COMPETICIÓN DE CLASIFICACIÓN

Visión por computador

Alexis Gómez Chimeno, David García Lleyda y Álvaro Martínez Parpolowicz
MAIS 5A

1. Temática y características de la clasificación realizada y su justificación.

b. ¿Qué tipo de clasificación se ha aplicado? ¿Por qué?

Se ha decidido emplear una clasificación multi-etiqueta. Existen varios motivos detrás de esta elección. El primero de los cuales está directamente relacionado con la originalidad del propio problema, comparándolo con problemas anteriormente desarrollados en el grado. Hasta el momento se han desarrollado varias redes neuronales para la clasificación binaria y multi-clase. Esta es entonces la primera oportunidad, al menos de manera curricular, de resolver un problema de clasificación multi-etiqueta.

El segundo motivo que justifica esta decisión estaría relacionado con la dificultad que acarrea el propio problema. Se trata de un reto nuevo que ha despertado en el equipo la necesidad de buscar información adicional, plantear diferentes modelos y evaluar varias opciones para tomar decisiones desde la forma en la que reetiquetar el conjunto de datos hasta la arquitectura de la red.

Por último, los criterios de calificación del propio ejercicio, y, un estudio preliminar de las intenciones del resto de grupos, han influido también en la decisión. Aprovechando la esencia de competición del ejercicio, se ha buscado la diferenciación respecto al resto de equipos.

a. ¿Qué objetos habéis decidido clasificar? ¿Por qué?

Se han clasificado un total de cinco objetos diferentes, definidos por las siguientes clases: ‘Animal', 'Car', 'Cycle', 'Person' y 'Water'. Aunque no todas estas categorías están directamente relacionadas con el propio concepto de deporte, si que es cierto que todas las imágenes del dataset empleado contienen al menos uno (o más) de estos objetos. Esto permite inicialmente que la clasificación multi-etiqueta sea posible.

Sin embargo, la elección concreta de estas cinco clases se basa también en la creencia del equipo que la red sería capaz de clasificar bien estos objetos. Esta creencia está fundada en el hecho que todos los objetos presentan patrones muy distintos entre ellos. Es decir, el agua tiene forma, color y matices muy diferentes a los que tiene una persona. Debido a este motivo, otras categorías, como la separación de la etiqueta persona en hombre-mujer, fueran descartadas.

Aún así, estas clases pueden no resultar evidentes a la hora de que un humano real clasifique las imágenes en base a esta separación. Por eso se hacen las siguientes aclaraciones:

> Dentro de la clase ‘Animal’ se incluyen animales cuadrúpedos terrestres (que tienden a estar relacionados con el deporte asociado). Animales como caballos, perros y toros entran en esta categoría. Quedan descartados otros animales como los pájaros por presentar patrones esencialmente distintos a los primeros (y ser circunstanciales de cada foto y no de un deporte en cuestión).
>
> Dentro de la clase ‘Car’ se incluyen todo tipo de coches: deportivos, de Fórmula 1, de rally e incluso coches de caballos.
>
> Dentro de la clase ‘Cycle’ se incluyen todo tipo de ciclos como motocicletas o bicicletas. Además, se probó a incluir las sillas de ruedas por presentar patrones similares y los resultados fueron suficientemente satisfactorios como para no retractarse de esta decisión.

En las clases de 'Person' y 'Water' entran todo tipo de imágenes en las que aparezca un humano en cualquier tipo de postura y agua en estado líquido respectivamente.

c. ¿Se han realizado modificaciones al dataset? Indica cuales y por qué.

Por un lado, para este problema en concreto ha sido necesario redefinir las etiquetas del dataset entero. Cada imagen ha sido incluida en una o varias de las clases mencionadas en el apartado anterior.

Para hacer esta separación se ha optado por diferentes métodos. En primer lugar, se pensó en utilizar un modelo externo preentrenado que hiciera esta separación. Se extrajeron varios modelos de Hugging Face y modelos YOLO de la librería Ultralytics. Atendiendo al concepto de "Garbage In Garbage Out" y observando que no todas las clasificaciones eran perfectas, se decidió hacer el reetiquetado a mano para asegurar que el rendimiento no se viera afectado por este factor.

