
# GRADO EN INGENIERÍA MATEMÁTICA E INTELIGENCIA ARTIFICIAL

# TRABAJO FIN DE GRADO

Redes de tensores para modelado generativo de datos continuos

**Autor: Marcos Garrido Ferrer**

**Director: Pablo Díez Valle**

**Co-Director: Sergio Altares López**

Madrid, Junio 2026

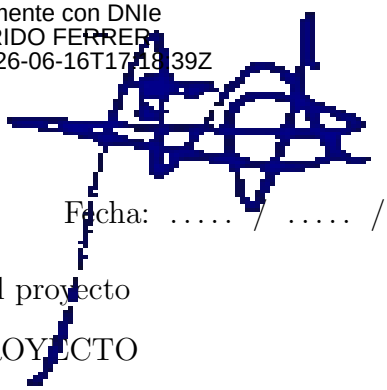
Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título

**Redes de tensores para modelado generativo de datos continuos**

en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el curso académico 2025/2026 es de mi autoría, original e inédito y no ha sido presentado con anterioridad a otros efectos.

El Proyecto no es plagio de otro, ni total ni parcialmente, y la información que ha sido tomada de otros documentos está debidamente referenciada.

Firmado electrónicamente con DNIe
Por MARCOS GARRIDO FERRER
Con fecha y hora 2026-06-16T17:18:39Z



Fdo.: Marcos Garrido Ferrer Fecha: ..... / ..... / 2026

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

Firmado por DIEZ VALLE PABLO - ***5708** el día 16/06/2026
con un certificado emitido por AC FNMT Usuarios

Fdo.: Pablo Díez Valle Fecha: ..... / ..... / 2026

EL CODIRECTOR DEL PROYECTO (si aplica)

ALTARES LOPEZ SERGIO - 50330419W

Firmado digitalmente por ALTARES LOPEZ SERGIO - 50330419W
Nombre de reconocimiento (DN): c=ES, serialNumber=IDCES-50330419W, givenName=SERGIO, sn=ALTARES LOPEZ, cn=ALTARES LOPEZ SERGIO - 50330419W
Fecha: 2026.06.16 16:05:42 +02'00'

Fdo.: Sergio Altares López Fecha: ..... / ..... / 2026


# GRADO EN INGENIERÍA MATEMÁTICA E INTELIGENCIA ARTIFICIAL

# TRABAJO FIN DE GRADO

## Redes de tensores para modelado generativo de datos continuos

**Autor: Marcos Garrido Ferrer**

**Director: Pablo Díez Valle**

**Co-Director: Sergio Altares López**

Madrid, Junio 2026

## Declaración sobre el uso de herramientas de Inteligencia Artificial

En la elaboración de este Trabajo de Fin de Grado se ha utilizado el asistente de inteligencia artificial Claude, de Anthropic (modelo Claude Opus), como herramienta de apoyo.

El código del modelo ha sido desarrollado de forma original por el autor. No obstante, los comentarios del código, así como los bucles de entrenamiento y de generación, han sido escritos con la ayuda de Claude.

La mayor parte de la redacción de esta memoria es obra personal del autor. Se han delegado a la herramienta de inteligencia artificial tareas de menor peso conceptual, como la redacción del resumen, la sección de alineación con los Objetivos de Desarrollo Sostenible y las especificaciones sobre el código.

Todo el contenido generado con ayuda de esta herramienta ha sido revisado y verificado por el autor, quien asume la plena responsabilidad sobre la originalidad, la veracidad y la corrección del trabajo presentado.

## Declaration on the Use of Artificial Intelligence Tools

In the preparation of this Bachelor's Thesis, the artificial intelligence assistant Claude, by Anthropic (Claude Opus model), has been used as a support tool.

The model code has been developed originally by the author. Nevertheless, the code comments, as well as the training and generation loops, have been written with the help of Claude.

Most of the writing of this report is the author's own work. Tasks of lesser conceptual weight have been delegated to the artificial intelligence tool, such as the writing of the abstract, the section on alignment with the Sustainable Development Goals, and the specifications about the code.

All content generated with the help of this tool has been reviewed and verified by the author, who assumes full responsibility for the originality, truthfulness, and correctness of the work presented.

REDES DE TENSORES PARA MODELADO GENERATIVO DE DATOS CONTINUOS

**Autor: Marcos Garrido Ferrer**
Director: Pablo Díez Valle
Co-Director: Sergio Altares López

# Resumen

Este trabajo desarrolla un modelo generativo de imágenes basado en una Tree Tensor Network que representa una densidad de probabilidad mediante la regla de Born. Su principal aportación es la generalización del modelo, tradicionalmente formulado sobre datos discretos, al dominio continuo. Aprovechando la forma canónica de la red, la función de partición se calcula de forma exacta, lo que permite optimizar la verosimilitud y muestrear sin aproximaciones. Se analizan las funciones de codificación y el sesgo que introducen, factor que resulta determinante: el modelo rinde satisfactoriamente en imágenes en escala de grises (MNIST), pero su rendimiento se degrada en color (CIFAR-10).

**Palabras clave**: Redes tensoriales; Máquina de Born; Modelado generativo; Tree Tensor Network; Datos continuos; Función de codificación (*embedding*).

# Resumen ejecutivo

# 1 Introducción

El modelado generativo consiste en aprender la distribución de probabilidad subyacente a un conjunto de datos (en este caso, de imágenes), de tal forma que el modelo sea capaz tanto de evaluar la verosimilitud de una imagen en función del conjunto de datos como generar nuevas muestras sintéticas conforme a él. La mayoría de los modelos generativos actuales, como las redes adversariales (GANs), autocodificadores variacionales (VAEs), modelos de difusión o basados en energía, comparten una limitación: la *función de partición* $Z$ que normaliza la distribución no se puede calcular de forma directa [1]. Esto significa que la verosimilitud no puede obtenerse de forma exacta, dependiendo de aproximaciones estocásticas.

Este trabajo explora una alternativa procedente de la física cuántica de muchos cuerpos: las *redes tensoriales* (TNs) [2]. Este modelo matemático es capaz de representar estados cuánticos eficientemente en cuanto a parámetros, lo que lo hace idóneo para realizar simulaciones en este campo. Para ello, presupone que las interacciones entre los cuerpos o partículas son locales (entre partículas vecinas o cercanas), condición que cumplen gran cantidad de sistemas en la naturaleza. Esta hipótesis es conocida como *ley de área local*, y es la que permite representar dichos estados complejos con un número limitado de parámetros: el estado se representa como una serie de nodos (tensores de pequeño tamaño) enlazados entre sí mediante

índices comunes, y al contraer dichos índices se obtiene el estado global. Existen varias familias de redes de tensores, según su topología: *Matrix Product States* (MPS), *Tree Tensor Networks* (TTN), *Projected Entangled Pair States* (PEPS)...

Para representar de forma sencilla estas diferentes topologías, existe una notación diagramática propia para las redes de tensores [2]. En ella, cada tensor se representa mediante una forma (de ahora en adelante, un círculo). El rango de un tensor queda determinado por el número de aristas salientes de él. La operación fundamental de las TNs, la contracción tensorial, también es representada en estos diagramas. Simplemente la unión de dos tensores o nodos mediante una arista significa que ambos comparten un índice "común". La Figura 1 representa la siguiente contracción:

$$C_\beta = \sum_\alpha A_\alpha B_{\alpha,\beta} \tag{1.1}$$

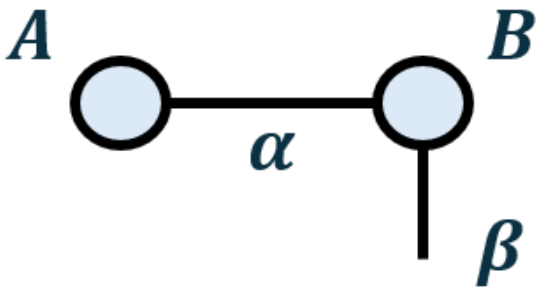



Figura 1: Contracción simple entre dos tensores. $A$ tiene un único índice (es un vector), que comparte con $B$ mediante un enlace. $B$ tiene un índice más, $\beta$, que queda libre. Como $B$ tiene dos índices, es un tensor de orden 2, es decir, una matriz. Este diagrama particular simboliza el producto de un vector y una matriz: como queda un índice libre en el diagrama, la salida de la contracción será un tensor de orden 1 (vector).

Pese a que su propósito original fuera mejorar la capacidad de simulación de estos sistemas, varios estudios recientes han descubierto en la capacidad expresiva de las TNs un gran potencial para diversas tareas de aprendizaje automático [1], [3], [4]. Muchos conjuntos de datos pueden ser representados mediante correlaciones locales (por ejemplo, los píxeles de una imagen están fuertemente correlacionados con los píxeles cercanos a ellos, y más débilmente a medida que aumenta la distancia entre sí). Además, la amplitud al cuadrado de los estados cuánticos que modela la red es proporcional a su probabilidad según la regla de Born, $p(x) = |\Psi(x)|^2/Z$, de tal forma que se obtiene un modelo generativo (denominado

*máquina de Born*, BM). Las BMs tienen una propiedad infrecuente: si la red se encuentra en *forma canónica* (una noción heredada del algoritmo DMRG para MPS [5]), la función de partición $Z$ se calcula de forma exacta, ya que es igual a la norma de un solo tensor de la red. Esto permite optimizar la verosimilitud directamente y muestrear sin necesidad de métodos aproximados.

El grueso del trabajo previo en este campo se ha desarrollado sobre datos discretos (por ejemplo, en imágenes binarizadas [1]), donde las redes de tensores han exhibido rendimientos satisfactorios, especialmente en conjuntos de datos con un número reducido de muestras. Esto se debe a que la propiedad de cálculo directo de la verosimilitud agiliza notablemente el aprendizaje, y la gran capacidad expresiva de las redes de tensores permite obtener resultados decentes con pocos parámetros [4].

# 2 Objetivos

No obstante, a día de hoy el campo del modelado de datos continuos con redes de tensores no ha sido explorado en la misma profundidad. La propia naturaleza de las TNs encaja fácilmente con los datos discretos, pero la generalización al dominio continuo no es trivial. Existen algunos estudios más recientes que dan grandes zancadas teóricas en esta dirección [3], pero las TNs continuas todavía no han sido demasiado exploradas para tareas generativas. En este trabajo, se desarrolla formalmente un modelo de *Tree Tensor Network* [2] para modelado generativo que, basado en [1], no está limitado a un conjunto de datos discreto como su predecesor. Además, se exploran los retos de la codificación de los datos continuos y su modelado por redes de tensores, las limitaciones y fortalezas de éstas en este ámbito, y las posibles líneas de investigación futuras que podrían mejorar la robustez de este modelo.

# 3 Descripción del modelo/sistema/herramienta

Para esta tarea, se ha diseñado e implementado un modelo generativo basado en una red de tensores de árbol binario (TTN). Cada píxel de la imagen de entrada se transforma mediante una función de codificación o *embedding* antes de contraerse con la red. La amplitud de un estado (imagen) $\Psi(x)$ y por consiguiente su probabilidad son el resultado de contraer el árbol totalmente con los vectores de los píxeles, y la densidad proviene de la regla de Born. El modelo se entrena por máxima verosimilitud (minimización de la *Negative Log-Likelihood*, NLL) mediante un algoritmo de barrido inspirado por el *Density Matrix Renormalization Group* (DMRG, [5], [6]). En este barrido, la red se recorre nodo a nodo desplazando el *centro de ortogonalidad*, y éste se optimiza dejando el resto del entorno fijo. Como se verá más adelante, la velocidad del entrenamiento es una de las grandes ventajas de este tipo de arquitectura con respecto a los modelos clásicos.

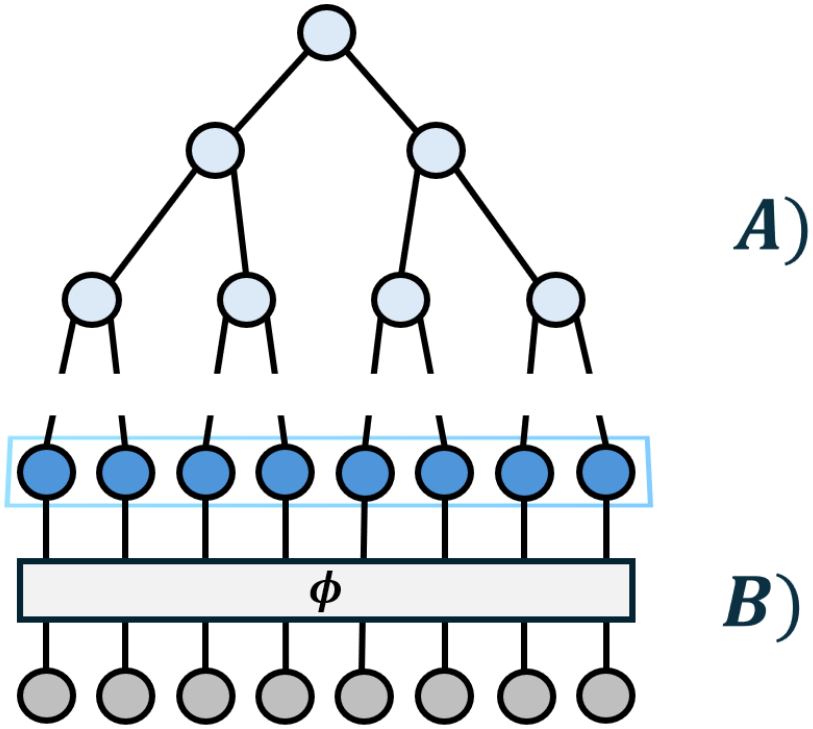



Figura 2: Diagrama de la arquitectura del modelo. La red A) está compuesta de nodos (tensores) con índices compartidos entre sí (enlaces virtuales). Estos índices serán colapsados mediante contracciones. B) representa los datos de entrada. En este caso, cada nodo gris de abajo corresponde a un píxel de la imagen. Los píxeles pueden ser vectores de longitud 1 (escala de grises) o 3 (RGB). $\phi$ es la *función de Embedding*, que toma como entrada uno de estos nodos y devuelve otro vector de longitud arbitraria: la representación del píxel en un espacio diferente. Como se verá más adelante, $\phi$ enriquece la expresividad de la red, llevando cada píxel a un dominio de mayor dimensión. Para calcular el valor de $\Psi(x)$ dada una imagen $x$, se une la red de árbol A) con las codificaciones de los datos B) a través de un enlace (índice físico). Hecho esto, la contracción total de todos estos enlaces devuelve el valor de $\Psi$, a partir del cual se puede calcular la probabilidad $p(x)$ según la regla de Born.

Además, en este trabajo se desarrolla un algoritmo de muestreo secuencial y exacto que permite regular la diversidad o nitidez de las imágenes generadas a través del mecanismo de temperatura $\tau$, similar al de otros modelos clásicos [7]. La exactitud del muestreo permite generar imágenes a partir de la distribución del modelo de forma precisa y veloz, en contraste a los métodos estocásticos usuales en modelos clásicos.

# 4 Resultados

El principal resultado de este estudio es la generalización formal del modelo TTN al dominio continuo. También se identifican tres propiedades deseables para la función de *embedding*: ortonormalidad, no degeneración y norma de la salida constante [3]. Como conclusión, como estas tres propiedades no son satisfechas de forma simultánea por ninguna de las funciones comentadas, se justifica el uso de la base de Legendre, pese a su sesgo hacia valores extremos.

# 5 Conclusiones

La velocidad de entrenamiento, la eficacia de parámetros y el algoritmo de generación exacto son las mayores ventajas de un modelo basado en TNs: el modelo sobre el MNIST completo se entrena en torno a 3 min 15 s y genera cada muestra en fracciones de segundo. El rendimiento, tanto en entrenamiento como en muestreo, es satisfactorio en imágenes en escala de grises de MNIST (1D); la temperatura de muestreo óptima, evaluada mediante la *Fréchet Inception Distance* [8], se sitúa en $\tau = 0{,}3$. No obstante, una de las limitaciones observadas es que el sesgo de la función de codificación, que en color se multiplica hasta un ratio de $1:64$, se traduce en resultados catastróficos en un conjunto de datos a color (CIFAR-10), donde el modelo llega a asignar mayor verosimilitud al ruido que a las imágenes reales.

# 6 Referencias

[1] S. Cheng, L. Wang, T. Xiang y P. Zhang, «Tree tensor networks for generative modeling», *Physical Review B*, vol. 99, n.º 15, pág. 155 131, 2019.

[2] R. Orús, «A practical introduction to tensor networks: Matrix product states and projected entangled pair states», *Annals of physics*, vol. 349, págs. 117-158, 2014.

[3] A. Meiburg, J. Chen, J. Miller, R. Tihon, G. Rabusseau y A. Perdomo-Ortiz, «Generative learning of continuous data by tensor networks», *SciPost Physics*, vol. 18, n.º 3, pág. 096, 2025.

[4] Z.-Y. Han, J. Wang, H. Fan, L. Wang y P. Zhang, «Unsupervised generative modeling using matrix product states», *Physical Review X*, vol. 8, n.º 3, pág. 031 012, 2018.

[5] U. Schollwöck, «The density-matrix renormalization group in the age of matrix product states», *Annals of physics*, vol. 326, n.º 1, págs. 96-192, 2011.

[6] S. R. White, «Density matrix formulation for quantum renormalization groups», *Physical review letters*, vol. 69, n.º 19, pág. 2863, 1992.

[7] D. H. Ackley, G. E. Hinton y T. J. Sejnowski, «A learning algorithm for Boltzmann machines», *Cognitive science*, vol. 9, n.º 1, págs. 147-169, 1985.

[8] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler y S. Hochreiter, «Gans trained by a two time-scale update rule converge to a local nash equilibrium», *Advances in neural information processing systems*, vol. 30, 2017.

TENSOR NETWORKS FOR GENERATIVE MODELLING OF CONTINUOUS DATA

**Author: Marcos Garrido Ferrer**
Director: Pablo Díez Valle
Co-Director: Sergio Altares López

# Abstract

This work develops a generative image model based on a Tree Tensor Network that represents a probability density through the Born rule. Its main contribution is the generalization of the model, traditionally formulated on discrete data, to the continuous domain. By exploiting the canonical form of the network, the partition function is computed exactly, which allows the likelihood to be optimized and samples to be drawn without approximations. The encoding (*embedding*) functions and the bias they introduce are analyzed, a factor that proves decisive: the model performs satisfactorily on grayscale images (MNIST), but its performance degrades on colour (CIFAR-10).

**Keywords**: Tensor networks; Born machine; Generative modelling; Tree Tensor Network; Continuous data; Embedding function.

# Executive Summary

# 1 Introduction

Generative modelling consists of learning the probability distribution underlying a dataset (in this case, of images), so that the model is able both to evaluate the likelihood of an image with respect to the dataset and to generate new synthetic samples consistent with it. Most current generative models, such as adversarial networks (GANs), variational autoencoders (VAEs), diffusion models or energy-based models, share a limitation: the *partition function* $Z$ that normalizes the distribution cannot be computed directly [1]. This means that the likelihood cannot be obtained exactly, and one must rely on stochastic approximations.

This work explores an alternative coming from quantum many-body physics: *tensor networks* (TNs) [2]. This mathematical model is able to represent quantum states efficiently in terms of parameters, which makes it well suited to simulations in this field. To do so, it assumes that the interactions between the bodies or particles are local (between neighbouring or nearby particles), a condition met by a large number of systems in nature. This hypothesis, known as the *local area law*, is what allows such complex states to be represented with a limited number of parameters: the state is represented as a series of nodes (small tensors) linked to each other through shared indices, and contracting those indices yields the global state. There are several families of tensor networks, according to their topology: *Ma-*

*trix Product States* (MPS), *Tree Tensor Networks* (TTN), *Projected Entangled Pair States* (PEPS)...

