

## EXPERIMENTACIÓN CON CARTOON VS. REAL (CVR)

**0025-CvR-CMCMCMCMD-0001**

Imgs. train	Imgs. val.	Imgs. test		
1648 (824+824)	400 (200+200)	60 (30+30)		
Optimizador	F. pérdida	Tam. Imgs.	Batch_size	Epochs
RMSProp( $lr=1e-4$ )	Binary_crossentropy	256 x 256	20	100
<b>Estructura</b>				
C(32, (3,3)) MP C(64, (3,3)) MP C(128, (3,3)) MP C(128, (3,3)) MP F D(512) D(1) Convolucionales y Densas con ReLU, salida con Sigmoid				

Su objetivo es distinguir entre dibujos animados e imágenes realistas. Utiliza el dataset *small\_dataset*, compuesto por 2048 imágenes, 1648 para entrenamiento y 400 para validación. Para el conjunto de test se utilizan 60 imágenes que no pertenecen a las series/películas utilizadas en entrenamiento y validación.

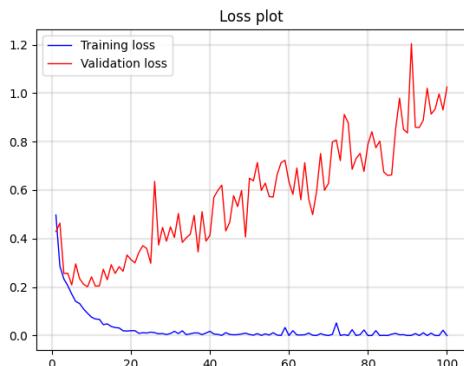


Fig. 1. 0024-CvR-CMCMCMCMD-0001-loss

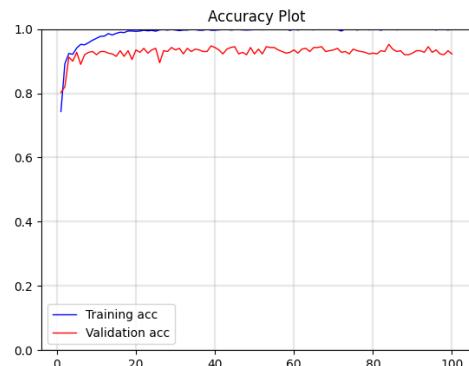
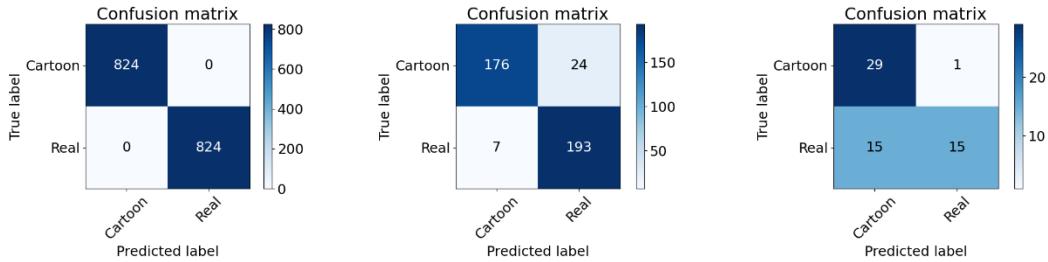


Fig. 2. 0024-CvR-CMCMCMCMD-0001-acc

La accuracy que se llega en el entrenamiento y en validación es muy alto, llegando al 100% y a más del 95% respectivamente. No obstante, el loss indica que puede haber pocas instancias y que puede estar ocurriendo un overfitting. Esto también pasaba con el conjunto de CvD, así que puede ser cosa de las imágenes. El loss de validación se separa demasiado rápido del entrenamiento, coincidiendo con el estancamiento de la curva en la acc.



En las matrices de confusión se puede ver que para el conjunto de test el acierto es no es malo del todo, un 73%. Se ve una tendencia a predecir imágenes de dibujos animados de forma exagerada, por eso acierta casi el 100% para cartoon y solo el 50% para las reales.

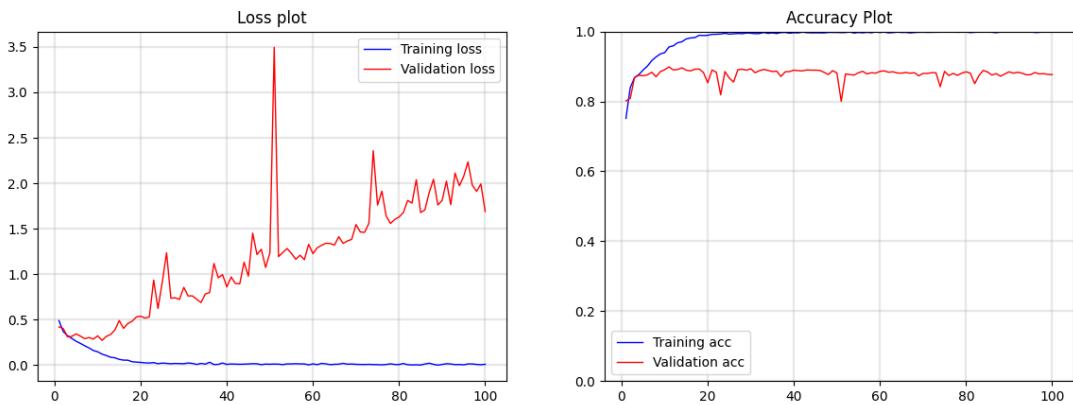
Parece que es factible distinguir entre dibujos animados e imágenes reales, además de tener un acierto en test que no es nada malo para una primera prueba donde no se ha hecho hincapié en la arquitectura del experimento.

## 0026-CvR-CMCMCMCMD-0002

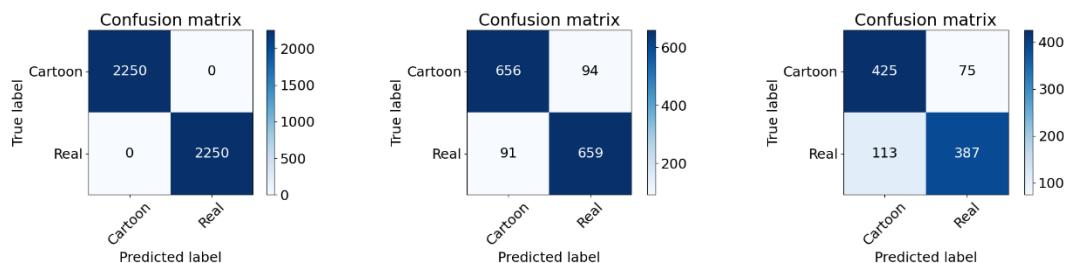
Imgs. train	Imgs. val.		Imgs. test	
4500 (2250+2250)	1500 (750+750)		1000 (500+500)	
Optimizador	F. pérdida	Tam. Imgs.	Batch_size	Epochs
RMSProp(lr=1e-4)	Binary_crossentropy	256 x 256	20	100
<b>Estructura</b>				
C(32, (3,3)) MP C(64, (3,3)) MP C(128, (3,3)) MP C(128, (3,3)) MP F D(512) D(1) Convolucionales y Densas con ReLU, salida con Sigmoid				

Este experimento trata de ver cómo se comporta la red con un conjunto mucho mayor, *medium\_dataset*, manteniendo el mismo principio de entrenamiento y validación con series diferentes al test. Además, hay que tener en cuenta que las clases están balanceadas en todos los conjuntos.

Las imágenes se han recogido de 15 series distintas, haciendo un total de 6000 (4500+1500). Para ello, se recogen 200 imágenes de cada serie, dividiendo estas 200 en cuatro subconjuntos de 50, correspondientes a cuatro capítulos distintos. Esto se hace para obtener una variedad de imágenes más rica, con escenas diferentes.



Parece que el loss tiene una curva menos pronunciada que en el experimento 0001, sin embargo, llega a valores más altos, con un rizado que en algunos casos tiene picos excesivos. El acierto se mantiene parecido, en torno al 88% para la validación y el acierto de entrenamiento es del 100%. El conjunto con el que se ha realizado es considerablemente grande, por lo que los resultados se pueden empezar a considerar como válidos.



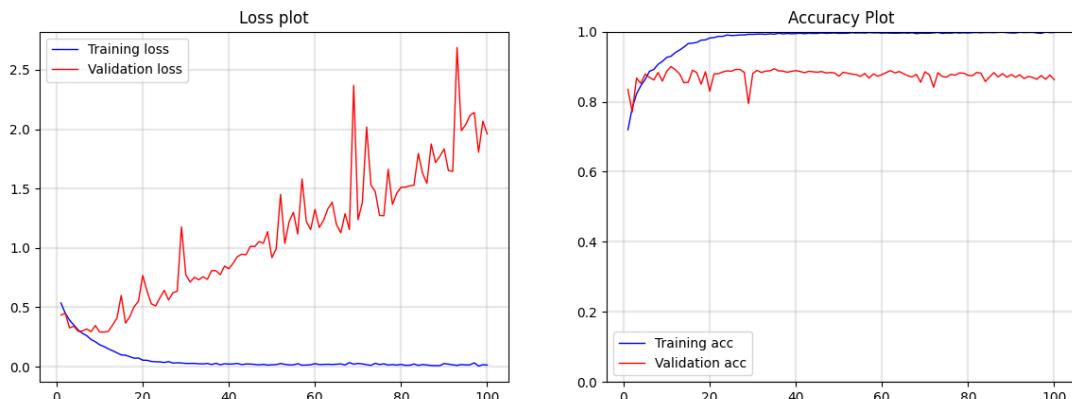
Al tener un conjunto de test de 1000 imágenes se pueden sacar conclusiones más sólidas y consistentes. Se obtiene un 81,2% de acierto en el conjunto de test, 85% en cartoon y 77,4% en las reales. Se tienden a predecir más de cartoon, por eso el acierto es superior para esta clase. La clave debería estar en conseguir un acierto más balanceado entre las dos clases, sin que haya una predominante. En el conjunto de validación no parece haber ningún sesgo entre las clases, aunque lo interesante siempre va a ser el conjunto de test porque tiene dibujo animados que no se han visto antes.

### 0027-CvR-CMCMCMCMD-0003

Imgs. train	Imgs. val.	Imgs. test
6500 (3250+3250)	1500 (750+750)	1000 (500+500)

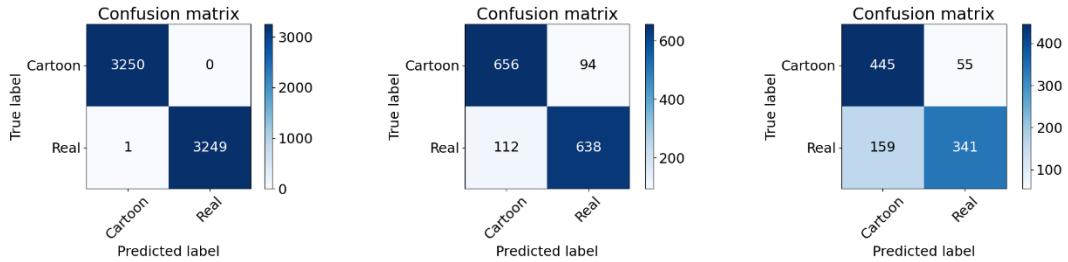
Optimizador	F. pérdida	Tam. Imgs.	Batch_size	Epochs
RMSProp( $lr=1e-4$ )	Binary_crossentropy	256 x 256	20	100
<b>Estructura</b>				
C(32, (3,3)) MP C(64, (3,3)) MP C(128, (3,3)) MP C(128, (3,3)) MP F D(512) D(1) Convolucionales y Densas con ReLU, salida con Sigmoid				

La CartoonGAN se ha usado para generar más imágenes de dibujos animados, partiendo de imágenes realistas del conjunto de entrenamiento. Las personas generadas tienden a ser muy rojizas, puede que afecte algo al rendimiento. El conjunto de validación y de test se mantienen igual que en el anterior experimento. El aporte que da la GAN es que no se probar que no se necesitan más imágenes de dibujos animados para mejorar el rendimiento. Sin embargo, se han tenido que añadir más imágenes reales para que estuvieran las clases balanceadas, por lo que sí se necesita una recogida de más imágenes realistas.



El loss es muy parecido al anterior experimento, mientras que en el acierto se detecta algo de overfitting, ya que la curva tiene pendiente negativa. El hecho de que el acierto no baje significativamente quiere decir que no afecta negativamente añadir imágenes que se han generado a partir de las reales de validación. Es posible que, si estas imágenes se generasen desde otro conjunto que no fuera el de entrenamiento, podría enriquecerse más la diferencia entre real y dibujo.

También ha de tenerse en cuenta que la diferencia entre una imagen animada y real puede estar en los colores más planos, donde una variedad más pequeña en cuanto a la diferencia de colores. Las formas varían mucho entre los dibujos animados, sobre todo para las personas.

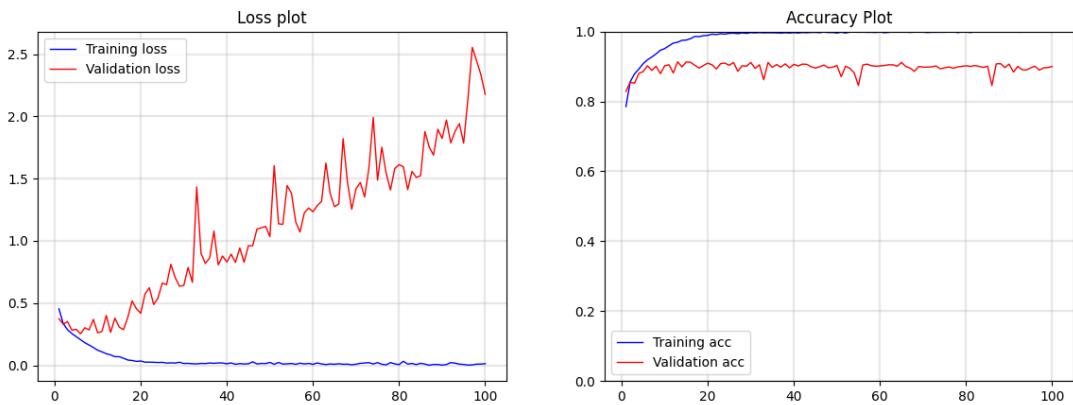


En validación se supera el 86% de acierto, por lo que es similar a lo mostrado anteriormente, puede que algo inferior. En test hay un acierto del 78,6% que, si bien no es poco, es inferior al de antes. De hecho, hay una diferencia muy importante entre el cartoon y real, 89% y 68,2% de acierto respectivamente. Puede cotejarse con la matriz del anterior epoch, donde se muestra una diferente notablemente inferior, 84,8% y 74,2%. Así que puede ser puntual este bajón que ha habido al final. Sí es verdad que se predicen muchos más cartoon, de ahí su gran acierto.

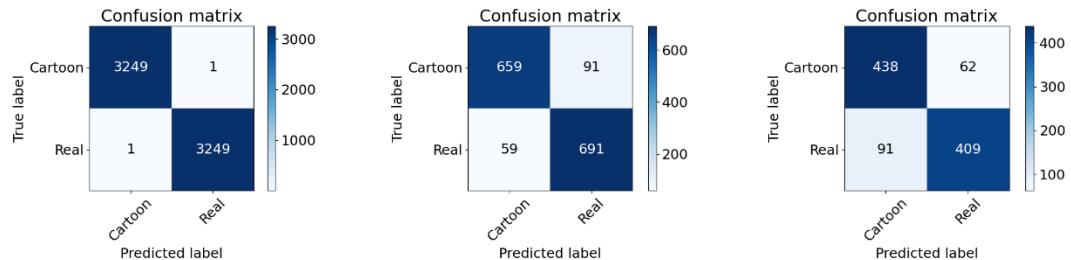
#### 0028-CvR-CMCMCMCMD-0004

Imgs. train	Imgs. val.		Imgs. test	
6500 (3250+3250)	1500 (750+750)		1000 (500+500)	
Optimizador	F. pérdida	Tam. Imgs.	Batch_size	Epochs
RMSProp(lr=1e-4)	Binary_crossentropy	256 x 256	20	100
<b>Estructura</b>				
C(32, (3,3)) MP C(64, (3,3)) MP C(128, (3,3)) MP C(128, (3,3)) MP F D(512) D(1) Convolucionales y Densas con ReLU, salida con Sigmoid				

Se vuelve a usar CartoonGAN para general 1000 imágenes cartoon, esta vez a partir de otras imágenes de dibujos animados. El resultados de aplicar los filtros de la GAN hace que las imágenes cambien los colores radicalmente y puedan parecer distintas. Aunque las formas de los objetos se van a mantener, puede que se modifiquen ligeramente pero no se debería apreciar.



La gráfica correspondiente del loss es prácticamente igual a los dos experimentos anteriores. No obstante, el acierto sí que cambia, ahora no tiene pendiente positiva y parece que simplemente se estanca en valores en torno al 90% para validación. En algún punto ha llegado casi al 92%. El rizado no es muy grande, aunque tiene algún pico sobresaliente.



Tanto la de entrenamiento como la de validación son muy similares a los dos experimentos anteriores, aunque la de validación tiene un acierto algo superior. En cuanto a la matriz de test, el acierto es del 84,7%, siendo la mejor cifra alcanzada hasta ahora y superando a las anteriores con creces. Para cartoon 87,6% de acierto y para las reales 81,8%, por lo que la diferencia comienza a hacerse más pequeña y a balancearse entre ambas clases.

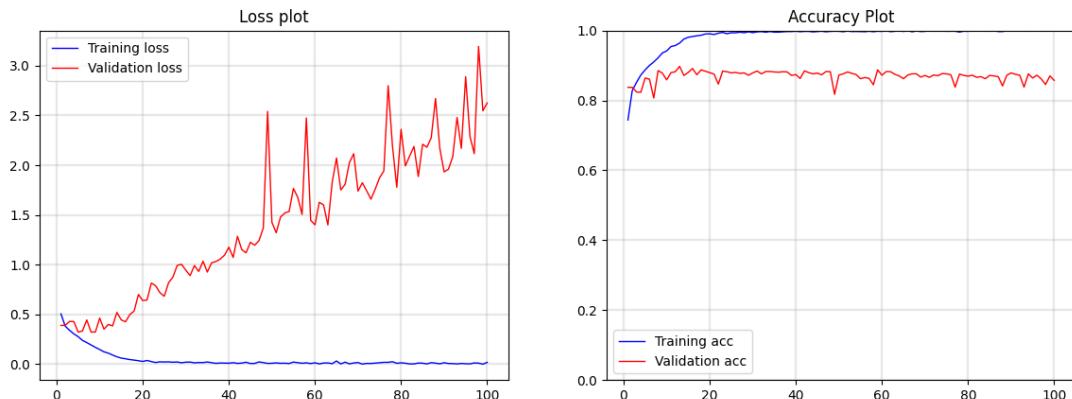
Parece entonces que la GAN puede ser más útil para generar más dibujos aplicando los filtros a los propios dibujos que a imágenes reales. Incluso puede haber imágenes reales a las que no interese aplicar la GAN.

## 0029-CvR-CMCMCMCMD-0005

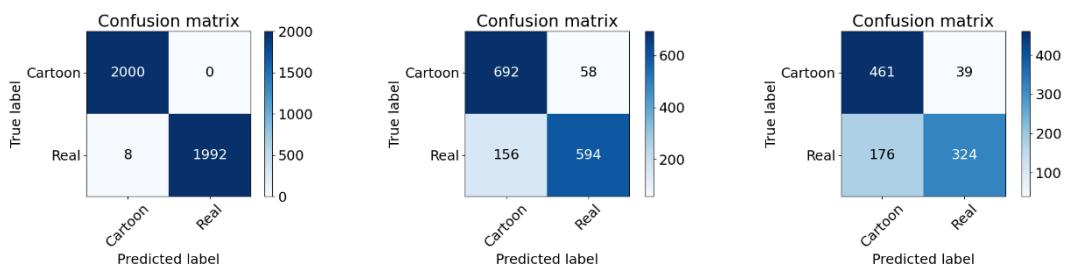
Imgs. train	Imgs. val.	Imgs. test
4000 (2000+2000)	1500 (750+750)	1000 (500+500)

Optimizador	F. pérdida	Tam. Imgs.	Batch_size	Epochs
RMSProp(lr=1e-4)	Binary_crossentropy	256 x 256	20	100
<b>Estructura</b>				
C(32, (3,3)) MP C(64, (3,3)) MP C(128, (3,3)) MP C(128, (3,3)) MP F D(512) D(1) Convolucionales y Densas con ReLU, salida con Sigmoid				

Esta prueba tiene un total de 4000 imágenes para entrenar, todas correspondientes a imágenes de dibujos animados originales. (sin usar la GAN), del conjunto *pruebas\_GAN/cartoon*. El objetivo es comparar con los dos siguientes experimentos, así se verá la diferencia entre todo dibujos reales y mezcla entre reales y la GAN provenientes de imágenes realistas y otros dibujos.



El loss es similar a los anteriores experimentos, no parece que vaya a diverger más tarde. Diverge demasiado pronto la validación del entrenamiento, pero con un conjunto más grande como en el anterior experimento, esto tampoco parece que se mitigue. Puede que sea por cómo son las imágenes, al igual que ocurría en CvD. El acierto se sitúa por encima del 85% para la validación, aunque lo que interesa es comparar con las otras dos pruebas.



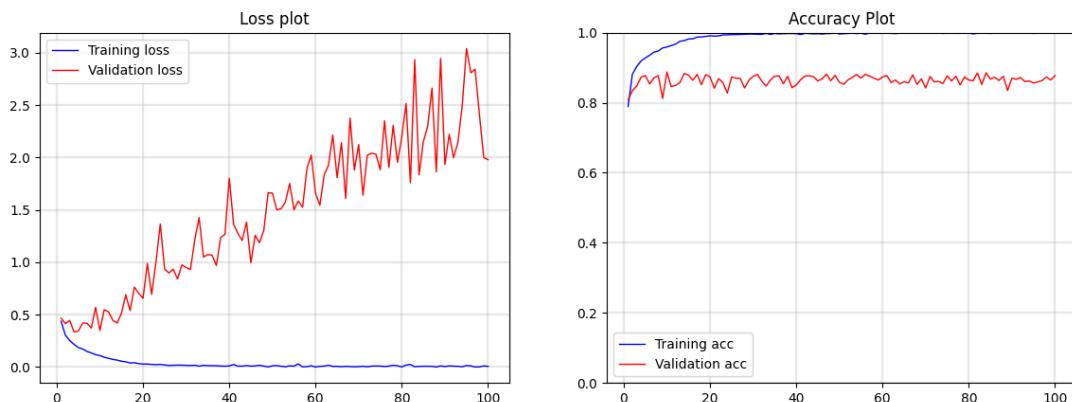
Vuelve a verse cierto sesgo hacia cartoon, ya que se predicen muchas más de esta clase, tanto en la validación como en el test. El acierto en test es del 78.5%, habiendo una diferencia entre cartoon y real muy grande, 92,2% y 64,8% respectivamente. En pruebas

anteriores parecía que no había tanta diferencia entre las clases, por lo que en estas pruebas se podrá determinar si la GAN favorece a disminuir esa diferencia. Pese a ello, la media no es baja, aunque es muy importante igualar el acierto entre ambas clases.

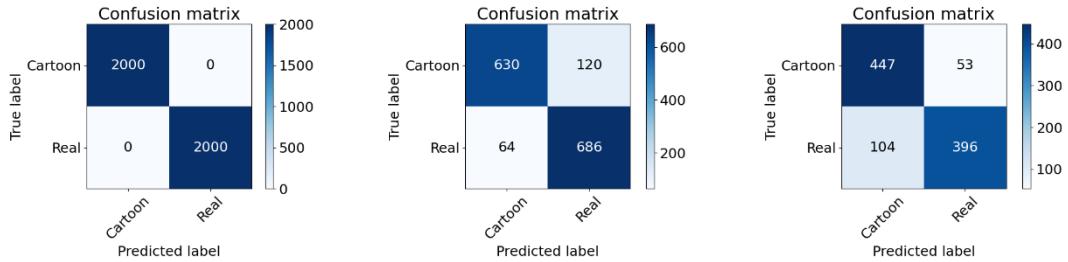
### 0030-CvR-CMCMCMCMD-0006

Imgs. train	Imgs. val.	Imgs. test		
4000 (2000+2000)	1500 (750+750)	1000 (500+500)		
Optimizador	F. pérdida	Tam. Imgs.	Batch_size	Epochs
RMSProp( $lr=1e-4$ )	Binary_crossentropy	256 x 256	20	100
Estructura				
C(32, (3,3)) MP C(64, (3,3)) MP C(128, (3,3)) MP C(128, (3,3)) MP F D(512) D(1) Convolucionales y Densas con ReLU, salida con Sigmoid				

Se tratan de imágenes generadas a partir de imágenes de dibujos animados. Hay 1000 generadas con la GAN y 1000 originales en el conjunto para la clase cartoon, del conjunto *pruebas\_GAN/cartoon\_from\_cartoon*.



El loss es muy parecido al experimento anterior, manteniendo ese rizado con picos que parece ser característica de este conjunto de datos. El acierto parece estable, con un rizado aceptable. Se finaliza el entrenamiento con más del 87% de acierto, por lo que es ligeramente superior a lo que se había obtenido. En general está parejo con el experimento anterior según se aprecia en la gráfica de la acc.

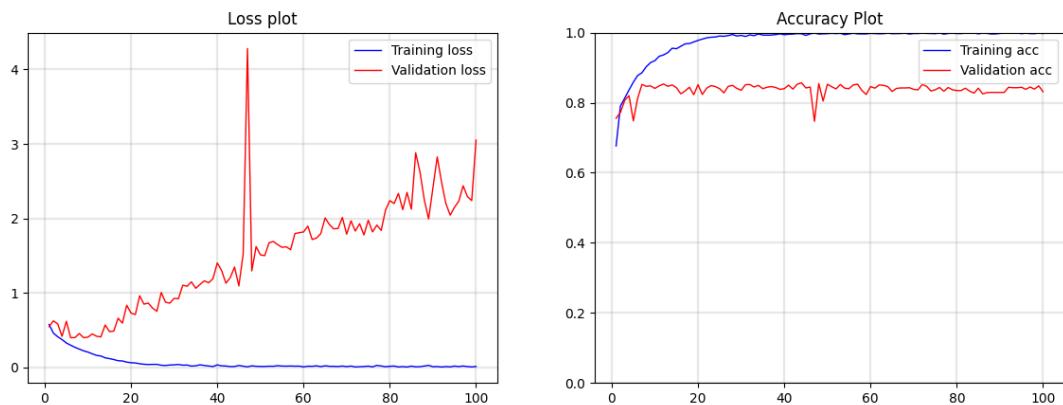


En la matriz de confusión de test se puede ver que el acierto se sitúa en el 84,3%, lo cual es muy parecido a lo que se había visto en la prueba 0004. En test, para cartoon se obtiene 89,4% y para real 79,2%. Se sigue observando ese sesgo hacia las imágenes de dibujos. Sorprende que tenga más acierto este conjunto con respecto a la anterior prueba que eran todo imágenes original de dibujos. Aunque el experimento 0004 tiene ahora mismo el mejor resultado de todos los probado, el cual sigue el mismo formato de imágenes generadas a partir de dibujos animados.

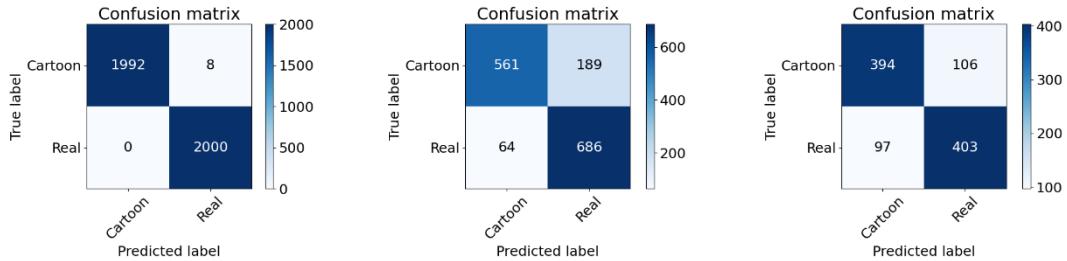
### 0031-CvR-CMCMCMCMD-0007

Imgs. train	Imgs. val.		Imgs. test	
4000 (2000+2000)	1500 (750+750)		1000 (500+500)	
Optimizador	F. pérdida	Tam. Imgs.	Batch_size	Epochs
RMSProp( $lr=1e-4$ )	Binary_crossentropy	256 x 256	20	100
<b>Estructura</b>				
C(32, (3,3)) MP C(64, (3,3)) MP C(128, (3,3)) MP C(128, (3,3)) MP F D(512) D(1) Convolucionales y Densas con ReLU, salida con Sigmoid				

Se tratan de 1000 imágenes que han sido generadas a partir de imágenes realistas con la GAN, del conjunto *pruebas\_GAN/cartoon\_from\_real*.



Se aprecia un pico muy grande en el loss, aunque el rizado en general puede ser algo inferior a los experimentos previos. El acierto ha bajado bastante, se sitúa en valores por debajo del 85% para el conjunto de validación. Se nota que no en los experimentos anteriores era más alto.



En la matriz de confusión se acierta un 83,13% de las instancias, aunque el epoch 100 es más bajo que el resto, en las demás ronda el 84%. Es curioso que en este caso de predicen muchas más imágenes reales que cartoon, por ello también se aciertan más. En la matriz de test también se puede ver un comportamiento similar, obteniendo un 79,7% de acierto, 78,8% para cartoon y 80,6% para reales. El acierto de test en el conjunto con solo imágenes originales y este es muy parecido, sin embargo, el de las imágenes generadas a partir de otros dibujos es bastante superior, sobre un 5% de diferencia.

