UNIVERSIDAD DE VALLADOLID

FACULTAD DE MEDICINA

ESCUELA DE INGENIERÍAS INDUSTRIALES

TRABAJO DE FIN DE GRADO

GRADO EN INGENIERÍA BIOMÉDICA

Guiado mediante visión artificial de un asistente robotizado en operaciones de cirugía laparoscópica.

Autor/a:

D. Diego Benavides Cobos

Tutor/a:

D. Eusebio de la Fuente López

D. Juan Carlos Fraile Marinero

Valladolid, 13 de julio de 2022

TÍTULO:	**Guiado mediante visión artificial de un asistente robotizado en operaciones de cirugía laparoscópica.**
AUTOR/A:	**D. Diego Benavides Cobos**
TUTOR/A:	**D. Eusebio de la Fuente López**
TUTOR/A:	**D. Juan Carlos Fraile Marinero**
DEPARTAMENTO:	**Departamento de Ingeniería de Sistemas y automática**
TRIBUNAL	
PRESIDENTE:	**D. Roberto Hornero Sánchez**
SECRETARIO:	**D. Eusebio de la Fuente López**
VOCAL:	**D. Juan Carlos Fraile Marinero**
SUPLENTE 1:	**D. Javier Pérez Turiel**
SUPLENTE 2:	**D. Rogelio Mazadea Echevarria**
FECHA: 10/07/2023	
CALIFICACIÓN:	

Resumen: La cirugía mínimamente invasiva y las cirugías asistidas robóticamente son dos de las técnicas quirúrgicas más comunes en la actualidad. A pesar de sus ventajas en comparación con la cirugía tradicional, también presentan algunas dificultades que, normalmente, las hacen más difíciles que sus alternativas.

Uno de estos problemas es el control de la cámara. En la práctica clínica, es manejada por un ayudante y es necesario un tiempo para aprender a coordinarse con el cirujano correctamente. Esto también provoca muchos errores que suponen retrasos en el tiempo de recuperación y una disminución en la calidad de vida de los pacientes.

En este proyecto, proponemos una conducción automática de la cámara basada en un algoritmo capaz de rastrear las coordenadas de las herramientas de cirugía y enviarlas a una arquitectura de control del robot.

Palabras clave: Laparoscopia. Aprendizaje automático. Detección y localización de objetos

Abstract: Minimally invasive surgery and robotic assistive surgery are two one of most common surgical techniques nowadays. Despite of its advantages in comparison with traditional surgery, it has also some difficulties that, normally, make them harder than its alternatives.

One of these problems is the control of the camera. In clinical practice, it is operated by an assistant, and it takes time to learn how to coordinate it correctly. It also causes many errors that imply delays in recovery and decreases in the quality of life of patients.

In this project, we propose an automatic driving of the camera based on an algorithm able to track the surgery tools coordinates and send them to a robot control architecture.

Keywords: Laparoscopy. Deep Learning. Object detection and tracking

Agradecimientos:

En primer lugar, quiero mencionar que, personalmente, he enfocado este trabajo como la puesta en práctica de las destrezas adquiridas en estos 4 años de esta bonita etapa. Con ello no me refiero solamente a los conocimientos, que son inabarcables en un proyecto como este, sino a otro tipo de habilidades, como la manera de afrontar los desafíos, la rigurosidad a la hora de buscar información o la forma de solucionar los contratiempos que aparecen durante el desarrollo de este trabajo.

Es por ello, que quiero dar gracias a todos los que han hecho posible, no solamente este proyecto, sino este paso por el grado tan satisfactorio. En primer lugar, quiero agradecérselo a mi familia, por siempre mostrar una confianza plena en mis habilidades y en mi talento, y por darme todas y cada una de las oportunidades que he considerado necesarias.

En segundo lugar, me gustaría darles las gracias a mis amigos de siempre, a los que nunca han faltado para empujar cuando nadie más lo hacía, y que me han sacado de la burbuja que es a veces ser un apasionado por la tecnología y la salud.

También quiero agradecerle a Roberto Hornero, Jesús Poza y cada una de las personas que hayan estado involucradas en la creación de este grado. El resultado ha sido increíble y creo que hay poco margen de mejora. Se nota que se ha creado con dedicación y cariño, y eso es algo que tiene mucho mérito.

Por último, quiero dar las gracias a los centros que nos han acogido durante estos 4 años para formarnos con sus mejores profesionales, la Facultad de Medicina y la Escuela de Ingenierías Industriales. En concreto, quiero agradecer al grupo de Robótica Médica de ITAP su acogida y facilidades que me han dado para integrarme y sentirme una pieza importante en el día a día. Y por supuesto, a mi delegación de estudiantes de la Facultad de Medicina todo lo que me han dado en estos años. Espero haber devuelto un pequeño porcentaje de lo que me habéis dado porque os habéis convertido en mi familia.

Índice de contenidos:

Índice de figuras:

Parte I. Introducción y conocimientos previos

Capítulo 1. Introducción. Motivación. Hipótesis y objetivos.

1.1. Introducción

En la actualidad la cirugía mínimamente invasiva y la cirugía robótica son la técnica de elección para muchas cirugías de especialidades muy diversas debido a sus múltiples beneficios respecto de la cirugía abierta más tradicional. En cambio, aún cuentan con algunas limitaciones que provocan en este tipo de intervenciones algunas complicaciones que finalmente repercuten en la recuperación del paciente. Algunas de estas limitaciones están relacionadas con el control de la cámara por un asistente, en el caso de la cirugía mínimamente invasiva, o por el propio cirujano en caso de la cirugía robótica. El objetivo de este proyecto es desarrollar un algoritmo capaz de implementarse en un control de un robot de laparoscopia, de forma que sea capaz de mover la cámara o endoscopio de forma automática.

La cirugía mínimamente invasiva, también conocida como cirugía laparoscópica, representa uno de los avances más significativos en la historia de la cirugía, junto con la asepsia quirúrgica y las técnicas de anestesia más seguras. Estas técnicas se basan en realizar intervenciones utilizando instrumentos alargados que se introducen a través de pequeñas incisiones en las zonas cercanas al área de interés [1]. Hasta la aparición de este tipo de técnicas, estas intervenciones se realizaban mediante una incisión cuyo tamaño depende del espacio de trabajo necesario para llevarlas a cabo, las cuales están sujetas a algunos inconvenientes como los sangrados, el trauma quirúrgico o las infecciones.

Para comprender correctamente la técnica laparoscópica hay que situarla en un contexto histórico adecuado. Las ideas que conforman el marco de la cirugía laparoscópica se informaron inicialmente hace más de un siglo. Sin embargo, su introducción en el campo de la cirugía general ha sido un desarrollo relativamente reciente. La palabra "laparoscopia" proviene del griego "laparo", que significa "costado", y "scope", que significa "instrumento de observación". Es un término que se utiliza para describir el procedimiento en el que se examinan los contenidos peritoneales con un endoscopio. Un endoscopio se define como "un instrumento utilizado para examinar el interior de una víscera hueca". La cirugía laparoscópica debe gran parte de su historia al desarrollo de la técnica endoscópica, que surgió como respuesta a la curiosidad inherente de las personas y nuestro deseo de inspeccionar las cavidades ocultas del cuerpo humano.

A lo largo de los siglos, hubo avances significativos en la endoscopia. En 1805, el médico alemán Phillip Bozzini visualizó con éxito la uretra humana utilizando una cámara de luz iluminada por una vela de cera. En la década de 1850, Desormeaux diseñó un endoscopio de tubo abierto que permitía examinar la uretra y la vejiga con una iluminación continua. En 1868, Kussmaul utilizó la técnica de endoscopia para explorar el esófago. En 1877, Nitze creó el primer cistoscopio moderno utilizando un alambre de platino calentado con corriente eléctrica.

En 1901, el cirujano alemán George Kelling intentó realizar laparoscopias en perros vivos, insuflando oxígeno en el abdomen y utilizando un cistoscopio para examinar los contenidos abdominales. Hans Christian Jakobaeus fue el responsable de popularizar la técnica en humanos y describió numerosas patologías a través de la laparoscopia. Durante las décadas siguientes, se realizaron avances significativos en la técnica, instrumentación y aplicaciones clínicas de la laparoscopia.

En la década de 1980, con el desarrollo de procesadores de video que permitían la proyección de imágenes en pantallas de televisión, la cirugía laparoscópica se integró realmente en la disciplina de la cirugía general. En 1987, se realizó la primera colecistectomía laparoscópica en un paciente humano, llevada a cabo por el cirujano francés Mouret [2], [3].

La aparición de las técnicas mínimamente invasivas supuso un cambio radical, no solamente sobre las intervenciones en sí mismas, sino sobre la forma de interpretar y afrontar la cirugía en ese momento, aceptando que la respuesta del cuerpo al estrés quirúrgico, al dolor o a los distintos daños traumáticos a los que se expone una persona en una operación es una respuesta fisiológica normal, y no está provocada por una patología subyacente. Este tipo de técnicas son uno de los resultados más importantes de este cambio de paradigma en las intervenciones quirúrgicas, ya que minimizar el daño quirúrgico, y por tanto, la respuesta que genera nuestro cuerpo ante él. La reducción del trauma quirúrgico pasó a estar en el centro de la preocupación de los cirujanos y, por ello, la cirugía mínimamente invasiva tomó un papel fundamental en esta nueva filosofía [2].

Las tres ventajas principales de la cirugía laparoscópica radican fundamentalmente en la reducción del daño quirúrgico. El hecho de minimizar el tamaño de las heridas o incisiones necesarias para llevar a cabo la cirugía está directamente relacionado con el tiempo de recuperación de los pacientes, así como los dolores que sufren después de la intervención. La cirugía laparoscópica permite reducir los dolores postoperatorios, las complicaciones como infecciones y la morbilidad postoperatoria, reduciendo el riesgo de sufrir patologías relacionadas con la inactividad prolongada como pueden ser las trombosis venosas. La reducción del daño quirúrgico, junto con el de las complicaciones nos lleva a la segunda gran ventaja de estas técnicas, la reducción de la estancia hospitalaria y, por tanto, una mejora de la situación del paciente, que puede recuperarse en su propio domicilio [4], y un ahorro al Sistema Nacional de Salud (SNS), especialmente importante en países como España, con un modelo que supone un gasto de un 10,9% del Producto Interior Bruto (PIB), de los cuales un 8,0% es financiado con recursos públicos y un 2,9% con recursos privados [5]. Por último, la tercera ventaja de estas

técnicas radica en un componente estético, ya que supone una mejora importante en cuanto a la recuperación del tejido, formando cicatrices más pequeñas y menos llamativas que las provenientes de una cirugía convencional, lo que supone una mejora de la calidad de vida de los pacientes tanto a nivel físico como psicológico [6].

Por otro lado, la cirugía mínimamente invasiva tiene varias desventajas en comparación con la cirugía abierta. Algunas de estas desventajas incluyen una curva de aprendizaje más larga, falta de control preciso de la posición y dificultad en el control del sangrado y localización de estructuras profundas, ya que se pierde la percepción de profundidad que sí se tiene en cirugías abiertas [7], [8]. Además, aunque se reducen en gran medida los riesgos de complicaciones con la anestesia, el sangrado y la infección, estos no desaparecen en su totalidad. [9].

Esta nueva evolución hacia la cirugía laparoscópica se ha podido llevar a cabo gracias a la minimización de las cámaras y a la aparición de nuevos dispositivos de adquisición de imágenes aptos para introducirse por estas pequeñas incisiones y proporcionar imágenes capaces de eliminar la necesidad de realizar una cirugía abierta. Otro de los desarrollos imprescindibles para la evolución de esta tecnología son los sistemas de insuflación de distintos gases que permiten expandir las distintas cavidades corporales para crear una zona de trabajo donde las distintas herramientas tienen el rango de movimiento necesario para realizar los distintos procedimientos que conforman la cirugía completa [6].

En la línea de los avances en cirugía laparoscópica, cabe resaltar el importante papel que ha tomado la cirugía robótica en los últimos años. La aparición y la reducción del coste de equipos como el *Da Vinci*, de la empresa Intuitive Surgical®, junto con las ventajas que aporta respecto de la cirugía laparoscópica tradicional han llevado a que el uso de estos equipos sean prácticamente un procedimiento habitual en muchos hospitales españoles. Este robot en concreto, uno de los más utilizados hoy en día en la práctica clínica, ha dado solución a muchas de las limitaciones que tiene la cirugía laparoscópica, ofreciendo una visión en 3D inmersiva y un mayor rango de movimientos debido a sus múltiples articulaciones a lo largo de la herramienta, que les dotan de grados de libertad mucho mayores a las herramientas de cirugía laparoscópica tradicionales [10].

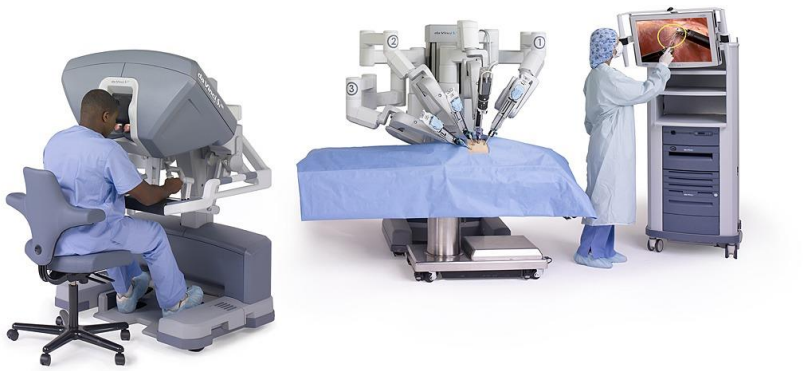


Figura 1. Partes que componen al robot quirúrgico DaVinci. De derecha a izquierda: consola, robot y torre de visión.

Además del DaVinci, se han desarrollado otros equipos durante estas últimas décadas que dan solución a muchos de los problemas que van apareciendo en el día a día de la práctica clínica. El robot quirúrgico Senhance® se trata de un equipo compuesto por 4 brazos robóticos independientes, conectados todos ellos a un nodo único, al que mandan constantemente información sobre su posición, orientación y modo de funcionamiento. Este equipo tiene algunas ventajas sobre el DaVinci®, entre las que se encuentran su facilidad de recolocar al paciente o pasar a realizar una cirugía laparoscópica convencional, ya que hace uso de trocares generales de laparoscopia. Otra gran ventaja es su coste por uso, siendo de unos 1200$ en el caso del DaVinci Xi, y 200-500 en el caso del Senhance [11]. Por último, este equipo se diferencia de los anteriores en dos aspectos más. El primero se trata de un control de respuesta háptica, que se transmite hasta los mandos de la consola dependiendo de las características del tejido que perciban las herramientas. El segundo es un sistema de control semiautomático para mover la cámara, que se basa en un *eye-tracking* del cirujano previa calibración que permite mover el endoscopio a la zona donde el cirujano se está fijando [12].

El último de este tipo de equipos en ver la luz ha sido Hugo RAS, de la famosa empresa Medtronic®. Este robot cuenta con 4 brazos independientes y una pantalla capaz de mostrar las imágenes del endoscopio en 3D. También cuenta con la posibilidad de adaptar el escalado de movimientos y el aumento de rotación de las herramientas respecto de los *joysticks*. Las herramientas, al igual que en el DaVinci cuentan con 7 grados de libertad [13]. Entre sus ventajas respecto al DaVinci, el robot más utilizado actualmente, se encuentra el hecho de contar con una pantalla y unas gafas que facilitan tener una visión 3D sin necesidad de introducir la cabeza en ningún tipo de consola, lo que facilita la comunicación entre el equipo quirúrgico. Otra de sus ventajas es que cuenta con 4 brazos independientes, lo que aumenta su versatilidad y facilita la colocación del paciente y del robot en la sala, comúnmente llamado docking [14]. No obstante, la gran ventaja de este equipo respecto a los dispositivos que se utilizan actualmente, y la razón por la que puede posicionarse como una gran alternativa a estos, es su reducción en los costes. Los informes de las primeras cirugías llevadas a cabo por estos equipos sitúan el ahorro entorno al 30-40% respecto a las intervenciones anteriores, por lo que este equipo puede hacer que estas técnicas sean más accesibles y, por tanto, mucho más utilizadas en todo el mundo [15], [16] .

Figura 2. Partes que componen el Hugo RAS. De derecha a izquierda: torre, consola y pantalla, brazos robóticos.

La cirugía robótica posee diversas ventajas sobre la cirugía laparoscópica. De forma objetiva ya se han demostrado que los resultados obtenidos con estas nuevas técnicas son mejores que los obtenidos con cirugías laparoscópicas tradicionales [17]. En cambio, lo que realmente decide a un profesional a usar una técnica u otra teniendo ambas disponibles, son las ventajas que les aportan de forma subjetiva. Se ha podido comprobar que, desde el punto de vista del cirujano, la cirugía robótica mejora de forma notable la destreza, calidad de imagen, la percepción de la profundidad, la comodidad y la fatiga ocular respecto a las técnicas laparoscópicas. También mejora, aunque en menor medida, el rango, la precisión y la velocidad de los movimientos. Su gran desventaja radica en la sensibilidad táctil a la hora de interaccionar con los tejidos, ya que se pierde en gran medida al no trabajar directamente sobre los instrumentos que están en contacto con el tejido [18].

Por otro lado, aunque los costes de las cirugías asistidas mediante robots son bastante más altos comparándolos con la cirugía laparoscópica (véase Tabla 1), su uso no ha parado de aumentar en los hospitales de todo el mundo, lo que se espera que reduzca sus precios. Realmente no se ha demostrado que la cirugía robótica aporte resultados significativamente mejores que la cirugía laparoscópica comparado con su mayor coste, lo que si se ha hecho con la cirugía abierta tradicional. Por ello, se ha planteado que las ventajas de estas técnicas pueden radicar en otros enfoques de los resultados. En algunas cirugías, la cirugía mínimamente invasiva solo se consideró como una alternativa realmente factible con la aparición de la cirugía robótica. Además, la mayoría de los análisis coste-beneficio que se realizan se hacen a corto plazo, lo cual puede ocultar algunos beneficios que aporta la cirugía robótica. Se ha visto que en las nefrectomías parciales la cirugía robótica tiene un porcentaje de éxito un 52% superior a la cirugía laparoscópica. Realizar este tipo de cirugías en lugar de las nefrectomías totales aporta a los pacientes un año más de supervivencia y una reducción de las posibilidades de sufrir un fracaso renal. Por último, estos equipos pueden permitir llevar a cabo un mayor número de intervenciones, no porque estas sean más rápidas, sino porque reducen las estancias hospitalarias, un factor limitante común a la hora de programar estas intervenciones, lo que hacen que puedan ser más rentables [19].

Tabla 1. Coste de diversos procedimientos, laparoscopia vs RAS [19].

Procedimiento	***Laparoscopia (US$)***	***RAS (US$)***	***% Cambio***
Colecistectomía	9.660	10.980	13,7
Hernia incisional	10.750	13.440	25,0
Hemicolectomía derecha	12.540	15.030	19,9
Hemicolectomía izquierda	14.140	18.110	28,1
APR	17.730	20.320	14,6
Histerectomía	9,340	9.940	6,4

En cambio, aunque estos equipos son llamados robots quirúrgicos, hoy en día los aparatos más utilizados no están dotados de ningún tipo de autonomía más allá del tratamiento de imagen y la corrección de algunos temblores. Actualmente estos equipos siguen siendo únicamente máquinas teleoperadas por los cirujanos que han sido entrenados para utilizarlas. Debido a estos factores, las cirugías pueden derivar en algunas complicaciones debido a errores humanos, ya que el cirujano debe realizar movimientos precisos para el correcto control del robot. También se pueden cometer errores debido a la fatiga de los profesionales durante la intervención. Además, el cirujano debe controlar tanto las herramientas que se utilizan como la cámara que le otorga la visión en el interior del cuerpo, por lo que en muchas ocasiones la operación debe ser pausada para recolocar la cámara y poder tener el campo de visión adecuado. Estos sucesos pueden derivar en un aumento en la duración de la cirugía, relacionado directamente con el riesgo de sufrir complicaciones, así como en un aumento de los errores que comenten los cirujanos al realizar estas intervenciones debido, nuevamente, a la fatiga que generan [20], [21].

Un aumento de la autonomía en estos equipos podría conllevar grandes beneficios en las intervenciones asistidas por robots. Sistemas robotizados con mayor autonomía serían capaces de realizar tareas repetitivas con una mayor precisión que los humanos, evitando muchos de sus errores, sobre todo los causados por la fatiga. Además, también reducirían la curva de aprendizaje en el entrenamiento de estos equipos, que actualmente es un gran limitante a la hora de encontrar personal debidamente cualificado para su uso. Por último, esta autonomía podría llegar a ser necesaria si queremos llevar a cabo cirugías a grandes distancias, ya que la latencia de datos provocada por el tiempo que tarda en viajar la señal de un punto a otro del planeta lleva a un aumento de los errores durante la intervención [1].

Durante muchos años, el aumento de autonomía de estos equipos se ha basado en la planificación de trayectorias de las herramientas a partir de pruebas de imagen, o la realización de maniobras fundamentalmente mecánicas dependientes de la anatomía del paciente. Actualmente los avances en el aumento de autonomía por parte de los sistemas robotizados se dirigen a realizar tareas automáticas y minimizar errores mecánicos de los profesionales sanitarios con un nivel de adaptación al medio mucho mayor. La gran evolución que se está llevando a cabo en campos de *machine learning* o visión artificial, junto con el aumento en la capacidad de cómputo de los equipos está derivando en equipos que pueden dar una respuesta automática a los movimientos de los profesionales, a la anatomía del paciente y en algunos casos, llevar a cabo algunas tomas de decisiones en base a los datos que van recolectando. Algunos ejemplos de las direcciones en las que se está enfocando esta evolución en la autonomía de los equipos pueden ser la preparación de los huesos en reconstrucciones articulares o la manipulación o control automático de la cámara [22].

En concreto, la manipulación automática de la cámara puede ser una solución realmente interesante a algunos de los problemas a los que se enfrentan los cirujanos en estos tipos de intervenciones. Como ya se ha explicado, en las cirugías mínimamente invasivas se pierde la visión directa del campo quirúrgico, y esta debe obtenerse mediante una cámara, controlada normalmente por un asistente. En este trabajo, la cooperación entre el cirujano y el asistente

es crucial, ya que el primero debe dar indicaciones concretas de cómo mover la cámara y cuál es la zona de interés, y el segundo debe comprenderlas a la perfección y mover la cámara de forma que enfoque el lugar deseado. Como es lógico, esto trae consigo algunos inconvenientes, ya que en muchas ocasiones se cometen errores en la coordinación de estas dos partes que llevan a complicaciones quirúrgicas. Por ello, la experiencia de estos dos profesionales trabajando juntos toma un papel fundamental en este tipo de intervenciones, lo que en muchas ocasiones es realmente complicado, ya que no siempre es posible que un cirujano trabaje con el mismo asistente durante un tiempo prolongado, y alarga aún más la curva de aprendizaje en las cirugías mínimamente invasivas [23].

Dar solución a este problema se ha convertido en las últimas décadas en una línea de investigación muy popular. En los últimos años se han propuesto multitud de soluciones en busca de un sistema de control de la cámara lo más automático posible. Los fundamentos técnicos de estos modelos son muy variados, y van desde métodos de segmentación por el color de la herramienta o el uso de marcadores en las herramientas para poder localizarlos en las imágenes, hasta modelos de aprendizaje automático y algoritmos de semejanza con modelos sintéticos 3D. En el marco teórico del proyecto se analizará el estado del arte y se desarrollarán más detalladamente este tipo de técnicas.

1.2. Motivación del proyecto

Como se ha explicado en el apartado anterior, en las cirugías laparoscópicas la cámara es manejada por un asistente del cirujano. Este proceso está condicionado por la coordinación que tengan estos dos profesionales entre ellos a la hora de mover las herramientas, la cámara, y al dar las órdenes al otro de forma clara e inequívoca. Para que esto sea posible, es necesario que las dos partes tengan experiencia trabajando juntos, lo que no siempre es posible. Además, el movimiento de la cámara está sujeto a temblores o fallos humanos, en muchas ocasiones causados por la fatiga que genera una cirugía laparoscópica.

Para solucionar gran parte de estos contratiempos se empezó a desarrollar la cirugía robótica. Este avance eliminó los temblores y la coordinación entre profesionales, ya que lo habitual es que el propio cirujano controle la cámara. En cambio, el control de todas las herramientas por el cirujano lleva a errores debido, por ejemplo, a las pausas que hay que realizar para recolocar el endoscopio o a la fatiga por el aumento del tiempo que lleva realizar estas técnicas. La solución a estos limitantes puede radicar en aumentar la autonomía de estos equipos y conseguir que alguna de las partes del robot se mueva de forma automática según los intereses del cirujano. Más concretamente, el movimiento del endoscopio automáticamente, de forma que enfoque directamente al punto de interés sin necesidad de telemanipularlo, puede aportar grandes ventajas al desarrollo de este tipo de cirugías, corrigiendo algunos de los limitantes más significativos.

En cambio, como se ha mencionado anteriormente, en la actualidad los sistemas robotizados que intervienen en las cirugías laparoscópicas apenas poseen autonomía suficiente para llevar a cabo estas acciones, e incluso para llamarlos robots. La definición actual de robot según ISO (*Internacional Organization for Standardization*) es la siguiente [24]:

> *"Actuated mechanism programmable in two or more axes with a degree of autonomy, moving within its environment, to perform intended tasks."*

Según las características que consideran necesarias para que un dispositivo sea considerado robot, prácticamente ninguno de los que se utilizan actualmente en cirugía laparoscópica podría entrar dentro de esta clasificación.

Es por ello por lo que en este proyecto se busca desarrollar un algoritmo capaz de aportar la información suficiente a un robot para que sea capaz de enfocar el endoscopio de forma constante hacia el punto de interés. Para ello se utilizará un modelo de *Deep Leaning* basado en redes neuronales convolucionales que nos aporte información sobre la posición de los extremos distales de las herramientas, encargadas de interaccionar directamente con los tejidos.

El objetivo es que este modelo pueda ser implementado en robots de cirugía laparoscópica junto con otras arquitecturas de control. En concreto, en el Grupo de Robótica médica de ITAP (Instituto de Tecnologías de la Producción) se ha desarrollado una arquitectura de

control para estos equipos basado en la fuerza que ejerce sobre las paredes, lo que otorga la ventaja sobre los métodos tradicionales de eliminar la necesidad de saber la posición precisa del RCM (*Remote Center of Motion*). Este factor le puede proporcionar a los dispositivos una versatilidad mucho mayor que la que tienen, ya que el algoritmo puede adaptarse a movimientos del RCM sin necesidad de cambiar sus coordenadas. Este control, junto con el modelo que se busca obtener, puede conseguir que una cámara pueda reaccionar de forma autónoma a movimientos del paciente, manteniendo a las herramientas continuamente en la imagen.

El enfoque del proyecto se debe fundamentalmente a dos factores. El primero es la variabilidad de situaciones que podemos encontrarnos en las múltiples cirugías, dificultando en gran medida su abordaje mediante el modelado cerrado del problema y un algoritmo posterior que lo solucione. En este sentido, el avance del Deep Learning y la visión artificial, junto con el aumento de la carga computacional que soportan nuestros procesadores actuales nos brinda una oportunidad nunca vista para desarrollar este tipo de algoritmos, capaces de dar soluciones robustas a problemas de este tipo. Además, estos algoritmos nos permiten localizar las herramientas sin necesidad de modificar las herramientas con marcadores, lo que sería un problema a la hora de validar nuevamente los instrumentos debido a la introducción de objetos distintos en el interior del cuerpo de los pacientes. El segundo enfoque es puramente académico, en un campo que día a día se supera y se transforma veo especialmente interesante desarrollar habilidades en el campo del *Deep learning,* en este caso mediante la creación de un modelo de red convolucional y todos los archivos complementarios para que esta realice su función de la mejor forma posible.

1.3. Hipótesis y objetivos

En este trabajo se parte de la base de que en cirugía laparoscópica y cirugía robótica existe un problema común que hoy en día aún no ha solucionado, el control de la cámara. Aunque el problema en estos dos casos no tiene el mismo fundamento, una solución común para ambos facilitaría enormemente la realización de estas intervenciones.

Para ello, un control automático de la cámara, sin necesidad de un asistente o de que el cirujano tenga que hacerlo él mismo, podría estar cerca de ser la solución ideal a este paradigma. Para ello, el desarrollo en visión artificial nos da herramientas para poder obtener la información necesaria a partir del endoscopio y poder elaborar una arquitectura de control para un equipo robotizado. El fundamento de este trabajo, por tanto, consiste en la capacidad de la visión artificial de obtener información de las herramientas a partir de imágenes, de forma semejante a como lo haría una persona humana.

Por tanto, el objetivo general de este proyecto es conseguir un modelo capaz de informar sobre la posición de una herramienta laparoscópica en tiempo real, de forma que pueda ser integrado en el control de un robot quirúrgico con la finalidad de asistir al cirujano en esta tarea. Los objetivos específicos de este proyecto serían los siguientes:

- **Objetivo específico 1:** Recopilar un conjunto de datos lo suficientemente amplio y robusto, capaz de describir el problema y entrenar una red con resultados óptimos. Para ello se hará una búsqueda de bases de datos públicas y se mezclarán con datos adquiridos en el laboratorio

- **Objetivo específico 2:** Encontrar una arquitectura de red neuronal que nos aporte resultados suficientemente precisos para poder mantener el campo de visión enfocado en un máximo de dos herramientas.

- **Objetivo específico 3:** Programar la red neuronal escogida, junto con las distintas funciones necesarias para adaptarlo al conjunto de datos escogido, las funciones de pérdidas y métricas adecuadas para asegurar la precisión del modelo.

- **Objetivo específico 4:** Entrenar a la red para que sea capaz de reconocer herramientas quirúrgicas y comprobar su funcionamiento mediante el desarrollo de distintas métricas.

- **Objetivo específico 5:** Ajustar los distintos parámetros de inferencia mediante técnicas de filtrado o generación de trayectorias para eliminar en la medida de lo posible predicciones que no se ajusten a la posición de la herramienta real.

- **Objetivo específico 6:** Detección en tiempo real. Elaboración de un programa capaz de tomar imágenes y realizar predicciones en tiempo real. Este aspecto es fundamental debido al contexto en el que se desarrolla el proyecto. El robot debe moverse a la vez que las herramientas para aportar continuamente una visión adecuada del espacio de trabajo.

Capítulo 2. Conocimientos previos.

