Clasificador Imágenes

April 8, 2020

1 Clasificador COVID-19

Este proyecto trata de resolver, mediante deep learning, la detección de virus de familia Coronaviridae. El objetivo es introducirle como entrada una muestra de microoscopio y comprobar si en la muestra se encuentra algún virus de la familia mencionada. Para ello, se realizará un clasificador de imágenes que obtenga como salida tres posibles clases: - Coronaviridae - Otros Virus - Muestra Vacía

En la primera clase, se trataría de una imagen en la que se detecte al menos un virus de la familia Coronaviridae. En la segunda, se detectarían otros virus, pero ninguno de la familia Coronaviridae. Y por último, se trataría de una muestra en la que no se detecte ningún tipo de Virus.

Acompañando a cada resultado, se adjuntará la fiabilidad del resultado.

Bibliotecas necesarias

[1]:
```
import tensorflow as tf

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Conv2D, Flatten, Dropout, MaxPooling2D
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from sklearn.model_selection import KFold

import os
import numpy as np
import matplotlib.pyplot as plt

import math
```

1.1 Constantes de la red neuronal

[2]:
```
batch_size = 10
epochs = 15
IMG_HEIGHT = 150
IMG_WIDTH = 150
classes = ["Muestra Vacía", "Coronaviridae", "Otros Virus"]
n_split = 3
```

1.2 Funciones para la visualización de datos

[3]:
```
def plotOneImage(image):
    plt.imshow(image)
    plt.axis('off')
    plt.tight_layout()
    plt.show()
```

[4]:
```
def plotFiveImages(images_arr):
    fig, axes = plt.subplots(1, 5, figsize=(20,20))
    axes = axes.flatten()
    for img, ax in zip( images_arr, axes):
        ax.imshow(img)
        ax.axis('off')
    plt.tight_layout()
    plt.show()
```

1.3 Carga de datos

Los datos usados son …

[5]:
```
PATH = os.path.join(os.getcwd(), 'COVID-19')

train_dir = os.path.join(PATH, 'photo_set')

blank_data = os.path.join(train_dir, 'blank')
virus_data = os.path.join(train_dir, 'other')
covid_data = os.path.join(train_dir, 'coronaviridae')
```

1.4 Entendiendo los datos

Mostrar el tamaño del dataset y algunas imágenes de los datos.

[6]:
```
num_blank = len(os.listdir(blank_data))
num_virus = len(os.listdir(virus_data))
num_covid = len(os.listdir(covid_data))

total_data = num_blank + num_virus + num_covid

print("Número de instancias de muestras en blanco: ", num_blank)
print("Número de instancias de muestras con otros virus: ", num_virus)
print("Número de instancias de coronavidae: ", num_covid)
print("---------------------------------------------")
print("Total de datos: ", total_data)
```

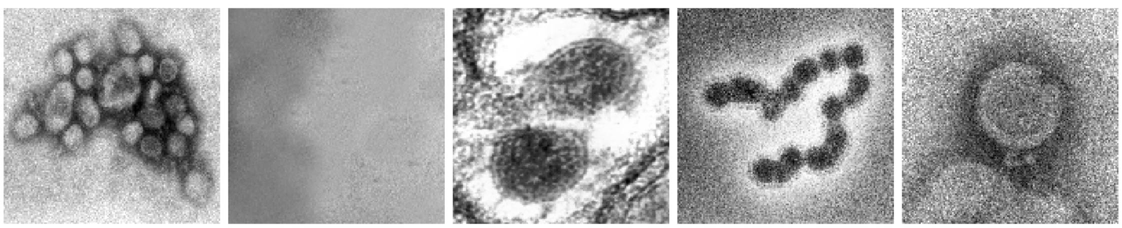
```
Número de instancias de muestras en blanco:  40
Número de instancias de muestras con otros virus:  46
Número de instancias de coronavidae:  40
```

```
---------------------------------------------
Total de datos:  126
```

[7]:
```
train_image_generator  = ImageDataGenerator(rescale=1./255)
train_data_generator = train_image_generator.
↪flow_from_directory(batch_size=batch_size,
                                                           directory=train_dir,
                                                          ␣
↪target_size=(IMG_HEIGHT, IMG_WIDTH),
                                                           )
```

```
Found 126 images belonging to 3 classes.
```

[8]:
```
train_images, _ = next(train_data_generator)

plotFiveImages(train_images[:5])
```



1.5 Crear el modelo de la red

El modelo consiste en 3 bloques de convolución, con un bloque denso (Completamente conectado) al final. Se usa relu. Entre los bloques de convolución, se usa maxpool, que sirve para el tratamiento de imágenes.

También se añade después de cada bloque un dropout, para evitar *overfitting*.

[9]:
```
train_image_generator  = ImageDataGenerator(rescale=1./255)
train_data_generator = train_image_generator.
↪flow_from_directory(batch_size=batch_size,
                                                           directory=train_dir,
                                                          ␣
↪target_size=(IMG_HEIGHT, IMG_WIDTH),
                                                           )
```

```
Found 126 images belonging to 3 classes.
```

[10]:
```
model = Sequential([
    Conv2D(16, 3, padding='same', activation='relu', input_shape=(IMG_HEIGHT,␣
↪IMG_WIDTH ,3)),
    MaxPooling2D(),
```

```
    Dropout(0.2),
    Conv2D(32, 3, padding='same', activation='relu'),
    MaxPooling2D(),
    Dropout(0.2),
    Conv2D(64, 3, padding='same', activation='relu'),
    MaxPooling2D(),
    Dropout(0.2),
    Flatten(),
    Dense(512, activation='relu'),
    Dense(3)
])
```

