BusinessCase

October 9, 2019

1 Images of satellites classification

```
In [64]: from google_images_download import google_images_download   #importing the library
         import pandas
         import cv2 #to load and displays images
         import matplotlib.pyplot as plt
         import imutils
         import os
         import pandas as pd
```

1.1 Download images of satellites / non satellites

```
In [35]: nb_images = 2000 # The number of images to download (both from satellite and non satell
         download = False  # Are we doanloading the images ?
         chromeDriverPath = "/usr/bin/chromedriver"
```

```
In [36]: def downloadImages(tags, nbImgs):
             response = google_images_download.googleimagesdownload()   #class instantiation

             arguments = {"keywords":','.join(tags),"limit":nbImgs,"print_urls":False, "extract_
             if download:
                 response.download(arguments)   #passing the arguments to the function
```

download list of satellites categories

```
In [37]: # categories taken from https://www.omicsonline.org/conferences-list/types-of-satellite
         categories = ["Communications Satellite","Remote Sensing Satellite","Navigation Satelli
                       "MEO satellite", "HEO satellite","GPS satellite","GEO satellite","Drone S
                       "Polar Satellite","Nano Satellites","CubeSats","SmallSats"]
```

```
In [38]: downloadImages(categories, nb_images)
```

```
In [39]: catpd = [pandas.read_json("logs/" + str(cat) + ".json") for cat in categories]
```

1.1.1 download of images

```
In [40]: downloadImages(["satellite","images"], nb_images)
```

```
In [41]: imgpd = pandas.read_json("logs/images.json")
         satpd = pandas.read_json("logs/satellite.json")

In [42]: imgpd.image_filename.count(), satpd.image_filename.count()

Out[42]: (397, 398)

In [43]: satpd.iloc[0]

Out[43]: image_description      Virgin Orbit Tests its Satellite-Delivery Rock...
         image_filename                          virgin-orbit-first-release-2.jpg
         image_format                                                         jpg
         image_height                                                         615
         image_host                                                universetoday.com
         image_link              https://www.universetoday.com/wp-content/uploa...
         image_source            https://www.universetoday.com/142812/virgin-or...
         image_thumbnail_url     https://encrypted-tbn0.gstatic.com/images?q=tb...
         image_width                                                         1000
         Name: 0, dtype: object
```

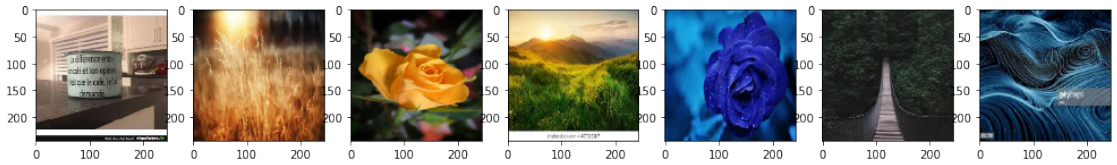
1.1.2 display of the first images

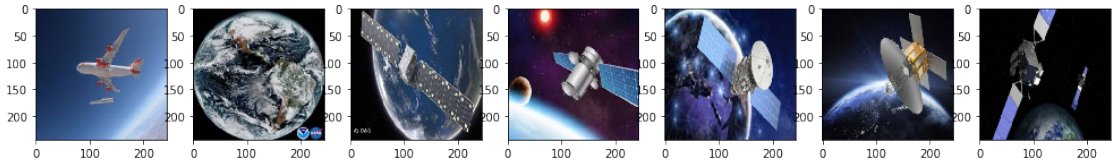
```
In [44]: def draw_imgs(imgs_list):
             l = int(len(imgs_list))
             n = int(len(imgs_list[0]))
             _, axs = plt.subplots(l, n, figsize=(17, 17))
             axs = axs.flatten()
             for img, ax in zip([item for sublist in imgs_list for item in sublist], axs):
                 ax.imshow(img)
             plt.show()

In [45]: listnonsat = []
         for i in range(0,7):
             img =cv2.imread("./downloads/images - thumbnail/"+imgpd.image_filename[i])
             img = cv2.resize(img, (244,244), cv2.INTER_AREA)
             img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
             listnonsat.append(img)

         listsat = []
         for i in range(0,7):
             img =cv2.imread("./downloads/satellite - thumbnail/"+satpd.image_filename[i])
             img = cv2.resize(img, (244,244), cv2.INTER_AREA)
             img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
             listsat.append(img)

         draw_imgs([listnonsat, listsat])
```



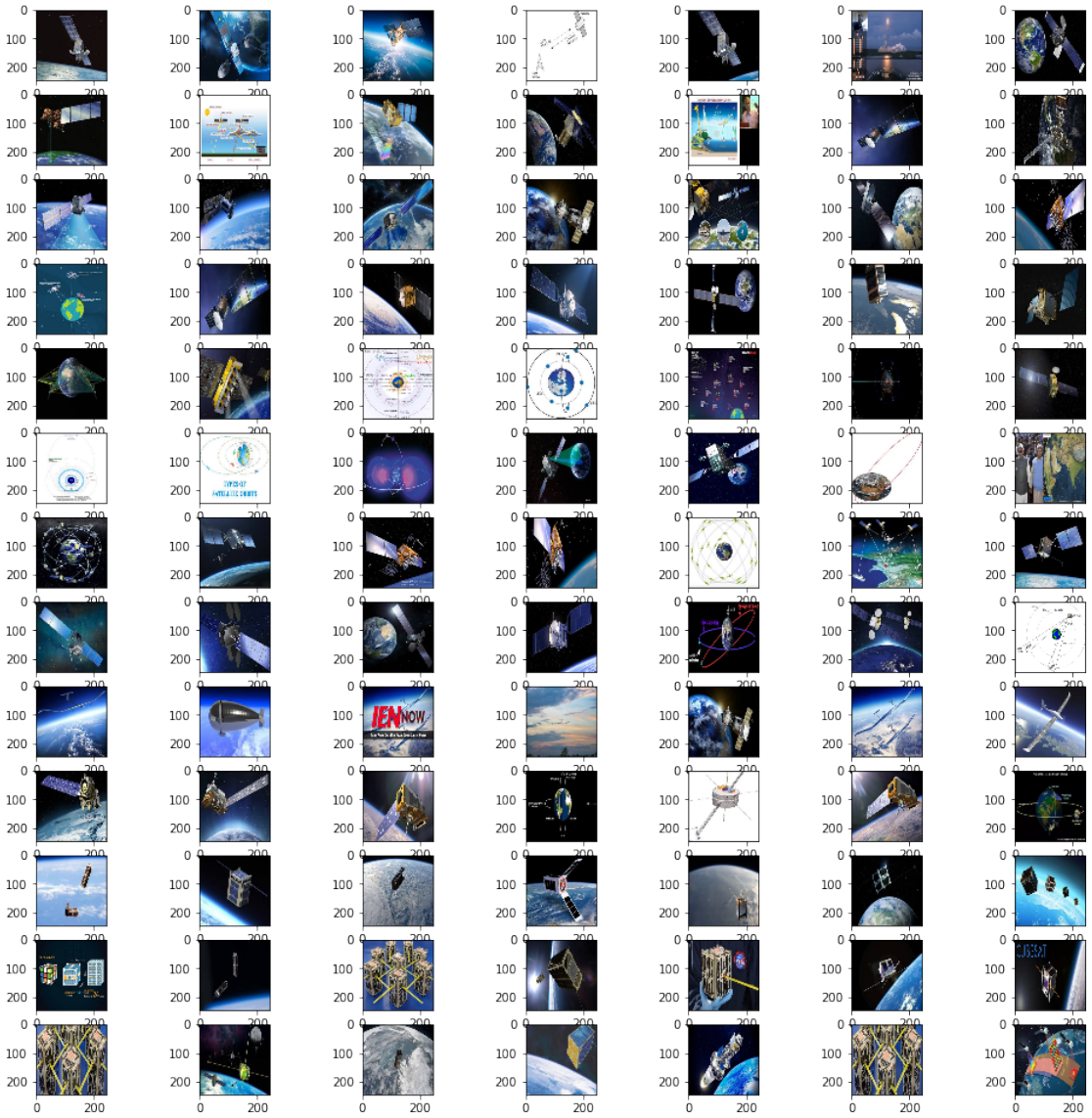


```
In [46]: listcat = []
         for current,cat in zip(catpd, categories):
             print(cat)
             listcurrent = []
             for i in range(0,7):
                 img =cv2.imread("./downloads/"+str(cat)+" - thumbnail/"+current.image_filename[
                 img = cv2.resize(img, (244,244), cv2.INTER_AREA)
                 img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
                 listcurrent.append(img)
             listcat.append(listcurrent)

         draw_imgs(listcat)
```

```
Communications Satellite
Remote Sensing Satellite
Navigation Satellite
LEO satellite
MEO satellite
HEO satellite
GPS satellite
GEO satellite
Drone Satellite
Polar Satellite
Nano Satellites
```

```
CubeSats
SmallSats
```



1.2 Classify satellite images

```
In [ ]: #saving images in the right directories for keras CNN

        train_size = 200
        test_size = 100

        try:
```

```
        os.mkdir("./classification")
        os.mkdir("./classification/test")
        os.mkdir("./classification/test/images")
        os.mkdir("./classification/test/satellite")
        os.mkdir("./classification/train")
        os.mkdir("./classification/train/images")
        os.mkdir("./classification/train/satellite")
        os.mkdir("./classification/valid")
        os.mkdir("./classification/valid/images")
        os.mkdir("./classification/valid/satellite")
    except:
        print("directories already in place")

    def path_from_number(n):
        if i<=train_size:
            path="train"
        else:
            if i<=train_size+test_size:
                path = "test"
            else:
                path = "valid"
        return path

    listnonsat = []
    for i in range(0,imgpd.shape[0]):
        try:
            img =cv2.imread("./downloads/images - thumbnail/"+imgpd.image_filename[i])
            img = cv2.resize(img, (64,64), cv2.INTER_AREA)
            img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
            cv2.imwrite("./classification/" + path_from_number(i)+"/images/"+str(i)+".jpg",
            listnonsat.append(img)
        except:
            print("image not resizable")

    listsat = []
    for i in range(0,satpd.shape[0]):
        try:
            img =cv2.imread("./downloads/satellite - thumbnail/"+satpd.image_filename[i])
            img = cv2.resize(img, (64,64), cv2.INTER_AREA)
            img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
            cv2.imwrite("./classification/" + path_from_number(i)+"/satellite/"+str(i)+".jpg
            listsat.append(img)
        except:
            print("image not resizable")
```

In [ ]:
```
len(listnonsat), len(listsat)
```

In [1]:
```
from keras.models import Sequential
```

```
        from keras.layers import Conv2D, MaxPooling2D
        from keras.layers import Activation, Flatten, Dense, Dropout
        from keras.preprocessing.image import ImageDataGenerator
        import time
        IMG_SIZE = 64 # Replace with the size of your images
        NB_CHANNELS = 3 # 3 for RGB images or 1 for grayscale images
        BATCH_SIZE = 32 # Typical values are 8, 16 or 32
        NB_TRAIN_IMG = 200 # Replace with the total number training images
        NB_VALID_IMG = 50 # Replace with the total number validation images
```

```
/home/franck/data/anaconda3/lib/python3.6/site-packages/h5py/__init__.py:34: FutureWarning: Conv
  from ._conv import register_converters as _register_converters
Using Theano backend.
```

```
In [2]: cnn = Sequential()
        cnn.add(Conv2D(filters=32,
                       kernel_size=(2,2),
                       strides=(1,1),
                       padding='same',
                       input_shape=(IMG_SIZE,IMG_SIZE,NB_CHANNELS),
                       data_format='channels_last'))
        cnn.add(Activation('relu'))
        cnn.add(MaxPooling2D(pool_size=(2,2),
                             strides=2))
        cnn.add(Conv2D(filters=64,
                       kernel_size=(2,2),
                       strides=(1,1),
                       padding='valid'))
        cnn.add(Activation('relu'))
        cnn.add(MaxPooling2D(pool_size=(2,2),
                             strides=2))
        cnn.add(Conv2D(filters=128,
                       kernel_size=(2,2),
                       strides=(1,1),
                       padding='valid'))
        cnn.add(Activation('relu'))
        cnn.add(MaxPooling2D(pool_size=(2,2),
                             strides=2))
        cnn.add(Flatten())
        cnn.add(Dense(32))
        cnn.add(Activation('relu'))
        cnn.add(Dropout(0.25))
        cnn.add(Dense(1))
        cnn.add(Activation('sigmoid'))
        cnn.compile(loss='binary_crossentropy', optimizer='rmsprop', metrics=['accuracy'])
```

```
In [3]: print(cnn.summary())
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 64, 64, 32)        416
_________________________________________________________________
activation_1 (Activation)    (None, 64, 64, 32)        0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 32, 32, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 31, 31, 64)        8256
_________________________________________________________________
activation_2 (Activation)    (None, 31, 31, 64)        0
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 15, 15, 64)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 14, 14, 128)       32896
_________________________________________________________________
activation_3 (Activation)    (None, 14, 14, 128)       0
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 7, 7, 128)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 6272)              0
_________________________________________________________________
dense_1 (Dense)              (None, 32)                200736
_________________________________________________________________
activation_4 (Activation)    (None, 32)                0
_________________________________________________________________
dropout_1 (Dropout)          (None, 32)                0
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 33
_________________________________________________________________
activation_5 (Activation)    (None, 1)                 0
=================================================================
Total params: 242,337
Trainable params: 242,337
Non-trainable params: 0
_________________________________________________________________
None
```

```
In [6]: train_datagen = ImageDataGenerator(
                rescale=1./255,
                shear_range=0.2,
                zoom_range=0.2,
                horizontal_flip=True)

        test_datagen = ImageDataGenerator(rescale=1./255)
```

```
        train_generator = train_datagen.flow_from_directory(
                'classification/train',
                target_size=(64, 64),
                batch_size=32,
                class_mode='binary')

        validation_generator = test_datagen.flow_from_directory(
                'classification/valid',
                target_size=(64, 64),
                batch_size=32,
                class_mode='binary')
```

```
Found 400 images belonging to 2 classes.
Found 193 images belonging to 2 classes.
```

```
In [ ]: cnn.fit_generator(
                train_generator,
                steps_per_epoch=100,
                epochs=20,
                validation_data=validation_generator,
                validation_steps=80)
```

```
In [ ]: cnn.save_weights('cnn.h5')
```

```
In [7]: cnn.load_weights('cnn.h5')
```

```
In [8]: test_generator = test_datagen.flow_from_directory(
                'classification/test',
                target_size=(64, 64),
                batch_size=32,
                class_mode='binary')
        test_imgs, test_labels = next(test_generator)
```

```
Found 200 images belonging to 2 classes.
```

```
In [30]: test_imgs.shape
```

