
**UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO**

**FACULTAD DE INGENIERÍA**


# Desarrollo de una red neuronal convolucional para el procesamiento de imágenes placentarias

**TESIS**

Que para obtener el título de

**Ingeniero Mecatrónico**

**P R E S E N T A**


Omar Emilio Contreras Zaragoza

**DIRECTORA DE TESIS**

Dra. Fabiola Miroslaba Villalobos Castaldi



**Ciudad Universitaria, Cd. Mx., 2018**

# Índice


I. Introducción ..... 3
I.I Objetivos ..... 3
i. Objetivo General ..... 3
ii. Objetivos particulares ..... 4
I.II Hipótesis ..... 4
I.III Antecedentes ..... 4
I.IV Estado del Arte ..... 5
i. Segmentación con CNN ..... 5
ii. Segmentación CNN para aplicaciones médicas ..... 6
II. Síndrome de Transfusión Gemelo-Gemelo ..... 8
II.I Procedimiento de Coagulación Láser ..... 9
II.II Resultado del embarazo ..... 11
II.III Problemas con la Coagulación Láser ..... 13
III. Marco Teórico ..... 15
III.I Clasificación de Imágenes ..... 15
III.II Redes Neuronales Convolucionales ..... 17
i. Capa Convolucional ..... 19
ii. Capa Pooling ..... 26
iii. Capa totalmente conectada ..... 27
iv. Entrenamiento ..... 27
v. Función de costo ..... 27
vi. Backpropagation ..... 28
vii. Optimizador ..... 28
viii. Prueba ..... 32
ix. Diagnóstico ..... 32
IV. Segmentación ..... 34
IV.I Base de datos ..... 34
IV.II Etiquetado ..... 35
IV.III Entrenamiento y Prueba ..... 38
IV.IV Partición ..... 38

IV.V Métricas ..... 39
i. ROC ..... 41
ii. Precision-Recall ..... 41
iii. Jaccard Similarity ..... 42
iv. Matthews Correlation Coefficient ..... 42
V. Iteraciones ..... 42
V.I Arquitectura Base ..... 43
V.II Segunda arquitectura ..... 45
V.III Más filtros ..... 48
V.IV Parámetros ..... 50
V.V Comparación ..... 52
V.VI Arquitectura final ..... 54
VI. Validación ..... 59
VII. Conclusiones y trabajo futuro ..... 63
VIII. Bibliografía ..... 65

# I. Introducción

En el trabajo se le da una nueva aplicación a una red neuronal convolucional profunda en la que se segmentan venas y arterias de una imagen placentaria inyectada con colorante para facilitar su análisis y poder obtener más información sobre el síndrome de transfusión gemelo-gemelo, se presenta la metodología utilizada con la que se llegó a una arquitectura novedosa y a través de la cual se llega a muy buenos resultados que también se exponen numéricamente y visualmente.

La arquitectura presentada obtiene excelentes resultados con poco tiempo de entrenamiento, gracias a la baja cantidad de parámetros y las pocas capas que posee que disminuyen el costo computacional de la red neuronal sin comprometer el desempeño de las predicciones que hace. Además, las predicciones son de buena calidad, obteniendo resultados que logran evitar uno de los más grandes problemas presentados en el campo de la segmentación semántica de imágenes, borde borrosos debido a la perdida de información durante las capas de la red neuronal convolucional.

En el primer capítulo se muestran los antecedentes de las redes neuronales convolucionales, se presenta el estado del arte y las aplicaciones de segmentación que ya existen. En el segundo capítulo se introduce al lector el síndrome de transfusión gemelo-gemelo, sus consecuencias, riesgos y los tratamientos actuales. En el tercer capítulo se tratan las redes neuronales convolucional, explicando las capas que la conforman, la forma en la que las redes lograr aprender para poder predecir el resultado de nuevas imágenes, se explica el entrenamiento y prueba y posibles mejoras para obtener un mejor rendimiento. En el cuarto capítulo se exponen las métricas que se utilizan para obtener parámetros que den una referencia de la calidad del desempeño que se obtiene de las redes neuronales. EL quinto capítulo muestra las principales iteraciones propuestas, en las que se fueron haciendo diferentes cambios en busca de una mejor predicción, se presentan las métricas de cada iteración y se incluyen imágenes para mostrar la predicción lograda por cada una de esas iteraciones. En el sexto capítulo se valida el diseño de la arquitectura de la red neuronal convolucional utilizando una base de datos de gran calidad y comparando los resultados obtenidos por otras publicaciones con esa red para obtener una referencia de la calidad de la red neuronal. En el séptimo y último capítulo se habla del trabajo que se puede hacer para obtener mejores resultados, así como de lo más destacado de la metodología presentada y lo que se aprendió del trabajo realizado.

## I.I Objetivos

### i. Objetivo General

Diseño, desarrollo e implementación de una red neuronal convolucional capaz de segmentar venas y arterias del resto de imágenes placentarias inyectadas con colorante para ayudar a la cirugía de coagulación láser y evitar anastomosis residuales para mejorar el porcentaje de supervivencia de bebés nacidos con el síndrome de transfusión gemelo-gemelo.

### ii. Objetivos particulares

a. Crear una base de datos de imágenes placentarias inyectadas con colorante con el síndrome transfusión gemelo-gemelo con el fin de validar algoritmos de análisis de imágenes basados en redes neuronales convolucionales.

b. Entrenar, probar y validar la red neuronal convolucional.

c. Validar con diferentes configuraciones de red neuronal convolucional para obtener métricas de desempeño y elegir la que mejore el rendimiento del sistema.

## I.II Hipótesis

¿A través del entrenamiento de una red profunda de CNNs con una base de datos de imágenes placentarias inyectadas con colorante es posible segmentar dichas imágenes en una red arteriovenosa para facilitar su análisis?

## I.III Antecedentes

El Deep learning es una técnica que ha ido creciendo en el análisis de datos. El aprendizaje profundo es una mejora de las redes neuronales artificiales, que consta de más capas que permiten mayores niveles de abstracción y predicciones mejoradas a partir de datos. Hasta la fecha ha emergido como la principal herramienta de aprendizaje automático en los dominios de visión computarizada [1].

En particular las redes neuronales convolucionales (CNNs) han probado ser poderosas herramientas para una amplia gama de tareas de visión computarizada. Deep CNNs automáticamente aprenden abstracciones de nivel medio y de alto nivel obtenidas a partir de datos brutos. Los resultados recientes indican que las características extraídas son extremadamente efectivas en el reconocimiento y localización de objetos en imágenes naturales. Los grupos de imágenes médicas en todo el mundo están ingresando al campo rápidamente y están aplicando CNN y otras metodologías de Deep learning para una amplia variedad de aplicaciones [1].

En imágenes médicas, el diagnóstico o evaluación precisa de una enfermedad depende tanto de la adquisición de la imagen como de la interpretación de la misma. La adquisición de imágenes ha mejorado sustancialmente en los últimos años y los dispositivos adquieren datos a mayor velocidad y aumentan la resolución. Sin embargo, el proceso de interpretación de imágenes recientemente comenzó a beneficiarse de la tecnología informática. La mayoría de las interpretaciones de imágenes médicas son realizadas por doctores; sin embargo, la interpretación de imágenes por humanos es limitada debido a su subjetividad, grandes variaciones entre intérpretes y fatiga [1].

Muchas tareas de diagnóstico requieren un proceso de búsqueda inicial para detectar anomalías y cuantificar las mediciones y los cambios a lo largo del tiempo. Las herramientas computarizadas, específicamente el análisis de imágenes y el machine learning, son los habilitadores claves para mejorar el diagnóstico, al facilitar la identificación de los hallazgos que requieren tratamiento y para apoyar el flujo de

trabajo del experto. Entre estas herramientas el Deep learning está demostrando ser la base del estado de arte, obteniendo una mejora en precisión [1].

## I.IV Estado del Arte

Hoy en día las redes neuronales convolucionales han tenido gran éxito en reconocimiento y segmentación de imágenes, propiciando avances en campos como automóviles autónomos, reconocimiento facial y su uso ha ido en aumento en el campo de la medicina para diagnóstico o material de apoyo para doctores. A continuación, se mencionan lo más destacada en cuanto a segmentación de imágenes y al trabajo de segmentación hecho en la medicina.

### i. Segmentación con CNN

La segmentación semántica tiene una amplia gama de aplicaciones que van desde la comprensión de escenas, hasta la conducción autónoma. SegNet es una red neuronal convolucional diseñada para ser una arquitectura eficiente para la segmentación semántica en pixeles. Está principalmente motivada por aplicaciones de comprensión de la escena del camino que requieren la capacidad de modelar la apariencia (camino, edificio), la forma (automóviles, peatones) y comprender la relación espacial (contexto) entre diferentes clases, como el camino y la banqueta. El motor también tiene la capacidad de delinear objetos en función de su forma a pesar de su pequeño tamaño [2].

***Figura 1*** *Predicciones de SegNet en escenas de carretera [2].*

En DeconvNet proponen un nuevo algoritmo de segmentación semántica por el aprendizaje de una red deconvolucional. Se compone de capas convolucionales y capas deconvolucionales, las deconvolucionales están compuestas también por capas unpooling. El método de segmentación identifica estructuras detalladas y maneja objetos de múltiples escalas de manera natural [3].

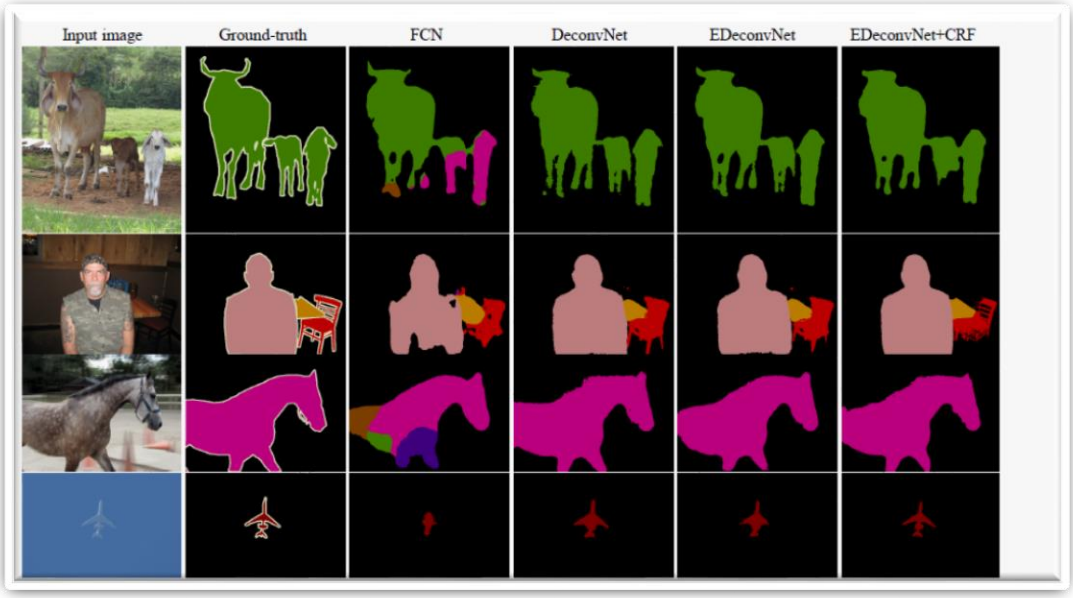



***Figura 2*** *Varias propuestas de DeconvNet comparadas con las imágenes etiquetadas manualmente (ground truth) y con una red con capas totalmente conectadas (FCN) [3].*

### ii. Segmentación CNN para aplicaciones médicas

La condición de la red vascular del ojo humano es un importante factor de diagnóstico en oftalmología. Su segmentación en la imagen del fondo de ojo es una tarea no trivial debido al tamaño variable de los vasos, el contraste relativamente bajo y la presencia potencial de patologías como micro aneurismas y hemorragias. Muchos algoritmos, tanto no supervisados como supervisados, se han propuesto para este propósito en el pasado. En [4] proponen una técnica de segmentación supervisada que utiliza una red neuronal profunda capacitada en una muestra grande (hasta 400 000) de ejemplos preprocesados con normalización de contraste global, blanqueamiento de fase cero y aumento de datos mediante transformaciones geométricas y correcciones gamma [4].

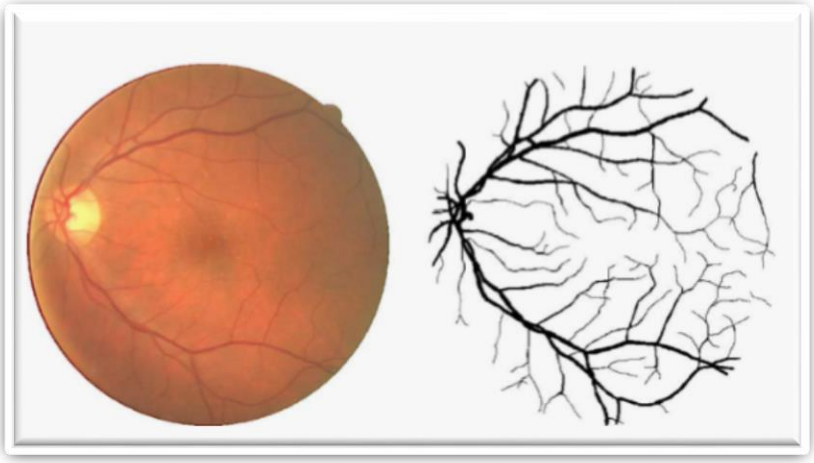


***Figura 3*** *Segmentación de imágenes de fondo de ojo [4].*

Esta aplicación de fondo de ojo es muy parecida a lo que se propone en este trabajo dado que se trata de separar venas y arterias del resto de la imagen y el resultado será utilizado para ayudar en el tratamiento y diagnóstico de los pacientes. Las técnicas de Deep learning cada vez son más utilizadas en la medicina.
En U-net, presentan una red y una estrategia de capacitación que se basa en el fuerte uso del aumento de datos para utilizar las muestras etiquetadas disponibles de manera más eficiente. La arquitectura consiste en un camino de contracción (disminuir el tamaño de la imagen) para capturar el contexto y una ruta simétrica de expansión (aumentar el tamaño de la imagen) que permite la localización precisa. Utilizan muy pocas imágenes para la segmentación de estructuras neuronales en conjuntos de imágenes de microscopios electrónicos [5].

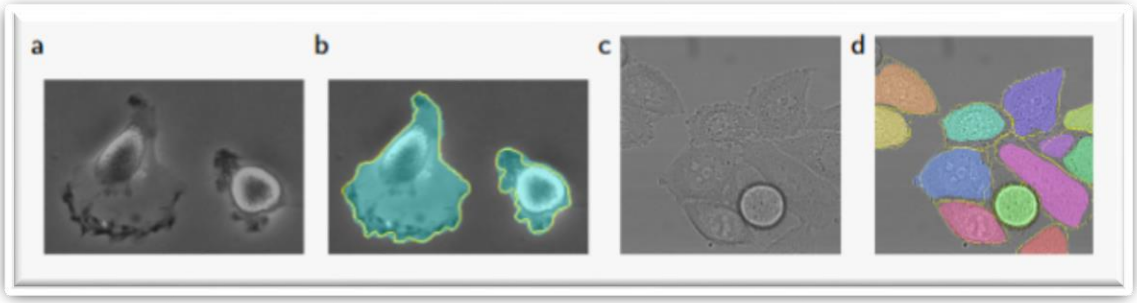



***Figura 4*** *Resultados de U-Net. (a) parte de la imagen de entrada, (b) resultados de segmentación con etiquetado manual (borde amarillo), (c) imagen de entrada, (d) segmentación con etiquetado manual (borde amarillo) [5].*

Las redes neuronales convolucionales cada vez muestran mejores resultados en la tarea de segmentación y gracias a la creación de varias competencias como ImageNet [6] o ISBI [7] y a bases de datos específicas como DRIVE [8] o STARE [9] para el fondo de ojo o las imágenes neuronales de microscopios electrónicos o CamVid, imágenes de escenas de carretera, han favorecido este crecimiento puesto que grandes cantidades

de imágenes con etiquetas manuales, facilitan la tarea para el Deep learning por la gran cantidad de datos y que los diseñadores no necesitan invertir tiempo en conseguir más.

# II. Síndrome de Transfusión Gemelo-Gemelo

En este capítulo se explica el síndrome de transfusión gemelo-gemelo en los embarazos, su origen, el daño que causa a los bebés y el riesgo que les causa. Posteriormente, se mencionan los tratamientos que existen y los resultados que han tenido los bebés después de ser tratados. Finalmente se tratan las complicaciones que pueden presentarse en la cirugía y áreas en las que la tecnología puede ayudar.

En partos con gemelos monocigóticos, la separación del embrión ocurre dentro de los 3 días posteriores a la fertilización en aproximadamente un tercio de los casos, resultando en dos fetos separados con circulaciones de placenta independientes. La separación de embrión posterior al tercer día es asociada con comunicaciones vasculares entre las placentas; cuando la segmentación se retrasa después del día 12, los fetos están unidos. Un desbalance en el flujo neto de sangre a través de las comunicaciones vasculares de la placenta de un feto (donador) al otro (receptor) resulta en el síndrome de transfusión gemelo-gemelo, que ocurre aproximadamente en el 15% de embarazos con gemelos monocigóticos [10].

Las conexiones anormales de los vasos sanguíneos provocan que los fetos compartan el suministro de sangre, y el gemelo más pequeño (donante) bombea sangre al gemelo más grande (receptor). Debido a que tiene más sangre, el gemelo receptor produce demasiada orina, lo que puede agrandar la vejiga, producir demasiado líquido amniótico y causar insuficiencia cardíaca. El gemelo donante tiene niveles más bajos de sangre, líquido amniótico y orina, y una vejiga más pequeña [11].

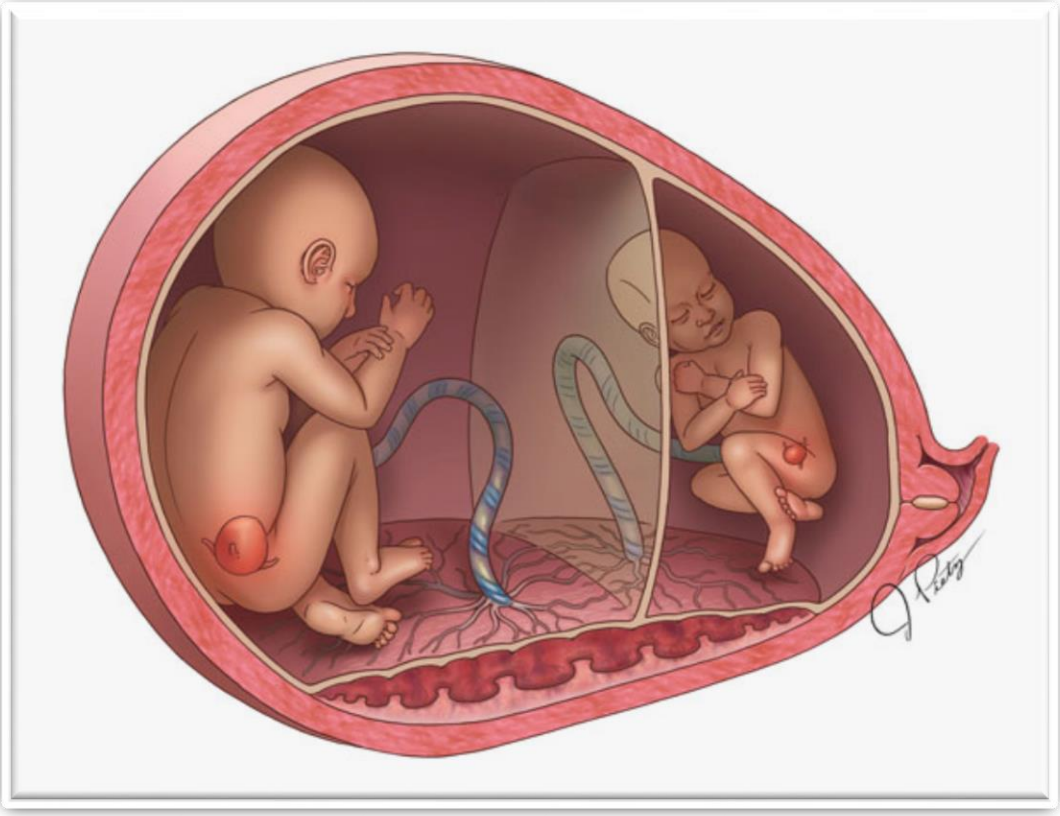


***Figura 5*** *Síndrome de transfusión gemelo-gemelo [11].*

Aunque la supervivencia puede mejorar mediante amniodrenaje, sigue existiendo un riesgo grave de discapacidad en 15 a 50% de los sobrevivientes. Un desarrollo más reciente en el manejo de esta condición en fetoscopía y coagulación láser para interrumpir las comunicaciones vasculares de la placenta entere los gemelos que podría constituir el mecanismo subyacente del síndrome. Datos preliminares sugieren que la supervivencia de por lo menos uno de los gemelos se puede alcanzar en aproximadamente el 70% de los embarazos y esta mejora de supervivencia también está asociada a una reducción substancial del riesgo de discapacidad (5%). Resultados de un centro de estudios en coagulación láser endoscópica reportan en el manejo de 132 embarazos con casos severos del síndrome de transfusión de gemelo-gemelo, incluyendo por lo menos un año de seguimiento [12].

## II.I Procedimiento de Coagulación Láser

Se realiza un ultrasonido detallado para localizar la placenta, la membrana amniótica entre los gemelos, la inserción placentaria de los cordones umbilicales y para determinar cualquier anomalía fetal. Se realiza una evaluación Doppler de la arteria umbilical para determinar la presencia o ausencia de flujo diastólico final. El sitio apropiado de entrada en el abdomen materno se escoge para evitar una lesión a la placenta o el feto y para permitir acceso a través del saco receptor hacia la membrana entre gemelos, idealmente perpendicular al eje del feto donador [12].

