2-9-2020

SISTEMAS INTELIGENTES 2

PROYECTO FINAL

BRYAN ARROYAVE, JORGE PIEDRAHITA, WILLIAM VASQUEZ
UNIVERSIDAD DE CALDAS

PROYECTO INTELIGENTES 2

Las redes neuronales convolucionales son un algoritmo de Deep Learning que esta diseñado para trabajar con imágenes, tomando estas como input, asignadonle importancias (pesos) a ciertos elementos en la imagen y de este modo poder diferenciar unas de otras.

Las redes convolucionales contienen varias hidden layers, donde las primeras puedan detectar líneas, curvas y así se van especializando hasta poder reconocer formas complejas como un rostro, siluetas, etc. Las tareas comunes de este tipo de redes son: *Detección o categorización de objetos, clasificación de escenas y clasificación de imágenes en general.*

La convolución es uno de los procesos más distintivos de estas redes son las convoluciones. El cual consiste en tomar un grupo de píxeles de la imagen de entrada e ir realizando un producto escalar con un kernel. El kernel recorrerá todas las neuronas de entrada y obtendremos una nueva matriz, la cual será una de las hidden layers. En el caso de que la imagen sea de color se tendrán 3 kernels del mismo tamaño que se sumarán para obtener una imagen de salida.

Para este proyecto hicimos uso de lo mencionado anteriormente para realizar una red convulocional que diferenciara entre los diferentes personajes de la famosa serie de los simpson. Posterior a esto generamos un servicio REST que recibiera como parametro una imagen, dicha imagen es enviada al metodo predict para obtener una respuesta de nuestro modelo previamente generado.

Para obtener un modelo capaz de realizar predicciones, fue necesario realizar diversas pruebas con diferentes configuraciones, a continuacion, se presentan algunas de las configuraciones realizadas.

2 capas. 15 épocas y 500 pasos.

```
500/500 [==============================] - 381s 761ms/step - loss: 1.1856 - accuracy: 0.6715
Epoch 11/15
500/500 [==============================] - 362s 723ms/step - loss: 1.1400 - accuracy: 0.6793
Epoch 12/15
500/500 [==============================] - 362s 724ms/step - loss: 1.0996 - accuracy: 0.6955
Epoch 13/15
500/500 [==============================] - 361s 721ms/step - loss: 1.0743 - accuracy: 0.6947
Epoch 14/15
500/500 [==============================] - 369s 738ms/step - loss: 1.0539 - accuracy: 0.7039
Epoch 15/15
500/500 [==============================] - 364s 727ms/step - loss: 1.0034 - accuracy: 0.7135

D:\Universidad\Semestre 10\Inteligentes 2\proyecto-inteligentesII>
```

```
284 / 990 = 28.68686868686869 %
```

2 capas. 20 épocas y 600 pasos.

```
600/600 [==============================] - 506s 843ms/step - loss: 0.9124 - accuracy: 0.7392
Epoch 16/20
600/600 [==============================] - 530s 883ms/step - loss: 0.8922 - accuracy: 0.7445
Epoch 17/20
600/600 [==============================] - 521s 868ms/step - loss: 0.8792 - accuracy: 0.7501
Epoch 18/20
600/600 [==============================] - 673s 1s/step - loss: 0.8386 - accuracy: 0.7583
Epoch 19/20
600/600 [==============================] - 542s 904ms/step - loss: 0.8142 - accuracy: 0.7614
Epoch 20/20
600/600 [==============================] - 513s 854ms/step - loss: 0.7953 - accuracy: 0.7677
```

```
386 / 990 = 38.98989898989899 %
```

3 capas. 15 épocas y 500 pasos

```
500/500 [==============================] - 407s 814ms/step - loss: 1.1720 - accuracy: 0.6750
Epoch 11/15
500/500 [==============================] - 381s 761ms/step - loss: 1.1357 - accuracy: 0.6816
Epoch 12/15
500/500 [==============================] - 360s 721ms/step - loss: 1.1057 - accuracy: 0.6888
Epoch 13/15
500/500 [==============================] - 358s 716ms/step - loss: 1.0303 - accuracy: 0.7061
Epoch 14/15
500/500 [==============================] - 359s 718ms/step - loss: 1.0067 - accuracy: 0.7172
Epoch 15/15
500/500 [==============================] - 359s 719ms/step - loss: 0.9802 - accuracy: 0.7223
```

```
277 / 990 = 27.979797979797983 %
```

