Galaxy Morphologies
X
Machine Learning

EURECA meeting
Feb 14, 2025

Yongda Zhu (JWST NIRCam & MIRI Teams)

Galaxy Morphologies X Machine Learning

EUREKA meeting

Feb 14, 2025

⚠️ I am not a professional Machine Learning Engineer

Yongda Zhu (JWST NIRCam & MIRI Teams)

Galaxy Morphologies
X
Machine Learning

EUREKA meeting
Feb 14, 2025

Yongda Zhu (JWST NIRCam & MIRI Teams)

⚠️ I am not a professional Machine Learning Engineer

😀But at least some tricks work for galaxies!

GPT: “go and use ML!”

Image: https://jades.idies.jhu.edu
JADES Team

GPT: “go and use ML!”

Image: https://jades.idies.jhu.edu
JADES Team

In this image, the two galaxies can be classified based on their visual morphology:

1. **Galaxy on the left:**
    - Appears elliptical or lenticular (S0), with a smooth, featureless structure and a bright central core.
    - No visible spiral arms or disk-like features, which is characteristic of elliptical or early-type galaxies.
2. **Galaxy on the right:**
    - Clearly a spiral galaxy, with visible spiral arms and a bright core.
    - Likely a star-forming galaxy, given the structured arms that may contain regions of active star formation.

This classification can be refined using tools such as machine learning models that analyze multiple features, including color, light distribution, and structural components. Would you like a demonstration on how ML might analyze similar images?

GPT: “go and use ML!”

Image: https://jades.idies.jhu.edu
JADES Team

In this image, the two galaxies can be classified based on their visual morphology:

1. **Galaxy on the left:**
   - Appears elliptical or lenticular (S0), with a smooth, featureless structure and a bright central core.
   - No visible spiral arms or disk-like features, which is characteristic of elliptical or early-type galaxies.
2. **Galaxy on the right:**
   - Clearly a spiral galaxy, with visible spiral arms and a bright core.
   - Likely a star-forming galaxy, given the structured arms that may contain regions of active star formation.

This classification can be refined using tools such as machine learning models that analyze multiple features, including color, light distribution, and structural components. Would you like a demonstration on how ML might analyze similar images?

Machine learning basics

Dealing with galaxies

Machine learning basics

Dealing with galaxies

- Case 1 - traditional method
  - Extract features (brightness, color, axis ratio, etc.) by hand, pre-label (spiral/elliptical, SF/QG, AGN/non-AGN) by hand, and then train a model (Random Forests, Support Vector Machine (SVM), Gaussian Process, etc.)
  - The input have to be well structured
  - Pros: every step is well defined, minimal hyper-parameter tuning

Machine learning basics

Dealing with galaxies

- Case 1 - traditional method
  - Extract features (brightness, color, axis ratio, etc.) by hand, pre-label (spiral/elliptical, SF/QG, AGN/non-AGN) by hand, and then train a model (Random Forests, Support Vector Machine (SVM), Gaussian Process, etc.)
  - The input have to be well structured
  - Pros: every step is well defined, minimal hyper-parameter tuning
- Case 2 - automatic feature extraction (unsupervised ML)
  - Automatic segmentation, de-blending, measurements
  - Pros: labor saving (really?) Cons: lots of debugging and hyper-parameter tuning

Machine learning basics

Dealing with galaxies

- Case 1 - traditional method
    - Extract features (brightness, color, axis ratio, etc.) by hand, pre-label (spiral/elliptical, SF/QG, AGN/non-AGN) by hand, and then train a model (Random Forests, Support Vector Machine (SVM), Gaussian Process, etc.)
    - The input have to be well structured
    - Pros: every step is well defined, minimal hyper-parameter tuning
- Case 2 - automatic feature extraction (unsupervised ML)
    - Automatic segmentation, de-blending, measurements
    - Pros: labor saving (really?) Cons: lots of debugging and hyper-parameter tuning
- Case 3 - Deep Learning / Neural Network
    - ~~Magic~~

Case 1: traditional methods

ML-101: iris flowers classification

Features
(can be n-D pixels,
measurements, etc.)
[Length of the petal / sepal]
[photometry /
parametric shapes]

Some linear algebra
(Not a black box!)
[Random Forest, SVM, etc.]

Classification
[setosa, versicolor, virginica]
[SF/QG/AGN with probability]

A very helpful notebook: https://github.com/Apaulgithub/oibsip_taskno1

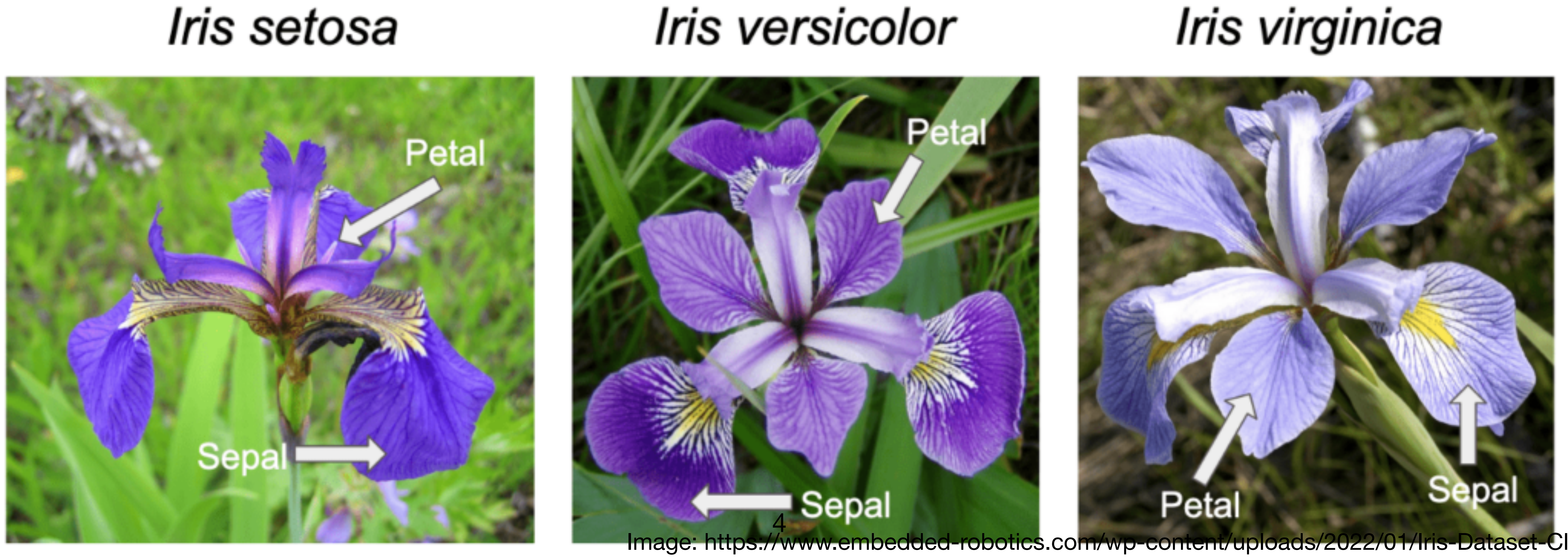
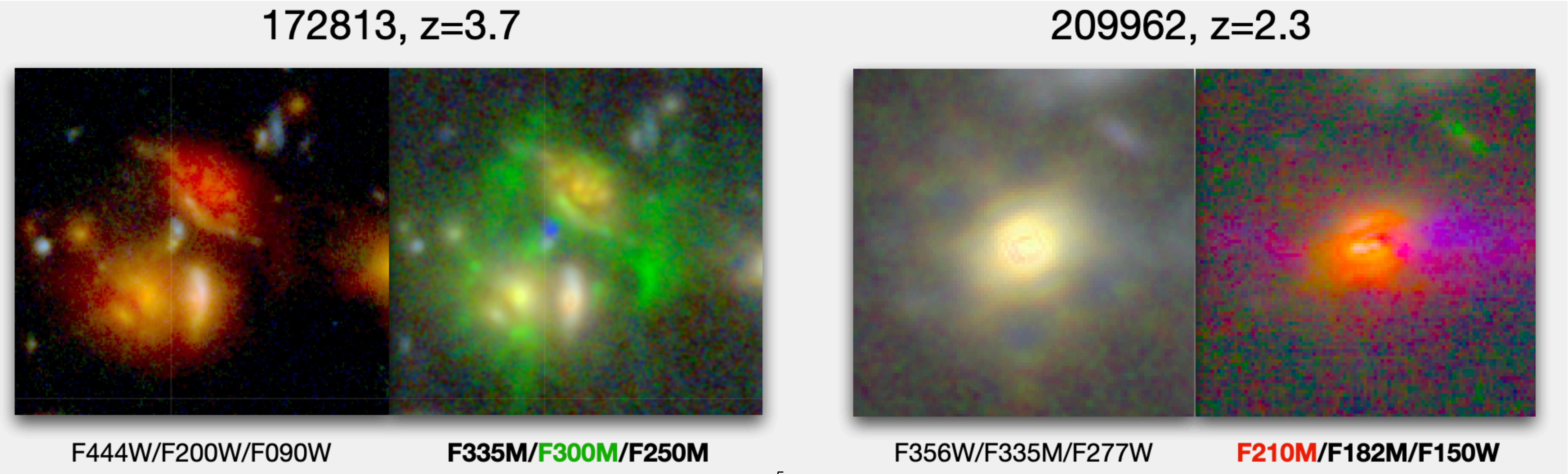


Image: https://www.embedded-robotics.com/wp-content/uploads/2022/01/Iris-Dataset-Classification-1024x367.png