To represent these different topologies in a simple way, there is a diagrammatic notation specific to tensor networks [2]. In it, each tensor is represented by a shape (from now on, a circle). The rank of a tensor is determined by the number of edges leaving it. The fundamental operation of TNs, tensor contraction, is also represented in these diagrams. Simply joining two tensors or nodes by an edge means that both share a "common" index. Figure 1 represents the following contraction:

$$C_\beta = \sum_\alpha A_\alpha B_{\alpha,\beta} \tag{1.1}$$

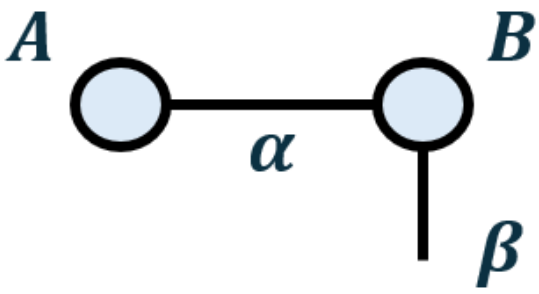



Figure 1: Simple contraction between two tensors. $A$ has a single index (it is a vector), which it shares with $B$ through a link. $B$ has one more index, $\beta$, which remains free. Since $B$ has two indices, it is an order-2 tensor, i.e. a matrix. This particular diagram symbolizes the product of a vector and a matrix: as one free index remains in the diagram, the output of the contraction is an order-1 tensor (a vector).

Although their original purpose was to improve the simulation capability of these systems, several recent studies have found in the expressive power of TNs great potential for various machine-learning tasks [1], [3], [4]. Many datasets can be represented through local correlations (for example, the pixels of an image are strongly correlated with nearby pixels, and more weakly as the distance between them increases). Moreover, the squared amplitude of the quantum states modelled by the network is proportional to its probability according to the Born rule, $p(x) = |\Psi(x)|^2/Z$, yielding a generative model (called a *Born machine*, BM). BMs have an uncommon property: if the network is in *canonical form* (a notion inherited from the DMRG algorithm for MPS [5]), the partition function $Z$ is computed exactly, since it equals the norm of a single tensor of the network. This allows the likelihood to be optimized directly and sampling to be performed without approximate methods.

The bulk of previous work in this field has been carried out on discrete data (for example, on binarized images [1]), where tensor networks have shown satisfactory performance, especially on datasets with a small number of samples. This is because the direct-likelihood-computation property substantially speeds up learning, and the great expressive power of tensor networks makes it possible to obtain decent results with few parameters [4].

# 2 Objectives

However, to date the field of continuous-data modelling with tensor networks has not been explored in the same depth. The very nature of TNs fits discrete data easily, but the generalization to the continuous domain is not trivial. There are some more recent studies that take large theoretical strides in this direction [3], but continuous TNs have not yet been much explored for generative tasks. In this work, a *Tree Tensor Network* model [2] for generative modelling is formally developed which, based on [1], is not restricted to a discrete dataset like its predecessor. In addition, the challenges of encoding continuous data and modelling it with tensor networks are explored, along with their limitations and strengths in this domain, and the possible future research lines that could improve the robustness of this model.

# 3 Description of the Model/System/Tool

For this task, a generative model based on a binary-tree tensor network (TTN) has been designed and implemented. Each pixel of the input image is transformed through an encoding or *embedding* function before being contracted with the network. The amplitude of a state (image) $\Psi(x)$, and therefore its probability, is the result of fully contracting the tree with the pixel vectors, and the density comes from the Born rule. The model is trained by maximum likelihood (minimization of the *Negative Log-Likelihood*, NLL) through a sweep algorithm inspired by the *Density Matrix Renormalization Group* (DMRG, [5], [6]). In this sweep, the network is traversed node by node, moving the *orthogonality centre*, which is optimized while the rest of the environment is kept fixed. As will be seen, the training speed is one of the great advantages of this kind of architecture compared to classical models.

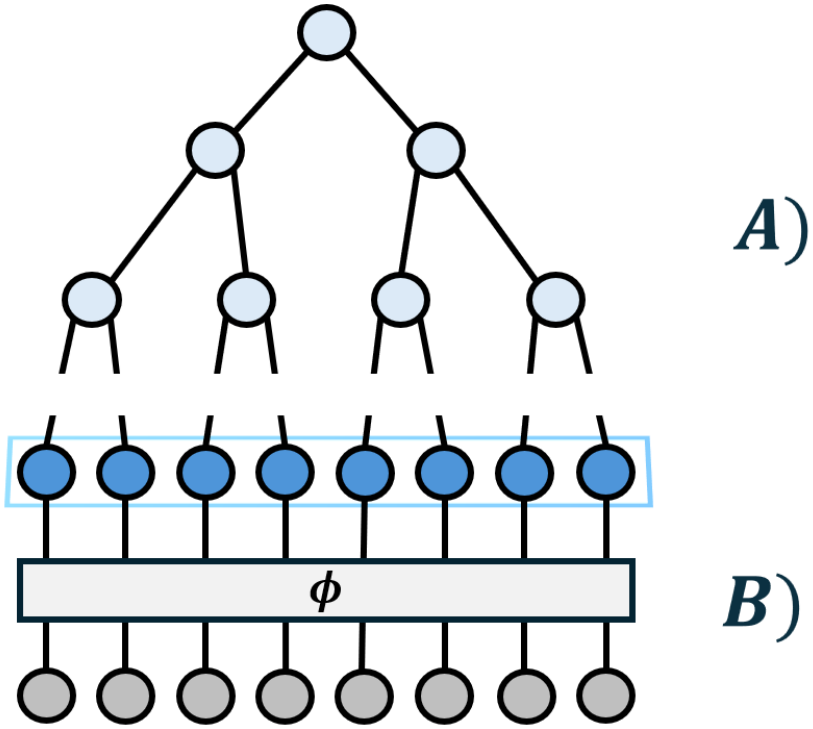



Figure 2: Diagram of the model architecture. Network A) is composed of nodes (tensors) with indices shared among them (virtual links). These indices will be collapsed through contractions. B) represents the input data. In this case, each grey node at the bottom corresponds to a pixel of the image. Pixels can be vectors of length 1 (grayscale) or 3 (RGB). $\phi$ is the *embedding function*, which takes one of these nodes as input and returns another vector of arbitrary length: the representation of the pixel in a different space. As will be seen, $\phi$ enriches the expressiveness of the network, mapping each pixel to a higher-dimensional domain. To compute the value of $\Psi(x)$ given an image $x$, the tree network A) is joined with the data encodings B) through a link (physical index). Once this is done, the full contraction of all these links returns the value of $\Psi$, from which the probability $p(x)$ can be computed according to the Born rule.

In addition, this work develops an exact sequential sampling algorithm that makes it possible to regulate the diversity or sharpness of the generated images through the temperature mechanism $\tau$, similar to that of other classical models [7]. The exactness of the sampling allows images to be generated from the model's distribution precisely and quickly, in contrast to the stochastic methods usual in classical models.

# 4 Results

The main result of this study is the formal generalization of the TTN model to the continuous domain. Three desirable properties are also identified for the *embedding* function: orthonormality, non-degeneracy and constant output norm [3]. As a conclusion, since these three properties are not satisfied simultaneously by any of the functions discussed, the use of the Legendre basis is justified, despite its bias towards extreme values.

# 5 Conclusions

The training speed, the parameter efficiency and the exact generation algorithm are the greatest advantages of a TN-based model: the model on the full MNIST dataset trains in about 3 min 15 s and generates each sample in fractions of a second. Its performance, both in training and in sampling, is satisfactory on grayscale MNIST images (1D); the optimal sampling temperature, assessed with the *Fréchet Inception Distance* [8], is $\tau = 0.3$. However, one of the observed limitations is that the bias of the encoding function, which for colour is amplified up to a ratio of $1 : 64$, leads to catastrophic results on a colour dataset (CIFAR-10), where the model even assigns higher likelihood to noise than to real images.

# 6 References

[1] S. Cheng, L. Wang, T. Xiang, and P. Zhang, «Tree tensor networks for generative modeling», *Physical Review B*, vol. 99, no. 15, p. 155 131, 2019.

[2] R. Orús, «A practical introduction to tensor networks: Matrix product states and projected entangled pair states», *Annals of physics*, vol. 349, pp. 117–158, 2014.

[3] A. Meiburg, J. Chen, J. Miller, R. Tihon, G. Rabusseau, and A. Perdomo-Ortiz, «Generative learning of continuous data by tensor networks», *SciPost Physics*, vol. 18, no. 3, p. 096, 2025.

[4] Z.-Y. Han, J. Wang, H. Fan, L. Wang, and P. Zhang, «Unsupervised generative modeling using matrix product states», *Physical Review X*, vol. 8, no. 3, p. 031 012, 2018.

[5] U. Schollwöck, «The density-matrix renormalization group in the age of matrix product states», *Annals of physics*, vol. 326, no. 1, pp. 96–192, 2011.

[6] S. R. White, «Density matrix formulation for quantum renormalization groups», *Physical review letters*, vol. 69, no. 19, p. 2863, 1992.

[7] D. H. Ackley, G. E. Hinton, and T. J. Sejnowski, «A learning algorithm for boltzmann machines», *Cognitive science*, vol. 9, no. 1, pp. 147–169, 1985.

[8] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, «Gans trained by a two time-scale update rule converge to a local nash equilibrium», *Advances in neural information processing systems*, vol. 30, 2017.

# Índice

# Índice de figuras

# Índice de tablas

# Capítulo 1 Introducción

El aprendizaje automático generativo se ocupa de un problema fundamental: dado un conjunto de datos, aprender la distribución de probabilidad que los ha producido. A diferencia de los modelos discriminativos (que aprenden a separar o etiquetar datos), un modelo generativo aspira a capturar la estructura completa de los datos, lo que le permite tanto evaluar cómo de verosímil es un ejemplo nuevo como sintetizar ejemplos que no pertenecen al conjunto original pero son plausibles. En el caso de las imágenes, esto se traduce en aprender qué configuraciones de píxeles constituyen una imagen "natural" dentro de un dominio dado, y poder generar imágenes nuevas conforme a esa noción.

Los modelos generativos más extendidos en la actualidad, como las redes adversarias (GANs) [1], los autocodificadores variacionales (VAEs) [2], los modelos de difusión [3] y los modelos basados en energía; han alcanzado resultados notables, pero comparten un problema común. Todos definen, de forma explícita o implícita, una distribución de probabilidad sobre los datos; sin embargo, el factor de normalización de esa distribución (la *función de partición* $Z$) es, en general, intratable: su cálculo exige sumar o integrar sobre todas las configuraciones posibles, un espacio de tamaño impracticablemente grande. Como consecuencia, la verosimilitud que el modelo asigna a un dato no puede obtenerse de forma exacta, y el entrenamiento debe apoyarse en aproximaciones estocásticas que complican el aprendizaje, ralentizan la convergencia y dificultan la interpretación del modelo.

Este trabajo se sitúa en la intersección entre el aprendizaje automático y un modelo matemático procedente de la física cuántica de muchos cuerpos: las *redes tensoriales* (TNs) [4]. Una red tensorial representa un objeto de dimensión muy alta (el estado de un sistema de muchas partículas) mediante una colección de tensores pequeños conectados por índices compartidos. Al contraer esos índices se reconstruye el objeto global. Esta representación es eficiente porque explota una hipótesis que cumplen numerosos sistemas físicos: la *ley de área local* [5], según la cual las correlaciones relevantes son fundamentalmente locales, entre partes vecinas del sistema. Lo notable es que muchos conjuntos de datos del mundo real comparten esa misma estructura: en una imagen, cada píxel está fuertemente correlacionado con sus vecinos y cada vez más débilmente a medida que aumenta la distancia. Esta coincidencia es la que ha motivado, en los últimos años, la adaptación de las redes tensoriales al aprendizaje automático [6]-[8].

La conexión con el modelado generativo se establece a través de la regla de Born. Estipula que la probabilidad de un estado $x$ está relacionada con la amplitud $\Psi(x)$ que la red asigna a dicha configuración: $p(x) = |\Psi(x)|^2/Z$. En esta fórmula, $Z$ es el factor de normalización que asegura que la integral a lo largo del dominio de posibles entradas $x$ es igual a 1, para que la densidad esté bien definida. El modelo resultante, conocido como *máquina de Born* [9], posee la propiedad que lo distingue de los modelos generativos clásicos: cuando la red se dispone en su *forma canónica*, la función de partición $Z$ deja de ser intratable y se calcula de forma exacta como la norma de un único tensor. Esto permite optimizar la verosimilitud directamente, sin aproximaciones, lo que es útil no sólo en el entrenamiento sino a la hora de generar muestras sintéticas.

El éxito de las máquinas de Born tensoriales se ha demostrado principalmente sobre datos discretos (típicamente imágenes binarizadas [6]), donde el formalismo encaja de manera natural: cada píxel toma un número finito de valores, que se corresponden directamente con las dimensiones de los índices de la red. Sin embargo, la mayoría de los datos de interés en aprendizaje automático son continuos: las intensidades de los píxeles de una imagen en escala de grises, o los tres canales de color de una imagen RGB, son valores reales en un intervalo. La generalización de las redes tensoriales a este dominio continuo no es directa y ha recibido, hasta la fecha, menores atención. los avances teóricos más recientes en esta línea [8] son prometedores, pero aún no se han llevado muchas veces a la práctica en conjuntos de datos reales.

Este trabajo aborda precisamente esa generalización. El reto central no es solo extender la arquitectura, sino resolver una pregunta que el caso discreto no plantea: cómo codificar un valor de píxel continuo mediante una función de *embedding* para que la densidad resultante esté correctamente normalizada y el modelo aprenda de forma eficaz. Como se mostrará, esta elección de codificación condiciona por completo el rendimiento del modelo.

## 1.1. Motivación

El interés de explorar este enfoque es doble. Por un lado, las máquinas de Born tensoriales ofrecen algo escaso entre los modelos generativos: garantías matemáticas. La normalización es exacta, no aproximada; la verosimilitud es calculable y optimizable de forma directa; el muestreo es exacto y no requiere cadenas de Markov; y la capacidad expresiva del modelo se puede diagnosticar en términos de entrelazamiento [10]. Estas propiedades hacen del modelo un objeto interpretable [11] y bien fundamentado. La rapidez de su entrenamiento, consecuencia directa de poder calcular la verosimilitud sin aproximaciones, es además una ventaja práctica relevante, especialmente ante conjuntos de datos escasos.

Por otro lado, llevar estas garantías al dominio continuo es un paso necesario para que el enfoque sea aplicable a datos reales, y plantea problemas teóricos y prácticos que deben ser estudiados por sí mismos: la codificación de variables continuas, su efecto sobre la normalización y sobre el sesgo del modelo, y las limitaciones de capacidad de la arquitectura.

## 1.2. Objetivos

### 1.2.1. Objetivo general

El objetivo general de este Trabajo de Fin de Grado es **diseñar, implementar y analizar un modelo generativo de imágenes basado en una Tree Tensor Network** (TTN), que represente una densidad de probabilidad mediante la regla de Born y se entrene por máxima verosimilitud, extendiendo el modelo (tradicionalmente formulado sobre datos discretos) al dominio de los **datos continuos**: imágenes en escala de grises y en color.

### 1.2.2. Hipótesis principal

El objetivo principal de este trabajo es formular un modelo basado en TTNs para modelado generativo basado en el modelo propuesto en [6], extendiendo el dominio del modelo a uno continuo.

La hipótesis de partida es que las propiedades distintivas de las máquinas de Born tensoriales, como la normalización exacta de la función de partición, el entrenamiento por barrido y el muestreo exacto pueden trasladarse de forma rigurosa al dominio continuo, y que el factor de diseño determinante para conseguirlo es la función de codificación (*embedding*) de los píxeles, cuya elección repercute tanto en la validez de la normalización como en el comportamiento del modelo durante el aprendizaje y la generación.

### 1.2.3. Objetivos específicos

Para abordar el objetivo general y contrastar la hipótesis, se plantean los siguientes objetivos específicos:

1. **Formular el modelo para datos continuos.** Establecer la regla de Born sobre una TTN con píxeles en $[0, 1]$ y caracterizar la condición que debe cumplir el *embedding* (su ortonormalidad, expresada mediante la matriz de Gram) para que la densidad esté bien definida y normalizada.

2. **Analizar las funciones de codificación.** Estudiar distintas familias de *embeddings* (Legendre, trigonométricas, producto tensorial para color...) [8], comparando las propiedades que pueden o no satisfacer simultáneamente (ortonormalidad, norma constante y no degeneración) y el sesgo que cada una induce sobre el modelo.

3. **Implementar la normalización y generación exactas.** Aprovechar la forma canónica para calcular la función de partición $Z$ de forma exacta, optimizar la verosimilitud mediante un algoritmo de barrido tipo DMRG con una actualización por sitio o nodo.

4. **Implementar el muestreo exacto.** Generar imágenes nuevas mediante muestreo secuencial exacto a partir de las probabilidades condicionales, con un mecanismo de temperatura que regule el compromiso entre nitidez y diversidad, sin recurrir a métodos de Monte Carlo.

5. **Evaluar el modelo.** Entrenar y validar el modelo sobre conjuntos de datos estándar en escala de grises (MNIST) y en color (CIFAR-10), analizar el comportamiento del entrenamiento y la calidad de las muestras, y discutir métricas de evaluación adecuadas para un modelo de densidad continua.

## 1.3. Alineación con los ODS

Por su naturaleza de trabajo de investigación fundamental en aprendizaje automático, este proyecto contribuye principalmente a los siguientes Objetivos de Desarrollo Sostenible (ODS) de la Agenda 2030 de las Naciones Unidas:

- **ODS 9 (Industria, innovación e infraestructura).** Es el objetivo con el que el trabajo se alinea de forma más directa, en particular con su meta 9.5, que insta a *aumentar la investigación científica y mejorar la capacidad tecnológica*. El proyecto explora un enfoque generativo poco transitado (las máquinas de Born basadas en redes de tensores) y aporta una generalización formal al dominio de datos continuos, ampliando el conocimiento disponible sobre una alternativa a los modelos generativos dominantes. Se trata, por tanto, de una contribución de carácter investigador e innovador al estado del arte.
- **ODS 13 (Acción por el clima) y ODS 7 (Energía asequible y no contaminante).** El entrenamiento de los modelos generativos actuales (GANs, modelos de difusión, etc.) conlleva un coste computacional (y por tanto energético y de emisiones) considerable. El modelo estudiado presenta ventajas relevantes en este aspecto: al calcular la función de partición de forma exacta, la verosimilitud es optimizable directamente sin recurrir a aproximaciones estocásticas costosas, y el algoritmo de barrido converge en muy pocas pasadas sobre los datos con una arquitectura eficiente en número de parámetros. Una menor demanda de cómputo para entrenar y muestrear se traduce en un menor consumo energético y una menor huella de carbono asociada, lo que vincula el trabajo con la sostenibilidad del propio desarrollo de la inteligencia artificial.

## 1.4. Estructura del trabajo

El resto del documento se organiza como sigue. En la Sección 1.2 se concretan los objetivos del trabajo. La Sección 3 desarrolla los fundamentos teóricos necesarios (redes tensoriales, forma canónica y la regla de Born), así como el modelo de Tree Tensor Network para datos continuos: la codificación mediante *embeddings*, el cálculo exacto de la función de partición, el algoritmo de entrenamiento por barrido y el muestreo exacto. La Sección 4 presenta y discute los experimentos realizados sobre datos en escala de grises y en color. Finalmente, la Sección 5 recoge las conclusiones y las líneas de trabajo futuro.

# Capítulo 2 Estado del arte

## 2.1. Conceptos fundamentales

El modelado generativo tiene como objetivo aprender una distribución de probabilidad $p(x)$ a partir de un conjunto de muestras, de modo que el modelo permita tanto evaluar la verosimilitud de un dato nuevo como generar muestras sintéticas conforme a la distribución aprendida. Conviene distinguir entre modelos de densidad *explícita*, que parametrizan directamente $p(x)$ y, en el caso ideal, permiten evaluarla; y modelos de densidad *implícita*, que únicamente proporcionan un mecanismo de muestreo sin una expresión evaluable de la densidad.

El obstáculo central, común a la mayoría de estos modelos, es la *función de partición* $Z$: el factor que normaliza la distribución para que integre a uno. Su cálculo exige sumar o integrar la densidad sin normalizar a lo largo de todo el espacio de configuraciones, un dominio de tamaño exponencial en la dimensión del dato. Cuando $Z$ es intratable, la verosimilitud no puede evaluarse de forma exacta y el entrenamiento debe apoyarse en aproximaciones estocásticas, lo que ralentiza la convergencia y dificulta el diagnóstico del modelo.

