Universidad de Oviedo

Facultad de Ciencias

Grado en Física

Reconstrucción de trazas con aprendizaje automático en el experimento CMS del LHC

Trabajo de Fin de Grado

Celia González Fernández

Tutor: Javier Fernández Menéndez

Junio 2025

Declaración de Originalidad

De acuerdo con lo expresado en el artículo 8.3 del Reglamento sobre la asignatura Trabajo Fin de Grado en la Universidad de Oviedo, aprobado por su Consejo de Gobierno el 5 de marzo de 2020 (BOPA de 30 de marzo de 2020), quiero expresar lo siguiente:

Yo, **Celia González Fernández**, con NIE **21090972H**, en relación con la memoria que presento ante el Tribunal, para su valoración como Trabajo Fin de Grado, quiero **DECLARAR** que soy la autora de la misma, habiendo citado debidamente las fuentes utilizadas en su desarrollo.

Para que conste, firmo el presente documento.

En Oviedo, a 18 de junio de 2025

Fdo.: GONZALEZ FERNANDEZ CELIA - 21090972H

Firmado digitalmente por GONZALEZ FERNANDEZ CELIA - 21090972H
Fecha: 2025.06.18 12:06:06 +02'00'

A Javi, por su confianza, su tiempo y su dedicación sincera

A papá, mamá y mi familia, por su amor incondicional

Y a mis amigos, por ser mi refugio y mi segunda casa

Índice general

Capítulo 1

Introducción a la Física de Altas Energías

1.1. El Modelo Estándar

Desde mediados del siglo pasado, el trabajo colectivo de miles de físicos ha permitido desarrollar una comprensión profunda de los componentes más básicos de la materia. De esta manera se ha demostrado que todo lo que existe en el universo que nos rodea está formado por un conjunto reducido de partículas fundamentales, cuyas interacciones están gobernadas por cuatro fuerzas elementales.

La teoría que mejor describe las relaciones entre estas partículas y tres de las fuerzas se resume en el Modelo Estándar de la física de partículas. Esta formulación teórica, consolidada a principios de los años 70, ha logrado dar explicación a una enorme cantidad de resultados experimentales, además de predecir fenómenos que posteriormente han sido confirmados con éxito en numerosos experimentos, especialmente en grandes aceleradores de partículas como el LHC.

El Modelo Estándar se basa entonces en la existencia de un conjunto limitado de partículas elementales y en las interacciones que se producen entre ellas. Estas partículas, representadas en Figura 1.1, se agrupan en dos principales grupos según su naturaleza cuántica: los **fermiones**, que poseen espín semientero y constituyen la materia ordinaria, y los **bosones**, con espín entero, que son los responsables de transmitir las fuerzas fundamentales. [1]

En total, se conocen doce **fermiones** elementales organizados en dos familias: los quarks y los leptones. Cada una de estas familias incluye seis partículas distintas distribuidas en tres generaciones. Las partículas de la primera generación (como el conocido electrón o los quarks up y down) son las más

ligeras y estables y forman toda la materia que nos rodea. Mientras que las generaciones superiores contienen partículas más masivas y menos estables, que suelen desintegrarse rápidamente en partículas de generaciones anteriores.

Los quarks nunca se observan de forma aislada debido al confinamiento de color que les hace combinarse dando lugar a partículas compuestas incoloras llamadas hadrones, como los protones o neutrones. Se conocen seis sabores de quarks con carga fraccionaria +2/3 o -1/3, y sus correspondientes antipartículas resultantes de invertir los signos de las cargas.

Los leptones en cambio sí que existen de forma libre en la naturaleza. De nuevo son seis tipos que incluyen partículas como el electrón, el muón y el tau de cargas enteras negativas y sus correspondientes neutrinos (ν). Los neutrinos son partículas neutras y de masa muy pequeña que apenas interactúan con la materia, lo que los hace difícilmente detectables.

Por otro lado, los **bosones** son las partículas encargadas de mediar las interacciones entre fermiones. El Modelo Estándar contempla tres de ellas: la interacción electromagnética mediada por el fotón (γ), la interacción débil mediada por los bosones $W^{\pm}$ y Z^0 y la interacción fuerte causante del confinamiento de los quarks, mediada por los gluones. A todas estas se le suma el bosón de Higgs, descubierta en 2012 en el CERN, cuya interacción con otras partículas explica el origen de su masa [2].

Es importante por último remarcar que aunque la gravedad es también una fuerza fundamental, no está incluida en el Modelo Estándar ya que aún no se dispone de una formulación cuántica completa para ella.

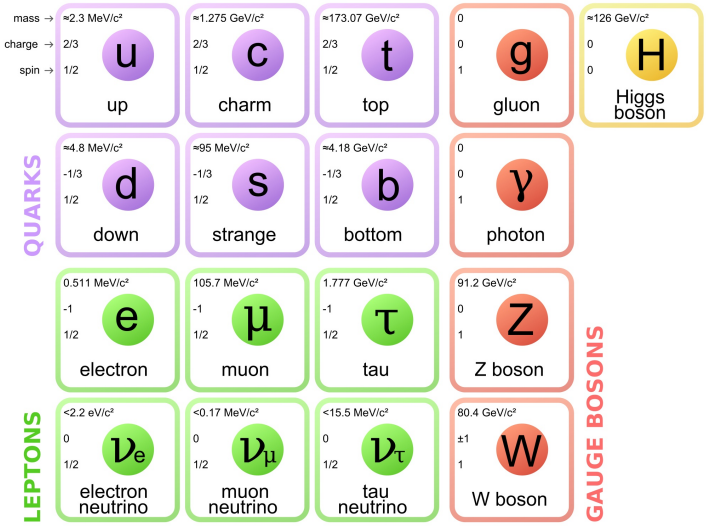


Figura 1.1: Partículas elementales descritas por el Modelo Estándar. Recuperada de [3]

No obstante, a pesar de que el Modelo Estándar describe con gran precisión los constituyentes de la materia y sus interacciones, muchas de las partículas son extremadamente masivas o simplemente inestables con vidas medias demasiado cortas y no es posible observarlas en condiciones normales. Esto hace que para poder estudiar este mundo subatómico tan complejo, sea necesario alcanzar escalas de energía muy grandes y resoluciones espaciales extremadamente pequeñas, es decir, es necesario recurrir a la física de altas energías.

1.2. Física de Altas Energías

En física de partículas se trabaja habitualmente con magnitudes relativistas como la energía, el momento o la masa. Para simplificar la notación, es común adoptar unidades naturales, en las que $c = \hbar = 1$ de modo que podemos hablar simplemente de la relación $E^2 = p^2 + m^2$. Bajo estas condiciones masa, energía y momento son físicamente equivalentes y se expresan en electron-voltios (eV) y sus correspondientes múltiplos.

De acuerdo con los principios de la mecánica cuántica, para poder sondear estructuras cada vez más pequeñas, es necesario utilizar partículas con momentos cada vez más elevados. Concretamente existe una relación inversa entre la resolución espacial Δr y el momento transferido q que sigue aproximadamente $\Delta r \sim \hbar/q$.

De esta manera, por ejemplo con haces de partículas que transfieran un momento del orden de 10GeV, es posible alcanzar una resolución del orden de 10^{-16}m, es decir, unas diez veces mayor que el radio del protón. Este principio es esencial para poder estudiar la estructura interna de hadrones como el protón o el neutrón y revelar así la presencia de sus constituyentes internos, los quarks. [4]

Luego para acceder al mundo subatómico es necesario recurrir al uso de aceleradores, estos dispositivos utilizan campos electromagnéticos para acelerar partículas cargadas a altas velocidades y así hacerlas colisionar posteriormente con otras partículas. Es aquí dónde nacen los esquemas experimentales principalmente usados en física de altas energías: los experimentos de blanco fijo y los experimentos de colisión de haces.

- En los experimentos de **blanco fijo**, un haz de partículas aceleradas indice sobre un objetivo estacionario. Este enfoque es técnicamente más sencillo ya que solo se tiene que acelerar un único haz y permite altas tasas de interacción.

  Sin embargo, tiene una limitación importante pues la energía disponible en el centro de masas ($\sqrt{s}$) para la creación de nuevas partículas crece proporcional a la raíz cuadrada de la energía (E_A) del haz ($\sqrt{s} \sim \sqrt{E_A}$). [4]

- Por el contrario, en los **colisionadores de haces**, dos haces de partículas de la misma energía (E) se mueven en direcciones opuestas y colisionan frontalmente. En este caso la energía del centro de masas viene dada por $\sqrt{s} = 2E$.

  Esta diferencia es crucial ya que permite alcanzar energías mucho mayores de manera más eficiente, haciendo posible la creación de partículas muy masivas. Luego es precisamente por ello que este sea el esquema que se utiliza en los grandes aceleradores modernos como el LHC.

Además de las diferencias energéticas de ambos tipos de experimentos, hay otras magnitudes clave que permiten cuantificar la eficiencia de un experimento para producir ciertos eventos físicos. Dos de las más importantes son la sección eficaz y la luminosidad.

- La **sección eficaz** (σ) es una medida de la probabilidad de que ocurra un determinado proceso. Se expresa en unidades de superficie y depende del tipo y la energía de las partículas interactuantes.
- La **luminosidad instantánea** ($\mathcal{L}$) es una característica del experimento y representa cuántas partículas inciden por unidad de área y tiempo en la zona de colisión. De esta manera, una mayor luminosidad implica una mayor probabilidad de registrar eventos interesantes.

Estas dos magnitudes se pueden combinar para determinar el número de eventos esperados (N) a través de la siguiente relación:

$$N = \mathcal{L}_{int}\, \sigma \tag{1.1}$$

Donde $\mathcal{L}_{int}$ es la luminosidad integrada, es decir, la luminosidad instantánea acumulada a lo largo del tiempo. Esta magnitud tiene por tanto unidades inversas de área y suele expresarse comúnmente en múltiplos inversos de barn (b^{-1}), dónde $1b = 10^{-28}\mathrm{m}^2$. [5]

En general, los experimentos de blanco fijo ofrecen mayores luminosidades debido a la alta densidad del blanco y la estabilidad del haz, pero su baja energía en el centro de masas limita el acceso a ciertos procesos. Por ello, son más adecuados para estudiar fenómenos frecuentes. En cambio, los colisionadores alcanzan energías mucho mayores, lo que permite acceder a procesos raros o que requieren gran energía, como la producción de partículas masivas.

Posteriormente, una vez seleccionado el método experimental que se seguirá, es necesario contar con sistemas de detección sofisticados que sean capaces de estudiar las partículas producidas en las colisiones. Estos detectores se colocan cercanos a la zona de interacción y están diseñados para medir con gran precisión las propiedades de las partículas generadas: sus trayectorias, energías, momentos...

En definitiva, para explorar el mundo subatómico es necesario hacer colisionar haces de partículas a

altas energías y estudiar sus productos con detectores precisos. Es precisamente esto lo que motiva el proximo capítulo en el que se presentará el LHC, el mayor colisionador de partículas actual.

Capítulo 2

El LHC y su experimento CMS

El Gran Colisionador de Hadrones o LHC por sus siglas en inglés (*Large Hadron Collider*), es el acelerador de partículas más grande y potente del mundo. Comenzó a funcionar el 10 de septiembre de 2008 y es a día de hoy la última incorporación al complejo de aceleradores del CERN.

Concretamente, el LHC es un acelerador circular de 27km de circunferencia ubicado a una profundidad de unos 100m bajo tierra, en la frontera entre Suiza y Francia. En el interior de este acelerador, dos haces de partículas de mucha energía viajan a velocidades cercanas a la de la luz antes de que se les haga colisionar. Los haces viajan en direcciones opuestas en dos tubos que se mantienen en altas condiciones de vacío para evitar posibles choques con moléculas de gas.

Las partículas que se aceleran en el LHC son hadrones, concretamente protones o núcleos de plomo. Esta elección no es casual: por un lado, las partículas tienen que tener carga eléctrica para poder manipularlas fácilmente usando campos electromagnéticos, y además deben ser estables para poder ser aceleradas durante un tiempo prolongado. Estas condiciones ya reducen mucho el abanico de posibilidades, dejando solamente candidatos como protones, electrones o núcleos atómicos. En el caso concreto del LHC, al tratarse de un acelerador circular, resulta más ventajoso utilizar partículas pesadas como los protones, ya que pierden mucha menos energía por radiación sincrotrón que partículas más ligeras como los electrones.

Para mantener estos haces de protones circulando en la órbita circular del LHC, se utiliza un sistema de imanes extremadamente potente. En particular, se usan imanes dipolares para curvar las trayectorias de las partículas cargadas e imanes cuadrupolares para mantener los haces enfocados y agrupados. Estos imanes son superconductores, de manera que a bajas temperaturas conducen la electricidad sin resistencia haciendo el sistema mucho más efectivo. En total, hay más de mil imanes dipolares y unos cientos cuadrupolares repartidos a lo largo de todo el anillo. [6]

2.1. La Cadena de Inyección

No obstante, antes de que los haces puedan ser inyectados en el anillo principal del LHC y alcanzar las energías necesarias para las colisiones, deben pasar por un proceso previo de aceleración. Para ello se utiliza una cadena de aceleradores intermedios conocida como **cadena de inyección**.

Esta comienza en el Linac4, un acelerador lineal diseñado para impulsar iones de H^- hasta 160MeV. Justo antes de entrar en el siguiente acelerador, el Proton Synchrotron Booster (PSB), estos iones pierden sus dos electrones quedando solamente los protones [7]. A partir de aquí, los protones son acelerados en etapas progresivas, primero en el PSB hasta unos 2GeV, luego en el Proton Synchrotron (PS) hasta unos 25GeV y finalmente en el Super Proton Synchrotron (SPS), que los entrega al LHC con una energía de 450 GeV [8]. Una vez en el LHC, los haces de protones son acelerados en sentidos opuestos hasta alcanzar energías del orden de los TeV por haz antes de colisionar. Este proceso puede verse representado en Figura 2.1.

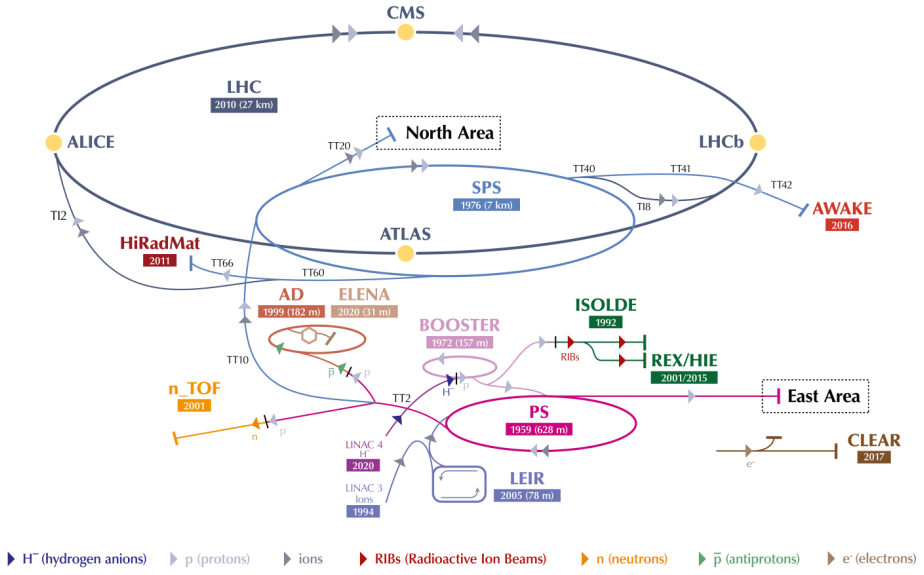


Figura 2.1: Complejo de aceleración del CERN. Se muestran los diferentes aceleradores y detectores. Modificado de [9]

Para maximizar la probabilidad de interacción entre partículas, los haces no se inyectan como un flujo continuo sino que se estructuran en pequeños grupos denominados paquetes o *bunches*. Concretamente, cada haz tiene del orden de unos 2808 *bunches* con aproximadamente unos 10^{11} protones por *bunch* [10]. Debido a las altas velocidades a las que viajan, son capaces de completar muchas vueltas por segundo, generando hasta 40 millones de colisiones por segundo en cada punto de interacción.

2.2. Experimentos

Una vez los haces han alcanzado las energías deseadas dentro del anillo del LHC, se dirigen hacia zonas específicas de interacción donde tienen lugar las colisiones. Concretamente hay cuatro experimentos principales en torno a los puntos de colisión: LHCb, ALICE, ATLAS y CMS cuyas posiciones aparecen representadas en Figura 2.1.

- **LHCb** (*LHC beauty*) está diseñado para explorar la asimetría materia-antimateria en el universo estudiando la física del quark b.
- **ALICE** (*A Large Ion Collider Experiment*) está dedicado a experimentos con iones pesados. Estos dan lugar a un estado de la materia conocido como plasma de quarks-gluones que se cree que estuvo presente justo después del Big Bang.
- **ATLAS** (*A Toroidal LHC ApparatuS*) y **CMS** son los dos grandes experimentos del LHC. Ambos son detectores de proposito general, es decir, utilizados para estudiar una amplia variedad de fenómenos como la búsqueda de partículas que forman la materia oscura o del bosón de Higgs.

Además también existen experimentos más pequeños que complementan el trabajo de los cuatro detectores principales. Entre ellos se encuentran: **TOTEM** dedicado a estudiar la dispersión elástica de protones y la sección eficaz total; **LHCf** centrado en los rayos cósmicos mediante la detección de partículas en la dirección del haz; **MoEDAL** diseñado para buscar monopolos magnéticos y otras partículas exóticas con carga múltiple; **FASER** diseñado para detectar partículas muy ligeras que pueden escapar en la dirección del haz tras las colisiones; y **SND@LHC** que complementa al anterior detectando neutrinos.

2.3. Evolución y campañas de datos

El LHC no es un proyecto estático, sino que está en constante evolución. Concretamente su funcionamiento se organiza en campañas denominadas *Run* que abarcan varios años de obtención de datos de manera continua. Entre *Run*, el acelerador entra en periodos de parada conocidos como *Long Shutdowns* (LS) durante los cuales se llevan a cabo tareas de mantenimiento y mejora tanto en el acelerador como en los experimentos asociados. Es gracias a estas mejoras, que cada nuevo *Run* ha permitido alcanzar energías de colisión más altas y recopilar datos con mayor luminosidad.

La campaña de datos actual se conoce como *Run* 3 y fue iniciada en 2022 tras el *Long Shutdown 2*. Como puede observarse en Figura 2.2, la luminosidad integrada registrada por en el LHC ha ido creciendo a lo largo de cada campaña, y esta tendencia continuará en el futuro con el inicio del programa

***High-Luminosity* LHC** (HL-LHC) que tiene como objetivo alcanzar una luminosidad integrada de 3000 fb^{-1} para el año 2042.

Concretamente, el crecimiento de la luminosidad se debe, en parte, a que se introducen cada vez más protones en los paquetes o *bunches* que se hacen colisionar con el fin de aumentar la probabilidad de que se produzcan interacciones útiles.

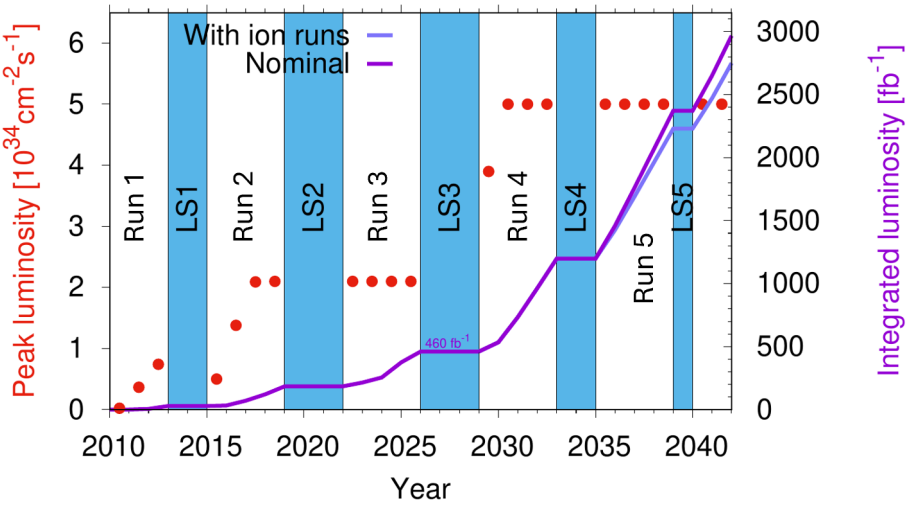


Figura 2.2: Evolución de la luminosidad alcanzada hasta la actualidad y prevista tras el LS3 en el LHC. Se muestran tantos los picos de luminosidad instantánea máxima en cada año operativo (en rojo, eje izquierdo) como la luminosidad integrada acumulada a lo largo del tiempo (en morado, eje derecho). Las zonas sombreadas en azul corresponden a los periodos de parada (LS). Recuperada de [11]

2.4. El experimento CMS

El Solenoide Compacto de Muones o CMS por sus siglas en inglés es uno de los cuatro grandes experimentos del LHC y está diseñado como un detector de propósito general. Su objetivo es estudiar una amplia variedad de procesos físicos que pueden ocurrir en la colisión protón-protón.