Case 2: Automatic segmentation, de-blending, measurements

Example 201: searching for outflow candidates by looking for extended emission lines in medium band imaging

arXiv: 2409.11464

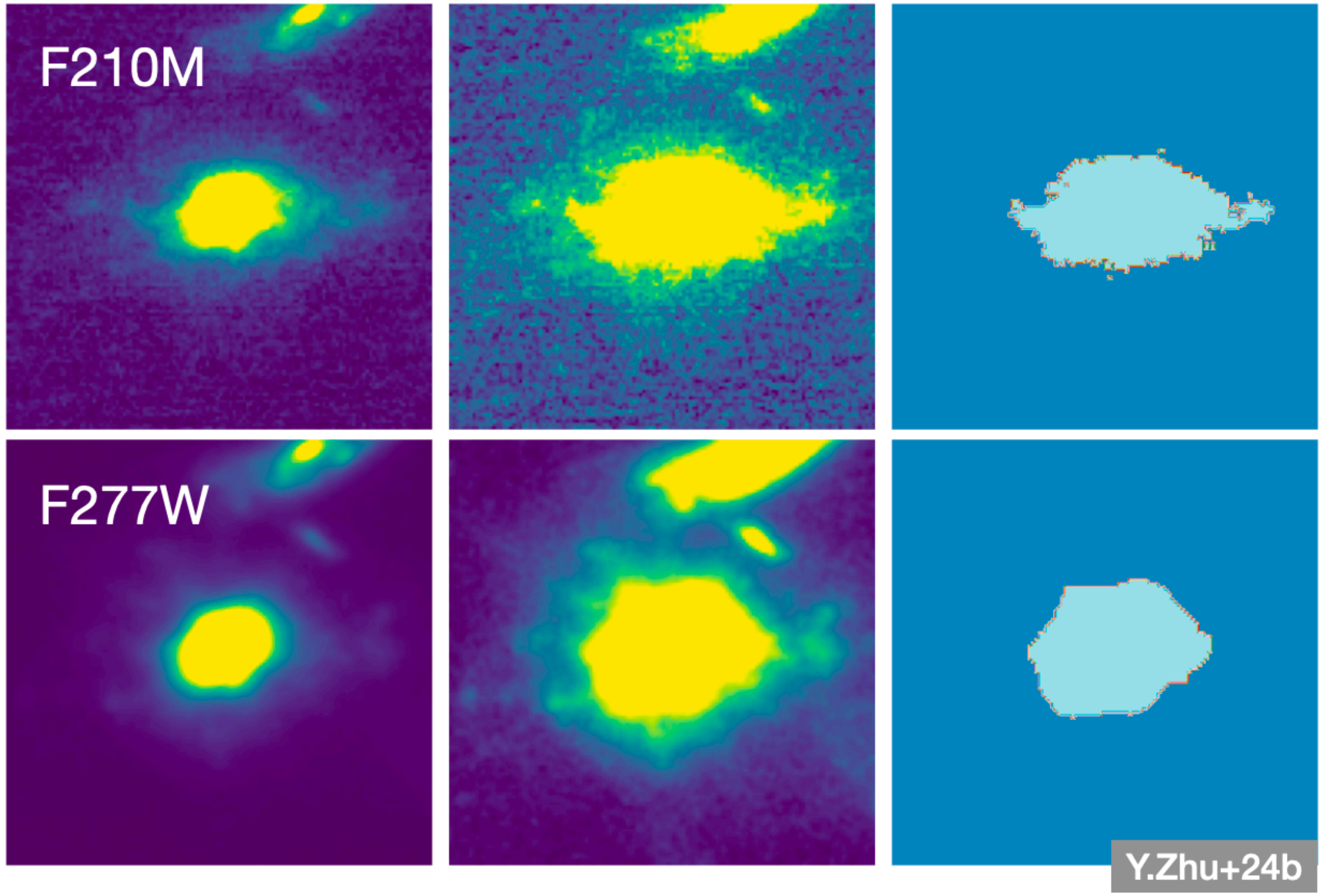


Case 2: Automatic segmentation, de-blending, measurements

Example 201: searching for outflow candidates by looking for extended emission lines in medium band imaging

Case 2: Automatic segmentation, de-blending, measurements

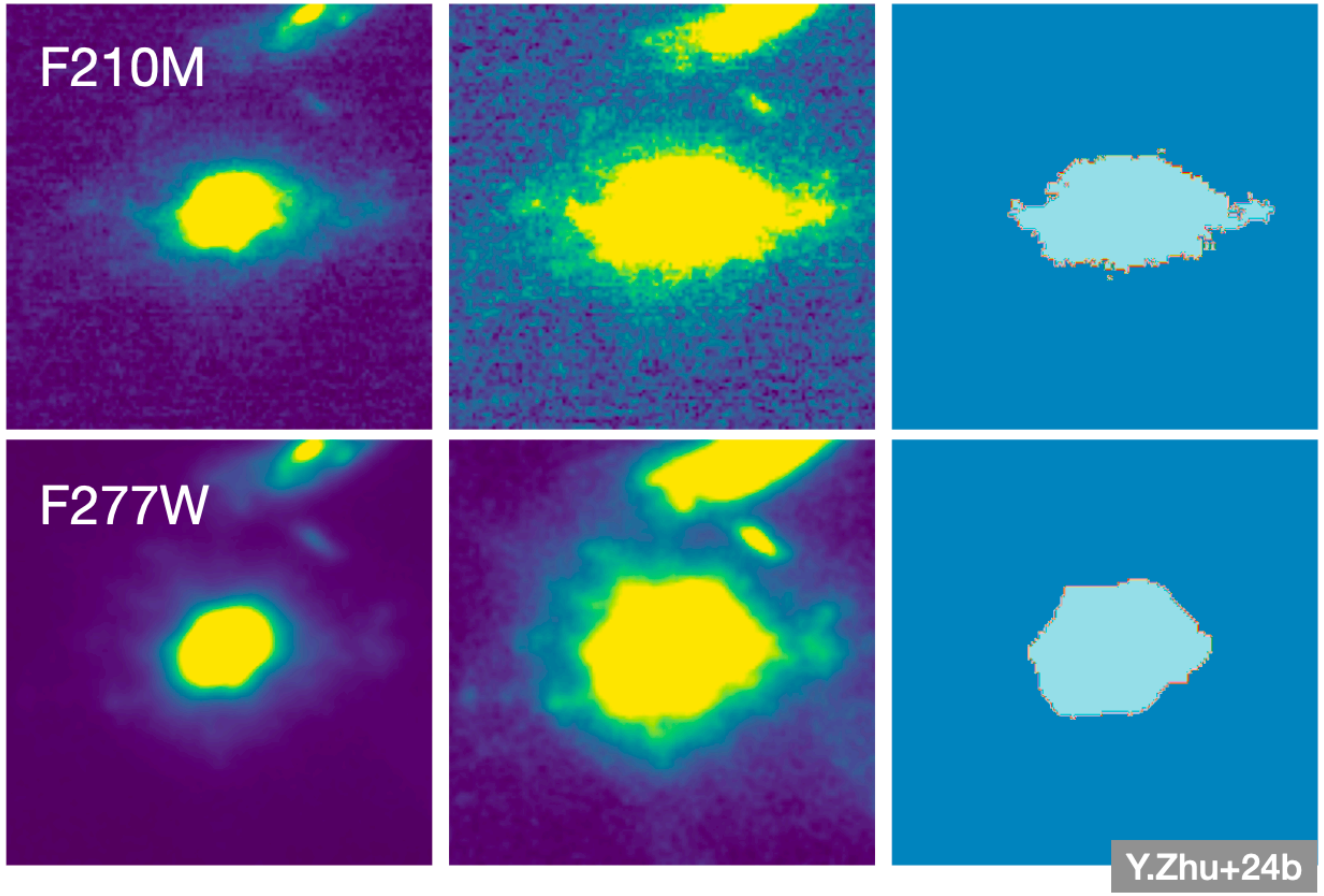
Example 201: searching for outflow candidates by looking for extended emission lines in medium band imaging



Case 2: Automatic segmentation, de-blending, measurements

Example 201: searching for outflow candidates by looking for extended emission lines in medium band imaging

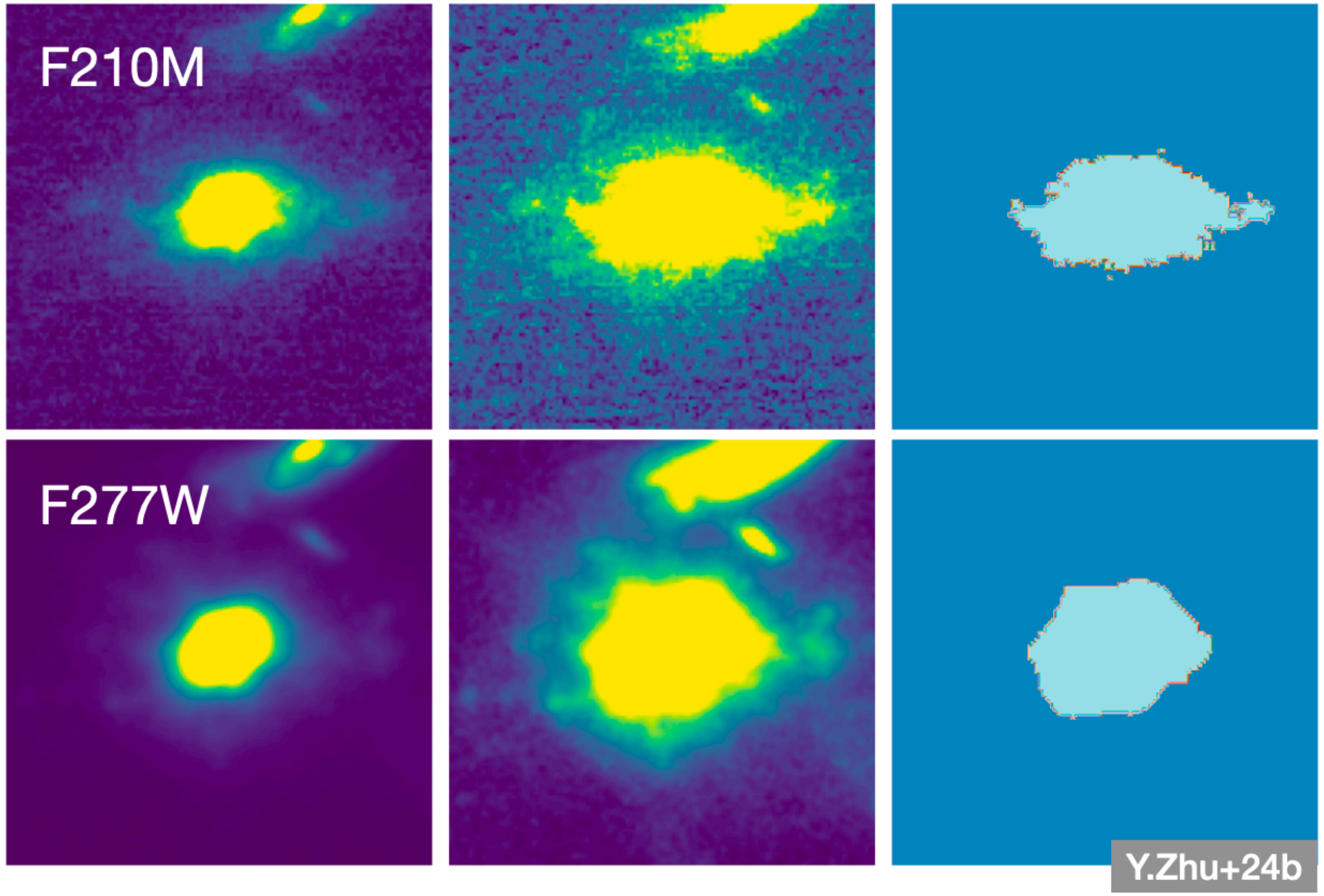
- DBSCAN (Ester et al. 1995): Density-based spatial clustering of applications with noise
  - Arbitrarily-shaped clusters
  - Arbitrary number of clusters (vs k-means)
  - Robust to outliers and noise
  - Fast



