UNIVERSITAT POLITÈCNICA DE VALÈNCIA

MÁSTER UNIVERSITARIO EN INTELIGENCIA ARTIFICIAL,
RECONOCIMIENTO DE FORMAS E IMAGEN DIGITAL

TRABAJO FIN DE MÁSTER

Análisis del funcionamiento cardíaco mediante redes neuronales

---

Autor:
Álvaro López Chilet

Tutores:
Jon Ander Gómez Adrián
Roberto Paredes Palacios

Pattern Recognition and Human Language Technology (PRHLT)
Departamento de Sistemas Informáticos y Computación

curso académico 2019-2020

Resumen

Álvaro López Chilet

Análisis del funcionamiento cardíaco mediante redes neuronales

La detección de la disminución del funcionamiento cardíaco es un factor clave para el diagnóstico de enfermedades del corazón. Para analizarlo se obtiene el volumen del ventrículo izquierdo al final de la fase de sístole y diástole para estimar la sangre eyectada por el corazón, pero este proceso es lento y quita mucho tiempo a los médicos. En este trabajo se han implementado modelos basados en redes neuronales profundas para estimar el volumen del ventrículo izquierdo, buscando una buena generalización y aplicabilidad de estos para un caso de uso real. Para ello, además se han aplicado técnicas de *model interpretability* para analizar el comportamiento de los modelos y poder tomar más confianza sobre sus predicciones.

Anàlisi del funcionament cardíac per mitjà de xarxes neuronals

La detecció de la disminució del funcionament cardíac és un factor clau per al diagnòstic de malalties del cor. Per a analitzar-ho s'obté el volum del ventricle esquerre al final de la fase de sístole i diàstole per a estimar la sang ejectada pel cor, però aquest procés és lent i lleva molt temps als metges. En aquest treball s'han implementat models basats en xarxes neuronals profundes per a estimar el volum del ventricle esquerre, buscant una bona generalització i aplicabilitat d'aquests per a un cas d'ús real. Per a això, a més s'han aplicat tècniques de *model interpretability* per a analitzar el comportament dels models i poder tindre més confiança sobre les seues prediccions.

Analysis of cardiac function using neural networks

Detection of declining heart function is a key factor in diagnosing heart disease. To analyze it, the volume of the left ventricle is obtained at the end of the systole and diastole phases to estimate the blood ejected by the heart, but this process is slow and takes a lot of time from the doctors. In this work, models based on deep neural networks have been implemented to estimate the volume of the left ventricle, seeking a good generalization and applicability of these for a real use case. For this, model interpretability techniques have also been applied to analyze the behavior of the models and to be able to gain more confidence about their predictions.

Índice general

Índice de figuras

Índice de cuadros

Lista de Abreviaciones

SAX	**S**hort **A**xis
2CH	**2 Ch**amber
4CH	**4 Ch**amber
RVV	**R**ight **V**entricle **V**olume
LVV	**L**eft **V**entricle **V**olume
ESV	**E**nd **S**ystolic **Volume**
EDV	**E**nd **D**iastolic **Volume**
EF	**E**jection **F**raction
MLP	**M**ulti**l**ayer **P**erceptron
CRPS	**C**ontinuous **R**anked **P**robability **S**core
MSE	**M**ean **S**quared **E**rror
MAE	**M**ean **A**bsolute **E**rror
FFT	**F**ast **F**ourier **T**ransform

1 Introducción

Las enfermedades del corazón son una de las principales causas de muerte en el mundo (Zhen y col., 2016), poder diagnosticarlas rápidamente y con previsión es fundamental para poder tratarlas. Un factor clave para el diagnóstico es analizar el funcionamiento del corazón, basándose en la estimación de los volúmenes ventriculares y la cantidad de sangre eyectada a cada latido. Estos son indicadores muy relevantes para detectar enfermedades y posibles riesgos de muerte. Por ejemplo, se sabe que los pacientes con una eyección de sangre baja que han sobrevivido a un ataque cardíaco, tienen más posibilidades de morir en el transcurso del próximo año que los pacientes con un funcionamiento normal del corazón. También existen enfermedades que agrandan el corazón antes de que se aprecien cambios en la cantidad de sangre eyectada, como el caso de una válvula aórtica con fugas graves (Bekeredjian y Grayburn, 2005).

Con el objetivo de obtener los indicadores para el diagnóstico, los médicos cardiólogos emplean mucho tiempo en obtener las medidas con las que estimar los valores volumétricos deseados. Este tiempo de trabajo humano podría reducirse empleando un sistema basado en técnicas de aprendizaje automático que realice la tarea de calcular los volúmenes, dando a los médicos más tiempo para realizar otras tareas como atender a los pacientes.

1.1. Descripción del problema

Uno de los indicadores más utilizados para el diagnóstico es el volumen en mililitros del ventrículo izquierdo al final de las fases de sístole[1] (ESV) y diástole[2] (EDV), con ellos se puede calcular el *ejection fraction* (Fórmula 1.1), el cual nos indica la cantidad de sangre que es expulsada por el corazón en cada latido y con la que se pueden detectar anomalías en el funcionamiento cardíaco.

$$EF = 100 * \frac{EDV - ESV}{EDV} \tag{1.1}$$

Para la obtención de los valores ESV y EDV se utilizan imágenes de resonancia magnética cardiovascular (RMC), este tipo de imagen es el estándar gracias a que su método de adquisición es poco invasivo, emite poca radiación y al mismo tiempo aporta imágenes de buena calidad.

El método actual empleado por los médicos se basa en tomar varias imágenes a lo largo del eje longitudinal del corazón y realizar la segmentación manual del contorno de las cavidades ventriculares, en su punto de más dilatación (diástole) y en el de menos (sístole), para posteriormente calcular el área segmentada por cada contorno y acumular las áreas a lo largo del eje longitudinal para obtener el volumen total del ventrículo. Este proceso es tedioso e introduce cierta subjetividad dependiendo del experto que la realice (Luo y col., 2020; Abdelmaguid

[1] Fase de contracción para bombear sangre a los vasos sanguíneos.
[2] Fase de relajación para que entre la sangre en el corazón.

y col., 2018). Un cardiólogo experto tarda unos 20 minutos en realizar el proceso (Korshunova y col., 2016), por lo que desarrollar un sistema automático podría ahorrar mucho tiempo humano muy valioso y eliminar la subjetividad.

1.1.1. Dataset utilizado

Para este trabajo se han tomado los datos de una competición de *Kaggle*[3], ya terminada, donde el objetivo es estimar el volumen del ventrículo izquierdo al final de la fase de sístole y de diástole. Este conjunto de datos cuenta con 1140 casos con sus respectivas etiquetas de volumen del ventrículo izquierdo, ESV y EDV. Cada caso está compuesto por varios cortes que a su vez pueden pertenecer a diferentes vistas. Cada corte es un vídeo de 30 *frames*, exceptuando algunos casos de menos, donde se aprecia el ciclo completo de un latido. Las diferentes vistas a las que puede pertenecer un corte son las siguientes:

- **Short Axis (SAX):** Este es el tipo de vista básico y con el que todos los casos cuentan con al menos 1 corte y hasta 25 en el mayor de los casos. Este tipo de vista es la que emplean los expertos para realizar el cálculo de los volúmenes. Debido a su localización en plano perpendicular al eje longitudinal del corazón es la vista que mejor muestra los ventrículos izquierdo y derecho. En la Figura 1.1 se puede ver un ejemplo.

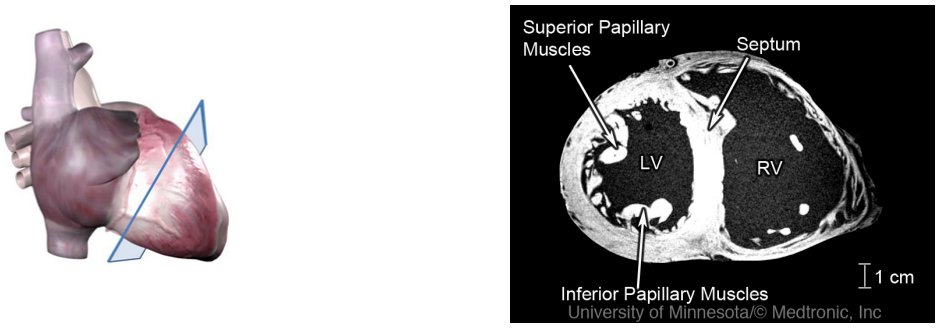


FIGURA 1.1: Visualización de la posición de la vista SAX y un ejemplo de la imagen 2D que genera.

- **2 Chamber (2CH):** Esta es una vista auxiliar con la que no cuentan todos los casos. Desde ella se puede ver el ventrículo izquierdo y parte del atrio izquierdo. En la Figura 1.2 se puede ver un ejemplo.

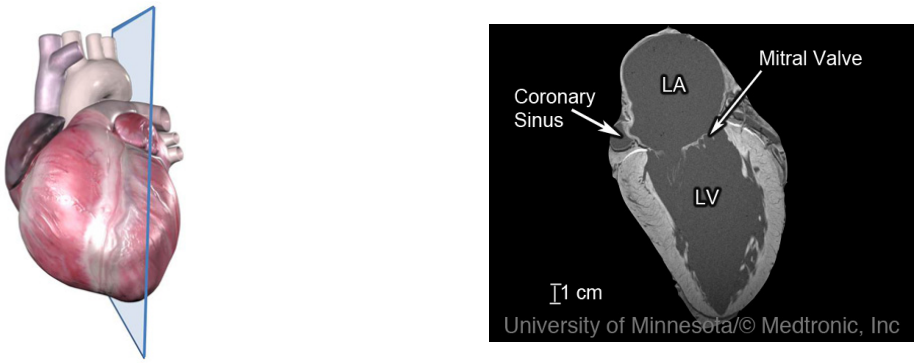


FIGURA 1.2: Visualización de la posición de la vista 2CH y un ejemplo de la imagen 2D que genera.

[3] https://www.kaggle.com/c/second-annual-data-science-bowl/overview

- **4 Chamber (4CH):** Al igual que la vista 2CH, esta también es auxiliar y no se encuentra disponible para todos los casos. Desde ella se pueden ver cuatro cámaras del corazón: el ventrículo izquierdo, el ventrículo derecho, el atrio izquierdo y el atrio derecho. En la Figura 1.3 se puede ver un ejemplo.

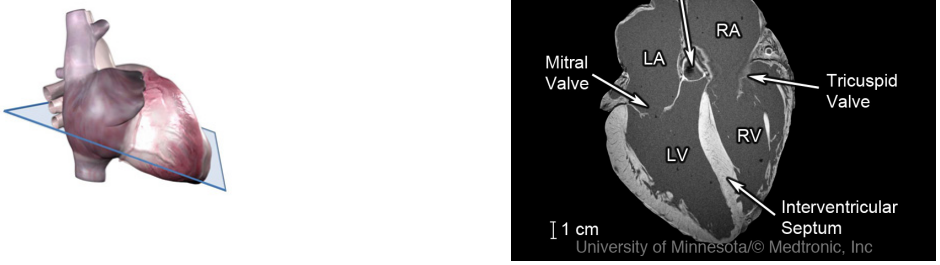


FIGURA 1.3: Visualización de la posición de la vista 4CH y un ejemplo de la imagen 2D que genera.

Este dataset es muy heterogéneo ya que el numero de cortes de cada caso puede variar mucho dificultando la estimación del volumen en los casos más extremos. Además no todos los casos cuentan con todos los tipos de vista previamente comentados (SAX, 2CH y 4CH), siendo el SAX el único que está disponible para todos debido a que es el plano utilizado por los expertos para la segmentación manual.

1.2. Objetivos

Los objetivos propuestos en este trabajo son los siguientes:

- Realizar un análisis exploratorio del conjunto de datos y un estudio de los distintos tipos de preprocesado aplicables para eliminar datos ruidosos y facilitar a los modelos basados en redes neuronales la tarea de estimar volúmenes.
- Desarrollar modelos robustos y capaces de generalizar tal que puedan ser aplicables en casos de uso reales.
- Analizar el comportamiento de los modelos mediante técnicas de *model interpretability*, con el fin de facilitar a los expertos en el dominio de aplicación (en este caso médicos cardiólogos) la comprensión de los resultados obtenidos por los modelos.
- Conseguir la mejor posición posible en el ranking de la competición de *Kaggle* correspondiente al conjunto utilizados en este trabajo.

2 Estado del arte

La estimación automática del volumen ventricular es una tarea compleja que ha sido abordada de múltiples formas. Principalmente se pueden dividir en dos grupos: los métodos basados en segmentación y los de estimación directa. A continuación se van a describir algunos de los trabajos más destacables de cada uno de estos grupos.

2.0.1. Basados en segmentación

Esta metodología se basa en obtener primeramente la segmentación del ventrículo para cada uno de los cortes tomados a lo largo del eje longitudinal del corazón y posteriormente, mediante diferentes aproximaciones, tomar las segmentaciones y calcular el volumen final del ventrículo. Este método es muy parecido a el que emplea un experto cuando calcula el volumen ya que también se basan en la segmentación, en este caso manual, de los ventrículos.

En Avendi, Kheradvar y Jafarkhani, 2016 utilizan un dataset de 45 casos donde además de los valores de ESV y EDV también contiene la segmentación de la cámara del ventrículo izquierdo. A partir de estos datos, primero entrenan una red neuronal convolucional que toma como entrada un corte 2D y a la salida devuelve una máscara que delimita la región de interés (el *ground truth* de estas máscaras se obtiene a partir de las segmentaciones del dataset). Tras este primer paso de extracción de la región de interés utilizan un *stacked autoencoder* convolucional como modelo de segmentación. Debido a la metodología seguida para el entrenamiento de un *stacked autoencoder*, los autores afirman que ha ayudado a lidiar con el *overfitting* y mejorar la generalización, ya que al tratarse de datasets pequeños es fácil que los modelos acaben haciendo *overfitting*. Tras obtener la segmentación de los cortes 2D utilizan modelos deformables para aproximar el volumen del ventrículo, estos modelos son contornos dinámicos que van evolucionando siguiendo una función de energía hasta alcanzar su mínimo cuando el contorno está sobre el objeto de interés. De forma que las segmentaciones son utilizadas como método de inicialización de los modelos deformables. Como *data augmentation* han empleado rotación, translación y variación de la intensidad de los píxeles.

En Liao y col., 2019[1] emplean el mismo conjunto de datos de la competición de *Kaggle* que ha sido utilizado para este trabajo, además incluyen el dataset *Sunnybrook*[2] el cual es más reducido, con 45 muestras de diferentes pacientes donde entre las etiquetas se encuentra la segmentación del ventrículo izquierdo. De los datos de *Kaggle* detectan manualmente para todo el dataset los *frames* del final de las etapas de sístole y diástole, y además extraen solo 6 de los 30 *frames* originales de manera que el vídeo siempre empiece y acabe en diástole, con el *frame* de sístole en el centro. Este etiquetado y extracción manuales son una tarea laboriosa que en caso de contar con un dataset grande no sería viable de realizar. Con los datos ya preparados, su *pipeline* comienza con un modelo de detección de regiones de interés, este se basa en un primer modelo que propone múltiples regiones y otro que da una puntuación a cada una para poder quedarse así con la mejor. El modelo que propone las regiones ha sido entrenado con datos sintéticos, ya

[1] Los autores de esta publicación han quedado los cuartos de la competición de *Kaggle*.

[2] Para más información sobre este dataset consultar `https://www.cardiacatlas.org/studies/sunnybrook-cardiac-data/`.

que lo han entrenado con los datos de *Kaggle* y dando como regiones de interés objetivo unas regiones generadas con un modelo entrenado con los datos de *Sunnybrook*, el cual sí cuenta con la segmentación manual y validada. De las regiones propuestas por el modelo generador, han etiquetado manualmente las que eran correctas y las que no, para poder entrenar el modelo que dé las puntuaciones a las regiones propuestas. Una vez extraída la región de interés, utilizan un modelo convolucional con dos salidas, una para segmentación y otra para regresión, este se puede ver en la figura 2.1. La parte que realiza la regresión también toma como entrada la segmentación concatenada, por lo que la estimación va a depender de la segmentación en gran parte. Por ello este modelo se entrena alternando entre *epochs* de segmentación y de regresión, ya que las *epochs* de regresión se entrenan con los datos de *Kaggle* y las de segmentación con los de *Sunnybrook*. Como *data augmentation* solo utilizan rotaciones de 90º, 180º y 270º.

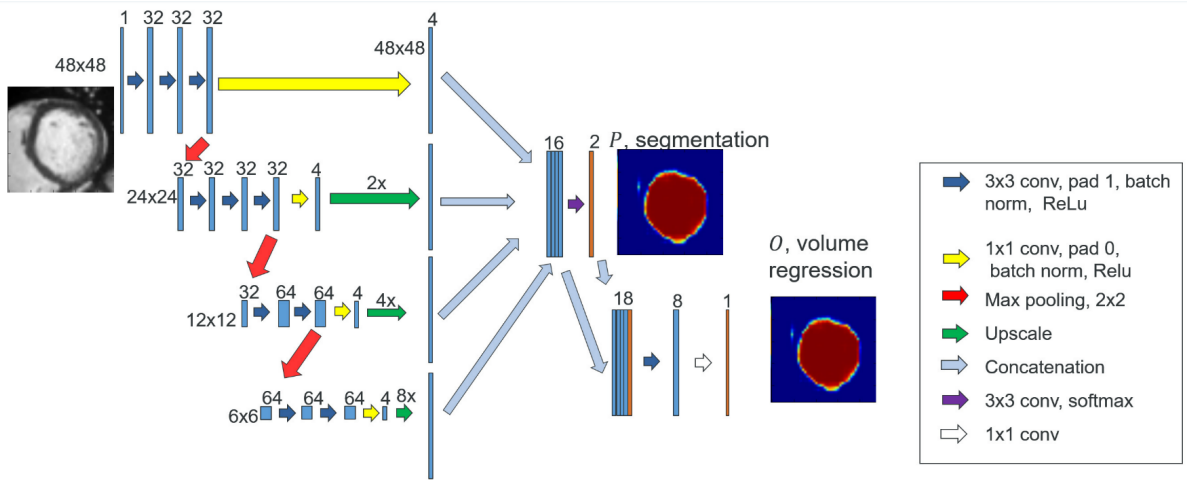


FIGURA 2.1: Topología del modelo empleado por Liao y col., 2019. Utiliza una estructura por columnas con diferentes escalados de la imagen. Tiene dos salidas, una para segmentación (*P*) y otra para regresión (*O*).

En Abdelmaguid y col., 2018 utilizan tres conjuntos de datos: el de la competición de Kaggle para realizar la regresión de los valores del volumen; el *Sunnybrook* comentado anteriormente para segmentación; y el *Automated Cardiac Diagnosis Challenge* (ACDC) que también es para segmentación. Como primer preprocesado, utilizan los metadatos de las imágenes para orientarlas todas de la misma forma mediante rotaciones, y las reescalan de forma que todas tengan unos píxeles que representen 1mm^2 en volumen real[3]. En esta publicación implementan un sistema de extracción de la región de interés, el cual no depende de datos anotados como los vistos anteriormente. Este se basa en la utilización de la transformada rápida de Fourier que permite detectar el movimiento[4] (ya que disponemos de vídeos de 30 *frames* por cada corte del corazón). Gracias a esto, se puede obtener un mapa de intensidades sobre la imagen donde se ven más resaltadas las zonas con más movimiento. A partir de este mapa emplean *clustering* mediante *k-means* para eliminar las zonas con poca intensidad en los píxeles, y una vez acotada la región, emplean detección de círculos mediante la transformada de Hough, la cual es una técnica de visión por computador que permite detectar figuras que puedan ser expresadas matemáticamente (como un círculo); finalmente utilizan los círculos detectados para calcular el rectángulo que delimita la región de interés. En la figura 2.2 se pueden ver los pasos realizados durante la extracción.

