UNIVERSIDAD NACIONAL DE ASUNCIÓN
FACULTAD DE INGENIERÍA

Resumen Ejecutivo

Página oficial: www.ing.una.py

Segmentación de instancia aplicada a la detección de partículas capturadas por los detectores LArTPC del Experimento DUNE

José Toledo[a,1], Felipe Espinoza[a,2], Dr. Diego Stadler[b,3], Dr. Jorge Molina[b,4], Prof. Ing. Gustavo Verón[b,5]

[a]*Alumno. Facultad de Ingeniería, Universidad Nacional de Asunción, Paraguay*
[b]*Tutor. Facultad de Ingeniería, Universidad Nacional de Asunción, Paraguay*

Palabras clave

Segmentación Instancias
LArTPC
DUNE
Aprendizaje Profundo
SCNN

Resumen

Los detectores de argón líquido son utilizados para la detección y el estudio de los neutrinos, estos detectores se caracterizan por su alta precisión de imágenes de aproximadamente 1 milímetro por píxel que permiten analizar la interacción de los neutrinos con mucho detalle. A pesar de la alta resolución de imágenes y recopilación de información producida mediante este tipo de detectores, existe el desafío de reconstrucción precisa de la información de cada partícula de forma individual a partir de las imágenes 2D y 3D generadas, el uso de aprendizaje automático ha presentado avances significativos tanto para la detección de objetos y la segmentación semántica y de instancias, donde el esfuerzo actual de los investigadores va dirigido en la búsqueda de nuevas implementaciones con las mejores prestaciones a nivel de desempeño tanto a nivel computacional como en la correcta clasificación y agrupamiento de las partículas. En este estudio utilizamos datos creados por simulación de eventos capturados por detectores de argón líquido de cámara de proyección temporal, que consisten en 100.000 eventos representados en matrices dispersas de 3 dimensiones y un tamaño espacial de 512 elementos, y con ello realizamos una comparación entre algunas de las arquitecturas actuales utilizadas para el procesamiento de imágenes. Utilizamos dos muestras diferentes de 10.000 eventos, una para el entrenamiento y la otra para el testeo de las redes mencionadas y con los datos obtenidos creamos una tabla donde comparamos las métricas de precisión, pureza y eficiencia de las arquitecturas seleccionadas.

Keywords

Instance Segmentation
LArTPC
DUNE
Deep Learning
SCNN

Abstract

Liquid argon detectors are used for the detection and study of neutrinos. These detectors are characterized by their high precision images of approximately 1 millimeter per pixel, which allow the interaction of neutrinos to be analyzed in great detail. Despite the high resolution of images and collection of information produced by this type of detectors, there is the challenge of accurately reconstructing the information of each particle individually from the generated 2D and 3D images, the use of machine learning has presented significant advances both for the detection of objects and the semantic and instance segmentation, where the current effort of the researchers is directed in the search for new implementations with the best features in terms of performance, both at the computational level and in the correct classification and particle clustering. In this study we use data created by simulation of events captured by time projection chamber liquid argon detectors, consisting of 100,000 events represented in 3-dimensional sparse matrices and a spatial size of 512 elements, and with this we make a comparison between some of the current architectures used for image processing. We use two different samples of 10,000 events, one for training and the other for testing the mentioned networks and with the data obtained we create a table where we compare the precision, purity and efficiency metrics of the selected architectures.

1. Introducción

En el campo de la visión artificial, la clasificación de imágenes consiste en detectar clases de objetos que se encuentran en una imagen y también proporciona la ubicación de éstos en forma de un cuadro delimitador en la misma imagen. La segmentación semántica logra una detección a nivel de píxeles, donde predice y etiqueta los píxeles de acuerdo a la

[1]email: joseotoledo@gmail.com
[2]email: felipespinozaa@gmail.com
[3]email: dstalder@ing.una.py
[4]email: jmolina@ing.una.py
[5]email: gveron@ing.una.py

clase del objeto logrando definir los límites de estos objetos dentro de la imagen. Agregando a éste último la capacidad de distinguir diferentes objetos de la misma clase, es lo que se conoce como segmentación de instancia.

Las redes neuronales convolucionales (CNN, por sus siglas en inglés) se han convertido en el estándar en las técnicas de aprendizaje automático o machine learning (ML) en el campo de visión artificial. Actualmente, una de las aplicaciones de los métodos de aprendizaje profundo o deep learning es en los experimentos que estudian las oscilaciones de los neutrinos, incluyendo los que utilizan los detectores de argón líquido, Liquid Argon Time Project Chamber (LArTPC).

Los LArTPC son un tipo de detector de partículas, que puede generar imágenes en 2D o 3D de la trayectoria de partículas cargadas con una alta resolución (∼ mm/pixel) a través de muchos metros de detección volumétrica. Entre los experimentos actuales y futuros que utilizan los detectores LArTPC se incluyen MicroBooNE, Short Baseline Near Detector (SBND), Imaging Cosmic And Rare Underground Signals (ICARUS) y el Deep Underground Neutrino Experiment (DUNE). El volumen de Argón líquido que utilizan los detectores de estos experimentos son de 90, 112, 600 y 40.000 toneladas respectivamente. Aunque los LArTPC producen imágenes de alta resolución con información detallada sobre la topología de eventos, los desafíos restantes incluyen la reconstrucción precisa de la información de partículas individuales, así como la inferencia de la energía y el "sabor" de los neutrinos a partir de imágenes 2D y 3D.

En este trabajo nos enfocaremos en identificar los siguientes tipos de partículas: Minimum Ionizing Particles (MIP), Heavily Ionizing Particles (HIP), Electromagnetic Shower, Delta-Rays y Michel electrons utilizando las arquitecturas U-ResNet y Submanifold Sparse Convolutional Networks (SSCN). La detección de éstas dentro de los detectores LArTPC indica una interacción de los neutrinos con los átomos de Argón, por lo cual permite un estudio indirecto de los neutrinos.

2. Objetivos

2.1. General

- Realizar la segmentación de instancias de las partículas capturadas por el tipo de detector LArTPC.

2.2. Específicos

- Analizar los criterios físicos utilizados en los detectores de LArTPC.
- Obtener el conjunto de datos de entrenamiento necesario para la segmentación de imágenes.
- Analizar las arquitecturas del estado del arte y realizar la segmentación de instancias de las partículas capturadas por el detector LArTPC.
- Comparar e identificar el algoritmo de aprendizaje profundo con mayor desempeño.

3. Fundamentos teóricos

3.1. Neutrinos

Los neutrinos son partículas elementales y pertenecen al grupo de los leptones. Fueron teorizadas por primera vez en 1930 para explicar una pequeña pérdida de energía de otras interacciones de partículas, y se detectaron por primera vez en 1956.

Los científicos han descubierto que hay tres tipos (o "sabores") de partículas de neutrinos, que estas partículas tienen masa (al contrario de lo que se creía cuando fueron teorizadas) y que oscilan entre los tres sabores mientras viajan. Los descubrimientos en física de neutrinos han dado lugar a múltiples premios Nobel, y no hay indicios de que estas partículas se quedarán sin secretos para revelar en el corto plazo [Zuber (2020)].

