

UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE INFORMÁTICA
VALPARAÍSO - CHILE



“IDENTIFICATION OF BOOSTED TAU PAIRS AT ATLAS EXPERIMENT USING DEEP NEURAL NETWORKS”

JAVIER ALEJANDRO ZAVALA TORRES

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL EN INFORMÁTICA

Profesor Guía: Dra. Raquel Pezoa Rivera
Profesor Co-guía: Dr. Edson Carquín López
Profesor Correferente: Dr. Claudio Torres López

Enero - 2022

DEDICATORIA

Dedicado a mis padres Pablo Zavala y Pamela Torres, y a mi hermano Marcos Zavala por haberme apoyado y acompañado, especialmente durante la pandemia.

AGRADECIMIENTOS

Primero quiero agradecer a Patricio Calderón por ayudarme a obtener el conjunto de datos utilizados para desarrollar este trabajo, y a Raquel Pezoa y Edson Carquín por haberme guiado y resuelto mis dudas.

También quiero agradecer a Sebastián Bórquez, Patricio Campaña, Pablo Flores y John Rodríguez por haberme acompañado durante mis años en la universidad y por hacer divertidas todas las noches que pasamos sin dormir.

Este trabajo fue parcialmente financiado por el proyecto FONDECYT Postdoc N 3190740, y por el proyecto ANID PIA/APOYO AFB180002.

RESUMEN

Resumen— Mediciones relacionadas con los eventos de física de altas energías que producen un par de partículas tau es fundamental en la búsqueda de nueva física, debido a que es uno de los decaimientos más importantes del bosón de Higgs. Científicos en el experimento ATLAS de CERN trabajan en el desarrollo de algoritmos dedicados a la identificación del par de partículas tau. Este trabajo se enfoca en el desarrollo de un algoritmo computacional para la identificación de la generación de un par de partículas tau del tipo *boosted*, lo que incluye (i) reducción y validación del conjunto de datos compuesto por la señal de pares tau *boosted* y el background correspondiente a *jets* de hadrones (ii) desarrollo de un algoritmo basado en redes neuronales profundas para clasificar los eventos en señal o background, (iii) validación del método propuesto y comparación con la solución actual. Los experimentos realizados mostraron que se obtuvo un F1 score de 0.989, alcanzando en un 99.9 % a la solución actual, y se obtuvo un *background rejection* de $9.460 \cdot 10^2$ para el *working point very loose* con eficiencia de señal 0.95, alcanzando en un 74.6 % a la solución actual.

Palabras Clave— experimento ATLAS, lepton tau, identificación de un par de taus, máquinas de aprendizaje, redes neuronales profundas.

ABSTRACT

Abstract— Measurements related to high-energy physics events producing a pair of tau particles are essential to search for new physics because it is one of the most important decays of the Higgs boson. Scientists in CERN's ATLAS experiment are working on the development of algorithms dedicated to the identification of pairs of tau particles. This work focuses on the development of a computational algorithm for identifying the generation of a pair of tau particles of the type *boosted*, which includes (i) reduction and validation of the data set composed of the pair signal of taus *boosted* and the background corresponding to hadron jets (ii) development of an algorithm based on deep neural networks to classify events into signal or background, (iii) validation of the proposed method and comparison with the current solution. The experiments carried out resulted in an F1 score of 0.989, reaching 99.9% of the current solution, and a background rejection of $9,460 \cdot 10^2$ was obtained for the working point very loose with signal efficiency 0.95, reaching 74.6 % of the current solution.

Keywords— ATLAS experiment, tau pair identification, machine learning, deep neural networks.

GLOSARIO

Siglas

ALFA Absolute Luminosity For ATLAS. 7

ALICE A Large Ion-Collider Experiment. 4

ANN Artificial Neural Network. 2, 12, 20

ATLAS A Toroidal LHC Apparatus. 1–4, 6–8, 10, 11, 24, 26, 28

BDT Boosted Decision Tree. 1, 2, 9, 10, 12, 19, 20, 50, 52–54, 57

CCTVal Centro Científico Tecnológico de Valparaíso. 2, 11

CERN Conseil européen pour la recherche nucléaire. 1, 3, 11

CMS Compact Muon Solenoid. 4, 7

DT Decision Tree. 17–19

FCNN Fully Connected Neural Network. 24

FCNN07 Fully Connected Neural Network 07. 49

FCNN1 Fully Connected Neural Network 1. 52–54, 58, 59

FCNN2 Fully Connected Neural Network 2. 50, 52–54, 58

FCNN3 Fully Connected Neural Network 3. 49, 50, 52–54, 57, 58

HEP High Energy Physic. 12, 16

LAr Argón Líquido. 6

LHC Large Hadron Collider. 1, 3, 6–8, 12, 26, 28, 58

LHCb LHC b-hadron experiment. 4

LHCf LHC forward experiment. 4

LUCID Luminosity measurement using Cerenkov Integrating Detector. 7

MoEDAL Monopole and Exotics Detector At the LHC. 4

QCD Quantum chromodynamics. 7, 9, 58

ROC Receiver Operating Characteristic. 16, 50, 52

TOTEM TOTal Elastic and diffractive cross-section Measurement. 4

UTFSM Universidad técnica federico santa maria. 1, 10, 26

ZDC Zero-Degree Calorimeter. 7

ÍNDICE DE CONTENIDOS

RESUMEN	IV
ABSTRACT	IV
GLOSARIO	VI
ÍNDICE DE FIGURAS	X
ÍNDICE DE TABLAS	XI
INTRODUCCIÓN	1
CAPÍTULO 1: DEFINICIÓN DEL PROBLEMA	3
1.1 Contexto	3
1.1.1 Experimento ATLAS	6
1.1.2 Leptón Tau	7
1.1.3 Producción de pares Tau	7
1.1.4 Pares tau boosted	8
1.2 Solución Actual	8
1.2.1 Boosted Decision Trees	9
1.3 Objetivos	10
1.4 Contribución	11
CAPÍTULO 2: MARCO CONCEPTUAL	12
2.1 Estado del arte	12
2.2 Machine Learning	12
2.2.1 Tipos de aprendizaje	13
2.2.2 Representación de datos	13
2.2.3 Métricas de desempeño	14
2.3 Decision Trees	17
2.3.1 Boosted Decision Trees	19
2.4 Redes Neuronales Artificiales	20
2.4.1 Redes Feed-Forward	21
2.4.2 Hiperparámetros	22
2.4.3 Funciones de activación	22
CAPÍTULO 3: PROPUESTA DE SOLUCIÓN	24
3.1 Obtención de los datos	26
3.2 Análisis de los datos	30
3.3 Pre-procesamiento de los datos	36
3.4 Modelo propuesto y búsqueda de hiperparámetros	42
3.4.1 Primera búsqueda de hiperparámetros	44

3.4.2 Segunda búsqueda de hiperparámetros	46
3.4.3 Tercera búsqueda de hiperparámetros	47
CAPÍTULO 4: VALIDACIÓN DE LA SOLUCIÓN	50
4.1 Métricas	50
4.2 Matriz de confusión	50
4.3 Curva ROC	52
4.4 Distribución de Puntuaciones	54
CAPÍTULO 5: CONCLUSIONES	58
5.1 Trabajo futuro	59
REFERENCIAS BIBLIOGRÁFICAS	60

ÍNDICE DE FIGURAS

1	Large Hadron Collider	4
2	Modelo Estándar	5
3	Detector ATLAS	6
4	Reconstrucción objetos Boosted	8
5	Ejemplo matriz de confusion clasificación binaria	15
6	Ejemplo curva ROC	17
7	Estructura Binary Decision Tree	18
8	Diagrama BDT	20
9	Red Neuronal Profunda	21
10	Proceso de trabajo	25
11	Sistema de coordenadas de ATLAS	27
12	Distribuciones de variables antes de filtrar	32
13	Distribuciones de variables antes de filtrar	33
14	Distribuciones de variables después de filtrar	34
15	Distribuciones de variables después de filtrar	35
16	Cantidad de datos background	36
17	Histograma cantidad de datos señal	37
18	Gráfico de torta: Desbalance de clases	38
19	Gráfico de torta: Undersampling	38
20	Histograma cantidad de datos muestra	39
21	Distribuciones de variables	40
22	Distribuciones de variables	41
23	Matrices de confusión	51

24	ROC Curve	52
25	Distribución de puntuaciones FCNN1	55
26	Distribución de puntuaciones FCNN1	55
27	Distribución de puntuaciones FCNN3	56
28	Distribución de puntuaciones BDT	56

ÍNDICE DE TABLAS

1	Nombre de datasets	27
2	Espacio de búsqueda primera optimización de hiperparámetros	45
3	Hiperparámetros mejor modelo primera optimización	45
4	Métricas mejor modelo primera optimización	46
5	Espacio de búsqueda segunda optimización de hiperparámetros	46
6	Hiperparámetros mejor modelo segunda opitmización	47
7	Métricas mejor modelo segunda opitmización	47
8	Espacio de búsqueda tercera optimización de hiperparámetros	48
9	División espacio de búsqueda tercera optimización de hiperpámetros	48
10	Métricas tercera optimización de hiperparámetros	48
11	Hiperparámetros mejor modelo tercera opitmización	49
12	Métricas mejores modelos	50
13	Working points y eficiencias mejores modelos	53

INTRODUCCIÓN

Los orígenes de la física de partículas se remontan a 1897 con el descubrimiento del electrón por J. J. Thomson. Desde este evento, muchas partículas han sido descubiertas. Más adelante, durante la década de 1960, se comenzó a desarrollar un modelo, que unificó todas las partículas conocidas y dió explicación a las interacciones existentes entre ellas. Este modelo logró predecir la existencia de nuevas partículas y luego paso a ser conocido como modelo estándar.

La organización Europea para la investigación nuclear (CERN) ha tenido un papel importante, midiendo experimentalmente diversas predicciones del modelo estándar. En 2008 se inauguró el Large Hadron Collider (LHC) en CERN, cuyo objetivo principal en aquella época era probar la existencia del bosón de Higgs junto con la búsqueda de física más allá del modelo estándar. El bosón de Higgs fue descubierto en 2012 por los experimentos ATLAS y CMS, siendo la última partícula faltante del modelo estándar.

A pesar de que el modelo estándar predice con precisión muchos de los eventos que ocurren a nivel sub-atómico, existen interrogantes que el modelo estándar no logra responder, siendo una de ellas el *strong CP problem* [di Wu, 1997]. Por esto se cree que existe física más allá del modelo estándar. El modelo *Two-Higgs-Doublet* [Branco *et al.*, 2012] es una extensión del modelo estándar y predice la existencia de 4 bosones de Higgs adicionales. El bosón de Higgs se desintegra antes de alcanzar a los detectores, por lo que solo pueden ser detectados los productos de su decaimiento. Un modo importante de decaimiento del bosón de Higgs en el LHC es en $\tau^+\tau^-$ (o un par de *taus*), ya que son el único decaimiento leptónico que ha sido observado hasta el momento.

Debido a la naturaleza de los decaimientos de tipo boosted, los *taus* tienden a solaparse en el detector, volviendo difícil su identificación mediante los métodos tradicionales. El método del estado del arte, utiliza el enfoque Boosted Decision Tree (BDT) para realizar esta identificación [Kirchmeier, 2015].

Considerando que el enfoque de redes neuronales profundas o *deep learning* ha logrado resultados sobresalientes en la solución de diversos problemas, es que en este trabajo se investiga el uso de este tipo de técnicas para la identificación del par *tau* del tipo *boosted*. Para ello, se genera un conjunto de datos en colaboración con miembros de ATLAS UTFSM, con el que se entrenan distintos modelos, utilizando un algoritmo de búsqueda en rejilla o *grid search* para la optimización de hiperparámetros. Los experimentos computacionales se realizaron utilizando el lenguaje de programación Python¹, junto con diversos módulos que incluyen Pandas², Keras³, y Talos⁴. Este trabajo se llevó a cabo dentro de un equipo

¹<https://www.python.org/>

²<https://pandas.pydata.org/>

³<https://keras.io/>

⁴<https://github.com/autonomio/talos>

de investigación multidisciplinario, que incluye la colaboración de ATLAS, el Departamento de Informática, el Departamento de Física y el Centro Científico Tecnológico de Valparaíso (CCTVal).

En el Capítulo 1 se contextualizan los conceptos de física de partículas con las que se trabajará. Además, se describen las herramientas actuales que se utilizan para resolver el problema y se presentan los objetivos del trabajo. En el Capítulo 2 se explica el estado del arte y se describen los métodos Boosted Decision Tree y Artificial Neural Network. En el Capítulo 3 se describe el proceso realizado para la obtención de los datos y se exponen los modelos propuestos. En el Capítulo 4 se comparan los resultados obtenidos con la solución actual. Finalmente en el Capítulo 5 se presentan las conclusiones del trabajo.

CAPÍTULO 1

DEFINICIÓN DEL PROBLEMA

En este capítulo se contextualiza y presenta el problema a resolver, junto con los objetivos de este trabajo. En específico, se introduce el Large Hadron Collider (Large Hadron Collider (LHC)), el experimento A Toroidal LHC Apparatus (ATLAS), y los detectores que éste tiene. Luego se presenta el modelo estándar y se introduce a la partícula *tau*. Después se explica la problemática que existe al identificar pares *tau* tipo *boosted*. Finalmente se presenta la solución actual y los objetivos de este trabajo.

1.1. Contexto

El Large Hadron Collider (LHC) ubicado en Conseil européen pour la recherche nucléaire (CERN), la organización Europea para la investigación nuclear, es el acelerador de partículas más grande y poderoso que existe actualmente [Evans y Bryant, 2008]. Este enorme colisionador consiste en un anillo de 27 kilómetros de circunferencia compuesto por imanes superconductores, ubicado en la frontera entre Francia y Suiza. Dentro del LHC, haces de protones collisionan con una frecuencia de 40 MHz, con una energía de centro de masa de 13 TeV, de los cuales solo una muestra de 1 kHz con eventos de interés físico es guardada para su posterior análisis.



Figura 1: Ubicación de los distintos experimentos en el Large Hadron Collider. Fuente: <https://home.cern/news/news/cern/25-years-large-hadron-collider-experimental-programme>

Como se muestra en la Figura 1, existen 4 experimentos principales: ATLAS, CMS, LHCb, y ALICE.

A Toroidal LHC Apparatus (ATLAS) y Compact Muon Solenoid (CMS) son dos de los experimentos con propósito general, cuyo objetivo principal es estudiar diversos aspectos de la producción y decaimiento de partículas. En cada uno de ellos participan más de 3000 científicos e ingenieros, casi 200 instituciones de investigación, y cerca de 40 países, provenientes de todos los continentes exceptuando la Antártica.

Los otros experimentos, donde participan alrededor de 1000 científicos e ingenieros, son más especializados: A Large Ion-Collider Experiment (ALICE) se centra en la investigación de colisiones de iones pesados, y LHC b-hadron experiment (LHCb) se centra principalmente en la investigación relacionada con el decaimiento de hadrones que contienen un bottom quark. Estos experimentos principales son complementados con tres experimentos más pequeños: LHC forward experiment (LHCf), Monopole and Exotics Detector At the LHC (MoEDAL) y TOTem Elastic and diffractive cross-section Measurement (TOTEM) [Harrison, 2011].

Han pasado 9 años desde que se anunció el descubrimiento del bosón de Higgs. Las colaboraciones ATLAS [Aad *et al.*, 2012] y CMS [Chatrchyan *et al.*, 2012] descubrieron esta nueva partícula con una masa de aproximadamente 125 GeV, lo que confirma la predicción del Modelo Estándar, abriendo una nueva área de estudio, planteando nuevas preguntas por resolver [ATLAS, 2018].

El modelo estándar de física de partículas es la teoría que describe todas las partículas elementales conocidas y cómo interactúan entre sí. Este modelo ha explicado casi todos los fenómenos detectados en los colisionadores y ha predicho con precisión una amplia variedad de estos.

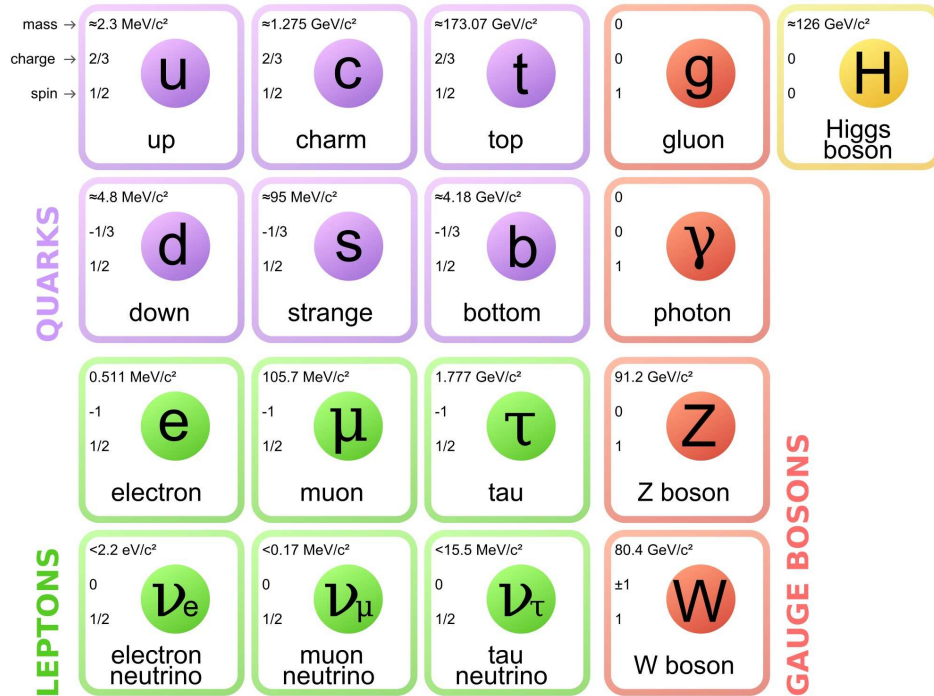


Figura 2: Partículas del Modelo Estándar. Fuente: https://cds.cern.ch/record/2315477/files/2000px-Standard_Model_of_Elementary_Particles.png

El modelo estándar divide las partículas elementales en dos categorías: fermiones y bosones. Los fermiones son los componentes básicos de la materia. Hay dos tipos de fermiones llamados quarks y leptones (Figura 2). Cada categoría consta de seis partículas y se divide en 3 generaciones. Cada generación es más pesada que la anterior, siendo la primera, la más estable. Toda la materia del universo está hecha de partículas que pertenecen a la primera generación, porque cualquier partícula más pesada se desintegraría rápidamente a otras más estables. Quarks *up* y *down* pertenecen a la primera generación; *charm* y *strange*, a la segunda; y *top* y *bottom*, a la tercera. El *electrón* y *neutrino electrónico* pertenecen a la primera generación de leptones; *muon* y *neutrino muónico*, a la segunda; y *tau* y *neutrino tauónico*, a la tercera. Los bosones son partículas que transfieren tres de las cuatro fuerzas fundamentales del universo (gravedad, electromagnética, fuerte y débil). *Gluones* llevan la fuerza fuerte; *photones*, la fuerza electromagnética; y los bosones Z y W, la fuerza débil. Finalmente, la interacción con el bosón de Higgs da masa a todas las demás partículas. [CERN,]

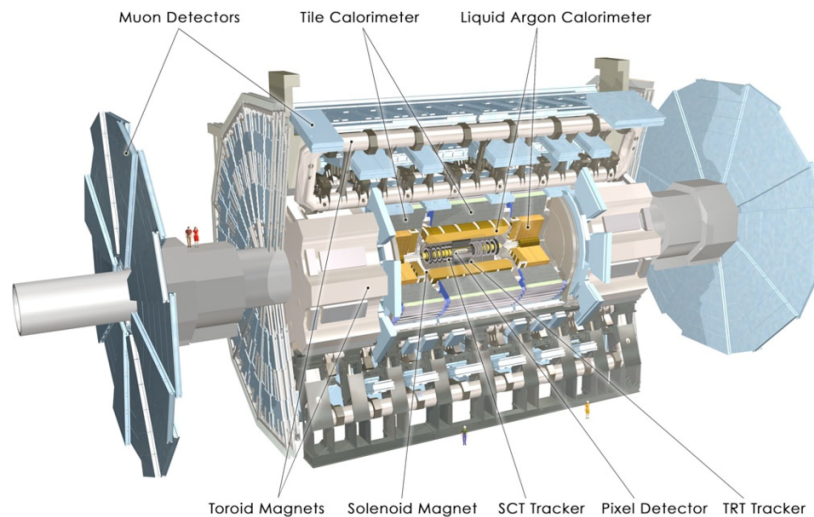


Figura 3: Componentes del detector ATLAS. Fuente: <http://cds.cern.ch/record/1364846/files/atlas-schematic-diagram.png>

1.1.1. Experimento ATLAS

ATLAS es uno de los cuatro detectores principales del LHC. Tiene 40m de largo, 25m de alto y pesa alrededor de 7000 t. ATLAS está diseñado para detectar la mayoría de las partículas producidas en *eventos* o colisiones protón-protón, debido a sus millones de sensores especialmente diseñados para medir diferentes propiedades de las partículas, como momento, energía, carga eléctrica, entre otras. La Figura 3 muestra los diferentes componentes del detector ATLAS. Consiste en cuatro detectores principales: un detector interno, un sistema de calorímetro, un espectrómetro de muones y un detector de avance [Aad *et al.*, 2008].

