Segmenting neutron images using machine learning

Anders Kaestner

Jan 08, 2021

CONTENTS

CHAPTER

ONE

LECTURE OUTLINE

In this lecture about machine learning to segment neutron images we will cover the following topics

1. Introduction
2. Limited data problem
3. Unsupervised segmentation
4. Supervised segmentation
5. Final problem: Segmenting root networks using convolutional NNs
6. Future Machine learning challenges in NI

1.1 Importing needed modules

This lecture needs some modules to run. We import all of them here.

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import skimage.filters as flt
import matplotlib as mpl
from sklearn.cluster import KMeans
from matplotlib.colors import ListedColormap
from lecturesupport import plotsupport as ps
import pandas as pd
from sklearn.datasets import make_blobs
%matplotlib inline


from IPython.display import set_matplotlib_formats
set_matplotlib_formats('svg', 'png')
mpl.rcParams['figure.dpi'] = 150
```

```
import importlib
importlib.reload(ps);
```

CHAPTER

TWO

INTRODUCTION

- Introduction to neutron imaging
  - Some words about the method
  - Contrasts
- Introduction to segmentation
  - What is segmentation
  - Noise and SNR
- Problematic segmentation tasks
  - Intro
  - Segmenation problems in neutron imaging

2.1 What is an image?

A very abstract definition:

- **A pairing between spatial information (position)**
- **and some other kind of information (value).**

In most cases this is a two- or three-dimensional position (x,y,z coordinates) and a numeric value (intensity)

2.2 Science and Imaging

Images are great for qualitative analyses since our brains can quickly interpret them without large *programming* investements.

2.2.1 Proper processing and quantitative analysis is however much more difficult with images.

- If you measure a temperature, quantitative analysis is easy, $50K$.
- If you measure an image it is much more difficult and much more prone to mistakes, subtle setup variations, and confusing analyses

2.2.2 Furthermore in image processing there is a plethora of tools available

- Thousands of algorithms available
- Thousands of tools
- Many images require multi-step processing
- Experimenting is time-consuming

2.3 Some word about neutron imaging

Neutron imaging is a method based on transmission of the neutron radiation through a sample, i.e. the fundamental information is a radiograph. In this sense it is very much similar to the more known x-ray imaging. The intensity in a radiographic image proportional to the amount of radiation that remains after it was transmitted through the sample. The transmitted radiation is described by Beer-Lambert's law which in its basic form look like

