PROCEEDINGS OF SCIENCE

Detecting Fermi LAT Gamma-ray Sources with Neural Networks

Diana Horangic,[a,*] **Elena Orlando,**[b,c] **Andrew Strong**[d] **and Saptashwa Bhattacharyya**[e]

[a] *SLAC National Accelerator Laboratory, 2575 Sand Hill Road, Menlo Park, CA, USA*
[b] *Department of Physics, University of Trieste, Piazzale Europa, 1, Trieste, TS, IT*
[c] *Physics Department, Stanford University, 450 Serra Mall, Stanford, CA, USA*
[d] *Max Planck Institute for Extraterrestrial Physics, Gießenbachstraße 1, Garching bei München, DE*
[e] *Centre for Astrophysics and Cosmology, University of Nova Gorica, Vipavska 11c, SI-5270 Ajdovščina, Slovenia*
E-mail: diana.horangic@unh.edu, orlandele@gmail.com, aws@mpe.mpg.de, sbhattacharyya@ung.si

...

27th European Cosmic Ray Symposium - ECRS
25-29 July 2022
Nijmegen, the Netherlands

*Speaker

PoS(ECRS)117

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 and knowing the various classes of gamma-ray sources and how many there 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 attempt at a flexible neural architecture Python package, *fermidetect*, designed specifically to train and predict on simulated and real Fermi LAT data. This package is based heavily on Meta AI's Detectron2 [1] deep learning framework and will be used to test the performance of different algorithms and hyperparameters on simulated LAT data.

1. Introduction

Machine Learning (ML) is the study of algorithms that improve gradually at some task by learning from available data. Among various applications of ML, focusing on vision, two essential tasks are object detection and segmentation. Object detection is about locating objects with bounding boxes but segmentation is about performing classification at the pixel level of an image or video frame. Within segmentation, semantic segmentation performs pixel-level labeling for a set of object categories (tree, bird etc.), while instance segmentation extends this further by detecting and delineating each object of interest in the image. Neural networks are a subset of machine learning that involve layers of nodes that depend on representations, or transformations, of input [2]. In the input case of an image, a representation could be features like a dog's ear shape or a balloon's color. The most simple networks have one loss function which gives information on whether a model describes data well, this loss function must be minimized. However, more complex models might use a combination of loss functions. For example, 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?) [3]. 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, which is a Huber loss with an additional adjustable parameter β,

$$\mathcal{L}_{local} = \begin{cases} \frac{x^2}{2\beta} & \text{if } |x| < \beta \\ |x| - \frac{\beta}{2} & \text{else.} \end{cases} \tag{1}$$

For a single example then, $x = \hat{y} - y$ where $\hat{y}$ is the predicted label and y is the ground truth label. The segmentation loss $\mathcal{L}_{seg}$ is the average binary cross-entropy. The multi-task loss to be minimized is therefore

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

As the loss function of a model is minimized, a set of weights is developed that can be used to predict the existence, location, and class of an object in an image or other media. In the case of Mask R-CNN, the 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.

The size of a network and the scale of the training data determine success at a task. When it comes to large training sets, neural networks tend to dominate in performance over lower level supervised learning algorithms like support vector machines.

Fermi-LAT measures gamma-ray photons at energies ranging from MeV to TeV. In 12 years, LAT can measure a billion gamma-ray photons. Traditionally, sources are detected in LAT data through a maximum likelihood analysis where $\mathcal{L}_{H_0}$ is the likelihood of a model without the candidate source and $\mathcal{L}_{H_A}$ is the likelihood of a model with the candidate source. A test statistic is given by

$$TS = -2 \times \log \frac{\mathcal{L}_{H_0}}{\mathcal{L}_{H_A}}.$$

If the test statistic TS is larger than a threshold of detection, the null hypothesis is rejected and the candidate source is confirmed [4].

Source detection in LAT data is still an active area of research. Traditional methods require an approximation of background radiation and therefore introduce a degree of uncertainty. LAT data is largely unstructured and there is no clear detection of faint sources. Neural networks show impressive accuracy at interpreting unstructured data and a previous attempt has been made to localize and classify simulated gamma-ray sources based on LAT catalog [5]. However, architecture design space for this task is at the moment vast and more models (and their hyper-parameters) like RetinaNet or Mask R-CNN should be tested. In addition to this, pipelines for prediction on real data using trained models still need to be built.

2. Simulated Data

The response of a telescope to a point source of radiation can be roughly approximated by a 2-dimensional Gaussian

$$f(x, y) = \exp\left(-\left(\frac{(x - x_0)^2}{2\sigma_x^2} + \frac{(y - y_0)^2}{2\sigma_y^2}\right)\right),$$

where (x_0, y_0) is the center of the Gaussian. The Gaussian width σ_x is in the x direction and the Gaussian width σ_y is in the y direction. LAT data can be simulated through this basic level modeling [6] or through the fermitools Python package.