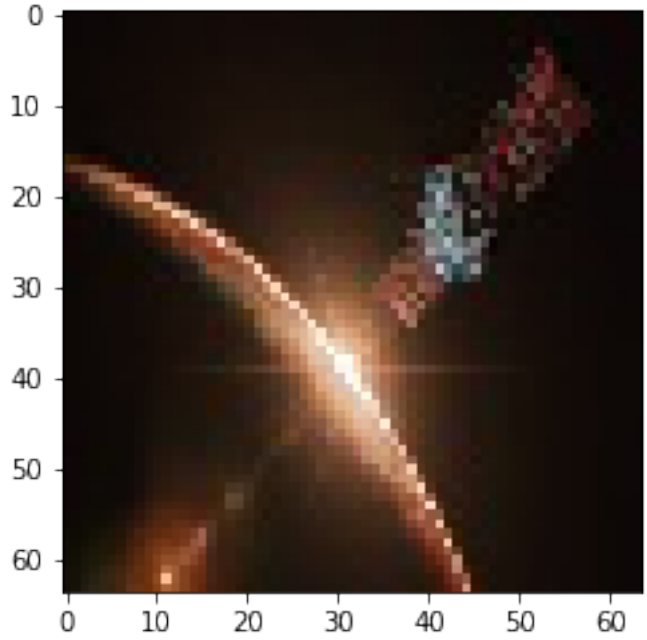
```
Out[30]: (32, 64, 64, 3)
```

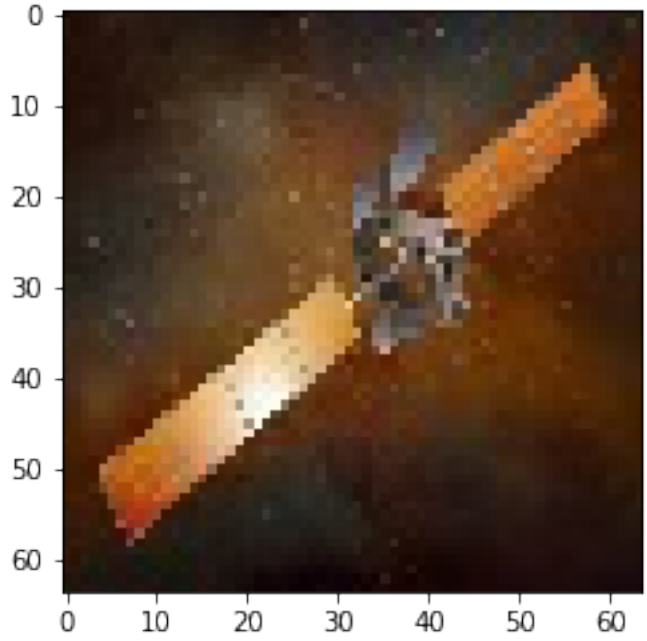
```
In [20]: predictions = cnn.predict_generator(test_generator, steps=1)
```

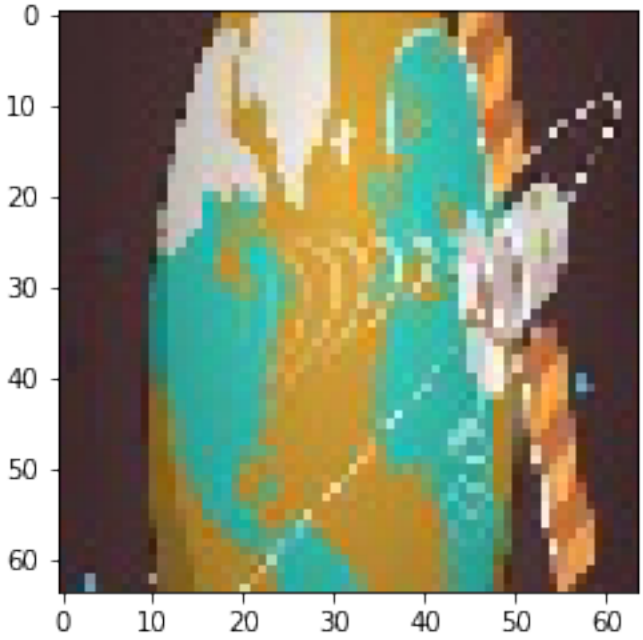
```
In [31]: predictions.shape
```

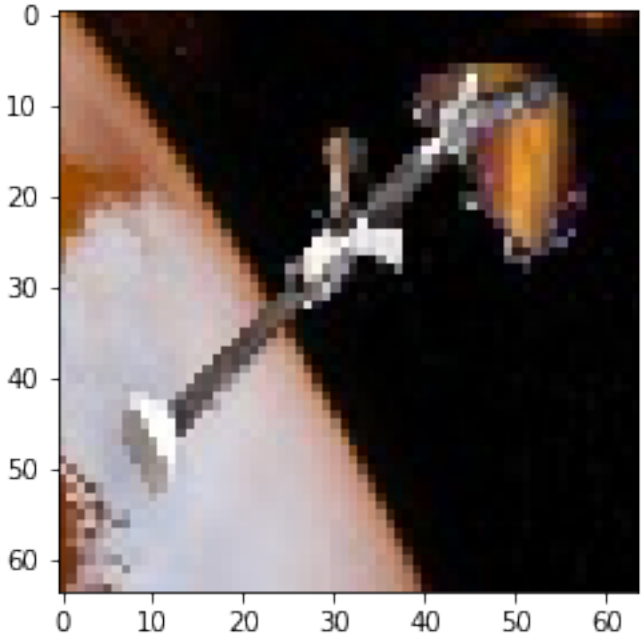
```
Out[31]: (32, 1)
```

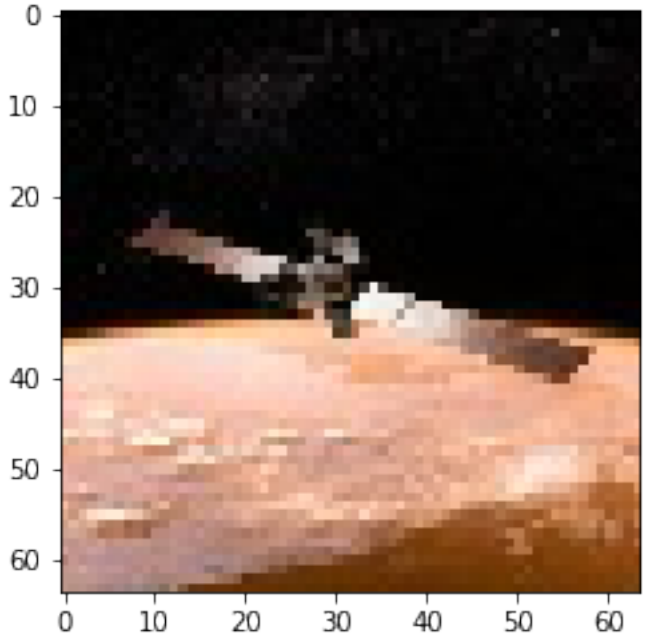
```
In [33]: # check satellites
         for i in range(0,predictions.shape[0]):
             if predictions[i]>0.5:
                 plt.figure()
                 plt.imshow(test_imgs[i])
```

