Driven Data Challenge - Segmenting buildings from drone imagery

Jeetendra Gan   
*Department of Computer Science and Engineering*

*University at Buffalo*New York, United States  
jgan2@buffalo.edu

Chanadana Singh

*School of Engineering and Applied Sciences*  
*University at Buffalo*New York, United States  
csingh9@buffalo.edu

Karun Parashar  
*School of Engineering and Applied Sciences*  
*University at Buffalo*New York, United States  
karunpar@buffalo.edu

***Abstract*— Our project is based on a challenge hosted on Driven Data (**[**Link**](https://www.drivendata.org/competitions/60/building-segmentation-disaster-resilience/)**). The goal in this challenge is to segment building footprints from aerial imagery. The data consists of drone imagery from 10 different cities and regions across Africa. Our goal is to identify the presence or absence of a building on a pixel-by-pixel basis.**

# Data Description

* 1. Train data:

The data is around 200 GBs large, consisting primarily of images. Spatial resolution varies from region to region. All images include 4 bands: red, green, blue and alpha. The alpha band can be used to mask out NoData values.

Given that the labels vary in quality (e.g. how exhaustively an image is labeled, how accurate the building footprints are), the training data have been divided up into tier 1 and tier 2 subsets. The tier 1 images have more complete labels than tier 2.

* 1. Test Data:

The test set consists of 11,481 1024 x 1024 pixel COG "chips" derived from a number of different scenes. None of these scenes are included in the training set. Some of the test scenes are from regions that are present in the training set while others are not. The correct geo-references for the test chips have been removed. The test set labels (unavailable to participants) have a level of accuracy commensurate with the tier 1 data.

* 1. Labels:

Each image in the train set corresponds to a GeoJSON, where labels are encoded as FeatureCollections. geometry provides the outline of each building in the image. Your goal is only to classify the presence (or lack thereof) of a building on a pixel-by-pixel basis.

train\_metadata.csv links the each image in the train set with its corresponding GeoJSON label file. This csv also includes the region and tier of the image. Note that region information is not provided for the test set.

Label GeoJSON files have been clipped to the extents of the non-NoData portions of the images, all building geometries will overlap with image data.

.

# Data Preparation

1. The data is in the form of .tiff files which are very large. We have segmented them in smaller chunks.
2. We have resized the train and test data into 512\*512 ,256\*256 and 128\*128 sized images retaining the RGB channels.
3. Response images which are masks of the training data, are also resized and dimensions expanded to add one channel.
4. Preprocessed images are saved.

# Visualization

Following are the samples of the original image and their corresponding mask

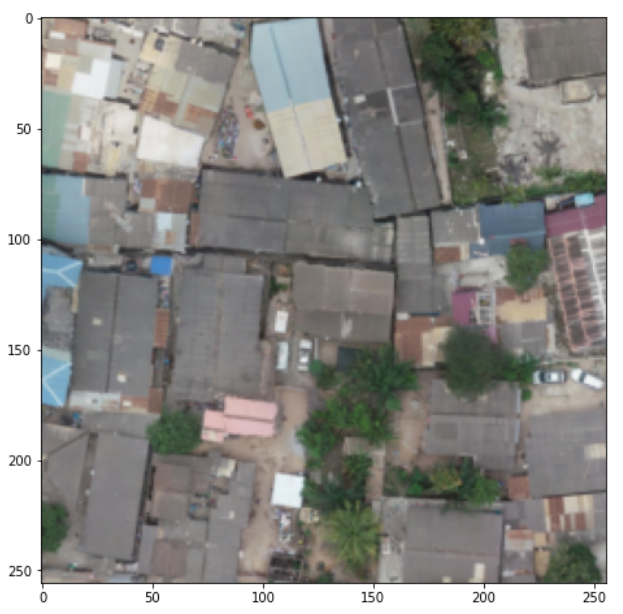
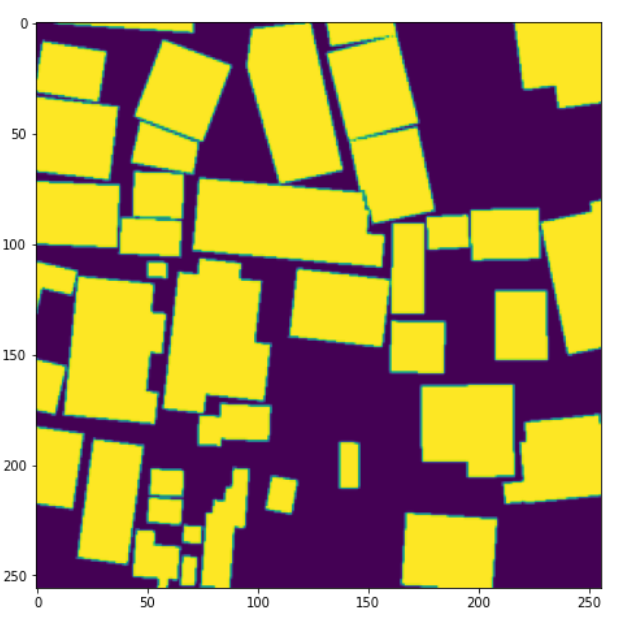


