



Grado en Ingeniería Informática

Curso 2019-2020

Aprendizaje Automático

Trabajo Final

Autores:

Alba Reinders Sánchez	100383444
Alejandro Valverde Mahou	100383383

Índice

1. Definición del Problema	3
2. Datos y Obtención de Datos	4
3. Preproceso de Datos	4
4. Modelado	6
5. Evaluación	10
6. Uso	11
7. Anexo	12
8. Referencias	12

Índice de figuras

1. Muestra del virus <i>COVID-19</i>	3
2. Muestras víricas de ejemplo	4
3. Imágenes generadas por rotación	5
4. Imágenes generadas por volteo	5
5. Imágenes generadas por traslación	5
6. Imágenes generadas con zoom	6
7. Operación convolucional en una matriz 3x3 con un <i>kernel</i> de 2x2	7
8. Arquitectura de la red neuronal usada para la clasificación	8
9. Función de activación ReLU	9
10. Gráfico de evaluación del modelo	10
11. Imágenes de evaluación del modelo	11

1. Definición del Problema

El problema a resolver es la clasificación de imágenes de muestras víricas para detectar el virus *SARS-CoV-2*, también conocido como *COVID-19*.

Esta idea nace de una propuesta realizada para el *Hackathon* de la Comunidad de Madrid 'Vence al virus', el pasado 1 y 2 de Abril. La idea es identificar los pacientes infectados a través de las muestras víricas observadas con un microscopio TEM (*Transmission Electron Microscopes*), para así conseguir una prueba rápida, efectiva y de bajo coste.

Esta prueba debe ser capaz de reconocer si el paciente está infectado por algún virus de la familia *Coronaviridae* y, dado que existen muy pocos virus de esta familia que afectan a humanos, se puede asumir con cierta probabilidad que el paciente está infectado con el *COVID-19*.

Se ha planteado resolver este problema a través de técnicas de aprendizaje automático porque se trata de una tarea de clasificación. Dado que las imágenes usadas para el entrenamiento tienen una clase asociada, se pueden usar técnicas de **aprendizaje supervisado**.

Esto permitiría crear tests para detectar el *COVID-19* con una probabilidad elevada de acierto, siendo una prueba rápida, que se podría realizar a mucha gente y en muy poco tiempo. Además, reduciría la carga de los especialistas porque se convertiría en un proceso automatizado.

Respecto a la viabilidad de la solución propuesta, se cree que supondría una reducción de coste y tiempo respecto a los medios actuales usados para detectar este virus. Sin embargo, la obtención de las imágenes necesarias para el entrenamiento puede ser complicada, ya que el microscopio que se requiere para obtenerlas no se encuentra en todos los laboratorios de los hospitales.

La fiabilidad que se espera obtener de esta prueba es bastante elevada puesto que se conoce que los virus de la familia *Coronaviridae* poseen una característica distintiva: suelen tener forma esférica y unas púas alrededor de todo su cuerpo, de forma que facilitará la diferenciación de estos virus respecto a otros.

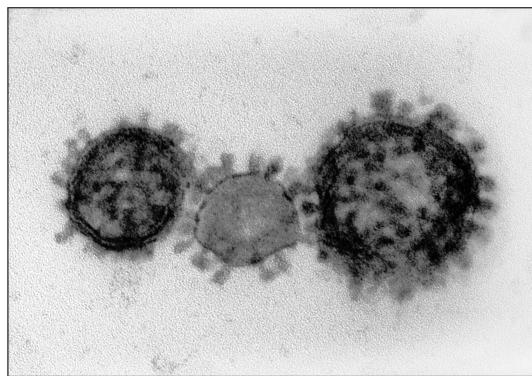


Figura 1: Muestra del virus *COVID-19*

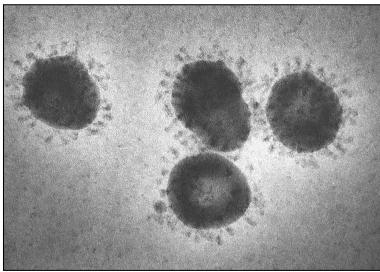
El proceso que hay que llevar a cabo es:

1. Recopilar un número de imágenes de muestras considerable entre las que se encuentren todo tipo de virus, entre ellos, el *COVID-19*. Estas imágenes deberán estar previamente etiquetadas.
2. Realizar un preprocessado a las imágenes.
3. Entrenar el modelo de aprendizaje automático.
4. Evaluar el modelo generado.
5. Utilizar este modelo para clasificar nuevas imágenes.

