Morpheus Model Tutorial: Deep Learning Galaxy Classifications for Roman

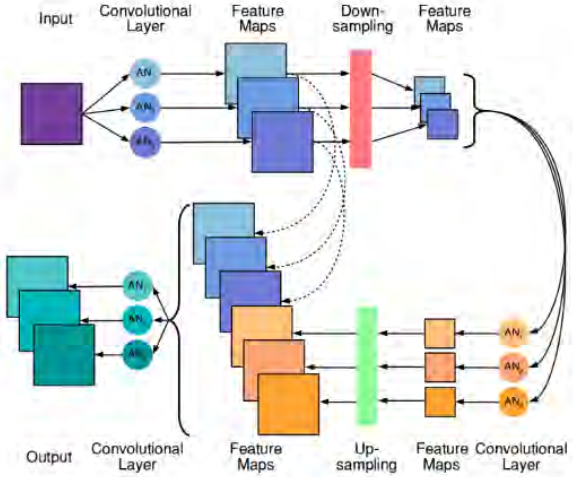
Ryan Hausen
Roman Science Team Community Briefing
November 19, 2021

This notebook accessed here:https://tinyurl.com/morpheus-roman

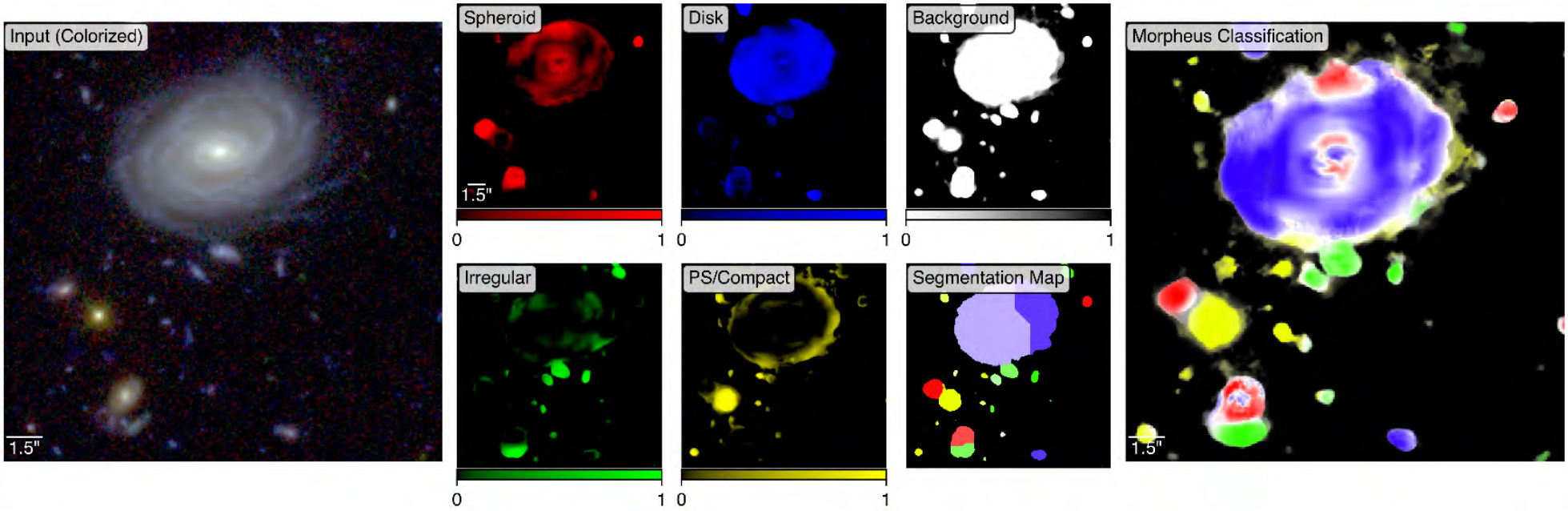
Morpheus

Morpheus (arXiv:1906.11248, ApJS, 248, 20) is a deep learning model and software package for the pixel-level morphological classifications of large scale astronomical images.

