Trabajo de Visión por Computador

TRABAJO-2: Redes neuronales convolucionales

Implementación

VALORACIÓN TOTAL: 8 puntos

Fecha de entrega: 26 de noviembre

Informe a presentar

Para este trabajo como para los demás proyectos debe presentar un informe escrito con sus valoraciones y decisiones adoptadas en cada uno de los apartados de la implementación. También deberá incluirse una valoración sobre la calidad de los resultados encontrados. (hacerlo en pdf). La entrega de código sin memoria explicativa no puntúa.

Normas de la entrega de Prácticas: EL INCUMPLIMIENTO DE ESTAS NORMAS SIGNIFICA PERDIDA DIRECTA DE 1 PUNTO CADA VEZ QUE SE DETECTE UN INCUMPLIMIENTO.

1. El código se debe estructurar como un dos ficheros, uno para los experimentos con CIFAR y otro para los experimentos con Caltech-UCSD, que irán llamando de forma secuencial a distintas funciones, una por cada apartado de la práctica.
2. El código debe estar obligatoriamente comentado explicando lo que realizan los distintos apartados y/o bloques.
3. Todos los ficheros juntos se podrán dentro de un fichero zip/rar..
4. SOLO ENTREGAR EL CODIGO FUENTE. (NO INCLUIR IMÁGENES)
5. Los path que se usen en la lectura de imágenes o cualquier fichero de entrada debe ser siempre "imagenes/nombre_fichero"
6. Todos los resultados numéricos serán mostrados por pantalla. No escribir nada en el disco.
7. La práctica deberá poder ser ejecutada de principio a fín sin necesidad de ninguna selección de opciones. Para ello se deberá de fijar los parámetros por defecto que se consideren óptimos.
8. Poner puntos de parada para mostrar imágenes o datos por consola y el final de cada apartado

Forma de entrega:. Subir el zip al Tablón docente de CCIA (decsai.ugr.es).

Este trabajo de implementación tiene como objetivo principal mostrar cómo usando técnicas de filtrado lineal es posible extraer información relevante presente en una imagen que permite su interpretación.

El objetivo de esta práctica es obtener experiencia práctica en el diseño y entrenamiento de redes neuronales convolucionales profundas, usando Keras. A partir de una arquitectura de base que proporcionamos, hay que diseñar una arquitectura de red profunda mejorada para clasificar imágenes (pequeñas) en 25 categorías.

Se proporcionan dos plantillas para realizar esta práctica, junto con las funciones básicas de lectura de datos, creación de gráficas para la evolución del porcentaje de clasificación en el conjunto de entrenamiento y en el de validación, y el cálculo del porcentaje de entrenamiento en el conjunto de prueba. La primera plantilla debe usarse para los apartados 1 y 2, y la segunda para el apartado 3.

Apartado 1: BaseNet en CIFAR100 (2 puntos)

Conjunto de datos

Para esta parte de la tarea, trabajará con una parte del conjunto de datos CIFAR100. Este conjunto de datos consta de imágenes en color 60K 32x32 de 100 clases, con 600 imágenes por clase. Hay 50K imágenes de entrenamiento y 10K imágenes de prueba. Las imágenes en CIFAR100 son de tamaño 32x32x3, es decir, imágenes en color de 3 canales de 32x32 píxeles.

En las funciones dadas para esta parte de la práctica, hemos modificado este conjunto de datos, reduciendo el número de clases a 25, y por tanto el conjunto de entrenamiento tiene 12500 imágenes y el de prueba 2500. Del conjunto de entrenamiento se reservará un 10% para validación. Puede ajustar su modelo en el conjunto de validación y obtener su rendimiento en el conjunto de prueba.

BaseNet

Creamos un modelo base llamado BaseNet, que puede ejecutar y obtener una precisión de referencia. Para crearlo (y las posteriores mejoras), utiliza las siguientes capas de redes neuronales:

Convolucional, es decir, Conv2D.
Agrupación, p. MaxPooling2D.
Completamente conectado (lineal), es decir, Dense.
Activaciones no lineales, p. relu.
Aplanar, es decir, Flatten.
Normalización, p. BatchNormalization.

