UNIVERSIDAD MICHOACANA DE SAN NICOLÁS DE HIDALGO

INSTITUTO DE FÍSICA Y MATEMÁTICAS

APLICACIÓN DE REDES NEURONALES CONVOLUCIONALES A LA DISCRIMINACIÓN GAMMA HADRÓN EN CHUBASCOS ATMOSFÉRICOS

T E S I S

QUE PARA OBTENER EL GRADO DE:

MAESTRO EN CIENCIAS EN EL AREA DE FÍSICA

PRESENTA:

ARMANDO MADRIGAL LUCATERO

DIRECTOR DE TESIS:

DR. JOSE ANTONIO GONZALEZ CERVERA

CODIRECTOR:

DR. CEDERIK LEÓN DE LEÓN ACUÑA

MORELIA MICH. AGOSTO DE 2021

Esta tesis la dedico con mucho cariño a mis padres:

El Sr. Javier Madrigal y la Sra. Paula Lucatero.

AGRADECIMIENTOS

En primer lugar, quiero agradecer a mi asesor el Dr. José Antonio González Cervera quien me apoyó y me guió en la realización de este proyecto para poder alcanzar los resultados deseados.

Agradezco también al Dr. Cederik León de León Acuña y al Dr. Umberto Cotti Gollini por su ayuda y colaboración en este trabajo.

Por último agradezco al Instituto de Física y Matemáticas y a la Universidad Michoacana de San Nicolás de Hidalgo por la oportunidad que me brindaron de cursar mis estudios de maestría en esta institución.

Morelia, Michoacán, México, 16 de agosto de 2021

RESUMEN

Las observaciones de rayos gamma de alta energía provenientes del espacio exterior, se basan en experimentos en tierra que permiten detectar partículas secundarias de chubascos atmosféricos (EAS, por sus siglas en inglés) inducidos por rayos cósmicos de alta energía. Sin embargo, el obstáculo en la observación de fuentes de rayos gamma, es el gran fondo de rayos cósmicos de origen hadrónico. Es por ello la importancia de poder identificar cuales EAS son inducidos por rayos gamma y cuales por hadrones.

En este trabajo se propone un modelo computacional de inteligencia artificial basado en redes neuronales convolucionales (RNC) para la discriminación de EAS inducidos por rayos gamma o hadrones (partículas primarias), a partir de su huella al piso, es decir, en base a las partículas secundarias observadas en tierra. Se aborda este problema como un problema clasificación en el contexto del aprendizaje supervisado. Se proponen dos casos de estudio: la discriminación entre el tipo de partícula primaria (gamma o hadrón) y su energía a partir del EAS inducido.

Los datos de entrenamiento para el modelo de aprendizaje supervisado fueron producidos a partir de simulaciones numéricas de EAS generadas con el simulador de rayos cósmicos CORSIKA (COsmic Ray SImulations for KAscade). Se consideraron EAS verticales ((c)zenith $\theta = 0°, ac(z)imuth\phi = 0°$), inducidos por rayos gamma y protones (hadrones) con rangos de energía entre 0.5 y 30 TeV.

Se muestra que las RNC son un modelo con una alta eficiencia en la discriminación de EAS

inducidos por rayos gamma o hadrones.

Palabras clave: redes neuronales convolucionales, discriminación gamma-hadrón, chubascos atmosféricos, CORSIKA, aprendizaje supervisado.

ABSTRACT

High-energy gamma-ray observations from outer space are based on ground-based experiments that detect secondary particles of extensive air showers (EAS) induced by high-energy cosmic rays. However, the obstacle in observing gamma-ray sources is the large background of cosmic rays of hadronic origin. It is therefore important to be able to identify which EAS are induced by gamma rays and which by hadrons.

This paper proposes a computational model of artificial intelligence based on convolutional neural networks (CNN) for discrimination of EAS induced by gamma rays or hadrons (primary particles), from their footprint to the ground, that is, based on the secondary particles observed on the ground. This problem is addressed as a classification problem in the context of supervised learning. Two cases of study are proposed: the discrimination between the type of primary particle (gamma or hadron) and its energy from induced EAS. Training data for the supervised learning model were produced from numerical simulations of EAS generated with the cosmic ray simulator CORSIKA (COsmic Ray SImulations for KAscade). Vertical EAS ($\theta = 0°, \phi = 0°$), induced by gamma rays and protons (hadrons) with energy ranges between 0.5 and 30 TeV, were considered. It is shown that CNN is a model with a high efficiency in the discrimination of EAS induced by gamma rays or hadrons.

Keywords: convolutional neural networks, gamma-hadron discrimination, extensive air showers, CORSIKA, supervised learning.

Índice general

Capítulo 1

Introducción

Los rayos cósmicos (RC) están compuestos por diferentes tipos de partículas procedentes del espacio exterior cuya energía es muy elevada (superior a los 0.3 GeV). Los RC están compuestos principalmente por protones (86 %), partículas alfa (11 %), electrones (2 %) y núcleos pesados como el uranio (1 %) [1]. Cuando los RC interactúan con la atmósfera terrestre se producen chubascos atmosféricos (EAS, extensive air showers). Los EAS están constituidos por diferentes tipos de procesos muy complejos, los cuales involucran interacciones fundamentales de la física de partículas, la física de RC y procesos astrofísicos [2]. Un tipo particular de EAS son los inducidos por rayos gamma los cuales generan un gran interés de estudio en la física de altas energías y la astrofísica, debido a su importancia en el estudio de fuentes de rayos gamma. Sin embargo, el obstáculo en la observación de fuentes de rayos gamma, es el gran fondo de RC de origen hadrónico. El número de eventos inducidos por hadrones que se registran, es varios ordenes de magnitud más grande que los eventos inducidos por rayos gamma. Por tal motivo, los métodos de separación gamma-hadrón juegan un papel crucial en el análisis de datos. En el presente trabajo se desarrolla un método de separación gamma-hadrón usando un modelo de aprendizaje automático.

El aprendizaje automático es la rama de la inteligencia artificial (IA) cuyo objetivo es desarrollar técnicas que permitan que las computadoras aprendan, sin ser programadas explícitamente, es decir, un modelo de aprendizaje automático es capaz de convertir una muestra de datos en un programa informático capaz de extraer inferencias de nuevos conjuntos de datos para los que no ha sido entrenado previamente. El aprendizaje automático ha tenido un gran avance en los últimos años, gracias al creciente interés en desarrollar técnicas capaces de estructurar y obtener información de grandes cantidades de datos, así como a la capacidad de cómputo que se ha desarrollado en la actualidad en los sistemas computacionales. Dentro de la variedad de técnicas que existen en el aprendizaje automático están las redes neuronales artificiales (RNA), que surgieron como una técnica vagamente inspirada en el funcionamiento del cerebro humano. Actualmente muchas de estas técnicas son utilizadas para resolver problemas dentro del ámbito científico, particularmente en la física. Existen varios tipos (arquitecturas) de RNA que se enfocan en resolver diferentes tipos de problemas. Un tipo particular de RNA son las redes neuronales convolucionales (RNC), las cuales son útiles en problemas de análisis de señales y visión artificial. En el presente trabajo desarrollamos un método basado en RNC dedicado a la separación de EAS inducidos por rayos gamma o hadrones [3]. La fenomenología de producción de EAS fue generada con simulaciones numéricas por computadora usando el simulador CORSIKA (COsmic Ray SImulations for KAscade), el cual es un programa de simulación detallada de EAS inducidos por partículas de rayos cósmicos de alta energía [4].

En los siguientes capítulos, se estudian las técnicas de RNC para posteriormente ser aplicadas al problema de la discriminacion gamma hadron a partir de los EAS inducidos.

En el capítulo 2 se discute la ubicación de las RNC dentro del campo del aprendizaje automático y la IA. Se discuten los fundamentos del funcionamiento de las RNC y el tipo de problemas que permiten resolver, así como la descripción de su implementación y entrenamiento basado en fundamentos matemáticos.

En el capítulo 3 se expone la generación de simulaciones de EAS usando el simulador CORSIKA, así como los detalles de los datos producidos en las simulaciones.

En el capítulo 4 se plantea el problema de discriminación gamma-hadrón como un problema que puede ser abordado con técnicas de aprendizaje automático en particular usando RNC. Se abordan dos problemas complementarios, el primero consiste en separación gamma-hadrón y en el segundo la separación de las partículas en intervalos de energía. Posteriormente se presentan los resultados obtenidos para cada problema planteado.

Por último, en el capítulo 5 se presentan las conclusiones obtenidas en la realización de este trabajo además del posible trabajo futuro que se puede realizar en la misma dirección.

Capítulo 2

Redes Neuronales Artificiales

Las redes neuronales artificiales (RNA) constituyen uno de los modelos computacionales más importantes dentro del campo del aprendizaje automático y la inteligencia artificial (IA) debido a sus amplias aplicaciones en diversas disciplinas y al gran desarrollo que estas han tenido en los últimos años. Las RNA están inspiradas vagamente en el funcionamiento del cerebro humano, sin embargo, es importante tener en cuenta que estos modelos no tienen gran parecido al funcionamiento, la escala y la complejidad del cerebro humano. La analogía con el funcionamiento de las neuronas biológicas viene más bien dada por la forma en la que las neuronas artificiales se conectan para formar una red artificial más compleja que es capaz de resolver algunos problemas que una sola unidad básica (neurona) no podría resolver por sí misma. Dentro del campo de las RNA existen diferentes arquitecturas, que no son otra cosa que la forma en que se conectan las unidades básicas o neuronas para formar una estructura que indica cual será el flujo de la información. En el presente trabajo nos enfocamos en estudiar el perceptron multicapa (PM) y las redes neuronales convolucionales (RNC) que nos serán útiles para abordar los problemas que se pretenden resolver. A continuación, se da una breve descripción sobre el aprendizaje au-

tomático para posteriormente pasar a discutir el funcionamiento de las RNA centrándonos en las arquitecturas antes mencionadas.

2.1. El Aprendizaje Automático

El aprendizaje automático es la rama de la IA que se encarga del desarrollo de técnicas que permiten que las computadoras aprendan a partir de datos, sin haber sido programadas explícitamente. Es decir, el objetivo del aprendizaje automático es el desarrollo de algoritmos computacionales que permitan realizar el aprendizaje a partir de un conjunto de datos obtenidos del mundo real sin la intervención del ser humano en la definición explicita de las reglas de dicho aprendizaje. Existe actualmente una variedad de algoritmos y métodos de aprendizaje automático especializados en la solución de los diferentes tipos de problemas, entre estos métodos se encuentran las RNA.

En términos de aplicaciones de la IA, los algoritmos de aprendizaje automático constituyen la principal técnica para la construcción de los sistemas inteligentes modernos. Existen dos paradigmas dentro del aprendizaje automático: el aprendizaje supervisado y el aprendizaje no supervisado. A continuación, se describe cada uno de ellos.

2.1.1. Aprendizaje supervisado

El aprendizaje supervisado consiste en entrenar un modelo con un conjunto de datos previamente etiquetado, es decir, un conjunto de muestra a partir del cual el modelo es capaz de generalizar (aprender). Este conjunto de datos esta formado por pares $(\mathbf{x}, \mathbf{y})$, donde $\mathbf{x}$ es un vector de características o atributos que representa una entrada del modelo e $\mathbf{y}$ representa la salida deseada o etiqueta que caracteriza dichos atributos. La salida $\mathbf{y}$ puede ser una variable cuantitativa (escalar) o una variable categórica (vectorial). Si esta variable es cuantitativa entonces $\mathbf{y}$ representa un valor numérico, por otro lado, si es una variable categórica entonces $\mathbf{y}$ contiene la información de la categoría a la cual pertenecen los atributos de la

entrada **x**.

Un ejemplo de un modelo de aprendizaje supervisado es una regresión lineal múltiple, la cual consiste en estudiar la relación entre una variable de interés y (variable dependiente) y un conjunto de variables explicativas $x_1, x_2, ..., x_p$ (variables independientes), para lo cual se supone un modelo lineal de la forma

$$\hat{y} = \beta_0 + \beta_1 x_1 + ... + \beta_p x_p, \tag{2.1}$$

aquí la entrada del modelo es $\mathbf{x} = (x_1, x_2, ..., x_p)$ y la salida es $\hat{y}$. En este caso el problema consiste encontrar los parámetros $\beta_0, \beta_1, ..., \beta_p$ tal que la salida $\hat{y}$ sea lo más aproximada posible a la salida deseada y. Para el entrenamiento de este modelo se debe contar con un conjunto de datos de muestra formado por pares $(\mathbf{x}, y)$, en este caso la salida deseada y es una variable cuantitativa pues representa un valor numérico.

2.1.2. Aprendizaje no supervisado

El aprendizaje no supervisado consiste en entrenar un modelo con un conjunto de datos de los cuales no se tiene conocimiento *a priori*. Mientras que para el aprendizaje supervisado se cuenta con un conjunto de datos que utilizamos para predecir una determinada salida, en el aprendizaje no supervisado no contamos con una salida deseada, es decir, no es necesario tener datos etiquetados para entrenar el modelo. Como solo conocemos datos de entrada pero no datos de salida, que se corresponden con dichas entradas, solo se puede describir la estructura de los datos para intentar encontrar algún tipo de organización en estos, en ese sentido el aprendizaje no supervisado tiene un carácter exploratorio.

Ejemplos de aprendizaje no supervisado son los problemas de agrupación o *clustering*, los cuales consisten en clasificar los datos en grupos, atendiendo a sus características. Esta clasificación no se realiza en clases predefinidas, sino que, el modelo identifica similitudes entre los datos y los agrupa según esas características.