Case 2: Automatic segmentation, de-blending, measurements

Example 201: searching for outflow candidates by looking for extended emission lines in medium band imaging

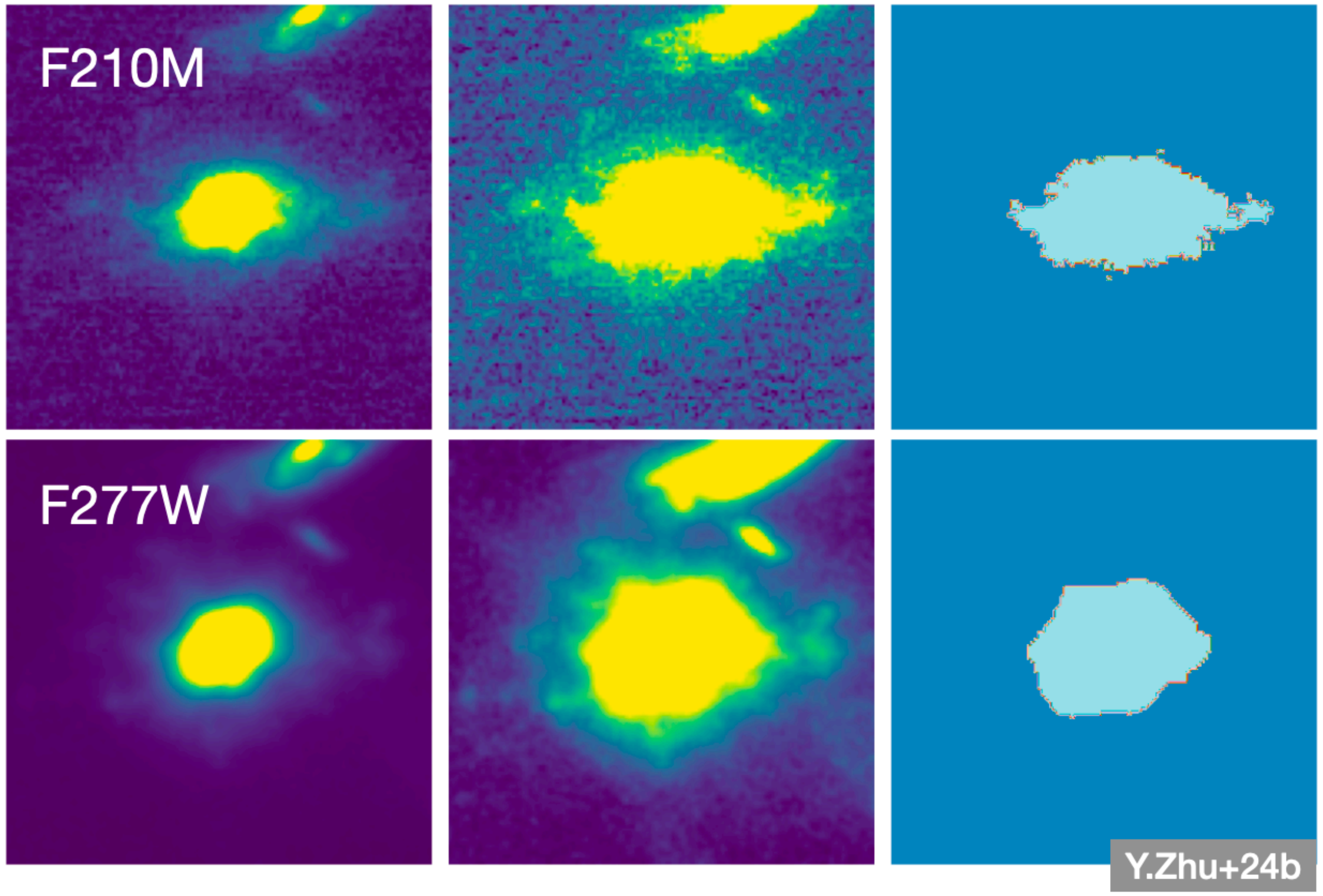
- DBSCAN (Ester et al. 1995): Density-based spatial clustering of applications with noise
  - Arbitrarily-shaped clusters
  - Arbitrary number of clusters (vs k-means)
  - Robust to outliers and noise
  - Fast
- Grouping pixels based on their density in n-D space (color, flux, etc)



Case 2: Automatic segmentation, de-blending, measurements

Example 201: searching for outflow candidates by looking for extended emission lines in medium band imaging

- DBSCAN (Ester et al. 1995): Density-based spatial clustering of applications with noise
  - Arbitrarily-shaped clusters
  - Arbitrary number of clusters (vs k-means)
  - Robust to outliers and noise
  - Fast
- Grouping pixels based on their density in n-D space (color, flux, etc)
- https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html



Case 2: Automatic segmentation, de-blending, measurements

Example 201: searching for outflow candidates by looking for extended emission lines in medium band imaging

```
>>> from sklearn.cluster import DBSCAN
>>> import numpy as np
>>> X = np.array([[1, 2], [2, 2], [2, 3],
...               [8, 7], [8, 8], [25, 80]])
>>> clustering = DBSCAN(eps=3, min_samples=2).fit(X)
>>> clustering.labels_
array([ 0,  0,  0,  1,  1, -1])
>>> clustering
DBSCAN(eps=3, min_samples=2)
```

Case 2: Automatic segmentation, de-blending, measurements

Example 201: searching for outflow candidates by looking for extended emission lines in medium band imaging

```
>>> from sklearn.cluster import DBSCAN
>>> import numpy as np
>>> X = np.array([[1, 2], [2, 2], [2, 3],
...               [8, 7], [8, 8], [25, 80]])
>>> clustering = DBSCAN(eps=3, min_samples=2).fit(X)
>>> clustering.labels_
array([ 0,  0,  0,  1,  1, -1])
>>> clustering
DBSCAN(eps=3, min_samples=2)
```

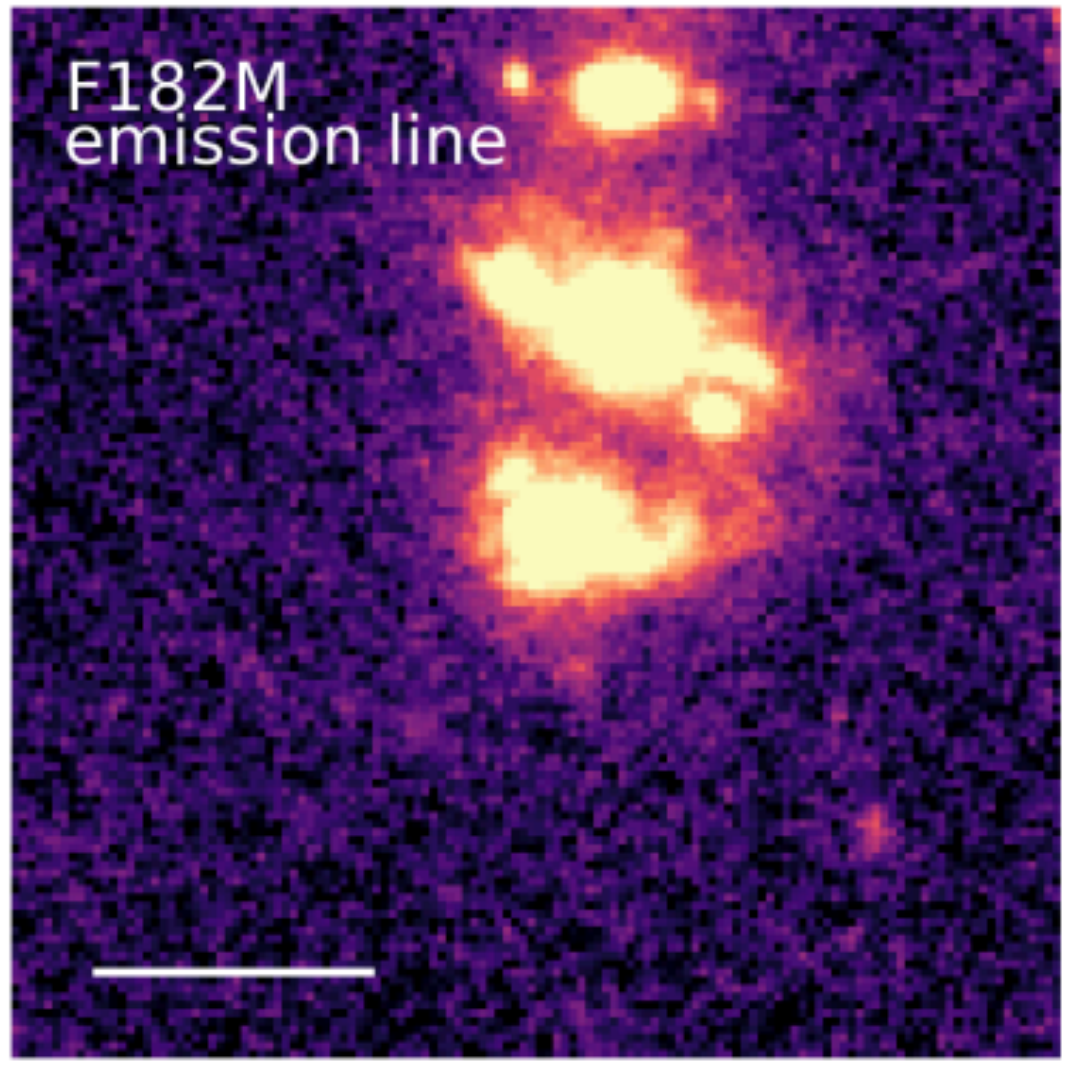
- X = np.array([[x1, y1, flux1], …[xn, yn, fluxn]])
- eps: the parameter you need to tune
- min_samples: the min sample size to grow a cluster
- Then measure each clusters (n_pix, total flux, axis ratio, gini, AI, etc.)