$I = I_0 \cdot e^{-\int_L \mu(x)dx}$

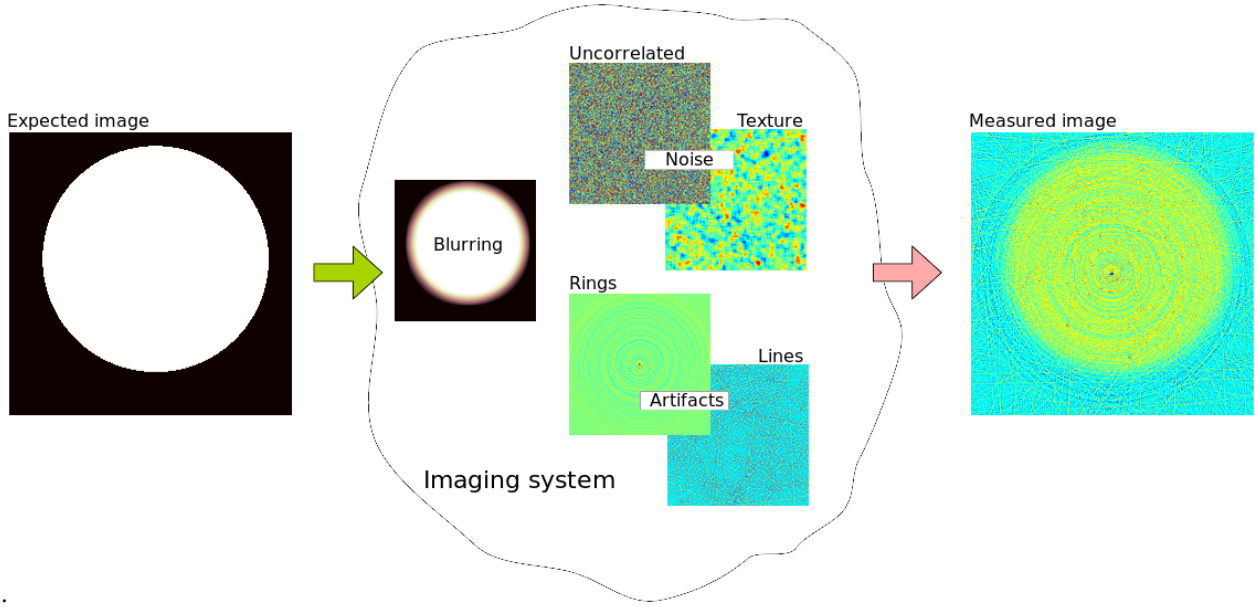
Where $\mu(x)$ is the attenuation coefficient of the sample at position x.

Single radiographs are relatively rare. In most experiments the radiographs are part of a time series aquisition or projections for the 3D tomography reconstuction. In this lecture we are not going very much into the details about the imaging method as such. This is a topic for other schools that are offered.

2.4 Neutron imaging contrast

2.5 Measurements are rarely perfect

There is no perfect measurement. This is also true for neutron imaging. The ideal image is the sample is distorted for many reasons. The figure below shows how an image of a sample can look after passing though the acquisition system. These quailty degradations will have an impact on the analysis of the image data. In some cases, it is possible to correct for some of these artifacts using classing image processing techniques. There are however also cases that require extra effort to correct the



artifacts.

2.5.1 Factors affecting the image quality

The list below provides some factors that affect the quality of the acquired images. Most of them can be handled by changing the imaging configuration in some sense. It can however be that the sample or process observed put limitiations on how much the acquisition can be tuned to obtain the perfect image.

- Resolution (Imaging system transfer functions)
- Noise
- Contrast
- Inhomogeneous contrast
- Artifacts

Resolution

The resolution is primarily determined optical transfer function of the imaging system. The actual resolution is given by the extents of the sample and how much the detector needs to capture in one image. This gives the field of view and given the number pixels in the used detector it is possible to calculate the pixel size. The pixel size limits the size of the smallest feature in the image that can be detected. The scintillator, which is used to convert neutrons into visible light, is chosen to

1. match the sampling rate given by the pixel size.
2. provide sufficient neutron capture to obtain sufficient light output for a given exposure time.

Noise

An imaging system has many noise sources, each with its own distribution e.g.

1. Neutron statistics - how many neutrons are collected in a pixel. This noise is Poisson distributed.
2. Photon statistics - how many photons are produced by each neutron. This noise is also Poisson distributed.
3. Thermal noise from the electronics which has a Gaussian distribution.
4. Digitation noise from converting the charges collected for the photons into digital numbers that can be transfered and stored by a computer, this noise has a binominal distribution.

