Segmentación de imágenes CT en pacientes con COVID-19.

Mario Horacio Garrido Czacki
Licenciatura en Ciencia de Datos
IIMAS, UNAM
mgczacki@gmail.com

Enrique David Guzmán Ramírez
Licenciatura en Ciencia de Datos
IIMAS, UNAM
david_guzmanr@ciencias.unam.mx

Alejandro Hernández Rodríguez
Licenciatura en Ciencia de Datos
IIMAS, UNAM
roher1727@gmail.com

Resumen

Utilizando tomografías computarizadas axiales de pacientes que padecen COVID-19 se entrenó una red neuronal para segmentar automáticamente tres lesiones comunmentente presentadas en pacientes con la enfermedad: vidrio, consolidación y pleura.

Index Terms

Imagen, segmentación, convolucional, red, neuronal, Python, MobileNetV2, U-net.

I. Introducción

Para poder realizar la predicción de segmentos, se utilizó el conjunto de imagenes presentes en [1]. Este conjunto de imagenes constaba de dos conjuntos de datos distintos, tr_im y tr_mask. En el conjunto tr_im se tienen imagenes de resolución 512×512 en escala de gris, que es la tomografía como tal. El conjunto tr_mask son imagenes de resolución 512×512 en donde están segmentadas, manualmente, las clases de cada máscara. Las clases son las siguientes

- Opacidad en vidrio esmerilado: Es un área nebulosa de aumento de la atenuación del pulmón con marcas bronquiales y vasculares preservadas. [10]
- Consolidación: Es un proceso de llenado alveolar que reemplaza el aire dentro de los espacios aéreos afectados, aumentando la atenuación pulmonar. [11]
- Derrame Pleural: Es el exceso de fluidos entre las capas de pleura, fuera de los pulmones. [12]

Es de interés encontrar estos segmentos en una CT por que los especialistas consideran que pueden alertar cuando el COVID-19 esta presente en el paciente. Entonces encontrar un método que puede detectar esta segmentación de manera automatizada y confiable utilizando nuevas tecnologías es una necesidad.
Para solucionar este problema proponemos el uso de algoritmos de aprendizaje profundo, que es una rama de la Inteligencia Artificial basada en las Redes Neuronales Artificiales, que en los últimos años ha ganado popularidad y se ha usado con mayor frecuencia en el sector salud debido a los buenos resultados que ha tenido para el diagnóstico de enfermedades. En particular nosotros utilizamos una arquitectura U-Net, que fue diseñada para segmentar con relativamente pocas cantidades de datos y desarrollada para la segmentación de imagenes médicas.

II. Marco Teórico

II-A. Red Neuronal Artificial

Es un algoritmo de aprendizaje automático inspirado en el funcionamiento de las redes neuronales biológicas, estadísticamente se puede definir como una regresión de regresiones no lineales. En la figura 1 se observa un diagrama de una neurona piramidal, que se encuentran en el hipocampo o amígdala. Las partes más importantes para el modelo matemático son las dendritas y el axon. Las dendritas son las entradas de voltaje de otras neuronas y el axon es el voltaje que dispara la neurona.

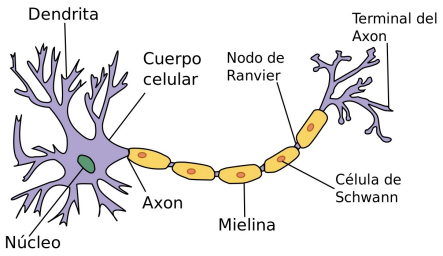


Figura 1: Representación gráfica de una neurona. Tomada de [4]

El funcionamiento es sencillo. Dentro del cuerpo de la neurona tiene una polaridad negativa, cuando recibe carga positiva a través de las dendritas de parte de otras neuronas y supera el umbral de estimulación, el cuerpo cambia de polaridad y descarga el voltaje a través del axón hacía las dendritas de otras neuronas, como se observa en la figura 2.

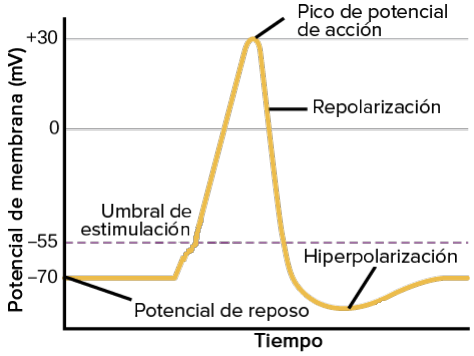


Figura 2: Umbral de estimulación.

Para simular este comportamiento, se desarrolló el modelo perceptrón, donde en lugar de dendritas se tiene la combinación lineal de las señales o datos de entrada, y el umbral de estimulación es una función escalonada a la que se llama función de activación, lo que se observa en la figura 3

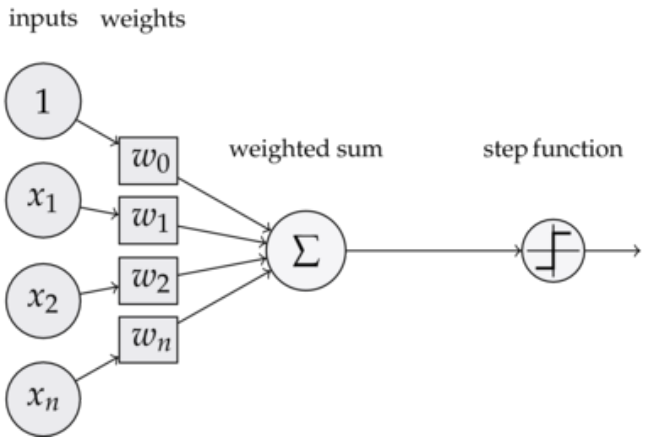


Figura 3: Representación gráfica de un perceptrón.

El modelo de perceptrón se creó con el objetivo de realizar clasificación estadística binaria y poder aproximarse a alguna función $f : X \to Y$ desconocida recompilando datos de ambos conjuntos. Entonces el modelo se puede escribir de la siguiente manera:

$$h(x) = \begin{cases} 1 & \sum_i w_i x_i \geq 0 \\ 0 & \sum_i w_i x_i \leq 0. \end{cases}$$

Una vez definido el modelo se necesita saber qué tan preciso es, para lo cual se utiliza el error cuadrático como función de perdida L

$$L(x, y) = \sum_i^n (y_i - h(x_i))^2,$$

este error se utilizará en el algoritmo de actualización de pesos para modificar el vector w hasta reducir el error lo mayor posible.

El perceptrón presentaba limitaciones como la imposiblidad de clasificar modelos que no pudieran ser separados geométricamente por un hiperplano lineal. Como el caso de la función XOR, que se ilustra en la figura 4.