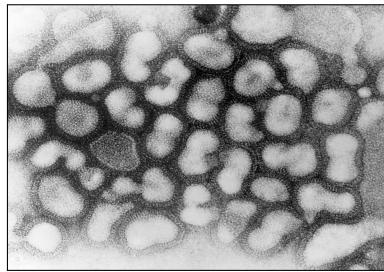
Dado que el conjunto de datos son imágenes, se usa una **red de neuronas convolucionales** para realizar la tarea de aprendizaje automático. Esta red tiene que ser capaz de diferenciar muestras con *COVID-19*, muestras con otros virus, y muestras vacías.

2. Datos y Obtención de Datos

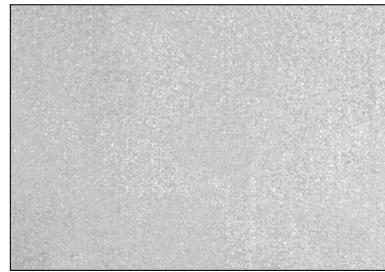
El conjunto de datos que se usan para entrenar y evaluar el modelo son imágenes de muestras víricas clasificadas. Estas imágenes son tomadas por el microscopio TEM, son en blanco y negro y de un tamaño variado. Están clasificadas en tres categorías: **COVID-19**, **otros virus** y **vacía**.



(a) Muestra de *COVID-19*



(b) Muestra de otros virus



(c) Muestra vacía

Figura 2: Muestras víricas de ejemplo

Los datos se obtienen del *NIAID (National Institute of Allergy and Infectious Diseases)* y de la *Biblioteca Sanitaria Pública de EEUU*. También, y en base al *Hackathon*, se estableció contacto con el *CSIC* para conseguir mayor número de imágenes, aunque todavía no se han conseguido.

El problema principal de estas imágenes es su poca cantidad (alrededor de 50 imágenes de cada clase), por eso ha sido necesario contactar con el *CSIC*. Ya que es de vital importancia conseguir un conjunto de imágenes lo suficientemente grande como para poder entrenar a la red de la mejor forma posible.

Otro problema que tienen estas imágenes es la falta de consistencia unas con otras, ya que no comparten ni tamaño ni método de obtención. Muchas de ellas poseen además una gran cantidad de ruido.

Estas dificultades hacen más complicada la tarea de aprendizaje.

3. Preproceso de Datos

Dada la naturaleza de los datos, es necesario realizar ciertas transformaciones. La primera es que, a pesar de que las imágenes son de por sí en blanco y negro, hay ciertos *pixels* que se detectan con formato *RGB* y por tanto se transforman a **escala de grises**.

A continuación, dado que las imágenes no tienen el mismo tamaño, se hace un reescalado para que tengan todas un tamaño de **500x500px**.

Después, se divide el conjunto de datos en **set de entrenamiento** y **set de evaluación**. Al hacer esta separación se ha decidido dividir en una proporción de 20 % evaluación (24 datos) y 80 % entrenamiento (102 datos).

Como el número de datos que se tienen es muy reducido, se opta por usar técnicas para aumentarlos. Esto se hace con el propósito de evitar el *overfitting*, dado que con un número pequeño de datos, es más fácil que se produzca.

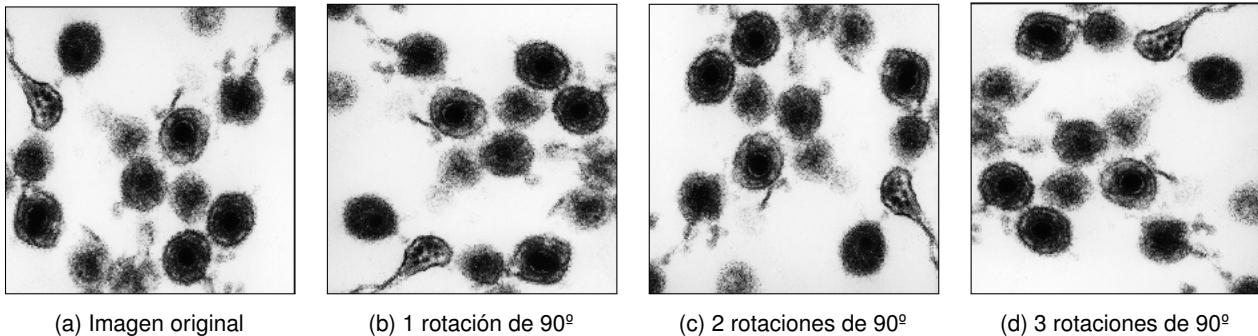
Este aumento de imágenes se realiza tanto en el set de entrenamiento como en el set de evaluación. Aunque las transformaciones no son las mismas.