Los neutrinos son la segunda partícula mas abundante en todo el universo, siendo superados solamente por los fotones. Cada vez que los núcleos atómicos se unen (como en el sol) o se separan (como en un reactor nuclear) se producen los neutrinos. El mayor generador de neutrinos, que se conoce, en nuestro sistema solar es el sol.

La única forma en que los neutrinos interactúan con su alrededor es a través de la fuerza de gravedad y la fuerza nuclear débil. Esta fuerza nuclear débil es importante solo a muy cortas distancias y al tener una masa tan pequeña la fuerza de gravedad también es muy débil. Por esta razón y porque son eléctricamente neutros, son muy raras las interacciones del neutrino con la materia, lo que significa que pequeños neutrinos pueden atravesar los átomos de objetos masivos sin interactuar.

La mayoría de los neutrinos atraviesan la Tierra sin interactuar en absoluto. Para aumentar las probabilidades de verlos, los científicos deben utilizar enormes detectores y crear rayos intensos de neutrinos en los aceleradores de partículas.

3.2. El experimento DUNE

El Deep Underground Neutrino Experiment (DUNE) es un experimento internacional de vanguardia para la ciencia de los neutrinos y los estudios de desintegración de protones. Los descubrimientos durante el último medio siglo han puesto a los neutrinos, las partículas de materia más abundantes en el universo, en el centro de atención para futuras investigaciones sobre varias preguntas fundamentales sobre la naturaleza de la materia y la evolución del universo, preguntas que DUNE buscará responder [collaboration *et al.* (2020)].

DUNE constará de dos detectores de neutrinos por los cuales atravesará el haz de neutrinos más intenso del mundo. Un detector registrará las interacciones de las partículas cerca de la fuente del rayo, en el Laboratorio Nacional del Acelerador Fermi en Batavia, Illinois. Un segundo detector, mucho más grande, se instalará a más de un kilómetro bajo tierra en el Laboratorio de Investigación Subterráneo de Sanford en Lead, Dakota del Sur, a 1.300 kilómetros de la fuente. Estos detectores permitirán a los científicos buscar nuevos fenómenos subatómicos y potencialmente transformar nuestra comprensión de los neutrinos y su papel en el universo.

3.2.1. Los detectores LArTPC

El detector de neutrinos de Argón líquido de cámara de proyección temporal o LArTPC por sus siglas en inglés [Rubbia (1977)] es el tipo de detectores de neutrinos seleccionado para ser utilizado en la colaboración DUNE. El argón líquido es ventajoso como medio sensible por varias razones, al ser éste un elemento noble tiene una electronegatividad practicamente nula, por lo cual, los electrones que se producen por la radiación ionizante no serán absorbidos mientras son desplazados hacia la lectura del detector. El argón también centellea cuando pasa una partícula cargada de energía, liberando una cantidad de fotones

de centelleo que es proporcional a la energía depositada en el argón por la partícula que pasa. Otra ventaja de éste elemento es costo relativamente económico, lo que hace que los proyectos a gran escala sean económicamente viables. Sin embargo, una de las principales motivaciones para usar argón líquido como medio sensible es su densidad. El argón líquido es uno de los medios más densos utilizado en este tipo de detectores, lo que aumenta la probabilidad de que una partícula interactúe con éste. Esta característica es particularmente útil en la física de neutrinos, donde las secciones transversales de interacción neutrino-nucleón son pequeñas.

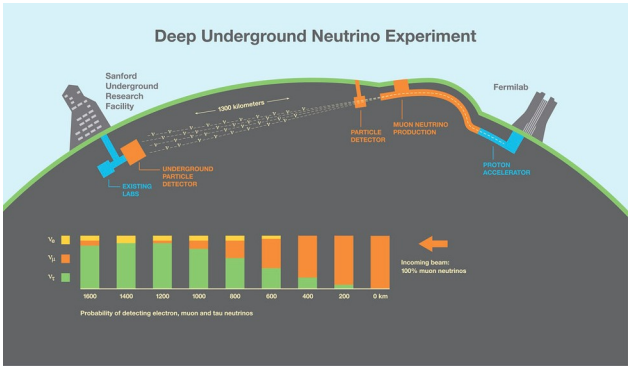

Fig. 1: Esquema de operación del DUNE y la oscilación de neutrinos. **Fuente:** dunescience.org

Las imágenes capturadas por estos detectores son luego procesadas y utilizadas para reconstruir los eventos asociados a cada detección realizada y de tal manera por estudiar la interacción de los neutrinos. Para facilitar el trabajo de interpretación de los eventos utilizamos herramientas computacionales que permiten identificar las partículas producidas en cada evento así como su trayectoria.

3.3. Inteligencia Artificial y sus subconjuntos

3.3.1. Inteligencia Artificial

La inteligencia artificial, o "IA", se refiere a la capacidad de una máquina de imitar las funciones cognitivas que hasta ahora se asociaban exclusivamente a los humanos, es decir: una máquina inteligente, es aquel agente flexible capaz de percibir su entorno para llevar a cabo sus acciones maximizando sus probabilidades de éxito.

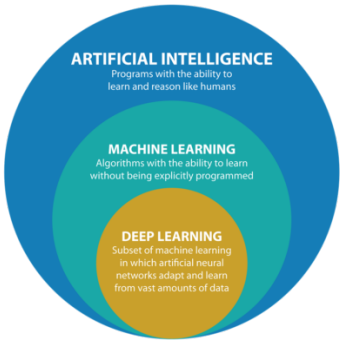

Fig. 2: Inteligencia artificial, Aprendizaje Automático, Aprendizaje Profundo. **Fuente:** Qubole, "Deep Learning: The latest trend in AI and ML"

Lo que diferencia a la Inteligencia Artificial de otros programas de ordenador es que no hay que programarla específicamente para cada escenario. Podemos enseñarle cosas (aprendizaje automático), pero también puede aprender por sí mismo (aprendizaje profundo). Si bien existen múltiples variantes de cada uno, se pueden definir de manera general de esta forma:

- Inteligencia Artificial: una máquina que es capaz de imitar el razonamiento humano.
- Aprendizaje Automático: un subconjunto de la Inteligencia Artificial donde las personas "entrenan" a las máquinas para reconocer patrones basados en datos y hacer sus predicciones.
- Aprendizaje Profundo: un subconjunto del Aprendizaje Automático en el que la máquina es capaz de "razonar" y sacar sus propias conclusiones, aprendiendo por sí misma.

3.4. Detección de objetos y segmentación

El aprendizaje profundo ha tenido mucho éxito al trabajar con imágenes como datos y actualmente se encuentra en una etapa en la que funciona mejor que los humanos en múltiples casos de uso. Los problemas más importantes que los humanos han estado interesados en resolver con la visión por computadora son clasificación de imágenes, detección y segmentación de objetos en el orden creciente de su dificultad.

