Instituto Tecnológico y de Estudios Superiores de Monterrey

Campus Monterrey

"Yo, como integrante de la comunidad estudiantil del Tecnológico de Monterrey, soy consciente de que la trampa y el engaño afectan mi dignidad como persona, mi aprendizaje y mi formación, por ello me comprometo a actuar honestamente, respetar y dar crédito al valor y esfuerzo con el que se elaboran las ideas propias, las de los compañeros y de los autores, así como asumir mi responsabilidad en la construcción de un ambiente de aprendizaje justo y confiable"

"Inteligencia artificial avanzada para la ciencia de datos I"

Momento de Retroalimentación Individual:
Implementación de un modelo de Deep Learning

Daniel Saldaña Rodríguez A00829752

Fecha de entrega: 3 de noviembre de 2023

El código creado en este trabajo tiene como objetivo principal construir y entrenar un modelo de clasificación de imágenes utilizando la técnica de Transfer Learning. La idea principal detrás de Transfer Learning es utilizar modelos pre-entrenados en grandes conjuntos de datos para tareas de clasificación de imágenes y adaptarlos a tareas específicas con conjuntos de datos más pequeños y especializados. En este caso, se emplea el modelo VGG16, pre-entrenado en el conjunto de datos ImageNet, como punto de partida para la clasificación de imágenes.

Importación de Dependencias y Carga de Datos

Comenzamos importando las bibliotecas necesarias, como NumPy, Matplotlib, y Keras, para trabajar con modelos de aprendizaje profundo y manipular datos de imágenes. Además, utiliza la biblioteca zipfile para extraer datos de un archivo comprimido, olivetti_faces.npy.zip, que contiene imágenes y etiquetas de rostros humanos. Esta base de datos proviene de Kaggle y puede ser encontrada en la siguiente liga: https://www.kaggle.com/code/serkanpeldek/face-recognition-on-olivetti-dataset

A continuación, se extraen las imágenes y etiquetas del archivo comprimido y se organizan en directorios según su clase, lo que facilita el manejo de los datos.

Preparación de Datos

Para garantizar que los datos estén listos para la alimentación del modelo, se utilizan las capacidades de aumento de datos de Keras. El aumento de datos implica aplicar transformaciones como rotación, zoom y volteo horizontal a las imágenes de entrenamiento para aumentar la diversidad de los datos y mejorar la capacidad del modelo para generalizar.

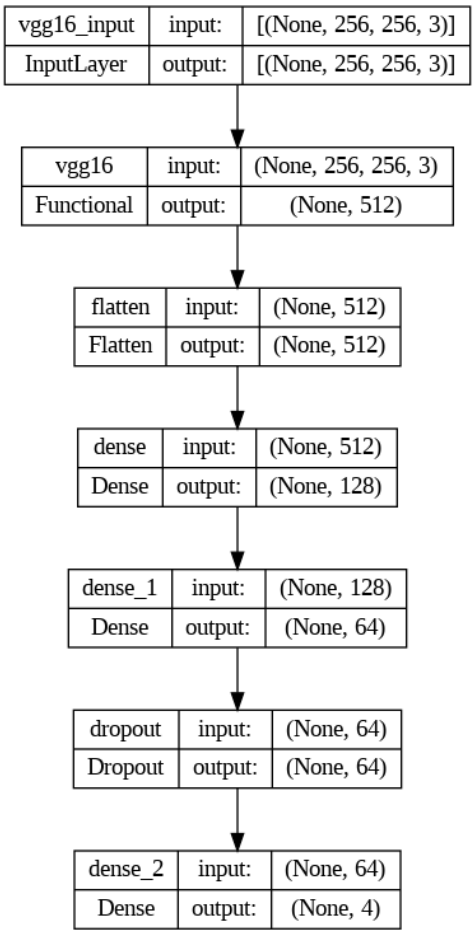
El conjunto de datos se divide en conjuntos de entrenamiento y validación utilizando ImageDataGenerator, que carga y preprocesa las imágenes de acuerdo con la configuración específica.

Definición y Entrenamiento del Modelo

El código define un modelo de clasificación de imágenes mediante la utilización del modelo pre-entrenado VGG16 como base. Se congela la base pre-entrenada para evitar que sus pesos se modifiquen durante el entrenamiento, lo que es una característica clave del Transfer Learning.

Luego, se agrega una capa de aplanamiento (Flatten) seguida de capas densas (Fully Connected) con funciones de activación ReLU y una capa de salida con activación softmax. Este último modelo se compila con una función de

pérdida de entropía cruzada categórica y el optimizador Adam.

