Práctica: Entendiendo las redes neuronales convolucionales

Introducción

El aprendizaje profundo (Deep Learning, DL) se ha convertido recientemente en una de las técnicas más potentes en el reconocimiento de patrones, con muchas aplicaciones en una amplia gama de áreas, como el procesamiento del lenguaje natural, el procesamiento de señales o la visión por computadora. En particular, el uso de redes neuronales convolucionales (CNN) para la clasificación de imágenes ha sido una de las aplicaciones más exitosas de las técnicas de DL.

En este ejercicio práctico vamos a explorar las CNN para tener una mejor comprensión de su estructura interna. El objetivo es desarrollar una comprensión básica de cómo funciona CNN y, en particular, mejorar su capacidad de interpretación en términos de las características extraídas por las capas convolucionales.

Competencias

En este enunciado se trabajan las siguientes competencias generales de máster:

- Capacidad para proyectar, calcular y diseñar productos, procesos e instalaciones en todos los ámbitos de la ingeniería informática.
- Capacidad para el modelado matemático, cálculo y simulación en centros tecnológicos y de ingeniería de empresa, particularmente en tareas de investigación, desarrollo e innovación en todos los ámbitos relacionados con la ingeniería informática.
- Capacidad para aplicar los conocimientos adquiridos y solucionar problemas en entornos nuevos o poco conocidos dentro de contextos más amplios y multidisciplinares, siendo capaces de integrar estos conocimientos.
- Poseer habilidades para el aprendizaje continuo, autodirigido y autónomo.
- Capacidad para modelar, diseñar, definir la arquitectura, implantar, gestionar, operar, administrar y mantener aplicaciones, redes, sistemas, servicios y contenidos informáticos.

Las competencias específicas de esta asignatura que se trabajan en esta prueba son:

- Entender qué es el aprendizaje automático en el contexto de la inteligencia artificial

- Distinguir entre los diferentes tipos y métodos de aprendizaje
- Aplicar las técnicas estudiadas a un caso real

Descripción

Para resolver un problema de clasificación de imágenes usando CNN, normalmente necesitamos una gran base de datos de imágenes de entrenamiento que contenga alrededor de 500-1000 imágenes para cada una de las etiquetas de clase que se identificarán en el problema. En este ejemplo, vamos a utilizar un conjunto de datos de imágenes conocido como CIFAR-10 que incluye un gran conjunto de imágenes de aviones, automóviles, pájaros, gatos, venados, perros, ranas, caballos, barcos y camiones. En particular, el conjunto de datos CIFAR-10 consta de 60000 imágenes en color de 32x32 en 10 clases, con 6000 imágenes por clase. Hay 50000 imágenes de entrenamiento y 10000 imágenes de test. La información detallada del conjunto de datos se describe en el siguiente sitio web:

https://www.cs.toronto.edu/~kriz/cifar.html

El manejo de grandes conjuntos de datos de imágenes y CNN requiere una cierta cantidad de recursos computacionales. La mayoría de las computadoras personales de hoy tratarán razonablemente bien los cálculos requeridos en este ejercicio, pero típicamente requerirán unos minutos de tiempo de CPU para ejecutarlos. Una opción razonable si desea acelerar sus resultados es ejecutar sus códigos en la plataforma colaborativa de Google, que es una plataforma de cuaderno jupyter gratuita basada en la nube que le permite escribir y ejecutar código python:

https://colab.research.google.com/notebooks/welcome.ipynb

Además de las bibliotecas que normalmente usamos en el curso (numpy, scipy, sckit-learn, matplotlib, pandas, etc.), este ejercicio requerirá el uso de algunas bibliotecas de python adicionales:

1. Keras con Tensorflow backend:

Keras es una biblioteca que permite diseñar redes neuronales, redes neuronales convolucionales e incluye varias utilidades relacionadas. La documentación se puede encontrar en https://keras.io/ . Antes de instalar Keras, debe instalar un motor de back-end, que es una plataforma informática de bajo nivel para calcular convoluciones 2D, productos tensoriales y algunas otras operaciones matemáticas básicas. Hay diferentes opciones para el backend, pero recomiendo utilizar el backend de Tensorflow. Para hacerlo, solo siga las instrucciones en https://www.tensorflow.org/install/ . En términos generales, la idea es ejecutar los siguientes comandos PIP:

instalación de pip:

pip install --upgrade pip

pip install tensorflow

pip install keras

2. Scikit-image: se requieren bibliotecas de procesamiento de imágenes en Python para el manejo básico de imágenes

https://scikit-image.org/

pip install sckit-image

Ejercicio 1: Exploración de los datos

a) La primera tarea es escribir un código para cargar el conjunto de datos como matrices numpy. Puede cargar el conjunto de datos desde la base de datos incorporada de tensorflow https://www.tensorflow.org/api_docs/python/tf/keras/datasets/cifar10/load_data o descargar directamente los datos del sitio web del conjunto de datos CIFAR-10 que se informó anteriormente.

