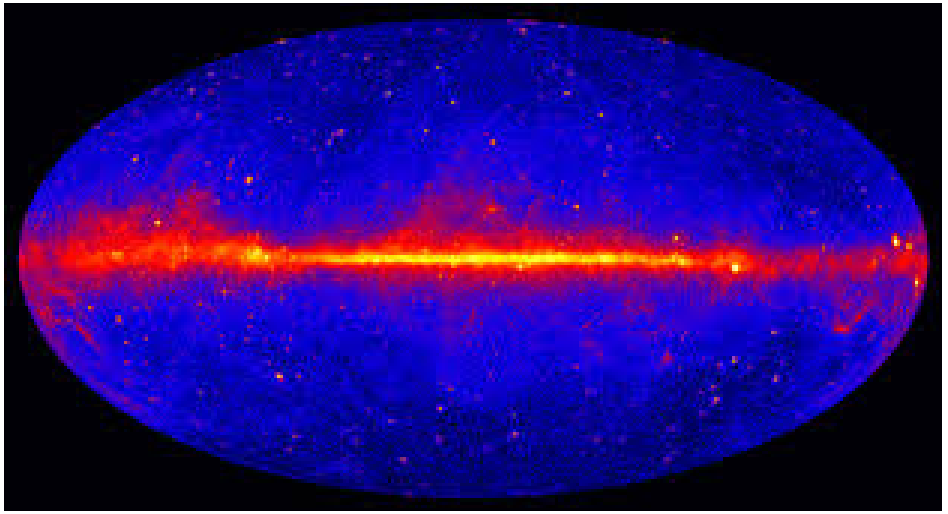
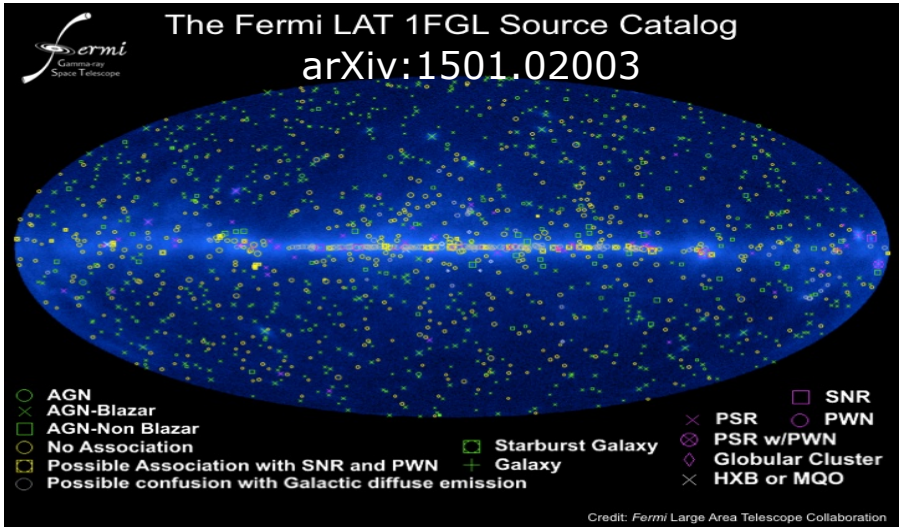
Using machine learning to localize Fermi-LAT point sources: Hypothesis algorithm based on UNET times k-means

Boris Panes, PUC and Dark Machines

September 18, 2019

Definition of the problem: 1/2





Fermi-LAT Observed Gamma-ray Sky in the region 10 GeV - 100 TeV

- **Usual shape of the input tensor:**

(400,800,5) => (lat, lon, ene-bin)

Such that in each pixel and energy bin we know the value of the **gamma ray flux**

$$\text{Input} \in \mathbb{R}^{400\times800\times5}$$

The energy division of 5 bins has been choosen in order to enhance spectral differences between sources

Solid angle coverage: 180ºx360°

Machine learning models in general, preferentially Neural Networks

Fermi-LAT like catalog of point sources. Initially, the type of source is not required

- **Possible shape of the output:**

(N, 2) => (source-index, (lat, lon))

