Identificación de partículas por medio de una red neuronal aplicable a los datos provenientes de un Detector de Radiación Cherenkov de agua

Mayra Betsabé Silva Carranza

UVG UNIVERSIDAD DEL VALLE DE GUATEMALA

UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ciencias y humanidades

Identificación de partículas por medio de una red neuronal aplicable a los datos provenientes de un Detector de Radiación Cherenkov de agua

Trabajo de graduación en modalidad de tesis presentado por
Mayra Betsabé Silva Carranza
para optar al grado académico de Licenciado en Física

Guatemala 2020

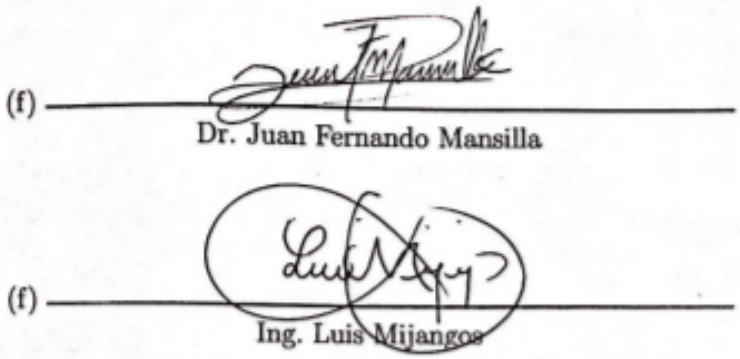
UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ciencias y humanidades

Identificación de partículas por medio de una red neuronal aplicable a los datos provenientes de un Detector de Radiación Cherenkov de agua

Trabajo de graduación en modalidad de tesis presentado por
Mayra Betsabé Silva Carranza
para optar al grado académico de Licenciado en Física

Guatemala 2020

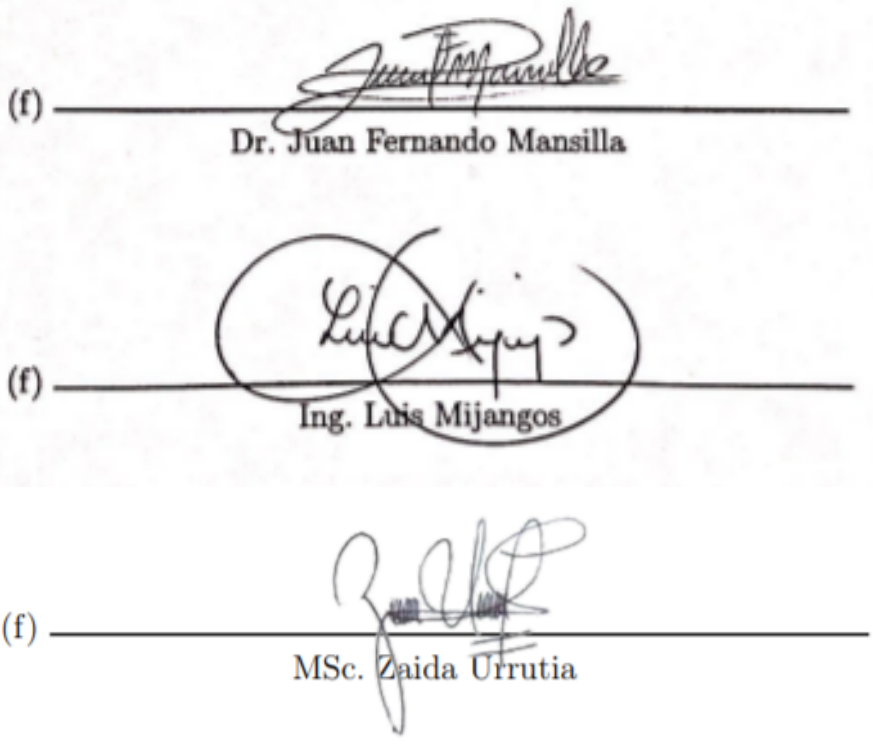
Vo.Bo.:



(f) _____
Dr. Juan Fernando Mansilla

(f) _____
Ing. Luis Mijangos

Tribunal Examinador:



(f) _____
Dr. Juan Fernando Mansilla

(f) _____
Ing. Luis Mijangos

(f) _____
MSc. Zaida Urrutia

Fecha de aprobación: Guatemala, 11 de noviembre de 2020.

Prefacio

Cuando me decidí por estudiar Licenciatura en Física no sabía realmente en qué me iba a especializar más adelante, solo sabía que quería estudiar Física para entender el mundo de una manera distinta. Tampoco tenía idea del reto que iba a implicar estudiar una carrera científica y trabajar al mismo tiempo, pero gracias a Dios heme aquí, escribiendo mi trabajo de graduación, después de 4 años de haber tomado esa decisión.

A lo largo de mi carrera tuve varios acercamientos a la física de partículas, comenzando por cursos de CERN, interacción con personas que trabajan en el CERN y luego la oportunidad de trabajar conjuntamente con la comunidad LAGO, el cual considero que es un proyecto muy interesante y que tiene bastante futuro. Todas estas pequeñas acciones dieron lugar a que se desarrollara en mi un interés especial por la física de partículas. Pero también, durante mis estudios tuve un pequeño acercamiento a lo que es *Machine Learning*, lo cual me pareció sorprendente.

Es por ello que nació la idea de este trabajo, quería combinar los dos aspectos que más me interesaron durante mis estudios: física de partículas y *Machine Learning* y a partir de ello, se desarrolló todo este proyecto.

Agradecimientos

Quiero comenzar agradeciéndole a Dios por la vida, salud y por permitirme llegar a esta etapa tan importante. Así también quisiera agradecerle a mi madre: Mayra Carranza por enseñarme a ser una persona esforzada y valiente. A mis hermanos: Roberto Carlos y Jessica, por su apoyo durante todo el desarrollo de mi carrera profesional, su amor y motivación. Asimismo, a Marcela agradezco mucho por su apoyo incondicional en todo momento y por creer siempre en mí. A mi padre: Roberto por inculcarme ese amor y pasión por la ciencia; en particular: la física.

Quiero extender un agradecimiento especial por su gestión, conocimiento y recursos al Ingeniero Luis Mijangos, quien me brindó los contactos necesarios para tener asesoría y comunicación por parte de la comunidad LAGO, en partícular: Iván Sidelnik, Hernán Asorey y María Graciela Molina. Agradezco profundamente a Luis, por su apoyo durante todo el desarrollo del proyecto, por el conocimiento extra impartido, los consejos que me brindó, gracias a él conocí el proyecto LAGO y por haberme puesto en contacto con mi Asesor de Tesis: Juan Fernando Mansilla, a quien también agradezco mucho por todo el tiempo dedicado, el conocimiento impartido acerca de *Machine Learning* y las herramientas brindadas para el desarrollo de este trabajo.

También me gustaría agradecerle a Héctor Pérez de la Universidad de San Carlos de Guatemala, sin su gran apoyo con sus conocimientos sobre *Geant 4* no hubiera sido posible para mi realizar por completo este proyecto.

Como mencioné con anterioridad a la Comunidad LAGO, también les debo un agradecimiento muy grande por haberme brindado la oportunidad de participar en sus reuniones donde pude aprender mucho de ellos y asimismo, me brindaron recursos necesarios y la retroalimentación correspondiente a mis resultados.

Un agradecimiento muy grande a mi compañero Abraham Roquel, quien me brindó los recursos computacionales, conocimientos sobre computación, tiempo y su comprensión sin importar la hora para el desarrollo y generación de datos de este proyecto.

En cuanto a la comunidad de la Universidad del Valle de Guatemala, agradezco mucho a Zaidy de Urrutia, quien a lo largo de toda mi carrera me brindó todo el apoyo necesario. A mis catedrática Irene Aguilar porque siempre fue más que una catedrática para todos nosotros, por sus enseñanzas, lecciones de vida y consejos. A mis compañeros de carrera les debo un agradecimiento muy especial en particular a Odalis, por enseñarme, por su paciencia, su apoyo sin medida, por estar siempre en las buenas y en las malas. También a Ivan por su compañía para todo, literalmente, y su ayuda. Gracias a mis queridos compañeros de la carrera, fueron unos 4 años muy alegres gracias a ustedes.

Índice

Lista de figuras

Lista de cuadros

Resumen

El objetivo general de este trabajo consiste en la identificación de partículas provenientes de cascadas de aire extensas de rayos cósmicos que inciden en la atmósfera. El estudio de física de partículas y la clasificación de las mismas, representan una contribución para entender el Universo y su composición.

A partir de una simulación realizada en Geant4 que contiene las características del detector de radiación Cherenkov de agua que se encuentra en la universidad, se generaron suficientes datos para que estos pudieran ser clasificados. Donde también se tomó información proveniente de Corsika, una simulación que permite conocer la afluencia de partículas y su nivel de energía dados ciertos parámetros, como lo son los metros sobre el nivel de mar, la latitud, etc. Estos datos fueron procesados por medio de una red neuronal la cual permitió la clasificación de partíclas. Dado este modelo de clasificación, se tendrá un aporte significativo para la comunidad científica y en particular, a la comunidad de la Universidad del Valle de Guatemala puesto que con la documentación del procedimiento, se busca que este pueda ser replicable en cualquier conjunto de datos y de esta manera realizar estudios en base a la clasificación de partículas.

CAPÍTULO 1

Introducción

LAGO es un proyecto latinoamericano (Latin American Giant Observatory, por sus siglas en inglés) que consiste en una red integrada de detectores de agua de radiación Cherenkov, distribuidos desde México hasta la Patagonia. Tiene como objetivos científicos centrales los estudios de las astropartículas de alta energía, meteorología, climatología espacial, la radiación atmosférica y sus aplicaciones. Su objeto principal de estudio son los rayos cósmicos y el resultado de la interacción con la atmósfera. Es por ello que la red se encuentra integrada por detectores de radiación Cherenkov de agua.

La radiación Cherenkov es un fenómeno que ocurre cuando una partícula cargada eléctricamente atraviesa un medio que le permite ir a una velocidad superior a la velocidad de la luz. Un método para su detección es la utilización de detectores de agua de radiación Cherenkov, cuya finalidad es registrar lluvias de rayos cósmicos. Generalmente, consta de un tanque de agua hermético, completamente aislado de la luz con fotomultiplicadores que registran el paso de una partícula a través de una salida de voltaje.

Dicho detector genera una gran cantidad de datos dado que para una sola partícula cargada que atraviesa el dieléctrico presente en el detector, el PMT es capaz de detectar una cantidad considerable de fotones. Estos resultados son empleados para varios estudios, como la muóngrafía, el estudio de la climatología espacial, el estudio de los rayos cósmicos, etc.

A partir de la gran cantidad de datos generados, surge la necesidad de implementar herramientas que permitan su análisis y clasificación de forma eficiente. Es por ello que se plantea el objetivo principal de este proyecto, donde se busca implementar algoritmos de *Machine learning* para la clasificación de partículas, en particular los muones, dado que son las partículas que más abundan en las cascadas de aire extensas y que generan una mayor cantidad de fotones, ya que inciden con rangos de energía entre 3 a 5 GeV en el municipio de Guatemala.

CAPÍTULO 2

Objetivos

2.1. Objetivo general

Clasificar partículas detectables por la electrónica del tanque Cherenkov de agua Kinich Ahau perteneciente a la red LAGO, ubicado en la Universidad del Valle de Guatemala, por medio de algoritmos de Machine Learning.

2.2. Objetivos específicos

- Afinar el procedimiento utilizado en años anteriores para la clasificación de partículas captadas por el detector Cherenkov de agua, ubicado en la Universidad del Valle de Guatemala.
- Entrenar un modelo a partir de datos generados con simulaciones, que contengan los atributos del detector de Radiación Cherenkov de agua.
- Caracterizar los pulsos de las partículas detectables en el tanque.

CAPÍTULO 3

Justificación

La Comunidad LAGO busca crear impacto por medio del estudio de la climatología espacial, LAGO es un proyecto que con el paso del tiempo se ha ido expandiendo y desarrollando. Comenzando por la creación de una red de detectores de Radiación Cherenkov de agua, implementación de protocolos para su instalación, creación de documentación para simulaciones que permiten el estudio de partículas de altas energía y el análisis de los datos recabados, es en este aspecto donde se encuentra el objetivo principal de este proyecto. Puesto que se cuenta con la existencia de un detector en el campus de la Universidad del Valle de Guatemala, el cual fue resultado de un megaproyecto realizado en 2016, se busca una forma de analizar los datos que se obtienen del detector, en particular la clasificación de las partículas detectables.

Partiendo de la necesidad de automatizar el proceso de identificar las partículas detectables, dado que se recolecta una gran cantidad de datos cuando el detector está en funcionamiento, surge la necesidad de optimizar el proceso de análisis de datos. Como se mencionó con anterioridad, en el megaproyecto donde surge la construcción del detector se buscó aplicar algoritmos de inteligencia artificial para el análisis de la gran cantidad de datos. Para ello se empleó la clasificación de Naive Bayes, asumiendo que los datos son independientes del tiempo. También se aplicaron algoritmos de aprendizaje no supervisado como lo son el *clustering* de los datos y K-Means, reportando así un nivel de precisión en el análisis de 95 %.

