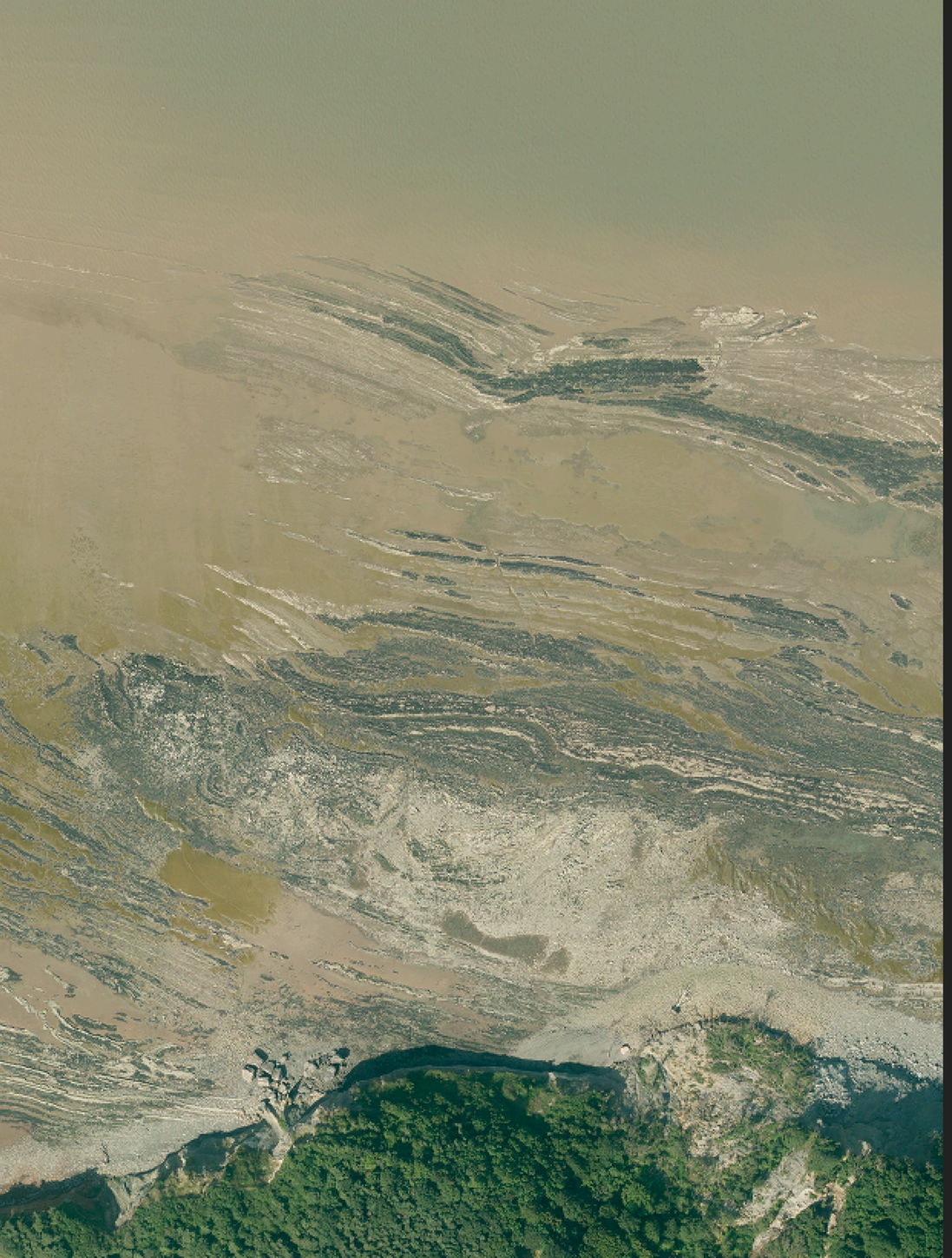
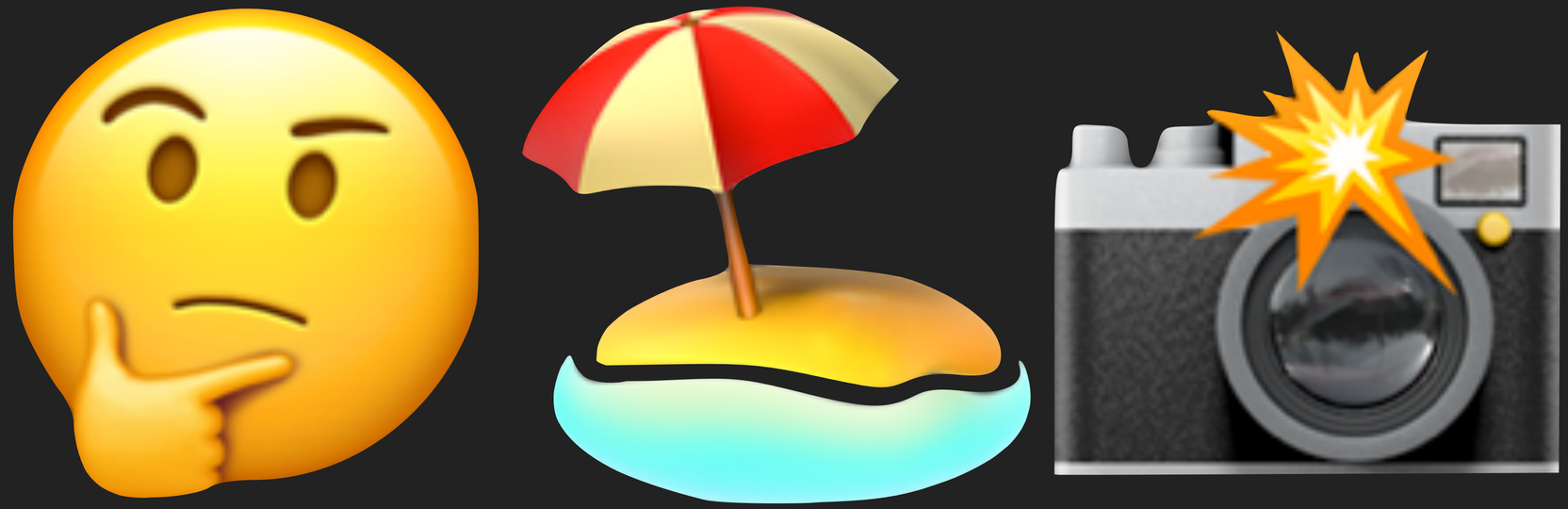


