UNIVERSIDAD AUTÓNOMA DE CHIHUAHUA

FACULTAD DE INGENIERÍA

SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

MODELO DE DETECCIÓN DE LESIONES EN TOMOGRAFÍAS COMPUTARIZADAS BASADO EN REDES DE APRENDIZAJE PROFUNDO

POR:

ING. KATHIA VICTORIA RASCÓN CERVANTES

TESIS PRESENTADA COMO REQUISITO PARA OBTENER EL GRADO DE

MAESTRO EN INGENIERÍA EN COMPUTACIÓN

CHIHUAHUA, CHIH., MÉXICO OCTUBRE 2021

Modelo de detección de lesiones en tomografías computarizada basado en redes de aprendizaje profundo. Tesis presentada por Kathia Victoria Rascón Cervantes como requisito parcial para obtener el grado de maestro en ingeniería en computación, ha sido aprobado y aceptado por:

M.I. Javier González Cantú
Director de la Facultad de Ingeniería

Dr. Alejandro Villalobos Aragón
Secretario de Investigación y Posgrado

Karina Requena Yáñez
Coordinador(a) Académico

Graciela Ramirez A.
Dra. Graciela María de Jesús Ramírez Alonso
Director(a) de Tesis

Octubre 2021

Fecha

COMITÉ
Dra. Graciela María de Jesús Ramírez Alonso
Dr. Manuel Montes y Gómez
Dr. Luis Carlos González Gurrola
Dr. Jesús Roberto López Santillán

UNIVERSIDAD AUTÓNOMA DE
CHIHUAHUA

28 de octubre de 2021.

I.M. Kathia Victoria Rascón Cervantes
Presente.-

En atención a su solicitud relativa al trabajo de tesis para obtener el grado de Maestra en Ingeniería en Computación, nos es grato transcribirle el tema aprobado por esta Dirección, propuesto y dirigido por la directora **Dra. Graciela María de Jesús Ramírez Alonso** para que lo desarrolle como tesis, con el título: **"MODELO DE DETECCIÓN DE LESIONES EN TOMOGRAFÍAS COMPUTARIZADAS BASADO EN REDES DE APRENDIZAJE PROFUNDO".**

Índice de Contenido

FACULTAD DE INGENIERÍA
Circuito No.1, Campus Universitario 2
Chihuahua, Chih., México. C.P. 31125
Tel. (614) 442-95-00
www.fing.uach.mx

UNIVERSIDAD AUTÓNOMA DE
CHIHUAHUA

CAPÍTULO 6: Experimentos

6.1. Configuración experimental
6.2 Resultados
6.3 Análisis de resultados

Capítulo 7: Discusión y conclusiones

7.1 Discusión y conclusiones
7.2 Trabajo futuro

Bibliografía

A T E N T A M E N T E
"Naturam subiecit aliis"

EL DIRECTOR

M.I. JAVIER GONZÁLEZ CANTÚ

FACULTAD DE INGENIERÍA U.A.CH.

DIRECCIÓN

EL SECRETARIO DE INVESTIGACIÓN Y POSGRADO

DR. ALEJANDRO VILLALOBOS ARAGÓN

FACULTAD DE INGENIERÍA
Circuito No.1, Campus Universitario 2
Chihuahua, Chih., México. C.P. 31125
Tel. (614) 442-95-00
www.fing.uach.mx

Per Alba, la mia futura moglie,

l'amore della mia vita.

Agradecimientos

A CONACYT, por la beca que se me otorgó e hizo posible mis estudios de posgrado.

A la Dra. Graciela María de Jesús Ramírez Alonso, por ser mi directora de tesis durante la maestría y anteriormente en la licenciatura. Y también por apoyarme constantemente, no sólo académicamente.

Al Dr. Manuel Montes y Gómez, por ser mi co-director, por estar al pendiente de los progresos de mi investigación y siempre aportar ideas nuevas.

A la Dra. Vania Carolina Alvarez Olivas, por su constante apoyo en la preparación precipitada de mi protocolo.

A mi profesores de la maestría, por los conocimientos que me ayudaron a adquirir, y por las nuevas áreas que me permitieron descubrir.

A la M.S.I. Karina Requena Yáñez, actualmente coordinadora de la maestría, por su apoyo durante la maestría.

Resumen

Las lesiones o daños internos en el cuerpo, causados por accidentes, caídas, golpes, quemaduras, u otras causas, representan un 9 % de la mortalidad mundial. Las técnicas de diagnóstico asistido por computadora, permiten la identificación y el análisis de lesiones, beneficiando el diagnóstico en etapas tempranas. La mayoría de estas técnicas se basan en modelos de detección de objetos, que permiten, en este caso, el estudio de tomografías computarizadas. En este trabajo se analiza e implementa el modelo *EfficientDet*, para la tarea de detección de lesiones en imágenes médicas, utilizando la base de datos *DeepLesion*. La cual fue creada con el objetivo de generar un detector universal de lesiones; un problema poco desarrollado hasta ahora.

Una aportación importante en este trabajo de tesis, es el estudio de dos técnicas de aumento de datos utilizando imágenes médicas para determinar la relevancia del contexto de la lesión en dicho proceso. También se busca el mejor conjunto de muestras para generar el aumento de datos. Se parte de los conocimientos previos sobre detección de objetos en imágenes naturales.

En el modelo *EfficientDet* se analizan e implementan estratégicamente técnicas de atención en distintas capas del modelo: fusión de contexto 3D y módulos de atención de canal y espacial. Los resultados muestran que se logra una sensitividad de 73 % con un umbral de 7 falsos positivos por imagen (FPs), empleando el modelo *EfficientDet-D0*.

Posteriormente, se interpreta el impacto que tienen los distintos niveles de *EfficientDet-D0* a *D6* en la detección de lesiones. Esto se realiza de manera cuantitativa mediante la métrica de sensitividad y, además, se visualizan algunas de las activaciones de sus *feature-maps*. Esto último, para realizar un análisis comparativo entre niveles de *EfficientDet*, tipo y tamaño de lesiones, e imágenes naturales *versus* imágenes médicas.

Índice general

Índice de tablas

Índice de figuras

Capítulo 1

Introducción

1.1. Motivación

Las lesiones constituyen una amenaza para la salud mundial, ya que representan un 9 % de la mortalidad, lo que equivale a más de cinco millones de muertes cada año. Las estimaciones de la Organización Mundial de la Salud (OMS) señalan que por cada persona que muere a causa de lesiones, en promedio, 30 son hospitalizadas y 300 atendidas en servicios de urgencias. En la región de las Américas, aproximadamente 300 mil personas mueren a causa de lesiones, lo que las convierte en la cuarta causa de muerte [8].

Las tomografías computarizadas (CTs) de rayos X, son un tipo de imágenes médicas de uso común, que muestra detalles finos dentro del cuerpo humano. Estas imágenes pueden mostrar lesiones en huesos, abdomen, mediastino, hígado, pulmón, riñón, tejidos blandos[1] y pelvis [9]. En las CTs los radiólogos buscan encontrar lesiones, caracterizarlas y medirlas, para luego describirlas en un informe radiológico. Esta tarea suele ser tediosa y consume tiempo, especialmente detectar lesiones pequeñas, es uno de los procesos más laboriosos para los médicos [10], [11].

En los últimos años, las técnicas de diagnóstico asistido por computadora (CAD), se han convertido

[1] lesiones diversas en la pared del cuerpo, músculos, piel, grasa, extremidades y cuello

en un campo de investigación importante en el procesamiento de imágenes médicas [12]. El CAD permite la identificación y el análisis de lesiones en la práctica clínica, lo que beneficia al diagnostico de enfermedades en etapa temprana [5]. Estas técnicas emplean modelos de detección de objetos, cuya base está compuesta por el aprendizaje profundo y la visión por computadora.

1.2. Problemática

La mayoría de los trabajos existentes sobre detección de lesiones se centran en un tipo específico de lesión u órgano; una razón para esto puede ser la escasez de bases de datos de lesiones universales. Tan solo hace un par de años, se hizo pública la base de datos *DeepLesion*, esta es la base de datos de imágenes médicas más grande reportada actualmente. Cuenta con tomografías de 9 regiones del cuerpo, lo que permite trabajar en un detector universal, el cual resultaría de mayor utilidad que un detector centrado en una única región u órgano. Las tomografías que conforman a *DeepLesion* se recabaron durante casi dos décadas [13].

Recientemente, han surgido investigaciones basadas en *DeepLesion* con el fin de proponer un detector universal de lesiones [3], [5] - [7]. En dichas investigaciones se han tenido problemas en la detección de lesiones pequeñas, lo cual es común en cualquier detector. Sin embargo, es importante atacarlo ya que el 61 % de las lesiones en *DeepLesion* son pequeñas. La cantidad de estudios por región no está equilibrada, esto genera problemas al detectar lesiones en tomografías de hueso y tejidos blandas, al tener pocas muestras. Por otro lado, las lesiones de abdomen y pelvis son difíciles de distinguir de su entorno.

Estos problemas con determinadas lesiones se puede atacar con una mayor cantidad de muestras. Generar nuevos datos etiquetados no es factible ya que es un proceso costoso, tardado y requiere cierta expertiz. Por otro lado, se puede optar por emplear estrategias de aumento de datos, en propuestas previas se han propuesto los siguientes métodos: *flipping, cropping, resize, random resizing, random translation y 3D augmentation* [4], [6], [7].

Los modelos que se suelen implementar en la detección de lesiones en imágenes emplean un *backbone* como VGG-16 [14], ResNet [15] o DenseNet [16], pre-entrenados con ImageNet [17], y en su mayoría usan una *Feature Pyramid Network* (FPN) o el modelo de *Faster R-CNN* [18] como red de características. Para los BBox, se suelen proponer entre 3-6 *anchor scales* y 3-5 *anchor ratios*. El entrenamiento lo realizan por no más de 15 épocas, empleando el optimizador *stochastic gradient descent* (SGD), y reportan sus resultados para distintos falsos positivos por imagen (FPs) [2] - [7].

La mayoría reportan problemas en la detección de lesiones pequeñas, esto puede estar ligado al uso de *Faster R-CNN*. Uno de los problemas con este detector es que al ser de tipo *two-stage*, no puede ser optimizado *end-to-end*. Los detectores de tipo *two-stage* emplean un generador de propuestas para generar un conjunto disperso de propuestas, y extraer características de cada una, seguido por un clasificador de región. Los detectores de tipo *one-stage* realizan predicciones directamente en cada ubicación de los mapas de características. Por lo que estos ultimos son significativamente más eficientes en el tiempo, y tienen mayor aplicabilidad en la detección de objetos en tiempo real [19].

1.3. Objetivos

El objetivo general de este trabajo de investigación es:

Implementar un modelo computacional basado en aprendizaje profundo, que sea capaz de detectar lesiones en tomografías de múltiples regiones del cuerpo.

Los objetivos específicos que dan pie a la presente investigación son los siguientes:

1. Determinar si incluso al emplear una base de datos extensa como *DeepLesion*, es necesario realizar aumento de datos.

2. Precisar la relevancia del contexto inmediato a la lesión en la estrategia de aumento de datos.

3. Establecer si es mejor el aumento de datos indiscriminado o el centrado en el tipo de lesiones más difíciles de detectar.

4. Deducir si el estado del arte en imágenes naturales, tiene el mismo efecto al trabajar con imágenes médicas.

5. Interpretar el impacto que tienen distintos niveles de profundidad de *EfficientDet* en la detección de lesiones.

6. Identificar estrategias de atención a incorporar en *EfficientDet*, que permitan mejorar la detección de lesiones.

1.4. Metodología

En esta investigación se trabaja con la base de datos *DeepLesion*, con la finalidad de cumplir con los primeros tres objetivos específicos, se aplican distintas configuraciones de aumento de datos. Estas se basan en *horizontal flip* y *cut-and-paste*, y el modelo base para ello es *EfficientDet-D0*.

Para estudiar el cuarto objetivo se compara cuantitativamente y cualitativamente el comportamiento de los resultados obtenidos, al emplear imágenes médicas e imágenes naturales. En el quinto objetivo se emplean los niveles de *EfficientDet*-D0 a D6, se obtienen la métrica de sensitividad empleando un umbral de falsos positivos por imagen. Además, se visualizan las activaciones de las primeras capas convolucionales de los distintos niveles para los distintos tipos de lesiones.

Finalmente, se realiza fusión de contexto con los cortes vecinos al corte que contiene la lesión, y se incorporan módulos de atención espaciales y de canal. Estos últimos se colocan entre el *backbone* y la red de características BiFPN, y en la red de clasificación.

1.5. Organización del documento

La presente tesis está organizada como se describe a continuación. El capítulo 2 está compuesto por el marco conceptual, aquí se presentan conceptos importantes para entender los modelos de detección de objetos basados en aprendizaje profundo. En el capítulo 3 se encuentra el marco teórico, en él se describen varios detectores de objetos, especialmente *EfficientDet*. El capítulo 4 presenta el trabajo relacionado. En el capítulo 5, se describe la metodología seguida para el problema de detección universal de lesiones. En el capítulo 6, se presentan los resultados de sensitividad para las distintas implementaciones, y un análisis cualitativos y cuantitativo. Finalmente, en el capítulo 7, se presenta la discusión y conclusiones.

Capítulo 2

Marco Conceptual

En este capítulo se definirán conceptos fundamentales como: visión por computadora, detección de objetos, aprendizaje máquina, aprendizaje profundo, redes neuronales convolucionales y mecanismos de atención. Posteriormente, en la sección 2.1 se detallan elementos de una red neuronal convolucional. Y en la sección 2.3 se explican elementos básicos para la detección de objetos.

Detección de objetos

La detección de objetos (DO) basada en procesamiento de imágenes extrae características de éstas, y posteriormente obtiene y analiza la información de los objetos (categoría, posición y dirección).No solo reconoce la categoría de los objetos, además predice la localización de cada uno por medio de un recuadro delimitador o *bounding box* (BBox) [20]. Algunas de sus aplicaciones más estudiadas son detección de rostros y detección de peatones [20]. Sin embargo, existen muchas más áreas de aplicación tales como: medicina, seguridad y vigilancia, mercadotecnia, control de tráfico, aeropuertos, interacción humano-computadora, control y visión de robots [21], [22]. Algunos de los detectores que han conformado el estado del arte en los últimos años son: YOLO [23], SSD [24], YOLOv3 [25], y actualmente *EfficientDet* [1].

En la figura 2.1 se muestra un diagrama de venn, que muestra como la intersección del aprendizaje profundo y la visión por computadora permite realizar la detección de objetos.

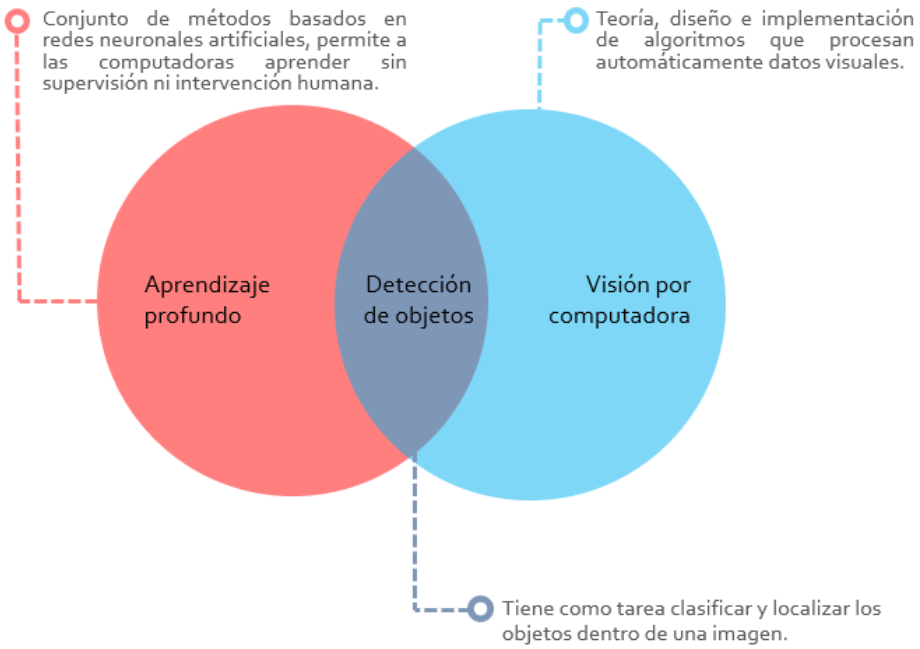

Figura 2.1: La detección de objetos es la intersección entre dos áreas: visión por computadora y aprendizaje profundo.

La detección de rostros junto con la detección de peatones son de las aplicaciones más estudiadas [20], [26]. Sin embargo, existen muchas más áreas de aplicación tales como: seguridad y vigilancia, medicina, mercadotecnia, control de tráfico, aeropuertos, interacción humano-computadora, control y visión de robots [21], [22].

Visión por computadora

La visión por computadora es una disciplina científica que se enfoca al desarrollo de algoritmos para que las computadoras obtengan una comprensión de alto nivel de su entorno a partir del análisis de imágenes o video. Se trata de la teoría, el diseño y la implementación de algoritmos que pueden procesar automáticamente datos visuales para reconocer objetos, rastrear y recuperar su forma y diseño espacial. En este campo existen varios problemas fundamentales de reconocimiento visual como: clasificación de

imágenes, detección de objetos, segmentación de instancias y segmentación semántica [27], [19].

El objetivo principal de la visión por computadora es entender las escenas visuales [26]. Hoy en día, este campo se emplea en una amplia variedad de aplicaciones de la vida real, que incluyen: reconocimiento óptico de caracteres (OCR), inspección de máquinas, reconocimiento de objetos para líneas de pago automatizadas, construcción de modelado 3D, imágenes médicas, seguridad automotriz, captura de movimiento (mocap), vigilancia, reconocimiento de huellas dactilares, biometría, entre otros [28].

Las estrategias más recientes que se han propuesto para la DO, se basan completamente en redes neuronales convolucionales. Específicamente, en un detector de objetos que ha conformado el estado del arte como *Faster R-CNN* [18], YOLOv1 [23], Yolov3 [25] y SSD [24]. En la actualidad, el detector de objetos del estado del arte es *EfficentDet* [1].

Aprendizaje máquina

El aprendizaje máquina o *machine learning* (ML) es un subcampo dentro de la inteligencia artificial (AI), en donde a su vez, se encuentra el área de aprendizaje profundo o *deep learning* (DL) [29]. El DL contiene una gran cantidad de áreas de aplicación tales como: procesamiento de voz y audio, procesamiento de lenguaje natural, robótica, bioinformática, química, videojuegos, publicidad, finanzas, motores de búsqueda, visión por computadora, entre otras [30].

El aprendizaje máquina busca que las computadoras modifiquen o adapten sus acciones, para que sean más precisas. La precisión se mide por cuán bien las acciones elegidas reflejan las correctas. Los algoritmos de ML se dividen en: supervisados, no supervisados, reforzados y evolutivos [31].

Transferencia de aprendizaje

La transferencia de aprendizaje o *transfer learning*, se define como la capacidad de usar el conocimiento o las habilidades aprendidas, tomadas de un conjunto original de tareas en un dominio original. Esto para resolver problemas en un conjunto diferente de tareas y/o en un dominio diferente. Como no hay dos situaciones exactamente iguales, todo aprendizaje implica la transferencia de algún tipo. Sin embargo, la cantidad de transferencia que se logra depende de muchos factores, el principal es el grado de similitud entre los conjuntos de tareas, y dominios originales y nuevos [32].

Aprendizaje profundo

El aprendizaje profundo o *deep learning* (DL) es un conjunto de métodos basados en redes neuronales artificiales, que permiten a las computadoras aprender de los datos sin supervisión ni intervención humana [33]. El aprendizaje profundo, permite que las redes neuronales aprendan múltiples niveles de abstracción. Permitiendo así mejoras en el estado de arte, en áreas como reconocimiento de voz y visión por computadora [34].

En el DL se emplea un algoritmo de retropropagación o *backpropagation*, para encontrar una estructura compleja a partir de una base de datos extensa. Durante el entrenamiento, este algoritmo le indica a la red como actualizar sus parámetros internos. De acuerdo con LeCun, ”las redes convolucionales profundas han producido avances en el procesamiento de imágenes, video, voz y audio, mientras que las redes recurrentes lo han hecho en datos secuenciales como el texto y la voz”[34].

Mecanismos de atención

Una propiedad importante del sistema visual humano, es que no se presta atención a toda la escena a la vez. En lugar de ello, los seres humanos explotan una secuencia de destellos parciales y se centran se-

lectivamente en partes sobresalientes. Por lo que la atención juega un papel importante en la percepción humana [35].

Los mecanismos de atención han demostrado recientemente un gran potencial para mejorar el rendimiento de las redes neuronales convolucionales. Estos mecanismos, permiten a la red realizar una recalibración de características. Con lo cual pueden aprender a usar información global para enfatizar selectivamente características informativas y suprimir las menos útiles. Las propuestas varían entre prestar atención sólo al aspecto espacial, sólo a los canales o una combinación de ambos aspectos [35],[36], [37].

2.1. Redes Neuronales

En esta sección, se describen las redes FPN y RPN, los pasos del entrenamiento de una red, qué es la convolución, las capas de convolución y *pooling*, y funciones de pérdida y activación. Y se abordarán estrategias para mejorar el rendimiento de una red, como transferencia de aprendizaje y aumento de datos.

2.1.1. Convolutional neural network

Una red neuronal convolucional o *convolutional neural network* (CNN), pertenece al aprendizaje profundo, esta red emplea un mecanismo de aprendizaje supervisado, que se conoce como el algoritmo de descenso de gradiente estocástico (SGD) [38]. El esquema de SGD ha mostrado buenos resultados hasta la fecha, sin embargo, se han creado nuevas variantes de este algoritmo como: descenso de gradiente por *mini-batch*, descenso de gradiente con *momentum*, Nesterov *momentum*, AdaGrad, RMSprop y Adam [39].

El nombre CNN hace alusión a que estas redes emplean una operación matemática llamada convolución, que es un tipo especial de operación lineal. Estas redes son usadas comúnmente para el reconocimiento de objetos, ya que permiten la extracción de características dentro de sus capas. El interés de la CNN es

clasificar directamente la entrada en bruto, dado que no siempre es posible saber el tipo de características para extraer, es mejor dejar que la red extraiga las características más discriminantes mediante la construcción de características de alto nivel, durante el proceso de propagación [40].

La estructura general de una red neuronal convolucional es la siguiente [41]:

- La primer parte es responsable de la extracción de características, contiene una capa de entrada, capas convolucionales y capas de agrupamiento. Las capas convolucionales y las de agrupación se apilan capa por capa en la red, son extractores de características.
- La segunda parte se encarga de la clasificación y contiene capas totalmente conectadas. Reciben las características obtenidas por la última capa de agrupamiento como entrada, y realizan tareas de clasificación.

2.1.2. Redes en la detección de objetos

Las pirámides de características o *Feature Pyramid Network* (FPN), se han convertido en un componente básico, en los sistemas de reconocimiento para detectar objetos a diferentes escalas. FPN se compone de una ruta de abajo hacia arriba (*bottom-up)* y una de arriba hacia abajo (*top-down*). La ruta *bottom-up* es la red convolucional habitual para la extracción de características. Donde se reduce la resolución espacial y se aumenta la profundidad, lo cual aumenta el valor semántico de cada capa. La ruta *top-dpwn* se emplea para construir capas de mayor resolución a partir de una capa rica en semántica [42].

La red de regiones de propuesta o *Region Proposal Networks* (RPN), es una red totalmente convolucional. RPN toma como entrada una imagen de cualquier tamaño, y arroja como salida un conjunto de propuestas rectangulares de donde podría haber un objeto [18].

2.1.3. Entrenamiento de una red neuronal

Los pasos para entrenar una red neuronal, se pueden dividir las siguientes secciones principales [33]:

1. **Inicialización de pesos:**

   Los pesos se pueden inicializar de manera aleatoria, pero también se pueden implementar estrategias de *transfer learning*, las cuales han mostrado experimentalmente beneficiar el rendimiento de las redes.

2. **Propagación hacia adelante o *forward propagation*:**

   Se calcula la salida de la red para una muestra o un conjunto de muestras (*batch*). En este proceso se ven involucradas operaciones de convolución, entre *kernels* y los mapas de características. Las salidas de cada capa oculta dependerán de parámetros asociados como: *stride*, *pooling* y el número y dimensión de los *kernels*. En cada capa se tendrá asociada una función de activación, las más comunes son: *sigmoid*, *ReLU* y *Softmax*.

3. **Cálculo del error:**

   Se obtiene el error entre la predicción de la red y el *target*. Comúnmente se emplean funciones simples como el error cuadrático medio, sin embargo, se podría usar otra, siempre y cuando cumpla con ser diferenciable.

4. **Propagación hacia atrás o *backpropagation*:** Se emplea un algoritmo de optimización basado en el cálculo del gradiente. Este puede ser *stochastic gradient descent* (SGD), SGD por *mini-batch*, AdaGrad, RMSprop, o Adam.

5. **Actualización de parámetros:**

   Se actualizan los valores de los pesos y *bias* de la red neuronal.

2.1.4. Convolución

La convolución[1] es un tipo especial de operación lineal. De acuerdo con [30] la convolución para el caso continuo y discreto se definen como:

Para el caso continuo:

$$s(t) = \int x(a)w(t-a)da \tag{2.1}$$

Para el caso discreto:

$$s(t) = \sum_{a=-\infty}^{\infty} x(a)w(t-a) \tag{2.2}$$

Particularmente para el caso de dos dimensiones, que es el requerido cuando se trabaja con imágenes, la convolución se define como:

$$S(i,j) = (K * I)(i,j) = \sum_{m}\sum_{n} I(i-m, j-n)K(m,n) \tag{2.3}$$

2.1.5. Capas de convolución y capas de *pooling*

En una convolución se involucran dos elementos: un mapa de características con dimensiones $m \times n \times d$, y un *kernel* o filtro con dimensiones $f_h \times f_w \times d$. Para que se pueda dar la operación de convolución, la profundidad (d) de los *kernels* tiene que coincidir con la del mapa de características. El mapa de características resultante tiene una dimensión de $(m - f_h + 1) \times (n - f_w + 1) \times 1$. El número de *kernels* determina la profundidad del próximo mapa de características [30], [33]. En la figura 2.2, se puede observar de manera gráfica, la convolución entre un mapa de características y un *kernel*.

Los *kernel* más comunes son de tamaño 3×3, 5×5 y 7×7, esto permite ir reduciendo la dimensión de los mapas de características. La cantidad de *kernels* permite modificar su profundidad. Dadas estas

[1] La operación de convolución usualmente se denota por un asterisco: $s(t) = (x * w)(t)$

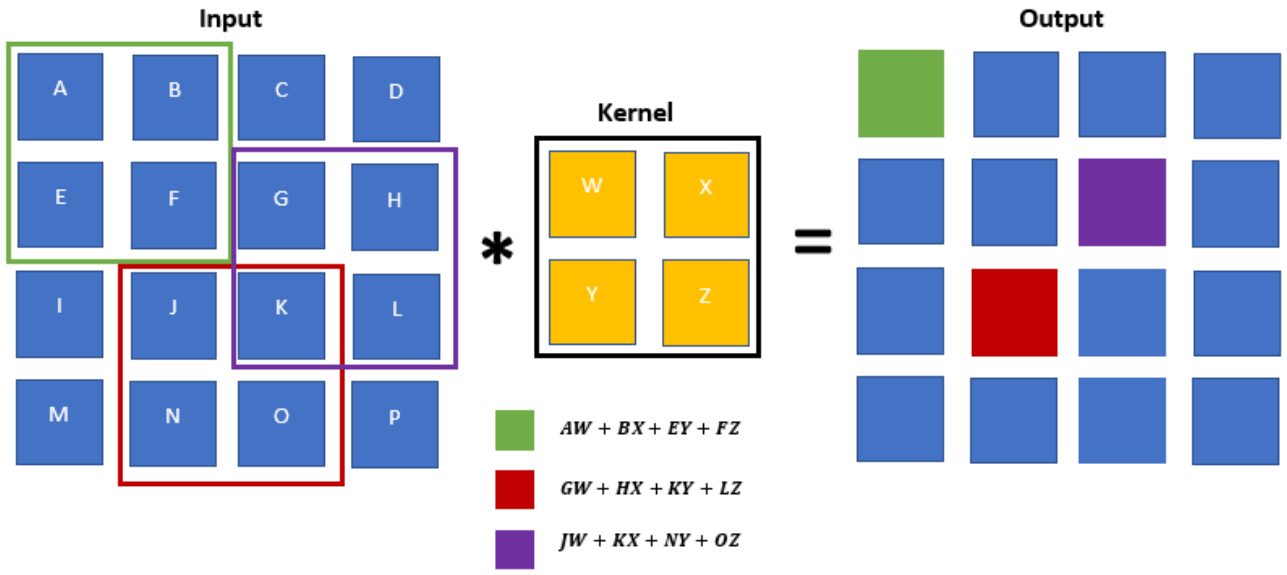

Figura 2.2: Representación gráfica de la convolución entre un mapa de características y un *kernel*

propiedades, se definen dos tipos de convolución comunes, además de la convolución estándar [43]:

- Convolución en profundidad o *depthwise convolution*: Aplica un filtro convolucional diferente a cada mapa de características para capturar características visuales. Esto implica que la profundidad se mantendrá en el siguiente mapa de características.

- Convolución puntual o *pointwise convolution*: Es un caso especial de convolución estándar, donde el tamaño de *kernel* es uno, para combinar linealmente diferentes canales de entrada. Además, sirve para compactar los canales de entrada, y hacerlos coincidir si se emplea un bloque residual invertido.

Las capa de *pooling* o agrupación son empleadas en casi todas las CNN. Se parte de una capa convolucional, seguida por una función de activación y posteriormente se aplica la capa de *pooling*. Esta capa reemplaza la salida de la red en una ubicación determinada, con una estadística resumida de las salidas cercanas. Los tipos más comunes son *max-pooling* y *average-pooling* [30].

2.1.6. Funciones de activación

A continuación se presentan tres de las funciones de activación más empleadas en las redes neuronales convolucionales.