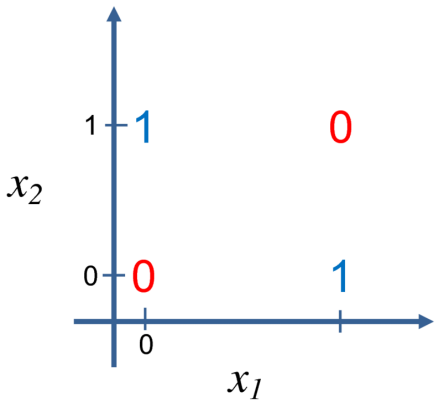


Figura 4: Gráfica de la función XOR.

Para solucionar este problema se desarrollo el perceptrón multicapa, que consta de varias capas de perceptrones apiladas, como un conjunto de neuronas interconectas, por ello el nombre **Red Neuronal Artificial**.
Hay unas cuantas diferencias con el perceptrón original además de la cantidad de perceptrones, como lo es una diferente función de activación y las entradas de cada perceptrón. En la figura 5 se observa la notación que tiene el perceptrón multicapa, a^l representa la $l-$ésima capa, W^l es la matriz de pesos de todos los perceptrones entre la capa a^l y a^{l-1}, y también b^l son los sesgos o pesos constantes de la capa a^l. a^L representa la capa final, y $a^1 = x$ es la capa de entrada.

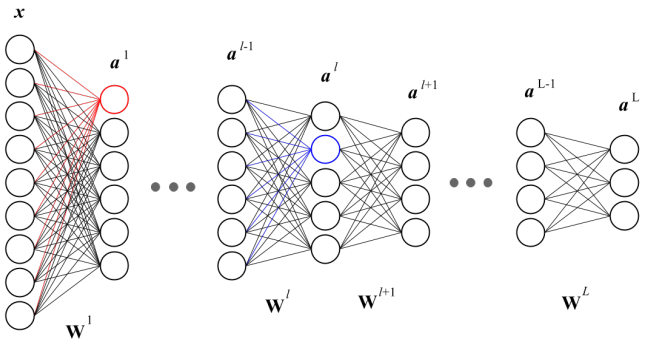


Figura 5: Representación gráfica de Perceptrón Multicapa.

Ahora las entradas del perceptrón que se encuentra en la capa l de tamaño K son las salidas de los perceptrones de la capa $l-1$ de tamaño J:

$$z_k^l = \sum_j w_{kj}^l a_j^{l-1} + b_k^{l-1}, \tag{1}$$

$$a_k^l = \sigma(z_k^l), \tag{2}$$

o en notación matricial

$$a^l = \sigma(W^l a^{l-1} + b^{l-1}). \tag{3}$$

La función de activación se cambió de una función escalonada a una función sigmoide (ecuación 3) para que sea diferenciable y se pueda aplicar el algoritmo de propagación hacía atras o **Backpropagation** en inglés.

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{4}$$

Esta función es parecida a la función escalonada gráficamente.

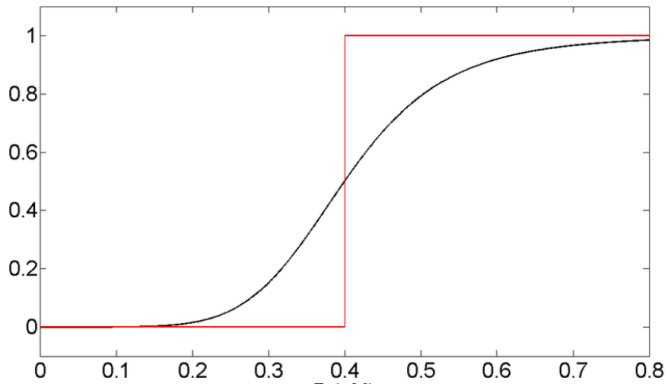


Figura 6: Gráfica función escalonada (rojo) y sigmoide (negro).

La función de costo es la misma que el perceptrón normal pero la diferencia sería unicamente con la capa a^L

$$L(x, y) = \sum_i^n (y_i - a^L)^2,$$

donde

$$a^L = \sigma(W^{(L)}\sigma(\dots\sigma(W^{(2)}\sigma(W^{(1)}x+b^{(1)})+b^{(2)})\dots)+b^{(L)}).$$

Entonces para reducir la perdida igual que en el perceptrón original el objetivo es encontrar el tensor de pesos que reduzca la función de perdida lo mayor posible, es decir, encontrar el mínimo de la función L. Debido a que L es una función real y convexa podemos utilizar los criterios de la primera y segunda derivada para encontrar el mínimo. El problema es que es casi imposible obtener analíticamente las derivadas parciales de la función de perdida respecto a cada peso

$$\frac{\partial L}{\partial w_{ij}^l},$$

para eso es el algoritmo de backpropagation, que consiste en encontrar numéricamente las derivadas parciales respecto a cada peso.

Backpropagation: Suponiendo que tenemos las capas de la figura 5, sabemos por regla de la cadena que

$$\frac{\partial L}{\partial w_{kj}^l} = \frac{\partial L}{\partial a_k^l}\frac{\partial a_k^l}{\partial z_k^l}\frac{\partial z_k^l}{\partial w_{kj}^l},$$

aplicando nuevamente regla de la cadena

$$\frac{\partial L}{\partial w_{kj}^l} = \left(\sum_m \frac{\partial L}{\partial z_m^{l+1}}\frac{\partial z_m^{l+1}}{\partial a_k^l}\right)\frac{\partial a_k^l}{\partial z_k^l}\frac{\partial z_k^l}{\partial w_{kj}^l},$$

sustituyendo y simplificando

$$\frac{\partial L}{\partial w_{kj}^l} = \left(\sum_m \frac{\partial L}{\partial z_m^{l+1}} w_{mk}^{l+1}\right)\sigma'(z_k^l)a_j^{l-1}.$$

Por convención se define δ que es el *error*.

$$\delta_k^l \equiv \frac{\partial L}{\partial z_k^l}$$

Y sabemos que

$$\delta_k^l = \left(\sum_m \frac{\partial L}{\partial z_m^{l+1}} w_{mk}^{l+1}\right)\sigma'(z_k^l) = \left(\sum_m \delta_m^{l+1} w_{mk}^{l+1}\right)\sigma'(z_k^l)$$

Llegando a la ultima ecuación podemos observar que las δ de la capa l están en función de la capa $l+1$, así es como se *propaga* el error hacía atrás. Así que lo primero que importa es obtener δ^L, para obtener las demás δ.

