

UNIVERSIDAD DE CÁDIZ  
FACULTAD DE CIENCIAS  
GRADO EN MATEMÁTICAS

AUMENTO DE DATOS BASADO EN GANs  
PARA LA CLASIFICACIÓN DE PLANOS  
CEREBRALES EN NEUROIMAGEN NEONATAL

Trabajo de fin de grado presentado por

Isaac Sánchez Sánchez

Tutor: Lionel Cervera Gontard

Firma del alumno

*Isaac Sánchez*

Firma de los tutores

Puerto Real, Cádiz, Septiembre de 2023



## **Abstract**

In this study, we explore the application of Artificial Intelligence (AI) to a medical problem. Specifically, we synthesize an artificial dataset using Generative Adversarial Networks (GANs) to train a classifier for ultrasound neuroimages of preterm neonates. Additionally, we analyze key concepts of deep learning, offering a mathematical perspective on the architecture of artificial neural networks and the backpropagation algorithm, and explain the fundamental principles of GANs. We conclude by presenting the results, discussing the limitations, and suggesting potential improvements of the methods explored in this work.



*A mis padres por darme la oportunidad de perderme y hallar mi propio  
camino.*



## **Resumen**

En este estudio, exploramos la aplicación de técnicas de Inteligencia Artificial (IA) a un problema médico. Específicamente, sintetizamos un conjunto de datos artificial utilizando Redes Generativas Antagónicas (GANs) para entrenar un clasificador de neuroimágenes de ultrasonidos de neonatos prematuros. Adicionalmente, analizamos conceptos clave del aprendizaje profundo, ofreciendo una perspectiva matemática sobre la arquitectura de las redes neuronales artificiales y el algoritmo de retropropagación, y explicamos los principios fundamentales de las GANs. Concluimos presentando los resultados, discutiendo las limitaciones y sugiriendo mejoras potenciales de los métodos explorados en este trabajo.



## **Agradecimientos**

En primer lugar agradecer a mi tutor, Lionel Cervera Gontard, que me presentó la idea y contagió su entusiasmo, no solo me ha apoyado, ha abierto un nuevo horizonte de posibilidades que me apasionan.

Gracias de corazón a mis amigos, gracias por los momentos de risa que alivian la presión, por las charlas donde compartíamos miedos y dudas, por los debates en la biblioteca cuando no entendíamos algún concepto. Gracias Ayla, por darme la paz cuando más lo necesitaba.

Gracias a todos aquellos profesores que comparten su pasión y tiempo con nosotros.

Agradezco también la colaboración de los neonatólogos Isabel Benavente y Simón Lubián del Hospital Universitario Puerta del Mar de Cádiz por compartir con nosotros los datos de base empleados en este proyecto.

Por último gracias a mi familia, por su apoyo incondicional, por su paciencia, por darme la oportunidad de estudiar y apreciar las matemáticas.

Isaac Sánchez Sánchez

diciembre 2023



# Índice general

<b>1</b>	<b>Introducción</b>	<b>1</b>
1.1	Problema . . . . .	3
1.2	Objetivo . . . . .	5
<b>2</b>	<b>Metodología y diseño del experimento</b>	<b>9</b>
2.1	Fases del experimento . . . . .	9
2.1.1	Fase 1: Preparación de datos para aprendizaje profundo . . . . .	10
2.1.1.1	Datos disponibles . . . . .	10
2.1.1.2	Preprocesado de datos . . . . .	11
2.1.2	Fase 2: Entrenamiento de modelos de aprendizaje profundo . . . . .	13
2.1.2.1	Modelo Generativo de imágenes sintéticas de ultrasonido	14
2.1.2.2	Modelo Clasificador de planos cerebrales . . . . .	14
2.1.3	Fase 3: Evaluación y comparación de modelos . . . . .	15
2.2	Recursos de software y hardware . . . . .	16
<b>3</b>	<b>Redes neuronales. Un enfoque matemático</b>	<b>19</b>
3.1	Redes neuronales discriminativas . . . . .	20
3.1.1	Modelo matemático . . . . .	20
3.1.1.1	Elementos de una red neuronal . . . . .	21
3.1.2	Aprendizaje máquina . . . . .	22
3.1.2.1	Propagación hacia adelante . . . . .	25
3.1.2.2	Retropropagación (o <i>Backpropagation</i> ) . . . . .	26
3.1.2.3	Optimización por descenso de gradiente . . . . .	31
3.1.3	Métricas y evaluación de Redes Neuronales . . . . .	32
3.1.3.1	Exactitud ( <i>Accuracy</i> ) . . . . .	33

## ÍNDICE GENERAL

---

3.1.3.2	Precisión ( <i>Precision</i> ) . . . . .	33
3.1.3.3	Sensibilidad ( <i>Recall</i> ) . . . . .	33
3.1.3.4	F1-Score . . . . .	34
3.2	Redes Generadoras Adversarias - GANs . . . . .	35
3.2.1	Fundamentos teóricos . . . . .	36
3.2.1.1	Función objetivo . . . . .	37
3.2.1.2	Discriminador óptimo . . . . .	38
3.2.1.3	Mínimo global . . . . .	40
3.2.1.4	Algoritmo de convergencia . . . . .	43
3.2.2	Métricas y evaluación de GANs . . . . .	46
3.2.2.1	Inception score . . . . .	47
3.2.2.2	FID . . . . .	49
3.2.2.3	KID . . . . .	51
<b>4</b>	<b>Resultados y discusión</b>	<b>53</b>
4.1	Creación del conjunto de imágenes de entrenamiento . . . . .	54
4.1.1	Aumento de datos mediante extracción de planos oblícuos . . . . .	55
4.2	Entrenamiento del modelo GAN . . . . .	58
4.2.1	Parámetros de entrenamiento de la GAN . . . . .	58
4.2.2	Evaluación del modelo . . . . .	59
4.3	Entrenamiento de clasificadores . . . . .	61
4.4	Descripción del código software desarrollado en el proyecto . . . . .	63
4.5	Discusión de resultados . . . . .	64
<b>5</b>	<b>Conclusiones y Trabajo futuro</b>	<b>67</b>
<b>Bibliografía</b>		<b>69</b>
<b>A Imágenes Generadas</b>		<b>73</b>

CAPÍTULO

# 1

## Introducción

En los últimos años el aprendizaje automático (o *machine learning*, en inglés) está redefiniendo la manera en que interactuamos con los datos y la tecnología. Esta rama de la **inteligencia artificial** (IA) se centra en el **desarrollo de algoritmos y modelos que permiten a las máquinas aprender y realizar predicciones o decisiones basadas en datos, sin ser explícitamente programadas para cada tarea específica**. Desde su concepción, el aprendizaje automático ha encontrado aplicaciones en una amplia gama de dominios, desde el reconocimiento de patrones y la visión por computadora hasta la toma de decisiones en negocios y la medicina personalizada [14].

El desarrollo del aprendizaje automático ha experimentado tres olas principales de evolución [4]. La primera ola (1950-1970) se caracterizó por un enfoque basado en el **conocimiento y la representación simbólica**. Los algoritmos, que utilizaban bases de conocimientos y reglas heurísticas para tomar decisiones, fueron una característica destacada de esta época. Se aplicaron en una variedad de campos, desde la medicina hasta la industria. Sin embargo, se encontraron limitaciones en la capacidad de estos sistemas para lidiar con la incertidumbre y el razonamiento fuera de su base de conocimientos. Tras una etapa en la que se redujo el interés, hubo un resurgimiento en los años 80, con un renovado interés por la aproximación conocida como **conexionismo** que marcó el advenimiento de algoritmos que aprenden de los datos. Estos incluyen modelos lineales, árboles de decisión, y máquinas de soporte vectorial, entre otros. En esta etapa se popularizó el uso de **redes neuronales más profundas** gracias al desarrollo del algoritmo de retropropagación. También

## 1. INTRODUCCIÓN

---

se hicieron avances importantes en la modelación de secuencias, con la introducción de la red LSTM (Long Short-Term Memory). La segunda ola permitió a los sistemas aprender patrones a partir de grandes conjuntos de datos, mejorando significativamente la capacidad de generalización y adaptación a nuevos datos. Sin embargo, estos métodos aún luchaban con datos de alta dimensionalidad y estructuras complejas. A mediados de los 90 hubo un nuevo declive en la investigación en la IA que duró hasta 2006 gracias a la mejora exponencial en la capacidad/coste de la computación y al acceso más eficiente a datos. Más concretamente, en cuanto a hardware la eficiencia y el gran número de servidores asequibles económicamente de las unidades de procesamiento gráfico (GPU) ha sido crucial; Y a nivel de datos se abrió el acceso a una cantidad masiva de estos (*Big Data*) etiquetados de forma gratuita y pública: *ImageNet* (con más de 14 millones de imágenes etiquetadas), *MNIST*, *CIFAR*, *COCO*, *Pascal VOC* . . . Esto permitió explotar los modelos conexionistas basados en redes neuronales profundas, con muchos miles de millones de parámetros, y que han demostrado ser extraordinariamente eficaces en el manejo de datos de alta dimensión y complejidad. El diseño y aprendizaje con estas redes neuronales profundas se conoce como aprendizaje profundo o *deep learning*, y ha llevado a avances muy significativos en campos como la visión por computadora, el reconocimiento de voz, y el procesamiento del lenguaje natural. Entre las herramientas de aprendizaje profundo en el estado del arte cabe mencionar las arquitecturas *transformer* que son la base de los grandes modelos de generación de lenguaje del tipo *ChatGPT* (GPT significa "*Generative Pre-trained Transformer*") así como para la generación de imágenes<sup>1</sup>. Además, en la etapa actual de desarrollo de la IA se busca trascender el modelo conexionista para desarrollar sistemas que no solo aprendan de los datos, sino que también comprendan y razonen, acercándose más a la inteligencia artificial general, o AGI (*Artificial General Intelligence* en inglés).

La aplicación de estos métodos de aprendizaje automático ha transformado numerosos campos de conocimiento, y en particular se espera un enorme impacto en el ámbito de la salud. Podemos mencionar aplicaciones de diagnóstico de retinopatía diabética a partir de imágenes fundoscópicas [10],[8], diagnóstico dermatológico de cáncer de piel [5], la

---

<sup>1</sup>Originalmente diseñados para tareas de procesamiento de lenguaje natural (NLP), los *transformer* son una base crucial para generar imágenes. Son conocidos por su capacidad para manejar secuencias de datos (como texto) de manera eficiente, capturando dependencias a largo plazo. DALL-E, desarrollado por OpenAI, es un ejemplo de cómo esta arquitectura se puede adaptar para generar imágenes a partir de descripciones textuales.

segmentación y clasificación automática de imágenes radiológicas [21] o de microscopía patológica [1].

La implementación de sistemas de aprendizaje profundo en medicina clínica es más lento que en otros ámbitos. La principal dificultad en el caso de métodos que usan imágenes médicas es la escasez de bases de datos de calidad (con muchos datos, etiquetados, formateados...). Conseguir crear buenos conjuntos de datos de entrenamiento requiere capturar imágenes en entornos muchas veces sensibles, heterogéneos a nivel humano y de gestión, siguiendo estrictos protocolos éticos, con consentimientos expresos de pacientes o familiares, añadido a la estricta legislación de privacidad. Para superar estos limitantes la IA generativa (por ejemplo, las redes generativas antagónicas (GANs) de Goodfellow et al. [7]) son un potente recurso que permite crear conjuntos sintéticos de imágenes indistinguibles de imágenes reales. Así, hay una intensa investigación en la síntesis de imágenes médicas en diferentes modalidades (MRI, CT, rayos X, etc.)

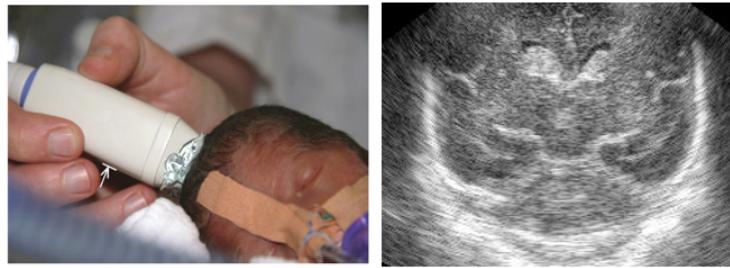
### 1.1 Problema

Se estima que, en 2020, 13.4 millones de niños nacieron pretérmino o prematuros [23]. Las diferencias en el pronóstico de estos pacientes entre países con ingresos altos e ingresos bajos es sustancial. En la mayoría de países de África y el sudeste asiático la prematuridad es la principal causa de defunción en los niños menores de cinco años [24]. A muchos supervivientes les espera una vida de discapacidad, lo que incluye dificultades de aprendizaje y problemas visuales y auditivos, lo cual se traduce en costes adicionales del estado para ayudarlos. En los países con mayores ingresos, se ha reducido el número de fallecidos y las complicaciones en el crecimiento de estos niños gracias a la prevención, detección e intervención. **Con el apoyo de técnicas de aprendizaje máquina, se espera que esta detección y prevención puedan ser mejoradas y abaratadas, reduciendo así el tiempo que deben dedicar los expertos a cada paciente y posibilitando intervenciones tempranas y eficaces.**

Unos de los problemas que enfrentan los neonatos prematuros es la hemorragia germinal matricial-intraventricular (GM-IVH). Esta complicación se presenta en aproximadamente el 20-25 % de los infantes que nacen con muy bajo peso (VLBWI, por sus siglas en inglés, *very low birth weight infants*) y, de estos casos, entre un tercio y la mitad desarrollan dilatación ventricular post-hemorrágica (PHVD en inglés) que hace que los ventrículos

## 1. INTRODUCCIÓN

---



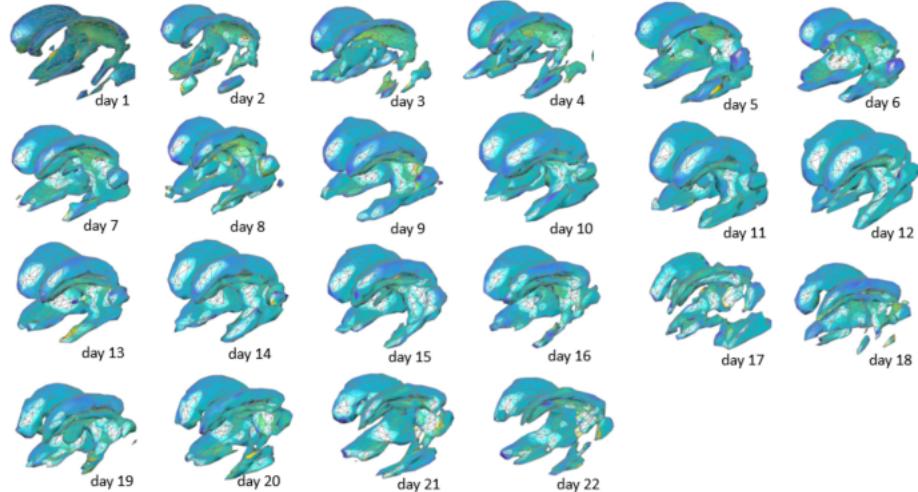
**Figura 1.1:** A la izquierda, imagen de un neonatólogo sujetando el cabezal de un ecógrafo de ultrasonido en la fontanela craneal de un bebé prematuro para monitorizar el correcto desarrollo cerebral. A la derecha, ejemplo de imagen coronal de ultrasonidos del cerebro de un neonato como las que estudiaremos en este proyecto. Las imágenes suelen ser ruidosas, y las intensidades de grises de los píxeles dependen de la densidad local.

laterales se llenen de líquido cefalorraquídeo (CSF) y crezcan aumentando la presión interna craneal y afectando a la maduración correcta del cerebro. Los VLBWI con PHVD tienen un alto riesgo de sufrir problemas neurológicos que afectarán gravemente su calidad de vida, como parálisis cerebral, epilepsia, deterioro cognitivo y problemas de comportamiento [3]. Aproximadamente el 20 % de los neonatos con PHVD necesitarán drenaje con un *shunt*<sup>1</sup> ventrículo-peritoneal permanente [6]. Actualmente, el drenaje de líquido CSF es el tratamiento principal para PHVD, pero el momento óptimo para este drenaje en los infantes con PHVD sigue siendo motivo de debate.

Los médicos neonatólogos y personal sanitario trabajan por el crecimiento saludable de los bebés prematuros (que nacen entre los 7 y 9 meses de embarazo) e investigan métodos que mejoren su salud presente y futura. Entre sus tareas cabe destacar por su importancia vital la detección temprana y precisa de posibles anomalías del desarrollo cerebral de los neonatos.

La ecografía (o imagen de ultrasonidos) craneal es una herramienta de diagnóstico por neuroimagen valiosa debido a su naturaleza no invasiva, su facilidad de uso y la ausencia de radiación ionizante. La figura 1.1 muestra un ecógrafo y una imagen típica del cerebro de un neonato. En la práctica clínica los neonatólogos hacen mediciones mediante ecografía

<sup>1</sup>Un *shunt* ventrículo-peritoneal (VP) es un dispositivo médico que se utiliza para tratar la hidrocefalia, una condición en la cual se acumula líquido cefalorraquídeo (CSF) en exceso dentro del cerebro, causando una presión anormalmente alta. El *shunt* VP consta de un sistema de tubos y una válvula que se implanta quirúrgicamente para desviar el exceso de líquido desde los ventrículos cerebrales hacia la cavidad peritoneal, que es el espacio dentro del abdomen que alberga varios órganos. La válvula controla el flujo del líquido para mantener la presión dentro de un rango normal.



**Figura 1.2:** Evolución temporal del sistema ventricular de un paciente durante 22 días. Imagen obtenida del artículo [6]

de la evolución del tamaño de los ventrículos laterales de los bebés para decidir el momento óptimo para la intervención con un *shunt*. La figura 1.2 muestra una imagen del seguimiento durante 22 días del crecimiento de los dos ventrículos laterales de un bebé. Se observa la inflamación de los ventrículos y como disminuyen de tamaño después de una intervención con *shunt* para drenar el líquido CSF.

Actualmente, los neonatólogos experimentados son los encargados de realizar un laborioso trabajo manual de analizar y medir las ecografías una a una. Esto, además de quitar tiempo a otras tareas más importantes de los médicos supone un mayor riego de errores intra-observador (2 médicos distintos pueden dar resultados distintos). Parece obvio que es de gran interés automatizar estas tareas, pero esto era algo que hasta hora no era posible, ya que las imágenes de ultrasonidos son en general de baja calidad (poca resolución y muy ruidosas) lo que dificultaba enormemente automatizar su análisis. Ahora, gracias al aprendizaje profundo, esto ya no es una quimera. Un ejemplo es la segmentación de los ventrículos a partir de ecografías tridimensionales de prematuros como se explica en el trabajo de Gontard et al. [6].

## 1.2 Objetivo

**La primera motivación de este TFG es aportar soluciones para mejorar el análisis automático de imágenes médicas en el ámbito de la neuroimagen neonatal. ¿Cómo**

## 1. INTRODUCCIÓN

---

podemos afrontar este problema? Creemos que mediante técnicas de IA.

Hoy en día, podemos dividir los modelos de aprendizaje máquina en dos grandes grupos:

- Los modelos de IA **Discriminativa**, que son aquellos entrenados para adjudicar una categoría a los datos que les proporcionamos. Por ejemplo, si se entrena un modelo clasificador al que le pasamos datos clínicos de los pacientes, y que los clasifique según el riesgo de sufrir un infarto.
- Los modelos de IA **Generativa**. A estos modelos no les proporcionamos datos; les pedimos que nos generen datos a partir de una etiqueta. Por ejemplo, a partir de una imagen aleatoria con ruido, pueden generar una distribución de píxeles que se adecúe a una imagen indistinguible de una real que requerimos. Forman parte de este grupo la herramientas *DALL-E* para generar imágenes o *ChatGPT* para generar texto.

La limitación que más frena el desarrollo de herramientas de IA en el ámbito hospitalario, es decir, trasladar un trabajo de investigación a una herramienta clínica, es la dificultad en obtener bases de datos de calidad. **Una herramienta que analice imágenes (basada en entrenar modelos de aprendizaje profundo) para uso clínico debe ser muy robusta, para lo cual debe entrenarse y validarse con miles o mejor en decenas de miles de imágenes que abarquen el mayor número posible de casos:** imágenes de distintas resoluciones, orientaciones y nivel de ruido, adquiridas con distintos ecógrafos, hospitales y médicos, y que cubran el más amplio rango posible de estados clínicos. No se trata sólo de una cuestión práctica, sino de limitantes éticos que hacen que por privacidad de datos sea en general muy complicado obtener un conjunto de datos suficientemente amplio y representativo. Esto es un problema aún mayor cuando se trabaja con pacientes que no pueden dar su consentimiento (caso de bebés o niños).

**En este trabajo, hemos investigado el uso tecnologías de IA (aprendizaje profundo) con objeto de investigar métodos que pudieran facilitar el trabajo de los neonatólogos en el diagnóstico por neuroimagen de ultrasonidos del correcto desarrollo cerebral de bebés prematuros [6].** Más concretamente, en este trabajo **hemos desarrollado un clasificador (IA Discriminativa) de imágenes cerebrales de ultrasonidos neonatales que sea robusto mediante métodos que reduzcan el problema de la escasez de datos reales.** Para ello exploramos técnicas de entrenamiento semi supervisado en combinación

## **1.2 Objetivo**

---

**con la síntesis de imágenes artificiales mediante redes generativas antagónicas (GANs o Generative Adversarial Networks en inglés), de suficiente calidad como para formar parte del conjunto de entrenamiento del clasificador (IA Discriminativa).** Para ello, contamos con imágenes médicas de ultrasonidos sin etiquetar proporcionadas por médicos colaboradores de la unidad de neonatología del Hospital Universitario Puerta del Mar de Cádiz.

Además de explorar la aplicación práctica de estas tecnologías en el campo de la neonatología, **este TFG también tiene como objetivo entender la base matemática que impulsa estos algoritmos de aprendizaje máquina y las redes generativas antagónicas.**

Esta memoria está estructurada en lo siguientes capítulos:

En el capítulo 2, explicaremos los experimentos y la metodología aplicada, el contexto médico en el que se ubica, los problemas que se presentan y como usamos el aprendizaje máquina para solventarlos.

En el capítulo 3, abordaremos las bases de una red neuronal desde un punto de vista matemático, que son el pilar para todos los nuevos métodos de aprendizaje profundo. Pondremos un ejemplo de una red neuronal sencilla y veremos como se aplica el algoritmo de retropropagación. También analizaremos la teoría matemática que impulsa las redes generativas antagónicas, así como los principales métodos de evaluación de la calidad de los datos generados con estas redes.

En el capítulo 4, describimos y analizamos los resultados de los experimentos llevados a cabo en este TFG.

Por último, en el capítulo final exponemos las conclusiones, valoramos los resultados, sus implicaciones y posibles ampliaciones de la investigación que se pueden llevar a cabo.



CAPÍTULO

# 2

## Metodología y diseño del experimento

En este capítulo describimos los métodos y técnicas empleados para alcanzar los objetivos planteados en este TFG.

El experimento que desarrollamos en este trabajo esta inspirado en el trabajo [22] pero con marcadas diferencias metodológicas en cuanto al tipo de imágenes, al modo de generar el conjunto de datos, al tamaño del conjunto de datos, así como, al desarrollo de la estrategia de entrenamiento semi-supervisado del clasificador que concluye nuestro proyecto.

Adicionalmente, en este capítulo dedicamos un apartado a explicar los medios técnicos que hemos utilizado para llevar a cabo el proyecto, tanto a nivel de código de programación como de servidores y hardware.