2.1.3. Problemas de regresión y clasificación

Los problemas que se pueden resolver usando aprendizaje supervisado se dividen en dos grupos: problemas de regresión y problemas de clasificación. Hemos mencionado previamente que para los modelos de aprendizaje supervisado la salida deseada puede ser, o bien una variable cuantitativa, o bien una variable categórica. En el primer caso estamos tratando con un problema de regresión, mientras que en el segundo caso nos referimos a un problema de clasificación. La diferencia está en el tipo de resultado que queremos que el modelo produzca. Para problemas de regresión la salida deseada será un valor numérico dentro de un conjunto infinito de posibles resultados, por ejemplo, el problema de regresión lineal que se menciona en la sección 2.1.1 es un caso de este tipo de problemas. Para problemas de clasificación la salida deseada representa una clase, entre un conjunto finito de clases, es decir, el modelo clasifica los datos en categorías previamente definidas. Por ejemplo, un clasificador de *spam* en correos electrónicos, recibirá como entrada un correo y deberá ser capaz de elegir a que clase pertenece: *spam* o no *spam*. Para el propósito de este trabajo, nos centraremos en los problemas de clasificación. Supongamos que queremos entrenar un modelo de aprendizaje automático, para lo cual contamos con un conjunto de datos de entrenamiento que consta de n elementos, cada uno de estos elementos son pares $(\mathbf{x}, \mathbf{y})$, donde $\mathbf{x}$ es un vector de características o atributos que corresponde a la entrada del modelo, e $\mathbf{y}$ es un vector que corresponde a la salida deseada del modelo y contiene la información de la clase a la cual pertenece los atributos de la entrada $\mathbf{x}$. Si a este conjunto de datos lo llamamos $\boldsymbol{D}$, podemos escribir:

$$\boldsymbol{D} = \left\{(\mathbf{x}_i, \mathbf{y}_i) : \mathbf{x}_i \in \mathbb{R}^{N_x}, \mathbf{y}_i \in \mathbb{R}^{N_y}, i = 1, ..., n\right\}, \tag{2.2}$$

es decir, el par ordenado $(\mathbf{x}_i, \mathbf{y}_i)$ es el elemento i-ésimo del conjunto de datos, además N_x es el número de atributos de cada entrada y N_y es el número de clases en el que están

clasificados los datos. Por lo general, las entradas del vector $\mathbf{x}_i$ puede ser cualquier número real que represente algún atributo, no así, los vectores $\mathbf{y}_i$, pues lo más lógico sería asociar un único vector a cada una de las N_y clases, para tal propósito escogemos los vectores de la base canónica de $\mathbb{R}^{N_y}$. Asociamos cada vector a una de las clases como sigue:

$$clase \quad 1: \quad (1, 0, \cdots, 0), \tag{2.3}$$

$$clase \quad 2: \quad (0, 1, \cdots, 0), \tag{2.4}$$

$$\vdots \qquad \vdots \tag{2.5}$$

$$clase \quad N_y: \quad (0, 0, \cdots, 1). \tag{2.6}$$

El conjunto de datos de entrenamiento $\boldsymbol{D}$, está conformado por pares de vectores $(\mathbf{x}_i, \mathbf{y}_i)$ que representan puntos de coordenadas en un espacio, para los cuales, a cada punto $\mathbf{x}_i$ en $\mathbb{R}^{N_x}$ se la asocia el punto $\mathbf{y}_i$ en $\mathbb{R}^{N_y}$, es decir, existe una especie de mapeo entre los puntos $\mathbf{x}_i$ e $\mathbf{y}_i$. Un problema de clasificación en aprendizaje supervisado, consiste precisamente en encontrar un mapeo que relacione las muestras $\mathbf{x}_i$ con los objetivos $\mathbf{y}_i$, de manera tal que para una nueva muestra $\mathbf{x}_{n+j}$ que no pertenezca al conjunto de datos D, este mapeo sea capaz de estimar $\mathbf{y}_{n+j}$ que clasifique de manera correcta la nueva muestra. El conjunto de datos $\mathbf{x}_i$ tal que $i = n+1, ..., n+m$ se conoce como conjunto de generalización. En términos matemáticos, un modelo de aprendizaje supervisado es un mapeo $\mathbf{g} : \mathbb{R}^{N_x} \longmapsto \mathbb{R}^{N_y}$, este mapeo tiene la forma

$$\hat{\mathbf{y}} = \mathbf{g}(\mathbf{x}, \mathbf{w}), \tag{2.7}$$

donde $\mathbf{w}$ es un vector de parámetros del modelo que transforman $\mathbf{x}$ en $\hat{\mathbf{y}}$ bajo la función vectorial $\mathbf{g}$, a $\hat{\mathbf{y}}$ se le llama salida del modelo. Para cada muestra $\mathbf{x}_i$ obtenemos una salida

$\hat{\mathbf{y}}_i$ que en principio es diferente a la salida deseada $\mathbf{y}_i$. El proceso de aprendizaje consiste en encontrar el conjunto de parámetros $\mathbf{w}$ tal que las salidas $\hat{\mathbf{y}}_i$ coincidan con los objetivos $\mathbf{y}_i$ (o estén muy próximos a coincidir), y además el modelo sea capaz de generalizar los resultados para muestras que no pertenezcan al conjunto de entrenamiento. Esta búsqueda de parámetros del modelo se hace mediante una función de error. Existen diferentes tipos de funciones de error para modelos de aprendizaje automático, las cuales se eligen dependiendo de la naturaleza del problema que se quiera resolver. Un ejemplo clásico de este tipo de funciones es la función de error cuadrático medio, la cual se define por:

$$e(\mathbf{w}) = \frac{1}{n} \sum_{i=1}^{n} \left\| \mathbf{y}_i - \hat{\mathbf{y}}_i \right\|^2, \tag{2.8}$$

donde $e(\mathbf{w})$ es el error cometido por el modelo, el cual está dado por la suma promedio de las distancias al cuadrado entre los puntos $\mathbf{y}_i$ e $\hat{\mathbf{y}}_i$. El proceso de entrenamiento de un modelo de aprendizaje automático se reduce entonces a minimizar una función de error, es decir, a un problema de optimización. Más adelante discutiremos la forma de abordar este tipo de problemas, para el caso particular de las RNA.

2.2. La neurona artificial y el perceptron multicapa

La unidad básica de una red neuronal biológica es la neurona biológica, si bien existen distintos tipos de neuronas biológicas, un esquema simplificado del tipo más común de neuronas se muestra en la figura 2.1. La función principal de las neuronas es la transmisión de impulsos nerviosos. Estos viajan por toda la neurona comenzando por las dendritas hasta llegar a las terminaciones del axón, donde pasan a otras neuronas por medio de las conexiones sinápticas. La neurona artificial representa vagamente, una analogía simplificada de este esquema. En la figura 2.2 se muestra un diagrama de una neurona artificial, esta recibe como entrada el vector $\mathbf{x} = (x_1, x_2, ..., x_d)$ a partir del cual genera la salida z

mediante ciertas operaciones matemáticas. Estas operaciones se representan por medio de la siguiente ecuación:

$$z = f\left(\sum_{i=1}^{d} w_i x_i + b\right), \tag{2.9}$$

donde x_i y w_i representan la i-ésima coordenada del vector de entradas $\mathbf{x}$ y vector de pesos $\mathbf{w}$ respectivamente, el parámetro b se conoce como sesgo, $f(t)$ es una función de activación y z es la salida de la neurona. Las entradas x_i son el estímulo que recibe la neurona a partir del cual puede aprender; los pesos y el sesgo son parámetros ajustables del modelo; la función $f(t)$ determina si la neurona se activa o no para transmitir la información por medio de la salida z. Existen diversos tipos de funciones de activación, algunas de las más comunes se describen a continuación.

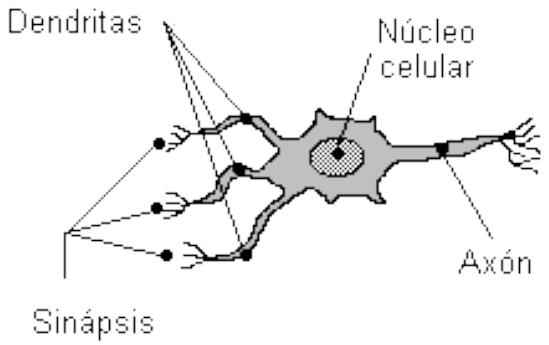


Figura 2.1: La imagen muestra un esquema gráfico simplificado del tipo mas común de neurona biológica.

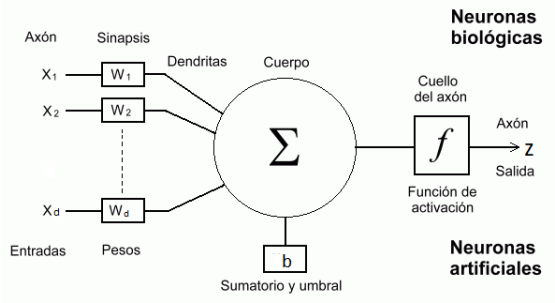
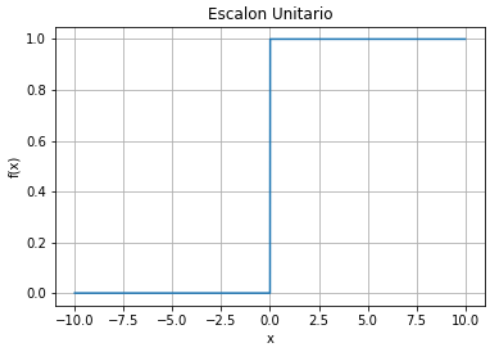


Figura 2.2: Se muestra el esquema gráfico de una neurona artificial, la cual recibe como entradas los números x_1,x_2,...,x_d que multiplica con sus respectivos pesos para posteriormente sumarlos y adicionar el sesgo, que luego se evalúan en una función de activación. El esquema muestra la analogía que existe con una neurona biológica.

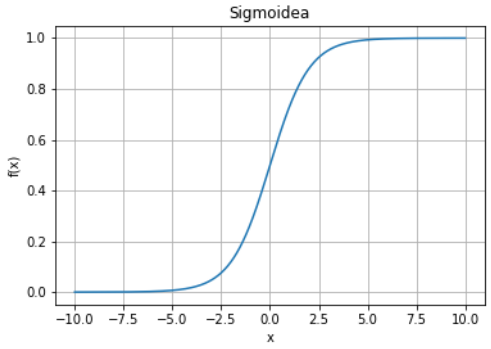
- **Escalón unitario o función de *Heaviside***



La función escalón unitario o función de *Heaviside* es una función que para valores negativos vale 0 y para valores positivos vale 1. En una neurona artificial, el valor 1 indica que ésta se activó y el valor 0 que no se activó.

$$f(t) = H(t) := \begin{cases} 0 & \text{si } t < 0 \\ 1 & \text{si } \geq 0 \end{cases} \tag{2.10}$$

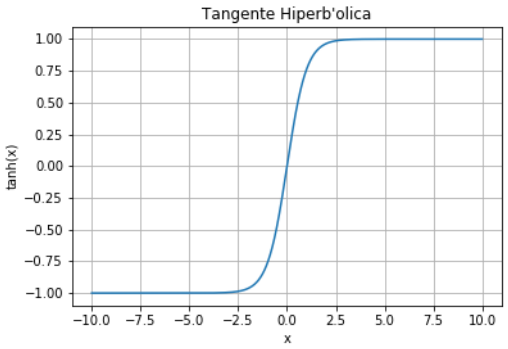
- **Función sigmoide**



La función sigmoide se utiliza por lo general para clasificación binaria. Una neurona con esta función de activación corresponde al modelo de una regresión logística, que es un modelo que se utiliza para clasificación multiclase, ver [5].

$$f(t) = \sigma(t) := \frac{1}{1 + e^{-t}} \tag{2.11}$$

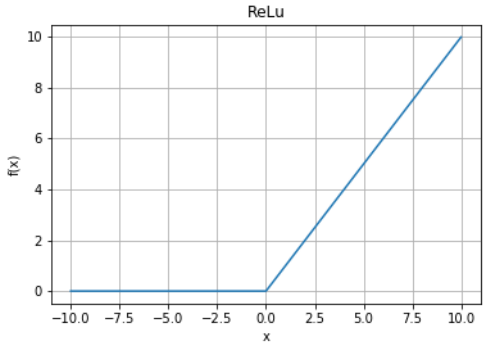
- **Tangente hiperbólica**



La función tangente hiperbólica tiene una forma muy parecida a la sigmoide, en cuanto que tiene valores acotados, la diferencia es que estos valores corresponden al intervalo $(-1, 1)$, es decir, puede tomar valores positivos y negativos.

$$f(t) = tanh(t) = \frac{e^t - e^{-t}}{e^t + e^{-t}} \tag{2.12}$$

- **Unidad lineal rectificada (ReLU)**



La función ReLU es una función de activación propuesta recientemente, usada comúnmente en las redes neuronales convolucionales y redes neuronales profundas que tiene un mejor rendimiento que las funciones sigmoide o tangente hiperbólica, ver [6].

$$f(t) = ReLU(t) = \begin{cases} t & \text{si } t > 0 \\ 0 & \text{si } t \leq 0 \end{cases} \tag{2.13}$$

