudiado anteriormente en [14], [57] donde se comenta el cold posterior effect que presenta este prior, lo cual sugiere una mala especificación del mismo.
- Matrix-valued Gaussian: Una simple extensión de la distribución Gaussiana que permite correlación entre los parámetros. Su función de densidad es la siguiente,

$$p(W) = \mathcal{MV}(M, U, V) = \frac{\exp\left(-\frac{1}{2}\text{tr}[V^{-1}(W-M)^T U^{-1}(W-M)]\right)}{[(2\pi)^p \text{det}U \text{det}V]^{\frac{n}{2}}} \tag{3.7}$$

  con M la media la matriz de medias, U y V las covarianzas de filas y columnas respectivamente, y tr$[\cdot]$ el operador traza, y (n, p) corresponde a la dimensión de la matriz W. Este prior ha demostrado aumentar el rendimiento con respecto al Gaussiano isotropico en diferentes problemas utilizando inferencia variacional.
- Laplace: Se comenta acerca de la distribución de Laplace como prior y cómo en el caso de un perceptrón multicapa no posee el cold posterior effect, según lo estudiado anteriormente en [14].
- Horseshoe prior: El cual es para cada peso w,

$$p(w) = \mathcal{N}(0, \tau^2 \sigma_w^2) \tag{3.8}$$
$$p(\tau) = \mathcal{C}^+(0, b_0) \tag{3.9}$$
$$p(\sigma_w) = \mathcal{C}^+(0, b_1) \tag{3.10}$$

  con b_0 y b_0 reales positivos y $\mathcal{C}^+$ corresponde a la distribución half-Cauchy. Este prior se utiliza en áreas como la genómica.

Por último, cabe señalar que los priors que se mostraron anteriormente, y como es usual, no utilizan los datos para su definición. Sin embargo, existe un enfoque diferente que sí los utiliza, esto es conocido en la literatura como Bayes empírico (*empirical Bayes*).

En [14] se realizó esto, donde manualmente se escogió el prior al observar la distribución de los parámetros ya entrenados a través de un algoritmo de descenso del gradiente. Otro enfoque es aproximar la log verosimilitud marginal, que corresponde simplemente a log $p(\mathcal{D})$, a través de un método denominado Laplace-Generalized-Gauss-Newton [20], para posteriormente optimizar los hiperparámetros del prior, y la arquitectura de la red.

En conclusión, es posible notar que no ha sido posible encontrar un prior universal que sirva para cualquier arquitectura. La idea de ahora en adelante es poder explotar el argumento utilizado en [14], y definir diferentes priors a través de esa idea, considerar además otro prior basado en la solución de la red determinista y evaluarlos en diferentes arquitecturas y conjuntos de datos.

Capítulo 4

Metodología

4.1. Datos y modelos

En esta sección se presentarán los datos y modelos para el desarrollo de los experimentos.

4.1.1. Datos

Los conjuntos de datos que se utilizarán serán MNIST, FashionMNIST y CIFAR-10.

MNIST es un conjunto de datos de imágenes en blanco y negro de dígitos escritos a mano [33], con su etiqueta correspondiente al dígito que se escribió. Las imágenes están codificadas a través de una matriz de 28×28 y sus elementos están en $\{0, 1, 2, \ldots, 255\}$ dependiendo de la intensidad del color negro.

Se presenta en la Figura 4.1 un ejemplo para cada clase.

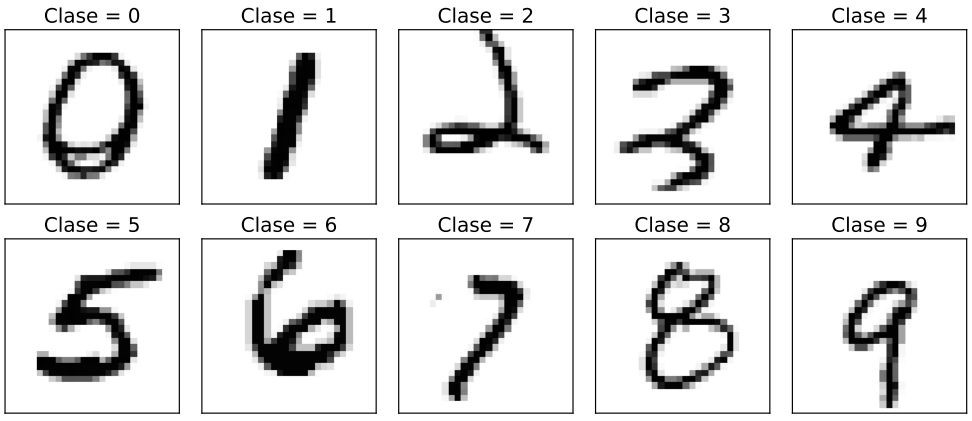

Figura 4.1: Ejemplos de MNIST. Elaboración propia.

Por otro lado, FashionMNIST es un conjunto de datos de imágenes en blanco y negro

de diferentes artículos de vestir [58], con su etiqueta correspondiente al tipo de articulo. Las imágenes también están codificadas a través de una matriz de 28×28 y sus elementos están en $\{0, 1, 2, \ldots, 255\}$ dependiendo de la intensidad del color negro.

Se presenta en la Figura 4.2 un ejemplo para cada clase.

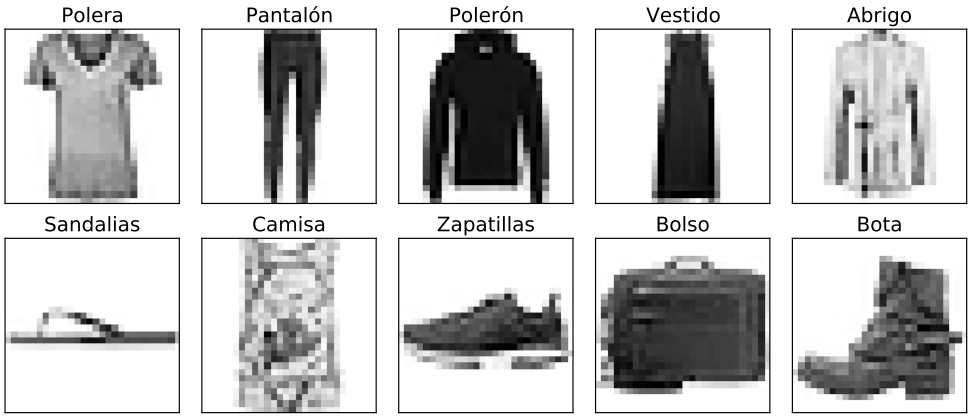

Figura 4.2: Ejemplos de FashionMNIST. Elaboración propia.

Finalmente CIFAR-10 es un conjunto de datos de imágenes a color de diferentes tipos [28], en este caso, estas se representan a través de un tensor de $32 \times 32 \times 3$ donde la primera, segunda y tercera matriz corresponde a la intensidad del color rojo, verde y azul respectivamente, y sus elementos están en $\{0, 1, 2, \ldots, 255\}$ dependiendo de la intensidad del color respectivo.

Se presenta en la Figura 4.3 un ejemplo para cada clase.

Figura 4.3: Ejemplos de CIFAR-10. Elaboración propia.

4.1.2. Modelos

Los modelos que se utilizarán será un perceptrón multicapa y una red neuronal convolucional. Se describen con más detalle a continuación:

El perceptrón multicapa, de ahora en adelante MLP, la capa de entrada de esta red depende de la dimensión del input. Como se está trabajando con imágenes, y debido a la arquitectura, simplemente se cambia la dimensión del input de 28×28 a 784 en el caso de MNIST y FashionMNIST y de $32 \times 32 \times 3$ a 3072 en el caso de CIFAR-10. Luego hay 3 capas ocultas de 50 neuronas cada una, con ReLU como función de activación, y finalmente la capa de salida con la cantidad de neuronas igual a la cantidad de clases de los datos (10 en todos los conjuntos de datos que se están considerando).

La red convolucional, de ahora en adelante CNN, al igual que la red anterior, la capa de entrada depende de la dimensión del input. Luego, hay una capa convolucional, con 32 kernels de 3×3 con un stride y un padding de 1, con ReLU como función de activación y un average pool, de 2×2, con un stride de 2, y un padding de 1. Luego existe una capa convolucional igual a la anterior, pero con 64 kernels, posteriormente se aplana el tensor, y se utiliza como entrada de un perceptrón multicapa con una capa oculta de 256 neuronas, con ReLU como función de activación y una capa de salida con la cantidad de neuronas igual a la cantidad de clases de los datos (10 en todos los conjuntos de datos).

4.2. Propuestas y contribuciones

Con el objetivo de encontrar una distribución a priori para redes neuronales bayesianas que supere a lo anteriormente estudiado (es decir, que logren tener un mejor rendimiento que la versión determinista de la red, en cuanto a precisión, y calibración), se proponen dos enfoques diferentes, pero ambos basado en los parámetros ya entrenados de forma determinista.

4.2.1. Propuesta 1

Debido a la cantidad de parámetros que existen en las redes neuronales, y la no interpretabilidad de estos, es que se hace difícil tener una creencia a priori acerca de cómo distribuyen [14]. Por lo tanto, se propone estudiar en primera instancia, a los parámetros de la red determinista luego de ser entrenada. Para así definir un prior similar a lo ya realizado en [14], sin embargo se explota aún más su argumento, donde no solamente se toma como inspiración la distribución de los parámetros ya entrenados, sino que se utiliza directamente para la definición del prior. Para ello se realizará lo siguiente:

- Inspección de los parámetros: Se visualiza la distribución de los parámetros por capa entrenados de forma determinista.
- Máxima verosimilitud para la definición del prior: Se utilizan los parámetros entrenados de forma determinista por cada capa para la creación de un prior ajustando una distribución Gaussiana y de Laplace a través de máxima verosimilitud.
- Flujos normalizadores para la definición del prior: Se extiende la idea anterior utilizando flujos normalizadores.

