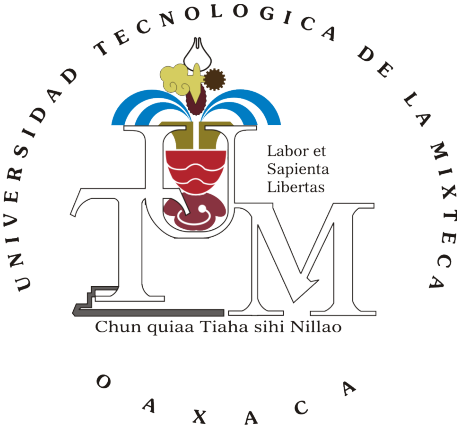

UNIVERSIDAD TECNOLÓGICA DE LA MIXTECA

DIVISIÓN DE ESTUDIOS DE POSGRADO

SEGMENTACIÓN DISTRIBUIDA MEDIANTE REDES NEURONALES CONVOLUCIONALES PARA DETECCIÓN DE NÓDULOS PULMONARES

TESIS

PARA OBTENER EL TÍTULO DE
MAESTRO EN TECNOLOGÍAS DE CÓMPUTO APLICADO

Presenta:
Ing. Vicente Hernández Solis

Director:
Dr. Raúl Cruz Barbosa

Co-director:
Dr. Arturo Téllez Velázquez

Huajuapan de León, Oaxaca, marzo de 2022

Agradecimientos

A mis directores de tesis: Dr. Raúl Cruz Barbosa y Dr. Arturo Téllez Velázquez.

A mis sinodales: Dr. José Aníbal Arias Aguilar, Dr. Eduardo Sánchez Soto, MTCA. Moisés E. Ramírez Guzmán y MC. José Antonio Moreno Espinosa.

A la Universidad Tecnológica de la Mixteca y al Consejo Nacional de Ciencia y Tecnología.

Gracias a todos y cada uno de ustedes por todo el apoyo brindado durante mis estudios de posgrado. Muchas Gracias!

Índice general

Índice de figuras

Índice de cuadros

Resumen

Actualmente, el cáncer de pulmón ha tomado relevancia a nivel mundial, ya que es el tipo de cáncer que provoca un mayor número de muertes. En México, la situación no es diferente, ya que anualmente causa la muerte de aproximadamente ocho mil personas. Para la detección del cáncer pulmonar, se utilizan radiografías y tomografías computarizadas, las cuales son analizadas por un especialista, quien busca nódulos que puedan ser potencialmente cancerígenos. Las tomografías computarizadas son las que mayor confiabilidad ofrecen, por tratarse de una representación tridimensional de los pulmones. Una etapa crucial para la detección mediante métodos computacionales es la segmentación de los objetos contenidos en las imágenes médicas, ya que una segmentación incorrrecta produciría detecciones erróneas de nódulos.
Por lo anterior, en el presente trabajo de tesis se implementan, de manera distribuida y secuencial, dos algoritmos de segmentación de imágenes con redes neuronales convolucionales, el primero basado en la arquitectura AlexNet y el segundo en la arquitectura U-Net, donde se propone una arquitectura adaptada para la segmentación de nódulos. Ambas redes se benefician de las bondades del cómputo distribuido, con la finalidad de ayudar a mejorar la eficiencia de ejecución en tiempo y eficacia en la segmentación, y consecuentemente, ayudar en la detección de nódulos pulmonares. Los resultados experimentales obtenidos muestran que ambos algoritmos ofrecen buen rendimiento de segmentación, lo cual es medido en términos del Coeficiente de Similaridad Sørensen–Dice, ya que la AlexNet logra un procentaje del 74 % mientras que la U-Net modificada alcanza el 88 %. Respecto a los tiempos de procesamiento, se logró acelerar la AlexNet en 16.8 veces durante el entrenamiento y 9.3 veces durante la etapa de predicción respecto a su versión secuencial. Asimismo, se logró acelerar la U-Net modificada en 18.1 veces en la etapa de entrenamiento y 15.8 veces durante la etapa de predicción. Dichas aceleraciones son evidencia del mejoramiento del tiempo de ejecución de la implementación distribuida de las redes neuronales convolucionales para la segmentación de nódulos pulmonares comparado con su versión secuencial.

Publicación derivada

La publicación generada a partir del presente trabajo de tesis es:

- Hernández-Solis V., Téllez-Velázquez A., Orantes-Molina A., Cruz-Barbosa R. (2021) *Lung-Nodule Segmentation Using a Convolutional Neural Network with the U-Net Architecture*. In: Pattern Recognition. MCPR 2021. Lecture Notes in Computer Science, vol 12725. Springer, Cham. `https://doi.org/10.1007/978-3-030-77004-4_32`

En la publicación se presenta la modificación de una red neuronal convolucional adaptada a la segmentación de nódulos pulmonares. Se detalla la implementación y la metodología para obtener los hiper-parámetros de la red que maximizan las medidas de rendimiento. Adicionalmente, se presenta un análisis de los resultados obtenidos y una comparación con otros métodos del estado del arte.

Capítulo 1

Introducción

Hoy en día, existe una fuerte necesidad clínica para diagnosticar a los pacientes y recopilar datos que a largo plazo permitan identificar diversas enfermedades, y en consecuencia, iniciar o retomar algún tratamiento médico [Fourcade and Khonsari, 2019]. Debido al auge del *Big Data* y de los grandes avances en la tecnología actual, es posible expandir la capacidad para generar nuevo conocimiento que sea útil en distintas áreas de la ciencia y particularmente en la medicina y el cuidado de la salud [Sze et al., 2017].

De todos los tipos de cáncer que existen, el cáncer de pulmón es el que más muertes causa anualmente a nivel mundial [WHO, 2022]. De estas, ocho mil muertes ocurren en México [SALUD/SSA, 2018]. Para la detección de este tipo de cáncer, se realiza una búsqueda visual de nódulos pulmonares basada en estudios médicos avanzados, tales como radiografías (CR, por sus siglas en inglés) y tomografías computarizadas (CT, por sus siglas en inglés). La búsqueda suele ser una tarea compleja que debe ser realizada por un especialista. Ante la creciente demanda de diagnósticos y los pocos especialistas disponibles, se requiere del uso de herramientas que apoyen en la detección y agilizen el tiempo requerido para dicho proceso.

La segmentación de imágenes es una tarea central en muchas aplicaciones de procesamiento de imágenes. Por ejemplo, existen estudios que han tratado la segmentación de imágenes mediante lógica difusa, algoritmos de contorno activo y agrupamiento [Nithila and Kumar, 2016]. Sin embargo, una nueva alternativa ha surgido con el desarrollo del aprendizaje profundo, ya que muchos métodos de redes neuronales convolucionales (CNN, por sus siglas en inglés) han mostrado resultados sorprendentes en la segmentación de imágenes en tomografías y radio-

grafías computarizadas, superando a las técnicas tradicionales o convencionales [Ni et al., 2020].

El presente trabajo de tesis aborda la segmentación de nódulos pulmonares, mediante redes neuronales convolucionales, procesando dichos algoritmos en un tiempo considerablemente inferior, debido al uso del cómputo distribuido. La idea básica es que las redes convolucionales permiten una mayor eficacia en la segmentación de los nódulos, mientras que el ćomputo distribuido permite una eficiencia mayor en tiempo a la que se conseguiría con cómputo secuencial.

Para lograr este propósito, se utilizan tomografías computarizadas de la base de datos LIDC-IDRI [CIA, 2022] y se procesan en un clúster computacional, ya que éste tiene la capacidad para realizar tareas simultáneamente, dividiendo el problema en subproblemas más pequeños y tratando cada uno de ellos independientemente.

Por lo anterior, el presente proyecto de tesis propone la modificación de una red neuronal convolucional U-Net para segmentar nódulos pulmonares, a partir de la región de interés proporcionada por expertos. El desempeño de esta propuesta se mide mediante el uso de dos medidas de rendimiento: primero el Coeficiente de Similaridad Sørensen–Dice (DSC, por sus siglas en inglés) permite validar la calidad de la segmentación y segundo, la aceleración distribuida permite validar cómo la plataforma de Apache Spark ayuda a mejorar los tiempos de procesamiento asociados al entrenamiento de la red neuronal propuesta, mediante el algoritmo de retropropagación distribuida. Adicionalmente, y con fines de comparación, se implementa un segmentador basado en la red neuronal AlexNet usando también el algoritmo de retropropagación distribuido.

1.1. Planteamiento del problema

Actualmente, el cáncer es la segunda causa de mortalidad en seres humanos. A nivel mundial se le atribuyen 10 millones de defunciones solo en 2020 (aproximadamente un 16 % del total). De éstas, 1.8 millones de muertes son causadas por el cáncer de pulmón, que es el tipo de cáncer más frecuente, siguiéndole el cáncer de colon y el de mama [WHO, 2022]. En México, anualmente el cáncer de pulmón causa la muerte de aproximadamente 8 mil personas, siendo la primera causa de muerte por cáncer [SALUD/SSA, 2018]. A pesar de esto, se sabe que puede ser prevenido e incluso curado con una detección temprana y el tratamiento adecuado.

Una de las pruebas médicas para diagnosticar cáncer de pulmón en un paciente es la tomografía simple de tórax, en la cual se detecta el tamaño y la cantidad de nódulos pulmonares, tanto benignos como malignos. Esta prueba consiste en utilizar un conjunto de imágenes digitales que son tomadas transversalmente mediante rebanadas, realizadas cada cierta distancia (normalmente 3 mm), para obtener una representación tridimensional de los pulmones del paciente. A partir de dicha representación, un especialista puede realizar una inspección visual para detectar las posibles lesiones [Wu and Qian, 2019].

Ante la creciente demanda de especialistas que realicen la detección de nódulos pulmonares, se han diseñado herramientas que apoyen dicha detección, mediante técnicas de clasificación y segmentación. Algunas propuestas utilizan algoritmos de segmentación basados en contorno activo y en lógica difusa, para la detección automatizada de nódulos pulmonares. Por ejemplo, Nithila y Kumar desarrollaron una técnica para segmentación de nódulos basada en un algoritmo fuzzy c-means y un modelo de contorno activo. La reconstrucción del parénquima pulmonar se hizo mediante un filtro gaussiano, finalizando la segmentación con un algoritmo de agrupamiento [Nithila and Kumar, 2016].

Recientemente, las técnicas de aprendizaje profundo han impactado positivamente en varios problemas de clasificación y segmentación de imágenes. En la detección de nódulos pulmonares, también han habido algunos trabajos en los cuales consideran una segmentación voxel a voxel y las redes neuronales convolucionales de vista múltiple (MV-CNN por sus siglas en inglés), con las que han determinado si un voxel pertenece o no a un nódulo [Litjens et al., 2017]. Estos avances en el desarrollo de las MV-CNN han logrado mejorar la precisión en las tareas de clasificación y segmentación de imágenes. Sin embargo, estas redes neuronales son grandes y complejas y requieren de recursos de cómputo sustanciales para su entrenamiento y evaluación.

Una de las formas de aumentar la capacidad de cómputo de un sistema es mediante la computación distribuida. Este modelo permite resolver problemas de computación masiva utilizando varias computadoras autónomas, interconectadas a través de una red. Estos sistemas trabajan como un solo recurso computacional para resolver un problema que ha sido dividido en subproblemas más pequeños que el problema original. Adicionalmente, constituyen una solución flexible, de bajo costo y gran escalabilidad para aplicaciones que requieren una elevada capacidad de cómputo y memoria [Milone et al., 2001].

Por lo anterior, en este trabajo de tesis se utiliza un sistema distribuido basado en un clúster computacional que permite procesar, mediante una red neuronal convolucional U-Net modificada, un número grande de imágenes de tomografías

en las cuales se puedan segmentar automáticamente nódulos pulmonares. En otras palabras, se sientan las bases de un sistema asistido por computadora que podría ayudar a los especialistas a agilizar su labor de detección de cáncer de pulmón.

1.2. Justificación

Debido a la creciente demanda de diagnósticos sobre el cáncer de pulmón, es importante atender la detección temprana de nódulos pulmonares utilizando tomografías computarizadas. Sin embargo, esto no ha sido tratado adecuadamente en México. Esto se debe principalmente a la falta de especialistas que puedan dar un diagnóstico acertado, y los pocos que existen demoran en promedio tres minutos en detectar un nódulo de entre 3 y 5 mm, ya que la tarea de detección de nódulos realizada visualmente es compleja [Rubin et al., 2015]. Por otro lado, la escasez de herramientas que apoyen el diagnóstico disminuyen las posibilidades de un resultado oportuno.

Las herramientas de segmentación existentes que permiten aislar los nódulos pulmonares tienen una eficacia alta. Sin embargo, se han buscado nuevas técnicas de segmentación que permitan obtener una mayor eficacia. Las redes neuronales convolucionales han demostrado tener una eficacia mayor que otros métodos que no usan aprendizaje computacional. Sin embargo, éstas demandan recursos computacionales sustanciales, por lo que el uso de estas técnicas se ha restringido a equipos costosos.

Por otro lado, los beneficios que se obtienen al utilizar el cómputo distribuido impactan directamente en la cantidad de recursos disponibles y la velocidad de cómputo que se exhibe al ejecutar programas en el sistema. Desde esta perspectiva, las redes neuronales convolucionales son factibles, además de que permiten el procesamiento de grandes cantidades de datos para la detección de nódulos pulmonares en distintas tomografías simultáneamente.

Además, la Universidad Tecnológica de la Mixteca cuenta con la infraestructura suficiente para la implementación de este tipo de algoritmos distribuidos sobre un clúster computacional. De esta forma, se continua con la línea de investigación del cómputo distribuido que se ha establecido en la Universidad.

1.3. Hipótesis

Con el uso de cómputo distribuido, es posible mejorar el rendimiento en la ejecución de redes neuronales convolucionales para la segmentación de nódulos pulmonares en tomografías computarizadas, facilitando la detección temprana del cáncer de pulmón.

1.4. Objetivos

1.4.1. Objetivo general

Implementar de manera distribuida un algoritmo de segmentación de imágenes mediante redes neuronales convolucionales, con la finalidad de detectar nódulos pulmonares, reduciendo el tiempo de procesamiento de grandes cantidades de imágenes de tomografías.

1.4.2. Objetivos particulares

1. Revisar el estado del arte sobre algoritmos de segmentación de imágenes, tanto secuenciales como distribuidos, basados en redes neuronales convolucionales para detección de nódulos pulmonares.

2. Evaluar las herramientas que facilitan el desarrollo de aplicaciones a través de procesamiento distribuido.

3. Seleccionar dos algoritmos para la segmentación de nódulos pulmonares e implementarlos secuencial y distribuidamente.

4. Seleccionar una base de datos lo suficientemente grande que justifique el procesamiento masivo de una red neuronal convolucional en un sistema distribuido.

5. Comparar el desempeño entre las implementaciones secuenciales y distribuidas de los algoritmos seleccionados.

1.5. Metas

1. Elaboración de un reporte sobre algoritmos de segmentación de nódulos pulmonares en imágenes de tomografías basados en redes neuronales convolucionales, así como de las herramientas que permiten realizar cómputo distribuido.

2. Implementación tanto distribuida como secuencial de dos algoritmos que permitan identificar nódulos pulmonares.

3. Creación de una biblioteca en Python para la ejecución de los algoritmos seleccionados.

4. Elaboración de un cuadro comparativo del rendimiento de las implementaciones secuencial y distribuida de los algoritmos seleccionados.

5. Publicación de un artículo.

1.6. Trabajo relacionado

En medicina, la segmentación de imágenes es usada, por ejemplo, para la detección de órganos, tales como el cerebro, el corazón, los pulmones o el hígado, usando para ello tomografías computarizadas o resonancias magnéticas (MR, por sus siglas en inglés); esto, con la finalidad de distinguir tejido patológico, es decir, diferenciar entre un tumor y el tejido normal [Dougherty, 2009].

El análisis de las tomografías computarizadas, para detección de nódulos pulmonares, inicia con el proceso de segmentación, lo cual es de vital importancia, ya que una segmentación incorrecta daría como resultado diagnósticos clínicos erróneos que afectan directamente a los pacientes. En [Armato and Sensakovic, 2004], se estima que se puede perder entre el 5 % y 17 % de nódulos pulmonares de los datos de prueba, debido a las deficiencias que pudiera tener el algoritmo de segmentación.

Una de las formas más comunes de realizar la segmentación de nódulos, tanto con métodos convencionales como con aprendizaje computacional, es mediante el enfoque basado en regiones, como el presentado en [Savic et al., 2021] que utiliza el método de marcha rápida para dividir la imagen en regiones con características

similares. Para segmentar los nódulos pulmonares, se utilizan múltiples puntos semilla para producir un conjunto de regiones apropiado en la imagen de entrada. Para finalizar, las regiones pertenecientes a cada semilla se fusionan aplicando el algoritmo de K-Means.

Un caso particular se muestra en [Nithila and Kumar, 2016], donde se realizó la segmentación de nódulos pulmonares mediante un modelo de contorno activo, basado en la técnica de crecimiento de regiones y la técnica de Fuzzy C-Means o agrupamiento difuso. La metodología utilizada se centra en tres etapas. La primera de ellas se encarga de la obtención de las tomografías computarizadas; la segunda en la reconstrucción del parénquima pulmonar y la última en la segmentación de nódulos. La reconstrucción del parénquima se realizó mediante un filtro gaussiano, con una nueva función de presión con signo y utilizando agrupamiento para la detección y segmentación de los nódulos.

Por otro lado, en [Lu et al., 2015] se propone un sistema híbrido que hace uso de las siguientes técnicas: operación morfológica, mejora de puntos basada en la matriz de Hesse, segmentación de conectividad difusa, el algoritmo de densidad local máxima, el mapa de distancia geodésica y clasificación mediante árboles de regresión. Primero, se generan todos los nódulos candidatos para clasificarlos en tres tipos: periféricos, de pared torácica y de mediastino. Posteriormente, mediante una estructura de árbol, se detectan los nódulos para finalmente clasificarlos según su localización, tamaño y forma.

En contraste, [Liu and Raj, 2013] hace uso de campos aleatorios discriminativos (DRF, por sus siglas en inglés) para segmentar volúmenes tridimensionales de nódulos en tomografías computarizadas, usando para ello un solo punto semilla por nódulo. Primero, calcularon varias características, como el radio estimado, mediante un filtro morfológico. Posteriormente, incorporaron un filtro morfológico más simple y aproximado en un conjunto de características y restricciones por pares, con la finalidad de lograr una segmentación general más precisa.

Otra técnica menos convencional es la mostrada en [Suji et al., 2020], donde han utilizado el método de Flujo Óptico para procesar los cortes de las tomografías computarizadas. Para ello, crearon una secuencia de fotogramas, a partir de los cortes de la tomografía computarizada, en la cual se encuentra un nódulo. Estos fueron ordenados en una línea de tiempo con la finalidad de obtener la segmentación del nódulo pulmonar mediante el cálculo de los cambios detectados entre los fotogramas.

La segmentación de nódulos pulmonares se ha tratado ampliamente y con distintas técnicas. Por ejemplo, en [Chunran et al., 2018] se propuso un método

basado en una red neuronal convolucional completa (FCN, por sus siglas en inglés). Primero, se realizó la segmentación de los pulmones mediante la FCN, después se umbralizó para detectar los nódulos dentro del área pulmonar, para finalmente segmentar los nódulos mediante el método de ajuste de nivel y el método de umbral, basado en la transformación del sistema de coordenadas.

Otra propuesta de CNN se presenta en [Cao et al., 2020], la cual utiliza una red residual de doble rama (DB-ResNet, por sus siglas en inglés), que captura simultáneamente tanto las características de intensidad provenientes de múltiples vistas como las escalas de diferentes nódulos en las imágenes de CT y, en consecuencia, realiza la segmentación de los nódulos. Adicionalmente, proponen una capa de submuestreo de intensidad central (CIP, por sus siglas en inglés), para extraer las características de intensidad del vóxel central del bloque, y posteriormente obtener el vector de características del vóxel central para realizar la clasificación.

Adicionalmente, estudios realizados por [Wang et al., 2017a] presentan una red neuronal convolucional multi-vista para la segmentación de nódulos pulmonares. Esta red neuronal está especializada en detectar características de posibles nódulos pulmonares en las vistas axial, coronal y sagital en CT. El método propuesto incorpora tres ramas que conjuntamente, son capaces de procesar cada vóxel de la CT, es decir, una vista por cada rama.

Por otro lado, en [Drokin and Ericheva, 2021], se presenta una segmentación de nódulos, de principio a fin, basada en la proyección de características. Como entrada al modelo se usan, no solo la vista axial, sino también la sagital y la coronal, con la finalidad de hacer robusto al sistema sin la necesidad de incrementar el conjunto de datos. El modelo consiste de dos módulos. El primero de ellos utiliza la proyección de intensidad máxima, el cual funciona como un extractor de características preliminares, las cuales son pasadas al segundo módulo, una red neuronal de segmentación similar a la U-Net.

Los trabajos mencionados anteriormente ocupan, en su mayoría, cómputo secuencial o paralelo para realizar la segmentación de nódulos pulmonares en tomografías computarizadas. El hardware más utilizado es una computadora personal de última generación, a la cual algunos autores le agregan una Unidad de Procesamiento de Gráficos (GPU, por sus siglas en inglés) [Ni et al., 2020]. De acuerdo con [Litjens et al., 2017], la mayoría de las contribuciones en el área de análisis y segmentación de imágenes médicas, mediante aprendizaje profundo, se beneficia del procesamiento en GPU. Esto debido principalmente a la gran complejidad del algoritmo y su tiempo de procesamiento asociado. Sin embargo, este tipo de hardware resulta económicamente costoso.

Por ejemplo, en [Keuper, 2016] se realiza la tarea de clasificación con la red neuronal AlexNet, utilizando una computadora personal con varias GPUs para implementar el algoritmo de retropropagación de forma distribuida (con una arquitectura centralizada). Lo anterior, mejora el rendimiento de ejecución al calcular los gradientes de los pesos sinápticos más rápido.

Otra alternativa para procesar grandes cantidades de datos, relacionados con tomografías computarizadas, es el uso del cómputo distribuido, ya que permite crear clústeres de un variado tipo de computadoras en una red y asignar tareas transparentemente a los nodos que lo componen. Los trabajos de [Liu et al., 2016, Liu et al., 2017] han utilizado el paradigma de MapReduce (Apache Hadoop y Apache Spark) para entrenar de forma distribuida un perceptrón multicapa, usando un aprendizaje de conjunto.

En [Liu et al., 2020] se analiza y propone una arquitectura descentralizada para el entrenamiento de redes neuronales multicapa de forma distribuida, con la cual se obtiene un rendimiento similar al obtenido con una arquitectua centralizada. Sin embargo, pocos trabajos existen que aborden la segmentación distribuida de imágenes, explícitamente usando redes neuronales convolucionales.

Otro ejemplo similar es el de [Wang et al., 2016], donde utilizan procesamiento distribuido con Apache Hadoop (predecesor de Apache Spark), para realizar reconocimiento de acciones en vídeo, utilizando redes neuronales convolucionales para extraer la información sobre el movimiento a través del tiempo. Por otra parte, [Akhtar et al., 2018] proponen un algoritmo eficiente para segmentar un gran conjunto de imágenes, realizando operaciones simples como filtros y umbralizaciones, entre otras operaciones, mediante Hadoop y OpenCV.

