Universidad ORT Uruguay
Facultad de Ingeniería
Posgrado de Analítica en Big Data

Obligatorio Deep Learning

Tanya Cohen - 190941
William Chamorro - 167204
Alfredo Rodríguez - 5515

Profesores: Sergio Yovine / Germán Milano

Diciembre 2020

Introducción y Objetivos

Las redes neuronales convolucionales son un tipo de NN artificiales que son utilizadas, entre otras cosas, para el análisis y reconocimiento de imágenes.

El objetivo de este trabajo es analizar fotos de abejas y clasificarlas por especie y estado de salud, mediante redes neuronales convolucionales.

El denominado Colony Collapse Disorder (CCD) ha sido una gran amenaza para las colonias de abejas en todo el mundo, y afecta la polinización de cultivos alimentarios vitales. La disminución de la población de abejas puede tener consecuencias trágicas, tanto para los seres humanos como para el ecosistema en general. La salud de las abejas ha sido un importante motivo de preocupación para los agricultores y científicos de todo el mundo durante al menos una década, sin embargo, aún no se ha identificado una causa específica para este fenómeno de manera concluyente. En este sentido se han aplicado técnicas de machine learning para clasificar imágenes de abejas según determinados criterios de entrenamiento.
Para este tipo de aplicaciones, donde es de mucha utilidad identificar determinadas características o features de un set de imágenes, las redes convolucionales se adaptan mucho mejor que los algoritmos tradicionales de machine learning e incluso a los MLP (multi-layer perceptron) [1]

Metodología

En primera instancia, se construyen dos clasificadores; uno para la predicción de la sub-especie y otro para la predicción del estado de salud. Se cuenta con las imágenes en formato png y diferentes tamaños, el dataset con las etiquetas correspondientes a la sub-especie y estado de salud de cada abeja, y el dataset con la información a predecir.

En primer lugar, se realiza un análisis exploratorio de datos, luego una partición en train, validation y test. Se eligen los parámetros adecuados para mejorar la arquitectura de la red, en cuanto a capas convolucionales y regularización.

Primero se crearon modelos personalizados individuales, luego los mismos se ensayaron por Transfer learning y por último por Multitasking Learning.

Para la mejora del modelo, siempre teniendo como objetivo lograr una macro average accuray superior al 85% en test, se ensayaron diferentes alternativas.
La macro average recall es una métrica usada en problemas muti-clase como es nuestro caso, y mide el recall promedio por clase. Recordar que recall R se calcula como:

$$R = \frac{T_p}{T_p + F_n}$$

Un macro average recall de 1, significa que las predicciones del modelo son perfectas.

Este valor puede bajar si el modelo solamente performa bien en las clases más frecuentes, y no predice bien para las clases poco frecuentes. Para solucionar esto se utiliza class weights, definiendo un diccionario de mapeo de clases que indica el peso que se le da a cada clase para calcular la función de loss durante el entrenamiento. El propósito es penalizar la clasificación errónea realizada por la clase minoritaria estableciendo un peso de clase más alto y al mismo tiempo reduciendo el peso para la clase mayoritaria.

A continuación se brinda una explicación de los pasos seguidos en la optimización de los modelos.

i) Tunning de los parámetros

Estos parámetros son epochs, learning rate, batch_size y optimizador.
Para mayores epochs, hay un aumento de la performance, pero tal como se puede ver en la práctica, a partir de un determinado número de epochs, no hay reducción de la función de loss en train. Esto hace que el algoritmo de early stopping en todos los casos, corte el proceso de entrenamiento antes de llegar a la cantidad máxima de batchs.
El learning rate debe ser seleccionado tal que sea lo suficientemente alto como para mejorar la velocidad del algoritmo, pero lo suficientemente bajo como para que el método del descenso de gradiente no se "saltee" ningún mínimo en alguna iteración. Se ensayó con un lr = 0.01, 0.001 y 0.0001. Se comprobó que con un learning rate mayor (0.01), la exactitud del modelo cae apruptamente en todos los casos. Con un learning rate menor (0.0001) el tiempo de entrenamiento aumenta drásticamente pero en ciertos modelos mejora sensiblemente el macro average recall.
Respecto al batch_size, un valor muy pequeño puede empeorar la exactitud y además dar como resultado una curva de loss con "dientes de sierra" más pronunciados. Por otro lado, un batch_size muy alto va en contra la velocidad del algoritmo, dado que entrena en cada iteración con mayor cantidad de muestras de train.
Como optimizador las variantes son Adam, SGD o RMSProp. Se probó con SGD y Adam, pero los mejores resultados se obtuvieron con Adam.

ii) Data augmentation

Una CNN tiene la habilidad de aprender features automáticamente a partir de los datos, lo cual es solo posible cuando se dispone de una gran cantidad de datos de entrenamiento. Esto es lo que se trata de lograr con el llamado data augmentation.
A partir del parámetro steps_per_epoch se logra esa cantidad de imágenes aleatorias a partir de un batch. Estas imágenes son generadas aleatoriamente a través de los parámetros rotation, zoom, width_shift_range, height_shift_range, horizontal_flip y vertical_flip. También se probó con el parámetro *shear_range*, que distorsiona la imagen en un eje para crear perspectivas, pero no se obtuvieron mejoras.
En los algoritmos, el parámetro step_per_epoch se calcula en función del tamaño de la muestra del train y del batch_size: *steps_per_epoch = train_X.shape[0] // batch_size,* debido a que, en la mayoría de las combinaciones de estos valores, el algoritmo no puede entrenar. Esto constituye una limitante dado que por lo general se busca un batch_size relativamente alto y un valor de steps_per_epoch también elevado para mejorar el data augmentation.