Por otro lado, en cuanto a modificaciones como tal de la cantidad de imágenes pertenecientes al dataset, se ha recurrido a técnicas de data augmentation para añadir imágenes artificiales al conjunto. Las transformaciones que se han aplicado a estas imágenes artificiales han sido rotación y reflejo sobre el eje vertical (horizontal flip). No se ha incluido el reflejo sobre el eje vertical porque pocas imágenes eran fotos aéreas, y, en la mayoría, la referencia del horizonte marca la lógica del propio deporte. Es decir, es imposible encontrar entre las imágenes de test un ejemplo de ciclismo en el que el cielo esté en la parte inferior de la foto y el ciclista esté cabeza abajo. Sin embargo, es igual de probable encontrar a un ciclista mirando hacia la derecha de la imagen que hacia la izquierda. Por esta misma razón, la cantidad de rotación con la que se han creado nuevas imágenes tampoco es muy pronunciada.

El aumento del conjunto de datos permitió obtener mayor cantidad de ejemplos de imágenes con la etiqueta 'Car' (clase minoritaria) sin alterar la distribución inicial.

2. Decisiones en cuanto a la arquitectura de la CNN.

a. ¿Cuántas capas tiene vuestro modelo? ¿Por qué?

El modelo finalmente presentado tiene un total de 21 capas: 1 de entrada con los parámetros por defecto de Keras, 19 capas ocultas, y, una capa de salida. El desglose de las capas ocultas intermedias es el siguiente:

Por cada bloque que sigue a la capa de entrada, se han introducido las siguientes capas:

- 2 capas convolucionales apiladas con el mismo número de filtros, la misma dimensión del kernel, función de activación relu y padding same. Apilar las capas convolucionales aumenta el campo receptivo, lo que permite a las capas más profundas encontrar características más globales.
- 1 capa de normalización (BatchNormalization) con el objetivo de estabilizar el modelo y agilizar la convergencia. Aún así, dado que el uso de la regularización, en el ámbito de las CNN, permite aumentar la invarianza ante el cambio de las imágenes; una posible mejora del modelo sería añadir aún más regularización para mejorar los resultados en la fase de entrenamiento.

- 1 capa de pooling para reducir las dimensiones (cantidad de información pasada a la siguiente capa). Con esto se consigue reducir la cantidad de cálculos necesarios, aumentando la velocidad de cómputo.
- No se ha introducido capa de Dropout. En un principio sería circunstancial hasta que fuera necesario o no evitar el overfitting. Sin embargo, la introducción de una capa de Dropout podría mejorar el comportamiento de generalización de la red, reduciendo de forma directa el overfitting.

Este bloque o conjunto de capas se ha repetido un total de cuatro veces de manera encadenada. Además, se han ido cambiando el número de filtros y la dimensión del kernel en cada pareja de capas convolucionales apiladas:

Por un lado, se ha ido aumentando la cantidad filtros de cada pareja al doble de la cantidad de filtros del bloque anterior. Comenzando en el primero bloque por 32. Basamos esta decisión en el hecho de que a medida que se profundiza en la red neuronal, el número de canales nuevos de los datos irá variando conforme al número de filtros que se pongan. Esto hace que se deba ir aumentando el número de filtros a medida que profundizamos en la red ya que permite evitar la pérdida de información a la hora de que los filtros evalúen los consecutivos mapas de características generados. En otras palabras, el tamaño de la ventana debe ir aumentado a medida que profundicemos para que las capas finales sean capaces de tener una mejor representación a alto nivel de las estructuras de la imagen.

Por otro lado, se han ido variando las dimensiones de los kernels empleados en las capas convolucionales entre 3x3 y 5x5. Estas dos dimensiones empleadas son comunes en la mayoría de la literatura revisada, aunque, la elección de una u otra en cada capa se debe al resultado de experimentación empírica. El modelo final alterna bloques con capas de 3x3 con bloques de capas 5x5.

A continuación, se produce el aplanado y se introduce una capa densa con tantas neuronas como píxeles tuviera la imagen original. En este caso 64x64 neuronas en total.

Para tratar de reducir y resumir, en un espacio de características latentes, la información de la capa anterior se ha introducido otra capa densa intermedia de 256 neuronas.