$$\delta_j^L = \frac{\partial L}{\partial a_j^L}\sigma'(z_j^L)$$

Recordando que una de las propiedades de la función sigmoide es

$$\frac{d\sigma(x)}{dx} = \sigma(x)(1-\sigma(x)),$$

podemos reescribir δ^L como:

$$\delta_j^L = \frac{\partial L}{\partial a_j^L}\sigma(z_j^L)(1-\sigma(z_j^L).$$

Con esta última expresión podemos observar que para realizar el backpropagation unicamente tenemos que calcular una derivada parcial $\frac{\partial L}{\partial a_j^L}$, y todas las demás las obtenemos propagando $\delta^{L-1} = \delta^L W^L \circ \sigma'(z^L)$, hasta δ^1.
Teniendo las definiciones anteriores el algoritmo es el siguiente:

$$\delta^L = \nabla_{a^L} L \circ \sigma'(z^L),$$

$$\delta^l = (W^{(l+1)})^T \delta^{l+1} \circ \sigma'(z^l),$$

$$\frac{\partial L}{\partial b_k^l} = \delta_k^l,$$

$$\frac{\partial L}{\partial w_{kj}^l} = \delta_k^l a_j^{l-1}.$$

Una vez obtenido el gradiente, se utilizan algoritmos de optimización basados en gradiente, para encontrar el mínimo de la función L, por ejemplo descenso por gradiente:

$$w_{kj}^l = w_{kj}^l - \alpha \frac{\partial L}{\partial w_{kj}^l}.$$

II-B. Red Neuronal Convolucional

Por el teorema de aproximación universal, el perceptrón multicapa es capaz de aproximar a cualquier función del tipo $f : X^m \to [0,1]^n$. Teóricamente es cierto pero prácticamente no se tienen los recusos computacionales para desarrollar un MLP (Multilayered Perceptron) con una cantidad infinita de perceptrones. Dada esta limitación el desempeño de los MLP no es muy buena al momento de clasificación de vectores de alta dimensionalidad, dado que se tiene que entrenar por mucho tiempo, necesitan cantidades exorbitantes de datos y es muy sensible a variaciones de la entrada.
Por ejemplo en clasificación de imagenes, los imagenes RGB de dimensión $512 \times 512 \times 3$ tendrían que ser transformadas a vectores fila de dimensión 786432, esta dimensionalidad causa los problemas descritos anteriormente.
Para la solución de clasificación de imágenes se desarrollaron las Redes Neuronales Convolucionales, llamadas así porque en lugar de tener perceptrones tiene unidades de convolución.

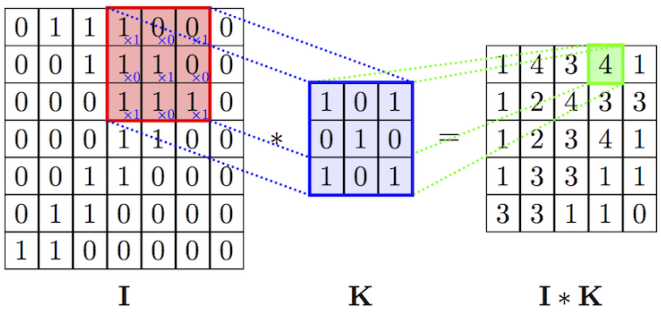


Figura 7: Operación de convolución.

Recordemos la operación de convolución

$$I * K(x,y) = \sum_{a=0}^{k-1} \sum_{b=0}^{k-1} I(x-a, y-b) K(a,b)$$

Una unidad de convolución en lugar de tener un peso con cada unidad de la capa anterior, cómo los perceptrones, convoluciona con la capa anterior, como se muestra en la figura 7. Esto permite que aprenda relaciones locales de los datos. En el caso de las imagenes existen relaciones fuertemente locales entre pixeles, ya que un pixel tiende a estar con pixeles similares, en caso de no ser una esquina o un borde.
Usando la notación del MLP podemos definir la una unidad de convolución cómo:

$$z_{(x,y)}^l = w_{(x,y)}^l * a_{(x,y)}^{l-1}) = \sum_{x'} \sum_{y'} w_{(x',y')}^l a_{(x-x',y-y')}^{l-1}$$

Donde x' y y' son las dimensiones de la unidad de convolución $a_{(x-x'+1,y-y'+1)}^l$

$$a_{(x,y)}^l = \sigma(z_{(x,y)}^l) + b_{x,y}^l.$$

Es importante notar que ahora cada capa de la red será bidimensional, en la figura 8 hay una ilustración gráfica.

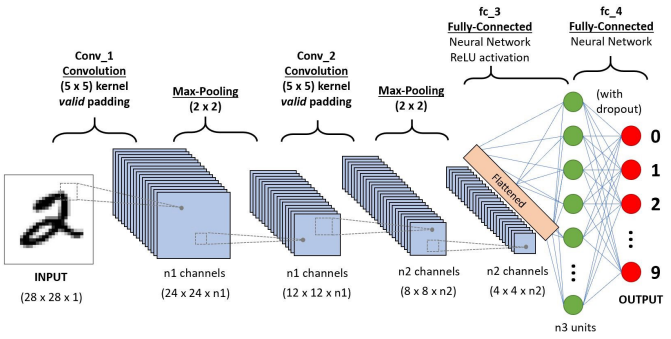


Figura 8: Red Neuronal Convolucional.

Similar al MLP, se utiliza la misma función de perdida, también se optimiza utilizando Backpropagation y descenso con gradiente.
El algoritmo de BackPropagation tendra una ligera variante, para la unidad convolucional, pero se basa en el mismo principio, encontrar deltas de error y propagarlo a capas anteriores.
Como en el MLP tenemos que

$$\frac{\partial L}{\partial w_{x,y}^l} = \sum_{x'} \sum_{y'} \frac{\partial L}{\partial z_{x',y'}^l} \frac{\partial z_{x',y'}^l}{\partial w_{x,y}^l}. \tag{5}$$

Ahora las deltas de error, no serán vectores sino matrices.

$$\delta_{(x,y)}^l \equiv \frac{\partial L}{\partial z_{(x,y)}^l} \tag{6}$$

También nos será util obtener lo siguiente

$$\frac{\partial z_{x',y'}^l}{\partial w_{x,y}^l} = \sigma(z_{(x-x',y-y')}^{l-1}) \tag{7}$$

$$\frac{\partial L}{\partial w_{x,y}^l} = \sum_{x'} \sum_{y'} \delta_{(x',y')}^l \sigma(z_{(x-x',y-y')}^{l-1}) \tag{8}$$

Una vez definido lo anterior podemos proceder a obtener las $\delta^{(l+1)}$

$$\frac{\partial L}{\partial z_{x,y}^l} = \sum_{x'} \sum_{y'} \frac{\partial L}{\partial z_{x',y'}^{l+1}} \frac{\partial z_{x',y'}^{l+1}}{\partial z_{x,y}^l} = \delta_{x',y'}^{(l+1)} \frac{\partial z_{x',y'}^{l+1}}{\partial z_{x,y}^l} \tag{9}$$