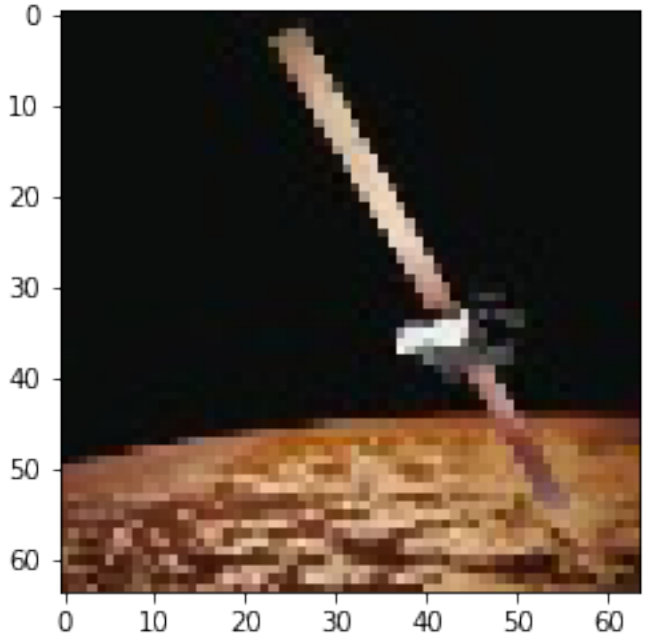

0
10
20
30
40
50
60
0
10
20
30
40
50
60

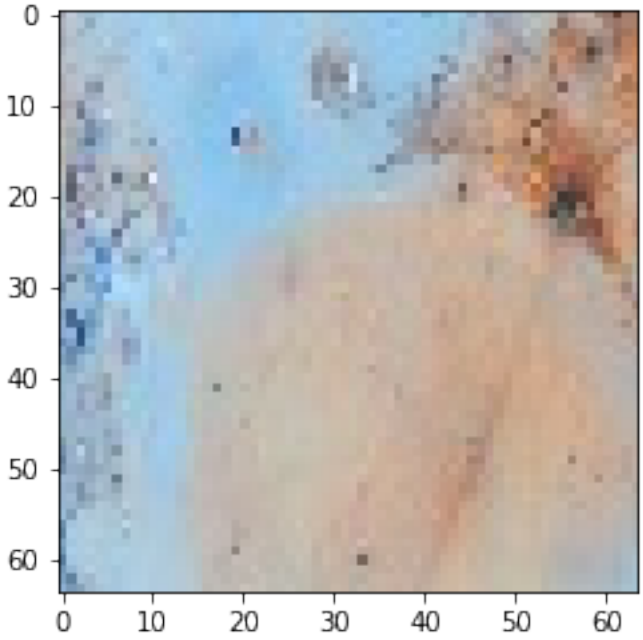

0
10
20
30
40
50
60
0
10
20
30
40
50
60

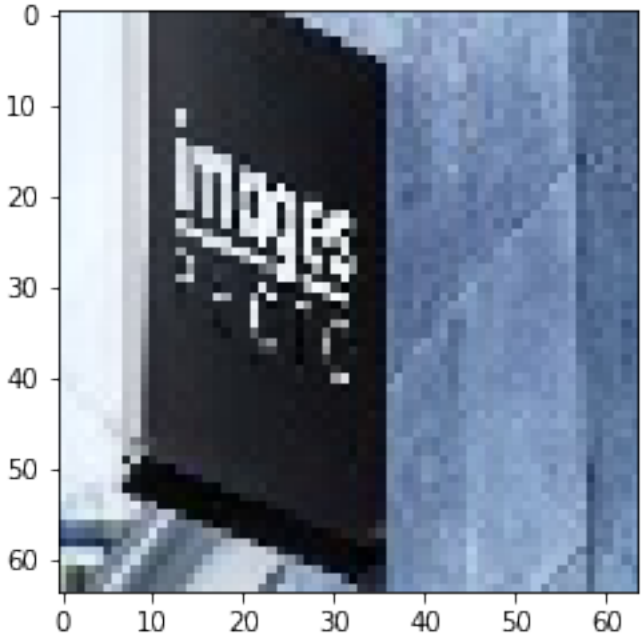

0
10
20
30
40
50
60
0
10
20
30
40
50
60

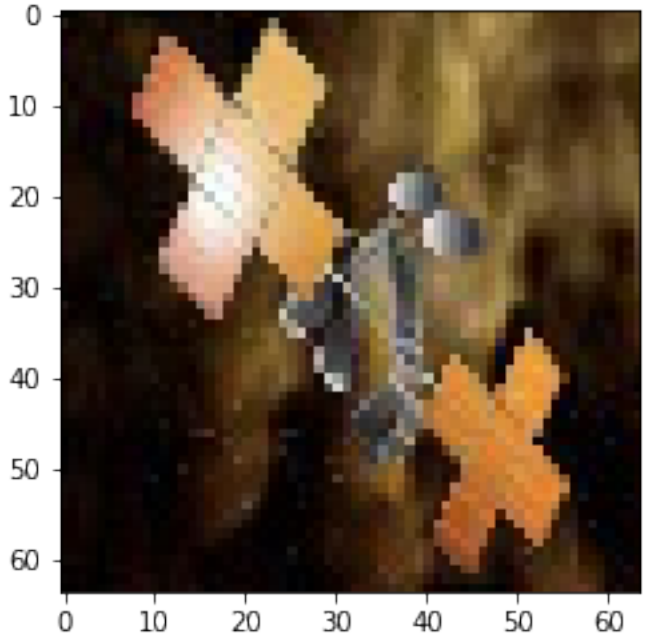

0
10
20
30
40
50
60
0
10
20
30
40
50
60

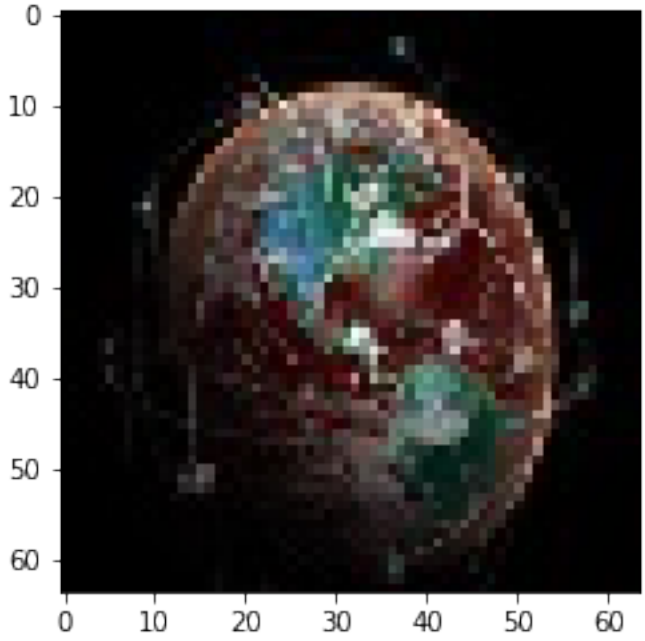

0
10
20
30
40
50
60
0
10
20
30
40
50
60

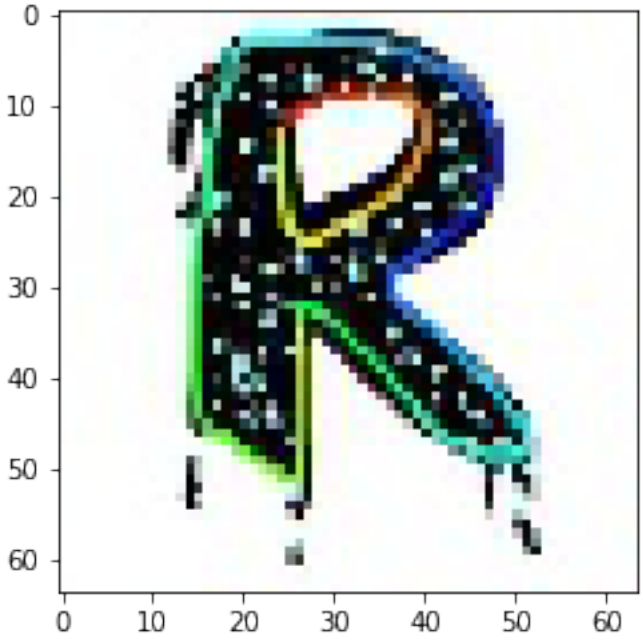

0
10
20
30
40
50
60
0
10
20
30
40
50
60

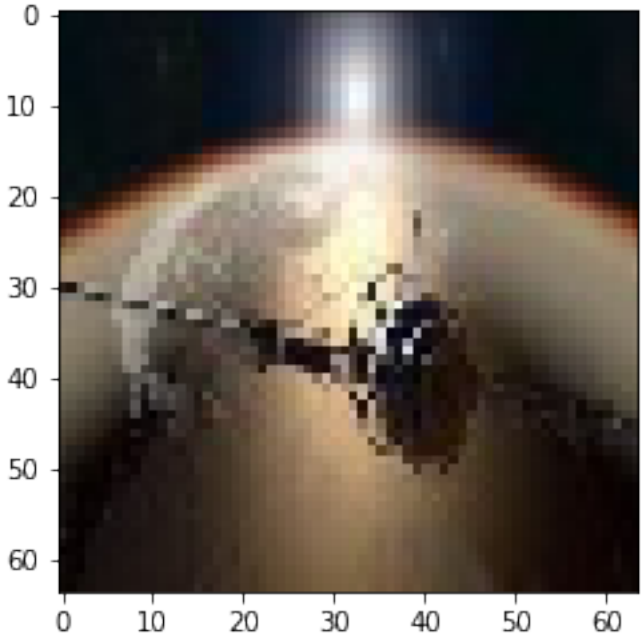

0
10
20
30
40
50
60
0
10
20
30
40
50
60

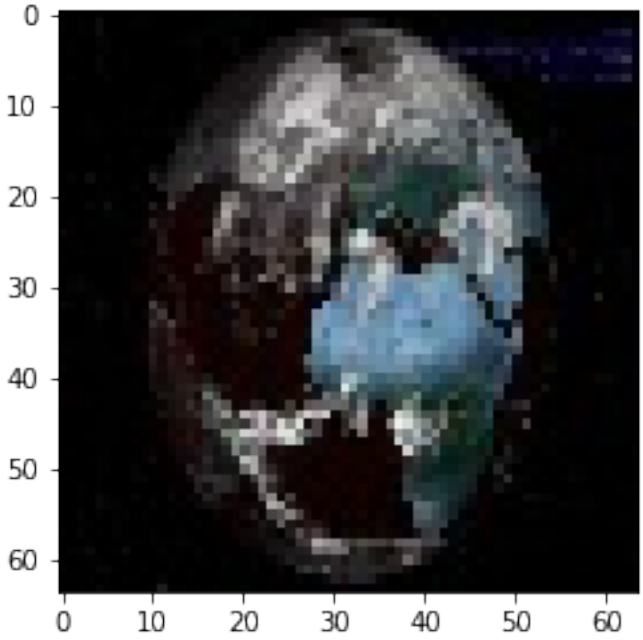

0
10
20
30
40
50
60
0
10
20
30
40
50
60

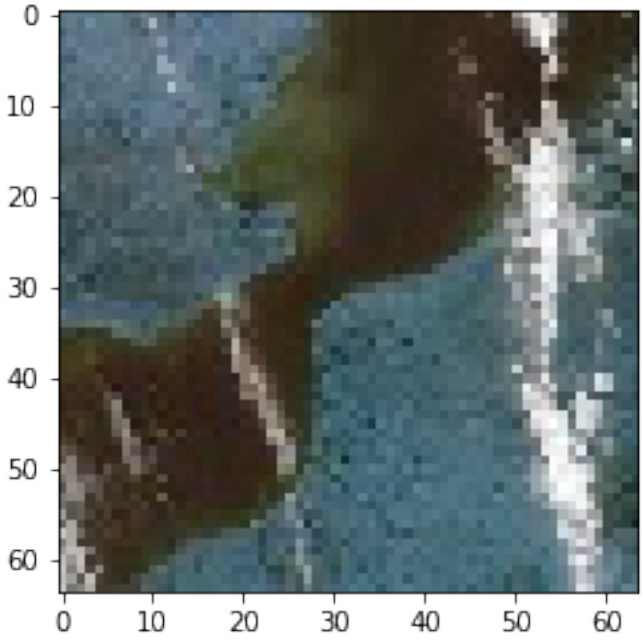

0
10
20
30
40
50
60
0
10
20
30
40
50
60