- ***Sigmoid***

  La función de activación *sigmoid* tiene dos propiedades positivas: su forma es de "S" y su derivada es fácil de obtener. Matemáticamente se define como [44]:

$$f(x) = \frac{1}{1 + e^{\alpha x}} \tag{2.4}$$

- ***ReLU***

  Rectified linear activation function, es la función de activación recomendada por defecto en la mayoría de las redes neuronales *feedforward*. Su comportamiento es simple, convierte en cero los valores negativos y deja intactos los positivos. Matemáticamente se expresa como [30]:

$$f(x) = \begin{cases} 0 \quad si \quad x \leq 0 \\ \\ x \quad si \quad x > 0 \end{cases} \tag{2.5}$$

- ***Softmax***

  La función *Softmax* es comúnmente empleada en la capa de salida que irá directamente como entrada a la función de pérdida. Por lo general, la función de pérdida involucra un logaritmo, se sabe que las funciones logaritmo y exponencial son inversas, y que las funciones exponenciales son fáciles de derivar. Esto facilita los cálculos y por lo tanto el tiempo requerido para ellos. Matemáticamente se define como [44].

$$f(x) = \frac{e^{x_i}}{\sum_i e^{x_i}} \tag{2.6}$$

2.1.7. Funciones de pérdida

La selección de la función de pérdida depende del tipo de problema abordado. Para un problema de clasificación, se quiere predecir una distribución de probabilidad sobre un conjunto de clases. En problemas de regresión, sin embargo, se busca predecir un valor específico en lugar de una distribución. A continuación se muestran las funciones de pérdida básicas definidas en [33].

- **Error cuadrático medio (MSE)**

  El error cuadrático medio calcula el promedio de la diferencia entre la predicción de clasificación y el objetivo. Entrenar con el MSE minimiza la diferencia de magnitud. Un inconveniente con MSE es que es susceptible a los valores atípicos ya que la diferencia es al cuadrado. Normalmente se emplea en problemas de regresión.

$$E(y, \hat{y}) = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{2.7}$$

- **Error absoluto medio**

  El error absoluto medio, calcula el valor absoluto de la diferencia entre la predicción de clasificación y el objetivo. Usarlo minimiza la magnitud del error sin tener en cuenta dirección, haciéndolo sensible a los valores atípicos.

$$E(y, \hat{y}) = \frac{1}{n} \sum_{i=1}^{n} |y_i - \hat{y}_i| \tag{2.8}$$

- **Probabilidad de registro negativo**

  La probabilidad de registro negativo o *Negative Log Likelihood* (NLL), también conocida como función de pérdida de entropía cruzada multiclase o *multiclass cross-entropy loss*, es la función de pérdida que más se utiliza para problemas de clasificación multiclase. La función *softmax* propor-

ciona una distribución de probabilidad sobre las clases de salida. El cálculo de la entropía es una probabilidad de registro promedio ponderada, sobre los posibles eventos o clasificaciones en un problema de clasificación multiclase. Esto hace que la pérdida aumente a medida que la distribución de probabilidad de la predicción diverge de la etiqueta objetivo.

$$E(y, \hat{y}) = -\frac{1}{n}\sum_{i=1}^{n}(y_i log(\hat{y}_i) - (1 - y_i) log(1 - \hat{y}_i)) \tag{2.9}$$

2.2. Aumento de datos

El aumento de datos o *data augmentation* (DA), es una técnica utilizada para generar nuevas muestras de entrenamiento a partir de las originales, comúnmente mediante transformaciones geométricas como: traslación, rotación, cambios de escala y desplazamiento horizontales (algunas veces verticales). El objetivo de emplear esta técnica es aumentar la generalización del modelo, ya que, si la red ve constantemente nuevas versiones ligeramente modificadas de los datos de entrada, puede aprender funciones más sólidas [45].

Algunas otras estrategias de DA son: *random erasing* donde se elimina aleatoriamente recuadros de las imágenes; *mixing images* que consiste en unir secciones de distintas imágenes; transformaciones de espacio de color; agregar ruido, por ejemplo agregando *salt and pepper noise*; *kernel filters*, es decir, función de procesamiento de imágenes para enfocar y desenfocar [46].

Recientemente, se han comenzado a emplear con éxito estrategias como *copy-and-paste* [47], [48] y *cut-and-paste* [49], [50]. Estos métodos de aumento de datos se emplean en la problemática de detección de objetos, requieren la información del *bounding box* que contiene a los objetos. Los nombres de estas estrategias son intuitivos, *copy-and-paste* consiste en colocar alrededor de la imagen copias del objeto, mientras que *cut-and-paste* consiste en tomar la sección que contiene al objeto y colocarlo en otro lugar de la imagen u otra imágen.

Una analogía de como funciona la transferencia de aprendizaje en los humanos, se da por ejemplo, en la música. Si dos personas están tratando de aprender a tocar la guitarra y una ya sabe tocar el piano, es probable que el pianista aprenda a tocar la guitarra más rápido [46].

2.3. Detección de objetos

La detección de objetos (DO) tiene como tarea, a partir de una imagen, clasificar y localizar los objetos que contiene. Es decir, indicar las coordenadas que delimitan al objeto, y la clase a la que pertenece. Anteriormente, se atacaba esta problemática empleando máquinas de vectores de soporte (SVM). Esto se hacia, cuando se tenían limitaciones para emplear redes neuronales, como: pocos datos de entrenamiento (causa de sobre-entrenamiento), recursos computacionales limitados y poco soporte teórico [19].

Los modelos de detección de objetos pueden ser divididos en dos familias: *one-stage* y *two-stage*. Los detectores *two-stage* primero usan un generador de regiones de interés (ROI) y extraen características de cada una, seguido de un clasificador de región que predice la categoría. Mientras que los detectores *one-stage* realizan directamente predicciones categóricas de objetos en cada ubicación de los mapas de características sin el paso de clasificación de regiones en cascada [19].

Fundamentalmente, la diferencia entre éstos es que los detectores *one-stage* eliminan el paso de agrupación de las RoI; combinan la detección y regresión en un mismo paso. La ventaja que tienen los modelos de *one-stage* es que pueden ser optimizados *end-to-end*. Por otro lado, una problemática con los detectores *two-stage* es que requieren grandes recursos computacionales, y son frecuentemente lentos lo que les resta utilidad en aplicaciones de la vida real [11].

Dentro de los detectores de *two-stage* se encuentran R-CNN [51], SPP-net [52], *Fast* R-CNN [53], *Faster* R-CNN [18]. Por otro lado, algunos detectores de *one-stage* son YOLO [25], SSD [24], CornerNet [54],

RetinaNet [55] y *EfficientDet* [1].

2.3.1. *Bounding boxes*

Los *bounding boxes*, son recuadros que delimitan el espacio que contiene a un objeto en una imagen. Pueden estar dados por las coordenadas de las esquinas inferior izquierda y superior derecha, o por las dimensiones de ancho y alto, y las coordenadas del centro. En la detección de objetos se tendrán *bounding box* de predicción y *ground truth bounding box* [25], [24]. Para evaluar que tan bien se realiza la predicción se emplea la métrica de *Intersection over Union* (IoU) [56].

Las formas y tamaño de los *bounding boxes* para predicción son fijos. Dependiendo del clasificador o la base de datos, se fijan dimensiones (ancho y alto), y la cantidad. En SSD, los llaman *priors*, y se fijan 6 distintas formas. En YOLO y la mayoría de los detectores, los llaman *anchors*, en YOLO se fijan 3 distintas formas. Para determinarlos se emplean distintas estrategias, una de las más usadas es *k-means* empleando la información del *ground truth* [56].

2.3.2. Funciones de pérdida en detectores de objetos

En la problemática de detección de objetos, la función de pérdida debe considerar dos aspectos: la clasificación del objeto y la localización de sus coordenadas. A continuación se explicarán las funciones de pérdida de los modelos SSD, YOLO y *EfficientDet*.

La función de pérdida de SSD emplea dos funciones; $Dmooth_{L1}$ para la localización y *categorical cross-entropy* para la clasificación, en conjunto están dadas por [24]:

$$L(y, \hat{y}) = L_{conf} + \alpha L_{loc} \tag{2.10}$$

$$L(y,\hat{y}) = \frac{1}{N}\sum_{i\in Pos}^{N}\sum_{m\in\mathbb{R}} X_{ij}^{k} smooth_{L1}(l_i^m - g_i^m) + \frac{1}{N}\alpha\left(-\sum_{i\in Pos}^{N} X_{ij}^{k} log(c_i^k) - \sum_{i\in Neg}^{N} log(c_i^0)\right) \quad (2.11)$$

Donde:

- $\alpha = 1$
- N es el número de muestras.
- c_i^k es la probabilidad correspondiente a la clase k.
- $X_{ij}^k \in \{0,1\}$, es 1 solo si i-ésimo *bounding box* (BBox) se traslapa con el j-ésimo *ground truth box* (GT).
- $l_i^m = (l_i^x, l_i^y, l_i^h, l_i^w)$ y $g_i^m = (g_i^x, g_i^y, g_i^h, g_i^w)$ son las coordenadas del BBox y GT, respectivamente. Estas coordendas indican las coordenadas del centro en x y y, la altura y ancho.
- $Smoot_{L1}$ se define como [19]:

$$Smooth_{L1} = \begin{cases} 0.5x^2 & si \quad |x| < 1 \\ \\ |x| - 0.5 & si \quad |x| \geq 1 \end{cases} \quad (2.12)$$

Al observar la composición de la función $smooth_{L1}$, se aprecia que lo que se está calculando es el error cuadrático medio (MSE), para valores pequeños ($|x| < 1$) y para valores más grandes ($|x| \geq 1$) se calcula el error absoluto medio.

Cuando se obtiene la diferencia entre el BBox y GT con $l_i^m - g_i^m$ dados $l, g \in \mathbb{R}^4$, se está haciendo el cálculo de la distancia Manhattan que se define como [57]:

$$x = l_i^m - g_i^m = |l_i^x - g_i^x| + |l_i^y - g_i^y| + |l_i^h - g_i^h| + |l_i^w - g_i^w| \quad (2.13)$$

El resultado obtenido x será el valor que se evaluará en $Smooth_{L1}$.

Al estudiar qué es lo que hace esta función de pérdida, se analizaron los posibles valores que arrojan los logaritmos al evaluar probabilidades, es decir, números en el intervalo $[0, 1]$. Dado el comportamiento de los logaritmos, se notó que mientras la probabilidad c_i más se acerca a 1, menor es el valor arrojado por $log(c_i)$. Esto derivó en en las siguientes conclusiones sobre la función de pérdida:

- Si un elemento se clasifica con la clase correcta, mientras más baja sea la probabilidad con la que se clasificó mayor será el grado de penalización.
- Si un elemento se clasifica con la clase incorrecta, mientras más alta sea la probabilidad con la que clasificó mayor será el grado de penalización.

La función de pérdida de YOLO se obtiene empleando la suma del error cuadrático y está dada por [23]:

$$Loss = \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{I}_{ij}^{obj} \left[(x_i - \hat{x_i})^2 + (y_i - \hat{y_i})^2 \right] + \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{I}_{ij}^{obj} \left[(\sqrt{w_i} - \sqrt{\hat{w_i}})^2 + (\sqrt{h_i} - \sqrt{\hat{h_i}})^2 \right] +$$

$$\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{I}_{ij}^{obj} (c_i - \hat{c_i})^2 + \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{I}_{ij}^{noobj} (c_i - \hat{c_i})^2 + \lambda_{coord} \sum_{i=0}^{S^2} \mathbb{I}_{ij}^{obj} \sum_{c \in Classes} (p_i(c) - \hat{p_i(c)})^2 \quad (2.14)$$

Donde:

- $\lambda_{coord} = 5$ cuando hay objeto en la cuadrícula i.
- $\lambda_{nocoord} = 0.5$
- $\mathbb{I}_{ij} \in \{0, 1\}$ es 1 si hay traslape de objeto en la cuadrícula i con el BBox j.
- x, y son las coordenadas reales centro del objeto.
- $\hat{x}, \hat{y}$ son las coordenadas de la predicción.
- h, w son las dimensiones reales del GT.

- $\hat{h}, \hat{w}$ son las dimensiones de BBox de la predicción.
- c es el valor del IoU entre el GT y el BBox$_i$, si no hay objeto $c = 0$.
- $\hat{c} = Pr(Object) \cdot IoU$ (probabilidad de que exista un objeto en la cuadrícula i).
- Los límites de las sumas indican que se tendrán $B - 1$ *bounding boxes* y $S^2 - 1$ cuadrículas dadas por $S \cdot S$.

Con la finalidad de determinar por qué es importante trabajar con las raíces de las variables w, h en esta función, se hicieron cálculos de los cuales se llegó a la siguiente conclusión:

- Se podrían clasificar los objetos en grandes y pequeños, definiendo para un objeto grande un desfase pequeño entre GT y BBox, de tamaño Δx, suponiendo que un objeto pequeño tiene ese mismo desfase, ahora es posible considerar a Δx como un desfase significativo. Por lo tanto, un mismo incremento representa un mayor error en los objetos pequeños al considerar la relación entre Δx y w, h. Al trabajar con las raíces de los valores, dado un mismo desfase, se le permite a la función de pérdida penalizar en mayor grado a los BBox de objetos pequeños.

En *EfficientDet* se nombran dos predicciones: *class prediction net* y *box prediction net*. Para la parte de clasificación se emplea *focal loss* y para la localización se utiliza $smooth_{l1}$ *loss* (definida previamente en la ecuación 2.12).

Focal loss es una función de pérdida, especialmente diseñada para abordar el escenario de detección de objetos de una etapa, en el que existe un desequilibrio grande entre las clases de primer plano y de fondo durante el entrenamiento.

$$FL = -\alpha_t \cdot (1 - p_t)^{\gamma} \cdot log(p_t) \tag{2.15}$$

El valor de γ controla la forma de la curva de la función de pérdida. Cuanto mayor sea γ, menor será la pérdida para los ejemplos bien clasificados, por lo que se puede dirigir la atención del modelo hacia los

ejemplos difíciles de clasificar [58]. La función de pérdida completa, está dada por la ecuación 2.16.

$$L(y,\hat{y}) = \frac{1}{N}\sum_{i\in Pos}^{N}\sum_{m\in\mathbb{R}} X_{ij}^{k} smooth_{L1}(l_i^m - g_i^m) - \frac{1}{N}\sum_{i\in Pos}^{N} \alpha_t \cdot (1-p_t)^{\gamma} \cdot log(p_t) \tag{2.16}$$

2.3.3. Métricas en detección de objetos

La precisión es la fracción de detecciones realizadas por el modelo, que fueron correctas. El recall o sensitividad, es la fracción de eventos verdaderos que se detectaron. La *specificity*, es la fracción de eventos negativos que fueron clasificados como negativos. El objetivo o tarea que realiza el modelo es lo que indica cuál métrica es más importante de considerar [30]:

- Es mejor usar precisión.

  Si se quiere tener más confianza en los verdaderos positivos. Por ejemplo, correos electrónicos no desados. Es preferible tener algunos correos spam en la bandeja de entrada, que tener correos importantes en los correos no deseados.

- Es mejor usar *specificity*.

  Si se desea cubrir todos los verdaderos negativos, es decir, no se desea tener falsos positivos. Por ejemplo, si se están realizando pruebas antidrogas en las que las personas con resultados positivos tendrán consecuencias legales, no se quiere inculpar a un inocente.

- Es mejor usar recall o sensitividad.

  Si la idea de falsos positivos es mucho mejor que los falsos negativos, es decir, si la ocurrencia de falsos negativos es inaceptable. Por ejemplo, en lo referente a diagnósticos médicos, se preferiría tener pacientes sanos etiquetados como enfermos, que dejar a una persona enferma etiquetada como saludable.

- Es mejor usar sensitividad a determinado número de falsos positivos (FPs).

  Si se trabaja con imágenes médicas, es decir, si la información de la que aprende el modelo depende de características perceptivas humanas para interpretar las imágenes. En esta métrica se considera

la existencia de un nivel de confianza critico, que un observador emplea para distinguir imágenes positivas de negativas [59].

Métrica para DO: *Intersection over Union (IoU)*

La intersección sobre unión, es una métrica que se calcula entre la región de los objetos (*ground truth bounding box*), y los *bounding box* de predicción. Matemáticamente se calcula como se muestra a continuación:

$$IoU(b_{pre}, b_{gt}) = \frac{\text{Área}(b_{pred} \cap b_{gt})}{\text{Área}(b_{pred} \cup b_{gt})} \tag{2.17}$$

Donde b_{pre} y b_{gt} representan el *bounding box* de la predicción del modelo y el *bounding box* real, respectivamente. De manera gráfica, el IoU se puede ver como se muestra en la figura 2.3, donde en azul se tiene el objeto real y en rojo la predicción.

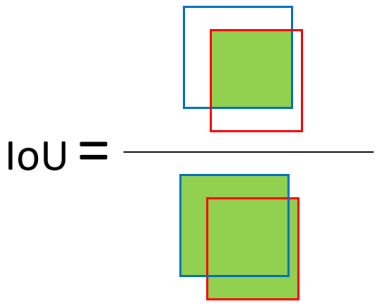

Figura 2.3: Representación gráfica de *Intersection over Union*

En la detección de objetos se fija un valor ω, de tal manera que sirve como criterio, para determinar si una predicción de localización se considera positiva o negativa. Generalmente se considera un $IoU > 0.5$ es una buena predicción. El criterio es el siguiente [19], [56]:

$$\text{Predicción} = \begin{cases} Positiva & c_{pre} = c_{gt} \quad y \quad IoU(b_{pre}, b_{gt}) > \omega \\ Negativa & Otro \quad caso \end{cases} \tag{2.18}$$

Donde, c_{pre} indica la clase de la predicción, y c_{gt} la clase real.

Métricas de evaluación para DO según MS COCO

De acuerdo con la página oficial de MS COCO (https://cocodataset.org/detection-eval), las siguientes 12 métricas se utilizan para caracterizar el desempeño de un detector de objetos:

- *Average Precision (AP)*:
  - AP: un AP con IoU=.50:.05:.95[2]
  - AP$^{IoU=.50}$: un AP con IoU=.50 (métrica de PASCAL VOC)
  - AP$^{IoU=.75}$: un AP con IoU=.75 (métrica estricta)
- *AP Across Scales*:
  - AP^{small}: AP para objetos pequeños (área $< 32^2$)
  - AP^{medium}: AP para objetos medianos ($32^2 <$ área $< 96^2$)
  - AP^{large}: AP para objetos grandes (área $> 96^2$)
- *Average Recall* [3] *(AR):*
  - AR$^{max=1}$: AR dada 1 detección por imagen
  - AR$^{max=10}$: AR dadas 10 detecciones por imagen
  - AR$^{max=100}$: AR dadas 100 detecciones por imagen
- *AR Across Scales:*
  - AR^{small}: AR para objetos pequeños (área $< 32^2$)
  - AR^{medium}: AR para objetos medianos ($32^2 <$ área $< 96^2$)
  - AR^{large}: AR para objetos grandes (área $> 96^2$)

Nota: Si no se especifica un valor de IoU, el AP y AR, son equivalentes a mAP y mAR.

[2] IoU=.50:.05:.95 indica que se obtiene un promedio entre 10 IoU que van de 0.5 a 0.95 con incrementos de 0.05
[3] *Recall* también es conocido frecuentemente como *sensitivity*

2.3.4. Validación cruzada

La validación cruzada o *cross-validation*, es una técnica que permite reducir la incertidumbre estadística, en torno al error de prueba estimado. Para esto, se particiona k veces el conjunto de datos en: entrenamiento/validación y prueba. Una vez que el algoritmo es evaluado con las k particiones o *k-folds*, se calcula el promedio [30]. En la imagen 2.4, se muestra una representación gráfica de las particiones para una validación cruzada.

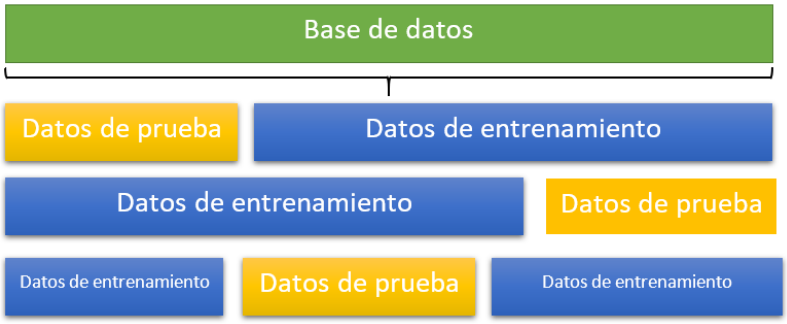

Figura 2.4: Representación de validación cruzada

2.3.5. Bases de datos

Las bases de datos que se mencionarán en esta sección, son las más utilizadas en el área de detección de objetos.

- **PASCAL VOC2007**

  Pascal VOC2007 es una base de datos de escala mediana, para detección de objetos con 20 categorías que incluyen: personas, animales, medios de transporte y objetos dentro de casa. Está dividida en entrenamiento, validación y prueba con 2501, 2510 y 5011 imágenes, respectivamente. Esta base de datos se puede encontrar en: http://host.robots.ox.ac.uk/pascal/ VOC/voc2007/.

- **PASCAL VOC2012**

  Pascal VOC2012 es una base de datos de escala mediana, para detección de objetos con 20 categorías que incluyen: personas, animales, medios de transporte y objetos dentro de casa. Está

dividida en los conjuntos de entrenamiento, validación y prueba con 5717,2510 y 10991 imágenes, respectivamente. Esta base de datos se puede encontrar en: http://host.robots.ox.ac.uk /pascal/VOC/voc2012/

- **MS COCO**

  MS COCO es una base de datos de detección, segmentación y *captioning* de gran escala creada por Microsoft, que consta de 80 categorías. Se divide en tres conjuntos: entrenamiento, validación y prueba con 5717, 5000 y 10991 imágenes, respectivamente. El *target* del conjunto *test* no se encuentra disponible. Esta base de datos se puede obtener en: http://cocodataset.org/# download.

- **ImageNet**

  ImageNet es un conjunto de datos de imágenes organizadas de acuerdo con la jerarquía WordNet. Cada concepto significativo en WordNet, es descrito por un conjunto de sinónimos denominados sysnets. Hay más de 100,000 sysnets en WordNet. El objetivo de ImageNet es ilustrar cada synset con un promedio de 1000 imágenes. Estas tienen control de calidad y anotaciones humanas [17].

- **DeepLesion**

  DeepLesion es la base de datos de tomografías computarizadas más grande reportada hasta la fecha. Esta base de datos nació por la necesidad de un conjunto de datos de imágenes a gran escala, orientado al ámbito médico. Este tipo de base de datos no es fácil de crear por dos principales razones: no se pueden aplicar métodos tradicionales como en el caso de ImageNet, ya que se requiere amplia experiencia clínica, por otra parte, con frecuencia ocurre una variabilidad significativa entre observadores e intra-observadores [2].

Capítulo 3

Marco teórico

Este capítulo está dividido en 3 secciones: modelos de detección de objetos, *EfficientDet* y módulos de atención en detección de objetos. En la primer sección, se explica de manera general modelos de detección. La segunda sección, se centra en la descripción del modelo *EfficientDet*. Finalmente, se describen módulos de atención propuestos para la detección de objetos.

3.1. Modelos de detección de objetos

Anteriormente, se atacaba la problemática de detección de objetos con la implementación de algoritmos del área de ML, como máquinas de vectores de soporte o redes neuronales. El problema con esto es identificar adecuadamente las características, para lograr buenos resultados. Por otro lado, el aprendizaje profundo no se consideraba una opción viable por dos razones principales: bases de datos pequeñas que causaban sobre-entrenamiento y recursos computacionales limitados. En la actualidad ya no se tienen esas problemáticas, gracias a la generación de bases de datos públicas con miles o millones de muestras, y con la llegada de GPUs potentes lo que permite desarrollar soluciones *end-to-end* [19].

Las estrategias que han marcado el estado del arte en los últimos años en la detección de objetos se basan en su totalidad en redes neuronales convolucionales (CNN). Estas arquitecturas se pueden dividir

en dos familias: detectores de una etapa y detectores de dos etapas. En este trabajo se centrará la atención en los detectores de una etapa ya que éstos pueden ser optimizados de extremo a extremo o *end-to-end* [20]. Algunos de los detectores de esta familia, en el orden en que se publicaron son: YOLOv1 [23], SSD [24], YOLOv2 [60], YOLOv3 [25], YOLOv4 [61] , PP-YOLO [62] y *EfficentDet* [1].

Hasta YOLOv3, los sistemas de detección de objetos que conformaban el estado del arte variaban en cuanto a los siguientes aspectos: hipotetizar cuadros delimitadores, volver a muestrear píxeles o características para cada cuadro o aplicar un clasificador de alta calidad [24]. Posteriormente, en detectores como YOLOv4, PP-YOLO y *EfficentDet*, se realizan variaciones en cuanto al *backbone* para extraer mapas de características a diferentes escalas, en el *neck* para construir una pirámide de características con conexiones laterales entre mapas de características, y en el *head* donde se realiza la predicción y localización [62].

En la figura 3.1, se muestra una línea de tiempo de detectores de objetos del estado del arte de los años 2015 - 2020. En el 2015 se tiene *Faster R-CNN*; en 2016 surge YOLOv1 y posteriormente en ese mismo año SSD; en 2017 se propone YOLOv2 y en 2018 YOLOv3; finalmente en 2020 surgen 3 propuestas: YOLOv4, *EfficientDet* y PP-YOLO.

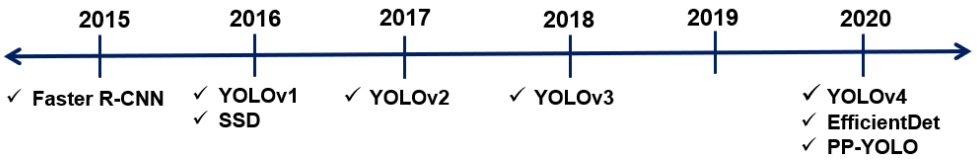

Figura 3.1: Linea de tiempo de los detectores de objetos del periodo de tiempo 2015 - 2020

.

A continuación, se describirá de manera general las implementaciones reportadas en varios algoritmos de detección de objetos basados en CNN: *Faster R-CNN*, *Single Shot Detector (SSD)*, versiones de *You*

Only Look Once (YOLO) y EfficentDet.

Faster R-CNN es una versión mejorada de *Fast R-CNN*, aquí se introduce *Region Proposal Network* (RPN). Este modelo está compuesto por dos módulos. El primero es una red profunda totalmente convolucional que propone regiones. Es decir, predice simultáneamente los contornos de los objetos y probabilidad de clase en cada posición. El segundo módulo es el detector *Fast R-CNN* que utiliza las regiones propuestas. En este trabajo se introduce el termino de *anchor*, que sirven como referencia en múltiples escalas y proporciones. Empleando *Faster R-CNN*, con la base de datos MS COCO, se reporta un 42.7mAP@0.5 (*mean average precision*) [18].

Single Shot Detector (SSD) es una red *feed-forward*. Este tipo de redes trabajan con cierta operación denominada convolución, y además, no se forman ciclos a diferencia de las redes recurrentes [24]. De estas características se origina el nombre del algoritmo; "*single shot*" debido a que la localización y detección se realiza en un solo paso hacia adelante. Esto se logra al no tener retroalimentación o ciclos, y "*detectors*" porque localiza en una imagen las coordenadas del centro de los objetos, sus dimensiones (*x,y,w,h*) y las etiquetas de sus clases [56].

Originalmente, los autores de SSD utilizaron la VGG16 como red base, sin embargo, actualmente existen otras alternativas de redes base que pueden incrementar la exactitud o rapidez del modelo [56]. La arquitectura de SSD está dividida en bloques de convoluciones, cada bloque contiene una capa de convolución seguida por una función ReLU y una capa de *max-pooling*. Esta primera parte, tiene la función de ir reduciendo el tamaño del mapa de características e ir incrementando la profundidad de la red. Esto permite realizar una detección a seis niveles. La segunda parte del bloque está compuesta por dos capas de convolución, donde, una va orientada a la detección de clase y la otra a la predicción de posición. SSD ha reportado un 73.1 mAP con la base de datos PASCAL VOC2012, un 75.1 mAP con PASCAL VOC2007 y 43.7 mAP con MS COCO [24].

You Only Look Once (YOLO) lleva dicho nombre debido a que el algoritmo "solo mira una vez" la imagen, en el sentido de que solo requiere un paso de propagación directa, a través de la red para hacer predicciones [23]. Este algoritmo ha tenido tres versiones propuestas por el autor original Joseph Redmon, cada una es una mejora de la anterior por lo que se centrará la atención en YOLO v3.

El modelo YOLO emplea herramientas de agrupamiento, para definir un número fijo de *anchors*. En YOLO v1 se definen 5 *anchors* mientras que en YOLO v2 y v3 se incrementa a 9 *anchors*. En YOLO v3 se realiza una detección a 3 niveles y se definen 3 *anchors* para cada nivel. La red está compuesta por 53 capas convolucionales, emplea una nueva métrica para detección, prediciendo una puntuación de "objetividad" para cada *bounding box*. Se emplea regresión logística y la función de pérdida de entropía cruzada binaria para predicciones de clase durante el entrenamiento [60], [25].

Los resultados reportados por YOLO v2 empleando la base de datos PASCAL VOC2007 es de 78.6 mAP, con PASCAL VOC2012 73.4 mAP y en MS COCO un 44 mAP [60]. Mientras que YOLO v3 ha reportado un 57.9 mAP_{50} con MS COCO [25].

A partir de las implementaciones de YOLO v4, PP-YOLO y EfficentDet, se define a los detectores de objetos mediante 4 elementos básicos: *input*, *backbone, neck* y *head* [42], [61], [63].

- El *input* es la imagen de entrada, que puede tener diferentes dimensiones de ancho y alto, y comúnmente profundidad de 3 al trabajar con imágenes en escala RGB.
- El *backbone* está compuesto por un modelo pre-entrenado con ImageNet y diseñados para clasificación . Algunos de los *backbones* que se emplean al trabajar con GPU son: VGG16, ResNet, Darknet y EfficentNet.
- El *neck* está compuesto por un conjunto de capas con una combinación de conexiones de arriba hacia abajo y de abajo hacia arriba, para fusionar características en diferentes escalas. Como ejemplos de *necks* se tienen: NAS-FPN, FPN, BiFPN, PAN.

- Finalmente, se tiene el elemento *head*, es el modelo de detección de objetos que no sólo determina la clase de los objetos en una imagen, sino que también predice su localización. Como ejemplos de *heads* que se toman frecuentemente son las de: SSD, YOLO en sus diferentes versiones, RetinaNet y *EfficentDet*.

En el modelo de YOLO v4 [61], se parte de la existencia una gran cantidad de características, que se afirma mejoran la precisión de las redes neuronales convolucionales. El propósito del trabajo realizado es realizar pruebas prácticas de combinaciones de dichas características en conjuntos de datos grandes. Para esto parten de la división de características en dos tipos: *bag of freebies* y *bag of specials*. Se define *bag of freebies* como el conjunto de estrategias para reducir el *bias*, mejorando la precisión sin aumentar el costo de inferencia. Sólo se cambia la estrategia de entrenamiento o se aumenta. Las *bag of specials* son módulos de complemento y métodos de pos-procesamiento, que ó aumentan el costo de inferencia en una pequeña cantidad.