Una vía alternativa para definir una densidad procede de la mecánica cuántica: la *regla de Born*, $p(x) = |\Psi(x)|^2/Z$, que asocia la probabilidad de una configuración al módulo al cuadrado de una amplitud $\Psi(x)$. Un modelo generativo construido sobre esta regla recibe el nombre de *máquina de Born* (*Born machine*) [9]. La amplitud $\Psi$ admite múltiples representaciones; este trabajo, al igual que sus predecesores, la representa mediante una *red tensorial*.

## 2.2. Técnicas y modelos relevantes

El panorama del modelado generativo está hoy dominado por modelos basados en redes neuronales profundas: las redes generativas adversarias (GANs) [1], los autocodificadores variacionales (VAEs) [2], los modelos de difusión [3] y los modelos autorregresivos [12], junto con los modelos basados en energía (EBMs), entre los que se cuentan las máquinas de Boltzmann [13]. Estos enfoques han alcanzado una calidad de muestreo notable, pero comparten la dificultad de la función de partición intratable, que abordan de distintas maneras: las GANs evitan modelar $p(x)$ de forma explícita (densidad implícita), los VAEs optimizan una cota inferior de la verosimilitud en lugar de la verosimilitud exacta, y los EBMs recurren a técnicas de muestreo aproximado (MCMC, divergencia contrastiva) para estimar el gradiente de $\log Z$. En ninguno de ellos se dispone, de forma simultánea, de verosimilitud exacta y muestreo exacto.

Frente a este panorama, las redes tensoriales (TNs) ofrecen un punto de partida distinto. Surgieron en la física cuántica de muchos cuerpos como herramienta para representar de forma compacta el estado de sistemas de muchas partículas [4], [14], y su eficiencia se apoya en la *ley de área* [5]: cuando el entrelazamiento entre dos subsistemas escala con la frontera que los separa y no con su volumen, el estado admite una descripción con un número de

parámetros manejable. Según su topología se distinguen varias familias, como los *Matrix Product States* (MPS) [15], las *Tree Tensor Networks* (TTN) [16], los *Projected Entangled Pair States* (PEPS) [17] o el *Multi-scale Entanglement Renormalization Ansatz* (MERA) [18]. El algoritmo de referencia para optimizarlas es el *Density Matrix Renormalization Group* (DMRG) [19], [20], del que el presente trabajo hereda la estrategia de barrido y la noción de forma canónica.

La observación de que muchos conjuntos de datos exhiben correlaciones predominantemente locales, análogas a las de los sistemas físicos, ha motivado la adaptación de las redes tensoriales al aprendizaje automático. Más allá de su uso en tareas supervisadas [21], resultan especialmente atractivas para el modelado generativo a través de la regla de Born, pues al disponer la red en forma canónica la función de partición $Z$ se vuelve tratable y se calcula de forma exacta como la norma de un único tensor, lo que permite optimizar la verosimilitud directamente y muestrear sin métodos aproximados [6], [7]. Su capacidad expresiva, así como su relación con otros modelos probabilísticos, se ha analizado en términos de entrelazamiento, lo que les confiere un grado de interpretabilidad poco habitual [10], [11].

## 2.3. Trabajos relacionados

La línea de trabajo más próxima a este proyecto es la de las máquinas de Born tensoriales para modelado generativo. En [7] se introduce una máquina de Born basada en un MPS, entrenada por máxima verosimilitud mediante un algoritmo de tipo DMRG que aprovecha la forma canónica para calcular $Z$ de forma exacta y muestrear secuencialmente; los resultados se presentan sobre conjuntos discretos, típicamente imágenes binarizadas. El trabajo de [6] sustituye la topología de cadena del MPS por una *Tree Tensor Network*, cuya estructura jerárquica captura mejor las correlaciones bidimensionales de las imágenes y reduce la distancia entre píxeles vecinos en el árbol; este modelo constituye la base directa sobre la que se construye el presente trabajo. La expresividad de estas representaciones y su comparación con otros modelos generativos ha sido estudiada de forma sistemática [10]. Más recientemente, [8] da pasos teóricos hacia el modelado de datos *continuos* con redes tensoriales, proponiendo familias de funciones de codificación (*embeddings*) y técnicas de muestreo por transformada inversa, y es la referencia más cercana a la extensión que aquí se persigue.

## 2.4. Limitaciones de las soluciones actuales

Los modelos generativos clásicos, pese a su madurez y a la calidad de sus muestras, arrastran la intratabilidad de $Z$: no ofrecen una verosimilitud exacta, su entrenamiento descansa en aproximaciones estocásticas costosas y su comportamiento es difícil de interpretar. Las máquinas de Born tensoriales resuelven precisamente estas carencias, pero el grueso del trabajo previo se ha desarrollado sobre *datos discretos* (imágenes binarizadas o con un número reducido de niveles), donde el formalismo encaja de forma natural porque cada valor del píxel se corresponde con una dimensión del índice físico de la red.

La generalización al dominio *continuo* (imágenes en escala de grises o en color, cuyos píxeles toman valores reales en un intervalo) no es directa y permanece poco explorada. Los avances teóricos recientes [8] son prometedores, pero rara vez se han llevado a la práctica sobre conjuntos de datos reales, y dejan abiertas cuestiones determinantes: qué función de codificación emplear para que la densidad resultante esté correctamente normalizada, qué sesgo introduce cada codificación sobre la distribución aprendida, y hasta qué punto la capacidad de la arquitectura permite modelar datos más complejos, como las imágenes a color.

Este es precisamente el vacío que aborda el presente trabajo: trasladar de forma rigurosa las garantías de las máquinas de Born tensoriales (normalización exacta, verosimilitud optimizable directamente y muestreo exacto) al dominio continuo, situando la elección de la función de *embedding* como el factor de diseño central del problema.

# Capítulo 3 Metodología

## 3.1. Redes de tensores como modelo generativo

Las redes de tensores de árbol binario (Binary TTNs) [16] siguen una estructura jerárquica. Están formadas por una serie de capas, de forma que cada una de ellas contiene la mitad de tensores (también llamados nodos) que su capa hija. Cada tensor puede tener hasta tres enlaces: un nodo padre y dos hijos.

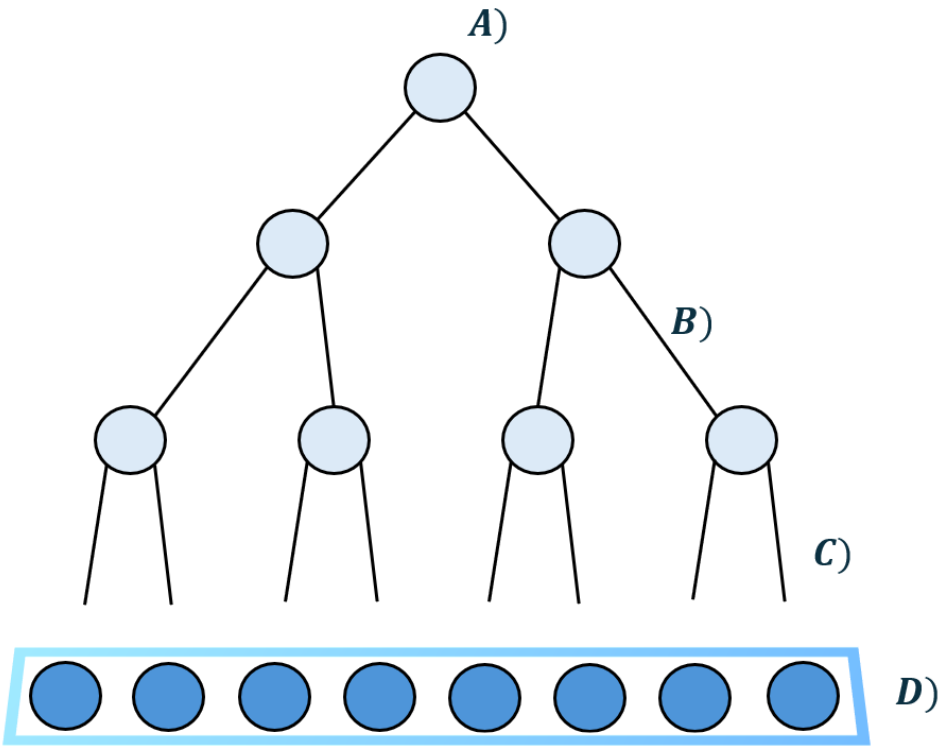



Figura 1: Estructura de una TN de árbol binario genérica. Cada círculo representa un nodo (tensor) de la red. A) Nodo raíz. B) Índice virtual. C) Índice físico. D) Capa de datos.

La Figura 1 muestra la estructura de una TTN. Cada nodo en la figura tiene una arista saliente por índice del tensor. Cada enlace entre dos nodos representa un índice compartido, también denominado índice virtual. Las ramas expuestas de la capa inferior o índices físicos serán determinadas por las muestras del conjunto de datos.

Para calcular la salida $S$ de cada nodo $T$, se realiza la siguiente contracción:

$$L \in \mathbb{C}^{B_l} \tag{3.1}$$

$$R \in \mathbb{C}^{B_r} \tag{3.2}$$

$$T \in \mathbb{C}^{B \times B_l \times B_r} \tag{3.3}$$

$$S \in \mathbb{C}^{B} = \sum_{i,j} T_{b,i,j} L_i R_j \tag{3.4}$$

donde $L$ y $R$ son las salidas de los nodos hijos de $T$. Nótese que ambas son vectores de longitudes arbitrarias. Por simplicidad, los tensores de una misma capa tendrán las mismas dimensiones, y, por lo tanto, sus salidas serán vectores igual longitud ($B_l = B_r$). No obstante, nada impide que los tensores de capas diferentes puedan tener distinto tamaño. El término $B$ de las ecuaciones anteriores se denomina dimensión de enlace del nodo $T$. Aumentar o reducir la dimensión de enlace de una capa permite de cierto modo fijar la cantidad de entrelazamiento que permitimos en esa capa, y por lo tanto su expresividad.

## 3.2. Geometría de la red

Para un árbol binario de N capas, existen multitud de topologías diferentes según la orientación de los enlaces de cada capa. En la Figura 2 se muestran dos criterios distintos para reducir la forma de cada capa. La red A) es el resultado de aplanar la imagen, de forma que $p$ es un vector de dimensión $H^W$, donde $H$ es la altura de la imagen original y $W$ es su anchura. De esta forma, $p$ está formado por los píxeles de la imagen en orden lexicográfico. La red se ha construido uniendo los nodos de dos en dos, por lo que primero se realizan todas las contracciones horizontales, seguidas de todas las contracciones verticales (ver figura). B) simplemente sigue un patrón alternante: se realizan pares de contracciones horizontal-vertical hasta que, si la imagen no es cuadrada, la única opción es contraer en la dimensión mayor.

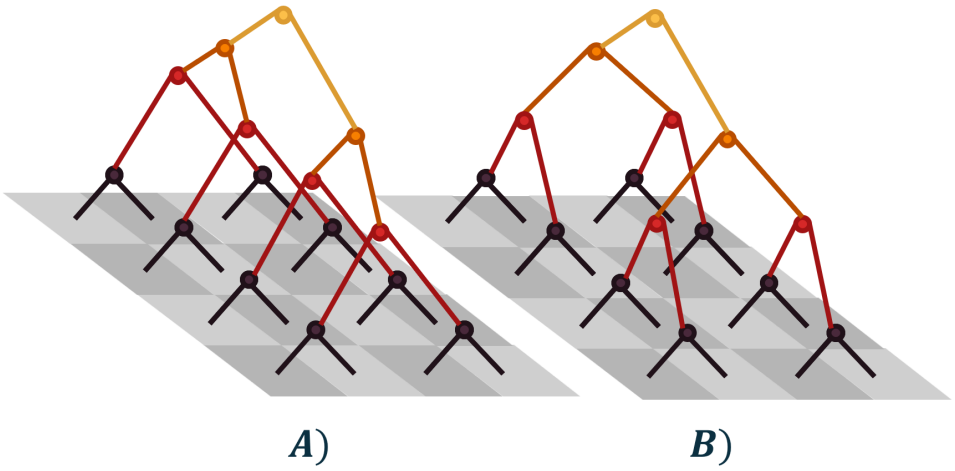



Figura 2: Diferentes topologías de árbol binario. A) no alternante en ejes. B) alternante.

Para cualquier TTN, pero en especial un modelo generativo, existe gran diferencia estas dos geometrías. Como los nodos entrelazados son aquellos entre los cuales existe una arista en el árbol, la red B) va a ser capaz de capturar mejor la relación entre los píxeles señalados en azul en la Figura 3, dado que el camino que conecta ambos en el árbol es más corto que en el caso de A). Esta diferencia se acentúa a medida que aumenta el tamaño de la imagen, pese a que la proporción entre sus lados permanezca constante. La red B) minimiza la distancia esperada entre un píxel y sus vecinos cercanos, por lo que será la empleada de ahora en adelante.

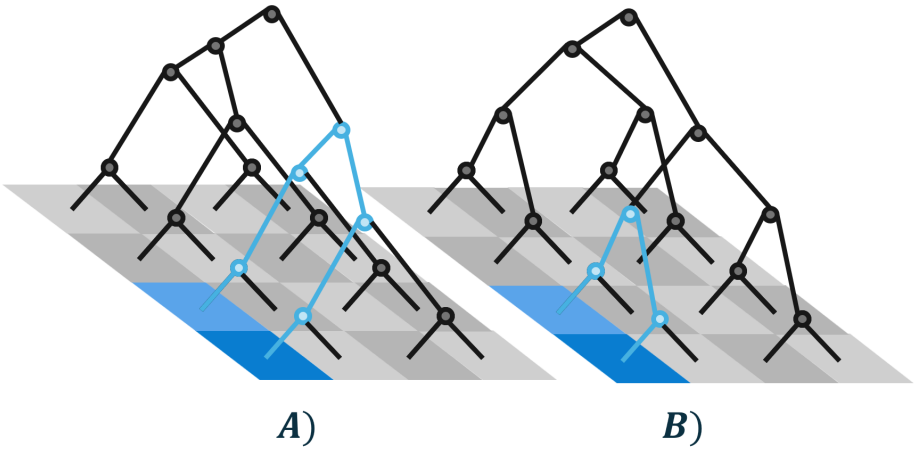



Figura 3: Recorrido necesario para conectar dos píxeles contiguos según topología.

Se puede calcular directamente el número de nodos de una red cuya entrada es de dimensión $H \times W$. Como la capa más cercana a los datos tendrá un nodo por cada dos píxeles de la imagen, necesitará $(H \times W)/2$ de ellos. Cada capa contendrá la mitad de nodos que la anterior, de forma que la última será tan solo el nodo raíz. Eso significa que se verifica:

$$N_{\text{nodos}} = \sum_{n=1}^{\log_2(H\times W)-2} 2^n = H \times W - \sum_{i=-1}^{-\infty} 2^i = H \times W - 1 \tag{3.5}$$

## 3.3. Normalización y forma canónica

Partiendo de este modelo base, es posible definir un modelo generativo. En particular, como debe retornar una probabilidad (escalar) dada una muestra, la dimensión de enlace del nodo raíz debe ser igual a 1. Sea $E$ el conjunto de posibles entradas del modelo. Para que la función de densidad de probabilidad continua que representa el modelo esté bien definida, se debe cumplir:

$$p(x) \geq 0 \quad \forall x \in E \tag{3.6}$$

$$\int_E p(x)\, dx = 1 \tag{3.7}$$

Una forma natural de garantizar que se cumplen estas ecuaciones es definir la probabilidad de un estado mediante la regla de Born, es decir:

$$p(x) = \frac{|\Psi(x)|^2}{Z} \tag{3.8}$$

$$Z = \int_E |\Psi(x)|^2\, dx \tag{3.9}$$

donde $\Psi(x)$ es la contracción total de la red dada una entrada $x$. Para ello, es preciso extender la red de la Figura 1, añadiendo un nodo por cada entrada de $x$ conectado a la capa inferior del árbol a través del índice físico correspondiente. El conjunto de nodos de este nuevo nivel se denomina *capa de datos*.

Siguiendo la convención de numeración lexicográfica de los nodos de un árbol binario, tal que el nodo raíz tiene índice 1 y los hijos izquierdo y derecho del nodo $n$ tienen índice $2n$ y $2n+1$, respectivamente:

$$\Psi(x) = \sum_{\{\alpha\}} T^{[1]}_{\alpha_2,\alpha_3} \prod_{n=2}^{N_t} T^{[N]}_{\alpha_n,\alpha_{2n},\alpha_{2n+1}} \mid N_t := \text{número de tensores en la red extendida.} \tag{3.10}$$

El cálculo de $\Psi(x)$ es directo. Además, como la contracción tensorial cumple las propiedad asociativa y distributiva, es posible cambiar el orden o la agrupación de las operaciones. El

caso más sencillo consiste en contraer los tensores de cada capa con sus respectivos padres hasta alcanzar la raíz. Obtener el factor de normalización $Z$ es más complejo. Podría realizarse un muestreo de Montecarlo en el espacio de posibles entradas, calcular sus amplitudes $\Psi$ y tratar de aproximar el valor de $Z$ de este modo. Esto requeriría un gran tiempo de cómputo cada vez que se modificara cualquiera de los nodos de la red (por ejemplo, al entrenar al modelo). Además, como la mayoría de la densidad probabilística en un modelo entrenado se acumula en un subconjunto muy reducido del espacio de posibles muestras, lo más probable es que la aproximación fuera pobre. No obstante, existe una solución para calcular $Z$ de forma exacta y considerablemente más rápida [19], [20].

Se dice que un tensor $T^{[n]} \in \mathbb{C}^{B\times B_l\times B_r}$ es canónico para el primer índice (hacia la raíz) si:

$$\sum_{ij} T^{[n]}_{b,i,j}(T^{[n]}_{b',i,j})^* = \delta_{b,b'} \tag{3.11}$$

Siendo $\delta_{b,b'} = \begin{cases} 1 & b = b' \\ 0 & b \neq b' \end{cases}$ la delta de Kronecker. Agrupando los índices sobrantes en la misma dimensión ($M \in \mathbb{C}^{(B_l\otimes B_r)\times B}$), es lo mismo que decir:

$$M^\dagger M = \mathbb{I}_B \tag{3.12}$$

Esto se puede conseguir aplicando descomposición QR sobre las columnas de $M$, de tal forma que:

$$M = QR \mid Q \text{ tiene columnas ortonormales}, R \text{ triangular superior} \tag{3.13}$$

De este modo, queda garantizado que:

$$Q^\dagger Q = \mathbb{I}_B \tag{3.14}$$

$$M^\dagger M = R^\dagger R \neq \mathbb{I}_B \tag{3.15}$$

De esta forma, las columnas de $Q$ serán linealmente independientes. Como $M$ es una matriz de tamaño $(B_l \times B_r, B)$, es preciso que se cumpla:

$$B \leq B_l \times B_r \tag{3.16}$$

Si no fuera el caso, sería imposible obtener $B$ columnas ortogonales entre sí. La interpretación natural de esta restricción es la siguiente: en el caso límite en el que se cumple la igualdad, para cada par de entradas de $B_l$ y $B_r$ existe una entrada de $B$, por lo que la contracción está conservando la información de sus entradas íntegramente. A partir de este punto, reducir la dimensión de enlace $B$ limita la cantidad de correlación (o entrelazamiento, si pensamos en $\Psi$ como un estado cuántico) que se permite entre las entradas del nodo.

