UB-DataScience-CapstoneProject

Introducción al Data Science y al Machine Learning. http://www.ub.edu/datascience/postgraduate/

View the Project on GitHub at https://github.com/

Capstone Project - Cápsula endoscópica

Introducción

Metodología

Datos

Resultados

Conclusiones - Perspectivas

Bibliografía

Contacto

Proyecto presentado por **Lucas Martínez Rodrigo**. El código fuente para reproducir estos experimentos se encuentra disponible en github.

Barcelona, 30 de junio de 2022

Project maintained by **lumaro77**

Hosted on GitHub Pages — Theme by **orderedlist**

Introduccion

La red neuronal

Las neuronas son estructuras biológicas del sistema nervioso capaz de enviar mensajes electroquímicos a través del cuerpo. Las neuronas se catacterizan por tener una entrada y una salida o muchas de ellas. Las neuronas son capaces de analizar las entradas y enviar dicho análisis en un mensaje por sus salidas. Es decir, una neurona toma decisiones basadas en la información disponible en sus inputs.

Las neuronas se interconectan formando redes, donde cada una tiene una función específica: las primeras reciben datos del exterior, las neuronas ocultas hacen cálculos y procesos internos, mientras que las últimas ofrecen las respuestas al exterior. Las neuronas ocultas aprenden gracias a ejemplos cuya respuesta se conoce, mediante el entrenamiento. Tras éste, son capaces de contestar también a las preguntas sobre nuevos casos que nunca vieron previamente.

Un conjunto grande de neuronas es capaz de aprender a resolver problemas complejos. Por esto en el Deep Learning existe una rama denominada redes neuronales artificiales que tomando como base el concepto de red neuronal desarrolla algoritmos capaces de aprender de una manera similar a las redes neuronales.

Redes neuronales artificiales

Los orígenes de las redes neuronales se situan en la mitad del siglo XX, pero resurgió con intensidad a principios del XXI gracias a nuevos trabajos y a la capacidad de los ordenadores y las GPUs para procesar datos de manera masiva.

Las redes neuronales son sistemas basados en combinaciones lineales a la que se aplica una función de activación que le dota de un caracter no lineal. Los coeficientes de dichas combinaciones lineales se ajustan de manera iterativa con el conocido método de “back propagation” que mejora los parámetros en función de su gradiente.

Las redes neuronales se agrupan por capas, que se relacionan entre ellas. Hay diversos tipos de capas de redes neuronales entre las que destacan las densas y las convolucionales. Las neuronas de una capa densa actúan como se ha descrito previamente, mientras que las capas convolucionales están formadas por filtros de ventana flotante que se desplazan sobre una imagen para obtener alguna característica geométrica 2D o característica tonal de la misma. Un conjunto de capas convolucionales aspira a caracterizar la mayoría de rasgos que el sistema visual humano observa y reconoce en una imagen.

Frameworks para redes neuronales

La expansión y generalización del lenguaje de programación Python y de sus extensiones para cálculo científico, fue un verdadero terreno abonado para la aparición de frameworks expecíficos de Data Science y en particular de redes neuronales o inteligencia artificial. Entre las múltiples propuestas que se encuentran en la bibliografía, destaca a criterio del autor TensorFlow y su API Keras.

Por otro lado, herramientas y paradigmas como los Jupyter-Notebooks, así como los servicios de cálculo en línea como los ofrecidos por Google-Colab, han democratizado el acceso a la vez que extendido el uso y aplicaciones de la inteligencia artificial tanto para problemas clásicos como para nuevos desafíos.

Cápsula endoscópica

Uno de los campos de aplicación de cualquier desarrollo tecnológico es el ámbito de la salud. La miríada de pruebas diagnósticas que tienen por resultado una imagen o serie de imágenes crece de manera ingente cada año. El análisis automático o asistido de dichas imágenes tiene una indudable utilidad por lo que centra el interés de muchos grupos de trabajo.

En el marco anterior, existe un trabajo para la elaboración de un repositorio de imágenes endoscópicas del sistema digestivo humano que constituye el punto de partida de este Capstone Project. Así se pretende hacer una prospección de la capacidad de las redes neuronales convolucionales para la identificación de lesiones y regiones características en imágenes medicas de dicho dataset.

En los siguientes apartados de detalla la metodología empleada y se describe el dataset. A continuación se muestran los resultados y las conclusiones de este trabajo.

Atrás – Índice – Siguiente

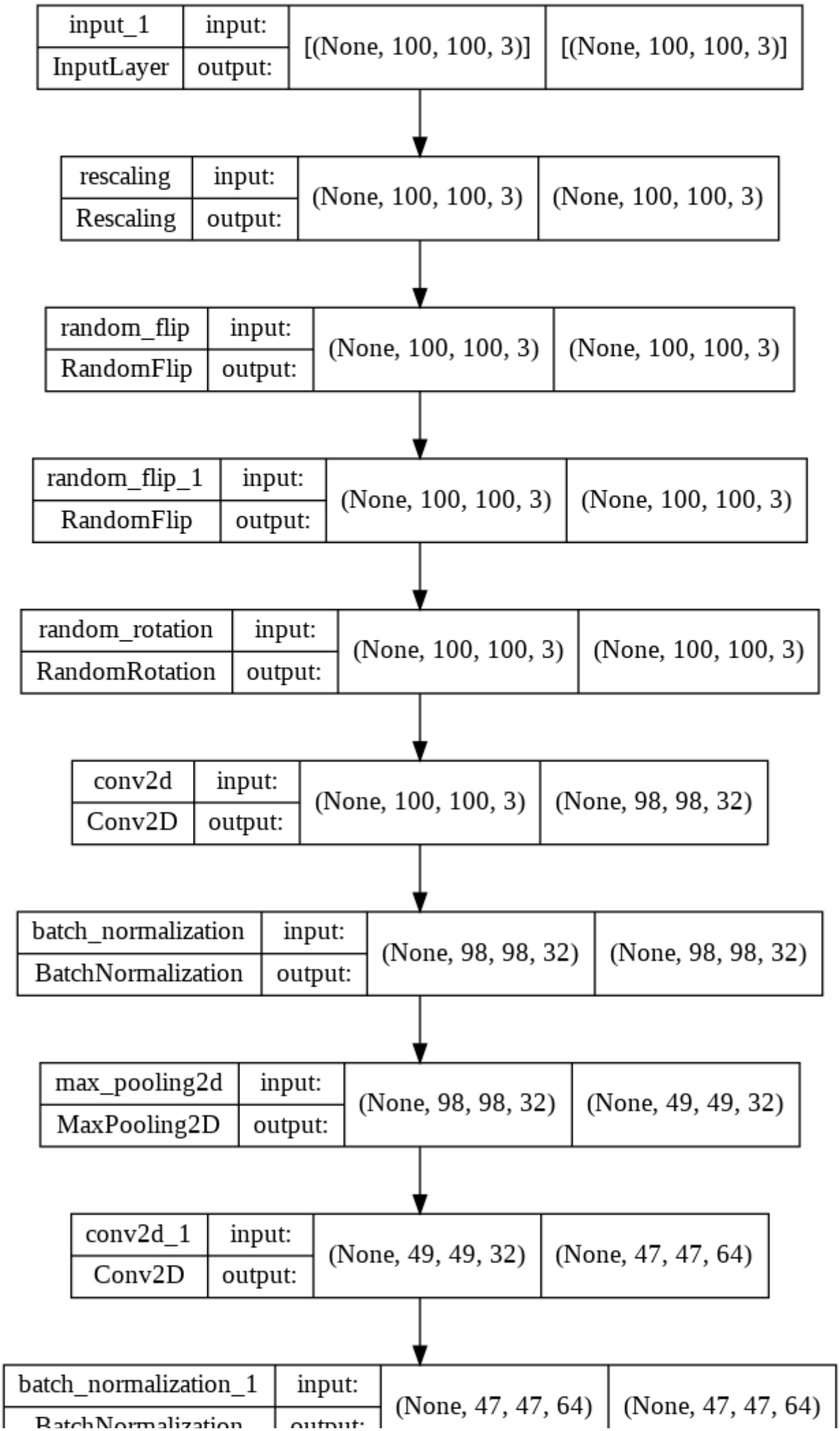
Project maintained by **lumaro77**
Hosted on GitHub Pages — Theme by **orderedlist**

Metodología

A continuación se describen las redes neuronales empleadas y principales características para su uso. El framework sobre el que se ha implementado es Keras, como API de Tensorflow. Los modelos se construyen sobre el sistema Sequential, que permite integrar al modelo la mayoría de elementos (capas de neuronas, capas de normalización, aumentación de imágenes, etc.).

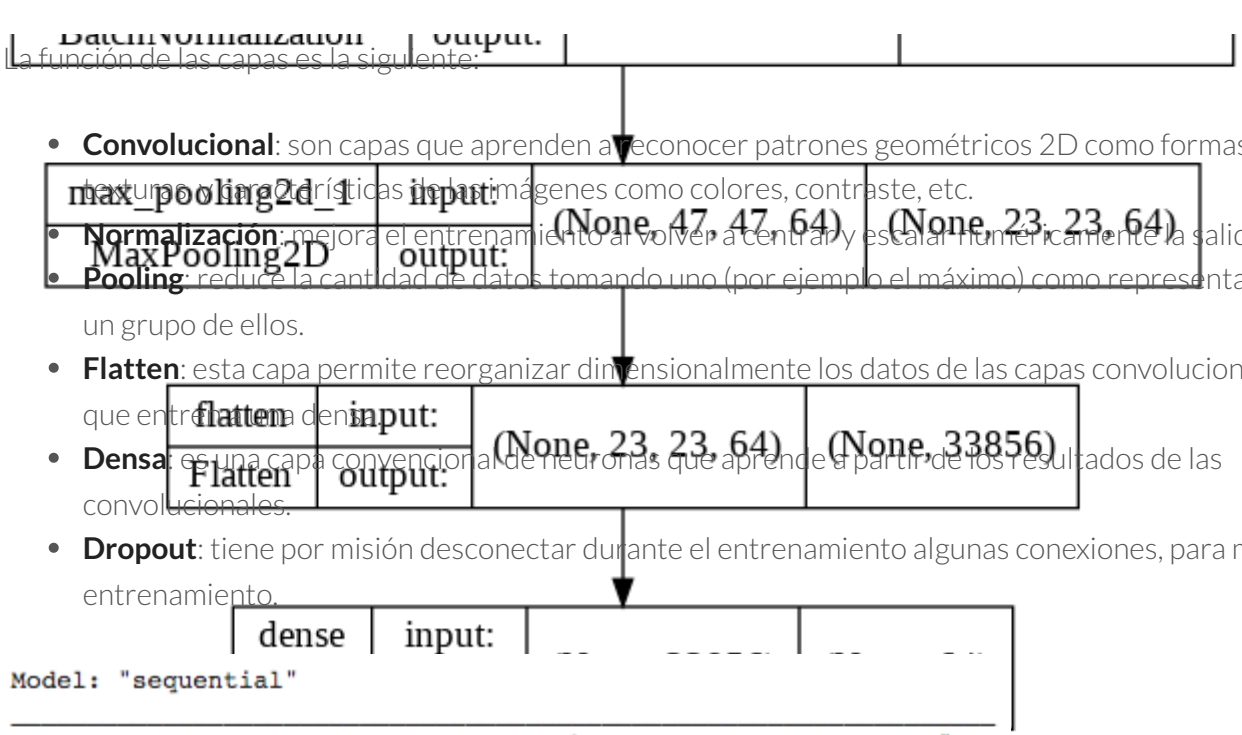
Red convolucional simple ConvoRGB

El primer modelo empleado, que denominaré, ConvoRGB está formado por dos capas convolucionales, seguidas de normalización y pooling. A continuación se realiza un flatten y se pasa por una capa densa para alcanzar la capa de salida a través de un dropout.


input_1
InputLayer
input:
output:
[(None, 100, 100, 3)]
[(None, 100, 100, 3)]
rescaling
Rescaling
input:
output:
(None, 100, 100, 3)
(None, 100, 100, 3)
random_flip
RandomFlip
input:
output:
(None, 100, 100, 3)
(None, 100, 100, 3)
random_flip_1
RandomFlip
input:
output:
(None, 100, 100, 3)
(None, 100, 100, 3)
random_rotation
RandomRotation
input:
output:
(None, 100, 100, 3)
(None, 100, 100, 3)
conv2d
Conv2D
input:
output:
(None, 100, 100, 3)
(None, 98, 98, 32)
batch_normalization
BatchNormalization
input:
output:
(None, 98, 98, 32)
(None, 98, 98, 32)
max_pooling2d
MaxPooling2D
input:
output:
(None, 98, 98, 32)
(None, 49, 49, 32)
conv2d_1
Conv2D
input:
output:
(None, 49, 49, 32)
(None, 47, 47, 64)
batch_normalization_1
input:
(None, 47, 47, 64)
(None, 47, 47, 64)

La función de las capas es la siguiente:

- **Convolucional**: son capas que aprenden a reconocer patrones geométricos 2D como formas y texturas y características de las imágenes como colores, contraste, etc.
- **Normalización**: mejora el entrenamiento al volver a centrar y escalar numéricamente la salida.
- **Pooling**: reduce la cantidad de datos tomando uno (por ejemplo el máximo) como representación de un grupo de ellos.
- **Flatten**: esta capa permite reorganizar dimensionalmente los datos de las capas convolucionales para que entren en la densa.
- **Densa**: es una capa convencional de neuronas que aprende a partir de los resultados de las convolucionales.
- **Dropout**: tiene por misión desconectar durante el entrenamiento algunas conexiones, para mejorar el entrenamiento.




```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 rescaling (Rescaling)       (None, 100, 100, 3)       0

 random_flip (RandomFlip)    (None, 100, 100, 3)       0

 random_flip_1 (RandomFlip)  (None, 100, 100, 3)       0

 random_rotation (RandomRota  (None, 100, 100, 3)      0
 tion)

 conv2d (Conv2D)             (None, 98, 98, 32)        896

 batch_normalization (BatchN  (None, 98, 98, 32)       128
 ormalization)

 max_pooling2d (MaxPooling2D  (None, 49, 49, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 47, 47, 64)        18496

 batch_normalization_1 (Batc  (None, 47, 47, 64)       256
 hNormalization)

 max_pooling2d_1 (MaxPooling  (None, 23, 23, 64)       0
 2D)

 flatten (Flatten)           (None, 33856)             0

 dense (Dense)               (None, 64)                2166848

 dropout (Dropout)           (None, 64)                0

 dense_1 (Dense)             (None, 11)                715

=================================================================
Total params: 2,187,339
Trainable params: 2,187,147
Non-trainable params: 192
_________________________________________________________________
```

Como puede observarse la red ConvoRGB tiene algo más de 2 millones de parámetros para ser entrenados, dado que no hay inicialización previa (el sistema asigna valores aleatorios a los mismos antes del

entrenamiento).

