Retina VesselNet
Giovanni Gamaliel López Padilla

Índice

1. Introducción

La caracterización de Vessel juega un importante rol en los diagnósticos médicos. Es por ello, que las tareas como la caracterización de su anchor, color, reflectividad, tortuosidad y ramas abnormales son necesarias. El conocimiento de la localización de las lineas de Vessel es de ayuda para la vista de una retinopatía diabética, ya que reduce el número de falsos positivos en la detección de microaneurismas.[1–3] Cuando el número de ramas o de imágenes es grande, la tarea manual de caracterizar el deliniado de las lineas de Vessel se vuelve un trabajo tedioso y complicado. Con la ayuda de algoritmos de segmentación se ha logrado subsanar la tarea de localizar las lineas de Vessel en una gran cantidad de imágenes. El estudio de esta tarea aumento al incio del milenio, planteando algoritmos basados en detección de bordes[4] hasta hoy en día que el estado del arte se encuentra en el uso de aprendizaje profundo[5] para clasificar y obtener diagnósticos de las lineas de Vessel.

2. Materiales y métodos

2.1. Red neuronal Convolucional

Las redes neuronales convolucionales tienen sus bases en el Neocognitron introducido por Kunihiko Fukushima.[6] Este modelo fue mejorado por Yann LeCun[7] al emplear un método de aprendizaje basado en la propagación hacia atrás para entrenar el sistema correctamente. En el año 2012, Dan Ciresan refino e implemento el modelo para unidades de procesamiento gráfico (GPU) consiguiendo mejores resultados.[8]

Las redes neuronales convolucionales consisten en múltiples capas de filtros convolucionales de una o más dimensiones. Por lo general, después del proceso de una capa de filtro se añade una capa que aplica una función que realizar un mapeo causal no-lineal. En la figura 1 se muestra la arquitectura de una red convolucional.

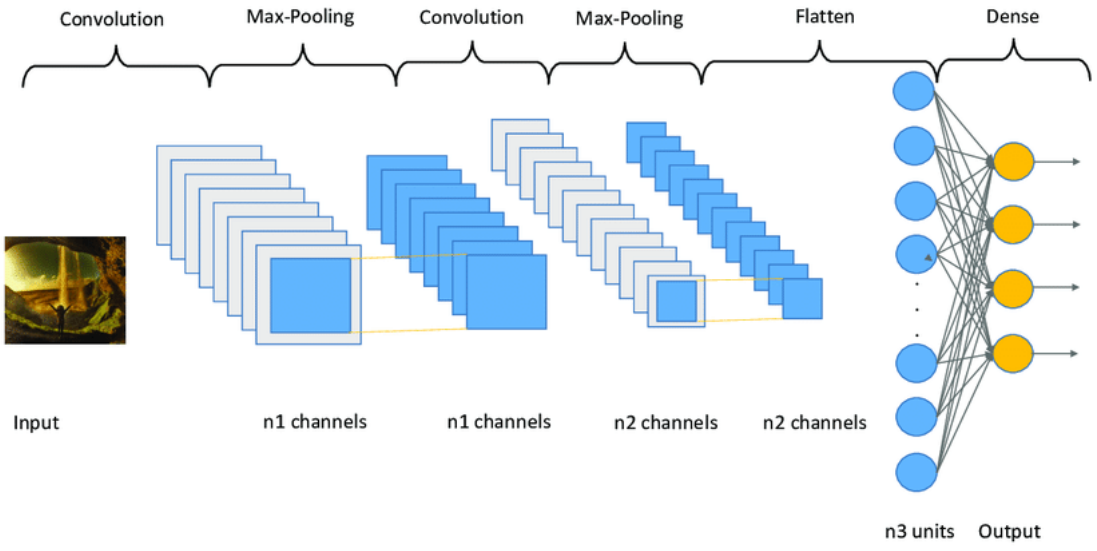


Figura 1: Representación visual de una red convolucional.[9]

2.1.1. Capa Max pooling

La capas que aplican la función de Max pooling realizan una operacion que calcula el valor máximo de una submatriz. Este valor máximo es guardado en una matriz con menor dimensión que la original. En la figura 2 se muestra un ejemplo de como se reduce una matriz de dimensión $4x4$ a una de $2x2$.

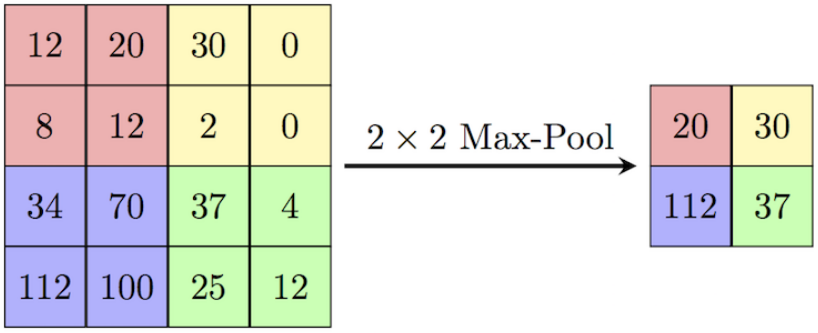


Figura 2: Representación vidual de una capa con la operacion max pooling.