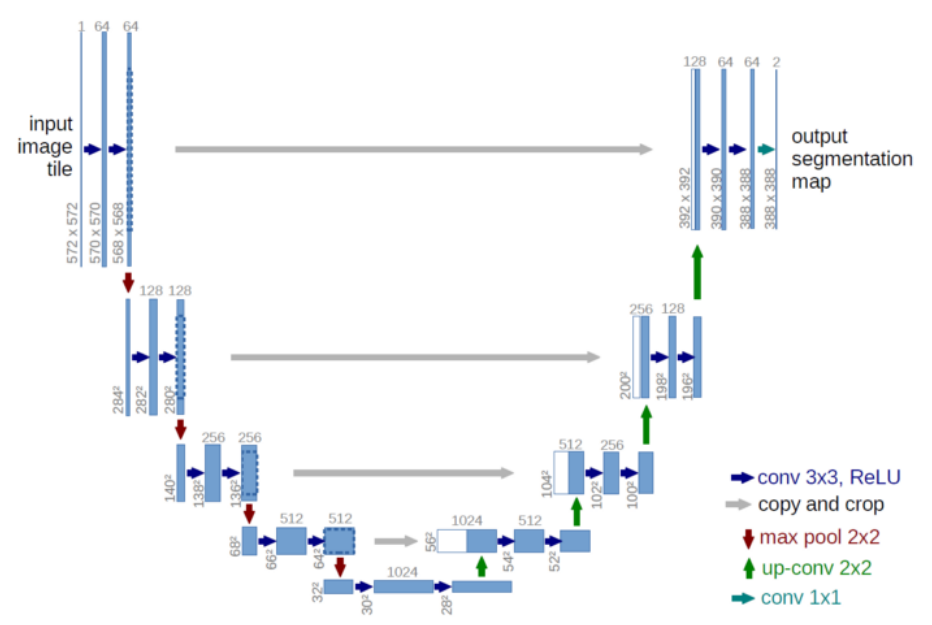
Image sample



Mask Sample

# MODEL ARCHITECTURE USED

UNET

****

1. U-Net Architecture

The main idea is to supplement a usual contracting network by successive layers, where pooling operations are replaced by up-sampling operators.

Hence these layers increase the resolution of the output. Down-sampling and then up-sampling and concatenating with the respective down sampled .

## Training

Convolution layer uses RELU activation and padding is kept the same. The weights of the filters are initialized with normal distribution with size 3\*3. There are dropout layers and max pool layers after each convolution layer.

We have built up the model with various sized images 512\*512 , 128\*128 , 256\*256 and analysed the total number of parameters required for training the images.

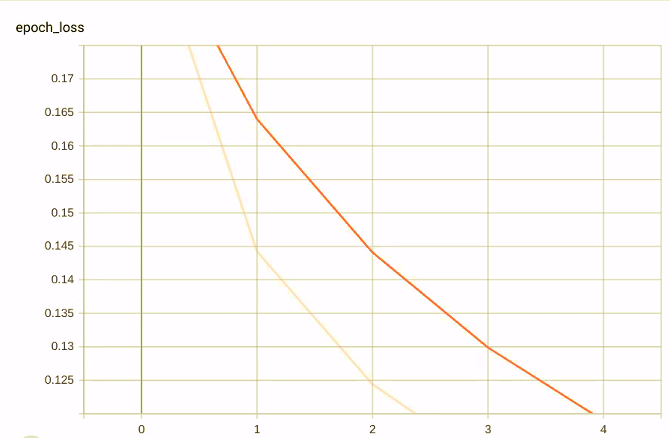
It took us around 9 hours to train the 128\*128 model, 13 hours to train the 256\*256 model and more than 20 hours for the 512\*512 model.

## Testing

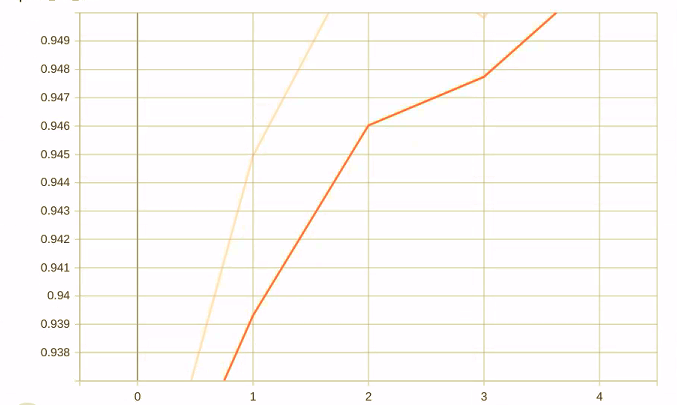
The model has been tested on unseen images which were downloaded from the same source and corresponding masks were created.

