**Modelo CNN manual**

**2. Creación de modelo**

Texto

Descripción generada automáticamente

En este fragmento de código se diseña una red convolucional básica para clasificación binaria de imágenes. Las capas convolucionales buscan extraer patrones visuales, las capas de MaxPooling reducen dimensionalidad, y la capa densa final realiza la clasificación. La función de activación 'softmax' es adecuada para problemas de clasificación.

El objetivo es que diferencie entre imágenes de perros y gatos, pero un modelo tan simple necesitará ajustes adicionales.

Texto

Descripción generada automáticamente

El summary nos muestra como queda nuestro modelo.

La primera capa convolucional tiene 16 filtros, tamaño 3x3 y genera salidas de forma (254, 254, 16). Luego, una capa de MaxPooling reduce la dimensión a la mitad (127, 127, 3).

La segunda capa convolucional duplica a 32 el numero de filtros y el MaxPooling reduce de nuevo la dimensión a la mitad, 62\*62.

La capa Flatten toma las salidas tridimensionales generadas por las capas convolucionales y las transforma en un vector unidimensional. (123008 indica el tamaño de este vector aplanado 62\*62\*32)

La capa densa final tiene 2 unidades, correspondientes a una clasificación binaria, y utiliza la función de activación softmax.

La primera capa convolucional tiene 448 parámetros., la segunda 4640 y la capa densa final 246018

**3. Entrenamiento**

Texto

Descripción generada automáticamente

Este código entrena el modelo durante 5 épocas utilizando el conjunto de entrenamiento, valida su rendimiento en el conjunto de validación y luego evalúa su rendimiento final en el conjunto de prueba.

Este proceso permite evaluar la capacidad del modelo para generalizar a datos no vistos y proporciona métricas de rendimiento.

Texto

Descripción generada automáticamente con confianza baja

Problemas de este modelo:

1. La precisión en el conjunto de entrenamiento es significativamente mayor (93.95%) en comparación con la precisión en el conjunto de prueba (51.76%) lo que indica overfitting.
2. La loss en el conjunto de prueba es alta (2.20), indicando que el modelo no realiza bien la clasificación en el conjunto de datos no vistos, por culpa del overfitting.
3. La precisión en el conjunto de validación varía a lo largo de las épocas (55.0% en la época 1 y 56.7% en la época 5). Puede indicar que el modelo no está convergiendo de manera consistente o que la tasa de aprendizaje podría necesitar ajustes.

Para atacar estos problemas podemos usar capas de DropOut para el overfitting, modificar la red aumentando o reduciendo la complejidad y experimentar con la tasa de aprendizaje.

**4.Trabajo adicional**

Texto

Descripción generada automáticamente

En este nuevo modelo:

-Se han añadido más capas convolucionales y cambiado el tamaño de los filtros.

-Para evitar el overfitting se han añadido capas de normalización y de tipo Dropout.

-Hay otra capa densa antes de la capa densa final

Texto

Descripción generada automáticamente con confianza media

Texto

Descripción generada automáticamente

En la parte del entrenamiento guardaré el modelo en cada época para comprobar luego cual tiene más precisión.

Imagen que contiene Texto

Descripción generada automáticamente

Estos resultados comparados con los anteriores:

-La precisión alcanza un 96.75%y la loss se ha ido reduciendo cada época, indicando una capacidad de aprendizaje a mejorado.

-En el conjunto de validación también hay mejoras, ha alcanzado un 70.9% en la última época. Esto indica que el modelo está generalizando mejor los datos no vistos y no hay tanto overfitting.

-También ha sido entrenado durante más épocas, lo que ha permitido una mayor capacidad de ajuste.

Texto

Descripción generada automáticamente

Después del entrenamiento compruebo cómo funciona el modelo en cada fase de su entrenamiento y se puede observar que aún puede haber problemas de sobreajuste ya que la precisión varía demasiado entere las épocas. Alcanza su mejor precisión y menor loss en la época 4.

Para mejorarlo más puede ser conveniente aumentar la cantidad de datos, usar data augmantation y usar más técnicas contra el overfitting. También hay que experimentar con la arquitectura del modelo, la tasa de aprendizaje y otros parámetros

**Transfer learning**

Texto

Descripción generada automáticamente

A diferencia del modelo anterior aquí utilizamos la CNN preentreanada EfficientNetB0 como base.

Al cargar el modelo especificamos que se carguen los pesos parendidos durante el entrenamiento con ImageNet; se indica que no se deben incluir las capas densas superiores (solo se carga la parte convolucional de EfficientNetB0); Se especifica el tamaño de los atos de entrada.

En el nuevo modelo, se “congela” el conjunto de capas convolucionales del modelo EfficientNetB0 para evitar que se actualicen durante el entrenamiento y que no se olvide lo que ya tiene aprendido.