[3]Para saber la cantidad de zoom a aplicar para el reescalado utilizan el metadato de «PixelSpacing» el cual muestra el volumen real que representan los píxeles.

[4]Para más información sobre este método consultar Petitjean y Dacher, 2011.

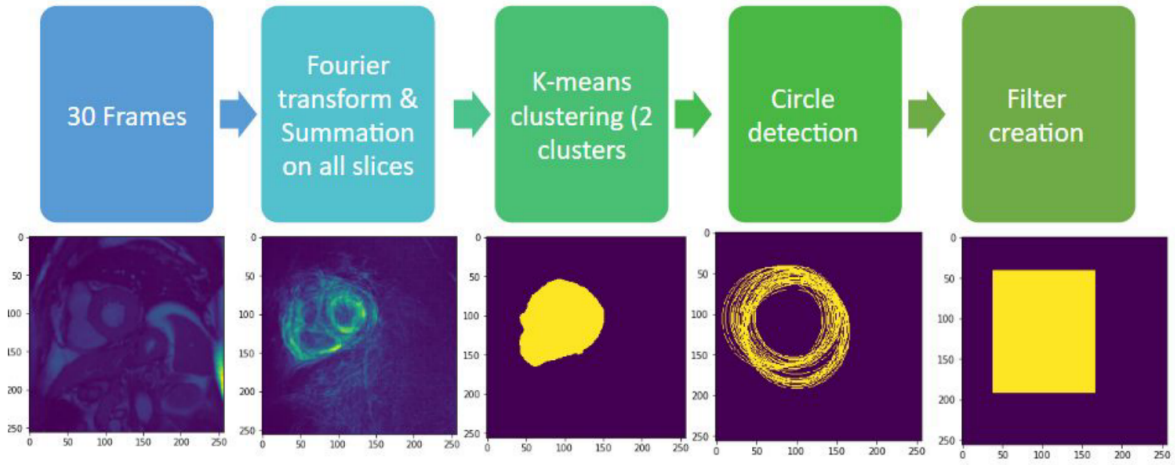


FIGURA 2.2: Pasos realizados para la extracción de la región de interés en la publicación Abdelmaguid y col., 2018.

Estas regiones extraídas son la entrada de un modelo de segmentación utilizando los datos de *Sunnybrook* y ACDC. Concretamente utilizan una topología muy popular en el campo de la imagen médica que es la U-Net (Ronneberger, Fischer y Brox, 2015), esta se caracteriza por funcionar muy bien con conjuntos de datos reducidos como en este caso. Una vez obtenidas las segmentaciones, se puede calcular el área de cada corte en mm^2; combinándolo con los metadatos como la separación entre los cortes, se pueden obtener todos los valores necesarios para aproximar el volumen del ventrículo mediante el cálculo de los volúmenes de conos truncados, donde cada cara son dos cortes adyacentes, y posteriormente sumando todos los volúmenes de los conos se obtiene el volumen total del ventrículo. Una vez obtenido el modelo entrenado, utilizan los datos de la competición de *Kaggle* solo para la fase de test, por no tener las etiquetas de segmentación. Como *data augmentation* para el entrenamiento de la U-Net han utilizado: rotación de 90º, translación vertical y horizontal de 0-5 % de la imagen y zoom de 0.95-1.05.

En Tencia y Woshialex, 2016[5] utilizan el dataset de *Sunnybrook* para segmentar y además le añaden entorno a 200 casos más al conjunto de entrenamiento con la segmentación etiquetada por ellos mismos, tomando estos casos del conjunto de *Kaggle*. Los autores afirman que estas muestras etiquetadas manualmente mejoran considerablemente el rendimiento de la red. Para el preprocesado de las imágenes aplican la extracción de la región de interés mediante la detección del movimiento como en Abdelmaguid y col., 2018, utilizando la transformada rápida de Fourier. Además, rotan todas las imágenes para que partan de la misma orientación. Como *data augmentation* aplican rotaciones aleatorias, translación y normalización del contraste. Para la realización de la segmentación utilizan un *ensemble* de 10 modelos convolucionales, donde cada uno toma como entrada las imágenes con un tamaño y preprocesado distinto. Para el cálculo final del volumen, calculan el volumen de cada corte multiplicando el área de la segmentación por el grosor del corte, y posteriormente suman los volúmenes para todos los cortes.

En Wit, 2016 utilizan solo el conjunto de datos de la competición de *Kaggle*. Este dataset no provee las segmentaciones, por lo que el autor ha segmentado manualmente todas las imágenes que ha utilizado para el entrenamiento, entorno a 2000 imágenes las cuales han sido obtenidas a partir de extraer manualmente los *frames* del final de las etapas de sístole y diástole para todas las muestras. Por lo que solo utiliza dos *frames* (de los 30 originales) por cada corte. Para el preprocesado, primero emplea un escalado de las imágenes de manera que cada píxel

[5]Los autores son los primeros de la competición de *Kaggle*.

represente 1mm^2 de volumen real, y finalmente realiza un *crop* central de 180x180 píxeles con el que afirma capturar la región de interés para todas las imágenes. Para segmentar utiliza un modelo convolucional con la arquitectura U-Net (Ronneberger, Fischer y Brox, 2015). De las segmentaciones obtenidas de cada corte y mediante los metadatos, como la posición de los cortes y su grosor, calcula el volumen final del ventrículo.

La aproximación a partir de la segmentación tiene algunos inconvenientes debido a su dependencia de un buen segmentado y disposición de los cortes. Por ejemplo, en Abdelmaguid y col., 2018 dicen que han tenido problemas con casos donde la segmentación les saca varias regiones, llevando a confusión, ya que solo hay un ventrículo izquierdo. Lo que les ha llevado a realizar una solución *ad hoc* para eliminar los falsos negativos. Este método también presenta grandes dificultades al tratar casos con un reducido número de cortes, debido a que la aproximación de utilizar metadatos para estimar el volumen final, como si fuera una figura geométrica, se vuelve muy imprecisa; para estos casos en algunos trabajos utilizan un modelo de regresión lineal que toma el sexo y la edad para predecir los volúmenes (Wit, 2016; Tencia y Woshialex, 2016; Abdelmaguid y col., 2018). También existe el problema de casos donde hay un buen número de cortes pero estos no están repartidos de principio a fin, por lo que se pueden dejar un trozo del extremo por estimar.

2.0.2. Estimación directa

Este método se basa en realizar una estimación directa de los volúmenes de sístole y diástole a partir de las imágenes, sin tener que realizar una segmentación previa. Por lo que para este tipo de aproximación, se aplican modelos de regresión que estimen los valores volumétricos directamente tomando como entrada las imágenes preprocesadas. Este método necesita de menos metadatos que el basado en segmentación y además puede llegar a captar características en las imágenes que puedan ser relevantes, y que en la aproximación por segmentación no se pueden tener en cuenta ya que no se entrenan de la misma manera *end-to-end*.

En Korshunova y col., 2016[6] utilizan únicamente el dataset de la competición de *Kaggle*. Concretamente aprovechan además de las imágenes SAX, las imágenes 2CH y 4CH. Para la extracción de la región de interés aplican dos técnicas: una basada en la transformada rápida de Fourier para detectar movimiento, al igual que en Abdelmaguid y col., 2018; y otra utilizando las vistas 2CH y 4CH con sus metadatos, con los que se calcula la intersección en el espacio 3D obteniendo así el centro de la región de interés. Este segundo método se aprovecha de los metadatos para saber las posiciones de cada corte en el espacio, con las que obtener las intersecciones y posteriormente proyectarlas sobre cada imagen consiguiendo así el centro de la región de interés en todas las vistas, el problema principal que presenta es que no todos los casos disponen de las imágenes 2CH y 4CH. Además, las imágenes se reescalan para que todas tengan la misma resolución en milímetros de volumen real representado. Como *data augmentation* aplican rotaciones aleatorias, desplazamientos, zoom (corrigiendo posteriormente las etiquetas según el zoom aplicado) y *shift* de los *frames* en el eje temporal[7]. En cuanto a los modelos empleados para la regresión utilizan diferentes tipos, todos ellos basados en la arquitectura de la VGG-16 (Simonyan y Zisserman, 2014), estos se diferencian según sus datos de entrada: por paciente, donde toma como input todos los cortes del paciente haciendo *padding* hasta 22 cortes[8] en caso de no tenerlos; por corte, donde como entrada toma los 30 *frames* de un corte; y los modelos de 2CH y 4CH que también toman solo un corte como entrada. En el caso de los modelos por paciente, emplean los modelos de un corte como pesos preentrenados ya que el modelo tiene

[6] Los autores son los segundos de la competición de *Kaggle*.

[7] Aunque los cortes suelan empezar por el *frame* de sístole esto aveces no se cumple.

[8] 22 es el número máximo de cortes que tiene un paciente en la partición de entrenamiento del dataset de *Kaggle*.

22 entradas de un solo corte; cabe destacar que en caso de aplicar *padding* por no tener 22 cortes, estos cortes no se tienen en cuenta en la estimación final de la red. En la figura 2.3 se puede apreciar un esquema del modelo por paciente. Los modelos que han implementado tienden a hacer *overfitting*, por lo que además del *data augmentation* que aplican, también utilizan *dropout* con una probabilidad p=0.5 para lidiar con este problema.

Para la estimación final durante la fase de test, utilizan un *ensemble* de los modelos previamente comentados, donde la media final se calcula de forma ponderada dando más peso a los modelos que han dado mejores resultados durante la validación. Además, en caso de no disponer de alguna vista como la 2CH o 4CH, su respectivo modelo no se utiliza y por tanto se reescalan los pesos de los demás modelos para el cálculo de la media ponderada. Los modelos también pueden ser eliminados en caso de dar un valor estimado muy alejado de la media del resto de modelos.

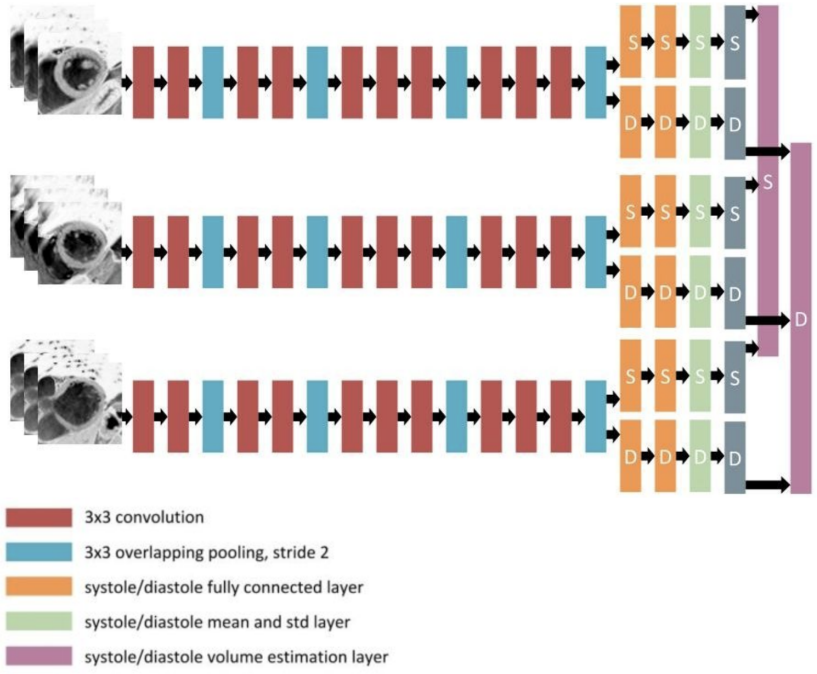


FIGURA 2.3: Estructura del modelo por paciente de Korshunova y col., 2016.

En Luo y col., 2020 utilizan la mayor cantidad de datos vista mediante la combinación de múltiples datasets: STACOM2011 (200 casos con LVV), RV2012 (48 casos con RVV), ACDC2017 (150 casos con LVV y RVV) y el de *Kaggle* (1140 casos con LVV). De los cortes disponibles por cada caso extraen el corte de arriba, el central y el 2CH (figura 2.4); ya que como muestran en su previo trabajo (Luo y col., 2018), son las mejores vistas para estimar el volumen. Una vez extraídos los cortes de interés, solo toman de ellos los *frames* del final de las etapas de sístole y diástole, por lo que de cada caso acaban tomando solo seis imágenes 2D.

Como preprocesado, primero localizan la región de interés mediante la intersección de las vistas SAX y 2CH en el espacio 3D, luego reescalan las imágenes para que cada píxel ocupe el mismo espacio real de 1.4mm^2, después aplican normalización de la intensidad de los píxeles y finalmente extraen un parche de 100x100 píxeles sobre el centro de la región de interés. Durante el entrenamiento, aplican *data augmentation* mediante rotaciones aleatorias y *flipping*.

El modelo que proponen para realizar la regresión es un modelo con una construcción dinámica (figura 2.5). Este está pensado para poder utilizar diferentes topologías de redes convolucionales como *backbone* de la arquitectura. La construcción dinámica se basa en que comienzan a

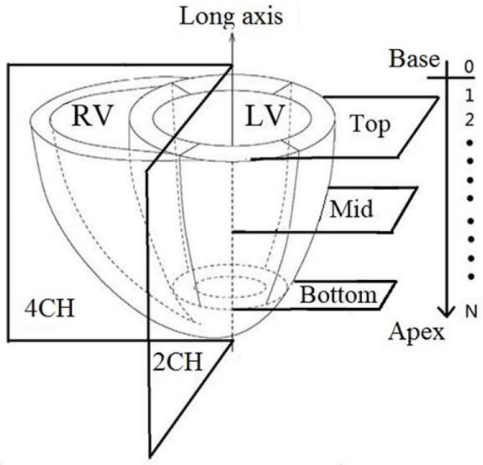


FIGURA 2.4: Ejemplos de tipos de adquisición de imágenes sobre el corazón (imagen extraída de Luo y col., 2018).

entrenar un modelo convolucional elegido como *backbone* tanto con *frames* de de la fase de sístole como de diástole, pasando para cada caso la etiqueta correspondiente (ESV o EDV). Tras realizar este entrenamiento, se pasa a una segunda fase donde los pesos se duplican para obtener dos copias de la misma red, a partir de este punto, una de ellas se especializa en calcular el volumen de sístole y la otra el de diástole, por lo que se realiza otra fase de entrenamiento pero esta vez pasando los *frames* de sístole y diástole a su modelo correspondiente. Además, para el entrenamiento de la segunda fase se añade un factor adicional a la función de perdida, donde además del error cuadrático ya empleado para los volúmenes, se añade como factor regularizador el error cuadrático del Ejection Fraction (EF) controlado por un hiperparámetro. Tras la segunda fase, se pasa ya a la última en la que se añaden dos nodos residuales, uno por cada red (sístole y diástole), que van a estimar el error producido por la red al predecir los volúmenes. Estos nodos toman como entrada la salida de la última capa *fully connected* de su red correspondiente. Los pesos que los conectan se inicializan por un método estocástico antes de empezar el entrenamiento *end-to-end*. El objetivo de estos nodos es realizar una corrección del error que produce la red para así aumentar la precisión.

Los modelos que han probado como *backbone* han sido VGG-16 (Simonyan y Zisserman, 2014), ResNet-50 (He y col., 2015) y DenseNet-121 (Huang, Liu y Weinberger, 2016), donde DenseNet-121 ha sido el que mejores resultados ha obtenido.

En Zhen y col., 2016 utilizan un conjunto de datos de 47 casos sin etiquetar y 100 etiquetados, tanto con LVV como RVV. Su aproximación se basa en primero extraer las características de las imágenes mediante aprendizaje no supervisado y posteriormente utilizaras con *random forests* para realizar la regresión. Para el preprocesado realizan una extracción de la región de interés igual que en Abdelmaguid y col., 2018 basada en la transformada rápida de Fourier para detectar el movimiento.

Su modelo propuesto se denomina «multi-scale convolutional deep belief network» (MCDBN), este se basa en un primera *convolutional restricted Boltzmann machine* (CRBM) con tres entradas de distinto tamaño para los filtros (de 17x17, 13x13 y 9x9), tras cada entrada se utilizan dos capas convolucionales que extraen las características de las imágenes. Posteriormente, las características extraídas pasan a una *restricted Boltzmann machine* (RBM) para obtener una representación todavía más compacta en un vector de 3000 valores. Con este modelo realizan un aprendizaje no supervisado utilizando todas las imágenes del dataset, con y sin etiqueta, para

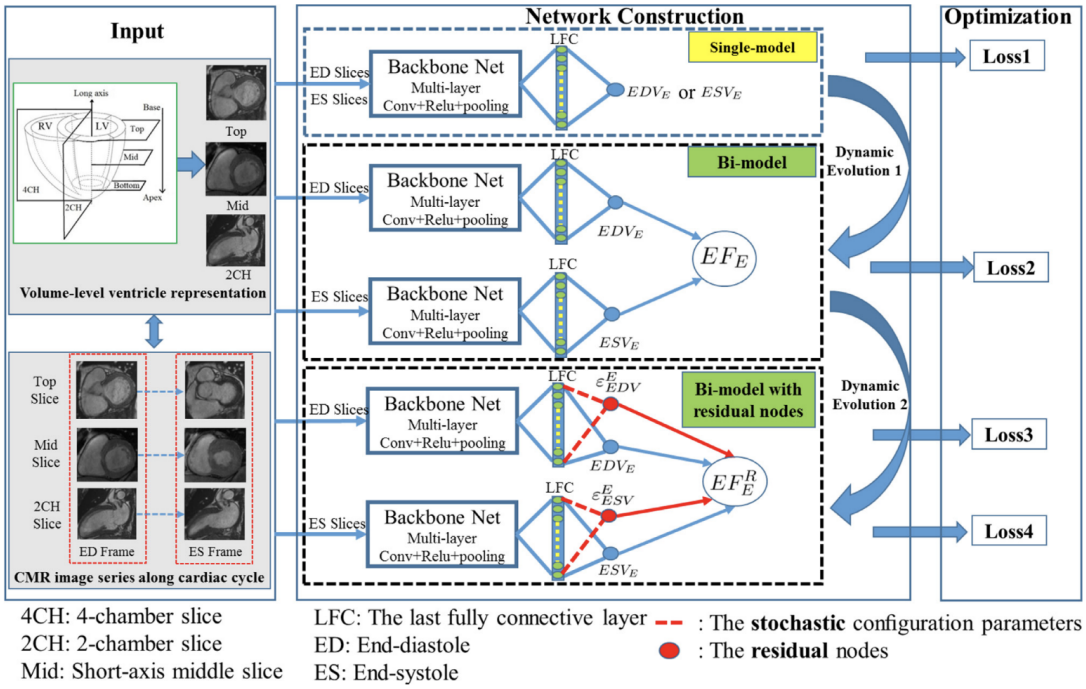


FIGURA 2.5: Modelo dinámico propuesto por Luo y col., 2020.

que el modelo aprenda a codificar las características de las imágenes y lograr mejores representaciones. Posteriormente, utilizando solo muestras etiquetadas, entrenando así un modelo de *random forests* con las representaciones creadas por el MCDBN que realice la regresión de los valores LVV y RVV.