The neutron statistics are mostly dominant in neutron imaging but in some cases it could also be that the photon statistics play a role.

Contrast

The contrast in the sample is a consequence of

1. how well the sample transmits the chosen radiation type. For neutrons you obtain good contrast from materials containing hydrogen or lithium while many metals are more transparent.
2. the amount of a specific element or material represented in a unit cell, e.g. a pixel (radiograph) or a voxel (tomography).

The objective of many experiments is to quantify the amount of a specific material. This could for example be the amount of water in a porous medium.

Good contrast between different image features is important if you want to segment them to make conclusions about the image content. Therefore, the radiation type should be chosen to provide the best contrast between the features.

Inhomogeneous contrast

The contrast in the raw radiograph depends much on the beam profile. These variations are easily corrected by normalizing the images by an open beam or flat field image.

- **Biases introduced by scattering** Scattering is the dominant interaction for many materials use in neutron imaging. This means that neutrons that are not passing straight though the sample are scattered and contribute to a background cloud of neutrons that build up a bias of neutron that are also detected and contribute to the
- **Biases from beam hardening** is a problem that is more present in x-ray imaging and is caused by that fact that the attenuation coefficient depends on the energy of the radiation. Higher energies have lower attenuation coefficient, thus will high energies penetrate the thicker samples than lower energies. This can be seen when a polychromatic beam is used.

Artifacts

Many images suffer from outliers caused by stray rays hitting the detector. Typical artefacts in tomography data are

- Lines, which are caused by outlier spots that only appear in single projections. These spot appear as lines in the reconstructed images.
- Rings are caused by stuck pixels which have the same value in a projections.

2.6 Introduction to segmentation

What is segmentation

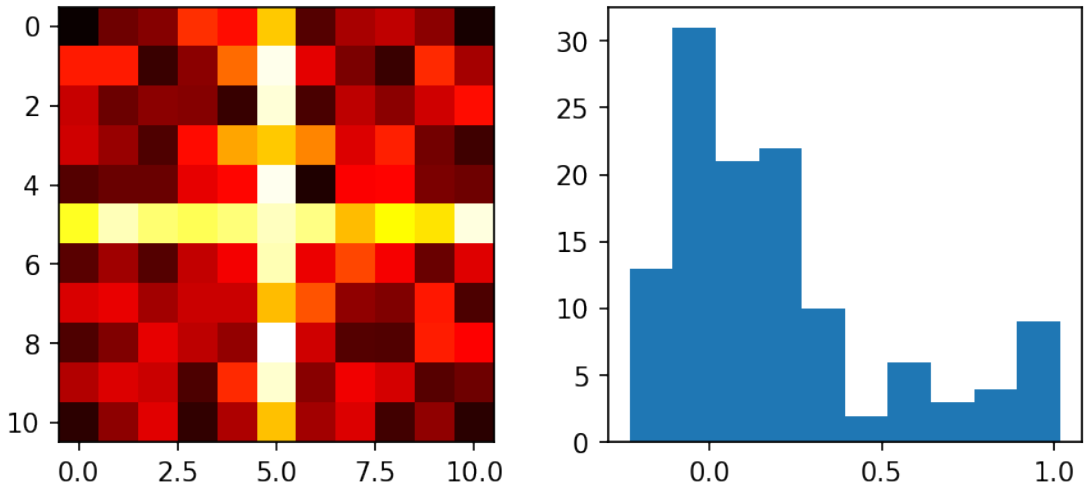
2.6.1 Basic segmentation: Applying a threshold to an image

Start out with a simple image of a cross with added noise

$$I(x, y) = f(x, y)$$

Here, we create a test image with two features embedded in uniform noise; a cross with values in the order of '1' and background with values in the order '0'. The figure below shows the image and its histogram. The histogram helps us to see how the graylevels are distributed which helps to decide where to put a threshold that segments the cross from the background.