El detector interno mide la dirección, el impulso y la carga de las partículas cargadas eléctricamente. Consiste en un detector de píxeles, un rastreador de semiconductores y un rastreador de radiación de transición [ATLAS, c].

Los calorímetros miden la energía que pierde una partícula al pasar por el detector. Por lo general, absorben la mayoría de las partículas provenientes de una colisión, excepto los muones y neutrinos. El sistema de calorímetro ATLAS consiste en el Calorímetro de Argón Líquido (LAr), que es un calorímetro electromagnético, y el Calorímetro Hadrónico Tile, que es un calorímetro hadrónico. Los calorímetros electromagnéticos miden la energía de los electrones y fotones y los calorímetros hadrónicos miden la energía de los hadrones, como los protones, neutrones y piones [ATLAS, b].

El espectrómetro de muones identifica y mide los momentos de los muones. Consta de cuatro subsecciones: cámaras de espacio delgado, cámaras de placas resistivas, tubos de deriva supervisados y cámaras de tiras catódicas [ATLAS, d].

Los detectores de avance de ATLAS miden partículas que se han alejado muchos metros del punto de interacción en el detector principal de ATLAS. Luminosity measurement using Cerenkov Integrating Detector (LUCID) y Absolute Luminosity For ATLAS (ALFA) miden la luminosidad entregada a ATLAS, y Zero-Degree Calorimeter (ZDC) determina la colisión de iones pesados cerca de la línea del haz. Además, LUCID es el principal monitor de luminosidad relativa en línea para ATLAS [ATLAS, a].

1.1.2. Leptón Tau

El leptón *tau* (τ) es una partícula elemental con carga eléctrica negativa, un *spin* de $\frac{1}{2}$ y una masa de $1.777[GeV]$. El leptón *tau* decae alrededor de 65 % del tiempo a hadrones (principalmente piones), y leptones más ligeros (muones y electrones) el resto del tiempo. Debido a su corta longitud de desintegración, el leptón *tau* decae antes de alcanzar la capa interna del detector de pixeles y solo se puede observar sus productos de desintegración. Por esta razón, es necesario utilizar los productos de su descomposición para reconstruir la partícula original.

Las desintegraciones hadrónicas del leptón tau pueden, fácilmente, identificarse erróneamente como *QCD jets*⁵ debido a su naturaleza, aunque los *jets* producidos por la desintegración hadrónica del *tau* suelen ser más delgadas y contienen solo unos pocos *tracks*⁶ en comparación con los *QCD jets*.

1.1.3. Producción de pares Tau

Los pares *tau* o también llamados *di-tau* se producen en abundancia durante las colisiones protón-protón en el LHC y algunos de ellos provienen de la desintegración de resonancias pesadas como los bosones Z o de Higgs. Las mediciones que involucran pares *tau* provenientes del bosón de Higgs han sido muy importantes en la determinación del acoplamiento del bosón de Higgs (descubierto en 2012 en el LHC por las colaboraciones ATLAS y CMS), ya que este es el único modo de desintegración leptónica del Higgs que es accesible en el LHC y observado hasta la fecha [Aad *et al.*, 2015] [Sirunyan *et al.*, 2018] [Aaboud *et al.*, 2019]. Los pares tau también se han utilizado como herramienta en la búsqueda de copias más pesadas del bosón de Higgs o de los bosones Z [Aad *et al.*, 2020] [Sirunyan *et al.*, 2019].

⁵Corrientes de partículas producidas a partir de quarks y gluones expulsados de los protones originales o de las desintegraciones hadrónicas de otras partículas más pesadas

⁶partículas individuales cargadas eléctricamente

1.1.4. Pares tau boosted

La identificación de pares tau boosted es un proceso importante para muchas búsquedas más allá de la física del modelo estándar, ya que es útil para extender el alcance a masas más altas que los análisis actuales realizados por los experimentos del LHC, por ejemplo, en un evento como los estados finales de producción de un par Higgs $X \rightarrow HH$ o $A \rightarrow ZH$ cuando decaen a $b\bar{b}\tau\tau$ (o en general para decaimientos de la forma $YY\tau\tau$). Cuando, en un evento, los bosones de Higgs o Z se producen con un gran impulso, sus productos tienden a superponerse en el detector y las técnicas para identificar la partícula *tau* se vuelven menos eficientes. Es aquí donde las técnicas dedicadas de identificación de pares *boosted* entran en juego (Figura 4).

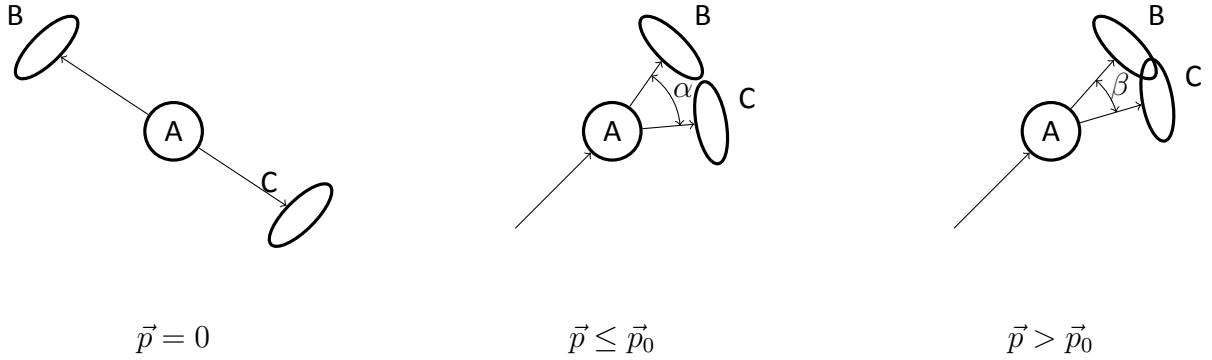


Figura 4: La Figura muestra la partícula A desintegrándose en las partículas B y C. Cuando el momentum transversal de A es 0, B y C toman direcciones opuestas. A medida que aumenta el momentum transversal de A, sus productos de desintegración quedan cada vez más cerca. Cuando el momentum transversal de A es menor o igual a $\vec{p}_0$, se puede realizar una reconstrucción de ambas partículas por separado utilizando conos. Sin embargo, cuando el momentum transversal de A es mayor a $\vec{p}_0$, los conos utilizados en la reconstrucción se superponen, dificultando la reconstrucción.

Pero la identificación de pares *tau boosted* es aún más desafiante que la identificación de un solo *tau* dada su topología más compleja y el hecho de que involucra dos procesos de desintegración de *taus* independientes. Los pares *tau boosted* decaen antes de alcanzar los detectores de ATLAS. Por ésta razón, es necesario reconstruir las partículas de *tau* a partir de sus productos de decaimiento. Aquí nos ocuparemos de solo de una de estas posibilidades de decaimiento. Esta es, el caso completamente hadrónico en el que ambos taus se desintegran en hadrones.

1.2. Solución Actual

En 2015, Kirchmeier propuso un método de reconstrucción e identificación de boosted di-tau, basado en la reconstrucción e identificación de un solo *tau* [Kirchmeier, 2015]. Para la re-

construcción de un solo *tau* se utiliza como base el algoritmo *anti-kt* [Cacciari *et al.*, 2008b], el cuál entrega candidatos a *tau*. Luego es necesario realizar un proceso de identificación para poder diferenciar aquellos candidatos que realmente son *taus* (señal) de aquellos que no lo son (background).

Como se mencionó anteriormente, cuando una partícula se produce con gran impulso, sus productos tienden a superponerse en el detector. Esto dificulta la tarea de reconstrucción de los pares *taus boosted*. Es por esto, que Kirchmeier propuso la reconstrucción de un objeto que contuviera ambos *taus* y luego desarrolló un método para la identificación de estos objetos.

A continuación, se describe el método desarrollado por Kirchmeier [Kirchmeier, 2015] para la identificación de *boosted di-taus*.

1.2.1. Boosted Decision Trees

Los Boosted Decision Tree (BDT) son una variación de los árboles de decisión, donde se utiliza un conjunto de árboles para mejorar la robustez de la respuesta. En un árbol de decisión se utilizan datos de señal y background previamente etiquetados para su entrenamiento. Durante éste, se eligen cortes de variables óptimos que separen los datos de señal y background. Cada vez que se realiza un corte, los datos se dividen en dos. Este proceso se repite hasta alcanzar un mínimo de datos en un nodo o un máximo en la profundidad del árbol. Cada nodo hoja⁷ cuenta con una proporción entre datos de señal y background que determina la etiqueta que predice el nodo.

El objetivo del *boosting* es incrementar la capacidad de separación, combinando varios árboles de decisión. Cada árbol es optimizado para reclasificar datos que fueron mal clasificados por el árbol anterior. La predicción de cada árbol de decisión es ponderada mediante una técnica de votación, obteniendo un valor entre 0 y 1. Luego se utiliza un umbral o *working point* para determinar la clase a la que pertenece un dato, es decir, si el valor obtenido es mayor al umbral el dato pertenece a la clase señal y si es menor, pertenece a la clase background.

Los datos utilizados en el entrenamiento fueron obtenidos a partir de simulaciones de Monte Carlo. Se utilizó POWHEG BOX [Alioli *et al.*, 2010] para generar los eventos y PYTHIA 8.1 [Sjöstrand *et al.*, 2008] para la simulación de los *jets* tanto de señal como de background. Se utilizó el decaimiento $A \rightarrow Zh \rightarrow ll + \tau\tau$ para la obtención de un conjunto de datos enriquecido en pares tau con alto p_T como señal y *jets* QCD como background.

Debido a la compleja topología del problema, [Kirchmeier, 2015] propuso un nuevo algoritmo de reconstrucción de objetos taus. Se utiliza el algoritmo anti- k_T [Cacciari *et al.*, 2008a] con un *seed-jet* anti- k_t con radio $R = 1.0$ grande y otros *jets* anti- k_T con radio $R = 0.2$

⁷último nodo en una rama de un árbol de decisión

más pequeño (*sub-jets*) para la formación de los *jets* que son utilizados como entrada para el algoritmo de reconstrucción de los objetos *di-tau*.

El conjunto de objetos *di-tau* es utilizado para entrenar un BDT. En el entrenamiento se utilizó AdaBoost para el *boosting*, con un máximo de 50 árboles y con una profundidad máxima de 8 nodos. Un máximo de 200 cortes son aplicados por variable. El criterio de parada es un mínimo del 1 % de la data en alguno de los nodos.

Para el *working point* 0.65 se obtuvo una eficiencia de señal de 0.9, es decir, 90 % de los datos de señal se predijeron correctamente; y se obtuvo un rechazo de background de 10^2 , es decir, 1 de cada 100 datos de background es confundido por señal. Además, el modelo logró incrementar el rango de energía en el que se puede reconstruir e indentificar *taus* desde 500 GeV hasta 1200 GeV.

1.3. Objetivos

Objetivo General:

Desarrollar un algoritmo para identificar los pares *tau boosted*, generados en el experimento ATLAS, utilizando técnicas de redes neuronales profundas para comparar los resultados con el estado del arte.

Objetivos específicos:

- *Generar un conjunto de datos adecuado.* Trabajar en obtener una muestra manejable y suficientemente limpia, para ser utilizada como entrada para el entrenamiento posterior del algoritmo de identificación de pares *tau boosted*. Esto se hará en cooperación con miembros de la colaboración ATLAS y otros estudiantes de la UTFSM del departamento de física e informática.
- *Diseñar e implementar un algoritmo basado en aprendizaje automático.* Desarrollo de un nuevo algoritmo para la identificación del par *tau boosted*, basado en redes neuronales profundas.
- *Validar el algoritmo.* Evaluar el método propuesto, comparando el rendimiento de esta propuesta con la solución actual.

1.4. Contribución

Este trabajo pretende generar un conjunto de datos, utilizando el software de colaboración ATLAS, y desarrollando un método de aprendizaje automático para la identificación del estado final del par *tau boosted*, completamente hadrónico y semileptónico, basado en redes neuronales profundas. Este trabajo intenta mejorar los resultados del estado del arte, contribuyendo a la búsqueda de nueva física en CERN. El trabajo desarrollado puede ser encontrado en el repositorio [Zavala, 2021]⁸, junto con los pesos de las mejores redes neuronales encontradas para la reproducibilidad del trabajo realizado. Este trabajo se desarrolló dentro de un entorno de investigación multidisciplinario que incluye la Colaboración ATLAS, el Departamento de Informática, el Departamento de Física y CCTVal.

⁸El repositorio solo puede ser accedido por miembros de ATLAS

CAPÍTULO 2

MARCO CONCEPTUAL

En este capítulo se describe el estado del arte junto con el marco teórico de los métodos de *machine learning* o aprendizaje automático, utilizados en este trabajo. Se explican los métodos de Boosted Decision Tree (BDT), y Artificial Neural Network (ANN), describiendo sus componentes, procesos de entrenamiento y prueba.

2.1. Estado del arte

El aprendizaje automático ha sido fundamental para el análisis de datos de física de alta energía (HEP), generados en el LHC. Las redes neuronales y los Boosted Decision Tree [Drucker y Cortes, 1995] han sido las técnicas más utilizadas durante los últimos años, sobre todo en problemas de clasificación donde se debe distinguir señal de background [Albertsson *et al.*, 2018].

El aprendizaje profundo es el enfoque actual de aprendizaje automático que ha revolucionado el análisis de datos en muchos campos científicos, y particularmente en HEP, el aprendizaje profundo ha permitido obtener altos niveles de desempeño en la tarea de clasificación de eventos. En los experimentos del LHC se han implementado múltiples algoritmos de deep learning para resolver problemas como clasificación de *jets*, tracking y simulaciones rápidas [Guest *et al.*, 2018].

Actualmente, para la identificación del leptón *tau* con decaimiento hadrónico, se utiliza un método basado en redes neuronales recurrentes [ATL, 2019], el cual reemplazó al antiguo método basado en BDT. Sin embargo, el método no es aplicable para *boosted di-taus*, a pesar de estar compuesto por dos *taus*, debido a la compleja topología del problema.

En [Kirchmeier, 2015] se presentó un método basado en Boosted Decision Tree (BDT) para el etiquetado de *boosted di-taus*. Este logró un rechazo de background de 10^2 para una eficiencia de señal de 0.9, e incrementó el rango de energía en el que se puede reconstruir e identificar *boosted di-taus* hasta un p_T de 1200 GeV. Pero aún existe margen de mejora, en particular para rangos mayores a 1200 GeV.

2.2. Machine Learning

Según Arthur Samuel, *machine learning* es el campo de estudio que proporciona a los computadores la capacidad de aprender sin programarse explícitamente [Samuel, 1959]. A diferencia de un algoritmo normal, donde existe una lista de instrucciones que se le entrega al compu-

tador para resolver un problema, en machine learning el computador cuenta con un modelo, el cual puede evaluar ejemplos, y con una pequeña lista de instrucciones, modifica el modelo a medida que éste aprende. Existen 3 tipos de formas en que un modelo aprende: aprendizaje supervisado, aprendizaje no supervisado y aprendizaje reforzado [Hastie *et al.*, 2009].

2.2.1. Tipos de aprendizaje

- **Supervisado:** Este tipo de aprendizaje está basado en la idea de corregir el modelo cuando entrega un resultado erróneo basado en ciertos márgenes preestablecidos. Para ello, es necesario contar con ejemplos cuya respuesta es conocida, y, de esta manera, el modelo ajusta sus respuestas para futuras predicciones [Hastie *et al.*, 2009].
- **No supervisado:** A diferencia del aprendizaje supervisado, en este tipo de aprendizaje no se cuenta con respuestas conocidas, sino que el modelo trata de aprender patrones en los datos y formas de representarlos. Ejemplos de este tipo de aprendizaje son: reconocimiento de patrones, clustering y reducción de dimensionalidad [Ghahramani, 2004].
- **Reforzado:** Este tipo de aprendizaje toma lugar en un “*ambiente*”, donde un “*agente*” puede tomar acciones, y es recompensado de acuerdo a éstas. De esta manera, el agente intenta encontrar una “*política*” para tomar decisiones que maximice futuras recompensas [Hastie *et al.*, 2009].

En este trabajo se utilizará un método de aprendizaje supervisado. En este tipo de aprendizaje existen dos tipos principales de problemas: problemas de regresión y problemas de clasificación. En problemas de regresión el objetivo es predecir valores continuos, pertenecientes a alguna distribución de probabilidad. En problemas de clasificación, por otro lado, se intenta predecir a qué categoría pertenece un objeto.

2.2.2. Representación de datos

Los datos son una parte importante en machine learning, pero muchas veces estos no vienen con el formato más adecuado para el tipo de problema que se quiere resolver. Es por ello que se requiere un pre-procesado de los datos, el cual deriva en un sistema de información que representa el conocimiento almacenado en los datos crudos. Existen distintos tipos de sistemas de información, pero algunos de los más comunes son tablas, secuencias e imágenes.

Cuando se utiliza un sistema de información de tipo tabla, las filas de la tabla representan una muestra u observación y las columnas representan una característica o atributo de la muestra [Gopal, 2019]. Existen 3 tipos principales de atributos en este tipo de representación: atributos categóricos, atributos continuos y atributos discretos.

Los atributos categóricos contienen un número finito de categorías y pueden no tener un orden lógico. A este tipo de atributos se les suele transformar a *one-hot vector*, donde cada categoría es transformada a un atributo con valor 1 si el dato pertenece a la categoría y valor 0 si no pertenece. Esta transformación se realiza para poder representar de forma numérica las categorías.

Los atributos continuos son atributos numéricos que tienen un número infinito de valores entre dos valores cualesquiera. A este tipo de atributos se les suele aplicar algún tipo de transformación para evitar realizar operaciones aritméticas entre dos valores con ordenes de magnitud muy distintos, debido a que esto puede producir errores en los cálculos. Las transformaciones más comunes son estandarización, normalización y escalar el atributo a un rango determinado.

Los atributos discretos son atributos numéricos que tienen un número de valores contable entre dos valores cualesquiera. Este tipo de atributo puede ser tratado como un atributo continuo o categórico, siendo este último solo posible si existe una cantidad finita de valores.

Por último, una cantidad elevada de atributos puede producir problemas en el entrenamiento, porque se suele necesitar un modelo más complejo. En este tipo de casos se aplican técnicas de reducción de dimensionalidad para disminuir la cantidad de atributos.

En este trabajo se utiliza una representación de tipo tabla con atributos de tipo continuo. En la Sección 3.1 se detallan los atributos utilizados.

2.2.3. Métricas de desempeño

Medir el desempeño de un modelo es crucial para evaluar que tan buenas son las predicciones de éste y también para compararlo con otros modelos. La métrica más común es el *accuracy* o exactitud, la cual se calcula dividiendo la cantidad de datos bien clasificados por la cantidad total de datos.

$$\text{Accuracy} = \frac{\text{cantidad de datos bien clasificados}}{\text{cantidad de datos total}}$$

Sin embargo, el *accuracy* puede ocultar el desempeño que un modelo tiene sobre una clase. Este problema se vuelve especialmente notorio cuando existe desbalance de clases, ya que la clase minoritaria tiende a ser una pequeña fracción de la cantidad total de datos, por lo que no influye de manera significativa en el *accuracy*. Una métrica común para medir el desempeño de un modelo por cada clase es la matriz de confusión.

		Predicted classes	
		Negative 0	Positive 1
Actual classes	Negative 0	TN	FP
	Positive 1	FN	TP

Figura 5: Matriz de confusión para clasificación binaria junto con la abreviación de los términos utilizados para cada casilla. Fuente: https://cdn-images-1.medium.com/max/950/1*PPgItHcPSaskyjLMWFC-Kw.png

La matriz de confusión mide la cantidad de datos bien clasificados y mal clasificados, separandolos por clase. Esto permite detectar las clases que el modelo confunde y medir el desempeño del modelo de manera más precisa. En la Figura 5 se puede observar una matriz de confusión para clasificación binaria. A continuación se describen los términos que aparecen en la figura:

- **Verdadero Positivo o True Positive (TP):** Corresponde a observaciones positivas que fueron predichas correctamente.
- **Falso Negativo o False Negative (FN):** Corresponde a observaciones positivas que fueron predichas incorrectamente.
- **Falso Positivo o False Positive (FP):** Corresponde a observaciones negativas que fueron predichas incorrectamente.
- **Verdadero Negativo o True Negative (TN):** Corresponde a observaciones negativas que fueron predichas correctamente.