El detector CMS es una de las estructuras más grandes y pesadas jamás construidas en el ámbito de la física de partículas. Mide aproximadamente 21m de largo, 15m de diámetro y alcanza un peso de unas 14000 toneladas [12]. Su elemento más característico es el potente imán solenoidal alrededor del cual está construido. Este es capaz de generar un campo magnético de 3.8T capaz de curvar las trayectorias de las partículas cargadas facilitando su caracterización.

2.4.1. Sistema de Coordenadas

Para describir la geometría del detector y las trayectorias de las partículas es necesario introducir el sistema de coordenadas que se usa en este detector.

Concretamente CMS sitúa el origen de coordenadas en el punto de colisión entre los dos haces. Así, el eje x apunta hacia el centro del anillo del LHC, el eje y apunta hacia arriba siendo perpendicular al plano del acelerador y el eje z se orienta a lo largo de la dirección del haz que circula en sentido antihorario (geográficamente se puede ver como que este apunta hacia el macizo del Jura) [13]. Debido a la forma del detector, resulta muy útil trabajar en coordenadas cilíndricas, dónde el plano definido por los ejes x e y se conoce como **plano transverso** y el eje z sigue la dirección longitudinal.

De esta manera, en el plano transverso se define el ángulo azimutal $\phi \in [-\pi, \pi)$ medido desde el eje x positivo y moviéndose en sentido antihorario a lo largo de todo el plano. En este mismo plano, se define también la coordenada radial (r) que representa la distancia desde el eje del haz hasta el punto donde se detecta la partícula. Por otro lado, el ángulo polar $\theta \in [0, \pi)$ se mide desde el eje z. Esto puede verse representado a continuación en Figura 2.3.

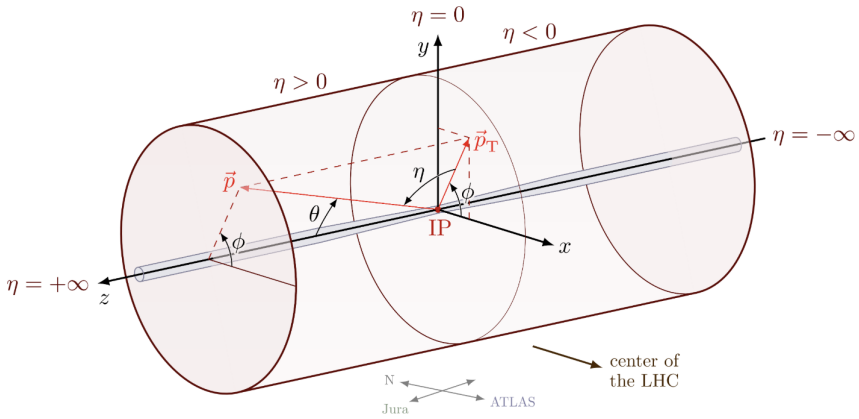


Figura 2.3: Sistema de coordenadas en CMS con los ángulos θ, ϕ y la pseudorapidez η. Por convención, se considera positivos los valores en la dirección del macizo del Jura. Recuperada de [14]

No obstante, en vez de el ángulo polar θ, en este ámbito prefiere utilizarse la **pseudorapidez** (η). Esta se define como

$$\eta = -\ln\left(\tan\left(\frac{\theta}{2}\right)\right) \tag{2.1}$$

y puede tomar valores en el rango $(-\infty, \infty)$ de manera que $\eta = 0$ corresponde con una salida en el plano perpendicular al haz ($\theta = 90°$) mientras que $\eta = \infty$ implica un desplazamiento longitudinal a lo largo del eje z [15]. Esta variable resulta muy ventajosa ya que en experimentos como CMS, el sistema

de referencia del laboratorio no siempre coincide exactamente con el centro de masas del sistema debido a posibles diferencias en la energía longitudinal de los protones. Aunque la pseudorrapidez no es estrictamente invariante bajo *boosts* longitudinales, las diferencias de pseudorrapidez en el régimen de altas energías ($p >> m$) sí lo son, lo que la hace especialmente útil para comparar eventos sin que los resultados dependan del sistema de referencia elegido.

Esto lleva a que el sistema de coordenadas habitual para definir el momento de una partícula sea (p_T, η, ϕ), dónde p_T hace referencia al momento medido en el plano transverso, es decir no es más que $p_T = \sqrt{p_x^2 + p_y^2}$.

2.4.2. Estructura general del detector

CMS es un detector de capas concéntricas que rodean el punto de interacción de los haces de protones. Cada capa está diseñada para detectar un tipo específico de partícula con el fin de lograr una reconstrucción eficiente de los eventos.

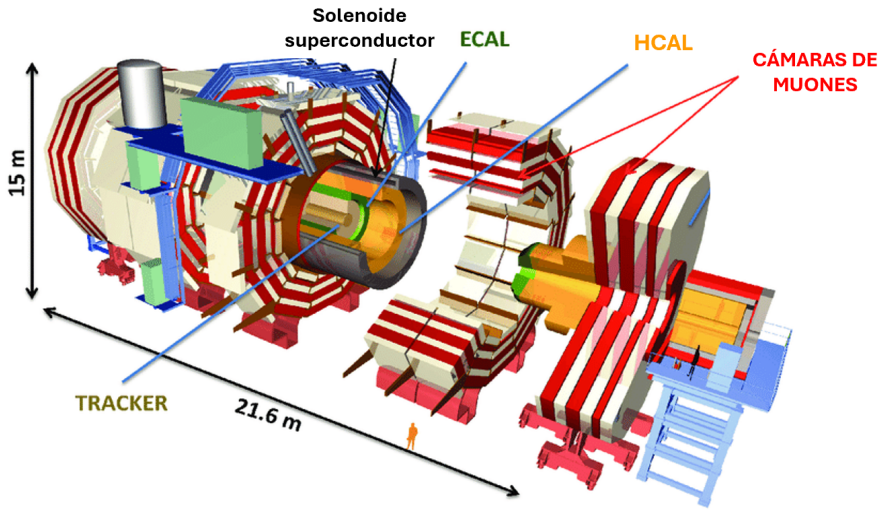


Figura 2.4: Esquema general de la estructura del detector CMS. Modificado de [16]

Como se muestra en Figura 2.4, desde la parte más interna hasta la más externa incluye:

- El **detector de trazas** o ***tracker*** es la capa más interna y cercana al punto de interacción. Se hablará más en profundidad de ella posteriormente.
- El **calorímetro electromagnético** (ECAL) está diseñado para medir la energía de electrones y fotones. Su función es detener completamente estas partículas y medir su energía a partir de su interacción con la materia. Para ello, utiliza cristales de tungstanato de plomo ($PbWO_4$) que producen luz de centelleo cuando una partícula electromagnética los atraviesa. Esta luz es

proporcional a la energía de la partícula incidente y se recoge con fotodetectores situados detrás de cada cristal, se convierte en señal eléctrica, se amplifica y se analiza posteriormente. [17]

- El **calorímetro hadrónico** (HCAL) se encarga de medir la energía de hadrones y proporciona una medición indirecta de la presencia de partículas sin carga no interactuantes como los neutrinos. Para ello el HCAL está diseñado con una geometría altamente hermética que maximiza la absorción de energía en todas direcciones, permitiendo identificar desequilibrios de energía en el plano transverso del detector, lo que constituiría una señal de partículas invisibles.

  Su estructura consiste en una sucesión de capas dispuestas de forma que se minimicen los huecos entre ellas, alternando materiales densos con láminas de plástico centellador. Cuando los hadrones interactúan con el material denso, generan una cascada de partículas secundarias cuya energía es registrada a través de la luz emitida en los centelladores. Esta señal luminosa es recogida por fibras ópticas y posteriormente transformada en señales eléctricas.[18]

- El **solenoide superconductor** se encarga de generar un campo magnético uniforme de 3.8T capaz de curvar las trayectorias de las partículas cargadas que atraviesan el sistema. Esto permite determinar el momento de las partículas a partir del radio de curvatura mientras que el signo de su carga puede deducirse del sentido de curvatura.

- Las **cámaras de muones** rodean completamente al resto de componentes y permiten identificar las trayectorias de los muones, que atraviesan el resto de las capas casi sin interaccionar. Para ello utiliza varios tipos de cámaras distribuidas en capas: los tubos de deriva (DT), las cámaras de tiras catódicas (CSC), las cámaras de placas resistivas (RPC) y las más recientes, los multiplicadores gaseosos de electrones (GEM).

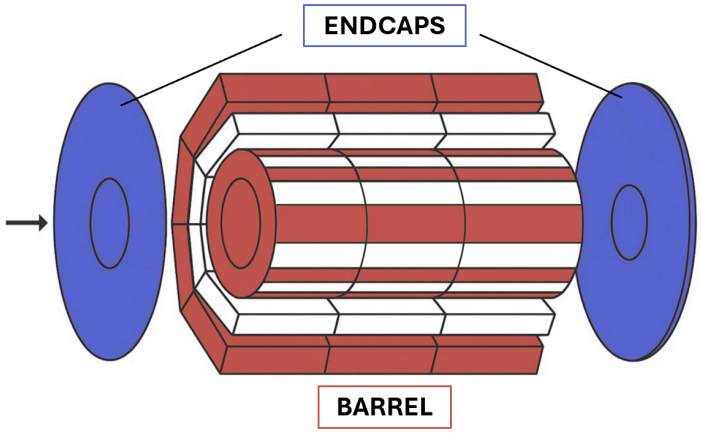


Figura 2.5: Vista esquemática del detector CMS donde pueden diferenciarse el *barrel* y los *endcaps* representados en colores rojo y azul respectivamente

Es importante puntualizar además que dado que el detector está construido en una geometría cilíndrica en torno al eje del haz, se suele hablar de dos regiones principales: el ***barrel***, que constituye la región central y rodea de forma cilíndrica al punto de interacción, y los dos ***endcaps*** que se sitúan en los extremos del detector perpendiculares al eje del haz. Esto puede verse en Figura 2.5

2.4.3. El detector de trazas

El detector de trazas, también conocido como *tracker* constituye el componente más interno del sistema de detección en CMS y es fundamental para el estudio detallado de las trayectorias de las partículas producidas tras las colisiones. Para hacer esto posible consta de dos regiones, el **detector de píxeles** y el **detector de tiras de silicio**. Puesto que es con los datos de esta zona con los que se trabajará posteriormente, se hablará con más detalle durante este apartado.

El **detector de píxeles** constituye la región más interna del *tracker* y está diseñado para registrar los primeros pasos en la trayectoria de las partículas cargadas que emergen del punto de interacción. Para ello cuenta con una gran cantidad de módulos, como los que se muestran en Figura 2.6, formados por un sensor de silicio pixelado y uno o varios chips de lectura.

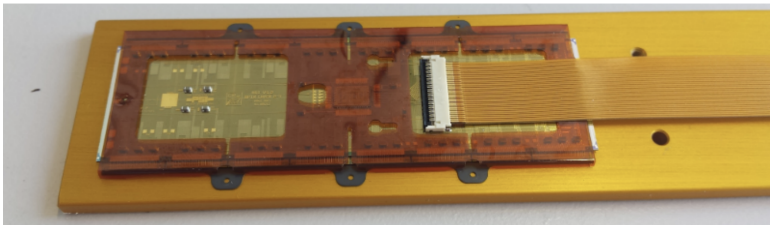


Figura 2.6: Imagen de un modulo real. Recuperada de [19]

Concretamente, el sensor de silicio es una lámina delgada segmentada en miles de pequeños píxeles donde cada uno tiene un tamaño típico de $100 \times 150\ \mu m^2$. De esta manera cuando una partícula cargada atraviesa el silicio, deposita energía e ioniza los átomos del material generando pares electrón-hueco. Estos portadores de carga se recogen mediante un campo eléctrico y se genera una señal eléctrica proporcional a la energía depositada que es amplificada y digitalizada por el chip de lectura.

No obstante, en la práctica, cuando una partícula atraviesa el sensor, no suele activar un único píxel, sino que la carga depositada se reparte entre varios píxeles cercanos. Estos píxeles que han sido activados y cuya carga es superior a un cierto umbral, forman lo que se conoce como un **clúster**. A partir de las señales de todos los píxeles del clúster, se calcula con gran precisión la posición por la que pasó la partícula. Gracias a la gran segmentación del detector y al tratamiento de los clústeres, se consigue una resolución espacial de unos 10μm que permite una reconstrucción muy precisa del punto de paso

de la partícula.

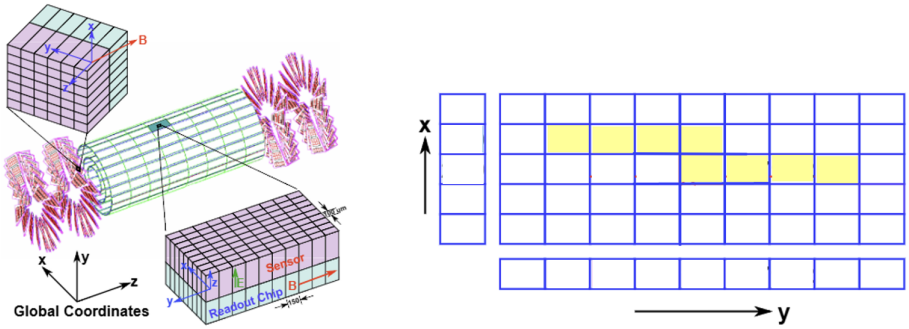


Figura 2.7: A la izquierda la geometría del detector de píxeles previa a la actualización *Phase-1*, donde se muestra la composición pixelada del sensor y el chip de lectura de los módulos. A la derecha la deposición de carga en varios píxeles cercanos que construyen un clúster. Modificado de [20]

Tras su mejora implementada en 2017, el detector de píxeles está en su versión *Phase-1* y cuenta actualmente con unos 1800 módulos distribuidos en dos regiones principales:

- El ***Barrel Pixel Detector*** **(BPIX)** cubre la región central del detector y está formado por cuatro capas cilíndricas concéntricas dispuestas a $r = 29, 68, 109, 160$ mm del eje del haz. Además de una capa extra más interna conocida como ***inner shield*** diseñada para proteger de la enorme radiación. El BPIX proporciona una cobertura completa hasta $\eta \approx 1.5$ aunque algunas trayectorias más inclinadas hasta $\eta \approx 2.1$ pueden atravesar al menos dos capas.

- El ***Forward Pixel Detector*** **(FPIX)** cubre los *endcaps*, es decir regiones de alta pseudorapidez, y está compuesto por tres discos por cada extremo del detector. Cada disco se divide en dos para poder optimizar la cobertura: el disco interno y el externo. El FPIX cubre la región $1.5 < \eta < 2.5$ complementando al BPIX en la región de transición entre el *barrel* y los extremos.

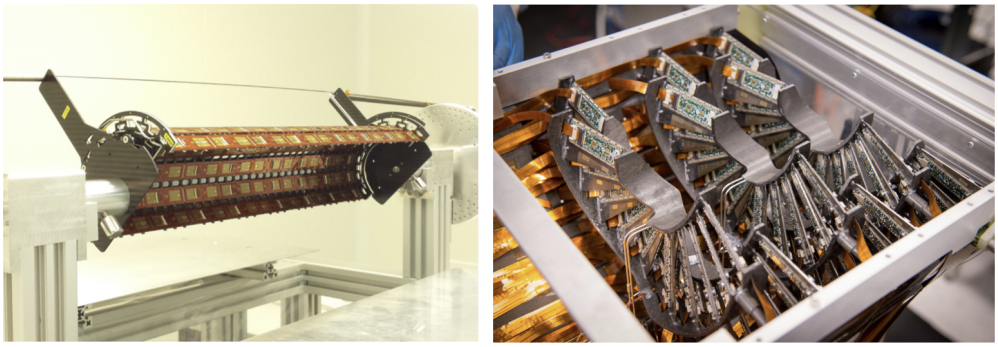


Figura 2.8: Imágenes reales de la estructura de módulos en el detector de píxeles. A la izquierda una mitad de una de las cuatro capas que forman el BPIX. A la derecha una mitad de la estructura de discos del FPIX en uno de los extremos del detector. Recuperadas de [19]

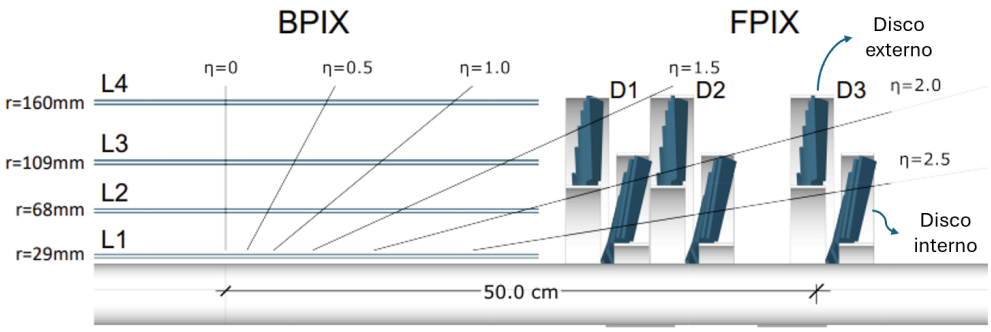


Figura 2.9: Vista esquemática del detector de píxeles del *tracker* de CMS. Se muestra la disposición de las capas del *barrel* y de los discos del *endcap*, así como su cobertura angular expresada en términos de pseudorapidez $|\eta|$. Modificada de [19]

Por otro lado, el **detector de tiras de silicio** se encuentra rodeando al detector de píxeles, extendiendo la cobertura hasta radios más grandes. Su función es también detectar el paso de partículas cargadas pero en este caso, a diferencia del caso anterior en el que cada módulo estaba dividido en una matriz bidimensional de pixeles, ahora cada módulo tiene una sola fila de tiras colocadas en paralelo. Las tiras son mucho más largas que los píxeles del orden de 10cm de largo y de 80-180 μm de ancho lo que hace que la resolución espacial sea menor, típicamente de unos 20-50μm.

Su funcionamiento es similar al anterior, basado en la ionización del silicio cuando una partícula cargada lo atraviesa. Sin embargo, debido a que todas las tiras de un módulo están alineadas en la misma dirección, solamente se obtiene información precisa sobre la posición de la partícula en la dirección perpendicular a las tiras. Es decir, se conoce en qué tira ocurrió la señal, pero no dónde a lo largo de la tira. Para poder recuperar la segunda coordenada espacial se utilizan módulos de doble cara (*double-sided*) que consisten en dos sensores de una sola cara colocados uno encima del otro y girados ligeramente entre sí.

A su vez el detector de tiras de silicio se divide en varias subregiones:

- El ***Tracker Inner Barrel*** **(TIB)** cubre la región del *barrel* para radios pequeños justo por encima del BPIX y está formado por cuatro capas cilíndricas.
- El ***Tracker Outer Barrel*** **(TOB)** también en la región del *barrel* rodea el TIB hasta radios más grandes. Está compuesto por seis capas cilíndricas.
- Los ***Tracker Inner Disks*** **(TID)** son discos ubicados en la región de transición entre el *barrel* y los *endcaps* diseñados para cubrir los huecos entre ambos. Hay tres discos a cada lado colocados perpendicularmente al eje del haz.

- Los ***Tracker Endcaps*** **(TEC)** cubren las regiones de alta pseudorrapidez hasta $|\eta| \sim 2.5$ en ambos extremos. Están formados por nueve discos dispuestos perpendicularmente al eje del haz.

Todas las estructuras que se han ido diferenciando durante este apartado pueden verse representadas a continuación en Figura 2.10.