Según el trabajo de tesis anterior, se propone en este trabajo un plan de mejora donde se involucren algoritmos que sí tomen en cuenta la dependencia del tiempo, dado que los datos recolectados por el detector consisten en 12 valores de voltaje con respecto al tiempo, lo cual indica que el orden sí importa; dicho factor es muy importante a considerar al momento de realizar el análisis de datos. Así también, se busca documentar un proceso que automatice la identificación de partículas. Dicho proceso incluirá, que es parte del plan de mejora, etiquetas basadas en una simulación, lo cual da más precisión al método dado que el algoritmo aprenderá en base a valores reales y no sujetos a la subjetividad de un individuo.

Otro aspecto importante a mencionar es que los resultados de este trabajo serán un aporte para la comunidad LAGO, dado que en el transcurso del año, los encargados de LAGO en Argentina están buscando involucrar herramientas de *machine learning* con física de partículas. Asimismo, dicho proceso podrá ser aplicado en futuros análisis de datos provenientes del detector presente en el campus de la Universidad.

CAPÍTULO 4

Marco teórico

4.1. Partículas

El modelo estándar establece que la materia está conformada por un conjunto de partículas elementales. Las interacciones entre múltiples partículas elementales se conocen como campos, los cuales pueden representar fuerzas .

Hasta el momento se conocen 12 partículas elementales, las cuales se pueden dividir en dos familias: Leptones y Quarks. Dentro los leptones se pueden encontrar los electrones y muones. Por otro lado, en la familia de los quarks se tienen 6 sabores, que al combinarse en una forma específica dan lugar a la formación de partículas como: protones, neutrones, piones, anti-piones, etc [15] .

4.1.1. Clasificación de partículas

En el campo de física de altas energías, la clasificación de partículas juega un papel importante. La clasificación de partículas puede darse entre clases las cuales pueden ser leptones los cuales tienen espín $\frac{1}{2}$ y pueden tener carga o ser neutros, hadrones quienes tienen interacciones con las fuerzas fuertes, electromagnéticas y débiles y por último los quarks .

Asimismo, las partículas pueden ser clasificadas según el tipo de interacción las cuales están basadas en las 4 fuerzas fundamentales: electromagnéticas, e interacciones fuertes y débiles [9] .

Cabe mencionar que la clasificación de partículas permite el estudio de las fuerzas fundamentales así como también contribuye en el descubrimiento y estudio de partículas exóticas. A continuación, se describen las partículas más conocidas:

4.1.2. Electrón

Es la partículas subatómica más liviana que existe. Tiene una carga negativa de $1.602176634 \times 10^{-19}$ coulomb y una masa de $9.1093837015 \times 10^{-31} kg$ y es denotado por el símbolo e^- .

En un átomo se encuentran ubicados en los orbitales y la estructura de los mismos se conoce como la configuración electrónica de un átomo .

De acuerdo con la física de partículas el electrón se puede clasificar en dos tipos: es un fermión dado que tiene valores de medios enteros de espín. También se considera como un leptón dado que los leptones son aquellas partículas que reaccionan ante las fuerzas electromagnéticas, débil o gravitatoria [15] .

4.1.3. Positrón

El positrón es la anti-partícula del electrón, dada esta descripción, un positrón tiene la misma masa y magnitud de carga que el electrón. Es una partícula que se encuentra positivamente cargada .

Es un tipo de partícula que es estable en en el vacío y pueden ser un producto del decaimiento de partículas con tiempos de vida relativamente cortos, como lo son los muones. Cabe mencionar que al interactuar con un electrón, ambas partículas son aniquiladas [15].

4.1.4. Muón

Es una partícula subatómica bastante parecida al electrón, a excepción de que alrededor es de 207 veces más pesado. Se le considera como una partícula inestable con un tiempo de vida muy corto: $2.2\mu s$. Cuando un muón decae bajo la acción de una fuerza débil, el producto es un electrón y dos neutrinos.

La energía promedio de un muón a una altura sobre el nivel del mar es de $4GeV$, y como resultado de su interacción ionizan la materia. A medida que estos interactúan con la materia van perdiendo energía, dicha pérdida de energía es proporcional a la cantidad de materia que atraviesan [21] .

4.1.5. Protón

Es una partícula subatómica con carga positiva de un e^-, i.e $1.602176634 \times 10^{-19}$ coulomb. Tiene una masa de $1.67262 \times 10^{-27} kg$.

El protón juega un papel importante en química, dado que la cantidad de protones presentes en el núcleo de un elemento químico hace referencia al número atómico del elemento. A partir de este número es que los elementos son clasificados y posicionados en la tabla periódica [15] .

4.1.6. Neutrón

Como su nombre lo indica, el neutrón es una partícula subatómica neutra. Carece de carga eléctrica y tiene una masa de $1.67493 \times 10^{-27} kg$.

La importancia de esta partícula recae en el hecho de que varios elementos al ser bombardeados por neutrones, entran en un proceso conocido como fisión. La fisión es un tipo de reacción nuclear y es el principio sobre el cual han sido construidas las bombas atómicas [15] .

4.1.7. Pión positivo

Se representa por el símbolo π^+, tiene una masa de alrededor un sexto de la masa del protón, espín cero y pariedad intrínseca negativa .

Es el mesón más liviando y su detección puede ser utilizada para predecir el rango máximo de interacciones fuertes. Tiene un tiempo promedio de vida de $2.6 \times 10^{-8} s$ y está compuesto por un quark *up* y un quark textitanti- down[15] .

4.1.8. Pión negativo

Es el equivalente a la anti-partícula del pión positivo, junto con el pión positivo se les considera como partículas inestables y también se les conoce por ser los hadrones más livianos. Cuando decaen, se obtiene un anti-muón y un muón anti-neutrino. Otro tipo de decaímiento conocido para este tipo de partícula, es aquel donde se obtiene un electrón y un electrón anti-neutrino [15] .

4.2. Rayos cósmicos

Los rayos cósmicos están constantemente impactando la atmósfera: consisten en núcleos ionizados cuya composición corresponde en un 90 % protones, 9 % de Partículas alfa y el 1 % restante a núcleos más pesados, y estos son de especial interés dado que son partículas de altas energías. [8] .

El origen de los rayos cósmicos no está muy claro aún, aunque se cree que se originan en las afueras del sistema solar y son el producto de procesos astrofísicos como las explosiones supernova, fulguraciones solares, pulsares, etc. Su descubrimiento inicia en 1912 donde Victor Hess, quien pretendía medir la ionización de la atmósfera en función de la altura sobre el nivel del mar, y encontró una proporcionalidad directa entre dichos factores [6] .

4.2.1. Espectro de los Rayos Cósmicos

El espectro de los rayos cósmicos hace referencia al flujo de partículas en función de la energía de las mismas. Para el caso de los rayos cósmicos, el espectro de energía se encuentra dominado por el viento solar y los campos magnéticos de la heliosfera y magnetósfera .

Se pueden observar tres regiones, donde se da un cambio en el valor del índice espectral y se les conocen como: rodilla, tobillo y corte; como se muestra a continuación:

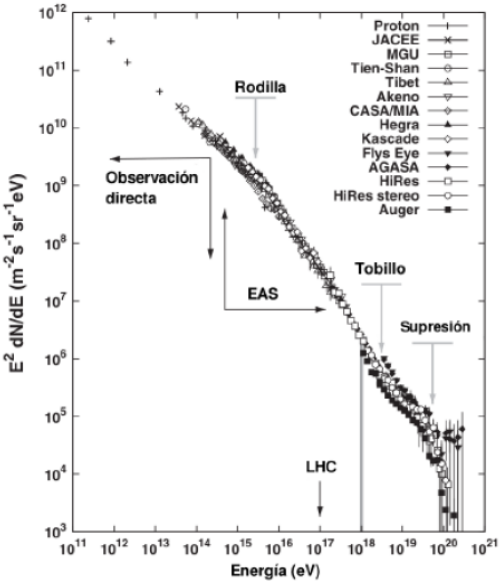


Figura 4.1: Regiones del espectro de energía de los Rayos Cósmicos, extraída de [2]

Cada región tiene diversos fundamentos físicos que la respaldan:

- Rodilla: cambio de composición química hacia elementos que son más pesados.
- Tobillo: cambio de componentes galácticas a extragalácticas.
- Corte: se debe a la producción de fotopiones, resultado de la interacción de los rayos cósmicos con los fotones de la radiación de fondo de microondas.

[2]

4.2.2. Cascadas de aire extensas de rayos cósmicos

Las partículas primarias presentes en un rayo cósmico, al incidir en la atmósfera e interactuar con las moléculas que componen a la misma, dan lugar a las cascadas de aire extensa (EAS por sus siglas en inglés: *Extensive Air Shower*) . Las partículas que componen las cascadas se les conoce como partículas secundarias y son el resultado de la interacción y decaimiento de partículas con la atmósfera. Dichas interacciones pueden ser cuantificadas por medio de la siguiente expresión para el *camino libre medio*:

$$\lambda_i = \frac{N_A}{A}\sigma_i[gr/cm^2] \tag{4.1}$$

En esta expresión, N_A es el número de Avogadro, A es el número atómico y $sigma_i$ hace referencia a la sección transversal de interacción. [6].

Como resultado de las interacciones, se tiene un decaimiento en cascadas formadas por nuevas partículas que van produciendo partículas con menor energía.

4.3. Radiación de Vavilov-Cherenkov

Este fenómeno se observa cuando la velocidad de una partícula en un medio es mayor que la velocidad de la luz. Dicho fenómeno se descubrió en 1930 por P.A Cherenkov y luego Vavilov se encargó de realizar publicaciones en base a los experimentos de Cherenkov. El efecto Cherenkov se caracteriza por la emisión de luz cuya longitud de onda se encuentra entre 350 nm a 500 nm, el cual es un rango admisible para los tubos fotomultiplicadores o tubos PMT [19].Un aspecto importante a destacar es el tipo de simetría que presenta, donde la radiación se propaga a un ángulo del movimiento de la partícula, según la siguiente expresión:

$$cos\theta = \frac{1}{\beta n} \tag{4.2}$$

A continuación se muestra la geometría de la radiación de Cherenkov:

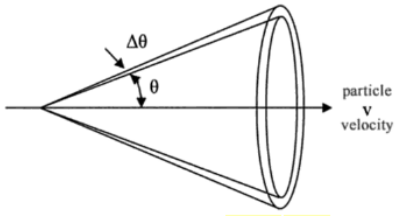


Figura 4.2: Geometría de radiación Cherenkov. Extraída de [19]

Como se muestra en 4.3, la radiación se emite con un ángulo θ con respecto a un cascarón cónico. El grosor de este cascarón hace referencia a la dispersión de la sustancia por medio de la cual se propaga, la difracción y la dispersión de las partículas [19] .

De esta manera, se puede cuantificar el número de fotones emitidos por unidad de longitud por medio de la siguiente expresión:

$$\frac{dN}{dx} = \frac{z^2\alpha}{c}\int d\omega\left(1 - \frac{1}{\beta^2 n(\omega)^2}\right)$$

Donde z es la carga de la partícula, α corresponde a la constante de estructura fina y c es la velocidad de la luz en el vacío. De esta expresión se puede ver que la emisión de la radiación Cherenkov se da si se cumple la siguiente expresión:

$$\beta > \frac{1}{n(\omega)} \tag{4.3}$$

[14]

4.3.1. Detector de Radiación Cherenkov de agua, WCD

Conocido también por sus siglas en inglés, *Water Cherenkov Detector* (WCD), consiste en un calorímetro homogéneo. Está construido a partir de un cilindro recubierto que no permite el paso de la luz y su interior está recubierto de un material reflector. El cilindro contiene agua hiperpura, que se utiliza como el medio dieléctrico dispersivo, y un tubo fotomultiplicador .

El funcionamiento se basa en el hecho de que una partícula cargada atraviesa el agua hiperpura, y a partir de esta interacción se observa el fenómeno de radiación Cherenkov, la cual es cuantificada por medio del PMT.[14]

4.3.2. Tubo fotomultiplicador PMT

Un tubo fotomultiplicador, o PMT, es un tubo vacío que contiene las siguientes partes:

- Fotocado: consiste en cátodos foto-sensibles que convierten los fotones incidentes en electrones. Su funcionamiento se basa en el efecto fotoeléctrico.
- Un multiplicador de electrones: se crea una amplificación de la fotocorriente.
- Ánodo: recolecta el flujo total de electrones.

El funcionamiento de un PMT se basa en que cuando ingresa un rayo de luz a través del fotocado se emiten fotoelectrones en el tubo. Luego, los fotoelectrones son guiados por los electrodos hacia el multiplicador de electrones y en el multiplicador de electrones son transmitidos a dinodos. Finalmente, el ánodo se encarga de recolectar el flujo total [16] .

