



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	5
3. Preproceso de Datos	6
4. Modelado	7
5. Evaluación	9
6. Uso	10

Índice de figuras

1. Muestra del virus <i>COVID-19</i>	3
2. Muestras víricas de ejemplo	5
3. Imágenes generadas por rotación	6
4. Imágenes generadas por volteo	6
5. Imágenes generadas por traslación	7
6. Imágenes generadas con ruido	7
7. Operación convolucional en una matriz 3x3 con un <i>kernel</i> de 2x2	8

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 deberá 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 podría asumir con cierta probabilidad que el paciente estaría infectado con el *COVID-19*.

Se ha pensado resolver este problema a través de técnicas de aprendizaje automático porque se trata de un problema de clasificación. Dado que las imágenes usadas para el entrenamiento tendrán una clase asociada, se podrán usar técnicas de **aprendizaje supervisado**. De esta forma, se reduciría el tiempo necesario para clasificar cada una de las muestras, porque eliminaría la necesidad de los especialistas de tener que revisar cada una de las muestras para comprobar si el paciente está o no infectado.

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.

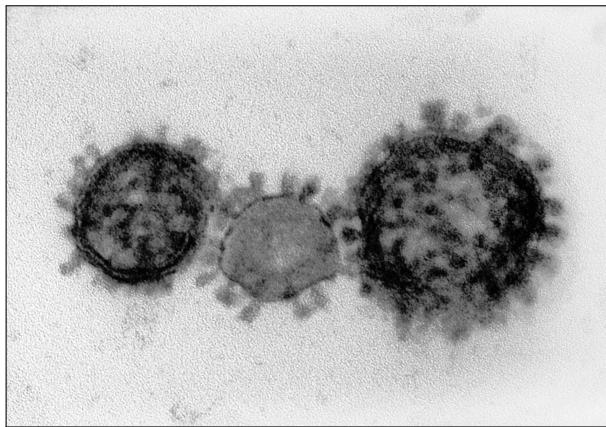


Figura 1: Muestra del virus *COVID-19*

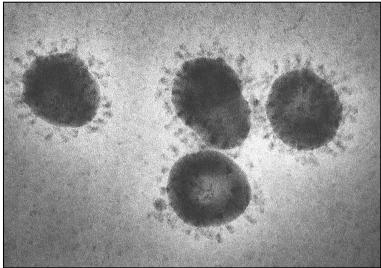
El proceso que habría que llevar a cabo sería:

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.

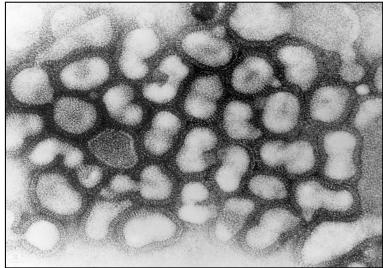
Dado que el conjunto de datos son imágenes, se va a usar una **red de neuronas** para realizar la tarea de aprendizaje automático. Esta red tendrá que ser capaz de diferenciar muestras con *COVID-19*, muestras con otros virus, y muestras vacías.

2. Datos y Obtención de Datos

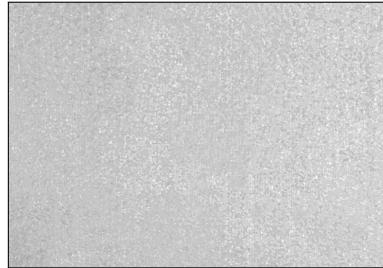
El conjunto de datos que se usan para entrenar y evaluar el modelo son imágenes de muestras víricas clasificadas. Estas imágenes son tomadas por el microscopio TEM, son en blanco y negro y de un tamaño variado. Estarán clasificadas en tres categorías: **COVID-19, otros virus y vacía**.



(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.

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, había ciertos *pixels* que se detectaban como formato *RGB* y por tanto se han transformado a **escala de grises**.

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

Después, se dividió 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).

Dado que el número de datos que se tienen es muy reducido, se va a optar por usar técnicas para aumentar el número de datos. Esto se hace para evitar el *overfitting*, dado que con un número pequeño de datos, es más fácil que se produzca.

El aumento de datos se realizará en todas las imágenes del set de entrenamiento, para aumentar el máximo posible el número de datos. En un principio, no se va a realizar un aumento al set de evaluación.

Las técnicas de aumento de datos que se van a utilizar son: **rotación**, **volteo**, **traslación** y **añadir ruido**.

Otras técnicas que serían también útiles pero que no se van a implementar a menos que sean necesarias son: ampliar o alejar la imagen, rotaciones no solo de 90º y mezclar varias técnicas.

