Jorge Armando Castro Escudero

Cc. 98708993

Entrega 2

Objetivo

Identificar tipos de suelo según su granulometría empleando un modelo predictivo y descriptivo de Deep learning

Introducción

La clasificación de suelos es un proceso muy importante en diferentes campos como: la ingeniería geotecnia, la agricultura y la construcción, una manera sencilla de clasificar los suelos, es por el tamaño de los granos o partículas que lo componen para posteriormente determinar si son gravas, arenas, limos o arcillas. Por medio de la evaluación de otras características como el color, la composición o la plasticidad, podemos determinar si son, orgánicos, cenizas, lateríticos, turbas o ricos en hierro. Por ejemplo, tamaños por encima de 2mm son gravas, entre 2mm y 75 micrómetros son arenas, por debajo de 78 micrómetros son arcillas o limos

Gravel

Sand

Silt

.

El proyecto fue realizado empleando las técnicas de la unidad 4 del curso “redes convolucionales”. Para este caso se construyó un modelo de predicción mediante el uso de redes neuronales convolucionales, posteriormente se ejecutará un modelo de clasificación de imágenes de acuerdo con las tres categorías establecidas para este ejercicio.

Desarrollo

Estructura de los notebooks, solución e iteraciones.

Se elaboraron 4 notebooks con las siguientes características:

Notebook 01_exploracion_datos.ipynb: contiene un primer bloque con las librerías requeridas para el funcionamiento del proyecto, un segundo bloque con la url y el dataset que se extrae de la web para el funcionamiento del proyecto y finalmente se hace una exploración de las imágenes extraídas.

Las imágenes utilizadas en el dataset presentan las siguientes características:

- Formato jpg
- Peso de oscilación entre 40 y 44 KB
- Dimensiones de 256px X 256px
- Resolución de 96ppp

El conjunto de datos corresponde al repositorio a datos de libre acceso del autor John Hedengren quien ha realizado estudio diferentes aplicaciones de Deep Learning entre los cuales se encuentran ejercicios aplicados al área de los suelos y geotecnia; el dataset se encuentra contenido en la siguiente URL: http://apmonitor.com/pds/uploads/Main/.

Notebook 02_preprocesado.ipynb: este notebook corresponde a la herramienta de procesamiento de datos iniciales, el cual incluye, las transformaciones de *data augmentation* necesarias para el procesamiento de imágenes de train que se realiza en la CNN a través de la clase ***ImageDataGenerator*** con transformaciones de **re-escalamiento, giro horizontal, zoom, una rotación de 10 grados, recorte, y un cambio de altura y de ancho para las imágenes**. Para los datos de test, la única operación de ***data augmentation*** realizada es el re escalamiento de los datos de 0 a 1, pues este dataset debe ser lo más parecido a las imágenes que en verdad recibiría el modelo. Después, los datos son cargados utilizando el ***método*** Flow_from_directory a los datasets de train y test, con un batch size de 32 y una clasificación categórica.

Notebook 03_construccion_modelo.ipynb: en el siguiente notebook se seleccionan **los hiperparámetros** del modelo como el número de capas de convolución(2 en este parámetro), el número de capas densas(1 en este parámetro), el tamaño de las capas(32), cuántas épocas se van a realizar(20) y el nombre del modelo.

Posteriormente se crea la arquitectura del modelo a entrenar. En este caso se utiliza un ***objeto Sequential de keras*** en el cual se agrega primeramente una capa convolucional con ***un filtro*** (kernel) de *3x3*, y un *padding SAME*, en el cual se rellena con zeros para ***no perder dimensiones*** de filas y columnas. Luego se agrega una función de *activación* tipo **relu** para esta capa, seguido de un *MaxPooling* con un filtro(kernel) de 2x2.

El total de capas adicionales de convolución pueden variar según el hiperparámetro de ***num_conv_layers*** que se encuentra **en *el ciclo for***, para así agregar más de estas ***por medio de iteraciones si se desea***, En este caso solo se agregan ***otras dos capas***. Una con ***filtro*** *de 3x3*, activación relu y maxpooling de 2x2, y la siguiente con una convolución de ***filtro*** *2x2*, activación relu y maxpooling de 1x1.

Después, se ***agrupan todos los datos en un solo vector con la función Flatten para poder empezar a colocar las capas densas***. Solo se implementa una para este modelo, pero esto también puede variar según el ***hiperparámetro num_dense_layer*** gracias al ***ciclo for***. Por último, en esta arquitectura, ***se agrega la capa densa de clasificación de los tres tipos de suelos*** seguido con una función de activación ***de tipo softmax***, que mostrará la ***distribución de probabilidad categórica*** y así poder obtener la mejor coincidencia entre las opciones de suelos.

Además, es importante aclarar que ***la función de pérdida*** (penalización al modelo) que se aplicó es una ***cross_entropy categórica***, con un ***optimizador de tipo Adam,*** que suele ser más rápido que un SGD, y se medirá el ***rendimiento del modelo sobre la métrica de accuracy***.

En las siguientes líneas el modelo se ***entrena y valida*** utilizando el método fit() sobre el dataset de train y test. Por último se guarda el modelo conforme a los resultados obtenidos en su precisión, obtenidos en la metrica accuracy y loss function.