4.3.3. Circuito análogo a un PMT

Para efectos de simulación, un PMT puede ser simulado por medio de un circuito con una fuente de corriente en paralelo con una resistencia R_0 y un capacitor C_0, como se muestra a continuación:

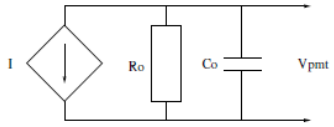


Figura 4.3: Circuito equivalente a un tubo fotomultiplicador, extraída de 4.5

Resolviendo el circuito por medio del método de mayas de Kirchoff se obtiene la siguiente expresión:

$$I(t) + \frac{V_{PMT}(t)}{R_0} + \frac{dC_0 V_{PMT}(t)}{dt} = 0 \tag{4.4}$$

La expresión anterior se puede resolver aplicando el teorema de convolución y la transformada de Fourier, obteniendo así el siguiente resultado:

$$V_{PMT}(t) = \frac{-e^{t/R_0C_0}}{C_0} \int_{\infty}^{t} dx I(x) e^{x/R_0C_0} \tag{4.5}$$

[14]

4.4. Machine learning

Machine learning es un tipo de aplicación de la Inteligencia Artificial que se asocia con cambios en un sistema. Este procedimiento puede involucrar tareas de clasificación, predicción, diagnóstico, reconocimiento y planeación. La importancia de esta aplicación está en el hecho de poder crear relaciones entre variables para conjuntos grandes de datos, donde se busca que la computadora sea quien cree dichas relaciones. También, otro factor, es el tiempo operativo donde si se se designa

una larga cantidad de tareas para un conjunto considerable de datos a una computadora le tomará mucho menos en ejecutar las tareas en comparación de lo que le tomaría a una persona. Y por último se puede mencionar la capacidad que tienen las computadoras adaptarse con mayor facilidad a los cambios y evoluciones [17] .

El principio básico de *Machine learning* se basa en crear aproximaciones que sean útiles para la identificación, reconocimiento y clasificación de un conjunto de datos. Dichas aproximaciones, permiten la creación de patrones o regularidades las cuales permiten la realización de predicciones en un conjunto de datos. Para llevar a cabo dichas tareas, se le provee a la computadora con un criterio utilizando ejemplos o experiencias del pasado para luego llegar al proceso de aprendizaje. El aprendizaje de una computadora se basa en la ejecución de las tareas que optimicen los parámetros del modelo, en este proceso la computadora debe utilizar el conjunto de datos de entrenamiento o experiencias pasadas. Para su funcionamiento, *Machine Learning* utiliza herramientas estadísticas con el fin de poder crear un modelo matemático el cual permita crear inferencias a partir de la muestra brindada. El hecho de *aprender una regla* implica un proceso de extracción de conocimiento, el cual puede ser características, relaciones, correlaciones, etc. Asimismo, dentro del proceso de aprendizaje de una regla la computadora también realiza la tarea *compresión* donde ajusta la regla aprendida al conjunto de datos. Este procedimiento es repetido tantas veces sean necesarias para asegurar el aprendizaje de la computadora y de esta forma obtener un resultado [1] .

Hoy en día, *Machine learning* tiene aplicaciones en ciencias, como lo es astronomía, biología y ciencia, donde optimiza el procesamiento de grandes cantidades de datos.

4.5. Importancia de Machine Learning en el proyecto

La utilización de *Machine Learning* en este proyecto se basa en el hecho de que esta herramienta permite la realización de tareas de clasificación y optimización de una manera mucho más eficiente y rápida si se compara con el tiempo que le tomaría a una persona completar dichas tareas para una cantidad considerable de datos .

Además, parte de la evolución de *Machine Learning* es que la precisición y efectividad del modelo aumenta conforme el número de datos también aumenta. Es aquí donde recae la importancia y se justifica el uso de esta herramienta para el análisis de los datos, puesto que en este trabajo se empleará una basta base de datos .

4.5.1. Redes neuronales artificiales

Las redes neuronales son redes computacionales que, como su nombre lo indica, intentan simular la conexión de las neuronas de un sistema biológico para la toma de decisiones y la transmisión de información. Se basan en una simulación donde se van creando neuronas, las cuales van descubriendo atributos importantes para modelos de predicción y optimización. Es un tipo de red compuesta por elementos que no son lineales, los cuales están interconectados por pesos que se ajustan por el usuario. La red neuronal devuelve como salida la suma promedio de los pesos pertenecientes a las características que permiten la identificación o clasificación de un fenómeno (es en esta parte donde toman como base las conexiones nerviosas), dado que permite utilizar operaciones computacionales bastante simples para resolver problemas complejos que no se comportan de forma lineal, o problemas estocásticos [17]. Asimismo, permite la organización de atributos de forma propia, donde se busca retener características que permitan la predicción u optimización de un fenómeno [11] .

Con el tiempo, el desarrollo de las redes neuronales ha ido evolucionando, y se pueden destacar los dos tipos de redes neuronales más utilizadas: redes neuronales convolucionales (Convolutional Neural Network, CNN por sus siglas en inglés) o las redes neuronales recurrentes (Recurrent Neural

Network, RNN por sus siglas en inglés).

CAPÍTULO 5

Antecedentes

5.1. LAGO

LAGO (Latin American Giant Observatory, por sus siglas en inglés) es un proyecto de escala global, cuyo enfoque está orientado en las ramas de la astrofísica de partículas: fenómenos de altas energías, climatología espacial y la radiación atmósferica al nivel de la tierra. Consiste en una red de detectores de radiación Cherenkov de agua ubicados en diferentes latitudes y altitudes (desde México hasta Antártida) .

Este proyecto es coordinado por más de 80 científicos de 9 países de latinoamérica distintos, entre ellos: Argentina, México, Perú y Venezuela [13] .

5.2. LAGO UVG (Megaproyecto)

En 2016 un grupo de estudiantes de la Universidad del Valle de Guatemala desarrollaron el diseño y construcción de un detector de radiación Cherenkov de agua para la toma y procesamiento de datos provenientes de rayos cósmicos. Este proyecto se desarrolló a partir de varios módulos: química, física, electrónica, computación y gestión .

El módulo de mayor importancia para este trabajo de graduación será el de computación y análisis, dado que se planteará un plan de continuación y mejora a este módulo del proyecto. En este Megaproyecto, se emplearon también algortimos de Inteligencia Artificial y *Machine Learning*. Se aplicaron algoritmos de clasificación de aprendizaje supervisado y no supervisado. Para la Clasificación No Supervisada se utilizó *K-Means* y para la Clasificación Supervisada se utilizó el algoritmo Naïve Bayes. En este punto es donde se identifican aspectos a mejorar, dado que al momento de utilizar este tipo de algoritmo se asume que todos los datos son independientes aunque esto no necesariamente es cierto. Esto es porque en el histograma de cargas por evento, el orden de la carga registrada si importa .

Otro aspecto importante a mencionar en este módulo es que utilizaron etiquetas creadas de forma manual, i.e los estudiantes observando la forma del histograma clasificaban la partícula y a partir de ello creaban la etiqueta para designar el tipo de partícula. Luego, al momento de generar eventos en la simulación de Kinich Ahau se basaron en la distribución de Michael para conocer la energía

promedio de las partículas a utilizar. En este caso fueron: electrones de 37 MeV, muones de 4 GeV y rayos gamma de 100 MeV. A partir de estos datos, se realizaron entre 500 a 1000 eventos para cada tipo de partícula y se buscó conocer la amplitud promedio por partícula.

5.2.1. Simulaciones realizadas en Geant4 para un detector de radiación Cherenkov de Agua: Kinich Ahau

KinichAhau es una simulación realizada en Geant 4, el cual es un programa desarrollado por el CERN diseñado para la simulación del paso de partículas a través de la materia. Tiene aplicaciones en áreas de la física como: física de altas energías, física nuclear y física de aceleradores de partículas [4] .

A partir de la herramienta de Geant 4, en 2016, Daniel Conde [5] trabajó en la simulación que contiene los parámetros asignados según el detector que se encuentra en el campus de la Universidad del Valle de Guatemala. Esta simulación se desarrolló basándose en los ejemplos que ya ofrece Geant 4 entre sus librerías, entre ellos *OpNovice* el cual hace referencia a una simulación con radiación Cherenkov y también el ejemplo *HitLXE* el cual implementa un PMT en la simulación [5] .

CAPÍTULO 6

Alcance

A lo largo del desarrollo de este trabajo se espera lograr la clasificación de partículas presentes en cascadas de aire extensas, provenientes de Rayos Cósmicos y el tipo de partículas a trabajar son aquellas que se conoce pueden llegar a ser detectadas. Este trabajo a su vez, pretende ser una continuación y aporte al proyecto LAGO en Guatemala, el cual fue iniciado en 2016 con el trabajo de Daniel Conde y compañeros [5] quienes construyeron el primer detector de radiación Cherenkov de agua en la Universidad del Valle de Guatemala. En dicho proyecto, se aplicaron algoritmos de *Machine Learning* y por lo mismo, este trabajo ofrece un plan de mejora al mismo. Así como también una herramienta de clasificación de partículas, la cual puede ser implementada en el futuro cuando el tanque vuelva a estar en funcionamiento.

Para el desarrollo del plan de mejora mencionado se utilizarán simulaciones de partículas, de esta manera se asegurará que las etiquetas creadas para la clasificación de partículas no tengan un sesgo o error proveniente de etiquetas creadas .ª mano", donde el individuo en base al histograma determina que tipo de partícula es. Cabe mencionar, que en estas simulaciones realizadas por cuestiones de limitaciones en cuanto a herramientas computacionales, no se utilizarán eventos donde las partículas incidan en el tanque con diferentes ángulos. Mas bien, en las simulaciones, se considera únicamente el caso donde las partículas inciden de forma vertical.

Por último, dado que se espera que esta sea una herramienta para la comunidad LAGO - Guatemala a los datos con los que ya se cuentan del funcionamiento del tanque Kinich Ahau en el pasado, se le aplicará la red neuronal para poder obtener una clasificación sobre el tipo de partículas que detecta el tanque.

CAPÍTULO 7

Simulaciones y muestreo de partículas

7.1. Simulaciones de cascadas de aire extensas en CORSIKA

7.1.1. Distribución de cascadas de aire extensas en Guatemala

CORSIKA (COsmic Ray SImulations for KAscade, por sus siglas en inglés) es un programa para la realización de simulaciones de cascadas de aire extensas. Permite simular las interacciones de partículas primarias, trabajando con protones, átomos de hierro y fotones [18]. A partir de esta simulación y la contribución de la comunidad LAGO, se pudo obtener la distribución de partículas y su energía dada la altura sobre el nivel del mar del municipio de Guatemala (correspondiente a 1500 m)[20]. Cabe mencionar que los datos simulados hacen referencia a 4 horas de conteo de partículas como se muestra a continuación:

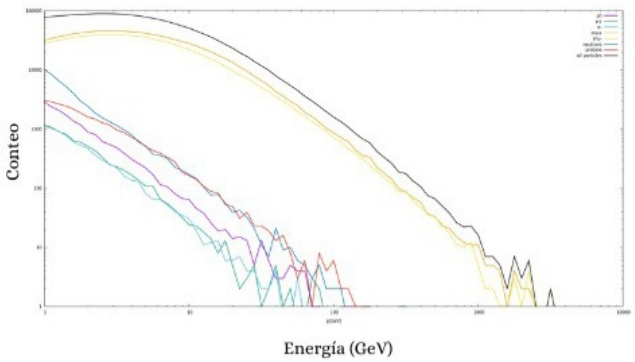


Figura 7.1: Distribución de partículas en Guatemala

A partir de la distribución de partículas, también se hizo un conteo de partículas captadas en general para conocer cuáles son las que más abundan a un nivel sobre el suelo cercano al nivel del

mar.

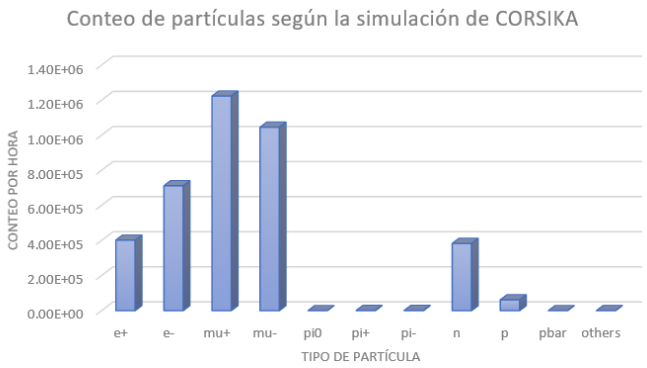


Figura 7.2: Conteo de partículas según los resultados de $CORSIKA$

A continuación, se muestra la distribución para tres partículas en partícular: μ^-, e^-, e^+ dado que son las más abundantes en los resultados obtenidos de la simulación de CORSIKA .

