# Progress Report

**Author**: Xiong Zheng zhxiong@upenn.edu , Zhenzhao Xu zhenzhao@upenn.edu

## Progress

### Download Data


It takes effort to download the image data from the original SpaceNet S3 bucket. Instead, we obtain the same data from Kaggle, there are 1408 items in the data. Within each item,there are a high-resolution images (1024*1024 RGB) and a GeoJson file with geometry attributes of buildings.

### Format Data with missing pixels

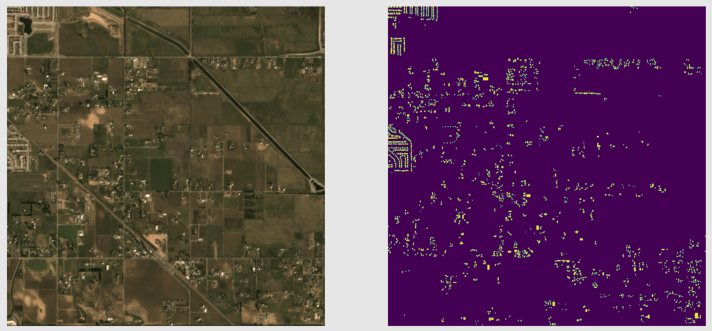
Some images have the shape of 1023 * 1023 instead of 1024 * 1024. So to format the data, we enlarge all images into 1024 * 1024 pixels by filling black pixels to the right and bottom edges of those images. We don't have to do the same change to GeoJson, with its coordinates starting at the upper left point of the image. So it fits well with those modified satellite images.

```
def imagePathToArray(path):
  # read satellite img as numpy array
  image = Image.open(path)
  image = np.array(image)[:,:,:3]
  # fill black pixels to the right and bottom edges of 1023*1023 images
  zeros = np.zeros((1024,1024,3), dtype=np.uint8)
  zeros[:image.shape[0],:image.shape[1],:] = image
  return zeros
```

### Rasterize GeoJson Data

```
buildingMask = features.rasterize(geom, out_shape=(1024,1024))
```

We are using `rasterio` to rasterize GeoJson Data of buildings as `Building Masks`. The masks are taken as the output of the Image Segmentation model. The shape of mask is the same as the satellite image (1024 * 1024 * 1)



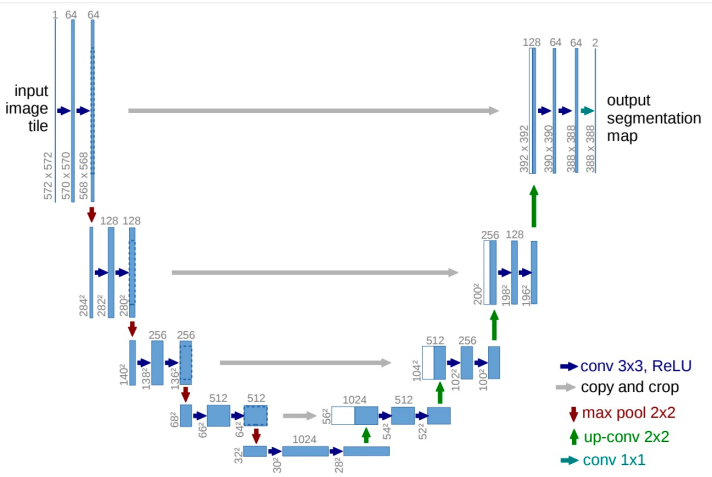
## Split Images

To make the training process efficient, we split each `Satellite Image` and `Building Masks` (1024 * 1024) to 16 images (256 * 256).

```
def splitArray(arr, splitNum=4):
  # split the image into smaller chunks
  return [np.hsplit(x, splitNum) for x in np.vsplit(arr,splitNum)]
```

# Next Steps & Challenges

- Based on the use case, we choose U-Net as the model for image segmentation, the concept is shown as follows:




*Reference: [U-Net: Convolutional Networks for Biomedical Image Segmentation]*

- However, the bandwidth of google Colab is not stable so it is practically impossible (4 hours) to load the 22,529 images data and masks. (Both in `.png` format.) So we have to convert it into `.npy` (Numpy array binary storage files) file in our local computers and import it into Colab as a whole.
- Importing all 22,529 data at once at the beginning of the modeling process may also be a challenge to our RAMs and Neural Network Training. A better solution will be reading a batch of data at a time. However, to achieve that, over-complicated tensorflow hand-written codes are required.