Por último, una capa densa de cinco neuronas (una por cada etiqueta) y función de activación sigmoide. Esto permite discernir de manera binaria la presencia o no de cada una de las etiquetas en la imagen original.

Cabe destacar que las capas densas van acompañadas de regularización L2. Esto se podría haber introducido en las capas convolucionales, aunque no se ha llegado a probar porque los resultados obtenidos eran buenos.

b. ¿Qué modificaciones habéis introducido? ¿Por qué?

El modelo comenzó siendo más simple que el presentado y se fueron haciendo modificaciones en función de los problemas que se iban presentando. Empezó con una arquitectura simple de tres capas convolucionales seguidas de sus capas de regularización. Al final de esta red había dos capas densas (aplanado y salida).

Lo primero que fue imprescindible fue introducir las capas de pooling para reducir el tiempo de entrenamiento, entre otros, porque iban a ser necesarios muchos entrenamientos para refinar la arquitectura.

A continuación, fue necesario lidiar con un bajo rendimiento de la red. Primero se aumentó el tamaño de la ventana de las capas mencionadas. Después se recurrió al uso de las capas apiladas. Como se ha mencionado antes, aumentan el campo receptivo a la vez que evitan tener que utilizar filtros mayores para las primeras capas (dado que toda convolución es siempre separable en dos convoluciones más simples). Esto es a su vez, paralelizable, y, por lo tanto, más eficiente.

Entre tanto, como se menciona al principio de esta memoria, se modifica el dataset con técnicas de data augmentation. Aumentando la complejidad de la red y el conjunto de entrenamiento se consiguen reducir los síntomas de underfitting del modelo.

Además, se introdujo un callback adicional ("Reduce LR on Plateau") para reducir la tasa de aprendizaje o learning rate, de manera lineal cuando no se produjeran mejoras en épocas sucesivas. De esta manera, cada tres épocas consecutivas sin que se produzcan mejoras de f1-score, se reduce la learning rate al 20% de su valor. Se han probado diferentes valores de reducción hasta 50% pero no se han obtenido mejores resultados.

Un aspecto de especial interés para este estudio fue la modificación de la topología de la red empleando optimizadores de hiperparámetros. La librería empleada ha sido Optuna. Sin embargo, después de implementarlo fue necesario buscar soluciones distintas. El equipo carece de la capacidad de cómputo suficiente para ejecutarlo suficientes iteraciones como para que mejorase el rendimiento de la red neuronal de manera más efectiva que el propio equipo. Es por eso, que, aunque se adjunta también el código empleado, la mayoría de las modificaciones han estado basadas en los conocimientos propios y la revisión de literatura.

c. Justificación de las métricas, función de pérdida y optimizador.

De acuerdo con los criterios de evaluación de la práctica, la principal métrica monitorizada ha sido el F1-score. Para agregar las métricas de cada una de las etiquetas se ha utilizado "average=macro". El valor de este parámetro resulta en un promediado de las métricas de F1-score de cada una de las etiquetas. Al no ser ponderado no tiene en cuenta el desbalanceo en la cantidad de imágenes de cada etiqueta. Esto permite entrenar el modelo sin que la métrica de f1-score de la clase con mayor soporte disfrace los resultados globales obtenidos.

Aun así, se han hecho las comprobaciones habituales monitorizando el valor de la función de pérdida en entrenamiento. Asegurar que esta función decrece tanto en entrenamiento como en validación nos permite comprobar que la red está aprendiendo.

Por otro lado, se han estudiado también las métricas en las que se descompone la F1-score (precisión y recall) para detectar posibles problemas.

Se ha elegido la funció Binary-Cross-Entropy. Esto se debe a que uno de los objetivos principales es que la función de pérdida nos aporte, para el caso de uso realizado, una probabilidad de pertenencia a cada una de las clases dentro de la etiqueta. Por este motivo necesitamos una función cuyo resultado esté acotado de forma probabilística. Se podría pensar que por esta razón también podemos utilizar alguna métrica habitual, como la f1 score, como función de ganancia de la red y maximizarla. No obstante, estas funciones no están pensadas para ser utilizadas como

tal, puesto que al evaluar su gradiente pueden que tenga comportamientos tanto computacional como matemáticamente ineficientes. Es por ello, por lo que se utiliza una función de pérdida habitual que esté pensada para trabajar con el gradiente. Se ha de mencionar que cuando se habla de comportamientos relacionados con el gradiente, para este caso, nos referimos tanto a manera analítica como numérica.