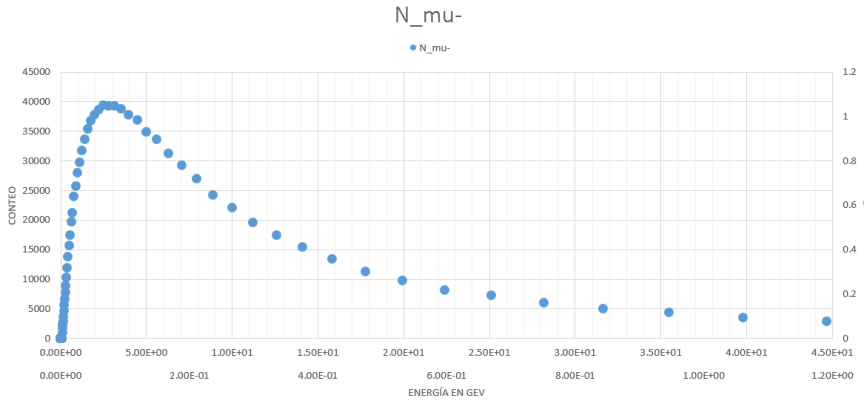


Figura 7.3: Distribución de muones en Guatemala

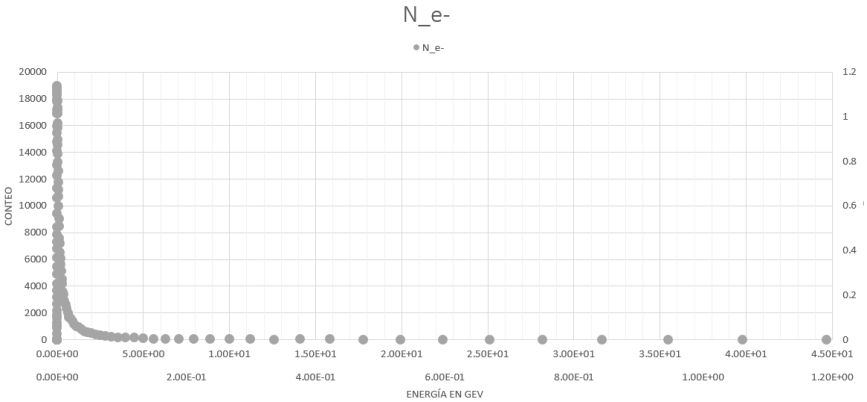


Figura 7.4: Distribución de electrones en Guatemala

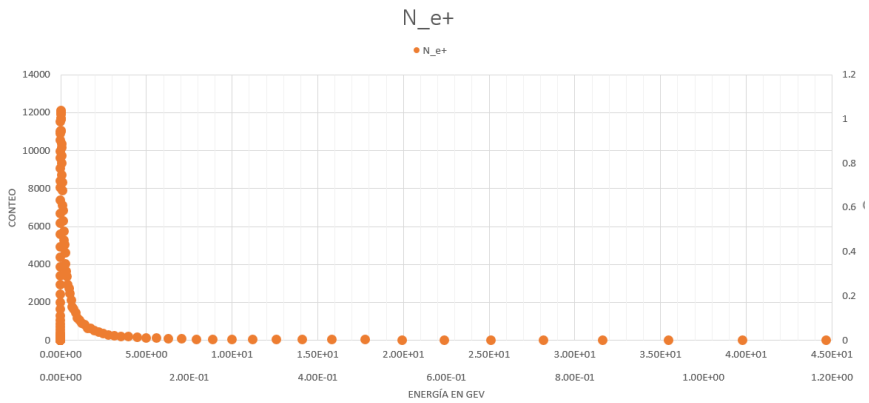


Figura 7.5: Distribución de positrones en Guatemala

Como se puede observar en las figuras 7.4 y 7.7 la cantidad de e^- y e^+ es mayor para niveles de energía relativamente bajos, en rangos menores a 1 GeV. También se tiene el caso opuesto, como en la Figura 7.3, donde la mayor cantidad de muones que se captan a la altura sobre el nivel del mar están en el rango de 3 a 5 GeV .

7.1.2. Análisis de la distribución de partículas

Para escoger cuáles eran las partículas con las que se iban a trabajar en las simulaciones, se analizó la distribución de las mismas. A continuación se muestran las estadísticas para la distribución en general:

```
    Energy              photon               e.                e..1              mu.               mu..1        
Min.   :    0.00   Min.   :      0   Min.   :     0   Min.   :     0   Min.   :      0   Min.   :      0  
1st Qu.:    0.01   1st Qu.:      3   1st Qu.:     0   1st Qu.:     0   1st Qu.:      0   1st Qu.:      0  
Median :    0.56   Median :   6383   Median :    65   Median :   298   Median :      5   Median :      3  
Mean   :  703.12   Mean   : 120341   Mean   :  4872   Mean   :  8630   Mean   :  14808   Mean   :  12702  
3rd Qu.:   63.10   3rd Qu.: 137342   3rd Qu.:  3844   3rd Qu.:  7573   3rd Qu.:   7828   3rd Qu.:   6591  
Max.   :58000.00   Max.   :9930000   Max.   :402000   Max.   :712000   Max.   :1220000   Max.   :1050000  
      pi..              pi.              pi..1              n                 p                pbar             others      
Min.   :0   Min.   :  0.00   Min.   :  0.00   Min.   :     0   Min.   :    0.0   Min.   : 0.0000   Min.   :  0.000  
1st Qu.:0   1st Qu.:  0.00   1st Qu.:  0.00   1st Qu.:     0   1st Qu.:    0.0   1st Qu.: 0.0000   1st Qu.:  0.000  
Median :0   Median :  0.00   Median :  0.00   Median :     0   Median :    0.0   Median : 0.0000   Median :  0.000  
Mean   :0   Mean   : 10.81   Mean   : 12.04   Mean   :  4644   Mean   :  750.6   Mean   : 0.5818   Mean   :  2.097  
3rd Qu.:0   3rd Qu.: 11.00   3rd Qu.: 12.00   3rd Qu.:    21   3rd Qu.:   23.0   3rd Qu.: 0.0000   3rd Qu.:  1.000  
Max.   :0   Max.   :892.00   Max.   :993.00   Max.   :383000   Max.   :61900.0   Max.   :48.0000   Max.   :173.000  
     total        
Min.   :       0  
1st Qu.:    3561  
Median :   77760  
Mean   :  167009  
3rd Qu.:  151013  
Max.   :13800000
```

Figura 7.6: Estadísticas sobre la distribución de partículas

A partir de los máximos para cada tipo de partícula, se pudo determinar que partículas tienen un aporte significativo al conjunto de datos. Como lo fueron: muones, anti-muones, protones, neutrones. También se tiene el caso contrario, donde las partículas que no tenían un aporte cuyo conteo era comparable con el resto de partículas fueron: piones, neutrones y otro tipo de partículas. Cabe mencionar que el orden de magnitud del conteo para los piones positivos y piones negativos difería casi en dos órdenes de magnitud para el resto de partículas, sin embargo se decidió incluir este tipo de partículas en las simulaciones .

Es así como en este punto se decidieron las partículas con las cuales se iba a trabajar en las simulaciones, las cuales fueron: muones, anti-muones, electrones, positrones, protones, neutrones, pión positivo y pión negativo .

7.1.3. Muestreo a partir de la simulación de la fluencia de partículas

Con el fin de evitar cualquier tipo de sesgo se realizó un muestreo aleatorio a partir de la distribución de las partículas y su energía. Para ello se le asignó un identificador, el cual es un número aleatorio generado en Excel, a cada una de las energías. Luego, según dicho identificador se ordenaron los datos en forma ascendente. A partir de esto, se procedió con tomar la cantidad de datos cuya sumatoria resultara en alrededor 150 mil partículas. Cabe recordar es que solamente se tomaron en cuenta las partículas cuyo conteo era representativo y comparable en órden de magnitud a las demás partículas, en particular fueron: electrón, muón, anti-muón, positrón, pión positivo, anti pión y protón .

Hasta este punto, se tenía un archivo donde para una energía y partícula dada se tenía el conteo de n partículas. Partiendo de este archivo de Excel, se procedió con generar un archivo de tipo .csv donde para cada tipo de partícula se generaron n líneas con la partícula y energía correspondiente. Al finalizar, se obtuvo un total de 220,498 filas en el archivo. Para cada fila se tenía una casilla de identificador, una de tipo de partícula y una de energía en eV .

La generación de este archivo fue muy importante en el desarrollo del trabajo dado que éste contenía los parámetros a utilizar en las simulaciones de Geant4, y brindó un acercamiento de las simulaciones a los datos reales, puesto que la fluencia de partículas fue creada a partir de los parámetros de Guatemala (básicamente la altura sobre el nivel del mar) .

Al tener el archivo que contenía los parámetros para ingresar a la simulación de Geant4, se realizó un programa en Python, el cual se encargó de la etapa de generación de eventos, limpieza de datos y procesamiento de datos .

El programa de Python recibía como parámetro el archivo de tipo de partícula, identificador y energía generado con anterioridad y la ubicación en la computadora de donde se irán guardando los resultados. Para comenzar, por cada línea existente en el archivo mencionado con anterioridad, la leía y procedía con escribir los parámetros en el archivo *KinichAhau.in* donde básicamente le indicaba a la simulación el tipo de partícula y la energía con la que se iba a realizar el evento .

Seguido de esto, en la etapa de limpieza de datos, dado que para cada evento generado (un evento es la simulación para una partícula y su energía correspondiente) se escribía un archivo de tipo .csv donde las primeras 469 líneas del archivo correspondían a la versión de Geant 4, los parámetros y procesos que lleva a cabo la simulación; se requería eliminar estas líneas de los archivos dado que no representaban información importante para el conteo de fotones Cherenkov. A parte para cada tiempo de llegada de fotones Cherenkov, al PMT, este tiempo al ser registrado en picosegundos y nanosegundos, ps y ns respectivamente con sus abreviaturas en vez de la cantidad que representan en forma numérica, se tuvo que realizar la sistitución equivalente a:

$$ps = 1 \times 10^{-12} s$$
$$ns = 1 \times 10^{-9} s$$

Aquí finaliza la etapa de limpieza de datos.

Por otro lado, cabe mencionar el total de eventos significativos que se generaron fueron un total de 159420, correspondientes a diferentes partículas y diferentes energías. Es importante que solamente consideró el caso donde el ángulo de incidencia de las partículas en el detector era perpendicular. A continuación la distribución de datos, según el tipo de partícula:

7.1.4. Histogramas característicos de las partículas

A continuación, se inició la etapa de procesamiento de datos. Para ello, con el archivo ya libre de texto, se procedió la realización un histograma que tuviera 12 clases. Se escogió este número de clases

Tipo de partícula	Porcentaje representativo en el conjunto de datos
Positrón (e^+)	10.82 %
Muón positivo (μ^+)	37.89 %
Muón negativo (μ^-)	26.75 %
Electrón (e^-)	12.66 %
Pión positivo (pi^+)	0.50 %
Pión negativo (pi^-)	0.54 %
Protón (p)	10.84 %

Tabla 7.1: Porcentaje representativo para cada tipo de partícula

ya que en los datos con los que se cuentan provenientes del detector ubicado en la Universidad del Valle, la electrónica permite captar 12 pulsos por evento. A continuación se muestra un histograma generado en la simulación de Geant4 para un muón de 4 GeV .

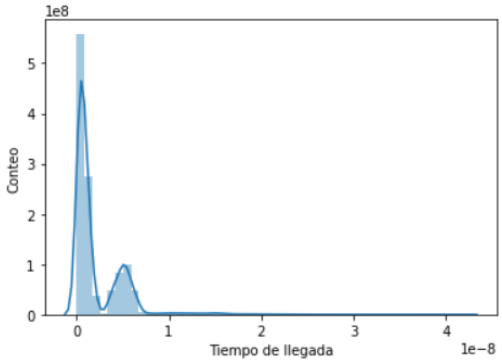


Figura 7.7: Conteo de Fotones Cherenkov detectados para μ^- de 4 GeV

Hasta este punto se contaba ya con histogramas característicos para el tipo de partícula y distintas energías. Sin embargo, dado que el PMT y el circuito con el que se cuenta en el detector del campus de la Universidad del Valle funcionan como un osciloscopio, era preciso tener valores de voltaje y tiempo para poder comenzar con la clasificación de partículas .

7.1.5. Integración numérica para la obtención de histogramas de voltaje y tiempo

Fue necesario llevar a cabo un proceso de integración numérica dado que la salida de datos de la simulación en Geant 4 era el conteo de fotones Cherenkov detectados y su tiempo de llegada al PMT. Para ello se utilizó la expresión 4.5 para aplicar a cada arreglo de conteo y tiempo según el tipo de partícula. Se realizó una integración numérica con los siguientes parámetros (los cuales fueron tomados según la electrónica del detector Kinich Ahau y [5]):

$$R = 50\Omega \qquad\qquad C = 4.7 \times 10^{-12} F$$

Este procedimiento se realizó con el fin de obtener el voltaje, dado que el resultado que se tiene de la simulación es una corriente I que depende del tiempo, puesto que tenemos la cantidad de fotones Cherenkov en un tiempo dado. Es por ello que los 12 valores del histograma realizado con anterioridad, hacen referencia a los 12 valores de corriente que se van a integrar utilizando la expresión para el circuito equivalente del PMT.

A continuación se describe el procedimiento empleado en el algoritmo:

1. Dado que la integral se puede definir como una sumatoria acumulada, se creó una lista que irán almacenando los valores del voltaje donde el siguiente elemento de la lista incluye la suma del anterior.

2. Para cada valor en la lista del conteo del histograma se multiplicó por el término de la exponencial que se encuentra adentro de la integral en 4.5 cuyo argumento son los parámetros de R y C definidos con anterioridad y x es el tiempo.