Transferencia de aprendizaje

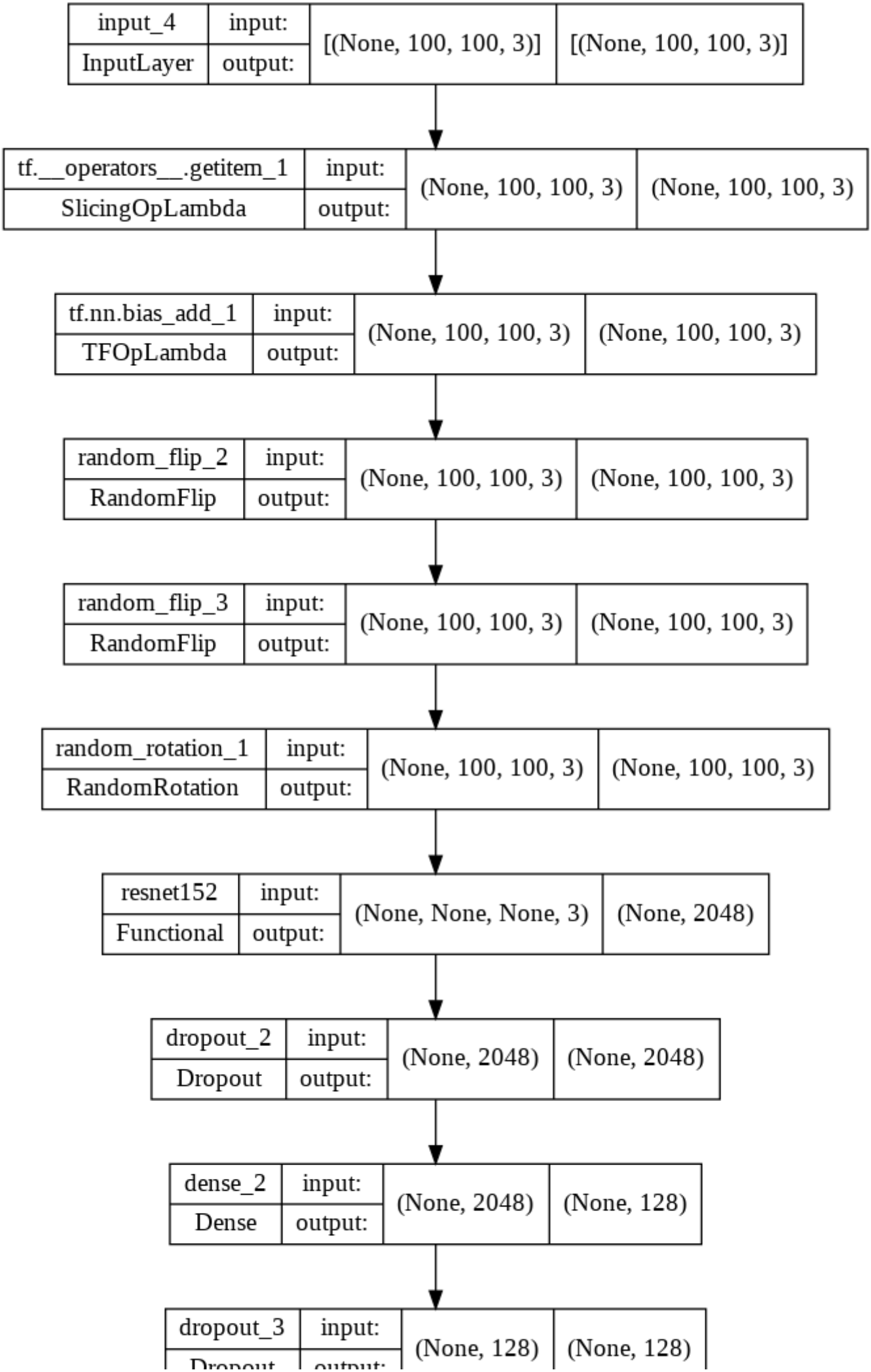
La segunda opción planteada será el uso de redes pre-entrenadas, adaptadas al número de clases concreto de nuestro estudio. Esta técnica permite el uso de redes mucho más complejas que la anterior entrenadas con bases de datos de imágenes ingentes como la Imagenet (14 millones de imágenes etiquetadas en 1.000 clases).

En este caso emplearé un par de redes que han sido usadas previamente para trabajos similares con el dataset problema. Son la ResNet152 y la DenseNet169.

La principal ventaja de esta técnica es permitir el uso de redes complejas, para cuyo entrenamiento no sería suficiente con el dataset problema. En definitiva es un recurso para tratar de mejorar los resultados con un bajo coste computacional.

Transferencia de aprendizaje con ResNet

La ResNet es una red neuronal de tipo residual cuya principal característica es que permite conexiones o saltos sobre algunas capas, que pueden ser dobles o triples.


input_4 | input: | [(None, 100, 100, 3)] | [(None, 100, 100, 3)]
InputLayer | output:
tf.__operators__.getitem_1 | input: | (None, 100, 100, 3) | (None, 100, 100, 3)
SlicingOpLambda | output:
tf.nn.bias_add_1 | input: | (None, 100, 100, 3) | (None, 100, 100, 3)
TFOpLambda | output:
random_flip_2 | input: | (None, 100, 100, 3) | (None, 100, 100, 3)
RandomFlip | output:
random_flip_3 | input: | (None, 100, 100, 3) | (None, 100, 100, 3)
RandomFlip | output:
random_rotation_1 | input: | (None, 100, 100, 3) | (None, 100, 100, 3)
RandomRotation | output:
resnet152 | input: | (None, None, None, 3) | (None, 2048)
Functional | output:
dropout_2 | input: | (None, 2048) | (None, 2048)
Dropout | output:
dense_2 | input: | (None, 2048) | (None, 128)
Dense | output:
dropout_3 | input: | (None, 128) | (None, 128)

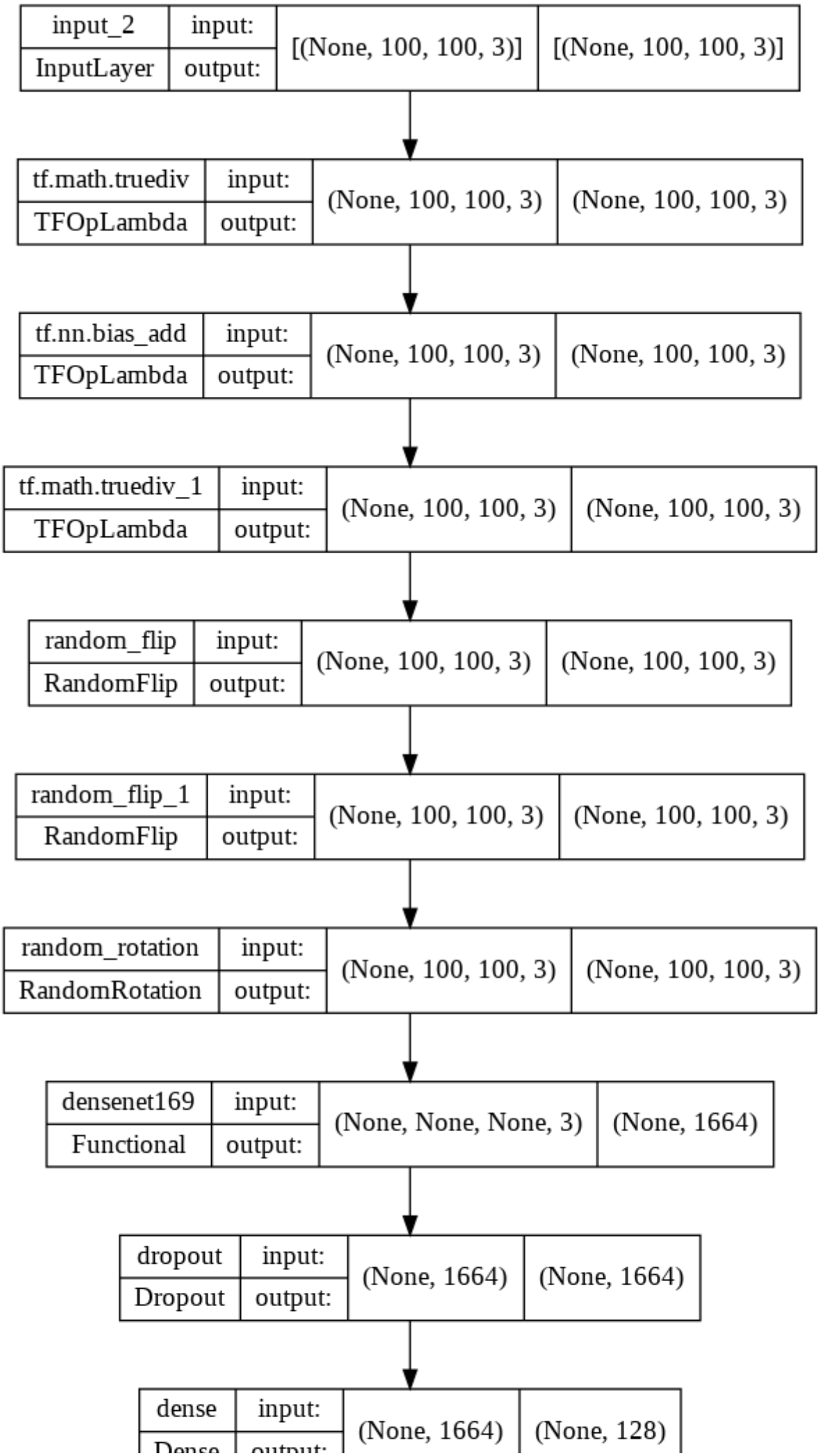
A su estructura básica, sin la capa final, añadimos una capa densa previa a la capa de salida.

```
Model: "model_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_4 (InputLayer)        [(None, 100, 100, 3)]     0

 tf.__operators__.getitem_1   (None, 100, 100, 3)      0
 (SlicingOpLambda)

 tf.nn.bias_add_1 (TFOpLambd  (None, 100, 100, 3)      0
 a)

 random_flip_2 (RandomFlip)  (None, 100, 100, 3)       0

 random_flip_3 (RandomFlip)  (None, 100, 100, 3)       0

 random_rotation_1 (RandomRo  (None, 100, 100, 3)      0
 tation)

 resnet152 (Functional)      (None, 2048)              58370944

 dropout_2 (Dropout)         (None, 2048)              0

 dense_2 (Dense)             (None, 128)               262272

 dropout_3 (Dropout)         (None, 128)               0

 dense_3 (Dense)             (None, 11)                1419

=================================================================
Total params: 58,634,635
Trainable params: 263,691
Non-trainable params: 58,370,944
_________________________________________________________________
```

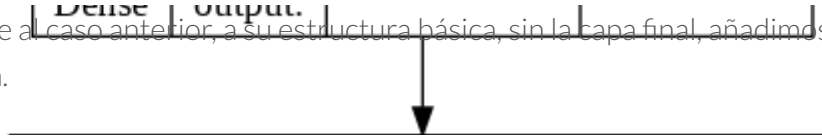
Pese a que todo el modelo suma casi 60 millones de parámetros, apenas poco más de 250 mil son entrenables y para el resto se hace uso de los pesos calculados para las imágenes de ImageNet.