Ahora, se puede ajustar la forma de $Q$ a la del tensor original y sustituirlo por ésta. Para no alterar el valor de $\Psi(x)$ para ninguna entrada, el residuo $R$ ha de ser absorbido por el nodo padre de $T^{[n]}$ a través de la siguiente contracción:

$$T^{[\lfloor n/2 \rfloor]\prime} = \begin{cases} \sum_k R_{i,k} T^{[\lfloor n/2 \rfloor]}_{b,k,j} & \text{si } T^{[n]} \text{ es el hijo izquierdo.} \\ \sum_k R_{j,k} T^{[\lfloor n/2 \rfloor]}_{b,i,k} & \text{si } T^{[n]} \text{ es el hijo derecho.} \end{cases} \tag{3.17}$$

Desde un punto de vista geométrico, la sustitución de $T^{[n]}$ por $Q$ y la contracción de $T^{[\lfloor n/2 \rfloor]}$ con $R$ puede entenderse como un cambio de la base común entre ambos tensores. Al igual que en el producto de matrices se cumple: $AB = A\mathbb{I}B = ACC^{-1}B$ siempre que se respeten las dimensiones de las matrices, este es un caso particular de la contracción tensorial. Así, $Q$ representa la nueva base ortonormal en el índice del enlace, mientras que $R$ es el cambio de coordenadas entre la base antigua y la nueva. De este modo, la salida de la red es idéntica para cualquier entrada $x$.

De manera análoga, en una TTN un tensor puede ser canónico hacia su hijo izquierdo (canónico para el segundo índice) o derecho (canónico para el tercer índice). También se puede ortonormalizar un tensor cualquiera para cumplir con estas condiciones de forma similar. La única diferencia destacable es que, al contraer el residuo $R$ con el hijo $T^{[m]}$ correspondiente, en cualquier caso se realiza la misma contracción, dado que el índice que $T^{[m]}$ comparte con su padre siempre es el primero independientemente de si es el hijo izquierdo o derecho de éste:

$$T^{[m]\prime} = \sum_k R_{b,k} T^{[m]}_{k,i,j} \tag{3.18}$$

Es posible canonizar mediante un barrido los tensores de cualquier TTN, de forma que se acabe con un solo tensor genérico no canónico, denominado nodo central. Esta forma toma el nombre de forma canónica mixta de la red. A modo de ejemplo, se puede partir de los tensores de la capa inferior, canonizar éstos hacia la raíz y repetir el proceso con cada una de las capas siguientes sucesivamente, de forma que el nodo central sea la propia raíz.

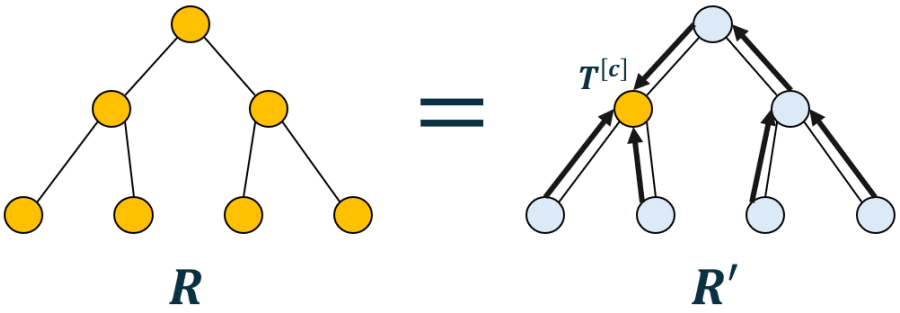



Figura 4: TTN binaria genérica (izquierda) y canonizada con respecto a un centro cualquiera $T^{[c]}$. Los nodos naranjas no son canónicos. Los azules son canónicos hacia la dirección de la flecha saliente de ellos. La dirección de canonización de los nodos es hacia arriba, excepto para aquellos que se encuentran en el camino de la raíz al nodo central, que apuntan hacia éste.

Para hallar el factor de normalización $Z$, lo más sencillo es partir de un caso base y llegar a la ecuación general por inducción. La red más sencilla posible es aquella que tiene un único nodo $T \in \mathbb{C}^{1\times D\times D}$ y por lo tanto su entrada $x := [x_1, x_2]$ son dos vectores complejos de longitud $D$. Si las muestras son imágenes en escala de grises, deberán tener tan sólo dos píxeles: $\{p_1, p_2\} \in [0, 1]$. Tanto $x_1$ como $x_2$ serán construidos a partir de un píxel de la imagen cada uno mediante la siguiente función de *embedding*:

$$\phi : [0, 1] \longmapsto \mathbb{C}^D \tag{3.19}$$

$$p \longmapsto [\phi_1(p), \ldots, \phi_D(p)] \tag{3.20}$$

De este modo, la amplitud de la entrada $x$ bajo el modelo es:

$$\Psi(x) = \Psi(x_1, x_2) = \sum_{i,j} T_{i,j}\phi_i(p_1)\phi_j(p_2) \tag{3.21}$$

El factor de normalización es entonces:

$$
\begin{aligned}
Z &= \int_0^1 \int_0^1 |\Psi(x_1, x_2)|^2 \, dx_1 \, dx_2 \\
&= \int_0^1 \int_0^1 \left| \sum_{i,j} T_{i,j} \phi_i(p_1) \phi_j(p_2) \right|^2 dp_1 \, dp_2 \\
&= \sum_{i,j} \sum_{k,l} T_{i,j} \overline{T_{k,l}} \int_0^1 \int_0^1 \phi_i(p_1) \overline{\phi_k(p_1)} \phi_j(p_2) \overline{\phi_l(p_2)} \, dp_1 \, dp_2 \\
&= \sum_{i,j} \sum_{k,l} T_{i,j} \overline{T_{k,l}} \int_0^1 \phi_i(p_1) \overline{\phi_k(p_1)} \, dp_1 \int_0^1 \phi_j(p_2) \overline{\phi_l(p_2)} \, dp_2
\end{aligned}
\tag{3.22}
$$

Ahora, suponiendo que la función de *embedding* $\phi$ cumple la siguiente identidad:

$$
\int_0^1 \phi_a(p_n) \overline{\phi_b(p_n)} \, dp_n = \delta_{a,b} \tag{3.23}
$$

Tan sólo sobrevivirán los términos del sumatorio de la Ecuación (3.22) tales que $(i = k) \wedge (j = l)$, de forma que:

$$
Z = \sum_{i,j} T_{i,j} \overline{T_{i,j}} \cdot 1 \cdot 1 = \sum_{i,j} |T_{i,j}|^2 = \|T\|^2 \tag{3.24}
$$

Siguiendo un proceso similar, es posible demostrar que, dado un árbol de tres nodos en forma canónica mixta, su factor de normalización es igual a la norma al cuadrado del tensor central no canónico. De esta forma, es posible llegar a las identidades de la Figura 5. Siguiendo el procedimiento de barrido para canonizar una red de tensores de árbol binario genérica (ver Figura 4), y aplicando acto seguido estas identidades de forma recursiva, se obtiene que el factor de normalización de la red es precisamente la norma al cuadrado del nodo central.

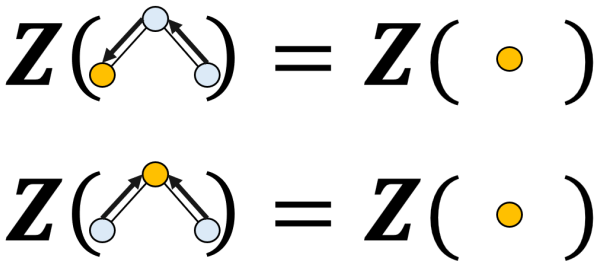


Figura 5: Forma canónica mixta de una TTN binaria de tres nodos. Los nodos azules son canónicos hacia la dirección indicada por las flechas. El nodo naranja es el tensor central de la red (no canónico). En ambos casos el factor de normalización se reduce a la norma al cuadrado del tensor central: $Z = \|T^{[c]}\|^2$.

## 3.4. Amplitudes reales frente a complejas

Todo el formalismo matemático anterior está escrito sobre $\mathbb{C}$. La regla de Born (Ecuación (3.8)) proviene de la mecánica cuántica, donde el estado $\Psi$ es complejo. No obstante, todas las ecuaciones se cumplen, en particular, en $\mathbb{R}$. En esta implementación, dado que no hay ninguna fase física que modelar (los datos son reales) y tan sólo es necesario el módulo de un estado para calcular su probabilidad, todos los tensores son reales. Esta elección es deliberada, ya que los parámetros reales reducen a la mitad la memoria empleada y evitan la aritmética compleja en las contracciones. Es también la convención de los trabajos de referencia por la misma razón. En [7] y [6], se entrenan modelos reales con resultados satisfactorios.

## 3.5. *Embeddings*

Hasta ahora, se ha supuesto que la función de *embedding* $\phi$ cumple con la Ecuación (3.23). Para facilitar la notación conviene introducir la *matriz de Gram de* $\phi$:

$$G_{ij}^{[\phi]} = \int_0^1 \phi_i(x)\phi_j(x)\,dx \tag{3.25}$$

Entonces, la Ecuación (3.23) es equivalente a:

$$G^{[\phi]} = \mathbb{I} \tag{3.26}$$

Para que la red esté correctamente normalizada según el método anterior, es preciso utilizar un $\phi$ que verifique esta identidad. A la hora de elegir una función de *embedding* para imágenes en escala de grises ($x \in [0, 1]$), la función más sencilla posible es utilizar el valor del propio píxel. De este modo, el *embedding* de cada píxel sería un vector de longitud 1. El problema de este planteamiento es el siguiente: según la Ecuación (3.16), la dimensión de

enlace de un nodo no puede superar el producto de las dimensiones de sus dos vectores de entrada. En este caso, la dimensión de enlace de la primera capa (y, por inducción, la del resto de capas) sería 1. Para que la dimensión de las capas pueda crecer con respecto a su profundidad, la dimensión del *embedding* debe ser, por lo tanto, mayor o igual a 2. Asimismo, el embedding determina el espacio de funciones que la red puede representar para cada píxel, ya que $\Psi$ es multilineal en $\phi(x_k)$. Usar el valor crudo $\phi(x) = [x]$ restringe ese espacio a funciones lineales que pasan por el origen. En particular, el modelo asignaría densidad nula a cualquier píxel de valor 0. Dicho esto, una elección natural es la función que transforma el intervalo [0, 1] en una semicircunferencia centrada en el origen:

$$\phi(x) = [\phi_1(x), \phi_2(x)] = \sqrt{2}[\cos(\pi x), \sin(\pi x)] \tag{3.27}$$

Se puede demostrar de forma sencilla que esta $\phi$ cumple la Ecuación (3.23):

$$G_{11} = \int_0^1 \phi_1(x)\phi_1(x)\,dx = \int_0^1 2\cos^2(\pi x)\,dx = 1 \tag{3.28}$$

$$G_{22} = \int_0^1 \phi_2(x)\phi_2(x)\,dx = \int_0^1 2\sin^2(\pi x)\,dx = 1 \tag{3.29}$$

$$G_{12} = \int_0^1 \phi_1(x)\phi_2(x)\,dx = G_{21} = \int_0^1 \phi_2(x)\phi_1(x)\,dx = \int_0^1 \sin(2\pi x)\,dx = \left[\frac{-\cos(2\pi x)}{2\pi}\right]_0^1 = 0 \tag{3.30}$$

Sin embargo, esta $\phi$ presenta un problema: es *degenerada*. Como $\phi(0) = -\phi(1)$, invertir el valor de un píxel de negro a blanco sólo cambia el signo de $\Psi$. Por lo tanto, como la probabilidad de un estado depende solamente de su módulo y no de su fase, $p(x_k = 0 \mid \cdot) = p(x_k = 1 \mid \cdot)$. El modelo, entonces, asigna siempre la misma probabilidad a un píxel negro y a uno blanco, lo que hace a esta función de *embedding* inadecuada para datos prácticamente binarios.

Una solución a esto podría ser restringir el movimiento de la imagen de el tramo [0, 1] a un cuarto de vuelta ($\phi(x) = [\cos(\pi x/2), \sin(\pi x/2)]$). Si bien el modelo ahora es capaz de distinguir entre píxeles negros y blancos, se rompe la condición de ortonormalidad de los *embeddings*: $G \neq \mathbb{I}$.

Existen varias funciones clásicas que cumplen con la Ecuación (3.26). En [8] se proponen varias de ellas, empezando por la base de Legendre de grado 1 desplazada, ortonormal en [0, 1]:

$$\phi(x) = [1, \sqrt{3}(2x - 1)] \tag{3.31}$$

La matriz de Gram de esta función es la identidad, por lo que es un *embedding* candidato para el modelo. Asimismo, la imagen de un píxel blanco bajo esta función difiere de la de uno negro, lo que elimina el problema de la no discriminación. De esta forma, la función de

densidad de probabilidad que representa el modelo está bien definida (esto es, cumple con las Ecuaciones (3.6) y (3.7)). Sin embargo, la norma del *embedding* no es constante a lo largo de la escala de grises (Ecuación (3.32)), lo que introduce un sesgo a priori sobre los datos. En concreto, la norma de un píxel negro es el doble de la de un tono intermedio, de modo que el modelo le asigna a priori mayor densidad esperada.

$$\begin{aligned} \phi(0) &= [1, -\sqrt{3}]; \quad \|\phi(0)\| = 2 \\ \phi(1) &= [1, +\sqrt{3}]; \quad \|\phi(1)\| = 2 \\ \phi(1/2) &= [1, 0]; \quad \|\phi(1/2)\| = 1 \end{aligned} \tag{3.32}$$

En [8] se cuantifica de forma exacta este sesgo. Para un MPS inicializado con elementos independientes e idénticamente distribuidos con media cero, la distribución marginal esperada en cada sitio es:

$$P_{\text{init}}(x) = \frac{1}{D}\|\phi(x)\|^2 = \frac{1}{D}\sum_{k=1}^{D}\phi_k(x)^2 \tag{3.33}$$

Es decir, es proporcional a la norma al cuadrado del *embedding*. En el ejemplo anterior, la probabilidad esperada a priori de un píxel negro o blanco sería cuatro veces mayor a la de uno intermedio. Además, esta diferencia se acentúa a medida que se incrementa el grado de la base de Legendre: para grado 2, la norma alcanza el máximo en 9, mientras que su mínimo sigue siendo 1. El crecimiento de este sesgo es cuadrático con respecto al grado por cómo se construye la base de Legendre.

El resto de funciones propuestas en [8], aunque resultarían interesantes para otros conjuntos de datos, no son indicadas para esta tarea. Los polinomios de Fourier se pueden utilizar para el intervalo [0, 1], pero son degenerados de la misma forma que los *embeddings* angulares vistos anteriormente. Su aplicación sería más útil en dominios periódicos. Los polinomios de Laguerre permiten utilizar datos no negativos sin una cota superior, pero para cumplir con la Ecuación (3.26) es preciso utilizar todo el intervalo $[0, \infty]$. Los polinomios de Hermite, de forma similar, requieren trabajar en $[-\infty, \infty]$. Se podría realizar una transformación de los datos (por ejemplo, utilizar $1/x$ en vez de $x$), pero nuevamente se estaría sesgando la distribución de probabilidad.

Ninguna de las funciones mencionadas hasta ahora cumple con las tres propiedades deseadas: ortonormalización de los *embeddings*, no degeneración y norma constante. Aunque la $\phi$ de Legendre introduzca sesgo en el modelo, le permite distinguir entre los extremos de intensidad de los píxeles. Además, si el entrenamiento es suficientemente bueno cabe la posibilidad de que aprenda de tal forma que contrarreste este sesgo.

Para imágenes a color se puede utilizar la siguiente propiedad. La matriz de Gram del producto tensorial de varias funciones de *embedding* es igual al producto tensorial de sus matrices de Gram:

$$G^{[\phi_1 \otimes \phi_2 \otimes \phi_3]} = G^{[\phi_1]} \otimes G^{[\phi_2]} \otimes G^{[\phi_3]}. \tag{3.34}$$

Esto se debe a que cada factor depende de un canal distinto $(r, g, b)$ y estos se integran de forma independiente, por lo que la integral que define la Gram se separa en producto de tres integrales:

$$\begin{aligned} G^{[\phi\otimes\phi\otimes\phi]}_{(ijk),(i'j'k')} &= \int \phi_i(r)\phi_{i'}(r)\,dr \int \phi_j(g)\phi_{j'}(g)\,dg \int \phi_k(b)\phi_{k'}(b)\,db \\ &= G^{[\phi]}_{ii'} G^{[\phi]}_{jj'} G^{[\phi]}_{kk'}. \end{aligned} \tag{3.35}$$

De esta forma, partiendo de cualquier función de *embedding* cuya matriz de Gram sea la identidad, su producto tensorial por ella misma también tendrá Gram identidad ($\mathbb{I}\otimes\mathbb{I}\otimes\mathbb{I} = \mathbb{I}$), y el modelo seguirá estando bien normalizado. Tomando el *embedding* de Legendre de orden 1, se obtiene una dimensión física $D = 2^3 = 8$.

## 3.6. Función de pérdida

Como se ha introducido anteriormente (Ecuación (3.8)), el modelo define una densidad de probabilidad sobre las imágenes mediante la regla de Born, y se entrena por máxima verosimilitud. Esto es, se buscan los parámetros que maximizan la probabilidad del conjunto de datos $\mathcal{D} = \{x^{[1]}, \ldots, x^{[N]}\}$. La verosimilitud de estos ejemplos se define como:

$$\prod_n p(x^{[n]}) \tag{3.36}$$

Maximizar esta verosimilitud es equivalente a minimizar el logaritmo negativo de su media, lo que da lugar a la NLL:

$$\mathcal{L} = -\frac{1}{N}\sum_{n=1}^{N} \log p(x^{[n]}) = -\frac{1}{N}\sum_{n=1}^{N} [2\log|\Psi(x^{[n]})| - \log Z] \tag{3.37}$$

Minimizar la Ecuación (3.37) equivale a minimizar la divergencia de Kullback-Leibler [22] entre la distribución empírica de los datos $p_{\text{dat}}$ y la del modelo $p$:

$$\mathrm{KL}(p_{\text{dat}}\|p) = \sum_x p_{\text{dat}}(x)\log\frac{p_{\text{dat}}(x)}{p(x)} = -H(p_{\text{dat}}) - \frac{1}{N}\sum_{n=1}^{N}\log p(x^{[n]}) \tag{3.38}$$

El término $-H(p_{\text{dat}})$ depende del conjunto de datos y no de los parámetros del modelo, por lo que la minimización de ambos criterios es equivalente. La NLL es la función de pérdida estándar en [7], [6] y [8]. Volviendo a la Ecuación (3.37), el primer término recompensa maximizar la probabilidad de las entradas del conjunto de datos, mientras que el segundo término penaliza que la integral de la probabilidad de las entradas de todo dominio aumente.

Cabe mencionar que, al tratarse de un modelo generativo continuo, la probabilidad de una región del dominio puede ser superior a 1 y, por lo tanto, la función de pérdida puede tomar valores negativos arbitrariamente grandes, siempre y cuando la distribución sea suficientemente apuntada. Esto es imposible en el caso de un modelo discreto, puesto que como la suma de las probabilidades de cada punto del dominio tiene que ser 1 y no pueden tomar valores negativos, la probabilidad en cada punto debe ser menor o igual a 1. Esto hace que la NLL pierda parte de su interpretabilidad. Por ello, quedará reservada para el entrenamiento del modelo, mientras que su diagnóstico será realizado a partir de otras métricas derivadas, como la diferencia de log-verosimilitud entre datos reales y ruido.

## 3.7. Algoritmo de barrido

Como se ha mencionado en la Sección 3.3, para poder realizar el cálculo del factor de normalización $Z$ es necesario que la TTN esté en forma canónica. Esto significa que el algoritmo de entrenamiento debe re-canonizar la red después de cada actualización de sus pesos, o bien mantener la forma canónica de la red en cada actualización. La primera opción es, en la práctica, significativamente más lenta y computacionalmente intensiva que la segunda. La canonización de una red genérica supone realizar, por lo menos, una factorización QR por cada uno de sus nodos, así como una contracción del residuo R con otro de los nodos por cada factorización. Sin embargo, la evaluación de la red dada una entrada es directa siempre y cuando esté en forma canónica. Esta es la motivación del algoritmo de barrido de la TTN: en todo momento se mantiene la red canonizada, y en cada paso se actualiza únicamente el nodo central no normalizado. Este esquema de barrido es heredado directamente del algoritmo DMRG (*Density Matrix Renormalization Group*) para MPS [19], [20] y es el estándar en otros trabajos de referencia [6]-[8].