### 2.1 Fases del experimento

En nuestro proyecto podemos distinguir tres fases consecutivas del experimento que son:

1. Preparación de datos (sección 2.1.1),
2. Entrenamiento de los modelos (sección 2.1.2),
3. Evaluación de los resultados (sección 2.1.3).

Estas tres fases son comunes a cualquier proyecto de aprendizaje máquina [17], y que pasaremos a explicar en el resto del capítulo.

## **2. METODOLOGÍA Y DISEÑO DEL EXPERIMENTO**

---

### **2.1.1 Fase 1: Preparación de datos para aprendizaje profundo**

En esta fase nuestro objetivo es obtener un conjunto de datos válido en cantidad, diversidad y calidad para poder llevar a cabo la siguiente fase consistente en el entrenamiento de los modelos de redes neuronales. Para ello vamos a construir un conjunto de imágenes reales de ultrasonidos con planos cerebrales de 3 tipos que usaremos por un lado para entrenar una GAN , y también para poder entrenar un clasificador.

#### **2.1.1.1 Datos disponibles**

Los datos originales obtenidos del Hospital Puerta del Mar son imágenes de ecografía tridimensional (3D), es decir, volúmenes de todo el cerebro de bebés prematuros adquiridos en distintos días mediante un ecógrafo manual 3D. A este respecto, cabe mencionar que los neonatólogos del Hospital Puerta del Mar son además investigadores de prestigio internacional en el ámbito del uso y desarrollo de la ecografía 3D neonatal.

Las ecografías 3D que nos han suministrado son archivos en formato NRRD (\*.nrrd<sup>1</sup>). Este formato de imagen es particularmente popular en la comunidad de investigación debido a su simplicidad y capacidad para manejar datos complejos de imágenes, lo que lo hace valioso para el análisis y la visualización avanzada en diversas áreas de la ciencia y la medicina. Para este trabajo hemos usado 4 volúmenes .nrrd provenientes de 3 pacientes distintos (los dos primeros del mismo paciente) a los que los médicos tomaron ecografías diarias en series temporales de hasta un mes. Para asegurar que los conjuntos de entrenamiento tienen suficiente variedad en las imágenes, los volúmenes se corresponden con distintos días del seguimiento por ecografía. Las referencias de los ficheros empleados son:

<sup>1</sup>El formato NRRD (*Nearly Raw Raster Data*) es un formato de archivo diseñado para almacenar y visualizar datos multidimensionales de imágenes médicas y científicas. Fue desarrollado como parte del proyecto *Teem*, que tiene como objetivo proporcionar un conjunto de herramientas para procesar y visualizar datos de imágenes médicas. Algunas características clave del formato NRRD son:

1. Multidimensionalidad: NRRD es capaz de manejar imágenes 3D y 4D. Esto es especialmente útil en aplicaciones médicas donde se requieren imágenes de tomografía computerizada, resonancia magnética y otras técnicas de imagen que producen datos en múltiples dimensiones.
2. Flexibilidad: Puede almacenar diferentes tipos de datos y metadatos, incluyendo información sobre el tamaño de los píxeles, la orientación de la imagen, la escala de grises, entre otros, o información de pacientes.
3. Formato de Archivo: NRRD puede utilizarse tanto en formatos de archivo de texto (legibles por humanos) como binarios, lo que permite una fácil interpretación y manipulación de los datos.
4. Interoperabilidad: Aunque no es tan ampliamente adoptado como otros formatos como DICOM en el ámbito médico, NRRD es compatible con varias herramientas de procesamiento de imágenes médicas y científicas, facilitando el intercambio y análisis de datos. Está diseñado para ser simple y fácil de usar, con una estructura de archivo clara que facilita la comprensión y el procesamiento de los datos.

## **2.1 Fases del experimento**

---

1\_2013\_11\_08.nrrd  
1\_2013\_11\_15.nrrd  
4\_2013\_06\_30.nrrd  
4075542\_2015\_10\_04.nrrd

### **2.1.1.2 Preprocesado de datos**

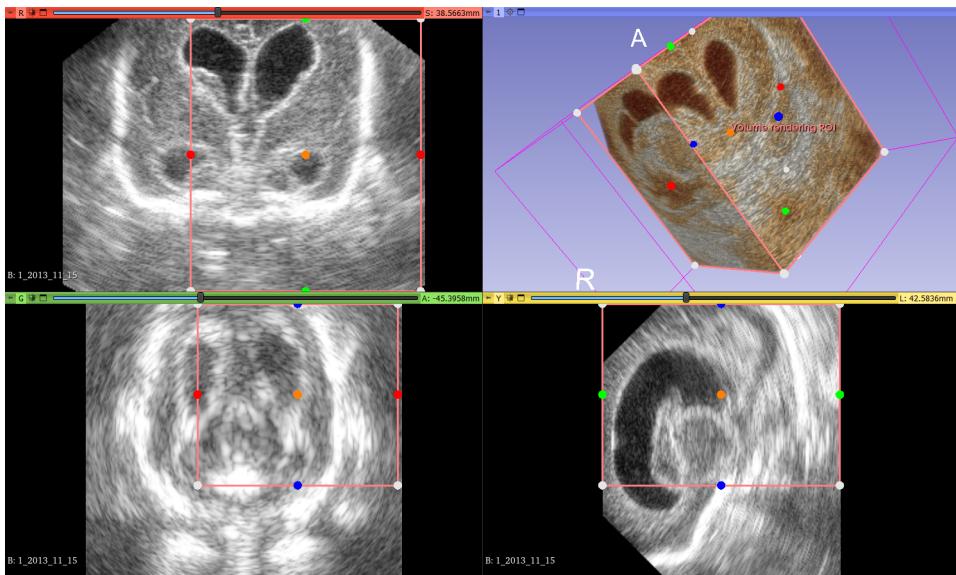
Aunque los datos suministrados son 3D, el diagnóstico por ecografía craneal suele hacerse típicamente a partir de imágenes 2D. Para llevar a cabo una evaluación completa del paciente los neonatólogos suelen examinar tres planos estándar del cerebro:

- *Plano Coronal*: Proporciona vistas del cerebro que incluyen las estructuras de la línea media, los ventrículos laterales, las cisuras interhemisféricas y las regiones frontales y occitales. Las imágenes coronales son esenciales para evaluar la simetría cerebral y la anatomía ventricular.
- *Plano Sagital*: Este plano permite visualizar la línea media del cerebro, incluyendo el cuerpo calloso, los ventrículos laterales, el tercer ventrículo, el cerebelo, y el tronco encefálico. Es particularmente útil para evaluar la anatomía de la línea media y la presencia de posibles malformaciones.
- *Plano Axial*: Este plano es útil para examinar las relaciones anatómicas transversales del cerebro, incluyendo los ganglios basales, los ventrículos laterales, las estructuras del cerebelo y la morfología hemisférica.

Estos tres planos proporcionan una visión integral del cerebro, permitiendo la detección y evaluación de anomalías estructurales, hemorragias, lesiones, y otros problemas neurológicos en neonatos y lactantes. En la figura 2.1 se muestra un volumen cerebral de los suministrados por los médicos para este trabajo. Concretamente, en la figura se visualizan 3 vistas de planos cerebrales que son ortogonales, y que se denominan como vista coronal (arriba izquierda), vista sagital (abajo a la derecha), y vista axial (abajo izquierda).

## 2. METODOLOGÍA Y DISEÑO DEL EXPERIMENTO

---



**Figura 2.1:** Ejemplo de ecografía cerebral 3D de un prematuro se observan los ventrículos laterales en tres planos ortogonales estándar con vista sagital, axial y coronal. Imagen del autor

En este trabajo, nos hemos centrado en desarrollar un clasificador de neuroimagen eco-gráfica neonatal que sea capaz de distinguir de manera automática entre tres tipos de imágenes bidimensionales: sagitales, coronales, o ruido (sin información de interés). Los datos de los que disponemos son ecografías tridimensionales de modo que como primer paso, a partir de los ficheros que nos han suministrado, hemos tenido que diseñar un algoritmo para generar un conjunto de datos de entrenamiento de ecografías 2D, extrayendo planos de las ecografías 3D.

A continuación, las imágenes se han organizado en distintas carpetas para distinguir entre datos de entrenamiento y validación, y dentro de cada uno, hemos separado las imágenes según una de tres clases o etiquetas: coronal, sagital y ruido.

Posteriormente, hemos redimensionado todas las imágenes al tamaño de entrada de los modelos de redes neuronales que hemos empleado. Estas redes están generalmente diseñadas para operar con imágenes de  $256 \times 256$  pixeles y en color RGB<sup>1</sup> este formato es estándar en el procesamiento de imágenes y permite que la red maneje información de color consistente y completa, y para evitar problemas de inconsistencia en el tamaño de los

<sup>1</sup>El RGB es un modelo de color basado en la mezcla aditiva de colores, donde R, G, y B se refieren a Rojo, Verde y Azul, respectivamente

## **2.1 Fases del experimento**

---

datos de entrada. Las imágenes de ultrasonido son en escala de grises que hemos convertido a RGB apilando 3 veces la imagen. Y hemos ajustado el ancho y alto de las imágenes al de entrada de la red, de modo que hemos creado para cada imagen una nueva imagen de dimensiones  $256 \times 256 \times 3$ .

Al final del proceso de preparación de datos hemos obtenido un conjunto de datos con 15536 imágenes en total, de las cuales 6318 son planos coronales, 7128 son del plano sagital y 2088 son clasificadas como ruido.

### **2.1.2 Fase 2: Entrenamiento de modelos de aprendizaje profundo**

Aquí debemos distinguir 2 apartados. Por un lado está el entrenamiento del modelo discriminativo clasificador por otro el modelo generativo tipo GAN.

Existen tres principales formas de entrenar los modelos de aprendizaje máquina:

#### **Entrenamiento supervisado**

En el aprendizaje supervisado, el conjunto de datos es una colección de ejemplos etiquetados  $\{(X_i, y_i)\}_{i=1}^N$ . Cada  $X_i$  es un ejemplo y cada  $y_i$  es la etiqueta correspondiente. El objetivo es entrenar un modelo que pueda tomar un vector de características  $x$  como entrada y producir una salida que permita deducir la etiqueta  $y$  correspondiente.

En nuestro caso cada ejemplo  $X_i$  es una imagen de  $256 \times 256 \times 3$  píxeles, es decir un vector de dimensión 196608, su etiqueta  $y_i$  pertenece a un conjunto de clases con 3 elementos {"planocoronal", "planosagital", "ruido"}

#### **Entrenamiento no supervisado**

En el aprendizaje no supervisado, el conjunto de datos consiste en una colección de ejemplos no etiquetados  $\{X_i\}_{i=1}^N$ . El objetivo aquí es entrenar un modelo que tome un ejemplo  $x$  como entrada y lo transforme en otro vector o en un valor que pueda usarse para resolver un problema práctico.

## 2. METODOLOGÍA Y DISEÑO DEL EXPERIMENTO

---

### Entrenamiento semi-supervisado

El aprendizaje semi-supervisado utiliza un conjunto de datos que contiene tanto ejemplos etiquetados como no etiquetados  $\{(X_i, y_i)\}_{i=1}^{N_L} \cup \{X_j\}_{j=1}^{N_U}$ . Normalmente el tamaño de datos no etiquetados  $N_U$  es mucho mayor que el de los etiquetados  $N_L$ . El objetivo es el mismo que en el aprendizaje supervisado. La idea es que la presencia de muchos ejemplos no etiquetados pueda ayudar al algoritmo a encontrar un mejor modelo. Este enfoque es muy útil cuando resulta costoso y tedioso a nivel de tiempo realizar un etiquetado a "mano" de todos los datos.

#### 2.1.2.1 Modelo Generativo de imágenes sintéticas de ultrasonido

Para el modelo generativo hemos usado la arquitectura GAN *StyleGAN2-ADA* [16]. El entrenamiento de las redes GANs se suele explicar con el siguiente ejemplo: pongamos que tienes 2 personas, una actuará como critica de arte y la otra intentará engañar al critico mediante falsificaciones de cuadros. La idea es que la interacción entre estas dos personas acabará mejorando la actuación de ambas, el crítico será cada vez mas bueno detectando falsificaciones y el falsificador tendrá que esforzarse más para engañar al crítico. Una red GAN es esencialmente un modelo clasificador y un modelo generativo interactuando para superarse entre ellos.

Existen distintos tipos de GAN. En este proyecto nos decidimos por usar una GAN no condicionada, es decir, una red que se entrena suministrando imágenes de planos sagitales y coronales juntos, sin etiquetar. De este modo, una vez entrenada, debería ser capaz de generar imágenes de ambos planos. Se podría haber optado por realizar un entrenamiento de dos GAN por separado, cada una especializada en un plano, o haber entrenado un modelo GAN condicionado que es entrenado con las imágenes etiquetadas. Estas opciones no se han llevado a cabo por falta de tiempo y no aumentar la extensión del trabajo. En cualquier caso, planteamos estas alternativas para una extensión futura de este trabajo.

#### 2.1.2.2 Modelo Clasificador de planos cerebrales

Optamos por utilizar una estrategia de *fine tuning* de arquitecturas de redes neuronales pre-entrenadas. Concretamente, hemos empleado una red denominada *convnext\_tiny*[19] que hemos reentrenado con los datos etiquetados a nuestra disposición. La eficacia de este proceso se ve influenciada por varios hiperparámetros que se pueden ajustar. Entre ellos se

## **2.1 Fases del experimento**

---

incluyen el ratio de aprendizaje, el número de *epochs* y las transformaciones aplicadas a las imágenes de entrada.

En una primera fase entrenamos el clasificador base de forma supervisada, después, en la siguiente fase, **hemos empleado un entrenamiento semi-supervisado a partir de los datos sintéticos** con el objetivo de investigar si es posible obtener al menos las mismas métricas pero con un menor número de imágenes reales.

El procedimiento para el entrenamiento semi supervisado es el siguiente:

1. Nos quedamos con un porcentaje  $X$  del conjunto original de datos reales
2. Generamos  $n$  imágenes sintéticas con el modelo que hemos entrenado
3. Ahora entrenamos un clasificador con el porcentaje  $X$  del conjunto original.
4. Con el modelo obtenido en el paso anterior clasificamos las  $n$  imágenes generadas sintéticamente y que están sin etiquetar. Aquí aplicamos un umbral de confianza (*threshold*, en inglés), donde solo las imágenes clasificadas con una seguridad del 0,9 se utilizarán en la próxima etapa.
5. La siguiente etapa consiste en añadir esas nuevas imágenes sintéticas a las que hemos puesto una pseudo-etiqueta al conjunto original, ampliando el número de imágenes de entrenamiento. Volvemos a entrenar el modelo con este conjunto ampliado, vemos que tal es su actuación e iteramos de nuevo sobre el conjunto de imágenes generadas restantes, así hasta que llegamos a un resultado satisfactorio o terminemos con las imágenes artificiales.

En nuestro trabajo hemos experimentado con el uso de distintos porcentajes de repartición del conjunto de datos original, dedicando al entrenamiento del clasificador un 70 %, 50 % o 30 % de imágenes reales, y complementar el 30 %, 50 % o 70 % respectivo con las imágenes sintéticas generadas con la GAN.

### **2.1.3 Fase 3: Evaluación y comparación de modelos**

Las imágenes generadas con la GAN las evaluamos según distintas métricas que cuantifican la calidad y variedad de la imágenes generadas. Esta métricas las explicamos en la sección 3.2.

## 2. METODOLOGÍA Y DISEÑO DEL EXPERIMENTO

---

Por otra parte, evaluamos el clasificador original entrenado con un 100 % de datos reales, y lo usamos como referencia para evaluar el rendimiento de los distintos clasificadores entrenados con aprendizaje semi supervisado (mezclando imágenes reales y sintéticas).

La validación la hacemos con un conjunto de datos que no haya estado presente en ninguno de los entrenamientos. Para ello hemos llevado a cabo las misma operaciones de extracción y preparación de imágenes explicadas en la sección 2.1.1.2 a 3 archivos \*.nrrd con las siguientes referencias

```
1_2013_11_05.nrrd  
1_2013_11_10.nrrd  
1_2013_11_20.nrrd
```

Para finalizar, en la figura 2.2 se muestra esquemáticamente la relación entre las distintas tareas llevadas a cabo durante el proyecto.

### 2.2 Recursos de software y hardware

Para este proyecto hemos trabajado sobre todo con las librerías de *fast.ai* que proporciona unos módulos y funciones sencillas de implementar para ir directamente al entrenamiento de estos modelos sin tener que perder demasiado tiempo en detalles de programación.

El entorno de programación se ha basado en *Pytorch*<sup>1</sup> y los *Jupyter Notebooks* de *jupyter lab*<sup>2</sup>.

Para el entrenamiento del modelo generativo hemos hecho uso del repositorio de StyleGAN2-ada <https://github.com/NVlabs/stylegan2-ada-pytorch>.

El entrenamiento se ha realizado con GPUs disponibles gratuitamente en los servidores de *paperspace*<sup>3</sup>. Más concretamente, el modelo GAN fue entrenado con la GPU P-500, durante dos días, hasta alcanzar un número de 1200 kimg<sup>4</sup>.

---

<sup>1</sup>PyTorch es una biblioteca de código abierto de aprendizaje profundo (deep learning) desarrollada por Facebook's AI Research (FAIR). Se utiliza para desarrollar y entrenar modelos de redes neuronales artificiales, y se ha convertido en una de las bibliotecas más populares en el campo del aprendizaje profundo debido a su flexibilidad y facilidad de uso.

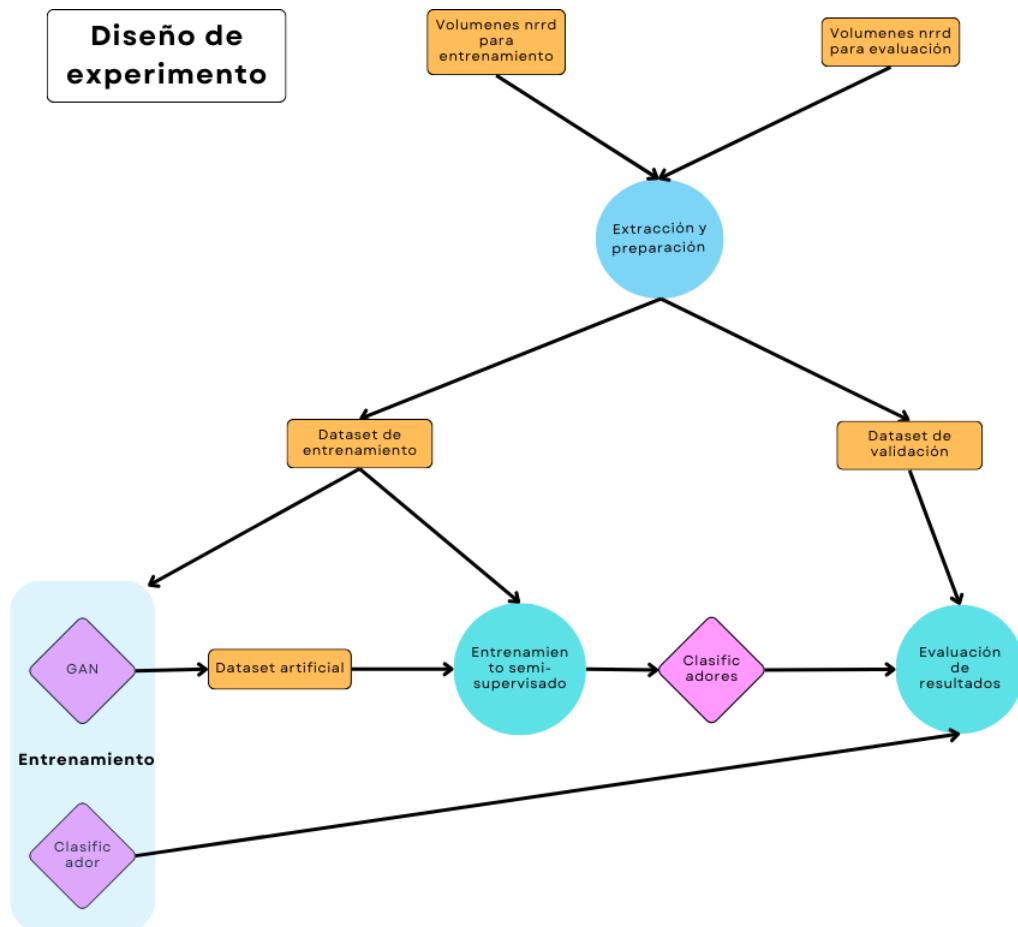
<sup>2</sup>Son una aplicación web de código abierto que permite crear y compartir documentos interactivos que contienen código, texto explicativo, visualizaciones y otros elementos. Están diseñados principalmente para facilitar la programación y la exploración de datos de una manera colaborativa y reproducible. El nombre "Jupyter" es un acrónimo de tres lenguajes de programación populares que son compatibles con la plataforma: Julia, Python y R.

<sup>3</sup><https://www.paperspace.com>

<sup>4</sup>1 kimg = 1000 imágenes (de modo que 1200 kimg son  $1,2 \cdot 10^6$  imágenes)

## 2.2 Recursos de software y hardware

---



**Figura 2.2:** Diagrama de las distintas fases del experimento realizado en este proyecto de investigación. Imagen del autor.



CAPÍTULO  
**3**

## **Redes neuronales. Un enfoque matemático**

Ahora que nos hemos hecho una idea general de las etapas del proyecto y lo que ocurre en cada una de ellas, pasaremos a definir y ver desde un punto de vista matemático el funcionamiento de las redes neuronales artificiales, que son la base del funcionamiento de los modelos de redes convolucionales profundas, y también de arquitecturas más recientes como los *transformers* y las redes LSTM entre otras. Ya que hay muchas técnicas que podríamos expandir y el documento se extendería demasiado no entraremos en esas nuevas técnicas. De cualquier forma, todas estas nuevas capas de complejidad y refinamiento que amplían las posibilidades de la IA se construyen sobre la estructura que explicaremos a lo largo de este capítulo, el cual dividiremos en dos secciones.

En esta sección introduciremos como se origino la arquitectura de las redes neuronales, las interpretaremos desde un punto de vista matemático y definiremos los elementos que la construyen. Después profundizaremos en el aprendizaje máquina, analizando los distintos pasos que lo forman. Por último explicaremos como podemos medir los resultados de nuestro modelo.

En la segunda sección explicaremos que son las redes generadoras adversarias, veremos como interactúan dos redes neuronales en un mismo sistema y demostraremos cuales son los fundamentos matemáticos que justifican su uso y utilidad. Por último analizaremos los puntos fuertes y débiles de las distintas métricas para evaluar nuestros resultados.

### 3. REDES NEURONALES. UN ENFOQUE MATEMÁTICO

---

#### 3.1 Redes neuronales discriminativas

La estructura de las redes neuronales artificiales fue inspirada por el comportamiento biológico de las neuronas. En el cerebro las neuronas reciben información en forma de pulsos eléctricos analógicos a través de células especializadas que están en nuestros órganos sensoriales. Esta información es procesada por redes de neuronas interconectadas en alguna parte de nuestro cerebro donde se disparan múltiples conexiones en forma de impulsos eléctricos, que terminan dando lugar a una respuesta o salida (por ejemplo en forma de un movimiento muscular).

Los padres de la IA se plantearon como simular matemáticamente el comportamiento neuronal. Y empezando por la red más simple modelaron una neurona, tal y como se muestra en la figura 3.1.