Localizar las herramientas utilizadas en cirugías laparoscópicas y robóticas ha sido un desafío que se ha intentado abordar desde numerosos campos con fundamentos muy distintos entre sí. En este capítulo se realizará una breve descripción de algunos de estos enfoques, junto con sus ventajas y limitaciones, con la finalidad de situar de una forma más concisa cuál es el entorno en el que se va a desarrollar el proyecto.

En primer lugar, una forma muy común de localizar objetos en una imagen es mediante marcadores. Esta técnica ha sido usada durante décadas en industrias como el cine o los videojuegos para poder localizar las coordenadas de distintas estructuras corporales y así poder definir la pose en la que se encuentra un individuo. Esta metodología se basa en usar pequeños marcadores pegados al cuerpo y dos o más cámaras, junto con un sistema de detección de los marcadores. Así, teniendo la posición de los marcadores en dos dimensiones de las dos cámaras, se puede calcular de forma trigonométrica la posición en 3 dimensiones de cada uno de ellos. Para una simplificación de dos cámaras frontoparalelas, la estimación de profundidad se obtendría a partir de la siguiente fórmula y el planteamiento que se muestra en la figura 3:

$$\frac{Z-f}{Z} = \frac{b}{b+\left(\frac{w}{2}-L_r\right)+\left(R_r-\frac{w}{2}\right)} \tag{1}$$

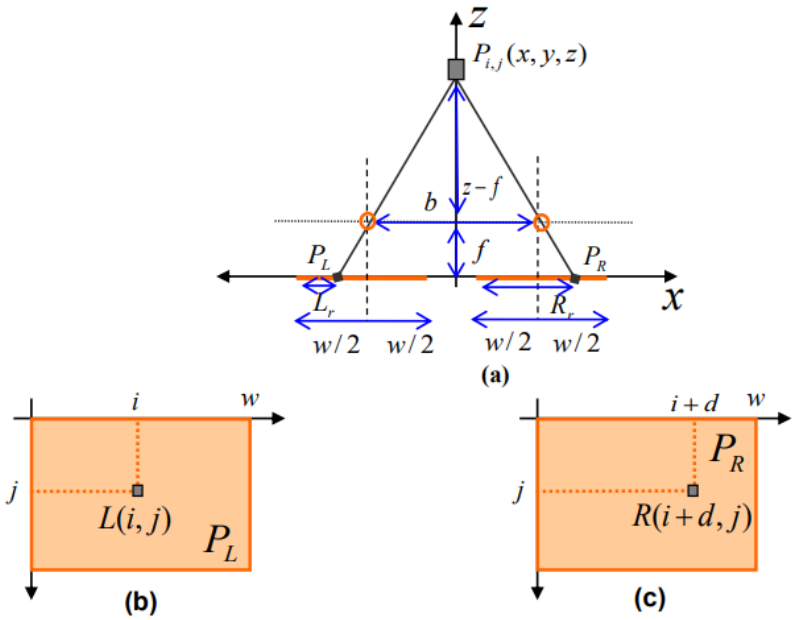


Figura 3. (a) Planteamiento de la estimación de profundidad a partir de dos cámaras frontoparalelas. (b) Imagen en cámara izquierda. (c) Imagen en cámara derecha. Fuente: [25]

En este contexto, se han desarrollado numerosos métodos de localización de herramientas basados en el método explicado anteriormente. En muchos de estos casos, la cámara empleada no es una cámara estereoscópica, por lo que se cuenta únicamente con una imagen en 2 dimensiones, y no 2 como en el caso anterior. Para solucionar este problema se transforma el método para tener un problema inverso al visto anteriormente. En este caso, en vez de tener dos puntos de vista o cámaras, tendremos una sola, pero varios puntos definidos que sabemos perfectamente cómo están dispuestos y orientados.

En este caso, también serán necesarios métodos de extracción de características de las imágenes como, por ejemplo, segmentaciones por intensidades de colores o localización de líneas entre distintos puntos de la imagen. En el artículo publicado por *Cartucho et al.* [26] se explica cómo se puede extraer tanto la posición como la orientación de una herramienta a partir de un marcador cilíndrico marcado con distintos patrones que lo sitúan en el espacio. Mediante una segmentación del marcador y sus marcas, que codifican un número de 8 bits, se pueden alinear los puntos centrales de las marcas y diferenciar si se refiere a un 1 o a un 0, que hacen referencia a elipses paralelas o antiparalelas al eje de la herramienta, respectivamente. Con estos datos, el algoritmo puede diferenciar tanto su posición como su orientación en el espacio.

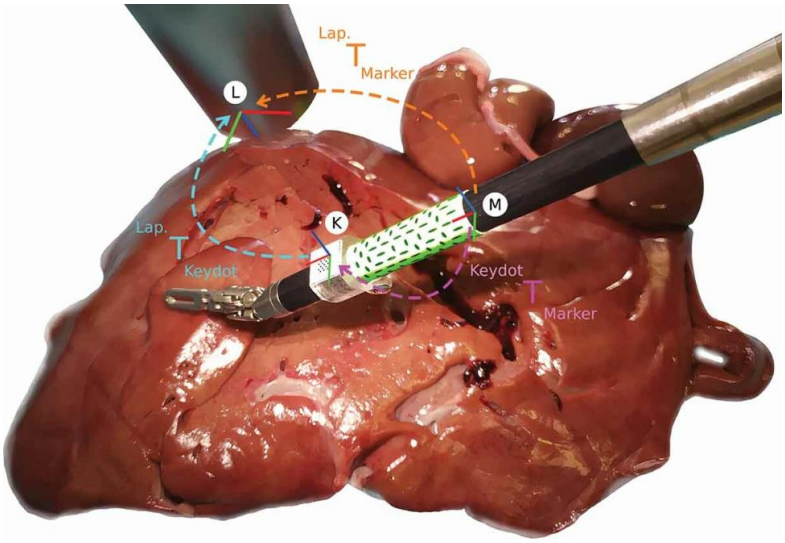


Figura 4. Representación en un modelo de hígado porcino exvivo del campo quirúrgico simulado con los ejes de coordenadas del instrumental. (L) eje perteneciente al laparoscopio. (M) eje perteneciente al marcador. (K) eje perteneciente al KeyDot® para obtener las etiquetas reales de los datos. Fuente: [26]

El limitante de este tipo de algoritmos se divide en dos problemas. El primero de ellos es que tiene un funcionamiento robusto para localizar una de las herramientas, pero su rendimiento decrece cuando debe hacerlo con más de una, ya que los métodos de extracción de características no son capaces de diferenciar las marcas que pertenecen a una herramienta y a otra, lo que puede dar lugar a falsos positivos. En este caso la codificación binaria nos brinda buenos resultados para la localización de dos herramientas, pero están sujetos a que no se pierda la visión directa del marcador por sangre o humo, común en estos tipos de intervenciones. En este caso, sería esperable que el algoritmo no pudiese decodificar las marcas de la herramienta y obtener correctamente su eje de coordenadas. El otro gran problema se basa en la introducción de objetos externos al espacio quirúrgico. Esta limitación veremos que es común a otros tipos de métodos, pero es un limitante a la hora de mantener la asepsia en el campo quirúrgico y obtener certificados para su uso en la práctica clínica.

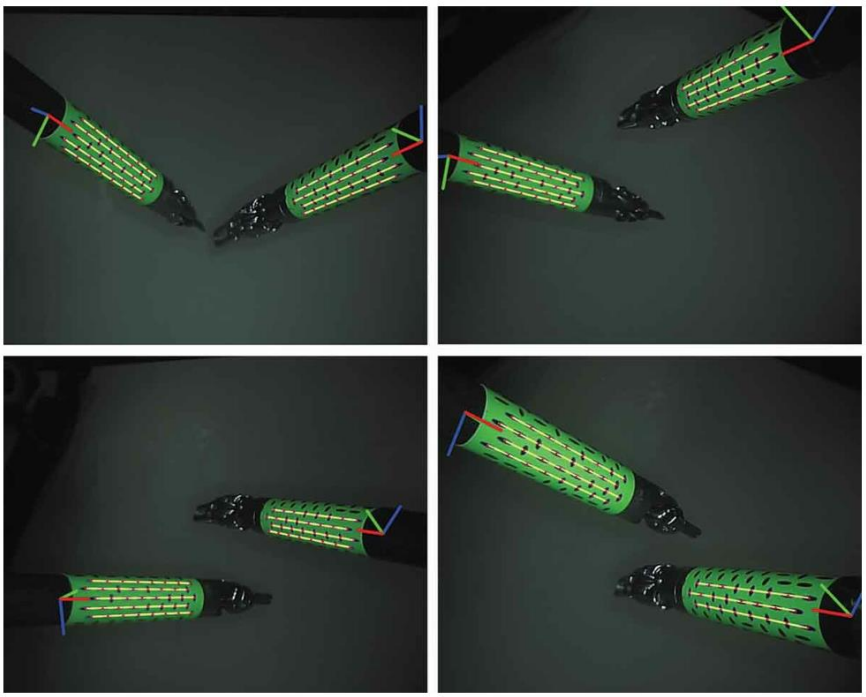


Figura 5. Resultados de la localización de múltiples herramientas. Fuente: [26]

En segundo lugar, otro tipo de métodos para localizar las herramientas en los campos quirúrgicos se basan en sistemas electromagnéticos. Estos equipos están dotados de tres componentes diferenciadas. La primera de ellas es un generador magnético capaz de generar un campo lo suficientemente uniforme para que las deformaciones que se generen, en su mayoría, sean causadas por la posición de la herramienta. El segundo es un sensor conectado mediante un cable que se coloca en el interior del campo generado y que provoca alteraciones en este campo magnético. La tercera de estas partes es un sensor que detecta las modificaciones en el campo magnético que provoca el sensor cableado. El funcionamiento de estas tres partes se fundamenta en obtener el eje de coordenadas del sensor cableado que

está adherido a la zona de interés. Una vez tenemos el eje del sensor cableado respecto del sensor que se encuentra en el endoscopio, se obtiene la transformación del eje del sensor del endoscopio al del endoscopio propiamente dicho, mediante una calibración previa que supone que la pieza siempre se encuentra en la misma posición respecto del endoscopio, ya que este se fija.

En el artículo publicado por *Liu et al.* [27] se utiliza esta aproximación para detectar una sonda de ultrasonidos (LUS) en cirugías laparoscópicas con el propósito de poder fusionar en 3 dimensiones la imagen generada por el ecógrafo y la obtenida por un endoscopio 3D. Los resultados mediante esta técnica se cuantificaron mediante el error al objetivo, definido como la distancia desde el eje de coordenadas supuesto al eje de coordenadas real. En este caso el error se situó en torno a los 2,5 mm. Este error contrasta con el método anterior, en el que el error oscilaba entre 0,1 y 0,4 mm, siendo en todos los casos submilimétrico. Este incremento del error se puede deber a diversos factores. Entre ellos encontramos los fallos inherentes a la calibración del equipo o a las interferencias que generan todos los cuerpos metálicos que intervienen en la cirugía en el campo magnético.

Figura 6. Funcionamiento del endoscopio y el ecógrafo por separado a en las dos imágenes de la izquierda. Funcionamiento del sistema combinando las imágenes del endoscopio y el ecógrafo. [27]

Para evitar gran parte de estos errores, lo ideal sería colocar el sensor del endoscopio lo más cerca posible del sensor del ecógrafo, pero esto nos llevaría a la situación descrita anteriormente, en la cual sería necesario introducir *hardware* externo en el campo quirúrgico. En este caso, además del problema inherente de la seguridad y esterilidad al introducir objetos extraños en el campo, nos encontramos con la necesidad de utilizar un trocar más grande y, por tanto, realizar una mayor herida quirúrgica, lo que aumenta el trauma y, por tanto, afecta a la recuperación del paciente.

Por último, los métodos basados en imagen han cobrado gran popularidad en los últimos años. Hasta 2015, los métodos basados en imagen se fundamentaban en algoritmos

de reconocimiento de características y *machine learning* como el método de Haar, *Local Binary Pattern* o *AdaBoost*. La precisión de estos algoritmos era limitada en la práctica estaban muy restringidos a condiciones bastante controladas. Fue en 2015 cuando apareció el primer método basado en *Deep Learning* y desde entonces el desarrollo en este campo no ha parado de crecer. Estos métodos han mejorado los resultados de los algoritmos anteriores, la velocidad y han eliminado la necesidad de marcadores externos [28].

Este tipo de algoritmos están divididos en dos grupos, detección de objetos en dos pasos y detección de objetos en un paso. El primero de estos grupos comienza con una primera etapa de selección de regiones de interés donde puede encontrarse el objeto, seguida por una segunda de clasificación y posicionamiento de estas regiones de interés para obtener el resultado final. Un ejemplo de este tipo de algoritmos es el modelo R-CNN, capaz de realizar en primer lugar una búsqueda de regiones de interés, para posteriormente realizar predicciones sobre ellas como puede verse en la Figura 7. Para ello, se diferencian 3 partes en este algoritmo. El primero de ellos genera un conjunto de regiones de interés independientes. A continuación, cada una de las regiones pasa por una red convolucional (CNN) para extraer un número fijo de características. Por último, estas características se introducen en una máquina de vector soporte (SVM) para clasificarlas en las diferentes clases. Su principal ventaja es su elevada precisión sin un uso elevado de memoria, pero su gran limitación es su elevado tiempo de inferencia, ya que tiene que procesar con la CNN cada una de las regiones propuestas, lo que limita su uso en tiempo real. Otra desventaja de esta red es que hay que entrenar cada una de sus partes por separado, lo que también ralentiza su entrenamiento [29].

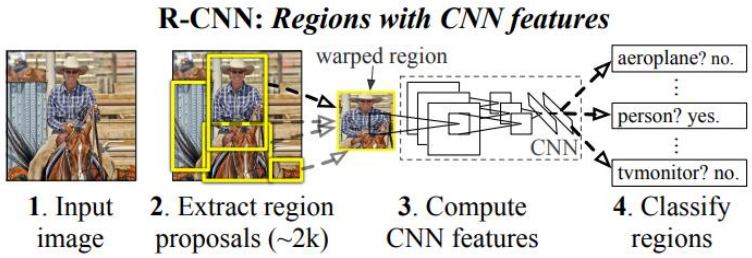


Figura 7. Funcionamiento del modelo R-CNN con cada una de sus partes. Fuente: [29]

En cuanto a los modelos de detección en un paso, se definen por obtener la posición de los objetos tras una única evaluación, lo que les permite realizar inferencias más rápidas. Además, al contrario que en los detectores en dos pasos, que necesitan entrenar cada una de sus partes por separado, en este caso todo el modelo puede entrenarse de principio a fin de la misma forma para un objetivo en concreto. Un ejemplo de un detector de objetos en un paso es YOLO (You Only Look Once), que enmarca la detección de objetos como un problema de regresión a cajas delimitadoras espacialmente separadas y probabilidades de

clase asociadas. Una sola red neuronal predice cajas delimitadoras y probabilidades de clase directamente desde imágenes completas en una evaluación. La arquitectura unificada de YOLO es extremadamente rápida y puede procesar imágenes en tiempo real a 45 cuadros por segundo. Esto lo convierte en una opción atractiva para aplicaciones en tiempo real que requieren detección rápida y precisa de objetos.

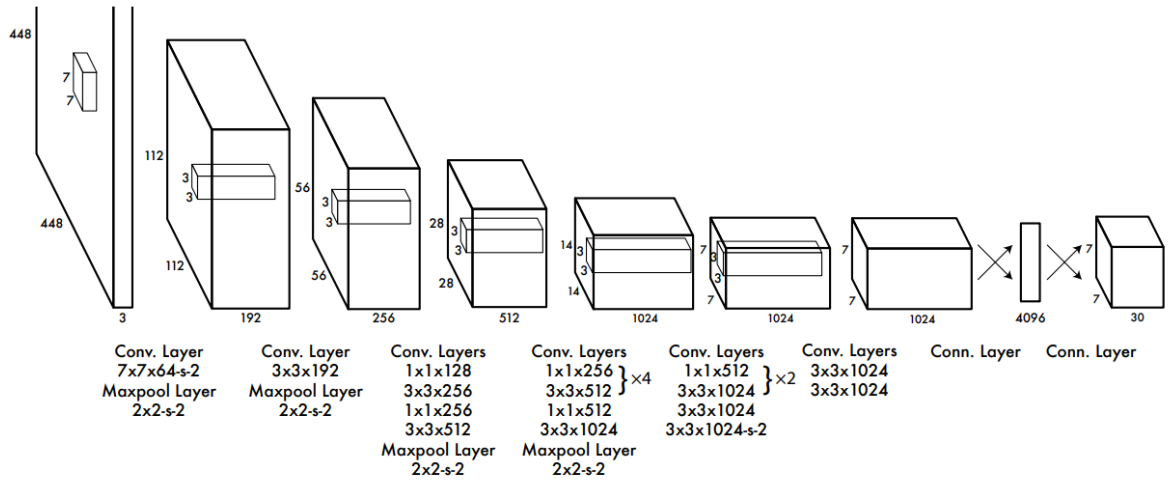


Figura 8. Arquitectura de la red YOLO. Fuente: [30]

YOLO está inspirada en el modelo GoogLeNet para la clasificación de imágenes. La red tiene 24 capas convolucionales para extraer características seguidas de 2 capas completamente conectadas para predecir probabilidades y coordenadas. La capa final predice tanto las probabilidades de clase como las coordenadas de las cajas delimitadoras. Las coordenadas x e y se parametrizan para ser desplazamientos de una ubicación particular de celda de cuadrícula, por lo que también están limitadas entre 0 y 1. La función de perdidas también ha sido optimizada para tener en cuenta tanto la precisión de la predicción como la localización del objeto predicho [30].

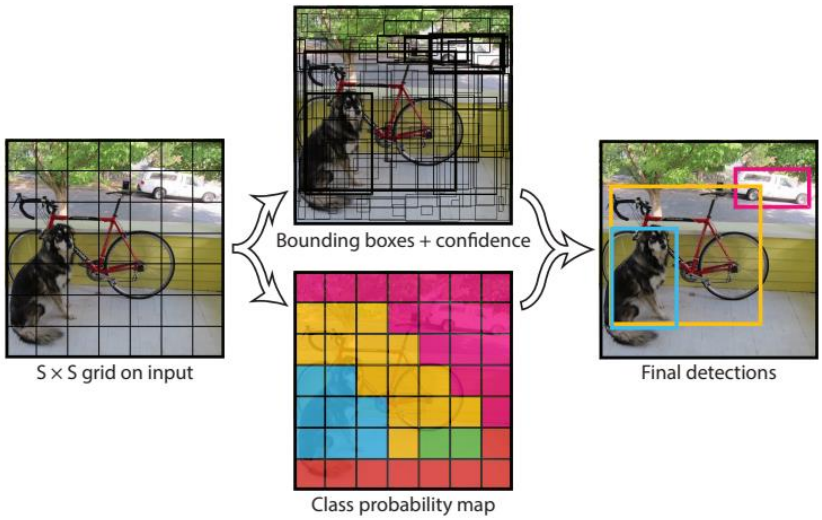


Figura 9. Interpretación conceptual del funcionamiento de la red YOLO. Fuente: [30]

Las limitaciones de estos modelos se basan en que sus predicciones parten de la extracción cajas y la división de las imágenes por celdas. En cada celda de cuadrícula solo predice dos cajas y solo puede tener una clase, por lo que tiene dificultades a la hora de realizar predicciones sobre muchos objetos pequeños que se encuentran muy juntos en la imagen. Además, este modelo tiene dificultades a la hora de predecir cajas con relaciones de aspecto inusuales debido a la arquitectura de sus capas de convolución. Por último, la función de pérdidas no diferencia los errores en cajas pequeñas y cajas grandes, lo que genera que un pequeño error en una caja grande (normalmente insignificante) se tenga en cuenta de la misma manera que un error pequeño en una caja pequeña (mucho más importante).

Es importante mencionar que estos dos modelos son solamente los más representativos en su grupo. Desde su publicación, se han publicado otras redes totalmente distintas o modificaciones de estas redes ya explicadas que mejoran sustancialmente sus resultados y corrigen muchas de sus limitaciones. En los siguientes apartados del proyecto se mencionarán muchos de ellos y se resaltarán cuáles son sus ventajas e inconvenientes respecto a los modelos previos, y respecto a otros modelos que tienen una funcionalidad similar.

Capítulo 3. Descripción del problema

Como se ha explicado en el Capítulo 1 de este trabajo, las técnicas de cirugía laparoscópica y cirugía asistida mediante robots han tomado una gran importancia en las últimas décadas y se espera que estos procedimientos sean más comunes con el paso de los años debido a sus grandes ventajas respecto a las técnicas tradicionales.

En cambio, aunque estas técnicas aportan importantes beneficios sobre las técnicas tradicionales, y sobre todo sobre las cirugías abiertas, aún tienen algunas limitaciones que conllevar errores en el quirófano, afectando directamente al paciente. También pueden provocar un aumento de la curva de aprendizaje de los diferentes profesionales al realizar estas técnicas, que conlleva retrasos en las intervenciones, aumentos en los gastos sanitarios y un menor uso de estas técnicas.

Uno de estos limitantes se basa en el posicionamiento de la cámara, controlada por un asistente en las cirugías laparoscópicas o por el propio cirujano en las operaciones asistidas mediante robots, con los impedimentos que esto conlleva y que ya han sido explicados anteriormente.

Este problema ha sido estudiado durante años desde distintos puntos de vista con el objetivo de obtener un algoritmo capaz de posicionar la cámara de forma que enfoque continuamente al punto de interés. Para ello, en los últimos años se han publicado numerosos estudios sobre la localización de herramientas en entornos de cirugía laparoscópica, algunos de ellos desarrollados en el capítulo 2.

En el caso de este trabajo, queremos abordar el problema de la localización de herramientas sin la necesidad de *hardware* adicional, lo que puede provocar riesgos adicionales en materia de asepsia o aumento del tamaño de las incisiones. Además, en el caso concreto de los dispositivos electromagnéticos, su precisión depende en gran medida de las características de la sala donde se realice la intervención, ya que son especialmente susceptibles a interferencias que pueden estar provocadas por equipos electrónicos indispensables en los quirófanos actuales.

Por ello, el objetivo es encontrar un algoritmo que pueda localizar las herramientas en una imagen a partir únicamente de la imagen que captura el endoscopio. En nuestro caso, las imágenes que se utilizarán serán en 2D ya que es el tipo de imagen que se muestra en la gran mayoría de monitores de cirugía laparoscópica.

Los algoritmos utilizados para localizar herramientas a partir de imágenes han sido muy variados durante las últimas décadas. Hasta 2015 el rendimiento de estos modelos era bastante limitado, pero a partir de ese momento el desarrollo del *Deep Learning* cambió por

completo un campo de investigación que parecía estancado. Actualmente un gran número de modelos de detección de objetos se publican día tras día, mejorando cada vez más su rendimiento.

En el caso concreto en el que se desarrolla este proyecto, que son las técnicas de cirugía laparoscópica, y en menor medida la cirugía robótica, las características de nuestro problema en concreto son algo distintas a las del resto de problemas genéricos.

En primer lugar, para mantener la cámara en su posición continuamente, la respuesta de nuestro algoritmo debe realizarse en tiempo real. Esto significa que el tiempo de inferencia de nuestro modelo debe ser realmente pequeño para que pueda implementarse en una arquitectura de control real.

En segundo lugar, el modelo debe tener una robustez suficientemente alta para asegurar que las predicciones sean correctas. En este caso, es importante minimizar los falsos positivos, que en nuestro caso sería una predicción en una coordenada que no entra en el rango deseado, y los falsos negativos, que se refiere a no realizar una predicción de la posición de la herramienta cuando esta sí se encuentra en la imagen. En cambio, aunque lo ideal sería minimizar los dos, es especialmente importante evitar los primeros de estos errores, ya que, si el algoritmo no logra localizar la herramienta en algún *frame* esporádico, la predicción de los siguientes podría subsanar el error, o incluso el profesional podría mover la cámara de forma manual. Por el contrario, en caso de aparecer una predicción en un lugar erróneo, el profesional podría perder momentáneamente el campo de visión sobre la zona de interés, lo que en algunos instantes cruciales de la cirugía podría suponer un gran peligro para el paciente.

Por último, en nuestro ámbito nos encontramos un gran limitante a la hora de entrenar este tipo de modelos, los datos. Para entrenar este tipo de modelos es necesario poseer una gran cantidad de datos, que representen de forma completa el problema al que se quiere enfrentar. Debido al origen que deben tener estos datos, no es fácil poder acceder a ellos para personal ajeno al hospital. Además, el problema realmente no se encuentra en la cantidad de imágenes de cirugías laparoscópicas a las que se puede acceder, sino la cantidad de ellos que se encuentran debidamente etiquetados. Los videos de cirugías laparoscópicas son de gran utilidad didáctica y existen repositorios dedicados a recopilar estos videos para ayudar a futuros cirujanos a mejorar sus técnicas. La barrera que nos encontramos en este caso es que estos videos no están etiquetados para poder entrenar un modelo, y el número de profesionales sanitarios capacitados y dispuestos a etiquetarlos es realmente reducido, lo que resulta en una cantidad de datos reales en muchos casos insuficiente. Este contratiempo nos obliga a utilizar imágenes extraídas de simulaciones, más sencillas de obtener, pero menos representativas del problema en cuestión.

En resumen, nos enfrentamos a un problema muy variable que depende de las estructuras anatómicas, el tipo de intervención y los artefactos que afecten a las imágenes. Por ello, necesitamos un algoritmo que sea capaz de generalizar con un rendimiento lo suficientemente bueno para aplicarlo en un sector como el de la cirugía. En este caso, la visión artificial nos ha proporcionado la posibilidad de entrenar un modelo que cumpla con estos criterios. El impedimento que nos encontramos con estos modelos es la reducida cantidad de datos etiquetados que existen y la necesidad de obtener una inferencia realmente rápida, lo que se abordará en el resto del trabajo.

Parte II. Trabajo desarrollado.

Capítulo 4. Marco teórico del proyecto.

4.1. Redes Neuronales y Deep Learning

El siguiente marco teórico se desarrollará basándose en las referencias [31], [32], que dan una visión global del Deep learning y de las distintas arquitecturas de redes neuronales.

Los paradigmas tradicionales de la programación consisten en indicarle a un ordenador qué hacer mediante instrucciones comprensibles por un sistema informático para transformar un gran problema en algo que un ordenador puede realizar de forma sencilla. La aproximación del aprendizaje profundo y las redes neuronales es totalmente distinta, ya que en ningún momento vamos a indicarle como solucionar el problema, sino que le suministraremos un conjunto de datos y el modelo de forma automática se ajustará y optimizará con esa información para resolver el problema que le planteemos.

Hasta el año 2006, los desarrollos en este campo estaban paralizados debido a la falta de conocimiento a la hora de entrenar estos modelos. Más allá de algunos modelos entrenados para tareas muy específicas, ninguno de los nuevos modelos superaba el rendimiento de los algoritmos tradicionales. Fue con la llegada de las redes neuronales profundas cuando cambió el paradigma por completo, ya que estas técnicas mostraron mejores rendimientos en campos tan distintos como visión artificial, procesamiento del lenguaje o reconocimiento de voz. Es tan importante el impacto que están teniendo estas técnicas que empresas como Microsoft o Google tienen importantes secciones dedicadas a solucionar problemas de este modo.

Las redes neuronales han sido creadas para replicar el sistema nervioso humano, de forma que, mediante operaciones matemáticas, se busca recrear la forma en la que los cerebros humanos son capaces de aprender en base a la información que adquieren del ambiente. Actualmente, el ordenador más ponente del mundo tiene únicamente una pequeña fracción de la potencia de procesamiento que tiene un cerebro humano, pero teóricamente una red neuronal con los suficientes datos para entrenar podría realizar cualquier tarea, lo que se denomina *Turing Completeness.*

La unidad funcional de una red neuronal la denominamos perceptrón. El perceptrón tiene un comportamiento similar a una neurona individual en nuestro organismo, la unidad estructural y funcional del sistema nervioso humano. En el caso humano, la información se transmite a través del sistema nerviosos mediante impulsos eléctricos en forma de potenciales de acción. Las encargadas de transmitir esta información son las neuronas, compuestas por tres partes principales. En primer lugar, nos encontramos con las dendritas, estructuras conectadas a otras neuronas y encargadas de recibir la información o potenciales de acción de las neuronas adyacentes. En segundo lugar, tenemos el cuerpo de la neurona, encargado de albergar el núcleo de la célula y de procesar los potenciales aferentes para elaborar una respuesta. Por último, tenemos el axón, el responsable de propagar el potencial de acción a las neuronas a las que están conectadas sus terminaciones axónicas.

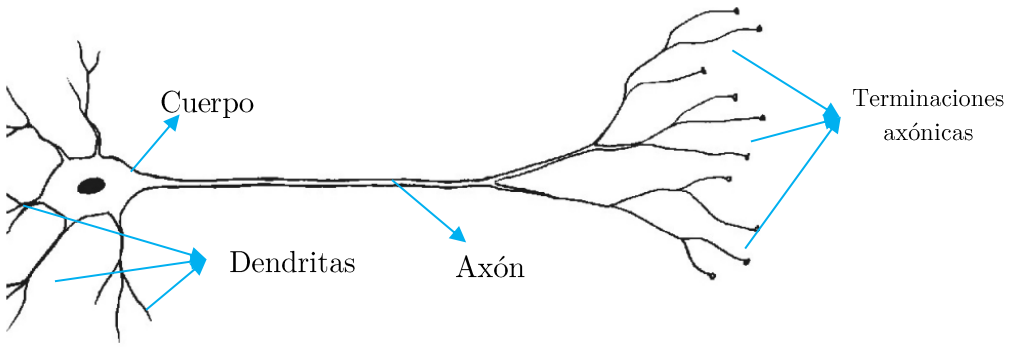


Figura 10. Representación de la estructura de una neurona humana. Fuente: Elaboración propia

En los humanos, el potencial de acción que se dispara es binario, es decir, es una señal eléctrica de todo o nada. El funcionamiento de una neurona se basa en recibir los potenciales aferentes y ponderar cada uno de ellos. Si esta ponderación supera un límite de potencial eléctrico, todos los canales iónicos se abren despolarizando la célula y disparando un potencial de acción eferente que se transmitirá a las células siguientes mediante la sinapsis.