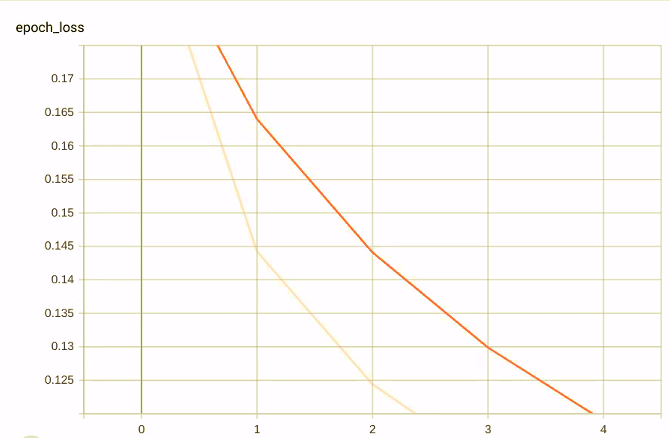
The overall performance of the model was very smooth. The validation accuracy and loss were not much different from the training counterparts.

## Performance Evaluation for training and validation data

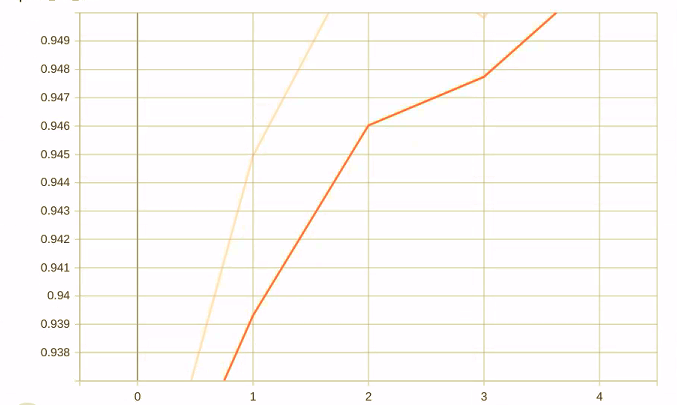
Epochs V/S Training Loss 

Epochs V/S Training Accuracy

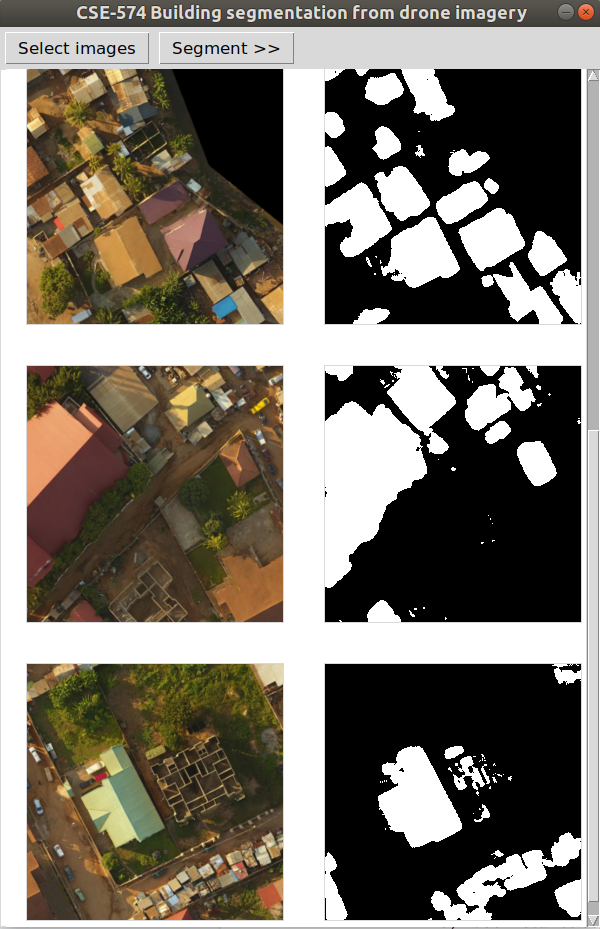


Epochs V/S Validation Loss 

Epochs V/S Validation Accuracy



##### USER INTERFACE



The user interface is shown on the Top. There is an option for the user to “Select images”. Any number of images can be selected from the file system. Images of any size can be selected, but they will be converted to 256\*256 images since our model accepts that size.

After “Segment” is clicked, the U-Net model (trained using cosine similarity as the loss, and accuracy as metric) is used to find predictions. The predictions are between 0 and 1 for each pixel. We set a threshold of 0.5 to classify a pixel as 0/1.

We have used the python’s Tkinter library for building the desktop application.

VI.CONCLUSION

The best and most efficient model turned out to be the one with 256\*256 resolution, because it led to improved accuracy score than 128\*128 and at the same time took far less time than 512\*512 image model.

##### References

<https://towardsdatascience.com/understanding-semantic-segmentation-with-unet-6be4f42d4b47>

<https://towardsdatascience.com/unet-line-by-line-explanation-9b191c76baf5>