Universidad Córdoba

Ingeniería Informática
Mención en Computación
3º Curso

Introducción al Aprendizaje Automático

Trabajo práctico: Dogs vs Cats

Valentín Gabriel Avram Aenachioei
Pablo Fernández Arenas
Damián González Carrasco
Víctor Rojas Muñoz
Ángel Simón Mesa

Año académico 2022-2023
Córdoba, 11 de Mayo, 2023

Índice

Índice de tablas

Índice de figuras

En este documento, vamos a realizar las explicaciones y análisis de resultados pertinentes al trabajo de prácticas de la asignatura, la resolución del problema **Dogs versus Cats**, de la plataforma Kaggle.
No se va a especificar nada sobre el desarrollo del código. Este se ha realizado usando Google Colab. Usando **este enlance** se puede visualizar y ejecutar el código usado para la implementación de los modelos y pruebas realizadas para esta práctica.

1. Introducción

La clasificación de imágenes es una tarea importante en el campo del aprendizaje automático y la visión artificial. Uno de los problemas de clasificación de imágenes más populares es la identificación de perros y gatos en fotografías.
Este problema es desafiante debido a las similitudes visuales entre estas dos especies, lo que lo convierte en un problema de clasificación binaria interesante y relevante para muchos escenarios. En este trabajo, abordaremos el problema de clasificación de perros y gatos utilizando un conjunto de datos proporcionado por la plataforma Kaggle.

Para abordar el problema, trataremos de crear distintos modelos de redes neuronales convolucionales, pues sabemos que son la mejor opción al tratar con imágenes.
Para obtener distintos resultados y poder comparar, crearemos varios modelos distintos, tanto propios como precreados, analizando sobretodo las diferencias estructurales entre ellas.

2. Descripción del problema

El problema a tratar es el conocido problema Dogs versus Cats, de la plataforma Kaggle. La propia plataforma nos proporciona dos conjuntos de datos, un conjunto de entrenamiento compuesto por 25 mil imágenes etiquetadas y un conjunto de datos de evaluación, compuesto por 12500 imágenes sin etiquetar.

El problema de estos conjuntos de datos es el no tener etiquetado el conjunto de evaluación. Al no poder saber si las predicciones realizadas sobre el son correctas o no, no podemos obtener métricas de evaluación fiables.

Para ello, se ha decidido tratar el conjunto de datos por completo.

Para ello, se ha eliminado el conjunto de evaluación, pues no hemos encontrado forma de usarlo de forma adecuada. El conjunto de entrenamiento se ha dividido en cinco, formando cinco subproblemas. Al tener cinco problemas, podemos usar test estadísticos y métodos de comparación *Todos contra Todos* en lugar de *Uno contra Uno*.

Por último, cada uno de los problemas se dividirá en particiones de entrenamiento, validación, y evaluación. Dentro de cada subproblema, el 20 % del conjunto de datos se usa para la evaluación, y el 80 % restante se ha dividido en un 80 % para el entrenamiento y un 20 % para la validación.
Para tratar las imágenes por separado, se han reescalado a un tamaño de 128 por 128 píxeles. Para dar variedad a las imágenes, teniendo mas posibles datos de entrenamiento e intentado aprender diferentes posibilidades dentro de las imágenes, aumentamos aleatoriamente las imágenes de los conjuntos de datos, ya sea rotándolas, aplicando zoom o deformándolas de forma aleatoria.

3. Descripción de los modelos

Para resolver el problema, hemos desarrollado cinco modelos de red neuronal convolucional distintos, tres propios y dos modelos precreados. Se han decidido usar modelos precreados para poder comparar los resultados obtenidos por nuestros propios diseños con los de modelos ampliamente usados y de reconocida fiabilidad.

Para que los modelos sean comparables, todos los modelos han sido entrenados por 15 épocas de 50 imágenes cada uno, con sus respectivas fases de validación, usando la *Cross Entropy* como función de pérdida, y usando el optimizador *Adam*.