Comenzaremos estableciendo algunos términos. La información externa a una neurona (que podría ser por ejemplo las señales ópticas de distintos puntos de la retina del ojo humano) serán las variables de entradas. Estas señales pueden estar organizadas matemáticamente como vectores, matrices o estructuras más complejas. Estas entradas o *inputs* en inglés, formarán la denominada **capa de entrada** o *input layer*.

La reacción de nuestro cuerpo a unos estímulos vendría a ser el resultado o señal que genera la neurona artificial a su salida. En el caso de una red neuronal, formada por 2 o más neuronas, las salidas conformarían lo que se denomina la **capa de salida** o *output layer*. Si tenemos además una red con varias capas de neuronas que están conectadas y no son ni la capa de salida ni la capa de entrada, estamos hablando de las **capas ocultas** o *hidden layers*.

##### 3.1.1 Modelo matemático

Las estructuras de redes neuronales se representan como grafos dirigidos, donde los nodos son las "neuronas" y los arcos serán las conexiones entre neuronas. En la capa de entrada y la capa de salida estos nodos representan las variables de entrada y salida respectivamente.

Podemos entender una red neuronal discriminativa como una función  $F$  que toma imágenes del conjunto de entrenamiento  $\mathcal{X} = \{(X_i, \hat{y}_i)\}$  con  $i = 1 \dots N$ , donde  $N$  es el número de imágenes que forman nuestro conjunto, en este caso  $N = 15536$ , que es el número total de imágenes que obtuvimos. En nuestro caso cada imagen  $X_i$  está formada por  $256 \times 256 \times 3$  píxeles, es decir, 196608 variables de entrada  $x$ . Por otro lado la salida

de nuestro modelo será la probabilidad de que nuestra imagen pertenezca a una de las 3 categorías, por lo tanto,  $F(X_i) = (y_1, y_2, y_3), i = 1 \dots N$ .

En general la función  $F$  va del espacio  $\mathbb{R}^n$  al espacio  $\mathbb{R}^m$ , donde  $n$  es el número de entradas  $x_1, \dots, x_n$  y  $m$  es el número de salidas  $y_1, \dots, y_m$ .

$$F : \mathbb{R}^n \rightarrow \mathbb{R}^m$$

#### 3.1.1.1 Elementos de una red neuronal

La notación que suele emplearse para describir las redes neuronales es la siguiente:

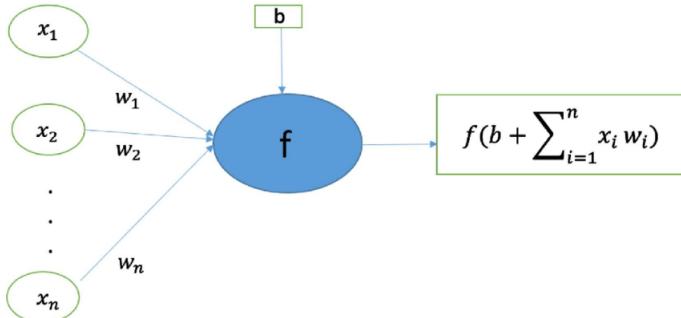
- $L$ : Número de capas en la red.
- $n^l$ : Número de neuronas en la capa  $l$ .
- $a_j^l$ : Activación de la  $j$ -ésima neurona en la capa  $l$ .
- $w_{j,i}^l$ : Peso de la conexión entre la  $i$ -ésima neurona en la capa  $l - 1$  y la  $j$ -ésima neurona en la capa  $l$ .
- $b_j^l$ : Sesgo de la  $j$ -ésima neurona en la capa  $l$ .
- $z_j^l$ : suma ponderada que forma la variable de entrada de la  $j$ -ésima neurona de la capa  $l$ ,  $z_j^l = \sum_k^{n^{l-1}} w_{j,k}^l a_k^{l-1} + b_j^l$ .

Antes de definir una neurona debemos entender que significan los arcos. Estas conexiones tienen asociado un parámetro que llamaremos **peso**, los cuales modifican la fuerza de estas conexiones, estos pesos los denotamos como  $w_{j,i}^l$ , el índice  $l$  simboliza el número de la capa en la que incide el arco,  $i$  indica la neurona de la que proviene y  $j$  la neurona en la que incide el arco. Un arco es el producto del peso asociado por la variable del nodo de origen. Además cada neurona  $j$  de la capa  $l$  tiene asociado un término  $b_j^l$  llamado **bias** o sesgo. La variable de entrada en una neurona es la suma ponderada de estas conexiones más el sesgo, lo que representaremos como  $z_j^l = \sum_k^{n^{l-1}} w_{j,k}^l a_k^{l-1} + b_j^l$ .

La neurona o nodo es una función que, siguiendo la similitud con la biología llamaremos función de activación, esta introduce una componente de no linealidad que permite modelar relaciones más complejas. Algunos ejemplos son la función *Step*, la *Sigmoide*, la *Tangente hiperbólica*, la función *ReLU* (y su variante *Leaky ReLU*). Esta última y sus variantes es de las más usadas hoy en día. Matemáticamente se definen como sigue:

### 3. REDES NEURONALES. UN ENFOQUE MATEMÁTICO

---



**Figura 3.1:** Estructura de una neurona artificial con entradas, salidas, un bias y una función de activación. [12]

$$\text{Step}(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases}$$

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

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

$$\text{ReLU}(x) = \max(0, x)$$

$$\text{Leaky ReLU}(x) = \begin{cases} x & \text{if } x \geq 0 \\ \alpha x & \text{if } x < 0 \end{cases}$$

La figura 3.2 muestra gráficamente algunas de estas funciones de activación.

A partir de ahora, por concretar, nuestra función de activación  $f$  será la función sigmoidal  $\sigma$ . Esta no tiene nada de especial, todos los resultados de este capítulo son aplicables a otras funciones de activación, de hecho distintas capas pueden tener distintas funciones de activación.

#### 3.1.2 Aprendizaje máquina

Siendo honestos, una red neuronal no aprende, al menos no como entendemos los humanos el aprendizaje. Lo que llamamos aprender en el contexto de una red neuronal debería llamarse: La búsqueda de ajustar los valores de los parámetros hasta obtener un modelo que resuelve un problema específico. Pero por razones de marketing se llamó *machine learning*.

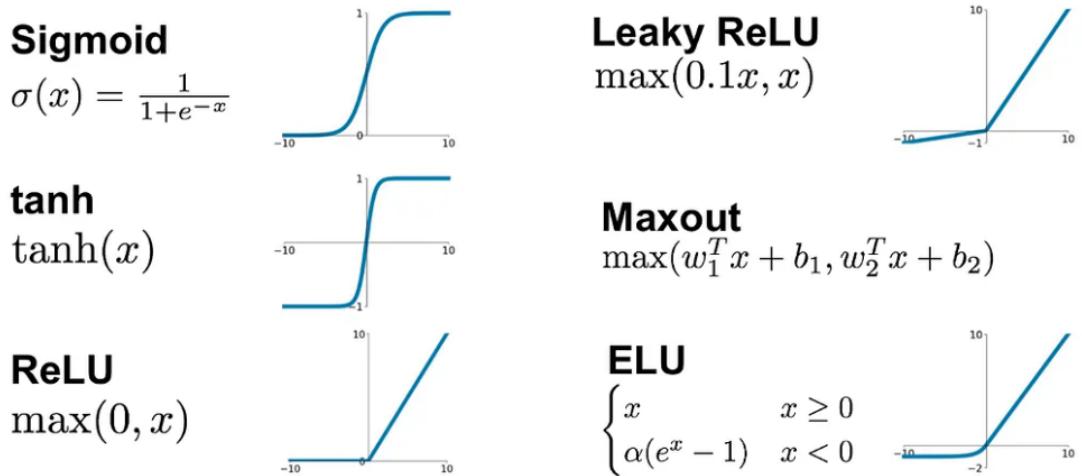


Figura 3.2: Ejemplos de las funciones de activación comunes. [12]

En esta sección vamos a definir la **función de coste** (*loss function* en inglés), la cual nos permite evaluar que tan bien se aproximan los resultados devueltos por nuestra red neuronal a los reales, cuanto más alto sea el valor de la función de coste, más alejados de la realidad estarán los valores que devuelve nuestra red. Entonces podemos interpretar que nuestra red “aprende” o mejora si conseguimos minimizar la función de coste. Por sencillez nosotros utilizaremos el error cuadrático medio como nuestra función de coste,  $C$ , que definimos como:

$$C = \frac{1}{2N} \sum_X (\hat{y}(X) - a^L(X))^2$$

Donde  $N$  es el número de ejemplos en el conjunto de entrenamiento,  $X$  son los ejemplos individuales del conjunto de entrenamiento,  $\hat{y}(X)$  es el vector con los valores reales del ejemplo  $X$ ,  $L$  es el numero de capas de la red ,  $a^L(X)$  es el vector con los valores de salida de las funciones de activación de la última capa.

Recordemos que nuestro conjunto de entrenamiento esta formado por la colección  $\mathcal{X} = \{(X_i, \hat{y}_i)\}$  con  $i = 1..N$ , por ejemplo una imagen  $X_1$  es un vector formado por las intensidades de los píxeles  $x_k$ ,  $k = 1.., 196608$ , que tiene asociado el vector  $\hat{y}_1$  formado las  $y_j$ ,  $j = 1.., 3$  clases en las que etiquetamos las imágenes.

Vamos a centrarnos en un solo ejemplo  $X$ .

$$C_X = \frac{1}{2} \sum_j^m (\hat{y}_j(X) - a_j^L(X))^2$$

### 3. REDES NEURONALES. UN ENFOQUE MATEMÁTICO

---

Entonces el coste de todos los ejemplos lo podemos expresar como:

$$C = \frac{1}{N} \sum_{i=1}^N C_{X_i}$$

Veamos como se relacionan las funciones  $C$  y  $F$ . Al inicio de este capítulo decíamos que podemos entender la red neuronal como una función

$$F : \mathbb{R}^n \rightarrow \mathbb{R}^m$$

También la podemos expresar como  $F(X; \theta)$ , donde  $\theta$  es el conjunto de los parámetros, es decir los pesos y sesgos. Esta función toma como variables los ejemplos  $X$ , mientras que los parámetros han sido inicializados aleatoriamente y son constantes para esta función. Observemos que la salida de la red neuronal para un ejemplo concreto es el vector formado por las funciones de activación de la última capa, es decir  $F(X; \theta) = (y_i) = a^L(X)$ .

Análogamente podemos entender la función de coste  $C_X$  como

$$C_X : \mathbb{R}^p \rightarrow \mathbb{R}$$

Donde  $p$  es el número de parámetros (pesos y sesgos). Para ser más exactos, vamos a expresar la función como  $C_X(\theta; F(X, \theta), \hat{y})$ . En este caso nuestras variables son los parámetros, mientras que  $\hat{y}$  es conocido y la salida de  $F(X, \theta)$  también se calcula previamente durante la propagación hacia adelante.

Estas dos funciones interactúan iterativamente entre ellas, primero en la **propagación hacia adelante** (paso *forward*) calculamos los valores de las distintas funciones de activación de  $F$  y obtenemos su salida que serán nuestras predicciones. Después calculamos el **error cometido** a partir de  $C_X(\theta; F(X, \theta), \hat{y})$ . Sobre esta función aplicamos la **retro-propagación**, que es una forma de obtener los gradientes de los pesos y sesgos, además aplicaremos el **descenso de gradiente** para actualizar los parámetros que volveremos a pasar a  $F$  con la esperanza de mejorar nuestro resultado, cuando entrenamos un modelo estamos iterando sobre este ciclo. Los pasos descritos podemos resumirlos con el siguiente algoritmo en forma de pseudocódigo.

---

**Algorithm 1** Algoritmo de Retropropagación y Descenso de Gradiente

---

```

1: Inicializar los pesos  $W^l$  y sesgos  $b^l$  para cada capa  $l$ 
2: for cada época do
3:   Realizar el paso forward para calcular  $a^l$  para cada capa  $l$ 
4:   Calcular el costo  $C$  usando  $a^L$                                  $\triangleright$  — Iniciar Retropropagación —
5:   Calcular  $\delta^L = \nabla_a C \odot \sigma'(z^L)$                    $\triangleright$  Error en capa de salida
6:   for  $l = L - 1, L - 2, \dots, 1$  do
7:     Calcular  $\delta^l = (W^{l+1})^T \delta^{l+1} \odot \sigma'(z^l)$ 
8:     Calcular  $\frac{\partial C}{\partial W^l} = \delta^l (a^{l-1})^T$ 
9:     Calcular  $\frac{\partial C}{\partial b^l} = \delta^l$ 
10:    end for                                                  $\triangleright$  — Fin de la Retropropagación —
11:    for  $l = 1, 2, \dots, L$  do                                 $\triangleright$  — Descenso de Gradiente —
12:      Actualizar  $W^l := W^l - \eta \frac{\partial C}{\partial W^l}$ 
13:      Actualizar  $b^l := b^l - \eta \frac{\partial C}{\partial b^l}$ 
14:    end for                                               $\triangleright$  — Fin del Descenso de Gradiente —
15:    Evaluar el rendimiento en el conjunto de validación (opcional)
16:  end for

```

---

### 3.1.2.1 Propagación hacia adelante

Durante esta fase, los datos de entrada se propagan a través de la red neuronal desde la capa de entrada hasta la capa de salida. Se calculan las funciones de activación de las distintas capas y las predicciones del modelo.

Veamos un ejemplo sencillo en una red con arquitectura 4-1-1 como la figura 3.3.

La red tiene 3 capas, con 4 variables en la capa de entrada, 1 neurona en la capa oculta, y una neurona en la capa de salida. La función de activación  $\sigma$  en la capa oculta toma el siguiente valor como entrada.

$$z_1^1 = w_{1,1}^1 x_1 + w_{1,2}^1 x_2 + w_{1,3}^1 x_3 + w_{1,4}^1 x_4 + b_1^1$$

Llamaremos  $a_1^1 = \sigma(z_1^1)$  al valor que devuelve la función de activación y que en este caso será el valor de entrada para la neurona en la capa de salida.

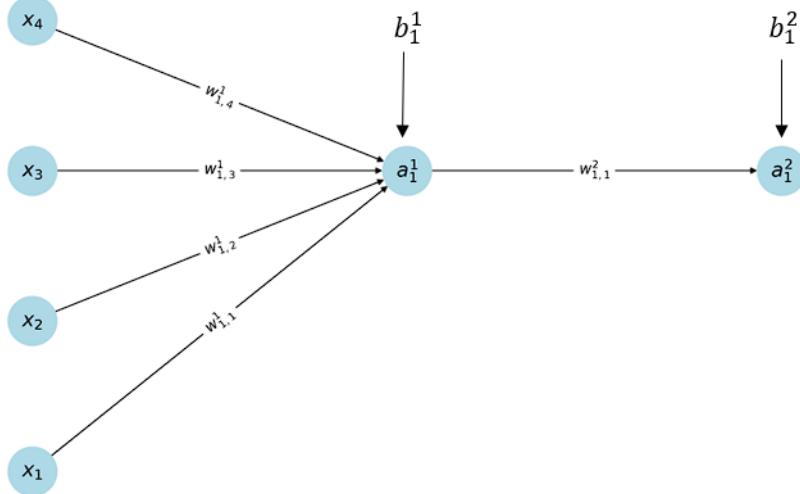
$$a_1^2 = \sigma(w_{1,1}^2 a_1^1 + b_1^2)$$

Para simplificar, definiremos:

$$z_1^2 = w_{1,1}^2 a_1^1 + b_1^2$$

### 3. REDES NEURONALES. UN ENFOQUE MATEMÁTICO

---



**Figura 3.3:** Red neuronal con arquitectura 4-1-1

entonces

$$a_1^2 = \sigma(z_1^2)$$

Para cada neurona  $j$  en cada capa  $l$ , calculamos  $z_j^l$  y  $a_j^l$  como

$$z_j^l = \sum_k w_{j,i}^l a_k^{l-1} + b_j^l, \quad a_j^l = \sigma(z_j^l).$$

#### 3.1.2.2 Retroproyagación (o Backpropagation)

En esta fase, el gradiente de la función de pérdida se calcula retrocediendo desde la capa de salida hacia la capa de entrada de la red neuronal. Esto se hace aplicando la regla de la cadena para calcular cómo afectan los cambios en los pesos y sesgos a la pérdida. El gradiente resultante se utiliza para actualizar los parámetros de la red utilizando el descenso del gradiente.

Veamos matemáticamente como se lleva a cabo la retropropagación. Introduciremos una variable intermedia,  $\delta_j^l$ , la cual viene a representar como afecta la suma ponderada  $z_j^l$  de la neurona  $j$  en la capa  $l$  a la función de coste final, es decir, nos da una medida de que tan responsable es dicha neurona sobre el error cometido. La retropropagación nos proporciona un proceso de computación del error  $\delta_j^l$  que luego se relacionará con  $\frac{\partial C}{\partial w_{j,i}^l}$  y  $\frac{\partial C}{\partial b_j^l}$ . Los cuales son los gradientes que buscamos obtener.

### 3.1 Redes neuronales discriminativas

---

Para cada neurona  $j$  en cada capa  $l$ , necesitamos calcular el error  $\delta_j^l$  asociado a dicha neurona. Este error se calcula de manera diferente para la última capa  $L$  y para las capas ocultas.

Definimos el error de la neurona  $j$  en la capa  $l$ ,  $\delta_j^l$ , como

$$\delta_j^l = \frac{\partial C}{\partial z_j^l}$$

Utilizaremos la notación  $\delta^l$  para denotar el vector de errores asociados con la capa  $l$ .

Sigamos con el ejemplo de nuestra red simplificada 4-1-1 y calculemos el error en la última capa de nuestro ejemplo anterior. La función de coste es  $C = \frac{1}{2}(\hat{y}_1 - a_1^2)^2$

En este caso,

$$\delta^L = \delta^2 = \delta_1^2 = \frac{\partial C}{\partial z_1^2} = (\hat{y}_1 - a_1^2)(a_1^2)' = (\hat{y}_1 - a_1^2)\sigma'(z_1^2)$$

En general se cumple que el error asociado a la neurona  $j$  en la última capa  $L$  es

$$\delta_j^L = \frac{\partial C}{\partial a_j^L} \sigma'(z_j^L)$$

*Demostración.* Por definición

$$\delta_j^l = \frac{\partial C}{\partial z_j^l}$$

Aplicamos la regla de la cadena para encontrar este gradiente:

$$\delta_j^L = \sum_k^{n^L} \frac{\partial C}{\partial a_k^L} \cdot \frac{\partial a_k^L}{\partial z_j^L}$$

Como la función de activación  $a_k^L$  de la  $k$ -ésima neurona depende solo de la suma ponderada  $z_j^L$  de la  $j$ -ésima neurona cuando  $k = j$ , el término  $\frac{\partial a_k^L}{\partial z_j^L}$  es nulo si  $k \neq j$ . Por lo que nos queda la expresión

$$\delta_j^L = \frac{\partial C}{\partial a_j^L} \cdot \frac{\partial a_j^L}{\partial z_j^L}$$

El primer término,  $\frac{\partial C}{\partial a_j^L}$ , es la derivada parcial de la función de coste respecto a la salida  $a_j^L$  de la neurona  $j$  en la capa  $L$ .

El segundo término,  $\frac{\partial a_j^L}{\partial z_j^L}$ , es la derivada de la función de activación,  $a_j^L = \sigma(z_j^L)$ .

Por lo tanto, podemos sustituir estos términos en la ecuación original:

$$\delta_j^L = \frac{\partial C}{\partial a_j^L} \cdot \sigma'(z_j^L)$$

□

### 3. REDES NEURONALES. UN ENFOQUE MATEMÁTICO

---

El primer término de esta expresión, indica como cambia la función de coste respecto la salida de la función de activación  $j$  de la última capa. Además  $\sigma'(z_j^L)$  es el ratio de cambio de la función de activación sigma en  $z_j^L$ .

Veamos como podemos expresarlo con operaciones matriciales, para ello definimos el producto de Hadamard.

#### Producto de Hadamard

Sean dos matrices  $A$  y  $B$  de la misma dimensión  $m \times n$ , el producto de Hadamard,  $A \odot B$ , devuelve otra matriz de la misma dimensión  $m \times n$ , cuyos elementos vienen dados por el producto  $a_{i,j} b_{i,j}$

Por ejemplo. Dadas las matrices  $A$  y  $B$  como:

$$A = \begin{pmatrix} 3 & 5 & 7 \\ 4 & 0 & 2 \end{pmatrix}$$

$$B = \begin{pmatrix} 1 & 6 & 0 \\ 3 & 7 & 1 \end{pmatrix}$$

El producto de Hadamard  $C = A \odot B$  se obtiene como:

$$\begin{aligned} C &= \begin{pmatrix} 3 \times 1 & 5 \times 6 & 7 \times 0 \\ 4 \times 3 & 0 \times 7 & 2 \times 1 \end{pmatrix} \\ &= \begin{pmatrix} 3 & 30 & 0 \\ 12 & 0 & 2 \end{pmatrix} \end{aligned}$$

Observemos como la expresión matricial para calcular  $\delta^L$  viene dada por

$$\delta^L = \left[ \frac{\partial C}{\partial a_1}, \dots, \frac{\partial C}{\partial a_{n^L}} \right] \odot \sigma'(z^L)$$

$z^L$  es el vector de los  $z_j$ , con  $j = 1, \dots, n^L$ . Luego el producto de Hadamard está bien definido entre dos vectores de dimensión  $1 \times n$ .

Ahora vamos a calcular el error de las capas intermedias. Siguiendo con nuestro ejemplo de la red con arquitectura 4-1-1, calculemos el error en la capa oculta 1, en este caso

$$\frac{\partial C}{\partial z_1^1} = (\hat{y}_1 - a_1^2) \frac{\partial a_1^2}{\partial z_1^1} \tag{3.1}$$

Recordemos que  $z_1^2 = w_{1,1}^2 \sigma(z_1^1) + b_1^2$ , aplicando la regla de la cadena

$$\frac{\partial a_1^2}{\partial z_1^1} = \sigma'(z_1^2) \frac{\partial z_1^2}{\partial z_1^1} = \sigma'(z_1^2) w_{1,1}^2 \sigma'(z_1^1) \tag{3.2}$$

### 3.1 Redes neuronales discriminativas

---

Notemos que  $(\hat{y}_1 - a_1^2)\sigma'(z_1^2) = \frac{\partial C}{\partial z_1^2} = \delta_1^2$ . Es decir que podemos reescribir nuestra expresión anterior como

$$\frac{\partial C}{\partial z_1^1} = \delta_1^2 w_{1,1}^2 \sigma'(z_1^1) \quad (3.3)$$

La segunda ecuación del algoritmo de retropropagación expresa el error de la capa actual  $\delta^l$  en términos del error en la siguiente capa,  $\delta^{l+1}$ , se cumple que:

$$\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l) \quad (3.4)$$

*Demostración.* Aplicamos la regla de la cadena

$$\delta_j^l = \frac{\partial C}{\partial z_j^l} = \sum_k^{n^{l+1}} \frac{\partial C}{\partial a_k^{l+1}} \frac{\partial a_k^{l+1}}{\partial z_k^{l+1}} \frac{\partial z_k^{l+1}}{\partial a_j^l} \frac{\partial a_j^l}{\partial z_j^l} \quad (3.5)$$

Por definición de  $\delta_k^{l+1}$  tenemos que

$$\delta_k^{l+1} = \sum_j^{n^{l+1}} \frac{\partial C}{\partial a_k^{l+1}} \frac{\partial a_k^{l+1}}{\partial z_k^{l+1}} \quad (3.6)$$

Por otro lado observemos que

$$z_k^{l+1} = \sum_j^{n^l} w_{k,j}^{l+1} a_j^l + b_k^{l+1} = \sum_j^{n^l} w_{k,j}^{l+1} \sigma(z_j^l) + b_k^{l+1}$$

Luego la derivada  $\frac{\partial z_k^{l+1}}{\partial a_j^l} \frac{\partial a_j^l}{\partial z_j^l}$  nos queda

$$w_{k,j}^{l+1} \sigma'(z_j^l) \quad (3.7)$$

sustituyendo en la expresión (3.5)

$$\delta_j^l = \sum_k^{n^{l+1}} \delta_k^{l+1} w_{k,j}^{l+1} \sigma'(z_j^l) \quad (3.8)$$

□

Esto es lo mismo que la ecuación (3.4) pero expresado con sus componentes.