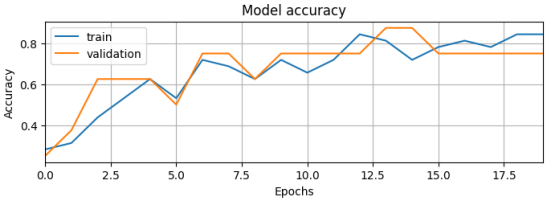


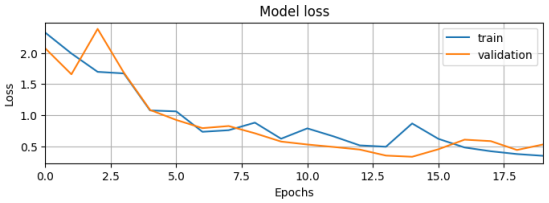
El entrenamiento del modelo se realiza a lo largo de 20 épocas, con un control de punto de control para guardar el mejor modelo en función de la precisión en el conjunto de validación. Esto asegura que el modelo final sea el mejor posible en términos de rendimiento de clasificación.

Evaluación del Modelo

Después del entrenamiento, el código carga el mejor modelo y lo utiliza para realizar predicciones en un conjunto de validación. Luego, calcula métricas de rendimiento como la precisión, la recuperación y el puntaje F1 para evaluar el desempeño del modelo en la clasificación de imágenes.

Resultados





Podemos ver que la accuracy del modelo es alta y la pérdida es baja, señales positivas.

Por su parte las otras métricas de evaluación nos dan los siguientes resultados para las clases que creamos:

	precisión	recall	f1
1	0.67	1.00	0.80
2	1.00	1.00	1.00
3	1.00	0.50	0.67
avg	0.92	0.88	0.87

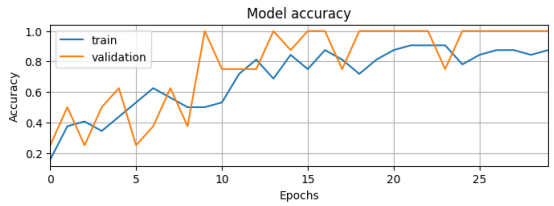
Resultados que igualmente son altos

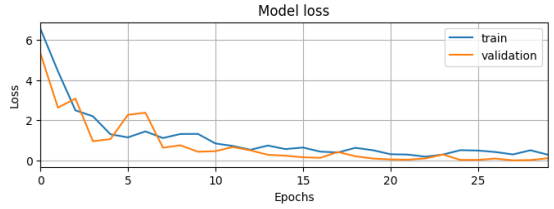
Cambios personales

Como contribución personal al proyecto, hice cambios a la arquitectura con el objetivo de mejorar el rendimiento de

nuestra red neuronal. Modifiqué las capas densas para tener un total de tres, con dimensiones más altas. También cambié el dropout de 0.15 a 0.5. Adicionalmente aumenté el número de épocas a 30.

Una vez aplicadas estas modificaciones obtuve los siguientes resultados:





	precisión	recall	f1
1	1.00	1.00	1.00
2	1.00	1.00	1.00
3	1.00	1.00	1.00
avg	1.00	1.00	1.00

Se obtuvo un 100% en todas las métricas analizadas, lo cual son noticias engañosas, ya que aunque paree bueno a primera vista, podría indicar que nuestro modelo está sobreentrenado.

Predicción con consola

También existe la habilidad de que el usuario introduzca nuevas fotos para que sean evaluadas por el modelo y recibir una predicción. Para esto el usuario debe de introducir manualmente la ruta de la imagen que desea que se evalúe y el programa imprimirá la clase predicha.

Ejemplos:

```
Ingrese la ruta de la imagen de prueba: ejemplo.png
1/1 [==============================] - 0s 21ms/step
Clase predicha: 0
```

```
Ingrese la ruta de la imagen de prueba: ejemplo2.png
1/1 [==============================] - 0s 18ms/step
Clase predicha: 2
```

Se puede ver como se introducen dos archivos diferentes a través de la terminal y se recibe su clase.

Conclusión

Nuestro código demuestra cómo construir y entrenar un modelo de clasificación de imágenes utilizando Transfer Learning. Al aprovechar un modelo pre-entrenado como punto de partida, se acelera significativamente el proceso de entrenamiento y se logra un rendimiento sólido en la tarea de clasificación de imágenes. La capacidad de transferir el conocimiento de modelos pre-entrenados a tareas específicas es una habilidad poderosa que nos permite abordar problemas complejos de manera efectiva.