iii) Topología de red convolucional

Una red de múltiples capas, puede aprender features en varios niveles de abstracción. Por tanto, en general se prefieren redes profundas a las redes más “llanas”.
Sin embargo, lo que se quiere es lograr la red más pequeña posible que obtenga buenos resultados. Las redes muy profundas, son computacionalmente caras de entrenar.
En nuestros ensayos, cuando quisimos agregar alguna etapa convolucional, cada una de estas consta de un layer Conv2D que realiza la convolución propiamente dicha (que es la que detecta y hace el mapeado de las features de las imágenes), una capa de activación con relu (que introduce las propiedades no lineales a la red), y una capa MaxPool2D (que es la que hace la función de pooling o de reducción espacial). También en cuanto al número de features o filtros en cada Conv2D, se fue probando de menos a más, también tratando de alcanzar un buen punto de trade-off entre exactitud y velocidad de entrenamiento.
Otros parámetros a ajustar dentro de las capas Conv2D es el tamaño del kernel, y el tamaño de la ventana en el que toma el máximo la capa MaxPooling2D.

iv) Métodos para evitar el sobreajuste

Además del data augmentation, ya mencionado anteriormente, se realizó early stopping de una fase (con paciencia 10), se ensayaron varios esquemas con Dropout entre 10% y 30% de fracción de dropeo, y regularización L2 en las capas Conv2D. No se apreciaron mejores resultados con este último método.

Resultados y discusión

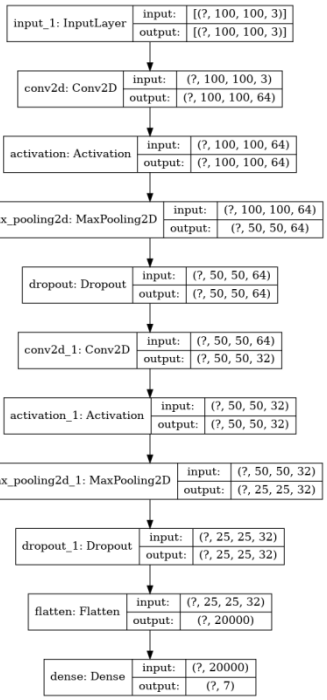
Predicción de sub-especie

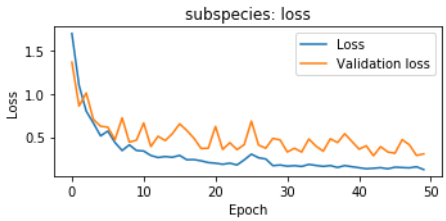
En el modelo de predicción de sub-especie, probamos varias alternativas en cuanto a parámetros y arquitectura: agregar capas convolucionales, luego aumentar el número de features para cada capa Conv2D, aplicando más data augmentation mediante el incremento del parámetro *steps_per_epoch*, y aplicando regularización Dropout y regularización Ridge de norma 2 (earlystopping de fase 1 ya estaba implementada en el código).
Los mejores resultados, se obtuvieron con dos etapas convolucionales, agregándose al final de cada etapa, una capa de Dropout de regularización.
A medida que aumentamos el número de features o filtros de las capas Conv2D, obtuvimos mejoras en las predicciones, hasta que superados los 64 y 32 features en la 1er y 2da capa convolucional respectivamente, no se obtuvieron mejoras.

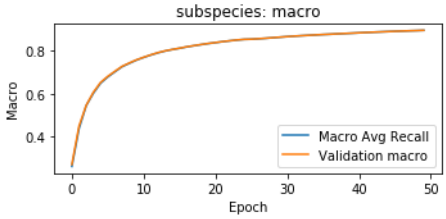
Los mejores resultados se obtuvieron con un dropout de 20% y sin regularización L2, con un batch size de 60, 38 steps per epoch y 50 epochs.
En el cuadro siguiente se muestra un resumen de los distintos ensayos, destacándose el mejor en el que se obtuvo un Macro Average Recall de 94% (Kaggle: 94%).
La tabla siguiente muestra un resumen de los distintos ensayos realizados, destacándose en verde el modelo con mejor resultado.

#capas Conv	# Features	Batch_size	epochs	Steps_per_epoch	dropout	L2 reg	Flip	zoom_range	width_shift_range	height_shift_range	Macro Average Recall
1	8	15	50	30	no	none	True	0.1	0.1	0.1	72%
1	16	15	50	30	no	none	True	0.1	0.1	0.1	79%
1	32	15	50	30	no	none	True	0.1	0.1	0.1	83%
1	64	15	50	30	no	none	True	0.1	0.1	0.1	84%
1	64	15	50	30	0.2	none	True	0.1	0.1	0.1	82%
1	32	15	50	30	0.1	none	True	0.1	0.1	0.1	83%
1	64	15	50	30	0.1	none	True	0.1	0.1	0.1	84%
1	64	15	50	30	0.2	none	True	0.1	0.1	0.1	83%
1	64	20	50	116	0.2	none	True	0.1	0.1	0.1	86%
2	64 + 32	20	50	116	0.2	none	True	0.1	0.1	0.1	92%
2	64 + 32	15	50	155	0.1	0.0001	True	0.1	0.1	0.1	77%
2	64+32	70	50	33	0,2	none	True	0,1	0,1	0,1	93%
2	64+32	60	50	38	0,2	none	True	0,1	0,1	0,1	94%

A continuación se muestran los principales resultados obtenidos con el mejor modelo y un esquema de la arquitectura de la red.