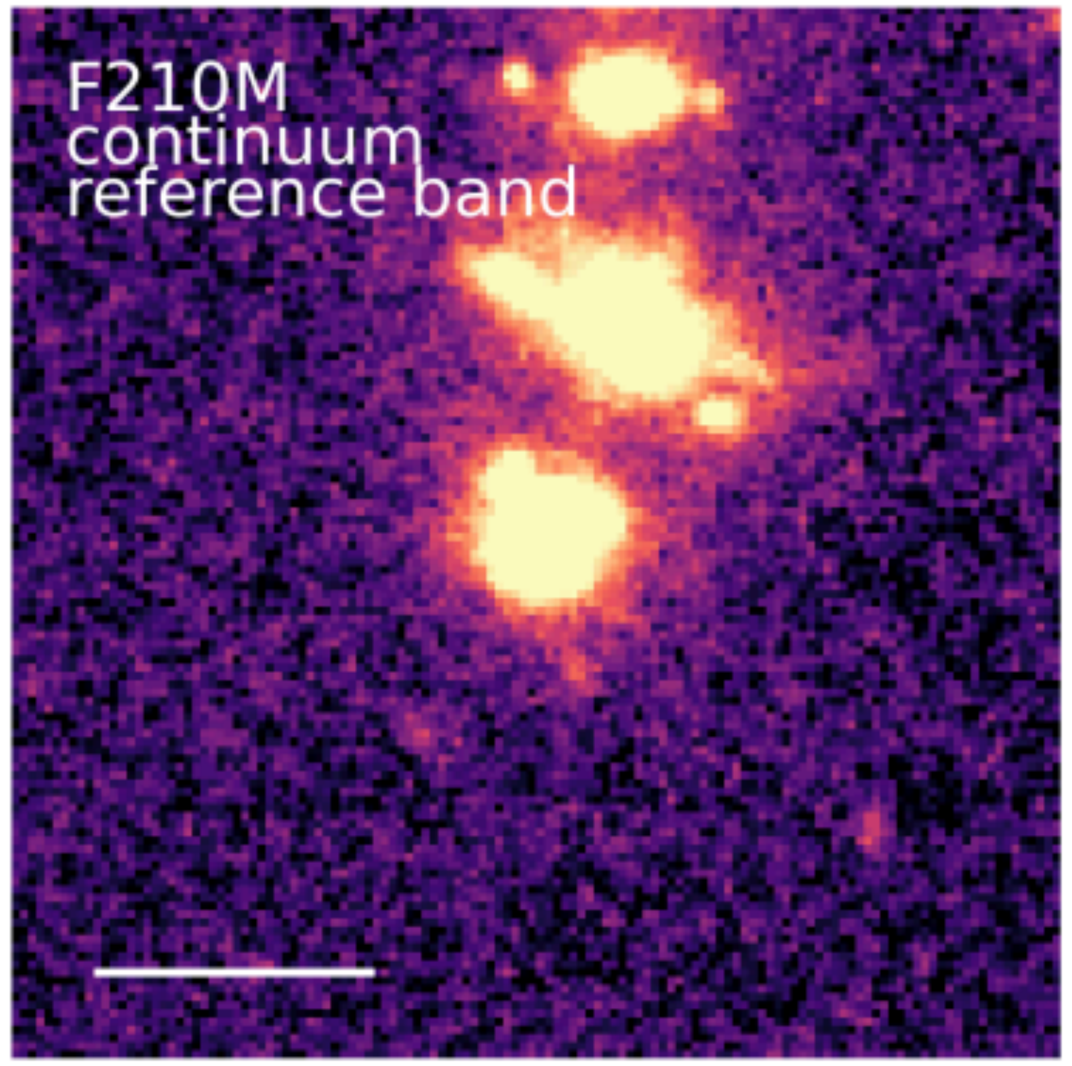
Case 2: Automatic segmentation, de-blending, measurements

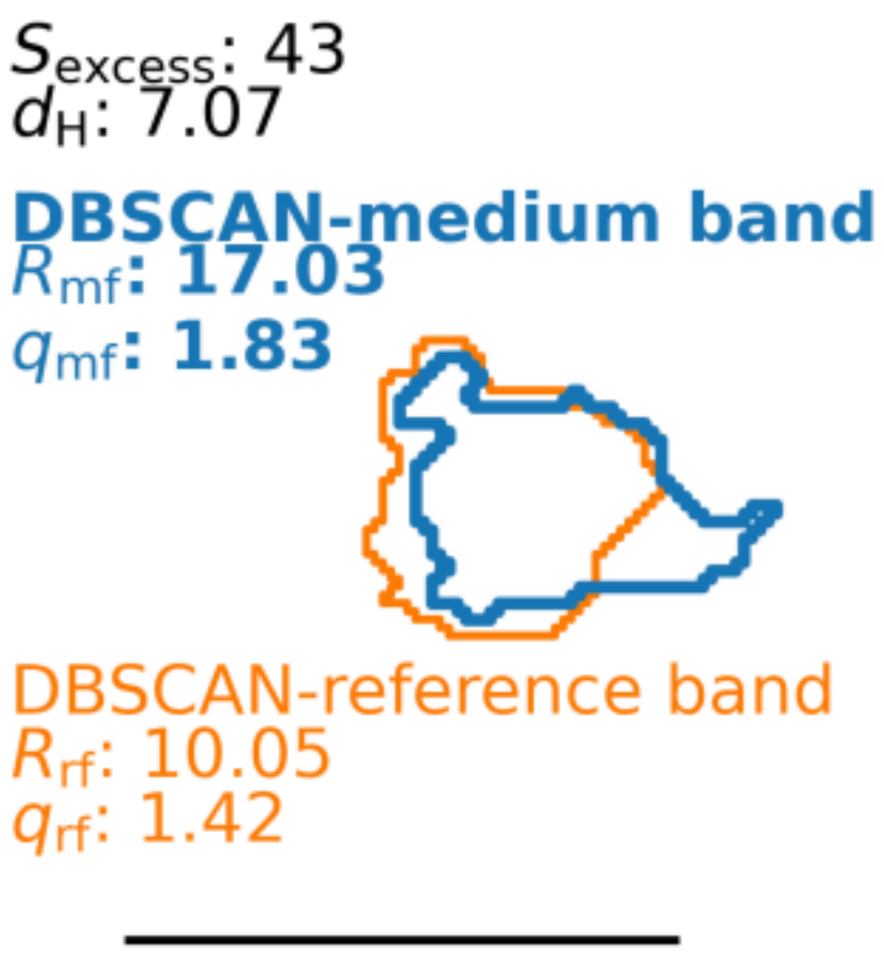
Example 201: searching for outflow candidates by looking for extended emission lines in medium band imaging

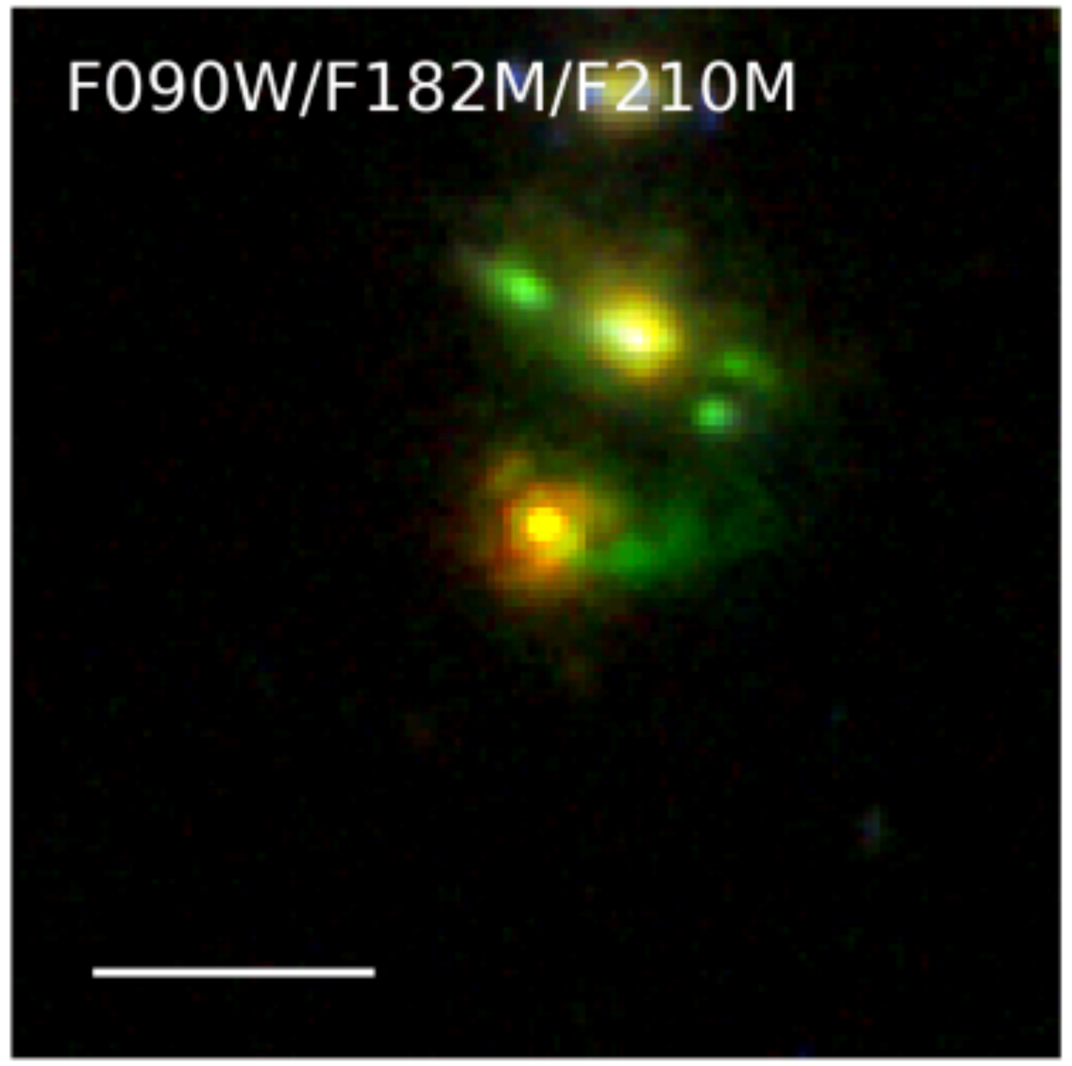
```
>>> from sklearn.cluster import DBSCAN
>>> import numpy as np
>>> X = np.array([[1, 2], [2, 2], [2, 3],
...               [8, 7], [8, 8], [25, 80]])
>>> clustering = DBSCAN(eps=3, min_samples=2).fit(X)
>>> clustering.labels_
array([ 0,  0,  0,  1,  1, -1])
>>> clustering
DBSCAN(eps=3, min_samples=2)
```

- X = np.array([[x1, y1, flux1], …[xn, yn, fluxn]])
- eps: the parameter you need to tune
- min_samples: the min sample size to grow a cluster
- Then measure each clusters (n_pix, total flux, axis ratio, gini, AI, etc.)