Después de probar las combinaciones con los elementos propuestos, se elige la combinación que arroja los mejores resultados: CSPDDarknet53 como *backbone*, SPP, PANet *path-aggregation* como *neck*, y YOLOv3 como *head*, funciones de activación Mish, CIoU como *BBox regression loss*, CutMix y Mosaic como estrategias de *data augmentation*, DropBlock como método de regularización y *cross stage partial connections*. Con esta arquitectura reportan un AP de 43 % con la base de datos MS COCO [61].

PP-YOLO, fue nombrado así ya que esta implementación emplea YOLOv3, y dado que las experimentaciones están desarrolladas en PaddlePaddle. A diferencia de YOLOv4, en este trabajo no se exploran diferentes redes para el *backbone*, técnicas de *data augmentation*, ni NAS para buscar los mejores hiperparámetros. Emplean como *backbone* ResNet, que es una de las redes más empleadas, para *data augmentation* usan MixUp básico. Se crea una nueva versión de ResNet, denominada ResNet50-vd-dcn, esto bajo ciertas modificaciones y remplazando algunas capas por capas de convolución deformables. En general lo que se busca en el trabajo de PP-YOLO no es presentar un método de detección de objetos

innovador, sino introducir una serie de ”trucos”, que permitan mejorar el rendimiento de YOLOv3 con solo un pequeño costo en eficiencia. En PP-YOLO reportan con la base de datos MS COCO, un mAP de 45.2 % [62].

EffidentDet es un detector de objetos que actualmente reporta resultados del estado del arte. Se caracteriza por su enfoque centrado en la fusión eficiente de características de múltiples escalas, y un escalado compuesto de modelos. Se emplea *EfficentNet* y BiFPN como *backbone* y *neck*, respectivamente. Los autores de *EfficentDet* nombran de manera diferente estos elementos, ellos señalan que su modelo está compuesto por: *backbone network*, *feacture network* y *class/box prediction network*. Se crean 8 modelos *EfficentDet*-D0 a D7, cada uno con mayor complejidad, el uso de estos va a depender de la aplicación que se le quiera dar al modelo. Respecto a la base de datos MS COCO, se reporta un AP de 55.1 %, con 77M de parámetros [1]. En la sección 3.2 se detallará este modelo.

3.2. *EfficientDet*

EffidentDet es un detector de objetos que actualmente reporta resultados del estado del arte. En esta sección se detallarán los elementos que lo conforman: *backbone network*, *feacture network* y *class/box prediction network* [1]. La arquitectura de *EfficentDet* está dada por la figura 3.2:

3.2.1. *Backbone Network - EfficentNet*

EfficentNet es un modelo propuesto en 2019 por Tan & Le [64], en este trabajo se hace hincapié sobre como en trabajos anteriores se solía emplear estrategias como aumentar alguna característica de la red como: la profundidad (número de capas), el ancho (número de canales) o la resolución (tamaño de la imagen). Mediante una observación empírica se dieron cuenta de que estas 3 características no son independientes. En este modelo se propone un método nuevo de escalado compuesto, empleado para escalar uniformemente todas las dimensiones de profundidad, ancho y resolución.

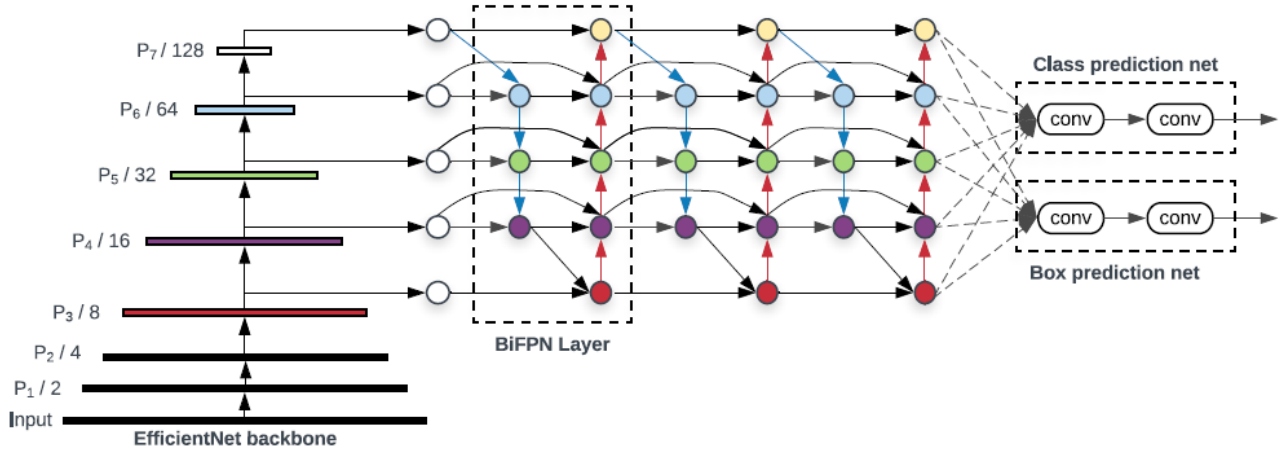

Figura 3.2: Aquitectura de EfficentDet: Emplea EfficentNet como *backbone network*, BiFPN como *feature network*, y *class/box prediction network* [1]

.

En *EfficentNet*, se propone un nuevo método de escalada compuesta que emplea un coeficiente ϕ, para escalar uniformemente la profundidad, ancho y resolución. Bajo las restricciones $\alpha \cdot \beta^2 \cdot \gamma^2 \approx 2$ y $\alpha, \beta, \gamma \geq 0$, d (profundidad), w (ancho) y r (resolución) se definen como:

$$d = \alpha^\phi$$

$$w = \beta^\phi$$

$$r = \gamma^\phi$$

El valor de ϕ controla cuantos más recursos hay disponibles para escalar el modelo. Los valores de α, β, γ se seleccionaron por medio de una *grid search*, estos especifican como se asignaran los recursos extra para d, w, r.

Dado el valor de ϕ se generan 8 modelos, todos parten de una *baseline EfficentNet*-B0 con $\phi = 0$. Estos modelos son generados empleando valores de $\phi = \{0, 1, 2, 3, ..., 7\}$. Los mejores parámetros para $\phi = 0$ son $\alpha = 1.2, \beta = 1.1, \gamma = 1.15$, para obtener los modelos de EfficentNet-B0 a B7 se fijan los parámetros de α, β, γ y se emplean los diferentes valores de ϕ. En la figura 3.3, se muestra la tabla con la arquitectura de *EfficentNet*-B0.

Stage i	Operator $\hat{\mathcal{F}}_i$	Resolution $\hat{H}_i \times \hat{W}_i$	#Channels $\hat{C}_i$	#Layers $\hat{L}_i$
1	Conv3x3	224×224	32	1
2	MBConv1, k3x3	112×112	16	1
3	MBConv6, k3x3	112×112	24	2
4	MBConv6, k5x5	56×56	40	2
5	MBConv6, k3x3	28×28	80	3
6	MBConv6, k5x5	14×14	112	3
7	MBConv6, k5x5	14×14	192	4
8	MBConv6, k3x3	7×7	320	1
9	Conv1x1 & Pooling & FC	7×7	1280	1

Figura 3.3: Descripción de los bloques de convolución, resolución de entrada, canales de salida y número de capas en EfficentNet-B0 [1].

Los bloques de MBConv, se basan en los propuestos en Mobilenetv2 [65] denominados bloques residuales invertidos o *inverted residual blocks*, y también emplean la estrategia de compresión y excitación o *squeeze-and-excitation* (SE) propuesta en [66].

Los bloques residuales invertidos emplean convoluciones separables en profundidad, acelerando con esto la extracción de características y guardando la escala del modelo con una precisión de predicción aceptable. Esto al descomponer la operación convolucional estándar en dos pasos: convolución en profundidad y convolución puntual [67]. Su estructura se muestra en la figura 3.4.

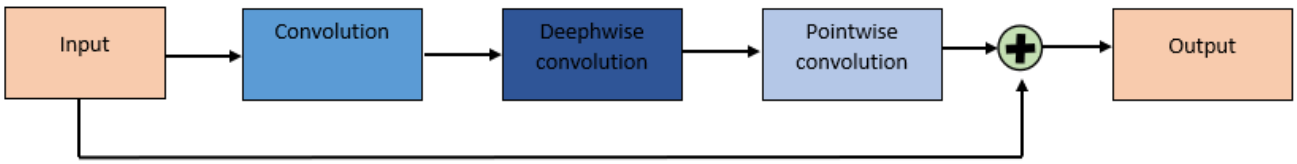

Figura 3.4: Arquitectura de *inverted residual blocks*

Los bloques de convolución en una CNN permiten a las redes construir características informativas fusionando información espacial y de canal. Una amplia gama de investigadores se han centrado en el componente espacial de esta relación, en la implementación de SE en cambio se enfoca en la relación del canal, que recalibra de manera adaptativa las respuestas de las características del canal, modelando

explícitamente las interdependencias entre canales [66]. Su arquitectura se muestra en la figura 3.5.

En la figura 3.4, se puede observar que la imagen pasa por una convolución estándar, donde el *kernel* puede ser de tamaño 3x3 o 5x5. Esta se sigue por una convolución en profundidad o *deephwise convolution* con *kernel* del mismo tamaño, la diferencia con este tipo de convolución es que aplica sólo un filtro convolucional a cada mapa de características y en consecuencia, el número de canales de salida se mantiene igual a la entrada.

Posteriormente se aplica una convolución puntual o *pointwise convolución*, con tamaño de kernel de 1x1, lo que combina linealmente los canales de entrada. Además, compacta los canales de la entrada para hacer coincidir la entrada y las salidas. Finalmente, la salida del bloque es la suma entre la salida de la ultima convolución y la entrada al bloque [67]. Este procedimiento se puede resumir en la ecuación 3.2.1 (se omite el *bias* por simplicidad):

$$x^{(n,1)} = x^{(n,l-1)} + \sum_{m=1}^{M} w_{1x1}^{(m)} * \left[w_{3x3}^{(m)} \sum_{n=1}^{\alpha N} \left(w_{3x3}^{(n,m)} \right) \right]$$

En la figura 3.5, se tiene inicialmente una operación de *pooling*, seguida por una capa totalmente conectada con función de activación ReLU, y después otra capa totalmente conectada con función de activación sigmoid [66].

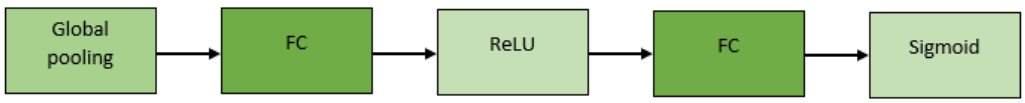

Figura 3.5: Arquitectura de squeeze and excitation (SE)

En la figura 3.6, se muestra como se combinan estas dos arquitecturas en una, para crear los elementos denominados MBConv que se emplea en EfficentNet.

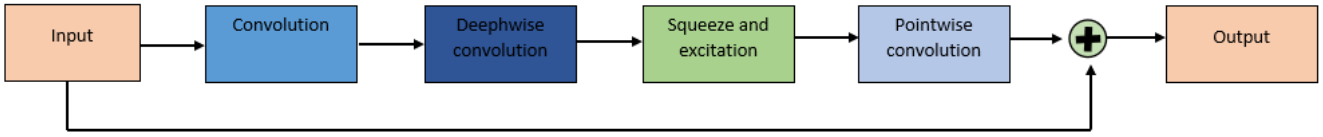

Figura 3.6: MBConv empleado en EfficentNet, combinación de los elementos mostrados en las figuras 3.4 y 3.5

3.2.2. *Feacture Network* - BiFPN

Las características de entrada al BiFPN son las correspondientes a los niveles 3 a 7 de *EfficentNet*, donde la i-ésima entrada corresponde a la característica del nivel con resolución $1/2^i$ de la imagen de entrada. Por ejemplo, si la resolución de la imagen es 640x640, en el nivel 3 $(640/2^3 = 80)$ la resolución es de 80x80 [1].

Weighted Bi-directional Feature Pyramid Network (BiFPN), es un tipo de red piramidal de características que permite la fusión de características de múltiples escalas (resoluciones) de manera fácil y rápida. El objetivo de BiFPN es que, dada una lista de características a múltiple escala $\vec{P}^{in} = (P^{in}_{l1}, P^{in}_{l2}, ..., P^{in}_{ln})$, donde P^{in}_{ln} representa la característica en el nivel l_i, encontrar una transformación f que pueda agregar de manera efectiva diferentes características y generar una lista de nuevas características $\vec{P}^{out} = f(\vec{P}^{in})$. Para esto se agrega un peso adicional para cada entrada y se deja que la red aprenda la importancia de cada característica.

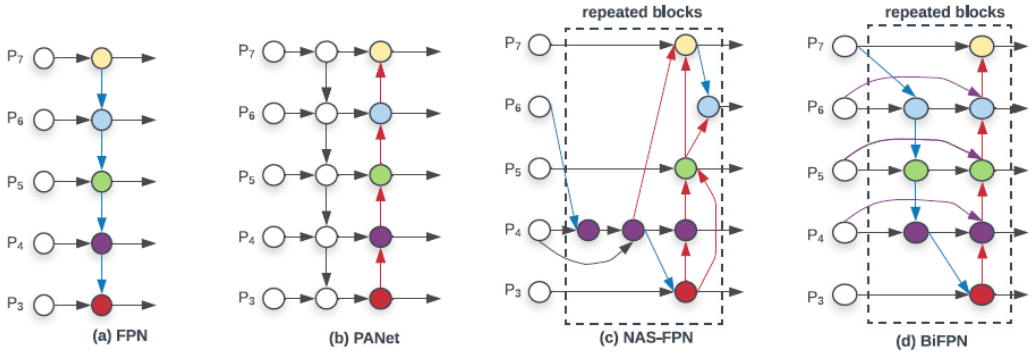

Figura 3.7: Diseño de la red de características. a) FPN, b) PANet, c) NAS-FPN, d) BiFPN [1]

BiFPN incorpora la idea de fusión de funciones multinivel de FPN [68], PANet [69] y NAS-FPN [42]. La cual permite que la información fluya en las direcciones de arriba hacia abajo y de abajo hacia arriba, mientras se utilizan conexiones regulares y eficientes. En la figura 3.7, se pueden apreciar las diferentes arquitecturas, originalmente en FPN se tienen conexiones unilaterales, en PANet ya se tienen conexiones bilaterales pero sólo entre niveles consecutivos, posteriormente en NAS-FPN se agregan conexiones entre distintos niveles, y finalmente, en BiFPN se tomaron las siguientes consideraciones [1]:

- Se remueven los nodos que sólo tienen una entrada. La intuición detrás de esta decisión es que si un nodo sólo tiene una arista de entrada, esta no tendrá fusión de características, por lo que contribuye menos a la red.
- Se agregan aristas extra de las entradas originales a la salida si están en el mismo nivel. Esto para fusionar más características sin agregar mucho costo.
- Se utilizan rutas bidireccionales como una capa de características, que se repite varias veces para una fusión de alto nivel.
- Dado el problema de normalización de ponderaciones que se tiene al emplear fusión basada en softmax, la cual causa desaceleración en el GPU. Se propone una fusión rápida normalizada, que es $30\,\%$ más rápida en GPU que con softmax, está se expresa de la siguiente manera:

$$O = \sum_i \frac{w_i}{\epsilon + \sum_j w_j} \cdot I_i \tag{3.1}$$

Donde:

$w_i \geq 0$ se asegura aplicando la función ReLU después de cada w_i,

$\epsilon = 0.001$ es un valor pequeño para evitar la inestabilidad numérica.

Dado que se proponen distintos modelos *EfficentDet*-D0 a D7, se establece un sistema de escalado para la entrada, ancho, profundidad y salida de BiFPN, todo esto respecto a un mismo parámetro ϕ empleado en el *backbone*. Estos escalamiento se describen a continuación:

- Para el ancho (número de canales), se establece un incremento exponencial

$$W_{BiFPN} = 64 \cdot (1.35^{\phi}) \tag{3.2}$$

Donde el valor de 1.35 se obtuvo de una *grid search* con los valores $\{1.2, 1.25, 1.3, 1.35, 1.4, 1.45\}$

- La profundidad se genera mediante un incremento lineal

$$D_{BiFPN} = 3 + \phi \tag{3.3}$$

- Dado que los niveles de fusión 3-7 se utilizan en BiFPN, la resolución de entrada debe ser divisible por $2^7 = 128$, por lo que se aumenta linealmente la resolución.

$$R_{in} = 512 + \phi \cdot 128 \tag{3.4}$$

3.2.3. *Class/box prediction network*

Se realizan 2 predicciones dadas por: *class prediction net* y *box prediction net*. La función de pérdida considera ambas predicciones. Para la parte de clasificación emplean *focal loss* y para la localización utilizan $smooth_{l1}$ *loss*. La función de pérdida se detalló previamente con mayor detalle en la ecuación 2.16.

3.3. Módulos de atención en detección de objetos

La precisión de detección actual está limitada por la incapacidad de las CNN en las lesiones a escalas diferentes. Los modelos de atención aumentan la capacidad de la red de seleccionar la respuesta de características relevantes para la detección de lesiones [5].

En [66] se propone el bloque *Squeeze-and-Excitation* (SE), se enfoca en el cambio en la relación de

canal. Este mecanismo se divide en dos partes: *squeeze* o compresión, y *excitation* o excitación. La primer parte se logra aplicando un *average global pooling*. En la segunda parte se crea un cuello de botella, empleando dos capas totalmente conectadas, y se aplica una función de activación sigmoid al final. Esta estructura se representa en la figura 3.8.

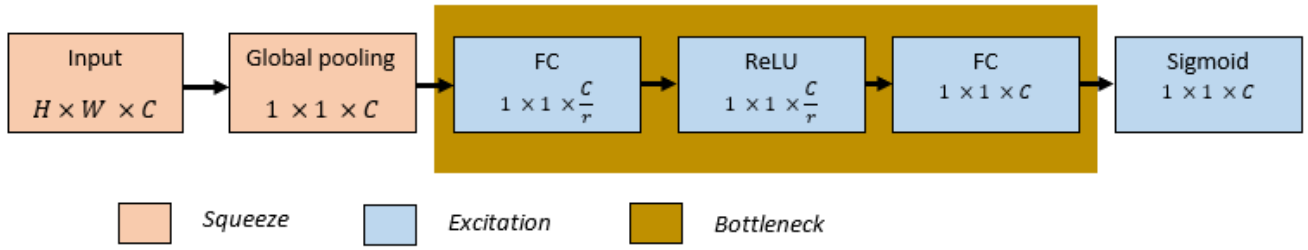

Figura 3.8: Módulo *Squeeze-and-Excitation*

Posteriormente, en [35] se propone un módulo basado en la propuesta de *Squeeze-and-Excitation*. El módulo propuesto, CBAM, infiere secuencialmente mapas de atención en dos dimensiones separadas, espacial y de canal. A continuación, los mapas de atención se multiplican con los mapas de características de entrada, para el refinamiento de características adaptativas.

En las figuras 3.9 y 3.10, se muestran los elementos de los módulos de atención de canal y espacial, respectivamente. Estos se colocan en medio de dos bloques de convolución en una red. El mapa de características arrojado por el bloque previo, entra al módulo de atención espacial. Se multiplica el mapa de características de entrada con el resultado del módulo de atención de canal (M_C). La salida ingresa al módulo de atención espacial, el resultado (M_S), se multiplica con lo que se ingresó. Este ultimo resultado se suma con la entrada proveniente del bloque de convolución anterior.

Por otro lado, en [37] se propone *Efficient Chanel Attention (ECA) module*. Aquí se toma *Squeeze-and-Excitation* como base, pero se modifica la sección del cuello de botella. Al igual que en SE, se aplica un *average global pooling*, a continuación en lugar de tener un cuello de botella, se tiene una convolución de 1D. Para esto se establece adaptativamente el tamaño de kernel (k), el resultado pasa por una función de

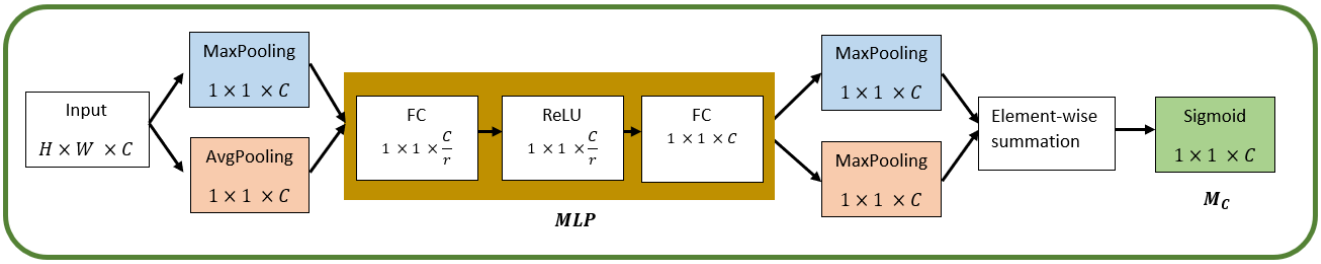

Figura 3.9: Módulo atención de canal de CBAM

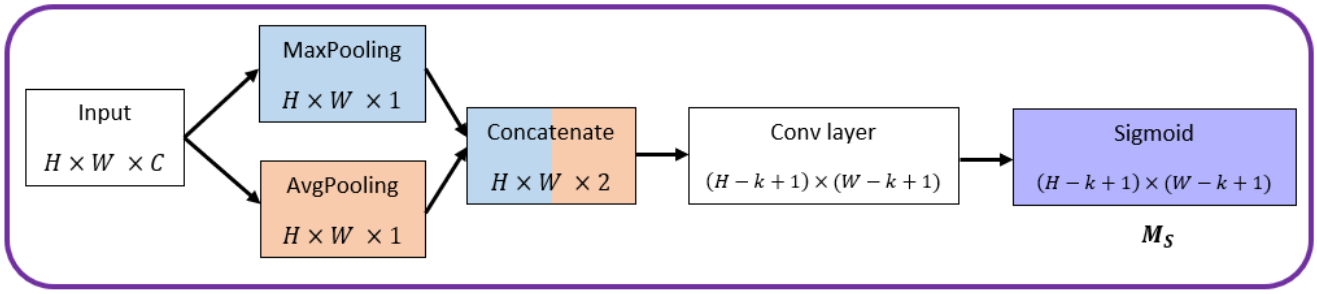

Figura 3.10: Módulo de atención espacial de CBAM

activación sigmoid. Finalmente, se multiplica esta salida con la entrada al módulo. Las partes del módulo ECA se pueden observar en la figura 3.11.

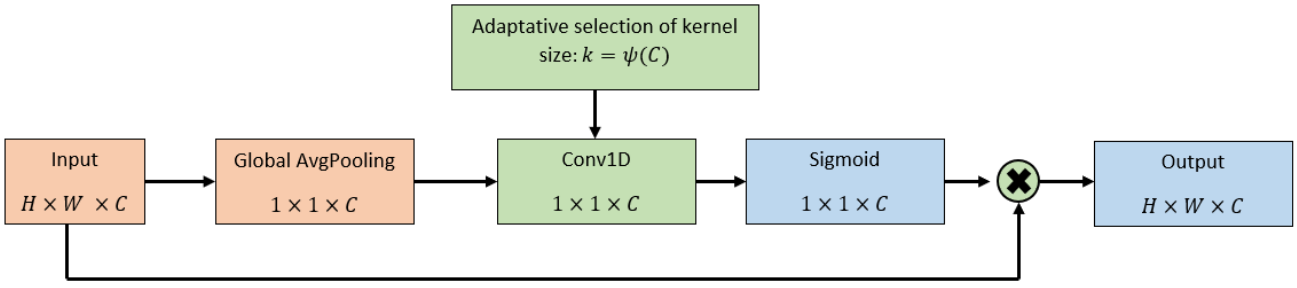

Figura 3.11: Módulo de ECA

Estos módulos de atención tienen la cualidad de ser ligeros y generales. Por lo que pueden ser empleados en distintas arquitecturas de CNN y con distintas bases de datos. En cada una de estas propuesta se proponen distintos enfoques, en SE y ECA se enfocan en la atención de canal, y en CBAM se hace un enfoque en atención de canal y espacial.

Capítulo 4

Trabajo relacionado

En este capítulo se presentará la base de datos *DeepLesion* [3], y se describirán trabajos relevantes, en los que se realiza detección de lesiones con dicha base de datos.

4.1. Base de datos *DeepLesion*

Deeplesion es una base de datos de tomografías computarizadas (CT) de rayos X, obtenidas a través de *picture archiving and communication system (PACS)*[1]. Estas contienen 8 tipos de lesiones, ya que fueron tomadas de distintas partes del cuerpo. La motivación para esto, es que sirva para la creación de un detector universal de lesiones. Las lesiones pueden estar localizadas en las siguientes regiones: hueso, mediastino, hígado, pulmón, riñón, pelvis, tejidos blandos[2] y abdomen [3]. En la figura 4.1, se presenta un ejemplo de cada uno de estos tipos de lesión.

La distribución y el total de cada tipo de lesión, se muestran en la gráfica de barras de la figura 4.2. En la primer barra (azul) se tiene la mayoría de los datos, estos no están etiquetados por lesión. Sin embargo, esto no representa un problema ya que se busca generar un detector de lesiones universal y no es indispensable dicha información. Dentro de las lesiones de las que si se conoce el tipo, las de pulmón

[1]Los PACS son una tecnología de imágenes que ayuda en la transmisión de imágenes desde el sitio de adquisición de imágenes a múltiples ubicaciones físicamente dispares

[2]Los tejidos blandos son lesiones diversas en la pared del cuerpo, músculos, piel, grasa, extremidades y cuello.

[3]Abdomen: lesiones en la cavidad abdominal que no se encuentran en el hígado o riñón

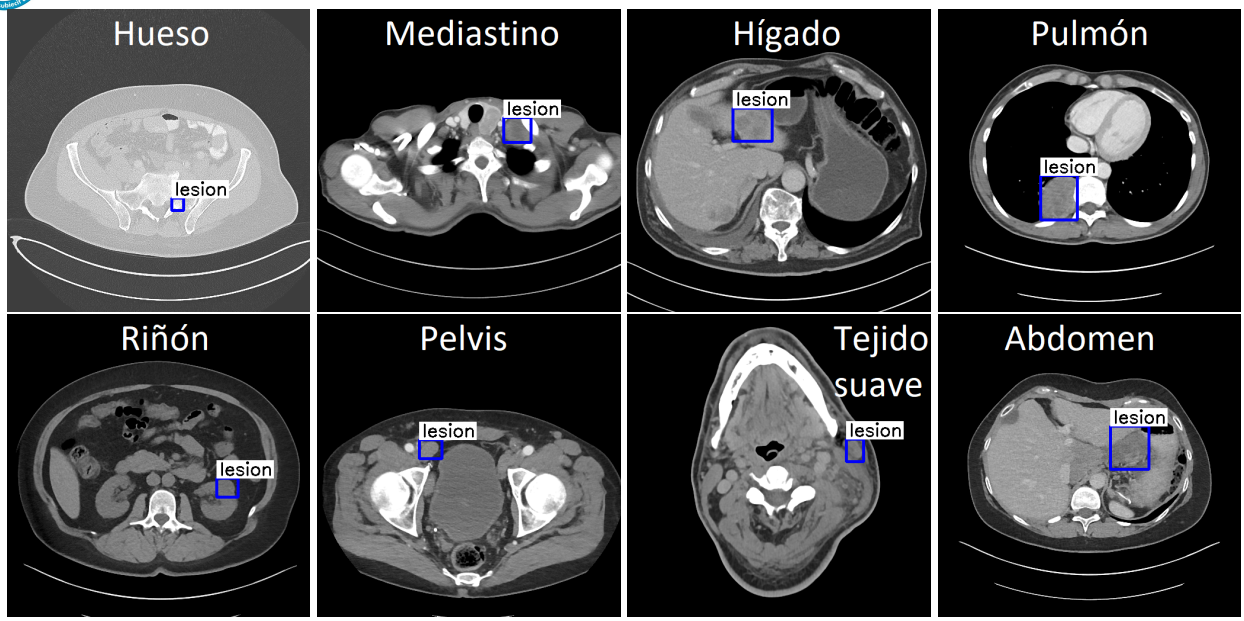

Figura 4.1: Muestras los distintos tipos de lesiones contenidas en la base de datos *DeepLesion*.

y abdomen son de las que se cuenta con más muestras, mientras que de las lesiones de riñón y hueso son de las muestras que menos se proporcionan.

DeepLesion fue generada en el *National Institutes of Health Clinical Center*, a partir de la recolección de tomografías (CT) por casi dos décadas. El ultimo año de recolección fue el 2017, los años donde se concentra la mayor cantidad de CT, son del 2015 a 2017. La decisión de crear la base de datos con este tipo de imágenes médicas, se basó en que el $59.53\,\%$ de los estudios médicos requieren una CT, seguido por las resonancias magnéticas con el $21.64\,\%$.

La base de datos contiene $33,688$ imágenes de radiología marcadas de $10,825$ estudios de $4,477$ pacientes. Las imágenes son de dimensión 512×512 [4], cada estudio contiene entre 1 - 3 lesiones, en total se tienen $32,735$ lesiones. La distancia entre los cortes de las tomografías varia entre 0.25 y 22.5 mm. La media de los *bounding boxes* es de 22.9 mm (ancho) y 22.3 mm (alto).