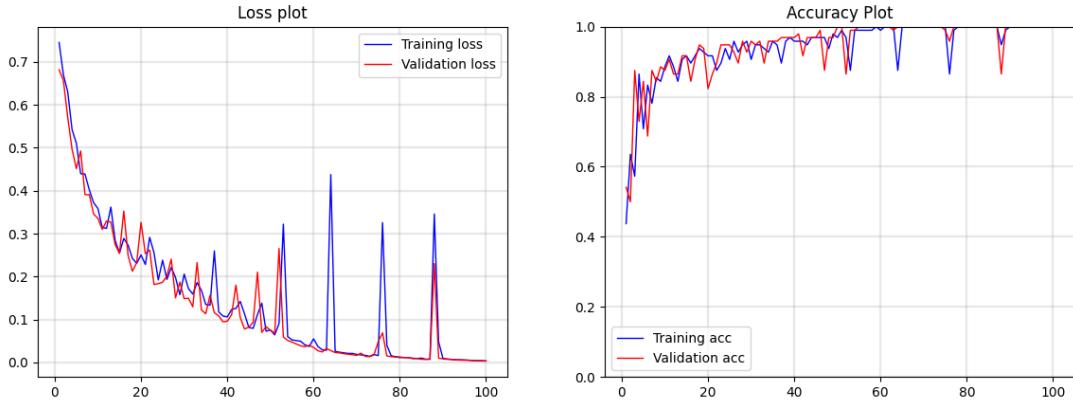
Como conclusión de estos tres últimos experimentos se puede afirmar que crear imágenes de dibujos a partir de otros dibujos aumenta el rendimiento. Esto se puede deber a que las formas cambian ligeramente y la gama cromática también, dependiendo del filtro usado.

## 0032-CvR-CMCMCMCMD-0008

Imgs. train	Imgs. val.		Imgs. test	
96 (48+48)	96 (48+48)		96 (48+48)	
Optimizador	F. pérdida	Tam. Imgs.	Batch_size	Epochs
RMSProp( $lr=1e-4$ )	Binary_crossentropy	256 x 256	16	100
<b>Estructura</b>				
C(32, (3,3)) MP C(64, (3,3)) MP C(128, (3,3)) MP C(128, (3,3)) MP F D(512) D(1) Convolucionales y Densas con ReLU, salida con Sigmoid				

Debido a los resultados de pasar imágenes realistas a dibujos con CartoonGAN, puede que lo que afecte negativamente al rendimiento son las imágenes con personas. Por ello, se utiliza el conjunto *small* de *landscapes*, que incluye solo imágenes de paisajes. El objetivo de este experimento es tener 48 imágenes reales, pasarlal a cartoon con la GAN y ver cuánto tarda en distinguir entre real y cartoon (las mismas pero generadas con

GAN). En el siguiente experimentos se realizará con solo personas, para ver las diferencias. El conjunto de validación y test se está usando el mismo que para entrenar, porque lo que se busca es ver cómo entrena la red.



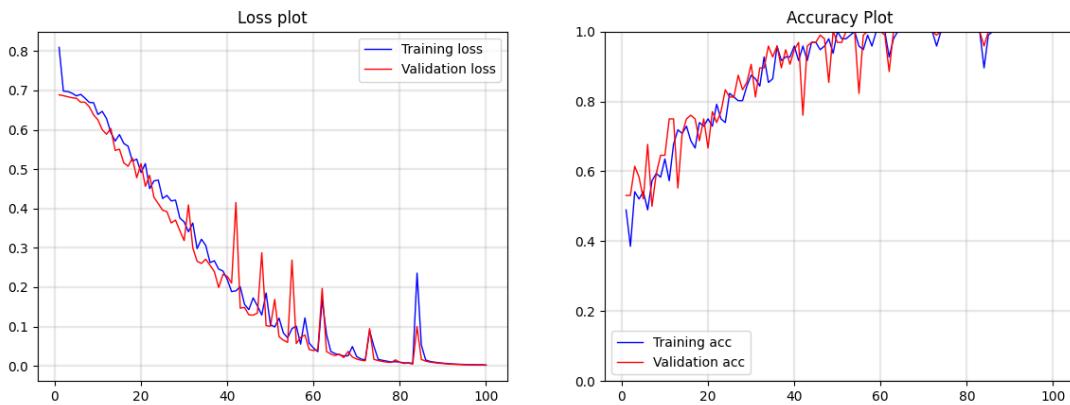
En la gráfica del loss se puede ver que esta vez si desciende hasta 0, pero es un efecto visual, ya que validación es el mismo conjunto que entrenamiento. Por tanto, tendrán los mismos resultados. En el acierto pasa lo mismo, se detectan algunos picos, pero se puede decir que hacia el epoch 60 ya se ha alcanzado el 100% de acierto. Además, se llega al 90% relativamente rápido, subiendo después más lentamente hasta el 100%.

Las matrices de confusión no se van a estudiar porque son las tres iguales y terminan con un 100% de acierto. Lo importante de este experimentos es ver la curva hasta llegar al acierto de todas.

### **0033-CvR-CMCMCMCMD-0009**

Imgs. train	Imgs. val.		Imgs. test	
96 (48+48)	96 (48+48)		96 (48+48)	
Optimizador	F. pérdida	Tam. Imgs.	Batch_size	Epochs
RMSProp(lr=1e-4)	Binary_crossentropy	256 x 256	16	100
<b>Estructura</b>				
C(32, (3,3)) MP C(64, (3,3)) MP C(128, (3,3)) MP C(128, (3,3)) MP F D(512) D(1) Convolucionales y Densas con ReLU, salida con Sigmoid				

Se usa del conjunto *small* el de *faces*, que contiene personas y caras. El procedimiento es igual que en el anterior experimento, se cogen 48 imágenes reales, se pasan a cartoon con la GAN y se intenta diferenciar entre las 96 obtenidas (cartoon vs. real).



El loss es parecido al anterior experimento, tiene una bajada debido a ser el entrenamiento, con un ritmo considerable. El acierto presenta una curva totalmente distinta a la anterior, aunque tarda poco más de 60 epochs para llegar al 100%. El caso es que se puede ver como en un comienzo le cuesta mucho más distinguir las personas reales de las generadas como cartoon, empezando con valores inferiores al 50% (clasificador aleatorio).

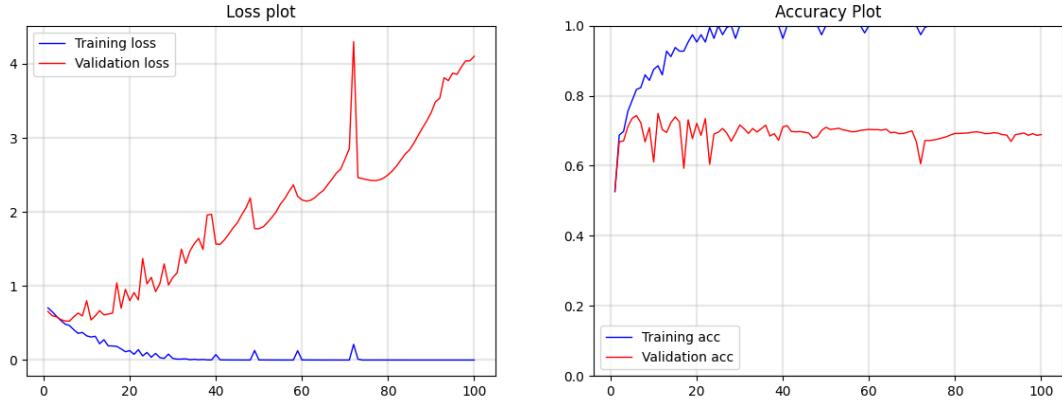
Esto puede ser indicio de que las imágenes de CartoonGAN generadas con imágenes reales de personas puede perjudicar al rendimiento. En el experimento 0003 hay una cantidad muy grande de personas que se han hecho cartoon, por lo que es una posible explicación a su peor rendimiento. Para tener unos resultados más consistentes, se plantearán tres pruebas más comprobando los resultados de las caras y los paisajes.

### 0034-CvR-CMCMCMCMD-0010

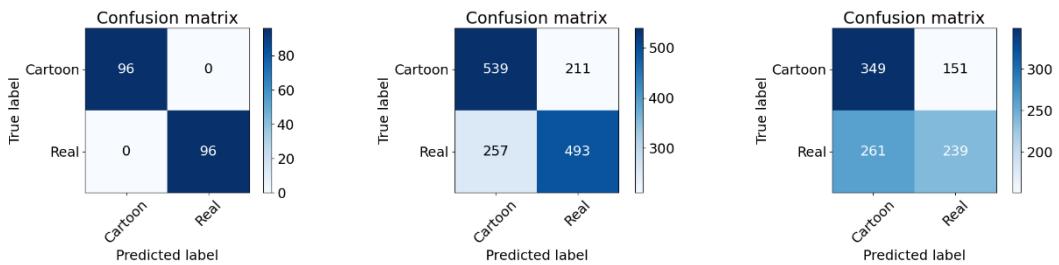
Imgs. train	Imgs. val.		Imgs. test	
192 (96+96)	1500 (750+750)		1000 (500+500)	
Optimizador	F. pérdida	Tam. Imgs.	Batch_size	Epochs
RMSProp( $lr=1e-4$ )	Binary_crossentropy	256 x 256	16	100
Estructura				
C(32, (3,3)) MP C(64, (3,3)) MP C(128, (3,3)) MP C(128, (3,3)) MP F D(512) D(1) Convolucionales y Densas con ReLU, salida con Sigmoid				

El conjunto de entrenamiento contiene 96 imágenes cartoon y 96 reales, del conjunto *big/faces*. De las de cartoon, la mitad son generadas por la GAN a partir de caras y la otra mitad son dibujos animados originales. El objetivo es ver el rendimiento teniendo el 50% generado con caras y el 50% con originales, comparándolo con otros dos experimentos

que tendrán paisajes uno y otro con todo dibujos originales. Los conjunto de validación y test son los grandes, con 1500 y 1000 imágenes respectivamente.



La curva del loss es bastante pronunciada, llegado a valores más altos que en experimentos anteriores, esto puede deberse a la cantidad pequeña de imágenes o a usar imágenes de personas generadas. Se obtendrá una conclusión durante los siguientes experimentos. El acierto para validación parece que va adquiriendo una ligera pendiente negativa, muy similar a lo ocurrido en el experimento 0003. En este caso, el conjunto de validación tiene imágenes de series que pueden no estar incluidas en las 192 imágenes, ya que en validación hay 1500 de 15 series distintas.



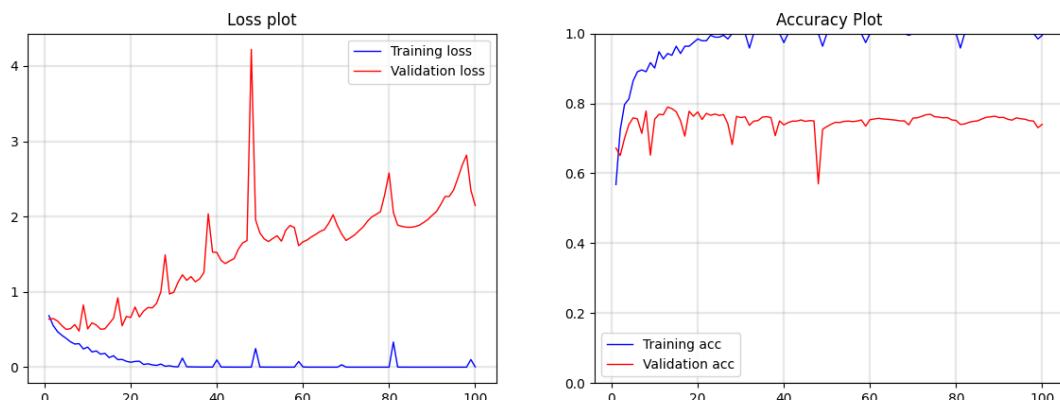
En la matriz de confusión de validación se finaliza con un 68,8% de acierto con un 71,87% para cartoon y un 65,73% para las reales. En la matriz de test se puede ver un comportamiento bastante peor, teniendo un 58,8% de acierto total, 69,8% cartoon y 47,8% para las reales. Durante todos los experimentos se ha venido viendo que el problema reside en diferenciar imágenes reales, donde se predicen muchísimas más cartoon. Para validación está más balanceada la predicción, pero para test hay un 61% de predicciones de cartoon, volviendo a tener un mayor acierto expresamente por esto.

Parece que las caras no son útiles para la diferenciación de la red, aunque como se ha dicho antes, la cantidad de imágenes también influye, por lo que se juzgará ene experimentos posteriores.

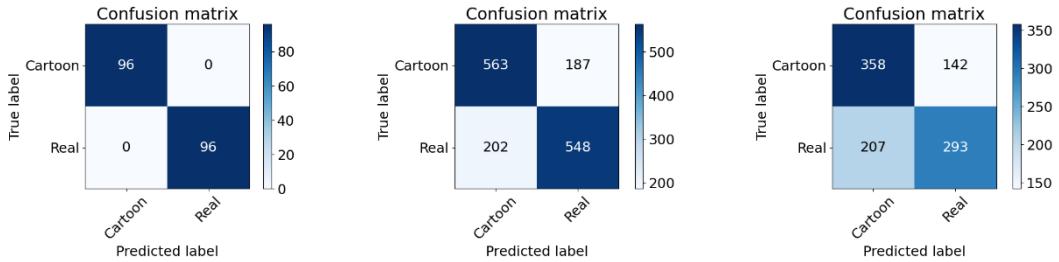
### 0035-CvR-CMCMCMCMD-0011

Imgs. train	Imgs. val.	Imgs. test		
192 (96+96)	1500 (750+750)	1000 (500+500)		
Optimizador	F. pérdida	Tam. Imgs.	Batch_size	Epochs
RMSProp( $lr=1e-4$ )	Binary_crossentropy	256 x 256	16	100
Estructura				
C(32, (3,3)) MP C(64, (3,3)) MP C(128, (3,3)) MP C(128, (3,3)) MP F D(512) D(1) Convolucionales y Densas con ReLU, salida con Sigmoid				

Se sigue la misma configuración del experimento anterior, en vez de caras esta vez la mitad de las imágenes son de paisajes, usando el conjunto *big/landscapes*.



Visualmente ya se puede ver que el loss tiene una curva menos pronunciada que el anterior experimento que contenía personas. Es cierto que tiene un rizado pronunciado con un pico exagerado. En cuanto al cierto, se nota que es mucho más alto que en la ocasión anterior, con valor cercanos al 75% Puede que haya una ligera pendiente negativa, pero no es tan claro como en la ocasión anterior. Partiendo que se tiene el mismo número de imágenes, parece que lo que estaba afectando antes eran las personas generadas.



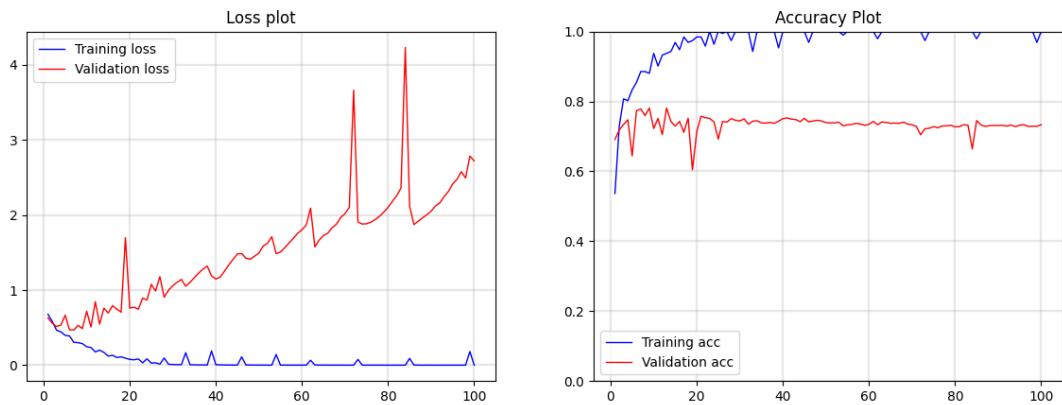
En la matriz de validación se puede ver que el acierto es superior al experimento anterior, 73,13% frente al 68,8% del anterior. Pese a ello, hay una diferencia grande entre el acierto de cartoon, 81,87% y el de reales, 64,4%, siendo una diferencia excesivamente grande. Esto se vuelve a confirmar con la matriz para test con un acierto para cartoon del 71,16% y real del 58,6%, haciendo una media del 65,1% acertadas. Aunque la diferencia sea grande entre ambas clases sea muy grande, esto sí se puede deber al número de imágenes disponibles, ya que en conjuntos más grandes también pasa.

Se puede concluir que tener imágenes de caras afecta de forma negativa al rendimiento. Ahora queda comparar cómo influye usar imágenes de paisajes, que son mejores que las de personas, pero falta ver cuánta diferencia hay con un conjunto compuesto solo de imágenes originales, que se podrá ver en el siguiente experimento.

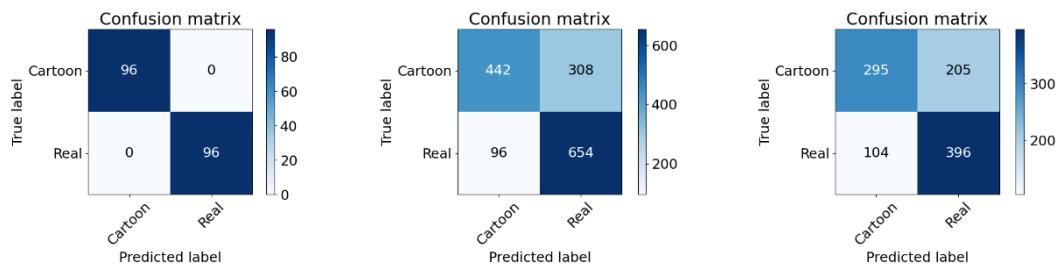
## 0036-CvR-CMCMCMCMD-0012

Imgs. train	Imgs. val.		Imgs. test	
192 (96+96)	1500 (750+750)		1000 (500+500)	
Optimizador	F. pérdida	Tam. Imgs.	Batch_size	Epochs
RMSProp(lr=1e-4)	Binary_crossentropy	256 x 256	16	100
Estructura				
C(32, (3,3)) MP C(64, (3,3)) MP C(128, (3,3)) MP C(128, (3,3)) MP F D(512) D(1) Convolucionales y Densas con ReLU, salida con Sigmoid				

Se usan 96 imágenes cartoon y 96 reales, las de cartoon son todas originales, por lo que esta vez no se ha generado ninguna con la GAN.



La grafica del loss es similar a la del experimento 0011 que contenía paisajes, teniendo picos altos, por lo que con todo imágenes originales también se da. Es curioso que el acierto parece ligeramente inferior que, en el experimento anterior, aunque la curva parece más estable en este.



En validación se tiene un 73,07% de acierto, 58,93% para cartoon y 87,2% para las reales. En todos los experimentos anteriores esta diferencia sucedía al revés, siempre tenía menos las reales. Se ha mirado el log para ver las matrices de confusión en los epochs anteriores, parece que en el 98 la media era similar mientras que era mejor en cartoon que en real. Esto quiere decir que el cambio ha ocurrido entre el epoch 98 y 99, siendo algo drástico, aunque la media no bajase. En la matriz de test se vuelve a ver este comportamiento donde el sesgo es favorable para las imágenes reales, obteniendo un 69,1% de acierto en total. Es superior al anterior experimento, sin embargo, no tiene un rendimiento muy superior a las imágenes generadas.

En los tres últimos experimentos se ha visto como utilizar personas generadas empeora los resultados frente a los paisajes. Los paisajes también están por debajo del conjunto que tiene todo dibujos originales. Sin embargo, puede que con un mayor número de imágenes esta diferencia se acortase. También hay que tener en cuenta que los paisajes

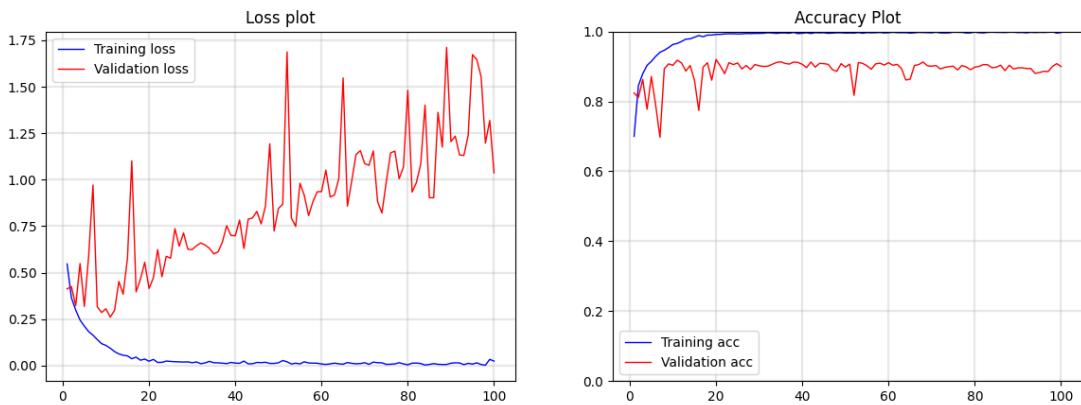
usado son exteriores, montes, playas, desiertos, etc. Puede que usar paisajes más urbanizados tuviesen un rendimiento algo superior. En general los dibujos animados no tienen paisajes con una amplitud tan grande, por ejemplo, puede haber secuencias que tengan bosques, pero bosques vistos desde dentro, no imágenes desde fuera como si fueran tomadas por un helicóptero.

De todas formas, esto puede ser una explicación aproximada de por qué usar imágenes realistas generadas tiene un rendimiento algo peor que usar imágenes generadas a partir de otros dibujos animados. Además, las series realistas tienen muchísimas imágenes de personas. Los dibujos animados representan las formas de las personas de manera distinta, el resto de elementos puede ser más parecido. Por tanto, esto también es un factor que debería influir, las imágenes reales no tienen variedad en cuanto a la forma de las personas.

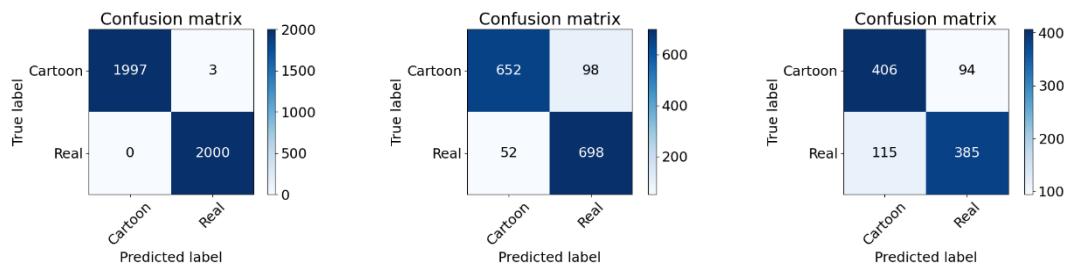
### **0037-CvR-CMCMCMCMD-0013**

Imgs. train	Imgs. val.	Imgs. test		
4000 (2000+2000)	1500 (750+750)		1000 (500+500)	
Optimizador	F. pérdida	Tam. Imgs.	Batch_size	Epochs
RMSProp(lr=1e-4)	Binary_crossentropy	256 x 256	20	100
<b>Estructura</b>				
C(32, (3,3)) MP C(64, (3,3)) MP C(128, (3,3)) MP C(128, (3,3)) MP F D(512) D(1) Convolucionales y Densas con ReLU, salida con Sigmoid				

Este experimento en conjunto con los siguientes tiene como objetivo comprobar si la GAN presenta alguna ventaja con respecto a métodos tradicionales de filtros, usando el conjunto *filter\_vs\_gan/big\_dataset/filter* y convirtiendo las mismas imágenes. En esta prueba se usan 2000 imágenes de cartoon, 1000 originales y 1000 generadas con filtros tradicionales. De estas 1000, se usan dos tipos de filtros, color quantization y filtro bilateral, quedándose en 500 y 500 respectivamente. Se incluye un tratamiento sobre los bordes de las figuras, un coloreado tipo pastel con algo de borroso para que sean colores más planos y no haya tantos distintos.



Pese a que la curva del loss sigue siendo pronunciada, ha bajado el valor máximo con respecto a los experimentos anteriores. El acierto se sitúa en niveles muy buenos, sobre pasando el 91% en alguna ocasión. El rizado sigue una forma similar a los experimentos que se han venido haciendo anteriormente.



Los resultados que se muestran son interesantes debido a que son bastante buenos, en todos los conjuntos. Para validación se tiene un 90% de acierto, mientras que para test disminuye hasta un 79.1%. Cabe destacar que el acierto entre cartoon y real no tiene una diferencia muy grande, lo cual es bueno. Se tiene un 81.2% para cartoon y un 77% para las reales.

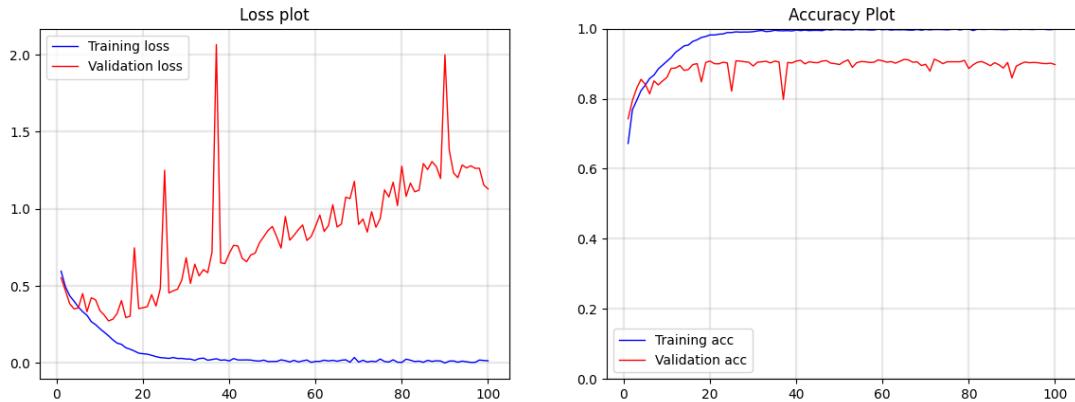
En el siguiente experimento se hará el mismo proceso, pero generando imágenes con CartoonGAN. A primera vista los resultados con los filtros tradicionales no son nada malos, teniendo una tasa de acierto muy buena para el número de imágenes que tienen.

#### 0038-CvR-CMCMCMCMD-0014

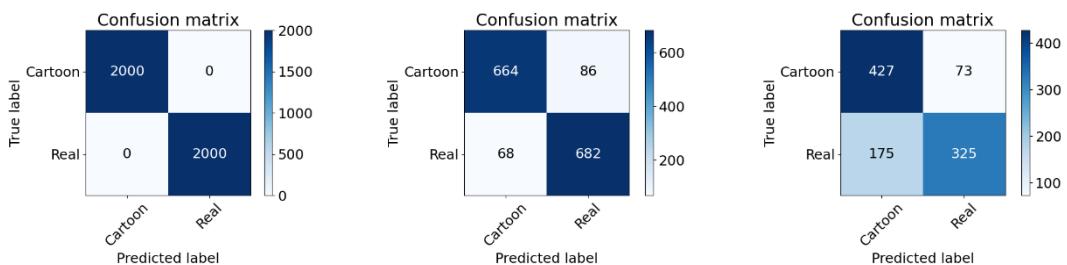
Imgs. train	Imgs. val.		Imgs. test	
4000 (2000+2000)	1500 (750+750)		1000 (500+500)	
Optimizador	F. pérdida	Tam. Imgs.	Batch_size	Epochs
RMSProp(lr=1e-4)	Binary_crossentropy	256 x 256	20	100

Estructura
C(32, (3,3)) MP C(64, (3,3)) MP C(128, (3,3)) MP C(128, (3,3)) MP F D(512) D(1) Convolucionales y Densas con ReLU, salida con Sigmoid

Esta prueba es similar a la anterior, solo que las 1000 imágenes de cartoon generadas se realizan con CartoonGAN. Se ha usado el conjunto *filter\_vs\_gan/big\_dataset/GAN* para este experimento.



En la anterior prueba parecía que los métodos tradicionales de aplicar filtros hacían que el loss bajase, sin embargo, en esta prueba también ha ocurrido. Esto indica que se debe a las imágenes que se han seleccionado. El acierto es alto, pero visualmente parece inferior que la anterior prueba, la 0014. En concreto se finaliza con un 89.73% de acierto, por lo que no se distancia apenas del experimento anterior, que tenía un 90%.



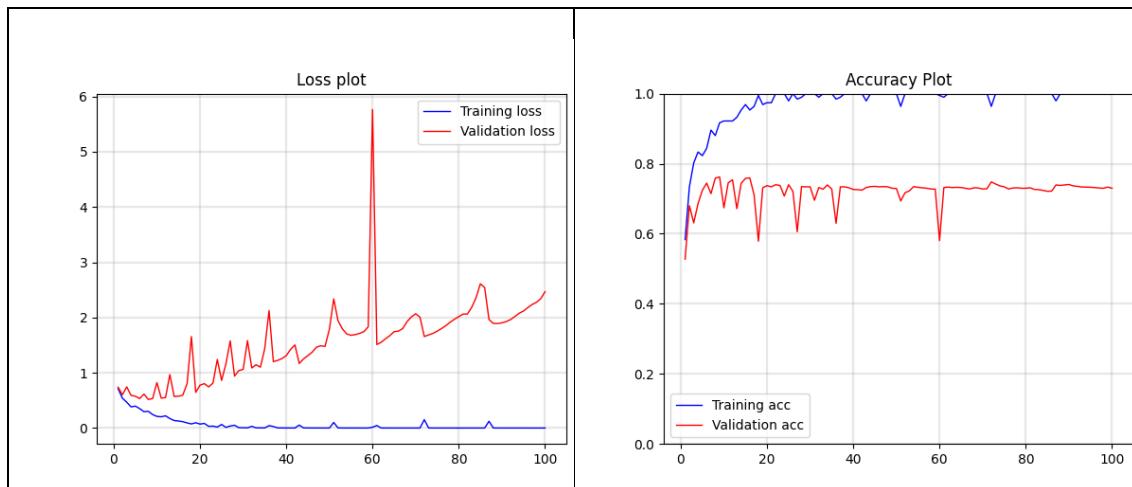
Lo primero que sorprende es el bajo rendimiento que ha habido en el acierto para las imágenes reales. En total se tiene un 75,2%, con un 85,4% para cartoon y 65% para reales. Después de todos los experimentos realizados, este valor es algo atípico y podría ser situacional, por lo que sería necesario lanzar de nuevo la prueba para comprobar si esto es cierto. Se confirma que no es situacional, se puede ver en el experimento 0014-B. No obstante, en esa prueba relanzada se acierta muchísimo más las reales que cartoon.

Teniendo en cuenta los resultados de los filtros tradicionales, parece que en este caso pueden sacar mejores resultados. Ya se sabía de antemano que las imágenes generadas con las GAN de imágenes realistas empeoraban el rendimiento, por lo que sería aventurado dar una conclusión ahora mismo. Se van a hacer algunas pruebas más con filtros tradicionales.