Ademas, en este archivo podemos encontrar la **Función def make_prediction().** Esta función permite leer imágenes con la librería cv2 y mostrarlas. También se realiza el proceso pertinente para cargar las imágenes como arrays y se ocupa de escalarlas. Adicionalmente se crea una lista *de clases con los 3 tipos de suelos diferentes.* En esta función de código también se encuentra contenida la función ***argmax***(*devuelve el valor máximo de la distribución de probabilidades empleado anteriormente, softmax*). *Seguido a eso se busca el valor real de la imagen, y se hace la respectiva comparación del valor del modelo con este valor real.*

Notebook 04_clasificacion_categoria.ipynb : como valor agregado al proyecto se emplea un método de clasificación función "Split_images" esta opción se aplica ya que una solo imagen puede contener los tres tipos o categorías de suelos.

Función Split_images:

La siguiente función permite crear ***patches (pequeños recortes de las imágenes reales) de 64 pixeles***. Las primeras líneas crearán carpetas y archivos diferentes para cada tipo de suelo a clasificar. Luego, cada imagen es leída y se extraen sus principales características en las variables h (height), w(width), c(channels). La *variable im_dim decidirá el ancho y largo del recorte a realizar*.

Desde la línea 16 se crean ***ciclos for anidados*** cuya finalidad es recorrer primero las filas y luego las columnas de cada imagen e ir ***creando cortes de 64 pixeles x 64 pixeles***. Después, se comprueba si las dimensiones de la imagen recortada coinciden con el criterio de 64 pixeles (línea 20) y se procede a almacenarlas.

Las celdas posteriores se encargan de crear las direcciones donde luego se almacenan los datasets de train y test con los recortes de las imágenes extraídas en el paso anterior, esto con el fin de hacer predicciones con el modelo previamente entrenado sobre estas nuevas imágenes.

Función classify_images:

Esta función es similar a la anterior función de clasificación, solo que se inicializa ***con 3 contadores para almacenar el número total de partes de la imagen reconocidas como cada clase*** (gravel_count, sand_count, silt_count). Se leen las imágenes y se aplica un resize a 1024 pixeles por 1024 para recorrer la imágenes por zonas de 256 pixeles x 256 pixeles, y realizar la clasificación en esta región con el modelo previamente entrenado.

Las líneas 11 a 24 tienen un código similar al de la función Split_images, solo que ahora se recorren las filas y columnas de a 256 pixeles y en cada una de estas zonas se realiza una clasificación con el modelo. Según la etiqueta dada por la predicción, se aumenta un valor a cada uno de los contadores anteriormente mencionados (gravel_count, sand_count, silt_count). Esto con el fin de sacar proporciones de clasificación. Esta función devuelve un arreglo con las proporciones de cada clase.

Función model_classify():

En esta función se ***estandarizan*** los recortes de las imágenes y se expanden sus dimensiones. Luego, se hace una predicción sobre estos recortes y se devuelve la clase que cuente con el porcentaje de predicción más alto de la variable ***prediction array****.*

Función classify_percentage*:*

La finalidad de esta función es devolver el tiempo que se demora la red en clasificar los diferentes porcentajes utilizando el método time.time(). Por esto, se aplica la función classify_images, cuya entrada es la imagen completa, y la salida es la proporción de los distintos suelos presentes en esta luego de ***ser clasificada por el modelo***. Se muestra también la imagen en cuestión y las distintas proporciones de clasificación en porcentajes, así como el tiempo de ejecución.

Resultados obtenidos

- De manera automática se extrae la data sin necesidad de realizar de forma manual el cargue de la data.
- Al evaluar las paramétricas del modelo ejecutado conforme a los parámetros asignados al modelo, se puede evidenciar que la precisión o accuracy esta entre 0.33 y 0.86. Lo que los parámetros seleccionados permiten una predicción mas exacta a medida que se aumenta el entrenamiento.
- En cuanto al resumen del modelo.

```
[10]  1 model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 256, 256, 32)      896

 activation (Activation)     (None, 256, 256, 32)      0

 max_pooling2d (MaxPooling2D  (None, 128, 128, 32)     0
 )

 conv2d_1 (Conv2D)           (None, 126, 126, 32)      9248

 activation_1 (Activation)   (None, 126, 126, 32)      0

 max_pooling2d_1 (MaxPooling  (None, 63, 63, 32)       0
 2D)

 conv2d_2 (Conv2D)           (None, 62, 62, 32)        4128

 activation_2 (Activation)   (None, 62, 62, 32)        0

 max_pooling2d_2 (MaxPooling  (None, 62, 62, 32)       0
 2D)

 flatten (Flatten)           (None, 123008)            0

 dense (Dense)               (None, 32)                3936288

 activation_3 (Activation)   (None, 32)                0

 dense_1 (Dense)             (None, 3)                 99

 activation_4 (Activation)   (None, 3)                 0

=================================================================
Total params: 3,950,659
Trainable params: 3,950,659
Non-trainable params: 0
_________________________________________________________________
```

En este modelo se obtuvieron los siguientes resultados; para la primera capa convolucionada se entonaron 9248 parámetros de 32 valores por batch, en la segunda capa de convolución se entrenaron 4128 parámetros y posterior al aplanamiento de los datos, la densificación y la activación de las capas se obtuvo que par aun total del 3.950.659 de datos entrenados de un total de parámetros igual, por lo cual no hubo perdida de información entre los datos

- Se evaluó el modelo con varias imágenes, al ingresar una imagen contenida en una de las categorías, el modelo permite predecir que esta imagen reposa en una caréta incorrecta o correcta e indica su etiquete o categoría respectiva
- En cuanto a la predicción realizada con las funciones **classify_images, Split_images, model_classify() y classify_percentag,** se logró una predicción concisa de las tipologías de suelos contenidas en un misma imagen de forma coherente**.**