Entrega final - Deep Learning

UNIVERSIDAD
DE ANTIOQUIA
1 8 0 3

WILMER ARLEY DE JESUS CEBALLOS HOYOS

MARCELO DE LA HOZ SIERRA

Facultad de Ingeniería

Universidad de Antioquia

Fundamentos de deep learning

Raul Ramos Pollan

19 de noviembre de 2023

Descripción de la estructura de los notebooks entregados

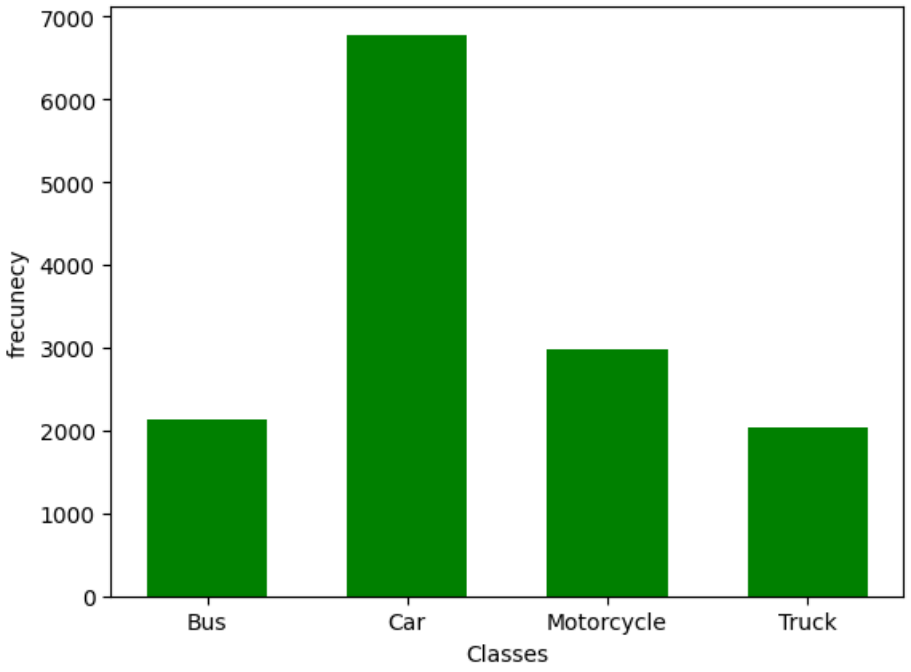
El notebook proporcionado contiene código en Python utilizando bibliotecas como TensorFlow y Keras para construir y entrenar modelos de redes neuronales convolucionales (CNN) para clasificación de imágenes. El código está estructurado en secciones, cada una con un propósito específico. Aquí hay una descripción general de las secciones:

1. **Carga y preprocesamiento de datos:**
    - Se descarga un archivo JSON de Kaggle con información sobre el conjunto de datos.
    - Se descarga y descomprime el conjunto de datos desde Kaggle, que contiene imágenes de vehículos divididas en clases como Car, Motorcycle, Truck y Bus.
    - Se realiza un análisis exploratorio de los datos, mostrando algunas imágenes aleatorias de cada clase y visualizando la frecuencia de cada clase.
2. **División de datos:**
    - Se utiliza la clase **ImageDataGenerator** de Keras para cargar y transformar imágenes, dividiendo el conjunto de datos en conjuntos de entrenamiento y validación.
3. **Definición de métricas y modelos:**
    - Se definen varias métricas, como precisión, recall, f1-score y matrices de confusión.
    - Se define un modelo de red neuronal convolucional (CNN) utilizando la API secuencial de Keras. Este modelo incluye capas convolucionales, capas de agrupación, capas completamente conectadas y capas de dropout.
4. **Entrenamiento y evaluación del modelo:**
    - El modelo se entrena utilizando el generador de datos de entrenamiento y se evalúa en el generador de datos de validación.
    - Se guarda el modelo entrenado en formato .h5.
    - Se genera y visualiza la matriz de confusión del modelo en el conjunto de validación.
5. **Visualización de la arquitectura del modelo:**
    - Se utiliza la biblioteca **visualkeras** para visualizar la arquitectura del modelo.
6. **Análisis de resultados:**
    - Se muestra la precisión y la pérdida del modelo a lo largo de las épocas mediante gráficos.
7. **Prueba del modelo con imágenes de validación:**