Se usa la capa Global Average Pooling 2D después de la arquitectura base para reducir las dimensiones de salida del modelo.

Se añade una capa densa adicional con activación relu y otra capa de Dropout para evitar el sobreentrenamiento.

Finalmente, la última capa densa con 2 salidas y función de activación softmax. En el compilador se le especifica una tasa de aprendizaje.

Texto

Descripción generada automáticamente

Podemos ver como EfficientNet viene con una dimensión de 8x8 con 1280 canales al final de su parte convolucional. GlobalAveragePooling2D educe las dimensiones a (None, 1280)

Texto

Descripción generada automáticamente

Se entrenará durante 22 épocas y se guardará el desarrollo del modelo en “saved\_models”

Epoch 1/22

63/63 [...] - ETA: 0s - loss: 0.2313 - accuracy: 0.9130

Epoch 1: saving model to saved\_models\model\_epoch\_01.keras

63/63 [...] - 196s 3s/step - loss: 0.2313 - accuracy: 0.9130 - val\_loss: 0.0543 - val\_accuracy: 0.9920

Epoch 2/22

63/63 [...] - ETA: 0s - loss: 0.0497 - accuracy: 0.9885

Epoch 2: saving model to saved\_models\model\_epoch\_02.keras

63/63 [...] - 170s 3s/step - loss: 0.0497 - accuracy: 0.9885 - val\_loss: 0.0298 - val\_accuracy: 0.9940

Epoch 3/22

63/63 [...] - ETA: 0s - loss: 0.0322 - accuracy: 0.9920

Epoch 3: saving model to saved\_models\model\_epoch\_03.keras

63/63 [...] - 143s 2s/step - loss: 0.0322 - accuracy: 0.9920 - val\_loss: 0.0215 - val\_accuracy: 0.9950

Epoch 4/22

63/63 [...] - ETA: 0s - loss: 0.0232 - accuracy: 0.9945

Epoch 4: saving model to saved\_models\model\_epoch\_04.keras

63/63 [...] - 85s 1s/step - loss: 0.0232 - accuracy: 0.9945 - val\_loss: 0.0172 - val\_accuracy: 0.9970

Epoch 5/22

63/63 [...] - ETA: 0s - loss: 0.0191 - accuracy: 0.9945

Epoch 5: saving model to saved\_models\model\_epoch\_05.keras

63/63 [...] - 159s 3s/step - loss: 0.0191 - accuracy: 0.9945 - val\_loss: 0.0151 - val\_accuracy: 0.9960

Epoch 6/22

63/63 [...] - ETA: 0s - loss: 0.0126 - accuracy: 0.9975

Epoch 6: saving model to saved\_models\model\_epoch\_06.keras

63/63 [...] - 172s 3s/step - loss: 0.0126 - accuracy: 0.9975 - val\_loss: 0.0140 - val\_accuracy: 0.9950

Epoch 7/22

63/63 [...] - ETA: 0s - loss: 0.0137 - accuracy: 0.9970

Epoch 7: saving model to saved\_models\model\_epoch\_07.keras

63/63 [...] - 173s 3s/step - loss: 0.0137 - accuracy: 0.9970 - val\_loss: 0.0119 - val\_accuracy: 0.9960

Epoch 8/22

63/63 [...] - ETA: 0s - loss: 0.0091 - accuracy: 0.9975

Epoch 8: saving model to saved\_models\model\_epoch\_08.keras

63/63 [...] - 173s 3s/step - loss: 0.0091 - accuracy: 0.9975 - val\_loss: 0.0115 - val\_accuracy: 0.9940

Epoch 9/22

63/63 [...] - ETA: 0s - loss: 0.0096 - accuracy: 0.9975

Epoch 9: saving model to saved\_models\model\_epoch\_09.keras

63/63 [...] - 173s 3s/step - loss: 0.0096 - accuracy: 0.9975 - val\_loss: 0.0118 - val\_accuracy: 0.9950

Epoch 10/22

63/63 [...] - ETA: 0s - loss: 0.0060 - accuracy: 0.9995

Epoch 10: saving model to saved\_models\model\_epoch\_10.keras

63/63 [...] - 174s 3s/step - loss: 0.0060 - accuracy: 0.9995 - val\_loss: 0.0103 - val\_accuracy: 0.9970

Epoch 11/22

63/63 [...] - ETA: 0s - loss: 0.0060 - accuracy: 0.9990

Epoch 11: saving model to saved\_models\model\_epoch\_11.keras

63/63 [...] - 173s 3s/step - loss: 0.0060 - accuracy: 0.9990 - val\_loss: 0.0101 - val\_accuracy: 0.9950