The preliminary dataset used to train fermidetect architectures was 767 test patches of sky, which were simulated across 5 energy bins. Bins were defined based on the sensitivity of LAT and are 0.3-0.5 GeV, 0.5-1 GeV, 1-2 GeV, 2-7 GeV, and 7-20 GeV. Sources were simulated based on the properties of various sources in LAT-DR2 catalog using fermitools. Galactic diffuse and isotropic emission, which typically make sources harder to pick up on with the traditional methods, were also taken into account. Sources were annotated with bounding boxes. Because different energy bins have different resolutions, and thus different dimensions by nature, all bins were upsampled or downsampled to an $n \times n$ patch size in preparation for stacking depth-wise. For a single training example across 5 energy bins, see Figure 1. Energy bins were then stacked depth-wise, preserving the counts of each bin, and final training data had the dimensions $n \times n \times 5$.

Bounding box annotations can be converted from .csv format to COCO format through the conversion module and saved in a .json file. Datasets generated with different interpolation methods or different end patch size n are stored and can be automatically trained on later.

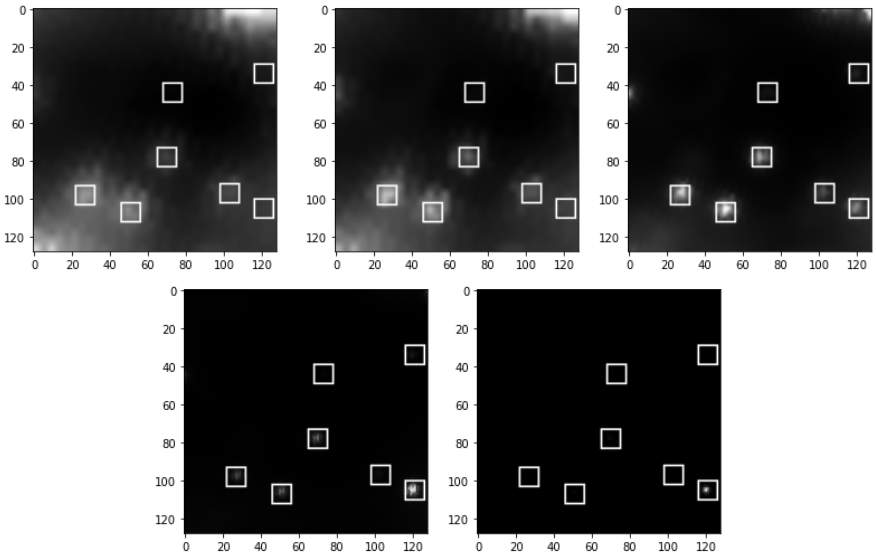
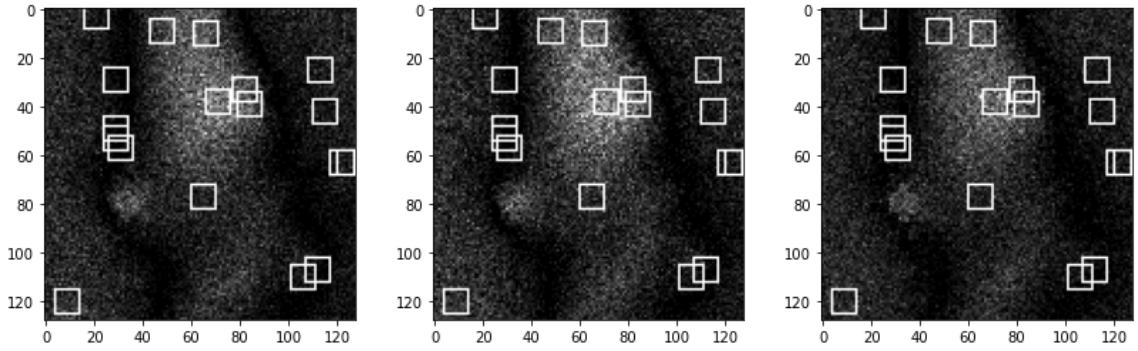


Figure 1: Simulated data across 5 different energy bins shown here in in increasing GeV order. Each patch and its counts have been bilinearly interpolated to a standard size of 128×128.

3. Fermidetect

In the case of LAT data, a neural architecture takes as input a patch of simulated sky with separate counts of 5 or more energy bins. The config module of fermidetect allows many different Generalized R-CNN configurations to be generated and then saved as .yaml files. Users can then load these configurations automatically, merge them with the default Generalized R-CNN configuration, and compare training results to determine optimal pre-processing methods, hyper-parameters, network backbones, region proposal methods, training methods and more.

3.1 Pre-processing



(a) Upsampled with a bicubic interpolation.

