# Detecting Sources in Fermi LAT Gamma-ray Maps Using Neural Networks

**Diana Horangic**[1], **Elena Orlando**[2], **and Andrew Strong**[3]

[1]University of New Hampshire and SLAC National Laboratory, [2]University of Trieste and Stanford University, [3]Max Planck Institute for Extraterrestrial Physics

## Abstract

The Fermi Large Area Telescope (LAT) has been in orbit of Earth since 2008 collecting gamma rays. One challenge in analyzing LAT data is detecting sources to know the various classes of gamma-ray sources and how many they are. Neural networks show impressive accuracy in many fields. Application of these networks to Fermi LAT data can potentially be more successful than traditional statistical methods of source detection. Here we present our first attempt to use a region-based convolutional neural network (Faster R-CNN) and then a Mask R-CNN, which has built-in instance segmentation, something that networks previously applied to data lack. We have generated three training and test datasets of simulated Fermi LAT images with different parameters such as noise and photon counts. These were used to separately train Facebook AI's Mask R-CNN model with a ResNet-50 backbone and feature pyramid network for instance segmentation of sources. We found this method to be promising and we present here our preliminary results.
We also discuss our current effort to create a new package based on Meta's Detectron2 research repository which would allow us to test different network configurations on simulated, binned LAT data.

## Introduction

Fermi-LAT detects gamma rays at energies between 30 MeV and 300 GeV. The response of LAT to a source can be roughly approximated by a 2-dimensional Gaussian. Traditionally, sources are detected in LAT data with a maximum likelihood analysis. This method depends on estimates of the non-point-source sky which introduce uncertainty, e.g. approximating interstellar emission from cosmic-ray interactions. Sources are fitted one by one but simultaneous detection of all sources is desirable for consistency. Machine learning may overcome these deficiencies, and therefore we are investigating the performance of various architectures.
Two important machine learning tasks are object detection and instance segmentation. Object detection is the localization of objects and instance segmentation is both the localization of objects and the delineation of separate objects with a *mask*. Neural networks involve layers of nodes that depend on representations, or transformations, of input [1]. A Mask R-CNN (region-based convolutional neural network) uses a multi-task loss function that combines the loss of classification (what *class* of object is this?), localization (*where* is this object?), and segmentation (what are the *boundaries* of this object?). The classification loss $\mathcal{L}_{class}$ is a log loss function over two classes (background or object of interest). The localization loss $\mathcal{L}_{local}$ is a smooth L1 loss. The segmentation loss $\mathcal{L}_{seg}$ is the average binary cross-entropy. The multi-task loss to be minimized is therefore

$$\mathcal{L} = \mathcal{L}_{class} + \mathcal{L}_{local} + \mathcal{L}_{seg}.$$

In Meta AI's configuration of a Mask R-CNN, multi-task loss is minimized via the Adam optimizer algorithm, an extension of stochastic gradient descent that is more well-suited for noise-related issues than gradient descent. A Mask R-CNN is built on a Faster R-CNN, which is only capable of object detection.
In training neural networks, the hyper-parameters of *base learning rate, momentum, gamma, batch size,* and *epoch* are often used. A network's optimization algorithm will start with the base learning rate, will accelerate according to the momentum, will reward itself according to the gamma, and will sample a number of images given by the batch size from the dataset for propagation. An epoch is how many times a model propagates every example in the dataset.

## Source Generator and Datasets

A pipeline for dataset generation was made with a source generator, a Python class which generates an annotated corresponding 400 by 400 pixel jpeg. This pipeline can either generate corresponding .xml files in Pascal VOC format or a .json file in COCO format. Per each image, an empty 2-dimensional array was created and filled with a 2D Gaussian(s) centered at each coordinate of a source. A background constant was added to each pixel, which were then transformed by a Poisson random number generator to some integer. For sources, the latitudinal and longitudinal coordinates, photon counts, and Gaussian width were used as parameters. For each patch of sky (and corresponding jpeg) generated, the background constant, number of sources, and pixel size were also parameters. Three training and test sets were generated. The first test set is 8 images with 15 sources in each image. The second and third test sets are 13 images that have either zero sources or 15 sources. All test sets have the same variation in source parameters as their corresponding train set.

| # Set | # Images | Counts | Gaussian Width | Background Constant | # Sources |
|---|---|---|---|---|---|
| 1 | 400 | 1000-5000 | 0.5-1.5 | 2.0-8.0 | 3-6 |
| 2 | 800 | 800-5000 | 0.5-1.5 | 2.0-10.0 | 3-6 |
| 3 | 1200 | 500-5000 | 0.5-1.5 | 2.0-12.0 | 3-6 |

Fig. 1: Parameters for each of the training sets.

