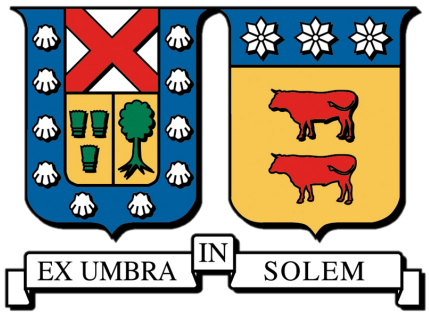
UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE ELECTRÓNICA
VALPARAÍSO - CHILE



Deep Learning Semi-supervised Strategy for Gamma/Hadron Classification of Imaging Atmospheric Cherenkov Telescope Events

Tesis de grado presentada por

Diego Vicente Riquelme Román,

como requisito parcial para optar al titulo de

Ingeniero Civil Electrónico

y para optar al grado de

Magíster en Ciencias de la Ingeniería Electrónica

Mención Telecomunicaciones y Procesamiento de Señales.

JULIO 2022

UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE ELECTRÓNICA
VALPARAÍSO, CHILE

TÍTULO DE LA TESIS:
DEEP LEARNING SEMI-SUPERVISED STRATEGY FOR GAMMA/HADRON CLASSIFICATION OF IMAGING ATMOSPHERIC CHERENKOV TELESCOPE EVENTS

AUTOR:
DIEGO VICENTE RIQUELME ROMÁN

Tesis de grado presentada por ***Diego Vicente Riquelme Román****, como requisito parcial para optar al titulo de* ***Ingeniero Civil Electrónico*** *y para optar al grado de* ***Magíster en Ciencias de la Ingeniería Electrónica***
Mención Telecomunicaciones y Procesamiento de Señales*.*

Mauricio Araya López
Profesor Guía

Deciembre 2022.
Valparaíso, Chile.

Abstract

En este trabajo se exploró una alternativa semi-supervisada de aprendizaje que permitiera al sistema de captación de imágenes de Cherenkov Telescope Array Observatory, detectar eventos producidos por rayos gamma, y distinguirlos de los -mucho más frecuentes- producidos por protones.

Como herramienta, se utilzó una red neuronal convolucional, donde se validó la estrategia de Cyclical Learning Rate, lo que permitió un entrenamiento más rápido. Además se presentó la oportunidad de estudiar el espacio latente de la red entrenada, y correlaccionarla con la estrategia convencional de parámetros de Hillas para, de cierta manera, poder comparar lo que estaba aprendiendo la red en el aspecto de clasificación y analizar si tiene sentido con lo que se conoce del fenómeno físico y sus estudios. Adicionalmente podemos compararlo con los parámetros físicos de Hillas lo cual -al ser la estrategia convencional- está bastante validada.

Debido a que el observatorio no está disponible aún, solo es posible entrenar los algoritmos con imágenes simuladas, por lo que una vez que se presenten los datos reales, no hay mucha certeza de como se comportarán. Por este motivo se propone una estrategia semi-supervisada, que permite entrenar el modelo con datos simulados y etiquetados. Para luego, mejorar las predicciones del modelo con datos simulados no-etiquetados. Esto se considera el trabajo preliminar para una mejor integración de los datos reales.

Los resultados muestran que este algoritmo es un atractivo candidato para aprovechar los datos reales no-etiquetados. Nuestro modelo fue capaz de seguir aprendiendo desde datos nunca antes vistos y de manera no-supervisada. Sin embargo debe motivarse la investigación en la integración de los datos reales. Adicionalmente, la exploración del espacio latente indicó que es una metodología atractiva para la exploración de CNN y su relación con los datos, su morfología y la tarea en cuestión.

Contents

Agradecimientos

Agradezco a Samuel Spencer por facilitarnos el HPC de Oxford y por su constante ayuda con Wilkes 3.

Agradezco también al grupo multidiciplinario al que me sume y que hizo posible esta tesis. Sebastian Borquez, Mauricio Araya, Boris Panes y Edson Carquin.

Agradecer especialmente al profesor Mauricio Araya por todo el apoyo y comprensión en los momentos dificiles. Me es grato agradecer también a Alejandro Quijada por su ayuda desinteresada, sin la cual este trabajo hubiera sido mucho más difícil.

Y al Centro AC3E, quienes brindaron un espacio agradable para trabajar.

Finalmente, deseo agradecer a mi familia y amigos, sin los cuales nada de esto hubiera sido posible.

This work was performed using resources provided by the Cambridge Service for Data Driven Discovery (CSD3) operated by the University of Cambridge Research Computing Service (www.csd3.cam.ac.uk), provided by Dell EMC and Intel using Tier-2 funding from the Engineering and Physical Sciences Research Council (capital grant EP/T022159/1), and DiRAC funding from the Science and Technology Facilities Council (www.dirac.ac.uk).

Chapter 1

Introducción

La investigación de rayos gamma es de gran interés en astronomía. Permite el estudio de eventos violentos y lejanos del universo. Recientemente hay una gran expectación debido a que podría, según algunos modelos teóricos actuales, aportar al estudio de Materia Oscura [1]–[5].

Motivado por este gran interés, está en construcción el proyecto más grande de Telescopios de imágenes atmosféricas de Cherenkov (IACT) del mundo, el *Cherenkov Telescope Array* (CTA) [6]. Este observatorio estará dividido en 2 instalaciones: una en Paranal, Chile y la otra en La Palma, España. Los Telescopios de Cherenkov son telescopios que se colocan sobre la superficie de la Tierra aprovechando la atmósfera como un detector. Esto se debe a que los rayos gamma interactúan con la atmósfera generando una *Extensive air shower* lo que produce un efecto lumínico llamado efecto Cherenkov, el cual es luego capturado por los telescopios. Lo bueno de esta estrategia es que aumenta el área de detección considerablemente. Lo problemático es que los rayos gamma no son los únicos eventos que producen efecto Cherenkov. En un observatorio actual (VERITAS), un arreglo de cuatro Telescopios de Cherenkov, la detección estaba dominada por protones con una frecuencia de 350 Hz, mientras los eventos de gamma tenían una frecuencia de detección de 1 Hz [7].

Debido a que la señal de interés son los rayos gamma, hay que diseñar un clasificador que filtre la señal del ruido. Los arreglos de telescopios existentes atacan el problema con clasificadores clásicos, como Random Forest [8] o Boasted Desicion Trees [7], [9], [10], para los observatorios de MAGIC, HESS y VERITAS respectivamente.

Debido a que CTA aún no está construido, el entrenamiento, refinamiento y modelado se debe hacer con eventos simulados por el software CORSIKA [11] y posteriormente las imágenes son procesadas con el software sim_telarray [12]. Idealmente se usarían imágenes reales, debido a que -pese a que las simulaciones son de excelente calidad-, un estudio demostró que se podía entrenar un clasificador para reconocer las imágenes simuladas de las reales [13]. Esto presenta un gran desafío, porque cabe la posibilidad que los clasificadores tengan un desempeño muy diferente al ser puestos a prueba en operación. Esta motivación revela una oportunidad para desarrollar técnicas que logren ser entrenadas

con datos simulados, pero sean capaces de añadir los datos reales en la medida que los datos estén disponibles.

Luego de la correcta clasificación del evento, la tarea que viene es extraer la información relevante de la imagen. A esta tarea se le denomina reconstrucción, e intenta inferir la energía y origen del evento. Actualmente la búsqueda es de técnicas que logren la estéreo clasificación y reconstrucción. Esto quiere decir que se logren hacer las tareas con la información de todos los telescopios que lograron capturar la imagen correspondiente al mismo evento. Esto presenta un desafío complejo debido a que hay 3 tipos de telescopios, el LST (Large Size Telescope), MST (Medium Size Telescope) y SST (Small Size Telescope) cada uno con resoluciones diferentes. Además habría que encontrar la mejor forma de aprovechar esta información.

En [14], los investigadores se proponen buscar técnicas de Deep Learning que compitan con el estado del arte, y exploran formas de toma de decisión con varios telescopios. Otros trabajos apuntan a desarrollar todo un flujo de trabajo que facilite el desarrollo de técnicas [15]. Luego hay trabajos que siguen dicho flujo y presentan nuevos resultados [16]. En otra investigación se trabaja en mono-telescopio y se prueban tres modelos diferentes que luego se comparan con Random Forest [17]. Mientras otros trabajan en redes convolucionales recurrentes (CRNN) que permiten el análisis estéreo con varios telescopios de distintas resoluciones [13].

Actualmente la mayoría de las lineas de investigación se focalizan en Deep Learning debido a que es el estado del arte hoy en día en clasificación en otros contextos. El trabajo aquí presentado, se desarrolla en este contexto, en donde se quiere aprovechar el uso de técnicas del aprendizaje profundo para la correcta detección de eventos producidos por rayos gamma. En particular, aprovechar el uso de datos reales adquiridos. Para esto, entrenaremos un modelo que logre clasificar los rayos gamma, y luego usaremos una técnica semi-supervisada para seguir mejorando el desempeño del clasificador. Se espera que la red mejore su desempeño a partir de eventos sin etiqueta de clase. De esta manera se estará dando un paso en la dirección de poder utilizar datos reales en el entrenamiento de las redes.

Chapter 2

Cherenkov Telescope Array

En este capitulo se hará una breve descripción del efecto que hace posible la observación de rayos cósmicos con telescopios terrestres.

2.1 Extensive Air Shower

La radiación de Cherenkov, descrita por el científico y premio nobel Pavel Cherenkov en 1934, consiste en la emisión de fotones, como consecuencia de la interacción de partículas que se desplazan a velocidades superiores a la velocidad de la luz en ese medio. Este efecto se va propagando, generando lo que se conoce como *Extensive Air Shower*, una ilustración de esta interacción se encuentra en la Figura 2.1.

La interacción de un rayo cósmico con la atmósfera produce varios hadrones altamente energéticos, que decaen a partículas más estables liberando radiación, los cuales son 2 de las componentes de una *Particle Shower*. De manera similar a como los aviones supersónicos generan un *Sonic Boom*, los rayos cósmicos generan luz de Cherenkov.

En el contexto de CTA la radiación emitida es capturada por los espejos de los telescopios y luego reflejada a los sensores, quienes la colectan digitalmente. Debido a que la primera interacción del rayo cósmico con la atmósfera ocurre a gran altura, la luz se dispersa sobre un gran área (e.g 250 metros), lo cual facilita su detección. Esta interacción dura una billonésima de segundo, por lo que los sensores de los telescopios son cámaras de alta velocidad sumamente sensibles.

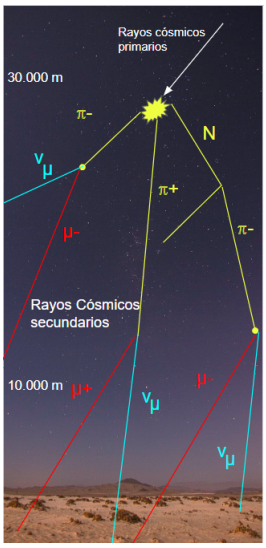


Figure 2.1: *Extensive Air Shower (Elaboración propia)*

Los rayos cósmicos pueden ser originados por rayos gamma (clase de interés), pero también es producido por protones, electrones, núcleos, entre otras partículas. Generalmente las capturas más frecuentes son producidas por protones, siendo éste, el ruido dominante.

2.2 Cherenkov Telescope Array

El arreglo de telescopios CTA Observatory (CTAO), consiste en 2 observatorios. Uno ubicado en Paranal, Chile y el otro en La Palma, España, ilustrado en Figura 2.2. Estos 2 observatorios cuentan con varios telescopios. En general hay tres tipos de telescopios *Short Size Telescope (SST), Medium Size Telescope (MST),*y *Large Size Telescope (LST)*, cada uno para capturar un rango de energías distintos. El SST captura desde 5[TeV]- 300[TeV], mientras el MST captura desde 150 [GeV] hasta 5[TeV] y finalmente el LST recorre desde 20[GeV] a 150[Gev]. Por lo que el rango completo de energía de CTA se considera de 20[GeV] - 300[TeV], como se ilustra en la Figura 2.3. Cada telescopio cuenta también con diferentes resoluciones. Además los telescopios no están uniformemente distribuidos en las dos localidades, *Northern* CTAO (La Palma) cuenta con 4 LST y 15 MST cubriendo un área de aproximadamente $0.25km^2$. Mientras *Southern* CTAO (Paranal) cuenta con 4 LST, 25 MST y 70 SST.

El CTAO proporcionará un rango de energía muy amplio y una resolución angular y sensibilidad excelentes en comparación con cualquier detector terrestre de rayos gamma existente. Las energías de hasta 20 GeV permitirán a CTAO estudiar los objetos más lejanos. Mientras las energías de hasta 300 TeV llevarán al CTAO más allá del límite del espectro electromagnético conocido, proporcionando una visión completamente nueva del cielo, como se ilustra en la figura 2.3.

Figure 2.2: Observatorio CTAO en La Palma, España (Rescatado de CTA)

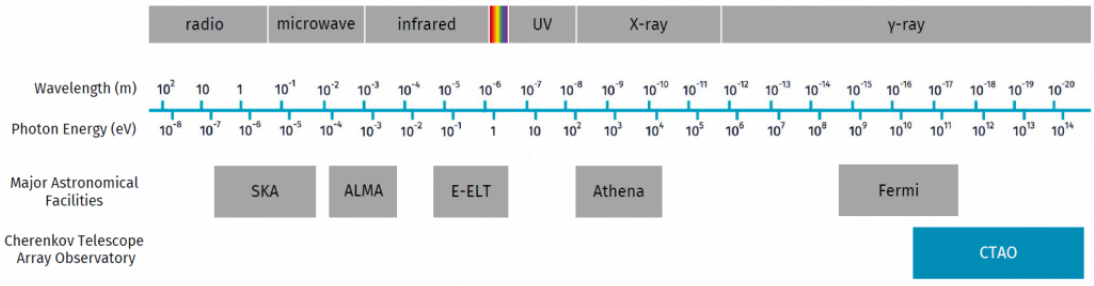


Figure 2.3: rango de energia de CTA (Rescatado de CTA)

Debido al aumento de rango de energía de captación, es de suma importancia identificar los eventos gamma de manera adecuada, de los muchos otros eventos que también producen luz de Cherenkov. Es en este contexto que se necesita un discriminador de partículas.