A partir de estos términos se pueden calcular varias métricas:

- **Sensibilidad o recall:** Razón entre los datos positivos correctamente clasificados y la cantidad total de datos positivos.

$$\text{Recall} = \frac{TP}{TP + FN}$$

- **Precisión o *precision*:** Razón entre los datos positivos correctamente clasificados y la cantidad total de datos predichos positivo.

$$\text{Precision} = \frac{TP}{TP + FP}$$

- **Valor F1 o *f1 score*:** Promedio armónico entre *recall* y precisión.

$$\text{F1 Score} = 2 \cdot \frac{\text{Recall} \cdot \text{Precision}}{\text{Recall} + \text{Precision}}$$

Otra métrica importante es la Receiver Operating Characteristic (ROC), la cual es una interpretación gráfica entre la razón de verdaderos positivos (TPR) y la razón de falsos positivos (FPR).

$$TPR = \frac{TP}{TP + FN} \quad FPR = \frac{FP}{FP + TN}$$

Como se puede ver en la Figura 6, mientras mayor es el área bajo la curva, mejor es la predicción del modelo. Esto se debe a que TPR mide la cantidad de datos correctamente predichos sobre el conjunto de observaciones positivas, mientras que FPR mide la cantidad de datos incorrectamente predichos sobre el conjunto de observaciones negativas.

Por último, en High Energy Physics (HEP) se suele calcular la curva ROC utilizando las métricas eficiencia de la señal (*signal efficiency*) y *background rejection* o *inverse background efficiency*.

$$\text{Signal Efficiency} = TPR \quad \text{Inverse Background Efficiency} = \frac{1}{FPR}$$

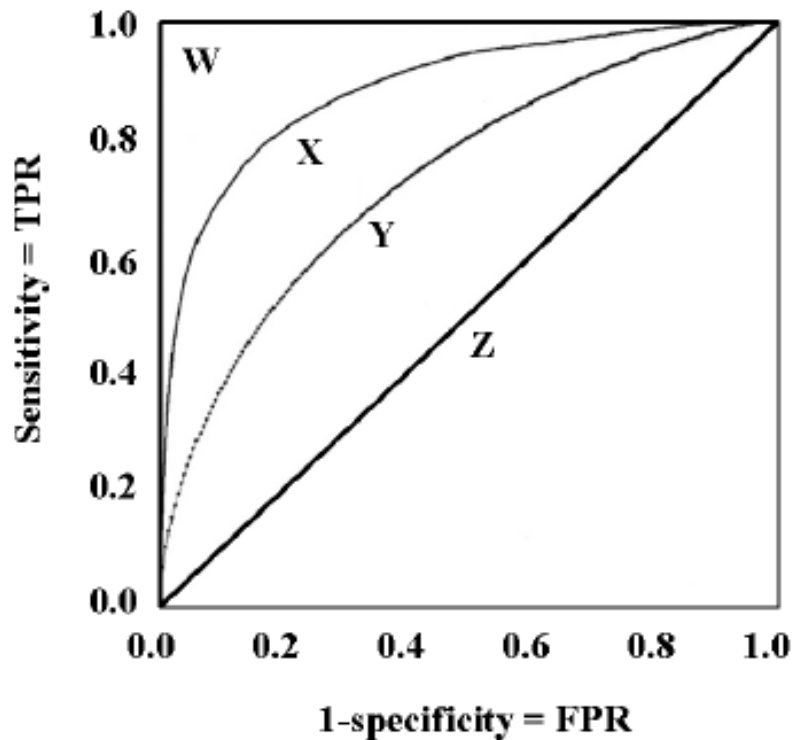


Figura 6: Ejemplo de curva ROC. El área bajo la curva indica que tan bien el modelo logra predecir los datos. En la figura, la curva X tiene una mayor área bajo la curva que Y , por lo que X predice mejor que Y . La curva W corresponde a un modelo con predicción perfecta y la curva Z corresponde a un modelo donde existe un 50% de probabilidad de predecir la clase correctamente. Fuente: https://www.researchgate.net/figure/Fitted-Receiver-Operating-Characteristic-ROC-curves-for-diagnostic-tests-with-varying_fig7_235328477

2.3. Decision Trees

Los Decision Tree (DT) son métodos de machine learning que se utilizan para tareas de regresión y clasificación. Esto se logra, estratificando o segmentando a través de decisiones el espacio de predicción y, como su nombre lo indica, este proceso se puede representar en un árbol.

Los Decision Tree para clasificación son construidos a partir de una muestras de datos, las cuales contienen objetos, cuyas clases son conocidas. Un Decision Tree se construye a partir de preguntas relacionadas con las características del objeto. A dichas preguntas se les llama nodo, y cada nodo apunta una cantidad de nodos hijos determinada por la cantidad de respuestas que haya a la pregunta. En específico, un *Binary Decision Tree* contiene solo nodos con respuesta Sí/No. Para ello se utilizan operaciones del tipo $=$ y $\neq$ para características de tipo categórica, y $<$, $\leq$, $>$ y $\geq$ para características de tipo continuo. Aunque es posible reali-

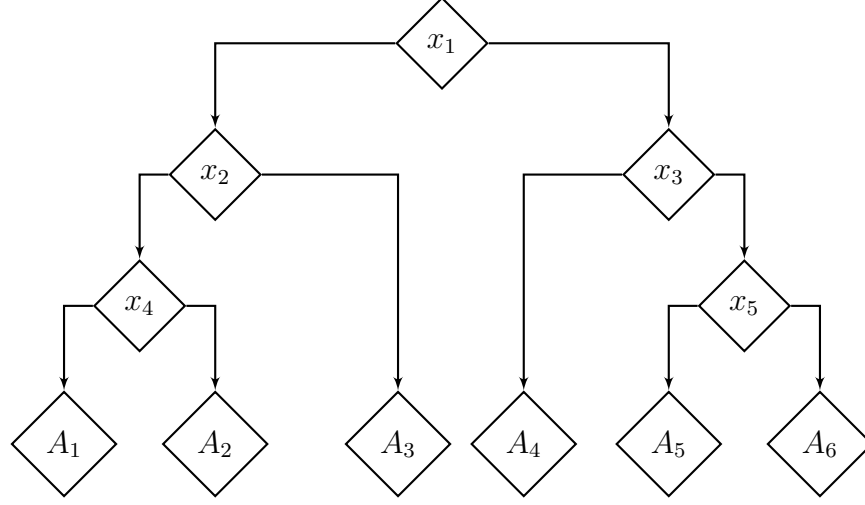


Figura 7: Estructura de un Binary Decision Tree. El nodo x_1 corresponde al nodo raíz y los nodos A_i corresponden a nodos hoja. Los nodos x_i representan condiciones, a través de las cuales se dividen los datos hasta obtener los subconjuntos A_i .

zar operaciones que evalúen más de una característica, por lo general cada nodo utiliza una sola característica para dividir el espacio del predicción.

Como se puede observar en la Figura 7, la muestra inicial se divide en dos subconjuntos o regiones cada vez que pasa por un nodo, a excepción del último nodo, el cual es llamado hoja. Idealmente, las preguntas realizadas separan de manera perfecta las categorías que se busca predecir, o, en su defecto, los subconjuntos son lo más homogéneos posible. De esta manera, cada una de las hojas corresponde a una categoría.

Durante el entrenamiento de un DT se busca encontrar una pregunta que minimice la heterogeneidad de los subconjuntos resultantes. Para ello se necesita una métrica que mida la homogeneidad de dichos conjuntos. Las dos métricas más utilizadas para esto son: la entropía y el índice Gini.

Sea E un conjunto de entrenamiento con m clases distintas y $p_i (i = 1, \dots, m)$ la fracción de objetos de E que pertenece a la clase i . Entonces la entropía $H(E)$, Eq. 1, de la distribución de probabilidad $(p_i)_{i=1}^m$ es una buena aproximación de la impureza del conjunto E .

$$H(E) = - \sum_{i=1}^m p_i \log p_i \quad (1)$$

Como se puede observar en la Ecuación 1, la entropía alcanza su punto mínimo cuando uno de los p_i es igual a 1 y los demás son 0, y alcanza su máximo cuando todos los p_i tienen el mismo valor. Por ende, minimizar la entropía equivale a minimizar la heterogeneidad del conjunto E .

$$G(E) = 1 - \sum_{i=1}^m p_i^2 \quad (2)$$

El índice Gini, Eq. 2, es 0 cuando el conjunto E contiene objetos de una sola clase, y por lo tanto, al igual que la entropía, minimizar el índice Gini equivale a minimizar la heterogeneidad del conjunto E .

Dada una métrica de impureza I , se elige una pregunta que minimiza la media ponderada de la impureza de las hojas resultantes. Esto es, si una pregunta con k posibles respuestas, divide E en los subconjuntos $E_1, \dots, E_k$, se escoge la pregunta que minimice:

$$\sum_{j=1}^k \frac{|E_j|}{|E|} I(E_j) \quad (3)$$

De esta manera, durante el entrenamiento, se enumeran todas las posibles preguntas que se pueden realizar para dividir el conjunto de entrenamiento, y se elige aquella pregunta que minimice la Ecuación 3.

Para predecir a qué categoría o clase pertenece un objeto determinado, cuya categoría es desconocida, se realizan distintas preguntas, siguiendo la estructura del árbol hasta llegar a una hoja. Esto permite que sea muy fácil interpretar los Decision Tree a diferencia de otros modelos donde no es tan claro cómo se realizan las predicciones. [Kingsford y Salzberg, 2008]

2.3.1. Boosted Decision Trees

Los árboles de decisión o Decision Tree tienen un gran desempeño cuando existen reglas claras que permiten dividir las muestras de entrenamientos en distintas clases. Sin embargo, cuando los problemas crecen en complejidad, y las reglas dejan de ser tan claras, Boosted Decision Tree (BDT) es una mejor opción.

Boosted Decision Tree es una técnica de ensamblado. Estas técnicas permiten combinar los resultados de varios Decision Tree para obtener una respuesta mucho más robusta. En específico, la técnica de *boosting* utiliza N modelos contruidos de forma secuencial y asigna un peso a cada objeto en la muestra de entrenamiento. Los pesos son utilizados para dar prioridad a aquellas muestras que han sido predichas de forma errónea en modelos pasados. De esta manera, una muestra que es predicha de forma errónea en el n -ésimo modelo, es impulsada (*boosted*), incrementando su peso antes de la construcción del $(n+1)$ -ésimo modelo.

Como se puede ver en la Figura 8, para predecir la categoría de una nueva muestra, cada

modelo realiza una predicción individual h_i que se pondera, basado en el *accuracy* α_i . Luego se utiliza una técnica de votación para decidir la categoría de la muestra [Myles *et al.*, 2004].

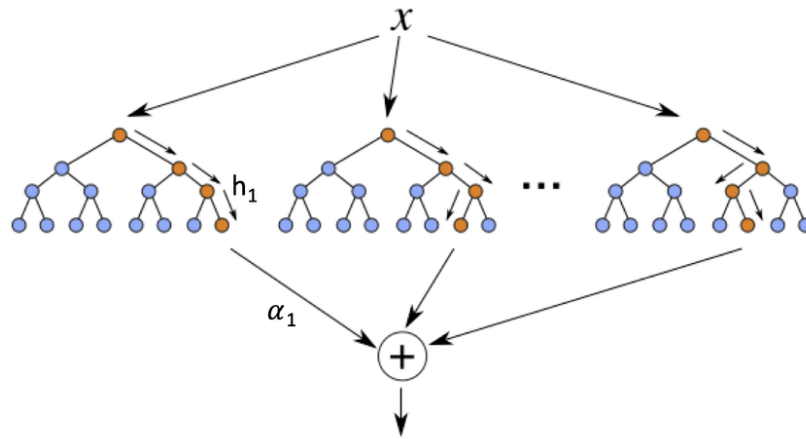


Figura 8: Diagrama de un Boosted Decision Tree. En esta técnica se utiliza un conjunto de Decision Trees, cuyas predicciones son ponderadas para obtener una predicción final. Fuente: Energy-Efficient Classification for Resource-Constrained Biomedical Applications - Figura Científica en ResearchGate. Disponible en: https://www.researchgate.net/figure/Schematic-diagram-of-a-boosted-ensemble-of-decision-trees_fig2_325632132 [Consulta: 5 diciembre 2021]

2.4. Redes Neuronales Artificiales

Las Artificial Neural Network (ANN), al igual que los BDT, son métodos de machine learning que se utilizan en problemas de regresión y clasificación. Esta técnica fue inspirada en las neuronas cerebrales, y en el hecho de que éstas se conectan entre sí para aprender.

En 1943 McCulloch y Pitts [McCulloch y Pitts, 1943] modelaron el funcionamiento de una neurona por primera vez. Ésta operaba como un switch, donde la neurona se activa o desactiva dependiendo de las entradas. Cada entrada tiene un peso asociado, el cual imita la sinapsis, excitando o inhibiendo la señal de entrada. En 1960 Rosenblatt [Rosenblatt, 1960] demostró que este tipo de neuronas llamadas perceptron tienen propiedades parecidas a las del cerebro, y una red simple de estas neuronas puede aprender a partir de ejemplos. En 1969 Minsky y Papert [Minsky y Papert, e MA] demostraron que el perceptron solo puede resolver problemas linealmente separables. A fines de los 80, el método de error *backpropagation* permitió que las redes neuronales pudieran ser aplicadas a problemas linealmente no separables.

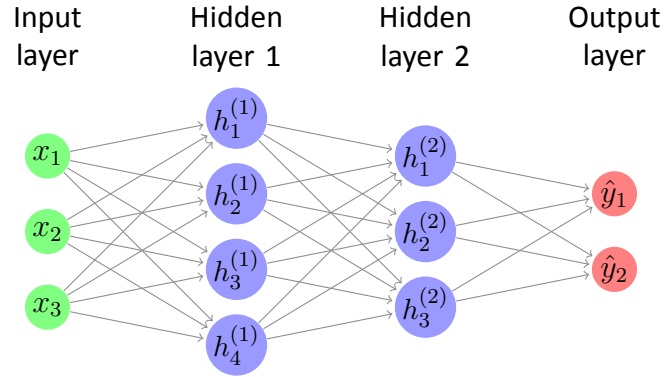


Figura 9: Red Neuronal Profunda. Cada nodo pondera los valores de la capa anterior y aplica una transformación para obtener el valor que pasará a la capa siguiente. Este proceso se repite hasta obtener la predicción en la capa de salida.

2.4.1. Redes Feed-Forward

Las redes *feed-forward* son redes secuenciales, lo cual quiere decir que las neuronas de una capa están conectadas solo a las neuronas de la capa siguiente, a diferencia de otro tipo de redes, donde pueden existir conexiones entre capas no consecutivas o conexiones a capas previas.

Como se puede ver en la Figura 9, una red neuronal utiliza como entrada las características de un objeto. Cada neurona en la primera capa pondera las características del objeto y luego filtra su ponderación a través de una función de activación. El valor obtenido se utiliza como entrada de la siguiente capa. Este proceso se repite hasta llegar a la última capa, la cual contiene tantas neuronas como clases se quieran predecir, o, una sola neurona en problemas de regresión o clasificación binaria. La función de activación de la última capa cambia, dependiendo del tipo de problema que se esté resolviendo. En problemas de regresión se suelen utilizar funciones lineales, pero se pueden utilizar otras funciones, dependiendo del problema; En problemas de clasificación binaria se utiliza la función sigmoid, y en problemas de clasificación multi-clases se utiliza la función softmax.

Durante el entrenamiento se busca encontrar los pesos que minimicen el error de predicción. Para ello, se mide la diferencia entre la salida de la red y la observación real, la cual se conoce con anterioridad, y se ajustan los pesos en relación a esta diferencia para mejorar el desempeño de la red. Este proceso se repite hasta estar satisfecho con las predicciones que realiza la red. Existen varios algoritmos que permiten lograr el proceso mencionado anteriormente, sin embargo, el más utilizado es *backpropagation*.

En *backpropagation* [Rumelhart *et al.*, 1986] no solo se mide la diferencia entre la salida estimada por la red y la observación real, sino que se calcula el error de cada neurona. Para ello es necesario conocer la salida esperada de cada neurona, la cual se calcula en base a los pesos de la red y la salida esperada de la capa siguiente. De esta manera, el cálculo parte con

la observación real y se propaga hacia atrás, obteniendo la salida esperada de cada neurona. Una vez obtenidos estos valores, se calcula la diferencia al cuadrado entre la salida esperada y la salida estimada por la neurona. Luego, se suman todas estas diferencias para obtener el error total o pérdida. Finalmente, se utiliza un optimizador para cambiar el valor de los pesos y minimizar la pérdida [Krogh, 2008]. En el algoritmo de *backpropagation* clásico se utiliza método de descenso de gradiente o *gradient descent* como optimizador, sin embargo, actualmente existen otras alternativas, siendo Adam una de las más populares por ser una extensión de *gradient descent* que lo hace más eficiente.

2.4.2. Hiperparámetros

Tanto la cantidad de capas como la cantidad de neuronas por capa varía dependiendo del problema que se quiere resolver. Este tipo de parámetros que varía de problema a problema, se les llama hiperparámetros. A continuación, se describen algunos de los hiperparámetros existentes.

- **Cantidad de capas ocultas:** Corresponde a la cantidad de capas de la red que no son de entrada o salida.
- **Cantidad de neuronas por capa:** Corresponde a la cantidad de neuronas que hay en cada capa de la red.
- **Función de activación:** Corresponde a la función de activación que se utiliza en cada capa de la red.
- **Optimizador:** Corresponde al optimizador de la pérdida que se utiliza durante el entrenamiento.
- **Learning rate:** Valor de ajuste que utilizan algunos optimizadores durante la modificación de pesos.
- **Número de épocas:** Número de veces que pasa el conjunto de datos completo por la red.
- **Batch size:** Tamaño de los sub-conjuntos de datos que se crean durante el entrenamiento. Un tamaño de *batch size* más grande implica un menor tiempo de entrenamiento, puesto que reduce la cantidad de veces que se calculan los errores.

2.4.3. Funciones de activación

Como se mencionó anteriormente, una neurona pondera sus entradas y se activa dependiendo del valor obtenido. El perceptrón utilizaba la función *binary step* para decidir si la neurona está activa o no, sin embargo, con la llegada de *backpropagation* se comenzó a

utilizar distintas funciones de activación. Esto permitió separar el espacio de predicción de forma no lineal, obteniendo predicciones más complejas.

A continuación se listan las funciones de activación implementadas en Keras:

- Rectified Linear Unit (ReLU)

$$f(x) = \begin{cases} 0 & \text{if } x > 0 \\ x & \text{if } x \leq 0 \end{cases}$$

- Sigmoid

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

- Softmax

$$\sigma(x)_i = \frac{e^{x_i}}{\sum_{j=1}^K e^{x_j}}$$

- Softplus

$$f(x) = \ln(1 + e^x)$$

- Softsign

$$f(x) = \frac{x}{|x| + 1}$$

- Tangente Hiperbólica

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

- Scaled Exponential Linear Unit (SELU)

$$f(x) = \begin{cases} \lambda \alpha (e^x - 1) & \text{if } x < 0 \\ \lambda x & \text{if } x \geq 0 \end{cases}, \text{ con } \alpha = 1.67326324 \text{ y } \lambda = 1.05070098$$

- Exponential linear unit (ELU)

$$f(x) = \begin{cases} \alpha (e^x - 1) & \text{if } x \leq 0 \\ x & \text{if } x > 0 \end{cases}$$

- Exponential

$$f(x) = e^x$$

CAPÍTULO 3

PROPUESTA DE SOLUCIÓN

En este capítulo se describe en profundidad todo el trabajo realizado. La Figura 10 muestra de forma general las tareas realizadas y el flujo de trabajo utilizado. El flujo de trabajo fue dividido en 5 categorías: obtención de los datos, análisis de los datos, pre-procesamiento de los datos, entrenamiento del modelo Fully Connected Neural Network (FCNN), validación del modelo FCNN. El código desarrollado durante este trabajo puede ser encontrado en el repositorio [Zavala, 2021], junto a los pesos de las mejores redes encontradas.

En la Sección 3.1 se describe el proceso que se realizó para transformar datos provenientes de ATLAS a un conjunto con las variables necesarias para el entrenamiento. En la Sección 3.2 se presenta el análisis realizado para determinar la validez de los datos. En la Sección 3.3 se describe el pre-procesamiento realizado a los datos para dejar aquellos datos relevantes para abordar el problema, la división en conjuntos de entrenamiento y prueba, y el sub-muestreo o *undersampling* realizado a los datos. En la Sección 3.4 se presenta el algoritmo desarrollado en este trabajo para la construcción de los modelos basados en deep learning. Esta tarea incluyó el diseño e implementación de distintas arquitecturas que incluían diversos número de capas y neuronas por capas, junto con la búsqueda de hiperparámetros óptimos, utilizando tres espacios de búsqueda diferentes, de manera de encontrar los mejores resultados.