1.7. Limitaciones

El presente trabajo de tesis contempla un estudio comparativo entre algoritmos de segmentación de tomografías para la detección de nódulos pulmonares, desde el punto de vista del cómputo distribuido y utilizando redes neuronales convolucionales.

Las tomografías computarizadas que se utilizan para este trabajo se obtienen de la base de datos pública LIDC-IDRI [CIA, 2022], por lo tanto los resultados que se obtienen se validan de acuerdo con dicha base.

Por lo anterior, se espera que la implementación distribuida obtenga un tiempo de ejecución inferior al procesar las tomografías, en comparación con el tiempo de ejecución usando cómputo secuencial. Por consecuencia, la disminución de tiempo de cómputo permitirá procesar un gran número de imágenes.

1.8. Metodología

Para realizar la segmentación de nódulos pulmonares en tomografías computarizadas se sigue la metodología mostrada en la figura 1.1, la cual se compone de tres etapas que se describen a continuación:

La primera etapa tiene como objetivo entrenar a la red neuronal convolucional, a partir de las tomografías computarizadas para obtener un modelo $M_i(x, \theta)$ de la misma. Los pasos se detallan a continuación:

1. **Adquisición de datos**. Las tomografías computarizadas, en las cuales se detectan nódulos pulmonares, se toman de la base de datos LIDC-IDRI. Para iniciar la etapa de entrenamiento, se requiere del 80 % de muestras de la base de datos.

2. **Distribución de datos**. El programa principal, encargado de realizar el cómputo paralelo y distribuido, reparte las imágenes entre cada uno de los procesos de Spark que residen en cada nodo trabajador.

3. **Procesamiento de datos**. Cada proceso de Spark realiza las tareas de preprocesamiento de las tomografías computarizadas, algunas de las cuales son el filtrado y normalización de las imágenes para tener una consistencia adecuada. Las imágenes preprocesadas son la entrada para que la red neuronal convolucional pueda realizar su entrenamiento.

4. **Transformación de datos**. Mediante una operación de reducción, que consiste en simplificar y unificar la información obtenida en el paso previo, se obtiene un modelo i-ésimo de la red neuronal convolucional y se guarda en el sistema local de archivos.

La segunda etapa consiste en realizar una prueba para analizar y evaluar el rendimiento de la red neuronal convolucional entrenada, para ello se realizan las siguientes actividades:

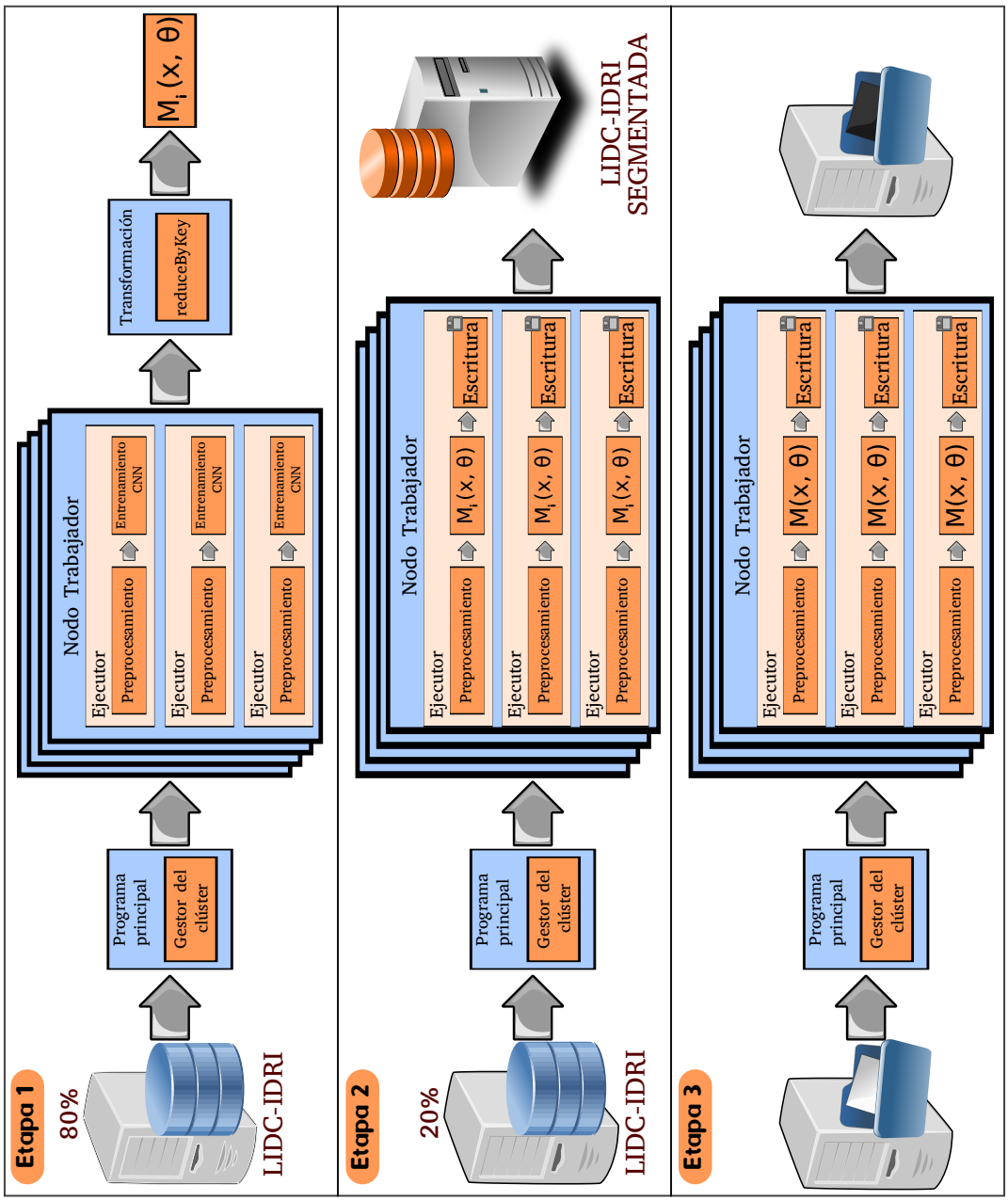

Figura 1.1: Metodología para segmentación distribuida de tomografías computarizadas. 1) Etapa de entrenamiento de la red neuronal convolucional. 2) Etapa de prueba para validar la eficiencia de la red neuronal convolucional. 3) Segmentación y detección de nódulos pulmonares.

1. **Obtención de muestras**. Para realizar la etapa de prueba del modelo de la red neuronal convolucional, obtenido en la etapa anterior, se ocupa el 20 % restante de muestras de la base de datos.

2. **Distribución de datos**. El programa principal se encarga de distribuir las imágenes en cada proceso del nodo trabajador para realizar las tareas simultáneamente.

3. **Prueba del modelo**. En cada nodo trabajador, se realiza el pre-procesamiento del conjunto de imágenes de prueba para ser enviadas al modelo y se obtenga una imagen segmentada que pueda ser guardada en el sistema local de archivos.

Las etapas uno y dos se realizan iterativamente N veces para obtener N modelos $M_1, M_2, ..., M_N$. De éstos, se selecciona, mediante algunas medidas de rendimiento, al mejor (óptimo) para ser utilizado en la etapa siguiente.

En la etapa tres, se hace uso del mejor modelo obtenido $M(x, \theta)$, de la red neuronal convolucional para segmentar nódulos pulmonares en tomografías computarizadas. Los pasos a seguir son:

1. **Recopilación de tomografías**. Para realizar la segmentación de nódulos pulmonares se utilizan los estudios realizados a personas susceptibles de padecer cáncer de pulmón.

2. **Distribución de datos**. El programa principal se encarga de distribuir las imágenes en cada proceso del nodo trabajador para realizar las tareas simultáneamente.

3. **Imágenes segmentadas**. El resultado son tomografías computarizadas segmentadas en las cuales existe la posibilidad de detección de nódulos pulmonares.

Capítulo 2

Marco Teórico

En este capítulo se proporciona una descripción breve de los principales temas que sustentan el presente proyecto de tesis. En la primera sección se hace una introducción al cáncer de pulmón, mencionando los tipos que existen y la forma en la cual un especialista realiza la detección de esta enfermedad. En la segunda sección se describe la segmentación de imágenes digitales y las técnicas más utilizadas para realizar dicho procesamiento. Algunas de estas técnicas hacen uso de la inteligencia artificial, por lo que el aprendizaje computacional, específicamente el aprendizaje profundo, también se presenta en la tercera sección. Dentro de este último, se introducen las redes neuronales convolucionales. Por último, en la cuarta sección se presentan algunos conceptos de cómputo distribuido y sus características, así como la plataforma Apache Spark que se usa para este propósito.

2.1. Detección del cáncer pulmonar mediante tomografías computarizadas

En esta sección se presenta una breve introducción sobre el cáncer de pulmón y las maneras comunes de hacer una detección temprana. Adicionalmente, se describen las tomografías computarizadas y la forma en que se usan para detectar este tipo de cáncer.

2.1.1. Cáncer pulmonar

Un tumor, en general, es un conjunto de células dentro del cuerpo humano, las cuales crecen de forma anormal y descontrolada. Los tumores que provocan el cáncer son llamados malignos, mientras los tumores que no lo hacen, son llamados benignos. Los tumores malignos suelen desplazarse a otras partes del cuerpo, logrando que éstas sean infectadas y que se produzcan nuevos tumores malignos.

El cáncer puede desarrollarse en cualquier parte del cuerpo humano. Cuando el crecimiento anormal empieza en los pulmones se le conoce como *cáncer de pulmón* [DeVita et al., 2019]. Dentro del cáncer pulmonar, existen principalmente dos tipos: el cáncer de células no pequeñas (NSCLC, por sus siglas en inglés) y el cáncer de células pequeñas (SCLC, por sus siglas en inglés). Los tumores del primer tipo son responsables del 80 % al 85 % de los casos de cáncer de pulmón, mientras que del 10 % al 15 % de los casos son provocados por los del segundo tipo [Niederhuber et al., 2020].

De acuerdo con [Gould et al., 2007, Travis, 2011] los principales subtipos de cáncer de células no pequeñas son los siguientes:

- Adenocarcinoma. Representa aproximadamente el 50 % de los nódulos pulmonares malignos. Éste suele encontrarse en la periferia del pulmón y es posible detectarlo antes de que empiece a extenderse.

- Carcinoma de células escamosas. Al igual que el adenocarcinoma, suele presentarse en fumadores. Sin embargo, los nódulos se encuentran en la parte central de los pulmones, es decir, cerca de los bronquios.

- Carcinoma de células grandes. Este tipo puede aparecer en cualquier parte de los pulmones y tiende a crecer y expandirse rápidamente, lo cual dificulta su tratamiento.

El cáncer de pulmón de células pequeñas tiende a crecer y expandirse más rápidamente que el de células no pequeñas. Aproximadamente, el 70 % de las personas tendrán extendido este cáncer en el momento que son diagnosticadas. Este cáncer, a pesar de ser tratado con quimioterapia o radioterapia, en la mayoría de las personas reaparecerá en algún momento.

2.1.2. Uso de tomografías computarizadas

El diagnóstico del cáncer de pulmón puede realizarse a partir de muestras del tejido, como las biopsias endobronquiales y transbronquiales. Otras alternativas son mediante la evaluación de muestras citológicas procedentes de procedimientos, tales como la aspiración con aguja fina (FNA, por sus siglas en inglés) transbronquial/transtorácica, los lavados bronquiales o mediante la evaluación del esputo o líquido pleural.

La eficacia del diagnóstico depende de varios factores, como la localización (accesibilidad) del tumor, el tamaño, el tipo y el nivel de experiencia del broncoscopista o patólogo. En general, las lesiones centrales, como los carcinomas de células escamosas y las SCLC, se diagnostican más fácilmente con un examen broncoscópico; mientras que las lesiones periféricas, como las adenocarcinomas y los carcinomas de células grandes, son más fáciles de diagnosticar con una FNA transtorácica o una biopsia [Niederhuber et al., 2020]. En la mayoría de los casos, los síntomas de cáncer no aparecen sino hasta que la enfermedad ya se encuentra en una etapa avanzada.

Para una detección temprana del cáncer de pulmón, se realizan pruebas o exámenes en personas que no presentan síntomas de la enfermedad. Algunas de estas pruebas son la radiografía regular de tórax y las tomografías de tórax de baja dosis (LDCT, por sus siglas en inglés o simplemente CT), en las cuales se realiza una inspección visual para detectar los posibles tumores. Sin embargo, las primeras no han ayudado a prevenir el cáncer de pulmón, por lo que en los últimos años, las CT han sido utilizadas en personas con un mayor riesgo de padecer cáncer de pulmón, obteniendo mejores resultados que con las radiografías, ya que éstas pueden ayudar a encontrar un mayor número de zonas anormales tales como nódulos, tejido cicatricial o cambios postoperatorios que puedan derivar en cáncer de pulmón [Niederhuber et al., 2020].

Un nódulo pulmonar es una pequeña área anormal de tejido que se desarrolla en los pulmones y que puede ser descubierta mediante la inspección visual de CTs de tórax, ya que estas proporcionan una evaluación *in vivo*, tanto de las características biológicamente relevantes de los nódulos como de la extensión de la lesión. El uso de las CT permite una evaluación rápida y completa de los pulmones, además de que tienen un aumento de la sensibilidad y una baja exposición a la radiación [medical, 2019].

La principal característica para determinar el riesgo de malignidad de un nódulo es su tamaño. Sin embargo, otras características como el borde, la forma

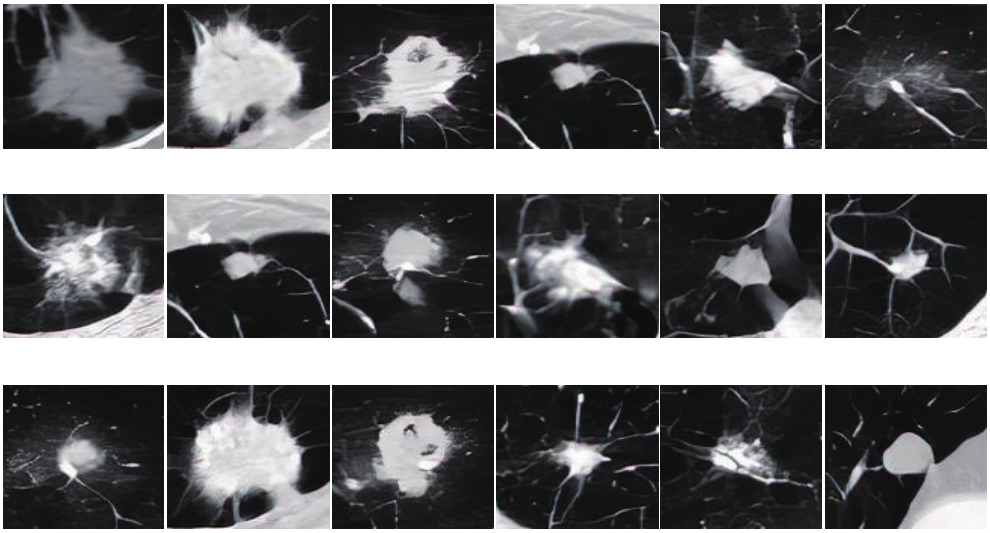

Figura 2.1: Diferentes formas de nódulos pulmonares: redondeados, espiculados, ovalados, etc. Fuente: [Gu et al., 2020]

(veáse Figura 2.1) y su localización, son también relevantes para clasificar un nódulo como maligno o benigno [Momen et al., 2007]. La mayoría de los nódulos benignos tienen un borde bien definido; sin embargo, en [Siegelman et al., 1980] se estima que el 21 % de nódulos con bordes bien definidos podrían ser nódulos malignos. Los nódulos espiculados, lo mismo que los nódulos con un borde lobular, suelen ser malignos [Young et al., 2007]. Sin embargo, hasta el 25 % de los nódulos benignos también pueden presentar bordes lobulares [Gurney et al., 1993].

La forma del nódulo también ofrece información sobre la malignidad del mismo. Las formas irregulares tienen una probabilidad más alta de ser malignos, a diferencia de los nódulos redondeados o con forma poligonal [Xu et al., 2008].

2.2. Métodos de segmentación de imágenes digitales

La segmentación es una etapa crucial para iniciar el proceso de análisis de una imagen digital, ya que a partir de ésta se obtiene la región de interés. Sin embargo, para garantizar la adecuada segmentación de alguna aplicación específica, es necesario una etapa de preprocesamiento. La presente sección describe brevemente tanto a los métodos de segmentación clásicos como a los modernos aplicados a problemas similares a los expuestos en este trabajo de tesis.

2.2.1. Métodos clásicos y modernos

Como se ha mencionado anteriormente, la segmentación es un proceso crucial en el análisis de imágenes digitales, en el cual se subdivide una imagen en las regiones u objetos que la constituyen. La granularidad a la que se requiera dicha subdivisión depende del problema que se desee resolver, es decir, la segmentación debería terminar cuando se hayan detectado los objetos o regiones de interés para una aplicación. En general, los algoritmos clásicos de segmentación se basan en una de las dos propiedades básicas de los valores de intensidad: *discontinuidad* y *similitud* [Gonzalez and Woods, 2008].

En la categoría de discontinuidad, las técnicas consisten en dividir una imagen en función de los cambios bruscos de intensidad, por ejemplo, usando los bordes. Por otro lado, las técnicas principales, en la categoría de similitud, se basan en la división de una imagen en regiones que son similares, de acuerdo con un conjunto de criterios claramente definidos.

Para segmentar objetos de una imagen, los algoritmos basados en discontinuidad se apoyan en los cambios bruscos de niveles de grises que existen entre píxeles adyacentes, permitiendo detectar puntos aislados, líneas y/o bordes. Para la detección de puntos aislados se utiliza el Laplaciano, el cual es mostrado a continuación:

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \tag{2.1}$$

Entonces la segmentación de define como:

$$g(x,y) = \begin{cases} 1 & \text{si } |R(x,y)| \geq \tau \\ 0 & \text{en caso contrario} \end{cases} \tag{2.2}$$

donde g es la imagen de salida, τ es el umbral no negativo y R es el resultado de aplicar la máscara (Laplaciano) de tamaño $n \times n$.

La detección de bordes es el enfoque más utilizado para segmentar las imágenes en función de los cambios abruptos de intensidad, es decir, mediante el cálculo del gradiente, típicamente usando una máscara de Sobel o de Prewitt. Los modelos de bordes se clasifican según sus perfiles de intensidad. Un borde implica una transición entre dos niveles de intensidad que se produce idealmente a lo largo de la distancia de 1 píxel. La magnitud de la primera derivada puede ser utilizada

para detectar la presencia de un borde en un punto de una imagen. De manera similar, el signo de la segunda derivada puede usarse para determinar si un píxel de un borde se encuentra en el lado oscuro o claro de un borde. Otra técnica sofisticada, como el detector de bordes Marr-Hildreth, utiliza operadores grandes y pequeños para detección de bordes; los primeros operadores se utilizan para bordes borrosos y los segundos para bordes finos y nítidos.

En cambio, el algoritmo de Canny, es el detector más popular que se usa para detección de bordes y probablemente, el más eficiente, comparado con los anteriormente mencionados. Dicha eficiencia se debe a que está enfocado en conseguir tres objetivos básicos: obtener el error mínimo, localizar todos los bordes y encontrar una única respuesta.

Otro método de segmentación de imágenes es el llamado umbralización del histograma [Gonzalez and Woods, 2008], el cual trata de encontrar un umbral τ dentro de los valores de niveles de gris, y a partir de él, binarizar la imagen para separar el fondo y los objetos. La forma más sencilla de este método es cuando se tiene un único umbral y se determina si cada píxel de la imagen pertenece al fondo o a un objeto, dependiendo de si el nivel de gris del píxel es mayor o menor que dicho umbral. Sin embargo, encontrar el valor de τ que garantice una buena segmentación no es sencillo, ya que se tiene que identificar inequívocamente los picos del histograma de la imagen. Para aliviar el problema de encontrar un umbral manualmente por inspección visual, el algoritmo de Otsu es el método más utilizado para la obtención del umbral de forma automática a partir del histograma, en el cual el umbral es seleccionado al maximizar la separación entre dos clases mediante la varianza.

Por otro lado, las técnicas de segmentación de imágenes basadas en crecimiento de regiones, realizan un agrupamiento de píxeles en zonas con características semejantes. Para una región dada, en general, los píxeles pertenecen a un objeto simple. Estas técnicas agrupan píxeles o subregiones en regiones cada vez más grandes con base en criterios de crecimiento predefinidos como el color, el valor del nivel de gris, la textura, entre otras. El algoritmo consiste en empezar con un conjunto de puntos *semilla* y, a partir de ellos, hacer crecer las regiones. Una semilla no es más que un nivel de gris de un píxel de la imagen y éstas son usualmente seleccionadas por el usuario de forma supervisada.

Los dos problemas principales que se presentan en el crecimiento de regiones son la selección de las *semillas generadoras* que representen correctamente las regiones de interés y la selección de las propiedades que permiten decidir si un pixel pertenece a una región o a otra. Otro problema en el crecimiento de regiones es la formulación de una condición de paro, ya que el crecimiento de la región debe

detenerse cuando no haya más píxeles que satisfagan los criterios de inclusión a la región en cuestión [Gonzalez and Woods, 2008].

Otra técnica basada en regiones es la *fusión y división de regiones*, la cual consiste en subdividir inicialmente una imagen en un conjunto de regiones arbitrarias y luego fusionar y/o dividir las regiones en un intento de satisfacer las condiciones de segmentación predefinidas. En esta técnica se empieza con la región entera. Si no se cumplen las condiciones, se divide la región en cuadrantes; si no se cumple para algún cuadrante, éste se subdivide en subcuadrantes, y así sucesivamente. Si sólo se utiliza la división, al final normalmente existen regiones adyacentes con propiedades idénticas. Este inconveniente puede remediarse permitiendo la fusión de algunas de estas subregiones. Para satisfacer las limitaciones de la segmentación, es necesario fusionar sólo las regiones adyacentes cuyos píxeles combinados satisfagan las condiciones.

Otro de los algoritmos clásicos de segmentación ampliamente utilizado es el llamado *Watersheds*, el cual incorpora muchos de los conceptos de las técnicas descritas anteriormente, por lo que suele producir resultados de segmentación estables. El método de segmentación Watershed se basa en los conceptos geográficos de cuenca y línea divisoria de aguas, así como en la visualización de una imagen en tres dimensiones: dos coordenadas espaciales y en la tercera la intensidad de los píxeles. La idea básica es simple. Se supone que, en la superficie de la representación tridimensional antes mencionada, se perfora un agujero en cada mínimo local y que toda la imagen se inunda desde abajo dejando que el agua suba por los agujeros a una velocidad uniforme. Cuando el agua que sube por las distintas cuencas está a punto de fusionarse, se construye una *presa* para evitar la fusión. Este procedimiento debe continuar hasta alcanzar el máximo global. Estos límites de las presas corresponden a las líneas divisorias que separan a los objetos del fondo. Para construir las líneas divisorias requeridas por el algoritmo de segmentación de Watershed, se requiere que la imagen sea binaria. La forma más sencilla de construir estas líneas que separan conjuntos de puntos binarios es utilizar la operación morfológica de la dilatación.

