Universidad de Valladolid

ESCUELA DE INGENIERÍAS INDUSTRIALES

UNIVERSIDAD DE VALLADOLID

ESCUELA DE INGENIERIAS INDUSTRIALES

Grado en Ingeniería en Electrónica Industrial y Automática

Sistema para la detección de instrumental quirúrgico en vídeos de cirugía laparoscópica empleando redes neuronales

Autor:

Valbuena Pardal, Raúl

Tutor:

De la Fuente López, Eusebio

Departamento de Ingeniería de Sistemas y Automática

Valladolid, julio de 2022.

Resumen

El objetivo de este trabajo es la detección de instrumental quirúrgico en vídeos de cirugía laparoscópica. Con ello, se puede indexar automáticamente los vídeos, sabiendo qué instrumentos aparecen en él y en qué instantes. También se puede utilizar para evaluar la destreza de los cirujanos que llevan a cabo la intervención a través del seguimiento del movimiento realizado por los fórceps o el instrumento detectado. Para ello se utilizará una red neuronal que, basada en un entrenamiento previo con imágenes de instrumental en cirugía, sea capaz de localizar pinzas y herramientas de sutura en cada instante de la operación. Si esta tarea se llega a realizar en tiempo real, sería posible su implementación en un robot autónomo de asistencia al cirujano en operaciones de cirugía laparoscópica. Este robot podría inferir la siguiente acción realizar antes de recibir la orden del cirujano, minimizando el tiempo de operación.

Palabras Clave

Cirugía laparoscópica, Instrumental quirúrgico, Aprendizaje profundo, Redes Neuronales, Conjunto de datos, Algoritmo de detección de objetos, YOLO, Python, Tensorflow, KERAS.

Abstract

The goal of this project is surgical instrument detection in laparoscopic surgery videos. With this tool, the videos can be automatically indexed, knowing which instruments and in which moments appear in it. It can also be used to evaluate the skill of the surgeons performing the intervention by tracking the movement made by the detected forceps or instrument. In order to reach this goal, a neural network will be used. Based on previous training with images of surgical instruments, it will be able to locate forceps and suture tools at each instant of the operation. If this task can be performed in real time, it would be possible to implement it in an autonomous robot to assist the surgeon in laparoscopic surgery. This robot could infer the next action to be done before receiving the order of the surgeon, minimizing surgery time.

Keywords

Laparoscopic surgery, surgical instruments, Deep Learning, Neural Networks, Dataset, Object detection algorithm, YOLO, Python, Tensorflow, KERAS

Agradecimientos

Quisiera agradecer en primer lugar, a mi familia, el apoyo constante e incondicional que me ha permitido alcanzar mis metas.

En segundo lugar, a mi tutor y al departamento de robótica médica del ITAP por facilitarme las herramientas necesarias y asesorarme en este trabajo.

También quisiera agradecerle a mis compañeros y amigos las buenas experiencias que hemos compartido en esta etapa.

Índice principal

Resumen 3
Palabras Clave 3
Abstract 3
Keywords 3
Agradecimientos 5
Índice de Figuras 9
Índice de tablas 11
Índice de ecuaciones 11
1. Introducción 13
2. Objetivos 15
3. Cirugía laparoscópica 17
3.1. Introducción y origen 17
3.2. Instrumental quirúrgico laparoscópico 18
4. Redes neuronales 21
4.1. Introducción 21
4.2. Estructura 22
4.3. Entrenamiento 25
4.4. Redes neuronales convolucionales 26
5. Entrenamiento de una red neuronal para detección de material quirúrgico 29
5.1. Elección del software 29
5.1.1. Python 30
5.1.2. Tensorflow y Keras 31
5.1.3. Entorno/entornos utilizados 32
5.2. Elección de la red neuronal 33
5.2.1. U-Net 34
5.2.2. Red YOLO (You Only Look Once) 36
5.3. Elección de la versión de la red YOLO 39
5.3.1. YOLOv2 y YOLO9000 40

5.3.2. YOLOv3 43
5.3.3. YOLOv4 y YOLOv5 44
5.3.4. Conclusión 45
5.3.5. Estructura de Darknet-53 47
5.4. Preparación del conjunto de datos 49
5.4.1. Obtención de vídeos de cirugía laparoscópica 50
5.4.2. Extracción de fotogramas 51
5.4.3. Etiquetado del instrumental 54
5.4.4. Personalización de anchor boxes 58
5.5. Métodos de entrenamiento y evaluación de resultados 60
5.5.1. Métricas 60
5.5.2. Métodos de entrenamiento y herramientas 66
5.6. Evaluación del entrenamiento 70
5.7. Entrenamientos realizados 71
5.7.1. Entrenamiento con parámetros por defecto 71
5.7.2. Modificación de anchor boxes 75
5.7.3. Entrenamiento de Tiny-YOLO 78
6. Resultados finales 81
6.1. Indexado de vídeos 82
6.2. Imágenes de la detección 83
7. Conclusiones y líneas futuras 87
8. Bibliografía 89

Índice de Figuras

Figura 1 Primer endoscopio de Désermaux [5] 17
Figura 2 Trocar laparoscópico [7] 18
Figura 3 Diferentes tipos de accesorios de endoscopia [8] 18
Figura 4 Herramienta de sutura para cirugía laparoscópica.. Sutura realizada sobre una réplica de tejido en silicona. Fuente: ITAP Uva 19
Figura 5 Laparoscopio [7] 19
Figura 6 Estructura jerárquica de una red neuronal artificial [11] 22
Figura 7 Funciones de activación. Elaboración propia 23
Figura 8 Arquitecturas básicas de redes neuronales [10] 23
Figura 9 Influencia de la tasa de aprendizaje (*Learning rate)* en la convergencia. 25
Figura 10 Resultado de aplicar un kernel de suavizado 26
Figura 11 Resultado de aplicar un kernel de extracción de bordes 27
Figura 12 Ejemplo básico de grafo de tensores [15] 31
Figura 13 Arquitectura de una U-Net [17] 34
Figura 14 Resultados de diferentes versiones de U-Net en un conjunto de datos de tomografías de hígado. Morado - Positivo real, Verde - falso positivo, Amarillo - falso negativo [18] 35
Figura 15 Detección de instrumental quirúrgico mediante red YOLO 36
Figura 16 Ejemplo de estimación de cuadros delimitadores y de pertenencia a clases de la red YOLO [19] 37
Figura 17 Arquitectura de la primera versión de la red Yolo [19] 38
Figura 18 Comparación de distribución en *anchor boxes (derecha)* de YOLOv2 frente a distribución en parrilla (*izquierda)* de YOLO (ejemplo ilustrativo) 40
Figura 19 Ejemplo de offset de predicción frente *anchor box* 41
Figura 20 Esquema de categorías de clasificación [20] 42
Figura 21 Estructura de Darknet-53 [21] 47
Figura 22 Combinación de las capas de características para ofrecer salidas en tres escalas [25] 48
Figura 23 Fotogramas de los vídeos tomados en el laboratorio, también utilizados en otros proyectos orientados a la visión artificial, como la segmentación de gasas quirúrgicas en imágenes de cirugía [2] 50
Figura 24 Simulación del efecto producido por el barrido entrelazado respecto al progresivo. Se ha modificado el número de filas que cambian en cada fotograma para mejorar la apreciación del efecto. 51
Figura 25 Fotograma extraído del vídeo original en comparación con un fotograma del vídeo desentrelazado 51
Figura 26 Comparativa visual de las secuencias de imágenes obtenidas utilizando diferentes tasas de toma de datos 52
Figura 27 Pinza etiquetada según el criterio establecido 54
Figura 28 Ejemplo de imagen en la que no se etiqueta la pinza 55

Figura 29 Interfaz de etiquetado de LabelImg 55
Figura 30 Ejemplo de etiqueta generada en XML con formato Pascal VOC ... 56
Figura 31 Generación de dimensiones de anchor boxes a partir de los cuadros delimitadores del conjunto de datos desarrollado 58
Figura 32 Visualización del tamaño de los *anchor boxes* utilizando el conjunto de datos de COCO o el conjunto de datos personalizado..... 59
Figura 33 Agrupación de etiquetas y visualización de anchor boxes 60
Figura 34 Representación de las áreas de intersección y de unión de dos cuadros delimitadores..... 65
Figura 35 Representación del descenso de gradiente con momento..... 67
Figura 36 Representación de la función de pérdidas en el entrenamiento de la red con los parámetros por defecto 71
Figura 37 Registro obtenido en la prueba de la red 72
Figura 38 Ejemplos de falsos positivos con esquinas en “V” 72
Figura 39 Detección de instrumento sin fórceps 73
Figura 40 Imágenes en las que no ha detectado el objeto 73
Figura 41 Detección de la red frente a etiquetado del conjunto de datos 74
Figura 42 Representación de la función de pérdidas en el entrenamiento de la red con agrupación de dimensiones 75
Figura 43 Registro obtenido en la prueba de la red 76
Figura 44 Falsas detecciones de la red 76
Figura 45 Objetos no detectados por la red 77
Figura 46 Detecciones de la red comparadas con etiquetado en el conjunto de datos..... 77
Figura 47 Representación de la evaluación de las pérdidas en el entrenamiento de la red Tiny-YOLO..... 78
Figura 48 Registro obtenido en la prueba de la red 79
Figura 49 Falsas detecciones de la red 79
Figura 50 Detecciones de la red comparadas con etiquetado en el conjunto de datos..... 80
Figura 51 Registro de detecciones en un vídeo 82
Figura 52 Resumen de detección de material quirúrgico en un vídeo de cirugía 82
Figura 53 Ejemplos de funcionamiento de la red neuronal 83

Índice de tablas

Tabla 1 Versiones del software utilizado en el entorno de la red YOLOv3 32
Tabla 2 Comparación de precisión media (AP) y fotogramas por segundo con otras redes [21].....43

Índice de ecuaciones

Ecuación 1 Métrica precisión utilizada para evaluar la detección de la red ...63
Ecuación 2 Métrica de sensibilidad utilizada para evaluar la detección de la red 63
Ecuación 3 Cálculo del área de intersección (zona verde de la figura) 65
Ecuación 4 Cálculo del área de unión (zona amarilla y zona verde de la figura) 65
Ecuación 5 Cálculo del IoU 65

1. Introducción

La evolución de la medicina ha permitido una serie de avances que minimizan las consecuencias que sufre un paciente al someterse a una intervención. La cirugía laparoscópica permite que la cirugía sea mínimamente invasiva, lo que favorece una recuperación más rápida.

Uno de los cambios más significativos al implantar la cirugía laparoscópica es la tecnología necesaria. Al no tratarse de una cirugía abierta en la que el cirujano tiene acceso directo a la intervención, se requiere de equipos ópticos y de instrumentos especializados para acceder a la cirugía. Por este motivo es necesario una adaptación de los cirujanos para poder utilizar este instrumental. Esta familiarización es fundamental para solventar los problemas que puedan surgir a lo largo de una operación y evitar las complicaciones que puede suponer no actuar con la suficiente rapidez. [1]

Sin embargo, la introducción de este instrumental supone una ventaja desde el punto de vista tecnológico. Al ser necesaria una cámara endoscópica para que el cirujano pueda ver la intervención que está realizando, toda la cirugía puede permanecer grabada. Esto puede facilitar un análisis que permita la detección de errores, como el olvido de gasas al adquirir el mismo color que los tejidos internos [2]. En este caso, el algoritmo detectará pinzas e instrumental de sutura. A partir del conocimiento de la ubicación y del tipo de instrumental detectado, esta detección permitirá clasificar temporalmente las fases de una intervención, por su presencia, ausencia o rapidez de movimientos. Si esta tarea se lleva a cabo en tiempo real, podría ser una herramienta utilizada en un robot autónomo de asistencia al cirujano, de forma que, conociendo la fase de la intervención en la que se encuentra, pueda anticiparse a la siguiente acción que le va a solicitar el interviniente, minimizando el tiempo de cirugía. Además puede permitir la evaluación de la destreza del cirujano analizando la posición de la pinza en la imagen, a partir de la vibración y velocidad de los movimientos que realiza.

En cuanto al algoritmo de detección, se utiliza una red neuronal convolucional YOLO. Este tipo de redes proporcionan un cuadro delimitador por cada elemento que detectan en una imagen de entrada. Para desarrollarlo se utilizan las bibliotecas de código abierto Tensorflow y Keras en lenguaje Python. Una de las principales ventajas de utilizar bibliotecas de código abierto es su fácil acceso, que carece de la necesidad de invertir presupuesto de los proyectos de investigación para acceder al software.

2. Objetivos

La finalidad de este proyecto es proporcionar una herramienta de detección que pueda ser fácilmente utilizada en el futuro para llevar a cabo los análisis que sean necesarios en la cirugía laparoscópica. Por este motivo, los principales objetivos del trabajo son los siguientes:

- Encontrar la red neuronal que se adapta mejor a las necesidades y medios disponibles del proyecto.

- Preparar un conjunto de datos, haciendo uso de vídeos tomados utilizando un endoscopio, que permita entrenar y evaluar la red neuronal. Lograr que sea un recurso ampliable, para facilitar su incorporación en nuevos proyectos, que permitan obtener un conjunto de datos cada vez más completo y versátil orientado a la robótica médica.

- Entrenar a la red para que sea capaz de reconocer pinzas quirúrgicas y probar su funcionamiento. Únicamente se detecta este instrumento debido a que es el instrumental visualizado en los vídeos que se utilizan como recurso para este proyecto. La posibilidad de ampliación del conjunto de datos también permitirá la posibilidad de detección de más objetos.

- Lograr un funcionamiento fiable de la detección. Al tratarse de un recurso que tiene una finalidad de uso en ámbito médico, es fundamental que no realice falsas detecciones, que puedan hacer que los resultados de los análisis en los que se emplee sean imprecisos.

- Detección en tiempo real. Permitir que sea una herramienta accesible para indexar vídeos de cirugía, a partir del conocimiento del instrumento que se visualiza en cada momento. Para permitir su incorporación en robots autónomos de asistencia al cirujano es necesario que la detección se lleve a cabo en tiempo real.

3. Cirugía laparoscópica

En este capítulo se hará una introducción a la técnica de cirugía laparoscópica y a las herramientas utilizadas en la misma, dado que es son los objetos detectados en los vídeos por la red neuronal.

3.1. Introducción y origen

La laparoscopia es un tipo de intervención mínimamente invasiva, que sustituye a la cirugía abierta convencional. En ella, el cirujano se guía por una señal de vídeo para operar utilizando instrumental introducido a través de pequeños orificios. [3]

El término laparoscopia tiene origen en las palabras griegas *lápara* (abdomen) y *skopó* (observar), haciendo referencia a que para este tipo de cirugía se introduce una cámara para ver dentro del abdomen. [4]

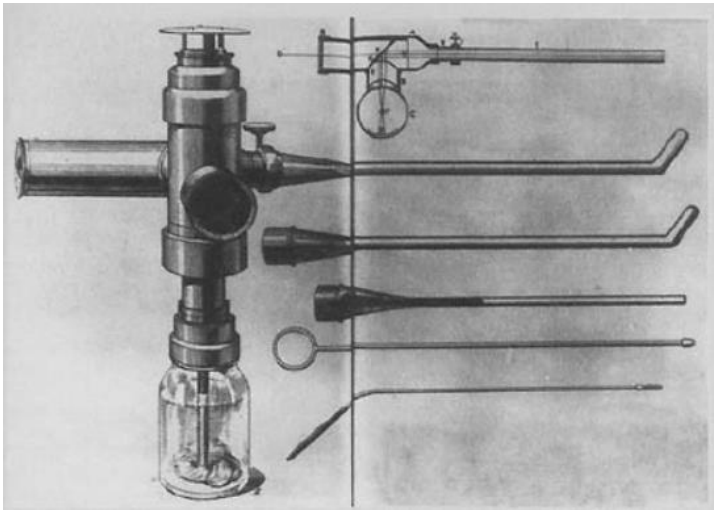

Figura 1 Primer endoscopio de Désermaux [5]

La invención del endoscopio se le atribuye a Antonin Jean Désormeaux (1815-1894), que mejoró un primer instrumento desarrollado por Philipp Bozzini (1773-1809) para lograr llevar a cabo cistoscopías y llevar a cabo intervenciones a través de la uretra. Este aparato constaba de una fuente lumínica con una óptica y un útil mecánico que se adaptaba a la apertura corporal. Desde la primera colecistectomía por vía laparoscópica, realizada por Erich Mühe en 1985, se ha producido una rápida expansión de la laparoscopia. Actualmente, esta evolución ha permitido que la mayoría de operaciones abdominales sean por vía laparoscópica, y que también se hayan beneficiado de ella otras disciplinas médicas como la ginecología, urología o cirugía torácica. [1] [4]

3.2. Instrumental quirúrgico laparoscópico

La forma de minimizar el daño de la cirugía laparoscópica frente a la cirugía abierta tradicional es sustituir una única incisión de 15 o 50 centímetros por 3 o 4 pequeñas incisiones de tamaño inferior a 1 cm, a través de las cuales se introduce el instrumental necesario para operar. [3]

Algunos de los instrumentos de cirugía laparoscópica son: [6]

- **Trocares**: Dispositivos de diámetro variable que ajustan a las incisiones y a través de los cuales se introduce el resto de instrumental. Evita lesiones mayores y permite la introducción de instrumentos más finos que la incisión sin pérdida del gas necesario para mantener la presión abdominal adecuada.

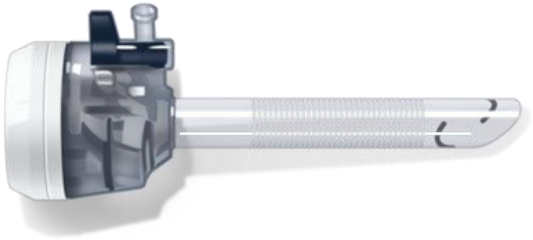

Figura 2 Trocar laparoscópico [7]

- **Instrumentos de sección y disección**: Similares a los utilizados en cirugía abierta, pero adaptados al tamaño necesario en la laparoscopia. La adaptación consiste en la reducción de su tamaño y su alargamiento, pudiendo llegar a ser de 30 centímetros o más. Existen diferentes tipos de fórceps, para adaptarse a las distintas necesidades que existen a lo largo de las intervenciones.

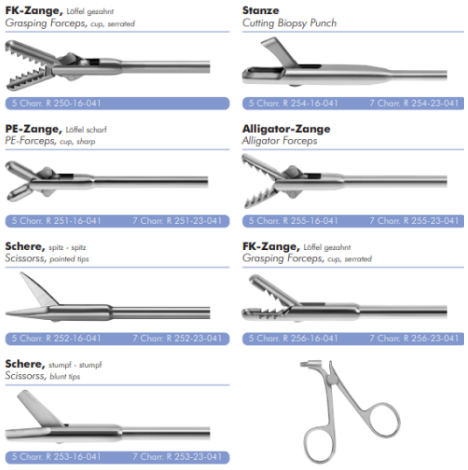

Figura 3 Diferentes tipos de accesorios de endoscopia [8]

- **Instrumentos de sutura**: La herramienta debe incorporar una aguja de un tamaño muy reducido para que esté situada en una posición de 90º para realizar la sutura y ser lo suficientemente pequeña para entrar a través del trócar.

Figura 4 Herramienta de sutura para cirugía laparoscópica.. Sutura realizada sobre una réplica de tejido en silicona. Fuente: ITAP Uva

- **Laparoscopio:** Es el instrumento fundamental para que el cirujano pueda realizar la operación viendo a través de un monitor. Incluye una fuente de iluminación y lentes de aumento para grabar una imagen que es procesada y mostrada en pantalla en tiempo real.

Figura 5 Laparoscopio [7]

La sustitución de los instrumentos tradicionales por los adaptados a la laparoscopia requiere de un significante proceso de adaptación de los cirujanos debido a que la percepción a través de todos los sentidos se ve alterada. En principio, existe una mejora para la vista, debido a que se ve la zona intervenida en una escala aumentada, y con posibilidad de observar zonas de difícil acceso. Sin embargo, al visualizarse una imagen en dos dimensiones se pierde la percepción de profundidad que aporta la tercera dimensión. Esto se ve agravado por la limitación del sentido del tacto al intervenir a través de instrumentos rígidos. [9]

Sumado a las dificultades de percepción, la visualización de los movimientos realizados de forma invertida a través de un monitor supone un problema de coordinación. Esto provoca que los movimientos ya automatizados por los cirujanos en cirugía abierta requieran de una mayor concentración en la laparoscopia y se realicen con mucha menos velocidad. Por estos motivos, este tipo de cirugía ha requerido una formación específica de los cirujanos, con entrenamiento de sutura y disección en simulador para finalmente acceder al quirófano dónde comienzan como ayudantes hasta que adquieren la habilidad suficiente para llevar a cabo la intervención completa. [9]

Este proyecto está en parte orientado a que pueda servir como herramienta de apoyo en la evaluación de los cirujanos en esta etapa de formación. Dada la importancia actual de la cirugía laparoscópica, es fundamental que, tanto los profesionales que se adaptan a esta tecnología como los estudiantes que aspiran a ser cirujanos, tengan las habilidades necesarias para implementar esta técnica.

4. Redes neuronales

En este capítulo se introducen los conceptos más básicos sobre inteligencia artificial y aprendizaje profundo, debido a que la detección del instrumental planteada se basa en el uso de una red neuronal.

4.1. Introducción

La ambición de la Inteligencia Artificial es lograr que las máquinas sean capaces de pensar como los humanos. Sin embargo, las acciones intuitivas para las personas, como pueden ser el reconocimiento de personas, animales u objetos resultan muy difíciles de implementar en computadores porque no se basan en reglas. Para lograr transferir el conocimiento necesario para realizar este tipo de acciones, se han desarrollado modelos capaces de aprender patrones a partir de conjuntos de datos. Esta tecnología se denomina aprendizaje automático, pero en muchos casos estos modelos no logran reconocer conceptos más abstractos a partir de esos patrones.