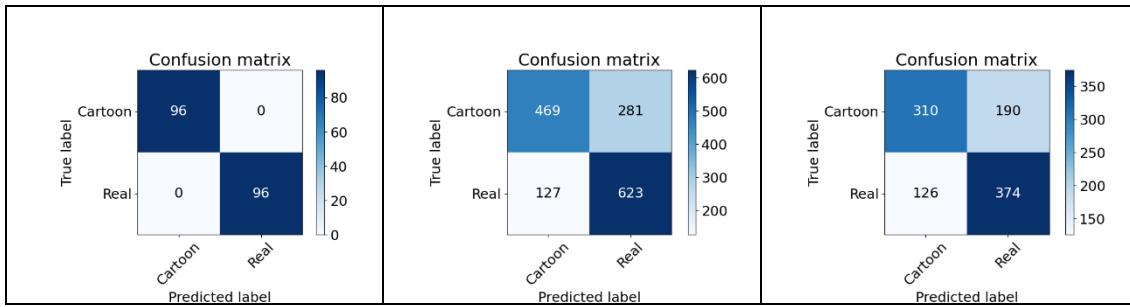
### 0039-CvR-CMCMCMCMD-0015

Imgs. train	Imgs. val.		Imgs. test	
192 (96+96)	1500 (750+750)		1000 (500+500)	
Optimizador	F. pérdida	Tam. Imgs.	Batch_size	Epochs
RMSProp( $lr=1e-4$ )	Binary_crossentropy	256 x 256	20	100
<b>Estructura</b>				
C(32, (3,3)) MP C(64, (3,3)) MP C(128, (3,3)) MP C(128, (3,3)) MP F D(512) D(1) Convolucionales y Densas con ReLU, salida con Sigmoid				

Al igual que en las pruebas 0010 y 0011, se va a probar el rendimiento sobre imágenes reales convertidas con filtros tradicionales, dividiendo en caras y paisajes. Para este primer experimento se utilizará el conjunto que contiene paisajes convertidas con la GAN, *filter\_vs\_GAN/faces\_vs\_landscapes/big/filter/landscapes*. De las 96 imágenes de cartoon, 48 son generadas utilizando los dos mismo filtros que se han explicado con anterioridad.



En este caso parece que la curva del loss es menos pronunciada, pero se puede deber a la representación de la gráfica, donde hay un pico tan grande que puede que al resto de la curva la baje. La tasa de acierto está por cerca del 75%, aunque más picos hacia abajo que de costumbre. De todas formas, la gráfica es similar a la vista en el experimento 0011, aunque el acierto en este caso parece inferior al de la prueba mencionada



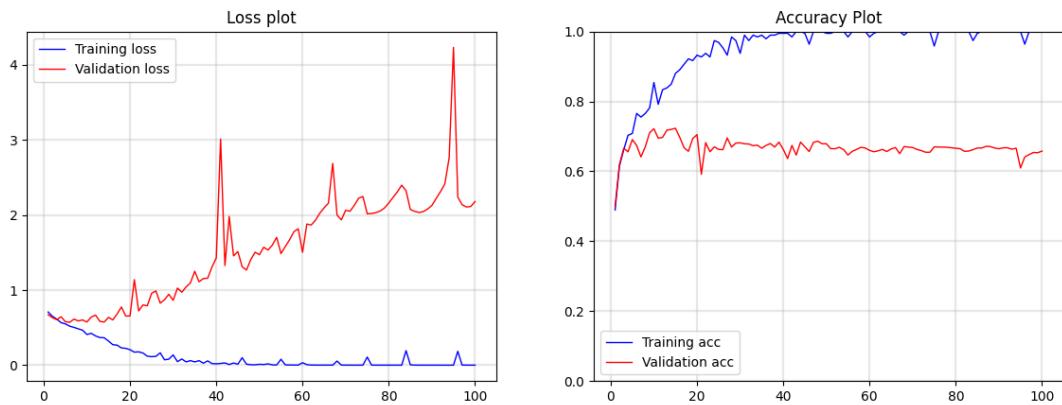
El entrenamiento finaliza con un 72,98% de acierto, como se puede ver en la matriz de validación. Tanto en la matriz de validación como en la de test se puede observar que se predicen bastante más imágenes reales que cartoon. La tasa de acierto de test se sitúa en un 68,4%, con un 62% para cartoon y un 74,8% para las reales. Si se compara de nuevo con el experimento 0011, el cual tenía un 73,13% de acierto, es muy superior a la presente prueba.

Se comienza a ver que las pruebas realizadas previamente, en concreto la 0013, no eran del todo verdaderas. Como se había dicho, las imágenes generadas por la GAN partiendo de imágenes realistas empeoraban considerablemente los resultados, mientras que los filtro tradicionales los empeoraban menos. Aquí se puede ver que en los paisajes obtiene un acierto bastante inferior, pero habrá que comprobarlo con las caras también.

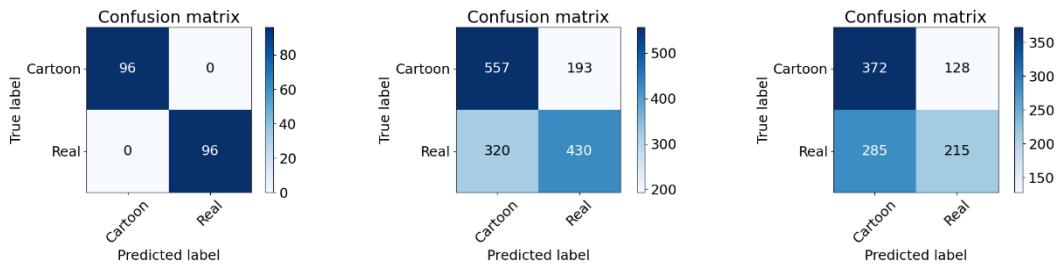
## 0040-CvR-CMCMCMCMD-0016

Imgs. train	Imgs. val.		Imgs. test	
192 (96+96)	1500 (750+750)		1000 (500+500)	
Optimizador	F. pérdida	Tam. Imgs.	Batch_size	Epochs
RMSProp(lr=1e-4)	Binary_crossentropy	256 x 256	20	100
<b>Estructura</b>				
C(32, (3,3)) MP C(64, (3,3)) MP C(128, (3,3)) MP C(128, (3,3)) MP F D(512) D(1) Convolucionales y Densas con ReLU, salida con Sigmoid				

Se sigue la misma estructura de la anterior prueba, 96 imágenes cartoon de las cuales 48 son generadas usando filtros tradicionales. Estas imágenes se corresponden a personas reales una vez se le aplican los dos filtros tradicionales, usando el conjunto *filter\_vs\_GAN/faces\_vs\_landscapes/big/filter/faces*.



El loss es parecido al experimento 0010, puede que algo más bajo pese a tener un último pico tan alto. También parece que el acierto es ligeramente inferior que la prueba mencionada, se sitúa en valores en torno al 65%, mientras que la 0010 estaba cerca del 70%.



Ahora se vuelven a predecir más cartoon que reales, como ha venido ocurriendo en la mayoría de los experimentos vistos, ocurriendo tanto en validación como en test. El acierto en test es de un 58,7%, con un 74,4% para cartoon y un 43% para reales, siendo esto una diferencia enorme. Un clasificador aleatorio acertaría más imágenes reales que este, ya que en un clase binaria sería del 50%. La prueba 0010 tiene el mismo resultado en test que esta, por lo que no hay nada destacable.

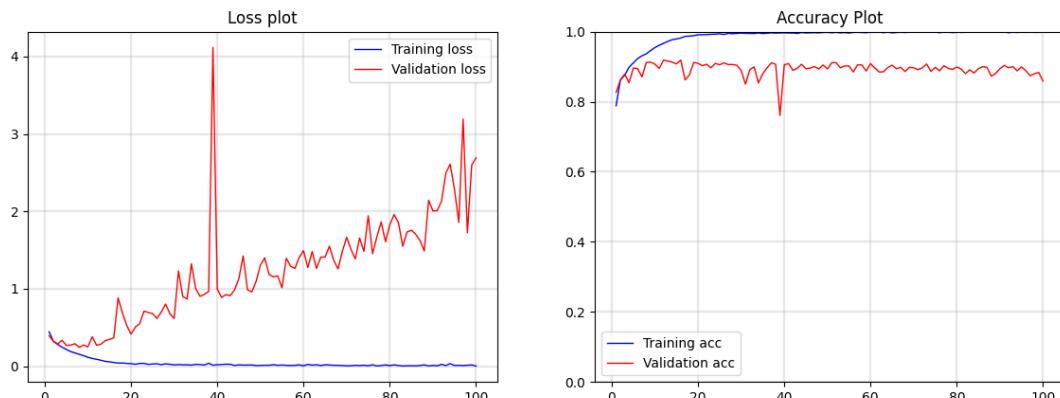
En la prueba anterior se había demostrado que tenía peor rendimiento en paisajes y ahora tiene el mismo rendimiento en personas. ¿Por qué entonces el experimento 0013 era mejor que el 0014? Puede deberse a muchos factores, puede que las imágenes seleccionadas hayan favorecido a la GAN, o incluso que las imágenes de esos experimentos favorecieran a los filtros tradicionales. Lo que sí se sabía era que utilizar imágenes reales bajaba el rendimiento en la GAN. También hay que comentar que se han hecho pruebas dividiendo entre paisajes y personas, pero hay muchas más situaciones que no se han

contemplado esta división, lo cual no implica que en las pruebas 0013 y 0014 no tengan imágenes de más tipos. Se propondrá un experimento final para determinar el resultado sobre filtros tradicionales y GAN.

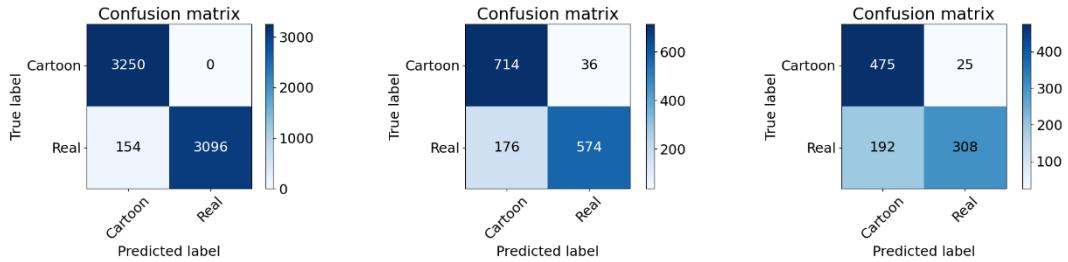
### 0041-CvR-CMCMCMCMD-0017

Imgs. train	Imgs. val.		Imgs. test	
6500 (3250+3250)	1500 (750+750)		1000 (500+500)	
Optimizador	F. pérdida	Tam. Imgs.	Batch_size	Epochs
RMSProp( $lr=1e-4$ )	Binary_crossentropy	256 x 256	20	100
<b>Estructura</b>				
C(32, (3,3)) MP C(64, (3,3)) MP C(128, (3,3)) MP C(128, (3,3)) MP F D(512) D(1) Convolucionales y Densas con ReLU, salida con Sigmoid				

La prueba usa una configuración muy parecida al experimento 0004, generando 1000 imágenes adicionales al conjunto cartoon. Esta vez se utilizan los dos filtros tradicionales que se han venido explicando, usando el conjunto *medium\_filter\_dataset*. Con este experimento se podrá determinar si los resultados de la GAN son mejores o no.



El loss parece que es superior a la prueba 0004, donde había un rizado grande, pero con picos muy contenidos. Además, en este experimento se llega a 3 de loss mientras que antes se quedaba antes del 2.5. En cuanto al acierto, parece que al final del entrenamiento decrece el acierto para la validación, estando en valores muy cercanos al 90%. Para dar una conclusión más precisa se ha de esperar a examinar el conjunto de test, donde se ofrece un conjunto no visto.



En las matrices de confusión se puede ver que se predicen muchos más cartoon. Incluso en la matriz de entrenamiento se fallan demasiadas instancias reales que han sido cartoon. En experimentos anteriores con un conjunto de más de 6000 imágenes, se fallaban menos de cinco instancias. Pasando a la de test, se aciertan un 78,3% de las imágenes, aunque se ha comentado que parecía que había overfitting. Revisando el log, parece que la cifra más alta que obtiene es en el epoch 99 con un 82,4%, siendo una cifra muy alta pese a que es inferior a la prueba 0004, que era del 84,7%. De forma general, los últimos 10 episodios de la prueba 0004 son mejores que esta prueba, por lo que no mejora a la GAN.

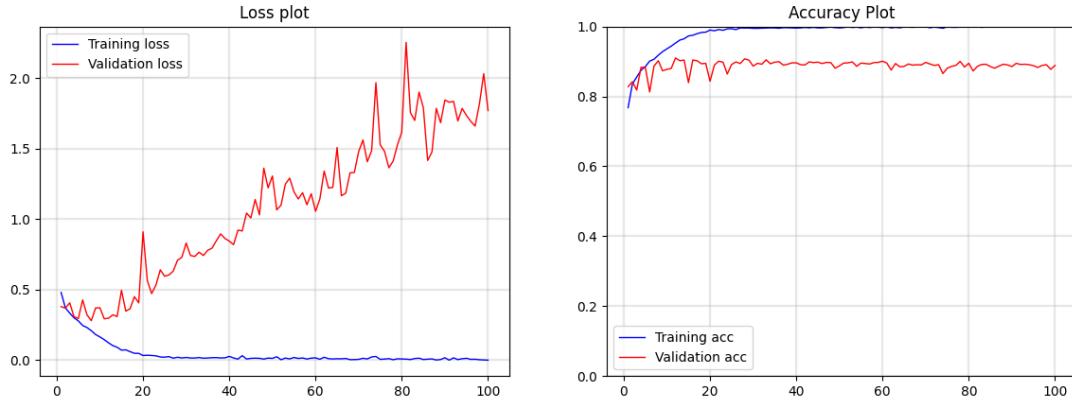
Se ha tratado de probar bajo las mismas condiciones, sacando los mejores resultados para ambos métodos, GAN y filtros tradicionales, donde todo parece indicar que la GAN tiene cierta ventaja frente a los filtros. Pese a que en una prueba los filtros tradicionales parecían mejores, la GAN ha ganado en el resto y sigue teniendo la tasa de acierto más alta en el conjunto de test. Le lleva más de un 2,3% de acierto al mejor resultado de los filtros tradicionales, por lo que se puede concluir que la mejor prueba hasta ahora sigue siendo la que usa CartoonGAN generando dibujos a partir de otros dibujos.

Revisando las imágenes de ambos métodos, la GAN tiene la capacidad de crear una gama de colores muy distinta, alterando ligeramente las formas. A vista de una persona, la GAN genera imágenes que debido a los colores podrían parecer una escena totalmente distinta, lo cual puede estar afectando positivamente al modelo.

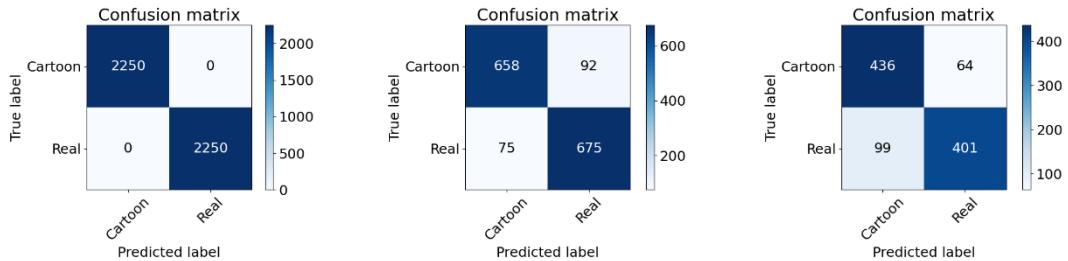
## 0042-CvR-CMCMCMCMD-0018

Imgs. train	Imgs. val.		Imgs. test	
4500 (2250+2250)	1500 (750+750)		1000 (500+500)	
Optimizador	F. pérdida	Tam. Imgs.	Batch_size	Epochs
RMSProp( $lr=1e-4$ )	Binary_crossentropy	256 x 256	20	100
<b>Estructura</b>				
C(32, (3,3)) MP C(64, (3,3)) MP C(128, (3,3)) MP C(128, (3,3)) MP F D(512) D(1) Convolucionales y Densas con ReLU, salida con Sigmoid				

Esta prueba utilizará el método de interpolación Lanczos en vez de el más cercano, como se había usado anteriormente, comparándose así directamente con las prueba 0002. Por tanto, se pretende ver si hay alguna diferencia notable entre los dos métodos de interpolación, usando el conjunto *médium\_dataset/interpolation\_lanczos*.



El loss es parecido al del experimento 0002, tiene algo menos de rizado y sobre todo no tiene picos exageradamente grandes. El acierto parece bastante estable a partir del epoch 30, teniendo un rizado contenido. Puede que sea una tasa de acierto superior a la vista en la prueba 0002, se podrá ver mejor con las matrices de confusión.



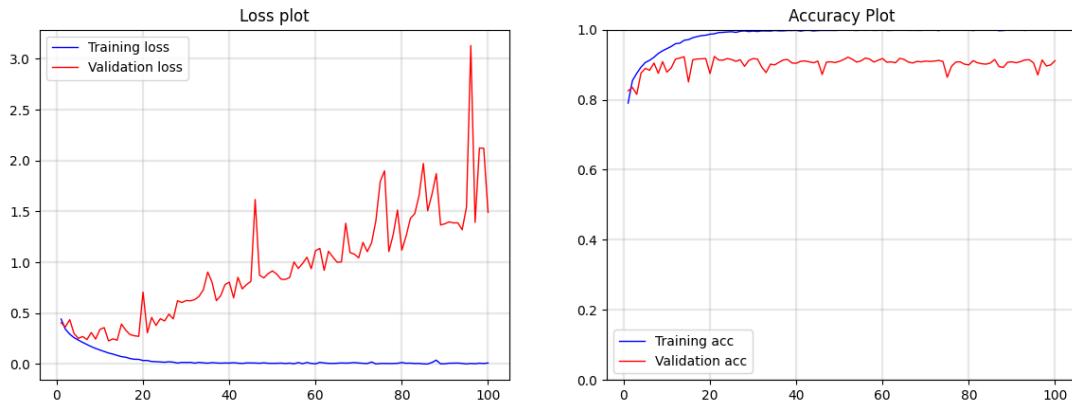
En la matriz de entrenamiento se ve cómo se aciertan todas las instancias. En la de validación se finaliza con un 88,87% de acierto, mientras que para la de test se tiene 83,7%. Este último resultado, se puede comprobar en el log, es el mejor de los diez últimos episodios. Es una tasa de acierto muy buena, que supera ligeramente a la prueba 0002. Para estar más seguro sobre si Lanczos tiene un impacto sobre el acierto del modelo, se realizará otro experimento con el conjunto más grande.

## 0043-CvR-CMCMCMCMD-0019

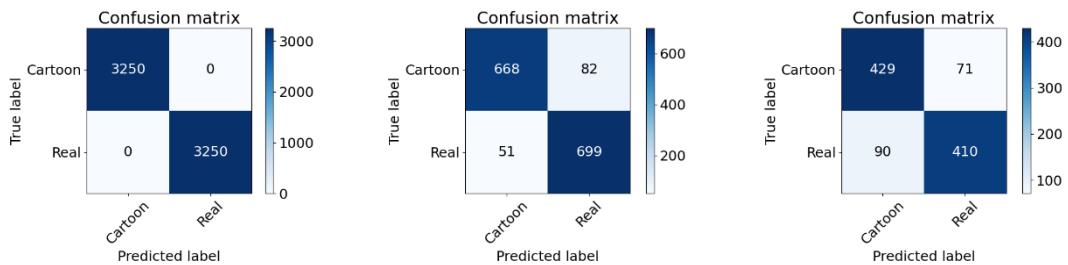
Imgs. train	Imgs. val.	Imgs. test
6500 (3250+3250)	1500 (750+750)	1000 (500+500)

Optimizador	F. pérdida	Tam. Imgs.	Batch_size	Epochs
RMSProp( $lr=1e-4$ )	Binary_crossentropy	256 x 256	20	100
<b>Estructura</b>				
C(32, (3,3)) MP C(64, (3,3)) MP C(128, (3,3)) MP C(128, (3,3)) MP F D(512) D(1) Convolucionales y Densas con ReLU, salida con Sigmoid				

La siguiente prueba consta de 6500 imágenes para entrenamiento, 1000 de ellas generadas con CartoonGAN a partir de otros dibujos animados. El método de interpolación para hacer las imágenes más pequeñas es Lanczos, usando el conjunto *médium\_cartoonGAN\_dataset/interpolation\_lanczos*. El objetivo es comparar el rendimiento con las prueba 0004, que es la mejor hasta ahora, para corroborar el experimento 0018, en el que se obtiene un buen acierto con un conjunto menor.



Un loss parecido a lo visto en la anterior prueba, con algún pico más grande. El acierto de validación parece haber subido ligeramente, aunque tiene algún pico inferior que antes no parecía haber. Con respecto a las prueba 0004, no parece que haya una diferencia significativa, siendo ambas gráficas muy parecidas.



Se termina con un 91,13% de acierto sobre el conjunto de validación, lo cual es un buen número, aunque se hubiese alcanzado en anteriores pruebas. Lo mismo ocurre con la matriz de confusión, que arroja un 83,9% de acierto, siendo un número muy bueno aunque

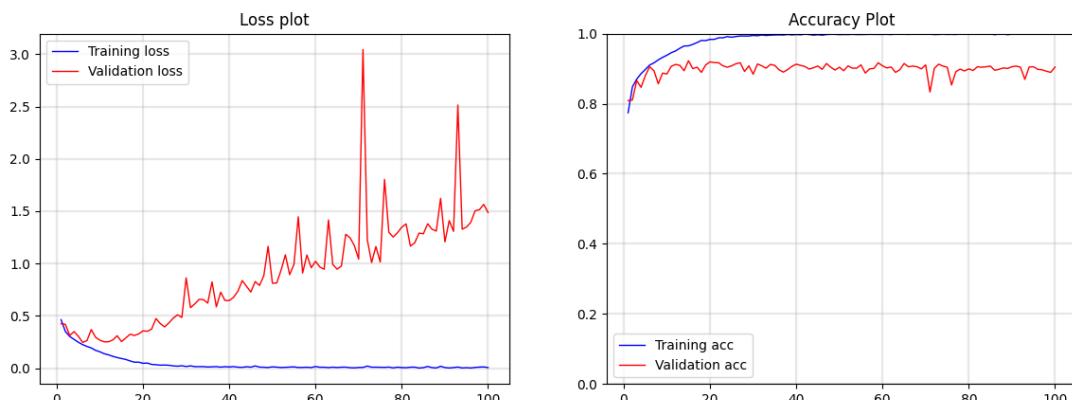
es algo inferior que el de la prueba 0004, que era del 84,7%. Cabe mencionar que las clases también están bien balanceadas en cuanto a acierto, donde no se encuentra una diferencia muy grande entre ellas.

Se puede concluir que los resultados entre los dos métodos (vecino más cercano y Lanczos) no tienen alguna diferencia perceptible que cause algún impacto a la hora de acertar instancias. Esto puede ser debido a que no afecta mucho elegir la interpolación cuando se trata de reducir la imagen. Al final estos métodos suelen ser más interesante cuando se tiene que escalar la imagen para hacerla más grande ya que afecta a la nitidez y a la forma de los objetos.

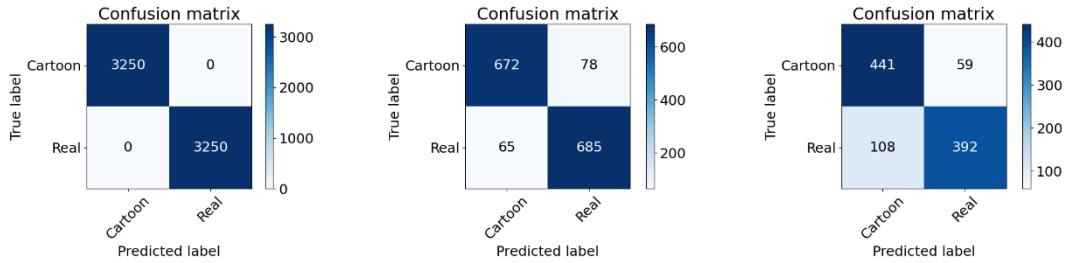
#### **0044-CvR-CMCMCMCMD-0020**

Imgs. train	Imgs. val.	Imgs. test		
6500 (3250+3250)	1500 (750+750)	1000 (500+500)		
Optimizador	F. pérdida	Tam. Imgs.	Batch_size	Epochs
RMSProp( $lr=1e-4$ )	Binary_crossentropy	200 x 200	20	100
<b>Estructura</b>				
C(32, (3,3)) MP C(64, (3,3)) MP C(128, (3,3)) MP C(128, (3,3)) MP F D(512) D(1) Convolucionales y Densas con ReLU, salida con Sigmoid				

Se van a realizar una serie de experimentos cuyo propósito es determinar si el tamaño de las imágenes repercute de forma significativa sobre la tasa de acierto. Para ello, en este experimento se reducirán las imágenes a 200x200px, frente a los 256x256px que tenían el resto de experimentos, usando el conjunto *images\_200px*. Cabe recordar que es el mejor conjunto hasta ahora, por tanto contiene imágenes generadas por la GAN a partir de dibujos animados.



De las gráficas del loss y el acierto se puede extraer que siguen las mismas formas que se han visto anteriormente. No parece que haya nada raro, en el loss los típicos picos y se mantiene por debajo de 2 en 100 epochs, donde sigue separándose muy rápido el loss de validación del de entrenamiento.

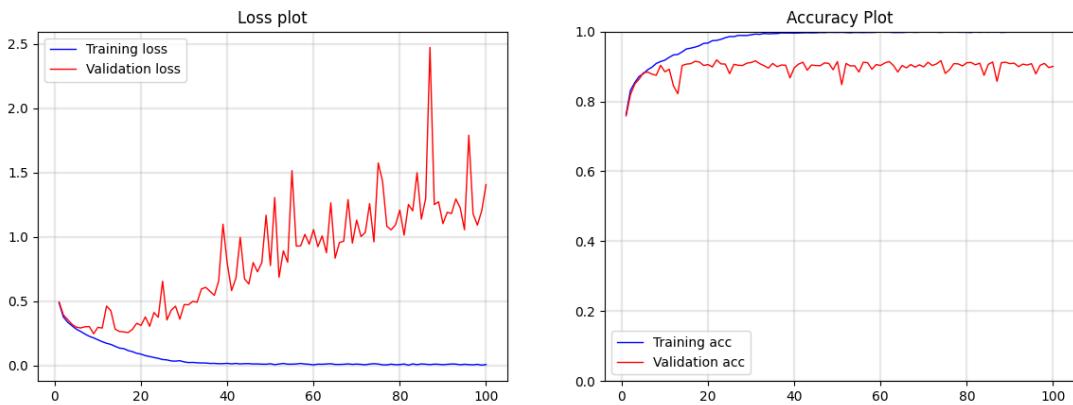


En entrenamiento parece que no ha perjudicado nada, manteniéndose el 100% de acierto. En validación se finaliza un 90,47% de acierto, que ha llegado a valores superiores al 91% durante el proceso. En cuanto al conjunto de test, se aciertan un 83,3% de las imágenes, aunque parece que el acierto está algo desbalanceado, 88,2% para cartoon y 78,4% para reales. Si se tiene como referencia el experimento 0004, el cual tiene imágenes de 256px, la tasa de acierto era del 84,7%, teniendo las clases más balanceadas. Por tanto, puede que el tamaño de la imagen esté influyendo algo, aunque se necesitan más pruebas para poder verlo.

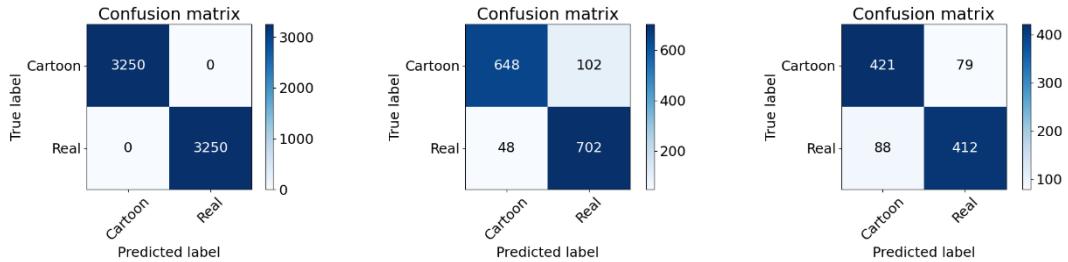
## 0045-CvR-CMCMCMCMD-0021

Imgs. train	Imgs. val.		Imgs. test	
6500 (3250+3250)	1500 (750+750)		1000 (500+500)	
Optimizador	F. pérdida	Tam. Imgs.	Batch_size	Epochs
RMSProp(lr=1e-4)	Binary_crossentropy	150 x 150	20	100
<b>Estructura</b>				
C(32, (3,3)) MP C(64, (3,3)) MP C(128, (3,3)) MP C(128, (3,3)) MP F D(512) D(1) Convolucionales y Densas con ReLU, salida con Sigmoid				

Esta prueba sigue las mismas instrucciones que la anterior, con la diferencia de que el tamaño de las imágenes ahora es de 150x150px, usando por tanto el conjunto *images\_150px*.



Tanto la gráfica del loss como la del acierto son prácticamente iguales que las anteriores, así que no hay mucha conclusión que sacar.