CIARAN EVANS
UK HYDROGRAPHIC OFFICE
DATA SCIENCE ACCELERATOR 2018/19

BEACH COMPOSITION CLASSIFICATION



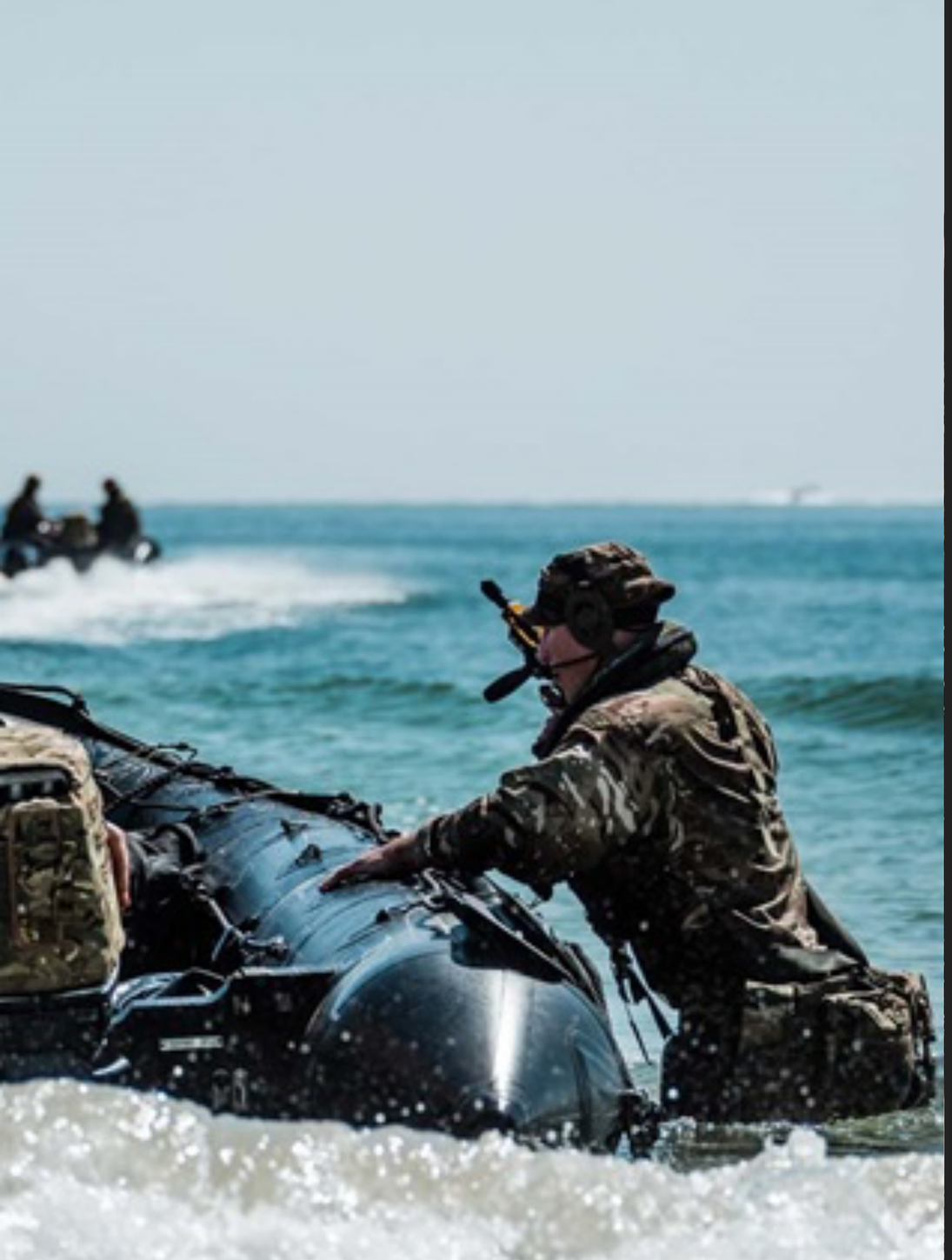
RAMBLING TOPICS

- Why?
- Data
- Data Science stuff
- Future plans

RAMBLING TOPICS

- Why?
- Data
- Data Science stuff
- Future plans

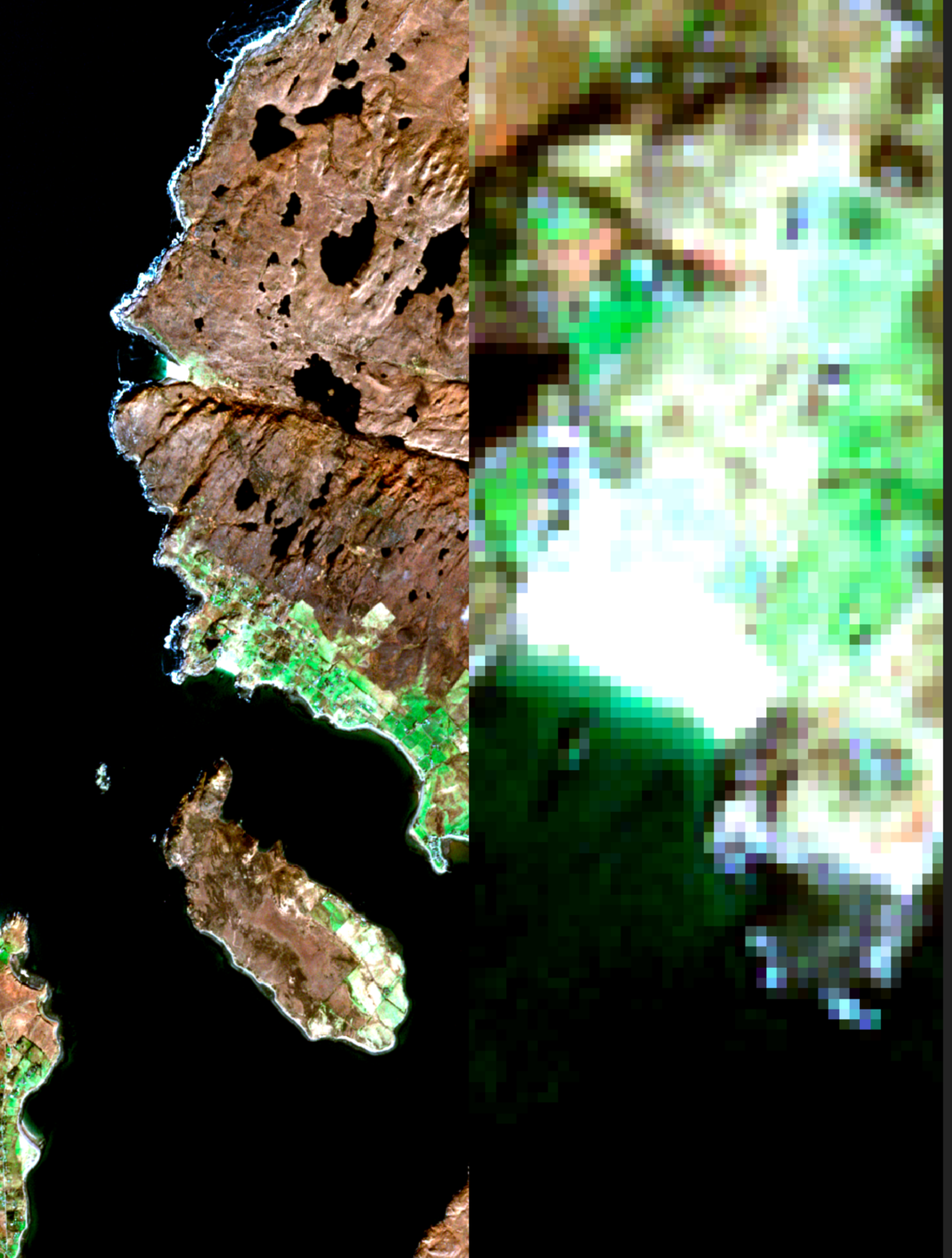
- Manually done
- Open source information (OSM, Google Maps)
- Satellite imagery & ground photography
- Vague sentences about composition
- ‘Sand’



- Automated
- Semantic segmentation (sand, rocks, mud)
- Spatially referenced
- Better than ‘eh, sand’

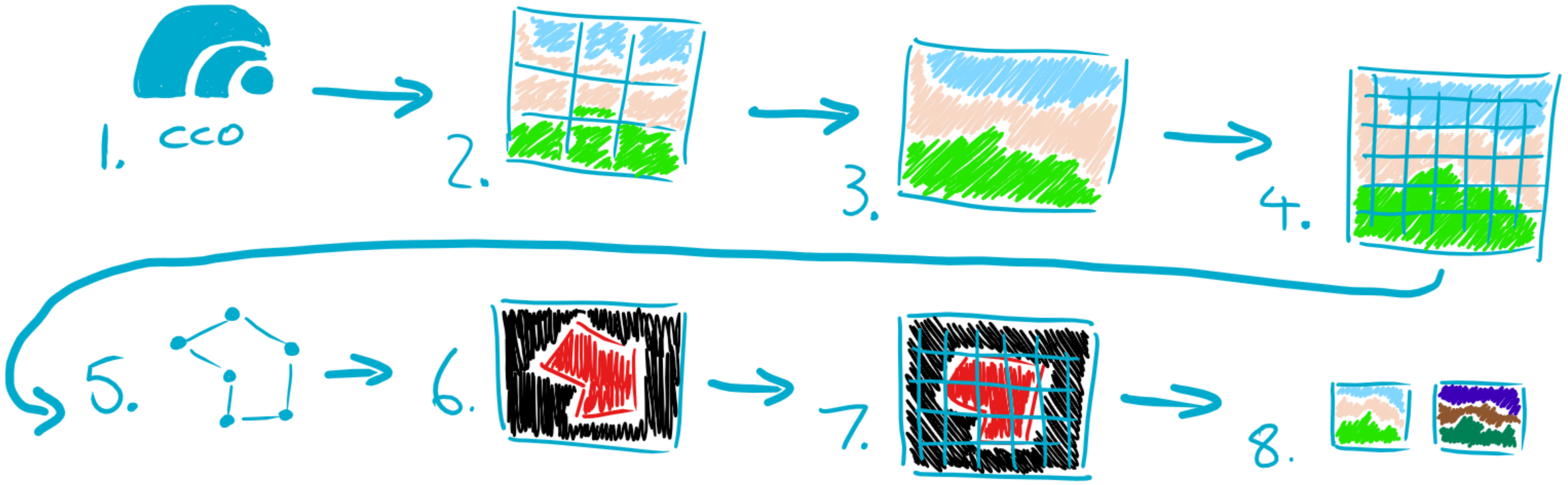
RAMBLING TOPICS

- Why?
- Data
- Data Science stuff
- Future plans



- ESA’s Sentinel 2
- Free!
- Happy face.
- 10m resolution
- Sad face.

- Channel Coast Observatory (CCO)
- Free!
- Happy face.
- 12.5cm resolution
- Even more happy face!
- Only covers English coast
- Small sad face.