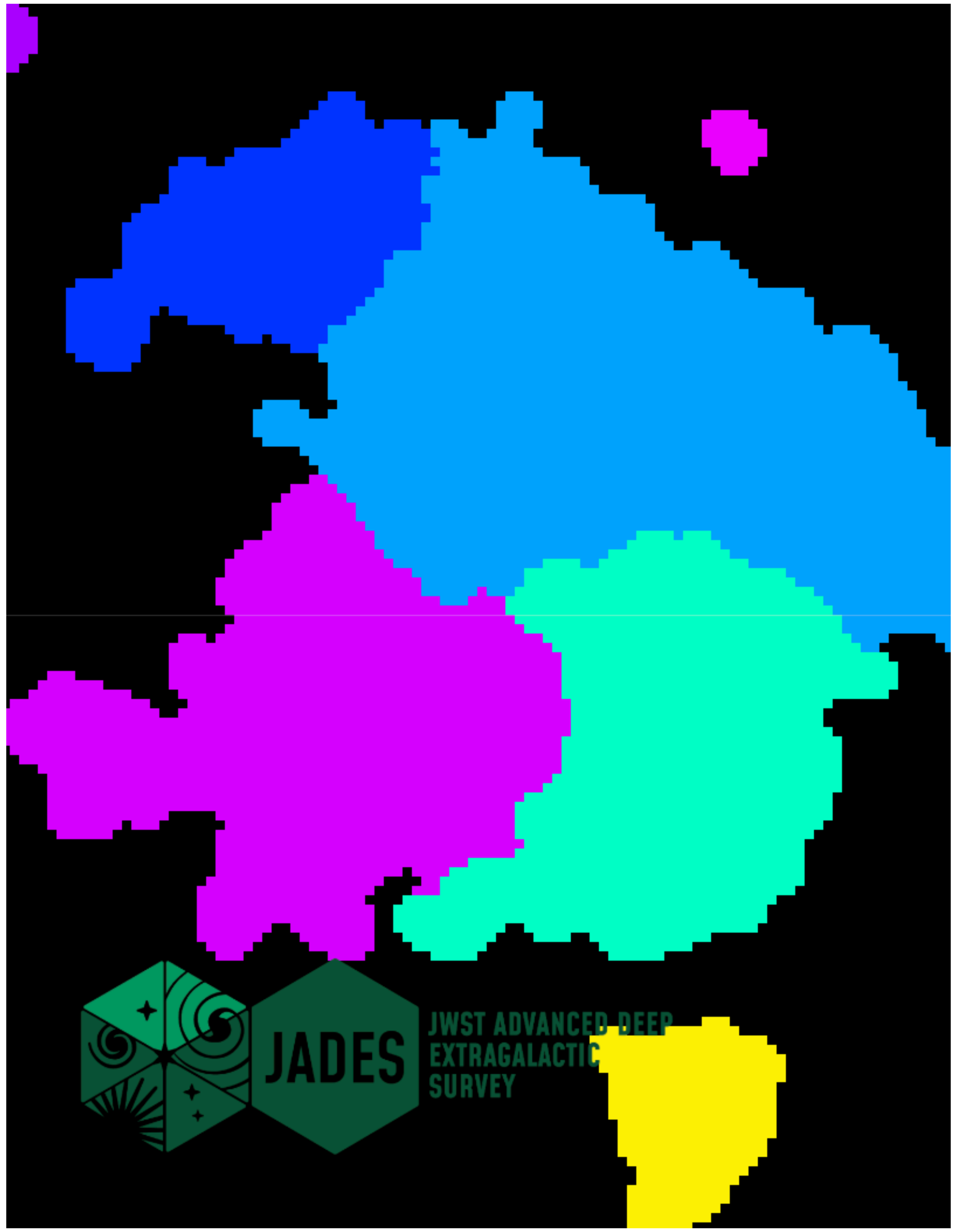


Case 2: Automatic segmentation, de-blending, measurements

Example 202: galaxy de-blending (or can be sub-structure de-blending)

Case 2: Automatic segmentation, de-blending, measurements

Example 202: galaxy de-blending (or can be sub-structure de-blending)



Case 2: Automatic segmentation, de-blending, measurements

Example 202: galaxy de-blending (or can be sub-structure de-blending)

Zhu+25, in prep

Case 2: Automatic segmentation, de-blending, measurements

Example 202: galaxy de-blending (or can be sub-structure de-blending)

Zhu+25, in prep

- Watershed segmentation: using topographic features to separate objects in an image.

Case 2: Automatic segmentation, de-blending, measurements

Example 202: galaxy de-blending (or can be sub-structure de-blending)

Zhu+25, in prep

- Watershed segmentation: using topographic features to separate objects in an image.
- Robust to noise and irregular shapes

Case 2: Automatic segmentation, de-blending, measurements

Example 202: galaxy de-blending (or can be sub-structure de-blending)

Zhu+25, in prep

- Watershed segmentation: using topographic features to separate objects in an image.
- Robust to noise and irregular shapes
- 1. The image is treated as a topographic surface.

Case 2: Automatic segmentation, de-blending, measurements

Example 202: galaxy de-blending (or can be sub-structure de-blending)

Zhu+25, in prep

- Watershed segmentation: using topographic features to separate objects in an image.
- Robust to noise and irregular shapes
- 1. The image is treated as a topographic surface.
- 2. The algorithm identifies catchment basins based on pixel intensity.

Case 2: Automatic segmentation, de-blending, measurements

Example 202: galaxy de-blending (or can be sub-structure de-blending)

Zhu+25, in prep

- Watershed segmentation: using topographic features to separate objects in an image.
- Robust to noise and irregular shapes
- 1. The image is treated as a topographic surface.
- 2. The algorithm identifies catchment basins based on pixel intensity.
- 3. The algorithm floods basins from user-defined markers.

Case 2: Automatic segmentation, de-blending, measurements

Example 202: galaxy de-blending (or can be sub-structure de-blending)

Zhu+25, in prep

- Watershed segmentation: using topographic features to separate objects in an image.
- Robust to noise and irregular shapes
- 1. The image is treated as a topographic surface.
- 2. The algorithm identifies catchment basins based on pixel intensity.
- 3. The algorithm floods basins from user-defined markers.
- 4. The algorithm separates each energy concentration region with watershed boundaries.

Case 2: Automatic segmentation, de-blending, measurements

Example 202: galaxy de-blending (or can be sub-structure de-blending)

Zhu+25, in prep

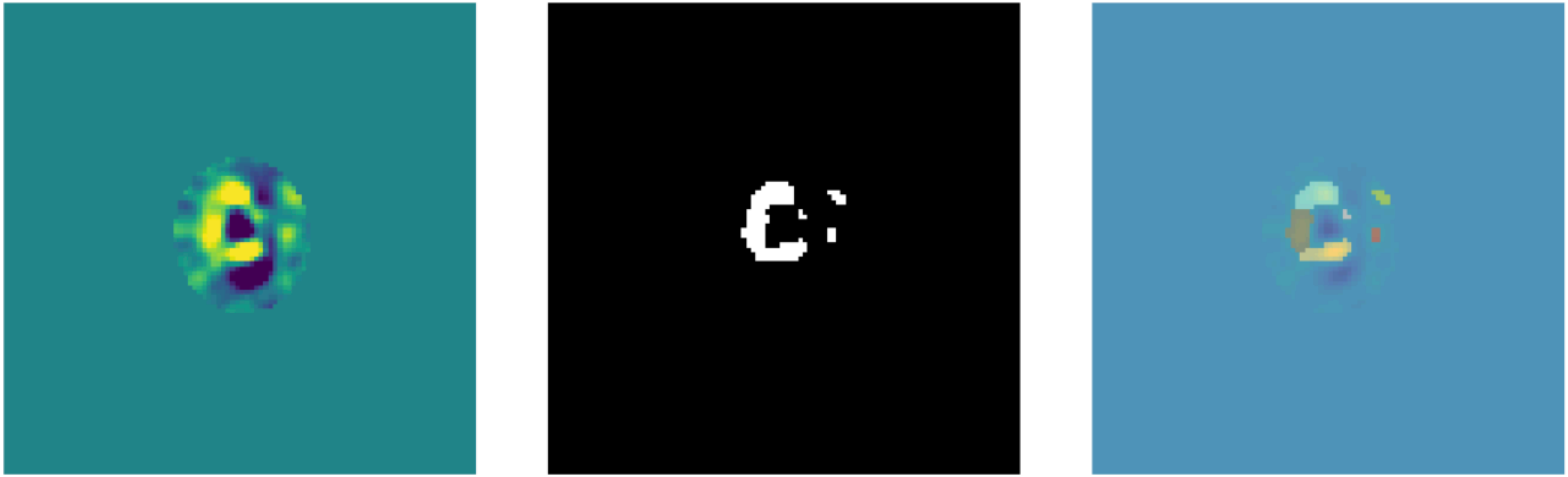
- Watershed segmentation: using topographic features to separate objects in an image.
- Robust to noise and irregular shapes
- 1. The image is treated as a topographic surface.
- 2. The algorithm identifies catchment basins based on pixel intensity.
- 3. The algorithm floods basins from user-defined markers.
- 4. The algorithm separates each energy concentration region with watershed boundaries.

```python
# necessary imports
import numpy as np
from skimage.filters import sobel
from skimage.measure import label
from skimage.segmentation import watershed
from skimage.feature import peak_local_max
from skimage.morphology import distance_transform_edt
def deblending(input_image, rms_noise):
    """
    Deblend a single-band image using a combination of thresholding, gradient magnitude, and watershed segmentation.
    author: Yongda Zhu
    Args:
    input_image (2D numpy array): The input image to be deblended.
    rms_noise (float): The RMS noise level of the input image.
    Returns:
    labels (2D numpy array): The deblended labels.
    binary_mask (2D numpy array): The binary mask used for deblending.
    """
    # Define a threshold for the bright regions (e.g., 3-sigma above the mean)
    threshold = rms_noise * 3
    binary_mask = input_image > threshold
    # Compute the gradient magnitude
    gradient_magnitude = sobel(input_image)
    # Compute a distance map for the binary mask
    distance = distance_transform_edt(binary_mask)
    # Identify local maxima for watershed segmentation
    local_maxi = peak_local_max(distance, footprint=np.ones((3, 3)), min_distance=5, labels=binary_mask)
    # Convert local_maxi (coordinates) to a binary mask of the same shape as `binary_mask`
    local_maxi_mask = np.zeros_like(binary_mask, dtype=bool)
    local_maxi_mask[tuple(local_maxi.T)] = True  # Convert coordinates to a mask
    # Create a markers array for watershed segmentation
    markers, _ = label(local_maxi_mask)
    # Apply watershed segmentation
    labels = watershed(-distance, markers, mask=binary_mask)
    return labels, binary_mask
```