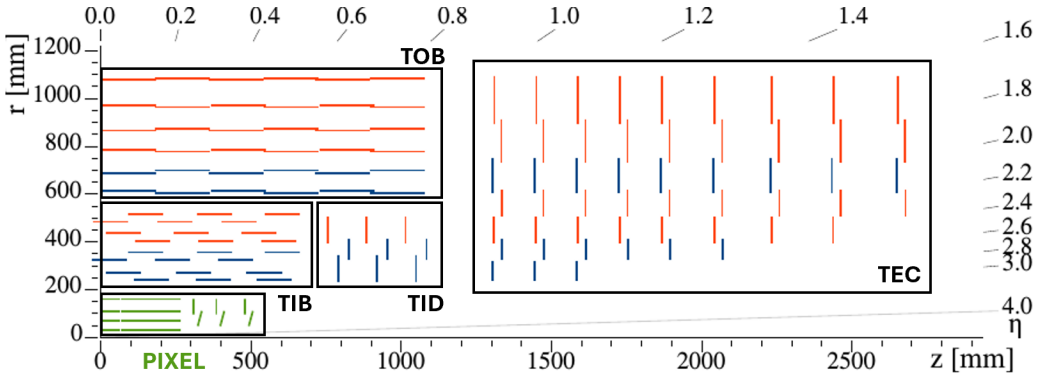


Figura 2.10: Esquema de una cuarta parte de la estructura del detector de trazas de CMS en el plano (r, z). El detector de píxeles se muestra en verde, nótese que no se diferencian el BPIX y FPIX que se mostraban en Figura 2.9. Los módulos del detector de tiras están codificados en rojo y azul dependiendo de si el sensor es *single-sided* o *double-sided* respectivamente. Modificado de [21]

En este punto es importante remarcar que esta disposición de los módulos en capas y discos que se ven tanto en Figura 2.9 como en 2.10, permite registrar múltiples puntos de impacto, conocidos como ***hits***, a medida que una partícula atraviesa el detector. Cada uno de estos *hits* deja una deposición de carga en un punto concreto de un sensor de silicio, y constituyen la única información directa que se puede obtener experimentalmente. De este modo, a partir de estos puntos dispersos distribuidos en las distintas capas del detector, es necesario reconstruir computacionalmente las trayectorias.

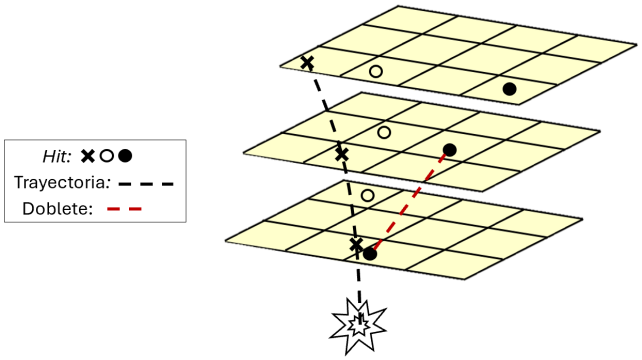


Figura 2.11: Representación esquemática del rastro dejado por varias partículas cargadas en el detector de trazas. Se representan las señales recogidas por el detector (*hits*). Así como la unión de estos puntos para dar lugar a dobletes (linea roja punteada) o trayectorias (linea negra punteada).

De hecho, al combinar la información recogida por dos de estos módulos, se pueden empezar a identificar los primeros indicios de una trayectoria. Concretamente, a este par de *hits* registrados por la misma partícula en dos módulos distintos se le conoce como **doblete** y será muy importante posteriormente en el contexto de análisis de los datos. Todos estos conceptos aparecen representados en Figura 2.11.

2.4.4. El sistema de trigger y tratamiento de datos

El LHC realiza hasta 40 millones de cruces de *bunches* por segundo (40MHz) generando una enorme cantidad de datos en cada cruce. Para poder seleccionar en tiempo real qué eventos merecen ser guardados para su analisis, CMS dispone de un sistema de selección de eventos conocido como ***trigger***. Concretamente, este sistema se organiza en dos niveles principales:

- El ***trigger* de Nivel-1** (L1T) se encarga de analizar parcialmente la información de los subdetectores más rápidos (los calorímetros y las cámaras de muones). Así es capaz de identificar en menos de 4 μs si un evento tiene indicios de ser interesante, es decir partículas con mucha energía o combinaciones inusuales. Actualmente, debido a las altas luminosidades con las que se trabaja, este nivel reduce la tasa de eventos a aproximadamente unos 110kHz. [22]
- Los eventos aceptados por el L1T son procesados a continuación por el ***High Level Trigger*** (HLT). Este sistema utiliza algoritmos más avanzados para realizar una reconstrucción más detallada del evento usando información del detector completo. De esta manera, actualmente, según se apunta en [22], se están reduciendo los eventos a unos 5kHz con este sistema.

Es importante destacar que el *trigger* no reduce el contenido de los eventos aceptados, sino que más bien, decide en tiempo real si un evento completo, con toda su información, es interesante para ser guardado. De esta manera, una vez seleccionados, los eventos registrados son enviados al sistema de almacenamiento para su posterior análisis. Debido al volumen masivo de información, CMS recurre a una infraestructura de computación distribuida a nivel mundial, conocida como ***Worldwide LHC Computing Grid*** (WLCG).

Esta red se organiza en varios niveles jerárquicos. El primero de ellos, el *Tier-0*, está situado en el CERN donde se realiza una reconstrucción inicial de los datos y se distribuyen al *Tier-1*, grandes centros internacionales encargados del almacenamiento seguro y reprocesado. Desde este último, los datos llegan a nosotros, los *Tier-2* y *Tier-3*, utilizados principalmente por grupos de investigación para realizar los análisis físicos. Concretamente, en el caso de Oviedo, contamos con el *Tier-ES-Oviedo*, ubicado en el edificio C3 del Campus de Mieres, que forma parte de esta infraestructura distribuida. Esto permite a miles de científicos acceder y analizar los datos de CMS de forma eficiente.

A pesar del sofisticado sistema de *trigger* para la reducción de eventos, la cantidad de información contenida en cada uno de ellos sigue siendo enorme. Ya se comentaba previamente que los haces están organizados en paquetes densos de protones llamados *bunches*. Esto hace que en cada cruce, no se produzca solamente una colisión entre dos protones de cada haz, sino que debido a la gran densidad de protones de cada *bunch*, pueden producirse múltiples colisiones simultáneamente. Este fenómeno, conocido como ***pile-up***, hace que en cada evento registrado no solo se recoja la información de una colisión principal que contenga la física interesante, sino también de muchas otras colisiones secundarias que se superponen espacial y temporalmente en los distintos subsistemas del detector.

Con los niveles actuales de luminosidad, el *pile-up* medio ronda las 60 colisiones por cruce [11]. Es decir, un solo evento puede contener huellas de más de 60 colisiones simultáneas, cada una de las cuales puede dar lugar a decenas de partículas cargadas. Además, debido al aumento progresivo de la luminosidad que ya se mostraba en Figura 2.2, y particularmente con la llegada del *High-Luminosity* LHC (HL-LHC), se esperará que se alcancen niveles de *pile-up* aún más elevados, del orden de 140–200 colisiones por cruce.

Esto genera un volumen de datos especialmente elevado en los distintos subdetectores como el *tracker*, que debe reconstruir cientos de trayectorias superpuestas. Esta complejidad puede verse en Figura 2.12 dónde se pueden apreciar la gran cantidad de trayectorias generadas en un solo cruce de haces. En este contexto, las técnicas de aprendizaje automático resultan de gran ayuda pues su capacidad para identificar patrones complejos en grandes volúmenes de datos los convierte en candidatos ideales para abordar este tipo de problemas.

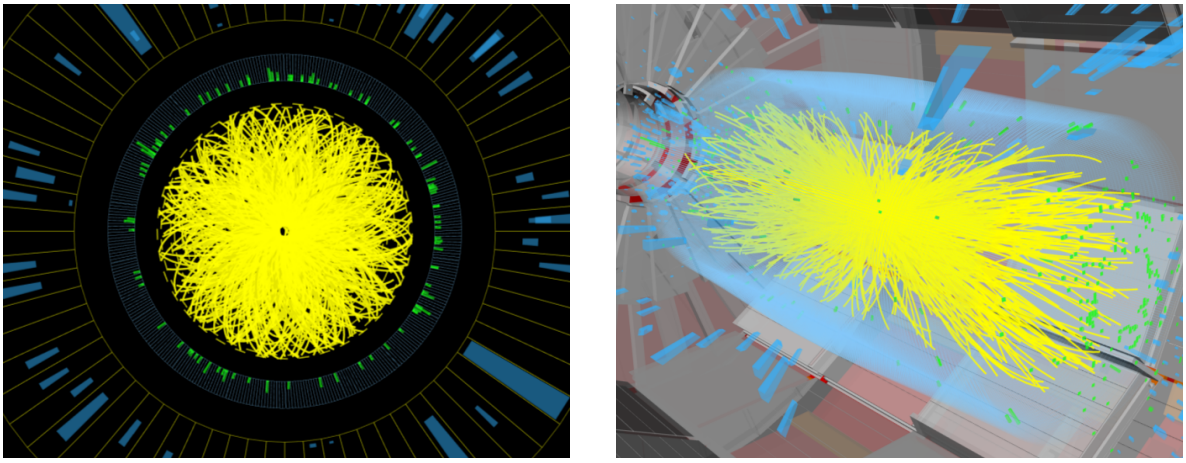


Figura 2.12: Visualizaciones de un evento en el experimento CMS con alto *pile-up*. A la izquierda se muestra una vista del plano $x - y$. A la derecha una vista oblicua en 3D. Las trayectorias de las partículas en el *tracker* se muestran en amarillo. Recuperadas de [24]

Capítulo 3

Aprendizaje automático

La enorme cantidad de información generada en cada colisión y la complejidad del entorno experimental hacen imprescindible el uso de técnicas avanzadas de procesamiento de datos. Es en este contexto dónde el aprendizaje automático ha logrado abrirse paso como una herramienta muy útil para afrontar los desafíos que supone la física de altas energías.

El aprendizaje automático o más comúnmente conocido como *machine learning* (ML) es una rama de la inteligencia artificial que permite a los sistemas informáticos aprender patrones y tomar decisiones a partir de unos datos concretos. De esta manera, a diferencia de los algoritmos tradicionales que siguen instrucciones definidas específicamente por el programador, los algoritmos de ML se entrenan sobre grandes conjuntos de datos para así identificar posibles patrones y realizar predicciones cuando se introducen ejemplos nuevos.

En general, los algoritmos de ML pueden clasificarse en distintas categorías según el tipo de datos con los que trabajan y la tarea que deben resolver. Entre ellos se encuentran el aprendizaje supervisado, el no supervisado y el aprendizaje por refuerzo. Concretamente en este trabajo se hará uso de **aprendizaje supervisado**, ya que como se expondrá con más detalle posteriormente, se dispone de un conjunto de ejemplos en los que se conoce tanto la información de entrada (*input*) como sus etiquetas de salida (*output*). Así, el objetivo del modelo será aprender a predecir la salida a partir de las entradas de manera que pueda generalizarlo a nuevos datos que no se han visto durante el entrenamiento.

3.1. Modelos de aprendizaje automático supervisado

Dentro del aprendizaje supervisado, los algoritmos pueden abordar principalmente dos tipos de problemas: de **regresión**, cuando el objetivo es predecir los datos de una variable continua como

puede ser una velocidad; o de **clasificación** cuando se trata de predecir una etiqueta discreta, es decir asignar a cada entrada una categoría determinada. Concretamente, para reconstruir las trayectorias del problema que se plantea se abordará un problema de clasificación binaria en el que se distinguirá entre dos clases posibles: `True` o `False`

Además se emplearám dos modelos de aprendizaje distintos: por un lado un ***Boosted Decission Tree***, que combina múltiples árboles de decisión para formar un modelo más robusto; y por otro, una **red neuronal profunda**, capaz de modelar relaciones más complejas.

3.1.1. Árboles de decisión

Los árboles de decisión son un tipo de modelo de aprendizaje supervisado. Su funcionamiento se basa en la división del conjunto de datos en subconjuntos cada vez más homogéneos, es decir, en los que las muestras pertenezcan mayoritariamente a una misma clase.

Su estructura, por tanto, es jerárquica y está formada por nodos de decisión y nodos hoja. De esta manera, cada nodo de decisión plantea una pregunta sobre una de las variables de entrada y cada rama representa una de las posibles respuestas. Para decidir cómo dividir los datos en cada nodo, el modelo selecciona la variable de entrada y el valor umbral que permiten separar de la forma más eficiente las clases presentes en el conjunto de datos. Como es lógico, el algoritmo busca minimizar la mezcla entre clases tras cada partición. Así, el proceso continúa hasta que alcanza un criterio de parada y se llega a los nodos hoja, que contienen la predicción final del modelo.

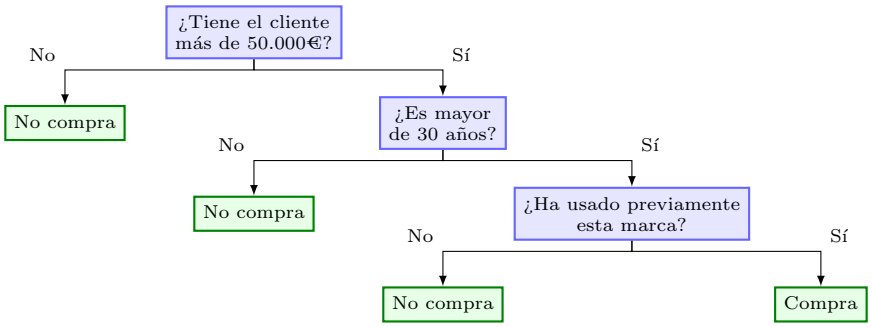


Figura 3.1: Ejemplo de árbol de decisión para la compra de un coche. Se muestran en color azul los nodos de decisión y en color verde los nodos hoja.

Un ejemplo muy sencillo de un árbol de decisión puede observarse en Figura 3.1. El árbol se construye sobre un conjunto de datos previamente etiquetado, en el que se sabe si el cliente finalmente compró el coche o no, y partir de esos datos, el algoritmo selecciona las preguntas que permiten separar mejor a los compradores de los no compradores. En este caso, para reducir la mezcla entre clases se han realizado

tres preguntas principales: el nivel de ingresos, la edad y su experiencia previa con la marca [26]. El objetivo del modelo es aprender patrones generales con esos datos para que luego pueda predecir el comportamiento de nuevos clientes, distintos a los que se usaron para construir el árbol.

Los árboles de decisión son muy fáciles de interpretar y rápidos de entrenar lo que los convierte en una herramienta muy útil para entender problemas de clasificación. No obstante, un único árbol suele tener una capacidad predictiva limitada, siendo necesario recurrir a modelos más complejos para tratar de manera adecuada los datos.

3.1.2. *Boosted Decision Trees*

Para superar las limitaciones de los árboles de decisión se utilizan técnicas de ensamblado como el *boosting* que combinan varios modelos débiles, como árboles de baja profundidad, para construir un modelo más potente capaz de superar el azar. Concretamente, uno de los algoritmos más utilizados es el *boosting* adaptativo o más comúnmente conocido como **AdaBoost**.

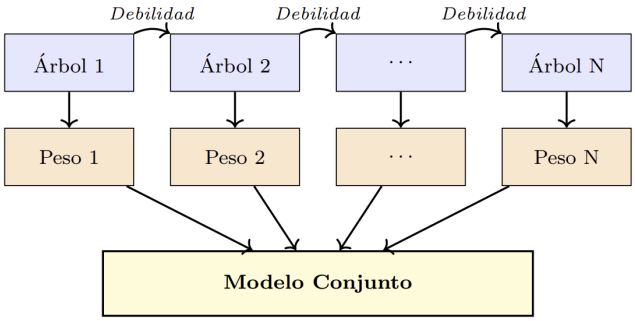


Figura 3.2: Esquema del funcionamiento de AdaBoost con árboles de decisión

Su funcionamiento parte de asignar el mismo peso a todas las muestras del conjunto de entrenamiento. A continuación, se entrena un primer árbol de decisión que tratará de clasificar correctamente dichas muestras. Para evaluar su rendimiento, se comparan las predicciones del árbol con las etiquetas reales de manera que si la clase predicha para una muestra no coincide con su clase verdadera, se considera mal clasificada. Estas muestras mal clasificadas contribuyen a un error ponderado, calculado como la suma de sus pesos, y este error determina cuánto influirá el árbol en la decisión final del modelo.

Una vez evaluado el rendimiento de un árbol, los pesos de las muestras clasificadas incorrectamente se incrementan de manera que el siguiente árbol se centre más en ellas. Este procedimiento se repite iterativamente, generando una sucesión de árboles que van corrigiendo los errores cometidos por los anteriores. Una característica muy importante en este enfoque es que cada árbol se entrena una única vez y no se modifica posteriormente, es decir, no hay un proceso de corrección sobre los árboles

anteriores. Así, una vez alcanzado el número de iteraciones que se desean, el proceso finaliza y las predicciones de todos los árboles se combinan de manera ponderada dónde cada árbol contribuye según su precisión dando lugar a un clasificador más robusto que los árboles individuales.

Cuando se emplea AdaBoost utilizando árboles de decisión como clasificadores, el modelo resultante recibe el nombre de ***Boosted Decision Tree*** (BDT).

3.1.3. Redes neuronales artificiales

Las redes neuronales artificiales o *artificial neural networks*(ANN) son un tipo de modelo de aprendizaje automático inspirado en la estructura y funcionamiento de las redes neuronales biológicas como las del cerebro humano. Están formadas por unidades o nodos conectados entre sí conocidos como neuronas artificiales. Habitualmente estas neuronas se agrupan en estructuras llamadas capas, donde: la primera capa se llama **capa de entrada** y es la que recibe directamente los datos del problema; la última capa es la **capa de salida** encargada de devolver la predicción del modelo; y entre ellas hay **capas ocultas** que procesan la información internamente.

De esta manera cuando una red neuronal incluye varias capas ocultas, se denomina **red neuronal profunda** (*deep neural network* o DNN). Un ejemplo de ello puede observarse en Figura 3.3.

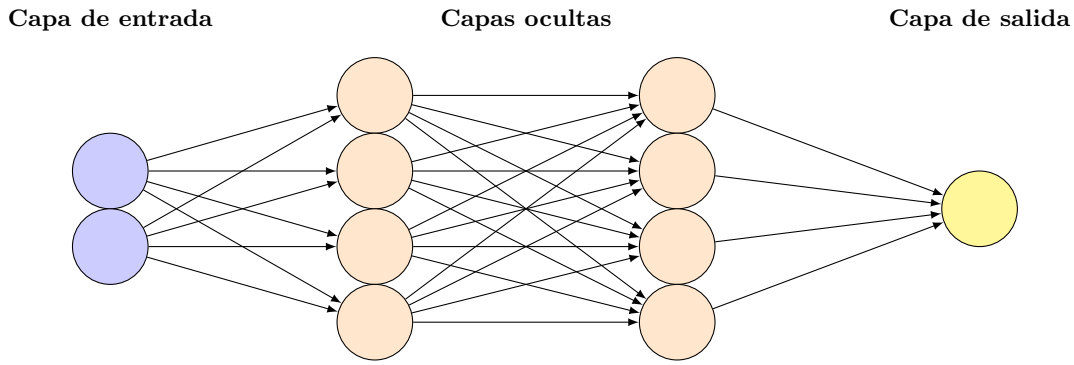


Figura 3.3: Arquitectura de una red neuronal profunda en la que se representan las neuronas de la capa de entrada, de salida y las ocultas en colores azul, amarillo y naranja respectivamente

En una red neuronal típica como las que se usarán, la información fluye hacia adelante desde la capa de entrada hasta la capa de salida, pasando por las capas ocultas intermedias. Además cada neurona de una capa está conectada con todas las de la siguiente capa, de manera que se las conoce como redes completamente conectadas.

Internamente, cada conexión entre neuronas lleva asociado un valor numérico denominado **peso**, que

indica la importancia relativa de cada entrada. Así, cuando una neurona recibe señales desde la capa anterior, lo que hace es calcular una suma ponderada de todas las entradas, es decir, multiplica cada una de ellas por su correspondiente peso y luego las suma. Además, a esta suma se le añade un valor adicional llamado **sesgo**, que no depende de ninguna entrada concreta y permite que la red aprenda incluso cuando las entradas son nulas o muy pequeñas.

Una vez obtenida esta combinación ponderada, el resultado se transforma mediante una **función de activación** que introduce una no linealidad entre entradas y salidas que permite a la red modelar relaciones más complejas. Las más habituales aparecen representadas en Figura 3.4 y son: la sigmoide, que comprime los valores de entrada en el intervalo (0, 1) y es útil para problemas de clasificación binaria; y la *ReLU* (Rectified Linear Unit), que devuelve un cero para valores negativos y deja pasar sin modificar los positivos, favoreciendo un entrenamiento más eficiente.