   Informe los siguientes indicadores: la cantidad de imágenes, la cantidad de clases, la cantidad de observaciones por clase y la cantidad de observaciones en los subconjuntos de entrenamiento y test.

b) Construya una figura de 2x5 (10 paneles dispuestos en 2 filas y 5 columnas) que contenga la primera imagen de cada clase en el conjunto de datos de entrenamiento. Agregue un título en cada panel que indique el nombre de la clase y el valor numérico de la etiqueta de clase correspondiente.

c) Construya una imagen que compare 10 imágenes diferentes de la clase "ship" seleccionada al azar del conjunto de entrenamiento. Describa las principales similitudes y diferencias entre ellos.

Ejercicio 2: Entrenando una CNN a medida

En este ejercicio comenzamos con un ejemplo de clasificación de imágenes que utiliza CNN y el conjunto de datos CIFAR-10 incluido en la documentación de la biblioteca de keras:

https://keras.io/examples/cifar10_cnn/

a) Ejecute el ejemplo usando batch_size = 32 y epochs = 2 sin usar la opción de aumento de datos. Utilice la función model.summary () https://keras.io/models/about-keras-models/ para imprimir un informe de las diferentes capas convolucionales definidas en el modelo. Proporcione una descripción básica de las diferentes capas.

b) Informe el tiempo de CPU necesario para entrenar el modelo. ¿Cuántos parámetros deben determinarse?

Las capas convolucionales tienen dos tipos de parámetros: coeficientes de filtro y coeficientes de sesgo. Una capa convolucional con N núcleos de tamaño mxm tiene un número total de parámetros dados por N * m ^ 2 * NC + N, donde NC es el número de canales de la imagen de entrada. El primer término (N * m ^ 2 * NC) corresponde al número de coeficientes del núcleo y el segundo (N) al número de coeficientes de sesgo (1 sesgo para cada núcleo).

Tenga en cuenta que en la primera capa convolucional, NC corresponde al número de canales de color de la imagen de entrada, mientras que en las capas convolucionales internas, NC corresponde al número de núcleos proporcionados por la capa anterior.

c) Verifique que el número de parámetros de las capas convolucionales informadas por la función model.summary() realmente coincida con los cálculos anteriores.

d) Evalúe la precisión general del modelo utilizando el subconjunto de prueba en la función keras model.evaluate() https://keras.io/models/model/#evaluate . Informe la matriz de confusión y el informe de clasificación proporcionado por las bibliotecas de sklearn.

https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html

https://scikit-learn.org/stable/modules/generated/sklearn.metrics.classification_report.html

¿Qué clase ofrece el mejor y peor rendimiento de clasificación? ¿Observas alguna clase que se clasifica particularmente erróneamente en la matriz de confusión? Proporcione una explicación para ello. Justifica tus respuestas.

e) Replicar los resultados en las secciones anteriores pero usando un modelo en el que el tamaño de las capas convolucionales es 5x5 en lugar de 3x3. Compare los resultados y proporcione una explicación de las diferencias observadas. ¿Qué modelo funciona mejor? ¿Cuál tiene más parámetros entrenables? ¿Cuál tiene más parámetros en las capas convolucionales?

Ejercicio 3: Utilizando una arquitectura CNN predefinida

En este ejercicio estamos utilizando una arquitectura CNN conocida como modelo VGG16 https://keras.io/applications/#vgg16

a) Cree un modelo utilizando la arquitectura VGG16 pero adaptando la forma de la entrada al tamaño de las imágenes del conjunto de datos CIFAR-10. Utilice el la función Model de Keras (https://keras.io/models/model/) especificando los parámetros include_top = True, weights = None, input_tensor = None, input_shape = (32,32,3) y classes = 10. Proporcione un resumen de la arquitectura resultante y describa las principales diferencias con el modelo utilizado en el ejercicio 1.

b) Entrene el modelo VGG16 con batch_size = 32 y epochs = 2 utilizando los datos de entrenamiento y evalúe con el conjunto de test. Compare los resultados con los obtenidos en el ejercicio 1 utilizando una CNN personalizada.

