# Neuron Finding

Team Hopper
Aashish Yadavally, Caleb Crumley, Rutu Gandhi

CSCI8360 Data Science Practicum
Prof. Dr. S. Quinn

# Task: Object Finding

Major challenges of the dataset:

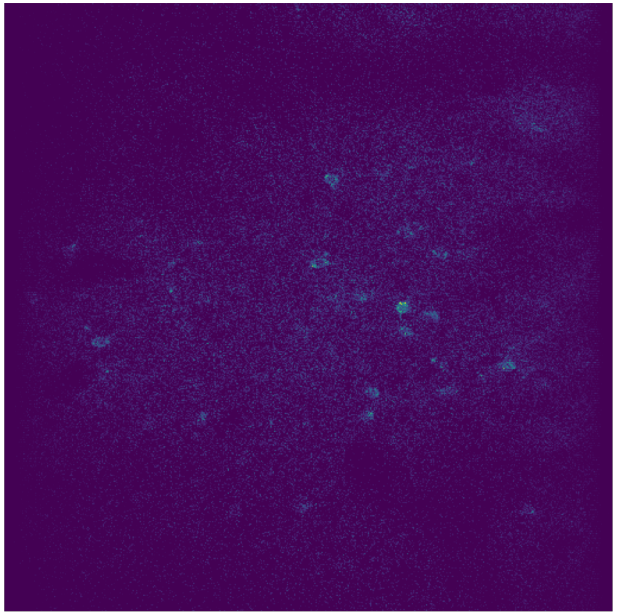
- Temporal
- Large scale
- Dark neurons

# Preprocessing

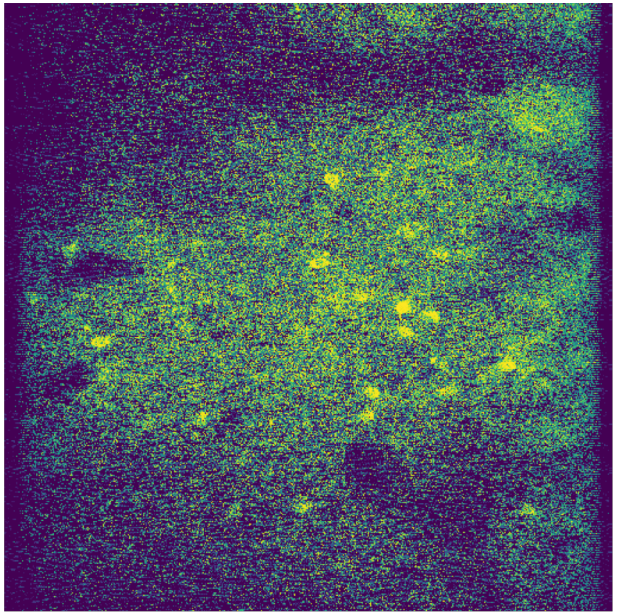
Histogram Equalization was used to bring out the contrast between darker areas and lighter areas of the image.

Creates one histogram to redistribute the pixel values.

Also used Adaptive Histogram Equalization (not pictured) that creates many histograms per different areas of the image.