    - Se seleccionan algunas imágenes del generador de datos de validación y se realizan predicciones con el modelo.
8. **Segundo modelo CNN:**
    - Se define y entrena otro modelo de CNN similar al primero con algunas diferencias en la arquitectura.
9. **Modelo VGG16 preentrenado:**
    - Se utiliza la arquitectura del modelo VGG16 preentrenado en ImageNet para clasificación de imágenes.
    - Se compila, entrena y evalúa el modelo VGG16 en el conjunto de datos proporcionado.
10. **Visualización de resultados del modelo VGG16:**
    - Se muestra la matriz de confusión y se visualiza la arquitectura del modelo VGG16.
11. **Visualización de resultados:**
    - Se muestran gráficos de precisión y pérdida para el modelo VGG16.
12. **Prueba del modelo VGG16 con imágenes de validación:**
    - Se seleccionan algunas imágenes del generador de datos de validación y se realizan predicciones con el modelo VGG16.

Descripción de tu solución (arquitectura, preprocesado, etc.)

Específicamente en el notebook que se entregará se inicia descargando el dataset el cual contiene una gran cantidad de imágenes de vehículos de los cuales solo nos quedaremos con 4 clases: car, bus, truck y motorcycle. Serán un total de 13933

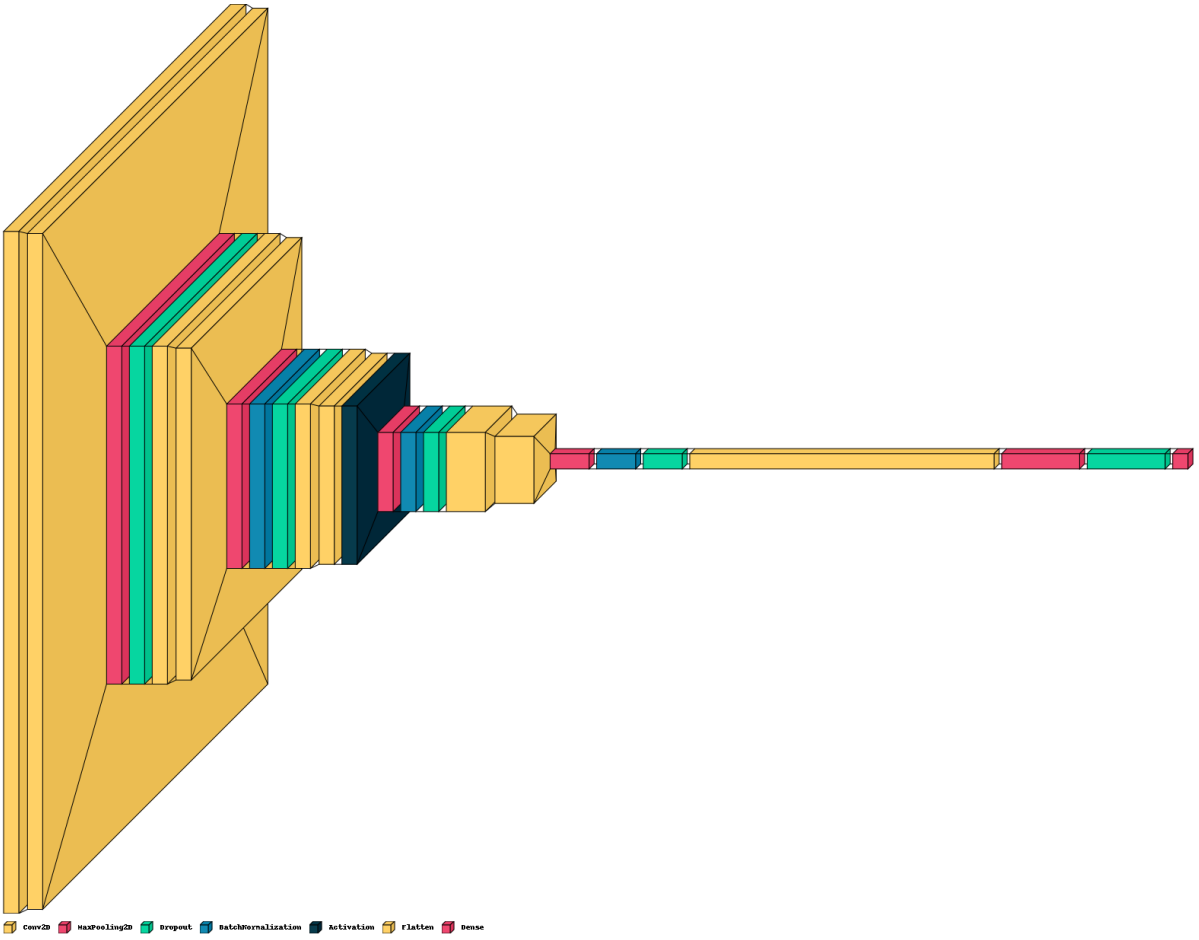


imágenes, las cuales se distribuirán de la siguiente manera:

En la siguiente parte del notebook se hará un pequeño preprocesamiento de los datos el cual será darle un tamaño de 224*224 píxeles a todas las imágenes para poder ser procesadas, además de esto se establecen las métricas que se usarán para evaluar los modelos, las cuales serán accuracy, loss, recall, f1.

En cuanto a las arquitecturas usadas se usaron 2 arquitecturas de CNN hechas por nosotros y una última de google llamada VGG 16 para comparar sus resultados.

Primer Modelo CNN:

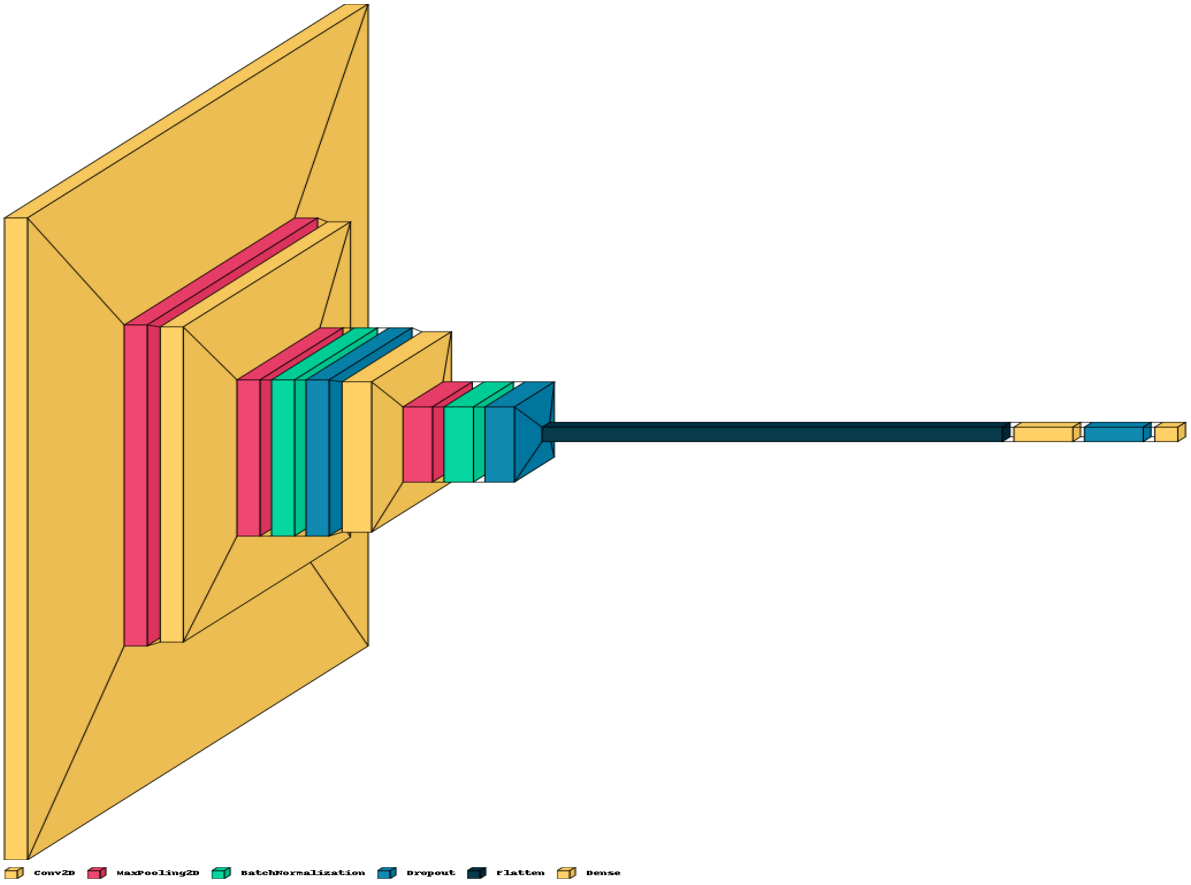


- **Arquitectura del Modelo:**
  - Varias capas convolucionales con funciones de activación ReLU.
    - Las cuales se usan para que la red aprenda las características en las capas iniciales y luego características más complejas en las capas posteriores, esto permite que el modelo aprenda a detectar patrones de las imágenes para diferenciar los vehículos.
  - Capas de agrupación máxima (max pooling).
    - Ayudan a lograr que el modelo detecte los patrones de la imagen a pesar de que estos no estén en la misma posición.

        - Además de esto también se encargan de reducir la dimensionalidad lo que reduce el número de parámetros en la red lo que ayuda a evitar el sobreajuste y puede ayudar a mejorar el tiempo de entrenamiento.