Las técnicas de aumento de datos que se utilizan sobre el set de entrenamiento son: **rotación, volteo, traslación, zoom, zoom con rotación y zoom con volteo**.

Las técnicas de aumento de datos que se utilizan sobre el set de evaluación son: **rotación y volteo**.

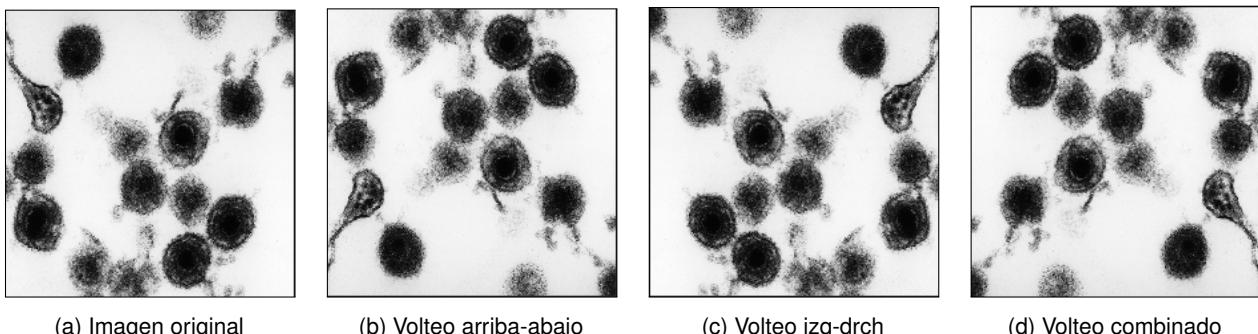
Otras técnicas que serían también útiles pero que no se van a implementar a menos que sean necesarias son: alejar la imagen, rotaciones no solo de 90°, añadir ruido, invertir el color de la imágenes y mezclar distintas transformaciones.

Estas técnicas se pueden aplicar a las imágenes debido a que no son especialmente simétricas, y por tanto, las nuevas imágenes generadas son diferentes a las originales y permiten a la red tener un mejor aprendizaje y reducir el *overfitting*.



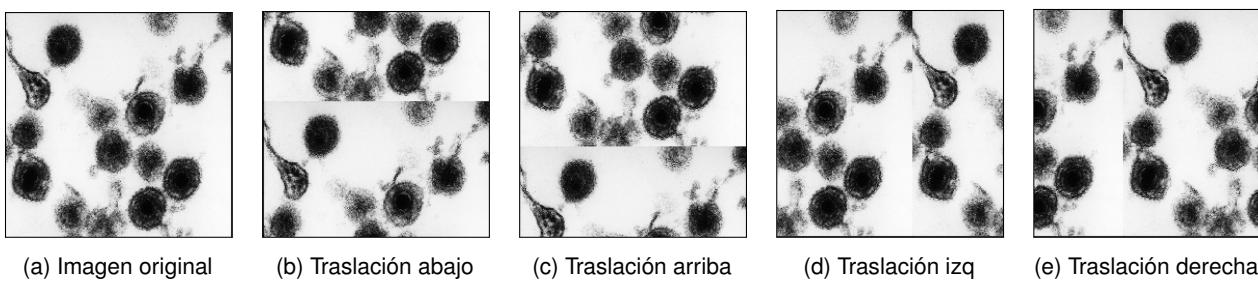
(a) Imagen original (b) 1 rotación de 90° (c) 2 rotaciones de 90° (d) 3 rotaciones de 90°

Figura 3: Imágenes generadas por **rotación**



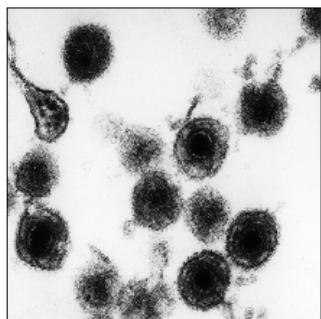
(a) Imagen original (b) Volteo arriba-abajo (c) Volteo izq-dcha (d) Volteo combinado