Original

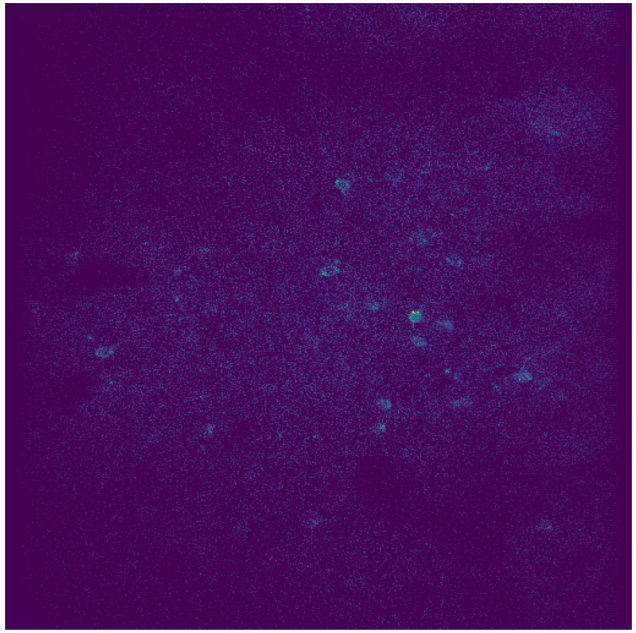


Histogram Equalization

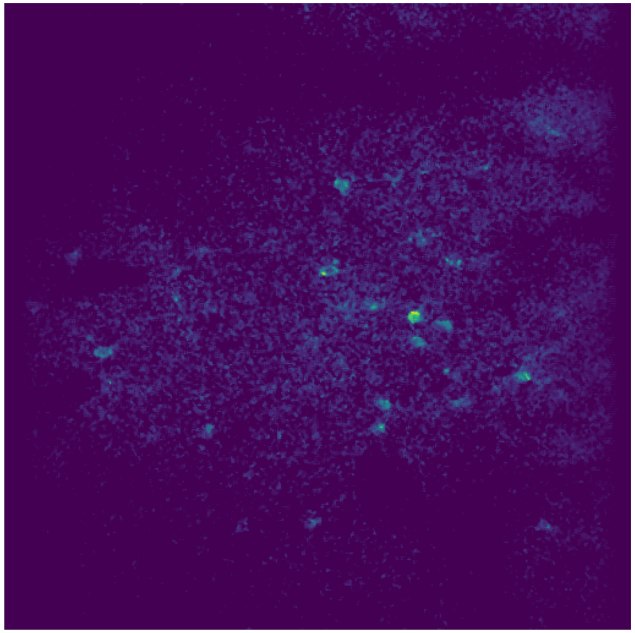
# Preprocessing

Median and Gaussian filtering to reduce the noise in the image.

The idea was it would bring out the dormant neurons more



Original

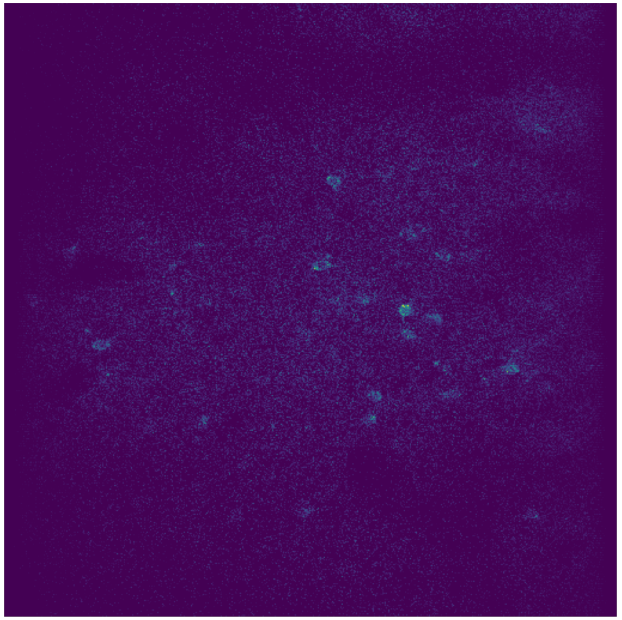


Median Filter

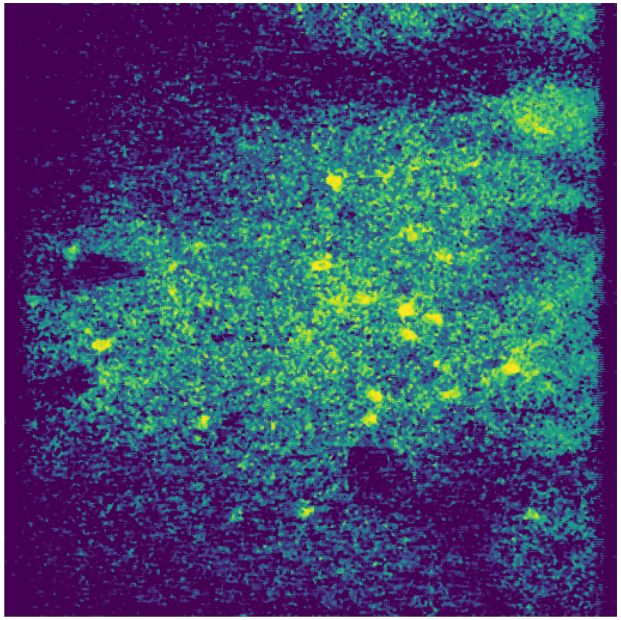
# Preprocessing

We even tried to combine the two techniques.