1.6 Compilar el modelo

[11]:
```
model.compile(optimizer='adam',
                  loss=tf.keras.losses.
 ↪CategoricalCrossentropy(from_logits=True),
                  metrics=['accuracy'])

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 150, 150, 16)      448
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 75, 75, 16)        0
_________________________________________________________________
dropout (Dropout)            (None, 75, 75, 16)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 75, 75, 32)        4640
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 37, 37, 32)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 37, 37, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 37, 37, 64)        18496
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 18, 18, 64)        0
_________________________________________________________________
dropout_2 (Dropout)          (None, 18, 18, 64)        0
_________________________________________________________________
flatten (Flatten)            (None, 20736)             0
_________________________________________________________________
dense (Dense)                (None, 512)               10617344
_________________________________________________________________
```

```
dense_1 (Dense)              (None, 3)                 1539
=================================================================
Total params: 10,642,467
Trainable params: 10,642,467
Non-trainable params: 0
_________________________________________________________________
```

1.7 Entrenar el modelo

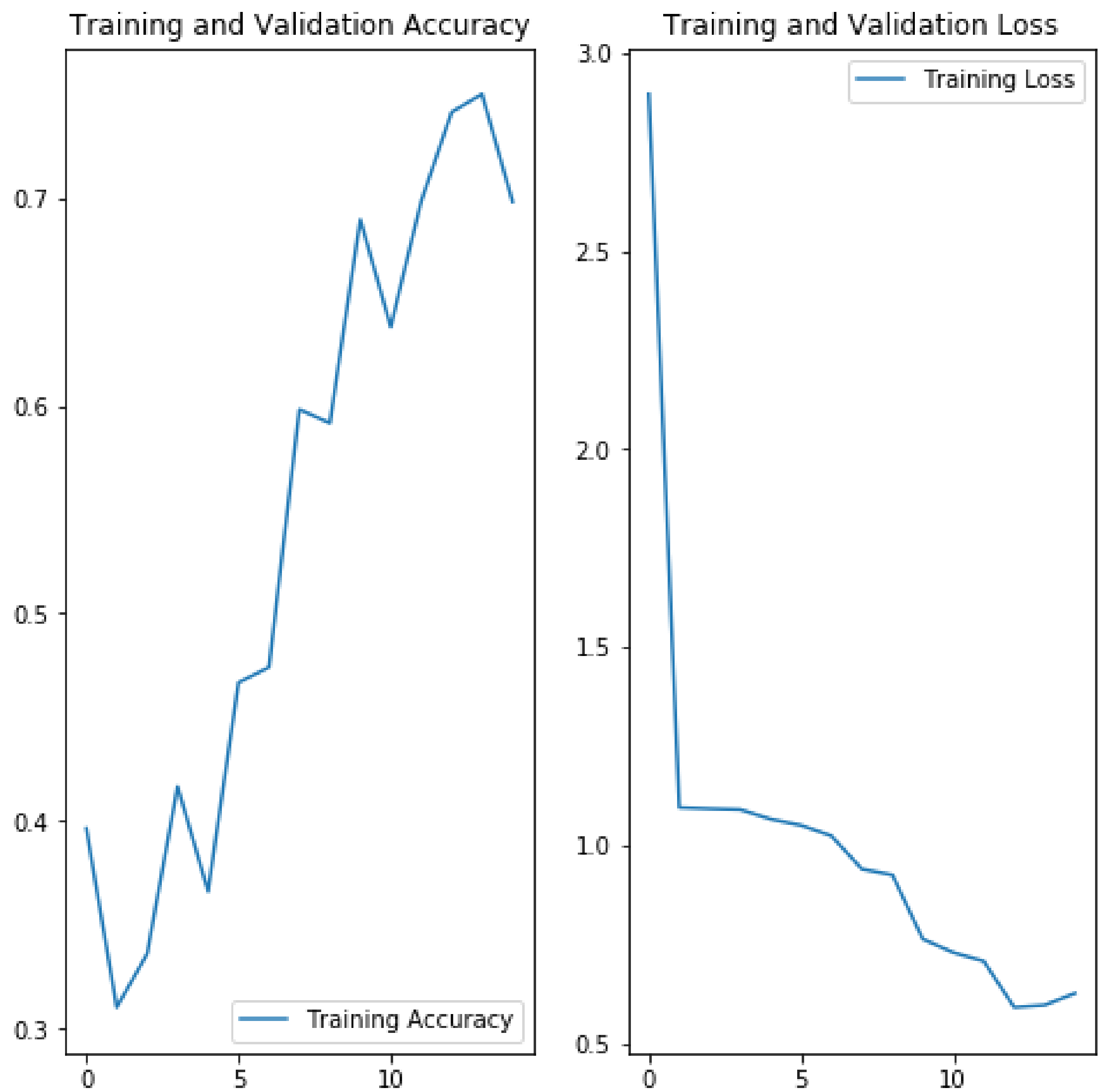
[12]:
```python
history = model.fit_generator(
    train_data_generator,
    steps_per_epoch=total_data // batch_size,
    epochs=epochs
)
```

```
Epoch 1/15
12/12 [==============================] - 4s 303ms/step - loss: 2.8348 -
accuracy: 0.3966
Epoch 2/15
12/12 [==============================] - 3s 269ms/step - loss: 1.0957 -
accuracy: 0.3103
Epoch 3/15
12/12 [==============================] - 3s 285ms/step - loss: 1.0913 -
accuracy: 0.3362
Epoch 4/15
12/12 [==============================] - 3s 271ms/step - loss: 1.0896 -
accuracy: 0.4167
Epoch 5/15
12/12 [==============================] - 3s 280ms/step - loss: 1.0652 -
accuracy: 0.3661
Epoch 6/15
12/12 [==============================] - 4s 353ms/step - loss: 1.0498 -
accuracy: 0.4667
Epoch 7/15
12/12 [==============================] - 3s 289ms/step - loss: 1.0289 -
accuracy: 0.4741
Epoch 8/15
12/12 [==============================] - 4s 302ms/step - loss: 0.9379 -
accuracy: 0.5982
Epoch 9/15
12/12 [==============================] - 4s 320ms/step - loss: 0.9246 -
accuracy: 0.5917
Epoch 10/15
12/12 [==============================] - 4s 299ms/step - loss: 0.7653 -
accuracy: 0.6897
Epoch 11/15
12/12 [==============================] - 5s 429ms/step - loss: 0.7334 -
accuracy: 0.6379
```

```
Epoch 12/15
12/12 [==============================] - 4s 330ms/step - loss: 0.7023 -
accuracy: 0.6983
Epoch 13/15
12/12 [==============================] - 4s 360ms/step - loss: 0.5827 -
accuracy: 0.7414
Epoch 14/15
12/12 [==============================] - 4s 307ms/step - loss: 0.5899 -
accuracy: 0.7500
Epoch 15/15
12/12 [==============================] - 4s 313ms/step - loss: 0.6345 -
accuracy: 0.6983
```