En la simple tarea de la clasificación de imágenes, solo nos interesa obtener las etiquetas de todos los objetos que están presentes en una imagen. En la detección de objetos damos un paso más y tratamos de saber, junto con todos los objetos que están presentes en una imagen, la ubicación en la que están presentes los objetos con la ayuda de cuadros delimitadores. La segmentación de imágenes lo lleva a un nuevo nivel al tratar de averiguar con precisión el límite exacto de los objetos en la imagen.

Segmentación semántica La segmentación semántica es el proceso de clasificar cada píxel que pertenece a una etiqueta en particular. No se diferencia para diferentes instancias del mismo objeto. Por ejemplo, si hay dos gatos en una imagen, la segmentación semántica da la misma etiqueta a todos los píxeles de ambos gatos.

Segmentación de instancias La segmentación de instancias difiere de la segmentación semántica en el sentido de que otorga una etiqueta única a cada instancia de un objeto particular en la imagen. Por ejemplo, si tenemos 3 perros en una imagen, aunque éstos pertenezcan todos a la misma clase "perro" se les asignan colores diferentes, es decir, etiquetas diferentes. Con la segmentación semántica a todos se les habría asignado el mismo color.

4. Metodología aplicada

4.1. Trabajo Base

El enfoque del trabajo base [Koh *et al.* (2020)] realiza la segmentación semántica y la segmentación de instancias utilizando dos ramas basadas en el tipo de red U-net. Las redes U-net tienen dos partes: la primera mitad reduce el tamaño espacial de la imagen de entrada con varios bloques de convolución . Utiliza un codificador multietapa para disminuir las dimensiones espaciales aumentando el número de canales. Cada etapa extrae las "features" (características o rasgos en inglés) relevantes de las imágenes. Luego, la segunda mitad de la red aplica un decodificador para realizar el proceso inverso, concatenando las características previamente guardadas en cada etapa para reconstruir una nueva imagen.

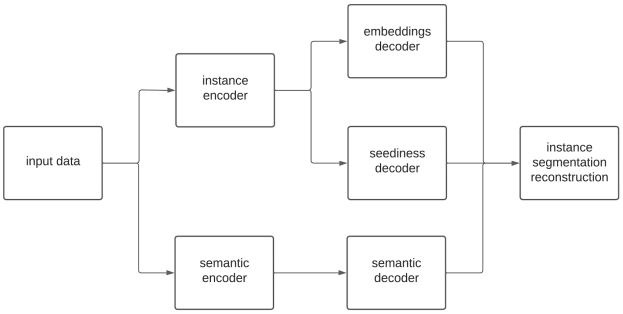

Fig. 3: Arquitectura de la red. **Fuente:** Elaboración propia.

La rama de segmentación de instancias toma la salida del "codificador de instancias" y la procesa en dos decodificadores diferentes. El "seedines decoder" identifica los centroides de cada instancia, dando como salida un tensor de dimensión $N \in \mathbb{R}$ donde N es el número de elementos de la matriz de entrada y su valor s_i corresponde a la probabilidad que cada elemento es un centroide. El "embeddings decoder" agrupa los elementos de cada instancia minimizando la distancia a su centroide correspondiente, dando como salida un tensor de dimensiones $N \in \mathbb{R}^4$ donde, para cada elemento, los tres primeros valores corresponden a su ubicación en el espacio embebido, el cual es una normalización del tamaño espacial a un cubo con valores que van del -1 al 1, mientras el ultimo valor corresponde al margen que es utilizado como desviación estándar en una función gaussiana para la determinación de cada instancia.

Para la reconstrucción final de cada instancia, se calcula la probabilidad de que cada punto pertenezca a cada instancia, para esto definimos un par de valores para cada clase: s_0 y p_0. El primer valor, s_0, establece la mínima probabilidad aceptable para que un punto sea centroide de la clase correspondiente (salida del "seediness decoder"), descartando aquellos que no satisfacen. El segundo valor, p_0, establece la mínima probabilidad aceptable para que un punto pertenezca a una instancia.

Para esto se itera a través de cada clase, separando todos los puntos según su clasificación en la rama de segmentación semántica, luego se elige el punto con mayor valor de s_i y con su correspondiente margen (σ) se calcula la probabilidad p_i de que cada punto pertenezca a la instancia asociada a esta centroide utilizando una función gaussiana con centro en el punto y desviación σ. Todos los puntos con p_i mayor a p_0 se agrupan en una instancia y se vuelve a elegir iterativamente el mayor valor de s_i hasta que todos los puntos estén asociados a una instancia o hasta que el mayor valor de s_i sea menor a s_0.

Función de perdida

La red completa analiza cuatro variables distintas que son suministradas al reconstructor de instancias, siendo estas para cada punto: la clasificación semántica, su ubicación en el espacio embebido, su margen y la probabilidad de ser un centroide.

Para la clasificación semántica, utilizamos la función de perdida CrossEntropy, definida como:

$$L_{semantic} = -\sum_{i=1}^{n} t_i.log(p_i) \quad (1)$$

Donde, n es el número de clases (cinco para este trabajo), t es la etiqueta real y p la probabilidad exponencial normalizada predicha por la red, siendo esta la función de perdida para la rama semántica.

Para la ubicación en el espacio embebido primero definimos para cada punto i la probabilidad de que pertenezca a la instancia k:

$$p_{ik} = e^{\frac{\|x_{pi}-c_k\|^2}{2\sigma_k^2}} \quad (2)$$

Donde x_pi son las coordenadas en el espacio embebido predichas por la red, c_k es el centroide real de la instancia k y σ_k es el margen real de la instancia k. Con este valor definimos la perdida de máscara que realiza un CrossEntropy binario para cada instancia:

$$L_{mask} = \frac{1}{K}\sum_{k=1}^{K}(-\frac{1}{N}\sum_{i=1}^{K}[y_{ik}.log(p_{ik})+(1-y_{ik}).log(1-p_{ik})]) \quad (3)$$

Donde, $y_{ik} = 1$ si el punto i pertenece a la instancia k, e igual a cero en el caso contrario, N es el número total de puntos y K el número total de instancias. Ésta perdida penaliza los puntos alejados de su centroide, atrayéndolos hacia el mismo. Como complemento tenemos la perdida inter-cluster que penaliza las pequeñas distancias entre distintas instancias:

$$L_{inter} = \frac{1}{K \times (K-1)}\sum_{i<j}[2\delta_d - \|c_i - c_j\|]^2 \quad (4)$$

Donde δ_d es un margen inter-cluster y c_i y c_j son los centroides de 2 instancias en el espacio embebido.