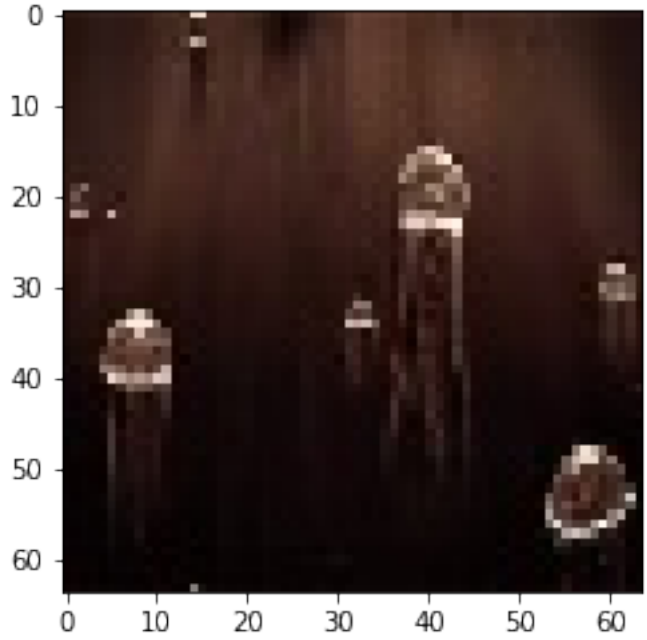

0
10
20
30
40
50
60
0
10
20
30
40
50
60

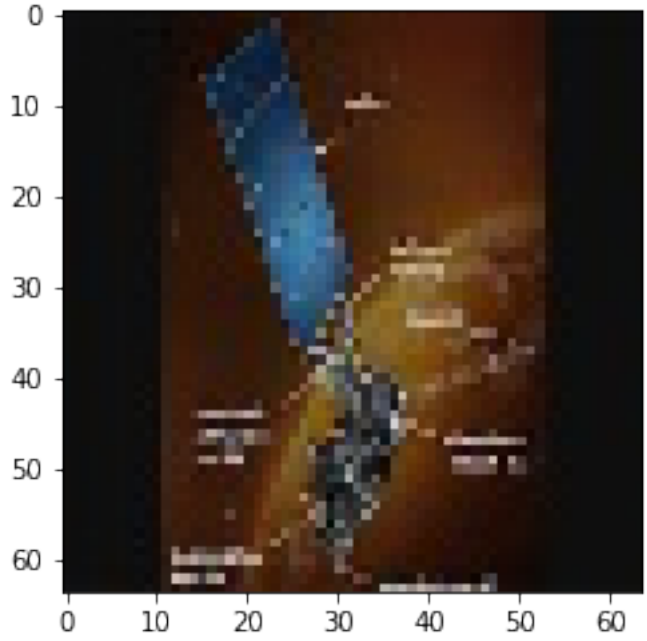

0
10
20
30
40
50
60
0
10
20
30
40
50
60

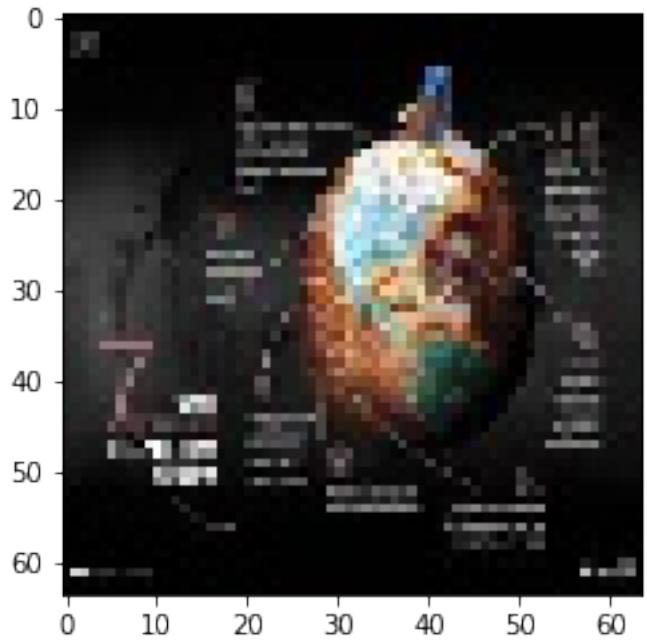

0
10
20
30
40
50
60
0
10
20
30
40
50
60


0
10
20
30
40
50
60
0
10
20
30
40
50
60

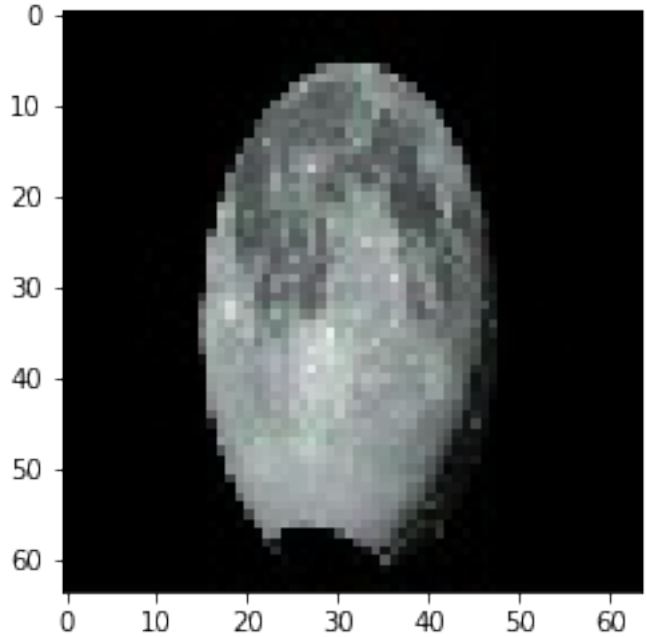

0
10
20
30
40
50
60
0
10
20
30
40
50
60

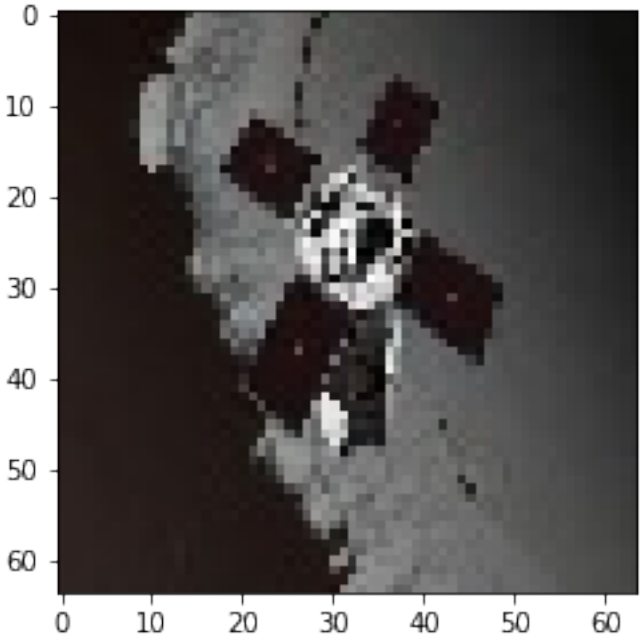

0
10
20
30
40
50
60
0
10
20
30
40
50
60

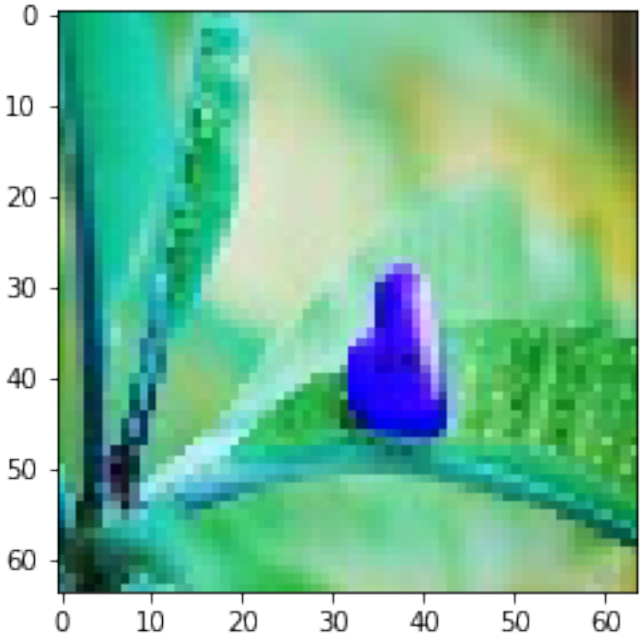

0
10
20
30
40
50
60
0
10
20
30
40
50
60

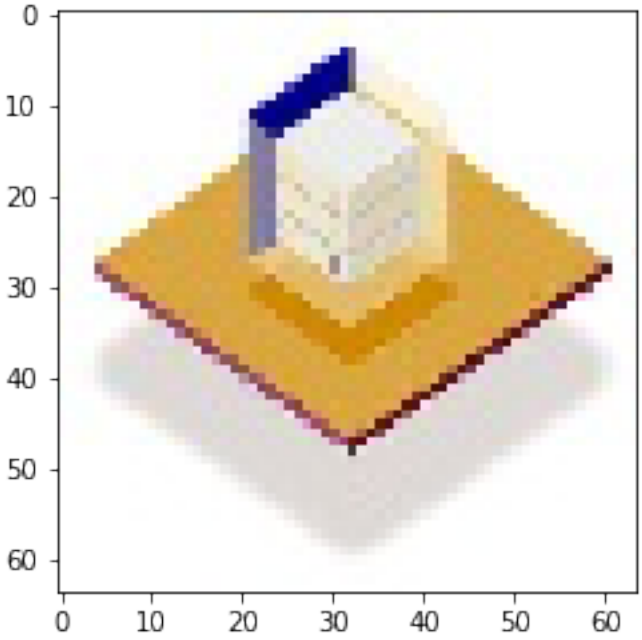


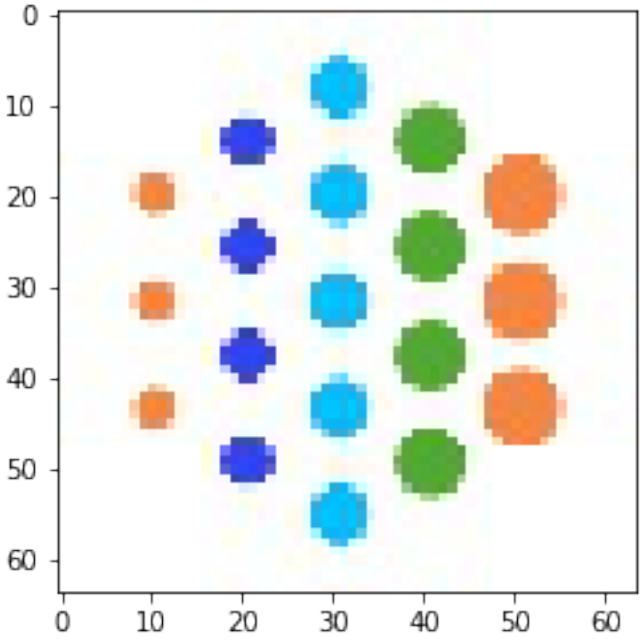
In [34]:
```
# check non satellites
for i in range(0,predictions.shape[0]):
    if predictions[i]<0.5:
        plt.figure()
        plt.imshow(test_imgs[i])
```