**Algoritmo 1** Barrido de entrenamiento del TTN.

**Require:** TTN canonizada hacia el nodo hoja más a la derecha
**Ensure:** TTN actualizada y canonizada hacia el nodo hoja más a la derecha

**for** dir $\in$ {izquierda, derecha} **do**
  marcar todos los nodos como *no actualizado*
  $T^c \leftarrow$ nodo central
  $C \leftarrow$ nodos adyacentes a $T^c$ (arriba, izq., dcha.) no actualizados
  **while** $C \neq \emptyset$ **do**
    **if** $|C| = 1$ **then**
      evaluar un *batch*; calcular $\mathcal{L}$; actualizar $T^c$ mediante SGD; marcar $T^c$ como *actualizado*
      $T^c_{\text{sig}} \leftarrow$ el único nodo de $C$
    **else if** $|C| = 2$ y el padre de $T^c \in C$ **then**
      $T^c_{\text{sig}} \leftarrow$ el nodo de $C$ que no es el padre
    **else**
      $T^c_{\text{sig}} \leftarrow$ hijo de $T^c$ opuesto a dir
    **end if**
    $M = QR$  (QR de $T^c$ hacia $T^c_{\text{sig}}$)
    $T^c \leftarrow Q$; absorber $R$ en $T^c_{\text{sig}}$
    $T^c \leftarrow T^c_{\text{sig}}$
    $C \leftarrow$ nodos adyacentes a $T^c$ no actualizados
  **end while**
**end for**

Este algoritmo intercala los pasos de descenso de gradiente estocástico (SGD) con los movimientos del tensor central a lo largo de la red, de forma que todo nodo es $T^c$ en al menos un paso, y cada nodo se actualiza una vez y solo una por dirección de barrido (izquierda/-derecha) a excepción de los dos nodos hoja de los extremos, que se actualizan una vez por ejecución del algoritmo en vez de dos. Además, cada uno de los enlaces entre dos nodos se recorre, como máximo, dos veces por dirección de barrido (una hacia arriba y otra hacia abajo).

Emplear un optimizador SGD estándar para este modelo no es directo, debido a la estrategia de actualización. Entre cada barrido los tensores cambian de manera arbitraria, al ser factorizados mediante el algoritmo QR. Por lo tanto, la "velocidad.º inercia tiene en cada actualización un valor obsoleto, ya que presupone un sistema de coordenadas estable. Podría simplemente descartarse el uso de la inercia, incrementando la probabilidad de que los parámetros del modelo quedaran atrapados en un mínimo local. Una solución teóricamente razonable sería cambiar de base el tensor de inercia de cada nodo de acuerdo con su factorización, de forma que nunca llegara a quedar obsoleto. Esto aumentaría la complejidad del sistema significativamente, y en vista a los resultados del entrenamiento sin ello, no parece que fuera a tener un impacto considerable en el rendimiento. Otra solución menos sofisticada pero más sencilla en su implementación es realizar varios pasos de descenso de gradiente

estocástico en cada actualización, utilizando así a partir del segundo paso el término de velocidad. Este es el método de esta implementación debido a su compromiso entre sencillez y eficacia.

Como se verá en la sección de Resultados, la NLL (Ecuación (3.37)) cae drásticamente duranteel primer barrido (de derecha a izquierda) del entrenamiento, hasta estabilizarse en una meseta a partir de la cual el descenso es ya marginal. Este comportamiento no es un artefacto, sino una consecuencia directa de la naturaleza del algoritmo.

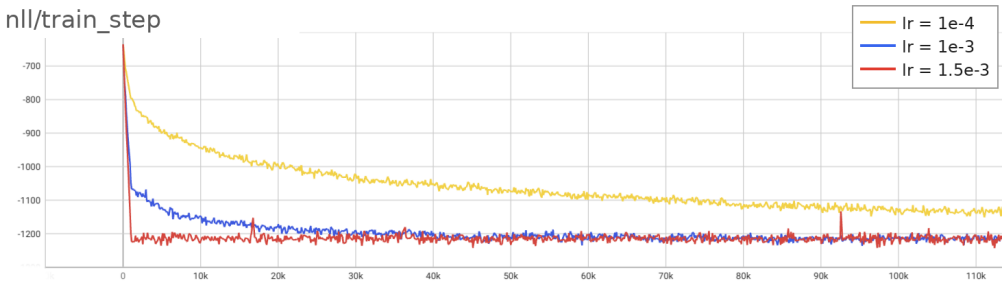



Figura 6: Pérdida NLL en el entrenamiento según el número de actualizaciones de nodos, sobre el conjunto de datos MNIST. Dimensión de enlace máxima de los nodos la red: 32. Tamaño de *batch*: 32. Función de *embedding*: polinomio de Legendre de orden 1.

La red se inicializa con tensores aleatorios (normalizados, obtenidos por factorización QR de matrices aleatorias), con el centro de ortogonalidad en el nodo hoja más a la derecha. El primer semibarrido recorre la red optimizando cada tensor por primera vez: al llegar a un nodo, se le aplica la actualización SGD, que lo lleva en un solo paso a un punto cercano a su óptimo condicionado al resto de la red. Es decir, el grueso de la mejora se produce aquí porque se están optimizando, uno tras otro, unos tensores que partían de valores puramente aleatorios. Cuando el barrido alcanza el último tensor, toda la red ha sido optimizada una vez, y los semibarridos posteriores ya solo encuentran nodos más próximos a su óptimo: la nueva actualización apenas los modifica, de modo que la curva de la pérdida se aplana.

En la Figura 6, se puede ver la evolución de la función de pérdida en el conjunto de aprendizaje del MNIST (conjunto de números dibujados a mano), según el número de nodos actualizados. Como las imágenes después del *padding* son de $32 \times 32$ píxeles, aplicando la Ecuación (3.5) se sabe que la red tiene un total de 1023 nodos. La meseta que se observa en el entrenamiento del modelo con mayor *learning rate* empieza precisamente en este paso, nada más acabar el primer semibarrido. Como se puede observar, el incremento en *learning rate* no supone en absoluto un deterioro en el régimen permanente de la pérdida.

Además, en el eje de ordenadas se aprecia cómo no sólo la pérdida se estabiliza en un valor negativo (alrededor de -1210), sino que empieza cerca de -650 antes de entrenar el modelo. Esto se debe al sesgo introducido por la función de pérdida de Legendre: como el modelo asigna mayor densidad de probabilidad a los píxeles con valores extremos, y las imágenes del MNIST están compuestas en su gran mayoría por píxeles totalmente blancos o negros, el modelo asigna una probabilidad exageradamente alta a las imágenes del *dataset* incluso

antes de ser entrenado en él.

Para apreciar la magnitud de este sesgo, se puede observar el comportamiento del modelo ante muestras de ruido binario o uniforme. En la Figura 7 se aprecia cómo el modelo asigna una probabilidad elevada al comienzo del entrenamiento (alrededor de 700, parecida en magnitud a la que asigna a las muestras del conjunto de datos en la Figura 6). Si bien la probabilidad de estas muestras aumenta sutilmente conforme se realizan más barridos de entrenamiento, alcanza un valle alrededor de 725, lo que indica que el modelo es capaz de distinguir entre estas muestras erróneas y aquéllas del conjunto de datos. Como se puede ver en la Figura 8, hay una gran diferencia entre la verosimilitud del ruido binario y el uniforme. El modelo comienza asignando una probabilidad alrededor de $e^{-200}$ a las muestras artificiales, y conforme avanza el entrenamiento va en descenso.

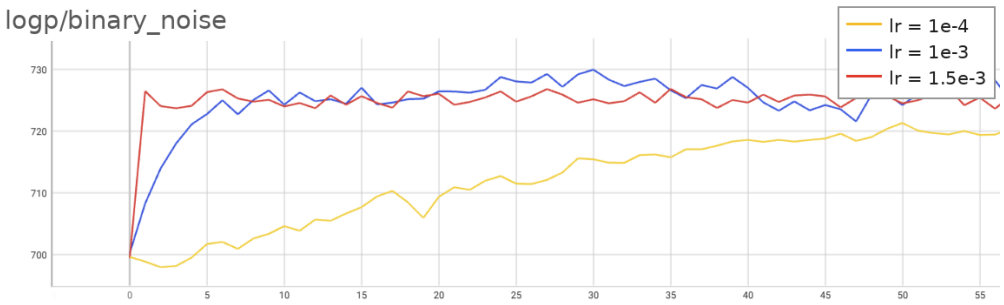



Figura 7: Log-probabilidad de muestras de ruido binario, según número de barridos en el entrenamiento.

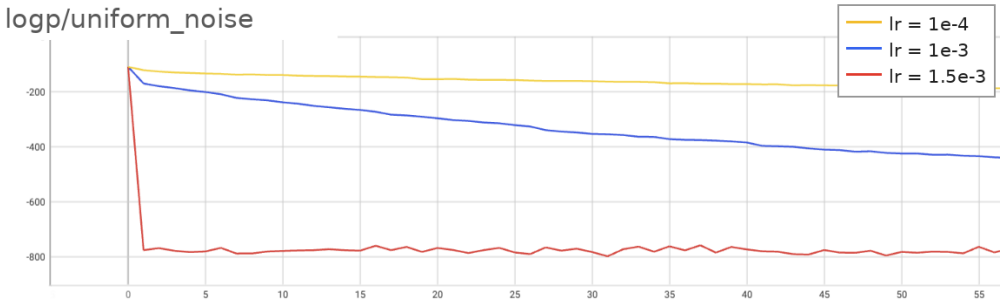



Figura 8: Evolución de la log-probabilidad que asigna cada modelo a muestras de ruido uniforme, según el número de barridos completos en el entrenamiento.

La rapidez de entrenamiento es una de las ventajas más notables del modelo frente a los generativos clásicos basados en redes neuronales. La diferencia de fondo es que aquí la función de partición $Z$ es tratable: gracias a la forma canónica se calcula de forma exacta, por lo que la verosimilitud puede optimizarse directamente, sin las aproximaciones estocásticas (muestreo MCMC, divergencia contrastiva, fases negativas aproximadas) [6], [7] que requieren los

modelos basados en energía y que ralentizan y dificultan su entrenamiento. Como consecuencia, el modelo alcanza una pérdida estable en un número reducido de barridos de la red, frente a las numerosas épocas de descenso por gradiente que necesitan otras arquitecturas.

## 3.8. Algoritmo de generación

Tras entrenar el modelo, generar una imagen nueva consiste en muestrear un punto de la densidad $p(x) = |\Psi|^2/Z$ del modelo. A diferencia de la mayoría de modelos generativos, la forma canónica de la red permite realizar dicho muestreo de forma exacta, sin tener que recurrir a métodos aproximados como cadenas de Markov o métodos de gradiente. Esto se debe a que el factor de normalización $Z$ se puede calcular de forma exacta a través de la Ecuación (3.24). Los píxeles de una imagen son generados de forma secuencial, siguiendo la regla de la cadena para descomponer el muestreo en un producto de condicionales:

$$p(x) = p(x_1, \ldots, x_N) = p(x_1) \cdot p(x_2 \mid x_1) \cdot \cdots \cdot p(x_N \mid x_1, \ldots, x_{N-1}) \tag{3.39}$$

Es decir, en vez de generar directamente todos los píxeles de la imagen, se muestrean individualmente, de forma condicionada a los píxeles ya generados. De este modo, el algoritmo comienza por el píxel más a la izquierda en el árbol de la red, y recorre todos los nodos de la capa de datos (píxeles de la imagen) según el recorrido del árbol. Para muestrear el píxel $x_k$ se necesita su densidad (probabilidad) condicionada al resto de píxeles ya generados: $p(x_k \mid x_1, \ldots, x_{k-1})$.

Para obtener esta probabilidad condicionada se puede dividir los píxeles de la imagen en dos categorías. En primer lugar, los píxeles ya muestreados tienen un valor concreto. Esos píxeles se fijan, sustituyendo los nodos correspondientes de la capa de datos por sus valores generados. En segundo lugar, los píxeles que aún no se han generado todavía no tienen un valor concreto. Para obtener la densidad condicionada $p(x_k \mid x_1, \ldots, x_{k-1})$, el efecto de todos los píxeles $\{x_{k+1}, \ldots, x_N\}$ se debe *marginalizar*, es decir, integrar a lo largo de su rango [0,1]. En el primer paso del muestreo, como aún no se ha fijado ningún píxel, todos menos el primero se marginalizan, quedando la expresión de la densidad:

$$p(x_1) = \int_0^1 p(x_1, x_2, \ldots, x_N)\, dx_2 \ldots dx_N \tag{3.40}$$

Desarrollando la regla de Born (Ecuación (3.8)):

$$p(x_1, \ldots, x_N) = \frac{|\Psi(x_1, \ldots, x_N)|^2}{Z} = \frac{\Psi(x_1, \ldots, x_N) \cdot \Psi(x_1, \ldots, x_N)}{Z} \tag{3.41}$$

Antes de entrar a la red como un nodo de la capa de datos, cada píxel pasa a través de una función de *embedding* $\phi$, como se ha visto en las secciones anteriores. Ahora bien, la amplitud $\Psi$ depende de cada píxel $x_n$ a través de su *embedding* $\phi(x_n)$, que es un vector de $D$ componentes $\phi_1(x_n), \ldots, \phi_D(x_n)$. Como en $|\Psi|^2$ la amplitud aparece dos veces, cada píxel aparece en esta fórmula como dos copias de su *embedding*: una procedente del primer factor

$\Psi$ y otra del segundo. La dependencia de $|\Psi|^2$ en un píxel concreto $x_n$ es, por tanto, de la forma $\phi_i(x_n) \cdot \phi_j(x_n)$.

Marginalizar ese píxel es integrarlo sobre todo su rango. Al hacerlo, lo único que depende de $x_n$ es ese producto de las dos copias, de modo que la integral recae únicamente sobre él:

$$\int_0^1 \phi_i(x_n)\phi_j(x_n)\, dx_n = G_{ij} \tag{3.42}$$

El resultado es exactamente la matriz de Gram (Ecuación (3.25)) del *embedding* $\phi$, introducido anteriormente. Esto significa que marginalizar un píxel equivale a reemplazar sus dos patas físicas por la matriz de Gram $G$. Y como el *embedding* elegido es ortonormal ($G = \mathbb{I}$), esta sustitución es equivalente a conectar mediante un enlace los dos índices físicos del nodo. Así, la marginalización se lleva a cabo mediante una contracción de un índice físico con otro.

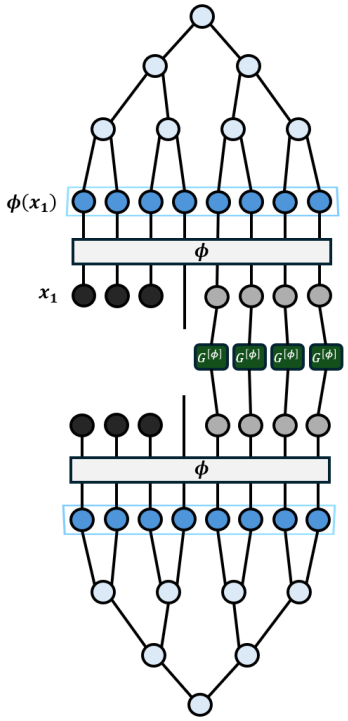



Figura 9: Diagrama de la generación de un píxel $x_k$. La red aparece duplicada, ya que la regla de Born eleva $\Psi$ al cuadrado, lo que equivale a contraerla con una copia idéntica suya. Los nodos negros son los píxeles ya generados, $\{x_1, \ldots, x_{k-1}\}$, cuyo valor se fija; los grises son los píxeles pendientes, $\{x_{k+1}, \ldots, x_N\}$, que se marginalizan enlazándolos a la matriz de Gram del *embedding*, $G^{[\phi]}$, a través de un índice conjunto. Los dos índices que quedan libres corresponden al píxel $x_k$ que se va a generar.

En el diagrama de la Figura 9, cuando el *embedding* es ortonormal ($G^{[\phi]} = \mathbb{I}$) el enlace a la matriz de Gram equivale a conectar cada nodo gris con su simétrico. Tras estas contracciones

quedan dos índices sueltos (los del píxel $x_k$), de modo que el resultado de contraer todo el sistema es un tensor de orden 2, es decir, una matriz. Esta matriz suele recibir el nombre de *matriz de densidad reducida* ($\rho$) en la literatura de redes de tensores, y contiene toda la información necesaria para generar $x_k$: su densidad condicional se obtiene cerrando los dos índices libres con el *embedding* del valor a evaluar, $p(x_k \mid x_1, \ldots, x_{k-1}) \propto \phi(x_k)^T \rho\, \phi(x_k)$, de la que se muestrea por transformada inversa.

Tras fijar los píxeles anteriores y marginalizar los posteriores, el resultado es una función que depende de un solo píxel: su densidad condicional $p(x_k \mid x_1, \ldots, x_{k-1})$. Las contracciones descritas dejan esta densidad en función de una sola variable sobre el intervalo $[0, 1]$. En el caso del *embedding* de Legendre, esto es simplemente un polinomio de grado 2 en $x_k$. Generar el píxel se reduce entonces a muestrear un valor de esa función de densidad, un problema sencillo al ser unidimensional y estar acotado.

Tener la expresión de la densidad de forma explícita sobre un intervalo finito permite muestrear de ella de forma exacta, mediante el método de la transformada inversa [23], empleado también en otros modelos generativos de redes tensoriales sobre datos continuos [8]. La idea es la siguiente: se calcula primero la función de distribución acumulada (CDF), que integra la densidad desde el inicio del intervalo:

$$F(t) = \int_0^t p(x_k \mid x_1, \ldots, x_{k-1})\, dx_k \tag{3.43}$$

una función monótona que crece de 0 a 1 a lo largo de $[0, 1]$. Para generar el píxel se toma un número aleatorio $u$ uniforme en $[0, 1]$ y se busca el valor $x_k$ tal que $F(x_k) = u$, es decir, $x_k = F^{-1}(u)$. Este procedimiento produce una muestra exacta de la densidad, puesto que los valores de $x_k$ son seleccionados de forma proporcional a la densidad que les asigna el modelo.

Una vez muestreado $x_k$, su valor se fija (sustituyendo su nodo por $\phi(x_k)$, como se describió antes) y se prosigue con el siguiente píxel. Al repetir el proceso hasta el último píxel $x_N$, se obtiene una imagen completa. Por la regla de la cadena, el producto de todas las condicionales muestreadas es igual a la densidad conjunta $p(x_1, \ldots, x_N)$. Para la generación de imágenes a color, el proceso es similar, salvo que se muestrea secuencialmente los canales de cada píxel.

Adicionalmente, esta implementación cuenta con un parámetro de temperatura para la generación. Esta es una práctica común en modelos generativos [13] que permite regular el grado de nitidez o diversidad de las muestras. Esto se lleva a cabo elevando la densidad condicional de cada píxel antes de muestrearlo a la potencia $1/\tau$ y normalizándola de nuevo:

$$p_T(x_k) \propto p(x_k \mid x_1, \ldots, x_{k-1})^{1/\tau} \tag{3.44}$$

El efecto de $\tau$ sobre la forma de la densidad es el siguiente: con $\tau = 1$, se mantiene la densidad original del modelo. Para $0 < \tau < 1$, se acentúa la probabilidad de los valores (aquéllos más probables se vuelven aún más probables, a costa de que los demás pierdan densidad). Esto da lugar a imágenes más nítidas que el muestreo simple, pero reduce su diversidad. En el caso extremo, $\tau = 0$, toda la densidad colapsa sobre el valor más probable, de modo que se obtiene un muestreo determinista sobre la moda.

## 3.9. Resultados del entrenamiento

Para validar el modelo, se ha comenzado por el conjunto de datos MNIST [24], un punto de partida natural por su simplicidad y por ser el banco de pruebas habitual en los trabajos previos sobre máquinas de Born tensoriales [6], [7]. El MNIST está compuesto por 70.000 imágenes en escala de grises de dígitos manuscritos (del 0 al 9), repartidas en 60.000 muestras de entrenamiento y 10.000 de test. Cada imagen tiene una resolución de $28 \times 28$ píxeles, con un único canal de intensidad por píxel en el rango $[0, 1]$. Antes de entrar en la red, las imágenes se rellenan con *padding* hasta alcanzar una dimensión de $32 \times 32$, de modo que el número de píxeles sea una potencia de dos y la red de árbol binario quede completa (ver Ecuación (3.5)). Como el fondo de estas imágenes es negro, el *padding* rellena los píxeles faltantes de este color.