[13]:

```
acc = history.history['accuracy']

loss=history.history['loss']

epochs_range = range(epochs)

plt.figure(figsize=(8, 8))
plt.subplot(1, 2, 1)
plt.plot(epochs_range, acc, label='Training Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

plt.subplot(1, 2, 2)
plt.plot(epochs_range, loss, label='Training Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.show()
```



1.8 *Cross-validation*

El proceso de cross-validation se ha hecho a mano, dividiendo en 5 conjuntos.

[14]:
```
def KFold(train_set_dir, validation_set_dir):
    train_kfold_ima_gen = ImageDataGenerator(rescale=1./255)
    train_kfold_data_gen = train_kfold_ima_gen.
 ↪flow_from_directory(batch_size=batch_size,                                 ␣
 ↪directory=train_set_dir,                                                    ␣
 ↪target_size=(IMG_HEIGHT, IMG_WIDTH),
                                                                              )
```

```
validation_kfold_ima_gen = ImageDataGenerator(rescale=1./255)
validation_kfold_data_gen = validation_kfold_ima_gen.
↪flow_from_directory(batch_size=batch_size,
                                                                          ␣
↪directory=validation_set_dir,
                                                                          ␣
↪target_size=(IMG_HEIGHT, IMG_WIDTH),
                                                                           )

model_aux = Sequential([
    Conv2D(16, 3, padding='same', activation='relu',
           input_shape=(IMG_HEIGHT, IMG_WIDTH ,3)),
    MaxPooling2D(),
    Dropout(0.2),
    Conv2D(32, 3, padding='same', activation='relu'),
    MaxPooling2D(),
    Dropout(0.2),
    Conv2D(64, 3, padding='same', activation='relu'),
    MaxPooling2D(),
    Dropout(0.2),
    Flatten(),
    Dense(512, activation='relu'),
    Dense(3)
])

model_aux.compile(optimizer='adam',
              loss=tf.keras.losses.
↪CategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])

history = model.fit_generator(
    train_kfold_data_gen,
    steps_per_epoch=total_data // batch_size,
    epochs=epochs,
    validation_data=validation_kfold_data_gen,
    validation_steps=total_val // batch_size
)

acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

loss=history.history['loss']
val_loss=history.history['val_loss']

epochs_range = range(epochs)

plt.figure(figsize=(8, 8))
plt.subplot(1, 2, 1)
```

```
    plt.plot(epochs_range, acc, label='Training Accuracy')
    plt.plot(epochs_range, val_acc, label='Validation Accuracy')
    plt.legend(loc='lower right')
    plt.title('Training and Validation Accuracy')

    plt.subplot(1, 2, 2)
    plt.plot(epochs_range, loss, label='Training Loss')
    plt.plot(epochs_range, val_loss, label='Validation Loss')
    plt.legend(loc='upper right')
    plt.title('Training and Validation Loss')
    plt.show()

    return [acc[-1], val_acc[-1], loss[-1], val_loss[-1]]
```

[15]:
```
set1_dir = os.path.join(PATH, "0000-0009")
set2_dir = os.path.join(PATH, "0010-0019")
set3_dir = os.path.join(PATH, "0020-0029")
set4_dir = os.path.join(PATH, "0030-0039")
set5_dir = os.path.join(PATH, "0040-0049")

set1_train = os.path.join(set1_dir, "train")
set1_validation = os.path.join(set1_dir, "validation")

set2_train = os.path.join(set2_dir, "train")
set2_validation = os.path.join(set2_dir, "validation")

set3_train = os.path.join(set3_dir, "train")
set3_validation = os.path.join(set3_dir, "validation")

set4_train = os.path.join(set4_dir, "train")
set4_validation = os.path.join(set4_dir, "validation")

set5_train = os.path.join(set5_dir, "train")
set5_validation = os.path.join(set5_dir, "validation")

total_val = 30
```