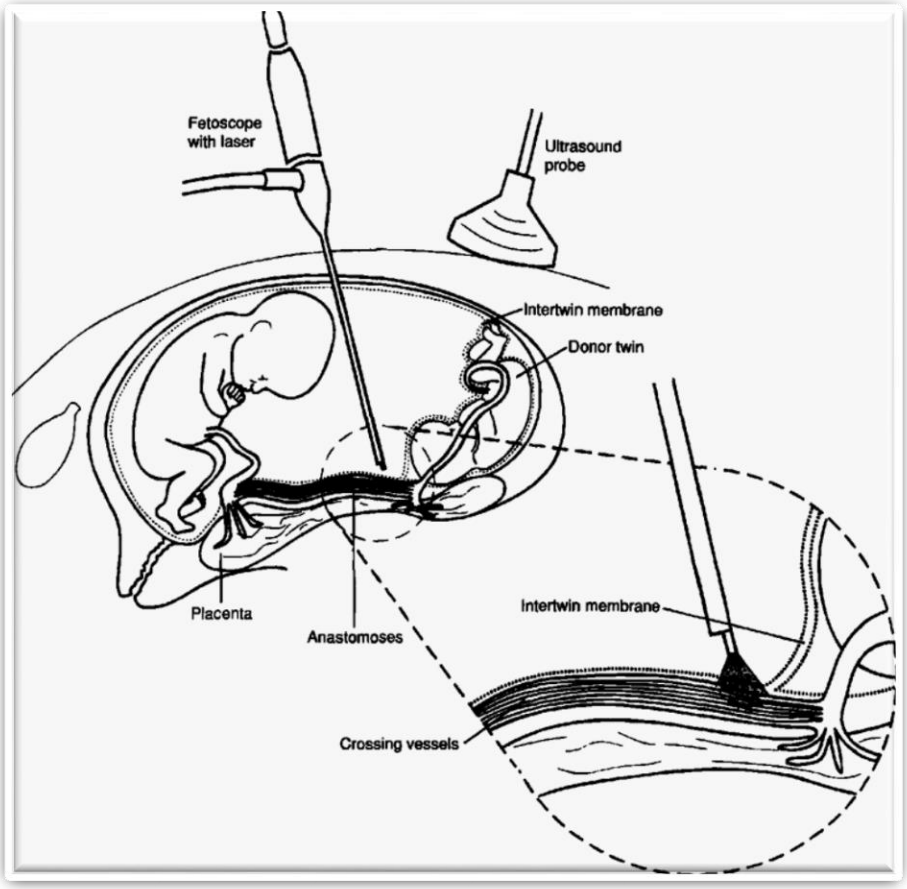



***Figura 6*** *Diagrama de la organización espacial de la coagulación laser [12].*

A la madre se le da sedación IV y anestesia local. Una epidural también puede darse, dependiendo de la localización de la placenta. Se administra medicina para prevenir el parto. El cirujano hace una pequeña incisión en el abdomen de la madre y un trocar (diminuto tubo metálico) se inserta en el útero [11].

El fetoscopio se pasa a través del trocar para examinar y mapear las conexiones de los vasos sanguíneos en la superficie de la placenta compartida por los gemelos. La imagen es vista en un monitor de computadora grande. Se usa ultrasonido continuamente para monitorear el procedimiento. Un láser es usado para sellar vasos sanguíneos anormales y desconectarlos permanentemente, después el cirujano drena el exceso de flujo amniótico a través del trocar antes de removerlo [11].

Después de sellar los vasos sanguíneos entre gemelos, el cirujano pasa el láser por una línea entre las conexiones para incluso coagular venas más pequeñas de un lado al otro. Este proceso se llama solomonización. Proceso responsable por un mejor porcentaje de supervivencia de ambos gemelos. Después del procedimiento, a la madre se le da medicina para prevenir el parto prematuro. La mayoría de

los pacientes permanecen en el hospital por un día. Cinco días después de la cirugía, se hacen ultrasonido y ecocardiografía fetal para evaluar la condición de los fetos [11].

***Figura 7*** *Solomonización [11].*

## II.II Resultado del embarazo

En 97 de los 132 embarazos (73%) había por lo menos un sobreviviente. En general, 144 de 264 fetos (55%) sobrevivieron. En 47 embarazos (36%) ambos bebés sobrevivieron, en 50 (38%) un bebé sobrevivió y en 35 (26%) no hubo sobrevivientes [12].


***Figura 8*** *Resultado de la coagulación láser endoscópica para el síndrome de transfusión gemelo-gemelo. Las líneas representan la duración del tiempo entre el procedimiento y el nacimiento de cada uno de los 132 embarazos tratados [12].*

Con cirugía endoscópica, la supervivencia se mantiene estable en el 55% de los fetos sin importar el tiempo, ni el centro en el que se atendieron y más del 70% de los embarazos con por lo menos un sobreviviente. En contraste, con el método alternativo de amniodrenaje serial, existen marcadas diferencias [12].

Con cirugía endoscópica, el porcentaje de discapacidad neurológica es <5% que es mucho menor que en embarazos tratados con amniodrenaje serial (discapacidad severa en 15% de los sobrevivientes) [12].

Causas de muerte fetal asociada a cirugía endoscópica:

1. Aborto involuntario con diversas contribuciones de polihidramnios causado por la misma enfermedad y el procedimiento en sí [12].

2. Muerte intrauterina, más común del feto donador. El crecimiento severamente restringido del feto donante, con características de insuficiencia placental primaria e hipoxia, es comúnmente sujeto a trauma adicional por obliteración aguda en parte de su placenta, como resultado de la coagulación de las venas que cruzan la membrana entre los gemelos [12].

3. Normalmente presentado como muerte intrauterina de ambos bebés, puede reflejar el fracaso de la cirugía para separar las venas anastomóticos. Como existen muchas anastomosis entre los gemelos, la coagulación incompleta de todas las venas podría acelerar el fallecimiento de los fetos empeorando la derivación de sangre entre gemelos [12].

## II.III Problemas con la Coagulación Láser

Estudios de inyección de placenta con colorante se desarrollaron para determinar la localización, tamaño, tipo y número de anastomosis residuales (AR). Han demostrados que los AR pueden estar presentes después de la cirugía y pueden ser detectados en el 33% de placentas. AR puede llevar a una recurrencia en el síndrome de transfusión gemelo-gemelo en 14% de los casos y secuencia de anemia-policitemia gemelar en 13% de los casos [13].

La mayoría de los AR están situados cerca del margen de la placenta. Las anastomosis no coaguladas (56%) se encontraban a 2 cm del margen de la placenta. Existen dos posibles explicaciones para esto: primero, el margen de la placenta puede ser difícil de visualizar durante la cirugía fetoscópica por razones técnicas asociadas a la posición y el ángulo del fetoscopio. Idealmente, el fetoscopio debería ser insertado perpendicularmente al ecuador vascular para permitir visión clara de todo el ecuador vascular. Pero, esto no siempre es posible, particularmente cuando la placenta se posiciona sobre la pared uterina anterior. Un ángulo sub óptimo del fetoscopio puede prevenir una completa visión del ecuador vascular y en particular del margen de la placenta. Una segunda posible explicación es que el margen de la placenta puede ser menos escudriñado durante la cirugía láser fetoscópica, incluso en la ausencia del problema del ángulo del fetoscopio [13].

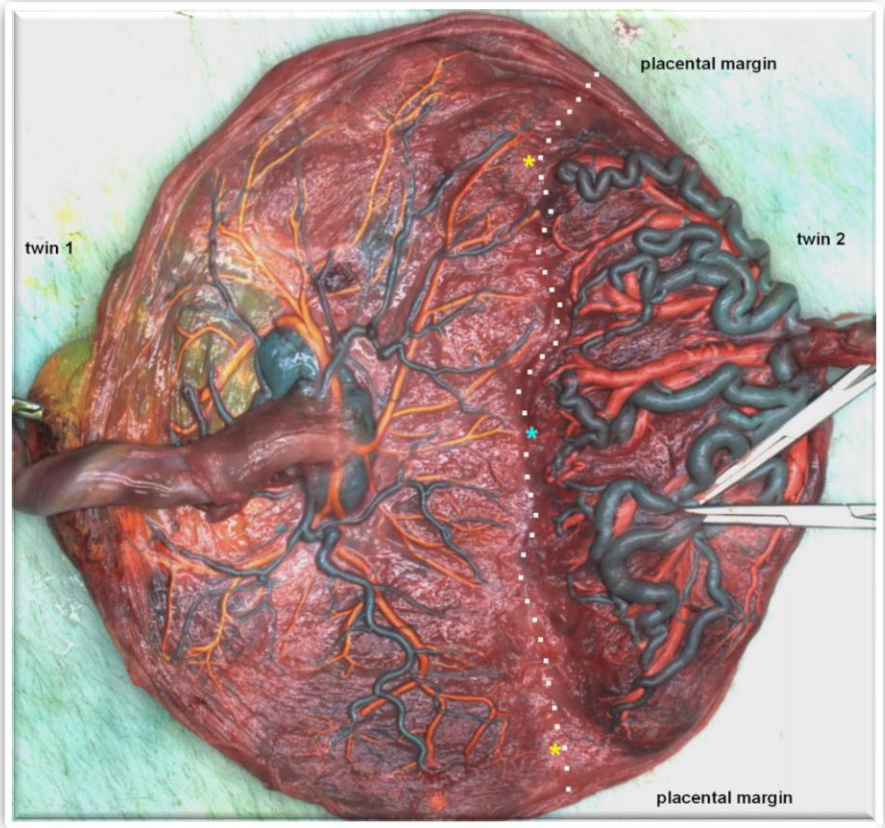



***Figura 9*** *Placenta con el síndrome de transfusión gemelo-gemelo tratado con cirugía láser a la semana 17 de gestación y nacimiento a las 33 semanas de gestación. Los puntos blancos muestran el ecuador vascular y la estrella azul indica el centro del ecuador vascular. La estrella amarilla indica 2 pequeñas AR cerca del margen de la placenta: una anastomosis arteriovenosa (parte superior de la imagen) en el 27% de distancia del margen de la placenta (con relación al centro de la placenta) y una anastomosis venovenosa (parte inferior de la imagen) en el 17% de distancia desde el margen de la placenta [13].*

La mayoría de los AR son muy pequeños (1 mm de diámetro). Las anastomosis son apenas visibles en la examinación de placenta con colorante y obviamente son aún más difíciles de detectar durante la fetoscopía que se hizo semanas antes. Las venas de la placenta del donador puede que hayan colapsado en el tiempo de la fetoscopía por hipovolemia y vasoconstricción [13].

De aquí surge la idea de presentar una solución que sea capaz de detectar con mayor precisión las anastomosis, ayudando a facilitar el análisis de placentas post cirugía. Usando visión computarizada para poder reconocer cualquier anastomosis sin importar su tamaño, ni su posición incluso si se encuentra muy cerca del margen y así poder agilizar el estudio del síndrome gemelo-gemelo, buscando a través del mismo mejorar el porcentaje de supervivencia, también mejorando el porcentaje de efectividad de la cirugía de coagulación láser, disminuyendo el índice de reincidencia del síndrome de transfusión gemelo-gemelo y sus otras consecuencias.

Este medio podría darle a futuro, una gran ayuda al cirujano al tener una herramienta capaz de extraer características de las placentas en poco tiempo y a través de eso, encontrar patrones como podrían ser ancho o bifurcaciones de venas y arterias, tortuosidad, número de anastomosis promedio por placenta o alguna otra característica que le pueda dar mayor información al cirujano para que se facilite la cirugía de coagulación.

# III. Marco Teórico

## III.I Clasificación de Imágenes

Clasificación de imágenes es la tarea de asignar una etiqueta de un conjunto fijo de categorías a una imagen de entrada. Este es uno de los problemas principales en visión de computadora, a pesar de su simplicidad, tiene una gran variedad de aplicaciones [14].

Para la computadora una imagen se representa como un arreglo tridimensional de números. En este ejemplo, el gato de la imagen es de 248 pixeles de ancho, 400 pixeles de alto y tiene 3 canales de color, rojo, verde y azul (RGB por sus siglas en inglés). Por lo tanto, la imagen consiste en 248 x 400 x 3 números o un total de 297,600 números. Cada número es un entero que va desde 0 (negro) hasta 255 (blanco). La tarea consiste en convertir este cuarto de millón de números en una sola etiqueta, como “gato” [14].

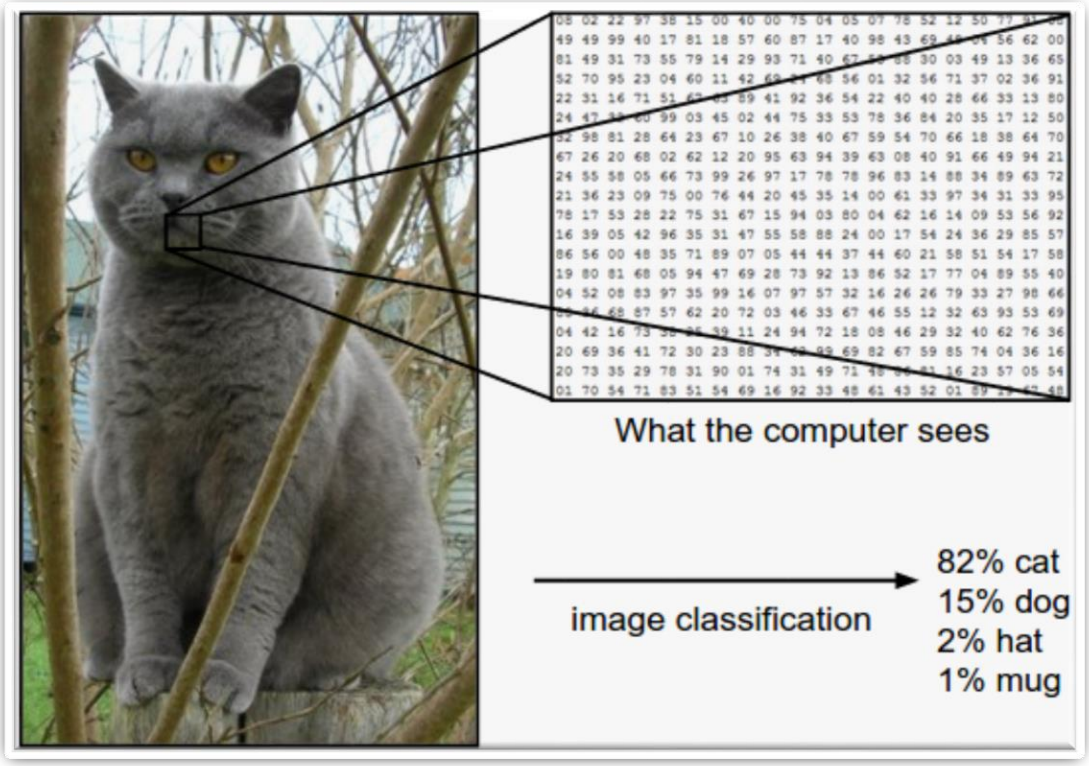



***Figura 10*** *Las imágenes son arreglos tridimensionales de enteros que van desde 0 hasta 255, de tamaño altura x ancho x 3. El 3 representa 3 colores de canal (RGB), las imágenes en escala de grises solamente tienen un canal [14].*

Como la tarea de reconocer un concepto visual (por ejemplo: “gato”) es relativamente trivial para un humano, vale la pena considerar los retos involucrados desde la perspectiva de los algoritmos de visión por computadora. La representación bruta de imágenes es un arreglo de valores de brillo:

- Variación del punto de vista. Los objetos se pueden orientar en muchas formas con respecto a la cámara [14].
- Variación de escala. Las clases visuales ofrecen una variación en su tamaño (tamaño en el mundo real, no sólo en términos de su extensión en la imagen) [14].
- Deformación. Muchos objetos de interés no son cuerpos rígidos y pueden ser deformados de muchas maneras [14].
- Condiciones de iluminación. Los efectos de iluminación son drásticos a nivel pixel [14].
- Fondo. Los objetos de interés podrían confundirse con su entorno, haciéndolos difícil de identificar [14].
- Variación dentro de la clase. Las clases de interés pueden ser muy bastas, como una silla. Existen muchos tipos diferentes de este objeto, cada uno con su propia apariencia [14].

Un buen modelo de clasificación debe ser invariante a cualquiera de estos cambios, mientras mantiene la sensibilidad hacia la variación dentro de la misma clase [14].

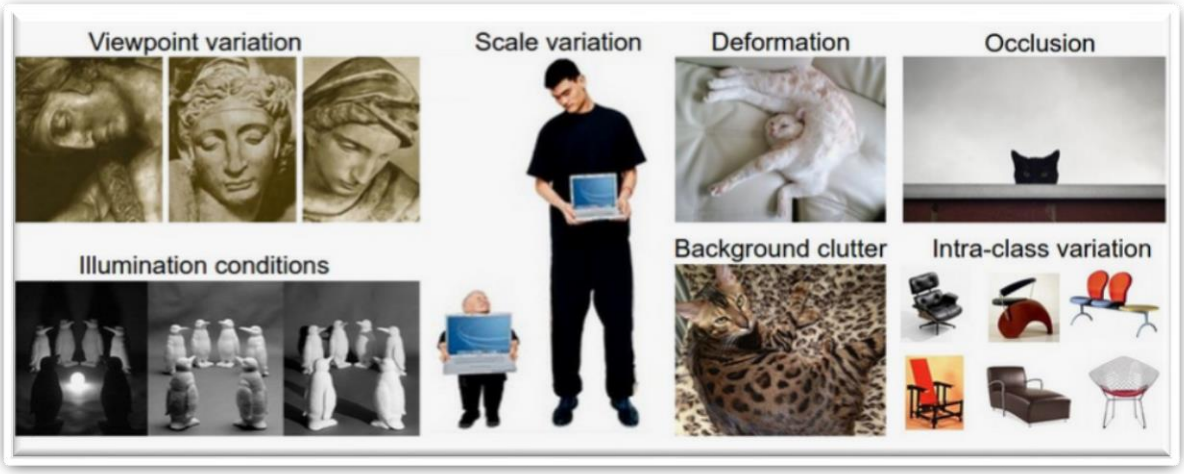



***Figura 11*** *Retos de la clasificación de imágenes [14].*

El flujo de clasificación de imagen es el siguiente:

- Entrada: Consiste en N imágenes, cada una etiquetada con K diferentes clases (conjunto de entrenamiento) [14].
- Aprendizaje: Uso del conjunto de entrenamiento para aprender cómo se ve cada conjunto de clases (entrenamiento de un clasificador o modelo de aprendizaje) [14].

- Evaluación: Probar la calidad del clasificador al pedirle que prediga la etiqueta de un nuevo conjunto de imágenes que no había visto antes. Comparar las etiquetas verdaderas de estas imágenes con las que predijo el clasificador. La meta es que muchas de las predicciones sean iguales con las respuestas verdaderas [14].

## III.II Redes Neuronales Convolucionales

En años recientes, modelos de aprendizaje profundo (Deep learning) han explotado múltiples capas de procesamiento de información no lineal, para extracción de características y transformación, así como análisis de patrones y clasificación, han demostrado superar todos estos retos. Entre ellas, las CNNs se han convertido en la arquitectura líder para la mayoría de las tareas de reconocimiento de imagen, clasificación y detección [15].

ImageNet Large Scale Visual Recognition Challenge (ILSVRC) es una evaluación comparativa en clasificación de categorías de objetos y detección de cientos de objetos y millones de imágenes. El reto se ha hecho anualmente desde 2010, atrayendo la participación de más de 50 instituciones. Consiste en dos componentes: 1. Conjunto de datos disponible al público, 2. Competencia anual. Los datos permiten el desarrollo y comparación de algoritmos de reconocimiento de categorías de objetos y la competencia brinda un camino para seguir el progreso y discutir las lecciones aprendidas de los más exitosos e innovadores de cada año. Este conjunto de datos contiene un conjunto de entrenamiento de imágenes etiquetadas a mano. Un conjunto de imágenes de prueba con las anotaciones manuales retenidas. Los participantes entrenan sus algoritmos usando el conjunto de entrenamiento y después automáticamente etiquetan las imágenes de prueba. Los etiquetados predichos se mandan al servidor de evaluación [6].

Las redes neuronales convolucionales profundas han dominado las tareas de clasificación. De hecho, han ganado en cada evento desde 2012 [15].

| Year | Team | Number of Layers | General Contribution | Position | References |
|---|---|---|---|---|---|
| 2010 | NEC | Shallow | Fast feature extraction, data compression, SVM classifier | First | Lin et al., 2011 |
| 2011 | XRCE | Shallow | High-dimensional image signatures, data compression, SVM classifier | First | Perronnin et al., 2010; Sánchez & Perronnin, 2011 |
| 2012 | SuperVision | 8 | Efficient GPU-based DCNN, with Dropout and several other innovations | First | Krizhevsky et al., 2012 |
| 2013 | Clarifai | 8 | DCNN architecture based on deconvolutional visualization technique | First | Zeiler & Fergus, 2014; Zeiler et al., 2011 |
| 2014 | GoogLeNet | 22 | DCNN architectural design based on Hebbian principle and multiscale ideas | First | Szegedy, Vanhoucke et al., 2015 |
| 2014 | VGG | 19 | Improvements to DCNN convolutional layers, increased network depth | Second | Simonyan & Zisserman, 2014 |
| 2015 | MSRA | 152 | Introduction of deep residual learning for ultra DCNNs | First | He et al., 2015b |

***Figura 12*** *ILSVRC Resultados de clasificación de imágenes desde 2010 [15].*

Las redes neuronales convolucionales ConvNet (Convolutional Network) o CNN (Convolutional Neural Networks) por sus siglas en inglés, son redes feedforward, en las que la información fluye en un solo sentido, de las entradas a las salidas. La arquitectura de las redes neuronales convolucionales tienen muchas variantes; pero en general, consisten en capas convolucionales y pooling (submuestreo), las cuales se agrupan en módulos. Seguidos de una o más capas totalmente conectadas (fully connected), como en redes neuronales feedforward comunes. Los módulos se apilan uno sobre otro para formar un modelo profundo (Deep) [15].