Transferencia de aprendizaje con DenseNet

La DenseNet es una red neuronal residual como la ResNet, pero además permite varios saltos paralelos entre capas o grupos de capas.


input_2
InputLayer
input:
output:
[(None, 100, 100, 3)]
[(None, 100, 100, 3)]
tf.math.truediv
TFOpLambda
input:
output:
(None, 100, 100, 3)
(None, 100, 100, 3)
tf.nn.bias_add
TFOpLambda
input:
output:
(None, 100, 100, 3)
(None, 100, 100, 3)
tf.math.truediv_1
TFOpLambda
input:
output:
(None, 100, 100, 3)
(None, 100, 100, 3)
random_flip
RandomFlip
input:
output:
(None, 100, 100, 3)
(None, 100, 100, 3)
random_flip_1
RandomFlip
input:
output:
(None, 100, 100, 3)
(None, 100, 100, 3)
random_rotation
RandomRotation
input:
output:
(None, 100, 100, 3)
(None, 100, 100, 3)
densenet169
Functional
input:
output:
(None, None, None, 3)
(None, 1664)
dropout
Dropout
input:
output:
(None, 1664)
(None, 1664)
dense
input:
(None, 1664)
(None, 128)

Análogamente al caso anterior, a su estructura básica, sin la capa final, añadimos una capa densa previa a la capa de salida.



```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, 100, 100, 3)]     0

 tf.math.truediv (TFOpLambda  (None, 100, 100, 3)      0
 )

 tf.nn.bias_add (TFOpLambda)  (None, 100, 100, 3)      0

 tf.math.truediv_1 (TFOpLamb  (None, 100, 100, 3)      0
 da)

 random_flip (RandomFlip)    (None, 100, 100, 3)       0

 random_flip_1 (RandomFlip)  (None, 100, 100, 3)       0

 random_rotation (RandomRota  (None, 100, 100, 3)      0
 tion)

 densenet169 (Functional)    (None, 1664)              12642880

 dropout (Dropout)           (None, 1664)              0

 dense (Dense)               (None, 128)               213120

 dropout_1 (Dropout)         (None, 128)               0

 dense_1 (Dense)             (None, 11)                1419

=================================================================
Total params: 12,857,419
Trainable params: 214,539
Non-trainable params: 12,642,880
_________________________________________________________________
```

De nuevo, pese a que este modelo suma casi 13 millones de parámetros, apenas poco más de 210 mil son entrenables y para el resto se hace de nuevo uso de los pesos calculados para las imágenes de ImageNet.

Data Augmentation

Por data augmentation entendemos una serie de técnicas para incrementar el dataset de entrenamiento, via transformaciones del mismo. Su función es evitar el sobreentrenamiento de la red (que la red memorize sin generalizar).

En nuestro caso realizaremos una normalización (adecuada para cada modelo) y tres transformaciones de las imágenes:

- Volteado horizontal
- Volteado vertical
- Rotación aleatoria

```
tf.keras.layers.Rescaling(scale=1./127.5, offset=-1),
tf.keras.layers.RandomFlip(mode="horizontal", seed=42),
tf.keras.layers.RandomFlip(mode="vertical", seed=42),
tf.keras.layers.RandomRotation(1, fill_mode="reflect", interpolation="bilinear", seed=42, fill_value=0.0),
```

Obtimizado

La obtimización es el método matemático que se emplea para calcular los parámetros libres del modelo durante el entrenamiento. En nuestro caso empleamos el Stochastic Gradient Descent o SGD que se parametriza con el ritmo de entrenamiento y con el momentum.

El ritmo de entrenamiento o learning rate es el paso empleado para modificar los parámetros del modelo, mientras que el momentum es un recurso para acelerar el entrenamiento cuando este está resultando eficiente.

Además, es posible configurar el tamaño de batch (número de imágenes que se procesarán simultáneamente) y el número de pasos o epochs de entrenamiento.

Para los modelos propuestos estos son los parámetros empleados:

Parámetro	**Red convolucional simple**	**Red basada en ResNet**	**Red basada en DenseNet**
Obtimizador	Stochastic gradient descent	Stochastic gradient descent	Stochastic gradient descent
Ratio de aprendizaje	0,0001	0,0001	0,0005
Momentum	0.9	0.9	0.9
Tamaño de batch	128	128	128
Número de epochs	1.000	3.000	3.000

La función que se minimiza (función de coste) es la CategoricalCrossentropy (entropía cruzada categórica), mientras que las métricas empleadas durante el ajuste son diversas, descatando la CategoricalAccuracy (exactitud categórica), F1Score (combina las medidas de precision y recall en un sólo valor) y el MatthewsCorrelationCoefficient (una medida no afectada por datasets desbalanceados).

Adicionalmente se realizan otros análisis de precisión, así como una matriz de confusión.

Tipos de callbacks

Los callbacks son funciones que se ejecutan tras cada ejecución y que permiten diversos servicios o funcionalidades. En estre trabajo se han definido algunos callbacks entre los que destacan:

- **CSVLogger**: guarda los parámetros del entrenamiento en formato estándar, permitiendo gráficas o análisis posterior.

- **ModelCheckpoint**: hace un seguimiento del entrenamiento y guarda los parámetros del modelo cuando se obtiene un resultado mejor que todos los anteriores.

El primero permite realizar gráficas, mientras que el segundo permite detener y reanudar o extender el entrenamiento.

Entorno de ejecución Google Colab

Para este trabajo se ha empleado Google Colab para desarrollar y ejectuar los entrenamientos. Esta elección está motivada por no disponer de RAM suficiente y de una GPU física en la que ejecutar el código en un plazo razonable.

Cuando las metodologías y entrenamientos definitivos fueron definidos, se recurrió a una cuenta de pago Colab Pro, para garantizar un acceso preferencial a los recursos y permitir entrenamientos de cuantas epochs fueran necesarios.

```
Tensorflow version:  2.8.2
Num GPUs Available:  1
Your runtime has 27.3 gigabytes of available RAM
(You are using a high-RAM runtime!)

GPU 0: Tesla T4 (UUID: GPU-1cc38db7-d6ad-65aa-5c25-d44417a718d1)
```

La persistencia de los entrenamientos se garantiza por el uso de Google-Drive en su versión gratuita para salvaguardar los logs y los mejores modelos.

Atrás – **Índice** – **Siguiente**

Project maintained by **lumaro77**
Hosted on GitHub Pages — Theme by **orderedlist**

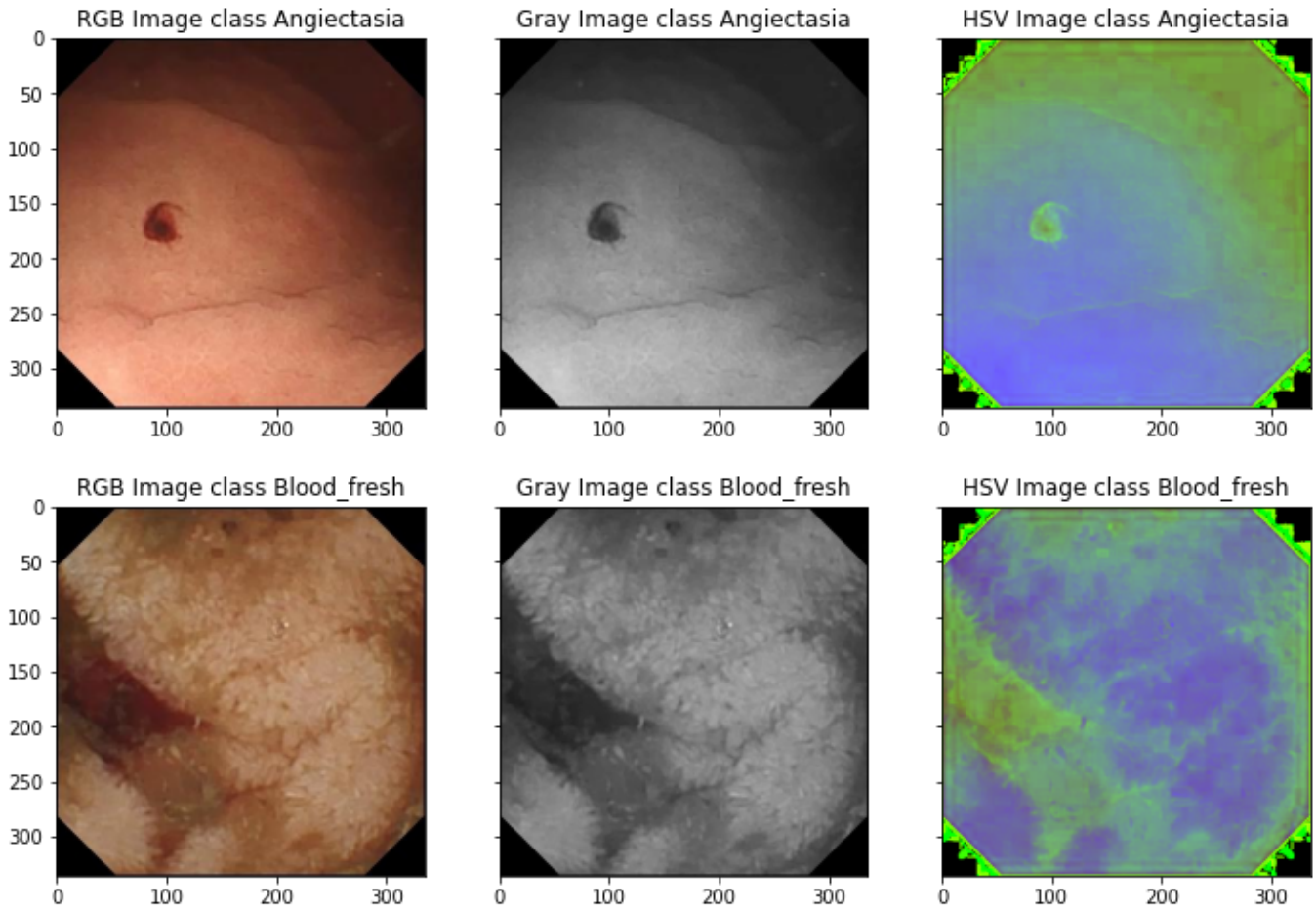
Datos

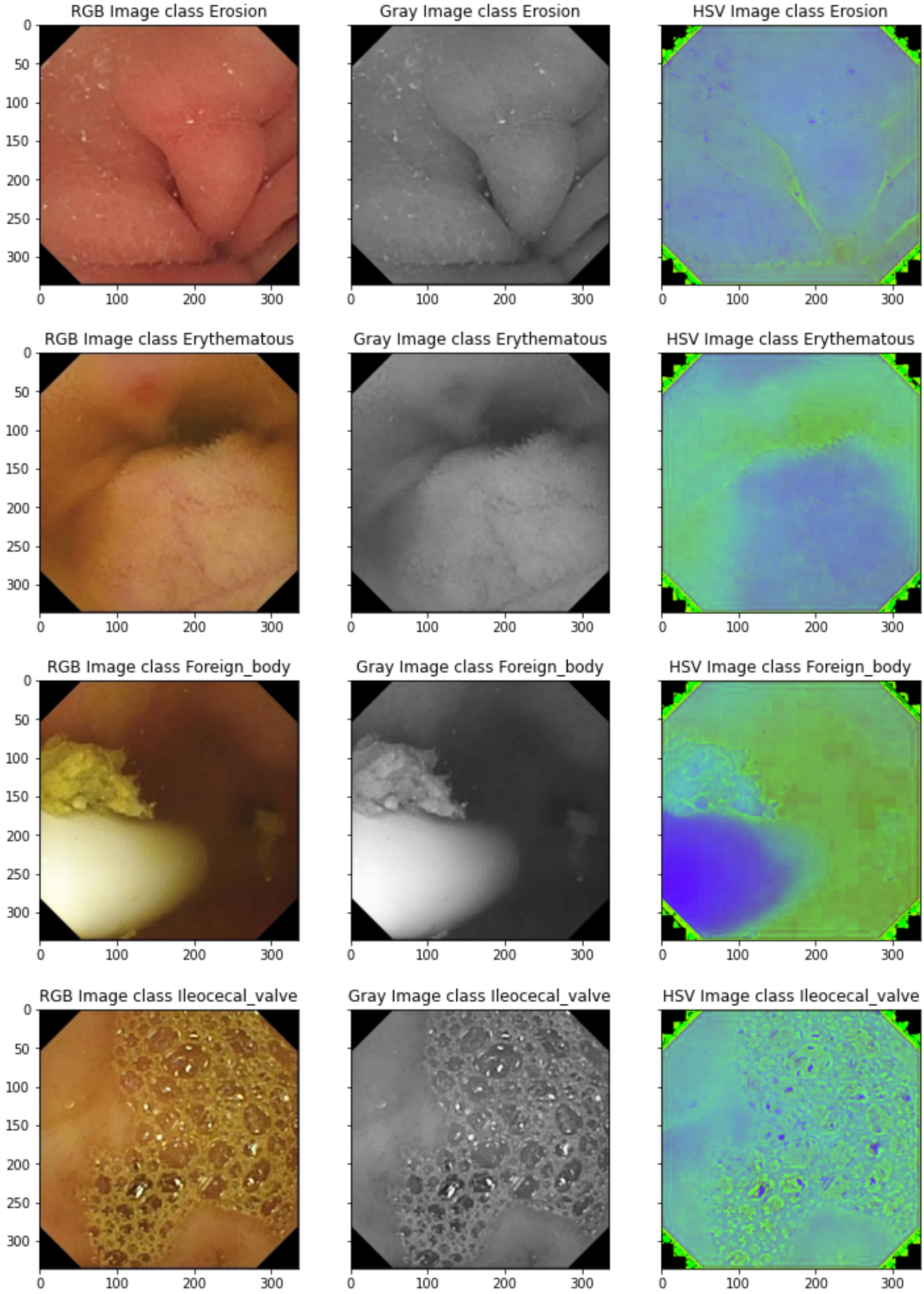
Para este proyecto se emplea el Kvasir-Capsule dataset disponible en link.