Case 2: Automatic segmentation, de-blending, measurements

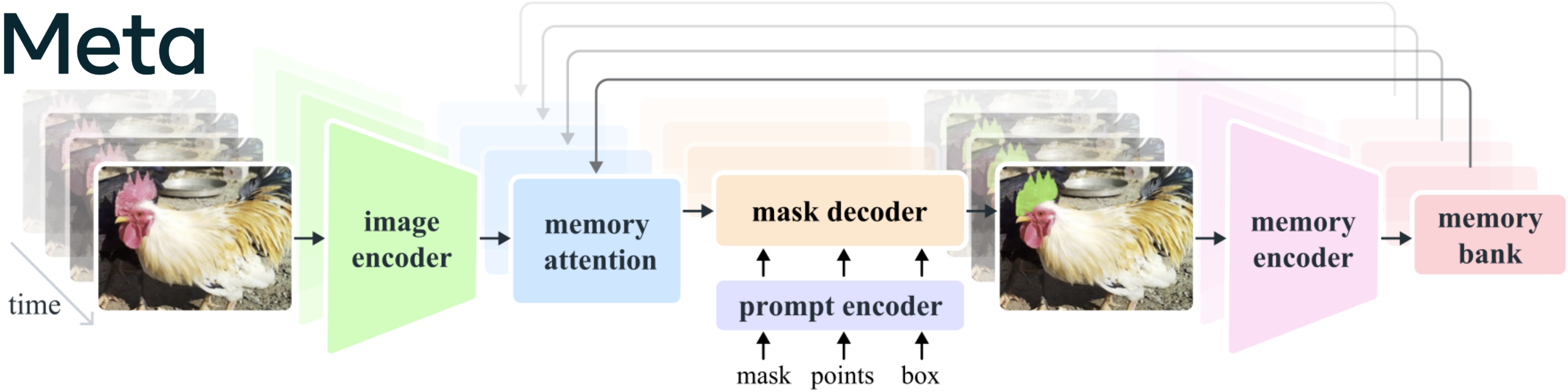
Example 202: galaxy de-blending (or can be sub-structure de-blending)

Zhu+25, in prep



Case 3: Neural Networks / Deep learning

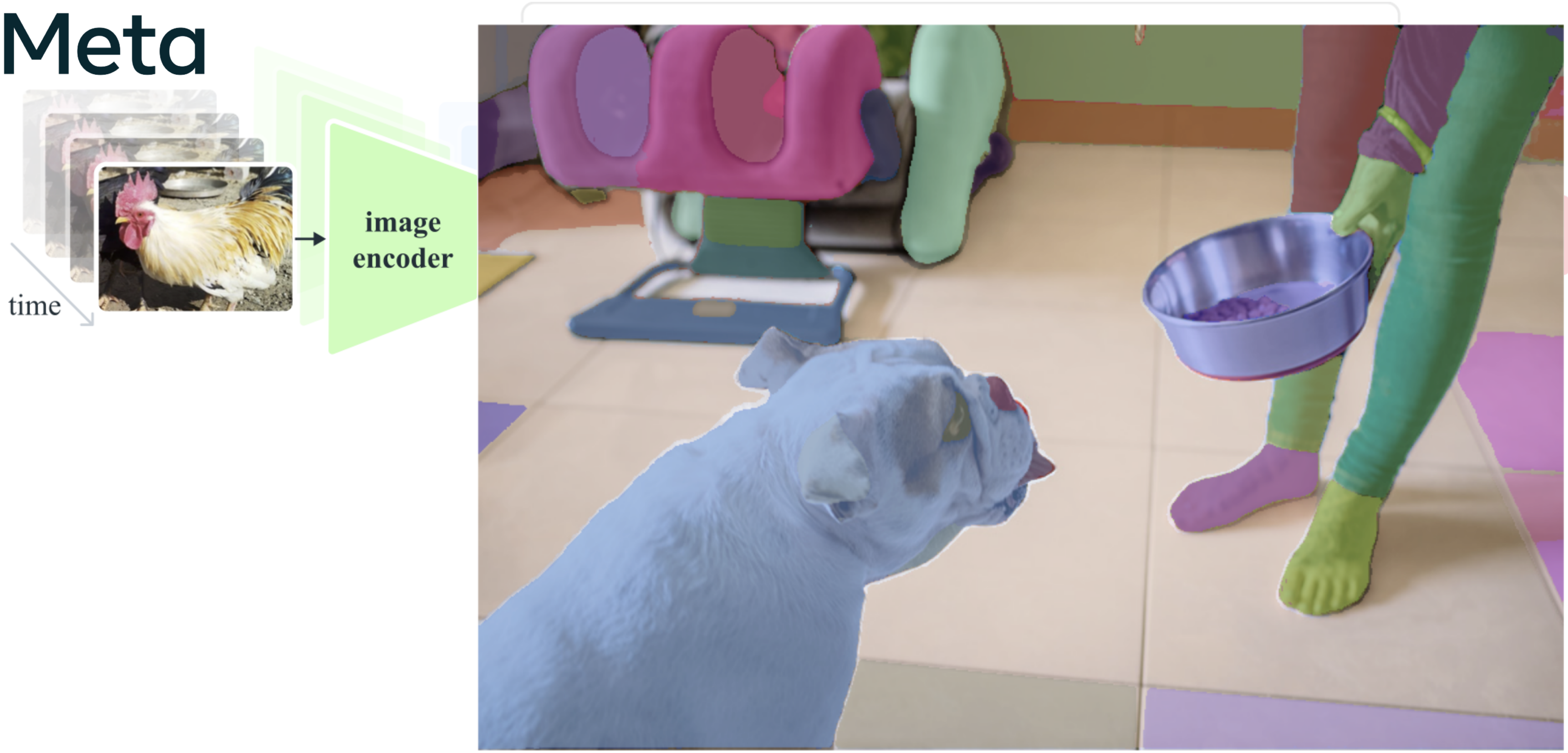
Example 301: Vision Transformers (ViTs) — SAM by Meta



https://segment-anything.com/

Case 3: Neural Networks / Deep learning

Example 301: Vision Transformers (ViTs) — SAM by Meta



https://segment-anything.com/

Case 3: Neural Networks

Example 302: Morpheus: A Deep Learning Framework For Pixel-Level Analysis of Astronomical Image Data

Hausen & Robertson: arXiv:1906.11248

Case 3: Neural Networks

Example 302: Morpheus: A Deep Learning Framework For Pixel-Level Analysis of Astronomical Image Data

Hausen & Robertson: arXiv:1906.11248

Case 3: Neural Networks

Example 302: Morpheus: A Deep Learning Framework For Pixel-Level Analysis of Astronomical Image Data

Hausen & Robertson: arXiv:1906.11248

THE ASTROPHYSICAL JOURNAL SUPPLEMENT SERIES, 248:20 (37pp), 2020 May

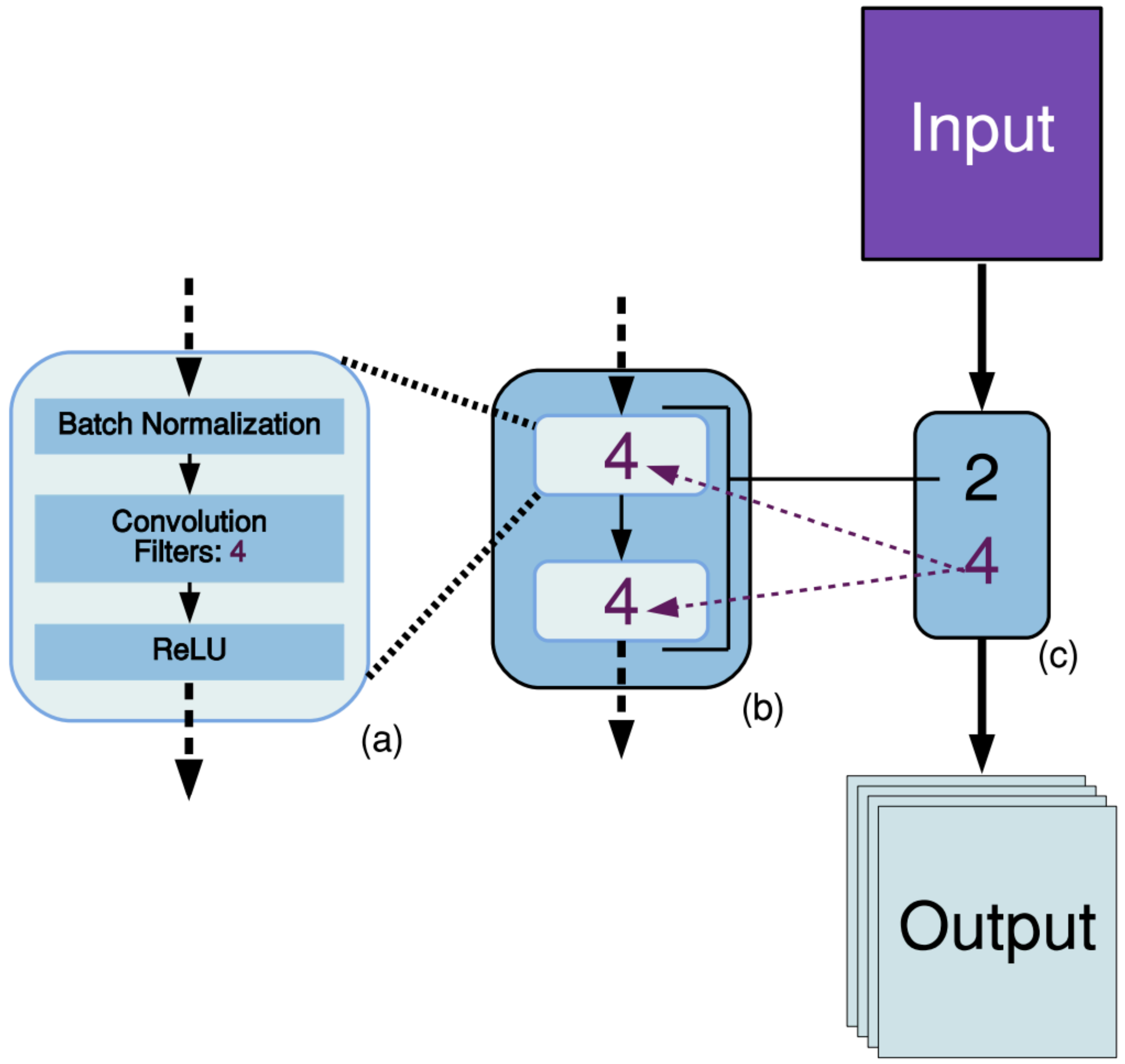


Figure 1. Diagram of a single block in the *Morpheus* neural network architecture (Figure 2). Panel (c) shows a single block from the architecture, parameterized by the number P (black) of block operations and the number Q (purple) of convolutional artificial neurons (CANs; Section A.3) in all of the convolutional layers within the block. Panel (b) shows an example zoom-in where there are $P = 2$ groups of $Q = 4$ block operations. Panel (a) shows a zoom-in on a block operation, which consists of batch normalization, $Q = 4$ CANs, and a rectified linear unit (ReLU). In the notation of Equation (1), this block operation would be written as $\mathrm{OP}_4(\boldsymbol{X})$.