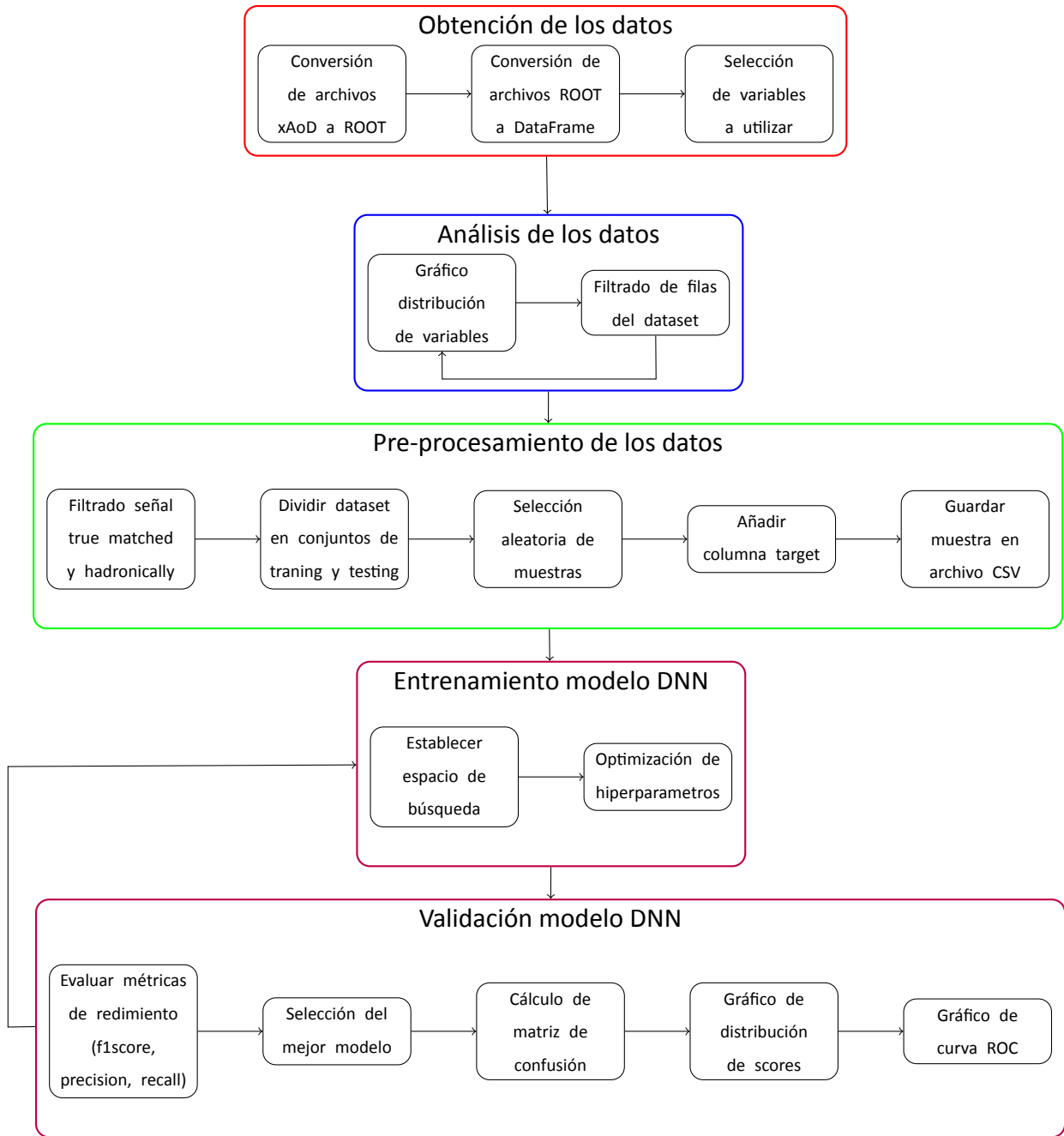


Figura 10: Proceso utilizado para el desarrollo de este trabajo.

3.1. Obtención de los datos

Esta sección describe los pasos realizados para la generación del conjunto de datos utilizados en este trabajo.

Los datos utilizados para señal provienen del modo de decaimiento del Gravitón⁹ $G \rightarrow hh \rightarrow 4\tau$ y los datos para background provienen de eventos donde se producen múltiples *jets* de hadrones. Los datos fueron obtenidos por el equipo de ATLAS mediante simulaciones de Monte Carlo. MadGraph [Alwall *et al.*, 2011] es un software que es utilizado para simular el evento principal de una colisión proton-proton y Pythia8 [Sjöstrand *et al.*, 2015] para simular la fragmentación y decaimiento de las partículas. El algoritmo anti- k_t [Cacciari *et al.*, 2008a] es utilizado para la formación de los *jets* necesario para la reconstrucción de los objetos con topología *di-tau*, es decir, un *seed-jet* anti- k_T ($R = 1.0$) grande y otros *jets* anti- k_T ($R = 0.2$) más pequeños (*sub-jets*). Estos *jets* son utilizados como entrada para el algoritmo de reconstrucción de *di-taus*.

El conjunto de datos es almacenado en archivos xAOD (Analysis Object Data) [Buckley *et al.*, 2015] en la *Grid*¹⁰ de los experimentos del LHC por miembros de la colaboración ATLAS. Este tipo de archivos contiene un resumen de eventos reconstruidos proveniente de simulaciones, o datos reales adquiridos mediante detectores, e información suficiente para los análisis más comunes.

Estos archivos fueron utilizados para calcular las variables asociadas a este problema mediante THOR¹¹, el cual es un paquete que provee un entorno fácil de utilizar para desarrollar algoritmos de reconstrucción e identificación para el decaimiento hadrónico del leptón tau.

Este trabajo fue realizado en conjunto con ATLAS UTFSM. Además de esto el conjunto de datos fue transformado a archivos ROOT utilizando esta misma herramienta. En la Tabla 1 se puede observar los nombres de los archivos obtenidos.

⁹Partícula hipotética de spin 2 que aparece en teorías con dimensiones extras

¹⁰Tecnología que permite utilizar de forma coordinada recursos heterogéneos (como procesadores, almacenamiento y aplicaciones específicas) que no están sujetos a un control centralizado

¹¹<https://gitlab.cern.ch/atlas-perf-tau/THOR>

Nombres datasets

Pythia8EvtGen_A14NNPDF23LO_jetjet_JZ12W_v10.root
 Pythia8EvtGen_A14NNPDF23LO_jetjet_JZ4W_v10.root
 Pythia8EvtGen_A14NNPDF23LO_jetjet_JZ5W_v10.root
 Pythia8EvtGen_A14NNPDF23LO_jetjet_JZ7W_v10.root
 MadGraphPythia8EvtGen_A14NNPDF23LO_RS_G_hh_4tau_c10_M1500.root
 MadGraphPythia8EvtGen_A14NNPDF23LO_RS_G_hh_4tau_c10_M1750.root
 MadGraphPythia8EvtGen_A14NNPDF23LO_RS_G_hh_4tau_c10_M2000.root
 MadGraphPythia8EvtGen_A14NNPDF23LO_RS_G_hh_4tau_c10_M2250.root
 MadGraphPythia8EvtGen_A14NNPDF23LO_RS_G_hh_4tau_c10_M2500.root

Tabla 1: Nombre de los datasets utilizados.

Los archivos ROOT almacenan datos en forma de árbol, donde cada rama representa una variable distinta y sus valores. Se utilizó la librería *uproot*¹² para leer los archivos ROOT y transformarlos a DataFrames de *pandas*¹³. Debido a la naturaleza de árbol de los archivos ROOT, al realizar la transformación a DataFrame, cada fila de la tabla resultante representaba uno de los eventos simulados, existiendo más de un objeto *di-tau* por fila. Como el objetivo de este trabajo es la identificación de objetos *di-tau*, se decidió dejar un objeto por fila y una variable por columna.

Una vez obtenido el DataFrame se seleccionaron las variables a utilizar basado en [The ATLAS Collaboration *et al.*, 2020], donde se describen las variables utilizadas en la solución actual.

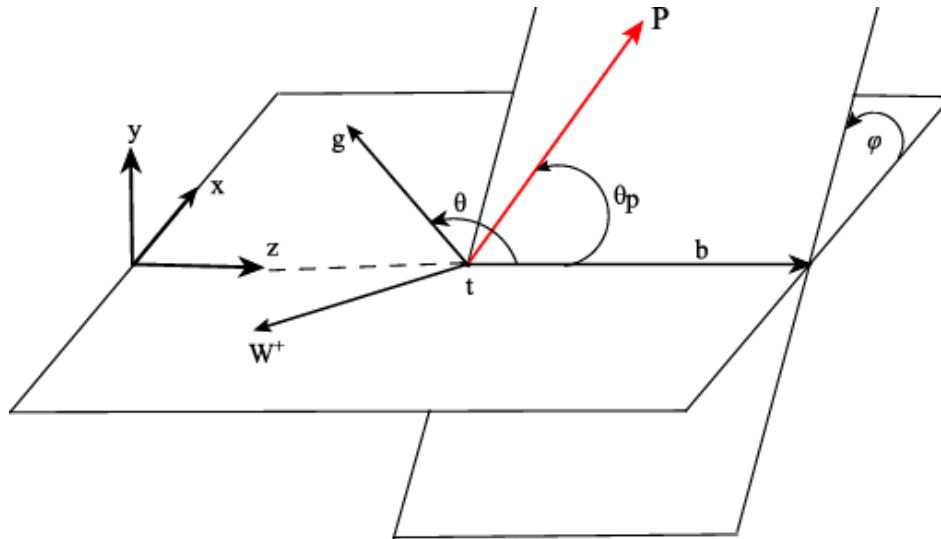


Figura 11: Sistema de coordenadas de ATLAS. El eje x apunta hacia el centro del LHC, el eje y , hacia la superficie de la tierra y el eje z , en la dirección en la que circulan los haces de protones.

¹²<https://uproot.readthedocs.io/en/latest/>

¹³<https://pandas.pydata.org/>

Para entender algunas de las variables es necesario conocer la geometría del detector. El origen del sistema de coordenadas del detector ATLAS es colocado en el punto de colisión proton-proton en el centro del detector. Se utiliza un sistema cartesiano de mano derecha. El eje z está definido a lo largo del conducto por donde circulan los haces de protones. El eje x apunta hacia el centro del LHC y el eje y apunta hacia la superficie de la tierra. El plano $x - y$ es transversal al conducto por el que circulan los haces y las variables proyectadas a este plano son llamadas transversales (Ej. momentum transversal p_T). El ángulo azimuthal ϕ es medido desde el eje x , y va alrededor del eje z (Ver Tabla 11). El ángulo polar θ se mide desde el eje z , aunque, en vez de θ se suele utilizar la *pseudorapidity* como coordenada para las partículas. La *pseudorapidity* es definida como:

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

Para medir la distancia entre dos partículas se utiliza la distancia angular *pseudorapidity-azimuthal* ΔR , definida como:

$$\Delta R = \sqrt{(\eta_1 - \eta_2)^2 + (\phi_1 - \phi_2)^2}$$

Se seleccionaron las mismas variables utilizadas en la solución actual. A continuación se definen las variables a utilizar:

- **ditaujets_f_core_lead:** Relación entre la energía depositada en el núcleo y la de todo el cono del *sub-jet* con p_T más alto (sj_1).

$$\frac{E_{\Delta R < 0.1}^{sj_1}}{E_{\Delta R < 0.2}^{sj_1}}$$

- **ditaujets_f_core_subl:** Relación entre la energía depositada en el núcleo y la de todo el cono del *sub-jet* con el segundo p_T más alto (sj_2).

$$\frac{E_{\Delta R < 0.1}^{sj_2}}{E_{\Delta R < 0.2}^{sj_2}}$$

- **ditaujets_f_subjet_subl:** Relación del p_T de sj_2 con el p_T del di- τ semilla del *jet* de radio grande.

$$\frac{p_T^{sj_2}}{p_T^{LRJ}}$$

- **ditaujets_f_subjets:** Relación de la suma escalar de p_T de los dos *sub-jets* principales con el p_T del di- τ semilla del *jet* de radio grande.

$$\frac{p_T^{sj_1} + p_T^{sj_2}}{p_T^{LRJ}}$$

- **ditaujets_f_isotracks:** Logaritmo del cociente de la suma del escalar p_T de los *iso-tracks*¹⁴ con el p_T del di- τ semilla del *jet* de radio grande.

$$\log \left(\sum p_T^{iso-tracks} / p_T^{LRJ} \right)$$

- **ditaujets_r_max_lead:** Mayor separación del track con el eje de su *sub-jet* asociado, para el *sub-jet* sj_1 .

$$\Delta R_{max}(\text{track}, sj_1)$$

- **ditaujets_r_max_subl:** Mayor separación del *track* con el eje de su *sub-jet* asociado, para el *sub-jet* sj_2 .

$$\Delta R_{max}(\text{track}, sj_2)$$

- **ditaujets_r_tracks_subl:** Suma p_T -ponderada del ΔR entre los tracks asociados a sj_2 con respecto a su eje.

$$\frac{\sum [p_T^{track} \Delta R(\text{track}, sj_2)]}{\sum p_T^{track}}$$

- **ditaujets_r_isotrack:** Suma p_T -ponderada del ΔR entre los *iso-tracks* y el eje del *sub-jet* más cercano.

$$\frac{\sum [p_T^{iso-track} \Delta R(\text{iso-track}, sj)]}{\sum p_T^{iso-track}}$$

- **ditaujets_m_core_lead:** Logaritmo de la masa invariante de los *tracks* en el núcleo de sj_1 .

$$\log \left(m_{\Delta R < 0.1}^{tracks, sj_1} \right)$$

- **ditaujets_m_core_subl:** Logaritmo de la masa invariante de los *tracks* en el núcleo de sj_2 .

$$\log \left(m_{\Delta R < 0.1}^{tracks, sj_2} \right)$$

- **ditaujets_m_tracks_lead:** Logaritmo de la masa invariante de los *tracks* con $\Delta R < 0.2$ desde el eje de sj_1 .

$$\log \left(m_{\Delta R < 0.2}^{tracks, sj_1} \right)$$

- **ditaujets_m_tracks_subl:** Logaritmo de la masa invariante de los *tracks* con $\Delta R < 0.2$ desde el eje de sj_2 .

$$\log \left(m_{\Delta R < 0.2}^{tracks, sj_2} \right)$$

- **ditaujets_d0_leadtrack_lead:** Logaritmo de la distancia más cercana en el plano transversal entre el vértice principal y el *leading track*¹⁵ de sj_1 .

$$\log \left(|d_{0, \text{lead-track}}^{sj_1}| \right)$$

¹⁴Tracks asociados a la región de aislamiento, es decir, tracks que no están asociados a ningún sub-jet.

¹⁵Track de mayor p_T

- **ditaujets_d0_leadtrack_subl**: Logaritmo de la distancia más cercana en el plano transversal entre el vértice principal y el *leading track* de sj_2 .

$$\log(|d_{0,\text{lead-track}}^{sj_2}|)$$

- **ditaujets_n_tracks_lead**: Número de *tracks* asociados a sj_1 .

$$n_{\text{tracks}}^{sj_1}$$

- **ditaujets_n_subjets**: Número de *tracks* asociados a todos los *sub-jets*.

$$n_{\text{tracks}}^{\text{sub-jets}}$$

Finalmente, además de las variables mencionadas anteriormente, se agregaron al DataFrame variables para utilizar en etapas siguientes:

- **ditaujets_jetbdt**: Puntuación obtenida para cada dato al realizar la identificación utilizando el modelo BDT del estado del arte. Las puntuaciones son utilizadas más adelante para determinar la clase de un dato y poder calcular las métricas de rendimiento.
- **ditaujets_pt**: Momentum transversal del objeto.
- **ditaujets_n_subjets**: Número de total de *sub-jets*.
- **ditaujets_n_tracks_subl**: Número de *tracks* asociados a sj_2 .
- **ditaujets_istruthmatched**: Objetos asociados geoméricamente con alguno de los *di-tau* producidos en las simulaciones.
- **ditaujets_istruthhadronic**: Objetos con decaimiento hadrónico.

3.2. Análisis de los datos

Una vez seleccionadas las variables se realizaron gráficos para la distribución de cada variable. Las Figuras 12 y 13 muestran los gráficos obtenidos al unir todos los archivos de background y señal en dos distribuciones distintas. Estos gráficos fueron comparados con las distribuciones presentadas en [Kirchmeier, 2015] y se concluyó que las variables:

- *ditaujets_f_core_subl*
- *ditaujets_f_subjet_subl*
- *ditaujets_f_subjets*

- *ditaujets_r_max_subl*
- *ditaujets_r_tracks_subl*
- *ditaujets_m_core_subl*
- *ditaujets_m_tracks_subl*
- *ditaujets_f_subjets*
- *ditaujets_d0_leadtrack_subl*

tienen una mayor cantidad de objetos en 0, en comparación. Es por esto que se quitaron los objetos con menos de 2 *sub-jets* y con 0 *lead tracks*, debido a que estas variables necesitan de este mínimo de *sub-jets* y *lead tracks* para ser calculadas. Las Figuras 14 y 15 muestra la distribución de cada variable después de reducir la cantidad de objetos.

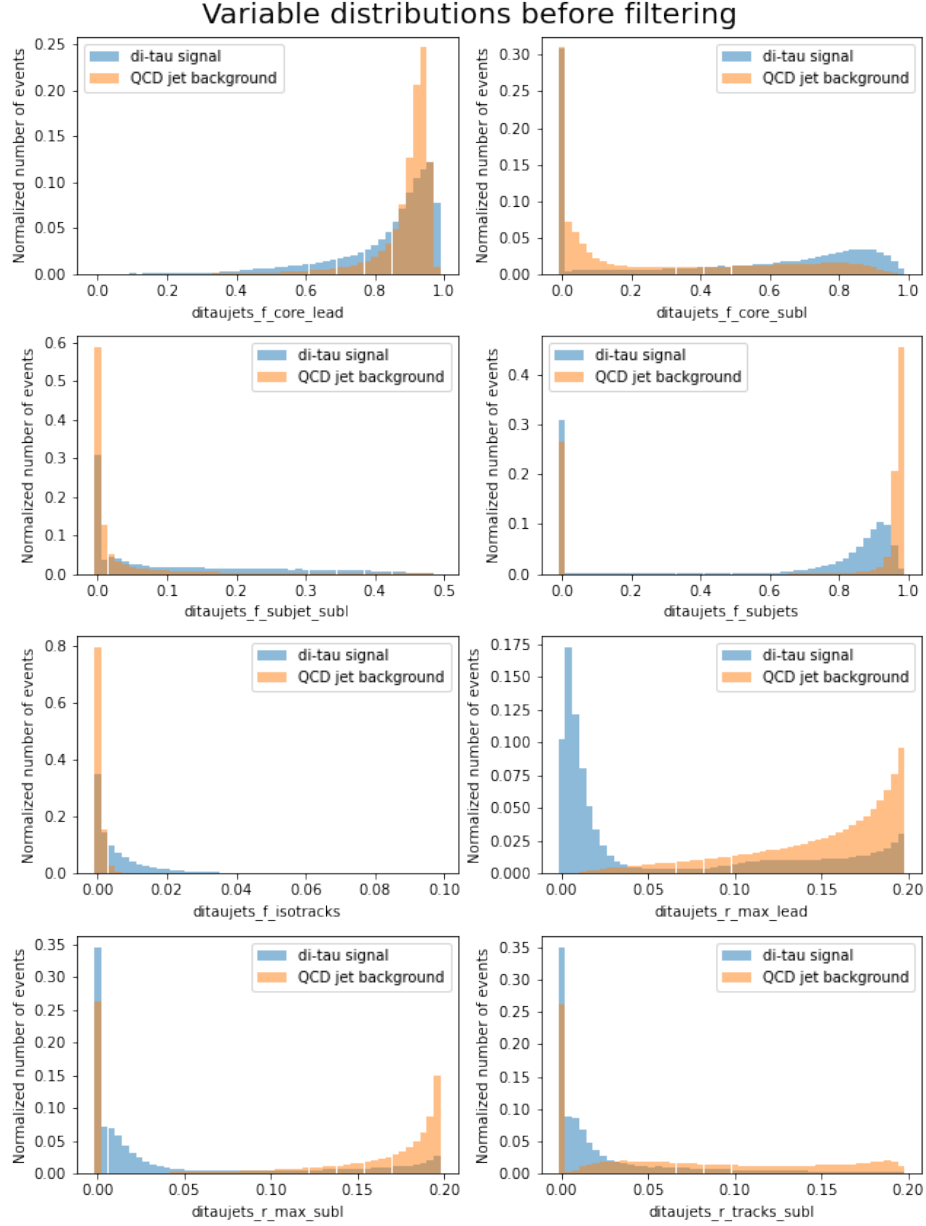


Figura 12: Histogramas de la distribución de las variables antes de realizar el filtrado de objetos di-tau. La distribución azul corresponde a la señal di-tau y la distribución naranja corresponde al background de QCD jet.