3.1. Modelo 1

Este modelo cuenta con dos capas de pooling y dos capas densas. Algunas características notables de este modelo son:

- La primera capa es una capa convolucional con 16 filtros y una ventana de 3x3. Esta capa utiliza la función de activación ReLU.
- A continuación, se aplica una capa de pooling con una ventana de 2x2. Este proceso se repite dos veces más, lo que reduce la dimensión espacial de la salida de la capa convolucional y ayuda a reducir el número de parámetros en el modelo
- Después de las capas de pooling, se utiliza una capa de aplanamiento para convertir los mapas de características en un vector unidimensional.
- Las dos capas densas (fully connected) siguen a la capa de aplanamiento. La primera capa densa tiene 32 neuronas con activación ReLU, seguida de una capa de Dropout con 0.2 de tasa de abandono para evitar el sobreajuste.
- La última capa es una capa densa con una cantidad de neuronas igual al número de clases, que utiliza la función de activación softmax, lo que permite obtener la probabilidad de cada clase.

Este modelo está diseñado para tener un ajuste insuficiente, lo que significa que el modelo no es suficientemente complejo para capturar toda la complejidad de los datos de entrenamiento.

3.2. Modelo 2

Este es un modelo que consta de varias capas convolucionales, capas de pooling y capas completamente conectadas.
El modelo utiliza la función de activación ReLU en las capas convolucionales y fully connected, excepto en la última capa, que utiliza la función de activación softmax para producir una distribución de probabilidad sobre las clases de salida.

- El modelo comienza con una capa convolucional de 64 filtros de 5x5 y una función de activación ReLU. A continuación, sigue una capa de pooling de tamaño 2x2 para reducir la resolución espacial de las características.
- Luego hay otra capa convolucional de 96 filtros de 3x3 seguida de otra capa de pooling de tamaño 4x4. Después, hay una tercera capa convolucional de 128 filtros de 3x3 y una capa de pooling de tamaño 2x2.
- A continuación, las características de salida se aplanan y se pasan a través de dos capas fully connected. La primera capa tiene 128 neuronas y utiliza la función de activación ReLU, mientras que la segunda capa tiene 64 neuronas y también utiliza la función de activación ReLU. Ambas capas tienen un valor de Dropout para evitar el sobre-ajuste.
- Finalmente, hay dos capas completamente conectadas más. La primera capa tiene 32 neuronas y utiliza la función de activación softmax para producir una distribución de probabilidad sobre las clases de salida, mientras que la última capa tiene 2 neuronas y también utiliza la función de activación Softmax para producir una distribución de probabilidad binaria sobre las clases de salida.

Cabe destacar que de los modelos propios, es el mas complejo, tanto en estructura como en funcionalidades en las capas.

3.3. Modelo 3

Algunas características notables de este modelo son:

- La primera capa es una capa convolucional con 32 filtros y una ventana de 3x3. Se utiliza la función de activación linear y se aplica un relleno ("padding") para asegurar que las dimensiones de la salida sean las mismas que las de la entrada. La capa recibe una entrada con forma (128, 128, 3), que corresponde a una imagen en formato RGB.
- A continuación, se aplica una capa de activación LeakyReLU con un valor de α de 0.1. Esta función de activación es similar a ReLU, pero también permite que los valores negativos pasen a través de la capa.
- Se aplica una capa de pooling con una ventana de 2x2 para reducir la dimensión espacial de la salida de la capa convolucional. Se utiliza un relleno para asegurar que las dimensiones de la salida sean las mismas que las de la entrada.
- Después de la capa de pooling, se aplica una capa de Dropout con 0.5 de tasa de abandono para reducir el sobre-ajuste. A continuación, se aplica una capa de aplanamiento para convertir los mapas de características en un vector unidimensional.
- Dos capas densas (fully connected) siguen a la capa de aplanamiento. La primera capa densa tiene 32 neuronas con activación "linear", seguida de otra capa de activación LeakyReLU con un valor de α de 0'1. Se aplica otra capa de Dropout con 0.5 de tasa de abandono para reducir el sobre-ajuste.
- La última capa es una capa densa con una cantidad de neuronas igual al número de clases, que utiliza la función de activación softmax.

Este modelo tiene una arquitectura más compleja que el modelo 1. Las capas de activación LeakyReLU y el aumento de la cantidad de filtros de la capa convolucional pueden permitir que el modelo capture más complejidad en los datos de entrenamiento. Además, el modelo utiliza una tasa de abandono más alta (0.5) en las capas de Dropout, lo que puede ayudar a reducir el sobre-ajuste.