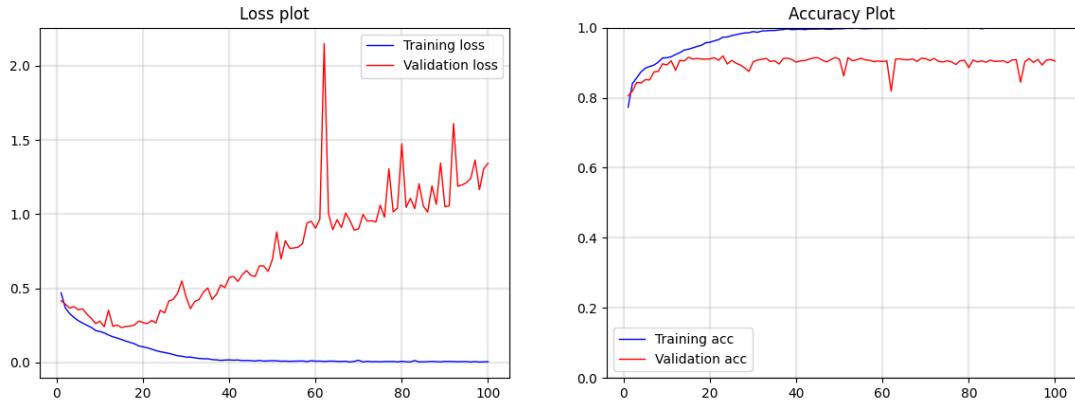


Al igual que antes, se acierta el 100% de las instancias para entrenamiento, mientras que para validación el acierto desciende hasta un 90%, lo cual no está nada mal. También se supera el 91% en algún epoch de manera similar al previo. Se finaliza el entrenamiento con un 83,3% de acierto, que está bastante bien teniendo en cuenta que además están las clases mejor balanceadas. Pese a ello, sigue estando a un 2% de el mejor experimento. Sin embargo, tiene el mis acierto que con la prueba anterior, por lo que parece que tampoco afecta mucho reducir el tamaño de las imágenes.

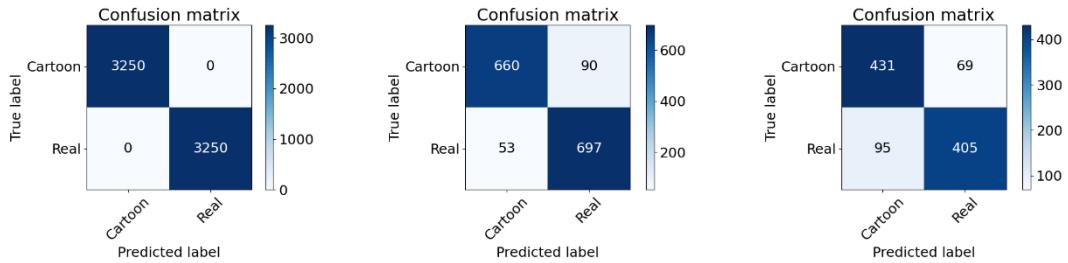
## 0046-CvR-CMCMCMCMD-0022

Imgs. train	Imgs. val.		Imgs. test	
6500 (3250+3250)	1500 (750+750)		1000 (500+500)	
Optimizador	F. pérdida	Tam. Imgs.	Batch_size	Epochs
RMSProp(lr=1e-4)	Binary_crossentropy	128 x 128	20	100
<b>Estructura</b>				
C(32, (3,3)) MP C(64, (3,3)) MP C(128, (3,3)) MP C(128, (3,3)) MP F D(512) D(1) Convolucionales y Densas con ReLU, salida con Sigmoid				

Mismo proceso que antes, con imágenes de 128x128px, siendo la mitad del tamaño original y usando el conjunto *images\_128px*.



Sigue habiendo la misma forma para ambas gráficas.



Se termina con un 90,47% de acierto sobre el conjunto de validación, mientras que para el de test se reduce hasta el 83,6%. No obstante, se puede ver en el log que en el epoch 99 se llega hasta 84,9%, siendo un valor muy alto que sobrepasa al mejor experimento previo. La diferencia es ínfima por lo que se toman como iguales.

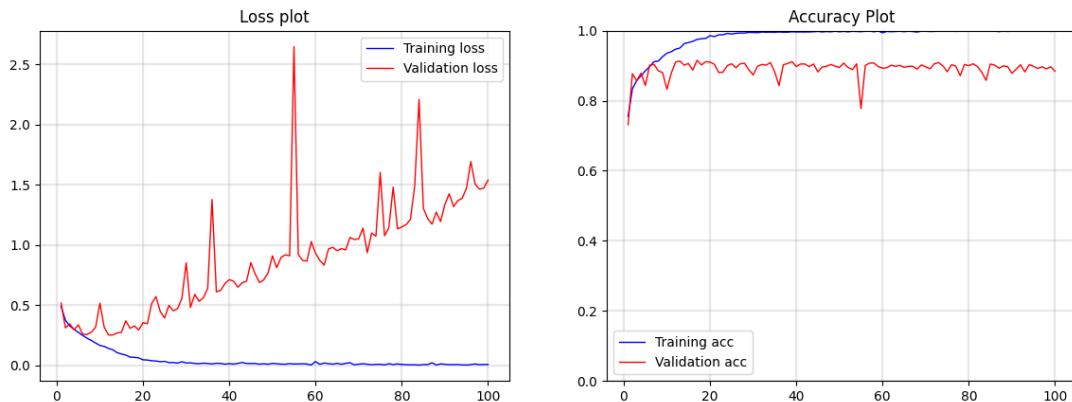
El caso es que todos los experimentos se han hecho con la misma red, sin modificar ningún parámetro. En ninguna capa convolucional hay padding, por lo que puede que haya pérdida de información con ciertos tamaños. Parece que 128 que es un divisor de 256 tiene un resultado prácticamente igual, mientras que 200 y 150 sacan unos resultados perores, aunque tampoco sea exagerado. Esto indica que la pérdida de información en estos dos últimos tamaños puede ser algo mayor y por tanto se tenga menos tasa de acierto. Debido a que los experimentos no tardan demasiado tiempo en finalizar, he decidido mantener el tamaño original propuesto, 256px, ya que las imágenes pueden tener algo de información extra sin que se tenga un sobrecoste computacional remarcable.

## 0047-CvR-CMCMCMCMD-0023

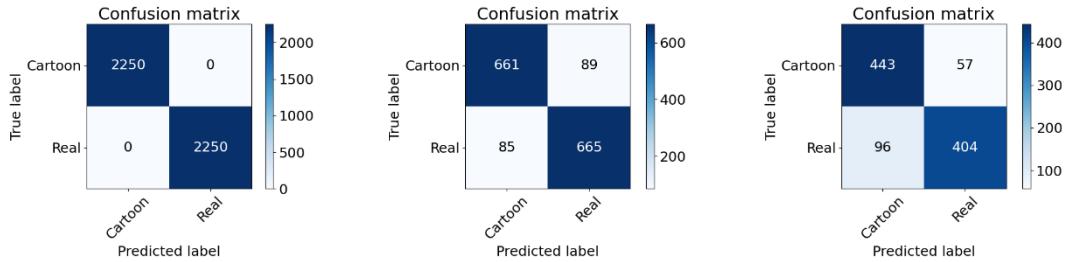
Imgs. train	Imgs. val.	Imgs. test		
4500 (2250+2250)	1500 (750+750)	1000 (500+500)		
Optimizador	F. pérdida	Tam. Imgs.	Batch_size	Epochs
RMSProp(lr=1e-4)	Binary_crossentropy	256 x 256	20	100
<b>Estructura</b>				
C(32, (3,3)) MP C(64, (3,3)) MP C(128, (3,3)) MP C(128, (3,3)) MP F D(512) D(1) Convolucionales y Densas con ReLU, salida con Sigmoid				

Esta prueba puede parecer la misma que la 0002, pero es totalmente diferente. La clave es que el conjunto de datos se ha modificado de tal forma que las imágenes originales sean más cuadradas para así evitar un estiramiento que altere la imagen. Para lograr esto, se ha recortado lateralmente las imágenes en un 20% del largo total, 10% por cada lado. Puede que en estas partes haya información relevante para alguna imagen, pero en general las escenas se suelen centrar y la información importante está en el centro.

Evitar el mencionado estiramiento puede ser importante porque altera totalmente la forma de las imágenes, haciendo que sean diferentes a como debería ser. Se ha usado el conjunto de datos *médium\_dataset\_squared*.



No parece que haya nada destacable en ninguna de las dos gráficas. En el loss se mantiene ese rizado característico con pico muy alto en determinadas ocasiones y en el acierto se registra alguna bajada atípica que siempre suele ocurrir. Por el resto los rizados son normales y el 100% de entrenamiento se alzan sobre el epoch 25.



En validación no parece haber ningún sesgo en las clases y se termina con un 88,4% de acierto. En test se alcanza en el epoch 100 un 84,7%, lo cual no está nada mal teniendo en cuenta que está igualando al mejor conjunto que dispone de 2000 imágenes más que este. Comparándolo con la prueba 0002 en la se tenía un 81,2% de acierto, sí que ha repercutido positivamente este cambio introducido. Revisando el log del actual experimento, se puede ver que en el epoch 97 se llega al máximo histórico de todas las pruebas, con un 85,4% de acierto sobre test. Son valores que realmente han mejorado el procesado que se hacía anteriormente, aunque conviene crear otra prueba para verificar esta hipótesis.

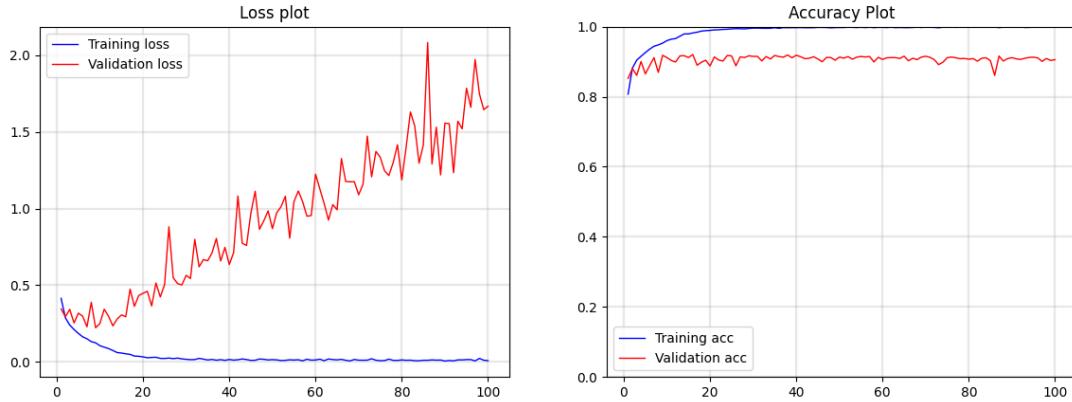
Para tener como referencia la media de los últimos 10 epochs, en la prueba 0002 se tiene una media de acierto sobre el conjunto de test del 80,84%, mientras que para la presente prueba (0023) es del 84,22%. La diferencia es de casi un 4%, siendo un valor muy alto según los acierto que se han venido viendo en los experimentos anteriores. Parece que se trata de una mejora que puede ser muy interesante.

#### 0048-CvR-CMCMCMCMD-0024

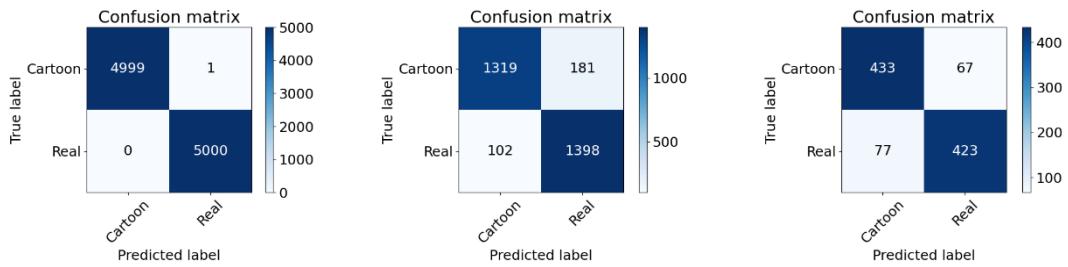
Imgs. train	Imgs. val.		Imgs. test	
10000 (5000+5000)	3000 (1500+1500)		1000 (500+500)	
Optimizador	F. pérdida	Tam. Imgs.	Batch_size	Epochs
RMSProp( $lr=1e-4$ )	Binary_crossentropy	256 x 256	20	100
<b>Estructura</b>				
C(32, (3,3)) MP C(64, (3,3)) MP C(128, (3,3)) MP C(128, (3,3)) MP F D(512) D(1) Convolucionales y Densas con ReLU, salida con Sigmoid				

Esta prueba presenta el tamaño del conjunto de imágenes definitivo sobre el que se va a desarrollar la red neuronal próximamente. Consta de 10000 imágenes para entrenamiento, 5000 reales y 5000 cartoon. De estas últimas, solo hay 2500 imágenes de dibujos animados originales, las otras 2500 se han generado con CartoonGAN a partir de los dibujos ya recogidos. Las imágenes de validación son originales y no hay ninguna

generada. Para el conjunto de test se mantiene el mismo que se ha venido viendo en los últimos experimentos, 1000 imágenes de 20 series/películas que jamás se han visto. Se usan imágenes que pueden sufrir de cierto estiramiento, por lo que la mejora introducida en el experimento 0023 no se está evaluando en este. Esto hace que se compare directamente con el mejor resultado antes visto que corresponde a la prueba 0004, que costa de 6500 imágenes. El conjunto usado es *big\_dataset*.



No parece que haya cambiado nada. Lo que sí es una buena noticia es que la curva de validación para el acierto se mantenga tan alta, ya que, aunque haya más imágenes para entrenar, también hay más para validar, donde podrían cometerse más errores. Por tanto, se puede ver que el acierto se mantiene sobre el 90%.



Se termina el entrenamiento con un 90,57% de acierto en validación y un 85,6% en test, siendo una marca que ha mejorado a la prueba 0004. Además, se puede ver que ambas clases están bien balanceadas y no parece haber ningún sesgo entre ellas. Revisando los últimos 10 epochs en el log, se encuentra un 87,1% de acierto siendo el máximo valor registrado hasta la fecha. La prueba 0004 tenía como acierto en test un 84,7%, por lo que se ha mejorado casi un 2,5% más introduciendo imágenes generadas por la GAN. Esto es muy importante ya que anteriormente se había demostrado que la GAN ayudaba a mejorar

el acierto generando más imágenes sin tener que recoger otras nuevas. Esto se vuelve a confirmar en este experimento.

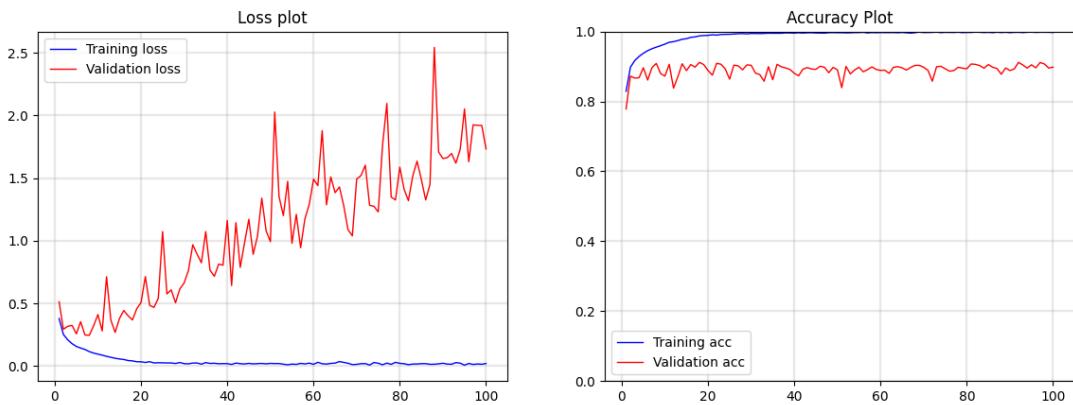
Para tener un registro más consistente, la media de los últimos 10 epochs de la prueba 0004 es del 83,22% mientras que para la presente prueba (0024) es del 85,57%. Esto quiere decir que la media de la prueba 0024 es superior al máximo de la prueba 0004, que era 84,7%. Puede concluirse que tener un conjunto generado artificialmente mejora el acierto, por lo que con conjuntos de entrenamiento mayores se consiguen aciertos mayores.

Podría hacerse un conjunto de 100000 imágenes, aunque sería excesivamente laborioso. Teniendo estos resultados tan próximos al 90%, se va a finalizar las pruebas con el conjunto de 10000 imágenes.

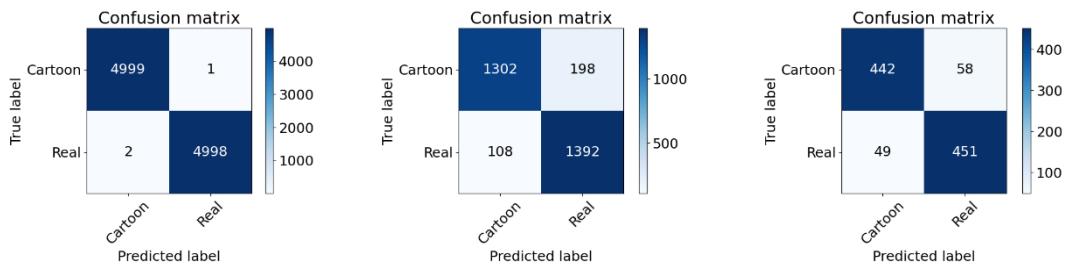
#### **0049-CvR-CMCMCMCMD-0025**

Imgs. train	Imgs. val.		Imgs. test	
10000 (5000+5000)	3000 (1500+1500)		1000 (500+500)	
Optimizador	F. pérdida	Tam. Imgs.	Batch_size	Epochs
RMSProp(lr=1e-4)	Binary_crossentropy	256 x 256	20	100
<b>Estructura</b>				
C(32, (3,3)) MP C(64, (3,3)) MP C(128, (3,3)) MP C(128, (3,3)) MP F D(512) D(1) Convolucionales y Densas con ReLU, salida con Sigmoid				

Este es el último experimento sobre el estudio del conjunto de datos recogido, teniendo un total de 25. Se introduce la mejora vista en la prueba 0023, que consiste en la reducción del largo de una imagen del 20% para evitar deformaciones en las formas al hacer las imágenes cuadradas. Se usa el conjunto más grande, 10000 imágenes, guardado en la carpeta *big\_dataset\_squared*.



En las gráficas no parece que haya diferencias remarcables, aunque esto ya había pasado en el experimento 0023 al introducir la mejora. En las matrices de confusión se apreciará mejor el resultado obtenido.



Las expectativas estaban muy altas y parece que se han cumplido. En el conjunto de validación no hay nada destacable, se mantiene sobre el 90% siempre. Pero hay algo muy interesante en el conjunto de test, el acierto es del 89,3%, volviéndose a romper la mejor marca conseguida hasta ahora. En la anterior prueba se conseguía un 87,1%, y ahora se ha mejorado un 2% más, teniendo las clases bastante bien balanceadas. Se considera un total éxito la mejora introducida, corroborando lo visto en la prueba 0023.

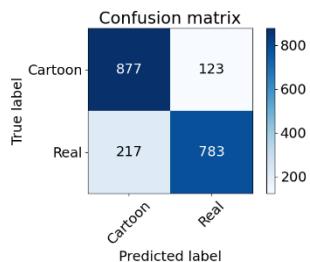
La media de los últimos 10 epochs de la prueba 0024 es de 85,57% de acierto, mientras que en la presente prueba (0025) se consigue una media de 88,08%. Se confirma totalmente que evitar que las formas se alteren es muy importante. El estiramiento que se estaba produciendo en las imágenes al pasar una imagen rectangular a cuadrada sin recorte afectaba negativamente al rendimiento de forma notable.

Tras los 25 experimentos se ha conseguido llegar a un valor muy cercano al 90%, por lo que se espera que con mejoras en la red se pueda llegar a superar. Como objetivo optimista, se pretende llegar al 95% de acierto.

## 0050-CvR-CMCMCMCMD-0026

Imgs. train	Imgs. val.		Imgs. test	
6500 (3250+3250)	3000 (1500+1500)		2000 (1000+1000)	
Optimizador	F. pérdida	Tam. Imgs.	Batch_size	Epochs
RMSProp(lr=1e-4)	Binary_crossentropy	256 x 256	20	100
<b>Estructura</b>				
C(32, (3,3)) MP C(64, (3,3)) MP C(128, (3,3)) MP C(128, (3,3)) MP F D(512) D(1) Convolucionales y Densas con ReLU, salida con Sigmoid				

En este experimento se incluye un nuevo conjunto de test de 2000 imágenes, recogidas de 40 series distintas que no se han visto por la red nunca. Se usa el modelo obtenido en la prueba 0004, para ver cómo va el rendimiento en este conjunto nuevo. Este conjunto se puede encontrar en *big\_dataset\_squared/test\_big*.



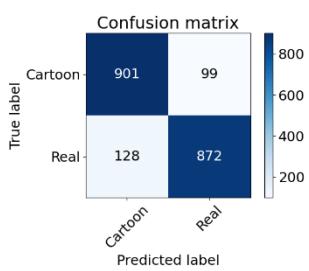
En la matriz de confusión puede verse que hay cierto sesgo donde se predicen bastantes más cartoon que reales. El acierto es del 83%, lo cual ha bajado ligeramente con respecto a la prueba 0004 y el conjunto de 1000 imágenes, que era del 84,7%. Lo más probable

eran que la tasa de acierto disminuyera, porque se han añadido el doble de imágenes con dibujos diferentes. En el caso de que los dibujos fueran similares podría haber subido perfectamente, pero se trata de sacar un conjunto de test que se ajuste a la realidad lo máximo posible.

## 0051-CvR-CMCMCMCMD-0027

Imgs. train	Imgs. val.		Imgs. test	
10000 (5000+5000)	3000 (1500+1500)		2000 (1000+1000)	
Optimizador	F. pérdida	Tam. Imgs.	Batch_size	Epochs
RMSProp(lr=1e-4)	Binary_crossentropy	256 x 256	20	100
<b>Estructura</b>				
C(32, (3,3)) MP C(64, (3,3)) MP C(128, (3,3)) MP C(128, (3,3)) MP F D(512) D(1) Convolucionales y Densas con ReLU, salida con Sigmoid				

Para esta prueba se ha hecho lo mismo que la anterior, usar el conjunto de test de 2000 imágenes, pero esta vez con el modelo 0025, que tenía el mejor resultado hasta ahora. Este conjunto se puede encontrar en *big\_dataset\_squared/test\_big*.

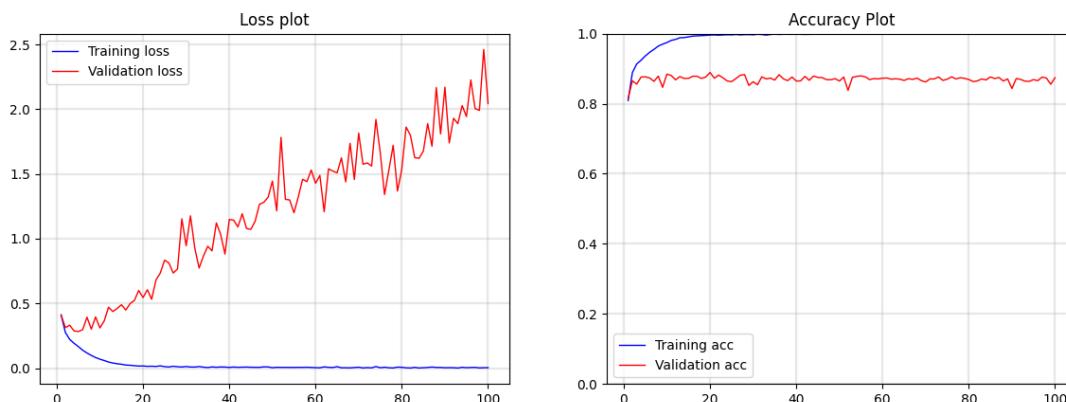


En este caso la tasa de acierto es mayor, 88,62% y no se despega tanto de la prueba 0025, que eran del 89,3%. Esto indica que el modelo de esta prueba era bastante bueno y generalizaba decentemente. Con un conjunto más grande apenas hay diferencia del 0,5% de acierto. El caso es que parece que los modelos no están desempeñando mal con un conjunto de test nuevo. Además, en este caso no hay tanto desbalanceo en las predicciones, están bastante igualados.

## 0052-CvR-CMCMCMCMD-0028

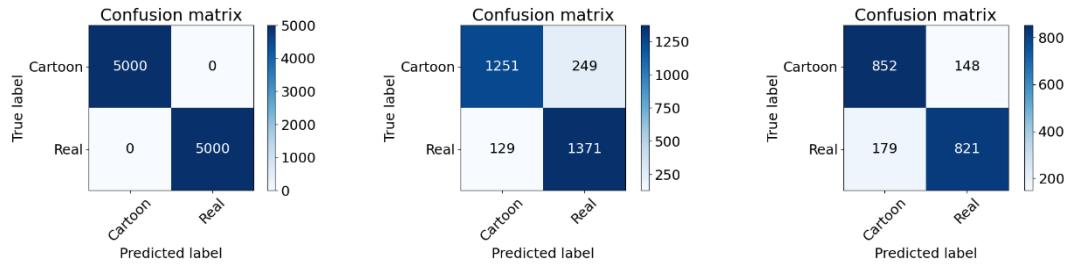
Imgs. train	Imgs. val.		Imgs. test	
10000 (5000+5000)	3000 (1500+1500)		2000 (1000+1000)	
Optimizador	F. pérdida	Tam. Imgs.	Batch_size	Epochs
RMSProp(lr=1e-4)	Binary_crossentropy	256 x 256	20	100
<b>Estructura</b>				
C(32, (3,3)) MP C(32, (3,3)) MP C(32, (3,3)) MP F D(512) D(1) Convolucionales y Densas con ReLU, salida con Sigmoid				

En esta prueba se comienzan a experimentar con las arquitecturas de la red de neuronas. Como se puede ver, la primera aproximación se hará con tres capas convolucionales de filtros de 3x3. El número de filtros para todas es 32 y también tienen padding para evitar pérdidas de información.



Lo primero que se puede apreciar es que en el loss se separa muy rápido validación de entrenamiento, aunque esto se ha venido viendo en casos anteriores. Parece que en cuanto al acierto se queda en valores ligeramente superiores al 85% para la validación. Si cabe destacar que el entrenamiento llega al 100% muy rápido, antes del epoch 20. Esto puede

indicar que está aprendiendo demasiado del entrenamiento, produciendo overfitting que implica una mala generalización.



En las matices se puede apreciar mejor lo descrito antes. En entrenamiento hay un 100%, en validación un 87,4% y en test un 83,65%. No son valores tan altos como se han visto anteriormente, de hecho, se quedan bastante lejos. Lo que sí podría decirse es que la clases están balanceadas en test, aunque no tanto en validación, donde se tienen a predecir más reales que cartoons. Como se ha venido haciendo en anteriores pruebas, puede ser muy útil ver la media de los últimos 10 epochs, que en este caso se sitúa en un 83,4%, así que es bastante parecido a lo visto en la matriz de test.

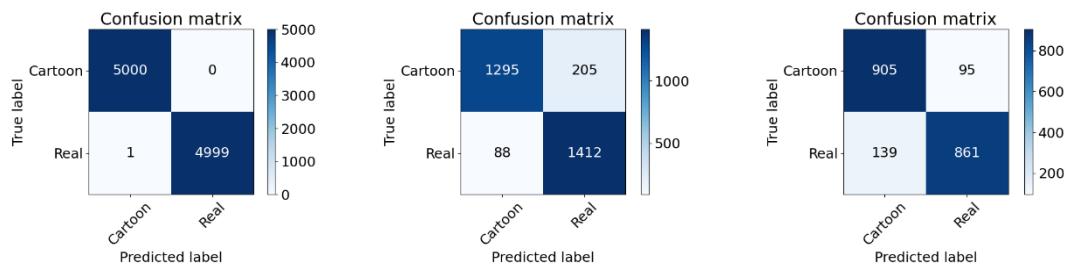
### 0053-CvR-CMCMCMCMD-0029

Imgs. train	Imgs. val.		Imgs. test	
10000 (5000+5000)	3000 (1500+1500)		2000 (1000+1000)	
Optimizador	F. pérdida	Tam. Imgs.	Batch_size	Epochs
RMSProp( $lr=1e-4$ )	Binary_crossentropy	256 x 256	20	100
<b>Estructura</b>				
C(32, (3,3)) MP C(32, (3,3)) MP C(32, (3,3)) MP C(32, (3,3)) MP F D(512) D(1) Convolucionales y Densas con ReLU, salida con Sigmoid				

En este caso se repite el mismo método que la prueba anterior, pero añadiendo una capa convolucional y maxpoling más, siendo un total de cuatro.



Se ve claramente que el loss ha bajado con respecto a antes, no llega a valores tan altos, aunque si tiene algún pico. Además, el loss de validación está algo más pegado al loss de entrenamiento. En el acierto se puede apreciar una subida en validación, mientras que para entrenamiento ahora se tarda más epochs en llegar al 100% (en torno a 30).



La matriz de validación vuelve a mostrar cierto sesgo hacia las reales, aunque en el log se puede ver que esto va cambiando de epoch en epoch. Se acaba con un 90,23% de acierto en validación, que es por donde ha estado fluctuando durante todo el entrenamiento. Para test se predicen más cartoon. En total se llega a un 88,3% de acierto y de media en los últimos 10 epochs se obtiene un 88,64%. Esto indica que el último epoch no ha sido un máximo, ya que la media está ligeramente por encima de él.

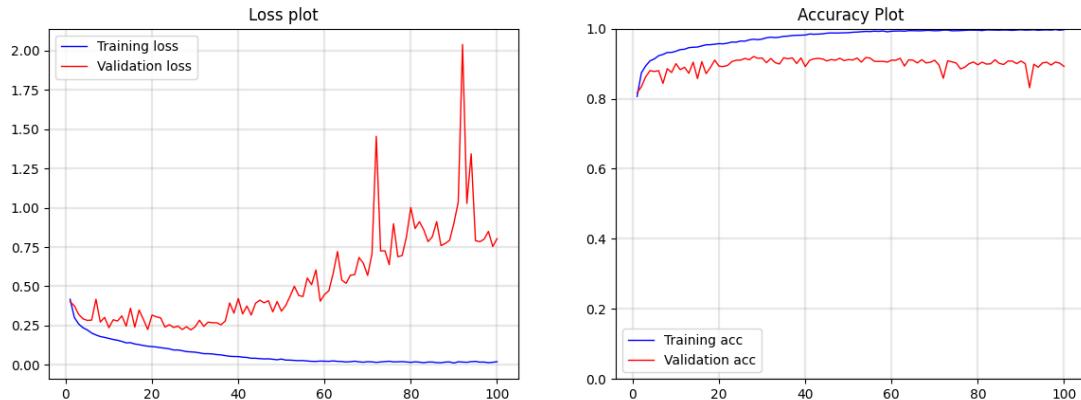
El caso es que añadir una capa convolucional (cuatro) parece tiene mejor rendimiento que tener tres. Las medias se distancian un 5%, lo cual es mucho y más para las cifras vistas a lo largo de toda la experimentación. Añadir más capas permite que se detecten más características, sobre todo cuando se dispone de un conjunto de datos grande. Por tanto, quedarse con tres capas no parece suficiente debido a que no recoge suficientes rasgos para generalizar mejor. En las imágenes reales hay rasgos muy parecidos, como son las formas de las personas, que en la vida real son casi iguales. Puede que en el caso de

cartoons se más complejo porque hay más formas distintas para personajes, aunque para el resto de objetos suelen ser realmente parecidas. Además, en dibujos animados las formas son muchísimo menos complejas que las reales. Aún así, el número de parámetros es bastante grande, 4 millones, lo que puede indicar que hay un exceso de parámetros ya que cuanto mayor sea la imagen de salida de las CNN mayor será los parámetros por las capas densas del clasificador final. Por tanto, aumentar el número de capas aumentará el número de parámetros en la capas convolucionales, pero en total habrá mucho menos por el clasificador final.