La aplicación directa del algoritmo de segmentación de Watershed puede producir una sobresegmentación que puede ser lo suficientemente grave como para hacer que el resultado del algoritmo sea prácticamente inútil, ya que el número de regiones segmentadas es muy grande. Para controlar la segmentación excesiva se utilizan *marcadores*. Un marcador es un componente conectado perteneciente a una imagen. Hay marcadores internos, asociados a objetos de interés, y marcadores externos, asociados al fondo. La selección de marcadores puede ir desde procedimientos sencillos basados en valores de intensidad y conectividad, hasta descripciones más complejas que implican tamaño, forma, ubicación, distancias

relativas, textura, etc.

En cuanto a las técnicas modernas de segmentación relacionadas con el aprendizaje computacional, existen algunas redes neuronales artificiales profundas basadas en la operación de convolución, las cuales trabajan de *extremo a extremo*, conocidas comúnmente como *redes convolucionales*. Esto significa que, para obtener una imagen segmentada de salida, la imagen de entrada puede utilizarse sin algún tipo de preprocesamiento. Esto quiere decir que no se requiere, por ejemplo, la eliminación de ruido, una transformación de color o el balance de blancos [Ratnasingam, 2019].

Estas redes son utilizadas ampliamente para clasificar objetos en una imagen; sin embargo, varias de ellas son utilizadas para segmentar los objetos que forman parte de una imagen. Una de las ventajas que tienen estas redes neuronales convolucionales profundas sobre los métodos clásicos es la capacidad de aprender las representaciones de las características de los objetos del problema en cuestión en un modo de entrenamiento completo o simultáneo, en lugar de calcular en una primera fase características generadas mediante fórmulas que requieren de experiencia por parte del usuario [Sultana et al., 2020].

2.3. Aprendizaje profundo

El aprendizaje computacional se define como el campo de estudio que permite programar computadoras para que éstas aprendan a partir de los datos [Géron, 2017]. El aprendizaje computacional se puede dividir en cuatro grandes categorías: el aprendizaje supervisado, el aprendizaje no-supervisado, el aprendizaje semi-supervisado y el aprendizaje por refuerzo [Müller and Guido, 2017]. A continuación, se presenta una breve descripción de los tipos de aprendizaje arriba mencionados.

Aprendizaje supervisado. En el aprendizaje supervisado se tienen datos de entrada y de salida conocidos, así el algoritmo es capaz de aprender mediante una etapa de entrenamiento que a partir de cierta entrada, se debe producir una salida específica. Es decir, se busca aproximar una función en particular, a partir de pares ordenados de datos entrada-salida. Una vez que el algoritmo está entrenado, éste puede etiquetar la salida de datos desconocidos de entrada.

Aprendizaje no supervisado. A diferencia del aprendizaje supervisado, en el no-supervisado solamente los datos de entrada son conocidos y los datos de salida, al ser desconocidos, no son proporcionados al algoritmo. Naturalmente, estos algoritmos son más difíciles de entender y evaluar, pero permiten explorar el espacio de características de los datos de entrada para encontrar una estructura u organización subyacente, lo cual suele ser conocimiento muy valioso para las empresas hoy en día.

Aprendizaje semi-supervisado. Algunos algoritmos de aprendizaje computacional operan sobre datos de entrenamiento parcialmente etiquetados. Es decir, la configuración usual es que se cuenta con pocos datos etiquetados y los datos sin etiquetar son muchos más que los datos etiquetados. Los algoritmos semi-supervisados se pueden utilizar para tareas no supervisadas y supervisadas.

Aprendizaje por refuerzo. En el aprendizaje por refuerzo, el modelo aprende a realizar acciones en un entorno para maximizar una función de recompensa. Así, para un problema dado, generalmente en cada tiempo t, un agente inicia en un estado (s_t), ejecuta una acción (a_t), alcanza un nuevo estado (s_{t+1}) y recibe una recompensa (r_t) del entorno. A través de las repeticiones de episodios (secuencias de estados, acciones y recompensas) el agente aprende a realizar acciones que maximicen la recompensa acumulada [Goldie and Mirhoseini, 2020].

Dentro de la categoría del aprendizaje supervisado, se encuentra el aprendizaje profundo, el cual es una extensión del campo de las redes neuronales artificiales. Los modelos dentro de este paradigma se encuentran estructurados en capas de neuronas. Estas capas se parametrizan mediante *pesos*, los cuales son optimizados durante el proceso de entrenamiento. Así, el *conocimiento* de un modelo se almacena en los pesos, y el proceso de aprendizaje consiste en encontrar valores adecuados para éstos [Bengio, 2009].

Los métodos utilizados en el aprendizaje profundo buscan aprender las características inherentes de los datos a través de jerarquías. Dichas jerarquías son representadas por diferentes niveles de abstracción, mediante los cuales, la composición de características simples o de niveles inferiores, permiten aprender características complejas o de niveles superiores. Al final de este tipo de algoritmos, se obtiene un mapeo adecuado entre la entrada y la salida.

Los modelos tradicionales basados en redes neuronales artificiales son llamados *poco profundos*, ya que su arquitectura está expresada mediante una, dos o tres capas, lo cual implica que solo cuenten con una o dos capas ocultas. Sin em-

bargo, los modelos de aprendizaje profundo, típicamente tienen una arquitectura de seis, siete o más capas. Inherentemente, un incremento en el número de capas está asociado a un incremento en la complejidad computacional, lo que suele traducirse en largos tiempos de procesamiento.

2.3.1. El algoritmo de retropropagación

El algoritmo de *retropropagación* (o de propagación hacia atrás) es un método de aprendizaje supervisado diseñado para que una red neuronal artificial *aprenda*. La finalidad principal es establecer la asociación que existe entre los patrones de entrada y sus etiquetas de clase correspondientes. Este método, está basado en la generalización de la *regla delta* y puede ser aplicado a modelos de redes con más de dos capas de neuronas [Hilera and Martínez Hernando, 1995].

Este algoritmo contempla dos fases de cálculo diferentes para que una red utilice un conjunto de pares entrada-salida como ejemplo de patrón y, por consecuencia, pueda aprenderlo: el *paso hacia adelante* y el *paso hacia atrás* [Hilera and Martínez Hernando, 1995, Haykin, 2009].

En el paso hacia adelante, los pesos sinápticos permanecen inalterados en toda la red neuronal. Posteriormente, se le presenta un patrón de entrada en la primera capa de la red y la información se propaga de neurona a neurona desde la capa de entrada hasta generar una salida en la última capa, pasando por todas las capas ocultas. A continuación, se calcula el error para cada una de las neuronas de salida, a partir del resultado obtenido y el valor deseado (la etiqueta de clase).

El paso hacia atrás comienza desde la capa de salida, como su nombre lo indica, propagando "hacia atrás" los valores de error obtenidos en el paso hacia adelante a través de la red, hasta que cada neurona de la capa de entrada haya sido alcanzada. Las neuronas de cada capa oculta producen un error de retropropagación asociado a su contribución sobre el error total de la red. Este proceso es iterativo y permite actualizar en cada iteración los pesos sinápticos de acuerdo con la regla delta (basado en el gradiente de los pesos sinápticos) para cada una de las neuronas, a partir del error que cada una de ellas recibe, de manera que, cuando nuevamente sea presentado el mismo patrón a la red, su salida esté más cercana del valor deseado.

La relación entre los pesos sinápticos actuales y los siguientes está dada por:

$$W_{t+1} = W_t - \eta \Delta w_t \tag{2.3}$$

donde W son los pesos sinápticos, t es la iteación actual, η es la tasa de aprendizaje y Δw son los gradientes de los pesos sinápticos.

El algoritmo de retropropagación permite adaptar los pesos de las neuronas en las capas ocultas para aprender la relación existente entre un conjunto de patrones y sus salidas correspondientes, logrando con ello la capacidad de *generalización*. Esta característica es importante, ya que permite a la red neuronal obtener salidas satisfactorias a patrones que no ha conocido durante la etapa de entrenamiento. Para ello, la red debe encontrar una representacion interna que le permita generar las salidas deseadas, tanto para los patrones de entrenamiento como para los patrones que no le fueron presentados. Para mayor detalle del algoritmo, ver el Anexo A.

El algoritmo de retropropagación, originalmente fue utilizado para entrenar redes con arquitectura de perceptrones multicapa (redes con más de dos capas), las cuales pretendían resolver las limitaciones de los perceptrones simples y empíricamente parecían capaces de aprender funciones complejas. Sin embargo, no estaba teóricamente claro si esta capacidad empírica tenía limitaciones no descubiertas [Bharath and Reza Bosagh, 2018]. Fue hasta que se demostró que los perceptrones multicapa eran capaces de representar funciones arbitrarias [Cybenko, 1989] que, el algoritmo de retropropagación tuvo un gran impulso logrando con ello su popularidad que ha mantenido hasta la actualidad.

Debido al incremento de los datos disponibles para realizar el entrenamiento de una red neuronal, mediante el algoritmo de retropropagación, se han propuesto diversas arquitecturas para realizar el entrenamiento de redes neuronales profundas con grandes cantidades de datos. Para lograr lo anterior es prioritario acelerar el entrenamiento debido a los largos tiempos de procesamiento que consume el algoritmo secuencial. Entonces, para acelerar el algoritmo del gradiente descendente estocástico integrado en el algoritmo de retropropagación se tienen dos alternativas: realizar un ajuste adaptativo de la tasa de aprendizaje o calcular los gradientes más rápido [Keuper, 2016]. Sin embargo, la primera no es tan viable, ya que se corre el riesgo de estancarse en un mínimo local en el intento. En cambio, la segunda es viable desde el punto de vista del cómputo distribuido.

Para realizar el entrenamiento distribuido de una red neuronal puede aprovecharse el paralelismo de los datos o el paralelismo propio del modelo. En el paralelismo de los datos se divide el conjunto de datos de entrenamiento en $b = N/n$

lotes, donde N es la cardinalidad del conjunto de entrenamiento y n el número de nodos trabajadores en el clúster; posteriormente es necesario replicar estos datos en cada nodo trabajador. De esta forma, cada uno de los lotes es procesado por un nodo trabajador, de modo que éste obtiene un conjunto de gradientes que se acumulan en el nodo maestro para lograr la actualización del modelo principal. Por otro lado, el paralelismo del modelo consiste en crear copias del modelo en cada nodo trabajador y que cada uno de estos modelos actualice cierto número de parámetros del modelo principal [Dean et al., 2012]. En cambio, en el paralelismo de datos se tienen inconvenientes, por ejemplo, cada réplica del modelo principal es entrenado con una parte del conjunto de datos. Este número pequeño de muestras puede afectar el rendimiento de la clasificación [Liu et al., 2016], por ello se debe considerar el número de nodos trabajadores para obtener un rendimiento similar al entrenamiento secuencial o paralelo.

La relación existente entre los pesos sinápticos actuales y los siguientes de forma distribuida está dada por [Liu et al., 2020]:

$$W_{t+1} = W_t - \eta \sum_{p=1}^{n} \Delta w_t^p \tag{2.4}$$

donde W son los pesos sinápticos, t es la iteración actual, η es la tasa de aprendizaje, Δw son los gradientes de los pesos sinápticos y p el número de nodos trabajadores. De esta forma, se puede utilizar el gradiente descendente estocástico y el algoritmo de retropropagación para entrenar de forma distribuida una red neuronal artificial.

En la literatura, [Keuper, 2016] proponen una arquitectura para acelerar el entrenamiento de una red neuronal basados en dos niveles de paralelismo, el primero es respecto a los datos y el segundo respecto a las tareas. El algoritmo consiste en distribuir los datos en diferentes GPUs, dentro de una misma computadora, para que cada partición sea procesada por un nodo trabajador (GPU). El procedimiento que llevan a cabo es el siguiente:

1. El maestro divide el conjunto de datos en lotes y los envía a los nodos trabajadores.

2. Cada nodo trabajador realiza el paso hacia adelante y el paso hacia atrás para obtener un conjunto de gradientes que son enviados de vuelta al maestro.

3. El maestro combina estos gradientes y realiza una actualización de los pesos.

4. Finalmente, los pesos actualizados son enviados a cada nodo trabajador y el proceso se repite.

En la Figura 2.2 se puede ver el proceso descrito anteriormente.

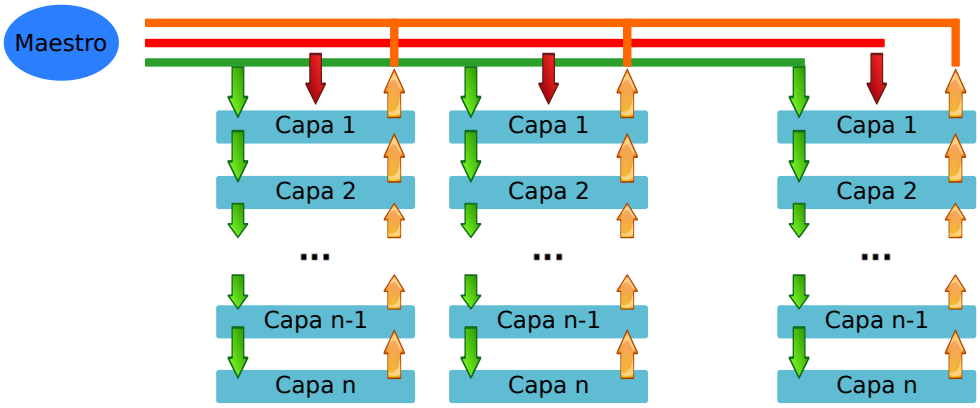

Figura 2.2: Algoritmo de retropropagación distribuido. Paso hacia delante (en verde), paso hacia atrás (en naranja) y actualización de los pesos (en rojo). Fuente: modificado de [Keuper, 2016]

Por otro lado, Apache Spark posee una biblioteca para aprendizaje computacional llamada MLLib, la cual es utilizada por [Gupta et al., 2017], para realizar el entrenamiento en cascada, el cual consiste en dar como entrada a un modelo la salida de otro previamente entrenado. Utilizan algoritmos como árboles de decisión y bosques aleatorios, para modificar el conjunto de entrenamiento, es decir, realizan una transformación de características para posteriormente emplear este conjunto de datos modificado como entrada a un perceptrón multicapa.

2.3.2. Redes neuronales convolucionales

Las redes neuronales convolucionales o CNNs se encuentran clasificadas dentro del aprendizaje profundo. Éstas surgieron del estudio del córtex visual del ojo humano, y han sido usadas en reconocimiento de imágenes desde 1980. En los últimos años, gracias al incremento en la capacidad de cómputo y la cantidad de información disponible para entrenamiento, las CNNs han tenido auge en algunas tareas complejas de visión por computadora. Además, las CNNs no están limitadas a tareas de visión, sino que también son útiles en otras tareas, tales como el reconocimiento de voz o el procesamiento del lenguaje natural [Géron, 2017].

Las CNNs utilizan aprendizaje supervisado, esto les permite clasificar objetos dentro de una imagen de entrada, utilizando para ello las distintas características obtenidas mediante sus capas de convolución y considerando la etiqueta de clase correspondiente. Para ello, las CNNs contienen varias capas ocultas especializadas y con una jerarquía; por ejemplo, las primeras capas pueden detectar líneas o curvas dentro de una imagen; y se pueden ir especializando hasta llegar a capas más profundas que reconocen formas complejas, como un rostro o la silueta de un animal. Esto significa que las neuronas de la primera capa detectan exactamente la misma característica, sólo que en diferentes lugares de la imagen de entrada. Esto es así, ya que los pesos y los sesgos son tales que las neuronas ocultas pueden detectar, por ejemplo, un borde vertical, y probablemente esta característica también aparezca en otros lugares de la imagen.

En la primera capa no es suficiente extraer una sola característica para obtener toda la información contenida en la imagen de entrada, por lo que se requiere detectar características en múltiples niveles. Al resultado de esta extracción se le conoce como *mapa de características*. Los pesos y sesgos en una misma capa definen un *núcleo* o un *filtro*. Por lo tanto, una capa convolucional completa consiste en varios mapas de características.

Las redes neuronales convolucionales están organizadas en capas de dos tipos: convolucionales y de submuestreo (ver Figura 2.3). La diferencia fundamental entre una capa densamente conectada, usada en redes poco profundas, y una capa de convolución, es que la capa densa aprende patrones globales dentro de su espacio de características de entrada, mientras que las capas convolucionales aprenden patrones locales, en el caso de imágenes, en pequeñas ventanas de dos dimensiones [Chollet, 2018].

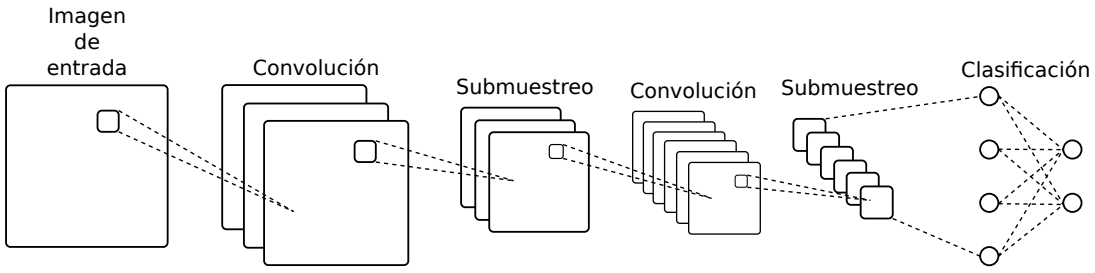

Figura 2.3: Red convolucional estándar con dos capas convolucionales (la primera con 3 filtros y la segunda con 6), dos capas de submuestreo y dos capas completamente conectadas. Fuente: Propia

Las capas convolucionales son acompañadas de una capa de submuestreo, la cual toma cada salida del mapa de características de la capa convolucional y genera un mapa de características condensado. Por ejemplo, cada unidad de

la capa de submuestreo puede condensar una región de 2×2 pixeles en la capa anterior a un solo pixel en la capa actual. Como ejemplo concreto, una técnica común para el submuestreo se conoce como *max-pooling*, en la cual se obtiene a la salida el valor máximo de la región de entrada (ver Figura 2.4c).

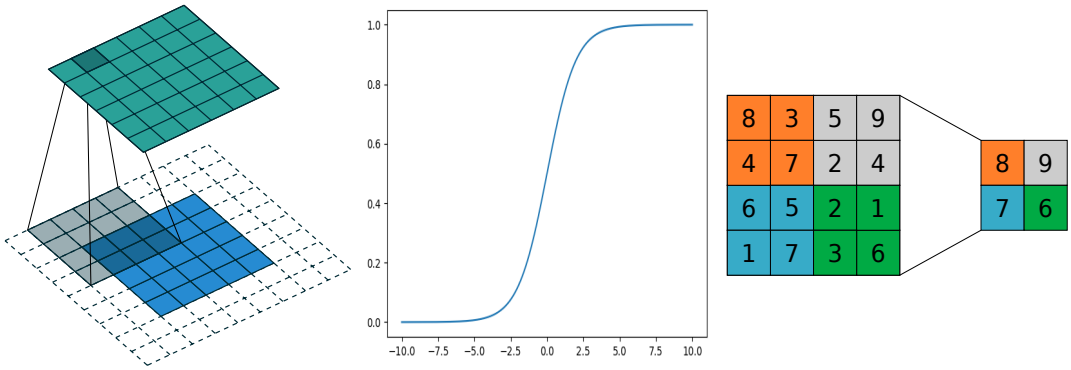

Figura 2.4: Operaciones básicas en una red neuronal convolucional: a) Operación de convolución: producto vectorial entre un kernel y la imagen. Fuente: [Dumoulin and Visin, 2016], b) Función de activación: proporciona no linealidad a la red. Fuente: Propia. c) Operación de pooling o submuestreo: reduce el tamaño de la imagen conservando sus principales características. Fuente: [Bagnato, 2018]

La capa convolucional normalmente está construida por más de un mapa de características. Así, cuando se aplica el submuestro, por ejemplo usando el max-pooling, se tiene que aplicar éste a cada mapa de características por separado (ver Figuras 2.3 y 2.4a). Por lo tanto, si hubiera tres mapas de características en la capa de convolución, se tendrían el mismo número de mapas de características a la salida de la capa de submuestreo. La idea detrás del submuestreo es que, una vez que se ha encontrado una característica, su ubicación exacta no es tan importante como su ubicación aproximada, en relación con otras características. Una gran ventaja es que hay muchas menos características agrupadas, y esto ayuda a reducir el número de parámetros necesarios en las capas posteriores.

La última capa de conexiones en la red convolucional es una capa totalmente conectada con neuronas que utilizan una función de activación para su salida (ver Figura 2.4b), usualmente utilizada para clasificación. Es decir, esta capa conecta cada neurona de la última capa de submuestreo a cada una de las neuronas de salida.

Las redes neuronales convolucionales típicamente están constituidas por seis, siete o más capas. Este número de capas tan grande como se requiera hace que

la actualización de los pesos sea difícil o imposible de realizar usando sistemas de cómputo convencionales durante la etapa de entrenamiento. A pesar de que el número de parámetros involucrados en el entrenamiento de una red convolucional, es menor que el utilizado en una red completamente conectada, los tiempos de procesamiento siguen siendo un problema que solamente puede resolverse con equipos de cómputo de alto rendimiento, tales como clústers computacionales.

A continuación, se presentan dos arquitecturas de redes convolucionales, las cuales son usadas frecuentemente para la segmentación de objetos dentro de una imagen digital: AlexNet y U-Net.

2.3.3. Arquitectura de la AlexNet

La red neuronal convolucional AlexNet fue propuesta en el año 2012 para la competición ImageNet [Krizhevsky et al., 2012]. Esta consiste en clasificar un gran número de imágenes de alta resolución dentro de las 1000 clases posibles. Originalmente, el entrenamiento de la red convolucional fue realizado en dos GPUs de forma paralela.

La arquitectura de la red se muestra en la Figura 2.5, la cual consiste en ocho capas de neuronas de las cuales cinco son convolucionales y tres son completamente conectadas.

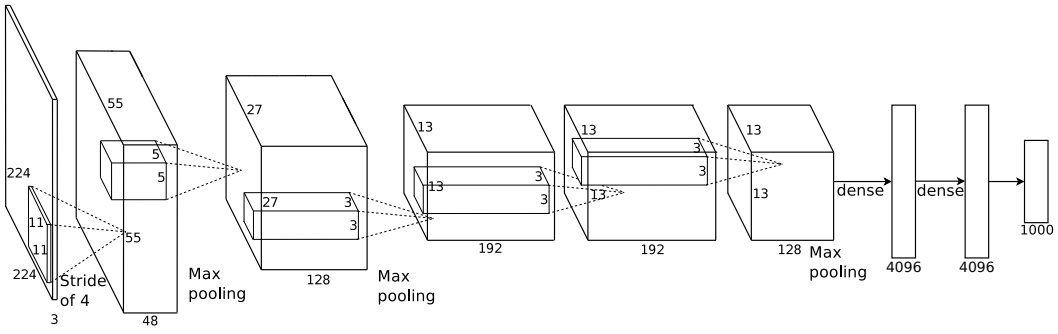

Figura 2.5: Arquitectura de la red neuronal convolucional AlexNet. Fuente: modificada de [Krizhevsky et al., 2012]

La red AlexNet fue utilizada originalmente para tareas de clasificación; sin embargo, para realizar segmentación de imágenes, esta arquitectura fue modificada para ser completamente convolucional (FCN, por us siglas en inglés), es decir, se reemplazaron las capas completamente conectadas por capas convolucionales [Rudolph et al., 2019], tal como se muestra en la Figura 2.6.