Se ha descrito el esquema matemático de una neurona artificial; si conectamos entre si varias neuronas formamos una RNA. En la figura 2.3 observamos el esquema de una RNA donde cada nodo representa una neurona, además cada capa de neuronas está totalmente conectada con la capa anterior de donde recibe la información que a su vez propaga hacia la capa siguiente. Este tipo de arquitectura se conoce como perceptron multicapa (PM), como podemos observar este consta de una capa de entrada, una capa de salida y un cierto número de capas ocultas; la información se inserta a la red por la capa de entrada y fluye en una misma dirección hasta llegar a la capa de salida [7]. La capa de entrada se encarga de distribuir la información que llega a la red, a cada una de las neuronas de la primera

capa oculta. Si pensamos en términos del conjunto de datos $\boldsymbol{D}$ de la sección 2.1.3, donde cada vector de entrada $\mathbf{x}_i$ tiene N_x coordenadas, la capa de entrada debe tener entonces N_x neuronas. La salida de cada una de las neuronas de la capa de entrada es

$$z_i^{(1)} = x_i, \tag{2.14}$$

donde $z_i^{(1)}$ representa la salida de la i-ésima neurona de la capa 1, y x_i representa la i-ésima coordenada del vector de entrada $\mathbf{x}$, con i = 1,...,N_x. La función de esta capa es distribuir la infracción de entrada a las capas ocultas. Para el resto de capas la salida es,

$$z_i^{(c)} = f\left(\sum_{j=1}^{N_c} w_{ij}^{(c)} x_j^{(c)} + b_i^{(c)}\right), \tag{2.15}$$

donde $z_i^{(c)}$ representa la salida de i-ésima neurona de la capa c, w_{ij}^c es el parámetro de peso que conecta a la j-esima neurona de la capa anterior con la i-esima neurona de la capa actual c, $b_i^{(c)}$ es el sesgo asociado a la i-ésima neurona de la capa c, $x_j^{(c)}$ representa las salidas de la capa anterior, es decir, $x_j^{(c)} = z_i^{(c-1)}$. Además, $c = 2, 3, \ldots, C$ es el número asociado a la capa actual, C es el número total de capas y $N_1 = N_x$, N_2, N_3,...,$N_C = N_y$ representan el número de neuronas en cada capa. El conjunto de parámetros del modelo para un PM, está formado tanto por los pesos como por los sesgos; el número total de parámetros para este modelo es entonces,

$$N = \sum_{i=1}^{C-1} (N_i N_{i+1} + N_{i+1}). \tag{2.16}$$

Como ya mencionamos en la sección 2.1.3, entrenar un modelo de aprendizaje automático consiste en buscar los valores de los parámetros del modelo que permiten que éste generalice el aprendizaje de los datos de entrenamiento a nuevos datos de entrada. Este problema se

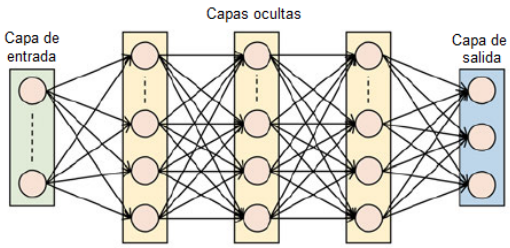


Figura 2.3: Esquema gráfico de un perceptron multicapa que consta de una capa de entrada, una capa de salida y tres capas ocultas. Cada una de las neuronas de cada capa están conectadas con la capa anterior de donde reciben información que propagan a cada neurona de la capa posterior, haciendo fluir la información desde la capa de entrada a la capa de salida.

vuelve un problema de optimización el cual se busca minimizar una función de error. Aun que existen diferentes tecnicas para resolver este problema, por lo general, en el contexto de las RNA este proceso de realiza mediante el método de descenso de gradiente.

2.3. El descenso de gradiente

El método de descenso de gradiente es uno de los algoritmos de optimización más populares en aprendizaje automático, particularmente por su uso extensivo en el campo de las RNA [8]. El descenso de gradiente es un método general para minimizar funciones, su utilidad radica en su fácil implementación y en que permite minimizar funciones que dependen de un número grande de variables. La idea central del descenso de gradiente se basa en el significado matemático del vector gradiente, el cual indica la dirección de máximo cambio de la función sobre la cual actúa. La idea es dar pasos de forma iterativa en la dirección opuesta al gradiente de la función en el punto actual, porque esta es la dirección de descenso más pronunciado. El método de descenso de gradiente se utiliza en las RNA para minimizar el error del modelo. En general para una función de error $e = e(\mathbf{w})$ (por ejemplo la función de la ecuación 2.8), el descenso de gradiente se hace por medio de la siguiente ecuación:

$$w_i^{(t+1)} = w_i^{(t)} - \eta \frac{\partial e(\mathbf{w})}{\partial w_i}, \tag{2.17}$$

donde $w_i^{(t)}$ es el valor del i-ésimo parámetro de la RNA en la iteración t y $\eta > 0$ es el tamaño del paso, conocido como tasa de aprendizaje dentro del campo del aprendizaje automático; el valor inicial de los parámetros $w_i^{(0)}$ se escogen de manera aleatoria. Aun que el descenso de gradiente es un algoritmo muy sencillo presenta el inconveniente que para conjuntos de datos grandes aumenta el tiempo de convergencia, incluso en algunos casos puede no converger. Una variante de este algoritmo que pretende solucionar el problema del tiempo de convergencia es el descenso de gradiente estocástico que a continuación se describe.

2.3.1. El descenso de gradiente estocástico

El descenso de gradiente estocástico es una variante del descenso de gradiente, la cual consiste en calcular el gradiente en cada iteración del proceso de aprendizaje, sólo para una muestra del total de datos de entrenamiento [9]. A esta muestra se le conoce como lote y es elegida de forma aleatoria. La principal ventaja del descenso de gradiente estocástico sobre el descenso de gradiente convencional, es la eficiencia en cuanto al tiempo de computo, que disminuye debido a que el calculo del gradiente se hace solo sobre una muestra del total de datos; otra ventaja es que brinda cierta aleatoriedad al proceso de optimización lo cual le permite escapar de mínimos locales. Las desventajas radican en que, se debe poner mayor atención en la elección de los parámetros y el número de iteraciones del algoritmo, debido a que el calculo del gradiente es es solo una aproximación del gradiente real. En general, una de las problemáticas del descenso de gradiente es la elección de la tasa de aprendizaje η. Los estudios muestran, que la elección de una tasa de aprendizaje grande le permite al algoritmo una mayor exploración del dominio de la función, que le permite escapar de mínimos locales, pero tiene la desventaja de que puede causar que el algoritmo no converja. Por el contrario, la elección de una tasa de aprendizaje pequeña garantiza la convergencia

del algoritmo a un mínimo de la función, pero tiene la desventaja que este mínimo puede ser un mínimo local, además, puede causar que el tiempo de convergencia sea mayor. Existen algunas variantes del descenso de gradiente estocástico que buscan solucionar el problema de la elección de la tasa de aprendizaje, a estos algoritmos se les denomina optimizadores en el contexto del aprendizaje automático.

2.3.2. Optimizadores

Los optimizadores son algoritmos basados en el descenso de gradiente estocástico que pretenden solucionar el problema de la elección de la tasa de aprendizaje. Una solución intuitiva a este problema consiste en hacer que la tasa de aprendizaje sea una función decreciente que depende del paso de tiempo, es decir, $\eta = \eta(t)$, por lo que las primeras iteraciones permiten hacer una mejor exploración del dominio de la función para evitar mínimos locales, mientras que las iteraciones posteriores estarán dedicadas a garantizar la convergencia del algoritmo. Existen diversos optimizadores que utilizan diferentes estrategias para la elección adaptativa de la tasas de aprendizaje, a continuación se mencionan algunos de ellos, enfatizando el optimizador Adam el cual se utilizó en el presente trabajo.

- ***AdaGrad***

  AdaGrad es un método de descenso de gradiente estocástico con tasa de aprendizaje adaptativa, publicado por primera vez en 2011, [10]. Realiza actualizaciones más pequeñas para parámetros asociados con características que ocurren con frecuencia, y actualizaciones más grandes para parámetros asociados con características que ocurren con poca frecuencia. Su estrategia se basa en guardar un historial de los gradientes calculados en los pasos anteriores, lo cual puede representar una desventaja después de muchas iteraciones por la acumulación de datos. Sus principales aplicaciones incluyen problemas de procesamiento del lenguaje natural y el reconocimiento de imágenes.

- ***RMSProp***

  RMSProp es un método de descenso de gradiente estocástico con tasa de aprendizaje adaptativa utilizado en RNA, que muestra significativas mejoras con respecto al descenso de gradiente estocástico convencional [11]. Su estrategia consiste en comparar los signos de los dos gradientes anteriores, si estos coinciden significa de que vamos en la dirección correcta por lo cual se debe aumentar la tasa de aprendizaje, por el contrario, si los signos no coinciden implica que hemos dado un paso demasiado grande y saltamos sobre un mínimo local, por lo que se debe disminuir la tasa de aprendizaje.

- ***Adam***

  Adam (Adaptive Moment Estimation) es un método de descenso de gradiente estocástico eficiente que solo requiere gradientes de primer orden y pocos requisitos de memoria [12]. Este método esta diseñado para combinar las ventajas de los métodos AdaGrad y RMSProp, es decir, funciona bien con gradientes dispersos y solo requiere información de los dos gradientes anteriores. La actualización de cada uno de los parámetros w viene dada por:

$$w^{(t+1)} = w^{(t)} - \eta \frac{\hat{m}}{\sqrt{\hat{v}} + \epsilon}, \tag{2.18}$$

  donde $w^{(t+1)}$ es la actualización del parámetro w que depende de su valor al tiempo anterior, el valor para $w^{(0)}$ se escoge de manera aleatoria, ϵ es un parámetro pequeño mayor que cero que impide dividir por cero. Los parámetros $\hat{m}$ y $\hat{v}$ se calculan de la siguiente manera:

$$\hat{m} = \frac{m_w^{(t+1)}}{1 - \beta_1^{(t+1)}} \tag{2.19}$$

y

$$\hat{v} = \frac{v_w^{(t+1)}}{1 - \beta_2^{(t+1)}}, \tag{2.20}$$

con

$$m_w^{(t+1)} = \beta_1 m_w^{(t)} + (1 - \beta_1)\frac{\partial e}{\partial w} \tag{2.21}$$

y

$$v_w^{(t+1)} = \beta_2 v_w^{(t)} + (1 - \beta_2)\left(\frac{\partial e}{\partial w}\right)^2, \tag{2.22}$$

$m_w^{(t)}$ y $v_w^{(t)}$ se conocen como primer y segundo momento respectivamente, los parámetros β_1 y β_2 se eligen de manera aleatoria cada paso de tiempo y son números reales en el intervalo $[0, 1)$.

2.4. Entrenamiento de redes neuronales artificiales

El entrenamiento de una RNA es el proceso mediante el cual el modelo aprende a partir de un conjunto de datos de entrenamiento, mediante la búsqueda de sus parámetros óptimos. La eficiencia de este aprendizaje se mide en cuanto a la capacidad de la RNA para generalizar el conocimiento adquirido a partir de los datos de entrenamiento a nuevas muestras de datos que el modelo no ha visto. Las métricas utilizadas para evaluar el aprendizaje en un problema de clasificación, son: el error y precisión del modelo. El error es precisamente el valor de la función de error; la precisión se define como la fracción de ejemplos que el modelo clasifica de manera correcta. A cada paso de entrenamiento se le llama época, la cual consiste en una o más iteraciones (pasos) del optimizador utilizado para minimizar la función de error, este optimizador calcula el gradiente sobre una muestra de los datos de entrenamiento, a esta muestra se le conoce como lote. Para garantizar la generalización del aprendizaje se utiliza la estrategia de dividir el conjunto total de datos con los que conta-

mos, en tres partes: un conjunto de entrenamiento, un conjunto de validación y un conjunto de prueba. A continuación se describe cada uno.

- **Conjunto de entrenamiento.** Este es el conjunto de datos de ejemplos utilizados durante el procesos de aprendizaje y se utiliza para ajustar los parámetros del modelo. En cada época del proceso de aprendizaje se miden el error y la precisión del modelo.

- **Conjunto de validación.** Este conjunto, también llamado conjunto de generalización, se utiliza para medir capacidad de aprendizaje del modelo. Al igual que para el conjunto de entrenamiento, en cada época del proceso de aprendizaje, se miden el error y la precisión del modelo, pero este no interviene en el ajuste de los parámetros. De este conjunto solo nos interesa medir la capacidad de generalización del modelo.

- **Conjunto de prueba.** El conjunto de prueba, se mantiene aparte de los otros dos, durante el proceso de aprendizaje. Este conjunto nos permite medir la capacidad de predicción del modelo,la métrica que utilizamos para ello es la precisión, es decir, la fracción de ejemplos clasificados correctamente.

La separación de los datos en estos tres conjuntos se hace de manera tal que, el conjunto de entrenamiento contenga más datos que los otros dos, una posible forma de separar los datos es la siguiente: 60 % para entrenamiento, 20 % para validación y 20 % para prueba. Además los datos deben estar distribuidos de manera uniforme en los tres grupos de datos, por lo cual se recomienda separarlos de manera aleatoria. La finalidad de esto, es que los tres conjuntos de datos sean muestras representativas del problema que se quiere resolver.
Como ya mencionamos, el proceso de aprendizaje se realiza usando los conjuntos de entrenamiento y validación. El objetivo es minimizar el error del modelo usando los datos de entrenamiento, a la vez que se mide su capacidad de generalización sobre los datos de validación. Este proceso se se realiza utilizando las métricas antes mencionadas, en particular, el error, el cual nos sera útil para decidir el momento en que se debe detener el proceso de aprendizaje. En la figura 2.4 se muestra el comportamiento del error en cada

época del proceso de aprendizaje, como podemos observar, el error de entrenamiento tiene un comportamiento decreciente durante todo el proceso, no así, el error de validación, pues este presenta un comportamiento decreciente en las primeras épocas, pero llega un punto después del cual empieza a aumentar. En éste punto mínimo del error de validación, el modelo alcanza su máxima capacidad de generalización, por tanto es la época donde se debe detener el entrenamiento del modelo. Después de esto entra en juego la métrica de precisión del modelo, la cual mide la capacidad de clasificación del modelo. Después de esta época, cuando el error de entrenamiento sigue disminuyendo y el error de validación aumenta, decimos que se tiene un problema de sobre ajuste, es decir, el modelo se ajusta bien a los datos de entrenamiento pero no es capaz de generalizar. Lo ideal sería que, una vez encontrada la época óptima de mínimo error de validación, también la precisión del modelo sea buena, es decir, que éste sea capaz de clasificar correctamente la mayor cantidad de ejemplos. El conjunto de datos de prueba se utilizada para medir la capacidad de de predicción del modelo, una vez este ha sido entrenado.