Por último vamos a obtener las ecuaciones que realmente nos interesan para aplicar el algoritmo del descenso de gradiente, el ratio de cambio de la función coste respecto los parámetros de sesgo,  $\frac{\partial C}{\partial b_j^l}$ , y, el ratio de cambio de la función coste respecto los pesos  $\frac{\partial C}{\partial w_{j,k}^l}$ .

$$\frac{\partial C}{\partial b_j^l} = \delta_j^l \quad (3.9)$$

### 3. REDES NEURONALES. UN ENFOQUE MATEMÁTICO

---

*Demostración.* La suma ponderada  $z_j^l$  para la neurona  $j$  en la capa  $l$  se define como:

$$z_j^l = \sum_k^{n^{l-1}} w_{j,k}^l a_k^{l-1} + b_j^l \quad (3.10)$$

Derivando  $z_j^l$  respecto a  $b_j^l$ , obtenemos que la derivada es simplemente 1:

$$\frac{\partial z_j^l}{\partial b_j^l} = 1 \quad (3.11)$$

Entonces, aplicando la regla de la cadena:

si  $l = L$

$$\frac{\partial C}{\partial b_j^L} = \sum_k^{n^L} \frac{\partial C}{\partial a_k^L} \frac{\partial a_k^L}{\partial z_j^L} \frac{\partial z_j^L}{\partial b_j^L} = \delta_j^L \cdot 1 = \delta_j^L \quad (3.12)$$

si  $l \neq L$

$$\frac{\partial C}{\partial b_j^l} = \sum_k^{n^{l+1}} \frac{\partial C}{\partial a_k^{l+1}} \frac{\partial a_k^{l+1}}{\partial z_k^{l+1}} \frac{\partial z_k^{l+1}}{\partial a_j^l} \frac{\partial a_j^l}{\partial z_j^l} \frac{\partial z_j^l}{\partial b_j^l} \quad (3.13)$$

Utilizando la expresión (3.5) nuestra ecuación queda así

$$\frac{\partial C}{\partial b_j^l} = \sum_k^{n^{l+1}} \delta_k^{l+1} w_{k,j}^{l+1} \sigma'(z_j^l) \cdot 1 = \delta_j^l \quad (3.14)$$

□

Calculemos el gradiente de los pesos de la capa  $l$

$$\frac{\partial C}{\partial w_{j,k}^l} = a_k^{l-1} \delta_j^l \quad (3.15)$$

*Demostración.* Partimos de nuevo de la suma ponderada  $z_j^l$ :

$$z_j^l = \sum_k^{n^{l-1}} w_{j,k}^l a_k^{l-1} + b_j^l \quad (3.16)$$

Esta vez, derivamos  $z_j^l$  respecto al peso  $w_{j,k}^l$ . La derivada es  $a_k^{l-1}$ :

$$\frac{\partial z_j^l}{\partial w_{j,k}^l} = a_k^{l-1} \quad (3.17)$$

Por lo tanto, usando la regla de la cadena:

si  $l = L$

$$\frac{\partial C}{\partial w_{j,k}^L} = \sum_i^{n^L} \frac{\partial C}{\partial a_i^L} \frac{\partial a_i^L}{\partial z_j^L} \frac{\partial z_j^L}{\partial w_{j,k}^L} = \delta_j^l \cdot a_k^{L-1} \quad (3.18)$$

si  $l \neq L$

$$\frac{\partial C}{\partial w_{j,k}^l} = \sum_i^{n^{l+1}} \frac{\partial C}{\partial a_i^{l+1}} \frac{\partial a_i^{l+1}}{\partial z_i^{l+1}} \frac{\partial z_i^{l+1}}{\partial a_j^l} \frac{\partial a_j^l}{\partial z_j^l} \frac{\partial z_j^l}{\partial w_{j,k}^l} \quad (3.19)$$

Sustituyendo por las ecuaciones del error de la neurona  $j$  en la capa  $l$  nos queda

$$\frac{\partial C}{\partial w_{j,k}^l} = \sum_i^{n^{l+1}} \delta_i^{l+1} w_{i,j}^{l+1} \sigma'(z_j^l) a_k^{l-1} = \delta_j^l a_k^{l-1} \quad (3.20)$$

□

Observemos que tenemos lo necesario para calcular cada una de estas ecuaciones, calculamos primero el error de la capa  $L$ ,  $\delta_j^L$ , el cual utilizamos para calcular el error en la capa  $L-2$  gracias a la ecuación  $\delta_j^l$ , procediendo iterativamente podemos obtener los errores de cada capa. Notemos que no hemos usado ninguna propiedad particular de la función de activación Sigmoide, por lo que estas ecuaciones de retropropagación sirven para cualquier función de activación.

### 3.1.2.3 Optimización por descenso de gradiente

El descenso del gradiente es un método de optimización utilizado para ajustar iterativamente los parámetros de un modelo de aprendizaje automático, como una red neuronal, con el objetivo de minimizar una función de pérdida. El algoritmo de descenso del gradiente se basa en la idea de que si conocemos el gradiente de la función de pérdida en un punto, podemos dar pasos pequeños en la dirección opuesta al gradiente para encontrar un mínimo local de la función.

A partir de ahora, por simplificar la notación, actuaremos sobre un ejemplo concreto  $C_X$  que llamaremos  $C$ . Una vez calculados  $\frac{\partial C}{\partial w^l}$  y  $\frac{\partial C}{\partial b^l}$  que son los vectores gradientes de los pesos y los sesgos respectivamente. El siguiente paso es actualizarlos.

$$w^l \rightarrow w'^l = w^l - \alpha \frac{\partial C}{\partial w^l} \quad (3.21)$$

### 3. REDES NEURONALES. UN ENFOQUE MATEMÁTICO

---

$$b^l \rightarrow b^{l'} = b^l - \alpha \frac{\partial C}{\partial b^l} \quad (3.22)$$

Tenemos que  $\alpha$  es el tamaño de paso, un hiperparámetro, que, en la jerga del *machine learning*, se llama ratio de aprendizaje, cuyo valor se utiliza para controlar la velocidad en el proceso de aprendizaje.

El vector gradiente de la función de coste  $C$  esta formado por

$$\nabla C = \left[ \frac{\partial C}{\partial b^1}, \frac{\partial C}{\partial w^1}, \dots, \frac{\partial C}{\partial b^l}, \frac{\partial C}{\partial w^l}, \dots, \frac{\partial C}{\partial b^L}, \frac{\partial C}{\partial w^L} \right] \quad (3.23)$$

Pero no olvidemos que estos son los vectores gradientes de una función de coste  $C_X$ . Realmente deberíamos calcular el gradiente de los pesos y sesgos para todos los ejemplos  $X_i$

$$\frac{\partial C}{\partial w^l} = \frac{1}{N} \sum_{i=1}^N \frac{\partial C_{X_i}}{\partial w^l} \quad (3.24)$$

$$\frac{\partial C}{\partial b^l} = \frac{1}{N} \sum_{i=1}^N \frac{\partial C_{X_i}}{\partial b^l} \quad (3.25)$$

En la práctica, debido al enorme coste computacional que tendría aplicar este algoritmo para todos los ejemplos  $X$  de la colección  $\mathcal{X}$ , se utiliza el **Descenso de gradiente estocástico**. El cual consiste en tomar pequeñas muestras aleatorias (*mini-batches*) de la colección de ejemplos, 32, 64 o 128 son las cantidades que se suelen escoger.

#### 3.1.3 Métricas y evaluación de Redes Neuronales

Las métricas de evaluación están diseñadas para proporcionar una comprensión de la efectividad de un modelo desde el punto de vista de un observador externo o en términos de rendimiento en una tarea específica. A diferencia de la función de pérdida, estas métricas no son utilizadas directamente en el proceso de entrenamiento de la red neuronal. En cambio, se usan después del entrenamiento para evaluar y comparar modelos, o para comunicar el rendimiento del modelo a terceros. Estas son algunas de las métricas más comunes a la hora de evaluar los modelos.

### **3.1 Redes neuronales discriminativas**

---

#### **3.1.3.1 Exactitud (*Accuracy*)**

La exactitud es una de las métricas más intuitivas y comúnmente utilizadas en la evaluación de modelos de clasificación. Se define como la proporción de predicciones correctas (tanto positivas como negativas) respecto al total de predicciones realizadas. Matemáticamente, se expresa como:

$$\text{Accuracy} = \frac{\text{VP} + \text{VN}}{\text{VP} + \text{VN} + \text{FP} + \text{FN}} \quad (3.26)$$

donde VP representa los Verdaderos Positivos, VN los Verdaderos Negativos, FP los Falsos Positivos, y FN los Falsos Negativos. Aunque la *Accuracy* proporciona una visión general rápida del rendimiento del modelo, puede ser engañosa en conjuntos de datos desequilibrados, donde una clase es mucho más frecuente que las otras.

#### **3.1.3.2 Precisión (*Precision*)**

La precisión, en el contexto de la clasificación, mide la proporción de predicciones positivas que fueron efectivamente correctas. Es particularmente útil en escenarios donde los falsos positivos son más críticos. Se calcula como:

$$\text{Precision} = \frac{\text{Verdaderos Positivos (VP)}}{\text{Verdaderos Positivos (VP)} + \text{Falsos Positivos (FP)}} \quad (3.27)$$

Esta métrica es crucial en situaciones donde es más costoso clasificar erróneamente un negativo como positivo, como en ciertos diagnósticos médicos.

#### **3.1.3.3 Sensibilidad (*Recall*)**

El *recall* mide la capacidad del modelo para detectar correctamente las predicciones positivas de entre todas las predicciones positivas reales. Es especialmente importante en situaciones donde los falsos negativos son críticos. Se define como:

$$\text{Recall} = \frac{\text{Verdaderos Positivos (VP)}}{\text{Verdaderos Positivos (VP)} + \text{Falsos Negativos (FN)}} \quad (3.28)$$

El *recall* es fundamental en contextos donde no detectar un positivo tiene mayores consecuencias, como en la detección de fraudes.

### **3. REDES NEURONALES. UN ENFOQUE MATEMÁTICO**

---

#### **3.1.3.4 F1-Score**

El F1-Score es una métrica que combina la precisión y el recall en un solo valor, proporcionando un balance entre ambos. Es particularmente útil cuando se desea un equilibrio entre evitar falsos positivos y falsos negativos. Se calcula como el promedio armónico de precisión y recall:

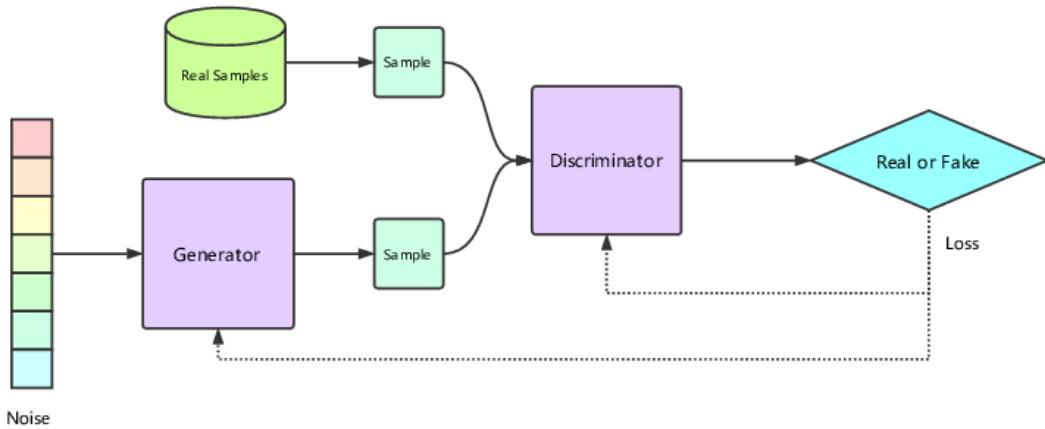
$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (3.29)$$

Esta métrica es altamente efectiva para comparar modelos en situaciones donde tanto los falsos positivos como los falsos negativos son importantes.

## 3.2 Redes Generadoras Adversarias - GANs

Las Redes Generadoras Adversarias (GANs, por sus siglas en inglés de Generative Adversarial Networks) representan una clase de algoritmos de aprendizaje automático en el ámbito de las redes neuronales. Introducidas por Ian Goodfellow y sus colaboradores en 2014 [7], las GANs han ganado una atención significativa debido a su capacidad para generar datos que son estadísticamente similares a los datos de entrenamiento. Esto las convierte en una herramienta poderosa para una variedad de aplicaciones que van desde la generación de imágenes y texto hasta tareas más complejas como la traducción de imágenes y el aprendizaje por refuerzo.

En términos generales, una GAN está compuesta por dos redes neuronales que funcionan en conjunto: el generador,  $G$ , y el discriminador,  $D$ . El generador tiene como objetivo generar datos que imitan la distribución de los datos reales, mientras que el discriminador aspira a diferenciar entre los datos generados y los datos reales. El entrenamiento de estas redes se realiza de forma adversaria, en un juego de suma cero que lleva a un equilibrio en el que el generador produce datos que el discriminador ya no puede distinguir de los reales.



**Figura 3.4:** Diagrama de la estructura básica de una GAN. Fuente [9]

Tanto  $D$  como  $G$  son modelos de redes neuronales y son entrenados con el algoritmo de retropropagación.

La pregunta que me surge es, ¿Por qué no entrenamos una red generadora de la misma forma que hemos entrenado el modelo clasificador?, ¿Por qué ahora estamos hablando de distribuciones de los datos y funciones objetivo que involucran dos modelos compitiendo entre ellos? y ¿Dónde aplicamos el algoritmo de retropropagación?

### **3. REDES NEURONALES. UN ENFOQUE MATEMÁTICO**

---

Vamos a contestar a estas preguntas. Pero primero recordemos el objetivo de nuestro proyecto. Dada la falta de datos reales etiquetados, buscamos ser capaces de generar de forma artificial y consistente, imágenes similares a las obtenidas con la máquina de ultrasonidos y que forman nuestro conjunto de datos reales.

¿Cómo cuantificamos que dos imágenes son similares? Podríamos pensar en usar un tipo de distancia, de esta forma dos objetos son parecidos si las distancias de sus componentes es pequeña. Pero esa idea no es útil en este contexto, no queremos que nuestro modelo genere imágenes con distancias cortas a ciertas imágenes del conjunto original  $X$ . En vez de eso, queremos generar una imagen, que igual no tiene la distancia más corta con las imágenes de entrenamiento, pero que al verla sentimos que pertenece a una de las categorías. Queremos una imagen que un neonatólogo al verla piense que es nueva, y no una copia de una muestra tomada anteriormente.

Esta originalidad en la creación de datos no es posible conseguirla con el entrenamiento habitual. Supongamos que seguimos empeñados en usar una función de coste como el error cuadrático medio. Si queremos diversidad en las imágenes generadas deberíamos etiquetar con clases cada imagen de entrenamiento lo cual ya es un desafío y un trabajo enorme, por hacerlo mas concreto, si nos interesa un modelo generador de caras humanas, ¿Deberíamos tener una etiqueta para cada posible variación que se puede dar? algo así como, 'cara con ojos marrones y nariz ancha que mira hacia arriba', 'cara con ojos marrones y nariz ancha con pecas que mira a la derecha', 'cara con ojos verdes y boca cerrada que mira a la derecha'... No parece un solución viable además del enorme trabajo de etiquetado que supondría. Por último puede ser más difícil optimizar la función de coste error cuadrático medio en un contexto donde no tienes etiquetas claras para tus datos generados, es decir, imaginemos que nuestro modelo está en proceso de aprendizaje y genera una imagen de una cara sin nariz y con los dos ojos en el lado derecho, ¿Tenemos una etiqueta para ello?, ¿A que otra imagen etiquetada se supone que se aproxima más?. Por estos inconvenientes en la siguiente sección planteamos una forma de aprendizaje máquina distinta a la de una red neuronal discriminativa.

#### **3.2.1 Fundamentos teóricos**

Por las razones mencionadas, adoptamos una perspectiva diferente en cuanto a las GANs. Definimos la similitud entre objetos como una distribución de probabilidad. Dos conjuntos

de datos son considerados similares si sus muestras tienen una misma distribución de probabilidad. Si tenemos nuestro conjunto de entrenamiento  $X \in \mathbb{R}^n$  con elementos con una distribución  $p_{\text{data}}$ (con función de densidad  $p_{\text{data}}(x)$  ), nosotros queremos generar muestras con distribución de probabilidad  $p_g$ (con función de densidad  $p_g(x)$  ) de tal forma que  $p_g$  es una buena aproximación de  $p_{\text{data}}$ . Por supuesto la distribución  $p_{\text{data}}$  es desconocida, solo tenemos un conjunto de datos finitos (nuestro conjunto de entrenamiento) que proviene de esta distribución.

El funcionamiento de las GANs se puede explicar como la interacción entre dos redes neuronales, una generativa,  $G$ , y una discriminativa,  $D$ . La red  $G$  con parámetros  $\theta_g$ ,  $G(z; \theta_g)$ , transforma un ruido inicial  $z$ , proveniente de  $p_z$ , en datos que buscan imitar la distribución real  $p_{\text{data}}$ . Por otro lado, la red  $D$ , con parámetros  $\theta_d$ ,  $D(x; \theta_d)$ , evalúa las muestras que recibe y asigna un probabilidad de que estas pertenezcan a la distribución de datos reales,  $p_{\text{data}}$ , en lugar de la distribución generada por  $G$ ,  $p_g$ . Durante el entrenamiento  $G$  y  $D$  ajustan los parámetros, mediante el algoritmo de retropropagación y descenso de gradiente, para lograr superarse mutuamente.

### 3.2.1.1 Función objetivo

La función objetivo min-max  $V(D, G)$  para una GAN (Red Generativa Antagónica) típica se define como:

$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_{\text{data}}(x)} [\log D(x)] + \mathbb{E}_{z \sim p_z(z)} [\log(1 - D(G(z)))]$$

Vamos a recapitular los términos y elementos presentados hasta ahora.

- $\min_G$  y  $\max_D$  indican que el Generador (G) intenta minimizar la función objetivo, mientras que el Discriminador (D) intenta maximizarla.
- $\mathbb{E}$  representa la esperanza matemática.
- $x$  son los datos reales, en este caso imágenes, y  $x \sim p_{\text{data}}(x)$  significa que  $x$  proviene de la distribución de datos reales  $p_{\text{data}}$ .
- $z$  es un vector de ruido aleatorio, y  $z \sim p_z(z)$  significa que  $z$  proviene de una distribución de ruido  $p_z$ .

### 3. REDES NEURONALES. UN ENFOQUE MATEMÁTICO

---

- El generador  $G : \mathbb{R}^k \rightarrow \mathbb{R}^n$  es una función diferenciable que transforma un vector de ruido  $z$  del espacio  $\mathbb{R}^k$  en una imagen  $G(z)$  en el espacio de imágenes  $\mathbb{R}^n$ .
- El discriminador  $D : \mathbb{R}^n \rightarrow [0, 1]$  es una función que toma una imagen  $x$  del espacio de imágenes  $\mathbb{R}^n$  y devuelve una probabilidad  $D(x)$  que representa la autenticidad percibida de  $x$ .
- $D(x)$  es la salida del Discriminador al recibir una muestra real  $x$ , y  $D(G(z))$  es la salida del Discriminador al recibir una muestra generada  $G(z)$ .

Los términos de la función objetivo se dividen en:

1.  $\mathbb{E}_{x \sim p_{\text{data}}(x)}[\log D(x)]$ : Este término representa la habilidad del Discriminador para identificar correctamente los datos reales como reales. El objetivo del Discriminador es maximizar este valor.
2.  $\mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))]$ : Este término representa la habilidad del Discriminador para identificar los datos generados como falsos. El Discriminador quiere maximizar este valor, mientras que el Generador quiere minimizarlo.

Nuestro objetivo, ahora, es probar que esta función objetivo tiene solución óptima y que podemos alcanzarla. Lo dividiremos en varias etapas.

Primero fijaremos  $G$  para encontrar el máximo valor  $D$  en  $V(D, G)$ , este es el discriminador óptimo  $D_G^*$ . En el siguiente paso demostraremos que el mínimo se alcanza cuando  $p_g = p_{\text{data}}$  y que este es mínimo global. Por último probaremos como el algoritmo de entrenamiento conseguirá que nuestra GAN, bajo 2 hipótesis bastante ideales, hace converger  $p_g$  hacia  $p_{\text{data}}$ .

#### 3.2.1.2 Discriminador óptimo

A continuación, presentaremos un teorema que será utilizado en la demostración de la siguiente proposición. Este teorema, no lo voy a demostrar debido a su complejidad y longitud, su demostración implica utilizar el teorema de Radon-Nikodym y otros resultados de la teoría de la medida. La demostración podemos encontrarla en [13]

Debo comentar que hay muchas demostraciones en internet, que, en el contexto de las GANs, utilizan la transformación mediante cambio de variables para su demostración. Esto

### 3.2 Redes Generadoras Adversarias - GANs

---

es un error, ya que requiere de la existencia de  $G^{-1}$ , lo cual no podemos dar por hecho, es más, en general para una red neuronal no existe.

**Teorema 1.** *Sea  $h$  una función medible,  $X$  una variable aleatoria con función de densidad  $f_X$ . La esperanza de  $h(X)$  se puede calcular como:*

$$\mathbb{E}[h(X)] = \int_x h(x) f_X(x) dx \quad (3.30)$$

**Proposición 2.** *Si fijamos  $G$ , el discriminador óptimo  $D$  es*

$$D_G^*(x) = \frac{p_{data}(x)}{p_{data}(x) + p_g(x)} \quad (3.31)$$

*Demostración.* Dado cualquier generador,  $G$ , el criterio de entrenamiento para  $D$  es maximizar  $V(G, D)$

$$\begin{aligned} V(G, D) &= \int_x p_{data}(x) \log(D(x)) dx + \int_z p_z(z) \log(1 - D(G(z))) dz \\ &= \int_x p_{data}(x) \log(D(x)) + p_g(x) \log(1 - D(x)) dx \end{aligned} \quad (3.32)$$

La última igualdad viene de usar en conjunto el resultado (3.30) y el hecho de que si  $z \sim p_z$  entonces  $G(z) = x \sim p_g$ . En este caso  $h(t) = \log(1 - D(t))$  es continua y por tanto medible,  $G$  es diferenciable por definición luego es medible también.

$$\begin{aligned} \mathbb{E}_{z \sim p_z} [\log(1 - D(G(z)))] &= \mathbb{E}_{z \sim p_z} [h(G(z))] \\ \mathbb{E}_{z \sim p_z} [h(G(z))] &= \int_{G(z)} h(G(z)) p_g dz \\ \int_{G(z)} h(G(z)) p_g dz &= \int_x h(x) p_g dx = \mathbb{E}_{x \sim p_g} [\log(1 - D(x))]. \end{aligned}$$

Para simplificar, consideramos el integrando (3.32) de la forma  $a \log(D) + b \log(1 - D)$  y queremos encontrar el valor de  $D$  que maximiza esta expresión. Donde  $a$  y  $b$  son las funciones de densidad  $p_{data}(x), p_z(x)$  respectivamente.