Ahora falta obtener $\frac{\partial z_{x',y'}^{l+1}}{\partial z_{x,y}^l}$, desarrollando $z_{x',y'}^{l+1}$

$$z^{l+1}_{x',y'} = \sum_{x''}\sum_{y''} w^{l+1}_{(x'',y'')}\sigma(z^{l}_{(x'-x'',y'-y'')}) \tag{10}$$

Entonces sabiendo por los indices de la ecuación 10 que $x'' = x' - x$, por que cuando $x'' = x' - x \implies \sigma(z^l_{(x'-x'',y'-y'')}) = \sigma(z^l_{(x,y)})$ y ese es el único coeficiente que no se vuelve cero después de aplicar la derivada.

$$\frac{\partial z^{l+1}_{x',y'}}{\partial z^l_{x,y}} = w^{l+1}_{(x'-x,y'-y)}\sigma'(z^l_{(x,y)}) \tag{11}$$

Sustituyendo 11 en 8 obtenemos las siguientes definiciones

$$\frac{\partial L}{\partial w^l_{x,y}} = \sum_{x'}\sum_{y'} \delta^{l+1}_{(x',y')} w^{l+1}_{(x-x',y-y')}\sigma'(z^l_{(x,y)}) \tag{12}$$

$$\frac{\partial L}{\partial w^l_{x,y}} = \delta^{l+1}_{(x,y)} * w^{l+1}_{(x,y)}\sigma'(z^l_{(x,y)}) \tag{13}$$

Obtenidas las derivadas parciales, solo es aplicar los pasos del backpropagation.

II-C. U-net

Existe un amplio consenso de que el entrenamiento exitoso de redes profundas requiere de miles de muestras de entrenamiento propiamiente anotadas. U-Net ofrece una estrategia de red y capacitación que se basa en uso del aumento de datos para usar las muestras anotadas disponibles de manera más eficientemente. La arquitectura consiste en una ruta de contracción para capturar el contexto y una ruta de expansión simétrica que permite una localización precisa.

Se puede observar la etapa de contracción en la figura 9, consiste en capas de MaxPooling, que reducen la dimensión de la capa anterior, y Convoluciones de 3×3.

La etapa de expansión está a la derecha, consiste en capas de up-conv, que incrementan la dimensión de la capa anterior, y convoluciones de 3×3. Además, después de cada up-conv se concatena el downsample de la misma dimensión de la etapa de contracción.

Omite tener capas fully connected para disminuir la cantidad de muestras necesarias para entrenar, esto es de gran utilidad porque como se menciona en las páginas anteriores, al agregar capas fully-connected se necesitan muchos más datos para entrenar.

En lugar de utilizar capas fully-connected al final (figura 8), utiliza una unida de upsampling, que consiste en aumentar la dimensión del tensor de entrada utilizando algún método de interpolación. Además del upsampling se concatena con su equivalente del downsampling, para agregar los bordes que son eliminados por el maxpooling.

Al omitir capas fully-connected permite una mayor flexibilidad para realizar data augmentation.

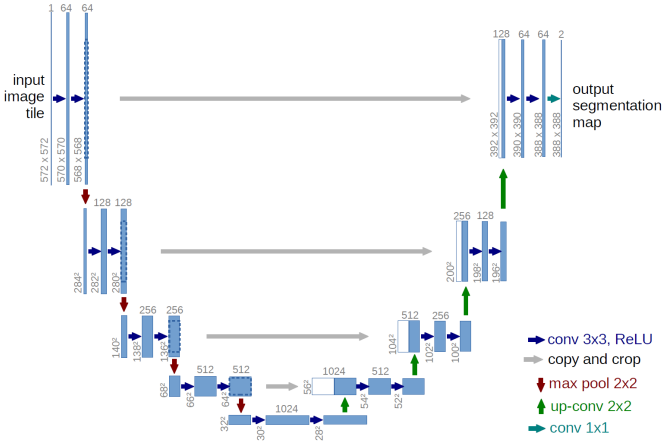


Figura 9: Arquitectura U-net.

II-D. MobileNetV2 y aprendizaje transferido

Para facilitar el entrenamiento de la U-net, utilizamos un modelo pre-entrenado llamado MobileNetV2 [2], el cual es una clase de modelos eficientes llamados MobileNets para aplicaciones de visión móviles e integradas. Las MobileNets se basan en una arquitectura simplificada que utiliza convoluciones separables en profundidad para construir redes neuronales ligeras.

Esto es de utilidad por el concepto de transferencia de aprendizaje, como se observa en 10, las primeras capas de cualquier red neuronal convolucional aprenden a caracterizar texturas simples en las imágenes, cada capa combina las características que fueron aprendidas por la capa anterior. Entonces partiendo de este principio al momento de entrenar una nueva red podemos utilizar las primeras capas de alguna red ya entrenada para que la nueva red no tenga que ajustar esos pesos y asi agilizar el proceso de entrenamiento.

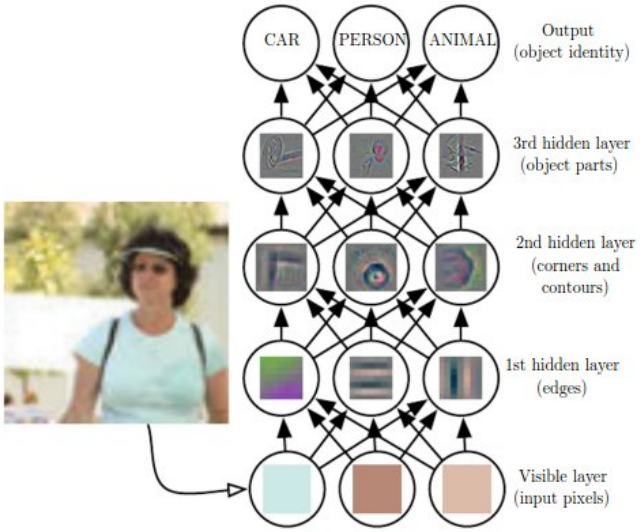


Figura 10: Abstracción capa por capa.

II-E. *Data Augmentation*

Data Augmentation o aumentación de datos es una técnica de remuestreo para generar bases de datos grandes en base a bases de datos de menor tamaño. Al aplicarle una función propia al dominio del problema a la base de datos original, es posible generar nuevos elementos pertenecientes a la misma clase. El que esta técnica sea efectiva depende fuertemente de que la función de transformación aplicada genere elementos lo suficientmente distintos al original, así como que estos sigan manteniendo los rasgos principales de la clase a la que pertenecen.

Existen diversas formas de aumentar los datos. En el caso de las imágenes, algunas de las principales son rotaciones, aumentos, cortes, imágen en espejo y transformaciones geométricas.