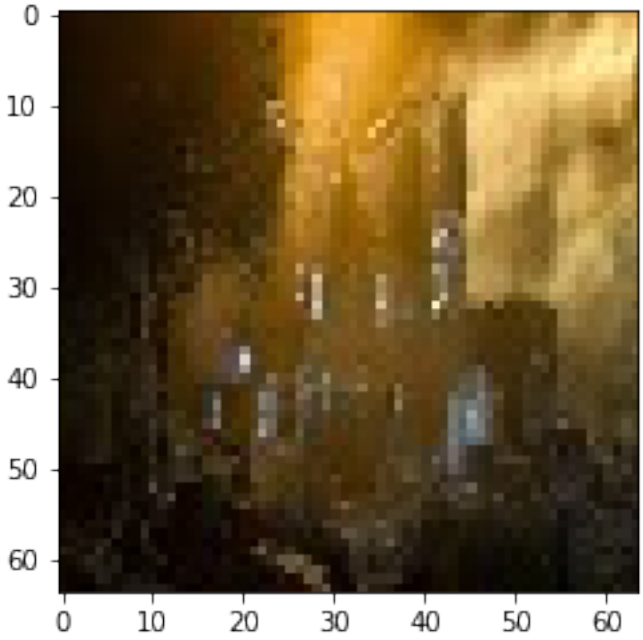


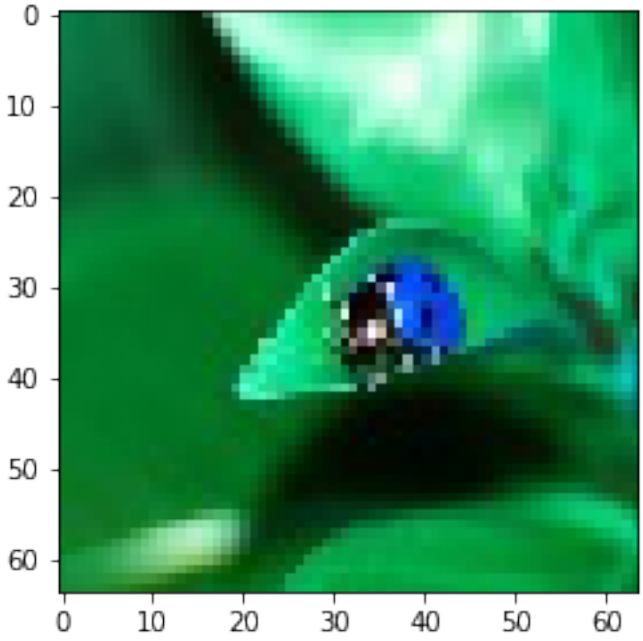

0
10
20
30
40
50
60
0
10
20
30
40
50
60

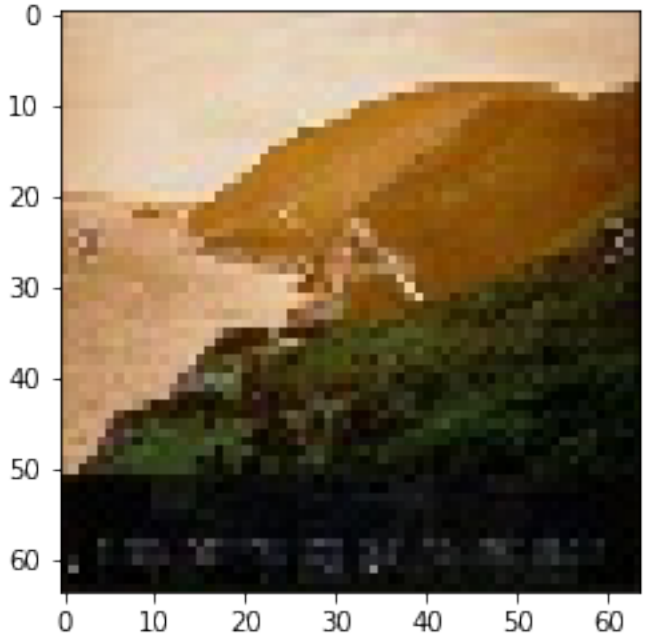

0
10
20
30
40
50
60
0
10
20
30
40
50
60

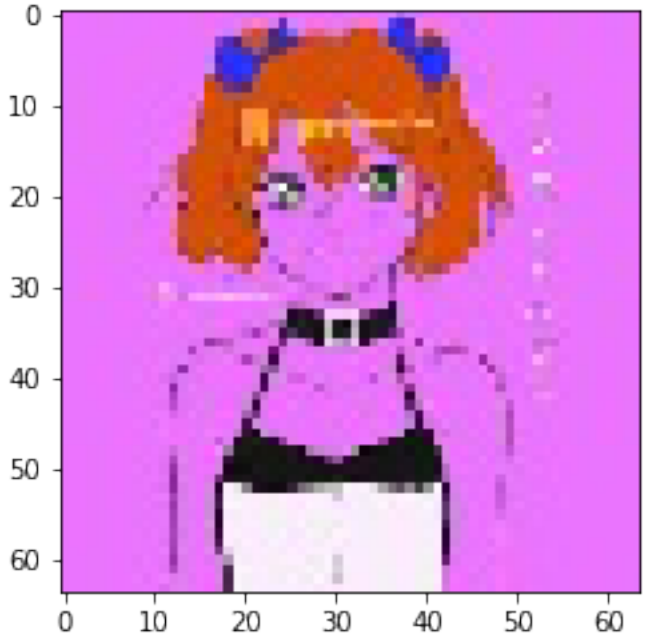

0
10
20
30
40
50
60
0
10
20
30
40
50
60

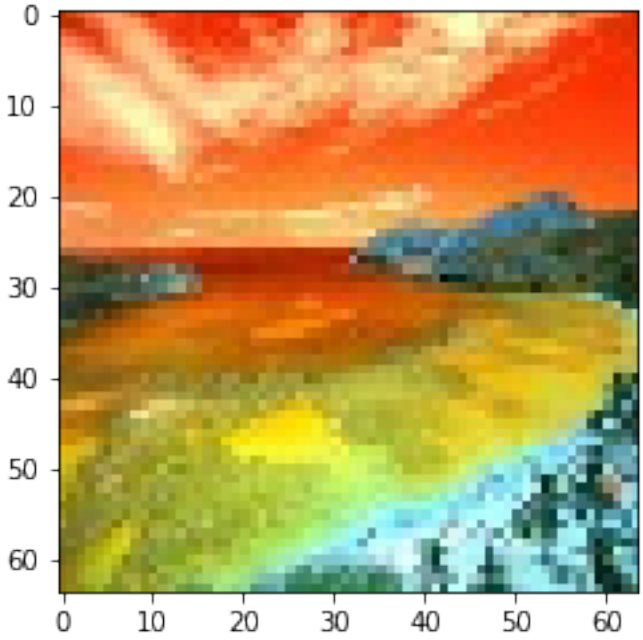

0
10
20
30
40
50
60
0
10
20
30
40
50
60

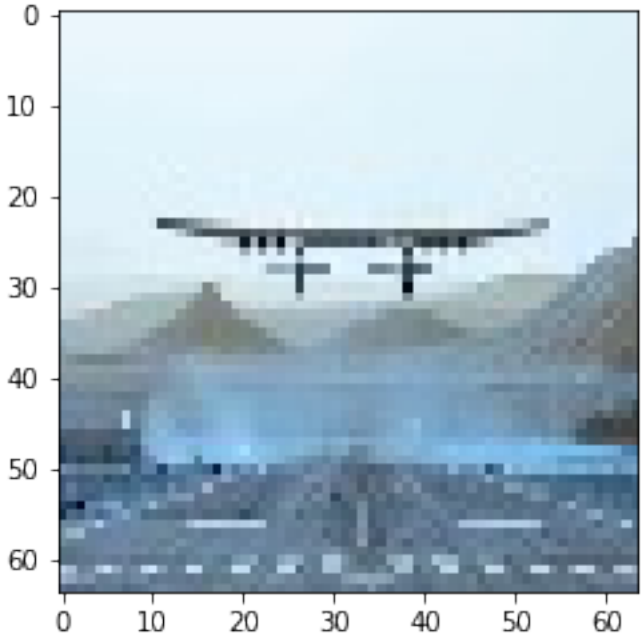

0
10
20
30
40
50
60
0
10
20
30
40
50
60

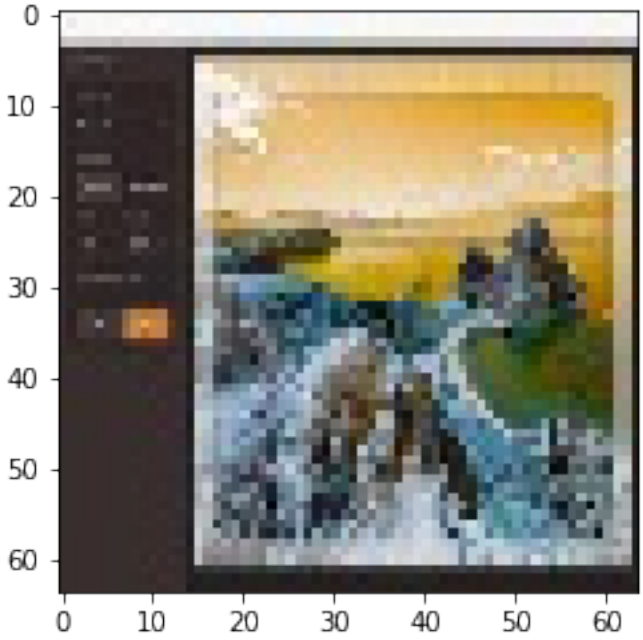

0
10
20
30
40
50
60
0
10
20
30
40
50
60

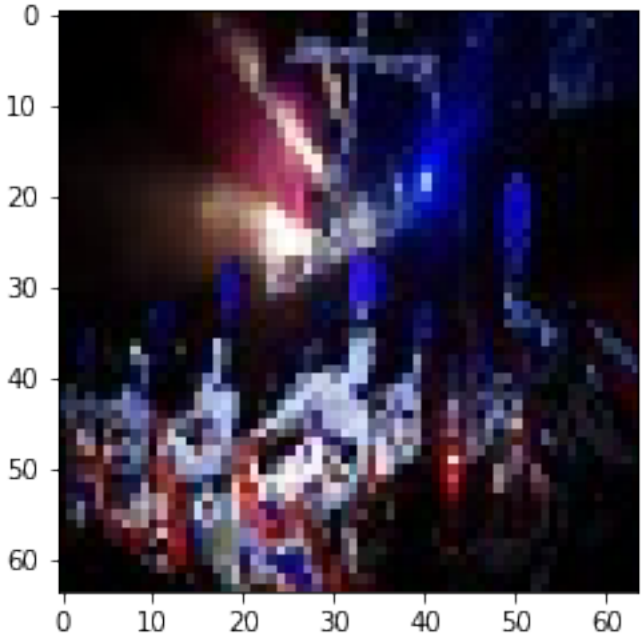

0
10
20
30
40
50
60
0
10
20
30
40
50
60

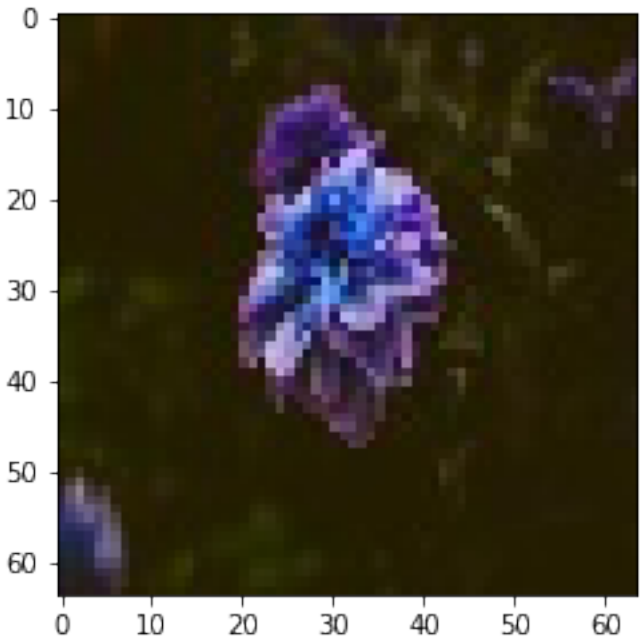

0
10
20
30
40
50
60
0
10
20
30
40
50
60

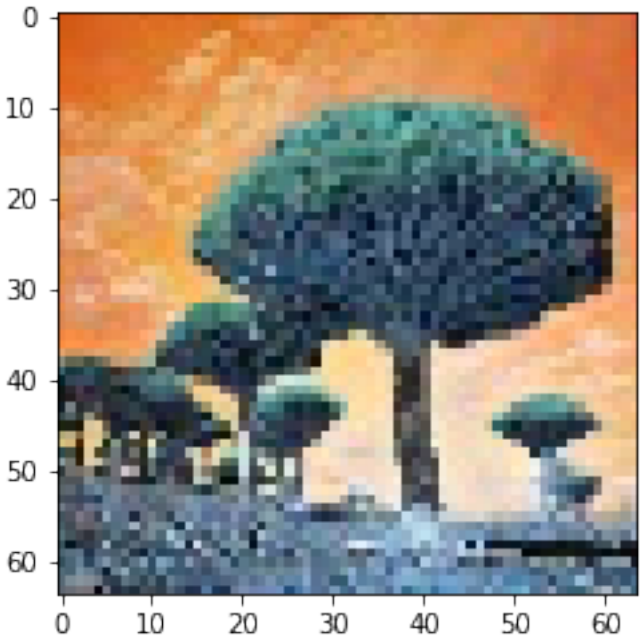

0
10
20
30
40
50
60
0
10
20
30
40
50
60


0
10
20
30
40
50
60
0
10
20
30
40
50
60


0
10
20
30
40
50
60
0
10
20
30
40
50
60


0
10
20
30
40
50
60
0
10
20
30
40
50
60





1.3 filtering specific satellites datasets

```
In [53]: try:
             os.mkdir("./specific")
```

```
            os.mkdir("./specific/Remote Sensing Satellite")
        except:
            print("directory exists")

        listspecific = []
        for i in range(0,catpd[1].shape[0]):
            try:
                img =cv2.imread("./downloads/Remote Sensing Satellite - thumbnail/"+catpd[1].im