Primero, diferenciamos la expresión con respecto a  $D$ :

$$\frac{\partial}{\partial D} (a \log(D) + b \log(1 - D)) = \frac{a}{D} - \frac{b}{1 - D}$$

Luego, igualamos la derivada a cero y resolvemos para  $D$ :

### 3. REDES NEURONALES. UN ENFOQUE MATEMÁTICO

---

$$\frac{a}{D} - \frac{b}{1-D} = 0 \quad (3.33)$$

$$\frac{a}{D} = \frac{b}{1-D} \quad (3.34)$$

$$a(1-D) = bD \quad (3.35)$$

$$a = bD + aD \quad (3.36)$$

$$a = D(a+b) \quad (3.37)$$

$$D = \frac{a}{a+b} \quad (3.38)$$

Por lo tanto, cuando  $a$  y  $b$  son constantes con respecto a  $D$ , el máximo de la función se produce en  $D = \frac{a}{a+b}$ . Notemos que se cumple  $a \neq -b$ , al ser las funciones de densidad mayores que cero.

Para confirmar que este punto es un máximo, revisamos la segunda derivada:

$$\frac{\partial^2}{\partial D^2}(a \log(D) + b \log(1-D)) = -\frac{a}{D^2} - \frac{b}{(1-D)^2}$$

Ambos términos son negativos, ya que  $a$  y  $b$  son las funciones de densidad, con la propiedad de ser mayores o iguales a cero, lo que implica que la función es cóncava, confirmado que el punto es un máximo. □

#### 3.2.1.3 Mínimo global

Ahora podemos reformular la función objetivo, llamamos

$$C(G) = \max_D V(G, D) = \mathbb{E}_{x \sim p_{\text{data}}} [\log(D^*(x))] + \mathbb{E}_{x \sim p_g} [\log(1 - D^*(x))] =$$

$$\mathbb{E}_{x \sim p_{\text{data}}} \left[ \log \left( \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_g(x)} \right) \right] + \mathbb{E}_{x \sim p_g} \left[ \log \left( \frac{p_g(x)}{p_{\text{data}}(x) + p_g(x)} \right) \right]$$

El segundo sumando viene de realizar las operaciones

$$1 - D^*(x) = 1 - \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_g(x)} = \frac{p_{\text{data}}(x) + p_g(x)}{p_{\text{data}}(x) + p_g(x)} - \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_g(x)} = \frac{p_g(x)}{p_{\text{data}}(x) + p_g(x)}$$

A continuación definimos algunos conceptos importantes para la próxima demostración.

### Divergencia de Kullback-Leibler

La Divergencia de **Kullback-Leibler** para variables aleatorias continuas con densidades de probabilidad  $p(x)$  y  $q(x)$  está definida como:

$$D_{\text{KL}}(p \parallel q) = \int_{-\infty}^{\infty} p(x) \log \left( \frac{p(x)}{q(x)} \right) dx \quad (3.39)$$

### Divergencia de Jensen-Shannon

La Divergencia de **Jensen-Shannon** es una forma simétrica y suavizada de la divergencia de *Kullback-Leibler*, y se define como:

$$D_{\text{JS}}(p \parallel q) = \frac{1}{2} D_{\text{KL}}(p \parallel m) + \frac{1}{2} D_{\text{KL}}(q \parallel m) \quad (3.40)$$

donde  $m = \frac{1}{2}(p(x) + q(x))$ .

Intuitivamente la idea es que son una forma de medir cuánto se alejan entre sí dos distribuciones. En particular nos interesa la propiedad de que  $0 \leq D_{\text{JS}}(p \parallel q) \leq 1$

**Teorema 3.** El mínimo global de  $C(G)$  es alcanzado si y solo si  $p_g = p_{\text{data}}$ . En ese punto,  $C(G) = -\log(4)$

*Demostración.* Veamos primero el recíproco, si  $p_g = p_{\text{data}}$  entonces,  $D_G^* = \frac{1}{2}$

$$C(G) = \log\left(\frac{1}{2}\right) + \log\left(\frac{1}{2}\right) = -\log(2) - \log(2) = -2\log(2) = -\log(4).$$

Este valor es candidato a mínimo global, veamos que en efecto lo es.

Sabiendo que  $-\log 4$  es el candidato para el mínimo global, queremos introducir ese valor en la ecuación. Por lo tanto, sumamos y restamos  $\log 2$  en cada integral, multiplicado por las densidades de probabilidad.

$$C(G) = \int_x (\log 2 - \log 2)p_{\text{data}}(x) + p_{\text{data}}(x) \log \left( \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_g(x)} \right)$$

$$+ (\log 2 - \log 2)p_g(x) + p_g(x) \log \left( \frac{p_g(x)}{p_g(x) + p_{\text{data}}(x)} \right) dx$$

$$C(G) = -\log 2 \int_x (p_g(x) + p_{\text{data}}(x)) dx +$$

$$\int_x [p_{\text{data}}(x)(\log 2 + \log(\frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_g(x)})) + p_g(x)(\log 2 + \log(\frac{p_g(x)}{p_g(x) + p_{\text{data}}(x)}))] dx \quad (3.41)$$

### 3. REDES NEURONALES. UN ENFOQUE MATEMÁTICO

---

Debido a la definición de densidades de probabilidad, integrar  $p_G$  y  $p_{\text{data}}$  en su dominio equivale a 1, es decir:

$$-\log 2 \int_x (p_g(x) + p_{\text{data}}(x)) dx = -\log 2(1+1) = -2\log 2 = -\log 4 \quad (3.42)$$

Por otro lado, usando las propiedades de los logaritmos tenemos

$$\log 2 + \log \left( \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_g(x)} \right) = \log \left( 2 \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_g(x)} \right) = \log \left( \frac{p_{\text{data}}(x)}{\frac{p_{\text{data}}(x) + p_g(x)}{2}} \right)$$

Ahora sustituyendo, la ecuación (3.41) nos queda

$$\begin{aligned} C(G) &= -\log(4) + \int_x [p_{\text{data}}(x) \log \left( \frac{p_{\text{data}}(x)}{\frac{p_{\text{data}}(x) + p_g(x)}{2}} \right)] dx \\ &\quad \int_x [p_g(x) \log \left( \frac{p_g(x)}{\frac{p_{\text{data}}(x) + p_g(x)}{2}} \right)] dx \end{aligned} \quad (3.43)$$

Observamos que estas expresiones se pueden formular en términos de la divergencia *Kullback-Leibler*

$$C(G) = -\log(4) + D_{\text{KL}} \left( p_{\text{data}} \parallel \frac{p_{\text{data}}(x) + p_G(x)}{2} \right) + D_{\text{KL}} \left( p_G \parallel \frac{p_{\text{data}}(x) + p_G(x)}{2} \right) \quad (3.44)$$

y, por la definición de divergencia Jensen-Shannon

$$C(G) = -\log(4) + 2D_{\text{JS}}(p_{\text{data}} \parallel p_G) \quad (3.45)$$

Utilizando la propiedad de no negatividad,  $0 \leq D_{\text{JS}}(p \parallel q)$ , y que vale 0 cuando  $p = q$ . Hemos probado que  $C^* = -\log(4)$  es el mínimo global de  $C(G)$  y que cuando alcanza ese valor, necesariamente  $p_g = p_{\text{data}}$ .

□

Hasta ahora hemos visto que para un  $G$  fijado el discriminador tiene un óptimo,  $D_G^* = \frac{p_{\text{data}}}{p_{\text{data}} + p_g}$ , el cual no es calculable, pues desconocemos  $p_{\text{data}}$ . También hemos visto que  $C(G)$  tiene mínimo global y es cuando  $p_g = p_{\text{data}}$ , lo cual tiene sentido, el discriminador devuelve la probabilidad de  $D_G^* = 1/2$  ya que esta totalmente confundido y no puede diferenciar entre imágenes generadas y reales. Sin embargo aun no hemos hablado de como alcanzar este generador óptimo.

Para finalizar esta sección vamos a ver como el algoritmo planteado en el artículo [7] es capaz de converger, en teoría, a la solución óptima.

### 3.2.1.4 Algoritmo de convergencia

El algoritmo debería recordarnos al que vimos en el capítulo anterior, aplicamos de nuevo la técnica del descenso de gradiente estocástico, donde trabajamos con una muestra aleatoria de  $m$  ejemplos. Para el modelo discriminador actualizamos sus parámetros ascendiendo el gradiente, ya que buscamos maximizarlo. Para el modelo generador, descendemos el gradiente, ya que buscamos minimizarlo. En este último caso, notar que nos falta un sumando de la función objetivo original,  $\log D(x)$ , este término lo obviamos ya que no depende del generador y su gradiente sería cero.

---

**Algorithm 2** Algoritmo de Convergencia para GANs

H

```

1: for cada iteración de entrenamiento do
2:   for  $k$  pasos do
3:     Muestrear un mini-lote de  $m$  muestras de ruido a priori  $p_z(z)$ 
4:      $\{z^{(1)}, \dots, z^{(m)}\}$ 
5:     Muestrear un mini-lote de  $m$  ejemplos de la distribución generadora de datos
        $p_{\text{data}}(x) \{x^{(1)}, \dots, x^{(m)}\}$ 
6:     Actualizar el discriminador ascendiendo su gradiente estocástico
7:      $\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^m [\log(D(x^{(i)})) + \log(1 - D(G(z^{(i)})))]$ 
8:   end for
9:   Muestrear otro mini-lote de  $m$  muestras de ruido a priori  $p_z(z)$ 
10:   $\{z^{(1)}, \dots, z^{(m)}\}$ 
11:  Actualizar el generador descendiendo su gradiente estocástico
12:   $\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^m \log(1 - D(G(z^{(i)})))$ 
13: end for

```

---

**Proposición 4.** Si  $G$  y  $D$  tienen suficiente capacidad, y en cada paso del algoritmo, el discriminador es capaz de alcanzar su óptimo dado  $G$ , y  $p_g$  se actualiza para mejorar el criterio

$$\mathbb{E}_{x \sim p_{\text{data}}} [\log D_G^*(x)] + \mathbb{E}_{x \sim p_g} [\log(1 - D_G^*(x))]$$

Entonces  $p_g$  converge a  $p_{\text{data}}$ .

Desglosemos las hipótesis y sus problemas asociados.

$G$  y  $D$  tienen suficiente capacidad. Con esto nos referimos a que las redes neuronales  $G(z; \theta_g)$ ,  $D(x; \theta_d)$  tengan suficientes grados de libertad, dicho de otra forma, nuestra red neuronal debe ser lo suficientemente compleja para modelar una distribución  $p_{\text{data}}$  que, a priori, desconocemos su complejidad.

### 3. REDES NEURONALES. UN ENFOQUE MATEMÁTICO

---

En cada paso, el discriminador es capaz de alcanzar su óptimo,  $D_G^*$ . Esto quiere decir escoger un número de pasos,  $k$ , y, un tamaño de paso adecuado para permitir que se mantenga el teorema del descenso de gradiente.

La última hipótesis es que en cada paso, mejoramos el criterio del generador,  $C(G)$ .

Ahora que hemos comentado las hipótesis, pasemos a la demostración. La dividiremos en dos partes, por un lado probaremos que la función de coste es convexa respecto la distribución del generador, por otro, que el descenso de gradiente en  $D_G^*$  converge a lo mismo que el descenso de gradiente para el óptimo global  $D$ .

#### Función convexa

Una función  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  se dice que es convexa en un conjunto  $S \subseteq \mathbb{R}^n$  si para todos los puntos  $\mathbf{x}, \mathbf{y} \in S$  y para todo  $t \in [0, 1]$ , se verifica que:

$$f(t\mathbf{x} + (1 - t)\mathbf{y}) \leq tf(\mathbf{x}) + (1 - t)f(\mathbf{y}) \quad (3.46)$$

#### Subgradiente y subdiferencial

Un *subgradiente*  $g$  de una función convexa  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  en un punto  $x \in \mathbb{R}^n$  es un vector que satisface la siguiente desigualdad para todo  $y \in \mathbb{R}^n$ :

$$f(y) \geq f(x) + \langle g, (y - x) \rangle.$$

La notación  $\langle g, (y - x) \rangle$  representa el producto escalar entre  $g$  y  $(y - x)$ .

El conjunto de todos los subgradientes en el punto  $x$  se llama *subdiferencial* de  $f$  en  $x$ , y se denota como  $\partial f(x)$ .

*Demostración.* Consideramos  $V(G, D) = U(p_g, D)$ , la función de coste dependiente de la distribución  $p_g$ .

Probemos que es convexa.

Aplicando la definición de función convexa, queremos comprobar que se cumple lo siguiente.

$$U(tp'_g + (1 - t)p_g, D) \leq tU(p'_g, D) + (1 - t)U(p_g, D) \quad (3.47)$$

para cualquier otra distribución  $p'_g$ .

Después de eliminar en ambos lados los términos que no dependen de  $p_g$ ,  $\mathbb{E}_{x \sim p_{data}}[D(x)]$ , tenemos que

$$\mathbb{E}_{x \sim tp'_g + (1 - t)p_g} [\log(1 - D(x))] \leq t\mathbb{E}_{x \sim p'_g} [\log(1 - D(x))] + (1 - t)\mathbb{E}_{x \sim p_g} [\log(1 - D(x))] \quad (3.48)$$

### 3.2 Redes Generadoras Adversarias - GANs

---

Para demostrarlo, vamos a probar una afirmación aún más fuerte, que es

$$\mathbb{E}_{x \sim tp'_g + (1-t)p_g} \equiv t\mathbb{E}_{x \sim p'_g} + (1-t)\mathbb{E}_{x \sim p_g} \quad (3.49)$$

En efecto:

$$\mathbb{E}_{x \sim tp'_g + (1-t)p_g}[f] = \int (tp'_g + (1-t)p_g)f = t \int p'_g f + (1-t) \int p_g f = t\mathbb{E}_{x \sim p'_g}[f] + (1-t)\mathbb{E}_{x \sim p_g}[f] \quad (3.50)$$

lo que demuestra la convexidad de  $U(p_g, D)$  con respecto a  $p_g$ .

Vamos a llamar  $U(p_g) = \sup_D U(p_g, D)$  como la función para el discriminador óptimo.

Dado que  $U(p_g, D)$  es convexa para cada  $D$ , tenemos que  $U(p_g)$  también es convexa. Vamos a probar esta última afirmación

Sea  $\{U(p_g, D_i)\}_{i \in I}$  la familia de funciones convexas. Si fijamos  $i$  se cumple que

$$U(tp_g + (1-t)p_{g2}, D_i) \leq tU(p_{g1}, D_i) + (1-t)U(p_{g2}, D_i)$$

Esto se cumple para cada  $i$ , ahora tomando el supremo en ambos lados.

$$\sup_D U(tp_{g1} + (1-t)p_{g2}, D) \leq t \sup_D U(p_{g1}, D) + (1-t) \sup_D U(p_{g2}, D)$$

$$U(tp_{g1} + (1-t)p_{g2}) \leq tU(p_{g1}) + (1-t)U(p_{g2})$$

Hemos probado la convexidad de  $U(p_g)$

Sigamos con la demostración, ahora vamos a demostrar que cualquier subgradiente de  $U(p_g, D_G^*)$  también es un subgradiente de  $U(p_g)$  cuando  $D_G^* = \arg \sup_D U(p_g, D)$ . De esta manera, el descenso de gradiente en  $U(p_g, D_G^*)$  converge al mismo valor que el descenso de gradiente en  $U(p_g)$ , ya que esta función es convexa y existe el mínimo global.

Entonces, lo que queremos probar es lo siguiente:

$$\partial U(p_g, \arg \sup_D U(p_g, D)) \subseteq \partial \sup_D U(p_g, D)$$

La demostración de ello es una cuestión de usar correctamente las definiciones. Tenemos que  $U(p_g, D_G^*) = U(p_g)$ . Si  $g \in \partial U(p_g, D_G^*)$ , entonces por definición tenemos  $U(p'_g, D_G^*) \geq U(p_g, D_G^*) + \langle g, p'_g - p_g \rangle$  para cualquier otra distribución  $p'_g$ , siendo el último término el producto escalar de funciones. Por definición de supremo tenemos  $U(p'_g) \geq U(p'_g, D_G^*)$ . Juntando todo obtenemos

$$U(p'_g) \geq U(p'_g, D_G^*) \geq U(p_g, D_G^*) + \langle g, p'_g - p_g \rangle = U(p_g) + \langle g, p'_g - p_g \rangle$$

### 3. REDES NEURONALES. UN ENFOQUE MATEMÁTICO

---

Y por lo tanto  $U(p'_g) \geq U(p_g) + \langle g, p'_g - p_g \rangle$  lo que muestra que  $g \in \partial U(p_g)$ .

□

Veamos por que en la realidad no es aplicable esta proposición.

El principal problema, es que, la demostración se basa en que trabajamos con la función ideal  $U(p_g, D)$  que depende de la distribución del generador. En la práctica estamos trabajando con los parámetros  $\theta_g$  que forman parte de la función  $G$ , los cuales son finitos, en contraste con, el teóricamente infinito espacio de las posibles distribuciones  $p_g$ . Esto provoca que perdamos la propiedad de convexidad en la función de coste, además es ampliamente conocido que las redes neuronales profundas tienen funciones de coste no convexas. [18]

Otro problema asociado, es que requerimos hallar un discriminador óptimo en cada etapa para luego aplicar el descenso de gradiente, pero esto genera un problema numérico, ya que un discriminador óptimo producirá un gradiente casi nulo, lo cual, al aplicar el algoritmo de retropropagación para actualizar los parámetros del generador, resultará en un entrenamiento que apenas mejora nuestro generador. En la práctica se suele limitar el aprendizaje del discriminador para que mejore más lentamente que el generador, por ejemplo con un tamaño de paso mucho menor.

A pesar de estos problemas, y la dificultad que supone entrenar los modelos GAN, no son una pérdida de tiempo, la comunidad de investigadores sigue trabajando activamente en mejorar la estabilidad y eficacia de los GANs. Con el tiempo, han surgido variantes como los WGAN, CycleGAN, BigGAN, entre otros, que han abordado muchos de los problemas iniciales. Desde la creación de obras de arte, pasando por la generación de música, hasta aplicaciones en medicina para crear imágenes médicas sintéticas, los GANs se han utilizado en un impresionante abanico de campos.

#### 3.2.2 Métricas y evaluación de GANs

En esta sección expondremos algunas de las formas que se utilizan para evaluar los resultados obtenidos por nuestro modelo GAN. Los dos principales criterios por los que nos guiarímos son, la calidad y la diversidad.

Calidad lo definimos como la semejanza entre las imágenes creadas por el generador y las imágenes reales del conjunto de datos.

Por diversidad entendemos que si nuestro conjunto de imágenes reales esta formado por varias clases, la proporción de imágenes generadas sea equitativa entre todas las clases. Además que dentro de una misma clase no generemos copias de las imágenes de entrenamiento si no que haya variabilidad dentro de la clase.

Estos dos criterios no los medimos con la función de coste  $V(G, D)$ , la cual nos da una medida de la relación entre el discriminador y el generador, que, como hemos visto en los teoremas anteriores, terminaría con una distribución de los resultados similar a los originales. Pero al contrario de lo que ocurre con los modelos clasificadores, donde la propia función de coste nos sirve para evaluar lo buenos o malos que son los resultados que devuelve el modelo, con las GANs es necesario crear métodos de evaluación alternativos. A continuación presentaremos los principales que se usan hoy en día, como veremos ninguno de ellos cubre todas las necesidades y todos tienen problemas asociados que los alejan de ser una forma de evaluación perfecta, sin embargo, son una buena referencia para ver que tal esta resultando el entrenamiento de nuestro modelo.

#### 3.2.2.1 Inception score

En el paper [26] presentan el *Inception Score*, IS, una métrica popular para evaluar la calidad y la diversidad de las imágenes generadas por una GAN. Se basa en un modelo preentrenado de clasificación de imágenes *InceptionV3* entrenada en el extenso conjunto de datos *ImageNet*. El IS se define matemáticamente como

$$IS = \exp(\mathbb{E}_x D_{KL}(p(y|x) || p(y)))$$

donde  $\mathbb{E}_x$  representa la expectativa sobre todas las imágenes generadas  $x$ ,  $D_{KL}$  es la divergencia de *Kullback-Leibler* entre las dos distribuciones,  $p(y|x)$  es la distribución condicional de la clase  $y$  dado el dato generado  $x$ , y  $p(y)$  es la distribución marginal de la clase  $y$  sobre todas las imágenes generadas.

La idea en que se basa el IS es usar el concepto de entropía, en el contexto de la teoría de información, y la probabilidad condicionada  $p(y|x)$ , donde  $y$  son las etiquetas de clase y  $x$  las imágenes generadas. En este caso, buscamos que la entropía de  $p(y|x)$  sea baja, lo que significa que, dada una imagen generada  $x$ , la etiqueta de clase  $y$  debería ser altamente predecible. Esto es un indicador de alta calidad en las imágenes generadas. Por otro lado, queremos una entropía alta asociada a la probabilidad marginal  $p(y)$ , lo cual sugiere que

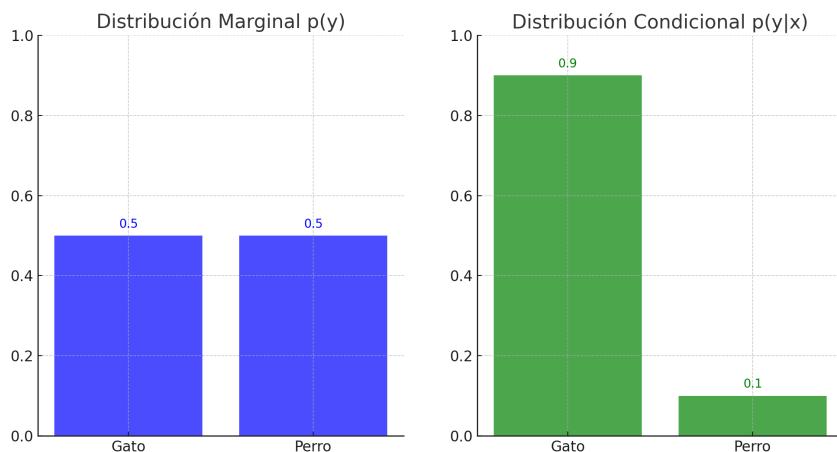
### 3. REDES NEURONALES. UN ENFOQUE MATEMÁTICO

---

el generador esta produciendo una amplia variedad de etiquetas, eso significa una mayor diversidad.

Después utiliza la divergencia de *Kullback-Leibler* para medir la distancia de estas distribuciones, en este caso nos interesa que las distribuciones estén lo más distanciadas entre si. ¿Por qué? Si la distribución condicional  $p(y|x)$  se asemeja demasiado a la marginal  $p(y)$ , que idealmente debería ser uniforme si el modelo es bueno, tendríamos un indicio de que el modelo asigna igual importancia a todas las clases para una etiqueta dada, lo cual es un signo de baja calidad. Por otro lado, si  $p(y)$  se parece mucho a  $p(y|x)$ , tendríamos una señal de falta de diversidad, ya que la distribución estaría concentrada en una región pequeña del espacio de etiquetas.

En la siguiente figura vemos un modelo ideal que genera con la misma frecuencia imágenes de sus etiquetas (gatos y perros), y dado una imagen generada (de gato) la clasifica con alta confianza lo cual indica buena calidad de la imagen.