[16]:
```
sets = [(set1_train, set1_validation), (set2_train, set2_validation),␣
↪(set3_train, set3_validation), (set4_train, set4_validation), (set5_train,␣
↪set5_validation)]

accuracy = 0
loss = 0


for i in range(4):
    values = KFold(sets[i][0], sets[i][1])
```

```
    accuracy += values[1]/4
    loss += values[3]/4
```

```
Found 98 images belonging to 3 classes.
Found 30 images belonging to 3 classes.
Epoch 1/15
12/12 [==============================] - 5s 456ms/step - loss: 0.5439 -
accuracy: 0.7627 - val_loss: 0.4210 - val_accuracy: 0.8667
Epoch 2/15
12/12 [==============================] - 5s 389ms/step - loss: 0.4586 -
accuracy: 0.7845 - val_loss: 0.3954 - val_accuracy: 0.8667
Epoch 3/15
12/12 [==============================] - 4s 370ms/step - loss: 0.5394 -
accuracy: 0.7333 - val_loss: 0.5447 - val_accuracy: 0.8667
Epoch 4/15
12/12 [==============================] - 5s 386ms/step - loss: 0.5017 -
accuracy: 0.8448 - val_loss: 0.7090 - val_accuracy: 0.6667
Epoch 5/15
12/12 [==============================] - 5s 447ms/step - loss: 0.4570 -
accuracy: 0.8220 - val_loss: 0.6551 - val_accuracy: 0.8000
Epoch 6/15
12/12 [==============================] - 5s 390ms/step - loss: 0.3369 -
accuracy: 0.8644 - val_loss: 0.6317 - val_accuracy: 0.8000
Epoch 7/15
12/12 [==============================] - 4s 359ms/step - loss: 0.3355 -
accuracy: 0.8644 - val_loss: 0.6787 - val_accuracy: 0.8000
Epoch 8/15
12/12 [==============================] - 4s 361ms/step - loss: 0.2337 -
accuracy: 0.9237 - val_loss: 0.8139 - val_accuracy: 0.8333
Epoch 9/15
12/12 [==============================] - 5s 397ms/step - loss: 0.2917 -
accuracy: 0.8621 - val_loss: 0.8657 - val_accuracy: 0.8000
Epoch 10/15
12/12 [==============================] - 5s 424ms/step - loss: 0.1685 -
accuracy: 0.9492 - val_loss: 0.7941 - val_accuracy: 0.7667
Epoch 11/15
12/12 [==============================] - 5s 376ms/step - loss: 0.0972 -
accuracy: 0.9831 - val_loss: 1.3441 - val_accuracy: 0.7667
Epoch 12/15
12/12 [==============================] - 4s 370ms/step - loss: 0.0791 -
accuracy: 0.9741 - val_loss: 1.1633 - val_accuracy: 0.7667
Epoch 13/15
12/12 [==============================] - 5s 441ms/step - loss: 0.0482 -
accuracy: 1.0000 - val_loss: 1.0947 - val_accuracy: 0.7667
Epoch 14/15
12/12 [==============================] - 4s 354ms/step - loss: 0.0299 -
accuracy: 0.9914 - val_loss: 1.3687 - val_accuracy: 0.7667
```