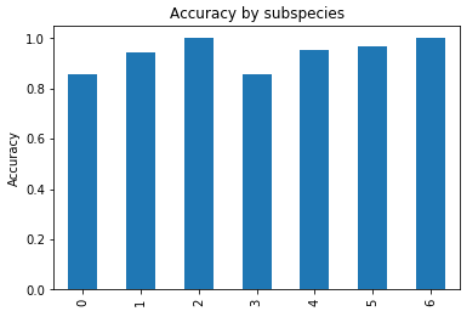




Classification report

	precision	recall	f1-score	support
-1	0.86	0.87	0.87	70
1 Mixed local stock 2	0.54	0.94	0.68	69
Carniolan honey bee	0.96	1.00	0.98	92
Italian honey bee	0.98	0.86	0.92	490
Russian honey bee	0.97	0.95	0.96	110
VSH Italian honey bee	0.88	0.97	0.92	29
Western honey bee	1.00	1.00	1.00	7
accuracy			0.90	867
macro avg	0.88	0.94	0.90	867
weighted avg	0.93	0.90	0.91	867

Loss function: 0.23732790350914001



Analizando las gráficas de loss se ve que no hay sobreajuste, dado que la curva de loss de validation acompaña el decrecimiento de la curva de loss de train. Por otro lado, la macro average recall obtenida, coincide con la de test de Kaggle, lo que es un indicador que nuestro modelo no sobreajusta.

Sin embargo, existe un gap continuo entre ambas curvas de loss. Esto puede ser síntoma de un dataset de train no representativo, es decir que no provee suficiente información para el entrenamiento, relativo al dataset de validation que lo evalúa [2].

En cuanto al tiempo de ejecución, este modelo tarda en entrenarse unos 450 segundos.

Respecto a la exactitud y al recall obtenidos para cada clase, es alta y pareja para todas las subespecies (todas se encuentran por encima del 80%), lo que es un indicador de la buena generalización del modelo.

Predicción de salud

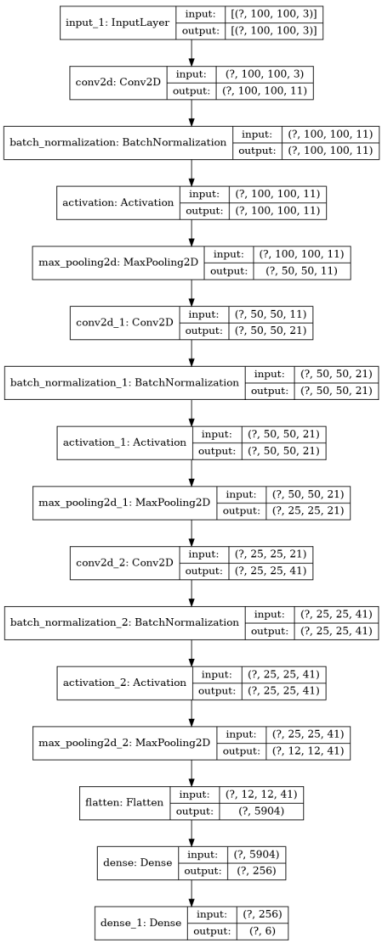
En el modelo de predicción multiclase de salud de las abejas, rápidamente vimos que con el mismo modelo que para sub-especies, no estábamos obteniendo buenos resultados. Como alternativa comenzamos a agregar una 3er etapa convolucional, y también se probaron distintos tamaños de kernel para las diferentes capas Conv2D. También se probó con una capa densa *Fully-Connected* antes de la etapa de salida de decisión, la cual ajusta los coeficientes w_i para crear una representación de probabilidad estocástica de cada clase basada en los mapas de activación generados por las etapas convolucionales [1].

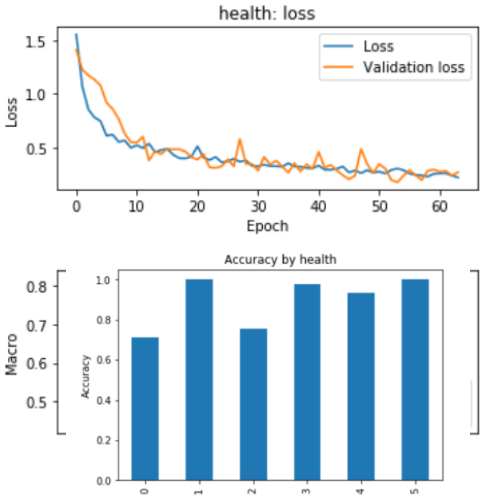
Los mejores resultados se obtuvieron con 3 etapas convolucionales de 11, 21 y 41 features respectivamente, agregando antes de la salida, una capa fully connected de 256 neuronas, alcanzándose un 90% de Macro Average Recall de test y 89.5% en Kaggle. También se agregó una capa de Batch Normalization luego de cada Conv2D.
Batch normalization es un algoritmo que reparametriza el modelo en el sentido que introduce ruido tanto aditivo como multiplicativo en las capas ocultas durante el entrenamiento. El principal propósito de batch normalization es mejorar la optimización, pero el ruido puede tener un efecto de regularización y a veces hace innecesario el dropout [3].

En la tabla siguiente se muestran los resultados y los parámetros de los principales ensayos, destacándose en verde el mejor modelo.