En Xue y col., 2018 utilizan un conjunto de datos de 145 casos recolectados y etiquetados por ellos en colaboración con varios hospitales. Como preprocesado extraen las regiones de interés, aunque sin decir como, pero dan a entender que de forma manual, ya que es un paso previo a el etiquetado manual de las imágenes que realizan. La característica principal de su método es que utilizan redes neuronales recurrentes, para intentar modelar las relaciones de la secuencia de *frames* que conforman el vídeo de cada corte. Concretamente su modelo comienza con una red convolucional de 5 capas, que a la salida devuelve un vector de 100 valores con las características extraídas de la imagen. Por esta red se pasa cada *frame* del corte para obtener su vector de características a modo de *embedding*. Con estos *embeddings* se genera una secuencia que entra en dos redes LSTM paralelas, una que estima varios valores volumétricos del ventrículo izquierdo, entre ellos el volumen ventricular, y otra que estima la fase cardíaca de cada *frame* (sístole o diástole). En la figura 2.6 se puede apreciar la estructura completa del modelo.

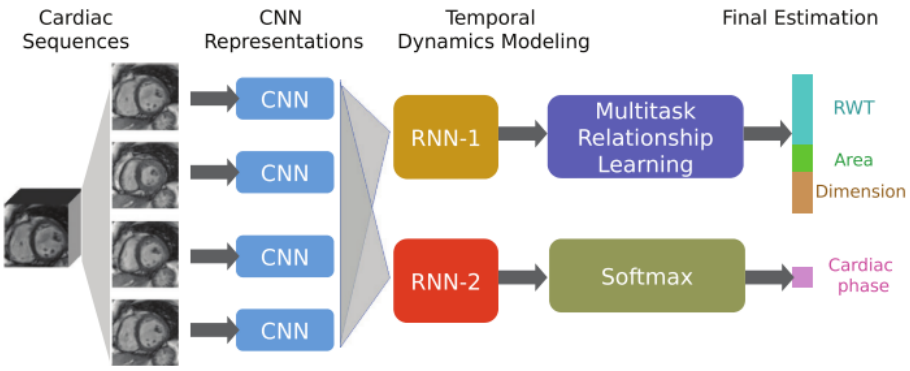


FIGURA 2.6: Estructura del modelo recurrente empleado en Xue y col., 2018.

3 Experimentación

En este apartado se van a describir con detalle los pasos, técnicas y herramientas empleadas para el desarrollo de las *pipelines* propuestas. Desde el análisis de los datos, pasando por el preprocesado, los modelos de redes neuronales implementados y finalmente la metodología empleada para el entrenamiento e inferencia de los modelos.

De las dos principales aproximaciones presentadas en el estado del arte, por segmentación y por estimación directa, se ha optado por la segunda. La estimación basada en segmentación se ha descartado debido a que para implementarla hay que utilizar otros conjuntos de datos que posean etiquetas de segmentación y además esta aproximación no puede estimar correctamente los casos donde hay pocos cortes; en la literatura utilizan modelos de regresión basados en la edad y sexo para los casos con pocos cortes. Esta opción queda descartada en nuestro caso ya que hemos buscado tomar una aproximación que sea aplicable en casos de uso reales, por lo que no podemos utilizar modelos que no tengan en cuenta el contenido de las imágenes.

Nuestra implementación se basa en utilizar un modelo para estimar el volumen de sístole y otro modelo para diástole. Además, la estimación se realiza para cada corte del caso individualmente, de manera que para calcular los volúmenes finales se calcula la media de los valores estimados de cada corte del caso.

Estos pares de modelos para sístole y diástole se entrenan por cada tipo de vista. De manera que también se ha utilizado un *ensemble* dinámico de los tres tipos de modelos (SAX, 2CH y 4CH), utilizando para cada caso las vistas disponibles, ya que no todos los casos tienen los tres tipos de vistas. Este *ensemble* obtiene los volúmenes estimados de cada modelo para cada vista y realiza la media aritmética para obtener los volúmenes finales de ESV y EDV.

3.1. Herramientas

Todos los scripts y librerías han sido implementadas utilizando *Python* 3, por su comodidad y rapidez para programar nuevos experimentos y el gran repertorio de librerías disponibles. Estas características han hecho de este lenguaje el más popular dentro del campo de *deep learning* ya que las principales librerías de este campo tienen sus APIs enfocadas a *Python*. En nuestro caso, hemos utilizado la librería *Pytorch* para desarrollar los modelos por su flexibilidad a la hora de experimentar con las topologías de las redes neuronales.

Otras librerías relevantes de *Python* utilizadas han sido: *Pydicom* para leer las imágenes en formato DICOM; *scikit-image* y *OpenCV* para funciones de preprocesado de imagen; *NumPy* para manejo de matrices; y *pandas* para trabajar con los archivos con formato *csv*.

En cuanto al *hardware* empleado para la experimentación, se han utilizando dos máquinas con procesadores Intel i5-8600K de 6 núcleos y dos tarjetas gráficas NVIDIA GeForce GTX 1080 de 8GB por cada máquina. Estas han sido proporcionadas por el grupo de investigación *Pattern Recognition and Human Language Technology* (PRHLT).

3.2. Tratamiento de los datos

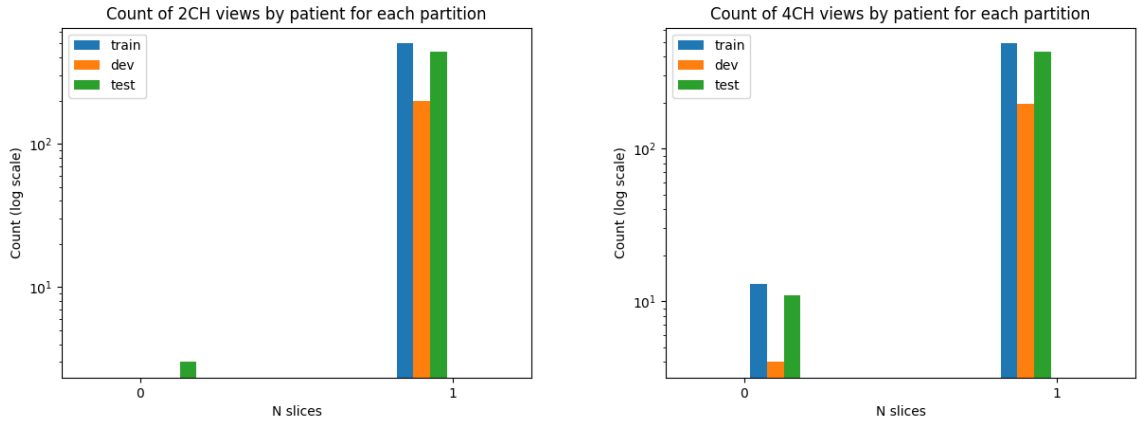
El conjunto de datos utilizado proviene de una competición *Kaggle* en la que ya vienen los datos con una partición propuesta. De los 1140 casos totales, 500 son del conjunto de entrenamiento, 200 de validación y 440 de test. Por cada caso disponemos de sus etiquetas de volumen del ventrículo izquierdo al final de las etapas de sístole (ESV) y diástole (EDV). Durante la competición no se disponían de las etiquetas del conjunto de test, pero al esta terminarse, las etiquetas han sido publicadas, por lo que hemos contado con ellas.

Para nuestra experimentación, hemos respetado las particiones para poder comparar posteriormente resultados con otros trabajos y tener un resultado justo en la clasificación en la competición, que aunque por el momento del desarrollo de este trabajo esté cerrada, aun permite hacer entregas para ver en que puntuación de la métrica obtendrías y así comparar con la clasificación final de la competición.

Como se ha comentado en la introducción en el apartado de los datos, el dataset es muy heterogéneo ya que la cantidad de datos en cada caso puede variar. Por lo que a continuación se va a mostrar el análisis realizado para la detección de toda la variabilidad del dataset y, posteriormente, los tipos de preprocesado implementados con el objetivo de sacar más partido a los datos.

3.2.1. Análisis

En primer lugar se ha realizado un recuento de los tipos de vista disponibles dentro de cada caso. Los tipos de vista disponibles son: SAX, 2CH y 4CH. Respecto a las imágenes SAX, todos los casos cuentan con este tipo de vista ya que es la vista que utilizan los expertos para el etiquetado manual, por lo que siempre está disponible. Las vistas 2CH y 4CH son auxiliares y pueden no aparecer en algunos casos, en la Figura 3.1 se aprecia un recuento del número de cortes con estas vistas por cada caso del dataset (además también se muestra por partición). Se puede ver como son muy pocos los casos donde no está disponible alguna de estas vistas y además en caso de disponer de alguna de ellas solo hay un corte.



(A) 2CH (B) 4CH

FIGURA 3.1: Recuento del número de cortes 2CH y 4CH por cada caso de cada partición.

Para conocer mejor los casos que no disponen de alguna vista 2CH o 4CH se ha analizado que vista es la que les falta o si en algunos casos faltan las dos, este recuento se muestra en la Figura

3.2 donde se puede apreciar que por lo general suelen faltar las vistas 4CH antes que 2CH, y solo para el conjunto de test hay dos casos con 4CH pero no 2CH y uno donde faltan las dos.

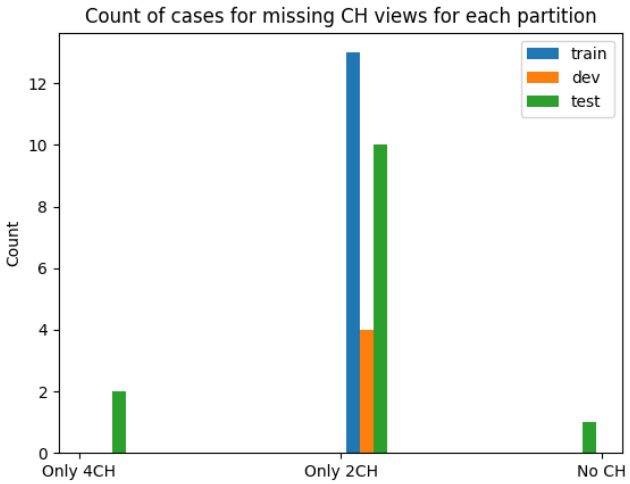


FIGURA 3.2: Diferentes casos de falta de alguna de las imágenes 2CH o 4CH.

En el caso de las vistas 2CH y 4CH solo disponemos de un corte por cada una de ellas en caso de tenerlas disponibles. Un corte implica una secuencia de 30 imágenes (*frames*) que componen un vídeo del ciclo cardíaco visto desde esa vista y ese corte en concreto.

A diferencia de las vistas 2CH y 4CH la vista SAX suele presentar más de un corte a lo largo del eje longitudinal del corazón. En la Tabla 3.3 se muestra un recuento con el número de cortes por cada caso para la vista SAX. El número de cortes más habitual es 11 para los tres subconjuntos del dataset, luego se dan algunos casos con unos pocos más o menos y finalmente existen *outliers* como los casos con menos de 5 cortes los cuales pueden complicar la estimación del volumen. Estos *outliers* afectan mucho más al resultado en caso de utilizar aproximación basada en segmentación que una de regresión directa, ya que a la hora de acumular las áreas segmentadas no se puede realizar una aproximación precisa por la falta de cortes. Por ello, en este trabajo hemos realizado estimación directa, para así evitar tener que usar modelos auxiliares como en algunos trabajos comentados en la sección del estado del arte, donde utilizan regresión lineal con la edad y el sexo para la estimación del volumen de los *outliers*. Esta es una aproximación que puede servir para obtener una mejor puntuación en la competición, pero en nuestro caso buscamos implementar un modelo más robusto y que pueda generalizar para cualquier caso.

También se han analizado el número de *frames* que se disponen por cada corte, ya que en principio los cortes son vídeos de 30 *frames* con el ciclo cardiaco completo. Tras hacer un recuento, se ha visto como existen cortes con menos *frames* y otros con bastantes más, este recuento se puede ver en la Figura 3.4. Los casos con más de 30 cortes poseen un múltiplo de 30, ya que se debe a que hay varias adquisiciones de las imágenes, donde se supone que la última adquisición es la de mejor calidad[1]. Para los cortes con menos de 30 *frames*, encontramos algunos con 21 y 22, los cuales sí que presentan una falta de datos respecto al resto de muestras que hay que solucionar en el preprocesado.

Cada *frame* de un corte viene en un archivo DICOM, este formato es el estándar de imagen médica a nivel internacional, el cual proporciona un formato tanto para las imágenes como los

[1] A partir del nombre del archivo se puede saber a que adquisición pertenece cada *frame*.

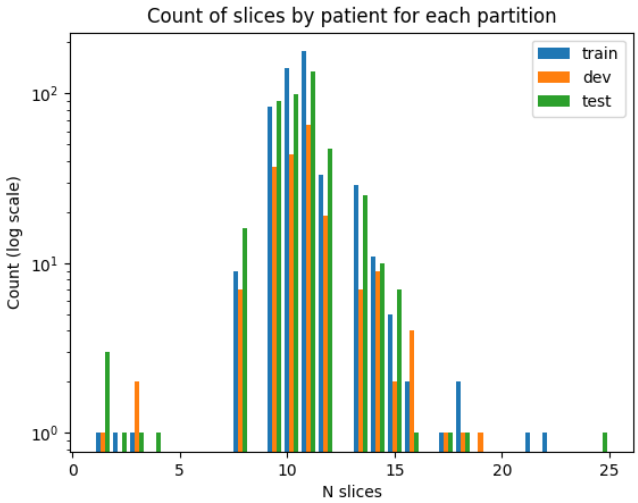
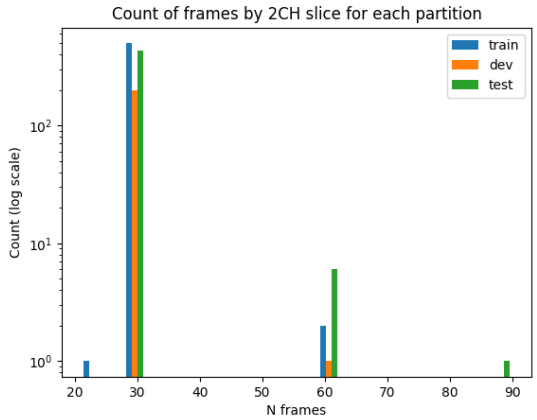
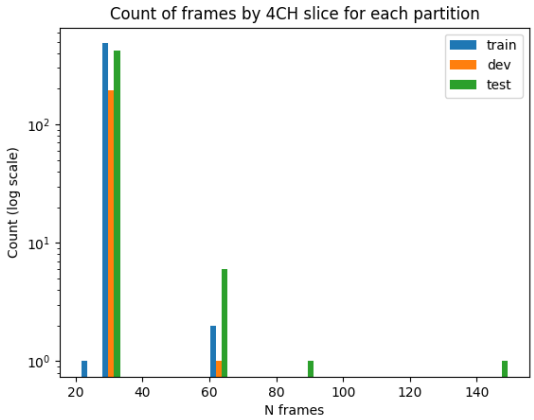


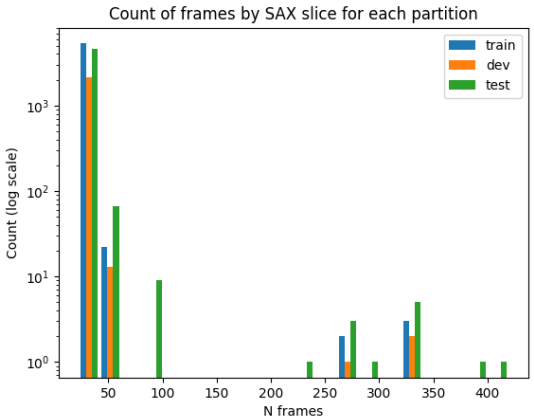
FIGURA 3.3: Recuento del número de cortes SAX por cada caso.



(A) 2CH



(B) 4CH



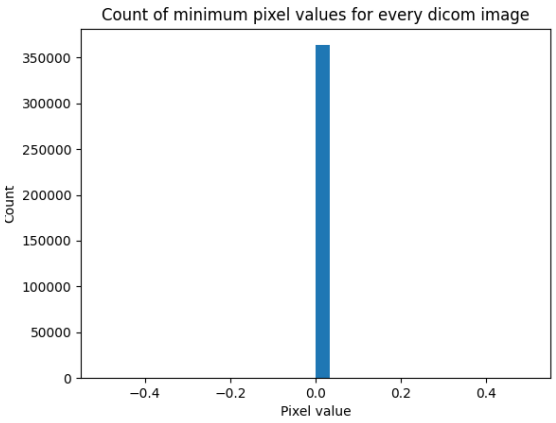
(C) SAX

FIGURA 3.4: Recuento del número de *frames* por cada corte.

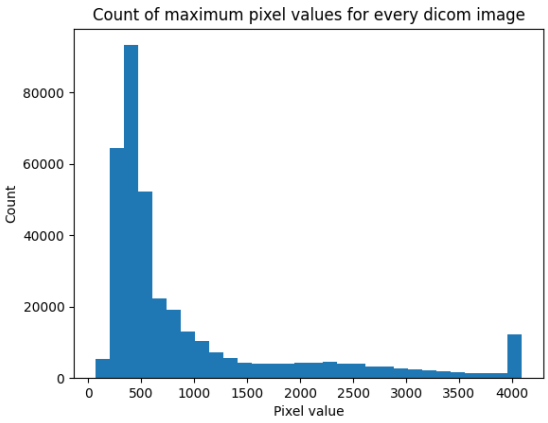
metadatos relacionados con la misma. El rango de valores de los píxeles puede ser muy variable en este tipo de imágenes, ya que no es como un archivo *png* donde los valores se encuentran en

un rango determinado de 0 a 255. En este caso cada píxel corresponde a la intensidad del campo magnético captada por el dispositivo de adquisición en esa posición, por lo que obtenemos una imagen en escala de grises con un mapa de intensidades del campo magnético.

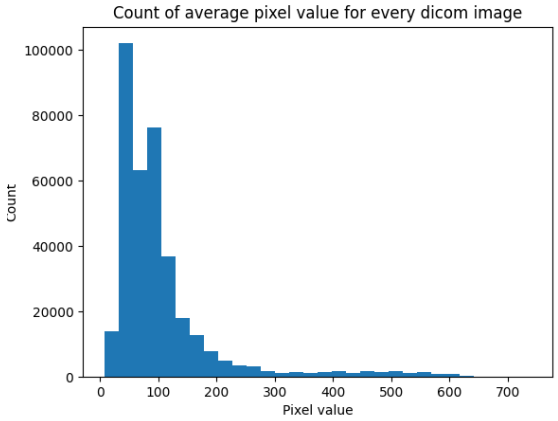
Hemos analizado los valores de los píxeles de todas las imágenes para ver en que rango de valores se encuentran y detectar posibles anomalías. En la Figura 3.5 se puede ver el recuento de los valores máximos, mínimos y promedios de los píxeles para todas imágenes. Donde podemos observar que hay algunas imágenes que se desvían mucho de la media y que poseen algunos píxeles con intensidades muy elevadas. Por lo que es interesante aplicar algún tipo de normalización de las intensidades para eliminar estos picos de intensidad, aportando así algo de regularización y evitando un posible *overfitting*.