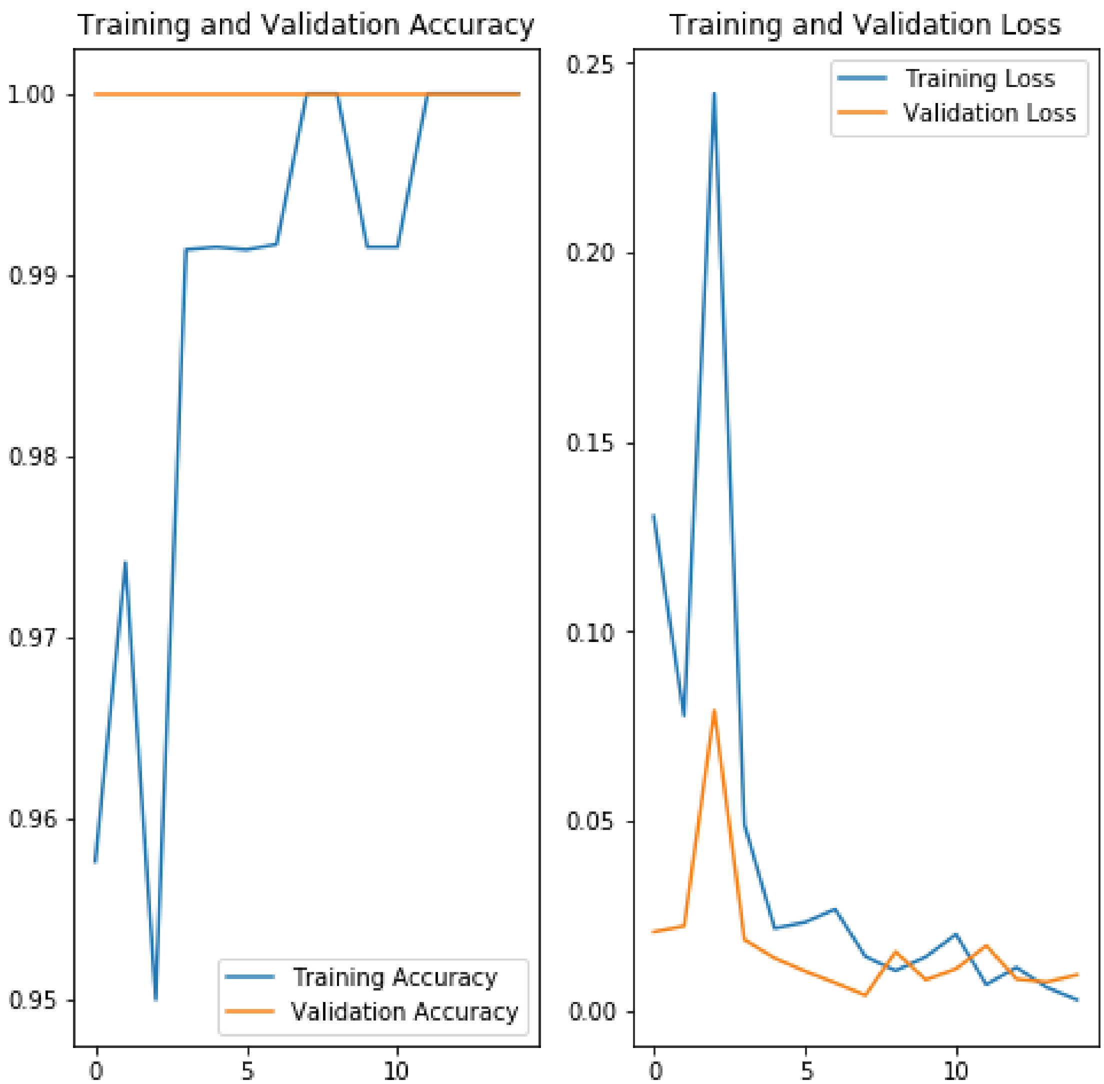
```
Epoch 15/15
12/12 [==============================] - 5s 429ms/step - loss: 0.0293 -
accuracy: 1.0000 - val_loss: 1.3569 - val_accuracy: 0.7667
```


```
Found 98 images belonging to 3 classes.
Found 30 images belonging to 3 classes.
Epoch 1/15
12/12 [==============================] - 5s 383ms/step - loss: 0.7788 -
accuracy: 0.8448 - val_loss: 0.7790 - val_accuracy: 0.7000
Epoch 2/15
12/12 [==============================] - 4s 335ms/step - loss: 0.5246 -
accuracy: 0.8083 - val_loss: 0.7272 - val_accuracy: 0.6667
Epoch 3/15
12/12 [==============================] - 4s 359ms/step - loss: 0.3224 -
```

```
accuracy: 0.8898 - val_loss: 0.6502 - val_accuracy: 0.7667
Epoch 4/15
12/12 [==============================] - 4s 350ms/step - loss: 0.3443 -
accuracy: 0.8448 - val_loss: 0.3445 - val_accuracy: 0.9333
Epoch 5/15
12/12 [==============================] - 5s 393ms/step - loss: 0.2852 -
accuracy: 0.8983 - val_loss: 0.7227 - val_accuracy: 0.7333
Epoch 6/15
12/12 [==============================] - 5s 391ms/step - loss: 0.1396 -
accuracy: 0.9576 - val_loss: 0.3006 - val_accuracy: 0.9333
Epoch 7/15
12/12 [==============================] - 4s 344ms/step - loss: 0.1073 -
accuracy: 0.9741 - val_loss: 0.2869 - val_accuracy: 0.9000
Epoch 8/15
12/12 [==============================] - 4s 361ms/step - loss: 0.0454 -
accuracy: 1.0000 - val_loss: 0.2740 - val_accuracy: 0.8667
Epoch 9/15
12/12 [==============================] - 4s 359ms/step - loss: 0.0320 -
accuracy: 1.0000 - val_loss: 0.2859 - val_accuracy: 0.9000
Epoch 10/15
12/12 [==============================] - 4s 374ms/step - loss: 0.0495 -
accuracy: 0.9915 - val_loss: 0.3879 - val_accuracy: 0.8667
Epoch 11/15
12/12 [==============================] - 4s 367ms/step - loss: 0.0661 -
accuracy: 0.9831 - val_loss: 0.2503 - val_accuracy: 0.8667
Epoch 12/15
12/12 [==============================] - 4s 357ms/step - loss: 0.0244 -
accuracy: 1.0000 - val_loss: 0.2428 - val_accuracy: 0.8333
Epoch 13/15
12/12 [==============================] - 4s 359ms/step - loss: 0.0419 -
accuracy: 0.9915 - val_loss: 0.2599 - val_accuracy: 0.8667
Epoch 14/15
12/12 [==============================] - 4s 355ms/step - loss: 0.0202 -
accuracy: 0.9915 - val_loss: 0.2540 - val_accuracy: 0.8333
Epoch 15/15
12/12 [==============================] - 4s 352ms/step - loss: 0.0184 -
accuracy: 1.0000 - val_loss: 0.2605 - val_accuracy: 0.8667
```