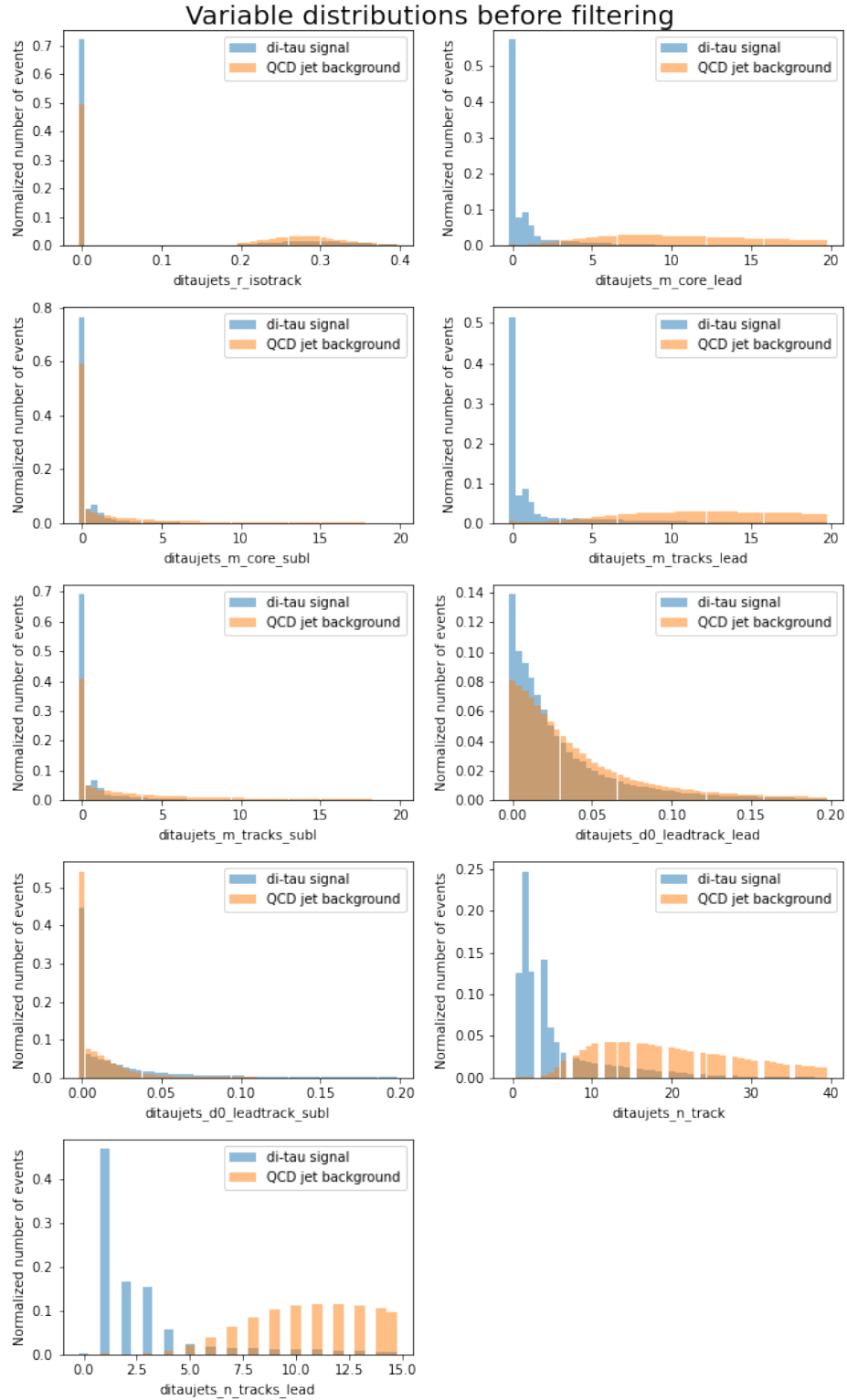


Figura 13: Histogramas de la distribución de las variables antes de realizar el filtrado de objetos di-tau. La distribución azul corresponde a la señal di-tau y la distribución naranja corresponde al background de QCD jet.

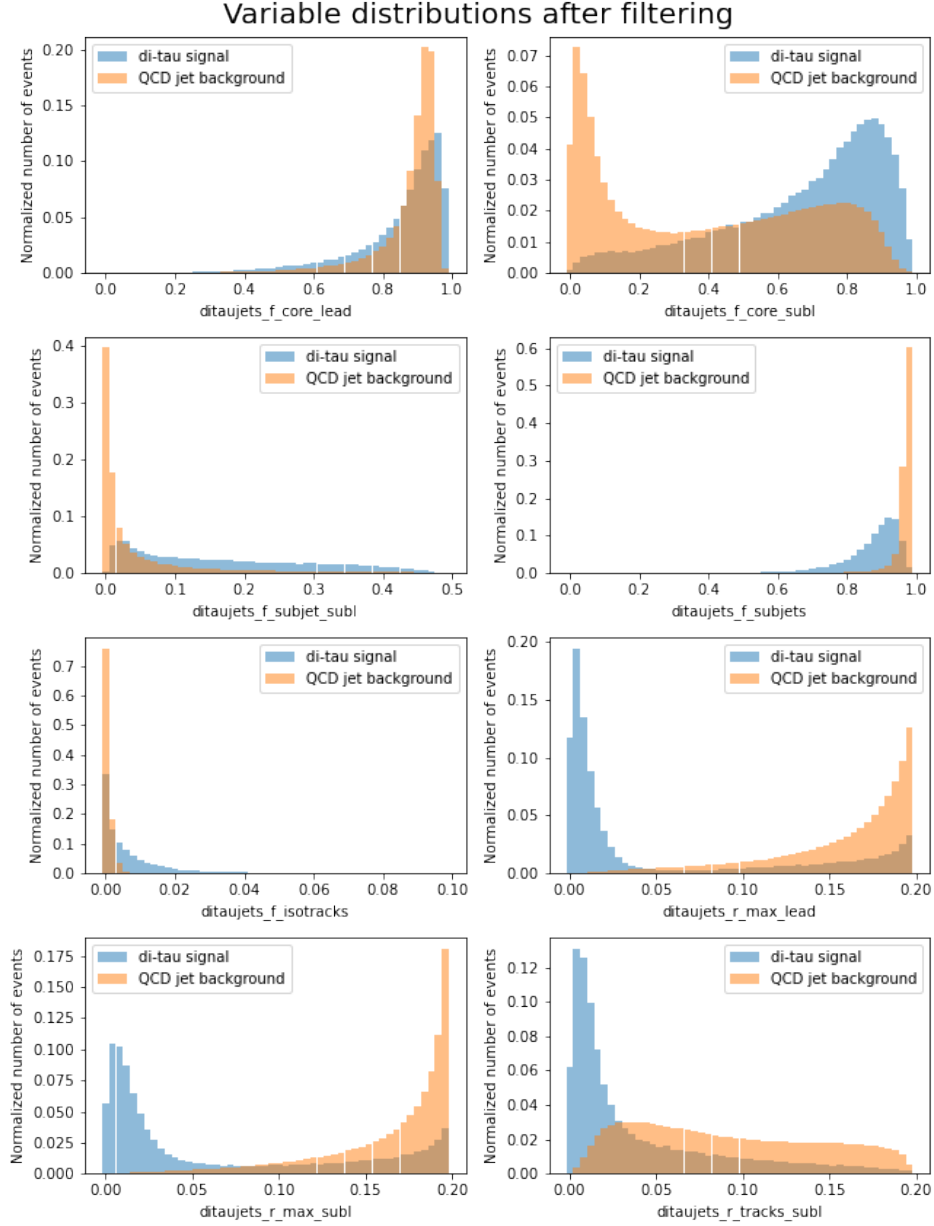


Figura 14: Histogramas de la distribución de las variables después de realizar el filtrado de objetos di-tau. La distribución azul corresponde a la señal di-tau y la distribución naranja corresponde a el background QCD jet.

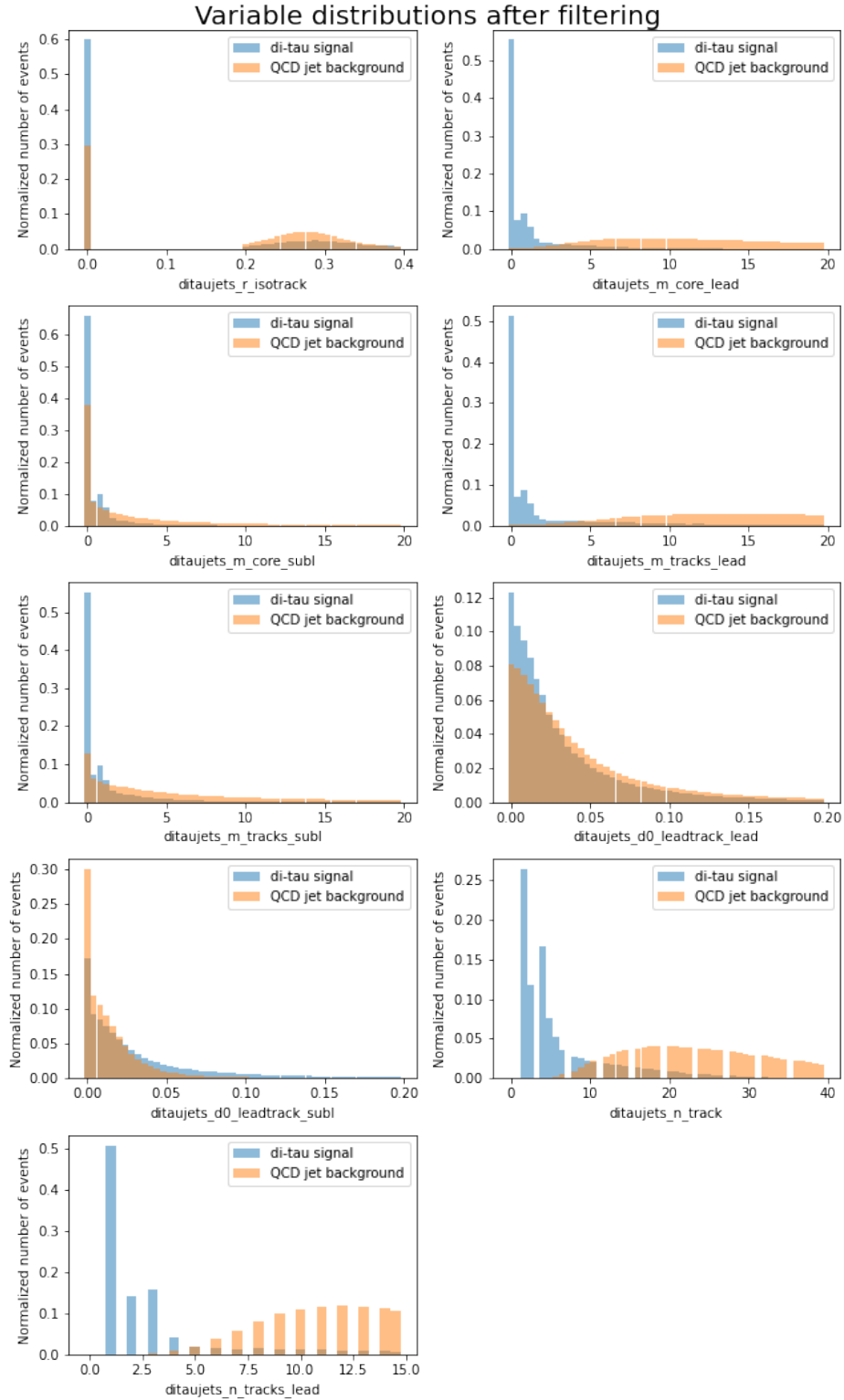


Figura 15: Histogramas de la distribución de las variables después de realizar el filtrado de objetos di-tau. La distribución azul corresponde a la señal di-tau y la distribución naranja corresponde a el background QCD jet.

3.3. Pre-procesamiento de los datos

En esta etapa se realizó un filtrado de los archivos de señal dejando aquellos objetos que cumplían con las condiciones $ditaujets_istruthmatched = 1$ y $ditaujets_istruthhadronic = 1$. La primera condición quiere decir que se dejaron aquellos objetos $di\text{-}taus$ que estaban asociados geoméricamente con alguno de los $di\text{-}taus$ producidos en las simulaciones, es decir, objetos correctamente reconstruidos. La segunda condición quiere decir que el objeto $di\text{-}tau$ corresponde a un $di\text{-}tau$ con decaimiento hadrónico.

Después se separó cada DataFrame (uno por cada archivo ROOT) en conjuntos de entrenamiento y prueba. Se utilizó el 80 % de los datos para entrenamiento y 20 % para prueba. Estos conjuntos fueron guardados en archivos CSV para su posterior uso. Las Figuras 16 y 17 muestran la cantidad de datos por cada archivo.

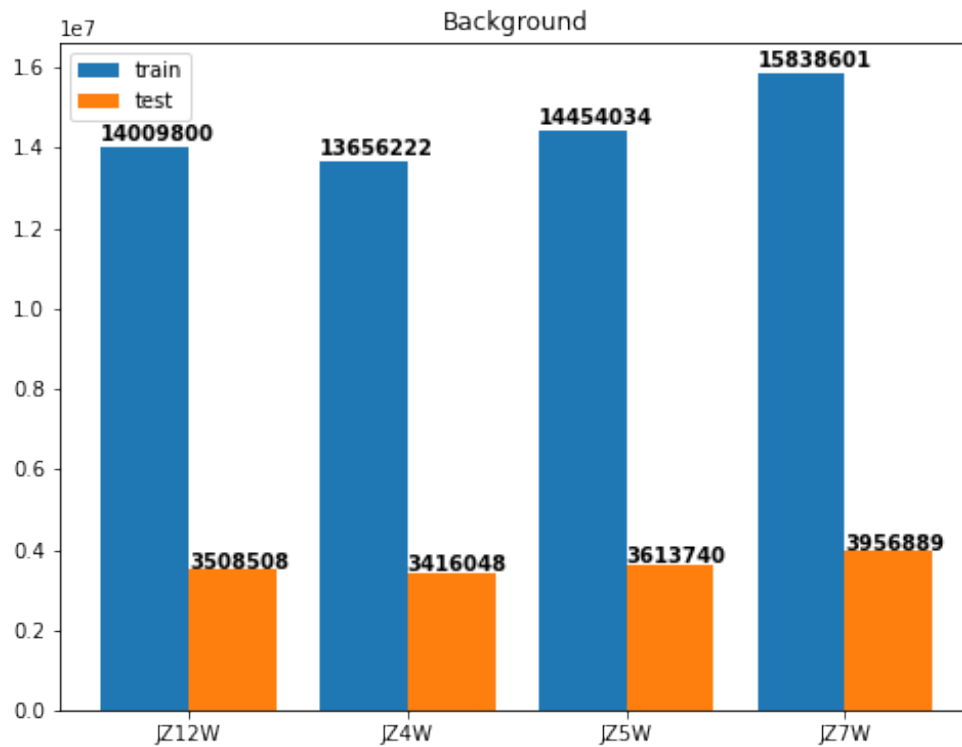


Figura 16: Cantidad de datos de background por cada archivo CSV.

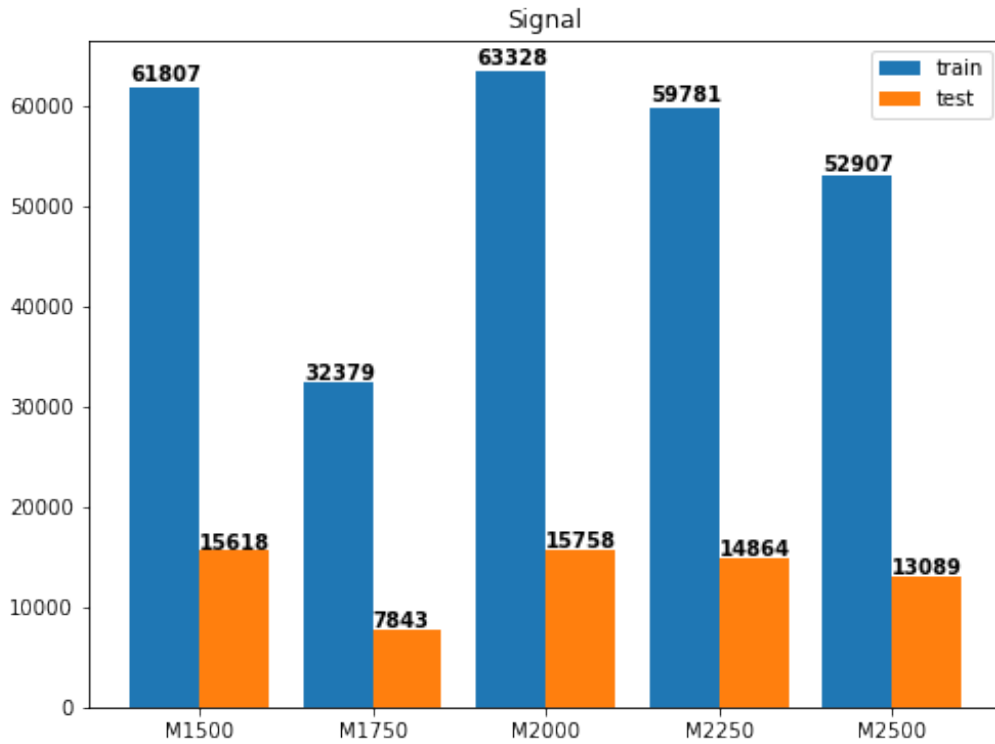


Figura 17: Cantidad de datos de señal por cada archivo CSV.

La Figura 18 muestra cantidad total de datos de señal comparados con al cantidad total de datos de background. En esta se puede observar un claro desbalance de los datos, lo cual puede traer complicaciones en el entrenamiento, por lo que se decidió realizar un sub-muestreo o *undersampling* de los datos. Para ello se tomó la cantidad total de señal y una cantidad equivalente de background para dejarlos en proporción 50/50. Se seleccionaron 67551 datos de manera aleatoria en cada uno de los archivos de background para entrenamiento y se seleccionaron 16793 datos de manera aleatoria en cada uno de los archivos de background para prueba. Finalmente se juntó los datos de señal y background en un conjunto para entrenamiento y uno para prueba. Se agregó una columna *target* para identificar los objetos de señal de los de background, siendo 1 la señal y 0 el background. Estos conjuntos se guardaron en archivos CSV.

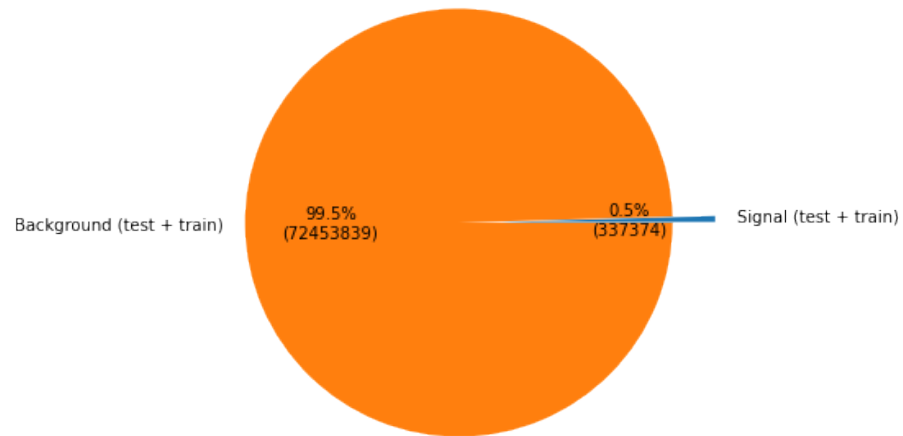


Figura 18: Comparación entre la cantidad total de datos de señal y background.

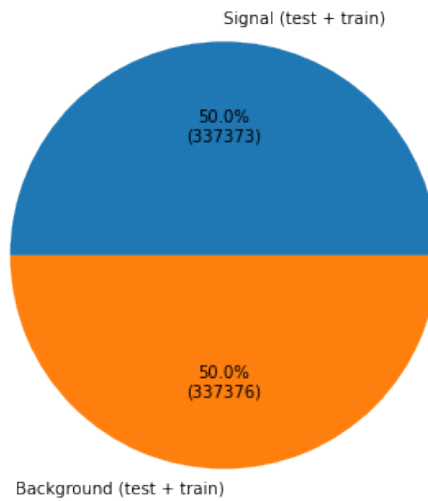


Figura 19: Comparación entre la cantidad de datos de señal y background después de realizar undersampling.

La Figura 19 muestra la cantidad de señal en comparación con la cantidad de background después de realizar *undersampling*. La Figura 20 muestra la cantidad de datos provenientes de cada archivo, con una cantidad total de datos para entrenamiento de 540406 y una cantidad total de datos para prueba de 134344. Las Figuras 21 y 22 muestra la distribución de las variables para el conjunto de entrenamiento, donde se puede observar que la forma en la distribución de las variables se sigue conservando.

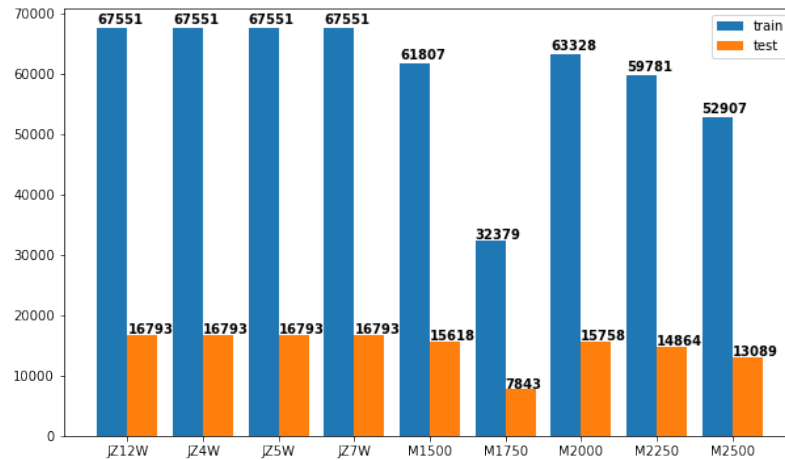


Figura 20: Cantidad de datos provenientes de cada archivo para conformar la muestra a utilizar.