Del total de pacientes, $18,151$ $(55.45\,\%)$ son hombres y $14,583$ $(44.55\,\%)$ son mujeres. El rango de

[4] El 0.12 % de las imágenes son de dimensión 768×768

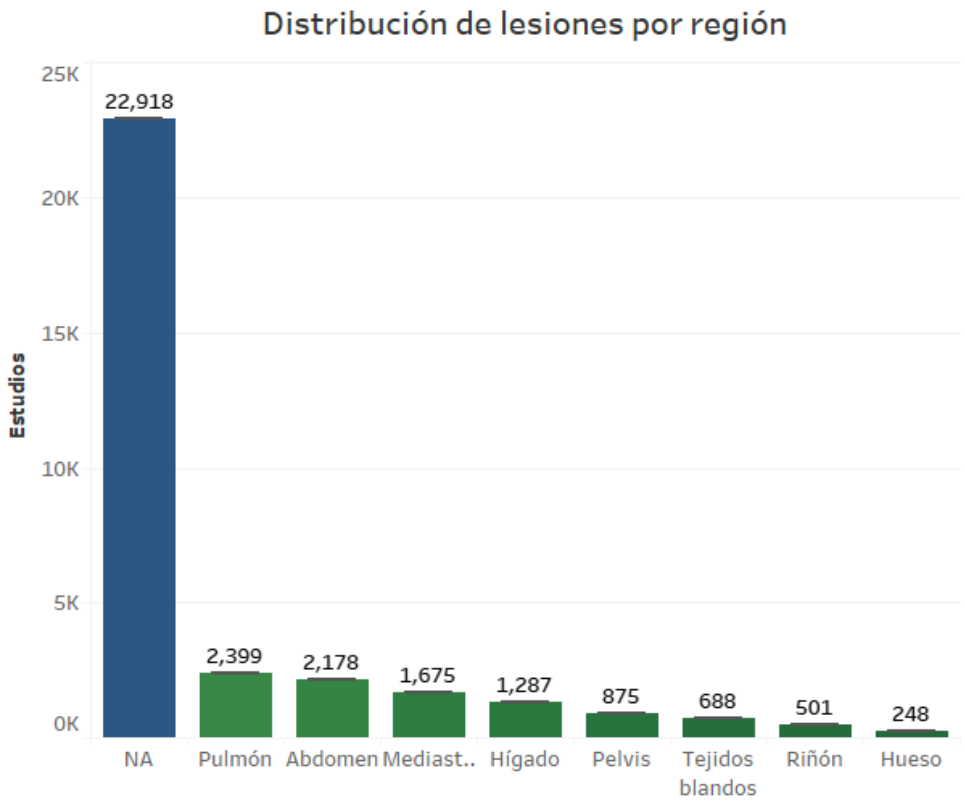

Figura 4.2: Distribución por región de las lesiones de *DeepLesion*

edades es grande, sin embargo, las edades de las mayores frecuencias se encuentran entre 50-70 años.

La base de datos está compuesta en un 61 % por imágenes con lesiones de tamaño *small*, 37 % de tamaño *medium* y un 2 % de tamaño *large*. Esto de acuerdo con el criterio de tamaños de objetos definido en la sección 2.3.3. En la figura 4.3, se muestran 3 ejemplos de cada tamaño de lesión.

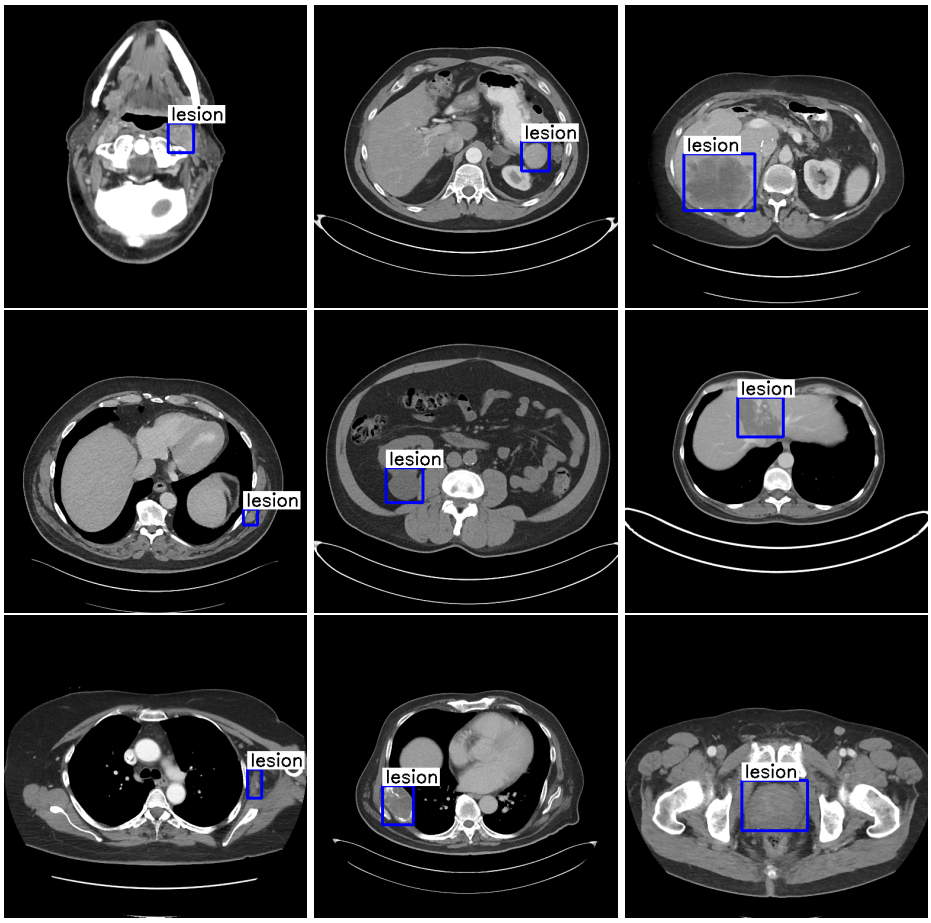

Figura 4.3: Muestras de tamaños de lesiones en las tomografías: *small*, *medium* y *large*

4.2. Detectores de lesiones

En esta sección, se presentan de manera general el trabajo relacionado a la detección universal de lesiones, y cada uno de estos modelos será acompañado por un diagrama de su estructura. La mayoría de dichos modelos, emplean FPN como red de características y el detector *faster* R-CNN. Estas propuestas muestran problemas principalmente en la detección de lesiones pequeñas, esto es debido a que ambos elementos, FPN y *faster R-CNN*, fallan precisamente en la detección de objetos pequeños.

Faster R-CNN es un detector *two-stage* que surgió en 2016, y ha sido superado por las distintas versiones de YOLO, SSD, y actualmente por el estado del arte que es EfficientDet, todos detectores *one-stage*. Por

otro lado FPN fue propuesto en 2017, pero de igual manera ha sido mejorado hasta llegar a BiFPN en 2020.

Los primeros dos trabajos reportado son realizados por los creadores de *DeepLesion*, en ambos se habla en gran medida sobre la base de datos. En la publicación del 2017, se propone emplear un modelo basado en *faster R-CNN* [18]. En esta no se emplea ningún método de aumento de datos. Se realiza el diseño del modelo de tal manera que se haga sólo una predicción con el *score* más alto por imagen, esto debido a que por cada imagen sólo se conoce una anotación. Esto es útil para mejorar las métricas, sin embargo, se está omitiendo el hecho de que en realidad pueden existir más de una lesión por imagen [2]. En la figura 4.4 se muestra la estructura de esta propuesta.

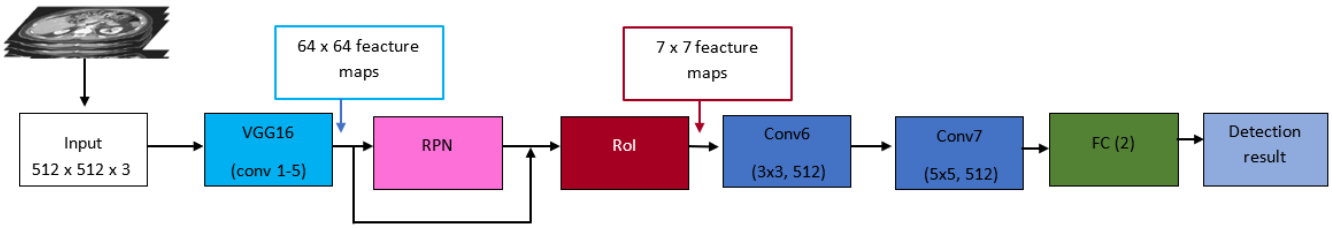

Figura 4.4: Estructura del modelo propuesto por Yan *et. al* [2]

Posteriormente, en 2018, Yan *et. al* realizan otra publicación donde llevan a cabo un análisis de los datos por tipo de lesión (8 tipos) y tamaño de lesión (7 grupos). Se propone el mismo modelo basado en *faster R-CNN*, pero aquí se considera el contexto 3D de las tomografías (esta información se emplea en todos los trabajos posteriores). Aquí afirman que no es necesario realizar ningún tipo de aumento de datos, ya que la base de datos es suficientemente grande. Sin embargo, no existe un criterio para decidir cuantos datos son suficientes para un modelo de aprendizaje profundo, y no se realizan pruebas realizando el aumento de datos para demostrar que su afirmación es verdadera [3]. En la figura 4.5 se muestra la estructura del modelo.

En el año 2019, se encontraron 4 trabajos, el primero de ellos es el realizado por Zhang *et. al* [4]. En este trabajo se emplean 3 métodos de aumento de datos: *flipping, cropping* y *resize*. Se emplea *flipping*

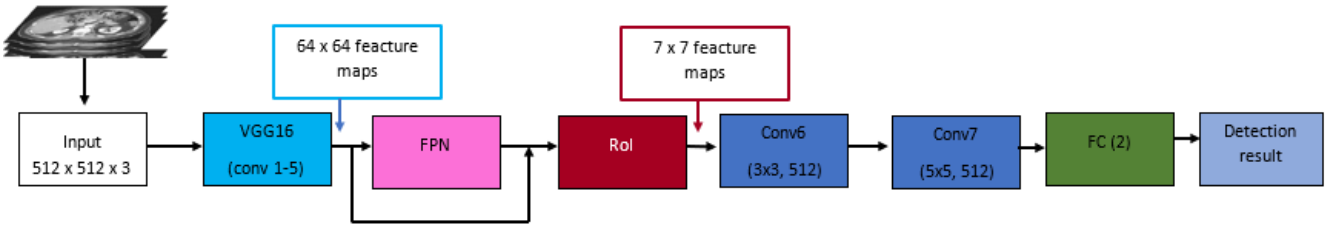

Figura 4.5: Estructura del modelo propuesto por Yan *et. al* [3]

ya que es el método más empleado para duplicar imágenes, y ha sido demostrado experimentalmente que es un método eficiente. Se emplea el método de *resize* para mejorar la detección de lesiones pequeñas y grandes, al cambiar la dimensión de las imágenes a 256 x 256 y 1024 x 1024, respectivamente. Finalmente, se emplea *cropping* para centrar las lesiones grandes.

En este trabajo se propone emplear ResNet y FPN, y partir de dos convoluciones morfológicas; erosión y dilatación. Estas tienen como objetivo reducir el ruido de las tomografías y hacer que las lesiones sean más obvias. Los autores emplean una cascada de 4 R-CNN con distintos umbrales de IoU, como predicción final se considera el *score* máximo. En este trabajo únicamente se consideran las imágenes con lesiones de riñón, este tipo de lesiones no son tan difíciles de detectar como otras, lo cual podría haber ayudado a mejorar las métricas.

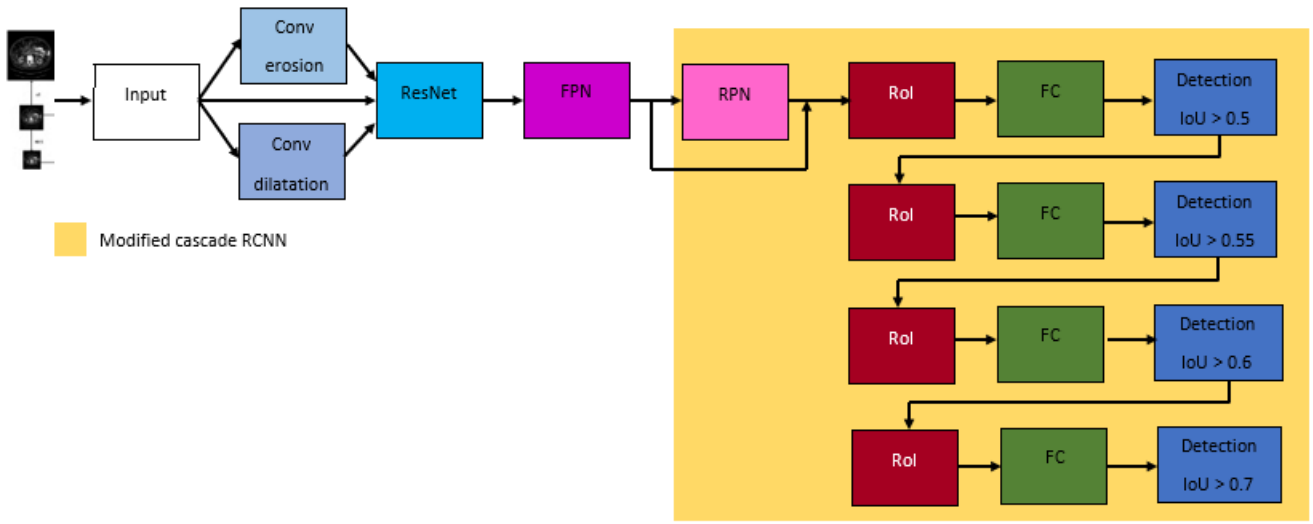

Figura 4.6: Estructura del modelo propuesto por por Zhang *et. al* [4]

La propuesta de Shao *et. al* [5], es una convolución jerárquica dilatada (el *stride* indica la tasa de dilatación). Se emplean ResNet50 como *backbone* y FPN como red de características, se proponen módulos de atención espacial y de canal (*squeeze-and-excitation*) a la salida de cada nivel del FPN. No se menciona el uso de estrategias de aumento de datos.

Se realiza una evaluación por tamaño de lesión, para esto se crean 5 intervalos, con esto buscan afirmar que su modelo detecta mejor los tamaños extremos, es decir, pequeños y grandes. Sin embargo, esto no coincide con las definiciones tradicionales de COCO (*small*, *medium* y *large*), lo cual dificulta que se pueda realizar una comparativa entre sus resultados y otros trabajos. En la figura 4.7 se puede observar la estructura de esta propuesta.

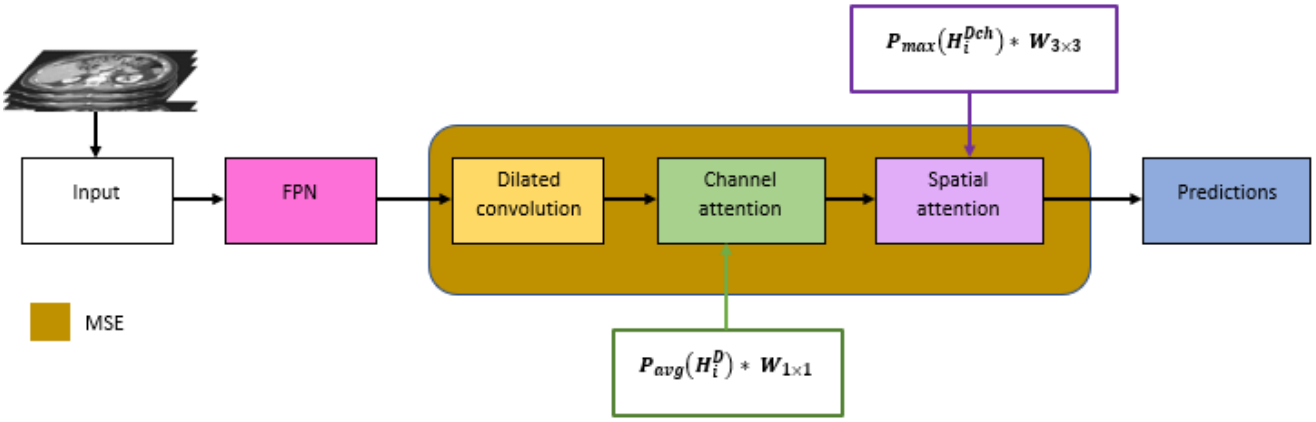

Figura 4.7: Modelo propuesto por Shao *et. al* [5]

Yan *et. al* [6] realiza una tercer propuesta, a diferencia de las dos propuestas previas de Yan *et. al* donde se afirmaba no era necesario el aumento de datos, ahora se realiza de tres formas por cada imagen. Se emplea *random resizing* con una proporción de $0.8 \sim 1.2$, *random translation* de $-8, 8$ píxeles y *3D augmentation*. En este ultimo se realiza tomando cortes vecinos a la que contiene la lesión, ya que deberían de contener la lesión aproximadamente en la misma posición. Cada uno de estos métodos de aumento de datos mejoró la precisión de detección en un $0.2\,\% \sim 0.4\,\%$.

Ésta propuesta consta de una red multitarea de 3 ramas que realizan tareas de detección, etiquetado y segmentación. Se combina información de las capas de detección y etiquetado en una capa de mejora de puntuación. Al momento de generar las etiquetas para la rama de etiquetado, se empelaron los informes radiológicos de *DeepLesion*. Estos informes no son públicos por lo que sería complicado reproducir el modelo. Los autores no muestran los resultados de la rama de detección sin la capa de mejora de puntuación, por lo que no dan a conocer que tanto ayuda a las métricas dicha combinación de información. La estructura de dicha propuesta se muestra en la figura 4.8.

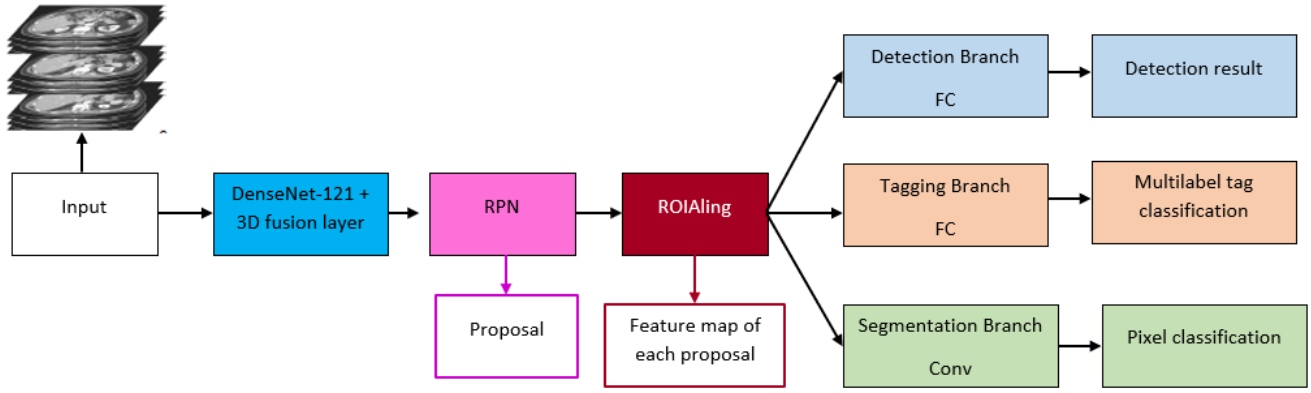

Figura 4.8: Estructura del modelo propuesto por Yan *et. al* [6]

En el trabajo realizado por Li *et. al* [7] se emplea sólo un método de aumento de datos; *horizontal flip*. Se propone tomar 9 imágenes por lesión y realizar fusión de contexto 3D en dos lugares: como se realizaba tradicionalmente (al inicio del modelo) y después del *backbone*. Además, se propone un modulo de atención para capturar información complementaria de 3 vías, cada vía contiene información de 3 imágenes. En esta propuesta afirman que al emplear FPN se mejora la detección de imágenes pequeñas, sin embargo, no se muestran resultados comparativos que lo demuestren. La arquitectura de esta propuesta se muestra en la figura 4.9.

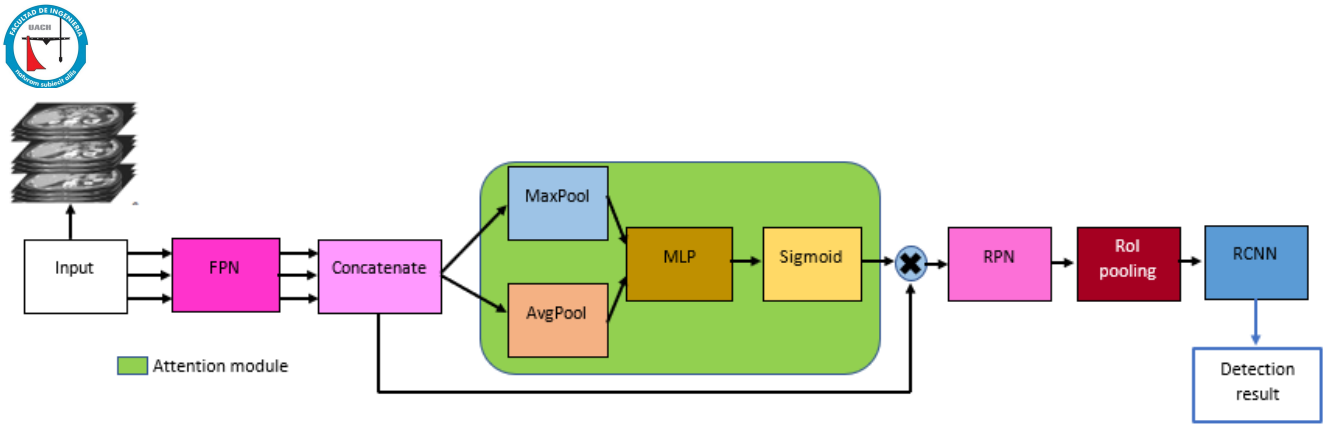

Figura 4.9: Estructura del modelo propuesto por Li *et. al* [7]

En las primeras dos propuestas de Yen *et. al* ni en la propuesta de Shao *et. al* se realizó aumento de datos. Sin embargo, en la tercer propuesta de Yen *et. al* se aplicaron 3 tipos de aumento a cada imagen, y se determinó el impacto de cada uno. Estos métodos junto con las modificaciones de su propuesta mejoraron en un 8 % su ultimo trabajo. En el trabajo de Li *et. al* se emplea únicamente *horizontal flip*, no se detalla la elección del método. Por otro lado, en la propuesta de Zhang *et. al* se emplean 3 métodos de aumento de datos, y se justifica la elección de cada uno de ellos.

Estas propuestas muestran la importancia de emplear algún método de aumento de datos, en uno de ellos se justifica la elección de métodos y en otro se evalúa el impacto de cada método aplicado. Pero en ninguno se analiza a profundidad el impacto de estos métodos en las lesiones y su detección

A continuación, en la Tabla 4.1 se muestra una comparativa de los métodos de aumento de datos, cantidad de *anchors*, *backbone* y equipo del trabajo relacionado. En la Tabla 4.2 se realiza una comparación entre sus aportaciones y problemáticas.

Tabla 4.1: Comparativa de métodos de aumento de datos, cantidad de *anchors*, *backbone* y equipo del trabajo relacionado

Autor	*anchor scales* / *ratios*	*backbone*	Aumento de datos	Equipo
Yen *et. al* 2017 [2]	3 / 3	VGG16	NA	Titan X Maxwell GPU
Yen *et. al* 2018 [3]	5 / 3	VGG16	NA	Titan X Pascal GPU
Zhang *et. al* 2019 [4]	6 / 3	ResNet50	*flipping, cropping* y *resize*	Core i5, NVIDIA GTX 1080
Shao *et. al* 2019 [5]	5 / 3	ResNet50	NA	NA
Yen *et. al* 2019 [6]	5 / 3	DenseNet121	*random resizing*, *random translation* y *3D augmentation*	Tesla V100 GPU
Li *et. al* 2019 [7]	5 / NA	ResNet50	*flip* horizontal	NA

Tabla 4.2: Tabla comparativa de aportaciones y problemas del trabajo relacionado.

Autor	Aportación	Problemas	Resultados (sensitividad)
Yen *et. al* 2017 [2]	Se da a conocer la base de datos Deeplesion. Se propone emplear el modelo *Faster R-CNN*.	El modelo es forzado a omitir detecciones y arrojar sólo 1 lesión por imagen.	64.3 % accuracy
Yen *et. al* 2018 [3]	Se considera el contexto 3D de las tomografías. Se estudian las lesiones por tamaño y región.	Afirman que no es necesario emplear una estrategia de aumento de datos sin demostrarlo.	81.1 % 5FPs
Zhang *et. al* 2019 [4]	Se propone partir de dos convoluciones morfológicas: erosión y dilatación. Se emplea una cascada de 4 R-CNN con distintos umbrales de IoU.	En este trabajo sólo se emplean las tomografías con lesiones de riñón.	83.8 %
Shao *et. al* 2019 [5]	Se propone una convolución jerárquica dilatada. Se propone usar módulos de atención.	Se generan intervalos de tamaños de lesión diferentes al estándar, por lo que no se puede hacer una comparativa directa con otros trabajos.	89 % 4FPs 92 % 8FPs
Yan *et. al* 2019 [6]	Se propone una red para detección, etiquetado y segmentación. Se emplea una capa de mejora de puntuación, para esta se combina información de las ramas de detección y etiquetado.	Emplean información de los informes radiológicos que no es publica. No muestran los resultados obtenidos sólo de la rama de detección sin mejora de puntuación.	86.12 % *average* 94.71 % 8 FPs
Li *et. al* 2019 [7]	Se propone un modelo de 3 vías, donde se vuelve a realizar fusión de contexto después de *backbone*. Emplean módulos de atención para capturar información complementaria de dichas vías.	Mencionan que se emplea FPN para mejorar la detección de imágenes pequeñas, sin embargo, no muestran resultados que demuestren que es verdadero.	3 *slice*: 89 % 9 *slice*: 91.3 % 4FPs

Capítulo 5

Metodología

En esta sección se habla del proceso que se llevó a cabo para realizar la detección de lesiones en imágenes médicas el cual está compuesto por dos secciones: pre-procesamiento y detección de lesiones. En la primera, se realizaron mejoras en el contraste de las imágenes y se emplean técnicas de aumento de datos. Para la detección de lesiones se emplea contexto en 3D y se implementa el modelo *EfficientDet* con la integración de módulos de atención.

En la figura 5.1, se muestra un diagrama a bloques que indica las etapas seguidas para la creación del modelo propuesto. Los bloques azules representan la etapa de pre-procesamiento, y los verdes representan la etapa de detección de lesiones. La descripción de todas estas etapas se desarrolla más adelante, en las secciones 5.1 y 5.2.

5.1. Pre-preocesamiento

Adquisición de *DeepLesion*

La base de datos empleada es *DeepLesion*, cuya descripción se realizó en la sección 4.1. Está compuesta por 56 archivos en formato .zip, estos se encuentran en https://nihcc.app.box.com/v/DeepLesion/folder/51877983116. En total, se compone de $10,825$ carpetas, cada una contiene un estudio diferente, y cada es-

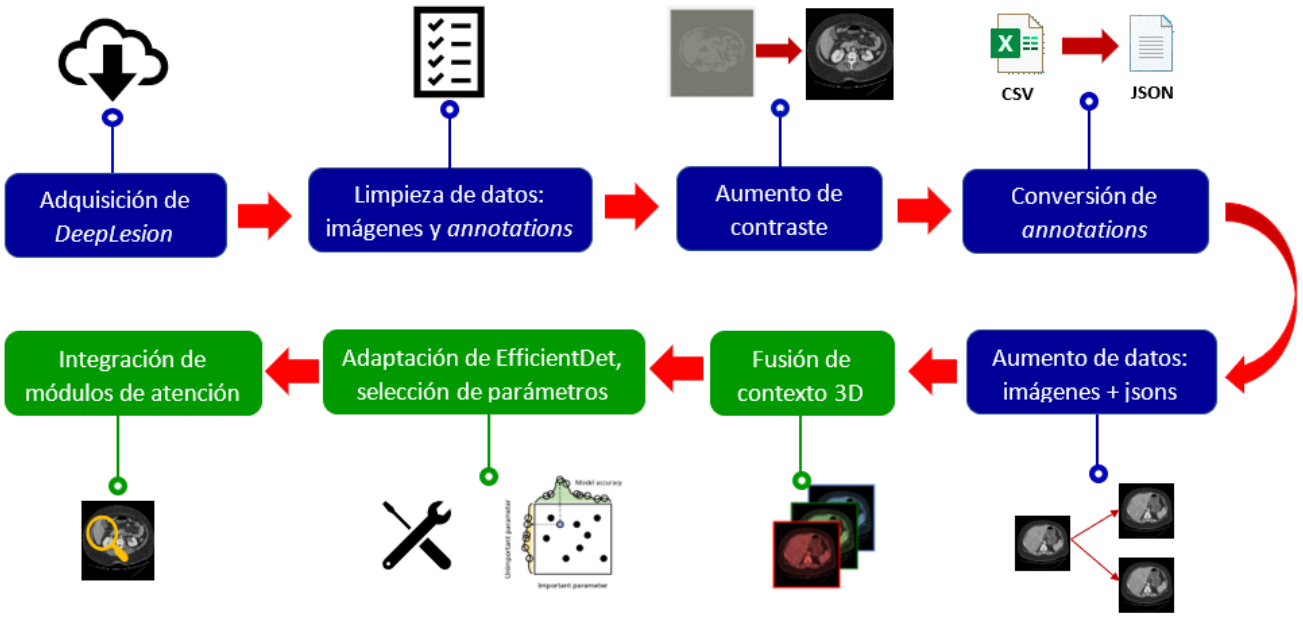

Figura 5.1: Diagrama a bloques del proceso para la implementación del modelo propuesto

tudio está compuesto por varias imágenes en formato .png.

Limpieza de datos

Algunos de estos estudios presentaban uno de tres posibles problemas, por lo cual fueron eliminados 44 de ellos. Las especificaciones de éstos son las siguientes:

- Se contaba con el *annotation*, pero no con las imágenes del estudio 001535_02_01.
- Los autores de la base de datos etiquetaron 35 estudios como *possibly noisy*, lo que indica posible ruido en las imágenes de dichos estudios.
- Algunos de los *bounding boxes* se salían del margen de las imágenes, es decir, del intervalo $[0, 512]$. Por este motivo, se eliminaron 8 estudios.

Aumento de contraste

Las imágenes de esta base de datos presentan un nivel de contraste bajo, ejemplo de ello son las imágenes que muestran en la figura 5.2.

Figura 5.2: Ejemplos de imágenes de la base de datos *DeepLesion*

Para mejorar el contraste de las imágenes, se siguieron dos etapas. Primero, se obtuvo la *Hounsfield unit* sustrayendo $32,768$ de la intensidad de cada píxel en cada imagen. En la segunda etapa se hizo *intensity windowing*, es decir, se aplicó la formula (5.1) a cada imagen:

$$I = min(255, max(0, (HU - A)/(B - A) * 255)) \tag{5.1}$$

donde:

HU es la unidad de Hounsfield,

A y B los extremos de la ventana para la imagen correspondiente.

Los valores de A, B y $32,768$, son proporcionados por los autores de la base de datos. Especialmente, los valores A y B, están contenidos en la columna *DICOM_window* [1] del archivo de excel, que contiene la información de la base de datos. Algunos ejemplos de las imágenes resultantes, después de aplicar los pasos mencionados, se muestran en la figura 5.3. En comparativa contra las imágenes originales (figura 5.2, se pude apreciar una mejora considerable, ahora es posible distinguir huesos, órganos y tejidos blandos.