#capas Conv	# Features	FC layer	MaxPool2D	Batch_size	epochs	Steps_per_epoch	dropout	L2 reg	Flip	rotation_range	zoom_range	width_shift_range	height_shift_range	Macro Average Recall	Kaggle
1	32	no	2	15	50	155	0.1	none	True	180	0.1	0.1	0.1	79%	
1	64	no	2	15	50	155	0.1	none	True	180	0.1	0.1	0.1	77%	
1	8	no	2	20	50	30	0.1	none	True	180	0.1	0.1	0.1	65%	
2	32 + 32	no	2	15	50	155	0.1	none	True	180	0.1	0.1	0.1	80%	
2	64 + 32	no	2	15	50	155	0.1	none	True	180	0.1	0.2	0.2	85%	85%
2	48/48 + 64/64	no	2	15	50	155	0.1	none	True	180	0.1	0.2	0.2	79%	
2	64 + 64	no	2	15	100	155	0.1	none	True	180	0.1	0.2	0.2	84%	
2	64 + 32	no	2	15	50	155	0.1	none	False	180	0.1	0.2	0.2	82%	
2	64 + 32	no	2	15	50	155	0.3	none	True	180	0.3	0.3	0.3	83%	
2	64 + 32	no	2	15	50	155	0.3	0.0001	True	180	0.3	0.3	0.3	82%	
3	64 + 32 + 32	no	2	15	50	155	0.1	none	True	180	0.2	0.2	0.2	84%	
3	16 + 64 + 32	no	2	15	50	155	0.1	none	True	180	0.1	0.2	0.2	85%	81%
3	16 + 64 + 32	no	2	15	50	155	0.1	none	True	180	0.1	0.2	0.2	81%	
3	16 + 64 + 32	no	2	60	50	155	0.1	none	True	180	0.1	0.2	0.2	83%	
3	256 + 512 + 256	no	2	60	50	38	0.1	none	True	180	0.1	0.2	0.2	87%	
3	32 + 32 + 64	512	2	60	50	38	0.1	none	True	180	0.1	0.2	0.2	76%	
3	32 + 32 + 64	512	2	20	50	116	0.1	none	True	180	0.1	0.2	0.2	89%	82%
3	32 + 32 + 64	512	2	20	50	116	0.2	none	True	180	0.1	0.2	0.2	89%	85%
3	32 + 32 + 64	512	2	20	50	116	0.2	0.0001	True	180	0.1	0.2	0.2	79%	
3	128 + 128 + 256	512	2	20	50	116	0.2	none	True	180	0.1	0.2	0.2	79%	
2	64 + 64	256	3	20	100	116	0.2	none	True	180	0.1	0.2	0.2	87%	88%
2	16 + 32	256	2	40	150	58	0.2	none	True	180	0.1	0.2	0.2	89%	86%
3	11 + 21 + 41	256	3	50	150	46	0.2	none	True	180	0.1	0.2	0.2	88%	83%
3	11+21+41+Batch_normalization	256	2	50	100	46	0.2 (FC)	none	True	20	0.2	0.3	0.3	90%	89%

A continuación se muestran los resultados obtenidos y un esquema de la red para el mejor modelo.





Classification report

	precision	recall	f1-score	support
Varroa, Small Hive Beetles	0.79	0.71	0.75	69
ant problems	0.95	1.00	0.98	84
few varrao, hive beetles	0.73	0.78	0.75	81
healthy	0.98	0.98	0.98	584
hive being robbed	0.91	0.93	0.92	44
missing queen	1.00	1.00	1.00	5
accuracy			0.94	867
macro avg	0.90	0.90	0.90	867
weighted avg	0.94	0.94	0.94	867

Loss function: 0.15982720255851746

Viendo las curvas de loss, se ve que no hay sobreajuste, pero la curva de loss de validation presenta movimientos ruidosos alrededor de la curva de loss de validation. Esto es un síntoma de un dataset de validation no representativo. Es decir, que el dataset de validation no provee suficiente información para evaluar la habilidad de generalizar del modelo [2].
El macro average recall obtenido es muy similar al resultado en kaggle, lo que es un indicador de no sobreajuste.

Respecto al tiempo de ejecución, el modelo demora en entrenar 2.050 segundos.
En cuanto a la exactitud y al recall de las clases individuales, hay mayor variabilidad entre los distintos tipos de problemas de salud. Lo que hace que el modelo no sea bueno generalizando. Una posible solución a esto, es definiendo un nuevo diccionario de pesos para el entrenamiento de la función de loss ajustando los “class weights” de modo de dar mayor peso para aquellas clases donde los valores de accuracy y recall sean más bajos.

Transfer learning

Mediante esta técnica, en lugar de comenzar el proceso de aprendizaje desde cero, se comienza haciendo uso de patrones o modelos pre-entrenados que se han aprendido al resolver un problema diferente. Un modelo pre-entrenado es un modelo que fue entrenado con un conjunto de datos de referencia para resolver un problema similar al que queremos abordar. Debido al coste computacional del entrenamiento de tales modelos, así como en la complejidad a la hora de elegir la arquitectura óptima, Transfer Learning de modelos bien conocidos y precisos, se ha convertido es una práctica común. En nuestro caso, se utilizaron los modelos VGG-16, VGG-19 y Xception, para los modelos de Sub especie y Health, y luego solo se entrena la capa de clasificación.
Para la elección del modelo pre entrenado, se ensayaron los 3 mejores según Keras, en cuanto a la performance en clasificación de imágenes (el ranking que presenta Keras corresponde a la perfomance de los modelos usando el dataset de ImageNet como validation) [4].