2.2. Modelo U-Net

En la figura 3 se ilustra la arquitectura de una red del tipo U-Net. Este modelo consiste de dos partes, un encoder (lado izquierdo) y un decoder (lado derecho). El encoder tiene una arquitectura típica de una red convolucional en la que cada paso se aplica una convolución de $3x3$ seguida de una capa con la función rectificador (ReLU) y otra capa con un max pooling de 2 dimensiones para una reducción de dimensión. En la etapa del decoder se aplica un aumento de dimensión seguida de una capa de convolución de 2x2. Al final se concadena el resultado de este proceso con el mapa reducido. En seguida se aplica una serie de dos capas convolucionales de $3x3$ y se aplica la función ReLU. Al final del proceso se aplica una capa de convolución de $1x1$ que es usada para mapear cada componente a un número de canales. En total, la red tiene 23 capas convolucionales.

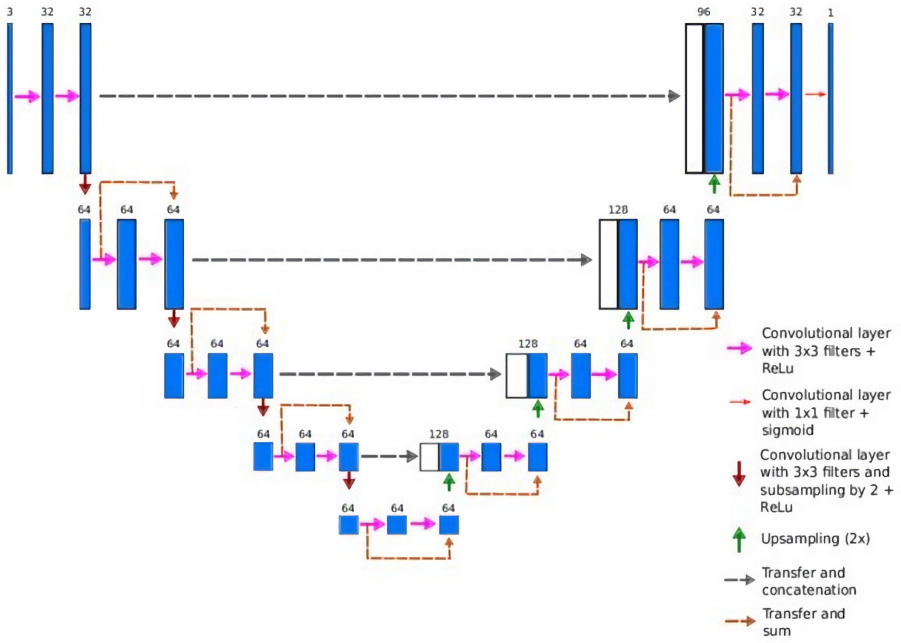


Figura 3: Arquitectura de una red del tipo U-Net.

2.3. Arquitectura del modelo

2.3.1. MobileNet v2

2.3.1.1 Depthwise Separable Convolutions

Depthwise Separable Convolutions es una herramienta para construir de manera eficiente arquitecturas para redes neuronales.[10–12] La idea de la herramienta es remplazar una capa convolucional completa con una versión factorizada en dos capas. La primer capa es llamada *depthwise convolution*, la cual realiza un filtro lihero aplicando una sola convolución en un canal. La segunda cada es una convolución de $1x2$ llamada *pointwise convolution*. Esta capa es la responsable de crear nuevos elementos usando combinaciones lineales de los canales de entrada.

2.3.1.2 Linear Bottlenecks

Consideremos una red neuronal con n capas, donde cada capa contiene la dimensión $h_i \times w_i \times d_i$. Para cada conjunto de imágenes dada, tenemos un conjunto de capas de activación que forman un *manifold of interest*. Este conjunto de capas pueden ser reducidas a un espacio de baja dimensionalidad. Este espacio debe de contener las siguientes propiedades:

- Si el espacio contiene un volumen después de una transformación realizada por la función ReLU, entonces este corresponde a una transformación lineal.
- La función RelU preserva la información del espacio, solo si el espacio esta contenido en un espacio de baja dimensionalidad del espacio de entrada.

Estas propiedades nos dan una pista empirica para la optimización de las arquitecturas de redes. Asumiendo que el espacio de interés corresponde a un espacio de baja dimensional, entonces se puede capturar capas *bottleneck* en un bloque de capas convolucionales. En la figura 4 se muestra el ejemplo de una capa bottleneck.

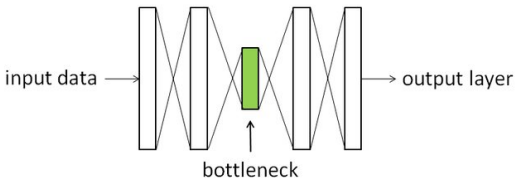


Figura 4: Ejemplo de una capa bottleneck.

2.3.1.3 Inverted residuals

Los bloques conpuestos de capas bottleneck se asemejan a un bloque de capas residuales donde cada bloque contiene su entrada seguida de una serie de capas bottleneck seguidas de euna expansión.[13] Como una capa de expansión puede verse como una serie de capas bottleneck acompañadas de una transformación no lienal, se usaron directamente las capas bottleneck. En la figura 5 puede verse una representación de esta decisión.