Como hemos mencionado anteriormente, la unidad funcional de una red neuronal es **el perceptrón**. Esta estructura intenta replicar la neurona que acabamos de explicar, de forma que posee unas entradas o *inputs* que se procesan y ponderan mediante unos pesos y una función lineal que da lugar a una salida. No es difícil ver que los inputs se comportan de forma similar a las dendritas, los pesos y la función lineal o de activación se comporta de forma similar al cuerpo neuronal y la salida se refiere al axón.

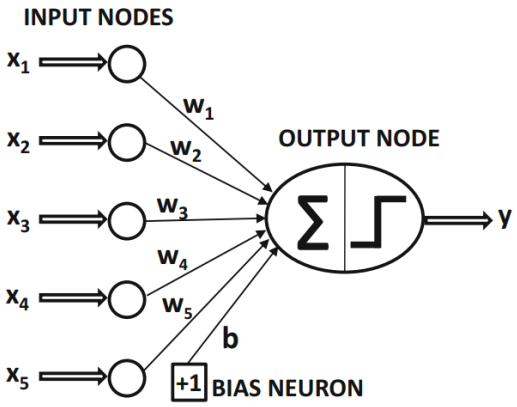


Figura 11. Estructura del perceptrón simple con entrada de bias. Fuente: [31]

Para entrenar estos tipos de modelos necesitamos datos con las entradas y las salidas que deseamos, con el objetivo de poder comparar el rendimiento de nuestro algoritmo con la realidad, y que así pueda ir ajustándose de forma automática a medida que avanzan las épocas, que es como denominamos a cada una de las veces que el conjunto de datos de entrenamiento ingresa en la red. Para el caso de este perceptrón, la salida será la función aplicada a las entradas ponderadas con los pesos de acuerdo con la siguiente ecuación:

$$\hat{y} = \text{func}\left(\sum_{i=1}^{5} X_i * W_i + b\right) \tag{2}$$

En la ecuación anterior, la salida dependerá de la función que utilice el modelo, que serán explicadas posteriormente en mayor profundidad. A continuación, se compararán las salidas del modelo con los valores reales que el modelo debería haber proporcionado según los datos que tenemos, obteniendo una medida de error que llamaremos **función de pérdidas** *(loss function)*.

$$\text{loss} = E(y - \hat{y}) \tag{3}$$

La intención del modelo es minimizar esta función de pérdidas, ya que pretendemos que las predicciones se ajusten a la realidad, siendo menores los errores conforme menor es nuestra función de pérdidas. Para minimizar esta función, se puede desglosar en la siguiente ecuación:

$$\text{loss} = E\left(y - \text{func}\left(\sum_{i=1}^{5} X_i * W_i + b\right)\right) \tag{4}$$

Como se puede ver en la ecuación 4, para minimizar la función de pérdidas habrá que obtener la derivada respecto de cada uno de los pesos y el *bias*, es decir, **el gradiente** de la función de pérdidas respecto de cada uno de los pesos. Una vez obtenemos el gradiente de la función respecto de cada uno de los pesos, los actualizaremos en sentido contrario a ese gradiente, para avanzar hacia zonas donde la función de pérdidas sea menor.

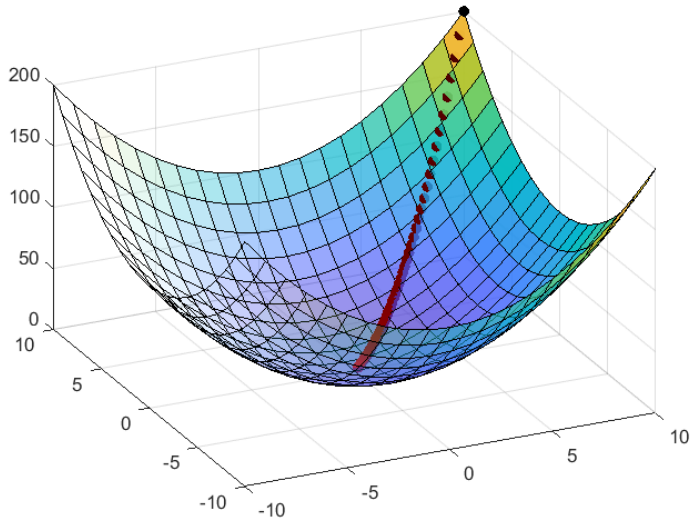


Figura 12. Simulación del funcionamiento del método de descenso de gradiente. Fuente: Elaboración propia

El perceptrón que hemos explicado aprendería a darles más importancia a unas entradas u otras en función de la variación de los pesos y elaboraría una salida en base a ello. En cambio, este perceptrón dista mucho del razonamiento humano ya que nosotros no utilizamos una única

neurona para tomar la decisión en base a sus entradas, sino que la información se procesa mediante millones de neuronas para extraer otras características que no son las entradas explícitamente. Por este motivo, el perceptrón no es capaz de ofrecer un buen rendimiento en muchas situaciones. La solución más intuitiva a este problema sería, una vez más, simular el funcionamiento del cerebro humano, conectando una gran cantidad de neuronas, en este caso, perceptrones, y formar una red más compleja.

Esta red tiene un funcionamiento más similar al que realiza un cerebro humano al

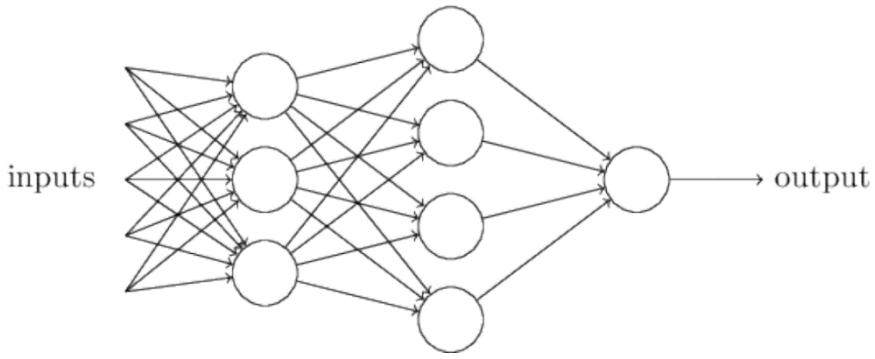
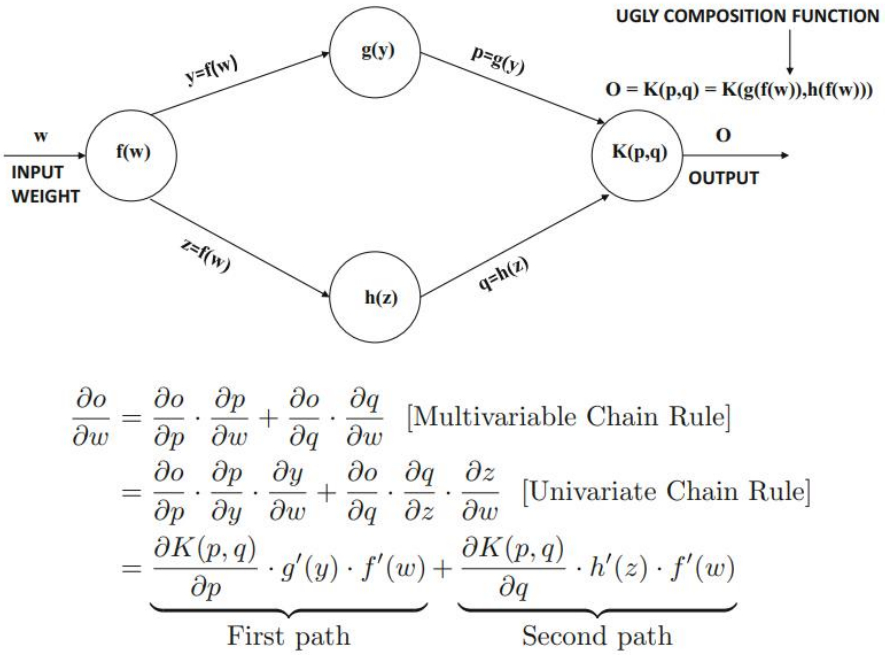


Figura 13. Combinación de perceptrones. Fuente: [31]

procesar la información. En primer lugar, posee una capa de perceptrones que generan un resultado a partir de una ponderación que realizan de las entradas que reciben, de forma idéntica al perceptrón simple. En segundo lugar, la red posee una segunda capa de neuronas o perceptrones, que procesan las salidas de la capa anterior, es decir, generan una respuesta en base a una ponderación de las salidas anteriores. Este procesamiento es mucho más complejo que el caso anterior, y la extracción de características que se lleva a cabo a partir de la información inicial es mucho más profunda y abstracta, replicando de una forma más realista el cerebro humano.

En el caso del perceptrón simple, es sencillo obtener el gradiente de la función de pérdidas respecto de cada uno de los pesos. En cambio, con un número elevado de capas ocultas esta relación deja de ser tan sencilla, ya que el gradiente depende de los pesos de la última capa, que a su vez dependen de la capa anterior, y estos a su vez de la anterior, hasta llegar a la capa de entrada. Para solucionar este problema se utiliza el **algoritmo de retropropagación** o *back-propagation*. Este algoritmo se compone de dos fases, la primera de ellas calcula la salida de la red a partir de una entrada utilizando los pesos actuales. De esta forma, conseguimos una salida que podemos comparar con el valor real mediante la función de pérdidas. La segunda de estas etapas utiliza la regla de la cadena en ecuaciones diferenciales para calcular la derivada respecto de cada uno de los pesos de la red a partir de la derivada de la función de pérdidas en cada neurona. Su funcionamiento se puede ver en el ejemplo plasmado en la figura 14.



$$\frac{\partial o}{\partial w} = \frac{\partial o}{\partial p}\cdot\frac{\partial p}{\partial w} + \frac{\partial o}{\partial q}\cdot\frac{\partial q}{\partial w} \quad \text{[Multivariable Chain Rule]}$$

$$= \frac{\partial o}{\partial p}\cdot\frac{\partial p}{\partial y}\cdot\frac{\partial y}{\partial w} + \frac{\partial o}{\partial q}\cdot\frac{\partial q}{\partial z}\cdot\frac{\partial z}{\partial w} \quad \text{[Univariate Chain Rule]}$$

$$= \underbrace{\frac{\partial K(p,q)}{\partial p}\cdot g'(y)\cdot f'(w)}_{\text{First path}} + \underbrace{\frac{\partial K(p,q)}{\partial q}\cdot h'(z)\cdot f'(w)}_{\text{Second path}}$$

Figura 14. Funcionamiento del algoritmo *back-propagation* mediante la regla de la cadena. Fuente: [31]

Mediante este planteamiento, se pueden calcular de forma recursiva la derivada respecto de cada peso para cada capa. Este desarrollo supuso un cambio drástico en un campo de conocimiento que parecía estancado para su llegada. Este método, desarrollado por Rumelhart, Hinton y Williams en 1986 supuso un cambio de paradigma y permitió entrenar modelos con resultados incomparables a los conocidos hasta la fecha.

Otro aspecto fundamental de estas redes es el uso de distintas **funciones de activación**. Existen multitud de funciones de activación, que se utilizarán dependiendo del problema al que nos estemos enfrentando. Las primeras que se utilizaron son las funciones lineales, cuya principal ventaja es el bajo coste computacional que supone su uso. En cambio, este tipo de funciones de activación actualmente ya no se utilizan debido a que, al formar una combinación de funciones lineales, su salida también es lineal, no siendo capaz de realizar predicciones en conjuntos de datos con distribuciones no lineales.

Para solucionar este problema se han desarrollado diversas funciones no lineales que mejoran sustancialmente los resultados anteriores. La primera de estas funciones es **ReLU** (*Rectified Lineal Unit*), que tiene una salida lineal para valores de entrada mayores que 0 y un cero absoluto para valores negativos. Su ventaja principal es la introducción de la no linealidad con una función que no requiere un gran coste computacional. Su principal limitación es la creación de "neuronas muertas", que se producen cuando las entradas a esas neuronas son valores negativos. Para solucionar este problema, se han desarrollado algunas modificaciones de esta función como **LeakyReLU**, que para valores negativos de entrada no genera una salida nula, sino que tiene una pequeña pendiente en la zona negativa del eje.

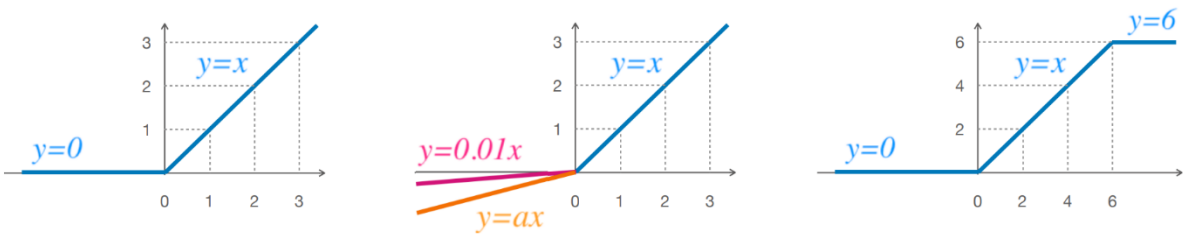


Figura 17. Funciones de activación de la familia ReLU. ReLU (izquierda), LeakyRelu (centro) y ReLu6 (derecha) [41].

Además de estas funciones, podemos encontrarnos algunas otras que nos permiten mejorar el rendimiento de nuestra red en función del objetivo que queremos conseguir. Entre estas funciones, algunas de las más conocidas son la función **sigmoide** o la **tangente hiperbólica**, en caso de que nuestro objetivo sea que la red nos proporcione una probabilidad. Otra de las funciones más importantes es **softmax**, que nos devuelve una distribución de probabilidad para cada una de las salidas de la red, de forma que la suma de todas ellas es igual a 1.

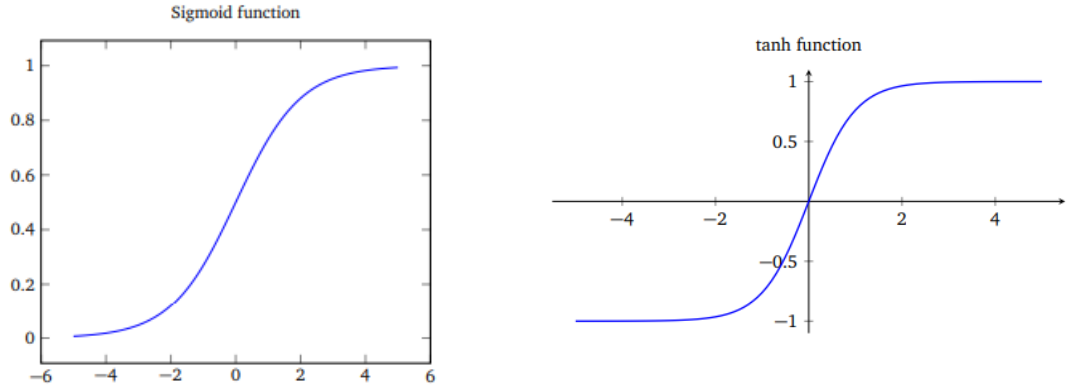


Figura 15. Representación de la función de activación sigmoide y tangente hiperbólica. Fuente: [32]

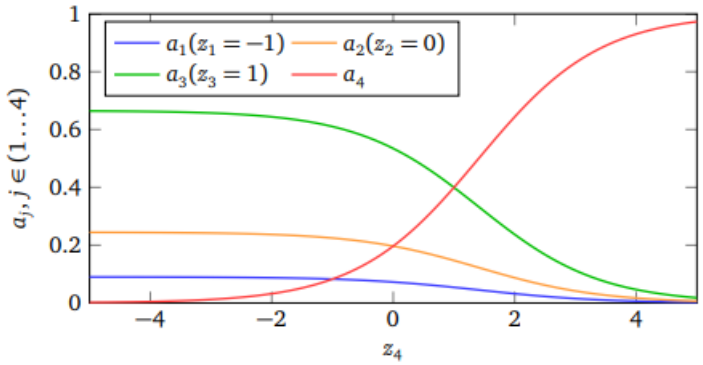


Figura 16. Representación de la variación de las salidas de una neurona con función de activación *softmax* al variar uno de los pesos de las entradas. Se puede ver que al tratarse de una función de probabilidad siempre suma 1. Fuente: [32]

En los entrenamientos de redes profundas se hace uso de **algoritmos de optimización**. Los optimizadores son algoritmos que sirven para entrenar de forma más eficiente nuestro modelo. Como se ha explicado anteriormente, se utiliza el método de *back-propagation* para actualizar los pesos en contra de gradiente. Para ello se obtiene el gradiente de la función de

pérdidas respecto de cada peso. A continuación, se multiplica ese valor del vector del gradiente por el ***learning rate***, definido en muchas funciones de programación como *lr*. Este valor define cuánto cambiarán los pesos de la red cada vez que se actualicen.

Si tenemos un valor de *lr* muy pequeño, puede ser que la red tarde mucho en converger debido a que el avance en contra de gradiente sea muy lento. En cambio, si el *lr* tiene un valor demasiado grande, podemos encontrarnos con un modelo que no es capaz de converger hasta el mínimo de la función de coste.

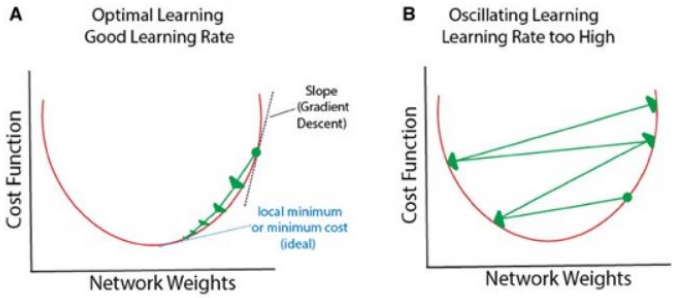


Figura 18. Diferencia entre la elección de un *learning rate* óptimo y uno con un valor demasiado elevado. Fuente: [42]

Lo ideal en estos casos es hacer uso de **algoritmos de optimización**. Este tipo de algoritmos se encargan de hacer variaciones en el *learning rate* dependiendo de algunos parámetros del entrenamiento, como la frecuencia de cambio de signo de la dirección de gradiente, o el valor tiempo pasado de entrenamiento. Permitir que el *lr* cambie con el tiempo es una primera aproximación para optimizar un entrenamiento, pero en la actualidad se usan otros tipos de optimizadores cuyos resultados son mucho mejores.

El modelo más sencillo se denomina ***Stochastic Gradient Descent***. Este algoritmo actualiza los pesos en base a la estimación del gradiente habiendo calculado la función de pérdidas solamente en algunas de las muestras. Esta implementación supone un ahorro en el coste computacional del modelo y una menor probabilidad de que el modelo converja hacia un mínimo local mediante el componente estocástico del algoritmo, en vez de llega hasta el mínimo global.

Otra de estas implementaciones es el uso del momento. Esta aproximación nos permite tener en cuenta la dirección de descenso de gradiente de momentos anteriores y aplicarle un "momento" a este descenso, de forma que, si se encuentra con un mínimo local, sea menos probable que el entrenamiento se encierre en esta zona, y evolucione hacia el máximo global.

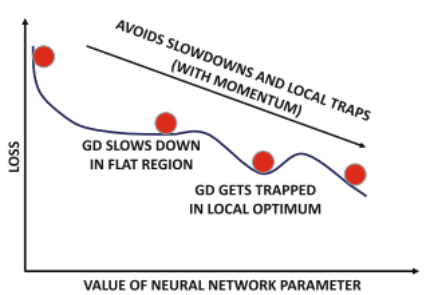


Figura 19. Representación de la utilidad del uso del momento en la actualización de los pesos. Fuente: [43]

Uno de los optimizadores más usados hoy en día es el optimizador ***AdaGrad***. Este optimizador se basa en actualizar los pesos de manera inversamente proporcional al valor del vector de gradiente en ese punto. De esta forma, cuando la función de coste es muy elevada respecto de alguno de los pesos, estos se actualizarán en menor medida que en los casos donde el gradiente sea mucho menor. El resultado de esta aproximación es un modelo con un *learning rate* variable que en zonas donde la función de coste tiene grandes pendientes tendrá actualizaciones pequeñas de los pesos, que estabiliza en gran medida el modelo, y que en zonas donde la función de pérdidas es mucho más plana, el *learning rate* se actualizará de forma mucho más abrupta, acelerando mucho la convergencia del modelo. La principal limitación de este método es la acumulación de los cuadrados del gradiente en el denominador de la función, lo que provoca que, con el paso del tiempo, el *learning rate* pueda volverse demasiado pequeño y provoque un entrenamiento demasiado largo. Para ello se han desarrollado algoritmos alternativos como ***Adam*** o ***RMSProp***.

$$\mathrm{A_i} \Leftarrow \mathrm{A_i} + \frac{\partial \mathrm{L}}{\partial \mathrm{w_i}}^2 , \forall A_i \tag{5}$$

$$\mathrm{w_i} \Leftarrow w_\mathrm{i} - \frac{\alpha}{\sqrt{A_i}} \frac{\partial L}{\partial w_i}, \forall A_i \tag{6}$$

En las ecuaciones 5 y 6 se pueden ver las funciones que se utilizan para actualizar los pesos según el método *Adagrad*. En la ecuación 7 podemos ver la diferencia a la hora de actualizar los pesos que realiza *RMSProp.* En este caso, en vez de hacer una acumulación de los cuadrados de los gradientes, se realiza una ponderación exponencial de estos cuadrados, de forma que se evita el desvanecimiento del gradiente.

$$\mathrm{A_i} \Leftarrow \rho \mathrm{A_i} + (1-\rho) \frac{\partial \mathrm{L}}{\partial \mathrm{w_i}}^2 , \forall A_i \tag{7}$$

En el caso del algoritmo *Adam*, se utiliza una aproximación similar a la realizada anteriormente con el momento, añadiendo otro nuevo parámetro a la hora de actualizar los pesos. Esta modificación tiene dos principales ventajas. La primera de ellas es la aplicación del ya mencionado momento, y la segunda es que ahora el *learning rate* depende del su valor en la iteración *t*. Las funciones de este método se definen a continuación.

$$\mathrm{A_i} \Leftarrow \rho \mathrm{A_i} + (1-\rho) \frac{\partial \mathrm{L}}{\partial \mathrm{w_i}}^2 , \forall A_i \tag{8}$$

$$\mathrm{F_i} \Leftarrow \rho_\mathrm{f} \mathrm{F_i} + (1-\rho_\mathrm{f}) \frac{\partial L}{\partial w_i}, \forall A_i \tag{9}$$

$$w_i \Leftarrow w_i - \frac{\alpha}{\sqrt{A_i}} F_i, \forall A_i \tag{10}$$

No hay un consenso a la hora de elegir qué optimizador utilizar para cada modelo. Actualmente hay algunos que son mucho más utilizados que otros, como *Adam* o *RMSProp*, pero la elección a la hora de utilizar unos u otros se basa fundamentalmente en la experiencia y la puesta en práctica.

Una vez hemos diseñado nuestro modelo, un problema habitual al que nos enfrentamos al entrenar este tipo de redes es el **sobreajuste** *(overfitting)*. Este fenómeno se da cuando nuestro modelo sabe adaptarse de una forma excelente a los datos que tenemos por su elevado número de parámetros entrenables, pero no es buen modelo ya que no es capaz de generalizar bien en situaciones ligeramente distintas a las utilizadas para entrenarlo. Este efecto en nuestros modelos los podemos ver en la diferencia entre las métricas que consigue el modelo con el conjunto de entrenamiento y con el de validación o test, dependiendo el caso.

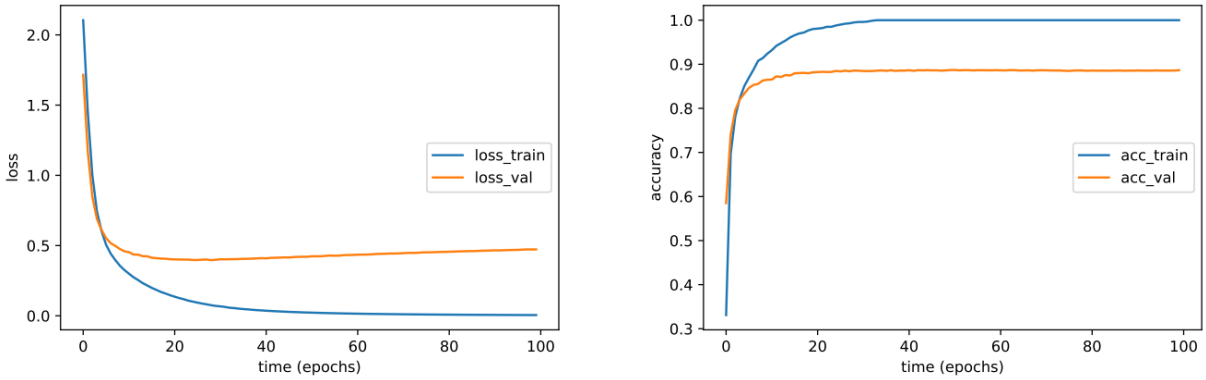


Figura 20. Visualización del *overfitting* en un modelo mediante la función de *loss* (a) y exactitud (b)**.** Fuente: [44]

En la figura 20 podemos ver como el modelo sobreajusta los sus pesos a los datos de entrenamiento, por lo que ofrece muy buenos resultados para estos datos, pero no para datos con pequeñas diferencias, como los que se incluyen en el set de validación. Este fenómeno suele ser habitual a medida que los modelos, en nuestro caso las redes neuronales, se vuelven más complejas y profundas. También nos lo encontramos cuando tenemos pocos datos de entrenamiento, y la red debe entrenar repetidamente sobre los mismos datos para mejorar su rendimiento.

Para solucionar este problema existen distintas aproximaciones. Las más evidentes serían conseguir un conjunto de datos aún mayor con mayor variabilidad, lo que permitiría a la red adaptarse a múltiples situaciones y mejorar su generalización. El problema en este caso radica en que no siempre es posible conseguir esta cantidad de datos, y menos aún en entornos clínicos. La otra sería reducir el número de capas de la red. Esto sería válido si el modelo llevase a cabo su función sin necesidad de tantas capas, puesto que un modelo más sencillo siempre es mejor si mantiene su rendimiento. El problema que nos encontramos es que no suele ser lo habitual. Normalmente al reducir el tamaño del modelo, este tiende a empeorar sus resultados, por lo que tampoco es una solución universal al problema.

Un conjunto de soluciones que se han ideado para que los modelos sean más robustos ante el *overfitting* se denominan **técnicas de regularización**. Una de las técnicas de regularización más usada es la regularización L2. Esta técnica consiste en agregar un término adicional en la función de coste que penalice los pesos con valores absolutos altos, que pueden ser una de las causas del *overfitting.*

$$C = C_0 + \frac{\lambda}{2n}\sum_w w^2 \tag{11}$$

Como se puede ver en la ecuación 11, los valores altos de lambda se asociarán con pesos más pequeños, lo que podría significar en casos extremos que el modelo no entrene de forma correcta para ninguno de los conjuntos de datos, que denominamos **subajuste** *(underfitting).* Otra técnica de regularización derivada de la regularización L2 es la regularización L1. En este caso el término que se le añade a la función de coste se refleja en la ecuación 12.

$$C = C_0 + \frac{\lambda}{2n}\sum_w |w| \tag{12}$$

Se puede ver que en ambos tipos de regularización se castigan los valores altos de los pesos. En cambio, la forma que las dos funciones tienen de restringir los valores de los pesos son ligeramente distintas. En el caso de la regularización L2, si derivamos la función para ver cómo se actualizarían los pesos, la componente que introducimos con el nuevo término penaliza los pesos grandes con una gravedad proporcional a los pesos. En cambio, si hacemos lo mismo con la regularización L1, esta penaliza los pesos distintos de cero, por lo que tenderá a llevar a los pesos a valores cercanos a cero.

Otra técnica de regularización comúnmente usada, y totalmente distinta a las anteriores es la eliminación de neuronas o ***dropout***. Esta técnica se basa en inhabilitar neuronas de forma aleatoria y temporal para obligar a la red a entrenar sin este tipo de neuronas. Esta técnica consigue que a la red le cueste más "memorizar" los datos de entrada y entrene de forma robusta ante distintos tipos de entradas. Una aproximación cualitativa de lo que sería utilizar esta técnica sería el entrenamiento consecutivo de un conjunto de redes ligeramente diferentes y que el resultado final sea un promediado de todas ellas. Con un poco de suerte, cada una de ellas habrá sobreentrenado en un sentido distinto, y al combinarlas todas podremos evitar el *overfitting.*

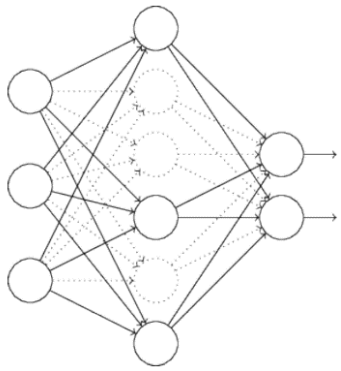


Figura 21. Representación de la aplicación de la técnica *dropout* en una red. Fuente: [32]

En ocasiones, el *overfitting* es causado por un exceso de épocas de entrenamiento, en el cual la red ya ha llegado a los mejores pesos que puede ofrecer para ese modelo. A partir de ahí, únicamente ajusta sus pesos para adaptarse mucho mejor a los datos de entrenamiento, empeorando su generalización para los datos de test. Una posible solución es hacer uso de una técnica llamada ***EarlyStopping*** o parada prematura. En esta técnica se monitorizan las métricas y se para el entrenamiento cuando las que nos informan sobre el set de validación empiezan a empeorar. De esta forma, nos quedamos con el modelo que mejor ha sabido generalizar en muestras aparentemente distintas.

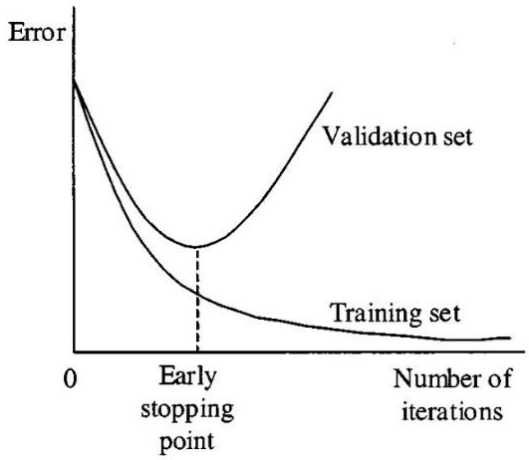


Figura 22. Funcionamiento de la técnica *EarlyStopping*. Fuente: [45]

Por último, otra técnica comúnmente utilizada al entrenar las redes consiste en **guardar el mejor modelo** en base a una métrica en concreto. Con esto conseguimos realizar una tarea similar al *EarlyStopping* pero a la hora de quedarnos con un modelo. Nuestro objetivo en este caso es guardar únicamente el modelo que mejores resultados nos aporte en el set de validación, de forma que posteriormente utilizaremos para hacer inferencia en el conjunto de test únicamente los modelos que no han sobreentrenado.