Para ambos modelos, el VGG 16 pre-entrenado fue el que dio mejores resultados. Una de las condiciones de la validez para este modelo es que los datos de entrenamiento del VGG 16 sean mucho mayores que los datos de entrenamiento de nuestro modelo. Esto se cumple dado que el VGG 16 fue entrenado con más de 14 millones de imágenes en el dataset de train.

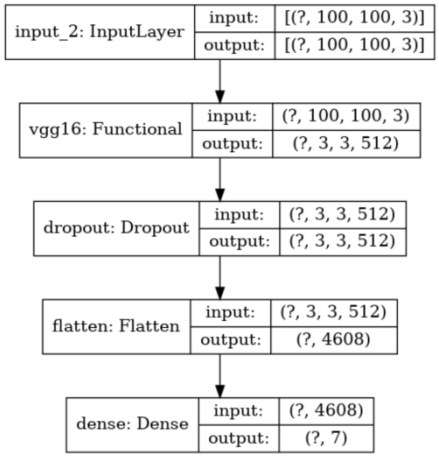
Se probaron diferentes estrategias, variando los valores de batch size, epochs, steps per epoch, y también la regularización por dropout.

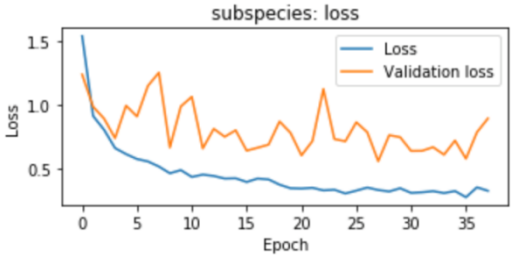
i) Subespecie

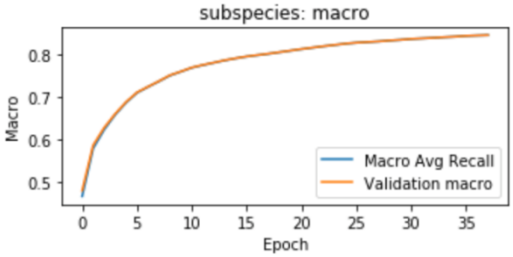
Para el modelo de subespecie, se logró un macro average recall de 87% con un batch size de 30, 50 epochs, 77 steps per epoch, y dropout 0.1, sin alterar los parámetros de transformación de imágenes para el data augmentation.
En la tabla siguiente se muestran los distintos ensayos realizados, destacándose en verde, el mejor modelo.

Modelo	Batch_size	epochs	Steps_per_epoch	pooling	dropout	flatten	Flip	zoom_range	width_shift_range	height_shift_range	Macro Average Recall
VGG16	20	50	116	none	none	sí	True	0.1	0.2	0.2	81%
VGG16	30	50	77	none	none	sí	True	0.1	0.2	0.2	83%
VGG16	30	50	77	none	none	sí	True	0.1	0.1	0.1	84%
VGG16	50	50	46	none	none	sí	True	0.1	0.1	0.1	86%
VGG16	50	50	46	sí	0.2	sí	True	0.1	0.1	0.1	79%
VGG16	30	50	77	sí	0.1	sí	True	0.1	0.1	0.1	80%
VGG16	30	50	77	none	0.1	sí	True	0.1	0.1	0.1	87%
VGG16	30	50	77	global_average_pooling2	0.1	no	True	0.1	0.1	0.1	78%
XCEPTION	30	50	77	global_average_pooling2	0.1	no	True	0.1	0.1	0.1	81%
XCEPTION	30	50	77	none	0.1	sí	True	0.1	0.1	0.1	76%
XCEPTION	20	50	116	global_average_pooling2	0.1	no	True	0.1	0.1	0.1	78%
VGG19	20	50	116	global_average_pooling2	0.1	no	True	0.1	0.1	0.1	75%

Se muestran a continuación, los resultados de las predicciones del modelo y un esquema de la arquitectura de la red.





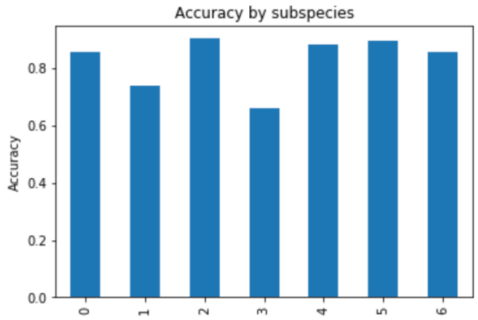



```
Classification report
                      precision    recall  f1-score   support

                  -1       0.50      0.89      0.64        70
1 Mixed local stock 2       0.68      0.75      0.72        69
  Carniolan honey bee       0.94      0.91      0.93        92
     Italian honey bee      0.91      0.72      0.81       490
     Russian honey bee      0.85      0.90      0.88       110
VSH Italian honey bee       0.47      0.93      0.63        29
     Western honey bee      0.35      1.00      0.52         7

             accuracy                           0.79       867
            macro avg       0.67      0.87      0.73       867
         weighted avg       0.84      0.79      0.80       867

Loss function: 0.515125036239624
```