(A) Valores mínimos



(B) Valores máximos



(C) Valores promedio

FIGURA 3.5: Análisis de los valores de los píxeles para todas las imágenes.

En cuanto al tamaño de las imágenes, este puede variar mucho. Los tamaños más comunes son: 256x192, 192x256 y 512x384. Además, el tamaño más grande por dimensión es de 736 píxeles y de 120 como mínimo. El recuento total de tamaños se encuentra en el Apéndice A.

Como se ha comentado previamente, las imágenes vienen en formato DICOM, el cual también posee algunos metadatos que pueden ser relevantes. Algunos valores interesantes pueden ser la edad del paciente, el sexo... Y también existen otros más relacionados con la imagen, como el grosor de los cortes en espacio real, el volumen real que representa cada píxel o incluso la posición del corte en el espacio. En las aproximaciones basadas en segmentación, utilizan estos valores de grosor de los cortes y volumen de los píxeles para poder calcular el volumen final a

partir de las segmentaciones. En nuestro caso hemos utilizado el volumen de los píxeles para mejorar el preprocesado como se muestra en el siguiente apartado.

Finalmente, se ha realizado un recuento de los valores de volumen a estimar, los cuales representan los volúmenes en mililitros del ventrículo izquierdo tanto al final de la fase de sístole (ESV) como de diástole (EDV). En la Figura 3.6 se puede ver un conteo de los valores de las etiquetas para todos los casos del dataset. Puesto que sístole es la fase de contracción y diástole la de expansión, los valores de diástole son por lo general más altos. Cabe destacar, que existen casos que se desvían bastante de la media, tanto por debajo como por encima, los cuales pueden presentar problemas a la hora de estimar sus volúmenes. Un detalle que puede ser relevante en cuanto a la dificultad de estimación de cada etiqueta, es que los valores de EDV cubren un rango más amplio que los de ESV, que como veremos en los resultados, puede influir en que la estimación de ESV sea más precisa que EDV.

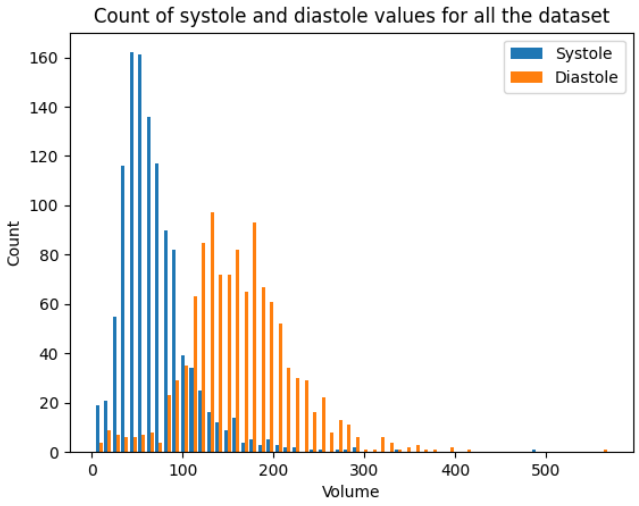


FIGURA 3.6: Conteo de los valores de las etiquetas para todo el dataset.

3.2.2. Preprocesado

Como se ha visto en el análisis de los datos, el dataset presenta muchas irregularidades que hay solucionar antes de poder entrenar los modelos. A continuación, se van a describir dos *pipelines* de preprocesado que se han implementado, una simple y otra más compleja. La primera tiene como objetivo realizar un preprocesado básico para poder entrenar los modelos, de forma que luego se pueda tomar como *baseline* para comparar con la segunda *pipeline*, la cual sí busca un preprocesado más complejo que realmente ayude a abordar el problema.

Pipeline 0

Puesto que el preprocesado de esta *pipeline* pretende ser el justo para poder entrenar los modelos, solo se basa en dar el formato a las imágenes para que estas puedan ser introducidas en las redes neuronales, de manera que todas tengan el mismo tamaño y tengan sus valores normalizados dentro del mismo rango.

Para ello se realizan dos pasos:

1. **Reescalado:** Para conseguir el mismo tamaño en todas las imágenes, estas se reescalan a un tamaño de 150x150 píxeles. Con este método se pierden las proporciones originales de las imágenes, por lo que esto puede afectar a las estimaciones de los modelos.

2. **Normalización:** Los valores de los píxeles se normalizan en un rango de 0 a 1 de manera local por cada caso. Por lo que para cada caso se toma el píxel máximo como el valor 1 y el resto se reescalan proporcionalmente. Cabe destacar que con esta normalización directa no se ha lidiado con el problema de la alta variabilidad y *outliers* que hay en los valores de los píxeles, por lo que esto puede afectar al comportamiento de las redes.

***Pipeline* 1**

Para los pasos aplicados en esta *pipeline* se ha buscado solucionar algunos de los problemas vistos en el apartado de análisis de los datos. Además, se facilita la tarea a las redes extrayendo la región de interés (ROI) de las imágenes.

Los pasos seguidos en este preprocesado son:

1. **Reescalado 1mm^2/píxel**: Dentro de los archivos DICOM, uno de los metadatos que podemos encontrar es el «PixelSpacing», con el que podemos saber el volumen real que representan los píxeles. A partir de este, podemos ajustar los parámetros del reescalado para que reescale la imágen de manera que los píxeles representen volúmenes de 1mm^2. Con esto conseguimos que la representación a lo largo del dataset sea más consistente, haciendo que el tamaño de los ventrículos en la imagen sí sea relevante y representativo.
2. **Normalización por ecualización de histograma adaptativo**: Como se ha mostrado en el apartado de análisis, existen imágenes con valores de píxeles muy elevados que pueden suponen un desequilibrio muy grande en la distribución de valores sobre las imágenes. Para lidiar con este problema, las imágenes se han normalizado utilizando la técnica de «ecualización de histograma adaptativa»[2], que realiza una normalización por histograma de manera local dentro de la imagen, utilizando para la transformación de cada píxel su región vecinal. Esta aproximación adaptativa aporta mejores resultados en casos donde la intensidad de los píxeles puede variar mucho de unas zonas a otras de la imagen, como en este caso. Con esta técnica conseguimos mejorar la distribución de valores y el contraste de las imágenes.
3. **Extracción ROI**: En las imágenes originales hay una gran área que no pertenece al corazón donde aparecen otros órganos y además el fondo oscuro de la imagen. Para facilitar la tarea eliminando estas regiones ruidosas y que las redes solo se concentren en el corazón, se ha implementado la extracción de la región de interés. Este método se basa en encontrar el centro de la ROI y extraer un parche de 150x150 píxeles centrado en el. De esta manera obtenemos también las imágenes en el tamaño deseado sin tener que deformarlas, perdiendo así sus proporciones originales que pueden ser de interés.

   Para la extracción de la ROI se realizan los siguientes pasos:

   a) **Transformada rápida de Fourier:** Cada corte del dataset compone un vídeo del ciclo cardíaco visto desde esa posición. El movimiento que se produce en el vídeo por el latido del corazón puede ser detectado utilizando la transformada rápida de Fourier multidimensional. Gracias a esta, podemos obtener de la matriz tridimensional[3] que representa un corte, una imagen en dos dimensiones con un mapa de intensidades que nos dice donde hay movimiento en la imagen. Esta función se aplica por cada corte del paciente, obteniendo así tantas imágenes como cortes posee el caso.

   b) **Segmentación:** Para poder trabajar mejor con los mapas de intensidades generados por la FFT, les aplicamos *Otsu thresholding* para crear máscaras de ceros y unos. Este

[2]Más información en `https://en.wikipedia.org/wiki/Adaptive_histogram_equalization#CLAHE`.
[3]Las dimensiones que componen un corte son (*frames*, Altura, Anchura).

método encuentra un valor de *threshold* para separar los píxeles de la imagen en dos clases de forma que se maximice la varianza entre ellas. Cogiendo solo los píxeles con valores superiores al *threshold* obtenemos una máscara que marca las zonas con movimiento en la imagen.

c) ***Binary erosion*:** De las máscaras generadas por cada corte hay que eliminar algunas partes que se han detectado como de interés pero que no pertenecen al corazón, ya que estas se generan por otros movimientos indeseados. Estos movimientos suelen ser sutiles y generan pequeños artefactos en las máscaras, estos se pueden eliminar utilizando la técnica «binary erosion»[4] con la que podemos pasar una figura a modo de ventana por la imagen, y eliminar los píxeles donde la figura no encaja con el resto de píxeles de esa posición; en nuestro caso hemos utilizado una máscara circular de radio 2 píxeles.

d) **Detección de la ROI:** De las máscaras generadas por cada corte ya preparadas, se obtiene la media de las coordenadas de los píxeles de cada máscara, obteniendo así el centro de cada una sobre la imagen. De los centros obtenidos de cada corte se realiza otra vez la media, obteniendo así el centro de la región de interés.

e) **Extracción del parche:** Sobre las coordenadas de la región de interés obtenidas se extrae un parche de 150x150 píxeles para todos los *frames* de todos los cortes del paciente, obteniendo así las imágenes finales del preprocesado.

En la Figura 3.7 se muestra un ejemplo de cada paso de la *pipeline* aplicado sobre el mismo *frame*. Como se puede ver, consigue detectar con buena precisión donde se localiza el corazón y además le otorga un buen contraste a la imagen.

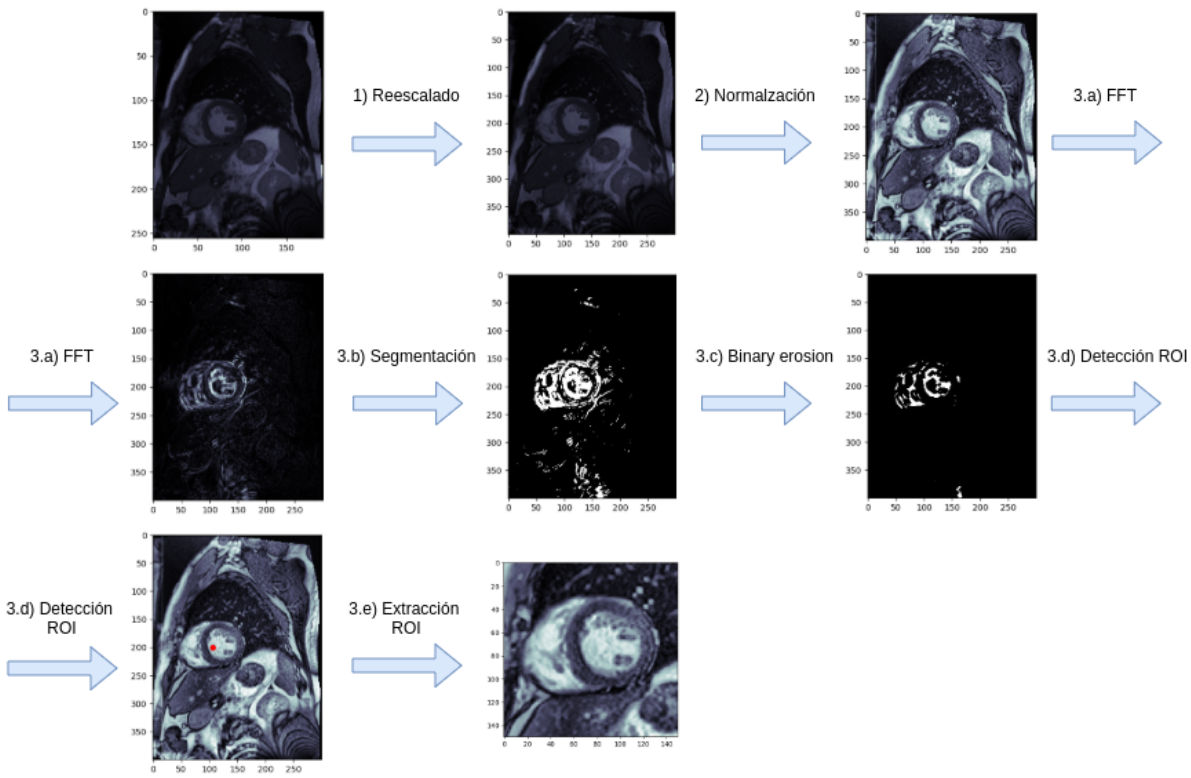


FIGURA 3.7: Pasos del preprocesado aplicado por la *pipeline* 1.

[4]Más información en `https://en.wikipedia.org/wiki/Erosion_%28morphology%29`.

Debido al alto coste computacional de este preprocesado, este se ejecuta antes de entrenar los modelos y las matrices resultantes con los datos se guardan en tensores de la librería *Pytorch*, para posteriormente poder acceder a ellos directamente a la hora del entrenamiento de los modelos.

Como se ha mostrado en la sección del análisis de los datos, existen casos donde hay cortes con menos de 30 *frames*. Para estos casos hemos rellenado los tensores con *frames* con todos los píxeles a cero a modo de *padding* hasta tener 30 canales.

3.2.3. *Data augmentation*

Un práctica muy importante de cara a mejorar los resultados mejorando la generalización de los modelos y evitar el *overfitting*, es la utilización de técnicas de *data augmentation* (DA). Además, en el caso del dataset utilizado no hay disponibles muchas muestras, por lo que es todavía más importante sacar más partido de estas y evitar que los modelos consigan memorizarlas. Debido a que cada muestra de entrenamiento corresponde a un corte, los tensores no tienen un número de canales habitual para una imagen, como puede ser 1 (escala de grises) o 3 (RGB), sino que tenemos 30 canales, uno por cada *frame*. Debido a esto, hemos implementado nuestras funciones de *data augmentation* para poder trabajar con este número de canales.

Las transformaciones que hemos implementado son:

- **Rotación:** Dado un rango valores, toma aleatoriamente un número de dicho rango y rota todos los *frames* del corte en ese número de grados.
- **Translación:** Dado un rango de valores entre -1 y 1, toma dos valores aleatorios de ese rango para trasladar la imagen, en cada uno de esos valores, en cada dimensión. El valor representa la proporción respecto a cada dimensión en la que se realiza el desplazamiento y el signo determina su dirección.
- ***Shift* de canales:** Dado un rango de valores entre -1 y 1, toma un número aleatorio de ese rango y desplaza los *frames* en el tiempo de manera circular, tantas veces como indique el valor elegido. El valor indica la proporción respecto al numero total de canales en que se tiene que realizar el desplazamiento, si el número es negativo los canales se desplazan en la dirección contraria.

A partir de estas transformaciones se han implementado tres *pipelines* de *data augmentation*:

- **DA 1:** Aplica rotación con un rango desde -15 a 15 grados y translación de hasta el 10 % en cada dimensión.
- **DA 2:** Esta *pipeline* es igual que la 1 pero además añade *Shift* de canales del 40 % de los *frames* en cada dirección.
- **DA 3:** Esta es una versión más fuerte de la 2. La rotación es desde -30 a 30 grados, la translación de hasta el 20 % y el *shift* de canales del 60 % en cada dirección.

3.3. Modelos utilizados

Para el diseño de los modelos se ha optado por utilizar redes neuronales convolucionales, debido a que son el tipo de redes más potentes para trabajar con imágenes gracias a su capacidad de captar características y patrones de las imágenes mediante sus filtros. A continuación, se van a describir los modelos utilizados, algunos emplean topologías propuestas por nosotros y otros utilizan topologías ya conocidas que se han aplicado en tareas de clasificación de imágenes.

Los modelos descritos a continuación se han utilizado para los tres tipos de vistas gracias a la compatibilidad en el tamaño de estas, ya que todas pasan por el mismo preprocesado.

La topología del modelo propuesto está compuesta por 6 bloques convolucionales de: convolución, *batch normalization* y activación ReLU. Cada dos bloques se utiliza *MaxPooling* de 2x2 para reducir la dimensionalidad y *Dropout* con la probabilidad $p = 0{,}4$ de apagar las neuronas para añadir regularización y evitar el *overfitting*, debido al reducido número de muestras disponibles para el entrenamiento. Después de los 6 bloques convolucionales se utiliza otra vez *Dropout* y un *GlobalMaxPooling* para reducir el tamaño de los canales a un solo píxel y así obtener un vector de una dimensión con las características extraídas de la imagen, este entra posteriormente en la parte densamente conectada. Esta se compone de dos capas densas, una de 1024 neuronas y otra de 1 neurona para estimar valor del volumen. Después de la primera capa densa, se utiliza una activación ReLU y *batch normalization*; la función de activación de la capa de salida es lineal para sí poder estimar directamente el valor del volumen, el cual se encuentra en el rango de 0 a 599.

De este modelo se han implementado 5 variantes para probar distintos tamaños de filtros en la capa de entrada, con el objetivo de ver que tamaño puede capturar mejor las características de la imagen que son útiles para la estimación del volumen. Todos los modelos utilizan 64 filtros en las dos primeras capas convolucionales, luego 128 en las dos siguientes, 256 en las dos siguientes y en caso de tener más capas utilizan 512. Todas las capas tienen filtros de 3x3, un *stride* de 1 y *padding* de 0. Estos hiperparámetros varían en la primera capa según la variante del modelo, y por lo general se utiliza un *stride* de 2 sin *padding* en la primera capa convolucional para reducir la dimensionalidad justo a la entrada de la red, disminuyendo así el coste computacional.

A continuación se enumeran las modificaciones que cada variante del modelo presenta respecto a la estructura comentada:

- **Modelo 0:** La primera capa utiliza filtros de 5x5.
- **Modelo 1:** La primera capa utiliza filtros de 3x3.
- **Modelo 2:** La primera capa utiliza filtros de 7x7.
- **Modelo 3:** La primera capa es como una convencional con filtros de 3x3 y con *stride* de 1; el padding sigue siendo 0. Con este modelo se pretende comparar con el resto de variantes, para ver si la reducción que realizan los otros modelos en la primera capa es demasiado agresiva. Debido a que no se produce esta reducción al inicio, a modo de compensación del tamaño del tensor antes de entrar en la parte densamente conectada, este modelo tiene dos bloques convolucionales más que el resto, por lo que utiliza una capa *MaxPooling* adicional antes de estos dos bloques para reducir la dimensionalidad.
- **Modelo 4:** Este modelo es igual que el modelo 3 pero en lugar de añadir dos bloques convolucionales solo añade uno de ellos, reduciendo el número de parámetros y manteniendo el modelo lo más pequeño posible para evitar el *overfitting*.

En la Figura 3.8 se muestra la topología de la variante número 0 del modelo propuesto.

Además de estos modelos diseñados por nosotros, hemos empleado algunas topologías conocidas dentro del campo de visión por computador, estas son: VGG-19, WideResNet-50 y la DenseNet-121. Los modelos y sus pesos preentrenados sobre ImageNet han sido adquiridos a través de la librería «torchvision» de *Pytorch*.