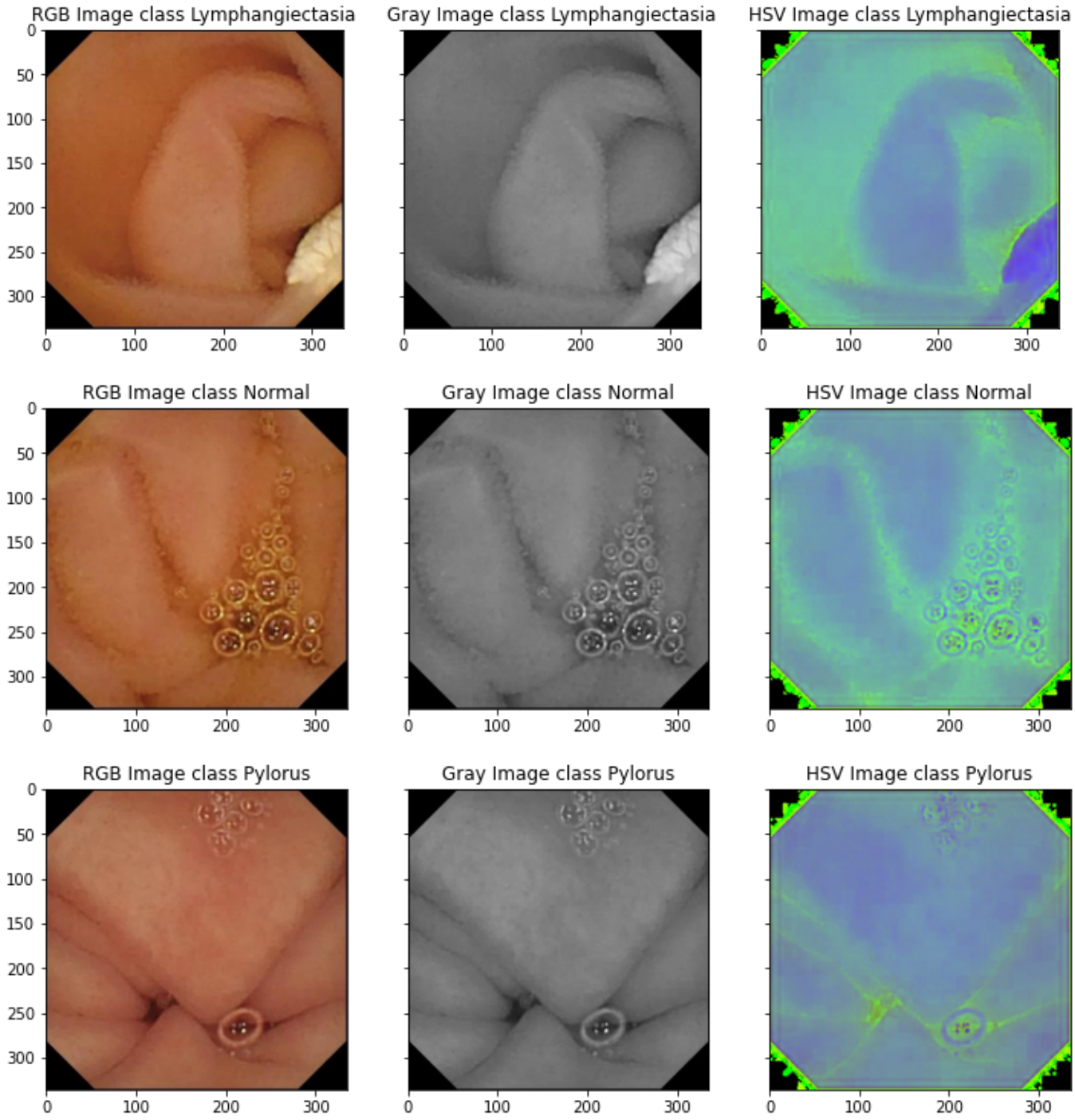
Se trata de un dataset de video del que hay disponibles una serie de imágenes etiquetadas con hasta 14 categorías así como gran cantidad de imágenes y videos sin etiquetar.

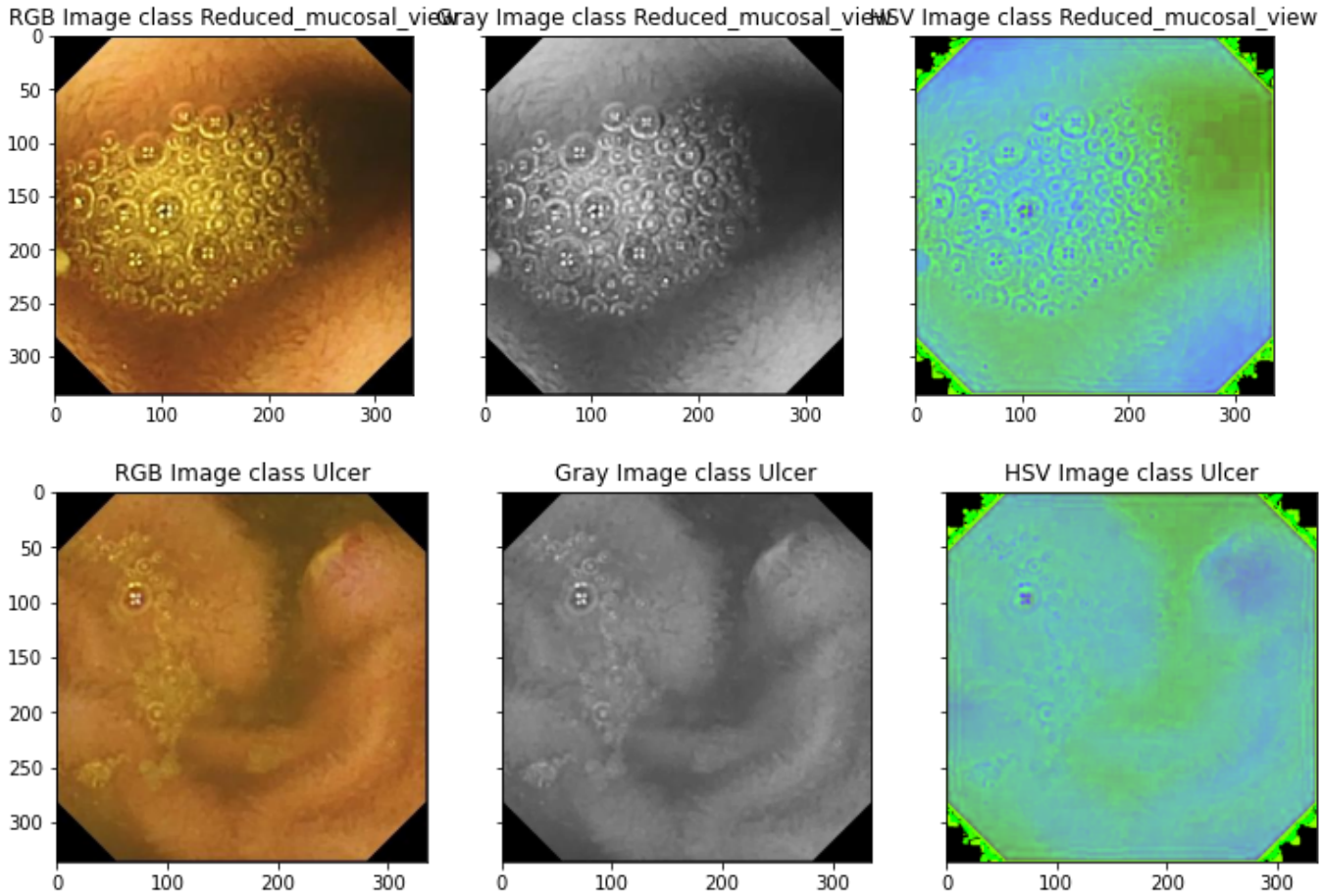
El Kvasir-Capsule dataset

Las imágenes corresponden a regiones y lesiones del sistema digestivo humano. Dado que algunas clases están excesivamente infrarepresentadas, se han tomado las siguientes 11 clases como válidas para realizar este estudio.




RGB Image class Erosion
Gray Image class Erosion
HSV Image class Erosion
RGB Image class Erythematous
Gray Image class Erythematous
HSV Image class Erythematous
RGB Image class Foreign_body
Gray Image class Foreign_body
HSV Image class Foreign_body
RGB Image class Ileocecal_valve
Gray Image class Ileocecal_valve
HSV Image class Ileocecal_valve
0
50
100
150
200
250
300


RGB Image class Lymphangiectasia
Gray Image class Lymphangiectasia
HSV Image class Lymphangiectasia
RGB Image class Normal
Gray Image class Normal
HSV Image class Normal
RGB Image class Pylorus
Gray Image class Pylorus
HSV Image class Pylorus
0
50
100
150
200
250
300



Las imágenes son de un ámbito muy específico (campo de las ciencias de la salud) del que el autor se considera lego, más allá del sentido común. La simple inspección de las imágenes genera bastante confusión puesto que no es sencillo deducir la etiqueta asignada y surgen dudas (compatibles con la ignorancia en la materia) de la idoneidad del etiquetado.

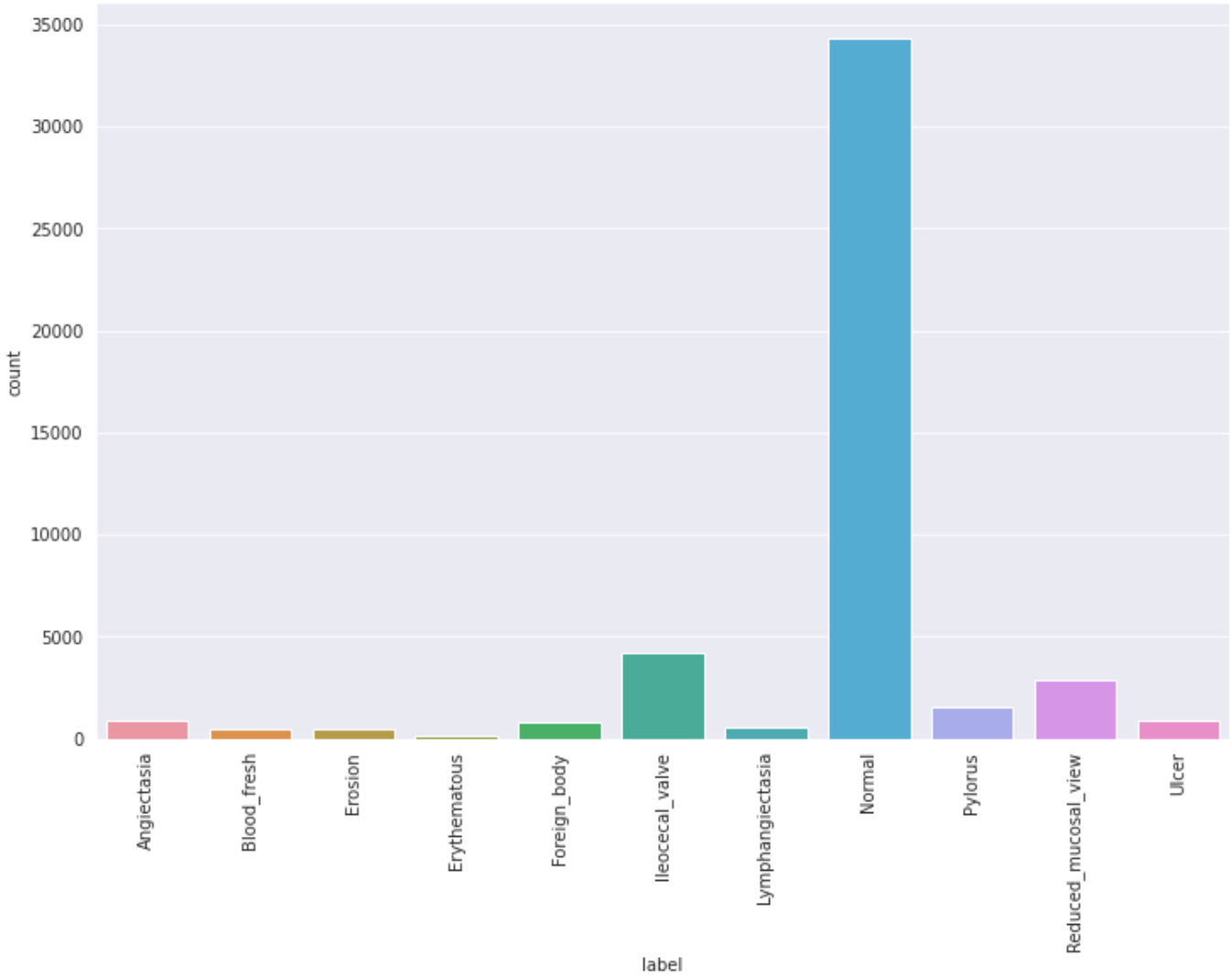
Análisis del dataset

El número de imágenes presente en todo el dataset oculta la incómoda realidad de muchos datasets: desbalanceo entre clases, es decir, no hay la misma cantidad de imágenes para cada clase.