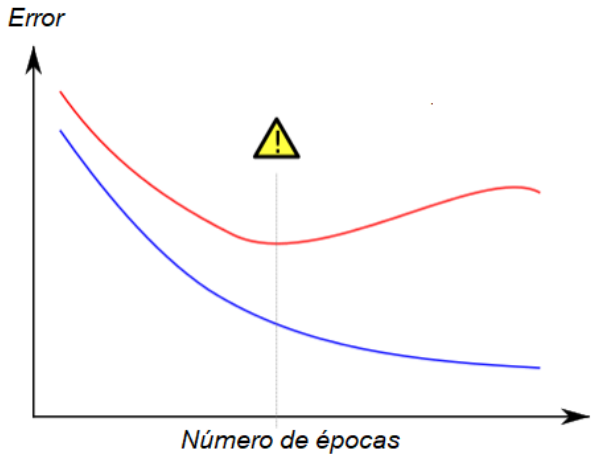


Figura 2.4: Se muestra un gráfico clásico en el entrenamiento de una RNA, en el cual se observa el comportamiento del error del conjunto de entrenamiento (color azul) y el conjunto de validación (color rojo) para cada iteración del proceso de aprendizaje. El error de validación empieza decreciendo hasta que en un cierto momento empieza a crecer, mientras el error de entrenamiento sigue decreciendo, es decir, el modelo a entrado en la región de sobre ajuste.

2.5. Redes neuronales convolucionales

Las redes neuronales convolucionales (RNC) son un tipo de arquitectura de RNA que son comúnmente utilizadas para resolver problemas donde se tienen datos de alta dimensión, por ejemplo, imágenes o vídeos [13]. Una RNC esta compuesta por dos fases: una primera fase de capas convolucionales y de reducción de muestreo que permiten la extracción de características locales, seguida de una segunda fase que consiste en un PM, encargado de realizar la clasificación final a partir de las características extraídas en la primera fase. Dada la naturaleza de las convoluciones dentro de las RNC, estas son especialmente aptas para clasificar o caracterizar imágenes y vídeos, además de que también son ampliamente aplicadas en el análisis de señales y series de tiempo.

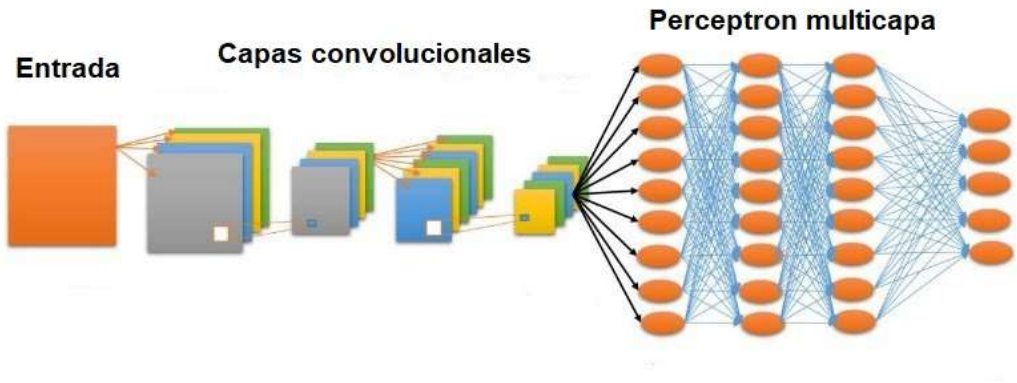


Figura 2.5: Esquema gráfico de una RNC, la cual consiste de dos fases, la primera es una fase de capas convolucionales que extraen las características del objeto de entrada, la segunda fase es un PM que se encarga de clasificar las características de la primera fase.

2.5.1. Capas convolucionales

La primera fase de una RNC, es la fase de capas convolucionales, cada capa convolucional se compone de dos subcapas, estas son las capas de convolución y las capas de agrupación, las cuales se describen enseguida.

- **Capa de convolución**

Como su nombre lo indica, la operación básica de este tipo de capas es la convolución, la cual se compone de un conjunto de filtros también llamados núcleos convolucionales los cuales están conectados con la entrada a partir de la cual generan mapas de características que componen la salida. Consideremos el siguiente vector de características $\mathbf{x} = (x_1, x_2, ..., x_d)$, y consideremos también el filtro convolucional $\mathbf{w} = (w_1, w_2, ..., w_k)$ con $k < d$, se define la operación convolución como se muestra en la ecuación 2.23,

$$z_j = \sum_{i=1}^{k} x_{j+i-1} w_i, \tag{2.23}$$

donde z_j es la j-ésima coordenada del vector $\mathbf{z}$, con $j = 1, 2, ..., d - k + 1$. Esta es la operación de convolución en una dimensión, pero podemos definir de forma similar para más dimensiones, en particular para dos dimensiones, definimos la operación de convolución como se muestra en la ecuación 2.24,

$$(X * W)_{r,s} = M_{r,s} = \sum_{i=1}^{k} \sum_{j=1}^{l} X_{r+i-1,s+j-1} W_{i,j}, \tag{2.24}$$

donde X corresponde a una matriz de características de tamaño m por n y W es un filtro de tamaño k por l, con $k < m$ y $l < n$, además $r = 1, 2, ..., m - k + 1$, $s = 1, 2, ...n - l + 1$. En la figura 2.6 se se ilustra una operación de convolución en 2 dimensiones, en la cual la entrada se corresponde con un objeto de tamaño 4×4 con un filtro de convolución de tamaño 2×2, en la salida obtenemos un objeto de tamaño 3×3. Como podemos observar el filtro de convolución se va desplazando sobre el objeto de entrada para obtener la salida, los pasos de este desplazamiento son de una celda a la vez, tanto de izquierda a derecha como de arriba hacia abajo. En el ámbito de las RNC estos pasos de desplazamiento pueden ser de más de una celda, en tal caso debemos modificar las ecuaciones 2.23 y 2.24. Por ejemplo,

en el caso de la convolución 2D, si queremos realizar desplazamientos horizontales de 2 celdas y desplazamientos verticales de 3 celdas la ecuación 2.24 se modifica de acuerdo con la ecuación 2.25

$$(X * W)_{r,s} = M_{r,s} = \sum_{i=1}^{k} \sum_{j=1}^{l} X_{3r+i-1,2s+j-1} W_{i,j}, \tag{2.25}$$

donde $r = 1, 2, ..., \lceil \frac{m-k+1}{3} \rceil$ y $s = 1, 2, ..., \lceil \frac{n-l+1}{2} \rceil$, con $\lceil t \rceil$ la función *ceil* (menor entero igual o mayor que t). Para propósito del presente y trabajo nos enfocaremos en convoluciones 2D.

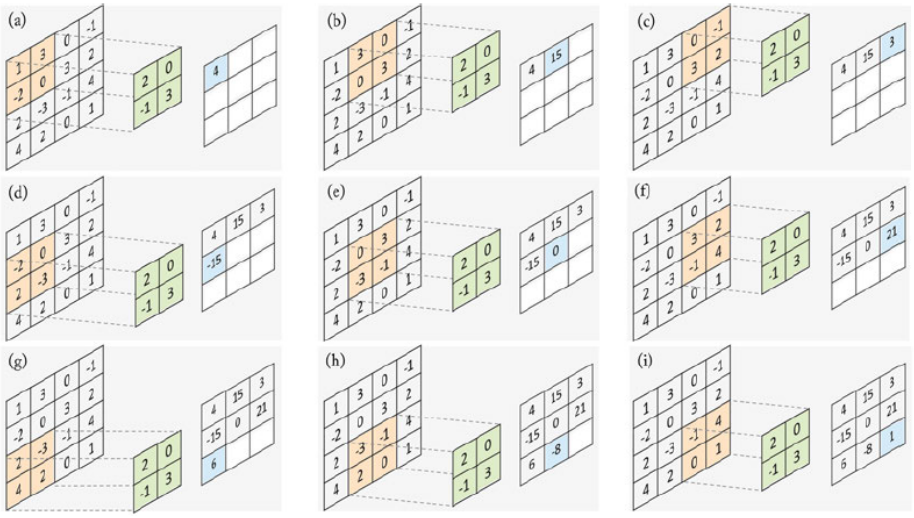


Figura 2.6: En la figura se muestra el proceso de convolución para un objeto bidimensional de tamaño 4×4 y un filtro de convolución de tamaño 2×2. El filtro se va recorriendo dando saltos de una celda, tanto de izquierda a derecha como de arriba hacia abajo, la operación realizada es la descrita por la ecuación 2.24, resultando un objeto de tamaño 3×3.

Una capa de convolución se compone precisamente de estas operaciones antes definidas. Supongamos que tenemos un objeto $\mathbf{X}$ que representa un mapa bidimensional compuesto por I canales, $\mathbf{X}^{(1)}$,$\mathbf{X}^{(2)}$,...,$\mathbf{X}^{(I)}$, este mapa es la entrada de una capa de convolución, la cual nos da como salida otro mapa bidimensional $\mathbf{Z}$ de J canales, $\mathbf{Z}^{(1)}$,$\mathbf{Z}^{(2)}$,...,$\mathbf{Z}^{(J)}$. La operación

que realiza una capa de convolución es la siguiente:

$$\mathbf{Z}^{(j)} = f\left(\sum_{i=1}^{I} \mathbf{X}^{(i)} * \mathbf{W}^{(j)} + B_j\right), \tag{2.26}$$

donde $\mathbf{W}^{(j)}$ es el j-ésimo filtro de la convolución y B_j es un parámetro de sesgo, con $j = 1, 2, \ldots, J$, donde J es el número de filtros de la convolución, además, $f(t)$ es una función de activación. Lo más común es que la función de activación para las capas de convolución sea la una función ReLU. Como podemos notar, las dimensiones del mapa de entra $\mathbf{X}$ son mayores que las dimensiones del mapa de salida $\mathbf{Z}$, si el mapa $\mathbf{Z}$ tiene dimensiones $m' \times n'$ y el mapa X tiene dimensiones $m \times n$, la relación entre estas dimensiones viene dada por las ecuaciones 2.27 y 2.28,

$$m' = \left\lfloor \frac{m - k + r}{r} \right\rfloor \tag{2.27}$$

$$n' = \left\lfloor \frac{n - l + s}{s} \right\rfloor \tag{2.28}$$

donde k y l son las dimensiones del filtro de convolución, r y s son los pasos de desplazamiento vertical y lateral respectivamente, estos pasos representan el número de celdas que se desplaza el filtro de convolución sobre el mapa bidimensional de entrada, al momento de realizar la operación de convolución. Los parámetros de una capa convolucional que se ajustan en el proceso de aprendizaje son los parámetros de los filtros y los sesgos. Es fácil contar el número de parámetros de una capa convolucional, pues, por cada filtro de convolución tenemos $k \times l$ parámetros más un parámetro de sesgo, es decir, para una capa con J filtros se tiene un total de $k \times l \times J + J$ parámetros.

- **Capa de agrupación**

Una capa de agrupación va inmediatamente después de una capa de convolución, y recibe como entrada el mapa bidimensional que da como salida la capa de convolución. Una capa de agrupación consiste de una ventana de tamaño $k \times l$ la cual recorre el mapa de entrada realizando alguna operación de agrupación, las mas usadas son, la operación *MaxPooling* que extrae el elemento máximo de cada ventana y la operación *AveragePooling* que calcula el promedio de los elementos de la ventana. En la figura 2.7 se observa con más detalle estas dos operaciones. Si $\mathbf{X}$ representa el mapa de entrada y $\mathbf{Z}$ el mapa de salida de una capa de agrupación, el número de canales de estos mapas es el mismo, lo que cambia es la dimensión de estos mapas. Supongamos que $\mathbf{X}$ tiene dimensión $m \times n$, y $\mathbf{Z}$ es de dimensión $m' \times n'$, la relación entre estas dimensiones esta dada por las ecuaciones 2.29 y 2.30,

$$m' = \left\lfloor \frac{m}{k} \right\rfloor \tag{2.29}$$

$$n' = \left\lfloor \frac{n}{l} \right\rfloor \tag{2.30}$$

donde $\lfloor t \rfloor$ es la función $floor$ (mayor entero igual o menor que t).

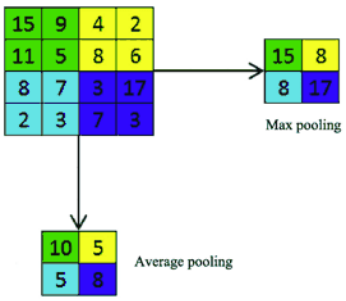


Figura 2.7: Se muestran las operaciones de agrupación *MaxPooling* y *AveragePooling* para un objeto bidimensional de tamaño 4×4 y una ventana de agrupación de tamaño 2×2. En el caso de la operación *MaxPooling* se extrae el elemento máximo de cada ventana, por otro lado, para la operación *AveragePooling* se extrae el promedio de los elementos de cada ventana. El resultado final para cada caso, es un objeto bidimensional de tamaño 2×2.

2.5.2. Capas totalmente conectadas

La segunda fase de capas de una RNC corresponde a un PM, el cual consiste en capas totalmente conectadas como ya se vio en secciones anteriores. Este tipo de capas totalmente conectadas las llamaremos capas densas de ahora en adelante. En una RNC la información fluye de las capas convolucionales a las capas densas, no obstante, la salida de las capas convolucionales puede contener objetos de varias dimensiones, por lo cual esta información antes de ser propagada hacia las capas densas debe ser aplanada de algún modo, es decir, estos objetos multidimensionales se deben transformar en vectores de características que es lo que recibe de entrada un PM. Esta fase de la RNC es la encargada del proceso de clasificación a partir de las características extraídas por las capas convolucionales. Para problemas de clasificación es común utilizar una capa de salida *softmax*, la cual da como resultados un vector de probabilidades. La salida de una capa *softmax* se define por,

$$\hat{y}_j = P_j(\mathbf{z}) = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \tag{2.31}$$

donde $\hat{y}_j$ es la salida de la j-ésima neurona de la capa de salida, ademas,

$$z_j = z_j^{(C)} = \sum_{k=1}^{N_C} w_{jk}^{(C)} x_k^{(C)} + b_j^{(C)}, \tag{2.32}$$

$w_{jk}^{(C)}$ es el peso de conexión entre la j-ésima neurona de la última capa y la k-ésima neurona de la penúltima capa, $x_k^{(C)}$ es la salida de la k-ésima neurona de la penúltima capa y $b_j^{(C)}$ es el sesgo de la j-ésima neurona de la última capa, además N_C es el numero de neuronas de la capa de salida.

