COM4402 Introducción a Inteligencia Artificial - Primavera 2023

Tarea N°3: Redes Neuronales Convolucionales

Profesores: Ignacio Bugueño, Alfonso Ehijo
Profesora ayudante: Camila Rapu
Ayudante corrector: Felipe Gómez
Plazo de entrega: Jueves 2 de Noviembre, 23:59 hrs.

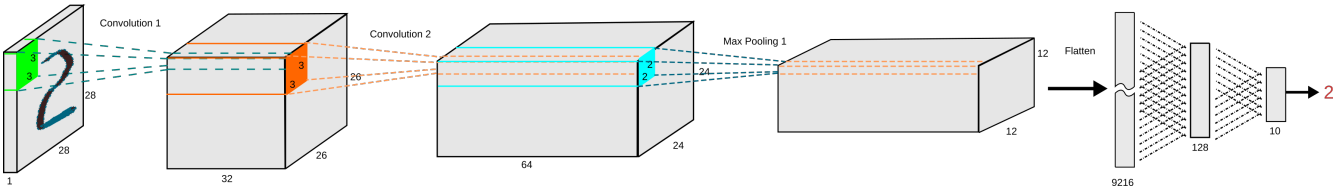
El objetivo de esta tarea es implementar una Red Neuronal Convolucional que logre clasificar los labels correspondientes a la base de datos de manuscritos MNIST y la base de datos de imágenes pequeñas CIFAR-10. La tarea debe ser realizada en Python usando TensorFlow.

En esta tarea se entregará un código base, el cual debe ser usado para realizar las actividades pedidas en la tarea.

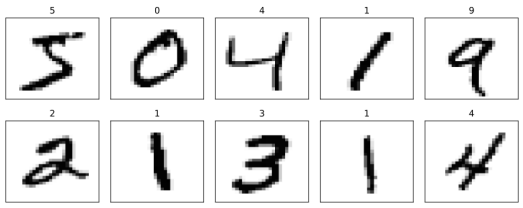
`https://colab.research.google.com/drive/1K2JVSEu1OGeIRPOu2Q9HC-_ruZSX2oO9?usp=sharing`

Convolutional Neural Networks

Las redes neuronales convolucionales (CNN o ConvNet) son una clase de red neuronal profunda, aplicada con mayor frecuencia al análisis de imágenes visuales. También se conocen como redes neuronales artificiales invariantes al desplazamiento o al espacio, basadas en la arquitectura de pesos compartidos de los filtros de convolución que se deslizan a lo largo de las características de entrada, y proporcionan respuestas equivariantes a la traslación conocidas como feature maps. Las CNN tienen aplicaciones en el reconocimiento de imágenes y vídeos, sistemas de recomendación, clasificación de imágenes, segmentación de imágenes, análisis de imágenes médicas, procesamiento del lenguaje natural, interfaces cerebro-ordenador, y series temporales financieras, entre otras.



Implementación de una CNN para MNIST



1) **Implementar una Función de Normalización de Imágenes**
   Objetivo: Implementar una función que normalice las imágenes al intervalo [0,1].

   - Las entradas son números enteros en el intervalo [0,255]
   - Las salidas deben ser flotantes en el intervalo [0,1]

2) **Ampliar la dimensión de entrada**
Objetivo: Escribir una pieza de código que añada una nueva dimensión a 'x_train' y 'x_test'.

- La dimensionalidad de 'x_train' debe ser (60000, 28, 28, 1)
- La dimensionalidad de 'x_test' debe ser (10000, 28, 28, 1)

3) **Implementar una función para One-Hot Encoding**
Objetivo: Implementar una función que codifique un vector de números en una matriz de K clases:

- El primer argumento es un vector con N muestras (dimensiones).
- El segundo argumento es un número K que significa el número de clases.
- Para cada muestra del vector se creará un array con K dimensiones.
- La matriz codificada one-hot debe tener ceros en todas las posiciones excepto en la posición indicada por la muestra actual en el vector de entrada

4) **Implemente una CNN**
Objetivo: Cree una función 'net_1()' que implemente la red especificada en el Google Colab.

5) **Defina los hiperparámetros y entrena la red**
Objetivo: Afina los hiperparámetros hasta que su 'loss' y 'val_loss' converjan a valores bajos.