# pre-processing

April 16, 2022

[ ]:
```
!pip install geopandas rasterio
```

[ ]:
```
from google.colab import drive
drive.mount('/content/drive')
```

[ ]:
```
import os, re
from pathlib import Path
import pandas as pd
import geopandas as gpd
import numpy as np
import rasterio as rio
from rasterio import features
from PIL import Image
import matplotlib.pyplot as plt
from tqdm.notebook import tqdm

# move to the dir of this notebook
os.chdir("/content/drive/MyDrive/650-fin/")
```

[ ]:
```
# read train csv
trainCSV = pd.read_csv('SN7_buildings_train_csvs/csvs/
 ↪sn7_train_ground_truth_pix.csv')
trainCSV.head()
```

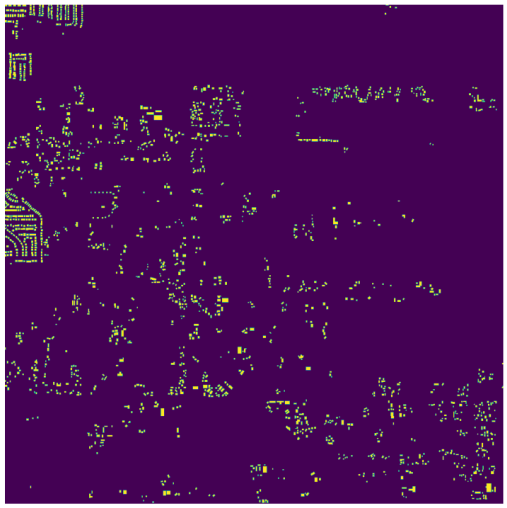
[ ]:
```
                                            filename    id  \
0  global_monthly_2018_01_mosaic_L15-0331E-1257N_…    91
1  global_monthly_2018_01_mosaic_L15-0331E-1257N_…  1452
2  global_monthly_2018_01_mosaic_L15-0331E-1257N_…  1616
3  global_monthly_2018_01_mosaic_L15-0331E-1257N_…   950
4  global_monthly_2018_01_mosaic_L15-0331E-1257N_…  1765

                                            geometry
0  POLYGON ((814.9857745971531 845.6005742002744,…
1  POLYGON ((886.3093001581728 842.7000443374272,…
2  POLYGON ((930.1175056053326 840.4646877695341,…
3  POLYGON ((923.3884236379527 840.6308379401453,…
4  POLYGON ((926.2129765632562 838.8862611351069,…
```

[ ]:
```
# transfer building wkt into gpd
geom = gpd.GeoSeries.from_wkt(trainCSV.geometry)
trainGPD = gpd.GeoDataFrame(trainCSV[["filename","id"]],geometry=geom)
trainGPD = trainGPD.set_index("filename")
fileNames = list(trainCSV.filename.unique())
```

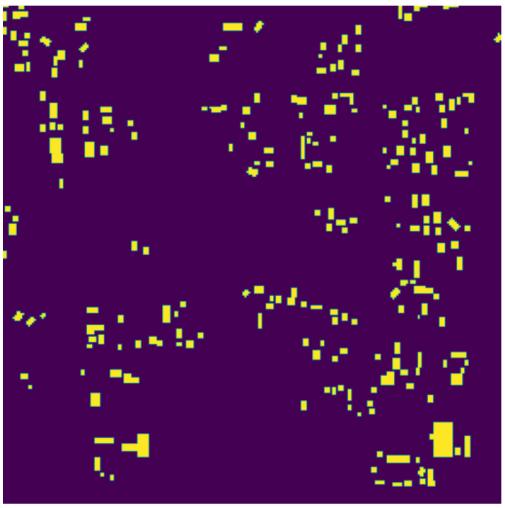
[ ]:
```
def fileNameToPath(file):
  # return real image path
  folder = Path("./SN7_buildings_train/train/")
  folder /= re.search(r"L.*", file).group()
  folder /= "images_masked"
  folder /= file + ".tif"
  return str(folder)

def imagePathToArray(path):
  # read satellite img as numpy array
  image = Image.open(path)
  image = np.array(image)[:,:,:3]

  zeros = np.zeros((1024,1024,3), dtype=np.uint8)
  zeros[:image.shape[0],:image.shape[1],:] = image
  return zeros

def fileNameToMask(gdf, size=(1024,1024)):
  # rasterize building geodataframe into numpy array
  geom = trainGPD.query(f"filename == '{fileName}'").geometry
  buildingMask = features.rasterize(geom, out_shape=size)
  return buildingMask

def splitArray(arr, splitNum=4):
  # split the image into smaller chunks
  return [np.hsplit(x, splitNum) for x in np.vsplit(arr,splitNum)]

def saveImgs(imgs, masks, id, folder=("train2/x","train2/y")):
  # save list of list of images
  for i in range(len(imgs)):
    for j in range(len(imgs[0])):
      img = Image.fromarray(imgs[i][j])
      mask = Image.fromarray(np.uint8(masks[i][j] * 255), 'L')

      path = [Path(f)/f"{id:0>6}_{i:0>2}_{j:0>2}.png" for f in folder]
      img.save(path[0])
      mask.save(path[1])
```