Como ya se ha mencionado, el proceso de aprendizaje de una RNA consiste en minimizar una función de error. En el caso de un problema de clasificación, cuando se cuenta con una capa de salida *softmax*, es conveniente usar como función de error la función de entropía

cruzada categórica, la cual se define por la ecuación 2.33,

$$e(\mathbf{w}) = -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{N_C} y_{ij} log(P_{ij}) \tag{2.33}$$

donde n es el número de ejemplos de entrenamiento en cada lote, y_{ij} es la salida deseada para el i-ésimo ejemplo en la j-ésima coordenada del vector de salida, además P_{ij} es el valor de la probabilidad de la j-ésima coordenada para el i-ésimo ejemplo, es decir, $P_{ij} = \hat{y}_j$.

En este capítulo se estudiaron dos modelos de RNA: el PM y las RNC. Se discutió su funcionamiento basado en fundamentos matemáticos así como su importancia dentro del campo del aprendizaje automático y la IA. Se analizó la aplicación de estos modelos a problemas de clasificación en aprendizaje automático, así como, su implementación y la metodología que se debe seguir para su entrenamiento. En el siguiente capítulo se discute la importancia que tienen las técnicas de separación gamma-hadrón en EAS para las observaciones de rayos gamma; se discute la generación de simulaciones de EAS, así como, los datos que se producen en dichas simulaciones.

Capítulo 3

Chubascos atmosféricos

Los rayos cósmicos (RC) son partículas de alta energía provenientes del espacio exterior de fuentes galácticas y extragalácticas. Los RC son la fuente de mayor energía que se ha observado en el universo, éstos han sido estudiados en un amplio rango de energías, desde energías menores que 10^9 eV hasta energías superiores a 10^{20} eV. Es interesante el estudio de los RC, ya que sus mecanismos de producción y aceleración son diversos y no se han podido reproducir condiciones experimentales como esas en laboratorios. La composición de los RC son diferentes tipos de partículas, principalmente protones (86 %), partículas alfa (11 %), electrones (2 %) y núcleos pesados como el uranio (1 %) [1]. Cuando estas partículas, denominadas primarias, impactan con la atmósfera terrestre se inducen chubascos de partículas secundarias, conocidos como chubascos atmosféricos (EAS por sus siglas en ingles, extensive atmospheric showers). El resultado es una mezcla de electrones, positrones y rayos gamma relativistas que se propagan hasta el suelo en un perfil delgado de partículas con velocidades cercanas a la de la luz. Los EAS están constituidos por la superposición de procesos muy complejos, los cuales involucran interacciones fundamentales, así como aspectos de la física de partículas, física de los rayos cósmicos y astrofísica [2].

Las observaciones de rayos gamma de origen astrofísico y de muy alta energía, están basadas en experimentos en la superficie terrestre donde se observan las partículas secundarias al nivel del piso de los EAS inducidos mediante el empleo de dispositivos tecnológicos muy sofisticados llamados detectores de partículas. En la astronomía y la astrofísica se estudian los rayos gamma como herramienta para diversos fines, por ejemplo, estudiar explosiones de supernova, descubrir nuevos objetos astrofísicos, estudiar agujeros negros, estudiar la materia en condiciones extremas para la búsqueda de nueva física, etc. Sin embargo, como ya se mencionó en el capítulo 1, el número de eventos inducidos por hadrones que se registran, es varios ordenes de magnitud más grande que los eventos inducidos por rayos gamma. En los observatorios de rayos gamma como HAWC (High Altitude Water Cherenkov,) se detectan EAS inducidos tanto por rayos gamma como por otros tipos de RC. Debido a que una gran mayoría de los eventos EAS son inducidos por RC hadrónicos, la supresión del ruido de fondo juega un papel importante en el proceso de análisis de datos del observatorio [14].

En este trabajo desarrollamos un modelo de aprendizaje automático usando RNC para la discriminación gamma-hadron de partículas primarias a partir de de las partículas secundarias medidas en tierra del EAS inducido. Los datos de entrenamiento del modelo se generaron a partir de simulaciones utilizando un simulador de EAS. En la secciones 3.1 y 3.2 se discute la generación de simulaciones de EAS así como los datos que produce el simulador.

3.1. Simulación de EAS

Para simular la fenomenología de EAS se utilizó el simulador CORSIKA, considerando como partículas primarias fotones (rayos gamma) y protones (hadrones) con rangos de energía entre 0.5 y 30 TeV. La morfología de la huella en los EAS no depende ex-

clusivamente de la energía o la composición del primario, existe una dependencia angular cenital θ, en el presente trabajo consideramos solamente EAS verticales ((c)zenith $\theta = 0°, ac(z)imuth\phi = 0°$).

CORSIKA (COsmic Ray SImulations for KAscade) es un programa de simulación detallada de EAS inducidos por partículas de rayos cósmicos de altas energías basado en el método de Monte Carlo. Fue desarrollado en el Instituto de Tecnolog´ıa de Karlsruhe, Alemania, para realizar las simulaciones del experimento KASCADE (KArlsruhe Shower Core and Array DEtector). Las partículas son rastreadas a través de la atmósfera hasta el momento que experimentan reacciones con los núcleos del aire o (en el caso de que se trate de partículas inestables) hasta su decaimiento. Las interacciones hedónicas a altas energías pueden describirse por cinco modelos de reacción: *VENUS*, *QGSJET* y *DPMJET* (basados en la teoría Gribov–Regge), *SIBYLL* (modelo minijet), HDPM es un generador fenomenológico ajustado a datos experimentales en la medida de lo posible. Las interacciones hadrónicas a bajas energías son descritos ya sea por las rutinas sofisticadas de interacción de GHEISHA o por el modelo ISOBAR que es más simple. Para los procesos electromagnéticos se usan el programa EGS4 o la formula analítica NKG. También existen las opciones para la generación de radiación Cherenkov y neutrinos [15].

3.2. Datos de simulación

Los datos producidos en una simulación de un EAS, inducido por alguna partícula primaria, son varios parámetros de las partículas secundarias en la huella al piso. Para los fines del presente trabajo solo nos interesan, el tipo de cada partícula secundaria, su posición con respecto al eje vertical entre la partícula primaria y el piso, y su momento lineal. En la figura 3.1 se observa una tabla con los posibles valores de estos parámetros para algún EAS inducido por un primario. En la primera columna, ID de la partícula se refiere al tipo de partícula, estas etiquetas están detalladas en la guía de usuario de CORSIKA [4]. Las

demás columnas de la tabla muestran los valores para las componentes P_x, P_y y P_z del momento lineal, y las componentes de la posición x e y.

ID Partícula	$P_x[eV]$	$P_y[eV]$	$P_z[eV]$	$x[m]$	$y[m]$
1	0.0013842	0.00619146	0.0181968	36540.9	202339
3	-0.000463009	0.00170897	0.0013097	21935.1	-6366.1
13	-0.201099	-0.108235	1.21352	-103682	-56964.4
⋮	⋮	⋮	⋮	⋮	⋮
2	0.00212595	-0.00754431	0.0281506	-2382.11	-660.024

Tabla 3.1: Se muestran diferentes variables en la salida de una simulación para algunas de las partículas secundarias. La primera columna muestra la etiqueta referente al tipo de partícula, la columnas 2, 3 y 4 muestran las componentes P_x, P_y y P_z del momento lineal, las dos ultimas columnas muestran las coordenadas x e y de la posición de la partícula en la huella al piso.

En principio, para dos EAS inducidos por un primario del mismo tipo y con la misma energía, la huella al piso es diferente pues los procesos físicos involucrados en la producción son aleatorios, y en efecto, al analizar los datos producido por las simulaciones nos damos cuenta de estos efectos. Otro dato interesante es que el número de partículas secundarias medidas en la huella al piso, varia con respecto a la energía del primario. En la figura 3.1 se muestra un gráfico donde se compara el número de partículas secundarias en la huella al piso, para simulaciones de EAS inducidos por primarios con diferentes energías, se observa una tendencia creciente en el numero de partículas secundarias con respecto a la energía de la partícula primaria. Por otro lado, al analizar el tipo de partículas secundarias en la huella al piso de los EAS, se observa que estas son en su mayoría fotones. En la figura 3.2 se compara el porcentaje de fotones en la huella al piso, para EAS inducidos por partículas primarias con diferentes valores de energía, se observa que el porcentaje es superior al 80 %, llegando en algunos casos hasta un 92 %; pero no se observa una relación entre este porcentaje y la energía del primario.

La morfología de la huella al piso para los diferentes EAS generados en las simulaciones, depende tanto de la energía como del tipo de partícula primaria. En la figura 3.3 se muestra

la distribución de partículas secundarias en la huella al piso para EAS inducidos por rayos gamma y hadrones de diferentes energías, se observa que, en general, la concentración de partículas es mayor en el centro de la huella al piso del EAS.

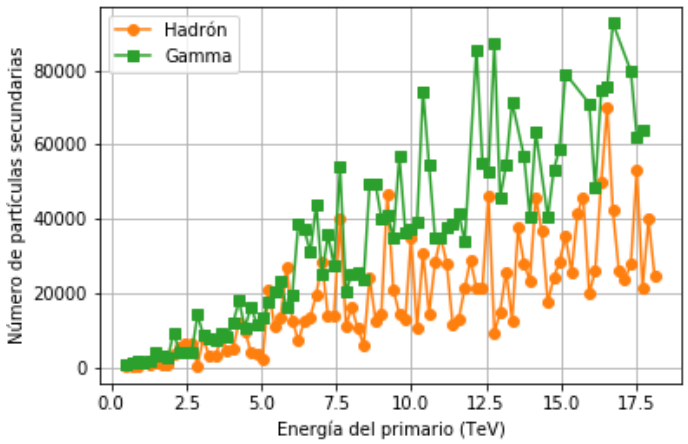


Figura 3.1: Incremento del número de partículas secundarias en la huella al piso del EAS. Se muestra como aumenta, dicho numero, con respecto a la energía de la partícula primaria. Los hadrones son representados por los círculos naranja, los gammas son representados por los cuadros verdes.

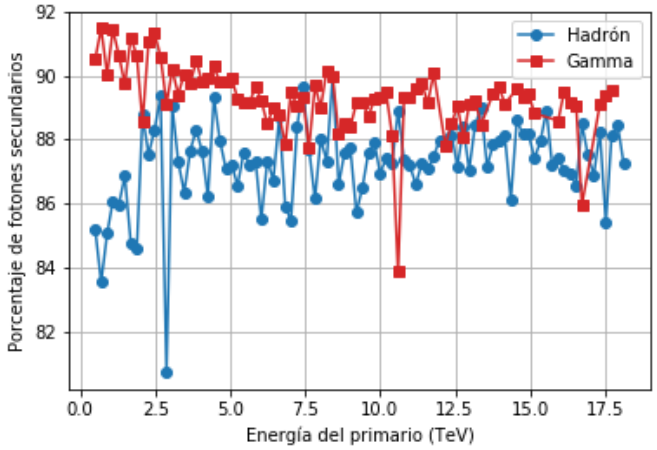
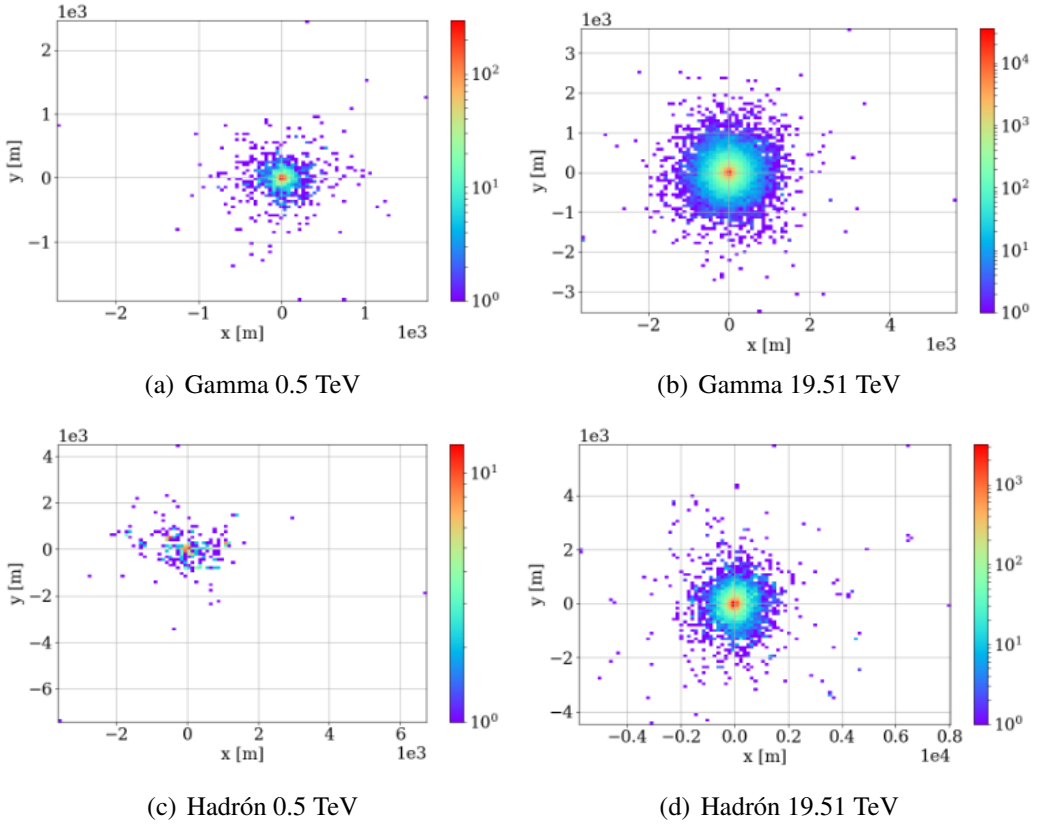


Figura 3.2: Porcentaje de fotones en partículas secundarias en la huella al piso del EAS. Se muestra que este porcentaje varia entre el 80 y 92 % y no muestra una dependencia de la energía de la partícula primaria. Los hadrones son representados por los círculos azules, los gammas son representados por los cuadros rojos.