2 capas. 30 épocas y 500 pasos

```
500/500 [==============================] - 691s 1s/step - loss: 0.8169 - accuracy: 0.7622
Epoch 26/30
500/500 [==============================] - 810s 2s/step - loss: 0.7913 - accuracy: 0.7703
Epoch 27/30
500/500 [==============================] - 628s 1s/step - loss: 0.7970 - accuracy: 0.7669
Epoch 28/30
500/500 [==============================] - 615s 1s/step - loss: 0.7753 - accuracy: 0.7763
Epoch 29/30
500/500 [==============================] - 457s 914ms/step - loss: 0.7426 - accuracy: 0.7781
Epoch 30/30
500/500 [==============================] - 546s 1s/step - loss: 0.7367 - accuracy: 0.7838
```

```
388 / 990 = 39.1919191919192 %
```

2 capas. 30 épocas y 600 pasos

```
Epoch 26/30
600/600 [==============================] - 527s 878ms/step - loss: 0.7003 - accuracy: 0.7951
Epoch 27/30
600/600 [==============================] - 514s 857ms/step - loss: 0.6912 - accuracy: 0.7946
Epoch 28/30
600/600 [==============================] - 584s 973ms/step - loss: 0.6744 - accuracy: 0.7993
Epoch 29/30
600/600 [==============================] - 560s 934ms/step - loss: 0.6580 - accuracy: 0.8028
Epoch 30/30
600/600 [==============================] - 552s 920ms/step - loss: 0.6467 - accuracy: 0.8045
```

```
369 / 990 = 37.27272727272727 %
```

Una vez entrenada la red neuronal y puesta a prueba con el dataset de validación para cada una de las configuraciones, podemos concluir que:

- A un mayor número de épocas, los resultados obtenidos en la fase de validación presentaban porcentajes mucho mayores que otras configuraciones.
- Fue posible evidenciar que el rango de 500 a 600 pasos (*mayoritariamente 500)* en combinación con elevados números de épocas, presentaban la mejor configuración obtenida hasta ahora, para el modelo de datos presentado.
- Dada la complejidad presentada en el modelo de datos (*47 clases y 20.934 muestras),* no se presentó el escenario ideal con el cual obtener un modelo aceptable, dado las capacidades computacionales que tenemos a nuestro alcance. Por lo cual nos permitimos concluir que con un escenario más favorecido (*mayor poder de cómputo)* podríamos obtener mejores resultados en la fase de validación con configuraciones más demandantes.

En el siguiente enlace, se encuentra una carpeta de drive con evidencias del consumo del servicio REST, previamente mencionado:

- https://drive.google.com/drive/folders/1x0us8GEqDUK0jxjtcXn6GAS3yYWLjrJs?usp=sharing

Finalmente, para hacer la validación pertinente de la funcionalidad, se deben correr los archivos de la siguiente manera (*en el proyecto ya se encuentra el modelo generado, no es necesario ejecutar lo siguiente*):

- Cambiar las rutas absolutas de los códigos, por las respectivas rutas
- Correr el archivo main para general el modelo
- Correr el archivo predict para obtener los valores de acierto del modelo

Para ejecutar el proyecto *proyectobackend*, se deben hacer lo siguiente:

- Primer validar que se tengan todas las dependencias de requiments.txt, si usa algún entorno IDE este le puede reconocer el archivo y asistirle en la instalación.

- Después cambiar el modelo si es que genero uno nuevo en el proyecto anterior y reemplazar el que ya se encuentra de no ser así, omita este paso.
- Pasamos a correr el API que nos proporciona un servidor embebido, pasamos a consumir según la ruta que nos arroje este servidor, en las evidencias valga la redundancia podemos evidenciar como hacer consumo de este api a la cual le enviamos una imagen y esta se encarga de devolver el resultado de la predicción.