Proceso de evaluación continuada (Punto 2)

Deep Learning:

Solución PEC 1

Estudiantes:

Iván Fernando Bermúdez López

Diego Luis Restrepo Urrea

Docente:

Sonia Jaramillo

Universidad del Quindío – Maestría en Ingeniería con énfasis en Análisis de Datos

Facultad de Ingenierías

1. *Use very Deep convolution para clasificar imágenes. Implemente un ejemplo que incluya data augmentation. Existen arquitecturas tales como ResNet 101 V2 y DenseNet 201, que pueden ser usadas. Describa en un párrafo la arquitectura de la infraestructura usada.*

***Solución:***

Para dar solución a este ítem, se presenta la siguiente metodología para abordar el problema. En ella se encuentran cada uno de los procesos realizados para lograr la clasificación de imágenes, teniendo como eje central la implementación de la arquitectura Mobilenetv2, la cual se seleccionó para dar solución a los requerimientos.

Diagrama

Descripción generada automáticamente

La metodología propuesta sigue un enfoque sistemático para desarrollar y entrenar un modelo de clasificación de imágenes, destacando la implementación de MobileNetV2 como la base del modelo. A continuación, se explica cada paso del diagrama con un enfoque en la integración de MobileNetV2:

1. **Importación de librerías:**

Se debe iniciar importando las librerías necesarias para todo el proceso. Esto incluye herramientas para la gestión de archivos y directorios (os, shutil), procesamiento de imágenes (numpy, PIL, cv2), visualización de datos (matplotlib), y especialmente para la construcción y entrenamiento del modelo (tensorflow, tensorflow\_hub), donde se encuentra la arquitectura MobileNetV2.

1. **Gestión de datos de imagen:**

Preparar los datos creando carpetas para cada categoría de imagen (banano, manzana, piña) y balancear las clases copiando la misma cantidad de imágenes en cada carpeta. Verificar que las cantidades sean correctas para asegurar que el modelo entrenará con datos balanceados, lo cual es esencial para evitar sesgos durante el entrenamiento.

1. **Generación y aumento de datos:**

Utilizar un generador de datos (ImageDataGenerator) para aplicar transformaciones a las imágenes, como rescalado, rotaciones, desplazamientos, zoom y corte. Esto no solo aumenta el tamaño del dataset, sino que también mejora la capacidad del modelo para generalizar. Dividir los datos en entrenamiento (80%) y validación (20%) para evaluar el desempeño del modelo en datos no vistos. También se realiza 70% y 30%, respectivamente.

1. **Construcción del modelo:**

En este paso, se integra la arquitectura MobileNetV2 desde TensorFlow Hub como la base del modelo. MobileNetV2 es una red neuronal convolucional pre-entrenada, optimizada para dispositivos móviles y aplicaciones que requieren eficiencia en términos de velocidad y tamaño.

Esta arquitectura es conocida por su capacidad para retener alta precisión con un número reducido de parámetros, lo que la hace ideal para ser usada en aplicaciones con recursos limitados. El modelo recibe imágenes de entrada de tamaño 224x224 con 3 canales (RGB). Se conecta una capa densa de salida con 3 neuronas y activación softmax para clasificar las imágenes en tres categorías: banano, manzana y piña. El modelo se compila con el optimizador Adam y la función de pérdida categorical\_crossentropy, que es adecuada para tareas de clasificación multiclase.

1. **Entrenamiento del modelo:**

Utilizar los generadores de datos para entrenar el modelo durante 50 épocas, valor que puede variar para observar diferentes comportamientos del modelo. La integración de MobileNetV2 permite aprovechar los pesos preentrenados de la red, lo que acelera el proceso de entrenamiento y mejora la precisión al inicio. Durante el entrenamiento, se registra la precisión y la pérdida en los conjuntos de entrenamiento y validación, lo cual ayuda a monitorear el rendimiento del modelo y a realizar ajustes si es necesario.

1. **Evaluación y gráficos:**

Graficar las métricas de precisión y pérdida para evaluar cómo se desempeña el modelo durante el entrenamiento y la validación. MobileNetV2 contribuye al rendimiento del modelo debido a su diseño eficiente, que equilibra bien la precisión y el tiempo de inferencia.

1. **Predicción y clasificación de imágenes externas:**

Implementar una función para procesar y clasificar imágenes descargadas desde URLs. MobileNetV2, con su capacidad para extraer características de alto nivel de las imágenes, facilita la predicción precisa de las clases, devolviendo 0 para banano, 1 para manzana y 2 para piña.

1. **Guardado y exportación del modelo:**

Guardar el modelo entrenado para su uso en TensorFlow Serving, lo cual permite implementar el modelo en aplicaciones de producción. Comprimir el modelo para facilitar su distribución. La integración de MobileNetV2 asegura que el modelo final sea eficiente y adecuado para despliegues en entornos con recursos limitados, como dispositivos móviles o aplicaciones web.