(a) Gamma 0.5 TeV (b) Gamma 19.51 TeV

(c) Hadrón 0.5 TeV (d) Hadrón 19.51 TeV

Figura 3.3: Se muestran gráficos de la distribución de partículas secundarias en la huella al piso para EAS inducidos por gammas y hadrones con diferentes valores de energía: a) gamma de 0.5 TeV, b) gamma de 19.51 TeV, c) hadrón de 0.5 TeV y d) hadrón de 19.51 TeV. La distribución de partículas secundarias muestra una mayor concentración en la parte central de la huella al piso del EAS.

En este capítulo estudiamos la importancia de las observaciones de rayos gamma de alta energía provenientes del espacio exterior. Se discutió, el obstáculo que representa la radiación cósmica de origen hadrónico en estas observaciones, así como el papel que juegan las técnicas de separación gamma hadrón en la supresión del ruido de fondo hadrónico. Se analizaron los datos producidos por las simulaciones de EAS inducidos por rayos gamma y hadrones. En siguiente capítulo se aborda el problema de discriminación gamma hadrón como un problema de clasificación, haciendo uso de un modelo de RNC.

Capítulo 4

Discriminación gamma-hadrón como un problema de clasificación

En este capítulo nos enfocamos en la solución del problema plateado en este trabajo, que consiste en aplicar la técnica de RNC a la discriminación de EAS inducidos por rayos gamma y hadrones, a partir su huella al piso. Abordamos este problema, como un problema de clasificación en aprendizaje supervisado. Los datos de entrenamiento para el modelo, fueron producidos a partir de simulaciones numéricas de EAS con el simulador CORSIKA, como se mencionó en el capítulo 3. Se estudiaron dos casos, en el primer caso se clasifican los EAS en base al tipo de partícula primaria (gamma o hadrón); en el segundo caso se clasifican los EAS en base a la energía de la partícula primaria. Para resolver estos problemas se utilizó un modelo de RNC de dos dimensiones.

4.1. Procesamiento de datos de las simulaciones

En las simulación de EAS se extrajo información de algunas variables físicas de las partículas secundarias en la huella al piso, es decir, lo que se obtiene para cada simulación, es una lista como la de la tabla 3.1, con la información de cada EAS simulado. Para usar estos datos en el entrenamiento de una RNC, primero debemos procesarlos, de tal forma que la dimensión de todas las entradas de la RNC sean iguales, pues es un requisito de este tipo de modelos. Esto no se podría lograra si solamente le damos como entrada las listas generadas en las simulaciones, ya que estas tienen diferente dimensión, pues el número de partículas secundarias es diferente para cada simulación. Como queremos utilizar un modelo de RNC de dos dimensiones, debemos trasformar estos datos en objetos bidimensionales del mismo tamaño. Para ello construimos mapas bidimensionales que contengan información de la huella al piso de cada EAS. En el capítulo 3 vimos que las partículas secundarias de los EAS son mayormente fotones, en base a esto, las etiquetamos como fotones y no-fotones. Usando esta información, construimos mapas bidimensionales de 4 canales, donde cada canal contiene algún tipo de información del EAS.

- **Canal 1:** contiene información de la distribución de fotones en la huella al piso.
- **Canal 2:** contiene información del promedio del momento lineal de los fotones en cada región de la huella al piso.
- **Canal 3:** contiene información de la distribución de no-fotones en la huella al piso.
- **Canal 4:** contiene información del promedio del momento lineal de los no-fotones en cada región de la huella al piso.

Cada canal es un arreglo bidimensional de tamaño 100×100, el cual, es representado por una malla como la de la figura 4.1, donde cada celda contiene información de las partículas que cayeron en ella. Decimos que una partícula cayó en la celda (i, j), si para su posición

(x, y) se cumple que,

$$x_{min} + (j-1)\Delta x \leq x \leq x_{min} + j\Delta x \tag{4.1}$$

y

$$y_{max} - i\Delta y \leq y \leq y_{max} - (i-1)\Delta y, \tag{4.2}$$

con

$$\Delta x = \frac{x_{max} - x_{min}}{100} \tag{4.3}$$

y

$$\Delta y = \frac{y_{max} - y_{min}}{100}, \tag{4.4}$$

donde, x_{min} y x_{max} son el mínimo y el máximo la coordenada x, y_{min} e y_{max} son el mínimo y el máximo de la coordenada y, de las partículas secundarias. En el ejemplo de la figura 4.1 se marca con color azul la celda (2,7) y con color verde la celda (10,1), por ejemplo, una partícula que tenga posición (x_{min}, y_{min}) decimos que cayó en la celda marcada con verde. Para los canales 1 y 3, cada celda contiene valor del número de fotones y no-fotones respectivamente, que cayeron en ella. Para los canales 2 y 4, cada celda contiene valor del promedio del momento lineal de los fotones y no-fotones respectivamente, que cayeron en ella, este promedio se calcula con la ecuación 4.5,

$$< P >= \frac{1}{N} \sum_{k=1}^{N} p_k, \tag{4.5}$$

donde, N es el número de partículas y p_k es el momento de la k-ésima partícula que cayó en alguna de las celdas, con $p = \|\mathbf{p}\| = \sqrt{p_x^2 + p_y^2 + p_z^2}$.

Cada canal de un mapa se puede representar por una matriz de tamaño 100×100 donde

la celda (i, j) se corresponde precisamente con la entrada (i, j) de la matriz, es decir, cada mapa se representa entonces como un arreglo de dimensiones $100 \times 100 \times 4$. Cada mapa generado a partir de los datos de una simulación de un EAS, mas su etiqueta (gamma o hadrón), conforman un ejemplo o muestra del conjunto de datos de entrenamiento, para la RNC.

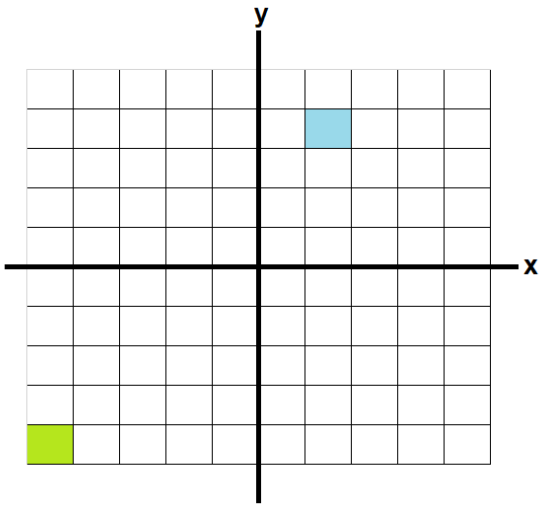


Figura 4.1: En la figura se observa una cuadricula de 10×10 centrada en el origen del plano de coordenadas que representa la imagen de un mapa bidimensional en el cual se recoge la huella al piso dejada por un chubasco atmosférico.

4.2. Clasificación gamma-hadrón

El primer problema consiste en clasificar los EAS en base al tipo de partícula primaria (gamma o hadrón). Para esta prueba contamos con un conjunto de datos, que consta de 527 simulaciones, con 150 valores diferentes de energía, en el intervalo 0.5 – 30 TeV, estos valores están dados por $E_n = 0 \cdot 5 + n\Delta E$ con $n = 0, 1, \ldots, 149$ y $\Delta E = \frac{30-0 \cdot 5}{150}$. Estos datos de dividieron de la siguiente manera: 318 para entrenamiento, 106 para validación, y 103 para prueba. Con estos datos se entrenó una RNC con la estructura que se muestra en la tabla 4.1, esta RNC consta de 2 capas convolucionales y 3 capas densas. En las capas

convolucionales, los filtros de convolución se recorren a lo largo de los mapas dando pasos horizontales y verticales de una celda, la ventana en las capas de agrupación da saltos horizontales y verticales de 2 celdas. La salida del muestreo al pasar por las dos capas convolucionales es un mapa bidimensional de 22×22 con 10 canales, es decir, un arreglo de dimensiones $22 \times 22 \times 10$. Este objeto se convierte en un vector de 4,840 coordenadas antes de pasar a la primera capa densa. Si contamos los parámetros de esta RNC tenemos un total de $(4\times5\times5\times20+20)+(20\times4\times4\times10+10)+(4,840\times80+80)+(80\times40+40)+(40\times2+2) = 395,832$ parámetros. Para el entrenamiento se utilizó la función error de entropía cruzada categórica y el optimizador Adam. La RNC se entrenó durante 180 épocas con 11 pasos por época, con lotes de 30 datos en cada paso.

Número de capa:	Tipo de capa:	Información de los hiperparámetros:
1	Capa Convolucional	20 filtros de tamaño 5x5, función de activación *ReLU*
2	Capa MaxPooling	ventana *MaxPooling* de tamaño 2x2
3	Capa Convolucional	10 filtros de tamaño 4x4, función de activación *ReLU*
4	Capa MaxPooling	ventana de *MaxPooling* de tamaño 2x2
5	Capa Densa	80 neuronas, función de activación *ReLU*
6	Capa Densa	40 neuronas, función de activación *sigmoide*
7	Capa Densa	2 neuronas, función de activación *softmax*

Tabla 4.1: Información de la estructura de la RNC utilizada para la clasificación gamma-hadrón. En la primera columna se muestra el orden de las capas de RNC, en la segunda columna el tipo de capa y en la tercera columna la información de cada capa.

4.2.1. Resultados

Las métricas medidas durante el proceso de aprendizaje de la RNC fueron el error y la precisión del modelo. En la gráfica de la figura 4.2 se muestra la evolución del error en cada época para los conjuntos de entrenamiento y validación. Como es de esperar, el valor del error de validación desciende durante las primeras épocas del entrenamiento, pero luego de un tiempo comienza a crecer. El mínimo en el error de validación se encuentra en la época 52, lo cual nos indica que es en esta época donde obtenemos los parámetros de

mayor rendimiento. En la gráfica de la figura 4.3 se observa la evolución de la precisión del modelo para los conjuntos de entrenamiento y validación. La precisión máxima obtenida para el conjunto de validación es de 97 %, precisamente este máximo lo obtenemos en la época 52, aunque se mantiene en ese mismo valor en la mayor parte del entrenamiento a partir de la época 60. El valor óptimo de los parámetros del modelo se encuentra pues en la época 52, que la época donde se tiene una intersección entre el error mínimo y la precisión máxima sobre el conjunto de validación. La predicción del modelo sobre el conjunto de datos de prueba se muestra en la gráfica de la figura 4.4. En color rojo se muestra los casos errados y en color verde los casos acertados, además se grafica el valor de la probabilidad que da como salida el modelo para el resultado correcto de cada dato en el conjunto de prueba. Del total de los 103 datos de prueba, el modelo es capaz de clasificar correctamente 99, es decir, la capacidad de predicción es del 96 %.

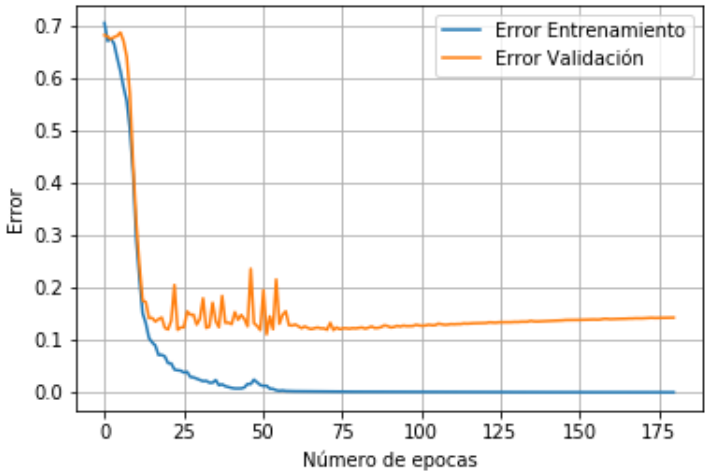


Figura 4.2: Evolución del error de entrenamiento y validación para cada época del proceso de aprendizaje de la RNC. La linea azul representa el error de entrenamiento, la linea naranja representa el error de validación. El valor mínimo del error de validación se encontró en la época 52 del proceso de aprendizaje.

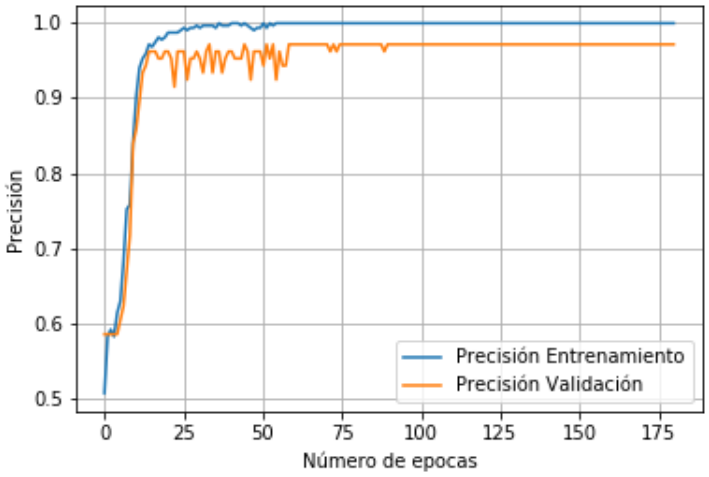


Figura 4.3: Precisión del modelo para los datos de entrenamiento y validación, en cada época del proceso de entrenamiento. La linea azul representa la precisión de entrenamiento, la linea naranja representa la precisión de validación. Se alcanza una precisión de entrenamiento del 100 % y una precisión de validación máxima del 97 %, a partir de la época 52.