De los modelos descritos a continuación solo se ha utilizado la parte convolucional. Para la parte densamente conectada de todos ellos se han utilizado las siguientes capas: una capa densa

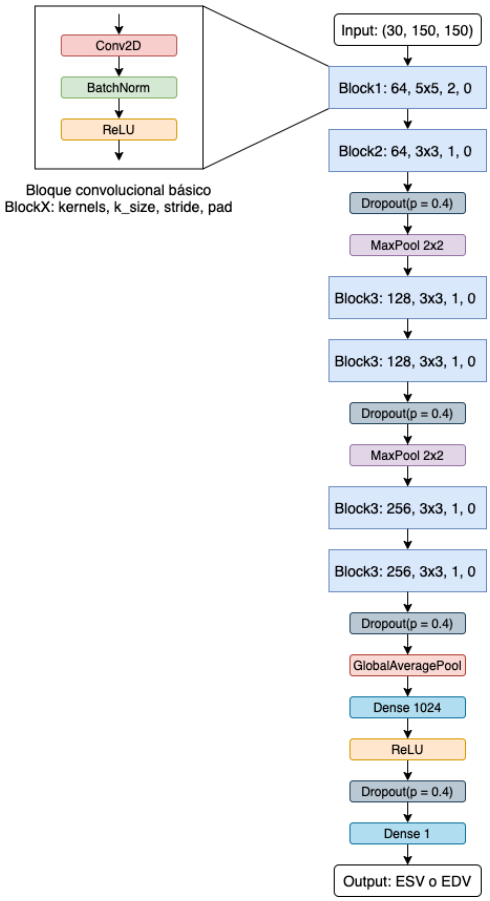


FIGURA 3.8: Topología del modelo 0 propuesto.

de 512 neuronas con función de activación ReLU; una capa *Dropout* con $p = 0{,}4$; y la capa densa de salida con una neurona y activación lineal.

La VGG-19 (Simonyan y Zisserman, 2014) es uno de los primeros modelos propuestos de redes convolucionales profundas, el cual consiguió alcanzar los mejores resultados para estado del arte en clasificación de imágenes sobre el conjunto de datos de la competición de ImageNet de 2014. La topología VGG presenta algunas variantes con diferentes profundidades, en nuestro caso hemos utilizado la más profunda de ellas, la VGG-19. Esta es la variante que mejores resultados aporta y la que mayor cantidad de parámetros tiene (144 millones de parámetros). Este modelo emplea bloques de capas convolucionales donde el número de capas por bloque y filtros de las capas va aumentando a cada bloque, utilizando en todos los casos filtros de 3x3 para las convoluciones. Entre los bloques de las capas convolucionales emplean el *MaxPooling* para reducir la dimensionalidad. La idea detrás de emplear varias capas convolucionales seguidas sin *Pooling* entre ellas, es que, por ejemplo, a partir de un par de capas con filtros de 3x3 pueden conseguir un campo receptivo como el de un filtro de 5x5, en caso de tres capas, uno de 7x7, etc. Optar por esta arquitectura en lugar de usar los filtros más grandes ayuda a reducir los parámetros de la red, introduciendo así algo de regularización y añadiendo más funciones no lineales de activación entre las capas lo que aporta una mayor capacidad de discriminación al modelo.

De la VGG-19 solo hemos tomado la parte convolucional, las capas densas las hemos desechado para introducir las nuestras, debido a que no vamos a realizar clasificación sino regresión, por lo que además la última capa va a tener solo una neurona y no una por cada clase del dataset de ImageNet. Hemos probado a entrenar dos variantes del modelo, una preentrenada y otra sin

preentrenar. La idea es comparar si los pesos preentrenados siguen sirviendo aunque la tarea no sea de clasificación. Por lo que se sabe del comportamiento de las redes convolucionales, las primeras capas captan patrones más simples como lineas, figuras simples o texturas; y a medida que se profundiza en la red ya se van construyendo patrones mucho más complejos, como puede ser la cara de una persona. De estas capas, las primeras nos podrían servir para nuestra tarea, por lo que hemos probado a utilizarlas.

Antes de poder utilizar la topología de la VGG-19 hemos tenido que ajustar la primera capa convolucional del modelo ya que está diseñada para un input con imágenes RGB, por lo que esta primera capa la hemos sustituido por una nueva que tome nuestros cortes que son de 30 canales. Para el caso de la red sin preentrenar esto no presenta un problema, pero para el caso de utilizar los pesos preentrenados sobre ImageNet sí, porque estamos desechando la primera capa preentrenada, la cual puede ser muy importante ya que las siguientes están ajustadas a partir de lo que esta primera capa devuelve como salida. Para abordar este problema, hemos sustituido esta primera capa como hemos descrito antes y además hemos añadido *batch normalization* parametrizado, dando más flexibilidad a la red para ajustar mejor las activaciones de salida y ayudar al modelo a ajustar mejor la primera capa sin destruir los pesos preentrenados. Para el entrenamiento de la versión preentrenada, se congelan los pesos entrenados sobre ImageNet y se entrena el modelo hasta que se estabiliza, posteriormente se descongelan los pesos para realizar un *fine tuning* general.

Otra topología empleada ha sido la *Wide Residual Network* (Zagoruyko y Komodakis, 2016), esta es una versión de la *Residual Net* (He y col., 2015). La WideResNet pretende mejorar algunos aspectos como el coste computacional mediante la reducción de la profundidad de la red a cambio de ensancharla, añadiendo más filtros en las capas. En Zagoruyko y Komodakis, 2016 también demuestran que para mejorar el rendimiento no solo es más efectivo ensanchar las capas que añadir más, sino que también se obtienen mejores resultados.

La característica principal de los modelos residuales es la utilización de conexiones que saltan algunas capas convolucionales, dando más caminos al error a la hora de retropropagarlo por la red. Esto permite entrenar redes profundas con mayor facilidad ya que dificulta la aparición del desvanecimiento del gradiente, por el cual el error va tendiendo[5] a 0 a medida que se retropropaga por la red, haciendo que las primeras capas no lleguen a aprender. En la Figura 3.9 se muestra el esquema de un bloque residual, donde después pasar un tensor por un conjunto de capas, se vuelve a sumar dicho tensor al resultado para dar otro camino al error que no pase por las capas intermedias.

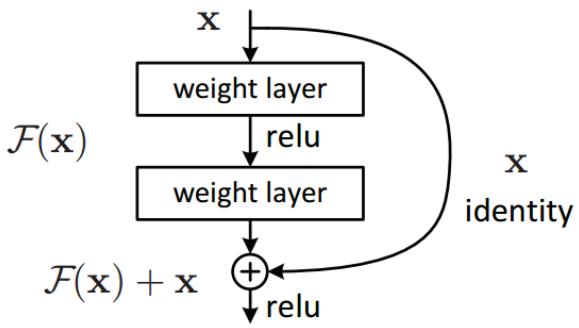


FIGURA 3.9: Esquema de la arquitectura de un bloque residual. Sacado de He y col., 2015.

[5]Esta reducción progresiva se produce al ir multiplicando el error por números pequeños repetidamente, lo que hace que al final este sea prácticamente 0.

De las versiones disponibles de la WideResNet hemos elegido la de 50 capas de profundidad, de ella hemos tomado los 13 primeros bloques[6] convolucionales para reducir el número de parámetros de la red, debido a que para la cantidad de datos con los que contamos no necesitamos muchos parámetros, y de esta manera también se reduce el coste computacional. Al igual que con la VGG-19, hemos tenido que reemplazar la capa de entrada para hacer el modelo compatible con nuestros datos, perdiendo así estos pesos de la versión preentrenada. También hemos añadido *batch normalization* parametrizado tras esta capa para facilitar a la red el ajuste sobre los pesos preentrenados, y para realizar el entrenamiento hemos congelado primero los presos preentrenados hasta un punto de estabilización de la red, para posteriormente descongelarlos y realizar un *fine tuning*.

El último modelo utilizado es DenseNet (Huang, Liu y Weinberger, 2016), este ha dado los mejores resultados en la publicación Luo y col., 2020 donde utilizan también el dataset de *Kaggle*. La característica principal de esta topología es la utilización de bloques convolucionales densamente conectados, en los que la salida de cada capa está conectada a la entrada de todas las capas siguientes mediante concatenación. En la Figura 3.10 se puede ver un ejemplo de un bloque denso de la DenseNet.

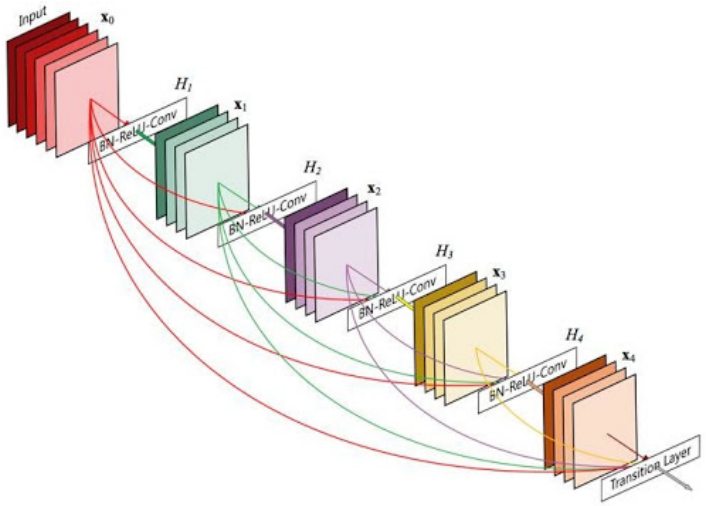


FIGURA 3.10: Estructura de un bloque densamente conectado de la DenseNet.
Extraída de Huang, Liu y Weinberger, 2016.

Los bloques densamente conectados utilizan la concatenación para agregar los tensores, por lo que todos ellos han de tener la misma forma en los mapas de características, esto hace que dentro de un bloque densamente conectado no se pueda reducir el tamaño de los mapas. Para ello la DenseNet utiliza varios bloques densos los cuales están conectados por capas intermedias para realizar la reducción de tamaño, estas capas son una convolución con filtros de 1x1 para realizar la combinación lineal a través de todos los canales del mapa de características de salida y un capa *AveragePooling* con una ventana de 2x2 para realizar la reducción de dimensionalidad. Para la ultima capa del bloque convolucional utilizan *GlobalAveragePooling*, reduciendo el mapa de características a un vector 1D con la codificación para el clasificador MLP. En la Figura 3.11 se puede ver la estructura completa del modelo DenseNet.

De las variantes disponibles del modelo hemos utilizado la DenseNet-121. Al igual que con los otros modelos, solo hemos tomado la parte convolucional y hemos sustituido la capa de entrada por una compatible con nuestras muestras de 30 canales. La capa introducida es una convolucinal con 64 filtros de 5x5 con *stride* de 2 para reducir la dimensionalidad antes de entrar

[6]Consultar el Apéndice B para más detalles sobre el modelo basado en WideResNet-50.

al bloque densamente conectado, puesto que este requiere de una gran cantidad de memoria debido a la concatenación constante de los tensores intermedios. Al igual que en los modelos descritos anteriormente, hemos utilizado *batch normalization* parametrizado después de esta capa para facilitar a la red el ajuste de los pesos mientras estos están congelados. Debido al gran coste computacional de este modelo, también se ha añadido una capa *MaxPolling* después del *batch normalization*.

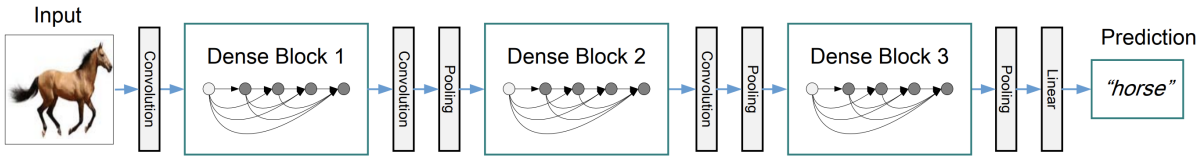


FIGURA 3.11: Estructura del modelo DenseNet. Extraída de Huang, Liu y Weinberger, 2016.

3.4. Entrenamiento

Para la regresión de los volúmenes ventriculares ESV y EDV se han utilizado activaciones lineales, para así poder estimar directamente el valor del volumen, el cual se puede encontrar en el rango de 0 a 599 mililitros.

Como funciones de perdida se han probado *Mean Squared Error* (MSE) y *Mean Absolute Error* (MAE). Normalmente, en una tarea de regresión se utiliza MSE por su mayor sensibilidad ante los *outliers*, ya que en caso de querer contemplar estos casos esta función proporciona mejores resultados. Pero en datasets muy ruidosos puede ser contraproducente, y en el caso de este dataset hay bastantes casos especiales y al mismo tiempo no se cuenta con muchos datos, por lo que también se ha probado la función MAE por ver si puede aportar mejores resultados.

Los optimizadores probados han sido *Stochastic Gradient Descent* (SGD) con *momentum* de 0.9 y Adam por ser los que mejores resultados suelen aportar. Para el *learning rate* se ha utilizado un valor inicial de 0.001 para ambos optimizadores. Esto es así debido a que hemos querido empezar el entrenamiento con un *learning rate* alto para posteriormente ir reduciéndolo a medida que los modelos se estabilicen. Para ello hemos utilizado un *learning rate scheduler*, que en caso de que el modelo no mejore la función de perdida en el conjunto de validación durante las últimas 10 *epochs*, este reduce el *learning rate* a la mitad.

Para los modelos que no estaban utilizando pesos preentrenados el número de *epochs* de entrenamiento han sido 100. Para el caso de los modelos preentrenados, estos han entrenado durante 150 *epochs*, donde durante las primeras 45 los pesos preentrenados han estado congelados. Con este número de *epochs* ha sido suficiente para que todos los modelos lleguen a estabilizarse sin mostrar signos de que fueran a mejorar con más *epochs* de entrenamiento.

3.5. Inferencia

Para realizar la inferencia sobre el conjunto de test, la competición de *kaggle* pide que no solo se aporten los volúmenes ESV y EDV de las muestras como etiqueta, sino que se calcule una distribución de probabilidad acumulada por cada una en la que se tenga en cuenta el margen de error del modelo. El objetivo de la competición con este método de evaluación, es que cada método haga su aproximación más apropiada según su tipo de modelo implementado y su desempeño, para así aportar una estimación más informativa que simplemente el valor estimado de cada volumen.

En la Figura 3.12 se puede ver una descripción gráfica del tipo de distribución que hay que crear para la entrega de la inferencia sobre el conjunto de test para la competición. En cada punto del eje X se representa la probabilidad de que el volumen del ventrículo sea menor que X. Se ha de crear una distribución de este tipo por cada uno de los volúmenes a estimar, ESV y EDV.

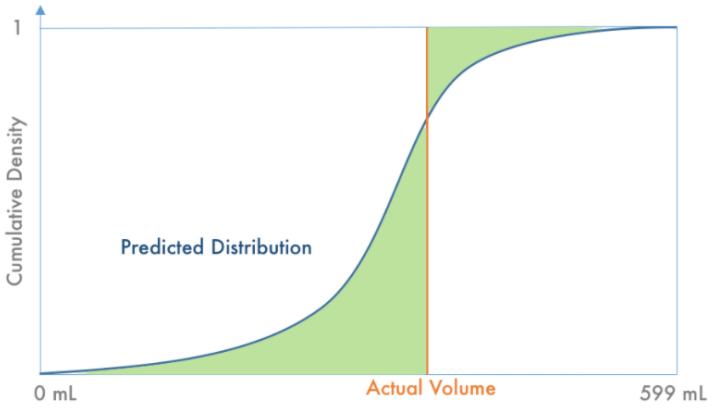


FIGURA 3.12: Descripción gráfica de la distribución de probabilida acumulada para la competición de *Kaggle*.

Para calcular el valor de la métrica a partir de estas distribuciones de probabilidad, el juez automático de la competición calcula la *Continuous Ranked Probability Score* (CRPS) a partir de las distribuciones estimadas y los valores reales de los volúmenes siguiendo la Fórmula 3.1.

$$CRPS = \frac{1}{600N} \sum_{m=1}^{N} \sum_{n=0}^{599} (P(y \leq n) - H(n - V_m))^2 \tag{3.1}$$

donde N es el numero total de distribuciones de probabilidad (dos por cada caso), P es la distribución estimada, V es el volumen real y H es una función *step* ($H(x) = 1$ para $x \geq 0$ y por lo contrario 0).

Para crear esta distribución a la hora de realizar la inferencia hemos probado dos métodos:

- **Básico:** Este produce una *step function* de manera que los volúmenes menores que el valor estimado por los modelos tienen una probabilidad de 0 y los que son iguales o mayores tienen una probabilidad de 1. Este es un método muy básico que sirve como *baseline* para comparar, ya que tras realizar las primeras pruebas se ha observado como este método resulta muy perjudicado en caso de no acertar el volumen estimado, a diferencia del siguiente método.
- **CDF:** Con una función de la librería *Scipy* para crear distribuciones de probabilidad acumuladas, creamos una a la que le asignamos como media el volumen estimado por el modelo, y para regular la pendiente le damos como desviación estándar el error en mililitros que el modelo ha tenido en el conjunto de validación durante el entrenamiento. Esta función sí que nos da un resultado más parecido a lo que se está buscando en la competición y también nos proporciona mejores valores de CRPS.

Además de la inferencia a partir de un solo modelo, también se ha implementado un *ensemble* dinámico para poder aprovechar todos los datos de los casos, para cualquier conjunto de vistas disponibles en estos. Por lo que hemos entrenado modelos para cada tipo de vista (SAX, 2CH y 4CH) y hemos creado un *ensemble* con un modelo de cada tipo. Como hay casos que les faltan algunas de las vistas 2CH o 4CH, el *ensemble* solo utiliza los modelos que tengan su vista

correspondiente disponible, y posteriormente, calcula la media de los valores que ha estimado con cada modelo para obtener el valor de volumen final. En este caso, para obtener el valor de la desviación estándar para la CDF, se calcula la media del error que han tenido los modelos utilizados por el *ensemble* en la partición de validación.

4 Resultados y análisis

4.1. Resultados

En este apartado se van a mostrar los resultados de los experimentos realizados basados en la metodología y técnicas descritas en el apartado de experimentación.

La métrica empleada para medir el rendimiento de nuestros modelos es el *Mean Absolute Error* (MAE), el cual calcula la media de los valores absolutos de los errores cometidos al estimar el volumen en mililitros (Fórmula 4.1). Los resultados mostrados sobre las particiones de entrenamiento (tr) y validación (dev) se calculan mediante el MAE, a partir de las estimaciones individuales de cada corte del subconjunto correspondiente. En el caso de los resultados mostrados sobre la partición de test, estos se obtienen del juez automático de la competición de *Kaggle*, el cual calcula la métrica CRPS (apartado 3.5) a partir de las distribuciones de probabilidad estimadas de sístole y diástole para cada paciente, donde se han utilizado todos los cortes de cada paciente para el calculo sus volúmenes ESV y EDV.

$$MAE = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n} \tag{4.1}$$

En primer lugar podemos observar en la Tabla 4.1 la comparación entre los resultados del entrenamiento para las variantes de la topología propuesta. Se puede ver como todos los modelos tienden a hacer algo de *overfitting* a pesar de tratarse de topologías bastante reducidas, sobretodo en comparación a los otros empleados como WideResNet-50 y DenseNet-121. Teniendo en cuenta el balance entre el MAE de sístole y diástole las mejores variantes son la 0 y la 1. De estas hemos tomado el modelo 1 como referencia para los siguientes experimentos, debido a que es algo más eficiente computacionalmente que el modelo 0, por utilizar filtros más pequeños en la primera capa, y por tener una media de error total entre sístole y diástole un poco menor.