Para el margen, definimos la perdida de suavizado para mantener el valor de σ consistente a través de los puntos de la misma instancia:

$$L_{smoothing} = \frac{1}{N_k}\sum_{i=1}^{N_k}\|\sigma_i - \frac{1}{N_k}\sum_{i=1}^{N_k}\sigma_i\| \quad (5)$$

Donde N_k representa el número de puntos de la instancia k y σ_i es el margen del punto i. Estas tres perdidas corresponden al "Embeddings decoder":

$$L_{emb} = L_{mask} + L_{inter} + L_{smoothing} \quad (6)$$

Para la probabilidad de ser un centro, que corresponde al "Seediness decoder" definimos:

$$L_{seed} = \frac{1}{K}\sum_{k=1}^{K}(-\frac{1}{N_k}\sum_{i\in C_k}\|s_i - p_{ik}\|^2) \quad (7)$$

Donde C_k representa la instancia k y s_i la probabilidad de ser un centro. Finalmente, definimos la perdida de la rama de instancias como:

$$L_{instance} = L_{emb} + L_{seed} \quad (8)$$

4.2. Dataset

El dataset utilizado fue extraído de "PILArNet: Public Dataset for Particle Imaging Liquid Argon Detectors in High Energy Physics" [Adams *et al.* (2020)], está conformado por cien mil eventos principales, éstos se encuentran representados en matrices dispersas de 3 dimensiones y un tamaño espacial de 512 elementos. Cada punto cuenta con diversas características, de las cuales se utilizan la energía depositada, código pdg, clasificación semántica y clasificación en instancias.

Cuenta con 5 clases semánticas que son:

- Partículas altamente ionizantes (HIP, del inglés Highly Ionizing Particles)
- Partículas mínimamente ionizantes (MIP, del inglés Mininum Ionizing Particles)
- Lluvia electromagnética (Shower, del inglés Electromagnetic Shower)
- Rayos-delta (Deltas, del inglés Delta Rays)
- Electrón Michel (Michel)

De los cuales ochenta mil eventos fueron utilizados para el entrenamiento, 19.800 destinados al proceso de validación y los restantes 200 son para el testeo final.

En la Figura 4, tenemos una imagen de muestra del dataset donde podemos observar que en este evento capturado se encuentran las 5 clases de interés para este estudio. También podemos recalcar que éstas clases están separadas por instancias, debido a que estamos utilizando de ejemplo las etiquetas de instancias reales de las partículas.

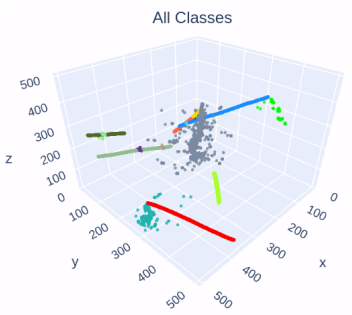

Fig. 4: Muestra del dataset donde se observan las 5 clases de interés. **Fuente:** Elaboración propia.

4.3. Arquitectura de la red propuesta

Para la rama de Segmentación Semántica, la red U-Net cuenta con cuatro niveles, denominados "strides" (pasos o zancadas en inglés), o lo que es lo mismo, cuenta con tres operaciones de reducción de muestreo denominadas "downsample", donde en cada uno se va reduciendo la dimensión espacial a la mitad. En la primera etapa, hay ocho filtros, aumentando en la misma cantidad en cada nivel (Figura 5).

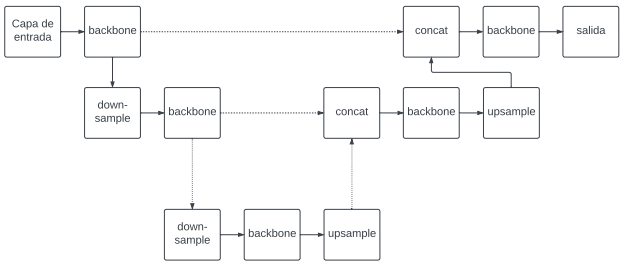

Fig. 5: Arquitectura U-Net empleada. **Fuente:** Elaboración propia.

Para la rama de Segmentación de instancias la red U-net cuenta con 6 strides y 32 filtros en el primer nivel, aumentando en esta cantidad en cada nivel.

La capa de entrada, o "input layer", es la que genera el tensor a partir de los datos de entrada que son brindados en una tupla (coordenadas, features). Está compuesta por dos módulos, en primer lugar se encuentra el "sparseconvnet InputLayer" [Graham *et al.* (2018)] el cual genera el tensor a partir de la tupla mencionada.

Luego tenemos el "Submanifold Convolution", éste módulo realiza la primera convolución de la red, la cual genera los treinta y dos filtros a partir de "kernels" de dimensión 3. Donde un kernel se puede interpretar como la vecindad o conjunto de pixeles que se toman por cada pixel que representa a la imagen a convolucionar.

El backbone es una estructura de red compuesta de uno o varios bloques de una red neuronal clásica, cantidad que depende del parámetro "reps" el cual define cuantas veces se repite un bloque en un nivel. El backbone utilizado como base en este trabajo está compuesto por bloques del tipo ResNet, explicados anteriormente, y con el parámetro reps=2.

El bloque ResNet (Figura 6) utilizado en este trabajo esta compuesto por dos normalizaciones y dos convoluciones. Para la normalización, utilizamos el módulo "sparseconvnet BatchNormLeakyReLU" el cual utiliza una activación del tipo "LeakyReLU". Por otro lado, las dos convoluciones son realizadas por el módulo "SubmanifoldConvolution".

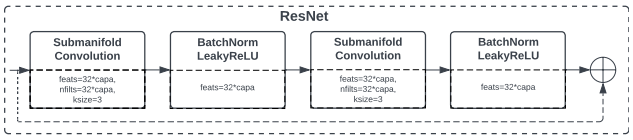

Fig. 6: Bloque ResNet. **Fuente:** Elaboración propia.

El bloque downsample realiza una normalización y una convolución, en el caso de este bloque, se utiliza el módulo "sparseconvnet Convolution". A diferencia del módulo "SubmanifoldConvolution", que no aumenta la cantidad de puntos dispersos y mantiene el tamaño espacial, la operación de convolución realizada en este bloque aumenta la cantidad de puntos lo cual permite realizar la disminución espacial para pasar a la siguiente capa y también, a su vez, permite usar su operación inversa con el módulo "sparseconvnet Deconvolution".

El bloque upsample realiza la operación inversa al bloque downsample con el módulo sparseconvnet Deconvolution.

El bloque concat cuenta con dos módulos "sparseconvnet JoinTable" y "NetworkInNetwork", el primero realiza una concatenación directa entre los features de los tensores que tiene como entrada, mientras que el segundo realiza una convolución 1×1 sin variar el tamaño espacial, pero reduciendo el número de filtros al correspondiente de su capa.

4.4. Ajuste fino de la arquitectura

4.4.1. Experimento 1: Selección de bloques

Para éste experimento nos centramos en la rama de instancias de la red, comparamos bloques de distintas arquitecturas de red y seleccionamos aquella con los mejores resultados para, finalmente, analizar el mismo bloque en la rama semántica.

Para la realización del primer experimento además del bloque base ResNet, seleccionamos tres bloques de red diferentes: ResNeXt, ResNeXtSE y MBConv6 analizando además 2 variaciones de esta última que serán detallados a continuación. Y realizamos el entrenamiento con los mismos parámetros definidos en el trabajo base.

Bloque ResNeXt (Figura 7): con base en el bloque ResNet, pero tomando la estrategia de Inception, el bloque ResNeXt realiza una convolución 1×1 dividiendo primeramente a la mitad el número de filtros, luego realiza un par de normalización y convolución, pero este último se realiza en grupos de a 4 filtros que son concatenados de vuelta al final de la convolución. A continuación, se recupera el número original de filtros realizando una convolución 1×1 para finalizar sumando con el tensor residual.