El aprendizaje profundo busca dar un paso más dentro del aprendizaje automático y utilizar las características extraídas para componer conceptos más complejos. Se denomina aprendizaje profundo debido a que se jerarquiza en varias capas desde el reconocimiento de patrones más simples a otros más abstractos. Existen diferentes modelos que implementan aprendizaje profundo, pero el más extendido en la actualidad son las redes neuronales. [10]

Las redes neuronales artificiales son modelos computacionales basados en la estructura del sistema nervioso del cerebro humano. La investigación para imitar el comportamiento de las células biológicas mediante modelos matemáticos comenzó en 1943 con el modelo de McCulloch y Pitts. En los siguientes años las investigaciones continuaron evolución con el desarrollo del concepto perceptrón (Widrow y Hoff, 1960), hasta que en 1969 Minsky y Papert publicaron un libro en el que demostraban las limitaciones de los perceptrones y el interés de la comunidad científica disminuyó. En los ochenta se retomó el interés en las redes neuronales y se lograron importantes avances como el desarrollo del algoritmo de "retropropragación" (Rumelhart. 1986). Sin embargo, el resurgimiento de las redes neuronales se ha producido con la popularización del aprendizaje profundo, al ser capaces de entrenar cada vez redes más profundas. [10] [11]

4.2. Estructura

Tal y como se ha explicado en la introducción, las redes neuronales artificiales tratan de imitar el funcionamiento del cerebro humano para alcanzar la capacidad de aprender a distinguir patrones que a priori un ordenador no puede diferenciar. Por este motivo, la estructura de una red neuronal artificial está compuesta por neuronas organizadas en distintos niveles, al igual que en un sistema neuronal biológico.

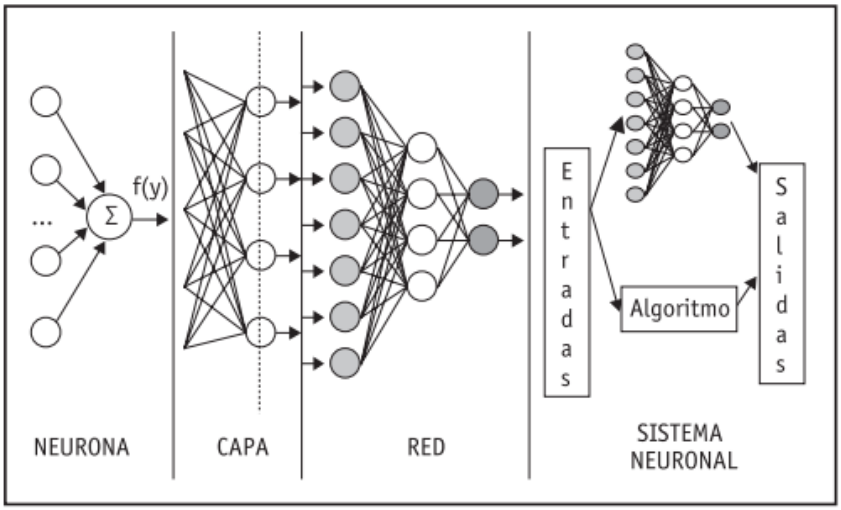

Figura 6 Estructura jerárquica de una red neuronal artificial [11]

Las neuronas son *"procesadores"* que reciben una serie de entradas para proporcionar una única salida, en función de la importancia que se considere para cada una de ellas. El valor de esta salida en función de las entradas se conoce como función de entrada. Esta función suele ser una suma ponderada, en la que cada entrada tiene un "*peso"*. El valor de la salida obtenido a partir del resultado de la función de entrada está determinado por la función de activación.

A la hora de determinar la función de activación o de transferencia se ha observado que si consiste en una relación lineal, la relación entre las entradas y las salidas de las neuronas serían únicamente lineales, lo que implica que la red no sería capaz de modelar sistemas no lineales con la suficiente precisión. Por este motivo se utilizan funciones que introducen escalones u otras formas para relacionar la salida de la función de entrada con la salida de la neurona. [12]

Algunas de las más utilizadas son las siguientes:

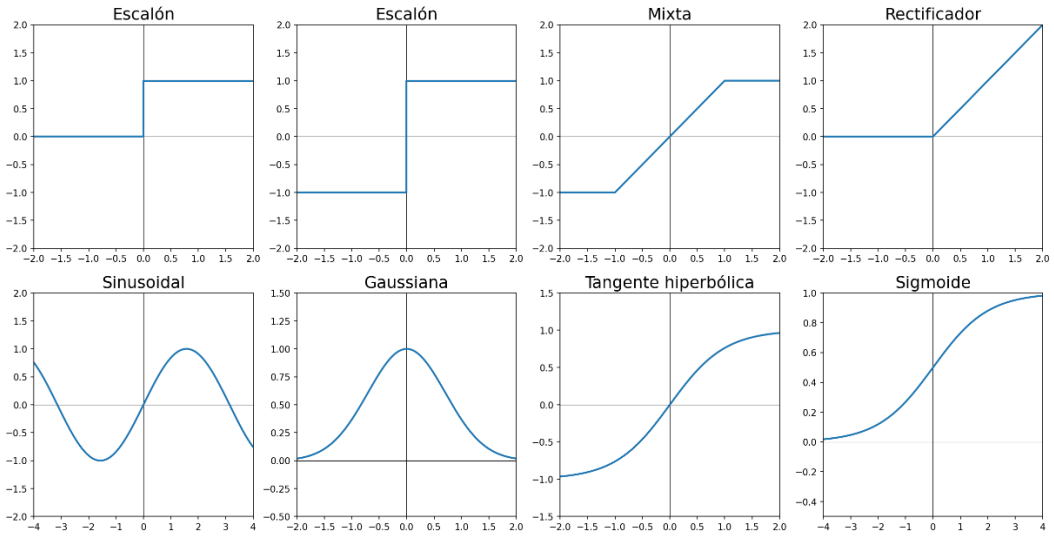

Figura 7 Funciones de activación. Elaboración propia.

Las neuronas presentadas anteriormente se organizan en capas. La estructura más simple de una red neuronal la compone una sola capa de entrada con una capa oculta de procesamiento con múltiples neuronas conectadas a todas las entradas disponibles y una capa de salida. La capa de salida de la red depende de la aplicación concreta de la misma. En el caso de que interese una sola predicción binaria, una capa de salida conectada a las neuronas de la capa anterior es suficiente. Sin embargo, si la finalidad de la red es, por ejemplo, clasificar una imagen, puede ser necesaria una neurona para cada una de las categorías a las que puede pertenecer la imagen.

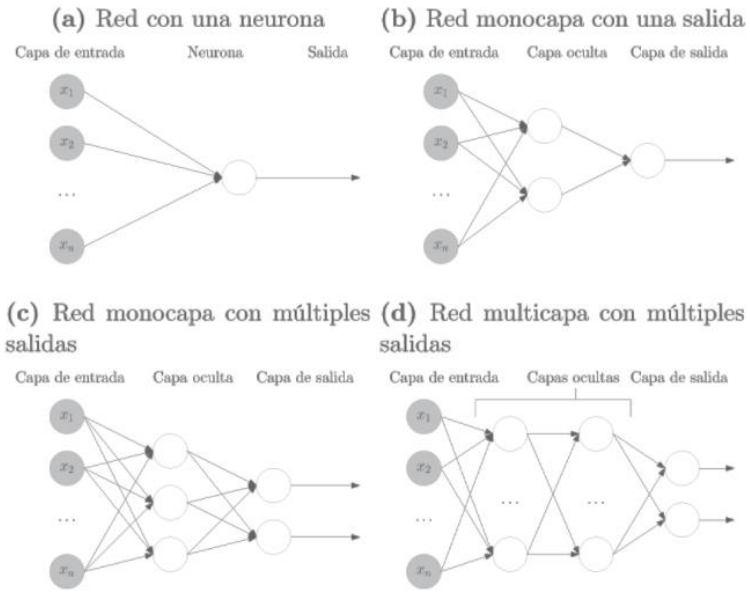

Figura 8 Arquitecturas básicas de redes neuronales [10]

Para aumentar la capacidad de predicción de la red y obtener resultados más precisos se debe aumentar el número de capas ocultas y de neuronas en las mismas, teniendo en cuenta el problema que puede suponer la sobreespecialización. Al aumentar el número de neuronas y capas se incrementa también el coste computacional y el tiempo de entrenamiento.

Si las entradas de las neuronas de cada capa se componen únicamente de salidas de las capas precedentes se consideran redes unidireccionales o de propagación hacia adelante. En cambio, cabe la posibilidad de realimentar las neuronas con las salidas de su propio nivel o de niveles posteriores para procesar mejor las secuencias de datos o las series temporales. Este tipo de redes se denominan redes de propagación hacia atrás y ofrecen las ventajas mencionadas a cambio de un mayor coste computacional. [10] [11]

4.3. Entrenamiento

Para que la arquitectura de la red sirva para obtener una salida a partir de la información de entrada, es necesario someterla a un aprendizaje para que pueda extraer los conocimientos a partir de unos patrones de ejemplo.

En este proceso de entrenamiento, la red modifica los valores de los pesos de las conexiones entre neuronas para obtener la salida deseada. En el caso del aprendizaje supervisado, es un agente externo el que compara la salida que ofrece la red frente a la deseada y ajusta iterativamente los pesos de las neuronas. Uno de los métodos más utilizados para ello es el descenso de gradiente.

El descenso de gradiente se basa en la representación del error cometido como una función continua que depende de los pesos. De esta forma, se puede buscar el mínimo de las funciones de error para encontrar los pesos que ofrecen la mejor salida posible. Para ello, la modificación del peso es proporcional a la pendiente de esta función y al parámetro de tasa de aprendizaje, de forma que cuanto mayor es la pendiente de la curva de error, mayor es la modificación de los pesos, llegando a cero cuando alcanza el mínimo (pendiente cero). La tasa de aprendizaje permite ajustar la velocidad con la que debe converger el algoritmo. Debe estar correctamente ajustada, ya que si es demasiado pequeño tardará demasiado en converger, aumentando el tiempo de entrenamiento; y si es demasiado grande puede saltarse el mínimo y oscilar sin llegar a converger.

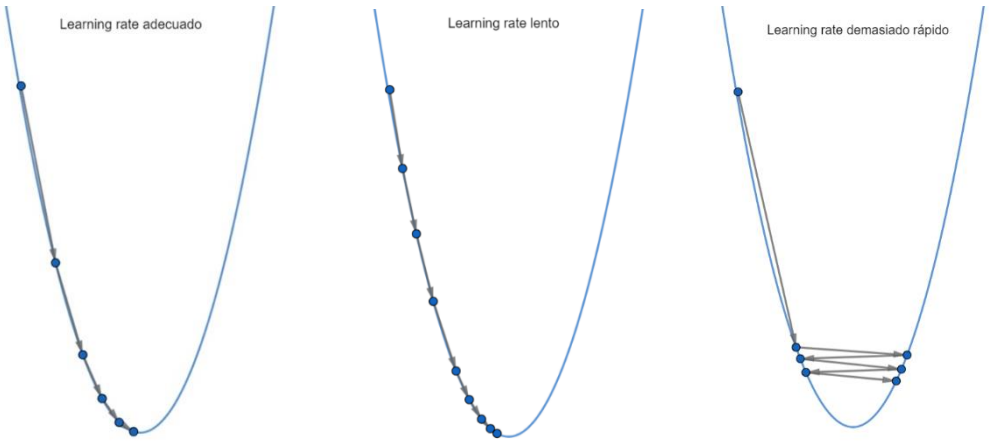

Figura 9 Influencia de la tasa de aprendizaje (*Learning rate)* en la convergencia.

4.4. Redes neuronales convolucionales

Dependiendo del tipo de datos que se desean proporcionar al algoritmo y del tipo de salida se espera obtener, se pueden implementar diferentes estructuras de redes neuronales. Se ha considerado de especial relevancia para este proyecto introducir las redes neuronales convolucionales, ya que son la base de las arquitecturas dedicadas a la visión artificial.

En primer lugar, hay que considerar que las imágenes que se proporcionan a la red son en realidad matrices de píxeles, y que por tanto se pueden llevar a cabo operaciones matemáticas sobre ellas. Por ejemplo, las imágenes utilizadas en el proyecto se pueden descomponer en 3 canales de matrices de 576 píxeles de alto y 720 píxeles de ancho. La descomposición en canales se debe a que al ser imágenes en color, el color de cada píxel se compone utilizando un valor de rojo, verde y azul.

Como su propio nombre indica, la operación matemática fundamental en este tipo de redes es la convolución. Esta operación consiste en recorrer toda la matriz de la imagen aplicando un kernel (otra matriz de menor tamaño), mediante el cual se le otorga un valor de salida a cada píxel a partir de la suma ponderada de los píxeles de su entorno.

Las convoluciones se pueden utilizar con diferentes finalidades, únicamente variando los valores del kernel. Por ejemplo, si se utiliza un kernel que le otorga el mismo peso a todos los píxeles utilizados, el resultado de cada píxel será la media de la zona que afecta esta matriz. De esta forma, la salida es una imagen con menos definición pero el ruido más filtrado. En la siguiente imagen se puede observar un ejemplo, en el que se ha utilizado un kernel de 18x18 píxeles, un tamaño muy elevado, para que se aprecien los resultados a simple vista.

Figura 10 Resultado de aplicar un kernel de suavizado

Sin embargo, la finalidad de las convoluciones en las redes neuronales es la de extraer características que permitan reconocer objetos en concreto. Para conseguirlo, los pesos de los píxeles del kernel no estarán distribuidos de manera uniforme. Un ejemplo con el que se puede visualizar esta extracción de

características son los kernel utilizados para obtener los bordes de una imagen. Para ello se utilizan matrices de convolución en las que se les otorga peso opuesto a cada extremo del kernel y peso nulo a los centrales. De esta forma, cuando a ambos laterales del píxel evaluado el valor obtenido es similar, la salida es nula, y cuando se produce una variación, esta se ve destacada en la imagen resultado. En la siguiente imagen se puede apreciar cómo el kernel aplicado destaca las variaciones en la dirección del eje vertical.

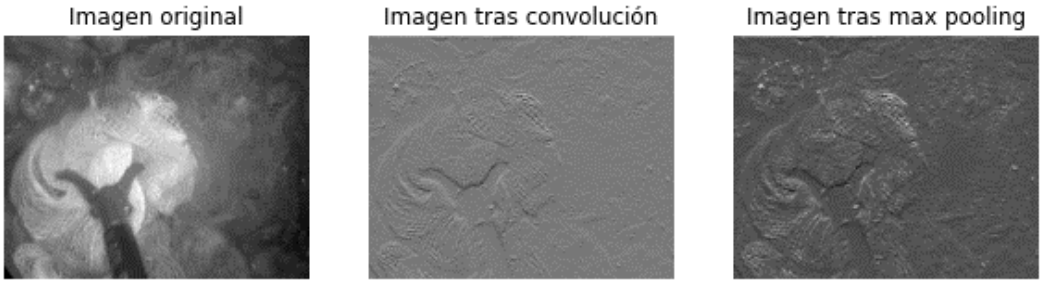

Figura 11 Resultado de aplicar un kernel de extracción de bordes

Debido a la necesidad de utilizar diferentes características para reconocer los objetos de una imagen, en cada capa de convolución se aplican diferentes kernels. Esto provoca que el resultado sea una matriz con el mismo alto y ancho que la imagen original, pero con muchas más capas de profundidad, debido a que se añade una capa por el resultado de cada convolución. Por este motivo, es necesario reducir la cantidad de parámetros que se introducen a la siguiente capa de la red. Existen diferentes métodos para reducir la resolución de las capas, en este caso se ha aplicado *max pooling*. El *max pooling* consiste en dividir la matriz en muchas matrices pequeñas y obtener el máximo de cada una de ellas, y con ello la información más relevante. A modo de ejemplo, en las figuras anteriores se ha aplicado un *max pooling* de 3x3, de modo que el valor de cada píxel está determinado por el máximo valor de 9 píxeles de la imagen original, reduciendo por tanto la resolución desde 576x720 píxeles hasta 192x240 píxeles.

Estas operaciones se llevan a cabo en cada capa de las redes neuronales convolucionales y su salida se proporciona como entrada a la siguiente capa. A medida que se profundiza en las capas de la red, se obtienen matrices cada vez más “estrechas” y “profundas”. Finalmente, hay que relacionar el tensor obtenido con la salida de la red que deseamos. Una de las formas de conseguirlo es mediante una capa densa, que relaciona cada valor del tensor con cada categoría de salida a través de un peso específico, calculado en el entrenamiento para que se ajuste al comportamiento deseado.

5. Entrenamiento de una red neuronal para detección de material quirúrgico

En este capítulo se describe el proceso seguido hasta la obtención de una herramienta capaz de detectar instrumental quirúrgico. En él se justifican todas las decisiones tomadas a lo largo del proyecto, los procedimientos para elaborar el conjunto de datos, la etapa de aprendizaje resultado y la evaluación de los resultados obtenidos.

5.1. Elección del software

Uno de los aspectos que se ha considerado más relevantes a la hora de iniciar este proyecto ha sido la elección de software. Esta decisión puede ser un factor muy relevante en un proyecto de investigación, ya que el uso de software libre permite, debido al alto coste de software de nivel industrial, la inversión de más recursos en otros aspectos que pueden mejorar los resultados del proyecto. Además, este tipo de software facilita la colaboración con otras investigaciones y su incorporación en futuros trabajos. Por estos motivos, se ha optado por trabajar con las bibliotecas de código abierto *Tensorflow* y *Keras*, en un entorno Python.

5.1.1. Python

Python es un lenguaje de programación de alto nivel y de código abierto, diseñado por Guido van Rossum en 1991. Su propósito era, a modo de afición, crear un lenguaje que simplificara e hiciera más sencilla la lectura del código. Desde su lanzamiento ya ofrecía posibilidades como el uso de clases con herencia o manejo de excepciones, reduciendo además considerablemente el número de líneas necesarias respecto a C, C++ o Java. Desde la versión 2.1 Python gestionado por la fundación *Python Software Foundation.* El objetivo de esta fundación sin ánimo de lucro es proteger este lenguaje, al mismo tiempo que ofrecer soporte y favorecer el crecimiento de la comunidad de programadores. La versión más reciente disponible de Python es la 3.10 y ya se han comenzado a publicar versiones beta de Python 3.11. Actualmente, Python es muy popular en el ámbito de la ciencia de datos y el aprendizaje automático, uno de los motivos por los que ha sido seleccionado para este trabajo. [13] [14]

Para la instalación de Python se ha optado por utilizar entornos de Anaconda. Anaconda es una plataforma que permite disponer al mismo tiempo de diferentes entornos de desarrollo, en los cuales se pueden instalar diferentes versiones, tanto de Python, como del resto de librerías utilizadas durante la programación. Es un aspecto especialmente relevante debido a que estas librerías sufren modificaciones en cada actualización, añadiendo y eliminando funciones. Por este motivo, a la hora de utilizar una red neuronal, es fundamental estar utilizando la versión correcta del software, ya que puede haber funciones de una versión más reciente que aún no están disponibles en la versión de la instalación o versiones que se hayan quedado obsoletas.

5.1.2. Tensorflow y Keras

Para la implementación de inteligencia artificial en Python se ha optado por Tensorflow, una biblioteca de código abierto para el aprendizaje automático que reúne modelos y algoritmos que simplifican y agilizan la programación.

En 2011, Google comenzó a desarrollar el proyecto *Google Brain*, con la finalidad de desarrollar sistemas de aprendizaje automático que se pudieran implementar en sus productos. El primer sistema desarrollado fue DistBelief, que ya permitía clasificación de imágenes, detección de objetos, algoritmos de aprendizaje supervisado o reconocimiento de voz entre otras características. Este sistema fue implementado en gran variedad de servicios de Google, como *Google Maps, Youtube, Google Photos* o su propio motor de búsqueda. [15]

Partiendo de la base de este software, el equipo de desarrolladores comenzó a trabajar en Tensorflow, un sistema que permitiera implementar mecanismos de aprendizaje automático en el mayor rango posible de dispositivos. De esta forma, ofreciendo más flexibilidad en la parametrización de los modelos, se pueden utilizar redes neuronales con cierto paralelismo en un rango muy heterogéneo de dispositivos a nivel de hardware. Esta variedad en la aplicación ha sido la que ha provocado que Tensorflow sustituyera a *Distbelief* y a que se haya extendido hasta su posición actual. [15]

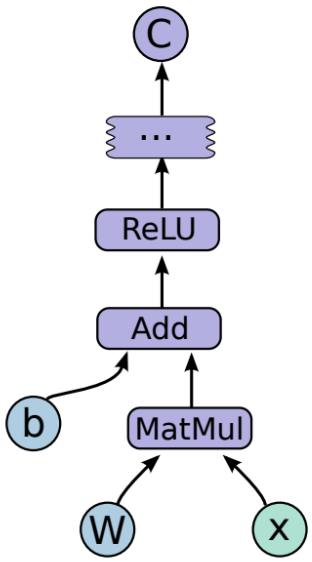

Figura 12 Ejemplo básico de grafo de tensores [15]

El funcionamiento de Tensorflow se basa en representar los modelos utilizando grafos de nodos, en los cuales se realizan operaciones con tensores. Los tensores son arrays multidimensionales que pueden contener distintos tipos de variables, como números enteros, cadenas de caracteres o números tipo *double.* Las operaciones que se realizan con los tensores en los nodos

engloban desde operaciones matemáticas básicas como la suma hasta algoritmos específicos de redes neuronales como *ReLU* o *SoftMax*. [15]

Para implementar Tensorflow se ha optado por utilizar Keras. Esta interfaz tiene una finalidad similar a la del origen de Python, la simplificación del código para hacerlo más accesible. Esto hace posible minimizar el número de líneas necesarias para desarrollar software de aprendizaje automático, al mismo tiempo que facilita su uso en plataformas como servidores, tarjetas gráficas o dispositivos móviles. [16]

Esta facilidad de uso le ha permitido establecerse como el API de referencia para formar a los estudiantes en las universidades y para desarrollar proyectos de investigación, llegando a utilizarse en organizaciones como el CERN o la NASA.

5.1.3. Entorno/entornos utilizados

Debido a la relevancia, previamente mencionada, de las versiones de las bibliotecas utilizadas, a continuación se indican las versiones de los paquetes más significativos utilizados en el proyecto:

SOFTWARE	VERSIÓN
Python	3.7.7
Anaconda	4.11.0
Tensorflow (Tensorflow-GPU)	2.1.0
Keras	2.2.4
OpenCV	4.5.5.62
Numpy	2.6.3

Tabla 1 Versiones del software utilizado en el entorno de la red YOLOv3

5.2. Elección de la red neuronal

Una vez escogido el lenguaje y entorno de programación, es necesario evaluar qué tipo de red neuronal es el óptimo para el proyecto. El objetivo principal de la red neuronal es localizar instrumental quirúrgico en vídeos de cirugía laparoscópica. A partir de esta localización, se podrá llevar a cabo un análisis del progreso de la operación o de la destreza del cirujano. Con este objetivo, se puede centrar la investigación en redes de visión artificial y se pueden descartar las redes de clasificación de imágenes. Con este tipo de redes únicamente se podría distinguir si hay presencia o no de material quirúrgico, lo cual podría ser útil para clasificar las fases de la operación pero hace imposible el seguimiento de la trayectoria del mismo.