4.2. Redes Neuronales Convolucionales

Las redes neuronales convolucionales se tratan de modelos desarrollados para trabajar con estructuras de datos en forma matricial cuyas relaciones espaciales son especialmente relevantes. Uno de los ejemplos más evidentes de este tipo de estructura de datos son las imágenes en dos dimensiones. Una propiedad de las imágenes que no poseen otros tipos de datos es su alto grado de invarianza traslacional, que se refiere a que las características de un objeto no dependen del lugar que ocupe en la imagen. Las redes convolucionales, como su propio nombre indica, hacen uso de una operación matemática llamada **convolución** para extraer características de distintas regiones de la imagen, como el color, los contornos o las formas, para usarlo en sus predicciones posteriores.

Hasta ahora, las redes explicadas usarían el valor de intensidad de cada píxel para introducirlo en una red mediante un número de neuronas igual al número de píxeles y realizar una predicción en base a ellos. Esto nos puede funcionar bien en algunas ocasiones, pero no es muy utilizado en el análisis de propiedades de imagen, ya que este modelo interpreta cada píxel como una unidad individual y no tiene en cuenta las características espaciales de la imagen en su funcionamiento.

Las redes convolucionales sí que tienen en cuenta la información posicional, lo que las hace ideales para el análisis de imágenes, cuyas características dependen en gran medida de la información espacial. Para ello, cada píxel no será una neurona de entrada independiente, sino que cada una de las entradas a las neuronas convolucionales serán un conjunto de píxeles de una región, y la salida será una imagen del mismo tamaño, o un tamaño similar a la imagen de entrada.

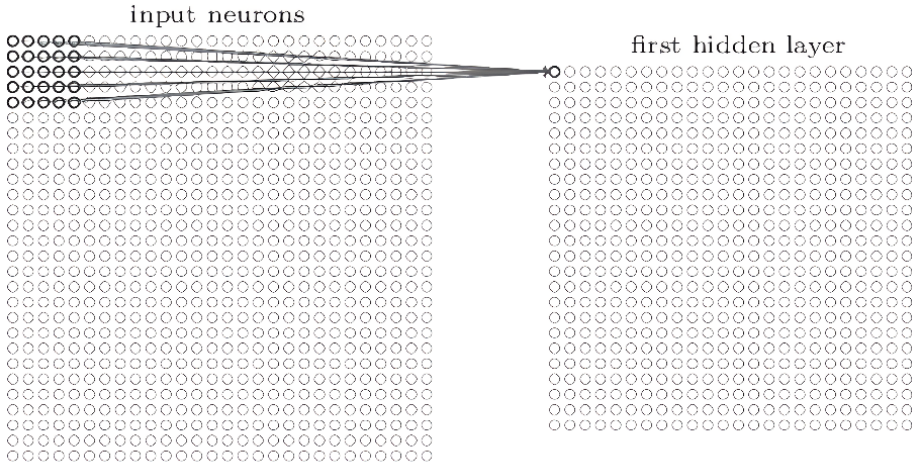


Figura 23. Representación del campo espacial que sirve de entrada a cada neurona. Fuente: [32]

La operación principal de este tipo de redes es la convolución, mediante el uso de distintas **máscaras o *kernels***. La convolución se define por una suma del valor de cada uno de los píxeles multiplicados por el valor del *kernel* de convolución en esa posición. De esta forma, dependiendo de los tamaños y distribuciones de los valores de los *kernels* de convolución, el resultado serán activaciones más elevadas en zonas con unas u otras características.

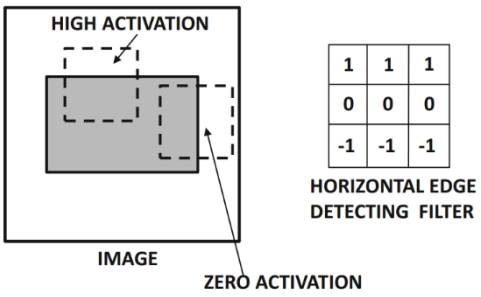


Figura 24. Activación de un *kernel* de detección de bordes horizontales. Fuente: [32]

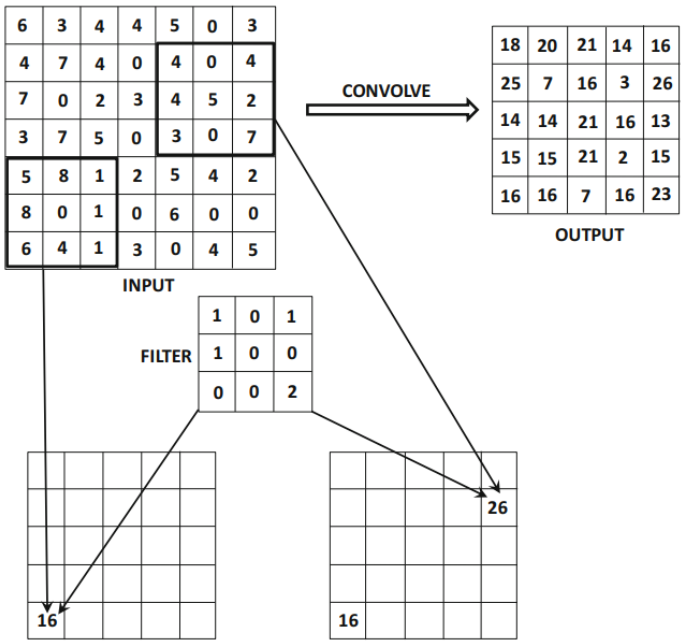


Figura 25. Representación de la operación de convolución y su resultado para un filtro 3x3. Fuente: [32]

Como se puede ver en la figura 22, la operación de convolución reduce el tamaño de la imagen debido al efecto de bordes. Esta reducción de tamaño depende del tamaño del filtro, siendo mayor a medida que aumentamos sus dimensiones. Para solucionar este problema existe la opción de rellenar los bordes de la imagen ***(padding)***. Esta operación consiste en añadir valores o píxeles a nuestra matriz para que, al realizar la convolución, conserve las dimensiones iniciales. Existen distintas formas de realizar *padding* en una imagen, añadiendo "ceros" en los bordes, dando valores en forma de espejo o con un valor constante, entre otros.

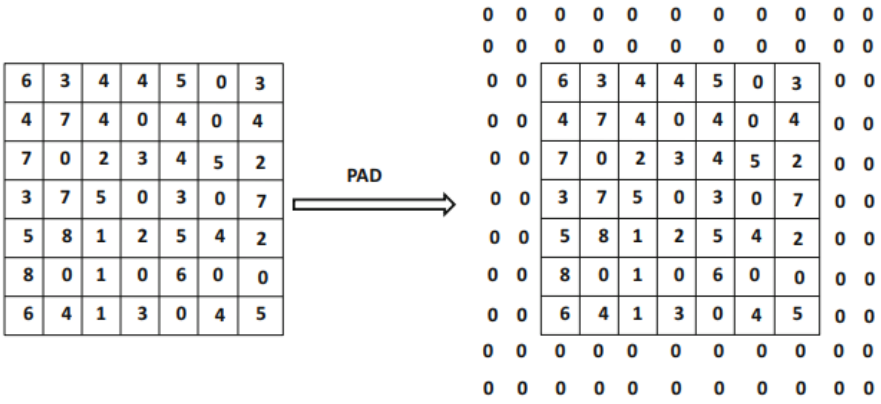


Figura 27. Operación de *padding* con ceros. Fuente: [32]

Otra de las operaciones principales de este tipo de modelos es la **agrupación o *pooling***. Esta operación nos sirve para reducir la dimensionalidad de los mapas de características. Reducir la dimensionalidad es especialmente importante en redes convolucionales donde las redes trabajan con grandes cantidades de datos. Nos permite codificar la información en numerosos mapas de características, pero en dimensiones reducidas, ahorrando coste computacional. Otra de las ventajas de este tipo de operaciones es su funcionalidad a la hora de permitir obtener características de espacios mayores en la imagen. Al reducir la dimensionalidad de los mapas de activación, un píxel de un mapa profundo puede hacer referencia a una característica que ocupa muchos más píxeles en la imagen original.

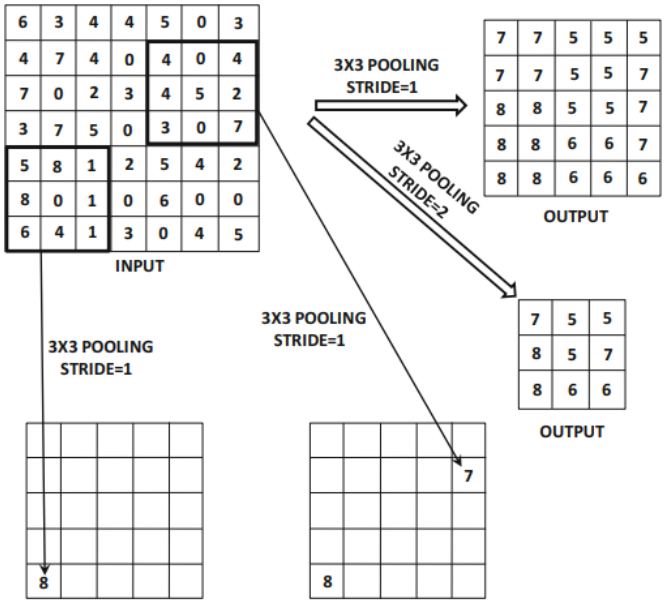


Figura 26. Representación de la operación de *MaxPooling* con un tamaño de filtro de 3x3. Fuente: [32]

Tanto en las operaciones de *pooling* como en las de convolución, existe otro hiperparámetro que puede modificarse para alterar el comportamiento del modelo, el **tamaño de paso o *stride***. Este parámetro nos indica cuántos pixeles se desplaza el filtro antes de realizar una operación. En la figura 24 podemos apreciar la comparación entre establecer un valor de *stride* igual a 1 o igual a 2. En algunos casos, se establece este parámetro en un valor mayor a 1 dentro de operaciones de convolución, sustituyendo a las operaciones de *pooling*.

El último tipo de capa comúnmente usada en este tipo de redes es la capa de activación ReLU. Su funcionamiento en este caso es idéntico al que tiene en las redes anteriormente explicadas. Es común hacer uso de los tres tipos de capas de forma conjunta, formando módulos donde se realiza una convolución, un *pooling* y se aplica la función de activación ReLU.

Una vez se extraen las características de las imágenes, el final de estas redes es diverso. Algunas de ellas se continúan con una serie de capas densas que clasifican las imágenes a partir de las características extraídas. Otras, como veremos más adelante, utilizan esas características para resaltar zonas de la imagen original, aumentando la dimensionalidad de los mapas de activación.

En vista de todo lo anterior, podemos ver la gran variedad posibilidades que nos otorgan las redes neuronales con todas sus posibles combinaciones a la hora de crear distintas estructuras. Esto nos permite trabajar con una gran variedad de imágenes y extraer sus características de una forma similar a cómo lo hace una corteza visual humana, y aplicarlo al control de distintos equipos con la finalidad de facilitar multitud de tareas.

Capítulo 5. Materiales y métodos empleados.

En este capítulo del trabajo se hará una exposición de los materiales utilizados y la metodología que se ha seguido a la hora de entrenar el modelo. En cuanto a los materiales, lo más importante que nos encontraremos en esta sección será la elección de los datos de entrenamiento, ya que son un componente indispensable en el proceso de aprendizaje del modelo y su rendimiento estará sujeto a la calidad de estas muestras. En cuanto a la sección de metodología, en ella se expondrán cada uno de los procedimientos que han sido necesarios para que el funcionamiento del modelo sea correcto, desde el preprocesado de los datos o la elaboración del modelo, hasta la creación de las métricas de rendimiento y las funciones de coste personalizadas para este algoritmo en cuestión

5.1. Materiales

5.1.1. Datos de entrenamiento

Como hemos comentado en la parte teórica, para entrenar una red neuronal son necesarios una elevada cantidad de datos. En muchas ocasiones, la gran cantidad de datos que se precisan puede suponer un problema grave a la hora de entrenar un modelo de red neuronal.

Una opción factible para conseguir estos datos sería crearlos nosotros mismos en el laboratorio mediante simulaciones, o pedírselos a hospitales que realizan estas cirugías diariamente. El problema que nos encontramos en estos casos, incluso consiguiendo una gran cantidad de imágenes de cirugías reales, es el tiempo necesario para realizar el etiquetado de todas las imágenes, lo que impediría el desarrollo del resto del trabajo.

Por ello, en un primer momento es mucho más factible intentar entrenar la red con datos ya etiquetados pertenecientes a *datasets* públicos. El problema que nos encontramos en estos casos es que los *datasets* que contienen imágenes de cirugías laparoscópicas son limitados, y muchos de ellos no están etiquetados al completo, lo que nos limita mucho más la cantidad de información que podemos utilizar para entrenar

Este problema es común en muchos otros campos de la medicina, ya que la dificultad de encontrar datos se acentúa exponencialmente por varios factores. El primero de ellos es la naturaleza de los datos, ya que en muchos casos están protegidos por estrictas políticas de privacidad que hace extremadamente complicado acceder a ellos. Otro inconveniente asociado al origen de los datos en el ámbito médico tiene relación con el etiquetado de los datos. Debemos tener presente que no solamente necesitamos los datos para entrenar los modelos, sino que necesitamos que estén debidamente etiquetados y, en el caso de los datos médicos, este proceso debe haber sido realizado por profesionales sanitarios.

En cuanto a los datos necesarios para este entrenamiento, necesitamos imágenes de cirugías laparoscópicas y cirugías robotizadas que estén etiquetadas para localizar objetos. Esto significa que necesitamos la localización en forma de coordenadas de píxeles de los objetos, lo que limita nuestra búsqueda enormemente. Necesitamos datos que, o nos indican la localización de los centros de las herramientas quirúrgicas para poder buscarlos con nuestro modelo, o nos dan la información sobre los vértices de su *bounding box*, el rectángulo en el que se encierra la herramienta.

Para realizar este trabajo se han utilizado tres conjuntos de datos. El primero de ellos se trata de un conjunto simulado ex-vivo en el Grupo de Robótica Médica del ITAP, perteneciente a la Universidad de Valladolid. Este conjunto de datos está compuesto por 3532 imágenes simuladas de cirugías laparoscópicas. Para ello se utilizó instrumental quirúrgico de la casa comercial Storz, y distintos órganos de origen porcino. La gran ventaja de estos datos es que las herramientas etiquetadas son las mismas que las utilizadas en los distintos proyectos del Grupo de Robótica Médica de ITAP, por lo es mucho más sencillo que la red tenga un buen rendimiento a la hora de implementarla en la arquitectura de control de estos proyectos.

Figura 28. Ejemplos de las imágenes contenidas en el *dataset* creado en el Grupo de Robótica Médica de ITAP. Fuente: Elaboración propia

El problema que nos podemos encontramos con este conjunto de datos es su escasa cantidad de imágenes etiquetadas en las que aparece una herramienta. De un conjunto de 3532 imágenes, únicamente 609 contienen una herramienta. Esto puede generar un problema a la hora de que el modelo aprenda a predecir la existencia de la herramienta, puesto que sería esperable que la red tienda a no hacer predicciones si con ello es capaz de acertar en un gran porcentaje de las imágenes.

Los datos en este caso se tratan de imágenes con sus correspondientes etiquetas en un archivo *xml* en formato *PascalVOC*. Las imágenes poseen información sobre el tamaño de la imagen, la dificultad de la acción, si la pinza aparece truncada por el campo visual de la imagen o no y las coordenadas de los vértices que se encuentran en la zona superior izquierda e inferior derecha del *bounding box*.

Figura 29. Visualización de las imágenes etiquetadas en ITAP con sus *bounding box*. Fuente: Elaboración propia

El segundo de los conjuntos que se ha utilizado es el conjunto de datos *Atlas Dione*, proporcionado por *Roswell Park Comprehensive Cancer Center* [33]. Este *dataset* está

compuesto por 86 videos de simulaciones (unos 910 *clips*) tomados por 10 cirujanos, realizando 6 tareas distintas con el robot quirúrgico DaVinci. 99 de esos *clips* están etiquetados en cada *frame*, aportando información sobre los tipos de herramienta que aparecen en las imágenes y la dificultad de la acción que se está realizando, así como la posición de los vértices del *bounding box* de una forma idéntica al *dataset* anterior. Estas etiquetas se proporcionan igualmente en un archivo *xml* en formato *PascalVOC*.

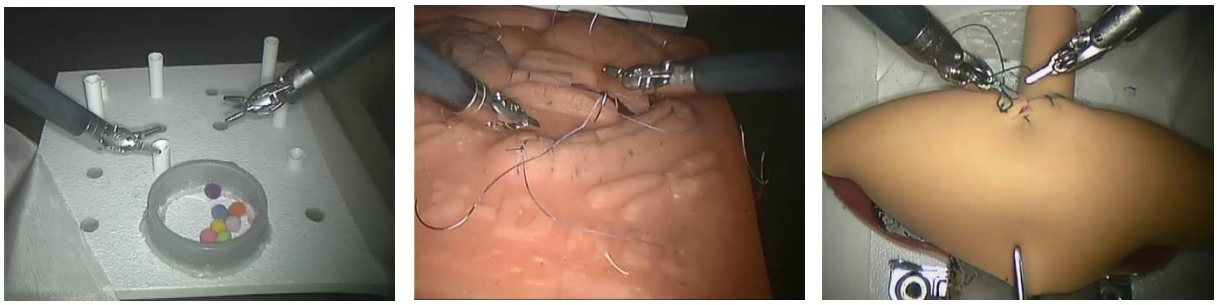


Figura 30. Distintos ejemplos de las actividades registradas en el *dataset* AtlasDione.

La gran ventaja que nos proporciona este *dataset* es su cantidad de datos etiquetados, ya que contamos con más de 22.000 imágenes en 6 ámbitos muy distintos, por lo que nos proporciona, aparte de su cantidad, una gran variabilidad de imágenes. Este aspecto es fundamental a la hora de entrenar nuestro modelo, ya que su rendimiento y su capacidad de generalización se verá limitada en gran parte por estos aspectos. Además, nos da la capacidad de tener un conjunto de validación y de test robusto, con distintos tipos de actividades y entornos, y un número de imágenes suficiente para tener una mínima confianza de que las métricas que nos aporta el modelo son fiables.

El problema que nos encontramos con estos datos es el hecho de que se tratan de simulaciones realizadas con herramientas del robot quirúrgico DaVinci, cuyas herramientas no tienen las mismas características que las utilizadas en nuestro laboratorio. Esto podría causar problemas de generalización a la hora de utilizar este modelo para el control de los robots utilizados en ITAP con su tipo de instrumentos. Además, al no haber imágenes extraídas de cirugías reales, el modelo podría empeorar su rendimiento a la hora de enfrentarse a este tipo de entornos, con imágenes ligeramente distintas y artefactos a los que nunca se ha enfrentado.

Para finalizar, el último conjunto de datos utilizado en este trabajo ha sido *EndoVis'15 Dataset* [34]. Este conjunto de datos ha sido registrado con motivo de la competición *Endoscopic Vision Challenge,* dentro del subapartado *Instrument Segmentation and Tracking.* Este set de datos está compuesto por 4535 imágenes, de las cuales 180 están etiquetadas. Estas imágenes se corresponden con 4 simulaciones de cirugías ex-vivo con instrumentos rígidos. Las etiquetas de estas imágenes se proporcionan mediante un archivo *csv*, que contiene el nombre de cada *frame*, y las coordenadas del centro de la herramienta, que está situado entre la parte rígida y la herramienta del instrumental quirúrgico.

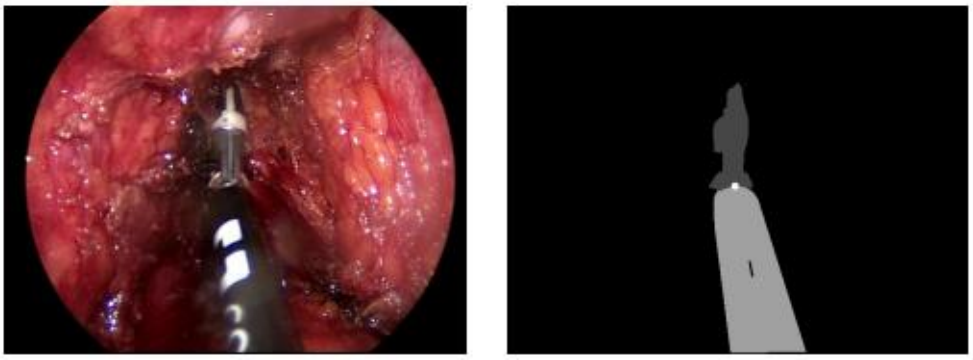


Figura 31. Imagen original (izquierda) y segmentación de la herramienta (derecha) en el conjunto de datos *EndoVis'15*. Fuente: [34].

Como se puede ver en la figura 31, el centro de la herramienta se localiza en un punto distinto al que hemos considerado hasta ahora. Hasta ahora el centro de la herramienta se situaba en el centro del *bounding box* que encierra la herramienta, por lo que aproximadamente queda en el centro de la parte distal del instrumental quirúrgico. En el caso de este *dataset*, se considera el centro de la herramienta el límite entre la parte rígida del instrumental quirúrgico y la herramienta, que se encuentra en la zona más distal de la misma. Este cambio a la hora de interpretar el centro de la herramienta en los datos etiquetados nos podría generar dificultades a la hora de entrenar nuestro modelo y poder comparar el rendimiento en los distintos conjuntos de datos.

Otro inconveniente que no encontramos con este set de datos es la escasa información que tenemos etiquetada, ya que únicamente tenemos 180 imágenes, que comparado con las más de 21.000 que teníamos en el conjunto de datos anterior, se nos hacen escasas para entrenar un modelo.

Las ventajas que nos encontramos en este conjunto de datos respecto de los anteriores se basan en el origen de las imágenes. Comparándolos con el primero de los conjuntos de datos, la principal ventaja que tienen estas imágenes tiene su origen en que son extraídas de cirugías ex-vivo simuladas realizadas por profesionales sanitarios, de forma que se ajustan con mayor fidelidad a la realidad clínica. Respecto al segundo de nuestros conjuntos de datos, la principal ventaja que nos encontramos, aparte de un mayor realismo en las intervenciones, es que los instrumentos utilizados son mucho más parecidos a los que poseemos en el laboratorio, lo que nos puede dar una visión de cuál es el rendimiento de nuestro modelo con este tipo de instrumentos en casos reales.

En definitiva, como hemo podido ver, obtener datos en el ámbito médico es una tarea complicada por diversos motivos, especialmente por la privacidad de los datos personales de los pacientes. En concreto, en el ámbito de la cirugía, nos encontramos también con la escasa cantidad de datos debidamente etiquetados por profesionales sanitarios, y con la diversidad de aproximaciones que hay a la hora de etiquetarlos, con diferencias entre los tipos de actividades a realizar o en la posición donde se establece el centro de la herramienta. Este factor puede ser un limitante a la hora de entrenar un modelo de red neuronal que sea capaz de localizar con exactitud las herramientas. Por ello en este trabajo se utilizarán los tres conjuntos de forma combinada, cada uno con un objetivo distinto, esperando que esto mejore su rendimiento

5.2. Metodología

En esta sección se desarrollarán todos los pasos que se han seguido para crear el modelo en su totalidad, desde la forma de obtener los datos desde los ficheros o su preprocesado, hasta las métricas de rendimiento o las funciones de pérdidas. También se explicará el modelo elegido y sus diferentes partes y funciones.

5.2.1. Lenguaje de Programación y librerías

Para llevar a cabo el desarrollo de este proyecto es necesario decidir cuál será el lenguaje de programación que se adecuará de forma óptima a él. En este caso, el lenguaje que se ha utilizado para programar todo el algoritmo es *Python*. Este lenguaje se ha elegido debido a su mayor facilidad y simplicidad de programación respecto de otros lenguajes como C o C++, y a la existencia de un gran número de librerías dedicadas al procesamiento de datos como puede ser *tensorflow* o *pyTorch*. Estos dos factores han derivado en que el lenguaje *Python* es uno de los más populares en aplicaciones de inteligencia artificial y *deep learning*.

Además de la elección del lenguaje de programación, ya hemos indicado que *Python* cuenta con una gran cantidad de librerías dedicadas al procesamiento de datos y a inteligencia artificial, por lo que es necesario elegir algunas de ellas. En este caso se ha elegido *tensorflow*, y más concretamente, la librería *keras*.

Tensorflow es una biblioteca que nace a partir de un proyecto llevado a cabo por *Google* llamado *Google Brain*. Este proyecto tenía como finalidad el desarrollo de herramientas de aprendizaje automático para poder implementar en sus productos. Una vez terminado, los mismos ingenieros decidieron desarrollar un entorno de código abierto capaz de facilitar a cualquier persona el desarrollo de modelos de aprendizaje automático, siendo compatible con una amplia gama de dispositivos y *harware*. Esta librería trabaja mediante grafos de nodos y operaciones con tensores. Estas operaciones van desde operaciones básicas como sumas o restas, hasta funciones más complejas como funciones de activación $ReLU$ o *Softmax* [35].

Además, para ser más concisos, se ha utilizado la librería *keras* que nos proporciona *tensorflow*. Esta librería nos proporciona funciones de alto nivel para implementar modelos de aprendizaje automático y profundo que simplifican en gran medida la tarea de programar el algoritmo, reduciendo en gran medida las líneas de código necesarias y haciéndolo más accesible.

Por último, el entorno de programación elegido para integrar todo el proyecto en lenguaje *Python* ha sido *PyCharm*. Este entorno tiene varias propiedades que facilitan el desarrollo del proyecto. La primera de ellas es que es un entorno que funciona por proyectos, y cada uno de ellos puede utilizar un intérprete, lo que aporta una gran versatilidad a la hora de mover proyectos entre distintos equipos, necesario en este caso ya que los modelos son entrenados en otro equipo con mayores prestaciones. Además, posee un instalador de paquetes permite hacer una gestión de estas librerías de forma mucho más eficiente, simplificando la elección de versiones y compatibilidades.

5.2.2. Modelo inicial Anchor Free Hourglass

Para comenzar con el apartado de la metodología que se ha seguido a la hora de realizar este trabajo, comenzaremos con el modelo que se ha creado y entrenado, y en torno al cual se desarrollará el resto del trabajo. Se considera imprescindible comenzar explicando qué modelo se utilizará ya que, gran parte del trabajo realizado posteriormente como el preprocesado de datos o la creación de distintas métricas han estado sujetos totalmente al tipo de modelo que se pretende entrenar.

Como hemos mencionado en apartados anteriores, es fundamental que el modelo cumpla con algunos requisitos para que sea realmente útil en el ámbito en el que se va a desarrollar. El primero de ellos es la precisión de la predicción. Necesitamos un modelo fiable en las predicciones que genere, ya que la cámara se moverá en el campo quirúrgico con esta información. En caso de que no seamos capaces de generar un modelo suficientemente fiable, nos podríamos encontrar con un modelo que genera precisamente los problemas que queremos evitar, que son interrupciones en las operaciones y oclusiones de los instrumentos quirúrgicos. Otra de las características importantes que debe tener nuestro modelo es la velocidad en la predicción. El objetivo de este modelo es poder implementarlo en una arquitectura de control capaz de aportarle los datos de las coordenadas en la imagen para que el robot realice la transformación necesaria de esas coordenadas a movimiento y mantenga la herramienta en la imagen. Es por ello que el modelo debe ser capaz de aportar información suficiente al robot en tiempo real con el objetivo de que se pueda mover de acuerdo con la posición de las herramientas.

Para elegir el tipo de modelo, inicialmente se ha realizado una búsqueda bibliográfica para situarnos en cuanto a los distintos modelos existentes y así poder hacer una elección más precisa. Como se ha mencionado en el capítulo 2, podemos distinguir los modelos en dos tipos principales. El primero de ellos son los modelos en dos etapas. El principal inconveniente de estos modelos se basa en su naturaleza modular. En estas redes, las distintas partes que las forman deben haber sido entrenadas por separado, lo que ralentiza mucho este proceso y, además, hace necesario un conjunto de datos etiquetados mucho mayor, y de varios tipos distintos, lo cual es uno de los principales limitantes en nuestro ámbito. Además, al estar formado por varias partes, su tiempo de inferencia también es mayor.

Por este motivo, en este trabajo se ha centrado la búsqueda en algoritmos de detección en un paso. El modelo más representativo de este tipo de algoritmos en la localización de objetos son las arquitecturas YOLO (*You Only Look Once*). Existen múltiples versiones de estos modelos, con distintas características y requerimientos computacionales, según la potencia necesaria para entrenarlas.

En trabajos anteriores realizados en el Grupo de Robótica Médica, se había utilizado un modelo YOLOv3 para realizar esta misma tarea, con resultados limitados en comparación con la aplicación en la que se pretende aplicar este algoritmo. Es por ello que se ha buscado un modelo capaz de mejorar el rendimiento de este modelo, preservando la velocidad de inferencia.

Mediante esta búsqueda, se ha encontrado un modelo basado en módulos tipo reloj de arena o *hourglass*, capaz de detectar la posición de la herramienta sin la necesidad de definir unos tamaños predeterminados de cajas, lo que sí es necesario en los modelos YOLO. Este

modelo ha demostrado tener un mejor rendimiento que YOLOv3 con una velocidad de inferencia muy similar, por lo que se ha elegido para ser un modelo base de cara al desarrollo [36].

