Proyecto de Fundamentos de Deep Learning

Informe Proyecto Final

Cristian Camilo Julio Mejía

Ingeniería de Sistemas

Universidad de Antioquia

Fundamentos de Deep Learning

Raúl Ramos Pollan

noviembre

2023

Descripción de la estructura de los Notebooks entregados

El proyecto de clasificación e identificación de imágenes de flores, utilizando redes convolucionales consta de 3 entornos virtuales, estructurados de la siguiente manera:

1.1. *Exploración de Datos*: este notebook cuenta en primera instancia con la importación de las diferentes librerías utilizadas en el proyecto, además de la conexión a Google Drive en donde se encuentran alojadas las imágenes para la ejecución del mismo, seguidamente se cargan las imágenes y etiquetas al entorno virtual (Google Colaboraty) para su procesamiento, y de esta manera crear un primer modelo básico de redes neuronales convolucionales (CNN) para la clasificación de los objetos, y así preparar, realizar pruebas y predicciones del modelo.

1.2. *Preprocesado*: además de las secciones anteriores, (Exploración de Datos) en este apartado, se crea un generador de datos aumentado en donde se normaliza los valores de los pixeles de las imágenes (0, 1), se preparan los datos, tanto para el entrenamiento como para la validación, además, se busca crear versiones modificadas de las imágenes para evaluarlas en diferentes ítems, como rotaciones, cambios de escala y volteos; también, se visualizan los datos con su etiquetado, se guarda el modelo y nuevamente se realizan predicciones en donde se desea comprobar si son correctas o no.

1.3. *Arquitectura de línea base*: básicamente en este notebook se definieron, compilaron, entrenaron, evaluaron y se realizaron predicciones de 4 modelos (baseline, cnn_deeper, cnn_regularized y transfer_learning) para que finalmente comparar sus resultados según las métricas dadas (Precisión, Exhaustividad y Puntaje F1) y de este modo determinar el modelo más adecuado para la ejecución del proyecto.

2. **Descripción de la solución:** como lo que se busca es predecir un tipo de flor determinada (margarita, diente de león, rosa, girasol, tulipán), a través de la implementación de redes neuronales convolucionales, se desarrollaron cuatro (4) arquitecturas diferentes (baseline, cnn_deeper, cnn_regularized y transfer_learning) para dar solución a ello. En primera instancia, se realizó un modelo de redes convolucionales básico, en donde se configuraron dos capas de agrupación máxima con su capa convolucional cada una, además se estableció una capa de aplanamiento y dos capas densas para la salida del modelo. Luego, se creó una red convolucional más profunda, con una arquitectura similar a la básica, solo que en este caso se agregaron dos capas convolucionales con su respectiva capa de agrupación. Seguidamente, se interactuó con un modelo regularizado, en el cual se ajustó el Kernel en cada capa convolucional para prevenir el sobreajuste y finalmente, se creó un modelo con transferencia de aprendizaje utilizando VGG16. La selección del modelo más adecuado dependerá de las métricas de evaluación utilizadas (Accuracy, Precision, Recall, F1 Score y AUC) y de los resultados obtenidos.

3. Descripción de las iteraciones

Importando librerías y accediendo a los datos

```
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense

# Montar Google Drive (esto es necesario para acceder al conjunto de datos)
from google.colab import drive
drive.mount('/content/drive')
```

Fig. 1.0. Importación de librerías

Cargando imágenes

```
# Rutas a las carpetas de entrenamiento, validación y prueba
train_dir = '/content/drive/MyDrive/UDEA/2023-02/FundamentosDeepLearning/Proyecto/train'
val_dir = '/content/drive/MyDrive/UDEA/2023-02/FundamentosDeepLearning/Proyecto/val'
test_dir = '/content/drive/MyDrive/UDEA/2023-02/FundamentosDeepLearning/Proyecto/test'
```

Fig. 2.0. Accediendo a la información

Preprocesamiento de imágenes

```
data_gen = ImageDataGenerator(
    rescale=1./255,  # Normalizar los valores de los píxeles a [0,1]
    rotation_range=40,  # Rango de grados para rotaciones aleatorias
    width_shift_range=0.2,  # Rango de cambio aleatorio en el ancho de la imagen
    height_shift_range=0.2,  # Rango de cambio aleatorio en la altura de la imagen
    shear_range=0.2,  # Rango para las transformaciones de corte
```