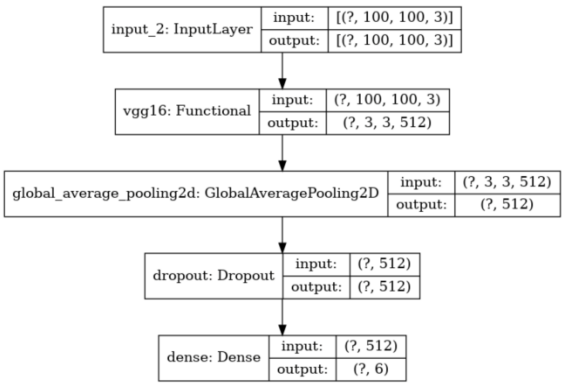
La gráfica de loss muestra que existe un sobreajuste ya desde los primeros epochs, dado que la curva de loss de validation se va separando de la de train, que disminuye a medida que aumentan los epochs. Nuestro modelo tomó los valores de w_i para el epoch 28, por lo que es de esperarse algo de sobreajuste en las predicciones. Dado que en este modelo ya está implementado early stopping y dropout en la parte entrenable, las alternativas que quedan para disminuir la varianza es incrementando el data augmentation (más steps por epoch, mayor shift en la generación aleatoria de imágenes), pero tal como se ve en la tabla comparativa de ensayos (1er fila), esto no arrojó un buen macro average recall.
Se observa además que la exactitud y recall entre las distintas subespecies, es bastante despareja. Esto nuevamente hace que el modelo no generalice lo suficiente para cualquier dataset.
El tiempo de entrenamiento fue de 610 segundos.

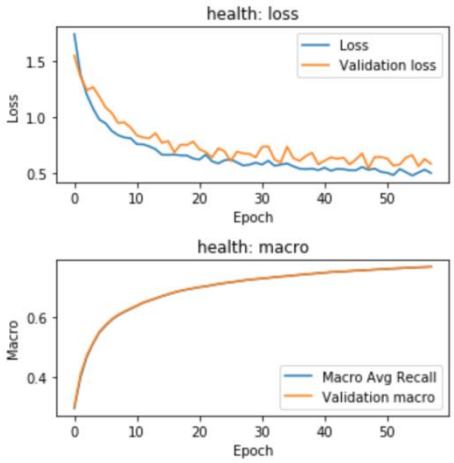
ii) Health

Para el modelo de health, se logró un macro average recall de 85% con un batch size de 10, 160 epochs, 233 steps per epoch, dropout 0.1, sin alterar tampoco los parámetros de transformación de imágenes de data augmentation.

Modelo	Batch_size	epochs	Steps_per_ep	pooling	dropout	flatten	Flip	zoom_range	width_shift_r	height_shift_	Macro Average Recall
VGG 16	20	50	47	global_average_po	0.1	si	VERDADERO	180	0.1	0.1	80%
XCEPTION	20	50	47	global_averag	0,1	si	VERDADERO	180	0,1	0.1	79%
XCEPTION	70	50	47	global_averag	0,1	si	VERDADERO	180	0,1	0.1	80%
XCEPTION	20	100	23	global_averag	0,1	si	VERDADERO	180	0,1	0.1	80%
VGG 16	70	100	23	global_averag	0,1	si	VERDADERO	180	0,1	0.1	81%
VGG 16	70	100	23	global_averag	0,1	si	VERDADERO	180	0,2	0.2	81%
VGG 16	70	100	23	global_averag	0,1	si	VERDADERO	180	0,5	0.5	79%
VGG 16	70	100	23	global_averag	0,5	si	VERDADERO	180	0,5	0.5	71%
VGG 16	70	100	23	global_averag	0,5	si	VERDADERO	180	0,2	0.2	79%
VGG 16	50	100	23	global_averag	0,1	si	VERDADERO	180	0,2	0.2	79%
VGG 19	50	100	23	global_averag	0,1	si	VERDADERO	180	0,2	0.2	73%
VGG 16	70	150	16	global_averag	0,1	si	VERDADERO	180	0,2	0.2	81%
VGG 16	50	150	16	global_averag	0,1	si	VERDADERO	180	0,2	0.2	82%
VGG 16	25	150	16	global_averag	0,1	si	VERDADERO	180	0,2	0.2	83%
VGG 16	15	150	16	global_averag	0,1	si	VERDADERO	180	0,2	0.2	83%
VGG 16	15	150	16	global_averag	0,2	si	VERDADERO	180	0,2	0.2	82%
VGG 16	15	150	16	global_averag	0,1	si	VERDADERO	180	0,1	0.1	83%
VGG 16	10	150	16	global_averag	0,1	si	VERDADERO	180	0,1	0.1	84%
VGG 16	10	180	13	global_averag	0,1	si	VERDADERO	180	0,1	0.1	80%
VGG 16	10	160	15	global_averag	0,1	si	VERDADERO	180	0,1	0.1	85%
VGG19	20	100	116	global_averag	0.1	no	True	0.1	0.1	0.1	79%

Se muestran los principales resultados obtenidos en la predicción.