3. A cada valor obtenido, se le multiplicó por la exponencial que se encuentra multiplicando la integral en 4.5, cuyo argumento t hace referencia nuevamente al tiempo de cada *bin* del histograma.

4. Se procuró que el elemento sucesivo de la lista incluyera la suma del elemento anterior y así sucesivamente.

A continuación, se muestra el resultado obtenido.

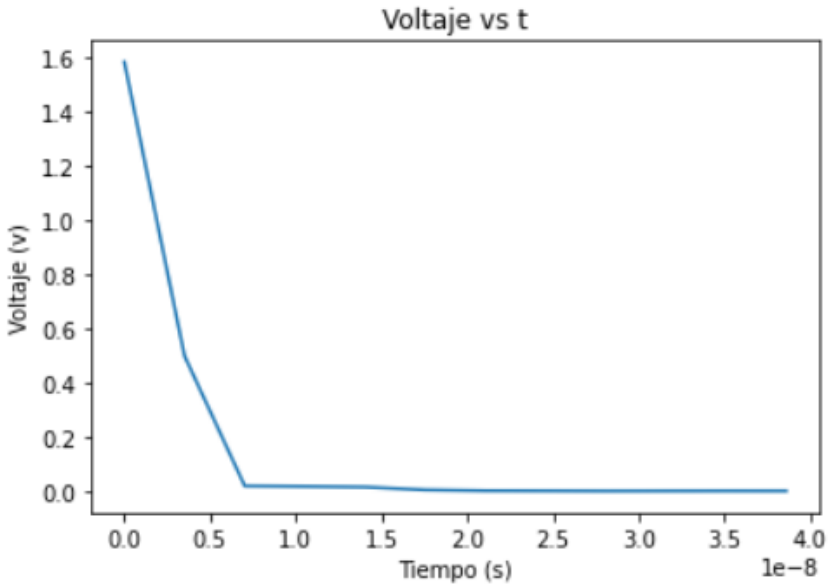


Figura 7.8: Gráfica de voltaje, luego de haber realizado la integración numérica

CAPÍTULO 8

Clasificación de partículas

Antes de iniciar las pruebas para un clasificador de partículas por medio de una red Neuronal, se utilizó el Software $WEKA$ el cual es una interfaz gráfica que contiene algoritmos ya implementados de *Machine learning*. A continuación, se describen los resultados obtenidos con los siguientes algoritmos:

8.0.1. Naïve Bayes

A continuación se muestran los resultados luego de aplicar el algoritmo de clasificación de Naïve Bayes:

Como se muestra en los resultados, el rendimiento no es bueno. Esto es porque las suposiciones para la utilización de este algoritmo no se cumplen. Para este tipo de clasificador, todos los atributos se clasifican como independientes unos de los otros así como también se considera que cada contribución de los atributos es igual para todos. Ambos aspectos no se cumplen para el conjunto de datos dado que cada valor de voltaje es dependiente de la posición y el tiempo y por lo mismo, no es el método adecuado .

En la matriz de confusión también se puede observar que para las categorías de μ^-, μ^+ no se clasifican bien estas partículas, además cabe resaltar que no se tiene una matriz diagonal. Es de

Clase	Precisión	Recall	F-1
Electrón (e^-)	0.795	0.803	0.799
Muón positivo (μ^+)	0.550	0.356	0.432
Protón (p)	0.016	0	0
Positrón (e^+)	0.214	0.748	0.422
Muón negativo (μ^-)	0.482	0.613	0.540
Pión negativo (π^-)	0	0	0
Pión positivo (π^+)	-	0	-

Tabla 8.1: Métricas luego de aplicar Naive Bayes

Electrón (e^-)	**Muón positivo** (μ^+)	**Protón** (p)	**Positrón** (e^+)	**Muón negativo** (μ^-)	**Pión negativo** (π^-)	**Pión positivo** (π^+)	
3906	0	0	955	3	0	0	**Electrón** (e^-)
331	0	68	2694	6145	95	0	**Muón positivo** (μ^+)
14	1565	0	2260	334	0	0	**Protón** (p)
291	303	426	3031	0	0	0	**Positrón** (e^+)
359	2246	13	1291	6181	0	0	**Muón negativo** (μ^-)
5	49	17	53	69	0	0	**Pión negativo** (π^-)
10	52	11	30	89	0	0	**Pión positivo** (π^+)

Tabla 8.2: Matriz de confusión de Naive Bayes

especial atención ver que en la matriz de confusión, el algoritmo no es capaz de clasificar los piones, así como también se tiene un rendimiento pobre para los muones positivos .

Dados los resultados, se decidió crear variables independientes, tales como el valor máximo de voltaje y la energía. A acontinuación se presentan los resultados luego de haber aplicado nuevamente Naive Bayes al conjunto de datos:

	Precisión	**Recall**	**F-1**
Muón positivo (μ^+)	0.707	0.322	**0.442**
Muón negativo (μ^-)	0.767	0.514	**0.289**
Protón (p)	0.120	0.147	**0.132**
Electrón (e^-)	0.314	0.115	**0.169**
Positrón (e^+)	0.616	0.648	**0.603**
Pión negativo (π^-)	0	-	**-**
Pión positivo (π^+)	0	-	**-**

Tabla 8.3: Métricas luego de aplicar Naive Bayes con variables independientes del tiempo

En efecto, se puede observar cierta mejoría con la aplicación de Naive Bayes anterior, basándose en el valor de precisión se tiene que para los muones negativos, electrones, protones si hay una mejor precisión .

Muón positivo (μ^+)	**Muón negativo**(μ^-)	**Protón** (p)	**Electrón** (e^-)	**Positrón** (e^+)	**Pión negativo** (π^-)	**Pión positivo** (π^+)	
6649	11960	435	**1472**	134	0	0	μ^+
2548	11064	748	**33**	36	0	0	μ^-
91	2604	866	**0**	2320	0	0	p
9	2515	3545	**791**	0	0	0	e^-
0	174	1482	**223**	4070	0	0	e^+
50	174	57	**2**	36	0	0	π^-
58	139	58	**1**	16	0	0	π^+

Tabla 8.4: Matriz de confusión de Naive Bayes con variables independientes del tiempo

Al observar la matriz de confusión, en definitiva, no se tiene una matriz diagonal. Por lo tanto, se puede decir que Naive Bayes no es el mejor algoritmo para utilizar con este conjunto de datos .

8.0.2. Regresión logística

A continuación se muestran los resultados luego de aplicar una función logística para la clasificación de datos.

Si se compara con el algoritmo de *Naive Bayes* la precisión para la clasificación de partículas aumentó. Sin embargo al observar la matriz de confusión en 8.6 se puede ver sigue sin poder clasificar e identificar las partículas piones, así como también falla bastante en la clasificación de protones. Esto es porque, nuevamente no se cumplen las suposiciones: para este algoritmo los datos deberían

Clase	Precisión	Recall	F-1
Electrón (e^-)	0.811	0.768	0.789
Muón positivo (μ^+)	0.621	0.792	0.696
Protón (p)	0.729	0.440	0.549
Positrón (e^+)	0.584	0.775	0.668
Muón negativo (μ^-)	0.571	0.402	0.472
Pión negativo (π^-)	-	0	-
Pión positivo (π^+)	-	0	-

Tabla 8.5: Métricas luego de aplicar Regresión logística

se independientes unos de otros y este no es el caso. Además, es recomendable aplicar este algoritmo para casos de clasificación binaria y en este caso se tiene una clasificación de multiclases .

Electrón (e^-)	**Muón positivo (μ^+)**	**Protón (p)**	**Positrón (e^+)**	**Muón negativo (μ^-)**	**Pión negativo (π^-)**	**Pión positivo (π^+)**	
3737	70	0	955	102	0	0	**Electrón (e^-)**
58	11471	415	134	2412	0	0	**Muón positivo (μ^+)**
303	1071	1836	2953	10	0	0	**Protón (p)**
63	215	167	3139	467	0	0	**Positrón (e^+)**
401	5410	85	140	4054	0	0	**Muón negativo (μ^-)**
23	112	15	17	26	0	0	**Pión negativo (π^-)**
22	132	1	8	29	0	0	**Pión positivo (π^+)**

Tabla 8.6: Matriz de confusión al aplicar Regresión Logística

8.0.3. Random Forest

A continuación se muestran los resultados, luego de haber aplicado el algoritmo *Random Forest* al conjunto de datos. Como se conoce, este tipo de algoritmos tiende a sobre ajustar el modelo dado que solo es posible para el usuario modificar los hiperparámetros. En este caso, se decidió ajustar los hiperparámetros, esto es porque si se dejaba que el modelo se sobre-ajustara se podría obtener una precisión de hasta 99 %, sin embargo estos resultados no tendrían sentido común. La explicación a este argumento radica en el hecho de que un solo árbol de decisión crearía reglas que se basa en valores puntuales de voltaje para clasificar cada tipo de partícula, esto en la vida real no es así. Las partículas, al decaer en el tanque no se registrarán los mismos y exactos valores de voltaje por el PMT para cada tipo de partícula. Este es el caso que considera tan solo un árbol de decisión, y para *Random Forest* se tienen alrededor de cien árboles de decisión, lo cual en definitiva lleva al sobre-ajuste del modelo .

A continuación, se puede observar como la precisión es mejor que los métodos probados con anterioridad .

Clase	Precisión	Recall	F-1
Electrón (e^-)	0.845	1	0.916
Muón positivo (μ^+)	0.694	0.903	0.784
Protón (p)	0.990	0.730	0.840
Positrón (e^+)	0.952	0.869	0.909
Muón negativo (μ^-)	0.813	0.537	0.647
Pión negativo (π^-)	-	0	-
Pión positivo (π^+)	-	0	-

Tabla 8.7: Métricas luego de aplicar Random Forest

Otro aspecto importante a mencionar, es que al momento de aplicar este algoritmo, se modificó

con el fin de limitar la expresividad del mismo para obtener los siguientes resultados .

Electrón (e^-)	**Muón positivo** (μ^+)	**Protón** (p)	**Positrón** (e^+)	**Muón negativo** (μ^-)	**Pión negativo** (π^-)	**Pión positivo** (π^+)	
4864	0	0	0	0	0	0	**Electrón** (e^-)
175	13768	0	101	446	0	0	**Muón positivo** (μ^+)
14	1635	2517	0	7	0	0	**Protón** (p)
728	88	53	3182	0	0	0	**Positrón** (e^+)
430	7478	0	94	2088	0	0	**Muón negativo** (μ^-)
7	138	0	25	23	0	0	**Pión negativo** (π^-)
20	140	0	16	16	0	0	**Pión positivo** (π^+)

Tabla 8.8: Matriz de confusión luego de aplicar Random Forest

Nuevamente, el algoritmo falla en el momento de clasificar los piones, a pesar de que esta matriz tiene una forma más parecida a una matriz diagonal .

8.1. Clasificación de partículas por medio de una red neuronal

8.1.1. Construcción de la red neuronal

Para la construcción de la red neuronal fue necesario dividir los datos de las simulaciones obtenidos en tres conjuntos: uno de validación, prueba y entrenamiento, como se muestra a continuación:

A partir de la separación de los datos se decidió trabajar de la siguiente manera: el conjunto de prueba se utilizaría hasta que la red neuronal esté terminada para evaluar el rendimiento de la misma. Por otro lado, los conjuntos de entrenamiento y validación serán utilizados mientras se construye y entrena la red neuronal.

Para la construcción de la red neuronal se trabajó con Tensor Flow y Keras de Python. En cuanto a la preparación de los conjuntos de datos cabe mencionar que se categorizó el tipo de partícula de forma numérica, asignándole un número del cero al seis para cada partícula. De la siguiente manera:

Seguido de esto se le aplicó a los datos de voltaje una función para estandarizar estos atributos y escalarlos, para ello se utilizó la función $MinMaxScaler$ de la librería *sklearn* de Python. Esta función se encarga de transformar los atributos escalándolos en un rango entre $(0, 1)$.

Luego, para la red neuronal se utilizó un modelo secuencial el cual consta de un conjunto de capas donde cada capa tiene como entrada y salida un tensor. Tiene un total de 10 capas, donde 3 de ellas son de normalización, utilizando la capa *Batch Normalization* de Keras. Estas tres capas se encargan de mantener el promedio de los datos de salida cercano a 0 y la desviación estándar de los mismos cercano a 1.

La última capa de la red neuronal incluye una función de activación tipo *sigmoide*, básicamente aplica la función sigmoide a cada entrada. Cabe mencionar que los valores de salida están comprendidos en el intervalo de $(0, 1)$.

Para la compilación del modelo se utilizó un modelo de pérdida de entropía cruzada (para datos categóricos). La cual en este caso se encargó de hacer un promedio de la distribución de probabilidades entre la diferencia del valor real y el valor predicho .

Conjunto de datos	**Cantidad**	**Porcentaje**
Validación	54360	34.1 %
Entrenamiento	67007	42.0 %
Prueba	38053	23.9 %

Tabla 8.9: Separación de los datos de simulaciones en tres conjuntos.