**Figura 3.5:** Ejemplo de distribuciones. Imagen del autor

La principal limitación de IS es que no es capaz de diferenciar entre diversidad entre clases y diversidad dentro de la clase. Es decir, podemos generar solo una imagen de calidad para cada clase y el IS será alto. Esto es un gran problema pues queremos un generador capaz de crear una gran diversidad de imágenes para cada clase.

#### 3.2.2.2 FID

La *Frechet Inception Distance*, FID, introducida en el artículo [11] es otra métrica ampliamente utilizada para evaluar las imágenes generadas por una GAN, aunque su enfoque es ligeramente distinto al del IS. Mientras que el IS evalúa tanto la calidad como la diversidad de las imágenes generadas únicamente en función de las mismas, el FID las compara directamente con un conjunto de datos real. Para ello, se utiliza también un modelo preentrenado, frecuentemente *InceptionV3*, pero en este caso, se extraen características de las capas internas de la red tanto para las imágenes generadas como para las imágenes reales.

La métrica se calcula mediante la distancia *Fréchet* entre las distribuciones de estas características extraídas. Matemáticamente, se define como:

$$\text{FID}(x, g) = \|\mu_x - \mu_g\|^2 + \text{Tr}(\Sigma_x + \Sigma_g - 2(\Sigma_x \Sigma_g)^{1/2})$$

donde  $\mu_x$  y  $\mu_g$  son las medias de las características extraídas para las imágenes reales  $x$  y generadas  $g$ , respectivamente. Y  $\Sigma_x$  y  $\Sigma_g$  son las matrices de covarianza de las características para  $x$  y  $g$ .

En el artículo, apoyados en que la distribución que maximiza la entropía dados una media y covarianza, es una gaussiana, asumen que las características de  $x$  y  $g$  siguen una distribución gaussiana multivariante.

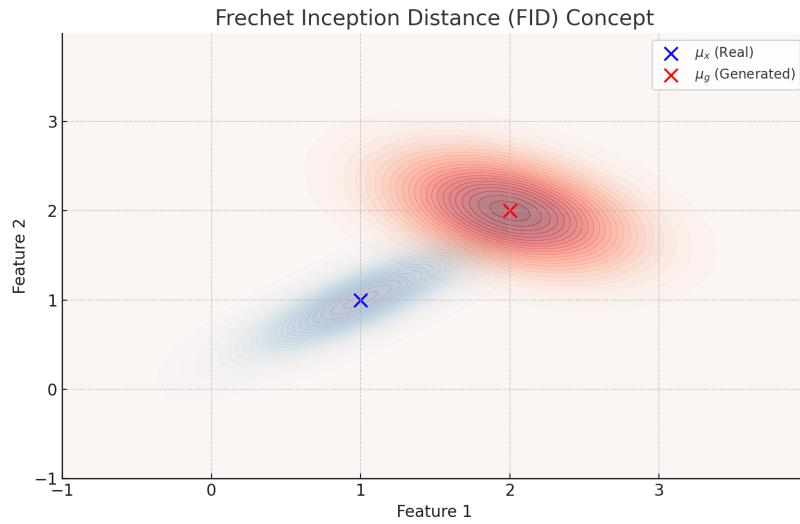
Una puntuación FID baja indica que las dos distribuciones son similares y, por lo tanto, que las imágenes generadas son en calidad similares a las imágenes reales. Esto es especialmente útil cuando se tiene un conjunto de datos objetivo específico en mente.

Es importante señalar que, a diferencia del IS, el FID es una métrica bidireccional, esto quiere decir que tiene en cuenta tanto la calidad de las imágenes generadas como su relación con un conjunto de datos real. Esto le da una visión más completa del rendimiento del modelo, aunque también significa que necesita un conjunto de datos real de referencia para la comparación.

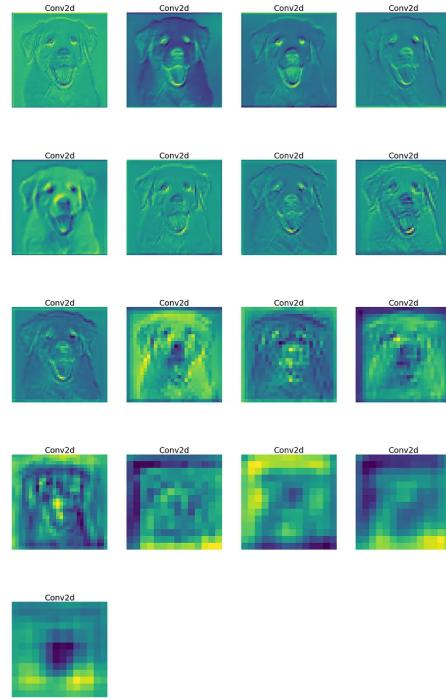
El mapa de características son los vectores de características que se obtienen al pasar imágenes a través de un modelo de red neuronal preentrenado, el *Inception*. Estas representaciones se extraen de ciertas capas internas y encapsulan información abstracta sobre la imagen, como textura, color, formas, etc. En la siguiente figura podemos ver un ejemplo.

### 3. REDES NEURONALES. UN ENFOQUE MATEMÁTICO

---



**Figura 3.6:** Ilustración de la distancia Fréchet entre las distribuciones de las imágenes reales y generadas. Imagen del autor.



**Figura 3.7:** Mapa de características de las distintas capas ocultas. Imagen obtenida de <https://ravivaishnav20.medium.com/visualizing-feature-maps-using-pytorch-12a48cd1e573>.

### **3.2 Redes Generadoras Adversarias - GANs**

---

Entre las limitaciones de FID mencionaremos que:

- Al contrario que el IS, necesita un conjunto de referencia, los datos reales, que por lo general debe ser de gran tamaño
- Depende del modelo preentrenado *Inception V3* para extraer las características.
- Por la razón anterior también es computacionalmente costoso ya que tiene que pasar todas las imágenes por el modelo para extraer las características
- Asume una distribución gaussiana multivariante para las características.
- Al igual que pasa con las otras métricas, carece de una interpretación intuitiva, sabemos que cuanto más bajo mejor pero el número en si mismo no dice cuanto de mejor es la imagen.

#### **3.2.2.3 KID**

El *Kernel Inception Distance*, KID, presentado en el artículo [2] es otra métrica que se utiliza para evaluar la calidad de las imágenes generadas por una GAN. A diferencia del FID que usa una distancia *Fréchet* para comparar distribuciones, KID emplea un test de dos muestras basado en la similitud del kernel entre las características extraídas.

KID también emplea un modelo preentrenado, típicamente *InceptionV3*, para extraer características de las capas internas de la red para ambas imágenes generadas y reales. Sin embargo, en lugar de calcular directamente la distancia entre las medias y las matrices de covarianza, para calcular el KID, se utiliza un kernel polinómico definido como:

$$k(x, y) = \left(\frac{1}{d}x^T y + 1\right)^3$$

donde  $d$  es la dimensión de la representación, es decir el número de elementos que forman los vectores características,  $x$  e  $y$ , asociados a las imágenes reales y generadas respectivamente.

El KID se define como el cuadrado de la *Maximum Mean Discrepancy*, MMD, de las características de las imágenes reales  $x$  y generadas  $y$ .

$$\text{MMD}^2(\mathcal{X}, \mathcal{Y}) = \frac{1}{n^2} \sum_{i=1}^n \sum_{j=1}^n k(x_i, x_j) + \frac{1}{m^2} \sum_{i=1}^m \sum_{j=1}^m k(y_i, y_j) - \frac{2}{nm} \sum_{i=1}^n \sum_{j=1}^m k(x_i, y_j)$$

### **3. REDES NEURONALES. UN ENFOQUE MATEMÁTICO**

---

donde  $\mathcal{X}, \mathcal{Y}$  son los conjuntos que contienen las muestras  $x_i, y_i$  respectivamente.

Una puntuación KID baja indica que las dos distribuciones son similares, lo cual es un buen indicador de que las imágenes generadas son de alta calidad. Sin embargo, a diferencia del FID, no es necesario que el conjunto de datos de referencia sea grande, lo que hace que KID sea más flexible y menos costoso computacionalmente.

Aquí hay algunas limitaciones de esta métrica:

- Al igual que el FID, depende de un modelo preentrenado para extraer características.
- Aunque es menos costoso computacionalmente que el FID, aún requiere un tiempo considerable para calcular las similitudes del kernel.
- Similar al FID y al IS, carece de una interpretación intuitiva.

Para concluir, **utilizaremos FID y KID para la evaluación de nuestro modelo GAN** siguiendo las recomendaciones del artículo de *StyleGAN2-ADA* [16].

CAPÍTULO  
**4**

## **Resultados y discusión**

Nuestro proyecto comenzó con un problema, la escasez de datos en el ámbito de neuroimagen neonatal que limita el posible desarrollo de herramientas para el análisis automático de las imágenes basadas en algoritmos de aprendizaje automático. Nuestra hipótesis es que podemos generar imágenes artificiales de suficiente calidad y diversidad para complementar un conjunto de entrenamiento pequeño de imágenes reales para poder entrenar un modelo clasificador de planos cerebrales, y obtener métricas similares e incluso mejores que con un conjunto grande de datos reales.

En este capítulo vamos a exponer los resultados que hemos obtenido de nuestros experimentos aplicando las herramientas y metodologías explicadas en capítulos anteriores. Comenzaremos explicando el proceso de generación del conjunto de datos formado por imágenes 2D de planos cerebrales neonatales de ultrasonidos que se usarán para el entrenamiento de la GAN, continuaremos con la descripción del procesos de generación de imágenes artificiales, y terminaremos describiendo los resultados del entrenamiento semi supervisado por reemplazamiento de datos reales con datos sintéticos en el entrenamiento de modelos clasificadores de imágenes de planos cerebrales. Además, discutiremos los resultados y sus consecuencias, comentaremos las problemas que presenta nuestro enfoque y las futuras investigaciones que se pueden llevar a cabo.

## 4. RESULTADOS Y DISCUSIÓN

---

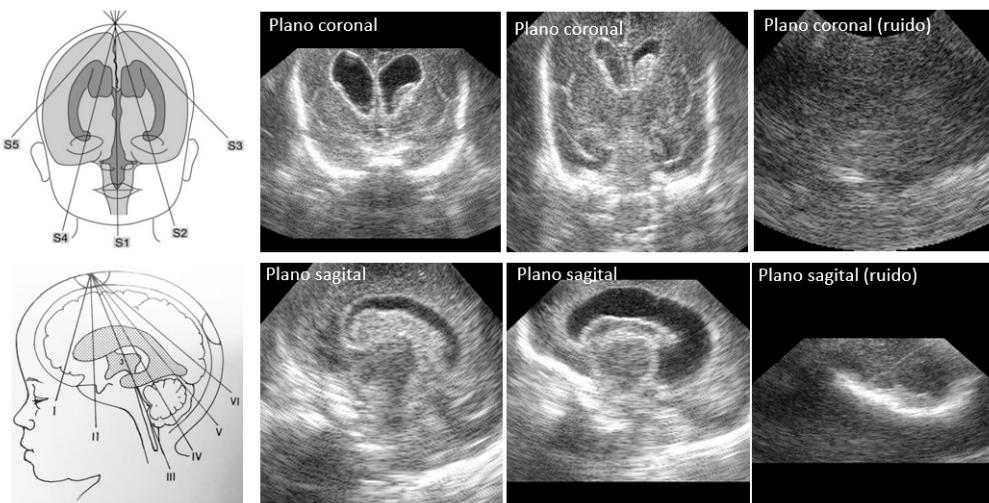
### 4.1 Creación del conjunto de imágenes de entrenamiento

El objetivo final de nuestro proyecto es entrenar una red neuronal para clasificar planos cerebrales de ultrasonidos. Para ello, es necesario disponer de un conjunto de imágenes representativo de las clases de imágenes que queremos ser capaces de clasificar.

Durante este proceso hemos hecho una simplificación. Lo ideal sería que nuestro modelo de machine learning aprendiera a diferenciar y clasificar distintos tipos de planos sagitales y coronales con idea de detectar PHVD (un crecimiento excesivo de los ventrículos laterales). Es decir, sería un clasificador para distinguir entre imágenes anómalas y no anómalas. Sin embargo, en nuestro experimento lo que hemos estudiado es el problema de diferenciar si la imagen es un corte (plano) sagital, coronal, o si por el contrario, es mayoritariamente una imagen sin información o ruido.

La decisión de incluir una categoría de ruido reside en que al realizar la extracción de imágenes observamos que en todos los volúmenes las primeras y las últimas imágenes en cada vista contienen poca o ninguna información. Si nuestro modelo es entrenado con tres categorías será capaz de diferenciar el ruido de los planos con información útil.

En la figura 4.1 se muestran esquemáticamente y también con imágenes reales los distintos tipos de planos cerebrales comentados más arriba, y que son de interés para este trabajo.



**Figura 4.1:** En la primera columna de la izquierda, esquema de planos cerebrales observables a través de la fontanela de un recién nacido. Imagen extraída de [20] y [15]. En la fila inferior ejemplos de imágenes reales de planos sagitales.

## 4.1 Creación del conjunto de imágenes de entrenamiento

---

### 4.1.1 Aumento de datos mediante extracción de planos oblícos

Para simular las condiciones de disponer de pocos datos hemos trabajado solo con 4 ecografías 3D (volúmenes en formato nrrd). Las imágenes las extraemos con un programa de software realizado *ex profeso* que hemos denominado *extraccion.py*. Las dimensiones de uno de estos volúmenes son aproximadamente  $[0, 200] \times [0, 200] \times [0, 200]$  de modo que las imágenes extraídas tendrán las dimensiones  $200 \times 200$  píxeles. Si generamos el conjunto de entrenamiento usando solo las imágenes ortogonales obtendríamos aproximadamente 200 imágenes sagitales, 200 coronales, y de estas unas 80 imágenes (20 al principio y final del rango sagital, y lo mismo para el coronal) con ruido por cada volúmen nrrd. Si disponemos de 3 volúmenes serían aproximadamente 1921 imágenes de entrenamiento (891 imágenes sagitales, 790 coronales y 240 con ruido). Así, nos quedaría un conjunto de datos o *dataset* con un número muy pequeño de imágenes para cada clase. Un conjunto de datos pequeño es más probable que lleve a problemas de sobreajuste (o *overfitting* en inglés) durante el entrenamiento<sup>1</sup>. Aunque la arquitectura GAN empleada en este proyecto es especialmente efectiva para entrenar con conjuntos de datos pequeños, el número de imágenes del que disponemos es de un tamaño demasiado pequeño para entrenarla ya que disponemos de unas 1921 imágenes para entrenar, una cantidad que es mucho menor que las 5-10 kimg que idealmente se necesitarían [16].

Para evitar el sobreajuste en *deep learning* es común hacer lo que se llama ***data augmentation*** o aumento de datos. Consiste en aumentar la cantidad de datos de entrenamiento disponibles para un modelo de aprendizaje automático ya sea con o sin recopilar nuevos datos reales. Si no se puede aumentar el conjunto con datos reales, el método más común es aplicar transformaciones y manipulaciones diversas que pueden incluir:

1. Rotaciones: Rotar una imagen en diferentes ángulos.
2. Volteos horizontales o verticales: Reflejar una imagen horizontal o verticalmente.
3. Traslaciones: Mover una imagen ligeramente en diferentes direcciones.
4. Cambios en el brillo y el contraste: Ajustar el brillo y el contraste de una imagen.

---

<sup>1</sup>El sobreajuste sucede cuando nuestro modelo no aprende a generalizar. Generalizar en el contexto del aprendizaje automático se refiere a la capacidad de un modelo de machine learning para aplicar el conocimiento adquirido durante su entrenamiento a nuevos datos o ejemplos que no ha visto antes. En otras palabras, un modelo que generaliza bien es capaz de hacer predicciones precisas y útiles sobre datos que no formaron parte de su conjunto de entrenamiento.

## 4. RESULTADOS Y DISCUSIÓN

---

5. Recorte aleatorio: Recortar una parte aleatoria de una imagen original.
6. Adición de ruido: Agregar ruido aleatorio a una imagen.
7. Modificaciones de color: Cambiar la tonalidad, saturación o intensidad de color en una imagen.

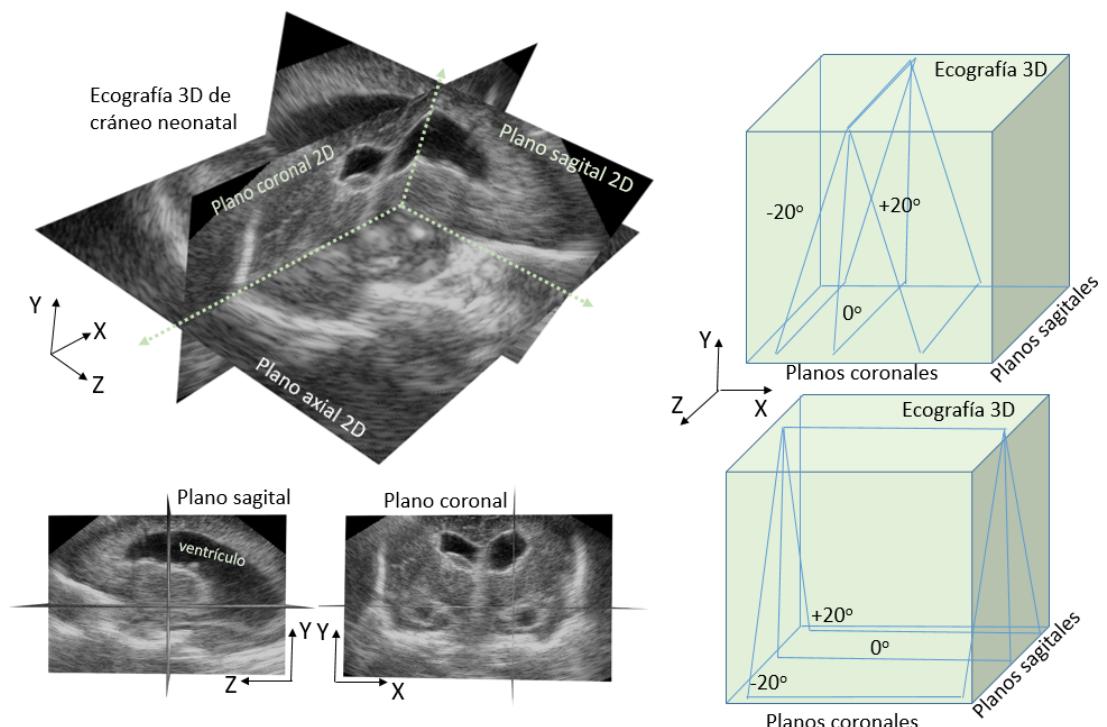
Nosotros queremos aumentar los datos de entrenamiento de una GAN que a su vez se empleará para aumentar el conjunto de entrenamiento de un clasificador. Para ello, **hemos explorado una técnica nueva de aumento de datos** basándonos en que disponemos de datos 3D. Al tener volúmenes, podemos **no sólo extraer imágenes ortogonales sino también imágenes oblicuas** lo que amplia enormemente el número potencial de imágenes contenidas en cada volumen. Más concretamente, hemos extraído a partir de cada plano (sagital o coronal) nuevas imágenes rotando el volumen de  $5^\circ$  en  $5^\circ$ , y extrayendo el plano oblicuo correspondiente. Procedemos así hasta llegar a los  $+20^\circ$  partiendo desde una inclinación de  $-20^\circ$  grados. De este modo obtenemos un conjunto de datos que es 8 veces más grande ( $-20, -15, -10, -5, 5, 10, 15, 20$ ) por cada archivo 3D.

¿Por qué extraemos las imágenes en ese rango, hasta los  $\pm 20^\circ$ , y en pasos de  $5^\circ$ ? No hay nada especial en esos números, podríamos haber extraído planos cada grado de rotación y así obtener un conjunto de datos 40 veces más grande para cada archivo nrrd. Fue una decisión tomada para contrarrestar en el futuro un posible problema de *overfitting*. Y la razón de llegar hasta los 20 grados es por fidelidad con el tipo de imágenes que extraen los neonatólogos, no tendría sentido sacar planos con rotaciones que no se dan en la realidad y entrenar a los modelos de aprendizaje automático con ellos.

Tras extraer todos los planos, hemos procesado las imágenes cómo describimos en la sección 2.1.1.2, redimensionadas a un tamaño de  $256 \times 256 \times 3$  píxeles.

**Al finalizar el procedimiento de extracción de planos de los volúmenes 3D (con aumento de datos mediante extracción oblícua) hemos obtenido un conjunto de imágenes 9 veces mayor que el original.** Más concretamente, el conjunto de entrenamiento disponible para la GAN es de 15536 imágenes en total, de las cuales 6318 son planos coronales, 7128 son del plano sagital y 2088 son clasificadas como ruido.

#### 4.1 Creación del conjunto de imágenes de entrenamiento



**Figura 4.2:** Creación del conjunto de datos por extracción de planos coronales y sagitales 2D a partir de ecografías 3D. Hemos realizado un aumento de datos novedoso consistente en extraer para cada plano ortogonal del volumen ( $0^\circ$  en la figura), otros planos inclinados en el rango  $[-20^\circ, +20^\circ]$  en pasos de  $5^\circ$ .

## 4. RESULTADOS Y DISCUSIÓN

---

### 4.2 Entrenamiento del modelo GAN

Como hemos mencionado uno de las principales formas de evitar el sobreajuste a los datos de entrenamiento es mediante el aumento de datos. El acrónimo ADA que se incluye en el nombre de la arquitectura GAN empleada en este proyecto, *StyleGAN2-ADA* [16], se refiere a que incluye un preprocesado para el aumento de datos que los autores denominan *adaptive discriminator augmentation* o por sus iniciales, ADA. Esta GAN es capaz de autojustarse, con una probabilidad  $p$  de aplicar una transformación o no a las imágenes de entrada, en función del nivel de sobreajuste que detecte durante el entrenamiento. Para ello emplea dos métricas

$$r_v = \frac{\mathbb{E}[D_{\text{train}}] - \mathbb{E}[D_{\text{validation}}]}{\mathbb{E}[D_{\text{train}}] - \mathbb{E}[D_{\text{generated}}]} \quad \text{y} \quad r_t = \mathbb{E}[\text{sign}(D_{\text{train}})]$$

El valor  $r_t$  indica si el discriminador da alta probabilidad a la mayoría del conjunto de entrenamiento, lo cual podría ser un signo de sobreajuste, y  $r_v$  expresa la relación entre la salida del conjunto de entrenamiento con las salidas de los conjuntos de validación y generados. En ambos casos, un valor de 1 indica un completo sobreajuste. El entrenamiento comienza con un valor de  $p = 0$  y comienza ha aumentar o disminuir según los valores de  $r_t$  y  $r_v$ .

#### 4.2.1 Parámetros de entrenamiento de la GAN

La GAN empleada en este proyecto puede ser entrenada seleccionando distintos parámetros que enumeramos a continuación

- *snap* es un número que indica la frecuencia con la que se guarda el estado del modelo en ese momento y se calculan las métricas. Esto es útil para ver si estamos mejorando y por si nos interesa volver a alguna versión anterior del modelo.
- *augpie* establece los tipos de transformaciones que queremos aplicar para aumentar los datos.
- *metrics*, que es para elegir las métricas de evaluación de nuestro modelo.
- El valor de *gamma*, es una forma de regularización y evitar el colapso de modo.
- El parámetro *mirror* se pone a *True* si queremos utilizar las imágenes espejo de las originales para aumentar así el conjunto de entrenamiento.