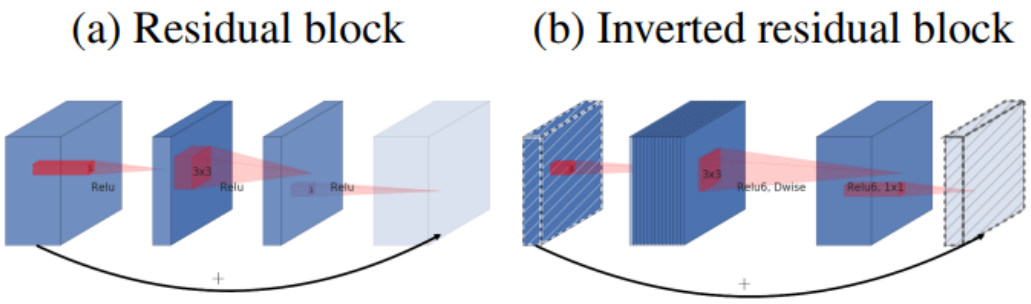


Figura 5: Representación visual de las capas residuales y las capas bottleneck.[14]

2.3.1.4 Arquitectura

La arquitectura de MobileNetV2 contiene una capa convolucional de 32 filtros seguida de 19 capas residuales del tipo bottleneck. Se uso la función ReLU6 como una función no lineal por su robustes en una baja precisión numerica.[10] En la tabla 1 se encuentra descrita la arquitectura de MobileNetV2.

Entrada	Operador	t	c	n	s
$224^2 \times 3$	conv2d	-	32	1	2
$112^2 \times 32$	bottleneck	1	16	1	1
$112^2 \times 16$	bottleneck	6	24	2	2
$56^2 \times 24$	bottleneck	6	32	3	2
$28^2 \times 32$	bottleneck	6	64	4	2
$14^2 \times 64$	bottleneck	6	96	3	1
$14^2 \times 96$	bottleneck	6	160	3	2
$7^2 \times 160$	bottleneck	6	320	1	1
$7^2 \times 320$	conv2d 1×1	-	1280	1	1
$7^2 \times 1280$	avgpool 7×7	-	-	1	-
$1 \times 1 \times 1280$	convd2d 1×1	-	k	-	

Tabla 1: Cada linea describe a cada secuencia de uno o más capas identicas. Todas las capas en la misma secuencia tienen el mismo número de canales de salida (c). La primer capa de cada secuencia tiene s pasos y las demás un solo paso. Cada capa convolucional usa un kernel de $3x3$. El factor de expansión t siempre es aplicado como se muestra en la tabla 2.

Entrada	Operador	Salida
$h \times w \times k$	1×1 cond2d, ReLU6	$h \times w \times tk$
$h \times w \times tk$	3×3 dwise s=s, ReLU6	$\frac{h}{s} \times \frac{w}{s} \times tk$
$\frac{h}{s} \times \frac{w}{s} \times tk$	linear 1×1 cond2d	$h \times w \times k'$

Tabla 2: Capa residual del tipo bottleneck que transforma de k a k' canales con s pasos y un factor de expansión de t.

2.3.2. Pix2Pix

2.3.2.1 Redes Generativas Antagónicas

Las Redes Generativas Antagónicas (GAN's) es un modelo que se enfrenta contra un modelo discriminativo que aprende a determinar cuando una muestra es producida por un modelo o es de los datos.

2.3.2.2 Estructura de modelación

Los problemas de imagen a imagen por lo regular son tratados como una clasificación pixel a pixel. Para ello, cada pixel es tratado con una probabilidad independiente a su alrededor. Las GANs condicionales aprenden una estructura de perdida, la cual penaliza las diferencias entre la configuración final y la configuración objetivo.

2.3.2.3 Arquitectura

Para definir el problema de la traducción imagen a imagen, es definir una imagen de alta resolución que difiere en apariencia a la imagen de salida. Es por ello que la estructura en la entrada difiere a la de salida. Es por ello que se considero un generador que considera estas propiedades. Las soluciones propuestas con anterioridad es el uso de una red encoder-decoder. En estas redes la información atraviesa una serie de capas que reducen su dimensión hasta llegar a una capa del tipo bottleneck, en seguida la información pasa por un proceso de expansión.

Para muchos problemas de traducción de imagen es una gran idea realizar una conexión entre la entrada y la salida. Para implementar esta idea se añaden conexiones formando una red de tipo U-Net. Especificamente se añaden conexiones entre la capa i y la capa $n - i$. En la figura 6 se visualiza la arquitectura de pix2pix.