CUADRO 4.1: Comparación de las diferentes variantes de la topología de red neuronal propuesta. Estos resultados se han obtenido con la *pipeline* de preprocesado 1 y con el *data augmentation* 1.

	Sístole		**Diástole**	
Modelo	**MAE tr (ml)**	**MAE dev (ml)**	**MAE tr (ml)**	**MAE dev (ml)**
modelo 0	11.21	18.033	15.61	**26.349**
modelo 1	10.84	**17.430**	14.69	26.512
modelo 2	11.96	19.343	18.66	26.738
modelo 3	9.91	17.820	13.04	26.781
modelo 4	25.81	27.925	34.06	36.266

Debido al *overfitting* que realizan todos los modelos, hemos probado distintos niveles de *data augmentation* para ver cual nos puede ayudar más a lidiar con este problema. En la Tabla 4.2 se puede ver la comparación de los distintos conjuntos de transformaciones y los resultados que aportan. Aunque la métrica sobre validación no mejore constantemente a medida que se

aumenta el nivel de DA, sí que se puede ver como el *overfitting* se va reduciendo cada vez más, donde podemos llegar a ver un gran salto del DA 2 al DA 3 en el caso de diástole donde se mejora mucho la generalización. Debido al tipo de caso de uso que estamos tratando y la variabilidad que se puede dar en este tipo de imágenes, es muy importante que el modelo muestre buena generalización además de un buen resultado, porque nos indica que ante un nuevo conjunto de datos va a ser más probable que se comporte de manera parecida.

CUADRO 4.2: Comparación de los diferentes conjuntos de *data augmentation* implementados. Los resultados se han obtenido utilizando el modelo propuesto 1 y la *pipeline* de preprocesado 1.

	Sístole		**Diástole**	
DA	**MAE tr (ml)**	**MAE dev (ml)**	**MAE tr (ml)**	**MAE dev (ml)**
sin DA	10.46	18.122	12.01	27.267
DA 1	10.84	17.430	14.69	26.512
DA 2	13.72	18.942	19.67	26.705
DA 3	15.54	19.237	25.61	28.125

En cuanto al preprocesado empleado, al final de la *pipeline* de preprocesado 1, en la que se extrae la región de interés que contiene el corazón, se extrae un parche sobre la posición de la ROI para obtener la imagen final. Hemos probado varios tamaños para la extracción de este parche y así ver cual nos aporta mejores resultados. En la Tabla 4.3 se pueden ver los resultados obtenidos con distintos tamaños. Estos obtienen resultados parecidos, aunque en promedio el tamaño de 150x150 es el que mejores resultados aporta. Además, inspeccionando manualmente algunas imágenes se han visto algunos casos donde el parche de 150x150 ya toca los bordes del corazón, por lo que no se debería reducir más el tamaño ya que empezaríamos a quitar píxeles de la región de interés.

CUADRO 4.3: Comparación de diferentes tamaños de imagen de salida de la pipeline de preprocesado 1. Resultados obtenidos utilizando el modelo propuesto 1 y DA 1.

	Sístole		**Diástole**	
Tamaño	**MAE tr (ml)**	**MAE dev (ml)**	**MAE tr (ml)**	**MAE dev (ml)**
120x120	11.96	18.667	15.41	25.951
135x135	10.93	18.097	14.88	26.166
150x150	10.84	17.430	14.69	26.512

Tomando el tamaño de 150x150 píxeles como referencia para el preprocesado 1, hemos comparado las dos *pipelines* de preprocesado implementadas para ver como de beneficiosa es la extracción de la ROI y el preprocesado más complejo que la *pipeline* 1 tiene respecto a la 0, la cual es muy básica. En la Tabla 4.4 se puede ver la comparación de las dos *pipelines* para distintos modelos implementados. Para todas las topologías y versiones tanto de sístole como diástole, la *pipeline* de preprocesado 1 aporta mejores resultados, por lo que podemos deducir que la extracción de la región de interés y su preprocesado (escalado y contraste) están ayudando al modelo a encontrar los patrones más útiles para resolver la tarea.

Todos los resultados y comparativas que se van a mostrar a continuación utilizan la *pipeline* de preprocesado 1 con un tamaño de imagen de 150x150 y el DA 3, por ser lo que mejores resultados aporta.

En el caso de las topologías tomadas de la literatura (VGG-19, WideResNet-50 y DenseNet-121) tenemos acceso a pesos preentrandos sobre el dataset de Imagenet para una tarea de clasificación. En nuestro caso estamos abordando un problema de regresión sobre un conjunto

CUADRO 4.4: Comparación de las dos *pipelines* de preprocesado. Los resultados se han obtenido utilizando DA 3.

	preproc. pipeline 0				preproc. pipeline 1			
	Sístole		Diástole		Sístole		Diástole	
Modelo	**MAE tr (ml)**	**MAE dev (ml)**	**MAE tr (ml)**	**MAE dev (ml)**	**MAE tr (ml)**	**MAE dev (ml)**	**MAE tr (ml)**	**MAE dev (ml)**
modelo 1	26.29	28.131	28,65	36,933	15.54	19.237	25.61	28.125
VGG-19	7.76	22.024	40.41	38.946	18.03	20.989	31.11	25.050
WideResNet-50	5.57	18.129	10.72	25.984	7.69	17.275	11.32	22.550
DenseNet-121	4.87	20.429	8.46	28.331	6.02	16.496	9.14	23.115

de imágenes muy distintas al de Imagenet, por lo que hemos experimentado sin y con pesos preentrenados para ver en que medida nos ayudan para una tarea tan distinta.

Por lo que sabemos de las redes, en las primeras capas se reconocen patrones de elementos simples como lineas, texturas o figuras simples; lo que sí nos puede servir para nuestros modelos. Aunque cabe destacar que para poder utilizar estos modelos hemos tenido que cambiar la primera capa por una que acepte nuestras muestras con treinta canales, a diferencia de los tres originales para imágenes RGB que utilizan en Imagenet, haciendo que perdamos los pesos preentrenados de la primera capa. Realizando el entrenamiento como se describe en el apartado de experimentación, mediante el congelado y descongelado de los pesos preentrenados para el *fine tuning*, hemos comparado los resultados con y sin pesos preentrandos para las distintas topologías implementadas. En la Tabla 4.5 se puede ver la comparación. Por lo general, para los pesos preentreandos se obtienen mejores resultados en el conjunto de validación, llegando a bastante mejoría en el caso de la WideResNet-50. El comportamiento en cuanto al *overfitting* varía según el modelo, para la VGG-19 con pesos preentrandos se aprecia menos diferencia entre los conjuntos de entrenamiento y validación, pero en el caso de la WideResNet-50 ocurre al contrario, a pesar de obtener resultados bastante mejores en validación. Y en el caso de la DenseNet-121, esta es a la que menos le afecta tanto en resultados como en *overfitting* el utilizar los pesos preentrandos o no. Siendo tan dispar el comportamiento según el modelo, no se pueden extraer conclusiones claras, aunque por lo general sí que parecen mejorar los resultados de validación utilizando los pesos preentrenados.

CUADRO 4.5: Comparación de los resultados obtenidos con y sin pesos preentrenados sobre Imagenet.

	Pesos preentrenados Imagenet				Pesos no preentenados			
	Sístole		Diástole		Sístole		Diástole	
Modelo	**MAE tr (ml)**	**MAE dev (ml)**	**MAE tr (ml)**	**MAE dev (ml)**	**MAE tr (ml)**	**MAE dev (ml)**	**MAE tr (ml)**	**MAE dev (ml)**
VGG-19	18.03	20.989	31.11	25.050	13.86	19.097	13.86	26.566
WideResNet-50	7.69	17.275	11.32	22.550	22.66	22.250	26.65	30.931
DenseNet-121	6.02	16.496	9.14	23.115	9.05	18.814	10.59	25.029

Para el optimizador empleado durante el entrenamiento hemos probado con los dos más populares, Adam y SGD. En la Tabla 4.6 se puede ver la comparación de los resultados para ambos optimizadores y los diferentes tipos de modelos implementados. Adam nos ha aportado una convergencia mucho más rápida y con mejores resultados en validación en todos los casos, la única ventaja que parece proporcionar SGD es que los modelos no hacen tanto *overfitting*. Cabe destacar que SGD ha resultado muy inestable numéricamente para los modelos de la literatura ya que estos son mucho más grandes que el modelo propuesto, resultando en problemas de *exploding gradients* donde el entrenamiento se vuelve muy inestable resultando en pesos muy grandes que acaban haciendo overflow y convirtiéndose en valores NaN. En el caso de la DenseNet-121 se ha tenido que reducir el *learning rate* de 0.001 a 0.00005 para que no sucediera este problema, además de incrementar el tamaño del *batch* lo máximo posible para tener más estabilidad en las actualizaciones del modelo. Y en el caso de la VGG-19, a pesar de emplear los mismos métodos que con la DenseNet-121 y añadiendo regularización L2 para controlar el tamaño de los pesos, no se ha conseguido evitar que la red se estabilice.

CUADRO 4.6: Comparación de los optimizadores Adam y SGD.

	Adam				SGD			
	Sístole		Diástole		Sístole		Diástole	
Modelo	**MAE tr (ml)**	**MAE dev (ml)**	**MAE tr (ml)**	**MAE dev (ml)**	**MAE tr (ml)**	**MAE dev (ml)**	**MAE tr (ml)**	**MAE dev (ml)**
model 1	15.54	19.237	25.61	28.125	26.61	23.528	35.90	32.226
WideResNet-50	7.69	17.275	11.32	22.550	23.49	22.964	19.82	26.174
DenseNet-121	6.02	16.496	9.14	23.115	21.85	20.663	24.18	27.815

Como función de pérdida también hemos probado dos alternativas, *Mean Squared Error* (MSE) y *Mean Absolute Error* (MAE). La diferencia principal entre ambas funciones es la mayor sensibilidad a los *outliers* de la MSE, lo que es beneficioso cuando se quieren contemplar mejor algunos casos especiales, pero en conjuntos de datos ruidosos puede ser contraproducente y sesgar el modelo a causa de ciertos casos. En la Tabla 4.7 se pueden ver los resultados comparativos para ambas funciones. Para los modelos de tamaño más reducido como el modelo 1 y la VGG-19 la MAE ha tenido mejores resultados, mientras que para los de más capacidad, WideResNet-50 y DenseNet-121, ha ocurrido al contrario. Por lo que ninguna de las dos funciones resulta claramente mejor que la otra, así que las dos opciones se han tenido en cuenta para los siguientes experimentos mostrados.

CUADRO 4.7: Comparación de las funciones de pérdida MSE y MAE.

	MSE loss				MAE loss			
	Sístole		Diástole		Sístole		Diástole	
Modelo	**MAE tr (ml)**	**MAE dev (ml)**	**MAE tr (ml)**	**MAE dev (ml)**	**MAE tr (ml)**	**MAE dev (ml)**	**MAE tr (ml)**	**MAE dev (ml)**
modelo 1	15.54	19.237	25.61	28.125	19.11	17.843	25.39	26.426
VGG-19	18.03	20.989	31.11	25.050	12.12	16.325	15.82	24.897
WideResNet-50	7.69	17.275	11.32	22.550	11.14	17.200	20.66	24.447
DenseNet-121	6.02	16.496	9.14	23.115	13.19	16.778	10.20	23.844

En cuanto a las vistas auxiliares 2CH y 4CH, en la Tabla 4.8 se muestran los resultados para cada vista y los distintos modelos implementados con la función de pérdida MSE, y en la Tabla 4.9 lo mismo pero para la función de pérdida MAE. Se puede ver como el modelo DenseNet-121 presenta claramente los mejores resultados, obteniendo el mejor valor de la métrica para los dos tipos de modelo (Sístole y Diástole) de cada vista y para las dos funciones de pérdida.

Los modelos basados en vistas auxiliares no pueden ser utilizados para todos los casos ya que hay algunos que no cuentan con alguna de estas vistas. Para poder realizar la estimación sobre todo el conjunto de test y obtener el resultado de la métrica de la competición, se ha utilizado como modelo auxiliar para los casos sin alguna de las vistas el WideResNet-50, por ser el que mejores resultados ha obtenido en test. Utilizando esta metodología, los resultados para cada modelo y por cada vista con la métrica de la competición, se pueden ver en la Tabla 4.10.

CUADRO 4.8: Comparación de los modelos para las vistas auxiliares 2CH y 4CH. Modelos entrenados utilizando la función de pérdida MSE.

	2CH				4CH			
	Sístole		Diástole		Sístole		Diástole	
Modelo	**MAE tr (ml)**	**MAE dev (ml)**	**MAE tr (ml)**	**MAE dev (ml)**	**MAE tr (ml)**	**MAE dev (ml)**	**MAE tr (ml)**	**MAE dev (ml)**
modelo 1	17.45	16.479	20.31	22.974	18.16	18.961	19.91	21.579
VGG-19	19.12	17.621	38.13	37.542	21.71	18.558	29.28	24.058
WideResNet-50	15.66	16.325	23.04	22.839	10.88	15.188	14.95	21.248
DenseNet-121	13.98	**15.119**	12.94	**19.831**	10.45	**13.412**	18.58	**18.274**

Tras analizar los resultados previos, se ha realizado la inferencia para cada tipo de modelo con aquellas variantes más prometedoras, es decir, utilizando como preprocesado la *pipeline* 1, el conjunto de transformaciones DA 3 y Adam como optimizador. Basándose en esta configuración, se ha realizado la inferencia sobre el conjunto de test utilizando tanto los modelos con función de pérdida MSE como MAE para cada tipo de vista. En la Tabla 4.10 se muestra el valor de la métrica calculado por el juez automático de la competición de *Kaggle* para cada variante.

CUADRO 4.9: Comparación de los modelos para las vistas auxiliares 2CH y 4CH. Modelos entrenados utilizando la función de pérdida MAE.

	2CH				4CH			
	Sístole		Diástole		Sístole		Diástole	
Modelo	**MAE tr (ml)**	**MAE dev (ml)**	**MAE tr (ml)**	**MAE dev (ml)**	**MAE tr (ml)**	**MAE dev (ml)**	**MAE tr (ml)**	**MAE dev (ml)**
modelo 1	15.58	16.332	20.43	21.854	16.73	14.851	20.65	21.480
VGG-19	8.68	14.841	18.29	21.131	23.63	20.120	22.53	20.093
WideResNet-50	14.77	14.445	20.60	20.471	13.96	13.887	17.69	19.990
DenseNet-121	8.74	**13.718**	17.68	**19.305**	7.39	**13.035**	14.86	**16.803**

De los resultados para la vista SAX, el modelo WideResNet-50 con función de pérdida MSE ha sido el mejor, este ya presentaba el mejor resultado junto a WideResNet-121 durante el entrenamiento (Tabla 4.7). En cuanto a la vista auxiliar 2CH, sí que ha habido un resultado algo inesperado debido a que el mejor modelo ha sido la VGG-19 con la función de pérdida MAE, ya que tanto la WideResNet-50 como la DenseNet-121 tenían unos resultados mejores en el entrenamiento (Tabla 4.9); esto en parte puede ser debido al grado de incertidumbre que existe al comparar la métrica durante el entrenamiento, la cual se calcula a partir del error individual para cada corte y la métrica de inferencia, la cual se calcula a partir de los volúmenes estimados para cada caso utilizando la media de los volúmenes de sus cortes. En el caso de la vista 4CH, sí que se ha obtenido un resultado esperado, ya que la DenseNet-121 con función de pérdida MAE ha sido el mejor modelo, como ya se adelantaba en los resultados del entrenamiento (Tabla 4.9).

CUADRO 4.10: Resultados obtenidos en la competición sobre el conjunto de test para cada vista, tipo de modelo y función de pérdida.

	CRPS Competición					
	SAX		**2CH**		**4CH**	
Modelo	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**
modelo 1	0.02349	0.02466	0.02821	0.02613	0.02787	0.02534
VGG-19	0.02281	0.02079	0.03595	**0.02330**	0.02948	0.02764
WideResNet-50	**0.01933**	0.01978	0.02672	0.02453	0.02328	0.02332
DenseNet-121	0.02060	0.01994	0.05177	0.02454	0.02242	**0.02213**

Una vez entrenados y probados todos los modelos, hemos utilizado los mejores para crear *ensembles* dinámicos con los diferentes tipos de vistas, como se ha descrito en el apartado 3.5 de inferencia. Las combinaciones probadas y sus resultados se pueden ver en la Tabla 4.11. Según los resultados obtenidos individualmente por los modelos, la mejor combinación de ellos para el *ensemble* debería ser WideResNet-50 con MSE para la vista SAX, VGG-19 con MAE para la vista 2CH y DenseNet-121 con MAE para la vista 4CH; pero esta combinación no ha resultado la mejor, ya que tras probar varias combinaciones hemos visto que para la vista 4CH, la DenseNet-121 con MSE funciona un poco mejor al combinarla con los otros modelos. Ya que al realizar la media de las estimaciones de los modelos del *ensemble*, puede ser que algunos modelos combinen mejor con otros.

Tras todas las pruebas realizadas, hemos obtenido como mejor aproximación la utilización de un *ensemble* para las diferentes vistas disponibles, frente a un solo modelo. Obteniendo para el mejor modelo en solitario una valor de CRPS de 0.01933 y para el mejor *ensemble* de 0.01775.

CUADRO 4.11: Comparación de distintas combinaciones de *ensembles* de modelos con la métrica obtenida de la competición sobre el conjunto de test.

Ensemble			
SAX	**2CH**	**4CH**	**CRPS Competición**
WideResNet-50 + MSE	WideResNet-50 + MSE	DenseNet-121 + MSE	0.01853
WideResNet-50 + MSE	WideResNet-50 + MSE	DenseNet-121 + MAE	0.01864
WideResNet-50 + MSE	DenseNet-121 + MAE	DenseNet-121 + MSE	0.01794
WideResNet-50 + MSE	VGG-19 + MAE	DenseNet-121 + MAE	0.01783
WideResNet-50 + MSE	VGG-19 + MAE	DenseNet-121 + MSE	**0.01775**
WideResNet-50 + MAE	VGG-19 + MAE	DenseNet-121 + MSE	0.01778
DenseNet-121 + MAE	VGG-19 + MAE	DenseNet-121 + MSE	0.01791

4.2. Clasificación en la competición

La competición de *Kaggle* tuvo lugar a principios de 2016 y contaba con 200.000$ en premios para repartir entre los tres primeros equipos. En total participaron 192 equipos. En la Tabla 4.12 se muestra un resumen de las puntuaciones de la clasificación y la posición respectiva de la puntuación obtenida por nosotros, la cual nos situaría en la posición 27. Teniendo en cuenta que las mejores aproximaciones han utilizado datos adicionales e incluso datos etiquetados manualmente, además de emplear en algunos casos técnicas no aplicables en un caso real, como la utilización de modelos de regresión a partir de solamente la edad y el sexo para los casos con pocos cortes; consideramos que la posición obtenida es buena, puesto que en nuestro caso hemos empleado una metodología que podría ser aplicable en un caso de uso real, sin discriminar ningún caso por tener pocos datos.