Id	Clase	Número de muestras
0	Angiectasia	866
1	Blood_fresh	446
2	Erosion	506
3	Erythematous	159
4	Foreign_body	776
5	Ileocecal_valve	4.189
6	Lymphangiectasia	592
7	Normal	34.338
8	Pylorus	1.529

Id	Clase	Número de muestras
9	Reduced_mucosal_view	2.906
10	Ulcer	854

Este desbalanceo es palmario al realizar un histograma de frecuencia de todo el dataset.



Datos de entrenamiento y validación

Este trabajo se considera una aproximación al problema presentado, con fines didácticos, por lo que tomé la decisión de aplicar los modelos a todo el dataset, reservando un 30% para validación.

```
Found 47161 files belonging to 11 classes.
Using 33013 files for training.
```

Los datos son alimentados en el sistema via un objeto Dataset que toma las imágenes a 3 canales RGB y las resamplea a un tamaño 100x100 píxels, para facilitar el entrenamiento y para que el entorno de proceso fuera capaz de aceptar los datos sin exceder los límites de RAM y capacidades del GPU asignado.

Project maintained by **lumaro77**
Hosted on GitHub Pages — Theme by **orderedlist**

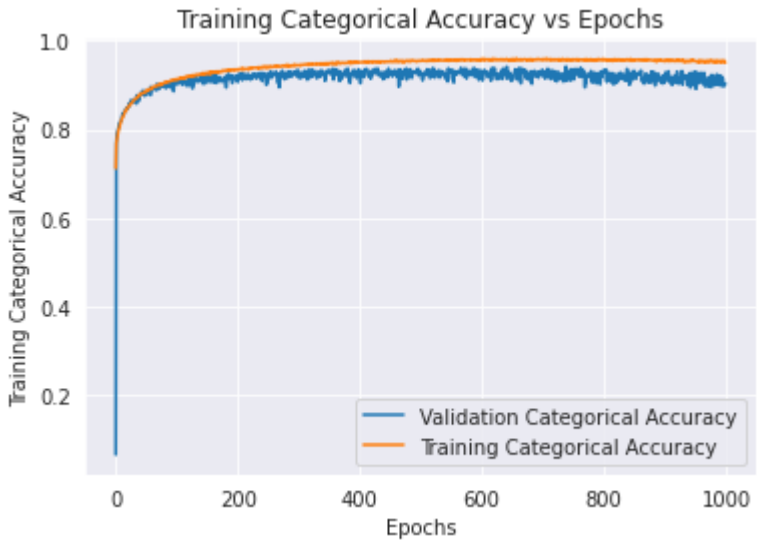
Resultados

Resultados con red convolucional simple ConvoRGB

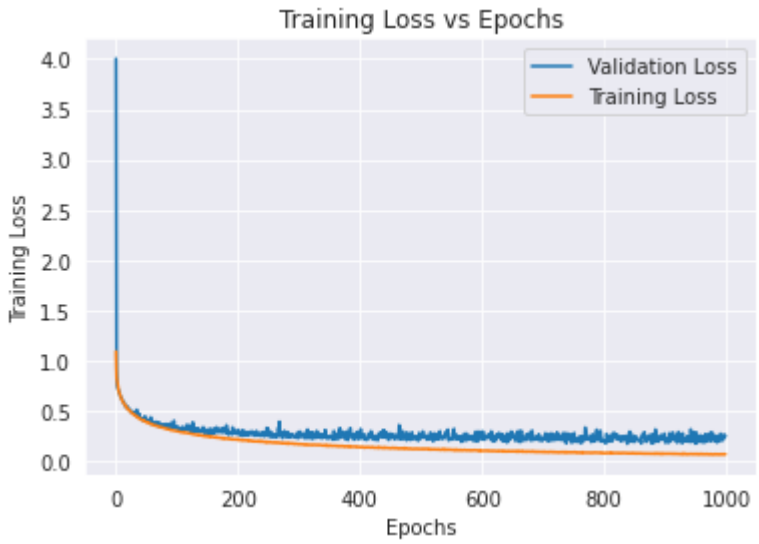
La red convolucional simple, que denominamos ConvoRGB, se ajusta para todos sus parámetros, según los parámetros y el dataset descritos en los apartados anteriores con los siguientes resultados y métricas.

Ajuste de red convolucional simple ConvoRGB

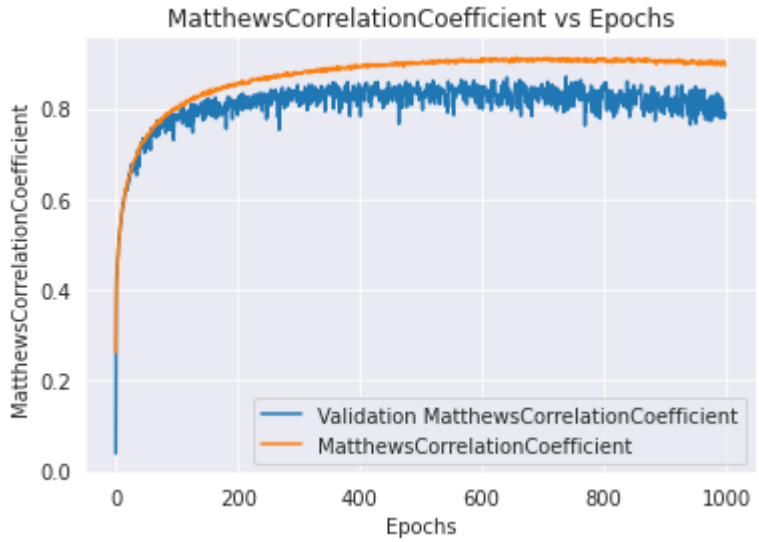
Las gráficas de accuracy muestran que la red aprende a niveles superiores del 90% de los datos de entrenamiento, pero presenta rápidamente un sobreentrenamiento (overfitting) leve porque los resultados en el dataset de validación empeoran con las epochs.



Las gráficas de loss tienen un comportamiento opuesto (como es lógico) a las de accuracy.



Las gráficas del coeficiente de Matthews validan lo observado en las anteriores, e incluso ayudan a ver que entre 400 y 600 epochs se alcanza un máximo en el entrenamiento y al final del entrenamiento realizado ambas gráficas (entrenamiento y validación) tienden a empeorar.



En relación a lo anterior, hay que indicar que los callbacks permiten guardar cada modelo o solo los mejores (se configuró de esta manera), por lo que es posible resolver problemas de sobreentrenamiento (por el dataset o la configuración) reiniciando el entrenamiento con una mejor configuración del sistema, salvaguardando los resultados alcanzados previamente.

Precisiones con red convolucional simple ConvoRGB

Se muestran a continuación los diversos indicadores estadísticos del ajuste de la red ConvoRGB.

```
loss = 0.171
categorical_accuracy = 0.937
categorical_crossentropy = 0.171
false_negatives =  20
false_positives =  59332
true_negatives =  82148
true_positives =  14128
f1_score = 0.937
precision = 0.192
recall = 0.999
MatthewsCorrelationCoefficient = 0.860
```

Los valores de las métricas permite afirmar que la red ConvoRGB parece modelar el dataset de trabajo.

```
Accuracy = 0.937
Accuracy Balanced = 0.782
Precision micro = 0.937
Precision macro = 0.867
Precision weighted = 0.937
Recall micro = 0.937
Recall macro = 0.782
Recall weighted = 0.937
F1 micro = 0.937
F1 macro = 0.813
F1 weighted = 0.936
MCC = 0.860
Kappa = 0.860
```

	precision	recall	f1-score	support
0	0.74	0.83	0.78	271
1	0.98	0.94	0.96	133
2	0.59	0.43	0.50	150
3	0.92	0.44	0.60	52
4	0.91	0.84	0.87	217
5	0.84	0.92	0.88	1263
6	0.93	0.66	0.77	191
7	0.97	0.97	0.97	10336
8	0.86	0.83	0.84	431
9	0.92	0.90	0.91	847
10	0.89	0.85	0.87	257
accuracy			0.94	14148
macro avg	0.87	0.78	0.81	14148
weighted avg	0.94	0.94	0.94	14148

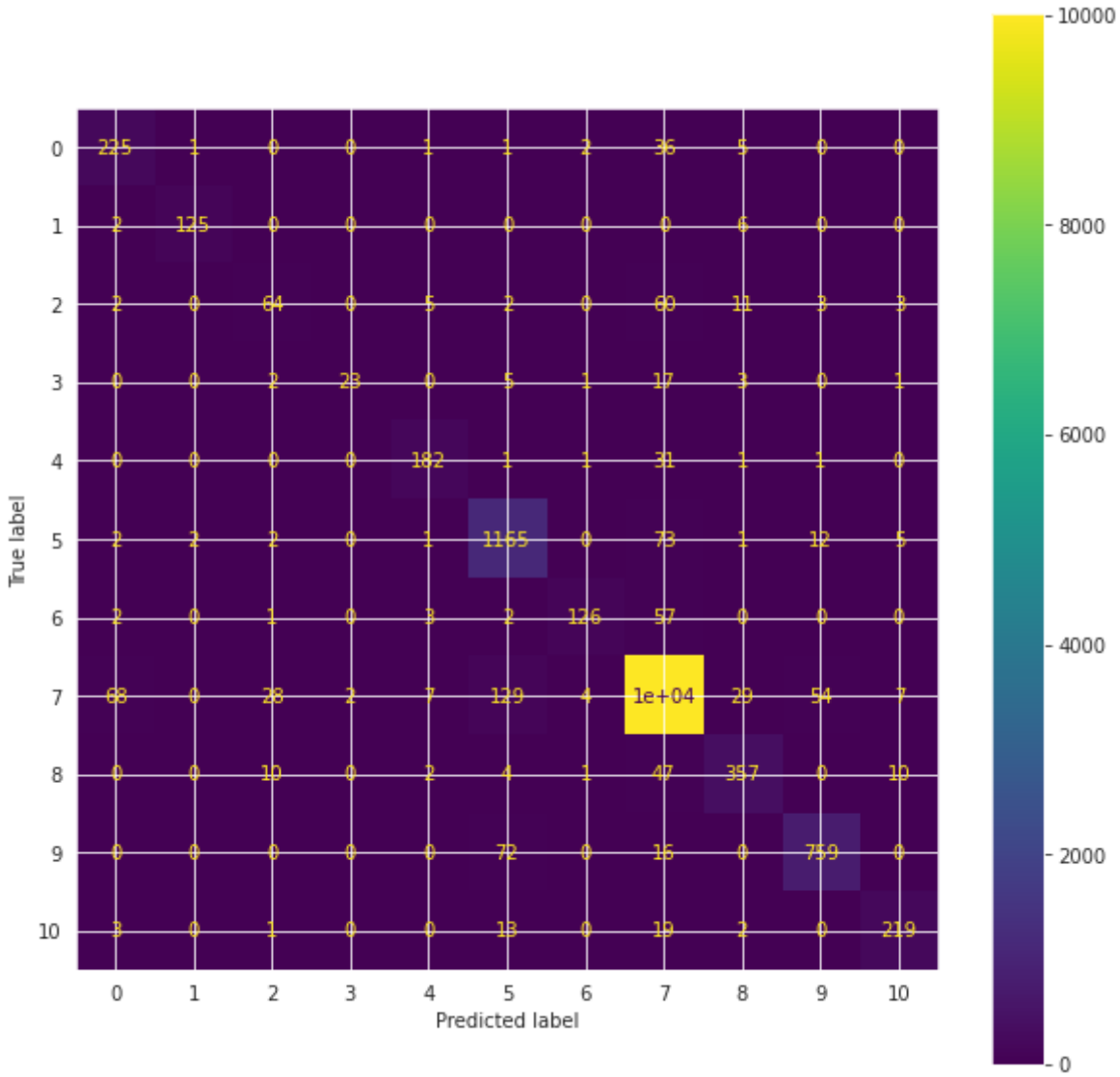
Aquellas metricas que tienen en cuenta cada una de las clases presentan unos resultados más discretos. Esto tiene sentido por la presencia de una clase superrepresentada (la clase normal) que puede enmascarar los problemas con otras clases subrepresentadas.