The idea here was that the filter would help in reducing the noise that was brought out by the histogram equalization



Original

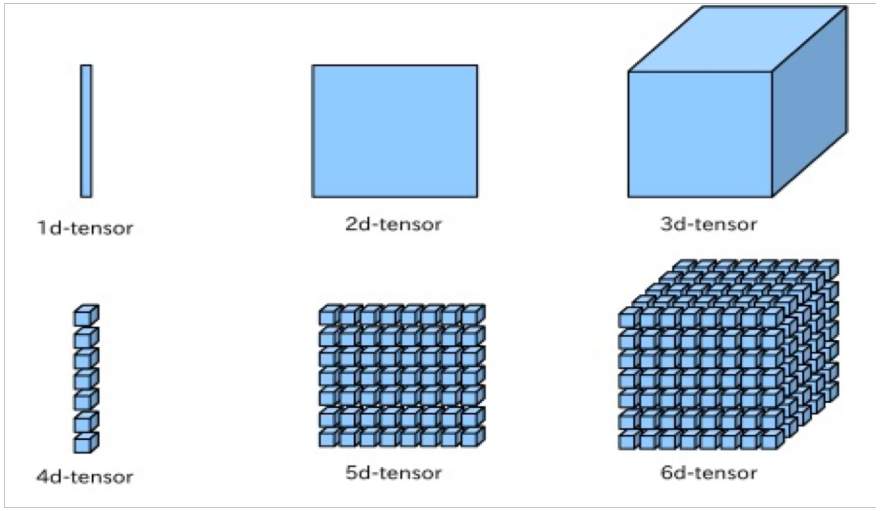


Histogram equalization and Median filtering

# UNet

1) Used the 3D version of all layers (Conv3D, UpSampling3D, MaxPooling3D)
2) Sampled frames from videos and treated it as an image segmentation problem

Input: 5D tensor with shape:



# Mask to Region conversion

Used Matlab's bwbackground for this conversion using pymatbridge

The interiors were removed using the following:

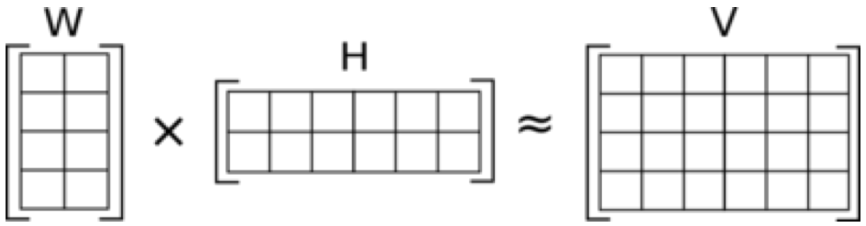
```
        """
        Removes interior pixels in neuron regions

        Arguments
        ---------
        L : 2D numpy array
                Matrix containing labels for each neuron
        region : list
                List of all pixels in a neuron label
        """
        for pixel in region:
                # Creating grid around pixel
                grid = L[pixel[0]-1:pixel[0]+2, pixel[1]-1:pixel[1]+2]
                # Removing pixels which are surrounded by similar values
                if np.unique(grid).size == 1:
                        region.remove(pixel)
        return region
```

# SparsePCA

- SparsePCA is popularly used for foreground-background separation
- We subclassed the thunder API and modified it to use this.
- A particular disadvantage of ordinary PCA is that the principal components are usually linear combinations of all input variables. Sparse PCA overcomes this disadvantage by finding linear combinations that contain just a few input variables.

# NMF

- Factorization algorithm
- The problem is not exactly solvable in general, it is commonly approximated numerically.



# Metrics

- Recall: TP/TP+FN
- Precision: TP/ TP+FP
- Inclusion: Number of intersecting pixels divided by the number of total pixels in the ground-truth regions
- Exclusion: Number of intersecting pixels divided by the number of total pixels in your regions

# NMF

- **k (components):** If this is not set, all features are kept. k number of features are used for identifying similar data. Seems to have an effect on the recall. Recall answers the question: What proportion of actual positives was identified correctly?

  More number of features = more accurate predictions

- **percentile:** thresholding ( higher the percentile lower the number of features being considered). It directly affects Inclusion and can have an effect on Exclusion in terms of number of intersecting regions.

# What we learned

- **Ethical Implications**
  - These questions are important to address before the project, and can shape how the project is setup and how the data is collected.
  - Its needed to address the alternative applications of this model and results.
  - Even though improving the accuracy of the model is the goal, checking if there exist and disparate errors is a necessity
  - Assuring that the summary statistics and metrics used that appropriate for the research project.
- **Better software engineering techniques**
  - Unit testing and test cases
  - Addressing a solid directory structure before the project takes off
- **Testing the span of the applications**