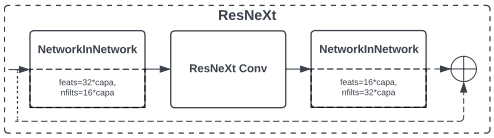

Fig. 7: Bloque ResNeXt. **Fuente:** Elaboración propia.

Bloque ResNeXtSE (Figura 8): siendo una variación del bloque ResNeXt, el ResNeXtSE agrega el bloque SE al final de la última convolución 1×1.

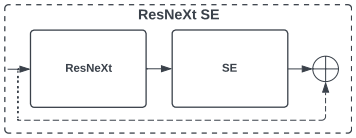

Fig. 8: Bloque ResNeXtSE. **Fuente:** Elaboración propia.

El Bloque SE [Hu *et al.* (2018)] toma su nombre de los términos squeeze y excitation. La primera hace referencia a la reducción espacial que toma el tensor de 512 a 1, esto se consigue con un "AveragePooling" global, o lo que es lo mismo, con poolsize 512 y stride 512. La segunda refiere a una "excitación" del tensor al entrar a una red lineal de dos convoluciones, la primera con una reducción de filtros a ¼ del actual con un "BatchNormLeakyRelu" y la segunda devolviendo el número de filtros, pero con una normalización usando una sigmoide como activación. Esto da como resultado un tensor con features de $1 \times n$-Filtros siendo n-Filtros igual a 32×layer que es multiplicado con el tensor residual con lo que se consigue dar pesos a los distintos filtros del tensor.

Bloque MBConv6 (Figura 9): Usando también como base al bloque ResNet, este bloque también cuenta al inicio con una convolución 1×1, pero esta vez duplica la cantidad de filtros, pasa por un BatchNormLeakyRelu y una convolución que esta vez se realiza de forma independiente en cada filtro (agrupado de a uno), pasa de nuevo por una normalización y luego por un bloque SE que es igual al del bloque ResNeXtSE a diferencia del número de filtros, luego se realiza otra convolución 1×1 para devolver a su número inicial de filtros y por último se pasa por un Dropout para descartar la mitad de los resultados, ayudando a evitar el overfitting.

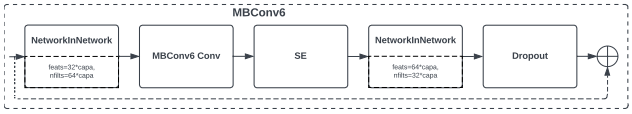

Fig. 9: Bloque MBConv6. **Fuente:** Elaboración propia.

Las otras dos variaciones, que también utilizaremos, del bloque MBConv6 son removiendo el bloque SE, en el primer caso, y removiendo tanto el bloque SE como el Dropout para el segundo.

Los hiperparámetros utilizados para estas pruebas son los que fueron definidos en el trabajo base, los cuales son: Número de filtros = 32, Tamaño de los filtros en la Input Layer = 3, Número de Strides = 6, Número de bloques en cada Backbone = 2, Learning Rate = 0.005.

4.4.2. Experimento 2: Ajuste de hiperparámetros

Experimento 2.1: Ajuste manual:

Finalizado el experimento 1, seleccionaremos la arquitectura con mejores resultados para continuar con las siguientes pruebas. Para analizar el comportamiento de la red con distintos hiperparámetros, realizamos un "grid search manual", un grid search consiste en un método de búsqueda que toma en cuenta diferentes combinaciones de hiperparámetros y se anotan los resultados en una cuadrícula para facilitar la comparación. Para realizar este análisis se tienen en cuenta los siguientes hiperparámetros: número y tamaño de filtros, número de strides y el número de bloques en cada backbone.

Experimento 2.2: Ajuste automatizado:

En este experimento utilizamos el framework Optuna, estableciendo los resultados de este como el modelo a realizar un entrenamiento con el dataset de entrenamiento completo (80.000 muestras). Para esto dividimos el estudio en 3 partes, realizando 5 épocas de entrenamiento con 4000 datos seleccionados aleatoriamente en batches de 16 del total de 100.000 muestras.

Esta última prueba es dividida en tres partes, la primera parte realizará la búsqueda de los parámetros que influyen en el tamaño de la red, la segunda efectuará una búsqueda de pesos para las perdidas en la rama de segmentación en instancias utilizando los resultados de la primera parte. Por último, la tercera buscará configurar el optimizador, eligiendo entre 3 variaciones del optimizador "Adam" y ajustando la tasa de aprendizaje.

5. Análisis e interpretación de los resultados

5.1. Métricas de Evaluación

5.1.1. Métricas para la segmentación de instancias

Para la rama de segmentación de instancias utilizamos ocho métricas, de las cuales cuatro de ellas miden el rendimiento computacional y las cuatro restantes el rendimiento en el agrupamiento correcto de las partículas.

Entre las computacionales se encuentran:

- **Train time (Tiempo de entrenamiento)**: Es el tiempo total que toma el proceso de entrenamiento de la red.
- **Test time (Tiempo de prueba)**: Es el tiempo total que toma el proceso de prueba de la red.
- **Memory (Memoria)**: Es el espacio ocupado en memoria durante el entrenamiento.
- **Weight (Peso)**: Es el número de pesos (o neuronas) total de la red.

Y las métricas utilizadas para el agrupamiento son:

- **Accuracy (Exactitud)**: Es la métrica que se utiliza durante el entrenamiento, la cual puede ser definida como la razón para cada instancia entre los puntos correctamente predichos y el total de puntos predichos y verdaderos, siendo n el número verdadero de instancias:

$$accuracy = \frac{1}{n}\sum_{n}\frac{predicted \cap true}{predicted \cup true} \tag{9}$$

- **Purity (Pureza)**: La pureza se define como el porcentaje del número total de puntos que fueron clasificados correctamente. Para calcularlo se utiliza una tabla de contingencia entre los labels predichos y los verdaderos, siendo m la cantidad de instancias predichas, c cada elemento de la matriz de contingencia y p_j la cantidad de puntos predichos en la instancia j:

$$purity = \frac{1}{m}\sum_{j=1}^{m}\frac{max_{i=1}^{m} c_{ij}}{p_j} \tag{10}$$

- **Efficiency (Eficiencia)**: La eficiencia se puede entender como el opuesto de la Pureza, analizando los elementos verdaderos en lugar de los predichos. Por lo que siendo p_i el número de puntos verdadero en la instancia i:

$$efficiency = \frac{1}{n}\sum_{i=1}^{n}\frac{max_{j=1}^{m} c_{ij}}{p_i} \tag{11}$$

- **Adjusted Rand Index (Índice de Rand ajustado)**: El ARI es utilizado en estadística como una medida de la similitud entre dos agrupamientos. Para calcularlo utilizamos la función "adjusted_rand_score" de la librería sklearn.