Chapter 3

Clasificación de Gamma/Hadrón

3.1 El problema

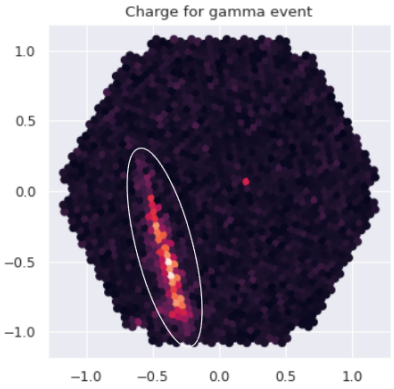
La luz de Cherenkov que captan los telescopios no es exclusiva de los rayos gamma, sino que también son producidas por protones, electrones, etc. El problema, es que estas otras partículas tienen ocurrencias mucho más frecuentes. Por ejemplo, en VERITAS la captación esta dominada por protones 350 [Hz], mientras los gamma sólo tienen una frecuencia de 1 [Hz][7]. Esto presenta problemáticas con el balance de las clases y una asimetría entre la importancia de las clases. En otras palabras, ya que la mayoría del trabajo se basa en clasificar gamma/protón, no es lo mismo equivocarse en la clasificación de un abundante protón, que de un escaso gamma.

3.2 Soluciones Actuales

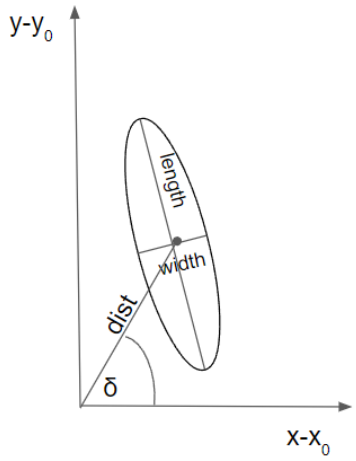
Este problema ya ha sido atacado con diferentes enfoques por los ya existentes telescopios de efecto Cherenkov. En los enfoques actuales, lo que se hace es parametrizar la imagen capturada por el telescopio. La parametrización que más se utiliza se denomina parametrización de Hillas[18], ejemplificada en la Figura 3.1. Esta caracteriza parámetros geométricos de la imagen, donde a partir de las diferencias en sus distribuciones, se puede hacer inferencia acerca del origen de la partícula [19].

Por ejemplo, en VERITAS y H.E.S.S, dos observatorios terrestres con telescopios de efecto Cherenkov, solucionaron este problema utilizando *Boosted Decision Trees (BDT)* con parámetros derivados de los parámetros de Hillas. Con esta estrategia, se logra hacer un clasificador (gamma/hadrón) y un regresor, para determinar la energía y posición de la fuente junto con la energía con la que la partícula interactuó con la atmósfera. Este clasificador, en general, logra un buen desempeño, y es robusto entre las imágenes reales/simuladas, debido a la simpleza de parametrizar la imagen que lleva a cabo la solución.

Por su lado, MAGIC, hace la misma tarea con *Random Forest* (RF), utilizando los



(a) Imagen captada por la cámara



(b) Parametrización de Hillas

Figure 3.1: Parámetros de Hillas, ejemplificado (Elaboración propia)

parámetros de Hillas como entrada. Con frecuencia a esta estrategia se le denomina Hillas-RF. La cual también logra un buen desempeño en imágenes reales. En parte, su éxito se le atribuye a la simpleza de la solución. Ya que parametrizan la imagen a aproximadamente 12 parámetros. En esa simplificación se codifican precisamente las características morfológicas que permiten la distinción entre protón-gamma.

Ambas estrategias son utilizadas, y debido a su rendimiento, siguen utilizándose en dichos observatorios. Producto de la propuesta ambiciosa de CTA, se quieren implementar modelos (clasificadores y regresores) del estado del arte en la literatura actual de *Machine Learning*.

3.3 Nuevas Propuestas

En el estado del arte, se pueden encontrar las Redes Neuronales Convolucionales (CNN). (ver Capitulo 5). Estos modelos tienen la ventaja de que utilizan la imagen completa como entrada, sin necesidad de ser parametrizada. Esto presenta la ventaja de que puede aprovechar de aprender las características morfológicas más sutiles, y no tan generales como los parámetros de Hillas. El problema que presenta esto es que requiere de muchos datos de entrenamiento y podría aprender particularidades exclusivas de los datos simulados, llevando a un mal desempeño cuando es enfrentado a datos reales [13], [20].

Debido a su popular uso en variadas aplicaciones, trabajos actuales intentan aplicar CNN en diversas arquitecturas. Buscando modificaciones que permitan integrar varios telescopios, con distintas resoluciones, a la estimación de la tarea en cuestión. Por ejemplo, en [13], los autores implementan una arquitectura con redes neuronales recurrentes que pueden admitir como entrada imágenes de distinta resolución, presentando un posible enfoque para aprovechar la información de varios telescopios. Otros autores diseñan

librerías de trabajo de procesamiento de imágenes de telescopio que consideran en su flujo de datos la aplicación de arquitecturas de Deep-learning para que sean aprovechados en el diseño [16]. Mientras la mayoría de los estudios fueron entrenados con datos simulados, algunos de ellos comparan con datos reales [13], [21], [22] o fueron entrenados/testeados con una combinación de ambos [8], [23].

3.4 Nuestra propuesta

Nuestra propuesta trabaja sobre el problema de clasificación gamma/protón y propone una estrategia semi-supervisada que busca mejorar el rendimiento de redes entrenadas con datos etiquetados. Propone una metodología para aprender a partir de datos diferentes sin etiqueta. Debido a que los modelos deben ser entrenados de manera supervisada con datos simulados, ya que es la única forma de estar seguros de la partícula de origen (i.e. de la etiqueta), hay una brecha con los datos reales que debe ser suplida (ver Capitulo 4). Es en este contexto que proponemos un método que podría usarse en datos reales de manera no-supervisada, de modo de no necesitar las etiquetas de las imágenes y que pueden ayudar a mejorar el desempeño de la red al ir aprendiendo y modificando las características aprendidas de los datos simulados, pero ahora con datos reales.

Para esto primero diseñamos una CNN la cual entrenamos de manera supervisada con datos simulados y etiquetados. Luego, re-entrenamos con otros datos simulados, ahora sin etiqueta, utilizado una versión modificada de una estrategia denominada Pseudo-label (ver capitulo 6). Este experimento permitirá validar nuestras modificaciones de la estrategia semi-supervisada. Luego el segundo experimento permitirá validar la estrategia propuesta al entrenar la red con datos de alta intensidad de manera supervisada y luego re-entrenarla con datos de menor intensidad y verificar que la red aprende características nuevas, mejorando las métricas de rendimiento. Todos los experimentos son explicados en detalle en el capítulo 7.

La idea principal de nuestra propuesta es mostrar una estrategia que permita, hacer un *fine-tuning* de manera no supervisada, a un modelo ya entrenado de manera supervisada. Esto se propone en el contexto de CTA, donde los modelos son entrenados de manera supervisada con datos simulados, y luego son expuestos a datos reales, donde no se conoce con certeza la naturaleza de la partícula que origina la imagen capturada. Luego este modelo podría someterse a un *fine-tuning* con nuestro método y mejorar su representación de las imágenes capturadas, y por lo tanto mejorar su desempeño.

Chapter 4

Datos

Con la construcción de nuevos observatorios, el entrenamiento de nuevos clasificadores que logren discriminar la partícula de origen, deben generarse a partir datos simulados debido a que los datos reales no están disponibles y las ventajas que estos presentan. Entre sus virtudes, se encuentra que se puede tener certeza de la partícula de origen, además se pueden generar datos en los balances que se deseen con las partículas que se requieran. Adicionalmente, se pueden probar distintas configuraciones espaciales de los telescopios y distintos tipos de telescopios que logren el mejor desempeño. Esto se presenta como una herramienta sumamente útil para poder diseñar la mejor configuración posible de los telescopios, previo a su construcción.

4.1 Prod5_DL1

Para la reproducibilidad de este trabajo se describirán, a continuación, los datos utilizados y como estos son creados. Los datos simulados, fueron facilitados gracias al CTA consortium[6]. El dataset corresponde al Prod5_DL1. Este dataset tiene simulaciones de testeo para gamma-difuso, gamma puntual, protón, electrón. Además cuenta con un subset para entrenamiento compuesto de gamma difuso y protones. La razón de que sólo se entrene con gamma-difuso y protón es debido a varias razones. Por un lado, se utiliza gamma-difuso para que la red no genere un sesgo respecto a la fuente de origen. El protón se utiliza debido a que, al ser la partícula más frecuente, su correcta clasificación aumentaría el *Signal to Noise Ratio* (SNR). Por otro lado las *Extensive Air Showers* producidas por electrones, son menos frecuentes y de dificil clasificacion, ya que comparte muchas características morfológicas con los rayos gamma. Además, no se ha logrado caracterizar bien su comportamiento en los programas que simulan sus *Extensive air showers*, por lo que no se depende de ellos a la hora de entrenar un clasificador.

Estas simulaciones se llevan a cabo en dos etapas. Primero se simula la fuente, en este caso un rayo cósmico, y su interacción con la atmósfera, generando luz de Cherenkov. La segunda etapa se dedica a simular la interacción con el telescopio y a generar la respuesta en los sensores correspondientes. Esto se lleva a cabo con dos programas diferentes [12].

La ventaja de contar con datos simulados es que permite generar datos con el balance que se desee, probar distintas configuraciones de los telescopios y generar datos variados en sus distintos parámetros (algo que es de suma importancia para entrenar algoritmos de Machine Learning). Esto último permite al algoritmo generalizar de mejor manera al ser entrenado con datos en todo su espacio muestral. En el contexto de Cherenkov, esto se traduce en generar datos en un amplio rango de energías, lo que permitirá al modelo aprender las diferentes características que forman las distintas partículas a lo largo de un amplio rango de energía, condiciones atmosférica, fuentes, alturas de interacción, etc. para de esta manera poder clasificar su fuente de mejor manera.

En el caso particular del Prod5_DL1, se cuenta con 1.7 Tb de datos dispuestos entre 4 casos distintos de fuentes. Los detalles se muestran en la Tabla 4.1:

Table 4.1: Data Details

Type/Size	gamma	gamma-diffuse	proton	electron	TOTAL
Test (Gb)	772	72	161	77	1082
Training (Gb)		309	326		635

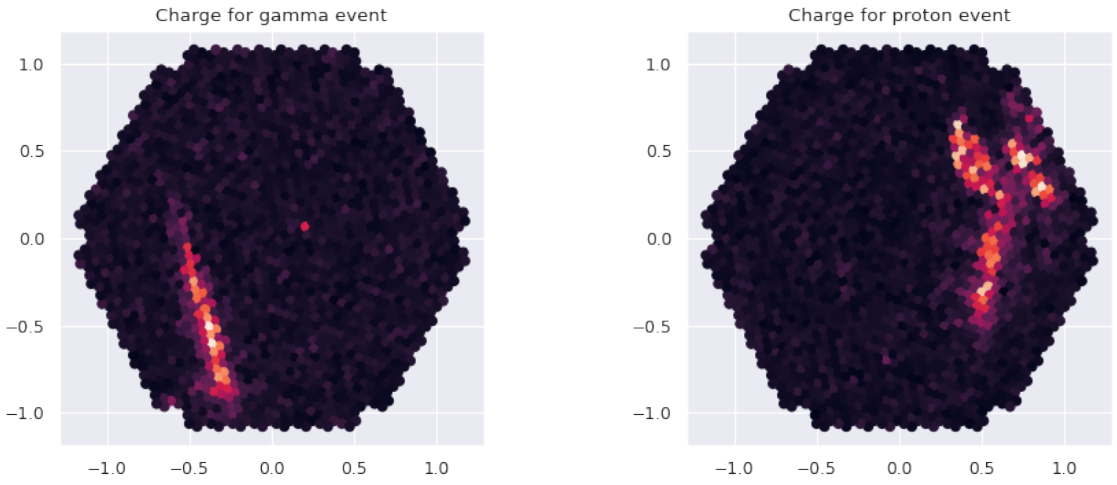
En general, los datos simulados son creados en dos etapas. La primera simula la fuente y como ésta interactúa con la atmósfera, generando el efecto Cherenkov, esto se logra con el software CORSIKA [11]. Luego las imágenes son obtenidas con ayuda del software sim_telarray [12] que simula la interacción de la luz de Cherenkov con los sensores de los telescopios, generando una imagen. Las imágenes que se generan de este proceso se ejemplifican en la Figura 4.1.

Debido a que las simulaciones de CORSIKA son cadenas de Markov, se le suelen llamar *MC data/MC simulation* (por sus siglas en ingles) a los datos simulados. En particular, las *MC simulations* son similares a las presentadas en [24], pero usando CORSIKA 7.71 (con URQMD + QGSJET-II-04), una actualización en el modelo del detector de los telescopios del CTAO(CTA Observatory), y un arreglo optimizado llamado *Production 5* o *Prod 5*. Además los datos utilizados corresponden al DL1 lo que significa que el contenido de las imágenes está procesada, obteniendo finalmente una imagen de tres canales, uno con la intensidad del píxel en carga y otro con el tiempo de llegada de la partícula luego del pulso de calibración del píxel, y el último canal con la mascara de los pixeles activados. Para mayor información acerca de los datos, referirse a la documentación de CTA [1].

4.2 Datos Reales y *MC simulations*

Idealmente se usarían imágenes reales, debido a que, pese a que las simulaciones son de excelente calidad, un estudio demostró que se podía entrenar un clasificador para reconocer las imágenes simuladas de las reales con gran precisión [13]. Esto presenta una grave problemática, porque cabe la posibilidad que los clasificadores tengan un desempeño muy

[1] https://www.cta-observatory.org/



(a) Imagen de gamma

(b) Imagen de protón

Figure 4.1: Imágenes simuladas de gamma y protón para LST. (Elaboración propia)

diferente al ser puestos a prueba en operación al estar expuestos a datos reales. Esto presenta una oportunidad para desarrollar técnicas que logren ser entrenadas con datos simulados, pero sean capaces de añadir los datos reales en la medida que estos estén disponibles.

Además los datos simulados presentan algunas ventajas no compartidas por su pares reales. Por ejemplo, se pueden generar simulaciones de partículas en las proporciones deseadas, cosa que favorece al algoritmo entrenado y que no se presenta en los datos reales, debido a su alto desbalance, natural de la incidencia de partículas. También permite generar distintos sensores de los telescopios, probar diferentes configuraciones espaciales entre los telescopios, con el objetivo de mejorar la detección y poder probar todo ésto, previo a la construcción del mismo, para de esta manera, minimizar los riesgos.

4.3 Sub-sets de datos para los Experimentos

Para los experimentos diseñados fue necesario dividir los datos en base a ciertos criterios. Estos criterios dependen del experimento en cuestión. Sin embargo para mejorar el entrenamiento de la red, se utilizaron datos que superaran cierto criterio de calidad. En el trabajo aquí presentado, el corte se hizo sobre el parámetro de intensidad de Hillas.

