Informe Ejecutivo – Proyecto final del curso Fundamentos de Deep Learning

Medellín, 11 de diciembre de 2022

Juan Manuel Fernández Ospina
Estudiante de Maestría en Ingeniería
Universidad de Antioquia

En el presente informe se pretende mostrar de manera sintetizada la implementación del proyecto final del curso Fundamentos de Deep Learning.

En el desarrollo se desea segmentar la serie de edificaciones presentes en una imagen satelital dada como entrada, ofreciendo como salida una imagen que permita distinguir las zonas construidas de aquellas con ausencia de edificaciones. La imagen resultante es binarizada, lo cual contribuye a una mejor interpretación de los resultados.

El dataset es obtenido de Kaggle, su nombre es Synthetic Word OCR. Cuenta con un tamaño de 4.79GB, conteniendo un total de 280800 imágenes satelitales en las que se encuentran zonas con edificaciones de características diversas. También trae consigo una serie de etiquetas, para cada imagen, alojadas en un archivo .json.

Se desglosa la estructura de la entrega a continuación:

Descripción de la estructura de los notebooks entregados

Se sintetiza la entrega en un solo notebook, llamado FinalProjectDL.ipynb, internamente está claramente segmentado en diversas secciones, una es la de obtención y acondicionamiento del dataset necesario para el desarrollo de todo el proceso, en esta parte se especifica la importación de las imágenes desde Kaggle, por medio de un API Token que será incluido en la entrega, se muestra también la generación del fichero donde será descargado el archivo .zip en Colab y posteriormente descomprimido para su uso.

Adicionalmente se especifican las rutas para la interacción con las imágenes y con el archivo .json que contiene las etiquetas.

La otra sección es para la importación de librerías fundamentales para el desarrollo y el procesamiento de las etiquetas, ya que vienen fragmentadas, entonces se procede a hacer un cruce de ellas por medio de una clave, que en este caso es un id asignado a cada imagen, finalmente queda un dataframe que contiene el id, el id de la imagen, la segmentación de la imagen especificada, el área, el polígono que la configura y una categorización interna.

Posteriormente está la sección de conversión de polígonos a máscaras, le sigue la separación de datos para entrenamiento y validación, generación de lotes, luego la sección en que se define la red neuronal usada, posteriormente se definen las métricas y pérdidas del modelo, luego la sección de entrenamiento y, finalmente, la sección donde se visualizan los resultados de la implementación. Se plantea tal estructura del notebook pensando en una entrega compacta y se tuvo en cuenta el respeto de las capacidades computacionales ofrecidas por Colab.

Informe Ejecutivo – Proyecto final del curso Fundamentos de Deep Learning

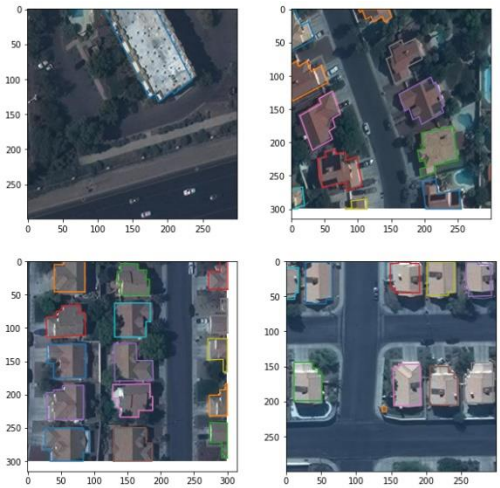
Descripción de la solución

La solución busca básicamente crear una segmentación de los edificios presentes en imágenes satelitales, de forma que estas estructuras queden plenamente identificadas más allá de su entorno; para la consecución de lo anterior se busca producir como salida una imagen binarizada, en la que aparezcan en color negro el área representada por las edificaciones y en color blanco su entorno.

Para la consecución de lo anterior se accede al dataset descrito previamente y se inicia con la construcción de un dataframe que contiene información para cada imagen, esta información fue previamente obtenida de un archivo llamado annotations.json que se encuentra con el dataset descargado directamente desde Kaggle.