    - Capas de dropout para regularización.
        - Ayuda a que el modelo no tenga sobreajuste, dado que al apagar aleatoriamente algunas neuronas durante el entrenamiento permite que el modelo mejore su capacidad de generalización.
    - Capas completamente conectadas para la clasificación final.
        - permite que el modelo pueda utilizar lo aprendido para realizar la clasificación de la imágenes

- **Métricas y Entrenamiento:**
    - Definición de métricas como precisión, recall, f1-score y matrices de confusión.
    - Entrenamiento con 20 épocas y optimizador Nadam.

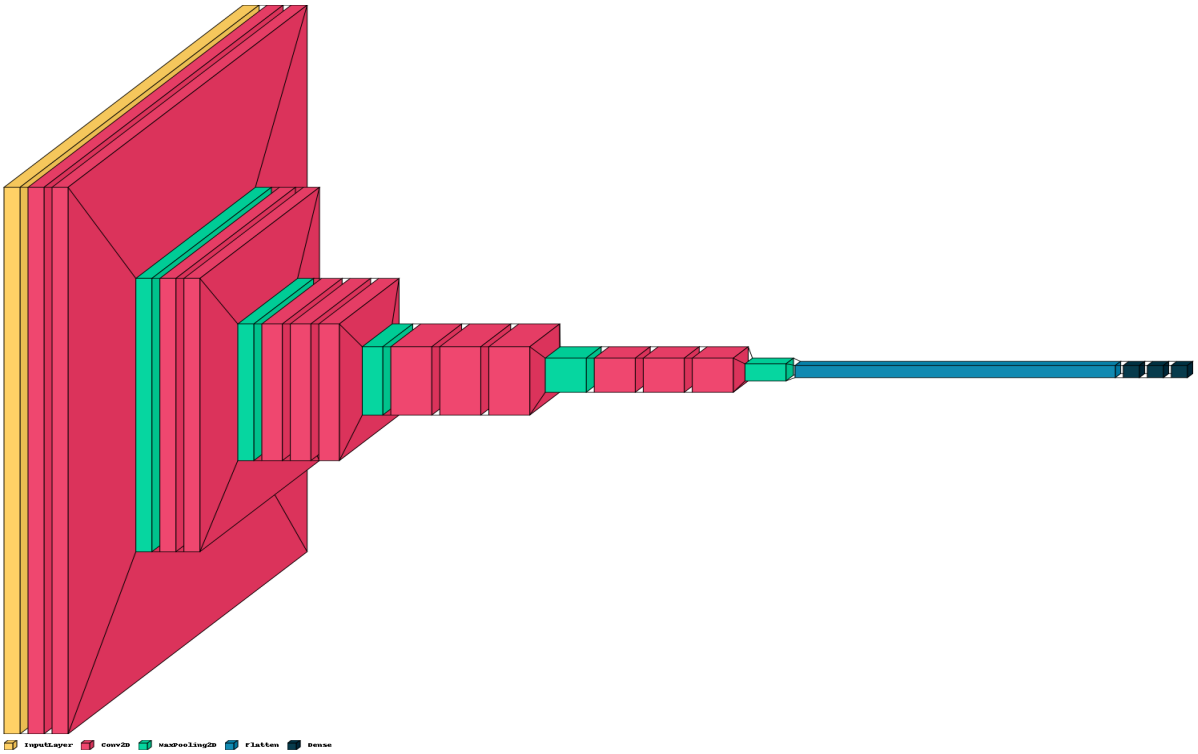
Segundo Modelo CNN:



- **Arquitectura del Modelo:**
    - Similar al primer modelo pero con algunas diferencias en la arquitectura, como el uso de capas BatchNormalization.

- **Preprocesamiento de Datos:**
    - Reutilización del mismo preprocesamiento que el primer modelo.
- **Métricas y Entrenamiento:**
    - Definición de las mismas métricas que el primer modelo.
    - Entrenamiento con 20 épocas y optimizador Adam.

Modelo VGG16 preentrenado:



- **Arquitectura del Modelo:**
    - Utilización del modelo VGG16 preentrenado en ImageNet.
    - Agregación de capas densas personalizadas para la clasificación.
- **Preprocesamiento de Datos:**
    - Reutilización del mismo preprocesamiento que el primer modelo.
- **Métricas y Entrenamiento:**
    - Uso de las mismas métricas que en los modelos anteriores.
    - Transferencia de aprendizaje con pesos congelados en las capas convolucionales de VGG16.

Evaluación y Visualización:

- **Matrices de Confusión:**
    - Utilización de matrices de confusión para evaluar el rendimiento del modelo en el conjunto de validación.
    - Visualización de las matrices de confusión utilizando la biblioteca Seaborn.
- **Visualización de Arquitectura:**

    - Uso de la biblioteca **visualkeras** para visualizar la arquitectura de cada modelo.
- **Gráficos de Precisión y Pérdida:**
    - Visualización de la precisión y pérdida a lo largo de las épocas para cada modelo.

Descripción de las Iteraciones:

Durante el proceso de desarrollo, se realizaron varias iteraciones para ajustar y mejorar el rendimiento de los modelos. A continuación, se detallan algunas de las decisiones clave tomadas durante estas iteraciones:

1. **Tamaño del Batch:**
    - Se comenzó utilizando un tamaño de lote (batch) de 32 imágenes durante el entrenamiento.
    - Después de algunas iteraciones, se decidió aumentar el tamaño del lote a 64. Esta modificación resultó en una reducción del tiempo de entrenamiento y una mejora en la suavidad de las curvas de precisión y pérdida.
2. **Número de Épocas:**
    - Se inició con un número inicial de 20 épocas para entrenar los modelos.
    - En un intento por mejorar el rendimiento, se aumentó el número de épocas a 30 en una iteración posterior.
    - Al observar que no se traducía en mejoras significativas y solo prolongaba el tiempo de entrenamiento, se decidió revertir a 20 épocas. Esta decisión se basó en la evaluación del rendimiento y en la ausencia de mejoras sustanciales.
3. **Creación de un Segundo Modelo:**
    - Considerando las limitaciones y errores identificados en el primer modelo, se optó por diseñar un segundo modelo de red neuronal convolucional (CNN).
    - El segundo modelo se diseñó con ajustes en la arquitectura, incorporando capas adicionales como BatchNormalization, con el objetivo de abordar las deficiencias percibidas en el primer modelo.
4. **Transferencia de Aprendizaje con VGG16:**
    - Se exploró la aplicación de transferencia de aprendizaje utilizando el modelo preentrenado VGG16 en ImageNet.
    - Esta iteración se centró en aprovechar el conocimiento previamente adquirido por VGG16 para mejorar el rendimiento en la tarea específica de clasificación de imágenes de vehículos.
5. **Análisis de Mejoras y Tiempo de Entrenamiento:**