Para explicar la idea de la inspección de los parámetros, se utilizará un perceptrón multicapa, sin embargo se generaliza fácilmente a otras arquitecturas. Sea entonces f_W un perceptrón multicapa, con $W = \{(W_i, b_i)\}_{i=1}^{L+1}$ sus parámetros, $L \in \mathbb{N}$ la cantidad de capas ocultas, y $\mathcal{D}$ un conjunto de datos, se entrenará esta red de forma determinista, mediante algún algoritmo de descenso del gradiente, obteniendo así $\hat{W} = \{(\hat{W}_i, \hat{b}_i)\}_{i=1}^{L+1}$ los parámetros W ya entrenados. Se graficará un histograma para cada $\hat{W}_i$ con $i \in \{1, \ldots, L+1\}$, y adicionalmente al histograma, se realizará un gráfico Q-Q, este tipo de gráficos son utilizados para comparar los cuantiles de dos distribuciones, se utilizarán en este caso para comparar la distribución de estos pesos con respecto a una distribución Gaussiana y una distribución de Laplace. Esto con el objetivo de visualizar a qué tipo de distribución está más cercana la distribución de los pesos de cada capa. Habiendo realizado esto, se proponen los siguientes priors, todos basados en la misma idea descrita al inicio de esta propuesta:

- Caso Gaussiano: Para cada $w_j \in \hat{W}_i$ con $i \in \{1, \ldots, L+1\}$ y $j \in \{1, \ldots, K_i K_{i-1}\}$ definidos en la sección anterior, se hará el siguiente supuesto

$$w_j \sim \mathcal{N}(\mu_i, \sigma_i). \tag{4.1}$$

  y además $(w_j)_{j=1}^{K_i K_{i-1}}$ serán independientes. De esta manera, la propuesta consiste en encontrar un modelo que haya generado tales muestras a través de máxima verosimilitud, y ocupar tal distribución como prior para la versión bayesiana de la red, es decir:

$$p(W_i) \sim \mathcal{N}(\hat{\mu_i}, \hat{\sigma_i}), \tag{4.2}$$

  con $\hat{\mu_i}$, $\hat{\sigma_i}$ los estimadores de máxima verosimilitud.

- Caso Laplace: También se propone realizar lo mismo que antes, pero ahora considerando que,

$$w_j \sim \mathcal{L}(\varphi_i, \delta_i). \tag{4.3}$$

  Y $(w_j)_{j=1}^{K_i K_{i-1}}$ también independientes, de este modo,

$$p(W_i) \sim \mathcal{L}(\hat{\varphi}_i, \hat{\delta}_i), \tag{4.4}$$

  con $\hat{\varphi}_i$, $\hat{\delta}_i$ los estimadores de máxima verosimilitud y $\mathcal{L}$ la distribución de Laplace.

- Caso Híbrido: Se propone un esquema mixto con estas dos distribuciones, para ello se considera $\hat{l}_{\mathcal{N}}(\hat{\mu_i}, \hat{\sigma_i}; W_i)$, y $\hat{l}_{\mathcal{L}}(\hat{\varphi_i}, \hat{\delta_i}; W_i)$ las funciones de verosimilitud de las distribuciones normal y de Laplace respectivamente, con $\hat{\mu_i}$, $\hat{\sigma_i}$ y $\hat{\varphi_i}$, $\hat{\delta_i}$ los estimadores de máxima verosimilitud de la distribución normal y de Laplace respectivamente. Luego,

$$p(W_i) \sim \mathcal{N}(\hat{\mu_i}, \hat{\sigma_i})\mathbf{1}_{\{\hat{l}_{\mathcal{N}}(\hat{\mu_i}, \hat{\sigma_i}; W_i) > \hat{l}_{\mathcal{L}}(\hat{\varphi_i}, \hat{\delta_i}; W_i)\}} + \mathcal{L}(\hat{\varphi_i}, \hat{\delta_i})\mathbf{1}_{\{\hat{l}_{\mathcal{N}}(\hat{\mu_i}, \hat{\sigma_i}; W_i) \leq \hat{l}_{\mathcal{L}}(\hat{\varphi_i}, \hat{\delta_i}; W_i)\}}, \tag{4.5}$$

  con $\mathbf{1}$ la función indicatriz. Básicamente se ocupa como distribución a priori a la distribución que obtuvo más verosimilitud.

- Flujos Normalizadores: Con la idea de poder adaptarse a distribuciones más complejas, se utilizará un flujo normalizador. Para los pesos de W_i, $i \in \{1, \ldots, L+1\}$, (siguiendo

la misma notación que se ha utilizado en este Capítulo) se asume nuevamente que son independientes y que vienen de una distribución desconocida. Sea F_θ un flujo normalizador con θ sus parámetros, una distribución inicial $p_u(u)$ con $u \in \mathbb{R}$, pues se quiere aproximar una distribución unidimensional. Luego, el prior a considerar es el siguiente:

$$p(W_i) \sim F_{\hat{\theta}}(u) \tag{4.6}$$

$$u \sim p_u(u), \tag{4.7}$$

con $\hat{\theta}$ el parámetro que maximice la verosimilitud $p_x(W_i|\theta)$.

4.2.2. Propuesta 2

Esta propuesta sigue una idea similar a la propuesta anterior, sin embargo, es aún más específica con respecto a cómo se utilizan los parámetros ya entrenados a través de la red determinista. Se denota como $W = \{w_i\}_{i=1}^P$ a los parámetros de la red bayesiana y $\hat{W} = \{\hat{w}_i\}_{i=1}^P$ como los parámetros ya entrenados a través de la red determinista, $P \in \mathbb{N}$ corresponde a la cantidad de parámetros de la red.

La distribución a priori propuesta para $w \in W$ será simplemente,

$$p(w) \sim \mathcal{N}(\hat{w}, \sigma) \tag{4.8}$$

con $\sigma > 0$. Es decir, la distribución a priori para cada peso, corresponde a una distribución normal centrada en la solución dada por la red determinista de tal peso, σ será la desviación estándar del prior, que corresponde a un hiperparámetro a estudiar.

4.2.3. Contribuciones

Las principales contribuciones se detallan a continuación:

- Diseño de prior que mejoran la versión determinista de la red, en cuanto a precisión y log verosimilitud.
- Estudio de las muestras de la posterior para los priors propuestos encontrando resultados relevantes como que a pesar de que el prior propuesto se base en la solución de la red determinista no se concentre ahí.
- Demostración de un uso práctico de lo propuesto, a través de datos de salud, e ilustración de la incertidumbre de las predicciones utilizando métodos de reducción de la dimensionalidad.
- Se proponen priors que explotan el argumento desarrollado en [14] y se estudia su rendimiento en diferentes conjuntos de datos y modelos.

4.3. Experimentos

4.3.1. Implementación propuesta 1

El primer experimento consiste simplemente en implementar la propuesta 1, y a partir de ello poder analizar la idea.

Para estudiar los parámetros ya entrenados de la red determinista se considera que se tiene una red neuronal y un conjunto de datos, la metodología del entrenamiento es la siguiente, dados los datos $\mathcal{D} = \{\mathcal{D}_{\text{train}}, \mathcal{D}_{\text{test}}\}$. La red se entrenará a través del algoritmo adam, durante 30 épocas, se ocupará un conjunto de validación, $\mathcal{D}_{\text{val}}$ correspondiente al 10 % de los datos de entrenamiento para la elección de los siguientes hiperparámetros: la cantidad de épocas, el learning rate $\in \{0.01, 0.001, 0.0001\}$ y el tamaño del batch $\in \{64, 128, 256, 512\}$, de acuerdo a la precisión que obtengan en $\mathcal{D}_{val}$. Finalmente se visualizan los parámetros según lo comentando anteriormente, considerando los mejores hiperparámetros encontrados.

Se utilizan los datos y modelos descritos al inicio de este Capítulo.

La implementación de los tres primeros priors descritos en esta propuesta fue directa, pues el estimador de máxima verosimilitud tanto para el caso Gaussiano como para el caso de una distribución de Laplace, tienen forma cerrada, tal como se explicó en las proposiciones 2.6 y 2.7.

Para la implementación del prior de esta propuesta que considera un flujo normalizador se utilizará $\mathcal{N}(0, 1)$ como distribución inicial, mientras que el flujo es el rational quadratic neural spline flow [11] (se comentó con más detalle en la sección 2.3.1), el cual se entrenó a través del algoritmo adam con un learning rate igual a 0.5.

4.3.2. Evaluación de los priors

Habiendo definido los priors a utilizar en las secciones anteriores. Se dará una primera aproximación sobre el estudio de estos, donde simplemente se propone obtener muestras de la posterior del modelo a través de MCMC, y luego evaluar el modelo, y por consiguiente el prior, a través de la precisión, la log verosimilitud, y el error esperado de calibración. Además se estudiará el cold posterior effect.

A modo de síntesis, las distribuciones a priori a considerar se muestran en la Tabla 4.1, donde además se muestra el nombre con el cual se hará referencia de ahora en adelante.

El nombre señalado en la Tabla 4.1 de los priors definidos en la la Subsección 4.2.1 siguen el mismo orden en el cual fueron descritos en tal Subsección. Con respecto al prior determinista, se considerará un $\sigma \in \{0.1, 0.01\}$ y se identificarán como Determinista $\sigma = 0.1$ y Determinista $\sigma = 0.01$.

Antes de describir cómo se evaluarán los priors se detalla la metodología de la inferencia: Se denotará como f_W a la red neuronal, W sus parámetros, N como la cantidad de datos y a

Nombre	**Distribución**
Normal	$\mathcal{N}(0,1)$
Laplace	$\mathcal{L}(0,1)$
MLE Normal	Definida en 4.2.1
MLE Laplace	Definida en 4.2.1
MLE Híbrido	Definida en 4.2.1
NF	Definida en 4.2.1
Determinista	Definida en 4.2.2

Tabla 4.1: Priors a considerar. $\mathcal{N}$ y $\mathcal{L}$ corresponden a la distribución Normal y de Laplace respectivamente.

B como el tamaño del batch. La técnica de inferencia a utilizar será SGLD [56]. Se utilizarán $28\lceil\frac{N}{B}\rceil$ muestras de calentamiento, es decir muestras que no se utilizarán para la aproximación de la distribución posterior, y $2\lceil\frac{N}{B}\rceil$ muestras que sí. La búsqueda de hiperparámetros se hará a través de GridSearch, considerando un conjunto de validación proveniente del 10 % de los datos de entrenamiento. Los hiperparámetros serán el step size $\in \{8\cdot 10^{-7}, 10^{-6}, 1.2\cdot 10^{-6},\}$ y el tamaño del batch $\in \{64, 128, 256, 512\}$. La elección de los hiperparámetros se hará a través de la precisión de los modelos considerando como la clase a predecir de un input x en el conjunto de validación, como el argmax de,

$$\frac{1}{2\lceil\frac{N}{B}\rceil}\sum_{i=1}^{2\lceil\frac{N}{B}\rceil}\text{softmax}(f_{W^{(i)}}(x)), \tag{4.9}$$

con $W^{(i)}$, $i \in \{1,\ldots,2\lceil\frac{N}{B}\rceil\}$ las muestras obtenidas a través de SGLD. Notar que la ecuación (4.9) es una aproximación de la distribución de la predicción $p(y|x,\mathcal{D})$. Finalmente, para cada prior propuesto, se realiza la inferencia a través de la ecuación (4.9) considerando los hiperparámetros según el criterio recién definido.

Con el fin de ver la evolución de los modelos se graficarán las métricas descritas anteriormente para cada aproximación de la posterior predictiva: $\frac{1}{N}\sum_{i=1}^{N}\text{softmax}(f_{W_i}(x))$ con $N \in \{1,\ldots,10^3\}$. En este caso se considerarán las muestras de calentamiento, con el objetivo de ver qué tan rápido aprenden los modelos.