Dentro de la amplia variedad existente de estructuras de redes neuronales, para este proyecto se plantean U-Net y Yolo, de segmentación y detección de objetos, respectivamente. A continuación se introducen las mismas y los criterios empleados para su elección

5.2.1. U-Net

En primer lugar se estudian las redes U-Net de segmentación de imágenes. La segmentación consiste en identificar el área de la imagen en el que se localiza el objeto, ajustándose estrictamente a sus bordes. Por este motivo, en lugar de predecir una clase de objeto para una imagen, predice la pertenencia de cada píxel de la imagen a una clase.

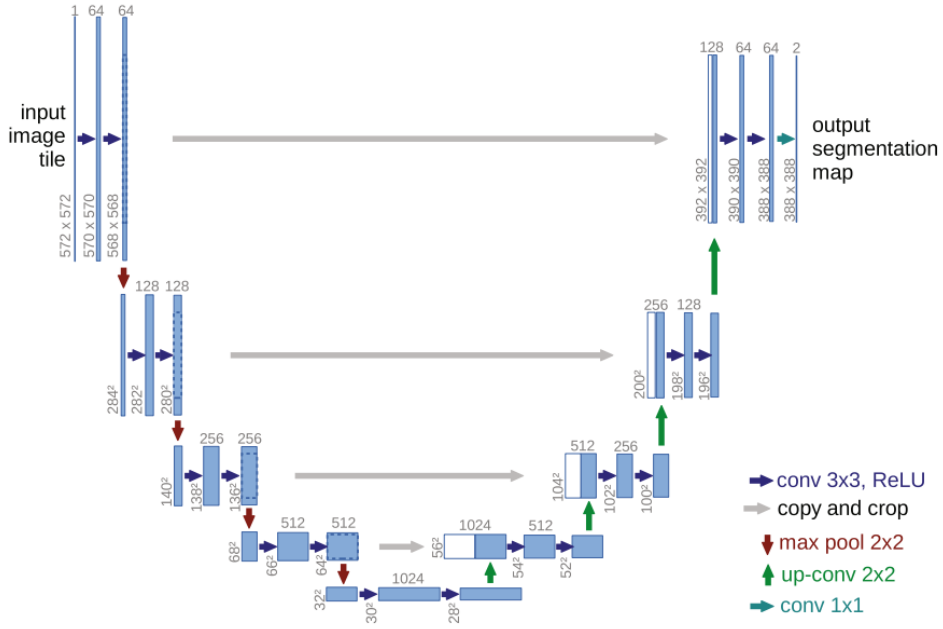

Figura 13 Arquitectura de una U-Net [17]

Las redes U-Net reciben este nombre debido a su arquitectura en forma de U. La entrada de la red y su fase de contracción son similares a las de una red convolucional clásica, en la que se aplican convoluciones seguidos de rectificaciones (ReLU), tras las cuales se aplica un *max pooling*, para reducir la resolución de la imagen. Cada vez que se reduce la resolución se dobla el número de canales de características. [17]

En la parte derecha de la figura se puede observar el proceso inverso, de reescalado. Para realizar cada reescalado se aplica una convolución de transformación y se concatena con las capas de características de la parte de contracción de la red. Cabe destacar que para utilizar estos mapas de características hay que recortarlos, debido a la pérdida de los píxeles de borde en cada convolución. [17]

Este tipo de estructuras de codificación-decodificación son populares en el segmentado de imágenes en medicina, debido a que la obtención de capas de características de diferentes escalas es útil para obtener resultados fiables.

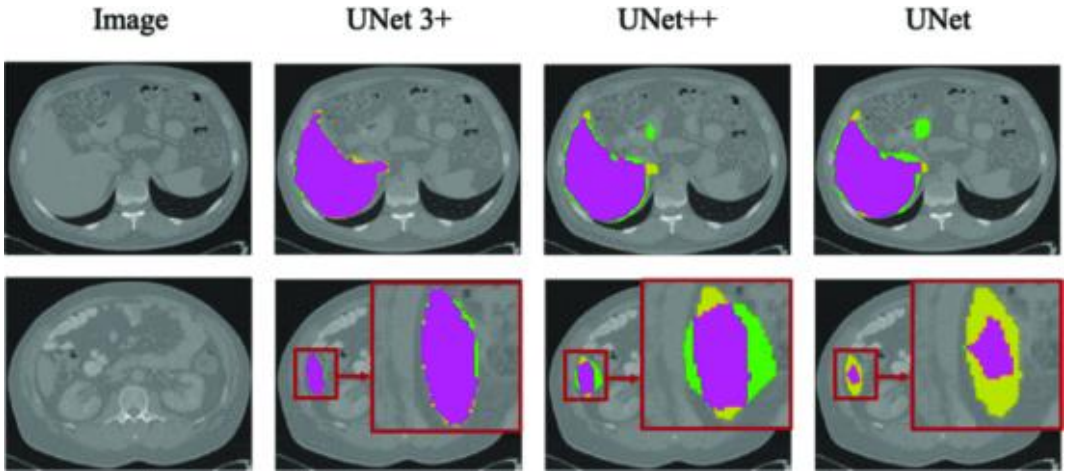

Figura 14 Resultados de diferentes versiones de U-Net en un conjunto de datos de tomografías de hígado. Morado - Positivo real, Verde - falso positivo, Amarillo - falso negativo [18]

En la figura anterior se puede apreciar el comportamiento de diferentes U-Net en la detección de masas en una tomografía de hígado. La diferencia más significativa entre las diferentes versiones de la red es la conexión entre sus capas. La red U-Net++ utiliza conexiones anidadas y densas para aumentar la relación entre las capas de codificación y las de decodificación. La red U-Net 3+ conecta capas de diferentes escalas para lograr más detalle en la escala completa. [18]

Esta red se podría implementar en el proyecto de forma que segmentara las pinzas quirúrgicas para obtener su área y con ello estimar la distancia o para analizar el movimiento de su centro de gravedad. Sin embargo, el principal inconveniente encontrado ha sido el conjunto de datos necesario. Para entrenar este tipo de redes hay que utilizar un conjunto de datos con una máscara para cada imagen en la que se indique el área que debe reconocer. El conjunto de datos del que se dispone no incluye este tipo de etiquetado para instrumental quirúrgico y se ha considerado que no es viable elaborarlo para este trabajo. El motivo ha sido que para obtener buenos resultados es necesaria una cantidad elevada de imágenes y el etiquetado de las mismas es una labor que ocuparía un porcentaje elevado del tiempo del trabajo en una labor repetitiva que no aporta valor a nivel de investigación para el mismo. Además, la segmentación precisa del instrumento no es relevante (con detectar la herramienta utilizada es suficiente) para inferir el estado o fase de la operación, el objetivo principal de uso de la red desarrollada.

5.2.2. Red YOLO (You Only Look Once)

En contraste con las redes U-Net, en las que el resultado es una segmentación del área exacta a identificar, las redes YOLO se centran simplemente en la detección de objetos. Esta detección consiste en la detección de su presencia y la estimación de un cuadro delimitador (rectángulo mínimo que incluye el objeto).

Figura 15 Detección de instrumental quirúrgico mediante red YOLO

La red YOLO fue presentada en 2015 por un grupo de investigadores encabezado por Joseph Redmon, de la universidad de Washington. El propósito de la misma era optimizar el proceso de detección de objetos en imágenes. Este tipo de clasificador ya existía anteriormente, pero constaba de dos redes neuronales separadas. La primera, llamada DPM (*Deformable Parts Model*) buscaba cuadros delimitadores en la imagen que contuvieran objetos completos. Tras esta localización, una red neuronal convolucional (R-CNN), determinaba que objeto estaba contenido en el área. La red YOLO recibe este nombre debido a que aúna las dos redes en una misma, en la que se localizan los objetos y se clasifican, por lo que solo tiene que "mirar" una vez la imagen (*You Only Look Once).* Esta simplificación logró que fuera más sencillo optimizar y entrenar el clasificador (previamente requería entrenamientos separados) e hizo posible la detección en tiempo real con el doble de precisión que el resto de clasificadores. [19]

5.2.2.1. Funcionamiento básico y estructura

Inicialmente, la red divide la imagen de entrada en un número de celdas determinado por el parámetro S, obteniendo una matriz con SxS elementos. Al unificar localización y clasificación, es necesario estimar cuadros delimitadores y probabilidad de que pertenezcan a una clase. Cada celda estima un número de cuadros delimitadores determinado por el parámetro *B*, proporcionando para cada una de ellas un total de 5 valores que definen la coordenada central del cuadro delimitador, altura y anchura y confianza de la predicción. Para determinar la clase a la que pertenece, cada celda ofrece una confianza para cada clase disponible. La celda en la que se encuentre el centro del objeto a reconocer será la responsable de la predicción.

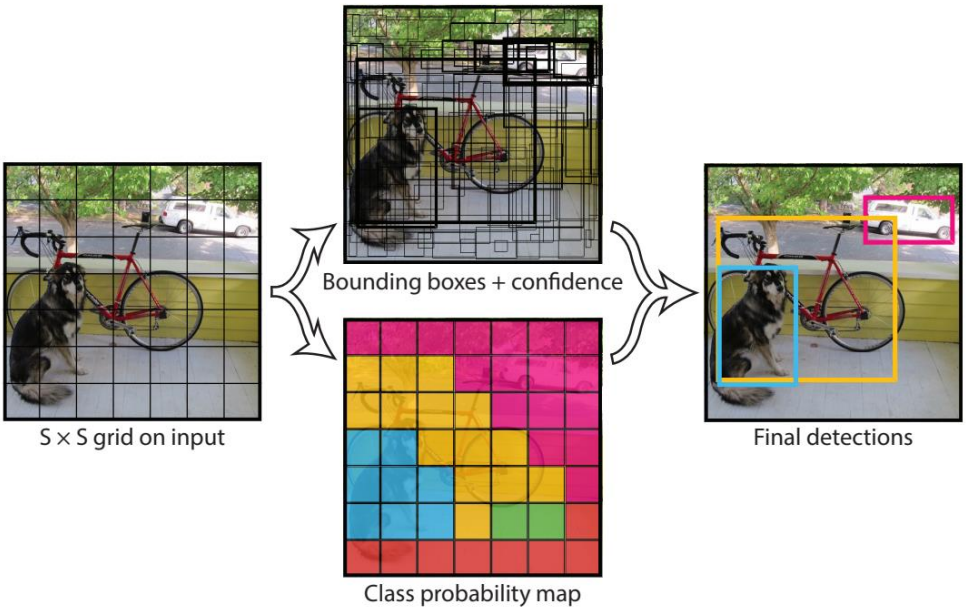

Figura 16 Ejemplo de estimación de cuadros delimitadores y de pertenencia a clases de la red YOLO [19]

La estructura de la red YOLO se basa en 24 capas convolucionales, inspiradas por la red de clasificación de imágenes GoogleNet, seguidas por dos capas densas, que relacionan todos los valores del tensor obtenido con el tensor de salida, que proporciona las predicciones indicadas anteriormente.

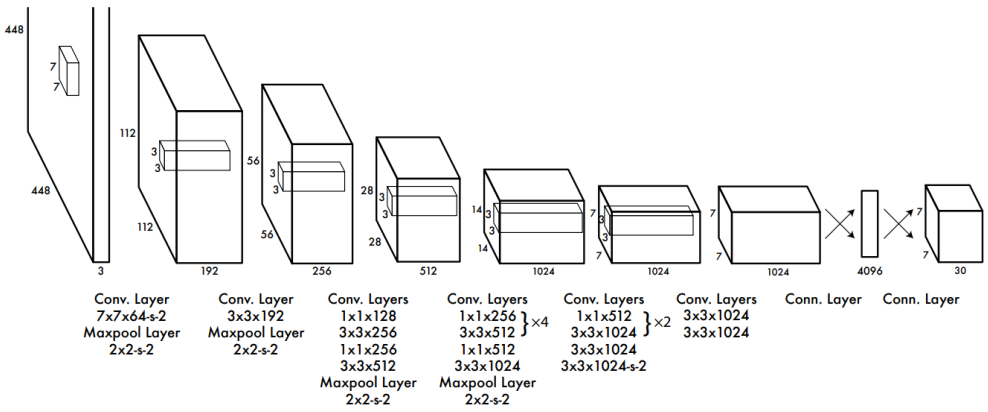

Figura 17 Arquitectura de la primera versión de la red Yolo [19]

En el desarrollo de esta red también trabajaron en una estructura que se centrara completamente en la detección en tiempo real, teniendo en cuenta que se disminuye la precisión del reconocimiento. En ese caso las 24 capas de convolución se sustituyen por solamente 9, con menos kernels de convolución. En la figura 17 se aprecia que el tensor de salida tiene un tamaño de 7x7x30. Esto se debe a que se ha utilizado un parámetro S = 7 y por tanto la imagen está dividida en 49 celdas. Además este ejemplo se basa en el conjunto de datos Pascal VOC, por lo que cada celda estima la confianza de pertenencia a 20 clases, a lo que hay que sumar los 10 parámetros generados por la predicción de dos cuadros delimitadores en cada celda, ya que se ha utilizado un parámetro *B* = 2.

El funcionamiento de esta primera versión de la red YOLO se ve limitado por el tamaño de las celdas obtenidas al hacer la división. Esto provoca dificultades si los objetos a reconocer son más pequeños que las matrices en las que se dividen la imagen. El número de objetos detectado en cada celda está limitado por el parámetro *B,* y si se encuentran dos objetos distintos en el mismo conjunto deben compartir los parámetros de predicción de clases. Por este motivo, es importante realizar una buena elección de los parámetros, adaptados a la finalidad de la red, para que pueda reconocer con la suficiente precisión sin incrementar en exceso el número de parámetros.

5.3. Elección de la versión de la red YOLO

Dadas las características de la red YOLO, se ha considerado la estructura idónea para este proyecto. El criterio principal que se ha tenido en cuenta es que la información que proporciona la salida es suficiente para llevar a cabo el análisis hacia el que está enfocada la herramienta desarrollada en este trabajo. La presencia de instrumental quirúrgico en cada fase de la operación se puede estudiar con la detección del objeto, y el análisis de la trayectoria se puede llevar a cabo utilizando el centro del cuadro delimitador detectado. El otro factor que ha influido en esta decisión es la viabilidad de elaborar un conjunto de datos. Mientras que para una red como la U-Net es necesario segmentar todas las imágenes del mismo seleccionando únicamente el área del instrumental, lo cual imposibilita adquirir un volumen lo suficientemente grande para alcanzar precisión en este trabajo, para la red YOLO es suficiente con etiquetar las imágenes utilizando cuadros delimitadores. Este factor agiliza considerablemente el proceso de etiquetado, haciendo posible la elaboración de un conjunto de datos específico para este trabajo.

Partiendo de la estructura unificada de clasificación y localización de objetos planteada en la red YOLO original, se han desarrollado una serie de evoluciones en las que se solucionan problemas y se incorporan nuevas características, por este motivo es necesario llevar a cabo un análisis de las distintas versiones existentes para estudiar cual es la que mejor se adapta al entorno en el que se desarrolla el proyecto. A continuación se describen los cambios más significativos en cada una de las opciones y las ventajas y desventajas que ofrecen para este trabajo.

5.3.1. YOLOv2 y YOLO9000

Un año después de la publicación de la primera red YOLO, su creador, Joseph Redmon, publicó un nuevo artículo en el que presentaba su evolución y una versión para la clasificación de un número mayor de categorías, las redes YOLOv2 y YOLO9000, correspondientemente.

El principal problema observado en la red YOLO en comparación con los clasificadores en los que la localización y clasificación de forma independiente era la menor precisión a la hora de localizar los objetos. Por este motivo, se desea mejorar la sensibilidad, al mismo tiempo que se mantiene la precisión de la red original a la hora de clasificar en categorías los objetos detectados. El planteamiento seguido por los investigadores para mejorar los resultados es el de añadir mecanismos a la red que incrementen la precisión de la red sin necesidad de elaborar un modelo más grande. De esta forma se logra mejorar la capacidad de aprendizaje sin perjudicar la velocidad de la red, para mantener la posibilidad de detectar en tiempo real. [20]

Para mejorar la convergencia de la red esta versión utiliza normalización por lotes en cada capa. La normalización por lotes es un método que busca mejorar la convergencia de los entrenamientos. Normaliza la salida de cada capa para evitar que las variaciones en sus parámetros afecten a las siguientes.

Para mejorar la clasificación, durante la primera fase del entrenamiento entrena a resolución completa, para que la red aprenda a clasificar las imágenes a resoluciones más altas, para después continuar mejorando la detección de la red mientras se utiliza una resolución más baja.

Otro cambio significativo en esta segunda versión de la red es la sustitución de predicción de coordenadas de los cuadros delimitadores basado en una parrilla por el empleo de *anchor boxes*.

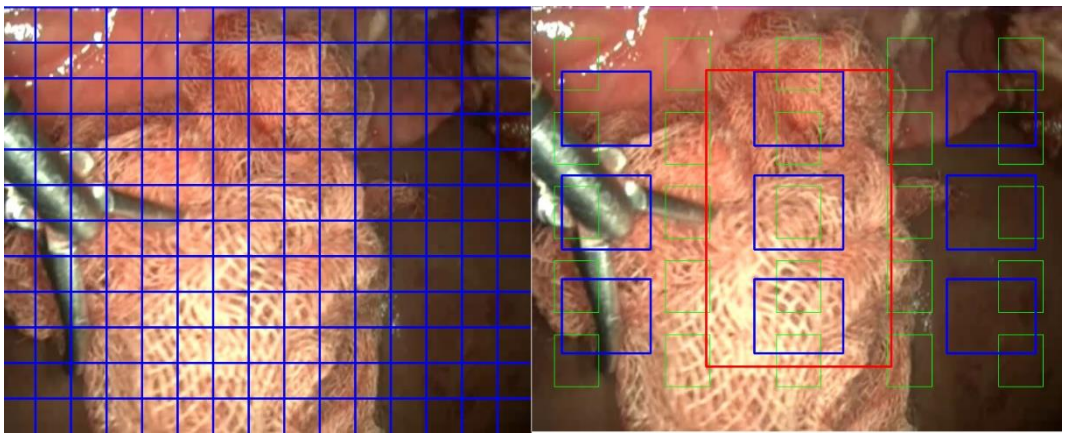

Figura 18 Comparación de distribución en *anchor boxes (derecha)* de YOLOv2 frente a distribución en parrilla (*izquierda)* de YOLO (ejemplo ilustrativo)

El uso de *anchor boxes* se basa en definir diferentes tamaños de cuadro delimitador de detección, que se distribuyen por la imagen con una separación definida por el parámetro de "paso", La predicción realizada sobre cada una de estas áreas es el offset que se le debería aplicar para que incluyera el objeto a detectar, el IOU de esta detección y la clase a la que se asocia. De esta forma, se evita la capa final densa para generar la predicción de coordenadas que utiliza la YOLO original, convirtiéndolo en una característica obtenida de forma convolucional, lo que mejora su precisión. Además facilita el reconocimiento de varios objetos superpuestos, ya que la predicción de clasificación pasa a estar asociada a cada *anchor box* en lugar de a cada celda de la parrilla. Este cambio motiva un cambio de resolución, para que el ancho y el alto de la imagen sean impares y la distribución del *anchor boxes* de mayores dimensiones coincida con un único cuadro delimitador en el centro de la imagen, útil para detectar objetos grandes que ocupan gran parte de la imagen.

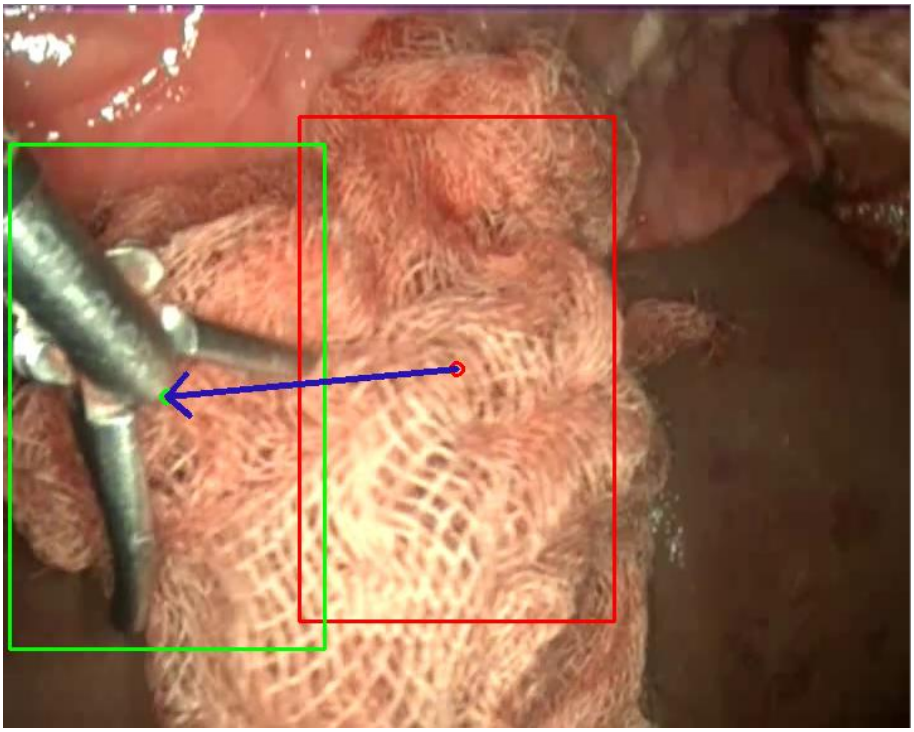

Figura 19 Ejemplo de offset de predicción frente *anchor box*

En el desarrollo de la red se planteó que a pesar de que la red puede aprender a ajustar el tamaño de los *anchor boxes*, el funcionamiento será mejor si los tamaños elegidos inicialmente son los correctos. La solución planteada se corresponde con emplear *k-means* en las cuadros delimitadores del conjunto de datos usado. *K-means* es un algoritmo que elementos por similaridad de alguna de sus características. De esta forma se obtienen 5 tamaños que se puedan adaptar a todos los tipos de objetos a detectar presentes en las imágenes proporcionadas a la red.