    - Se realizó un análisis constante de las métricas de rendimiento, como precisión y pérdida, durante el entrenamiento y la validación.
    - La toma de decisiones se basó en buscar mejoras sustanciales en el rendimiento sin incurrir en un aumento significativo en el tiempo de entrenamiento.

Estas iteraciones reflejan un enfoque iterativo y reflexivo para ajustar los modelos y los parámetros, buscando constantemente un equilibrio entre el rendimiento y la eficiencia en términos de tiempo de entrenamiento.

Descripción de los Resultados

Resultados del Primer Modelo CNN:

- **Entrenamiento y Validación:**
    - El modelo se entrenó durante 20 épocas en el conjunto de entrenamiento y se validó en un conjunto separado.
    - Se observó una mejora en la precisión y la pérdida a lo largo de las épocas.
- **Matriz de Confusión:**
    - Se generó y visualizó una matriz de confusión en el conjunto de validación.
    - La matriz de confusión proporciona una visión detallada del rendimiento del modelo para cada clase.
- **Visualización de Arquitectura:**
    - Se utilizó **visualkeras** para visualizar la arquitectura del modelo.
    - Esto puede ayudar a comprender la estructura y las conexiones entre las capas del modelo.
- **Prueba con Imágenes de Validación:**
    - Se seleccionaron aleatoriamente algunas imágenes del conjunto de validación para realizar pruebas.
    - El modelo realizó predicciones y se compararon con las etiquetas reales.

Resultados del Segundo Modelo CNN:

- **Entrenamiento y Validación:**
    - Similar al primer modelo, se observó que la perdida presento un comportamiento muy malo, en el sentido de que no se apego a la curva genera en el entrenamiento; por otro lado el accuracy tambien presento un peor rendimiento.
- **Matriz de Confusión:**
    - Se generó y visualizó otra matriz de confusión para evaluar el rendimiento en el conjunto de validación.
- **Visualización de Arquitectura:**
    - Se utilizó **visualkeras** para visualizar la arquitectura del segundo modelo.

Resultados del Modelo VGG16 preentrenado:

- **Entrenamiento y Validación:**

    - Se aplicó transferencia de aprendizaje utilizando el modelo VGG16 preentrenado en ImageNet.
    - El modelo se entrenó durante 20 épocas en el conjunto de entrenamiento y se evaluó en el conjunto de validación.
- **Matriz de Confusión:**
    - Se generó y visualizó una matriz de confusión específica para evaluar el rendimiento del modelo VGG16.
- **Visualización de Arquitectura:**
    - Se utilizó **visualkeras** para visualizar la arquitectura del modelo VGG16.

Análisis General de Resultados:

- **Precisión y Pérdida a lo largo de las Épocas:**
    - Se graficaron las curvas de precisión y pérdida a lo largo de las épocas para cada modelo.
    - Esto proporciona información sobre la convergencia del modelo y su rendimiento en los conjuntos de entrenamiento y validación.
- **Matrices de Confusión:**
    - La evaluación con matrices de confusión permite identificar qué clases son más propensas a errores de predicción y proporciona información sobre el rendimiento general del modelo.
- **Visualización de Arquitectura:**
    - La visualización de la arquitectura ayuda a comprender la complejidad del modelo y cómo se conectan las diferentes capas.

Desafíos y Oportunidades de Mejora:

- Los desafíos comunes incluyen el sobreajuste, donde el modelo aprende demasiado los datos de entrenamiento y no generaliza bien a nuevos datos.
- Se podría explorar la posibilidad de ajustar hiperparámetros, probar diferentes arquitecturas de modelos o utilizar técnicas avanzadas como la regularización para mejorar el rendimiento.