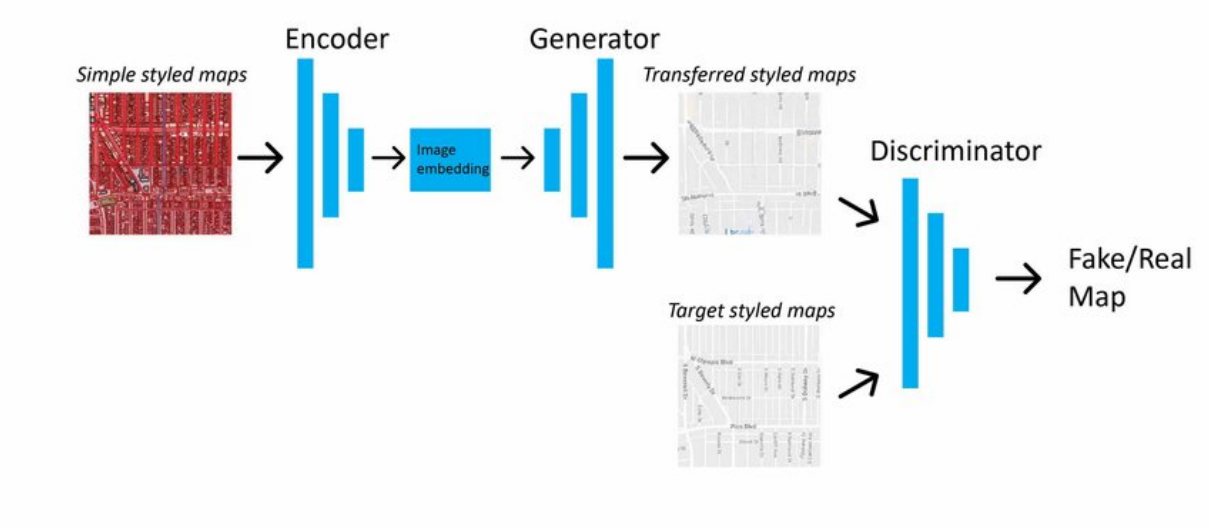


Figura 6: Arquitectura de pix2pix.[15]

2.4. Modelo propuesto

Se extrajeron las siguientes capas de la red MobileNet v2.

- block_1_expand_relu
- block_3_expand_relu
- block_6_expand_relu
- block_13_expand_relu
- block_16_project

Las cuales seran usadas para realizar la concadenación con las capas de expansión. De igual manera se extrajeron las siguientes capas del modelo pix2pix.

- pix2pix.upsample(512, 3)
- pix2pix.upsample(256, 3)
- pix2pix.upsample(128, 3)
- pix2pix.upsample(64, 3)

Estas capas conformaran las capas de expansión del modelo U-Net. El modelo en su totalidad se encuentra ilustrado en la figura 7.

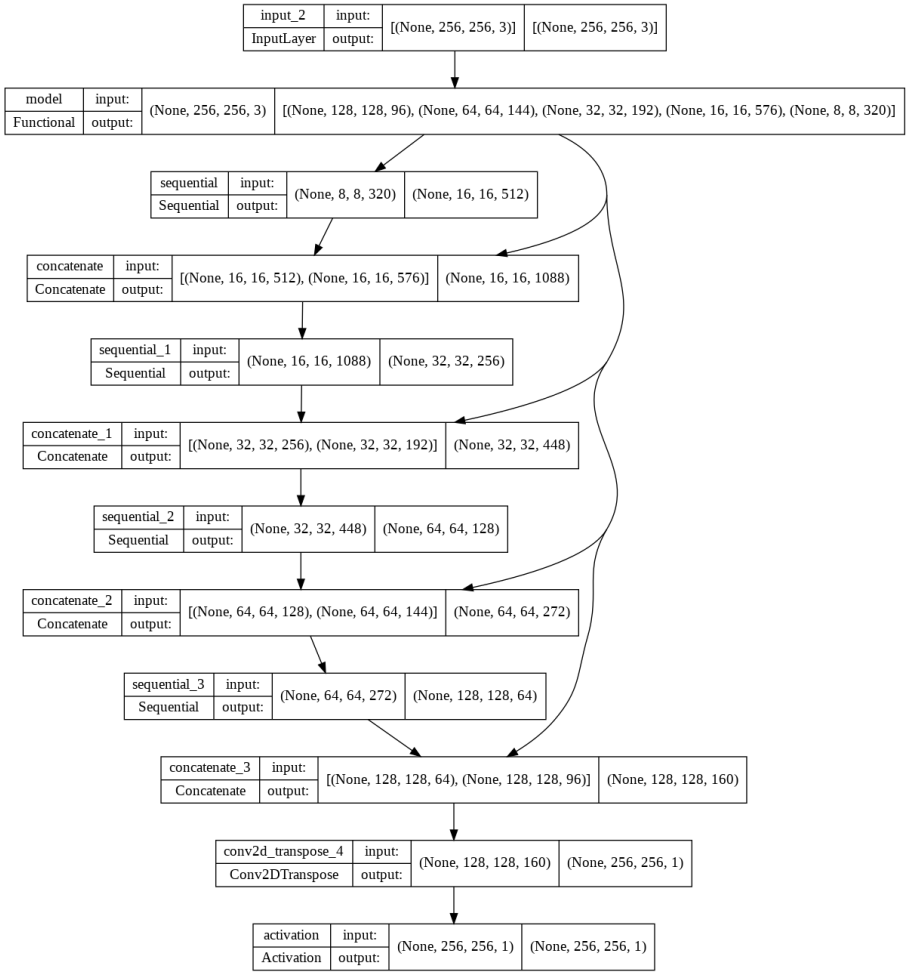


Figura 7: Modelo usando capas de la red MobileNet v2 y pix2pix.

2.5. Tipos de entrenamiento

El modelo fue entrenado de diferentes estrategias. Eso para observar los tiempos de entrenamiento y los resultados que se obtienen con cada modelo.

2.5.1. Conexion directa

En este tipo de entrenamiento se congelaron los pesos de las capas extraidas de MobileNet v2.