1. cco
2.
3.
4.
5.
6.
7.
8.

```
import rasterio
import os
import numpy
import fiona
from shapely import geometry
from rasterio.mask import mask
from rasterio.merge import merge
from rasterio.plot import show, show_hist
from rasterio import features
import matplotlib
%matplotlib inline
```

```
BEACH_AERIAL_PATH = "/Users/ciaran/data/imagery/cco/seaton/cco_data-20190107184417/data/aerial"
BEACH_GENERATED_GEOTIFF_PATH_WITHOUT_EXT = "/Users/ciaran/data/imagery/tifs/seaton/seaton_2012"
BEACH_GENERATED_GEOTIFF_PATH = BEACH_GENERATED_GEOTIFF_PATH_WITHOUT_EXT + ".tif"
BEACH_CELL_PATH_WITHOUT_EXT = "/Users/ciaran/data/imagery/tifs/seaton/train/images/seaton_"
BEACH_CELL_LABELS_PATH_WITHOUT_EXT = "/Users/ciaran/data/imagery/tifs/blue_anchor/train/labels/blue_anchor_"
BEACH_ALL_CELLS_PATH = "/Users/ciaran/data/imagery/tifs/blue_anchor/cells/*"
NO_DATA_SHAPEFILE_PATH = "/Users/ciaran/data/shapefiles/blue_anchor/no_data.shp"
ROCK_SHAPEFILE_PATH = "/Users/ciaran/data/shapefiles/blue_anchor/rock.shp"
SAND_SHAPEFILE_PATH = "/Users/ciaran/data/shapefiles/blue_anchor/sand.shp"
PEBBLE_SHAPEFILE_PATH = "/Users/ciaran/data/shapefiles/blue_anchor/pebble.shp"
PEBBLE_2_SHAPEFILE_PATH = "/Users/ciaran/data/shapefiles/blue_anchor/pebble_2.shp"
RASTERISED_ROCK_PATH = "/Users/ciaran/data/imagery/tifs/blue_anchor/rock.tif"
RASTERISED_SAND_PATH = "/Users/ciaran/data/imagery/tifs/blue_anchor/sand.tif"
RASTERISED_PEBBLE_PATH = "/Users/ciaran/data/imagery/tifs/blue_anchor/pebble.tif"
RASTERISED_PEBBLE_2_PATH = "/Users/ciaran/data/imagery/tifs/blue_anchor/pebble_2.tif"
RASTERISED_NO_DATA_PATH = "/Users/ciaran/data/imagery/tifs/blue_anchor/no_data.tif"
MERGED_LABELS_PATH = "/Users/ciaran/data/imagery/tifs/blue_anchor/merged_labels.tif"
BLUE_ANCHOR_LABELLED_PATH = "/Users/ciaran/data/imagery/tifs/blue_anchor/blue_anchor_2013_labels.tif"
NO_DATA_VALUE=115
ROCK_VALUE = 100
SAND_VALUE = 105
PEBBLE_VALUE = 110
```

Channel Coastal Observatory

[Help] [Reset] [Feedback]

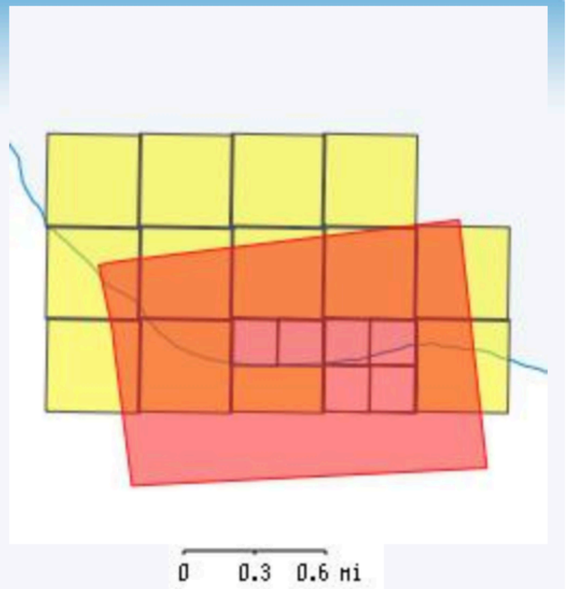
1 Search by geographic area 2 Refine your search 3 View your results

Summary of results

Data type	Results	In basket
Aerial photography	79 (600 MB)	0

[View your results]
[Add all results to basket]
[Remove results from basket]
[View your basket]

Extent of result set



Extent of result set shown on current page (click on other results pages to view their extent)

MAP VIEWER & DATA SEARCH

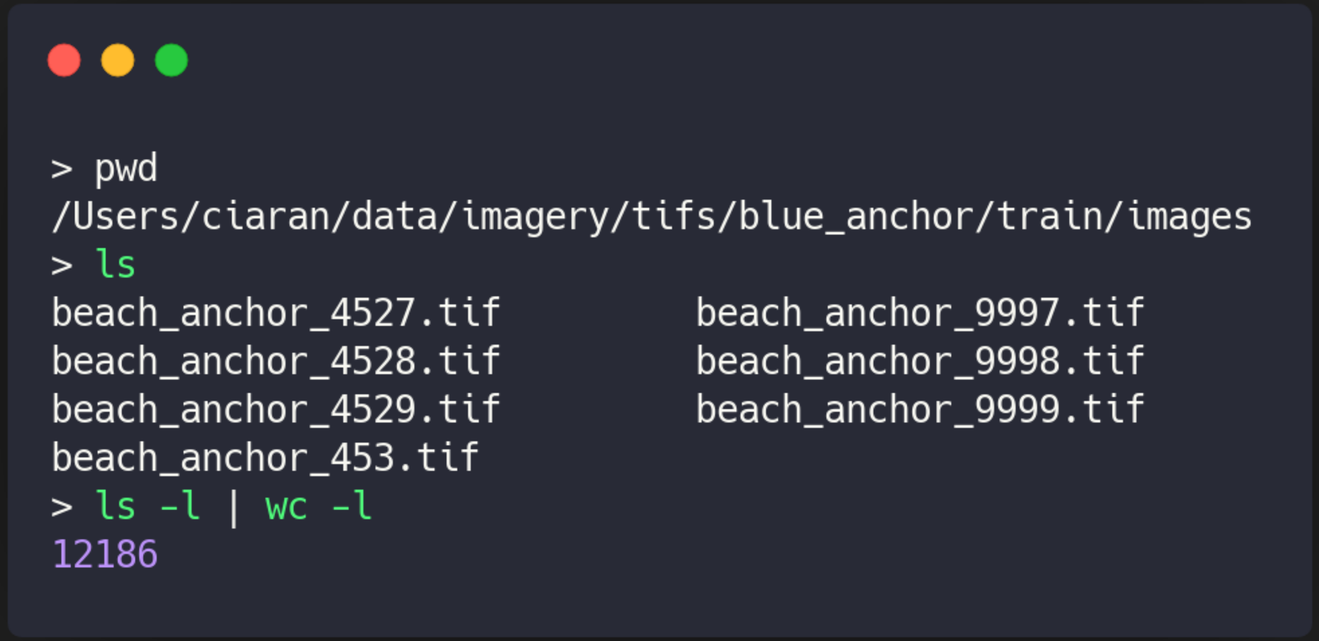
Back to map | Back to refine search | 3 View results

« 1-20 21-40 41-60 61-79 » 20 results per page

Date	Type	Details	Size	File name	Metadata	Data format
25/06/2013	aerial	ortho-rectified	16 MB	ST0444_20130822ortho.ecw	View	Enhanced Compressed Wavelet (ECW) raster fo
25/06/2013	aerial	ortho-rectified	14 MB	ST0245_20130822ortho.ecw	View	Enhanced Compressed Wavelet (ECW) raster fo
25/06/2013	aerial	ortho-rectified	54 MB	ST0143_20130822ortho.ecw	View	Enhanced Compressed Wavelet (ECW) raster fo
25/06/2013	aerial	ortho-rectified	71 MB	ST0443_20130822ortho.ecw	View	Enhanced Compressed Wavelet (ECW) raster fo
25/06/2013	aerial	ortho-rectified	5 MB	ST0345_20130822ortho.ecw	View	Enhanced Compressed Wavelet (ECW) raster fo
25/06/2013	aerial	ortho-rectified	56 MB	ST0343_20130822ortho.ecw	View	Enhanced Compressed Wavelet (ECW) raster fo
25/06/2013	aerial	ortho-rectified	60 MB	ST0044_20130822ortho.ecw	View	Enhanced Compressed Wavelet (ECW) raster fo
25/06/2013	aerial	ortho-rectified	41 MB	ST0243_20130822ortho.ecw	View	Enhanced Compressed Wavelet (ECW) raster fo
25/06/2013	aerial	ortho-rectified	43 MB	ST0144_20130822ortho.ecw	View	Enhanced Compressed Wavelet (ECW) raster fo
25/06/2013	aerial	ortho-rectified	23 MB	ST0244_20130822ortho.ecw	View	Enhanced Compressed Wavelet (ECW) raster fo
25/06/2013	aerial	ortho-rectified	16 MB	ST0344_20130822ortho.ecw	View	Enhanced Compressed Wavelet (ECW) raster fo
25/06/2013	aerial	ortho-rectified	67 MB	ST0045_20130822ortho.ecw	View	Enhanced Compressed Wavelet (ECW) raster fo
25/06/2013	aerial	ortho-rectified	28 MB	ST0145_20130822ortho.ecw	View	Enhanced Compressed Wavelet (ECW) raster fo
25/06/2013	aerial	ortho-rectified	45 MB	ST0043_20130822ortho.ecw	View	Enhanced Compressed Wavelet (ECW) raster fo
24/04/2010	aerial	ortho-rectified	105 KB	st0344sw_20100314ortho.ecw	View	Enhanced Compressed Wavelet (ECW) raster fo
24/04/2010	aerial	ortho-rectified	1 MB	st0243sw_20100523ortho.ecw	View	Enhanced Compressed Wavelet (ECW) raster fo
24/04/2010	aerial	ortho-rectified	405 KB	st0144nw_20100424ortho.ecw	View	Enhanced Compressed Wavelet (ECW) raster fo
24/04/2010	aerial	ortho-rectified	4 MB	st0443nw_20100314ortho.ecw	View	Enhanced Compressed Wavelet (ECW) raster fo
24/04/2010	aerial	ortho-rectified	577 KB	st0443se_20100314ortho.ecw	View	Enhanced Compressed Wavelet (ECW) raster fo