Case 3: Neural Networks

Example 302: Morpheus: A Deep Learning Framework For Pixel-Level Analysis of Astronomical Image Data

Hausen & Robertson: arXiv:1906.11248

The Astrophysical Journal Supplement Series, 248:20 (37pp), 2020 May

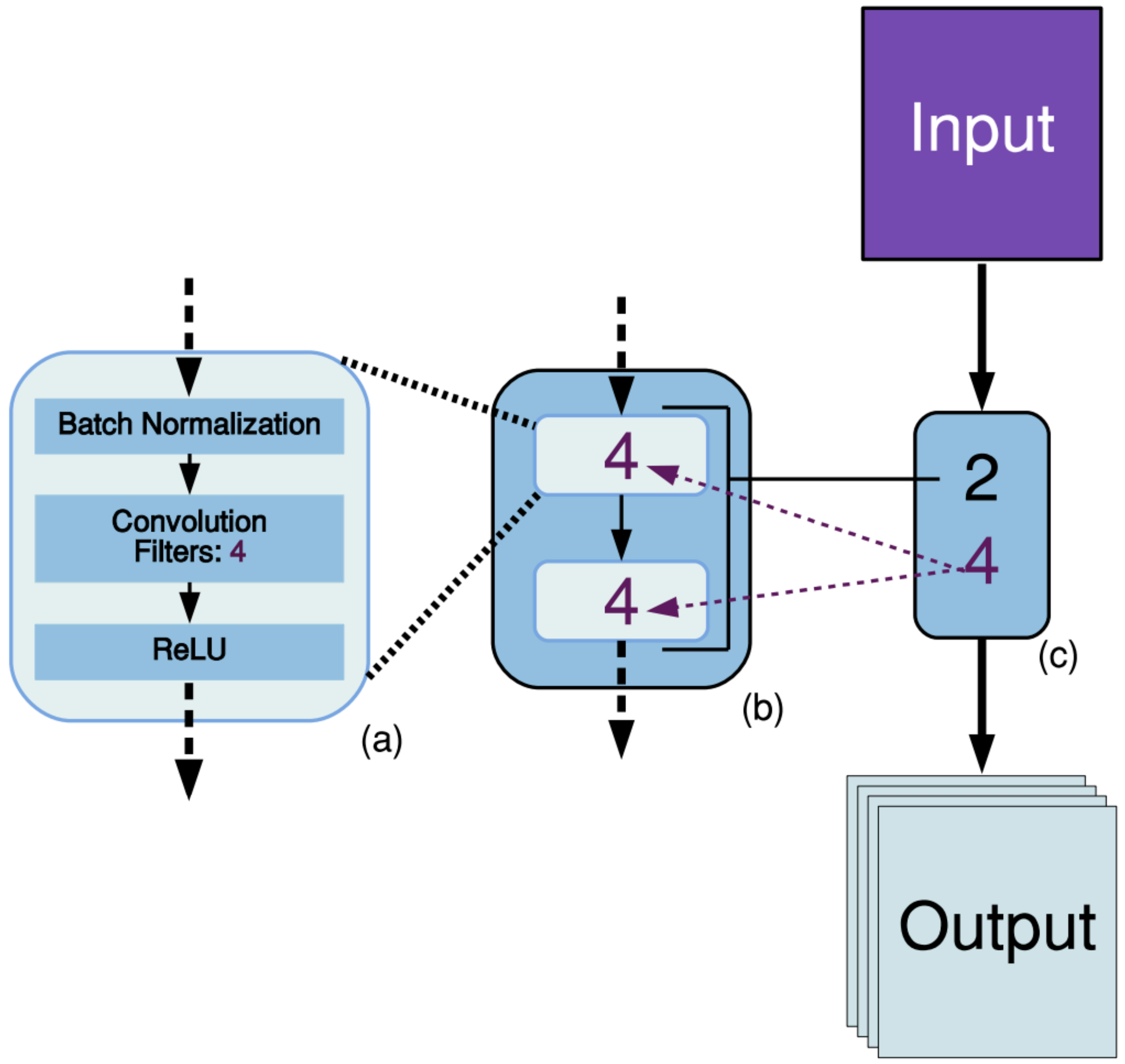
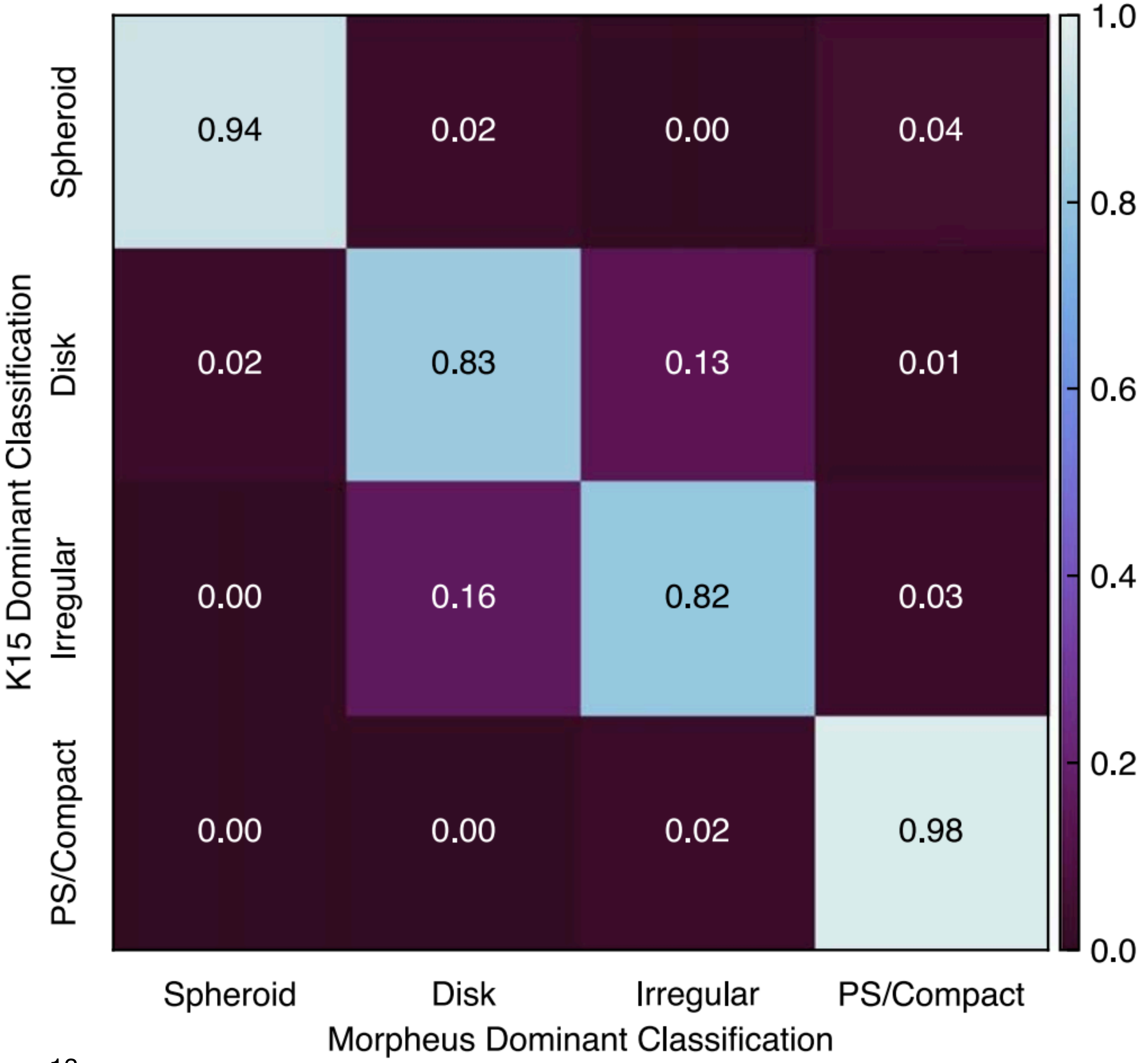


Figure 1. Diagram of a single block in the *Morpheus* neural network architecture (Figure 2). Panel (c) shows a single block from the architecture, parameterized by the number P (black) of block operations and the number Q (purple) of convolutional artificial neurons (CANs; Section A.3) in all of the convolutional layers within the block. Panel (b) shows an example zoom-in where there are $P = 2$ groups of $Q = 4$ block operations. Panel (a) shows a zoom-in on a block operation, which consists of batch normalization, $Q = 4$ CANs, and a rectified linear unit (ReLU). In the notation of Equation (1), this block operation would be written as $OP_4(X)$.