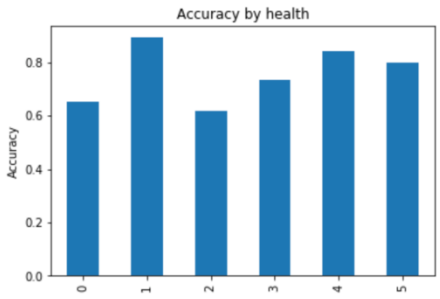



```
Classification report
                          precision    recall  f1-score   support

Varroa, Small Hive Beetles       0.56      0.70      0.62        69
             ant problems       0.88      0.92      0.90        84
  few varrao, hive beetles       0.49      0.73      0.58        81
                   healthy       0.97      0.80      0.88       584
        hive being robbed        0.58      0.93      0.71        44
            missing queen        0.24      1.00      0.38         5

                 accuracy                           0.81       867
                macro avg       0.62      0.85      0.68       867
             weighted avg       0.86      0.81      0.82       867

Loss function: 0.5766732096672058
```



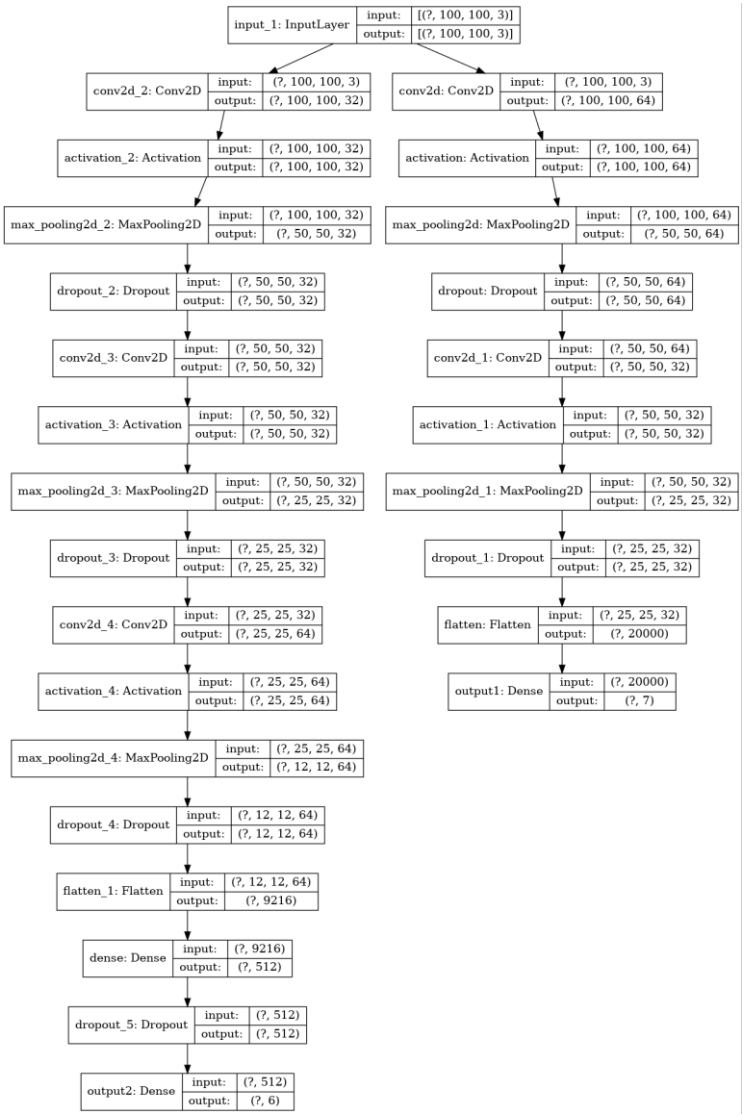
Del análisis de las curvas de loss se ve que no hay sobreajuste.
Sin embargo se observa bastante diferencia en la exactitud y recall entre los distintos tipos de estado de salud. Esto, como ya se mencionó puede hacer que el modelo no generalice lo suficiente.
Este modelo demoró en entrenarse, 735 segundos.

Multi tasking

Multi tasking es un tipo de aprendizaje en el que se resuelven varias tareas de aprendizaje al mismo tiempo. Esto puede resultar en una mejora en la eficiencia del entrenamiento y la precisión de la predicción para los modelos específicos de la tarea, en comparación con el entrenamiento de los modelos por separado.

Para esta parte se ensayaron algunos modelos. Básicamente se incluyeron los mejores modelos de subespecie y health por separado, cada uno en una rama del nuevo modelo multitask, y se entrenó todo junto considerando una única función de loss.

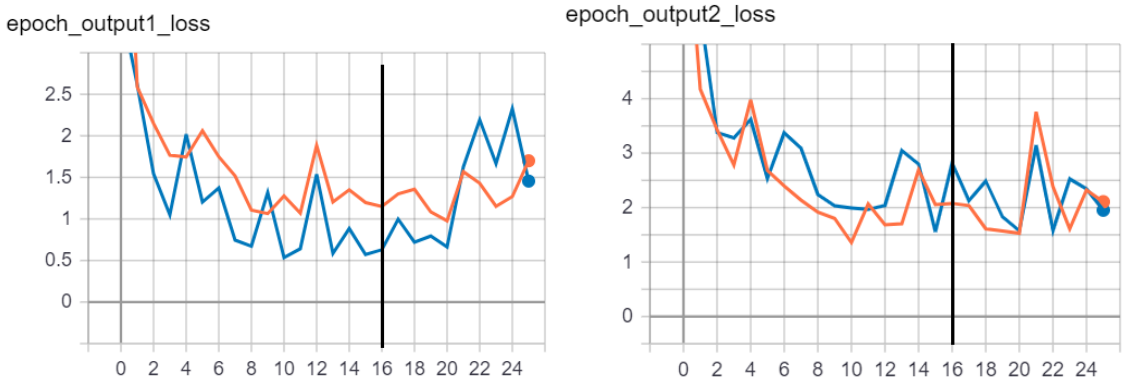
Nuevamente con health, aquí tuvimos dificultades. Incluyendo el mejor modelo por separado de health, en multitask solamente obtuvimos un 30% de macro average recall. El mejor modelo fue incluyendo en la rama de health, a una red con tres capas convolucionales con 256, 512 y 256 de #features respectivamente.
En la figura siguiente, se muestra el modelo implementado.