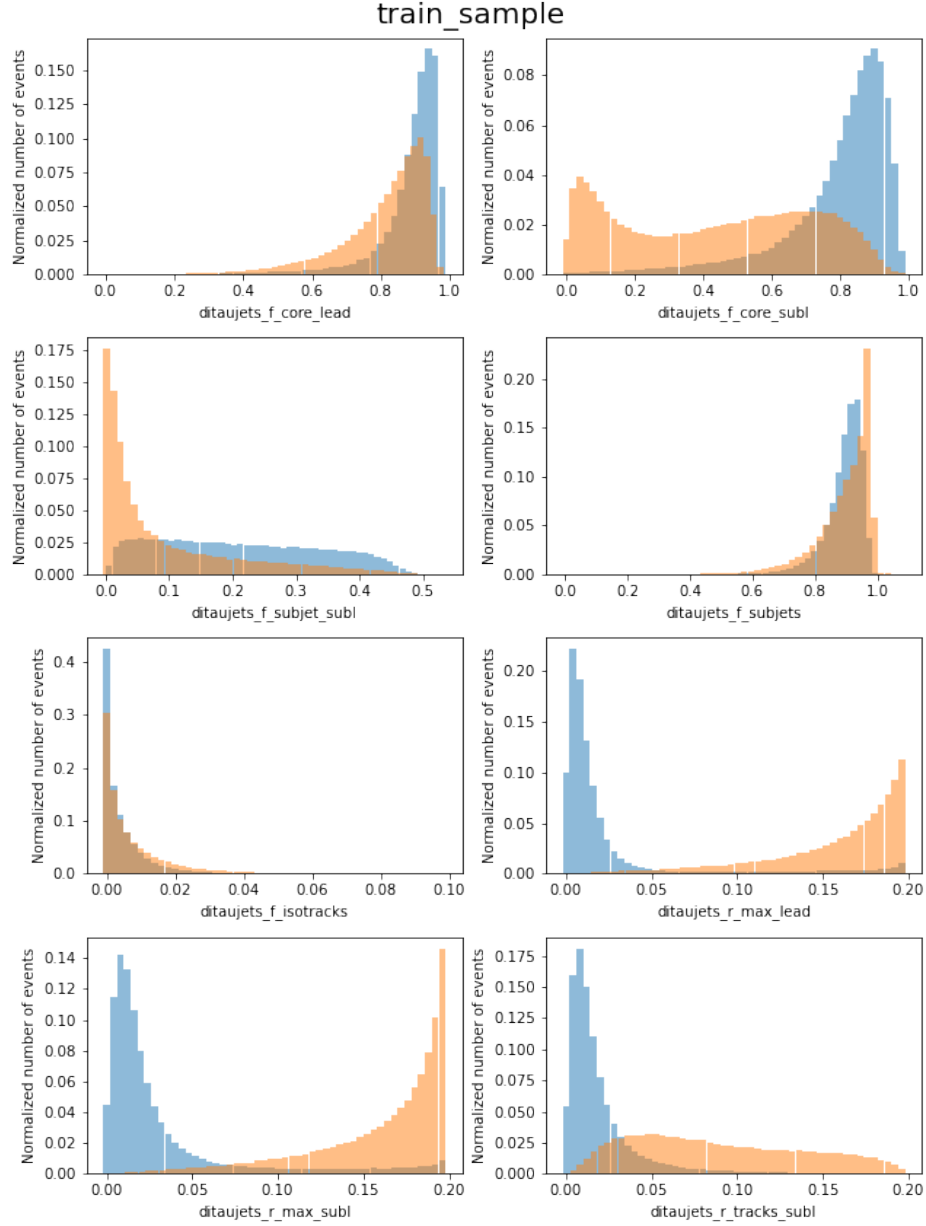


Figura 21: Histogramas de la distribución de las variables de la muestra utilizada para entrenamiento. La distribución azul corresponde a la señal di-tau y la distribución naranja corresponde al background de QCD jet.

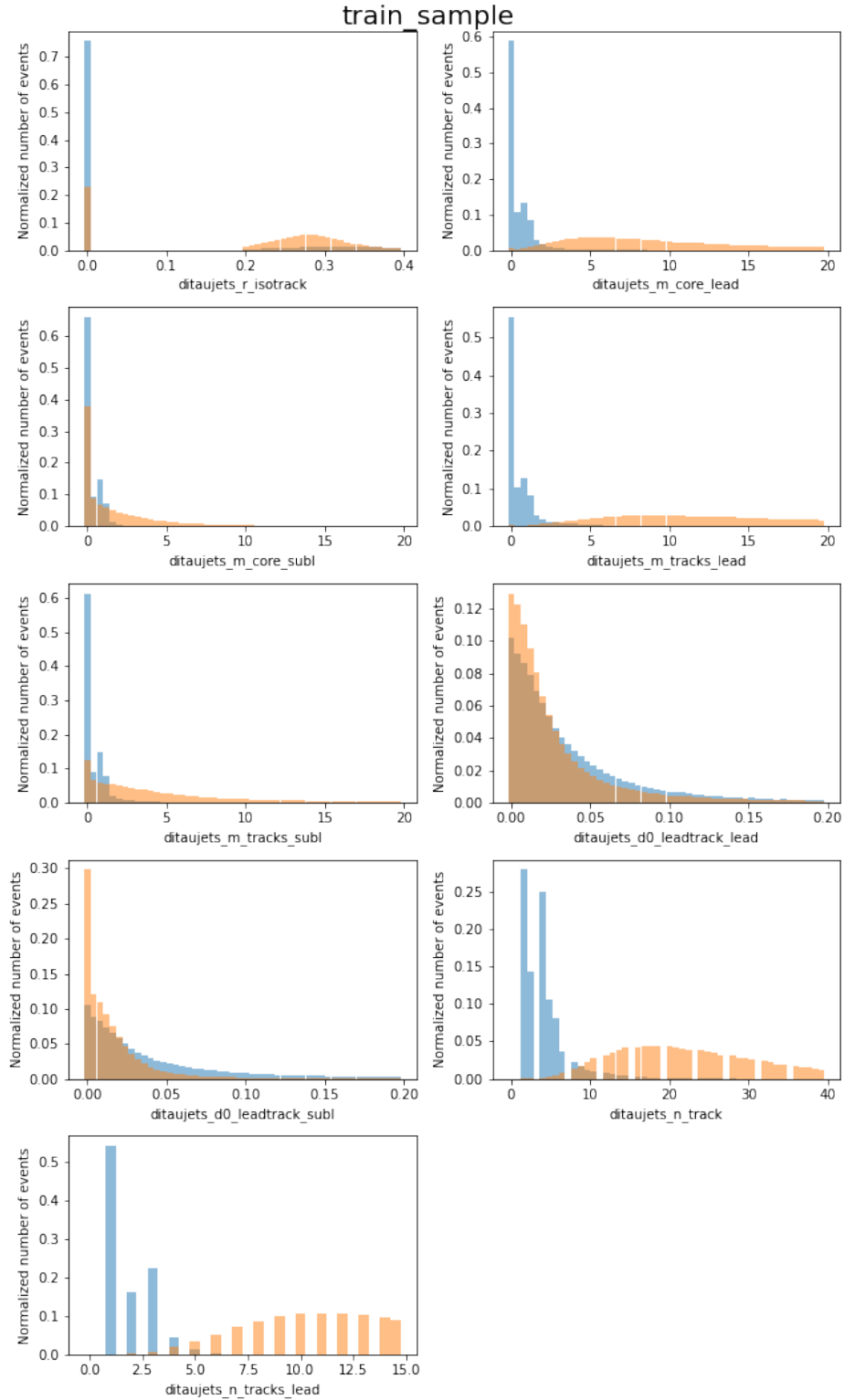


Figura 22: Histogramas de la distribución de las variables utilizadas.

3.4. Modelo propuesto y búsqueda de hiperparámetros

Se utilizó una red neuronal completamente conectada para resolver este problema y se realizaron 3 búsquedas distintas para encontrar los hiperparámetros que tuvieran un mejor rendimiento que la solución actual.

Se utilizó un ambiente en Anaconda¹⁶, donde se instalaron las librerías utilizadas en este proyecto. Talos¹⁷ fue la principal librería utilizada. Esta librería implementa búsquedas de hiperparámetros. Se eligió esta librería, ya que permite utilizar Keras¹⁸ directamente para la programación de redes neuronales.

En Talos se debe implementar una función que recibe como parámetros el conjunto de entrenamiento y validación, y un diccionario con los parámetros de la red. Dentro de la función se debe construir, compilar y entrenar la red utilizando dichos parámetros.

Una vez implementada la función se utiliza la función *Scan()* para realizar la optimización de hiperparámetros. La función *Scan()* tiene como parámetros un conjunto (más adelante se divide en 80 % entrenamiento y 20 % validación), un diccionario de parámetros, donde cada llave es un hiperparámetro y el valor es una lista con los valores que cada hiperparámetro puede tomar, y la función que implementa el entrenamiento de una de estas redes. Por defecto, la función *Scan()* utiliza *grid search*¹⁹ para la optimización de hiperparámetros, pero también puede ser utilizada para realizar *random search*²⁰.

Por último, la función *Scan()* devuelve un objeto que puede ser entregado a la función *Deploy()* para almacenar el mejor modelo encontrado. Este modelo es seleccionado utilizando alguna de las métricas de entrenamiento. Además este método guarda en un archivo CSV los parámetros y métricas de cada uno de los modelos entrenados.

El Algoritmo 1 muestra la red neuronal implementada, la cual tiene una capa de entrada de 17 variables con una cantidad de neuronas determinada por el parámetro *first_neuron* y con función de activación determinado por el parámetro *activation*. La siguiente capa es de *dropout*²¹ y su porcentaje es determinando por el parámetro *dropout*.

La función *HIDDEN_LAYERS* (Algoritmo 2) es implementada por Talos y permite añadir capas escondidas a la red. La cantidad de capas es determinanda por el parámetro *hidden_layers*. La cantidad de neuronas en cada capa es determinda por el parámetro *shapes* y *first_neuron*, y la función de activación por el parámetro *activation*. Además entremedio de cada capa densa existe una capa de *dropout*.

¹⁶<https://www.anaconda.com/>

¹⁷<https://github.com/autonomio/talos>

¹⁸<https://keras.io/>

¹⁹Método que realiza una búsqueda exhaustiva de una solución óptima

²⁰Método que realiza una búsqueda aleatoria de una solución óptima

²¹Técnica de regularización en la que neuronas seleccionadas aleatoriamente son ignoradas durante el entrenamiento.

Algorithm 1: Arquitectura del modelo DNN

input : x_{train} , y_{train} , x_{val} , y_{val} , $first_neuron$, $activation$, $dropout$, $shapes$,
 $last_activation$, $optimizer$, $learning_rate$, $loss$, $batch_size$, $epochs$

```
metrics = [ f1score, precision, recall ];  
model = Sequential();  
model.add(DENSE( $first\_neuron$ ,  $input\_dim=17$ ,  $activation$ ));  
model.add(DROPOUT( $dropout$ ));  
HIDDEN_LAYERS( $model$ ,  $shapes$ ,  $first\_neuron$ ,  $dropout$ ,  $hidden\_layers$ ,  $activation$ );  
model.add(DENSE(1,  $last\_activation$ ));  
model.compile( $optimizer$ ,  $learning\_rate$ ,  $loss$ ,  $metrics$ );  
train_info = model.fit( $x_{train}$ ,  $y_{train}$ ,  $batch\_size$ ,  $epochs$ ,  $x_{val}$ ,  $y_{val}$ );
```

output: $train_info$, $model$

Algorithm 2: HIDDEN LAYERS

input : $model$, $shapes$, $first_neuron$, $dropout$, $hidden_layers$, $activation$

```
 $i = 0$ ;  
difference = ROUND(( $first\_neuron - last\_neuron$ ) /  $hidden\_layers$ );  
while  $i < hidden\_layers$  do  
  if  $shape == brick$  then  
     $nth\_layer = first\_neuron$ ;  
  else  
    if  $shape == funnel$  then  
       $nth\_layer = [nth-1]_{layer} - difference$ ;  
    end  
  end  
   $model.add(DENSE(nth\_layer, activation))$ ;  
   $model.add(DROPOUT(dropout))$ ;  
   $i = i + 1$ ;  
end
```

La capa de salida consiste en una capa de una neurona con función de activación determinada por el parámetro *last_activation*. Finalmente se compila la red, definiendo el optimizador, *learning rate*, función de pérdida y métricas que se utilizarán en el entrenamiento, y se entrena la red. El algoritmo retorna la información del entrenamiento y el modelo.

La búsqueda fue implementada utilizando un algoritmo de *grid search*, es decir, se probaron todas las combinaciones posibles de los hiperparámetros en los espacios de búsqueda descritos a continuación.

3.4.1. Primera búsqueda de hiperparámetros

El espacio de búsqueda utilizado por cada hiperparámetro se seleccionó en base a criterios empíricos determinados por los valores que suelen tomar para estos hiperparámetros en este tipo de modelos. La Tabla 2 muestra el espacio de búsqueda utilizado. A continuación, se describen los hiperparámetros y valores seleccionados.

- **Loss:** Corresponde a la función de pérdida. Se empleó Binary cross entropy, porque el problema que se intenta resolver es un problema de clasificación binaria.
- **Optimizer:** Corresponde a los optimizadores utilizados. Se seleccionaron 3 optimizadores distintos: Adam, RMSprop y Nadam.
- **Learning rate:** Se escogió 5 valores entre 0.001 y 10 para la optimización. Estos valores son normalizados usando una función de Talos, la cual varía dependiendo del optimizador utilizado. Para los 3 optimizadores seleccionados, los valores son divididos por 1000 en la normalización.
- **Shape:** Corresponde a la forma de la red. Se seleccionaron dos tipos de formas: Ladrillo y Embudo. La forma de ladrillo corresponde a una red neuronal con la misma cantidad de neuronas en cada capa y la forma de embudo corresponde a una red neuronal, cuya n -ésima capa tiene una menor cantidad de neuronas que la capa anterior ($n-1$).
- **Hidden layers:** Corresponde a la cantidad de capas ocultas. Se escogió 4 valores entre 0 y 16.
- **Activation function:** Corresponde a la función de activación. Se seleccionaron 2 funciones de activación: Relu y Elu.
- **Epochs:** Se utilizarán 100 *epochs* para todos los entrenamientos y se utilizó *early stopping* para parar el entrenamiento en caso de divergencia de la pérdida.
- **Batch size:** Se emplearon valores cercanos al 1 % de los datos. Se seleccionarán 3 valores: 2048, 4096 y 8192.
- **Dropout:** Se utilizaron 4 valores de *dropout*: 0, 0.1, 0.2 y 0.4.

- **First layer:** Corresponde a la cantidad de neuronas en la primera capa. Se seleccionaron 3 valores: 32, 64 y 128.

Parámetros	Valores				
Loss	Binary Cross Entropy				
Optimizer	Adam	RMSprop	Nadam		
Learning rate	0.001	2,0008	4,0006	6,0004	8,0002
Shape	Brick	Funnel			
Hidden layers	0	4	8	16	
Activation function	Relu	Elu			
Epochs	100				
Batch size	2048	4096	8192		
Dropout	0	0.1	0.2	0.4	
First layer	32	64	128		
Metrics	F1 Score	Precision	Recall		

Tabla 2: *Espacio de búsqueda utilizado en la primera optimización de hiperparámetros.*

Una vez aplicado el algoritmo de optimización de hiperparámetros y probadas todas las combinaciones posibles, se obtuvieron 8640 modelos. La Tabla 3 muestra los hiperparámetros del mejor modelo obtenidos, el cual fue escogido basado en la métrica *validation f1 score* (Ver Tabla 4).

Parámetros	Valores
Loss	Binary Cross Entropy
Optimizer	Adam
Learning rate	2,0008
Shape	Funnel
Hidden layers	16
Activation function	Elu
Epochs	100
Batch size	2048
Dropout	0
First layer	64

Tabla 3: *Hiperparámetros del mejor modelo encontrado en la primera optimización realizada.*

Métricas	Valores
F1 score	0.991283
Precision	0.992127
Recall	0.990456
Validation f1 score	0.990906
Validation Precision	0.991345
Validation Recall	0.990476

Tabla 4: Métricas del mejor modelo encontrado en la primera optimización realizada.

3.4.2. Segunda búsqueda de hiperparámetros

Una vez terminada la primera optimización de hiperparámetros, se modificó el espacio de búsqueda de algunos de estos hiperparámetros para buscar mejores resultados. La Tabla 5 muestra los nuevos hiperparámetros utilizados en la búsqueda.

Los hiperparámetros modificados fueron los siguientes:

- **Learning Rate:** Se utilizarón 10 valores entre 1 y 100.
- **Hidden layers:** Se modificó el rango del espacio de búsqueda para incluir mayor cantidad de capas.
- **Batch size:** Se cambió el rango del espacio de búsqueda para incluir valores más pequeños.
- **First layer :** Se modificó el rango del espacio de búsqueda para incluir mayor cantidad de neuronas en la primera capa.

Parámetros	Valores									
Loss	Binary Cross Entropy									
Optimizer	Adam	RMSprop	Nadam							
Learning rate	1	10	20	30	40	50	60	70	80	90
Shape	Brick	Funnel								
Hidden layers	4	8	16	32						
Activation function	Relu	Elu								
Epochs	100									
Batch size	512	1024	2048							
Dropout	0									
First layer	64	128	256							

Tabla 5: Espacio de búsqueda utilizado en la segunda optimización de hiperparámetros.

Una vez aplicado el algoritmo de grid search, se obtuvieron 4320 modelos. La Tabla 6 muestra los hiperparámetros del mejor modelo obtenido, el cual fue escogido utilizando la métrica *validation f1 score* (Ver Tabla 7).

Parámetros	Valores
Loss	Binary Cross Entropy
Optimizer	Adam
Learning rate	10
Shape	Brick
Hidden layers	4
Activation function	Relu
Epochs	100
Batch size	512
Dropout	0
First layer	64

Tabla 6: *Hiperparámetros del mejor modelo encontrado en la segunda optimización realizada.*

Métricas	Valores
F1 Score	0.991446
Precision	0.992277
Recall	0.990660
Validation f1 Score	0.991557
Validation precision	0.991687
Validation recall	0.991460

Tabla 7: *Métricas del mejor modelo encontrado en la segunda optimización realizada.*

3.4.3. Tercera búsqueda de hiperparámetros

Se realizó una tercera búsqueda de hiperparámetros para investigar el rendimiento de otras funciones de pérdida. Como se muestra en la Tabla 8, las funciones de pérdida utilizadas fueron Hinge y Squared hinge, y se entrenaron un total de 3888 modelos.

Parámetros	Valores			
Epochs	100			
Activation	Relu			
Batch size	512	1024	2048	
Dropout	0	0.2	0.4	
First Layer	32	64	128	
Hidden Layer	4	8	16	
Last Activation	tanh			
Loss	Hinge	Squared hinge		
Learning Rate	2.5	5	7.5	10
Optimizer	Adam	RMSprop	Nadam	
Shapes	Funnel	Brick		

Tabla 8: *Espacio de búsqueda utilizado en la tercera optimización de hiperparámetros.*

El espacio de búsqueda fue dividido en 8 sub espacios para entrenar los modelos en 8 GPUs distintas. La Tabla 9 muestra la división realizada al espacio de búsqueda, donde se fijó el valor de los hiperparametros *learning rate* y la función de pérdida.

Red Neuronal	Learning Rate	Loss
FCNN00	2.5	Hinge
FCNN01	2.5	Squared hinge
FCNN02	5	Hinge
FCNN03	5	Squared hinge
FCNN04	7.5	Hinge
FCNN05	7.5	Squared hinge
FCNN06	10	Hinge
FCNN07	10	Squared hinge

Tabla 9: *División del espacio de búsqueda utilizado en la tercera optimización de hiperparámetros.*

Red Neuronal	F1 Score	Precision	Recall	Val F1 Score	Val Precision	Val Recall
FCNN00	0.991255	0.991684	0.990844	0.990986	0.992292	0.989694
FCNN01	0.993443	0.994227	0.992692	0.992339	0.992518	0.992193
FCNN02	0.990225	0.991505	0.988975	0.990968	0.990734	0.991211
FCNN03	0.993093	0.993691	0.992528	0.992142	0.992586	0.991727
FCNN04	0.989918	0.989255	0.99061	0.990421	0.990747	0.990104
FCNN05	0.992846	0.993588	0.992135	0.992272	0.993317	0.991259
FCNN06	0.990181	0.98922	0.991159	0.99014	0.988265	0.992032
FCNN07	0.993002	0.993553	0.992481	0.992684	0.991914	0.993482

Tabla 10: *Métricas del mejor modelo encontrado en cada sub-espacio para la tercera optimización de hiperparámetros.*

Una vez realizada la tercera optimización de hiperparámetros, se seleccionó por cada sub-espacio el modelo con mayor *validation f1 score*. La Tabla 10 muestra las métricas de rendimiento de dichos modelos.