Epoch 12/22

63/63 [...] - ETA: 0s - loss: 0.0066 - accuracy: 0.9985

Epoch 12: saving model to saved\_models\model\_epoch\_12.keras

63/63 [...] - 173s 3s/step - loss: 0.0066 - accuracy: 0.9985 - val\_loss: 0.0095 - val\_accuracy: 0.9960

Epoch 13/22

63/63 [...] - ETA: 0s - loss: 0.0046 - accuracy: 0.9990

Epoch 13: saving model to saved\_models\model\_epoch\_13.keras

63/63 [...] - 172s 3s/step - loss: 0.0046 - accuracy: 0.9990 - val\_loss: 0.0099 - val\_accuracy: 0.9950

Epoch 14/22

63/63 [...] - ETA: 0s - loss: 0.0037 - accuracy: 1.0000

Epoch 14: saving model to saved\_models\model\_epoch\_14.keras

63/63 [...] - 173s 3s/step - loss: 0.0037 - accuracy: 1.0000 - val\_loss: 0.0092 - val\_accuracy: 0.9960

Epoch 15/22

63/63 [...] - ETA: 0s - loss: 0.0053 - accuracy: 0.9985

Epoch 15: saving model to saved\_models\model\_epoch\_15.keras

63/63 [...] - 172s 3s/step - loss: 0.0053 - accuracy: 0.9985 - val\_loss: 0.0084 - val\_accuracy: 0.9970

Epoch 16/22

63/63 [...] - ETA: 0s - loss: 0.0040 - accuracy: 0.9995

Epoch 16: saving model to saved\_models\model\_epoch\_16.keras

63/63 [...] - 175s 3s/step - loss: 0.0040 - accuracy: 0.9995 - val\_loss: 0.0095 - val\_accuracy: 0.9970

Epoch 17/22

63/63 [...] - ETA: 0s - loss: 0.0039 - accuracy: 0.9995

Epoch 17: saving model to saved\_models\model\_epoch\_17.keras

63/63 [...] - 102s 2s/step - loss: 0.0039 - accuracy: 0.9995 - val\_loss: 0.0087 - val\_accuracy: 0.9960

Epoch 18/22

63/63 [...] - ETA: 0s - loss: 0.0028 - accuracy: 0.9995

Epoch 18: saving model to saved\_models\model\_epoch\_18.keras

63/63 [...] - 145s 2s/step - loss: 0.0028 - accuracy: 0.9995 - val\_loss: 0.0089 - val\_accuracy: 0.9960

Epoch 19/22

63/63 [...] - ETA: 0s - loss: 0.0026 - accuracy: 1.0000

Epoch 19: saving model to saved\_models\model\_epoch\_19.keras

63/63 [...] - 185s 3s/step - loss: 0.0026 - accuracy: 1.0000 - val\_loss: 0.0085 - val\_accuracy: 0.9960

Epoch 20/22

63/63 [...] - ETA: 0s - loss: 0.0024 - accuracy: 1.0000

Epoch 20: saving model to saved\_models\model\_epoch\_20.keras

63/63 [...] - 187s 3s/step - loss: 0.0024 - accuracy: 1.0000 - val\_loss: 0.0095 - val\_accuracy: 0.9950

Epoch 21/22

63/63 [...] - ETA: 0s - loss: 0.0017 - accuracy: 1.0000

Epoch 21: saving model to saved\_models\model\_epoch\_21.keras

63/63 [...] - 188s 3s/step - loss: 0.0017 - accuracy: 1.0000 - val\_loss: 0.0092 - val\_accuracy: 0.9940

Epoch 22/22

63/63 [...] - ETA: 0s - loss: 0.0019 - accuracy: 0.9995

Epoch 22: saving model to saved\_models\model\_epoch\_22.keras

63/63 [...] - 189s 3s/step - loss: 0.0019 - accuracy: 0.9995 - val\_loss: 0.0085 - val\_accuracy: 0.9960

Podemos observar como la loss disminuye rápidamente desde el principio, lo cual es un buen indicativo de que el modelo está aprendiendo eficazmente. La accuracy también aumenta y alcanza un nivel muy alto, llegando al 99.95% en la última época. Esto indica que el modelo está capturando con éxito los patrones en los datos de entrenamiento.

Lo mismo pasa con el conjunto de validación, esto sugiere que el modelo no está sobreajustando y puede generalizar bien a nuevos datos.

A lo largo de las épocas, las métricas de loss y precisión se estabilizan, indicando que el modelo ha convergido a una solución que generaliza bien y no está sobreajustando.