2.5.2. Fine tunning

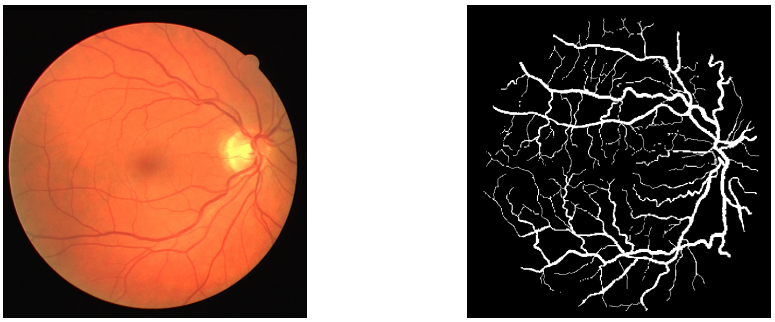
En este tipo de entrenamiento se descongelo unicamente la capa block_16_project del modelo MobileNet v2. Las demás se dejaron congeladas.

2.5.3. Full tunning

En este tipo de entrenamiento se descongelaron todas las capas extraidas de MobileNet v2.

2.6. Vessel Dataset

El conjunto de datos utiizado se compone de 1280 imágenes (1008 para entrenamiento, 252 para validación y 20 para pruebas). Las imágenes fueron obtenidas del siguiente repositorio VesselNet. El conjunto de imágenes usaron fueron unicamente las contenidas en la carpeta Drive_1000. En la figura 8 se muestra un ejemplo de la imágen de entrada y la salida esperada.

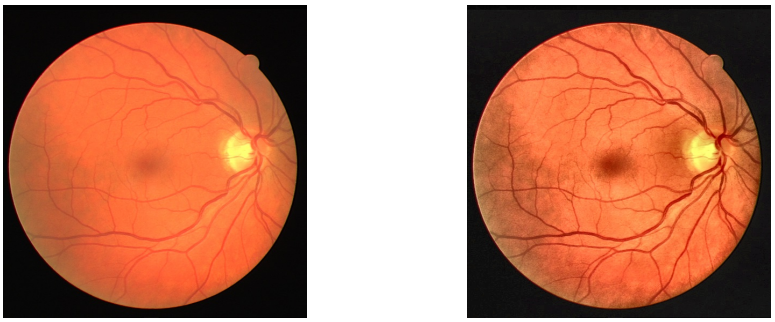


(a) Imágen de un ojo humano. **(b)** Vasos capilares del ojo.

Figura 8: Conjunto de imágenes de la base de datos de VesselNet.

2.6.1. Filtro de alto contraste

Se realizo un prepocesamiento a las imágenes para lograr difentes resultados. Uno de estos preprocesamientos fue el uso de un filtro de alto contraste para resaltar los vasos capilares en las imágenes de entrada. Esto fue implementado haciendo uso de la libreria de opencv de python. En la figura 9 se muestra el cambio visual que se obtiene al aplicar el filtro.

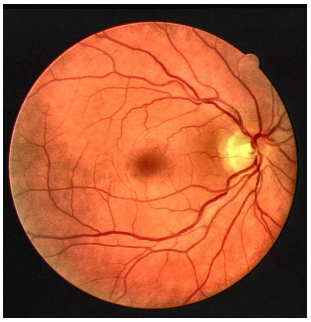


(a) Imágen original. **(b)** Imágen con alto contraste.

Figura 9: Aplicación del alto contraste a la base de datos VesselNet.

2.6.2. RGB a escala de grises

A partir de las imágenes con alto contraste se transformaron estas imágenes para tenerlas en un espacio de grises. En la figura se muestra un ejemplo del resultado del realizar esta transforación. opencv de python. En la figura 10 se muestra el cambio visual que se obtiene al aplicar el filtro.



(a) Imágen con alto contraste.

(b) Imágen en escala de grises.

Figura 10: Transformación al espacio de grises.

2.6.3. Aumentación de imágenes

Se realizaron diversas transformaciones para realizar una aumetación de los datos en el proceso de entrenamiento. Las transformaciones realizadas son las siguientes:

- Rotaciones de no más de 90 grados.
- Desplazamientos de 0.3.
- Ampliación de la imegn de 0.3.
- Reflexiones de manera horizontal y vertical.

3. Resultados

3.1. Imagenes originales

En la figura 11 se observan los resultados obtenidos por el modelo con las diferentes maneras de entrenamiento.