Se seleccionó el modelo FCNN07 (a partir de ahora referido como FCNN3) como el mejor modelo del espacio de búsqueda, puesto que es el que tiene mayor *validation f1 score*. La Tabla 11 muestra los hiperparámetros de esta red.

Parámetros	Valores
activation	Relu
batch_size	2048
dropout	0
epochs	100
first layer	128
hidden layers	8
last_activation	tanh
loss_fun	squared_hinge
lr	0.0025
optimizer	rmsprop
shapes	brick

Tabla 11: *Hiperparámetros del mejor modelo encontrado en la tercera optimización realizada.*

CAPÍTULO 4

VALIDACIÓN DE LA SOLUCIÓN

En este capítulo se validan los resultados del algoritmo propuesto, y se comparan los mejores modelos obtenidos con el modelo BDT del estado del arte. Como se mencionó en la Sección 3.1, el conjunto de datos incluye la predicción realizada por el modelo BDT para cada dato. Estas predicciones fueron utilizadas para el cálculo de las métricas y gráficos descritos a continuación. Se utilizan la matriz de confusión, gráficos de la distribución de puntuaciones y curva ROC, para realizar las comparaciones.

4.1. Métricas

Red Neuronal	F1 Score	Precision	Recall
FCNN1	0.983767	0.996153	0.971684
FCNN2	0.989997	0.989049	0.990948
FCNN3	0.958805	0.997826	0.922720
BDT	0.990898	0.993978	0.993978

Tabla 12: Métricas de los mejores modelos encontrados y el modelo BDT del estado del arte. Las métricas fueron calculadas utilizando el conjunto de prueba.

En la Tabla 12 se observan las métricas usuales de aprendizaje automático para medir el desempeño de un clasificador, incluyendo *F1 score*, *Precision* y *Recall* de los mejores modelos encontrados. Se puede observar que el modelo FCNN2 cuenta con un *F1 score* de 0.989 y *recall* de 0.990, siendo el modelo con mejor *F1 score* y *recall*, sin embargo, no logró superar el rendimiento del modelo BDT con *F1 score* de 0.990 y *recall* de 0.993. El modelo FCNN3 tiene una precisión de 0.997, siendo el modelo con mayor precisión.

4.2. Matriz de confusión

Se graficaron las matrices de confusión (Figura 23) de los distintos modelos para una mejor comprensión de la capacidad de su separación. Las matrices de confusión fueron obtenidas utilizando el conjunto de prueba. Se utilizó el umbral 0.5 para el cálculo de estas matrices, es decir, los eventos cuya predicción fue mayor igual a 0.5 fueron clasificados como señal, y los eventos cuya predicción fue menor a 0.5 fueron clasificados como background. Como se puede observar todos los modelos tienen una buena capacidad de separación entre background y señal.

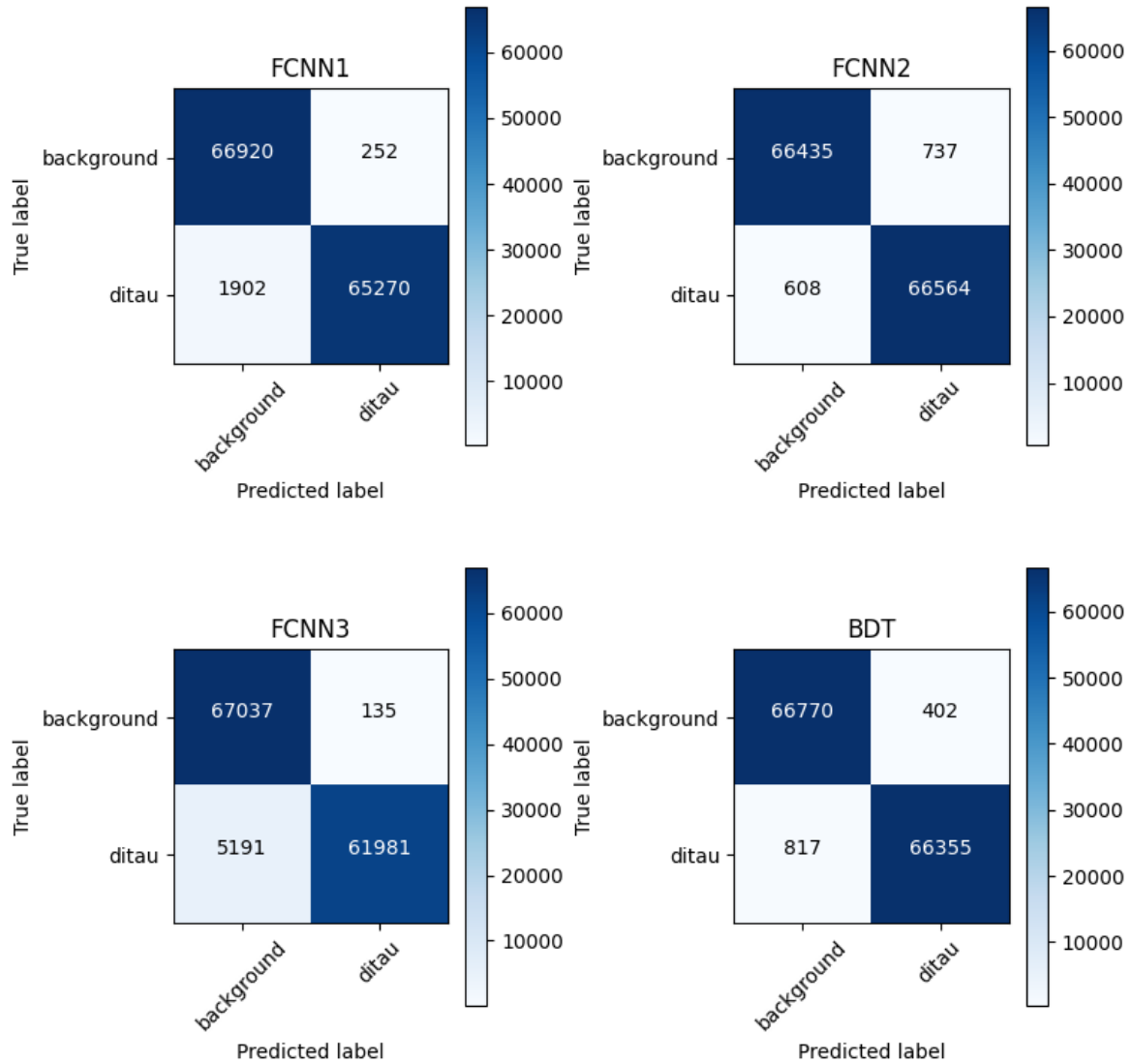


Figura 23: Matrices de confusi3n de los mejores modelos encontrados y el modelo BDT del estado del arte. Se utiliz3 un umbral de 0.5 y el conjunto de prueba para el c3lculo de todas las matrices.

El modelo FCNN1 tiene 252 falsos positivos y 1902 falsos negativos, lo cual indica que el modelo logra distinguir el background un poco mejor que la señal. El modelo FCNN2 tiene 737 falsos positivos y 608 falsos negativos, siendo el modelo que mejor predice la señal y que peor predice el background. El modelo FCNN3 tiene 135 falsos positivos y 5191 falsos negativos, siendo el modelo que peor predice la señal y que mejor predice el background.

Básandose en las matrices de confusión de la Figura 23 y sus respectivas métricas descritas en la Tabla 12 **se puede observar que de los 3 modelos propuestos, el modelo FCNN2 es el que obtuvo el resultado más cercano a la solución actual (BDT), alcanzando en un 99.9 % al F1, 99.5 % a la precisión, y 99.7 % al recall.**

4.3. Curva ROC

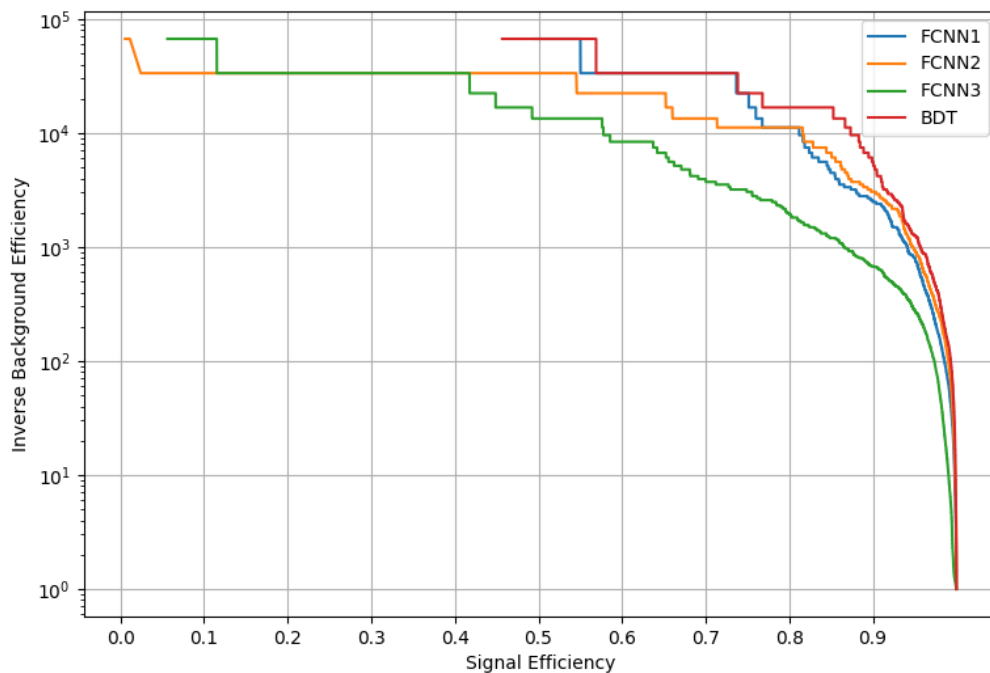


Figura 24: Curva ROC de los mejores modelos encontrados y el modelo BDT del estado del arte. La curva fue calculada utilizando el conjunto de prueba.

La Figura 24 muestra la curva ROC de los mejores modelos encontrados junto con el modelo BDT del estado del arte. El gráfico compara la eficiencia de señal (*signal efficiency*), es decir, la fracción de datos clasificados correctamente del conjunto de datos con etiqueta de señal; y el inverso de la eficiencia de background, es decir, el inverso de la fracción de datos clasificados incorrectamente del conjunto de datos con etiqueta de background. Este quiere decir

que con una eficiencia de señal de 0.8 para el modelo FCNN1, uno de cada mil objetos de background es mal clasificado como señal. Esto nos lleva a concluir que entre mayor es la eficiencia de señal y mayor es el inverso de la eficiencia de background, mejor es la capacidad de separación de modelo.

De los modelos propuestos para una eficiencia de señal mayor a 0.8 el modelo FCNN2 es el que tiene una mejor capacidad de separación, y para una eficiencia de señal menor a 0.8 el modelo FCNN1 es el que tiene una mejor capacidad de separación. Sin embargo, ninguno de los modelos propuestos logra superar la capacidad de separación de la solución actual.

Red Neuronal	Signal Efficiency	Background Rejection	Working point
FCNN1	0.95	$7.996 \cdot 10^2$	0.8103
	0.9	$2.487 \cdot 10^3$	0.9610
	0.55	$6.717 \cdot 10^4$	0.9995
	0.2	N/A	0.9997
FCNN2	0.95	$9.460 \cdot 10^2$	0.9873
	0.9	$3.053 \cdot 10^3$	0.9976
	0.55	$2.239 \cdot 10^4$	0.9999
	0.2	$3.358 \cdot 10^4$	0.9999
FCNN3	0.95	$2.787 \cdot 10^2$	-0.5275
	0.9	$6.785 \cdot 10^2$	0.3217
	0.55	$1.343 \cdot 10^4$	0.9924
	0.2	$3.358 \cdot 10^4$	0.9994
BDT	0.95	$1.267 \cdot 10^3$	0.5972
	0.9	$5.597 \cdot 10^3$	0.6426
	0.55	$6.717 \cdot 10^4$	0.7360
	0.2	N/A	0.7799

Tabla 13: *Working points, eficiencia de señal y eficiencia de background de los mejores modelos encontrados. Las métricas fueron calculadas a partir de la curva ROC, utilizando el conjunto de prueba.*

A partir de la curva ROC se calcularon distintos *working points*, los cuales son utilizados en física de partículas para seleccionar la pureza de los datos dependiendo del análisis a realizar. Como se puede observar en la Tabla 13 se seleccionaron 4 eficiencias de señal distintas para el cálculo de los distintos *working points*: very loose (0.95), loose (0.9), medium (0.55) y tight (0.2). Además, cada *working point* tiene un background rejection asociado.

Para una eficiencia de señal de 0.95, el modelo entrenado con mejor rendimiento fue el modelo FCNN2, puesto que presenta un background rejection de $9.460 \cdot 10^2$, siendo el mayor de los tres modelos, es decir, es el modelo que confunde un dato de background por cada $9.460 \cdot 10^2$ datos de señal para esta eficiencia de señal. **El modelo FCNN2 logró alcanzar en un 74.6 % al background rejection del modelo BDT para el *working point* very loose.** El modelo FCNN1 alcanzó en un 63.1 % al background rejection del modelo BDT y el modelo FCNN3 alcanzó en un 21.9 % al background rejection del modelo BDT.

Para una eficiencia de señal de 0.9, el modelo entrenado con mejor rendimiento fue el modelo FCNN2, puesto que presenta un background rejection de $3.053 \cdot 10^3$. **El modelo FCNN2 alcanzó en un 54.5 % al background rejection del modelo BDT para el *working point loose*.** El modelo FCNN1 alcanzó en un 44.4 % al background rejection del modelo BDT y el modelo FCNN3 logró alcanzar en un 12.1 % al background rejection del modelo BDT.

Para una eficiencia de señal de 0.55, el modelo entrenado con mejor rendimiento fue el modelo FCNN1, puesto que presenta un background rejection de $6.717 \cdot 10^4$. **El modelo FCNN1 logró alcanzar en un 100 % al modelo BDT para el *working point medium*.** El modelo FCNN2 alcanzó en un 33.3 % al background rejection del modelo BDT y el modelo FCNN3 alcanzó en un 20 % al modelo BDT.

Para una eficiencia de señal 0.2, el modelo entrenado con mejor rendimiento fue el modelo FCNN1, puesto que presenta una eficiencia de señal de 0, es decir, el modelo no confunde background con señal para esta eficiencia de señal. **El modelo FCNN1 logró alcanzar un 100 % al modelo BDT para el *working point tight*.**

Se puede observar que para los 3 modelos propuestos presentan *working points* medium y tight muy cercanos, lo cual impide separar los datos, utilizando esta eficiencia de señal.

4.4. Distribución de Puntuaciones

A continuación se graficaron las distribuciones de puntuaciones de los modelos obtenidos. Se utilizó el conjunto de entrenamiento para obtener las predicciones de cada modelo y graficar los histogramas Di-Tau Signal Train y QCD background Train, y se utilizó el conjunto de prueba para obtener las predicciones de cada uno de los modelos y graficar los histogramas Di-Tau Signal Test y QCD background Test. Además se incluyó los *working points* de la Tabla 13 para su visualización.

En la Figura 25 se puede observar que la distribución de puntuaciones del modelo FCNN1 tiende a concentrarse en los extremos del rango, permitiendo alcanzar una buena separación entre background y señal. Además, las curvas de *Test* y *Train* tienden a parecerse, lo cual indica que el modelo generaliza correctamente. Por último, no se logró distinguir entre los *working points medium* y *tight*, debido a que sus valores son muy cercanos. Esta cercanía entre los valores se debe a la gran cantidad de datos que se encuentran en esta zona, ya que impide una transición suave de la eficiencia de señal.

En la Figura 26 se puede observar la distribución de puntuaciones del modelo FCNN2. En este histograma se observa que las predicciones tienden a concentrarse en los extremos del rango, permitiendo alcanzar una buena separación entre background y señal. Además, al igual que el modelo FCNN1, las curvas de *Train* y *Test* tienden a parecerse, lo cual indica que el modelo generaliza correctamente. Por último, no se logró distinguir entre los *working points loose*, *medium* y *tight*, y al igual que en el modelo FCNN1, esto se debe a la concentración

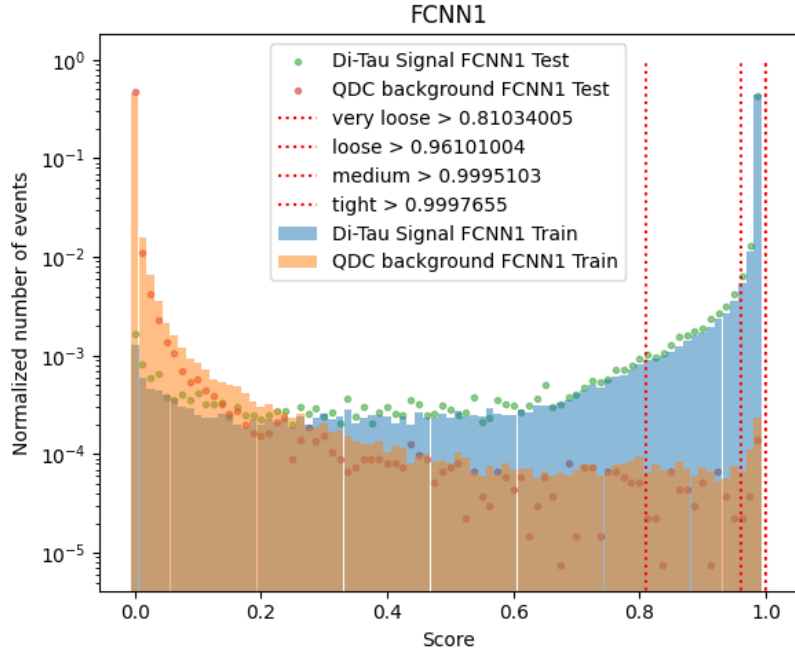


Figura 25: Distribución de puntuaciones del modelo FCNN1 para los conjuntos de entrenamiento y prueba, y working points calculados para el modelo.

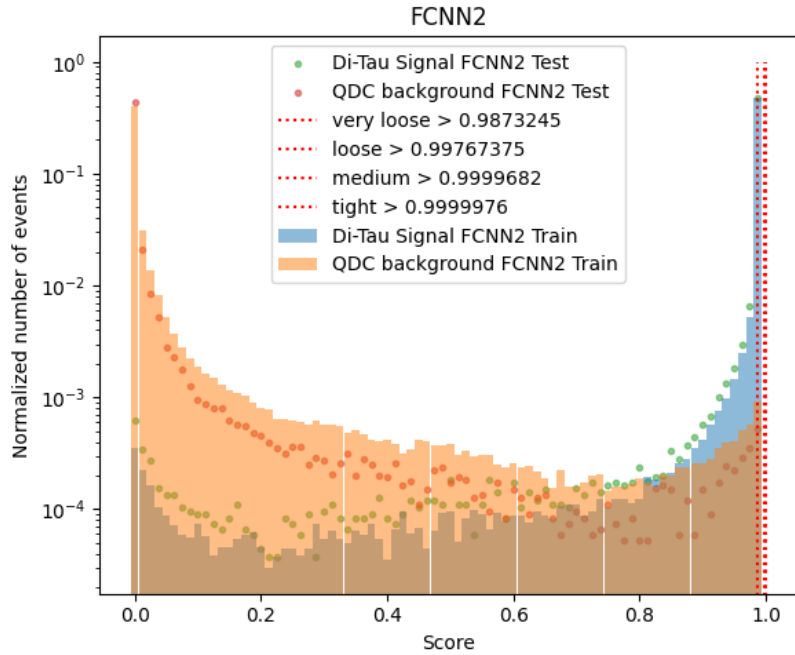


Figura 26: Distribución de puntuaciones del modelo FCNN2 para los conjuntos de entrenamiento y prueba, y working points calculados para el modelo.

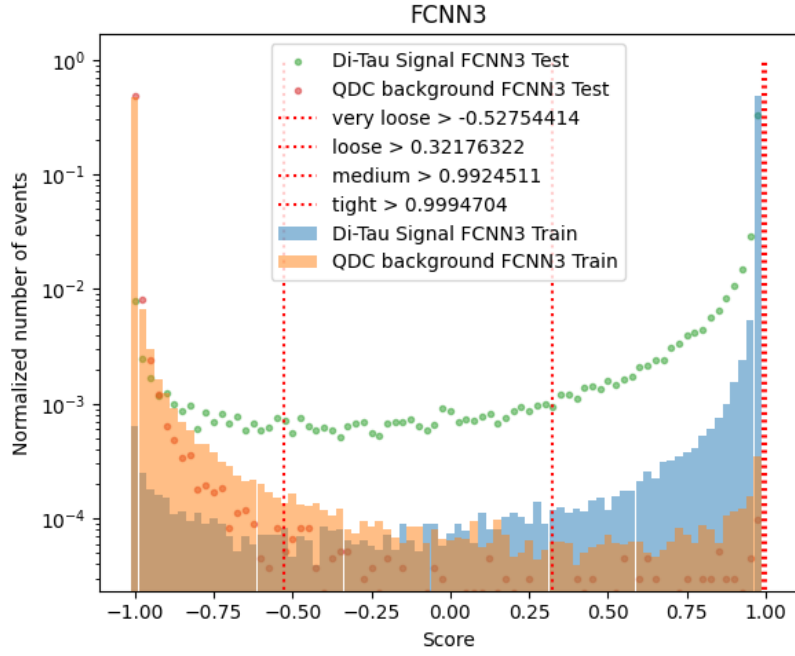


Figura 27: Distribución de puntuaciones del modelo FCNN3 para los conjuntos de entrenamiento y prueba, y working points calculados para el modelo.