El parámetro de intensidad de Hillas mide la cantidad de carga capturada por los distintos píxeles del telescopio. Este parámetro se mide en *photo-electrons (phe)*.

Para los experimentos se utilizaron los cortes de alta intensidad ($>$1000 phe) y los de mediana intensidad ($>$300 phe). Los cortes de alta intensidad implica que el evento esta bien centrado, que el telescopio logro captar la totalidad del evento y no solo una porción.

O bien que la sección parcial capturada del evento correspondía a uno de alta energía. Generalmente cuanto mayor es la energía de los eventos es más fácil clasificarlos, debido a que las diferencias morfológicas de la *Extensive Air Shower* se hacen más evidentes entre las distintas partículas.Para el detalle de los datos utilizados separados por experimento, referirse a la tabla 7.1.

Chapter 5

Convolutional Neural Networks

Las redes neuronales convolucionales son una estrategia de Aprendizaje de Máquinas, un desarrollo que proviene de una categoría más genérica, llamada Redes Neuronales Artificiales.

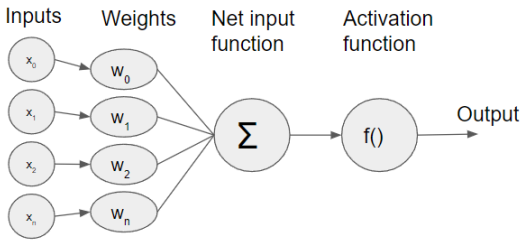


Figure 5.1: Modelo del Perceptron (Elaboración propia)

Para entender las redes neuronales, hay que introducir de su unidad básica, el perceptron (Figura 5.1), desarrollada por el psicólogo Frank Rosenblatt en el año 1958 [25], [26]. Si bien el trabajo de Rosenblatt descansaba sobre investigaciones neuronales del siglo XIX [27], se le suele identificar como el padre del Deep Learning.

El modelo del perceptron intenta simular una neurona en su expresión más sencilla. Su modelo se puede expresar en términos de estímulos y salidas, según la siguiente expresión:

$$y = \sum_{i}^{N} f(w_i x_i + b), \tag{5.1}$$

donde x_i serían las entradas (o estímulos), w_i los pesos que ponderan cada entrada, en otras palabras la importancia que se le da a esa entrada, como se muestra en la Figura 5.1. Y b es un *bias* que le agrega complejidad al modelo, tomando el papel de un umbral para las entradas. Luego la función $f(\cdot)$ es una función no lineal de respuesta de la neurona, llamada función de activación que suele ser una tangente hiperbólica, y N es la cantidad de entradas.

Si estas neuronas se van apilando en capas unas con otras se forman las redes profundas donde las salidas de las neuronas se convierten en las entradas de las neuronas

de la siguiente capa. La apilación de neuronas en varias capas dio paso al desarrollo del *Deep Learning* (Aprendizaje Profundo), el cual al tener mayor cantidad de capas, tiene el potencial de resolver problemas más complejos. El desarrollo del *backpropagation* permitió la modificación de los pesos que minimicen una función de pérdida y, en consecuencia, que maximice el rendimiento en alguna tarea.

Este sencillo modelo y su posterior exploración permitió el desarrollo de clasificadores/regresores y amplió el campo de estudio de metodologías supervisadas y no-supervisadas. Actualmente se sitúan como estado del arte en una serie de aplicaciones y tareas.

En general, hoy se han desarrollado una variedad de funciones de activaciones y capas. Y se sigue investigando cuales maximizan el rendimiento de las redes. Para que la red aprenda los valores de los pesos w_i que minimicen la función de pérdida con el fin de alcanzar un objetivo, hay una variedad de parámetros que afectan el proceso de aprendizaje de la red: estos parámetros se conocen como hyperparámetros. Estos regulan la cantidad de neuronas por capa, la cantidad de capas, tipos de capas, normalización, parámetros de las capas, las funciones de activación, la composición de las diferentes capas, la tasa de aprendizaje (learning rate), los valores con los que se inicializa la red, entre otros más.

En la medida que distintas soluciones van surgiendo, estas configuraciones de hyperparámetros se van usando en distintos contextos. A la configuración estructural de la red (numero de capas y numero de neuronas) se le denomina *arquitectura*. Luego, se van utilizando las mismas arquitecturas para diferentes problemas, validando la capacidad de estas redes de generar soluciones. Estas constantemente se van actualizando y reutilizando para distintas aplicaciones.

5.1 Redes Convolucionales

Las redes neuronales convolucionales (CNN) nacieron como respuesta a las investigaciones de las neuronas presentes en la retina. En el año 1959, Hubel y Wiesel investigaron la respuesta individual que tenían las neuronas de la retina a su campo receptivo [28]. Posteriormente, en el año 1980 Fukushima dio origen al *neocognitron* que -inspirado por las investigaciones previas de Hubel y Wiesel-, desarrolló dos capas: las convolucionales y las de downsampling. La capa convolucional corresponde al modelo de la neurona, cuyos pesos emulan el campo receptivo de las neuronas de la retina, cubriendo un parche (campo receptivo) de la capa anterior, cuyos pesos se le suele llamar filtro o kernel. Usualmente, las capas de neuronas, van avanzando en complejidad de sus filtros. Las capas de downsampling se encargan de reducir la dimensionalidad de la imagen de salida de las capas convolucionales y estas también cubren un campo receptivo. Típicamente se utilizaba el cálculo del promedio de las activaciones de las unidades en el campo receptivo. Luego esta capa fue reemplazada por la de Max-pooling la cual computa la máxima activación de las unidades del campo receptivo, y busca forzar la compresión de la información. El max-pooling es lo que se utiliza en las CNN modernas.

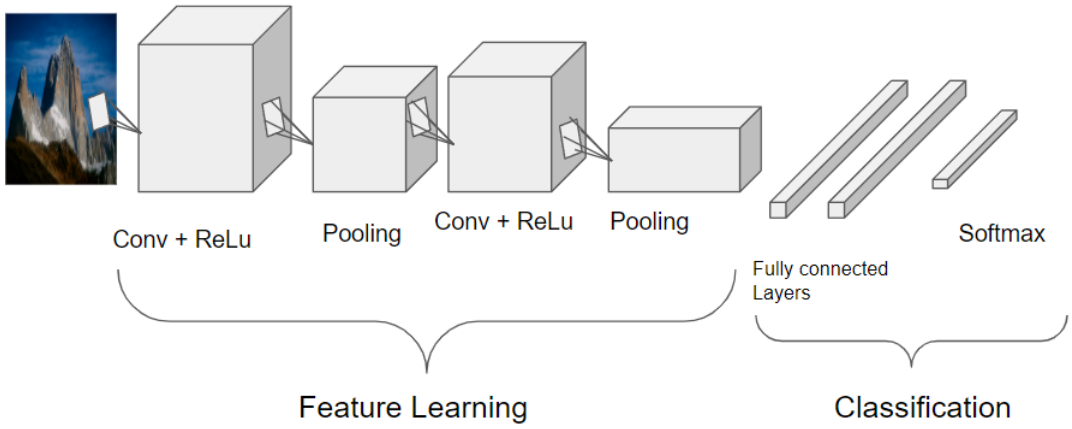


Figure 5.2: CNN genérica (Elaboración propia)

Pese a que las CNN fueron un desarrollo inspirado en las células de la retina, hoy no sólo se limitan a aplicaciones de visión y procesamiento de imágenes. Debido a sus implementaciones en GPUs (*Graphical Processing Unit*), lo que favorece su tiempo de entrenamiento e inferencia, hoy en día se utilizan en una variada gama de aplicaciones. Actualmente son ampliamente utilizadas llegando a alcanzar rendimientos del estado del arte en reconocimiento de imágenes, análisis de vídeo, procesamiento de lenguaje natural, pronóstico de series de tiempo, etc.

Para la clasificación, una CNN típica cuenta con una estructura que se puede entender en dos etapas. La primera, denominada *encoding* (o *feature learning* en la figura 5.2) entrena las capas convolucionales para obtener características morfológicas que permitan minimizar la función de costo. Luego, la segunda etapa, denominada *lógica* (o *classification* en la figura 5.2) genera compuertas lógicas a partir de las características obtenidas en la etapa anterior. Generalmente, la etapa de *encoding* se compone de filtros generales en sus primeras capas, por ejemplo detectores de bordes, cambios de intensidad en ciertas direcciones, etc. Luego en las capas más profundas comienzan a generar características que van creciendo en complejidad y abstracción. La capa lógica se suele componer de *feed forward networks*, también conocidas como capas densas, que corresponden a capas de neuronas que van decreciendo hasta alcanzar el número de clases de clasificación del problema.

Normalmente una CNN se compone de una combinación de capas convolucionales, max-pooling, batch normalization (una capa de normalización de las entradas que permite entrenamientos mas rápidos y estables), y capas densas, cuya configuración se determina con metodologías como *grid search*, random search, etc.

5.2 Nuestra Red

Para nuestra propuesta nosotros usamos una red basada en VGG16 [29], debido a las diferencias en dimensión, acortamos la profundidad de la red, y además modificamos los parámetros. Para encontrar los parámetros correctos hicimos una búsqueda tipo grilla (grid search). A modo de ejemplo en la Tabla 5.1 mostramos la grilla con algunos de los experimentos que se usaron para determinar el *learning rate*.

Table 5.1: Experiment with tiny dataset and different Learning rates and adam optimizer, the CNN is the default gerumo network

20 epochs LST (tiny dataset)	Default CNN	Default CNN	Default CNN
LR	0.001	0.01	0.1
slurm-code	56115987	56103238	56110068
validation acc	0.6151	0.5804	0.5730
Time [min]	84.1	84.928	84.8

De la tabla 5.1, podemos apreciar el *learning rate* (LR), el codigo de procesamiento en el HPC (slurm-code), la precisión medida en validación (validation acc) y el tiempo de ejecución del entrenamiento (Time).

Solo para fijar parámetros de la arquitectura hicimos aproximadamente 70 experimentos. Debido a que estos experimentos los hacíamos sobre una sección pequeña del dataset que no tenía cortes de calidad (sobre intensidad de Hillas) el aprendizaje de la red se veía dificultado debido al gran volumen de imágenes cuya clasificación es complicada, debido al rango de energías en el que se encuentran, o al centrado que podría tener la imagen. Luego, una vez seleccionado un parámetro se procedía a experimentar con el siguiente.

5.3 Arquitectura

Luego de una amplia búsqueda de arquitectura y diferentes valores de las capas intermedias, se llegó a la arquitectura presentada en la Tabla 5.2. La cual fue inspirada en VGG16 como modelo base, debido a la diferente dimensionalidad, se modificó la arquitectura para que la dimensión no disminuyera demasiado.

Con esta arquitectura final, que contiene 7,299,714 de parámetros entrenables, nosotros entrenamos la red para obtener un desempeño que nos permitiera sacar conclusiones de clasificación. Para ésto nos propusimos igualar o superar el rendimiento obtenido con un clasificador Random Forest entrenado con los parámetros de Hillas como entrada. Los parámetros de entrenamiento que se usaron con el objetivo de obtener un resultado competitivo con Random Forest son los que se utilizaron en Protopipe [30].

Luego, el objetivo era alcanzar con el entrenamiento de la CNN al clasificador RF. Para entrenar ambas redes se utilizó el corte de calidad de sobre los eventos. Para este caso

Table 5.2: Summary of CNN Architecture

Layer	**Shape**
Input	(55,47,3)
ConvLayer	(53,45,64)
ConvBock	(26,22,128)
ConvBlock	(13,11,256)
ConvBlock	(6,5,512)
ConvLayer	(6,5,512)
Flatten	(15360)
Dense	128
Dense	128
Dense	64
Output	2

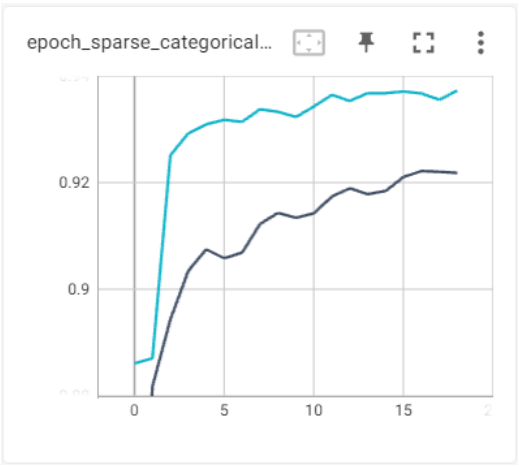
Table 5.3: ConvBlock Layer

ConvBlock	**Kernel**	**Filters**
ConvLayer	128 256 512	5x5 3x3 3x3
Dropout	0.25	
ConvLayer	256 512 1024	5x5 3x3 3x3
BatchNormalization		
MaxPool		2x2

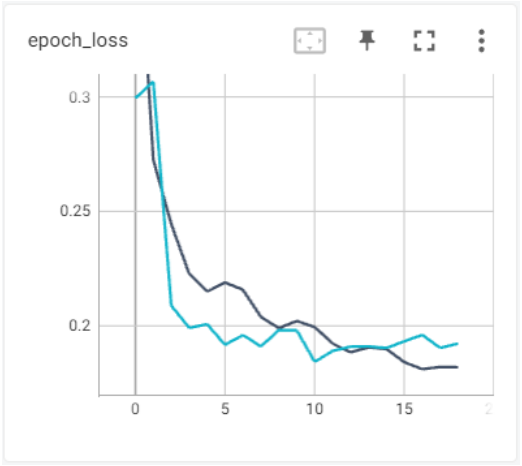
se utilizó un corte $>$1000 phe en el parámetro de intensidad de Hillas y se utilizó todo el dataset que cumpliera con ese criterio de calidad. Además se testeo sobre una porción de los datos correspondiente al 20%.

Adicionalmente para entrenar la CNN se utilizó *Cyclical Learning Rate*, lo cual es una técnica de entrenamiento que va regulando el learning rate a lo largo del entrenamiento (ver Capitulo 5.4).

Como se aprecia de la Figura 5.3, la red alcanza su *plateau* luego de aproximadamente 15 epochs. Alcanzando un *accuracy* de 93.57%, superando al RF que alcanzó solo un 87.58% sobre el mismo dataset. Hay que notar la diferencia en el tiempo de entrenamiento, donde la CNN se demoró aproximadamente 20hrs, mientras RF solo 2hrs, un orden de magnitud menos. Esto se debe a que RF usaba solo 12 parametrizaciones de la imagen, mientras CNN la imagen completa. Además los parámetros de entrenamiento de RF eran considerablemente menores. Sin embargo se cumplió con el objetivo de superar a RF por lo que se usó esa arquitectura con los parámetros de entrenamiento descritos en la tabla 5.4.