```
fig,ax = plt.subplots(1,2,figsize=(7,3))
nx = 5; ny = 5;
xx, yy = np.meshgrid(np.arange(-nx, nx+1)/nx*2*np.pi,
                     np.arange(-ny, ny+1)/ny*2*np.pi)
cross_im =   1.5*np.abs(np.cos(xx*yy))/(np.abs(xx*yy)+(3*np.pi/nx)) + np.random.
↪uniform(-0.25, 0.25, size = xx.shape)
im=ax[0].imshow(cross_im, cmap = 'hot');
ax[1].hist(cross_im.ravel(),bins=10);
```



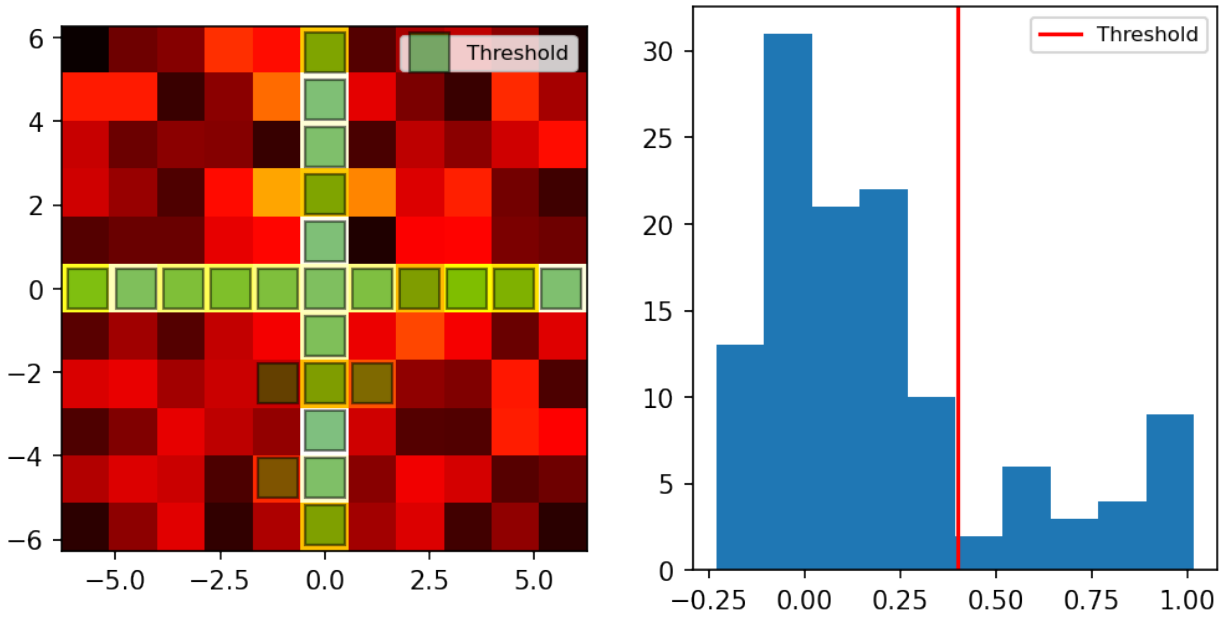
2.6.2 Applying a threshold to an image

By examining the image and probability distribution function, we can *deduce* that the underyling model is a whitish phase that makes up the cross and the darkish background

Applying the threshold is a deceptively simple operation

$$I(x, y) = \begin{cases} 1, & f(x, y) \geq 0.40 \\ 0, & f(x, y) < 0.40 \end{cases}$$