Fig. 3.0. Creado generador aumentado

```
train_data = data_gen.flow_from_directory(
    '/content/drive/MyDrive/UDEA/2023-02/FundamentosDeepLearning/Proyecto/train',
    target_size=(150, 150),
    batch_size=32,
    class_mode='categorical',
    subset='training'
)
```

Fig. 4.0. Preparando generador

Creación de un primer modelo básico

```
model = Sequential()
model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(128, 128, 3)))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dense(len(labels), activation='softmax'))

model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

```

Fig. 5.0. CNN básico (Baseline)

Creación de una red convolucional más profunda

```
'cnn_deeper': tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(32, (3,3), activation='relu', input_shape=(150, 150, 3)),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Conv2D(128, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(5, activation='softmax')  # 5 tipos de flores
]),
```

Fig. 6.0. CNN Deeper

Creación de una red convolucional regularizada

```
'cnn_regularized': tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(32, (3,3), activation='relu', kernel_regularizer=regularizers.l2(0.01), input_shape
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Conv2D(64, (3,3), activation='relu', kernel_regularizer=regularizers.l2(0.01)),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(128, activation='relu', kernel_regularizer=regularizers.l2(0.01)),
    tf.keras.layers.Dense(5, activation='softmax')  # 5 tipos de flores
]),
```

Fig. 7.0. CNN Regularized

Creación de un modelo con transferencia de aprendizaje

```
  'transfer_learning': tf.keras.models.Sequential([
      VGG16(weights='imagenet', include_top=False, input_shape=(150, 150, 3)),
      tf.keras.layers.Flatten(),
      tf.keras.layers.Dense(128, activation='relu'),
      tf.keras.layers.Dense(5, activation='softmax')  # 5 tipos de flores
  ]),
```

Fig. 8.0. Modelo VGG16

4. Descripción de los resultados

Model	Accuracy	Precision	Recall	F1 Score	AUC
baseline	0.625000	0.738636	0.507812	0.601852	0.874451
cnn_deeper	0.625000	0.710000	0.554688	0.622807	0.900261
cnn_regularized	0.500000	0.510204	0.195312	0.282486	0.781487
transfer_learning	0.554688	0.756757	0.437500	0.554455	0.850334

Fig. 9.0. Resumen de los modelos

Según los resultados obtenidos en la imagen (Fig. 9.0) el modelo que arrojó las mejores métricas, fue cnn_deeper, equivalente a la red convolucional más profunda, el cual tiene una exactitud de 63%, una precisión de 71%, una sensibilidad de 55% un F1 de 62% y AUC de 90% lo que indica que este modelo tiene una buena capacidad para distinguir entre diferentes clases de flores. En general, estos resultados sugieren que el modelo de CNN más profunda podría ser una opción efectiva para esta tarea de clasificación de imágenes.

Resultado de Predicciones

```python
from tensorflow.keras.preprocessing import image
import numpy as np

# Carga una imagen desde un archivo
img = image.load_img('/content/drive/MyDrive/UDEA/2023-02/FundamentosDeepLearning/Proyecto/IMG/Vn.jpg', t

# Convierte la imagen en un array de numpy
img_array = image.img_to_array(img)

# Expande las dimensiones para que se ajuste al formato de entrada del modelo (1, alto, ancho, canales)
img_batch = np.expand_dims(img_array, axis=0)

# Normaliza la imagen dividiendo por 255
img_batch /= 255.

# Haz predicciones en la imagen
for name, model in models.items():
    predictions = model.predict(img_batch)
    predicted_class = np.argmax(predictions[0])
    print(f'El modelo {name} predice la clase {predicted_class} para la imagen nueva.')
```

Fig. 10. Realizando Predicciones

Etiquetado

Tipo de Flor	Margarita	Diente de León	Rosa	Girasol	Tulipán
Clase	0	1	2	3	4

Predicción # 1

Entrada: Diente de León (1), los modelos predijeron bien

Resultados:

```
1/1 [==============================] - 0s 90ms/step
El modelo baseline predice la clase 1 para la imagen nueva.
1/1 [==============================] - 0s 73ms/step
El modelo cnn_deeper predice la clase 1 para la imagen nueva.
1/1 [==============================] - 0s 49ms/step
El modelo cnn_regularized predice la clase 0 para la imagen nueva.
1/1 [==============================] - 0s 55ms/step
El modelo transfer_learning predice la clase 1 para la imagen nueva.
```