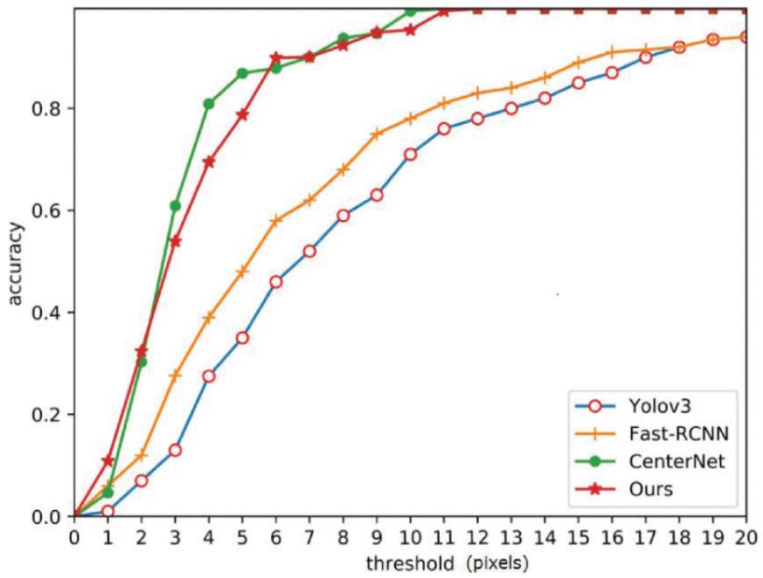


Figura 32. Rendimiento del modelo encontrado con respecto a otros tres modelos comúnmente utilizados Fuente: [36].

Este modelo se basa en dos módulos *hourglass* dispuestos en serie. Estos módulos fueron implementados por primera vez por *Newell et al* [37]. en la detección de las orientaciones y articulaciones humanas. Este módulo tiene como objetivo la detección de características a distintas escalas. Se trata de un tramo de disminución de dimensionalidad de los mapas de características y otro de aumento de estos, de forma similar al modelo U-Net o a la estructura de los *autoencoders*. Se diferencia del modelo U-Net en las conexiones de escape, que en este caso tiene una etapa intermedia de procesado. La mayor diferencia de este módulo respecto de otras arquitecturas tradicionales radica fundamentalmente en que tiene una mayor simetría entre la fase de reducción y aumento de dimensionalidad de los mapas de características. Además, en este módulo se reemplazan operaciones típicas en fases de aumento de dimensionalidad comunes en otras arquitecturas, como la convolución inversa, por una capa de *upsampling* mediante el algoritmo *k-means*, lo que supone un menor coste computacional al tratarse de capas no entrenables [37].

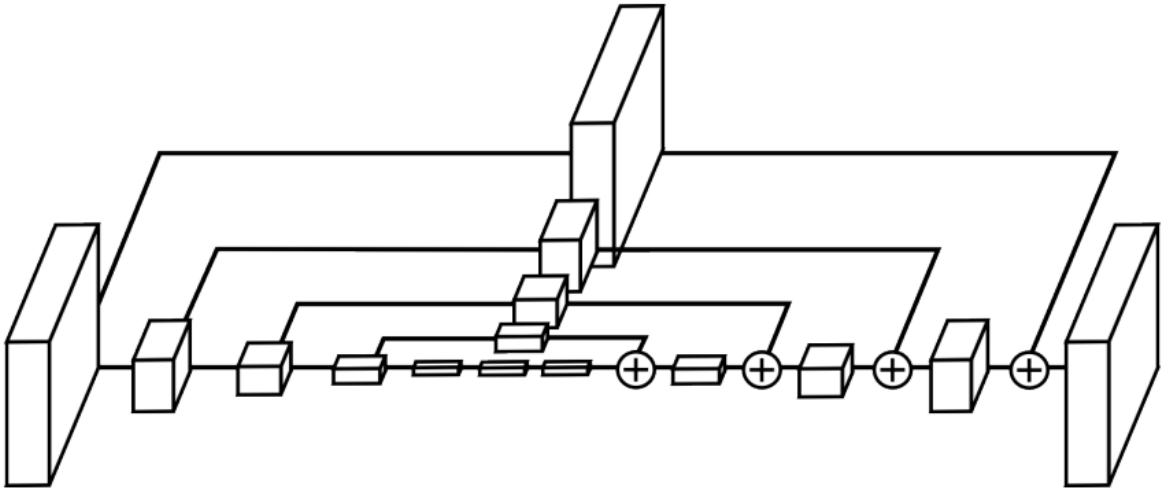


Figura 33. Estructura básica de un módulo *Hourglass*. Cada bloque se refiere a un módulo residual en el modelo original. Fuente: [37].

La estructura básica del modelo consta de un módulo de convolución 7x7, y un módulo residual, que combinados reducen la dimensionalidad de las imágenes en un factor de 4. A continuación, las imágenes entran en cada uno de los dos módulos *hourglass*, reduciendo su dimensionalidad para posteriormente, volver a aumentarla de la forma explicada anteriormente.

En este caso, los módulos residuales que se utilizan en el modelo original se sustituyen por módulos *fire*. Estos módulos son ligeramente diferentes a los módulos residuales, sobre todo en el coste computacional que conllevan. Las convoluciones 3x3 que se realizan en los módulos residuales buscan encontrar relaciones de forma combinada en las dos dimensiones de la imagen y, además, en las dimensiones de los canales. Los módulos *fire* desglosan estas operaciones para buscar en primer lugar relaciones entre canales y, en segundo lugar, relaciones espaciales, lo que les hacen más eficientes. Para ello, utilizan en primer lugar una convolución 1x1, que reduce el número de canales, seguida de una bifurcación que realiza de forma paralela una convolución 1x1, similar a los módulos residuales, y una convolución separable. Esta operación es más eficiente que una convolución convencional ya que a cada canal se le asignan unos filtros, evitando las combinaciones lineales de filtros aplicados a distintos canales [38].

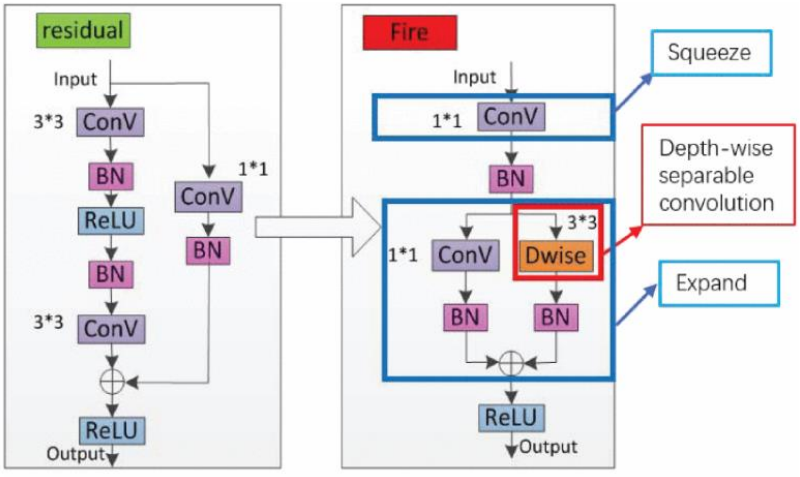


Figura 34. Diferencias entre el módulo residual convencional y un módulo *fire*. Fuente: [36].

En los módulos *hourglass* se reduce la dimensionalidad de los mapas de activación en dos ocasiones más. Para llevar a cabo esta reducción no se requiere de ninguna capa de *pooling*, sino que se utiliza un tamaño de paso de 2 en la operación de convolución, ahorrando parámetros al modelo. A continuación, la salida se introduce en una Red Neuronal Convolucional *(CNN)* de propagación hacia adelante simple para interpretar las características extraídas por el módulo *hourglass.*

Una vez se han extraído las características después del primer módulo *hourglass*, se llevan los mapas anteriores al módulo hasta su salida, con una convolución 1x1 y una capa de *batch-normalization* a modo de supervisión. Finalmente, los mapas vuelven a entrar en un segundo módulo *hourglass* idéntico al primero y la inferencia se realiza con una *CNN* idéntica a la explicada en el primer módulo. En este caso, las salidas de las tres ramas de la *CNN* no se suman, sino que cada una haría referencia a una característica (centro de la herramienta, *localización del bounding-box* y *offset*). En nuestro caso, solamente se busca localizar el centro de la herramienta, por lo que, aunque el modelo está creado para obtener estas tres características, solo se ha utilizado la salida relativa al centro de la herramienta con la intención de ahorrar parámetros en el modelo.

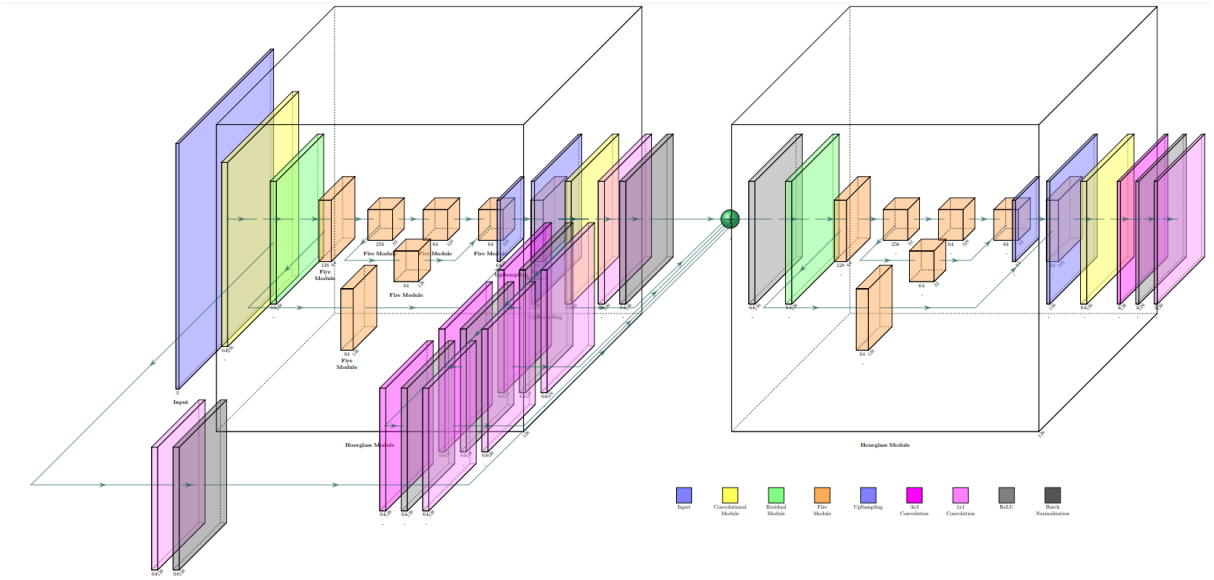


Figura 35. Representación de la versión inicial del modelo propuesto. Fuente: Elaboración propia. Ver Anexo I

Finalmente, la salida que proporciona el modelo son dos imágenes monocromáticas, es decir, únicamente tienen un canal de intensidad, a diferencia de las RGB *(Red Green Blue)* que tienen tres. La función de activación de la última capa es una función sigmoide, lo que otorga a cada píxel la probabilidad de pertenecer o no al centro de la herramienta. Por ello, cada una de las imágenes de salida hace referencia al mapa de calor del centro de las herramientas con una distribución similar a una función *gaussiana* en dos dimensiones, cuyo valor máximo hará referencia al centro de la herramienta. Esta forma en la salida del modelo se impone mediante las etiquetas que generamos, lo cual se explicará en un apartado posterior.

5.2.3. Lectura y preprocesado de los datos.

Como se ha mencionado a lo largo de todo este trabajo, los datos son una parte fundamental a la hora de entrenar un modelo como el nuestro. Es imprescindible tener un conjunto de datos con la cantidad suficiente de muestras para poder entrenar el modelo de forma robusta y una variabilidad mínima que le permita generalizar a otras situaciones y evitar el *overfitting*.

El problema que nos encontramos en muchas ocasiones es que los datos tienen distintos formatos y hay que adaptarlos para nuestro modelo. Estas diferencias en el caso de las muestras que actuarán como entrada en la red no son muy significativas, ya que, normalmente, se trata de imágenes RGB cuya variación radica fundamentalmente en sus dimensiones.

En el caso de las etiquetas el problema tiene una mayor complejidad ya que, en cada conjunto de datos, la información puede ser proporcionada en distintos formatos, lo que requiere de programas diferentes para poder ser leídas por el modelo e implementadas a un conjunto de datos adecuado para entrenarlo. Además, en nuestro caso en concreto, los datos que nos proporcionan las etiquetas en los archivos se refieren a las coordenadas de los centros de la herramienta. En cambio, lo que buscamos que genere la red es un mapa de calor cuyos valores máximos se correspondan con los centros predichos de la herramienta. Por todo esto, hay que transformar las coordenadas que se nos proporcionan en imágenes del tamaño de las capas de salida de la red con características similares a las que esperamos que nos proporcione el modelo.

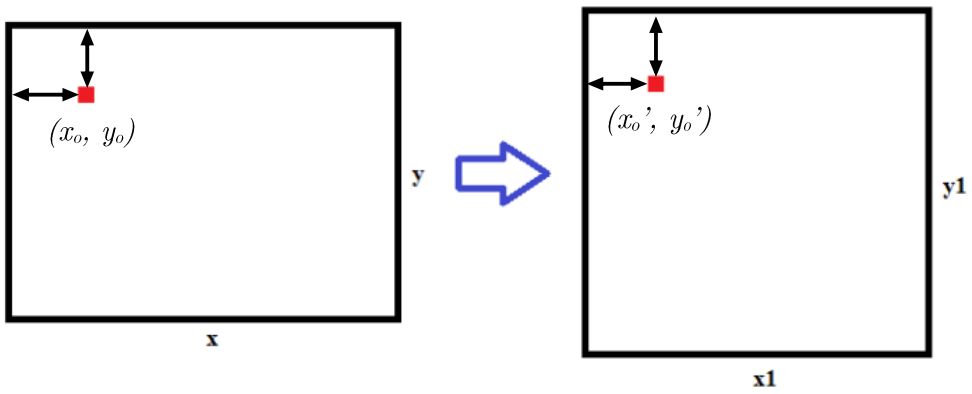
Para comenzar con el proceso de preprocesado de los datos, es importante mencionar la limitación a la que nos hemos enfrentado respecto a la memoria RAM. Para desarrollar este proyecto se ha usado un equipo con 8 Gb de RAM. Únicamente se ha utilizado un equipo con mayores prestaciones, como una GPU dedicada Nvidia RTX 3060 Ti para los entrenamientos cuando tanto los programas previos como el modelo funcionaba correctamente, debido a la reducida disponibilidad este equipo en el laboratorio. Esto supone que todas las tareas de desarrollo de los programas y pruebas de funcionamiento, que han servido para asegurar su correcto funcionamiento a la hora de trasladarlo al equipo capaz de entrenar el modelo, se han llevado a cabo en un equipo con características muy limitadas, lo que ha condicionado la funcionalidad de muchos de los scripts. En consecuencia, por ejemplo, ha sido necesario desarrollar un programa capaz de cargar únicamente en la memoria RAM las imágenes del lote o *batch* de imágenes que va a utilizar la red en ese paso, ya que, de lo contrario, esta sería insuficiente para cargar también todos los pesos del modelo.

El *script* que realiza este conjunto de funciones está compuesto por tres clases fundamentales, una para leer cada tipo de conjunto de datos. En estas clases puedes elegir el tamaño de *batch*, los directorios de donde queremos extraer imágenes y etiquetas, y el tamaño de entrada y salida de las imágenes de la red.

La primera de estas clases es para el conjunto de datos Atlas Dione. Cada vez que el modelo principal haga una llamada al objeto de esta clase, este tomará un número de imágenes correspondiente con el tamaño del *batch*. Estas imágenes serán redimensionadas al tamaño de entrada de la red, que en el caso del modelo inicial es 512x512 y serán normalizadas antes de mandarse al modelo. En cuanto a las etiquetas, este conjunto de datos las proporciona en un archivo *xml*, en formato *PascalVOC*. Este formato se trata de un archivo de código con distintos

campos o variables anidadas, por lo que hay que irlo recorriendo para obtener los valores numéricos de las coordenadas. Además de las coordenadas, también extraemos el tamaño de las imágenes originales para poder hacer las transformadas que explicaremos a continuación.

Una vez tenemos las coordenadas, se transforman para que se ajusten a la imagen redimensionada que usamos para entrenar la red. Para ello, se utiliza un factor de reescalado en cada uno de los dos ejes de la imagen que dependerá de la imagen original de entrada, de forma que el algoritmo se adapta a cualquier tamaño de entrada, lo que lo hace universal para cualquier conjunto de datos, sea cual sea su origen.



$$R_x = \frac{\mathrm{x1}}{\mathrm{x}}$$
$$\mathrm{R_y} = \frac{\mathrm{y1}}{\mathrm{y}} \tag{13}$$

$$\mathrm{x_o'} = \left\lfloor \frac{x_o}{R_x} \right\rfloor$$
$$y_o' = \left\lfloor \frac{y_o}{R_y} \right\rfloor \tag{14}$$

Una vez se ha transformado el tamaño de la imagen y las coordenadas del centro de la herramienta en base al nuevo tamaño de imagen, se les dan a los píxeles donde se encuentran los centros un valor de 1, mientras que el resto permanecen con una intensidad nula. Posteriormente, se aplica un filtro *gaussiano* para formar el mapa de calor con el que entrenaremos el modelo. El filtro tiene propiedades personalizadas para cada herramienta, estableciendo el valor de la desviación estándar de manera acorde al tamaño de cada una de ellas. Para ello se ha tomado el tamaño de la herramienta, aproximándolo como el máximo entre su longitud en los ejes vertical y horizontal, y se ha dividido entre 3, de forma similar a lo que realizó *Hei Law et al.* [39].

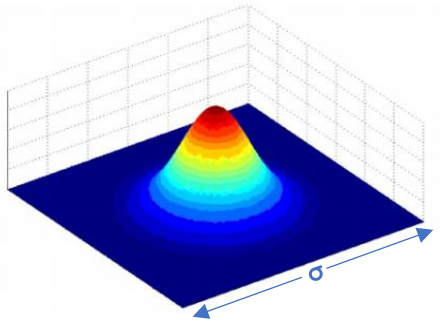


Figura 38. Forma del filtro *gaussiano* aplicado a la imagen para generar las etiquetas. Fuente: [46].

Finalmente, obtenemos dos mapas de calor, uno para cada una de las dos herramientas, cuyo máximo se refiere al centro de la herramienta y cuyo tamaño se relaciona con el tamaño de la herramienta. En caso de querer obtener en algún momento el *bounding box* de la herramienta, sería interesante adaptar la desviación estándar del filtro en los dos ejes al tamaño de la herramienta en los dos ejes, ya que el modelo puede hacer una predicción de esta medida en base a las propiedades del mapa de calor entorno al centro de la herramienta.

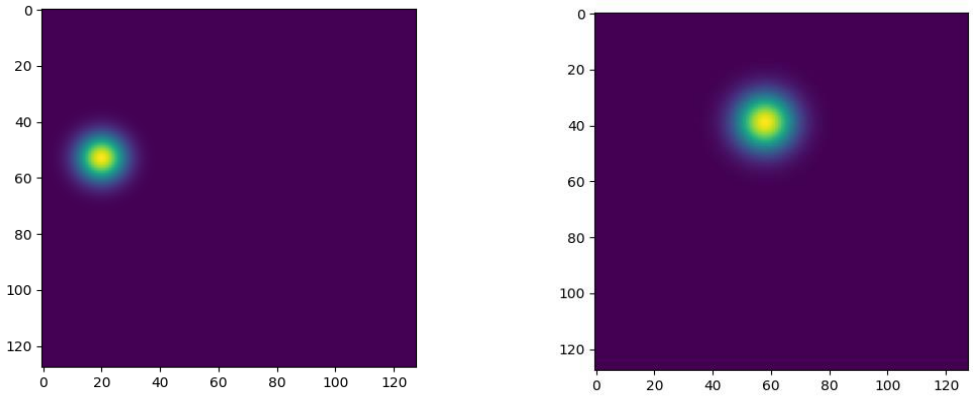


Figura 37. Mapas de calor para una imagen con dos herramientas. Fuente: Elaboración propia

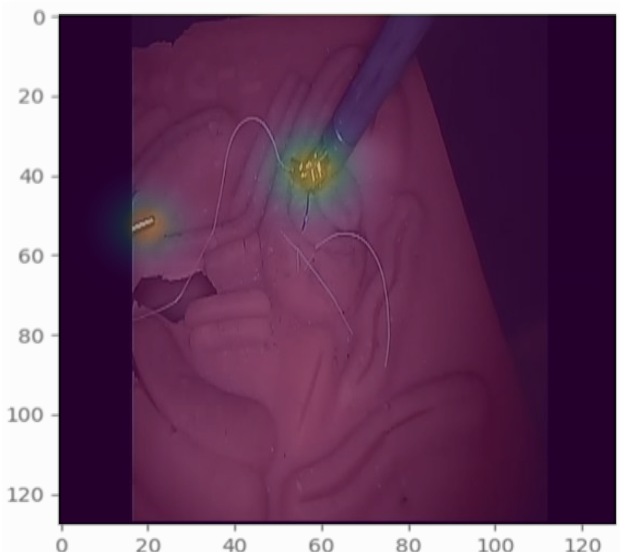


Figura 36. Correspondencia de la etiqueta de la imagen con la imagen reescalada que sirve de entrada para el modelo. Fuente: Elaboración propia.

En segundo lugar, se ha realizado una clase prácticamente idéntica para el conjunto de datos obtenido en ITAP. Los pasos que sigue este algoritmo son idénticos a la anterior, y las únicas diferencias entre ambas clases radica en que el archivo *xml* que contiene las características de la imagen tiene datos que se asignan de forma distinta, como el ancho y alto de la imagen, lo que condiciona también los factores de reescalado y la transformación de coordenadas.

En el caso de este conjunto de datos, además, se adaptó otra clase para poder leer de forma conjunta imágenes del conjunto Atlas Dione e imágenes de ITAP, debido a la escasa cantidad de imágenes que incluye el segundo de ellos. Esta modificación se realizó con la intención de hacer un entrenamiento combinando ambos conjuntos de datos. Esta elección se realiza atendiendo a las propiedades del alto y ancho de imagen, ya que depende el formato en el que se encuentren etiquetados, podemos diferenciar un conjunto u otro.

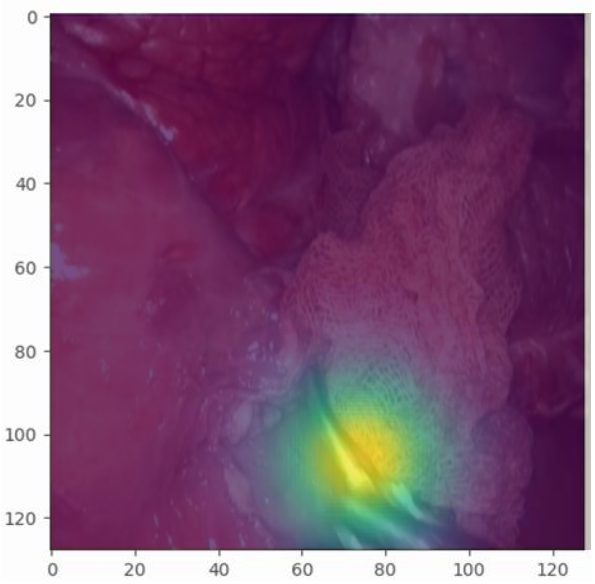


Figura 39. Correspondencia de la etiqueta creada con la imagen redimensionada. Fuente: Elaboración propia.

Por último, se ha creado una última clase, para poder leer y cargar los datos desde el fichero correspondiente al modelo, dedicada al conjunto de datos *EndoVis'15*. Como se ha mencionado anteriormente, este conjunto de datos nos proporciona las etiquetas en un formato totalmente distinto a los dos anteriores. Mientras que en los casos anteriores se nos proporcionaba un archivo *xml* con una estructura definida conforme al estándar *Pascal VOC*, en este caso los datos se encuentran en un archivo *csv*, junto con un documento explicativo que indica a qué se refiere cada una de las columnas de la tabla.

En este caso, el archivo contiene las coordenadas del centro de la herramienta, que en este caso es importante recordar que no se encuentra exactamente en el centro del *bounding box* que encierra la herramienta, sino que se encuentra en la zona donde se une la herramienta y la zona rígida del instrumental.

Para leer este archivo, el procedimiento ha sido ligeramente distinto a los anteriores. Al descargar este archivo, nos encontramos con cuatro directorios distintos que contienen imágenes numeradas, las máscaras de las imágenes que se utilizarían para realizar tareas de segmentación

de imagen, y el archivo *csv* con las coordenadas de los centros de las imágenes. Para poder leer las imágenes de los cuatro directorios y poder generar el conjunto de datos de forma que a cada imagen le corresponda su etiqueta se han realizado varios procedimientos.

En primer lugar, antes de cargar las imágenes sin procesar se elaboró un *script* cuyo objetivo es cambiar el nombre de todas ellas, añadiendo un *'X_'* a cada una de las imágenes, siendo X el número del directorio entre 1 y 4. De esta forma, al cargar las imágenes en el modelo ya podemos distinguir a cuál de los directorios pertenece cada una de ellas. Este paso previo es clave para poder emparejar posteriormente cada imagen con su etiqueta, al igual que para indicar en la clase a cuál de las cuatro ubicaciones debe dirigirse para buscar la imagen.

En segundo lugar, se realizó una tarea similar en los archivos *csv* que contienen las coordenadas de la imagen. Para ello, se utilizó la librería de Python *Pandas*, que contiene librerías optimizadas para leer con facilidad archivos *csv* y los guarda en objetos de tipo *dataframe*, con los que posteriormente se pueden realizar con facilidad distintas transformaciones y operaciones. Este archivo puede contener cinco o nueve columnas, dependiendo del número de herramientas que se puedan ver en cada *frame*. La primera de las columnas contiene los nombres de cada una de las imágenes a las que hace referencia. Esta columna nos servirá para localizar la columna que contenga el nombre de la imagen que se va a cargar al modelo, por lo que necesitamos que los nombres de las imágenes y de la tabla sean iguales.

Para ello, una vez generado el objeto de tipo *dataframe*, se le añade a cada uno de los nombres de las imágenes el mismo prefijo que se ha añadido con anterioridad a las imágenes, correspondiendo con el directorio al que pertenecen. Una vez hemos cambiado los nombres, es sencillo buscar la fila en el *dataframe* y poder obtener las coordenadas del centro de las herramientas. Los pasos posteriores se llevaron a cabo de forma idéntica a los anteriores sets de datos.

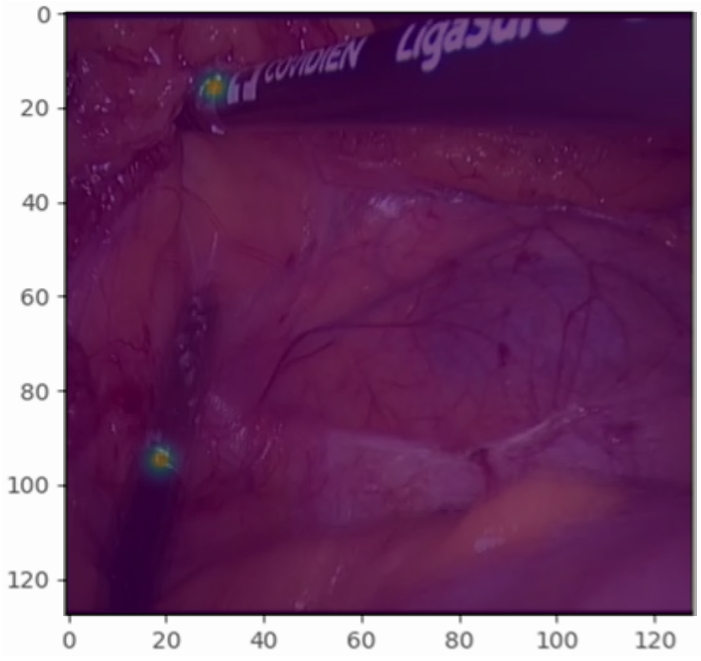


Figura 40. Correspondencia de la etiqueta generada con la imagen redimensionada original. Fuente: Elaboración propia.

En este caso se pueden ver dos aspectos fundamentales que diferencian este tipo de datos con los anteriores. El primero de ellos se refiere al tamaño de los mapas de calor, que en este caso podemos ver que no se adaptan a la herramienta, sino que tienen el mismo tamaño para todas ellas. Esto se debe a que las etiquetas no nos proporcionan el tamaño del *bounding box*, sino que nos indican directamente la localización del centro de la herramienta. Sería interesante de cara a aplicaciones futuras usar las máscaras de segmentación para adaptar este tamaño, ya que a partir de estos datos si podríamos saber el tamaño de la herramienta. Otro aspecto diferencial es la localización del centro del mapa de calor, que en este caso no situamos en el centro de la herramienta, sino que, como hemos mencionado anteriormente, se situa entre la herramienta y la parte rigida del instrumental. En el futuro también podría ser interesante obtener el centro a partir de algoritmos como el cálculo del centro de masas a partir de las máscaras de segmentación.

5.2.4. Funciones de coste del modelo

Un aspecto fundamental a la hora de entrenar un modelo de *deep* learning es la función de coste utilizada. Dependiendo de la función seleccionada el modelo tendrá un comportamiento u otro, por lo que es fundamental elegir una función que obligue al modelo a aprender en base al funcionamiento deseado.

En este caso se debe recordar que el objetivo del modelo es generar dos mapas de calor donde los píxeles con valor máximo de cada uno de ellos hagan referencia a los centros de cada una de las herramientas. Realmente, nos estamos enfrentando a un problema de clasificación, similar al que nos podemos encontrar con modelos de segmentación de imagen, donde a cada píxel le vamos a proporcionar una probabilidad de ser el centro de la herramienta.

El problema que nos encontramos con este tipo de modelos es que, al contrario que en tareas de segmentación, existe un escaso número de puntos donde la predicción deba ser positiva. En nuestro caso únicamente tendrán valores distintos de cero los píxeles muy próximos al centro, por lo que sería previsible que la red tienda a realizar un entrenamiento donde todos los píxeles valgan cero, lo que supondría que acertaría en un gran porcentaje de ellos.

Con la intención de evitar este fenómeno, se ha elegido la función de pérdidas *focal loss,* utilizada por *Lin et al* [40]. Esta función se trata de una interpretación derivada de la entropía cruzada utilizada para clasificación binaria definida por la siguiente ecuación 15, siendo y el valor real y p la probabilidad de pertenecer a $y = 1$, predicha por el modelo.

$$\mathrm{CE}(p, y) = \begin{cases} -\log(p) & \text{if } y = 1 \\ -\log(1-p) & \text{otherwise.} \end{cases} \tag{15}$$

El problema que nos encontramos con esta función es que para predicciones que son fácilmente clasificadas, con un valor de $p >> 0.5$, el valor de la función de pérdidas no es despreciable y si existe un gran número de instancias fácilmente clasificables de una clase, comparado con las de la otra, puede ocurrir que la suma de esos pequeños valores solape el valor de pérdidas generado por la clase menos frecuente, facilitando que el modelo aprenda a clasificar únicamente la clase más frecuente.