En esta versión también se llevaron a cabo otras modificaciones con la intención de mejorar el rendimiento. La definición del cuadro delimitador de la predicción a partir del offset respecto al *anchor box* producía problemas de estabilidad al iniciar el entrenamiento con valores aleatorios debido a que se emplazaban los cuadros delimitadores en cualquier lugar de la imagen. Para solucionarlo, se limita el offset de la predicción mediante una función de activación. También se logra mejorar el rendimiento para imágenes de alta resolución. En cuanto a la detección de detalles y en distintas resoluciones, se logra incrementar el rendimiento a base de concatenar la capa de características desde la resolución 26x26 celdas a la de 13x13 que ofrece la salida y entrenando en diferentes resoluciones que varían dinámicamente.

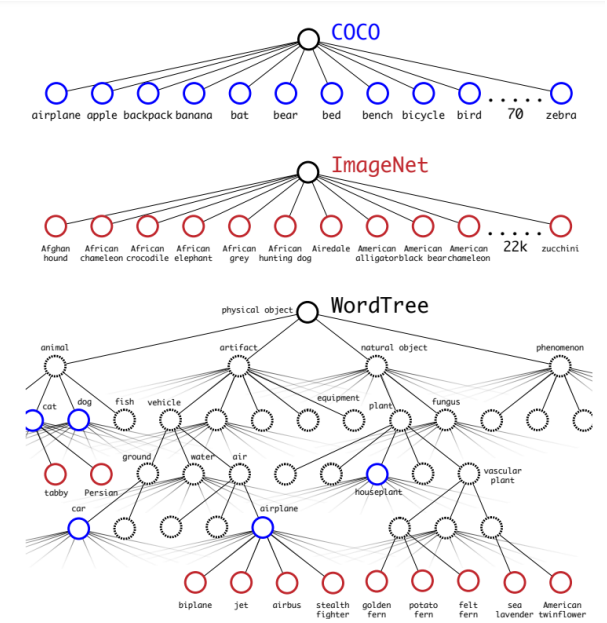

Figura 20 Esquema de categorías de clasificación [20]

Junto con esta versión de YOLO, se desarrolló otra estructura enfocada a la detección de una variedad mucho mayor de clases, la red YOLO9000. Esta red se basa en la disponibilidad de conjuntos de datos diferentes de cada tipo. En los conjuntos de datos de detección las imágenes están clasificadas para detectar categorías más generales, como objetos o animales, mientras que en los de clasificación se puede distinguir por ejemplo entre distintas razas de perro. Para juntar los conjuntos de datos se elabora un esquema jerárquico de las distintas categorías que distingue cada uno, denominado *WordTree.* A la hora de entrenar si la entrada es una imagen etiquetada para clasificación y localización se entrena la red entera, con una estructura similar a YOLOv2, y si únicamente está etiquetada para la clasificación se entrena solamente esta parte de la red. De esta forma se puede entrenar la red para distinguir entre más de 9000 clases, lo cual le otorga el nombre de YOLO9000.

5.3.2. YOLOv3

En 2018, los investigadores que desarrollaron desde el inicio la red YOLO, Redmon y Farhadi, publicaron su última mejora de la red, la versión YOLOv3. En ella se perfecciona la versión anterior de YOLO para mejorar aún más el rendimiento. [21]

Los parámetros empleados para definir el cuadro delimitador detectado continúan siendo el offset respecto al *anchor box*, el cual se limita mediante una función de activación y se suma al origen del *anchor box.* En esta red se incluye una medida de *Objectness* que determina el grado de superposición del cuadro delimitador de la predicción con el objeto. Si es el mejor que se puede seleccionar para el mismo, esta salida es 1. Para valores inferiores se establece un umbral que determine si la detección es correcta.

Para lograr extraer información a partir de diferentes escalas, también se llevan a cabo operaciones entre las capas de características de diferentes resoluciones, lo que permite obtener detalles de las capas con más resolución e información más general de las capas con menos.

Método	Backbone	AP_{50}	FPS
Faster R-CNN+++	ResNet-101-C4	55.7	53
YOLOv2	DarkNet-19	44	171
YOLOv3	Darknet-53	57.9	78

Tabla 2 Comparación de precisión media (AP) y fotogramas por segundo con otras redes [21]

Para incorporar nuevos elementos, como características de redes residuales y conseguir que sea más potente que la red empleada en YOLOv2 es necesario modificar la arquitectura de la red, ampliando considerablemente el tamaño de la misma. Con ello se demuestra que se pueden conseguir resultados significantemente más precisos que en las versiones anteriores a costa de reducir el número de fotogramas por segundo que es capaz de procesar.

5.3.3. YOLOv4 y YOLOv5

Tras publicar la red YOLOv3 el equipo de desarrolladores no continuó con su mejora debido a su preocupación por su uso en tratamiento de datos privados por parte de empresas como Facebook o Google, o con fines militares en los ejércitos. Por ello, las sucesivas mejoras han sido publicados por otros autores.

En primer lugar, con la YOLOv4 [22] se pretende optimizar la relación entre resolución de entrada de la red, número de parámetros y número de capas. Para ello se emplean diferentes redes de extracción de características, dependiendo también del equipo al que esté destinado. Por ejemplo, se emplean diferentes arquitecturas para GPUs o VPUs. Al escoger la arquitectura se tienen en cuenta los factores que influyen en la capacidad de detección en redes con una buena estructura para clasificación. Esta detección se puede mejorar incrementando la resolución de entrada, para reconocer mejor objetos pequeños e incrementando el número de capas y de parámetros, para afrontar el aumento de resolución y mantener la capacidad de reconocer varias clases u objetos.

Finalmente, la estructura utilizada en YOLOv4 es similar a la de YOLOv3 añadiendo un mecanismo de CSP en la extracción de características y mecanismos como SSP o PAN en el "cuello" de la red. Las operaciones introducidas mencionadas anteriormente afectan a la frecuencia de detección de la red debido a que aumentan el tiempo de inferencia, sin embargo, se han añadido otros métodos que no afectan a la velocidad de la red. Las operaciones de aumento de conjunto de datos ofrecen más imágenes a la red a la hora de entrenarla pero no aumentan el tiempo de evaluación de cada imagen, alguno de los mecanismos implantados es la elaboración de mosaicos o el *Dropblock.*

Estas modificaciones permiten alcanzar mejores resultados manteniendo la relativa accesibilidad, ya que se puede entrenar en GPUs desde 8GB.

Por último, la versión más reciente publicada de la red fue YOLOv5 [23], publicada por Glen Jocher, tan solo un mes más tarde que la red YOLOv4. A diferencia del resto de versiones, no se ha publicado un artículo detallando las novedades, sino que es un proyecto de código abierto publicado en GitHub. Cuenta con documentación en línea orientada únicamente a su uso e instalación. De esta forma no están disponibles las diferencias respecto a las versiones anteriores y su extracción requeriría un análisis profundo del código.

5.3.4. Conclusión

El anterior análisis de todas las versiones de la red disponibles ha permitido tomar una decisión para el proyecto en base a las necesidades y fines del mismo.

La primera versión de YOLO, explicada en el apartado en el que se describe el funcionamiento básico de este tipo de redes, ofrece la ventaja de que es el tipo de red más sencillo. Esto, a nivel académico, permite entender mejor los mecanismos utilizados y facilita la introducción de mejoras que se adapten al conjunto de datos en concreto a utilizar. En cambio, en las siguientes versiones se demuestra que algunos cambios de planteamiento en la red ayudan a mejorar su rendimiento, por lo que utilizar esta primera versión de la red conllevaría perder parte de su potencial.

La segunda versión, YOLOv2, incluye uno de los cambios más significativos en la evolución de la red, el uso de *anchor boxes*. Su uso en las siguientes versiones ha confirmado que este planteamiento ofrece más ventajas que inconvenientes y que debe estar incluido en la red utilizada en este trabajo. La versión YOLO9000 descrita en el mismo artículo científico que YOLOv2 queda descartada debido a que la mejora que introduce de detección de multitud de clases no tiene ninguna relevancia en el ámbito que se va a utilizar esta red. En este trabajo únicamente se clasificará una clase de instrumento. En el caso de dar continuidad al proyecto y mejorar la red, lo más conveniente es elaborar un conjunto de datos médico y utilizar únicamente las clases estrictamente necesarias para su uso en cirugía.

La siguiente evolución de la red, YOLOv3, no supone grandes cambios de planteamiento, pero con un incremento en el número de capas convolucionales desde 19 hasta 53, consigue unos resultados significativamente mejores. Esta modificación ocasiona un consiguiente aumento del tamaño de una red y del tiempo de inferencia, reduciendo los fotogramas por segundo que se pueden procesar. Sin embargo, se considera que, con los objetivos que se han planteado para este proyecto, la reducción de velocidad no supone una ventaja significativa. Las aplicaciones de este trabajo no incluyen detección en tiempo real, si no que tienen fines de análisis posterior, por lo que se puede priorizar la precisión sobre la velocidad. Además, al utilizar una sola clase se le puede dar prioridad a la detección sobre la clasificación, y el número de capas convolucionales es un factor clave para ello.

La versión YOLOv4 continúa implementando mejoras a nivel de rendimiento, pero a medida que se introducen nuevos mecanismos, se aumenta la complejidad de la red, dificultando su análisis y comprensión. También se incrementa su tamaño y se imposibilita su entrenamiento en el equipo utilizado. A pesar de que esté orientado a poder utilizarlo en una amplia variedad de equipos, el límite inferior que se utiliza de ejemplo es el modelo de tarjeta gráfica NVDIA GTX1080Ti. La tarjeta gráfica empleada es el modelo NVIDIA GTX1050 de ordenador portátil, que pertenece a la misma generación pero de una gama muy inferior a la mencionada en el artículo [22]. También hay una diferencia significativa en la memoria entre los 3GB de la tarjeta gráfica utilizada y los 8GB mínimos que menciona en el mismo artículo.

Por último la red YOLOv5 ha quedado descartada porque su opacidad hace imposible el aprendizaje y su comprensión mediante su uso en el proyecto. También ha influido en la decisión que está orientada a su implementación utilizando PyTorch en plataformas en línea que ofrecen servicios de uso de sus servidores. Este aspecto contradice la decisión tomada en el inicio del proyecto de uso de Tensorflow y los argumentos en los que se basó la decisión. En la documentación de esta red [23] también se recomienda que el número de etiquetas por clase sea mayor a 10.000, una cantidad inasequible para los datos disponibles en este proyecto.

Valorando las ventajas y desventajas expuestas para cada versión se decide que la red óptima para el trabajo es YOLOv3, debido a que guarda una buena relación entre precisión obtenida y complejidad de la red. Un factor clave en la decisión ha sido que no es necesario el tiempo real, debido a que con el equipo y la red utilizados no es posible lograrlo, por lo que si en una línea futura se quiere incluir en el proyecto, será necesario considerar el hardware utilizado o elegir una red más pequeña.

5.3.5. Estructura de Darknet-53

La red de extracción de características utilizada para implantar la red es Darknet-53, en concreto una versión de código abierto de Git-Hub [24]. Recibe este nombre debido al uso de 53 capas convolucionales.

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Figura 21 Estructura de Darknet-53 [21]

Según su artículo científico, en la versión YOLOv3 se preentrena la extracción de características para llevar a cabo clasificación y posteriormente se eliminan las capas finales y se usan las características extraídas. Como se puede observar en la figura anterior, esta estructura consiste de ocho grupos de bloques residuales en los que se aplica una convolución de tamaño 1x1 seguido de una de tamaño 3x3. [21]

Los bloques residuales son grupos de capas en las que se concatenan o se suman características de capas interiores para favorecer la propagación de información a las capas más profundas.

El número de filtros comienza siendo 32 y se duplica en cada grupo residual hasta alcanzar los 1024 filtros en la última convolucional. En cada grupo también se produce una reducción de escala de la mitad de ancho y alto.

Una vez preentrenado el clasificador, se entrena el detector. Este detector extrae las capas de características de los tres últimos grupos residuales para realizar la detección en tres escalas. Para combinarlas se emplea una estructura similar a las redes de tipo FPN (*Feature Pyramide Network)*, para aprovechar la información general que ofrecen las resoluciones más bajas y la información más específica en alta resolución. Para ello, las capas de menor resolución se reescalan tras aplicar una convolución para concatenarlas con la de menor resolución y finalmente extraer las características de detección. [25]

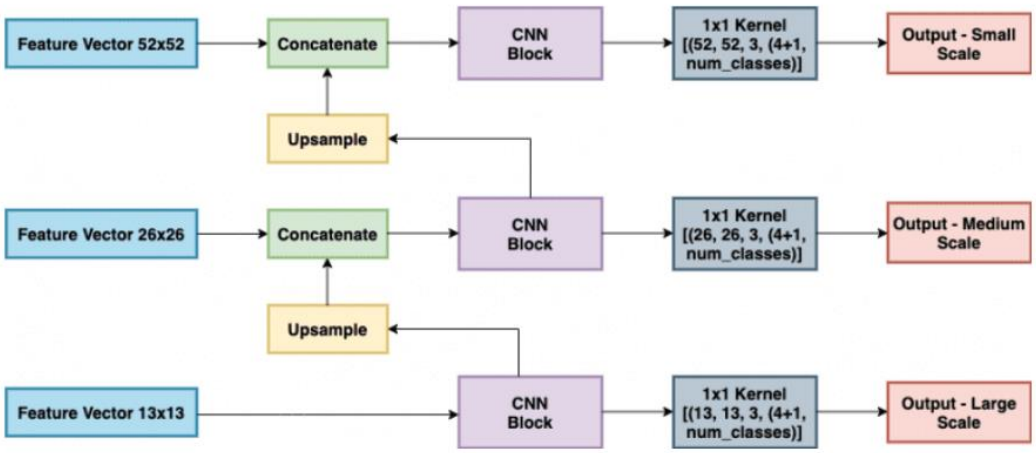

Figura 22 Combinación de las capas de características para ofrecer salidas en tres escalas [25]

Para predecir la clase de cada cuadro delimitador se utilizan clasificadores sigmoides, debido a que el uso de capas de tipo softmax hace que las clases de salida sean excluyentes. Esta es una característica que resulta contraproducente en conjuntos de datos en los que diferentes clases se pueden solapar, por ejemplo, si distingue entre personas y animales pero también entre distintos tipos de animales.

5.4. Preparación del conjunto de datos

Para entrenar una red neuronal es necesario disponer de una muestra significativa de datos etiquetados, que permitan entrenar la red utilizando imágenes en las que se proporciona la salida esperada de la red al evaluar esa imagen. La variedad de imágenes con las que se entrena la red puede limitar el rango de funcionamiento de la misma. Por ejemplo, si se entrena utilizando únicamente un escenario y un tipo de iluminación, la red puede utilizar para el aprendizaje patrones específicos que se dan solamente en esas condiciones y no ser útil en otros entornos.

La red YOLO, al unificar clasificación y localización necesita un conjunto de datos etiquetado con cuadros delimitadores que además indiquen la clase de objeto que engloban. Para entrenar este tipo de redes con propósitos generales de detección de objetos comunes, existen conjuntos de datos públicos de gran tamaño que permiten disponer de datos suficientes para el entrenamiento. El conjunto de datos más utilizado específico de detección es MS COCO [26] y contiene 328.000 imágenes con objetos de 80 categorías etiquetados. En versiones de la red como la YOLO9000 [20] se puede entrenar únicamente su parte de clasificación de objetos con conjuntos de datos de clasificación, en los que únicamente se indica el tipo de objeto que aparece en la imagen. Debido a la mayor sencillez y al uso previo de las redes de clasificación respecto a las de detección, estos conjuntos de datos son más extensos.[1]

El objetivo de este proyecto hace que el conjunto de datos necesario para entrenarlo tenga que ser específico. En primer lugar, en lugar de distinguir entre objetos de muchas clases diferentes, el propósito de esta red es distinguir pocas clases de objetos y específicos del campo de la cirugía laparoscópica. Otro factor a considerar es que las imágenes se corresponden únicamente a fotogramas de vídeos tomados a través de un endoscopio. Por tanto, la especialización que se requiere del conjunto de datos para adaptarse a las condiciones mencionadas provoca que tenga que ser elaborado manualmente, debido a que no se dispone de un conjunto de datos público de estas características.

Esta decisión permite entrenar una red YOLO para la detección requerida en el proyecto, pero limita los resultados al tamaño del conjunto de datos elaborado.

[1] Un ejemplo de ello es ImageNet, un conjunto que contiene 14.197.122 imágenes, etiquetadas siguiendo la jerarquía de Wordnet. [26]

5.4.1. Obtención de vídeos de cirugía laparoscópica

Los vídeos utilizados para la extracción de imágenes, tanto de entrenamiento como de validación, de la red utilizada han sido capturadas simulando las condiciones de una cirugía con instrumentos reales. Para estas simulaciones se han empleado órganos de animales muertos procedentes de un matadero certificado. Esta labor se ha llevado a cabo en el Instituto de Tecnologías Avanzadas de la Producción de la Universidad de Valladolid (ITAP) por Eusebio de la Fuente López y Guillermo Sánchez Brizuela, para el uso de las imágenes en distintos proyectos de investigación.

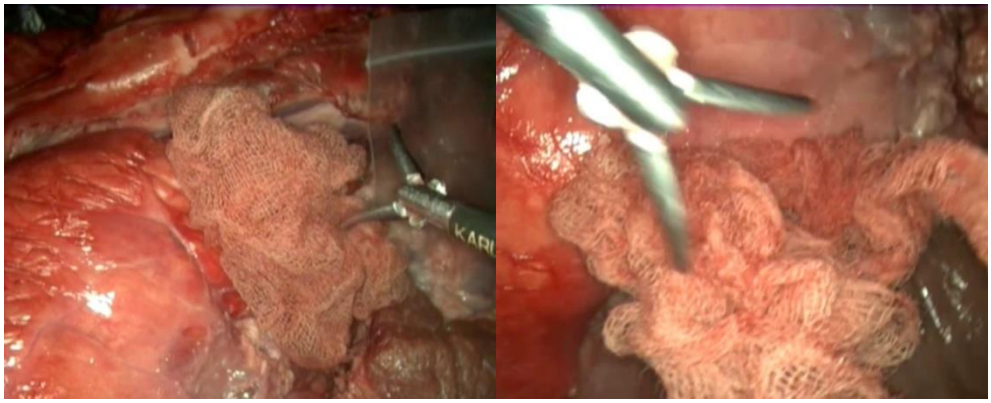

Figura 23 Fotogramas de los vídeos tomados en el laboratorio, también utilizados en otros proyectos orientados a la visión artificial, como la segmentación de gasas quirúrgicas en imágenes de cirugía [2]

A partir de los vídeos disponibles de las simulaciones de cirugía, se escogen aquellos en los que se aprecie la herramienta que se quiere detectar durante un fragmento de tiempo suficiente para disponer de varios fotogramas etiquetados, ya que únicamente se entrena la red con imágenes que contengan objetos. Finalmente se utilizan un total de 17 vídeos con una suma de duración de 26 minutos y 15 segundos.

5.4.2. Extracción de fotogramas

Tras seleccionar los vídeos que se utilizarán en el proyecto, es necesario extraer fotogramas para obtener imágenes individuales que se puedan etiquetar e introducir a la red para su entrenamiento o evaluación.

La primera consideración a tener en cuenta es el formato de salida de vídeo del endoscopio utilizado. Los vídeos utilizados cuentan con barrido entrelazado, de forma que en cada fotograma, se representan únicamente las líneas pares o las impares, alternativamente. De esta forma, al reproducir los fotogramas con la frecuencia de muestreo del vídeo, el ojo humano no aprecia que en cada fotograma se actualizan únicamente la mitad de las líneas de píxeles.

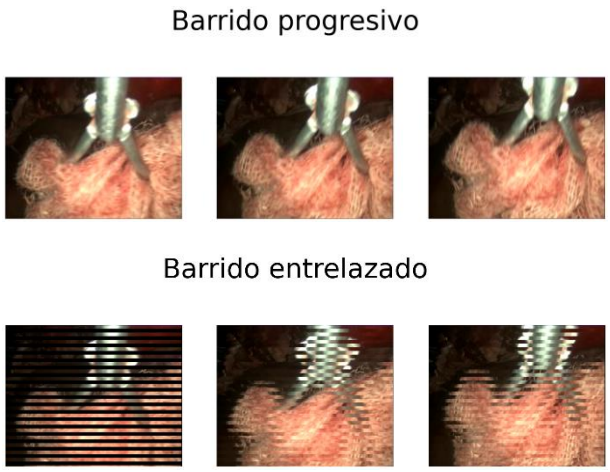

Figura 24 Simulación del efecto producido por el barrido entrelazado respecto al progresivo. Se ha modificado el número de filas que cambian en cada fotograma para mejorar la apreciación del efecto.

El problema al utilizar este tipo de barrido es que al extraer cada fotograma, la imagen obtenida consta de únicamente la mitad de filas, y al conservar el mismo número de columnas, la relación de aspecto se ve modificada.

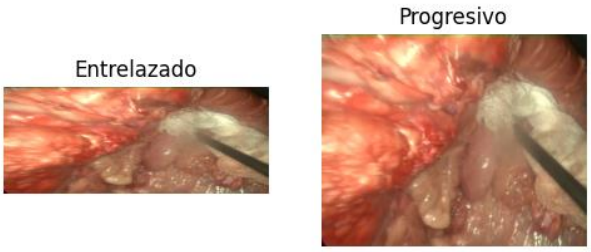

Figura 25 Fotograma extraído del vídeo original en comparación con un fotograma del vídeo desentrelazado

Por este motivo es necesario modificar el vídeo para transformarlo a un barrido progresivo, de forma que cada fotograma contenga información de todas las líneas de la imagen. Para llevar a cabo el desentrelazado, existen algoritmos que interpolan el valor de cada píxel espacial o temporalmente para completar

la imagen de cada fotograma a partir del resto de la secuencia. En este caso se utiliza la herramienta de desentrelazado que facilita el reproductor multimedia VLC, de software libre. Se aplica este procesamiento a todos los vídeos escogidos para el conjunto de datos y, por tanto, todas las operaciones realizadas posteriormente para la obtención de datos se ejecutan sobre los vídeos con barrido progresivo.

Tal y como se ha mencionado anteriormente, para poder etiquetar la pinza en imágenes con las que se pueda entrenar a la red, es necesario extraer los fotogramas de los vídeos del endoscopio. Sin embargo, dado que se dispone de 25 fotogramas por segundo, extraer todas las imágenes supondría un volumen de datos demasiado grande para etiquetar manualmente y con mínimas variaciones entre las imágenes consecutivas, que no aportarían información significativa a la red.