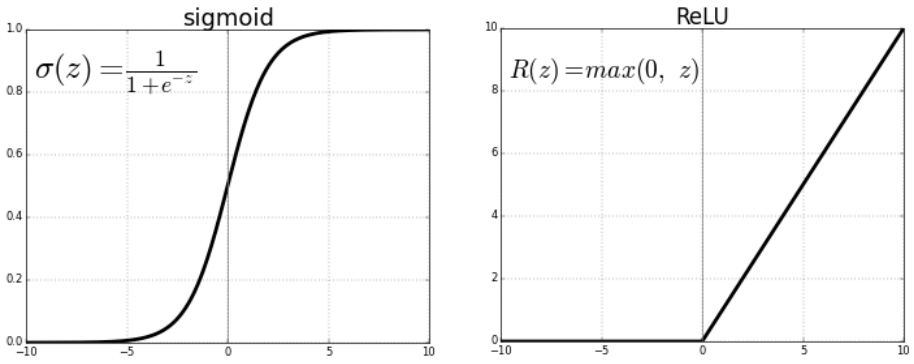


Figura 3.4: Funciones de activación típicas de redes neuronales. A la izquierda la función sigmoide acotada entre 0 y 1. A la derecha la función ReLU que crece indefinidamente. Recuperada de [29]

Este mecanismo se repite en cada una de las neuronas de todas las capas de la red y permite una transformación progresiva de la información desde la entrada hasta la salida. Además, durante el entrenamiento, la red va ajustando progresivamente sus pesos y sesgos con el objetivo de minimizar el error que comete en las predicciones. Este proceso implica varias pasadas completas sobre el conjunto de datos, conocidas como **épocas**. De esta manera, en cada época, el modelo procesa todos los ejemplos una vez, y tras cada pasada, los parámetros se ajustan para mejorar el rendimiento de la red.

Aunque cada operación individual es matemáticamente sencilla, el conjunto de transformaciones que se produce a lo largo de la red puede llegar a ser muy complejo y difícil de interpretar, lo que hace que a menudo se hable de las redes neuronales como 'cajas negras' [30].

3.2. Control del aprendizaje de los modelos

Una vez definidos los modelos de aprendizaje supervisado, el siguiente paso fundamental consiste en entrenarlos adecuadamente. Para ello, lo habitual es dividir el conjunto de datos en dos subconjuntos: un **conjunto de entrenamiento** más grande; y un **conjunto de prueba** más reducido para evaluar qué tan bien el modelo ha aprendido patrones útiles y es capaz de generalizarlos a situaciones que no conoce.

En este contexto, se habla de **bajo entrenamiento** (*underfitting*) cuando el modelo es demasiado simple y no consigue distinguir correctamente las relaciones que existen entre los datos incluso en el conjunto de entrenamiento. Por otro lado, el **sobreentrenamiento** (*overfitting*) ocurre cuando un modelo es demasiado complejo y se ajusta demasiado a los datos de entrenamiento, incluyendo también ruido o posibles excepciones particulares. Esto lleva a que su rendimiento con los datos de la muestra de prueba no sea bueno ya que no ha sido capaz de aprender patrones generales, sino detalles específicos de ejemplos que ya conocía.

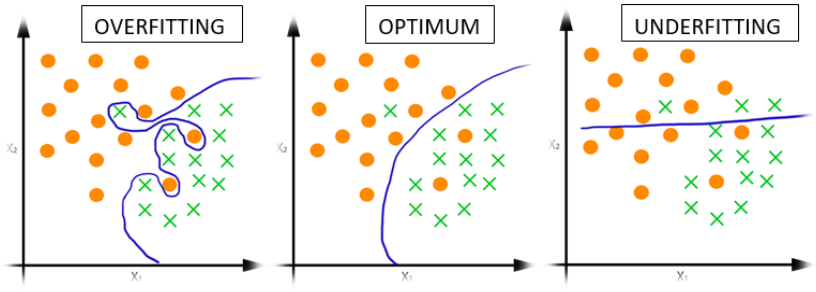


Figura 3.5: Ejemplos de un modelo sobreentrenado, óptimo y poco entrenado. Recuperado de [31]

En algunos modelos, como las redes neuronales, se suele reservar una parte del conjunto de entrenamiento como **conjunto de validación** que permita controlar el rendimiento del modelo durante el proceso de aprendizaje y detectar posibles sobreajustes a los datos de entrenamiento. No obstante, en modelos como los BDTs, este subconjunto no existe pues la evaluación se hace directamente sobre el conjunto de prueba una vez terminado el entrenamiento.

Además para reducir el riesgo de sobreentrenamiento se utilizan diversas técnicas de regularización. La más conocida en redes neuronales es el ***dropout***, que consiste en desactivar aleatoriamente una fracción de las neuronas durante cada época del entrenamiento. De esta manera, al desconectar ciertas conexiones, se obliga al modelo a no depender de combinaciones específicas de neuronas, evitando que la red memorice patrones concretos. Al finalizar el entrenamiento todas las neuronas se reactivan de nuevo.

3.3. Evaluación de los modelos

3.3.1. La matriz de confusión

Una de las herramientas para evaluar el rendimiento de un modelo de clasificación en aprendizaje supervisado es la **matriz de confusión** que compara las predicciones del modelo con las etiquetas reales del conjunto de prueba. Para ello se representa como una tabla cuadrada donde cada fila corresponde a la clase real y cada columna a la clase predicha [32]. Para un caso binario como el que se trata, los elementos de la matriz se interpretan según:

- **Verdaderos positivos (TP)**: casos en los que el modelo predice la clase positiva y acierta.
- **Falsos positivos (FP)**: casos en los que el modelo predice la clase positiva y se equivoca, es decir, la clase es realmente negativa.
- **Falsos negativos (FN)**: casos en los que el modelo predice la clase negativa y se equivoca, es decir, la clase es realmente positiva.
- **Verdaderos negativos (TN)**: casos en los que el modelo predice la clase negativa y acierta.

La matriz de confusión con la que se tratará tiene entonces la forma:

		Clase predicha
Clase real	Negativa	Positiva
Negativa	TN	FP
Positiva	FN	TP

Figura 3.6: Matriz de confusión para un problema de clasificación binaria

Es importante tener en cuenta que, en general, se habla de clase positiva como aquella que representa el evento de interés, mientras que la clase negativa representa la ausencia de ese evento. Para seguir la notación que se muestra en Figura 3.6, en este trabajo se considerará la clase `True` como la clase positiva, y la clase `False` como la clase negativa.

3.3.2. Métricas de evaluación global

Una forma más intuitiva y directa de visualizar la información que aporta la matriz de confusión puede hacerse recurriendo a métricas globales que resuman sus elementos dando una medida del rendimiento del modelo. Concretamente, las más destacadas se muestran a continuación:

La ***accuracy* o exactitud** mide el porcentaje total de predicciones correctas que realiza el modelo sobre el total de las muestras, es decir:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.1}$$

Relacionado con esta última, también se puede hablar de la fracción de muestras que el modelo clasifica incorrectamente. Esto se conoce como **error** y puede calcularse como $1 - \text{accuracy}$.

3.3.3. La curva ROC

En muchos modelos de clasificación, como las redes neuronales o los modelos de *boosting*, la salida no está formada por unos valores discretos, sino que lo que se devuelve es una probabilidad comprendida entre 0 y 1 de que la muestra pertenezca a una clase u otra. De esta manera para poder clasificarla en sus valores binarios, se define un **umbral de decisión** (k). De esta manera, si la probabilidad predicha es mayor que k, se clasifica como positiva, mientras que si es menor será negativa

El valor de k se puede ajustar como se desee. No obstante hay que tener cuidado pues por ejemplo, un valor de k más bajo hace que sea más fácil que una muestra se clasifique como positiva, pero también se corre el riesgo de clasificar como positivas muestras que no lo son (falsos positivos). Así, para analizar el efecto de variar este umbral de decisión sobre el rendimiento del modelo se recurre a la **curva *Receiver Operating Characteristic* (ROC)**.

Concretamente, el eje vertical representa la **tasa de verdaderos positivos (TPR)**, que mide la proporción de positivos correctamente clasificados por el modelo como positivos, es decir:

$$\text{TPR} = \frac{\text{TP}}{\text{FN+TP}} \tag{3.2}$$

Mientras que el eje horizontal representa la **tasa de falsos positivos (FPR)** encargada de medir la proporción de negativos que han sido incorrectamente clasificados por el modelo como positivos:

$$\text{FPR} = \frac{\text{FP}}{\text{FP+TN}} \tag{3.3}$$

Luego cada punto de la curva ROC corresponde a un valor específico del umbral k, con su correspondiente TPR y FPR. Un modelo ideal se aproximaría a la parte superior izquierda de la gráfica, es decir, un valor alto de TPR manteniendo baja la FPR. Mientras que un modelo sin capacidad de discriminación entre clases, como una clasificación aleatoria, daría lugar a una curva que se mantendría cercana a la diagonal.

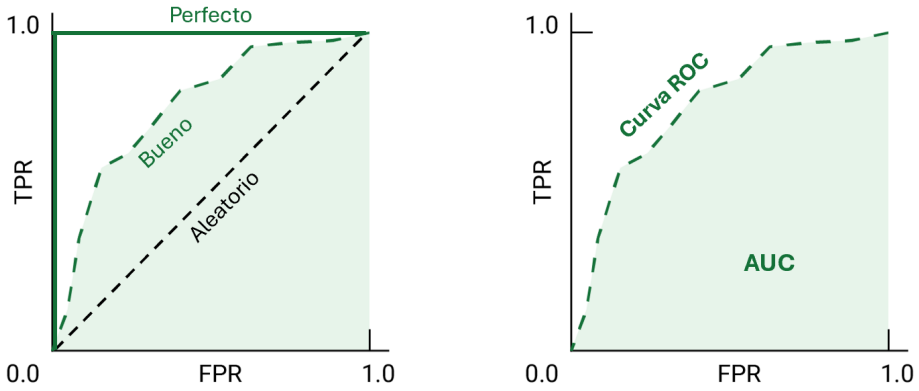


Figura 3.7: Ejemplos de curvas ROC. En la imagen de la izquierda se comparan tres modelos. En la imagen de la derecha se muestra el AUC sombreado bajo la curva ROC. Modificado de [33]

Para resumir la información que aporta la curva ROC se suele usar el ***Area Under the Curve* (AUC)**, que representa el área bajo la curva ROC y proporciona una medida del rendimiento global del modelo independientemente del valor de umbral de decisión utilizado. Concretamente, el AUC toma valores entre 0 y 1 de manera que cuánto mas cercano sea este a 1, el rendimiento del modelo a la hora de clasificar clases será mejor.

Capítulo 4

Clasificación de dobletes para la reconstrucción de trayectorias

La reconstrucción de trayectorias de partículas cargadas en un detector de trazas como el del experimento CMS, es una de las tareas fundamentales en el análisis de eventos de colisiones protón-protón, pues permite conocer los primeros pasos de las partículas tras la interacción. Esta información resulta esencial para identificar las partículas producidas o estudiar los procesos físicos que han tenido lugar en la colisión.

El primer paso en este proceso consiste en transformar los datos obtenidos directamente de la electrónica de lectura (*raw data*) en *digis* que representan los píxeles activados por el paso de partículas cargadas. Estos *digis* se agrupan formando clústeres, es decir, conjuntos de píxeles vecinos activados por la misma partícula. A continuación se forman dobletes que posteriormente se agruparán en N-tupletes con los que posteriormente se reconstruirán las primeras trayectorias de píxeles. Finalmente, se identifican los vértices de interacción agrupando las trayectorias cuyo punto de origen es común, lo que permite determinar el lugar exacto de la colisión o de una posible desintegración secundaria. [34]

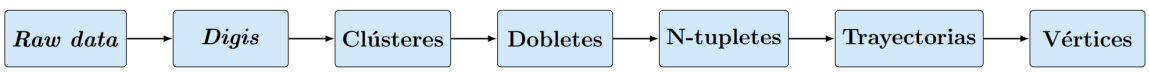


Figura 4.1: Diagrama del proceso de reconstrucción de trayectorias en el detector de píxeles de CMS

En este trabajo se centra la atención en una etapa específica de la reconstrucción: la clasificación de dobletes. Se considerará que un **doblete** está formado por dos *hits* o impactos en distintos puntos del detector de píxeles de silicio, que podrían haber sido producidos por una misma partícula cargada en

su trayectoria tras la colisión. Sin embargo, en cada evento se generan miles de *hits*, lo que da lugar a un número enorme de combinaciones posibles entre capas del detector. Como consecuencia, muchos de los dobletes formados no reflejan trayectorias reales, sino que surgen de combinaciones de *hits* no relacionados entre sí, ya sea por provenir de diferentes partículas o por efecto del ruido del detector.

El objetivo entonces será discriminar entre dobletes generados por una misma partícula, que se catalogarán como verdaderos (`True`); y dobletes falsos (`False`) debidos a combinaciones entre *hits* no vinculados físicamente.

Para abordar este problema, se propone el uso de técnicas de aprendizaje automático supervisado, buscando un modelo capaz de predecir a partir de las características geométricas y topológicas de un doblete, si este pertenece o no a una trayectoria real.

4.1. Muestra de datos

Para tratar el problema propuesto, se ha utilizado una muestra simulada de eventos del experimento CMS, accesible a través del portal de datos abiertos del CERN [35]. Dicha muestra fue generada mediante técnicas de simulación MonteCarlo que permiten modelar de forma probabilística tanto la colisión como la respuesta del detector. Además, el proceso se hace bajo condiciones controladas en las que el funcionamiento del detector es ideal. Este enfoque es indispensable en problemas que usan modelos de clasificación supervisada, ya que solo mediante simulación es posible conocer con certeza el origen de cada *hit* y por tanto, determinar si un doblete ha sido realmente `True` o `False`.

Concretamente, la muestra original está compuesta por aproximadamente 20 000 eventos simulados de colisiones protón-protón con una energía en el centro de masas de 13 TeV, en condiciones similares a las del *Run* 2 del LHC. En cada uno de esos eventos, se ha forzado la producción de un par quark top-antitop ($t\bar{t}$) acompañado además de un *pile-up* de 50 colisiones simultáneas adicionales que simulan el entorno real del detector. Los quarks top son las partículas más masivas conocidas actualmente, lo que las hace extremadamente inestables y provoca que se desintegren rápidamente en algunas de las partículas que se muestran en Figura 1.1. Esto hace que en cada evento se tenga una amplia representación de todas las partículas no neutras explicadas por el Modelo Estándar.

A partir de esta muestra completa, se han extraído alrededor de **1 millón de dobletes** balanceados entre verdaderos y falsos. Cada entrada representa un doblete individual y contiene tanto información del evento global como las propiedades de los dos *hits* que lo componen. De esta manera, en cada doblete, se distingue entre: el ***hit* de entrada** (`in`) que se encuentra más cerca del punto de interacción; y el ***hit* de salida** (`out`) localizado en capas más externas que el anterior.

Para cada uno de estos *hits*, se registran sus coordenadas espaciales en el detector, tanto en el sistema cartesiano (`X`, `Y`, `Z`) como en el cilíndrico (`Phi`, `R`). Además se incluyen otras variables asociadas al clúster de píxeles como su carga media (`AvgCharge`), tamaño (`ClustSize`), forma (`ClustSizeX`, `ClustSizeY`, `Skew`), la posición de su centro (`ClustX`, `ClustY`) y su orientación principal en el plano del sensor (`Ax1`, `Ax2`). Por otro lado, también destaca `IsBarrel` que indica si el *hit* se encuentra en la región del *barrel* o en los *endcaps*.

Cada una de estas variables se encuentra duplicada con los prefijos `in` y `out`, según si se refieren al *hit* de entrada o de salida del doblete. Además, cada doblete incluye una etiqueta (`label`) que indica si ha sido clasificado como `True` o `False`. Es importante notar que esto es un pequeño resumen simplemente de algunas de las variables más destacables, no obstante, toda la información detallada sobre el conjunto completo puede consultarse en la bibliografía correspondiente [35].

4.1.1. Visualización exploratoria

Para familiarizarse con la estructura de datos y visualizar mejor las diferencias entre dobletes `True` y `False`, se ha realizado una representación sobre el plano $r - z$ del detector. De esta manera, a raíz de Figura 4.2 se puede observar cómo los dobletes verdaderos tienden a conectar *hits* cercanos que siguen trayectorias suaves entre capas consecutivas, mientras que, por otro lado, los dobletes falsos tienden a conectar capas lejanas o incluso regiones diferentes del detector, como el *barrel* y los *endcaps*.

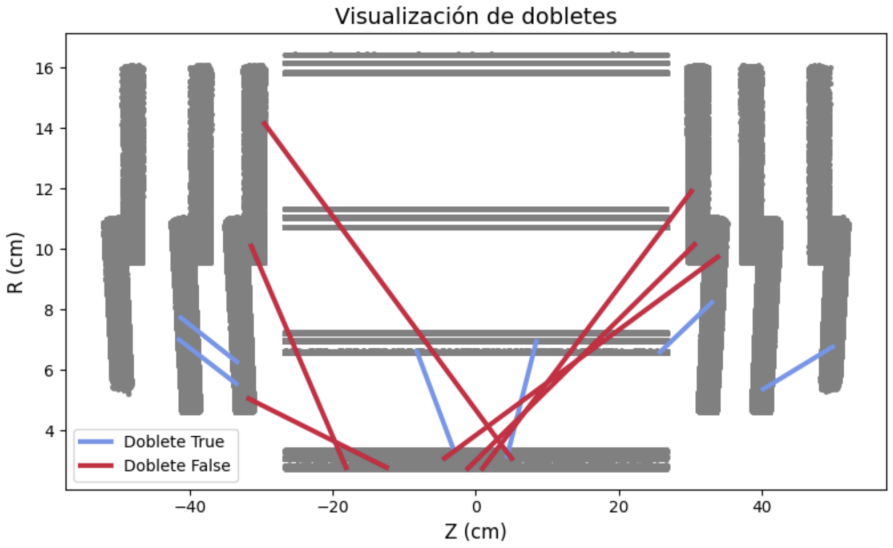


Figura 4.2: Visualización de algunos dobletes en la proyección $r - z$. Se representan en azul los dobletes `True`, y en rojo los `False`. Las zonas en gris son las distintas capas o discos del píxel

El fuerte campo magnético del detector orientado a lo largo del eje z provoca que las partículas cargadas

describan trayectorias helicoidales, girando en el plano transverso en torno a este eje longitudinal. Sin embargo, debido al alto momento transverso con el que se producen muchas de estas partículas tras las colisiones, la curvatura de las trayectorias en dicho plano es bastante pequeña [1]. En este trabajo, al construir dobletes a partir de pares de *hits*, solo se dispone de dos puntos por trayectoria, lo que impide mostrar una representación gráfica de la hélice completa o cualquier curvatura. No obstante, es posible inferir información sobre esto a partir de la diferencia en el ángulo ϕ entre el *hit* de entrada y el de salida ($\Delta\phi$). De esta manera, al representar la distribución para ambos tipos de dobletes, en Figura 4.2, se observa que los `True` tienden a concentrarse más en torno a $\Delta\phi = 0$ en contraste con la distribución más dispersa de los dobletes falsos.

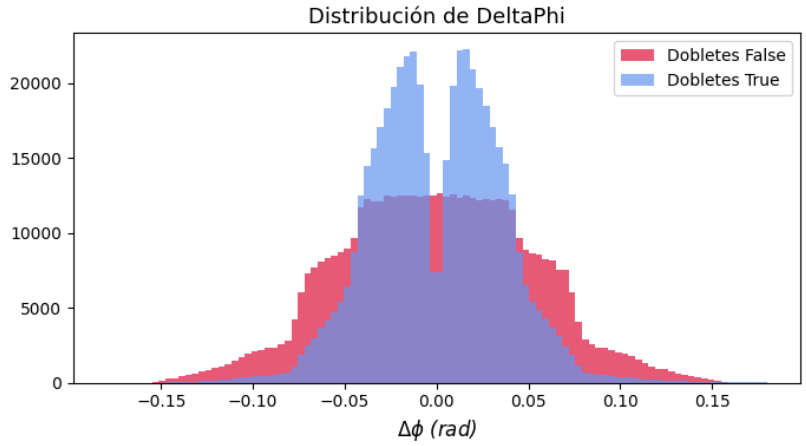


Figura 4.3: Distribución de $\Delta\phi$ para dobletes falsos (en rojo) y verdaderos (en azul)

4.1.2. Selección de variables de entrada

La elección de las variables de entrada que usarán los modelos para aprender y poder generalizar los datos, no se ha hecho de forma arbitraria, sino que ha sido resultado de un análisis cuidadoso del conjunto de observables de la muestra. En primer lugar, se consideraron de manera lógica aquellas variables directamente relacionadas con la posición de los *hits* en el detector, así como las propiedades del clúster asociadas a cada uno de ellos que ya se describían previamente.