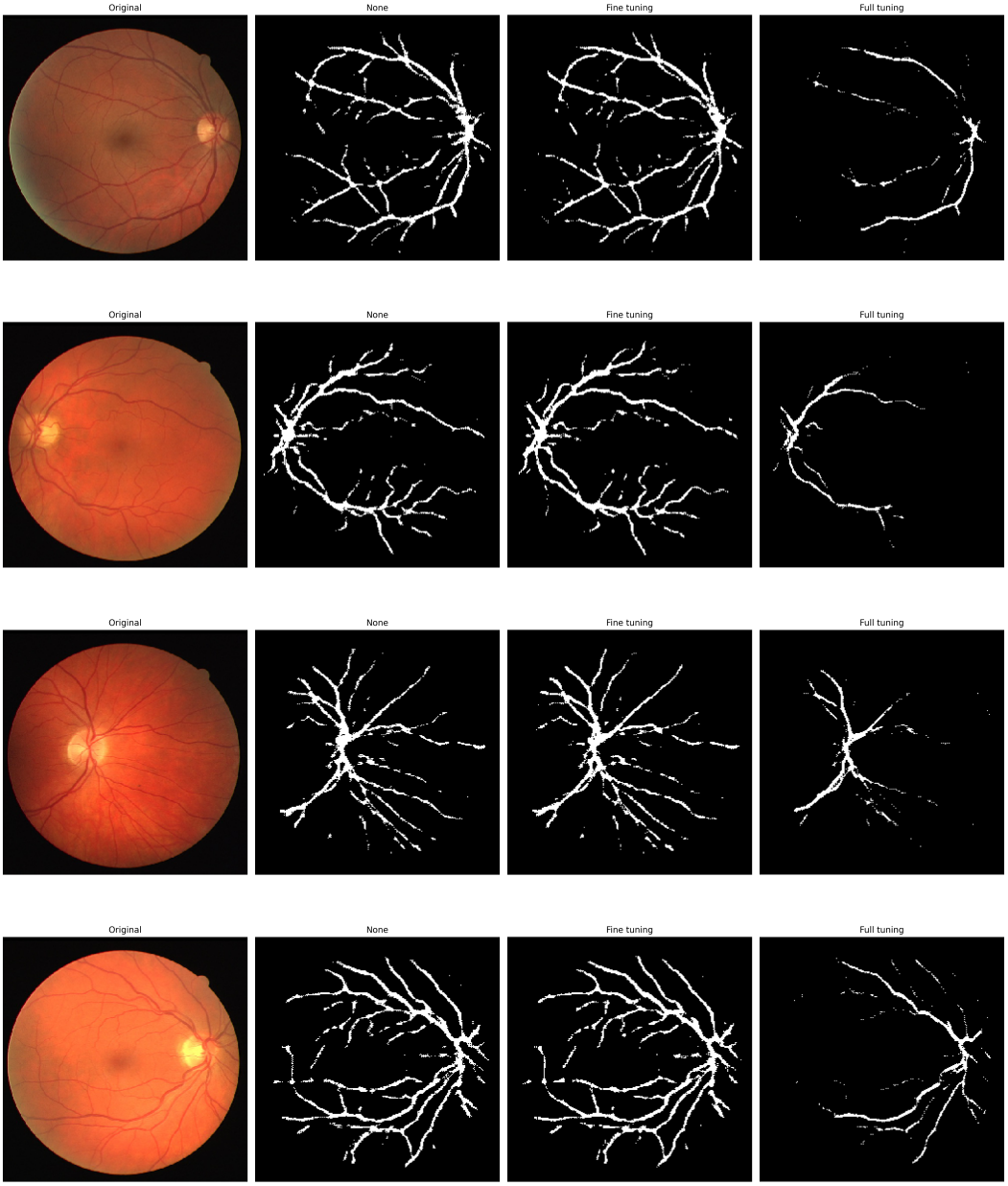


Figura 11: Resultados para una muestra del conjunto de prueba con los diferentes modelos usando como entrada las imágenes originales.

3.2. Imágenes con alto contraste

En la figura 12 se observan los resultados obtenidos por el modelo con las diferentes maneras de entrenamiento.