Las imágenes que se encuentran en el dataset son como las que se muestran a continuación:

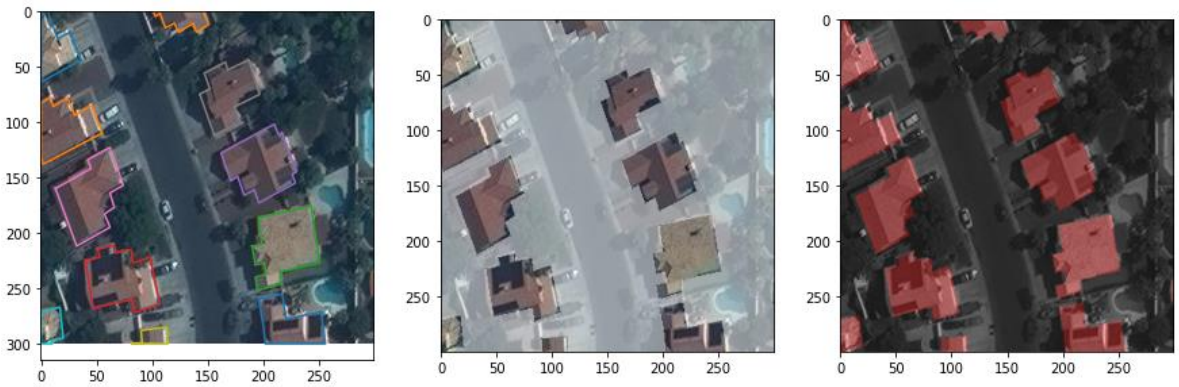
Las imágenes de 300x300 y corresponden a vistas superficiales tomadas por satélite. A partir del archivo annotatios.json se realizan las segmentaciones de los edificios, las cuales quedan de esta manera:



Informe Ejecutivo – Proyecto final del curso Fundamentos de Deep Learning

Partiendo de las segmentaciones se crean las máscaras, de las edificaciones presentes en la imagen. La obtención de las máscaras se logra con el uso de la función meshgrid de numpy para obtener los arrays con los puntos de interés de cada edificación, luego se tiene un matriz de ceros a partir de la cual se reconstruye la imagen teniendo en cuenta las coordenadas de las edificaciones para crear la máscara.

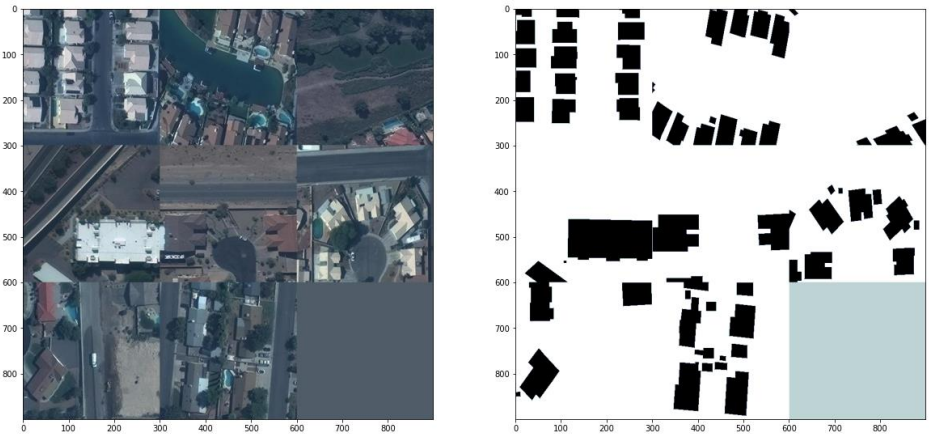
El proceso de construcción de las máscaras se da de la siguiente manera:



Luego, por medio de train_test_split, de sklearn se hace una segmentación de los datos, para tener un lote para entrenamiento y otro para validación, el test_size fue de 0.25.

Paso siguiente es la generación de lotes para el procesamiento de las imágenes, para esto se realiza una función que envía un lote de la cantidad especificada por el usuario y es controlado por un yield, en lugar de un return. Se busca que no sea superado el tamaño de lote especificado.

En este punto ya se tiene la salida deseada, que es de esta manera:



Lo siguiente es la definición de la red neuronal convolucional para el procesamiento de las imágenes, inicia con una capa de entrada, se integra ruido gaussiano para contrarrestar posibles efectos de overfitting, luego se usa una serie de capas sucesivas entre batch normalization, conv2d y activation, que finalmente llegan a un concatenate y, en este caso, se usa un spatial_dropout2d ya que tiene mejor gestión sobre los mapas de características

resultantes de las capas convolucionales que el dropout. Para la adecuación final de la imagen se usa un cropping2d y un zeropadding2d, para, finalmente, llegar a la salida.