Se trata, por tanto, de datos en escala de grises (unidimensionales por píxel) y, en la práctica, casi binarios: la inmensa mayoría de los píxeles son blancos o negros, con relativamente pocos valores intermedios. Esta característica condiciona la elección de la función de codificación. Salvo que se indique lo contrario, en todos los experimentos se emplea el *embedding* de Legendre de orden 1 descrito en la sección anterior, $\phi(x) = [1, \sqrt{3}(2x - 1)]$, por ser la opción que mejor compromiso ofrece para esta tarea: su matriz de Gram es la identidad (Ecuación (3.26)), por lo que la densidad está correctamente normalizada, y a la vez distingue entre píxeles negros y blancos, a diferencia de las codificaciones angulares degeneradas. El sesgo hacia los valores extremos que introduce (Ecuación (3.32)) resulta, además, especialmente adecuado para la naturaleza casi binaria del MNIST.

### 3.9.1. Diferencia entre *embeddings* de Legendre de distinto grado

Para validar el impacto de aumentar la dimensión de los *embeddings* y, por lo tanto, la dimensión de enlace de las capas más cercanas a los datos (según Ecuación (3.16)), se ha entrenado dos modelos usando como codificación diferente grado de Legendre. Estos modelos han sido entrenados sobre un subconjunto del MNIST, que contiene solo 8s, ya que se ha comprobado empíricamente mediante la NLL que es el caracter más complicado de representar para la red. De esta forma, la diferencia entre ambos modelos debería ser más notable.

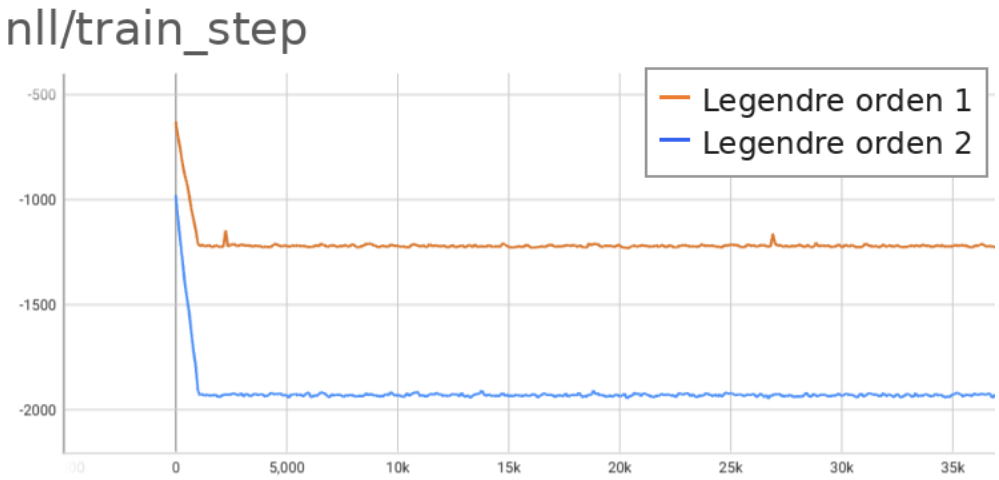



Figura 10: Pérdida NLL del modelo en el conjunto de entrenamiento, según el número de actualizaciones de nodos. Ambas líneas representan al mismo modelo ($B_{\text{máx}} = 32$), pero con diferente función de codificación.

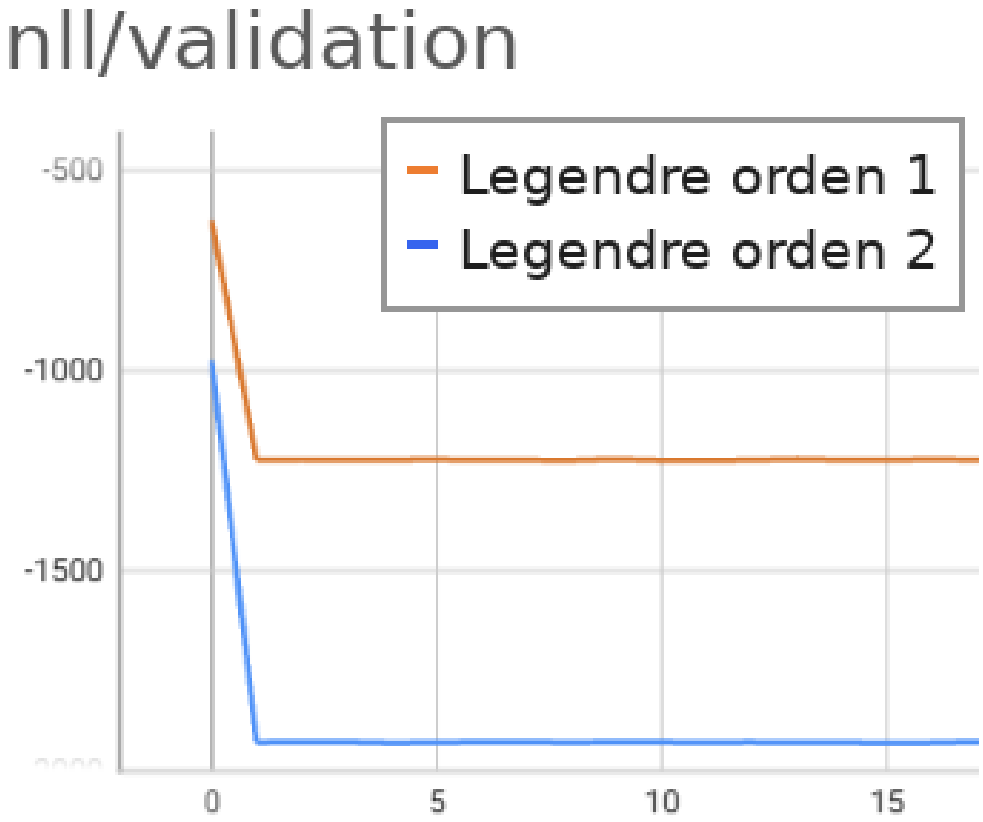



Figura 11: Pérdida NLL del modelo en el conjunto de validación, según barrido.

La Figura 10 muestra la diferencia de la pérdida de log-verosimilitud negativa en función del *embedding* utilizado. Como se menciona en la sección de *Embeddings*, aumentar el orden del polinomio de Legendre incrementa asimismo el sesgo hacia los extremos de la distribución de probabilidad que define el modelo. Por esto mismo, la magnitud de estas dos pérdidas no puede compararse justamente. Para ello, se puede utilizar el margen entre la verosimilitud de las muestras reales del MNIST y ruido uniforme o binario:

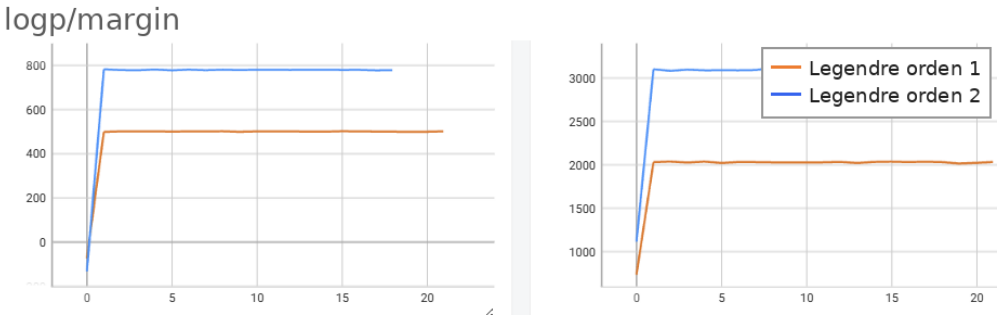



Figura 12: Margen entre la verosimilitud de muestras del MNIST y ruido, según grado de Legendre. A la izquierda, margen de MNIST con respecto a ruido binario. A la derecha, MNIST frente a ruido uniforme.

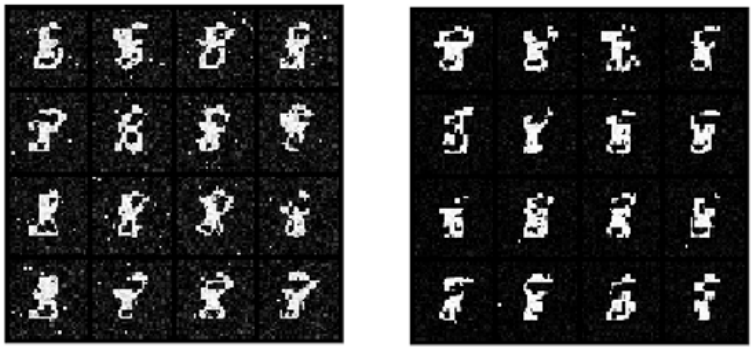


Figura 13: Muestras generadas por cada modelo para $\tau = 0{,}3$. Izquierda: Legendre de orden 1. Derecha: orden 2.

En términos de rapidez de entrenamiento, el modelo que utiliza la base de orden 1 ha tardado 1:31.58 minutos en realizar los 10 primeros barridos. El modelo de Legendre de orden 2, al tener un número superior de parámetros, ha sido algo más lento, tardando 2:11.39 minutos (alrededor del 144 % del tiempo del modelo anterior).

En la Figura 12 se observa cómo el margen entre la log-probabilidad que asigna el modelo al ruido es superior para el embedding de Legendre de orden 2. A priori, podría interpretarse este resultado como una mejora. Sin embargo, nuevamente esto puede deberse a que el sesgo del modelo (que aumenta al aumentar el grado de la codificación) favorece a la NLL para este conjunto de datos. Como se trata de modelos generativos, pueden ser comparados en atención a las imágenes que generan. En la Figura 13 se aprecia una clara reducción del ruido en el modelo más pesado con respecto al modelo simple, a costa de que los números generados son peores. Dicho esto, ambos modelos han sido capaces de aprender la distribución de los datos de entrada y generar muestras satisfactorias, en especial para el tiempo de entrenamiento que han requerido.

### 3.9.2. Dimensión de enlace

La dimensión de enlace máxima $B_{\text{máx}}$ es el hiperparámetro que gobierna el compromiso entre capacidad y coste del modelo. Cada nodo es un tensor de tamaño $B \times B_l \times B_r$, de modo que el número de parámetros crece con $B$, y con él la memoria y el tiempo de cómputo de cada contracción. Pero su efecto va más allá del coste: como se vio en Ecuación (3.16), la dimensión de enlace acota la cantidad de correlación (o entrelazamiento) que la red puede capturar entre los dos subárboles que une cada nodo [10]. Una $B_{\text{máx}}$ pequeña limita esa correlación, restringiendo el espacio de distribuciones representables (puede provocar *subajuste*, pero también actúa como regularización); una $B_{\text{máx}}$ grande la amplía, a costa de más parámetros y, potencialmente, mayor riesgo de sobreajuste.

Lo relevante es que la relación entre $B_{\text{máx}}$ y el rendimiento no es evidente a priori: aumentar la capacidad no garantiza un mejor modelo. Si las correlaciones del conjunto de datos ya quedan capturadas con una dimensión de enlace reducida, el incremento adicional no se traduce en una mejora apreciable de la verosimilitud ni de la calidad de las muestras, y solo añade parámetros y tiempo de entrenamiento. Para determinar empíricamente si ese es el caso (hasta qué punto se aprovecha de verdad la capacidad extra) se ha entrenado cinco modelos idénticos salvo por su dimensión de enlace máxima, uno con $B_{\text{máx}} = 32$, otro con $B_{\text{máx}} = 16$, $B_{\text{máx}} = 8$, $B_{\text{máx}} = 4$ y $B_{\text{máx}} = 2$.

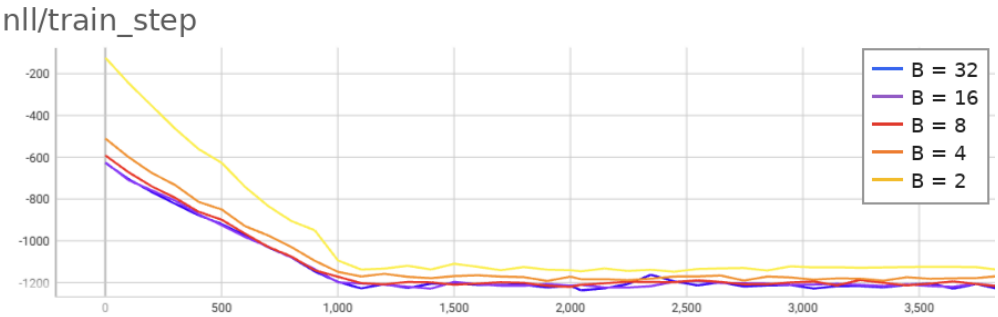



Figura 14: Pérdida NLL en el entrenamiento según diferente dimensión de enlace máxima $B_{\text{máx}}$.

Los resultados verifican la hipótesis inicial: el incremento de la dimensión de enlace máxima por sí solo no produce un descenso de la pérdida del modelo. Para este conjunto de datos, el rendimiento del modelo $B_{\text{máx}} = 32$ es idéntico al de $B_{\text{máx}} = 16$, e incluso al de $B_{\text{máx}} = 8$. Las muestras generadas por estos estos tres modelos son indistinguibles, tanto a simple vista como por su NLL (Figura 14). Esto es una clara muestra de que el incremento de la capacidad expresiva del modelo no está siendo utilizado en este conjunto de datos. Sin embargo, para un *dataset* más exigente habría que aumentar $B_{\text{máx}}$.

### 3.9.3. Diferencia en la temperatura para el muestreo

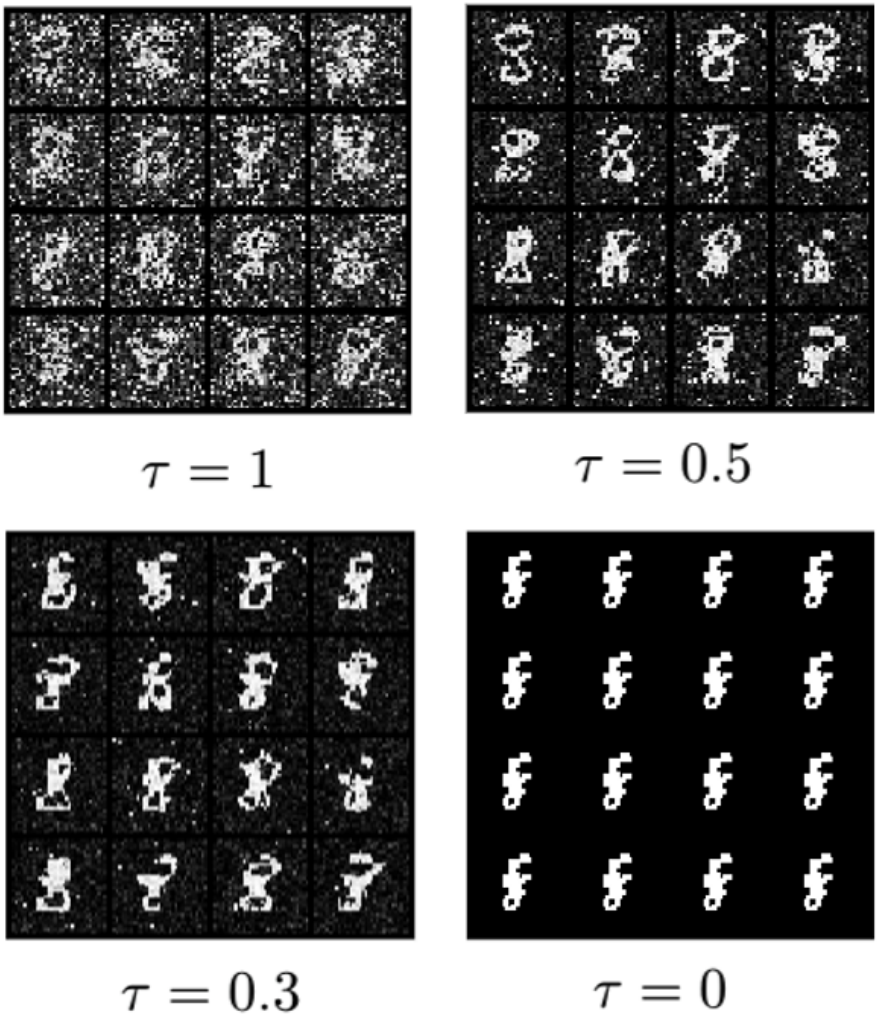



Figura 15: Distintas muestras generadas por un modelo según temperatura. El modelo ha sido entrenado con un subconjunto de 5000 imágenes del MNIST, todas ellas de 8s. *Embedding*: Legendre de orden 1 (2D). $B_{\text{máx}} = 32$, número de barridos: 1. Tiempo total de entrenamiento: 17,24s. Tiempo promedio de generación por muestra: 0,17s.

En la Figura 15 se puede ver claramente cómo afecta el valor de $\tau$ a las muestras generadas por el modelo. El muestreo simple ($\tau = 1$) es propenso a producir ruido: los valores que acumulan mayor densidad siguen teniendo una probabilidad despreciable en comparación con el resto del dominio. Al reducir $\tau$, por tanto, también decrece la varianza de las muestras,

y con ella su ruido. Finalmente, para $\tau = 0$ el muestreo es determinista, motivo por el cual todas las imágenes generadas son idénticas. El hecho de que en este último muestreo todos los píxeles sean prácticamente blancos o negros no es casualidad. Además de la naturaleza del conjunto de datos, que está compuesto de píxeles mayoritariamente blancos y negros, es consecuencia del sesgo a priori introducido por los *embeddings* de Legendre (Ecuación (3.32)), que tiene mayor impacto en este caso extremo de $\tau = 0$. El punto de equilibrio entre nitidez y variedad de las muestras se encuentra alrededor de $\tau = 0{,}3$ para este conjunto de datos en específico.

La elección de $\tau = 0{,}3$ se ha apoyado hasta aquí en la inspección visual de las muestras. Para justificarla de forma cuantitativa conviene una métrica que compare las imágenes generadas con las reales. Nótese que la verosimilitud exacta del modelo no sirve a este fin, ya que la temperatura solo reescala la distribución de muestreo y no la densidad $p(x)$ que el modelo asigna; se necesita, por tanto, una métrica sobre las propias muestras. La más extendida es la *Fréchet Inception Distance* (FID) [25].

La FID mide la distancia entre la distribución de las imágenes reales y la de las generadas en un espacio de características. Para ello, ambos conjuntos de imágenes se hacen pasar por una red Inception-v3 preentrenada, y se toman las activaciones de una de sus capas profundas (un vector de 2048 componentes por imagen). Cada conjunto de características se aproxima por una distribución gaussiana multivariante, caracterizada por su media y su matriz de covarianza ($\mu_r, \Sigma_r$ para las reales y $\mu_g, \Sigma_g$ para las generadas), y la FID se define como la distancia de Fréchet entre ambas gaussianas:

$$\mathrm{FID} = \|\mu_r - \mu_g\|^2 + \mathrm{Tr}\big(\Sigma_r + \Sigma_g - 2\,(\Sigma_r \Sigma_g)^{1/2}\big) \tag{3.45}$$

Un valor menor indica que las imágenes generadas son, en distribución, más parecidas a las reales: el primer término penaliza la diferencia entre las medias de las características, y el segundo, la diferencia entre sus covarianzas (su dispersión y correlaciones).