Para solucionar este suceso, se puede introducir un peso que pondere el valor de la función de pérdidas en una y otra clase. Este peso, que llamaremos α se puede ajustar atendiendo a distintas perspectivas, como puede ser la inversa de la frecuencia que supone cada una de las clases, o un hiperparámetro ajustable por validación cruzada en el modelo. Este ajuste se define con la siguiente ecuación.

$$\mathrm{CE}(p_t) = -\alpha_t \log(p_t).$$

$$p_t = \begin{cases} p & \text{if } y = 1 \\ 1-p & \text{otherwise,} \end{cases} \tag{16}$$

Aunque la aproximación anterior es capaz de balancear la contribución a la función de pérdidas de las instancias de una y otra clase, no es capaz de diferenciar las instancias fácilmente clasificables y las más complicadas. Para solucionar este problema, se desarrolla la función *focal loss*. Esta función se centra en dar menos valor al valor de la función de las instancias fácilmente clasificables y centrar al modelo en entrenar las instancias difícilmente clasificables. Para ello, se añade al modelo un factor de modulación adaptativo con un parámetro arbitrario γ mayor o igual a cero.

$$\mathrm{FL}(p_t) = -(1-p_t)^{\gamma}\log(p_t) \tag{17}$$

En el siguiente gráfico proporcionado en el artículo original y que se muestra en la figura 41, podemos ver la diferencia que se produce en la función de coste a medida que se aumenta el parámetro γ.

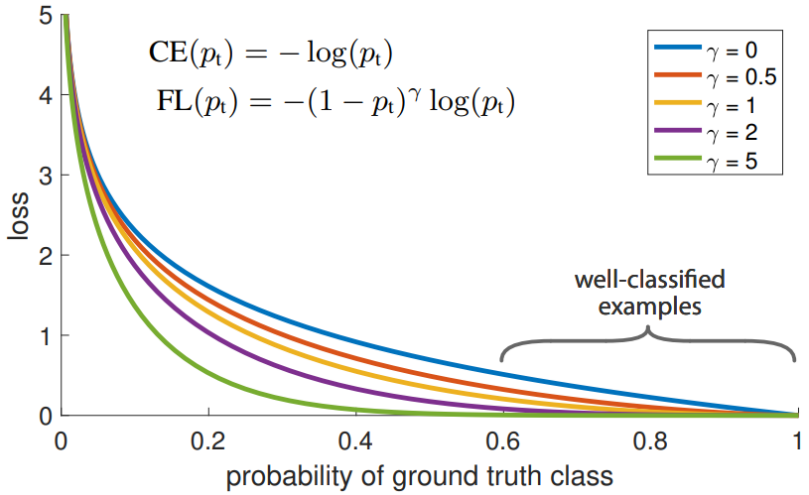


Figura 41. Representación de la variación de la función de pérdidas al variar el parámetro γ. Fuente: [40].

A partir de esta expresión, se ha implementado en el modelo una función de pérdidas *focal loss* modificada según siguiendo el modelo propuesto por *Law et al* [39]. donde incluyen un factor de modulación más que depende del valor real del píxel. Esto, como se puede ver en la ecuación 17, reduce la penalización de esta función entorno al punto del centro de la herramienta, lo que es especialmente útil con etiquetas formadas por filtros *gaussianos*, donde hay un único punto con valor unitario y el resto contienen valores cercanos a 1 que si fuesen clasificados como tal serían gravemente penalizados [39].

$$L_{det} = \frac{-1}{N}\sum_{c=1}^{C}\sum_{i=1}^{H}\sum_{j=1}^{W}\begin{cases}(1-p_{cij})^{\alpha}\log\left(p_{cij}\right) & \text{if } y_{cij}=1\\ (1-y_{cij})^{\beta}\left(p_{cij}\right)^{\alpha}\log\left(1-p_{cij}\right) & \text{otherwise}\end{cases} \tag{18}$$

5.2.5. Métricas de rendimiento utilizadas para evaluar el modelo

Para evaluar el rendimiento de nuestro modelo se han desarrollado dos métricas de rendimiento distintas. Estas métricas están basadas fundamentalmente en la distancia que existe entre la predicción que realiza el modelo del centro de la herramienta y el centro real.

En la literatura existen otro tipo de métricas más extensamente utilizadas en detección y localización de objetos, como puede ser el IoU (*Intersection over Union*), pero esta métrica está planteada para modelos que localizan los vértices del *bounding box* que encierra la herramienta. El funcionamiento de esta métrica es relativamente sencillo, pues se basa en dividir la intersección del *bounding box* predicho por el modelo con el real, entre la unión de ambos, de forma que un valor cercano a 1 se refiere a una superposición prácticamente perfecta.

Esta métrica no tiene demasiado sentido en nuestro modelo, ya que en este caso nos hemos centrado en obtener únicamente las coordenadas del centro de la herramienta, y no las coordenadas del *bounding box* que la encierra. Es por ello por lo que se ha decidido desarrollar otro tipo de métricas que podemos encontrar en algunos artículos, como la precisión basada en la distancia.

En primer lugar, se ha desarrollado una métrica cuyo objetivo es el cálculo de la **distancia media** desde la predicción hasta el centro real de la herramienta. Esta métrica es fundamental en el modelo, pues es la que nos indica en etapas iniciales si el modelo está aprendiendo a situar más cerca del centro real su predicción o si, por el contrario, está divergiendo de este objetivo. Dicho de otro modo, en un principio, métricas como la precisión pueden tener valores muy bajos durante varias épocas porque aún la predicción no es lo suficientemente precisa, pero podemos ver que la distancia media desde la predicción que realiza el modelo hasta el punto real sí que disminuye, lo que significa que la red está entrenando correctamente. Además, esta métrica será la base para otras que se explicarán a continuación.

El funcionamiento de esta métrica es relativamente sencillo como concepto. Partimos de la base que los mapas de calor generados por el modelo poseen un pico o valor máximo en las coordenadas donde considera que se encuentra el centro de la herramienta. Cada uno de los mapas de calor, por tanto, predice la posición de una de las herramientas. También sabemos que el valor de este pico variará entre 0 y 1, siendo una predicción más robusta o fiable cuanto más cercano a 1 sea este valor, por lo que podemos incluir un índice de confianza en esta métrica.

Teniendo en cuenta estas premisas, el procedimiento para obtener la distancia media desde las predicciones hasta los valores reales se resume en los siguientes pasos. En primer lugar, se buscan las coordenadas de los valores máximos ambas imágenes, teniendo en cuenta únicamente los que poseen un valor mayor a un umbral que podemos modificar. Una vez tenemos estas coordenadas, únicamente queda calcular la distancia desde cada punto predicho hasta el centro real de la herramienta. Hay que tener en cuenta que nuestro modelo localiza dos herramientas, por lo que esta medida de distancia será la media entre las dos distancias a cada una de las herramientas.

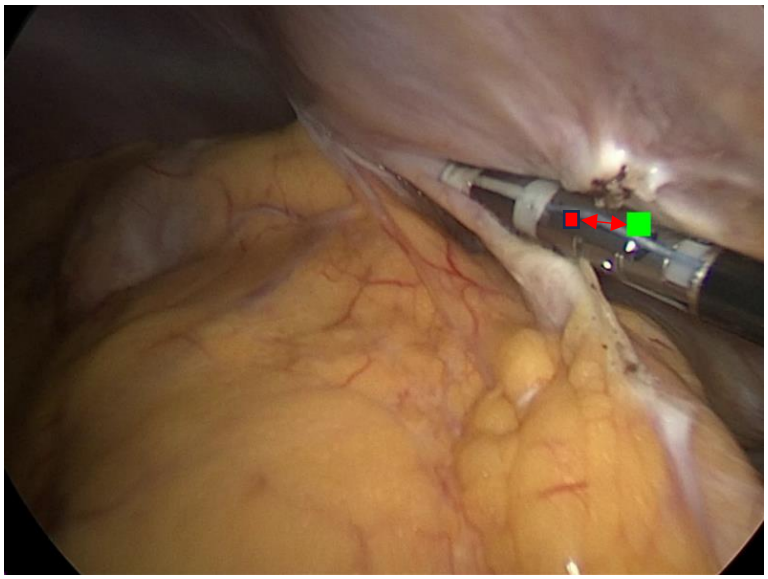


Figura 43. Representación de la medida de distancia calculada desde el punto predicho (verde) hasta el punto real (rojo). Fuente: Elaboración propia.

Por otro lado, al desarrollar estas métricas se han debido tener cuenta distintos aspectos que han complicado en cierta medida su cálculo. Al poseer dos mapas de calor como salida, el modelo detecta dos herramientas cuando ambas se encuentran en la imagen, pero no podemos elegir cuál de las dos herramientas se localiza en cada mapa de calor. Por ello, no es trivial saber desde qué punto calculamos la distancia al centro de cada una de las herramientas. Para dar solución a este problema, se ha optado por obtener cuatro medidas de distancias que se refieren a la distancia desde cada predicción a cada una de las herramientas. Teniendo estas cuatro medidas, encontramos cuál es la distancia mínima, es decir, cuál es la mejor predicción que ha hecho el modelo para alguna de las herramientas. Una vez tenemos esta medida, únicamente queda por elegir la medida de distancia entre la otra herramienta con el otro punto predicho.

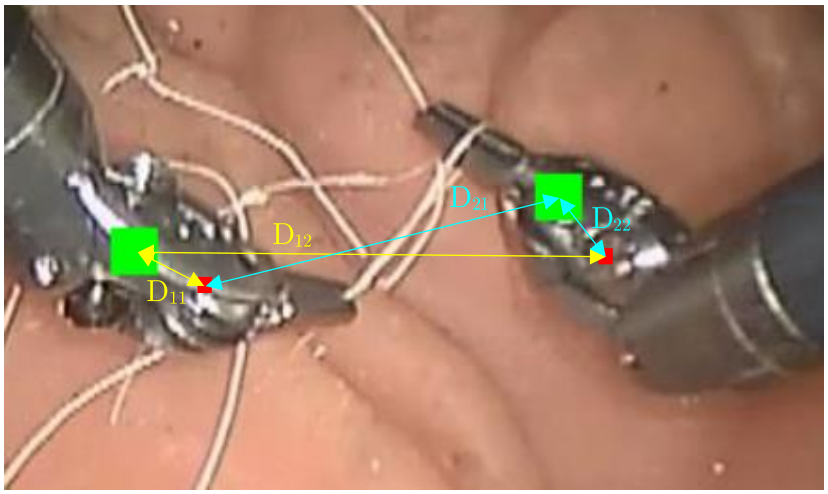


Figura 42. Representación de las 4 distancias calculadas. Fuente: Elaboración propia.

Otro problema que nos podemos encontrar radica en que estamos suponiendo que en las imágenes aparecerán dos herramientas, que es el máximo para el que la red puede predecir la posición, pero puede ocurrir que en la imagen aparezca únicamente una herramienta. En este caso, el modelo puede tomar dos tipos de decisiones. La primera de ellas se basa en predecir con una de las salidas la posición de esa herramienta, dejando la otra de las salidas inactivada o con una activación que no llega al umbral mínimo que hemos establecido. La segunda de ellas es que prediga la posición de la herramienta con ambas salidas de la red. Realmente ninguna de ellas nos supone un problema serio a la hora de localizar la herramienta, ya que en ambos casos estaríamos tomando una posición correcta de la misma. En cambio, sí que puede suponernos un problema a la hora de interpretar la métrica de la distancia tal cual la hemos explicado. Para solucionar este problema, mediante las etiquetas se detecta cuando una muestra tiene únicamente una herramienta, y en estos casos únicamente se toma la distancia mínima desde cada uno de los puntos predichos hasta la herramienta.

El último gran problema que nos encontramos en algunos casos es que no existan herramientas en la imagen. En estos casos se ha decidido darle peso en la métrica al acierto o error en estos caos mediante un umbral binario de confianza. Para una imagen donde no existe ninguna herramienta, si el modelo realiza una predicción con un valor mayor a un umbral establecido, la distancia que se añade al cálculo de la distancia media es el máximo del tamaño de la imagen. En cambio, si el valor de la predicción no llega al umbral, la distancia que se incluirá en el cálculo de la métrica será cero.

Por último, se ha decidido modificar el cálculo de la distancia respecto a lo encontrado en otros artículos. Normalmente, en estos artículos comparan el funcionamiento de un modelo único en cada uno de los conjuntos de datos. En cambio, nosotros queremos ver cómo se comporta nuestro modelo en distintas situaciones, con variaciones en los tamaños de entrada y salida, y compararlo, pudiendo crear modelos más complejos con tamaños de entrada y salida mayores, o probando con modelos más simples donde no sea necesaria una resolución tan elevada. Por ello, se ha decidido calcular la distancia en función de la longitud en píxeles de la imagen, representándolo como un porcentaje. Esto es útil cuando las imágenes no tienen las mismas resoluciones y tamaños, pudiendo comparar el rendimiento en unos y otros modelos de una forma más simple.

La segunda de las métricas que se ha utilizado es la **precisión**. Esta medida nos sirve para clasificar en cuales de las imágenes el modelo ha realizado una buena predicción y en cuales no lo ha hecho. Como hemos comentado anteriormente, en muchos de los artículos se utiliza la precisión calculada en base al IoU. En estos casos, se establece un umbral para este valor y las predicciones que lo superan, se consideran bien clasificadas, siendo las que no lo superan mal clasificadas.

En nuestro caso, al no obtener en ningún momento la localización y dimensión de los *bounding box*, se ha optado por elaborar una medida de precisión basada en la distancia desde los centros predichos por el modelo hasta los centros reales de la herramienta. De esta forma, podemos estimar cuántas de las imágenes han tenido una predicción correcta y cuáles no la han tenido.

Además, esta métrica la podemos hacer más o menos restrictiva en función del valor que establezcamos como umbral. En algunos de los artículos consultados, este umbral se da en forma de píxeles, pero, como hemos mencionado anteriormente, en nuestro caso nos interesa que esta métrica sea comparable con distintos tipos de modelos, con distintos tamaños de salida. Por ello, el umbral de esta métrica también se basa en el porcentaje respecto al ancho de la imagen y no a los píxeles directamente. En caso de querer compararlo con otros tipos de modelos con el mismo tamaño de salida, es trivial pasar del valor del porcentaje al valor en píxeles y viceversa.

5.2.6. Entrenamiento del modelo

Una vez se han elaborado cada una de las partes necesarias para llevar a cabo un correcto entrenamiento del modelo, el modelo ya está listo para poder entrenarse con los datos que hemos recopilado. En cambio, antes de hacerlo se le han añadido dos modificaciones al modelo con la intención de mejorar su rendimiento. Estas modificaciones se tratan de capas convolucionales en tres dimensiones. Estas capas se han añadido en dos lugares distintos del modelo, una capa en cada uno de ellos. En primer lugar, se ha añadido una capa en la entrada del modelo. La utilidad de esta capa radica en que los tres canales que entran al modelo no se tratan de canales independientes de características, sino que se tratan de canales de colores con relaciones muy fuertes entre ellos, por lo que una convolución que tenga en cuenta los tres canales podría mejorar la extracción de características en esta capa y generar mapas de características que codifiquen las propiedades de los tres canales de forma simultánea. La otra localización donde se ha añadido una capa de convolución en 3 dimensiones es al final del modelo. Esta capa tiene sentido en esta localización ya que, al final del modelo, hay que tener en cuenta que los dos mapas de calor no son independientes, sino que tienen relación entre ellos. Esta convolución precisamente encontrará las relaciones que existen entre ellos para intentar que cada uno localice una de las herramientas.

Respecto al entrenamiento, el modelo se ha entrenado en primer lugar durante 20 épocas, utilizando *early stopping* y *reduce on plateu* para controlar el entrenamiento del modelo y evitar tanto el estancamiento como el *overfitting*. En este primer entrenamiento el tamaño de entrada de la red ha sido de 512x512 y el de salida de 128x128, con un número de filtros máximo en las etapas más compactas de 256. También se han utilizado *checkpoints* para guardar únicamente el modelo en la siguiente época si en esta se mejoraban las prestaciones de la época anterior.

Como datos de entrenamiento, se ha utilizado el conjunto de datos Atlas Dione por dos motivos principalmente. En primer lugar, en comparación con el conjunto *Endovis'15*, Atlas Dione tiene una cantidad mucho mayor de datos para entrenar, siendo imposible llevar a cabo el entrenamiento con el otro. En segundo lugar, comparándolo con el conjunto tomado en ITAP, además de poseer un mayor número de imágenes, estas están mucho mejor etiquetadas, ya que en el conjunto de ITAP encontramos gran cantidad de imágenes donde aparece la herramienta y no aparece etiquetada en el archivo correspondiente, lo que dificulta enormemente el entrenamiento del modelo.

De un conjunto de 22.467 imágenes, se han tomado 18.782 imágenes como conjunto de entrenamiento y 3685 como conjunto de test. Además, el 20% de las imágenes del conjunto de entrenamiento se han reservado como conjunto de validación.

Una vez entrenado con este conjunto de datos, cuyos resultados se comentarán en capítulos posteriores, también se ha entrenado el modelo cambiando únicamente el número de filtros, pasando de 256 a 128. Este cambio se ha realizado para comprobar exactamente cuál es la pérdida de rendimiento al reducir el número de filtros. Recordemos que el objetivo de este proyecto es encontrar un modelo que nos ofrezca una precisión mínima para su aplicación y un tiempo de inferencia suficiente para aplicarlo en tiempo real. Esto nos obliga a intentar simplificar tanto como podamos el modelo mientras su rendimiento se mantenga.

Para entender la magnitud del cambio en este tipo de modelos, al pasar de 128 a 256 filtros se ha pasado de un modelo con 1.946.068 parámetros, a un modelo con 500.468 parámetros, prácticamente 4 veces menos. Este cambio se puede traducir en un menor tiempo de inferencia, lo que se discutirá en capítulos posteriores.

Por último, se entrenó otro modelo, de mucho menor tamaño que los anteriores. En este caso, su diferencia no se encontraba en el número de filtros utilizados, que fueron 128, sino en la arquitectura de la red y en el tamaño de las imágenes de entrada. En este caso, se buscó llevar la simplificación del modelo al extremo, utilizando únicamente el modelo inicial (se eliminaron las capas de convolución 3D para comprobar si esa función la podían llevar a cabo las capas adyacentes), y reduciendo el tamaño de entrada al mínimo para el que la predicción generada podía tener una sensibilidad suficiente, que fue 256x256, generando una imagen de tamaño 64x64.

Para compensar esta falta de procesamiento, se añadieron al entrenamiento las imágenes del *dataset* generado en el ITAP, buscando una mayor variabilidad, al encontrarse con herramientas distintas y entornos más similares a operaciones reales. El modelo recibió las imágenes de uno y otro *dataset* combinando las imágenes de manera aleatoria y el número total de parámetros que se tuvieron que entrenar fue de 443.854, menor que en los dos modelos anteriores.

Capítulo 6. Resultados obtenidos y análisis del modelo

En este capítulo se abordarán dos objetivos claramente diferenciados. En primer lugar, se mostrarán el rendimiento que ha alcanzado el modelo con las distintas configuraciones. En segundo lugar, se analizará brevemente el funcionamiento del modelo con imágenes del conjunto de datos con la intención de eliminar el falso mito de que los modelos de redes neuronales son una caja negra. El objetivo de esa parte será precisamente intentar aportar una explicación a porqué el modelo aporta los resultados que aporta y qué características de la arquitectura lo causan.

6.1. Resultados obtenidos

En cuanto a los resultados, hay que recordar que se han entrenado dos modelos con el conjunto de datos Atlas Dione, con un número máximo de filtros diferente, y uno con un *dataset* combinado. El primero del primer grupo tiene un número máximo de filtros de 256, siendo el segundo mucho menor, con 128. En cuanto al número de parámetros, hemos visto que cambia drásticamente de uno a otro modelo, pero es necesario ver cómo varía su rendimiento tanto en precisión en la predicción, como en tiempo de inferencia.

A su vez, en el modelo entrenado con datos combinados e imágenes reducidas, también se ha visto un cambio importante en el número de parámetros del modelo, pero hay que ver si en este caso, la pérdida de información deriva en un peor rendimiento del modelo desde el punto de vista de la exactitud de las predicciones.

Los resultados se han obtenido evaluando el modelo en el conjunto de test que habíamos reservado en el *dataset* Atlas Dione. Como se explicó en capítulos anteriores, este subconjunto está formado por 3685 imágenes en situaciones muy distintas, lo que nos permite obtener conclusiones con una mayor fiabilidad que en otros conjuntos donde el número de imágenes de test es mucho más reducido y homogéneo.

En cuanto a las métricas, es importante mencionar en qué condiciones se han obtenido. En cuanto a la exactitud o *accuracy*, esta se ha calculado estableciendo un rango máximo de error del 10% del ancho de la imagen. La elección de este umbral se ha elegido debido a que la finalidad del modelo es permitir a un robot quirúrgico localizar las herramientas y mantenerlas en la imagen continuamente. Para ello, se ha considerado que este error es un buen descriptor de la utilidad del modelo para esta aplicación. El error medio se ha calculado de acuerdo con la explicación expuesta en la sección 5.2.5.

Por último, los FPS (*frames per second*) que permite predecir el modelo se ha obtenido mediante un *script* que toma los distintos fotogramas de un video y obtiene las coordenadas de los centros de las herramientas a partir de los máximos del mapa de calor. Este *script* se ha ejecutado en un equipo dotado de un procesador Intel i5, X Gb de RAM y una tarjeta gráfica NVIDIA RTX 3060 Ti.

Tabla 2. Resultados de los modelos.

Modelo	***Exactitud (10%)***	***Error Medio (%)***	***FPS***
Hourglass Model (256 filtros)	0.8925	2.03	10.89
Hourglass Model (128 filtros)	0.8650	3.12	15.28
Hourglass Simple	**0.9286**	**1.72**	**22,64**

6.2. Análisis del modelo

Además de las métricas del modelo, también es conveniente ver cuál es el proceso que sigue el modelo para hacer sus predicciones, con la intención de poder ver posibles fallos, capas que puedan sobrar y, sobre todo, darle una explicación al funcionamiento de un tipo de modelos que tradicionalmente se han visto como cajas negras.

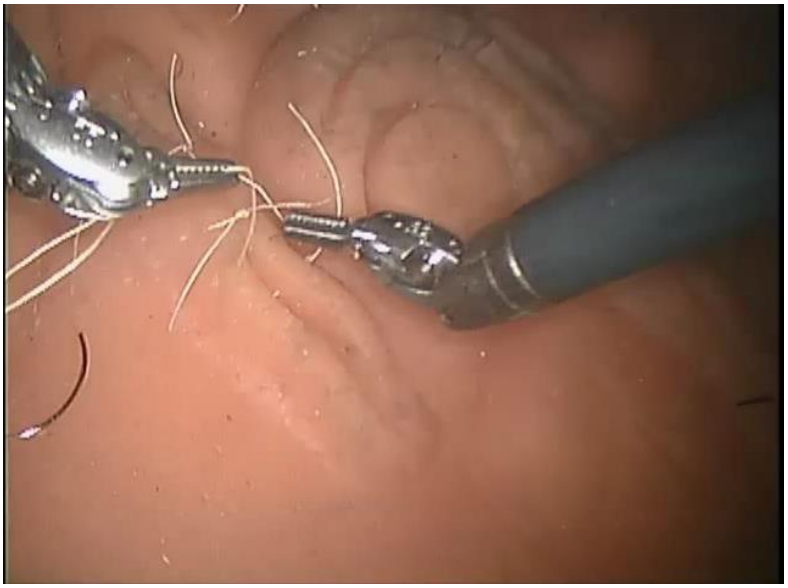


Figura 44. Imagen utilizada para intentar explicar el comportamiento del modelo. Fuente: Elaboración propia.

En la figura 44 podemos ver la imagen que hemos utilizado para observar las salidas intermedias del modelo y así poder explicar su funcionamiento. Los puntos elegidos para obtener las salidas intermedias del modelo han sido los siguientes:

- Salida de primera convolución 3D
- Entrada al primer módulo *hourglass* después de reducir la dimensionalidad
- Salida del primer módulo *hourglass*
- Salida de la primera arquitectura de predicción
- Salida del segundo módulo *hourglass*
- Salida de la segunda arquitectura de predicción
- Salida final

Antes de comenzar con el análisis de imágenes, es importante recordar que estas salidas intermedias están formadas por un elevado número de canales, es decir, vamos a obtener un gran número de imágenes con información muy diversa. Por ello, en este documento solo se expondrán algunas de ellas que tengan un especial interés.

En primer lugar, comenzaremos analizando el efecto que tiene la **capa de convolución 3D** en el modelo. Esta capa, como ya se ha explicado en la parte teórica del modelo, tiene como objetivo encontrar relaciones entre los tres canales que forman la imagen RGB. Esta capa es especialmente útil cuando tenemos imágenes con relaciones espaciales en la tercera dimensión, y no solamente en las dos primeras, por lo que podría ser útil con imágenes RGB, pero también con imágenes volumétricas como imágenes por resonancia magnética o tomografía computarizada.

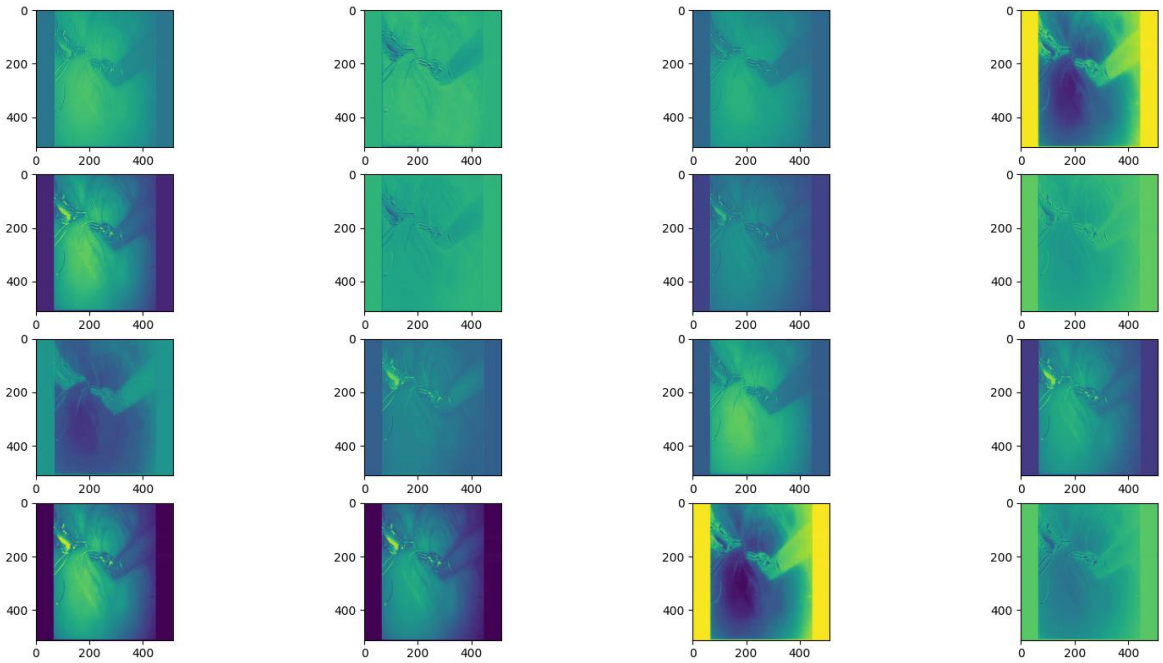


Figura 45. Salida de la red después de la primera convolución 3D. Fuente: Elaboración propia.

En la figura 45 podemos ver 16 de las 32 salidas de esta primera capa de convolución en tres dimensiones. Podemos ver como en esta capa, aún la información extraída posee pocas características estructurales y se basan principalmente en características espaciales. Podemos ver que en algunas de las imágenes se empiezan a detectar algunas propiedades de la imagen como los reflejos, los relieves, o los bordes laterales, de los que hablaremos posteriormente. Como hemos visto con el modelo más simple, en ocasiones, al menos con imágenes RGB, estas capas no son necesarias porque el resto del modelo en su conjunto es capaz de condensar esta información espacial. Su utilidad será mucho mayor en imágenes tridimensionales.

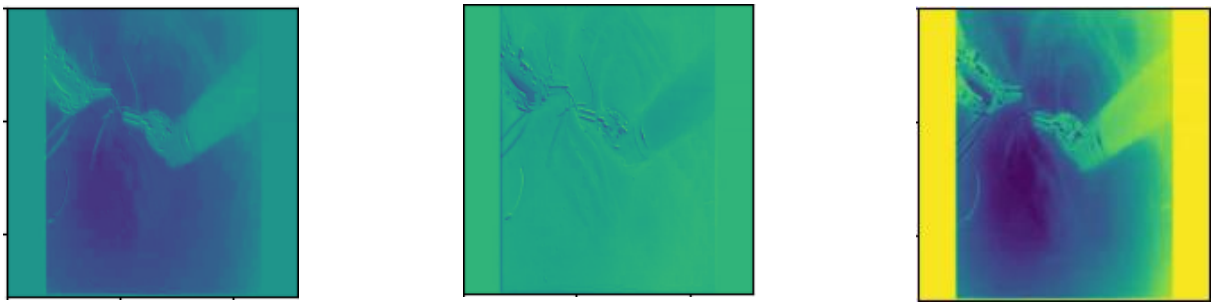


Figura 46. Canales de detección de bordes (izquierda), relieves (centro) y bordes laterales y objetos (derecha). Fuente: Elaboración propia.

A continuación, veremos cuál es la evolución que sufre una imagen al someterse a las dos capas de **reducción de dimensionalidad**. En estas capas cabe esperar que empiecen a codificarse algunas de las características algo más abstractas, como el reconocimiento de alguna estructura de gran tamaño mediante la convolución con un tamaño de filtro 7x7.