A partir de esta base, se exploraron también las demás variables de la muestra con el objetivo de evaluar si podían aportar una mejora a la hora de distinguir entre ambos tipos de dobletes. Para ello, se representaron las distribuciones para ambos tipos de dobletes y se observó si existían diferencias apreciables entre ambas. Como ejemplo ilustrativo de ello, en Figura 4.4 se comparan las distribuciones

[1] El radio de curvatura en el plano transversal está relacionado con el momento transverso por $R = \frac{p_T}{qB}$.

de dos variables: `inZ`, que muestra una clara diferencia entre dobletes verdaderos y falsos; e `inPhi`, cuya distribución es prácticamente indistinguible entre ambas clases. No obstante, el estudio de todas las variables puede visualizarse en Apéndice A.

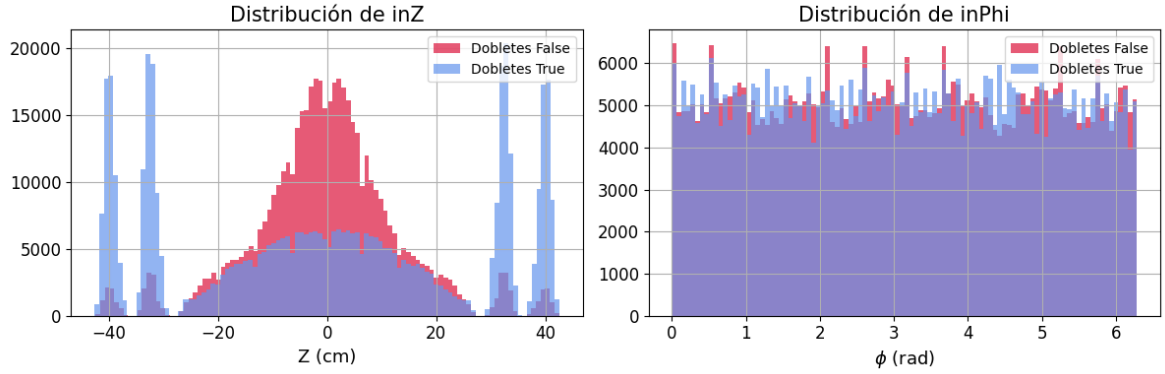


Figura 4.4: Ejemplo ilustrativo de las distribuciones para dobletes `True` y `False` en dos variables. A la izquierda `inZ`, discriminatoria. A la derecha `inPhi`, no discriminatoria.

No obstante, muchas de estas variables no mostraron diferencias significativas entre ambas clases, ni contribuyeron a mejorar los resultados de los modelos, por lo que se descartaron. Esto lleva a que, en total, se haga uso de: las coordenadas espaciales en ambos sistemas de referencia (`X`, `Y`, `Z`, `R`), las propiedades del clúster (`ClustX`, `ClustY`, `ClustSize`, `ClustSizeX`, `ClustSizeY`), su orientación en el plano del sensor (`Ax1`, `Ax2`), su asimetría (`Skew`), su carga media (`AvgCharge`) y una variable categórica que indica si el *hit* pertenece al *barrel* o a los *endcaps* (`IsBarrel`). Es decir, 14 variables que aparecen duplicadas para el *hit* de entrada y el de salida, a lo que se añade la variable `DeltaPhi`, lo que da lugar a un total de **29 variables de entrada**.

4.2. Resultados de la clasificación

Una vez descrita la muestra, durante este apartado se procede a entrenar y evaluar distintos modelos de clasificación para la catalogación de dobletes `True` o `False`. En concreto, se compararán los resultados obtenidos por un ***Boosted Decision Tree*** (BDT) y una **red neuronal profunda** (DNN).

Para el entrenamiento y evaluación de ambos clasificadores, los datos se han dividido en dos subconjuntos, con una proporción del $70 - 30\,\%$ respectivamente. Además a cada doblete se le ha asignado una etiqueta binaria: `1` en caso de que sean de clase `True`; y `0` para los `False`, permitiendo así plantear el problema como una tarea de clasificación binaria supervisada.

4.2.1. Resultados BDT

El primer modelo considerado es un algoritmo de AdaBoost que usa como clasificadores débiles árboles de decisión, es decir, como ya se explicó previamente, se trata de un ***Boosted Decision Tree***. Concretamente, el clasificador se ha construido usando un total de 100 árboles de decisión, cada uno con una profundidad máxima de 3 niveles. La justificación detallada de esta elección se abordará posteriormente.

Una vez entrenado el modelo, el histograma de Figura 4.5 muestra la distribución de salida del modelo, es decir, las probabilidades predichas por el BDT de que cada doblete sea verdadero. De esta manera, el modelo asigna probabilidades más altas a los dobletes verdaderos, mientras que asigna probabilidades bajas a los dobletes falsos, pues es capaz de reconocer que tienen características que no se corresponden con la clase `True`. Es decir, la separación entre ambas clases indica que el modelo ha aprendido a distinguir entre ambos tipos de dobletes.

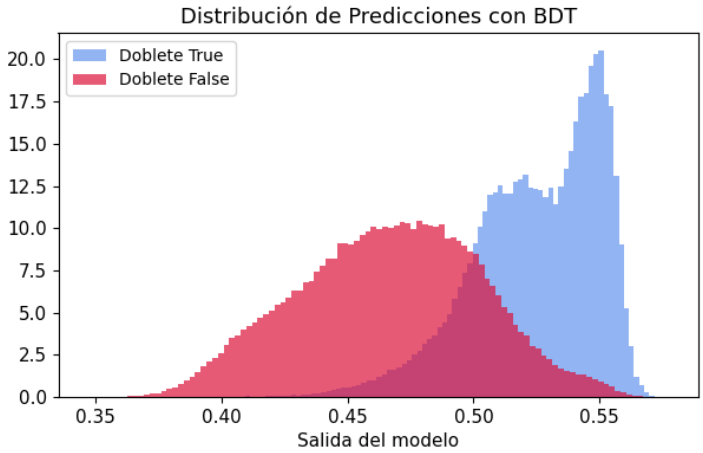


Figura 4.5: Distribución de las probabilidades predichas por el BDT para dobletes `True` y `False`. Las probabilidades representan la estimación del modelo de que un doblete pertenezca a la clase `True`

Resulta interesante resaltar que la distribución asociada a la clase `True` presenta dos máximos claramente diferenciados. La aparición de este doble pico puede estar causada porque el modelo esté diferenciando entre dos subpoblaciones con características ligeramente distintas dentro de esta clase, a las cuales asigna distintos niveles de confianza. Esta respuesta no necesariamente puede estar causada por una separación física real de los datos, sino que puede estar relacionada con las limitaciones naturales del modelo. Concretamente, debido a que carece de un mecanismo explícito de corrección de errores y a su estructura basada en decisiones mediante partición por umbrales, el BDT tiende a generar salidas más discretizadas y menos suaves que otros algoritmos más robustos.

Es importante destacar que se intentó identificar el conjunto de variables responsables de esta separación, pero no se logró establecer una causa clara. Es posible que esta respuesta del modelo surja de correlaciones sutiles entre múltiples variables difíciles de interpretar a simple vista. En cualquier caso, este comportamiento refuerza la idea de que su capacidad para modelar relaciones complejas es limitada.

No obstante, a continuación se discutirán los resultados obtenidos por el modelo. Para ello, se calculó la matriz de confusión (4.1) cuando se aplica un umbral de decisión de 0.5 sobre la salida:

$$\begin{pmatrix} 124\,056 & 28\,265 \\ 23\,897 & 128\,858 \end{pmatrix} \tag{4.1}$$

Dónde la fila superior está asociada a los dobletes `False` y la fila inferior hace referencia a los `True`. Es decir, las entradas en la diagonal muestran que, aunque la gran mayoría de los dobletes son clasificados correctamente, existe un número no despreciable de malas clasificaciones, tanto para dobletes verdaderos como falsos. Con todo, el modelo es capaz de alcanzar una **exactitud del 82.9** % al aplicar un umbral de decisión de 0.5, indicando que más del 80 % de los dobletes del conjunto de *test* fueron correctamente clasificados, siendo por ende el error del 17.1 %.

Por otro lado, para evaluar el rendimiento del modelo de forma independiente del umbral de decisión, se representa a continuación en Figura 4.6 la curva ROC correspondiente.

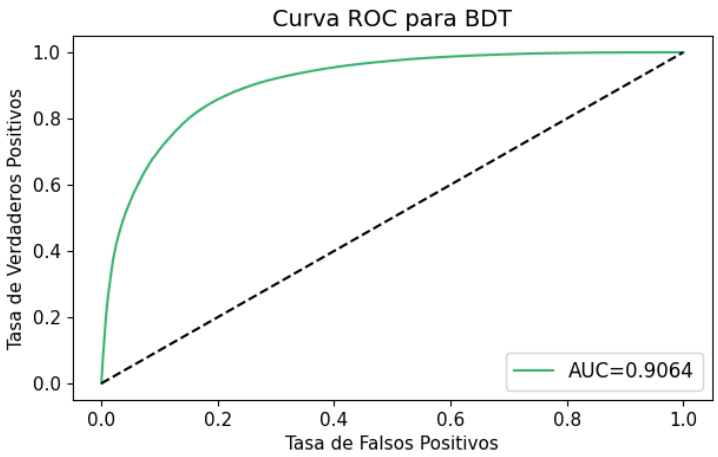


Figura 4.6: Curva ROC para el modelo BDT. Se representa la diagonal del azar mediante una linea punteada.

Con ello se observa que la curva ROC se mantiene por encima de la diagonal para todo valor del umbral de decisión. Esto confirma, que a pesar de lo comentado previamente, el modelo logra una separación

eficaz entre las dos clases, con un **AUC de 0.91**.

Por otro lado, es importante comentar que para determinar el número óptimo de árboles que se usarían para entrenar el modelo de AdaBoost, fue necesario realizar un análisis previo de validación en el que se fue variando este parámetro y evaluando el rendimiento en el conjunto de *test* con diferentes métricas. Este análisis se muestra en Figura 4.7 donde puede verse que las métricas de *test* mejoran considerablemente hasta 100 árboles, punto a partir del cual las mejoras son insignificantes.

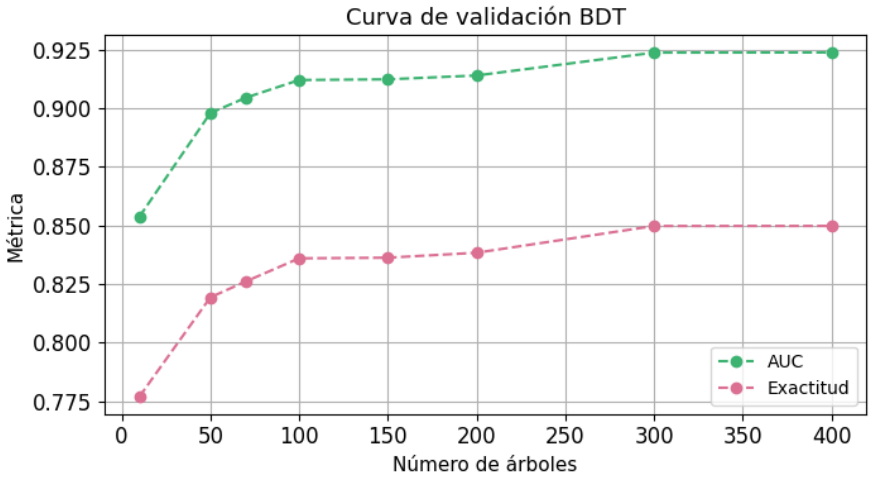


Figura 4.7: Evaluación de la exactitud y AUC cuando se varía el número de árboles del BDT

Hay que tener en cuenta además que cuánto mayor es el número de árboles que utilice el modelo, mayor es el tiempo de ejecución. Los valores exactos se muestran a continuación:

Nº Árboles	**Tiempo (s)**
10	86
50	404
70	574
100	810

Nº Árboles	**Tiempo (s)**
150	1202
200	1590
300	2396
400	3473

Tabla 4.1: Tiempo de entrenamiento de BDT según número de árboles

Es por ello, que para obtener los resultados que se mostraban anteriormente, se haya optado por fijar **el número de árboles en 100**, pues aunque valores superiores podrían ofrecer una mínima mejora, el tiempo de entrenamiento se incrementa de manera considerable, pasando de unos 13 minutos con 100 árboles a casi 1 hora con 400 árboles.

Además, se ha comprobado que por muy elevado que sea el número de árboles, el modelo **no se sobreajusta**. Para ello, se han evaluado las distintas métricas en el conjunto de entrenamiento y se

ha observado que estas evolucionan paralelas a las del conjunto de *test* (puede verse en Apéndice B). Esto es fundamental, pues si el modelo presentase métricas superiores en entrenamiento que en test, implicaría que memorizó los datos de entrenamiento pero no es capaz de generalizarlos.

Una de las principales razones que explican este buen comportamiento es la poca profundidad que se ha usado en los árboles de decisión, lo que limita la capacidad de sobreajuste de cada árbol y favorece la generalización del modelo.

4.2.2. Resultados DNN

El otro método utilizado en la clasificación binaria de dobletes fue una **red neuronal profunda** (DNN) que recibe de nuevo como entrada las mismas variables que se usaron durante el entrenamiento del BDT. En concreto, la capa de entrada está compuesta por 29 neuronas, una por cada variable utilizada para describir los dobletes.

En este caso fue necesario además hacer un preprocesamiento de los datos de entrada. Concretamente, se transformó cada variable de entrada restándole su valor medio y dividiéndola por su desviación típica de manera que no se modificasen las distribuciones relativas de los datos, pero sí se produjese un reescalado necesario para que variables con rangos muy distintos (como `AvgCharge`) no dominasen el aprendizaje frente a otras con valores numéricamente más pequeños. Es importante notar que este reescalado no se utilizó en el caso del BDT ya que los árboles de decisión se basan en comparaciones de umbrales y no en pesos como las DNN.

Así, la red cuenta con dos capas ocultas de 64 neuronas cada una. Además para mejorar la capacidad de generalización del modelo, se aplica un *dropout* entre ambas capas ocultas, que hace que durante el entrenamiento se apaguen aleatoriamente un 20 % de las neuronas conectadas. Finalmente, la capa de salida está formada por una única neurona con función de activación sigmoide encargada de transformar el valor de salida en un número entre `0` y `1`, que puede interpretarse como la probabilidad de que el doblete pertenezca a la clase `True`.

El modelo fue entrenado durante un máximo de 40 épocas, imponiendo un criterio de parada si la exactitud sobre el conjunto de validación no mejoraba tras 10 épocas consecutivas. Concretamente la evolución de esta exactitud en cada época se muestra a continuación en Figura 4.8.

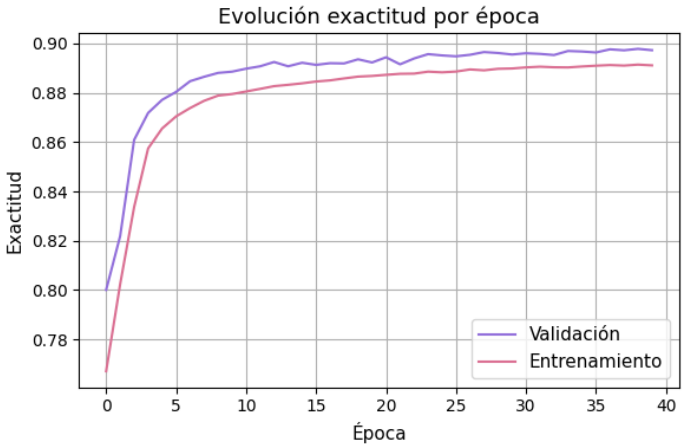


Figura 4.8: Evolución de la exactitud durante el entrenamiento de la DNN. Se muestran los resultados tanto para el conjunto de validación como para el de entrenamiento

Se observa que ambas curvas tienen una evolución cercana entre sí, sugiriendo que el modelo **no está sobreentrenado**. De hecho, la exactitud sobre el conjunto de validación crece ligeramente por encima que en el de entrenamiento, lo cual está causado por la implementación del *dropout* que actúa unicamente durante el entrenamiento. No obstante, durante la validación y el *test*, la red recupera todas sus neuronas y opera de manera 'completa', llevando a esta ligera mejora de su rendimiento.

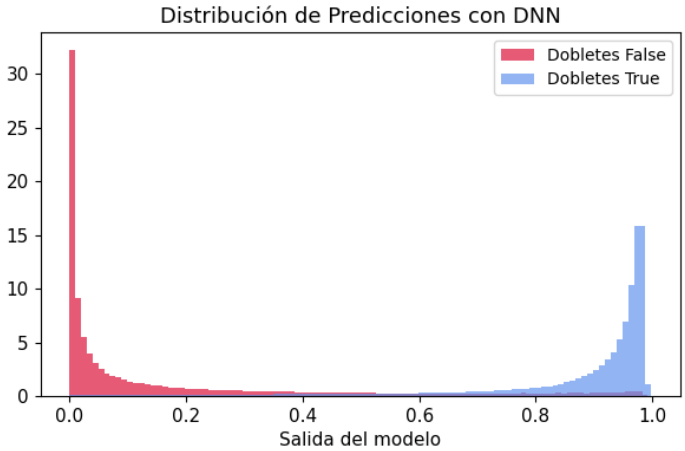


Figura 4.9: Distribución de las probabilidades predichas por la DNN para dobletes `True` y `False`.

Una vez entrenado el modelo, se evaluó su rendimiento sobre el conjunto de prueba. Concretamente, en Figura 4.9 se muestra la distribución de probabilidad de salida de la red para ambos tipos de dobletes. De nuevo, se observa una buena separación de clases, con acumulaciones de clases cercanas a `1` para los dobletes `True` y próximas a `0` para `False`.

En este punto resulta importante notar las claras diferencias visuales entre la salida de este modelo y

la del BDT que se veía en Figura 4.5. Resulta evidente que mientras el BDT tiende a predecir valores en un rango más estrecho de la zona intermedia, la DNN produce salidas más extremas, próximas a `0` o `1`. Esta diferencia se debe a la naturaleza de cada modelo. Por un lado, las modelos basados en árboles como los BDT, suelen producir valores más concentrados al combinar muchas decisiones binarias, mientras que las DNN, al contar con funciones de activación como la sigmoide en la capa de salida, están diseñadas para forzar la salida a los extremos.

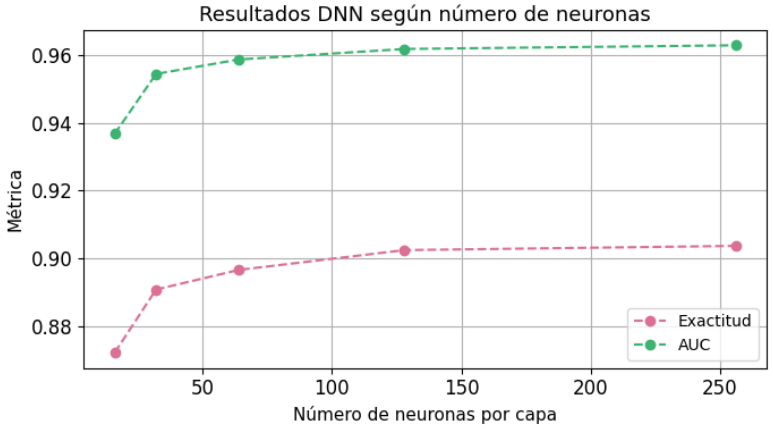
Además, se construyó de nuevo la matriz de confusión (4.2) a partir de la umbralización en 0.5 de las probabilidades de la red. Con ello, el modelo alcanzó un **89.7 % de exactitud** y por ende solamente **un error del 10.3 %**. También, se calculó la curva ROC visualmente con las mismas características que la que se mostraba para el BDT en Figura 4.6, y con un **AUC de 0.96** reflejando de nuevo la gran capacidad discriminatoria del modelo entre clases para cualquier valor del umbral.

$$\begin{pmatrix} 131\,853 & 20\,468 \\ 11\,019 & 141\,736 \end{pmatrix} \tag{4.2}$$

4.2.2.1. Justificación de la arquitectura de la red

Por otro lado, para determinar los parámetros que definen la arquitectura de la red, como el número de neuronas o el número de capas ocultas, fue necesario de nuevo hacer un estudio de la salida del modelo cuando estas toman distintos valores.

Número de neuronas por capa



Nº neuronas	Tiempo (s)
16	87
32	107
64	135
128	195
256	298

Figura 4.10: Resultados del rendimiento de la DNN según el número de neuronas por capa. A la izquierda se muestra la evolución de las métricas, y a la derecha el tiempo de entrenamiento.