Finalmente y por lo discutido en la Sección 3.2, se estudiará la distribución posterior temperada. Se realizará la inferencia tal como se detalló anteriormente, salvo que el tamaño del batch será de 256, esto para que en todos los modelos existan la misma cantidad de muestras. La distribución temperada se estudiará como es usual en la literatura con un $T \in \{10^{-3}, 10^{-2}, 10^{-1}, 1\}$. Se realizará un gráfico de la precisión, la log verosimilitud y el error esperado de calibración en función de la temperatura, para el prior Normal, Laplace, y el prior que tenga mejor rendimiento dentro de los propuestos, esto con el fin de tener una visualización más clara.

4.3.3. Estudio de las muestras de la posterior

Con el objetivo de ver cómo son las muestras de la posterior con respecto a la solución dada por la red determinista, se graficará un histograma de las muestras, junto a la solución de la red determinista.

Como todos los modelos considerados tienen más de 44700 parámetros, se considerarán 4 pesos por cada capa como ejemplos, para graficar los respectivos histogramas de las muestras de su posterior, junto a la solución dada por la red determinista. Cabe señalar que se considerarán los hiperparámetros según lo realizado en el experimento anterior y se realizará el experimento para los priors Normal, NF y Determinista 0.01.

4.3.4. Aplicación en salud

Como se ha comentado anteriormente, una de las ventajas de las redes neuronales bayesianas es poder estimar la incertidumbre de este tipo de modelos.

Debido a lo anterior, se ocupará a modo de ejemplo un conjunto de datos, correspondiente a información de 975 pacientes etiquetados de acuerdo a la distrofia muscular que posean [53], donde existen 10 opciones.

Dentro del conjunto de datos, existen 70 características, todas ellas del tipo ordinal, con valores en $\{0, 1, 2, 3, 4, 5\}$. La mayoría presenta un considerable porcentaje de valores nulos, como se puede ver en el gráfico A.1, por lo que se ocupará una estrategia simple de imputación a través de la media de la característica para no tener tales valores nulos. Es decir, que cada valor nulo, se reemplaza a través de la media de su respectiva característica.

Teniendo la limpieza de los datos lista, se dividió este conjunto de datos, en un 15 % para la evaluación del modelo, y en un 85 % para entrenamiento, del cual se utiliza un 15 % para la elección de hiperparámetros.

Como corresponden a datos tabulares, se ocupa el MLP. Además se utiliza la versión bayesiana de tal modelo para realizar el entrenamiento, con el prior Determinista $\sigma = 0.01$. Se ocupa la misma estrategia de selección de hiperparámetros que se utiliza en el experimento de la evaluación de los priors, descrito en la Subsección 4.3.2 y se evalúa el modelo simplemente con la precisión.

Además se propone, para las predicciones del conjunto de prueba, ocupar la información mutua (descrita en la ecuación 3.5) como métrica para la incertidumbre.

Con el objetivo de entender cómo se realizan las predicciones, se realizará un gráfico de la matriz de confusión. Por último, para visualizar a la incertidumbre del modelo, se reduce la dimensionalidad de los datos, a través de la técnicas UMAP [36] y t-SNE [52], se etiqueta en el gráfico la clase a la cual pertenece cada dato y se marca con una estrella a los 30 datos de prueba con más incertidumbre.

Capítulo 5

Análisis y resultados

En este Capítulo, se presentarán los resultados de los experimentos descritos en el Capítulo anterior. Una vez presentados, se procederá a realizar un análisis detallado de dichos resultados.

5.1. Implementación propuesta 1

En primera instancia se mostrarán los graficos Q-Q que permiten estudiar los parámetros de la red determinista, posteriormente se mostrarán los resultados de los priors basados en la aproximación por máxima verosimilitud, y finalmente se mostrará un gráfico de cómo el flujo normalizador define el prior.

5.1.1. Inspección de los parámetros

En la Figura 5.1 se pueden ver los resultados para el caso del perceptrón multicapa con el conjunto de datos MNIST.

Por otro lado, en la Figura 5.2 se pueden ver los resultados para el caso de la red neuronal convolucional en el mismo conjunto de datos. Y en el Anexo A.2 están las Figuras correspondientes al resto de conjunto de datos, tanto para el MLP, como para la CNN.

Con respecto al perceptrón multicapa, se puede ver en la Figura 5.1 que los parámetros entrenados de la primera capa, siguen más bien una distribución de Laplace, concentrándose entorno a 0, para el resto de capas no se ve una tendencia clara. Para el conjunto de datos FashionMNIST y CIFAR-10, se puede notar en la Figura A.2 y A.3 a través de los QQ-plots, que en todas las capas los parámetros siguen más bien una distribución normal.

Con respecto a la red convolucional, se puede notar en las Figuras 5.2, A.4 y A.5 que la primera capa sigue más bien una distribución normal, mientras que la segunda capa está más cerca de una distribución de Laplace, para el resto de capas no se puede observar una

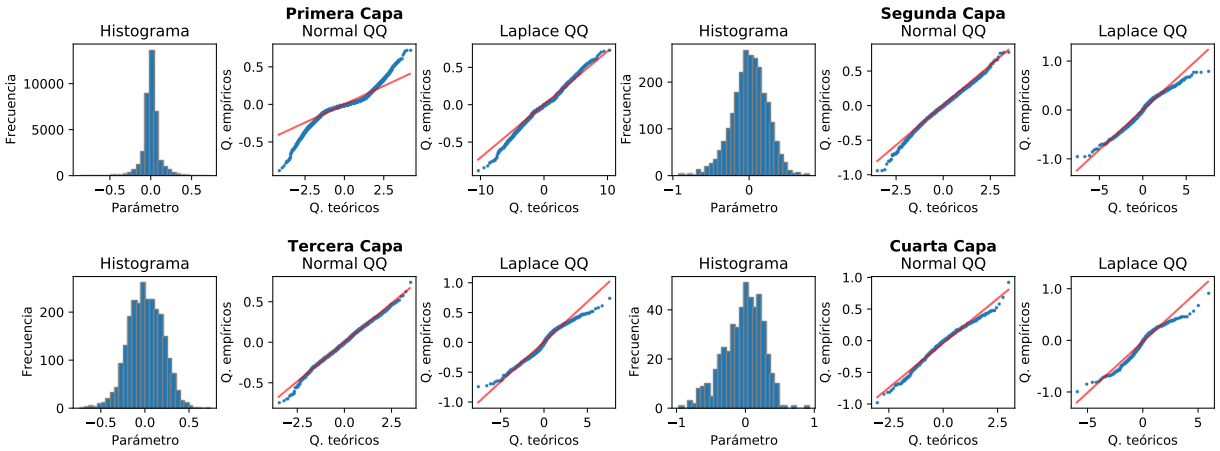

Figura 5.1: Histograma de los parámetros ya entrenados del perceptrón multicapa, separados por capa, en MNIST. Elaboración propia.

tendencia clara.

En resumen, se puede notar que en el caso de MNIST para el perceptrón multicapa y todos los conjuntos de datos de la red convolucional, la capa con más parámetros, está más cercana a una distribución de Laplace. Sin embargo, no es posible establecer con claridad para ninguna de las dos arquitecturas, que la distribución de los parámetros entrenados sigan una distribución en específica.

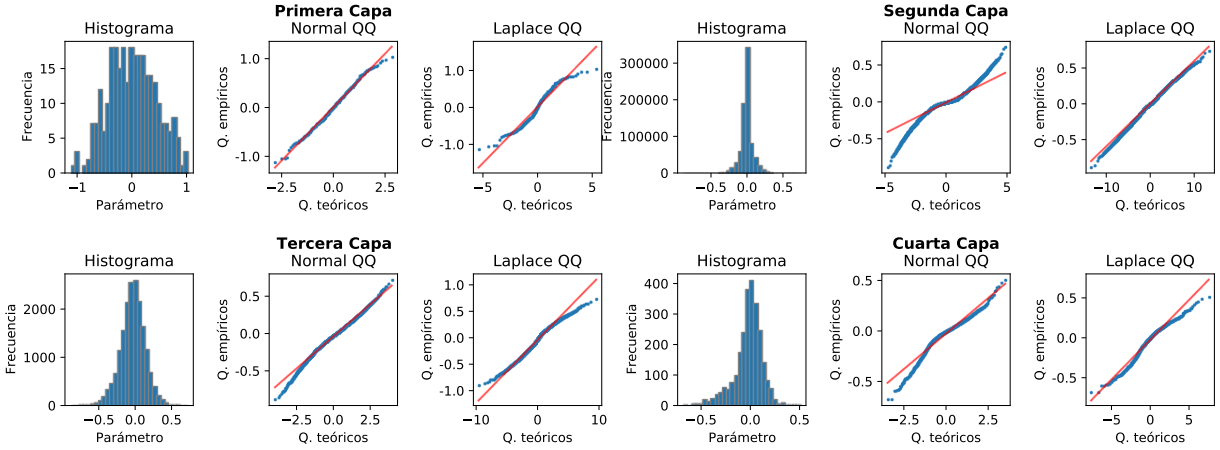

Figura 5.2: Histograma de los parámetros ya entrenados de la CNN, separados por capa, en MNIST. Elaboración propia.

5.1.2. Máxima verosimilitud para la definición de la distribución a priori

En las Tablas 5.1 y 5.2 se muestran los resultados tanto para el caso Gaussiano como para el de la distribución de Laplace, el caso híbrido se obtiene directamente de estas Tablas. La log verosimilitud corresponde a la log verosimilitud promedio. Como se comentó en 5.1.1 los

parámetros de la primera capa del MLP en el conjunto de datos MNIST, siguen más bien una distribución de Laplace, lo cual es coincidente con la estimación de máxima verosimilitud, pues para estos datos, se tiene que la máxima verosimilitud considerando una distribución de Laplace, es de 1.024 versus 0.865 en el caso de la distribución Normal. Con respecto al resto de conjunto de datos, pero aún considerando el MLP, se obtiene una verosimilitud mayor a través de la distribución normal.

Para la CNN, en el caso de la primera capa, se obtiene una mayor verosimilitud para la distribución normal, para la segunda capa también en los datos MNIST y CIFAR-10, sin embargo, acá se hubiera esperado una verosimilitud mayor de la distribución Laplace, pues en las Figuras 5.2 y A.5 se puede notar claramente, que los parámetros de la segunda capa siguen mejor una distribución de Laplace. Para el resto de capas, la distribución de Laplace, es la que obtiene más verosimilitud.

Si bien, la mayoría de los resultados obtenidos es coincidente con los gráficos Q-Q, se pudo notar que hay ciertos casos donde a través de los gráficos, se podía ver una tendencia clara hacia una cierta distribución, pero al calcular la máxima verosimilitud, se obtenía el resultado contrario. La principal posible causa de lo anterior, es que al comparar distribuciones de familias distintas, no son posible compararlas a través de la verosimilitud, justamente porque pertenecen a distribuciones diferentes.