[ ]:
```
# Visualize sample image
geom = trainGPD.query(f"filename == '{fileNames[0]}'").geometry
buildingMask = features.rasterize(geom, out_shape=(1024,1024))
```

```
img = imagePathToArray(fileNameToPath(fileNames[0]))

plt.figure(figsize=(10,5),dpi=200)
plt.subplot(1,2,1)
plt.imshow(imagePathToArray(fileNameToPath(fileNames[0])))
plt.axis("off")
plt.subplot(1,2,2)
plt.imshow(buildingMask)
plt.axis("off")
plt.show()
```



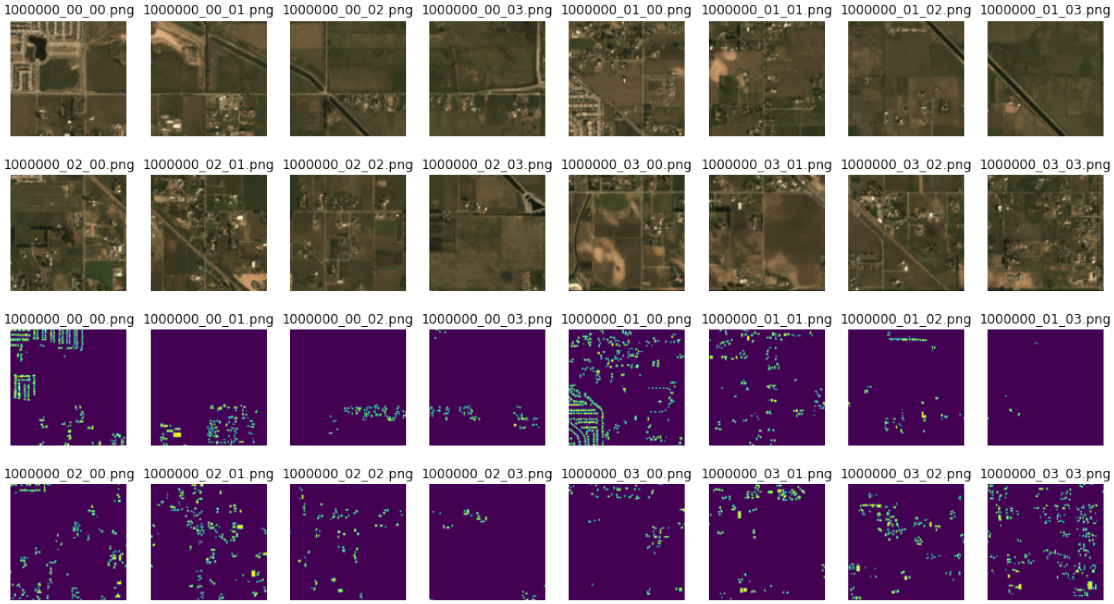
[ ]:
```
# split the image into smaller chunks

splitsImg = splitArray(img)
splitsMask = splitArray(buildingMask)

plt.figure(figsize=(10,5),dpi=200)
plt.subplot(1,2,1)
plt.imshow(splitsImg[3][3])
plt.axis("off")
plt.subplot(1,2,2)
plt.imshow(splitsMask[3][3])
plt.axis("off")
plt.show()
```



[ ]:
```
# PREPROCESSING!!!!!
for i, fileName in enumerate(tqdm(fileNames)):
  img = imagePathToArray(fileNameToPath(fileName))
  buildingMask = fileNameToMask(fileName, img.shape[:2])

  splitsImg = splitArray(img)
  splitsMask = splitArray(buildingMask)

  saveImgs(splitsImg,splitsMask,i+1000000)
  # break
```

```
  0%|          | 0/1408 [00:00<?, ?it/s]
```

[ ]:
```
f, axs = plt.subplots(4,8,figsize=(18,10))
for i,file in enumerate(os.listdir("train/x")):
  p = "train/x/"+file
  axs.flat[i].imshow(Image.open(p))
  axs.flat[i].axis("off")
  axs.flat[i].set_title(file)
  if i >= 15: break

for i,file in enumerate(os.listdir("train/y")):
  p = "train/y/"+file
  axs.flat[i+16].imshow(Image.open(p))
  axs.flat[i+16].axis("off")
  axs.flat[i+16].set_title(file)
  if i >= 15: break
```


1000000_00_00.png
1000000_00_01.png
1000000_00_02.png
1000000_00_03.png
1000000_01_00.png
1000000_01_01.png
1000000_01_02.png
1000000_01_03.png
1000000_02_00.png
1000000_02_01.png
1000000_02_02.png
1000000_02_03.png
1000000_03_00.png
1000000_03_01.png
1000000_03_02.png
1000000_03_03.png
1000000_00_00.png
1000000_00_01.png
1000000_00_02.png
1000000_00_03.png
1000000_01_00.png
1000000_01_01.png
1000000_01_02.png
1000000_01_03.png
1000000_02_00.png
1000000_02_01.png
1000000_02_02.png
1000000_02_03.png
1000000_03_00.png
1000000_03_01.png
1000000_03_02.png
1000000_03_03.png