(a) Accuracy de la CNN, el entrenamiento corresponde al gris y el validation al azul



(b) Perdida de la CNN, el entrenamiento corresponde al gris y el validation al azul

Figure 5.3: Imágenes simuladas de gamma y protón para LST.

Hyperparameters	Value
Base LR	0.001
batch size	128
Factor	2
Max LR	0.02
Loss	SparseCategoricalCrossentropy
Optimizer	SGD
Momentum	0.9

Table 5.4: Tabla con los valores de los hyperparametros

5.4 Cyclical Learning Rate

Esta es una técnica de modificación del learning rate (lr) a lo largo del entrenamiento, propuesta el 2015 por Smith et al. en [31]. Siendo el learning rate uno de los hyperparámetros más importantes, su correcta configuración es con frecuencia una de las tareas más demandantes en el entrenamiento de una red neuronal. Si es muy grande, el modelo recorrerá el espacio de soluciones demasiado rápido, llevando a comportamientos divergentes o probablemente termine en un óptimo local, mientras si lo hace muy lento, tomará demasiados recursos computacionales. Generalmente se utilizan dos estrategias: lr adaptativo, donde el *learning rate* va se va modificando en la medida que avanzan las épocas de entrenamiento. Normalmente, se comienza con un lr alto, para acelerar el aprendizaje y luego empieza a decaer, haciendo una búsqueda más delicada del óptimo. Y la otra estrategia lr constante, lo que implica que el valor del *learning rate* no cambia en el tiempo y se mantiene fijo.

La técnica de Cylical learning rate (CLR), lo que hace es variar el lr en un rango. La idea detrás de esto es proponer una estrategia para poder salir de los óptimos locales o

Figure 5.4: Learning Rate schedule

puntos silla -ambos puntos críticos en el espacio de soluciones-, en caso de encontrarse en uno de ellos. Esta tarea se dificulta con la estrategia convencional de lr que decaen monótonicamente, donde en caso de estar en un óptimo local, no logrará acumular suficiente gradiente para salir de ese estado. Mientras que con una política de CLR, si podría salir de estos estados, al tener máximos lr con frecuencias de pocas épocas.

Las ventajas de utilizar CLR, son en varios aspectos. En el trabajo original prueban la metodología en varias arquitecturas y problemas diferentes, y en todas alcanzaba valores casi óptimos en tan solo una fracción de tiempo que las otras metodologías. Además ésta estrategia actúa como un regularizador al utilizar grandes valores de lr. Además al ser una estrategia de fácil implementación reduce considerablemente los tiempos de entrenamiento.

En el trabajo original se proponen varios tipos de maneras en las que se puede modificar el lr, e.g. ventana triangular, triangular con decaimiento exponencial, etc. Sin embargo en nuestra implementación nosotros aplicamos un ventana triangular con decaimiento exponencial como se muestra en la figura 5.4.

Para la elección del rango máximo y mínimo del lr, el autor propone una estrategia llamada *LR Finder*. Entrenar el modelo unas pocas épocas con un barrido de lr, y en su gráfica de la pérdida buscar los rangos con mayor pendiente, ya que ésto significa que el modelo esta reduciendo su pérdida a una tasa deseable. La estrategia de *LR Finder* se presenta en la figure 5.5.

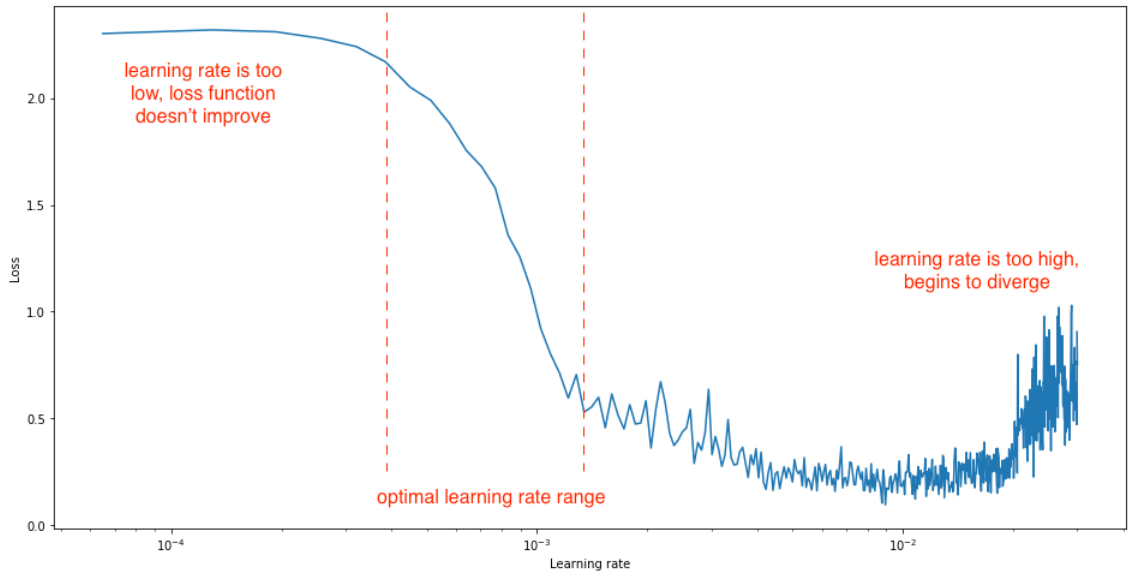


Figure 5.5: LR finder

Chapter 6

Pseudo-Label

En este capitulo se discutirá el método de Pseudo-labeling en su propuesta original y cómo fue modificado para el contexto de la propuesta de este trabajo.

6.1 Pseudo-Labeling

La estrategia de Pseudo-labeling fue propuesta por primera vez en [32] el año 2013. El autor presenta una propuesta semi-supervisada que permite entrenar un modelo en un contexto donde los datos etiquetados son una pequeña proporción de los datos totales.

Generalmente, una de las características que tienen los modelos de *Deep Learning* es su dependencia de un gran volumen de datos para poder representar bien el fenómenos subyacente a ellos. Esto se debe a que la red tiene que aprender que pesos (ver capitulo 5) minimizan la función pérdida (*Loss function*) de una tarea en particular. Para esto debe contar con datos etiquetados, para poder calcular el error y poder propagarlo por la red, modificando los pesos.

El autor, en su propuesta, presenta una modificación de la función de pérdida, haciendo una función mixta, que permite el cálculo convencional de la pérdida supervisada junto con una parte no supervisada, ponderada por un coeficiente de balance. Esto se aprecia en la ecuación 6.1. Donde n es el número de *mini-batches*(pequeñas agrupaciones de datos, utilizados para el cálculo de la perdida) para los datos etiquetados, y n' para los datos no etiquetados, f_i^m es la salida de la red para m muestras supervisadas, y_i^m son sus respectivas etiquetas, $f_i^{'m}$ es su equivalente para datos no etiquetados, $y_i^{'m}$ son las *Pseudo-labels* de las muestras no etiquetadas. $\alpha(t)$ es el término de balance entre ambas pérdidas.

$$L = \underbrace{\frac{1}{n}\sum_{m=1}^{n}\sum_{i=1}^{C} L(y_i^m, f_i^m)}_{\textit{Supervised Loss}} + \alpha(t)\underbrace{\frac{1}{n'}\sum_{m=1}^{n'}\sum_{i=1}^{C} L(y_i^{'m}, f_i^{'m})}_{\textit{Unsupervised Loss}} \tag{6.1}$$

Para obtener las *Pseudo-Labels* de los datos no etiquetados, se propone la siguiente

ecuación 6.2, donde y'_i es la etiqueta generada para el dato sin etiqueta y $f_i(x)$ es la predicción del modelo para el dato x:

$$y'_i = \begin{cases} 1 & \text{if i = } argmax f_i(x), \\ 0 & \text{otherwise} \end{cases} \tag{6.2}$$

lo que plantea esta ecuación, es que para las imágenes sin etiquetas, se genera la etiqueta con la predicción del modelo, como si fuera la etiqueta real. Por esto, se puede usar directamente en el cálculo convencional de pérdida de la red.

Por otro lado, el término $\alpha(t)$, que corresponde al coeficiente de balance, va cambiando con el tiempo. Al comienzo debe ser bajo, favoreciendo la pérdida supervisada. De esta manera, el modelo aprende una representación de los datos etiquetados. Cuando el modelo ya tiene un rendimiento considerable, se comienza a ponderar a favor de la pérdida no-supervisada, la cual ya puede generar *pseudo-labels* a partir de los datos sin etiqueta. Finalmente, el coeficiente favorece a la pérdida no supervisada, logrando aprender del volumen de datos no etiquetados.

Esta técnica hace dos suposiciones del espacio latente: la primera se denomina *Continuity Assumption*, que estipula que puntos que son cercanos, probablemente comparten clase. La segunda, *Cluster Assumption*, indica que los datos tienden a formar clusters, donde los puntos que comparten un cluster probablemente comparten la misma clase.

Esta propuesta se presenta en el contexto donde es difícil etiquetar los datos o simplemente no se cuenta con un gran volumen de datos etiquetados y, por lo tanto, se cuenta con un pequeño sub-conjunto de datos etiquetados y con un gran conjunto de datos no-etiquetados. En este contexto, esta propuesta resulta sumamente útil para aprovechar la totalidad de los datos. Pero en nuestro contexto, solo se cuenta con datos simulados para el entrenamiento y luego solo se cuenta con datos reales los cuales, pasado un tiempo, serán considerablemente mayores a los simulados. Debido esto se propone ciertas modificaciones al algoritmo original de *Pseudo-label*.

6.2 Pseudo-Label Modificado

$$self\_supervised\_data = \begin{cases} x \in dataset & \text{if } f_i(x) > \beta, \\ x \notin dataset & \text{otherwise} \end{cases} \tag{6.3}$$
$$i \in gamma, other$$

Nuestra modificación se da en el contexto de CTA donde los modelos son entrenados con datos simulados, y luego son expuestos a datos reales. Aprovechar estos datos reales podría ayudar a mejorar el rendimiento en clasificación de la red. Por esta razón nosotros separamos la pérdida supervisada de la no supervisada en dos etapas correspondientes a los dos términos de la ecuación 6.1. Primero entrenamos con datos etiquetados de manera

supervisada y luego entrenamos con datos no etiquetados de manera no supervisada (*self-supervised*). Para esto, además, agregamos un hyper-parámetro de umbral de incertidumbre. Modificamos el algoritmo original, agregando lo mostrado en la ecuación 6.3, donde β corresponde al umbral de incertidumbre. Lo que hace este parámetro es quedarse con los ejemplos donde la red supera el umbral, esto evitaría *confundir* a la red, al riesgo de que pueda generar overfitting. Si bien es cierto que, codificados en aquellos ejemplos clasificados con poca certidumbre, está la información más valiosa de la cual la red podría aprender. Pero se corre el riesgo de pasarle a la red un ejemplo mal clasificado, confundiendo la representación aprendida.

En resumen, nuestra modificación permitiría lo siguiente: primero, entrenar el modelo de manera supervisada con datos simulados. Y luego, una vez hayan suficientes datos, entrenar de manera no supervisada, con datos reales no-etiquetados. Donde las etiquetas son generadas con las mismas predicciones de la red, que superen el umbral β, como si fueran las etiquetas reales.

Chapter 7

Experimentos

7.1 La propuesta

Nuestra propuesta, como fue explicada también en los capítulos anteriores, consiste en proponer una metodología de entrenamiento semi-supervisado que tiene potencial para ser aprovechada con datos reales. Esta propuesta consiste en lo siguiente:

Primero entrenar un modelo con datos etiquetados, luego someter este modelo a un *fine-tuning* con datos sin etiqueta, donde la etiqueta es generada por el mismo modelo como si fuera la etiqueta real, siempre y cuando la predicción sea mayor al umbral de incertidumbre β.

Para probar nuestro método, primero se tiene que validar que las modificaciones que hicimos al algoritmo de Pseudo-labeling, detallados en el capitulo 6.2 no afectaron su capacidad de mejorar el modelo. Esto se prueba en el *Experimento de Validación*. Luego, una vez está validado, se puede aplicar el método de Pseudo-Labeling modificado en el contexto siguiente, con datos de dos calidades distintas. Lo anteriormente descrito se realiza en *Experimento de Aplicación*.

Para poder llevar a cabo ambos experimentos se dividió el dataset en diferentes subsecciones para los experimentos. Un resumen de cada uno de los experimentos y sus datos se muestra en la tabla 7.1.

Table 7.1: Resumen de los datos usados en los Experimentos

Experiments	**Training: #events**	**Validation**	**Testing**
Validation	supervised: 74945[a] self-supervised: 541911[a]	7898[a]	234276[a]
Application	supervised: 541911[a] self-supervised: 1519741[b]	108945[a]	605424[b]

[a] greater than 1000 phe, [b] greater than 300 phe

7.2 Experimento de Validación

Este experimento se diseñó para validar nuestra modificación de la técnica de Pseudo-labeling. El esquema comienza con seleccionar solo datos de alta intensidad (>1000phe). Estos se dividen en 2 sub-sets uno pequeño(10%), y uno más grande (90%). Y luego se utilizan de la siguiente manera:

- Primero se entrena un modelo de manera supervisada con el dataset pequeño (10%), hasta que logre una representación de los datos que le permita clasificar, pero que, sin embargo, no haya llegado al *plateau* en el proceso de entrenamiento.
- Luego, el modelo se sigue entrenando, ahora de manera no-supervisada con el sub-set más grande (90%) con los datos que superen el umbral de incertidumbre. Y se entrena hasta que la red alcance el *plateau*.

El fondo de este experimento es validar la modificación al primero entrenar la red para que alcance un mínimo de representación de los datos en el espacio latente para que cumpla con los dos supuestos del pseudo-labeling. Y luego ver si sigue aprendiendo información a partir de datos etiquetados por la misma red, ampliando su representación de los datos en el espacio latente. Si el modelo sigue aprendiendo, entonces la modificación quedaría validada y se podría seguir con el siguiente experimento. Adicionalmente, durante este experimento podemos encontrar el valor adecuado del umbral de incertidumbre.

7.3 Experimento de Aplicación

Una vez validado las modificaciones del pseudo-labeling, este nuevo experimento busca aplicar esta metodología al contexto de CTA. Para esto, utilizaremos dos tipos diferentes de datos: los de alta intensidad (>1000 phe), y los de mediana intensidad (>300 phe).

- Primero se entrena con los datos de alta intensidad de manera supervisada hasta alcanzar un *plateau*.
- Luego se entrena de manera no-supervisada con los datos de mediana intensidad que superen el umbral de incertidumbre. Hasta que la red alcance un *plateau*.