La estructura y conexión de la red se muestra a continuación:

Layer (type)	Output Shape	Param #	Connected to
RGB_Input (InputLayer)	[(None, 300, 300, 3)]	0	[]
gaussian_noise_2 (GaussianNoise)	(None, 300, 300, 3)	0	['RGB_Input[0][0]']
batch_normalization_12 (BatchNormalization)	(None, 300, 300, 3)	12	['gaussian_noise_2[0][0]']
conv2d_22 (Conv2D)	(None, 300, 300, 8)	216	['batch_normalization_12[0][0]']
batch_normalization_13 (BatchNormalization)	(None, 300, 300, 8)	32	['conv2d_22[0][0]']
activation_10 (Activation)	(None, 300, 300, 8)	0	['batch_normalization_13[0][0]']
conv2d_23 (Conv2D)	(None, 300, 300, 8)	576	['activation_10[0][0]']
batch_normalization_14 (BatchNormalization)	(None, 300, 300, 8)	32	['conv2d_23[0][0]']
activation_11 (Activation)	(None, 300, 300, 8)	0	['batch_normalization_14[0][0]']
conv2d_24 (Conv2D)	(None, 300, 300, 16)	1152	['activation_11[0][0]']
batch_normalization_15 (BatchNormalization)	(None, 300, 300, 16)	64	['conv2d_24[0][0]']
activation_12 (Activation)	(None, 300, 300, 16)	0	['batch_normalization_15[0][0]']
conv2d_25 (Conv2D)	(None, 300, 300, 16)	2304	['activation_12[0][0]']
conv2d_26 (Conv2D)	(None, 300, 300, 16)	2304	['activation_12[0][0]']
conv2d_27 (Conv2D)	(None, 300, 300, 16)	2304	['activation_12[0][0]']
conv2d_28 (Conv2D)	(None, 300, 300, 16)	2304	['activation_12[0][0]']
conv2d_29 (Conv2D)	(None, 300, 300, 16)	2304	['activation_12[0][0]']
conv2d_30 (Conv2D)	(None, 300, 300, 16)	2304	['activation_12[0][0]']
concatenate_2 (Concatenate)	(None, 300, 300, 99)	0	['batch_normalization_12[0][0]', 'conv2d_25[0][0]', 'conv2d_26[0][0]', 'conv2d_27[0][0]', 'conv2d_28[0][0]', 'conv2d_29[0][0]', 'conv2d_30[0][0]']
spatial_dropout2d_2 (SpatialDropout2D)	(None, 300, 300, 99)	0	['concatenate_2[0][0]']
batch_normalization_16 (BatchNormalization)	(None, 300, 300, 99)	396	['spatial_dropout2d_2[0][0]']
activation_13 (Activation)	(None, 300, 300, 99)	0	['batch_normalization_16[0][0]']
conv2d_31 (Conv2D)	(None, 300, 300, 32)	28512	['activation_13[0][0]']
batch_normalization_17 (BatchNormalization)	(None, 300, 300, 32)	128	['conv2d_31[0][0]']
activation_14 (Activation)	(None, 300, 300, 32)	0	['batch_normalization_17[0][0]']
conv2d_32 (Conv2D)	(None, 300, 300, 1)	33	['activation_14[0][0]']
cropping2d_2 (Cropping2D)	(None, 268, 268, 1)	0	['conv2d_32[0][0]']
zero_padding2d_2 (ZeroPadding2D)	(None, 300, 300, 1)	0	['cropping2d_2[0][0]']

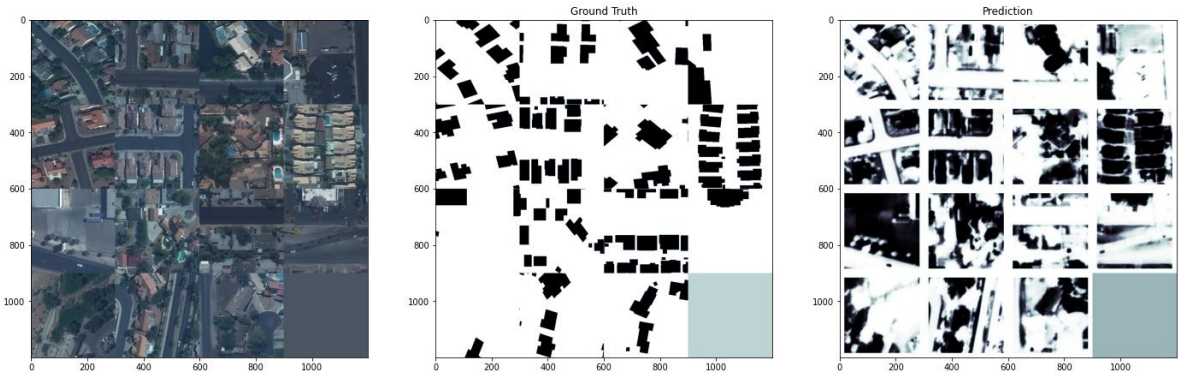
Total params: 44,977
Trainable params: 44,645
Non-trainable params: 332

Posteriormente, se define una parte fundamental del proceso, las métricas y pérdidas, de aquí se destaca la definición de la función de pérdida, que finalmente fue la crossentropy binary y el optimizador que, luego de varias pruebas, se determina será el Adam.