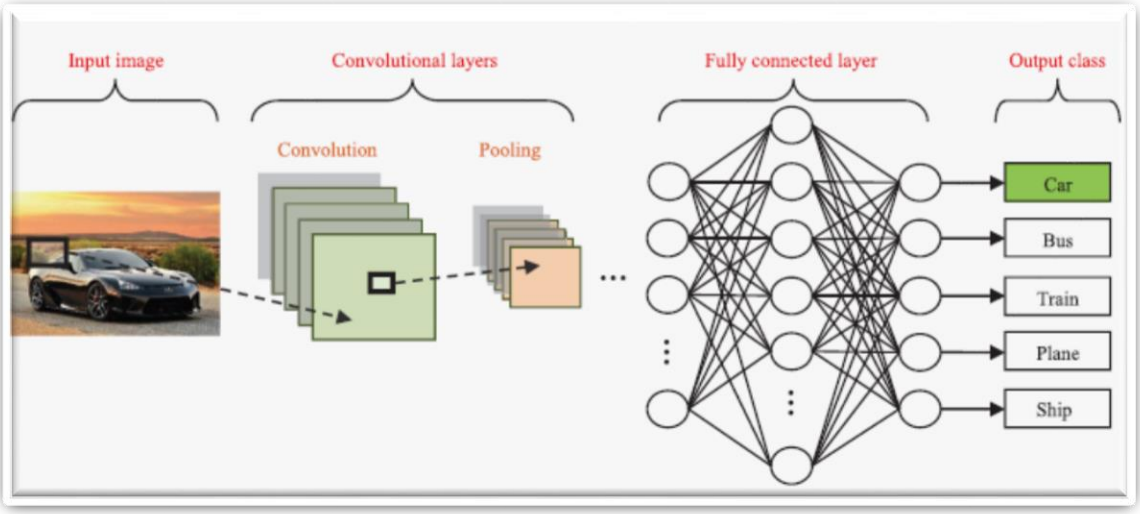



***Figura 13*** *Flujo de clasificación de imágenes CNN [15].*

### i. Capa Convolucional

Los parámetros de la capa convolucional consisten en un conjunto de filtros aprendibles. Cada filtro es pequeño espacialmente (en alto y ancho), pero se extienden a través de la profundidad total del volumen de entrada. Por ejemplo, un filtro típico en la primera capa de una red neuronal convolucional podría ser de tamaño 5x5x3 (5 pixeles de ancho y alto, 3 por 3 canales de color). Cada filtro se desliza (más preciso, se convoluciona) a través del alto y ancho de la entrada y se calcula el producto punto entre las entradas del filtro y la imagen de entrada en cualquier posición. Mientras desliza el filtro por el alto y ancho del volumen de entrada se produce un mapa de activación bidimensional que obtiene las respuestas del filtro en cada posición espacial. Intuitivamente, la red aprenderá los filtros que se activan cuando ven algún tipo de característica visual como el borde de alguna orientación o una mancha de algún color en la primera capa o eventualmente patrones completos como un panal de abeja o algo parecido a una llanta en capas más altas de la red. Si tenemos un conjunto de filtros en cada capa convolucional (Conv layer) por ejemplo, 12 filtros, cada uno de estos producirá un mapa de activación bidimensional. Estos mapas de activación se apilan a lo largo de la dimensión de profundidad y producirá un volumen de salida [14].

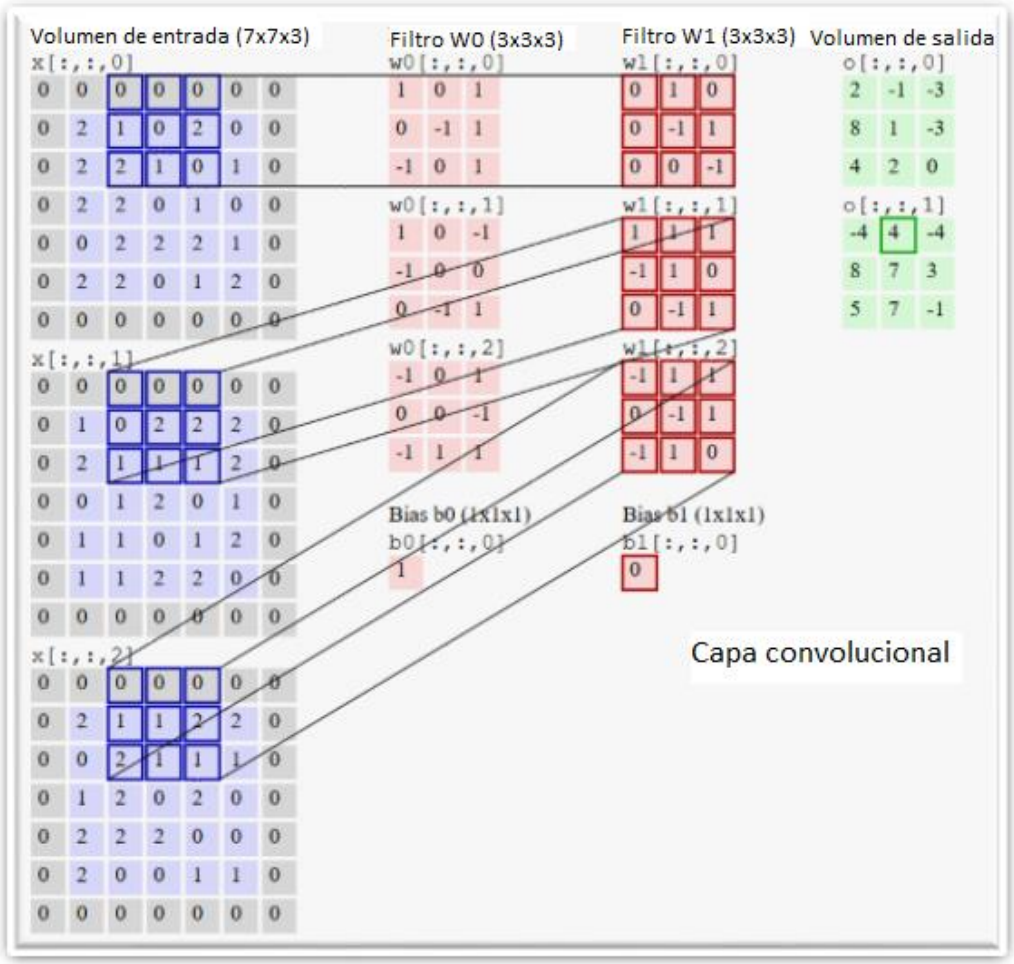



***Figura 14*** *La imagen muestra el volumen de salida (verde) y se observa que cada elemento se calcula multiplicando cada elemento de los números resaltados en el volumen de entrada (azul) por el filtro (rojo), sumándolos y agregando al resultado el sesgo (bias). [14].*

Tres hiperparámetros controlan el tamaño del volumen de salida: profundidad, stride (pasos) y zero-padding (margen cero):

1. La profundidad del volumen de salida corresponde al número de filtros que queremos ocupar, cada uno aprendiendo a buscar algo diferente en la entrada [14].

2. El stride, que son los pasos que damos al deslizar el filtro. Cuando el stride es 1 movemos los filtros un pixel a la vez. Cuando el stride es 2 los filtros brincan 2 pixeles a la ves cuando los deslizamos [14].

3. Algunas veces es conveniente cambiar el margen del volumen de entrada por ceros. La característica del zero-padding es que nos permite controlar el tamaño espacial de nuestro volumen de salida [14].

Podemos calcular el tamaño de nuestro volumen de salida como una función del volumen de entrada (W), el tamaño del campo receptivo de la para convolucional (F), el stride con el que se aplica (S) y la cantidad de zero-padding (P) usada en el borde. La fórmula para calcular el volumen de salida está dada por (W-F+2P) /S+1. Por ejemplo, para una entrada 7x7 y un filtro de 3x3 con un stride de 1 y pad 0, tenemos una salida de 5x5 (7-3/1+1=5) [14].

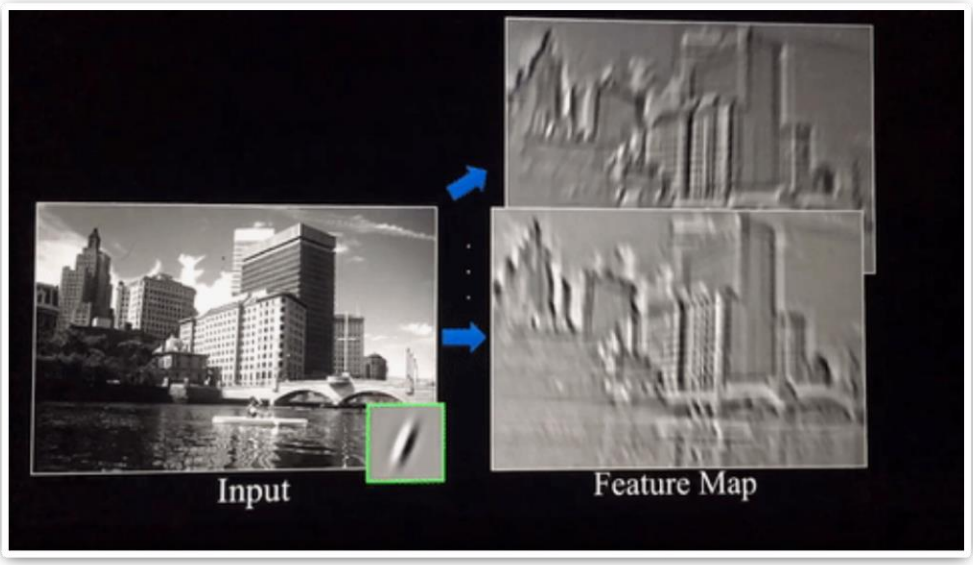



***Figura 15*** *Visualización del resultado del mapa de activación creado, después de pasar dos filtros a la imagen de entrada [16].*

### *Filtros*

En la siguiente tabla se pueden observar los efectos de la convolución de una imagen con diferentes filtros. Con los filtros se pueden detectar bordes, manchas o agudizar los colores de la imagen [16].

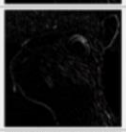
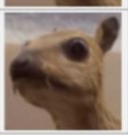
| Operation | Filter | Convolved Image |
|---|---|---|
| Identity | $\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$ |  |
| Edge detection | $\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$ |  |
| | $\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$ |  |
| | $\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$ |  |
| Sharpen | $\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$ |  |
| Box blur (normalized) | $\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$ |  |
| Gaussian blur (approximation) | $\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$ |  |

***Figura 16*** *Resultado de diferentes filtros aplicados a la misma imagen [14].*

Al ser aprendibles los parámetros de los filtros, cada red neuronal tendrá la capacidad de decidir qué tipo de características necesita obtener de las imágenes para poder tener el menor error posible y reconocer el patrón necesario para poder predecir de forma exitosa el resultado de imágenes no vistas antes.

### *Zero-padding*

La razón de utilizar el zero-padding es que, sin esta operación, cada vez que la imagen pase por una capa convolucional, la imagen disminuye su tamaño y existiría un límite de capas convolucionales utilizables en la arquitectura de la red neuronal. Otra desventaja que tiene el no utilizar el zero-padding es que las esquinas de la imagen, sólo se utilizan una vez en la capa convolucional, cada filtro sólo para una vez por esos pixeles, mientras que, si observamos un pixel al centro de la imagen, más de una vez, es utilizado para hacer la convolución, por lo tanto la información de las esquinas se vuelve menos relevante, por eso el zero-padding es una característica que mejora el rendimiento de la convolución.

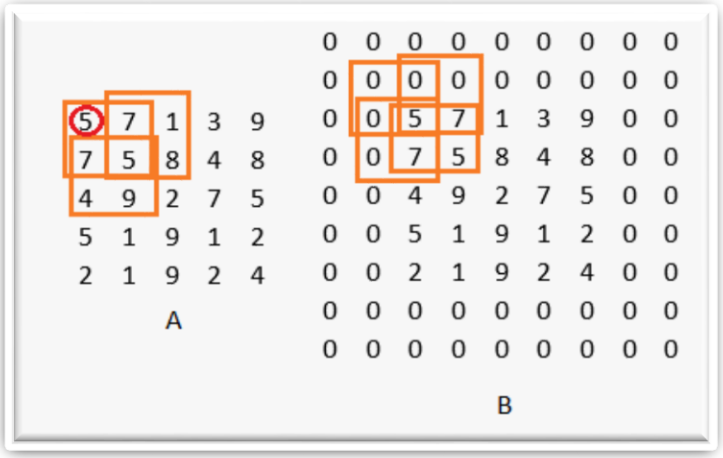



***Figura 17*** *En la imagen (A) podemos observar que al deslizar el filtro (matriz de 2 x 2), el primer pixel de la imagen, sólo se tomaría una vez sin zero-padding, mientras que en la imagen (B), con zero-padding, se le da más importancia al primer pixel de la imagen al ser usado la misma cantidad de veces (4 veces) que cualquier otro pixel en la convolución.*

### *Función de activación*

La función de activación en la capa convolucional, tiene el mismo propósito que la activación usada para cualquier red neuronal, normalmente se usa una función no lineal, la cual se aplica a todos los elementos de la matriz y sirve para formar funciones complejas que puedan adaptarse al problema que se busca predecir.

Las funciones de activación más comunes son:

Sigmoid. Tiene la forma matemática de σ(x)=1/(1+e−x). Tomar el valor de entrada y lo comprime a un valor entre 0 y 1. La función sigmoid se usa principalmente en cuando la red busca diferenciar entre dos clases [14].

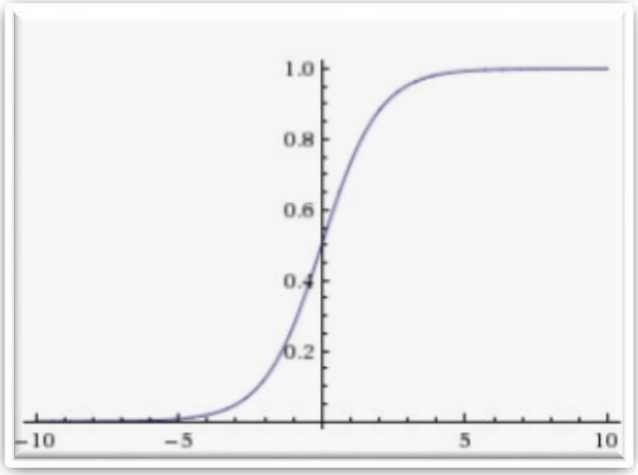



***Figura 18*** *La salida de la función sigmoid siempre son números entre [0,1] [14].*

Una propiedad indeseable de esta función es que cuando el valor de salida de la función se encuentra muy cerca de 0 o de 1, el gradiente en estas regiones es prácticamente cero y eso para la parte de backpropagation, que se explicará más adelante, no es algo deseado. Además, hay que tener cuidado en la forma de inicializar los pesos (parámetros de los filtros) para prevenir la saturación, si los pesos se inician en un valor demasiado grande entonces la mayoría de las neuronas se saturarán y casi no aprenderán. Otra condición indeseable de esta función es que no está centralizada en cero. Como los datos a la entrada de la neurona siempre serían positivos, el gradiente de los pesos (W) será siempre todo positivo o todo negativo. Esto introduce una dinámica indeseable y lenta al entrenar la red [14].

Tanh. La función no lineal comprime un valor de entrada a un rango entre [-1,1]. Como la función sigmoid, su activación se satura (el gradiente de la función cerca de -1 y 1 es prácticamente cero), pero a diferencia de la función sigmoid, la función se encuentra centrada en cero. La función tanh es una versión escalada de la sigmoid: $\tanh(x)=2\sigma(2x) -1$ [14].

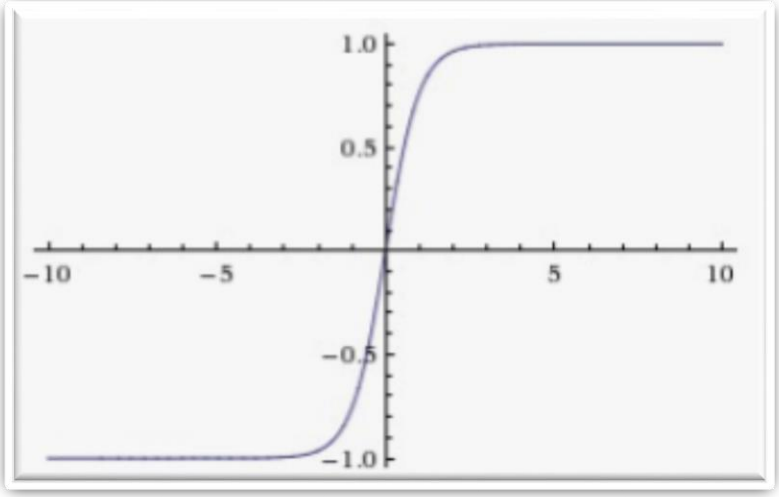



***Figura 19*** *Función no lineal tanh, comprime números a un rango entre [-1,1] [14].*

ReLu. Rectified Linear Unit por sus siglas en inglés, se ha vuelto muy popular en los últimos años. La función es: $f(x)=\max(0, x)$, simplemente la función tiene un límite en cero. Algunas ventajas de la función son: acelera la convergencia comparada con las funciones sigmoid o tanh debido a que nunca se satura la función, no involucra operaciones como funciones exponenciales. Una desventaja de la función ReLu es que podría pasar que los pesos se actualicen de cierta forma en la que la neurona nunca se active, es decir, que la salida de esa neurona siempre sea cero y cuando se llega a esa condición durante el entrenamiento no se puede salir de ella [14].

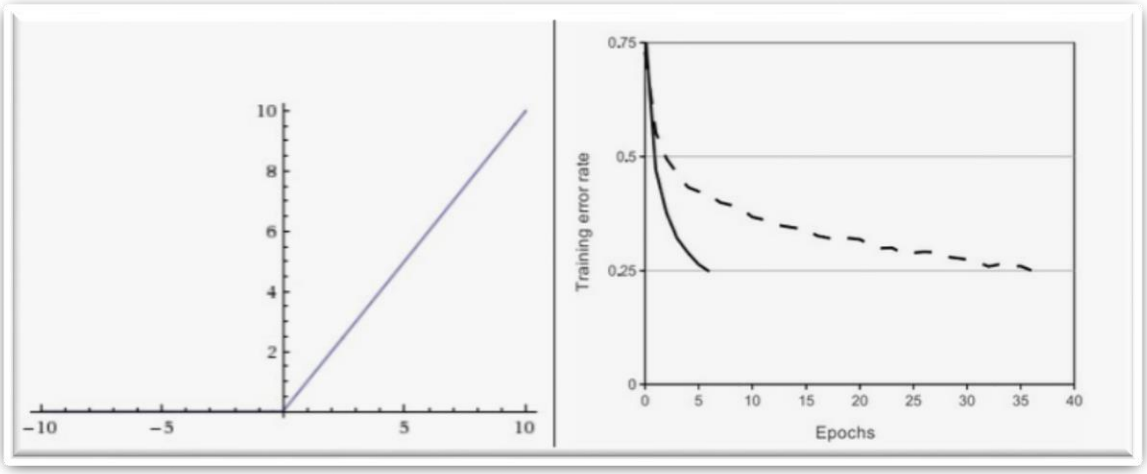



***Figura 20*** *Función de activación ReLu, cero cuando x < 0 y la parte lineal con pendiente 1 cuando x > 0 (izquierda). Gráfica de la mejora (6x) de la convergencia con ReLu, en comparación con tanh (derecha) [14].*

Leaky ReLu. Esta función es un intento de arreglar el problema de no activación de la función ReLu. En lugar de que la función sea cero cuando la entrada es menor a cero, la función tendrá una pequeña pendiente negativa (0.01 por ejemplo). La función es: f(x)=1(x<0) (αx) +1(x>=0) (x) donde α es una constante. Los resultados de esta función no son consistentes [14].

### *Inicialización de parámetros*

La inicialización de parámetros es clave para el aprendizaje de las redes neuronales, a continuación, se mencionan las formas más comunes de inicializar los pesos (filtros de la capa convolucional):

Inicialización cero. Al no saber con qué valor van a terminar los pesos, con la normalización de datos (entrada de la neurona centrada en cero), podemos asumir que la mitad de pesos serán positivos y la mitad negativos por lo que inicializar los parámetros en cero podría parecer lógico. Pero es un error inicializar así, puesto que, si cada neurona calcula la misma salida, todas las neuronas calcularán el mismo gradiente durante backpropagation y harán la misma actualización de parámetros. Por lo tanto, no existe asimetría entre las neuronas [14].

Números pequeños aleatorios. Queremos que los números se inicialicen cerca de cero, pero no igual a cero. Es común inicializar los pesos a números pequeños, la idea es que todos los pesos tengan números aleatorios y únicos al principio, para que calculen diferentes actualizaciones y se integren como partes diferentes de la red. No necesariamente en el caso de que los números sean muy pequeños, la red trabajará mejor, debido a una gran disminución en los gradientes [14].

Calibrando las varianzas con 1/sqrt(n). Un problema de la inicialización con números aleatorios pequeños es que la distribución de las salidas de una neurona inicializada aleatoriamente tiene una varianza que crece

con el número de entradas. Podemos normalizar la varianza de cada salida de neurona a 1 escalando el vector de su peso por la raíz cuadrada del número de entradas [14].

Inicialización normal Xavier. La inicialización propone Var(w)=2/($n_{in}$+$n_{out}$) donde $n_{in}$, $n_{out}$ son el número de entradas en la capa anterior y en la capa siguiente. Evita que la inicialización de parámetros sea muy grande o muy pequeña para todos los parámetros de la red [17].

He normal initializer. El autor propone que la inicialización específicamente para neuronas con función de activación ReLu, debe tener varianza de 2.0/n, en su publicación se basa en la inicialización normal Xavier [18].

### ii. Capa Pooling

Es común agregar periódicamente una capa Pooling entre capas convolucionales sucesivas. Su función es reducir progresivamente el tamaño espacial de la representación para reducir la cantidad de parámetros y cálculos en la red y por lo tanto controlar el overfitting (sobre aprendizaje del conjunto de entrenamiento, con baja precisión al predecir resultados de nuevos datos) y así lograr la invarianza espacial para distorsiones de entrada y traslación (lograr reconocer ciertas características de la imagen, sin importar en qué lugar de la imagen se encuentren). La capa Pooling opera de manera independiente en cada arreglo bidimensional a lo largo de la profundidad total de la entrada y cambia su tamaño espacialmente, usando normalmente la operación MAX. La forma más común es una capa Pooling con filtros de tamaño 2x2 aplicado con un stride de 2, que reduce cada arreglo bidimensional de la profundidad total de la entrada por 2, descartando 75% de las activaciones. Cada operación MAX en este caso tomaría el máximo de 4 números (región de 2x2). La profundidad permanece sin cambio alguno. Resumiendo, la capa pooling [14]:

1. Acepta un volumen de tamaño $W_1 \times H_1 \times D_1$

2. Requiere de dos hiperparámetros:
   - La extensión espacial F,
   - El stride S,

3. Produce un volumen de tamaño $W_2$ X $H_2$ X $D_2$ donde:
   - $W_2 = (W_1 - F) / S + 1$
   - $H_2 = (H_1 - F) / S + 1$
   - $D_2 = D_1$

4. No introduce parámetros puesto que calcula una función fija de la entrada

5. No es común usar zero-padding para capas pooling

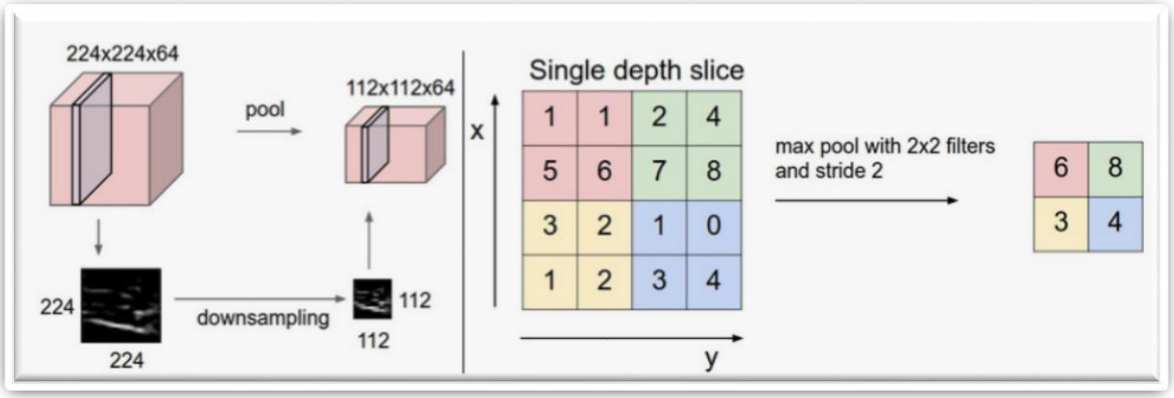



***Figura 21*** *La capa pooling reduce el volumen espacialmente, independientemente de la profundidad del volumen de entrada. Izquierda: en este ejemplo, al volumen de entrada [224x224x64] se le aplica el pooling con un filtro de tamaño 2, stride 2 y resulta en un volumen de salida de [112x112x64]. La profundidad del volumen se conserva. Derecha: La operación más común de reducción es MAX, aquí se muestra con un stride de 2 y un filtro de 2x2. Se toma el número más grande entre 4 posibles (por el filtro de 2x2) [14].*

### iii. Capa totalmente conectada

Muchas capas convolucionales y pooling se apilan una sobre otra para extraer más representaciones de características abstractas que se mueven a través de la red. Las capas totalmente conectadas que siguen estas capas interpretan esta representación de características y realizan la función de razonamiento de alto nivel [15].