Por este motivo, se ha analizado la velocidad de los movimientos de la herramienta quirúrgica en los videos del conjunto de datos, para establecer una frecuencia de toma de datos adecuada para observar diferencias significativas entre fotogramas. Se ha considerado importante optimizar entre la obtención de datos suficientes, debido a que el número de vídeos que se dispone es limitado, y el tiempo empleado etiquetando el conjunto de datos, debido a que la labor de etiquetado es puramente mecánica y emplea tiempo del proyecto que podría ser invertido en investigación. Este es el motivo por el cual se hace énfasis, en que los datos etiquetados aporten realmente nueva información a la red y no se desperdicie tiempo etiquetando imágenes similares.

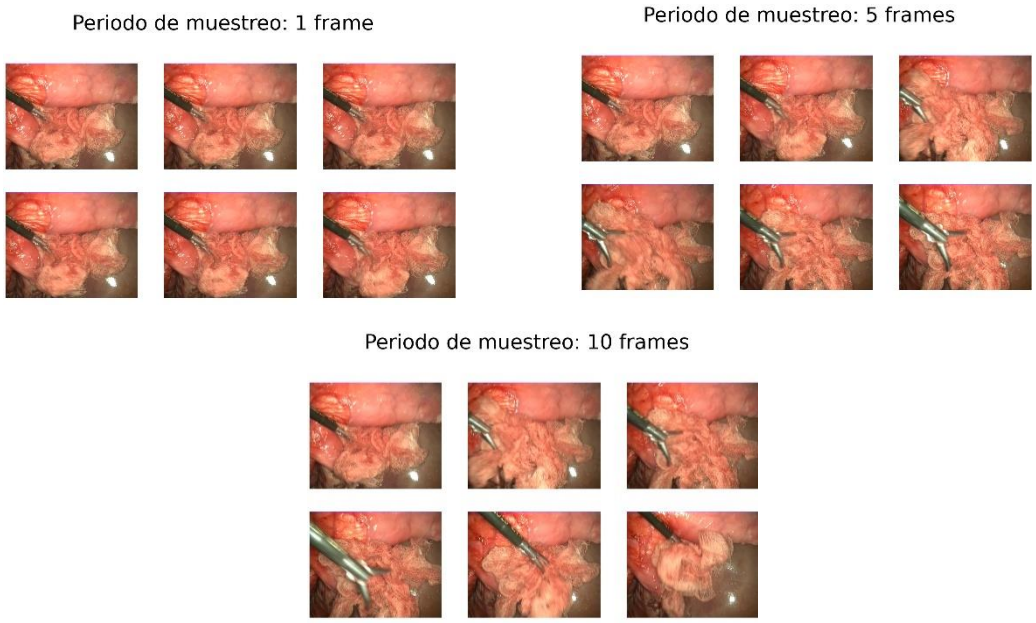

Figura 26 Comparativa visual de las secuencias de imágenes obtenidas utilizando diferentes tasas de toma de datos

Como se puede apreciar en la figura anterior, si no se establece un periodo de muestreo menor que el del propio vídeo, se obtienen numerosas imágenes con la misma información. Si se eleva el número de fotogramas entre los que se toma una muestra hasta 5, se puede observar que las diferencias aumentan entre los fotogramas, pero continúan guardando relación con el fotograma anterior. Al aumentar el periodo hasta 10 se observa que, en este caso, la posición de la pinza difiere más entre cada imagen, y por lo tanto no se aporta la misma información repetitivamente. A continuación se analiza el número de imágenes que se podrán obtener utilizando las distintas frecuencias consideradas:

- Se dispone de 14 vídeos con una duración total de 1.294 segundos, considerando 25 fotogramas por segundo, se puede estimar que se disponen de 39.375 fotogramas. Este volumen de imágenes confirma que no es asequible etiquetar todas las imágenes sin aplicar una frecuencia de muestreo.
- Si se aplica un periodo de 5 fotogramas por segundo, el conjunto de datos obtenido contaría con aproximadamente 7.875 imágenes. Se considera que es un número demasiado elevado, en relación al tiempo que requeriría en el proyecto y a las ventajas que puede ofrecer.
- Al aplicar un periodo de 10 fotogramas, el número de imágenes se debería reducir hasta, aproximadamente 3.937 imágenes. Se considera el valor óptimo, tanto por generar una cantidad de datos que permite el etiquetado manual en este trabajo, como por la variedad de información que le ofrece a la red. Estos han sido los criterios que han motivado que sea la frecuencia elegida.

Al aplicar esta frecuencia a los vídeos disponibles se cumple aproximadamente la estimación y se generan 4.011 imágenes. De ellas, se reservan las de un vídeo completo para el posterior testeo. El motivo de separar el conjunto de imágenes de prueba según este criterio, y no aleatoriamente, es que de esta forma se puede comprobar el funcionamiento de la detección en un vídeo con la garantía de que se pueden extraer conclusiones del resultado, debido a que sus fotogramas no han sido utilizados para el entrenamiento. El resultado al separarlas es de 3.539 fotogramas orientados al entrenamiento y 418 dedicados a la evaluación de la red.

La separación entre datos de entrenamiento y de validación durante el aprendizaje se realiza al iniciar el entrenamiento de la red, en este caso de forma aleatoria.

5.4.3. Etiquetado del instrumental

Una vez elegidas las imágenes que se utilizarán para el entrenamiento y evaluación de la red, es necesario etiquetar en ellas los objetos que se desean reconocer. A la hora de etiquetar, es importante establecer unos criterios claros y uniformes para que el entrenamiento se desarrolle correctamente. Por ejemplo, si en unas imágenes se etiqueta toda la herramienta y en otras solo el actuador de la misma, en una imagen puede adaptar sus valores para ofrecer una solución que va a provocar más error en otra imagen, ralentizando el entrenamiento y comprometiendo su estabilidad. Por ello, antes del etiquetado se establecen las siguientes consignas:

- Únicamente se etiquetan los fórceps del instrumental: Al etiquetar únicamente esta parte, se pretende que la red proporcione un valor más preciso del lugar en el que se encuentra actuando la pinza. Además, si en el futuro se amplía el conjunto de datos para incorporar otros instrumentos diferentes, será más fácil que la red distinga los distintos objetos debido a que el actuador de los instrumentos quirúrgicos de laparoscopia es la parte más diferencial entre ellos.

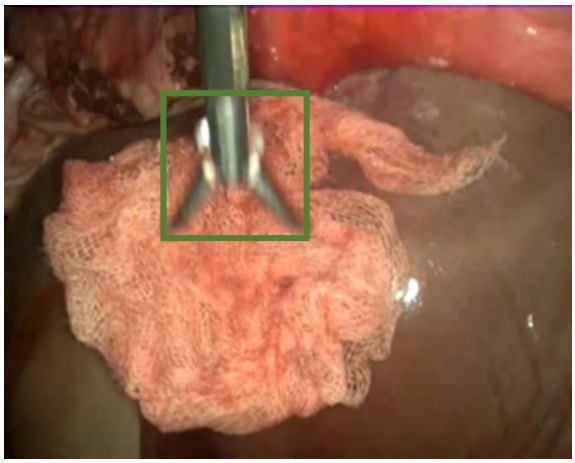

Figura 27 Pinza etiquetada según el criterio establecido

- Únicamente se etiquetan las imágenes en las que se visualiza con detalle la forma de la pinza: Durante los vídeos, se llevan a cabo una serie de acciones con la pinza, como el desplazamiento de gasas, que dificultan la percepción de la pinza, llegando a ocultarse parcial o completamente en algunos momentos. Se decide etiquetar el instrumento en estas imágenes a la red, debido a que se considera que no son suficientes para diferenciar a la pinza de otros instrumentos y que pueden ser constitutivos de falsos positivos en la detección con la red entrenada. Esta decisión no afecta negativamente al entrenamiento debido a que las pinzas no etiquetadas no afectan negativamente al entrenamiento, dado que las imágenes sin etiquetas no se usan durante el mismo, únicamente hacen que se disponga de menos datos

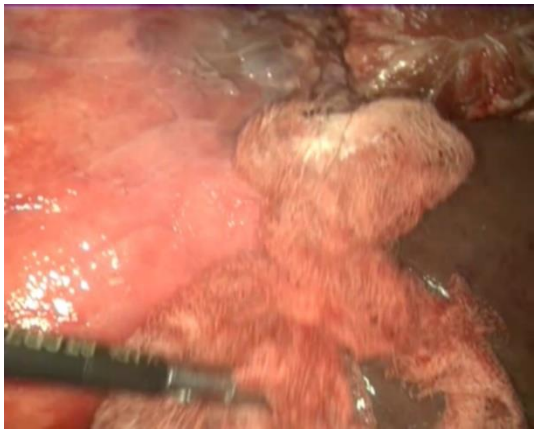

Figura 28 Ejemplo de imagen en la que no se etiqueta la pinza

Siguiendo los criterios marcados, se ha utilizado LabelImg [27], una herramienta de código abierto desarrollada en Python para etiquetar las imágenes. Este programa permite trazar cuadros delimitadores e indicar el objeto que contienen. Tras etiquetar todos los objetos de la imagen y validarla, se genera una etiqueta en un archivo independiente para cada imagen.

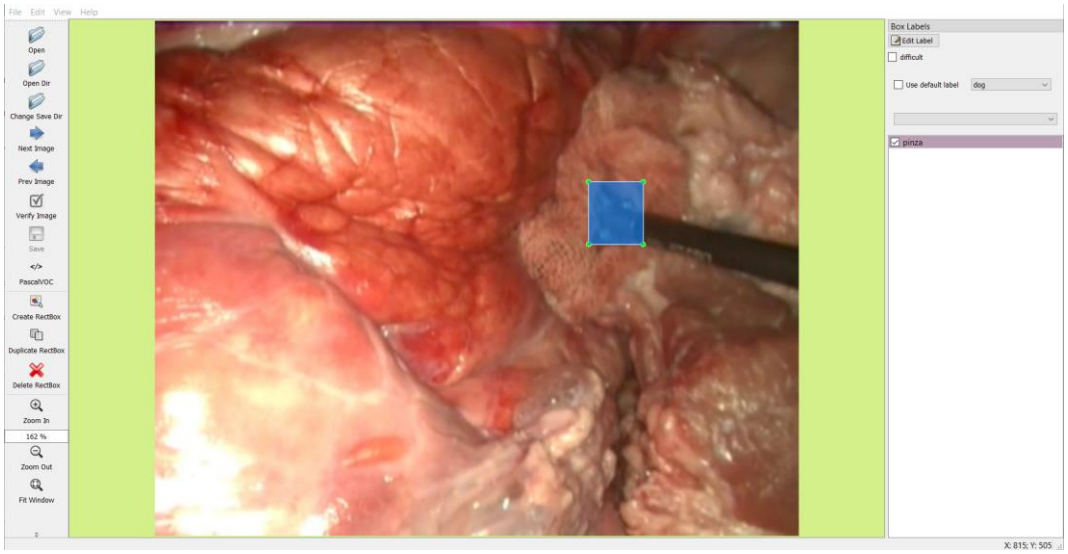

Figura 29 Interfaz de etiquetado de LabelImg

En este programa se puede elegir entre distintos tipos de formatos de etiquetas: CreateML, YOLO y PascalVOC. Se ha descartado la opción de utilizar el formato YOLO, debido a que está destinado a la versión original de la red. En este formato, la etiqueta generada indica la posición del centro del cuadro delimitador y el tamaño del mismo normalizados respecto al alto y ancho de la imagen. Esta notación no es la utilizada por la red entrenada y para convertirla a ella hay que tener en cuenta la dimensión de las imágenes, un proceso que puede dificultar la adición de imágenes de diferentes fuentes al conjunto de datos.

Por lo tanto, el formato elegido para exportar las etiquetas es PascalVOC. Esta configuración, genera para cada información un archivo XML que incorpora información básica de la imagen, como el tamaño o la ruta en la que se encuentra y la ubicación de la cuadro delimitador a partir de sus coordenadas *de (x mínima, y mínima)* y *(x máxima, y máxima)*.

Este tipo de etiquetas consolidan un conjunto de datos más útil y sencillo de usar, debido a que se puede acceder fácilmente a toda la información de las imágenes para generar etiquetas en el formato necesario para cualquier tipo de red, siempre que utilice cuadros delimitadores como entrada. También es más sencillo incorporar nuevas imágenes al conjunto de datos, aunque tengan diferentes dimensiones. Al incorporar información sobre la imagen etiquetada, se pueden tener en cuenta sus dimensiones para normalizar las coordenadas, si se desea entrenar una red que lo requiera.

```
<?xml version="1.0"?>
- <annotation verified="yes">
    <folder>frames</folder>
    <filename>VID002FRAME00000.jpg</filename>
    <path>C:\root_path\frames\VID002FRAME00000.jpg</path>
  - <source>
      <database>Unknown</database>
    </source>
  - <size>
      <width>720</width>
      <height>576</height>
      <depth>3</depth>
    </size>
    <segmented>0</segmented>
  - <object>
      <name>pinza</name>
      <pose>Unspecified</pose>
      <truncated>1</truncated>
      <difficult>0</difficult>
    - <bndbox>
        <xmin>1</xmin>
        <ymin>247</ymin>
        <xmax>395</xmax>
        <ymax>576</ymax>
      </bndbox>
    </object>
</annotation>
```

Figura 30 Ejemplo de etiqueta generada en XML con formato Pascal VOC

El formato con el que hay que introducir las etiquetas a la red entrenada es el siguiente:

```
Ruta de la imagen [cuadro delimitador 1] [cuadro delimitador 2] …

[cuadro delimitador] = xmin, ymin, xmax, ymax, clase[2]
```

[2] Xmin, ymin, xmax, ymax hacen referencia a las coordenadas de la esquina superior izquierda y de la esquina inferior derecha, correspondientemente, de la cuadro delimitador. Clase hace referencia al índice de la clase detectada en el listado de clases del conjunto de datos.

Utilizando este formato se introducen en un único documento de texto las etiquetas de todas las imágenes que las contienen. Para elaborar este fichero se ha programado un script de Python que accede a cada archivo XML incluido en una carpeta y lee los atributos para escribir la ruta y la etiqueta en el fichero de texto que se proporciona al entrenamiento. A pesar de que en el conjunto de datos utilizado solo aparezca una pinza al mismo tiempo y únicamente se detecte una clase de objeto, la programación está preparada para introducir más etiquetas y más clases. De esta forma se facilita el futuro mantenimiento y aplicación de este conjunto de datos para otros proyectos.

Al generar el fichero a partir del etiquetado en LabelImg se observa que el número de imágenes con objeto, y que por lo tanto serán utilizadas por la red, es de 609 para el entrenamiento y de 103 para la evaluación de la red. Estas carpetas contienen, respectivamente, 3.539 y 418 imágenes. De este dato se puede extraer la conclusión de que la presencia del instrumental en los vídeos utilizados es baja. A pesar de ralentizar el etiquetado, debido a que hay que clasificar numerosas imágenes que no se utilizan en el entrenamiento, es una situación similar a la que se presentaría utilizando vídeos de cirugías reales, y permite una mayor flexibilidad para crear conjuntos de datos de otros objetos que si se utilizaran vídeos orientados únicamente al reconocimiento de pinzas.

5.4.4. Personalización de anchor boxes

Desde la versión v2 de la red YOLO, en este tipo de redes se utilizan los *anchor boxes* para generar las predicciones. Tal y como Joseph Redmon concluye en el artículo científico de YOLOv2 y YOLOv9000 [20], a pesar de que la red pueda aprender a ajustar sus tamaños, se puede facilitar el aprendizaje de la red si las dimensiones introducidas para los mismos inicialmente son buenas elecciones. Para ello, se utiliza el algoritmo *k-means* para clasificar los cuadros delimitadores en una serie de grupos por sus dimensiones, introduciendo un *anchor* por cada grupo obtenido. Este método se denomina "agrupación de dimensiones".

En el caso de la red utilizada, el número de anchor boxes proporcionado a la red es 9, por lo que es el número de grupos en los que hay que clasificar el conjunto de datos creado para adaptar la red sin alterar su estructura.

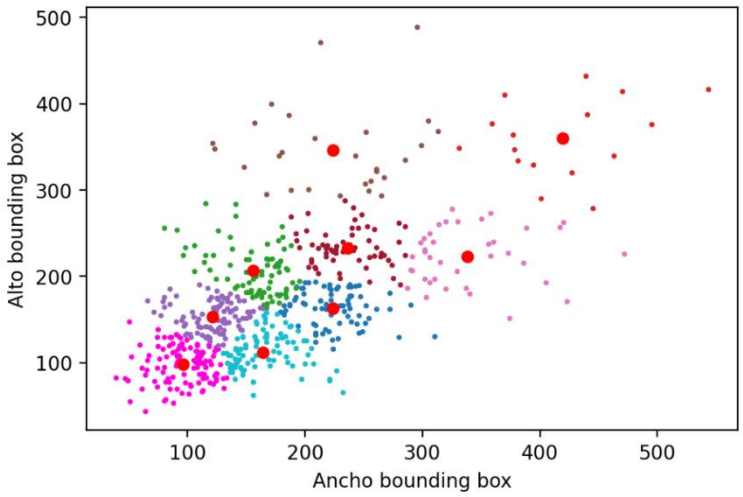

Figura 31 Generación de dimensiones de anchor boxes a partir de los cuadros delimitadores del conjunto de datos desarrollado

En la figura anterior se observa que las dimensiones de los cuadros delimitadores se concentran alrededor de la recta de pendiente = 1. Se debe a que cuando se visualiza correctamente la pinza completa, si está ligeramente inclinada, el cuadro delimitador que más se ajusta a ella tiende a ser un cuadrado. Las etiquetas en las que el ancho difiere significativamente del alto pueden ser originadas por pinzas que se visualizan desde su lateral y en una posición completamente horizontal o vertical, o que no se visualizan completamente. Los *anchors* escogidos por el algoritmo de *k-means* se ven menos afectados por estos datos porque la concentración de puntos es menor que en el centro.

En la siguiente figura se compara el resultado obtenido con el introducido por defecto en la red. Se observa que la variabilidad entre cuadros es mayor en el que ha sido entrenado con el conjunto de datos COCO. Es una consecuencia lógica, debido a que en el conjunto de datos personalizado la relación de aspecto suele ser cuadrada, como se ha indicado anteriormente, y la pinza no puede llegar a tener un tamaño de pocos píxeles porque la pinza no se puede encontrar alejada de la cámara. En el conjunto de datos de COCO, sin embargo, están etiquetados objetos de diferentes relaciones de aspecto y a diferentes distancias, por lo que se pueden captar con una escala muy reducida o ampliada.

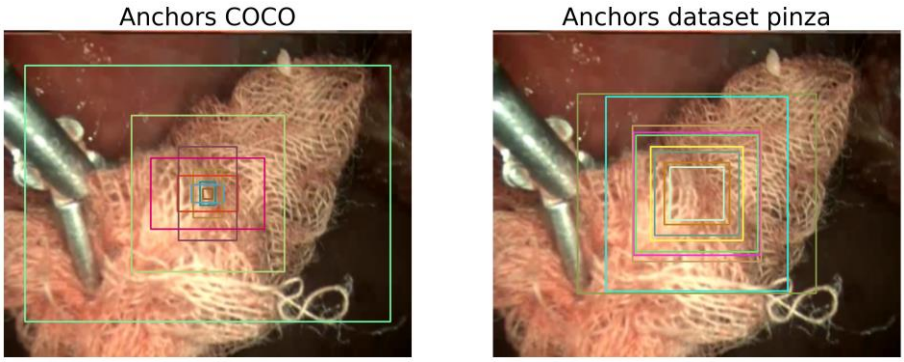

Figura 32 Visualización del tamaño de los *anchor boxes* utilizando el conjunto de datos de COCO o el conjunto de datos personalizado

Un inconveniente que se aprecia en las dimensiones generadas es que hay cuadros delimitadores prácticamente idénticos, debido a que los datos ofrecen una distribución que no permite la obtención de grupos muy diferenciados. Sin embargo, si se desea reducir el número de *anchor boxes*, se cambia la estructura de la red y ya no se podrían utilizar los pesos preentrenados de la red que se utilizan para entrenarla. Este factor hace que, para reducir el número de *anchor boxes* utilizados sea necesario disponer de otra red diferente con datos preentrenados que utilice menos *anchor boxes.*

En este caso, la otra red utilizada es una versión de menor tamaño de la YOLOv3 (*Tiny-yolo)*, que utiliza únicamente 6 anchor boxes. Para ello, se aplica de nuevo el algoritmo de k-means, agrupando los datos únicamente en 6 subconjuntos. El resultado de la clasificación es el siguiente:

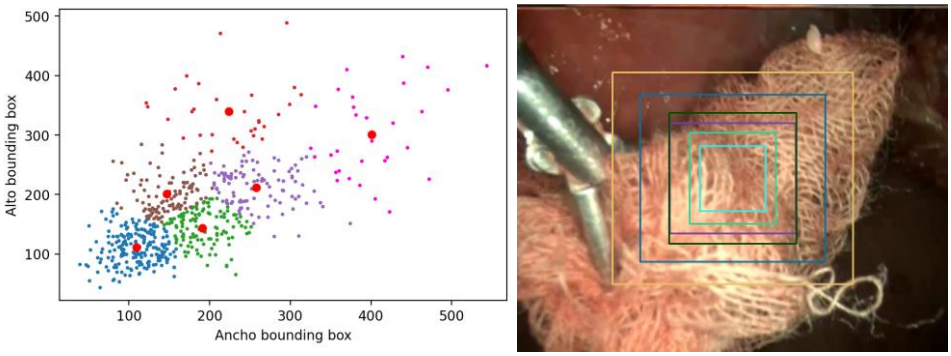

Figura 33 Agrupación de etiquetas y visualización de anchor boxes

5.5. Métodos de entrenamiento y evaluación de resultados

5.5.1. Métricas

A la hora de utilizar una red neuronal es necesario disponer de una serie de métricas que determinen cuánto se adaptan las predicciones realizadas a la salida ideal que deben ofrecer.

A lo largo del entrenamiento es necesario evaluar una función que estime el error cometido para ajustar los parámetros utilizando algoritmos que busquen minimizar su valor. Una vez se ha entrenado la red es útil disponer de medidas que indiquen lo fiables que son sus predicciones y que sirva de comparación con redes de estructura diferente o entrenadas con otros parámetros.