Luego se definen algunos callbacks para el proceso de entrenamiento del modelo.

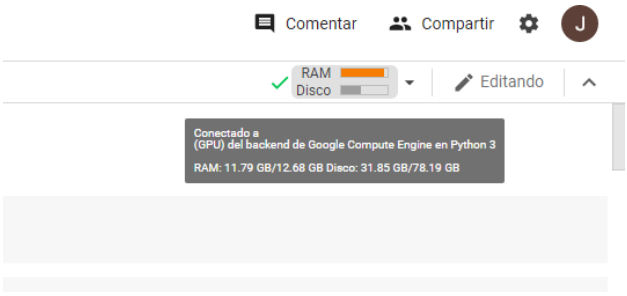
Posteriormente se entrena el modelo, pasando como argumentos el lote generado, se usa multiprocesamiento, en este caso se usan solo dos épocas, se pasa el lote de validación se llaman los callbacks, se definen pasos en términos del batch_size, el cual se definió en 16 dada la capacidad computacional ofrecida por Colab.

Finalmente se tiene la salida, la cual se da de la siguiente manera:



Puede apreciarse que se tienen aún bastantes fallas con el modelo, en cuanto a precisión, sin embargo, logra captar características esenciales. Se ve sujeta a algunos errores cuando la superficie mapeada por el satélite no tiene grandes contrastes. Se espera un mejor rendimiento del modelo al mejorar el proceso de entrenamiento, sin embargo, no se logró obtener un mejor proceso debido a que al aumentar parámetros como el batch size Colab excedía las capacidades de memoria RAM ofrecida en un plan no pago.

Entrenando el modelo, tal como se hizo, se llegó al siguiente consumo de capacidad computacional.



Al contar con un dataset tan grande se dificultó demasiado el proceso de entrenamiento.

Descripción de las iteraciones

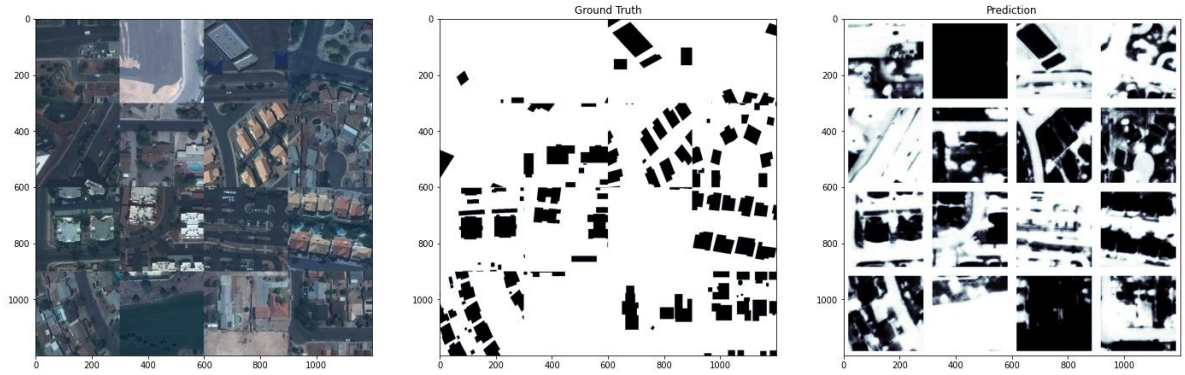
Informe Ejecutivo – Proyecto final del curso Fundamentos de Deep Learning

Sen el proceso de entrenamiento se trabajó con un batch_size de 16, con multiprocesamiento, un paso por época condicionado por el batch_size que, en este caso, por el valor definido, no será menor que cien, entonces siempre tendrá 100 pasos por época durante la presente ejecución.

Durante las dos épocas presentó los siguientes cambios:

Mostrando que efectivamente se reducía la pérdida y se incrementaba el porcentaje de acierto.

Descripción de los resultados



En la anterior imagen se aprecia la salida, se ve que hay casos en los que la segmentación se da de muy buena manera, como en la imagen de la fila 1 columna 3, otras en las que el error es inmenso, como la de la fila 1 columna 2, sin embargo, dando un mejor proceso de entrenamiento se contempla una mejoría considerable en el comportamiento del modelo.

A grandes rasgos se describe, entonces, el proceso de desarrollo del proyecto final del curso.