La desventaja de las capas totalmente conectadas es la cantidad de parámetros que requieren, a diferencia de la capa convolucional, en donde cada parámetro se concentra a extraer lo más importante de cierta zona de la imagen, en la capa totalmente conectada, cada parámetro se usará para toda la imagen, por eso se llama totalmente conectada.

### iv. Entrenamiento

Las redes neuronales usan algoritmos para ajustar sus parámetros (pesos) para obtener la salida deseada de la red. El algoritmo más común es backpropagation, el cual calcula el gradiente de una junción objetivo (función de costo) para determinar cómo ajustar los parámetros para minimizar el error que afecta el desempeño. Un problema común durante el entrenamiento con CNNs, es overfitting, que es un bajo desempeño en el conjunto de prueba después de que la red fue entrenada en un conjunto de dato de entrenamiento. Esto afecta la capacidad del modelo de generalizar en datos no vistos y es un reto para las redes neuronales convolucionales [15].

### v. Función de costo

Al ser un problema de aprendizaje supervisado, la red neuronal debe medir la diferencia entre la predicción y la etiqueta verdadera. La función de costo es el promedio de los errores de cada ejemplo individual. L=1/N

($\sum_i L_i$) donde N es el número de datos de entrenamiento de la red neuronal y $L_i$ es la función de pérdida (error en el dato i) [14].

Existen diferentes funciones de pérdida, entre las más comunes para problemas de clasificación binaria son:

Cross-entropy. Clasificador binario con clases (0, 1), calcula la probabilidad de la clase 1 como:

$$L_i = \sum_j y_{ij} \log(a) + \big(1 + y_{ij}\big)\log(1 - a)$$

Donde j son todas las entradas, y es la salida deseada y a es la salida de la red neuronal.

Error medio cuadrático. La razón por la que es cuadrada es que el gradiente se vuelve más simple, sin cambias los parámetros.

$$L_i = ||f - y_i||^2$$

Donde f es la salida deseada, y es la salida que resulto de la red neuronal.

Este tipo de función es difícil de optimizar, es menos robusta porque con valores atípicos puede introducir gradientes muy grandes al modelo.

### vi. Backpropagation

El objetivo de backpropagation es calcular las derivadas parciales de la función de costo, con respecto a los parámetros W(peso) y b(sesgo) iniciales. Al cambiar los pesos y sesgos en la red, cambiamos el costo de la función. Pero para calcular las derivadas de la función de costo con respecto a los parámetros W y b, debemos hacer la derivada de la función de costo con respecto de la función de activación utilizada en la red e ir recorriendo capa por capa cada derivada hasta llegar a nuestra primer W (parámetro de la primera capa utilizada en la red), que dependerá de todos los demás parámetros, por lo tanto, todos los parámetros varían (aprenden) durante el entrenamiento. Después de obtener las derivadas de los parámetros que se están utilizando, se usan en algún método de optimización que busque un mínimo global de la función costo, busca los parámetros W y b que obtengan el mínimo resultado de la función de costo.

### vii. Optimizador

Gradient descent es uno de los algoritmos más populares para realizar la optimización y el más común para optimizar redes neuronales. Gradient descent es una forma de optimizar una función objetivo J(θ) parametrizada por θ ∈ R actualizando parámetros en la dirección opuesta del gradiente de la función objetivo ∇θJ(θ) con respecto a los parámetros. La tasa de aprendizaje η determina el tamaño de los pasos que se toman para llegar a un mínimo. Se sigue la dirección de la pendiente de la superficie creada por la función objetivo en dirección hacia el mínimo hasta encontrar un valle [19].

$$\theta = \theta - \eta \nabla_\theta J(\theta)$$

Pero existen diferentes algoritmos que son variantes de gradient descent, a continuación, se muestran algunos:

Gradient descent con Momento. Gradient descent (GD) tiene problemas al navegar por curvas de superficies que son mucho más pronunciadas hacia una dimensión en comparación con la otra. En estos escenarios, Gradient descent oscila a través de las pendientes mientras progresa muy poco a en dirección hacia el local óptimo [19].

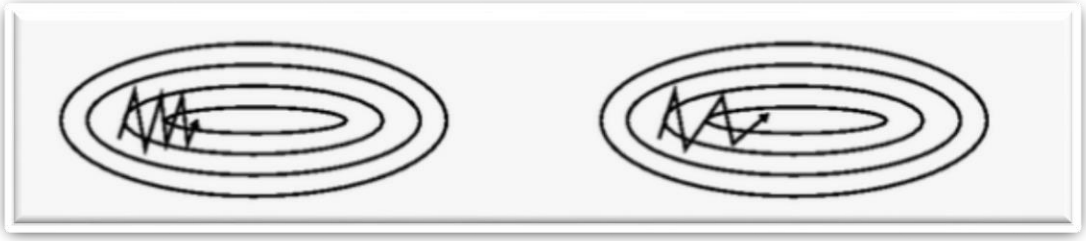


***Figura 22*** *(A)GD sin momento* *(B) GD con momento [19].*

Momento es un método que ayuda a GD en la dirección relevante y amortigua las oscilaciones en la otra dirección. Esto lo logra agregando una fracción γ al vector actualizado del paso de tiempo anterior [19].

$$v_t = \gamma v_{t-1} + \eta \nabla_\theta J(\theta)$$

$$\theta = \theta - v_t$$

El término de momento γ normalmente es 0.9 o algo similar [19].

Cuando se usa momento, es como si empujáramos una pelota colina abajo. La pelota acumula momento mientras gira colina abajo, incrementando cada vez más su velocidad hasta que llega a su velocidad final. Lo mismo sucede con la actualización de parámetros: el término de momento incrementa para las dimensiones cuyos gradientes se encuentran en la misma dirección y se reducen las actualizaciones para dimensiones cuyos gradientes cambian de dirección. Como resultado, obtenemos una convergencia más rápida y una reducción en las oscilaciones [19].

RMSprop. Busca que GD tenga momento en la dirección del mínimo y busca desacelerar la componente que se encuentre en otra dirección para amortiguar las oscilaciones, pero en este caso los parámetros en la dirección correcta también serán divididos entre un número muy pequeño para aumentar aún más su

velocidad y los parámetros en la dirección equivocada serán divididos por un número grande para penalizar más su velocidad y disminuir las oscilaciones [19].

$$E[g^2]_t = \gamma E[g^2]_{t-1} + (1-\gamma) g^2{}_t$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} g_t$$

RMSprop sugiere que $\gamma$ sea 0.9 y η de 0.001 [19].

Adam. Por sus siglas en inglés (Adaptive Moment Estimation) es otro método que calcula una tasa de aprendizaje adaptiva para cada parámetro. Es una combinación de RMSprop con SG con momento [19]:

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2$$

Como los parámetros $m_t$ y $v_t$se inicializan como vectores de 0's, su ordenada al origen se acerca a cero, especialmente durante los primeros pasos, incluso cuando los datos sean alejados a cero, por lo tanto, Adam también utiliza una corrección de la ordena al origen [19]:

$$m'_t = \frac{m_t}{1-\beta_1^t}$$

$$v'_t = \frac{v_t}{1-\beta_2^t}$$

Y después actualiza los parámetros así:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{v'_t} + \epsilon} m'_t$$

Los autores proponen valores de 0.9 para $\beta_1$, 0.999 para $\beta_2$ y $10^{-8}$ para ϵ. En la práctica Adam funciona mejor que muchos otros algoritmos [19].

Nesterov accelerated gradient (NAG). Volviendo a la analogía del momento, si una pelota rueda colina abajo, siguiendo la pendiente a ciegas, no es satisfactorio. Una pelota más inteligente, que tiene noción de hacia dónde va para saber disminuir la velocidad antes de que la pendiente de la colina cambie. NAG es una forma de dar esta presencia al momento, usando el término de momento γ $v_{t-1}$ para mover los parámetros

θ. Y calculando el gradiente no con respecto a los parámetros actuales θ, sino con respecto a la posición aproximada futura de nuestros parámetros [19]:

$$v_t = \gamma v_{t-1} + \eta \nabla_\theta J(\theta - \gamma v_{t-1})$$

$$\theta = \theta - v_t$$

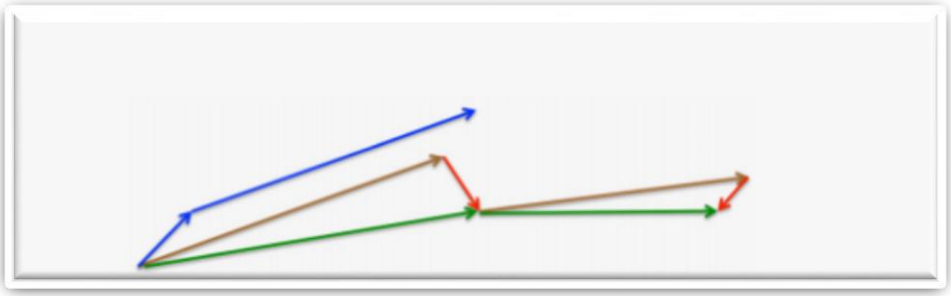


***Figura 23*** *Actualización Nesterov [19].*

El término de momento $\gamma$ se ajusta a un valor alrededor de 0.9. Mientras que SG con momento calcula el primer gradiente (vector azul pequeño) y toma un gran salto en la dirección del gradiente acumulado actualizado (vector azul grande), NAG primero toma un gran salto en la dirección del gradiente acumulado previo (vector café), mide el gradiente y hace una corrección (vector verde). Esta actualización anticipada mejora el desempeño en diferentes tareas [19].

Nadam. Como se mencionó anteriormente Adam se puede ver como una combinación de RMSprop y SG con momento. Nadam (Nesterov-accelerated Adaptive Moment Estimation) combina Adam con NAG [19].

Recordemos que Adam actualiza de la siguiente forma:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$

$$m'_t = \frac{m_t}{1 - \beta_1^t}$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{v'_t} + \epsilon} m'_t$$

Expandiendo la tercera ecuación se obtiene:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{v'_t} + \epsilon}\left(\frac{\beta_1 m_{t-1}}{1-\beta_1^t} + \frac{(1-\beta_1)g_t}{1-\beta_1^t}\right)$$

Reemplazando el término $\frac{\beta_1 m_{t-1}}{1-\beta_1^t}$ por $\beta_1 m'_{t-1}$:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{v'_t} + \epsilon}\left(\beta_1 m'_{t-1} + \frac{(1-\beta_1)g_t}{1-\beta_1^t}\right)$$

Se agrega NAG simplemente reemplazando la corrección de la ordenada estimada del vector momento del paso anterior $m'_{t-1}$con la corrección de la ordenada estimada del vector actual $m'_t$, lo que da la actualización Nadam como [19]:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{v'_t} + \epsilon}\left(\beta_1 m'_t + \frac{(1-\beta_1)g_t}{1-\beta_1^t}\right)$$

### viii. Prueba

Después de entrenar la red neuronal en el conjunto de imágenes de entrenamiento, se prueba la red neuronal en el conjunto de imágenes de prueba para comprobar si es capaz de generalizar y predecir correctamente la salida a un conjunto de imágenes no vistas previamente. La prueba también sirve como diagnóstico para mejorar la red neuronal.

### ix. Diagnóstico

Modelos de red neuronal con bias, son modelos que generalizan bien con nuevas imágenes, es decir, el error que hay en el conjunto de entrenamiento es muy parecido al error que existe en el conjunto de prueba, el problema es que, en ambos casos, la función generada por la red neuronal no logra encajar los datos de manera correcta, por lo tanto, ambos conjuntos de imágenes tienen un error grande, a pesar de que los dos tengan un error similar.

Modelos con variance, son modelos que en el conjunto de entrenamiento tienen un error cercano a cero, pero por la complejidad de la función generada para abarcar al conjunto de datos de entrenamiento en su totalidad, tienen un error grande en el conjunto de prueba, puesto que se le dificulta mucho al modelo generalizar de manera correcta imágenes no vistas.

Lo ideal es generar una función que tengan un error pequeño en el conjunto de entrenamiento (menor al %5) y que ese error sea sólo un poco más grande que el de entrenamiento (~+2%).

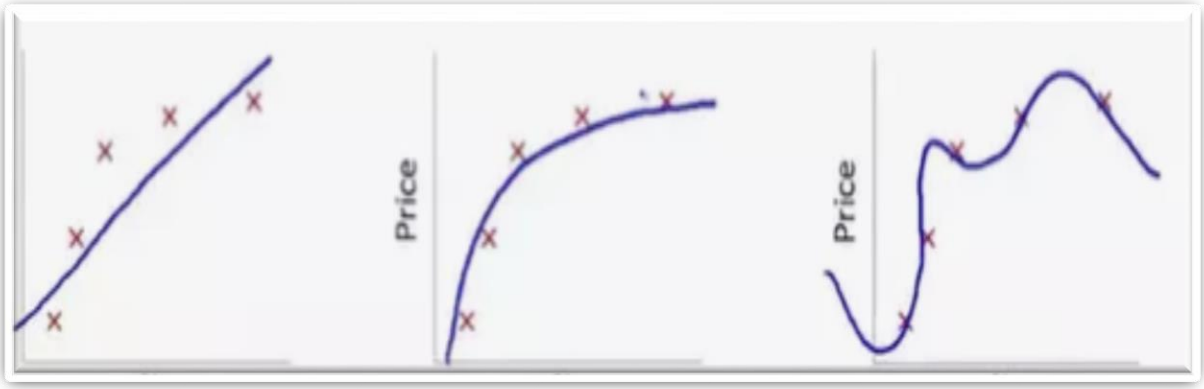



***Figura 24*** *Imagen que muestra ejemplos de: bias (izquierda). Mala predicción de la red neuronal, variance (derecha). Buena predicción en el conjunto de entrenamiento y un modelo ideal (centro) que logra predecir bien tanto en el conjunto de entrenamiento como en el conjunto de prueba.*

Cuando el modelo seleccionado para la red neuronal sufre de bias (underfitting), las soluciones son sencillas, modificar la arquitectura de la red, agregando más capas o entrenando el modelo por más tiempo.

En cambio, si se tiene variance (overfitting), la mejor solución para el modelo es obtener más datos, para poder entrenar la red con muchas imágenes diferentes y que se le facilite la generalización, pero existen casos en los que obtener más imágenes es muy complicado, para esas situaciones se puede utilizar dropout o alguna otra forma de regularización.

### *Dropout*

El término dropout se refiere a sacar capas de la arquitectura de la red, removerlas del modelo, incluyendo sus entradas y salidas. Estas capas se remueven aleatoriamente, con una probabilidad fija p independiente de otras capas, normalmente se le da el valor de 0.5. Al aplicar dropout obtenemos una red menor tamaño que la original, la cual consiste en las unidades sobreviviente del dropout. Para cada entrenamiento, una nueva red de menor tamaño a la original es entrenada [20].

Durante la prueba no es factible usar dropout, se debe usar la red neuronal completa, sin dropout. Los pesos de la red neuronal original se encuentran escalados con respecto a los pesos entrenados. Si una capa se retiene con probabilidad p durante el entrenamiento, los pesos de salida se multiplican por p a la hora de prueba. Esto asegura que, para cualquier capa oculta, la salida esperada (bajo la distribución utilizada usada para remover capas durante el entrenamiento) es la misma que la salida real durante la prueba. Al entrenar una red con dropout, durante la prueba conduce a significativamente menor error de generalización en una amplia variedad de problemas de clasificación [20].

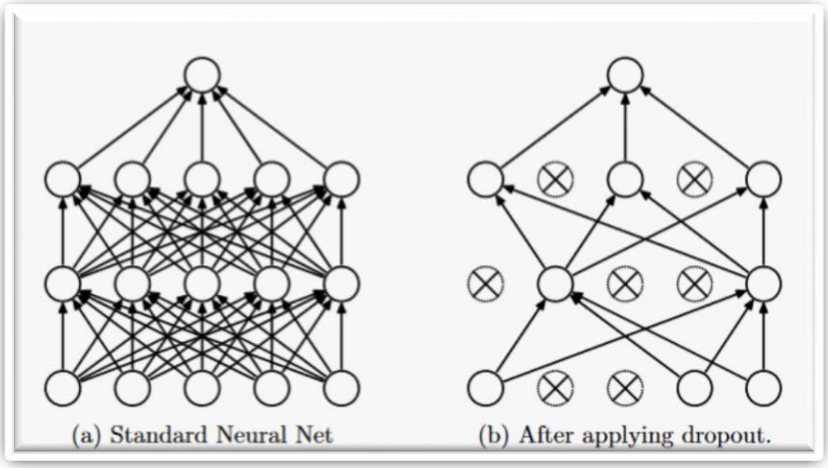



***Figura 25*** *Dropout a un modelo de redes neuronales común, en el que el dropout se aplica a unidades y no a capas, como en las redes neuronales convolucionales [20].*

En redes neuronales comunes, la derivada recibida por cada parámetro le dice cómo debe cambiar para que la función de costo se reduzca, dado lo que otras capas están haciendo. Por lo tanto, las capas cambian de forma que arreglen los errores de las otras capas. Esto lleva a una coadaptación compleja de capas, lo que, resulta en overfitting porque estas coadaptaciones, no generalizan correctamente con datos no vistos. Dropout previene la coadaptación al hacer la presencia de otras capas no confiables. Por lo tanto, ninguna capa puede depender de otra para corregir sus errores. Debe de desempeñarse bien en una amplia variedad de contextos [20].

# IV. Segmentación

## IV.I Base de datos

Se realizó una base de datos recabando imágenes RGB de diferentes fuentes: imágenes de estudios post cirugía láser fetoscópica, para el tratamiento de pacientes con embarazos monocigóticos en los cuales se utilizó inyección de colorante en las venas y arterias de la placenta resultando en colores contrastantes; las venas normalmente son de colores claros y las arterias de colores oscuros. De acuerdo con [21] el inyectar las placentas con colorante ayuda a extraer mejor las características de la misma sin que el colorante afecte las propiedades de venas y arterias o altere el punto final de los vasos.

Para estandarizar las imágenes y que la entrada de la red neuronal tenga el mismo tamaño, se redimensionan las imágenes a un tamaño de 365 x 488.

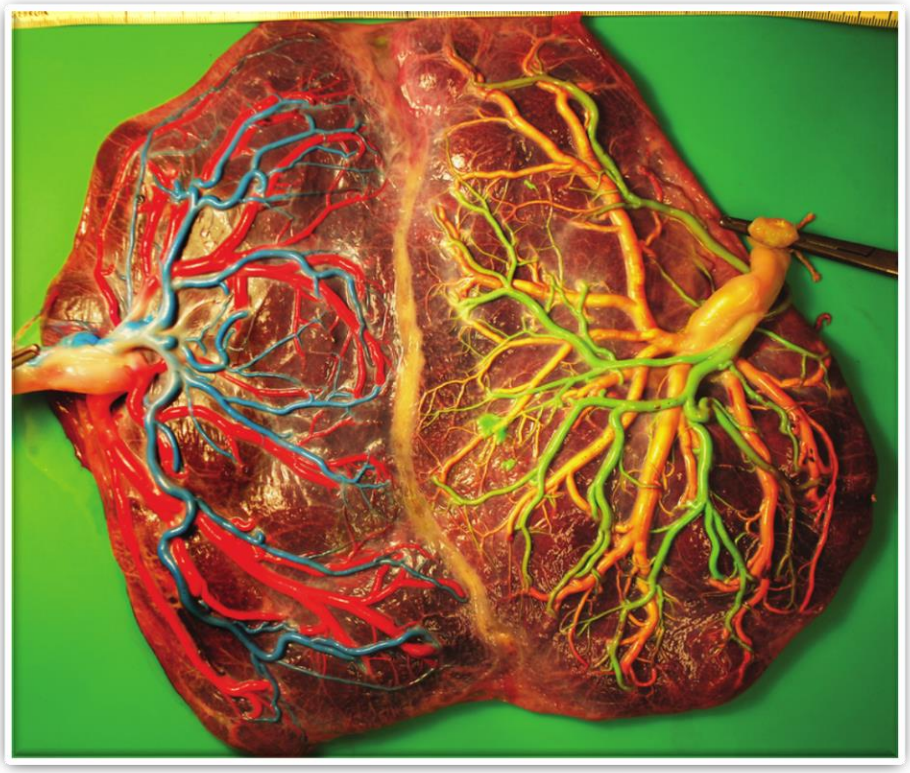


***Figura 26*** *Coloración de placenta tratada de síndrome de transfusión gemelo-gemelo,*

## IV.II Etiquetado

El proceso de etiquetado consiste en señalarle a la red neuronal, las diferentes clases que hay en la imagen, para este trabajo sólo existen dos clases, venas y arterias es una clase y la segunda clase es todo lo demás en la imagen, pero para que pueda diferenciarlas, es necesario pintar una clase de un color y lo demás de otro color (etiquetar cada clase).

Las imágenes recabadas se etiquetaron manualmente, utilizando el programa InteractLabeler 1.2.1, el cual es un programa que desempeña una anotación por región en fotografías. InteractLabeler hace una segmentación regional automática, que divide la imagen en secciones para el etiquetado [22].