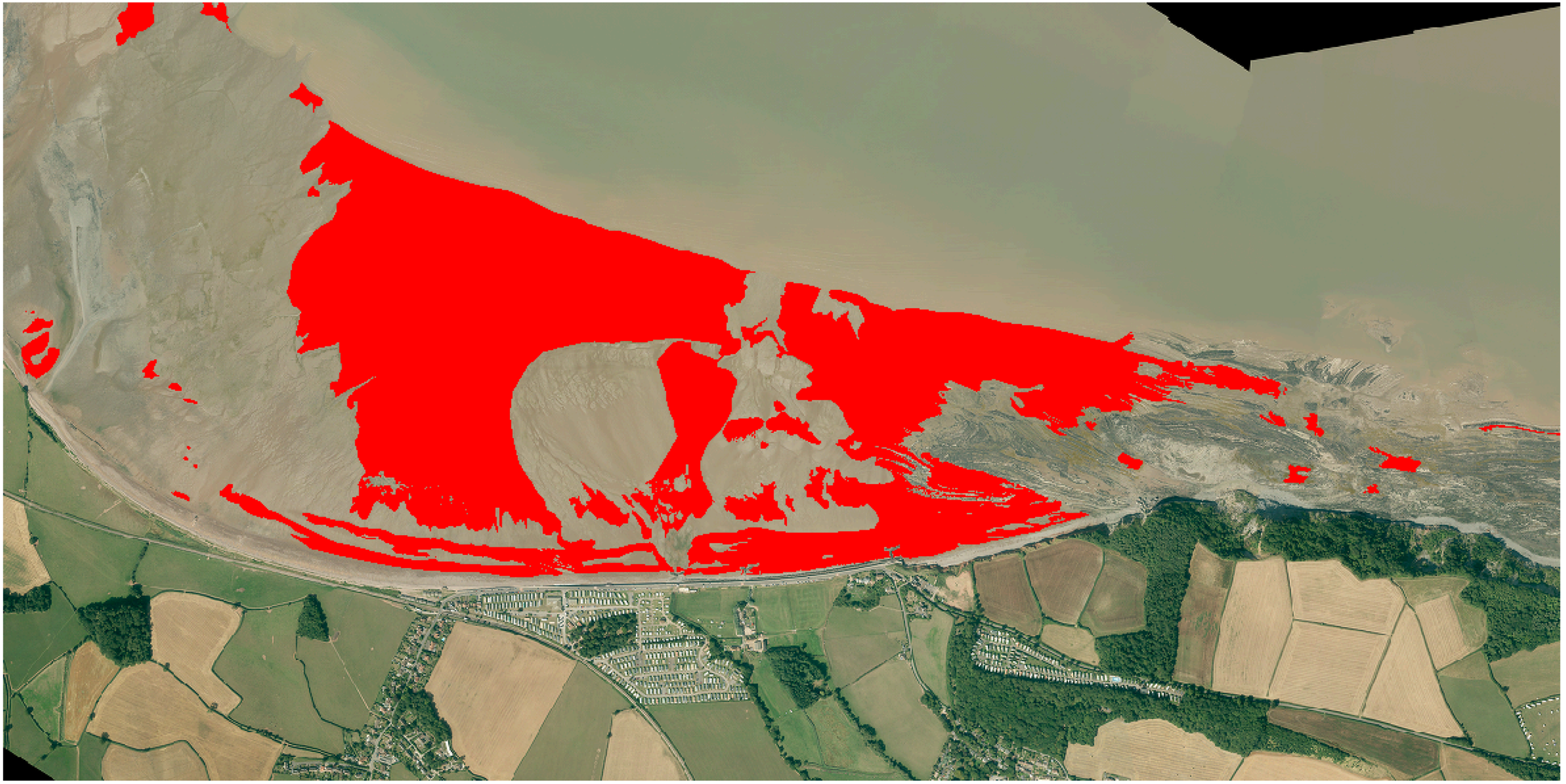
```python
# Making a whole beach from a bunch of beach sections

def generateBeachFromCells(pathToBeachImages, nameOfBeach):
    imagesToMerge = getAllImagesToMerge(pathToBeachImages)
    mergedImage, mergedImageTransform = merge(imagesToMerge)
    metadata = imagesToMerge[0].meta.copy()
    crs = imagesToMerge[0].crs
    writeImageAsGeoTIFF(mergedImage, mergedImageTransform, metadata, crs, nameOfBeach)

def getAllImagesToMerge(pathToBeachImages):
    return [rasterio.open(os.path.join(pathToBeachImages, image)) for image in
os.listdir(pathToBeachImages)]

def writeImageAsGeoTIFF(img, transform, metadata, crs, filename):
    metadata.update({"driver":"GTiff",
                     "height":img.shape[1],
                     "width":img.shape[2],
                     "transform": transform,
                     "crs": crs})
    with rasterio.open(filename+".tif", "w", **metadata) as dest:
        dest.write(img)

generateBeachFromCells(BEACH_AERIAL_PATH, BEACH_GENERATED_GEOTIFF_PATH_WITHOUT_EXT)
```

```python
# Take a big image and split it into squares of squareDim^2

def splitImageIntoCells(img, filename, squareDim):
    numberOfCellsWide = img.shape[1] // squareDim
    numberOfCellsHigh = img.shape[0] // squareDim
    x, y = 0, 0
    count = 0
    for hc in range(numberOfCellsHigh):
        y = hc * squareDim
        for wc in range(numberOfCellsWide):
            x = wc * squareDim
            geom = getTileGeom(img.transform, x, y, squareDim)
            getCellFromGeom(img, geom, filename, count)
            count = count + 1

def getTileGeom(transform, x, y, squareDim):
    corner1 = (x, y) * transform
    corner2 = (x + squareDim, y + squareDim) * transform
    return geometry.box(corner1[0], corner1[1],
                        corner2[0], corner2[1])

def getCellFromGeom(img, geom, filename, count):
    crop, cropTransform = mask(img, [geom], crop=True)
    writeImageAsGeoTIFF(crop,
                        cropTransform,
                        img.meta,
                        img.crs,
                        filename+str(count))

beach_raster = rasterio.open("beach.tif")
splitImageIntoCells(beach_raster, "beach_cell_", 256)
```