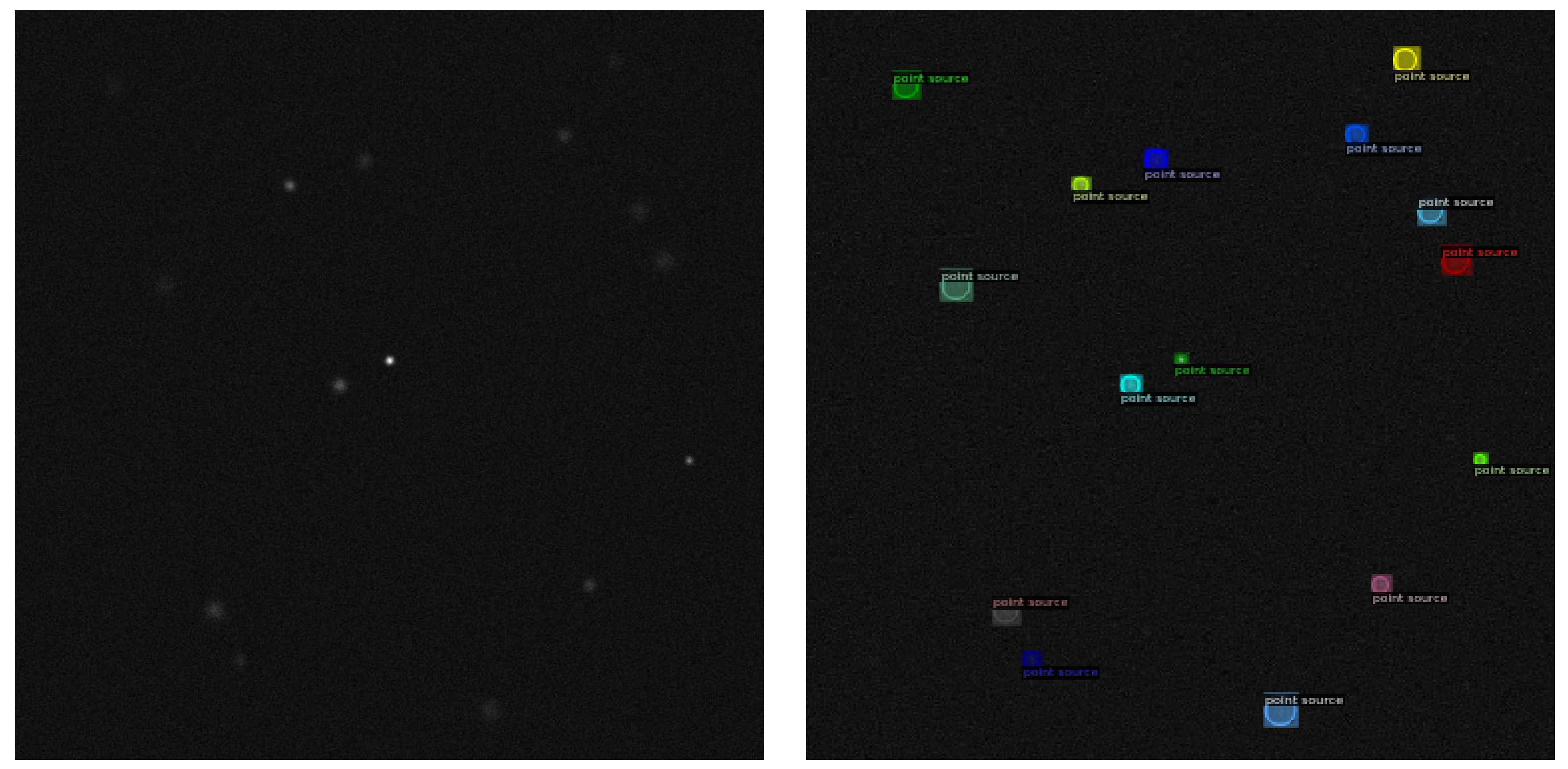

Fig. 2: Un-annotated and annotated image from the second test dataset, there are 15 sources in this image.

## Faster R-CNN and Mask R-CNN

A Faster R-CNN through the Detecto package was first trained. Then, a Mask R-CNN from Meta AI's Detectron2 model zoo was trained and tested on all datasets separately. The following hyper-parameters were used:

- The *base learning rate* is the learning rate that gradient descent or another optimization algorithm will start with.
- The *momentum* is the moving average of past gradients and is used to accelerate the gradient in the correct direction.
- The *gamma* parameter is between 0 and 1 and is a quantification of algorithm rewards.
- The *batch size* is the number of images that the model takes and propagates.
- An *epoch* occurs when the model has propagated every example in the dataset.