Conversión de *annotations*

Las *annotations* se obtuvieron en formato .csv con la siguiente información: carpeta del estudio, el nombre de la imagen donde se detectó la lesión, las dimensiones de *bounding box*, los diámetros de la lesión,

[1] DICOM es el estándar internacional para transmitir, almacenar, recuperar, imprimir, procesar y mostrar información de imágenes médicas

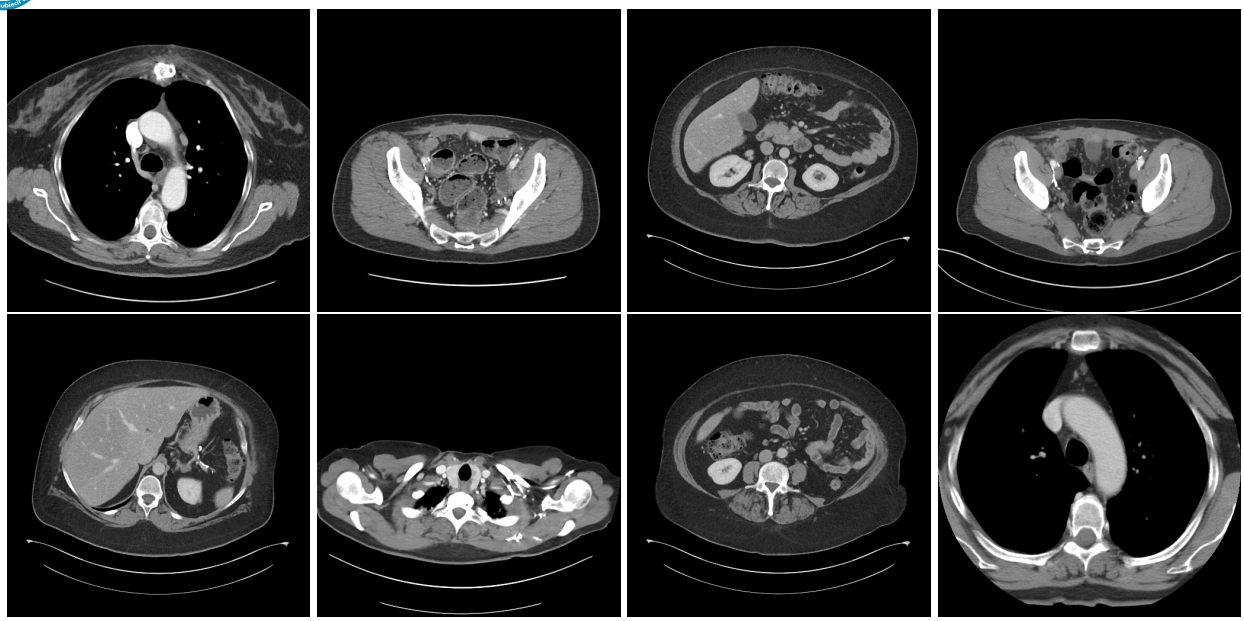

Figura 5.3: Ejemplos de imágenes después del proceso de mejora de contraste

la región de la lesión (para aproximadamente el 30 % de las lesiones), posibilidad de ruido en la muestra, dimensiones de las imágenes, los parámetros DICOM, género y edad del paciente, y a que conjunto pertenece cada muestra (*train/val/test*).

Para trabajar con *EfficientDet* se crearon 3 archivos .json para cada variante implementada. Estos archivos se generaron con la estructura de la figura 5.4.

```
datos = { }

licenses{"url": 'None', "id": 1, "name": 'All rights reserved by xxx'}
info{'description': 'NIH DeepLesion Dataset.', 'url': 'http://www.xxx.com', 'version': 'deeplesion micaai',
                    'year': 2018, 'contributor': 'NIH', 'date_created': '2019.01.10'}
categories{'supercategory': 'DeepLesion', 'id': 1, 'name': 'Lesion'}
datos['annotations'] = []
datos['images'] = []


annotations{'area': area, 'bbox': bbox, 'category_id': category_id, 'id': id_, 'image_id': image_id,
            'iscrowd': iscrowd, 'noisy': noisy, 'segmentation': segmentation}

images{ 'file_name': file_name, 'height': height, 'width' : width,  'slice_intv' : slice_intv, 'slice_no':slice_no,
        'spacing':spacing, 'windows' : windows,  'z_position' : z_position, 'id' :  image_id }
```

Figura 5.4: Estructura de las *annotations* de *DeepLesion*

Aumento de datos

Para el aumento de datos se establecieron dos estrategias: *flip* horizontal y *Cut-and-Paste*. La principal razón para esto, fue analizar el impacto del aumento de datos al modificar el contexto inmediato a las lesiones. Esto es posible ya que con *flip* horizontal se conserva, mientras que con *Cut-and-Paste* se modifica. En segundo lugar, este análisis también permite comparar *flip horizontal*, la estrategia más común al realizar aumento de datos con imágenes médicas [4], [7], y *Cut-and-Paste*, una estrategia reciente que ha sido empleada con éxito, en tareas de segmentación y clasificación [49], [50].

1. ***Flip* horizontal**

   Aleatoriamente, al 50 % de las imágenes se les aplicó un aumento de datos *flip* horizontal, es decir, se reflejaron los píxeles respecto al eje y. Este aumento de datos se ve como se muestra en la figura 5.5.

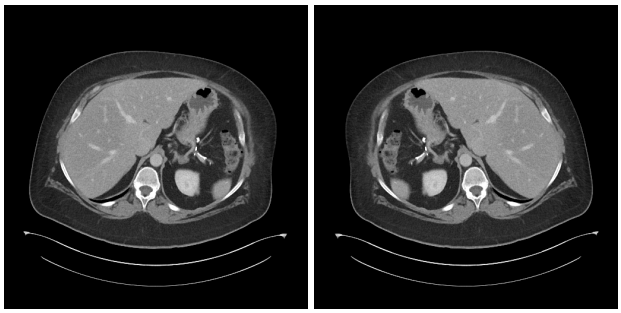

Figura 5.5: La imagen original se encuentra a la izquierda, y la imagen resultante de un *flip* horizontal, a la derecha.

   En tareas de clasificación, este procedimiento no requiere algo más. Sin embargo, para la tarea de detección, se requiere modificar los valores del *bounding box* que indica la posición de la lesión. Para esto se requieren las coordenadas $(x_1, y_1), (x_2, y_2)$ originales, y el número de columnas de la imagen, es decir, el ancho de la imagen. Estos elementos se indican en la figura 5.6. El procedimiento para encontrar las nuevas coordenadas es el siguiente:

   - Se obtienen las dimensiones del *bounding box*:

$$w = x_2 - x_1$$

$$h = y_2 - y_1$$

- Se obtiene la distancia entre el *bounding box* y el extremo derecho de la imagen.

$$d = cols - x_2$$

- A partir de lo anterior, se obtienen las nuevas coordenadas para el *bounding box* resultante del *flip* horizontal.

$$x_1^{new} = d$$

$$x_2^{new} = x_1^{new} + w$$

$$y_1^{new} = y_1$$

$$y_2^{new} = y_2$$

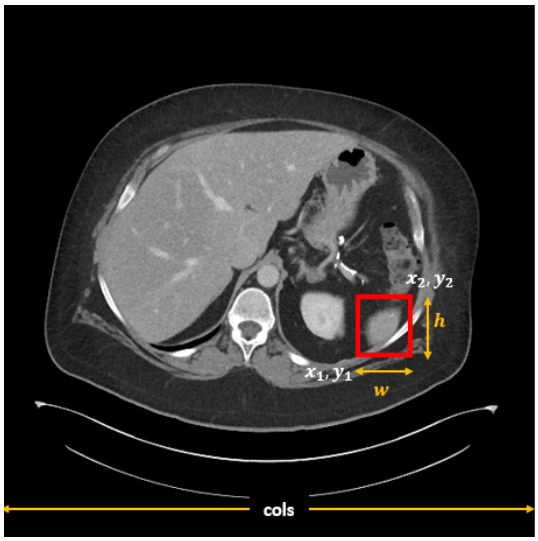

Figura 5.6: Dimensiones consideradas para modificar las coordenadas de un *bounding box*, después de aplicar un *flip* horizontal.

2. ***Cut and paste***

De manera simplificada, este método consiste en tomar el espacio del *bounding box* que contiene la lesión y colocarla en otro lugar de la imagen. El proceso que se siguió se describe a continuación:

- Por cada imagen se localizan las coordenadas del *bounding box* (x_1, y_1, x_2, y_2) que encierran la lesión (imagen 5.7 (a)).
- Se establecen tres posibles movimientos para las lesiones; movimiento en x, movimiento en y y movimiento en x, y. Esto de tal manera que las nuevas posiciones de las lesiones no queden fuera de la región del cuerpo (imágenes 5.7 (b)-(d)).
- Cada imagen tiene una probabilidad de 40 % de moverse en x, 40 % de moverse en y y 20 % de moverse en x, y.
- De acuerdo al tipo de movimiento, se realiza el movimiento de píxeles, la posición original se cubre con los píxeles de la posición destino. Además, se generan las *annotations* correspondientes. Las nuevas coordenadas están dadas por:

  a) Movimiento en x

  $$x_1^{new} = cols - x_2$$
  $$x_2^{new} = cols - x_1$$
  $$y_1^{new} = y_1$$
  $$y_2^{new} = y_2$$

  b) Movimiento en y

  $$x_1^{new} = x_1$$
  $$x_2^{new} = x_2$$
  $$y_1^{new} = rows - y_2$$
  $$y_2^{new} = rows - y_1$$

  c) Movimiento en x, y

  $$x_1^{new} = cols - x_2$$
  $$x_2^{new} = cols - x_1$$
  $$y_1^{new} = rows - y_2$$

$$y_2^{new} = rows - y_1$$

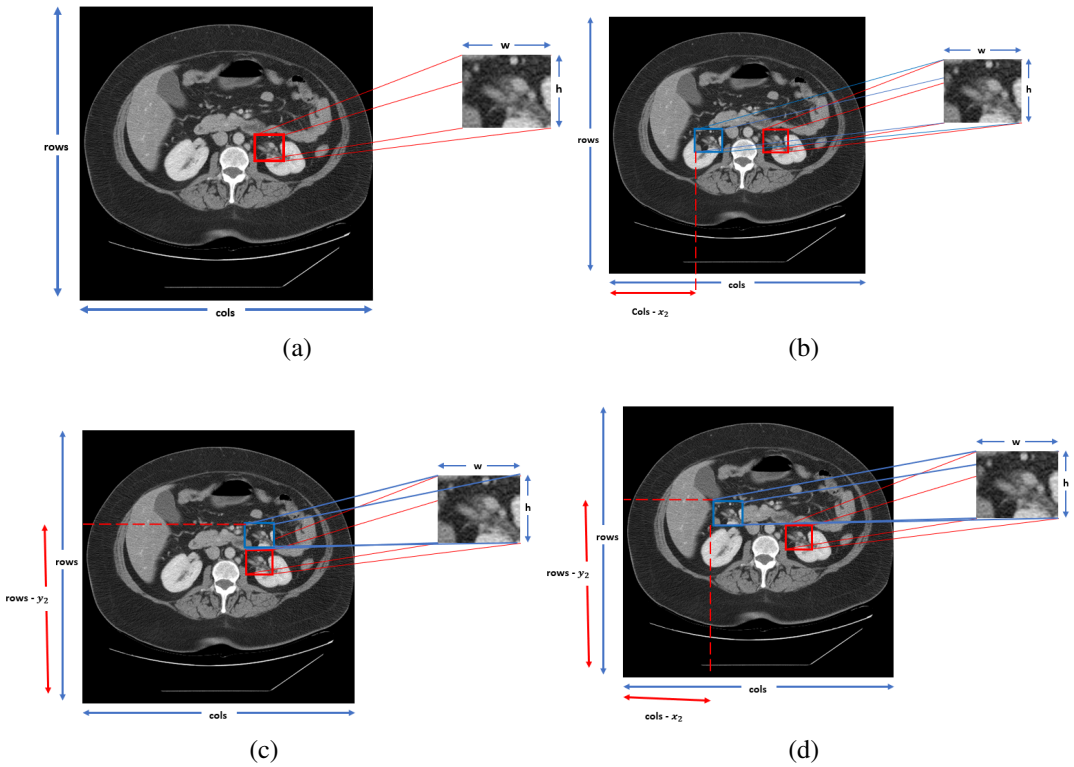

Figura 5.7: (a): lesión localizada, (b): lesión con movimiento en x, (c): lesión con movimiento en y, (d): lesión con movimiento en x, y. En rojo, se muestra la posición de la lesión original; en azul, se muestran las posiciones de las nuevas lesiones.

5.2. Detección de lesiones

Al detector *EfficientDet* se le integraron dos elementos: fusión de contexto 3D y módulos de atención. En esta sección se presentan tres secciones en el siguiente orden: fusión de contexto 3D, detector de objetos y módulos de atención. Este orden se basa en el orden de su aplicación, la fusión ocurre al inicio al estar generando el *input* previo a la primer convolución del detector, mientras que los módulos de atención se encuentran al inicio y final del modelo.

Fusión de contexto 3D

Cada estudio CT está compuesto por un conjunto $\{x_1, x_2, ..., x_n\}$ de n cortes o imágenes, el valor de n puede ser diferente para distintos estudios. La posición de la lesión varía en el volumen de cortes, es decir, puede estar en los cortes extremos x_1 o x_n, o en cualquier corte intermedio x_{n-i}.

Lo anterior es relevante, ya que para realizar la fusión de contexto son necesarias 2 imágenes vecinas más próximas a la imagen con lesión. El procedimiento fue el siguiente:

1. Dada la imagen del corte x_i que contiene la lesión, se tomaron los cortes anterior x_{i-1} y posterior x_{i+1}.

2. Si los cortes $x_{i-1} = x_1$ o $x_{i+1} = x_n$, entonces, $x_{i-1} = x_i$ o $x_{i+1} = x_i$. Es decir, si los cortes con la lesión se encuentra en los extremos del CT, sólo tendría un vecino, por lo que se repite el corte con lesión.

3. Se fusionan los 3 cortes, concatenando las imágenes en orden x_{i-1}, x_i, x_{i+1}.

A continuación, en la figura 5.8, se muestra de manera gráfica el proceso de la fusión.

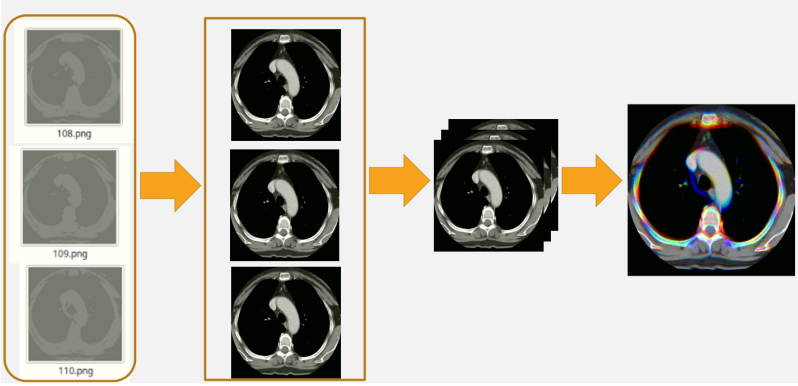

Figura 5.8: Proceso para llevar a cabo la fusión 3D

En la imagen 5.9 se muestran algunos ejemplos de las imágenes resultantes después de aplicar el procedimiento de fusión de contexto 3D.

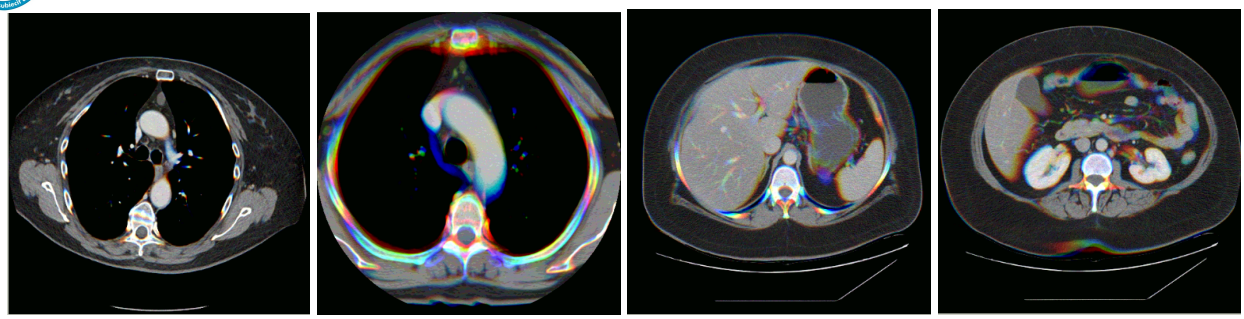

Figura 5.9: Ejemplos de imágenes después del proceso de fusión de contexto 3D

Detector de objetos

Se parte de un conjunto de $32,724$ imágenes. Al cargarlas, se aseguró que las dimensiones fueran de 512×512, en caso de tener alguna imágenes de otro tamaño se aplicó un *resizer*. Luego, se implementa una estrategia de aumento de datos como se describó en la sección anterior. Estás imágenes, que representan matrices de tamaño $512 \times 512 \times 3$ son normalizadas al intervalo $[0, 1]$, mediante la siguiente ecuación

$$x_{new} = \frac{x_i - x_{min}}{x_{max} - x_{min}} \tag{5.2}$$

Posteriormente, se realiza otra normalización a nivel de canal, esto mediante la ecuación 5.3. Los valores de media para los canales RGB son $[0.485, 0.456, 0.406]$, y para desviación estándar son $[0.229, 0.224, 0.225]$.

$$x_{new} = \frac{x_i - \mu_{ch}}{\sigma_{ch}} \tag{5.3}$$

Para utilizar el modelo de *EfficientDet* se requiere que las etiquetas de *annotations* tengan un formato x, y, w, h, por tal motivo, fue necesario aplicar una transformación ya que originalmente se tenían de la forma x_1, y_1, x_2, y_2. El primer formato se refiere a las coordenadas del centro y dimensiones, y el segundo a las coordenadas de los extremos izquierdo inferior y derecho superior. Esta transformación se logró empleando geometría básica, mediante las siguientes ecuaciones:

$$x = x_1 + \frac{x_2 - x_1}{2} \tag{5.4}$$

$$y = y_1 + \frac{y_2 - y_1}{2} \tag{5.5}$$

$$w = x_2 - x_1 \tag{5.6}$$

$$h = y_2 - y_1 \tag{5.7}$$

Se establecieron 4 *aspect ratios* y 3 *anchor ratios*, definidos con $[2, 4, 8, 16]$ y $[(1, 2), (1, 1), (2, 1)]$, respectivamente. Para hacer esta elección se hizo un *random search*, con base en parámetros propuestos en trabajos previos que han empleado la base de datos *DeepLesion*. Más adelante, se realizó otro *random search*[2] para determinar el mejor optimizador y *lr scheduler*.

En la Tabla 5.1 se muestran las diferentes configuraciones del optimizador que se probaron, se puede observar que el optimizador Adam arroja los mejores valores de pérdida (los más bajos), para las tareas de clasificación y regresión. En la Tabla 5.2, se muestran las configuraciones para el *learning rate scheduler*, es claro que con *ReduceLROnPlateau* (Plateau) se obtienen los mejores resultados para clasificación y regresión.

Tabla 5.1: Pérdida de entrenamiento empleando los optimizadores Adam, SGD y Nesterov

Optimizador	Adam	SGD	Nesterov
cls	0.457	0.789	0.462
reg	0.302	0.592	0.357

Tabla 5.2: Pérdida de entrenamiento empleando diferentes *learning rate scheduler*

lr scheduler	Plateau	Cosine
cls	0.4577	0.727
reg	0.302	0.592

Como *backbone* del modelo se empleó *EfficientNet*, sin el último bloque de convolución ni la capa FC. Esto permite una detección a 5 niveles. Lo que implica la generación de (3 *anchors scales*)(3 *anchors ratios*)(5 niveles) = 60 *anchors*. La estructura de *EfficientNet* permite ir reduciendo el tamaño de los

[2]Para la búsqueda de los mejores parámetros se consideraron 10 épocas.

mapas de características y aumentando la profundidad. Esto se puede visualizar en la figura 5.10, donde se indican las dimensiones ancho, largo y profundidad de los mapas de características en cada nivel.

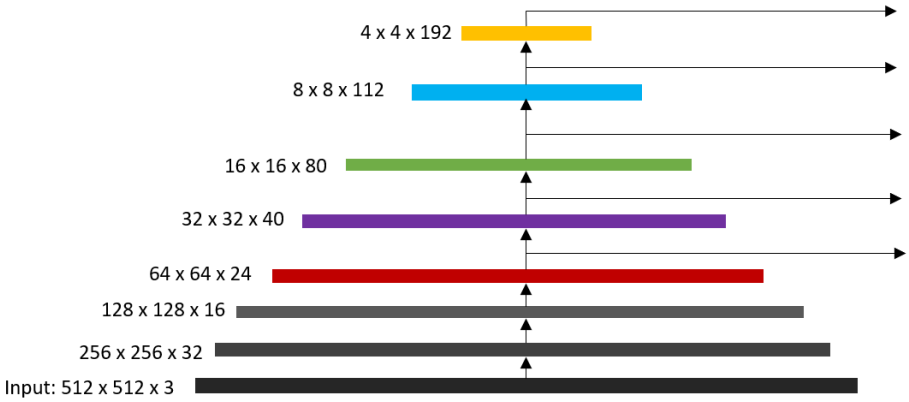

Figura 5.10: Estructura del *backbone* EfficientNet

En la figura 5.11, se muestra la estructura de los bloques que conforman el *backbone*. Los bloques de color indican la correspondencia de los niveles que alimentan la red de características. En cada bloque se indican los tamaños de *kernel*, donde sólo se consideraron tamaños de 3×3 y 5×5. La cantidad de *kernels* para obtener dichos bloques son: $24, 40, 80, 112, 192$. Los bloques indicados como MBConv1 y MBConv6, están compuestos por un par de convoluciones y un bloque de *squeeze-and-excitation*. Esto se muestra detalladamente en las figuras 5.12 y 5.13.

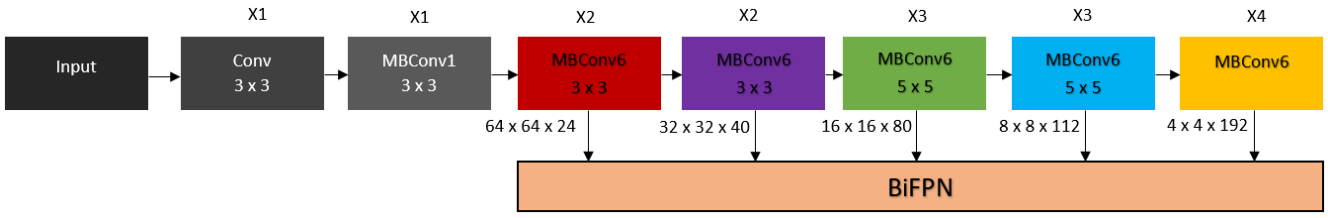

Figura 5.11: Diagrama del *backbone* EfficientNet

MBConv1, se compone de una convolución en profundidad, seguida de la función de activación *swich*. Se tiene un módulo *squeeze-and-excitation*, y otra capa de convolución tradicional. La salida de ésta se suma con la entrada original y ese es el resultado del bloque.

MBConv6, es similar a MBConv1, sólo que aquí se tiene una capa convolucional extra, seguida de nuevo de una función de activación *swish*. Además, se realiza un *dropout* antes de la suma entre la última convolución y la entrada.

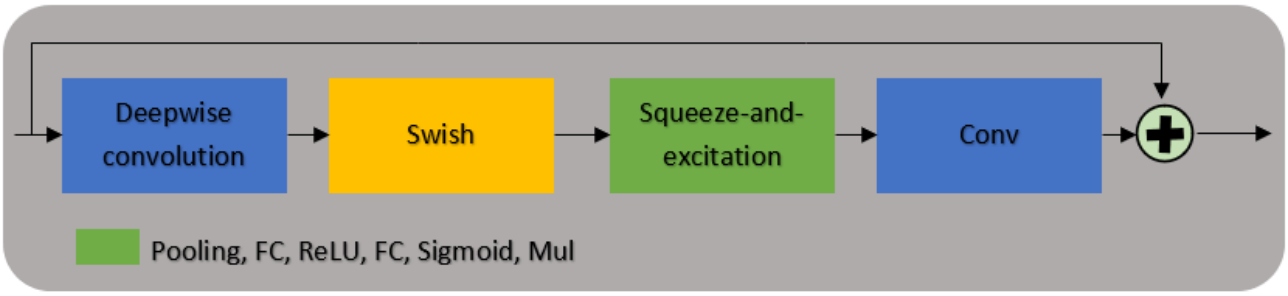

Figura 5.12: Estructura de MBConv1

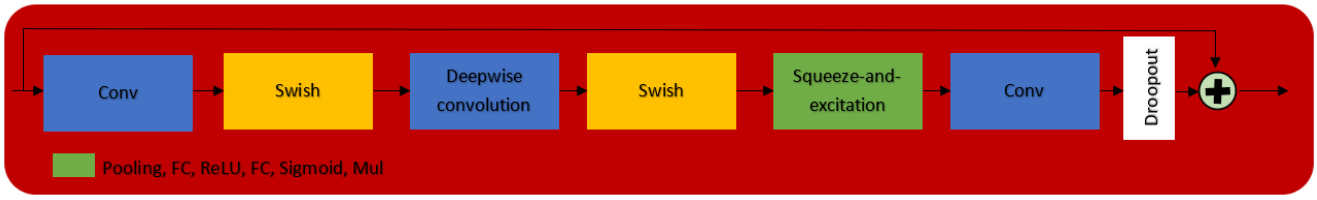

Figura 5.13: Estructura de MBConv6

Lo anterior se sigue de una red de características, compuesta por tres capas BiFPN. Cada BiFPN se compone de una serie de capas de características, representadas por nodos. Los nodos de los niveles superiores contienen características semánticamente fuertes. Mientras que los nodos de los niveles inferiores, contienen características con alta resolución.

En la figura 5.14, se muestra la arquitectura de esta red. Cada color representa el nivel correspondiente a los indicados en el *backbone*, la resolución se mantiene constante en cada fila. Se emplean convoluciones tradicionales con función de activación ReLU, *same padding* para las conexiones representadas por flechas negras, y *batch normalization*. Las flechas azules, representan un *up sample*, mientras que las flechas rojas representan un *down sample*. En ambos casos se emplea la función de activación *swish*. Para los nodos hacia los que se dirigen 2 o 3 flechas, se realiza una fusión de características, para esto se

emplea la función dada por la ecuación 5.8.

$$O = \sum_i \frac{ReLU(\omega_i)}{\epsilon + \sum_j \omega_j} \cdot I_i \tag{5.8}$$

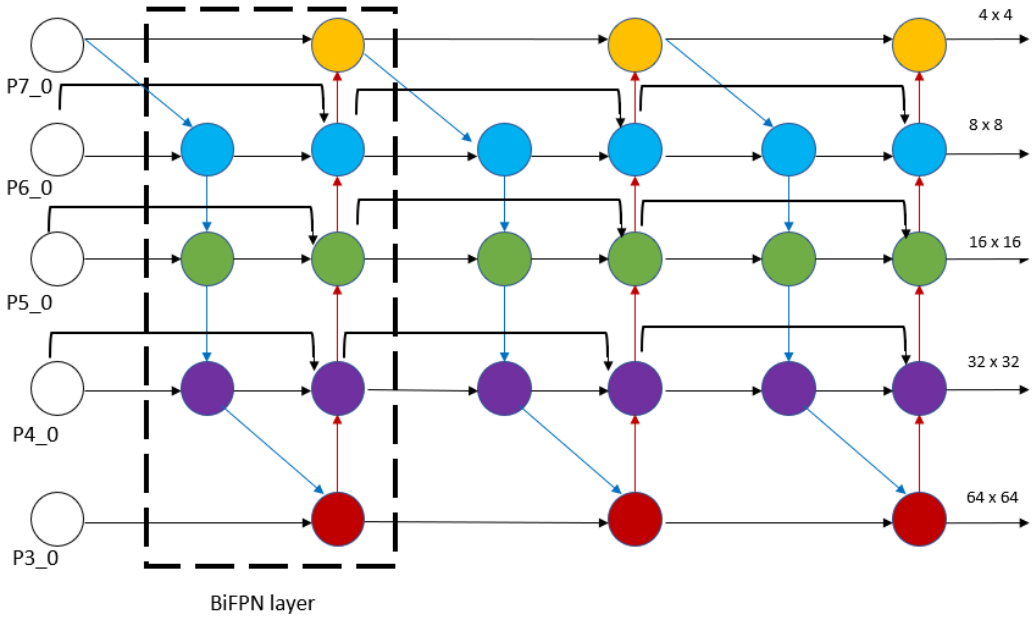

Figura 5.14: Estructura de la red de características con 3 BiFPN

Por ejemplo, en la figura 5.15 se consideran 4 nodos para explicar como ocurren las fusiones de 3 conexiones. La fusión de características ocurre en el nodo P5_2, y está dada por la siguiente expresión:

$$O = \frac{ReLU(P5\_\omega_2)}{\sum P5\_\omega_2 + 0.001} \cdot [P5\_0 + P5\_1 + Resize(P4\_2)]$$

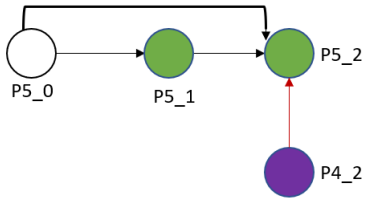

Figura 5.15: Ejemplo de fusión

Las salidas de esta red de características, ingresan de manera compartida a la red de clasificación y la red de predicción de *box*. La estructura de la red de clasificación se muestra en la figura 5.16, la cual

está compuesta por 3 bloques de una convolución *deepwise*, seguida por una convolución *pointwise*. La primera emplea *kernels* de tamaño 3×3, mientras que la segunda de tamaño 1×1. Se hace *batch normalization*. Se emplea la función de activación *swish* entre las capas de convolución, y al final se utiliza la función *sigmoid* que arroja las clasificaciones.

Por otro lado, la red de predicción de *box* tiene las mismas capas de convolución de la figura 5.16, sólo que no se emplea la función de activación *sigmoid*. En todas las capas se emplea la función *swish*, y se aplica *batch normalization*. La diferencia entre las capas para clasificación y *box*, son las dimensiones. Para clasificación se obtiene una salida de tamaño *anchors* $\times$ *clases*, y para los *bounding box* de un tamaño de [3]*4* $\times$ *anchors*.