3.4. Modelo VGG16

Este es un modelo precargado, popular en tareas de clasificación de imágenes, útil por se estructura particular.

Este modelo consta de 16 capas, incluyendo 13 capas convolucionales y 3 capas completamente conectadas (fully connected).
Las capas convolucionales se organizan en bloques, donde cada bloque tiene múltiples capas convolucionales seguidas por una capa de pooling. VGG16 utiliza filtros de 3x3 y convoluciones de paso 1, lo que significa que las dimensiones de las características de salida son iguales a las de las características de entrada.
El modelo utiliza la función de activación ReLU en todas las capas convolucionales y fully connected, excepto en la última capa, que utiliza la función de activación softmax para producir una distribución de probabilidad sobre las clases de salida. Además, VGG16 utiliza el relleno ("padding") para asegurarse de que las dimensiones de las características de salida sean iguales a las de las características de entrada.

Hemos decidido usar VGG16 ya que es conocido por su arquitectura profunda y su capacidad para aprender características complejas de las imágenes. Uno de las principales contra de este modelo es que debido a que tiene un gran número de parámetros (aproximadamente 138 millones), el modelo puede tardar mucho tiempo en entrenarse. En nuestro caso particular hemos configurado este modelo con una entrada de imagen de tamaño (128, 128, 3). También hemos establecido el argumento include_top en Falso para omitir la capa de clasificación superior, lo que significa que la salida de la red será una representación de características de las imágenes de entrada en lugar de una probabilidad sobre las clases.

Finalmente, se establece el número de clases en 2 y la función de activación de la capa de clasificación superior en softmax, lo que significa que la salida de la red será una distribución de probabilidad sobre las dos clases.

3.5. Modelo ResNet 50

Este es el segundo modelo de red neuronal convolucional precargado, llamado ResNet50, que se ha entrenado en el conjunto de datos ImageNet. Este modelo tiene una arquitectura profunda y utiliza una técnica llamada conexiones residuales, que evita problemas de degradación de la precisión a medida que la red se profundiza.
El modelo se configura con una entrada de imagen de tamaño (128, 128, 3), que se especifica tanto a través del argumento input_shape del input_tensor. También se establece el argumento include_top en False para omitir la capa de clasificación superior, lo que significa que la salida de la red será una representación de características de las imágenes de entrada en lugar de una probabilidad sobre las clases. Finalmente, se establece el número de clases en 2 y la función de activación de la capa de clasificación superior en Softmax, lo que significa que la salida de la red será una distribución de probabilidad sobre las dos clases.

4. Pruebas realizadas

En cuanto a pruebas, podíamos optar por pruebas Uno contra Uno, o Todos contra Todos.
Las pruebas Uno contra Uno, aunque podrían ser aplicadas, no son útiles al comparar más de dos modelos. Además, el usar éstos métodos supondría más pruebas y más complejidad a la hora de analizarlos.

El usar métodos Todos contra Todos es lo indicado, aunque el número de problemas recomendado es 10 o más, se realizará la comparación sobre solo 5 problemas. Usamos el test de Friedman para determinar si hay alguna diferencia significativa en los modelos. Al confirmar las diferencias, aplicaremos un test post-hoc, el test de Nemenyi, para determinar qué pares de algoritmos o modelos tienen una diferencia significativa en el rendimiento.

Como los test estadísticos no son muy interpretables a simple vista, hemos decidido realizar un ranking promedio sobre el rendimiento de todos los modelos en la fase de evaluación, para ver de forma sencilla que modelos son mejores y peores.

5. Resultados y comparación de modelos

5.1. Test de Friedman

Primero, realizamos el test de Friedman sobre las medidas de precisión tanto en entrenamiento, evaluación y testing. Obtenemos los resultados reflejados en la tabla 1.

Fase	**Estadístico de prueba**	**Valor de p**	α
Entrenamiento	11.68	0.0198	0.05
Validación	13.92	0.0075	0.05
Evaluación	11.52	0.0213	0.05

Tabla 1: Resultados del test de Friedman

Los valores de los estadísticos de prueba para las tres fases nos indican que hay una gran variabilidad en los datos entre los modelos. Por otro lado, los bajos valores de p nos indican que hay una probabilidad muy baja de que los resultados observados sean una casualidad.