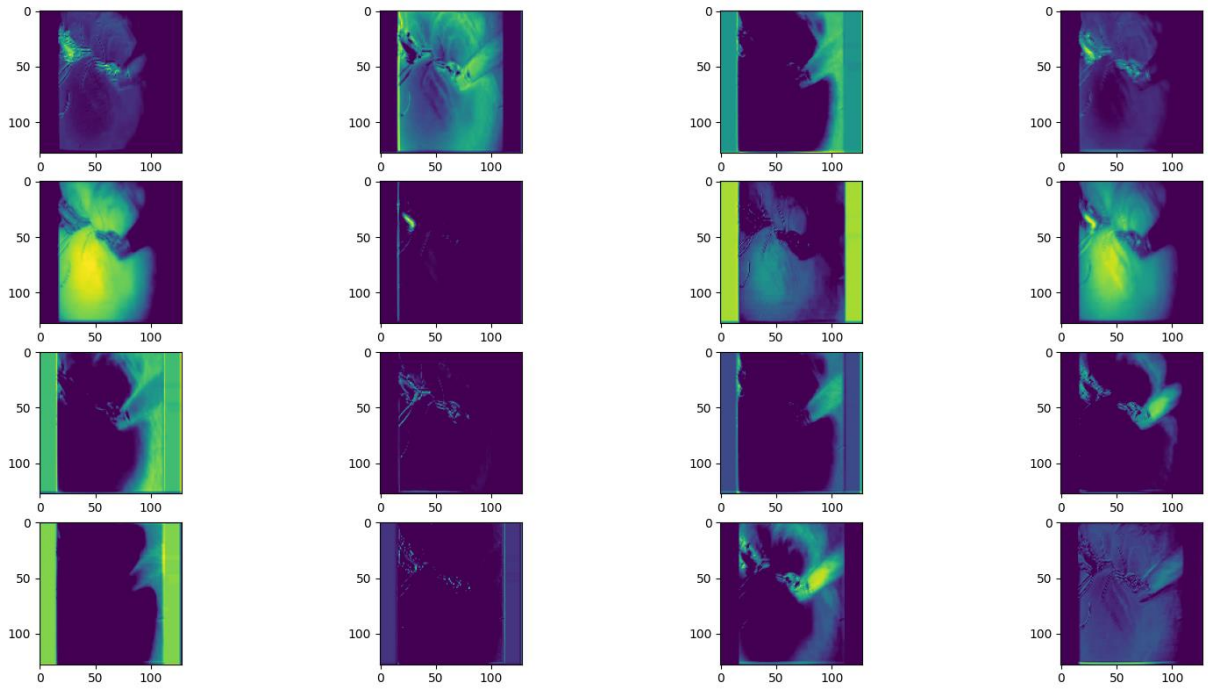


Figura 47. Salida después de las capas de reducción de dimensionalidad. Fuente: Elaboración propia.

Como se puede ver en la figura 47, ya podemos ver algunas de las características que el modelo busca en la imagen para realizar las predicciones. En primer lugar, podemos visualizar que en algunas imágenes se pueden diferencias las zonas oscuras y claras de la imagen. En otras se puede ver el relieve de la imagen con mayor detalle y, en algunas incluso, se puede empezar a distinguir una segmentación de la herramienta. Además, en estas capas se ha aplicado por primera vez una suma de canales, cuyo resultado se puede apreciar sobre todo en las imágenes que apenas posee información. Este suceso podría estar causado por los bordes negros de las imágenes, dado que el modelo en algunos de los canales prácticamente solo diferencia los bordes del resto de la imagen, teniendo estos un valor mucho mayor o menor, puede que la información de los bordes opaca la del resto de canales.

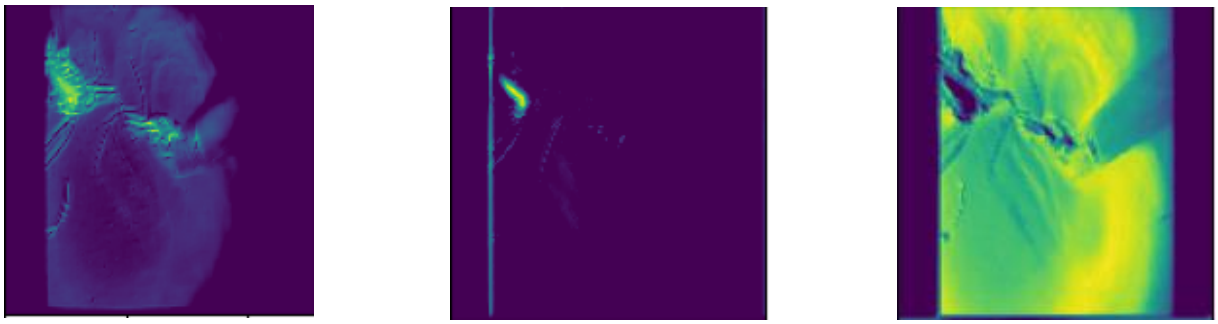


Figura 48. Canales donde se puede apreciar detección de herramientas (izquierda y centro) y diferenciación de bordes (derecha). Fuente: Elaboración propia.

La siguiente parte del modelo que analizaremos será el **módulo *hourglass***. Este módulo reduce la dimensionalidad de las imágenes para extraer características a distintas escalas, de forma que en la fase de aumento de dimensionalidad o *upsampling*, suma las características obtenidas a distintas escalas para obtener una imagen con características espaciales (obtenidas a partir de las imágenes con gran resolución) y estructurales (codificadas en las imágenes con menor resolución). Para ello, se analizará la salida al final de este módulo, pero antes de la arquitectura de predicción de herramientas. Em estos casos, sería esperable encontrarnos con imágenes de menor resolución en muchos casos, ya que esta se ha reducido para extraer características y, aunque exista una fase de reescalado, es imposible recuperar toda la información anterior.

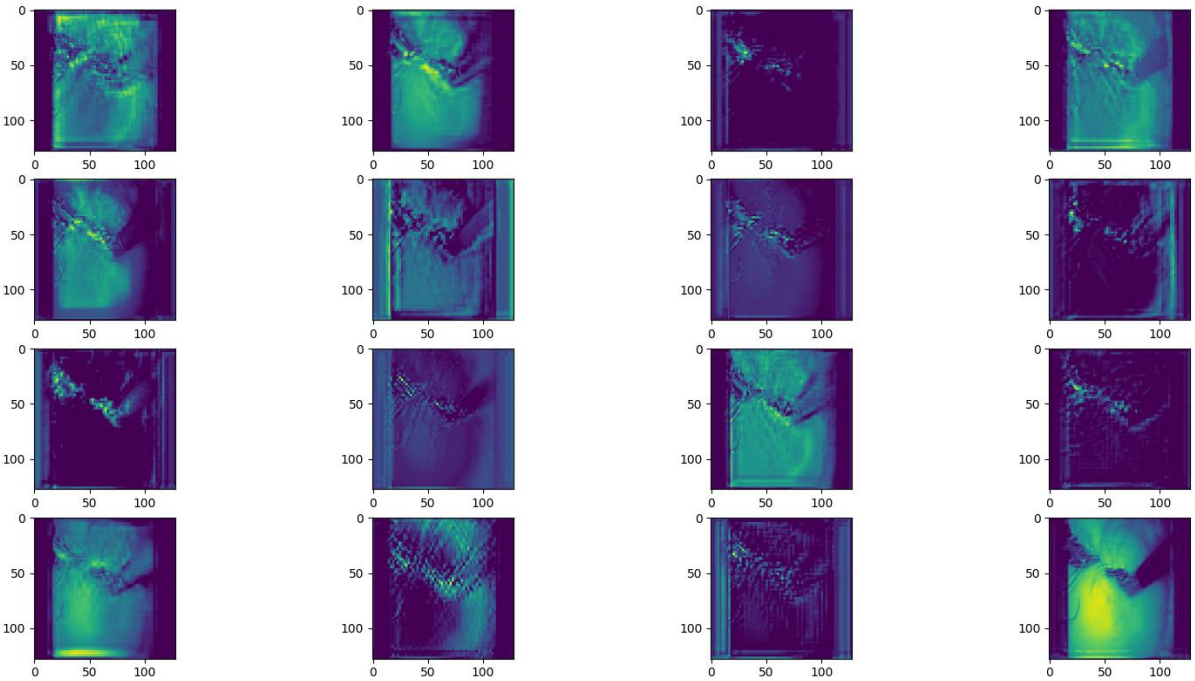


Figura 49. Salida del primer módulo *hourglass*. Fuente: Elaboración propia.

En la figura 49 podemos ver que, como esperábamos, la resolución de las imágenes ha disminuido notablemente. Seguimos viendo diversos canales con poca información, pero podemos empezar a ver canales que se aproximan a los centros de las herramientas. Otros, en cambio, vemos que hacen énfasis en el fondo de la imagen, lo que también es realmente útil a la hora de ser capaz de diferenciar los objetos de su entorno.

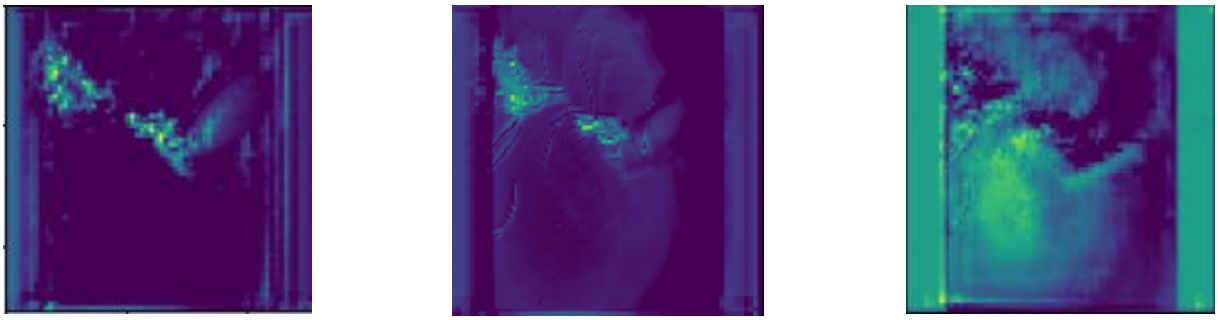


Figura 50. Canales donde se empieza a localizar la posición solo de la herramienta (izquierda y centro) y el fondo de la imagen, con las herramientas sin apenas iluminación. Fuente: Elaboración propia.

Una vez se ha procesado la imagen en el módulo *hourglass*, se utiliza una **red convolucional simple** para extraer las características más importantes. Además, la salida de esta arquitectura se suma con la imagen antes de ser procesada en el módulo, a modo de supervisión. Es decir, se crea una conexión de escape a la que únicamente se le aplica una convolución de tamaño de máscara 1x1 para encontrar correlaciones entre canales, pero manteniendo las características espaciales. De esta forma, al final de esta arquitectura podremos ver las características estructurales extraídas en el módulo *hourglass* junto con las características espaciales de la imagen de entrada, por lo que es esperable obtener imágenes más nítidas.

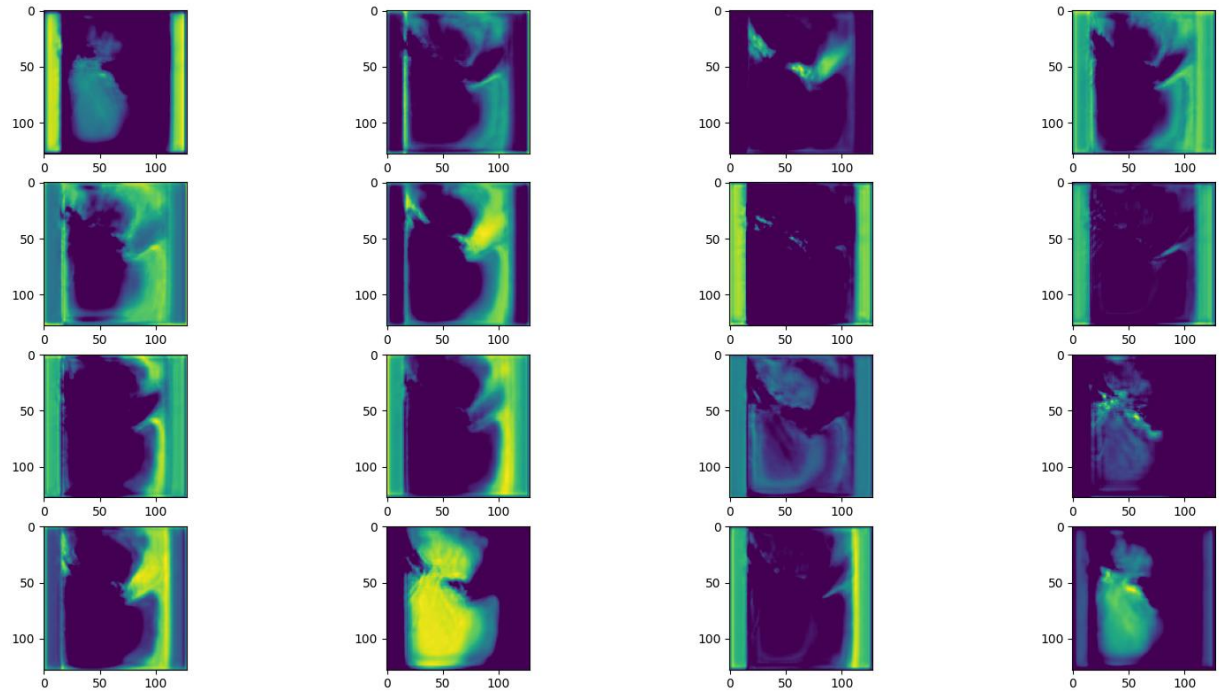


Figura 51. Imágenes a la salida de la arquitectura de predicción. Fuente: Elaboración propia.

A la salida de esta capa podemos ver varias de las características anteriormente mencionadas. En primer lugar, podemos ver imágenes mucho más nítidas, sobre todo en las que se resaltan las herramientas, siendo estas prácticamente segmentadas. También en el resto de las imágenes podemos ver una definición de las diferentes partes que forman el fondo, tanto en las zonas de mayor claridad de la imagen, como en las menos iluminadas, donde las tareas de predicción se dificultan en gran medida. Además, en algunas imágenes se puede ver como en las propias herramientas las conas cercanas al centro ya están tomando más importancia que las zonas más lejanas, lo que ya se trata de características más elaboradas.

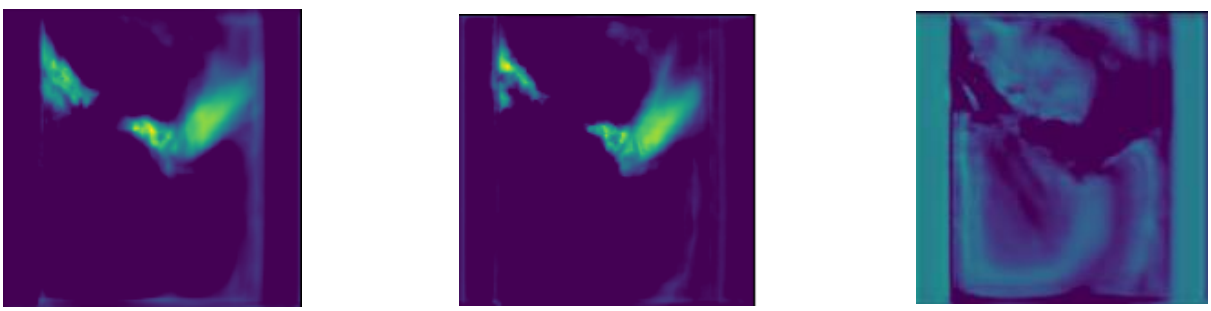


Figura 52. Canales con las herramientas resaltadas (izquierda y centro), y el fondo sin herramienta (derecha). Fuente: Elaboración propia.

Una vez las imágenes han pasado este primer procesado gracias a la primera parte de la arquitectura del modelo, se vuelven a introducir en otro **módulo *hourglass*** para extraer más características estructurales. Conforme lo observado en la etapa anterior, sería lógico que empezasen a aparecer mapas de activación donde los centros de la herramienta tuviesen una prevalencia mayor en la imagen. Además, en la figura 52 ya se puede observar en las imágenes de la izquierda y del centro que, aunque ambas resaltan las herramientas, cada una le otorga mayor intensidad a una y a otra, por lo que sería lógico también observar esta tendencia.

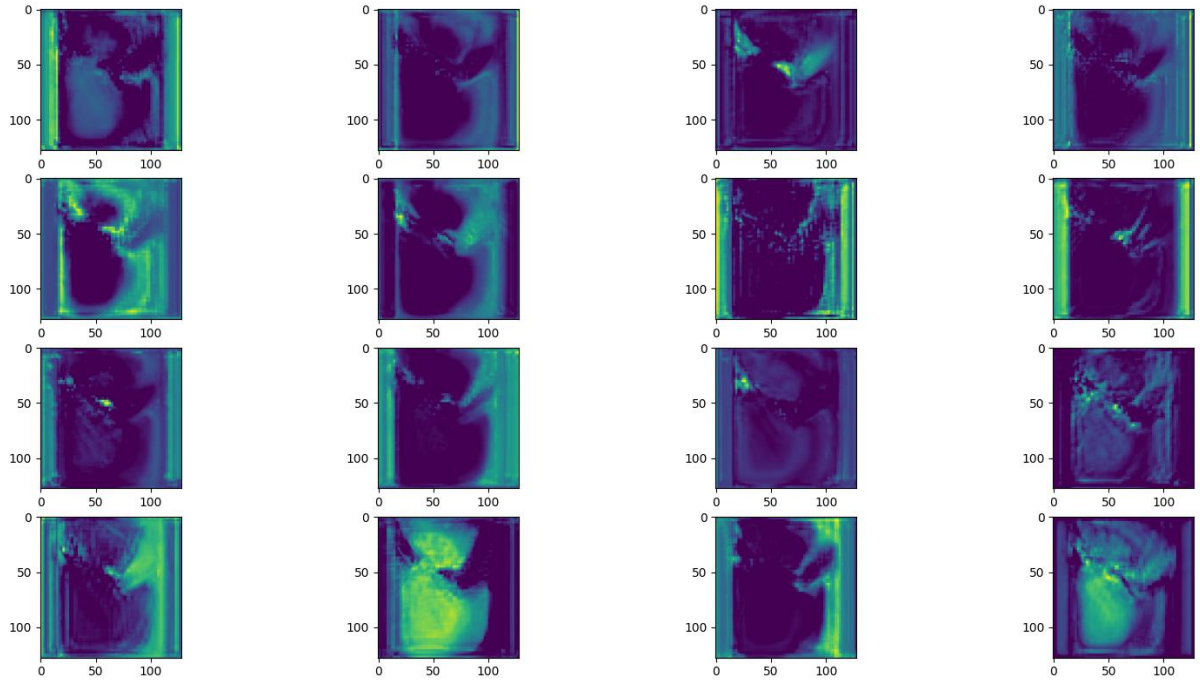


Figura 53. Imágenes a la salida del segundo módulo *hourglass.* Fuente: Elaboración propia.

En la figura 53 podemos observar lo que se presuponía. Existen imágenes en las que solamente se aprecian los centros de las herramientas, y en algunas de ellas, solamente se observa el centro de una de ellas. Además, siguen siendo abundantes los mapas que segmentan el fondo de la imagen, en este caso ya sin poder apreciar sus detalles más allá de su forma, lo que es indispensable a la hora de combinar toda esta información en la etapa que se explicará a continuación.

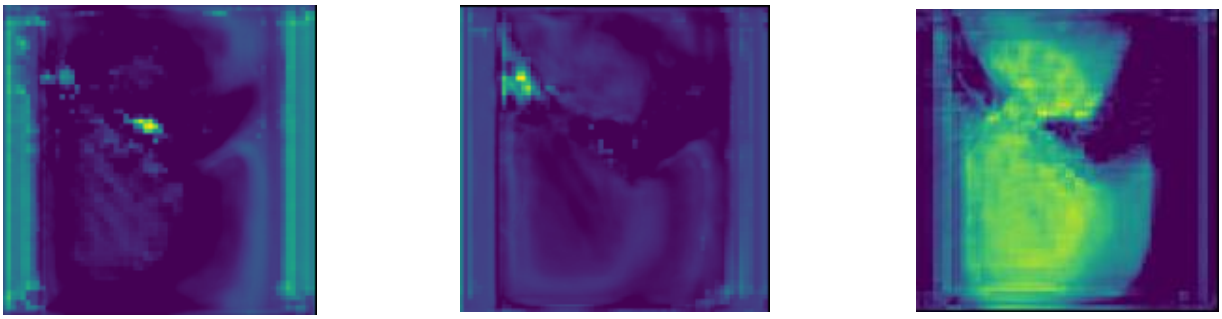


Figura 54. Imágenes donde podemos ver el centro de cada herramienta (izquierda y centro), y el fondo de la imagen (derecha). Fuente: Elaboración propia.

Una vez tenemos todas las características codificadas en los mapas de activación presentados anteriormente, se ha utilizado otra **red convolucional simple** para combinar todas estas características en solamente dos mapas de activación. En este caso, la arquitectura convolucional solamente se compone de dos convoluciones y una función de activación, con el objetivo de ahorrar parámetros en el modelo. La salida de esta arquitectura se compone de 2 imágenes, partiendo de una entrada de 32 o 64, dependiendo el número de filtros. Esto significa que esta arquitectura englobará todas las características codificadas en estos canales y las compactará en dos canales, que ya deberían reflejar la posición de las dos herramientas únicamente.

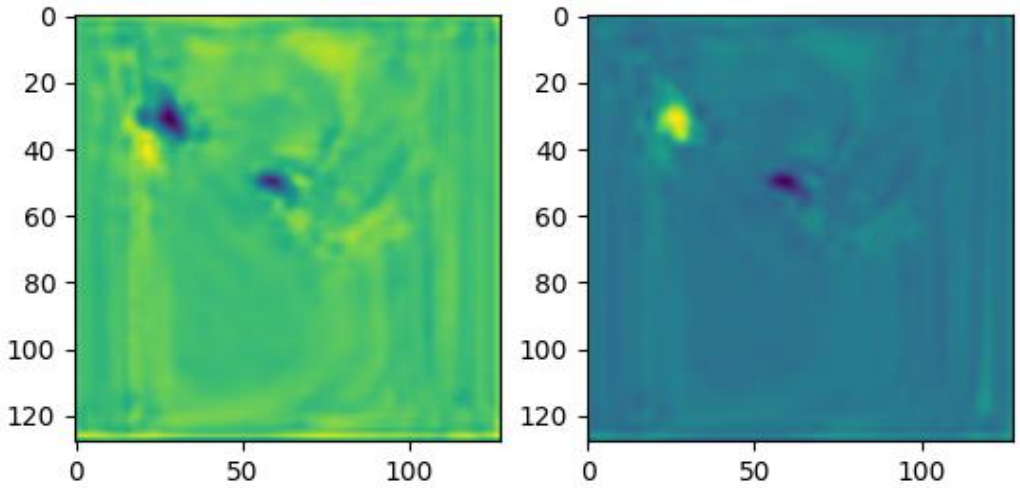


Figura 55. Mapas de activación después de la arquitectura de predicción. Fuente: Elaboración propia.

Como podemos ver en la figura 53, la capa de activación ya es capaz de darnos la posición de ambas herramientas. Es interesante ver cómo en uno de los mapas las dos herramientas toman la misma importancia, estando resaltada toda la imagen excepto los dos centros, pero en la otra el modelo es capaz de diferenciar una herramienta de la otra, teniendo toda la imagen un valor neutral, con anormalmente altos en uno de los centros y valores anormalmente bajos en la otra. Esto nos indica que el modelo no solamente es capaz de detectar las herramientas, sino que además está siendo capaz de diferenciar una de la otra.

Por último, se ha incluido una **última convolución en 3 dimensiones** con un tamaño de máscara 1x1x2. Este canal únicamente busca encontrar relaciones entre los dos mapas de activación, ya que queremos que cada uno nos aporte información únicamente de una herramienta. De esta forma, esta convolución, si en alguno de los dos mapas se diferencian las dos herramientas, como es el caso, será capaz de diferenciar la información que debe guardarse en uno u otro canal. En el caso del modelo entrenado con ambos *datasets*, la última convolución con máscara de tamaño 1x1 de la arquitectura de predicción podría llevar a cabo esta función, ya que hemos visto que sí poseía la información de ambas herramientas diferenciadas.

Es trivial saber que la salida deseada en esta última capa es una imagen lo más similar posible a las etiquetas, es decir, un mapa de calor donde se forme una función *gaussiana* entorno a los centros de las herramientas, y cuyo valor máximo corresponda al píxel descrito como

centro de la herramienta. La función de activación de esta última capa se corresponde con una función sigmoide, por lo que los píxeles tendrán valores que oscilen entre 0 y 1.

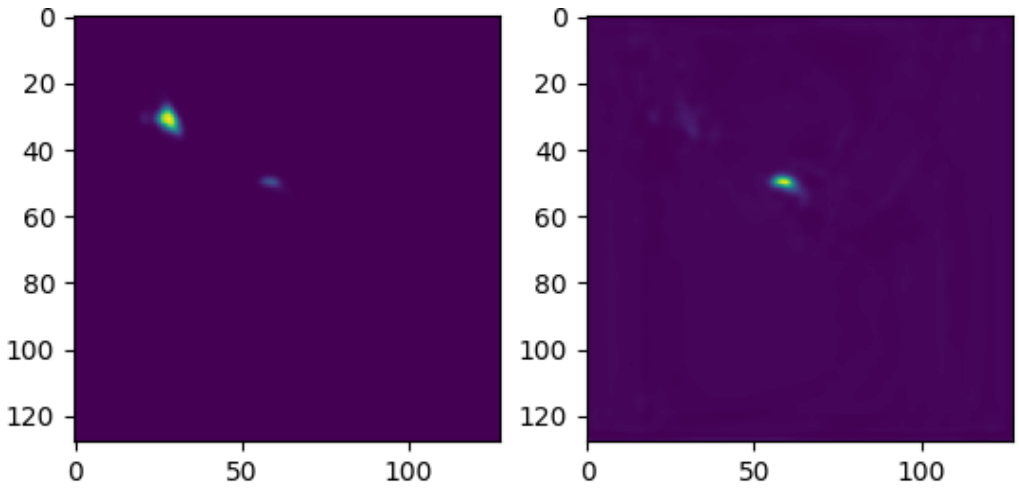


Figura 57. Imágenes de la salida final del modelo. Fuente: Elaboración propia.

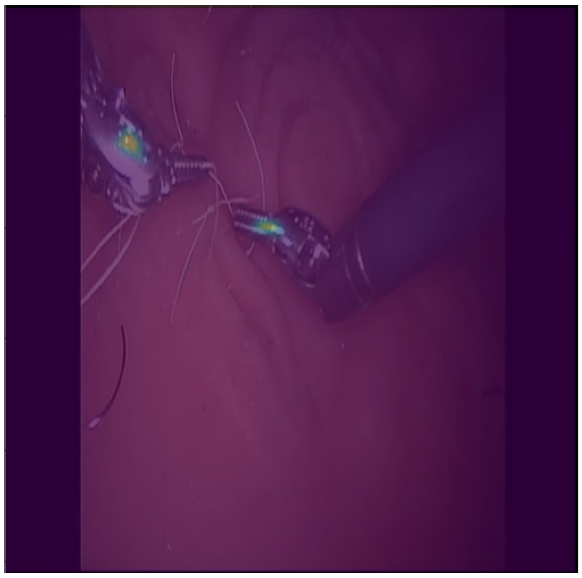


Figura 56. Superposición de las predicciones con la imagen original. Fuente: Elaboración propia.

Parte III. Análisis de resultados y conclusiones.

Capítulo 7. Análisis y discusión de resultados. Limitaciones del proyecto.

En este capítulo se hará una explicación más detallada de los distintos resultados obtenidos por el modelo, así como una justificación de estos resultados desde distintos puntos de vista. También se expondrán las distintas limitaciones que nos hemos encontrado a la hora de realizar el proyecto y las que hemos confirmado a la vista de los resultados, con el objetivo de poderlas dar solución en un futuro.

En primer lugar, comenzaremos analizando los resultados que ha mostrado el modelo con el *dataset* con Atlas Dione. Los resultados más generales se han mostrado en la tabla 2, pero hay otros aspectos que se pueden valorar. Además de obtener resultados para un umbral de error del 10%, se ha realizado un análisis del rendimiento del modelo variando el umbral de error. En este caso, al haberse realizado comparando el mismo *dataset*, sí que se ha utilizado la medida en píxeles, de forma similar a cómo lo realizan en muchos artículos de la literatura. Se ha visto la variación del modelo de 128 filtros máximos, con el modelo de 256, y se ha analizado su precisión (*accuracy*) variando el umbral entre 0 y 20 píxeles.

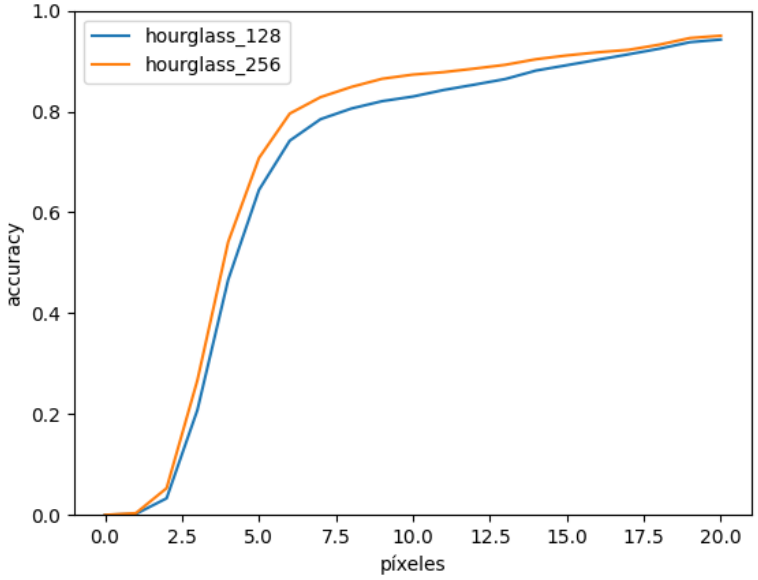


Figura 58. Comparación de la exactitud (*accuracy*) del modelo de 128 filtros y el modelo de 256. Fuente: Elaboración propia.

Como se puede apreciar, el modelo de 256 filtros tiene una exactitud mayor, no solamente para un umbral del 10%, sino que lo hace para cualquiera de los valores entre 0 y 20 píxeles. En la figura 58 podemos ver que, para los 20 píxeles, el modelo de mayor tamaño alcanza un exactitud máxima de 0.9502, muy similar al del modelo más pequeño. En cambio, la exactitud de cada modelo para intervalos intermedios de píxeles sí es mayor en el modelo más complejo, lo que puede ser útil cuando se necesite ser más preciso en la predicción que un 10%. En definitiva, es sencillo ver que el modelo más complejo, como cabría esperar, nos ofrece mejores resultados en términos de fiabilidad.

