Entrega redes neuronales de convolución - Ignacio Scuderi

March 23, 2022

1. Introducción El presente trabajo busca analizar desde la perspectiva de las redes neuronales convolucionales el dataset conformado por 1125 imágenes relativos al clima. El mismo fue descargado del repositorio 'Mendeley Data' (url: https://data.mendeley.com/datasets/4drtyfjtfy/1). El dataset contiene imágenes de cielos nublados, lluviosos, soleados, y de amaneceres. El objetivo es aplicar las redes neuronales, de manera de predecir el tipo de cielo de cada imagen.

El primer paso ha sido crear carpetas de 'train' y 'test' para cada tipo de cielo, con una proporción 0.8 y 0.2 respectivamente. Luego se han cargado dichas carpetas a google drive.

[1]:
```
# Monto la unidad del drive
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

2. Convolución Carga de librerías e inicialización de la CNN

[2]:
```
# Carga de librerías necesarias
from keras.models import Sequential
from keras.layers import Conv2D
from keras.layers import MaxPooling2D
from keras.layers import Flatten
from keras.layers import Dense
```

[3]:
```
# Inicialización de la CNN (no requiere la introducción de parámetros)
classifier = Sequential()
```

A partir de la imagen de entrada y utlizando diversos detectores de rasgos, se procede a la creación de los mapas de características que conforman la capa de convolución. Tal como se puede ver debajo, se han especificado 32 detectores de rasgos, de tamaño 3x3.

[4]:
```
classifier.add(Conv2D(filters = 32,kernel_size = (3, 3),
                      input_shape = (64, 64, 3), activation = "relu"))
```

A continuación se aplica un max pooling, de manera de reducir la complejidad y permitir a la red neuronal no solo aprender posiciones y colores, sino también transformaciones y escalados de imágenes.

[5]:
```
classifier.add(MaxPooling2D(pool_size = (2,2)))
```

Se añade una segunda capa de convolución y max pooling de iguales características a la ya añadida

[6]:
```
classifier.add(Conv2D(filters = 32,kernel_size = (3, 3), activation = "relu"))
classifier.add(MaxPooling2D(pool_size = (2,2)))
```

Se añade una tercera capa con 64 detectores de rasgos

[7]:
```
classifier.add(Conv2D(filters = 64,kernel_size = (3, 3), activation = "relu"))
classifier.add(MaxPooling2D(pool_size = (2,2)))
```

Se sigue con el aplanado de los datos (flattening), pasando de las capas pooling a un vector que se lleve toda la información en una sola dimensión vertical que servirá de capa de entrada de la RNA.

[8]:
```
classifier.add(Flatten())
```

Se concluye esta sección realizando la full conection, detallando una función de activación de la capa oculta de tipo 'rectificador lineal unitario'. De igual modo, se utiliza para la capa de salida una función 'softmax'.

[9]:
```
classifier.add(Dense(units = 64, activation = "relu"))
classifier.add(Dense(units = 4, activation = "softmax"))
```

Compilación de la CNN

[10]:
```
classifier.compile(optimizer = "adam", loss = "categorical_crossentropy",␣
↪metrics = ["accuracy"])
```

3. Ajustar la CNN a las imágenes a entrenar Se procede a cargar las imágenes mediante la técnica de aumento de imagen, de modo de evitar el sobreajuste. Mediante esta técnica se realizan transformaciones aleatorias a las imagenes, lo que permite 'multiplicar' artificialmente la cantidad de estas utilizada para entrenar la red neuronal.

[ ]:
```
from keras.preprocessing.image import ImageDataGenerator

train_datagen = ImageDataGenerator(
        rescale=1./255,
        shear_range=0.3,
        zoom_range=0.3,
        horizontal_flip=True)

test_datagen = ImageDataGenerator(rescale=1./255)

training_dataset = train_datagen.flow_from_directory('/content/drive/MyDrive/
↪Colab Notebooks/dataset/Train',
                                                     target_size=(64, 64),
                                                     batch_size=32,
                                                     class_mode='categorical')

testing_dataset = test_datagen.flow_from_directory('/content/drive/MyDrive/
↪Colab Notebooks/dataset/Test',
                                                   target_size=(64, 64),
```

```
                                                batch_size=32,
                                                class_mode='categorical')

classifier.fit(training_dataset,
                         steps_per_epoch= len(training_dataset),
                         epochs=50,
                         validation_data=testing_dataset,
                         validation_steps=len(testing_dataset))
```

Se observa un excelente comportamiento de la red neuronal, la cual ha obtenido un accuracy de 0.9811 en el conjunto de entrenamiento y un 0.9422 en el de validación.