5.1.2. *Métricas para la segmentación semántica*

Para la rama de segmentación semántica analizaremos cuatro métricas, las cuales son:

- **Accuracy (Exactitud)**: Representa el porcentaje de puntos que se clasificaron correctamente en la clase correspondiente.

$$accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{12}$$

Donde, TP (positivos verdaderos, del inglés true positive) representa los puntos clasificados correctamente de la clase analizada, mientras que, FN (negativos falsos, del inglés false negative) los clasificados incorrectamente, TN (negativos verdaderos, del inglés true negative) representa a los puntos no pertenecientes a la clase analizada clasificados correctamente y FP (positivos falsos, del inglés false positive) representa a aquellos clasificados incorrectamente no pertenecientes a la clase.

- **Precision (Precisión)**: Representa la capacidad del clasificador de no etiquetar como positiva una muestra que es negativa.

$$precision = \frac{TP}{TP + FP} \tag{13}$$

- **Recall (Sensibilidad)**: Representa la capacidad del clasificador para encontrar todas las muestras positivas.

$$recall = \frac{TP}{TP + FN} \tag{14}$$

- **Score F1 (Valor-F)**: Puede ser interpretada como una media armónica entre las métricas Precision y Recall.

$$F1 = 2\frac{precision \times recall}{precision + recall} \tag{15}$$

5.2. *Resultados parciales*

5.2.1. *Experimento 1*

Para elegir el bloque de red a utilizar analizamos cada uno de ellos luego de un entrenamiento de una época usando como dataset el archivo "dlprod_ 512px_00.root" y una prueba también de una época con el dataset del archivo "dlprod_512px_01.root", cada uno de ellos cuenta con 10 mil eventos principales. El mismo fue ejecutado con un tamaño de Batch igual a 10.

Para la reconstrucción de las instancias se utilizaron los valores de s_0 y p_0 descritos en el Cap. 3 y que se muestran en la Tabla 1, los cuales fueron extraídos del trabajo base.

Tabla 1: Valores iniciales de los parámetros s_0 y p_0.

	HIP	**MIP**	**Shower**	**Delta**	**Michel Electron**
s_0	0,65	0,5	0,25	0,85	0,0
p_0	0,29	0,087	0,036	0,27	0,087

Fuente: Elaboración propia.

En este experimento podemos observar diferencias poco significativas en las distintas métricas de agrupación de la red como se muestra en la Figura 10

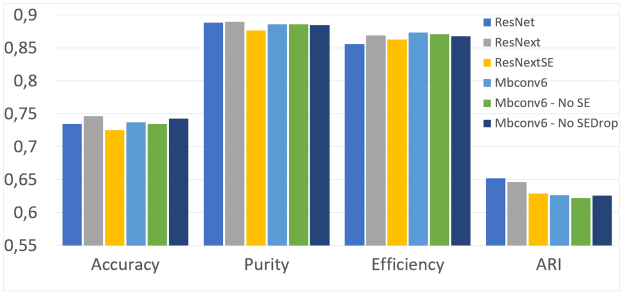

Fig. 10: Gráfico métricas de agrupamiento en el experimento 1.

En la Figura 11 podemos observar una comparación de los tiempos de entrenamiento y validación de los distintos bloques de red durante éste experimento. Se puede verificar que el bloque ResNeXt obtiene las mejores métricas con los menores tiempos de ejecución tanto para el entrenamiento como la validación.

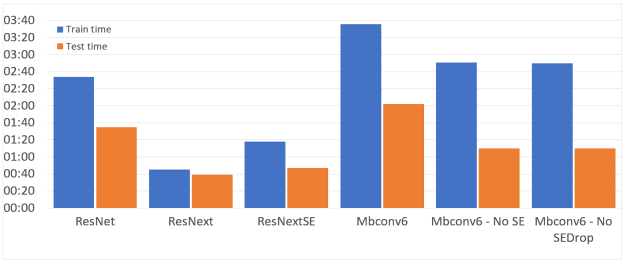

Fig. 11: Comparación de tiempos de entrenamiento y validación en el experimento 1.

5.2.2. *Experimento 2.1: Ajuste manual*

En este experimento utilizamos los mismos datasets utilizados en el experimento anterior. Se realiza el grid search manual entrenando durante 5 épocas por intento, con un Batch size de 50 sobre los siguientes hiperparámetros definidos anteriormente. Donde, f: Número de filtros, k: Tamaño de los filtros en la Input Layer, s: Número de Strides y r: Número de bloques en cada Backbone.

En este experimento obtuvimos como resultado que la combinación de hiperparámetros que obtuvieron un mejor desempeño computacional es la de "f8 k3 s5 r2". Por otro lado, la combinación de hiperparámetros con la cual se consiguió el accuracy más alto en esta prueba es la de "f16 k3 s6 r2".

En la Tabla 2 tenemos una visualización de los resultados obtenidos en este experimento.

Tabla 2: Análisis de los resultados obtenidos en el experimento 2.1.

	Accuracy	**Purity**	**Efficiency**	**ARI**
min	0,767100	0,876200	0,864100	0,604100
max	0,805000	0,909900	0,914400	0,717000
median	0,789800	0,899550	0,890900	0,645000
mean	0,790221	0,897550	0,888893	0,652793
std	0,011047	0,009196	0,014083	0,031467

Fuente: Elaboración propia.

En este experimento podemos observar escasa diferencia en las distintas métricas con una desviación estándar con un valor aproximado del 1.4 % de la media en el accuracy, y valores similares en las demás métricas.

5.2.3. *Experimento 2.2: Ajuste automatizado*

En la primera parte de este experimento se realizó la búsqueda de los hiperparámetros que influyen en el tamaño de la red, con lo que se obtuvo un accuracy de 0.664 realizando 550

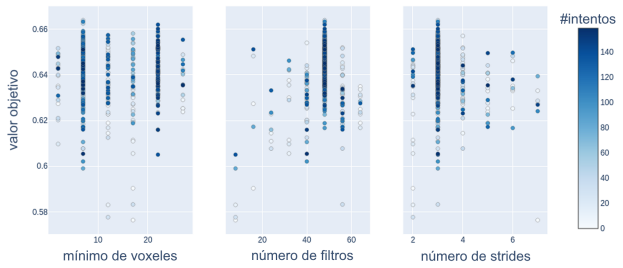

Fig. 12: Resultado del accuracy para cada intento de la primera parte del experimento 2.2.

La segunda parte realizó una calibración de la función de perdida en la rama de segmentación en instancias utilizando los resultados de la primera parte. Consiguió un accuracy de 0,671 luego de 400 intentos:

Para esta parte quedó fijo el peso del L_{emb} (Eq. 6) y se fueron variando el resto los parámetros.

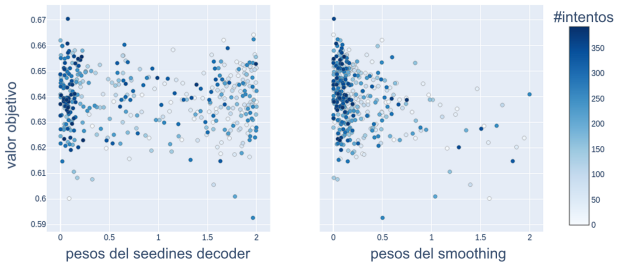

Fig. 13: Resultado del accuracy para cada intento de la segunda parte del experimento 2.2.