En la siguiente tabla, se muestran los resultados para los diferentes ensayos, destacándose en verde el mejor modelo.

Modelo supespecie	Modelo Health	Batch_size	epochs	Steps_per_ep	learning rate	dropout	Flip	zoom_range	width_shift_r	height_shift_	Macro Average Recall subespecie	Macro Average Recall health
64 + 32	32 + 32 + 64	70	160	150	0,001	0.2 / 0.1	True	180	0.2	0.2	92%	85%
64 + 32	256 + 512 + 256	70	50	33	0,001	0.2 / 0.1	True	180	0.2	0.2	92%	77%
64 + 32	11+21+41+Batch_normalization	50	100	46	0,0001	0.2 / 0.2	True	20	0.2	0.3	85%	30%

En las gráficas siguientes se ven las curvas de loss de train (rojo) y validation (azul) para output1 (subespecies) y output 2 (health), correspondientes al mejor modelo. También se marca con una recta, el epoch #16 que es donde durante el entrenamiento se logró minimizar la función de loss y por tanto será en el que son calculados los coeficientes.



Observando las gráficas, se ve que en el modelo de subespecie, tiende a haber un sobreajuste luego del epoch 20, mientras que para health, ambas curvas son decrecientes y tienden a estabilizarse, más allá del pico importante en el diente de sierra que se da en el epoch 21 (indica que el mecanismo de descenso de gradiente no logró bajar la función de Loss en esa iteración, la probabilidad de ocurrencia de estos dientes pronunciados de diente de sierra, aumenta al disminuir el batch_size).
El tiempo de entrenamiento fue de 1.189 segundos.
A priori, analizando las gráficas, la varianza no sería un problema para los modelos obtenidos. En cuanto a las curvas de subespecies, se ve que la loss de validation está por debajo de la de train. Esto indica que el dataset de validation puede ser más fácil de predecir para el modelo que el conjunto de datos de entrenamiento [2].

Conclusiones generales

En el siguiente cuadro se muestra un comparativo con los resultados de los mejores modelos para los modelos individuales de subespecie y health, transfer learning y multitask.

Modelo	Macro average recall	Tiempo de entrenamiento
Arquitectura propia subespecie	94%	450 seg
Arquitectura propia health	90%	2.050 seg
Transfer Learning subespecie	87%	610 seg
Transfer Learning health	85%	735 seg
Multi-task subespecie	92%	1.189 seg
Multi-task health	85%	1.189 seg

Las arquitecturas propias o modelos individuales, fueron las que mejor macro average recall obtuvieron.

Con transfer learning no se obtuvieron mejoras, dado que los modelos pre entrenados que usamos no generalizaron bien para nuestro caso. Sin embargo se obtuvo una mejora en los tiempos de ejecución, debido a que solamente se entrenó una parte mínima de la red.
Por otro lado, con transfer learning tal como se vio, empeoran los valores de recall y accuracy para las clases individuales tanto en subespecie como en health. Este efecto como fue comentado anteriormente, se puede tratar de minimizar definiendo un nuevo diccionario de pesos para el entrenamiento de la función de loss mediante el ajuste de los "class weights" de modo de dar mayor peso para aquellas clases donde los valores de accuracy y recall sean más bajos.

Con multitask learning, si bien el tiempo de entrenamiento es mayor que cada modelo individual por separado, hay que tener en cuenta que los dos modelos se entrenan a la vez. Por lo que es más rápido el entrenamiento en multitasking, que entrenar ambos modelos por por separado.
El modelo multitask arroja buenos resultados solo si las tareas aprendidas conjuntamente están relacionadas entre sí. Una tarea se refiere al aprendizaje de una salida a partir de una única fuente de entrada [5]. Dado que el modelo de multitask fue que el que peores resultados arrojó, es probable que las tareas entre subespecie y health no se encuentren relacionadas.

Cabe aclarar que los tiempos de entrenamiento para cada modelo, fueron calculados todos en las mismas condiciones, dado que las notebooks fueron ejecutadas en el mismo servidor (CPU) del cluster de la ORT.
En cuanto a la varianza, tal como fue analizado para casa caso, no tenemos sobreajuste salvo en el modelo de Transfer Learning con subespecies, para el cual es altamente probable que las predicciones presenten una varianza importante.

En la elección del mejor modelo consideramos que hay dos aspectos a tener en cuenta, el macro average recall y el tiempo de ejecución.
Por lo tanto si valoramos obtener un macro average recall alto nos quedaríamos con los modelos de arquitectura propia tanto para subespecie como para health. En cambio si podemos sacrificar algo del rendimiento para obtener una mejora importante en los tiempos de ejecución, sobre todo en heath, nos quedaríamos con el modelo de multitask ya que reduce los tiempos de entrenamiento a menos de la mitad, y no presentan sobreajuste.

Referencias

[1] A Deep Learning Approach to Recognizing Bees in Video Analysis of Bee Traffic. Astha Tiwari. Utah State University. 2018.
[2] How to use Learning Curves to Diagnose Machine Learning Model Performance. https://machinelearningmastery.com/learning-curves-for-diagnosing-machine-learning-model-performance/
[3] Deep Learning. Ian Goodfellow, Yoshua Bengio, Aaron Courville.
[4] Keras applications. https://keras.io/api/applications/
[5] A Brief Review on Multi-Task Learning. Kim-Han Thung, Chong-Yaw Wee