BaseNet consta de dos módulos convolucionales (conv-relu-maxpool) y dos capas lineales. La arquitectura precisa se define a continuación:

Layer No.	Layer Type	Kernel size (for conv layers)	Input | Output dimension	Input | Output channels (for conv layers)
1	Conv2D	5	32 | 28	3 | 6
2	Relu	-	28 | 28	-
3	MaxPooling2D	2	28 | 14	-
4	Conv2D	5	14 | 10	6 | 16
5	Relu	-	10 | 10	-
6	MaxPooling2D	2	10 | 5	-
7	Linear	-	400 | 50	-
8	Relu	-	50 | 50	-
9	Linear	-	50 | 25	-

Su objetivo es editar BaseNet para diseñar una arquitectura de red profunda más precisa. En su informe, deberá incluir una tabla similar a la anterior para ilustrar su red final.

Antes de diseñar su propia arquitectura, debe familiarizarse con la arquitectura BaseNet ya proporcionada, el significado de los hiperparámetros y la función de cada capa.

Apartado 2: Mejora del modelo (3 puntos)

Como se indicó anteriormente, su objetivo es crear una red profunda mejorada haciendo elecciones juiciosas de arquitectura e implementación. Una buena combinación de opciones puede hacer que su precisión se acerque al 50%. Para mejorar la red, debe considerar todo lo siguiente.

1. Normalización de datos. La normalización de los datos de entrada hace que el entrenamiento sea más fácil y más sólido. Utilice la clase ImageDataGenerator con los parámetros correctos para que los datos estén bien condicionados (media=0, std dev=1) para mejorar el entrenamiento. Después de las ediciones, asegúrese de que test_transform tenga los mismos parámetros de normalización de datos que train_transform.

2. Aumento de datos. Intente usar algunos de los parámetros de aumento de datos de la clase ImageDataGenerator, como zoom_range y / o horizontal_flip. No debería tener ningún aumento de datos en los conjuntos de validación ni test. Si necesita una mejor comprensión, intente leer el tutorial de Keras sobre transformaciones.

3. Red más profunda. Experimente agregando más capas convolucionales y totalmente conectadas. Agregue más capas conv con canales de salida crecientes y también agregue más capas lineales (fc). No coloque una capa de maxpool después de cada capa conv en su red más profunda, ya que conduce a una pérdida excesiva de información.

4. Capas de normalización. Las capas de normalización ayudan a reducir el sobreajuste y mejorar el entrenamiento del modelo. Las capas de normalización de Keras son una forma fácil de incorporarlas al modelo. Agregue capas de normalización después de las capas conv (BatchNormalization). Agregue capas de normalización después de capas lineales y experimente insertándolas antes o después de las capas ReLU.

5. "Early Stopping". ¿Después de cuántas épocas parar y dejar de entrenar? Esta respuesta en stack-exhange es un buen resumen del uso de divisiones train-val-test para reducir el sobreajuste. Este blog también es una buena referencia para "early stopping". Recuerde, nunca debe usar el conjunto de test en otra cosa que no sea la evaluación final. Mirando las gráficas de pérdida de entrenamiento y precisión en validación, decida cuántas épocas entrenará su modelo. No demasiados (ya que eso conduce a un sobreajuste) y no muy pocos (de lo contrario, su modelo no ha aprendido lo suficiente).

Apartado 3: Transferencia de modelos y ajuste fino con ResNet50 para la base de datos Caltech-UCSD (3 puntos).

Este conjunto de datos se compone de 6033 imágenes de 200 especies de pájaros. Tiene, por tanto, 200 clases, con 3000 imágenes en el conjunto de entrenamiento y 3033 en el de prueba. De nuevo, se dejará un 10% del conjunto de entrenamiento para validación.

Se pide, por una parte, usar ResNet50 como un extractor de características, usándolo preentrenado en ImageNet, y, por otra parte, hace un ajuste fino de toda la red ResNet50, también preentrenada en ImageNet, en el conjunto de datos. Caltech-UCSD.

BONUS : Solo se tendrán en cuenta los bonus si se ha logrado al menos el 75% de los puntos en la parte obligatoria.

Bonus.1 (Hacer propuestas) Finalmente, hay muchos enfoques para mejorar un modelo más allá de los enumeramos anteriormente. Siéntase libre de probar sus propias ideas o enfoques interesantes de ML / CV sobre los que haya leído.

Dado que Colab solo ofrece recursos computacionales limitados, intente limitar racionalmente el tiempo de entrenanmiento y el tamaño del modelo