Por último, con los valores de α, podemos afirmar que en caso de afirmar que tenemos diferencias significativas, lo podemos afirmar con un 95 % de confianza. Con esto valores, negamos la hipótesis inicial, que los modelos son iguales, con un 95 % de certeza.

5.2. Test de Nemenyi

Una vez hecho el test de Friedman, podemos realizar un test de Nemenyi, comparando cada modelo en cada métrica sobre todos los conjuntos de datos.

Para la fase del entrenamiento, podemos ver que los modelos propios tienen un rendimiento mejor en rendimiento, por la sencillez de la estructura, es decir, son modelos menos complejos. Además, los modelos precreados también han sido previamente entrenados con el conjunto de datos *ImageNet*, por lo que los pesos de la red ya estaban ajustados. Al volver a entrenar con nuestros propios conjuntos de datos, cambiamos el ajuste de los parámetros de los modelos, lo que lleva a un empeoramiento de su rendimiento.
Además, en la figura 1 podemos ver que las distancias críticas que nos indican las mínimas diferencias significativas entre modelos es de 2.72, lo que nos indica que aunque los modelos sean diferentes, tienen similitudes.

Podemos apreciar similitudes entre los modelos precreados y similitudes entre los modelos propios, separándose entre ellos. También es apreciable que el segundo modelo es el mejor en prácticamente todos los casos.

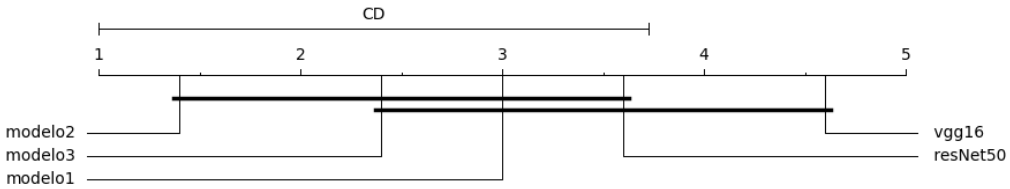

Figura 1: Gráfico del test de Nemenyi para Entrenamiento

En la fase de validación obtenemos resultados similares a los de la fase de entrenamiento. La gran diferencia es que el modelo ResNet50 parecer ser de los mejores. La distancia crítica en este caso es, de nuevo, 2.72, volviendo a diferenciar en gran medida el mejor modelo del peor, e igualando en cierta medida los otros modelos.

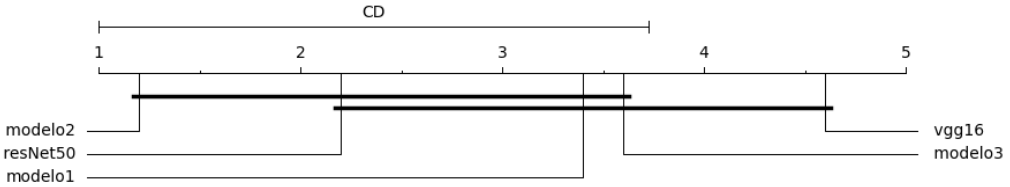

Figura 2: Gráfico del test de Nemenyi para Validación

En cuanto a la fase de evaluación, vemos de nuevo, resultados similares. El segundo modelo, que es el modelo propio mas complejo, resulta tener el mejor rendimiento, seguido por el modelo ResNet50. Los modelos 1 y 3 de creación propia tienen un rendimiento prácticamente igual, lo cual es comprensible por su similar estructura. Por último, el modelo VGG16 muestra ser el peor.

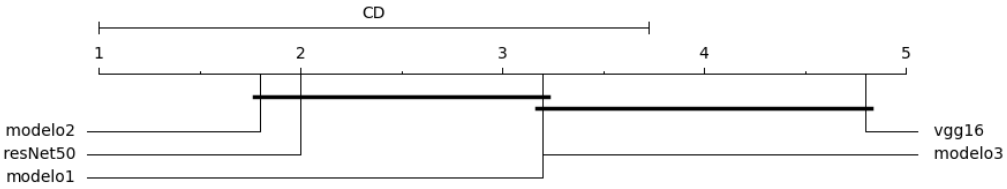

Figura 3: Gráfico del test de Nemenyi para Evaluación

Estos resultados, aunque insatisfactorios, son entendibles. Primeramente, el rendimiento de nuestro segundo modelo es tan bueno como era de esperar, pero en ningún caso era esperable que fuese mejor que los modelos precreados. Aún más inesperado es que el modelo VGG16 sea el de peor rendimiento. El mal rendimiento de los modelos precreados se explica por la segunda fase de entrenamiento, realizada sobre los datos del problema, y el posible desajuste de sus parámetros. Además, el test de Nemenyi no es especialmente acertado en situaciones como ésta, con tan solo 5 conjuntos de datos, cuando el mínimo recomendado son 10 problemas sobre los que comparar el rendimiento de los modelos.