		Log verosimilitud				
Modelo	**Datos**	**Capa 1**	**Capa 2**	**Capa 3**	**Capa 4**	**Promedio**
MLP	Mnist	0.865	0.232	0.040	-0.151	0.246
MLP	Fashion Mnist	1.125	0.257	0.238	0.090	0.427
MLP	Cifar10	1.677	0.431	0.403	0.310	0.705
CNN	Mnist	-0.579	0.354	1.046	0.523	0.336
CNN	Fashion Mnist	-0.647	0.078	0.712	-0.032	0.028
CNN	Cifar10	0.059	0.853	1.579	0.772	0.816

Tabla 5.1: Resultados de estimadores de máxima verosimilitud para el MLP y la CNN, caso Gaussiano.

		Log verosimilitud				
Modelo	**Datos**	**Capa 1**	**Capa 2**	**Capa 3**	**Capa 4**	**Promedio**
MLP	Mnist	1.024	0.179	0.016	-0.195	0.256
MLP	Fashion Mnist	1.115	0.197	0.189	0.059	0.390
MLP	Cifar10	1.647	0.360	0.332	0.230	0.642
CNN	Mnist	-0.656	0.343	1.203	0.583	0.368
CNN	Fashion Mnist	-0.732	0.100	0.848	0.077	0.073
CNN	Cifar10	-0.017	0.835	1.670	0.778	0.816

Tabla 5.2: Resultados de estimadores de máxima verosimilitud para el MLP y la CNN, caso Laplace.

5.1.3. Flujos normalizadores para la construcción de distribuciones a priori

En la Figura 5.3 se muestra un ejemplo de la densidad de probabilidad ajustada para la segunda capa convolucional de la CNN, esto para cada conjunto de datos. Se puede notar en el ejemplo, que este tipo de densidades de probabilidad, basados en flujos normalizadores, son lo suficientemente flexibles para poder ajustarse a los datos, por lo que ya no es necesario restringirse a una distribución normal o una distribución Laplace.

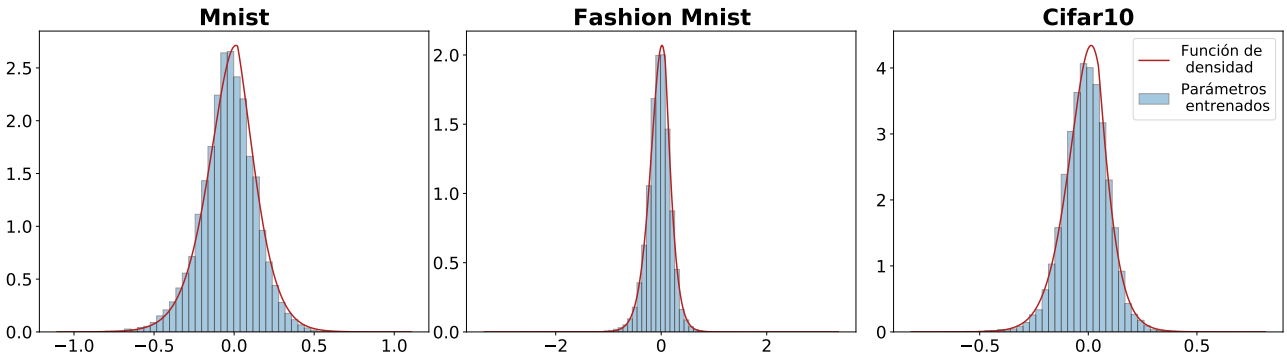

Figura 5.3: Histograma de los parámetros ya entrenados de la CNN y la función de densidad creada a través de flujos normalizadores. Elaboración propia.

5.2. Evaluación de los priors

Siguiendo el mismo orden que la definición del experimento, se mostrarán cuáles fueron los hiperparámetros escogidos, se mostrará cómo evolucionan los modelos, y finalmente se presentarán los resultados relativos al cold posterior effect.

5.2.1. Elección de hiperparámetros

En la Tabla 5.3 se pueden ver los resultados de la propuesta para la elección de los hiperparámetros en el MLP, en los conjuntos de datos MNIST, FashionMNIST, y CIFAR-10. Los resultados más relevantes son los siguientes:

- El prior Determinista con $\sigma = 0.01$ es el que obtiene mejores resultados, tanto en MNIST como en FashionMNIST. Si bien supera al resto de priors en todas las métricas consideradas, la diferencia no es sustancial. Con respecto a los hiperparámetros, es posible notar que en todos los casos, salvo en dos, el mejor tamaño para el batch fue de 64, por otro lado, el learning rate no exhibe una preferencia a un valor en particular.
- Para CIFAR-10, ninguno de los priors obtuvo buenos resultados, esto en parte a la complejidad de las imágenes, como se puede notar en los ejemplos dados en 4.3, y debido

a que la arquitectura del perceptrón multicapa no está diseñada para la tarea especifica de clasificación de imágenes, a diferencia de las redes neuronales convolucionales.

En la Tabla 5.4 se muestran los mismos resultados pero en este caso para la CNN. Los resultados más relevantes fueron los siguientes:

- Para CIFAR-10 es posible notar que el prior Determinista con $\sigma = 0.01$, supera en casi un 5 % de precisión al prior Normal. Para el resto de datos y de métricas este prior también obtiene los mejores resultados, sin embargo no existe una gran diferencia.
- Con respecto a los hiperparámetros, se puede notar que el prior Determinista con $\sigma = 0.01$ prefiere un tamaño del batch superior a 64, a diferencia del resto de priors. Por último, se puede notar que existe una tendencia a que el step size tome el valor $1.2e-06$.

Datos	**Prior**	**Step Size**	**Batch**	**Precisión**	**Logll**	**Ece**
Mnist	Determinista$_{\sigma=0.01}$	1.2e-06	64	97.02	-0.0975	0.0299
Mnist	MLE Normal	1.0e-06	64	96.97	-0.1105	0.0303
Mnist	MLE Laplace	1.0e-06	64	96.87	-0.1071	0.0313
Mnist	MLE Hibrído	1.2e-06	64	96.85	-0.1087	0.0316
Mnist	Determinista$_{\sigma=0.1}$	1.2e-06	64	96.80	-0.1062	0.0320
Mnist	Laplace	1.2e-06	64	96.72	-0.1222	0.0328
Mnist	NF	1.0e-06	64	96.67	-0.1244	0.0334
Mnist	Normal	1.2e-06	64	96.52	-0.1248	0.0348
Fashion Mnist	Determinista$_{\sigma=0.01}$	1.2e-06	256	89.78	-0.3056	0.1016
Fashion Mnist	MLE Normal	8.0e-07	64	89.17	-0.3223	0.1082
Fashion Mnist	MLE Hibrído	8.0e-07	64	89.17	-0.3223	0.1082
Fashion Mnist	NF	1.0e-06	64	89.07	-0.3211	0.1093
Fashion Mnist	MLE Laplace	1.2e-06	64	89.02	-0.3175	0.1099
Fashion Mnist	Determinista$_{\sigma=0.1}$	1.2e-06	64	88.98	-0.3154	0.1101
Fashion Mnist	Normal	1.2e-06	64	88.92	-0.3254	0.1108
Fashion Mnist	Laplace	1.0e-06	64	88.87	-0.3204	0.1113
Cifar10	MLE Normal	8.0e-07	64	51.26	-1.3895	0.4859
Cifar10	MLE Laplace	8.0e-07	64	51.22	-1.3774	0.4859
Cifar10	MLE Hibrído	1.2e-06	64	51.04	-1.3827	0.4872
Cifar10	Determinista$_{\sigma=0.01}$	1.2e-06	512	50.78	-1.4219	0.4807
Cifar10	Determinista$_{\sigma=0.1}$	1.2e-06	64	50.64	-1.4086	0.4920
Cifar10	Laplace	1.2e-06	64	49.94	-1.4515	0.4992
Cifar10	NF	1.2e-06	64	49.58	-1.4560	0.5025
Cifar10	Normal	1.0e-06	64	49.52	-1.4449	0.5038

Tabla 5.3: Resultados para el MLP bayesiano. Batch corresponde al tamaño del batch, Logll corresponde a la log verosimilitud, y Ece al expectation calibration error.

Observación Es importante mencionar que para el cálculo de estas métricas se utilizó el conjunto de validación, por lo que puede existir un sesgo de sobreajuste en los resultados. Esto, debido a que esta parte está hecha para la elección de hiperparámetros, por lo que era estrictamente necesario ocupar el conjunto de validación. Sin embargo, en el experimento asociado al cold posterior effect, sí se utiliza el conjunto de prueba.

Datos	Prior	Step Size	Batch	Precisión	Logll	Ece
Mnist	Determinista$_{\sigma=0.01}$	1.2e-06	128	99.22	-0.0416	0.0078
Mnist	Determinista$_{\sigma=0.1}$	1.0e-06	64	99.10	-0.0392	0.0088
Mnist	MLE Normal	1.0e-06	64	98.67	-0.0481	0.0133
Mnist	MLE Laplace	1.0e-06	64	98.62	-0.0487	0.0142
Mnist	Normal	1.2e-06	128	98.62	-0.0510	0.0142
Mnist	Laplace	1.0e-06	64	98.62	-0.0512	0.0140
Mnist	MLE Hibrído	1.2e-06	64	98.60	-0.0474	0.0138
Mnist	NF	1.0e-06	64	98.58	-0.0491	0.0142
Fashion Mnist	Determinista$_{\sigma=0.01}$	8.0e-07	256	93.53	-0.3233	0.0648
Fashion Mnist	Determinista$_{\sigma=0.1}$	1.2e-06	128	93.43	-0.2753	0.0660
Fashion Mnist	MLE Normal	1.0e-06	64	92.55	-0.2191	0.0747
Fashion Mnist	Normal	1.2e-06	64	92.45	-0.2193	0.0752
Fashion Mnist	Laplace	8.0e-07	64	92.30	-0.2234	0.0768
Fashion Mnist	NF	1.2e-06	64	92.27	-0.2186	0.0775
Fashion Mnist	MLE Hibrído	1.2e-06	64	92.23	-0.2237	0.0776
Fashion Mnist	MLE Laplace	1.0e-06	64	91.93	-0.2281	0.0808
Cifar10	Determinista$_{\sigma=0.01}$	8.0e-07	256	71.18	-0.8628	0.2879
Cifar10	Determinista$_{\sigma=0.1}$	1.2e-06	64	70.94	-0.9153	0.2900
Cifar10	MLE Normal	1.2e-06	64	69.92	-0.8992	0.3006
Cifar10	MLE Hibrído	1.0e-06	64	68.70	-0.9088	0.3125
Cifar10	MLE Laplace	1.2e-06	64	68.60	-0.9029	0.3144
Cifar10	Laplace	1.2e-06	64	67.28	-0.9547	0.3259
Cifar10	NF	1.2e-06	64	66.98	-0.9520	0.3304
Cifar10	Normal	1.2e-06	64	66.86	-0.9688	0.3303

Tabla 5.4: Resultados para la CNN bayesiana. Batch corresponde al tamaño del batch, Logll corresponde a la log verosimilitud, y Ece al error esperado de calibración.