En cuanto al optimizador empleado, se ha escogido Adam. Se han hecho comparaciones empíricas con RMSprop y con los resultados se ha concluido que el primero es mejor para este caso. Como se ha mencionado antes, se han introducido también mejoras sobre la tasa de aprendizaje del optimizador usando el callback "Reduce LR on Plateau".

3. Metodología y resultados, incluyendo gráficas que muestren claramente el rendimiento de la red implementada.

a. Hiperparámetros y selección de éstos.

En este apartado se comentarán exclusivamente los hiperparámetros que no se han mencionado ya a lo largo de la memoria. Es decir, no se repetirá el porqué de la elección de valores como el número de filtros de cada capa, el número de capas o el tamaño de los kernels.

Para empezar, se comenzó usando como tamaño de imagen el mínimo de entre todas las imágenes del dataset 224x224. Más adelante se comprobó que los resultados usando un tamaño de 64x64 apenas mostraba diferencias en rendimiento pero sí agilizaba el proceso de entrenamiento sustancialmente.

Se utiliza un inicializador de pesos del kernel en las capas densas que sigue una distribución normal de media 0 y sigma 0.05. Se podía haber probado con distintos inicializadores, pero con este se obtenían buenos resultados y no se probaron más.

Para el Early stopping se ha ido refinando el valor de la paciencia hasta alcanzar el valor de 8. Con esto se consiguió un buen equilibrio entre entrenamiento y tiempo empleado. Así se pudo ampliar la cantidad de epochs hasta 40 sin que impactara significativamente en el tiempo de entrenamiento.

El Batch size o tamaño del lote se estableció en 32 también por motivos de velocidad de entrenamiento.

Para el Step size de train y de validation se ha utilizado el número de muestras totales, de cada uno respectivamente, dividido por el batch size empleado (usando la parte entera del resultado). Se trata de una práctica habitual en este tipo de redes. Su imitación ha dado buenos resultados.

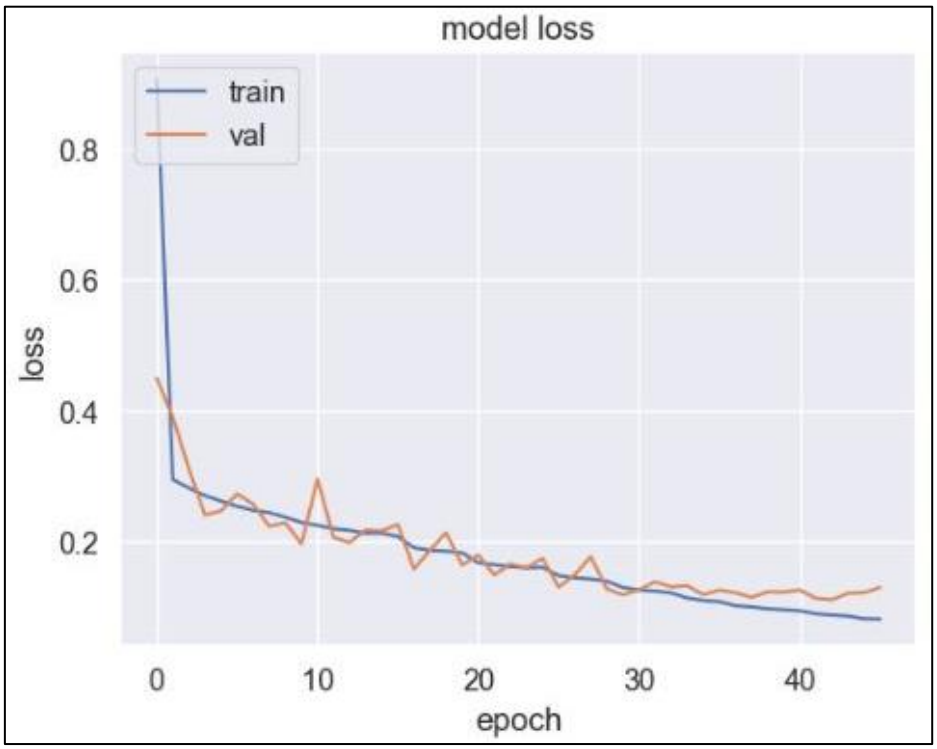
Parameters (24)