De esta manera, cuando se varía el número de neuronas por capa y se mantiene fija la arquitectura con dos capas ocultas, se obtienen los resultados que se representan en Figura 4.10. Así puede verse que tanto la exactitud como el AUC aumentan significativamente en un principio con el número de neuronas. No obstante, a partir de 64 neuronas, las mejoras se vuelven marginales mientras que el tiempo de entrenamiento crece de forma significativa, pasando de unos 2 minutos con 64 neuronas a 5 minutos para 256 neuronas, lo que representa casi el doble más de tiempo para resultados prácticamente idénticos.

A modo de comprobación también se observaron las curvas de evolución de la exactitud durante el entrenamiento para cada una de las arquitecturas. Con el fin de no sobrecargar demasiado esta sección, estas representaciones pueden verse en el Apéndice B. No obstante, gracias a las técnicas de regularización de *dropout* y los criterios de parada, no se observó en ningún caso sobreentrenamiento.

Con todo, se considera que la disposición más óptima es aquella que cuenta con **64 neuronas** por cada capa oculta, al conseguir buen rendimiento con una baja eficiencia computacional.

Número de capas ocultas

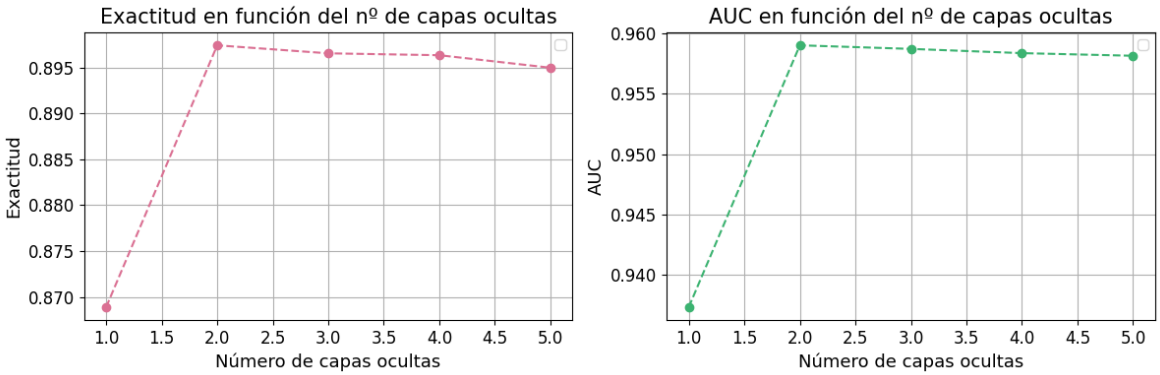


Figura 4.11: Resultados del rendimiento de la DNN cuando se varía el número de capas ocultas. A la izquierda se evalúa la exactitud y a la derecha el AUC

De manera análoga, fijando 64 neuronas por capa, también se exploró la influencia del número de capas ocultas en el rendimiento del modelo, considerando entre 1 y 5 capas. Los resultados obtenidos se presentan en Figura 4.11 dónde de nuevo se observa la evolución de la exactitud y el AUC. Así, el análisis muestra que al pasar de una sola capa oculta a dos supone una mejora significativa en ambas métricas, pero el aumentar el número de capas más allá no aporta beneficios pues se mantienen prácticamente estables o llegan incluso a bajar. Además, aunque ahora no se muestre, el tiempo de entrenamiento crece de nuevo sustancialmente, pasando de unos 2 minutos para 2 capas ocultas a casi

4 minutos para 5 capas. Por otro lado, tampoco en este caso se produjo sobreentrenamiento en ningún caso, reforzando una vez más la estabilidad del modelo (ver Apéndice B)

Con todo, la red que aporta mejor balance entre rendimiento y coste computacional es aquella compuesta por **2 capas ocultas**.

4.2.2.2. Relevancia de las variables de entrada

Para analizar la importancia de las variables de entrada en la red neuronal entrenada, se utilizó el método `SHAP`. Esta técnica asigna a cada variable de entrada unos determinados valores que miden sus respectivas contribuciones a la predicción de cada ejemplo individual, permitiendo entender qué variables influyen más en el modelo y cómo lo hacen. [36]

Concretamente, se calcularon las contribuciones de cada variable sobre 1000 ejemplos balanceados del conjunto de *test*. Aunque este contiene un número considerablemente mayor de muestras, se optó por reducir el análisis a unas pocas debido a que el cálculo es muy lento y computacionalmente costoso.

Así, a partir de estos valores, se construyó la representación de Figura 4.12 que muestra para cada variable, la contribución media a las predicciones del modelo. Es decir, cada barra representa el promedio calculado sobre los 1000 ejemplos de los valores `SHAP` asociados a una variable concreta.

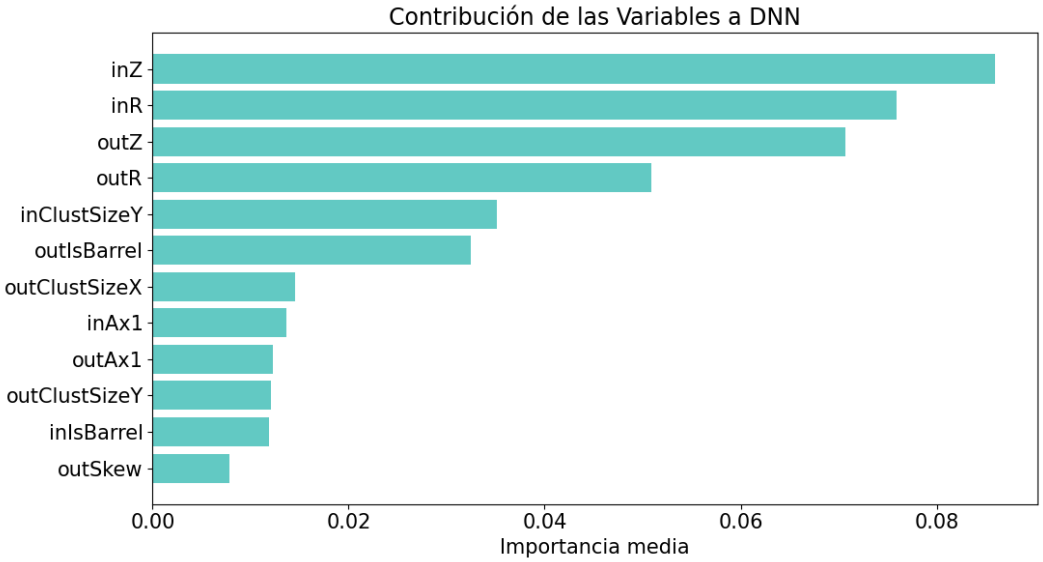


Figura 4.12: Importancia global de las 12 variables más influyentes en el modelo DNN, calculada como el valor absoluto medio de los valores `SHAP`

De esta manera, se observa que las variables más importantes en la decisión del modelo son las coor-

denadas espaciales `Z` y `R` tanto del *hit* de entrada (`in`) como de salida (`out`). En particular, variables como `inZ`, que ya se mostraban como claramente discriminatorias en Figura 4.4, confirman su relevancia dentro del modelo. Por el contrario, las variables asociadas al clúster o a la distinción entre *barrel* y *endcaps* tienen contribuciones significativamente menores.

En este punto es importante recordar que, a pesar de que previamente en Figura 4.3 se observó que la variable `DeltaPhi` ($\Delta\phi$) mostraba distribuciones visualmente distintas entre ambos tipos de dobletes, esta no aparece entre las variables más influyentes según los valores `SHAP`. Esto pone una vez más en manifiesto la capacidad de las redes neuronales profundas para identificar patrones complejos entre variables, más allá de lo que podría deducirse con una simple inspección visual o un análisis estadístico tradicional.

4.3. Comparación de modelos

Tras analizar por separado el rendimiento de ambos modelos de clasificación binaria, se presentan durante este apartado sus principales diferencias y similitudes.

Como se observa en Tabla 4.2 ambos modelos ofrecieron resultados satisfactorios. Concretamente la DNN logró unos valores de exactitud y AUC ligeramente superiores a los obtenidos por el BDT, reflejando una capacidad más elevada para distinguir entre las clases de dobletes. Además, en cuanto a coste computacional, la diferencia entre ambos modelos también fue significativa y favorable para la DNN. Esto se debe probablemente a que el elevado número de árboles y la optimización secuencial de ellos que caracteriza los métodos de *boosting*, produce un elevado número de iteraciones y por tanto aumenta considerablemente el tiempo de ejecución.

Métrica	**BDT**	**DNN**
Exactitud	82.9 %	89.7 %
AUC	0.91	0.96
Tiempo de entrenamiento	∼15 min	∼2 min

Tabla 4.2: Comparativa resumen entre BDT y DNN

No obstante, más allá de estas métricas globales, debe recordarse que el comportamiento del BDT mostró ciertas deficiencias estructurales. Concretamente, la distribución de la probabilidad predicha que se observaba en Figura 4.5 presentaba un doble pico para la clase `True`, sugiriendo que el modelo intentaba dividir erróneamente esta clase en subgrupos inexistentes. Este comportamiento apunta a

una menor capacidad del BDT para captar la verdadera estructura del problema. En cambio, la DNN generó una salida probabilística (Figura 4.9) más suave y acertada, reflejando una comprensión más realista del problema analizado.

Un punto a favor de ambos modelos es su robustez frente al sobreentrenamiento, pues en ninguno de los dos se observaron indicios claros de sobreajuste. Bien fuese por la implementación de técnicas de regularización como el *dropout* o los criterios de parada en la DNN, o bien por la profundidad reducida de los árboles que componían el BDT.

En conclusión, aunque ambos modelos ofrecieron resultados muy competentes, la **DNN** no solo superó al BDT en términos de exactitud y AUC, sino que además lo hizo con un coste computacional menor, lo que claramente la convierte en una **opción preferente** en este caso.

4.4. Impacto del balanceo en la muestra de *test*

Una vez se han estudiado y evaluado los modelos sobre una muestra de *test* balanceada, se quiso comprobar qué ocurría si se utiliza un conjunto de *test* que reflejase la proporción real de clases del problema. Y esque en la realidad, debido al fondo combinatorio, en cada evento se producen del orden de 10^6 dobletes pero solo unos 10^3 son verdaderos [35]. Esto implica que por cada doblete verdadero hay aproximadamente unos mil falsos, lo que da lugar a una proporción muy desbalanceada con el 99.9 % de `False` frente a un 0.1 % de `True`.

Es importante destacar que para realizar el entrenamiento de cada modelo, es necesario seguir usando una muestra balanceada pues cuando hay muchas más muestras de una clase, el modelo tiende a predecir siempre la clase más común, siendo incapaz de aprender bien los patrones característicos de cada clase y generalizando mal los resultados de su salida en el conjunto de *test*.

Así, los resultados obtenidos cuando solamente la muestra de *test* está desbalanceada se resumen a continuación:

Modelo	**Métrica**	***Test* balanceado**	***Test* realista**
BDT	AUC	0.9064	0.9031
	Exactitud (%)	82.9	81.6
DNN	AUC	0.9587	0.9579
	Exactitud (%)	89.6	86.8

Tabla 4.3: Comparación de AUC y exactitud para los modelos evaluados sobre una muestra de *test* balanceada y realista (desbalanceada).

Como se observa en Tabla 4.3, los resultados de la evaluación obtenidos sobre la muestra de *test* realista no difieren de forma significativa respecto a los obtenidos sobre la muestra de *test* balanceada. Aunque se observa una ligera disminución de las métricas para ambos modelos, las diferencias son pequeñas, especialmente en el AUC, que apenas varía en el tercer decimal.

Esto pone en manifiesto que ambos modelos son capaces de adaptarse correctamente a las condiciones extremas impuestas por el problema sin comprometer seriamente el rendimiento de sus resultados.

4.5. Importancia de la correcta clasificación de dobletes

La clasificación de dobletes es un paso crítico en la reconstrucción de trayectorias en CMS, pues como se veía en Figura 4.1 estos constituyen los primeros ladrillos sobre los que posteriormente se construyen las trayectorias más largas. De esta manera, si un doblete válido se rechaza, o si se acepta uno erróneo, la trayectoria resultante puede quedar incompleta o directamente carecer de sentido alguno. Estos errores cometidos en las primeras etapas son difíciles de corregir más adelante y afectan de primera mano a la calidad del *tracking*.

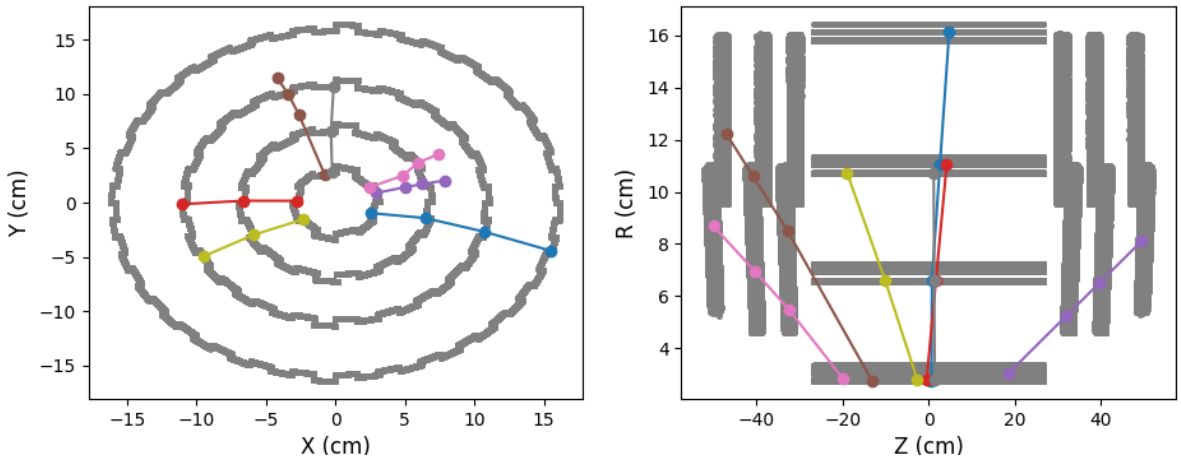


Figura 4.13: Ejemplo de cómo varios dobletes verdaderos pueden unirse para formar trayectorias en el subdetector de píxeles. A la izquierda la vista de estas trayectorias en el plano $x - y$. A la derecha las mismas en el plano $r - z$.

Con el fin de ilustrar el papel fundamental que juegan los dobletes en la reconstrucción de trayectorias, en Figura 4.13 se muestra un ejemplo simplificado en el que se enlazan varios dobletes cercanos en el detector de píxeles que podrían formar posibles trayectorias. Cabe destacar que esto solo es un esquema ilustrativo y que los algoritmos reales de reconstrucción en CMS siguen criterios geométricos, cinemáticos y estadísticos mucho más complejos, que están fuera del objetivo de este trabajo.

Al ser el primer subdetector que atraviesan las partículas cargadas tras la colisión, el píxel permite identificar con gran precisión la posición donde estas se originan. En este contexto, se conoce como **vértice primario** (PV) al punto del espacio donde ocurre la interacción entre los protones. No obstante, muchas de las partículas originadas en la colisión no son estables y tras recorrer una cierta distancia se desintegran, generando lo que se conoce como **vértices secundarios** (SV) y así sucesivamente. Estos vértices se encuentran buscando el punto del espacio donde múltiples (indicando PV) o un conjunto reducido (en el caso de SV o posteriores) de trayectorias confluyen.

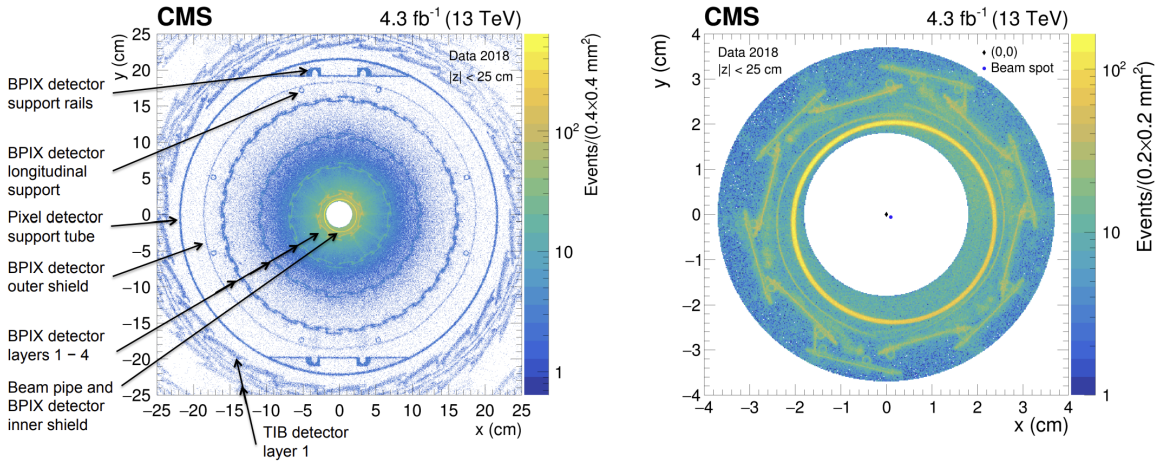


Figura 4.14: Densidad de vértices reconstruidos en la parte más interna del detector de trazas de CMS en el plano $x - y$ en la región del *barrel* ($|z| < 25$cm). La densidad de vértices se indica con la escala de color. A la izquierda una vista más externa donde aparecen indicadas las distintas partes del detector. A la derecha una vista ampliada que cubre la primera capa del BPIX. Recuperada de [19]

Esto puede verse en Figura 4.14, donde se observa que la parte más interna del detector de píxeles alberga una altísima densidad de vértices reconstruidos. Resulta curioso sobre todo fijarse en la gráfica de la derecha, dónde aparece representado tanto el origen de coordenadas como el punto de colisión medio de los haces de protones (*beam spot*), es decir, el mejor estimador de dónde deberían encontrarse los vértices primarios. Se observa que este está desplazado del centro geométrico y que a su alrededor se forma un anillo concéntrico en el que se da el máximo de reconstrucción de vértices.

Este anillo más interno y brillante es el BPIX *inner shield*, una capa de protección frente a la enorme radiación que se produce en las colisiones. Aunque esta capa no está diseñada para medir, las partículas cargadas que la atraviesan, interactúan con ella y producen partículas secundarias que sí son detectadas por las capas posteriores del BPIX. De esta manera, al aplicar los algoritmos de reconstrucción de vértices se observa una enorme acumulación justo en su superficie.

En este contexto, la correcta reconstrucción de los vértices depende directamente de la calidad de

las trayectorias formadas a partir de los impactos detectados, siendo esencial por tanto la correcta clasificación de dobletes. Por ejemplo, un caso particularmente importante es el de los quarks b, cuyos hadrones se desintegran a distancias del orden de milímetros, dando lugar a vértices secundarios claramente separados del punto de colisión. Detectar correctamente estos vértices permite identificarlos de forma eficiente, algo fundamental en casos como la búsqueda de desintegraciones del bosón de Higgs en $b\bar{b}$, o en la producción de quarks top ($t \to Wb$). [37]

Este problema se vuelve aún mas complejo con las condiciones de creciente luminosidad que se están alcanzando (Figura 2.2), culpables además de que el número de colisiones simultáneas (*pile-up*) se esté volviendo especialmente alto. A la izquierda, en Figura 4.15 se muestra cómo la eficiencia de detección de *hits* en las capas del detector de píxeles disminuye con la luminosidad, especialmente en las capas más internas como el *Layer 1*. Esto ocurre por la alta ocupación en el detector, que hace que se aumente la probabilidad de que ciertos módulos como este último, queden inactivos por saturación.