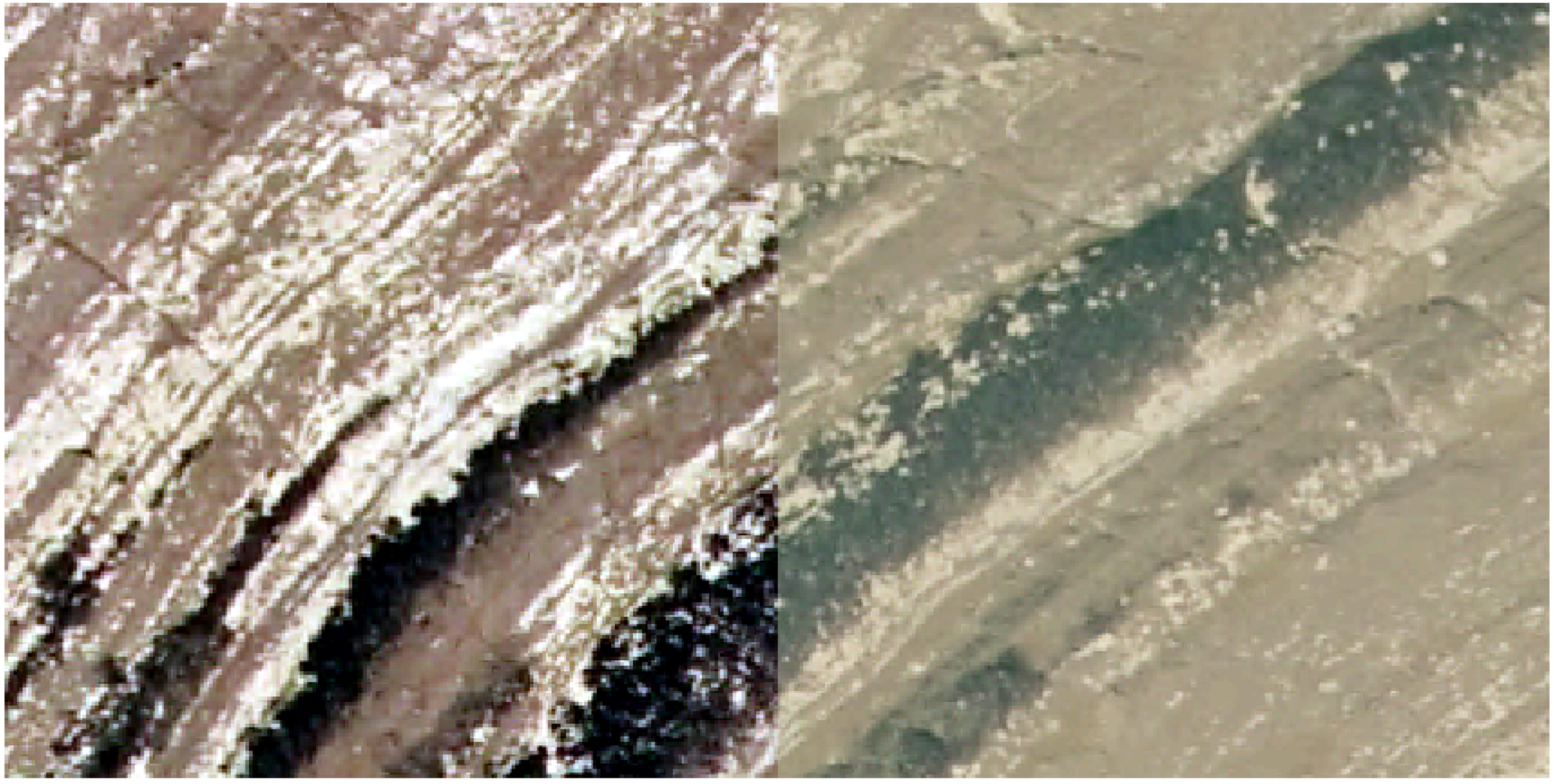

> pwd
/Users/ciaran/data/imagery/tifs/blue_anchor/train/images
> ls
beach_anchor_4527.tif
beach_anchor_4528.tif
beach_anchor_4529.tif
beach_anchor_453.tif
beach_anchor_9997.tif
beach_anchor_9998.tif
beach_anchor_9999.tif
> ls -l | wc -l
12186

```python
# Rasterise a shapefile (training label)

def getRasterisedShapefile(shapefilePath, baseCRS, baseMeta, baseShape, baseTransform, value):
    shapefile = fiona.open(shapefilePath)
    geometries = [shape['geometry'] for shape in shapefile if shape['geometry'] is not None]
    rasterisedShp = features.rasterize(geometries,
                                       out_shape=baseShape,
                                       transform=baseTransform,
                                       default_value=value)
    baseMeta.update({"driver":"GTiff",
                     "height":baseShape[0],
                     "width":baseShape[1],
                     "transform": baseTransform,
                     "crs": baseCRS,
                     "count": 1,
                     "nodata": 0})
    return (rasterisedShp, baseMeta)

def writeRasterisedShapefile(rasterisedShp, meta, filename):
    with rasterio.open(filename, "w", **meta) as dest:
        dest.write(rasterisedShp, 1)
```

```python
# Rasterise a shapefile (training label) #2

def rasteriseShapefileAndWriteToGeoTIFF(shapefileToRasterisePath, crs, meta, shape, transform,
                                        value, outPath):
    rasterisedShp, rasterisedShpMeta = getRasterisedShapefile(shapefileToRasterisePath,
                                                              crs,
                                                              meta,
                                                              shape,
                                                              transform,
                                                              value)
    writeRasterisedShapefile(rasterisedShp, rasterisedShpMeta, outPath)

rasteriseShapefileAndWriteToGeoTIFF(SAND_SHAPEFILE_PATH,
                                    blue_anchor.crs,
                                    blue_anchor.meta,
                                    blue_anchor.shape,
                                    blue_anchor.transform,
                                    SAND_VALUE,
                                    RASTERISED_SAND_PATH)
```



```
# Normalise training images from 0-255 to 0-1

for file in os.listdir("train/images"):
            if file.endswith(".tif"):
                image = rasterio.open(os.path.join("train/images", file))
                normalized_image = image.read()/255
                baseMeta = image.meta
                baseShape = image.shape
                baseCRS = image.crs
                baseTransform = image.transform
                baseMeta.update({"driver":"GTiff",
                                 "height":baseShape[0],
                                 "width":baseShape[1],
                                 "transform": baseTransform,
                                 "crs": baseCRS,
                                 "dtype": 'float64'})

                with rasterio.open(os.path.join("train/normalised_images", file),"w", **baseMeta) as dest:
                    dest.write(normalized_image)
                image = None
                normalized_image = None
```



- Training data
- Labels
- Spatially referenced
- Repeatable

RAMBLING TOPICS

- Why?
- Data
- Data Science stuff
- Future plans

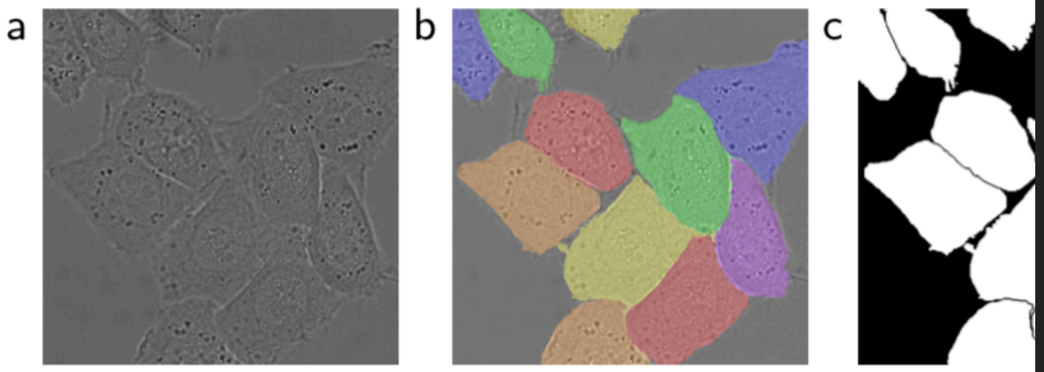


Fig. 3. HeLa cells on glass recorded with DIC (diffe
croscopy. (**a**) raw image. (**b**) overlay with ground tru
indicate different instances of the HeLa cells. (**c**) gene
foreground, black: background). (**d**) map with a pi
network to learn the border pixels.

where $\ell : \Omega \to \{1, \ldots, K\}$ is the true label of
a weight map that we introduced to give some
training.

We pre-compute the weight map for each grou
pensate the different frequency of pixels from
data set, and to force the network to learn the sm
introduce between touching cells (See Figure 3c

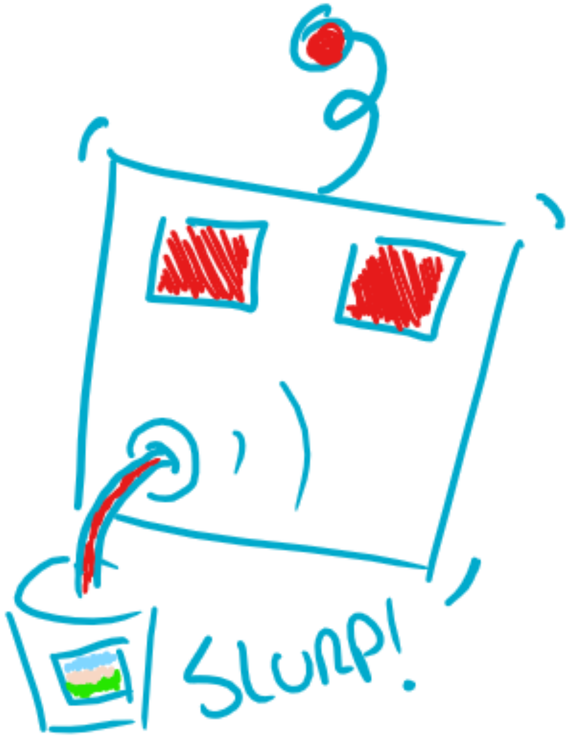
The separation border is computed using
weight map is then computed as

- U-Net
- Proven successes in Biomedical usages
- Proven successes in another UKHO Accelerator project

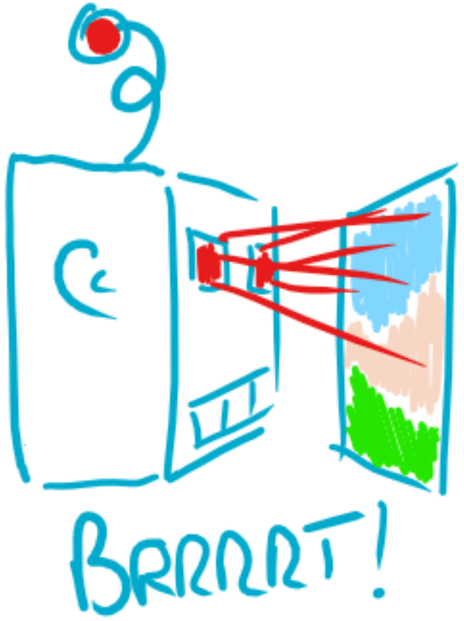
1.

2.