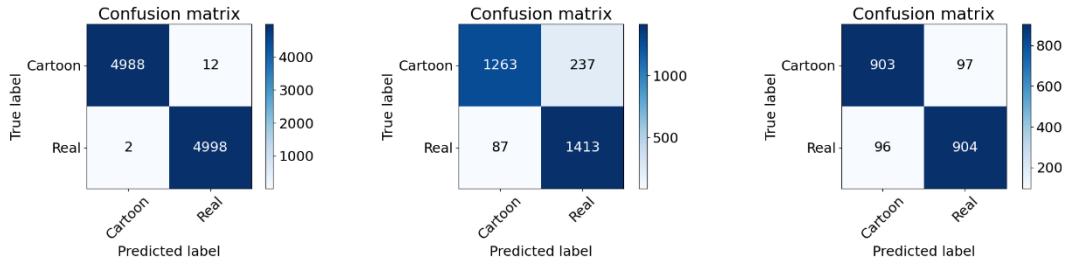
## 0054-CvR-CMCMCMCMD-0030

Imgs. train	Imgs. val.	Imgs. test		
10000 (5000+5000)	3000 (1500+1500)	2000 (1000+1000)		
Optimizador	F. pérdida	Tam. Imgs.	Batch_size	Epochs
RMSProp(lr=1e-4)	Binary_crossentropy	256 x 256	20	100
<b>Estructura</b>				
C(32, (3,3)) MP F D(512) D(1) Convolucionales y Densas con ReLU, salida con Sigmoid				

Vuelve a ser la misma idea de experimento pero añadiendo una capa más al anterior, teniendo un total de cinco capas.



Parece que algo está mejorando en el loss, todavía ha bajado más la curva, aunque haya esos picos tan altos. Ahora tarda más en despegarse que en ninguna otra prueba que se haya hecho en este documento. En el acierto de validación también se repite lo que hemos visto anteriormente, se tarda más en llegar al 100% de acierto en entrenamiento cuantas más capas tenga. Esto puede indicar que generaliza mejor porque no es capaz de ajustarse al conjunto de entrenamiento fácilmente.

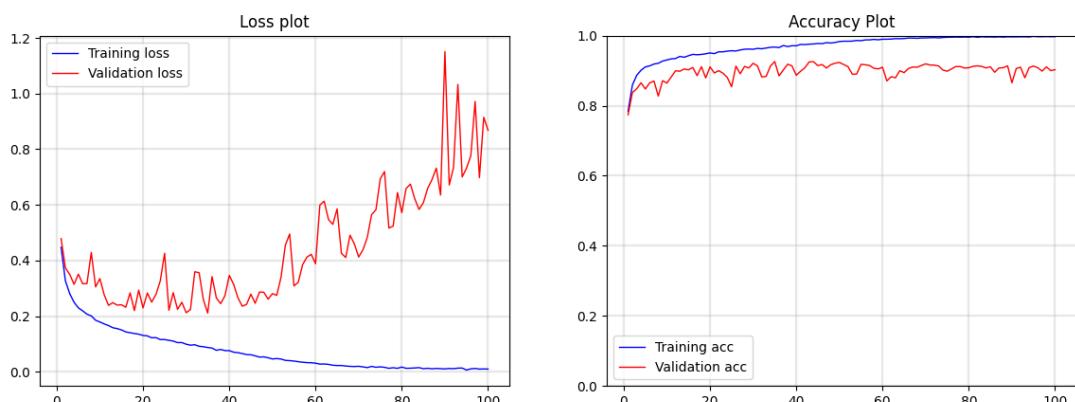


En validación se finaliza con un 89,2% de acierto, aunque durante a mayor parte del entrenamiento se ha estado en valores superiores al 90%. En test se pasa por primera vez la barrera del 90%, exactamente se llega a un 90,35% de acierto, como se puede ver en la matriz de test. El balance de las clases es excelente, ya que son casi exactamente igual. La media de los últimos 10 epochs se sitúa en un 89,57% de acierto. Cabe mencionar que los últimos 5 epochs parecen que han mejorado con respecto a los anteriores, ya que se han mantenido todos por encima del 90% de acierto. Puede que haya que experimentar con más epochs para ver si se necesita un proceso de entrenamiento más largo.

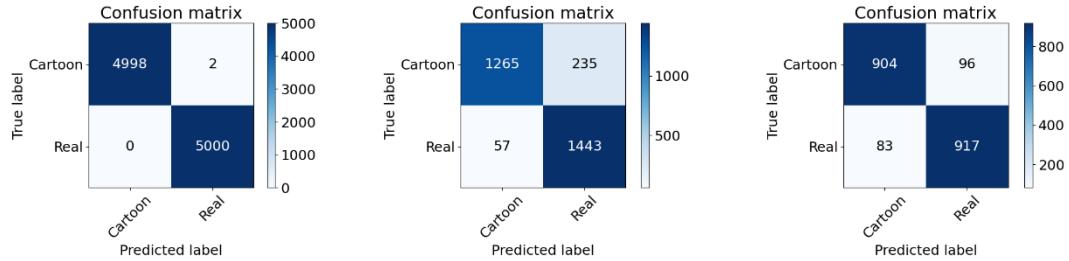
### 0055-CvR-CMCMCMCMD-0031

Imgs. train	Imgs. val.		Imgs. test	
10000 (5000+5000)	3000 (1500+1500)		2000 (1000+1000)	
Optimizador	F. pérdida	Tam. Imgs.	Batch_size	Epochs
RMSProp(lr=1e-4)	Binary_crossentropy	256 x 256	20	100
<b>Estructura</b>				
C(32, (3,3)) MP C(32 (3,3)) MP C(32, (3,3)) MP C(32, (3,3)) MP C(32, (3,3)) MP F D(512) D(1) Convolucionales y Densas con ReLU, salida con Sigmoid				

Se sigue continuando con los experimentos anteriores, llegando a seis capas convolucionales y maxpoling.



Parece que se siguen viendo los mismo patrones que anteriormente, el loss sigue bajando y el acierto de entrenamiento cada vez tarda más en llegar al 100%. Se debería probar con más epochs, como se ha dicho antes.



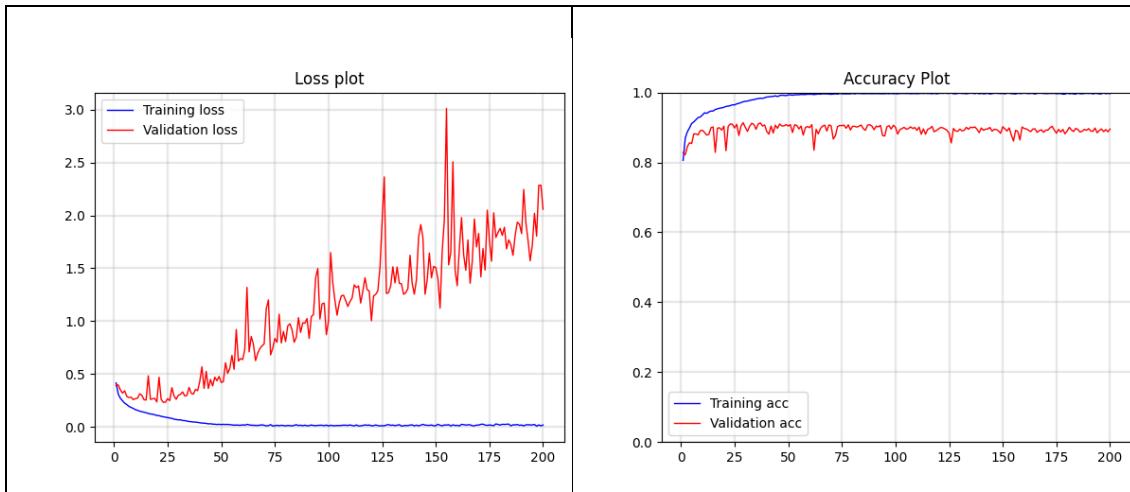
En validación el acierto se mantiene por encima del 90% en casi todo el proceso, finalizando con un 90,27% de acierto. Para el caso de test, vuelve a tenerse el mejor resultado, con un 91,05%. La media de los últimos 10 epochs para test es del 90,48%. Esta cifra vuelve a ser superior a la tratada anteriormente, que era del 89,57, aunque se había dicho que podía requerir más epochs para ver si hay una diferencia sustancial.

Para tener pruebas más consistentes, se repetirán la prueba 0030 con más epochs paa ver si se trataba de eso, aunque en validación no lo pareciera.

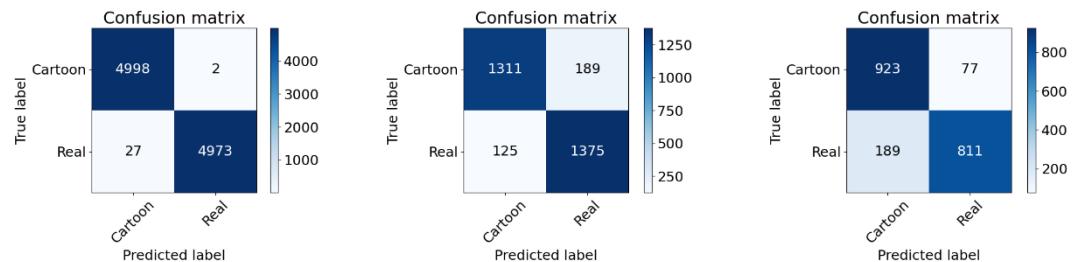
## 0056-CvR-CMCMCMCMD-0032

Imgs. train	Imgs. val.		Imgs. test	
10000 (5000+5000)	3000 (1500+1500)		2000 (1000+1000)	
Optimizador	F. pérdida	Tam. Imgs.	Batch_size	Epochs
RMSProp(lr=1e-4)	Binary_crossentropy	256 x 256	20	200
<b>Estructura</b>				
C(32, (3,3)) MP F D(512) D(1) Convolucionales y Densas con ReLU, salida con Sigmoid				

Se trata de la mis prueba que la 0030 pero cambiando el número de epochs a 200, para ver si la red aún no ha terminado de entrenar y podía mejorar.



No parece que cambien las gráficas con respecto al experimento 0030. Hay una ligera pendiente negativa en el acierto, que parece que cae poco a poco a partir del epoch 10. Esto significa que no parece haber mejorado nada en validación.



Las gráficas de entrenamiento y validación son muy parecidas a lo que se había visto, acabando con un 89,53% en validación. En test el acierto ha bajado mucho, se nota que ha ocurrido un sobre ajuste y no generaliza tan bien. El desbalanceo de las clases es tremendo, se tiene un 55,6% de predicciones para cartoon y un 44,4% para real. El acierto al final es del 86,7% y la media de los 10 últimos epochs es del 88%. En la prueba 0030 se tenía una media del 89,57% y se supera el 90% en algún epoch, mientras que en esta prueba se puede ver que eso nunca ocurre.

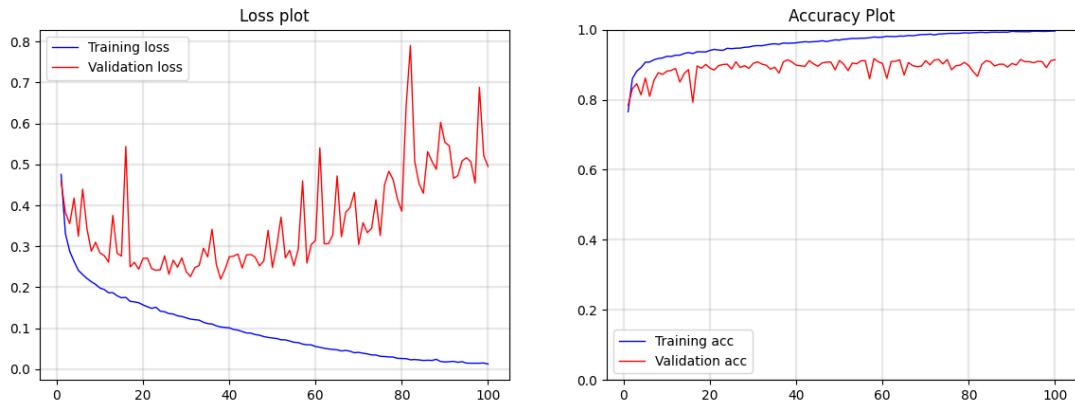
Por tanto, se descartan las cinco capas y se toman seis capas como la mejor opción. Parece que hay demasiados rasgos que se pueden aprender de un conjunto así de grande. Puede tener sentido por el hecho de la variedad de formas que hay debido a que se han usado muchas series a la hora de crear el conjunto.

### 0057-CvR-CMCMCMCMD-0033

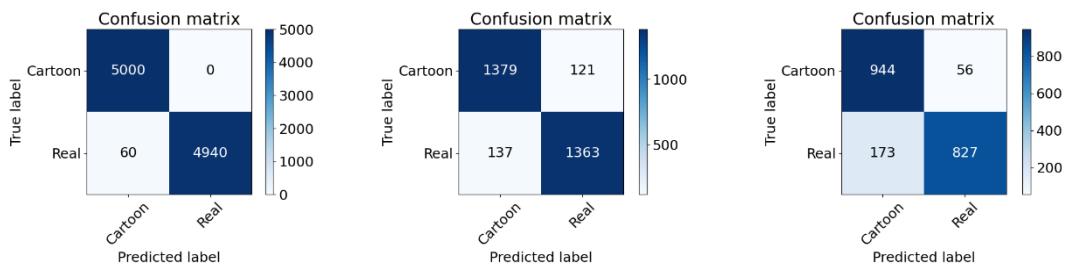
Imgs. train	Imgs. val.	Imgs. test
10000 (5000+5000)	3000 (1500+1500)	2000 (1000+1000)

Optimizador	F. pérdida	Tam. Imgs.	Batch_size	Epochs
RMSProp( $lr=1e-4$ )	Binary_crossentropy	256 x 256	20	100
<b>Estructura</b>				
C(32, (2,2)) MP C(32 (2,2)) MP C(32, (2,2)) MP C(32, (2,2)) MP C(32, (2,2)) MP F D(512) D(1)				
Convolucionales y Densas con ReLU, salida con Sigmoid				

Los siguientes experimentos tratarán de ver cuál es el mejor tamaño de los filtros para una arquitectura con seis capas. Esta primera prueba comienza con un tamaño del kernel de 2x2.



Partiendo como referencia el experimento 0031, a primera vista parece que el los es muy parecido, aunque parece que es algo más alto en esta ocasión. En la gráfica del acierto se puede ver como el entrenamiento llega al 100% pero algo más tarde, ya entrando en el epoch 100 casi. Puede que se necesiten más epochs para el entrenamiento, aunque en el log no parece que la cosa tenga muy buena pinta.



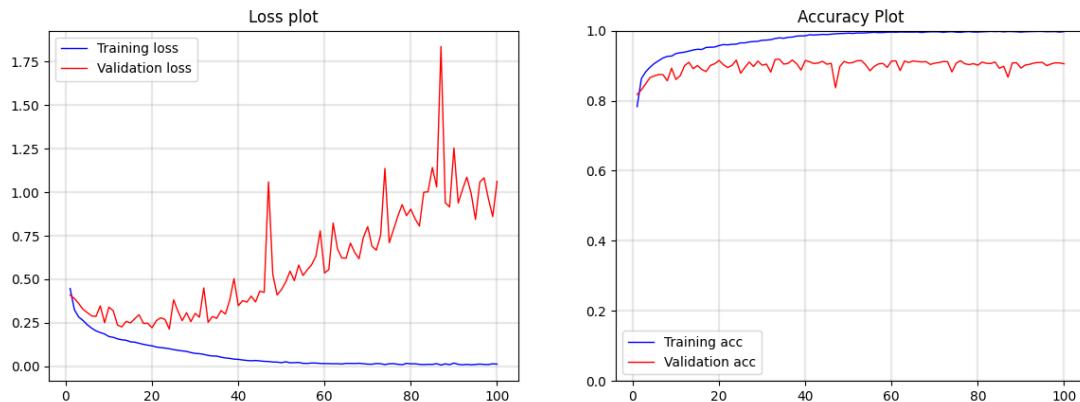
Llama la atención el gran desbalanceo que hay entre las clases para la matriz de test, esto ha pasado en varios epochs, aunque hay otros en los que no. Es necesario relanzar esta prueba por lo que se comentará más tarde. Dado que no parecen resultados de haber

completado el entrenamiento, espero a ver la prueba con más epochs para poder tener una conclusión mejor.

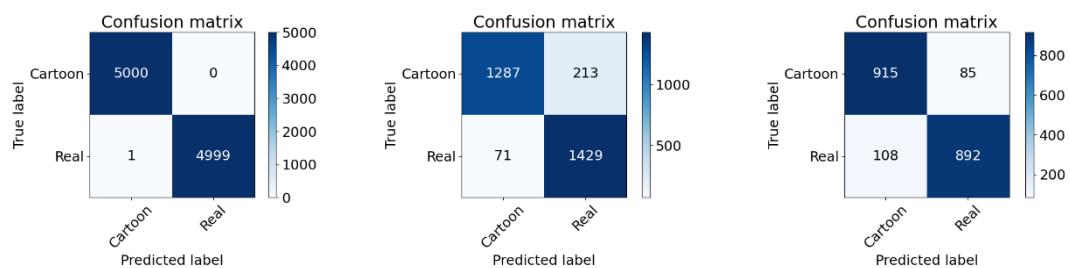
### 0058-CvR-CMCMCMCMD-0034

Imgs. train	Imgs. val.	Imgs. test		
10000 (5000+5000)	3000 (1500+1500)	2000 (1000+1000)		
Optimizador	F. pérdida	Tam. Imgs.	Batch_size	Epochs
RMSProp( $lr=1e-4$ )	Binary_crossentropy	256 x 256	20	100
<b>Estructura</b>				
C(32, (4,4)) MP C(32 (4,4)) MP C(32, (4,4)) MP C(32, (4,4)) MP C(32, (4,4)) MP F D(512) D(1) Convolucionales y Densas con ReLU, salida con Sigmoid				

Sigue la misma configuración que el experimento anterior, esta vez con tamaño de kernel 4x4.



Se nota que el loss tiene mayores valores con respecto a la prueba 0031, despegándose algo más al final del entrenamiento. La curva del entrenamiento indica que se alcanza el 100 antes que en la prueba 0031, aunque los valores de validación parecen que son muy parecidos.

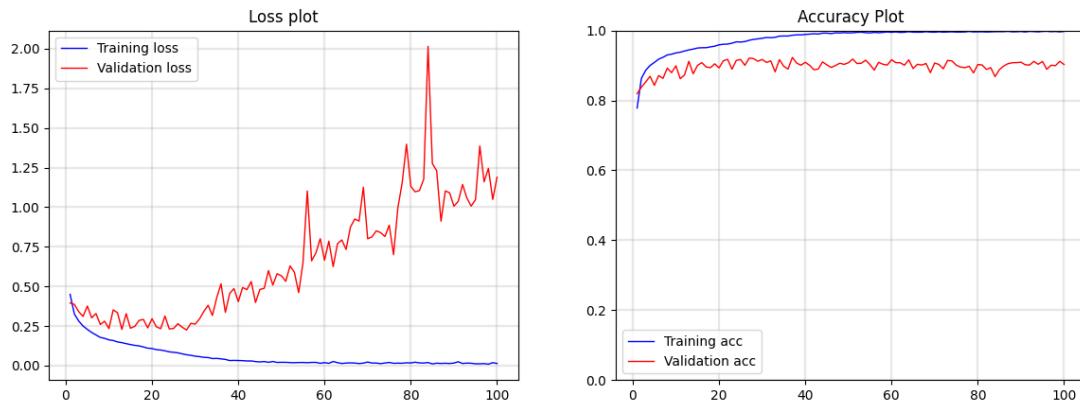


En validación se finaliza con un desbalanceo a favor de la clase real, con un 90,53% de acierto en total. No parece que haya grandes cambios en el resto de epochs (se puede ver en el log), es muy parecida a la prueba 0031. En test se consigue un acierto de 90,35% lo cual está en valores muy altos y parecidos, puede que no haya mucho cambio a la hora de elegir un tamaño de filtro u otro. La media de los últimos 10 epochs es del 89,79% por lo que es muy parecido a lo que se ha visto antes, además de que varios epochs superan el 90%. También es verdad que parece que hay un ligero desbalanceo de clase en la amtriz de test, pero es muy poco.

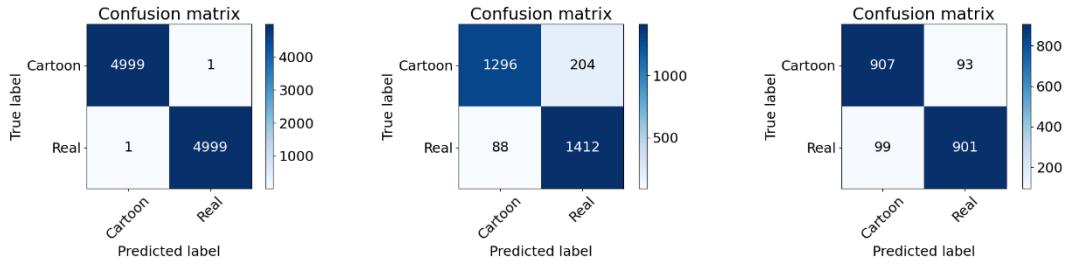
### 0059-CvR-CMCMCMCMD-0035

Imgs. train	Imgs. val.	Imgs. test		
10000 (5000+5000)	3000 (1500+1500)	2000 (1000+1000)		
Optimizador	F. pérdida	Tam. Imgs.	Batch_size	Epochs
RMSProp(lr=1e-4)	Binary_crossentropy	256 x 256	20	100
<b>Estructura</b>				
C(32, (5,5)) MP C(32 (5,5)) MP C(32, (5,5)) MP C(32, (5,5)) MP C(32, (5,5)) MP C(32, (5,5)) MP F D(512) D(1) Convolucionales y Densas con ReLU, salida con Sigmoid				

Sigue la misma configuración que el experimento anterior, esta vez con tamaño de kernel 5x5.



Ambas gráficas son muy parecidas al experimento anterior (0034), manteniendo que el entrenamiento llega al 100% antes que el experimento 0031.



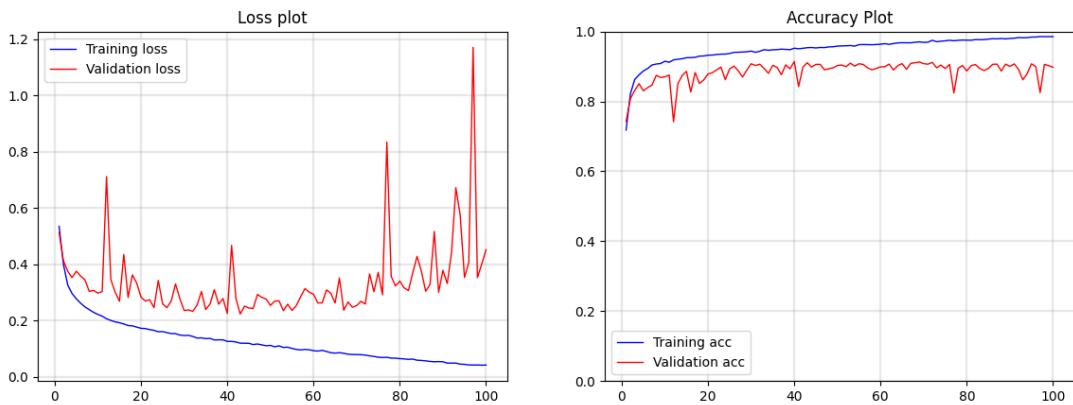
Vuelve a haber cierto sesgo hacia las imágenes reales para el conjunto de validación, llegando a un 90,27% de acierto al final del entrenamiento. En el log se puede ver que todos los epochs están en torno al 90%, muy similar al resto de tamaños de kernel. En cuanto a la matriz de test, sí está balanceada y se alcanza un 90,4% de acierto. La media de los últimos 10 epochs es del 90,43%, que es casi exactamente igual que la del experimento 0031 que era del 90,45%.

Después de estas pruebas sobre el tamaño del kernel, parece que no tiene una influencia relevante a la hora de tener más acierto sobre el conjunto de test. En general se suelen preferir tamaños de kernel impares porque así hay un píxel central, mientras que, si es positivo, pueden ocurrir distorsiones porque no se divide simétricamente al no tener píxel central. Por tanto, se seguirá teniendo como referencia el modelo 0031, que además tiene menos parámetros por lo que el modelo es más liviano.

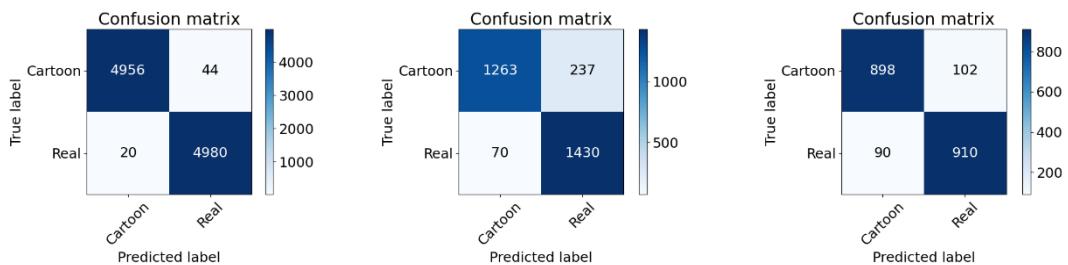
## 0060-CvR-CMCMCMCMD-0036

Imgs. train		Imgs. val.		Imgs. test	
10000 (5000+5000)		3000 (1500+1500)		2000 (1000+1000)	
Optimizador		F. pérdida	Tam. Imgs.	Batch_size	Epochs
RMSProp( $lr=1e-4$ )		Binary_crossentropy	256 x 256	20	100
<b>Estructura</b>					
C(16, (3,3)) MP C(16 (3,3)) MP C(16, (3,3)) MP C(16, (3,3)) MP C(16, (3,3)) MP C(16, (3,3)) MP F D(512) D(1) Convolucionales y Densas con ReLU, salida con Sigmoid					

Este experimento en conjunto de los tres siguientes tienen como objetivo ver cómo afecta tener más o menos filtros en las capas convolucionales.



Si se comparan las gráficas con las del experimento 0031 que es el de referencia, podría decirse que el loss es inferior y se despega menos la validación del entrenamiento. Aunque esto no es cierto, puede engañar ya que el loss no llega a 0 durante el entrenamiento. También se puede ver en el acierto que la curva no ha terminado de converger al 100% en entrenamiento, aunque la de validación ya parece estancada. Esto puede indicar que tener tan pocos filtros hace que no haya suficientes patrones reconocibles y por tanto se requiera más tiempo para entrenar y para aprender de las imágenes.



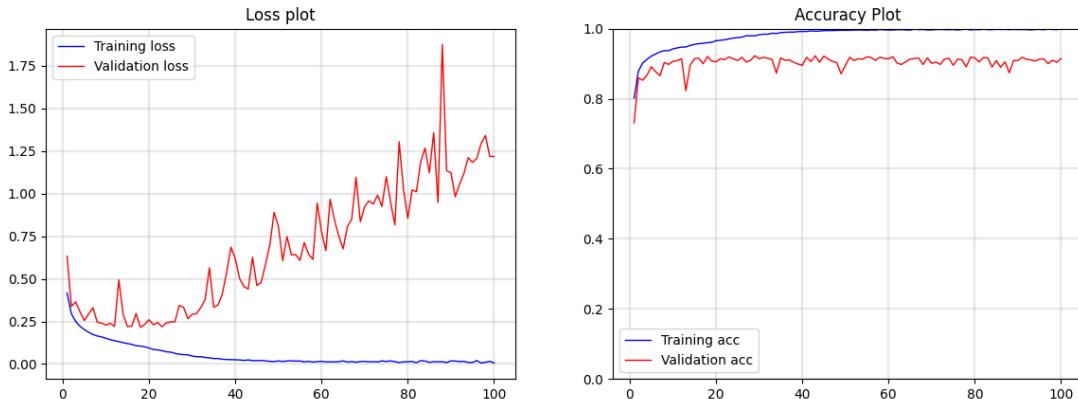
La matriz de confusión para validación acaba con un 89,77% de acierto, donde está permanentemente alrededor del 90%. Para test se termina con un 90,4% y la media de los últimos 10 epochs es del 89,27%, que es inferior a lo que se había visto antes, que era 90,48% en el 0031. Parece que sí hay una diferencia notable al pasar de 16 a 32 filtros obteniendo casi un 1% más con 32 filtros. Puede que no se detecten suficientes formas con tan poco filtro y por eso pierda ese rendimiento extra.

## 0061-CvR-CMCMCMCMD-0037

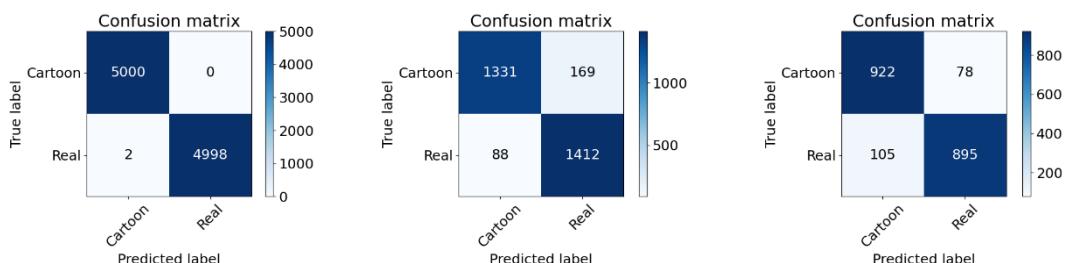
Imgs. train	Imgs. val.	Imgs. test		
10000 (5000+5000)	3000 (1500+1500)	2000 (1000+1000)		
Optimizador	F. pérdida	Tam. Imgs.	Batch_size	Epochs
RMSProp(lr=1e-4)	Binary_crossentropy	256 x 256	20	100

Estructura							
C(64, (3,3)) MP F D(512) D(1)							
Convolucionales y Densas con ReLU, salida con Sigmoid							

En esta prueba el número de filtros es 64.