Tipo de partícula	Número categórico
Positrón (e^+)	0
Muón positivo (μ^+)	1
Muón negativo (μ^-)	2
Electrón (e^-)	3
Pión positivo (pi^+)	4
Pión negativo (pi^-)	5
Protón (p)	6

Tabla 8.10: Identificadores categóricos de partículas para el procesamiento de datos

En cuanto al parámetro de optimización, se utilizó la clase *Adam* para esta tarea porque se ajusta bien para problemas con bastantes datos o parámetros. Cabe mencionar que dado que en un inicio la red neuronal no podía clasificar bien las partículas y no *estaba aprendiendo*, se le cambió la velocidad de aprendizaje por 1×10^{-4} .

Por último, para la inicialización del modelo y el ajuste del mismo, se decidió trabajar con un *batch size* de 100, puesto que se consideró que este número abarcaba la cantidad sufiente de datos. Asi también se realizó un total de 15 *epochs* .

8.1.2. Discusión de resultados

Luego de haber entrenado la red neuronal, se probó su rendimiento con el conjunto de datos de prueba. A continuación se muestran los resultados de la matriz de confusión:

Positrón (e^+)	**Muón positivo (μ^+)**	**Muón negativo (μ^-)**	**Electrón (e^-)**	**Pión positivo (π^+)**	**Pión negativo (π^-)**	**Protón (p)**	
5632	114	36	0	5	5	453	**Positrón (e^+)**
147	19432	3846	8	96	116	137	**Muón positivo (μ^+)**
12	455	10517	0	110	150	374	**Muón negativo (μ^-)**
58	84	30	6852	0	8	0	**Electrón (e^-)**
0	0	0	0	17	0	0	**Pión positivo (π^+)**
0	0	0	0	1	9	0	**Pión negativo (π^-)**
100	565	0	0	43	31	4917	**Protón (p)**

Tabla 8.11: Matriz de confusión de la red neuronal

Como se puede observar en 8.11, el algoritmo de clasificación implementado sí es capaz de clasificar piones. Otro aspecto importante a resaltar de estos resultados es que la matriz de confusión, en efecto cumple con que el elemento de la diagonal para cada tipo de partícula es el valor máxima de cada fila .

Como son de especial interés los muones, se puede ver que en efecto, son las partículas que el algoritmo es capaz de identificar en su mayoría .

Comparando los resultados de las métricas obtenidas en 8.12 con los resultados luego de aplicar Naive Bayes en 8.1 se tiene que hay un incremento en la precisión para la clasificación de partículas. Lo cual implica que la red neuronal es una mejoría para el análisis de partículas planteado en trabajos anteriores .

Haciendo referencia a la clasificación de muones, al comparar las figuras 8.12 y 8.7 se tiene que, para el algoritmo de *Random Forest* la precisión es mayor, teniendo un 0.813 comparado con un 0.749 de la red Neuronal lo cual es un indicador de que si el algoritmo clasifica una partícula como muón, hay mayor probabilidad de que en efecto sea muón. Sin embargo, al comparar el valor de *Recall* (el cual indica la fracción de clases que el modelo es capaz de identificar) para ambos algoritmos, se tiene que hay un valor mayor para la red neuronal en comparación a *Random Forest*. Este resultado implica que el algoritmo de *Random Forest*, deja pasar una mayor cantidad de muones y no los clasifica como tal .

Clase	Precisión	Recall	F-1
Positrón (e^+)	0.902	0.9467137334005715	0.923
Muón positivo (μ^+)	0.817	0.941	0.875
Muón negativo (μ^-)	0.905	0.729	0.808
Electrón (e^-)	0.974	0.100	0.989
Pión positivo (π^+)	1.0	0.0625	0.118
Pión negativo (π^-)	0.9	0.0282	0.0547
Protón (p)	0.869	0.836	0.852

Tabla 8.12: Métricas de la red neuronal

Un aspecto clave a mencionar es que la precisión para la red neuronal es mucho mejor para la clasificación de electrones y positrones, comparándolo con la precisión para el resto de las partículas.

CAPÍTULO 9

Clasificación de datos provenientes de Kinich Ahau

Uno de los objetivos de este proyecto fue habilitar una herramienta que pudiera ser aplicada a los datos recolectados por el detector que se encuentra en la Universidad del Valle.

A continuación se muestran los resultados obtenidos, luego de aplicar el modelo a los datos recolectados por el detector que se encuentra en el campus de la universidad . Cabe mencionar que para la clasificación de los datos obtenidos por el detector, también se les aplicó una función para escalar los datos y de esta manera la red neuronal los pudiera clasificar .

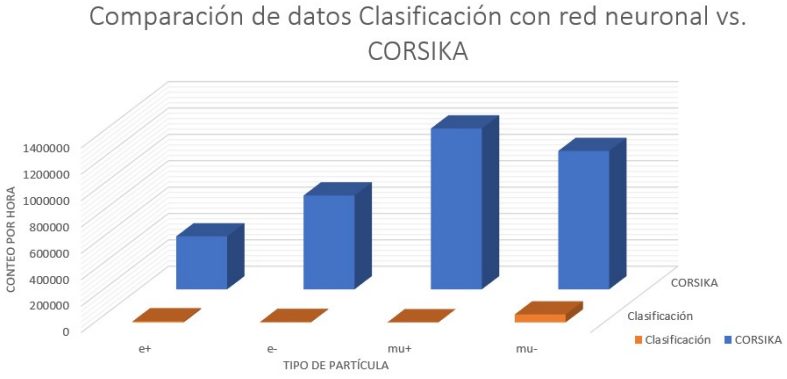


Figura 9.1: Modelo de clasificación aplicado a los datos del detector Kinich Ahau, comparado con la distribución de datos de CORSIKA

Al comparar con la distribución y conteo de partículas en 7.2 se puede observar que los datos no son directamente comparables dado que corresponden a diferentes períodos de medición. En la Figura 7.2 se tiene un período de cuatro horas, mientras que en la clasificación que se muestra en la Figura 9.1 se tiene un período de treinta minutos. Al calcular por medio de la herramienta que brinda Excel para el coeficiente de correlación, se obtiene el siguiente resultado:

En efecto, se comprueba que no hay una correlación fuerte entre la distribución de partículas

Coeficiente de correlación
0.274

Tabla 9.1: Coeficiente de correlación entre los dos conjuntos de datos

según la simulación de CORSIKA y los datos recolectados por el detector y que fueron clasificados por la red neuronal. Esto se debe a múltiples causas, entre ellas: los períodos de tiempo no son comparables, no se considera el voltaje de activación o voltaje de ánodo, no se considera la hora ni el momento del día en ambas simulaciones. Todas estas causas pueden influir en el hecho de que ambas clasificaciones sean o no comparables .

CAPÍTULO 10

Conclusiones

Con la construcción de la red neuronal se pudo realizar la clasificación de las partículas en base al pulso que generan luego de ser captadas por el PMT de un detector Cherenkov de agua. Esta red neuronal, cumple con que al ver la matriz de confusión el elemento de la diagonal es en efecto el valor máximo de cada fila. Otro aspecto importante a mencionar, es que a pesar de tener un conjunto de datos de entrenamiento (provenientes de la simulación) bastante desbalanceado, la red neuronal sí es capaz de detectar las partículas cuya presencia es mínima en el conjunto de datos.

Durante el desarrollo del proyecto, en base a los datos generados provenientes de la simulación de *KinichAhau* que contiene los parámetros del detector presente en la Universidad del Valle [5] se pudo entrenar la red neuronal sobre un conjunto de entrenamiento y luego pasó por dos etapas de validación, donde se evaluó en un conjunto de evaluación y finalmente sobre un conjunto de prueba que contenía datos que la red nunca antes había visto.

Como parte del proceso de aprendizaje y entrenamiento de la red neuronal, y según los resultados obtenidos se tiene que se pudo caracterizar los pulsos de voltaje correspondientes a cada tipo de partícula, entre las cuales estuvieron: muones, anti-muones, electrones, positrones, protones, piones y anti-piones.

Como resultado global, se logró afinar el procedimiento utilizado en años anteriores para la clasificación de partículas. En este caso se evaluó el rendimiento y la precisión de aplicar los algoritmos de clasificación en [5] así como también se consideró otro algoritmo bastante conocido de *Machine Learning* el cual fue *Random Forest* y al comparar estos tres algoritmos con la red neuronal, se obtuvieron mejores resultados de clasificación con la red neuronal. En cuanto a la precisión de la clasificación de muones, se obtuvo alrededor de un 20 % mejor que al aplicar Naive Bayes. Asimismo, se incluyen en anexos toda la documentación necesaria para poder replicar este procedimiento, se logró la compilación exitosa de la simulación creada en 2016 de [5] y se documentó la información necesaria para poder generar eventos en *KinichAhau*. También se creó un algoritmo en Python que permite la generación y procesamiento de n eventos, siendo n del tamaño que se guste, para la generación de histogramas de voltaje y tiempo para distintos tipos de partículas y energías. Este último aspecto cumple con uno de los objetivos específicos del trabajo, donde se planteó el deseo de tener un procedimiento repetible y aplicable a cualquier conjunto de datos, incluyendo los datos provenientes del detector de radiación Cherenkov de agua.

CAPÍTULO 11

Recomendaciones

Según los resultados obtenidos y al estudio del comportamiento de los datos se recomienda aplicar como método de clasificación una red neuronal recurrente de tipo LSTM (*Long Short Term Memory*) puesto que este tipo de red neuronal se aplica cuando se analiza un conjunto de datos que contiene series de tiempo y como los valores de voltaje, en definitiva, dependen del tiempo se considera que podría ser una oportunidad de mejora para el método de clasificación. En B se puede encontrar más información sobre las redes neuronales recurrentes y las de tipo LSTM.

Otra recomendación importante a considerar es el análisis de los piones, tanto pión positivo como su anti-partícula. En este proyecto la presencia de estas partículas en el conjunto de datos daba lugar a que se tuviera un conjunto desbalanceado de datos, lo cual afectó la precisión del método. Por lo mismo, se recomienda dedicarse en mejorar la clasificación categórica para clases desbalanceadas o bien, crear dos redes neuronales que trabajen en conjunto. Una de las redes neuronales se podría encargar de clasificar el resto de partículas y tener una clase de *otros*, donde se encuentren los piones y anti-piones. Luego, en base a esa categoría *otros* hacer una red neuronal que se encargue de hacer una clasificación binaria entre los dos tipos de piones.

Un aspecto que se propone como plan de mejora es incluir en el modelo de clasificación la energía con la que se simulan las partículas en Geant4, de esta manera al momento de aplicarlo a datos provenientes del detector aparte de conocer el tipo de partícula, también se podrá conocer la energía de la misma.

Durante el desarrollo de este proyecto y la búsqueda de literatura, se encontró que un aspecto interesante a evaluar y estudiar sería el de evaluar en base a la clasificación de partículas y pulsos, se podría estudiar el flujo de μ/EM en cuanto al *rise time*.

En cuanto a la colaboración que se tiene con la comunidad LAGO, dado que en el presente año se han realizado reuniones y grupos de trabajo donde se comienza a utilizar *Machine Learning* para el análisis de los datos provenientes de los detectores pertenecientes a la red, se recomienda que cuando el grupo de trabajo de tengan resultados, se podría evaluar el rendimiento de ambos algoritmos de clasificación. Actualmente están trabajando en una red neuronal convolucional y sería un buen análisi y aporte, comprar los resultados entre ambos métodos.

Un aspecto importante a mejorar y estudiar es el del PMT y el circuito planteado en la Figura 4.5, puesto que esta analogía es una generalización a los PMT. Se recomienda desarrollar un circuito particular para la simulación de Geant4 y también considerar el voltaje del ánodo de la electrónica

de LAGO en el circuito a desarrollar.
En cuanto a la comparación de los datos obtenidos de la distribución de CORSIKA y los datos clasificados por medio de la red neuronal, se recomienda abrir archivos de un tiempo más corto (por ejemplo, media hora) y calcular la varianza y la correlación entre los datos.

Por último, se recomienda obtener datos de otros detectores ubicados en otros países y a partir de estos datos, probar el rendimiento del método de clasificación. De esta manera, se puede realizar un aporte no solo para la Universidad del Valle de Guatemala, sino para las otras instituciones educativas que se deseen involucrar en esta contribución, puesto que se considera que la ciencia debería de trascender fronteras.

Bibliografía

[1] Alpaydin, E.: *Introduction to machine learning.* The MIT Press, 2004.

[2] Asorey, Hernan: *Los Detectores Chérenkov del Obsevatorio Pierre Auger y su Aplicación al Estudio de Fondos de Radiación.* Tesis de Doctorado, 2012.

[3] Brownlee, Jason: *Long Short - Term Memory Networks With Python: Develop Sequence Prediction Models with Deeep Learning.* Jason Brownlee, 2019.

[4] CERN: *Geant4: A simulation toolkit.* `https://geant4.web.cern.ch/`.

[5] Daniel Conde, Pablo Duque, Karen Guarcaz et al: *Diseño, construcción y caracterización de Detector de Radiación Vavilov-Cherenkov de Agua.* Universidad del Valle de Guatemala, 2016.

[6] Durán, Mauricio Suárez: *Instalación de un Detector Cherenkov de Agua para la detección de trazas de Rayos cósmicos a 956 metros sobre el nivel del mar.* Bucamaranga, 2010.