We can see that this ouput contains the information about **point source positions** and the **number of sources** in the catalogue

$$\text{Output} \in \mathbb{R}^{\text{N}\times2}$$

For classification we could add a new dimension to this output

Solid angle coverage: 180ºx360°

Definition of the problem: 2/2

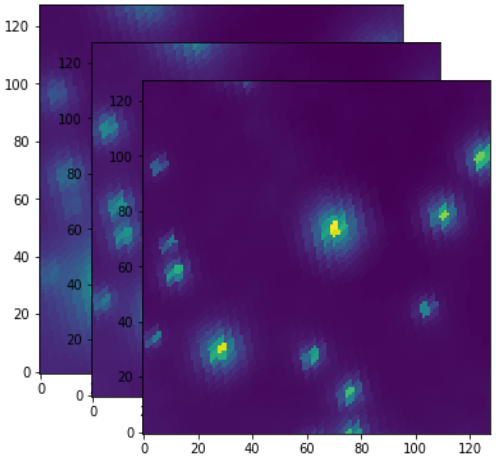
In practice, we have decided to solve the problem by considering a couple of assumptions: instead of real images we use simulations, we are going to use smaller images and we consider supervised machine learning algorithms (ML) as part of the solutions. Other relevant aspects of the pipeline are:

- **Input image**
  - Instead of one real full sky image (Fermi-LAT observation) we are going to consider multiple simulations, which are generated as indicated in the Google doc of the challenge
  - We divide each full image in smaller patches of shape (128,128,5)
  - Each of these patches cover a part of the sphere of 10°x10°
  - Since the full sphere cover approximately 40000 [deg^2], we can find approximately 400-500 independent patches for each image
  - Notice that each patch contains better resolution than the full image, approximately 1 pixel/0.1°

- **Machine learning algorithm**
  - Since we have simulations available we are able to consider a semi-supervised approach
  - In order to prepare the labels of the input images we need to specify the ML algorithms to consider
  - First, we use a fully convolutional neural network called **UNET** (arXiv:1505.04597) in order to transform the original image (128,128,5) into one segmented image of dimension (128,128,2) where each pixel contains the probability of having a point source or background
  - Therefore, the labels that we need to generate are masked versions of the simulated images
  - Secondly, we use an algorithm based in **k-means** to process the first layer of the UNET ouput in order to obtain as output the best list of centroids

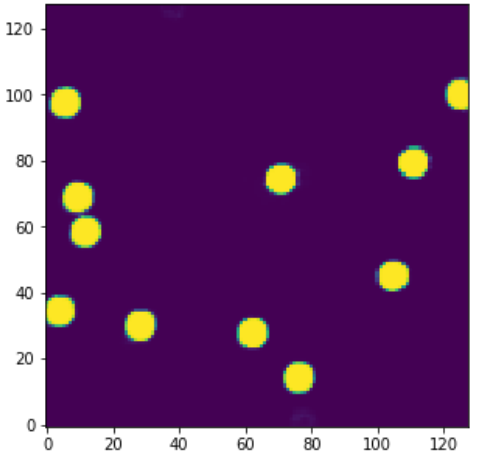
- **Outputs and evaluation**
  - The ouput of the algorithm includes the prediction of the number of sources and a list of positions
  - With this output we can build several plots and metrics to evaluate the performance of the algorithm
  - For this presentation, I only have prepared an evaluation concerning the prediction of the number of sources

Working solution pipeline: full view, prediction



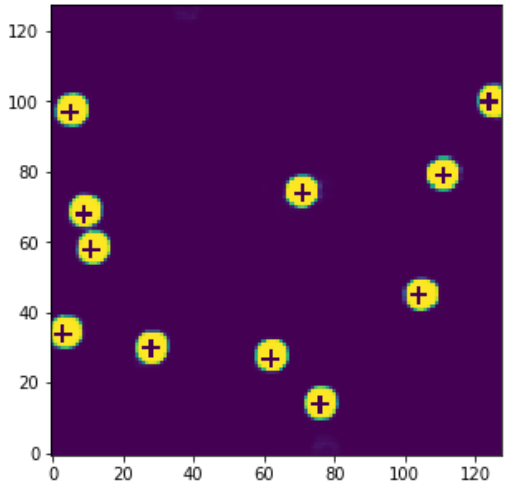
UNET input
(128,128,5)