Using a U-Net like type Convolutional Neural Network, it detects and morphologically classifies sources:



Morpheus


Input (Colorized)
1.5"
Spheroid
1.5"
0
1
Disk
0
1
Background
0
1
Irregular
0
1
PS/Compact
0
1
Segmentation Map
Morpheus Classification
1.5"

GOODS-South Survey

In [83]:
```
from IPython.display import display, HTML, IFrame, YouTubeVideo
display(IFrame(src="http://slate.ucsc.edu/~brant/morpheus/mapv12/index_light.html", width=1000, height=600))
```

Import Required Libraries

In [42]:
```
# if your running on Colab or you haven't installed morpheus
# you need to install the package.
!pip install morpheus-astro
```

...

In [79]:
```
import numpy as np # linear algebra and array operations
import matplotlib.pyplot as plt # plotting tools
import tensorflow.compat.v1 as tf # morpheus is running tf v1 so import compatible version
# scales astronomical images for plotting
from astropy.visualization import AsymmetricPercentileInterval, simple_norm

# Import Morpheus ==================================
from morpheus.classifier import Classifier
from morpheus.data import example
# ==================================================

plt.style.use('default')
tf.logging.set_verbosity(tf.logging.ERROR)
%matplotlib inline
```

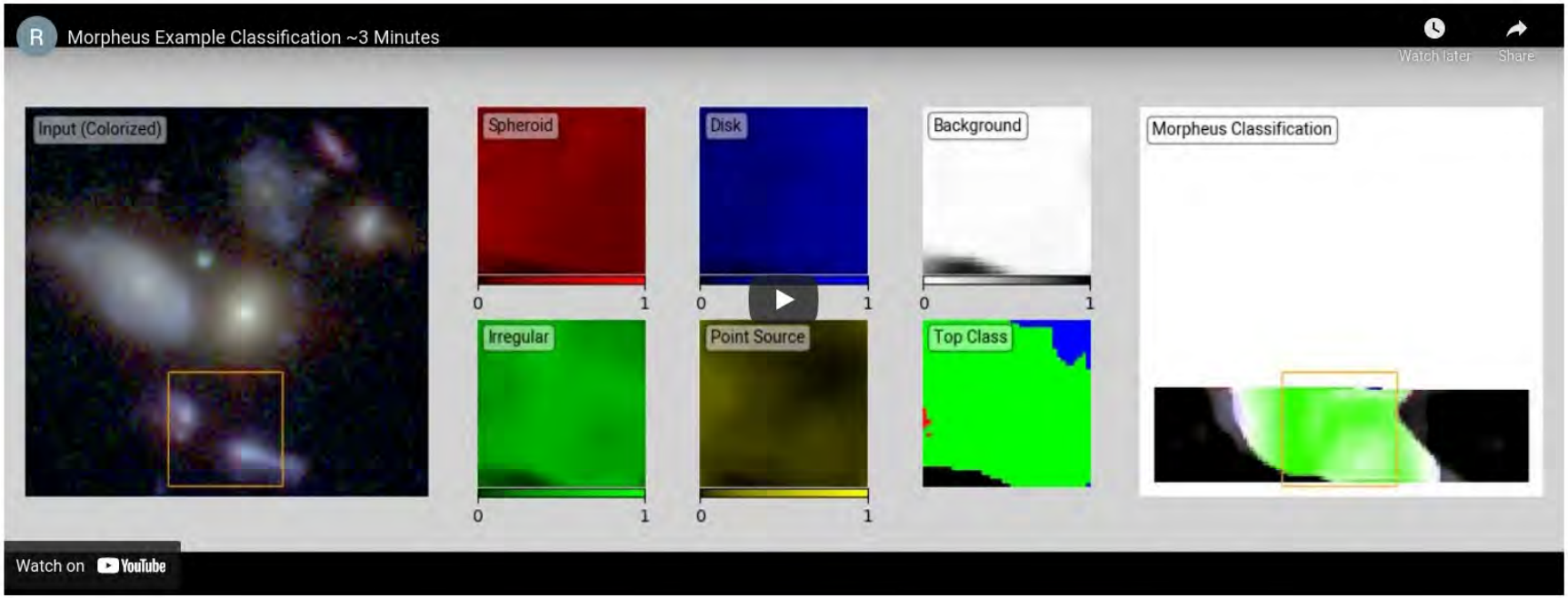
Classifying Variable Sized Images

The image can be any size as long as it's larger than 40×40. Morpheus can take care of the rest which includes:

- Windowing input image
- Aggregating outputs into a single output image
- Parallel classification over CPUs or NVIDIA GPUs