II-F. *Bagging*

El *Bagging* (Bootstrapping Aggregation) es un meta-algoritmo en aprendizaje máquina que combina dos técnicas:

Boostrapping es una técnica de remuestreo para poder mejorar la inferencia sobre un parámetro de una población. Dada una muestra aleatoria de la misma, se generan remuestreos con reemplazo de esta misma para así tomar la muestra aleatoria como un sustituto de la población. Si esta muestra aleatoria original captura bien la esencia de la población, la inferencia realizada sobre los remuestreos de la muestra aleatoria será sumamente similar a la inferencia realizada sobre la población.

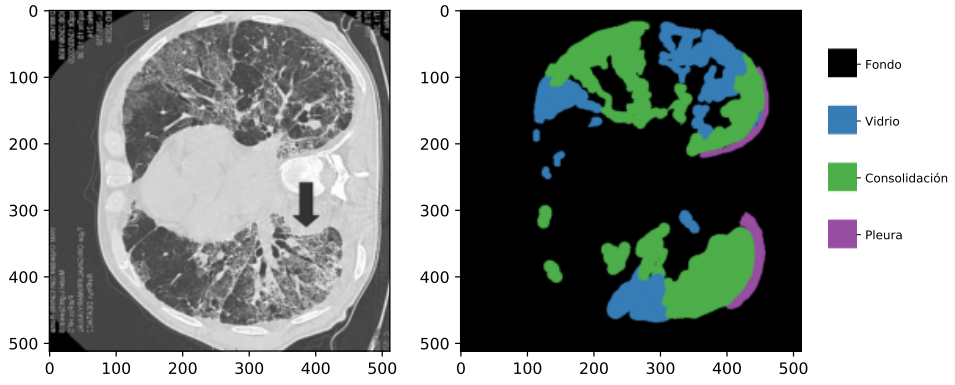
Aggregation es una técnica para generar agentes de aprendizaje fuertes mediante la agregación o combinación de agentes de aprendizaje débiles. En otras palabras, si se tienen k agentes de aprendizaje débiles y se combinan sus predicciones, la predicción agregada de los mismos tenderá a tener menos sesgo y varianza que el de cualquiera generada por uno de los agentes débiles.

III. Implementación

Para el desarrollo del se usó *COVID-19 CT segmentation dataset* de [1]. Para esto utilizamos tres archivos con extensión *.nii*, el cual es un formato médico de imagen para radiografías y tomografías. Las imágenes vienen cargadas en un vector conjunto de $512 \times 512 \times 100$, donde cada elemento de la tercera dimensión es una imagen de 512×512 píxeles. Así mismo se nos proporcionan las capas de segmentación para estas imágenes realizadas por profesionales médicos.

Las capas de segmentación muestran las clases de cada pixel de su imagen correspondiente mediante las siguientes clasificaciones.

- 0: Fondo Normal
- 1: Vidrio
- 2: Consolidacíon
- 3: Pleura



(a) Radiografía original (b) Segmentación con regiones de interés

Figura 11: Ejemplo de las imágenes y sus segmentaciones.

Las 100 imágenes contienen las segmentaciones de indicadores de enfermedad observadas en las radiografías axiales pulmonares de 40 pacientes infectados con SARS-CoV-2 (COVID-19).

Como se puede notar, este es un tamaño muy reducido para una base de datos, por lo decidimos combinar la aumentación de datos y el bootstrapping. Debido a que deseamos entrenar una red neuronal con entradas en una cierta resolución, primero reescalamos las imágenes a las siguientes resoluciones (512×512, 224×224, 192×192). Una vez realizado esto, generamos 5000 remuestreos con reemplazo para generar nuevos conjuntos de datos para cada resolución.

Después de obtener estos conjuntos remuestreados decidimos basarnos en el artículo *Best Practices for Convolutional Neural Networks Applied to Visual Document Analysis* [3] y aplicar deformaciones elásticas a estas imágenes. Debido a que este tipo de deformaciones simulan las diferentes formas que pueden tener las imágenes médicas en una radiografía, nos permiten realizar aumentación de datos. Asimismo rotamos las imágenes deformadas aleatoriamente para agregar otro factor de transformación a los datos aumentados.

Una vez terminado este proceso, obtuvimos 3 conjuntos de 5000 imágenes remuestreadas y aumentadas. Debido a que la deformación tiene un elemento aleatorio, todas las imágenes fueron en principio distintas.

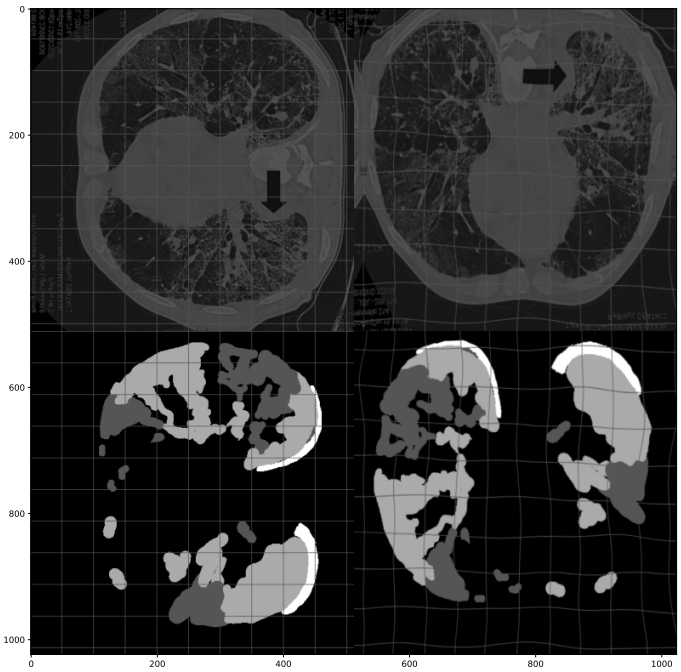


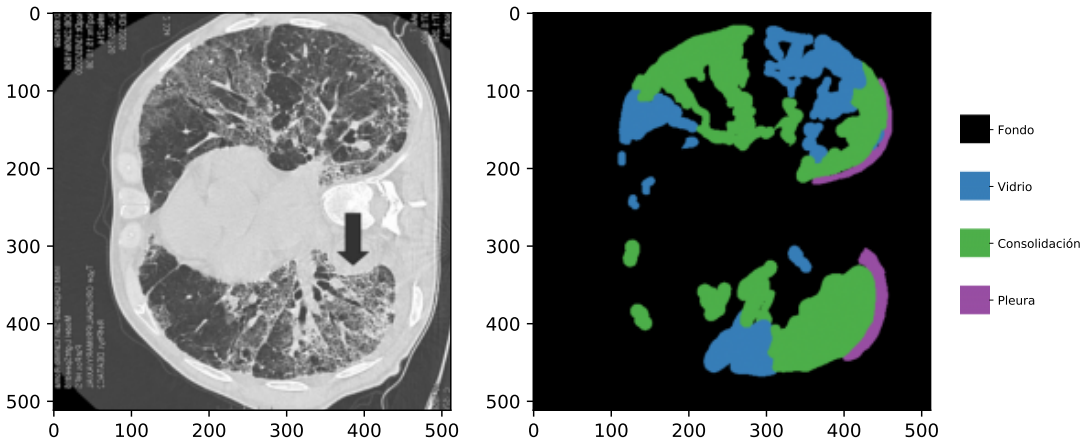
Figura 12: Ejemplo de como se realiza la deformación elástica para la aumentación de datos.