El aprendizaje transferido consiste en utilizar una arquitectura CNN entrenada previamente con otra base de datos de imágenes. La idea principal es que los mapas de características utilizados para resolver un determinado problema de clasificación de imágenes también podrían ser relevantes en otro problema. En esta sección, vamos a utilizar la arquitectura del modelo VGG16 pre-entrenada con la base de datos ImageNet: http://www.image-net.org/

c) Aplique el VGG16 previamente entrenado con el conjunto de datos de imagenet para extraer características y clasificar las características resultantes para entrenar una máquina de vectores de soporte con un núcleo lineal. Use la función Modelo de Kera con weights = 'imagenet' para obtener el modelo VGG16 pre-entrenado y especifique include_top = False para evitar incluir las capas completamente conectadas dedicadas a la clasificación, ya que vamos a usar el SVC para clasificar las características.

Ejercicio 4: visualización de mapas de características

Uno de los principales inconvenientes de los enfoques de aprendizaje profundo es que proporcionan una baja interpretabilidad del procedimiento utilizado para clasificar las imágenes. En esta sección vamos a obtener más información sobre las características extraídas por las capas convolucionales.

Los mapas de características son las imágenes (filtradas) devueltas por cada capa convolucional de una CNN. Para extraer un mapa de características, primero defina un nuevo modelo en el que la salida devuelva la salida de una determinada capa convolucional. Esto se puede hacer usando la función de Modelo de Keras https://keras.io/models/model/ . Por ejemplo, para crear un nuevo modelo (model_new) que proporcione la salida de la primera capa convolucional de un modelo anterior (model_old), ejecutamos:

```
from keras.models import Model
model_new = Model(inputs=model_old.inputs,
outputs=model_old.layers[0].output)
```

donde model_old.layers [0] se refiere a la primera capa convolucional en model_old.

a) Para los modelos en el ejercicio 2 con capas convolucionales 5x5 y 3x3, extraiga los mapas de características de la primera capa convolucional conv2d_25 de las imágenes de entrenamiento. Proporcione una representación gráfica de los mapas de características de cada modelo. Describa las principales diferencias observadas entre los modelos 3x3 y 5x5. Justifica tus respuestas.

b) Compare los mapas de características de la primera capa convolucional del modelo VGG16 pre-entrenado y el modelo VGG16 entrenado con sus datos de entrenamiento. Describa las principales diferencias observadas entre ellos.

c) Compare los mapas de características de la primera y la última capa convolucional del VGG16 pre-entrenado. ¿Qué capa captura más detalles de las imágenes?

Recursos

Esta PEC requiere los recursos siguientes:

Básicos:

Python libraries: Keras/Tensorflow, sckit-image, scikit-learn, numpy.

Complementarios: Manual de teoría de la asignatura, vídeos de la asignatura, web de scikit-learn.

Criterios de valoración

Los ejercicios tendrán asociada la valoración siguiente:

Ejercicio 1: 1 punto

Ejercicio 2: 3 puntos

Ejercicio 3: 3 puntos

Ejercicio 4: 3 puntos

Todo el código, tanto para la lectura y preprocesado de los datos como para la solución de los ejercicios, tiene que estar implementado y funcionar correctamente . El informe debe mostrar claramente las respuestas solicitadas sin interacción del usuario. Se deben razonar las

respuestas de todos los ejercicios. Las respuestas sin justificación no recibirán puntuación.

Formato y fecha de entrega

La PEC se ha de entregar antes del **próximo 27 de mayo** (incluído).

La solución tiene que consistir en un archivo zip que contenga un informe en formato PDF y el código python donde se debe implementar la solución a los ejercicios.

Adjuntad el fichero en el apartado de **Entrega y registro de EC (REC)**. El nombre del fichero debe ser ApellidosNombre_IAA_PRAC.zip.

Para dudas y aclaraciones sobre el enunciado, diríjase al consultor responsable del aula.

Nota: **Propiedad intelectual**

A menudo es inevitable, al producir una obra multimedia, hacer uso de recursos creados por terceras personas. Es por tanto comprensible hacerlo en el marco de una práctica de los estudios del Máster de Ingeniería Informática, siempre que esto se documente claramente y no suponga plagio en la práctica.

Por lo tanto, al presentar una práctica que haga uso de recursos ajenos, se presentará junto con ella un documento en el que se detallen todos ellos, especificando el nombre de cada recurso, su autor, el lugar donde se obtuvo y el su estatus legal: si la obra está protegida por copyright o se acoge a alguna otra licencia de uso (Creative Commons, licencia GNU, GPL ...). El estudiante deberá asegurarse de que la licencia que sea no impide específicamente su uso en el marco de la práctica. En caso de no encontrar la información correspondiente deberá asumir que la obra está protegida por copyright.

Deberán, además, adjuntar los archivos originales cuando las obras utilizadas sean digitales, y su código fuente si corresponde.