```
threshold = 0.4; thresh_img = cross_im > threshold

fig,ax = plt.subplots(1,2,figsize=(8,4))
ax[0].imshow(cross_im, cmap = 'hot', extent = [xx.min(), xx.max(), yy.min(), yy.
↪max()])
ax[0].plot(xx[np.where(thresh_img)]*0.9, yy[np.where(thresh_img)]*0.9,
           'ks', markerfacecolor = 'green', alpha = 0.5,label = 'Threshold',␣
↪markersize = 15); ax[0].legend(fontsize=8);
ax[1].hist(cross_im.ravel(),bins=10); ax[1].axvline(x=threshold,color='r',label=
↪'Threshold'); ax[1].legend(fontsize=8);
```



2.7 Noise and SNR

2.8 Problematic segmentation tasks

Intro

2.9 Segmenation problems in neutron imaging

CHAPTER

THREE

LIMITED DATA PROBLEM

3.1 Training data from NI is limited

3.2 Augmentation

3.3 Transfer learning

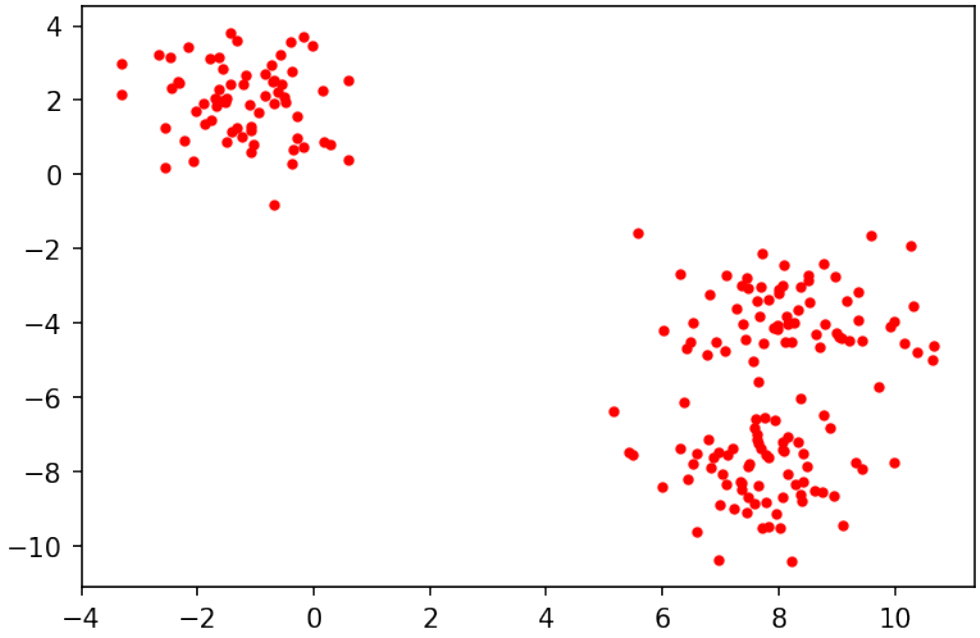
CHAPTER

FOUR

UNSUPERVISED SEGMENTATION

4.1 Introducing clustering

```
test_pts = pd.DataFrame(make_blobs(n_samples=200, random_state=2018)[
                        0], columns=['x', 'y'])
plt.plot(test_pts.x, test_pts.y, 'r.');
```



4.2 k-means

The k-means algorithm is one of the most used unsupervised clustering method. The user only have to provide the number of classes the algorithm shall find. **Note: If you look for N groups you will almost always find N groups with K-Means, whether or not they make any sense**

It is an iterative method that starts with a label image where each pixel has a random label assignment. Each iteration involves the following steps:

1. Compute current centroids based on the o current labels
2. Compute the value distance for each pixel to the each centroid. Select the class which is closest.

3. Reassign the class value in the label image.
4. Repeat until no pixels are updated.

The distance from pixel *i* to centroid *j* is usually computed as $||p_i - c_j||_2$.

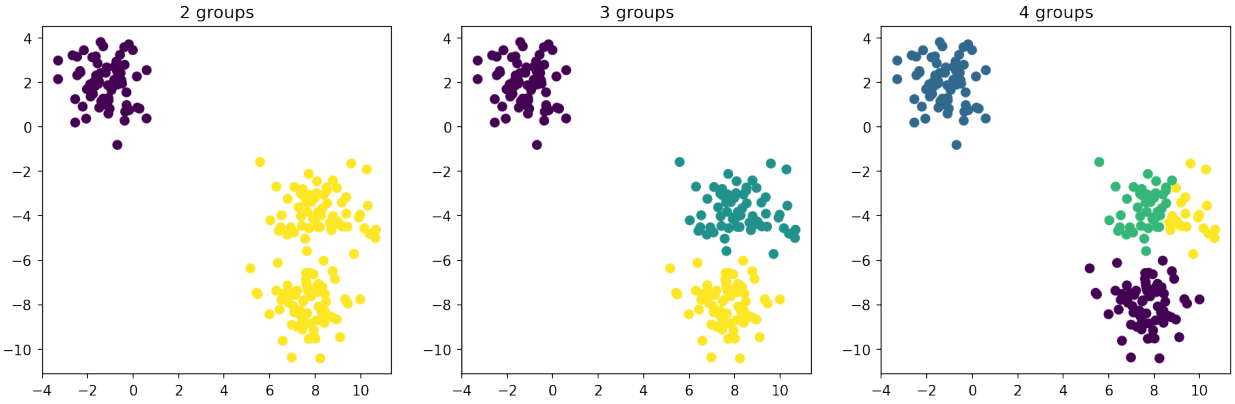
It is important to note that k-means by definition is not position sensitive. The position can however be included as additional components in the data vectors.

k-means makes most sense to use on vector valued images where each pixel is represented by several values, e.g.:

1. Images from multimodal experiments like combined neutron and X-ray.
2. Wavelength resolved imaging

4.3 Basic clustering example