A continuación se definen las métricas utilizadas en este proyecto, incluyendo las incorporadas en la propia red y las de evaluación de resultados obtenidos.

5.5.1.1. Función de pérdidas

Es la métrica utilizada por la red para evaluar las detecciones realizadas durante el entrenamiento. Se trata de una función que cuantifica el error cometido, de forma que el optimizador puede ajustar los parámetros de la red para minimizarlo.

A esta función se le proporciona la predicción de salida de la red, la salida ideal proporcionada en el etiquetado, los *anchors* utilizados y el número de clases. Además se establece un umbral de IoU (Intersect Over Union), a través del cual se determina si debe ignorar o no el valor de pérdidas del atributo *confianza*. La función ofrece como salida un único valor de *pérdidas.*

Al ser una métrica optimizada exclusivamente para mejorar el aprendizaje, no se puede extraer fácilmente una interpretación del valor que ofrece. Sin embargo, se puede observar que el valor de pérdidas que ofrece como resultado es el resultado de sumar diferentes valores de pérdidas obtenidos

comparando la predicción con la etiqueta proporcionada como valor correcto. Las funciones en las que se descompone son: pérdidas de xy (centro del cuadro delimitador), pérdidas de wh (dimensiones del cuadro delimitador), pérdidas de *confianza* (confianza de la predicción de que sea un objeto) y pérdidas de *clase* (confianza de la predicción de clase). Para obtener estos valores, además de funciones como el IoU utiliza algoritmos de entropía cruzada.

5.5.1.2. Precisión y sensibilidad

Para la evaluación de los resultados obtenidos por la red se ha optado por utilizar varias métricas. Esta decisión permite valorar individualmente las distintas características de la detección. Para ello, se evalúan las imágenes reservadas para evaluación (no han sido utilizadas en el entrenamiento) y se compara la salida que ofrece la detección con la de la etiqueta correspondiente del conjunto de datos.

En primer lugar, se desea medir la cantidad de predicciones correctas, sin tener en cuenta (siempre que se encuentre dentro de umbrales) la calidad de las mismas. Para ello se realizan cuatro mediciones en la detección, obteniendo dos indicadores de salida. Las medidas utilizadas son:

- Verdadero Positivo (VP): Cuadros delimitadores generados por la red que se corresponden con un objeto real en la imagen. Se establece un umbral de 0.2 en la métrica de IoU para descartar las imágenes que no están etiquetando el objeto, manteniendo los que lo detectan aunque la dimensión del cuadro delimitador generado difiera significativamente, debido a que detectar estos fallos es la finalidad de la otra métrica. Por ejemplo, en el caso de que la red detecte el instrumento completo en lugar de solamente la pinza, este indicador ofrecería un buen valor y el IoU descendería significativamente, destacando el error cometido.

- Falso Positivo (FP): Cuadros delimitadores en los cuales el IoU de la detección con cualquier etiqueta de la imagen es menor que el umbral, lo cual indica que el objeto detectado no está etiquetado en el conjunto de datos, o la diferencia de dimensión es lo suficientemente grande para considerarlo otro objeto.

- Verdadero Negativo (VN): Imágenes en las que la red no ha detectado ningún objeto, pero tampoco hay ninguno etiquetado en el conjunto de datos.

- Falso Negativo (FN): Imágenes en las que la red no detecta ningún objeto, y contienen etiquetas en el conjunto de datos.

Tras evaluar en cada una de las imágenes detectadas, se utilizan estos valores para extraer los indicadores que se utilizarán para evaluar el comportamiento de la red:

- Precisión: Indica la proporción de cuadros delimitadores detectadas que se corresponden con objetos reales de la clase detectada. Se considera de especial relevancia en este proyecto para inferir el estado de la operación a partir del etiquetado automático de los vídeos. Si el valor de esta métrica es alto indicará que los resultados que ofrece como positivos son fiables y minimizará el error cometido al estimar el estado de la cirugía. Se calcula utilizando la siguiente ecuación:

$$Precisión = \frac{\sum(VP)}{\sum(VP + FP)}$$

Ecuación 1 Métrica precisión utilizada para evaluar la detección de la red

- Sensibilidad*:* En este caso la métrica se utiliza para evaluar qué proporción de los objetos que se deberían detectar en las imágenes están siendo reconocidas por la red. De cara a la clasificación de vídeos se considera menos relevante que la precisión, debido a que si un instrumento está presente en un fragmento de vídeo se dispone de un elevado número de fotogramas en el que se puede detectar. En cambio, si la detección está detectando objetos que no están presentes el error será más significativo. Sin embargo es un indicador relevante de cara al seguimiento de la pinza para utilizarlo, por ejemplo, en la evaluación de la destreza del cirujano. Si no se dispone de la ubicación en un número elevado de fotogramas, los resultados obtenidos mediante el análisis propuesto pueden estar distorsionados o no reflejar la realidad. Para calcularlo se utiliza la siguiente ecuación:

$$Sensibilidad = \frac{\sum(VP)}{\sum(VP + FN)}$$

Ecuación 2 Métrica de sensibilidad utilizada para evaluar la detección de la red

5.5.1.3. IoU

Las métricas anteriores sirven para evaluar la cantidad de detecciones correctas, por lo que también es necesario utilizar otras fórmulas que indiquen la calidad de las mismas. Incluso es necesario para establecer umbrales mínimos de calidad en las métricas de precisión y sensibilidad. En este caso se ha optado por utilizar la métrica de IoU (Intersect Over Union o Intersección entre Unión). La decisión se ha tomado en base a la extensión de su uso y a la facilidad de interpretación de los resultados, que simplifican la comprensión del rendimiento de detección de la red, junto al resto de indicadores.

Como su nombre indica, esta medida obtiene una relación entre la intersección y la unión de dos áreas. En el caso de la red YOLO, al utilizar cuadros delimitadores para etiquetar las salidas, las áreas utilizadas son los cuadros de salida de la red, denominados predicciones; y las etiquetas proporcionados como solución real.

Si dos áreas cuadran perfectamente, su intersección y su unión son valores equivalentes. Este sería el caso de funcionamiento ideal, con un IoU de 1, o del 100% si se evalúa como un porcentaje. En el caso de que los cuadros delimitadores se desfasen, el área en el que se encuentran desfasadas contribuye a incrementar el área de unión, mientras que el área de intersección se reduce. Esto provoca que el IoU descienda por el incremento del denominador y el decremento del numerador. Partiendo de nuevo de la situación de dos áreas perfectamente alineadas, en el caso de que en una de ellas varíe la escala, únicamente uno de los dos valores utilizados en el cálculo se verá modificado. En el caso de reducción de escala de uno de los rectángulos, el área de unión no se varía, debido a que el cuadro delimitador de mayor tamaño conserva sus dimensiones y sigue conteniendo el de menor tamaño. Sin embargo, el área de intersección se reduce, debido a que está limitado al área del rectángulo menor. En este caso el IoU desciende por el descenso de su denominador. En el caso de que aumente la escala de uno de los rectángulos se da la situación inversa, en la que aumenta el área de intersección manteniéndose el de unión, por lo que se obtiene un denominador mayor para el mismo numerador, decrementando el IoU.

En la detección, se pueden dar las dos situaciones, diferente escala y desfase entre áreas, incluso diferente relación de aspecto, tanto independientemente como simultáneamente. La influencia de ambos factores en el IoU hace que sea una buena métrica para evaluar el funcionamiento de la red.

A la hora de implementar esta medida en el proyecto, es necesario calcular todas las áreas a partir de la notación utilizada para representar los cuadros delimitadores. En ella se define el rectángulo a partir de su esquina superior izquierda y su esquina inferior derecha, debido a que se corresponden con la x e y de índice mínimo y máximo, al tratar a la imagen como una matriz en Python. A continuación, se define la fórmula utilizada para obtener el IoU y se muestra una representación de las áreas de intersección (zona verde) y unión (zona amarilla y zona verde) y de los puntos utilizados en el cálculo.

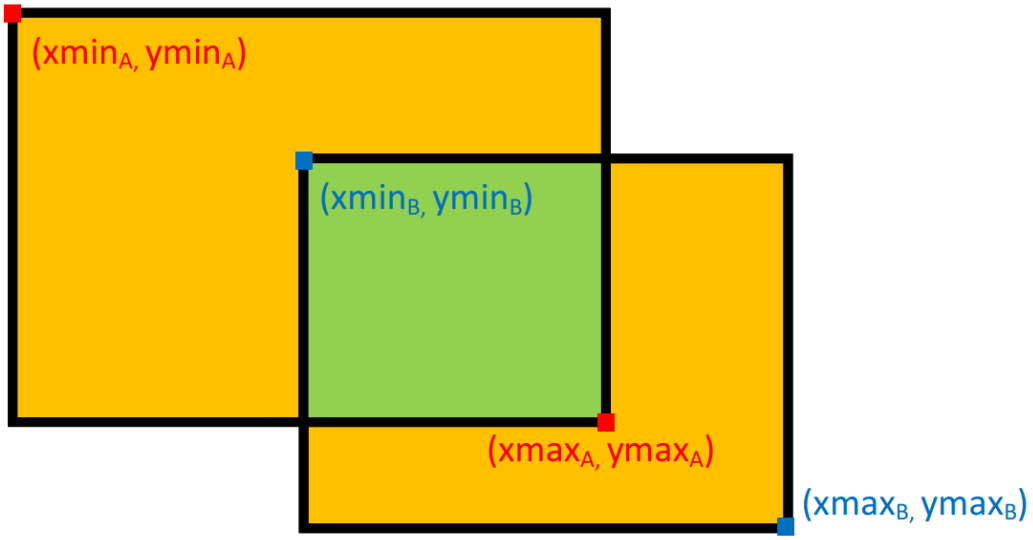

Figura 34 Representación de las áreas de intersección y de unión de dos cuadros delimitadores.

$$\begin{aligned} Área\ de\ intersección \\ = (\max(xmax_A, xmax_B) - \max(xmin_A, xmin_B)) \\ \cdot (\max(ymax_A, ymax_B) - \max(ymin_A, ymin_B) \end{aligned}$$

Ecuación 3 Cálculo del área de intersección (zona verde de la figura)

$$\begin{aligned} Área\ de\ unión \\ = (xmax_A - xmin_A) \cdot (ymax_A - ymin_A) + (xmax_B - xmin_B) \\ \cdot (ymax_B - ymin_B) - Área\ de\ intersección \end{aligned}$$

Ecuación 4 Cálculo del área de unión (zona amarilla y zona verde de la figura)

$$IoU = \frac{Área\ de\ intersección}{Área\ de\ unión}$$

Ecuación 5 Cálculo del IoU

5.5.2. Métodos de entrenamiento y herramientas

A la hora de implementar una red neuronal, la etapa fundamental es el entrenamiento. En ella se utilizan los datos generados en el conjunto de datos para ajustar los parámetros del modelo utilizado, de forma que, mediante un aprendizaje supervisado, la red adquiere la capacidad de reconocer los objetos etiquetados.

El entrenamiento de la red se simplifica debido al uso del entorno Keras. Inicialmente, hay que convertir el modelo de la red a uno compatible con los algoritmos del entorno. Una vez se dispone de este modelo, se pueden importar y utilizar tanto herramientas de Keras como propias para parametrizar el entrenamiento.

5.5.2.1. Carga y parametrización de los datos

Los datos proporcionados a la red son:

- Fichero de entrenamiento: Archivo de texto en el que se almacena la ruta de todas las imágenes utilizadas con sus correspondientes cuadros delimitadores en el formato correspondiente a la red, tal y como se indica en el apartado de *Preparación del conjunto de datos*.
- Clases: Archivo de texto que contiene una línea por cada clase que contiene el conjunto de datos. En este caso únicamente se etiquetan pinzas, pero la ventaja que ofrece la red de tipo YOLO es que se pueden añadir más clases si se completa el conjunto de datos con otros tipos de instrumental quirúrgico.
- Anchors: Archivo de texto que contiene las dimensiones de los *anchor boxes* utilizados por la red.
- Parámetros: Se determinan de forma explícita en el propio *script* de entrenamiento otros parámetros utilizados por la red, como el tamaño de las imágenes de entrada o el tamaño del conjunto de datos de validación.
- Pesos de la red: Se proporcionan los pesos iniciales de los parámetros de la red. En este caso se dispone de la red YOLOv3, basada en la red de extracción de características Darknet-53 y de una versión de menor tamaño, denominada Tiny-YOLO, destinada a una detección menos precisa pero con mayor velocidad. Es importante que el resto de parámetros se correspondan con el archivo de pesos, por ejemplo, si varía el número de anchor boxes se modifica el número de parámetros y no se pueden utilizar los pesos anteriores.

Una vez obtenidos estos datos, en primer lugar se crea un modelo de la red. Ésta se define a partir del tamaño de la imagen de entrada, los *anchors* y las clases y se le asignan los pesos iniciales. La creación del modelo de YOLOv3 y su versión reducida presentan diferencias, por lo que se utilizan funciones

distintas para crearlos. Para diferenciar la versión que se desea utilizar se comprueba el número de *anchors*, debido a que la versión completa utiliza 9 *anchor boxes*, mientras que su versión reducida tan solo utiliza 6.

Se leen todas las líneas del archivo que contiene los datos de entrenamiento y se desordenan para obtener un mejor comportamiento en el aprendizaje. De todas las imágenes se extrae un porcentaje determinado en los parámetros mencionados previamente para el conjunto de datos de validación en cada época, 10% en este caso.

5.5.2.2. Optimizador

Para supervisar el aprendizaje utilizando la función de pérdidas personalizada que se describió en el apartado de métricas es necesario compilar un optimizador.

El objetivo del optimizador es ajustar los parámetros de la red en cada iteración para minimizar las pérdidas obtenidas en la evaluación de las imágenes. Para ello, utiliza el gradiente de la función de coste, de forma que cuando la pendiente es mayor indica que el mínimo de la función se encuentra más alejado y es necesaria una mayor variación de los parámetros, y conforme se acerca el mínimo, el gradiente se reduce y el valor del parámetro se acerca más lentamente a su valor óptimo. Para ajustar la velocidad de convergencia, se multiplica el gradiente obtenido por un valor denominado tasa de aprendizaje.

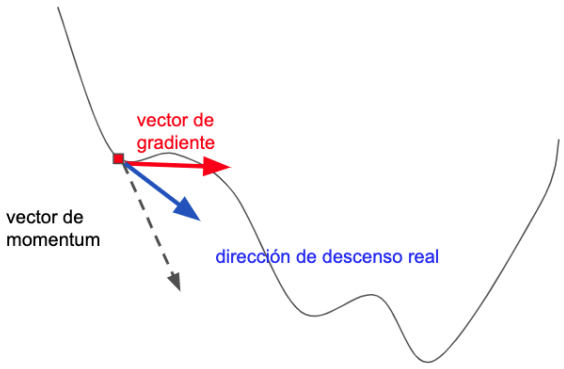

Figura 35 Representación del descenso de gradiente con momento

En este caso el utilizado es el denominado *Adam* (*Adaptative moment estimation*). Sus principales ventajas son que adapta la tasa de aprendizaje a cada peso, para ajustarse a su dimensión y su gradiente (mediante una media de los últimos valores). Además de adaptar la tasa de aprendizaje a partir de los últimos valores del gradiente, utiliza el vector de momento, que trata de

mantener la tendencia de los últimos vectores de gradiente utilizados para acelerar la convergencia. [28] [29]

5.5.2.3. Fases del entrenamiento

Una vez generado el modelo y elegido el compilador, se definen las etapas que seguirá el entrenamiento y los criterios para su finalización.

Resumiendo los apartados anteriores, los modelos utilizados son una red YOLOv3, que utiliza la estructura *Darknet-53.* También se llevarán a cabo pruebas con la versión de tamaño reducido de este modelo. El optimizador utilizado es *Adam* con una tasa de aprendizaje de $1{\cdot}10^{-3}$, con la función de pérdidas personalizada de esta estructura, denominada *yolo_loss.*

Como herramientas adicionales se utilizan los siguientes algoritmos:

- *ReduceLROnPlateau*: Algoritmo que reduce la tasa de aprendizaje si una métrica no mejora a lo largo del entrenamiento. En este caso se basa en la función de pérdidas de las imágenes de validación, que se corresponden con el 10% de las imágenes de entrenamiento. Si esta métrica no mejora en 3 épocas de entrenamiento se multiplica la tasa de aprendizaje por el factor 0,1.

- *EarlyStopping*: Detiene el entrenamiento si en una métrica no se observa un incremento mayor que el umbral determinado en un número de épocas. En este caso, si las pérdidas de validación no sufren incrementos positivos en una de las 10 últimas épocas, se finaliza la etapa del entrenamiento en la que se encuentre. Es importante utilizar las de validación y no las de entrenamiento, para basar las decisiones en la evaluación de imágenes no utilizadas para entrenar a la red, y de esta forma evitar la sobreespecialización de la misma en las imágenes de las que dispone. A pesar de ello, en el conjunto de datos que se utiliza la diferencia no es tan significativa, debido a que las condiciones de todas las imágenes utilizadas son similares, y a pesar de que las imágenes de validación se elijan aleatoriamente, al proceder de los mismos vídeos, presentarán una semejanza significativa con las imágenes utilizadas en el entrenamiento.

El entrenamiento de la red se divide en dos etapas. En la primera se congela el *cuerpo* de la red, exceptuando las tres últimas capas para ajustar la salida con unas pérdidas estables, debido a que el resto de valores pertenecen a una red preentrenada con un comportamiento estable. Esta etapa se lleva a cabo durante un máximo de 50 épocas, si no lo finaliza antes el algoritmo *EarlyStopping*. Debido a que el número de parámetros a ajustar es menor que

en la etapa con la red completa, se pueden utilizar lotes[3] de imágenes más grandes en el equipo utilizado. Con un tamaño de lote de 8 imágenes se ha logrado entrenar satisfactoriamente.

Una vez ha finalizado la etapa inicial con unas pérdidas prácticamente constantes, debido a que el número máximo de épocas es lo suficientemente elevado, se descongela el resto de capas de la red para que se ajusten sus pesos al conjunto de datos utilizado. En este caso, dado que se entrena la red completa, se requiere más memoria en la GPU, por lo que en el caso del equipo utilizado, con 3 GB disponibles, se ha observado que el número de imágenes máximo para cada lotees 2. El número máximo de épocas de esta etapa es 50 y se mantiene el algoritmo de *EarlyStopping*. Los pesos al finalizar esta etapa son los definitivos del entrenamiento y los que se utilizarán al evaluar imágenes con la red.

A lo largo del entrenamiento se almacenan copias de seguridad de los pesos de la red cada 3 épocas, con el mejor modelo obtenido en las mismas. Se ha tratado de registrar toda la información de cada época para su visualización y estudio pero debido a problemas de compatibilidad entre la versión utilizada de Tensorflow y la herramienta Tensorboard, no se han registrado correctamente. Por ello, en las gráficas de evolución de pérdidas se muestran estos valores extraídos de las copias almacenadas. De esta forma, al tomar muestras de la mejor versión cada 3 épocas se elimina el ruido de la función de las pérdidas obtenidas en cada etapa, pero se puede observar la evolución a lo largo de todo el entrenamiento.

[3] Al entrenar la red se pueden agrupar varias imágenes que se evalúan al mismo tiempo, formando lotes, para reducir el número de pasos en cada época. El tamaño de estos conjuntos está limitado por la memoria disponible en la tarjeta gráfica utilizada.

5.6. Evaluación del entrenamiento

Tras realizar el entrenamiento, se lleva a cabo la detección de las imágenes reservadas para la evaluación de la red. A pesar de que a lo largo del entrenamiento se reserva el 10% de las imágenes para validación, se opta por llevar a cabo esta prueba con imágenes de un vídeo independiente, del que no se utilizan imágenes en el entrenamiento. Además de para comparar los resultados del entrenamiento, este vídeo se utilizará para probar la detección de instrumentos de la red en vídeos de cirugía laparoscópica y no solamente en imágenes independientes.

Para evaluar los entrenamientos realizados, se desarrolla un script que ejecuta la siguiente secuencia de acciones:

- Carga de forma independiente cada imagen del conjunto de datos y su correspondiente etiqueta.
- Utiliza el modelo entrenado para obtener una lista de los objetos detectados en el fotograma.
- Si se han detectado objetos en la imagen se comparan las cuadros delimitadores con las de las etiquetas del conjunto de datos, obteniendo un valor de IoU. Este valor se acumula para obtener la media de IoU al finalizar la evaluación. Si el IoU supera un umbral se considera un verdadero positivo (VP), debido a que puede haber detectado la herramienta completa en lugar de solo los fórceps y para evaluarlo se estudia el IoU en los VP. En caso de que no supere el umbral se considera un falso positivo (VP).
- Si no se han detectado objetos en la imagen se comprueba si se había etiquetado la pinza en el conjunto de datos. En caso negativo se considera verdadero negativo, y si se ha etiquetado un objeto que no se ha detectado se cuenta como falso negativo.
- Tras evaluar todas las imágenes se utiliza el recuento de VP, VP, TN y FN para determinar precisión y sensibilidad y se calcula la media de IoU global y la de los verdaderos positivos. También, a partir de la duración de ejecución del *script* y del número de imágenes evaluadas se calcula la tasa de fotogramas por segundo con las que se realiza la detección.
- Se registran en un archivo de texto todas las detecciones y el resumen de valores obtenidos.

5.7. Entrenamientos realizados

5.7.1. Entrenamiento con parámetros por defecto

En primer lugar se entrena la red con los parámetros por defecto, en los que los *anchors* son los recomendados por el desarrollador, y por lo tanto no están adaptados al conjunto de datos utilizado. Como se ha explicado en el apartado de preparación del conjunto de datos, estas dimensiones utilizadas para determinar el cuadro delimitador de las detecciones se han obtenido agrupando las dimensiones de distintos objetos en el conjunto de datos COCO, que incluye todo tipo de objetos comunes.

