MAXAR SPATIAL CHALLENGE

WEBINAR TRAINING
08 September 2020 | 11 a.m. – 12.15pm AEST time
Dr Sebastien Wong | Consilium Technology

Agenda:

<table>
<thead>
<tr>
<th>Time</th>
<th>Agenda</th>
</tr>
</thead>
<tbody>
<tr>
<td>11.00-11.10</td>
<td>Maxar Spatial Challenge Updates</td>
</tr>
<tr>
<td>11.10-11.55</td>
<td>Introduction &amp; Training for GBDX Notebook</td>
</tr>
<tr>
<td>11.55-12.15</td>
<td>Q&amp;A and Wrap Up</td>
</tr>
</tbody>
</table>

Jointly Organized by:
Consilium Technology
We’re leaders in artificial intelligence

Founded
ADL, 2010

Employees
45+

Customers
Defence, Agriculture, Manufacturing, Mining and Resources
Machine Intelligence

Machine Learning

Modelling & Simulation

Data Analytics
DigitalGlobe AOI Selection
# add geojson colours

green_style = {'fillColor': '#228822', 'color': '#228822'}
blue_style = {'fillColor': '#0077ff', 'color': '#0077ff'}
red_style = {'fillColor': '#ff0000', 'color': '#ff0000'}

# define centroid and zoom level
liverpool_zoomed_out_map = folium.Map([-33.87, 150.92], zoom_start=10)

# add full image and AOI geojson files to map
folium.GeoJson(liverpool_2020_image, name="full image", style_function=lambda x:blue_style).add_to(liverpool_zoomed_out_map)
folium.GeoJson(shapecy_geom, name="Original AOI", style_function=lambda x:green_style).add_to(liverpool_zoomed_out_map)

# plugin for mini map in bottom-left corner
minimap = plugins.MinMap(toggle_display=True)

# add minimap to map
liverpool_zoomed_out_map.add_child(minimap)

# add full screen button to map
pluginsFullscreen(position='topright').add_to(liverpool_zoomed_out_map)

# display folium map
liverpool_zoomed_out_map
Verifying and Ordering Images

# confirm image ID has been ordered
gbdx.catalog_image.is_ordered('104001005E88B700')

True

gbdx.catalog_image.is_ordered('1040010044480F00')

False

# if false, order image by entering the catalog ID into gbdx.ordering.order
order_id = gbdx.ordering.order('1040010044480F00')

b319be9-5f81-4dc1-8ad5-b60712da80d5

# Ordering will not occur instantly (usually takes hours)
# check the status of the order by entering the order ID into gbdx.ordering.status
order_status = gbdx.ordering.status(order_id)

[{'acquisition_id': '1040010044480F00',
  'state': 'submitted',
  'location': 'not_delivered'}]

gbdx.ordering.status(order_id)

[{'acquisition_id': '1040010044480F00',
  'state': 'delivered',
  'location': 's3://receiving-dgcs-tiogplatform-com/012897493010_01_003'}]

# determine number of pixels in image
liverpool_2020_catalog_id = '104001005E88B700'
liverpool_2020_image = CatalogImage(liverpool_2020_catalog_id)
bands_2020, height_2020, width_2020 = liverpool_2020_image.shape
liverpool_2020_image.shape

(8, 42157, 14004)

# number of pixels in whole image
num_pixels_2020 = bands_2020 * height_2020 * width_2020
num_pixels_2020

4834965024
View Multispectral, Panchromatic & Pansharpened Images

# multispectral image
ms_zoomed_2020_img = CatalogImage(liverpool_2020_catalog_id, band_type='MS', geojson=zoomed_shapely_geom)

# panchromatic image
pan_zoomed_2020_img = CatalogImage(liverpool_2020_catalog_id, band_type='Pan', geojson=zoomed_shapely_geom)

# pansharpened image
psharp_zoomed_2020_img = CatalogImage(liverpool_2020_catalog_id, pansharpen=True, geojson=zoomed_shapely_geom)