```
fig, ax = plt.subplots(1,3,figsize=(15,4.5))

for i in range(3) :
    km = KMeans(n_clusters=i+2, random_state=2018); n_grp = km.fit_predict(test_pts)
    ax[i].scatter(test_pts.x, test_pts.y, c=n_grp)
    ax[i].set_title('{0} groups'.format(i+2))
```



4.4 Clustering applied to wavelength resolved imaging

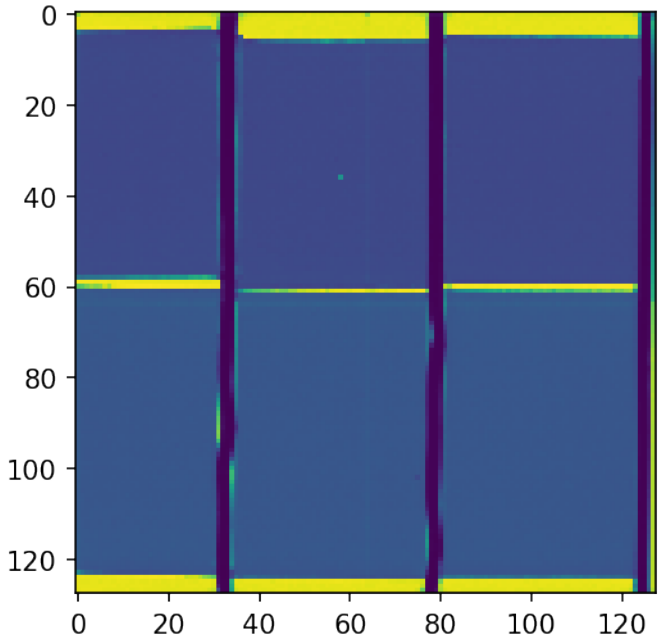
4.4.1 The imaging techniques and its applications

4.4.2 The data

In this example we use a time of flight transmission spectrum from an assembly of six rectangular steel blocks produced by means of additive manufactoring. The typical approach to analyse this kind of information is to select some regions of interest and compute the average spectra from these regions. The average spectra are then analyzed using single Bragg edge fitting methods or Rietveld refinement.

Here, we will explore the posibility to identify regions with similar spectra using k-means clustering. The data is provided on the repository and has the dimensions 128x128x661, where the first to dimensions reflect the image size and the last is th number of time bins of the time of flight spectrum. This is a downsized version of the original data which has 512x512 image frames. Each frame has a relatively low signal to noise ratio, therefore we use the average image *wtof* to show the image contents.

```
tof  = np.load('../data/tofdata.npy')
wtof = tof.mean(axis=2)
plt.imshow(wtof);
```



Reshaping

The k-means requires vectors for each feature dimension, not images as we have in this data set. Therefore, we need to reshape our data. The reshaping is best done using the numpy array method `reshape` like this:

```
tofr=tof.reshape([tof.shape[0]*tof.shape[1],tof.shape[2]])
tofr.shape
```

```
(16384, 661)
```

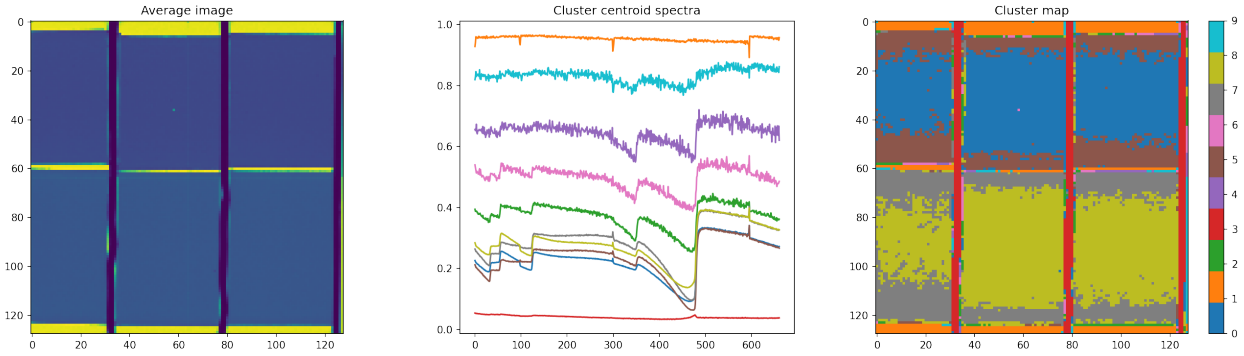
The reshaped data *tofr* now has the dimensions 16384x661, i.e. each 128x128 pixel image has been replaced by a vector with the length 16384 elements. The number of time bins still remains the same. The `reshape`method is a cheap operation as it only changes the elements of the dimension vector and doesn't rearrange the data in memory.

4.4.3 Setting up and running k-means

```
km = KMeans(n_clusters=10, random_state=2018)
c  = km.fit_predict(tofr).reshape(tof.shape[:2]) # Label image
kc = km.cluster_centers_.transpose()             # cluster centroid spectra
```

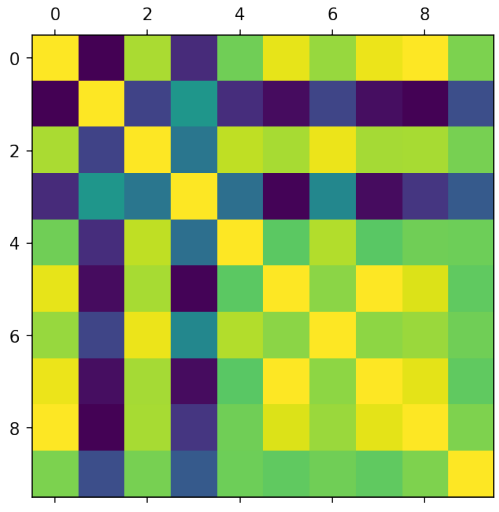
4.4.4 Results

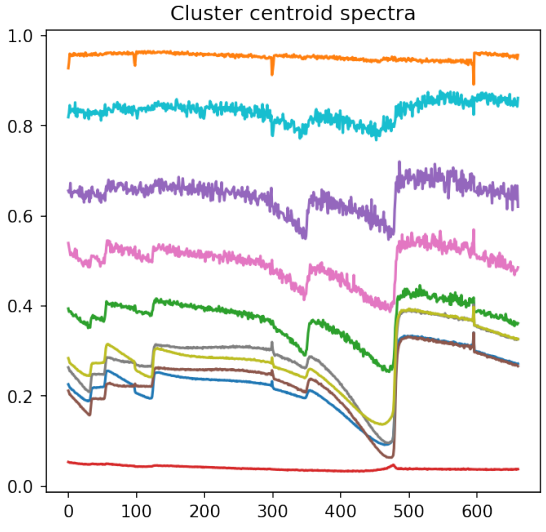
```
fig,axes = plt.subplots(1,3,figsize=(18,5)); axes=axes.ravel()
axes[0].imshow(wtof,cmap='viridis'); axes[0].set_title('Average image')
p=axes[1].plot(kc);                  axes[1].set_title('Cluster centroid spectra');␣
↪axes[1].set_aspect(tof.shape[2], adjustable='box')
cmap=ps.buildCMap(p) # Create a color map with the same colors as the plot

im=axes[2].imshow(c,cmap=cmap); plt.colorbar(im);
axes[2].set_title('Cluster map');
plt.tight_layout()
```



How similar are the classes?