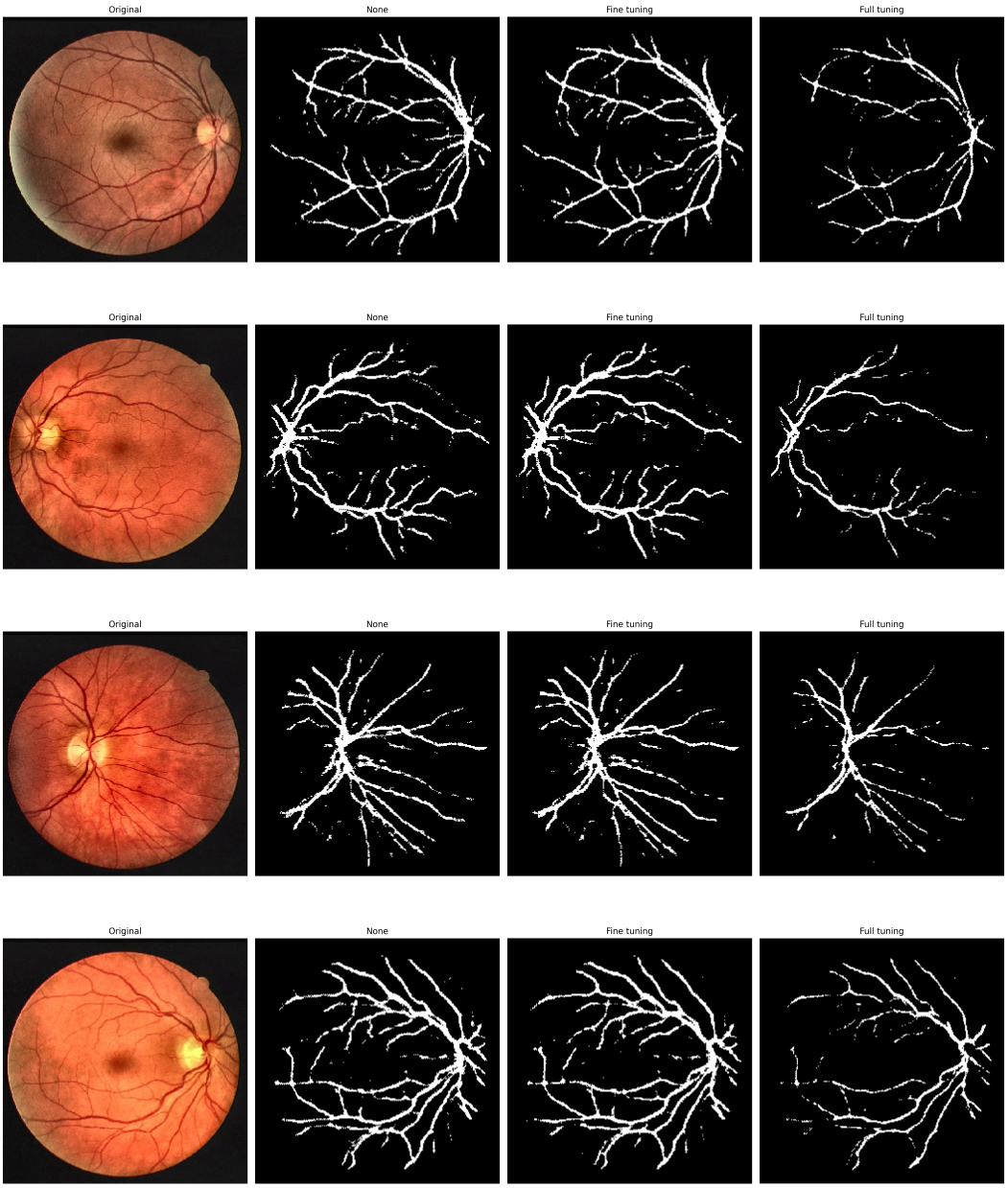


Figura 12: Resultados para una muestra del conjunto de prueba con los diferentes modelos usando como entrada las imágenes con el filtro de alto contraste.

3.3. Imágenes en escala de grises

En la figura 10 se observan los resultados obtenidos por el modelo con las diferentes maneras de entrenamiento.

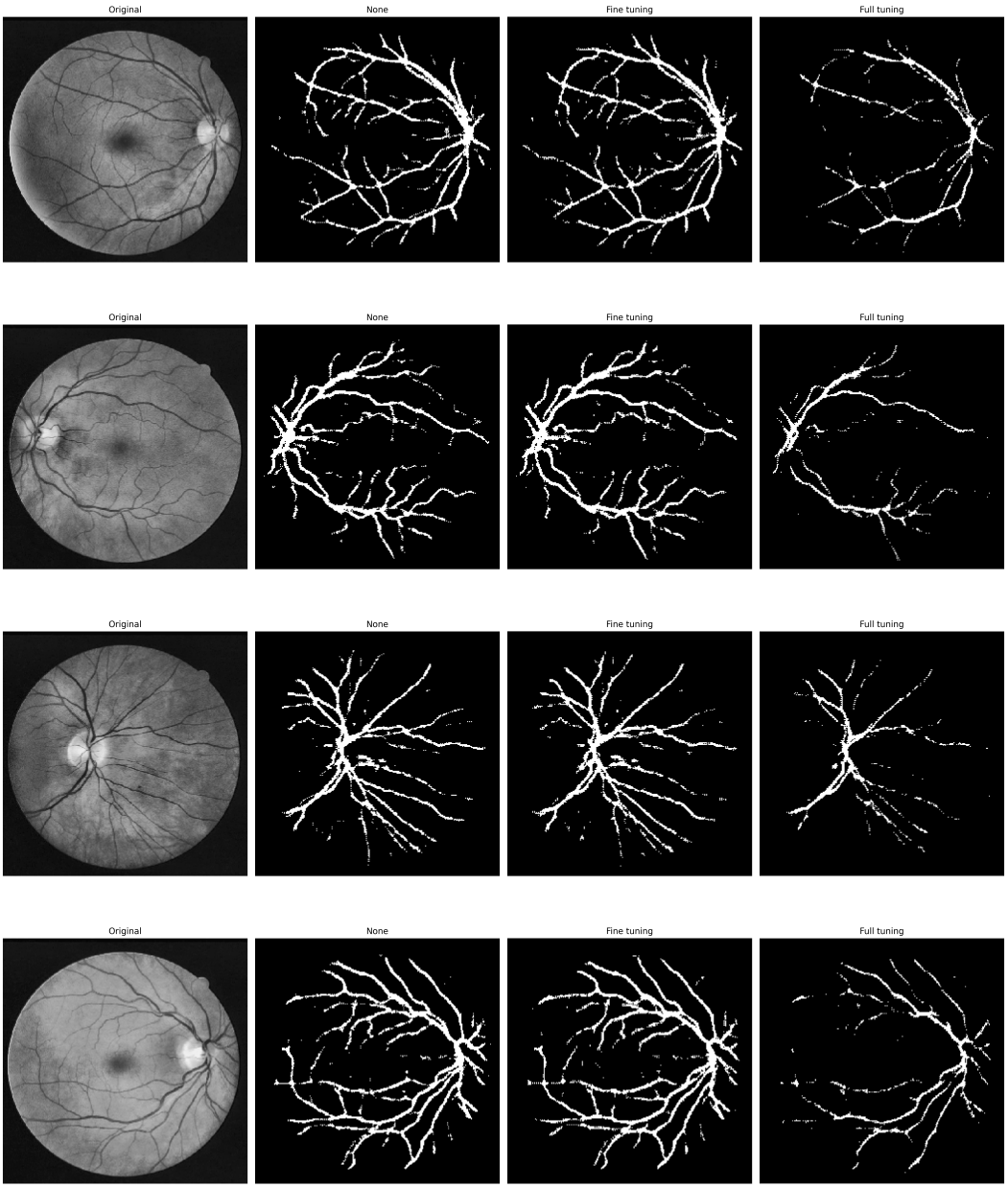


Figura 13: Resultados para una muestra del conjunto de prueba con los diferentes modelos usando como entrada las imágenes en escala de grises.