CUADRO 4.12: Comparación del resultado obtenido respecto a la clasificación de la competición de *Kaggle*.

Posición	CRPS
1	0.00948
2	0.01012
3	0.01013
10	0.01261
20	0.01522
27	**0.01775**
30	0.01888
50	0.02353
100	0.03255
150	0.04706
192	0.80727

4.3. Interpretabilidad de los modelos

Debido a la alta popularidad de los modelos de *deep learning* y su opacidad en cuanto a su funcionamiento, el campo de estudio sobre la interpretabilidad de los modelos se ha convertido en una área de mucho interés. Conocer a qué partes[1] de una muestra un modelo ha dado mayor

[1]Región (o regiones) de interés dentro de una imagen en el caso de imagen médica.

peso para calcular la salida proporcionada, puede ayudar tanto a mejorar los modelos, gracias a una mejor comprensión de su funcionamiento, como a la confianza en los mismos, de cara a poder utilizarlos en casos reales donde se necesita de una explicación de cómo un modelo ha obtenido el valor o valores de salida.

Un ejemplo claro de caso de uso donde interesa entender el funcionamiento de los modelos es en el ámbito médico, como es caso de este trabajo con la estimación del volumen ventricular. Ya que este tipo de modelos, por muy bien que funcionen durante la fase de test, no se puede contar con que al utilizarlos en entornos reales no puedan aparecer casos para los cuales el modelo no está preparado y, por tanto, dé una estimación errónea. Si el modelo es una caja negra donde no existe la posibilidad de saber como ha obtenido un valor de salida a partir de una muestra de entrada, entonces, es difícil confiar en modelos para cualquier tipo de tarea.

Para abordar este problema, dentro del campo de *model interpretability*, se han desarrollado múltiples algoritmos para medir la contribución de los datos de entrada respecto a la predicción de la salida. Por ejemplo, en el caso de clasificación de imágenes se puede atribuir a cada píxel de la imagen de entrada su impacto en el resultado de la clasificación final para cada etiqueta del vector de salida. Además de poder medir la contribución de la entrada respecto a la salida, también existen algoritmos para medir la contribución de las neuronas de las capas ocultas respecto a la salida y la contribución de la entrada respecto de las neuronas de las capas ocultas. En este trabajo nos hemos centrado en los algoritmos de contribución de la entrada respecto de la salida para tratar de aportar una explicación de en qué se fija la red para realizar la estimación.

Para la implementación de los algoritmos de interpretabilidad se ha utilizado una librería que funciona sobre *Pytorch* llamada *Captum*. Esta librería provee de implementaciones de los algoritmos de interpretabilidad más relevantes en el estado del arte los cuales se pueden ejecutar sobre modelos preentrenados con *Pytorch*. De los algoritmos disponibles se han utilizado el de *Saliency* e *Integrated Gradients*.

Saliency

El algoritmo *Saliency* (Simonyan, Vedaldi y Zisserman, 2013) es uno de los más sencillos y se basa el calcular el gradiente de la salida de la red respecto de la entrada. Para lograr esto se aproxima la función de la red neuronal mediante series de Taylor en la entrada obteniendo un vector del tamaño de la entrada con los gradientes para cada píxel, al multiplicar estos gradientes mediante el producto escalar con la imagen de entrada obtenemos un mapa que nos dice la importancia que ha tenido cada píxel sobre la predicción del modelo. Los coeficientes obtenidos pueden ser tanto positivos como negativos, indicando así la correlación que tienen respecto a la predicción de si afectan positiva o negativamente a esta.

Hemos aplicado el algoritmo utilizando los modelos preentrenados de sístole y diástole con la topología basada en WideResNet-50, por ser de las que aporta los mejores resultados[2]. En la Figura 4.1 se muestran los mapas de atribuciones de los píxeles de la entrada para un *frame* en concreto, utilizando el modelo de sístole y en la Figura 4.2 se puede ver otro *frame* del mismo sujeto para el modelo de diástole. Estos *frames* pertenecen al mismo corte, en el caso de sístole se ha extraído el *frame* del final de la contracción, y en el caso de diástole el del final de la expansión, ya que ambos son los puntos que deberían tener más relevancia para cada modelo.

En el caso de las atribuciones del modelo de sístole, se puede ver como la red se ha basado en la región interna del ventrículo izquierdo asignándole puntuaciones positivas a los píxeles de dicha región, lo cual nos aporta más confianza en el modelo, ya que puede verse que se está

[2]Estos modelos han sido entrenados con la *pipeline* de preprocesado 1 y utilizando DA 3.

basando en la región de interés; se puede ver por el borde de la imagen mucho ruido, esto nos estaría indicando que esos píxeles también tienen bastante relevancia para el modelo, lo cual no es de interés y podría ser una señal de que la red está dando importancia a zonas ruidosas que no deberían aportar información relevante. Si nos fijamos en las atribuciones negativas, volvemos a ver ruido por el borde de la imagen, y si nos fijamos en el ventrículo podemos ver como el modelo resalta el borde de éste, es decir, las paredes del ventrículo. Este comportamiento es interesante, ya que podría estar apoyándose en la delimitación del ventrículo para las atribuciones negativas, y en el interior de éste para las positivas. En el caso del *frame* de diástole, también se puede apreciar en las atribuciones positivas como se resalta el interior del ventrículo, aunque en este caso se aprecia más intensidad en el ruido de los bordes, esto se puede explicar debido a que el modelo de diástole tiene peores resultados que el de sístole, y no está llegando a captar con tanta precisión los patrones relevantes.

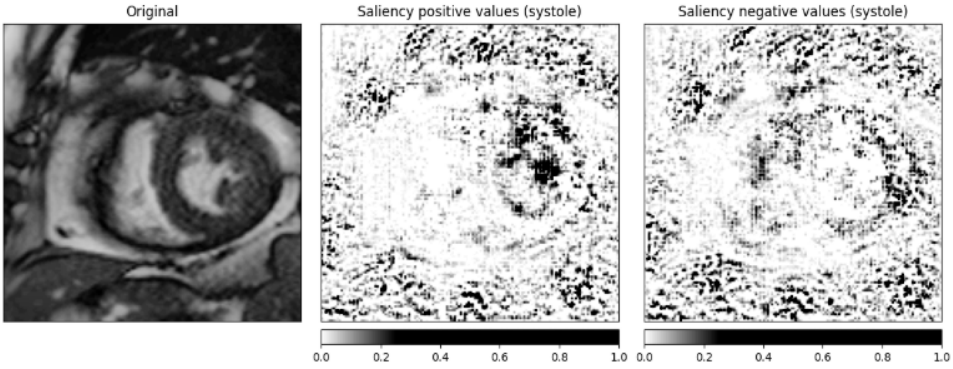


FIGURA 4.1: Visualización de las atribuciones del algoritmo *Saliency* para un *frame* utilizando un modelo de sístole.

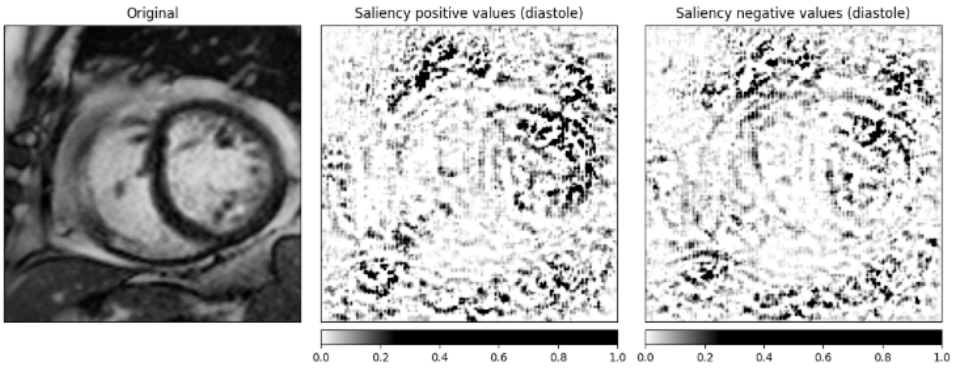


FIGURA 4.2: Visualización de las atribuciones del algoritmo *Saliency* para un *frame* utilizando un modelo de diástole.

Integrated Gradients

El algoritmo *Integrated Gradients* (Sundararajan, Taly y Yan, 2017) es uno de los más recientes y aporta unos resultados más fiables gracias a la aproximación axiomática que toman sus autores. Se basan en una serie de axiomas que un algoritmo de atribución debe cumplir para ser correcto, los dos axiomas fundamentales son *Sensitiviy* e *Implementation Invariance*. Los autores demuestran que otros métodos no los cumplen a diferencia de *Integrated Gradients*. A continuación se describen los dos axiomas comentados:

- ***Sensitivity*:** Un método de atribución satisface este axioma cuando para dos inputs, uno real y otro uno de referencia, que difieren en una característica y tienen diferentes predicciones a la salida de la red, se le da siempre una atribución distinta de 0 a la característica en que difieren. Este efecto no se puede dar en ocasiones debido a que los modelos pueden saturar las funciones de manera que a partir de un valor de entrada no se aprecie cambio en la salida, de manera que en la vecindad de los valores de la entrada no se pueden calcular correctamente los gradientes para extraer las atribuciones. Esto no se tiene en cuenta en el método *Saliency* descrito previamente, por lo que puede dar lugar a regiones ruidosas que no representan lo que la red está captando realmente.
- ***Implementation Invariance*:** Dos redes son funcionalmente equivalentes si sus salidas son iguales para todas las entradas, aunque las implementaciones de las redes sean distintas. Un algoritmo de atribución cumple con el axioma de *Implementation Invariance* si para dos redes funcionalmente equivalentes proporciona las mismas atribuciones. El hecho de que no se cumpla este requisito hace que las atribuciones sean sensibles a detalles de implementación de los modelos. Existen métodos de atribución que no cumplen este requisito como el caso *DeepLift* (Shrikumar, Greenside y Kundaje, 2017) donde utilizan una versión modificada de la retropropagación para emplear gradientes discretos lo que no generaliza correctamente para cualquier implementación.

Para cumplir con estos axiomas el método de *Integrated Gradients* se basa en aproximar la integral de los gradientes respecto a la entrada desde un *baseline* hasta el input dado. Concretamente lo que hacen es generar múltiples entradas que se encuentran desde la *baseline* hasta el input dado. En el caso de imágenes la *baseline* puede ser una imagen con los píxeles a cero y las imágenes intermedias son atenuaciones graduales del input hasta llegar a la *baseline* que está totalmente en negro. Por cada imagen intermedia se calculan sus gradientes y luego se realiza la media de todos ellos para finalmente multiplicar las medias por la imagen de input original y conseguir sus atribuciones. En la Fórmula 4.2 se muestra la descripción formal de este método.

$$ItegratedGrads_i(x) ::= (x_i - x'_i) \times \int_{\alpha=0}^{1} \frac{\partial(x' + \alpha \times (x - x'))}{\partial x_i} d\alpha \tag{4.2}$$

donde i es una característica del input; x es el input; x' es la *baseline* y α es el parámetro que controla el escalado.

Este planteamiento evita la saturación gracias a que para valores bajos de α no aparecen problemas de saturación y se aprecia un gran cambio en el gradiente, mientras que en los valores cercanos a los originales de la imagen de entrada el cambio en los gradientes es mínimo[3]. Así que realizando la media de todos los gradientes se puede obtener un gradiente mucho más representativo de lo que la red está teniendo más en cuenta para realizar la predicción.

En las Figuras 4.3 y 4.4 se muestran las atribuciones del algoritmo *Integrated Gradients* para los mismos *frames* analizados con el método *Saliency*. Se puede ver como el ruido de los bordes en

[3]En el Apéndice C se encuentra una explicación más detallada del comportamiento de los gradientes con este método.

la imagen se ve muy disminuido, y sigue manteniendo las altas atribuciones en el ventrículo izquierdo, lo cual nos indica que la red sí que se está basando principalmente en la región de interés. Cabe destacar que en el caso de *Saliency* el ruido de los bordes se puede deberse principalmente a el no cumplimiento del axioma *Sensitivity* comentado anteriormente, esto puede provocar que aparezcan regiones con mucha atribución que no son correctas. A pesar de la reducción del ruido de los bordes, aún siguen apareciendo zonas que no pertenecen al ventrículo izquierdo ni al corazón, que no deberían estar resaltadas y sí que podrían ser un signo de ruido que la red está tomando como relevante cuando no debería. Esto nos indica que con una mejor extracción de la región de interés, podríamos forzar más a la red a fijarse en las partes relevantes y así poder mejorar su rendimiento y generalización.

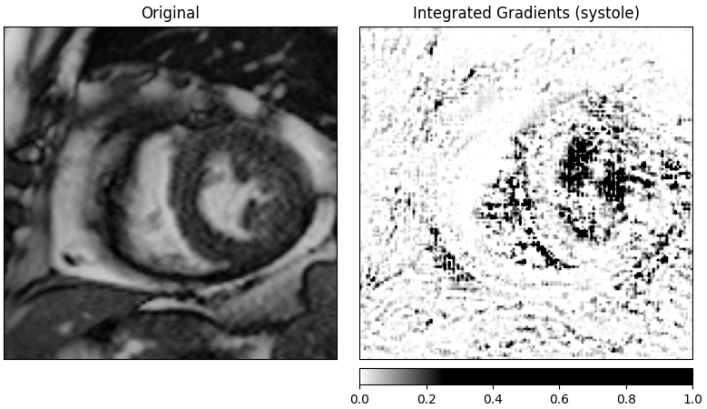


FIGURA 4.3: Visualización de las atribuciones del algoritmo *Integrated Gradients* para un *frame* utilizando un modelo de sístole.

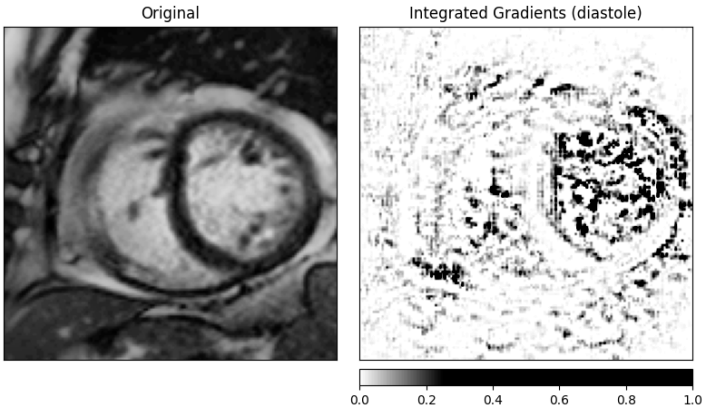


FIGURA 4.4: Visualización de las atribuciones del algoritmo *Integrated Gradients* para un *frame* utilizando un modelo de diástole.

5 Conclusiones y trabajo futuro

5.1. Conclusiones

En este trabajo se ha abordado un problema de gran relevancia como es la estimación del volumen del ventrículo izquierdo del corazón para la ayuda en el diagnostico de enfermedades o disfunciones del corazón. Se ha partido del conjunto de datos de una competición de la plataforma *Kaggle* con el que se ha pretendido tomar una aproximación lo más práctica posible de cara a una implementación real, evitando emplear metodologías sesgadas para la competición solo con el objetivo de obtener mejor puntuación en ella, sin tener en cuenta la generalización y aplicabilidad del modelo en un caso de uso real.

Siguiendo los objetivos propuestos para este trabajo, se ha conseguido realizar un preprocesado complejo que ayude a los modelos a realizar la estimación mediante la corrección de algunos problemas detectados durante el análisis en profundidad del dataset. Además, se ha implementado un método de extracción de la región de interés que no depende de datos adicionales y obtiene buenos resultados.

En cuanto a los modelos implementados, se han probado múltiples topologías, tanto propias como de la literatura, para encontrar aquella que mejor funcione para esta tarea. Se ha observado como utilizando pesos preentrenados para otra tarea de visión por computador bastante distinta se puede seguir realizando *transfer learning* para mejorar los resultados.

Como aproximación final se ha logrado implementar una solución mediante un *ensemble* de los mejores modelos, con el que se consigue obtener un resultado bastante bueno en la clasificación de la competición. El *ensemble* se ha implementado con el propósito de obtener una aproximación capaz de generalizar en cualquier caso de uso independientemente de los datos disponibles.

Por último, se ha analizado uno de los mejores modelos implementados mediante técnicas de *model interpretability*. Estas técnicas son cada vez más populares debido al gran potencial e importancia que tienen de cara a entender los modelos, para poder mejorarlos y que nos proporcionen predicciones más fiables de cara a utilizarlos en escenarios reales. En nuestro caso hemos comparado dos algoritmos de atribución y hemos visto como el modelo es capaz de centrarse en la región de interés del ventrículo izquierdo para sus predicciones. Lo cual genera mayor confianza en el modelo, ya que disponemos de evidencias de que ha tenido en cuenta la región relevante. Por otra parte, también hemos detectado atribuciones no deseadas por el borde de la imagen, lo que nos puede indicar que mejorando el preprocesado para eliminar estas zonas se podrían mejorar las predicciones al eliminar este ruido de los datos que está afectando a la estimación.

5.2. Trabajo futuro

Tras las conclusiones extraídas del trabajo realizado, las posibles líneas de trabajo futuro serían las siguientes:

- **Mejorar el preprocesado:** Tras ver que la *pipeline* de preprocesado ayuda a mejorar los resultados respecto a un preprocesado más simple como la *pipeline* 0[1], se podría seguir trabajando en este aspecto probando más técnicas de preprocesado tanto para la corrección de las intensidades como en la extracción de la región de interés. En la sección de *model interpretability* se ve que la red está tomando en cuenta píxeles no pertenecientes al corazón que no deberían influir en la estimación del volumen. Una posible aproximación para mejorar la extracción de la ROI sería entrenar modelos de segmentación con otros conjuntos de datos (que dispongan de la segmentación de la ROI) para usar un modelo como extractor del corazón y así poder eliminar una mayor cantidad de regiones ruidosas para el modelo.
- ***Data augmentation*:** Como se ha visto en la sección de resultados los modelos tienden a hacer *overfitting*. Para evitar este problema, uno de los puntos claves es mejorar el DA, por lo que probar más combinaciones de las transformaciones a las muestras de entrada podría ayudar a mejorar la generalización. Una transformación nueva a probar es aplicar zoom a las imágenes, modificando el volumen de acuerdo al zoom aplicado. Con esto se enriquece el conjunto de entrenamiento, disponiendo de muestras con volúmenes inexistentes en el conjunto de datos original, ayudando a mejorar la capacidad de generalización de los modelos.
- **Refinar modelos:** De las topologías implementadas aún se podrían realizar más versiones de todas ellas. En el caso de los modelos basados en arquitecturas de la literatura, se podría variar la topología de la parte densa o modificar la primera capa introducida en el modelo. También se podría jugar con la regularización de los pesos para intentar reducir el *overfitting* que se observa en la mayoría de los modelos probados, a pesar de utilizar regularización y *data augmentation*.