Para el entrenamiento de nuestras redes neuronales decidimos seguir el siguiente proceso:

1. Generar tres redes neuronales basadas en la red MobileNetV2 y Pix2Pix. Mediante transferencia de aprendizaje, podemos evitar el entrenamiento de una gran parte de la red.
2. La red de más alta definición, 512×512 tendrá unas capas convolucionales y de max pooling antes de la sección importada de MovileNetV2 para soportar esta resolución.
3. Las redes de menos definición, 224×224 y 192×192 carecerán de estas capas añadidas ya que MobileNetV2 tiene soporte para estas entradas.
4. Las capas de MovileNetV2 servirán como el codificador (*encoder*) de nuestra U-Net. Para el decodificador, utilizamos las capas de Pix2Pix para sobremuestreo para generar las máscaras de clasificación en la resolución necesaria.
5. Las dos redes con menor resolución fueron entrenadas por 40 épocas cada una. De estas, se selecciónó la red con menor pérdida respecto al conjunto de validación. El conjunto de validación utilizado fueron las 100 imágenes originales en la resolución propia para la red junto con sus segmentaciones.
6. Para la red de mayor resolución, debido a la falta de recursos de memoria tomamos un enfoque distinto.
   - Realizamos 10 iteraciones de entrenamiento diferentes, cada una con 20 épocas.
   - En cada iteración se generaba un conjunto de 500 imágenes y máscaras aumentadas diferente. Esto causaba que cada en cada iteración distinta se tuviera acceso a datos diferentes, lo que evitaba el sobreentrenamiento.
   - De igual manera a las otras redes, tomamos la red que presentaba una menor pérdida respecto al conjunto de validación (las 100 imágenes originales y sus máscaras.)
7. Una vez obtuvimos las tres redes neuronales, agregamos sus resultados con pesos iguales. Esto debido a que dependiendo de la resolución de la capa la red se vuelve mejor para detectar ciertos tipos de detalles, por lo que la predicción compuesta de las tres redes era bastante mejor que la de cualquiera de estas individualmente.

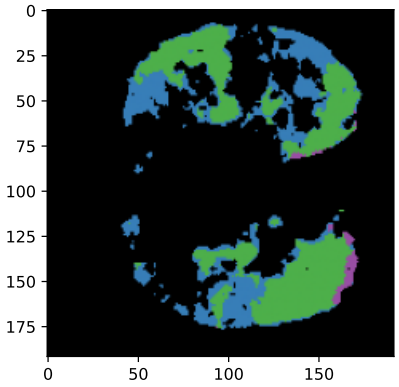
IV. Resultados

A continuación se muestran las segmentaciones generadas por cada red así como su agrupación. Cabe destacar que el nivel de detalle varía dependiendo de la resolución de entrada de la red, por lo que en conjunto dan una predicción más acercada a la de las imágenes que individualmente.

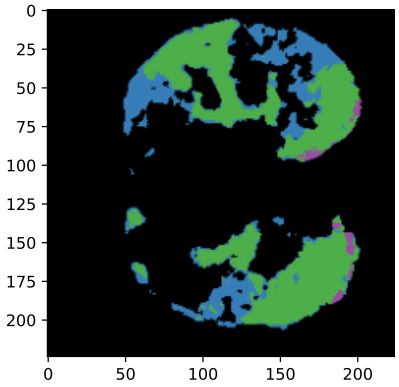


(a) Radiografía original

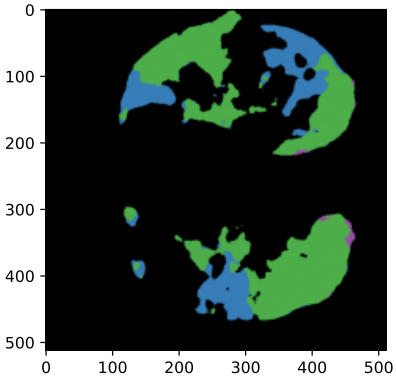
(b) Segmentación con regiones de interés