4. Conclusiones

Los resultados obtenidos por los modelos con los diferentes modos de entrenamiento realizan una buena tarea al segmentar los vasos capilares de imágenes de ojos humanos. La arquitectura propuesta es eficiente en espacio de memoria y tiempo de entrenamiento, esto debido a que con 15 epocas, el modelo tardaba entre 10 a 15 minutos de entrenamiento. Observando las figruas 11, 12 y 13 se muestra que los métodos de fine tunning y conexión dircta obtienen buenos resultados en comparación el método de full tunning. Los mejores resultados se obtuvieron cuando la imágen de entrada fue prepocesada con un filtro de alto contraste. Las predicciones obtenidas por los modelos donde la entrada fue una imágen en escala de grises son semejantes a las obtenidas por las imágenes con alto contraste, sin embargo estas presentan discontinuidades

no existentes. La ventaja que ofrecen estas predicciones es que el tiempo y memoria usada durante el entrenamiento fue menor, por lo que puede ser una alternativa cuando se tenga estos escenarios.

4.1. Código

El código de los modelos y predicciones, las bases de datos y este documento se encuentran en el siguiente repositorio GitHub.

5. Referencias

[1] Spencer T, Olson JA, McHardy KC, Sharp PF, Forrester JV. An Image-Processing Strategy for the Segmentation and Quantification of Microaneurysms in Fluorescein Angiograms of the Ocular Fundus. Computers and Biomedical Research. 1996 aug;29(4):284-302. Available from: https://doi.org/10.1006%2Fcbmr.1996.0021.

[2] Frame AJ, Undrill PE, Cree MJ, Olson JA, McHardy KC, Sharp PF, et al. A comparison of computer based classification methods applied to the detection of microaneurysms in ophthalmic fluorescein angiograms. Computers in Biology and Medicine. 1998;28(3):225-38. Available from: https://www.sciencedirect.com/science/article/pii/S0010482598000110.

[3] Larsen M, Godt J, Larsen N, Lund-Andersen H, Sjølie AK, Agardh E, et al. Automated Detection of Fundus Photographic Red Lesions in Diabetic Retinopathy. Invest Ophthalmol Vis Sci Visual Science. 2003 feb;44(2):761. Available from: https://doi.org/10.1167%2Fiovs.02-0418.

[4] Staal J, Abramoff MD, Niemeijer M, Viergever MA, van Ginneken B. Ridge-based vessel segmentation in color images of the retina. IEEE Transactions on Medical Imaging. 2004;23(4):501-9.

[5] Elsharif AAEF, Abu-Naser SS. Retina Diseases Diagnosis Using Deep Learning. International Journal of Academic Engineering Research (IJAER). 2022;6(2):11-37.

[6] Fukushima K. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. Biol Cybernetics. 1980 apr;36(4):193-202. Available from: https://doi.org/10.1007/bf00344251.

[7] Lecun Y, Bottou L, Bengio Y, Haffner P. Gradient-based learning applied to document recognition. Proc IEEE. 1998;86(11):2278-324. Available from: https://doi.org/10.1109%2F5.726791.

[8] Cireşan DC, Meier U, Masci J, Gambardella LM, Schmidhuber J. Flexible, High Performance Convolutional Neural Networks for Image Classification. In: Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Two. IJCAI'11. AAAI Press; 2011. p. 1237-42.

[9] García-Ordás MT, Benítez-Andrades JA, García-Rodríguez I, Benavides C, Alaiz-Moretón H. Detecting Respiratory Pathologies Using Convolutional Neural Networks and Variational Autoencoders for Unbalancing Data. Sensors. 2020 feb;20(4):1214. Available from: https://doi.org/10.3390%2Fs20041214.

[10] Howard AG, Zhu M, Chen B, Kalenichenko D, Wang W, Weyand T, et al.. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. arXiv; 2017. Available from: https://arxiv.org/abs/1704.04861.

[11] Chollet F. Xception: Deep Learning with Depthwise Separable Convolutions. arXiv; 2016. Available from: https://arxiv.org/abs/1610.02357.

[12] Zhang X, Zhou X, Lin M, Sun J. ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices. arXiv; 2017. Available from: https://arxiv.org/abs/1707.01083.

[13] He K, Zhang X, Ren S, Sun J. Deep Residual Learning for Image Recognition; 2015.

[14] Sandler M, Howard A, Zhu M, Zhmoginov A, Chen LC. MobileNetV2: Inverted Residuals and Linear Bottlenecks. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. IEEE; 2018. Available from: https://doi.org/10.1109%2Fcvpr.2018.00474.

[15] Salimans T, Goodfellow I, Zaremba W, Cheung V, Radford A, Chen X. Improved Techniques for Training GANs. arXiv; 2016. Available from: https://arxiv.org/abs/1606.03498.