En cambio, como ya se ha comentado en varias ocasiones, no solamente debemos fijarnos en la precisión del modelo. Es importante recordar que su función es al de localizar la herramienta para que un dispositivo robótico sea capaz de mantenerla en el plano. Para ello, además de una mínima precisión, se necesita que el modelo sea capaz de aportarle datos al robot en tiempo real sobre la posición de la herramienta. Por ello, habría que valorar el hecho de perder unos cuantos píxeles de precisión, si con ello ganamos una cantidad importante de *frames* por segundo.

Es fundamental remarcar que los *frames* por segundo a los que es capaz de funcionar el modelo dependen enormemente del equipo que se utilice. En el caso de un equipo portátil, con un procesador IntelCore i5 adaptado a equipos portátiles, una memoria RAM de 8Gb y una tarjeta gráfica dedicada NVIDIA MX130, el modelo de 256 filtros es capaz de rendir a 1.74 FPS y el modelo de 128 filtros lo hace a 4.19 FPS. En cambio, en un equipo de sobremesa, con un procesador IntelCore i5, 12 Gb de memoria RAM y una tarjeta gráfica dedicada NVIDIA RTX 3060 Ti, obtenemos con el modelo más complejo un rendimiento de 10.89 FPS, mientras que con el modelo más simple obtenemos 15.28 FPS.

A continuación, se evaluó el modelo en los otros *datasets* mencionados anteriormente. Para ello, en el caso del conjunto elaborado en el ITAP, se intentó realizar el entrenamiento de dos formas distintas. La primera de ellas se trató de tomar el modelo preentrenado con el conjunto Atlas Dione y entrenarlo con este nuevo conjunto, debido a que se consideró realmente complicado realizar el entrenamiento con 3532 imágenes, estando presentes las herramientas únicamente en 609 de ellas. El entrenamiento de este modelo fue un fracaso, pues no era capaz de converger, visible a través de la distancia media, que en ningún momento llegó a reducirse. Viendo estos resultados, se procedió a entrenarlo con ambos *datasets* mezclándolos de forma aleatoria y, en este caso, como hemos comentado antes, tuvo mucho mejores resultados.

Aunque en un primer momento este modelo no aparentaba ser mejor que los otros dos, debido a que sus métricas de entrenamiento son mucho menores, fue sorprendente ver su rendimiento con el conjunto de entrenamiento, como ha podido verse en la tabla 2. Una vez se han visto sus resultados, se procedió a comprobar porqué los resultados en el entrenamiento fueron peores que en el resto de los modelos y, en cambio, en el conjunto de test los resultados fueron tan equiparables.

La búsqueda se destinó sobre todo comprobar su rendimiento en ambos *datasets* para comprobar si en alguno de ellos su rendimiento difería. Tras la comprobación se confirmó que en el *dataset* elaborado en el ITAP el rendimiento era muy inferior que en el Atlas Dione.

Una vez se comprobó este aspecto, se hizo un análisis exhaustivo de este *dataset* en busca de las causas de esta diferencia de rendimiento. Entre ellas, las más importantes se basaban en fallos en el etiquetado del *dataset*. Se pudieron ver imágenes donde aparecía la herramienta y que, según su respectiva etiqueta, esta no se encontraba presente. Este factor influye enormemente en el modelo ya que se encuentra con imágenes que poseen herramienta y, en cambio, se ve penalizado si la detecta, por lo que el entrenamiento se complica exponencialmente. Además, algunas de las imágenes poseen un componente muy importante de artefactos de movimiento, y en algunas de ellas aparecen objetos que en extrañas ocasiones aparecerían en una cirugía laparoscópica real.

Por suerte, la fracción que representan estas imágenes defectuosas respecto de las del *dataset* Atlas Dione es muy reducida, y en lugar de impedir su entrenamiento, causaron que la detección de herramientas en este sea mucho más robusta, lo que podemos confirmar con una nueva métrica personalizada, el **valor máximo medio (VMM)**.

Esta nueva métrica se ha desarrollado para comprobar con qué grado de confianza el modelo es capaz de detectar la herramienta. Su cálculo tiene un fundamento tan sencillo como obtener el valor medio de los puntos máximos de los mapas de calor, lo que anteriormente hemos dicho que estaba relacionado con la confianza de la predicción. Por tanto, un valor medio cercano a 1 significaría que el modelo encuentra el centro con una confianza mucho mayor, mientras que si se acerca más a 0 implica que su confianza es menor.

Los resultados de esta nueva métrica fueron sorprendentes. Mientras que los dos modelos, de 256 y 128 filtros, entrenados únicamente con el *dataset* Atlas Dione, ofrecían un valor máximo medio de 0.7780 y 0.6115 respectivamente, en el caso del modelo entrenado con ambos conjuntos de datos, su valor máximo medio fue de 0.79413. De esta forma, podemos ver que el modelo entrenado con ambos *datasets* ofrece una robustez mayor que los otros dos modelos, ya que es capaz de diferenciar con mayor claridad el centro de la herramienta del resto de la imagen.

También se han querido comparar los modelos de una manera similar a la vista en la figura 58. En cambio, en este caso no podemos comparar los modelos a medida que aumenta el número de píxeles ya que la resolución de las imágenes de salida es distinta. Por ello, se ha llevado a cabo un análisis idéntico variando el porcentaje de error entre 0 y 10%.

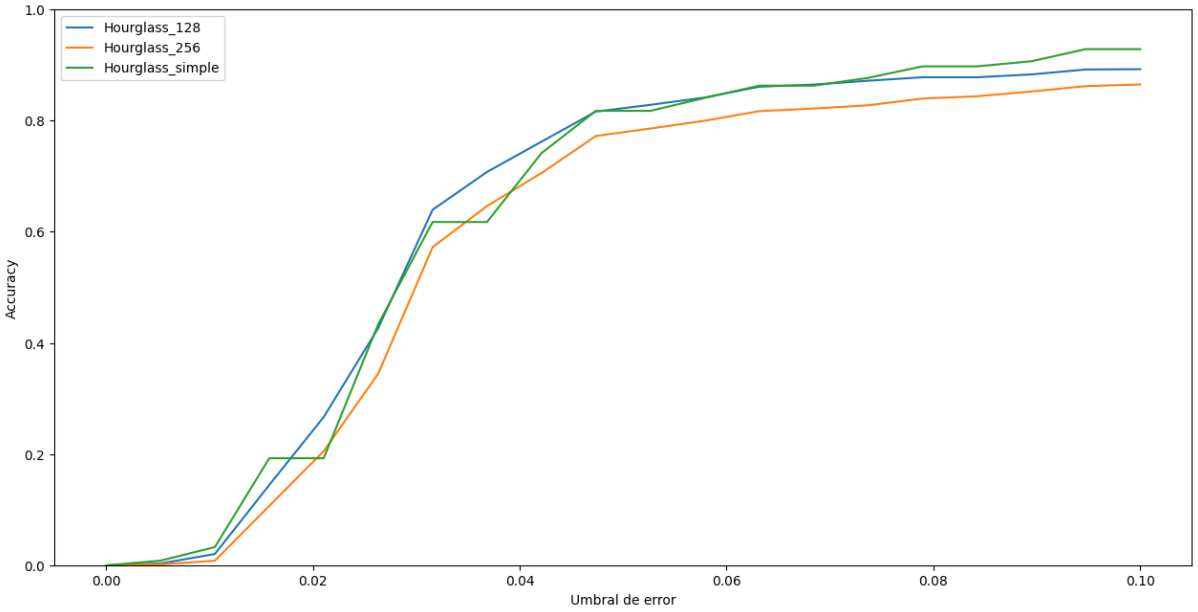


Figura 59. Comparación de los tres modelos. Fuente: Elaboración propia

Por último, se ha evaluado el modelo en el conjunto *EndoVis'15*. Este *datasets* se compone de muy pocas imágenes, 180, claramente insuficientes para realizar un entrenamiento con ellas. Por ello, se ha usado para evaluar el rendimiento del modelo en un conjunto de datos radicalmente distinto a los que se han utilizado para llevar a cabo su entrenamiento. Es lógico que los rendimientos de estos modelos en este set de datos sean muy inferiores a los vistos anteriormente, pues cambia el tipo de herramienta, el fondo de la imagen e incluso el enfoque de la lente. Los resultados en estos conjuntos se muestran en la tabla 3.

Tabla 3. Resultados de los modelos.

Modelo	***Exactitud (10%)***	***Error Medio (%)***	***VMM***
Hourglass Model (256 filtros)	0.3152	39.45	0.2762
Hourglass Model (128 filtros)	0.3533	30.27	0.2811
Hourglass Simple	**0.4293**	**20.70**	0.3034

A la vista de los resultados de la tabla 3, podemos ver que el modelo que mejor consigue generalizar es el más simple, entrenado con ambos *datasets*. Esto es esperable ya que ha sido expuesto a imágenes con mayor variabilidad. En cambio, es algo más sorprendente la diferencia de resultados entre los otros dos conjuntos de datos. Mientras que en el subconjunto de test de Atlas Dione el modelo con 256 filtros obtuvo mejores resultados que el de 128 desde el punto de vista de la precisión de la predicción, en este caso se da la situación contraria.

La explicación de este fenómeno podría basarse en que, al poseer un número mayor de filtros, el modelo más complejo ha sido capaz de aprender más características del conjunto de datos Atlas Dione, de forma que se ha adaptado demasiado a ese conjunto y su rendimiento es

peor en conjuntos distintos. Este suceso es similar al *overfitting*, pero en este caso ocurre cuando un modelo tiene muy buenos resultados y generaliza realmente bien para imágenes de características similares a las de su conjunto de entrenamiento, pero empeora su rendimiento cuando tiene que realizar predicciones sobre imágenes radicalmente distintas, en nuestro caso, con otro tipo de herramienta y estructuras de fondo totalmente distintas.

En cuanto a las limitaciones que tiene este proyecto, estas son bastante claras. Podemos considerar que el modelo tiene un rendimiento muy bueno con el único conjunto de datos con el que se ha podido entrenar de manera óptima. En el caso del modelo entrenado con ambos conjuntos de datos, que ha obtenido los mejores resultados, se cree que su rendimiento podía haber alcanzado valores mucho más altos si el conjunto elaborado en el ITAP estuviese correctamente etiquetado. Es por ello que se considera que la limitación más importante del proyecto se trata fundamentalmente de este apartado, los datos de entrenamiento.

Como se ha comentado en la sección 5.1, encontrar una cantidad suficiente de este tipo de datos, correctamente etiquetados y en ambientes que reflejen la realidad de una forma fiable no es sencillo. En nuestro caso, el único conjunto de datos que realmente tiene la capacidad de entrenar correctamente un modelo se trata de imágenes extraídas de simulaciones que nada tienen que ver con una cirugía real. Además, el *dataset* que podía reproducir de una forma más fiable estas condiciones, pues se había creado exclusivamente con este propósito, contenía errores importantes que hacían imposible poder entrenar un modelo con él. Por último, el *dataset EndoVis'15* que contiene imágenes de simulaciones exvivo, no contiene las imágenes suficientes para realizar un entrenamiento con él, ni siquiera utilizando técnicas de aumento de datos.

Capítulo 8. Conclusiones y líneas futuras.

Este capítulo se expondrá cuáles de los resultados se han conseguido finalmente con este proyecto, cuáles son las conclusiones que podemos extraer de él y cuáles son las mejoras que se podrían realizar en un futuro con el objetivo de elaborar un algoritmo útil y robusto que suponga realmente una ayuda a la hora de llevar a cabo cirugías mínimamente invasivas.

8.1. Consecución de objetivos

En cuanto a los objetivos, es importante recordar lo que se expuso en la sección 1.3. El objetivo principal era construir un modelo capaz de predecir la posición de la herramienta de cirugía laparoscópica o robótica en tiempo real, de forma que se pueda integrar en el control de un robot quirúrgico que pueda asistir al cirujano.

Centrándonos en este objetivo, se puede afirmar que se ha cumplido en su totalidad. Con este proyecto se ha conseguido un modelo capaz de tomar imágenes en tiempo real y predecir la posición de la herramienta con una buena precisión. El único limitante que encontramos en este apartado es el hecho de que solamente ha sido entrenado y probado con robustez en imágenes de simulación, y no en cirugías reales, lo que limita las afirmaciones que podemos realizar sobre la capacidad que puede tener este modelo en situaciones de cirugía reales.

En cuanto a los distintos objetivos específicos que se buscaban alcanzar con este proyecto, podemos decir que se han conseguido cumplir todos excepto el objetivo específico 1. En este caso, sí que hemos conseguido un conjunto de datos robusto y amplio, y ha conseguido entrenar un modelo de red neuronal con resultados muy buenos, pero no es capaz de hacer una descripción completa del problema, ya que sus imágenes tienen diferencias claras con las que nos podemos encontrar en situaciones de cirugías reales. En este caso, no hemos conseguido ningún conjunto de datos apto para entrenar nuestro modelo y que contenga estas características.

El resto de los objetivos se han cumplido, ya que los resultados del modelo han sido realmente buenos y este se ha programado desde cero. Además, se ha elaborado una métrica basada en el valor máximo que nos permite ajustar un umbral de confianza a la hora de realizar predicciones, de forma que podemos eliminar aquellas que no superen un cierto valor.

Por último, se ha conseguido realizar un *script* capaz de tomar imágenes de un video o de una cámara y realizar predicciones sobre él en tiempo real, con una tasa de refresco apta para este tipo de aplicaciones.

8.2. Conclusiones

Una vez se han valorado cuáles de los objetivos han sido cumplidos y cuáles no, es necesario hacer un resumen de las conclusiones que podemos extraer de este proyecto.

En primer lugar, en un campo que lleva tan poco tiempo instalado en la sociedad, como el de la inteligencia artificial, existen multitud de modelos con finalidades similares a las nuestras. De hecho, también son numerosos los que se centran en la detección de objetos en imágenes. En cambio, el rendimiento de una arquitectura depende de distintos factores, como las funciones de coste, los datos o incluso el equipo que se utilice, sobre todo en el tiempo de inferencia.

En este caso, se programó un modelo con todas sus funcionalidades complementarias que parecía tener un buen rendimiento en su respectivo artículo. A partir de ese momento, se modificó este modelo tanto en el tamaño de entrada y salida de la arquitectura, como en las distintas capas que lo forman, con el objetivo de intentar mejorar su rendimiento.

Se ha podido ver que para el único *dataset* que, debido a su tamaño y variabilidad, ha sido capaz de entrenar correctamente el modelo, los resultados han sido excelentes, permitiendo realizar predicciones precisas y con una latencia más que suficiente para ser implementado en la arquitectura de control de un sistema robotizado.

Aunque mediante los resultados en este conjunto de datos podemos intuir que el modelo tiene un comportamiento apto para este tipo de aplicaciones, no podemos confirmar que su comportamiento se reproduzca con estos resultados entrenándolo con imágenes de cirugías reales, ya que no ha sido entrenado y probado con ello. Los resultados con el conjunto Atlas Dione apuntan a que su comportamiento será bueno, ya que este conjunto, aunque no está formado por imágenes realistas de cirugías, sí posee imágenes complicadas de localizar, con diversos artefactos y estructuras extrañas en la imagen.

Otro factor remarcable del modelo que podíamos suponer con antelación es la mayor importancia del *dataset* utilizado para entrenar el modelo sobre el número de filtros y la resolución de los datos de entrada. En el modelo que ha sido entrenado con un *dataset* combinado hemos podido apreciar que, si se mantiene una resolución mínima y el set de entrenamiento posee una mayor variabilidad, los resultados pueden ser igual o mejores que en modelos más complejos, con un menor tiempo de inferencia. Este factor es realmente importante a la hora de poder utilizar nuestro algoritmo en el mayor número de equipos posible.

En definitiva, se ha conseguido elaborar un modelo capaz de realizar predicciones en tiempo real sobre la posición de una herramienta de cirugía laparoscópica con una precisión suficiente para que un sistema robótico sea capaz de seguirlas. Aunque aún hay que realizar pruebas con otros tipos de imágenes para comprobar su rendimiento en situaciones más realistas, viendo su rendimiento actual es esperable que sus resultados sean buenos si el conjunto de datos con el que se entrena lo es.

8.3. Líneas futuras

Aunque los objetivos de este trabajo se han cumplido prácticamente en su totalidad, aún existen algunos pequeños aspectos que son necesarios mejorar antes de incorporar este modelo a sistemas robotizados reales.

El primero de ellos se basa en encontrar un *dataset* realista y correctamente etiquetado, con un número suficiente de imágenes para entrenar el modelo en situaciones más similares a las que se enfrentará en un futuro. En caso de no encontrarse, otra opción sería construir uno propio, pero en unas condiciones óptimas. Estas condiciones pasarían por poder tener acceso videos de cirugías reales etiquetadas por profesionales sanitarios o generar escenas simuladas basadas en operaciones reales con especialistas sanitarios que puedan valorar si las simulaciones se ajustan a una cirugía o no.

La segunda de las líneas futuras de este proyecto se basaría en utilizar un modelo correctamente entrenado para crear una interfaz que permita controlar la inferencia del modelo. Algunas de las funciones de esta interfaz podrían ser activar o desactivar el modelo, de forma que en un momento dado se pueda elegir entre una navegación automática o manual. Otra de sus funciones podría ser ajustar el umbral de detección a la imagen que se esté visualizando, pudiendo ser mayor en imágenes más sencillas y disminuyéndolo algo más en imágenes en las que la predicción no sea tan robusta

Por último, en un futuro sería necesario adaptar este algoritmo a su implementación en sistemas robotizados, integrándolo en un sistema operativo como ROS (*Robotic Operative System*), de forma que las coordenadas se envíen mediante mensajes a los robots y estos puedan adaptar su pose en base a estos datos, mediante otro algoritmo que, nuevamente, habría que programar.

Bibliografía

[1] E. P. W. van der Putten, R. H. M. Goossens, J. J. Jakimowicz, and J. Dankelman, "Haptics in minimally invasive surgery--a review," *Minim. Invasive Ther. Allied Technol.*, vol. 17, no. 1, pp. 3–16, 2008.

[2] A. Darzi and Y. Munz, "THE IMPACT OF MINIMALLY INVASIVE SURGICAL TECHNIQUES," *Annu. Rev. Med*, vol. 55, pp. 223–260, 2003.

[3] S. J. Spaner and G. L. Warnock, "A Brief History of Endoscopy, Laparoscopy, and Laparoscopic Surgery," *https://home.liebertpub.com/lap*, vol. 7, no. 6, pp. 369–373, Jan. 2009.

[4] P. Lopez and M. Amada, "'BENEFICIO ECONÓMICO Y SOCIAL DE LA HOSPITALIZACIÓN DOMICILIARIA'."

[5] "Principales resultados Sistema de Cuentas de Salud," 2020.

[6] B. Jaffray, "Minimally invasive surgery," *Arch. Dis. Child.*, vol. 90, no. 5, pp. 537–542, May 2005.

[7] A. Romero-Tamarit, R. Reig-Viader, M. D. Estrada-Sabadell, and M. Espallargues-Carreras, "Eficacia, efectividad, seguridad y eficiencia de la cirugía robótica con el sistema quirúrgico Da Vinci," *Scientia*, 2020.

[8] M. De Prado, M. Cuervas-Mons, and V. De Prado, "Open vs Minimally Invasive Surgery: Advantages and Disadvantages," *Foot Ankle Disord.*, pp. 43–69, 2022.

[9] "About - Mayo Clinic." [Online]. Available: https://www.mayoclinic.org/tests-procedures/minimally-invasive-surgery/about/pac-20384771?p=1. [Accessed: 31-May-2023].

[10] E. Patel, S. Saikali, A. Mascarenhas, M. C. Moschovas, and V. Patel, "Muscle fatigue and physical discomfort reported by surgeons performing robotic-assisted surgery: a multinational survey," *J. Robot. Surg.*

[11] E. Liatsikos, A. Tsaturyan, I. Kyriazis, P. Kallidonis, D. Manolopoulos, and A. Magoutas, "Market potentials of robotic systems in medical science: analysis of the Avatera robotic system," *World J. Urol.*, vol. 40, no. 1, pp. 283–289, Jan. 2022.

[12] D. Stephan, H. Sälzer, and F. Willeke, "First Experiences with the New Senhance® Telerobotic System in Visceral Surgery," *Visc. Med.*, vol. 34, no. 1, pp. 31–36, Feb. 2018.

[13] M. Raffaelli *et al.*, "The new robotic platform Hugo™ RAS for lateral transabdominal adrenalectomy: a first world report of a series of five cases," *Updates Surg.*, vol. 75, no. 1, pp. 217–225, Jan. 2023.

[14] A. Race and S. Horgan, "Overview of Current Robotic Technology," *Innov. Endosc. Surg. Technol. GI Tract*, pp. 1–17, 2021.

[15] V. Elorrieta, J. Villena, Á. Kompatzki, A. Velasco, and J. A. Salvadó, "ROBOT Assisted Laparoscopic Surgeries For Nononcological Urologic Disease: Initial Experience With Hugo Ras System," *Urology*, vol. 174, pp. 118–125, Apr. 2023.

[16] O. A. Castillo, R. Sánchez-Salas, and C. Octavio Castillo, "CIRUGÍA ROBÓTICA EN UROLOGÍA BASES LAPAROSCOPICAS DE LA CIRUGIA ROBOTICA," *Arch. Esp. Urol*, vol. 60, pp. 4–357, 2007.

[17] V. K. Narula *et al.*, "A computerized analysis of robotic versus laparoscopic task performance," *Surg. Endosc. Other Interv. Tech.*, vol. 21, no. 12, pp. 2258–2261, Dec. 2007.

[18] J. A. Van Koughnett, S. Jayaraman, R. Eagleson, D. Quan, A. Van Wynsberghe, and C. M. Schlachta, "Are there advantages to robotic-assisted surgery over laparoscopy from the surgeon's perspective?," *J. Robot. Surg.*, vol. 3, no. 2, pp. 79–82, Jun. 2009.

[19] R. E. Perez and S. D. Schwaitzberg, "Robotic surgery: finding value in 2019 and beyond," *Ann. Laparosc. Endosc. Surg.*, vol. 4, no. 0, May 2019.

[20] A. Pandya *et al.*, "A Review of Camera Viewpoint Automation in Robotic and Laparoscopic Surgery," *Robot. 2014, Vol. 3, Pages 310-329*, vol. 3, no. 3, pp. 310–329, Aug. 2014.

[21] H. Cheng *et al.*, "Prolonged operative duration is associated with complications: a systematic review and meta-analysis," *J. Surg. Res.*, vol. 229, pp. 134–144, Sep. 2018.

[22] P. Fiorini, K. Y. Goldberg, Y. Liu, and R. H. Taylor, "Concepts and Trends in Autonomy for Robot-Assisted Surgery," *Proc. IEEE*, vol. 110, no. 7, pp. 993–1011, Jul. 2022.

[23] K. Omote *et al.*, "Self-guided robotic camera control for laparoscopic surgery compared with human camera control," *Am. J. Surg.*, vol. 177, no. 4, pp. 321–324, Apr. 1999.

[24] "ISO 8373:2012(en), Robots and robotic devices — Vocabulary." [Online]. Available: https://www.iso.org/obp/ui/#iso:std:iso:8373:ed-2:v1:en. [Accessed: 30-May-2023].

[25] B. Baykant Alagoz, "A Note on Depth Estimation from Stereo Imaging Systems."

[26] J. Cartucho, C. Wang, B. Huang, D. S. Elson, A. Darzi, and S. Giannarou, "An enhanced marker pattern that achieves improved accuracy in surgical tool tracking," *https://doi.org/10.1080/21681163.2021.1997647*, vol. 10, no. 4, pp. 400–408, 2021.

[27] X. Liu, S. Kang, W. Plishker, G. Zaki, T. D. Kane, and R. Shekhar, "Laparoscopic stereoscopic augmented reality: toward a clinically viable electromagnetic tracking solution," *https://doi.org/10.1117/1.JMI.3.4.045001*, vol. 3, no. 4, p. 045001, Oct. 2016.

[28] Y. Wang, Q. Sun, Z. Liu, and L. Gu, "Visual detection and tracking algorithms for minimally invasive surgical instruments: A comprehensive review of the state-of-the-art," *Rob. Auton. Syst.*, vol. 149, p. 103945, Mar. 2022.

[29] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation Tech report (v5)."

[30] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection."

[31] C. C. Aggarwal, "Neural Networks and Deep Learning."

[32] M. Nielsen, "Neural Networks and Deep Learning."

[33] D. Sarikaya, J. J. Corso, and K. A. Guru, "Detection and Localization of Robotic Tools in Robot-Assisted Surgery Videos Using Deep Neural Networks for Region Proposal and Detection," *IEEE Trans. Med. Imaging*, vol. 36, no. 7, pp. 1542–1549, Jul. 2017.

[34] "Data - Grand Challenge." [Online]. Available: https://endovissub-instrument.grand-challenge.org/Data/. [Accessed: 10-Jul-2023].

[35] M. Abadi *et al.*, "TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems."

[36] Y. Liu, Z. Zhao, F. Chang, and S. Hu, "An Anchor-Free Convolutional Neural Network for Real-Time Surgical Tool Detection in Robot-Assisted Surgery."

[37] A. Newell, K. Yang, and J. Deng, "Stacked Hourglass Networks for Human Pose Estimation."

[38] F. Chollet, "Xception: Deep Learning with Depthwise Separable Convolutions."

[39] Hei Law, "CornerNet: Detecting Objects as Paired Keypoints."

[40] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal Loss for Dense Object Detection."

[41] M. Motamed, "Approximation Power of Deep Neural Networks An explanatory mathematical survey," 2022.

[42] C. Krittanawong *et al.*, "Deep learning for cardiovascular medicine: a practical primer," *Eur. Heart J.*, vol. 40, no. 25, pp. 2058–2073, Jul. 2019.

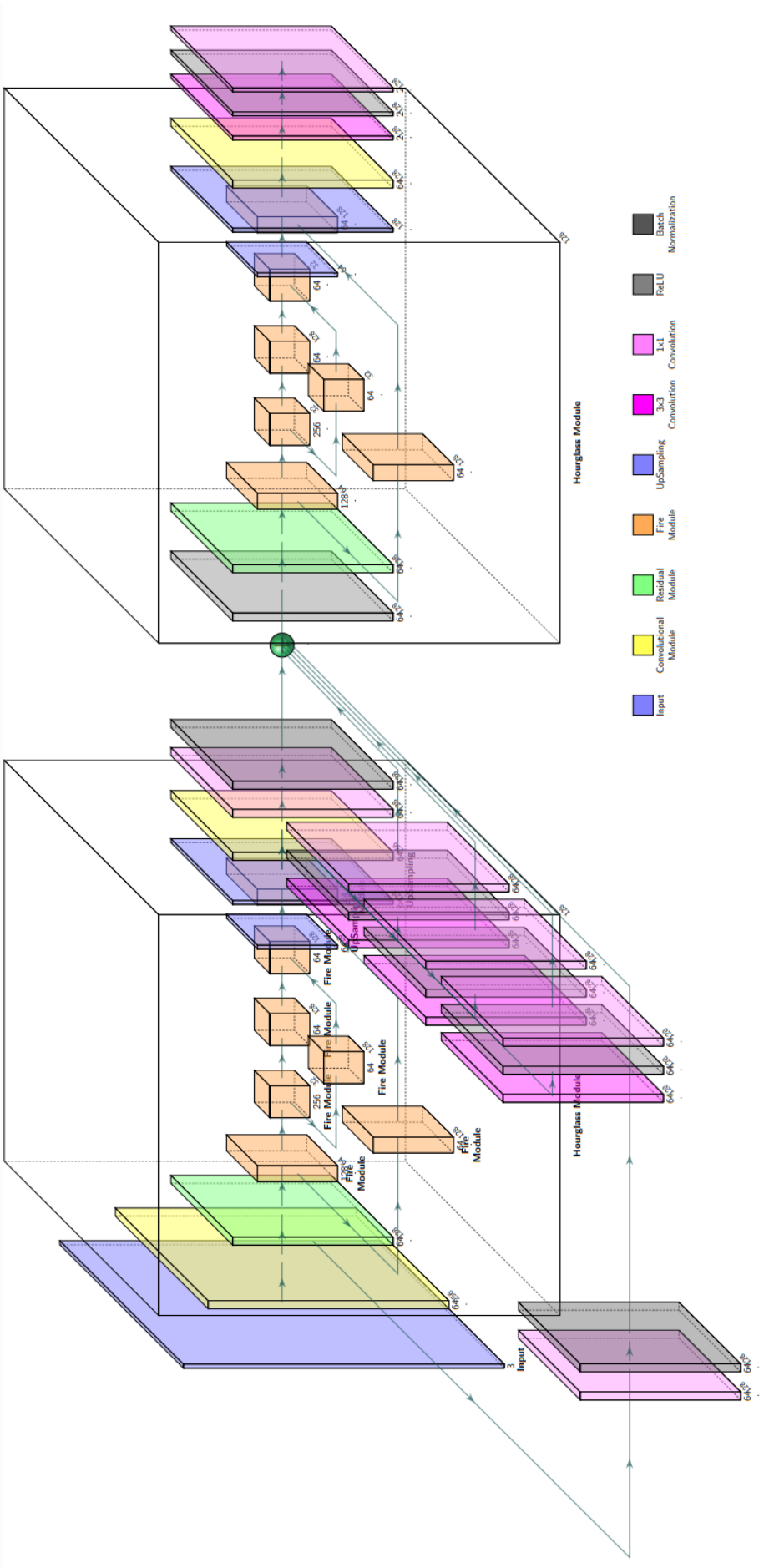
[43] C. C. Aggarwal, "Advanced Optimization Solutions," *Linear Algebr. Optim. Mach. Learn.*, pp. 205–253, 2020.

[44] S. Salman and X. Liu, "Overfitting Mechanism and Avoidance in Deep Neural Networks."

[45] R. Gençay and M. Qi, "Pricing and hedging derivative securities with neural networks: Bayesian regularization, early stopping, and bagging," *IEEE Trans. Neural Networks*, vol. 12, no. 4, pp. 726–734, Jul. 2001.

[46] M. Lyra, A. Ploussi, and A. Georgantzoglou, "MATLAB as a Tool in Nuclear Medicine Image Processing," *MATLAB - A Ubiquitous Tool Pract. Eng.*, Oct. 2011.

Anexo I


Hourglass Module
Hourglass Module
Fire Module
Fire Module
Fire Module
Fire Module
Fire Module
Fire Module
Fire Module
Input
Batch Normalization
ReLU
1x1 Convolution
3x3 Convolution
UpSampling
Fire Module
Residual Module
Convolutional Module
Input