Texto

Descripción generada automáticamente

Por último, hago las pruebas con cada modelo para ver cómo funcionan con información a la que no se han enfrentado, estos son los resultados brutos:

24/24 [...] - 69s 2s/step - loss: 0.2381 - accuracy: 0.9079

Modelo en la época 1 - Test accuracy: 0.9078590869903564

24/24 [...] - 62s 2s/step - loss: 0.1783 - accuracy: 0.9322

Modelo en la época 2 - Test accuracy: 0.9322493076324463

24/24 [...] - 66s 2s/step - loss: 0.1562 - accuracy: 0.9377

Modelo en la época 3 - Test accuracy: 0.9376693964004517

24/24 [...] - 62s 2s/step - loss: 0.1464 - accuracy: 0.9390

Modelo en la época 4 - Test accuracy: 0.9390243887901306

24/24 [...] - 67s 2s/step - loss: 0.1391 - accuracy: 0.9404

Modelo en la época 5 - Test accuracy: 0.9403793811798096

24/24 [...] - 62s 2s/step - loss: 0.1237 - accuracy: 0.9472

Modelo en la época 6 - Test accuracy: 0.9471544623374939

24/24 [...] - 67s 2s/step - loss: 0.1300 - accuracy: 0.9499

Modelo en la época 7 - Test accuracy: 0.9498645067214966

24/24 [...] - 62s 2s/step - loss: 0.1242 - accuracy: 0.9499

Modelo en la época 8 - Test accuracy: 0.9498645067214966

24/24 [...] - 62s 2s/step - loss: 0.1124 - accuracy: 0.9553

Modelo en la época 9 - Test accuracy: 0.9552845358848572

24/24 [...] - 27s 925ms/step - loss: 0.1253 - accuracy: 0.9485

Modelo en la época 10 - Test accuracy: 0.9485095143318176

24/24 [...] - 27s 916ms/step - loss: 0.1198 - accuracy: 0.9539

Modelo en la época 11 - Test accuracy: 0.9539295434951782

24/24 [...] - 27s 916ms/step - loss: 0.1269 - accuracy: 0.9485

Modelo en la época 12 - Test accuracy: 0.9485095143318176

24/24 [...] - 27s 912ms/step - loss: 0.1142 - accuracy: 0.9526

Modelo en la época 13 - Test accuracy: 0.9525745511054993

24/24 [...] - 27s 916ms/step - loss: 0.1230 - accuracy: 0.9499

Modelo en la época 14 - Test accuracy: 0.9498645067214966

24/24 [...] - 27s 919ms/step - loss: 0.1376 - accuracy: 0.9458

Modelo en la época 15 - Test accuracy: 0.9457994699478149

24/24 [...] - 27s 917ms/step - loss: 0.1134 - accuracy: 0.9580

Modelo en la época 16 - Test accuracy: 0.9579945802688599

24/24 [...] - 27s 912ms/step - loss: 0.1300 - accuracy: 0.9485

Modelo en la época 17 - Test accuracy: 0.9485095143318176

24/24 [...] - 27s 916ms/step - loss: 0.1241 - accuracy: 0.9539

Modelo en la época 18 - Test accuracy: 0.9539295434951782

24/24 [...] - 27s 920ms/step - loss: 0.1452 - accuracy: 0.9485

Modelo en la época 19 - Test accuracy: 0.9485095143318176

24/24 [...] - 27s 917ms/step - loss: 0.1216 - accuracy: 0.9580

Modelo en la época 20 - Test accuracy: 0.9579945802688599

24/24 [...] - 28s 913ms/step - loss: 0.1228 - accuracy: 0.9539

Modelo en la época 21 - Test accuracy: 0.9539295434951782

24/24 [...] - 27s 916ms/step - loss: 0.1375 - accuracy: 0.9512

Modelo en la época 22 - Test accuracy: 0.9512194991111755

De forma resumida:

Imagen que contiene exterior, teléfono, computadora

Descripción generada automáticamente

La precisión (accuracy) en el conjunto de prueba aumenta con cada época, alcanzando su punto más alto en la época 16 con un 95.80%

A partir de esta capa, el modelo se estabiliza, esto podría indicar que el modelo ha alcanzado su capacidad máxima de aprendizaje y podría estar empezando a hacer overfitting.

Para mejorar este modelo se pueden hacer varias cosas:

1. Ajustar Hiperparametros: tasa de aprendizaje, dropout, batchsize u otra arquitectura.
2. Aumentar cantidad de datos: generar variaciones de las imágenes de entrenamiento, como rotación, volteo horizontal, zoom... (data Augmentation)
3. Cambiar la arquitectura: arquitecturas más profundas o anchas, agregar más capas convolucionales y capas densas.
4. Al estar usando una red preentrenada, ajustar las capas convolucionales superiores para adaptarlas a tu conjunto de datos específico.