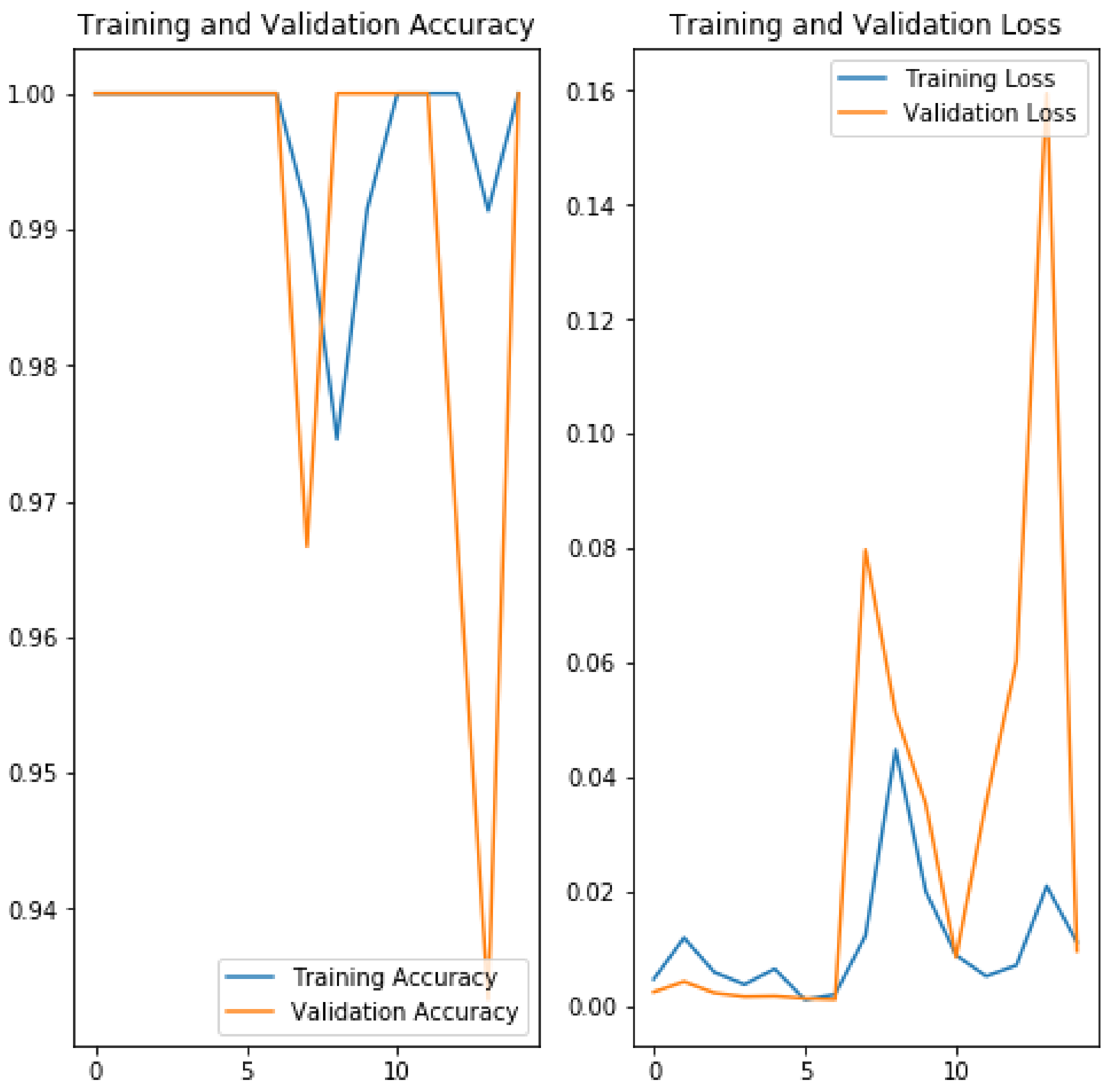

```
Found 98 images belonging to 3 classes.
Found 30 images belonging to 3 classes.
Epoch 1/15
12/12 [==============================] - 5s 403ms/step - loss: 0.1296 -
accuracy: 0.9576 - val_loss: 0.0207 - val_accuracy: 1.0000
Epoch 2/15
12/12 [==============================] - 5s 382ms/step - loss: 0.0771 -
accuracy: 0.9741 - val_loss: 0.0222 - val_accuracy: 1.0000
Epoch 3/15
12/12 [==============================] - 4s 355ms/step - loss: 0.2420 -
accuracy: 0.9500 - val_loss: 0.0791 - val_accuracy: 1.0000
Epoch 4/15
12/12 [==============================] - 4s 373ms/step - loss: 0.0485 -
accuracy: 0.9914 - val_loss: 0.0186 - val_accuracy: 1.0000
```

```
Epoch 5/15
12/12 [==============================] - 5s 389ms/step - loss: 0.0218 -
accuracy: 0.9915 - val_loss: 0.0137 - val_accuracy: 1.0000
Epoch 6/15
12/12 [==============================] - 5s 399ms/step - loss: 0.0227 -
accuracy: 0.9914 - val_loss: 0.0103 - val_accuracy: 1.0000
Epoch 7/15
12/12 [==============================] - 6s 479ms/step - loss: 0.0266 -
accuracy: 0.9917 - val_loss: 0.0072 - val_accuracy: 1.0000
Epoch 8/15
12/12 [==============================] - 5s 385ms/step - loss: 0.0140 -
accuracy: 1.0000 - val_loss: 0.0039 - val_accuracy: 1.0000
Epoch 9/15
12/12 [==============================] - 5s 407ms/step - loss: 0.0104 -
accuracy: 1.0000 - val_loss: 0.0154 - val_accuracy: 1.0000
Epoch 10/15
12/12 [==============================] - 6s 470ms/step - loss: 0.0139 -
accuracy: 0.9915 - val_loss: 0.0080 - val_accuracy: 1.0000
Epoch 11/15
12/12 [==============================] - 5s 385ms/step - loss: 0.0200 -
accuracy: 0.9915 - val_loss: 0.0109 - val_accuracy: 1.0000
Epoch 12/15
12/12 [==============================] - 5s 453ms/step - loss: 0.0069 -
accuracy: 1.0000 - val_loss: 0.0170 - val_accuracy: 1.0000
Epoch 13/15
12/12 [==============================] - 7s 594ms/step - loss: 0.0112 -
accuracy: 1.0000 - val_loss: 0.0082 - val_accuracy: 1.0000
Epoch 14/15
12/12 [==============================] - 6s 493ms/step - loss: 0.0063 -
accuracy: 1.0000 - val_loss: 0.0074 - val_accuracy: 1.0000
Epoch 15/15
12/12 [==============================] - 5s 438ms/step - loss: 0.0027 -
accuracy: 1.0000 - val_loss: 0.0093 - val_accuracy: 1.0000
```




```
Found 98 images belonging to 3 classes.
Found 30 images belonging to 3 classes.
Epoch 1/15
12/12 [==============================] - 5s 439ms/step - loss: 0.0046 -
accuracy: 1.0000 - val_loss: 0.0024 - val_accuracy: 1.0000
Epoch 2/15
12/12 [==============================] - 5s 376ms/step - loss: 0.0123 -
accuracy: 1.0000 - val_loss: 0.0043 - val_accuracy: 1.0000
Epoch 3/15
12/12 [==============================] - 5s 381ms/step - loss: 0.0058 -
accuracy: 1.0000 - val_loss: 0.0023 - val_accuracy: 1.0000
Epoch 4/15
12/12 [==============================] - 5s 424ms/step - loss: 0.0037 -
accuracy: 1.0000 - val_loss: 0.0016 - val_accuracy: 1.0000
```

```
Epoch 5/15
12/12 [==============================] - 6s 469ms/step - loss: 0.0064 -
accuracy: 1.0000 - val_loss: 0.0017 - val_accuracy: 1.0000
Epoch 6/15
12/12 [==============================] - 6s 509ms/step - loss: 0.0011 -
accuracy: 1.0000 - val_loss: 0.0013 - val_accuracy: 1.0000
Epoch 7/15
12/12 [==============================] - 7s 609ms/step - loss: 0.0020 -
accuracy: 1.0000 - val_loss: 0.0011 - val_accuracy: 1.0000
Epoch 8/15
12/12 [==============================] - 6s 522ms/step - loss: 0.0119 -
accuracy: 0.9914 - val_loss: 0.0797 - val_accuracy: 0.9667
Epoch 9/15
12/12 [==============================] - 6s 459ms/step - loss: 0.0442 -
accuracy: 0.9746 - val_loss: 0.0511 - val_accuracy: 1.0000
Epoch 10/15
12/12 [==============================] - 4s 364ms/step - loss: 0.0198 -
accuracy: 0.9915 - val_loss: 0.0352 - val_accuracy: 1.0000
Epoch 11/15
12/12 [==============================] - 5s 453ms/step - loss: 0.0087 -
accuracy: 1.0000 - val_loss: 0.0085 - val_accuracy: 1.0000
Epoch 12/15
12/12 [==============================] - 6s 459ms/step - loss: 0.0051 -
accuracy: 1.0000 - val_loss: 0.0356 - val_accuracy: 1.0000
Epoch 13/15
12/12 [==============================] - 5s 395ms/step - loss: 0.0070 -
accuracy: 1.0000 - val_loss: 0.0603 - val_accuracy: 0.9667
Epoch 14/15
12/12 [==============================] - 5s 410ms/step - loss: 0.0241 -
accuracy: 0.9914 - val_loss: 0.1595 - val_accuracy: 0.9333
Epoch 15/15
12/12 [==============================] - 5s 431ms/step - loss: 0.0110 -
accuracy: 1.0000 - val_loss: 0.0096 - val_accuracy: 1.0000
```