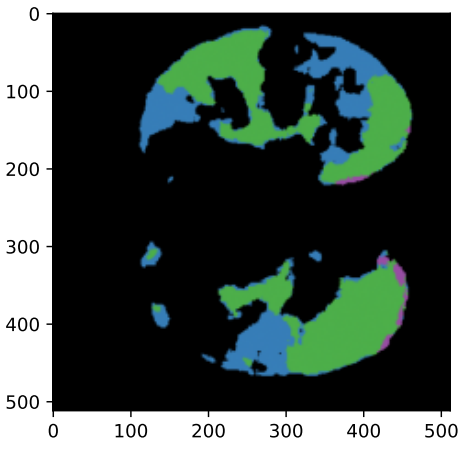
(d) Predicción de red 192x192



(e) Predicción de red 224x224



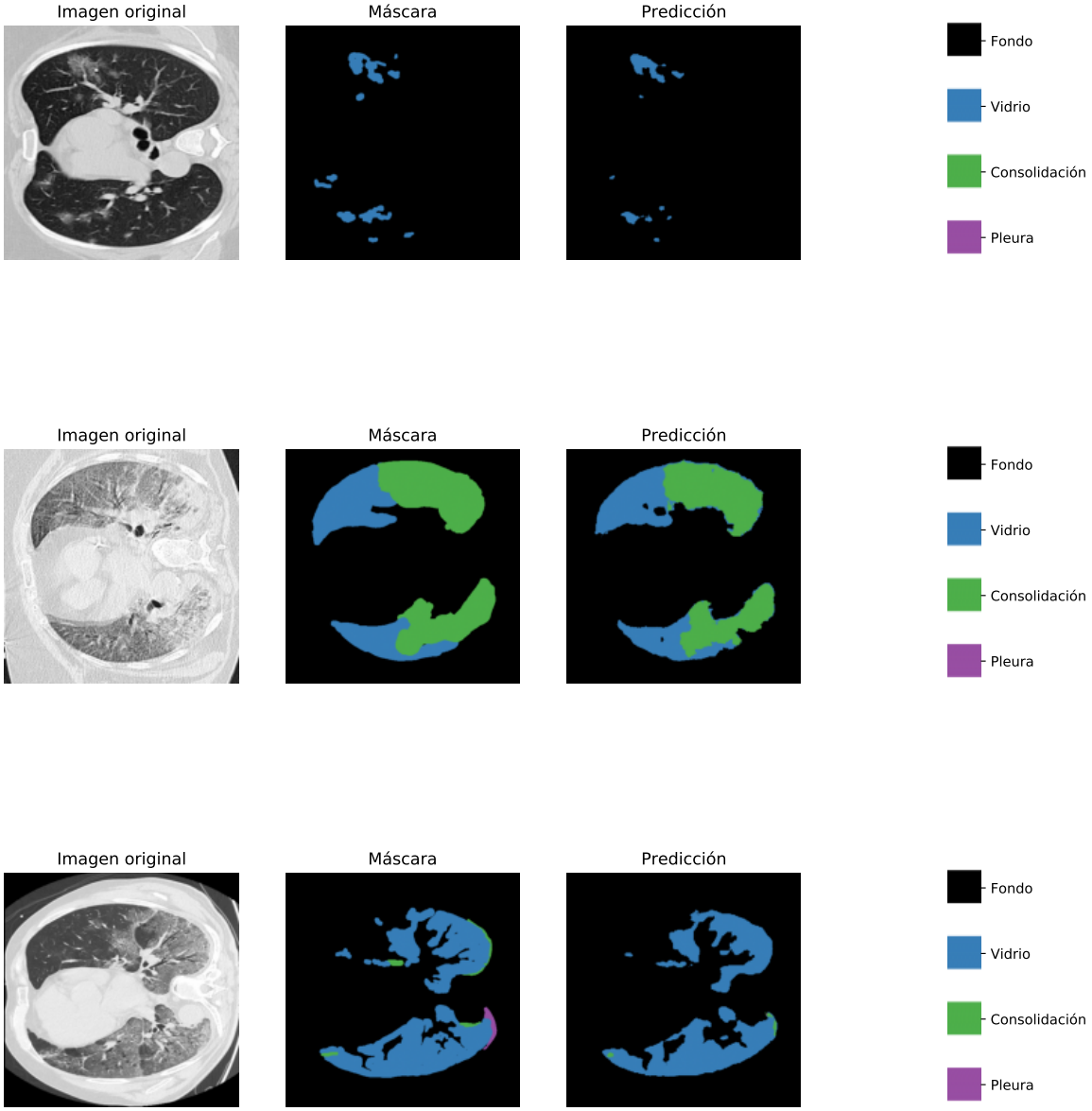
(f) Predicción de red 512x512



(g) Predicción conjunta agrupada por *bagging*

Figura 13: Predicciones individuales y agrupadas por *bagging*.

Como se puede notar, las predicciones generadas para la base de datos de la que se realizó aumentación de datos es bastante eficaz. En las siguientes imágenes se muestran algunas predicciones comparadas con la mascara verdadera. Al igual que en el caso donde se comparan las predicciones de las redes individuales, las predicciones conjuntas son muy acercadas a las máscaras reales. Debe notarse que debido al desbalance de clases, la efusión pleural no es detectada si esta no es pronunciada. Si se entrenara la red con más imágenes (cosa que nuestros recursos no nos permitían), esta sería reconocida más eficazmente.



(c)

Figura 14: Segmentación para imágenes de la base de datos aumentada.

IV-A. Desempeño

En la tabla I se muestra el desempeño de nuestro modelo para las distintas clases, lo primero que hay que notar es que hay un desbalance de clases, principalmente se observa que hay pocos ejemplos de pleura para que el modelo aprenda, lo que se refleja en que la segmentación para la pleura tenga un desempeño más bajo que las otras clases.

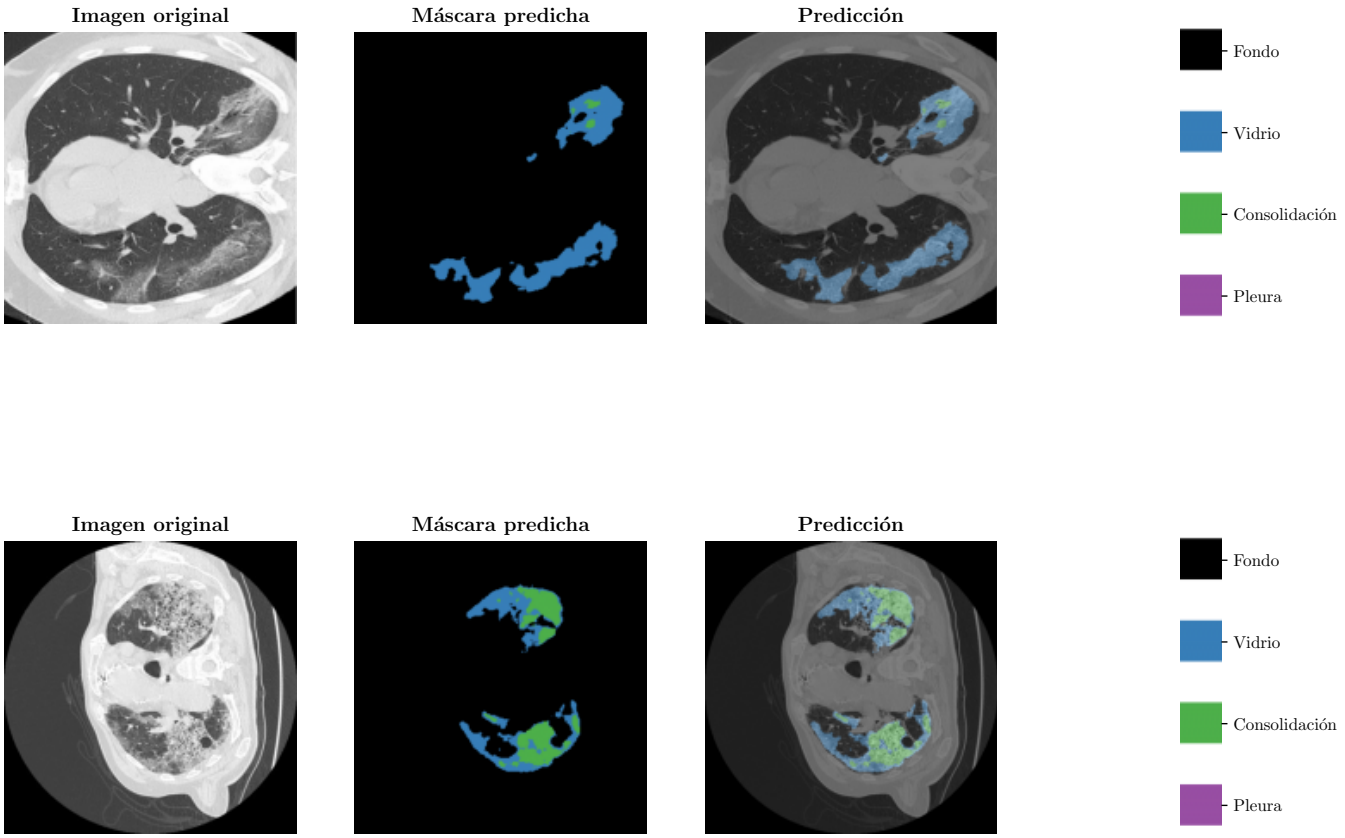
Clase	Precision	Recall	F1-score	Soporte
Fondo	0.98	1.00	0.99	24394464
Vidrio	0.80	0.63	0.71	1196461
Consolidación	0.94	0.63	0.75	589210
Pleura	0.99	0.19	0.31	34265

Tabla I: Desempeño de nuestro modelo para las distintas clases.