# display the 2020 multispectral image
ms_zoomed_2020_img.plot(w=10, h=5)

# display the 2020 panchromatic image
pan_zoomed_2020_img.plot(w=10, h=5)

# display the 2020 pansharpened image
psharp_zoomed_2020_img.plot(w=10, h=5)
# apply unique colour scales to each band

arrangement = "horizontal"

if arrangement == "horizontal":
    rows = 4
    cols = 2
    figsize = (30,8)
    font = 20

elif arrangement == "vertical":
    rows = 2
    cols = 4
    figsize = (30,25)
    font = 30

colourmaps = ['YlGnBu', 'Blues', 'Greens', 'Wistia', 'Reds', 'RdPu', 'Greys', 'Greys']
plt.figure(figsize=figsize)

for i, alias in zip(range(8), band_aliases):
    plt.subplot(cols, rows, i+1)
    plt.rc('font', size=font)
    plt.axis('off')
    plt.imshow(psharp_zoomed_2020_img[i,...], cmap=colourmaps[i])
    plt.title("{} Band".format(band_info[alias]['name']))

plt.show()
8 Bands Plotted with Unique Colour Scales
8 Bands Plotted with Consistent Colour Scale - Code

```python
# apply consistent colour scale to all bands

arrangement = "horizontal"

if arrangement == "horizontal":
    rows = 2
    cols = 4
    figsize = (30,10)
    pad = 0.1
    font = 15
elif arrangement == "vertical":
    rows = 4
    cols = 2
    figsize = (30,30)
    pad=0.01
    font = 30

count = 0
fig, axs = plt.subplots(rows,cols,figsize=figsize)
plt.rc('font',size=font)
plt.rc('xtick',labelsize=25)

for row in range(rows):
    for col in range(cols):
        axs[row,col].set_title(band_info[band_aliases[count]]['name'])
        count = count + 1

for idx,band,ax in zip(range(rows*cols),band_aliases,axs.flat):
    pcm = ax.pcolormesh(sharp_zoomed_2020_img[idx,...],cmap="viridis",label=band)
    ax.xaxis.set_visible(False)
    ax.yaxis.set_visible(False)

fig.colorbar(pcm,ax=axs[:,:,:],location='bottom',pad=pad)
plt.show()
```
8 Bands Plotted with Consistent Colour Scale
# assign Red and NIR bands to variables
NIR_band_2020 = data_2020_img[6]
red_band_2018 = data_2018_img[4]
NIR_band_2018 = data_2018_img[6]

# calculate NDVI with NDVI=(NIR-Red)/(NIR+Red)
NDVI_2020 = (NIR_band_2020 - red_band_2020)/(NIR_band_2020 + red_band_2020)

# display the 2020 NDVI image
plt.figure(figsize=(16,16))
plt.title("NDVI for 2020 Image")
plt.imshow(NDVI_2020,cmap='RdYlGn')

# display the 2018 NDVI image
plt.figure(figsize=(16,16))
plt.title("NDVI for 2018 Image")
plt.imshow(NDVI_2018,cmap='RdYlGn')
2018 and 2020 NDVI Comparison
NDVI Difference - Code

```python
# does not work as arrays are different sizes
NDVI_2018 - NDVI_2020

ValueError: operands could not be broadcast together with shapes (2068,5540) (1760,4715)

# rescale the 2020 NDVI image to match the size 2018 NDVI; scaling by a factor of 1.175
scaling_factor = NDVI_2018.shape[0]/NDVI_2020.shape[0]
NDVI_2020_scaled = scipy.ndimage.zoom(NDVI_2020, scaling_factor, order=0)

# calculate difference in NDVI between 2018 and 2020
NDVI_diff = NDVI_2018 - NDVI_2020_scaled

# plot the difference to identify changes in vegetation from 2018 to 2020
# red indicates a decrease, blue indicates an increase and green indicates minimal change in vegetation
plt.figure(figsize=(16,8))
plt.title("NDVI Difference between 2018 and 2020 Images")
plt.imshow(NDVI_diff,cmap='jet')
plt.colorbar(orientation='horizontal')
```
NDVI Difference