The hyper-parameter values of the model trained on the first, second, and third datasets were: a base learning rate of 0.01, 0.02, 0.02, a momentum of 0.9, 0.9, 0.9, a gamma parameter value of 0.5, 0.5, 0.5, and a batch size of 3, 3, and 3 respectively. Total epochs

## Results

A false positive (FP) denotes a detection where there is no source and a false negative (FN) denotes a failure to detect a source. A true positive is when a source is correctly predicted and a true negative is when it is correctly not predicted. We have, per pixel,

$$\text{Accuracy} = \frac{\text{True Positives} + \text{True Negatives}}{\text{All Detections}}.$$

The ratio of intersection over union (IOU) is the ratio of intersection between bounding box area predicted by the model and real bounding box area to the area of the union of the two. Metrics used to measure performance were average precision (AP) over 10 IOU values from 0.50 to 0.95, the Pascal VOC average precision metric (AP50), where AP is calculated across one IOU value of 0.5, a strict average precision metric (AP75) calculated across an IOU value of 0.75, and the average recall (AR). The rates of false negatives and positives and the accuracy were also generated. All metrics show the model is capable of accurate source detection.

| # Set | Task | AP | AP50 | AP75 | AR | FN | FP | Accuracy |
|---|---|---|---|---|---|---|---|---|
| 1 | bbox | 83.72 | 100.00 | 97.82 | 0.874 | | | |
| | segm | 80.09 | 100.00 | 97.82 | 0.839 | 0.032 | 0.088 | 95.31 % |
| 2 | bbox | 81.62 | 100.00 | 94.57 | 0.846 | | | |
| | segm | 84.08 | 100.00 | 97.81 | 0.870 | 0.034 | 0.112 | 94.45 % |
| 3 | bbox | 71.19 | 97.01 | 86.35 | 0.744 | | | |
| | segm | 73.60 | 97.01 | 88.43 | 0.767 | 0.052 | 0.119 | 92.94 % |

Fig. 3: Metrics for all sets and tasks. All metrics are evaluated on the test set.

## Package Development

Our source generator simulates single-channel images and therefore the trained models are only capable of performing detection on a single energy bin. We began developing a package for simultaneous detection on multiple energy bins. This package is based on Meta AI's detectron2 package for deep learning research and will allow users to test different architectures on simulated and binned Fermi LAT data.

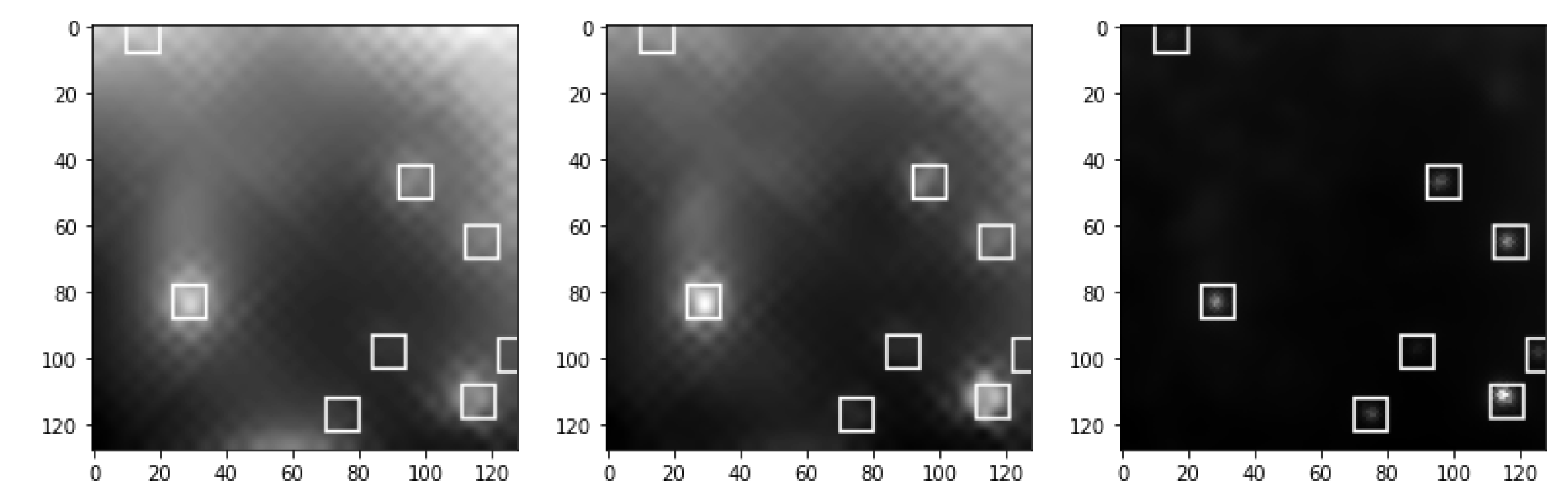

Fig. 4: Simulated data across 3 different energy bins shown here, courtesy of Saptashwa Bhattacharyya.

## References

[1] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. `http://www.deeplearningbook.org`. MIT Press, 2016.