UNET
arXiv:1505.04597



UNET output
(128,128,2)

Algorithm output
List of positions (crosses) on top of (128,128,2)



k-means optimized

Github code: **https://github.com/bapanes/Automatic-Sourcer**

Data preparation for the UNET

- Let us assume that we start with 1000 mock catalogs. Each simulation contain a .fits file that basically contain the gamma-ray sky images of background, agn and pulsars separated. Also they include a .xml file with the information about the localization of point sources in (lon,lat) coordinates

mock catalogs

files .fits and .xml

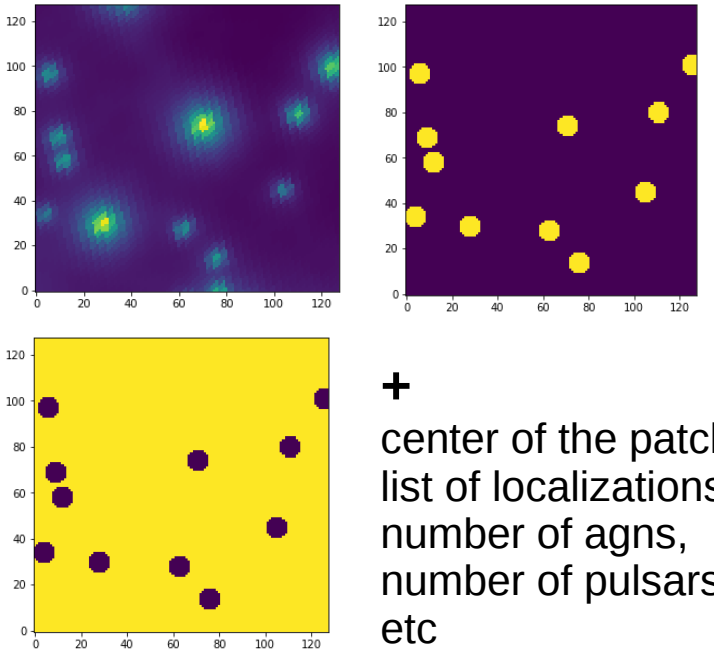
separation between agn, pulsar and background

1000 of them with fixed background

Rob and Chris algs.

patch-mask generator

- (128,128,5) patches
- (128,128,2) masked images with disks of radius 5 (arbitrary)
- true localizations in local Cartesian coordinates
- For the generation of patches I used Chris algorithm



+

center of the patch, list of localizations number of agns, number of pulsars, etc

code: from-mock-cats-to-h5-folder.py

- With this procedure we are able to generate 5 samples per catalog, each one containing the information of 100 patches. In total we have generated close to 200K patches in this way

- However, in order to train the UNET and avoid memory problems we have to merge these samples in only two big data sets, one for training and other for validation, each one containing 80% and 20% of the total amount of patches respectively