- Y por último *resume*, que sirve para que nuestro entrenamiento comience a partir de un modelo preentrenado. Esto es muy útil para no partir de cero, además de poder retomar el mismo entrenamiento de un modelo guardado anteriormente con *snap*.

Los parámetro empleados han sido con los siguientes parámetros: *snap* = 100 ciclos de 400 kimg; *augpie* = bp que aplicar las transformaciones *blit* y geométricos, los cuales esta probado que tienen mejor rendimiento que el estándar *bdc* en el ámbito de imágenes con escala de grises; *metrics* = *kid50\_full*, *kid50\_full*; *gamma* = 25; *mirror* = True; y *resume* = *ffhq256*, que indica que hacemos *fine tuning*, es decir que partimos partimos del modelo *ffhq256*, que ha sido preentrenado con caras humanas de  $256 \times 256$  píxeles.

El modelo fue entrenado en los servidores de *paperspace*, con una GPU P-500, entrenada con ciclos de 400 kimg hasta un total de 1200 kimg. Por cuestiones tiempo y coste computacional no hemos seguido con el entrenamiento. En el artículo de referencia [16] introduce la arquitectura *styleGAN2-ADA* se recomienda entrenar hasta 5000 kimg (5.000.000 de imágenes), pero si se usa *fine tuning*, como hemos hecho aquí, a partir de 1000 kimg deberían empezar a observarse buenos resultados.

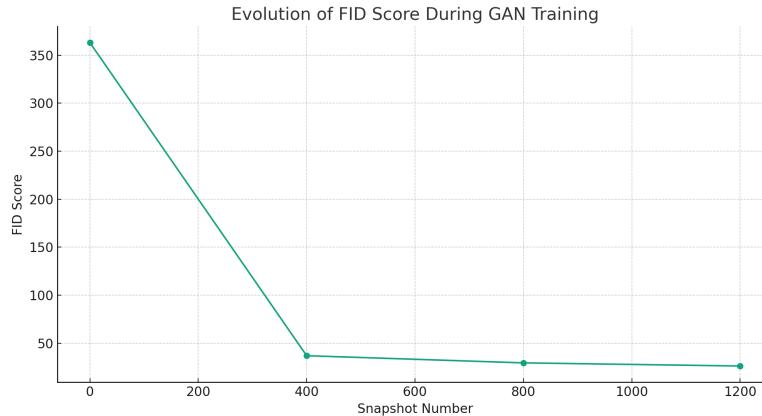
### 4.2.2 Evaluación del modelo

Las métricas usadas en el artículo [16] donde se presenta la arquitectura *styleGAN2-ADA* son *kid* y *fid*, estas son las métricas estándar y recomendadas para evaluar los resultados, en especial *kid*, ya que nuestro conjunto de datos original no era demasiado grande. Como hemos visto tienen la complicación de que son difíciles de interpretar, si bien indican que el generador va por buen camino, esta aun lejos de su óptimo resultado, al final la última palabra la debería tener un experto en la materia, que opinara sobre la verosimilitud de las imágenes generadas.

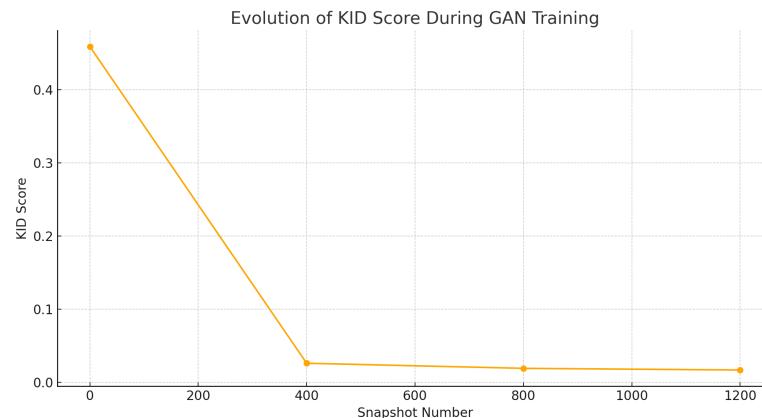
Las figuras 4.3 y 4.4 muestran la evolución de las métricas *fid* y *kid* durante el entrenamiento de la GAN. **Se observa que convergen asintóticamente a unos valores de *fid* = 26,158 y *kid* = 0,017.**

## 4. RESULTADOS Y DISCUSIÓN

---



**Figura 4.3:** Evolución del valor de *fid* durante la fase de entrenamiento de la GAN. Imagen del autor



**Figura 4.4:** Evolución del valor de *kid* durante la fase de entrenamiento de la GAN. Imagen del autor

Para finalizar, generamos un conjunto de 16000 imágenes sintéticas , empleando la GAN, y variando un parámetro de generación del modelo que se llama **truncamiento**<sup>1</sup>. Las figuras 4.5 y 4.6 son una muestra de las imágenes artificiales que genera la GAN una

<sup>1</sup>El truncamiento es un parámetro que permite ajustar la diversidad y la calidad de las imágenes generadas con la GAN. Si se emplea un valor bajo ( $\psi$  pequeño), el modelo tiende a generar imágenes que son más 'promedio' de la distribución de imágenes empleadas en el entrenamiento. Esto puede aumentar la calidad y realismo de las imágenes, pero reduce su diversidad. Si se emplea un valor alto de truncamiento ( $\psi$  grande) se permite que el modelo explore más opciones dentro del espacio latente y genere imágenes más diversas, pero potencialmente menos realistas o con mayores anomalías.

### 4.3 Entrenamiento de clasificadores

---

vez hemos finalizado el entrenamiento. Y en el Apéndice A hay varias muestras de las imágenes generadas con la GAN y para distintos valores de truncamiento.



**Figura 4.5:** Conjunto de 100 imágenes sintéticas generadas con la GAN para un valor de truncamiento 0,5. Imagen del autor

### 4.3 Entrenamiento de clasificadores

La métrica usada para evaluar los resultados de distintos modelos ha sido la exactitud o *accuracy* definida como

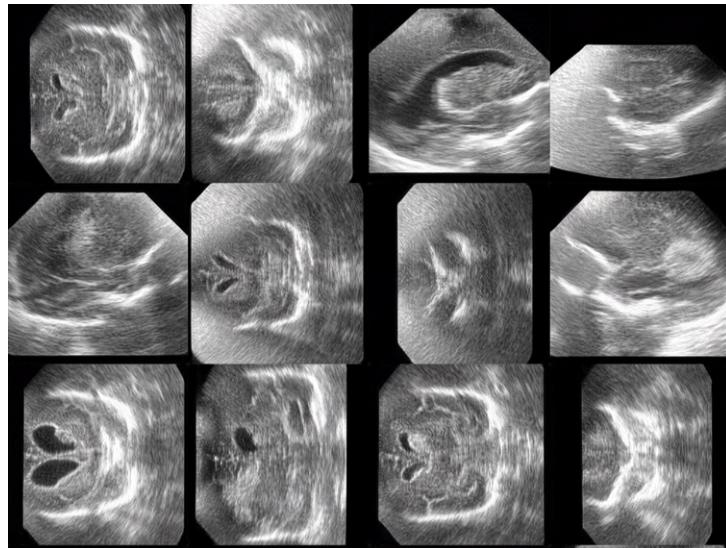
$$\text{Accuracy} = \frac{\text{VP} + \text{VN}}{\text{VP} + \text{VN} + \text{FP} + \text{FN}}$$

donde VP son los Verdaderos Positivos, VN los Verdaderos Negativos, FP los Falsos Positivos y FN los Falsos Negativos.

En nuestro experimento, entrenamos una red neuronal con un conjunto de entrenamiento formado por el 100 % de las imágenes reales. A continuación, repetimos el entrenamien-

## 4. RESULTADOS Y DISCUSIÓN

---



**Figura 4.6:** Detalle de imágenes sintéticas obtenidas. Se observa, al menos cualitativamente, el nivel de realismo que se obtiene con el procedimiento seguido en este proyecto. Imagen del autor

to de este clasificador pero utilizando solo un porcentaje del conjunto original y aplicaremos el entrenamiento semi supervisado con las imágenes generadas sintéticamente, que en nuestros experimentos alcanzan las 16000 imágenes.

%X de datos reales	Truncamiento $\psi$	Métrica	Valor
100 %	-	Accuracy	<b>0.9456</b>
70 %	0.8	Accuracy	0.9387
70 %	0.9	Accuracy	0.9404
70 %	1.0	Accuracy	0.9345
50 %	0.8	Accuracy	0.9378
50 %	1.3	Accuracy	0.9291
50 %	0.9	Accuracy	0.9365
30 %	1.0	Accuracy	0.9300
30 %	0.8	Accuracy	<b>0.9322</b>
30 %	0.5	Accuracy	0.9316

**Cuadro 4.1:** Comparativa de los valores de *exactitud* o *accuracy* de diferentes modelos de clasificadores obtenidos mediante aprendizaje semi supervisado con distintos valores de truncamiento  $\psi$  y del porcentaje  $X$  de imágenes reales empleadas como conjunto de entrenamiento.

La tabla 4.1 muestra la comparativa de la *accuracy* de distintos modelos de clasificador

#### **4.4 Descripción del código software desarrollado en el proyecto**

---

en función del porcentaje  $X$  de datos reales empleado para el entrenamiento y con distintos valores de truncamiento usado para generar el subconjunto de datos sintéticos.

#### **4.4 Descripción del código software desarrollado en el proyecto**

Todas las herramientas, algoritmos y datos empleados en este trabajo están accesibles de manera abierta en el siguiente repositorio: <https://github.com/isakez/TFG>.

Enumeramos a continuación, junto a una breve descripción, los algoritmos desarrollados como Jupyter Notebooks (formato \*.ipynb) y empleando el marco de trabajo *Pytorch*:

1. *Extraccion.ipynb*. Con este código extraemos los planos sagitales y coronales a partir de los volúmenes nrrd. Además los procesamos para convertirlos en RGB y reescalamos las imágenes para que tengan el mismo tamaño de 256x256x3 píxeles. Después de ejecutar el código obtenemos el conjunto de datos de entrenamiento en distintas carpetas según su etiqueta
2. *Clasificador\_Imagenes.ipynb*. Notebook donde exploramos e iteramos varias veces para entrenar un modelo clasificador basado en el conjunto etiquetado que hemos obtenido anteriormente. Al final de este notebook exportamos el clasificador base.
3. *StyleGAN2\_Model.ipynb*. Aquí hacemos uso del repositorio de *styleGAN2-ada* para entrenar nuestro modelo, en este caso hasta las 1200 kimg. Una vez entrenado usamos el mismo *notebook* para generar las imágenes sintéticas con distintos valores de truncamiento. Las imágenes aquí generadas formaran el conjunto de datos para el entrenamiento semi-supervisado.
4. *Entrenamiento\_Semisupervisado.ipynb*. En este *notebook* iteramos de la siguiente forma, entrenamos un clasificador base con solo un porcentaje (30, 50 y 70) del conjunto etiquetado original, aplicamos las pseudoetiquetas para un umbral especificado previamente, añadimos esas imágenes al conjunto de datos anterior y volvemos a entrenar el clasificador esperando mejorar nuestros valores de accuracy y loss. Este proceso lo podemos iterar hasta que el conjunto de datos artificiales esté completamente pseudo etiquetado o obtengamos mejoras en el rendimiento de clasificador.
5. *Probar\_Modo.ipynb*. Aquí simplemente podemos probar distintos modelos clasificadores con un conjunto de datos que no se ha usado durante su entrenamiento.

## 4. RESULTADOS Y DISCUSIÓN

---

Como ejemplo, con las siguientes líneas de código podemos:

1. Preparar el conjunto de datos reales

```
!python \textit{dataset}_tool.py --source /notebooks/Conjunto_Entrenamiento/
--dest /notebooks/conjunto_entrenamiento/Conjunto_Entrenamiento.zip
```

2. Entrenar la GAN

```
!python train.py --outdir resultados_nuevos --data /notebooks/
conjunto_entrenamiento/Conjunto_Entrenamiento.zip --gpus=1 --snap=100
--augpipe=bg --metrics=fid50k_full,kid50k_full --gamma=25 --mirror=True
--resume=ffhq256
```

3. Generar imágenes sintéticas

```
!python generate.py --outdir=imagenes_generadas_trunc15 --trunc=1.0 --seeds
=0-16000 --network=modelo_gan_V0.pkl
```

Más detalle de cómo hemos implementado los algoritmos puede obtenerse en el propio código en forma de líneas de comentarios.

### 4.5 Discusión de resultados

En este proyecto hemos sido capaces de extraer un conjunto de datos de imágenes 2D de ultrasonidos con variedad y calidad suficientes para poder entrenar una red generativa GAN. Para ello hemos desarrollado un método de aumento de datos novedoso consistente en extraer imágenes oblicuas 2D a partir de volúmenes obtenidos con un ecógrafo 3D, multiplicando el número de datos x 8, hasta las 15536 imágenes.

Con esos datos hemos reentrenado una GAN no condicionada de última generación con arquitectura *StyleGAN* [16]. Estos modelos son capaces de generar imágenes de alta resolución muy realistas de rostros humanos y otros objetos, pero hasta donde sabemos, esta es la primera vez que las redes *StyleGAN2* se utilizan como método de sintetizar imágenes de ultrasonido, de gran similitud con las imágenes reales, siendo en muchos casos indistinguibles para un ojo no experto. Este resultado está acorde con las buenas métricas de *kid* y *fid* obtenidas.

Hemos demostrado que las imágenes artificiales generadas por estos modelos pueden beneficiar a los clasificadores supervisados de aprendizaje profundo. Y hemos demostrado que se puede obtener una exactitud similar con menos ejemplos reales (experimentos de

## **4.5 Discusión de resultados**

---

reemplazo o entrenamiento semi supervisado). Ninguno de los clasificadores entrenados con el semi supervisado ha logrado batir la marca del modelo base de 0,9456. Esto puede indicar una falta de diversidad y generalización del modelo GAN que hemos entrenado, incapaz de generar imágenes diferentes pero realistas más allá del conjunto con el que ha sido entrenado. Sin embargo, cabe destacar que la diferencia es del 1 % en la exactitud (0.93 en lugar de 0.94) y esto reduciendo el conjunto de datos etiquetados reales necesarios hasta un 70 % (es decir 4660 en lugar de 15536 imágenes).

Una primera dificultad que hemos enfrentado es la recolección y limpieza de datos. Las decisiones tomadas de cómo extraer los planos 2D de los archivos 3D provoca que conjuntos de imágenes donde solo debería haber imágenes de planos sagitales y coronales tengan también muestras de ruido. Esto se puede solucionar limpiando manualmente el conjunto de datos. Además la decisión de cuantas rotaciones aplicar para la extracción de planos oblicuos, y, por tanto cuantas extracciones realizar, afecta a la diversidad de los datos como ya hemos explicado.

La segunda dificultad que hemos encontrado es evitar el colapso del modo, un problema frecuente en el entrenamiento de redes GAN. En el primer intento de entrenar nuestro modelo generativo tuvimos buenos resultados de las métricas *kid* y *fid*, sin embargo al generar las imágenes artificiales para usarlas en el aprendizaje semi supervisado observamos que el 90 % de las pseudo-etiquetas correspondían al plano coronal provocando un rendimiento muy malo del modelo clasificador semi supervisado al ponerlo a prueba con un conjunto nuevo. Este se puede solucionar aumentando los niveles de regularización o manteniendo un equilibrio entre las cantidades de datos para cada categoría.



CAPÍTULO

# 5

## Conclusiones y Trabajo futuro

Las dificultades de adquirir, procesar y distribuir las  $\sim 10^5 - 10^6$  imágenes necesarias para entrenar una GAN moderna de alta calidad y alta resolución es una empresa costosa. Esto limita el creciente uso de modelos generativos en campos como la medicina donde la escasez de datos impera por motivos prácticos y de confidencialidad.

Nuestro objetivo con este trabajo era probar la viabilidad de un clasificador de neuroimagen neonatal entrenado con un conjunto pequeño de datos reales y complementarlo con datos generados artificialmente para alcanzar un rendimiento similar o superior. **Los resultados obtenidos han sido óptimos ya que podemos conseguir resultados similares con un conjunto imágenes reales etiquetadas un 30 % menor.** Además, con el uso de entrenamiento semi supervisado hemos demostrado que estos modelos se pueden beneficiar de usar conjuntos de datos reales sin etiquetar para complementar su entrenamiento.

Una reducción significativa en el número de imágenes reales requeridas para entrenar modelos de IA para neuroimagen neonatal con ecografía como se ha demostrado es extrapolable a otras aplicaciones en el ámbito de la medicina. Cabe recalcar que el aumento de datos mediante imágenes sintéticas no es un sustituto de los datos reales; siempre se debe intentar recopilar primero un conjunto de datos de entrenamiento grande y de alta calidad, y luego, solo después, completar las lagunas mediante la ampliación. Aquí hemos demostrado que con sólo 3 ecografías 3D es posible obtener resultados prometedores. Nos preguntamos qué se podría lograr, teniendo en cuenta que lo neonatólogos del Hospital

## **5. CONCLUSIONES Y TRABAJO FUTURO**

---

Puerta del Mar disponen de cientos de ecografías 3D de muchísimos neonatos tomadas durante años.

Hay muchas maneras de expandir este trabajo y de experimentar para buscar mejores resultados, pero mencionaremos sólo dos.

Primera, una posibilidad es la de entrenar una red para clasificar de forma automática distintos planos del cerebro del bebé que sean de interés médico o, de manera más avanzada, identificar posibles anomalías o patologías en la imagen. Y se pueden explorar distintos enfoques de entrenamiento; por ejemplo, utilizar una GAN condicionada que tiene en cuenta las etiquetas de los planos; o bien, entrenar dos GANs distintas una para cada plano, coronal o sagital, así nos aseguramos evitar el colapso de modo.

Segunda, desde el punto de vista matemático, una mejora futura interesante podría ser investigar algún tipo de control de calidad sobre las imágenes generadas por la red con una métrica específica que podría utilizarse para filtrar automáticamente imágenes de mala calidad e incrementar el rendimiento general del clasificador. Se pueden modelar matemáticamente los cambios en las métricas en función de los parámetros involucrados en el proceso de extracción de las imágenes, ya sea variando en el rango de los ángulos, el fragmento de cortes que categorizamos como ruido, el intervalo entre ángulos y entre cortes. Todos estos cambios pueden tener consecuencias en el rendimiento final del modelo clasificador y en la GAN.

# Bibliografía

- [1] Balazs Acs, Mattias Rantalainen, and Johan Hartman. Artificial intelligence as the next step towards precision pathology. *Journal of Internal Medicine*, 288(1):62–81, 2020. 3
- [2] Mikołaj Bińkowski, Danica J. Sutherland, Michael Arbel, and Arthur Gretton. Demystifying mmd gans, 2021. 51
- [3] Margaretha J Brouwer, Linda S De Vries, Lou Pistorius, Karin J Rademaker, Floris Groenendaal, and Manon JNL Benders. Ultrasound Measurements of the Lateral Ventricles in Neonates: Why, How and When? A Systematic Review. *Acta paediatrica*, 99(9):1298–1306, 2010. 4
- [4] J. Copeland. *Inteligencia artificial: una introducción filosofica*. Alianza Universidad. Alianza Editorial, 1996. 1
- [5] Andre Esteva, Brett Kuprel, Roberto A Novoa, et al. Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, 542(7639):115–118, 2017. 2
- [6] Lionel C Gontard, Joaquín Pizarro, Borja Sanz-Peña, et al. Automatic Segmentation of Ventricular Volume by 3D Ultrasonography in Post Haemorrhagic Ventricular Dilatation Among Preterm Infants. *Scientific Reports*, 11, 01 2021. 4, 5, 6
- [7] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014. 3, 35, 42
- [8] Andrzej Grzybowski, Piotr Brona, Glen Lim, et al. Artificial intelligence for diabetic retinopathy screening: a review. *Eye*, 34(3):451–460, 2020. 2

## BIBLIOGRAFÍA

---

- [9] Yuxuan Gu, Qixin Chen, Kai Liu, Le Xie, and Chongqing Kang. Gan-based model for residential load generation considering typical consumption patterns. 11 2018. 35
- [10] Varun Gulshan, Lily Peng, Marc Coram, Martin C Stumpe, Derek Wu, Anand Narayanaswamy, Subhashini Venugopalan, Kasumi Widner, Tom Madams, Jorge Cuadros, Ramasamy Kim, Rajiv Raman, Philip C Nelson, Jessica L Mega, and Dale R Webster. Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs. *JAMA*, 316(22):2402–2410, 12 2016. 2
- [11] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium, 2018. 49
- [12] Wei Hu. Towards a real quantum neuron. *Natural Science*, 10:99–109, 01 2018. 22, 23
- [13] Pilar Ibarrola Muñoz, Leandro Pardo Llorente, and Vicente Quesada Paloma. *Teoría de la probabilidad*. Síntesis, 1997. 38
- [14] L.L. Iglesias and M.C. Cano. *Inteligencia artificial y medicina*. Los Libros de La Catarata, 2023. 1
- [15] Anitha C. James. Practical guide to neonatal cranial ultrasound (crus): basics. *Pae-diatrics and Child Health*, 28(9):424–430, 2018. 54
- [16] Tero Karras, Miika Aittala, Janne Hellsten, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Training generative adversarial networks with limited data, 2020. 14, 52, 55, 58, 59, 64
- [17] John D Kelleher and Brendan Tierney. *Data science*. MIT Press, 2018. 9
- [18] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets, 2018. 46
- [19] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s, 2022. 14
- [20] Vijetha Maller and Harris Cohen. Neurosonography: Assessing the premature infant. *Pediatric Radiology*, 47:1031–1045, 08 2017. 54

---

## BIBLIOGRAFÍA

- [21] Morgan P McBee, Omer A Awan, Andrew T Colucci, et al. Deep learning in radiology. *Academic Radiology*, 25(11):1472–1480, 2018. 3
- [22] Alberto Montero Agudo. Generative adversarial networks based data augmentation for ultrasound fetal brain planes classification, 1 2021. 9
- [23] Eric Ohuma, Ann-Beth Moller, and Elizabeth Bradley. National, regional, and worldwide estimates of preterm birth in 2020, with trends from 2010: a systematic analysis. *The Lancet*, 2023. En prensa. 3
- [24] Jamie Perin, Aaron Mulick, Dan Yeung, et al. Global, regional, and national causes of under-5 mortality in 2000-19: an updated systematic analysis with implications for the sustainable development goals. *The Lancet Child & Adolescent Health*, 6(2):106–115, 2022. 3
- [25] A. Rosebrock. *Deep Learning for Computer Vision with Python: Starter Bundle*. PyImageSearch, 2017.
- [26] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans, 2016. 47
- [27] Toan Pham Van, Tam Minh Nguyen, Ngoc N Tran, Hoai Viet Nguyen, Linh Bao Doan, Huy Quang Dao, and Thanh Ta Minh. Interpreting the latent space of generative adversarial networks using supervised learning. In *2020 International Conference on Advanced Computing and Applications (ACOMP)*, pages 49–54. IEEE, 2020.
- [28] Shawn Xu, Lin Yang, Christopher Kelly, Marcin Sieniek, Timo Kohlberger, Martin Ma, Wei-Hung Weng, Atilla Kiraly, Sahar Kazemzadeh, Zakkai Melamed, Jungyeon Park, Patricia Strachan, Yun Liu, Chuck Lau, Preeti Singh, Christina Chen, Mozziyar Etemadi, Sreenivasa Raju Kalidindi, Yossi Matias, Katherine Chou, Greg S. Corrado, Shravya Shetty, Daniel Tse, Shruthi Prabhakara, Daniel Golden, Rory Pilgrim, Krish Eswaran, and Andrew Sellergren. Elixr: Towards a general purpose x-ray artificial intelligence system through alignment of large language models and radiology vision encoders, 2023.