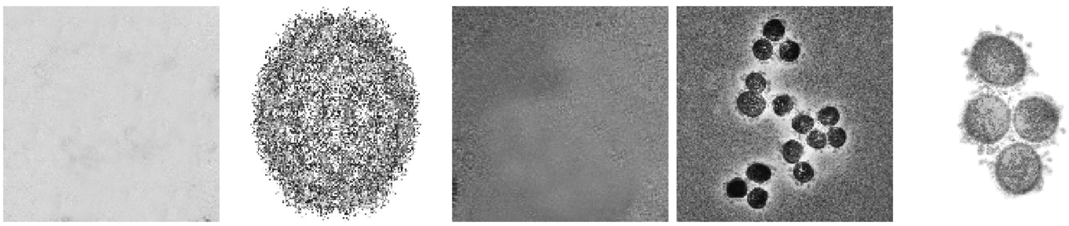
[17]:
```
print("Precisión del modelo generado: " + str(accuracy*100) + "%" )
print("Función de coste del modelo generado: " + str(loss))
```

```
Precisión del modelo generado: 90.83333313465118%
Función de coste del modelo generado: 0.40907337464159355
```

[18]:
```
test_dir = os.path.join(PATH, "test")

test_image_generator = ImageDataGenerator(rescale=1./255)

test_image = test_image_generator.flow_from_directory(batch_size=15,
                                                       directory=test_dir,
```

```
                                                          target_size=(IMG_HEIGHT,␣
↪IMG_WIDTH),
                                                          )

sample_test_images, sample_test_classes = next(test_image)
plotFiveImages(sample_test_images[:5])

probability_model = tf.keras.Sequential([model, tf.keras.layers.Softmax()])
result = probability_model.predict(test_image)
```

```
Found 5 images belonging to 3 classes.
```



[19]:
```
predictions = []
confidence = []

for i in result:
    confidence.append(max(i))
    predictions.append(np.argmax(i))
```

[20]:
```
print(predictions)
print(confidence)
```

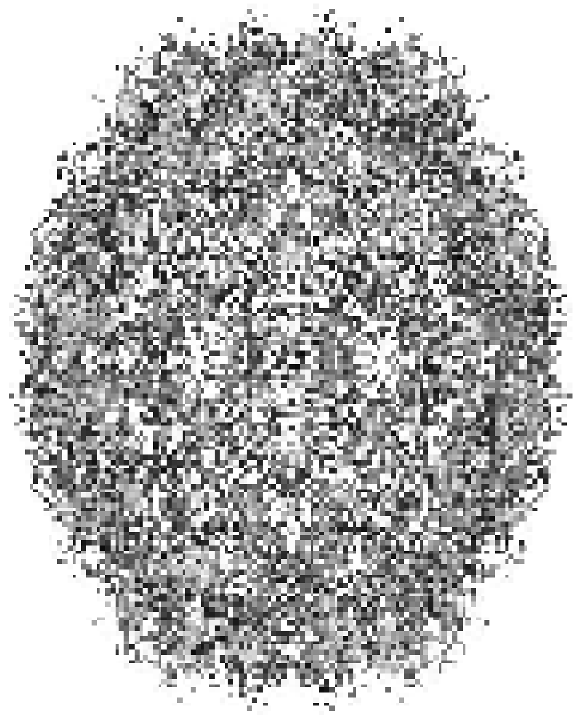
```
[0, 2, 0, 1, 1]
[0.97560775, 1.0, 0.9987023, 0.9999914, 0.9900417]
```