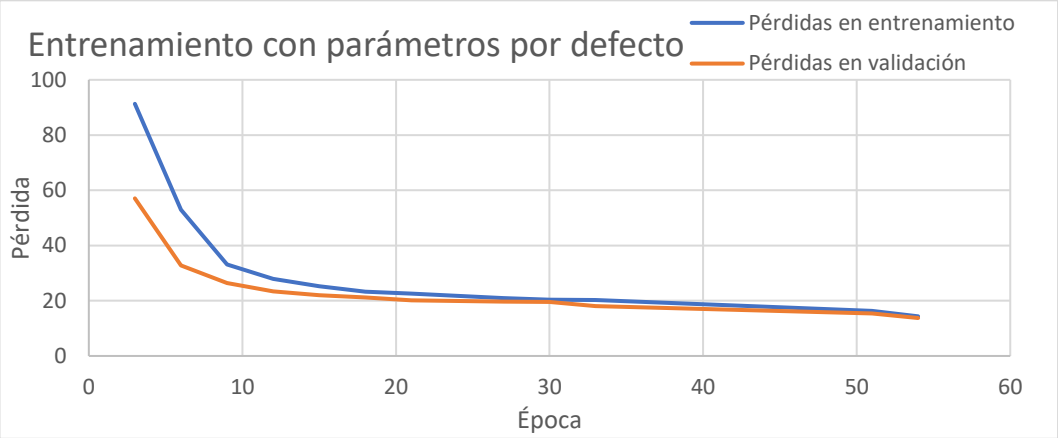

Figura 36 Representación de la función de pérdidas en el entrenamiento de la red con los parámetros por defecto

Se puede observar que el entrenamiento de la red converge sin problemas, y que la primera etapa finaliza tras 33 épocas debido al algoritmo del *EarlyStopping*. En la segunda etapa no mejoran las pérdidas de validación desde la época 54, posiblemente debido a la sobreespecialización y se finaliza el entrenamiento con unas pérdidas de entrenamiento de 14,35 y un 13,74 de validación.

Al ejecutar el script de evaluación sobre el Conjunto de datos de prueba se obtienen los siguientes resultados:

```
-----------------------------------------------------
RESUMEN
-----------------------------------------------------

Verdadero Positivo: 23
Verdadero Negativo: 284
Falso Positivo: 34
Falso Negativo: 78

Precisión: 0.40350877192982454
Sensibilidad: 0.22772277227722773

IoU en los verdaderos positivos: 0.2940890868794456

IoU global: 0.16857988914340077

Duración: 311.6212830543518

FPS: 1.3445808190415531
```

Figura 37 Registro obtenido en la prueba de la red

Se obtiene un valor de precisión menor al 50%, lo cual indica que menos de la mitad de las detecciones se pueden considerar como correctas. Tras observar este valor se recurre a estudiar las falsas detecciones que realiza la red y descubrir la razón que lo puede estar motivando.

En los siguientes ejemplos se observa que en algunas de las falsas detecciones que realiza aparecen esquinas en forma de “V”, característica que puede haber aprendido a reconocer de la pinza. También se puede deber al contraste entre el color de fondo y un objeto superpuesto, en este caso la gasa. Una posible solución es el uso de más datos de entrenamiento.

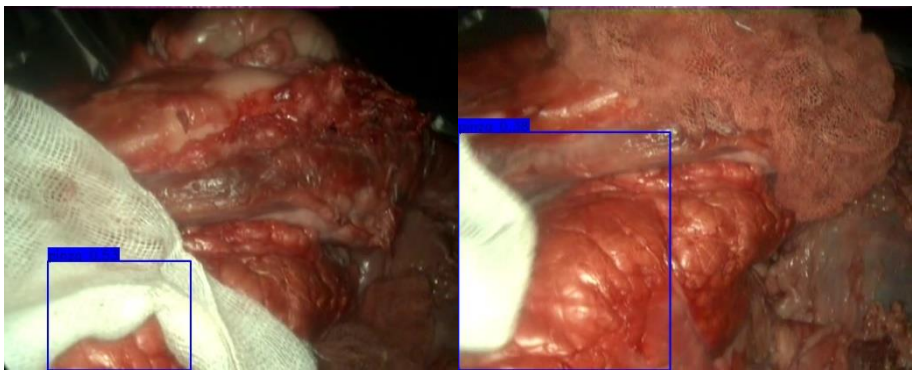

Figura 38 Ejemplos de falsos positivos con esquinas en “V”

También se observa que en algunas imágenes se ha detectado la herramienta cuando no aparece el fórceps, cuando se han etiquetado las imágenes para evitar este funcionamiento. Dado que todo el instrumental para cirugía laparoscópica debe acceder a través del trócar, esta parte es un elemento común y su detección haría imposible la distinción entre distintos instrumentos. Este comportamiento se puede deber al uso de *anchors* no personalizados. Al no utilizar *anchors* que se ajustan perfectamente a la herramienta a detectar, durante el entrenamiento, es posible que aprenda características del entorno que rodea a la pinza y que por ello detecte objetos en el mismo cuando no están presentes.

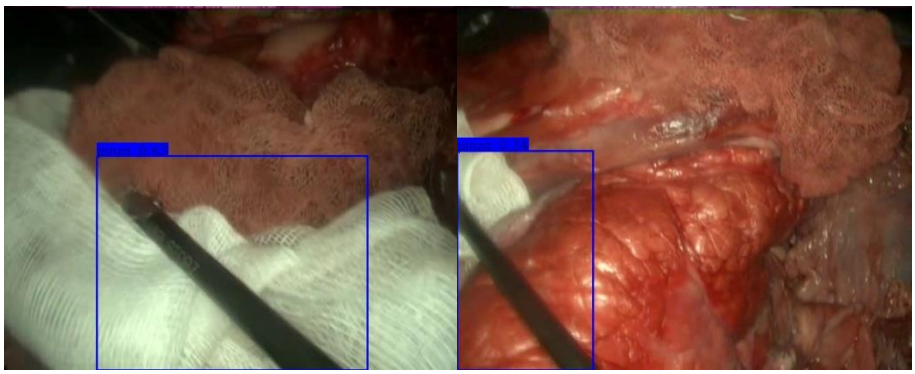

Figura 39 Detección de instrumento sin fórceps

La siguiente métrica analizada es la sensibilidad. Su valor del 22,7% indica que no se detectan tres cuartas partes de los objetos presentes. Este valor se puede deber a que los datos de entrenamiento no son suficientes y en situaciones de posición, tamaño o iluminación que no aparecen en el conjunto de datos de entrenamiento no se realiza la detección. También se observan problemas al detectar la pinza cuando se encuentra en movimiento o no se aprecia con definición.

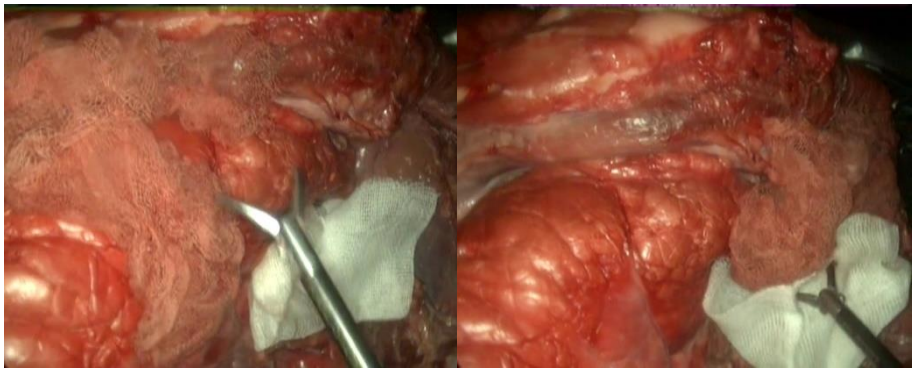

Figura 40 Imágenes en las que no ha detectado el objeto

En cuanto al IoU, globalmente tiene un valor del 16,85%, afectado muy negativamente por las bajas tasas de precisión y sensibilidad. Para observar mejor el comportamiento de la red se estudia el IoU en las detecciones correctas (VP). En este caso se obtiene un valor del 29,41%, lo cual indica que los cuadros delimitadores de la predicción no se ajustan al etiquetado de los objetos del conjunto de datos. Esto se debe al uso de *anchor boxes* estándar, que pueden ser útiles para detectar una gran variedad de objetos de distinta relación de aspecto y escala, pero se adaptan peor a las dimensiones de la pinza.

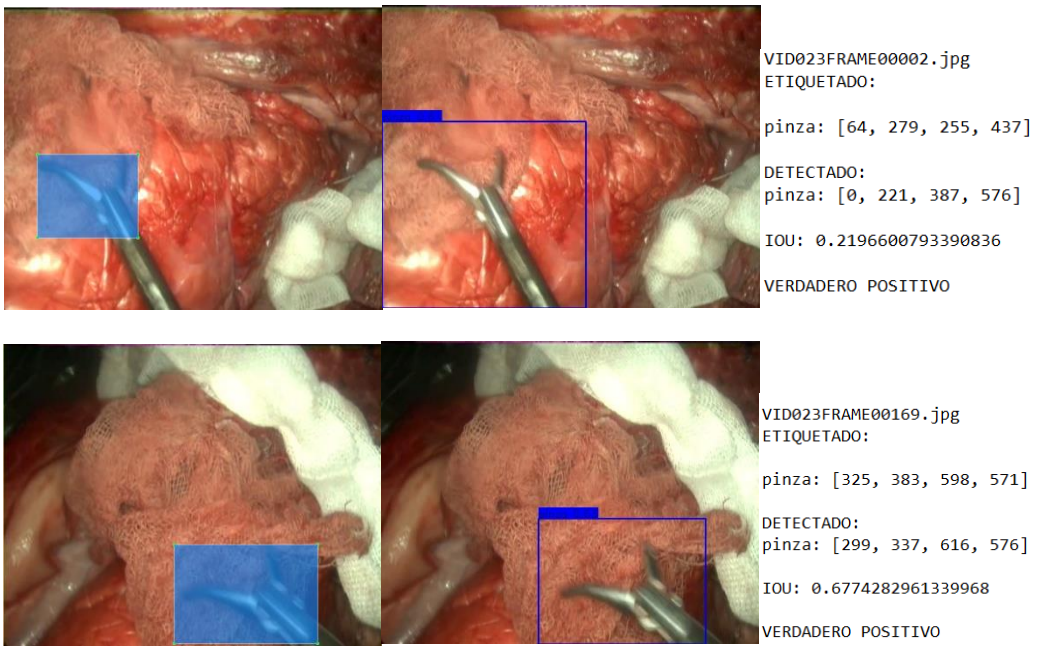

Figura 41 Detección de la red frente a etiquetado del conjunto de datos

Se observa también que la detección es lenta y no permite su uso en tiempo real en este equipo, dado que únicamente procesa 1,34 fotogramas por segundo. Sin embargo, este es un problema que se debería solucionar al utilizar un equipo profesional, debido a que la detección se está llevando a cabo con una tarjeta gráfica Nvidia GeForce GTX1050 de ordenador portátil con 3 GB de memoria interna y en la página web del desarrollador se menciona la detección en tiempo real a 30 FPS con una tarjeta gráfica Pascal Titan X [30], de gama muy superior a la utilizada.

5.7.2. Modificación de anchor boxes

Para mejorar los resultados obtenidos con los parámetros por defecto, se adaptan los *anchor boxes* utilizados al conjunto de datos etiquetado para el proyecto mediante la agrupación de dimensiones, tal y como se explica en el apartado de preparación del conjunto de datos. Al realizar de nuevo el entrenamiento se obtiene la siguiente evolución de las pérdidas:

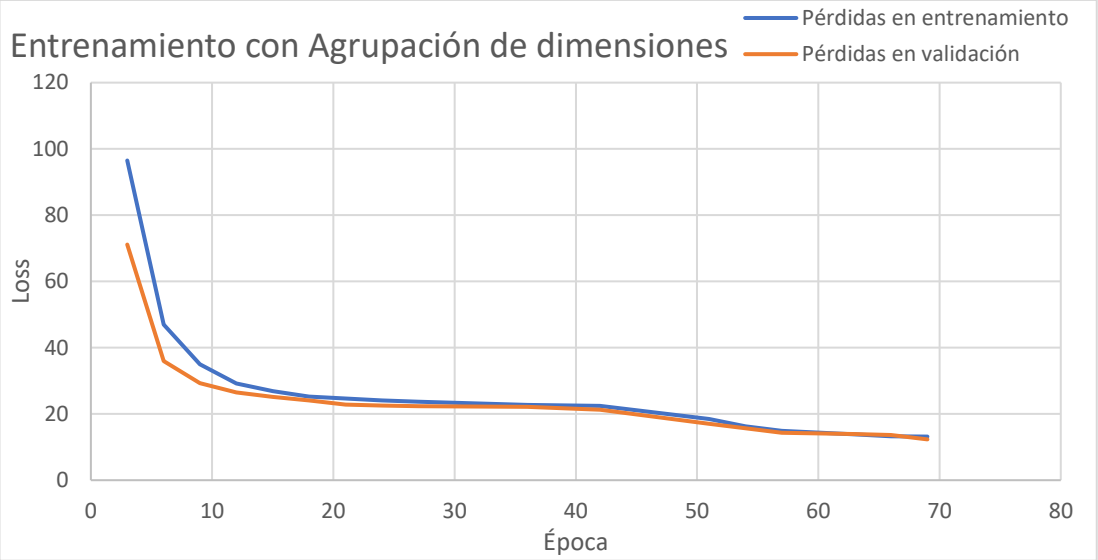

Figura 42 Representación de la función de pérdidas en el entrenamiento de la red con agrupación de dimensiones

Se observa que la evolución de las pérdidas es similar a la obtenida con los *anchors* por defecto. Sin embargo, en este caso el descenso del error cometido es menos rápido y la primera etapa dura 42 épocas y finaliza con unas pérdidas de entrenamiento de 22,45 y uno de validación de 21,27. La segunda etapa, que comienza en la época 50, dura hasta la 69, en la que alcanza un valor de 13,16 en las pérdidas de entrenamiento y 12,32 en el de validación. En base a las pérdidas obtenidas, el comportamiento de la red debería mejorar al de la prueba anterior, debido a que se ha reducido en 1,4 unidades.

De la misma forma que en el caso anterior, se ejecuta el *script* de evaluación y se obtienen los siguientes resultados:

```
-----------------------------------------------------
RESUMEN
-----------------------------------------------------

Verdadero Positivo: 23
Verdadero Negativo: 286
Falso Positivo: 37
Falso Negativo: 76

Precisión: 0.38333333333333336
Sensibilidad: 0.23232323232323232

IoU en los verdaderos positivos: 0.6017609410529509

IoU global: 0.09134328481128445

Duración: 333.02406215667725

FPS: 1.2581673446853634
```

Figura 43 Registro obtenido en la prueba de la red

Se observa que los valores obtenidos para Precisión y Sensibilidad se aproximan en gran medida a los obtenidos con los *anchors* originales. Sin embargo, se analizan de nuevo las falsas detecciones para observar posibles diferencias.

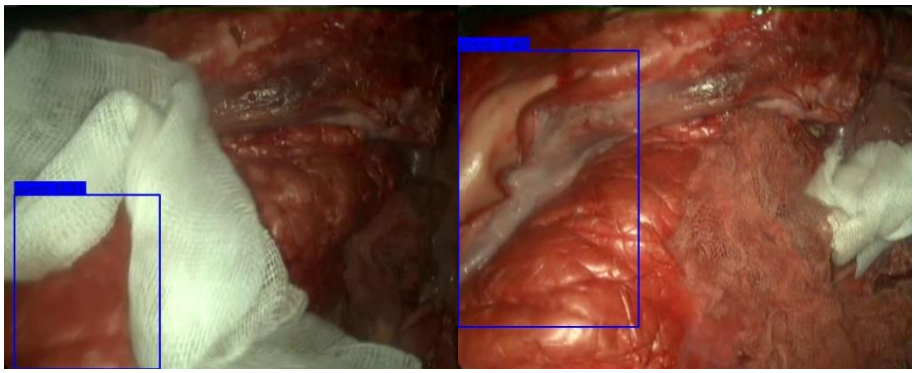

Figura 44 Falsas detecciones de la red

Se continúa apreciando el reconocimiento de la red de formas en "V" con contraste de color como pinzas, cuando en realidad se trata de gasas. Destaca un fallo concurrente en la detección, en la imagen de la derecha de la figura anterior se puede apreciar como una parte de grasa del órgano tiene una silueta que se puede asemejar a la de una pinza de cirugía laparoscópica. Esta similitud ha provocado que, al ser una parte del fondo que aparece en múltiples fotogramas, haya provocado al menos 15 falsas detecciones. La posible solución a estos errores es proporcionar más datos de entrenamiento con una mayor variedad para reconocer en base a más características. Se observa que el error de detección del instrumento cuando no están presentes los fórceps se ha eliminado casi completamente.

La sensibilidad es ligeramente superior, 23,2% frente al 22,7% del entrenamiento anterior, pero con la cantidad de datos utilizada no se trata de una diferencia reseñable. Sigue presentando problemas a la hora de detectar la pinza cuando hay ligeros movimientos o cuando es una situación relativamente nueva para la red.

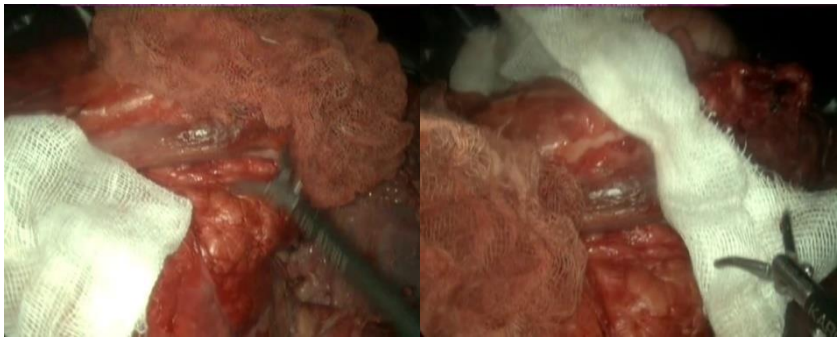

Figura 45 Objetos no detectados por la red

El IoU global obtenido tras este entrenamiento es menor que en el anterior. Esta diferencia se puede deber a que al personalizar los *anchor boxes*, los cuadros delimitadores de salida suelen tener menor tamaño, por lo que es más difícil que se produzca intersección entre las áreas si la detección no es precisa. Esta valoración también se basa en que el IoU en los objetos detectados como verdaderos positivos se ha duplicado respecto a la anterior red entrenada. En este caso es del 60,17%. Si se valoran las imágenes individualmente se puede apreciar que la detección se ajusta considerablemente mejor a la pinza que en el caso anterior:

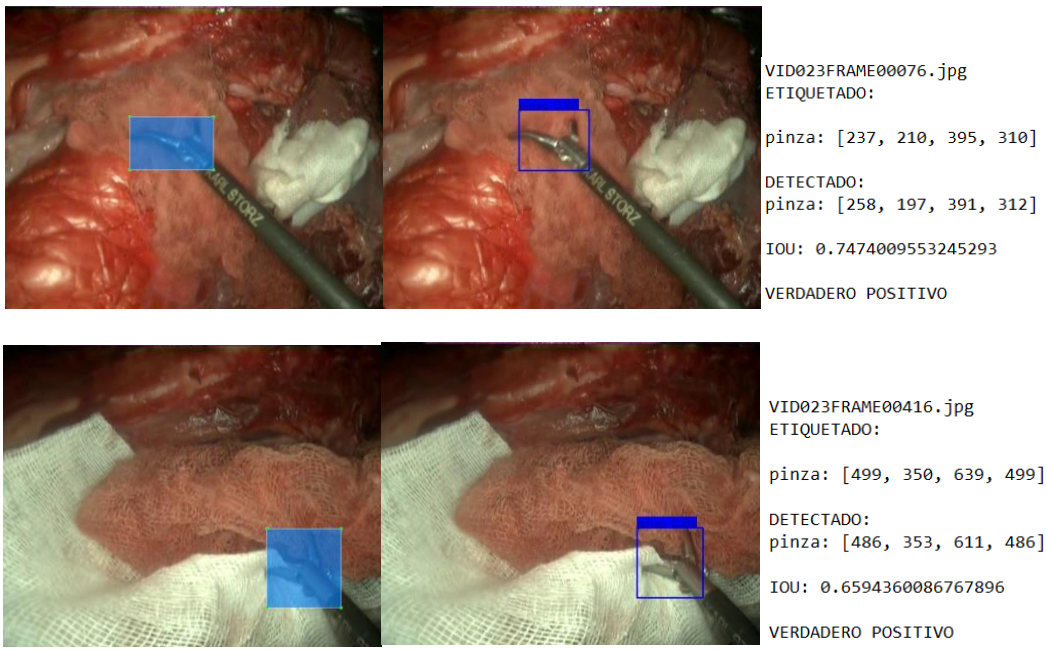

Figura 46 Detecciones de la red comparadas con etiquetado en el conjunto de datos

5.7.3. Entrenamiento de Tiny-YOLO

Al implementar la red que se decidió como óptima para el proyecto se han apreciado problemas de tamaño del conjunto de datos y tiempo de detección. La ampliación de la biblioteca de imágenes no es posible en este trabajo, por lo que se busca una solución al tiempo de inferencia de la red, para conseguir reducir las prestaciones mínimas para llevar a cabo detección en tiempo real. Con este fin, se implementa la versión reducida del modelo de la red YOLO utilizado, denominada Tiny-YOLO. Al tratarse de la misma versión de YOLO, las modificaciones necesarias para utilizar este modelo son mínimas. Únicamente hay que cambiar la propia definición del modelo, los pesos preentrenados de la red y los *anchor boxes*, debido a que utiliza 6, en lugar de los 9 de la versión completa. Para calcular los 6 *anchor boxes* se utiliza agrupación de dimensiones.

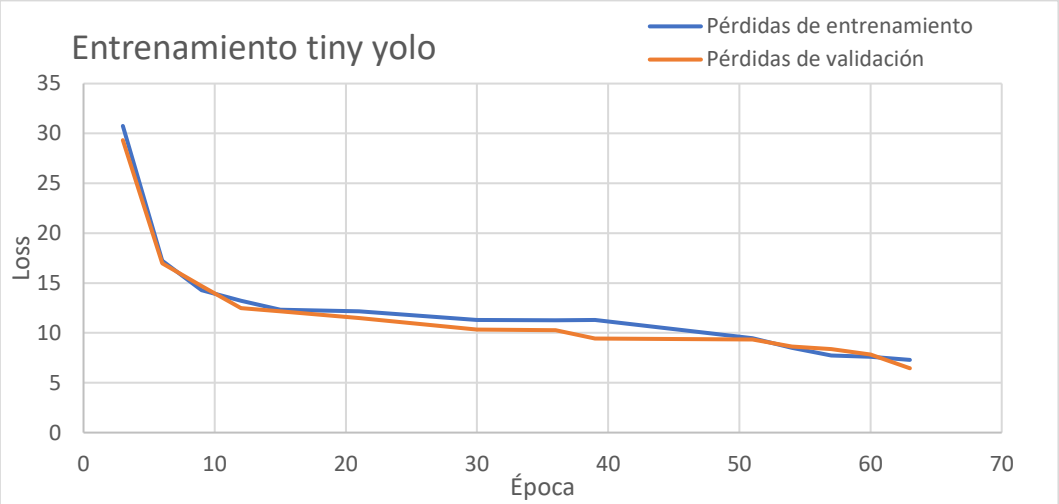

Figura 47 Representación de la evaluación de las pérdidas en el entrenamiento de la red Tiny-YOLO

En este caso se aprecia que las pérdidas son menores que en los casos anteriores desde el inicio del entrenamiento. Esta reducción no significa que la detección con este modelo sea mejor, si no que ha variado alguno de los parámetros que se utilizan en el cálculo de las pérdidas. Al igual que el resto de entrenamientos, termina la primera fase antes de alcanzar las 50 épocas, con 39 y con unas pérdidas de validación de 9.43 pasa a la segunda fase del entrenamiento. Las pérdidas de validación continúan reduciéndose hasta alcanzar el valor de 6,45 en la época 63, momento en el que se obtienen los pesos de la versión final de la red.