La tercera parte buscó configurar el optimizador, eligiendo entre 3 variaciones de Adam y ajustando la tasa de aprendizaje, consiguiendo un accuracy de 0.684 luego de 550 intentos:

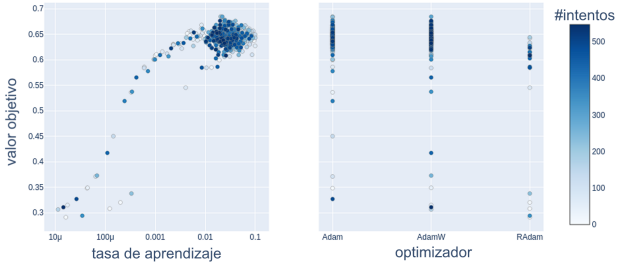

Fig. 14: Resultado del accuracy para cada intento de la tercera parte del experimento 2.2.

5.3. *Entrenamientos finales*

5.3.1. *Experimento 3: Resultado preliminar*

Luego del análisis realizado en cada uno de los experimentos seleccionamos los parámetros a ser utilizados para un entrenamiento preliminar con el dataset completo. Éstos son: Número de filtros = 48, Número de strides = 4, Número de bloques = 1, Número de voxeles = 7, Seediness = 0.0767, Smoothing = 0.0052, Embeddings = 1.0, Optimizador = Adam, Tasa de aprendizaje= 0.02.

Con estos valores se realizó un entrenamiento de 60 épocas, utilizando 80.000 muestras del dataset. De las 60 épocas, se congelaron los pesos de la rama de seediness en las primeras 20 épocas para evitar la divergencia de la rama antes de que se obtengan buenas agrupaciones. El experimento tuvo un tiempo de entrenamiento de 10 días con 12 horas y 32 minutos, con un Batch size de 32 representa un total de 150000 iteraciones, arrojando un promedio de 6,1 segundos de tiempo de entrenamiento por iteración. En la Figura 15 podemos observar las gráficas de accuracy y loss, los puntos amarillos representan el scatter plot de cada iteración, la línea roja representa una curva de regresión logística de orden 5 de los mismos y la línea azul representa el promedio de los valores en la última época.

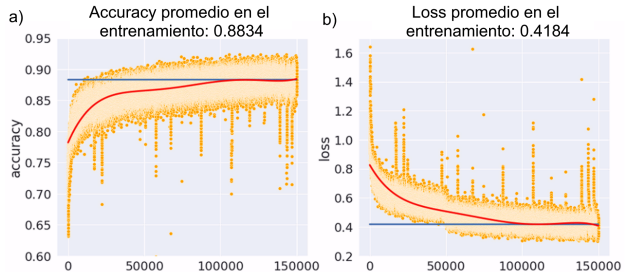

Fig. 15: Gráficas de accuracy y loss en el entrenamiento en el experimento 3.

En la Figura 15 se observan los valores de accuracy y loss a lo largo del entrenamiento. La gráfica dispersa naranja representa los valores para cada una de las 150.000 iteraciones, la roja es una curva de regresión logística y la azul es el valor de la media en la última época.

A lo largo del entrenamiento se corrieron inferencias de un conjunto de validación, el cual está constituido por 25 batches de 32 muestras para un total de 800. Esto representa un 4 % del restante conjunto de prueba que está constituido por 20.000 muestras.

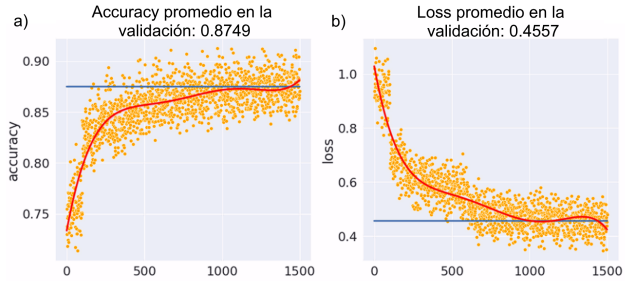

Fig. 16: Gráficas de accuracy y loss en la validación en el experimento 3.

En la Figura 16 se observa el comportamiento del conjunto de validación a lo largo del entrenamiento, en el cual no se observa signos de overfitting al mantener una variación mínima con respecto a los valores presentes en el conjunto de entrenamiento por lo que se tomaran los pesos de la última época para los siguientes análisis.

5.3.2. *Experimento 4: Prueba adicional*

Decidimos realizar una prueba adicional con los parámetros del trabajo base, con los mismos tamaños de dataset, Batch e iteraciones del entrenamiento anterior. Este entrenamiento duró 10 días, 1 hora y 44 minutos y los hiperparámetros para este entrenamiento fueron los siguientes: Número de filtros = 32, Número de strides = 6, Número de bloques = 2, Número de voxeles = 2, Seediness = 1.0, Smoothing = 1.0, Embeddings = 1.0, Optimizador = Adam, Learning Rate = 0.005.

En la Figura 17 se observan las gráficas de accuracy y loss tanto del conjunto de entrenamiento como el de validación, analizando con mayor detenimiento las gráficas de validación, se observa como los valores mínimos y los máximos se van alejando de la media aproximadamente en la época 40.

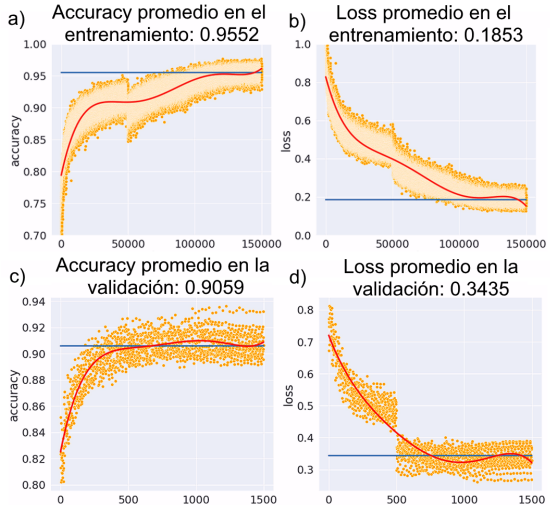

Fig. 17: Gráficas de accuracy y loss en el conjunto de entrenamiento y validación del experimento 4.

Calculando la media del accuracy para cada época (Figura 18) podemos obtener el valor máximo del mismo, el cual se obtiene en la época 39 con un valor de 0.9107. Por lo tanto, elegimos la época 39 junto con la última para los análisis finales, los cuales denominaremos Experimento 4.1 con el que utilizaremos los pesos aprendidos en la época 39 y Experimento 4.2 en el que utilizaremos los pesos de la época 60.

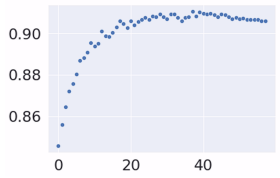

Fig. 18: Media del accuracy para cada época.

5.3.3. *Calibración de los umbrales de decisión para cada experimento*

Para el análisis de los resultados se obtuvieron las métricas con el conjunto de Prueba, el cual cuenta con 19200 muestras y los parámetros $s0$ y $p0$ fueron los al inicio del experimento 1.