5.3. Ranking promedio

Para poder apreciar una comparación entre los modelos a simple vista, realizaremos un ranking promedio, basándonos en la precisión de las predicciones realizadas. Aunque este ranking no tenga ninguna validez a nivel estadístico, nos permite comparar los modelos de forma sencilla y ordenarlos de mejor a peor de una forma claramente visible.

Dataset	**Modelo1**	**Modelo2**	**Modelo3**	**VGG16**	**ResNet50**
Dataset 1	0.682	0.649	0.643	0.724	0.825
Dataset 2	0.655	0.5	0.605	0.739	0.479
Dataset 3	0.662	0.5	0.731	0.759	0.494
Dataset 4	0.721	0.5	0.742	0.823	0.486
Dataset 5	0.706	0.5	0.747	0.81	0.511
Ranking Promedio	2.8	4.2	2.8	**1.2**	4

Tabla 2: Ranking promedio de los modelos

Con los resultados de la precisión en la fase de evaluación, podemos ver que el modelo VGG16 es el mas preciso con diferencia, mientras que los modelos 1 y 3 están igualados en rendimiento.
El modelo 2, que a su vez es el más complejo, ha demostrado una clara incapacidad de aprender las características del conjunto de entrenamiento, lo que nos refleja que los modelos demasiado complejos no son necesariamente mejores.

Tanto el segundo modelo como el ResNet50 muestran hacer predicciones aleatorias o incluso peores en el caso de ResNet50 para la gran mayoría de conjuntos de datos.

6. Conclusión

Podemos afirmar, en primer lugar, que hemos conseguido resolver el problema, usando distintas aproximaciones, con resultados mas o menos satisfactorios, dependiendo del modelo.

Además, hemos conseguido comparar estos modelos usando test estadísticos, lo que nos ha permitido cuantificar y clasificar estos en función de su rendimiento.

Como punto negativo, podemos ver que algunos de los modelos que preveíamos con un buen rendimiento han tenido peor rendimiento que los modelos propios, ya sea por realizar el entrenamiento de forma incorrecta, sobreentrenar arruinando los parámetros ya precargados o por usar pruebas poco indicadas.

7. Bibliografía

1. Kaggle. (2013). Dogs vs. Cats [Archivo de competencia]. Recuperado de https://www.kaggle.com/c/dogs-vs-cats

2. Brownlee, J. (2019, May 13). How to Develop a Convolutional Neural Network to Classify Photos of Dogs and Cats. Machine Learning Mastery. https://machinelearningmastery.com/how-to-develop-a-convolutional-neural-network-to-classify-photos-of-dogs-and-cats/

3. Statology. (2021, January 25). How to Perform a Nemenyi Test in Python. Retrieved from https://www.statology.org/nemenyi-test-python

4. Sebastián Ventura Soto. (2022). Presentaciones de clase. Introducción a la inteligencia artificial. UNiversidad de Cordoba

5. Llopis, J. (2013, November 14). Test de Friedman. Retrieved May 8, 2023, from https://jllopisperez.com/2013/11/14/test-de-friedman/

6. Python Software Foundation. (2021). Documentación de Python 3.10. Python 3.10. https://docs.python.org/es/3/tutorial/

7. Datagen. (2021). ResNet-50 Guide: Architecture, Implementation Applications. Datagen. https://datagen.tech/guides/computer-vision/resnet-50/

8. Perfectiods. (2020.). Dogs vs Cats Two CNN Models VGG 16 ResNet 50 [Jupyter Notebook]. Kaggle. Recuperado de https://www.kaggle.com/code/perfectiods/dogs-vs-cats-two-cnn-models-vgg-16-resnet-50/notebook