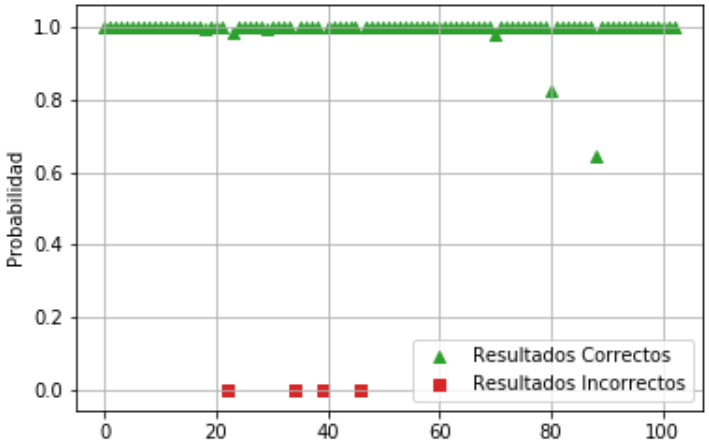


Figura 4.4: Predicción del modelo en los datos del conjunto de prueba para la clasificación gamma-hadrón. Los triángulos verdes representan los ejemplos clasificados correctamente, los cuadros rojos representan los ejemplos clasificados incorrectamente. El eje vertical representa la probabilidad que da como salida la RNC, para la categoría correcta de cada ejemplo.

4.3. Clasificación de intervalos de energía

En el apartado anterior se resolvió el problema de clasificación gamma-hadrón. Ahora nos planteamos el problema de distinguir tanto el tipo de partícula primaria, así como su energía, para cual dividimos el valor de la energía en diferentes intervalos. Las categorías en las cuales se pretenden clasificar las partículas primarias se muestran en la tabla 4.2.

Número de clase:	Información:	Etiqueta vectorial:
1	Gamma 0.5 - 5.0 TeV	(1,0,0,0,0,0)
2	Gamma 5.0 - 10.0 TeV	(0,1,0,0,0,0)
3	Gamma 10.0 - 15.0 TeV	(0,0,1,0,0,0)
4	Hadrón 0.5 - 5.0 TeV	(0,0,0,1,0,0)
5	Hadrón 5.0 - 10.0 TeV	(0,0,0,0,1,0)
6	Hadrón 10.0 - 15.0 TeV	(0,0,0,0,0,1)

Tabla 4.2: Categorías de clasificación de EAS tomando en cuenta el tipo de partícula primaria y su energía. En la primera columna se muestra el número de la clase, en la segunda columna se muestra la información de la clase, en la tercera columna se muestra la etiqueta vectorial que identifica cada clase en el entrenamiento de la RNC.

Para esta prueba contamos con un conjunto de datos conformado por $2,680$ simulaciones de EAS con rangos de energías entre 0.5 y 15 TeV, el cual se dividió de la siguiente manera: 60 % para entrenamiento, 20 % para validación y 20 % para prueba. En el presente problema se realizaron tres pruebas, para cada una de las cuales se utilizó una RNC con diferentes valores para los hiperparámetros, con la intensión de verificar como evoluciona la precisión del modelo al aumentar la profundidad de la RNC. En las tablas 4.3, 4.4 y 4.5 se muestra la información de los hiperparámetros de la RNC para cada una de las pruebas realizadas.

Número de capa:	Tipo de capa:	Información de los hiperparámetros:
1	Capa Convolucional	20 filtros de tamaño 5x5, función de activación *ReLU*
2	Capa *MaxPooling*	ventana *MaxPooling* de tamaño 2x2
3	Capa Convolucional	10 filtros de tamaño 4x4, función de activación *ReLU*
4	Capa *MaxPooling*	ventana de *MaxPooling* de tamaño 2x2
5	Capa Densa	80 neuronas, función de activación *ReLU*
6	Capa Densa	40 neuronas, función de activación *sigmoide*
7	Capa Densa	2 neuronas, función de activación *softmax*

Tabla 4.3: Información de la estructura de la RNC utilizada en la prueba 1. En la primera columna se muestra el orden de las capas de RNC, en la segunda columna el tipo de capa y en la tercera columna la información de cada capa.

Número de capa:	Tipo de capa:	Información de los hiperparámetros:
1	Capa Convolucional	20 filtros de tamaño 5x5, función de activación *ReLU*
2	Capa *MaxPooling*	ventana *MaxPooling* de tamaño 2x2
3	Capa Convolucional	10 filtros de tamaño 4x4, función de activación *ReLU*
4	Capa *MaxPooling*	ventana de *MaxPooling* de tamaño 2x2
5	Capa Convolucional	10 filtros de tamaño 4x4, función de activación *ReLU*
6	Capa *MaxPooling*	ventana de *MaxPooling* de tamaño 2x2
7	Capa Densa	100 neuronas, función de activación *ReLU*
8	Capa Densa	50 neuronas, función de activación *ReLU*
9	Capa Densa	50 neuronas, función de activación *sigmoide*
10	Capa Densa	6 neuronas, función de activación *softmax*

Tabla 4.4: Información de la estructura de la RNC utilizada en la prueba 2. En la primera columna se muestra el orden de las capas de RNC, en la segunda columna el tipo de capa y en la tercera columna la información de cada capa.

Número de capa:	Tipo de capa:	Información de los hiperparámetros:
1	Capa Convolucional	20 filtros de tamaño 5x5, función de activación *ReLU*
2	Capa *MaxPooling*	ventana *MaxPooling* de tamaño 2x2
3	Capa Convolucional	10 filtros de tamaño 4x4, función de activación *ReLU*
4	Capa *MaxPooling*	ventana de *MaxPooling* de tamaño 2x2
5	Capa Convolucional	10 filtros de tamaño 4x4, función de activación *ReLU*
6	Capa *MaxPooling*	ventana de *MaxPooling* de tamaño 2x2
7	Capa Convolucional	10 filtros de tamaño 4x4, función de activación *ReLU*
8	Capa *MaxPooling*	ventana de *MaxPooling* de tamaño 2x2
9	Capa Densa	100 neuronas, función de activación *ReLU*
10	Capa Densa	50 neuronas, función de activación *ReLU*
11	Capa Densa	50 neuronas, función de activación *sigmoide*
12	Capa Densa	50 neuronas, función de activación *sigmoide*
13	Capa Densa	6 neuronas, función de activación *softmax*

Tabla 4.5: Información de la estructura de la RNC utilizada en la prueba 3. En la primera columna se muestra el orden de las capas de RNC, en la segunda columna el tipo de capa y en la tercera columna la información de cada capa.

En la tabla 4.6 se muestra como se reduce el número de parámetros al aumentar la profundidad de la CNN, esto se debe a que, al aumentar el número de capas convolucionales se reduce la dimensión del muestreo que se introduce en la fase de capas densas, esto nos da como resultado un menor número de parámetros de conexión entre las capas convolucionales y las capas densas. En los tres casos se entrenó el modelo durante 150 épocas con 17 pasos por época, usando lotes de 100 datos en cada paso. Se utilizó la función como función de error la función de entropía cruzada categórica y el optimizador Adam.

Prueba:	Profundidad de la RNC:	Número de parámetros de la RNC:
1	7 Capas	395,996
2	10 Capas	95,846
3	13 Capas	28,006

Tabla 4.6: Se compara el numero de parámetros de cada prueba realizada. Se muestra como este disminuye al aumentar la profundidad de la RNC.

4.3.1. Resultados

A continuación, se muestran los resultados obtenidos para cada una de las pruebas realizadas.

- **Prueba 1:**

Para la primera prueba, se obtiene un mínimo para el error de validación en la época 32. En la figura 4.6 se observa como a partir de esta época de entrenamiento el error de validación empieza a crecer mientras que el error de entrenamiento sigue disminuyendo su valor. Si observamos la gráfica de la figura 4.7 vemos que la precisión de validación del modelo también es máxima alrededor de la época 32, a partir de la cual, esta precisión empieza a decrecer, aunque de forma menos abrupta que el aumento del error, este comportamiento indica que el modelo está siendo sobre entrenado.

- **Prueba 2:**

En la segunda prueba, se obtiene el mínimo de error de validación en la época 63. En la gráfica de la figura 4.8 se observa el comportamiento la evolución de los errores de entrenamiento y validación para esta prueba, como en el caso anterior observamos que el error de validación tiene una tendencia creciente a partir de la época 63 donde se encuentra su mínimo. En la gráfica de la figura 4.9 se observa la evolución de la precisión del modelo para los conjuntos de entrenamiento y validación. Se puede observar que a partir de la época 43 la precisión de validación presenta oscilaciones entre 70 % y 80 %, en particular para la época 63 la precisión obtenida es del 76 %.

- **Prueba 3:**

En la tercera prueba, obtenemos un mínimo de error de validación en la época 73. En las figuras 4.10 y 4.11 se observan la evolución del error y la precisión del modelo respectivamente, en donde observamos un comportamiento muy similar al caso anterior, con la excepción que en este caso el error de validación es menor que en el caso anterior. La

precisión de validación para la época 73 es del 78 %.

En resumen, nos damos cuenta que la eficiencia del modelo mejora conforme aumentamos el número de capas de la RNC, además de que el número de parámetros del modelo disminuye significativamente al aumentar la profundidad de la RNC. En la tabla 4.5 se compara la precisión que se obtiene en cada prueba para el mínimo de error de validación, así como la época en la cual se alcanza este valor mínimo. En las gráficas de las figuras 4.12 y 4.13 se comparan la evolución del error y la precisión de validación para cada una de las pruebas realizadas. Se puede observar como el error es menor para la prueba 3, además de que la evolución precisión es mayor para las pruebas 2 y 3 con respecto a la prueba 1, aunque para la prueba 3 es ligeramente mejor que para la prueba 2. Como la RNC de la prueba 3 tiene un mayor rendimiento, elegimos este modelo para predecir resultados sobre el conjunto de datos de prueba, los resultados se muestran en la figura 4.14, los puntos de color verde representan los resultados correctos y los de color rojo los resultados incorrectos, además la altura en la gráfica representa la probabilidad que predice la RNC para el resultado correcto con el cual se etiqueta cada dato. Este conjunto de datos de prueba contiene en total 536 ejemplos de EAS de los cuales el modelo clasifica correctamente 411, es decir, tiene una eficiencia aproximada del 76 %.

Prueba:	Época óptima:	Precisión de entrenamiento:	Precisión de validación:
1	32	82 %	71 %
2	63	84 %	76 %
3	73	82 %	78 %

Figura 4.5: La tabla muestra la precisión alcanzada por el modelo para los conjuntos de entrenamiento y validación en cada una de las pruebas, así como la época en la cual se obtiene una máxima precisión y un menor error para los datos de validación.

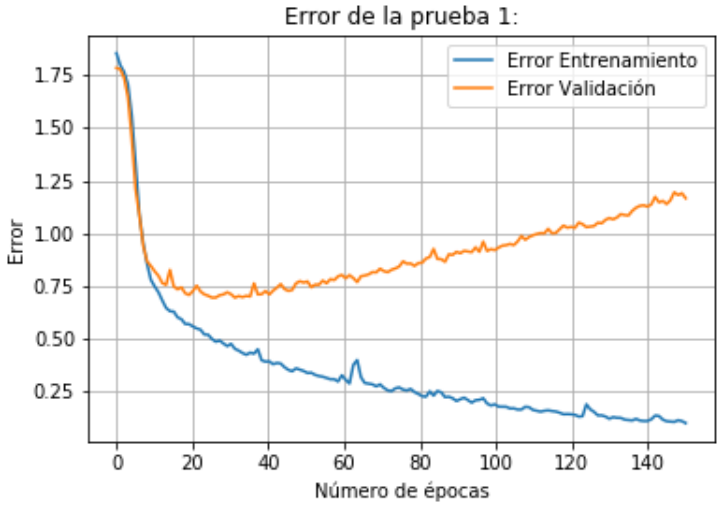


Figura 4.6: Evolución del error de entrenamiento y validación para la prueba 1. La linea azul representa el error de entrenamiento, la linea naranja representa el error de validación. El valor mínimo del error de validación se encontró en la época 32 del proceso de aprendizaje.

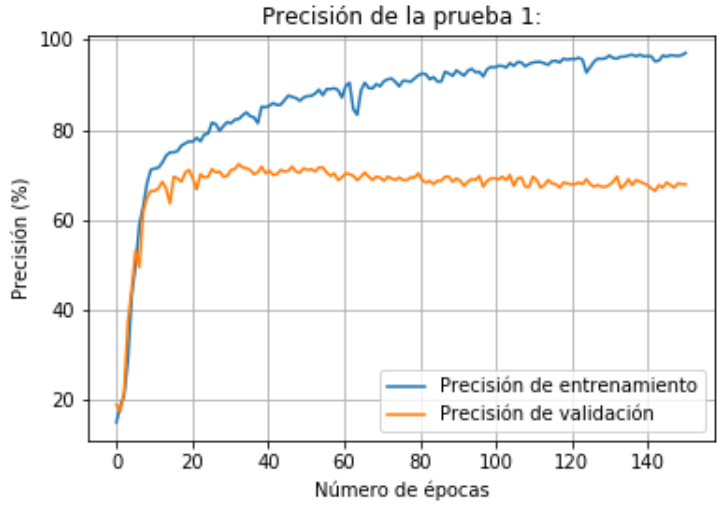


Figura 4.7: Precisión del modelo para la prueba 1, en cada época del proceso de entrenamiento. La linea azul representa la precisión de entrenamiento, la linea naranja representa la precisión de validación. Se alcanza una precisión de entrenamiento del 82 % y una precisión de validación máxima del 71 %.