Name	Value
activation	relu
batch_size	32
callbacks	[<keras.callbacks.ModelCheckpoint object at 0x000002416843AE00>, <keras.callbacks.EarlyStopping object at 0x00000241SFCF6290>, <keras.callbacks.ReduceLROnPlateau object at 0x0000024168439AE0>]
epochs	40
first_big_dense_layer	3
fourth_big_dense_layer	3
image_size	(64, 64)
initializer	random_normal
kernel_size_first_layer	(3, 3)
kernel_size_fourth_layer	(5, 5)
kernel_size_second_layer	(5, 5)
kernel_size_third_layer	(3, 3)
loss	binary_crossentropy
neurons_dense_layer	4096
neurons_dense_layer2	256
neurons_dense_layer3	5
neurons_first_layer	32
neurons_fourth_layer	256
neurons_second_layer	64
neurons_third_layer	128
optimizer	adam
regularizer	l2(0.01)
second_big_dense_layer	2
third_big_dense_layer	3

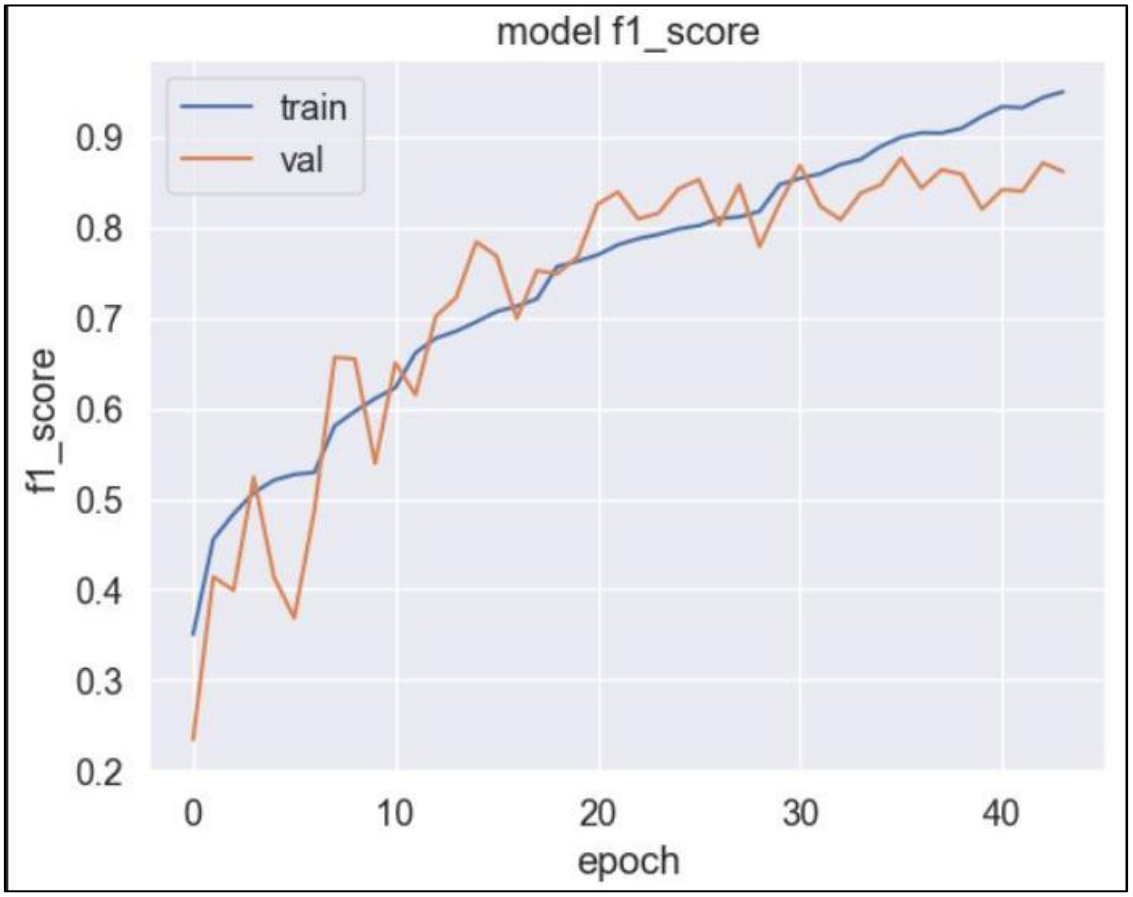
b. Resultado de las métricas seleccionadas