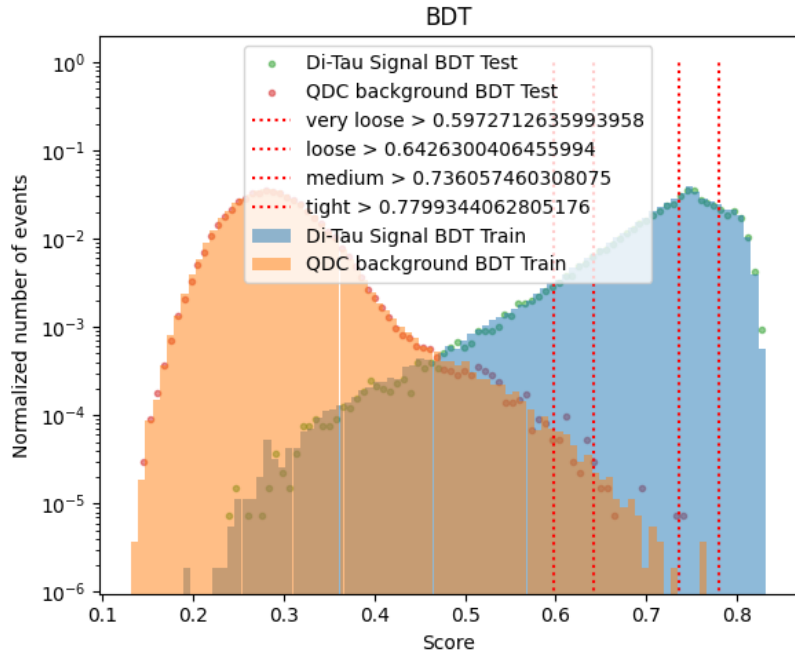


Figura 28: Distribución de puntuaciones del modelo BDT para los conjuntos de entrenamiento y prueba, y working points calculados para el modelo.

de predicciones en esta zona.

En la Figura 27 se puede observar que el modelo FCNN3 tiende a concentrar los valores en los extremos del rango. Además, las curvas de *Test* y *Train* para la señal no se parecen, lo cual indica que la generalización del modelo puede ser mejorada. Por último, los *working points* *very loose* y *loose* tienden a estar más separados que en los otros modelos. Esto se debe a que las predicciones de los datos de señal en el conjunto de prueba tienden a estar más repartidas a lo largo del rango que en los demás modelos. Sin embargo, los *working points* *medium* y *tight* no se logran distinguir, debido a la concentración de predicciones en esta zona.

Finalmente, en la figura 28 se puede observar la distribución de puntuaciones del modelo BDT del estado del arte. En este histograma, se logra distinguir los *working points* debido a que la concentración de las predicciones es menor que en los otros modelos.

CAPÍTULO 5

CONCLUSIONES

En este trabajo se abordó el problema de identificación de pares tau tipo boosted en el experimento ATLAS, utilizando redes neuronales profundas. En específico, se investigó el uso de redes neuronales profundas para la solución del problema cumpliendo con el objetivo principal de desarrollar un algoritmo para identificar el par tau boosted, al igual que con los objetivos específicos de generar un conjunto de datos adecuado, diseñar e implementar un algoritmo basado en aprendizaje automático, y validar el algoritmo. El trabajo desarrollado puede ser encontrado en el repositorio [Zavala, 2021], junto los pesos de las mejores redes encontradas para la reproducibilidad del trabajo presentado.

Inicialmente, fue necesario generar los conjuntos de datos que contenían los eventos a analizar. Se emplearon conjuntos de datos provenientes de simulaciones de Monte Carlo, los cuales se encontraban almacenados en la *Grid* de los experimentos del LHC. Se utilizó THOR para transformar los conjuntos de los archivos xAOD a archivos ROOT.

Cada archivo pertenecía a una de dos clases, background o señal. Los datos de señal provienen del decaimiento $G \rightarrow hh \rightarrow 4\tau$, que corresponden a pares *tau* verdaderos, y los datos de background provienen de *jet* de hadrones QCD y son pares *tau* falsos o mal identificados. Se analizaron los datos para comprobar la validez de estos y se eliminaron aquellos datos con menos de 2 *sub-jets* y 0 *lead tracks*. Luego se filtraron los datos para dejar aquellos con decaimiento hadrónico y asociados geométricamente con alguno de los *di-taus* producidos en las simulaciones.

Previo a la etapa de entrenamiento, se separó cada conjunto en 80 % para entrenamiento y 20 % para prueba, y debido al desbalance entre ambas clases, se realizó un *undersampling* de los datos. Se utilizaron todos los datos de señal disponibles y se seleccionó de manera aleatoria una cantidad equivalente de background.

Para el desarrollo del modelo de clasificación, se utilizó una red neuronal profunda completamente conectada y se utilizó un algoritmo de *grid search* para la optimización de los hiperparámetros del modelo. Se realizaron tres optimizaciones de hiperparámetros.

En la primera optimización se utilizó el espacio de búsqueda descrito en la Tabla 2, obteniendo el modelo FCNN1 con *F1 score* de 0.983. Se realizó una segunda optimización descrita en la Tabla 5 para buscar una mejor solución, de la cual se obtuvo el modelo FCNN2 con *F1 score* de 0.989. Finalmente se realizó una tercera optimización descrita en 8 para probar distintas funciones de pérdida. De esta última optimización se obtuvo el modelo FCNN3 con *F1 score* de 0.958.

Después de analizar los gráficos de la Sección 4, se concluyó que el modelo FCNN2 fue el mejor modelo obtenido para los *working points very loose* y *loose*, con un *background rejection*

de $9.460 \cdot 10^2$ para *very loose* y $3.053 \cdot 10^3$ para *loose*, alcanzando un 74.6 % y un 54.5 % a la solución actual respectivamente. El modelo FCNN1 fue el mejor modelo obtenido para los *working points medium* y *tight*, con un *background rejection* de $6.717 \cdot 10^4$ para *medium* y una eficiencia de background de 0 para el *tight*, alcanzando un 100 % a la solución actual en ambos *working points*.

5.1. Trabajo futuro

Los datos utilizados durante el entrenamiento pertenecían en su mayoría al rango de energía entre 100 GeV y 1300 GeV, dificultando el entrenamiento en zonas de mayor energía. Es por esto que, mejoras todavía pueden ser hechas en la identificación de taus, sobre todo en estas zonas de mayor energía.

Durante este trabajo, solo fueron consideradas las variables utilizadas por el modelo BDT del estado del arte, por ende, no se consideraron otro tipo de variables que podrían tener un mayor impacto en los clasificadores. Podrían ser incluidos datos crudos de los detectores y variables relacionadas con los *jets* que conforman los objetos.

Además de esto, se propone realizar un *oversampling* de los datos, en vez de un *undersampling*, para tener una mayor cantidad de datos sobre los que entrenar.

Finalmente, se propone la utilización de redes neuronales recurrentes para la identificación de objetos *di-tau*, ya que es el actual estado del arte en la identificación de leptones tau.

REFERENCIAS BIBLIOGRÁFICAS

- [ATL, 2019] (2019). Identification of hadronic tau lepton decays using neural networks in the ATLAS experiment. Technical Report ATL-PHYS-PUB-2019-033, CERN, Geneva. <<https://cds.cern.ch/record/2688062>>[Consulta: 2 julio 2020].
- [Aaboud *et al.*, 2019] Aaboud, M., Aad, G., Abbott, B., y *[et.al]* (2019). Cross-section measurements of the higgs boson decaying into a pair of τ -leptons in proton-proton collisions at $\sqrt{s} = 13$ TeV with the atlas detector. *Phys. Rev. D*, 99:072001. <<https://journals.aps.org/prd/abstract/10.1103/PhysRevD.99.072001>>[Consulta: 2 julio 2020].
- [Aad *et al.*, 2012] Aad, G., Abajyan, T., Abbott, B., y *[et.al]* (2012). Observation of a new particle in the search for the standard model higgs boson with the atlas detector at the lhc. *Physics Letters B*, 716(1):1 – 29. <<https://www.sciencedirect.com/science/article/pii/S037026931200857X>>[Consulta: 31 julio 2020].
- [Aad *et al.*, 2008] Aad, G., Abat, E. A., y *[et.al]* (2008). The ATLAS Experiment at the CERN Large Hadron Collider. *JINST*, 3:S08003.
- [Aad *et al.*, 2020] Aad, G., Abbott, B., Abbott, D. C., y *[et.al]* (2020). Search for heavy Higgs bosons decaying into two tau leptons with the ATLAS detector using pp collisions at $\sqrt{s} = 13$ TeV. Technical Report arXiv:2002.12223, CERN, Geneva. 32 pages in total, author list starting page 16, 2 figures, 2 tables, submitted to Phys. Rev. Lett.. All figures including auxiliary figures are available at <https://atlas.web.cern.ch/Atlas/GROUPS/PHYSICS/PAPERS/HDBS-2018-46>. <<https://cds.cern.ch/record/2711592>>[Consulta: 2 julio 2020].
- [Aad *et al.*, 2015] Aad, G., Abbott, B., Abdallah, J., y *[et.al]* (2015). Evidence for the higgs-boson yukawa coupling to tau leptons with the atlas detector. *J. High Energ. Phys.* <[https://link.springer.com/article/10.1007/JHEP04\(2015\)117](https://link.springer.com/article/10.1007/JHEP04(2015)117)>[Consulta: 2 julio 2020].
- [Albertsson *et al.*, 2018] Albertsson, K., Altoe, P., Anderson, D., y *[et.al]* (2018). Machine learning in high energy physics community white paper. *Journal of Physics: Conference Series*, 1085:022008. <<https://iopscience.iop.org/article/10.1088/1742-6596/1085/2/022008>> [Consulta: 20 junio 2020].
- [Alioli *et al.*, 2010] Alioli, S., Nason, P., Oleari, C., y Re, E. (2010). A general framework for implementing nlo calculations in shower monte carlo programs: the powheg box. *Journal of High Energy Physics*, 2010(6). [En línea] <<https://link.springer.com/article/10.1007%2FJHEP06%282010%29043>> [Consulta: 15 diciembre 2021].
- [Alwall *et al.*, 2011] Alwall, J., Herquet, M., Maltoni, F., Mattelaer, O., y Stelzer, T. (2011). Madgraph 5: going beyond. *Journal of High Energy Physics*, 2011(6). [En línea] <<https://link.springer.com/article/10.1007%2FJHEP06%282011%29128>> [Consulta: 25 noviembre 2021].

- [ATLAS, a] ATLAS. The atlas detector. [En línea] <<https://atlas.cern/discover/detector>>[Consulta: 29 julio 2020].
- [ATLAS, b] ATLAS. Calorimeter. [En línea] <<https://atlas.cern/discover/detector/calorimeter>>[Consulta: 29 julio 2020].
- [ATLAS, c] ATLAS. The inner detector. [En línea] <<https://atlas.cern/discover/detector/inner-detector>>[Consulta: 29 julio 2020].
- [ATLAS, d] ATLAS. Muon spectrometer. [En línea] <<https://atlas.cern/discover/detector/muon-spectrometer>>[Consulta: 29 julio 2020].
- [ATLAS, 2018] ATLAS (2018). Could a new type of quark fix the “unnaturalness” of the standard model? [En línea] <<https://atlas.cern/updates/physics-briefing/could-new-type-quark-fix-unnaturalness-standard-model>>[Consulta: 28 julio 2020].
- [Branco *et al.*, 2012] Branco, G., Ferreira, P., Lavoura, L., Rebelo, M., Sher, M., y Silva, J. P. (2012). Theory and phenomenology of two-higgs-doublet models. *Physics Reports*, 516(1-2):1–102. [En línea] <<https://www.sciencedirect.com/science/article/abs/pii/S0370157312000695>>[Consulta: 03 diciembre 2021].
- [Buckley *et al.*, 2015] Buckley, A., Eifert, T., Elsing, M., Gillberg, D., Koenek, K., Krasznahorkay, A., Moyse, E., Nowak, M., Snyder, S., y van Gemmeren, P. (2015). Implementation of the ATLAS run 2 event data model. *Journal of Physics: Conference Series*, 664(7):072045.
- [Cacciari *et al.*, 2008a] Cacciari, M., Salam, G. P., y Soyez, G. (2008a). The anti-kt jet clustering algorithm. *Journal of High Energy Physics*, 2008(04):063–063. [En línea] <<https://iopscience.iop.org/article/10.1088/1126-6708/2008/04/063>>[Consulta: 15 diciembre 2021].
- [Cacciari *et al.*, 2008b] Cacciari, M., Salam, G. P., y Soyez, G. (2008b). The anti-ktjet clustering algorithm. *Journal of High Energy Physics*, 2008(04):063–063. [En línea] <<https://iopscience.iop.org/article/10.1088/1126-6708/2008/04/063>>[Consulta: 13 diciembre 2021].
- [CERN,] CERN. The standard model. [En línea] <<https://home.cern/science/physics/standard-model>>[Consulta: 28 julio 2020].
- [Chatrchyan *et al.*, 2012] Chatrchyan, S., Khachatryan, V., Sirunyan, A., y *[et.al]* (2012). Observation of a new boson at a mass of 125 gev with the cms experiment at the lhc. *Physics Letters B*, 716(1):30–61. <<https://www.sciencedirect.com/science/article/pii/S0370269312008581>>[Consulta: 31 julio 2020].
- [di Wu, 1997] di Wu, D. (1997). A brief introduction to the strong cp problem. *Zeitschrift für Naturforschung A*, 52(1-2):179–182. [En línea] <<https://www.degruyter.com/document/doi/10.1515/zna-1997-1-245/html>>[Consulta: 03 diciembre 2021].

- [Drucker y Cortes, 1995] Drucker, H. y Cortes, C. (1995). Boosting decision trees. En *Proceedings of the 8th International Conference on Neural Information Processing Systems*, NIPS'95, p. 479–485, Cambridge, MA, USA. MIT Press.
- [Evans y Bryant, 2008] Evans, L. y Bryant, P. (2008). Lhc machine. *Journal of Instrumentation*, 3(08):S08001.
- [Ghahramani, 2004] Ghahramani, Z. (2004). *Unsupervised Learning. Advanced Lectures on Machine Learning. ML 2003. Lecture Notes in Computer Science*, pp. 72–112. Springer, Berlin, Heidelberg. [En línea] <https://link.springer.com/chapter/10.1007/978-3-540-28650-9_5>[Consulta: 30 julio 2020].
- [Gopal, 2019] Gopal, M. (2019). *Applied Machine Learning*. McGraw-Hill Education, New York, Chicago, San Francisco, Athens, London, Madrid, Mexico City, Milan, New Delhi, Singapore, Sydney, Toronto. [En línea] <<https://www.accessengineeringlibrary.com/content/book/9781260456844>>[Consulta: 22 julio 2021].
- [Guest *et al.*, 2018] Guest, D., Crammer, K., y Whiteson, D. (2018). Deep learning and its application to lhc physics. *Annu. Rev. Nucl. Part. Sci.*, 68:161–181. [En línea] <<https://www.annualreviews.org/doi/10.1146/annurev-nucl-101917-021019>>[Consulta: 25 noviembre 2021].
- [Harrison, 2011] Harrison, K. (2011). Discovering particles: fundamental building blocks of the universe. [En línea] <<http://www.ep.ph.bham.ac.uk/DiscoveringParticles/lhc/experiments/>>[Consulta: 28 julio 2020].
- [Hastie *et al.*, 2009] Hastie, T., Tibshirani, R., y Friedman, J. (2009). *The Elements of Statistical Learning*. Springer, 2a. ed. edición. [En línea] <<https://link.springer.com/book/10.1007/978-0-387-84858-7>>[Consulta: 29 julio 2020].
- [Kingsford y Salzberg, 2008] Kingsford, C. y Salzberg, S. L. (2008). What are decision trees? *Nature Biotechnology*, 26:1011–1013. [En línea] <<https://www.nature.com/articles/nbt0908-1011>>[Consulta: 24 julio 2021].
- [Kirchmeier, 2015] Kirchmeier, D. (2015). Reconstruction and Identification of Boosted Tau Pair Topologies at ATLAS. Tesis de máster, Dresden, Tech. U. <<https://cds.cern.ch/record/2105592>>[Consulta: 10 junio 2020].
- [Krogh, 2008] Krogh, A. (2008). What are artificial neural networks? *Nature Biotechnology*, 26(2):195–197. [En línea] <<https://www.nature.com/articles/nbt1386>>[Consulta: 31 julio 2021].
- [McCulloch y Pitts, 1943] McCulloch, W. S. y Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5:115–133. [En línea] <<https://link.springer.com/article/10.1007/BF02478259>>[Consulta: 30 julio 2021].

- [Minsky y Papert, e MA] Minsky, M. y Papert, S. (Cambridge, MA). *Perceptrons*. M.I.T. Press, 1969. [En línea] <<https://mitpress.mit.edu/books/perceptrons>>[Consulta: 31 julio 2021].
- [Myles *et al.*, 2004] Myles, A. J., Feudale, R. N., Yang Liu, N. A. W., y Brown, S. D. (2004). An introduction to decision tree modeling. *Journal of Chemometrics*, 18(6):275–285. [En línea] <<https://analyticalsciencejournals.onlinelibrary.wiley.com/doi/10.1002/cem.873>>[Consulta: 27 julio 2021].
- [Rosenblatt, 1960] Rosenblatt, F. (1960). Perceptron simulation experiments. *Proceedings of the IRE*, 48(3):301–309. [En línea] <<https://ieeexplore.ieee.org/document/4066017>>[Consulta: 31 julio 2021].
- [Rumelhart *et al.*, 1986] Rumelhart, D., Hinton, G., y Williams, R. (1986). Learning representations by back-propagating errors. *Nature*, 323:533–536. [En línea] <<https://www.nature.com/articles/323533a0>>[Consulta: 8 diciembre 2021].
- [Samuel, 1959] Samuel, A. L. (1959). Some studies in machine learning using the game of checkers. *IBM J. Res. Dev.*, 3:210–229. [En línea] <<https://www.semanticscholar.org/paper/Some-Studies-in-Machine-Learning-Using-the-Game-of-Samuel/e9e6bb5f2a04ae30d8ecc9287f8b702eedd7b772?p2df>>[Consulta: 25 noviembre 2021].
- [Sirunyan *et al.*, 2018] Sirunyan, A., Tumasyan, A., Adam, W., y *[et.al]* (2018). Observation of the higgs boson decay to a pair of τ leptons with the cms detector. *Physics Letters B*, 779:283 – 316. <<https://doi.org/10.1016/j.physletb.2018.02.004>>[Consulta: 2 julio 2020].
- [Sirunyan *et al.*, 2019] Sirunyan, A., Tumasyan, A., y *[et.al]* (2019). Search for a heavy pseudoscalar Higgs boson decaying into a 125 GeV Higgs boson and a Z boson in final states of two light leptons and two tau leptons at $\sqrt{s} = 13$ TeV. Technical Report CMS-PAS-HIG-18-023, CERN, Geneva. <<https://cds.cern.ch/record/2667215>>[Consulta: 2 julio 2020].
- [Sjöstrand *et al.*, 2015] Sjöstrand, T., Ask, S., Christiansen, J. R., Corke, R., Desai, N., Ilten, P., Mrenna, S., Prestel, S., Rasmussen, C. O., y Skands, P. Z. (2015). An introduction to pythia 8.2. *Computer Physics Communications*, 191:159–177. [En línea] <<https://www.sciencedirect.com/science/article/abs/pii/S0010465515000442?via%3Dihub>>[Consulta: 25 noviembre 2021].
- [Sjöstrand *et al.*, 2008] Sjöstrand, T., Mrenna, S., y Skands, P. (2008). A brief introduction to pythia 8.1. *Computer Physics Communications*, 178(11):852–867. [En línea] <<https://www.sciencedirect.com/science/article/abs/pii/S0010465508000441?via%3Dihub>>[Consulta: 15 diciembre 2021].
- [The ATLAS Collaboration *et al.*, 2020] The ATLAS Collaboration, Aad, G., Abbott, B., y *[et.al]* (2020). Reconstruction and identification of boosted di- τ systems in a search for higgs

boson pairs using 13 tev proton–proton collision data in atlas. *Journal of High Energy Physics*, 2020(163). [En línea] <[https://link.springer.com/article/10.1007%2FJHEP11\(2020\)163](https://link.springer.com/article/10.1007%2FJHEP11(2020)163)>[Consulta: 11 noviembre 2021].

[Zavala, 2021] Zavala, J. (2021). Repositorio boosted di-tau identification. [En línea] <https://gitlab.cern.ch/usm/boosteditauid/-/tree/python_analysis>[Consulta: 15 diciembre 2021].