NDVI Difference between 2018 and 2020 Images
NDVI and Green Band Thresholding - Code

Thresholding based on NDVI only

```python
high_ndvi_2020 = NDVI_2020 > 0.65
plt.figure(figsize=(16,16))
plt.title("NDVI Thresholding")
plt.imshow(high_ndvi_2020)
```

Thresholding on Green Values only

```python
plt.figure(figsize=(16,16))
plt.title("Green Band Thresholding")
plt.imshow(high_green_2020)
```

NDVI AND Green Values Combined Thresholding

```python
healthy_vegetation_2020 = high_green_2020 & high_ndvi_2020
plt.figure(figsize=(16,16))
plt.title("NDVI AND Green Band Thresholding")
plt.imshow(healthy_vegetation_2020)
```
NDVI and Green Band Thresholding

NDVI Thresholding

Green Band Thresholding

NDVI AND Green Band Thresholding
Creating and Uploading Files to S3 Bucket

Import satellite image into QGIS

```
# convert image into georeferenced tif (GeoTIFF)
ms_liverpool_tif_2020 = ms_zoomed_2020_img.geotiff(path="ms_liverpool_tif_2020.tif", proj="EPSG:4326")

# upload file from workspace into S3 bucket
gbdx.s3.upload("ms_liverpool_tif_2020.tif", "ms_liverpool_tif_2020.tif")
```

Save satellite images as Pickle files

```
# save the NDVI arrays as pickle files and upload to the S3 bucket
with open("NDVI_2018.pkl", "wb") as file_name:
    pk1.dump(NDVI_2018, file_name)

with open("NDVI_2020.pkl", "wb") as file_name:
    pk1.dump(NDVI_2020, file_name)

gbdx.s3.upload("NDVI_2018.pkl", "NDVI_2018.pkl")
gbdx.s3.upload("NDVI_2020.pkl", "NDVI_2020.pkl")
```

'gbd-customer-data/58a1b8a6-6161-431e-92bc-9706ec5af096/NDVI_2020.pkl'
Files in S3 Bucket

<table>
<thead>
<tr>
<th>Name</th>
<th>Last modified</th>
<th>Size</th>
<th>Storage class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Info</td>
<td>Aug 3, 2020</td>
<td>5.0 KB</td>
<td>Standard</td>
</tr>
<tr>
<td>AreaOfInterest.gjson</td>
<td>Aug 3, 2020</td>
<td>1.3 KB</td>
<td>Standard</td>
</tr>
<tr>
<td>AreaOfInterestZoomedIn.gjson</td>
<td>Aug 3, 2020</td>
<td>1.5 KB</td>
<td>Standard</td>
</tr>
<tr>
<td>NDLV2019.pbf</td>
<td>Aug 3, 2020</td>
<td>43.7 MB</td>
<td>Standard</td>
</tr>
<tr>
<td>NDLV2020.pbf</td>
<td>Aug 6, 2020</td>
<td>31.7 MB</td>
<td>Standard</td>
</tr>
<tr>
<td>liverpool.tif_2020.tif</td>
<td>Aug 3, 2020</td>
<td>18.0 KB</td>
<td>Standard</td>
</tr>
<tr>
<td>mx_liverpool.tif_2020.tif</td>
<td>Aug 5, 2020</td>
<td>15.8 KB</td>
<td>Standard</td>
</tr>
</tbody>
</table>
Machine Learning Classification Example - Labeling

Training Data (Coloured Masks)

- Buildings
- Trees
- Water
- Roads

Original Image

notebook:
https://notebooks.geobigdata.io/hub/notebooks/geayszwmx3e37sepu2il
Machine Learning Binary Classification Example - Output

Water Prediction (Dark Blue) Overlaid with Original Image
Machine Learning Multiclass Classification Example - Output

Multiclass Labels

Model Prediction
Thank You

Dr Sebastien Wong
Chief Technology Officer
sebastien.wong@consilium.technology