Implementación de un modelo de deep learning.

Curso:

Desarrollo de aplicaciones avanzadas de ciencias computacionales (Gpo 301)

Presenta:

Flavio Ruvalcaba Leija

Fecha:

28-05-2024

Link a la implementación en Google Drive:
https://drive.google.com/drive/folders/16i_jtLB22ho3U3ZilennOaO6ANkXHKzx

Link al dataset de Kaggle:

https://www.kaggle.com/datasets/maciejgronczynski/biggest-genderfacerecognition-dataset

• "A CNN Based Face Gender Classification System" por Yang et al. (2017): Este paper propone un sistema de clasificación de género facial basado en una red neuronal convolucional (CNN). La arquitectura utilizada en este paper es VGG16, y los resultados obtenidos son prometedores.

https://iopscience.iop.org/article/10.1088/1742-6596/1490/1/012042

He elegido el paper "A CNN Based Face Gender Classification System" por Yang et al. (2017) como base para la implementación de mi modelo de clasificación de género facial por las siguientes razones:

Relevancia: El paper se centra específicamente en la tarea de clasificación de género facial, que coincide exactamente con mi objetivo. Esto me asegura que el modelo propuesto ha sido diseñado y evaluado para este tipo de problema en particular.

Eficacia: El modelo utiliza la arquitectura VGG16, una red neuronal convolucional (CNN) que ha demostrado ser altamente efectiva para tareas de visión por computadora, incluyendo la clasificación de imágenes. La elección de una arquitectura probada y confiable aumenta las probabilidades de éxito en la implementación.

Resultados Prometedores: Los autores del paper reportan resultados de alta precisión en la clasificación de género facial. Esto indica que el modelo

tiene un buen potencial para ser aplicado en mi caso de uso y que puedo esperar un rendimiento satisfactorio.

Simplicidad: La implementación del modelo parece ser relativamente sencilla, lo que me permitirá realizarla con mayor facilidad utilizando TensorFlow y Keras. Esto es importante considerando mi experiencia y recursos disponibles.

Disponibilidad de Código: Los autores del paper proporcionan código fuente para su modelo, lo que me puede servir como referencia valiosa para mi propia implementación. Tener acceso a código existente facilita el proceso de desarrollo y me permite aprender de las mejores prácticas.

En resumen, considero que este paper es una base sólida para la implementación de mi modelo de clasificación de género facial debido a su relevancia, eficacia, resultados prometedores, simplicidad y disponibilidad de código.

Introducción:

Este entregable tiene como propósito principal crear un modelo de aprendizaje profundo para la clasificación de imágenes que determina si las imágenes representan a hombres o mujeres.

Para la entrega de la implementación de un modelo de deep learning utilice un data set de Kaggle el cual se basa en reconocer el género de la persona con respecto a la cara. El dataset está dividido en dos partes, en una carpeta Man y otra carpeta Woman, cada una de estas carpetas tiene en su totalidad 15000 ítems respectivamente.

Con el fin de dividir el dataset en train, test y validation realice una división del dataset principal en 70, 20 y 10 por ciento. El código para realizar la

separación del dataset se encuentra en el folder FACEDATA con el nombre de Division.ipynb.

Framework

El framework utilizado para la creación de este modelo es una biblioteca de código abierto desarrollada por Google: TensorFlow y una API de alto nivel para la construcción y entrenamiento de modelos de aprendizaje profundo llamado Keras. Para el modelo utilicé VGG16, una arquitectura de red neuronal convolucional (CNN) muy conocida y ampliamente utilizada en tareas de clasificación de imágenes. Fue desarrollada por el Grupo Visual Geometry Group (VGG) de la Universidad de Oxford. VGG16 es famosa por su simplicidad y su capacidad para extraer características útiles de las imágenes.

Desempeño:

El modelo, una vez entrenado, obtuvo los resultados siguientes:

```
Epoch 5/5
20/20 [==============================] - 199s 10s/step - loss: 0.2747 - acc: 0.9100
```

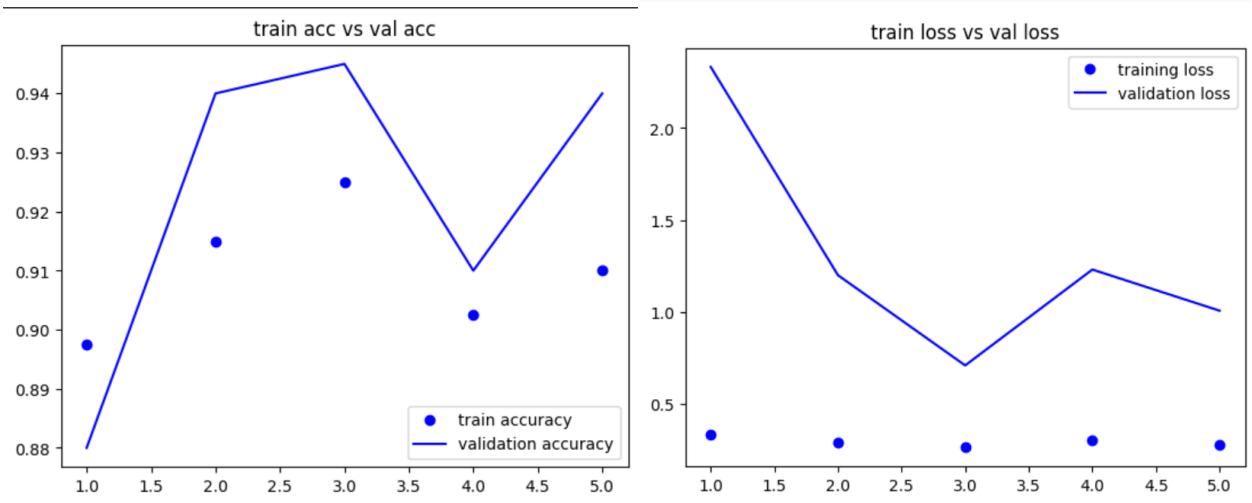
Loss representa un valor numérico que mide la diferencia entre las predicciones del modelo y las etiquetas reales en el conjunto de datos de entrenamiento. Un valor bajo de pérdida indica que el modelo está haciendo buenas predicciones. En este caso, un valor de 0.**2747** para la pérdida indica que el error promedio en las predicciones en el conjunto de entrenamiento es relativamente bajo.