Matriz de confusión con red convolucional simple ConvoRGB

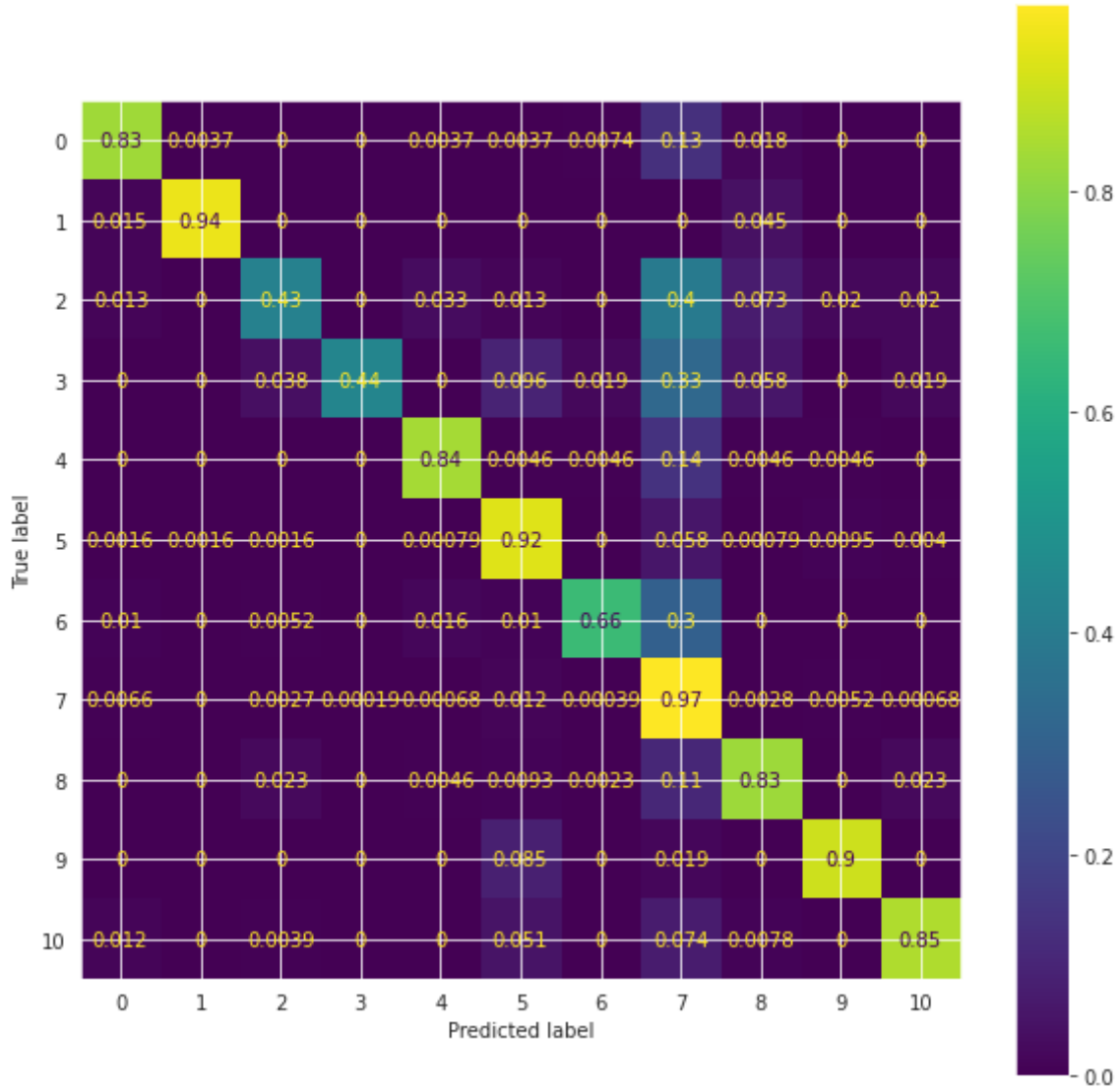
La matriz de confusión con los datos de validación para la red ConvoRGB tiene los siguientes valores:

	0	1	2	3	4	5	6	7	8	9	10
0	225	1	0	0	1	1	2	36	5	0	0
1	2	125	0	0	0	0	0	0	6	0	0
2	2	0	64	0	5	2	0	60	11	3	3
3	0	0	2	23	0	5	1	17	3	0	1
4	0	0	0	0	182	1	1	31	1	1	0
5	2	2	2	0	1	1165	0	73	1	12	5
6	2	0	1	0	3	2	126	57	0	0	0
7	68	0	28	2	7	129	4	10008	29	54	7
8	0	0	10	0	2	4	1	47	357	0	10
9	0	0	0	0	0	72	0	16	0	759	0
10	3	0	1	0	0	13	0	19	2	0	219

La matriz de confusión con los datos de validación para la red ConvoRGB representada gráficamente queda como:



Si normalizamos la matriz de confusión respeto a los datos verdaderos para la red ConvoRGB queda como:



Los mejores resultados se obtienen para la clase 7-Normal, así como las clases 1-Blood_fresh y 5-Ileocecal_valve . Por su lado los peores resultados se obtienen para las clases 2-Erosion 3-Erythematous y 6-Lymphangiectasia.

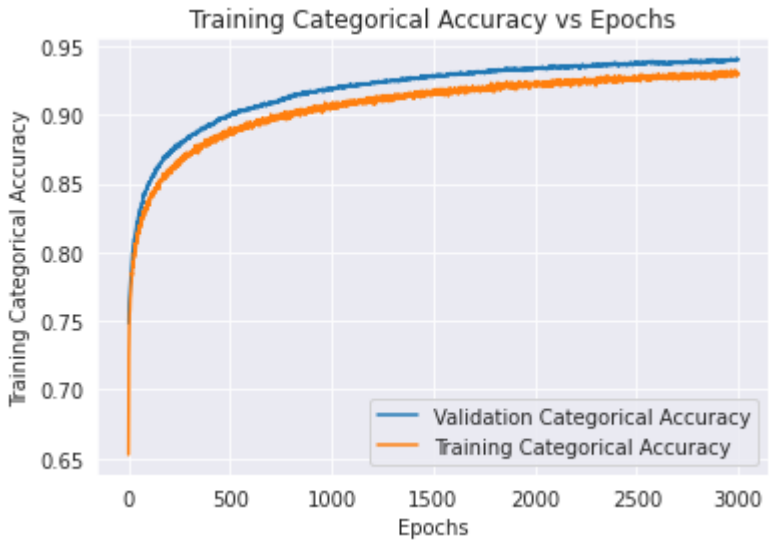
Una mayor número de imágenes tiende a ayudar a mejores resultados procentuales, aunque las características de las lesiones o regiones fisiológicas parece definitiva, como la clase 1, que pese a estar poco presente obtiene buenos resultados.

Resultado de transferencia de conocimiento con red ResNet

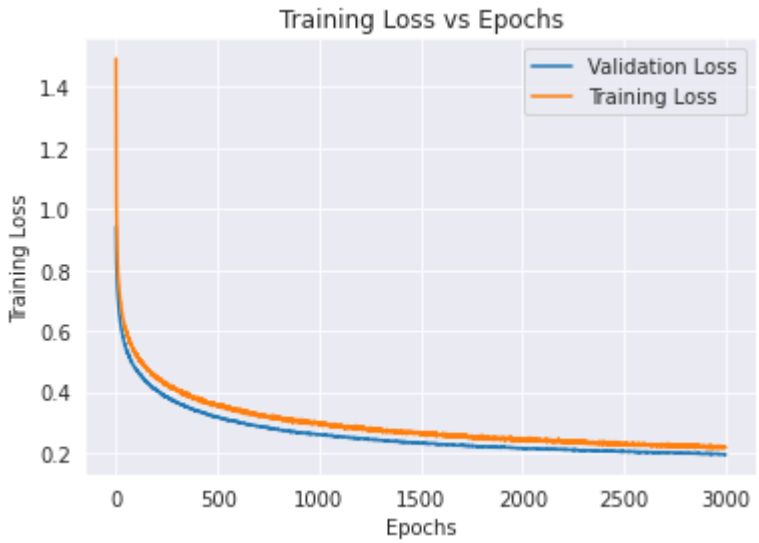
La red convolucional que denominamos ResNet, se ajusta para sus parámetros libres según se ha descritos en los apartados anteriores con los siguientes resultados y métricas.

Ajuste de red ResNet

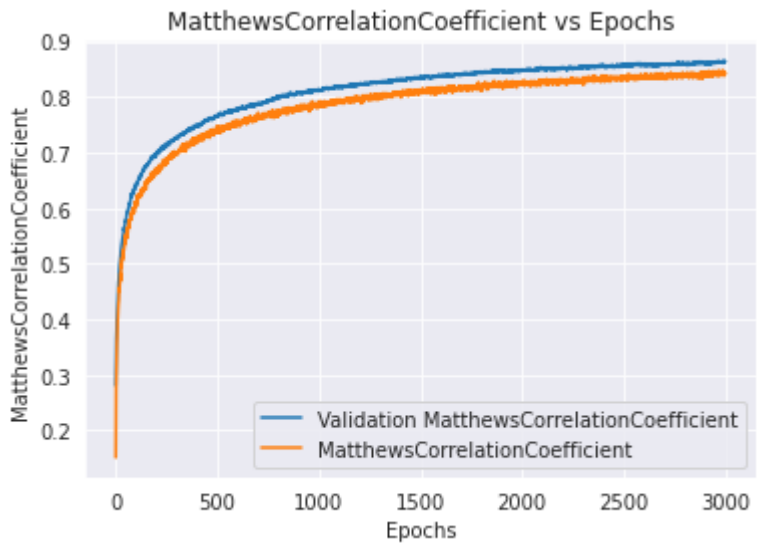
Las gráficas de accuracy muestran que la red basada en ResNet aprende a niveles superiores del 93% de los datos de entrenamiento, sin sobreentrenamiento. De hecho el entrenamiento se interrumpe quizás anticipadamente aunque tampoco parece que sea posible obtener valores muy superiores.



Las gráficas de accuracy y loss tienen los esperados comportamientos opuestos para el entrenamiento de la red ResNet.



El coeficiente de Matthews para la validación roza el 85%, aproximadamente igual a los mejores resultados de la red ConvoRGB del primer apartado.



Los resultados de la red ResNet son numéricamente análogos a la red ConvoRGB, y aunque las gráficas parecen menos ruidosas, hay que tener presente la escala horizontal (una es 3 veces la otra).

Precisiones con red ResNet

Se muestran a continuación los diversos indicadores estadísticos del ajuste de la red ResNet.

```
loss = 0.194
categorical_accuracy = 0.941
categorical_crossentropy = 0.194
false_negatives =  976
false_positives =  685
true_negatives =  140795
true_positives =  13172
f1_score = 0.941
precision = 0.951
recall = 0.931
MatthewsCorrelationCoefficient = 0.864
```

Los valores obtenidos para la red ResNet no parecen diferir sustancialmente de los obtenidos por la red ConvoRGB.

```
Accuracy = 0.941
Accuracy Balanced = 0.727
Precision micro = 0.941
Precision macro = 0.914
Precision weighted = 0.939
Recall micro = 0.941
Recall macro = 0.727
Recall weighted = 0.941
F1 micro = 0.941
F1 macro = 0.795
F1 weighted = 0.936
MCC = 0.864
Kappa = 0.861
```

	precision	recall	f1-score	support
0	0.98	0.79	0.88	271
1	0.98	0.90	0.94	133
2	0.75	0.35	0.47	150
3	0.83	0.29	0.43	52
4	0.98	0.79	0.88	217
5	0.93	0.88	0.90	1263
6	0.95	0.64	0.76	191
7	0.95	0.99	0.97	10336
8	0.84	0.62	0.71	431
9	0.92	0.93	0.93	847
10	0.94	0.82	0.88	257
accuracy			0.94	14148
macro avg	0.91	0.73	0.80	14148
weighted avg	0.94	0.94	0.94	14148