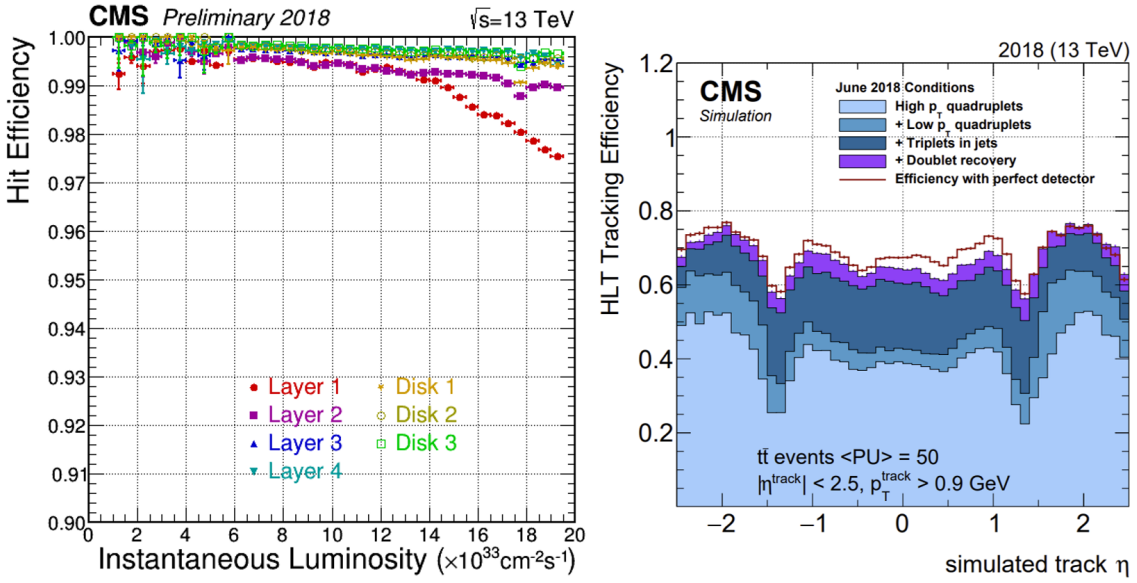


Figura 4.15: A la izquierda, eficiencia de detección de *hits* en función de la luminosidad instantánea para distintas capas (*layer*) y discos (*disk*) del detector de píxeles. A la derecha eficiencia de reconstrucción de trazas en CMS en función de la pseudorrapidez (η). Recuperadas de [38] y [39]

Esta pérdida de información directa en los *hits* registrados por los módulos tiene una consecuencia inmediata sobre la eficiencia global de reconstrucción de trayectorias, especialmente en las primeras fases del *High Level Trigger* (HLT), que debe tomar decisiones rápidas y fiables en muy poco tiempo. Para ilustrar esta situación, se puede observar la gráfica de la derecha en Figura 4.15 donde se muestra cómo distintas estrategias de reconstrucción contribuyen a la eficiencia total.

Entre ellas, destaca la iteración de recuperación basada en dobletes (banda morada), que permite reconstruir trayectorias en aquellas regiones del detector donde, debido a la inactividad de uno o varios módulos, no es posible formar tripletes o cuadrupletes. De hecho, tal y como se observa, en algunas regiones del detector, al incluir la iteración de dobletes, la eficiencia total logra alcanzar prácticamente la línea roja que representa el rendimiento ideal de un detector sin módulos inactivos.

Esto pone en manifiesto una vez más la importancia crucial de los dobletes en la reconstrucción de trayectorias, especialmente en condiciones de alta ocupación como las que están por llegar. Esta capacidad de reconstrucción es la que ha permitido a experimentos como CMS identificar fenómenos como el bosón de Higgs, y es la que seguirá permitiendo buscar nuevas partículas, observar interacciones raras o incluso estudiar desviaciones del Modelo Estándar.

Capítulo 5

Conclusiones

El experimento CMS del LHC genera una cantidad masiva de datos a cada segundo de funcionamiento, lo que plantea un gran reto a la hora de reconstruir con precisión y eficiencia las trayectorias de partículas cargadas que atraviesan el detector, sobre todo en las capas más internas. Esta complejidad del entorno experimental es lo que hace que sea imprescindible desarrollar algoritmos que puedan discriminar de forma eficaz entre combinaciones reales o inverosímiles de señales.

En este contexto, a lo largo de estas páginas se ha centrado la atención en una etapa concreta pero esencial del proceso de reconstrucción de trayectorias: la clasificación de dobletes, o dicho de otra manera, la identificación de dos señales en distintas capas del detector de píxeles que podrían o no haber sido generadas por la misma partícula. Esta etapa resulta fundamental, no solo para reducir la complejidad del proceso completo de *tracking*, sino también como método de recuperación de información en el *High-Level Trigger* (HLT) cuando el detector se encuentra saturado y algunos módulos pueden fallar, contribuyendo así a mejorar la robustez del sistema.

La dificultad del problema es notable pues se producen miles de eventos por segundo y en cada uno de ellos pueden generarse del orden de 10^6 dobletes, de los cuales solamente 10^3 son verdaderos. Esta desproporción hace que no cualquier algoritmo sencillo sea capaz de resolver el problema pues la probabilidad de clasificar correctamente por azar es mínima.

Es por ello que para abordar esta tarea, se han comparado dos enfoques basados en aprendizaje automático, pudiendo comprobarse que, si bien ambos son capaces de distinguir en cierta medida los dobletes verdaderos de los falsos, las redes neuronales ofrecen ventajas claras, tanto en términos de rendimiento como de eficiencia computacional. Además puesto que ambos fueron entrenados sobre un conjunto de datos simulados diseñados específicamente para reproducir las condiciones exactas del detector, los resultados obtenidos podrían ser perfectamente transferibles a escenarios reales.

Lejos de tratarse de una mera aplicación de un código, este trabajo ha requerido un análisis detallado de las variables relevantes, fundamentado en una comprensión profunda tanto del funcionamiento del detector como de la física que implica, lo que ha permitido diseñar un modelo eficaz y bien adaptado a las condiciones impuestas por el experimento.

Con todo, este estudio pone en manifiesto el potencial del aprendizaje automático, y en particular de las redes neuronales, para abordar problemas en física de altas energías. Concretamente se han obtenido resultados muy prometedores con un AUC de 0.96 y una exactitud cercana al 90 %, que respaldan la fiabilidad del modelo en esta tarea concreta. Es por ello que, esto demuestra que una estrategia centrada en fases intermedias como la clasificación de dobletes puede tener un impacto real y tangible en la eficiencia y robustez del proceso global de reconstrucción de trayectorias.

Bibliografía

[1] W. Cottingham and D. Greenwood. *An introduction to the standard model of particle physics*. Cambridge university press, 2007. págs. 1-20.

[2] CERN. The Standard Model. https://home.cern/science/physics/standard-model. Visitada el 21-05-2025.

[3] Quantum Diaries. The Standard Model: A Beautiful but Flawed Theory. https://www.quantumdiaries.org/2014/03/14/the-standard-model-a-beautiful-but-flawed-theory/, 2014. Visitada el 21-05-2025.

[4] D. Perkins. *Introduction to high energy physics*. Cambridge university press, 2000. pags. 1-14 y 20-26.

[5] W. Herr and B. Muratori. Concept of luminosity. *CERN*, 2006.

[6] CERN. Pulling together: Superconducting electromagnets. https://home.cern/science/engineering/pulling-together-superconducting-electromagnets. Visitada el 23-05-2025.

[7] CERN. Linear Accelerator 4. https://home.cern/science/accelerators/linear-accelerator-4. Visitada el 22-05-2025.

[8] A. Lopes and M. Perrey. Faq-LHC the guide. Technical report, CERN, 2022.

[9] H. Bartosik and G. Rumolo. Performance of the LHC injector chain after the upgrade and potential development. *arXiv preprint arXiv:2203.09202*, 2022.

[10] H. Gray. TASI 2022 lectures on LHC experiments. *arXiv preprint arXiv:2307.01894*, 2023.

[11] CERN. Longer term LHC schedule. https://lhc-commissioning.web.cern.ch/schedule/LHC-long-term.htm, 2024. Visitada el 11-06-2025.

[12] CERN. CMS. https://home.cern/science/experiments/cms. Visitada el 24-05-2025.

[13] CMS collaboration et al. Precision measurement of the structure of the CMS inner tracking system using nuclear interactions. *arXiv preprint arXiv:1807.03289*, 2018.

[14] I. Neutelings. CMS Coordinate System. https://tikz.net/axis3d_cms/, 2020. Visitada el 24-05-2025.

[15] E. Daw. Lecture 7-rapidity and pseudorapidity. *Lecture Notes*, 2012. págs. 7-8.

[16] CERN. CMS Experiment at the LHC. https://public-archive.web.cern.ch/en/lhc/CMS-en.html. Visitada el 26-05-2025.

[17] CERN. Energy of electrons and photons (ECAL). https://cms.cern/detector/measuring-energy/energy-electrons-and-photons-ecal. Visitada el 25-05-2025.

[18] CERN. Energy of hadrons (HCAL). https://cms.cern/detector/measuring-energy/energy-hadrons-hcal. Visitada el 25-05-2025.

[19] The Tracker Group of the CMS Collaboration. The CMS Phase-1 Pixel Detector Upgrade. *arXiv preprint arXiv:2012.14304*, 2020.

[20] F. Pantaleo. *New track seeding techniques for the CMS experiment*. PhD thesis, CERN, 2017.

[21] J. Lydrich. CMS Tracker Detector Performance Results. https://twiki.cern.ch/twiki/bin/view/CMSPublic/DPGResultsTRK, 2025. Visitada el 26-05-2025.

[22] E. Yigitbasi. New CMS trigger strategies for the Run 3 of the LHC. Technical report, CERN, 2023.

[23] CMS collaboration et al. The CMS trigger system. *arXiv preprint arXiv:1609.02366*, 2016.

[24] CMS Collaboration and Mc Cauley, T. Collisions recorded by the CMS detector on 14 Oct 2016 during the high pile-up fill. https://cds.cern.ch/record/2231915, 2016. Visitada el 31-05-2025.

[25] C. Janiesch, P. Zschech, and K. Heinrich. Machine learning and deep learning. *Electronic markets*, 31(3), 2021. págs. 685–695.

[26] GeeksforGeeks. Decision Tree in Machine Learning. https://www.geeksforgeeks.org/decision-tree-introduction-example/, 2025. Visitada el 27-05-2025.

[27] Y. Coadou. Boosted decision trees. In *Artificial Intelligence for High Energy Physics*. World Scientific, 2022. págs. 9–58.

[28] Steven Walczak. Artificial neural networks. In *Encyclopedia of Information Science and Technology, Fourth Edition*. IGI Global Scientific Publishing, 2018. págs. 120–131.

[29] Thakur, A. ReLU vs. Sigmoid Function in Deep Neural Networks. https://wandb.ai/ayush-thakur/dl-question-bank/reports/ReLU-vs-Sigmoid-Function-in-Deep-Neural-Networks--VmlldzoyMDk0MzI, 2022. Visitada el 11-06-2025.

[30] A. Krenker, J. Bešter, and A. Kos. Introduction to the artificial neural networks. *Artificial Neural Networks: Methodological Advances and Biomedical Applications. InTech*, 2011. págs. 1-18.

[31] S. Joglekar. Overfitting and Human Behavior. https://medium.com/@srjoglekar246/overfitting-and-human-behavior-5186df1e7d19, 2018. Visitada el 29-05-2025.

[32] S. Raschka. An overview of general performance metrics of binary classifier systems. *arXiv preprint arXiv:1410.5330*, 2014.

[33] Google Developers. Clasificación: ROC y AUC. https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc?hl=es-419, 2024. Visitada el 30-05-2025.

[34] A. Bocci, V. Innocente, F. Pantaleo, et al. Heterogeneous reconstruction of tracks and primary vertices with the CMS pixel tracker. *arXiv preprint arXiv:2008.13461*, 2020.

[35] A. Di Florio, F. Pantaleo, and M. Pierini. Sample with tracker hit information for tracking algorithm ML studies TTbar_13TeV_PU50_PixelSeeds. https://opendata.web.cern.ch/record/12320, 2019. Visitada el 30-05-2025.

[36] E. Mosca and F. Szigeti. SHAP-based explanation methods: a review for NLP interpretability. In *Proceedings of the 29th international conference on computational linguistics*, 2022. págs. 4593–4603.

[37] C. Ferro. B-tagging at CMS. In *EPJ Web of Conferences*, volume 28. EDP Sciences, 2012.

[38] G. Negro. CMS Inner Tracker Status and Performance. In *Proceedings of the 31st International Workshop on Vertex Detectors (VERTEX2022)*, 2024.

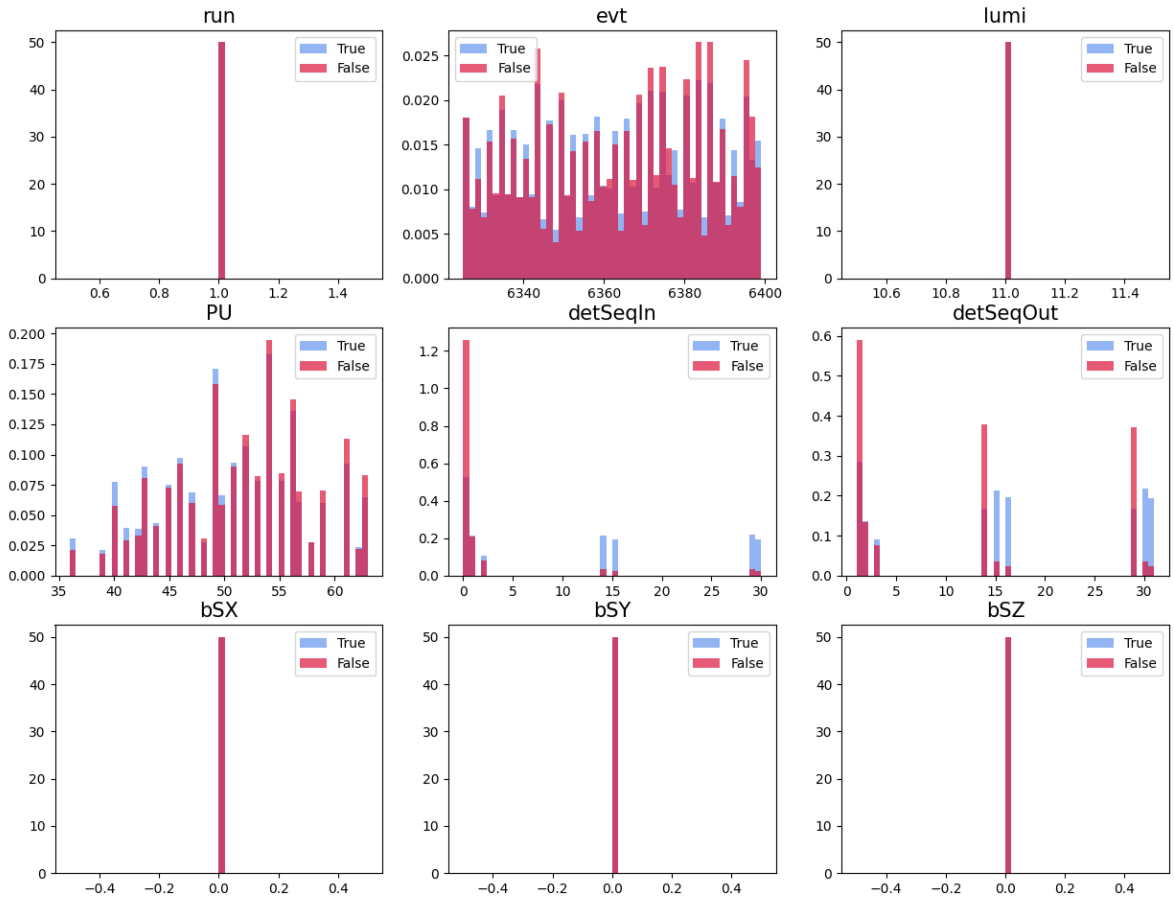
[39] CMS Collaboration et al. Tracking Performance in the CMS High Level Trigger-June 2018. Technical report, CMS-DP-2018-038, https://cds.cern.ch/record/2629370, 2018.

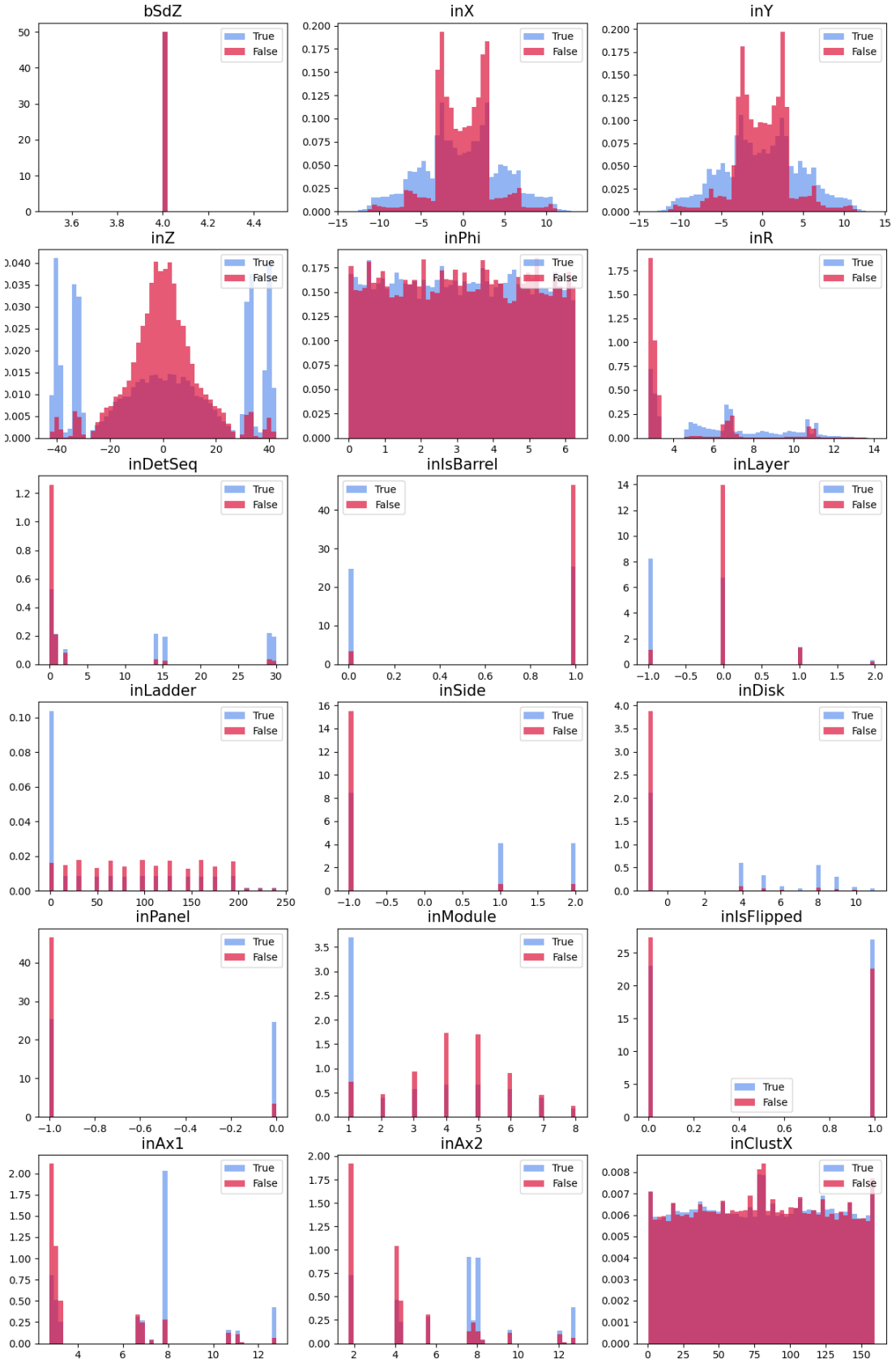
[40] A. Hayrapetyan, A. Tumasyan, W. Adam, et al. Performance of the CMS high-level trigger during LHC Run 2. *Journal of Instrumentation*, 2024.