El objetivo de este experimento es ver si el modelo puede aprender características nuevas con algún grado de significancia estadística en las métricas de clasificación medidas, con datos que nunca ha visto la red y que tienen algún parámetro de diferencia, en este caso el parámetro de *Hillas intensity*. Al disminuir el parámetro de intensidad de Hillas, cambia indirectamente la morfología de la imagen, o el lugar en el que se captó el evento en los sensores del telescopio.

Si mejora el rendimiento de la red, eso quiere decir que nuestra metodología logró extraer información nueva a partir de datos no etiquetados y por lo tanto es posible mejorar la representación de los datos en el espacio latente.

Chapter 8

Resultados

En este capitulo se mostrarán los resultados de los experimentos detallados en el capitulo 7.

Para todos los experimentos *recall* y *precision* son focalizados en gammas. Por otro lado, para poder tener una visión global de nuestra implementación, adicionalmente se evaluó un modelo de control, donde se entrenó de manera supervisada sobre el dataset total ($>$300 phe) (escenario de información completa). Estos resultados ayudan a contrastar la magnitud de nuestros resultados de nuestra implementación de nuestra propuesta.

Adicionalmente, para confirmar que nuestros resultados son significativos, se procedió a hacer test de hipótesis. El test que se uso es el test de McNemar [33] el cual es un test que se usa sobre datos que son dicotómicos. Se evalúa usando una matriz de contingencia, la cual es ampliamente usado en aplicaciones médicas para evaluar herramientas de diagnóstico. Para ambos experimentos, se puede rechazar la hipótesis nula con un $\rho << 0.5$.

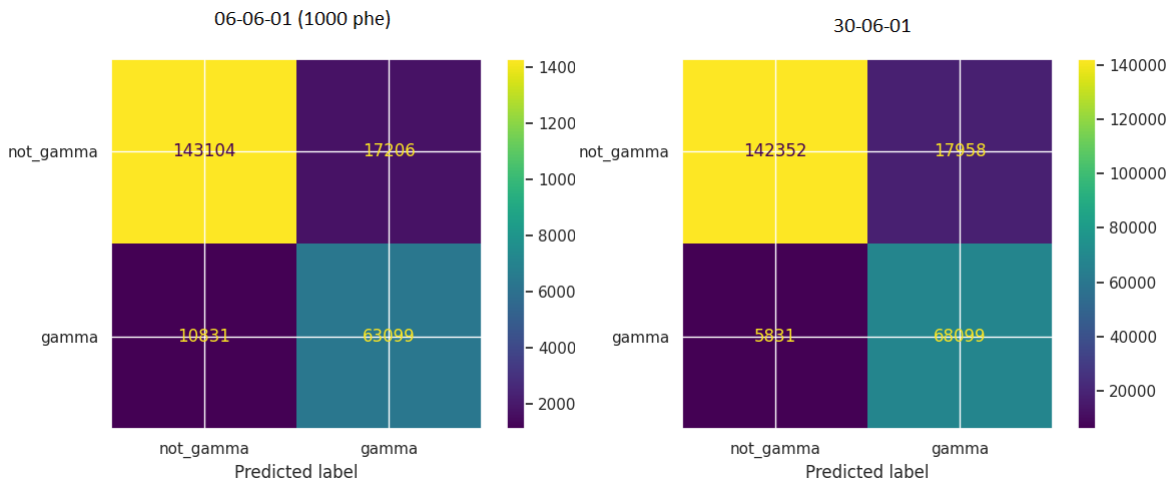
8.1 Experimento de Validación

En el experimento de validación el modelo fue entrenado usando *early stopping* hasta que el modelo encuentra un *plateau* en la función de pérdida. Debido a que se usó CLR, la red converge más rápido que los métodos convencionales.

Para el entrenamiento supervisado se entrenó durante 23 épocas, sin embargo el mínimo en la función de pérdida se encontró en la época 17. Los resultados para el experimento de validación evaluado en el dataset de testing se muestran en la tabla 8.1.

Para el entrenamiento no-supervisado, todo el entrenamiento se hizo en solo una época, para evitar overfitting. Debido a que se aplicó un umbral de incertidumbre, la red podría fácilmente tener overfitting con estos ejemplos que la red ya clasifica con alta certidumbre.

El parámetro β, descrito en la ecuación 6.3, se encontró con el método de *grid search*. La capacidad de aprendizaje del modelo, parece ser bastante sensible a este parámetro, pero el mejor rendimiento alcanzado se obtuvo con un valor de $\beta = 0.9$, el cual se aplicó para



(a) Matriz de confusión del modelo entrenado de manera supervisada

(b) Matriz de confusión del modelo entrenado de manera no-supervisada

Figure 8.1: Matriz de confusión de las distintas etapas del experimento de validación

ambos experimentos.

Como se muestra en la tabla 8.1, el entrenamiento no-supervisado mejoró el rendimiento del modelo en todas las métricas. La figura 8.1 muestra la matriz de confusión de ambas etapas, sobre el mismo dataset de testing. De esta figura se puede apreciar que la mejora en la clasificación correcta de eventos gamma, viene con un leve disminución de los eventos bien clasificados como no-gamma. Sin embargo, la prioridad de la tarea es detectar gammas. Adicionalmente, el resto de las métricas también mejoran, por lo que hoy una mejora en la clasificación. Nótese además como hay una migración de los eventos mal clasificados como no-gamma hacia una buena clasificación, producto del entrenamiento no supervisado.

En otras palabras, para la misma intensidad de Hillas, el modelo aprendió una mejor representación de los datos usando las etiquetas predichas por el mismo clasificador, como si fueran las etiquetas reales. De esto se concluye que la estrategia semi-supervisada general es un buen candidato para seguir estudiando como imágenes con diferente calidad afectarán el rendimiento de la red.

Table 8.1: Resultados del experimento de validación

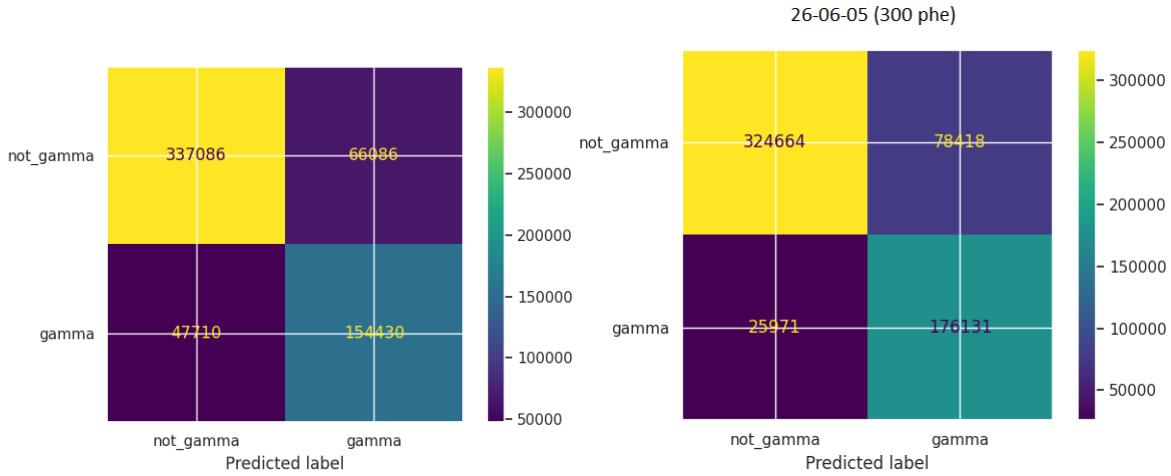
Validation Experiment	Accuracy	Recall	Precision	f1-score
Supervised	0.88	0.85	0.82	0.82
Unsupervised	0.90	0.92	0.85	0.85
Control (1000 phe)	0.91	0.90	0.83	0.87

Por último, nótese que los resultados obtenidos con el esquema semi-supervisado superan en *Recall* y *Precision* al experimento de control en la tabla 8.1. Mientras en *Accuracy* se encuentra bastante cercano, a sólo un 1% del control. En tanto el f1-score,

se encuentra a medio camino. Esto en el fondo nos permite ver, que nuestra metodología semi-supervisada es una posible opción para aprender de datos sin etiqueta, sin un gran sacrificio en el aprendizaje.

8.2 Experimento de Aplicación

Para el experimento de aplicación se puede apreciar de la tabla 8.2 que todas las métricas mejoraron, excepto por la precisión, la cual cayó un 1%. Sin embargo, el aumento en *recall* y *accuracy* nos dicen que la mejora en identificar de manera correcta los eventos gamma viene con el precio de un leve aumento del error tipo-I.



(a) Matriz de confusión del modelo entrenado de manera supervisada

(b) Matriz de confusión del modelo entrenado de manera no-supervisada

Figure 8.2: Matriz de confusión de las distintas etapas del experimento de aplicación

Table 8.2: Resultados del Experimento de Aplicación

Application Experiment	Accuracy	Recall	Precision	f1-score
Supervised	0.81	0.76	0.70	0.73
Unsupervised	0.83	0.87	0.69	0.77
Control (300 phe)	0.85	0.89	0.73	0.80

De los resultados de control de la Tabla 8.2 se aprecia que los resultados de la técnica semi-supervisada se encuentran en un término medio, excepto en precisión. Por lo que ayuda a reforzar la conclusión anterior.

Chapter 9

Espacio Latente

En este capitulo se abordará nuestro breve estudio complementario del espacio latente. Esto en el contexto de CTA, donde se cuenta con una parametrización de las imágenes y se cuenta con un amplio y documentado estudio de sus diferencias morfológicas entre partículas.

El espacio latente hace referencia a la representación, generalmente de hyperdimensional, del espacio de eventos de entrenamiento. Este espacio, aunque hyperdimensional, suele ser de una dimensión menor que la original. Y debido a que esta reducción dimensional se produce con el entrenamiento de una tarea especifica, ésta representación del espacio latente, es considerada como las *features* (o características) que la red aprende para el cumplimiento de la tarea.

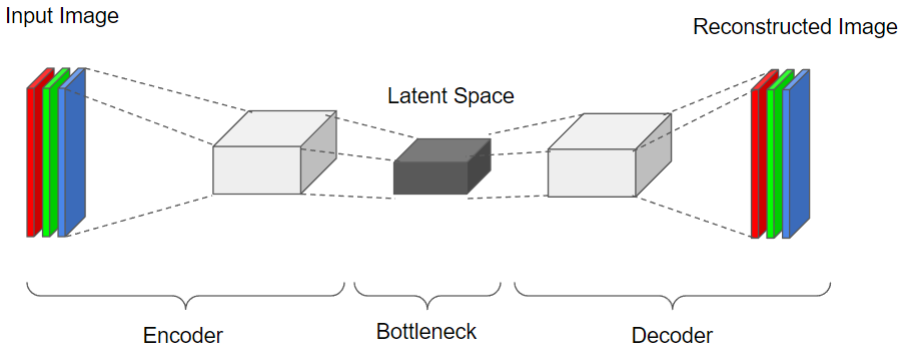


Figure 9.1: ubicación del Espacio Latente, en una red tipo autoencoder (Elaboración propia)

En las redes neuronales convolucionales, que fueron originalmente inspiradas por las neuronas de la retina, al igual que éstas, sus primeras neuronas detectan características muy generales, como bordes, cantos, cambios de colores, luego en las capas posteriores se comienzan a combinar estas características para formar otras de mayor complejidad, como formas particulares, formas con colores, etc. Luego, dependiendo del tipo de red -clasificadora, regresora, autoencoder-, viene la capa lógica o el cuello de botella como se

aprecia en la Figura 9.1, que es el punto más profundo de la red donde habitan las características de representación más complejas de la red, las cuales usa para el cumplimiento de su tarea. Este espacio de representación es comúnmente denominado Espacio Latente. El estudio de éste permite entender lo que la CNN usa para cumplir la tarea.

9.1 Nuestro diseño

Una vez que la red está entrenada, nos propusimos investigar el espacio latente. Para lograr este objetivo, primero se tiene que usar una CNN entrenada. Luego, definimos en que capas se extraerían las activaciones de las neuronas. Posteriormente hay que definir que estrategia de reducción de dimensionalidad usar, debido a que siguen siendo hyperdimensionales.

Siguiendo esta linea lógica, primero se usó la CNN entrenada con todos los datos de manera supervisada (modelo de control), luego se decidió usar la capa *flatten* de la red, debido a que ésta solo contiene los features aprendidos por las redes convolucionales, y aun no aplica los circuitos lógicos para el problema de clasificación. Finalmente para la reducción de dimensionalidad se utiliza PCA, debido a su capacidad de jerarquizar las dimensiones por la varianza que aportan, sin embargo exploramos otras técnicas también.

9.2 Reducción de dimensionalidad

Para las técnicas de reducción de dimensionalidad se usaron tres técnicas: PCA, tsne [34], umap [35]. En sus modalidades supervisadas (tsne, y umap) y no-supervisadas (pca, tsne, umap). Esto se aplicó sobre todas las capas lógicas e.g *flatten*, 128, 128, 64. Las imagenes se pueden apreciar en el Apéndice A.

Nótese en las imágenes que las estrategias supervisadas presentan una aparente mejor separación de clases, pero ésto es debido a que son sesgadas al tener información acerca de las clases de cada evento. Sin embargo las imágenes no supervisadas muestran agrupaciones que confirman ambas suposiciones de *Cluster assumption* y *Continuity assumption* en cualquiera de los gráficos.

9.3 Matriz de correlación

Una segunda exploración que se hizo fue tomar ciertas componentes y correlacionarlas con los parametros de Hillas. El objetivo de esto es validar la red, al ver si tiene correlaciones con los parámetros de Hillas que sabemos que tienen relación para el trabajo de clasificación (Hillas length y Hillas width) debido a la amplia literatura en el tema.

Para lograr esto, primero se extrajeron las activaciones de la capa flatten. Luego, debido a su gran número de dimensiones (15360 activaciones) se aplicó una estrategia de reducción de dimensionalidad, se eligió PCA, debido a que mantiene la representación de

Table 9.1: Varianza explicada para las distintas capas

70% threshold	**512 samples**	
Layer	**PCA Components**	**Exp. Variance**
Flatten	79	70,13%
Dense 1	9	71,56%
Dense 2	7	70,56%
Dense 3	5	70,16%

espacio métrico. Luego, se eligió cierto numero de componentes que superara un umbral de varianza explicada (70%). De la tabla 9.1, se puede apreciar que la primera capa densa explica el 70% de la varianza con solo nueve componentes, debido a esto, se eligió esa capa. Se priorizó el uso de una capa superior debido a que se quieren observar las características aprendidas de la red, con la menor influencia posible de las capas lógicas. Finalmente con esas componentes se arma la matriz de correlación con los parámetros de Hillas. Los datos con los que se hizo la matriz de correlación pueden variar el resultado. Por esto, se crearon gráficos con protones, gamma y ambos. Los tres gráficos se muestran en el apéndice B.