                img = cv2.resize(img, (64,64), cv2.INTER_AREA)
                img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
                cv2.imwrite("./specific/Remote Sensing Satellite/"+str(i)+".jpg", img)
                listspecific.append(img)
            except:
                print("image not resizable")

        draw_imgs([listspecific])
```

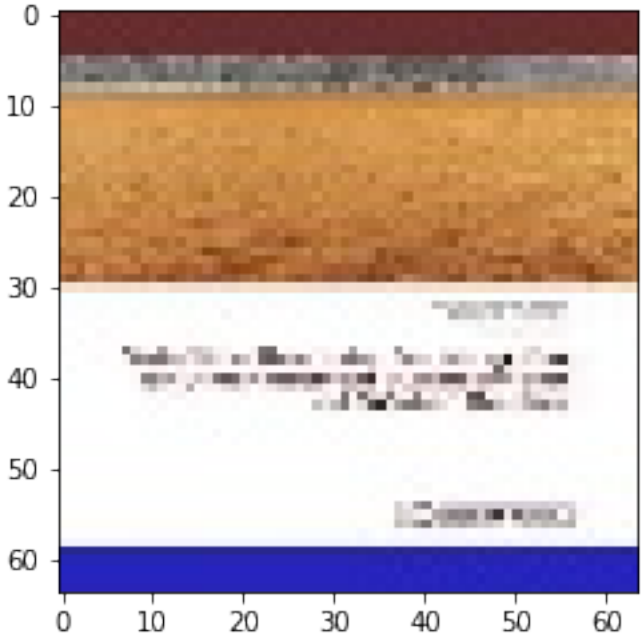
```
image not resizable
```

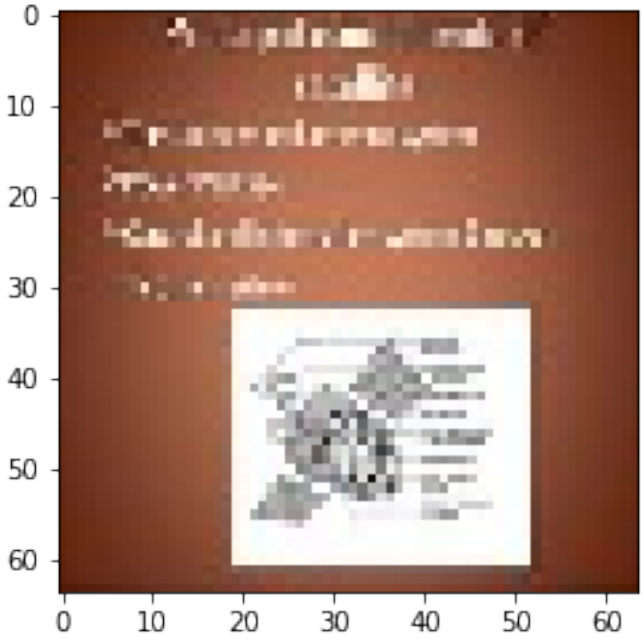
```
In [54]: specific_generator = test_datagen.flow_from_directory(
                'specific',
                target_size=(64, 64),
                batch_size=32,
                class_mode='binary')
         specific_imgs, specific_labels = next(specific_generator)
```

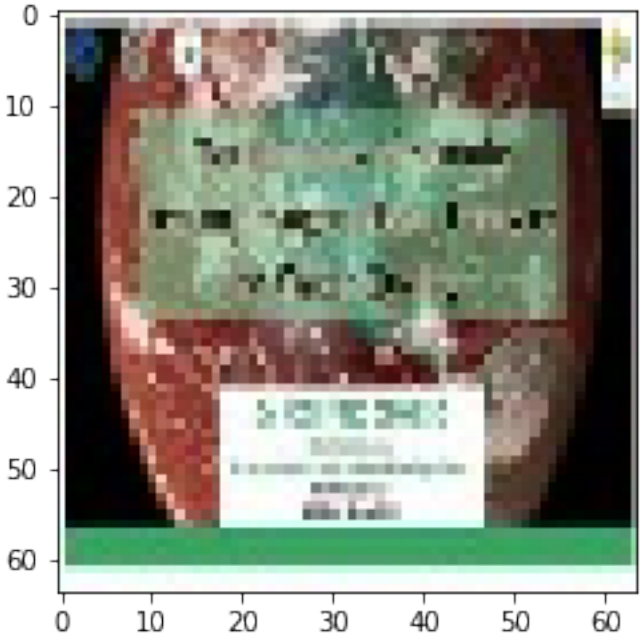
```
Found 692 images belonging to 1 classes.
```

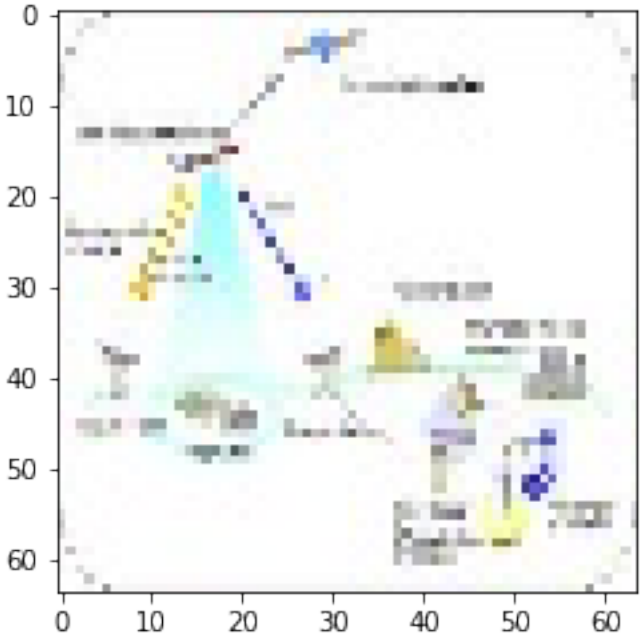
```
In [55]: specific_predictions = cnn.predict_generator(specific_generator, steps=1)
```

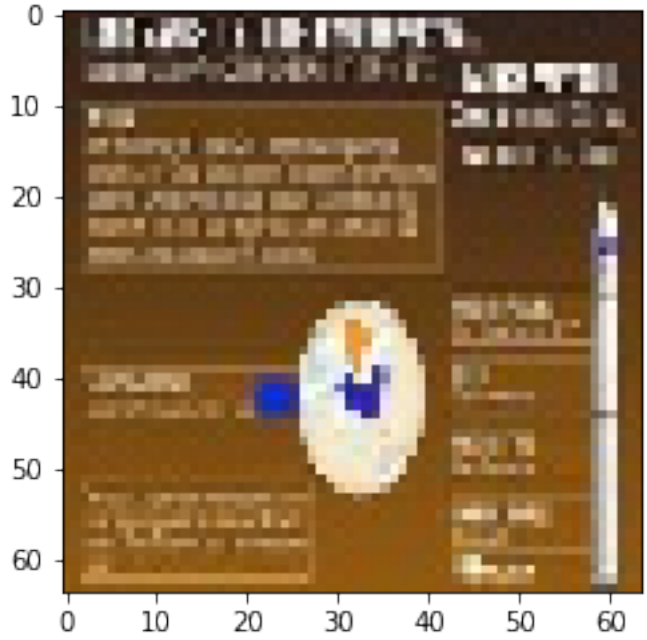
```
In [56]: # check satellites
         for i in range(0,specific_predictions.shape[0]):
             if specific_predictions[i]>0.5:
                 plt.figure()
                 plt.imshow(specific_imgs[i])
```

