Trabajo de Visión por Computador

TRABAJO-2: Redes neuronales convolucionales

Implementación

VALORACIÓN TOTAL: 8 puntos

Fecha de entrega: 28 de noviembre

Informe a presentar:

Para este trabajo como para los demás proyectos se 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 único fichero main que irá 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 LAS IMÁGENES DADAS)
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/.rar a PRADO

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 base de red que se proporciona, hay que aprender a experimentar con ella y mejorarla a partir de añadir, modificar o suprimir capas de dicha arquitectura en la tarea de clasificar imágenes en 25 categorías.

Para realizar esta práctica se proporciona el siguiente código/funciones de ayuda:

1. Funciones básicas de lectura de datos
2. Creación de gráficas para la evolución del porcentaje de clasificación en el conjunto de entrenamiento y en el de validación. (apartados 1 y 2)
3. Cálculo del porcentaje de clasificación en el conjunto de prueba (apartado 3)

Apartado 1: BaseNet en CIFAR100 (2 puntos)

Conjunto de datos

En este apartado, se trabajará con una parte del conjunto de datos CIFAR100. Este conjunto de datos consta de 60K imágenes en color de dimensión 32x32x3 (RGB) de 100 clases distintas, con 600 imágenes por clase. Hay 50K imágenes para entrenamiento y 10K imágenes de prueba. Para el desarrollo de práctica solo consideraremos 25 clases de las 100, por tanto el conjunto de entrenamiento tiene 12500 imágenes y el de prueba 2500. Del conjunto de entrenamiento se usará un 10% para validación. Usar las funciones dadas para conseguir dicha reducción.

Modelo base: BaseNet

Comenzamos creando un modelo base llamado BaseNet, que tras su entrenamiento y ejecución nos dará un porcentaje de clasificación de referencia para las posteriores mejoras. Para crearlo (y en las posteriores mejoras), utilizaremos las siguientes capas disponibles en Keras:

Convolucional, es decir, Conv2D.(normal y atrous)
Agrupación: Local y Global MaxPooling2D and AveragePooling2D().
Densa (lineal), es decir, Full connected
Activaciones no lineales, p.e. relu, leaky relu, etc
Aplanar, es decir, Flatten.
Normalización del batch, p.e. BatchNormalization.
Regularización: p.e. Dropout

El modelo 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	-

1.-. Familiarizarse con la arquitectura BaseNet ya proporcionada, el significado de los hiperparámetros y la función de cada capa. Crear el código para el modelo BaseNet

2.- Entrenar el modelo y extraer los valores de accuracy y función de perdida para el conjunto de test. Presentar los resultados de entrenamiento y test usando las funciones proporcionadas.

.

Apartado 2: Mejora del modelo BaseNet (3 puntos)

Ahora el objetivo es crear, haciendo elecciones juiciosas de arquitectura e implementación, una red profunda mejorada partiendo de BaseNet. Una buena combinación de capas puede hacer que la precisión del nuevo modelo se acerque al 50% sobre nuestros datos de Cifar-100. Para mejorar la red, puede considerar añadir cualquier combinación de entre las siguientes opciones de mejora:.

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. Ver la capa de Normalization.

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. Ver las capas de image processing and augmentation.

3. **Aumento profundidad**:. Experimente agregando capas convolucionales. No coloque siempre una capa de maxpool después de cada capa conv, ya que conduce a una pérdida excesiva de información por reducción del número de unidades. Si lo necesita use capas de UpSampling (Transposed Convolution) para aumentar el número de unidades

4. **Batchnormalization**. Las capas de Batch-normalizatión en la mayoría de los casos ayudan a reducir el sobreajuste y mejorar el entrenamiento del modelo. Las capas de Batch-normalizatión de Keras son una forma fácil de incorporarlas al modelo. Agregue capas de normalización después de capas lineales (conv) y antes de la capa ReLU, pero experimente también con después de las capas ReLU.

5.- Regularización con **Dropout**: esta capa selecciona de forma aleatoria un porcentaje de neuronas activas para cada mini-batch, Esto ayuda a la red a especializar las neuronas en informaciones especificas e independientes.

6. "**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 (Evaluar los mínimos y máximos y locales de. La perdida y precisión, respectivamente) No demasiadas (ya que eso conduce a un sobreajuste) y no muy pocas (de lo contrario, su modelo no ha aprendido lo suficiente).

Recuerde que en su informe, deberá incluir una tabla similar a la mostrada en el punto.1 para ilustrar la arquitectura de su red final. Además, deberá explicar los pasos dados con los resultados parciales obtenidos que finalmente le han llevado a introducir las modificaciones propuestas.

Apartado 3: T

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

En este apartado trabajaremos con el conjunto de datos **Caltech-UCSD.** 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. Para leer el conjunto de datos usar las funciones dadas.

Usaremos el modelo de red ResNet50 ya pre-entrenada con ImageNet y que está disponible en Keras. Se pide,

1.- Usar ResNet50 como un extractor de características para los datos de **Caltech-UCSD** disponible en (http://www.vision.caltech.edu/visipedia/CUB-200-2011.html**.** Para ello eliminaremos al menos las dos últimas capas del modelo descargado, añadiremos algunas capas de cálculo adicional y la capa de salida. En concreto realizar los siguientes experimentos:

A.- Eliminar las FC y la salida, substituirlas por nuevas FC y salida y reentrenarlas con CALTECH. Comparar resultados con un modelo en el que unicamente se cambia y reentrena la capa de salida

B.- Eliminar las capas de salida, FC y AveragePooling. Añadir nuevas capas, entrenar la red resultante y comparar con los resultados del punto.A

2.- Realizar un ajuste fino de toda la red ResNet50, al conjunto de datos. Caltech-UCSD. Recordar que el número de épocas a ejeutar debe ser pequeño.

La red ResNet50 pre-entrenada con ImageNet se puede descargar del repositorio de modelos de Keras.

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 (1-3 puntos) (Hacer propuestas) Hay muchas otras posibilidades para mejorar el modelo BaseNet sobre CIFAR-100 usando combinaciones adecuadas de capas. 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 entrenamiento y el tamaño del modelo.

Se valorará cada propuesta en función de su innovación, complejidad y buen uso de Keras. El número de clases usadas en el experimento también se tendrá en cuenta.