Implementación de un modelo de deep learning.

Curso:

Inteligencia artificial avanzada

para la ciencia de datos II

Presenta:

Flavio Ruvalcaba Leija

Fecha:

11-05-2023

Repositorio Github:
https://github.com/FlavioRr/InteligenciaArtificialGpo101

Link al dataset de Kaggle:
https://www.kaggle.com/datasets/maciejgronczynski/biggest-genderface-recognition-dataset

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 repositorio con el nombre de divisionDataset.py.

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

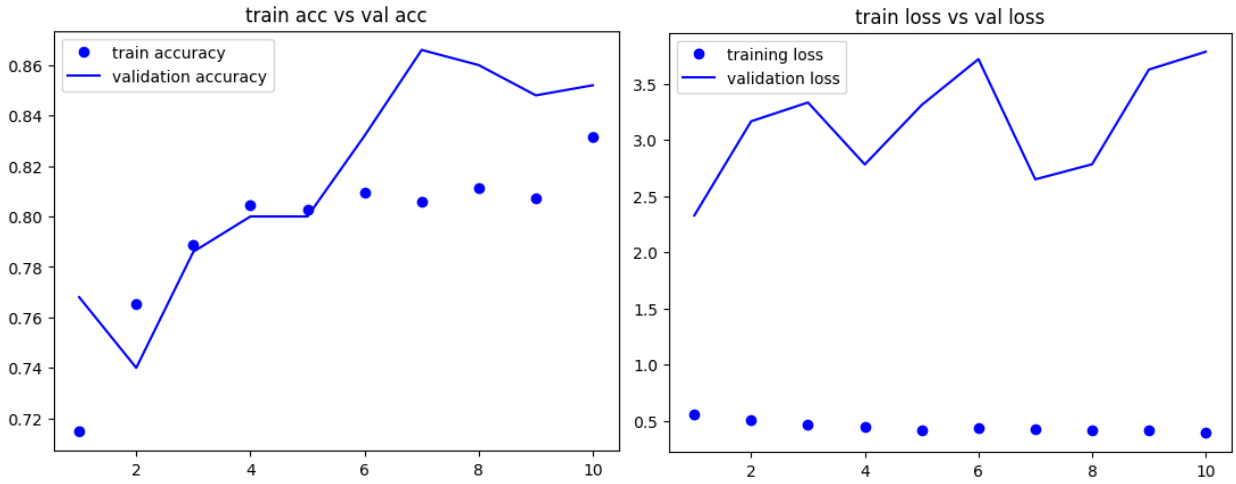
El modelo, una vez entrenado, obtuvo los resultados siguientes:

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 **0.3928** 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 **0.8313** para la precisión significa que el modelo está clasificando correctamente aproximadamente el 83.13% 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 **3.7855** 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 **0.8520** para la precisión en el conjunto de validación indica que el modelo clasifica correctamente aproximadamente el 85.20% 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 10 epochs que el modelo fue ejecutado.

Como se puede observar en el accuracy de train y de validation ambas se comportaron de manera muy parecidas en las primas seis épocas, sin embargo, hubo un pico en las siguientes que mejoro en la accuracy de validation.

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.

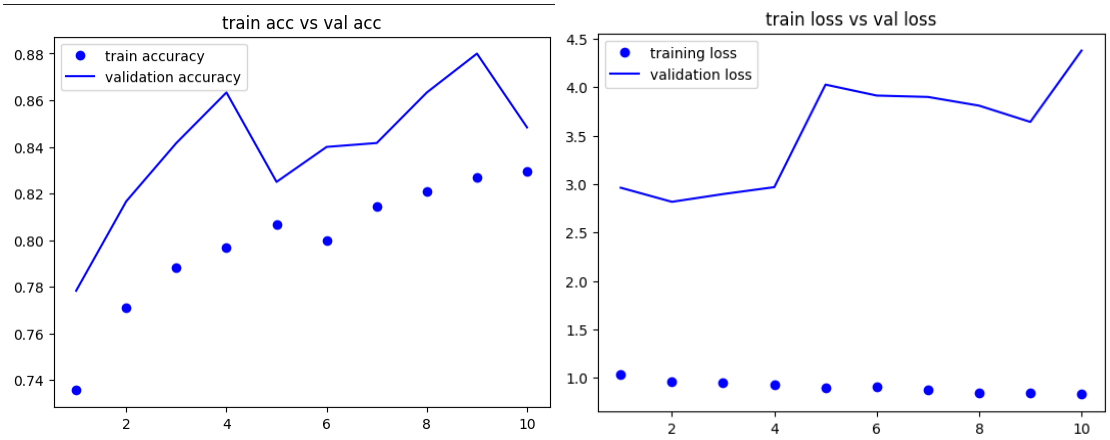
Los valores modificados fueron los siguientes:

steps_per_epoch de 75 a 80

validation_steps de 25 a 30

Los resultados obtenidos fueron:

```
loss: 0.8404 - acc: 0.8269 - val_loss: 3.6407 - val_acc: 0.8800
```



Como podemos observar con los resultados y las gráficas, con la técnica de regularización y modificación de hiperparametros obtuvimos una mejora considerable en el área de accuracy aumentando un 3 por ciento en validation, de la misma manera, el valor de perdida disminuyo un 0.1 para validation.

Por otra parte, en cuanto a train, el valor de perdida aumento considerablemente y el valor de accuracy disminuyo un porciento.

Ejecución

Una vez que el modelo se desempeña satisfactoriamente, procedemos a resguardarlo con un model.save(). El modelo, una vez guardado podrá ser ejecutado con el archivo Python llamado modelTest.py. Lo que este archivo hace es que con nuestro modelo entrenado predice si una foto es de un hombre o de una mujer.

Ejemplo:

```
2023-11-05 17:02:34.522685: I tensorflow/core/platfor
To enable the following instructions: SSE SSE2 SSE3 S
1/1 [==============================] - 0s 212ms/step
La imagen representa a un hombre.
```