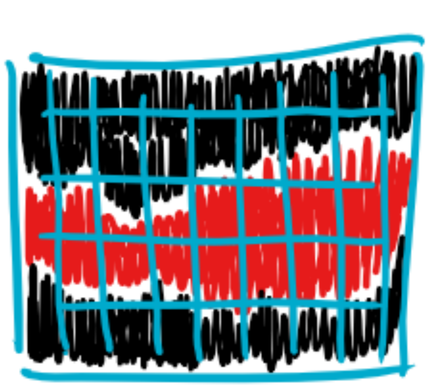
3.



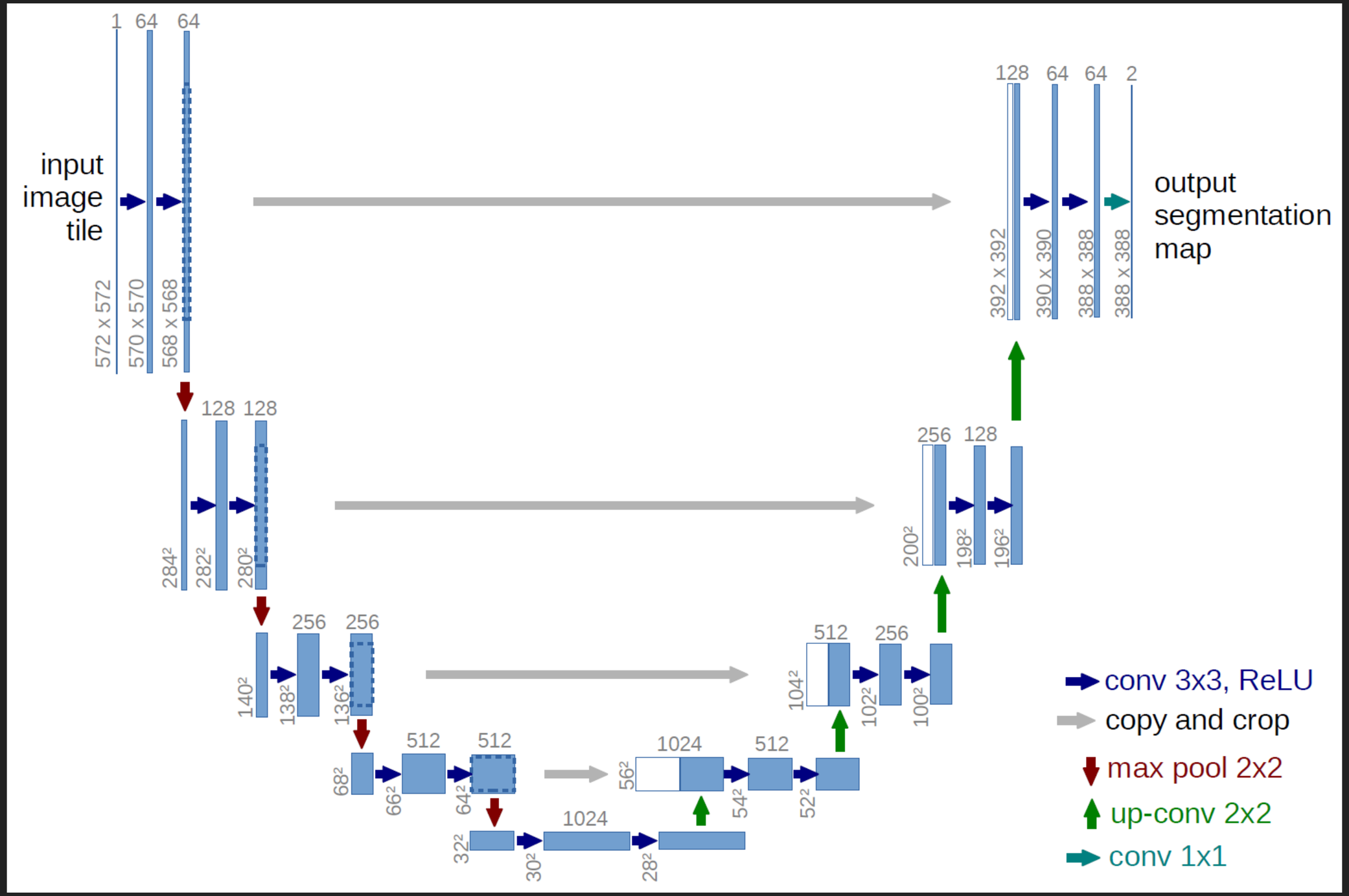
4.



5.



```
import os
import skimage.io as io
import skimage.transform as trans
import random
from keras import metrics
from keras import backend as K
from keras.initializers import *
from keras.layers import *
from keras.models import *
from keras.models import load_model
from keras.optimizers import *
from keras.losses import binary_crossentropy
from keras.metrics import binary_crossentropy as bc_met
from keras.callbacks import ModelCheckpoint, LearningRateScheduler

random.seed('beachclassification')
model_initializer = glorot_uniform
kernel_init = model_initializer(seed=random.randint(0,5))
model_learning_rate = 0.00001
```


input image tile
output segmentation map
1
64
64
572 x 572
570 x 570
568 x 568
128
64
64
2
392 x 392
390 x 390
388 x 388
388 x 388
128
128
284²
282²
280²
256
128
200²
198²
196²
256
256
140²
138²
136²
512
256
104²
102²
100²
512
512
68²
66²
64²
1024
512
56²
54²
52²
1024
32²
30²
28²
conv 3x3, ReLU
copy and crop
max pool 2x2
up-conv 2x2
conv 1x1

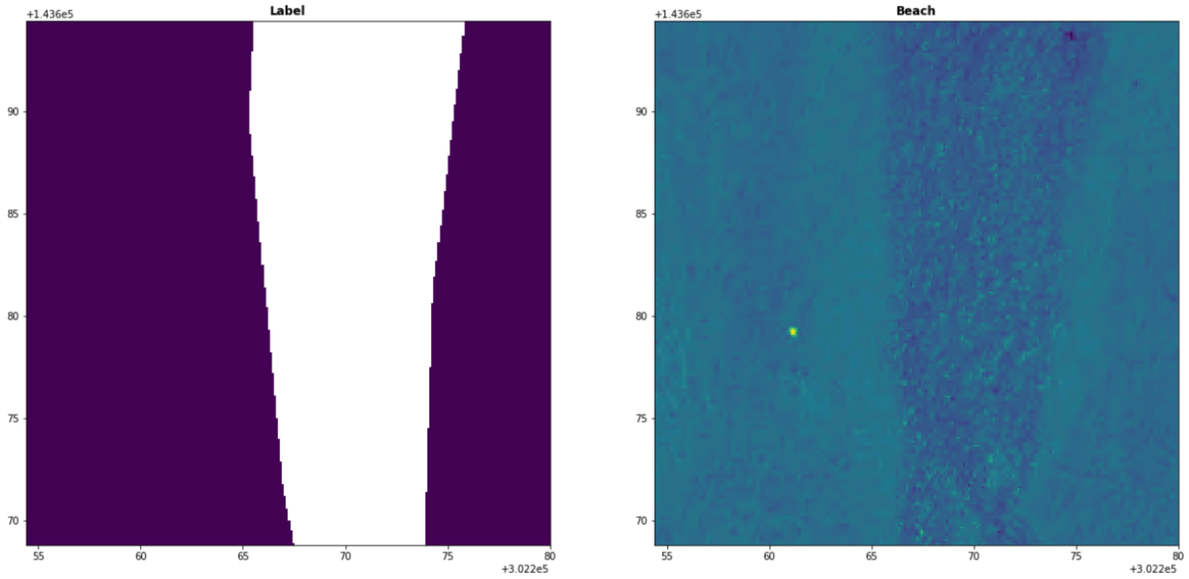
MACHINE LEARNING IS JUST TYPING ‘IMPORT MACHINE LEARNING, THEN MACHINE LEARNING DOT DO’