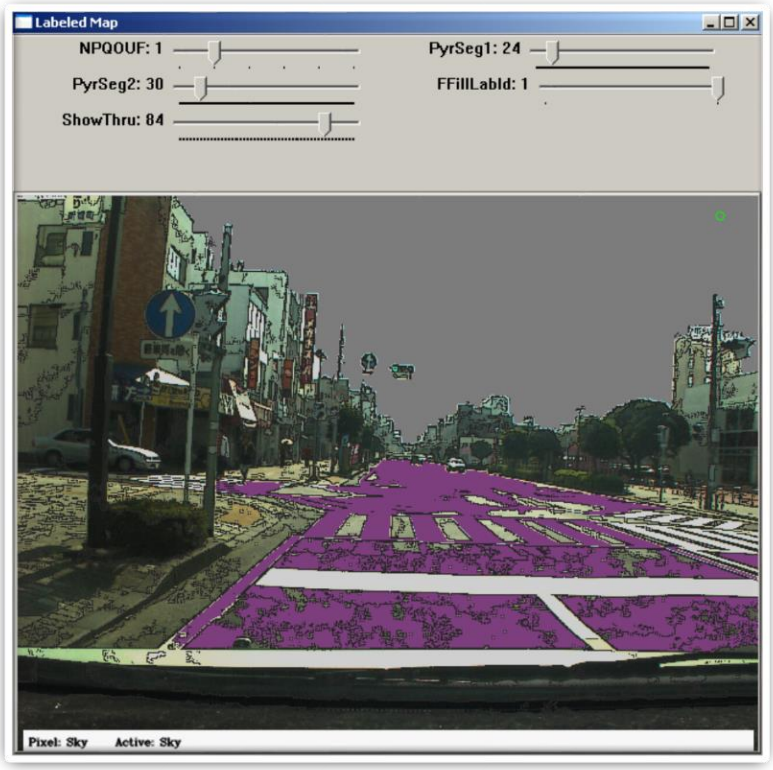



***Figura 27*** *Segmentación automática de InteractLabeler, con el pavimento pintado manualmente de morado [22].*

Como se puede observar en la imagen la segmentación automática, aunque no es perfecta, facilita mucho el etiquetado de la imagen, la división automática secciona la fotografía en regiones, dependiendo de la semejanza en los colores de la imagen original. De esta forma con un solo click, se pinta una región grande de la imagen, en este caso del pavimento, después con más detalle es necesario colorear las regiones más pequeñas que se segmentaron.

El programa es utilizado principalmente para etiquetar imágenes de videos tomados durante la conducción de un auto (base de datos CamVid), separando cada objeto en 32 categorías (32 colores diferentes).

Con la ayuda de InteractLabeler se etiquetaron todas las imágenes recabadas para la base de datos, una característica que tiene el programa y fue de gran ayuda fue la de zoom, para hacer un etiquetado más detallado en zonas de mucho ruido o con elementos muy pequeños, el cambio del grosor de la herramienta que pinta también fue de gran ayuda en sección de este tipo. Mientras se hace el etiquetado el programa te permite ver una ventana del zoom, una ventana de la imagen original y una ventana de la imagen que se forma con el etiquetado que se ha formado hasta el momento, lo cual es de gran ayuda para detectar cuando se etiquetó alguna sección incorrectamente.

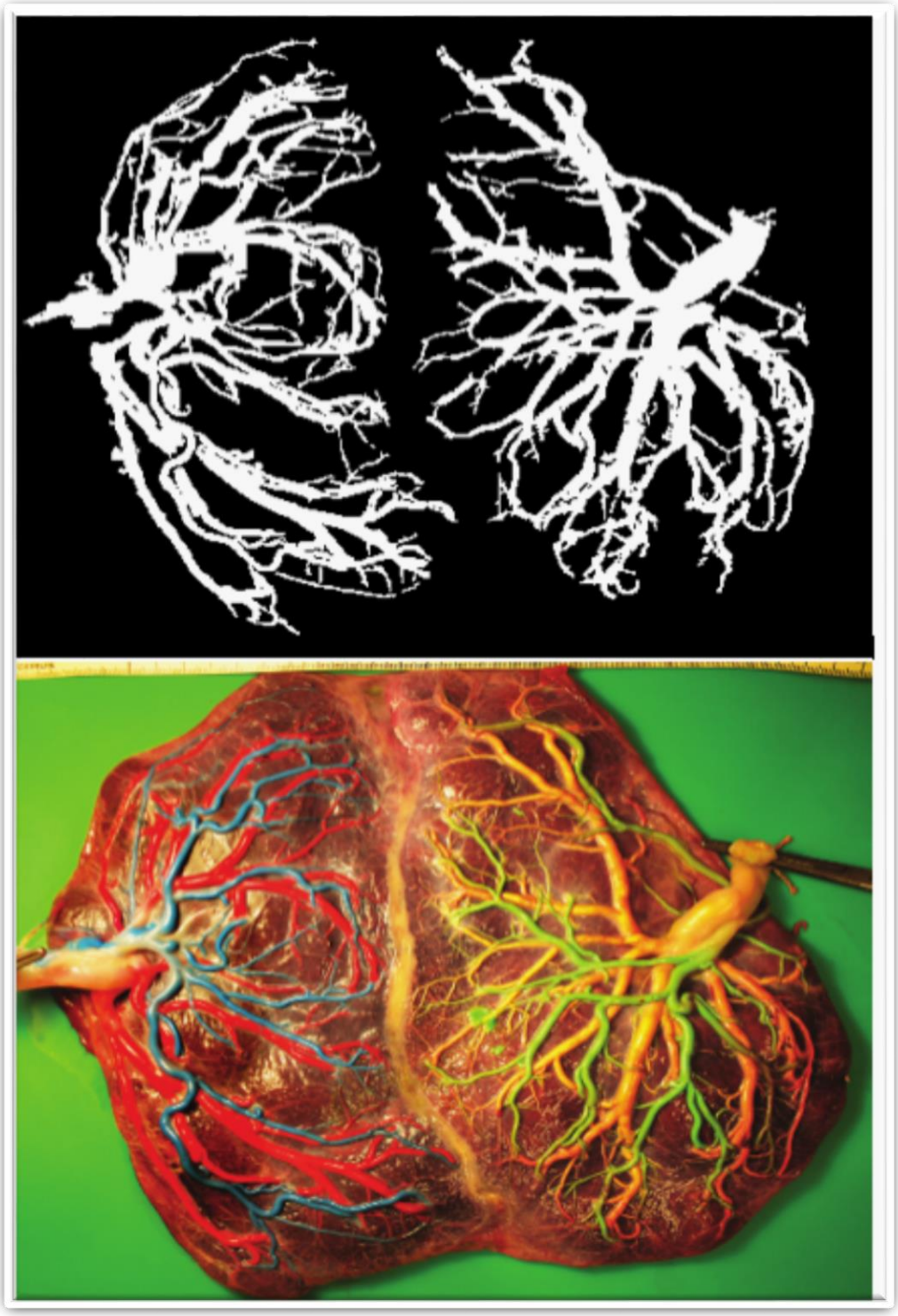


***Figura 28*** *Etiquetado a través de InteractLabeler.*

Las imágenes al ser de tamaño 365 x 488, son modificadas para que no exista conflicto con la reducción de las capas max pooling, que disminuyen el tamaño por un factor de 2 y al ser impar una de las dimensiones, generaría ruido en las operaciones. Se agrega un padding de ceros y la imagen termina que entra a la red neuronal termina siendo de tamaño: 368 x 496.

Para que las dimensiones de los pixeles de las imágenes sean parecidas y se le facilite el aprendizaje a la red neuronal debido a la distribución, se propone normalizar las imágenes, dividiendo el tensor de la imagen entre 255 (operación realizada para cada elemento), como el rango de valores de cada elemento en cada uno de los 3 canales (RGB) varía entre 0 y 255, el valor final después de normalizar estará entre 0 y 1, es decir, el formato de los tensores de las nuevas imágenes se convierte en flotante.

## IV.III Entrenamiento y Prueba

La forma común de entrenamiento y prueba consiste en partir la base de datos en dos subconjuntos mutuamente exclusivos (método holdout) llamados conjunto de entrenamiento y conjunto de prueba. El conjunto de entrenamiento se le da al clasificador y el mismo es probado en el conjunto de prueba. Asumiendo que la precisión incrementa mientras el clasificador vea más ejemplos, este método es un estimador pesimista porque sólo una parte de los datos se le dan para el entrenamiento. Entre más datos se dejen para el conjunto de prueba, el clasificador tendrá mayor bias; sin embargo, menos instancias de conjunto de pruebas significa que el intervalo de confianza para la precisión será más amplio [23].

## IV.IV Partición

Buscando aprovechar al máximo el tamaño de la base de datos obtenida, se propone usar el método Cross-validation K-fold. En lugar de dividir la base de datos en el conjunto de entrenamiento y en conjunto de prueba, se aprovecharán todas las imágenes para ambos casos. El conjunto de datos $D$ es dividido aleatoriamente en $k$ conjuntos mutuamente exclusivos (folds) $D_1, D_2, \ldots, D_k$ de aproximadamente el mismo tamaño. El clasificador es entrenado y probado k veces; cada tiempo $t \in \{1,2, \ldots, k\}$, es entrenado en $D/D_t$ y probado en $D_t$. El estimado de precisión es el número total de clasificaciones correctas, dividido entre el número de instancias en el conjunto de datos [23].

Por ejemplo, si k=10 y se cuenta con 200 datos, cada subconjunto contará con 20 datos; en 9 subconjuntos se entrenará la red neuronal y en el décimo se probará la red para el primer experimento, se deben hacer k experimentos y en cada uno de ellos irá cambiando el subconjunto de prueba, por lo tanto, el conjunto de datos que se tienen se utilizará en su totalidad tanto para entrenamiento como para prueba.

En este método la sensibilidad de entrenamiento (sensibilidad debido a cambios en el conjunto de entrenamiento) es mucho mayor que la sensibilidad de partición (sensibilidad debido a cambios en la partición, en el valor de k). Lo recomendado es usar k=10 porque se vuelve un modelo sin bias [24].

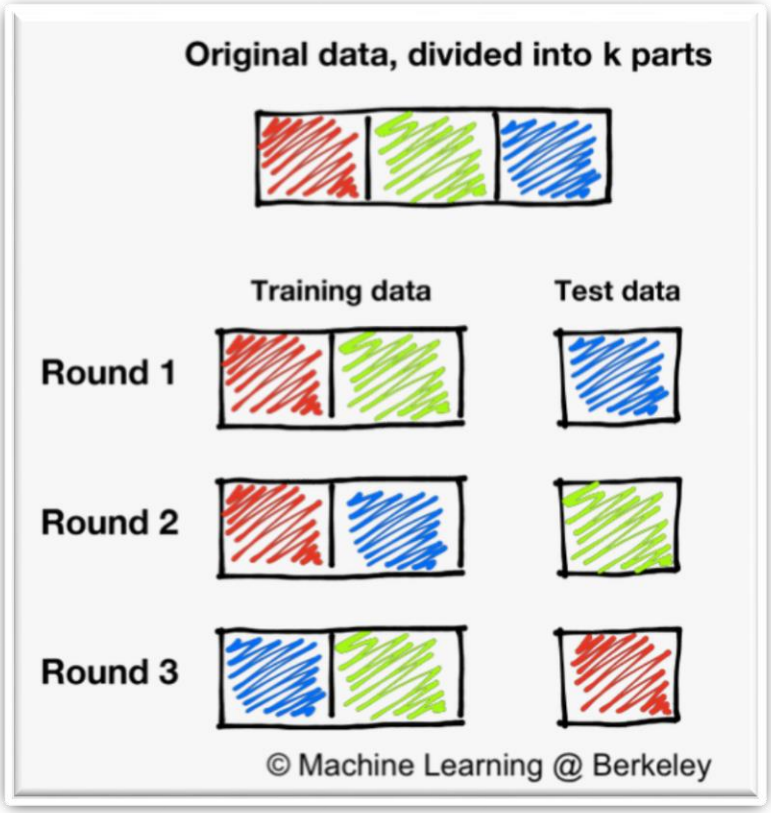



***Figura 29*** *Ejemplo de partición k- fold con k=3 [25].*

Por lo tanto K-fold no depende del valor que asignemos como partición, solamente depende de los datos de entrenamiento, es más robusto que el método holdout, pero tiene un costo computacional más grande debido a que para cada epoch (cuando se entrena el conjunto de entrenamiento completo) en holdout es necesario simplemente entrenar los datos del conjunto de entrenamiento pero en K-fold, se debe entrenar las k particiones, que en el caso de k=10, el epoch es 10 veces más grande en holdout.

## IV.V Métricas

Una métrica es una función usada para juzgar el desempeño del modelo. Las métricas explicadas a continuación serán utilizadas para medir el desempeño de los diferentes modelos que serán probados para encontrar el que segmente las imágenes con mejor calidad.

La información bruta producida por un esquema de clasificación durante la prueba son recuentos de las clasificaciones correctas e incorrectas de cada clase. Esta información normalmente se muestra en una matriz de confusión. Una matriz de confusión es una forma de tabla de contingencia que muestra las diferencias entre la verdad y las clases predichas para un conjunto de ejemplos etiquetados [26].

| n=165 | Predicted: NO | Predicted: YES |
|---|---|---|
| Actual: NO | 50 | 10 |
| Actual: YES | 5 | 100 |

***Figura 30*** *Matriz de confusión con 165 casos, 2 clases (si o no).*

La matriz de confusión tiene cuatro categorías: Verdaderos positivos, Falsos positivos, verdaderos negativos y falsos negativos [27].

Aunque la matriz de confusión muestra toda la información acerca del desempeño del clasificador, otras medidas útiles pueden ser extraídas de ella para ilustrar algún criterio de desempeño, por ejemplo [26]:

Exactitud. Proporción de resultados correctos que obtiene el clasificador [21].

$$\frac{T_p + T_n}{C_p + C_n} = P(C)$$

Sensibilidad (relevancia). Mide la proporción de positivos identificados correctamente como positivos [21].

$$\frac{T_p}{C_p} = P(T_p)$$

Especificidad. Mide la proporción de negativos correctamente identificados como negativos [21].

$$\frac{T_n}{C_n} = P(T_n)$$

$T_p$ y $T_n$ son el número de positivos verdaderos y negativos verdaderos respectivamente, $F_p$ y $F_n$ son los números de falsos positivos y falsos negativos respectivamente. Los totales de la fila, $C_p$ y $C_n$, son el número de ejemplos verdaderamente positivos y negativos, los totales de columna, $R_p$y $R_n$ son el número de positivos predichos y negativos predichos, $N$ es el número total de ejemplos ($N = C_p + C_n = R_p + R_n$) [26].

### i. ROC

La curva ROC (Receiver Operating Characteristic) se recomienda para evaluar problemas de decisión binaria (solamente dos clases), muestra cómo varía el número de ejemplos positivos correctamente clasificados con el número de ejemplos negativos incorrectamente clasificados [27].

En la curva ROC, se grafica la tasa de falsos positivos en el eje x y la tasa de verdaderos positivos en el eje y. La tasa de falsos positivos mide la fracción de ejemplos negativos que son incorrectamente clasificados como positivos. La tasa de verdaderos positivos mide la fracción de ejemplos positivos que son correctamente etiquetados. El objetivo de la curva ROC es estar en la esquina superior izquierda [27].

$$Tasa\ de\ verdaderos\ positivos = \frac{T_p}{T_p + F_n}$$

$$Tasa\ de\ falsos\ positivos = \frac{F_p}{F_p + T_n}$$

El área bajo la curva de la curva ROC es la probabilidad de que un miembro elegido al azar de la clase tenga una probabilidad estimada menor de pertenecer a la clase 0 que un miembro elegido al azar de la clase 0.

### ii. Precision-Recall

Es una medida útil para el éxito de predicción cuando las clases están muy desequilibradas. En la recuperación de información, la precisión es una medida de relevancia del resultado, mientras que recall es una medida de la cantidad de resultados verdaderamente relevantes que se devuelven [28].

La curva precision-recall muestra la compensación entre precisión y relevancia para diferentes umbrales. Cuando el área bajo la curva es grande, representa gran precisión y gran relevancia, donde gran precisión se refiere a una tasa baja de falsos positivos (predicción equivocada de positivos) y gran relevancia se refiere a una tasa baja de falsos negativos (predicción equivocada de negativos) [28].

Un sistema con alta relevancia, pero baja precisión regresa muchos resultados, pero muchas de las predicciones de etiquetas son equivocadas en comparación con las etiquetas de entrenamiento. Un sistema con alta precisión, pero baja relevancia es lo opuesto, entrega muy pocos resultados, pero la mayoría de las etiquetas que predice son correctas comparadas con las etiquetas de entrenamiento. Un sistema ideal con alta precisión y alta relevancia entrega muchos resultados con todos los resultados correctamente etiquetados [28].

Precisión ($P$) es definida como el número de verdaderos positivos ($T_p$) entre el número de verdaderos positivos más el número de falsos positivos ($F_p$) [28].

$$P = \frac{T_p}{T_p + F_p}$$

Relevancia ($R$) se define como el número de verdaderos positivos ($T_p$) entre el número de verdaderos positivos más el número de falsos negativos ($F_n$) [28].

$$R = \frac{T_p}{T_p + F_n}$$

Estas cantidades también están relacionadas con ($F_1\ score$) el puntaje, que se define como la media armónica de precisión y relevancia [28].

$$F1\ score = 2\frac{P\ \times\ R}{P + R}$$

En la curva Precision-Recall, grafica la relevancia en el eje x y la precisión en el eje y. El objetivo de la curva Precision-Recall es estar en la esquina superior derecha [27].

### iii. Jaccard Similarity

Mide la proximidad de dos conjuntos de datos eficientemente. Es el resultado de la división del número de características similares de ambos conjuntos entre el número de características totales [29].

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

### iv. Matthews Correlation Coefficient

Es una medida de la calidad de clasificadores binarios. Toma en cuenta verdaderos y falsos positivos y negativos y generalmente se considera como una medida equilibrada que se puede usar incluso si las clases son de tamaños muy diferentes. Es esencia un valor de coeficiente de correlación entre -1 y +1. Un coeficiente de +1 representa una predicción perfecta, 0 una predicción aleatoria promedio y -1 una predicción inversa [30].

## V. Iteraciones

Por la facilidad de programación y por ser de código abierto la red se programó en Python, utilizando Keras [31] (librería de código abierto escrita en Python que ya incluye funciones de redes neuronales, es decir, no es necesario programarlas desde cero), además de los recursos con los que se cuenta, una computadora con GPU, NVIDIA GeForce GTX 950M, con cuDNN v7.5, las principales arquitecturas propuestas se muestran a continuación.

## V.I Arquitectura Base

La red se compone de un codificador y un decodificador de 4 capas cada uno, como el propuesto de SegNet-Basic en [2]. Cada capa del codificador consta de una capa convolucional con 64 filtros y un tamaño de filtro de 3 x 3, activación ReLu seguido de una capa pooling con una ventana de 2 x 2 y un stride de 2, seguido de un dropout con probabilidad de 0.5. El decodificador también tiene una capa convolucional con 64 filtros y activación ReLu, utiliza una operación inversa de max pooling (upsampling) también en ventanas de 2 x 2 con un stride de 2, a esta operación le sigue el dropout.

La operación inversa de max pooling utilizada, incrementa el tamaño del tensor de entrada y repite los valores para llenar los espacios faltantes.

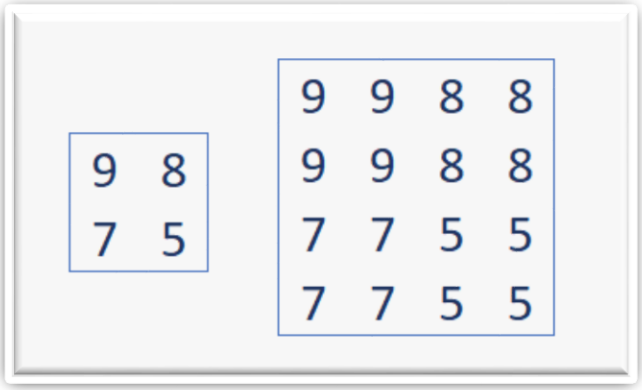



***Figura 31*** *Operación inversa de max pooling, la matriz de la izquierda es la de entrada, la matriz de la derecha repite los valores para aumentar su tamaño y llenar los espacios faltantes.*

La cantidad de filtros es mucho menor a la SegNet-Basic debido a la cantidad de parámetros y el costo computacional que involucra, entre más filtros, puede haber una función que se acople mejor a las imágenes, pero incrementa mucho el tiempo que tarda en entrenarse la red.

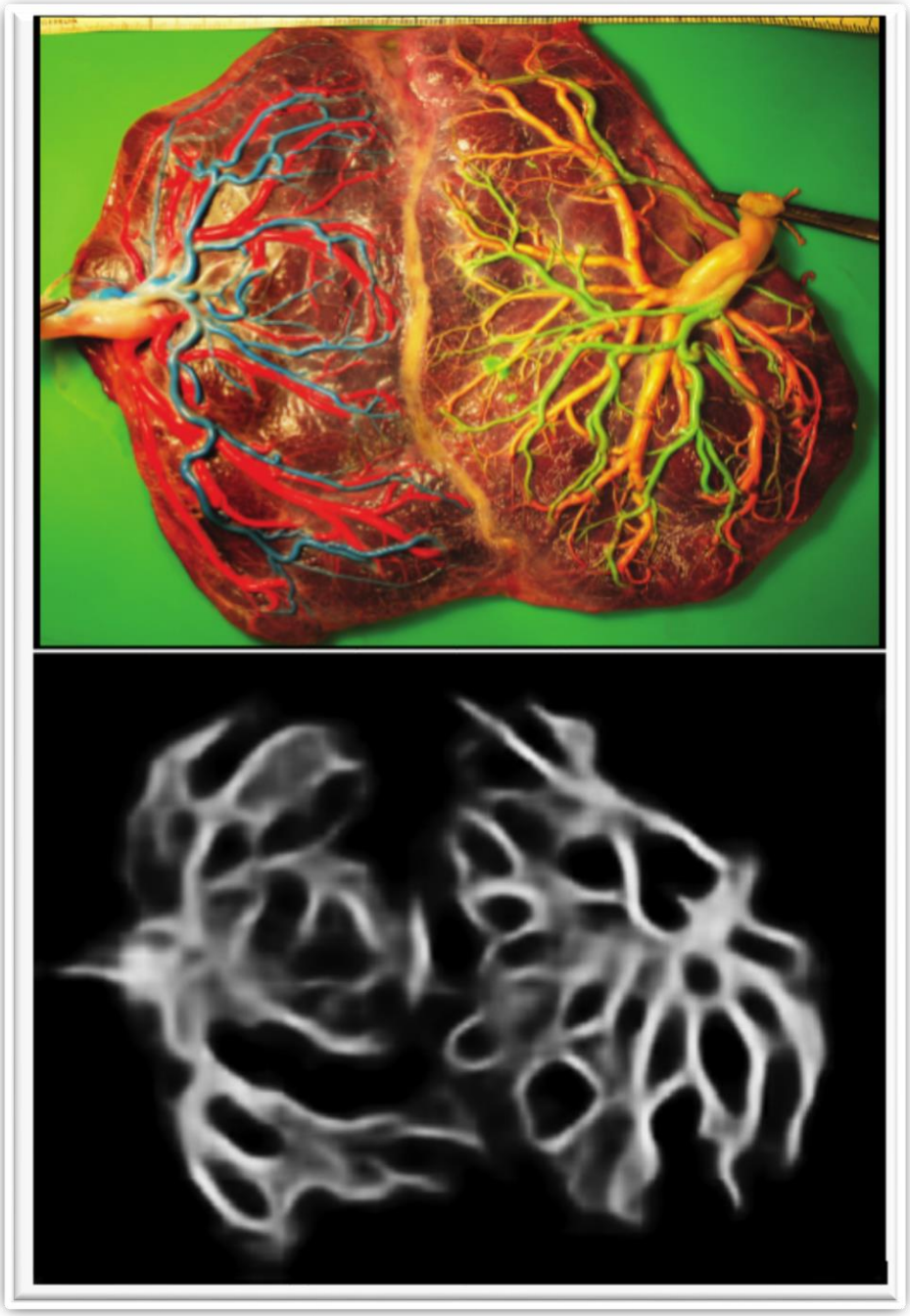


***Figura 32*** *Segmentación realizada por la arquitectura base.*

Con la arquitectura base la segmentación lograda separa las venas y arterias del resto de la imagen, pero no es de la mejor calidad la segmentación, hay zonas de mucha incertidumbre, la segmentación no está bien definida, pero como primera iteración es aceptable.