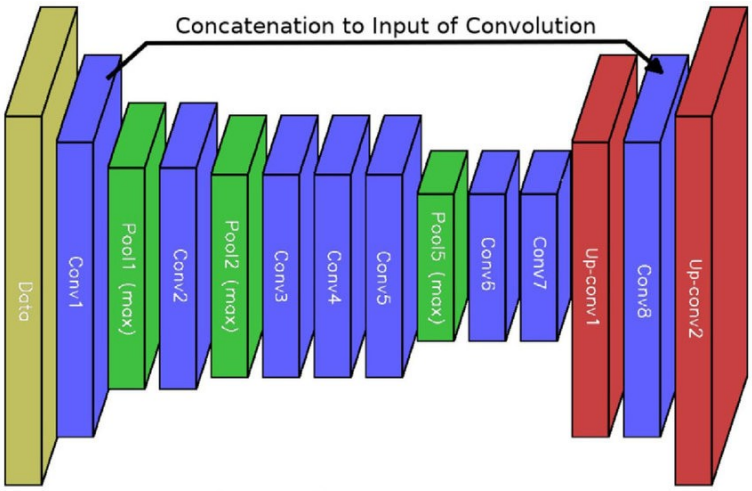

Figura 2.6: Arquitectura de la red neuronal convolucional FCN AlexNet. Fuente: [Rudolph et al., 2019]

2.3.4. Arquitectura de la U-Net

La red neuronal convolucional U-Net, fue originalmente usada para segmentar imágenes con estructuras celulares obtenidas a través de microscopios [Ronneberger et al., 2015]. La U-Net que se muestra en la Figura 2.7, tiene una arquitectura de una red encoder-decoder, aunque con algunas peculiaridades que la diferencian de ésta.

La arquitectura de la U-net consiste en dos etapas: contracción y expansión. La primera etapa es una red neuronal convolucional estándar, la cual consiste en la aplicación repetida de convoluciones con filtros de tamaño 3×3 y sin relleno, lo cual reduce las dimensiones de la imagen. A continuación, se utiliza una capa de unidades rectificadoras lineales (ReLu), las cuales son usadas como funciones lineales de activación y una capa de max-pooling de 2×2, con un tamaño de paso de 2 para lograr una reducción a la mitad de la dimensiones de la imagen, mientras en cada reducción se incrementa al doble el número de filtros, es decir, se obtiene el doble de mapas de características.

En cada paso de la etapa de expansión, se realiza una deconvolución del mapa de características, el cual se concatena con el correspondiente mapa de características de su contraparte de contracción; sin embargo, éste último debe recortarse para que ambos tengan las mismas dimensiones. El recorte que se realiza al mapa de características es necesario, ya que aun cuando en cada deconvolución las dimensiones de la imagen se incrementan, no se obtienen las mismas dimensiones.

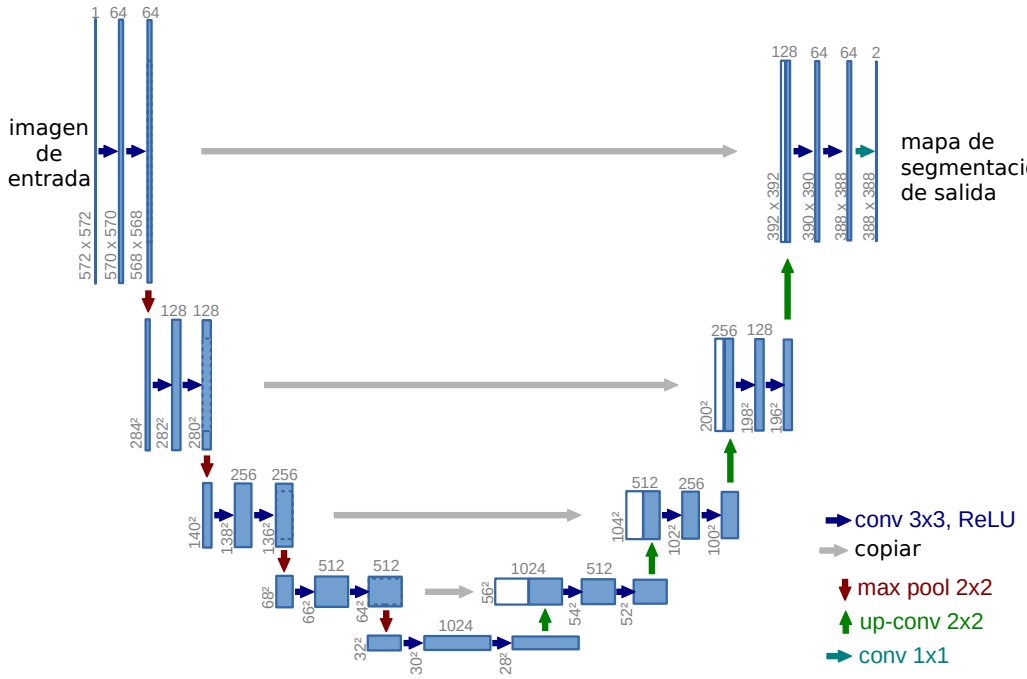

Figura 2.7: Arquitectura de la red neuronal convolucional U-Net. Fuente: [Ronneberger et al., 2015]

Posteriormente, se añaden dos convoluciones con filtros de 3×3 y una capa ReLu. En la capa final, se utiliza una convolución con un filtro de tamaño 1×1 para mapear cada característica al número de clases que se tienen. En total, la U-Net tiene 23 capas convolucionales [Ronneberger et al., 2015].

2.4. Cómputo distribuido

Por la relevancia que han adquirido los datos y la información en varios campos del conocimiento actualmente, a nuestro periodo de últimos años se le conoce como la era de la información y del *Big Data*. Es decir, durante los últimos años, se han creado más datos que en toda la historia de la humanidad. Este incremento se debe principalmente al uso de sensores y dispositivos conectados a través de Internet [Sze et al., 2017].

Una técnica para procesar cantidades grandes de datos y extraer información útil es el cómputo distribuido. Este según [Tanenbaum and Steen, 2008], se logra mediante un grupo de computadoras independientes que no comparten memoria ni ejecución de programas, pero que dan al usuario la impresión de constituir un

único sistema coherente. Además, las computadoras están interconectadas a través de una red de área local que les permite colaborar y realizar tareas específicas [Liu, 2004].

De lo anterior, se tiene que una característica importante en los sistemas distribuidos es que, las diferencias entre las computadoras (si hubiera alguna diferencia) y la manera en que se comunican entre sí, quedan ocultas para el usuario. Adicionalmente, se busca que los usuarios y aplicaciones puedan interactuar con un sistema distribuido de manera consistente y uniforme, sin importar cuando y dónde tenga lugar [Liu, 2004].

Un sistema distribuido, al estar formado por computadoras independientes, debería permitir un fácil crecimiento horizontal. Sin embargo, debe conservar oculta la manera de cómo éstas forman parte del sistema, ya que éste siempre debe estar disponible, incluso cuando algunos componentes llegaran a estar fuera de servicio.

Según [Coulouris et al., 2001] algunas ventajas que ofrecen los sistemas distribuidos son las siguientes:

1. Concurrencia. En una red de computadoras, la ejecución de programas simultáneamente es la norma.

2. Inexistencia de reloj global. Cuando los programas necesitan cooperar, coordinan sus acciones mediante el intercambio de mensajes.

3. Tolerancia a fallos. Cada componente del sistema puede fallar independientemente, permitiendo que los demás continúen su ejecución.

4. Escalable. El sistema conserva su efectividad cuando ocurre un incremento significativo en el número de recursos y el número de usuarios.

5. Heterogeneidad. Permite que los usuarios accedan a servicios y ejecuten aplicaciones sobre un conjunto heterogéneo de redes y computadoras.

2.4.1. Tipos de arquitecturas y software

La arquitectura de un sistema distribuido se refiere, por un lado, a la especificación de los componentes que la forman y, por el otro, a la manera en cómo interactúan entre ellos. A continuación se mencionan algunas de las arquitecturas

más empleadas cuando se trata de sistemas distribuidos.

Clúster

Esta arquitectura se refiere a un conjunto de computadoras construido, mediante el uso de hardware común y que se comportan como si fueran una única computadora [Liu, 2004]. En los sistemas distribuidos en clúster, los nodos de cómputo son controlados a través de una red de alta velocidad y se accede a ellos mediante un solo nodo maestro (ver Figura 2.8). Por lo general, el nodo maestro adminstra la participación de los nodos en una aplicación en particular, manteniendo una cola de procesamiento por lotes y actualizando una interfaz para los usuarios.

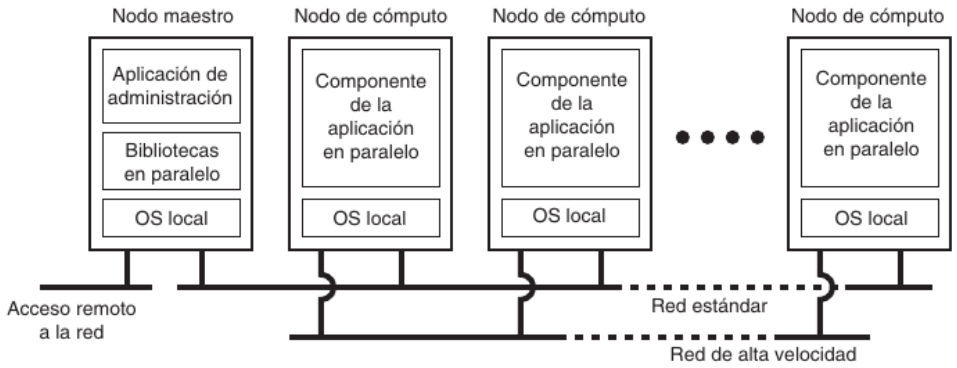

Figura 2.8: Ejemplo de sistema de cómputo en clúster, con un nodo maestro y varios nodos de cómputo. Fuente: [Tanenbaum and Steen, 2008]

Las características principales de un clúster son:

- Alto rendimiento. Ofece altas prestaciones en cuanto a la capacidad de cálculo se refiere.
- Alta disponibilidad. Si algún proceso o recurso falla dentro del clúster, éste es capaz de iniciarlo nuevamente en algún otro nodo del clúster.
- Escalabilidad. Permite incrementar el tamaño del clúster (número de nodos) sin alterar la percepción de una única entidad.
- Balanceo de carga. Los procesos se reparten completamente o por partes en los distintos nodos, considerando la carga que ya tenga cada uno de ellos.

Dependiendo del tipo de hardware y software configurado en cada nodo de un

clúster, éste puede ser: homogéneo, semihomogéneo o heterogéneo. Así, un cluster es homogéneo cuando tienen el mismo hardware y el mismo sistema operativo. Se dice que el clúster es semihomogéneo cuando el rendimiento de cada nodo es diferente pero guardan similitud ya sea en hardware o en el sistema operativo, mientras que en un cluster heterogéneo no existe tal similitud, ni en hardware ni en el sistema operativo.

Algunas de las aplicaciones de los clústers son el supercómputo para servidores web y comercio electrónico, además de bases de datos de alto rendimiento.

Grid

Esta arquitectura de cómputo distribuido, al igual que un clúster, permite que varias computadoras operen como si fueran una sola. A diferencia del clúster, no existe un nodo maestro que se encargue de coordinar las tareas que se desean resolver. Además, esta arquitectura tiende a crear sistemas completamente heterogéneos por lo que no se hacen especificaciones respecto al hardware, el sistema operativo, ni del tipo de red que se utiliza para interconectar a todos los dispositivos.

Algunos beneficios que proporcionan los sistemas distribuidos grid son :

- Explotación de recursos.
- Capacidad de CPU paralelos.
- Acceso a recursos adicionales.
- Balanceo de recursos.
- Fiabilidad.

Middleware

El *middleware* es la parte del sistema distribuido que permite la transparencia de recursos entre computadoras. Representa a una capa de software que se encuentra entre el sistema operativo y la aplicación. De manera general, el middleware es un conjunto de servicios, que permiten distribuir datos y procesos, a través de sistemas multi-tarea, dentro de una red local o remota.

En la Figura 2.9, se observan cuatro computadoras conectadas en red y tres aplicaciones, de las cuales la aplicación B está distribuida entre las computadoras 2 y 3. A cada aplicación se le ofrece la misma interfaz. El sistema distribuido proporciona los medios para que los componentes de una sola aplicación distribuida se puedan comunicar ente sí, pero también permite la comunicación entre las diferentes aplicaciones. Al mismo tiempo, oculta de la mejor manera posible, las diferencias que se presentan entre el hardware y los sistemas operativos para cada aplicación.

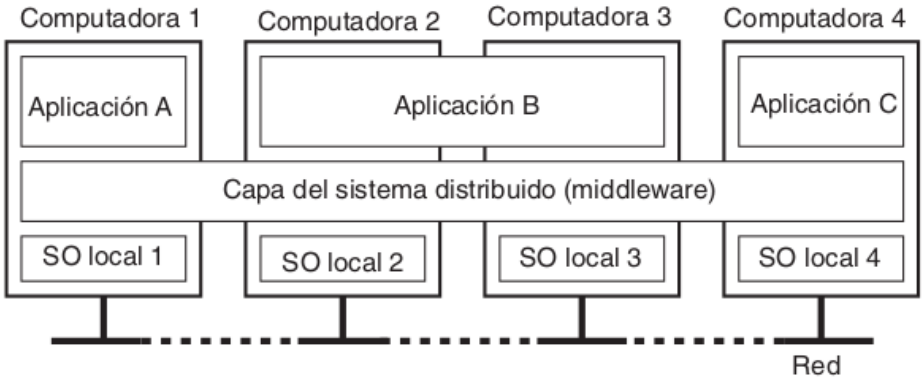

Figura 2.9: Ejemplo de sistema middleware, la capa de middleware se extiende sobre diversas computadoras. Fuente: [Tanenbaum and Steen, 2008]

2.4.2. Apache Spark

En un corto tiempo, Apache Spark ha emergido como la siguiente generación de motor de procesamiento para Big Data, usando cómputo distribuido, el cual está siendo usado en la industria cada vez con mayor frecuencia. Spark tiene mejoras con respecto a su familiar directo, Apache Hadoop (el cual inició la revolución del Big Data) en varios aspectos: es más rápido, más práctico debido a su amplia variedad de APIs, y va más allá de aplicaciones batch (por lotes), soportando una variedad de trabajos, tales como consultas interactivas, streaming, aprendizaje computacional y procesamiento de grafos.

Apache Spark es una plataforma de clúster computacional diseñado para ser rápido y de propósito general. Spark extiende el modelo MapReduce, para soportar eficientemente y a una velocidad superior, un mayor número de cálculos computacionales [Karau et al., 2015]. La velocidad es importante en el procesamiento de grandes conjuntos de datos, y es la diferencia entre explorar los datos interactivamente o esperar por minutos e incluso horas para obtener un resultado.

Una de las principales habilidades de Spark, es ejecutar cálculos en memoria. Además, logra una mayor eficiencia que Hadoop para aplicaciones complejas que se ejecutan en dispositivos de almacenamiento masivo. Spark ofrece simples APIs en diferentes lenguajes: Python, Java, Scala y R; ofreciendo una variedad amplia de bibliotecas e integrando herramientas para Big Data y aprendizaje computacional (ver Figura 2.10).

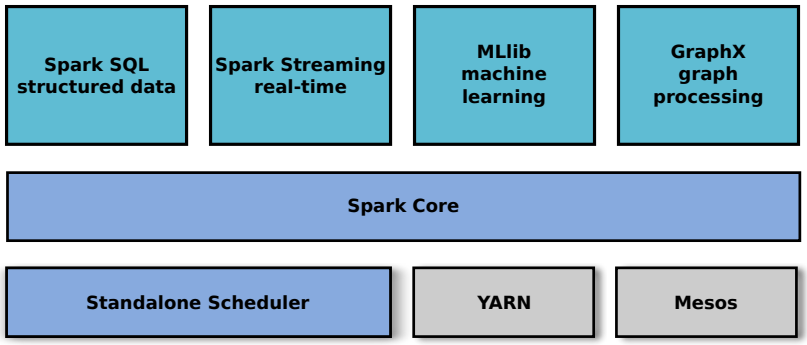

Figura 2.10: Estructura básica de Apache Spark. Fuente: [Karau et al., 2015]

A continuación se hace una descripción general de cada uno de los componentes principales de Apache Spark.

SparkCore contiene la funcionalidad básica de Spark. Incluye componentes para tareas como la administración, manejo de memoria, recuperación de fallas, interacción con sistemas de almacenamiento y más. SparkCore también ofrece la API para los Conjuntos de Datos Resilientes (RRD, por sus siglas en inglés), los cuales son una de las principales abstracciones de datos en Spark.

SparkSQL trabaja con datos estructurados, permitiendo consultas vía SQL y HQL y soportando diversos formatos de entrada de datos distribuidos, tales como tablas Hive, Parquet y JSON.

SparkStreaming habilita el procesamiento de transmisiones de datos en tiempo real. Ejemplos de estas transmisiones son: archivos de registro producidos por servidores web, colas de mensajes almacenando actualizaciones de estado publicados por usuarios a través de algún servicio web.

MLlib es la biblioteca de funcionalidades comunes de aprendizaje computacional, la cual contiene múltiples tipos de algoritmos para distintas tareas, incluyendo clasificación, regresión, agrupamiento y filtrado colaborativo.

GraphX es una biblioteca para manipular grafos y su rendimiento en cómputo paralelo. También ofrece varios operadores para manipular grafos (por ejemplo,

subgraph y mapVertices, entre otros.), y una biblioteca para algoritmos comunes para grafos (por ejemplo, PageRank).

2.4.3. La ley de Amdahl y la aceleración distribuida

Cuando se tiene un algoritmo, en el cual existe una porción del mismo que puede ser paralelizada y por lo tanto existe una porción no paralelizable (es decir, que debe ejecutarse exclusivamente en forma secuencial), la ley de Amdahl predice el comportamiento respecto al tiempo de ejecución, cuando se agrega más capacidad de cómputo, para realizar su ejecución más rápido que con la capacidad de cómputo secuencial [Pierfederici, 2016]. En este tipo de algoritmos se considera que, a partir de la suma del tiempo de ejecución de ambas porciones, se obtiene el tiempo de ejecución del algoritmo completo.

Sin embargo, el tiempo requerido para completar el algoritmo en ningún caso puede ser menor que el tiempo utilizado por la porción secuencial (debido a su naturaleza debe ejecutarse en un solo procesador), es decir, el incremento en la capacidad de cómputo solo afectará la parte del algoritmo que puede ser paralelizada. De esta forma, dado un algoritmo que es parcialmente paralelo, se cumple la siguiente relacion:

$$T(n) \geq S * T(1) + \frac{P * T(1)}{n} \tag{2.5}$$

donde $T(n)$ es el tiempo de ejecución del algoritmo usando n procesadores, $T(1)$ es el tiempo de ejecución del algoritmo usando un solo procesador, S es el porcentaje del código que es secuencial y P es el porcentaje del código que es paralelo.

Adicionalmente se puede ver que, cuando el número de procesadores tiende a infinito, el segundo término se vuelve muy pequeño en comparación con el primero. Como resultado, el tiempo de ejecución total es aproximadamente igual al tiempo de ejecucion de su parte secuencial en un solo procesador. Esto quiere decir, que la mayor parte de las ocasiones un algoritmo no puede ser paralelizado por completo.

La ley de Amdahl ayuda a estimar cuánto aumento en velocidad se puede esperar al incrementar la capacidad de cómputo y cuándo se debe dejar de utilizar el hardware como medio para reducir el tiempo de ejecución del algoritmo. La ley de Amdahl funciona, no solo para los sistemas paralelos, sino también para los sistemas distribuidos e híbridos, es decir, para los sistemas paralelos-distribuidos.

Sólo se debe tener en cuenta que, en estos últimos casos, n se refiere al número total de procesadores en el sistema.

En las aplicaciones distribuidas, según [Zuberek, 2015], se puede obtener una aproximación de la ley de Amdahl al dividir el total de la carga de trabajo entre los procesadores del sistema que suelen tener cierto grado de paralelismo de las tareas distribuidas. De esta forma, una de las medidas principales de rendimiento de una aplicación distribuida es su *aceleración* que suele definirse como el cociente entre el tiempo de ejecución de la aplicación en un único procesador $T(1)$ y el tiempo de ejecución de la misma carga de trabajo en un sistema compuesto por n procesadores $T(n)$, es decir:

$$S(n) = \frac{T(1)}{T(n)} \tag{2.6}$$

Capítulo 3

Desarrollo del proyecto

En la primera sección de este capítulo, se describen las características del hardware como del software utilizados en la implementación de las redes neuronales convolucionales de forma secuencial y distribuida. En la segunda sección, se presenta la configuración del clúster computacional empleada en ambas implementaciones. Finalmente, en la tercera sección se especifica cada uno de los módulos que componen la propuesta desarrollada en este trabajo de tesis.

3.1. Especificaciones de hardware y software

Para la implementación secuencial, los experimentos se realizaron en una computadora con un procesador Intel®Xeon®CPU E5-2630 v3 @ 2.40GHz, el cual tiene 32 hilos de procesamiento y ejecuta el sistema operativo Ubuntu 18.04.3 LTS, con 16 GB de memoria RAM, 20 MB en memoria caché, y dos discos de almacenamiento, el primero de 120 GB SSD para el sistema operativo y el segundo de 1 TB HDD para almacenamiento en general.

Para la implementación distribuida, los experimentos se realizaron sobre un clúster que consta de un nodo maestro y tres nodos trabajadores. Los nodos trabajadores tienen las características de la computadora utilizada en la implementación secuencial. Por otro lado, el nodo maestro cuenta con un procesador Intel®Xeon®CPU E5-2620 @ 2.00GHz, el cual tiene 12 hilos de procesamiento y ejecuta el sistema operativo Ubuntu 18.04.3 LTS, con 16 GB de memoria RAM, 15 MB en memoria caché, y discos de almacenamiento de 512 GB HDD y 1 TB HDD.

En cuanto al software se refiere, se ha utilizado Apache Spark 3.1.2 como principal administrador para realizar el cómputo distribuido. El lenguaje de programación utilizado fue Python 3.6.9 y las siguientes bibliotecas:

- keras 2.4.3
- numpy 1.19.0
- opencv 4.2.0.34
- pillow 7.1.2
- pip 20.2.3
- pydicom 2.0.0
- pyspark 3.0.0
- scikit-learn 0.23.1
- tensorflow 2.2.0
- matplotlib 3.2.2

3.2. Configuración del ambiente de trabajo

La alta complejidad computacional de las redes neuronales convolucionales se traduce en largos tiempos de procesamiento, razón por la cual se requiere acelerar el proceso de entrenamiento por medio del cómputo distribuido. Para realizar el entrenamiento distribuido de la red neuronal convolucional se ha utilizado un clúster computacional que tiene la arquitectura mostrada en la Figura 3.1. Este clúster está formado por un nodo maestro y tres nodos trabajadores, los cuales se comunican mediante una red de área local de alta velocidad.