APÉNDICE

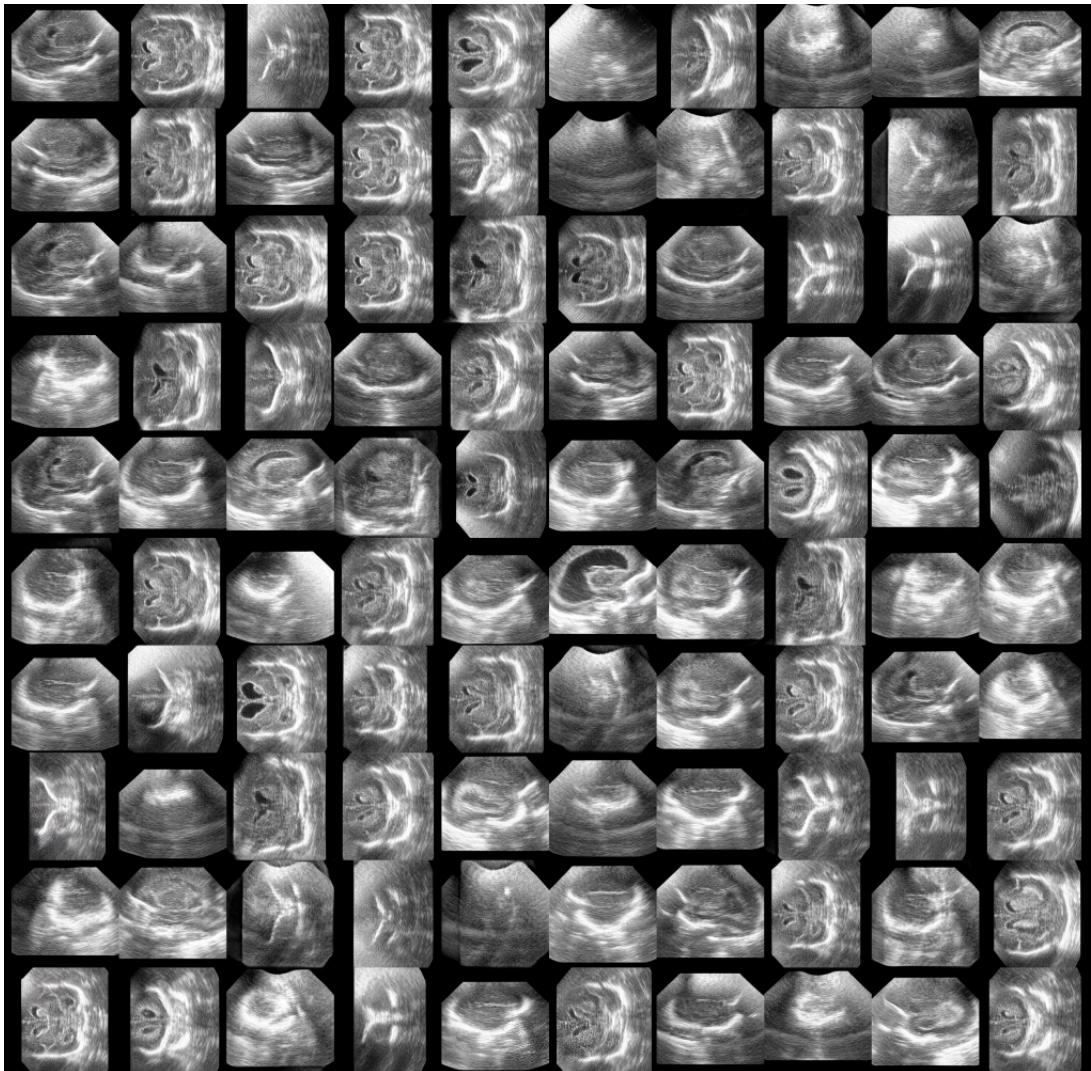
# A

## Imágenes Generadas

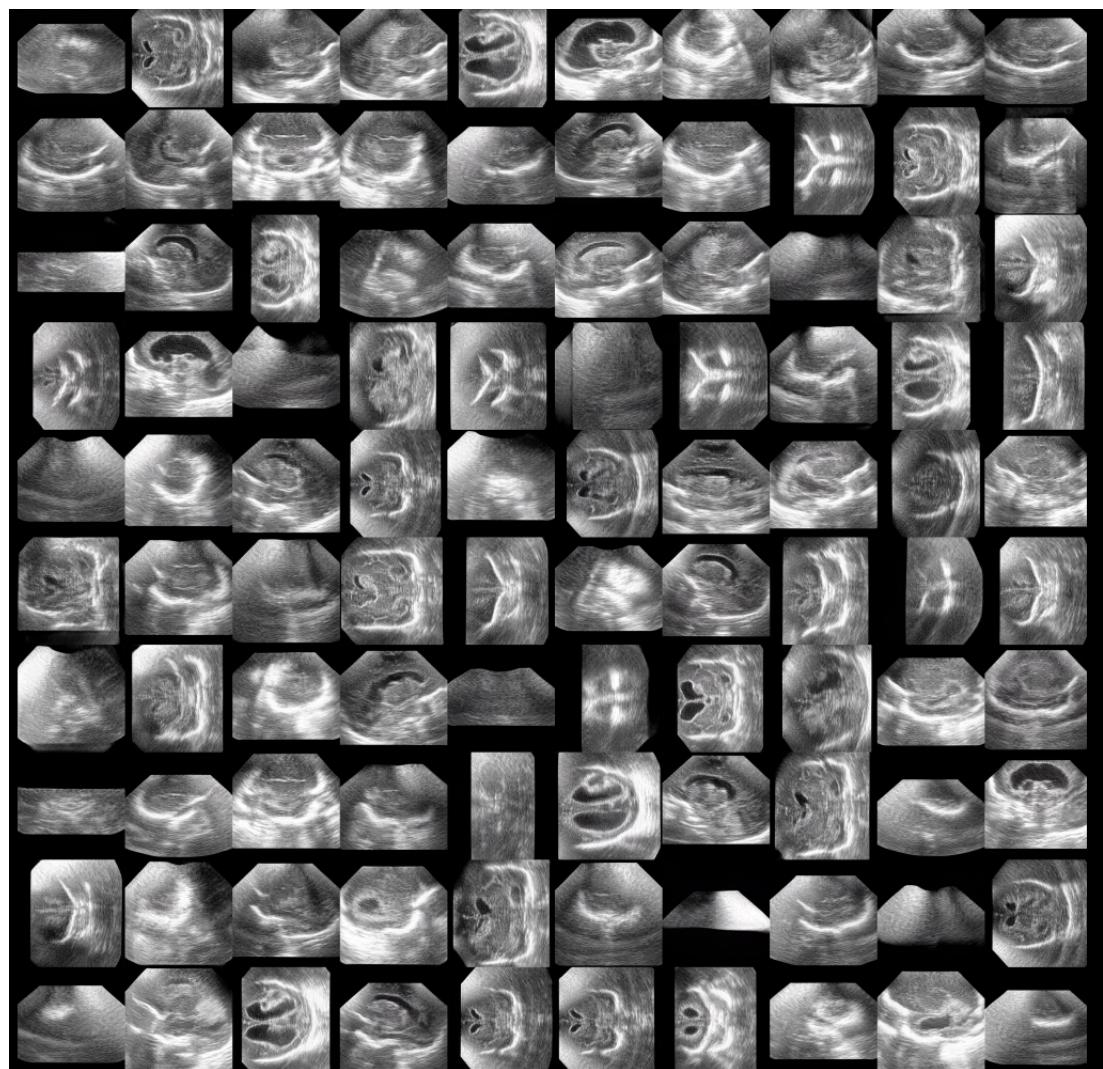
A continuación varios bloques de imágenes sintéticas con distintos valores de truncamiento.

## A. IMÁGENES GENERADAS

---



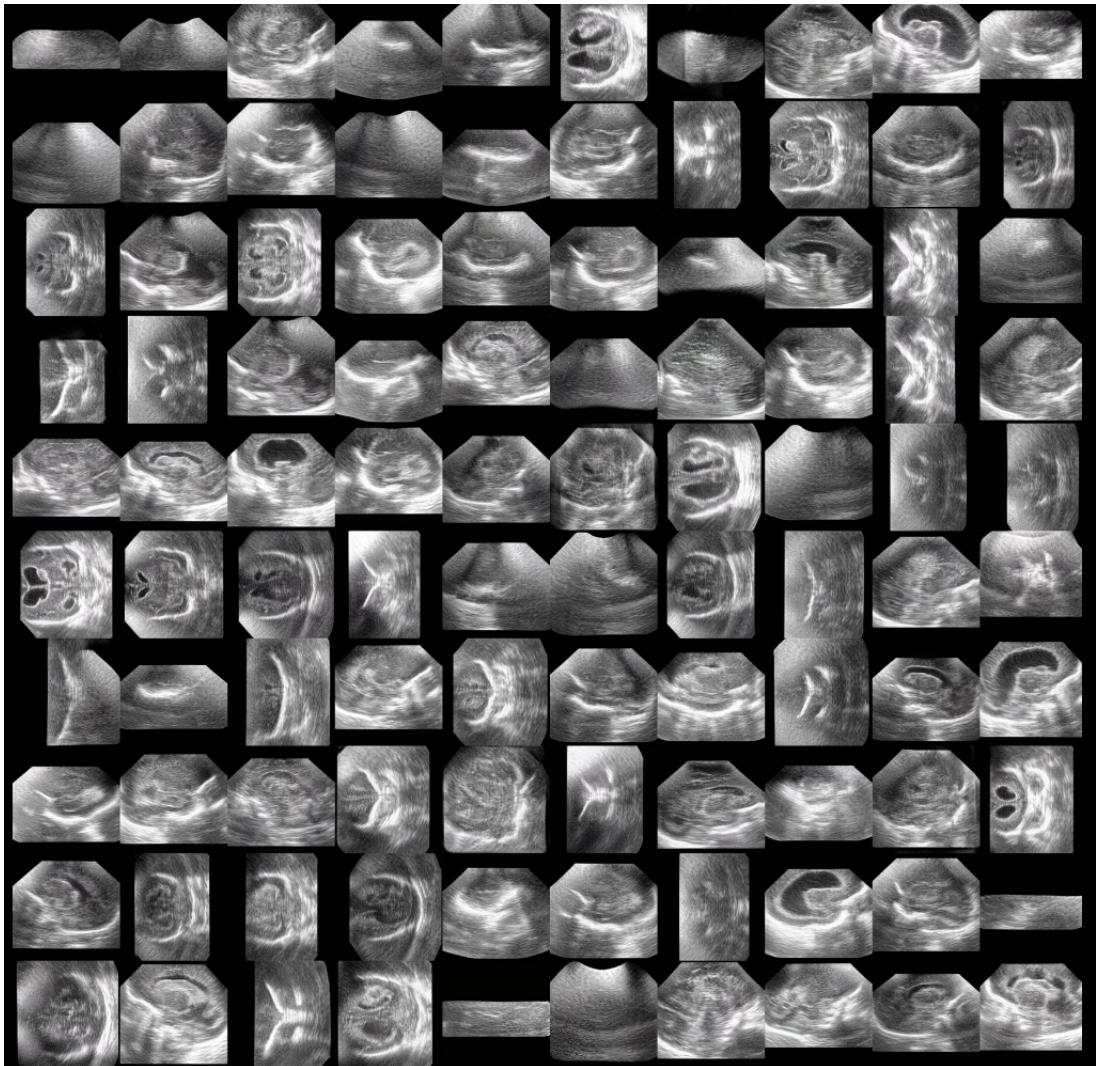
**Figura A.1:** Imágenes generadas con valor de truncamiento 0,5



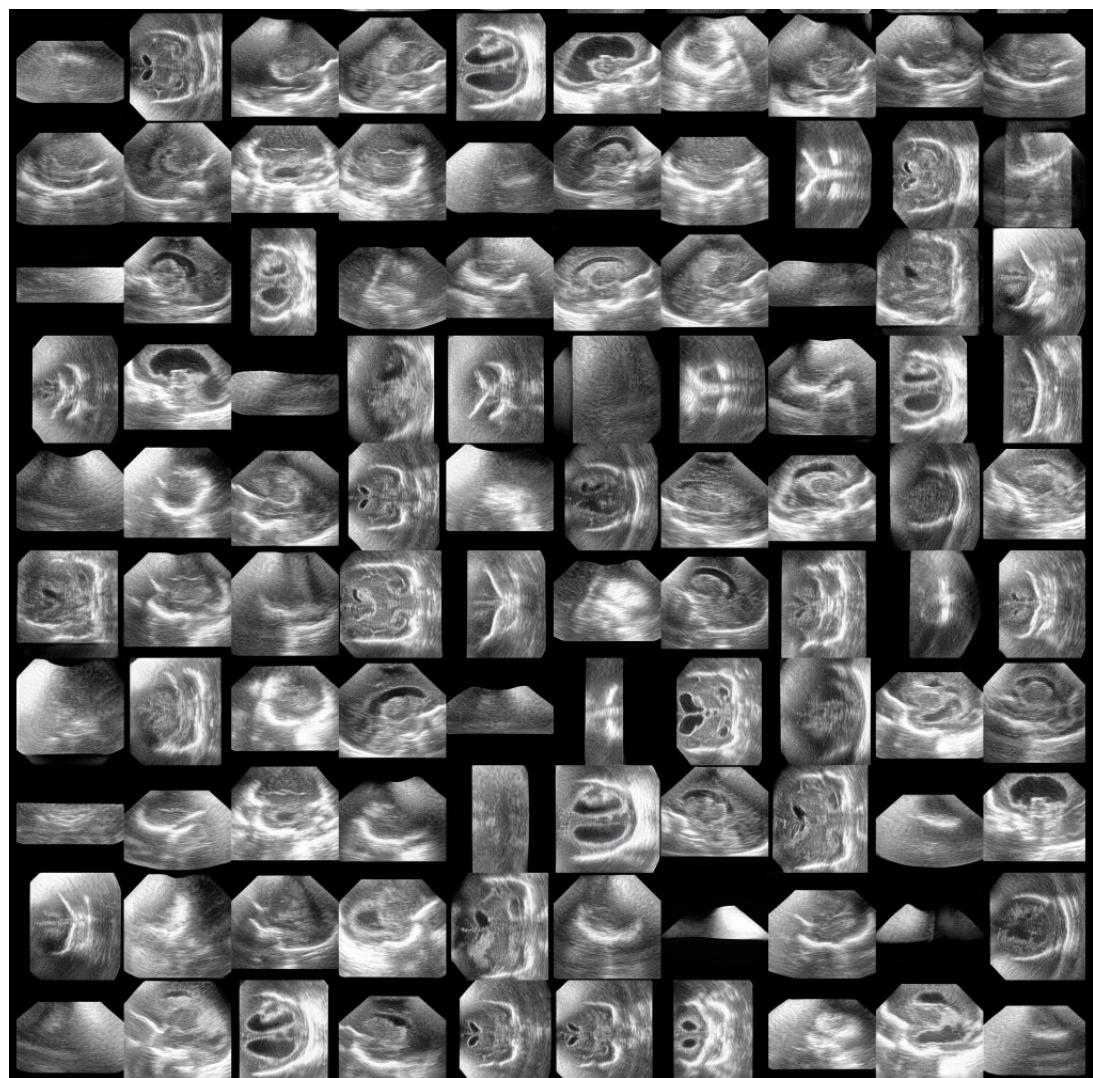
**Figura A.2:** Imágenes generadas con valor de truncamiento 0,8

## A. IMÁGENES GENERADAS

---



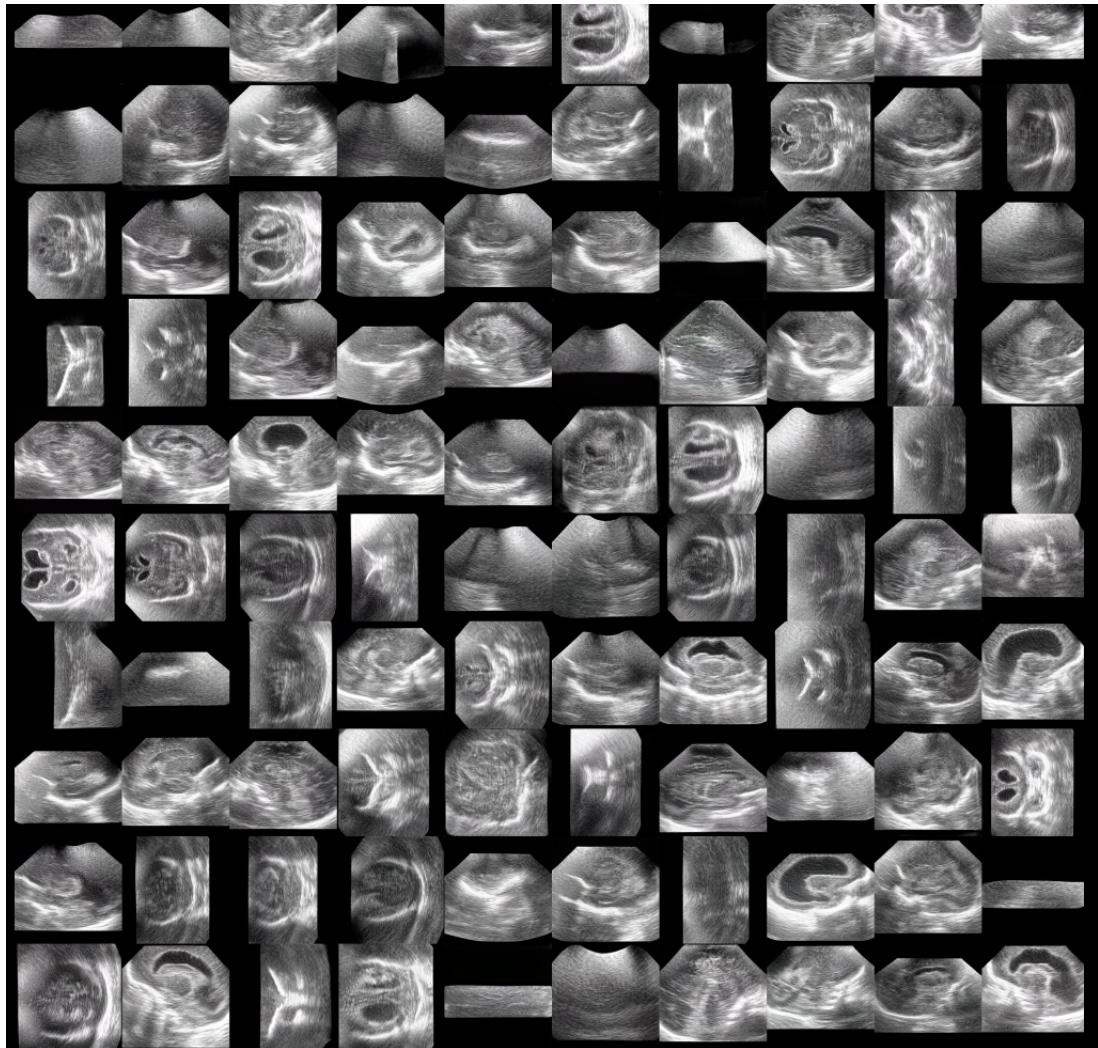
**Figura A.3:** Imágenes generadas con valor de truncamiento 1,0



**Figura A.4:** Imágenes generadas con valor de truncamiento 1,0

## A. IMÁGENES GENERADAS

---



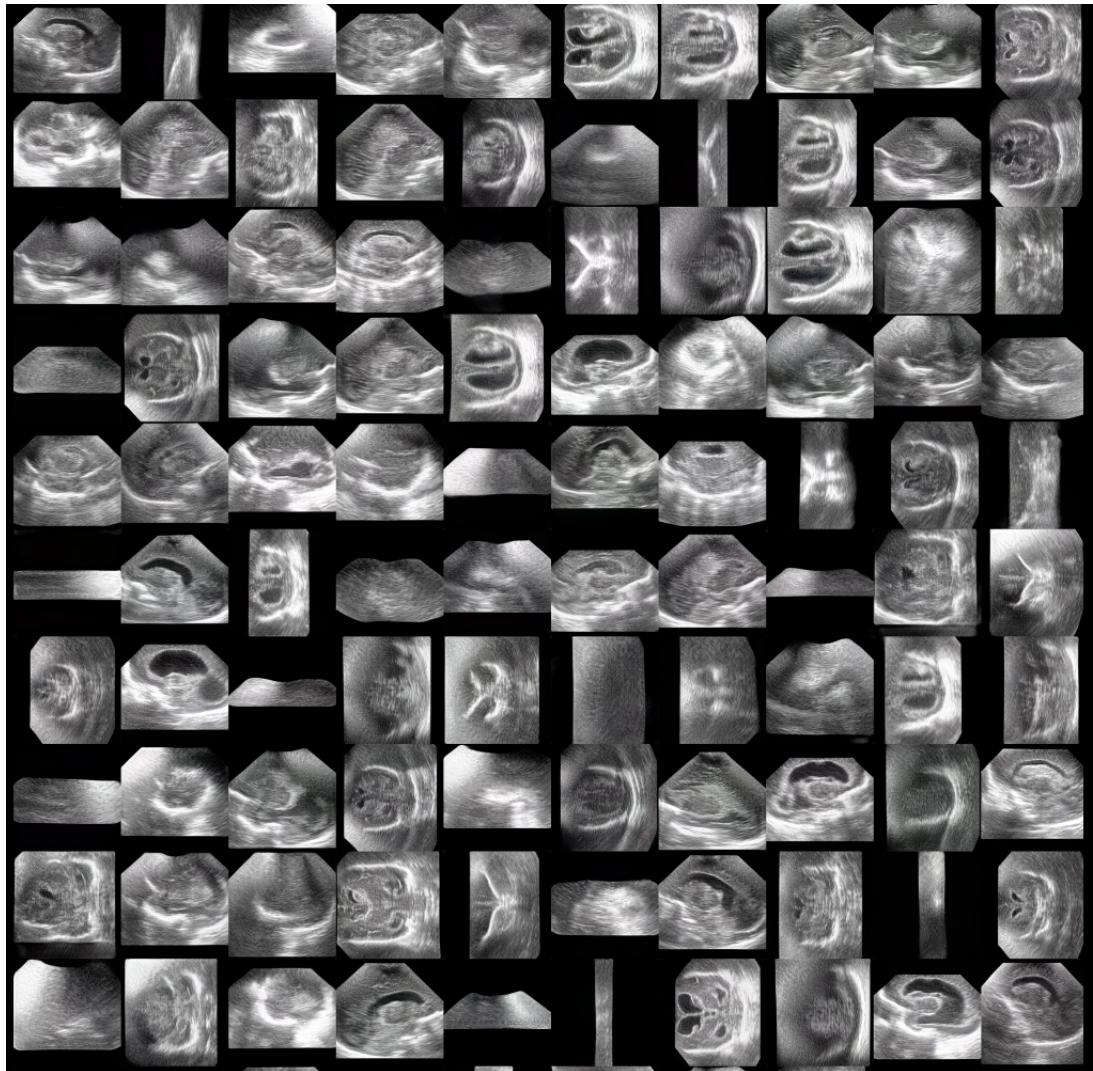
**Figura A.5:** Imágenes generadas con valor de truncamiento 1,3



**Figura A.6:** Imágenes generadas con valor de truncamiento 1,5

## A. IMÁGENES GENERADAS

---



**Figura A.7:** Imágenes generadas con valor de truncamiento 1,5