## V.II Segunda arquitectura

Para la segunda arquitectura sólo se hizo un cambio en la parte del decodificador, éste fue el cambio más significativo del trabajo, puesto que mostró una gran mejora con respecto a las iteraciones anteriores.

La arquitectura base segmenta la imagen con demasiado ruido, genera una imagen demasiado borrosa, principalmente en el borde de las venas y arterias, tiene problemas al delinear con detalle la orilla de los elementos, especialmente en zonas cercanas a otras venas o arterias. Este problema es resultado de las operaciones pooling, la cual en la parte del codificador ayuda a extraer las características principales de cada zona, pero al reducir la imagen de tamaño, se va perdiendo la referencia de la posición de cada elemento, se pierde mucha información, en especial la relativa a la posición precisa de los elementos, característica importante para la segmentación y, por lo tanto, al incrementar el tamaño de la imagen en el decodificador, la imagen producida es borrosa.

El nuevo decodificador consiste en buscar replicar la parte de upsampling propuesta en [2] y [3] para solucionar lo borroso de la imagen segmentada, en donde para cada operación pooling realizada, se guardan los índices, es decir, la posición en dónde se encontraron los valores máximos de cada ventana, los cuáles son posteriormente utilizados en el decodificador, para que al realizar el upsampling, se haga con la referencia de los índices guardados, llenando de ceros el resto de la matriz.

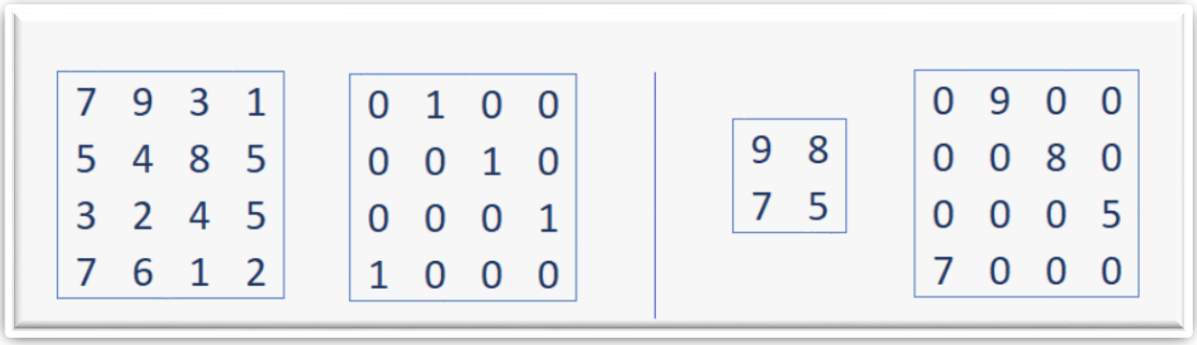



***Figura 33*** *En la parte izquierda de la imagen, la primera matriz es la matriz a la que se le aplicará la operación pooling, la segunda matriz son los índices guardados; en la parte derecha de la imagen, la primera matriz es la matriz a la que se le aplicará el upsampling, la segunda matriz es el resultado de la nueva operación inversa de pooling.*

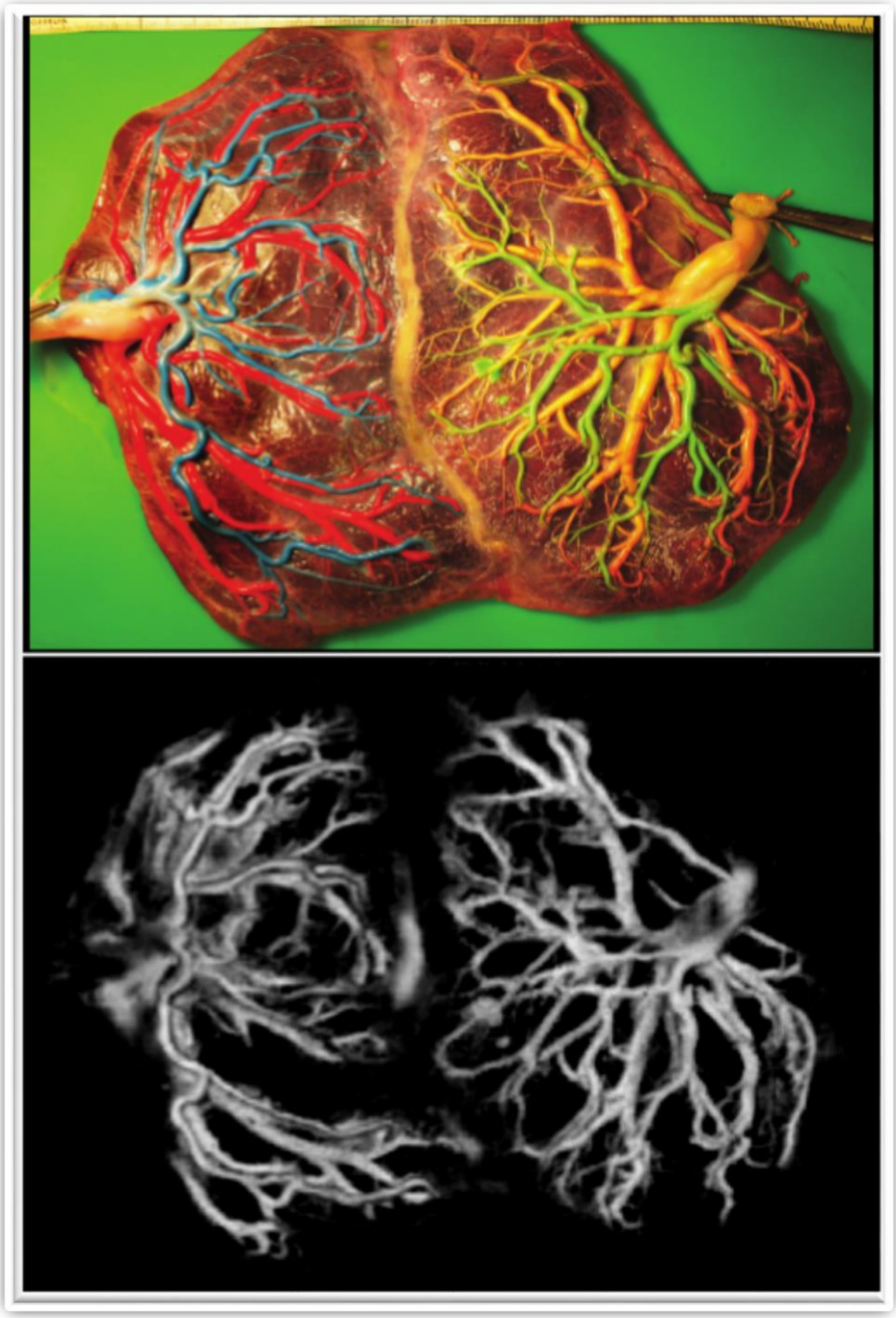


***Figura 34*** *Resultado de la segmentación de la segunda iteración de red neuronal.*

El cambio se puede apreciar visualmente, causó una mejoría en el detalle del borde de cada vena y arteria, con respecto a la arquitectura base. Se puede observar con mayor claridad el contorno formado por cada brazo de vena o arteria, aunque en zonas donde se encuentran muy cerca, aún se cuenta con algo de ruido. También se debe destacar que, en el ejemplo mostrado en la Figura 34 anterior, la parte más problemática

es donde se encuentran las venas de color rojo, color menos contrastante con respecto a la placenta, comparado con los otros tres colores utilizados y que, en las zonas ruidosas, la imagen de que entra a la red tiene mayor brillo.


***Figura 35*** *Otro resultado de la segmentación de la segunda iteración de red neuronal.*

En la imagen anterior, visualmente el resultado es mucho mejor, prácticamente no hay secciones con ruido, la calidad de la segmentación es muy buena, se logra apreciar con claridad incluso cuando existen cruces entre venas y arterias. Además de que los brazos más pequeños de venas y arterias también fueron correctamente segmentados.

## V.III Más filtros

Para la tercera iteración al ya tener una mejora importante y por lo tanto menor necesidad de hacer muchos cambios, se probó aumentar el número de filtros en las capas convolucionales buscando generar una función con mayor complejidad especialmente en las capas más profundas para ayudar a las características más complejas reconocidas por la red convolucional. Esta modificación, por el costo computacional que involucra, significa mayor tiempo de espera para observar los resultados, por eso se decidió intentar implementar más parámetros hasta esta etapa.

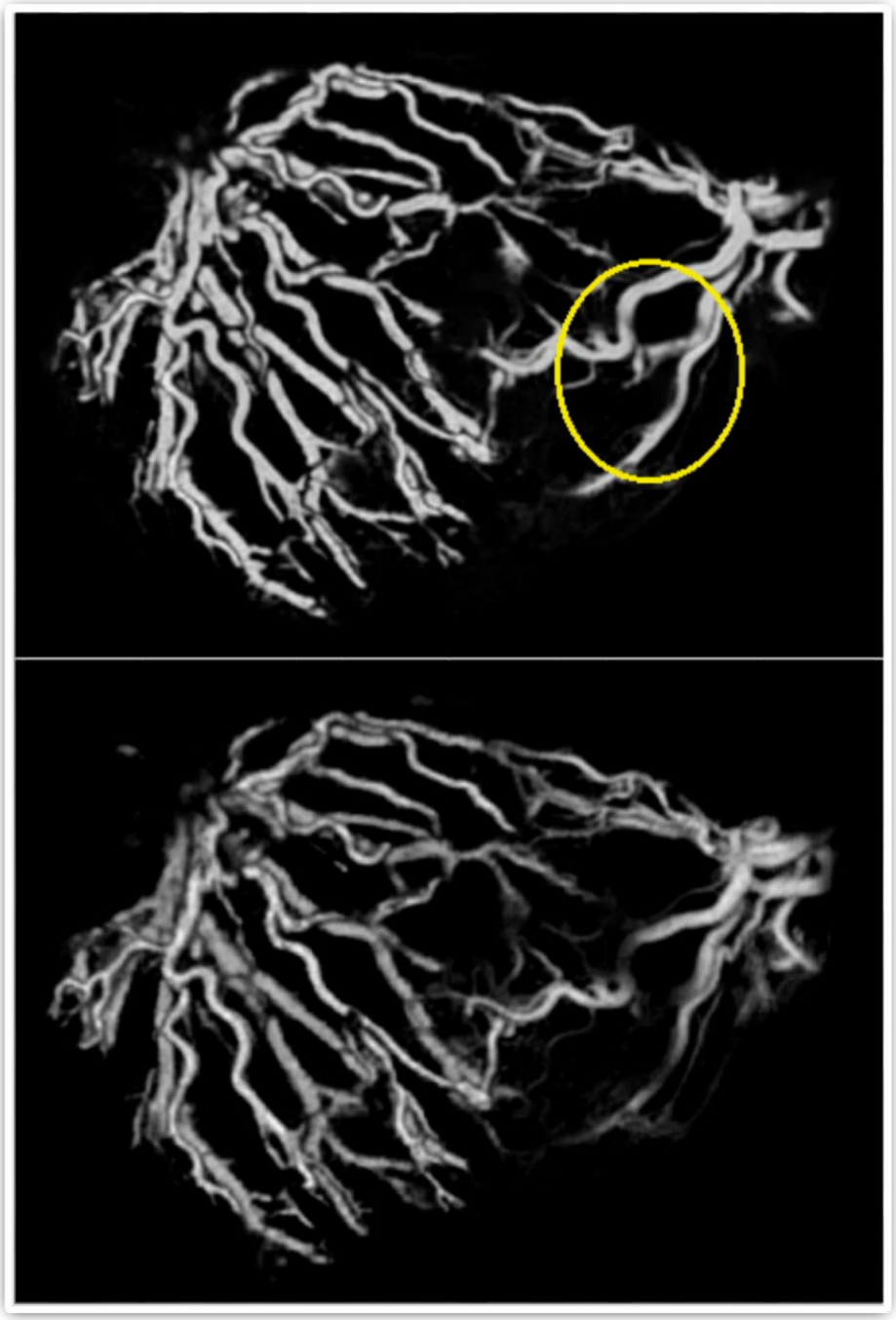


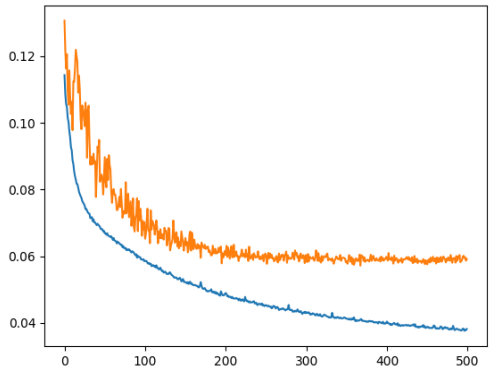
***Figura 36*** *En la parte superior se encuentra la imagen predicha de la red neuronal con mayor número de filtros y la segmentación inferior proviene de la red con menor número de filtros.*

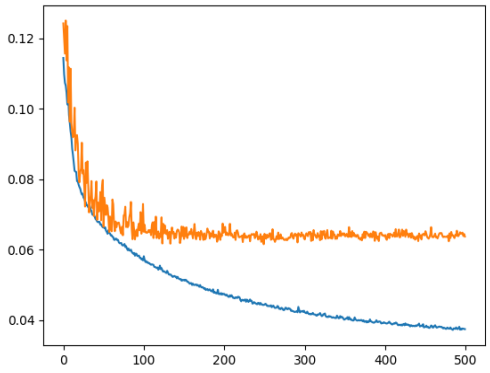
Con mayor número de filtros la imagen es más clara y definida en general, pero particularmente mejora en la zona marcada por el circulo amarillo donde la red omitía secciones de la arteria o vena y gracias a una mayor cantidad de parámetros, logra definirlas completamente.

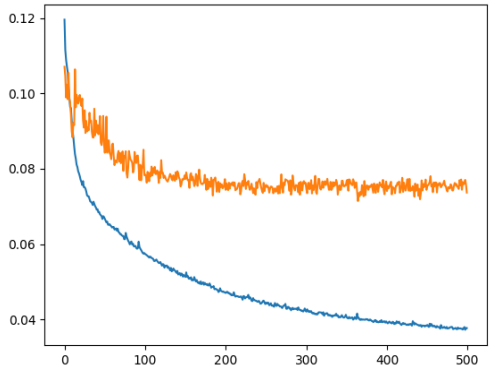
## V.IV Parámetros

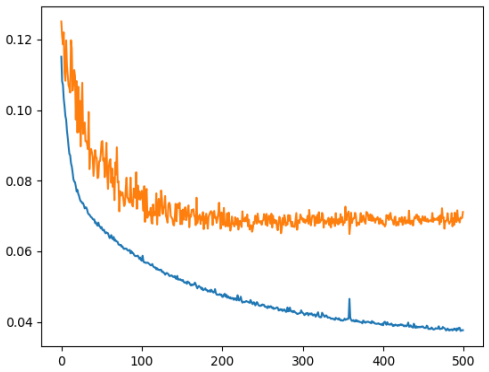
Se dividen los datos en 10 partes para ser entrenados con K-fold, se hacen 500 epochs para cada parte del K-fold, se usa el optimizador Nadam con un parámetro de tasa de aprendizaje de 0.0002, la función de pérdida es el error medio cuadrático, para asegurar que la función de pérdida disminuye se grafica la función por iteración. La inicialización de los parámetros se hace con inicialización normal Xavier.

Durante el entrenamiento se grafica la función de costo para asegurar que el modelo está funcionando correctamente, se espera siempre una disminución en cada epoch, pero como cada epoch utiliza varias imágenes como entrenamiento debido al método de Cross-validation, puede ser que no siempre disminuya, aunque la tendencia general siempre debe acercarse a cero.



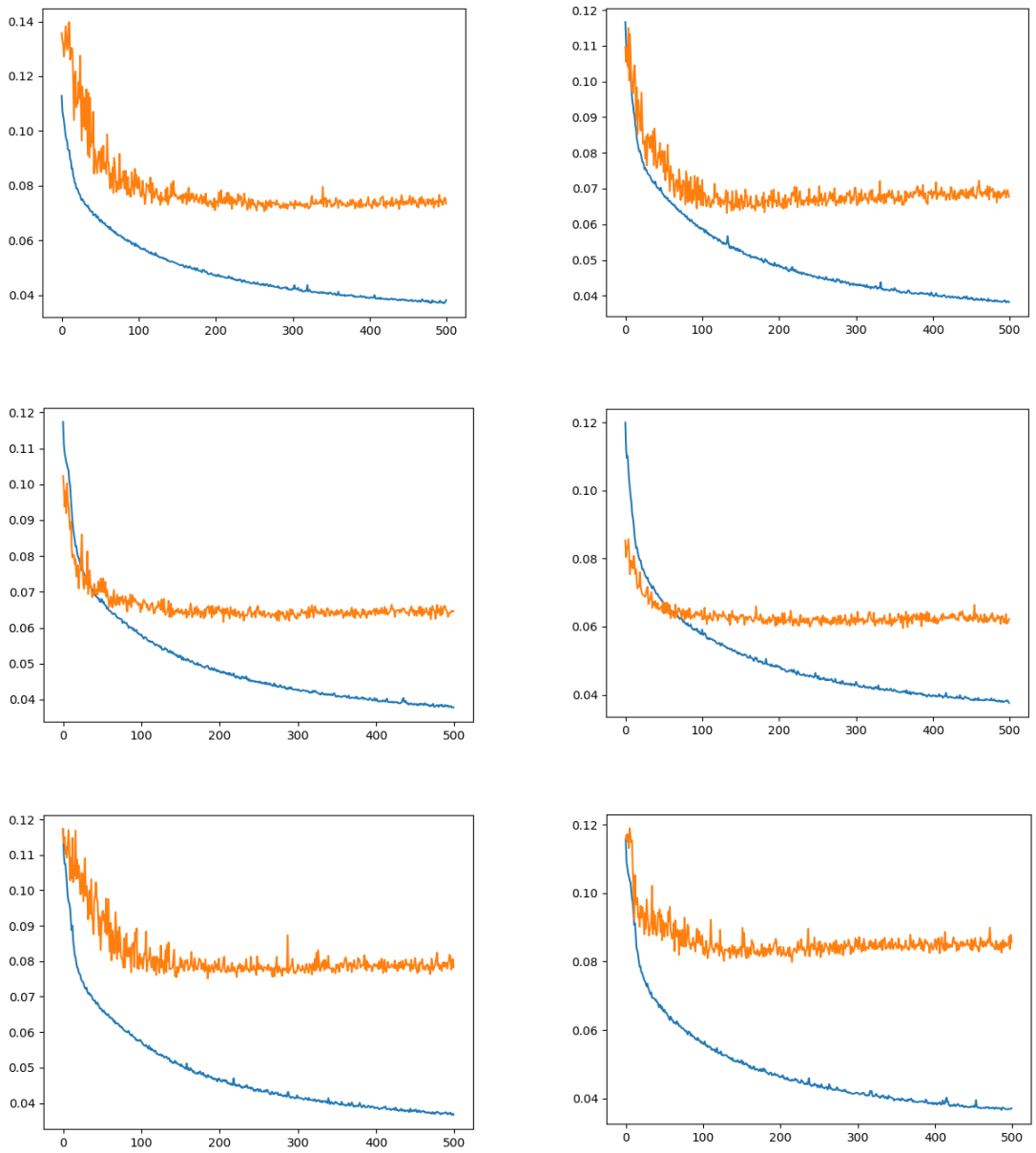



***Figura 37*** *Gráfica de la función de costo por epoch, en el eje x se muestran los epochs, en el eje y el valor de la función de costo, en azul se muestra la curva que representa las imágenes de entrenamiento y en naranja la curva que representa las imágenes de prueba, cada gráfica representa cada uno de los folds utilizados.*

Las curvas de las imágenes de prueba en varios casos se encuentran muy alejadas de la curva naranja, lo cual podría indicar variance en ese fold particular, debido a que en el conjunto de entrenamiento la función logra acercarse bastante a cero, pero la curva de entrenamiento tiene mucha dificultad para obtener el mismo resultado y ese fold en particular no generaliza de la mejor forma y por lo tanto se le complica predecir imágenes no vistas.

También se destaca en las gráficas que la curva azul, en el epoch 500 tiene una tendencia a seguir disminuyendo, lo que indica que con más epochs, puede disminuir y mejorar el resultado en las imágenes de entrenamiento, en cambio, las curvas naranjas parecieran haber alcanzado el mínimo resultado posible.

## V.V Comparación

En la siguiente tabla se muestran las métricas y los resultados de nuestras iteraciones, los cuáles serán determinantes para seleccionar la red con el mejor desempeño y también serán analizados para sugerir el trabajo futuro que se podría hacer para mejorar el comportamiento de la red.

| Métricas | Primera | Segunda | Última |
|---|---|---|---|
| Área bajo la curva Precision-Recall | 0.578 | 0.6635 | 0.6747 |
| Área bajo la curve ROC | 0.8889 | 0.8901 | 0.9013 |
| Precisión | 0.8961 | 0.9108 | 0.9132 |
| Parámetros | 260,865 | 260,865 | 463,761 |
| Time [h] | 25 | 30 | 47.5 |

Claramente la mejor iteración en resultados es la última, aunque el costo computacional es mayor como se esperaba debido al número de parámetros, pero es mucho menor el número de parámetros de esta propuesta en comparación a la mayoría de los utilizados comúnmente en redes neuronales que normalmente son millones. Los resultados mejoran en cada iteración, pero tampoco son muy diferentes y para la capacidad que tiene una red neuronal son un poco bajos, la explicación de este comportamiento.