IV-B. Predicción

Aunque claramente da buenos resultados en comparación a imágenes conocidas, ¿cómo nos aseguramos de que simplemente no se sobreentrenaron estas redes con respecto a nuestra base de datos original?

Para responder esta pregunta, decidimos segmentar las imágenes del dataset de validación que proporciona la misma página. Aunque no somos expertos, creemos que los resultados hablan por sí mismos.



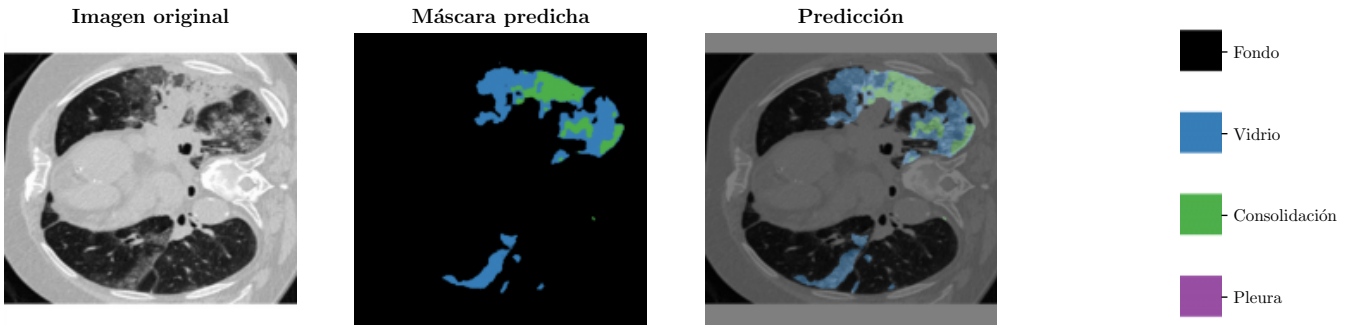


Figura 15: Segmentación para imágenes con máscara desconocida.

Conclusiones

Durante todo el curso hemos explorado diversas técnicas para tratar imágenes, poder analizarlas y generar predicciones sobre estas. Sin lugar a duda esta implementación de una U-net es la más poderosa que hemos visto hasta ahora. Sin embargo sin los conceptos vistos en clase como la convolución, filtros, métricas y algoritmos de concenso, no tendríamos las bases teóricas para entender lo que hace esta red.

Creemos que esta práctica fue de gran valor exploratorio para nuestro entendimiento del mundo de las redes neuronales artificiales. Por medio de estas pudimos enfrentar un problema muy real en el mundo actual, y los resultados aquí obtenidos podrían, implementados con más cuidado, llegar a salvar vidas. Es por esto que consideramos que este pequeño proyecto fue de gran valor para que podamos implementar soluciones similares a problemas igualmente importantes de forma correcta y eficaz.

Referencias

[1] 2020 Artificial Intelligence AS. *COVID-19 CT segmentation dataset*. Consultado el 5 de junio de 2020 en http://medicalsegmentation.com/covid19/.

[2] Google Inc. *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications*. Consultado el 9 de junio de 2020 en https://arxiv.org/pdf/1704.04861.pdf.

[3] Simard, P., Steinkraus, D., & Platt, J. (2003). Best Practices for Convolutional Neural Networks Applied to Visual Document Analysis. Consultado el 9 de junio de 2020 en http://cognitivemedium.com/assets/rmnist/Simard.pdf.

[4] Significados.com. *Significado de Neurona*. Consultado el 9 de junio de 2020 en https://www.significados.com/neurona/.

[5] Khan Academy *¿Qué causa la hiperpolarización y la despolarización del potencial de membrana y cómo el cambio en este activa los potenciales graduado y de acción para la transmisión de señales?* Consultado el 9 de junio de 2020 en https://es.khanacademy.org/science/biology/human-biology/neuron-nervous-system/a/depolarization-hyperpolarization-and-action-potentials.

[6] Spears, Brian *Contemporary machine learning: a guide for practitioners in the physical sciences*. (2017). Consultado el 9 de junio de 2020 en https://arxiv.org/pdf/1712.08523.pdf.

[7] Ihab, S. Mohamed. *Detection and Tracking of Pallets using a Laser Rangefinder and Machine Learning Techniques* (2017). Consultado el 9 de junio de 2020 en https://www.researchgate.net/publication/324165524_Detection_and_Tracking_of_Pallets_using_a_Laser_Rangefinder_and_Machine_Learning_Techniques.

[8] Sumit, Saha. *A Comprehensive Guide to Convolutional Neural Networks*. Consultado el 9 de junio de 2020 en https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53.

[9] Olaf Ronneberger and Philipp Fischer and Thomas Brox. *U-Net: Convolutional Networks for Biomedical Image Segmentation*. (2015). Consultado el 9 de junio de 2020 en https://arxiv.org/pdf/1505.04597.pdf.

[10] Sawani Jina/ *How Does COVID-19 Appear in the Lungs?*. (2020). Consultado el 9 de junio de 2020 en https://labblog.uofmhealth.org/lab-report/how-does-covid-19-appear-lungs

[11] Christophe M. Walker and Gerald F. Abbott and Reginald E. Greene and Jo-Anne O. Shepard. *Imaging Pulmonary Infection: Classic Signs and Patterns*. (2013). Consultado el 9 de junio de 2020 en https://www.ajronline.org/doi/pdf/10.2214/AJR.13.11463.

[12] Cleveland Clinic medical professional. *Pleural Effusion Causes, Signs & Treatment*. (2018). Consultado el 9 de junio de 2020 en https://my.clevelandclinic.org/health/diseases/17373-pleural-effusion-causes-signs--treatment#:~:text=Pleural%20effusion%2C%20sometimes%20referred%20to,to%20lubricate%20and%20facilitate%20breathing.