Classifying Variable Sized Images

In [84]:
```
1 display(YouTubeVideo("hEL1h_dODkU", width=1800, height=600))
```



Using Morpheus on the Example Image

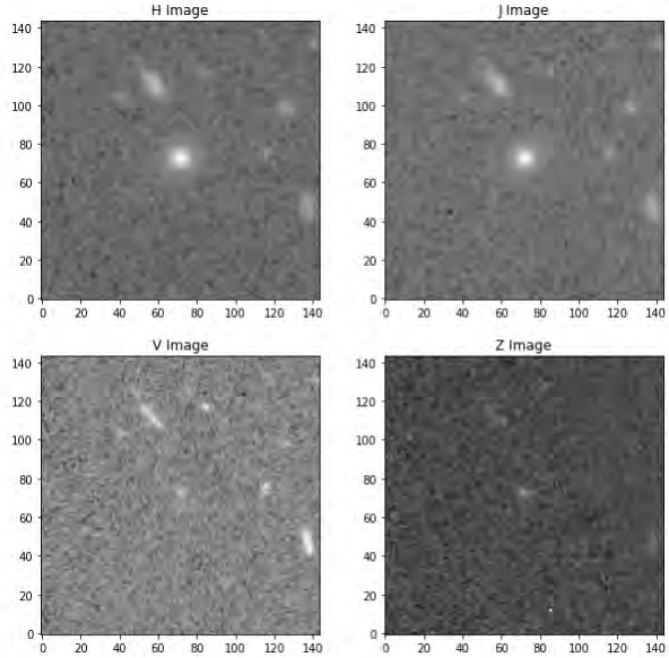
Morpheus was trained on the F125W, F160W, F606W, and F850LP band images and performs best when classifying an image with data in those bands.

In [45]:
```
h, j, v, z = example.get_sample()
```

Using Morpheus on the Example Image

In [77]:

```
f, axes = plt.subplots(nrows=2, ncols=2, figsize=(10,10))
axes = np.array(axes).flatten()

for ax, arr, band, s in zip(axes, [h,j,v,z], 'HJVZ', [*["log"]*2, *["asinh"]*2]):
    ax.set_title(f'{band} Image')
    if band:arr = np.clip(arr, *AsymmetricPercentileInterval(0.01, 99.999).get_limits(arr))
    ax.imshow(arr, origin='lower', cmap='gray', norm=simple_norm(arr, stretch=s))
```



Classifying Images

To classify an image, use Classifer.classify. The arguments can be either a file path or a numpy array. The output is a dictionary that contains the classifications as numpy arrays.

The output is a dictionary that contains a mapping for each pixel which represents the probability that a pixel belongs to one of the following classes:

- Spheroid
- Disk
- Irregular
- Point Source (Compact)
- Background

It also contains an output for the number of times a pixel was classified called `n`