Name	Value
f1_test	0.8847065315918767
precision_test	0.9020854897065049
recall_test	0.8681016374929419

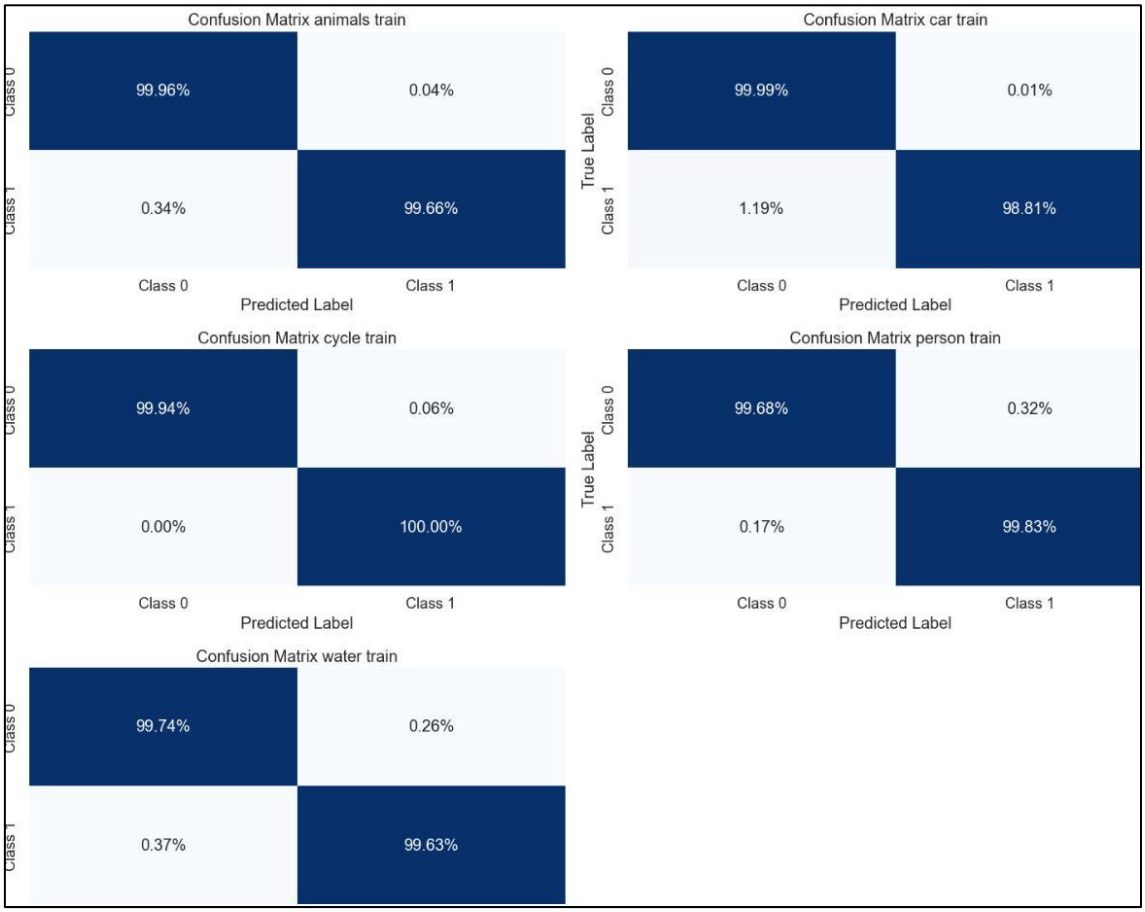
c. Gráficas de los resultados.