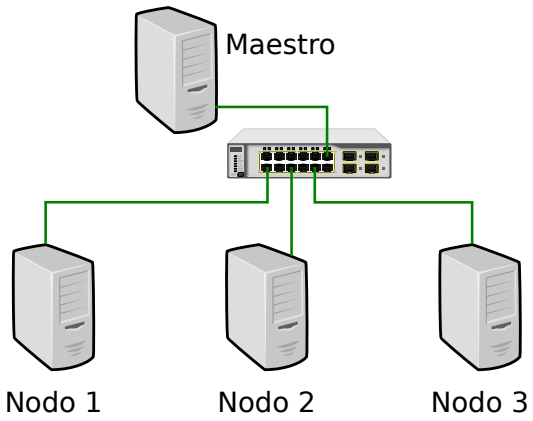

Figura 3.1: Arquitectura del clúster computacional para procesamiento distribuido. Fuente: propia

Para configurar correctamente el clúster y realizar el procesamiento de tareas distribuidas con Apache Spark, se crean tres archivos de configuración en el di-

rectorio de configuración (`$SPARK_HOME/conf`). El primero de ellos `$SPARK_HOME/conf/slaves`, almacena las direcciones IP de cada nodo trabajador dentro de la red local. El segundo `$SPARK_HOME/conf/spark-defaults`, contiene configuraciones que el usuario preestablece para cualquier aplicación de Spark, por ejemplo la dirección del nodo maestro, la cantidad de memoria para el maestro y los trabajadores, el número de hilos de procesamiento disponibles, el directorio de la máquina virtual de Java, entre otras configuraciones predeterminadas. Pot último, el tercer archivo `$SPARK_HOME/conf/spark-env` almacena las variables de entorno para identificar adecuadamente los nodos dentro de la red local y la configuración que debe adoptar.

3.3. Módulos del proyecto

La implementación de software, correspondiente al presente proyecto de tesis, consta de tres módulos principales. El primero de ellos es el *módulo de U-Net modificada*, el cual es una modificación realizada a la arquitectura original de la U-Net para adaptarla al problema de segmentación de nódulos pulmonares. El segundo módulo es llamado *módulo de segmentación de nódulos pulmonares*, el cual es requerido para realizar la segmentación de los nódulos pulmonares contenidos en las tomografías computarizadas, tanto en la implementación secuencial como en la distribuida. Por último, el tercer módulo es el del algoritmo de retropropagación distribuido, necesario para entrenar de manera general cualquier red de tipo *feed-forward*, y en particular realizar la segmentación distribuida con la red convolucional U-Net modificada en un sistema basado en Apache Spark.

Antes de que el módulo de segmentación pueda realizar su tarea, se requiere obtener las RoIs dentro de las CTs. Para ello, es necesario hacer un filtrado de las CTs almacenadas de forma local en archivos DICOM (*Digital Imaging and Communications in Medicine*), ya que no todas son útiles, según se indica en la Sección 4.1.

El proceso de filtrado tiene como primer paso la lectura de la base de datos para posteriomente realizar una normalización a las CTs, de unidades Hounsfield (HU por sus siglas en inglés) a valores de niveles de grises. Después, estas imágenes son almancenadas en formato PNG, tal como se muestra en la Figura 3.2.

Este proceso, adicionalmente, crea archivos de texto con información como: la ruta a la imagen, el tamaño de la imagen, las coordenadas de las regiones delimitadoras (cuadros), la clase a la que pertenece el nódulo, etc. Los cuadros

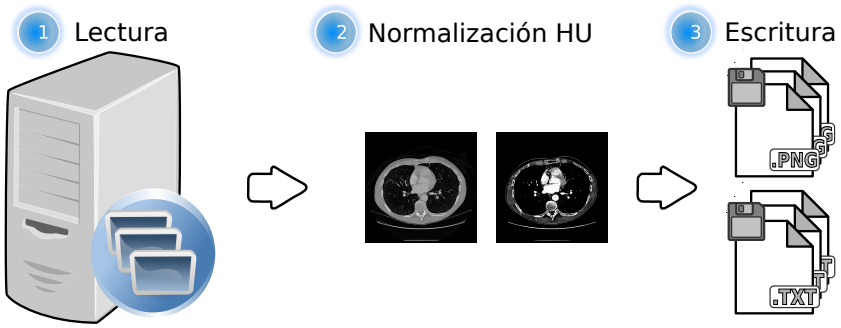

Figura 3.2: Diagrama del proceso de filtrado y normalización de las CTs. Fuente: propia

delimitadores han sido elegidos de acuerdo con el centroide proporcionado por los especialistas. Tanto las imágenes y los archivos de texto creados en este paso serán utilizados como punto de partida para realizar la segmentación de los nódulos pulmonares.

3.3.1. Módulo de U-Net modificada

En este trabajo de tesis se propone una modificación a la arquitectura de red convolucional U-Net original (ver Sección 2.3.4), la cual ha sido adaptada para realizar la segmentación de nódulos pulmonares con imágenes de la base de datos pública LIDC-IDRI. La arquitectura de la red propuesta se muestra en la Figura 3.3.

A diferencia de la arquitectura original, las imágenes de entrada y salida conservan el mismo tamaño, es decir, ambas tienen una resolución de 128×128 píxeles. Esto se debe a que tanto la convolución como la deconvolución se realizan con relleno cero, lo que evita que las dimensiones de la imagen tengan que reducirse en cada operación de convolución. Como consecuencia, la concatenación (indicada con las flechas grises y bloques blancos en la Figura 3.3) de los mapas de características en la etapa de deconvolución se realiza sin tener que recortarlos. Por esta razón, dicha operacion es directa, abarcando completamente la imagen, por lo que con la red U-Net modificada, no hay pérdida de información en la segunda etapa.

El número de filtros iniciales es de 32, cada uno con un tamaño de 3×3 y en cada paso de la primera etapa este número se duplica (64, 128, etc.) hasta alcanzar los 512 filtros. Al mismo tiempo, las dimensiones de la imagen se reducen

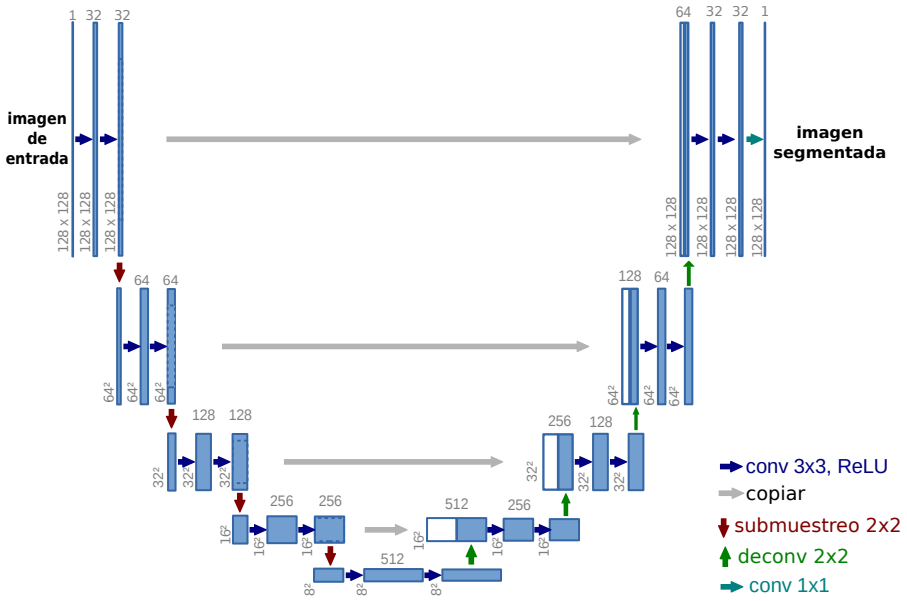

Figura 3.3: Arquitectura de la red neuronal convolucional U-Net modificada. Fuente: modificada de [Ronneberger et al., 2015]

a la mitad debido a la capa de submuestreo máximo (max pooling) de 2×2 y un paso de 2. En la etapa de deconvolución, el número de filtros se reduce a la mitad, mientras que las dimensiones de la imagen se duplican hasta alcanzar el tamaño de la imagen de entrada. En la última capa de la U-Net modificada, se realiza la segmentación para dos clases, es decir, se decide si un píxel pertenece a un nódulo o no.

3.3.2. Módulo de segmentación de nódulos pulmonares

Este módulo es el encargado de realizar la segmentación de los nódulos pulmonares a partir de las imágenes PNG creadas en la etapa de preprocesamiento y que fueron guardadas en la unidad de almacenamiento local. Este módulo también requiere de los archivos de texto, en los cuales se han guardado las coordenadas de los cuadros delimitadores e información adicional. El proceso de segmentación se muestra en la Figura 3.4. Específicamente, este procedimiento consiste en la lectura de las imágenes y los archivos de texto para obtener el cuadro delimitador. Un inconveniente es que los tamaños o dimensiones de los cuadros delimitadores tienden a variar dependiendo de la localización del nódulo y del tamaño del mismo. Dado que la red neuronal U-Net debe recibir imágenes de un tamaño fijo de 128 $\times$ 128 pixeles, entonces las coordenadas de los cuadros delimitadores, se ajustan

`float`	`train_network( ... )` Realiza el entrenamiento de la red, recibe los hiper-parámetros, y el número de épocas como argumentos.
`float`	`test_network( ... )` Realiza la etapa de prueba sobre un lote de imágenes a la vez.
`Generator`	`get_data( ... )` Obtiene el generador para cargar, por lotes, las imágenes en memoria.
`Model`	`create_model( ... )` Crea el modelo de la U-Net con los métodos para ser entrenado.

Cuadro 3.1: Funciones principales para ejecutar el algoritmo de segmentación.

(utilizando las coordenadas del centroide del nódulo) para obtener las dimensiones correctas, ya que en ningún caso el tamaño de los cuadros delimitadores coincide con las dimensiones requeridas.

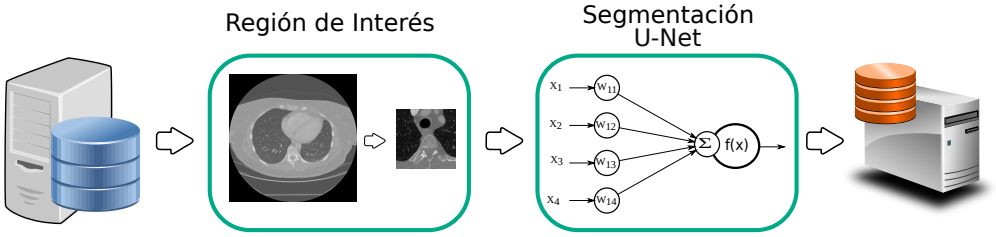

Figura 3.4: Diagrama del proceso para realizar la segmentación de nódulos en tomografías computarizadas. Fuente: propia

Las funciones principales para obtener la segmentación de nódulos se muestran en el Cuadro 3.1. La función principal es `train_network`, la cual inicia con la lectura de las imágenes y las pasa al modelo de la U-Net modificado para iniciar con la etapa de entrenamiento.

3.3.3. Módulo del algoritmo de retropropagación distribuido

Para realizar el entrenamiento distribuido de la red neuronal U-Net, se requiere de un proceso iterativo, el cual se divide en tres etapas principales: mapeo, entrenamiento y reducción, las cuales pueden observarse en la Figura 3.5. Este algoritmo

está inspirado en el mostrado por [Keuper, 2016], con la diferencia de que la presente propuesta está implementada para un clúster computacional gestionado por Apache Spark en vez de utilizar una sola computadora con múltiples GPUs.

La etapa de mapeo inicia en el nodo maestro (*driver*) y es la encargada de crear un modelo M de la U-Net modificada que servirá como referencia para obtener las medidas de rendimiento finales, así como para actualizar los pesos de la red neuronal en cada iteración. Esto se debe principalmente a que, los pesos de la red son difundidos (mediante una variable *broadcast* de Apache Spark) en los nodos trabajadores. Además, esta etapa tiene la tarea de particionar el conjunto de imágenes a procesar y enviar la información pertinente a cada nodo trabajador para procesar las imágenes que le corresponden. Para ello, todos los nodos del clúster deben tener almacenada la base de datos en la misma ruta, ya sea usando el sistema de archivos local o el sistema de archivos distribuido.

Una vez que cada nodo tiene las particiones correspondientes que le fueron asignadas y los pesos W_i difundidos por el nodo maestro, se inicia la etapa de entrenamiento en los nodos trabajadores. De esta manera, se puede crear un submodelo M_j,$(j = 1, ..., p)$ por cada partición. Posteriormente, se realiza el entrenamiento del submodelo M_j, con la partición correspondiente y durante una sola época. Cuando ha terminado el entrenamiento, se obtienen los pesos actualizados W_i' a partir de los datos contenidos en dicha partición, de modo que es posible obtener los gradientes de dicha época usando los pesos W_i difundidos por el maestro. El resultado al final de la época es un conjunto de p gradientes, es decir, una matriz de gradientes $\Delta W_{i,j}$ por cada partición creada.

En la etapa de reducción, el nodo maestro recupera los p conjuntos de incrementos de los pesos y sus respectivos sesgos, correspondientes a cada una de las particiones que fueron utilizadas en la etapa de entrenamiento. Esta etapa se realiza mediante una operación de *reducción* iniciada desde el nodo maestro, que aplica el promedio de cada gradiente de peso y sesgo, según el número de particiones existentes que están repartidas en todos los nodos trabajadores, tal como se realiza en la investigación hecha por [Liu et al., 2020], con la finalidad de obtener una única matriz de gradientes ΔW_i de pesos y sesgos, que serán sumados a los pesos y sesgos W_i del modelo principal M que reside en el nodo maestro. Este proceso de actualización de pesos W_{i+1} se debe realizar en cada época hasta cumplir con alguna condición de paro establecida por el usuario o hasta cumplirse el número total de épocas seleccionadas.

Apache Spark proporciona métodos Map y Reduce que facilitan el diseño del algoritmo distribuido mencionado en la Subseccion 2.3.1. También, TensorFlow proporciona las herramientas básicas para implementar de forma eficiente una red

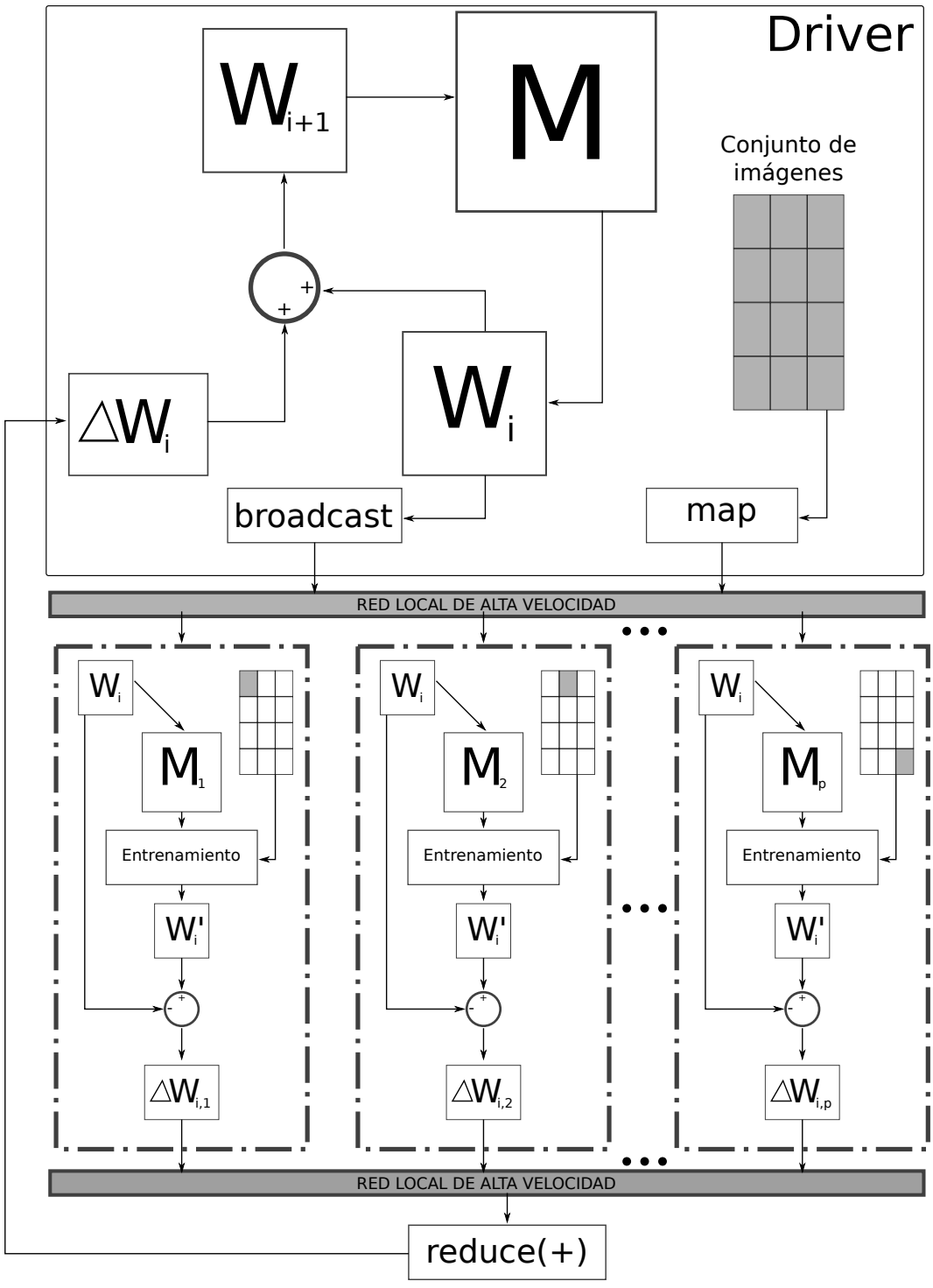

Figura 3.5: Algoritmo de retropropagación distribuido. Fuente: propia

neuronal profunda (entrenada mediante el algoritmo de retropropagación). En este proyecto de tesis, se realizó la integración para que ambos trabajen en conjunto para llevar a cabo el entrenamiento de la U-Net modificada. Además, para el funcionamiento distribuido Apache Spark proporciona características avanzadas como la tolerancia a fallos, replicación de datos y el balanceo de cargas [Liu et al., 2016]

Debido a las capacidades de Tensorflow, es posible llevar a cabo el entrenamiento de la U-Net modificada de forma distribuida, toda vez que en cada nodo trabajador se crean varias instancias independientes del modelo original, con los cuales es posible calcular los gradientes en una sola época, a partir del número de particiones de datos y el tamaño de los lotes seleccionados. Esto quiere decir que cada instancia se especializa en una partición de datos y ésta es capaz de devolver los gradientes correspondientes a esa partición.

Como es de esperarse, el procedimiento del algoritmo de retropropagación original que normalmente realiza una computadora convencional es el mismo que el descrito en párrafos anteriores. Cabe señalar que la única diferencia entre esta propuesta y el algoritmo original es que la tarea principal (el entrenamiento de la red U-Net modificada) es dividida en las subtareas antes planteadas, las cuales son ejecutadas en forma paralela y distribuida por los nodos trabajadores de Spark.

Adicionalmente y con fines comparativos, también se ha implementado en la plataforma distribuida de Apache Spark la FCN AlexNet utilizando el procedimiento descrito anteriormente para la U-Net. Para realizar el entrenamiento se ha utilizando en ambas redes el módulo aquí descrito.

Capítulo 4

Resultados

En la primera sección de este capítulo, se presentan los conjuntos de datos utilizados en este proyecto de tesis. En la segunda sección, se muestran ejemplos de uso de las redes convolucionales de manera paralela y distribuida para la clasificación de dígitos manuscritos. Finalmente, en la tercera sección se presentan los resultados de la segmentación de nódulos pulmonares utilizando las arquitecturas de redes neuronales convolucionales U-Net modificada y FCN AlexNet, realizando las comparaciones y el análisis pertinente de los resultados de tiempo y aceleración de la ejecución secuencial y distribuida, así como los resultados propios de la segmentación.

4.1. Conjuntos de datos

Con la finalidad de realizar pruebas preliminares del algoritmo de retropropagación distribuido descrito en la Sección 3.3.3, el primer conjunto de imágenes utilizado en este proyecto de tesis es el conjunto de dígitos manuscritos MNIST (*Modified National Institute of Standards and Technology*), el cual se describe en [LeCun et al., 1998b]. Este conjunto es utilizado ampliamente en métodos de reconocimiento de patrones, principalmente para clasificación, aunque también es comúnmente usado en sistemas de procesamiento de imágenes.

Los dígitos contenidos en esta base de datos, originalmente en blanco y negro, fueron tomados de la base de datos NIST (*National Institute of Standards and Technology*) y han sido normalizados respecto a su tamaño para tener dimensiones de 20×20 píxeles, conservando su relación de aspecto. Posteriormente a la norma-

lización, se le aplica al conjunto de imágenes un filtro *antialiasing* para obtener ciertos niveles de grises, las cuales fueron centradas en una imagen de 28×28 píxeles. En total, este conjunto de imágenes contiene 70000 muestras obtenidas de aproximadamente 250 personas [Hamidi and Borji, 2010]; 60000 se encuentran en el conjunto de entrenamiento y 10,000 en el conjunto de prueba, según se puede ver en el Cuadro 4.1.

Cuadro 4.1: Distribución de las clases en el conjunto de dígitos manuscritos MNIST. Fuente [Hamidi and Borji, 2010]

Dígito	MNIST	
	Entrenamiento	Prueba
0	5,923	980
1	6,742	1,135
2	5,958	1,032
3	6,131	1,010
4	5,842	982
5	5,421	892
6	5,918	958
7	6,265	1,028
8	5,851	974
9	5,949	1,009
Total	60,000	10,000

El segundo conjunto de imágenes se obtuvo de la base de datos pública LIDC-IDRI, la cual es proporcionada por el Consorcio de Bases de Datos de Imágenes Pulmonares y la Iniciativa de Recursos de Bases de Datos de Imágenes [CIA, 2022]. Esta base de datos ofrece 244617 imágenes de 512×512 píxeles, las cuales fueron obtenidas de 1308 estudios realizados a 1010 pacientes. Los nódulos marcados en esta base de datos se distinguen entre cuatro categorías, de acuerdo con la longitud del diámetro: nódulos ≥ 3 mm, nódulos < 3 mm, no-nódulos ≥ 3 mm y no-nódulos < 3 mm [McNitt-Gray et al., 2007].

Los nódulos identificados con un diámetro mayor o igual a 3 mm están caracterizados completamente, es decir, se tiene el contorno completo en tres dimensiones. De este contorno se pueden derivar otras medidas, como son el diámetro máximo y el volumen del nódulo. Adicionalmente, tiene una evaluación subjetiva de características, tales como la dificultad de detección, la estructura interna, la calcificación, la esfericidad, el margen, el grado de lobulación, la extensión de la espiculación, su solidez y la evaluación de la probabilidad de malignidad.

Para la categoría de nódulos < 3 mm, se cuenta con una aproximación del

centroide, ya que los nódulos son muy pequeños y resulta difícil definir su contorno con precisión. Debido a lo anterior, estos no cuentan con una evaluación subjetiva.

Para los no-nódulos $\geq$ a 3 mm, al igual que para la categoría anterior, solamente se cuenta con una aproximación del centroide y no cuentan con una evaluación subjetiva. Algunos de los objetos considerados (y que no son nódulos) son, por ejemplo, las cicatrices, las atelactasias y los cambios postoperatorios.

Para la última categoría, los no-nódulos $<$ a 3 mm, no tienen alguna anotación y por lo tanto no tienen evaluación subjetiva. El Cuadro 4.2 resume las anotaciones que tienen cada una de estas categorías.

Cuadro 4.2: Síntesis de las categorías de nódulos y sus anotaciones.