La gráfica del loss se parece más a la de la prueba 0031, aunque la curva se va más arriba. El entrenamiento finaliza antes, sobre el epoch 40 ya se ha terminado prácticamente, mientras que en el 0031 termina sobre el 70. Por eso parece que la curva tiene mayor pendiente, pero también parece que la curva del acierto de validación es algo más alta, aunque puede ser un efecto visual.

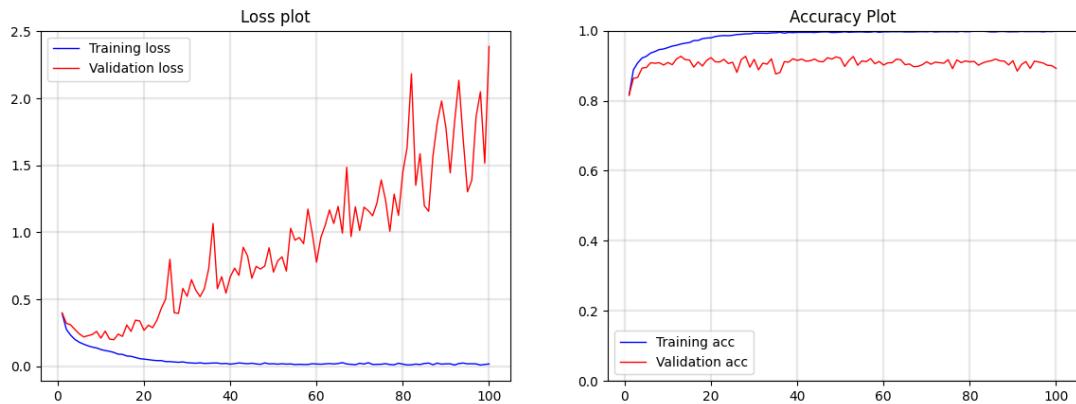


En validación se finaliza con un 91,43% de acierto y en la mayoría de los epochs se alcanza esta cifra, por lo que parece que la curva sí era algo superior al 0031. En test se termina con un 90,85% que es un acierto muy elevado, ligeramente a la prueba anterior (0036). La media de los últimos 10 epochs es del 90,4%, siendo casi igual que el 0031 que tenía 90,48%. Pasar de 32 a 64 no parece que haya tenido ninguna ventaja destacable, teniendo los mismo resultados. En este caso (0037), hay un ligero desbalanceo en la matriz de test a favor de cartoon, de la que se realizan más predicciones.

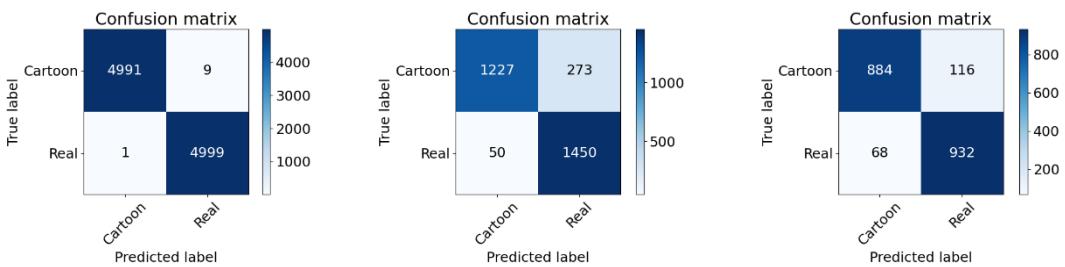
## 0062-CvR-CMCMCMCMD-0038

Imgs. train	Imgs. val.	Imgs. test		
10000 (5000+5000)	3000 (1500+1500)	2000 (1000+1000)		
Optimizador	F. pérdida	Tam. Imgs.	Batch_size	Epochs
RMSProp(lr=1e-4)	Binary_crossentropy	256 x 256	20	100
<b>Estructura</b>				
C(128, (3,3)) MP C(128, (3,3)) MP F D(512) D(1) Convolucionales y Densas con ReLU, salida con Sigmoid				

Se continúa experimentando con el número de filtros, en este caso 128.



El entrenamiento todavía finaliza antes, sobre el epoch 30, siguiendo la misma tendencia que se ha visto en ocasiones anteriores. En la gráfica del acierto parece que comienza a haber un ligero sobreajuste al final del proceso, aunque puedes ser simplemente un pico como los tantos que hay antes.



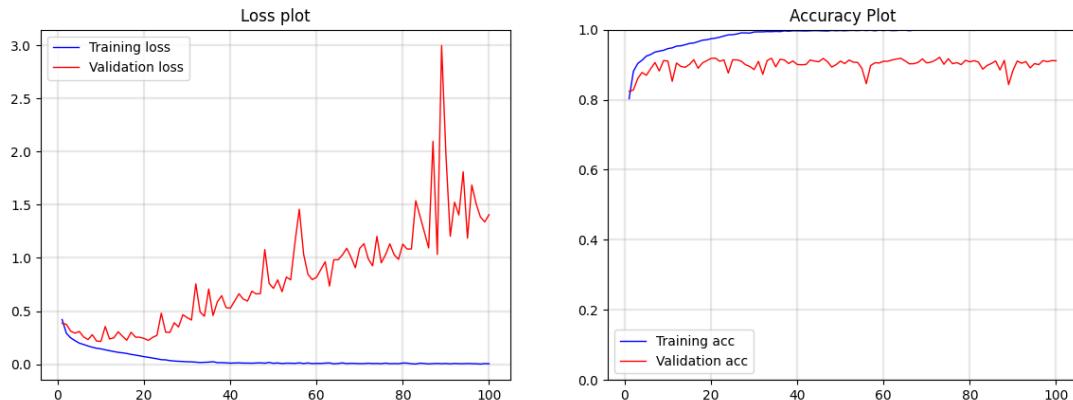
El conjunto de validación termina con un 89,23% de acierto, aunque sufre un fuerte desbalanceo en las clases, solo el 42,57% de las predicciones son cartoon. Este comportamiento se ve de nuevo en la matriz de test, pese a ello, llega a un 90,8% de acierto. La media de los últimos 10 epochs es del 89,2%, siendo bastante inferior a los que se había visto en al prueba 0031 y 0037. Hay algún epoch en el que el desbalanceo

es enorme, se llega a acertar el 96 en cartoon y 72% en real. Detectar estos casos es importante, no solo vale que el promedio tenga un acierto alto, si no que tiene que hacerlo en ambas clases.

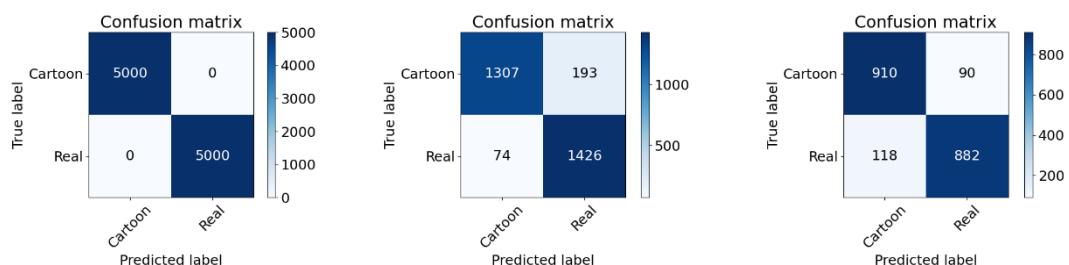
### 0063-CvR-CMCMCMCMD-0039

Imgs. train	Imgs. val.	Imgs. test		
10000 (5000+5000)	3000 (1500+1500)	2000 (1000+1000)		
Optimizador	F. pérdida	Tam. Imgs.	Batch_size	Epochs
RMSProp( $lr=1e-4$ )	Binary_crossentropy	256 x 256	20	100
<b>Estructura</b>				
C(32, (3,3)) MP C(32, (3,3)) MP C(64, (3,3)) MP C(64, (3,3)) MP C(128, (3,3)) MP C(128, (3,3)) MP F D(512) D(1) Convolucionales y Densas con ReLU, salida con Sigmoid				

Por último, se ha probado con un número de filtros ascendente conforme aumentan las capas. Las dos primeras tienen 32 filtros, las dos siguientes 64 y las dos últimos 128.



El loss tiene un rizado algo grande al finalizar el entrenamiento, con unos picos que, aunque son extraños, también se han dado en las demás pruebas. El acierto de validación también sigue siendo muy parecido y no hay nada destacable.



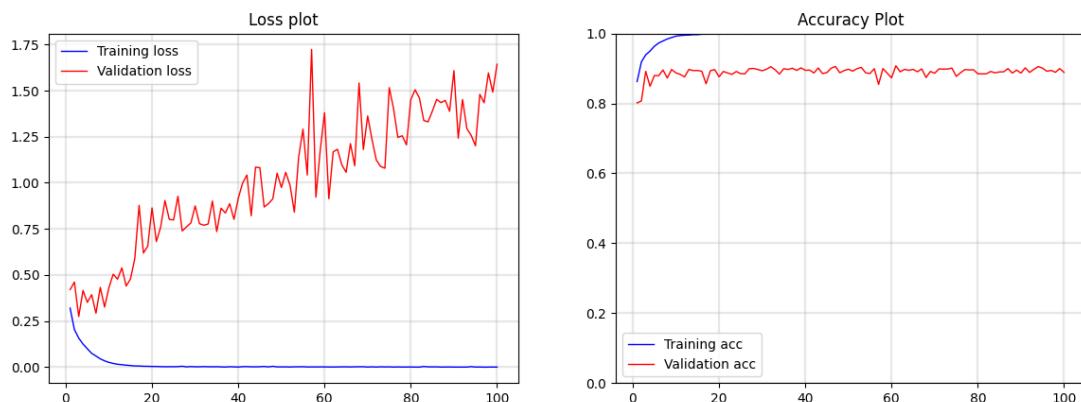
La matriz de validación muestra un 91,1% de acierto, siendo bastante alto y se repite a lo largo de todo el proceso. Sin embargo, el conjunto de test ha tenido un rendimiento peor, termina con un 89,6% y la media de los últimos 10 epochs es del 89,4%, donde vuelve a ser inferior. Como se han incluido capas de 128 puede que esto haya sido lo que ha perjudicado con un exceso de filtros como se ha mostrado en la prueba 0038.

Después de la experimentación realizada, el número de filtros óptimo es de 32 o 64. No obstante, se van a usar 32 filtros ya que tiene un número de parámetros muy inferior, de 700.000 con 64 pasan a 300.000 con 32, siendo la mitad y por tanto teniendo un peso inferior. Es curioso que tener demasiados filtros empeore el resultado porque se podría pensar que cuantos más filtros más características se podrían sacar. Sin embargo, puede que desorienta a la red a la hora de entrenar, haciendo que empeoren ligeramente los resultados.

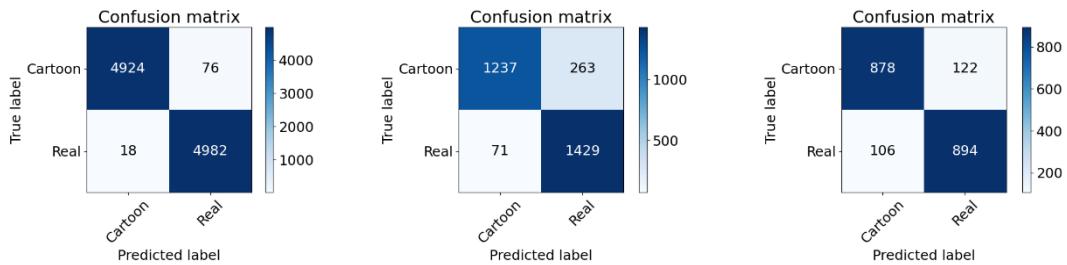
#### 0064-CvR-CMCMCMCMD-0040

Imgs. train	Imgs. val.	Imgs. test		
10000 (5000+5000)	3000 (1500+1500)	2000 (1000+1000)		
Optimizador	F. pérdida	Tam. Imgs.	Batch_size	Epochs
RMSProp(lr=1e-4)	Binary_crossentropy	256 x 256	20	100
<b>Estructura</b>				
C(32, (3,3)) B MP B C(32, (3,3)) B MP B C(64, (3,3)) B MP B C(64, (3,3)) B MP B C(128, (3,3)) B MP B C(128, (3,3)) B MP F D(512) D(1) Convolucionales y Densas con ReLU, salida con Sigmoid				

En esta prueba se incluyen capas de BatchNormalization después de cada capa convolucional y maxpooling. Estas capas normalizan los datos de entrada por lo que podría tener un efecto regularizador.



Es curioso lo que acelera el entrenamiento, el objetivo de las técnicas regularizadoras debería ser el contrario, ya que hace más difícil que se ajuste al conjunto de entrenamiento. Se puede observar que la curva de validación del acierto es más suave, teniendo un rizado menos pronunciado, pero tiene una tasa de acierto menor que en otros casos.



Se finaliza con un 88,87% de acierto en validación casi en ningún epoch ha pasado del 90%. Lo mismo ocurre en la matriz de test, que termina en 88,6% y la media de los últimos 10 epochs es de un 90%. Es curioso que el conjunto de test que tiene imágenes no vistas tengan mejor acierto que el conjunto de validación que son imágenes de las series del conjunto de entrenamiento.

El hecho de que no haya mejorado se puede deber a muchos factores, aunque algunos pueden ser:

- Para este conjunto no aporta nada relevante.
- Los píxeles ya entran normalizados en la red.
- Se utiliza la función ReLU, que a diferencia de la Sigmoide no debería saturarse,

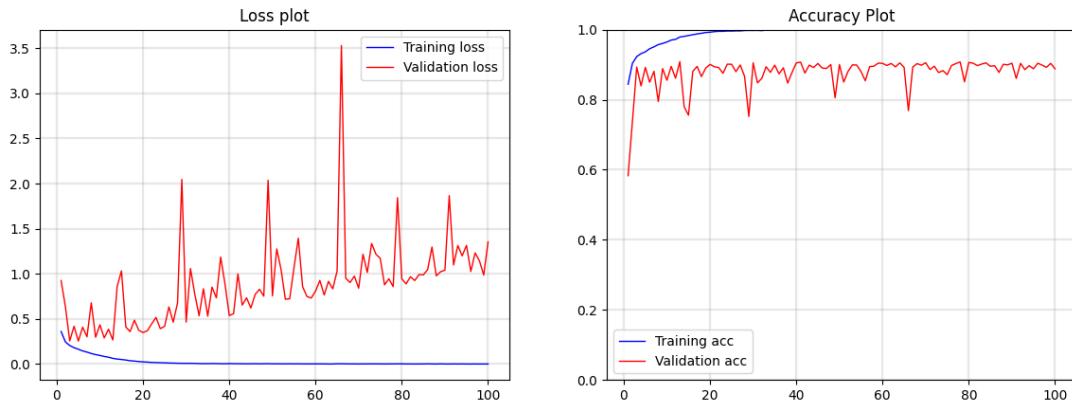
No obstante, experimentar con este tipo de capas era interesante y quizás podría aportar algo. Aunque no lo haga, se pueden sacar conclusiones de posibles motivos de por qué no lo hace.

## 0065-CvR-CMCMCMCMD-0041

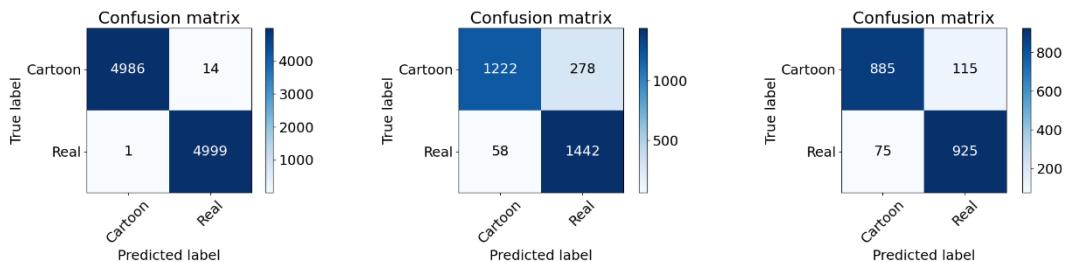
Imgs. train	Imgs. val.		Imgs. test		
10000 (5000+5000)	3000 (1500+1500)		2000 (1000+1000)		
Optimizador	F. pérdida	Tam. Imgs.	Batch_size	Epochs	
RMSProp(lr=1e-4)	Binary_crossentropy	256 x 256	20	100	
Estructura					
C(32, (3,3)) MP C(32, (3,3)) MP C(64, (3,3)) MP C(64, (3,3)) MP C(128, (3,3)) MP C(128, (3,3)) MP B F D(512) D(1)					

## Convolucionales y Densas con ReLU, salida con Sigmoid

Pese a que en la anterior prueba no se ha conseguido mejorar el resultado, se pretende intentar añadir solo una capa de BatchNormalization al final de la parte convolucional o principio del clasificador. Esto se intenta porque al clasificador pueden entrar valores muy altos y quizás esto permita reducir este comportamiento.



El loss muestra algún pico más alto que de costumbre, al igual que el acierto, donde la curva de validación tiene algunos picos algo anómalos.



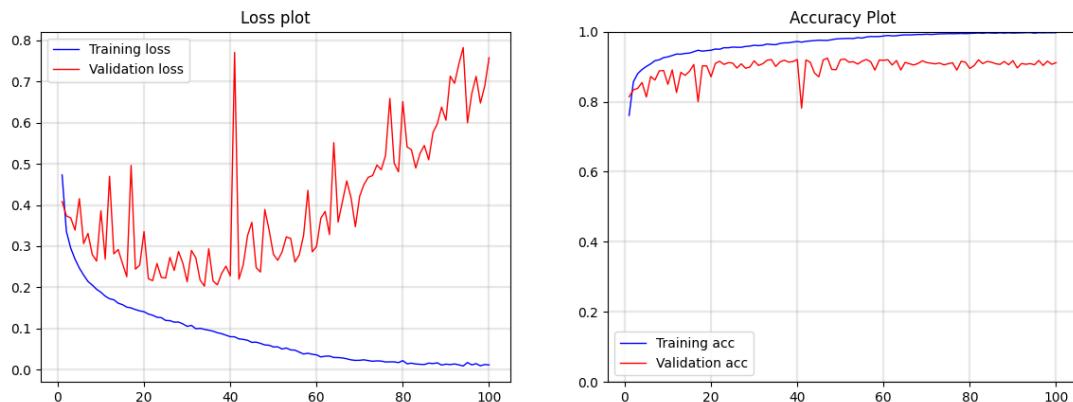
El entrenamiento finaliza con un 88,8% de acierto para validación, aunque se ha llegado en alguna ocasión al 90%. En test se termina con un 90,5% aunque las clases están algo desbalanceadas. La media de los últimos 10 epochs es del 89,77%, por lo que vuelve a no ser tan alta como en el experimento 0031 que estaba en un 90,4%. Parece que la normalización de los datos entre la parte convolucional y el clasificador tampoco ha dado ningún resultado. De hecho, es curioso que parezca que haya afectado más al conjunto de validación que al de test.

### 0066-CvR-CMCMCMCMD-0042

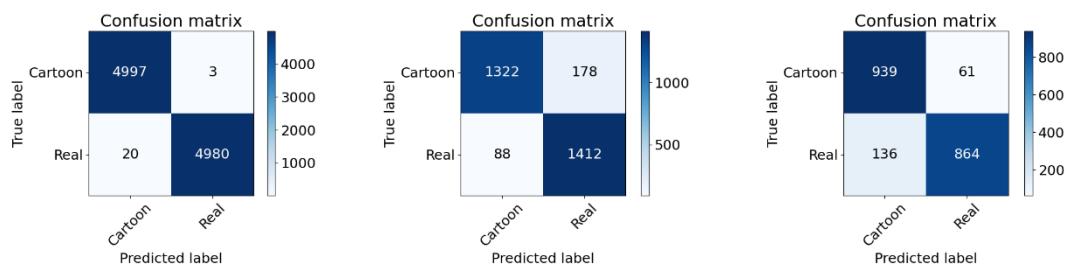
Imgs. train	Imgs. val.	Imgs. test
10000 (5000+5000)	3000 (1500+1500)	2000 (1000+1000)

Optimizador	F. pérdida	Tam. Imgs.	Batch_size	Epochs
RMSProp(lr=1e-4)	Binary_crossentropy	256 x 256	20	100
<b>Estructura</b>				
C(32, (3,3)) MP C(32, (3,3)) MP C(64, (3,3)) MP C(64, (3,3)) MP C(128, (3,3)) MP C(128, (3,3)) MP B F D(256) D(1)				
Convolucionales y Densas con ReLU, salida con Sigmoid				

Los siguientes experimentos servirán para determinar el clasificador que hay después de la parte convolucional. En esta primera prueba se comenzará con reducir el número de neuronas de la capa densa a 256.



El loss tiene valores bastante bajos, muy parecidos a los de la prueba 0031. Lo mismo ocurre con la gráfica del acierto, por lo que puede que el rendimiento no varíe mucho entre 512 y 256 neuronas.

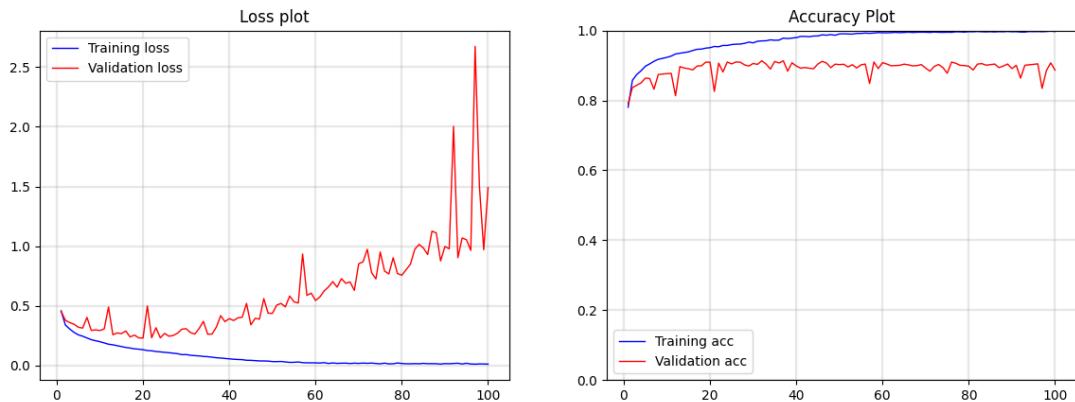


Finaliza con un 91,13% de acierto en validación y 90,15% en test, teniendo las clases algo desbalanceadas. La media de los últimos 10 epochs de test es del 90,15% también, por tanto, es un valor muy alto y cercano al 90,4% de la prueba 0031. Parece que apenas empeora el resultado reducir el número de neuronas, aunque sea algo más bajo tampoco es que se note mucho. Además, si se lanzasen varias veces podría verse alguna diferencia.

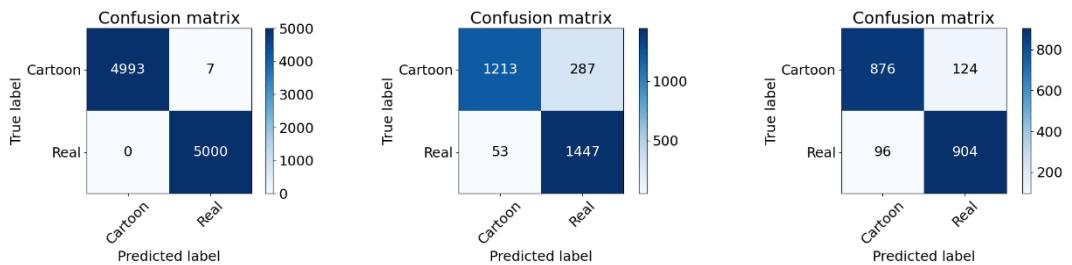
## 0067-CvR-CMCMCMCMD-0043

Imgs. train	Imgs. val.	Imgs. test		
10000 (5000+5000)	3000 (1500+1500)	2000 (1000+1000)		
Optimizador	F. pérdida	Tam. Imgs.	Batch_size	Epochs
RMSProp(lr=1e-4)	Binary_crossentropy	256 x 256	20	100
<b>Estructura</b>				
C(32, (3,3)) MP C(32, (3,3)) MP C(64, (3,3)) MP C(64, (3,3)) MP C(128, (3,3)) MP C(128, (3,3)) MP B F D(1024) D(1) Convolucionales y Densas con ReLU, salida con Sigmoid				

En este caso se duplicarán el número de neuronas con respecto a la prueba 0031, se usarán 1024 en una sola capa.



A primera vista parece que el loss ha subido algo y que el acierto en validación ha disminuido ligeramente.



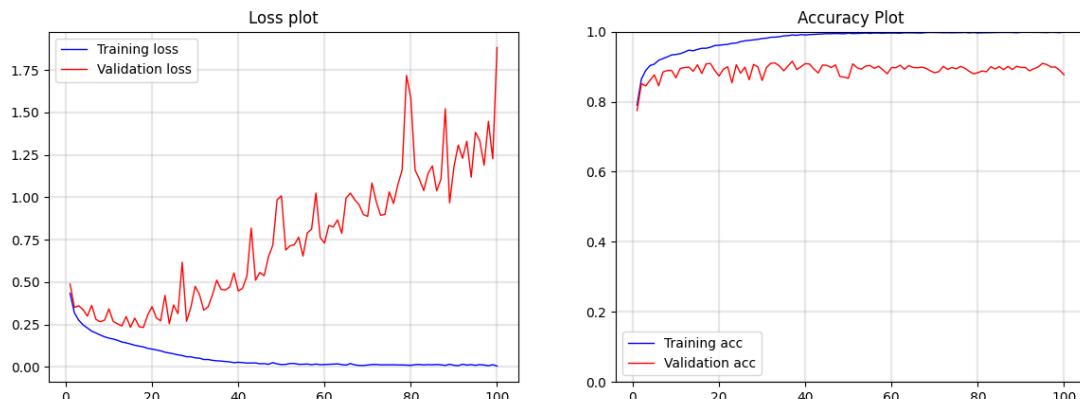
En las matrices de confusión sí que se aprecia el cambio, donde ha bajado algo el acierto. Para validación finaliza con un 88,67%, teniendo un sesgo importante entre las clases y para test termina con un 89%. La media de los últimos 10 epochs es del 88,43%, apreciándose una bajada importante con respecto al 90,4% del experimento 0031.

Definitivamente añadir neuronas ha empeorado el rendimiento, por lo que debe de haber demasiadas neuronas para las características que se extraen. Tener neuronas de más produce que se ajuste más al conjunto de entrenamiento y generalice de peor forma porque saca demasiados rasgos específicos. Además, tener excesivas neuronas puede desorientar al modelo al ser muy complejo.

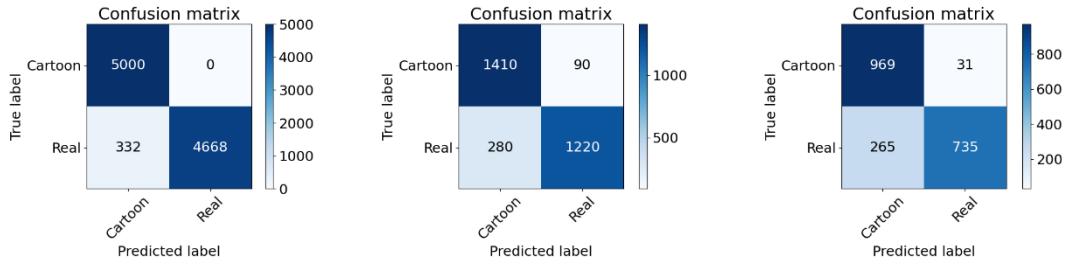
#### 0068-CvR-CMCMCMCMD-0044

Imgs. train	Imgs. val.	Imgs. test		
10000 (5000+5000)	3000 (1500+1500)	2000 (1000+1000)		
Optimizador	F. pérdida	Tam. Imgs.	Batch_size	Epochs
RMSProp(lr=1e-4)	Binary_crossentropy	256 x 256	20	100
Estructura				
C(32, (3,3)) MP C(32, (3,3)) MP C(64, (3,3)) MP C(64, (3,3)) MP C(128, (3,3)) MP C(128, (3,3)) MP B F D(512) D(256) D(1) Convolucionales y Densas con ReLU, salida con Sigmoid				

Ya que con 512 es el mejor resultado que se tiene, al igual que con 256 que era muy parecido, se va a probar con dos capas en el clasificador, la primera de 512 y la segunda de 256.



El loss parece algo más alto que la prueba 0031 pero el acierto parece algo más bajo. La cantidad de parámetros es mayor por lo que puede estar ocurriendo lo mismo que pasaba con una capa y 1024 neuronas.

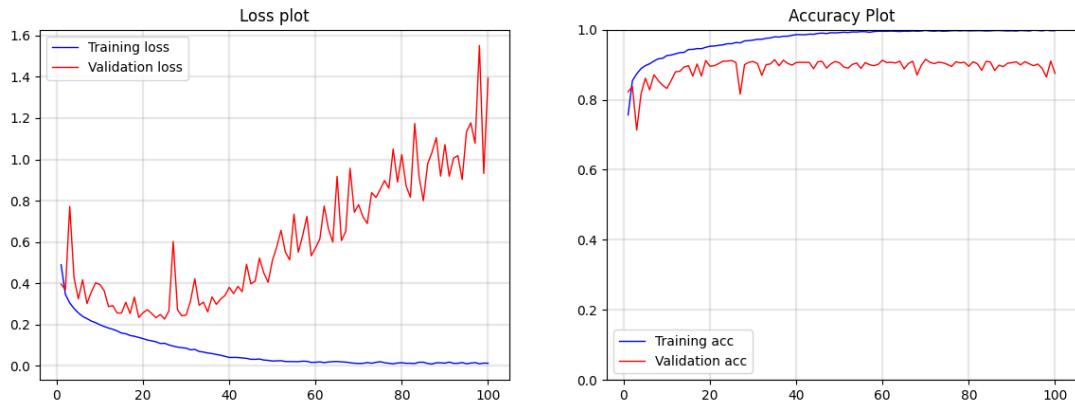


En validación se acaba 87,67% y durante el entrenamiento, casi ningún epoch pasa del 90%. En test hay alguna anomalía, ya que las clases están totalmente desbalanceadas, se predice un 61,7% de cartoon y un 38,3% de reales. El acierto en test es del 85,2% y la media de los últimos 10 epochs es del 88,44%, inferior a 512 con una capa que era del 90,4%. De hecho, el acierto es similar al caso anterior de las 1024 neuronas, por lo que sigue habiendo un exceso de neuronas. Puede que para este problema dos capas sean demasiado, pero se hará una última prueba para comprobarlo.