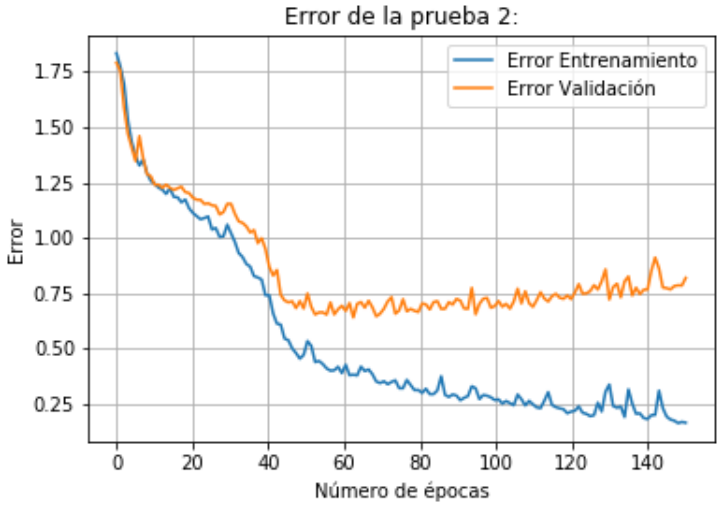


Figura 4.8: Evolución del error de entrenamiento y validación para la prueba 2. La linea azul representa el error de entrenamiento, la linea naranja representa el error de validación. El valor mínimo del error de validación se encontró en la época 63 del proceso de aprendizaje.

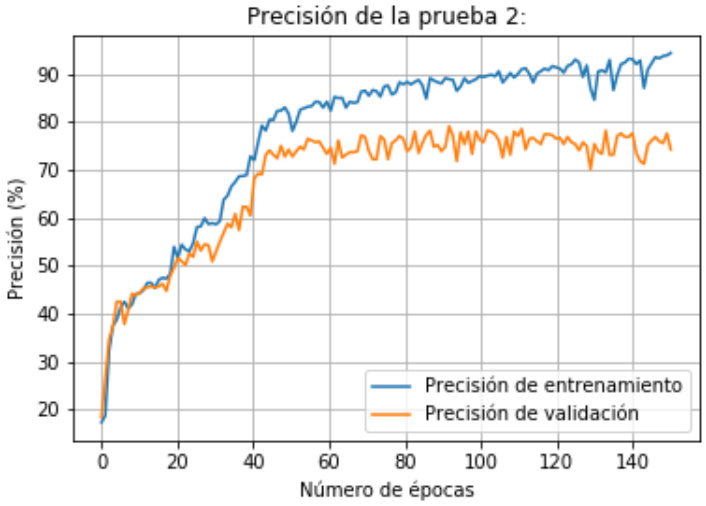


Figura 4.9: Precisión del modelo para la prueba 2, en cada época del proceso de entrenamiento. La linea azul representa la precisión de entrenamiento, la linea naranja representa la precisión de validación. Se alcanza una precisión de entrenamiento del 84 % y una precisión de validación máxima del 76 %.

Figura 4.10: Evolución del error de entrenamiento y validación para la prueba 3. La linea azul representa el error de entrenamiento, la linea naranja representa el error de validación. El valor mínimo del error de validación se encontró en la época 73 del proceso de aprendizaje.

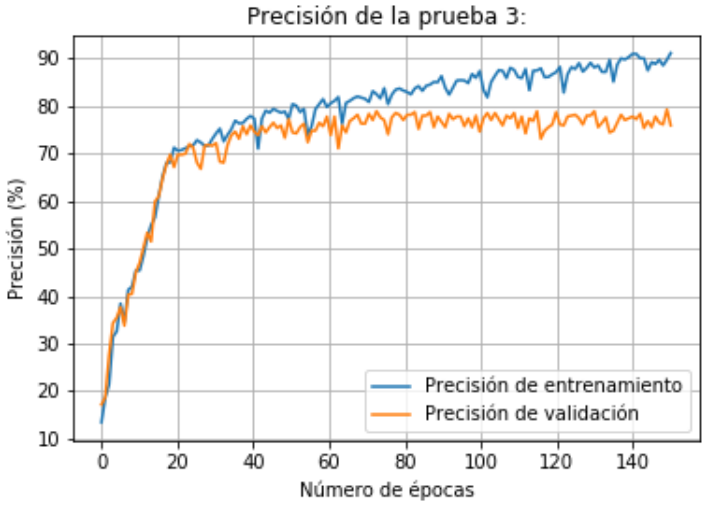


Figura 4.11: Precisión del modelo para la prueba 3, en cada época del proceso de entrenamiento. La linea azul representa la precisión de entrenamiento, la linea naranja representa la precisión de validación. Se alcanza una precisión de entrenamiento del 82 % y una precisión de validación máxima del 78 %.

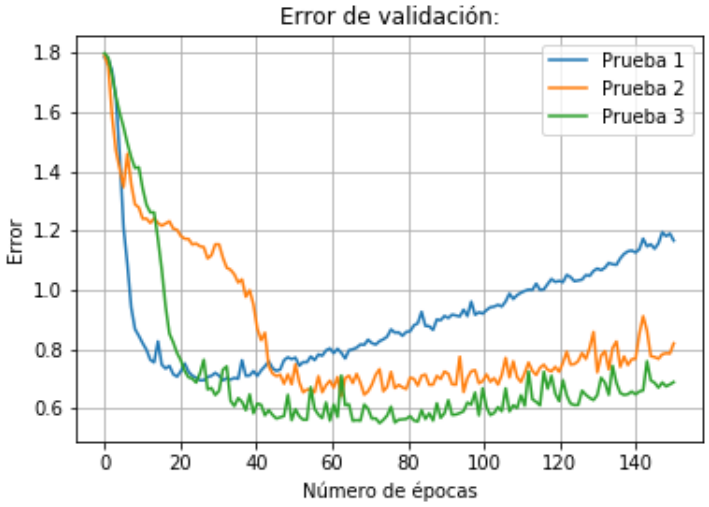


Figura 4.12: Error de validación en la clasificación de EAS en base a su energía, para cada una de las pruebas realizadas. La linea azul representa en error de la prueba 1, la linea naranja representa el error de la prueba 2, la linea verde representa el error de la prueba 3.

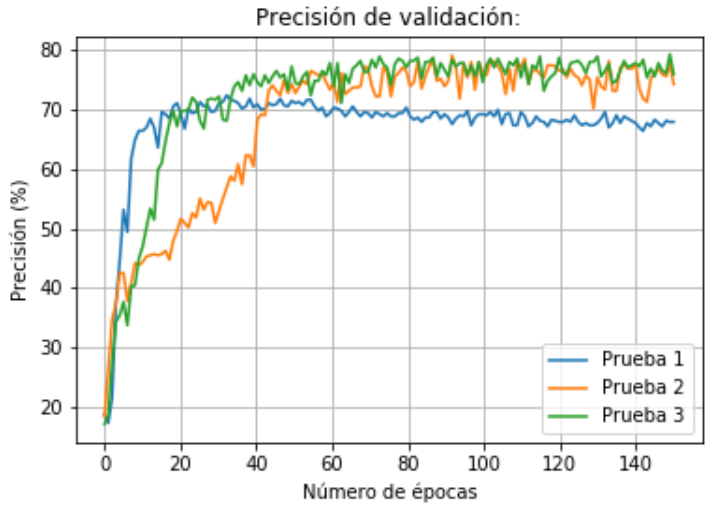


Figura 4.13: Precisión de validación de la clasificación de EAS en base a su energía, para cada una de las pruebas realizadas. La linea azul representa la precisión para prueba 1, la linea naranja representa la precisión para la prueba 2, la linea verde representa la precisión para la prueba 3. Se muestra que el modelo tiene una mayor eficiencia para la prueba numero 3.

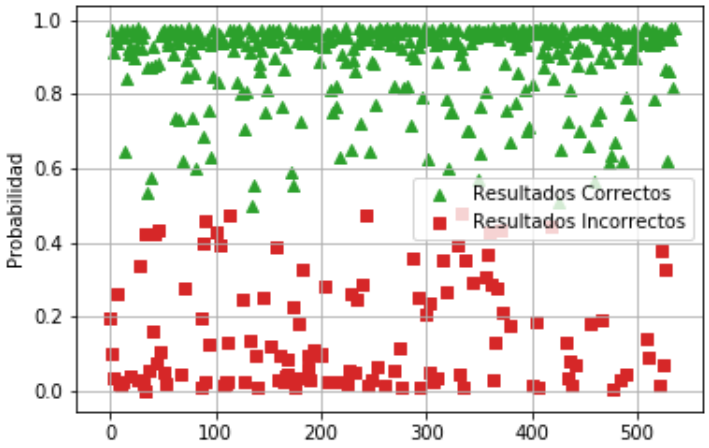


Figura 4.14: Predicción del modelo para la clasificación de energías. Los triángulos verdes representan los ejemplos clasificados correctamente, los cuadros rojos representan los ejemplos clasificados incorrectamente. El eje vertical representa la probabilidad que da como salida la RNC, para la categoría correcta de cada ejemplo.

Capítulo 5

Conclusiones y trabajo futuro

5.1. Conclusiones

Se implementó un modelo computacional de inteligencia artificial basado en RNC para la discriminación de EAS inducidos por hadrones y rayos gamma. Se abordaron dos casos de estudio, el primero consistió en discriminar el tipo de partícula primaria a partir de la huella al piso generada por el EAS inducido, para este problema se consiguió discriminar las partículas primarias con una eficiencia del 96 %. Para el segundo caso se planteó el problema de discernir entre el tipo de partícula primaria y su energía en un total de 6 categorías, para esta prueba se logró obtener una eficiencia máxima del 76 %.

Se mostró que el modelo propuesto en este trabajo tiene una muy alta eficiencia al momento de discernir entre el tipo de partícula, pero al momento de introducir la energía de la partícula primaria como una variable más en las categorías de clasificación, esta eficiencia baja del 96 % al 76 %, aun así, esta eficiencia es lo suficientemente buena teniendo en cuenta que se aumenta el número de categorías de 2 a 6.

Se puede concluir además que las RNC son un modelo muy eficiente en la discriminación

de EAS a partir de la huella al piso, debido al tipo de datos que estas huellas al piso generan.

5.2. Trabajo futuro

En el presente trabajo se consideraron solamente EAS verticales, además de rangos de energías inferiores a los 30 Tev. Para darle mayor rango de predicción a un modelo como el utilizado en este trabajo, se pueden considerar EAS con diferentes ángulos de inclinación y diferentes modelos de atmósferas, además de rangos de energías más altos. Esta generalización del problema queda pendiente como un posible trabajo futuro del cual el presente trabajo forma un precedente.

Otro posible salto para el mejoramiento del actual trabajo, es ir en la dirección de estudiar e implementar técnicas basadas en otros tipos de modelos de inteligencia artificial diferentes de las RNC, con la intención de comparar el rendimiento que estos ofrecen respecto al problema planteado.

Por último, al considerar más variables en la generación de simulaciones de EAS, será necesario generar más datos de entrenamiento, para lo cual aumenta la cantidad de operaciones de computo no solo para generar las simulaciones sino también para entrenar los modelos, en este sentido es necesario el utilizar técnicas de computo paralelo, con la finalidad de disminuir el tiempo de computo de los algoritmos.

Bibliografía

[1] D.H. Perkins. *Particle Astrophysics*. Oxford University, 2009.

[2] Peter K. F. Grieder. *Exensiver Air Showers*. Springer, University of Bern, Physikalisches Institut, first edition, 2010.

[3] E B Postnikov, A P Kryukov, S P Polyakov, D A Shipilov, and D P Zhurov. Gamma/hadron separation in imaging air cherenkov telescopes using deep learning libraries tensorflow and pytorch. *Journal of Physics: Conference Series*, 1181:012048, Feb 2019.

[4] D. Heck and T. Pierog. *Extensive Air Shower Simulations with CORSIKA: A User's Guide*. Institute for Nuclear Physics Karlsruhe Institute for Technology (KIT), Hermann-von-Helmholtz-Platz 176344 Eggenstein-Leopoldshafen, 1 edition, 9 2013. Version 7.4xxx from September 3, 2013.

[5] C.M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[6] Hinton G.E. Nair, V. *Rectified linear units improve restricted boltzmann machines.* Proceedings of the 27th International Conference on Machine Learning, pp. 807–814, 2010.

[7] M.D. Eggers, T.S. Khuon, and Lincoln Laboratory. *Learning Algorithms for the Multilayer Perceptron*. Technical report (Lincoln Laboratory). Massachusetts Institute of

Technology, Lincoln Laboratory, 1988.

[8] Carnegie-Mellon University. Computer Science Department and B.A. Pearlmutter. *An Investigation of the Gradient Descent Process in Neural Networks*. Research paper. Carnegie-Mellon University. Department of Computer Science, 1996.

[9] R. Kidambi. *Stochastic Gradient Descent for Modern Machine Learning: Theory, Algorithms and Applications*. University of Washington Libraries, 2019.

[10] *Adaptive Subgradient Methods for Online Learning and Stochastic Optimization*. Computer Science Division University of California, Berkeley Berkeley, CA 94720 USA.

[11] Goran Nakerst, John Brennan, and Masudul Haque. Gradient descent with momentum — to accelerate or to super-accelerate?, 2020.

[12] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014.

[13] Y. Chen and University of Waterloo. School of Computer Science. *Convolutional Neural Network for Sentence Classification*. University of Waterloo, 2015.

[14] E. Bourbeau, T. Capistrán, I. Torres, and E. Moreno. New gamma/hadron separation parameters for a neural network for hawc, 2017.

[15] D. Heck, J. Knapp, J. N. Capdevielle, G. Schatz, and T. Thouw. CORSIKA: A Monte Carlo code to simulate extensive air showers. 1998.