Figura 4: Imágenes generadas por **volteo**

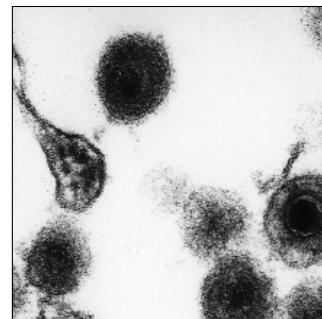


(a) Imagen original (b) Traslación abajo (c) Traslación arriba (d) Traslación izq (e) Traslación derecha

Figura 5: Imágenes generadas por **traslación**



(a) Imagen original



(b) Imagen con zoom

Figura 6: Imágenes generadas con **zoom**

Después de realizar estas transformaciones sobre las imágenes originales, se obtienen **2550** imágenes de entrenamiento y **168** de evaluación.

4. Modelado

Como la tarea definida es una clasificación de imágenes, el algoritmo que se usa es una **red neuronal convolucional**. Esta red está formada por la siguiente combinación de capas: convolucional, *pooling* y densa. Estas redes son especialmente buenas en problemas relacionados con imágenes, porque son capaces de obtener características de las mismas, útiles en esta clasificación.

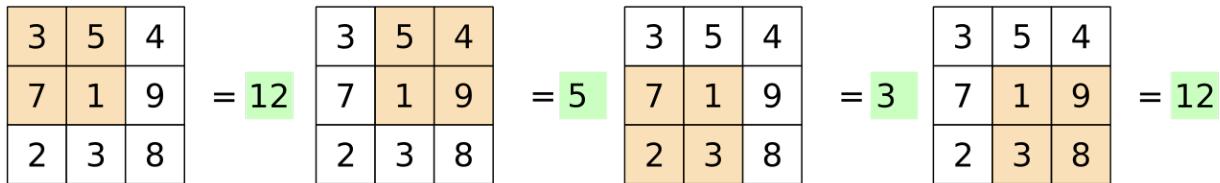
La arquitectura básica consiste en varias capas convolucionales y de *pooling* intercaladas, seguidas de una o varias capas densas.

- **Convolucional:** capa formada por neuronas convolucionales que se encargan de procesar matrices de entrada con una operación sobre ellas, en lugar de un único valor numérico. Usan un *kernel* para recorrer las matrices e ir extrayendo de ellas distintas características, con el objetivo de conseguir información útil. En este caso, la matriz es los valores de los *pixels* de las imágenes. La salida de cada neurona de una capa convolucional es:

$$Y_j = f \left(b_j + \sum_i K_{ij} \otimes Y_i \right)$$

Donde Y_j es la salida de una neurona j y consiste en una matriz que se calcula a través de la combinación lineal de las salidas de las neuronas en la capa anterior (Y_i), cada una de ellas operadas con el *kernel* K_{ij} correspondiente a esa conexión. Esta cantidad se suma a una constante b_j y luego se pasa por la función de activación de la neurona ($f()$).

La anterior operación se realiza de la siguiente manera:

**KERNEL:**

0	1
1	0

Resultado:

12	5
3	12

Figura 7: Operación convolucional en una matriz 3x3 con un *kernel* de 2x2

Donde el *kernel* se inicializa aleatoriamente, y va variando sus valores a lo largo del entrenamiento, buscando la configuración ideal.

Con esto se consigue reducir el tamaño de la matriz de forma que se obtienen las características más relevantes de cada sección en cada capa.

- **Pooling:** capa encargada de reducir el tamaño de la matriz de entrada drásticamente, realizando una operación para resumir las características de una región. En concreto *max-pooling*, encuentra el valor máximo dentro de un grupo de píxeles.
- **Densa:** capa formada por neuronas completamente enlazadas. La salida de cada neurona dentro de una capa densa depende de su entrada y sus pesos asociados, y sigue la siguiente fórmula:

$$Y_j = f \left(b_j + \sum_i \omega_{ij} \cdot Y_i \right)$$

Donde Y_j es la salida de una neurona j y consiste en un valor que se calcula a través de la multiplicación de las salidas de las neuronas en la capa anterior (Y_i), cada una de ellas operadas con el peso asociado a la neurona ω_{ij} . Este valor se suma a una constante b_j y luego se pasa por la función de activación de la neurona ($f()$).