5.2.2. Evolución de los modelos

En la Figura 5.4 se pueden ver los resultados correspondientes a la evolución de los modelos para el MLP en el conjunto de datos FashionMNIST. Se puede ver claramente que el prior Determinista $\sigma = 0.01$ no solamente obtiene los mejores resultados en todas las métricas, sino que también los obtiene con muchas menos muestras, lo cual es relevante, pues esto permitiría escalar a las redes neuronales bayesianas a arquitecturas más complejas, aunque existe el costo de tener que entrenarla en su forma determinista para poder definir el prior. Por otro lado, el resto de priors exhiben un rendimiento similar, con el prior Normal ligeramente superior al resto de priors.

Cabe señalar que era esperable que se obtuviera este resultado, pues el prior Determinista $\sigma = 0.01$, empieza ya con una ventaja al tener como media la solución dada por la red determinista.

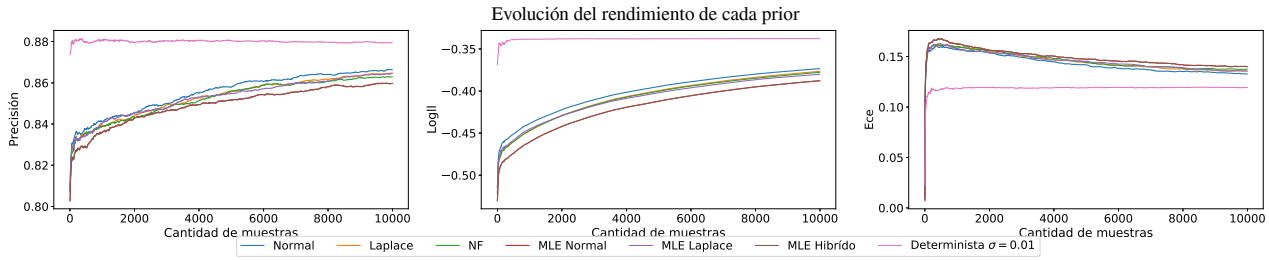

Figura 5.4: Evolución del rendimiento para cada prior para el MLP en FashionMNIST. Logll corresponde a la log verosimilitud y Ece corresponde al error de calibración esperado. Elaboración propia.

5.2.3. Estudio del cold posterior effect

En la Figura 5.5, y 5.6 se presentan los resultados para el MLP y CNN respectivamente al considerar la distribución temperada para diferentes temperaturas, para todos los conjuntos de datos,

Para el MLP, los resultados más relevantes son los siguientes:

- Es posible notar que a diferencia del prior Normal y Laplace, el prior Determinista $\sigma = 0.01$ no presenta el cold posterior effect en prácticamente ninguna métrica, salvo en el caso del error esperado de calibración en los datos MNIST. En cuanto a precisión y log verosimilitud, se puede notar que prior Determinista $\sigma = 0.01$ supera a la versión determinista de la red, en todas las temperaturas.
- En cuanto a la calibración, es posible notar resultados mixtos, lo que dificulta obtener una conclusión precisa. Por último, cabe señalar que a diferencia de lo estudiado en [14] el prior Laplace sí presenta el cold posterior effect. Una posible causa de esto es que no se utilizó la misma técnica de inferencia.

Para la CNN, los resultados más relevantes se señalan a continuación:

- Nuevamente se observa que el prior Determinista $\sigma = 0.01$ no presenta el cold posterior effect, esta vez en ninguno de los casos, a diferencia del resto de priors considerados, y es posible notar que el prior Determinista $\sigma = 0.01$ supera a la versión determinista de la red en cuanto a la precisión y verosimilitud. Con respecto a la calibración, nuevamente se obtienen resultados mixtos.
- Es posible notar que para CIFAR-10, el prior Determinista $\sigma = 0.01$, obtiene más de un 8 % mejor rendimiento en cuanto a la precisión, en comparación al prior Normal y Laplace. Este es un resultado no menor, ya que CIFAR-10 es el conjunto de datos más

realista que se ha utilizado. Adicionalmente, es mejor en cuanto a la log verosimilitud, pero no así en el error esperado de calibración, aunque sigue siendo mejor que la versión determinista de la red.

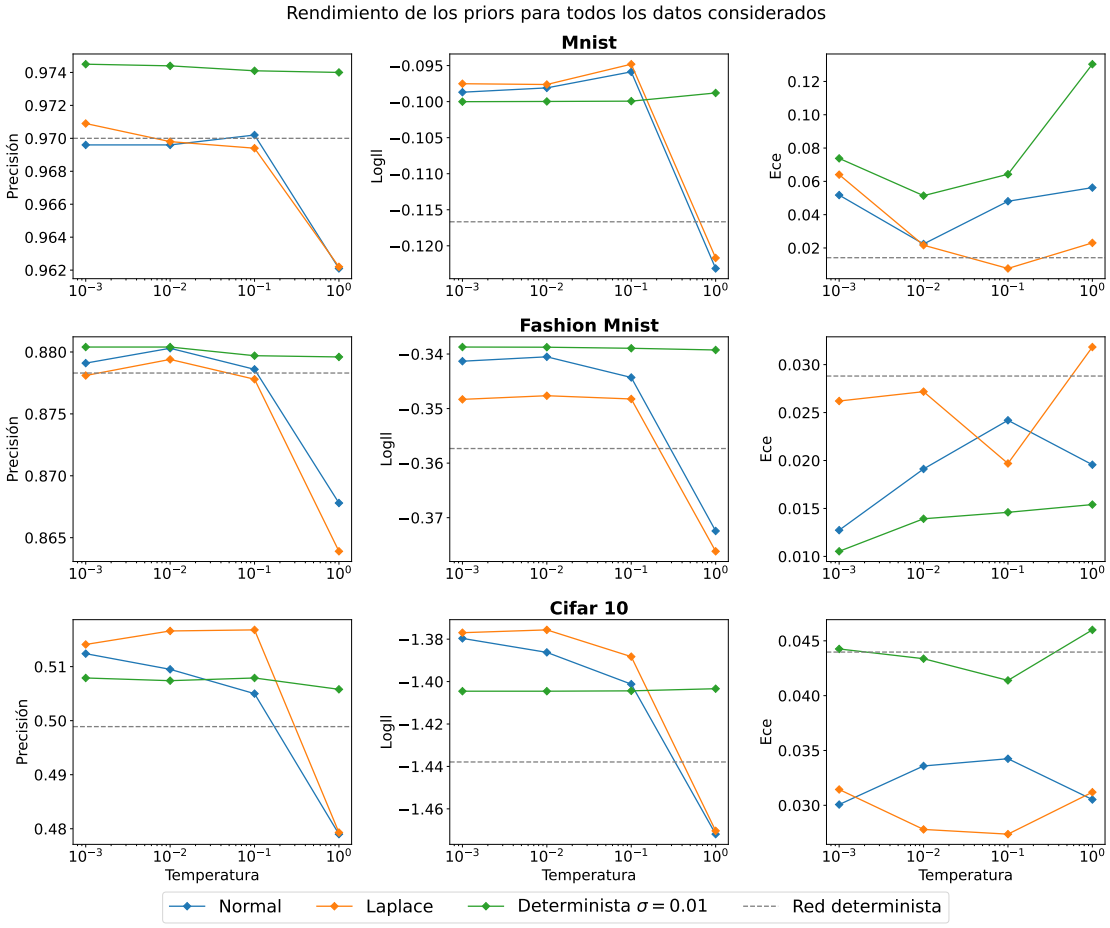

Figura 5.5: Cold posterior effect para cada prior, para el MLP en todos los conjuntos de datos considerados. Elaboración propia.

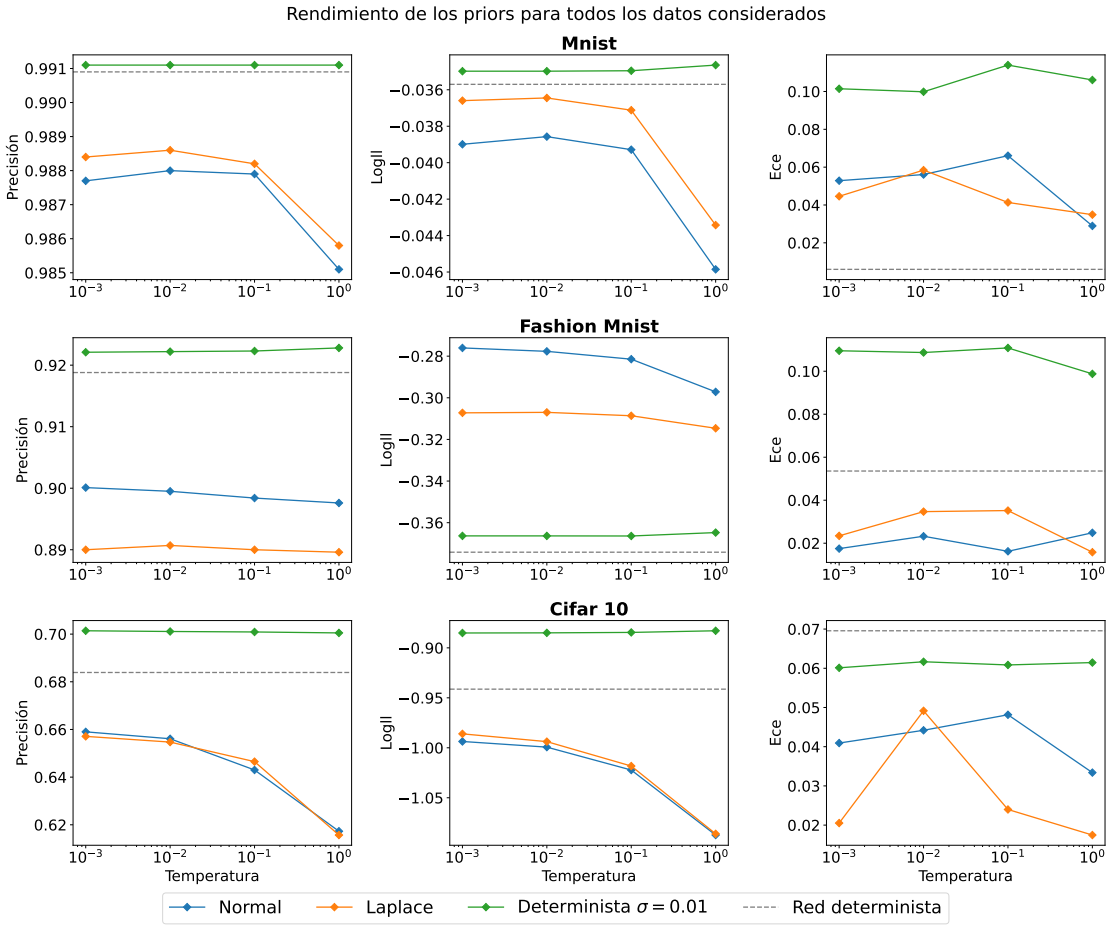

Figura 5.6: Cold posterior effect para cada prior, para la CNN en todos los conjuntos de datos considerados. Elaboración propia.