Fig. 11. Predicción imagen Diente de León

Predicción # 2

Entrada: Girasol (3), solo el modelo cnn_deeper, predijo bien

Resultados:

```
1/1 [==============================] - 0s 21ms/step
El modelo baseline predice la clase 1 para la imagen nueva.
1/1 [==============================] - 0s 19ms/step
El modelo cnn_deeper predice la clase 3 para la imagen nueva.
1/1 [==============================] - 0s 16ms/step
El modelo cnn_regularized predice la clase 1 para la imagen nueva.
1/1 [==============================] - 0s 17ms/step
El modelo transfer_learning predice la clase 4 para la imagen nueva.
```

Fig. 12. Predicción Girasol

Predicción # 3

Entrada: Tulipán (4), los modelos predijeron de manera incorrecta

Resultados:

```
1/1 [==============================] - 0s 64ms/step
El modelo baseline predice la clase 3 para la imagen nueva.
1/1 [==============================] - 0s 88ms/step
El modelo cnn_deeper predice la clase 1 para la imagen nueva.
1/1 [==============================] - 0s 53ms/step
El modelo cnn_regularized predice la clase 1 para la imagen nueva.
1/1 [==============================] - 0s 112ms/step
El modelo transfer_learning predice la clase 0 para la imagen nueva.
```

Fig. 13. Predicción Tulipán

Predicción # 4

Entrada: Rosa (2) solo el modelo con transferencia de aprendizaje, no predijo bien

Resultados:

```
1/1 [==============================] - 0s 17ms/step
El modelo baseline predice la clase 2 para la imagen nueva.
1/1 [==============================] - 0s 19ms/step
El modelo cnn_deeper predice la clase 2 para la imagen nueva.
1/1 [==============================] - 0s 17ms/step
El modelo cnn_regularized predice la clase 2 para la imagen nueva.
1/1 [==============================] - 0s 19ms/step
El modelo transfer_learning predice la clase 4 para la imagen nueva.
```

Fig. 14. Predicción Rosa

5. Conclusiones

Las redes neuronales convolucionales son modelos útiles y necesarios a la hora de realizar predicciones sobre clasificación de imágenes. Son una herramienta adecuada para la simulación y evaluación de los procesos; gracias a estas podemos tomar decisiones mas asertivas y hacer que nuestros negocios aumenten su rendimiento y reduzca el desperdicio.

Con las redes neuronales convolucionales no se buscan modelos perfectos, sino, arquitecturas útiles que nos ayude a la realización de nuestros proyectos o al crecimiento del negocio de una manera consciente y apropiada.

Queda evidenciado que entre mas se entrene un modelo, mejores resultados se van adquirir, teniendo presente lo que se desea de una manera clara y bien definida. Es fundamental la claridad de lo que se desea para no estar divagando en esfuerzos innecesarios.

6. Obtención de Datos

Los datos fueron obtenidos a través de una base de datos establecida en Kaggle por medio del siguiente enlace: https://www.kaggle.com/datasets/alxmamaev/flowers-recognition

7. Referencia

Hiary, H. &. (24 de Abril de 2018). *Researchgate*. Obtenido de Researchgate: https://www.researchgate.net/publication/324478963_Flower_Classification_using_Deep_Convolutional_Neural_Networks

MAMAEV, A. (25 de Abril de 2021). *Kaggle*. Obtenido de Kaggle: https://www.kaggle.com/datasets/alxmamaev/flowers-recognition

Neda Alipour, O. T. (19 de Mayo de 2021). *Ieeexplore*. Obtenido de Ieeexplore: https://ieeexplore.ieee.org/document/9443129/authors#authors

Yuanyuan Liu, F. T. (15 de Marzo de 2016). *Ieeexplore*. Obtenido de Ieeexplore: https://ieeexplore.ieee.org/document/7818296/authors#authors