Categoría	Anotación	Evaluación subjetiva
Nódulo $\geq$ 3 mm	Contorno 3D	Sí
Nódulo $<$ 3 mm	Centroide aproximado	No
No-nódulo $\geq$ 3 mm	Centroide aproximado	No
No-nódulo $<$ 3 mm	Sin anotación	No

Todos los nódulos en la base de datos fueron identificados hasta por cuatro especialistas. Sin embargo, las imágenes usadas para realizar los experimentos de este trabajo son aquellas que corresponden a los nódulos que están marcados por los cuatro especialistas, por lo que los nódulos identificados por tres o menos especialistas fueron descartados. De acuerdo con lo anterior, se cuenta con un total de 900 nódulos (representados mediante 6183 imágenes), los cuales están identificados completamente y son los que se han usado para realizar el entrenamiento de las redes neuronales convolucionales.

4.2. Resultados de ejemplo de redes convolucionales para clasificación de dígitos

Debido a que el entrenamiento, tanto de la FCN AlexNet como de la U-Net con el conjunto de datos LIDC-IDRI, requiere de un tiempo de cómputo considerable, en esta sección se muestran los resultados obtenidos en la clasificación de dígitos manuscritos mediante una red convolucional estándar y el conjunto MNIST. Esto con la finalidad de tener certeza de la eficacia del algoritmo descrito en la Subsección 3.3.3, ya que tanto la red neuronal como el conjunto de datos son pequeños y obtener resultados es relativamente rápido.

4.2.1. Implementación paralela

La arquitectura de la CNN estándar utilizada para realizar la clasificación de dígitos se muestra en el Cuadro 4.3. Consta de una capa de entrada, dos capas convolucionales con sus respectivas capas de submuestreo y una capa de salida, la cual es completamente conectada con diez neuronas, una por cada clase (dígito) que se va a clasificar. Los filtros utilizados en las capas convolucionales son de tamaño 3×3, mientras que el tamaño de paso en las capas de submuestreo es de 2 con un tamaño de 2×2. El entrenamiento de forma paralela se realiza de forma estándar, es decir, se utilizan los núcleos disponibles en un solo nodo trabajador. La medida de rendimiento principal en este tipo de aplicaciones es la exactitud de clasificación, la cual se obtiene a partir del conjunto de prueba.

Cuadro 4.3: Arquitectura de la CNN estándar utilizada para la clasificación de dígitos manuscritos.

Capa	Filtros	Salida	Parámetros
Entrada	–	(28, 28, 1)	0
Conv2D_1	32	(28, 28, 1)	320
MaxPool_1	–	(14, 14, 32)	0
Conv2D_2	64	(14, 14, 64)	18,496
MaxPool_2	–	(7, 7, 64)	0
Flatten_1	–	3136	0
Dense_1	–	10	31370

La Figura 4.1 muestra la evolución de la exactitud en el conjunto de prueba según se avanza en las 50 épocas realizadas, alcanzando una exactitud de 0.97531. El resultado alcanzado es coherente con lo reportado en la literatura [LeCun et al., 1998a], por lo que se comprueba el correcto funcionamiento del algoritmo de retropropagación de manera paralela.

4.2.2. Implementación distribuida

Con la finalidad de evaluar el rendimiento del entrenamiento distribuido de la red neuronal convolucional empleada en la presente propuesta de tesis, se definen los experimentos mostrados en el Cuadro 4.4. Éstos consisten en duplicar el número de núcleos de cada nodo trabajador iniciando con un núcleo (procesamiento secuencial) hasta incluir en el procesamiento la totalidad de los núcleos, para registrar los tiempos de procesamiento de la fase de entrenamiento y en consecuencia

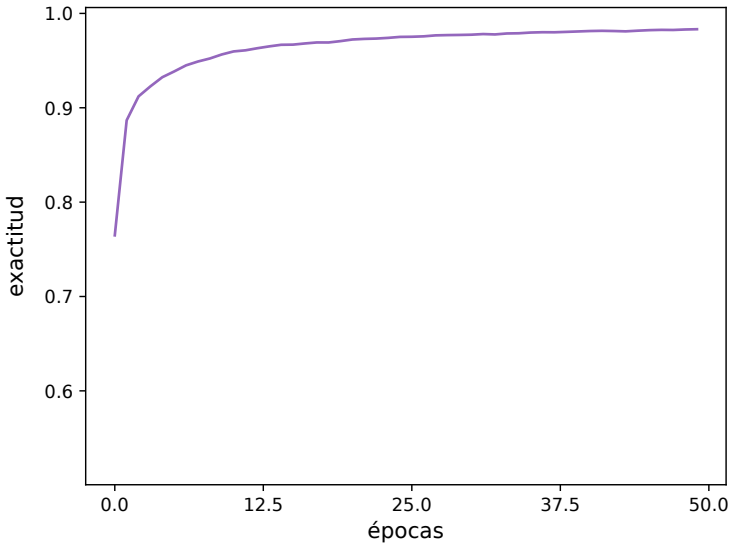

Figura 4.1: Exactitud de clasificación obtenida de una CNN estándar con el conjunto de prueba de MNIST. Fuente: propia.

calcular la aceleración correspondiente.

Cuadro 4.4: Experimentos de ejecución de forma distribuida utilizando un número diferente de núcleos en cada uno de ellos.

Prueba	Núcleos			Total
	Nodo1	Nodo2	Nodo3	
1	1	0	0	1
2	1	1	1	3
3	2	2	2	6
4	4	4	4	12
5	8	8	8	24
6	16	16	16	48
7	31	31	31	93

Las variables que determinan significativamente el rendimiento respecto al tiempo de cómputo de la red convolucional son el número de particiones realizadas sobre el conjunto de datos, y el número de núcleos utilizados durante la fase de entrenamiento.

El primer experimento realizado es ejecutado de forma secuencial (1 núcleo), ya que éste sirve como referencia para evaluar el tiempo y la aceleración de en-

Cuadro 4.5: Tiempo de entrenamiento en minutos de una CNN estándar para el conjunto de dígitos manuscritos MNIST.

Particiones	Núcleos						
	1	3	6	12	24	48	93
3	56.14	19.60	11.43	7.08	6.14	4.66	4.27
6	60.78	20.70	6.26	4.34	4.56	4.00	4.01
12	68.78	23.11	7.23	3.57	4.69	3.91	4.23
24	91.86	27.77	8.84	4.16	4.99	4.57	8.14
48	162.17	40.10	12.55	5.75	5.30	8.28	20.43

trenamiento. Éste consiste en ocupar un solo núcleo del nodo 1, mientras que los otros nodos están *fuera de línea*, ya que de éstos últimos no se ocupa núcleo alguno. Para los experimentos de forma distribuida, propiamente dicho, se toma un número determinado de núcleos en cada nodo, empezando por uno para tener un total de 3 núcleos, uno por cada nodo trabajador, incrementando al doble el número de núcleos en cada nodo trabajador, hasta alzanzar 31 núcleos (32 - 1) en cada uno de ellos. De los 32 núcleos disponibles por cada nodo trabajador, el núcleo que no se utiliza en el último experimento se deja disponible para el sistema operativo, ya que de no hacerlo así, podría llegar un momento en el que la aplicación termine de forma incorrecta.

Debido a que el número de particiones en que se divide el conjunto de datos es crucial en este tipo de experimentos, se han realizado variaciones en dicho número, tal como se presenta en la primera columna del Cuadro 4.5. En éste, se puede apreciar que se cumple el comportamiento predicho por la Ley de Amhdal, al variar el número de núcleos dependiendo del número de particiones empleado. Esto quiere decir que, el decremento en el tiempo de procesamiento suele estancarse, independientemente de la cantidad de recursos que se agreguen.

Por lo anterior y como consecuencia de las limitaciones de la memoria RAM en cada nodo trabajador, cuando se ocupan 12/24/48 particiones y 93 núcleos, los tiempos de cómputo se incrementan. Esto se debe principalmente a que se crea un modelo de CNN por cada partición.

Por otro lado, si se considera fijo el número de núcleos y se incrementa gradualmente el número de particiones, se puede observar lo siguiente. Por ejemplo, si se utiliza un solo núcleo, el tiempo de cómputo es mayor cuando se tienen 48 particiones que cuando se usan otras configuraciones de núcleos. En cambio, cuando se usan 12 y 24 núcleos esto no se cumple, ya que dependiendo del número

de particiones, las muestras que debe procesar cada submodelo pueden estar más balanceadas, lo que permite en algunos casos obtener un mejor rendimiento.

El mejor resultado respecto al tiempo de cómputo se obtuvo cuando se realiza el entrenamiento con 12 núcleos y 12 particiones. Cuando se utilizan 24 núcleos y se incrementa el número de particiones se observa una tendencia similar a la obtenida con 12 núcleos. El tiempo requerido cuando se utilizan 93 núcleos y 48 particiones es mayor que el resto de particiones, debido principalmente a problemas asociados con la memoria RAM, tal como se mencionó anteriormente.

En la Figura 4.2 puede observarse la aceleración obtenida para cada uno de los experimentos propuestos del Cuadro 4.5. Nótese que la mejor aceleración obtenida durante los experimentos se obtuvo con 48 particiones y 24 núcleos. Observe también que por cada partición, la gráfica de la aceleración también confirma el comportamiento distribuido con la Ley de Amdahl, especialmente cuando se tienen tres particiones.

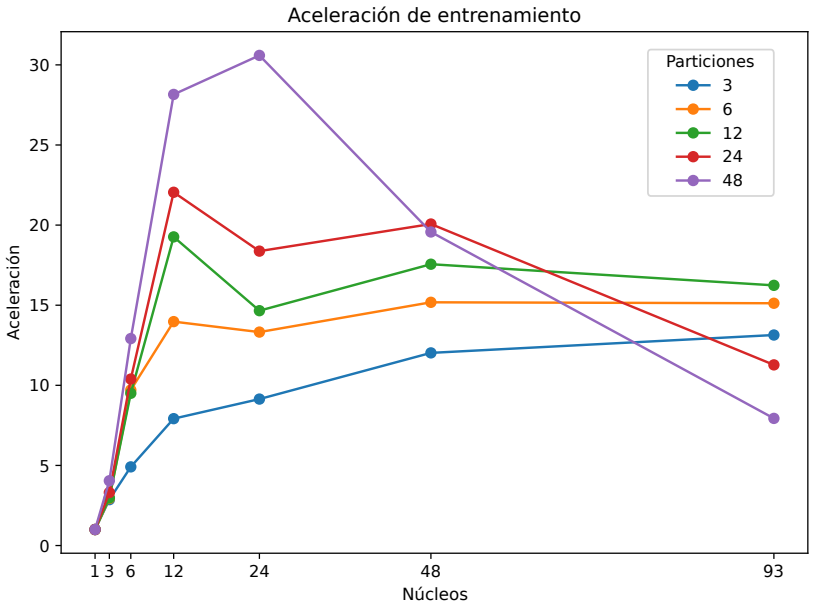

Figura 4.2: Aceleración obtenida en el entrenamiento de una CNN estándar y el conjunto MNIST. Fuente: propia.

Por otro lado, para la fase de predicción, los tiempos de cómputo se muestran en el Cuadro 4.6. En este caso, el mejor tiempo se logra cuando se tienen 24 particiones y se utilizan 24 núcleos. Respecto a la aceleración (veáse Figura 4.3) el experimento con 48 particiones presenta el mejor resultado, similar a la fase de entrenamiento, aunque éste no representa el mejor tiempo de procesamiento.

Cuadro 4.6: Tiempo de predicción en segundos de una CNN estándar para el conjunto de dígitos manuscritos MNIST.

Particiones	Núcleos						
	1	3	6	12	24	48	93
3	10.61	3.99	2.64	1.65	1.33	1.73	1.74
6	15.11	5.55	2.03	1.33	1.14	1.38	1.39
12	23.96	6.71	1.96	1.21	1.23	1.00	1.02
24	41.98	8.92	3.38	1.61	0.92	1.05	0.95
48	105.90	19.68	6.73	2.94	1.50	1.09	2.54

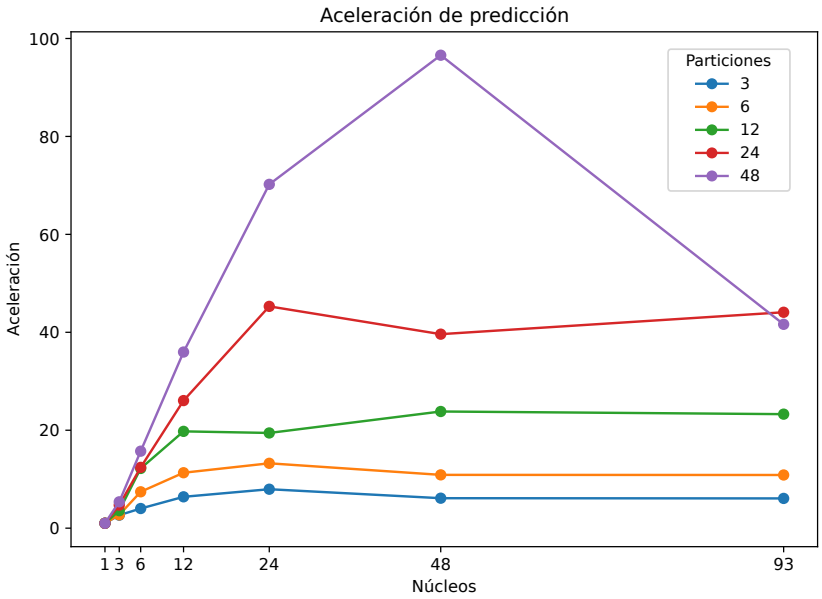

Figura 4.3: Aceleración obtenida en la predicción de una CNN estándar y el conjunto MNIST. Fuente: propia.

Como se mencionó anteriormente, la medida principal de rendimiento en estos experimentos es la exactitud de clasificación sobre el conjunto de prueba. Estos resultados se muestran en el Cuadro 4.7. Como se puede apreciar, existen cambios mínimos entre los experimentos, ya que la mayoría oscila entre 97.1 % y el 97.8 %. Sin embargo, en los casos más extremos, este valor decae hasta el 95 %. Lo anterior sucede debido a las limitaciones de la memoria RAM, entonces algunas muestras no son utilizadas en la fase de entrenamiento, ya que al saturarse la memoria, Spark finaliza algunas tareas y en consecuencia las muestras dentro de la partición con ejecución finalizada suelen ser desconocidas por el clasificador, afectando el rendimiento obtenido.

Cuadro 4.7: Exactitud de clasificación en el conjunto de prueba para el conjunto de dígitos manuscritos MNIST.

Particiones	Núcleos						
	1	3	6	12	24	48	93
3	0.9728	0.9716	0.9720	0.9730	0.9706	0.9722	0.9724
6	0.9746	0.9744	0.9780	0.9768	0.9765	0.9741	0.9767
12	0.9711	0.9706	0.9720	0.9724	0.9724	0.9726	0.9719
24	0.9716	0.9768	0.9707	0.9704	0.9719	0.9749	0.9652
48	0.9733	0.9736	0.9669	0.9662	0.9576	0.9506	0.9570

Los resultados de los experimentos anteriores han mostrado certeza de la convergencia de la red neuronal convolucional estándar utilizando el algoritmo de aprendizaje distribuido propuesto de la Subseccion 3.3.3, por lo cual se garantiza su funcionamiento para aplicarlo en la segmentación distribuida de nódulos pulmonares.

4.3. Resultados de segmentación de nódulos pulmonares con redes convolucionales

Esta sección presenta los resultados de segmentación de nódulos pulmonares de manera paralela, distribuida y secuencial. La segmentación se lleva a cabo mediante el módulo descrito en la Subsección 3.3.2 para el procesamiento de la red neuronal convolucional U-Net modificada.

Dada la anotación realizada por los especialistas en la base de datos LIDC-IDRI y una vez obtenido el resultado de segmentación de la CNN, el principal criterio de evaluación es el Coeficiente de Similaridad Sørensen–Dice (DSC, por sus siglas en inglés), dado por la siguiente Ecuación [Havaei et al., 2017]:

$$DSC = \frac{2\,|X \cap Y|}{|X| + |Y|} \tag{4.1}$$

donde X son las segmentaciones hechas por los especialistas y Y son las segmentaciones hechas por la U-Net modificada. En otras palabras, el DSC es utilizado para medir el solapamiento entre dos regiones, en este caso, los resultados de la

segmentación de los nódulos. Cuando el DSC es igual a 1, se puede decir que las regiones segmentadas están completamente superpuestas (es decir, la segmentación realizada por la U-Net modificada sería idéntica a la realizada por el radiólogo), mientras que, si el DSC es igual a 0, no existe solapamiento entre las regiones segmentadas.

4.3.1. Segmentación paralela de U-Net modificada

En una primera fase, se realiza el entrenamiento de la red neuronal convolucional utilizando los núcleos disponibles en un solo nodo trabajador, es decir, se utilizan 31 núcleos en esta fase. Para ello, el conjunto de nódulos disponibles fue particionado en los subconjuntos de entrenamiento (68 %), validación (12 %) y prueba (20 %). Los dos primeros subconjuntos fueron utilizados para obtener los mejores hiperparámetros de la red, los cuales resultan ser una tasa de aprendizaje de 0.0005, un número de filtros iniciales igual a 32, un tamaño de mini-batch de 32 y un tamaño de filtro de 3. Posteriormente, utilizando estos hiperparámetros, se entrena nuevamente la red con diferentes tamaños de subconjuntos de entrenamiento, para finalmente medir y comparar el rendimiento de la red utilizando los correspondientes subconjuntos de prueba.

Cuadro 4.8: Particionamiento realizado al conjunto de datos LIDC-IDRI en algunas publicaciones de la literatura

Experimento	Entrenamiendo [%] (# nódulos)	Prueba [%] (# nódulos)
1	56 (504)	44 (396)
2	78 (702)	22 (198)
3	45 (405)	55 (495)
4	73 (657)	27 (243)

Para los experimentos realizados, como se ha mencionado anteriormente, se han utilizado diferentes tamaños del conjunto de prueba; esto, con la finalidad de realizar una comparación con otros métodos reportados en la literatura que utilizan la misma base de datos (ver Cuadro 4.8). Se ha considerado un tamaño de 493 nódulos (experimento 3) como en el realizado por [Wang et al., 2017b]; y un tamaño de 393 nódulos (experimento 1) utilizado en [Wang et al., 2017a]. Adicionalmente, se midió el rendimiento de la red en un tamaño igual a la mitad de los tamaños anteriores (experimentos 2 y 4), con el objetivo de comparar los resultados con los reportados en [Mukherjee et al., 2017].

Los resultados de los experimentos con distintos tamaños del conjunto de

Cuadro 4.9: Resultados de exactitud de clasificación y DSC de la red convolucional U-Net modificada usando diferentes tamaños del conjunto de prueba

Conjunto de Prueba [Nod.]	Exactitud [%]	DSC [%]
493	99.72	84.27
393	99.75	85.62
246	99.76	87.46
196	99.78	88.10

prueba se muestran en el Cuadro 4.9, donde se utiliza la exactitud de clasificación y el coeficiente DSC como medidas de rendimiento de la red convolucional. Aquí, se puede observar que conforme disminuye el conjunto de prueba, aumenta la exactitud de clasificación y el coeficiente DSC. Lo anterior es un comportamiento esperado, ya que la red convolucional aprende a discriminar mejor cuando dispone de más información en el conjunto de entrenamiento.

Cuadro 4.10: Comparación entre la U-Net modificada (última fila) y otras propuestas en el estado del arte, usando la base de datos LIDC-IDRI. El símbolo " – " indica que no existe información disponible de las medidas en los trabajos correspondientes.

Método	Conjunto de Prueba [Nod.]	Exactitud [%]	DSC [%]
PN-SAMP [Wu et al., 2018]	–	97.58	74.05
MV-CNN [Wang et al., 2017a]	393	–	77.67
CF-CNN [Wang et al., 2017b]	493	–	82.15
DLGC [Mukherjee et al., 2017]	128	–	83.00
Fuzzy C-Mean [Nithila and Kumar, 2016]	–	98.95	84.00
U-Net mod. [Hernández-Solis et al., 2021]	196	99.78	88.10

Por otro lado, en el Cuadro 4.10, se muestran los resultados de segmentación obtenidos por nuestra propuesta de U-Net modificada, así como los publicados en algunos artículos que utilizan la misma base de datos (LIDC-IDRI). Comparando los resultados de los Cuadros 4.9 y 4.10, se observa que los resultados de la U-Net modificada son mejores en términos del DSC, para todos los tamaños de conjuntos de prueba utilizados. Es decir, el DSC obtenido con la U-Net modificada es superior que las publicaciones relacionadas. Esto mismo ocurre en el trabajo que utiliza un conjunto de prueba pequeño (128), en comparación con el conjunto más pequeño de la U-Net modificada (196).

Con la finalidad de ofrecer una mejor comprensión de los resultados de la segmentación de nódulos sobre el conjunto de prueba, algunos de ellos, pueden observarse en la Figura 4.4. Para mayor claridad, las imágenes se muestran en parejas con orientación vertical; la primera fila es la segmentación realizada por los especialistas (en color verde) y la segunda es la segmentación hecha por la red neuronal convolucional (en color rojo). Estos resultados de segmentación alcanzan individualmente más del 97 % del DSC, debido a la forma suave que poseen los nódulos que en su mayoría son de forma circular u ovalada. Cabe destacar que, también el tamaño del nódulo influye fuertemente, ya que la mayoría tienen un diámetro mayor a 4 mm, lo cual facilita su identificación.

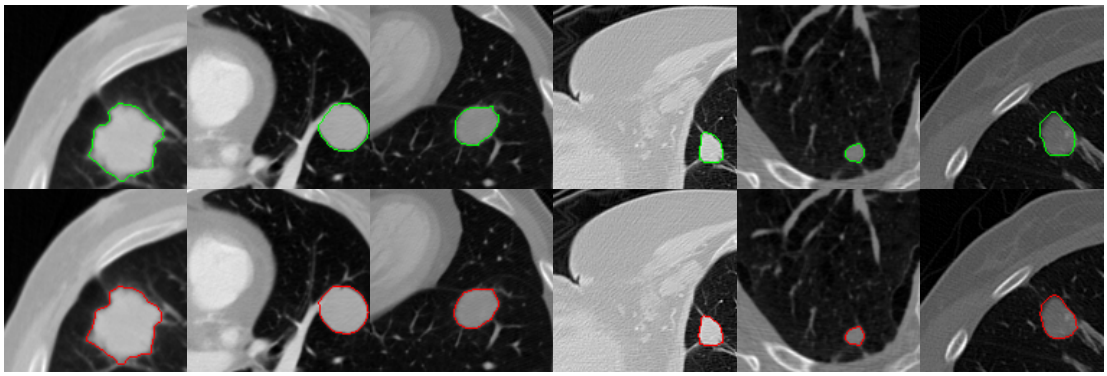

Figura 4.4: Nódulos con formas regulares, segmentados por expertos en verde (arriba) y nódulos segmentados por la U-Net modificada en rojo (abajo). Fuente: propia.

Debido a la existencia de variadas formas y tamaños de los nódulos pulmonares, no todos pueden ser segmentados correctamente. En la Fig. 4.5, se muestran algunos de los nódulos que resultan difíciles de segmentar, con un DSC individual inferior al 70 %, ya que dichos nódulos no tienen una forma suave o regular. En ocasiones, algunos nódulos presentan espículas o tienen un diámetro menor a 3 mm, lo cual dificulta en gran medida su identificación.