La detección también se puede ejecutar sin ser necesarias modificaciones, y tras ejecutarlo se obtienen los siguientes resultados:

```
-----------------------------------------------------
RESUMEN
-----------------------------------------------------

Verdadero Positivo: 15
Verdadero Negativo: 302
Falso Positivo: 16
Falso Negativo: 86

Precisión: 0.4838709677419355
Sensibilidad: 0.1485148514851485

IoU en los verdaderos positivos: 0.7093287940926476

IoU global: 0.10018678342456687

Duración: 51.96736407279968

FPS: 8.06275260398111
```

Figura 48 Registro obtenido en la prueba de la red

Se observa que el número de imágenes detectadas se reduce respecto a los modelos anteriores, disminuyendo la sensibilidad. Observando las imágenes no es posible establecer un criterio por el cual hay imágenes que ha dejado de detectar. En cambio, mantiene la precisión de los objetos que etiqueta. Se continúan observando los mismos errores que en el resto de entrenamientos de detección de contraste de colores con formas en “V”. Se observan datos de objetos con etiquetas demasiado grandes como para cumplir el IoU umbral e incluso se realizan detecciones que no se etiquetaron en el conjunto de datos debido a su escasa claridad.

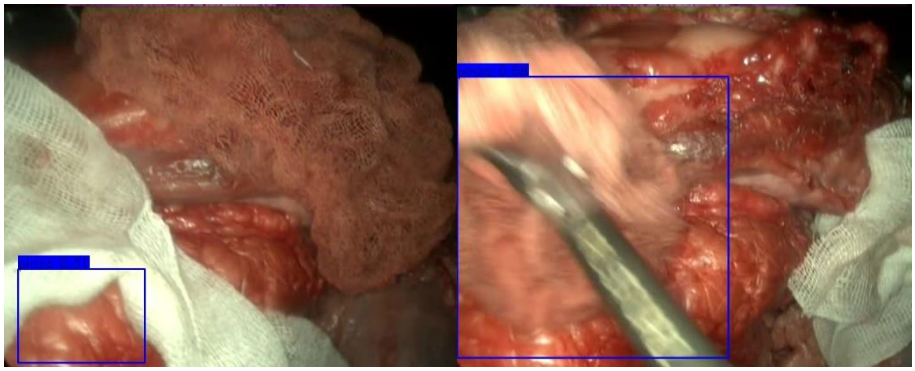

Figura 49 Falsas detecciones de la red

El IoU global se mantiene en un 10%, siendo ligeramente superior a la red que ya utilizaba agrupación de dimensiones con 9 *anchor boxes.* El IoU de los verdaderos positivos ha aumentado en un 10%, situándose en 70,93%, aunque hay que tener en cuenta que al tener una sensibilidad menor, este valor se corresponde a un número menor de objetos. Si se observan las detecciones realizadas, se aprecia un buen ajuste del cuadro delimitador a la pinza, excepto en los casos que ha etiquetado el instrumento completo.

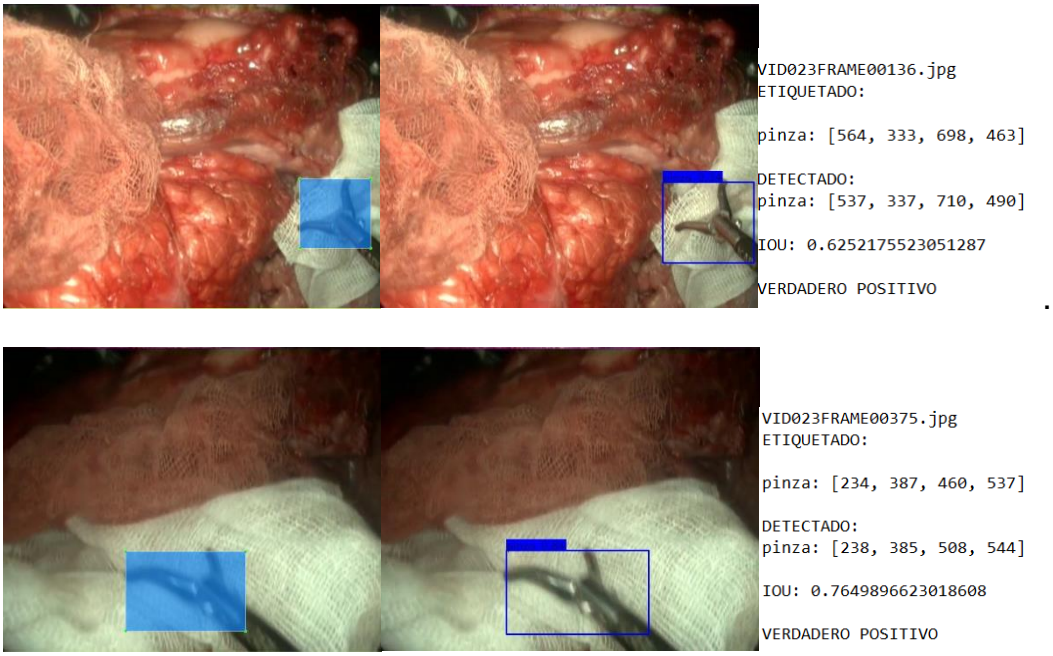

Figura 50 Detecciones de la red comparadas con etiquetado en el conjunto de datos

El factor diferencial que favorece el uso de esta red es la tasa de fotogramas por segundo obtenido por la misma al evaluar el conjunto de datos. En esta prueba se ha obtenido un rendimiento de 8,06 FPS, cuando con el modelo anterior no se superaba el valor de 1,34. Por lo tanto, utilizando este modelo, el coste asociado a los equipos necesarios para ejecutarlo en tiempo real va a ser menor.

6. Resultados finales

A partir de los resultados anteriores se puede extraer la conclusión de que la agrupación de dimensiones es clave para una buena detección de los objetos al utilizar un conjunto de datos tan específico como el de este trabajo. A la hora de elegir entre la red completa y la de tamaño reducido es importante valorar las necesidades del proyecto en el que se va a aplicar la herramienta.

Si se desea realizar un análisis de vídeos ya grabados, donde la velocidad de inferencia de la red no es determinante, es recomendable utilizar la red de mayor tamaño, debido a que se ha observado que, obteniendo el mismo valor de precisión, la sensibilidad es mayor, y se va a detectar la pinza en más instantes.

Si el objetivo es utilizar la herramienta en tiempo real, excepto que se disponga de una tarjeta gráfica con la suficiente potencia y memoria interna disponible, es recomendable utilizar la versión *"Tiny-YOLO"*, que detectará en menos ocasiones la pinza, pero con la misma tasa de predicciones correctas y ajustando incluso con más precisión los cuadros delimitadores.

Se ha observado en todas las redes entrenadas que el valor de precisión es mayor que el de sensibilidad. En múltiples casos se debe al desenfoque de la pinza en su movimiento, probablemente provocado en ocasiones por el desentrelazado de la imagen del laparoscopio. Esta característica hace que la herramienta pueda ser útil a la hora de indexar vídeos, aplicación en la cuál es fundamental que las predicciones sean correctas, pero no tiene tanta relevancia que se detecte la imagen en todos los fotogramas. En cambio, este valor de sensibilidad puede suponer un problema al implementar la herramienta para evaluar la destreza de los cirujanos, ya que no se puede analizar correctamente su precisión o sus trayectorias si no se dispone de todos los puntos por los que pasa la pinza.

6.1. Indexado de vídeos

Para comprobar la utilidad de la red a la hora de indexar vídeos de cirugía laparoscópica se ha desarrollado un script que carga un vídeo y analiza cada uno de sus fotogramas. Para registrar las detecciones abre dos ficheros de texto. En uno de ellos se registra cada etiqueta detectada, con su confianza y el fotograma en el que se encuentra. De esta forma se puede analizar la evolución temporal de la cirugía y distinguir las distintas etapas:

Objeto	Conf.	Fotograma
pinza	0.34	17
pinza	0.37	18
pinza	0.69	19
pinza	0.61	20
pinza	0.81	21
pinza	0.69	22
pinza	0.46	143
pinza	0.43	144
pinza	0.31	145

Figura 51 Registro de detecciones en un vídeo

En el otro archivo de texto, al finalizar de procesar todos los fotogramas del vídeo, se almacena un resumen de los objetos detectados, de esta forma se puede consultar fácilmente qué herramientas aparecen en cada vídeo y con que duración estimada. Actualmente solo reconoce pinzas porque son los datos de los que se dispone para el entrenamiento, pero no requeriría modificaciones si se ampliase el conjunto de datos:

```
Video cargado: PVID0023.mp4

-----------------------------------------------------
OBJETOS DETECTADOS
-----------------------------------------------------
Objeto    Número de detecciones    Número total de fotogramas del vídeo

pinza     630                      4175
```

Figura 52 Resumen de detección de material quirúrgico en un vídeo de cirugía

6.2. Imágenes de la detección

A continuación se muestran más imágenes de la detección realizada por la red YOLOv3 completa con agrupación de dimensiones, modelo que ha obtenido mejores resultados en el trabajo, incluyendo detecciones positivas y negativas:

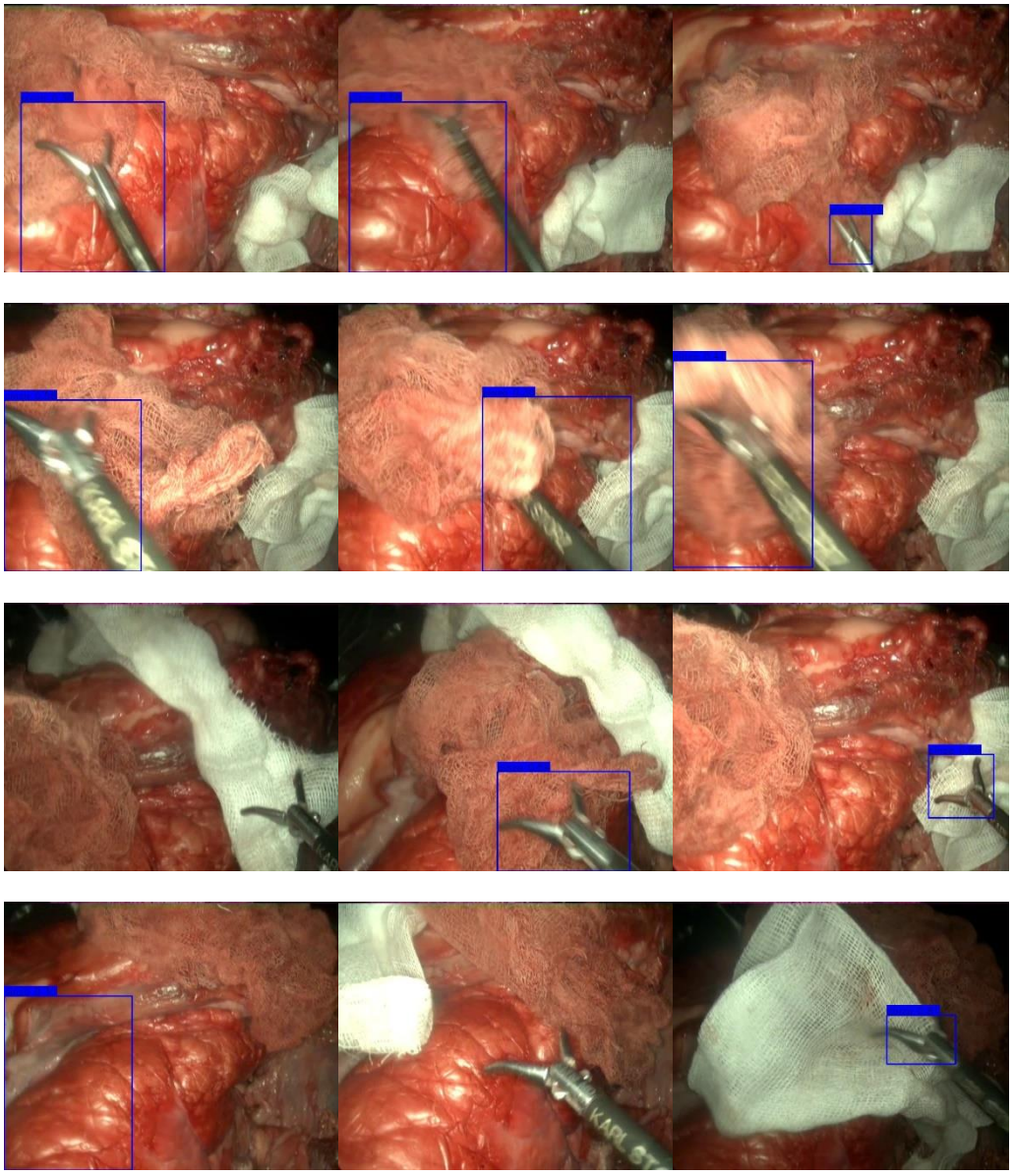

Figura 53 Ejemplos de funcionamiento de la red neuronal

7. Conclusiones y líneas futuras

La conclusión principal de este proyecto es que el aprendizaje profundo ofrece posibilidades para cubrir una gran variedad de necesidades que pueden surgir en cualquier proyecto y que la principal limitación en su implementación es la cantidad de datos disponible. Teniendo en cuenta que el trabajo parte desde la elaboración del conjunto de datos, los resultados se pueden considerar satisfactorios, ya que con un número limitado de objetos etiquetados se ha logrado, a través de herramientas como la agrupación de dimensiones, obtener un buen ajuste de los cuadros delimitadores en las detecciones correctas. Sin embargo, hay que tener en cuenta que para su uso en un ámbito tan crítico como es el médico, la red actual es un punto de partida, en el que hay que trabajar para mejorar la fiabilidad.

Al estar desarrollado íntegramente con herramientas de código abierto, se podrá incorporar y mejorar en otros proyectos sin ningún tipo de restricción.

La principal línea futura es el uso de esta herramienta como apoyo a otros proyectos, como puede ser la clasificación de cada una de las etapas de una operación en un vídeo ya registrado. Este indexado de las fases presentes en el vídeo permite búsquedas mucho más rápidas y eficientes.

Las posibilidades de uso de esta red aumentan si se logra llevar a cabo detección en tiempo real. La única opción para ello es la reducción del tiempo de inferencia hasta alcanzar un valor que permita obtener una tasa de fotogramas por segundo suficiente. Esta reducción posiblemente se pueda conseguir utilizando el hardware adecuado, no excesivamente costoso, aunque no se han podido realizar pruebas para certificarlo.

Detectando el instrumental en tiempo real puede ayudar a un robot autónomo a inferir el estado actual de la cirugía para determinar las siguientes acciones a ejecutar sin la necesidad de recibir órdenes del cirujano. Permitiría descargar al cirujano humano de labores automatizables, agilizando el proceso de la cirugía y evitando posibles errores humanos.

Otra mejora posible, al mismo tiempo necesaria para mejorar los resultados en el resto de aplicaciones es la ampliación del conjunto de datos. Actualmente se dispone de información de un número limitado de vídeos con relativamente poca variedad de instrumentos. Sin embargo, el uso de software libre y etiquetas XML de cada imagen que contienen la información del etiquetado, permiten que sea accesible tanto la modificación del conjunto de datos para etiquetar más elementos, como tejidos o gasas, como la adición de imágenes etiquetadas de otras fuentes.

Una posibilidad de mejora es sustituir la red YOLO por una U-Net, que en lugar de obtener un cuadro delimitador en el que se encuentre la pinza, segmente la imagen para localizar la zona en la que se encuentra. Esto ofrece posibilidades como, a partir del análisis de las formas obtenidas en la segmentación, obtener de forma más precisa los parámetros de ubicación y dimensiones de la misma. Puede ser útil en aplicaciones en las que no interese tanto reconocer de que instrumento se trata sino analizar características en concreto, por ejemplo, en la evaluación de cirujanos. Sin embargo, requiere de un amplio conjunto de datos con imágenes delimitadas cuidadosamente de forma manual. Además, la potencia computacional necesaria para llevar a cabo esta segmentación es significativamente superior a la requerida para detectar.

8. Bibliografía

[1] J. R. Romero Villar , D. Romero Vázquez y M. Navarro Maestre, Cirugía laparoscopia vs. Cirugía convencional, El Cid Editor, 2009.

[2] E. de la Fuente López, Á. Muñoz García, L. Santos del Blanco, J. C. Fraile Marinero y J. Pérez Turiel, «Automatic gauze tracking in laparoscopic surgery,» *Computer Methods and Programs in Biomedicine,* vol. 190, p. Artículo 105378, 2020.

[3] «Clínica Universidad de Navarra,» [En línea]. Available: https://www.cun.es/enfermedades-tratamientos/tratamientos/cirugia-laparoscopica#:~:text=La%20laparoscopia%20es%20la%20alternativa,orificios%20en%20la%20cavidad%20abdominal.. [Último acceso: 2022 04 10].

[4] A. García Ruiz, L. Gutiérrez Rodríguez y J. Cueto García, «Evolución histórica de la cirugía laparoscópica,» *Cirugía Endoscópica,* vol. 17, nº 2, pp. 93-106, 2016.

[5] R. M. B. M. Berger-KuhnkeA.B., «La biografía de Philipp Bozzini (1773-1809) un idealista de la endoscopia,» *Actas Urológicas España,* vol. 31, nº 5, p. 443, mayo 2007.

[6] M. P. Laguna, B. Lagerveld y J. d. l. Rosette, «Tácticas y trucos endourológicos en laparoscopia,» *Archivos Españoles de Urología,* vol. 58, nº 8, octubre 2005.

[7] «Laparoscopic MD,» [En línea]. Available: https://www.laparoscopic.md/es/surgery/instruments/trocar. [Último acceso: 4 Abril 2020].

[8] Endoscopy Ewald Bacher, «SONMEDICA,» [En línea]. Available: http://www.sonmedica.com/es/gastroenterologia/3413-accesorios-para-endoscopia-ewald-bacher.html. [Último acceso: 4 Abril 2022].

[9] J. Álvarez Fernández-Represa, J. de Diego Carmona, E. Ortiz Oshiro y J. Mayol Martínez, «Cirugía laparoscópica,» *Cirugía Española,* vol. 68, nº 4, pp. 304-308, Octubre 2000.

[10] A. Bosch Rué , T. Lozano Bagén y J. Casas Roma, Deep Learning: Principios y Fundamentos, Editorial UOC, 2020, pp. 45-89.

[11] R. Flórez López y J. M. Fernández Fernández, Las redes neuronales artificiales: fundamentos teóricos y aplicaciones prácticas, Netbiblo, 2008, pp. 16-37.

[12] J. D. Kelleher, Deep Learning, Cambridge, Massachusetts: MIT Press, 2019, pp. 65-88.

[13] S. Pramanick, «Geeksforgeeks,» [En línea]. Available: https://www.geeksforgeeks.org/history-of-python/. [Último acceso: 8 Mayo 2022].

[14] Python Software Foundation, «Python,» [En línea]. Available: https://www.python.org/about/. [Último acceso: 8 Mayo 2022].

[15] Google Research, «TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems,» 2015.

[16] Keras, «Why choose Keras?,» [En línea]. Available: https://keras.io/why_keras/. [Último acceso: 9 Mayo 2022].

[17] O. Ronneberger, P. Fischer y T. Brox, «U-Net: Convolutional Networks for Biomedical Image Segmentation,» 2015.

[18] H. H. e. al., «UNet 3+: A Full-Scale Connected UNet for Medical Image Segmentation,» de *2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020.

[19] J. Redmon , S. Divvala, R. Girshick y A. Farhadi, «You Only Look Once: Unified, Real-Time Object Detection,» 2015.

[20] J. Redmon y A. Farhadi, «YOLO9000: Better, Faster, Stronger,» 2016.

[21] J. Redmon y A. Farhadi, «YOLOv3: An incremental improvement».

[22] A. Bochkovskiy, C.-Y. Wang y H.-Y. Mark Liao, «YOLOv4: Optimal speed and Accuracy of Object Detection,» 2020.

[23] Ultralytics, «YOLOv5 Documentation,» [En línea]. Available: https://docs.ultralytics.com/. [Último acceso: Junio 2022].

[24] qqwweee, «GitHub,» [En línea]. Available: qqwweee.

[25] A. Sharma, «An Incremental Improvement with Darknet-53 and Multi-Scale Predictions (YOLOv3),» [En línea]. Available: https://pyimagesearch.com/2022/05/09/an-incremental-

improvement-with-darknet-53-and-multi-scale-predictions-yolov3/. [Último acceso: Junio 2022].

[26] «Papers With Code,» [En línea]. Available: https://paperswithcode.com/dataset. [Último acceso: Junio 2022].

[27] tzutalin, «Github,» [En línea]. Available: https://github.com/tzutalin/labelImg. [Último acceso: Abril 2022].

[28] D. P. Kingma y J. Ba, «Adam: A Method for Stochastic Optimization».

[29] L. Velasco, «Medium,» Abril26 2020. [En línea]. Available: https://velascoluis.medium.com/optimizadores-en-redes-neuronales-profundas-un-enfoque-pr%C3%A1ctico-819b39a3eb5. [Último acceso: Junio 2022].

[30] pjreddie, «YOLO: Real-Time Object Detection,» [En línea]. Available: https://pjreddie.com/darknet/yolo/. [Último acceso: Junio 2022].

[31] V. S. Subramanyam, «Medium,» [En línea]. Available: https://medium.com/analytics-vidhya/iou-intersection-over-union-705a39e7acef. [Último acceso: junio 2020].