[1] Esta *pipeline* simplemente reescala las imágenes para que tengan el mismo tamaño y las normaliza directamente en el rango de valores de 0 a 1.

A Tamaños de imagen en el dataset

El conjunto de datos utilizado es muy heterogéneo en muchos aspectos, uno de ellos es el tamaño de las imágenes el cual también varía mucho. A continuación se muestran los recuentos de los tamaños para todas las imágenes y por cada tipo de vista (SAX, 2CH y 4CH):

Shape	Count
(256, 192)	5596
(192, 256)	1808
(512, 384)	1167
(256, 218)	630
(448, 336)	500
(256, 256)	292
(352, 264)	253
(384, 512)	241
(256, 230)	226
(416, 312)	212
(336, 448)	139
(608, 456)	136
(218, 256)	102
(512, 416)	101
(264, 352)	91
(312, 416)	64
(256, 204)	61
(704, 528)	60
(416, 512)	46
(288, 216)	43
(320, 240)	43
(528, 704)	25
(230, 256)	24
(512, 432)	24
(256, 166)	23
(512, 368)	22
(416, 288)	22
(180, 256)	21
(256, 180)	20
(608, 608)	14
(384, 288)	13
(204, 256)	12
(736, 552)	11
(160, 120)	11
(704, 580)	11
(166, 256)	11
(512, 352)	11
(416, 300)	11
(216, 256)	9
(256, 206)	8
(244, 256)	3
(202, 256)	2
(240, 320)	1
(256, 202)	1

(A) Vista SAX

Shape	Count
(256, 192)	5567
(192, 256)	1627
(256, 218)	612
(512, 384)	317
(256, 256)	284
(256, 230)	205
(352, 264)	87
(256, 204)	44
(312, 416)	21
(416, 312)	21
(608, 608)	17
(218, 256)	15
(512, 432)	14
(204, 256)	13
(704, 704)	3
(512, 512)	3
(448, 452)	3
(448, 444)	3
(352, 352)	3
(416, 416)	3
(288, 288)	3
(256, 244)	3
(320, 320)	2
(512, 464)	1
(448, 360)	1
(736, 736)	1
(320, 240)	1
(256, 216)	1
(512, 576)	1
(512, 480)	1
(512, 560)	1
(448, 448)	1
(256, 166)	1
(244, 256)	1
(512, 496)	1
(512, 448)	1
(384, 384)	1

(B) Vista 2CH

Shape	Count
(256, 192)	4229
(192, 256)	1811
(256, 256)	272
(384, 512)	244
(512, 384)	159
(336, 448)	120
(218, 256)	105
(264, 352)	71
(312, 416)	67
(230, 256)	27
(416, 512)	27
(528, 704)	16
(204, 256)	14
(216, 256)	10
(456, 608)	3
(244, 256)	3
(240, 320)	3
(216, 288)	2
(432, 512)	2
(512, 512)	2
(166, 256)	1
(300, 448)	1
(552, 736)	1
(120, 160)	1
(608, 608)	1
(132, 176)	1
(206, 256)	1
(288, 384)	1

(C) Vista 4CH

CUADRO A.1: Recuento de los tamaños de imagen para cada vista.

B Modificación de la WideResNet50

Para el modelo que hemos implementado basado en la WideResNet hemos utilizado la versión de 50 capas de profundidad ya que era la más pequeña disponible en la librería *Pytorch* y con la disponibilidad de pesos preentrenados. Además de solo tomar la parte convolucional y cambiar la primera capa para que sea compatible con nuestras muestras de 30 canales, no hemos tomado toda la parte convolucional sino que solo hemos utilizado los 13 primeros bloques convolucionales del modelo eliminando así los tres últimos para reducir el tamaño de la red. En la figura B.1 se pueden ver los bloques que componen la ResNet original en sus distintas variantes.

Para la parte densamente conectada hemos utilizado dos capas, una de 512 neuronas y la de salida con solo una para la estimación del volumen. Tras la capa de 512 neuronas utilizamos una función de activación ReLU y *Dropout* con $p = 0{,}4$ para introducir más regularización al modelo para evitar el *overfitting*.

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

FIGURA B.1: Descripción de las variantes del modelo ResNet. Extraída de He y col., 2015.

C *Sensitivity* en *Integrated Gradiens*

Uno de los axiomas fundamentales en los que se basa el método de atribuciones *Integrated Gradients* es el de *Sensitivity*. Para cumplirlo el método ha de comportarse de manera que dada una diferencia en alguna de las características de la entrada respecto a un *baseline*[1] que provoque un cambio en la salida, la atribución que se le ha de dar a dicha característica ha de ser diferente de 0.

Este axioma no se cumple para algunos métodos basados en la retropropagación donde no tienen en cuenta el problema de saturación que las redes presentan. Este produce que para los valores cercanos a las características originales de una entrada las atribuciones que se les otorga son muy pequeñas. Un ejemplo podría ser el siguiente, si tomamos una función f como nuestro modelo donde se utiliza una función no lineal ReLU, $f(x) = 1 - ReLU(1 - x)$, y tomando como *baseline* $x = 0$. El resultado de la función solo cambia de 0 a 1, y a partir de 1 la función satura haciendo que para valores mayores siempre de 1, en estos casos los gradientes asignados harían que la contribución de x fuera 0.

En la figura C.1 se puede ver una imagen de ejemplo[2] de un barco de bomberos y la predicción que un modelo entrenado sobre Imagenet con la arquitectura Inception estima con bastante seguridad (*score* = 0,999961) acertando la existencia de un barco de bomberos en la imagen. A su derecha se pueden ver las atribuciones que aporta un método simple basado en gradientes. Se puede ver como a pesar de que la red ha realizado una predicción correcta y con mucha seguridad, las atribuciones no se muestran claras y además resaltan el fondo de la imagen por debajo del puente sin llegar a apreciar un patrón claro que detecte el barco.

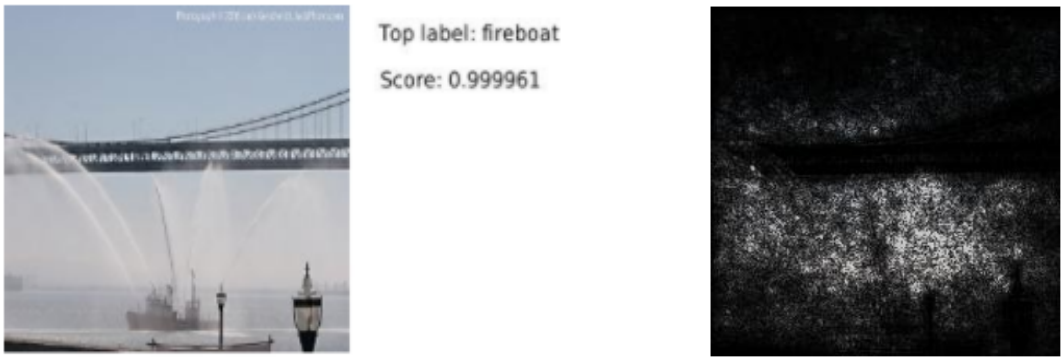


FIGURA C.1: En la figura de la izquierda se muestra la imagen de entrada al modelo más la predicción de la clase ganadora y en la de la derecha se muestran las atribuciones con un método simple de atribución basado en gradientes.

[1] En el caso de imágenes el *baseline* puede ser una imagen en negro.

[2] Este ejemplo ha sido extraído de `http://www.unofficialgoogledatascience.com/2017/03/attributing-deep-networks-prediction-to.html`, este es un post publicado por los autores del método *Integrated Gradients*.

Los autores del método *Integrated Gradients* muestran como desde la imagen original de entrada reduciendo las intensidades de los píxeles mediante un coeficiente α hasta llegar a un *baseline*, si se calculan los gradientes para diferentes valores de α entre 0 y 1, en la figura C.2 se puede apreciar como a partir de $\alpha = 0{,}2$ (en este caso) el valor de los gradientes se anula por lo que las atribuciones ya no son correctas. La intuición de este método es aprovechar ese rango de valores de α donde se aprecian valores significativos de los gradientes.

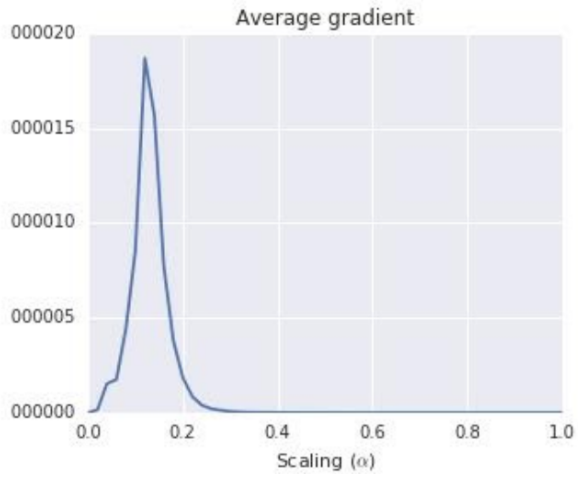


FIGURA C.2: Evolución de los gradientes al variar el coeficiente de escalado α en el método de atribución *Integrated Gradients*.

En la figura C.3 se pueden ver los gradientes para diferentes valores de α. Se puede apreciar como en el rango de $\alpha \in [0,\ 0{,}2]$ donde los valores de los gradientes son más altos (figura C.2), sí que se resalta claramente el barco y los chorros de agua que dispara. Y a medida que el valor de α aumenta sobre 0.2 se va generando cada vez más ruido en el resultado de las atribuciones.

Así pues, el método de *Integrated Gradients* se basa en obtener la media de los gradientes para diferentes valores de α desde 0 hasta 1 y posteriormente multiplicarlos por las características de la imagen para obtener las atribuciones finales. Aplicando este método se obtiene el resultado mostrado en la figura C.4 en el cual ya podemos apreciar una alta intensidad en las atribuciones de los píxeles pertenecientes al barco y los chorros de agua que expulsa.

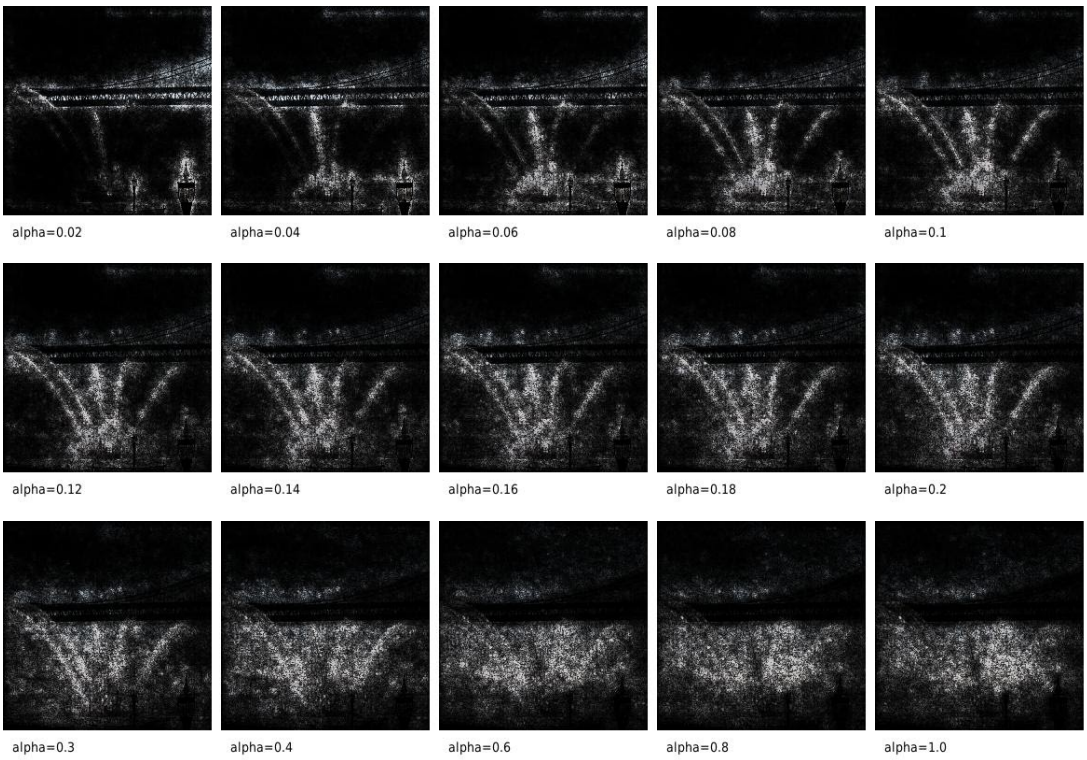


FIGURA C.3: Evolución de las atribuciones variando el coeficiente α en el método *Integrated Gradients*.

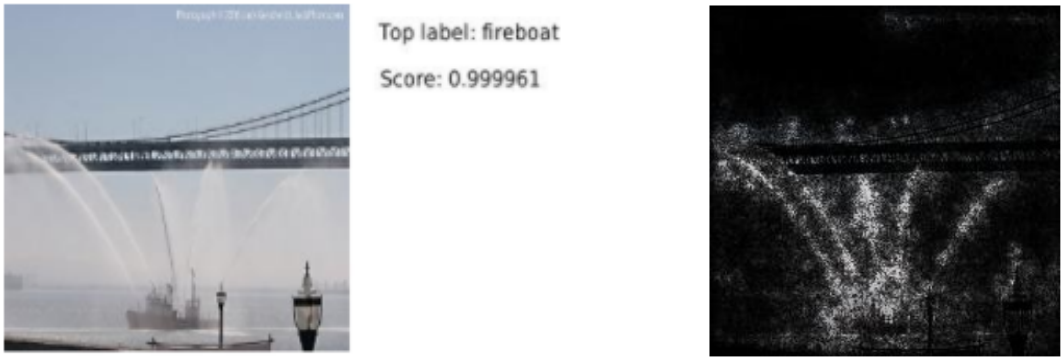


FIGURA C.4: En la figura de la izquierda se muestra la imagen de entrada al modelo más la predicción de la clase ganadora y en la de la derecha se muestran las atribuciones con el método de *Integrated Gradients*.

Bibliografía

Abdelmaguid, Ehab y col. (2018). «Left Ventricle Segmentation and Volume Estimation on Cardiac MRI using Deep Learning». En: *CoRR* abs/1809.06247. arXiv: 1809.06247. URL: http://arxiv.org/abs/1809.06247.

Avendi, M.R., Kheradvar, Arash y Jafarkhani, Hamid (2016). «A combined deep-learning and deformable-model approach to fully automatic segmentation of the left ventricle in cardiac MRI». En: *Medical Image Analysis* 30, págs. 108 -119. ISSN: 1361-8415. DOI: https://doi.org/10.1016/j.media.2016.01.005. URL: http://www.sciencedirect.com/science/article/pii/S1361841516000128.

Bekeredjian, Raffi y Grayburn, Paul A. (2005). «Valvular heart disease: aortic regurgitation». En: *Circulation* 112. PMID: 15998697, págs. 125-134. DOI: 10.1161/CIRCULATIONAHA.104.488825. URL: https://doi.org/10.1161/CIRCULATIONAHA.104.488825.

He, Kaiming y col. (2015). «Deep Residual Learning for Image Recognition». En: *CoRR* abs/1512.03385. arXiv: 1512.03385. URL: http://arxiv.org/abs/1512.03385.

Huang, Gao, Liu, Zhuang y Weinberger, Kilian Q. (2016). «Densely Connected Convolutional Networks». En: *CoRR* abs/1608.06993. arXiv: 1608.06993. URL: http://arxiv.org/abs/1608.06993.

Korshunova, Ira y col. (2016). *Kaggle competition second place description*. http://317070.github.io/heart/.

Liao, F. y col. (2019). «Estimation of the Volume of the Left Ventricle From MRI Images Using Deep Neural Networks». En: *IEEE Transactions on Cybernetics* 49.2, págs. 495-504.

Luo, G. y col. (2018). «Multi-Views Fusion CNN for Left Ventricular Volumes Estimation on Cardiac MR Images». En: *IEEE Transactions on Biomedical Engineering* 65.9, págs. 1924-1934.

Luo, Gongning y col. (2020). «Dynamically constructed network with error correction for accurate ventricle volume estimation». En: *Medical Image Analysis* 64, pág. 101723. ISSN: 1361-8415. DOI: https://doi.org/10.1016/j.media.2020.101723. URL: http://www.sciencedirect.com/science/article/pii/S1361841520300876.

Petitjean, Caroline y Dacher, Jean-Nicolas (2011). «A review of segmentation methods in short axis cardiac MR images». En: *Medical Image Analysis* 15.2, págs. 169 -184. ISSN: 1361-8415. DOI: https://doi.org/10.1016/j.media.2010.12.004. URL: http://www.sciencedirect.com/science/article/pii/S1361841510001349.

Ronneberger, Olaf, Fischer, Philipp y Brox, Thomas (2015). «U-Net: Convolutional Networks for Biomedical Image Segmentation». En: *CoRR* abs/1505.04597. arXiv: 1505.04597. URL: http://arxiv.org/abs/1505.04597.

Shrikumar, Avanti, Greenside, Peyton y Kundaje, Anshul (2017). «Learning Important Features Through Propagating Activation Differences». En: *CoRR* abs/1704.02685. arXiv: 1704.02685. URL: http://arxiv.org/abs/1704.02685.

Simonyan, Karen, Vedaldi, Andrea y Zisserman, Andrew (2013). «Deep inside convolutional networks: Visualising image classification models and saliency maps». En: *arXiv preprint arXiv:1312.6034*.

Simonyan, Karen y Zisserman, Andrew (sep. de 2014). «Very Deep Convolutional Networks for Large-Scale Image Recognition». En: *arXiv 1409.1556*.

Sundararajan, Mukund, Taly, Ankur y Yan, Qiqi (2017). «Axiomatic Attribution for Deep Networks». En: *CoRR* abs/1703.01365. arXiv: 1703.01365. URL: http://arxiv.org/abs/1703.01365.

Tencia y Woshialex (2016). *Kaggle competition first place summary*. https://www.kaggle.com/c/second-annual-data-science-bowl/discussion/19524.

Wit, Julian de (2016). *Kaggle competition third place description*. http://juliandewit.github.io/kaggle-ndsb/.

Xue, Wufeng y col. (2018). «Full left ventricle quantification via deep multitask relationships learning». En: *Medical Image Analysis* 43, págs. 54 -65. ISSN: 1361-8415. DOI: https://doi.org/10.1016/j.media.2017.09.005. URL: http://www.sciencedirect.com/science/article/pii/S1361841517301366.

Zagoruyko, Sergey y Komodakis, Nikos (2016). «Wide Residual Networks». En: *CoRR* abs/1605.07146. arXiv: 1605.07146. URL: http://arxiv.org/abs/1605.07146.

Zhen, Xiantong y col. (2016). «Multi-scale deep networks and regression forests for direct biventricular volume estimation». En: *Medical Image Analysis* 30, págs. 120 -129. ISSN: 1361-8415. DOI: https://doi.org/10.1016/j.media.2015.07.003. URL: http://www.sciencedirect.com/science/article/pii/S1361841515001024.