Además, y para conseguir el correcto funcionamiento de la red, es necesario añadir capas adicionales auxiliares:

- **Flatten:** esta capa no aporta información a la red, y su única funcionalidad es la de hacer de puente entre las capas convolucionales y las capas densas, ya que tanto las entradas como las salidas de las capas convolucionales consisten en vectores tridimensionales, mientras que las capas densas trabajan con vectores unidimensionales. Por tanto, esta capa transforma este vector tridimensional 'aplanándolo', es decir, poniendo todos sus datos de forma consecutiva, para que la capa densa pueda utilizarlos.
- **Dropout:** es una forma de regularizar el entrenamiento de la red. Sirve para prevenir el *overfitting* de forma rápida y sencilla.
- **Batch Normalization:** normaliza los valor de la función de activación en una capa oculta. Así se evitan pesos fuera de rango, permite una tasa de aprendizaje más alta y reduce el *overfitting*.

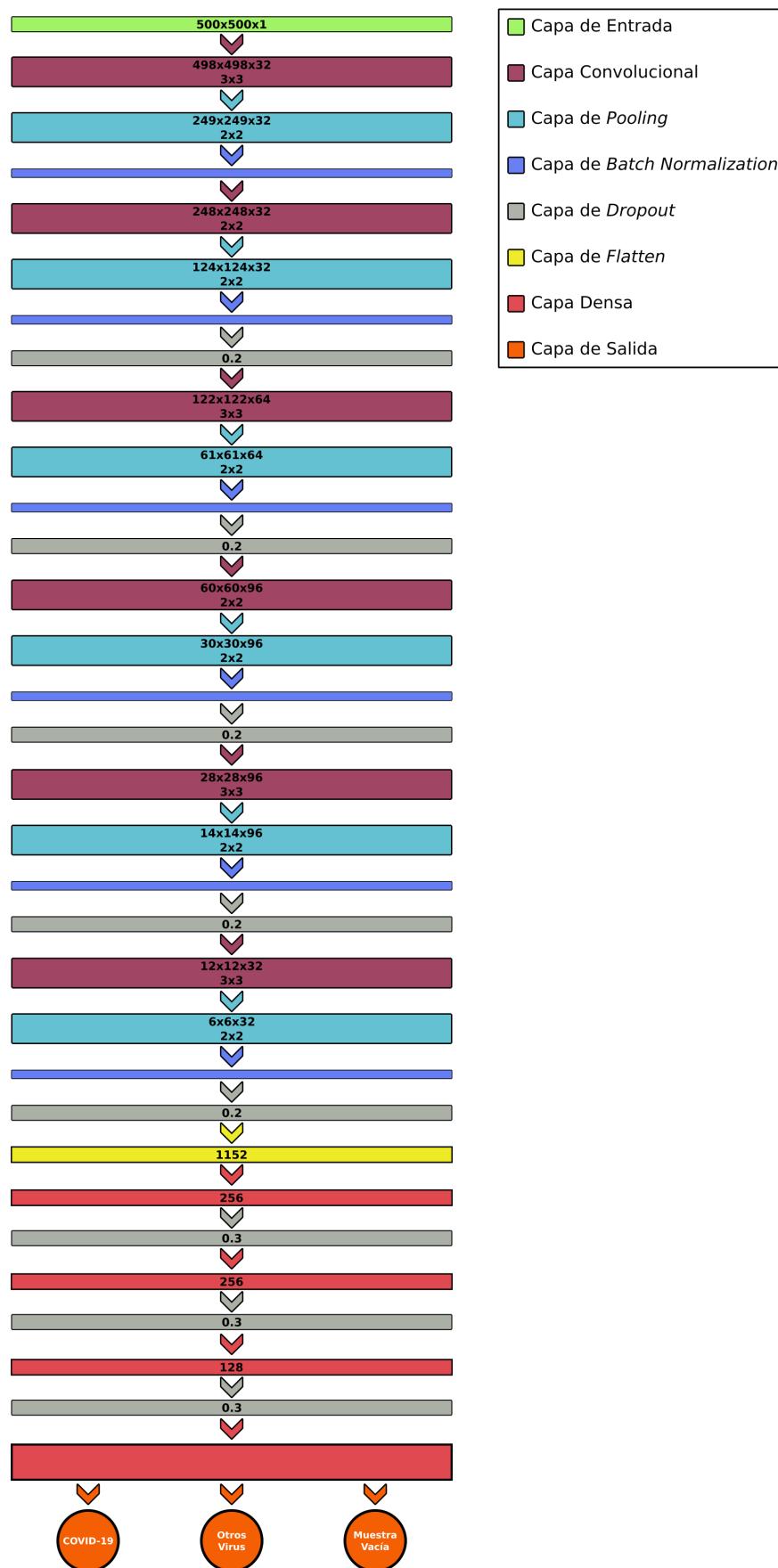


Figura 8: Arquitectura de la red neuronal usada para la clasificación

La arquitectura anterior es la usada en la clasificación. Se ha decidido usar esta arquitectura tras realizar pruebas con variantes de la misma.

Está formada por **31** capas. La entrada es una matriz tridimensional de dimensiones **500x500x1**. La salida son **3** valores distintos donde cada uno representa la confianza del modelo de que una imagen concreta sea de esa cada clase.

La arquitectura se divide en dos grandes bloques, con una capa conectora entre ellos:

1. Bloque Convolucional

Este bloque se encarga de ir extrayendo las características de las distintas secciones de la imagen, reduciendo así su tamaño hasta alcanzar una matriz con dimensiones $6 \times 6 \times 32$. Esto lo hace utilizando la siguiente combinación de capas: **Capa Convolucional**, **Capa de MaxPooling**, **Capa de Batch Normalization** y **Capa de Dropout**. Esta estructura de capas se repite 6 veces, para obtener el resultado deseado.