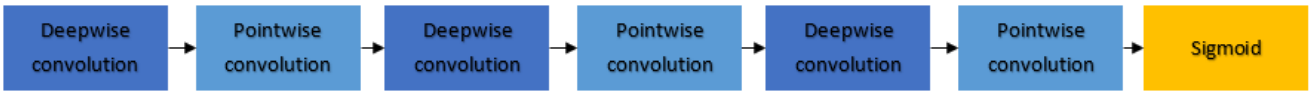

Figura 5.16: Red de clasificación

Para el entrenamiento, se emplearon las funciones de pérdida *focal loss* y $smooth_L 1$. Como optimizador se utilizó descenso de gradiente estocástico con *momentum* de 0.9 y se partió con un *learning rate* de 2e-4.

Para evitar detecciones duplicadas de la misma lesión, se empleó *non-max suppresion*. Donde, dadas las detecciones con mayor probabilidad, se calculan los IoU de las detecciones con probabilidad inferior, y se descartan si el IoU es grande.

Módulos de atención

Se integraron módulos de atención en dos partes del modelo. Se agregaron módulos mixtos, es decir, espaciales y de canal en los cinco niveles de detección del *backbone*. Por otra parte, se agregaron módulos de atención espacial en los bloques de convolución de la red de clasificación.

[3]El 4 se debe a que se predicen 4 elementos (x, y, w, h)

En la figura 5.17, se muestra en los recuadros verdes, la disposición de los cinco módulos de atención espacial y de canal, estos se describen a detalle en las figuras 5.19 y 5.20.

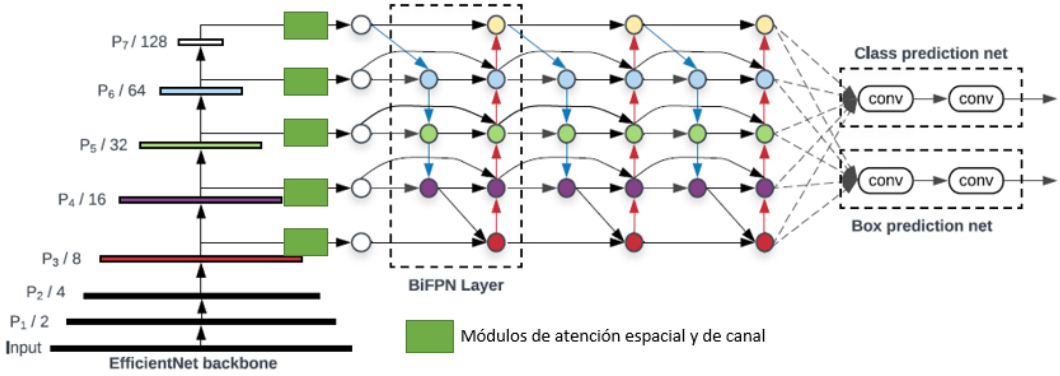

Figura 5.17: Módulos de atención de canal en la red de características.

En la figura 5.18, se muestra en los recuadros naranjas, la posición de los módulos de atención de canal en la red de clasificación. Este módulo se muestran detalladamente en la figura 5.19.

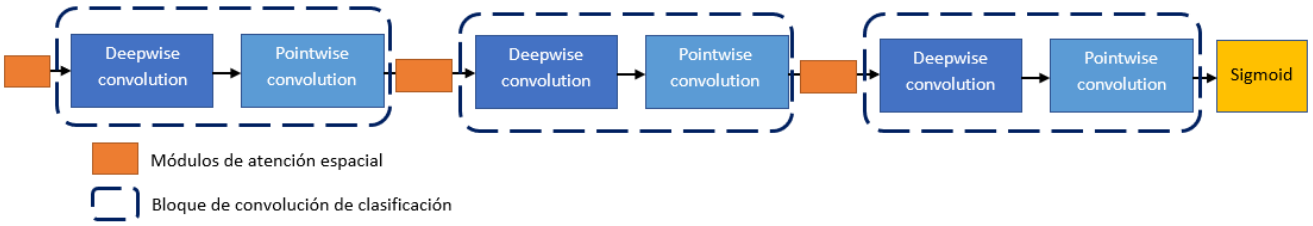

Figura 5.18: Módulos de atención de canal en la red de clasificación.

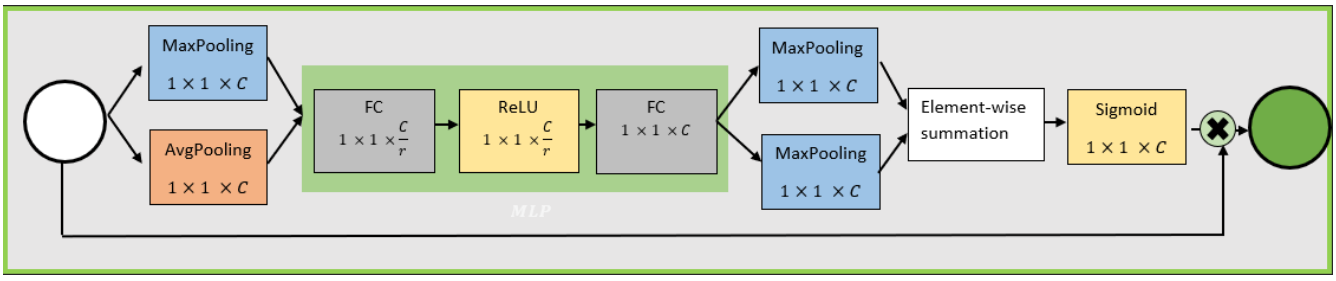

Figura 5.19: Representación de un módulo de atención de canal.

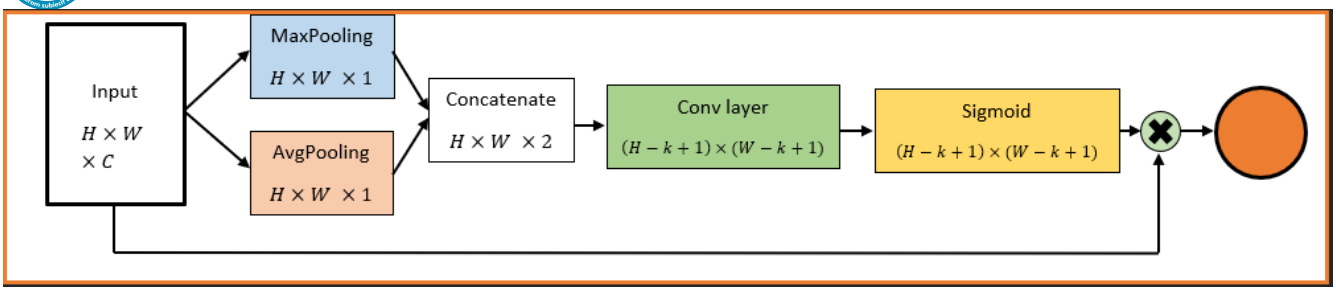

Figura 5.20: Representación de un módulo de atención espacial.

Capítulo 6

Experimentos

En este capitulo, se presenta la configuración experimental de las distintas implementaciones que se llevaron a cabo con *EfficientDet* y con sus distintos niveles. Se muestran los resultados obtenidos y se realiza un análisis cuantitativo y cualitativo.

6.1. Configuración experimental

Durante la experimentación con el modelo *EfficientDet*, se evaluaros distintas variantes orientadas a resolver los objetivos específicos, establecidos en el capítulo 1. Se desarrollan distintas propuestas para abordar los primeros 3 objetivos relacionados al aumento de datos: determinar si es necesario al emplear *DeepLesion*, precisar la relevancia del contexto de la lesión, y establecer la mejor estrategia para ello. Mediante la evaluación de los distintos niveles del modelo se estudian los objetivos 4 y 5, orientados a comparar los distintos niveles de *EfficientDet*, y la diferencia al trabajar con imágenes médicas en lugar de imágenes naturales. Finalmente, se incorporan estrategias de atención para llegar al objetivo 6.

De manera más detallada, en la tabla 6.1 se muestran la descripción de cada propuesta, y el propósito por el cual se llevaron a cabo.

Tabla 6.1: Propósito de las diferentes implementaciones realizadas empleando *EfficientDet*

Modelo	Propósito
EfficientDet-D0	Obtener los resultados de un modelo base que sirva de comparativa.
EfficientDet-D0 + *Flip* horizontal	Analizar el impacto del aumento de datos cuando se conserva el contexto inmediato a la lesión.
EfficientDet-D0 + *Cut-and-Paste*	Estudiar el impacto del aumento de datos cuando no se conserva el contexto inmediato a la lesión.
EfficientDet-D0 + *Cut-and-Paste* + *Flip* horizontal	Corroborar que el método de flip horizontal no realiza un aporte adicional al modelo, aparte del que ya realiza el método cut-and-pasate.
EfficientDet-D0 + *Cut-and-Paste* + Módulos	Medir el impacto en el aprendizaje del modelo, de la combinación de aumento de datos *Cut-and-Paste* y módulos de atención.
EfficientDet-D0 + Flip horizontal + Módulos	Identificar si hay mejora en el modelo, a partir de la combinación de aumento de datos flip horizontal y módulos de atención.
EfficientDet-D0 + Módulos + *Cut-and-Paste* (small, abdomen, pelvis)	Centra el aumento de datos en las lesiones más difíciles de detectar. Las small por su tamaño, y las lesiones de abdomen y pelvis ya que son difíciles de distinguir de su entorno
EfficientDet-D0 + Módulos + *Cut-and-Paste* (hueso, abdomen, tejidos blandos, pelvis, small) x1	Centrar el aumento de datos en las lesiones más difíciles de detectar y de las cuales se tienen pocas muestras. Es decir, en las lesiones mencionadas en el modelo anterior y adicionalmente en las muestras de hueso y tejidos blandos.
EfficientDet-D0 + Módulos + *Cut-and-Paste* (hueso, abdomen, tejidos blandos, pelvis, small) x3	Estudiar el aumento de datos centrado en las lesiones más difíciles y las de pocas muestras, pero generando 3 imágenes nuevas por cada lesión.
EfficientDet-D0 + Módulos + *Flip* horizontal + *Cut-and-Paste* (huesos, abdomen, tejidos blandos, pelvis, small)	Analizar la estrategia del modelo anterior, pero considerando además el aumento de datos flip horizontal.
EfficientDet-D1-D6 + Módulos + *Cut-and-Paste* (general)	Analizar el impacto de los distintos niveles de *EfficientDet*, como impacta en la sensitividad tener modelos cada vez más profundos.

6.2. Resultados

En la Tabla 6.2 se muestran de manera descendente los resultados de las diversas implementaciones que se llevaron a cabo, la métrica empleada es *sensitividad*. Se evaluaron las distintas configuraciones de *EfficientDet*-D0, planteadas previamente en la Tabla 6.1.

De acuerdo con los resultados obtenidos en la Tabla 6.2, la mejor configuración es *EfficientDet* con módu-

Tabla 6.2: Sensitividad de las distintas implementaciones con *EfficientDet*-D0

Modelo	**FPs**	**Sensitividad**
EfficientDet-D0 + *Cut-and-Paste* + Módulos	7	73 %
EfficientDet-D0 + *Cut-and-Paste*	7	70 %
EfficientDet-D0 + *Cut-and-Paste* + *Flip horizontal*	7	70 %
EfficientDet-D0 + *Flip horizontal*	4	68 %
EfficientDet-D0 + Módulos + *Cut-and-Paste* (hueso, abdomen, tejidos blandos, pelvis, small) x3	11	67 %
EfficientDet-D0 + Módulos + *Cut-and-Paste* (hueso, abdomen, tejidos blandos, pelvis, small) x1	6	66 %
EfficientDet-D0 + *Flip horizontal* + Módulos	3	63 %
EfficientDet-D0 + Módulos + *Cut-and-Paste* (small, abdomen, pelvis)	3	60 %
EfficientDet-D0 + Módulos + *Flip horizontal* + *Cut-and-Paste* (huesos, abdomen, tejidos blandos, pelvis, small) —	2.5	55 %
EfficientDet-D0	5	52 %

los de atención (a la salida de la red de características y en la red de clasificación), y aumento de datos *Cut-and-Paste* en todas las imágenes.

En la tabla 6.3 se muestran los resultados para dicha configuración, para los distintos niveles de *EfficientDet*-D0 a *EfficientDet*-D6. Se muestran los mejores resultados de cada nivel, y la cantidad de épocas de entrenamiento y FPs requeridos para obtener dichos resultados. Cada modelo fue entrenado durante un rango de 10-15 épocas, dependiendo de su evolución.

Tabla 6.3: Sensitividad para los distintos niveles de *EfficientDet*-D0 a *EfficientDet*-D6

Nivel	**Sensitividad**	**FPs**	**Épocas**
D0	73 %	7	8
D1	50 %	2.5	7
D2	48 %	14	5
D3	79 %	25	5
D4	57 %	26	5
D5	77 %	31	2
D6	43 %	2	7

En la tabla 6.4 se muestra el porcentaje de recall (sensitividad) respecto a los distintos tamaños de lesión

(*small, medium* y *large*), para los niveles de *EfficientDet*-D0 a *EfficientDet*-D6.

Tabla 6.4: Porcentaje de *recall* (sensitividad) por tamaño de lesión, en la etapa de entrenamiento para los distintos niveles de *EfficientDet*.

Nivel	***small***	***medium***	***large***
D0	31.7 %	36.1 %	47.1 %
D1	34 %	21.9 %	7.3 %
D2	25.4 %	18.9 %	7.1 %
D3	48.5 %	32.8 %	8.8 %
D4	30.6 %	13.1 %	5.1 %
D5	45.3 %	34.3 %	21.6 %
D6	30 %	23.3 %	7.7 %

En la tabla 6.5 se muestran porcentajes de sensitividad y cantidad de FPs, de la comparativa del estado del arte y el modelo propuesto en esta investigación. Este ultimo logra superar al trabajo de Cai *et. al* [12] en un 0.4 %, empleando 1 FPs menos.

Tabla 6.5: Comparativa de sensitividad entre el estado del arte y el modelo propuesto

Autor	**FPs**	**Sensitividad**
Yan et. al (2019) [6]	8	94.71 %
Shao et. al (2019) [5]	8	92 %
Li et. al (2019) [7]	4	91.3 %
Zhang et. al (2019) [4]	NE	83.8 %
Yen et. al (2018) [3]	5	81.1 %
Nuestra propuesta	**7**	**73 %**
Cai et. al (2020) [12]	8	72.6 %

6.3. Análisis de resultados

En esta sección se presenta un análisis cuantitativo de los resultados obtenidos en la sección de resultados. Posteriormente, se sigue un análisis cualitativo de las detecciones y visualizaciones de las activaciones, de las primeras capas del modelo.

6.3.1. Análisis cuantitativo

En la tabla 6.2 se presentaron los porcentajes de sensitividad para las experimentaciones realizadas con *EfficientDet*-D0. El mejor resultado logró un 73 % de sensitividad con 7 FPs, empleando el método de aumento de datos *cut-and-paste* y módulos de atención. Esto es 21 % superior al resultado obtenido sólo con *EfficientDet*-D0.

Sin utilizar módulos de atención, al emplear el aumento de datos *flip* horizontal se obtuvo un 68 % de sensitividad, mientras que con *cut-and-paste* un 70 %. Lo cual indica que modificar el contexto inmediato a la lesión, le permite al modelo aprender una mayor generalización.

Por otro lado, empleando módulos de atención y las distintas configuraciones de aumento de datos, se obtuvo una sensitividad de 60 % al aumentar los datos de dos regiones y las lesiones pequeñas. Al aplicar el aumento a 4 regiones de lesión y a las pequeñas, se logró un 66 %, incrementando este aumento al triple se incrementa a 67 %. Finalmente, al emplear aumento de datos indiscriminado se logró el 73 % de sensitividad, lo que indica que esta ultima estrategia es mejor que sólo centrarlo en ciertas regiones.

Los resultados para los distintos niveles de *EfficientDet-D0* a D6 se mostraron en la tabla 6.3. De acuerdo con lo observado en imágenes naturales, se esperaba que a mayor nivel, fueran mejorando los resultados. Sin embargo, los resultados fueron variantes, en D3 se obtuvo una sensitividad de 79 % con 25 FPs y en D5 un 77 % con 31 FPs. Aunque estos valores de sensitividad son altos, la cantidad de FPs es es muy

grande, es decir, es exagerado definir que en cada tomografía pueden existir hasta 25 o 31 lesiones. Por lo tanto, el mejor resultado sigue siendo el 73 % con 7 FPs, obtenido con D0, ya que 7 es un número aceptable de posibles lesiones sin marcar.

En figura 6.1 se muestra un gráfico de los resultados de la tabla 6.4. En ella se muestra la sensitividad por tamaño de lesión sin emplear FPs, para los distintos niveles de *EfficientDet*. Estos resultados muestran que el nivel D0 es mejor para detectar lesiones de tamaño *large* y *medium*, y que para el resto de niveles, en general es más fácil detectar lesiones pequeñas, aunque esto último puede ser debido a que la mayoría de las muestras contienen lesiones pequeñas.

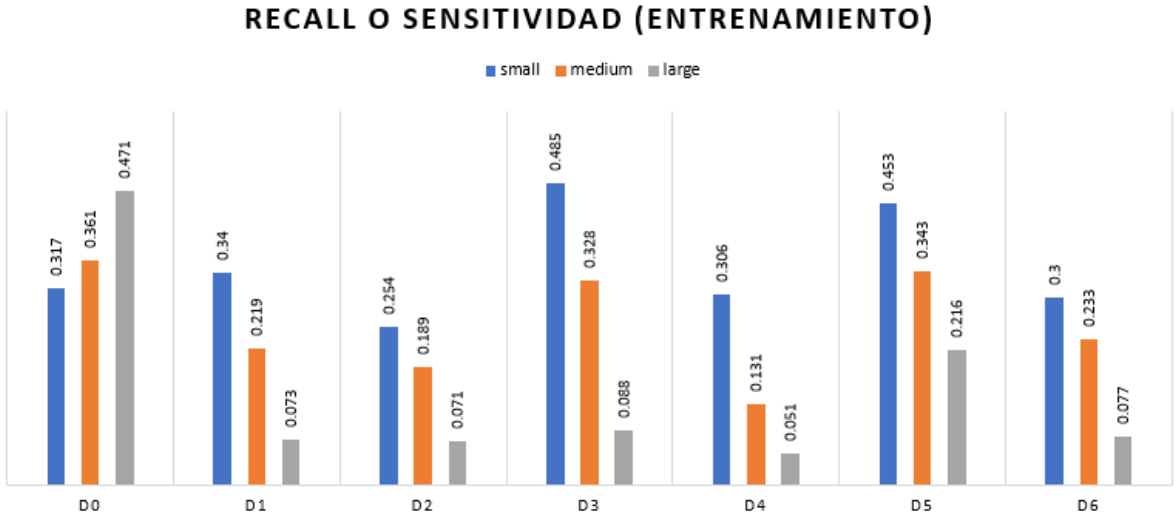

Figura 6.1: Gráfica comparativa de recall por tamaño de lesión en los niveles *EfficientDet*-D0 a *EfficientDet*-D6.

6.3.2. Análisis cualitativo

En esta sección se muestran ejemplos de detecciones realizadas por el modelo *EfficientDet*-D0 con módulos de atención y empleando aumento de datos *cut-and-paste*. En la figura 6.2 se muestran ocho ejemplos de las detecciones realizadas por dicho modelo.

Estos ejemplos muestran recuadros en color azul para señalar el *bounding box target* y en color verde para indicar las predicciones del modelo. En las primeras dos imágenes se muestran detecciones que se traslapan con el *bounding box*, la última muestra una detección incorrecta y el resto de los ejemplos presentan una detección correcta, acompañada de otra detección incorrecta (aunque con un menor *score*).

Al analizar detenidamente dichas predicciones, es posible ver que las regiones predichas como lesión tienen un aspecto muy similar a las regiones que realmente contienen la lesión. Lo cual podría indicar la localización de una lesión real, pero que no fue determinada como la lesión más significativa.

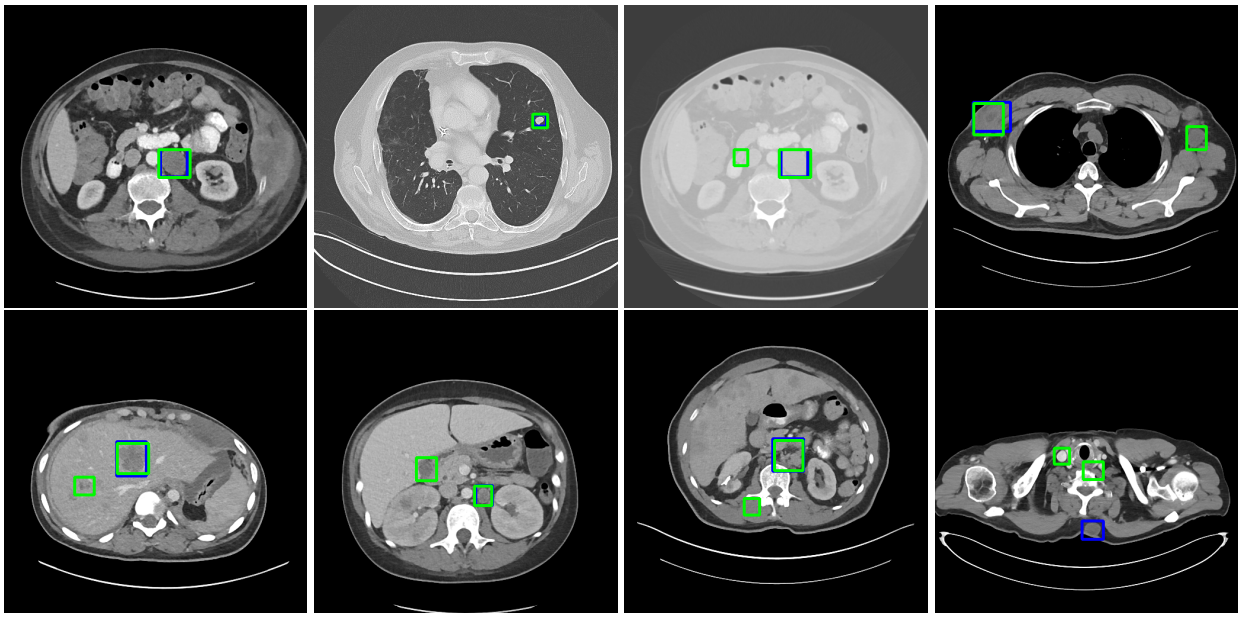

Figura 6.2: Ejemplos de detecciones realizadas. En color azul se tiene el *bounding box* de la lesión real y en verde se tienen las predicciones del modelo.

Con la finalidad de analizar los resultados obtenidos, dados distintos niveles de *EfficientDet*, se decidió realizar una visualización de las activaciones en las primeras capas del modelo. En la figura 6.3, se muestran las imágenes que se tomarán como entrada para visualizar sus activaciones. Se revisarán las activaciones de las capas convolucionales 1 y 5, esto debido a que es más fácil de interpretar las activaciones en las primeras capas, antes de que el modelo comience a trabajar con características más complejas.

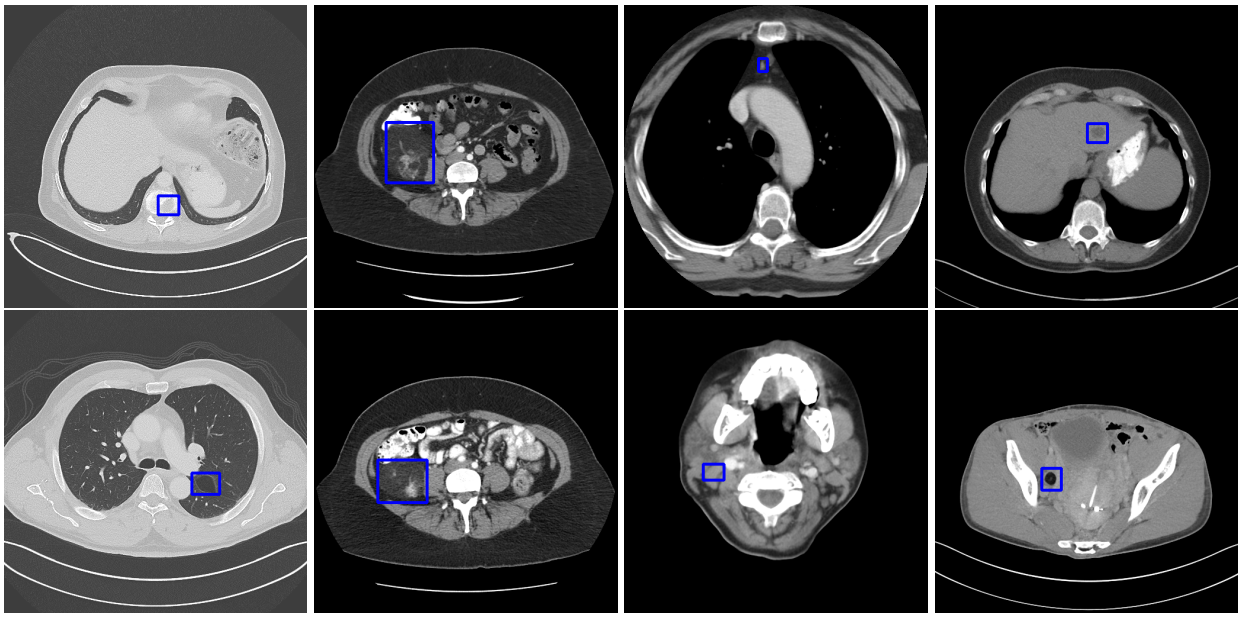

Figura 6.3: Imágenes tomadas como muestra para visualizar las activaciones del modelo al emplear las 8 regiones de lesiones posibles.

En el diagrama que se muestra en la figura 6.4, se indica en que partes del modelo se encuentran las capas convolucionales 1 y 5. Ambas capas se localizan en el *backbone* del modelo, como ya se mencionó en la sección de metodología, está compuesto por ciertos bloques: MBConv1 y MBConv6. La primer convolución (gris intermedio) a visualizar se sitúa previo al bloque MBConv1 (gris claro), mientras que la quinta convolución se encuentra en el primer bloque MBConv6 (rojo). Este último bloque corresponde al primer nivel que envía sus mapas de características a la red de características para la detección a 5 niveles.

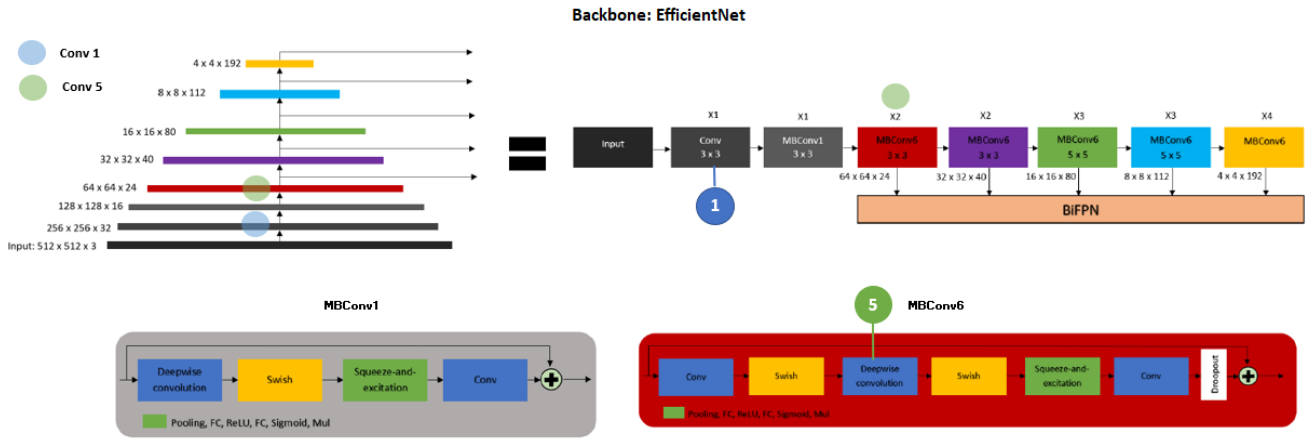

Figura 6.4: Diagrama de especificación de las posiciones de las convoluciones 1 y 5 del modelo.

A continuación, se muestran activaciones para las lesiones en: hueso, abdomen, mediastino, hígado, pulmón, riñón, tejidos blandos y pelvis. Se muestran 2 *kernels* aleatorios de las capas convolucionales 1 y 5, para cada uno de los niveles de *EfficientDet*-D0 a D6. El último renglón muestra algunos ejemplos de *kernels* muertos[1] encontrados en la capa convolucional 5. Cada figura contiene la imagen *input* con su *bounding box* correspondiente, y un mapa de colores del nivel de activación. Donde los colores más claros son más próximos a 1, mientras que los más obscuros son más cercanos a 0.

En la figura 6.5 se muestran las activaciones para un ejemplo de lesión de hueso. Aquí se presentan muy buenas activaciones, ya que son pocas las otras regiones con el mismo nivel de activación. En la convolución 1, el nivel de activación es ligeramente menos notorio en los niveles D5 y D6. Por otro lado, en la convolución 5, se mantienen las buenas activaciones en el nivel D0, se reducen en los niveles D1 - D4, D6, y se pierden en D6.

Lo anterior indica que el modelo D0, es el que mejor aprendió las lesiones de hueso, mientras que el nivel D6 es el que peor lo hizo. En todos los niveles se encontraron *kernels* muertos en la capa convolucional 5, esto ocurrió sobre todo en los niveles D5 y D6.