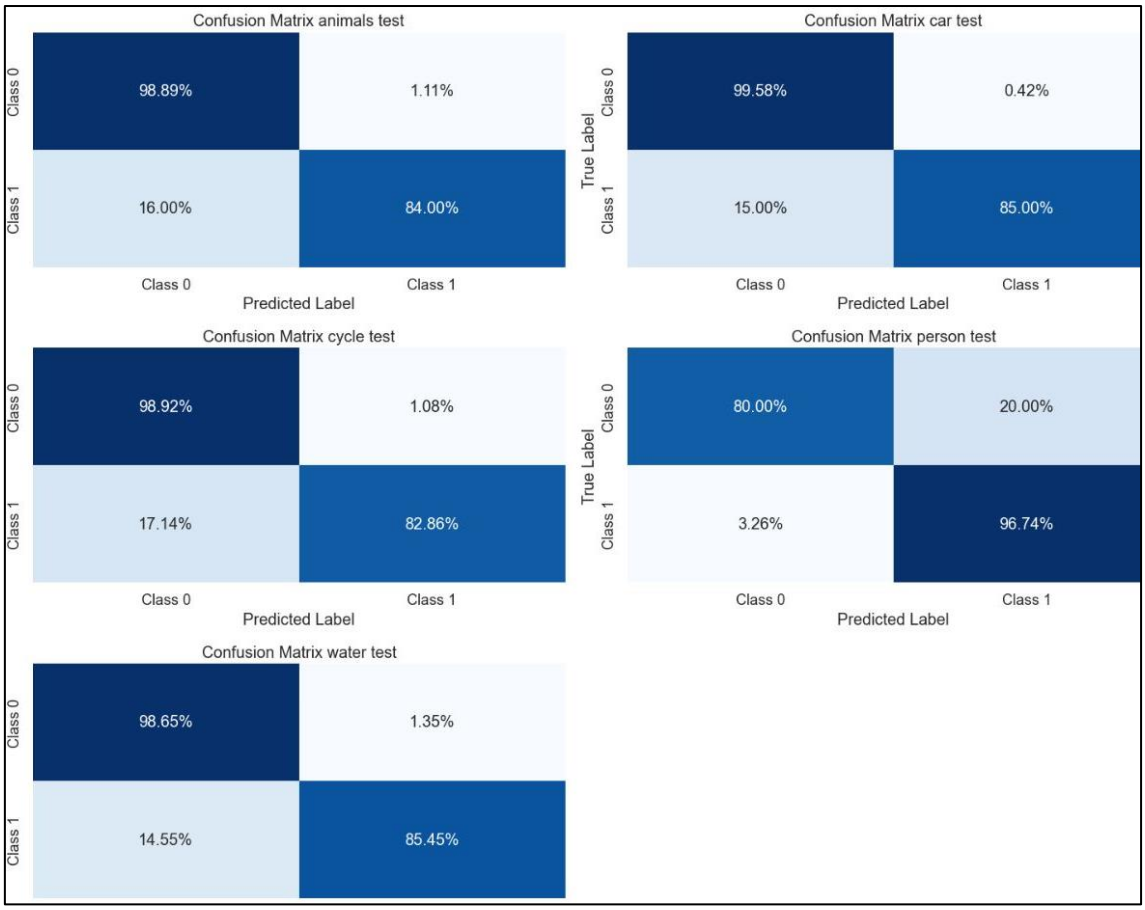
Observando la evolución del valor de la función de pérdida en la fase de entramiento se puede comprobar que el modelo efectivamente es capaz de minimizarla de manera consistente. Aunque con esto solo no es suficiente para evaluar el rendimiento de la red.



Es posible que sea más interesante observar la evolución de la métrica f1-score a lo largo del entrenamiento. Con esta gráfica se puede ver claramente la tendencia a mejorar a medida que avanza el entrenamiento y la convergencia de los valores de train y validación. No obstante, los valores finales obtenidos son bastante superiores a los de test, hay una clara sintomatología de overfitting. Esto se debe a que el modelo está sobre-aprendiendo la distribución de los datos. Cosa que se puede comprobar en la siguiente imagen:

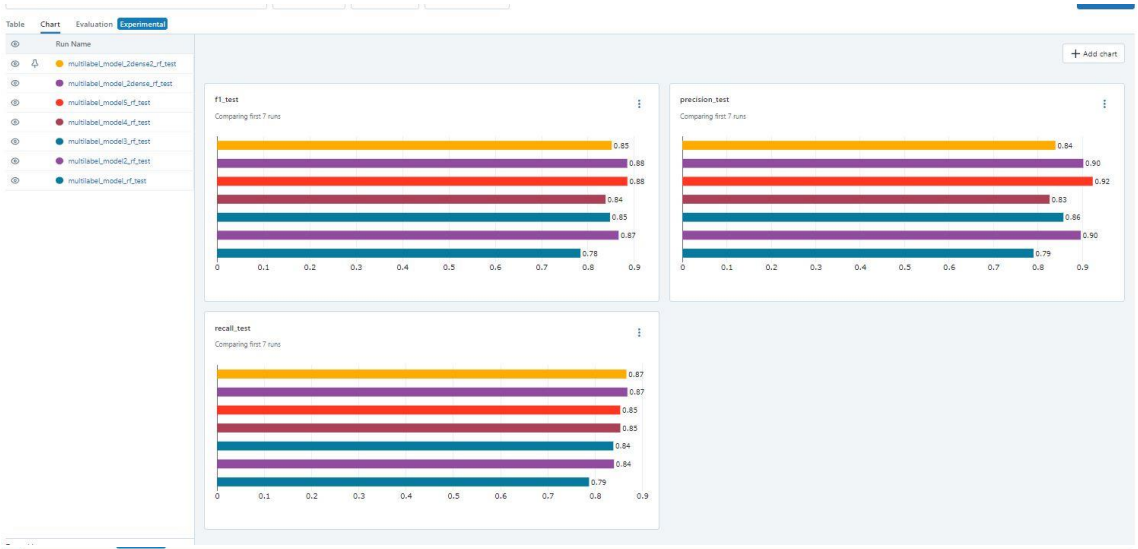


Analizando las matrices de confusión por cada clase se percibe un sesgo, aunque no es tan considerable como para tenerlo en cuenta, pese a que se discuta a continuación.



Y, sin embargo, se ha decido mantener los datos sintéticos porque con ellos se consigue mejorar el rendimiento de la red al evaluar el conjunto de test. Aún así, aun se perciben resquicios del sesgo mencionado en el apartado anterior, pues la red tiende a decir que no hay presencia de todas las etiquetas salvo la de la clase persona, que es la clase mayoritaria. Lo que reduce la puntuación de recall, dato que ya sabíamos que afectaba más a la f1-score que la precisión al ver la tabla del apartado de resultado de las métricas seleccionadas.

Podemos comprobar que entre todos los modelos propuestos, el mejor resultado se ha obtenido con este modelo. Esto viene especificado por las dos imágenes a continuación.



		Parameters														
	Run Name	Cre	first_big_dense_l	fourth_big_dens	image_size	kernel_size_fifth	kernel_size_first_	kernel_size_four	kernel_size_seco	kernel_size_thirc	neurons_dense_l	neurons_dense_l	neurons_dense_l	neurons_fifth_la	neurons_first_la	second_big_den	third_big_dens
	multilabel_model_2dense2_rf_test	(	3	3	(64, 64)	-	(3, 3)	(5, 5)	(5, 5)	(3, 3)	4096	256	5	-	32	2	3
	multilabel_model_2dense_rf_test	(	3	3	(64, 64)	-	(3, 3)	(5, 5)	(5, 5)	(3, 3)	4096	256	5	-	32	2	3
	multilabel_model5_rf_test	(	3	3	(224, 224)	-	(3, 3)	(5, 5)	(5, 5)	(3, 3)	224	-	-	-	32	2	3
	multilabel_model4_rf_test	(	3	3	(64, 64)	(7, 7)	(3, 3)	(5, 5)	(5, 5)	(3, 3)	64	-	-	512	32	2	3
	multilabel_model3_rf_test	(	3	3	(64, 64)	-	(3, 3)	(5, 5)	(5, 5)	(3, 3)	64	-	-	-	32	2	3
	multilabel_model2_rf_test	(	-	-	(64, 64)	-	(3, 3)	(5, 5)	(5, 5)	(3, 3)	64	-	-	-	32	-	-
	multilabel_model_rf_test	(	-	-	(64, 64)	-	(3, 3)	-	(5, 5)	(3, 3)	64	-	-	-	32	-	-