En el esquema de la página anterior, las capas convolucionales están acompañadas de dos valores: el primer valor, que toma formas como $122 \times 122 \times 64$, indica el tamaño de la matriz tridimensional que produce esa capa. El segundo valor, que es o 3×3 o 2×2 , indica el tamaño del *kernel* de esa capa.

Al igual que las capas convolucionales, las capas de *pooling* cuentan con dos valores, donde el primero indica el tamaño de la matriz tridimensional que devuelven como salida, y el segundo hace referencia a la reducción que realiza, indicando el 2×2 que convierte 2 pixels en 1 solo.

Las capas de *dropout* tan solo tienen un valor, que indica la fracción de unidades que se eliminan aleatoriamente de la red.

2. Capa Flatten

Esta capa no aporta valor al modelo, solo se encarga de transformar la información de salida del bloque convolucional para que pueda utilizarlo como entrada el bloque denso. La transformación que hace esta capa es convertir el vector tridimensional $6 \times 6 \times 32$ en el vector unidimensional 1152.

3. Bloque Denso

Este bloque se encarga de realizar la clasificación utilizando los valores que recibe de la capa *Flatten*. Para ello realiza los cálculos que determinan el valor de cada una de las neuronas de la última capa, que son las que definen la salida. Se usan 4 capas densas y 3 capas de *Dropout* entre ellas.

En el esquema de la página anterior, las capas densas están acompañadas de un valor que indica el número de neuronas de esa capa.

Todas las neuronas de las capas convolucionales y de las capas densas usan la función activación **ReLU**, cuya fórmula es $f(x) = \max(0, x)$ y tiene la siguiente forma:

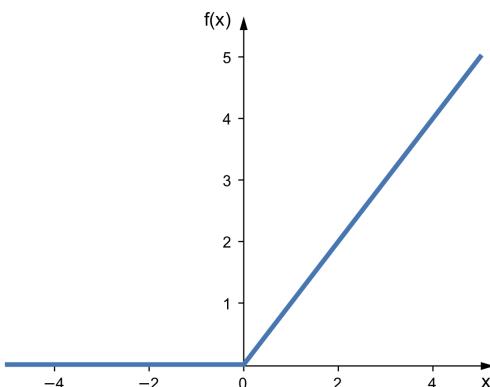


Figura 9: Función de activación ReLU

La función de optimización que se usa en el modelo es **adam**, que experimentalmente ofrece mejores resultados que *descenso del gradiente*, pues está basado en un ratio de aprendizaje adaptativo, a diferencia del ratio fijo que usa *descenso del gradiente*.

La función de coste va a ser **SparseCategoricalCrossentropy**, que calcula la entropía como la probabilidad entre dos distribuciones. Se utiliza generalmente en problemas con más de dos clases, como es el caso.