[1] Un *kernel* muerto es aquel donde todas sus activaciones son iguales

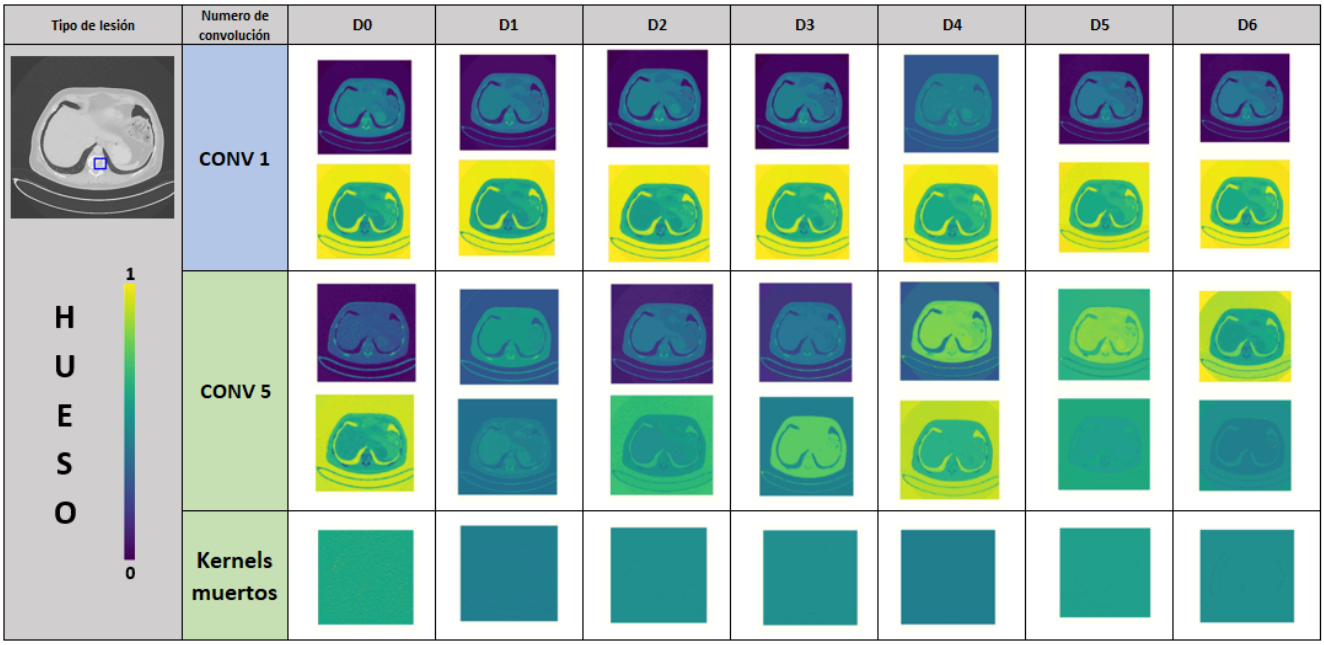

Figura 6.5: Visualización de las activaciones al tener como entrada una imagen de lesión de hueso.

En la figura 6.6 se muestran las activaciones para un ejemplo de lesión en abdomen. En este caso, el nivel de las activaciones en la región con la lesión no son sobresalientes. Esto hace que la lesión no sea distinguible ni el la convolución 1 ni en la 5. Aunque el tamaño de la lesión es grande, la razón probable de este suceso es el problema persistente en las lesiones de abdomen, ya que no son muy distinguibles por su entorno que contiene a los intestinos. En la figura 6.7, se muestran algunos ejemplos de este tipo de lesiones, donde se puede apreciar dicha problemática.

Con la información obtenida, no es posible determinar que nivel de *EfficientDet* es mejor para detectar lesiones de abdomen. En todos los niveles se encontraron *kernels* muertos en la capa convolucional 5, esto ocurrió sobre todo en los niveles D3 y D5.

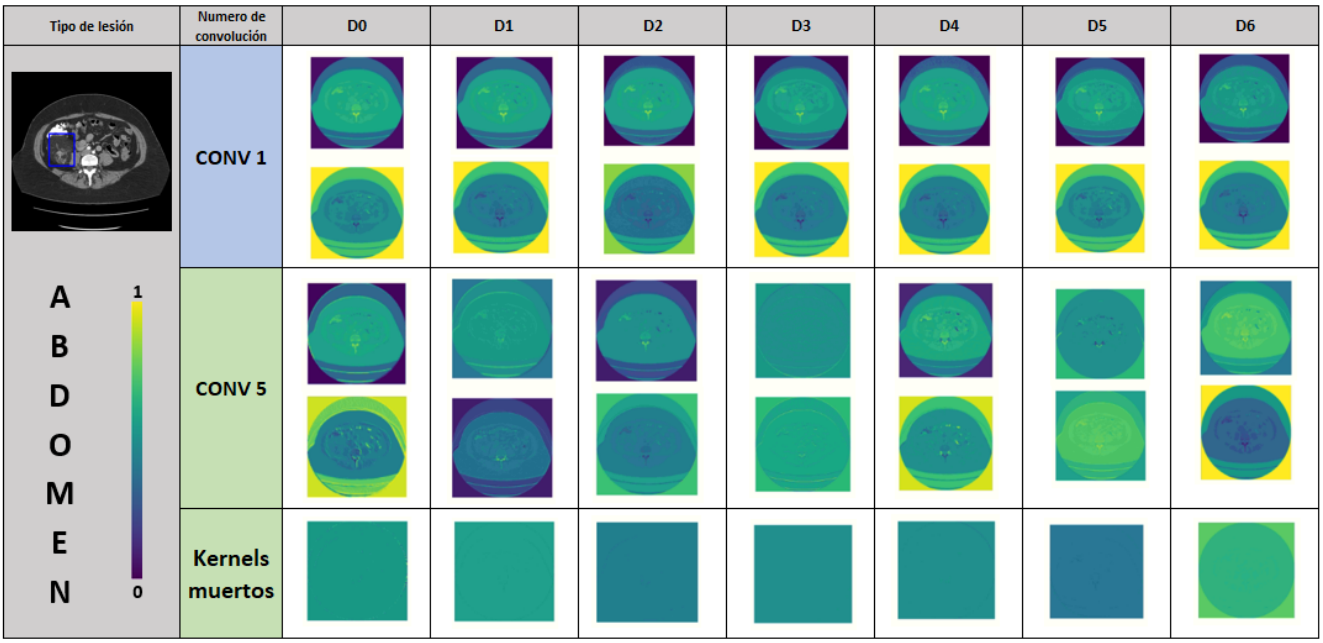

Figura 6.6: Visualización de las activaciones al tener como entrada una imagen de lesión de abdomen.

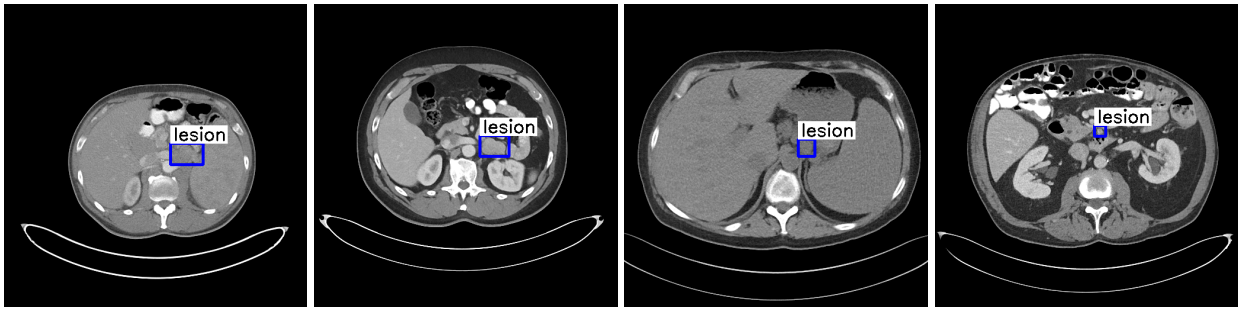

Figura 6.7: Muestras de lesiones de abdomen

En la figura 6.8 se muestran las activaciones para un ejemplo de lesión en mediastino. En este caso no se presentan activaciones que permitan distinguir claramente la región con lesión. Es posible que esto se deba a que la lesión es muy pequeña. En la figura 6.9, se muestran algunos ejemplos de lesión de mediastino, la primera es de la cual se están visualizando las activaciones, el resto son ejemplos aleatorios. Es posible percibir una diferencia considerable de tamaño.

Un suceso notorio en las activaciones de mediastino, es que no se encontraron *kernels* muertos en los niveles D0 y D1.

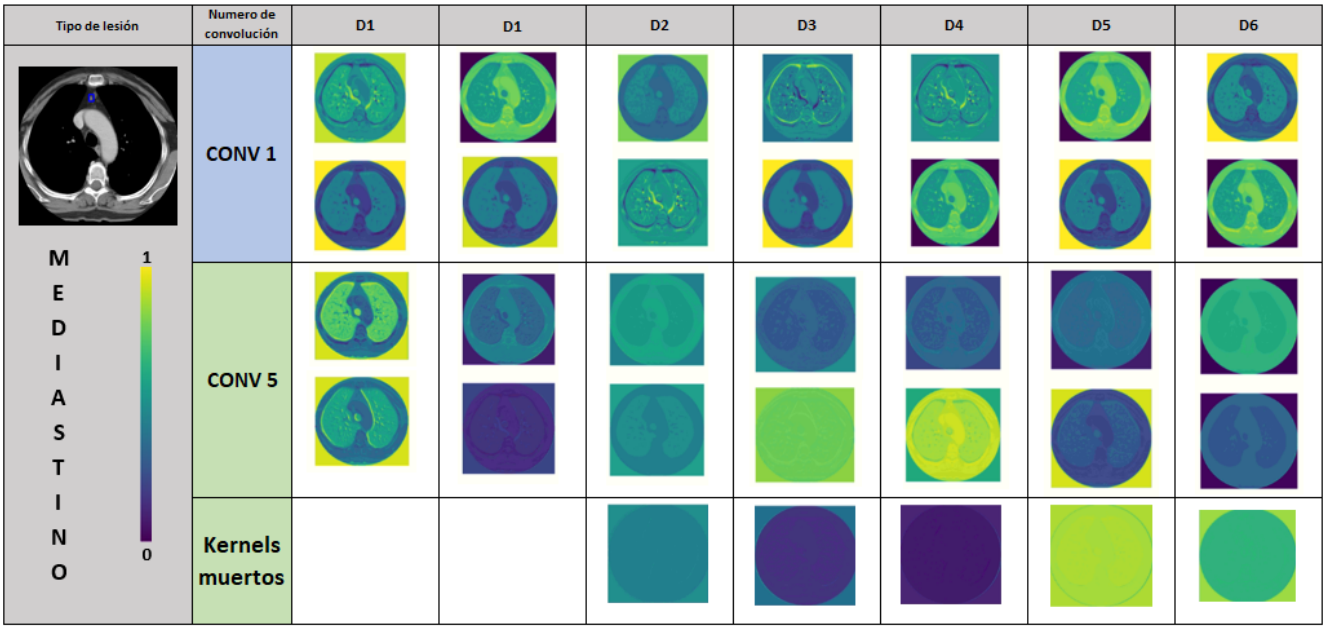

Figura 6.8: Visualización de las activaciones al tener como entrada una imagen de lesión de mediastino.

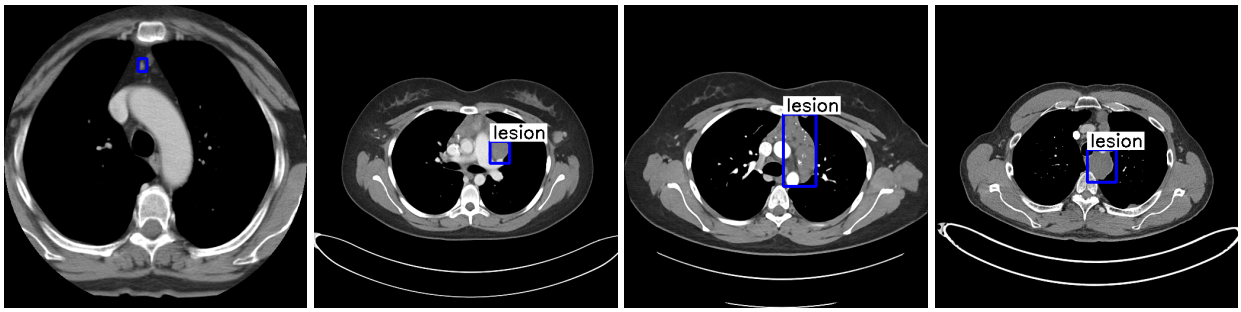

Figura 6.9: Muestras de lesiones de mediastino

En la figura 6.11 se muestran las activaciones para un ejemplo de lesión de hígado. En este tipo de lesión, para la convolución 1, se pueden observar activaciones que destacan la región de lesión, esto en los niveles D0 a D4, pero no en los niveles D5 y D6. Sin embargo, al llegar a la convolución 5, dichas activaciones ya no son perceptibles.

De lo anterior se puede decir que los niveles D5 y D6, son probablemente los que peor detectan las lesiones de hígado. En todos los niveles se encontraron *kernels* muertos en la convolución 5.

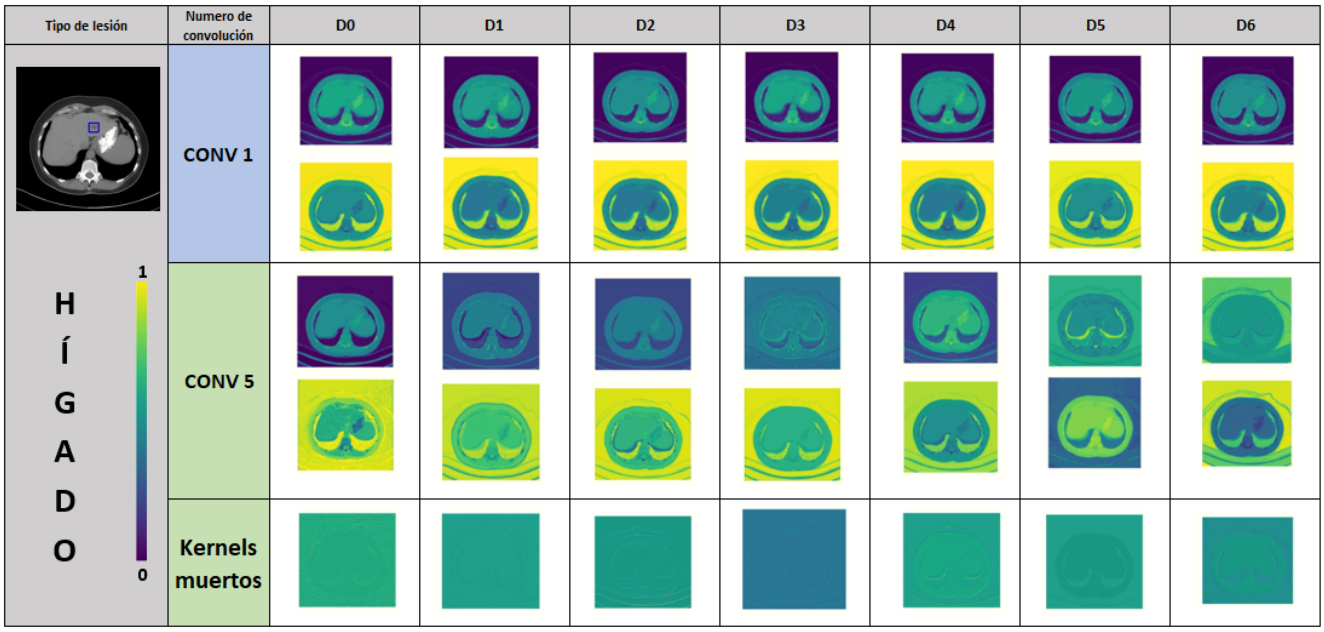

Figura 6.10: Visualización de las activaciones al tener como entrada una imagen de lesión de hígado.

En la figura 6.11 se muestran las activaciones para un ejemplo de lesión de pulmón. En este caso, se presentan buenas activaciones ya que hacen notoria la región de lesión en la primer convolución, en todos los niveles (D0 - D6). Posteriormente, en la convolución 5 dichas activaciones, se reducen en los niveles D0, D2 y D3, aumentan en el nivel D6 y se mantienen iguales en el resto de niveles.

De acuerdo a lo anterior, se podría considerar que el nivel de *EfficientDet*-D6 es el mejor para detectar las lesiones de pulmón. En todos los niveles se encontraron *kernels* muertos en la convolución 5, sobre todo en el nivel D2.

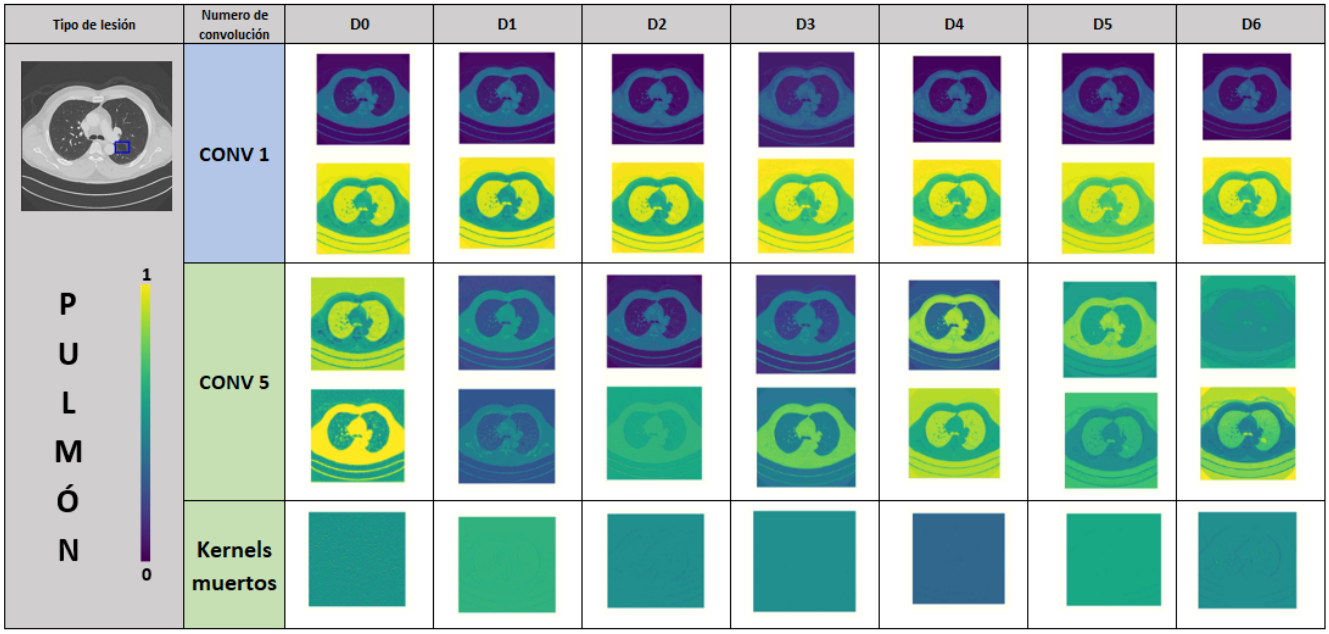

Figura 6.11: Visualización de las activaciones al tener como entrada una imagen de lesión de pulmón.

En la figura 6.12 se muestran las activaciones para un ejemplo de lesión de riñón. Para este tipo de lesión, en la convolución 1, se obtienen valores de activación intermedios. Posteriormente, en la convolución 5, dichas activaciones se mantienen en los niveles D0, D2 y D4, se reducen en D5, y se pierden en D1, D3 y D6. En todos los niveles se encontraron *kernels* muertos en la convolución 5, sobre todo en D3.

Se puede apreciar cierta similitud entre las tomografías estudiadas para lesiones de abdomen y riñón, sin embargo, las lesiones para riñón no se pierden tanto como las de abdomen. En la figura 6.13, se muestran las muestras empleadas para la visualización de lesiones de riñón y abdomen.

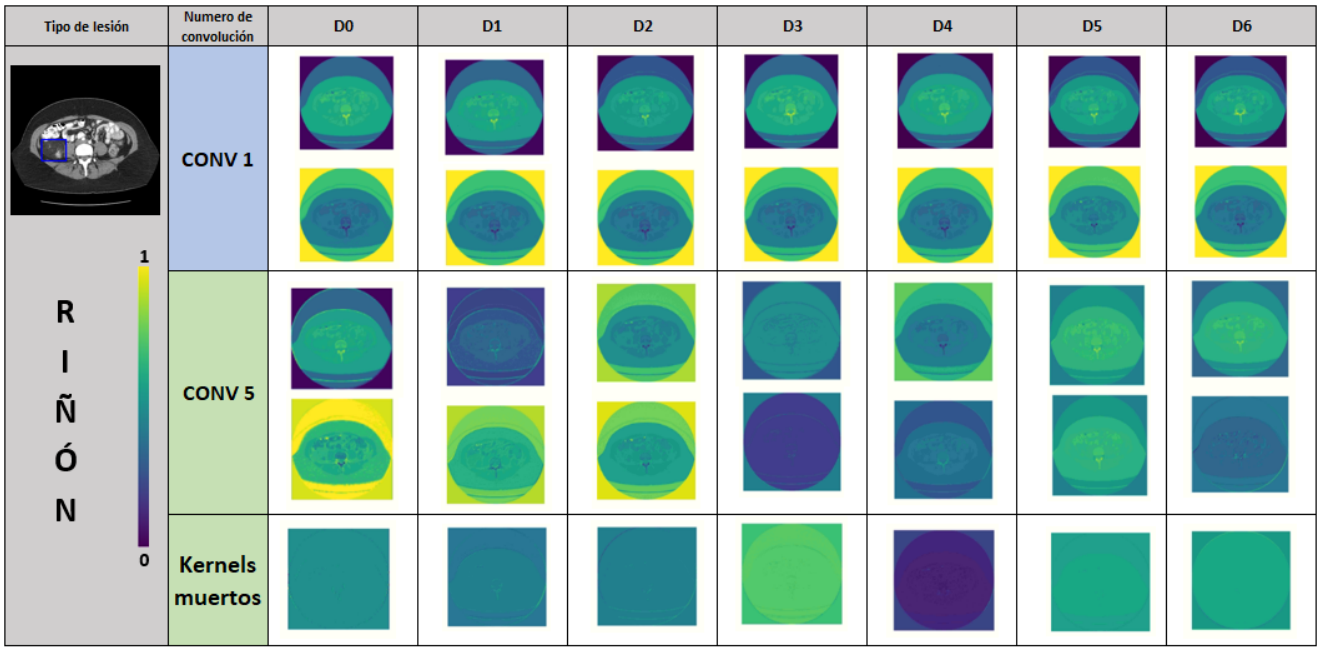

Figura 6.12: Visualización de las activaciones al tener como entrada una imagen de lesión de riñón.

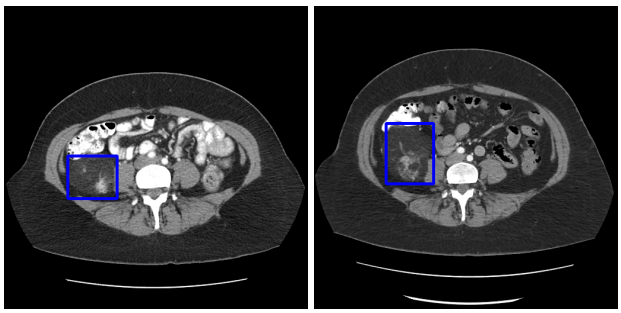

Figura 6.13: Comparativa entre lesiones de riñón (izquierda) y abdomen (derecha).

En la figura 6.14 se muestran las activaciones para un ejemplo de lesión de tejido blando. Para este tipo de lesión aparentemente no se encuentran activaciones que permitan distinguir la región con lesión de tu entorno, en ninguno de los niveles D0 a D6. En todos los niveles se encontraron *kernels* muertos en la convolución 5, sobre todo en D3.

Es posible que los modelos tengan dificultades con este tipo de lesión ya que se cuenta con pocas muestras. Como referencia, la cantidad de muestras de lesiones de tejido blando es aproximadamente la cuarta parte de las que se tienen para lesión de pulmón.

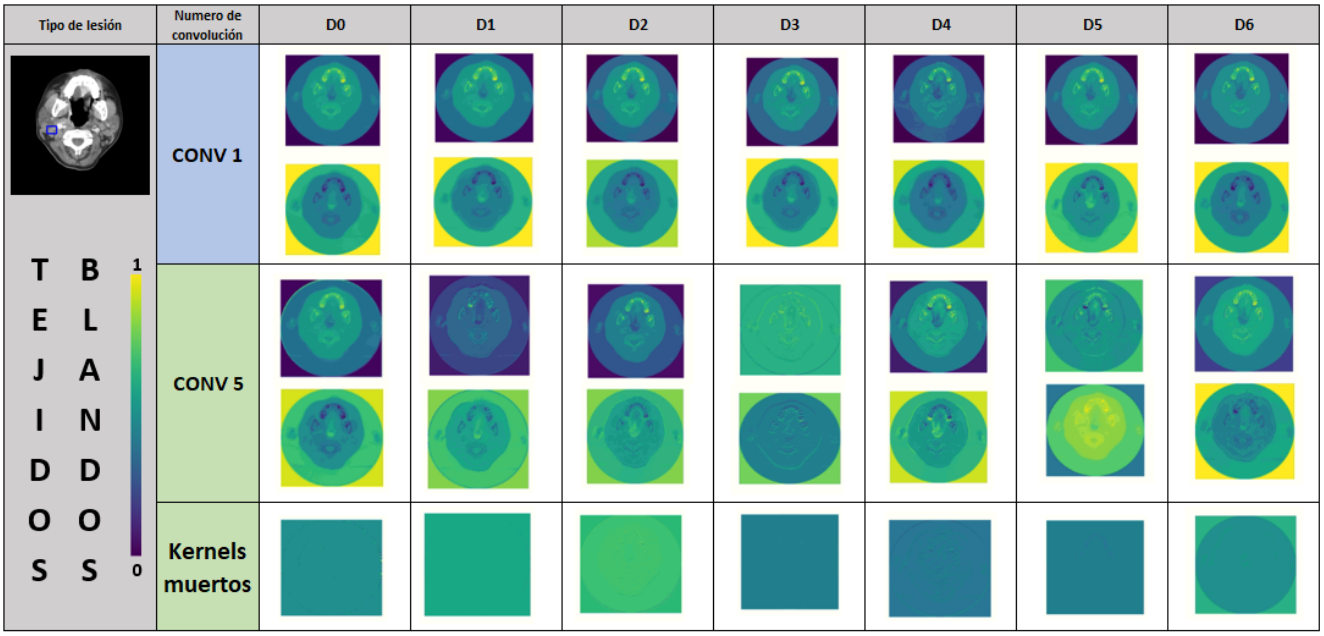

Figura 6.14: Visualización de las activaciones al tener como entrada una imagen de lesión de tejidos blandos.

En la figura 6.15 se muestran las activaciones para un ejemplo de lesión de pelvis. En la primer capa convolucional, se perciben ligeras activaciones en la región de la lesión. Posteriormente, al llegar a la convolución 5, dichas activaciones se intensifican en los niveles D0 y D2, se reducen en D1, D4 y D5, y se pierden en los niveles D3 y D6. Es importante recordar que las lesiones de pelvis son difíciles de distinguir de su entorno, en la figura 6.16, se muestran algunos ejemplos.

Se podría considerar que los niveles D3 y D6, son los que peor detectan las lesiones de pelvis. En todos los niveles se encontraron *kernels* muertos en la convolución 5.

Con el objetivo de analizar el comportamiento del modelo para los distintos tamaños de lesión, se decidió visualizar las activaciones dentro de la primer capa convolucional de los niveles de *EfficientDet*-D0 a D6. En las figuras 6.17 y 6.18, se muestran activaciones para lesiones de abdomen y mediastino, respectivamente. Se muestra una activación por cada tamaño de lesión *small, medium y large*.

Ambos tipos de lesión presentaron problemas previamente en el análisis por tipo de lesión, ya que no se distinguían activaciones centradas en la región de las lesiones. Aquí se observa que independientemente

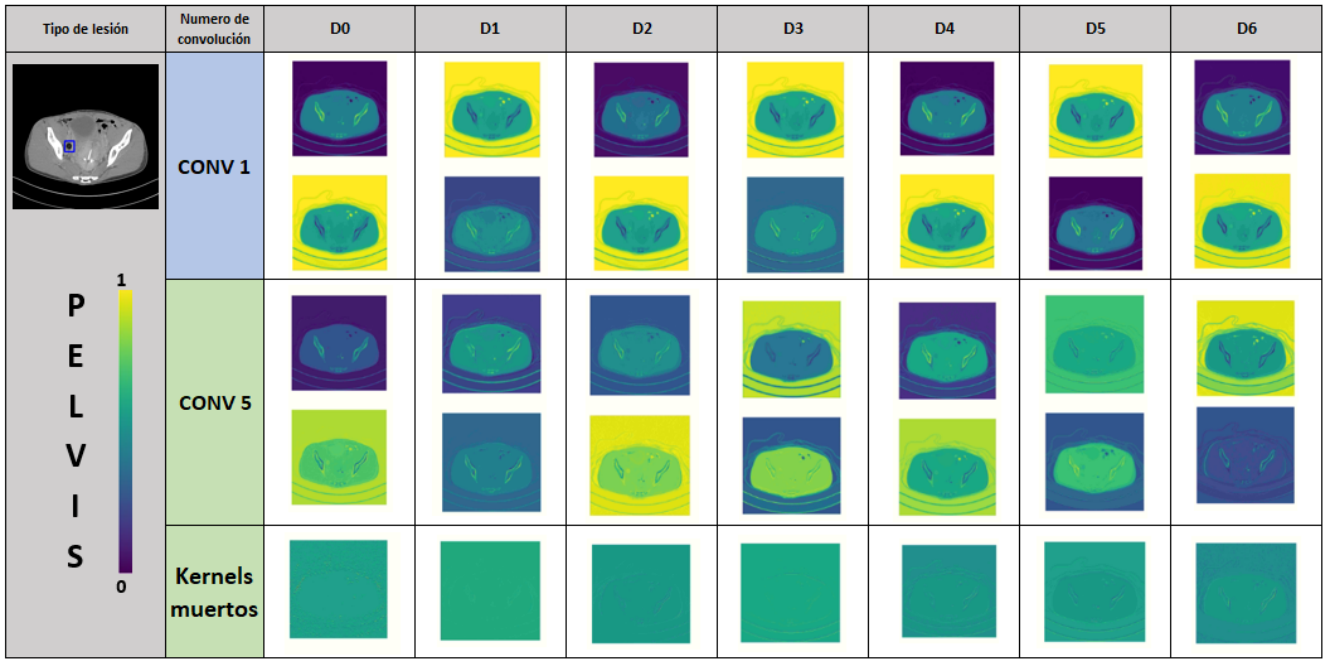

Figura 6.15: Visualización de las activaciones al tener como entrada una imagen de lesión de pelvis.