[7] Fries, D.: *New Phenomena in Lepton - Hadron Physics.* Springer, 1979.

[8] Gaisser, Thomas: *Cosmic Rays and Particle Physics.* Cambridge University Press, 1999.

[9] Gasiorowicz, S. y P. Langacker: *ELEMENTARY PARTICLES IN PHYSICS 1 Elementary Particles in Physics.*

[10] Goverment, Australian: *Alpha Particles.* `https://www.arpansa.gov.au/understanding-radiation/what-is-radiation/ionising-radiation/alpha-particles`.

[11] Graupe, Daniel: *Principles of Artificial Neural Networks: Basic Designs to Deep Learning.* World Scientific, 2019.

[12] Kostadinov, Simeon: *Recurrent Neural Networks with Python: Quick start guide.* Packt, 2018.

[13] LAGO: *LAGO.* `http://lagoproject.net/`.

[14] León Ardón, Rodrigo de: *Informe final: Simulación de un detector Cherenkov de Agua.* Universidad de San Carlos de Guatemala, 2012.

[15] Neil Ashby, *et all*: *Introduction to Particle and Astroparticle Physics: Multimessenger Astronomy and its Particle Physics Foundations.* Springer, 2018.

[16] Nijenhuis, Nadjezjda: *Characterization of the ANTARES photomultiplier R 7081-20.* Universiteit van Amsterdam, 2002.

[17] Nilsson, N., *et all*: *Introduction to machine learning: An early draft of a proposed textbook.* Standford, 2005.

[18] Nuclear Physics, Institute for: *CORSIKA.* `https://www.ikp.kit.edu/corsika/index.php`.

[19] Obodovskiy, Ilya: *Radiation: Fundamentals, Applications, Risks and Safety.* Amigos del País, Guatemala., 2004.

[20] País, Asociación de Amigos del: *Diccionario Histórico Biográfico de Guatemala.* Amigos del País, Guatemala., 2004.

[21] Plata, Universidad La: *Muon experiments@UNLP.* `http://www2.fisica.unlp.edu.ar/~veiga/index.html`.

[22] S.A: *TensorFlow.* `https://www.tensorflow.org/?hl=es-419`.

[23] Trappenberg, Thomas: *Fundamentals of Machine Learning.* Oxford University Press, 2020.

ANEXO A

Guía de simulación Kinich Ahau - Geant 4

A.1. Descripción de KinichAhau

Esta simulación se asemeja al detector ubicado en el campus de la Universidad. A continuación la descripción de los objetos de la simulación:

Nombre del objeto	**Material**	**Tipo de objeto**	**Dimensiones**
Tyvek	Tyvek	G4Cons	Altura = 57.1cm, Radio=40.1m
Agua	Agua	G4Cons	Altura = 57.1cm, Radio=40m
PMT	Borosilicato	G4Sphere	Radio=11.1 cm
Fotocátodo del PMT	Aluminio	G4Sphere	Radio = 11.1cm
Vacío del tubo de PMT	Vacío	G4SPhere	Radio= 11 cm

Tabla A.1: Objetos de la simulación de Kinich Ahau

Como se puede observar, la simulación incluye los componentes principales de un tubo PMT, así como también cumple con ser un cilindro y estar recubierto del material Tyvek para que las paredes sean reflectoras.

A.2. Compilación de KinichAhau

Dado que esta era una simulación que ya se había realizado en años anteriores, el único paso necesario para la generación de eventos fue la compilación exitosa de la simulación.
Para ello fue necesario haber instalado de forma exitosa Geant4, siguiendo la documentación y tomando los recursos disponibles en la página: *https* : *//geant4.web.cern.ch/* [4]. Luego, se deben ejecutar los siguientes comandos para compilar la simulación en una computadora:

- Se debe copiar la carpeta referente a la simulación del repositorio disponible en: *https* : *//github.com/GrayCygnus/LAGO − UVG*
- Crear una carpeta para çonstruir la simulación", de preferencia a la par de la carpeta que contenga todos los archivos de KinichAhau (Sugerencia: la carpeta se suele nombrar *Build*, siguiendo la documentación de Gean4).
- Acceder a través de la consola a la carpeta *Build* creada con anterioridad.
- Ejecutar el siguiente comando: çmake C:(Directorio) -DCMAKE_ INSTALL_PREFIX= C:(Directorio) (donde el primer directorio hace referencia a la carpeta donde se encuentra el archivo *Geant4config.cmake* en la carpeta de instalación de Geant4 y el segundo directorio hace referencia a la carpeta de la simulación *Kinich Ahau*.

Seguido de esto, en la carpeta *Build*, se escribirán los archivos relacionados con la compilación de la simulación, y bajo la carpeta *bin* se podrá encontrar un ejecutable el cual es el que se utilizará para la simulación de un evento.

A.3. Simulación de un evento

Habiendo compilado exitosamente la simulación en la computadora, se puede proceder con la simulación de eventos. Cabe mencionar que a cada evento se le pueden asignar los siguientes parámetros: tipo de partícula a lanzar, número de lanzamientos de partícula, energía de la partícula, ángulo de incidencia y dirección de la partícula. Es importante que los parámetros deben ser incluidos en el archivo que se crea bajo el nombre: *KinichAhau.in*, este archivo incluye las instrucciones de ejecución de la simulación. A continuación una breve explicación de los comandos que se pueden utilizar,

Comando	**Explicación**
/gun/particle <particle$_name$ >	Indica la partícula a lanzar
/gun/energy <val> <unit>	Hace referencia a la energía con la que será lanzada la partícula, las unidades están en eV
/gun/direction <ex> <ey> <ez>	Es un vector unitario que indica la dirección en la cual será lanzada la partícula
/gun/position <x> <y> <z> <unit>	Indica el punto desde el cual será lanzada la partícula.

Tabla A.2: Comandos básicos para las simulaciones

Habiendo configurado las instrucciones de la simulación, se procede a utilizar nuevamente la consola donde se accede a la carpeta en la que se encuentra el archivo ejecutable de KinichAhau, previamente generado en la compilación de la simulación. Luego, para crear un evento se debe de ejecutar el siguiente comando:

KinichAhau.exe -m KinichAhau.in > evento.csv

Cabe mencionar que el tercer parámetro se puede dejar en blanco y en la consola se imprimirán los resultados de la simulación. Por otro lado, los resultados pueden ser trasladados al tipo de archivo

que se desee, en este caso, por motivos de facilidad para el análisis de datos, se escriben en un archivo de extensión *.csv*.

A.3.1. Datos de salida

La simulación de Kinich Ahau brinda como salida la cantidad de fotones Cherenkov que logra captar el PMT y su tiempo de llegada al mismo. A partir de estos, se pueden generar histogramas para conocer la corriente generada por evento.

A continuación se muestra la salida desde la consola del *cmd*:

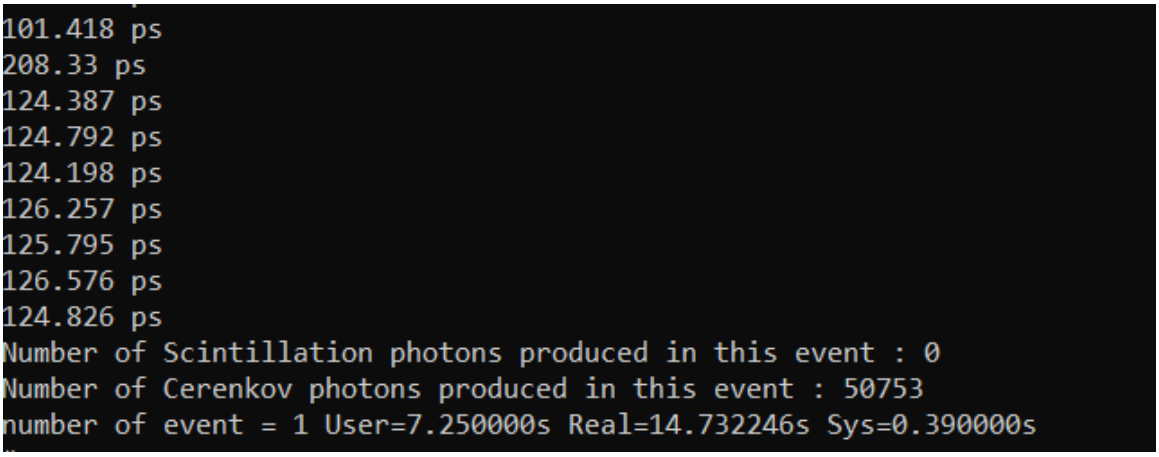


Figura A.1: Últimas líneas de la salida de la simulacion

Como se puede observar, al final se muestran la cantidad de Fotones Cherenkov capturados, el número de eventos generados. Así también se muestran los últimos tiempos de llegada para fotones al PMT en *ps*.

ANEXO B

Redes Neuronales Recurrentes: LSTM

B.1. Redes Neuronales Recurrentes: RNN

Es un modelo desarrollado a partir de *Deep Learning*; se basa en el desarrolllo de predicciones de datos secuenciales y se basa en una arquitectura computacional de retención de memoria. En este caso, se basa en el fundamento que, para realizar una predicción, almacena la información de la entrada anterior, y de esta manera el resultado no depende solamente de la entrada actual sino que también de la entrada anterior. En su manera más sencilla, una red neuronal recurrente funciona de la siguiente manera:

- Obtener los datos de entrenamiento.
- Se codifican los datos: se recomienda una representación numérica dado que se trabaja con un enfoque numérico.
- Construir la estructura del modelo.
- Entrenar el modelo hasta que el usuario esté satisfecho con los resultados.
- Se evalúa el modelo con datos que no se habían probado antes, para hacer una predicción nueva.

[12]

B.1.1. Long Short Term Memory: LSTM

Este es un tipo de RNN, donde se tiene el mismo caso que en una red neuronal recurrente donde se utilizan datos secuenciales y conexiones recurrentes. Este tipo de red neuronal surge como solución

al problema que presentan las RNN en que pueden no entrenarse de forma efectiva.
El funcionamiento de una red neuronal de este tipo se basa en diversas capas creadas por conexiones recurrentes de bloques. Cada una de las capas contiene çeldas de memoria", que contienen tres unidades: la celda de entrada, la salida y la celda de olvido, lo cual permite las operaciones de: escribir, leer y reiniciar la operación en cada una de las celdas.
Entra las aplicaciones más destacadas de este tipo de redes neuronales se encuentra para datos secuenciales, como por ejemplo la traducción automática de texto y el texto predictivo [3].

ANEXO C

Algoritmos implementados en Python

C.1. Procesamiento y generación de datos

El siguiente algoritmo se utilizó para generar los datos provenientes de la simulación hecha en Geant 4.