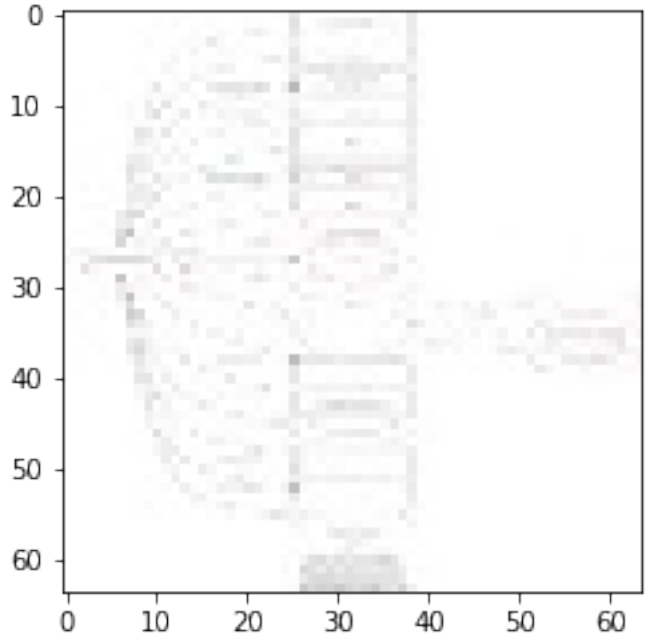

0
10
20
30
40
50
60
0
10
20
30
40
50
60

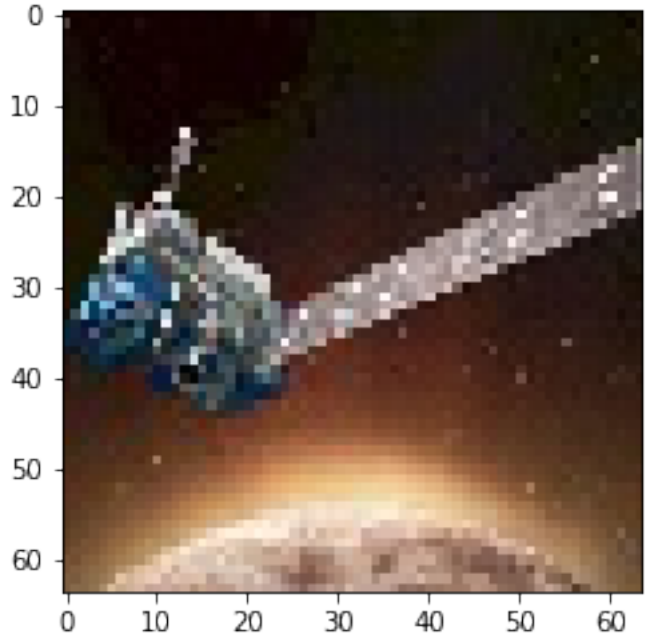

0
10
20
30
40
50
60
0
10
20
30
40
50
60

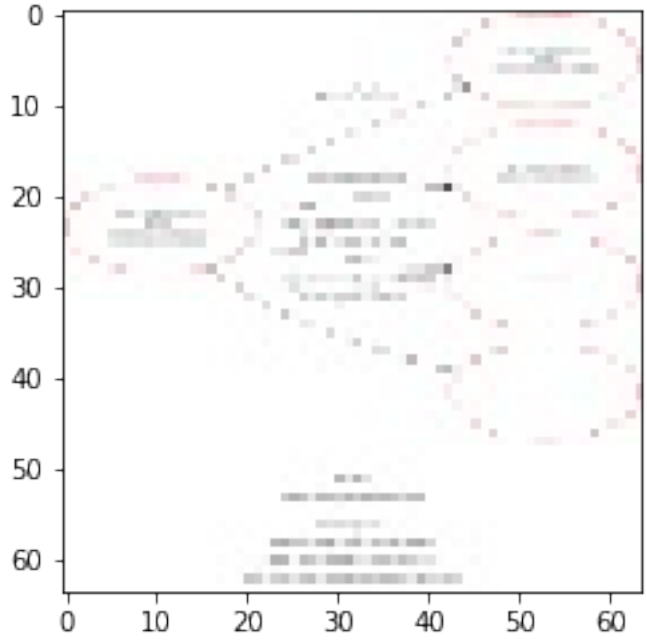

0
10
20
30
40
50
60
0
10
20
30
40
50
60

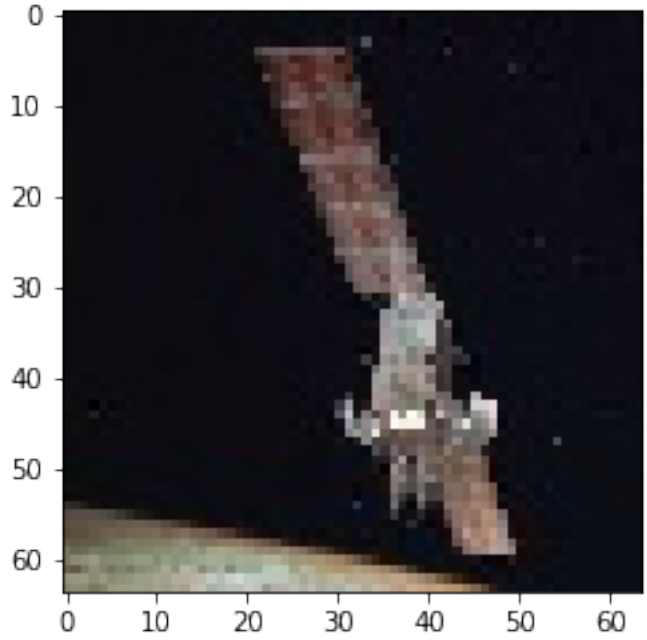

0
10
20
30
40
50
60
0
10
20
30
40
50
60

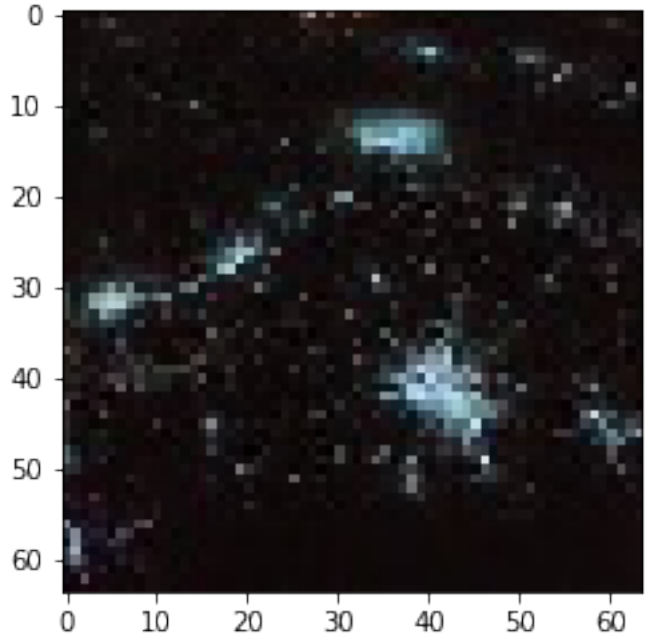

0
10
20
30
40
50
60
0
10
20
30
40
50
60

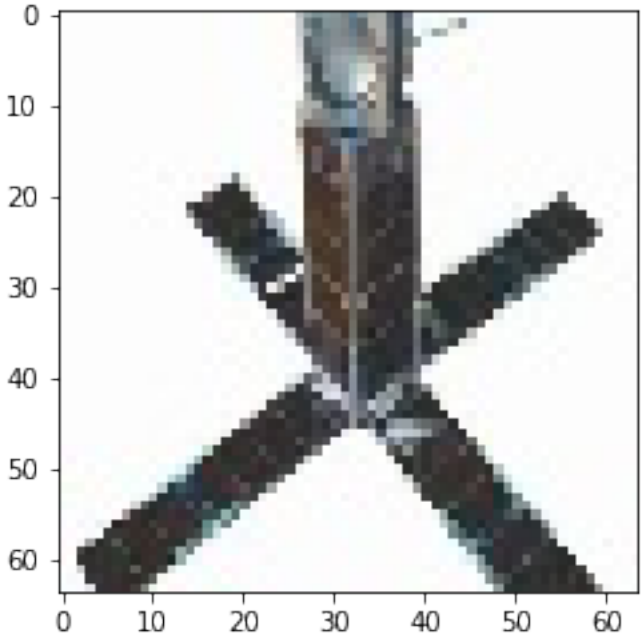

0
10
20
30
40
50
60
0
10
20
30
40
50
60

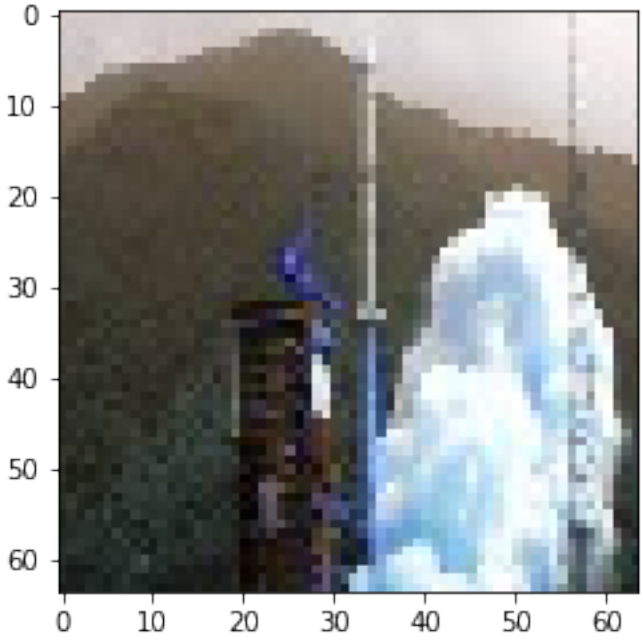

0
10
20
30
40
50
60
0
10
20
30
40
50
60

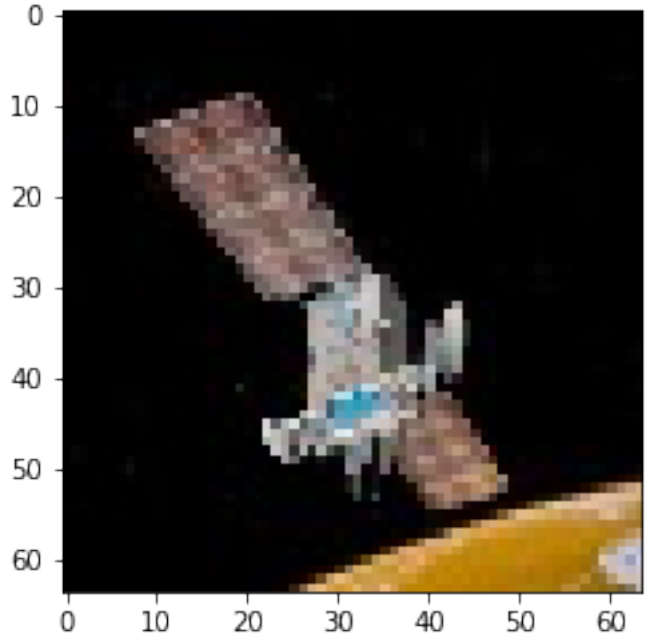

0
10
20
30
40
50
60
0
10
20
30
40
50
60

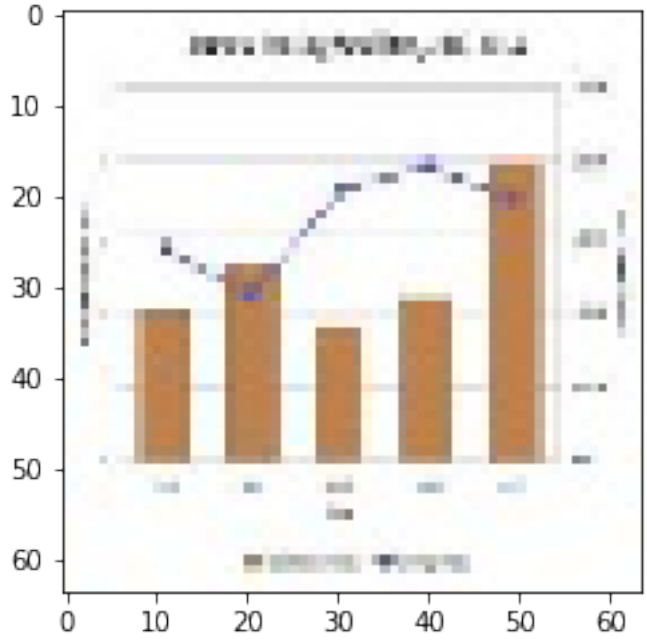

0
10
20
30
40
50
60
0
10
20
30
40
50
60

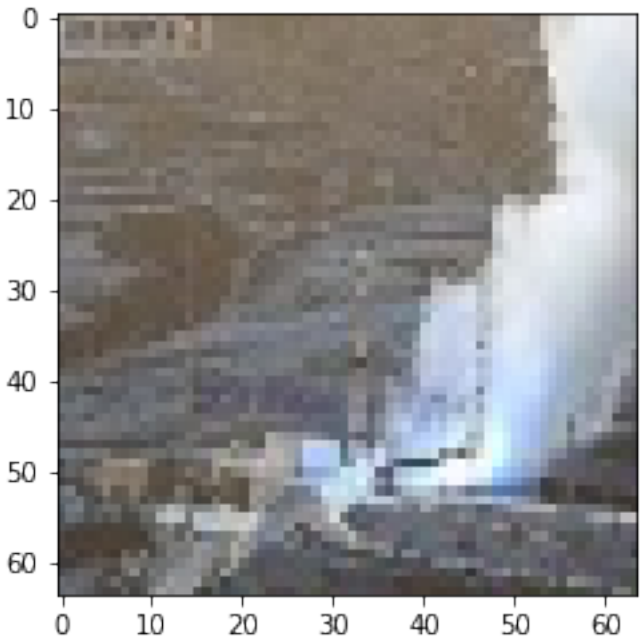

0
10
20
30
40
50
60
0
10
20
30
40
50
60


0
10
20
30
40
50
60
0
10
20
30
40
50
60

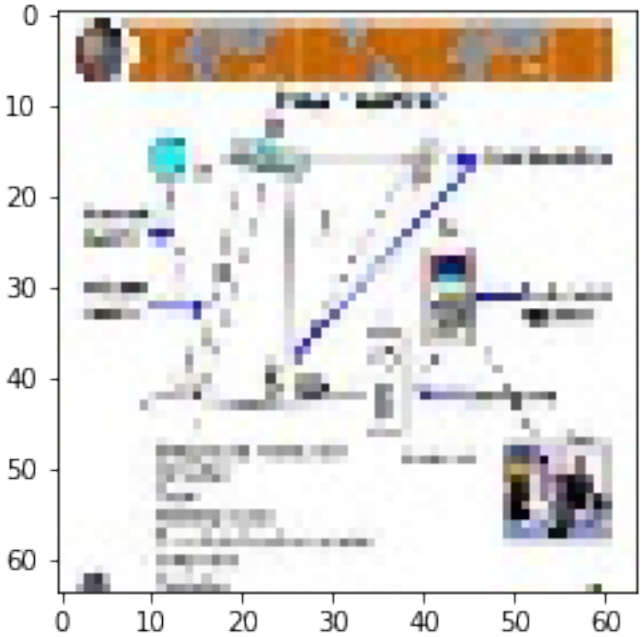

0
10
20
30
40
50
60
0
10
20
30
40
50
60

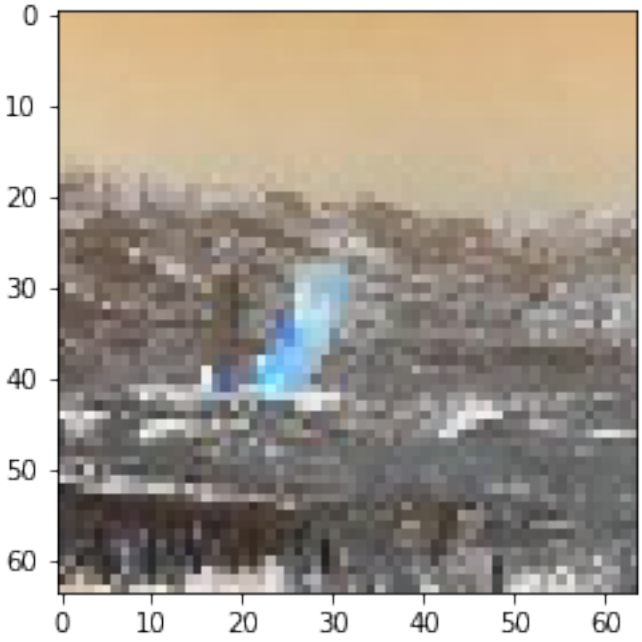

0
10
20
30
40
50
60
0
10
20
30
40
50
60

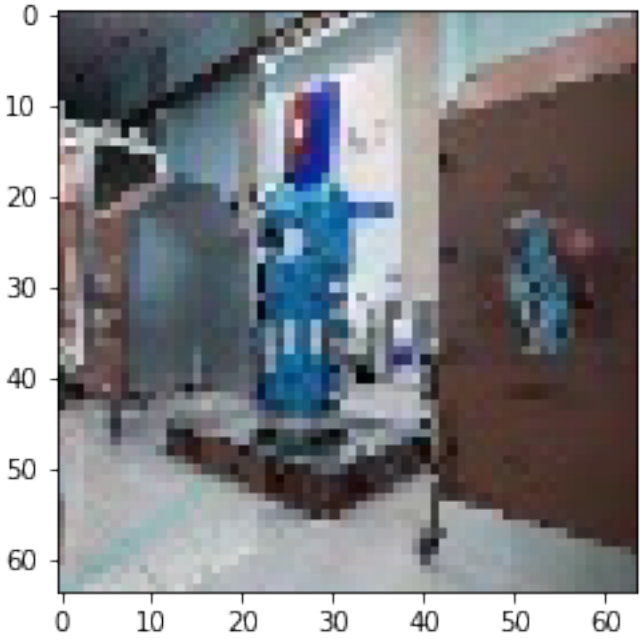

0
10
20
30
40
50
60
0
10
20
30
40
50
60

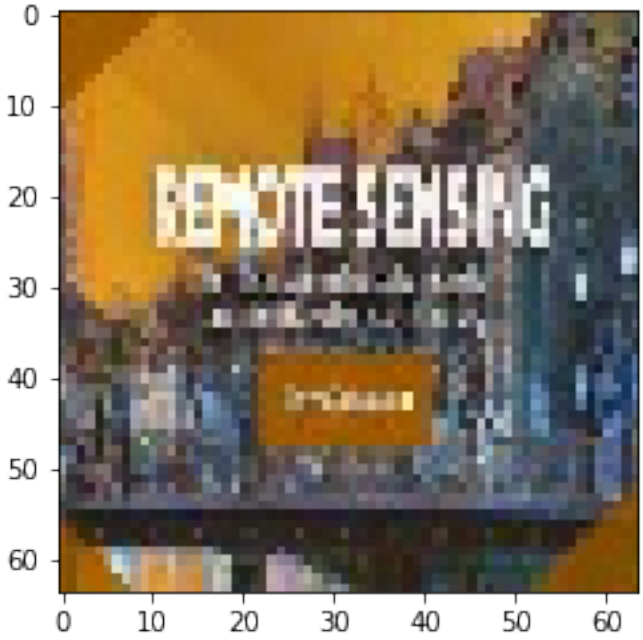


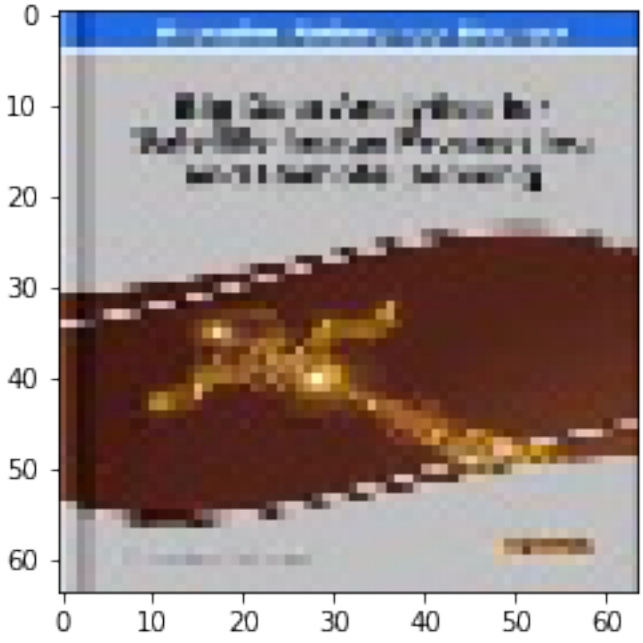
In [57]:
```
# check non satellites
for i in range(0,specific_predictions.shape[0]):
    if specific_predictions[i]<0.5:
        plt.figure()
        plt.imshow(specific_imgs[i])
```