5.3. Estudio de las muestras de la posterior

En la Figura 5.7 se pueden ver histogramas de las muestras de la distribución posterior considerando el prior Normal y NF para 4 pesos aleatorios de cada capa y la solución dada por la red determinista. Es posible ver que en la mayoría de casos las muestras no consideran la solución dada por la red determinista, pues, en los pesos de la primera capa, salvo segundo y el tercer peso, las muestras obtenidas concentran su masa en una región aparte a la solución dada por la red determinista. En el resto de capas el resultado es aún más evidente, pues solamente el primer peso considerado de la cuarta capa, considera a la solución de la red determinista. Por último, notar que las muestras de todos los pesos considerados a través del prior Normal y NF son similares.

La Figura 5.8 sigue la misma idea, sin embargo se considera el prior Determinista $\sigma = 0.01$. Se puede notar, y como era de esperar, que las muestras obtenidas consideran a la solución dada por la red determinista. Además, se puede ver que no se concentran ahí, por ejemplo, en el cuarto peso de la capa 2, se puede notar que si bien la solución determinista pertenece a la región de la distribución posterior, la masa de esta no se focaliza en tal punto, lo cual sucede en la mayoría de los casos estudiados.

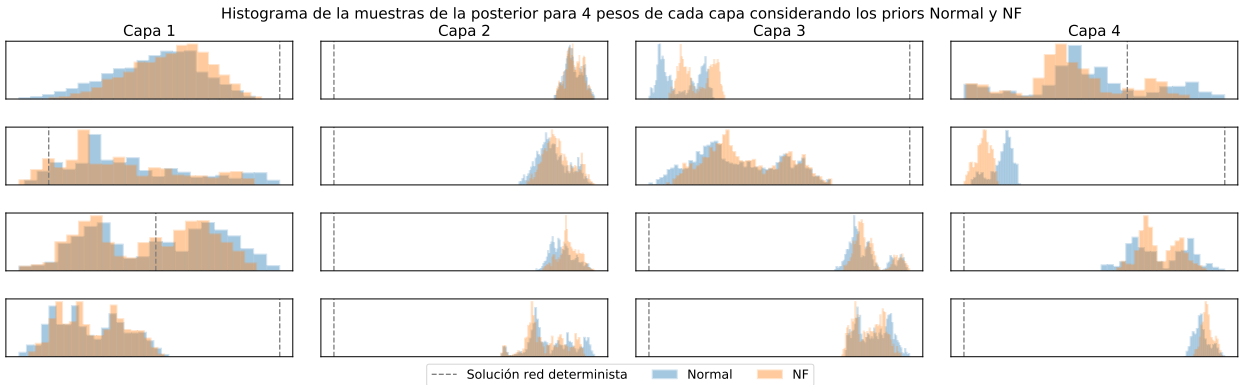

Figura 5.7: Histograma de las muestras de la posterior, para el MLP en los datos MNIST. Utilizando el prior Normal y NF. Elaboración propia.

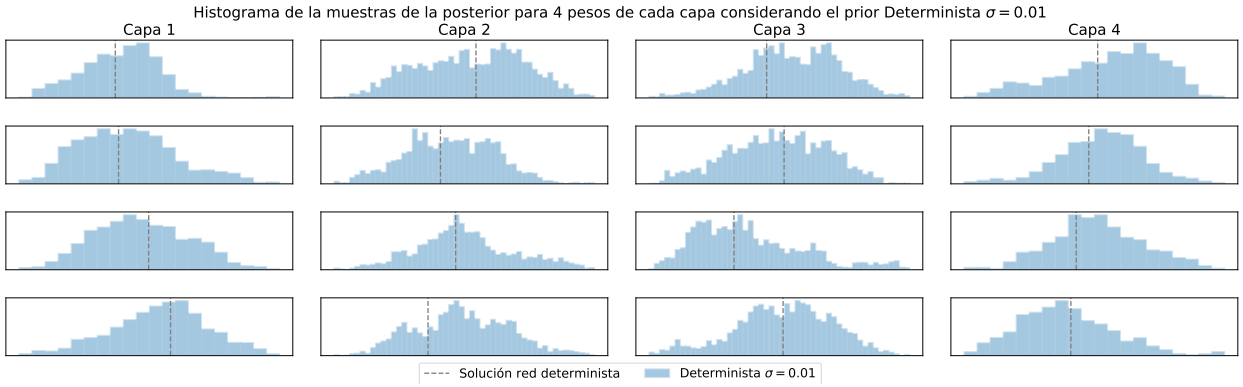

Figura 5.8: Histograma de las muestras de la posterior, para el MLP en los datos MNIST, utilizando el prior Determinista. Elaboración propia.

5.4. Aplicación en salud

En la Figura 5.9 se presenta la matriz de confusión para el conjunto de prueba. Se puede notar que para la clase 9, 2/3 pacientes se predijeron mal. Por otro lado, las clases con mayor presencia, tienen una mejor precisión, coincidente con la métrica obtenida (90.48 % de precisión, deducible del gráfico).

En la Figura 5.10 se puede notar la visualización de los datos al reducir su dimensionalidad. Al ver el gráfico realizado a través del algoritmo UMAP, se puede notar que los pacientes que presentan una mayor incertidumbre, están al borde de la región de sus clases. Por otro lado, existen pacientes que están en regiones diferentes a las de su clase, y que presentar una mayor incertidumbre también, como se puede ver en la parte superior derecha. Finalmente, los tres pacientes de la clase 9, presentan una alta incertidumbre, lo cual es coincidente con que sea la clase con peor desempeño. Se ve un resultado similar al usar la técnica t-Sne. Este es un resultado interesante, pues dice que para pacientes que sean diferentes de los pacientes con su misma enfermedad, se les da una mayor incertidumbre, lo cual es coherente con la idea de calcular la incertidumbre.

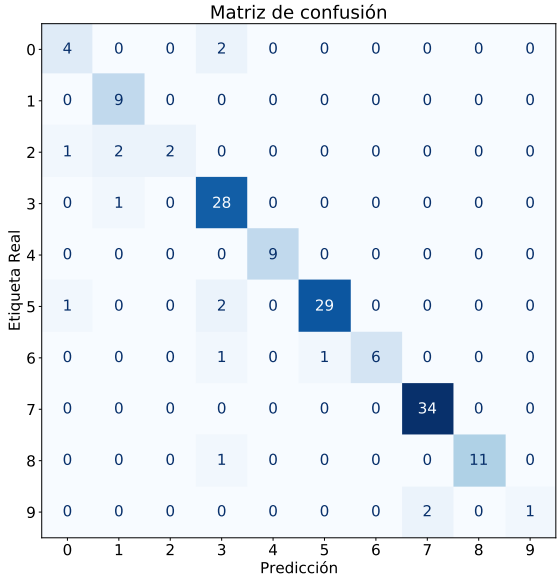

Figura 5.9: Matriz de confusión para las predicciones en el conjunto de prueba. Elaboración propia

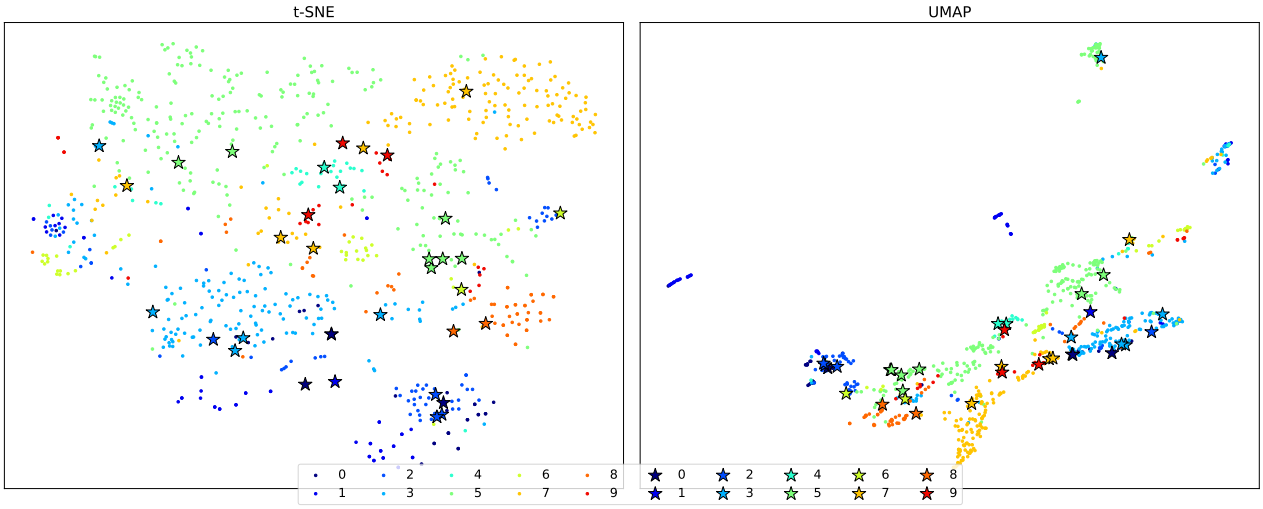

Figura 5.10: Disminución de la dimensionalidad, usando t-SNE y UMAP por separado. Elaboración propia.

Capítulo 6

Conclusión

Después de presentar y analizar los resultados obtenidos en los experimentos propuestos, se puede concluir que se ha logrado satisfacer el objetivo general establecido para este trabajo, el cual consistía en encontrar una distribución a priori que tuviera un mejor rendimiento con respecto a su versión determinista. Se encontró que uno de los priors propuestos además de obtener mejor precisión y log verosimilitud que su versión determinista, no presentó el cold posterior effect (salvo para el error esperado de calibración para ciertos conjuntos de datos). Si bien no siempre tuvo mejores resultados con respecto al error esperado de calibración, fue posible notar que en el conjunto de datos más realista, correspondiente a CIFAR-10, sí obtuvo un menor error, en el caso de la CNN, en el MLP tuvo un rendimiento similar en cuanto a ese error. Cabe señalar que el prior que obtuvo un mejor resultado fue el que se denominó Determinista con un $\sigma = 0.01$, que consistía en utilizar una distribución Gaussiana centrada en la solución dada por la red determinista.

Estos resultados demuestran que la selección adecuada de la distribución a priori puede mejorar significativamente la precisión de los modelos basados en redes neuronales, de hecho para el caso de CIFAR-10 en la CNN, se mejoró al rededor de un 8 % la precisión con respecto al prior Normal.