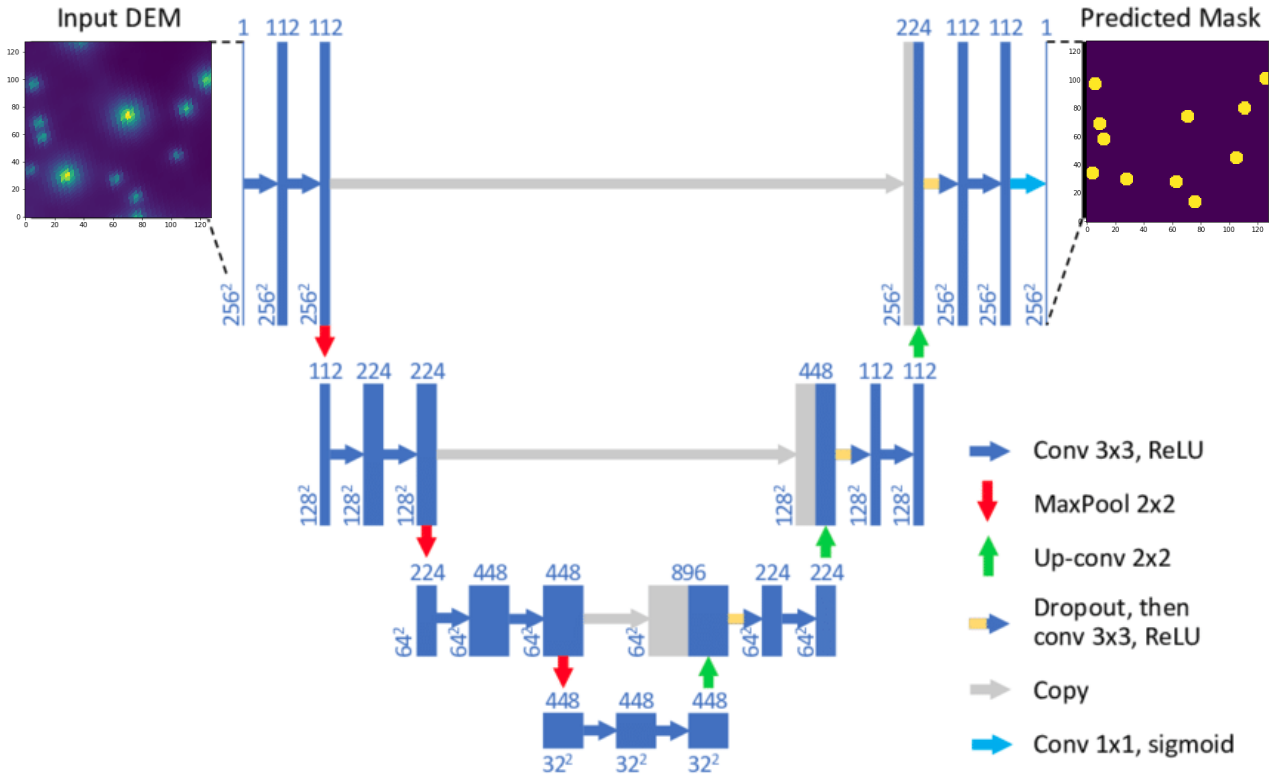
code: from-h5-folder-to-h5-dataset-full-info.py

UNET training

The UNET is a fully convolutional neural network that receives as input an image of shape (H,W,B) and is able to learn what is the pixel-by-pixel probability of the classes of objects in the input image

The first part includes a sequence of blocks that contain 2D convolutional layers and a max-pooling layer, this part is called encoder. Afterwards, the data is up-sampled by using a sequence of blocks that contain transposed 2D convolutional layers and concatenation layers that recover information from the first half of the net, this part is called decoder.

The output of the net has an image shape, where the height and wide of the image coincides with the input one and the depth represents the number of classes to be looked for



The UNET algorithm is able to work with different in-out shapes

We have trained the UNET with 50K and 100K patches

The final activation function is softmax (pixel by pixel)

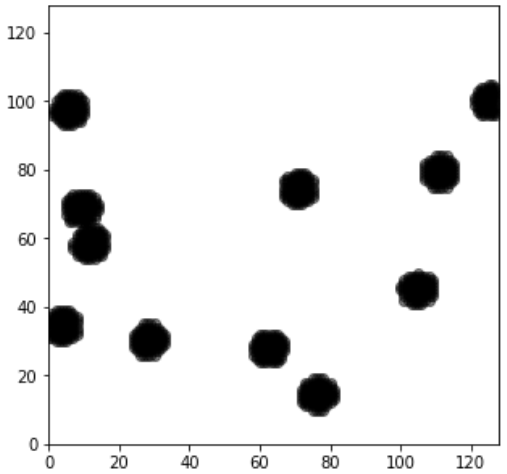
The loss function is defined as binary-cross-entropy