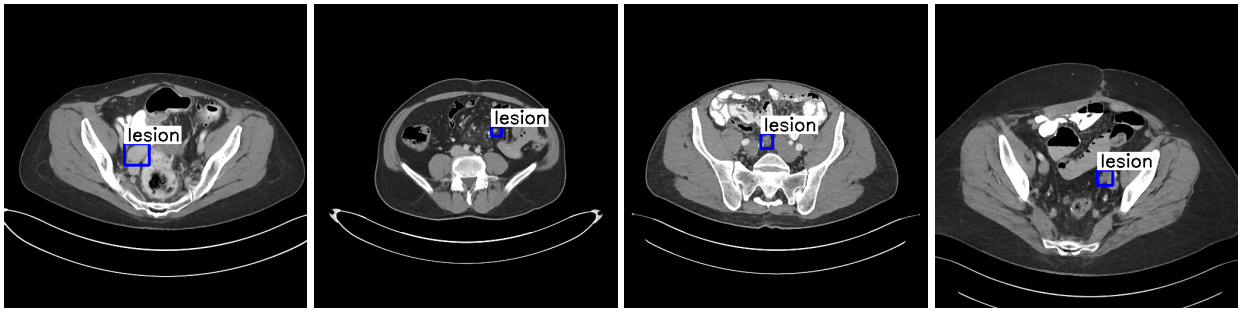

Figura 6.16: Muestras de lesiones de pelvis

del tamaño de la lesión, ocurre lo mismo y no se generaron activaciones especialmente centradas en la lesión. Un evento notorio,es que se muestra una mayor cantidad de activaciones para las lesiones *small*, en todos los niveles.

Con la finalidad de comparar el comportamiento de las activaciones de imágenes médicas e imágenes naturales, en la figura 6.19, se muestran las activaciones de la primer capa convolucional de *Efficient-Det*-D0 a D6 para un par de imágenes naturales. El modelo se empleó con los pesos pre-entrenados con ImageNet, y las imágenes para la visualización fueron tomadas de la base de datos COCO.

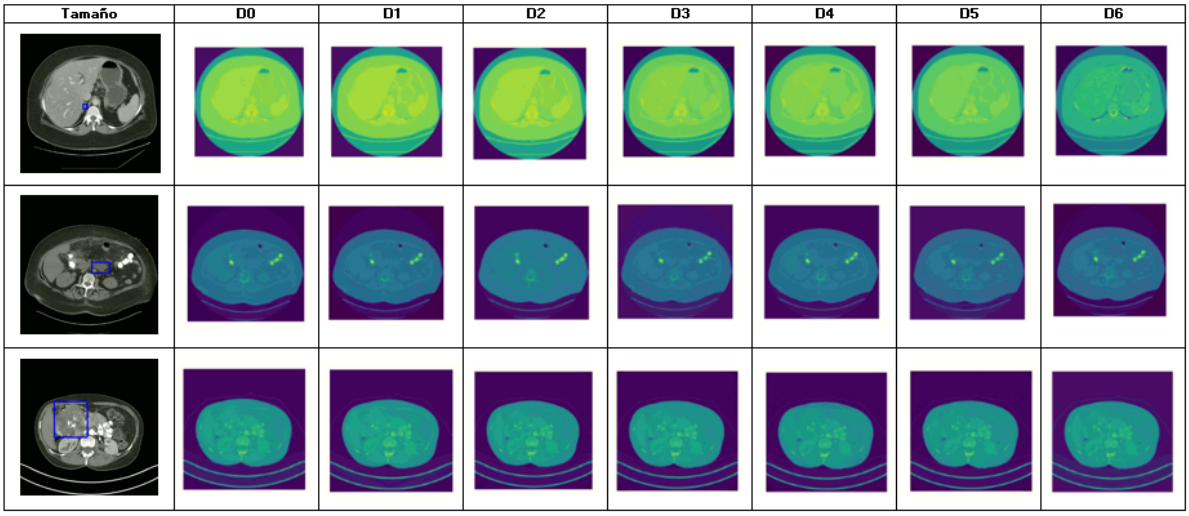

Figura 6.17: Comparativa de las activaciones por tamaño *small, medium y large* dada como entrada una tomografía con lesión de abdomen.

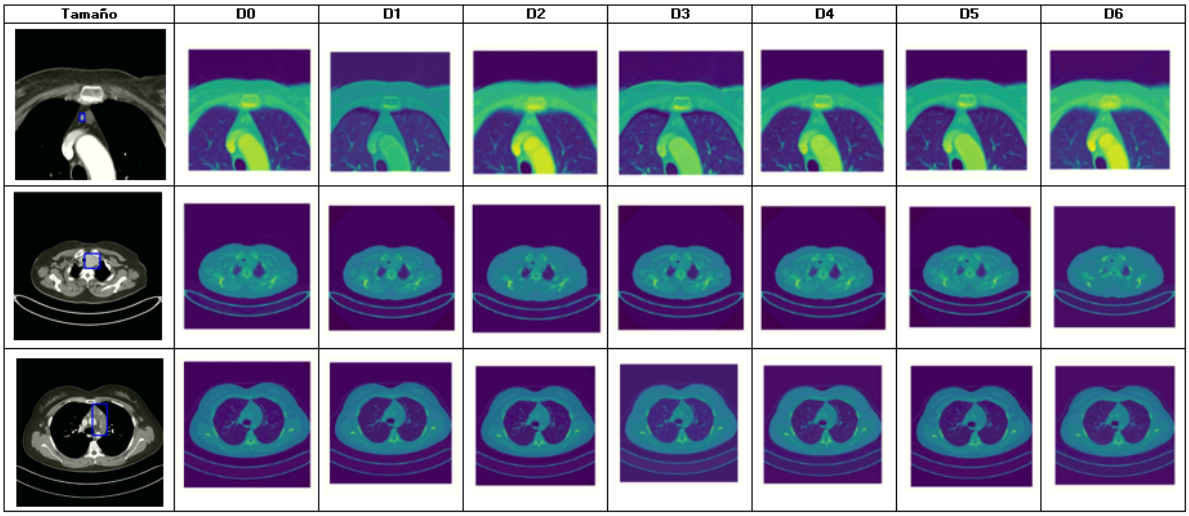

Figura 6.18: Comparativa de las activaciones por tamaño *small, medium y large* dada como entrada una tomografía con lesión de mediastino.

Al igual que con las tomografías, no se encontró una gran diferencia en las activaciones de la primer capa de convolución, de los distintos niveles de *EfficientDet*-D0 a D6. Por otra parte, estos resultados son más fáciles de interpretar, a diferencia de los resultados mostrados en las tomografías, para las cuales se necesita cierto nivel de expertiz en el área médica.

Se encontró una diferencia notable entre las activaciones de las imágenes médicas y naturales, al em-

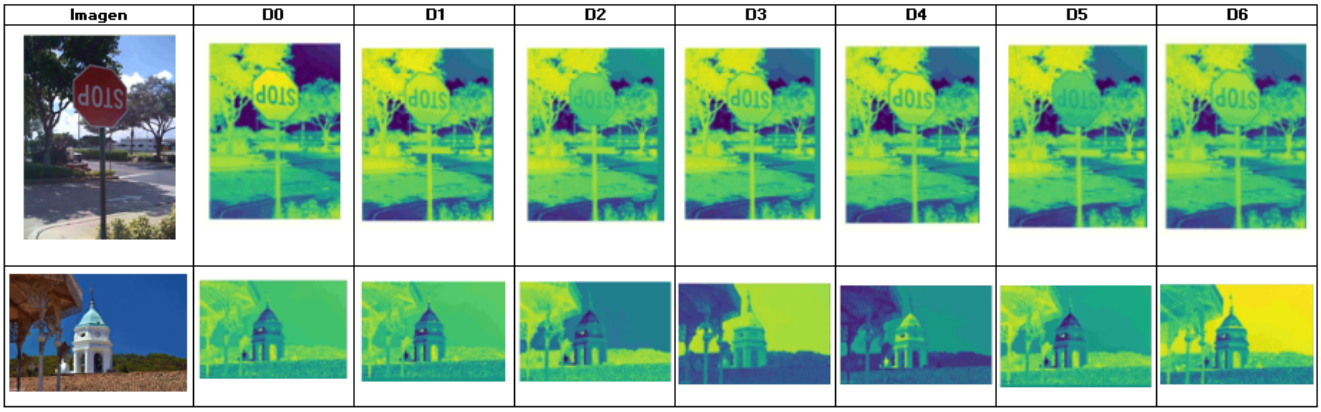

Figura 6.19: Ejemplos del comportamiento de las activaciones de *EfficientDet* al trabajar con imágenes naturales.

plear estas últimas se llegan a generar activaciones claramente centradas en el objeto a detectar. Es decir, en las activaciones de las lesiones no se logran observar *kernels* con activación exclusivamente centrada en la lesión, mientras que en las imágenes naturales si se encuentran *kernels* como los de la figura 6.20, donde, las activaciones se centran en la parada de alto y en la torre.

Figura 6.20: Activaciones de imágenes naturales.

Capítulo 7

Discusión y conclusiones

En este capitulo se presenta la discusión y conclusiones asociadas a los objetivos planteados inicialmente. Posteriormente, se presenta una sección de trabajo futuro.

7.1. Discusión y conclusiones

En esta investigación se abordó el problema de crear un detector universal de lesiones en tomografías, el cual comenzó a abordarse con estrategias de *deep learning*, hace tan sólo un par de años. Además, se estudió un detector de objetos complejo, y poco estudiado hasta el momento. A continuación, se discuten los objetivos específicos planteados previamente en la sección 1.

1. **Determinar si incluso al emplear una base de datos extensa como *DeepLesion*, es necesario realizar aumento de datos.**

   Los autores de la base de datos *DeepLesion* afirman que es suficientemente grande, y no se requiere realizar ninguna clase de aumento de datos. Sin embargo, aún no existe una forma exacta de determinar cuantas imágenes bastan para entrenar un modelo, sobre todo cuando se trata de *deep learning.*

Los resultados obtenidos muestran un aumento de 16 % de sensitividad al emplear *flip* horizontal y de 18 % al aplicar *cut-and-paste*. Esto muestra que aunque se esté trabajando con la base de datos de imágenes médicas más extensa hasta la fecha, el aumento de datos sigue siendo una estrategia fundamental.

2. **Precisar la relevancia del contexto inmediato a la lesión en la estrategia de aumento de datos.**

   Se estudiaron 2 tipos de aumento de datos; *flip* horizontal que es la estrategia seleccionada tradicionalmente, y se propuso emplear *cut-and-paste*. La principal diferencia entre estas dos estrategias es que con *flip* horizontal se mueve toda la imagen. Es decir, la lesión mantiene su contexto inmediato, mientras que con *cut-and-paste* se modifica.

   De acuerdo con los resultados obtenidos, la estrategia *cut-and-paste* es superior al *flip horizontal* en un 2 %. Esto indica que darle mayor variabilidad contextual a la red, permite que el modelo aprenda a realizar una mejor discriminación de lesiones.

3. **Establecer si es mejor el aumento de datos indiscriminado o el centrado en el tipo de lesiones más difíciles de detectar.**
   Para determinar en que datos centrar el aumento de datos, se consideraron los siguientes puntos:

   - Las lesiones pequeñas: la detección de objetos pequeños es un problema persistente para los detectores.
   - Las lesiones de abdomen y pelvis: Son lesiones difíciles de detectar ya que son difíciles de distinguir de su entorno.
   - Las lesiones en tejidos blandos y huesos: Son lesiones de las que se cuenta con menos muestras.

   Se realizaron 4 aumentos de datos, el primero centrado en las lesiones pequeñas y en las lesiones de abdomen y pelvis. El segundo centrado adicionalmente en las lesiones de tejidos blandos y huesos.

En tercer lugar, se realizó el mismo enfoque sólo que ahora se generaban 3 muestras nuevas por cada imagen. Finalmente, se realizó un aumento a todas las imágenes.

De acuerdo con los resultados, al emplear *EfficientDet*-D0 con módulos de atención, cuando el aumento de datos se centró en las lesiones pequeñas, y de abdomen y pelvis se mejoró la sensitividad del modelo en un 8 %. Cuando se centró además en las regiones de hueso y tejidos blandos, la sensitividad del modelo mejoró en un 14 %. Al aplicar esta última a una razón de 1:3, la mejora anterior solo se superó en un 1 %. Por otra parte, considerando el aumento de datos indiscriminado se logró una mejora de la sensitividad del modelo en un 21 %. Lo que indica que la mejor estrategia para el aumento de datos en *DeepLesion* es el indiscriminado.

Se podría pensar que un modelo puede aprender mejor si tiene más muestras de las lesiones más difíciles de aprender. Pero siguiendo este pensamiento, al aumentar indiscriminadamente todos los tipos de lesiones, el modelo tiene más muestras para aprender el tipo de lesiones que son más difíciles, y a su vez tiene más muestras para aprender que lesiones no son del tipo difícil.

4. **Deducir si el estado del arte en imágenes naturales, tiene el mismo efecto al trabajar con imágenes médicas.**

Actualmente el estado del arte en la detección de objetos es *EfficientDet*, este modelo se seleccionó para este trabajo por sus cualidades. Emplea *EfficientNet* como *backbone* que es parte del estado del arte en clasificación de imágenes. Es decir, ha demostrado obtener mejores resultados que otros *backbones* empleados en el trabajo relacionado (como ResNet). Su red de características es BiFPN, que por la disposición de sus conexiones ha demostrado ser mejor que FPN.

Los resultados obtenidos muestran que no todos los elementos, propuestas o estrategias que funcionan al trabajar con imágenes naturales aplicarán de igual manera al trabajar con imágenes médicas.

De igual manera, al analizar las activaciones del modelo con imágenes médicas y naturales, se apreció que al emplear imágenes naturales si hay *kernels* centrados específicamente en los objetos a detectar.

Esto se podría pensar como una analogía: para cualquier persona es fácil discriminar objetos de la vida cotidiana en una base de datos como COCO, incluso es fácil aprender como es un objeto que no conocíamos al ver varias muestras. Pero esto no pasa con las imágenes médicas, no cualquier persona puede distinguir una lesión, incluso después de ver varios ejemplos. La realidad es que las personas capacitadas para detectar lesiones tuvieron que estudiar años para poder realizar dichas detecciones. Y si los modelos de redes neuronales artificiales se basan en la forma en la que aprendemos las personas, es lógico pensar que a un modelo basado en redes neuronales también se le dificultará más detectar lesiones que objetos cotidianos.

5. **Interpretar el impacto que tienen distintos niveles de profundidad de *EfficientDet* en la detección de lesiones.**

   EfficientDet es reconocido actualmente como el estado del arte en la detección de imágenes naturales. Al probarse con la base de datos COCO, mostró resultados cada vez mejores conforme se incrementaba el nivel del modelo, debido a esto se esperaba que fuera el mismo caso al emplear imágenes médicas, y que los niveles más altos de *EfficientDet* arrojarán mejores resultados.

   Los mejores resultados se lograron con el nivel base (*EfficientDet*-D0) con un 73 % de sensitividad a 7 FPs. Este porcentaje fue superado en los niveles D3 con un 79 % y en D5 con un 77 %, con 25 y 31 FPs, respectivamente. Sin embargo, estos dos resultados no son considerados como los mejores de esta investigación ya que se está considerando un número muy grande de FPs.

   En la búsqueda de una mejor interpretación de los resultados arrojados por los distintos niveles

de *EfficientDet*, se visualizaron las activaciones de las primeras capas convolucionales del modelo. Se observó que mientras mayor era la profundidad del modelo, se iban perdiendo más activaciones y se presentaban más *kernels* muertos. Este suceso puede ser un indicador de que los modelos no fueron entrenados suficientes épocas o de que se empleó un *learning rate* grande.

En las implementaciones reportadas se empleó un *learning rate* de $1e-4$ que puede considerarse pequeño. Durante la búsqueda de parámetros se emplearon valores más grandes y más pequeños, pero ninguno fue mejor. Respecto a la cantidad de épocas para el entrenamiento de los modelos, se puede considerar que fueron relativamente pocas épocas de entrenamiento, ya que en los resultados reportados no superan las 8 épocas. Para determinar cuando detener el entrenamiento se tomó como criterio el progreso de las pérdidas arrojadas por época, si éstas no se reducían o incluso aumentaban después de algunas épocas y días de entrenamiento, se detenía el entrenamiento del modelo.

Al realizar un análisis por tamaño de lesión (*small, medium y large*), se tiene que EfficientDet-D0 es el mejor nivel detectando lesiones de tamaño *medium y large*; mientras que EfficientDet-D3 y D5 detectan mejor las lesiones *small*, además, presentaron resultados próximos a los de D0 para las lesiones *medium*.

El 61 % de las lesiones de la base de datos son de tamaño *small*, las cuales son mejor detectadas por los modelos más profundos. Esto indica que los modelos más profundos tienen mayor capacidad para aprender detalles a escalas tan pequeñas, siempre y cuando se tengan suficientes muestras para el entrenamiento.

Por otro lado, el 39 % de las imágenes repartidas entre *medium y large* fueron mejor detectadas por el modelo menos profundo. Es decir, la arquitectura base funciona mejor para lesiones medianas y grandes, pero no es tan profundo para alcanzar a aprender detalles muy pequeños.

Al realizar el análisis por tamaño de lesión de las activaciones para los distintos niveles de *EfficientDet*, se observó que para las lesiones pequeñas las activaciones se dan en una mayor porción de la tomografía, y las de lesiones medianas y grandes no varían mucho entre si.

Todo esto lleva a la conclusión de que sí es posible obtener mejores sensitividades con los distintos niveles de *EfficientDet*, pero es importante prestar atención a los tres elementos mencionados: cantidad suficiente de imágenes para el entrenamiento, una mayor cantidad de épocas y *learning rate* adecuado. Ya que se hicieron pruebas con distintos valores de *learning rate*, es probable que este no sea el factor que afecta al modelo. Para el caso de de la cantidad de épocas se requiere equipo de cómputo más potente que con el que se contó para esta investigación, o dejar entrenando los modelos por algunas semanas.

Si bien, el esfuerzo por generar *DeepLesion* fue un gran aporte por parte de Yan *et. al*, al ser la base de datos de tomografías computarizadas más grande hasta la fecha. Es necesaria la creación de una base de datos aún más grande, aunque esto requerirá de mucho tiempo de recolección y etiquetado con ayuda de expertos en la materia. Dado que dicho camino no es factible para obtener una mayor cantidad de imágenes para el entrenamiento, se podría optar por recolectar más imágenes de otras bases de datos, y por hacer pruebas con otras técnicas de aumento de datos o variantes de la técnica empleada en esta investigación.

6. **Identificar estrategias de atención a incorporar en *EfficientDet*, que permitan mejorar la detección de lesiones.**

La estrategia empleada por *default* al trabajar con CTs, es la atención del contexto 3D, esto para las imágenes previamente mejoradas en cuanto al contraste. Esta estrategia de atención fue aplicada, y además, se determinaron las siguientes estrategias:

- Se colocaron módulos de atención espacial y de canal en los mapas de características de salida de los niveles 3-7 del *backbone*, para capturar las características de las posibles lesiones desde la perspectiva de múltiples escalas/niveles.
- Se situaron módulos de atención espacial en los bloques de convolución de la red de clasificación, para filtrar información irrelevante, y que la red pudiera centrar su atención en las regiones con lesión. Especialmente se determinó colocarlos en esta red, ya que durante el entrenamiento el modelo presentaba mayor pérdida de clasificación que de regresión.

Los resultados obtenidos al empelar *EfficietDet*-D0 con aumento de datos *cut-and-paste* fue de 70 %, al colocar los módulos de atención propuestos, este porcentaje de sensitividad llegó a un 73 % con 7FPs.

La idea de aplicar atención en el modelo también puede pensarse como una analogía, por ejemplo, si estamos revisando varios artículos, y queremos identificar cuales tienen la temática de detección de objetos, vamos a centrar nuestra atención en determinados elementos, como el título o las palabras clave. Esta misma idea se quiere lograr al añadir los módulos de atención, que el modelo centre su atención en características de las tomografías que son clave para detectar lesiones.

7.2. Trabajo futuro

De acuerdo a lo mencionado previamente se han considerado los siguientes puntos como posible trabajo futuro:

- Hacer la base de datos más grande:

    1. Buscar bases de datos de tomografías de las regiones con las que se cuentan menos muestras.
    2. Explorar más estrategias de aumento de datos y analizar el por qué de su impacto.
    3. Modificar la estrategia *cut-and-paste* y además de intercambiar la posición de las lesiones, también hacerlo entre imágenes del mismo tipo de lesión.

- Entrenar los modelos de EfficientDet por una mayor cantidad de épocas, ya sea en un equipo de computo más potente o en el actual pero por varias semanas.

- Buscar la colaboración de un médico o radiólogo que pueda aportar sus conocimientos, sobre el proceso que lleva a cabo para detectar lesiones, y pueda dar una mejor interpretación de la visualización de las activaciones del modelo.

Bibliografía

[1] M. Tan, R. Pang, and Q. V. Le, "Efficientdet: Scalable and efficient object detection," pp. 10 781–10 790, 2020.

[2] C. L. L. from Large-scale, "Deeplesion: Automated deep mining, categorization and detection of significant radiology image findings using large-scale clinical lesion annotations."

[3] K. Yan, X. Wang, L. Lu, and R. M. Summers, "Deeplesion: automated mining of large-scale lesion annotations and universal lesion detection with deep learning," *Journal of medical imaging*, vol. 5, no. 3, p. 036501, 2018.

[4] H. Zhang, Y. Chen, Y. Song, Z. Xiong, Y. Yang, and Q. J. Wu, "Automatic kidney lesion detection for ct images using morphological cascade convolutional neural networks," *IEEE Access*, vol. 7, pp. 83 001–83 011, 2019.

[5] Q. Shao, L. Gong, K. Ma, H. Liu, and Y. Zheng, "Attentive ct lesion detection using deep pyramid inference with multi-scale booster," pp. 301–309, 2019.

[6] K. Yan, Y. Tang, Y. Peng, V. Sandfort, M. Bagheri, Z. Lu, and R. M. Summers, "Mulan: multitask universal lesion analysis network for joint lesion detection, tagging, and segmentation," pp. 194–202, 2019.

[7] Z. Li, S. Zhang, J. Zhang, K. Huang, Y. Wang, and Y. Yu, "Mvp-net: Multi-view fpn with position-aware attention for deep universal lesion detection," pp. 13–21, 2019.

[8] H. G. Dantés, "Las lesiones por causa externa en méxico: lecciones aprendidas y desafíos para el sistema nacional de salud," *salud pública de méxico*, vol. 53, pp. 99–101, 2011.

[9] K. Yan, X. Wang, L. Lu, and R. M. Summers, "Deeplesion: automated mining of large-scale lesion annotations and universal lesion detection with deep learning," *Journal of Medical Imaging*, vol. 5, no. 3, p. 036501, 2018.

[10] D. Jin, A. P. Harrison, L. Zhang, K. Yan, Y. Wang, J. Cai, S. Miao, and L. Lu, "Artificial intelligence in radiology," in *Artificial Intelligence in Medicine*. Elsevier, 2020, pp. 265–289.

[11] K.-Y. Lung, C.-R. Chang, S.-E. Weng, H.-S. Lin, H.-H. Shuai, and W.-H. Cheng, "Rosnet: Robust one-stage network for ct lesion detection," *Pattern Recognition Letters*, vol. 144, pp. 82–88, 2021.

[12] G. Cai, J. Chen, Z. Wu, H. Tang, Y. Liu, S. Wang, and S. Su, "One stage lesion detection based on 3d context convolutional neural networks," *Computers & Electrical Engineering*, vol. 79, p. 106449, 2019.

[13] K. Yan, X. Wang, L. Lu, L. Zhang, A. P. Harrison, M. Bagheri, and R. M. Summers, "Deep lesion graphs in the wild: relationship learning and organization of significant radiology image findings in a diverse large-scale lesion database," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 9261–9270.

[14] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[15] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[16] F. Iandola, M. Moskewicz, S. Karayev, R. Girshick, T. Darrell, and K. Keutzer, "Densenet: Implementing efficient convnet descriptor pyramids," *arXiv preprint arXiv:1404.1869*, 2014.

[17] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

[18] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: towards real-time object detection with region proposal networks," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 6, pp. 1137–1149, 2016.

[19] X. Wu, D. Sahoo, and S. C. Hoi, "Recent advances in deep learning for object detection," *Neurocomputing*, 2020. [Online]. Available: https://doi.org/10.1016/j.neucom.2020.01.085

[20] S. Lu, B. Wang, H. Wang, L. Chen, M. Linjian, and X. Zhang, "A real-time object detection algorithm for video," *Computers & Electrical Engineering*, vol. 77, pp. 398–408, 2019.

[21] S. Ojha and S. Sakhare, "Image processing techniques for object tracking in video surveillance-a survey," in *2015 International Conference on Pervasive Computing (ICPC)*. IEEE, 2015, pp. 1–6.

[22] H. K. Chavda and M. Dhamecha, "Moving object tracking using ptz camera in video surveillance system," in *2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS)*. IEEE, 2017, pp. 263–266.

[23] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[24] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European conference on computer vision*. Springer, 2016, pp. 21–37.

[25] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[26] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*. Springer, 2014, pp. 740–755.

[27] R. Cipolla, S. Battiato, and G. M. Farinella, *Computer Vision: Detection, recognition and reconstruction*. Springer, 2010, vol. 285.

[28] R. Szeliski, *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

[29] A. Rosebrock, *Deep Learning for Computer Vision with Python: Starter Bundle*. Pyimagesearch, 2017, vol. 1.

[30] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[31] S. Marsland, *Machine learning: an algorithmic perspective*. CRC press, 2015.

[32] DARPA, "Transfer learning proposer information pamphlet (pip) for broad agency announcement," *Defense Advanced Research Projects Agency*, 2005.

[33] U. Kamath, J. Liu, and J. Whitaker, *Deep learning for NLP and speech recognition*. Springer, 2019, vol. 84.

[34] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[35] S. Woo, J. Park, J.-Y. Lee, and I. S. Kweon, "Cbam: Convolutional block attention module," pp. 3–19, 2018.

[36] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," *arXiv:1709.01507v4*, pp. 7132–7141, 2018.

[37] W. Qilong, W. Banggu, Z. Pengfei, L. Peihua, Z. Wangmeng, and H. Qinghua, "Eca-net: Efficient channel attention for deep convolutional neural networks," pp. 11 531–11 539, 2020.

[38] B. Pourbabaee, M. J. Roshtkhari, and K. Khorasani, "Deep convolutional neural networks and learning ecg features for screening paroxysmal atrial fibrillation patients," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 48, no. 12, pp. 2095–2104, 2018.

[39] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[40] H. Cecotti and A. Graser, "Convolutional neural networks for p300 detection with application to brain-computer interfaces," *IEEE transactions on pattern analysis and machine intelligence*, vol. 33, no. 3, pp. 433–445, 2010.

[41] K. B. Lee, S. Cheon, and C. O. Kim, "A convolutional neural network for fault classification and diagnosis in semiconductor manufacturing processes," *IEEE Transactions on Semiconductor Manufacturing*, vol. 30, no. 2, pp. 135–142, 2017.

[42] G. Ghiasi, T.-Y. Lin, and Q. V. Le, "Nas-fpn: Learning scalable feature pyramid architecture for object detection," pp. 7036–7045, 2019.

[43] J. Wang, P. Fu, and R. X. Gao, "Machine vision intelligence for product defect inspection based on deep learning and hough transform," *Journal of Manufacturing Systems*, vol. 51, pp. 52–60, 2019.

[44] S. Marsland, *Machine learning: an algorithmic perspective*. Chapman and Hall/CRC, 2014.

[45] A. Rosebrock, *Deep Learning for Computer Vision with Python: Starter Bundle*. Pyimagesearch, 2017, vol. 2.

[46] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of Big Data*, vol. 6, no. 1, pp. 1–48, 2019.

[47] M. Kisantal, Z. Wojna, J. Murawski, J. Naruniec, and K. Cho, "Augmentation for small object detection," *arXiv preprint arXiv:1902.07296*, 2019.

[48] G. Ghiasi, Y. Cui, A. Srinivas, R. Qian, T.-Y. Lin, E. D. Cubuk, Q. V. Le, and B. Zoph, "Simple copy-paste is a strong data augmentation method for instance segmentation," *arXiv preprint arXiv:2012.07177*, 2020.

[49] H. Wang, Q. Wang, F. Yang, W. Zhang, and W. Zuo, "Data augmentation for object detection via progressive and selective instance-switching," *arXiv preprint arXiv:1906.00358*, 2019.

[50] D. Dwibedi, I. Misra, and M. Hebert, "Cut, paste and learn: Surprisingly easy synthesis for instance detection," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 1301–1310.

[51] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," pp. 580–587, 2014.

[52] P. Purkait, C. Zhao, and C. Zach, "Spp-net: Deep absolute pose regression with synthetic views," *arXiv preprint arXiv:1712.03452*, 2017.

[53] R. Girshick, "Fast r-cnn," pp. 1440–1448, 2015.

[54] H. Law and J. Deng, "Cornernet: Detecting objects as paired keypoints," pp. 734–750, 2018.

[55] H. Zhang, H. Chang, B. Ma, S. Shan, and X. Chen, "Cascade retinanet: Maintaining consistency for single-stage object detection," *arXiv preprint arXiv:1907.06881*, 2019.

[56] A. Rosebrock, *Deep Learning for Computer Vision with Python: Starter Bundle*. Pyimagesearch, 2017, vol. 3.

[57] D.-J. Chang, A. H. Desoky, M. Ouyang, and E. C. Rouchka, "Compute pairwise manhattan distance and pearson correlation coefficient of data points with gpu," in *2009 10th ACIS International Conference on Software Engineering, Artificial Intelligences, Networking and Parallel/Distributed Computing*. IEEE, 2009, pp. 501–506.

[58] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," pp. 2980–2988, 2017.

[59] J. Beutel, H. L. Kundel, and R. L. Van Metter, *Handbook of medical imaging*. Spie Press, 2000, vol. 1.

[60] J. Redmon and A. Farhadi, "Yolo9000: better, faster, stronger," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7263–7271.

[61] A. Bochkovskiy, C. Wang, and H. Liao, "Yolov4: Optimal speed and accuracy of object detection. arxiv 2020," *arXiv preprint arXiv:2004.10934*, 2020.

[62] X. Long, K. Deng, G. Wang, Y. Zhang, Q. Dang, Y. Gao, H. Shen, J. Ren, S. Han, E. Ding *et al.*, "Pp-yolo: An effective and efficient implementation of object detector," *arXiv preprint arXiv:2007.12099*, 2020.

[63] Z. Li, C. Peng, G. Yu, X. Zhang, Y. Deng, and J. Sun, "Detnet: A backbone network for object detection," *arXiv preprint arXiv:1804.06215*, 2018.

[64] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," *arXiv preprint arXiv:1905.11946*, 2019.

[65] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," pp. 4510–4520, 2018.

[66] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," pp. 7132–7141, 2018.

[67] J. Wang, P. Fu, and R. X. Gao, "Machine vision intelligence for product defect inspection based on deep learning and hough transform," *Journal of Manufacturing Systems*, vol. 51, pp. 52–60, 2019.

[68] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2117–2125.

[69] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, “Path aggregation network for instance segmentation,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8759–8768.