Adicionalmente se logró explotar el argumento de definir el prior de cada capa, con una distribución basada en la solución dada por la red determinista, donde no solamente se utilizó una técnica sencilla como lo es la estimación de máxima verosimilitud a través de una distribución normal, sino que también se utilizaron flujos normalizadores. Los priors utilizados a través de esa propuesta no obtuvieron buenos resultados.

Con respecto a la aplicación en salud, se obtuvieron resultados sumamente interesantes, pues al utilizar técnicas de reducción de dimensionalidad se pudieron graficar los resultados y darles interpretabilidad, en particular se pudo observar que la métrica utilizada para la incertidumbre era coherente con poder estimar qué tan confiado está el modelo en cuanto a su predicción.

En cuanto a las contribuciones de esta investigación a la comunidad, se destaca el estudio del prior Determinista y los resultados que obtuvo en los datos y modelos considerados, así como también el análisis realizado de las muestras de la posterior.

Una de las limitaciones de los priors propuestos es que es necesario entrenar la red de forma determinista para poder definirlos, lo cual agrega un costo extra a la implementación de tales priors. Otra de las limitaciones de esta investigación fue la selección de las arquitecturas y de los conjuntos de datos, que si bien son ampliamente utilizados en la literatura, podrían no ser representativos de todas las aplicaciones del aprendizaje profundo. Por lo que para trabajo a futuro, se propone estudiar lo realizado pero para otro tipo de arquitecturas, como lo son las redes neuronales recurrentes o para redes neuronales convolucionales más complejas, así como también la utilización de más conjuntos de datos que permitan concluir de una mejor manera.

Bibliografía

[1] Thomas Bayes. Lii. an essay towards solving a problem in the doctrine of chances. by the late rev. mr. bayes, frs communicated by mr. price, in a letter to john canton, amfr s. *Philosophical transactions of the Royal Society of London*, (53):370–418, 1763.

[2] Daniel S Berman, Anna L Buczak, Jeffrey S Chavis, and Cherita L Corbett. A survey of deep learning methods for cyber security. *Information*, 10(4):122, 2019.

[3] Michael Betancourt. A conceptual introduction to hamiltonian monte carlo. *arXiv preprint arXiv:1701.02434*, 2017.

[4] David M. Blei, Alp Kucukelbir, and Jon D. McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877, apr 2017.

[5] Emily E Cust, Alice J Sweeting, Kevin Ball, and Sam Robertson. Machine and deep learning for sport-specific movement recognition: A systematic review of model development and performance. *Journal of sports sciences*, 37(5):568–600, 2019.

[6] John S. Denker and Yann LeCun. Transforming neural-net output levels to probability distributions. In *Proceedings of the 3rd International Conference on Neural Information Processing Systems*, NIPS'90, page 853–859, San Francisco, CA, USA, 1990. Morgan Kaufmann Publishers Inc.

[7] John S. Denker, Daniel B. Schwartz, Ben S. Wittner, Sara A. Solla, Richard E. Howard, Lawrence D. Jackel, and John J. Hopfield. Large automatic learning, rule extraction, and generalization. *Complex Syst.*, 1, 1987.

[8] Shiv Ram Dubey, Satish Kumar Singh, and Bidyut Baran Chaudhuri. A comprehensive survey and performance analysis of activation functions in deep learning. *CoRR*, abs/2109.14545, 2021.

[9] Shiv Ram Dubey, Satish Kumar Singh, and Bidyut Baran Chaudhuri. Activation functions in deep learning: A comprehensive survey and benchmark, 2022.

[10] Vincent Dumoulin and Francesco Visin. A guide to convolution arithmetic for deep learning, 2016.

[11] Conor Durkan, Artur Bekasov, Iain Murray, and George Papamakarios. Neural spline flows, 2019.

[12] Andrew Y. K. Foong, David R. Burt, Yingzhen Li, and Richard E. Turner. On the expressiveness of approximate inference in bayesian neural networks, 2019.

[13] Vincent Fortuin. Priors in bayesian deep learning: A review, 2021.

[14] Vincent Fortuin, Adrià Garriga-Alonso, Sebastian W. Ober, Florian Wenzel, Gunnar Rätsch, Richard E. Turner, Mark van der Wilk, and Laurence Aitchison. Bayesian neural network priors revisited, 2021.

[15] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1050–1059, New York, New York, USA, 20–22 Jun 2016. PMLR.

[16] Yarin Gal, Riashat Islam, and Zoubin Ghahramani. Deep bayesian active learning with image data. *CoRR*, abs/1703.02910, 2017.

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[18] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 11 1997.

[19] David H Hubel and Torsten N Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology*, 160(1):106, 1962.

[20] Alexander Immer, Matthias Bauer, Vincent Fortuin, Gunnar Rätsch, and Mohammad Emtiyaz Khan. Scalable marginal likelihood estimation for model selection in deep learning, 2021.

[21] Pavel Izmailov, Sharad Vikram, Matthew D. Hoffman, and Andrew Gordon Wilson. What are bayesian neural network posteriors really like? *CoRR*, abs/2104.14421, 2021.

[22] Michael Jordan, Zoubin Ghahramani, Tommi Jaakkola, and Lawrence Saul. An introduction to variational methods for graphical models. *Machine Learning*, 37:183–233, 01 1999.

[23] Laveen N Kanal. Perceptron. In *Encyclopedia of Computer Science*, pages 1383–1385. 2003.

[24] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *CoRR*, abs/2001.08361, 2020.

[25] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014.

[26] Ivan Kobyzev, Simon J.D. Prince, and Marcus A. Brubaker. Normalizing flows: An introduction and review of current methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(11):3964–3979, nov 2021.

[27] John F. Kolen and Stefan C. Kremer. *Gradient Flow in Recurrent Nets: The Difficulty of Learning LongTerm Dependencies*, pages 237–243. 2001.

[28] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

[29] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[30] S. Kullback. *Information Theory and Statistics*. A Wiley publication in mathematical statistics. Dover Publications, 1997.

[31] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[32] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[33] Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010.

[34] Yinglong Li. Research and application of deep learning in image recognition. In *2022 IEEE 2nd International Conference on Power, Electronics and Computer Applications (ICPECA)*, pages 994–999. IEEE, 2022.

[35] David J. C. MacKay. A Practical Bayesian Framework for Backpropagation Networks. *Neural Computation*, 4(3):448–472, 05 1992.

[36] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction, 2018.

[37] Saroj K Meher and Ganapati Panda. Deep learning in astronomy: a tutorial perspective. *The European Physical Journal Special Topics*, 230(10):2285–2317, 2021.

[38] Kevin P. Murphy. *Probabilistic Machine Learning: Advanced Topics*. MIT Press, 2023.

[39] P. S. Myshkov and Simon J. Julier. Posterior distribution analysis for bayesian inference in neural networks. 2016.

[40] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Icml*, 2010.

[41] Ali Bou Nassif, Ismail Shahin, Imtinan Attili, Mohammad Azzeh, and Khaled Shaalan. Speech recognition using deep neural networks: A systematic review. *IEEE access*, 7:19143–19165, 2019.

[42] Anh Mai Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. *CoRR*, abs/1412.1897, 2014.

[43] Lorenzo Noci, Kevin Roth, Gregor Bachmann, Sebastian Nowozin, and Thomas Hofmann. Disentangling the roles of curation, data-augmentation and the prior in the cold posterior effect. *CoRR*, abs/2106.06596, 2021.

[44] Mohit Pandey, Michael Fernandez, Francesco Gentile, Olexandr Isayev, Alexander Tropsha, Abraham Stern, and Artem Cherkasov. The transformational role of gpu computing and deep learning in drug discovery. *Nature Machine Intelligence*, 4:211–221, 03 2022.

[45] Allan Pinkus. Approximation theory of the mlp model in neural networks. *Acta numerica*, 8:143–195, 1999.

[46] Pradheepan Raghavan and Neamat El Gayar. Fraud detection using machine learning and deep learning. In *2019 international conference on computational intelligence and knowledge economy (ICCIKE)*, pages 334–339. IEEE, 2019.

[47] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[48] David E. Rumelhart and James L. McClelland. *Learning Internal Representations by Error Propagation*, pages 318–362. 1987.

[49] Claude Sammut and Geoffrey I. Webb, editors. *Sequential Data*, pages 902–902. Springer US, Boston, MA, 2010.

[50] C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, 1948.

[51] Bernard W Silverman. *Density estimation for statistics and data analysis*. Routledge, 2018.

[52] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

[53] José Verdú-Díaz, Jorge Alonso-Pérez, Claudia Nuñez-Peralta, Giorgio Tasca, John Vissing, Volker Straub, Roberto Fernández-Torrón, Jaume Llauger, Isabel Illa, and Jordi Díaz-Manera. Accuracy of a machine learning muscle mri-based tool for the diagnosis of muscular dystrophies. *Neurology*, 94(10):e1094–e1102, 2020.

[54] Fei Wang, Lawrence Peter Casalino, and Dhruv Khullar. Deep learning in medicine—promise, progress, and challenges. *JAMA internal medicine*, 179(3):293–294, 2019.

[55] Lei Wang, Simon X. Yang, and Mohammad Biglarbegian. Bio-inspired navigation of mobile robots. pages 59–68, 2012.

[56] Max Welling and Yee Whye Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, page 681–688, Madison, WI, USA, 2011. Omnipress.

[57] Florian Wenzel, Kevin Roth, Bastiaan S. Veeling, Jakub Świątkowski, Linh Tran, Stephan Mandt, Jasper Snoek, Tim Salimans, Rodolphe Jenatton, and Sebastian Nowozin. How good is the bayes posterior in deep neural networks really?, 2020.

[58] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *CoRR*, abs/1708.07747, 2017.

[59] Aston Zhang, Zachary C. Lipton, Mu Li, and Alexander J. Smola. Dive into deep learning. *arXiv preprint arXiv:2106.11342*, 2021.

[60] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. Deep learning based recommender system: A survey and new perspectives. *ACM Computing Surveys (CSUR)*, 52(1):1–38, 2019.

[61] Zachary Ziegler and Alexander Rush. Latent normalizing flows for discrete sequences. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 7673–7682. PMLR, 09–15 Jun 2019.