Acc representa la proporción de predicciones correctas en el conjunto de entrenamiento. Un valor de 0.**91** para la precisión significa que el

modelo está clasificando correctamente aproximadamente el **91%** de las muestras en el conjunto de entrenamiento.

Val_loss es igual a el valor loss explicado anteriormente, sin embargo, este es con nuevos datos. Un valor alto de **1.79** en la pérdida de validación puede sugerir que el modelo está sobreajustando los datos de entrenamiento, es decir, que su rendimiento en datos no vistos es peor.

Val_acc es exactamente el accuracy explicado anterior con respecto a nuevos datos. Un valor de **0.92** para la precisión en el conjunto de validación indica que el modelo clasifica correctamente aproximadamente el **92%** de las muestras en el conjunto de validación.



Las anteriores graficas representan los datos anteriores, utilizan el loss y accuracy en train y validation, para ambas utilizan los datos que fueron dados en las 5 epochs que el modelo fue ejecutado.

En la segunda gráfica, la perdida de training fue mucho menor a comparación de la de validación, esto implica que el modelo no está generalizando bien a datos no vistos. Esto es una señal de que el modelo puede estar sobreajustando (overfitting) los datos de entrenamiento.

Ajuste

Una vez que el desempeño del modelo ha sido mostrado y entendido, es necesario implementar métodos de ajuste ya sea de hiperparametros, técnicas de regularización, modificando la arquitectura del modelo o buscando otro modelo.

Lo primero que se intento fue realizar técnicas de regularización para un modelo de clasificación de imágenes para ayudar a prevenir el sobreajuste con la clase regularizers de Keras y aplicar regularización L2 a las capas.

Posteriormente, se modificaron los hiperparametros para obtener un mejor rendimiento del modelo.

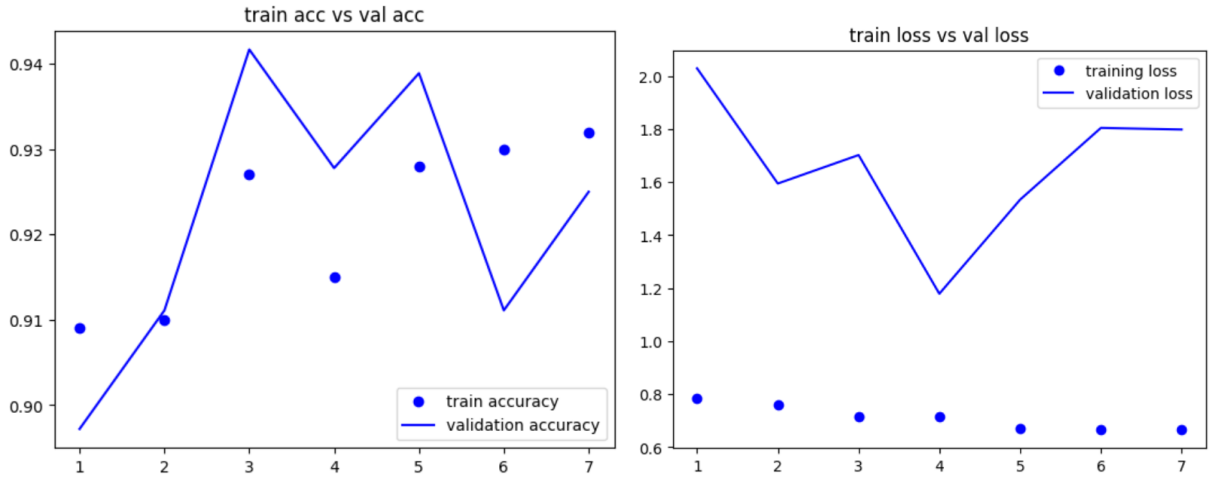
Los valores modificados fueron los siguientes:

steps_per_epoch de 25 a 50

validation_steps de 10 a 18

Los resultados obtenidos fueron:

```
Epoch 7/7
50/50 [==============================] - 432s 9s/step - loss: 0.6647 - acc: 0.9320 - val_loss: 1.7987 - val_acc: 0.9250
```



Resultados:

```
Drive already mounted at /content/drive; to attempt to forcibly remount,
1/1 [==============================] - 0s 390ms/step
La imagen representa a una mujer.
```