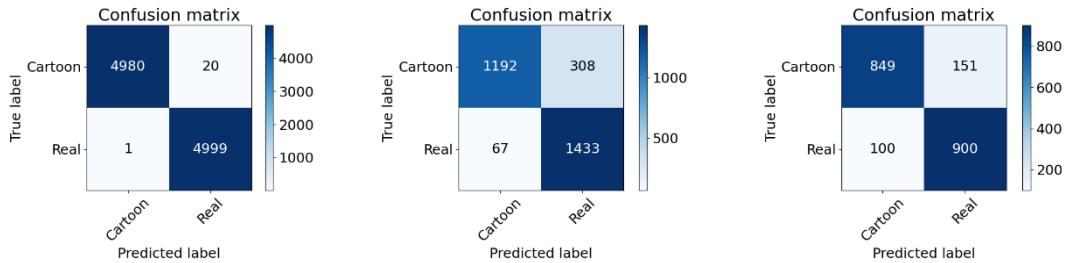
## 0069-CvR-CMCMCMCMD-0045

Imgs. train	Imgs. val.		Imgs. test	
10000 (5000+5000)	3000 (1500+1500)		2000 (1000+1000)	
Optimizador	F. pérdida	Tam. Imgs.	Batch_size	Epochs
RMSProp(lr=1e-4)	Binary_crossentropy	256 x 256	20	100
<b>Estructura</b>				
C(32, (3,3)) MP C(32, (3,3)) MP C(64, (3,3)) MP C(64, (3,3)) MP C(128, (3,3)) MP C(128, (3,3)) MP B F D(256) D(128) D(1) Convolucionales y Densas con ReLU, salida con Sigmoid				

La siguiente prueba volverá a tener dos capas, esta vez con menos neuronas a ver si puede mejorar algo a la anterior prueba. Habrá 256 y 128 en cada capa.



Ambas gráficas son parecidas a la prueba anterior, no parece que haya muchas diferencias a la hora de reducir neuronas usando dos capas.



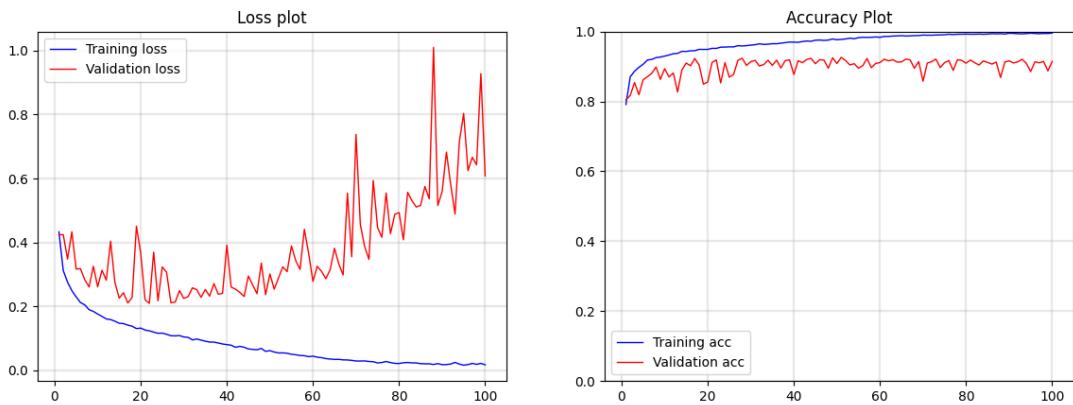
En la matriz de validación se llega a un 87,5% de acierto, aunque las clases están bastante desbalanceadas. Esta vez durante el entrenamiento sí que se supera en numerosas ocasiones el 90%, por lo que puede ser algo mejor la prueba que la 0044. En test se tiene un 87.45% de acierto y en los últimos 10 epochs se tiene un 87,22%. Es un acierto inferior al experimento anterior, pero en el log se puede ver algo extraño ya que en algún epoch el acierto desciende hasta el 80%, estando lejos de la media. Por tanto, puede que sin contar esos epochs se tuviera un acierto mejor, aunque para ser justos se han seguido las mismas reglas para todos los experimentos.

## 0070-CvR-CMCMCMCMD-0046

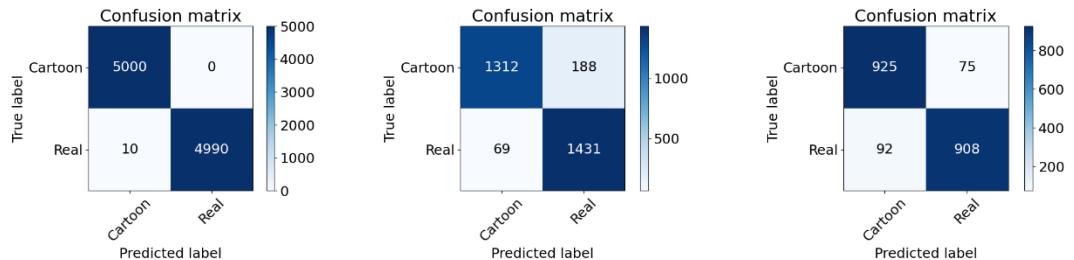
Imgs. train	Imgs. val.		Imgs. test	
10000 (5000+5000)	3000 (1500+1500)		2000 (1000+1000)	
Optimizador	F. pérdida	Tam. Imgs.	Batch_size	Epochs
RMSProp(lr=1e-4)	Binary_crossentropy	256 x 256	20	100
<b>Estructura</b>				
C(32, (3,3)) MP C(32, (3,3)) MP C(64, (3,3)) MP C(64, (3,3)) MP C(128, (3,3)) MP C(128, (3,3)) MP B F Dr(0.1) D(512) Dr(0.1) D(1) Convolucionales y Densas con ReLU, salida con Sigmoid				

Este experimento es el primero de los que se van a realizar probando el dropout, el cual suele tener un efecto regularizador bastante bueno. Se comenzará con un dropout leve, del 10%. Esto quiere decir que hay una probabilidad del 10% de que una neurona no salga elegida. Se han añadido dos capas, una después de la Flatten y otra antes de la salida.

Los siguientes experimentos se centrarán en ver el valor de dropout óptimo, ya que subirlo demasiado puede ser contraproducente. También hay que tener en cuenta que el dropout favorece a la generalización, pero empeora el entrenamiento porque le cuesta aprender demasiado de unos ejemplos.



De primeras se puede observar que en la gráfica del loss la curva de validación tarda más en divergir de la de entrenamiento que de costumbre. Además, al estar en valores pequeños, si se hiciese un zoom, todavía estarían más juntas. En entrenamiento se aprecia que se alcanza el 100% de acierto sobre el epoch 85. Para validación se tiene una curva similar al experimento 0031, siendo bastante alta y con un rizado algo contenido.



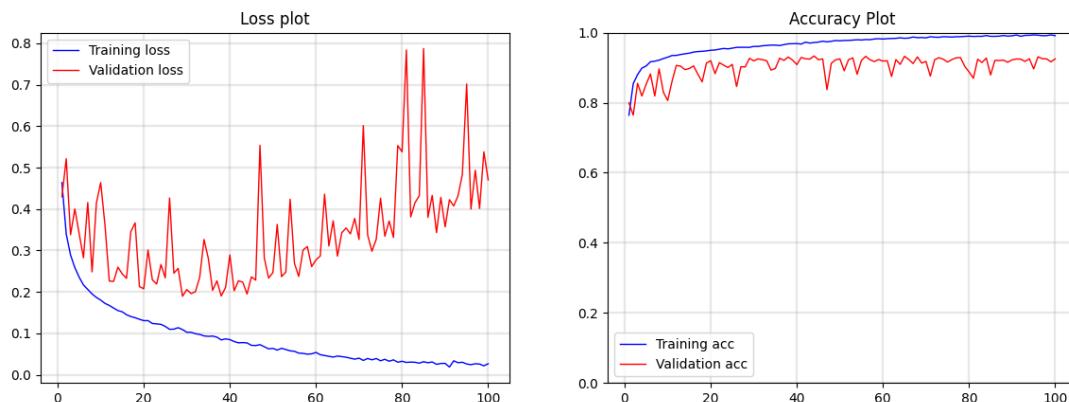
La matriz de confusión de validación muestra un 91,43% de acierto, teniendo las clases ligeramente desbalanceadas. En el conjunto de test se termina con un 91,65% de acierto, con una media de los últimos 10 epochs del 91,15%, siendo el mejor experimento hasta el momento. La prueba 0031 tenía un 90,48% de acierto en los últimos 10 epochs, por lo que el efecto regularizador se nota levemente, algo más de un 0.5% de mejora. Estos valores por encima del 91% ya son bastante altos, dado que se tienen 2000 imágenes para ser evaluadas que nunca se han visto.

## 0071-CvR-CMCMCMCMD-0047

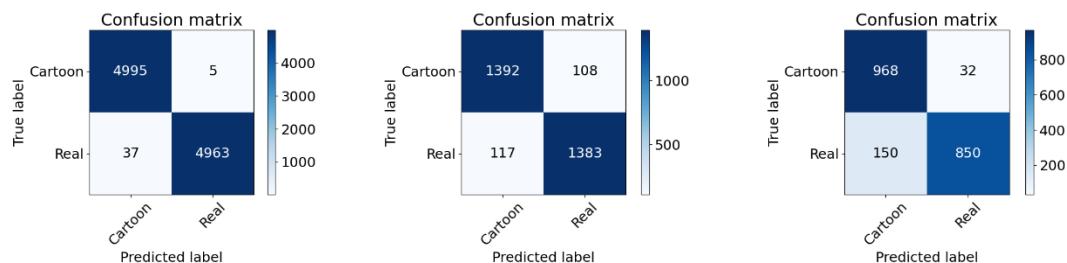
Imgs. train	Imgs. val.	Imgs. test		
10000 (5000+5000)	3000 (1500+1500)	2000 (1000+1000)		
Optimizador	F. pérdida	Tam. Imgs.	Batch_size	Epochs
RMSProp(lr=1e-4)	Binary_crossentropy	256 x 256	20	100
Estructura				

C(32, (3,3)) MP C(32, (3,3)) MP C(64, (3,3)) MP C(64, (3,3)) MP C(128, (3,3)) MP C(128, (3,3)) MP B
F Dr(0.2) D(512) Dr(0.2) D(1)
Convolucionales y Densas con ReLU, salida con Sigmoid

Se sigue con la misma configuración que antes, subiendo el dropout hasta 0.2, por lo que hay una probabilidad del 20% de la que una neurona no se tenga en cuenta.



Parece que se ha conseguido bajar aún más el error en validación. Parece que algunos epochs más no habrían estado mal, porque no se llega al 0 y en acierto está al 99% en entrenamiento. La curva de validación ha subido y puede que siga subiendo.



Es destacable que en la matriz de validación las clases están muy bien balanceadas, llegando a un acierto del 92,5%, cifra a la que se ha llegado en numerosas ocasiones durante el entrenamiento. Sin embargo, la matriz de test es mucho peor, es curioso porque la de validación es de las mejores. Las clases en test están muy desbalanceadas, pese a que el acierto es del 90,9%, donde hay muchas más predicciones hacia cartoon lo que eleva la media del acierto total. En los últimos 10 epochs se ha obtenido un 90,42%, aunque es una bajada en un epoch al 84% que baja la media drásticamente.

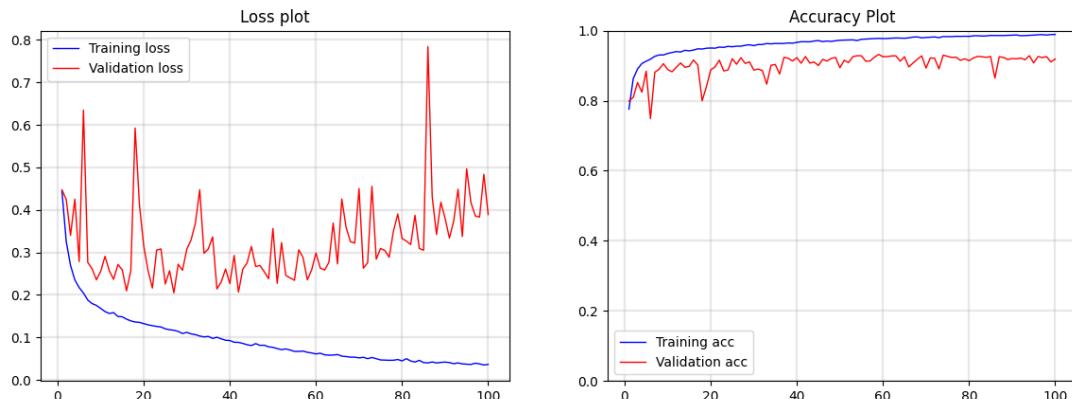
Debido a que el lanzamiento es algo confuso, se ha vuelto a correr el programa. En este caso, se obtiene de media en los últimos 10 epochs un 90,35% por lo que no ha sido

puntual en el caso anterior, ya que se ha repetido. Puede que con más epoch este resultado mejorase, aunque se seguirán probando porcentajes más altos de dropout.

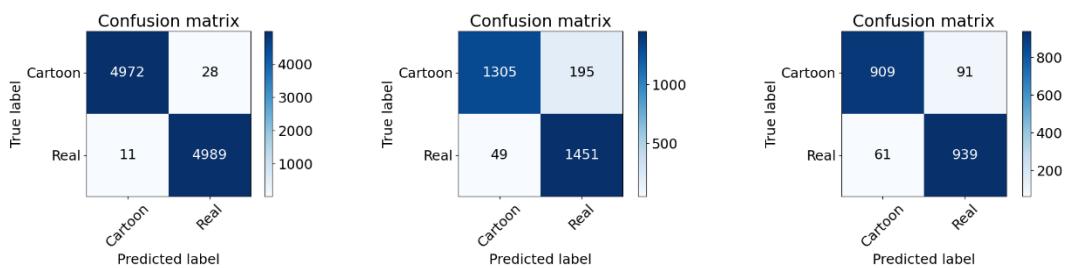
## 0072-CvR-CMCMCMCMD-0048

Imgs. train	Imgs. val.	Imgs. test		
10000 (5000+5000)	3000 (1500+1500)	2000 (1000+1000)		
Optimizador	F. pérdida	Tam. Imgs.	Batch_size	Epochs
RMSProp(lr=1e-4)	Binary_crossentropy	256 x 256	20	100
<b>Estructura</b>				
C(32, (3,3)) MP C(32, (3,3)) MP C(64, (3,3)) MP C(64, (3,3)) MP C(128, (3,3)) MP C(128, (3,3)) MP B F Dr(0.3) D(512) Dr(0.3) D(1) Convolucionales y Densas con ReLU, salida con Sigmoid				

Vuelve a probarse el dropout, en este caso con el 30% de probabilidad de ser eliminada una neurona.



Se aprecia que el error ha vuelto a bajar algo, convergiendo aún más al entrenamiento. El acierto está sobre el 99% para entrenamiento, no se ha llegado a converger aún del todo, aunque parece que para validación la curva está estancada.



La matriz de validación muestra un 91,87% de acierto, aunque en muchas ocasiones ha superado el 92%, incluso el 93%. En test se puede ver uno de los mayores aumento hasta

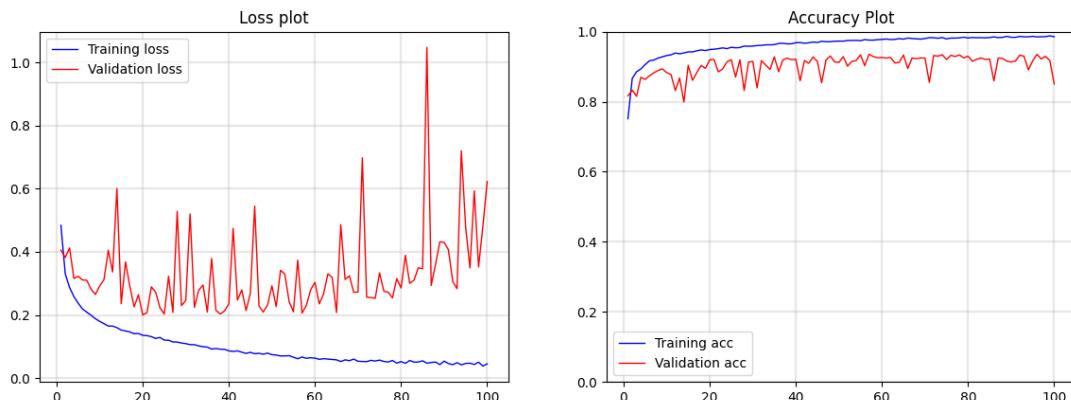
la fecha, un 92,4%. La media de los últimos 10 epochs es del 92,44%, diferenciándose del anterior mejor, la prueba 0046, un 0,8%. El epoch 98 ha llegado a superar el 94% de acierto. Lo que es más importante es que se puede revisar en el log que todos los epochs están balanceados y no se detecta ninguna anomalía como en otros casos.

Es curioso que con un 20% de dropout ha dado peor resultado que con un 30%. Puede que se haya dado un caso atípico en el que las neuronas desechadas hayan sido peores en el 0047 que en el 0048. De todas formas, en algún punto el dropout tiene que perjudicar notablemente, por lo que se seguirá probando con valores más altos.

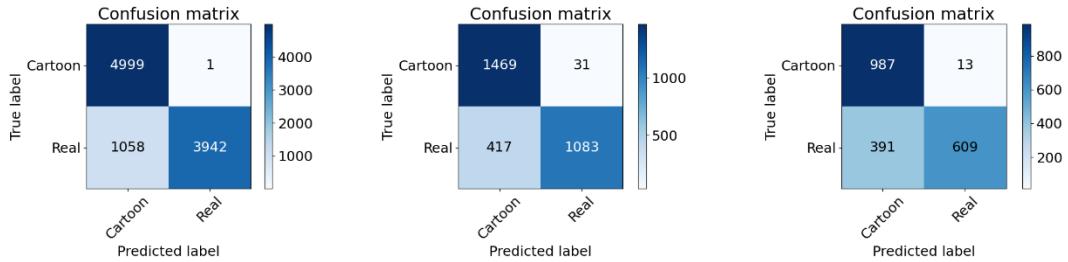
### 0073-CvR-CMCMCMCMD-0049

Imgs. train	Imgs. val.		Imgs. test	
10000 (5000+5000)	3000 (1500+1500)		2000 (1000+1000)	
Optimizador	F. pérdida	Tam. Imgs.	Batch_size	Epochs
RMSProp( $lr=1e-4$ )	Binary_crossentropy	256 x 256	20	100
<b>Estructura</b>				
C(32, (3,3)) MP C(32, (3,3)) MP C(64, (3,3)) MP C(64, (3,3)) MP C(128, (3,3)) MP C(128, (3,3)) MP B F Dr(0.4) D(512) Dr(0.4) D(1) Convolucionales y Densas con ReLU, salida con Sigmoid				

En esta prueba se aumenta aún más la probabilidad de dropout, llegando a un 40%.



El error de validación sigue bajando, acercándose cada vez más a la curva del entrenamiento. En el acierto se ve que se tiene un acierto parecido a los que se ha visto antes, aunque da la coincidencia de que termina con un pico, por lo que los resultados pueden no ser veraces al 100%.

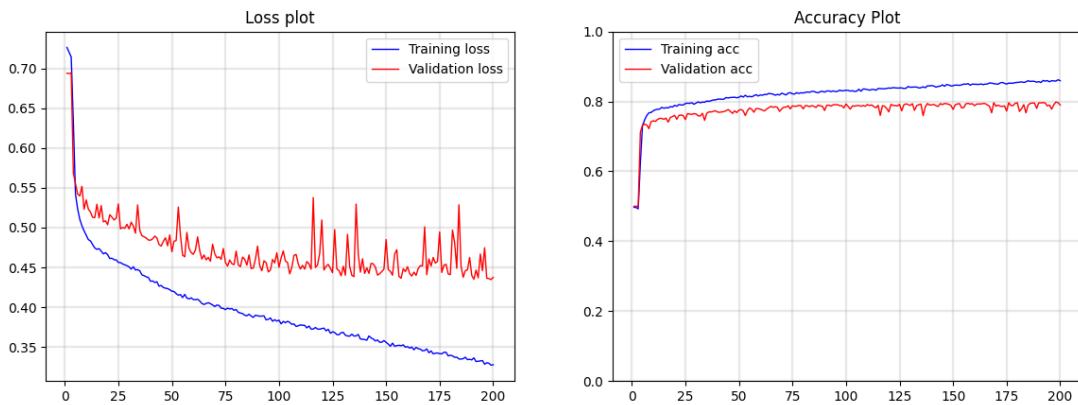


Destaca el desbalanceo que se produce en las tres matrices de confusión, es probablemente el más alto visto hasta la fecha. Como se ha dicho antes, esto es puntual porque ha habido un pico en el rizado de la curva. Solo se va a tener en cuenta la media de los 10 últimos epochs, excluyendo el último porque no aporta información real. La media se sitúa en un 91,56%, donde se puede ver que ha bajado ligeramente con respecto a la prueba anterior, casi un 1%. Parece que este es el punto en el que el dropout empieza a empeorar el rendimiento de todos los conjuntos, por lo que no se realizarán más pruebas sobre esta técnica regularizadora.

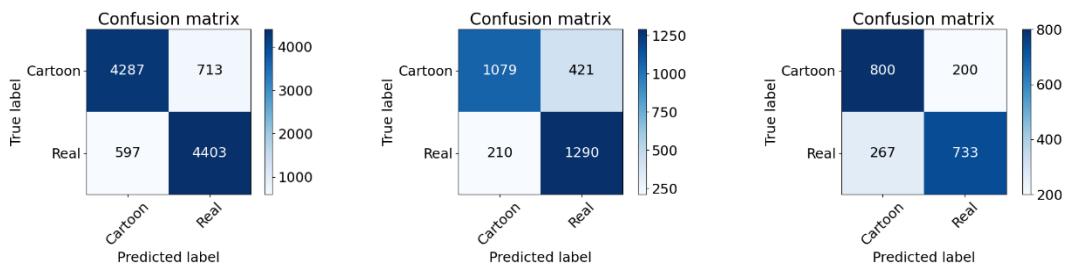
## 0074-CvR-CMCMCMCMD-0050

Imgs. train	Imgs. val.		Imgs. test	
10000 (5000+5000)	3000 (1500+1500)		2000 (1000+1000)	
Optimizador	F. pérdida	Tam. Imgs.	Batch_size	Epochs
RMSProp(lr=1e-4)	Binary_crossentropy	256 x 256	20	200
Estructura				
C(32, (3,3)) MP C(32, (3,3)) MP C(64, (3,3)) MP C(64, (3,3)) MP C(128, (3,3)) MP C(128, (3,3)) MP B F Dr(0.3) D(512) Dr(0.3) D(1) Convolucionales y Densas con Sigmoid, salida con Sigmoid				

Esta prueba da paso a una serie de experimentos en los que se pretende evaluar el tipo de función de activación usada. En los casos anteriores siempre ha sido la ReLU, por lo que se va a comprobar si es verdad que esta función es de las mejores que hay. La arquitectura usada será la de la prueba 0048, que hasta ahora tiene los mejores resultados. Las prueba se ha realizado con 200 epochs porque se había detectado que 100 no eran suficientes para esta función de activación.



Al tener más epochs puede parecer que el rizado es más pequeño porque la gráfica está más comprimida, aunque los picos son iguales que en otras ocasiones. La gráfica de entrenamiento indica que puede haber una posible mejora si se hubiesen dado más epochs. La gráfica tiene un pendiente muy poco pronunciada, se necesitarías muchísimos epochs para comprobar este experimento. En caso de que haya tiempo, se lanzará este experimento con más epochs. No obstante, la curva de validación se ve que diverge cada vez más de la de entrenamiento, y parece que empieza a ser bastante plana.

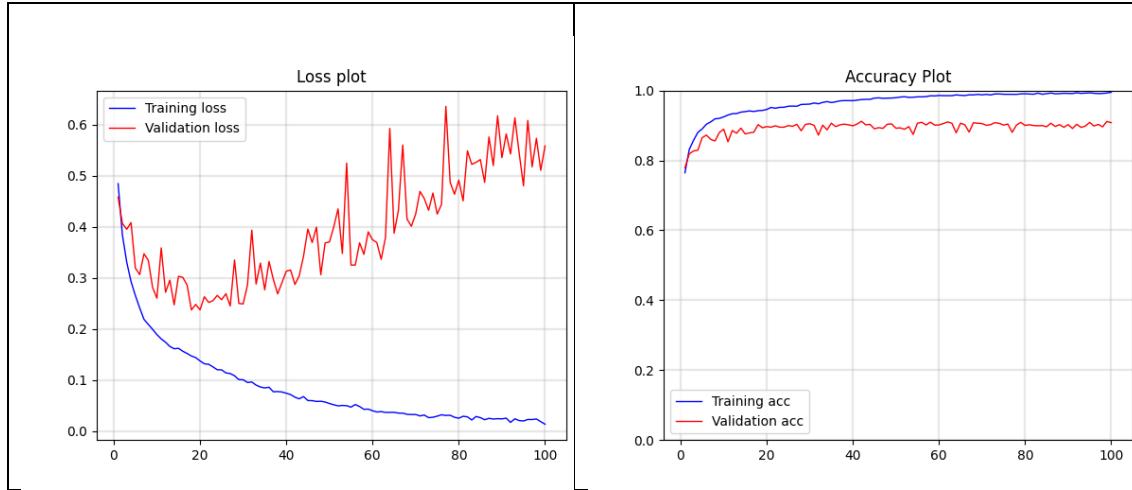


Como se ha comentado antes, no merece la pena dar cifras sobre este experimento ya que el entrenamiento tiene más posibilidades y se sitúa en porcentajes muy bajos para el resto de conjunto.

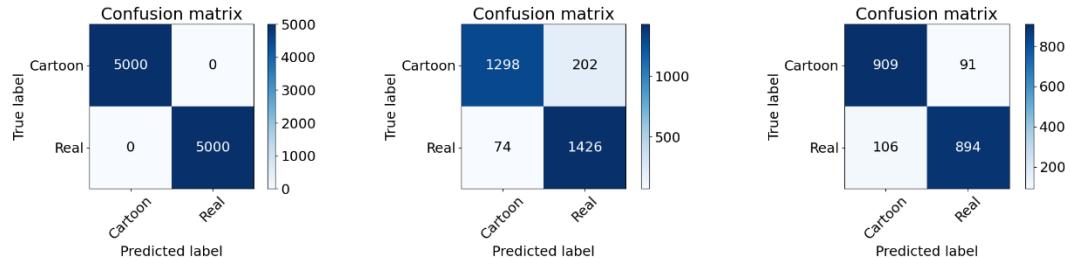
## 0075-CvR-CMCMCMCMD-0051

Imgs. train	Imgs. val.		Imgs. test	
10000 (5000+5000)	3000 (1500+1500)		2000 (1000+1000)	
Optimizador	F. pérdida	Tam. Imgs.	Batch_size	Epochs
RMSProp(lr=1e-4)	Binary_crossentropy	256 x 256	20	100
<b>Estructura</b>				
C(32, (3,3)) MP C(32, (3,3)) MP C(64, (3,3)) MP C(64, (3,3)) MP C(128, (3,3)) MP C(128, (3,3)) MP B F Dr(0.3) D(512) Dr(0.3) D(1) Convolucionales y Densas con TanH, salida con Sigmoid				

En este experimento se usa la función de activación TanH en todas las capas excepto la de salida, que obviamente es sigmoide.



Parece que el loss ha subido, pero ha bajado tanto el error que un mínimo cambio parece que hace que la gráfica suba, por lo que es un efecto visual. En el acierto parece que la curva de validación no ha terminado de converger, donde más epochs podrán arrojar un mejor acierto. Sin embargo, la curva en general parece más baja que en otras ocasiones.



Validación se termina con un 90,8% y con algo de desbalanceo. La matriz de test muestra un 90,15% y la media de los últimos 10 epochs es del 90,46%. Si bien no está mal, está lejos del actual mejor resultado, que tiene un 2% más de acierto.

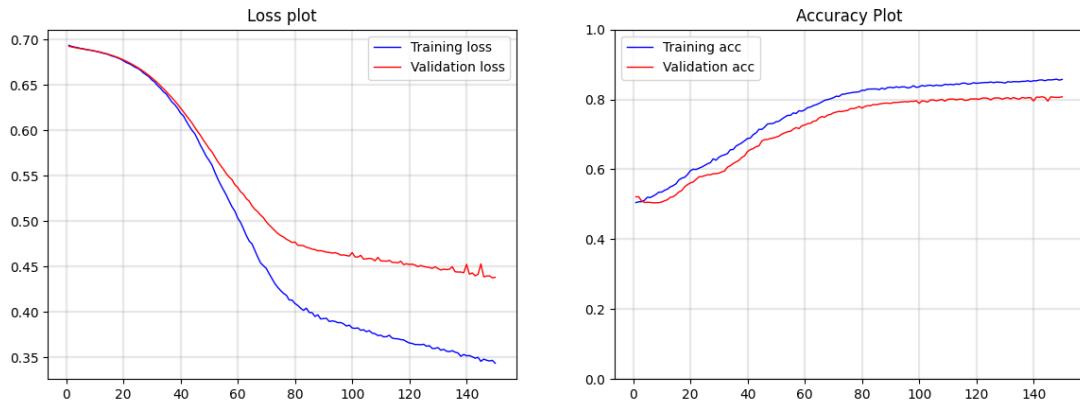
Con estas pruebas se puede ver que la mejor función para la arquitectura usada en este problema es la ReLU. La función sigmoide tiene alguna peculiaridades y problemas, como la saturación, aunque el conjunto está normalizado para intentar evitarlo. Sin embargo, la ReLU evita esos problemas, por eso debería ser mejor. TanH no tiene un mal rendimiento, pero está lejos de posicionarse como primera opción.