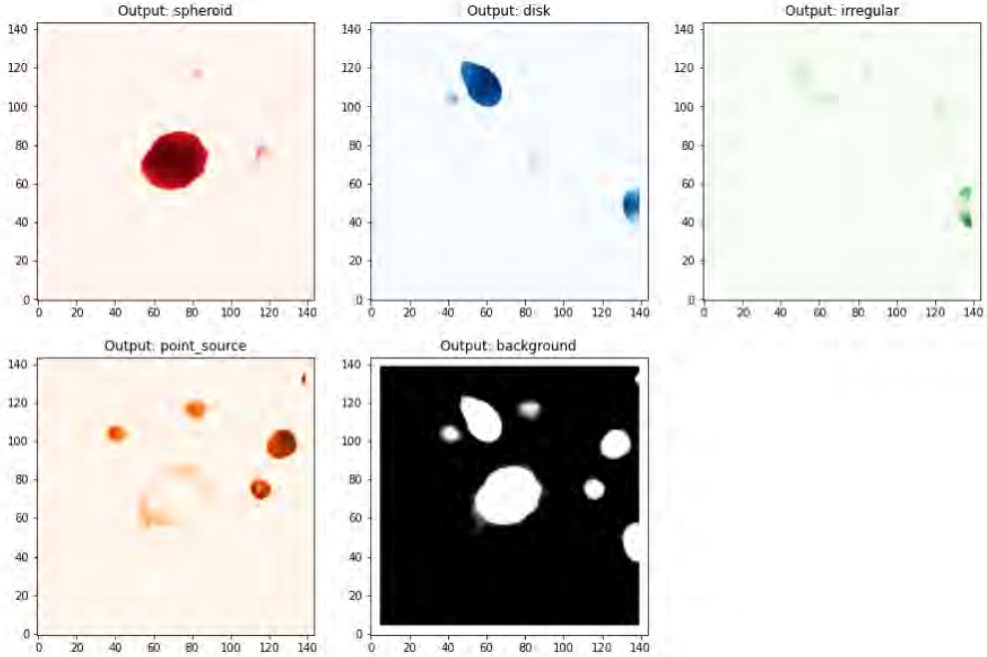
Classifying Images

Morpheus automatically extracts subsets of the image and classifies them in batches. In this example, Morpheus classifies approximately 10,000 subsets of the image and combines them back into the output.

```
In [47]: classified = Classifier.classify(h=h, j=j, v=v, z=z, batch_size=2500)

         classifying: 5batch [00:27,  5.43s/batch]
```

Classifying Images

In [48]:
```
f, axes = plt.subplots(nrows=2, ncols=3, figsize=(15, 10))
axes = np.array(axes).flatten()
cmaps = ["Reds", "Blues", "Greens", "Oranges", "binary"]

for i, k in enumerate(classified):
    if k=="n":axes[i].axis("off")
    else:
        axes[i].set_title(f'Output: {k}')
        axes[i].imshow(classified[k], origin='lower', vmin=0, vmax=1,cmap=cmaps[i])
plt.show()
```



Segmentation Map

After an image has been classified, a segmap can be created using Classifier.segmap_from_classified. For more information about the algorithm see the documentation for the function.

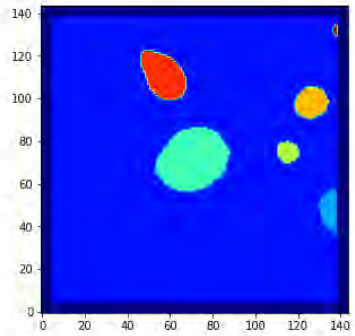
In [80]:

```
# the mask tells the segmentation mapping algorithm to ignore
# the areas unclassified as a result of the windowing classification
# method used.
mask = np.zeros_like(h, np.int)
mask[5:-5, 5:-5] = 1

segmap = Classifier.segmap_from_classified(classified, h, mask=mask)
print("Done!")
```

```
/tmp/ipykernel_262664/3855617634.py:4: DeprecationWarning: `np.int` is a deprecated alias for the builtin `int`. To silence this wa
rning, use `int` by itself. Doing this will not modify any behavior and is safe. When replacing `np.int`, you may wish to use e.g.
`np.int64` or `np.int32` to specify the precision. If you wish to review your current use, check the release note link for addition
al information.
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  mask = np.zeros_like(h, np.int)

Building Markers...
Watershedding...

/data/groups/comp-astro/ryanhausen/env3.8/lib/python3.8/site-packages/skimage/morphology/_deprecated.py:5: skimage_deprecation: Fun
ction ``watershed`` is deprecated and will be removed in version 0.19. Use ``skimage.segmentation.watershed`` instead.
  def watershed(image, markers=None, connectivity=1, offset=None, mask=None,
Deblending: 100%|[redacted]| 6/6 [00:00<00:00, 2552.06it/s]

Done!
```

Segmentation Map

In [51]:
```
plt.figure(figsize=(5, 5))
plt.imshow(segmap, origin='lower', cmap='jet')
plt.show()
```



Cataloging

Morpheus provides a cataloging functionality via Classifier.catalog_from_classified for images that will return all of the detected sources and their morphological classifications.

In [52]:
```
catalog = Classifier.catalog_from_classified(classified, h, segmap)
```

Cataloging

In [53]:

```
print('Source ID\tLocation(y,x)\tMorphology:[Sph,Dsk,Irr,Ps]')
for source in catalog:
    _id = source['id']
    loc = source['location']
    morph = np.round(source['morphology'], decimals=2) # round for readability
    print('{}\t\t{}\t{}'.format(_id, loc, morph))

```

```
Source ID       Location(y,x)   Morphology:[Sph,Dsk,Irr,Ps]
1               [44, 138]       [0.   0.69 0.31 0.  ]
2               [72, 72]        [0.98 0.   0.   0.01]
3               [76, 116]       [0.26 0.   0.   0.73]
4               [98, 126]       [0.04 0.02 0.07 0.87]
5               [110, 58]       [0.   0.95 0.05 0.  ]
6               [131, 138]      [0. 0. 0. 1.]
```