```
fig,axes = plt.subplots(1,2,figsize=(14,5)); axes=axes.ravel()
axes[0].matshow(np.corrcoef(kc.transpose()))
axes[1].plot(kc); axes[1].set_title('Cluster centroid spectra'); axes[1].set_
↪aspect(tof.shape[2], adjustable='box')
```





CHAPTER

FIVE

SUPERVISED SEGMENTATION

- e.g. k-NN, decision trees
- NNs for segmentation

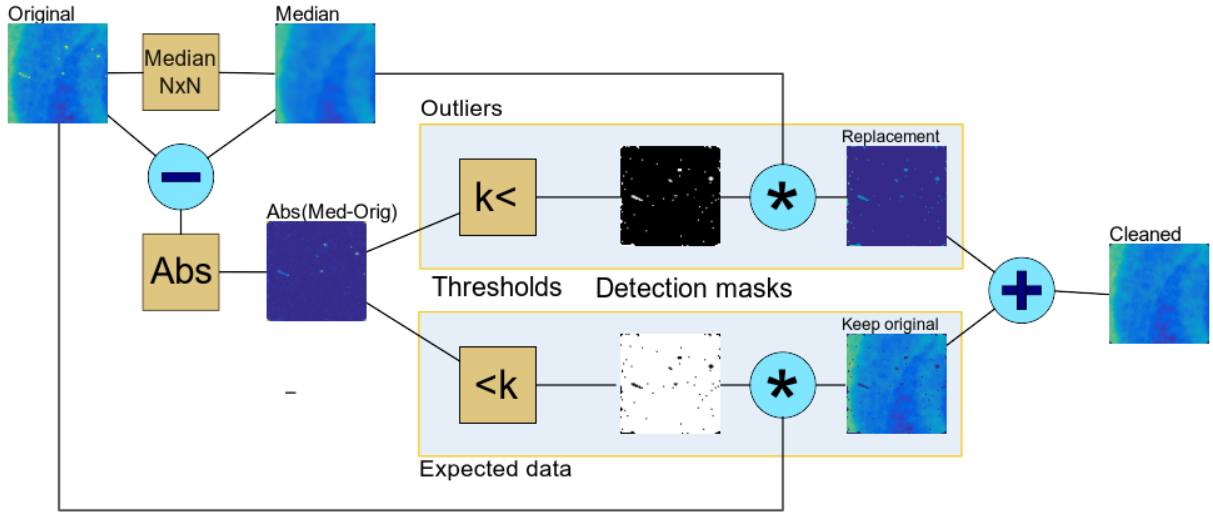
5.1 Example - Detecting and correcting unwanted outliers (a.k.a. spots) in neutron images

5.1.1 Training data

We have two choices:

1. Use real data
    - requires time consuming markup to provide training data
    - corresponds to real life images
2. Synthesize data
    - flexible and provides both 'dirty' data and ground truth.
    - model may not behave as real data

5.1.2 Baseline - Traditional spot cleaning algorithm



Parameters

- *N* Width of median filter.
- *k* Threshold level for outlier detection.

5.1.3 The spot cleaning algorithm

The baseline algorithm is here implemented as a python function that we will use when we compare the performance of the CNN algorithm. This is the most trivial algorithm for spot cleaning and there are plenty other algorithms to solve this task.

```
def spotCleaner(img, threshold=0.95, wsize=3) :
    mimg = flt.median(img,size=(wsize,wsize))
    timg = np.abs(img-mimg) < threshold

    cleaned = img * timg + mimg * (1-timg)

    return cleaned
```

5.1.4 Build a CNN for spot detection and cleaning

5.1.5 Performance evaluation

Any analysis system must be verified

For this we need to split our data into three categories:

1. Training data
2. Test data
3. Validation data

Compare results using ROC curve

CHAPTER

SIX

FINAL PROBLEM: SEGMENTING ROOT NETWORKS IN THE RHIZOSPHERE USING CONVOLUTIONAL NNS

- Problem definition
- NN model
- Loss functions
- Training
- Results

CHAPTER

SEVEN

FUTURE MACHINE LEARNING CHALLENGES IN NEUTRON IMAGING