5. Evaluación

En primer lugar, se ha entrenado a la red durante **20 epochs** (vuelta completa a los datos).

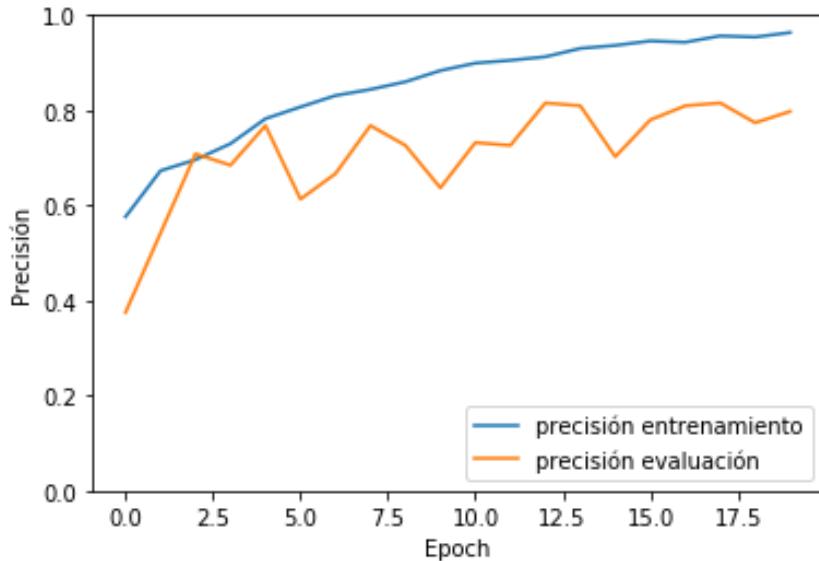


Figura 10: Gráfico de evaluación del modelo

Al finalizar el entrenamiento, la precisión alcanzada sobre el set de evaluación es **79.76 %** y la precisión alcanzada sobre el set de entrenamiento es **96.35 %**.

Como se ve en el gráfico anterior, la progresión sobre los datos de entrenamiento es exclusivamente creciente, mientras que no es estable en los datos de evaluación. Esto se debe a un claro *overfitting*. A pesar de esto, este es el modelo con mayor precisión que se ha conseguido, siendo bastante aceptable ya que clasifica correctamente el 80 % de las imágenes.

Las razones principales por las que se cree que se ha producido este *overfitting* son la necesidad de un mayor número de datos y la mala calidad de los datos que se tienen.

Algunos ejemplos de evaluación que se han obtenido son:

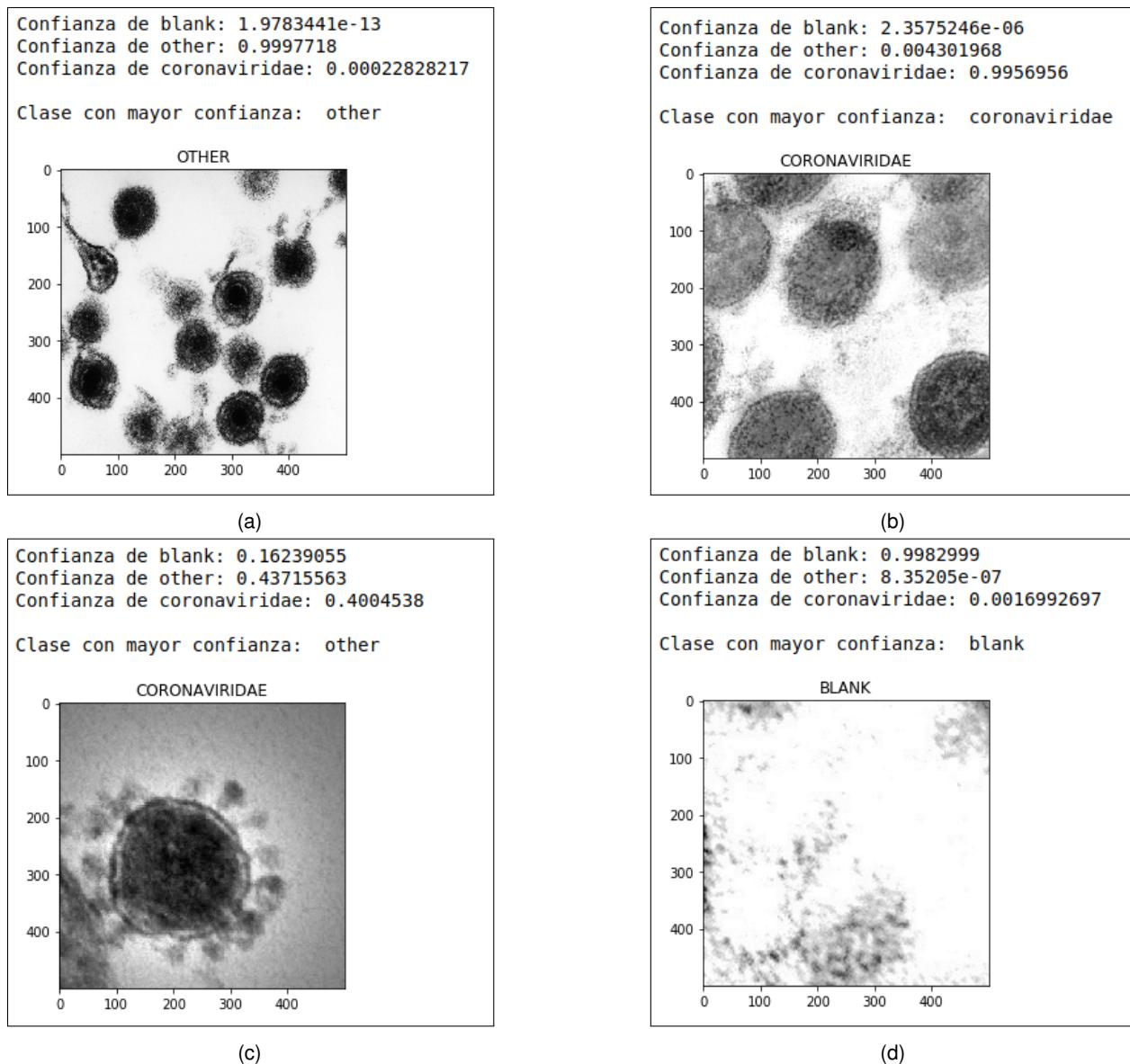


Figura 11: Imágenes de evaluación del modelo

6. Uso

El modelo actual no cumple con los requisitos necesarios para considerarlo terminado. Una vez se consigan más imágenes y se vuelva a entrenar a la red, si se obtienen resultados satisfactorios, el objetivo principal sería construir un servicio web que permita a los usuarios introducir imágenes de muestras víricas, tomadas por un microscopio TEM, de pacientes de los que se quiere saber si están infectados con el *COVID-19*. El sistema tendría que devolver la probabilidad de que la muestra pertenezca a cada una de las clases del problema, y una alerta o aviso en caso de que la red no consiga determinar con claridad la clase de la muestra.

Se piensa que la mejor forma de llegar al mayor número de usuarios es a través de un servicio web, ya que no requiere de ningún elemento adicional.

Además, podría considerarse útil permitir a usuarios mandar un *feedback* añadiendo nuevas imágenes de muestras, de forma que la red pueda reentrenarse con esta nueva información, y ser así cada vez más precisa.

7. Anexo

Para más información sobre el proyecto:

Diapositivas explicativas:

<https://drive.google.com/file/d/1AFAgb13CCIGQ49fmR0S3Yfys5A3NiyM2/view?usp=sharing>

Página creada en el *Hackathon* del proyecto:

<https://taiga.vencealvirus.software.imdea.org/project/danoloan-esquema-de-proyecto-renombrame/wiki/home>

PDF con el desarrollo y entrenamiento del modelo creado:

<https://github.com/akua21/Trabajo-Final-AA/blob/master/Memoria/Clasificador%20COVID-19/Clasificador%20COVID-19.pdf>

PDF con la evaluación del modelo:

<https://github.com/akua21/Trabajo-Final-AA/blob/master/Memoria/Evaluacion/Evaluacion.pdf>

8. Referencias

- TensorFlow CNN information:
<https://www.tensorflow.org/tutorials/images/cnn>, <https://www.tensorflow.org/tutorials/images/classification>
- CV-Tricks, Ankit Sachan:
<https://cv-tricks.com/tensorflow-tutorial/training-convolutional-neural-network-for-image-classification/>
- Medium, Connor Shorten:
<https://towardsdatascience.com/image-classification-python-keras-tutorial-kaggle-challenge-45a6332a58b8>
- Google:
<https://developers.google.com/machine-learning/glossary>