Anexo A

A.1. Aplicación en salud

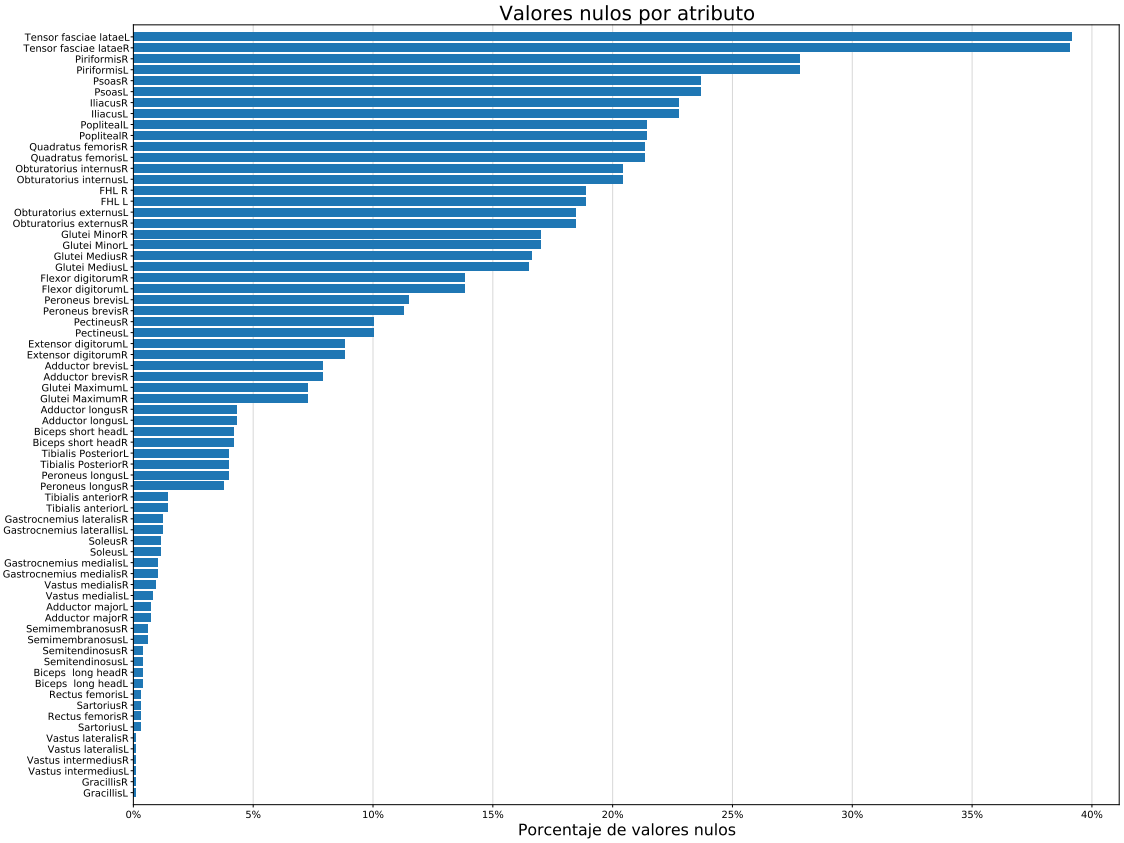

Figura A.1: Valores nulos por atributo. Elaboración propia.

A.2. Inspección de la posterior.

En las Figuras A.2 y A.3 se muestran los resultados para el perceptrón multicapa en el conjunto de datos FashionMNIST y CIFAR-10 respectivamente.

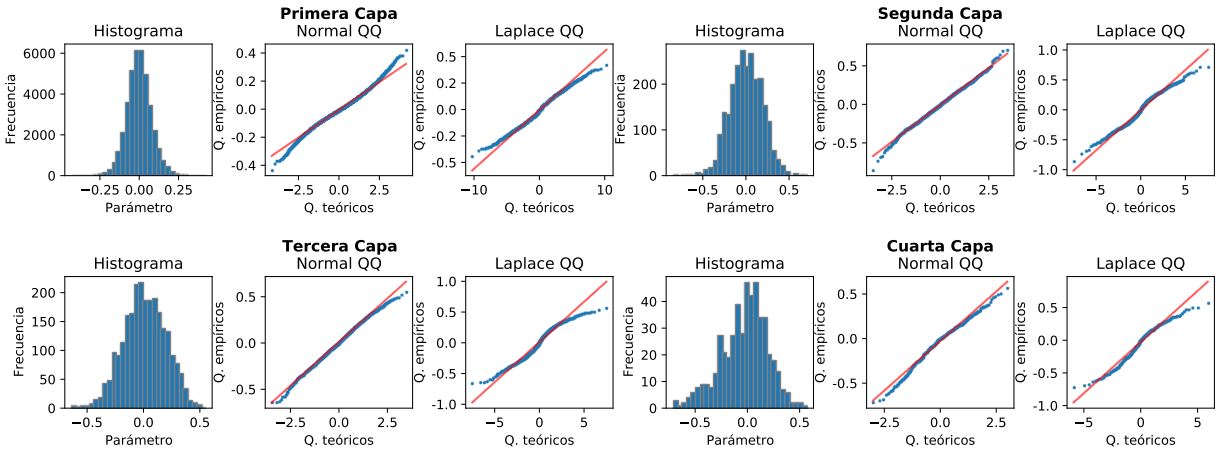

Figura A.2: Histograma de los parámetros ya entrenados separados por capa en Fashion-MNIST. Elaboración propia.

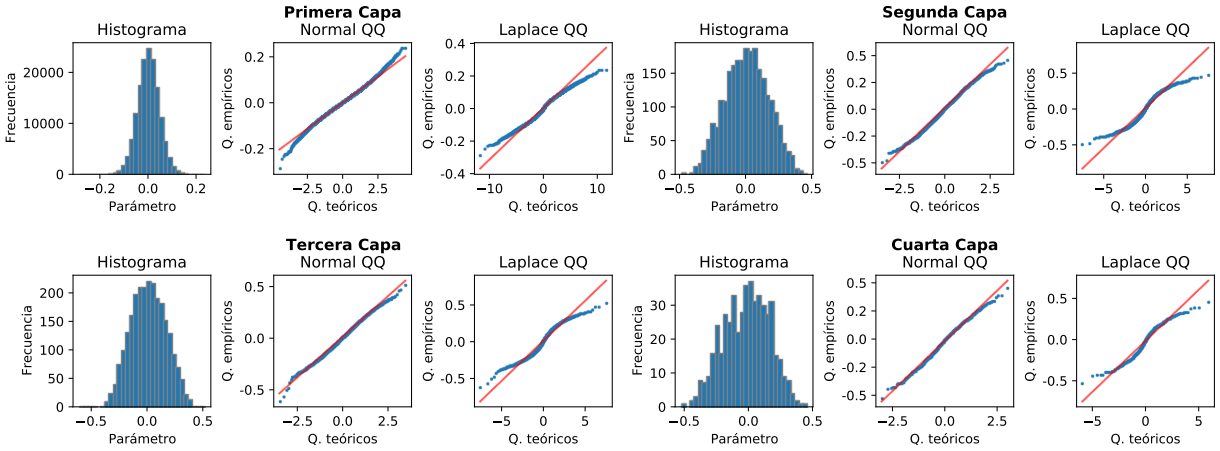

Figura A.3: Histograma de los parámetros ya entrenados separados por capa en CIFAR-10. Elaboración propia.

En las Figuras A.4 y A.5 se muestran los resultados para la red neuronal convolucional en el conjunto de datos FashionMNIST y CIFAR-10 respectivamente.

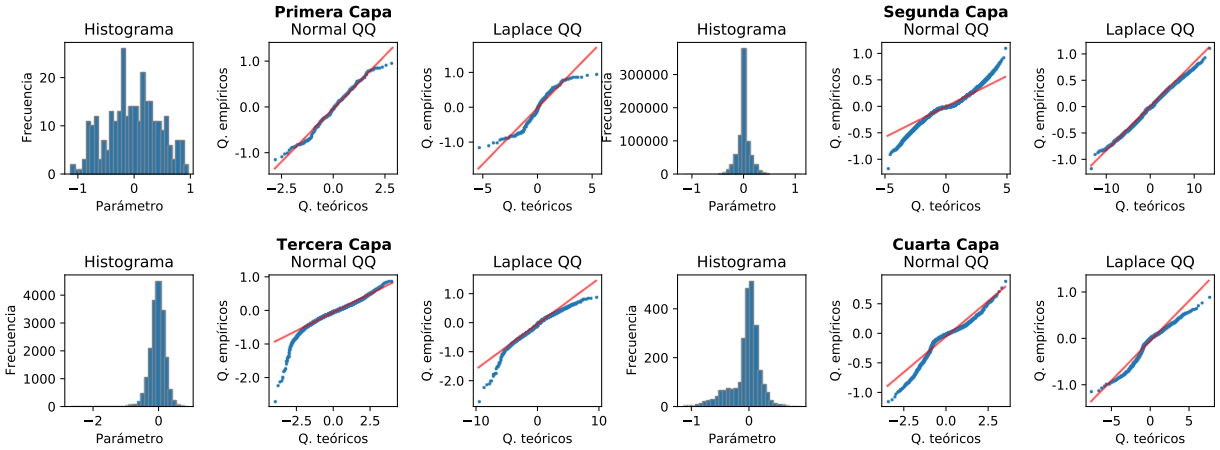

Figura A.4: Histograma de los parámetros ya entrenados separados por capa en Fashion-MNIST. Elaboración propia.

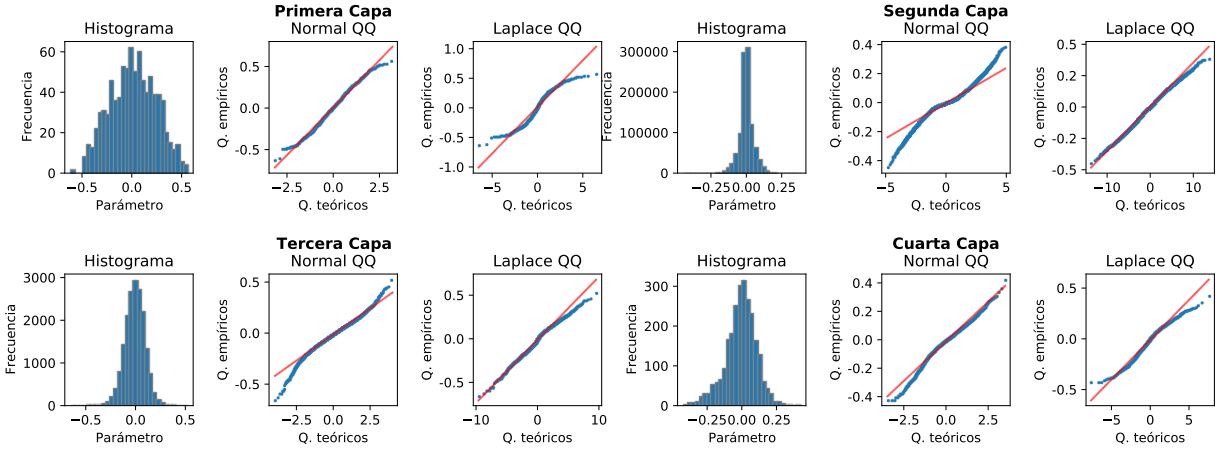

Figura A.5: Histograma de los parámetros ya entrenados separados por capa en CIFAR-10. Elaboración propia.