For this amount of images, the metric accuracy reports 0.9953
16302 of 16384 pixels ~ 1 PS

code: from-h5-dataset-to-unet-model.py

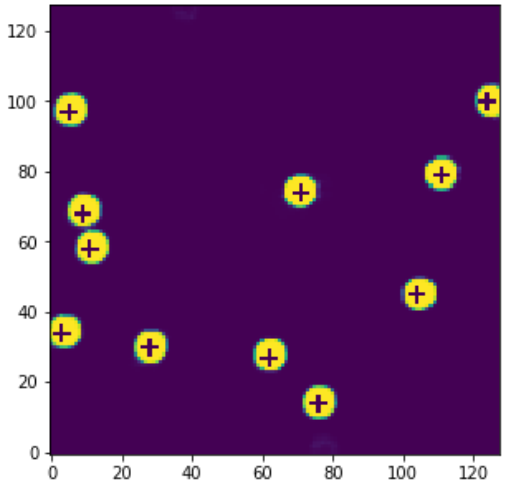
k-means optimized

Now, we take the first layer of the UNET output and generate a set of all the 2D positions that have a probability above 0.2 (arbitrary). This is the simplest input that the algorithm k-means can take in order to find the centroids of the observed clusters



(standard) k-means

1. generate k-centroids randomly
2. assign the observed 2D vectors to the closest centroid
3. update the value of the k-centroids using the mean of the vectors around each centroid
4. go to 2 or stop when centroids are stable



However, in order to get the positions of the centroids, the k-means algorithm must know the number k of clusters to look for. This is in principle a problem since we want to predict the number of sources

So, given the nice structure of UNET outputs we have implemented an algorithm based in k-means such that the final answer includes the number of sources (Luc's idea). The algorithm is as follows

for k in range(1,50):

- run k-means for k clusters

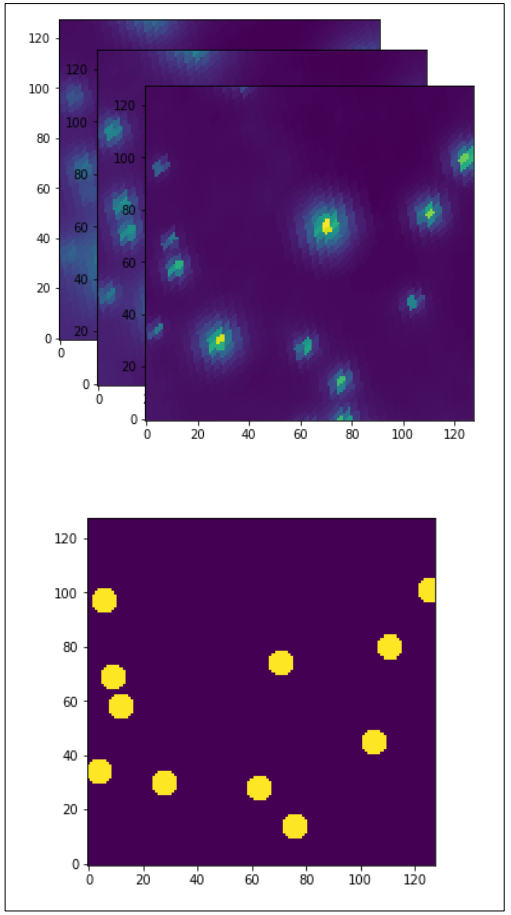
  for c in centroids:

  - compute the intensity of the pixels in the circle around c, which corresponds to the sum of the UNET probability in these pixels (we consider a radius = 3, which is arbitrary)
  - update total intensity associated to k centroids
  - define the probability of the pixels around c as -10 (arbitraty penalization)

generate as output the list of centroids for the value of k that maximizes the total intensity