Una de las razones por las que las redes neuronales no tienen valores de métricas por arriba de 0.96, que sería lo esperado, es que el etiquetado de algunas imágenes tiene muchas deficiencias por las herramientas con las que se contaron en el momento del etiquetado y por el tiempo dedicado para esa tarea. Pero a pesar de que no es algo deseable entrenar con imágenes incorrectamente etiquetadas, gracias a eso se logró observar que la red tiene la habilidad de sobreponerse a esas deficiencias.

En la siguiente imagen se puede comprobar el hecho de que la red neuronal es capaz de superar obstáculos como el que la imagen etiquetada manualmente no sea de la mejor calidad, se observa en la parte superior que el borde amarillo abarca una región que, comparándola con la imagen original (ground truth), no existen venas ni arterias, pero el aprendizaje de la red convolucional hizo que se adaptara incluso de mejor forma que la imagen etiquetada a mano. Igualmente se logra percibir que la segmentación genera más brazos pequeños de venas y arterias que la imagen etiquetada manualmente.

***Figura 38*** *En la parte superior de la imagen se encuentra la segmentada por la red neuronal, binarizada y con color amarillo se agrega el contorno de la imagen etiquetada manualmente. En la parte inferior se muestra la imagen etiquetada manualmente.*

A pesar de que el que la red convolucional sea lo suficientemente robusta para sobreponerse a errores en la imagen etiquetada manualmente, los resultados numéricos del desempeño de la red se verán afectados

negativamente, ya que las métricas de desempeño son una medida de la segmentación en comparación de la imagen etiquetada manualmente, aunque como ya se mencionó, la red hizo un gran trabajo que se puede distinguir a simple vista, el desempeño numérico se verá penalizado. Por ejemplo, en la imagen pasada, la zona de arriba que se encuentra mal etiquetada y la red no clasificó como vena o arteria, para las métricas contará como un falso negativo, a pesar de ser un verdadero negativo.

## V.VI Arquitectura final

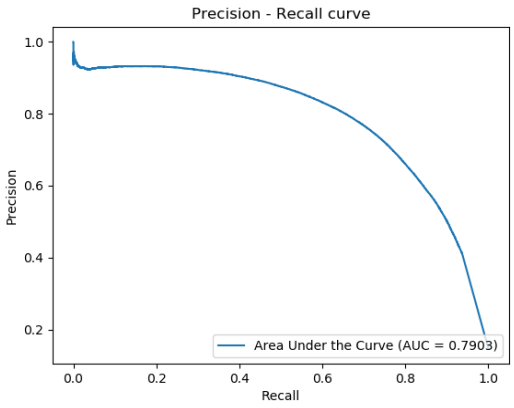
En la siguiente tabla se explica la arquitectura de la red neuronal que mejores resultados mostró, la capa de mayor número de parámetros, pero que en comparación con redes publicadas para segmentación como Segnet (14.7M) [2] o DeconvNet(252M) [3], la red solo cuenta con 463.7K.

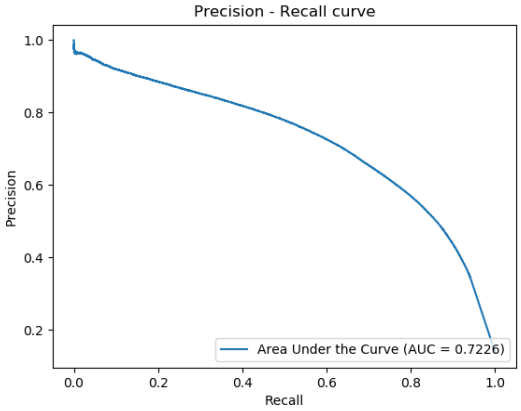
| Capa | Tamaño de filtro | Stride | Salida |
|---|---|---|---|
| capa de entrada | | | (3, 368, 496) |
| convolution_1 | 3 x 3 | 1 | (64, 368, 496) |
| maxpooling_1 | 2 x 2 | 2 | (64, 184, 248) |
| dropout_1 | | | (64, 184, 248) |
| convolution_2 | 3 x 3 | 1 | (80, 184, 248) |
| maxpooling_2 | 2 x 2 | 2 | (80, 92, 124) |
| dropout_2 | | | (80, 92, 124) |
| convolution_3 | 3 x 3 | 1 | (96, 92, 124) |
| maxpooling_3 | 2 x 2 | 2 | (96, 46, 62) |
| dropout_3 | | | (96, 46, 62) |
| convolution_4 | 3 x 3 | 1 | (112, 46, 62) |
| maxpooling_4 | 2 x 2 | 2 | (112, 23, 31) |
| unpooling_1 | 2 x 2 | 2 | (112, 46, 62) |
| convolution_5 | 3 x 3 | 1 | (96, 46, 62) |
| unpooling_2 | 2 x 2 | 2 | (96, 92, 124) |
| convolution_6 | 3 x 3 | 1 | (80, 92, 124) |
| dropout_4 | | | (80, 92, 124) |
| unpooling_3 | 2 x 2 | 2 | (80, 184, 248) |
| convolution_7 | 3 x 3 | 1 | (96, 184, 248) |
| dropout_5 | | | (96, 184, 248) |
| unpooling_4 | 2 x 2 | 2 | (64, 368, 496) |
| convolution_8 | 3 x 3 | 1 | (64, 368, 496) |
| dropout_8 | | | (64, 368, 496) |
| convolution_9 | 3 x 3 | 1 | (1, 368, 496) |

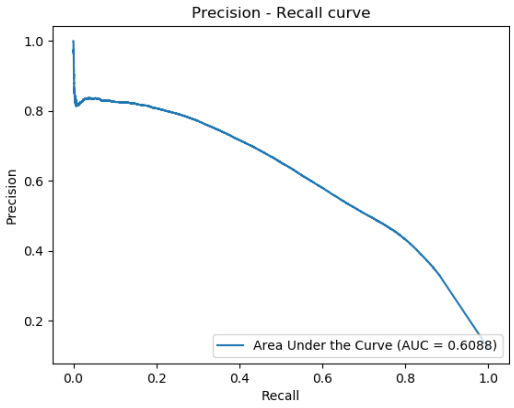
Otras métricas de desempeño de la arquitectura final se muestran a continuación:

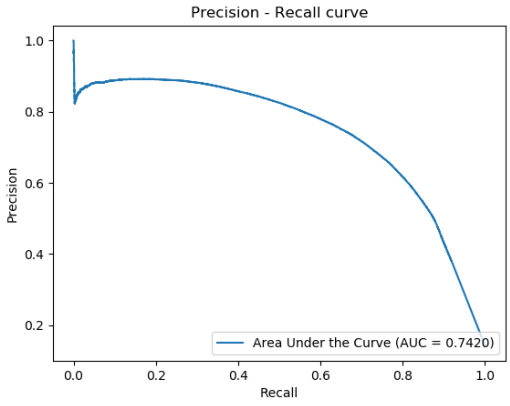
| Métricas | Última |
|---|---|
| Especificidad | 0.9688 |
| Sensibilidad | 0.5339 |
| Precisión | 0.7155 |
| Jaccard similarity | 0.9132 |
| Matthews correlation coefficient (MCC) | 0.5707 |
| F1 score | 0.6097 |

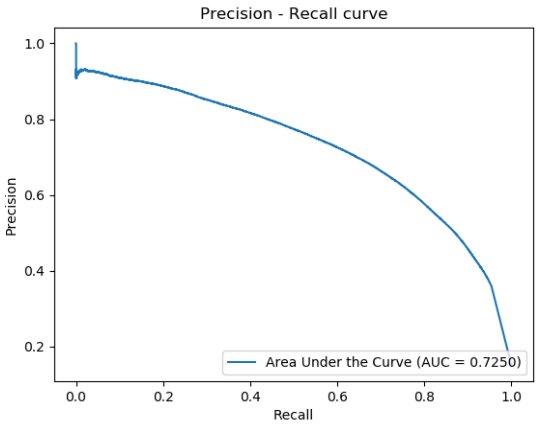
A continuación, se muestras las gráficas obtenidas por la arquitectura para cada fold:

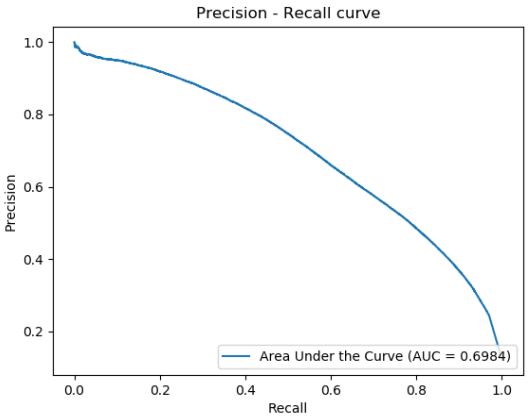

Precision - Recall curve
Precision
Recall
Area Under the Curve (AUC = 0.7903)


Precision - Recall curve
Precision
Recall
Area Under the Curve (AUC = 0.7226)


Precision - Recall curve
Precision
Recall
Area Under the Curve (AUC = 0.6088)


Precision - Recall curve
Precision
Recall
Area Under the Curve (AUC = 0.7420)


Precision - Recall curve
Precision
Recall
Area Under the Curve (AUC = 0.7250)


Precision - Recall curve
Precision
Recall
Area Under the Curve (AUC = 0.6984)

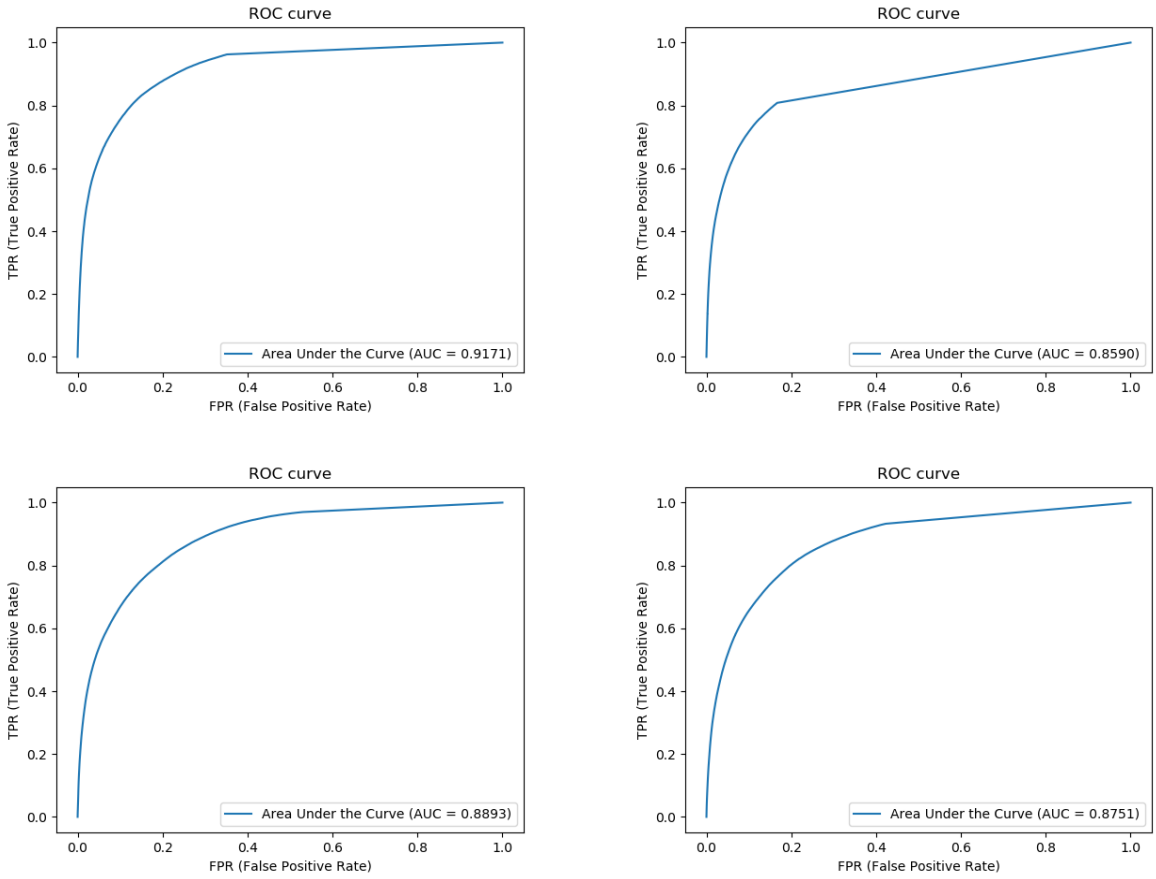
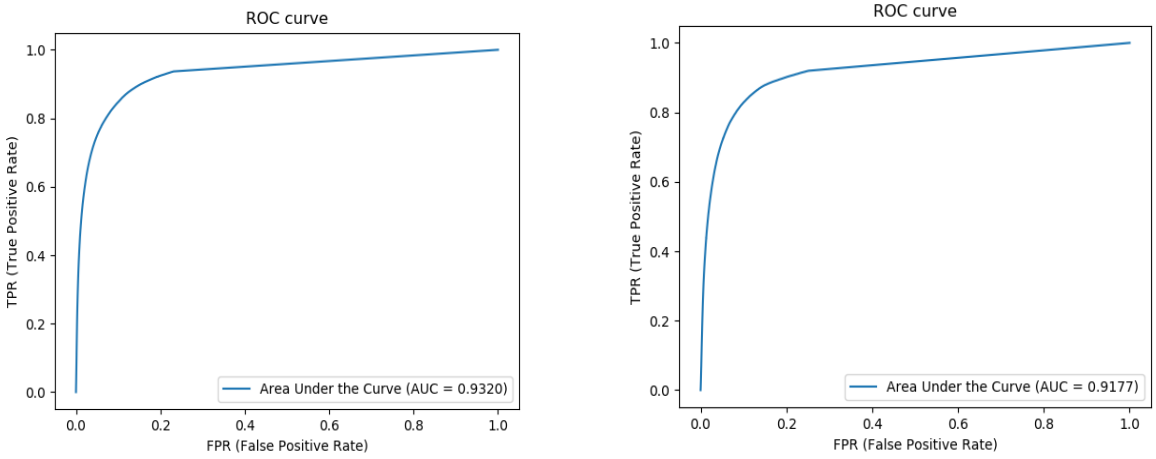
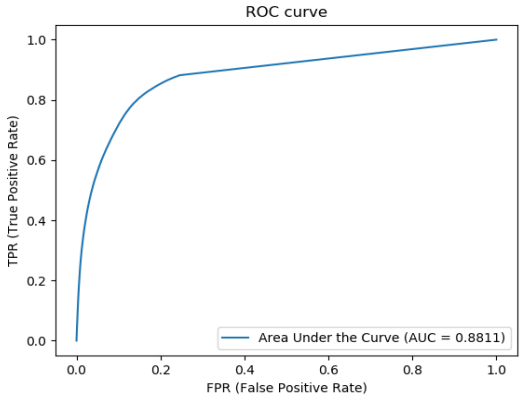
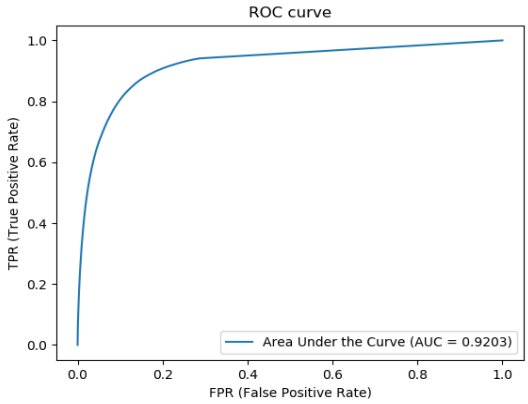
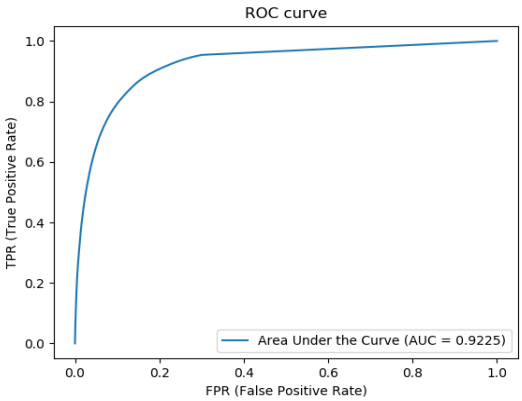


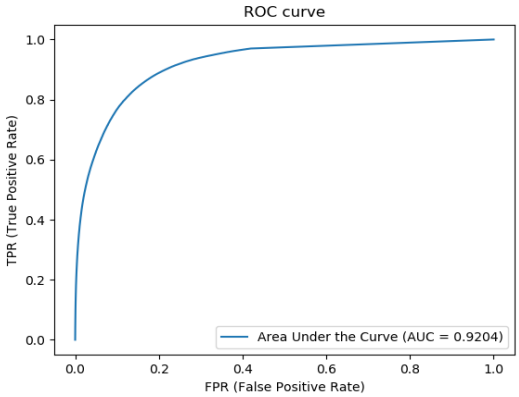

***Figura 39*** *Gráficas de Precision-Recall, donde el eje x tiene la tasa de falsos positivos y el eje y la tasa de verdaderos positivos, cada gráfica representa la métrica de cada fold.*

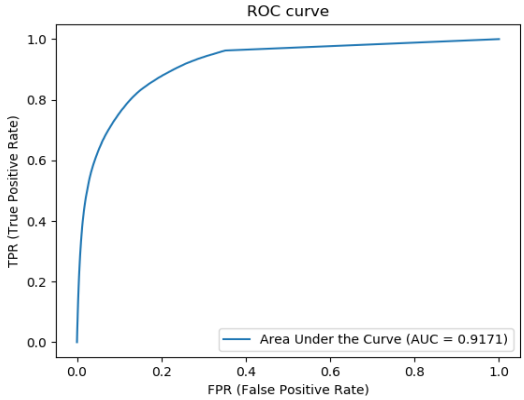


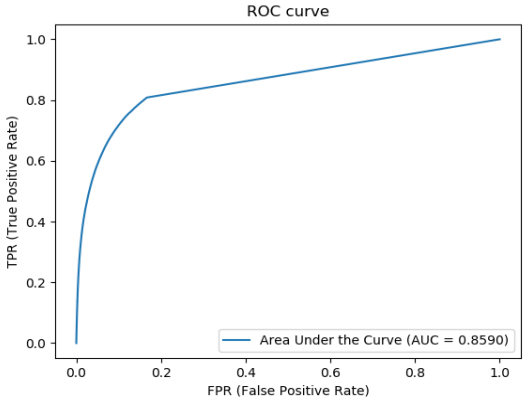

ROC curve
TPR (True Positive Rate)
FPR (False Positive Rate)
Area Under the Curve (AUC = 0.8811)

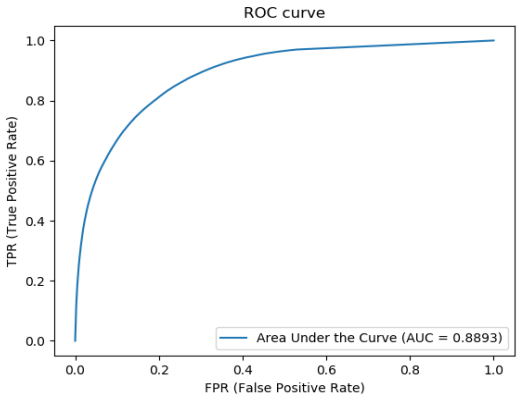

ROC curve
TPR (True Positive Rate)
FPR (False Positive Rate)
Area Under the Curve (AUC = 0.9203)


ROC curve
TPR (True Positive Rate)
FPR (False Positive Rate)
Area Under the Curve (AUC = 0.9225)


ROC curve
TPR (True Positive Rate)
FPR (False Positive Rate)
Area Under the Curve (AUC = 0.9204)


ROC curve
TPR (True Positive Rate)
FPR (False Positive Rate)
Area Under the Curve (AUC = 0.9171)


ROC curve
TPR (True Positive Rate)
FPR (False Positive Rate)
Area Under the Curve (AUC = 0.8590)


ROC curve
TPR (True Positive Rate)
FPR (False Positive Rate)
Area Under the Curve (AUC = 0.8893)

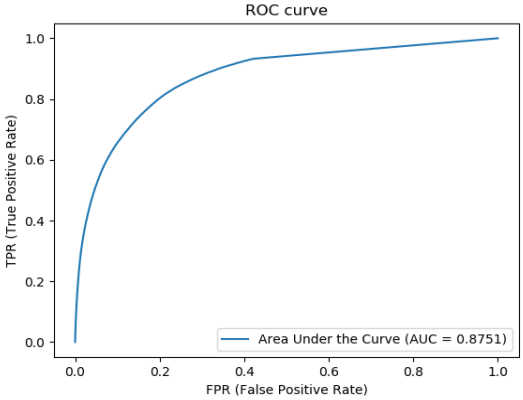

ROC curve
TPR (True Positive Rate)
FPR (False Positive Rate)
Area Under the Curve (AUC = 0.8751)

***Figura 40*** *Gráficas ROC, en el eje x se encuentra la tasa de falsos positivos y en el eje y se encuentra la tasa de verdaderos positivos.*

Durante el proceso de diseño se probaron diferentes cambios para mejorar el trabajo realizado por la primera iteración, entre ellos varios que no hicieron una mejora significativa y por esa razón no se mencionan en este trabajo, algunos de los otros cambios propuestos fueron cambiar la función de costo de error medio cuadrático a Cross-entropy, cambiar de inicialización Xavier a inicialización He, probar otros optimizadores como Gradient descent o Adam y se varió la tasa de aprendizaje para encontrar la mejor para el problema.

# VI. Validación

En el capítulo anterior se menciona la dificultad para obtener números concretos del desempeño de la red neuronal, para validar la arquitectura en este capítulo y poder comprar con otros métodos se segmentarán las imágenes de fondo de ojo, separando venas y arterias del resto de la imagen. La base de datos DRIVE [8] son imágenes utilizadas para el diagnóstico de algunas enfermedades a través de los ojos y el etiquetado fue hecho manualmente pero prácticamente no contienen errores y por lo tanto las métricas de desempeño serán acertadas y nos darán una mejor perspectiva de la calidad de la red neuronal convolucional.

DRIVE es una base de datos que permite el estudio en segmentación de venas en imágenes retinales. Invita a la comunidad a probar sus algoritmos en la base de datos y compartir los resultados con otros investigadores a través de sitio web [8].