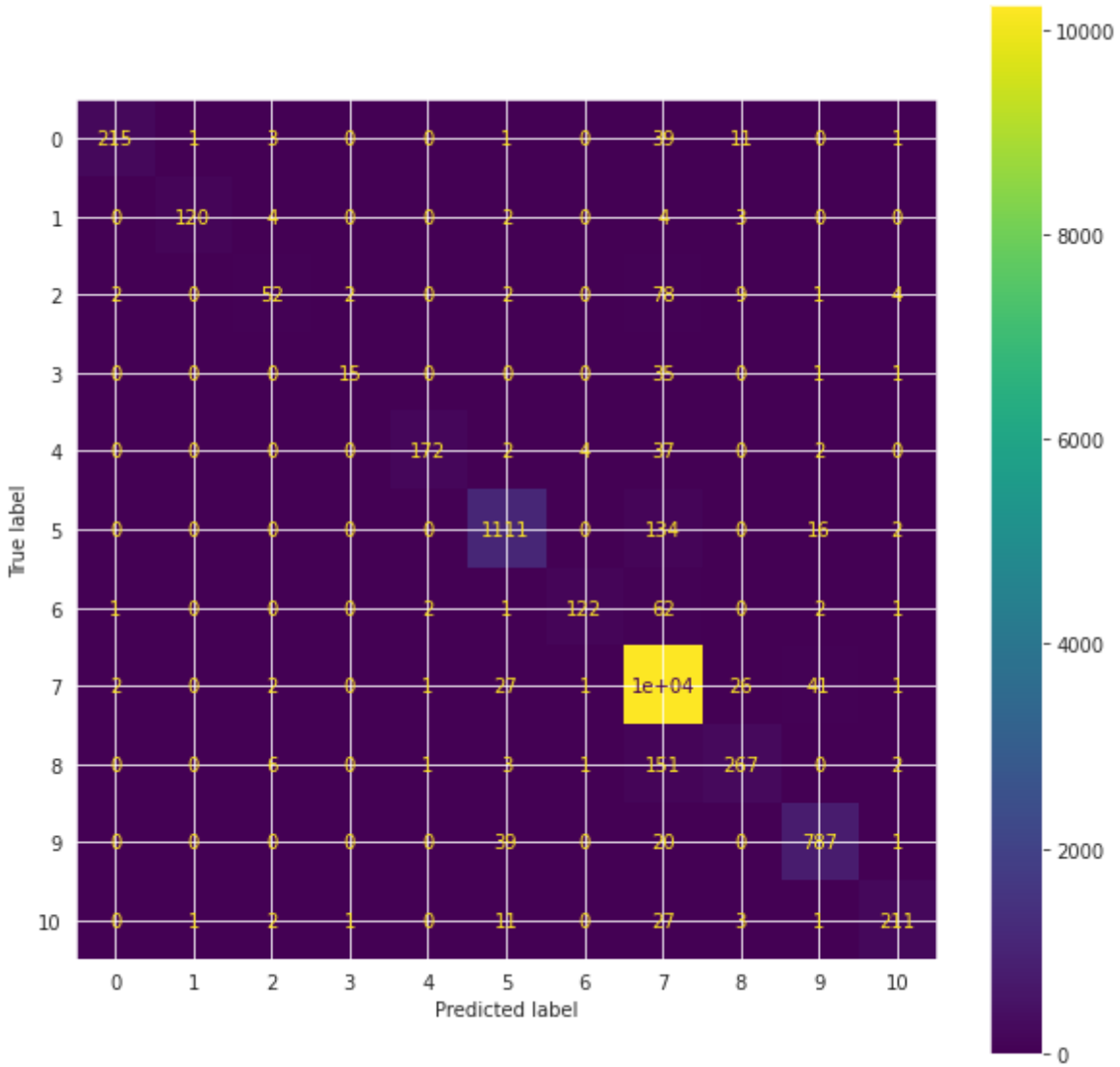
De nuevo, como en el caso anterior, las métricas macro (calculadas por clase y posteriormente agregadas) son peores que las micro (se toman todos los datos conjuntamente). Esto está indicando que los modelos no resuelven bien las clases poco representadas.

Matriz de confusión con red ResNet

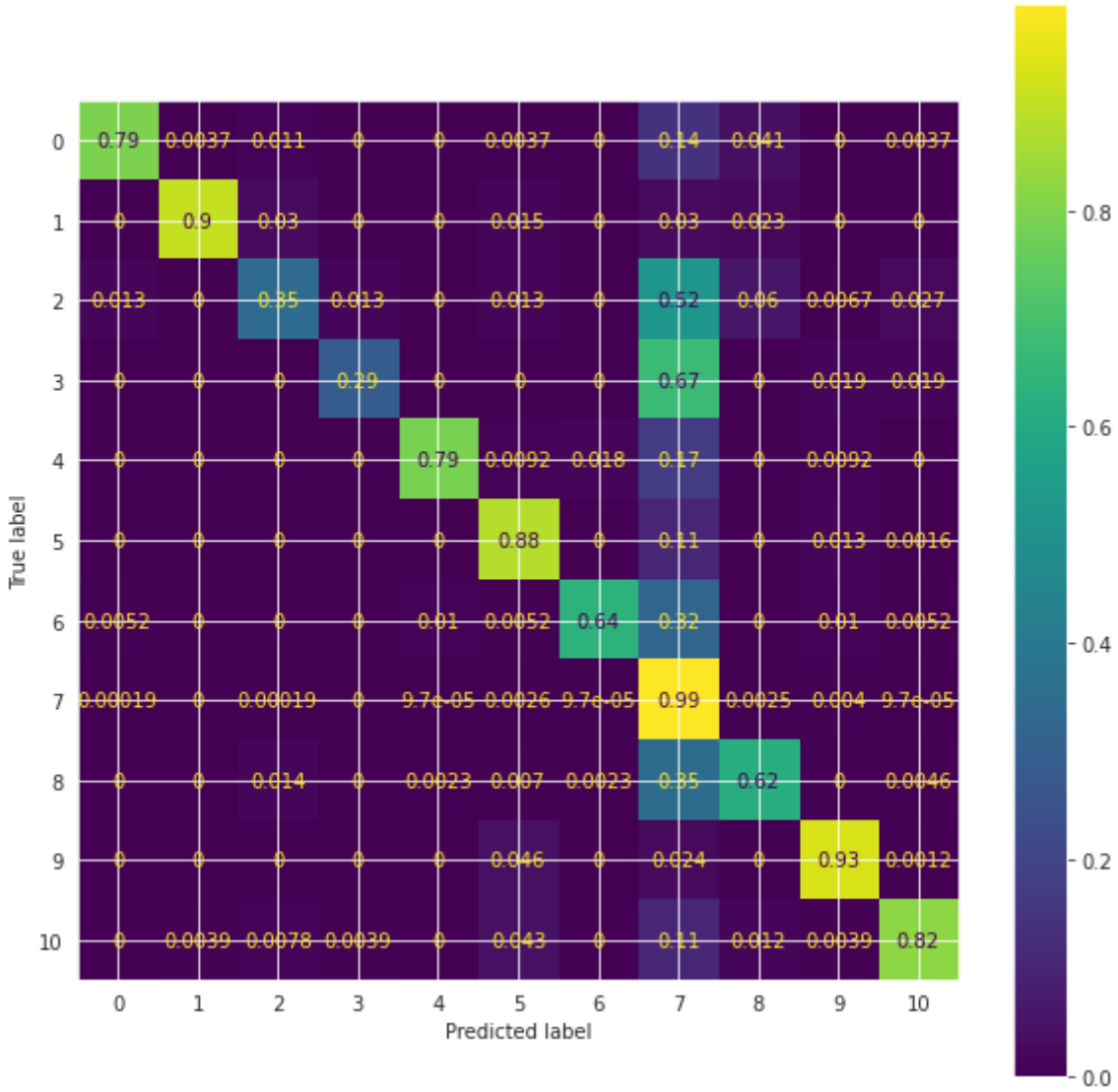
La matriz de confusión con los datos de validación para la red ResNet tiene los siguientes valores:

	0	1	2	3	4	5	6	7	8	9	10
0	215	1	3	0	0	1	0	39	11	0	1
1	0	120	4	0	0	2	0	4	3	0	0
2	2	0	52	2	0	2	0	78	9	1	4
3	0	0	0	15	0	0	0	35	0	1	1
4	0	0	0	0	172	2	4	37	0	2	0
5	0	0	0	0	0	1111	0	134	0	16	2
6	1	0	0	0	2	1	122	62	0	2	1
7	2	0	2	0	1	27	1	10235	26	41	1
8	0	0	6	0	1	3	1	151	267	0	2
9	0	0	0	0	0	39	0	20	0	787	1
10	0	1	2	1	0	11	0	27	3	1	211

La matriz de confusión con los datos de validación para la red ResNet representada gráficamente queda como:



Si normalizamos la matriz de confusión respeto a los datos verdaderos para la red ResNet queda como:



Los mejores resultados se obtienen para la clase 7-Normal, así como las clases 9-Reduced_mucosal_view y 1-Blood_fresh. Por su lado los peores resultados se obtienen para las clases 3-Erythematous, 2-Erosion y 8-Pylorus.

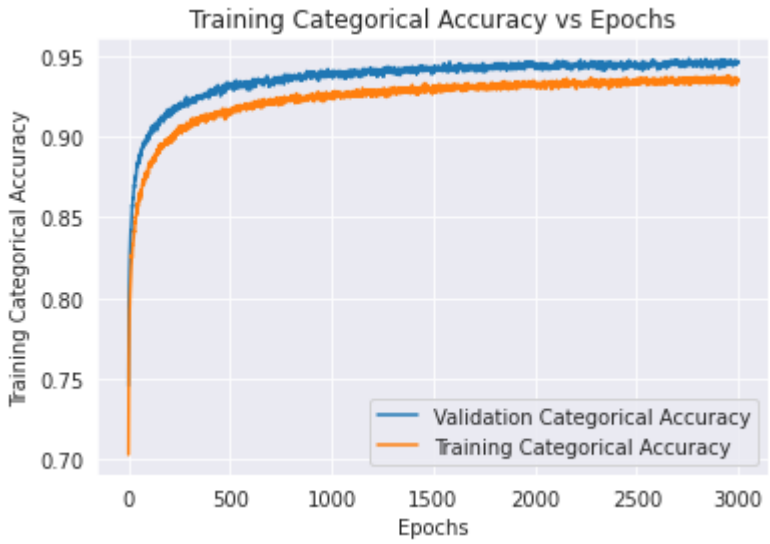
La semejanza de los resultados con los anteriores ratifica que una mayor número de imágenes tiende a generar mejores resultados procentuales, aunque las características de las lesiones o regiones fisiológicas concretas a identificar siguen siendo definitivas.

Resultado de transferencia de conocimiento con red DenseNet

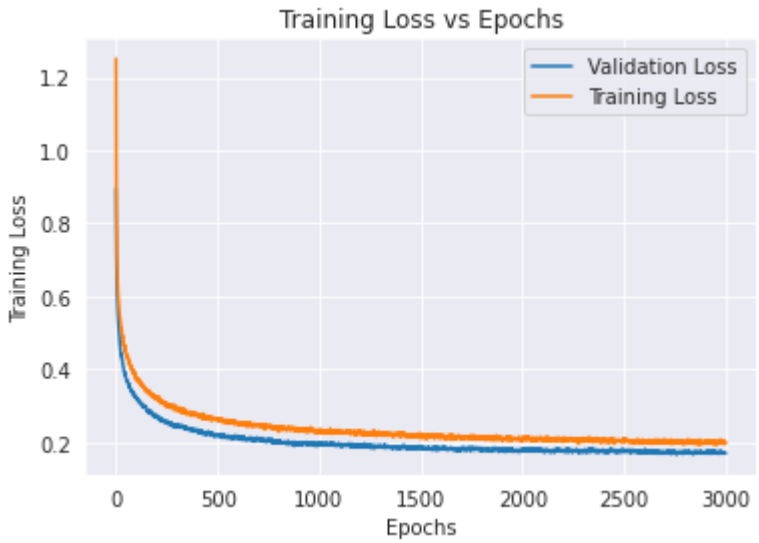
La red convolucional que denominamos DenseNet, como el caso anterior, se ajusta para sus parámetros libres según se ha descritos en los apartados anteriores con los siguientes resultados y métricas.

Ajuste de red DenseNet

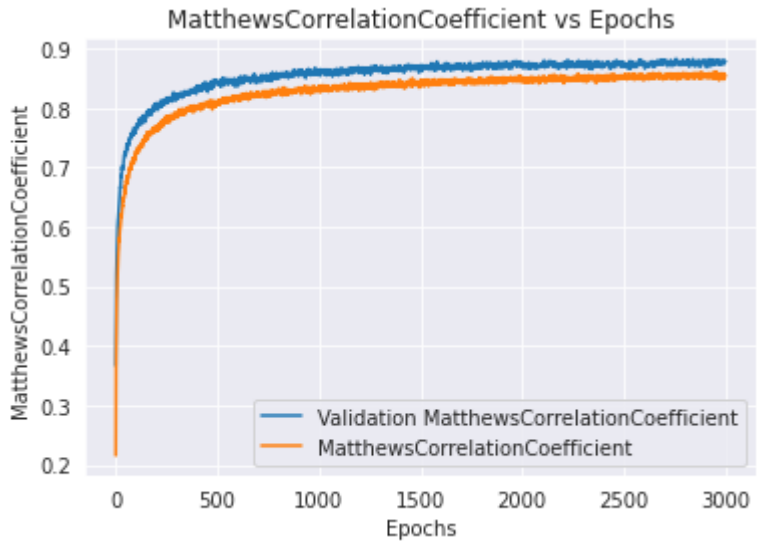
Las gráficas de accuracy muestran que la red basada en DenseNet aprende a niveles del 93% de los datos de entrenamiento, sin sobreentrenamiento. En este caso el entrenamiento parece suficiente porque las gráficas son bastante planas al final del mismo.



El loss se comporta como es esperado, aunque su valor absoluto sea complejo de interpretar en terminos absolutos, almenos más que la accuracy.



El coeficiente de Matthews con DenseNet reproduce lo observadado para ResNet, con valores muy similares y compatibles con ResNet e incluso ConvoRGB.