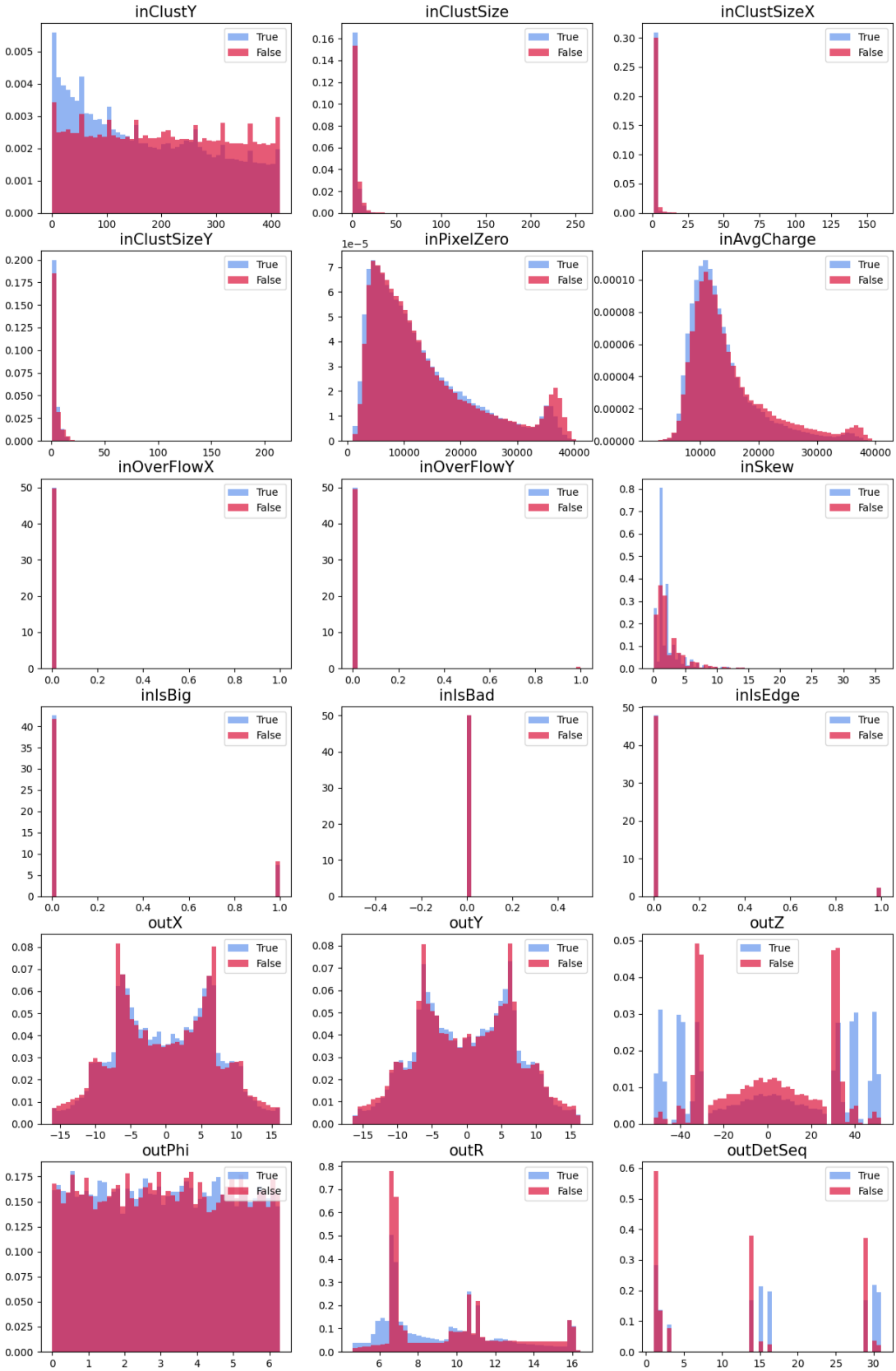
Apéndices

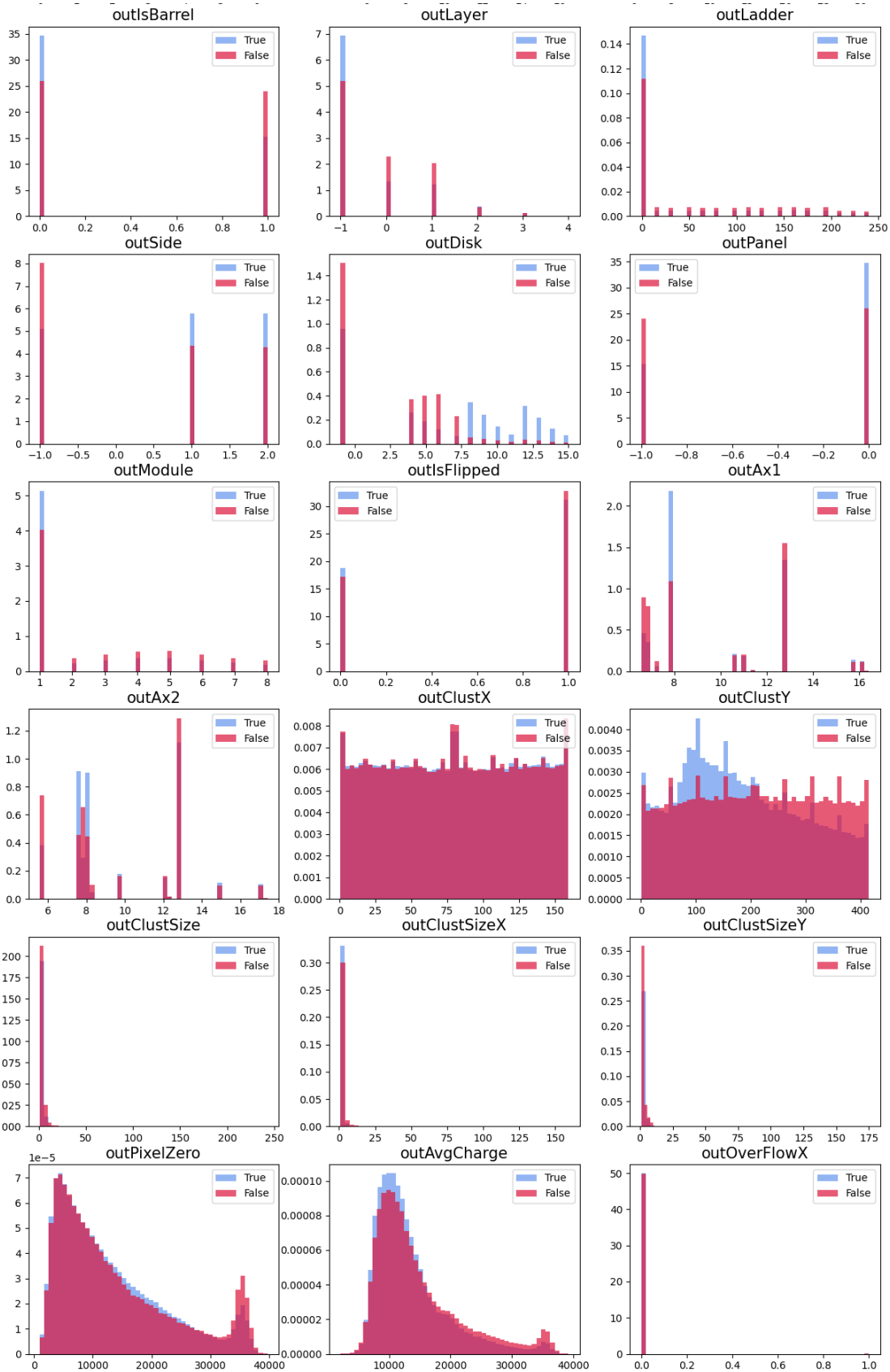
A. Distribución de todas las variables de la muestra

Se muestran durante esta sección **todas** las distribuciones de las variables que contenía la muestra de datos con la que se ha realizado este trabajo. Observar las características de cada variable ha resultado fundamental para seleccionar las variables que se incluirían en el análisis y cuales no.




bSdZ
inX
inY
inZ
inPhi
inR
inDetSeq
inIsBarrel
inLayer
inLadder
inSide
inDisk
inPanel
inModule
inIsFlipped
inAx1
inAx2
inClustX
True
False


inClustY
inClustSize
inClustSizeX
inClustSizeY
inPixelZero
1e−5
inAvgCharge
inOverFlowX
inOverFlowY
inSkew
inIsBig
inIsBad
inIsEdge
outX
outY
outZ
outPhi
outR
outDetSeq
True
False


outIsBarrel
outLayer
outLadder
outSide
outDisk
outPanel
outModule
outIsFlipped
outAx1
outAx2
outClustX
outClustY
outClustSize
outClustSizeX
outClustSizeY
outPixelZero
outAvgCharge
outOverFlowX
True
False
1e−5

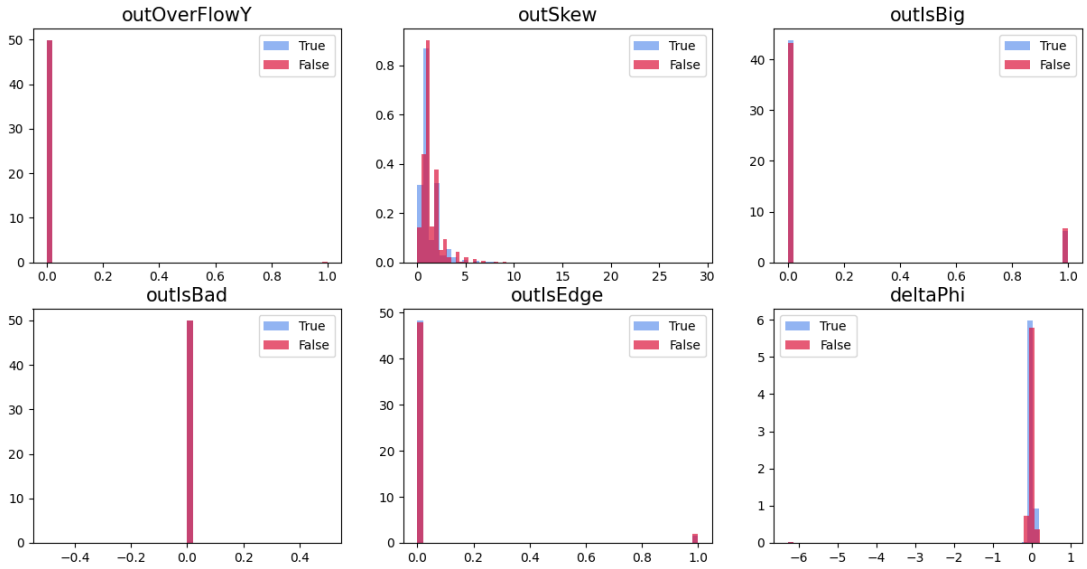


Figura A1: Distribución de todas las variables para dobletes `True` y `False`

Es importante notar que solamente se han incluido para el entrenamiento y evaluación de los modelos aquellas variables que resultaron discriminatorias entre ambos tipos de dobletes o que mejoraban el rendimiento de los modelos. Un claro ejemplo de este último caso es la variable `IsBarrel`, que, aunque en apariencia no resulta especialmente informativa al tratarse de una variable binaria, permite al modelo diferenciar eficazmente entre configuraciones situadas en el *barrel* y en los *endcaps*, lo que facilita la identificación de patrones específicos en cada región.

B. Evaluación del sobreentrenamiento

Se presentan las gráficas de evolución de la exactitud a lo largo del entrenamiento de los modelos de Boosted Decission Tree (BDT) y de la Red Neuronal Profunda (DNN). El objetivo principal de estas gráficas es evaluar la presencia o ausencia de sobreentrenamiento en cada uno de los modelos.

En el caso del **BDT**, se muestra la evolución de la exactitud al variar el número de árboles de decisión tanto en el conjunto de entrenamiento como en el conjunto de *test*.

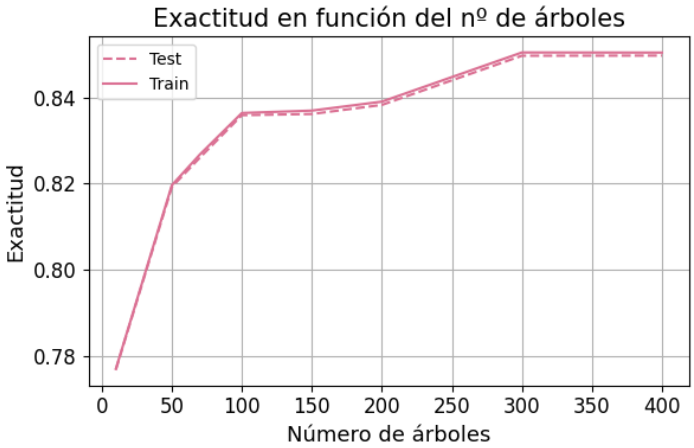


Figura B1: Evolución de la exactitud al variar el número de árboles de decisión del BDT en el conjunto de entrenamiento (*train*) y de prueba (*test*)

Se comprueba que las curvas en ambos conjuntos de datos crecen prácticamente paralelas para todos los valores de los árboles de decisión estudiados. Esto indica que el modelo **no sufre sobreentrenamiento**.

Por otro lado, para la **DNN** también se presentan las curvas de exactitud, pero en este caso correspondientes al conjunto de entrenamiento y validación en función del número de épocas para cada configuración evaluada.

Primero cuando se varió el **número de neuronas** por capa:

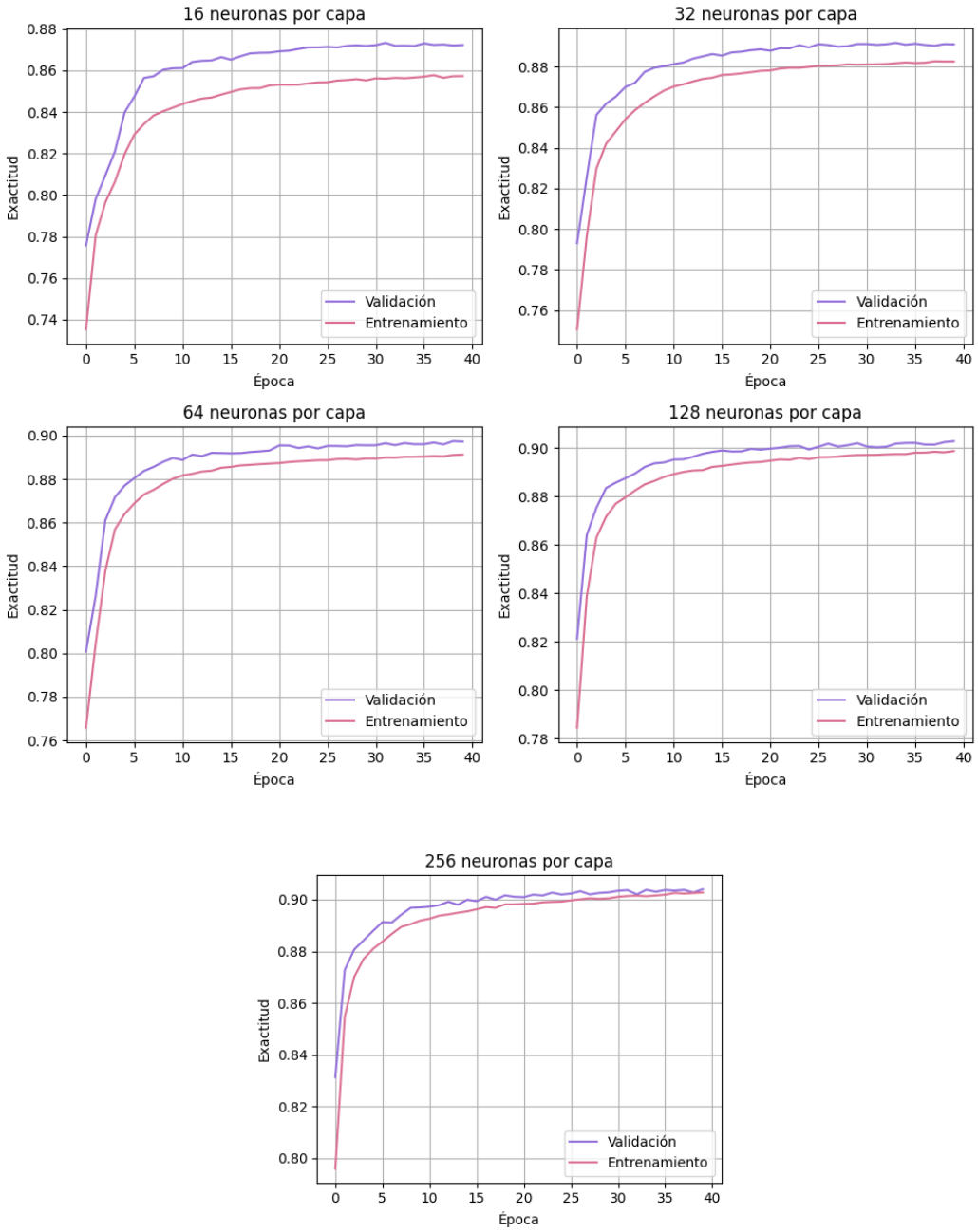


Figura B2: Evolución de la exactitud al variar el número de neuronas por capa oculta

Y cuando se varió el **número de capas ocultas**:

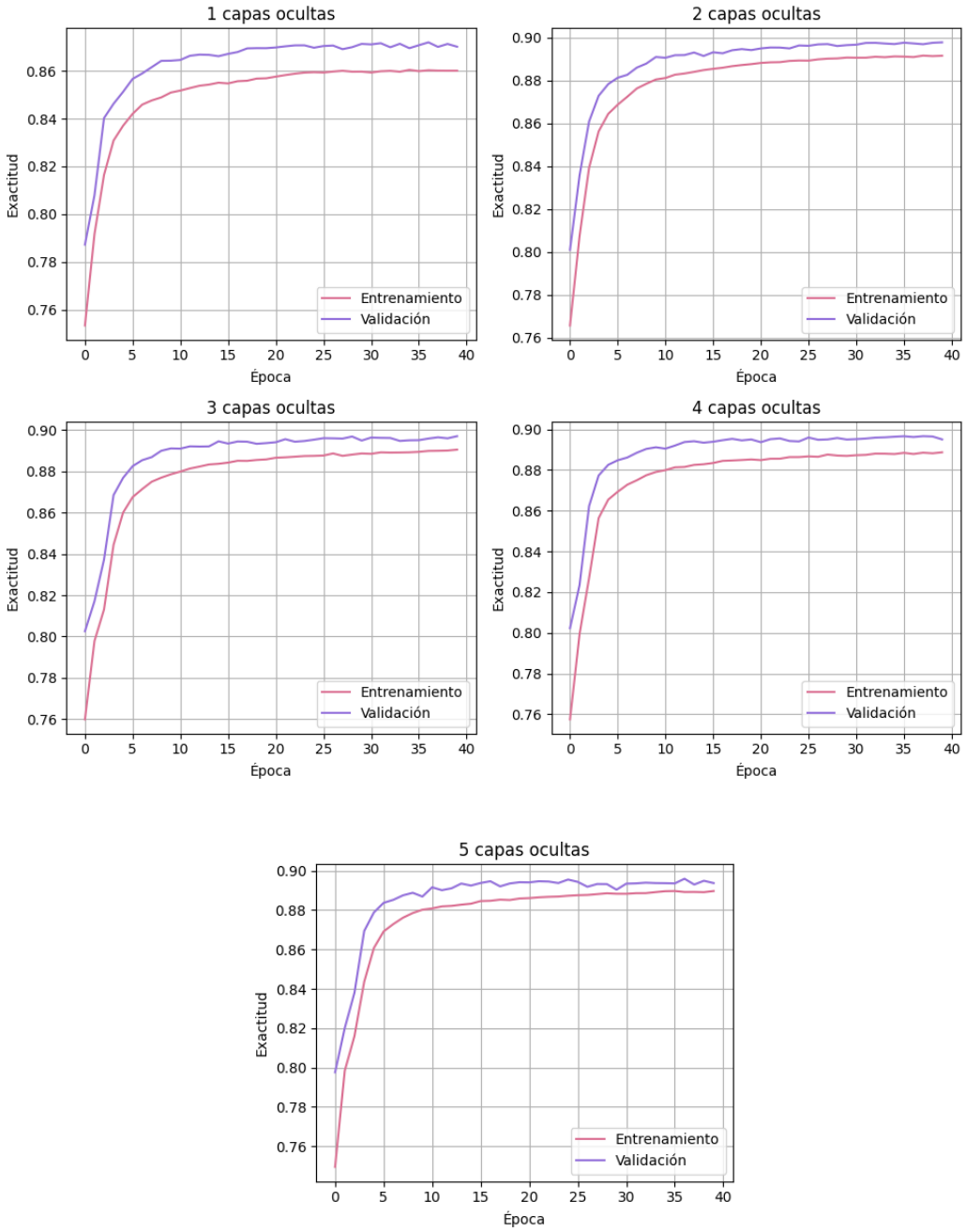


Figura B3: Evolución de la exactitud al variar el número de capas ocultas

Se observa que **no** se aprecia un comportamiento claro de sobreajuste en ninguno de los casos, pues la exactitud sobre el conjunto de validación evoluciona de forma paralela a la del conjunto de entrenamiento. No obstante, es importante notar cómo a medida que se aumentan tanto el número de neuronas como el de capas, las curvas de validación van acercándose cada vez más a las de entrenamiento, siendo probable que para valores más altos el modelo sí empezase a mostrar sobreentrenamiento.

C. El código

La parte más representativa de este trabajo y gracias a la cual fue posible obtener todos los resultados que se mostraron, es el código. Concretamente este se ejecutó en Kaggle, una plataforma en línea propiedad de Google que proporciona un entorno de *notebooks* en Python con acceso a GPU y un amplio conjunto de librerías preinstaladas, facilitando el desarrollo de modelos de *Machine Learning* y su visualización. A continuación se describen brevemente las principales librerías que se han utilizado:

- `numpy` y `pandas` para el manejo de datos numéricos y tabulares.
- `matplotlib` para la visualización de datos mediante representaciones gráficas.
- `Scikit-learn` para la implementación de modelos como el BDT o la evaluación mediante las distintas métricas.
- `TensorFlow` y `keras` para definir y entrenar redes neuronales.
- `shap` para interpretar la red neuronal explicando la contribución de cada entrada a las predicciones.

Puede accederse al código y la muestra de datos en el siguiente enlace:

`https://github.com/celiaglz/TFG-Dobletes.git`