La Figura 16 muestra la FID obtenida para cada temperatura, calculada sobre las muestras generadas por el modelo frente a un conjunto de referencia de imágenes reales de MNIST. La curva presenta un mínimo claro en $\tau = 0{,}3$: la FID desciende desde unos 296 en $\tau = 0$ hasta su valor mínimo (alrededor de 268) en $\tau = 0{,}3$, y a partir de ahí crece de forma monótona hasta superar los 420 en $\tau = 1$. Esto confirma de forma cuantitativa la elección anterior. En los extremos, el muestreo a $\tau = 1$ produce la peor FID, penalizado por el ruido de las muestras, mientras que $\tau = 0$ también empeora respecto al óptimo, en este caso por la falta de diversidad al colapsar sobre la moda. El valor $\tau = 0{,}3$ alcanza el mejor compromiso entre nitidez y variedad, en coherencia con lo observado cualitativamente en la Figura 15.

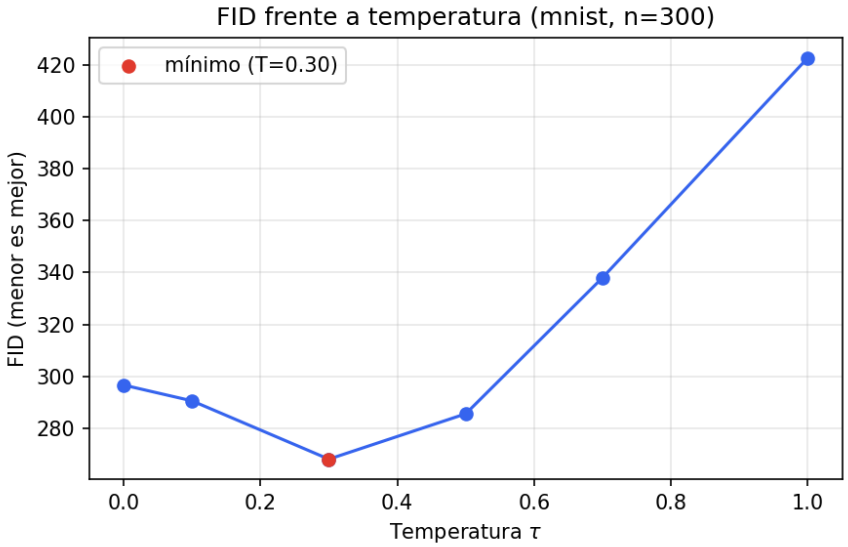



Figura 16: FID (*Fréchet Inception Distance*) de las muestras generadas frente a imágenes reales de MNIST, según la temperatura de muestreo $\tau$. Un valor menor indica mayor parecido con los datos reales. El mínimo se alcanza en $\tau = 0{,}3$.

Cabe mencionar que la FID emplea una red Inception preentrenada sobre ImageNet (imágenes a color naturales), por lo que su aplicación a dígitos en escala de grises (replicados a tres canales) es una aproximación. Además, su valor absoluto se ve inflado por el número finito de muestras empleadas (2000 por punto, para garantizar estabilidad). No obstante, lo relevante para esta discusión es la posición del mínimo, que debería ser robusta a estas consideraciones.

Figura 17: Muestras aleatorias generadas por un modelo ($\tau = 0{,}3$). Este modelo fue entrenado con todo el conjunto de *train*, tiene $B_{\text{máx}} = 64$ y utiliza *embeddings* de Legendre de orden 1. El tamaño de *batch* utilizado es de 128, y el tiempo de entrenamiento ha sido de 3min 15,21s.

### 3.9.4. Desempeño en generación de imágenes a color

El último experimento realizado tiene como objetivo comprobar cómo rinde el modelo ante un conjunto de datos a color. El *dataset* de elección es el CIFAR-10 [26]. A diferencia de MNIST, CIFAR-10 está formado por 60.000 imágenes en color (50.000 de entrenamiento y 10.000 de test), repartidas en 10 clases de objetos naturales (aviones, automóviles, pájaros, gatos...). Cada imagen tiene una resolución de $32 \times 32$ píxeles con tres canales de color (RGB), por lo que no requiere *padding* y encaja directamente en la red. Su naturaleza es radicalmente distinta a la de los dos conjuntos anteriores: los píxeles ya no son casi binarios, sino valores reales sobre un dominio tridimensional y continuo, con texturas, gradientes y correlaciones de color mucho más ricas:

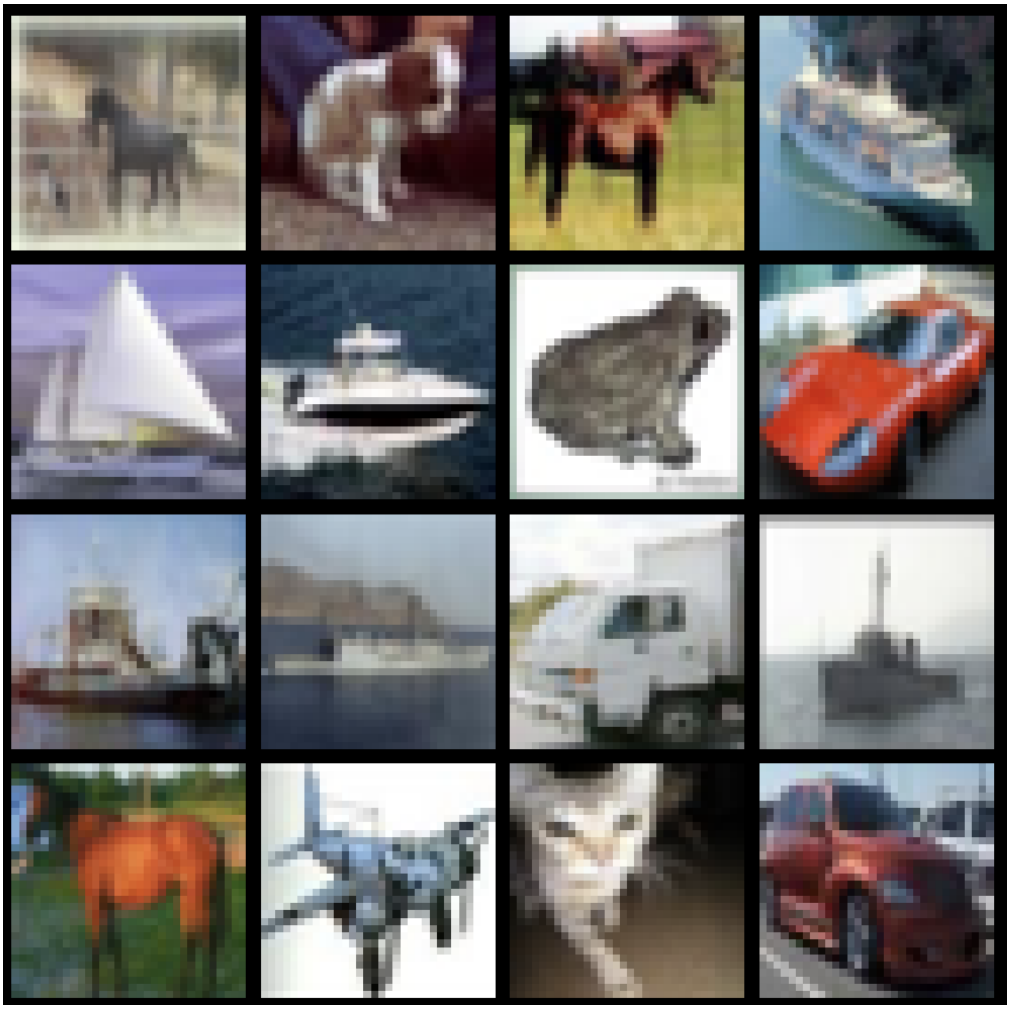


Figura 18: Algunas imágenes del conjunto de datos CIFAR10.

En el apartado de Embeddings se ha visto cómo emplear la función de codificación de Legendre resulta en el sesgo del modelo hacia los extremos. En particular, se ha visto cómo aumenta dicho sesgo en función del orden de la base (Ecuación (3.33)). Esto es lo que ocurre precisamente al utilizar el producto tensorial de esta codificación consigo misma para los tres canales R, G y B de una imagen a color. Si para una base de grado 2 el ratio entre la densidad a priori de dos píxeles antes podía llegar a ser de $1:4$, utilizar $\phi(R)\otimes\phi(G)\otimes\phi(B)$ hace que este ratio sea de $1:4^3=64$. Resultaría sorprendente que un modelo con tal cantidad de sesgo fuera capaz de entrenar correctamente. Además, estas imágenes son mucho más difíciles de modelar de por sí, al ser infinitamente más complejas que las del MNIST.

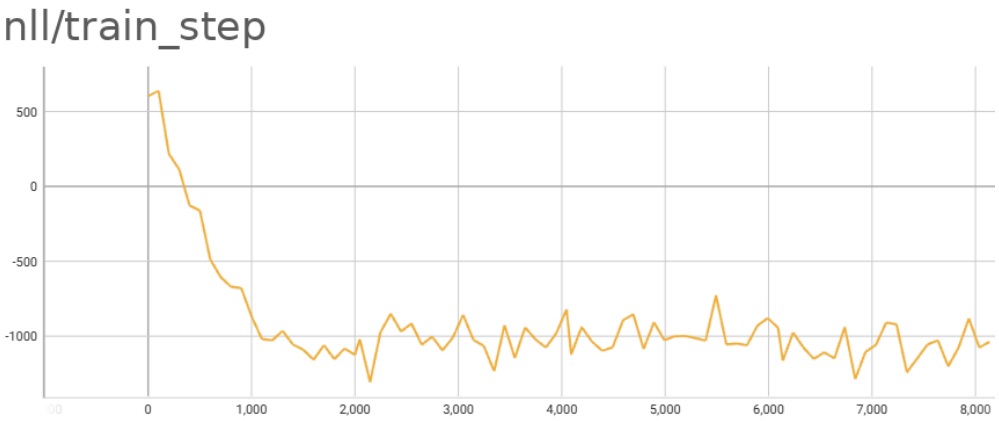



Figura 19: NLL a lo largo del entrenamiento sobre CIFAR10 de un modelo $B_{\text{máx}} = 64$, utilizando *embeddings* de Legendre de orden 1 para RGB (producto tensorial de codificación por canal).

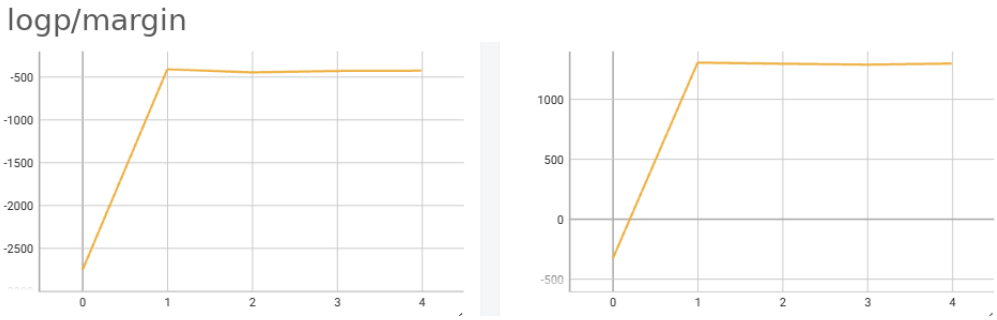



Figura 20: Margen entre la verosimilitud de muestras del CIFAR10 y ruido binario (izquierda) y uniforme (derecha), en el conjunto de validación.

En la Figura 19 se observa una curva de NLL más ruidosa que las de los modelos entrenados con el MNIST. Además, esta vez la pérdida empieza en positivo (alrededor de 530), ya que las imágenes del *dataset* generalmente no tienen píxeles con valores extremos en ninguno de los canales. La primera indicación de que el modelo entrenado está fallando estrepitosamente es el primer gráfico de la Figura 20, el margen entre la log-probabilidad que asigna el modelo a ruido binario y a los datos reales. Este margen es en todo momento negativo, lo que significa que el modelo atribuye mayor densidad de probabilidad a estas muestras de ruido puro. Además, el margen disminuye en valor absoluto conforme acaba el entrenamiento, lo que significa que el modelo pierde capacidad de diferenciación entre los dos tipos de muestras.

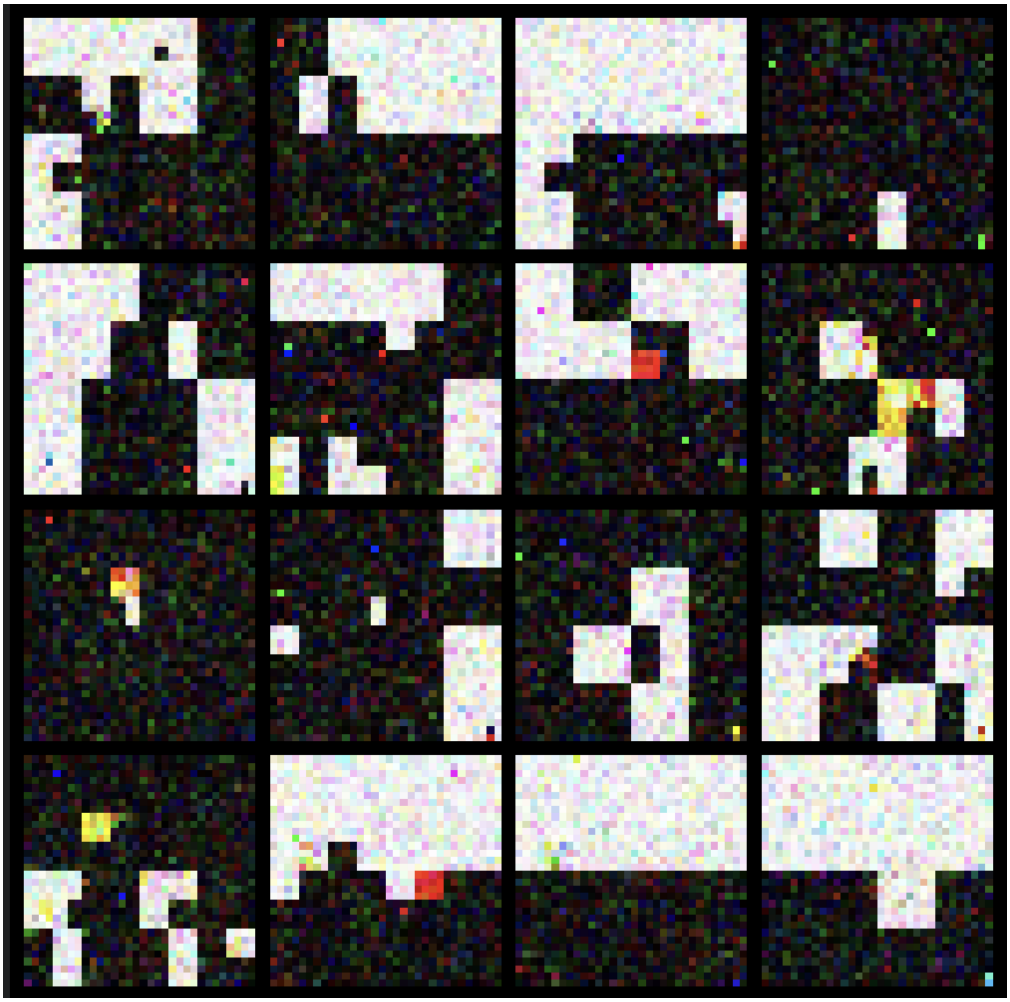


Figura 21: Muestras generadas por el modelo, para $\tau = 0{,}4$. La calidad de las muestras es similar para todos los valores de $\tau$.

Las imágenes generadas por el modelo confirman la sospecha inicial: el sesgo de la densidad de probabilidad es de tal magnitud que las imágenes que maximizan la verosimilitud tienen valores extremos en todos sus píxeles. En particular, se observan regiones oscuras y otras prácticamente blancas, así como otras en las que uno o dos de los canales valen 1 y los demás 0 (amarillo, azul, rojo, verde...). Como cabía esperar según el razonamiento anterior, el modelo ha fallado de forma catastrófica. No obstante, la imagen generada no es completamente inútil: sirve para ilustrar la correlación que captura la geometría de la red, formando artefactos cuadrados o rectangulares de diferente tamaño según la produndidad de los enlaces.

## 3.10. Implementación

### 3.10.1. Tecnologías y recursos HW/SW empleados

El modelo se ha implementado íntegramente en `Python` (versión 3.12) sobre `PyTorch`, que se emplea como motor de álgebra tensorial: los nodos de la red son tensores de `PyTorch` y todas las operaciones (contracciones mediante `einsum`, factorizaciones QR, descomposiciones para la canonización) se apoyan en su biblioteca de álgebra lineal y en su sistema de diferenciación automática, que proporciona el gradiente de la verosimilitud respecto al nodo central. Se utiliza `torchvision` para la descarga y el preprocesado de los conjuntos de datos (MNIST, Fashion-MNIST y CIFAR-10) y para componer las rejillas de imágenes generadas, `TensorBoard` para el registro de métricas y muestras durante el entrenamiento, `tqdm` para el seguimiento del progreso y `Matplotlib` para las figuras. Todos los cálculos se realizan en doble precisión (`float64`), una elección deliberada para preservar la estabilidad numérica de las factorizaciones QR repetidas que sostienen la forma canónica.

En cuanto al hardware, el código está escrito para ejecutarse de forma transparente tanto en CPU como en GPU: el dispositivo se selecciona automáticamente a través de la interfaz de aceleradores de `PyTorch`, recurriendo a la CPU cuando no hay GPU disponible. Los experimentos se han ejecutado sobre una GPU `NVIDIA` 4070 para portátiles con soporte `CUDA`, lo que acelera notablemente las contracciones, especialmente en los embeddings cuya función de partición exige la contracción de doble capa.

### 3.10.2. Estructura del código

El código se organiza como un paquete de `Python` (`src/`) con una separación clara de responsabilidades:

- `model.py` — define la red `BinaryTTN` y sus capas: la construcción del árbol binario, la contracción de la red dada una entrada, la canonización y el cálculo de la función de partición, así como la serialización del modelo (`save`/`from_file`).
- `qr.py` — rutinas de factorización QR sobre tensores y de absorción del residuo en el nodo contiguo, que implementan el desplazamiento del centro de ortogonalidad.
- `mnist.py` — carga de los conjuntos de datos (con filtrado opcional por clase), el *padding* a potencia de dos y las funciones de codificación (*embeddings*) con sus matrices de Gram: Legendre de orden 1 y 2, angular, y los productos tensoriales para color.
- `trainer.py` — la clase `Trainer`, que encapsula el bucle de barrido, los optimizadores por nodo, el cálculo de métricas, el registro en `TensorBoard` y el guardado de *checkpoints*.
- `generation_loop.py` — el muestreo exacto secuencial *top-down* con temperatura.

- Puntos de entrada (*scripts* de línea de comandos): `train_sweep.py` y `train_mixture.py` para entrenar, y `generate.py`, `generate_temps.py` y `sample_mixture.py` para generar.

### 3.10.3. Pipeline (preprocesado, entrenamiento, evaluación)

**Preprocesado.** Cada imagen se convierte a tensor y se escala al rango $[0, 1]$, se rellena con *padding* hasta una dimensión potencia de dos ($32 \times 32$) y se transforma con la función de *embedding* elegida; estas operaciones se encadenan en una única transformación de `torchvision`. Opcionalmente, el conjunto puede restringirse a una sola clase, lo que permite entrenar generadores especializados.

**Entrenamiento.** Los datos se sirven en *batches* mediante un `DataLoader` con barajado, y el `Trainer` ejecuta el algoritmo de barrido descrito en la Sección 3, actualizando en cada paso únicamente el nodo central. El optimizador por defecto es la actualización exacta de punto fijo, válida cuando la matriz de Gram es la identidad; cuando no lo es, se selecciona automáticamente una variante generalizada. Durante el entrenamiento se guardan *checkpoints* del mejor modelo y del último (`best.pt`, `last.pt`) junto con instantáneas periódicas.

**Evaluación.** Como se argumentó en la Sección 3.3, la NLL absoluta de un modelo continuo no es interpretable, por lo que la validación se apoya en métricas *relativas*: se registra la log-probabilidad de un subconjunto de validación frente a la de ruido (binario y uniforme) y su diferencia. Cualitativamente, el modelo se evalúa generando rejillas de muestras a distintas temperaturas y comparándolas con imágenes reales del conjunto de datos.