code: unet-plus-optimized-kmeans-localization-algorithm-evaluation.ipynb

Pre-evaluation algorithm



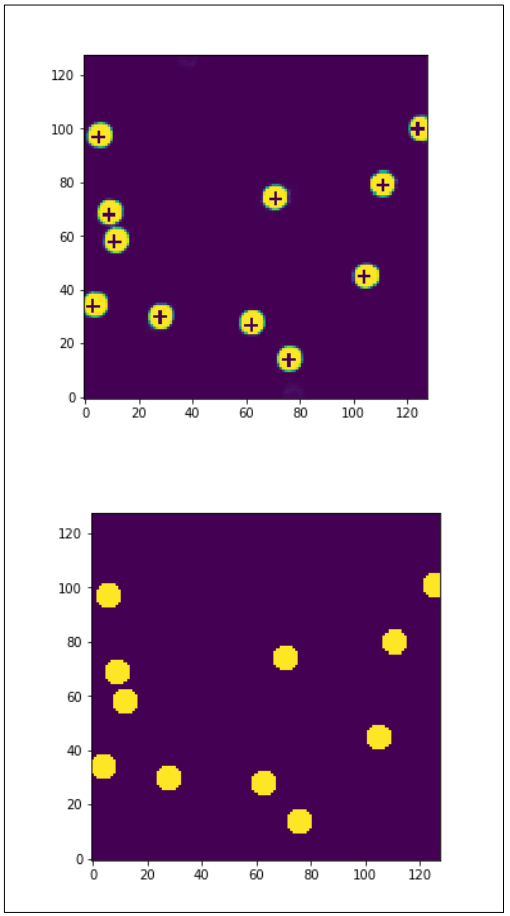
UNET input and mask
(128,128,5) x R and
(128,128,2) x 1/0
list of true centroids

UNET times k-means optimized

In this step we use the trained UNET with 100K patches and afterwards we apply the k-means based algorithm

We run this algorithm over 1000 unseen patches

Here we just copy the true information to the evaluation data set

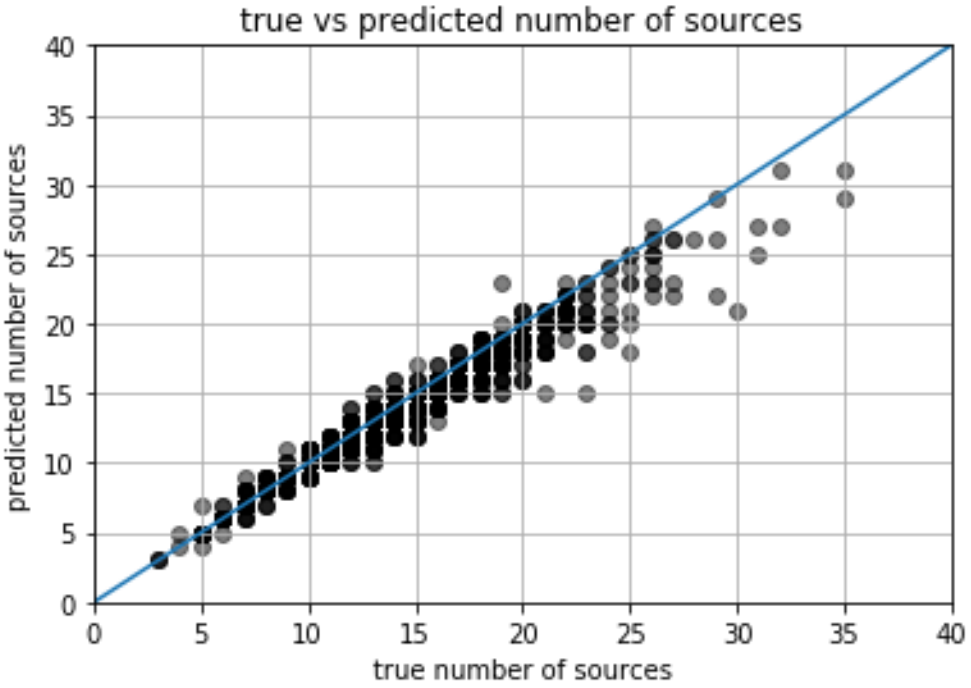


Algorithm output and mask
(128,128,2) x softmax
(128,128,2) x 1/0
list of true & predicted centroids