Ciaran ‘Knows nothing about ML’ Evans

```
def generateUnet():
    inputs = Input(shape=(256,256,4))
    conv1 = Conv2D(64, 3, activation = 'relu', padding = 'same',
kernel_initializer = 'he_normal')(inputs)
    conv1 = Conv2D(64, 3, activation = 'relu', padding = 'same',
kernel_initializer = 'he_normal')(conv1)
    pool1 = MaxPooling2D(pool_size=(2, 2))(conv1)
    conv2 = Conv2D(128, 3, activation = 'relu', padding = 'same',
kernel_initializer = 'he_normal')(pool1)
    conv2 = Conv2D(128, 3, activation = 'relu', padding = 'same',
kernel_initializer = 'he_normal')(conv2)
    pool2 = MaxPooling2D(pool_size=(2, 2))(conv2)
    conv3 = Conv2D(256, 3, activation = 'relu', padding = 'same',
kernel_initializer = 'he_normal')(pool2)
    conv3 = Conv2D(256, 3, activation = 'relu', padding = 'same',
kernel_initializer = 'he_normal')(conv3)
    pool3 = MaxPooling2D(pool_size=(2, 2))(conv3)
    conv4 = Conv2D(512, 3, activation = 'relu', padding = 'same',
kernel_initializer = 'he_normal')(pool3)
    conv4 = Conv2D(512, 3, activation = 'relu', padding = 'same',
kernel_initializer = 'he_normal')(conv4)
    drop4 = Dropout(0.5)(conv4)
    pool4 = MaxPooling2D(pool_size=(2, 2))(drop4)

    conv5 = Conv2D(1024, 3, activation = 'relu', padding = 'same',
kernel_initializer = 'he_normal')(pool4)
    conv5 = Conv2D(1024, 3, activation = 'relu', padding = 'same',
kernel_initializer = 'he_normal')(conv5)
    drop5 = Dropout(0.5)(conv5)

    up6 = Conv2D(512, 2, activation = 'relu', padding = 'same',
kernel_initializer = 'he_normal')(UpSampling2D(size = (2,2))(drop5))
    merge6 = concatenate([drop4,up6], axis = 3)
    conv6 = Conv2D(512, 3, activation = 'relu', padding = 'same',
kernel_initializer = 'he_normal')(merge6)
    conv6 = Conv2D(512, 3, activation = 'relu', padding = 'same',
kernel_initializer = 'he_normal')(conv6)

    up7 = Conv2D(256, 2, activation = 'relu', padding = 'same',
kernel_initializer = 'he_normal')(UpSampling2D(size = (2,2))(conv6))
    merge7 = concatenate([conv3,up7], axis = 3)
    conv7 = Conv2D(256, 3, activation = 'relu', padding = 'same',
kernel_initializer = 'he_normal')(merge7)
    conv7 = Conv2D(256, 3, activation = 'relu', padding = 'same',
kernel_initializer = 'he_normal')(conv7)

    up8 = Conv2D(128, 2, activation = 'relu', padding = 'same',
kernel_initializer = 'he_normal')(UpSampling2D(size = (2,2))(conv7))
    merge8 = concatenate([conv2,up8], axis = 3)
    conv8 = Conv2D(128, 3, activation = 'relu', padding = 'same',
kernel_initializer = 'he_normal')(merge8)
    conv8 = Conv2D(128, 3, activation = 'relu', padding = 'same',
kernel_initializer = 'he_normal')(conv8)

    up9 = Conv2D(64, 2, activation = 'relu', padding = 'same',
kernel_initializer = 'he_normal')(UpSampling2D(size = (2,2))(conv8))
    merge9 = concatenate([conv1,up9], axis = 3)
    conv9 = Conv2D(64, 3, activation = 'relu', padding = 'same',
kernel_initializer = 'he_normal')(merge9)
    conv9 = Conv2D(64, 3, activation = 'relu', padding = 'same',
kernel_initializer = 'he_normal')(conv9)
    conv9 = Conv2D(2, 3, activation = 'relu', padding = 'same',
kernel_initializer = 'he_normal')(conv9)
    conv10 = Conv2D(1, 1, activation = 'sigmoid')(conv9)

    model = Model(inputs=inputs,outputs=conv10)

    model.compile(optimizer = Adam(lr = model_learning_rate), loss =
'binary_crossentropy', metrics = ['accuracy'])

    return model
```

```
model = generateRealUnet()
training_images = getImagery()
training_labels = getLabels()

t_board = TensorBoard(log_dir="./models/1e-5/logs", histogram_freq=0,
                      write_grads=False, write_graph=True, write_images=False)
stop_early = EarlyStopping(patience=30)

# See! MachineLearning.do()
model.fit(training_images, training_labels, validation_split=0.2,
          epochs=30, batch_size=16, shuffle=True, callbacks=[stop_early, t_board])

model.save("./models/1e-5/model")

-------------------------------------------------------------------------------------

Train on 2400 samples, validate on 600 samples
Epoch 1/30
2400/2400 [==============================] - 174s 72ms/step - loss: 0.6537 - acc: 0.9817 -
val_loss: 0.6556 - val_acc: 0.9320
Epoch 2/30
2400/2400 [==============================] - 168s 70ms/step - loss: 0.6504 - acc: 0.9842 -
val_loss: 0.6547 - val_acc: 0.9320
```

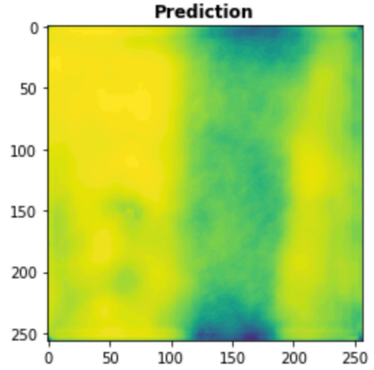
```
image_r = rasterio.open("/home/paperspace/ciaran/data/train_normalised/blue_anchor_"+str(no)+
".tif")
label_r = rasterio.open("/home/paperspace/ciaran/data/label/blue_anchor_"+str(no)+".tif")
image = io.imread("/home/paperspace/ciaran/data/train_normalised/blue_anchor_"+str(no)+".tif"
)
label = io.imread("/home/paperspace/ciaran/data/label/blue_anchor_"+str(no)+".tif")
fig, (axl, axb) = pyplot.subplots(1, 2, figsize=(21,21))
show(label_r, ax=axl, title="Label")
show(image_r, ax=axb, title="Beach")
pyplot.show()
```



In [4]:
```
image.shape
```

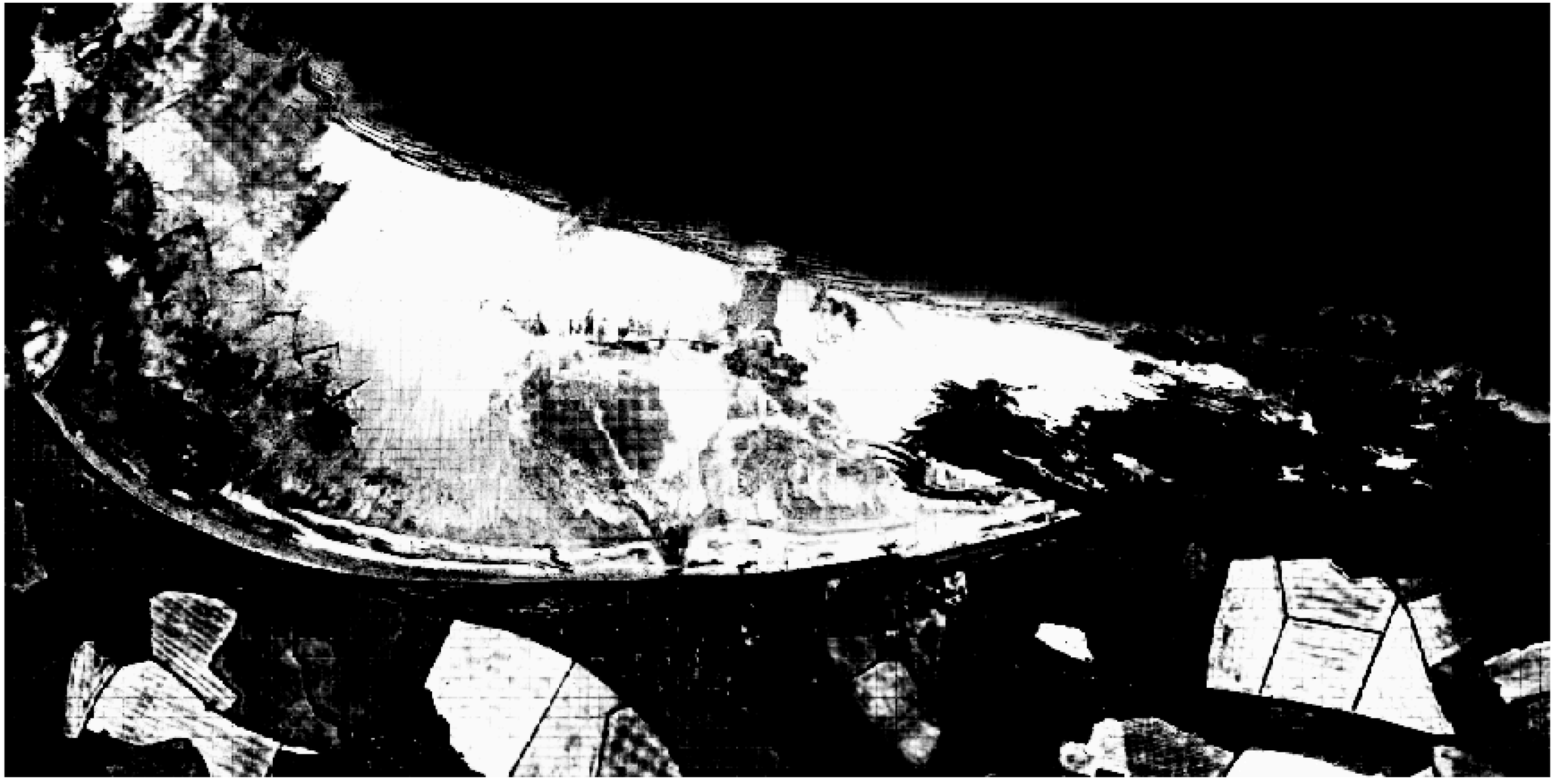
Out[4]: (256, 256, 4)