4.3.2. Segmentación distribuida de FCN AlexNet

Con el objetivo de comparar el rendimiento de la segmentación distribuida entre dos redes convolucionales, se retomaron los experimentos definidos en el Cuadro 4.4 para aplicarlos al conjunto de tomografías LIDC-IDRI utilizando la red neuronal FCN AlexNet, descrita en la Subsección 2.3.3. A continuación, se muestran los resultados obtenidos respecto al tiempo de procesamiento y la aceleración distribuida obtenida en cada uno de los experimentos.

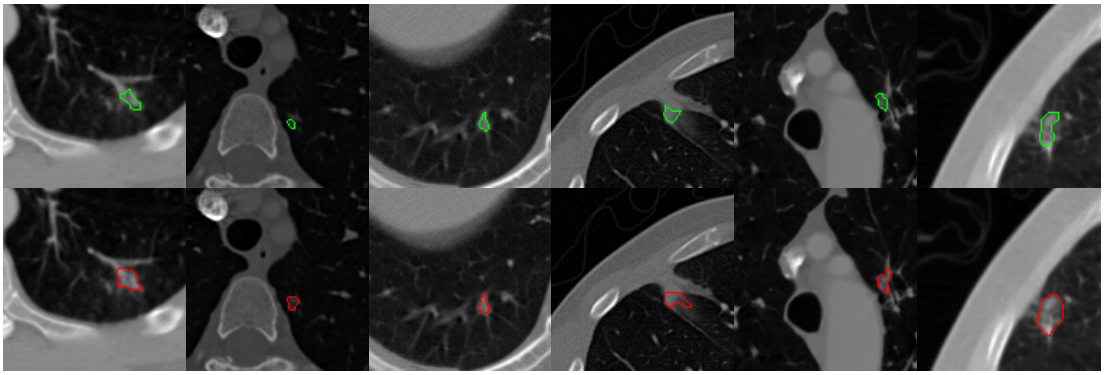

Figura 4.5: Nódulos con formas irregulares (espiculadas), segmentados por expertos en verde (arriba) y nódulos segmentados por la U-Net modificada en rojo (abajo). Fuente: propia.

El tiempo de entrenamiento requerido para cada configuración se muestra en el Cuadro 4.11. Para el análisis por fila, se puede apreciar que, conforme se incrementa el número de núcleos utilizados, el tiempo requerido para terminar la fase de entrenamiento disminuye. Sin embargo, a partir de los 24 núcleos la ganancia en tiempo es muy pequeña, lo cual está de acuerdo con la Ley de Amdahl. Este comportamiento no se cumple cuando se utilizaron 6/9 particiones y 48/93 núcleos, debido a los problemas con la RAM ya mencionados en la Subsección 4.2.2. El menor tiempo de procesamiento se logró cuando se utilizaron 6 particiones y 24 núcleos.

Cuadro 4.11: Tiempos de procesamiento para el entrenamiento (en minutos) de la FCN AlexNet para el conjunto de tomografías computarizadas LIDC-IDRI

Particiones	Núcleos						
	1	3	6	12	24	48	93
3	651.24	343.54	207.58	129.02	86.89	80.51	70.93
6	659.75	256.20	85.45	54.63	39.11	43.92	42.97
9	665.49	235.27	94.75	44.19	40.99	41.87	50.60

Por otro lado, los tiempos de procesamiento registrados cuando se realiza la fase de predicción se muestran en el Cuadro 4.12. Tal como se esperaba, se puede observar un comportamiento similar a la fase de entrenamiento, logrando el menor tiempo cuando se utilizaron 9 particiones y 24 núcleos. En cuanto a la mejora con respecto al procesamiento secuencial, el comportamiento de la aceleración obtenida durante la fase de entrenamiento puede verse en la Figura 4.6, en la cual se observa que la mejor aceleración se encuentra en el experimento que utiliza 6 particiones y 24 núcleos (poco más de 16x), lo cual también coincide con el menor

tiempo de procesamiento empleado durante la fase de entrenamiento, tal como se puede observar en el Cuadro 4.11. En cambio, la aceleración durante la fase de predicción es la obtenida con 12 núcleos y 9 particiones (poco más 9x), la cual se muestra en la Figura 4.7 y corresponde al comportamiento esperado según la Ley de Amdahl. De manera general, en ambas figuras se puede ver claramente los beneficios de utilizar cómputo distribuido, a medida que el número de núcleos y de particiones varía incrementalmente, especialmente cuando se compara contra el procesamiento secuencial (es decir, con la columna 2 de los Cuadros 4.11 y 4.12)

Cuadro 4.12: Tiempos de procesamiento de predicción (en segundos) de la FCN AlexNet para el conjunto LIDC-IDRI.

Particiones	Núcleos						
	1	3	6	12	24	48	93
3	77.36	46.98	33.44	20.99	17.20	16.61	15.83
6	67.74	29.58	18.91	13.57	14.42	13.81	13.67
9	121.94	28.78	26.28	13.07	13.06	13.24	13.10

Con respecto a los resultados de rendimiento de segmentación relacionados con la medida DSC (ver el Cuadro 4.13), se puede observar que el DSC oscila alrededor del 74 %, lo cual muestra una primera aproximación de la efectividad del módulo del algoritmo de aprendizaje de retropropagación distribuido propuesto utilizando la red convolucional FCN AlexNet.

4.3.3. Segmentación distribuida de U-Net modificada

Con el mismo objetivo de la sección anterior de comparar el rendimiento de segmentación entre dos redes convolucionales, se retomaron los experimentos del Cuadro 4.4 para evaluar el rendimiento de la U-Net modificada respecto al tiem-

Cuadro 4.13: Resultados del coeficiente DSC de la segmentación obtenida, usando el sub-conjunto de prueba de la FCN AlexNet para el conjunto LIDC-IDRI.

Particiones	Núcleos						
	1	3	6	12	24	48	93
3	0.74033	0.73011	0.73815	0.73581	0.74068	0.74484	0.74332
6	0.74318	0.73679	0.74729	0.74006	0.73486	0.73018	0.73910
9	0.74113	0.74437	0.73694	0.74715	0.73693	0.74878	0.73646

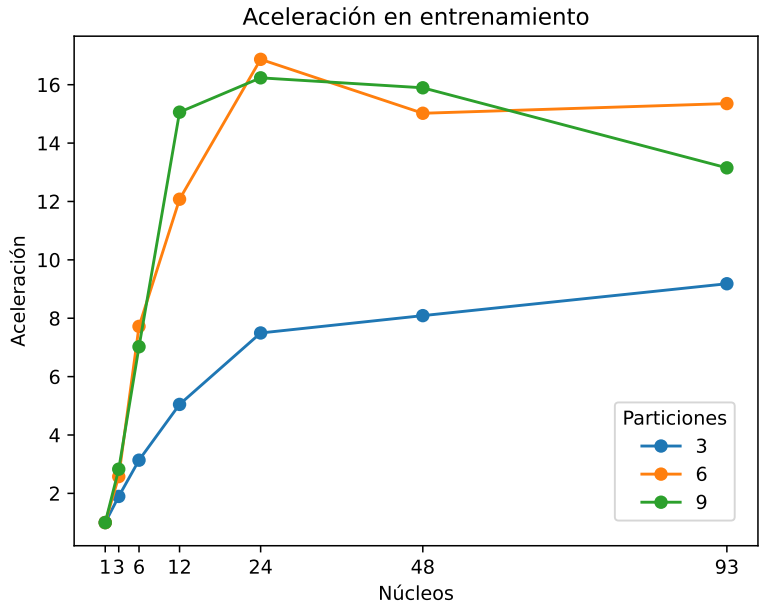

Figura 4.6: Aceleración obtenida en el entrenamiento de la FCN AlexNet y el conjunto LIDC-IDRI, usando tres configuraciones de particiones. Fuente: propia.

po de cómputo y aceleración distribuida. Sin embargo, para estos experimentos el número de particiones posibles a utilizar se ha reducido en gran medida debido a las limitaciones de memoria descritas en la Sección 4.2.2. Lo anterior significa que, la cantidad de parámetros a optimizar es mayor y el tamaño del modelo ocupa más espacio en memoria, a diferencia del modelo de clasificación de la CNN (ver Sección 4.2.2).

El experimento base para analizar los resultados de tiempo de procesamiento del entrenamiento de la U-Net modificada de forma distribuida es el que utiliza procesamiento secuencial, es decir, el experimento 1 del Cuadro 4.4. En el Cuadro 4.14 se muestran los tiempos de procesamiento en minutos de cada uno de los experimentos realizados, los cuales dependen tanto del número de particiones como del número de núcleos utilizados.

Observe que, cuando se utilizan tres particiones, se puede ver que el tiempo requerido para completar el entrenamiento de la U-Net modificada es mayor, en comparación con el resto de particiones. Esto, se debe al número reducido de particiones; esto es, al ser pocas, Spark no realiza una distribución de las muestras de forma balanceada (a diferencia de cuando se crea un mayor número de particiones). El comportamiento observado es que una de las tres particiones contiene

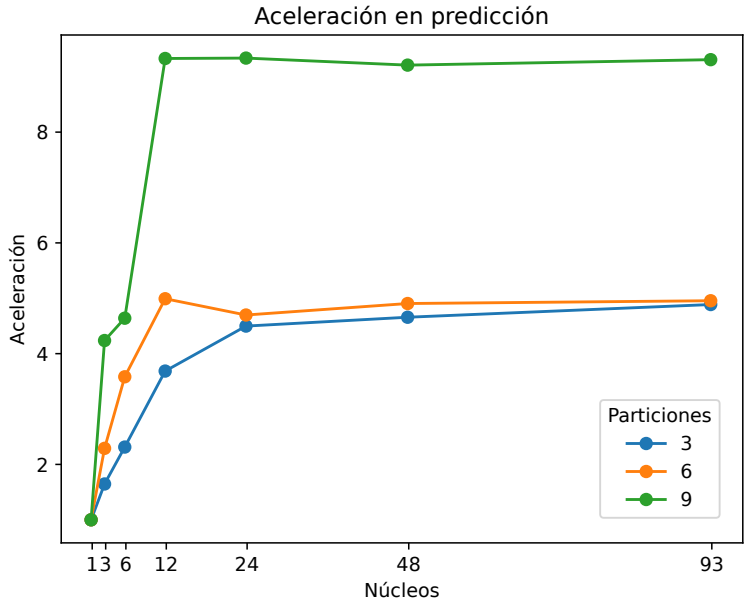

Figura 4.7: Aceleración obtenida en la predicción de la FCN AlexNet y el conjunto LIDC-IDRI, usando tres configuraciones de particiones. Fuente: propia.

Cuadro 4.14: Tiempo de procesamiento (en minutos) para el entrenamiento de la U-Net modificada utilizando el conjunto LIDC-IDRI

Particiones	Núcleos						
	1	3	6	12	24	48	93
3	2714.91	1260.45	783.05	509.26	368.85	313.99	294.75
6	2692.72	901.63	269.72	187.09	188.20	149.26	182.11
9	2697.53	916.98	344.59	184.73	172.78	189.19	220.69

un mayor número de muestras. Por tanto, cuando dos tareas terminan aproximadamente en el mismo tiempo de procesamiento, éstas deben esperar a que la última tarea termine las respectivas muestras de su partición para continuar con la siguiente iteración.

Las aceleraciones obtenidas mediante las distintas configuraciones de núcleos pueden verse en la Figura 4.8, las cuales reflejan también el comportamiento predicho por la Ley de Amdahl. Con respecto a los tiempos de procesamiento, se observa tanto en el Cuadro 4.14 como en la Figura 4.8 que se obtiene el menor tiempo de cómputo cuando se utilizan seis particiones y 48 núcleos.

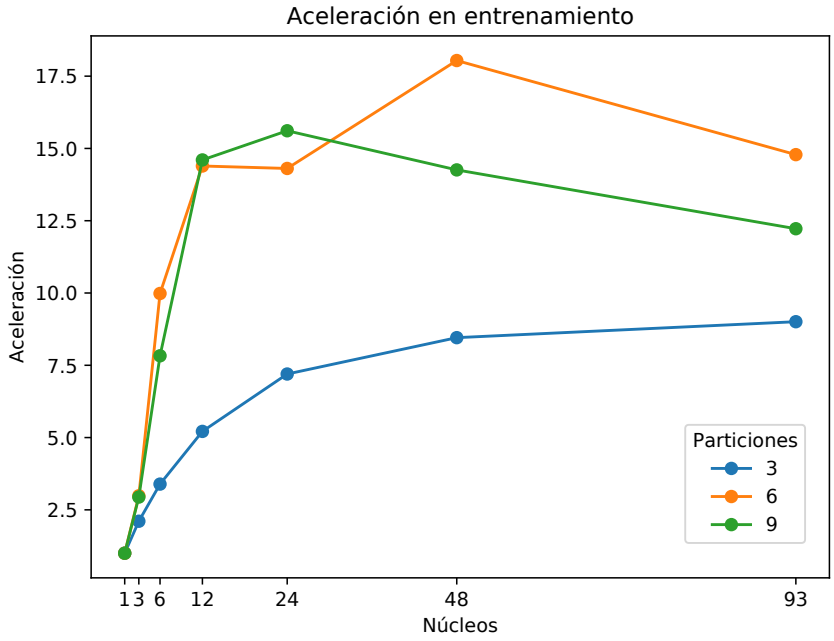

Figura 4.8: Aceleración obtenida en el entrenamiento de la U-Net modificada y el conjunto LIDC-IDRI. Fuente: propia.

Para la fase de segmentación usando el conjunto de prueba, los resultados correspondientes a los tiempos de procesamiento se muestran en el Cuadro 4.15, en donde se puede confirmar que los tiempos de predicción son más pequeños que los de entrenamiento, obteniendo el mejor resultado cuando se utilizan 9 particiones y 24 núcleos. En la Figura 4.9 pueden observarse las gráficas de aceleración correspondientes a los tiempos de procesamiento del Cuadro 4.15. La aceleración máxima obtenida es de 18.1x para la fase de entrenamiento, mientras que para la fase de segmentación se alcanzó una aceleración máxima de 15.8x.

Respecto a la medida de rendimiento usada para la segmentación de nódulos,

Cuadro 4.15: Tiempo de predicción (en segundos) de la U-Net modificada para el conjunto LIDC-IDRI.

Particiones	Núcleos						
	1	3	6	12	24	48	93
3	231.42	82.84	57.04	36.19	22.95	18.83	17.52
6	240.03	85.14	31.93	20.47	16.76	16.96	16.68
9	247.65	86.89	40.45	19.67	15.97	16.01	16.59

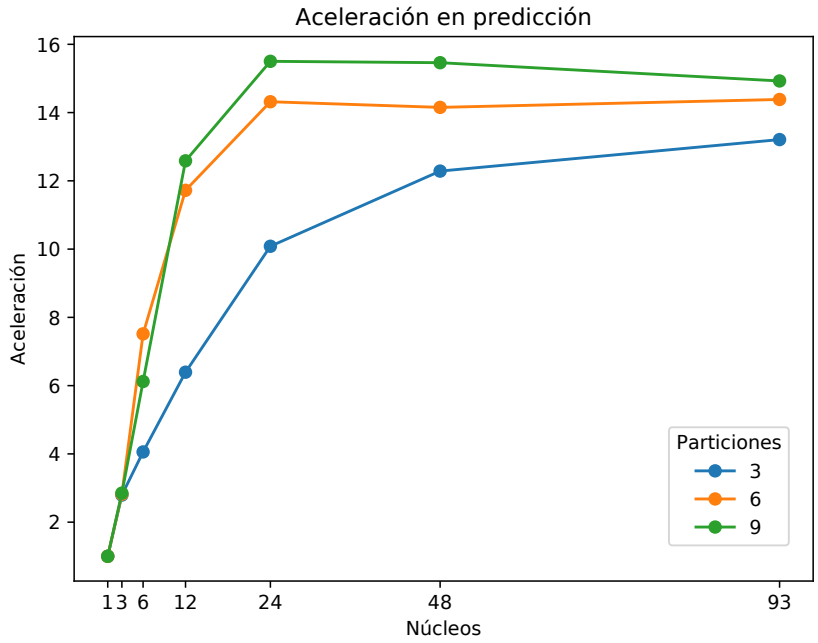

Figura 4.9: Aceleración obtenida en la predicción de la U-Net modificada y el conjunto LIDC-IDRI. Fuente: propia.

el Cuadro 4.16 muestra los diferentes DSC obtenidos en los experimentos. De acuerdo con los experimentos realizados de forma paralela (ver Sección 4.3.1), el DSC que se obtuvo es de 88.10 %, por lo que dicho resultado es el que se espera también en los experimentos realizados de forma distribuida. Los resultados del Cuadro 4.16 aunque oscilan entre 88.0 % y 88.2 %, se mantienen muy cercanos al valor esperado. Esta pequeña variación mostrada se debe a la forma en la que se le presentan las muestras a la U-Net modificada.

Con respecto a los resultados de segmentación obtenidos con la FCN Alex-Net, se puede afirmar que este tipo de red ofrece resultados aceptables, tal como

Cuadro 4.16: Resultados del DSC de la U-Net modificada con el sub-conjunto de prueba utilizando el conjunto LIDC-IDRI.

Particiones	Núcleos						
	1	3	6	12	24	48	93
3	0.88129	0.88143	0.88000	0.88100	0.88191	0.88213	0.88113
6	0.88177	0.88190	0.88213	0.88137	0.88193	0.88173	0.88171
9	0.88152	0.88088	0.88148	0.88116	0.88219	0.88012	0.88080

se puede observar en el Cuadro 4.13. Sin embargo, si comparamos el Cuadro 4.13 con el Cuadro 4.16 que corresponde a los resultados de segmentación de la U-Net modificada, se puede apreciar la diferencia significativa que existe entre los dos modelos de redes neuronales convolucionales empleados para la segmentación de nódulos pulmonares. Mientras que el rendimiento de segmentación de la FCN Alexnet oscila alrededor del 74 %, la U-Net modificada lo hace al rededor del 88 %. Esta diferencia repercute directamente en la calidad de la segmentación de los nódulos pulmonares y como consecuencia en la detección temprana del cáncer de pulmón. Aunque las velocidades de procesamiento obtenidas con AlexNet son inferiores que con U-Net, lo cual supone una ventaja con respecto a la U-Net (comparando los Cuadros 4.11 y 4.12 con los Cuadros 4.14 y 4.15), el mejor rendimiento de segmentación que ésta puede obtener (74 %) es inferior. Por tanto, el uso de la U-Net modificada en este proyecto de tesis para la tarea de segmentación de nódulos pulmonares es más adecuada que la arquitectura de FCN AlexNet.

Capítulo 5

Conclusiones y trabajo futuro

En este trabajo de tesis se ha realizado una comparación del desempeño entre las implementaciones secuencial y distribuida de dos redes neuronales convolucionales para la segmentación de nódulos pulmonares. La primera es una red neuronal convolucional con la arquitectura FCN AlexNet y la segunda una red con arquitectura U-Net, la cual fue modificada en este proyecto de tesis para adaptarla a la tarea de segmentación de nódulos. La fase de entrenamiento de ambas redes está implementada con Tensorflow y la plataforma de cómputo distribuido Apache Spark, las cuales son herramientas que juntas facilitan la implementación de redes neuronales profundas aceleradas en un sistema de cómputo en clúster. Para lo anterior, en el presente proyecto se propuso una adaptación del algoritmo de retropropagación distribuido para ejecutarse en un clúster computacional.

De manera general, el rendimiento de estas redes fue probado con diferentes configuraciones en las que se han realizado variaciones al número de particiones, el tamaño de los lotes y al número de núcleos a utilizar. En los resultados obtenidos, se ha podido observar la diferencia en los tiempos de procesamiento respecto al entrenamiento y la predicción de las redes neuronales empleadas. Como era de esperarse, los experimentos de forma distribuida ocuparon un tiempo inferior y mayor aceleración que los experimentos realizados de forma secuencial.

Después de realizar los experimentos, los índices DSC obtenidos al realizar la segmentación de nódulos con las redes FCN AlexNet y U-Net modificada utilizando los datos LIDC-IDRI, se han mantenido aproximadamente en 74 % y en 88.10 % respectivamente, sin importar el número de particiones ni el número de núcleos empleados. Esto es, los resultados obtenidos mostraron que la red U-Net modificada obtuvo un rendimiento de segmentación superior que la FCN AlexNet,

aunque para lograrlo se ha sacrificado tiempo de procesamiento por un mayor rendimiento de segmentación, el cual es el objeto de la presente tesis.

Con los resultados presentados en este trabajo de tesis, se concluye que el entrenamiento de una red neuronal puede ser mejorado respecto al tiempo de ejecución usando el cómputo distribuido (adaptando algoritmos de aprendizaje) que ofrece un clúster computacional gestionado con Apache Spark. De esta manera, se corrobora la veracidad de la hipótesis de este trabajo de tesis.

Debido a que la segmentación de los nódulos pulmonares fue realizada a partir de la región de interés definida por los especialistas, como trabajo futuro, se plantea implementar una red neuronal que sea capaz de identificar la región de interés, a partir del conjunto completo de rebanadas de la tomografía computarizada. De esta manera, podría obtenerse una implementación de extremo a extremo, en donde la salida de la red neuronal que identifica regiones sea la entrada para el modelo de red neuronal presentado en este trabajo de tesis.

Bibliografía

[Akhtar et al., 2018] Akhtar, M. N., Saleh, J. M., and Grelck, C. (2018). Parallel processing of image segmentation data using hadoop. *International Journal of Integrated Engineering*, 10(1):74–84.

[Armato and Sensakovic, 2004] Armato, S. G. and Sensakovic, W. F. (2004). Automated lung segmentation for thoracic CT. *Academic Radiology*, 11(9):1011–1021.

[Bagnato, 2018] Bagnato, J. I. (2018). ¿Cómo funcionan las convolutional neural networks? visión por ordenador. Recuperado octubre de 2020, de: https://www.aprendemachinelearning.com/como-funcionan-las-convolutional-neural-networks-vision-por-ordenador/.

[Bengio, 2009] Bengio, Y. (2009). Learning deep architectures for ai. *Foundations and Trends in Machine Learning*, 2(1):1–127.

[Bharath and Reza Bosagh, 2018] Bharath, R. and Reza Bosagh, Z. (2018). *TensorFlow for deep learning*. O'Reilly Media, Inc.

[Cao et al., 2020] Cao, H., Liu, H., Song, E., Hung, C.-C., Ma, G., Xu, X., Jin, R., and Lu, J. (2020). Dual-branch residual network for lung nodule segmentation. *Applied Soft Computing*, 86.

[Chollet, 2018] Chollet, F. (2018). *Deep learning with Python*. Manning Publications Co.

[Chunran et al., 2018] Chunran, Y., Yuanvuan, W., and Yi, G. (2018). *Automatic Detection and Segmentation of Lung Nodules on CT Images*. In *11th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics*, pages 1–6.

[CIA, 2022] CIA (2022). Lung image database consortium and image database resource initiative. Recuperado febrero de 2022, de: https://wiki.cancerimagingarchive.net/display/Public/LIDC-IDRI.

[Coulouris et al., 2001] Coulouris, G., Dellimore, J., and Kindberg, T. (2001). *Sistemas distribuidos: conceptos y diseño*. Pearson Education Inc.

[Cybenko, 1989] Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, 2(4):303–314.