Colorizing a Morphological Classification

Use Classifier.colorize_classified to make an RGB. The colors in the output have the following meanings:

- Red = Spheroid
- Blue = Disk
- Green = Irregular
- Yellow = Point Source (compact)
- Back = Background

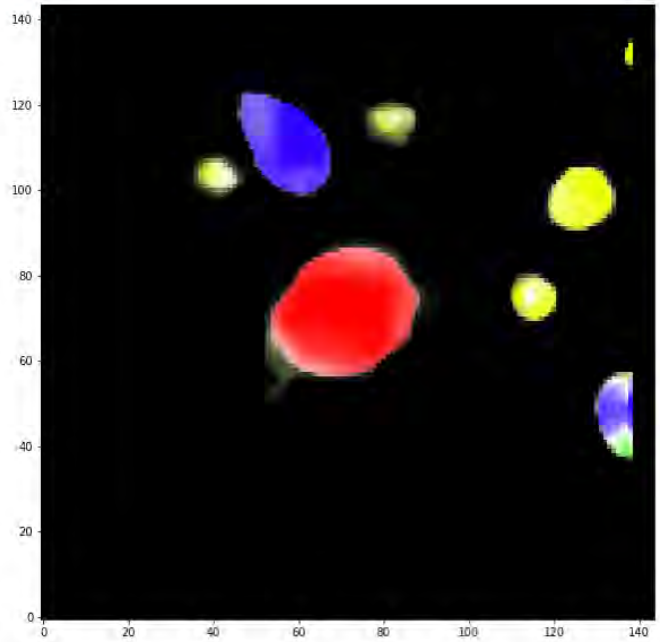
Note when Morpheus doesn't output a strong classification for any one class they are colored white.

For more information on the coloring scheme see the documentation.

Colorizing a Morphological Classification

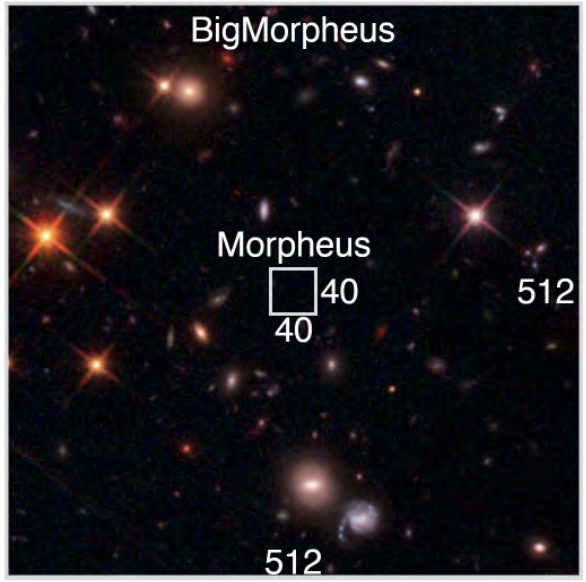
In [40]:
```
1 rgb = Classifier.colorize_classified(classified)
2 plt.figure(figsize=(10, 10))
3 plt.imshow(rgb, origin='lower')
4 plt.show()
```

```
100%|██████████| 144/144 [00:00<00:00, 6858.18it/s]
```



BigMorpheus

BigMorpheus (Hausen & Robertson in prep) leverages the images/classifications from Morpheus to train a much larger classifier that uses a single band.



Scaling Up Your Model: Morpheus-Core

Morphologies are neat, but can we also do ____?

Probably! We have released the core "windowing" algorithm from Morpheus as **Morpheus-Core** (https://github.com/morpheus-project/morpheus-core) which is also pip installable.

```
pip install morpheus-core
```

Morpheus-Core will take care of the following:

- Windowing a larger image into batches at the size (height, width) and stride you set
- Aggregating output classifications into an output image of the same size as the input image
- Parallelizing the classification/regression of a large image over CPUs or NVIDIA GPUs

Summary

- Morpheus is machine learning model that performs detects and performs pixel-level morphological classifications that can scale up survey sized images
- BigMorpheus demonstrates how the initial Morpheus data product can be leveraged to create larger faster models for specialized datasets
- Morpheus-Core can scale up any kind of pixel-level model to survey sized images, handling memory and CPU/GPU parallelization for the user