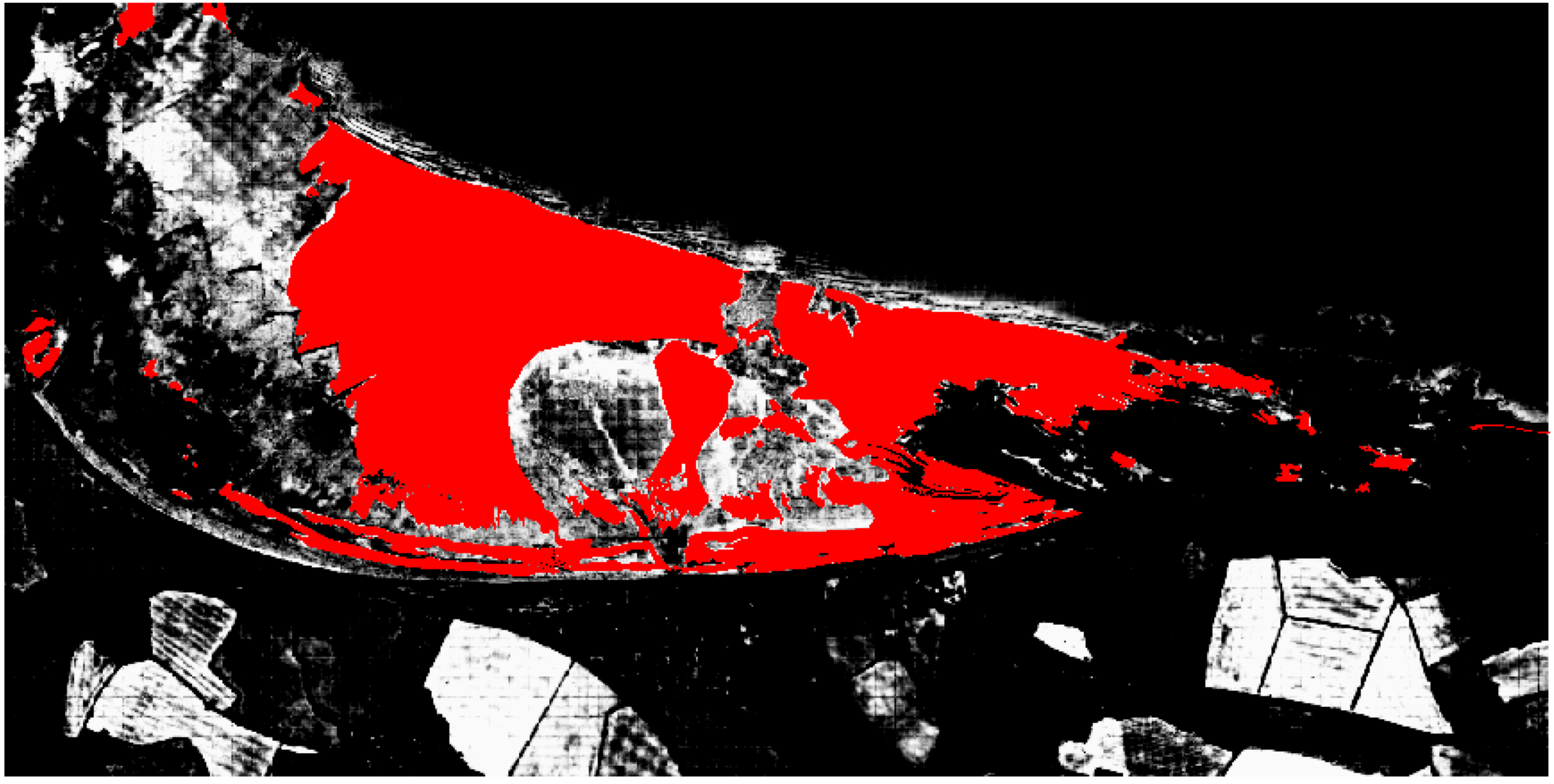
In [28]:
```
stuff = model.predict(x=np.array([image]))
show(stuff[0], title="Prediction")
stuff[0].shape
```

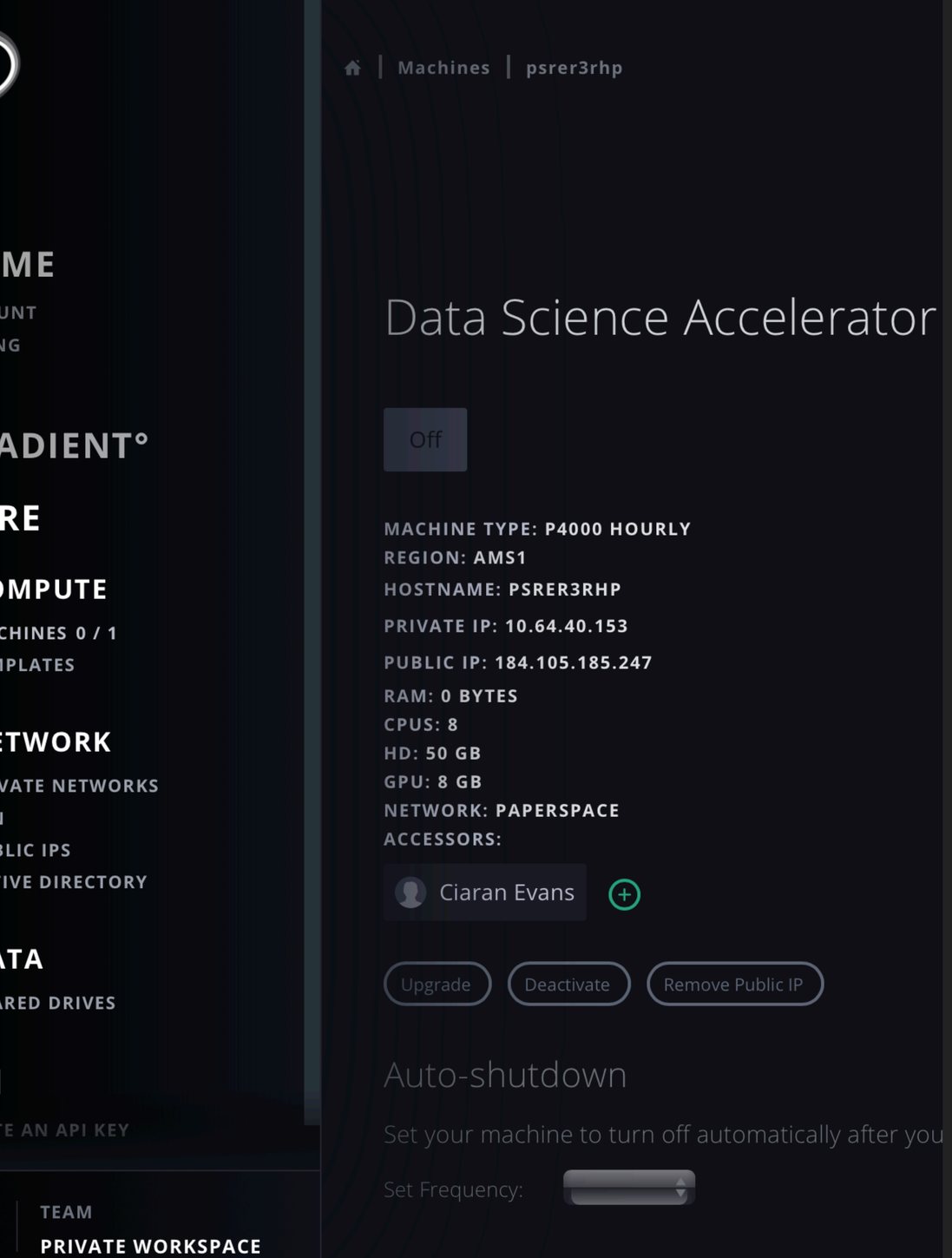


Out[28]: (256, 256, 1)

```
# Predicting all the things!

model = load_model('./models/1e-5_subset_2/model')
predictedImages = [file for file in os.listdir(PREDICTIONS_DIR)]

for file in os.listdir(IMAGES_TO_PREDICT_DIR):
            if file.endswith(".tif") and file not in predictedImages:
                imageToPredict = rasterio.open(os.path.join(IMAGES_TO_PREDICT_DIR, file)).read()
                imageToPredict = np.moveaxis(imageToPredict, 0, -1)
                imageToPredict = np.array([imageToPredict])
                prediction = model.predict(imageToPredict)[0]
                metadata = rasterio.open(os.path.join(IMAGES_TO_PREDICT_DIR, file)).meta
                metadata['count'] = 1
                metadata['dtype'] = 'float32'
                prediction = np.moveaxis(prediction, -1, 0)
                with rasterio.open(os.path.join(PREDICTIONS_DIR, file), "w", **metadata) as dest:
                    dest.write(prediction)
```







- Paperspace
- Cloud GPU instances
- Pre-built Fast.ai box
- Lots of downgrading needed
- $5/mth + $0.51/hr

RAMBLING TOPICS

- Why?
- Data
- Data Science stuff
- Future plans

- Create E2E pipeline
- Label more imagery
- Trial using unsupervised learning for making training data
- Generate models for other classes - rocks/mud
- Trial using CCO data to mimic S2 data