## 0076-CvR-CMCMCMCMD-0052

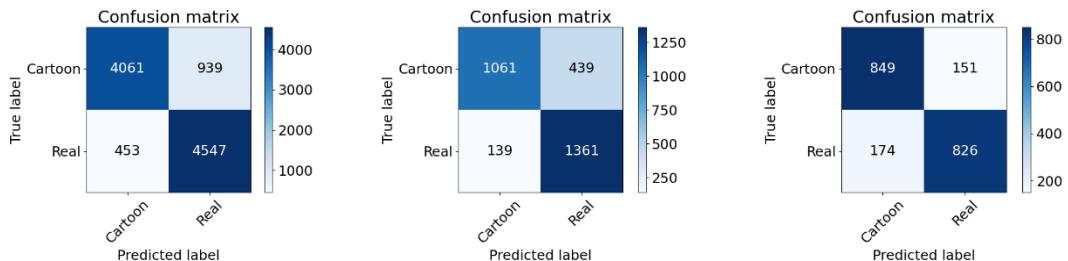
_imgs. train	_imgs. val.	_imgs. test
--------------	-------------	-------------

10000 (5000+5000)	3000 (1500+1500)	2000 (1000+1000)		
<b>Optimizador</b>	<b>F. pérdida</b>	<b>Tam. Imgs.</b>	<b>Batch_size</b>	<b>Epochs</b>
SGD(lr=1e-4)	Binary_crossentropy	256 x 256	20	150
<b>Estructura</b>				
C(32, (3,3)) MP C(32, (3,3)) MP C(64, (3,3)) MP C(64, (3,3)) MP C(128, (3,3)) MP C(128, (3,3)) MP B F Dr(0.3) D(512) Dr(0.3) D(1) Convolucionales y Densas con ReLU, salida con Sigmoid				

Para esta prueba se utiliza el optimizador SGD.



Aunque puede haber un pequeño margen de mejora, el acierto comienza a converger en valores cercanos al 80%, lo cual es muy bajo.



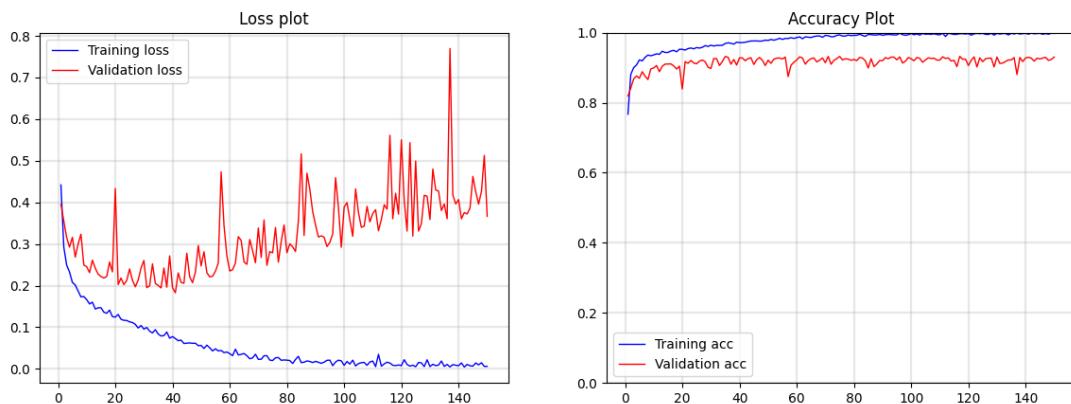
Se observa rápidamente que los resultados no son buenos. Este optimizador es uno de los primeros que hicieron aparición y los otros probado son mejoras aportadas posteriormente, por tanto, es totalmente normal que estos resultados sean tan bajos.

### 0077-CvR-CMCMCMCMD-0053

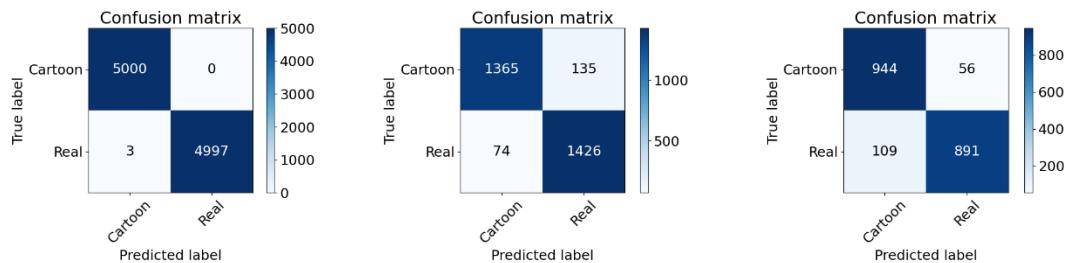
Imgs. train	Imgs. val.	Imgs. test		
10000 (5000+5000)	3000 (1500+1500)	2000 (1000+1000)		
<b>Optimizador</b>	<b>F. pérdida</b>	<b>Tam. Imgs.</b>	<b>Batch_size</b>	<b>Epochs</b>
Adam(lr=1e-4)	Binary_crossentropy	256 x 256	20	150
<b>Estructura</b>				

C(32, (3,3)) MP C(32, (3,3)) MP C(64, (3,3)) MP C(64, (3,3)) MP C(128, (3,3)) MP C(128, (3,3)) MP B F Dr(0.3) D(512) Dr(0.3) D(1) Convolucionales y Densas con ReLU, salida con Sigmoid
---

En esta ocasión se usa como optimizador Adam, manteniendo la misma arquitectura y 150 epochs.



Parece que el error tiene un rizado mayor que en ocasiones anteriores, aunque también se ha entrenado durante más epochs. Puede que la curva de validación muestre que con más epochs se podría llegar más lejos, así que queda pendiente comprobar eso.



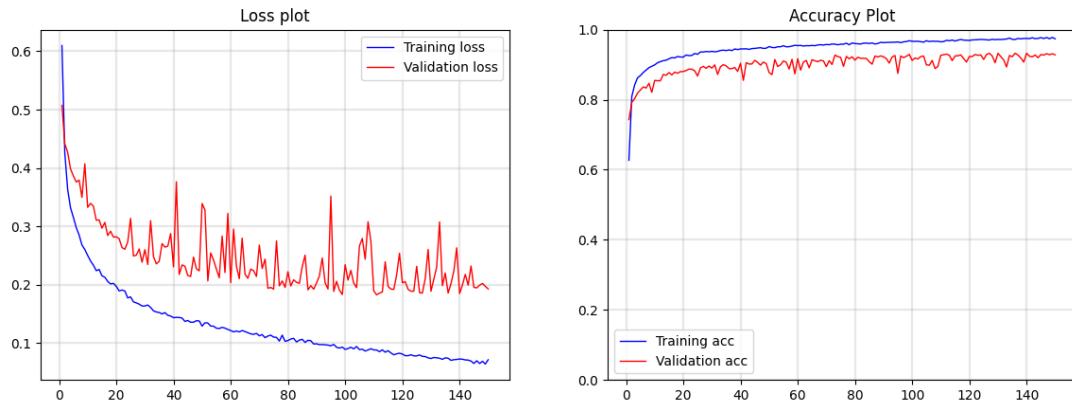
Las clases tienen un ligero desbalanceo tanto en validación como en test, con un 93,03% y 91,75% respectivamente. La media de los 10 últimos epochs para test es del 92,47%, siendo muy parecida a la prueba 0048, aunque parece que en este caso la curva de validación no ha terminado de converger.

#### 0078-CvR-CMCMCMCMD-0054

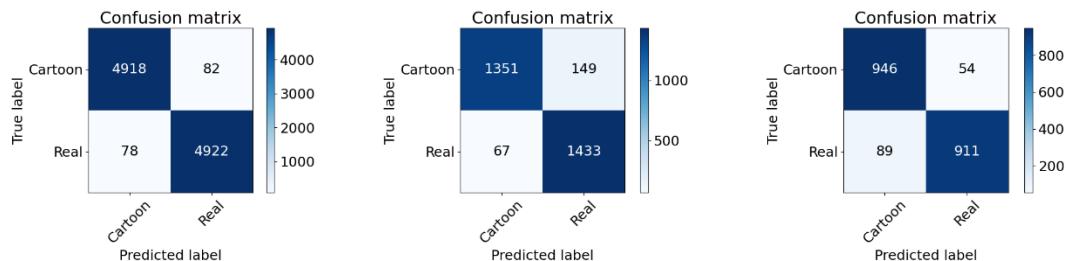
Imgs. train	Imgs. val.	Imgs. test		
10000 (5000+5000)	3000 (1500+1500)	2000 (1000+1000)		
Optimizador	F. pérdida	Tam. Imgs.	Batch_size	Epochs
Adamax(lr=1e-4)	Binary_crossentropy	256 x 256	20	150
<b>Estructura</b>				

C(32, (3,3)) MP C(32, (3,3)) MP C(64, (3,3)) MP C(64, (3,3)) MP C(128, (3,3)) MP C(128, (3,3)) MP B F Dr(0.3) D(512) Dr(0.3) D(1) Convolucionales y Densas con ReLU, salida con Sigmoid
---

En este caso se usa Adamax con 150 epochs.



Lo mismo ocurre que con el caso anterior, solo que ni la curva de entrenamiento ha convergido, sigue teniendo margen ya que no se sabe cómo influirá. Lo que sí es curioso es que el error sigue teniendo una tendencia bajista, algo que no había ocurrido en ningún momento.



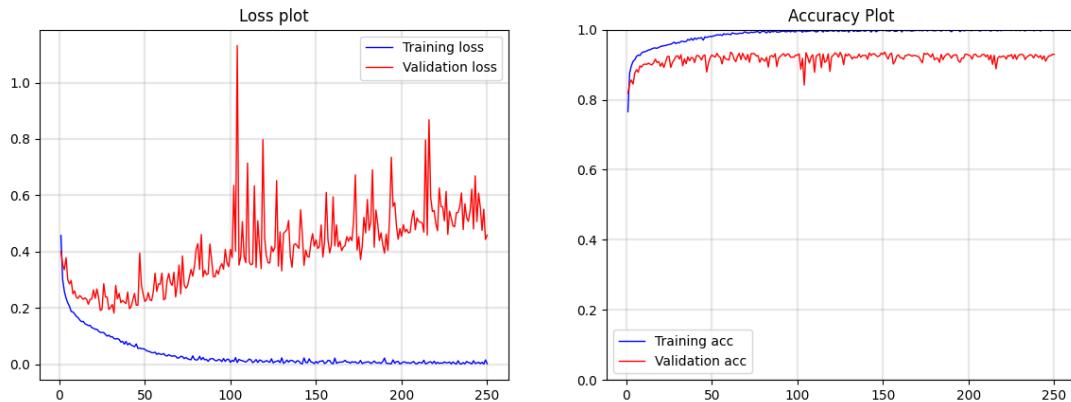
No se va a comentar con mucho detalle porque se relanzará con más epochs. Para validación se tiene un 92,8% de acierto y para test un 92,85%. La media de los últimos 10 epochs para test es del 92,31%. Los dos últimos experimentos se relanzarán.

## 0079-CvR-CMCMCMCMD-0055

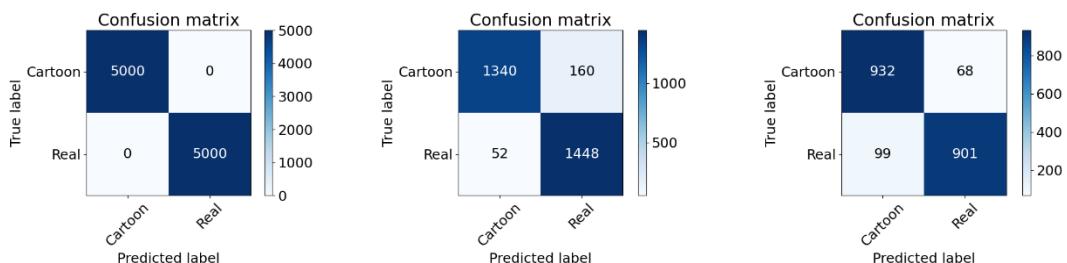
Imgs. train	Imgs. val.		Imgs. test	
10000 (5000+5000)	3000 (1500+1500)		2000 (1000+1000)	
Optimizador	F. pérdida	Tam. Imgs.	Batch_size	Epochs
Adam(lr=1e-4)	Binary_crossentropy	256 x 256	20	250
Estructura				
C(32, (3,3)) MP C(32, (3,3)) MP C(64, (3,3)) MP C(64, (3,3)) MP C(128, (3,3)) MP C(128, (3,3)) MP B F Dr(0.3) D(512) Dr(0.3) D(1)				

### Convolucionales y Densas con ReLU, salida con Sigmoid

Mismo experimento que el 0053 pero se aumentan los epochs hasta 250 para verificar si continua el entrenamiento.



Parece que tanto en el loss como en la acc se puede comprobar que el entrenamiento ha finalizado. No hay signos claros de *overfitting* pero se estanca el acierto.



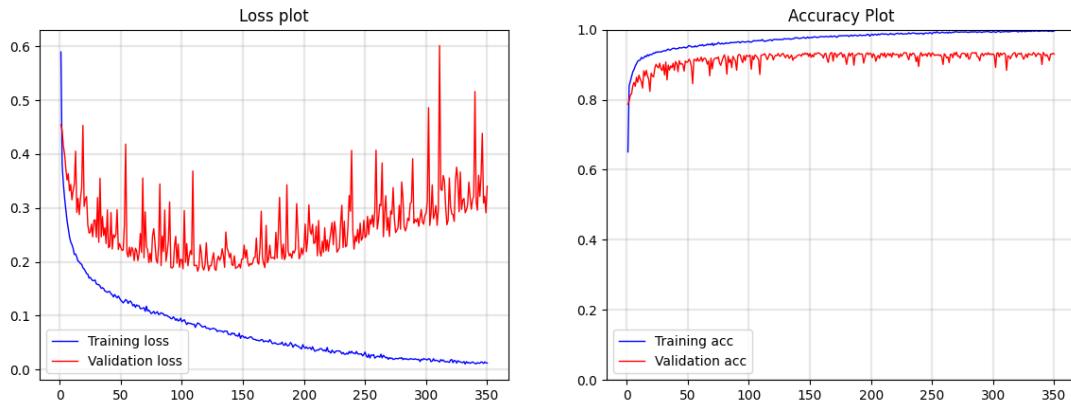
Validación se termina con un 92,93% de acierto mientras que para test se tiene un 91,65% de acierto. Las clases no están balanceadas al 100%, pero tampoco es un desbalanceo acusado. La media de los últimos 10 epochs en test es del 91,85%. Por tanto, pese a que el experimento 0053 parecía no haber terminado, ha bajado el acierto en un 0,6%. Esto es un indicio de sobre ajuste, pero no parece que vaya a obtener muchos mejores resultados que los visto en el 0053 pese a que se modifiquen los epochs.

### **0080-CvR-CMCMCMCMD-0056**

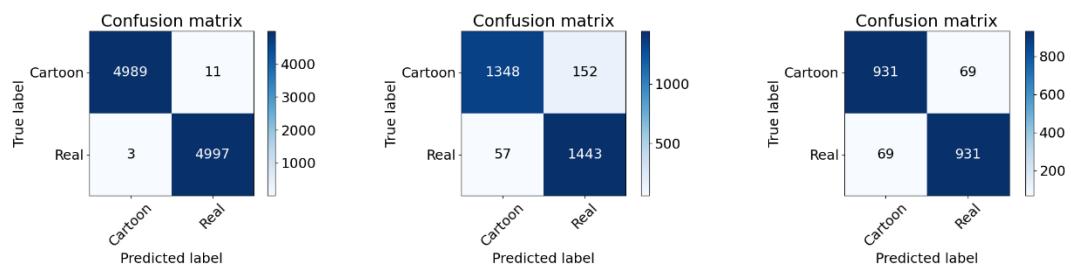
Imgs. train	Imgs. val.		Imgs. test	
10000 (5000+5000)	3000 (1500+1500)		2000 (1000+1000)	
Optimizador	F. pérdida	Tam. Imgs.	Batch_size	Epochs
Adamax(lr=1e-4)	Binary_crossentropy	256 x 256	20	350
Estructura				

C(32, (3,3)) MP C(32, (3,3)) MP C(64, (3,3)) MP C(64, (3,3)) MP C(128, (3,3)) MP C(128, (3,3)) MP B F Dr(0.3) D(512) Dr(0.3) D(1) Convolucionales y Densas con ReLU, salida con Sigmoid
---

Mismo experimento que el 0054 pero con 350 epochs.



La diferencia más clara con respecto a la prueba 0054 es que ahora si ha terminado de converger el entrenamiento. El loss parece que puede bajar más en el entrenamiento, pero eso no importa ya que fomentará el sobre ajuste.



Validación finaliza con un 93,03% de acierto y test termina con un 93,1%. La media de los últimos 10 epochs para test es del 93,25%, siendo el mejor porcentaje de acierto de todos los experimentos realizados. Además, en el log se puede revisar que las clases están bien balanceadas y no hay ninguna anomalía a simple vista.

## Resumen de objetivos

Experimento	Objetivo
<b>Experimentación con el conjunto</b>	
0025-CvR-CMCMCMCMD-0001	Primera prueba con un conjunto pequeño, 2.048 con distribución 1.648-400-60 (entrenamiento, validación, test).
0026-CvR-CMCMCMCMD-0002	Aumento a un conjunto de 7.000 imágenes, 4.500-1.500-1.000.
0027-CvR-CMCMCMCMD-0003	Aumento a un conjunto de 9.000 imágenes, 6.500-1.500-1000, de las cuales 1.000 son generadas a partir de imágenes realistas con <i>CartoonGAN</i> .
0028-CvR-CMCMCMCMD-0004	Mismo experimento que 0004 pero generando imágenes a partir de otros dibujos animados con <i>CartoonGAN</i> .
0029-CvR-CMCMCMCMD-0005	Conjunto de 6.500 imágenes, 4.000-1.500-1.000, ninguna de ellas es generada. Se pretende comparar resultados con anteriores y el siguiente experimento.
0030-CvR-CMCMCMCMD-0006	Mismo conjunto que 0005, pero 1.000 imágenes son generadas a partir de otros dibujos animados con <i>CartoonGAN</i> .
0031-CvR-CMCMCMCMD-0007	Mismo conjunto que 0006, pero las imágenes ahora se generan a partir de imágenes realistas con <i>CartoonGAN</i> .
0032-CvR-CMCMCMCMD-0008	Conjunto con 288 imágenes, 96-96-96, solo contienen paisajes. Los tres conjuntos son iguales
0033-CvR-CMCMCMCMD-0009	Conjunto con 288 imágenes, 96-96-96, solo contienen personas. Los tres conjuntos son iguales.
0034-CvR-CMCMCMCMD-0010	Conjunto de 2.692, 192-1.500-1.000, para comprobar con los mismos conjunto de evaluación que las primeras pruebas. Solo usa caras.
0035-CvR-CMCMCMCMD-0011	Igual que el 0011 pero se usan paisajes.
0036-CvR-CMCMCMCMD-0012	Conjunto de 2.692, 192-1.500-1.000, ninguna de ellas es generada.
0037-CvR-CMCMCMCMD-0013	Conjunto de 6.500 imágenes, 4.000-1.500-1.000, se crean imágenes (1.000) a partir de imágenes realistas usando filtros tradicionales.
0038-CvR-CMCMCMCMD-0014	Igual que el 0013, pero se generan 1.000 imágenes con <i>CartoonGAN</i> .
0038-CvR-CMCMCMCMD-0014-B	Prueba 0014 relanzada.
0039-CvR-CMCMCMCMD-0015	Conjunto de 2.692, 192-1.500-1.000, 38 imágenes son generadas con filtros tradicionales a partir de paisajes.
0040-CvR-CMCMCMCMD-0016	Similar al 0015, pero generando imágenes a partir de personas.
0041-CvR-CMCMCMCMD-0017	Igual que el 0004, pero se generan imágenes con filtros tradicionales.
0042-CvR-CMCMCMCMD-0018	Igual que el 0002, pero se usa interpolación <i>lanczos</i> para redimensionar las imágenes.
0043-CvR-CMCMCMCMD-0019	Igual que el 0002, pero se usa <i>lanczos</i> como método de interpolación.
0044-CvR-CMCMCMCMD-0020	Similar al 0004 pero el tamaño de imagen pasa de 256px a 200px.

0045-CvR-CMCMCMCMD-0021	Igual que 0020, con tamaño de 150px.
0046-CvR-CMCMCMCMD-0022	Igual que 0020, con tamaño de 128px.
0047-CvR-CMCMCMCMD-0023	Igual que el 0002, pero las imágenes se recortan horizontalmente para evitar estiramiento de imágenes al redimensionar y deformar las formas.
0048-CvR-CMCMCMCMD-0024	14.000 imágenes, 10.000-3.000-1.000, 2.500 imágenes generadas a partir de dibujos animados usando CartoonGAN.
0049-CvR-CMCMCMCMD-0025	Introducción de la mejora vista en la prueba 0003 a la prueba 0024.
0050-CvR-CMCMCMCMD-0026	Igual que 0004, pero el conjunto de test consta de 2.000 imágenes.
0051-CvR-CMCMCMCMD-0027	Igual que 0025, pero el conjunto de test consta de 2.000 imágenes.
<b>Experimentación con el modelo</b>	
0052-CvR-CMCMCMCMD-0028	32 filtros de tamaño 3x3.
0053-CvR-CMCMCMCMD-0029	Se añade una capa convolucional y <i>max-pooling</i> más que en el 0028.
0054-CvR-CMCMCMCMCMD-0030	Se añade una capa convolucional y <i>max-pooling</i> más que en el 0029.
0055-CvR-CMCMCMCMCMCMD-0031	Se añade una capa convolucional y <i>max-pooling</i> más que en el 0030.
0056-CvR-CMCMCMCMCMD-0032	Misma que 0030 pero con 200 <i>epochs</i> .
0057-CvR-CMCMCMCMCMCMD-0033	Igual que la 0031 pero con filtros de 2x2.
0058-CvR-CMCMCMCMCMCMD-0034	Igual que 0033 pero con filtros de 4x4.
0059-CvR-CMCMCMCMCMCMD-0035	Igual que 0034 pero con filtros de 5x5.
0060-CvR-CMCMCMCMCMCMD-0036	Igual que el 0031 pero con 16 filtros por capa.
0061-CvR-CMCMCMCMCMCMD-0037	Misma prueba que la 0036 pero con 64 filtros por capa.
0062-CvR-CMCMCMCMCMCMD-0038	Misma prueba que la 0037 pero con 128 filtros por capa.
0063-CvR-CMCMCMCMCMCMD-0039	Número de filtros ascendente, 32-32-64-64-128-128.
0064-CvR-CMCMCMCMCMCMD-0040	Prueba con capa BatchNormalization después de cada capa convolucional de la prueba 0031.
0065-CvR-CMCMCMCMCMCMD-0041	Igual que la anterior pero la capa BN se pone al principio del clasificador solamente.
0066-CvR-CMCMCMCMCMCMD-0042	Igual que 0031 pero la capa densa tiene 256 neuronas.
0067-CvR-CMCMCMCMCMCMD-0043	Igual que 0043 pero con 1024 neuronas.
0068-CvR-CMCMCMCMCMCMD-0044	Igual que 0042, pero se añade otra capa con 512 neuronas.
0069-CvR-CMCMCMCMCMCMD-0045	Similar al 0044, pero con 128 y 256 neuronas en cada una de las dos capas densas.
0070-CvR-CMCMCMCMCMCMD-0046	Igual que 0031 pero incluye <i>dropout</i> del 10%.
0071-CvR-CMCMCMCMCMCMD-0047	Igual que 0046 pero incluye <i>dropout</i> del 20%.
0071-CvR-CMCMCMCMCMCMD-0047-B	Relanzamiento del 0047.

0072-CvR-CMCMCMCMCMCMD-0048	Igual que 0047 pero incluye <i>dropout</i> del 30%.
0073-CvR-CMCMCMCMCMCMD-0049	Igual que 0048 pero incluye <i>dropout</i> del 40%.
0074-CvR-CMCMCMCMCMCMD-0050	Similar al 0048 pero con función <i>sigmoid</i> en las capas convolucionales.
0075-CvR-CMCMCMCMCMCMD-0051	Similar al 0048 pero con función <i>tanH</i> en las capas convolucionales.
0076-CvR-CMCMCMCMCMCMD-0052	Similar al 0048 pero con optimizador SGD.
0077-CvR-CMCMCMCMCMCMD-0053	Similar al 0048 pero con optimizador Adam con 150 <i>epochs</i> .
0078-CvR-CMCMCMCMCMCMD-0054	Similar al 0048 pero con optimizador Adamax con 150 <i>epochs</i> .
0079-CvR-CMCMCMCMCMCMD-0055	Mismo que el 0053 pero con 250 <i>epochs</i> .
0080-CvR-CMCMCMCMCMCMD-0056	Mismo que el 0054 pero con 350 <i>epochs</i> .

## Resumen de objetivos

Experimento	Entrenamiento	Validación	Test
<b>Experimentación con el conjunto</b>			
0025-CvR-CMCMCMCMMD-0001	100%	92,25%	73,33%
0026-CvR-CMCMCMCMMD-0002	100%	87,67%	81,2%
0027-CvR-CMCMCMCMMD-0003	99,98%	86,27%	78,6%
0028-CvR-CMCMCMCMMD-0004	99,97%	90%	84,7%
0029-CvR-CMCMCMCMMD-0005	99,8%	85,73%	78,5%
0030-CvR-CMCMCMCMMD-0006	100%	87,73%	84,3%
0031-CvR-CMCMCMCMMD-0007	99,8%	83,13%	79,7%
0032-CvR-CMCMCMCMMD-0008	100%	100%	100%
0033-CvR-CMCMCMCMMD-0009	100%	100%	100%
0034-CvR-CMCMCMCMMD-0010	100%	68,8%	58,8%
0035-CvR-CMCMCMCMMD-0011	100%	74,07%	65,1%
0036-CvR-CMCMCMCMMD-0012	100%	73,07%	69,1%
0037-CvR-CMCMCMCMMD-0013	99,92%	90%	79,1%
0038-CvR-CMCMCMCMMD-0014	100%	89,73%	75,2%
0038-CvR-CMCMCMCMMD-0014-B	93,6%	84,8%	71,1%
0039-CvR-CMCMCMCMMD-0015	100%	72,8%	68,4%
0040-CvR-CMCMCMCMMD-0016	100%	65,8%	58,7%
0041-CvR-CMCMCMCMMD-0017	97,63%	85,87%	78,3%
0042-CvR-CMCMCMCMMD-0018	100%	88,87%	83,7%

0043-CvR-CMCMCMCMD-0019	100%	91,13%	83,9%
0044-CvR-CMCMCMCMD-0020	100%	90,47%	83,3%
0045-CvR-CMCMCMCMD-0021	100%	90%	83,3%
0046-CvR-CMCMCMCMD-0022	100%	90,47%	83,6%
0047-CvR-CMCMCMCMD-0023	100%	88,4%	84,7%
0048-CvR-CMCMCMCMD-0024	99,99%	90,57%	85,6%
0049-CvR-CMCMCMCMD-0025	99,97%	89,8%	89,3%
0050-CvR-CMCMCMCMD-0026	99,97%	90%	83%
0051-CvR-CMCMCMCMD-0027	99,97%	89,8%	88,65%
<b>Experimentación con el modelo</b>			
0052-CvR-CMCMCMD-0028	100%	87,4%	83,65%
0053-CvR-CMCMCMCMD-0029	100%	90,23%	88,3%
0054-CvR-CMCMCMCMCMD-0030	99,86%	89,2%	90,35%
0055-CvR-CMCMCMCMCMCMD-0031	100%	90,27%	91,05%
0056-CvR-CMCMCMCMCMD-0032	99,71%	89,53%	86,7%
0057-CvR-CMCMCMCMCMCMD-0033	99,4%	91,4%	88,55%
0058-CvR-CMCMCMCMCMCMD-0034	99,99%	90,53%	90,35%
0059-CvR-CMCMCMCMCMCMD-0035	100%	90,27%	90,4%
0060-CvR-CMCMCMCMCMCMD-0036	99,36%	89,77%	90,4%
0061-CvR-CMCMCMCMCMCMD-0037	99,98%	91,43%	90,85%
0062-CvR-CMCMCMCMCMCMD-0038	99,9%	89,23%	90,8%
0063-CvR-CMCMCMCMCMCMD-0039	100%	91,1%	89,6%
0064-CvR-CMCMCMCMCMCMD-0040	99,06%	88,87%	88,60%
0065-CvR-CMCMCMCMCMCMD-0041	99,85%	88,8%	90,5%
0066-CvR-CMCMCMCMCMCMD-0042	99,8%	91,13%	90,15%
0067-CvR-CMCMCMCMCMCMD-0043	100%	88,67%	89%
0068-CvR-CMCMCMCMCMDD-0044	96,68%	87,67%	85,2%
0069-CvR-CMCMCMCMCMDD-0045	99,79%	87,7%	87,45%
0070-CvR-CMCMCMCMCMCMD-0046	99,9%	91,43%	91,65%
0071-CvR-CMCMCMCMCMCMD-0047	99,58%	92,5%	90,9%
0071-CvR-CMCMCMCMCMCMD-0047-B	99,91%	92,43%	90,6%
0072-CvR-CMCMCMCMCMCMD-0048	99,9%	91,87%	92,4%
0073-CvR-CMCMCMCMCMCMD-0049	89,41%	85,07%	79,8%
0074-CvR-CMCMCMCMCMCMD-0050	86,9%	78,96%	76,6%
0075-CvR-CMCMCMCMCMCMD-0051	100%	90,8%	90,15%
0076-CvR-CMCMCMCMCMCMD-0052	86,08%	90,73%	83,75%

0077-CvR-CMCMCMCMCMCMD-0053	99,97%	93,03%	91,75%
0078-CvR-CMCMCMCMCMCMD-0054	100%	92,93%	91,65%
0079-CvR-CMCMCMCMCMCMD-0055	100%	92,93%	91,65%
0080-CvR-CMCMCMCMCMCMD-0056	99,9%	93,03%	93,1%