De las figuras B.1 se puede apreciar que la correlación (o anti-correlación) más relevantes se acumulan en el parámetro de hillas_width y hillas_length. Esto confirmaría nuestra hipótesis inicial, donde contamos con el conocimiento previo de las metodologías convencionales que usan cortes precisamente en estos parámetros de Hillas para hacer un clasificador. Otra forma de interpretar este resultado es descartar aquellos parámetros que no se relacionan con el problema de clasificación. Los cuales, de mostrar relación, indicarían un comportamiento patológico.

Estos comportamientos patológicos pueden ser por ejemplo, que la CNN aprendiera a clasificar los eventos basados por ejemplo en un sesgo de la posición, lo cual en realidad no debería ser la metodología para clasificar, ya que la posición del evento no tiene relación con su partícula de origen.

Complementariamente incluimos en este trabajo, las imágenes de las correlaciones obtenidas solo a partir de imágenes de una misma clase en las Figuras B.2 y B.3. Donde podemos encontrar que para identificar los gamma hay una alta correlación con hillas_intensity, hillas_length_uncertainty, hillas_width_uncertainty, y hillas_y.

La explicación con el hillas_length_uncertainty, hillas_width_uncertainty, es entendible con el mismo razonamiento anterior. Debido a que la forma de la *extensive shower* producidas por rayos gamma es más uniforme que aquellas producidas por protones.

Los parámetro de hillas_intensity y hillas_y, tienen que analizarse con mayor detenimiento en un estudio posterior. La intensidad de Hillas, puede ser un sesgo que se agregó al hacer los cortes de calidad en este mismo parámetro. Podría deberse también a las diferencias en las distribuciones en intensidad de los datos, que produjeron este artefacto.

El hillas_y es el parámetro que más correlación tiene, lo cual es interesante, debido a que el clasificador está asignando importancia a distinguir la clase a partir de la posición, lo cual podría revelar fallas en la simulación y éste conocimiento podría utilizarse para lograr mejores simulaciones.

Chapter 10

Recursos

10.1 Wilkes 3 HPC

Para este trabajo se hizo uso de el *High Performance Computer (HPC)* Wilkes 3 de la universidad de Oxford. Este cuenta con 80 nodos basados en el Dell PowerEdge XE8545. Donde cada nodo contiene dos procesadores AMD EPYC 7763 64-Core, 1000 Gb de RAM, Dual Rail Mellanox HDR200 InfiniBand y además cuenta con cuatro Nvidia A100 SXM4 de 80 Gb, y se conecta a los 5PB/s disponibles a traves de LustreFS compartidos a través de Peta4.

Una de las dificultades que se tuvo en el desarrollo de este trabajo fue la correcta implementación de Gerumo en la plataforma de Wilkes 3. La correcta instalación de los paquetes y las restricciones del HPC dificultaron la implementación. Finalmente se optó por utilizar Dockers en Sigularity, para lo que se tuvo que aprender todo lo relativo a esta implementación.

Adicionalmente, para trabajar en Wilkes 3 se debe usar solicitudes de recurso con Slurm [36]. Esto también fue un motivo de demora debido a que se tuvo que aprender el uso de Slurm y estimar la cantidad necesaria de tiempo y recursos que se tenía que usar para las distintas tareas.

Considerando todo lo anterior, cada entrenamiento tomaba aproximadamente 20 hrs. Debido a esto se tuvo que restringir la cantidad de experimentos. Además, las cantidad de horas por grupo y por usuario son limitadas.

Chapter 11

Conclusiones

En el trabajo acá mostrado, se diseñó una estrategia semi-supervisada, basada en Pseudo-labeling, que permite el entrenamiento separado, secuencial de dos etapas, una supervisada y otra no-supervisada (self-supervised). Se diseñaron dos experimentos, el primero donde se validaban las modificaciones a la estrategia de Pseudo label. Y el segundo donde se aplicó la estrategia a imágenes nuevas, en el contexto de CTA. Ambos experimentos dieron resultados exitosos donde el modelo mejoró en todas sus métricas para el caso del experimento de validación y en todas las métricas excepto precisión (bajó 1%), en el caso de el experimento de Aplicación.

Además se hizo una exploración del espacio latente, en donde se tomaron las activaciones de capas intermedias y donde, con la ayuda de técnicas de reducción de dimensionalidad, se procede a explorar el espacio latente. Esta exploración nos permitió confirmar las suposiciones hechas por el Pseudo-labeling, *Continuity* y *cluster*.

Adicionalmente exploramos la relación entre las características aprendidas por la red y los parámetros de Hillas. Para obtener las características de la red, se hizo un PCA de 9 componentes en la capa Flatten. Luego lo correlacionamos con los parámetros de Hillas. Esto nos confirmó que el clasificador usa características altamente correlacionadas con los parámetros de Hillas usados en las metodologías convencionales, validando la CNN. Además, se puede utilizar para explorar y validar redes convolucionales en el contexto de CTA, debido a que pueden revelar problemas en las metodologías de entrenamiento o bien en la simulación.

Nuestra estrategia probó ser una metodología que permite aprender desde datos no etiquetados. Esto se podría utilizar en la integración de datos reales, donde el modelo implementado podría seguir aumentando su rendimiento usando las imágenes reales no etiquetadas para aprender sutilezas que ayuden a mejorar sus métricas de rendimiento. La forma de implementación de datos reales aun no esta clara, debido a que la forma en que se implementen los datos en nuestro algoritmo va a depender de las naturalezas de las diferencias entre las imágenes.

Por ejemplo se podrían implementar imágenes reales y simuladas juntas en la parte no

supervisada y que ambas sean de un corte en intensidad alto. Con ésto, la red aprendería algunas cualidades que no tienen las imágenes simuladas y que serían fáciles de distinguir debido a el alto corte en la calidad. Para luego ir bajando los cortes de calidad para aprender de todo el rango de energía de los datos.

En la medida de que vayan liberándose datos de CTAO se podrá hacer distintos experimentos para poder definir la mejor forma de integración de los datos.

Este trabajo motiva novedosas lineas de investigación para CTA. Por un lado, se presentó una estrategia para investigar las diferencias entre los datos reales y los simulados, mirando el espacio latente del clasificador, con el objetivo de lograr mejores simulaciones. Motiva además al desarrollo de técnicas que aborden el uso e implementación de datos reales en la estrategia de clasificación.

Bibliography

[1] T. Bringmann and C. Weniger, "Gamma ray signals from dark matter: Concepts, status and prospects," en, *Physics of the Dark Universe*, vol. 1, no. 1-2, pp. 194–217, Nov. 2012, ISSN: 22126864. DOI: 10.1016/j.dark.2012.10.005. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S221268641200009X (visited on 07/26/2022).

[2] L. Bergström, T. Bringmann, M. Eriksson, and M. Gustafsson, "Gamma Rays from Heavy Neutralino Dark Matter," en, *Physical Review Letters*, vol. 95, no. 24, p. 241 301, Dec. 2005, ISSN: 0031-9007, 1079-7114. DOI: 10.1103/PhysRevLett.95.241301. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.95.241301 (visited on 07/26/2022).

[3] L. Bergström, T. Bringmann, M. Eriksson, and M. Gustafsson, "Gamma Rays from Kaluza-Klein Dark Matter," en, *Physical Review Letters*, vol. 94, no. 13, p. 131 301, Apr. 2005, ISSN: 0031-9007, 1079-7114. DOI: 10.1103/PhysRevLett.94.131301. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.94.131301 (visited on 07/26/2022).

[4] C. Bœhm, M. J. Dolan, C. McCabe, M. Spannowsky, and C. J. Wallace, "Extended gamma-ray emission from coy dark matter," *Journal of Cosmology and Astroparticle Physics*, vol. 2014, no. 05, pp. 009–009, May 2014, Publisher: IOP Publishing. DOI: 10.1088/1475-7516/2014/05/009. [Online]. Available: https://doi.org/10.1088/1475-7516/2014/05/009.

[5] T. Daylan, D. P. Finkbeiner, D. Hooper, *et al.*, "The characterization of the gamma-ray signal from the central Milky Way: A case for annihilating dark matter," en, *Physics of the Dark Universe*, vol. 12, pp. 1–23, Jun. 2016, ISSN: 22126864. DOI: 10.1016/j.dark.2015.12.005. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S2212686416000030 (visited on 07/26/2022).

[6] The CTA Consortium, *Science with the Cherenkov Telescope Array*, en. WORLD SCIENTIFIC, Mar. 2019, ISBN: 978-981-327-008-4 978-981-327-009-1. DOI: 10.1142/10986. [Online]. Available: https://www.worldscientific.com/worldscibooks/10.1142/10986 (visited on 07/17/2022).

[7] M. Krause, E. Pueschel, and G. Maier, "Improved γ/hadron separation for the detection of faint gamma-ray sources using boosted decision trees," *Astroparticle Physics*, vol. 89, pp. 1–9, Mar. 2017, arXiv:1701.06928 [astro-ph], ISSN: 09276505.

DOI: 10.1016/j.astropartphys.2017.01.004. [Online]. Available: http://arxiv.org/abs/1701.06928 (visited on 07/17/2022).

[8] J. Albert, "Implementation of the Random Forest Method for the Imaging Atmospheric Cherenkov Telescope MAGIC," 2007, Publisher: arXiv Version Number: 2. DOI: 10.48550/ARXIV.0709.3719. [Online]. Available: https://arxiv.org/abs/0709.3719 (visited on 07/19/2022).

[9] S. Ohm, C. van Eldik, and K. Egberts, "Gamma-Hadron Separation in Very-High-Energy gamma-ray astronomy using a multivariate analysis method," 2009, Publisher: arXiv Version Number: 1. DOI: 10.48550/ARXIV.0904.1136. [Online]. Available: https://arxiv.org/abs/0904.1136 (visited on 07/26/2022).

[10] Y. Becherini, A. Djannati-Ataï, V. Marandon, M. Punch, and S. Pita, "A new analysis strategy for detection of faint gamma-ray sources with Imaging Atmospheric Cherenkov Telescopes," 2011, Publisher: arXiv Version Number: 1. DOI: 10.48550/ARXIV.1104.5359. [Online]. Available: https://arxiv.org/abs/1104.5359 (visited on 07/26/2022).

[11] D. Heck, J. Knapp, J. N. Capdevielle, G. Schatz, and T. Thouw, *CORSIKA: a Monte Carlo code to simulate extensive air showers.* Feb. 1998, Publication Title: CORSIKA: a Monte Carlo code to simulate extensive air showers ADS Bibcode: 1998cmcc.book.....H. [Online]. Available: https://ui.adsabs.harvard.edu/abs/1998cmcc.book.....H (visited on 07/26/2022).

[12] K. Bernlohr, "Simulation of Imaging Atmospheric Cherenkov Telescopes with CORSIKA and sim_telarray," 2008, Publisher: arXiv Version Number: 1. DOI: 10.48550/ARXIV.0808.2253. [Online]. Available: https://arxiv.org/abs/0808.2253 (visited on 07/26/2022).

[13] I. Shilon, M. Kraus, M. Büchele, *et al.*, "Application of deep learning methods to analysis of imaging atmospheric Cherenkov telescopes data," en, *Astroparticle Physics*, vol. 105, pp. 44–53, Feb. 2019, ISSN: 09276505. DOI: 10.1016/j.astropartphys.2018.10.003. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0927650518301178 (visited on 07/17/2022).

[14] D. Nieto, A. Brill, B. Kim, and T. B. Humensky, *Exploring deep learning as an event classification method for the Cherenkov Telescope Array*, arXiv:1709.05889 [astro-ph], Sep. 2017. [Online]. Available: http://arxiv.org/abs/1709.05889 (visited on 07/19/2022).

[15] D. Nieto, A. Brill, Q. Feng, *et al.*, *CTLearn: Deep Learning for Gamma-ray Astronomy*, arXiv:1912.09877 [astro-ph], Dec. 2019. [Online]. Available: http://arxiv.org/abs/1912.09877 (visited on 07/19/2022).

[16] T. Miener, D. Nieto, A. Brill, S. Spencer, and J. L. Contreras, *Reconstruction of stereoscopic CTA events using deep learning with CTLearn*, arXiv:2109.05809 [astro-ph], Sep. 2021. [Online]. Available: http://arxiv.org/abs/2109.05809 (visited on 07/19/2022).

[17] P. Grespan, M. Jacquemont, R. López-Coto, T. Miener, D. Nieto-Castaño, and T. Vuillaume, *Deep-learning-driven event reconstruction applied to simulated data from a single Large-Sized Telescope of CTA*, arXiv:2109.14262 [astro-ph], Sep. 2021. [Online]. Available: http://arxiv.org/abs/2109.14262 (visited on 07/17/2022).

[18] A. M. Hillas, "Cerenkov light images of EAS produced by primary gamma," en, vol. 3, no. OG-9.5-3, p. 4, 1985.

[19] A. M. Hillas, "Differences Between Gamma-Ray and Hadronic Showers," en, in *TeV Gamma-Ray Astrophysics*, H. J. Völk and F. A. Aharonian, Eds., Dordrecht: Springer Netherlands, 1996, pp. 17–30, ISBN: 978-94-010-6561-0 978-94-009-0171-1. DOI: 10.1007/978-94-009-0171-1_2. [Online]. Available: http://link.springer.com/10.1007/978-94-009-0171-1_2 (visited on 08/23/2022).

[20] J. Juryšek, E. Lyard, and R. Walter, *Full LST-1 data reconstruction with the use of convolutional neural networks*, arXiv:2111.14478 [astro-ph], Nov. 2021. [Online]. Available: http://arxiv.org/abs/2111.14478 (visited on 08/20/2022).

[21] T. Miener, R. López-Coto, J. L. Contreras, *et al.*, *IACT event analysis with the MAGIC telescopes using deep convolutional neural networks with CTLearn*, arXiv:2112.01828 [astro-ph], Dec. 2021. [Online]. Available: http://arxiv.org/abs/2112.01828 (visited on 08/20/2022).

[22] T. Vuillaume, M. Jacquemont, M. d. B. de Lavergne, *et al.*, *Analysis of the Cherenkov Telescope Array first Large-Sized Telescope real data using convolutional neural networks*, arXiv:2108.04130 [astro-ph], Aug. 2021. [Online]. Available: http://arxiv.org/abs/2108.04130 (visited on 08/20/2022).

[23] C.-C. Lu, *Improving the H.E.S.S. angular resolution using the Disp method*, arXiv:1310.1200 [astro-ph], Oct. 2013. [Online]. Available: http://arxiv.org/abs/1310.1200 (visited on 07/19/2022).