Precisiones con red DenseNet

Se muestran a continuación los diversos indicadores estadísticos del ajuste de la red DenseNet.

```
loss = 0.168
categorical_accuracy = 0.947
categorical_crossentropy = 0.168
false_negatives =  839
false_positives =  620
true_negatives =  140860
true_positives =  13309
f1_score = 0.947
precision = 0.955
recall = 0.941
MatthewsCorrelationCoefficient = 0.880
```

Los resultados de la evaluación con DenseNet es similar a las anteriores. Valores superiores al 90% para todas las muestras que se reducen a valores del 80% si se toman en consideración las clases (algunas de ellas muy sub-representadas en el dataset)

```
Accuracy = 0.947
Accuracy Balanced = 0.736
Precision micro = 0.947
Precision macro = 0.931
Precision weighted = 0.946
Recall micro = 0.947
Recall macro = 0.736
Recall weighted = 0.947
F1 micro = 0.947
F1 macro = 0.806
F1 weighted = 0.943
MCC = 0.880
Kappa = 0.876
```

	precision	recall	f1-score	support
0	0.99	0.81	0.89	271
1	1.00	0.84	0.91	133
2	0.84	0.35	0.50	150
3	0.76	0.25	0.38	52
4	0.99	0.79	0.88	217
5	0.95	0.86	0.90	1263
6	0.98	0.70	0.81	191
7	0.95	1.00	0.97	10336
8	0.93	0.72	0.81	431
9	0.93	0.94	0.94	847
10	0.92	0.84	0.88	257
accuracy			0.95	14148
macro avg	0.93	0.74	0.81	14148
weighted avg	0.95	0.95	0.94	14148

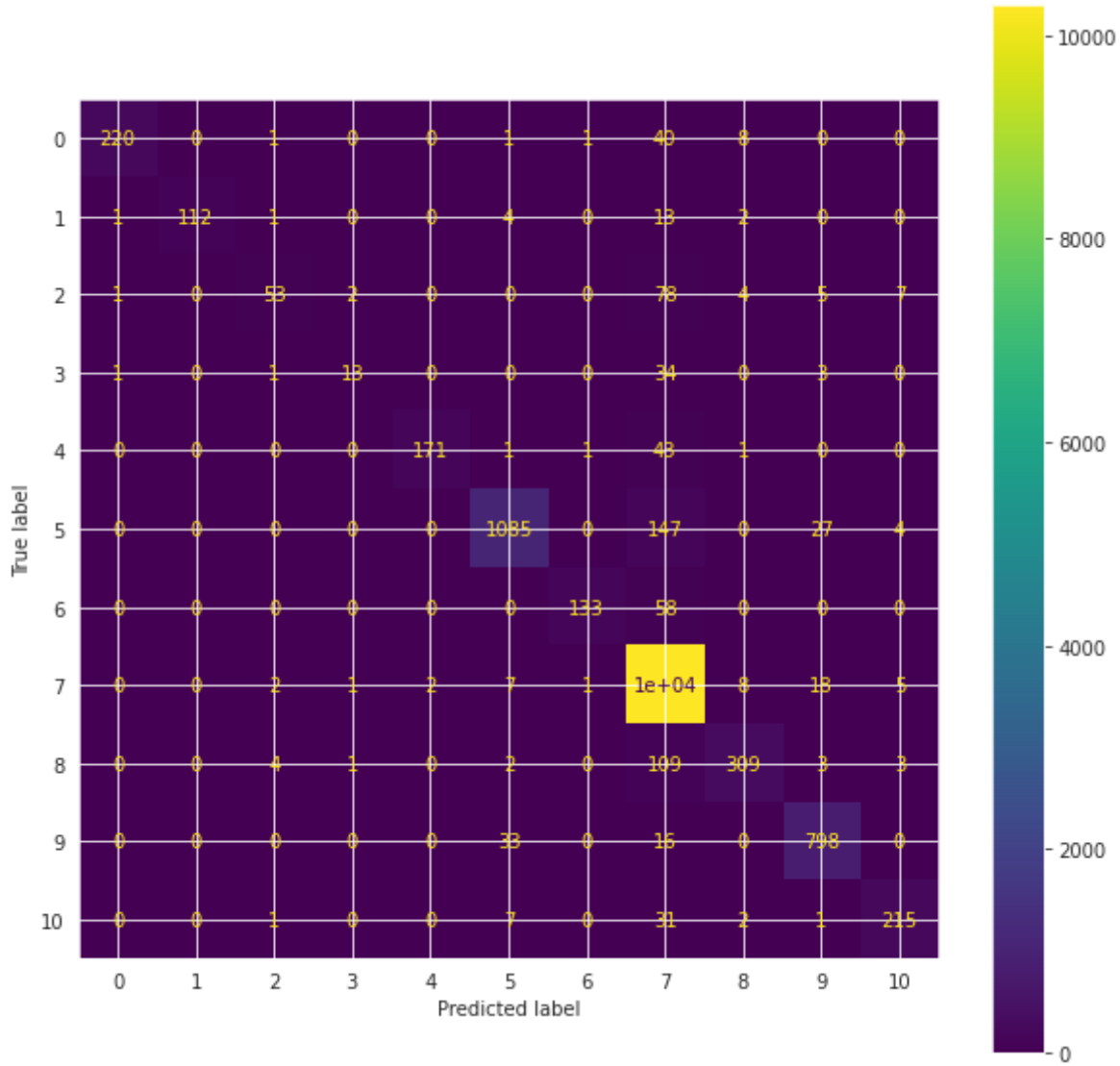
El resto de métricas muestran comportamiento similar a los modelos ConvoRGB y ResNet, con resultados que no difieren suficientemente como para establecer jerarquías entre los tres modelos.

Matriz de confusión con red DenseNet

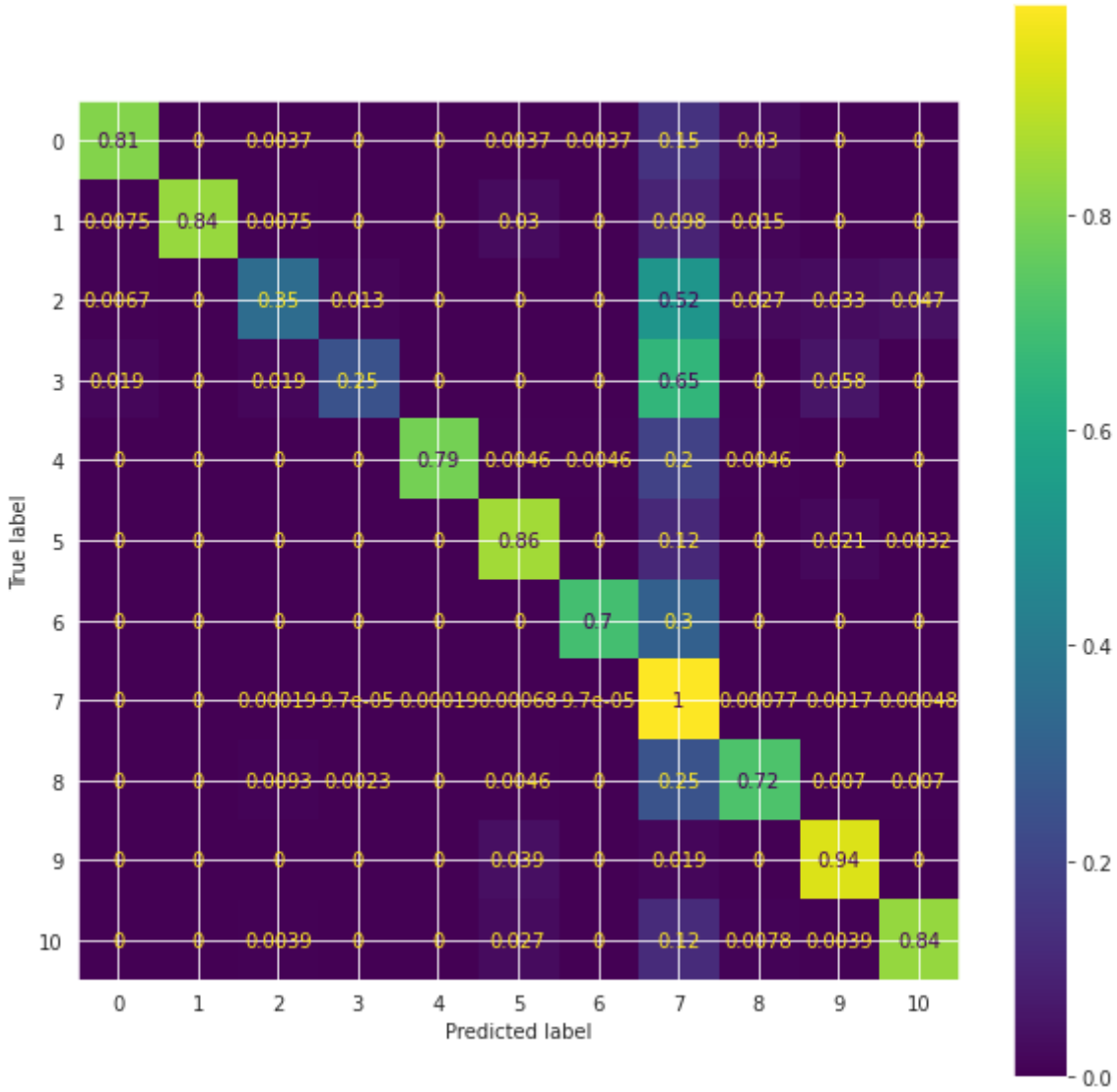
La matriz de confusión con los datos de validación para la red DenseNet tiene los siguientes valores:

	0	1	2	3	4	5	6	7	8	9	10
0	220	0	1	0	0	1	1	40	8	0	0
1	1	112	1	0	0	4	0	13	2	0	0
2	1	0	53	2	0	0	0	78	4	5	7
3	1	0	1	13	0	0	0	34	0	3	0
4	0	0	0	0	171	1	1	43	1	0	0
5	0	0	0	0	0	1085	0	147	0	27	4
6	0	0	0	0	0	0	133	58	0	0	0
7	0	0	2	1	2	7	1	10292	8	18	5
8	0	0	4	1	0	2	0	109	309	3	3
9	0	0	0	0	0	33	0	16	0	798	0
10	0	0	1	0	0	7	0	31	2	1	215

La matriz de confusión con los datos de validación para la red DenseNet representada gráficamente queda como:



Si normalizamos la matriz de confusión respeto a los datos verdaderos para la red DenseNet queda como:



Los mejores resultados se obtienen para la clase 7-Normal, así como las clases 9-Reduced_mucosal_view y 5-Ileocecal_valve. Por su lado los peores resultados se obtienen para las clases 3-Erythematous, 2-Erosion y 6-Lymphangiectasia.

Las clases peor identificadas se repiten para todos los modelos, mientras que la clase mejor identificada es siempre la 7-Normal, que como se comentó, está sobrerepresentada en el dataset.

Atrás – Índice – Siguiente

Project maintained by **lumaro77**
Hosted on GitHub Pages — Theme by **orderedlist**

Conclusiones

- Se ha descrito el objetivo del trabajo, aplicación de redes neuronales artificiales, a un dataset preexistente, el Kvasir-Capsule**, con 3 modelos basados en redes convolucionales y en el uso de transferencia de aprendizaje.

- El modelo ConvoRGB, entrenado completamente con el dataset, presenta métricas entre 80% y 90% según se consideren todas las muestras o se evaluen por métrica y posteriormente se agregen.

- Los modelos ResNet y DenseNet, que hacen uso de transferencia de conocimiento, obtienen resultados que no difieren sustancialmente de los obtenidos por ConvoRGB.

- El dataset está muy desbalanceado, lo que compromete la mejora de resultados con estas u otras metodologías. Además, despierta cierto recelo sobre su exactitud, lo que puede estar incidiendo en la calidad del modelado.

- Las redes neuronales artificiales parecen un instrumento adecuado para la clasificación de imágenes endoscópicas como las del dataset Kvasir-Capsule, con las limitaciones reseñadas previamente.

- De las pruebas realizadas no se ha evidenciado una mejora significativa en los resultados derivados del uso de redes complejas y profundas preentrenadas previamente (transferencia de aprendizaje).

Perspectivas abiertas

- Uso de otras redes neuronales: en los últimos años se han desarrollado nuevas redes neuronales capaces de mejorar los resultados de ResNet y DenseNet. Se suguiere explorar si otras redes son capaces de mejorar los resultados de los modelos de este trabajo.

- Desbalanceado: el dataset empleado está claramente desbalanceado. Se ha evidenciado que supone un inconveniente (a la luz de las métricas) y por tanto una limitación. Se sugiere la búsqueda de metodologías para compensar este balance, como pesos, augmentation selectiva u otra.

- Revisión dataset: desde el declarado desconocimiento del campo del dataset (ciencias de la salud), sería conveniente una revisión del mismo, puesto que no tiene demasiado sentido modelar a partir de datos en los que no se confía.

- Uso de splits: en trabajos previos sobre este dataset se emplearon unos splits que sería interesante estudiar y comparar en resultados con el uso de todo el dataset.

Project maintained by **lumaro77**
Hosted on GitHub Pages — Theme by **orderedlist**

Bibliografía

Libros

- Igual, L.; Seguí, S.; Vitrià, J.; Radeva, P.; Puertas, E.; Pujol, O.; Escalera,S.; Garrido,L.; Dantí, F. Introduction to Data Science – A Python Approach to Concepts, Techniques and Applications. Undergraduate Topics in Computer Science of Springer, 2017.

- Géron, A. Hands-On Machine Learning with Scikit-Learn and TensorFlow. O'Reilly Media, Inc., 2017.

- Ferrie C. & Kaiser S. Neural Networks for babies. Sourcebooks Inc., 2019.

Canales video - youtube

- Dot CSV. link

- Codificando bits. link

- Ringa Tech. link

- Sensio. link