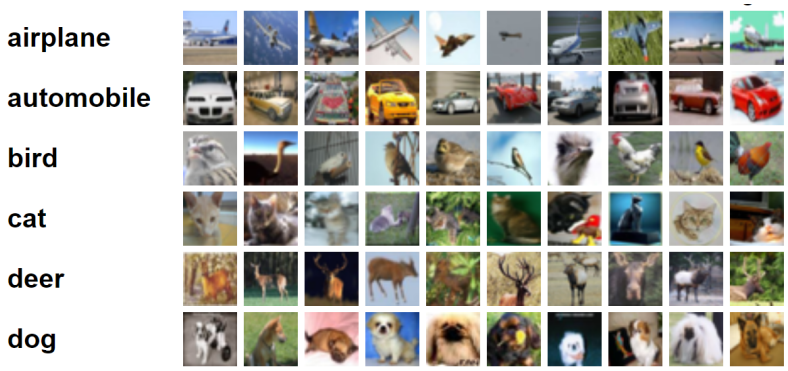
- Tamaño de batch
- Número de épocas de entrenamiento

6) **Implemente una CNN sin max-pooling**
Objetivo: Replique la red que construida anteriormente ('net_1()'), pero esta vez remueva la capa de max pooling y añada un stride=2 al segundo bloque de convolución.

7) **Defina los hiperparámetros y entrene su modelo**
Objetivo: Afina los hiperparámetros hasta que su 'loss' y 'val_loss' converjan a valores bajos. Como antes, se necesita definir algunos hiperparámetros y entrenar la red. Siéntase libre de reutilizar los hiperparámetros que encontró antes.

- Tamaño de batch
- Número de épocas de entrenamiento

Implementación de una CNN para CIFAR10



1) **Codificación One-Hot para los Labels**
Objetivo: Utilice la función 'one_hot()' que creó anteriormente para codificar 'y_train' e 'y_test'.

2) **Normalizar las imágenes**
Objetivo: Utilice la función 'normalize_images()' que creó anteriormente para normalizar las imágenes en 'x_train' y 'x_test'.

3) **Cree su CNN para CIFAR10**
Objetivo: Cree un modelo de red neuronal para entrenar CIFAR usando lo que hemos aprendido hasta ahora.

- Cree una nueva red utilizando 'net_1()' o 'net_2()'.
- Mostrar un resumen de la red.
- Compilar el modelo utilizando 'Adadelta', 'Adagrad', o 'Adam' como su optimizador.

4) **Entrene su modelo**
Objetivo: Entrenar el modelo creado en CIFAR10.

- Entrene la red utilizando 'epochs = 30'.
- Entrene la red utilizando un 'batch_size = 128'.
- Utilice 'validation_split = 0.2' cuando llame a 'fit()'.
- Analice las pérdidas y la precisión.
- Evalúe el rendimiento en el conjunto de pruebas.

Preguntas de discusión a responder en el informe

1) ¿Qué modelo funciona mejor?

2) ¿Qué optimizador funciona mejor?

3) ¿Existe alguna evidencia de overfitting?

4) ¿Cómo podemos mejorar aún más el rendimiento?

Consideraciones

Entregable: El informe y el enlace a su repositorio en Github (con el desarrollo de su código asociado a la evaluación) deben ser subidos a U-Campus a más tardar el día Jueves 2 de Noviembre a las 23:59 hrs. En su entrega final, debe incluir un breve comentario indicando cómo ejecutar su programa. El archivo de conjunto de datos a usar no puede ser renombrado ni modificado de ninguna forma. Cada día de retraso (incluyendo sábados y domingos) será penalizado con 10 décimas de descuento en la nota final.

El informe debe contener como mínimo: resumen, introducción, marco teórico, metodología (incluyendo partes relevantes del código), resultados, análisis, conclusiones generales, bibliografía. Cada elemento en el enunciado debe ser abordado en el informe. Su extensión máxima es de 10 páginas.

Importante: La evaluación de la tarea considerará el correcto funcionamiento del programa, la inclusión de los resultados de los pasos pedidos en el informe, la calidad de los experimentos realizados y de su análisis, la inclusión de las partes importantes del código en el informe, así como la forma, prolijidad y calidad del mismo.

Nota 1: Se subirá un documento indicando la estructura del informe y un segundo documento indicando las formas de entregar el informe (informe de extensión .PDF).

Nota 2: Se le proporcionará un código base para la realización de la evaluación.

Nota 3: Se realizará una revisión acuciosa para la detección de copias/plagios. Para evitar problemas, los estudiantes deben programar todo lo que se pide por sí mismos y escribir el informe en su totalidad. Las copias/plagios serán penalizadas(os).