TABLA 3: Comparación de métricas entre los experimentos 3, 4.1 y 4.2.

	Accuracy	**Purity**	**Efficiency**	**ARI**
Exp. 3	0,8784	0,9386	0,9314	0,7589
Exp. 4.1	0,9094	0,9328	0,9362	0,7657
Exp. 4.2	0,9133	0,9340	0,9350	0,7628

Fuente: Elaboración propia.

Para los tres modelos observamos un alto rendimiento en Accuracy, Purity y Efficiency, no así con el Adjusted Rand Index. Esto puede deberse a que los parámetros $s0$ y $p0$ fueron optimizados para otro tipo de red. Para obtener un mejor rendimiento en el ARI, realizamos una optimización de los parámetros $s0$ y $p0$ para cada experimento con los que obtuvimos los siguientes resultados:

Tabla 4: Métricas finales en los experimentos 3, 4.1 y 4.2 con los parámetros $s0$ y $p0$ ajustados.

	Accuracy	**Purity**	**Efficiency**	**ARI**
Exp. 3	0,878	0,939	0,916	0,819
Exp. 4.1	0,911	0,962	0,926	0,889
Exp. 4.2	0,913	0,963	0,927	0,881

Fuente: Elaboración propia.

En los 3 experimentos podemos observar un aumento leve tanto en accuracy como en Purity, también un leve descenso en el Efficiency pero un marcado aumento en el ARI de entre 6 y 12 puntos. Con estos resultados podemos afirmar que la optimización de los valores de $s0$ y $p0$ produjo una mejora en el proceso final de reconstrucción de instancias y se selecciona el experimento 4.1 al tener un promedio mayor en las métricas.

5.3.4. Entrenamiento final con rama semántica

Para este último análisis realizamos un último entrenamiento con los valores reales de la rama semántica de la red, para ello primero hacemos un entrenamiento de ésta rama utilizando el bloque ResNeXt con los siguientes parámetros: Número de filtros = 32, Número de strides = 6, Número de bloques = 2, Optimizador = Adam, Learning Rate = 0.005.

Se entrenó la red por un total de 4 días y 3 horas para un total de 40 épocas. Se consiguió durante el entrenamiento una media de 0.9932 de accuracy en el dataset de validación, pasando al dataset de test se obtuvieron los siguientes resultados:

Tabla 5: Métricas obtenidas luego del entrenamiento en la rama semántica.

	Accuracy	**Precision**	**Recall**	**F1**
Media	0,9927	0,9214	0,9104	0,9115
Std	0,0127	0,1088	0,1101	0,1101
10 %	0,9824	0,7496	0,7484	0,7476
50 %	0,9964	0,9816	0,9708	0,9710
90 %	0,9996	0,9988	0,9986	0,9976

Fuente: Elaboración propia.

Como se observa en la Tabla 5, si bien el accuracy da un resultado superior a 0,99, observamos un valor inferior en las demás métricas, el percentil 10 exhibe una gran cantidad de muestras con un rendimiento inferior a 0,75 pero analizando la mediana observamos que la mayoría de las muestras estuvieron por encima de 0,97.

Finalmente, realizando la reconstrucción de instancias incluyendo tanto la rama de instancias como la rama semántica obtenemos las métricas expresadas en la tabla en la cual comparamos con los resultados del Experimento 4.1:

Tabla 6: Métricas obtenidas luego del del entrenamiento final.

	Accuracy	**Purity**	**Efficiency**	**ARI**
Exp. 4.1	0,911	0,962	0,926	0,889
Recons. final	0,910	0,952	0,914	0,885

Fuente: Elaboración propia.

Las cuales difieren en un 1,5 % con los valores obtenidos utilizando los valores reales de la rama semántica en el experimento 4.1.

6. Conclusiones

El objetivo principal de este trabajo fue el de realizar la segmentación de instancias de las partículas capturadas por el tipo de detector LArTPC, utilizando datos generados por simulación de lo que serían las partículas capturadas por éstos detectores en el futuro proyecto DUNE. El resultado es una herramienta capaz de interpretar y analizar estos datos generados por los detectores de partículas, de tal forma a facilitar el estudio de éstas partículas de interés.

El trabajo abarcó un proceso integral, desde el estudio del estado del arte, el análisis de los datos simulados, la creación de la red neuronal, la investigación de las diversas arquitecturas de redes y por último el análisis de los resultados de cada experimento. De esta forma, se pudo dar cumplimiento a todos los objetivos propuestos. Entre esos objetivos, resulta adecuado mencionar:

- Mediante el adecuado estudio de los detectores del tipo LArTPC pudimos entender el proceso de generación de las imágenes de las partículas detectadas. También, nos ayudó a entender como los científicos diferencian éstas clases de partículas entre sí, por lo cual no tuvimos mayores inconvenientes a la hora de poder interpretar los resultados generados por nuestros experimentos.

- Los datos de entrenamiento (o dataset) son claves para el buen desempeño de las redes, generalmente las redes requieren una gran cantidad de datos distintos para que la red pueda ser "aprender" con la profundidad suficiente. En el caso de nuestro trabajo utilizamos datos simulados debido a que el experimento DUNE sigue en construcción.

- Comparando nuestros resultados con los del trabajo base se verifica que nuestra reconstrucción final obtuvo una media en el ARI del 88,5 %, mientras que en el trabajo base se obtuvo un 92 %. En lo que respecta al rendimiento computacional, para el entrenamiento de nuestra red se necesitaron aproximadamente unos 10 días, en cambio el entrenamiento de la red del trabajo base, extrapolando los datos obtenidos, calculamos que nos tomaría aproximadamente unos 33 días. Con lo que podemos determinar que nuestra red tiene una mayor velocidad de entrenamiento.

7. Referencias

Adams, C., Terao, K., y Wongjirad, T. (2020). Pilarnet: Public dataset for particle imaging liquid argon detectors in high energy physics. *arXiv preprint arXiv:2006.01993*.

collaboration, D. *et al.* (2020). Deep underground neutrino experiment (dune): Far detector technical design report. volume i. introduction to dune. *Journal of Instrumentation*, **15**(8).

Graham, B., Engelcke, M., y Van Der Maaten, L. (2018). 3d semantic segmentation with submanifold sparse convolutional networks. En: *Proceedings of the IEEE conference on computer vision and pattern recognition*. pp. 9224–9232.

Hu, J., Shen, L., y Sun, G. (2018). Squeeze-and-excitation networks. En: *Proceedings of the IEEE conference on computer vision and pattern recognition*. pp. 7132–7141.

Koh, D. H., de Soux, P. C., Dominé, L., Drielsma, F., Itay, R., Lin, Q., Terao, K., Tsang, K. V., y Usher, T. L. (2020). Scalable, proposal-free instance segmentation network for 3d pixel clustering and particle trajectory reconstruction in liquid argon time projection chambers.

Rubbia, C. (1977). The liquid-argon time projection chamber: a new concept for neutrino detectors. Reporte técnico.

Zuber, K. (2020). *Neutrino physics*. Taylor & Francis.