[24] K. Bernlöhr, A. Barnacka, Y. Becherini, *et al.*, "Monte Carlo design studies for the Cherenkov Telescope Array," en, *Astroparticle Physics*, vol. 43, pp. 171–188, Mar. 2013, ISSN: 09276505. DOI: 10.1016/j.astropartphys.2012.10.002. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0927650512001867 (visited on 08/25/2022).

[25] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain.," en, *Psychological Review*, vol. 65, no. 6, pp. 386–408, 1958, ISSN: 1939-1471, 0033-295X. DOI: 10.1037/h0042519. [Online]. Available: http://doi.apa.org/getdoi.cfm?doi=10.1037/h0042519 (visited on 09/01/2022).

[26] F. Rosenblatt, "Perceptron Simulation Experiments," *Proceedings of the IRE*, vol. 48, no. 3, pp. 301–309, Mar. 1960, ISSN: 0096-8390. DOI: 10.1109/JRPROC.1960.287598. [Online]. Available: http://ieeexplore.ieee.org/document/4066017/ (visited on 09/01/2022).

[27] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," en, *The Bulletin of Mathematical Biophysics*, vol. 5, no. 4, pp. 115–133, Dec. 1943, ISSN: 0007-4985, 1522-9602. DOI: 10.1007/BF02478259. [Online]. Available: http://link.springer.com/10.1007/BF02478259 (visited on 09/01/2022).

[28] D. H. Hubel and T. N. Wiesel, "Receptive fields of single neurones in the cat's striate cortex," en, *The Journal of Physiology*, vol. 148, no. 3, pp. 574–591, Oct. 1959, ISSN: 00223751. DOI: 10.1113/jphysiol.1959.sp006308. [Online]. Available: https://onlinelibrary.wiley.com/doi/10.1113/jphysiol.1959.sp006308 (visited on 09/01/2022).

[29] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," 2014, Publisher: arXiv Version Number: 6. DOI: 10.48550/ARXIV.1409.1556. [Online]. Available: https://arxiv.org/abs/1409.1556 (visited on 08/03/2022).

[30] M. Peresano, *Protopipe*, Apr. 2022. DOI: 10.5281/ZENODO.4586754. [Online]. Available: https://zenodo.org/record/4586754 (visited on 09/04/2022).

[31] L. N. Smith, "Cyclical Learning Rates for Training Neural Networks," in *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, Santa Rosa, CA, USA: IEEE, Mar. 2017, pp. 464–472, ISBN: 978-1-5090-4822-9. DOI: 10.1109/WACV.2017.58. [Online]. Available: http://ieeexplore.ieee.org/document/7926641/ (visited on 07/18/2022).

[32] D.-H. Lee, "Pseudo-Label : The Simple and Efficient Semi-Supervised Learning Method for Deep Neural Networks," Tech. Rep., 2013. [Online]. Available: https://www.researchgate.net/publication/280581078.

[33] N. Japkowicz and M. Shah, "Performance Evaluation in Machine Learning," en, in *Machine Learning in Radiation Oncology*, I. El Naqa, R. Li, and M. J. Murphy, Eds., Cham: Springer International Publishing, 2015, pp. 41–56, ISBN: 978-3-319-18304-6 978-3-319-18305-3. DOI: 10.1007/978-3-319-18305-3_4. [Online]. Available: http://link.springer.com/10.1007/978-3-319-18305-3_4 (visited on 12/30/2022).

[34] Laurens Van der Maaten and G. Hinton, "Visualizing data using t-SNE," vol. 9, no. 11, 2008.

[35] L. McInnes, J. Healy, and J. Melville, *UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction*, arXiv:1802.03426 [cs, stat], Sep. 2020. [Online]. Available: http://arxiv.org/abs/1802.03426 (visited on 07/18/2022).

[36] A. B. Yoo, M. A. Jette, and M. Grondona, "SLURM: Simple Linux Utility for Resource Management," in *Job Scheduling Strategies for Parallel Processing*, G. Goos, J. Hartmanis, J. van Leeuwen, D. Feitelson, L. Rudolph, and U. Schwiegelshohn, Eds., vol. 2862, Series Title: Lecture Notes in Computer Science, Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 44–60, ISBN: 978-3-540-20405-3

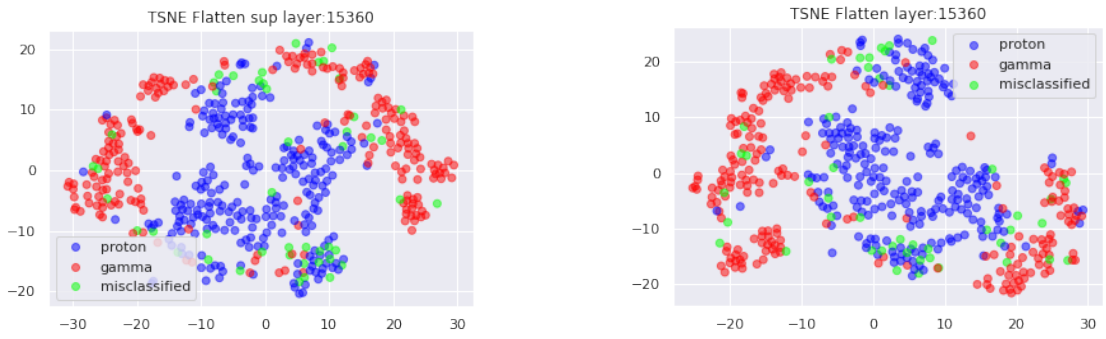
978-3-540-39727-4. DOI: 10.1007/10968987_3. [Online]. Available: http://link.springer.com/10.1007/10968987_3 (visited on 09/20/2022).

Appendix A

Espacio Latente

En este apéndice se muestra la reducción de dimensionalidad que se le aplicó a las distintas capas de la red. En sus versiones supervisadas y no-supervisadas.

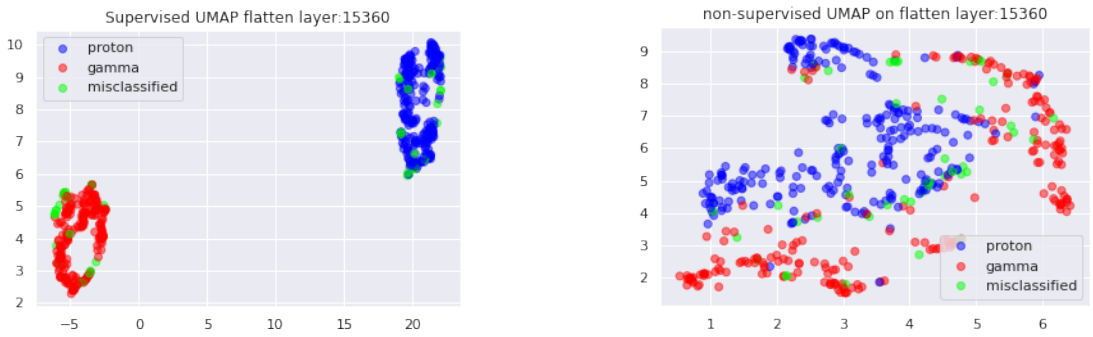
A.1 Flatten Layer



(a) tsne supervisado

(b) tsne no-supervisado

Figure A.1: tsne en la capa flatten



(a) umap supervisado (b) umap no-supervisado

Figure A.2: umap en la capa flatten

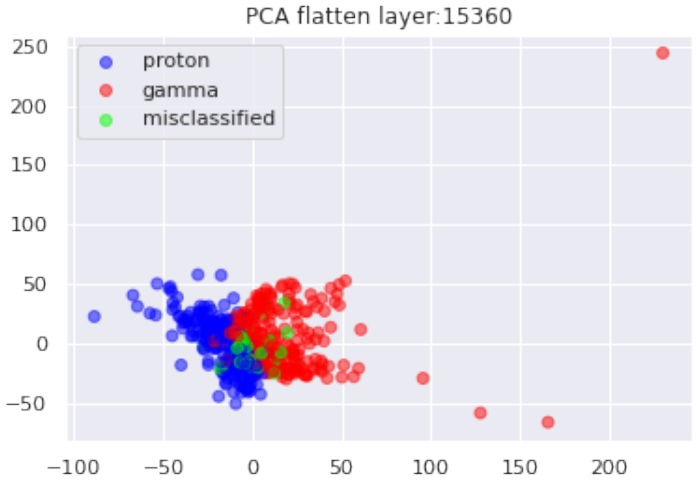
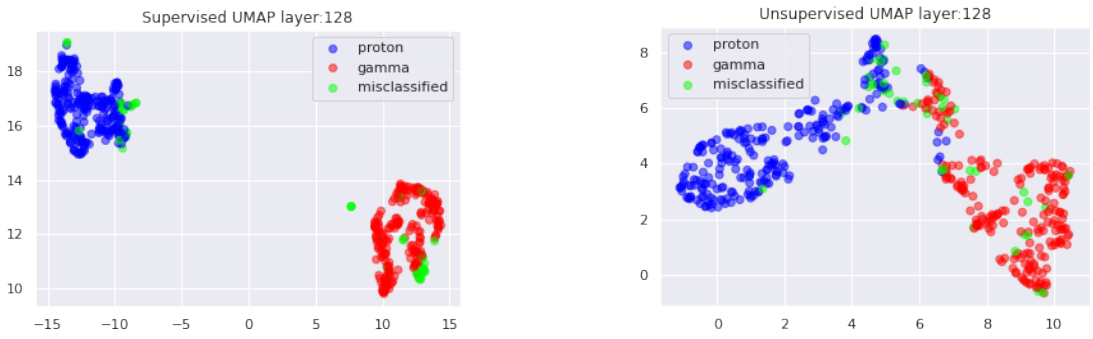


Figure A.3: PCA para la capa flatten

A.2 First Logic Layer

(a) tsne supervisado (b) tsne no-supervisado

Figure A.4: tsne en la primera capa lógica (128-1)



(a) umap supervisado (b) umap no-supervisado

Figure A.5: umap en la primera capa lógica (128-1)

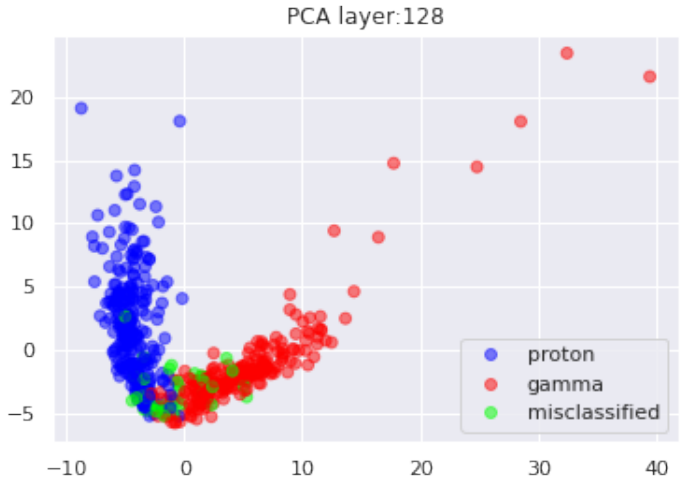
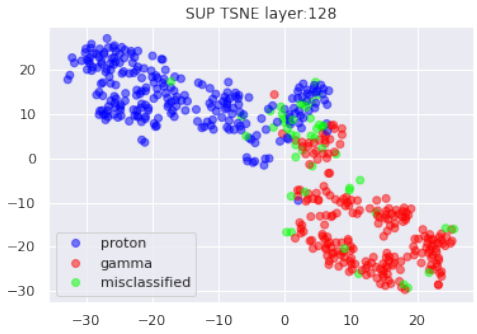
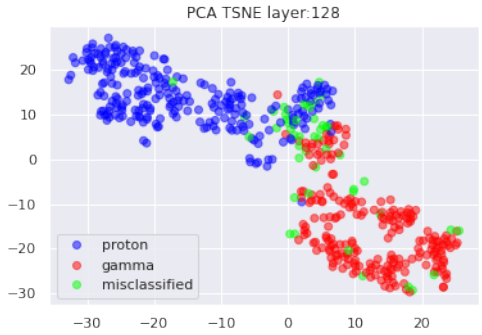


Figure A.6: PCA para la primera capa lógica

A.3 Second Logic Layer

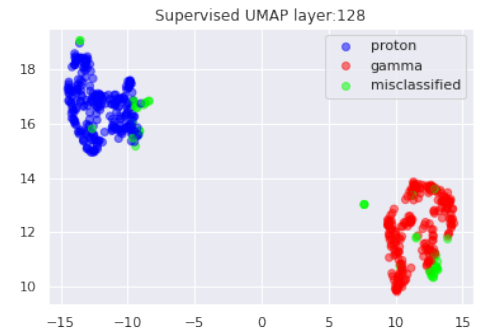


(a) tsne supervisado

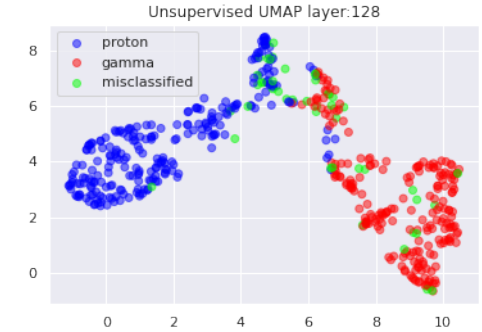


(b) tsne no-supervisado

Figure A.7: tsne en la segunda capa lógica (128-1)



(a) umap supervisado



(b) umap no-supervisado

Figure A.8: umap en la segunda capa lógica (128-1)