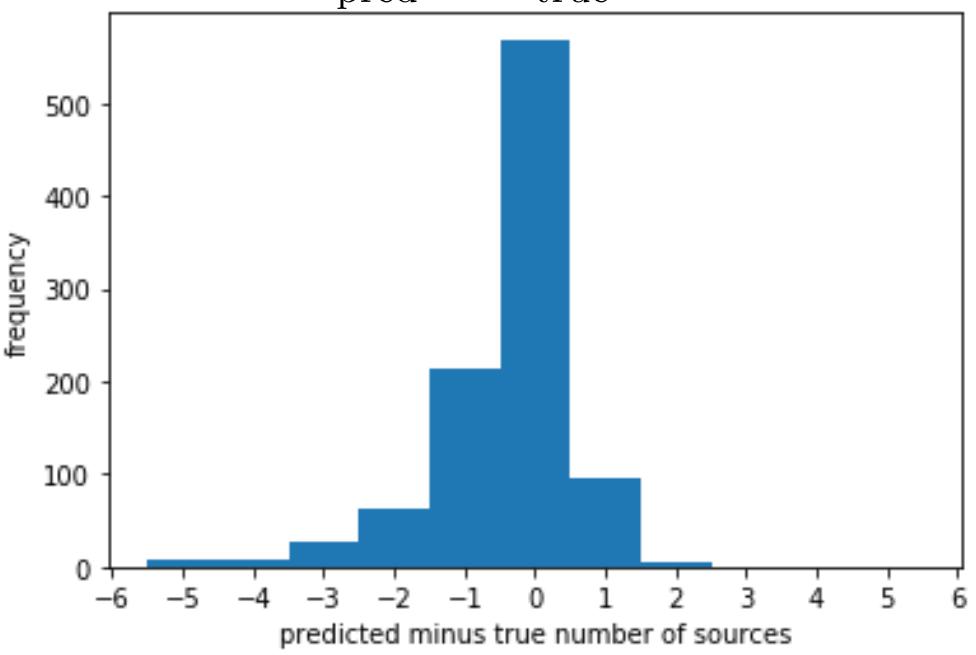
code: unet-plus-optimized-kmeans-localization-algorithm-evaluation.ipynb

Evaluation of the first working pipeline

Considering the data set generated by the pre-evaluation algorithm we are able to see the relation between the true number of sources and the predicted one for a set of 1000 unseen patches



$$N_{\mathrm{pred}} = N_{\mathrm{true}} \pm 1\,\mathrm{PS}$$



So, basically we can interpret that the error in the prediction of the number of sources is dominated and limited by the performance of the UNET

In the pre-evaluation dataset we also have available the positions of the sources and the true information from the simulations, so we can compute other metrics such as the number of true and false positives

The code-name of each patch is also saved in the pre-evaluation data set such that we can further investigate particular cases in order to understand the performance of the algorithm

code: unet-plus-optimized-kmeans-localization-algorithm-evaluation.ipynb

Comments

From the previous results we can see that in principle the chain UNET times k-means can give quite interesting results regarding the localization of Fermi point sources. However, there are some issues to be covered in order to optimize this procedure. Also, motivated by this approach we can speculate about possible solutions to the classification problem.

- **Regarding localization:**
  - The algorithm is running and evaluations are waiting
  - We can train the UNET with more data and increase the accuracy from current 0.9953
  - There are a handful of free arbitrary parameters that can be optimized
  - What about background?
  - We have to pass from patch like predictions to the full map coordinates in order to match the Fermi-Catalog properly
  - What about unsupervised localization?
  - Tests with observed data, when? after optimization I guess

- **Regarding classification:**
  - We can extend the output of the UNET from two to tree classes in order to include one layer with the AGN pixel to pixel probability, a second layer for pulsars and the third for background, but we have to be careful about activation functions (softmax or sigmoid) and overlap
  - Afterwards we can apply the optimized k-means algorithm to localize the respective sources
  - Also, we could use a fully convolutional neural network (LeNet like), logistic regression, with mini-patches that only contain the information of one source at each time.
  - Then, we could use the centroids predicted by our current algorithm to crop the original image classify one by one each source. This is working at 90% currently
  - The latter approach can be extended to some unsupervised classification algorithm, but in this case we still need to discuss how to make the localization unsupervised

Github code: **https://github.com/bapanes/Automatic-Sourcer**