Case 3: Neural Networks

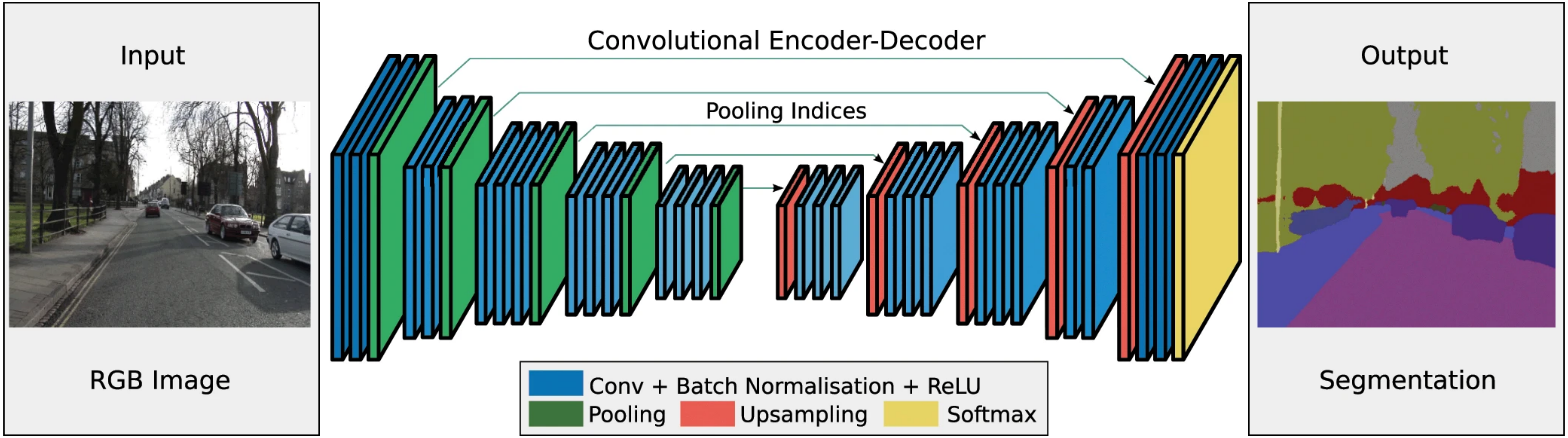
Example 302: Morpheus: A Deep Learning Framework For Pixel-Level Analysis of Astronomical Image Data

Hausen & Robertson: arXiv:1906.11248

- Code: https://github.com/morpheus-project/morpheus-core

Case 3: Neural Networks

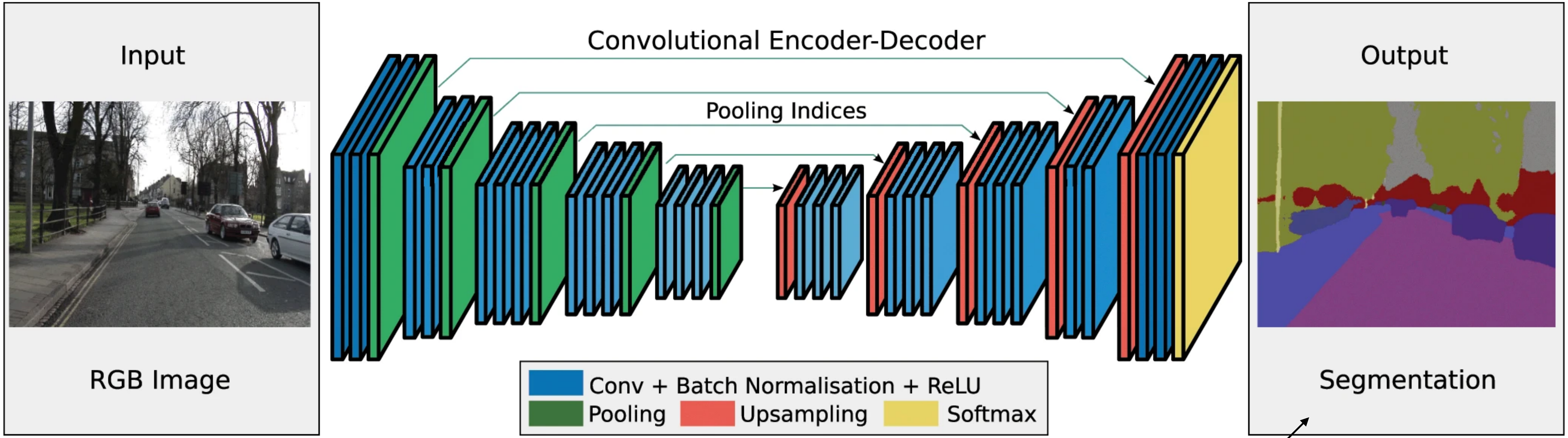
Example 303: Encoder-decoder — predict parameters directly from images



SegNet: https://doi.ieeecomputersociety.org/10.1109/TPAMI.2016.2644615

Case 3: Neural Networks

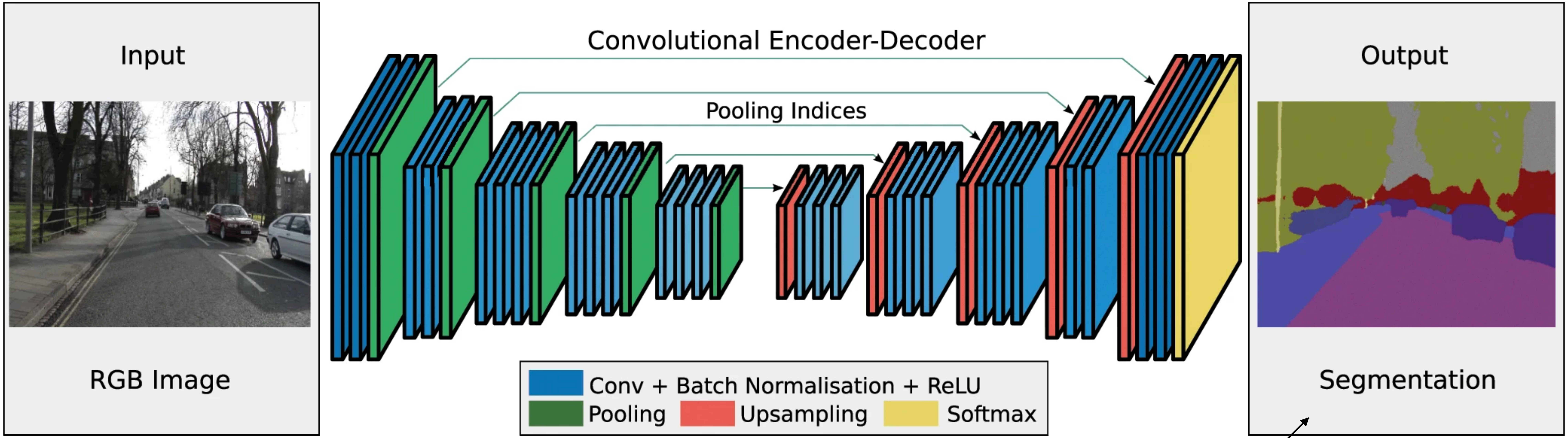
Example 303: Encoder-decoder — predict parameters directly from images



SegNet: https://doi.ieeecomputersociety.org/10.1109/TPAMI.2016.2644615

Case 3: Neural Networks

Example 303: Encoder-decoder — predict parameters directly from images



Can be changed to parameters, e.g., Sersic index, etc.

SegNet: https://doi.ieeecomputersociety.org/10.1109/TPAMI.2016.2644615

Case 3: Neural Networks

Build your own code:

Example 303: Encoder-decoder — predict parameters directly from images

```
import tensorflow as tf
from tensorflow.keras import layers, models

# Define input shape (e.g., 64x64 grayscale image of a galaxy)
input_shape = (64, 64, 1)

# Encoder part
def build_encoder(input_shape):
    inputs = layers.Input(shape=input_shape)
    x = layers.Conv2D(32, (3, 3), activation='relu', padding='same')
(inputs)
    x = layers.MaxPooling2D((2, 2))(x)
    x = layers.Conv2D(64, (3, 3), activation='relu', padding='same')(x)
    x = layers.MaxPooling2D((2, 2))(x)
    x = layers.Flatten()(x)
    encoded = layers.Dense(128, activation='relu')(x)
    return inputs, encoded

# Decoder part
def build_decoder(encoded_input):
    x = layers.Dense(16 * 16 * 64, activation='relu')(encoded_input)
    x = layers.Reshape((16, 16, 64))(x)
    x = layers.Conv2DTranspose(64, (3, 3), activation='relu',
padding='same')(x)
    x = layers.UpSampling2D((2, 2))(x)
    x = layers.Conv2DTranspose(32, (3, 3), activation='relu',
padding='same')(x)
    x = layers.UpSampling2D((2, 2))(x)
    decoded = layers.Conv2D(1, (3, 3), activation='sigmoid',
padding='same')(x)
    return decoded
```

```
# Sérsic profile prediction head
def build_prediction_head(encoded_input):
    prediction = layers.Dense(1, activation='linear',
name='sersic_index')(encoded_input)
    return prediction

# Combine the model
inputs, encoded = build_encoder(input_shape)
decoded = build_decoder(encoded)
prediction = build_prediction_head(encoded)

# Define the complete model with two outputs: reconstruction and
Sérsic index
model = models.Model(inputs=inputs, outputs=[decoded, prediction])

# Compile the model
model.compile(optimizer='adam',
              loss={'conv2d_3': 'binary_crossentropy', 'sersic_index':
'mse'},
              metrics={'sersic_index': 'mae'})

# Model summary
model.summary()

# galaxy_images = ...  # Shape: (num_samples, 64, 64, 1)
# sersic_labels = ...  # Shape: (num_samples, 1)
# model.fit(galaxy_images, {'conv2d_3': galaxy_images, 'sersic_index':
sersic_labels}, epochs=10, batch_size=32)
```

yongdaz@arizona.edu

Summary

yongdaz@arizona.edu

Summary

- Case 1 - traditional method
    - Extract features (brightness, color, axis ratio, etc.) by hand, pre-label (spiral/elliptical, SF/QG, AGN/non-AGN) by hand, and then train a model (Random Forests, Support Vector Machine (SVM), Gaussian Process, etc.)
    - The input have to be well structured
    - Pros: every step is well defined, minimal hyper-parameter tuning

yongdaz@arizona.edu

Summary

- Case 1 - traditional method
  - Extract features (brightness, color, axis ratio, etc.) by hand, pre-label (spiral/elliptical, SF/QG, AGN/non-AGN) by hand, and then train a model (Random Forests, Support Vector Machine (SVM), Gaussian Process, etc.)
  - The input have to be well structured
  - Pros: every step is well defined, minimal hyper-parameter tuning
- Case 2 - automatic feature extraction (unsupervised ML)
  - Automatic segmentation, de-blending, measurements
  - Pros: labor saving (really?) Cons: lots of debugging and hyper-parameter tuning

yongdaz@arizona.edu

Summary

- Case 1 - traditional method
  - Extract features (brightness, color, axis ratio, etc.) by hand, pre-label (spiral/elliptical, SF/QG, AGN/non-AGN) by hand, and then train a model (Random Forests, Support Vector Machine (SVM), Gaussian Process, etc.)
  - The input have to be well structured
  - Pros: every step is well defined, minimal hyper-parameter tuning
- Case 2 - automatic feature extraction (unsupervised ML)
  - Automatic segmentation, de-blending, measurements
  - Pros: labor saving (really?) Cons: lots of debugging and hyper-parameter tuning
- Case 3 - Deep Learning / Neural Network
  - ~~Magic~~