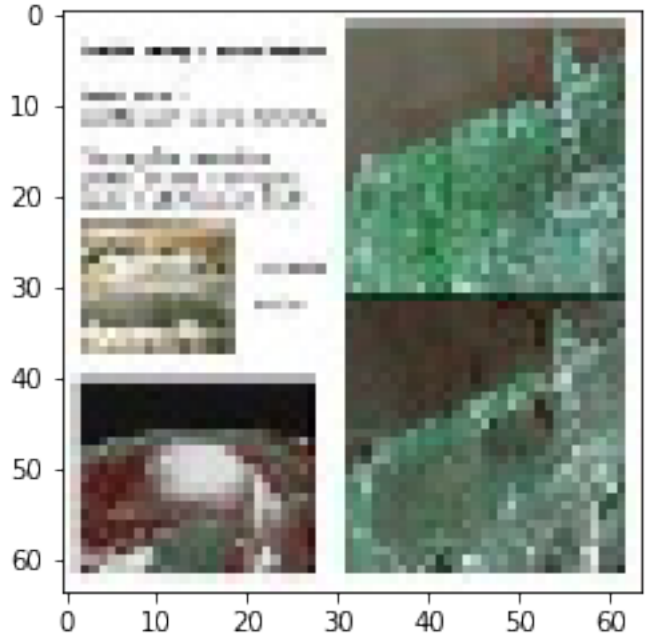


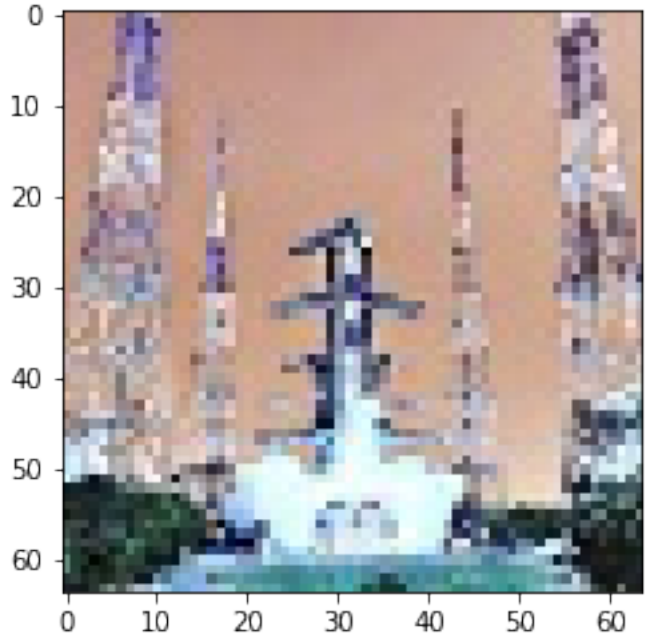

0
10
20
30
40
50
60
0
10
20
30
40
50
60

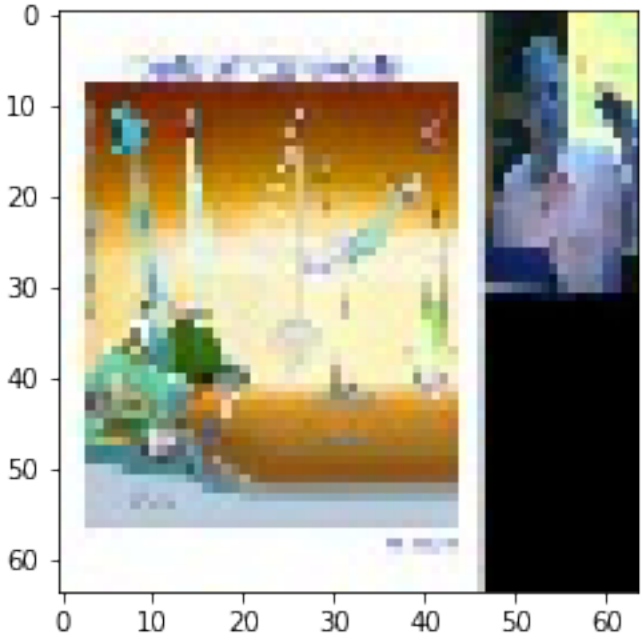

0
10
20
30
40
50
60
0
10
20
30
40
50
60

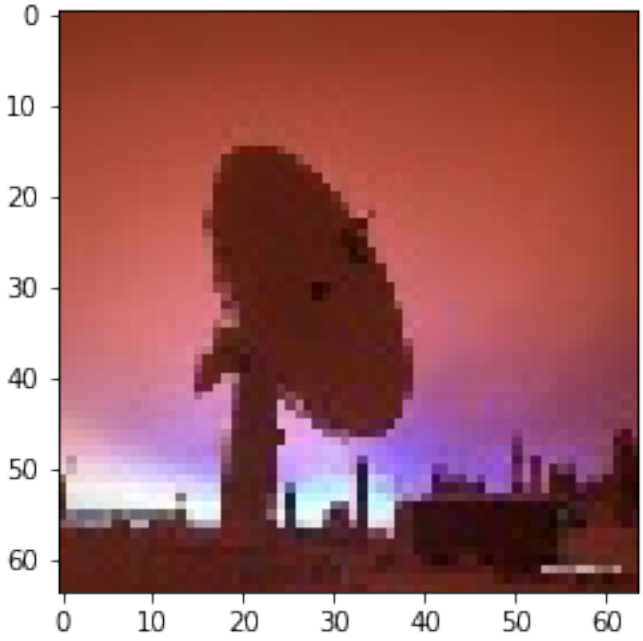

0
10
20
30
40
50
60
0
10
20
30
40
50
60

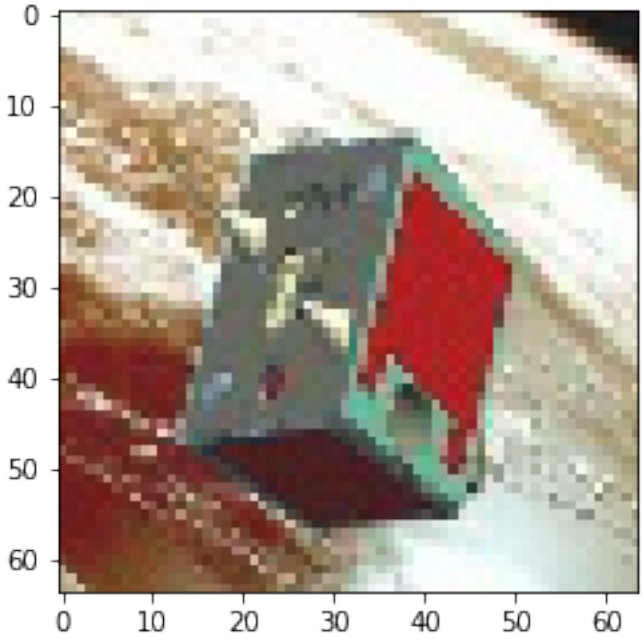

0
10
20
30
40
50
60
0
10
20
30
40
50
60

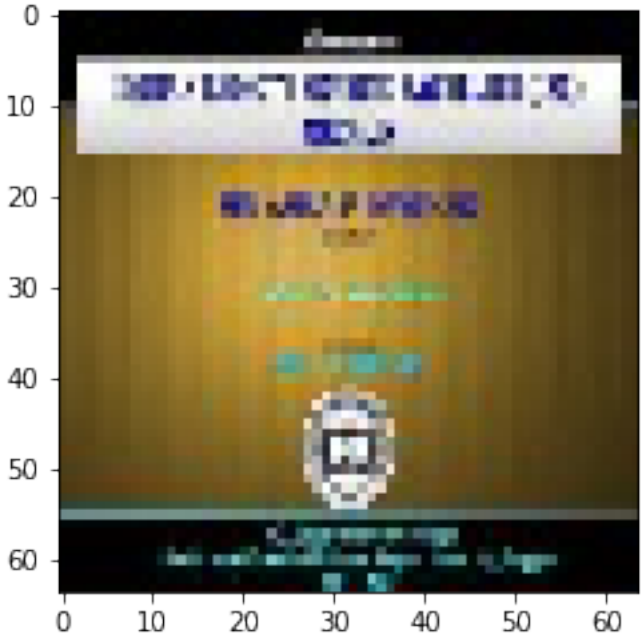

0
10
20
30
40
50
60
0
10
20
30
40
50
60

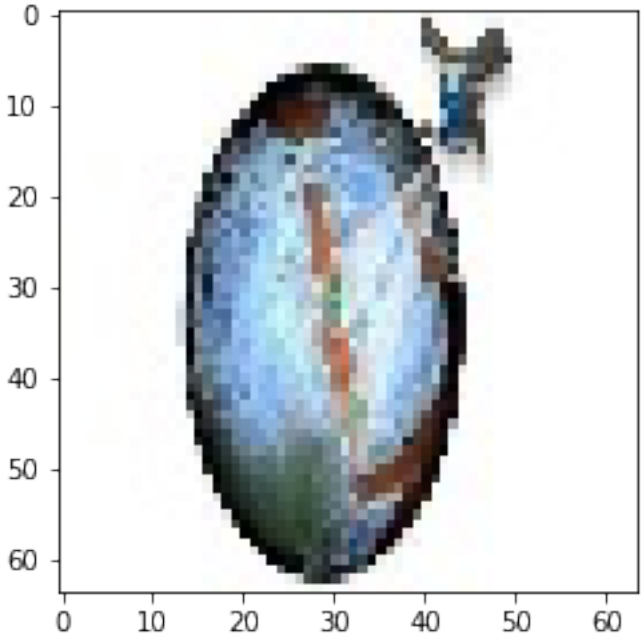

0
10
20
30
40
50
60
0
10
20
30
40
50
60

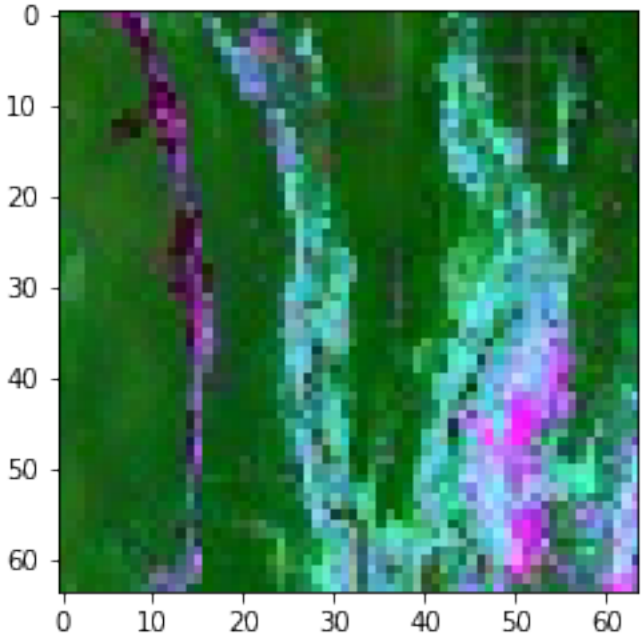

0
10
20
30
40
50
60
0
10
20
30
40
50
60

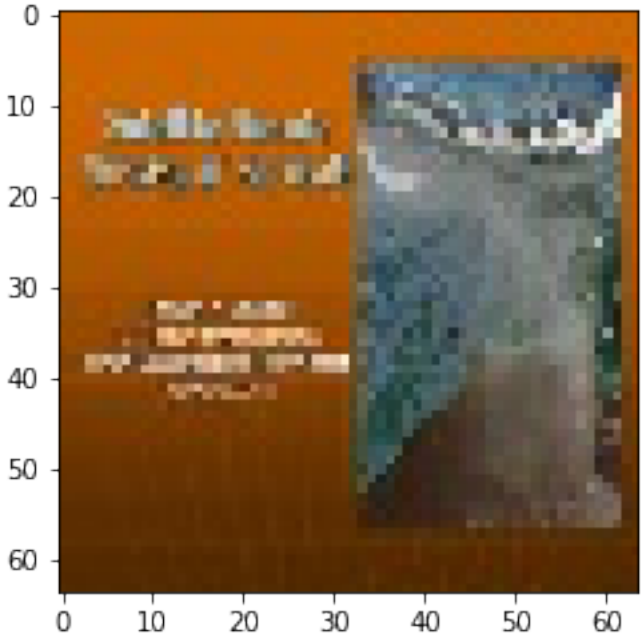

0
10
20
30
40
50
60
0
10
20
30
40
50
60

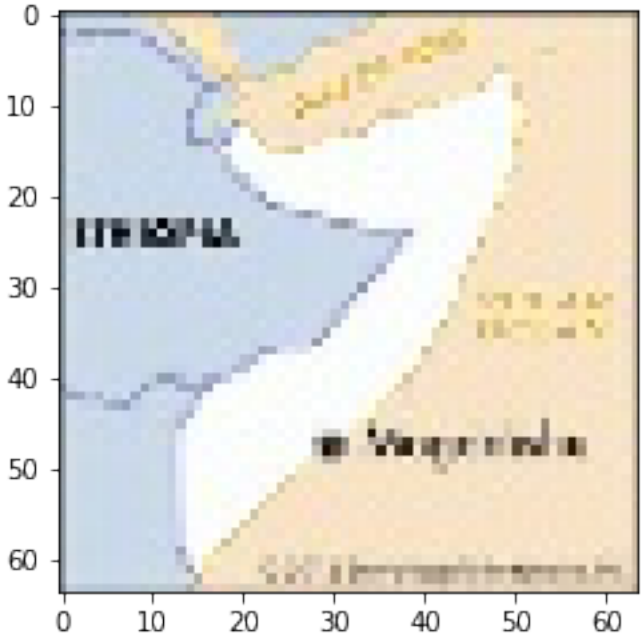

0
10
20
30
40
50
60
0
10
20
30
40
50
60

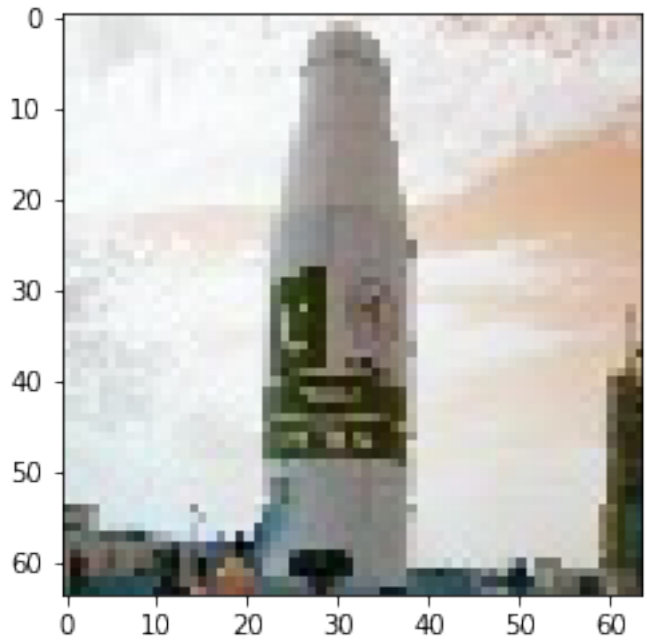

0
10
20
30
40
50
60
0
10
20
30
40
50
60

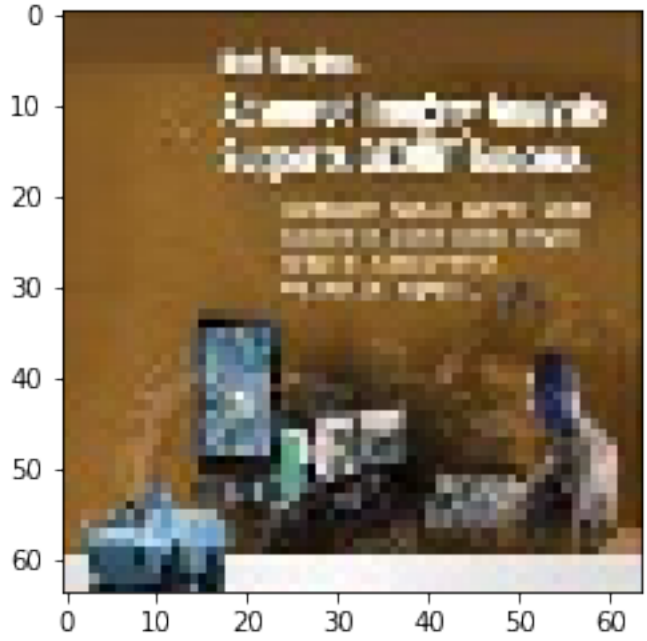

0
10
20
30
40
50
60
0
10
20
30
40
50
60


0
10
20
30
40
50
60
0
10
20
30
40
50
60


0
10
20
30
40
50
60
0
10
20
30
40
50
60

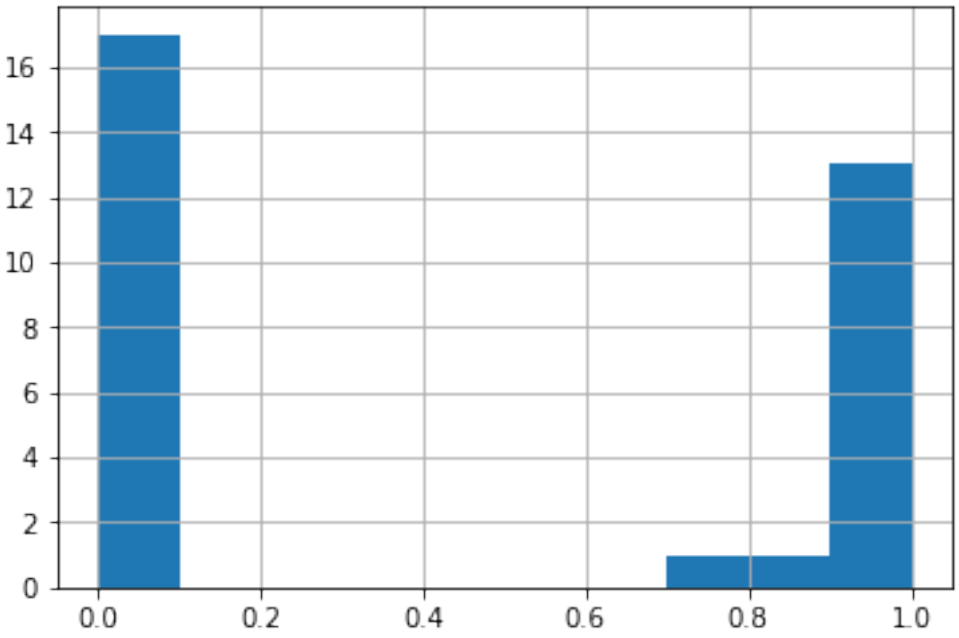

0
10
20
30
40
50
60
0
10
20
30
40
50
60






```
In [72]: pd.Series(specific_predictions.transpose()[0]).hist()
```

```
Out[72]: <matplotlib.axes._subplots.AxesSubplot at 0x7f78746ff898>
```


16
14
12
10
8
6
4
2
0
0.0
0.2
0.4
0.6
0.8
1.0