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

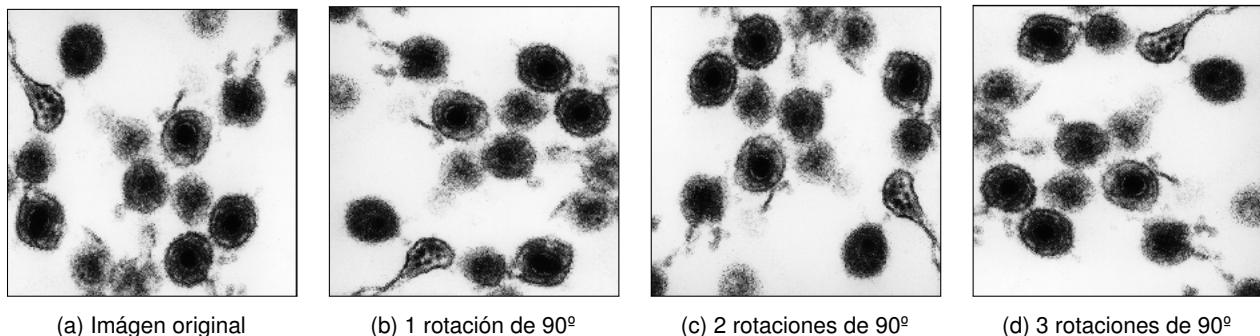


Figura 3: Imágenes generadas por **rotación**

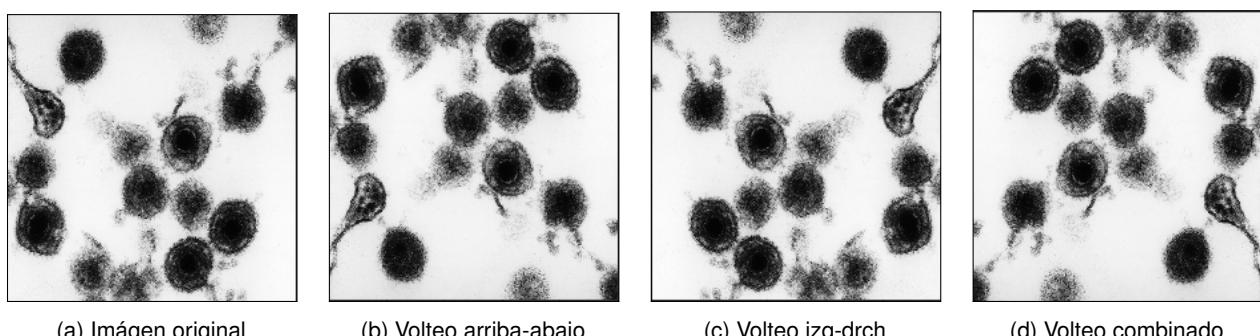
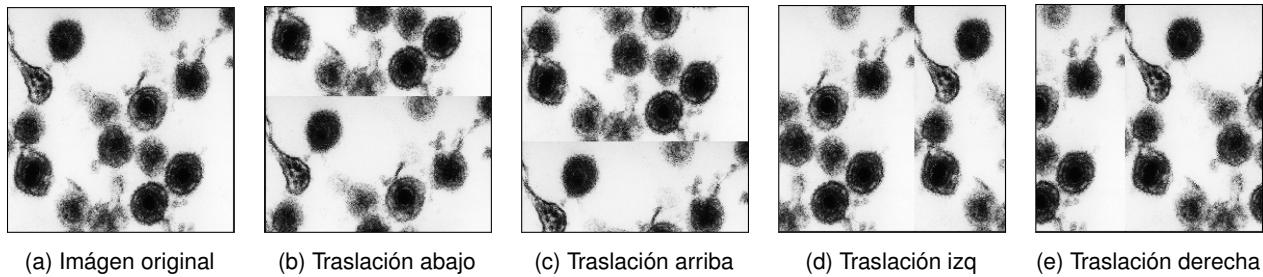
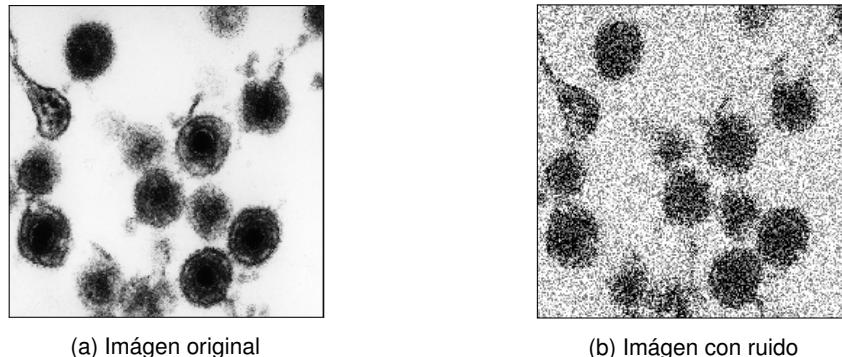


Figura 4: Imágenes generadas por **volteo**



(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 ruido

Figura 6: Imágenes generadas con **ruido**

4. Modelado

Como la tarea definida es una clasificación de imágenes, el algoritmo que se va a usar es una **red neuronal convolucional**. Esta es una red neuronal 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, útil en esta clasificación.

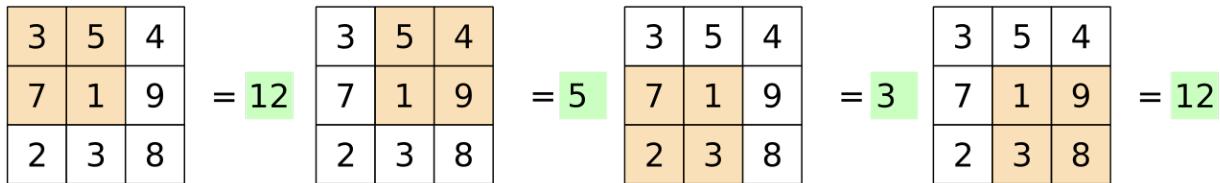
La arquitectura básica consiste en varias capas convolucionales y de *pooling* intercaladas, seguidas de una o varias capas densas.

- **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 van consiguiendo 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 fácil.
- **BatchNormalization:** 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*.

5. Evaluación

6. Uso