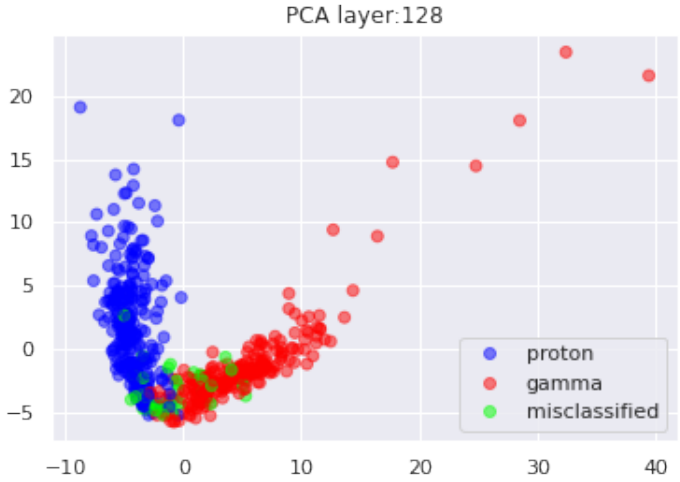
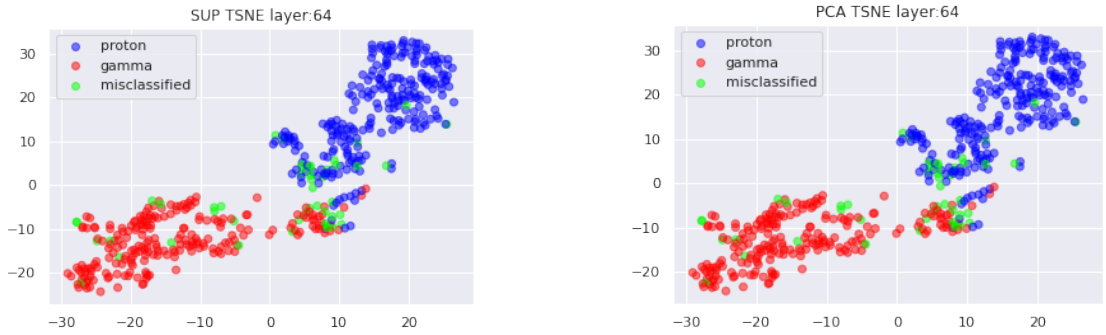


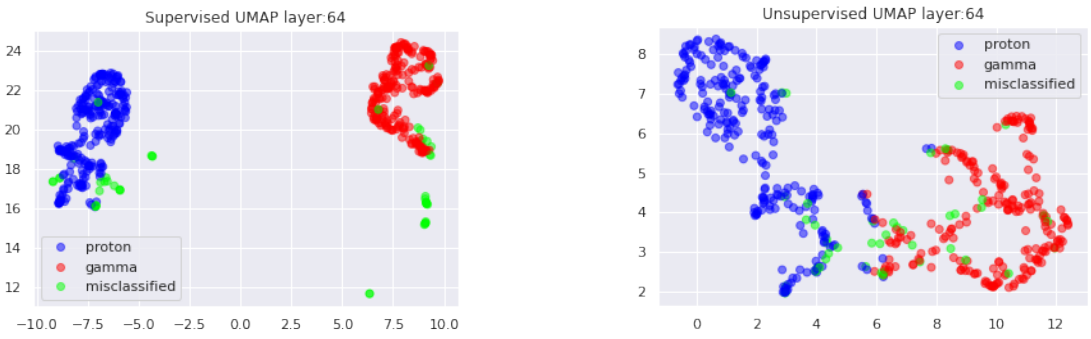
Figure A.9: PCA para la segunda capa lógica

A.4 Third Logic Layer



(a) tsne supervisado (b) tsne no-supervisado

Figure A.10: tsne en la última capa lógica (128-1)



(a) umap supervisado (b) umap no-supervisado

Figure A.11: umap en la última capa lógica (128-1)

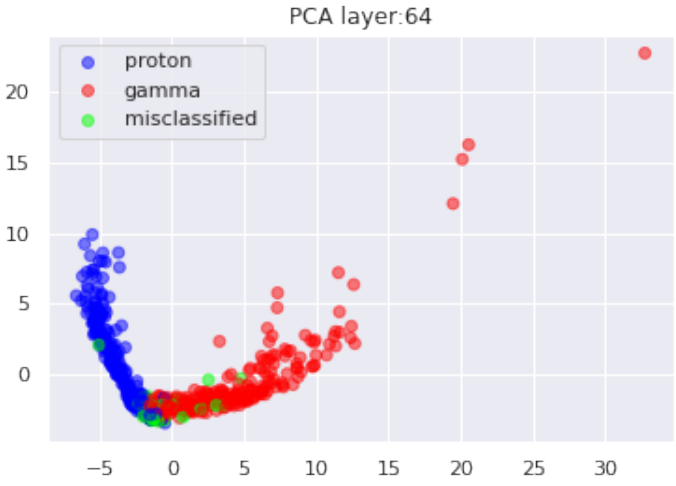


Figure A.12: PCA para la última capa lógica

Appendix B

Matriz de correlación

En este apéndice se muestran imágenes complementarias de la relación entre la representación de los datos en el espacio latente y los parámetros de Hillas.

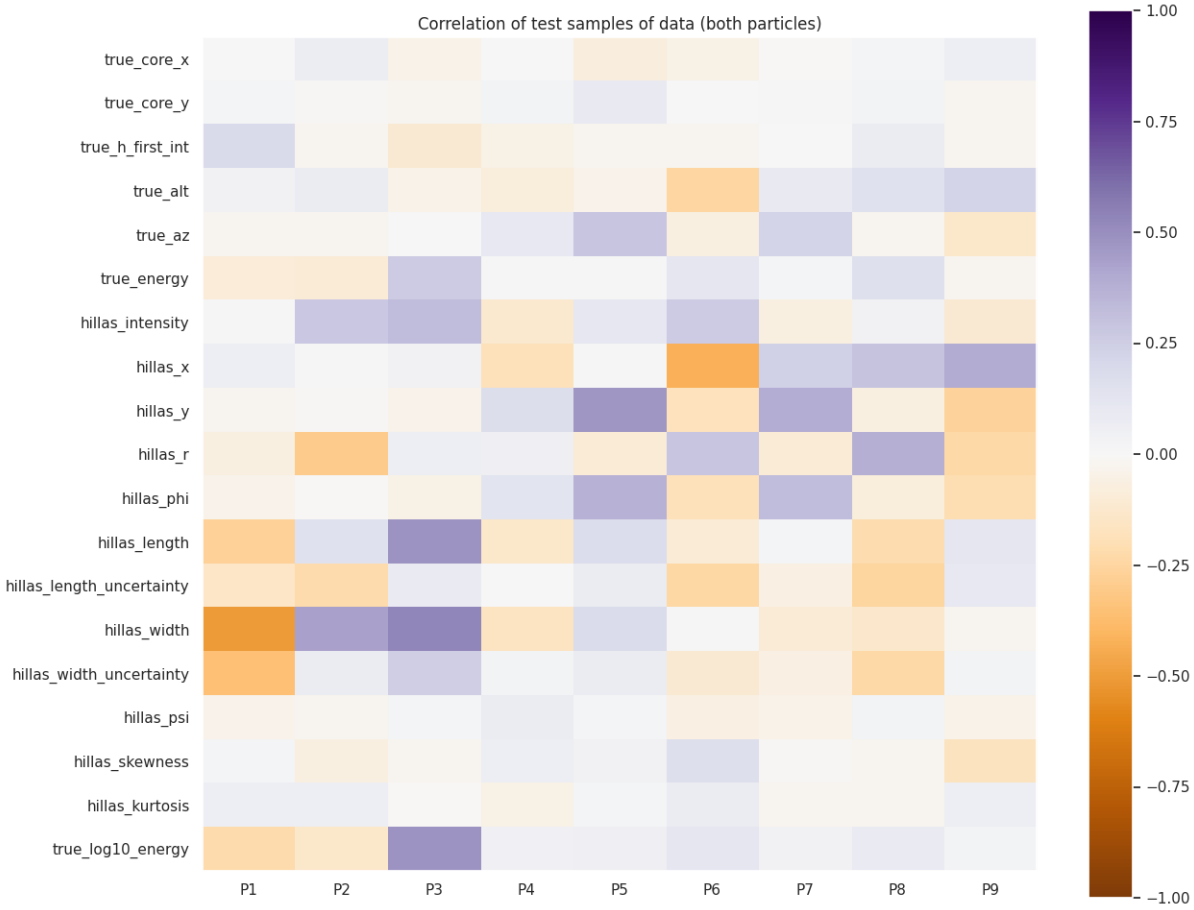


Figure B.1: Matriz de correlación entre ambas partículas y los parámetros de Hillas

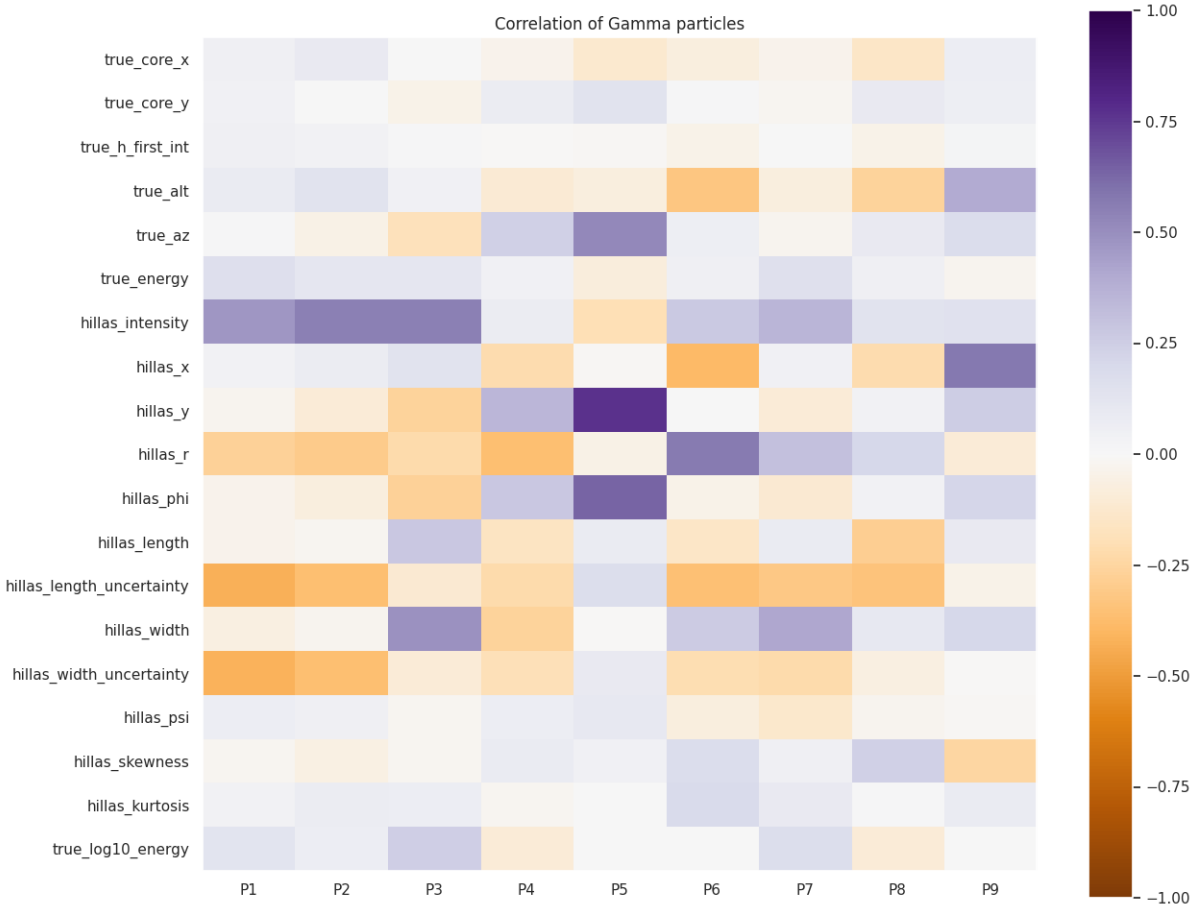


Figure B.2: Matriz de correlación entre gamma y los parámetros de Hillas

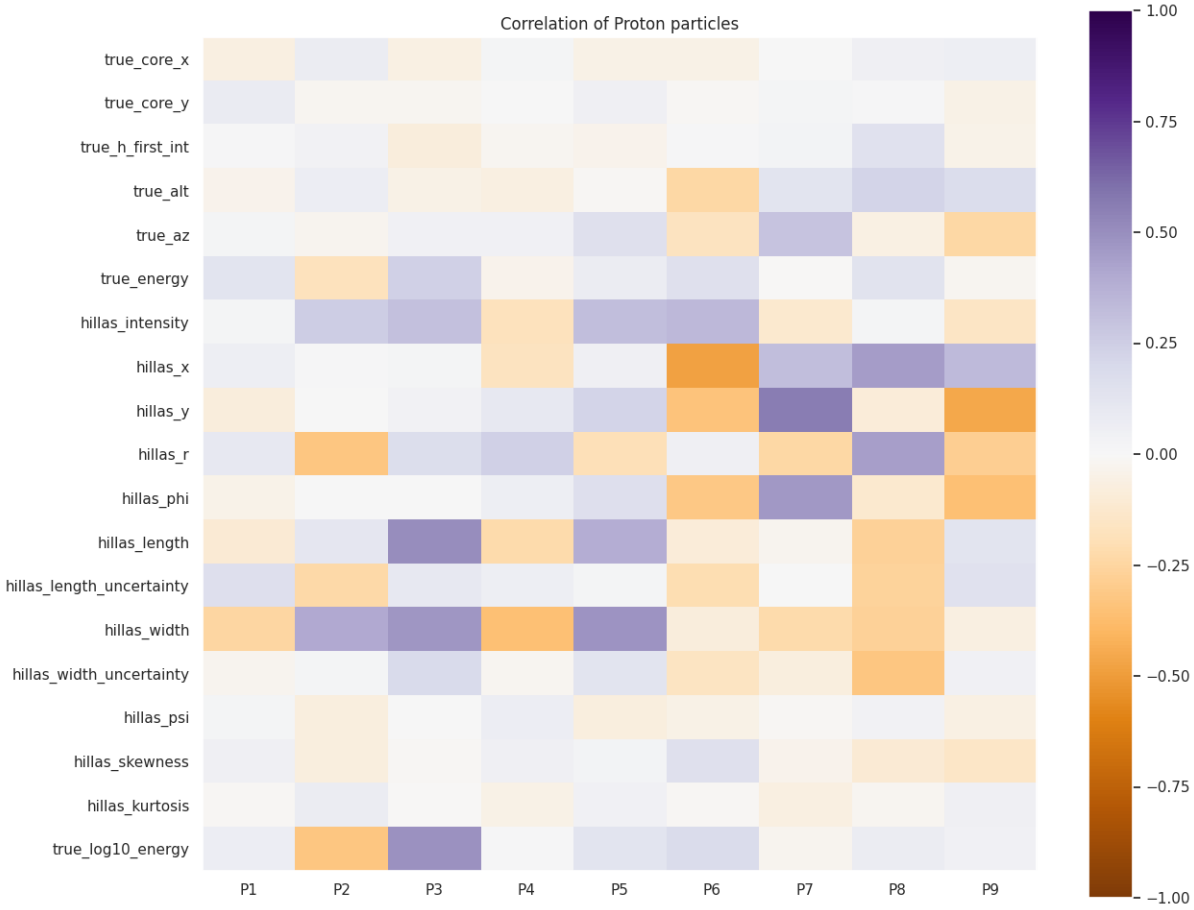


Figure B.3: Matriz de correlación entre protón y los parámetros de Hillas