[21]:
```
for i in range(5):
    real = ""
    predict = ""
    if sample_test_classes[i][0] == 1:
        print("Clase real de la muestra " + str(i) +  " : " + classes[0])
        real = classes[0]
    if sample_test_classes[i][1] == 1:
        print("Clase real de la muestra " + str(i) +  " : " + classes[1])
        real = classes[1]
    if sample_test_classes[i][2] == 1:
        print("Clase real de la muestra " + str(i) +  " : " + classes[2])
        real = classes[2]
```

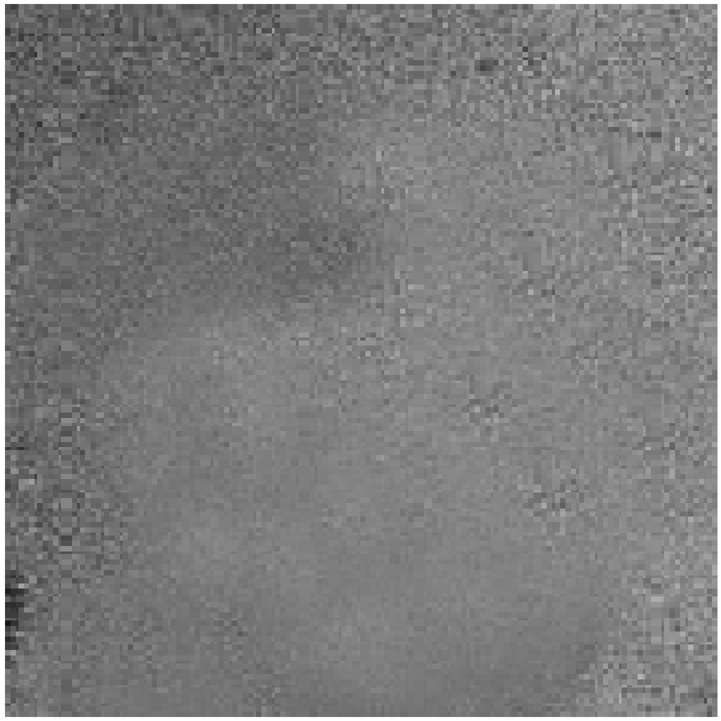
```
    if predictions[i] == 0:
        print("Clase predecida de la muestra " +  str(i) + " : " + classes[0])
        predict = classes[0]
    if predictions[i] == 1:
        print("Clase predecida de la muestra " +  str(i) + " : " +classes[1])
        predict = classes[1]
    if predictions[i] == 2:
        print("Clase predecida de la muestra " +  str(i) + " : " +classes[2])
        predict = classes[2]

    print("Con una confianza de " + str(confidence[i]))

    if real == predict:
        print("ACIERTO")
    else:
        print("ERROR")

    plotOneImage(sample_test_images[i])
```

```
Clase real de la muestra 0 : Muestra Vacía
Clase predecida de la muestra 0 : Muestra Vacía
Con una confianza de 0.97560775
ACIERTO
```

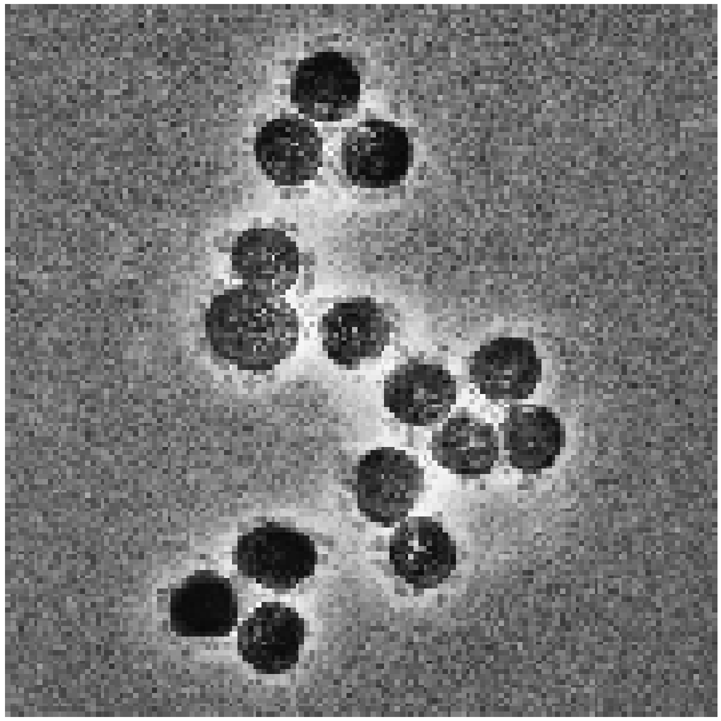
```
Clase real de la muestra 1 : Otros Virus
Clase predecida de la muestra 1 : Otros Virus
Con una confianza de 1.0
ACIERTO
```



```
Clase real de la muestra 2 : Muestra Vacía
Clase predecida de la muestra 2 : Muestra Vacía
Con una confianza de 0.9987023
ACIERTO
```



```
Clase real de la muestra 3 : Coronaviridae
Clase predecida de la muestra 3 : Coronaviridae
Con una confianza de 0.9999914
ACIERTO
```



```
Clase real de la muestra 4 : Coronaviridae
Clase predecida de la muestra 4 : Coronaviridae
Con una confianza de 0.9900417
ACIERTO
```