### 3.10.4. Reproducibilidad

La reproducibilidad se sustenta en tres elementos. En primer lugar, el entorno de ejecución está fijado: el fichero `requirements.txt` congela las versiones exactas de todas las dependencias, y los conjuntos de datos son estándar y se descargan de forma versionada a través de `torchvision`. En segundo lugar, los modelos entrenados se serializan junto con sus metadatos (forma de entrada, dimensión física y matriz de Gram), de modo que cualquier resultado puede reproducirse cargando el *checkpoint* correspondiente sin necesidad de reentrenar. En tercer lugar, la generación es determinista a partir de una semilla: los *scripts* de muestreo aceptan un parámetro `--seed` que fija el generador de números aleatorios, lo que garantiza que las mismas muestras puedan regenerarse de forma idéntica. La reproducción *bit a bit* del propio entrenamiento desde cero requeriría además fijar la semilla global del generador de `PyTorch` antes de la inicialización aleatoria de la red.

# Capítulo 4 Resultados

El resultado principal de este trabajo es la *generalización del modelo de máquina de Born tensorial del dominio discreto al continuo*. Partiendo del modelo binario de [6], se ha formulado la regla de Born sobre una Tree Tensor Network cuyos píxeles toman valores reales en $[0, 1]$, y se han trasladado a este nuevo escenario las propiedades que distinguen a estos modelos: la densidad está bien definida y correctamente normalizada, la función de partición $Z$ se calcula de forma exacta gracias a la forma canónica, y el muestreo es exacto y secuencial. Todo el aparato teórico desarrollado en la Sección 3 (la condición de ortonormalidad del *embedding* expresada mediante la matriz de Gram, el cálculo de $Z$ como la norma del nodo central, el algoritmo de barrido y la generación por marginalización y transformada inversa) es la materialización de esa generalización. Una vez establecida, el resto del trabajo se ha dedicado a estudiar empíricamente cómo se comporta el modelo en la práctica, qué factores de diseño condicionan su rendimiento y dónde están sus límites. Esta sección recapitula esos experimentos y las conclusiones que se desprenden de ellos.

## 4.1. La función de codificación como factor de diseño determinante

La conclusión transversal a todos los experimentos es que la función de *embedding* es el factor que más condiciona el comportamiento del modelo. Como se argumentó en la Sección 3, ninguna de las codificaciones estudiadas satisface de forma simultánea las tres propiedades deseables (ortonormalidad, no degeneración y norma de salida constante), de modo que toda elección implica una renuncia. La base de Legendre de orden 1 fue la opción adoptada por defecto: su matriz de Gram es la identidad (Ecuación (3.26)), lo que garantiza la normalización, y distingue entre píxeles negros y blancos, a diferencia de las codificaciones angulares, que son degeneradas. El precio que se paga es un sesgo *a priori* hacia los valores extremos de intensidad (Ecuación (3.32)), que resulta inofensivo (e incluso beneficioso) en datos casi binarios como el MNIST, pero que, como se verá, se vuelve catastrófico en datos a color.

Para cuantificar el efecto del grado de la codificación se compararon dos modelos idénticos entrenados sobre un subconjunto del MNIST (formado únicamente por dígitos 8, el carácter empíricamente más difícil de representar), uno con la base de Legendre de orden 1 y otro con la de orden 2. La principal lección de este experimento es metodológica: las pérdidas NLL de ambos modelos (Figuras 10 y 12) no son directamente comparables, porque al aumentar el grado de la base se incrementa también el sesgo del modelo, lo que infla artificialmente la verosimilitud que asigna a este conjunto de datos. De hecho, atendiendo solo a la NLL o al margen frente al ruido, el modelo de orden 2 parecería mejor; pero esa ventaja es en buena medida un artefacto del sesgo. La comparación justa entre dos modelos generativos es cualitativa, a través de las muestras que generan: en la Figura 13 se aprecia que el modelo de orden 2 reduce el ruido a costa de generar dígitos peores. En cualquier caso, ambos modelos aprenden la distribución de los datos y producen muestras satisfactorias, especialmente si se tiene en cuenta lo reducido del tiempo de entrenamiento.

## 4.2. Capacidad del modelo y dimensión de enlace

Un segundo experimento estudió el papel de la dimensión de enlace máxima $B_{\text{máx}}$, el hiperparámetro que regula el compromiso entre la capacidad expresiva de la red y su coste. Se entrenaron cinco modelos idénticos salvo por su $B_{\text{máx}}$ (32, 16, 8, 4 y 2). El resultado, recogido en la Figura 14, confirma que aumentar la capacidad no garantiza por sí solo un mejor modelo: para el MNIST, los modelos con $B_{\text{máx}} = 32$, 16 y 8 son indistinguibles, tanto en su NLL como en las muestras que generan. Esto indica que las correlaciones de este conjunto de datos ya quedan capturadas con una dimensión de enlace modesta, y que la capacidad adicional no se aprovecha. La consecuencia práctica es doble: por un lado, para un conjunto sencillo conviene mantener $B_{\text{máx}}$ bajo, lo que reduce parámetros y tiempo de cómputo sin penalización; por otro, cabe esperar que un conjunto de datos más exigente sí requiera una dimensión de enlace mayor para no incurrir en subajuste.

## 4.3. Generación de muestras y control por temperatura

La generación exacta es una de las ventajas más notables del modelo, y el mecanismo de temperatura $\tau$ permite regular el compromiso entre nitidez y diversidad de las muestras sin reentrenar. La Figura 15 ilustra este efecto: el muestreo a $\tau = 1$ (la distribución real del modelo) tiende a producir ruido, ya que la densidad acumulada en los modos sigue siendo pequeña frente al resto del dominio; al reducir $\tau$ disminuyen la varianza y el ruido, hasta el caso determinista $\tau = 0$, en el que todas las muestras colapsan sobre la moda. El punto de equilibrio entre nitidez y variedad se sitúa en torno a $\tau = 0{,}3$ para el MNIST. Resulta ilustrativo que en el caso extremo $\tau = 0$ las imágenes generadas sean prácticamente blanquinegras: además de la naturaleza casi binaria de los datos, es una manifestación directa del sesgo de la codificación de Legendre. Un modelo entrenado sobre el MNIST completo (Figura 17) genera muestras claramente reconocibles como dígitos, lo que confirma que la generalización al dominio continuo funciona en la práctica sobre datos reales en escala de grises. Conviene subrayar, además, la *rapidez* de todo el proceso: los modelos se entrenan en cuestión de minutos (del orden de minutos para el conjunto completo del MNIST) y cada muestra se genera en fracciones de segundo, lo que constituye una ventaja práctica frente a los modelos generativos clásicos.

## 4.4. Generación de imágenes a color

El último experimento llevó el modelo a su escenario más exigente, el conjunto de datos a color CIFAR-10 [26], que constituye la verdadera prueba de la generalización al dominio continuo: sus píxeles no son casi binarios, sino valores reales sobre un dominio tridimensional (RGB) con texturas y gradientes mucho más ricos. El resultado fue un fallo de carácter, anticipado por la propia teoría. El *embedding* de color se construye como el producto tensorial de la base de Legendre consigo misma sobre los tres canales, de modo que el sesgo hacia los extremos, ya presente en escala de grises, se eleva al cubo: el ratio de densidad *a priori* entre

un píxel saturado y uno intermedio pasa de $1:4$ a $1:4^3 = 64$. Las métricas lo reflejan con claridad: la NLL es mucho más ruidosa y arranca en valores positivos (Figura 19), y el margen frente al ruido binario es negativo durante todo el entrenamiento (Figura 20), lo que significa que el modelo asigna *mayor* densidad al ruido puro que a las imágenes reales. Las muestras generadas (Figura 21) confirman el diagnóstico: predominan regiones de color saturado en los vértices del cubo RGB, exactamente los valores que el sesgo premia. No obstante, incluso este fallo es informativo, ya que los artefactos cuadrados y rectangulares que aparecen permiten visualizar la estructura de correlaciones que impone la geometría de árbol de la red.

La conclusión de este experimento, lejos de invalidar el modelo, refuerza la tesis central del trabajo: la viabilidad del enfoque en el dominio continuo depende críticamente de la función de codificación. El buen comportamiento observado en escala de grises se debe en gran parte a que el sesgo de la base de Legendre resulta compatible con la naturaleza casi binaria del MNIST; en cuanto la información relevante reside en los tonos intermedios, como ocurre en el color, ese mismo sesgo se vuelve prohibitivo. Encontrar una codificación que concilie la normalización con la ausencia de sesgo es, por tanto, la línea de mejora más prometedora para extender el modelo a datos continuos más complejos.

# Capítulo 5 Conclusiones y Trabajo Futuro

## 5.1. Conclusiones

El objetivo general de este trabajo (diseñar, implementar y analizar un modelo generativo de imágenes basado en una Tree Tensor Network [6] y extenderlo del dominio discreto al continuo) se ha cumplido. Se ha formulado la regla de Born sobre una TTN cuyos píxeles toman valores reales en $[0, 1]$, se ha caracterizado la condición que debe satisfacer la función de codificación para que la densidad esté bien definida y normalizada (la ortonormalidad expresada mediante la matriz de Gram), y se ha demostrado que, en forma canónica, la función de partición se calcula de forma exacta como la norma del nodo central. Sobre esta base se ha implementado el entrenamiento por barrido y un algoritmo de muestreo secuencial exacto con control de temperatura. La generalización del modelo binario a uno continuo, que constituye la aportación principal de este trabajo, queda así establecida tanto teórica como prácticamente.

El estudio empírico ha confirmado la hipótesis de partida: el factor de diseño determinante es la función de codificación. La base de Legendre satisface la normalización y distingue los extremos de intensidad, pero introduce un sesgo *a priori* hacia los valores extremos que condiciona por completo el comportamiento del modelo [8]. En imágenes en escala de grises y casi binarias (MNIST), ese sesgo es compatible con los datos y el modelo rinde de forma satisfactoria: aprende la distribución y genera muestras reconocibles como dígitos (Figura 17), con un coste notablemente bajo (el modelo sobre el MNIST completo se entrena en 3 min 15 s y genera cada muestra en torno a 0,17 s). El control de la diversidad mediante la temperatura, evaluado de forma cuantitativa con la FID [25], sitúa el óptimo en $\tau = 0{,}3$ (FID $\approx 268$, frente a $\approx 296$ en el muestreo determinista $\tau = 0$ y más de 420 en el muestreo crudo $\tau = 1$). El experimento sobre la dimensión de enlace mostró, además, que los modelos con $B_{\text{máx}} = 32$, 16 y 8 resultan indistinguibles en MNIST, de modo que para un conjunto sencillo la capacidad expresiva de la red no es el factor limitante. Sin embargo, al llevar el modelo a un conjunto de datos a color (CIFAR-10), donde el sesgo de la codificación se eleva al cubo (un ratio de densidad *a priori* de hasta $1 : 4^3 = 64$ entre un píxel saturado y uno intermedio) y la información relevante reside en los tonos intermedios, el rendimiento se degrada de forma catastrófica: la NLL arranca en valores positivos (en torno a +530) y el margen de verosimilitud frente al ruido binario permanece negativo durante todo el entrenamiento, lo que significa que el modelo asigna mayor densidad al ruido puro que a las imágenes reales. La conclusión, lejos de invalidar el enfoque, refuerza su tesis central: la viabilidad de las máquinas de Born tensoriales en el dominio continuo depende críticamente de la elección del *embedding*.

Más allá del caso concreto estudiado, conviene situar las fortalezas de esta familia de arquitecturas. La principal es la combinación, infrecuente entre los modelos generativos, de garantías matemáticas exactas: la normalización es tratable, la verosimilitud es optimizable de forma directa y el muestreo es exacto, sin necesidad de cadenas de Markov ni de aproximaciones estocásticas. A ello se suman un coste de entrenamiento reducido, una notable

eficiencia en número de parámetros y un grado de interpretabilidad poco habitual, ya que la capacidad del modelo puede analizarse en términos de entrelazamiento [10], [11]. Reviste especial interés la ventaja estructural de la topología jerárquica de árbol frente a las cadenas (MPS): dos píxeles cualesquiera quedan conectados a través de un número de tensores que crece solo de forma logarítmica con su separación, de modo que las correlaciones pueden decaer de manera polinómica en lugar de exponencial. Esto dota a las TTN de una mayor capacidad para capturar correlaciones (es decir, entrelazamiento) a larga distancia que un MPS de la misma dimensión de enlace [16], una propiedad especialmente pertinente en imágenes, donde existen dependencias entre regiones alejadas.

Sus debilidades son, en buena medida, la contrapartida de ese mismo diseño. La capacidad expresiva está acotada por la dimensión de enlace y, en última instancia, por la ley de área que la red impone [5]: cuando las correlaciones del conjunto de datos superan lo que esa estructura puede representar, el modelo incurre en subajuste. Además, la geometría de árbol fija un patrón de correlaciones rígido (responsable de los artefactos rectangulares observados en las muestras a color) y obliga a ordenar los píxeles bidimensionales en una jerarquía que, pese a la elección alternante de ejes, no respeta plenamente la vecindad 2D de la imagen; arquitecturas como PEPS o MERA mitigarían esta limitación. A todo ello se añade la dependencia, central en este trabajo, respecto a la función de codificación, así como la pérdida de interpretabilidad de la verosimilitud absoluta en el dominio continuo, que dificulta la evaluación del modelo.

En conjunto, las máquinas de Born tensoriales se revelan como una alternativa rigurosa y eficiente para el modelado generativo, particularmente valiosa cuando se dispone de pocos datos o se requieren garantías exactas. Su aplicabilidad a datos continuos complejos está hoy limitada por dos frentes distintos (la codificación de la entrada y la capacidad de la arquitectura) que el trabajo futuro debería abordar de forma conjunta.

## 5.2. Trabajo futuro

La dirección más importante a partir de este trabajo es la búsqueda de alguna función de *embedding* sin sesgo. Ninguna de las codificaciones estudiadas preserva la ortonormalidad, la no degeneración y la norma de salida constante. Las dos primeras son requisitos obligatorios para el entrenamiento: sin una función de codificación ortonormal, el cálculo del factor de normalización no es correcto. Unos *embeddings* degenerados hacen que el modelo no sea capaz de distinguir entre píxeles con valores radicalmente diferentes (blanco-negro), lo que arruina el entrenamiento. La función de codificación actual (base de Legendre) socava el rendimiento en color debido a que la norma de su imagen no es constante. Resultaría natural explorar codificaciones de norma constante en dimensiones mayores, o bien *embeddings* aprendibles, parametrizados y optimizados conjuntamente con la red [21], que pudieran descubrir una representación de los píxeles libre de sesgo manteniendo la normalización. Una codificación adecuada permitiría retomar el caso del color, que en este trabajo ha quedado pendiente [8].

Si bien la topología de TTN descrita en este estudio es un primer acercamiento para representar imágenes debido a su carácter 2D, impone una estructura de correlaciones rígida que se manifiesta en los artefactos rectangulares observados en las muestras. Sustituir la

TTN por arquitecturas más ricas (como una *Projected Entangled Pair State* (PEPS), que respeta la estructura bidimensional de la imagen, o un *Multi-scale Entanglement Renormalization Ansatz* (MERA) [18], que captura correlaciones a múltiples escalas) podría mejorar la capacidad del modelo para representar datos más complejos, a costa de un mayor coste de contracción [4].

A lo largo del trabajo se ha visto que la NLL absoluta pierde su interpretabilidad en el caso continuo, lo que ha obligado a recurrir a métricas relativas como el margen de verosimilitud frente al ruido. Desarrollar métricas de evaluación más rigurosas y mejor calibradas para modelos de densidad continua (ya sean comparativas de verosimilitud sobre datos retenidos o análogos a las métricas perceptuales empleadas en otros modelos generativos [25]) sería una contribución metodológica valiosa, no solo para este modelo sino para el campo del modelado generativo con redes tensoriales en general.

# Capítulo 6 Bibliografía

[1] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza et al., «Generative adversarial nets», *Advances in neural information processing systems*, vol. 27, 2014.

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

[3] J. Ho, A. Jain y P. Abbeel, «Denoising diffusion probabilistic models», *Advances in neural information processing systems*, vol. 33, págs. 6840-6851, 2020.

[4] R. Orús, «A practical introduction to tensor networks: Matrix product states and projected entangled pair states», *Annals of physics*, vol. 349, págs. 117-158, 2014.

[5] J. Eisert, M. Cramer y M. B. Plenio, «Colloquium: Area laws for the entanglement entropy», *Reviews of modern physics*, vol. 82, n.º 1, págs. 277-306, 2010.

[6] S. Cheng, L. Wang, T. Xiang y P. Zhang, «Tree tensor networks for generative modeling», *Physical Review B*, vol. 99, n.º 15, pág. 155 131, 2019.

[7] Z.-Y. Han, J. Wang, H. Fan, L. Wang y P. Zhang, «Unsupervised generative modeling using matrix product states», *Physical Review X*, vol. 8, n.º 3, pág. 031 012, 2018.

[8] A. Meiburg, J. Chen, J. Miller, R. Tihon, G. Rabusseau y A. Perdomo-Ortiz, «Generative learning of continuous data by tensor networks», *SciPost Physics*, vol. 18, n.º 3, pág. 096, 2025.

[9] J.-G. Liu y L. Wang, «Differentiable learning of quantum circuit born machines», *Physical Review A*, vol. 98, n.º 6, pág. 062 324, 2018.

[10] I. Glasser, R. Sweke, N. Pancotti, J. Eisert e I. Cirac, «Expressive power of tensor-network factorizations for probabilistic modeling», *Advances in neural information processing systems*, vol. 32, 2019.

[11] Y. Liu, W.-J. Li, X. Zhang, M. Lewenstein, G. Su y S.-J. Ran, «Entanglement-based feature extraction by tensor network machine learning», *Frontiers in Applied Mathematics and Statistics*, vol. 7, pág. 716 044, 2021.

[12] A. Van Den Oord, N. Kalchbrenner y K. Kavukcuoglu, «Pixel recurrent neural networks», en *International conference on machine learning*, PMLR, 2016, págs. 1747-1756.

[13] D. H. Ackley, G. E. Hinton y T. J. Sejnowski, «A learning algorithm for Boltzmann machines», *Cognitive science*, vol. 9, n.º 1, págs. 147-169, 1985.

[14] J. C. Bridgeman y C. T. Chubb, «Hand-waving and interpretive dance: an introductory course on tensor networks», *Journal of physics A: Mathematical and theoretical*, vol. 50, n.º 22, pág. 223 001, 2017.

[15] D. Perez-Garcia, F. Verstraete, M. M. Wolf y J. I. Cirac, «Matrix product state representations», *arXiv preprint quant-ph/0608197*, 2006.

[16] Y.-Y. Shi, L.-M. Duan y G. Vidal, «Classical simulation of quantum many-body systems with a tree tensor network», *Physical Review A—Atomic, Molecular, and Optical Physics*, vol. 74, n.º 2, pág. 022 320, 2006.

[17] F. Verstraete y J. I. Cirac, «Renormalization algorithms for quantum-many body systems in two and higher dimensions», *arXiv preprint cond-mat/0407066*, 2004.

[18] G. Vidal, «Entanglement renormalization», *Physical review letters*, vol. 99, n.º 22, pág. 220 405, 2007.

[19] S. R. White, «Density matrix formulation for quantum renormalization groups», *Physical review letters*, vol. 69, n.º 19, pág. 2863, 1992.

[20] U. Schollwöck, «The density-matrix renormalization group in the age of matrix product states», *Annals of physics*, vol. 326, n.º 1, págs. 96-192, 2011.

[21] E. Stoudenmire y D. Schwab, «Supervised learning with tensor networks», *Advances in neural information processing systems*, vol. 29, 2016.

[22] C. M. Bishop y N. M. Nasrabadi, *Pattern recognition and machine learning*. Springer, 2006, vol. 4.

[23] L. Devroye, «Sample-based non-uniform random variate generation», en *Proceedings of the 18th conference on Winter simulation*, 1986, págs. 260-265.

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

[25] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler y S. Hochreiter, «Gans trained by a two time-scale update rule converge to a local nash equilibrium», *Advances in neural information processing systems*, vol. 30, 2017.

[26] A. Krizhevsky, «Learning Multiple Layers of Features from Tiny Images», University of Toronto, inf. téc., 2009.