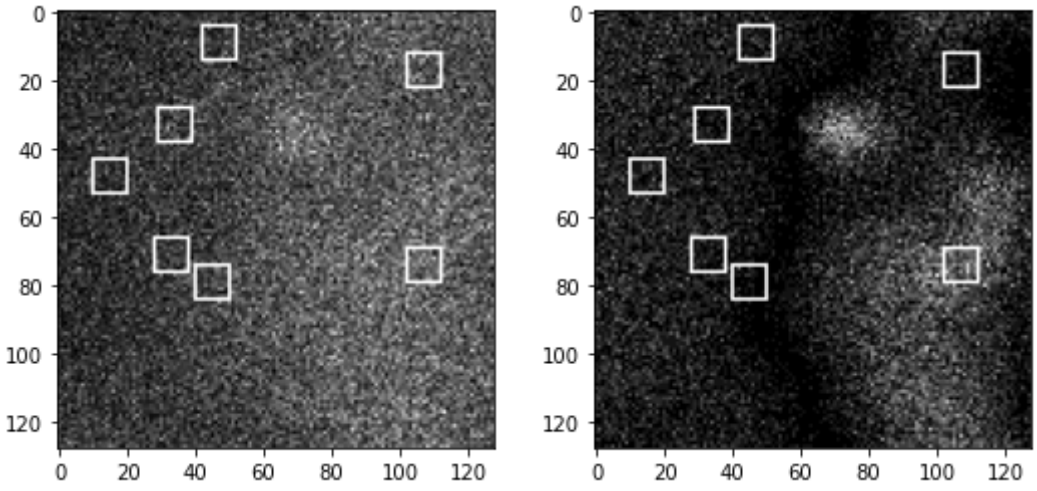
(b) Upsampled with a bilinear interpolation.

(c) Upsampled with a nearest-neighbor interpolation.

Figure 2: Patch of sky with simulated counts of a 0.3-0.5 GeV energy bin, upsampled with different interpolation methods from 32×32 to 128×128. Each image has been standardized through the configuration module in preparation for stacking.

The structure of fermidetect necessitates stacking of different energy bins before training. The available upsampling methods are the bilinear, bicubic, and nearest neighbor algorithms. For a comparison of the different methods on patches before stacking, see Figure 2.

Pre-processing can be tightly controlled using the configuration module of fermidetect. Counts in each energy bin were checked before interpolation and then scaled to their pre-interpolated value through a calculated ratio, preserving their original value. Two options for feature scaling of patches before propagation also exist: normalization via min-max scaling and standardization. For a comparison of these two pre-processes, see Figure 3.



(a) Feature scaled via normalization.

(b) Feature scaled via standardization.

Figure 3: Different feature scaling methods applied to another patch of sky with simulated counts of a 0.3-0.5 GeV energy bin.

3.2 Network Architectures

A generalized R-CNN template is provided with the package. Users can vary attributes of this architecture through the config module. The current available region proposal methods for an R-CNN double stage model are the selective search algorithm and a feature pyramid network (FPN). Users can select a ResNet of 50 or 101 layers for this FPN. RetinaNet is the only single stage model included. The following main hyper-parameters can also be adjusted:

1. The *base learning rate* is the learning rate that gradient descent or another optimization algorithm will start with. The base learning rate will decrease or increase according to other hyper-parameters and is usually neither independent nor constant. A learning rate too small may miss the loss minimum and land erroneously on local minima. It can also over-fit the data. A too-large learning rate will cause the model to converge too quickly and may lead to under-fitting.

2. One iteration is one batch being propagated through the network. *Maximum iteration* is the total number of times this occurs.

3. The *momentum* is the moving average of past gradients and is used to accelerate the gradient in the correct direction.

4. The *gamma* parameter is also called the discount factor. It is between 0 and 1 and is a quantification of algorithm rewards. A lower value of gamma will push the model to reward itself immediately for smaller tasks, a higher value of gamma will push the model to wait to solve more difficult tasks before rewarding itself.

4. Conclusion

So far, fermidetect loads a generalized R-CNN architecture from a YAML file, converts annotated and simulated LAT data to a COCO format, pre-processes these patches, and trains a network from scratch. Fermidetect is based on a package that supports segmentation learning and we also plan to add this feature in the future. Pre-training options will improve performance of networks and data used for this need not be as carefully simulated as our training data. A predictor module will also be built to apply trained networks to real patches of LAT data. The true test of fermidetect's usefulness will be its performance at source detection on real LAT data and how this compares to a traditional method of maximum likelihood analysis.

5. Acknowledgements

This research was funded in part by the Office of Science, Department of Energy's Science Undergraduate Laboratory Internship (SULI) under the direction of Elena Orlando at the Stanford Linear Accelerator Center (SLAC) Laboratory.

References

[1] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo and R. Girshick, "Detectron2." https://github.com/facebookresearch/detectron2, 2019.

[2] I. Goodfellow, Y. Bengio and A. Courville, *Deep Learning*, MIT Press (2016).

[3] K. He, G. Gkioxari, P. Dollár and R.B. Girshick, *Mask R-CNN*, *CoRR* **abs/1703.06870** (2017) [1703.06870].

[4] *Fssc: Fermi data analysis online documentation science tools: Cicerone - likelihood*, 2022.

[5] S. Caron, K. Dijkstra, C. Eckner, L. Hendriks, G. Jóhannesson, B. Panes et al., *Identification of point sources in gamma rays using u-shaped convolutional neural networks and a data challenge*, 2021.

[6] D. Horangic, E. Orlando and A. Strong, *Investigating Detection of Sources in Fermi Gamma-ray Data with Neural Networks*, in *American Astronomical Society Meeting Abstracts*, vol. 54 of *American Astronomical Society Meeting Abstracts*, p. 240.04, June, 2022.