Listing C.1: KAGenerator.py

```
# -*- coding: utf-8 -*-
"""
Created on Tue Sep  8 18:42:15 2020

Recibe de entrada un archivo con lista de particulas a generar, y produce
un archivo .in para ser utilizado en KinichAhau

El formato del archivo de entrada es un CSV que utiliza ; como separador,
con header y en cada fila hay 3 columnas: id - un identificador unico (numero entero),
energia - valor en GeV de la energ a de la part cula (punto flotante),
y tipo de particula (una cadena de caracteres)

salida un archivo por cada fila con el formato:

/control/verbose 2
/tracking/verbose 0
#
/gun/particle [particula]
/gun/energy [energia] keV
#
/KinichAhau/phys/verbose 0
#
/run/beamOn 1
#

donde [particula] y [energia] son valores que salen de cada fila del archivo
de entrada.

El programa tambien recibe como entrada la ubicacion del archivo y la ubicacion
de donde debe grabar las salidas. Opcionalmente puede recibir dos parametros mas:
el id de inicio (inclusivo) e id de parada (no inclusivo) si se quiere generar un rango de ids nada
```

```
@author: jmanc
"""


import sys, os
import pandas as pd
import numpy as np
from scipy import integrate
import matplotlib.pyplot as plt
import seaborn as sns

def createInFile(particle,energy):
    result = "/control/verbose 2\n/tracking/verbose 0\n#\n/gun/particle "
    result += particle
    result += "\n/gun/energy "
    result += str(energy) + " keV"
    result += "\n#\n/KinichAhau/phys/verbose 0\n#\n/run/beamOn 1\n#\n"
    return result

def getHistogram(evento):
    data = pd.read_csv(evento,sep=',',header=None)
    #data.hist(bins=12,grid=True)
    #plt.show()
    #ax = sns.distplot(data)
    #np.histogram(data)
    x=np.histogram(data,bins=12)
    I=x[0]
    tiempos=x[1]
    #print(I)
    #print(tiempos)
    R=50.0
    C=4.7e-12
    to=R*C
    def e(x):
        return np.exp(x/to)
    Voltaje=[]
    ans=0.0
    for i in range(len(I)):
        ans=ans+I[i]*e(tiempos[i])
        Voltaje.append(ans)
    voltage = []
    A = 1e-15
    for j in range(len(Voltaje)):
        ans2=(np.exp(-tiempos[j]/to)/C*Voltaje[j])*A
        voltage.append(ans2)
    #print("\n"+str(voltage))
    #plt.plot(tiempos[:-1],voltage)
    #plt.title("Voltaje vs t")
    #plt.xlabel("t")
    #plt.ylabel("Voltaje")
    #plt.show()
    return voltage

def main(argv):
    if len(argv) < 2:
        print ("Usage: KA_Generator.py archivo_de_entrada ubicacion_de_salida [inicio] [fin]")
        return -1
    ifile = argv[0]
    ofile = argv[1]
    if len(argv) >= 3: inicio = int(argv[2])
    else: inicio = -1
    if len(argv) >= 4: finish = int(argv[3])
    else: finish = -1
    i = 0
    generated = 0
    salida = open(ofile+"resultado.csv","w")
    salida.write("id,v1,v2,v3,v4,v5,v6,v7,v8,v9,v10,v11,v12,particle\n")
```

```
    with open(ifile) as f:
        for line in f:
            if i > 0:
                (num,energy,particle) = line.strip().split(';')
                if ((inicio < 0) or (i >= inicio)) and ((finish < 0) or (i < finish)):
                    generated += 1
                    # Create in file
                    inFileText = createInFile(particle,float(energy)*1000000)
                    of = open(ofile+"KinichAhau.in","w")
                    x = of.write(inFileText)
                    of.close()

                    #Run KinichAhau in in file
                    comando = "KinichAhau.exe -m " + ofile + "KinichAhau.in > " + ofile
                    \ + "evento.csv"
                    os.system('cmd /c "'+comando+'"')

                    #Process output of KinichAhau
                    ef = open(ofile+"evento.csv")
                    lines = ef.readlines()
                    startFound = False
                    endFound= False
                    values = []
                    for line in lines:
                        if not startFound:
                            if line=="### Run 0 start.\n":
                                startFound = True
                        elif not endFound:
                            if "Number" in line:
                                endFound = True
                            else:
                                values.append(line.strip())
                    if len(values)<=1:
                        continue
                    of = open(ofile+"salida.csv","w")
                    for v in values:
                        n,p = v.split(" ")
                        e = -1
                        if p == "ns": e = 1e-9
                        if p == "ps": e = 1e-12
                        if e == -1: print("ERROR! exponente no reconocido " + p)
                        val = float(n)*e
                        x = of.write(str(val)+"\n")
                    of.close()

                    #Calcula el histograma utilizando el codigo de jupyter notebook
                    result = getHistogram(ofile+"salida.csv")
                    salida.write(str(num)+",")
                    for r in result:
                        salida.write(str(r)+",")
                    salida.write(particle+"\n")

                    if (i%1000==0):
                        print(".",end="")
            i+=1
    salida.close()
    print("\nGenerado " + str(generated) + " histogramas en " + ofile +
          "resultado.csv a partir de " + ifile)
    return 0;

if __name__ == "__main__":
    main(sys.argv[1:])
```

C.2. Algoritmo de red neuronal

Listing C.2: Algoritmo para la construcción de la red neuronal

```
# -*- coding: utf-8 -*-
"""
Created on Fri Sep 25 09:53:32 2020

@author: jmanc & MayraSilva
"""

import tensorflow.keras as keras #Fundamental para la
utilizacion de Machine learning con Python
from keras.utils import to_categorical
from tensorflow.keras.layers import Dense, Flatten, Dropout, LSTM
from tensorflow.keras.models import Sequential
from tensorflow.keras import layers
from tensorflow.keras import regularizers
from tensorflow.keras import activations
import matplotlib.pylab as plt
import pandas as pd #Para el procesamiento de datos
import numpy as np
from sklearn.preprocessing import MinMaxScaler
#Librer a necesaria para la estandarizaci n de datos



scaler = MinMaxScaler()
useLstm = False

## Funci n para obtener las m tricas
def getStats(real,predicted):
    numClasses = real.shape[1]
    stats = []
    for i in range(0,numClasses):
        stats.append([])
        for j in range(0,numClasses):
            stats[i].append(0)
    for i in range(0,len(predicted)):
        p = np.argmax(predicted[i])
        r = np.argmax(real[i])
        stats[p][r]+=1
    metrics = {}
    for i in range(0,numClasses):
        tp = stats[i][i]
        fp = 0
        fn = 0
        for j in range(0,numClasses):
            if (i!=j):
                fp += stats[i][j]
                fn += stats[j][i]
        precision=-1
        if ((tp+fp)>0):
            precision=tp/(tp+fp)
        recall=-1
        if ((tp+fn)>0):
            recall=tp/(tp+fn)
        f1=-1
        if ((precision+recall)>0):
            f1=2*(precision*recall)/(precision + recall)
        metrics[i]= {'precision':precision,
                     'recall': recall,
                     'f1':f1 }
    return stats,metrics


#Visualizaci n de las m tricas
def printStats(stats):
    cm,m=stats
```

```
    print("Confusion Matrix")
    print("Real\Prediction\te+\tmu+\tmu-+\te-+\tpi++\tpi-+\tproton")
    print("e+\t"+str(cm[0][0])+"\t"+str(cm[0][1])+"\t"+str(cm[0][2])+"\t
    "+str(cm[0][3])+"\t"+str(cm[0][4])+"\t"+str(cm[0][5])+"\t"+str(cm[0][6]))
    print("mu+\t"+str(cm[1][0])+"\t"+str(cm[1][1])+"\t"+str(cm[1][2])+"\
    t"+str(cm[1][3])+"\t"+str(cm[1][4])+"\t"+str(cm[1][5])+"\t"+str(cm[
    1][6]))
    print("mu-\t"+str(cm[2][0])+"\t"+str(cm[2][1])+"\t"+str(cm[2][2])+"\
    t"+str(cm[2][3])+"\t"+str(cm[2][4])+"\t"+str(cm[2][5])+"\t"+str(cm[
    2][6]))
    print("e-\t"+str(cm[3][0])+"\t"+str(cm[3][1])+"\t"+str(cm[3][3])+"\t
    "+str(cm[3][3])+"\t"+str(cm[3][4])+"\t"+str(cm[3][5])+"\t"+str(cm[3
    ][6]))
    print("pi+\t"+str(cm[4][0])+"\t"+str(cm[4][1])+"\t"+str(cm[4][4])+"\
    t"+str(cm[4][3])+"\t"+str(cm[4][4])+"\t"+str(cm[4][5])+"\t"+str(cm[
    4][6]))
    print("pi-\t"+str(cm[5][0])+"\t"+str(cm[5][1])+"\t"+str(cm[5][5])+"\
    t"+str(cm[5][3])+"\t"+str(cm[5][4])+"\t"+str(cm[5][5])+"\t"+str(cm[
    6][6]))
    print("proton\t"+str(cm[6][0])+"\t"+str(cm[6][1])+"\t"+str(cm[6][6])
    +"\t"+str(cm[6][3])+"\t"+str(cm[6][4])+"\t"+str(cm[6][5])+"\t"+str(cm[6][6]))
    print("")
    print("Metrics")
    print("Particle\tPrecision\tRecall\tF-1")
    print("e+\t"+str(m[0]['precision'])+"\t"+str(m[0]['recall'])+"\t"+st
    r(m[0]["f1"]))
    print("mu+\t"+str(m[1]['precision'])+"\t"+str(m[1]['recall'])+"\t"+s
    tr(m[1]["f1"]))
    print("mu-\t"+str(m[2]['precision'])+"\t"+str(m[2]['recall'])+"\t"+s
    tr(m[2]["f1"]))
    print("e-\t"+str(m[3]['precision'])+"\t"+str(m[3]['recall'])+"\t"+st
    r(m[3]["f1"]))
    print("pi+\t"+str(m[4]['precision'])+"\t"+str(m[4]['recall'])+"\t"+s
    tr(m[4]["f1"]))
    print("pi-\t"+str(m[5]['precision'])+"\t"+str(m[5]['recall'])+"\t"+s
    tr(m[5]["f1"]))
    print("proton\t"+str(m[6]['precision'])+"\t"+str(m[6]['recall'])+"\t
    "+str(m[6]["f1"]))

def loadData(filename):
    labelCodes = {'e+': 0, 'mu+': 1, 'mu-': 2, 'e-': 3, 'pi+': 4, 'pi-':5, 'proton': 6}
    trainingFile = pd.read_csv(filename, delimiter=";")
    x = trainingFile[["v1","v2","v3","v4","v5","v6","v7","v8","v9","v10","v11","v12"]]
    y = trainingFile["particle"]
    particlesInSet = list(set(y.tolist()))
    for particleName in particlesInSet:
        y = y.replace([particleName],labelCodes[particleName])
    n = x.shape[0]
    timesteps = x.shape[1]
    x = x.to_numpy()
    x = scaler.fit_transform(x)
    y = y.to_numpy()
    if (useLstm):
        x = x.reshape((n,timesteps,-1))
    y = keras.utils.to_categorical(y)
    return (x,y)



(train_x, train_y) = loadData('C:/Users/Home/Documents/Tesis
/LagoUVG/particleClassifier/trainingFin.csv')
(validation_x, validation_y) = loadData('C:/Users/Home/Documents/Tesis/LagoUVG
/particleClassifier/validationFin.csv')
(test_x, test_y) = loadData('C:/Users/Home/Documents/Tesis/LagoUVG
/particleClassifier/testFin.csv')
```

```
#Definicion de red LSTM
model = Sequential()

if (useLstm):
    #Esta capa descubre los features
    model.add(LSTM(units=12,
                   input_shape=(12,1),
                   unroll=True,
                   bias_regularizer=regularizers.l1_l2(l1=1e-4, l2=1e-5),
                   activation=None))
    model.add(keras.layers.LeakyReLU(0.3))


#Estas capas ya se encargan de la clasificacion
model.add(Dense(12))
model.add(keras.layers.BatchNormalization())
model.add(Dense(120))
model.add(Dropout(0.1))
model.add(keras.layers.BatchNormalization())
model.add(keras.layers.LeakyReLU(0.2))
model.add(Dense(120))
model.add(keras.layers.BatchNormalization())
model.add(keras.layers.LeakyReLU(0.3))
model.add(Dense(7,activation='sigmoid'))

model.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=keras.optimizers.Adam(learning_rate=1e-4,amsgrad=True),
              metrics=['accuracy'])



#Overfitting if: training loss << validation loss

#underfitting if: training loss >> validation loss

history=model.fit(train_x, train_y, validation_data=(test_x,test_y), batch_size=100, epochs=15)
results = model.predict(validation_x)
model.summary()

#Graficas para visualizar accuracy
plt.plot(history.history["accuracy"])

plt.plot(history.history["val_accuracy"])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper right')
plt.show()
#Graficas para visualizar perdida
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()

confusionMatriz, metricas = getStats(validation_y,results)
printStats((confusionMatriz, metricas))
```

C.3. Algoritmo de colocación de etiquetas a datos reales

Este código se corrió desde la consola de Python, luego de tener la red neuronal entrenada

Listing C.3: Algoritmo para la colocación de etiquetas a datos reales

```
#Se lee el archivo de datos reales, sin etiquetas
datosDetector=pd.read_csv("DatosLuisv2.csv", sep=';', header= None)
#Se convierten a Numpy
datosDetector=datosDetector.to_numpy()
#Estandarizaci n de datos
datosDetector = scaler.fit_transform(datosDetector)

resultadosDetector=model.predict(datosDetector)
 particulas=[]

for i in range(len(resultadosDetector)):
    line=resultadosDetector[i]
    value=str(np.argmax(line))
    particula=""
    if value=="0":
        particula="e+"
    if value=="1":
        particula="mu+"
    if value=="2":
        particula="mu-"
    if value=="3":
        particula="e-"
    if value=="4":
        particula="pi+"
    if value=="5":
        particula="pi-"
    if value=="6":
        particula="p"
    particulas.append(particula)
```

Glosario

K-Means Es un método de agrupamiento de datos, utilizado en *Machine Learning* que realiza partición de conjuntos en k grupos, donde cada observación que se encuentra en un grupo, tiene un valor medio cercano al de cada grupo [1].. 3

Keras Es una biblioteca de redes neuronales de código abierto habilitada en Python [23].. 24

Leptones Es una partícula cargada que no interactúa con las fuerzas fuertes, no tiene una estructura interna y sus interacciones electromagnéticas se pueden describir por las leyes de la Teoría electromagnética cuántica [7].. 4

Partículas alfa Partículas compuestas por dos protones y dos neutrones, son el producto de un decaimiento radioactivo (conocido como el decaimiento alfa)[10].. 6

Quarks Fermiones elementales que se caracterizan por su interacción fuerte, dando lugar a la formación de materia nuclear y hadrones [15]. 4

sabores De acuerdo al modelo estándar de física de partículas, el sabor es el atributo que permite la distinción entre los 6 quarks: *up, down, strange, charm, bottom, top*.. 4

Tensor Flow Es una biblioteca dedicada al aprendizaje automático, es de código abierto y contiene funciones esenciales para la construcción y entrenamiento de redes neuronales[22].. 24

Lista de símbolos

μ^+	Muón positivo
μ^-	Muón
π^+	Pión positivo
π^-	Pión
c	Velocidad de la luz
e^+	Positrón
e^-	Electrón
N_A	Número de Avogadro
p	Protón