***Figura 41*** *Base de datos DRIVE, a la izquierda la imagen del fondo de ojo y a la derecha la imagen segmentada manualmente [8].*

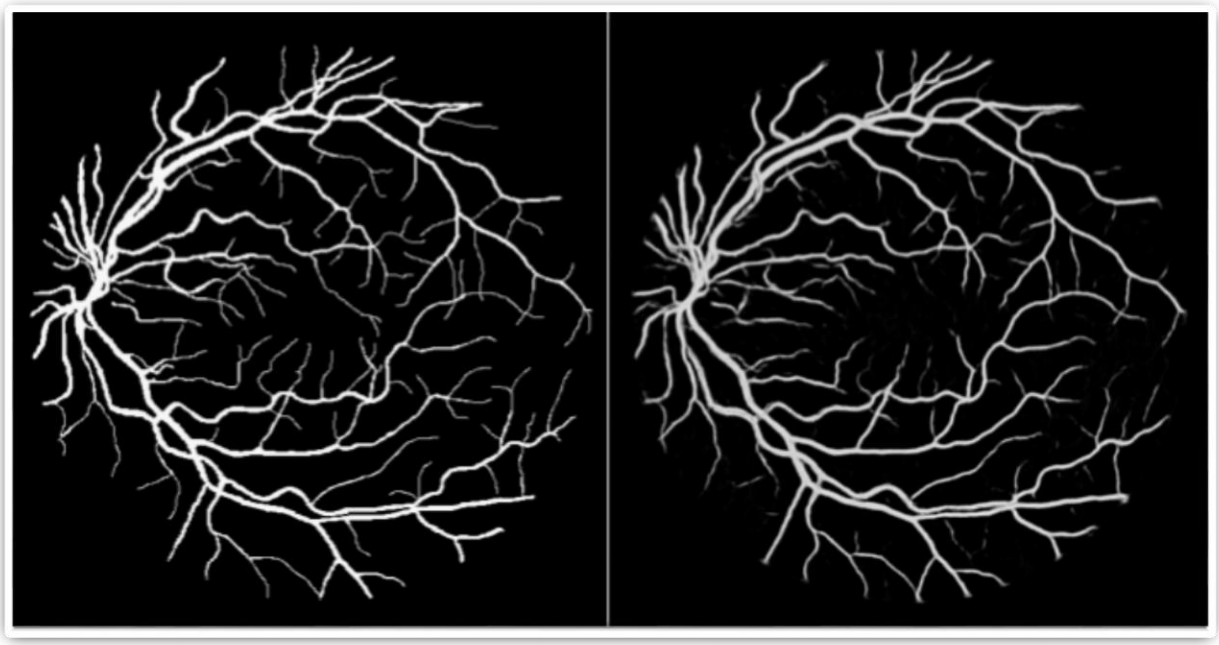


***Figura 42*** *En la izquierda se muestra la imagen etiquetada manualmente y en la derecha se muestra la imagen segmentada por la red neuronal.*

Visualmente se observa que la segmentación es muy buena, todos los parámetros y condiciones son las mismas que en la iteración de mayor número de filtros del capítulo anterior, la red hace un gran trabajo, teniendo un poco de dificultad solamente en los brazos más pequeños de venas y arterias, los más delgados, fuera de eso el resultado de la red neuronal es igual al de la ground truth.

En la siguiente imagen con mayor claridad se puede detectar que son muy pocos brazos de venas y arterias, los no detectados y segmentados, pero se muestra una imagen muy limpia, sin el ruido (zonas borrosas o grises) que se generaban con la base de datos de imágenes placentarias. Gracias a la calidad de las imágenes de entrenamiento se logra un mejor desempeño por lo menos visualmente, a continuación, se analizan las métricas de desempeño de la red convolucional.

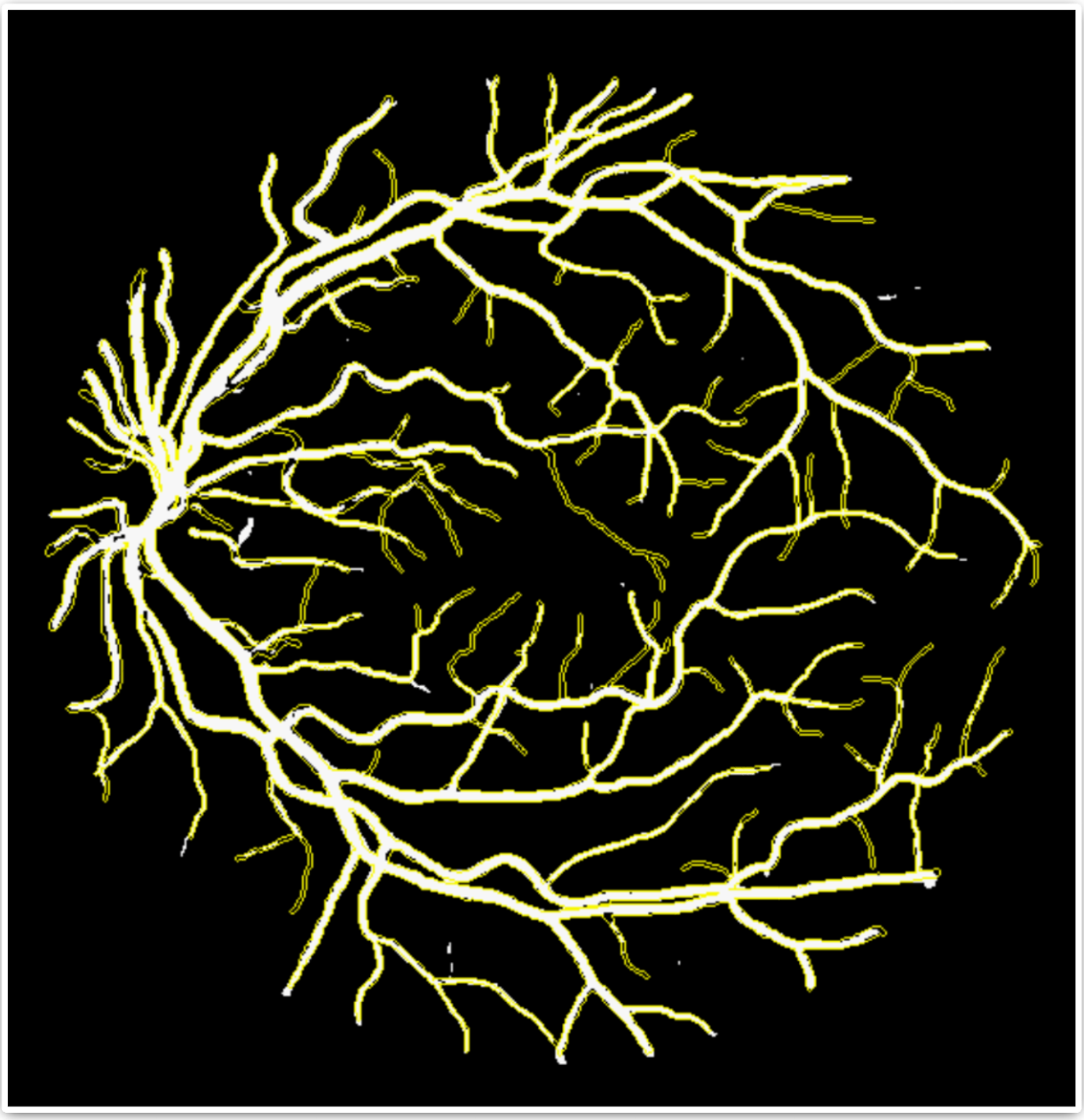


***Figura 43*** *De blanco se observa la segmentación hecha por la red neuronal y el contorno amarillo marca el borde de la ground truth.*

| Métricas | Propuesta | Mejores resultados | Publicación |
|---|---|---|---|
| Área bajo la curva Precision-Recall | 0.8804 | - | - |
| Área bajo la curva ROC | 0.965 | 0.965 | 0.972 |
| Precisión | 0.9676 | 0.9759 | 0.9495 |
| | | | |
| Parámetros | 463,761 | - | - |
| Tiempo [h] | 55 | - | - |

La tabla muestra los resultados de la red neuronal propuesta en este trabajo comparada con los mejores resultados obtenidos según [4] que han utilizado diferentes métodos y con los resultados obtenidos por la red neuronal convolucional presentada en [4]

En la tabla se muestra que los resultados son muy buenos, en comparación con los mejores resultados obtenidos según [4] y los resultados propiamente de [4], el desempeño es muy alto con tan solo 500 iteraciones y un tiempo de entrenamiento de 55 horas, el costo computacional es muy bajo comparado con los resultados obtenidos, además de no utilizar preprocesamiento, simplemente utilizar la imagen bruta de entrada, a diferencia de lo realizado en [4] que a través de una técnica llamada data augmentation, le hacen pequeñas modificaciones como escalar, rotar, voltear y variación de colores a las imágenes que se tienen para tener más datos de entrenamiento.

El tiempo de entrenamiento de la base de datos DRIVE, es mayor que la base de datos de imágenes placentarias debido al tamaño de las imágenes de fondo de ojo de DRIVE, son de tamaño 565 x 584, se usaron todas las imágenes para prueba y entrenamiento de nuevo con el método Cross-validation, 40 imágenes que, aunque menor número que las imágenes placentarias, tienen mejor calidad las imágenes de ground truth.

## VI.I Interfaz gráfica

Se programó una interfaz en Python con Tkinter [32] para mostrar gráficamente el funcionamiento de la red neuronal convolucional, usando los pesos entrenados previamente por el conjunto de imágenes de la base de datos para predecir la segmentación de una nueva imagen, las aplicaciones comunes que hoy en día se usan como por ejemplo el reconocimiento facial que algunos celulares tienen para desbloquear el teléfono, funcionan de esa forma.

El funcionamiento de la interfaz es muy sencillo, al oprimir el botón Imagen, se carga una imagen y se muestra en la ventana principal, si se oprime el botón CNN, se segmenta la imagen cargada, tarda algunos segundos en hacer la operación por la cantidad de recursos que necesita para hacer todas las operaciones de la red neuronal convolucional profunda, al terminar se muestra el botón Segmentada y si se oprime se muestra la predicción hecha por la red neuronal, al mostrarla aparecen dos nuevos botones, binarizar y esqueletizar; binarizar muestra la imagen solamente en blanco y negro, a diferencia de la predicción de la red que cuenta con algunos grises y el botón esqueletizar disminuye el grosor de las venas y arterias hasta que el grosor total es de un pixel para que solamente se analicen los brazos y la forma de los mismos.

Se puede volver a oprimir el botón imagen para cargar una imagen diferente y realizar una nueva predicción, en este caso la red neuronal no cuenta con información de la imagen etiquetada a mano, solamente necesita una imagen placentaria inyectada con colorante y que la computadora en la que se utilice el programa cuente con Python y las versiones de Keras utilizadas. Después de mostrar la imagen segmentada se puede pasar entre la imagen binarizada, la esqueletizada y la segmentada para poder compararlas.

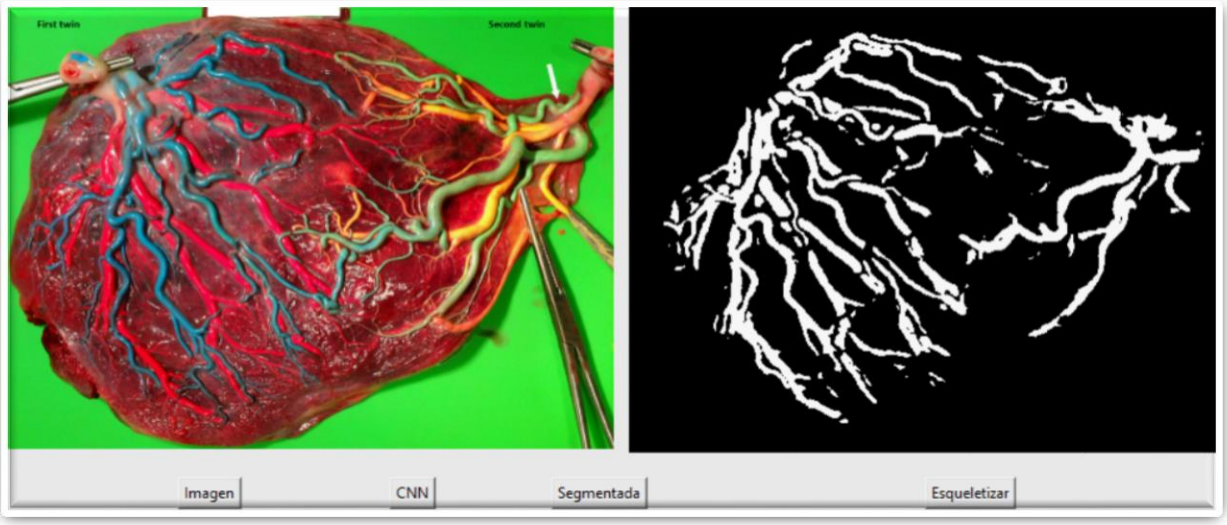



*Figura 44 Interfaz programada para demostrar funcionamiento*

# VII. Conclusiones y trabajo futuro

En el trabajo se presentó una base de datos de imágenes placentarias y se propone una red neuronal para su segmentación semántica en dos clases diferentes, debido a la calidad de las imágenes de la base de datos propuesta, no se obtienen números de desempeño reales, a pesar de que visualmente la predicción de la red es acertada.

Al probar con diferentes bases de datos se obtienen parámetros reales del desempeño de la red neuronal convolucional y se puede comparar con el trabajo hecho por otros, en el que con pocos parámetros se tiene una red convolucional robusta, capaz de segmentar con calidad incluso con pocas imágenes sin necesidad de hacer preprocesamiento.

Parte del trabajo futuro para la tarea de segmentación semántica de imágenes placentarias es mejorar la base de datos, etiquetando las imágenes manualmente con mayor calidad para obtener métricas precisas y poder proponer cambios que mejoren el rendimiento específicamente para las imágenes placentarias y ayudar al tratamiento del síndrome gemelo-gemelo.

También sería ideal conseguir más imágenes que ayuden a la red gracias a un mayor número de datos, buscar colaborar con hospitales o centros de investigación expertos en la materia, para poder enriquecer el trabajo y generar una propuesta con mejores resultados.

Probar técnicas como transfer learning también puede incrementar significativamente la calidad de segmentación, esta técnica consiste en entrenar la red neuronal en una base de datos muy grande, de millones de imágenes y usar los parámetros que se obtuvieron de ese entrenamiento, agregando tal vez una o dos capas extra que se entrenarán sobre las imágenes del problema en específico, en este caso las imágenes placentarias. Gracias a los parámetros obtenidos con la gran base de datos, esta técnica a resultado en mejor desempeño para tareas con muy pocas imágenes disponibles.

El uso de diferentes distribuciones de imágenes es otra forma que podría mejorar los números de la segmentación semántica, por ejemplo, se podrían combinar las bases de datos de fondo de ojo, con las imágenes placentarias y a pesar de que vienen de diferentes distribuciones, al ser parecidas y tener mucho mayor número de imágenes por utilizar todas las imágenes para entrenamiento y prueba podría ser una opción más para mejorar la tarea de segmentación.

En cuanto al objetivo general que es ayudar al tratamiento del síndrome gemelo-gemelo, el siguiente paso sería trabajar con la red neuronal para que sea capaz de distinguir venas y arterias de un bebé de venas y arterias del otro bebé y pintar cada una de las clases de color diferente, en este caso, el problema constaría con 3 clases, venas y arterias del primer bebé, venas y arterias del segundo bebé y lo demás que se presente en la imagen.

Por último, es necesario buscar que la red neuronal convolucional sea capaz de segmentar las diferentes clases en tiempo real, segmentando de un video para que al momento de la cirugía se le presente al doctor gráficamente las uniones entre bebés para poder coagularlas. El mayor desafío será realizar esta tarea con las imágenes fetoscópicas, que son de mucho menor tamaño y no se tiene un panorama completo del interior de la madre, por lo que se deberá buscar la forma de hacer que la red se sobreponga a esta complicación.

El síndrome de transfusión gemelo-gemelo es una enfermedad muy específica puesto que solo se da en un porcentaje muy bajo de los embarazos, pero con este trabajo se busca iniciar la investigación de visión computarizada en esta área, dando a conocer el caso de estudio y presentando bases sólidas que faciliten el trabajo a los interesados al ya contar con una base de datos que aunque puede mejorar, es aceptable y proponiendo una metodología para iniciarse en la tarea de segmentar semánticamente las imágenes placentarias.

# VIII. Bibliografía


[1] B. v. G. R. M. S. Hayit Greenspan, «Deep Learning in Medical Imaging: Overview and Future Promise of an Exciting New Technique,» *Guest editor,* may 2016.

[2] A. K. R. C. S. M. I. Vijay Badrinarayanan, SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation, arXiv:1511.00561v3 [cs.CV] 10 Oct 2016.

[3] S. H. B. H. Hyeonwoo Noh, «Learning Deconvolution Network for Semantic Segmentation,» *Department of Computer Science and Engineeringm, POSTECH, Korea.*

[4] P. L. a. K. Krawiec, «Segmenting Retinal Blood Vessels With Deep Neural Networks,» *IEEE TRANSACTIONS ON MEDICAL IMAGING, VOL. 35, NO. 11, NOVEMBER 2016.*

[5] P. F. a. T. B. Olaf Ronnenberger, «U-Net: Convolutional Networks for Biomedical Image Segmentiation,» *Computer Science Department and BIOSS Centre for Biological Signalling Studies,* Univertity of Freiburg, Germany.

[6] J. D. H. S. J. K. S. S. S. M. Z. H. A. K. A. K. M. B. A. C. B. a. L. F.-F. (. =. e. c. Olga Russakovsky*, ImageNet Large Scale Visual Recognition Challeng, IJCV, 2015.

[7] «ISBI Challenge: Segmentation of neuronal structures in EM stacks,» [En línea]. Available: http://brainiac2.mit.edu/isbi_challenge/.

[8] M. A. M. N. M. V. B. v. G. ". b. v. s. i. c. i. o. t. r. I. T. o. M. I. 2. v. 2. p. 5.-5. J.J. Staal, «DRIVE: Digital Retinal Images for Vessel Extraction,» [En línea]. Available: https://www.isi.uu.nl/Research/Databases/DRIVE/.

[9] V. K. a. M. G. ". B. V. i. R. I. b. P.-w. T. P. o. a. M. F. R. I. T. o. M. I. ,. v. 1. n. 3. p. 2.-2. M. 2. A. Hoover, «STructured Analysis of the Retina,» [En línea]. Available: http://cecas.clemson.edu/~ahoover/stare/.

[10] M. J. H. M. K. H. M. a. K. N. Yves Ville, *Preliminary Experience with Endoscopic Laser Surgery for Severe Twin–Twin Transfusion Syndrome,* M.D.: DOI: 10.1056/NEJM199501263320404, January 26, 1995.

[11] T. C. Hospital, «Selective Fetoscopic Laser Photocoagulation (SFLP),» 2018. [En línea]. Available: women.texaschildrens.org/program/texas-children's-fetal-center/fetoscopic-laser-photocoagularion-ttts.

[12] Y. H. K. G. A. S. N. H. J. a. N. K. Ville, Endoscopic laser coagulation in the management of severe twin-to-twin transfusion syndrome, BJOG: An International Journal of Obstetrics & Gynaecology: 105: 446–453. doi:1, 1998.

[13] S. F. M. J. Lopriore E, Residual anastomoses in twin-to-twin transfusion syndrome treated with selective fetoscopic laser surgery: localization, size, and consequences, 2009.

[14] Stanford, «CS231n Convolutional Neural Networks for Visual Recognition,» [En línea]. Available: http://cs231n.github.io/classification/.

[15] Z. W. Waseem Rawat, Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review, Massachusetts Institute of Technology: Neural Computation 29, 2352–2449 doi:10.1162/NECO_a_00990, 2017.

[16] U. Karn, «An Intuitive Explanation of Convolutional Neural Networks,» August 11, 2016. [En línea]. Available: https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/.

[17] Y. B. Xavier Glorot, «Understanding the difficulty of training deep feedforward neural networks,» *DIRO, Université de Montréal, Montréal, Québec, Canada.*

[18] X. Z. S. R. J. S. Kaiming He, «Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification,» *Microsoft Research,* p. arXiv:1502.01852v1 [cs.CV] 6 Feb 2015.

[19] S. Ruder, «An overview of gradient descent optimization algorithms,» *Insight Centre for Data Analytics, NUI Galway,* p. arXiv:1609.04747v2 [cs.LG] 15 Jun 2017, Aylien Ltd., Dublin.

[20] G. H. A. K. I. S. R. S. Nitish Srivastava, «Dropout: A Simple Way to Prevent Neural Networks from Overfitting,» *Department of Computer Science, University of Toronto.*

[21] S. F. J.-M. C. M. H. R. B. K. K. S. L. A. C. S. ELIN FARNELL, «A SHAPE-CONTEXT MODEL FOR MATCHING PLACENTAL CHORIONIC SURFACE VASCULAR NETWORKS,» *doi:10.5566/ias.1708.*

[22] G. B. a. J. Fauqueur, «InteractLabeler 1.2.1,» {gjb47,jf330}@cam.ac.uk, Cambridge University, february 2007. [En línea]. Available: http://mi.eng.cam.ac.uk/projects/cvgroup/software/index.html.

[23] R. Kohavi, «A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection,» Vols. %1 de %2Stanford, CA. 94605, p. Stanford University, Computer Science Department.

[24] A. P. a. J. A. L. M. I. Juan Diego Rodríguez, «Sensitivity Analysis of k-Fold Cross Validation in Prediction Error Estimation,» *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. 32, NO. 3,,* March 2010.

[25] Berkeley, «The official blog of Machine Learning @ Berkeley,» [En línea]. Available: https://ml.berkeley.edu/blog/2017/07/13/tutorial-4/.

[26] A. P. Bradley, «The Use of the Area Under the ROC Curve in the Evaluation of Machine Learning Algorithms,» 1995.

[27] M. G. Jesse Davis, «The Relationship Between Precision-Recall and ROC Curves,» *Department of Computer Sciences and Department of Biostatistics and Medical Informatics, Universitñy of Wisconsin-Madison.*

[28] s. learn, «Precision-Recall,» [En línea]. Available: http://scikit-learn.org/stable/auto_examples/model_selection/plot_precision_recall.html.

[29] J. S. E. N. a. S. W. Suphakit Niwarranakul, «Using of Jaccard Coefficient for Keyword Similarity,» *Proceedings of the International MultiConference of Engineers and Computer Scientists 2013 Vol 1, IMECS 2013, Hong Kong.*

[30] s. learn, «sklearn.metrics.matthews_corrcoef,» [En línea]. Available: http://scikit-learn.org/stable/modules/generated/sklearn.metrics.matthews_corrcoef.html.

[31] «Keras: The Python Deep Learning library,» [En línea]. Available: https://keras.io/.

[32] T. Python. [En línea]. Available: https://docs.python.org/2/library/tkinter.html.