[Dean et al., 2012] Dean, J., Corrado, G., Monga, R., Chen, K., Devin, M., Mao,

M., Ranzato, M. a., Senior, A., Tucker, P., Yang, K., Le, Q., and Ng, A. (2012). Large scale distributed deep networks. In Pereira, F., Burges, C. J. C., Bottou, L., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc.

[DeVita et al., 2019] DeVita, Vincent T., J., Lawrence, T. S., and Rosenberg, S. A. (2019). *DeVita, Hellman, and Rosenberg's Cancer: principles and practice of oncology*. Wolters Kluwe.

[Dougherty, 2009] Dougherty, G. (2009). *Digital image processing for medical applications*. Cambridge University Press.

[Drokin and Ericheva, 2021] Drokin, I. and Ericheva, E. (2021). End-to-end lung nodule detection framework with model-based feature projection block. In Lian, C., Cao, X., Rekik, I., Xu, X., and Yan, P., editors, *Machine Learning in Medical Imaging*, pages 91–100. Springer International Publishing.

[Dumoulin and Visin, 2016] Dumoulin, V. and Visin, F. (2016). A guide to convolution arithmetic for deep learning. *ArXiv*, abs/1603.07285.

[Fourcade and Khonsari, 2019] Fourcade, A. and Khonsari, R. H. (2019). Deep learning in medical image analysis: A third eye for doctors. *Journal of Stomatology, Oral and Maxillofacial Surgery*, 120(2019):279–288.

[Géron, 2017] Géron, A. (2017). *Hands-On Machine Learning with Scikit-Learn & Tensorflow*. O'Reilly Media, Inc.

[Goldie and Mirhoseini, 2020] Goldie, A. and Mirhoseini, A. (2020). Placement optimization with deep reinforcement learning. pages 3–7.

[Gonzalez and Woods, 2008] Gonzalez, R. C. and Woods, R. E. (2008). *Digital image processing*. Pearson Education Inc.

[Gould et al., 2007] Gould, M. K., Fletcher, J., Iannettoni, M. D., Lynch, W. R., Midthun, D. E., Naidich, D. P., and Ost, D. E. (2007). Evaluation of patients with pulmonary nodules: When is it lung cancer?: Accp evidence-based clinical practice guidelines (2nd edition). *Chest*, 132(3):108S–130S.

[Gu et al., 2020] Gu, J., Tian, Z., and Qi, Y. (2020). Pulmonary nodules detection based on deformable convolution. *IEEE Access*, 8:16302–16309.

[Gupta et al., 2017] Gupta, A., Thakur, H. K., Shrivastava, R., Kumar, P., and Nag, S. (2017). A big data analysis framework using apache spark and deep learning. In *2017 IEEE International Conference on Data Mining Workshops*, pages 9–16.

[Gurney et al., 1993] Gurney, J., Lyddon, D., and McKay, J. (1993). Determining the likelihood of malignancy in solitary pulmonary nodules with bayesian analysis. part ii. application. *Radiology*, 186(2):415–422.

[Hamidi and Borji, 2010] Hamidi, M. and Borji, A. (2010). Invariance analysis of modified c2 features: Case study-handwritten digit recognition. *Machine Vision and Applications*, 21:969–979.

[Havaei et al., 2017] Havaei, M., Davy, A., Warde-Farley, D., Biard, A., Courville, A., Bengio, Y., Pal, C., Jodoin, P.-M., and Larochelle, H. (2017). Brain tumor

segmentation with deep neural networks. *Medical Image Analysis*, 35:18–31.

[Haykin, 2009] Haykin, S. (2009). *Neural Networks and Learning Machines*. Prentice Hall.

[Hernández-Solis et al., 2021] Hernández-Solis, V., Téllez-Velázquez, A., Orantes-Molina, A., and Cruz-Barbosa, R. (2021). Lung-nodule segmentation using a convolutional neural network with the u-net architecture. In Roman-Rangel, E., Kuri-Morales, Á. F., Martínez-Trinidad, J. F., Carrasco-Ochoa, J. A., and Olvera-López, J. A., editors, *Pattern Recognition*, pages 335–344. Springer International Publishing.

[Hilera and Martínez Hernando, 1995] Hilera, J. and Martínez Hernando, V. (1995). *Redes neuronales artificiales : fundamentos, modelos y aplicaciones*. SERBIULA (sistema Librum 2.0).

[Karau et al., 2015] Karau, H., Konwinski, A., Wendell, P., and Zaharia, M. (2015). *Learning Spark*. O'Reilly Media, Inc.

[Keuper, 2016] Keuper, J. (2016). Distributed training of deep neuronal networks: Theoretical and practical limits of parallel scalability. *Proceedings of the Workshop on Machine Learning in High Performance Computing Environments*, abs/1609.06870:1–8.

[Krizhevsky et al., 2012] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60:84 – 90.

[LeCun et al., 1998a] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998a). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

[LeCun et al., 1998b] LeCun, Y., Cortes, C., and Burges, C. J. (1998b). The MNIST database of handwritten digits. Imagenes obtenidas de The MNIST database of handwritten digits, `http://yann.lecun.com/exdb/mnist/`.

[Litjens et al., 2017] Litjens, G., Kooi, T., Bejnordi, B. E., Setio, A. A. A., Ciompi, F., Ghafoorian, M., van der Laak, J. A., van Ginneken, B., and Sánchez, C. I. (2017). A survey on deep learning in medical image analysis. *Medical Image Analysis*, 42:60–88.

[Liu et al., 2020] Liu, B., Ding, Z., and Lv, C. (2020). Distributed training for multi-layer neural networks by consensus. *IEEE Transactions on Neural Networks and Learning Systems*, 31(5):1771–1778.

[Liu and Raj, 2013] Liu, B. and Raj, A. (2013). Discriminative random field segmentation oflung nodules in CT studies. *Computational and Mathematical Methods in Medicine*, 2013:1–9.

[Liu, 2004] Liu, M. L. (2004). *Computación distribuida. Fundamentos y aplicaciones.* Pearson Educación, S. A.

[Liu et al., 2016] Liu, Y., Jing, W., and Xu, L. (2016). Parallelizing backpropagation neural network using mapreduce and cascading model. *Computational Intelligence and Neuroscience*, 2016:1–11.

[Liu et al., 2017] Liu, Y., Xu, L., and Li, M. (2017). The parallelization of back propagation neural network in mapreduce and spark. *International Journal of Parallel Programming*, 45.

[Lu et al., 2015] Lu, L., Tan, Y., Schwartz, L. H., and Zhao, B. (2015). Hybrid detection of lung nodules on CT scan images. *Medical Physics*, 42(9):5042–5054.

[McNitt-Gray et al., 2007] McNitt-Gray, M. F., III, S. G. A., Meyer, C. R., Reeves, A. P., McLennan, G., Pais, R. C., Freymann, J., Brown, M. S., Engelmann, R. M., Bland, P. H., Laderach, G. E., Piker, C., Guo, J., Towfic, Z., Qing, D. P.-Y., Yankelevitz, D. F., Aberle, D. R., van Beek, E. J. R., MacMahon, H., Kazerooni, E. A., Croft, B. Y., and Clarke, L. P. (2007). The lung image database consortium (lidc) data collection process for nodule detection and annotation. *Academic Radiology*, 14(12):1464–1474.

[medical, 2019] medical, T. A. C. S. (2019). What is lung cancer? Recuperado febrero de 2021, de: https://www.cancer.org/cancer/lung-cancer/about/what-is.html.

[Milone et al., 2001] Milone, D. H., Azar, A. A., and Rufiner, L. H. (2001). *Desarrollo de una supercomputadora basada en un cluster de PCs*. Universidad Nacional de Entre Ríos.

[Momen et al., 2007] Momen, W., Joseph, G., Ranjit, G., Michael, G., and Douglas, M. (2007). Evidence for the treatment of patients with pulmonary nodules: When is it lung cancer? accp evidence-based clinical practice guidelines (2nd edition). *Chest*, 132(3):94S–107S.

[Mukherjee et al., 2017] Mukherjee, S., Huang, X., and Bhagalia, R. R. (2017). *Lung nodule segmentation using deep learned prior based graph cut*. In *14th IEEE International Symposium on Biomedical Imaging*, pages 1205–1208.

[Müller and Guido, 2017] Müller, A. C. and Guido, S. (2017). *Introduction to Machine Learning with Python*. O'Reilly Media, Inc.

[Ni et al., 2020] Ni, J., Wu, J., Tong, J., Chen, Z., and Zhao, J. (2020). GC-Net: Global context network for medical image segmentation. *Computer Methods and Programs in Biomedicine*, 190:1–10.

[Niederhuber et al., 2020] Niederhuber, J. E., Armitage, J. O., Kastan, M. B., Doroshow, J. H., and Tepper, J. E. (2020). *Abeloff's clinical oncology*. Elsevier, Inc.

[Nithila and Kumar, 2016] Nithila, E. E. and Kumar, S. S. (2016). Segmentation of lung nodule in CT data using active contour model and Fuzzy C-mean clustering. *Alexandria Engineering Journal*, 55(3):2583–2588.

[Pierfederici, 2016] Pierfederici, F. (2016). *Distributed Computing with Python*. Packt Publishing Ltd.

[Ratnasingam, 2019] Ratnasingam, S. (2019). Deep camera: A fully convolutional neural network for image signal processing. In *IEEE/CVF International Conference on Computer Vision Workshop*, pages 3868–3878.

[Ronneberger et al., 2015] Ronneberger, O., Fischer, P., and Brox, T. (2015). *U-*

Net: Convolutional Networks for Biomedical Image Segmentation. In *18th Medical Image Computing and Computer-Assisted Intervention*, pages 234–241.

[Rubin et al., 2015] Rubin, G. D., Roos, J. E., Tall, M., Harrawood, B., Bag, S., and Ly, D. L. (2015). Characterizing search, recognition, and decision in the detection of lung nodules on CT scans: Elucidation with eye tracking. *Radiology*, 274(1):276–286.

[Rudolph et al., 2019] Rudolph, R., Herzog, K., Toepfer, R., and Steinhage, V. (2019). Efficient identification, localization and quantification of grapevine inflorescences and flowers in unprepared field images using fully convolutional networks. *Vitis -Geilweilerhof-*, 58:95–104.

[SALUD/SSA, 2018] SALUD/SSA (2018). Cáncer de pulmón en méxico. Recuperado febrero de 2022, de: https://www.gob.mx/salud/prensa/145-cada-ano-mueren-cerca-de-ocho-mil-mexicanos-por-cancer-de-pulmon?idiom=es.

[Savic et al., 2021] Savic, M., Ma, Y., Ramponi, G., Du, W., and Peng, Y. (2021). Lung nodule segmentation with a region-based fast marching method. *Sensors*, 21.

[Siegelman et al., 1980] Siegelman, S., Zerhouni, E., Leo, F., Khouri, N., and Stitik, F. (1980). CT of the solitary pulmonary nodule. *American Journal of Roentgenology*, 135(1):1–13.

[Suji et al., 2020] Suji, R. J., Bhadouria, S. S., Dhar, J., and Godfrey, W. W. (2020). Optical flow methods for lung nodule segmentation on LIDC-IDRI images. *Journal of Digital Imaging*, 33:1306–1324.

[Sultana et al., 2020] Sultana, F., Sufian, A., and Dutta, P. (2020). Evolution of image segmentation using deep convolutional neural network: A survey. *Knowledge-Based Systems*, 201-202:106062.

[Sze et al., 2017] Sze, V., Chen, Y.-H., Emer, J., Suleiman, A., and Zhang, Z. (2017). Hardware for machine learning: Challenges and opportunities. *2017 IEEE Custom Integrated Circuits Conference*, pages 1–8.

[Tanenbaum and Steen, 2008] Tanenbaum, A. S. and Steen, M. V. (2008). *Sistemas distribuidos: principios y paradigmas*. Pearson Education Inc.

[Travis, 2011] Travis, W. D. (2011). Pathology of lung cancer. *Clinics in Chest Medicine*, 32(4):669–692.

[Wang et al., 2016] Wang, Q., Zhao, J., Gong, D., Shen, Y., Li, M., and Lei, Y. (2016). Parallelizing convolutional neural networks for action event recognition in surveillance videos. *International Journal of Parallel Programming*, 45(4):734–759.

[Wang et al., 2017a] Wang, S., Zhou, M., Gevaert, O., Tang, Z., Dong, D., Liu, Z., and Tian, J. (2017a). A multi-view deep convolutional neural networks for lung nodule segmentation. In *39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 1752–1755, Jeju, Korea (South).

[Wang et al., 2017b] Wang, S., Zhou, M., Liu, Z., Liu, Z., Gu, D., Zang, Y., Dong,

D., Gevaert, O., and Tian, J. (2017b). Central focused convolutional neural networks: Developing a data-driven model for lung nodule segmentation. *Medical Image Analysis*, 40:172–183.

[WHO, 2022] WHO (2022). Latest global cancer data. Recuperado febrero de 2022, de: https://www.who.int/news-room/fact-sheets/detail/cancer.

[Wu et al., 2018] Wu, B., Zhou, Z., Wang, J., and Wang, Y. (2018). Joint learning for pulmonary nodule segmentation, attributes and malignancy prediction. In *IEEE 15th International Symposium on Biomedical Imaging*, pages 1109–1113.

[Wu and Qian, 2019] Wu, J. and Qian, T. (2019). A survey of pulmonary nodule detection, segmentation and classification in computed tomography with deep learning techniques. *Journal of Medical Artificial Intelligence*, 2(8):1–12.

[Xu et al., 2008] Xu, D. M., van Klaveren, R. J., de Bock, G. H., Leusveld, A., Zhao, Y., Wang, Y., Vliegenthart, R., de Koning, H. J., Scholten, E. T., Verschakelen, J., Prokop, M., and Oudkerk, M. (2008). Limited value of shape, margin and CT density in the discrimination between benign and malignant screen detected solid pulmonary nodules of the nelson trial. *European Journal of Radiology*, 68:347–352.

[Young et al., 2007] Young, K. H., Mog, S. Y., Soo, L. K., Joungho, H., A, Y. C., and Kyung, K. Y. (2007). Persistent pulmonary nodular ground-glass opacity at thin-section CT: Histopathologic comparisons. *Radiology*, 245(1):267–275.

[Zuberek, 2015] Zuberek, W. (2015). Analysis of the speedup of distributed applications. *Radiology*, 274(1):276–286.

Anexos

Anexo A

Algoritmo de retropropagación

El algoritmo de retropropagación está diseñado para entrenar redes con dos o más capas de neuronas conectadas de forma que, las salidas de una capa se conviertan en las entradas de la capa siguiente. Además, las funciones de activación de las neuronas deben ser continuas para permitir el uso de la regla delta generalizada y obtener los errores en cada iteración.

A.1. Algoritmo de retropropagación secuencial

El algoritmo clásico de retropropagación utiliza el descenso del gradiente estocástico para actualizar los pesos del modelo (red neuronal), minimizando la función de pérdida. Para hacerlo, este algoritmo usa los gradientes de la función de pérdida con respecto a los pesos sinápticos. Este algoritmo se muestra a continuación:

A.2. Algoritmo de retropropagación distribuido

El algoritmo clásico de retropropagación fue adaptado de [Keuper, 2016] para ser utilizado en un clúster computacional utilizando cómputo distribuido. Esta adaptación ha sido llamado el algoritmo de retropropagación distribuido y es mostrado a continuación.

Algoritmo 1: Retropropagación con gradiente descendente

$X \leftarrow$ Conjunto de muestras de entrenamiento
$Y \leftarrow$ Conjunto de etiquetas de entrenamiento
Inicializar η y todos los pesos W en la red.
$out = feedforward(X, W)$; $L_{tot} = loss(X, Y, W)$
mientras $L_{tot} \geq \epsilon$ *(error mínimo u otro criterio)* **hacer**
    $\forall w \in W : \Delta w = -\frac{\partial L_{tot}}{\partial w}$;
    $w_{nuevo} \leftarrow w_{anterior} + \eta \Delta w$;
    Calcular *out* y L_{tot};
fin

Algoritmo 2: Retropropagación distribuido

Generar modelo M;
Inicializar η y W en cada partición de los nodos trabajadores;
para cada $epoca \in epocas$ **hacer**
    $\Delta W_p \leftarrow \{\}$;
    para cada $particion \in Dataset$ **hacer**
        $\Delta w_p \leftarrow \Delta w_p \cup Retropropagacion(particion, W)$;
    fin
    $\Delta w \leftarrow Reduce_{prom}(\Delta w_p)$;
    $W \leftarrow W + \Delta w$;
fin
Función `Retropropagacion`(P, W_t)**:**
    $W_{t\_nuevo} \leftarrow W_t$
    Crear submodelo M_t a partir de η_t y W_{t_nuevo};
    para cada $minilote \in P$ **hacer**
        $\forall w_{t\_nuevo} \in W_{t\_nuevo} : \Delta w_{t\_nuevo} = -\frac{\partial L_{tot}}{\partial w_{t\_nuevo}}$;
        $w_{t\_nuevo} \leftarrow w_{t\_anterior} + \eta_t \Delta w_{t\_nuevo}$;
    fin
    $\Delta w_t \leftarrow W_t - W_{t\_nuevo}$;
    return Δw_t;

Anexo B

Manual de usuario

En este anexo se indica cómo usar la biblioteca desarrollada para realizar el entrenamiento distribuido de redes neuronales artificiales, las cuales se implementaron en el lenguaje de programación Python versión 3.6.9. La estructura general del directorio raíz del proyecto se muestra en la Figura B.1. Esta estructura de directorio debe estar replicada en cada nodo, tanto en el maestro como en los trabajadores.

La descripción del contenido del directorio raíz del proyecto es la siguiente:

- basesDatos. Contiene las bases de datos (MNIST y LIDC-IDRI) utilizadas para el entrenamiento de las redes neuronales.
- src. Contiene los archivos necesarios para el entrenamiento y predicción de las redes neuronales.
    - models.py: Contiene los modelos de la FCN AlexNet y U-Net modificada.
    - generators.py: Contiene los generadores de lotes de imágenes para alimentar los modelos.
    - utils.py : Contiene funciones para la lectura y escritura de imágenes y otros tipos de archivos.
    - loss.py: Contiene las funciones de pérdida para los modelos.
    - metrics.py: Contiene las medidas de rendimiento para evaluar los mo-

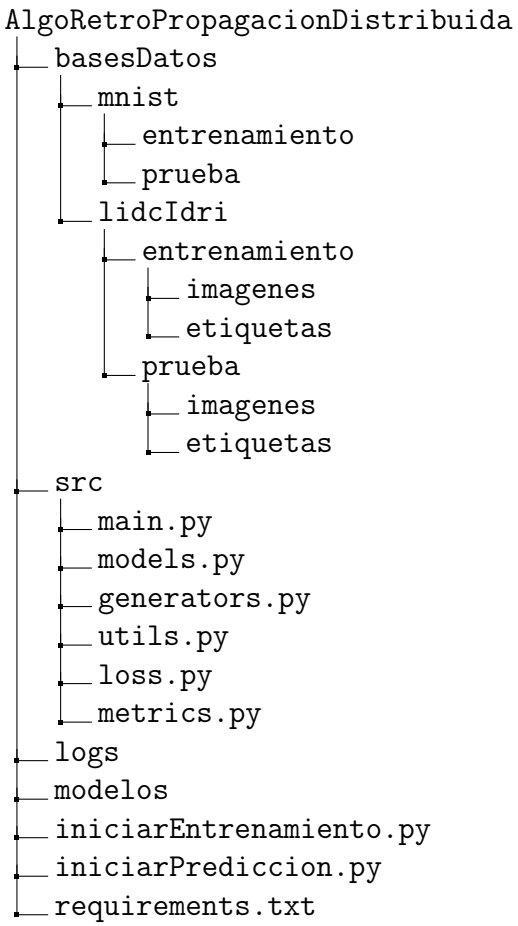

Figura B.1: Estructura del directorio raíz del proyecto.

delos.

- logs. Contiene los logs generados durante el entrenamiento o predicción.
- modelos. Contiene los modelos entrenados guardados en formato H5.
- iniciarEntrenamiento.py. Inicia el entrenamiento de la red neuronal especificada.
- iniciarPrediccion.py. Inicia la prediccion de la red neuronal especificada.
- requirements.txt. Contiene el listado de las dependencias del proyecto.

B.1. Instalación de dependencias

Para la instalación de las dependencias requeridas y asegurar el correcto funcionamiento de la biblioteca desarrollada, se hace uso del Instalador de Paquetes para Python (PIP, por sus siglas en inglés), el cual permite instalar rápidamente bibliotecas Python como pyspark, tensorflow, numpy, etc. Para este propósito, el archivo `requeriments.txt` contiene un listado de todas las bibliotecas que deben ser instaladas. Para iniciar el proceso de instalación, basta con ejecutar los comandos mostrados en el Listado B.1 dentro del directorio raíz del proyecto o indicar la ruta completa al archivo de requerimientos.

Listado B.1: Comandos para instalar PIP y las dependencias del proyecto.

```
$ sudo apt install python3-pip
$ sudo pip3 install -r requirements.txt
```

El primer comando instala la herramienta PIP para ser usada con Python3, mientras que el segundo instala todas las bibliotecas que son dependencias del proyecto creado en este trabajo de tesis.

B.2. Ejemplo práctico

Para iniciar la etapa de entrenamiento (o la etapa de predicción) de una red neuronal artificial, utilizando el algoritmo de retropropagación distribuido, basta con ejecutar el script principal, ya sea `iniciarEntrenamiento.sh` o `iniciarPrediccion.sh`, pasándole los argumentos correspondientes, de acuerdo con las opciones siguientes:

- -p. Número de particiones en las cuales se dividirá el conjunto de datos original.
- -e. Número de épocas para terminar la etapa de entrenamiento.
- -x. Número de ejecutores de Spark a crear.
- -c. Número de núcleos a utilizar.
- -r. Red neuronal a utilizar (`alexnet` para AlexNet y `unet` para la red U-Net).
- -s. Ruta para guardar el modelo entrenado.

- -m. Ruta para abrir modelo entrenado (en el caso de predicción).

Un ejemplo de ejecución de la etapa de entrenamiento utilizando 9 particiones, 100 épocas, 3 ejecutores, 4 núcleos por nodo, utilizando la U-Net y guardando el modelo como `unet_entrenada.h5`, se tiene que lanzar el comando mostrado en el Listado B.2:

Listado B.2: Comando para iniciar el entrenamiento de la U-Net.

```
$ bash ./iniciarEntrenamiento.sh -p 9 -e 100 -x 3 -c 4
   -r unet -s unet_entrenada.h5
```

Para iniciar la etapa de predicción el procedimiento es idéntico, solo que el script a ejecutar es distinto. Por ejemplo, para utilizar 9 particiones, 3 ejecutores, 4 núcleos por nodo, la red U-Net y el modelo previamente guardado, es necesario lanzar el comando mostrado en el Listado B.3:

Listado B.3: Comando para iniciar la predicción de la U-Net.

```
$ bash ./iniciarPrediccion.sh -p 9 -x 3 -c 4 -r unet -m
   ./modelos/unet_entrenada.h5
```

Una vez iniciada alguna de las etapas, se puede monitorear el proceso mediante la salida a terminal. Un ejemplo se muestra en el Listado B.4.

Listado B.4: Salida en terminal del entrenamiento distribuido.

```
Epoch: 49
843/843 [================] - 52s 10ms/step -  loss:
   0.0720 - acc: 0.9894 - my_io_u: 0.4947 - auc: 0.6224
   - DSC: 0.62250555
```