

# Studying deep convolutional neural networks with hexagonal lattices for imaging atmospheric Cherenkov telescope event reconstruction

---

**D. Nieto<sup>\*1</sup>, A. Brill<sup>2</sup>, Q. Feng<sup>2</sup>, M. Jacquemont<sup>3,4</sup>, B. Kim<sup>5</sup>, T. Miener<sup>1</sup>, T. Vuillaume<sup>3</sup>**

<sup>1</sup>*Instituto de Física de Partículas y del Cosmos and Departamento de EMFTEL, Universidad Complutense de Madrid, Madrid, Spain,* <sup>2</sup>*Columbia University, Department of Physics, New York, NY, USA,* <sup>3</sup>*Laboratoire d'Annecy de Physique des Particules, CNRS, Université Savoie Mont-Blanc, Annecy, France,* <sup>4</sup>*LISTIC, Université Savoie Mont-Blanc, Annecy, France,* <sup>5</sup>*University of California Los Angeles, Division of Astronomy and Astrophysics, Los Angeles, CA, USA*

*E-mail:* [d.nieto@ucm.es](mailto:d.nieto@ucm.es)

Deep convolutional neural networks (DCNs) are a promising machine learning technique to reconstruct events recorded by imaging atmospheric Cherenkov telescopes (IACTs), but require optimization to reach full performance. One of the most pressing challenges is processing raw images captured by cameras made of hexagonal lattices of photo-multipliers, a common layout among IACT cameras which topologically differs from the square lattices conventionally expected, as their input data, by DCN models. Strategies directed to tackle this challenge range from the conversion of the hexagonal lattices onto square lattices by means of oversampling or interpolation to the implementation of hexagonal convolutional kernels. In this contribution we present a comparison of several of those strategies, using DCN models trained on simulated IACT data.

*36th International Cosmic Ray Conference -ICRC2019-  
July 24th - August 1st, 2019  
Madison, WI, U.S.A.*

---

<sup>\*</sup>Speaker.

## 1. Introduction

Imaging atmospheric Cherenkov telescopes (IACTs) observe the Cherenkov light emitted by cosmic rays entering the atmosphere. The light is collected by an optical system and focused onto an ultra-rapid camera. The image of the atmospheric shower must then be analyzed to determine the physical parameters (mainly particle type, arrival direction and particle energy) of the primary cosmic ray. This analysis, also known as event reconstruction, is a complex task and several approaches have been developed in the past. Some well known reconstruction strategies are based on the parametrization of the images, first attempted in [1], combined with multi-variate analysis methods [2, 3, 4, 5], or on image template methods [6, 7, 8].

Late advances in image analysis based on machine learning, and especially using deep convolutional neural networks (DCN) [9], are a promising opportunity to improve on current methods of IACT event reconstruction, leading to better sensitivity to the gamma-ray sky. Recent works have demonstrated the potential application of these methods for IACT event reconstruction [10, 11, 12].

However, one of the issues when dealing with images from IACTs is their shape. Cameras used in IACTs often present non-rectangular shapes, or even non-Cartesian pixel layouts with pixels arranged in hexagonal lattices. This is an issue when using standard convolutional algorithms implemented in conventional libraries such as TensorFlow [13] or PyTorch [14] that have been developed only for rectangular images with Cartesian pixel lattices.

In Sec. 2, we introduce two classes of methods to tackle the issue of hexagonal-lattice images as input for DCNs. Sec. 3 describes the strategy for assessing the relative performance between the proposed strategies and the tools used for this purpose. Sec. 4 contains a description of the dataset. Finally, some results and discussion are presented in Sec. 5.

## 2. Hexagonal lattices as input for deep convolutional networks

Hexagonally arranged input images can be made to work with DCNs by modifying either the inputs or the convolution algorithm. With the first approach, a transformation is applied to the input image to morph it into a Cartesian lattice. With the second, a dedicated hexagonal convolution is used. While the first approach is simpler to implement, the second induces no changes in the input data that could impact the performance of DCN-based models.

### 2.1 Mapping methods

The aforementioned transformations can be realized by means of a mapping between input pixels in the hexagonal input layout to pixels in the Cartesian output layout through a weighted, linear combination. The weights can be stored in a sparse array dubbed a *mapping table*, with shape (*camera pixels*, *output x*, *output y*), that can be pre-computed or calculated at initialization time and then applied to the input images, ensuring that different mapping methods do not substantially modify the training time for a given DCN-based model.

We explore five different image transformations, or mapping methods: oversampling, rebinning, nearest interpolation, bilinear interpolation, and bicubic interpolation. For oversampling, we divide every hexagonal pixel into  $n$ -by- $n$  square pixels, assigning the charge of the new square pixels as  $n^{-2}$  of the original hexagonal pixel, where typically  $n = 2$ . For nearest interpolation the

charges of the output pixels are assigned by means of a nearest-neighbour algorithm that computes, for each pixel in the output layout, the nearest neighbouring pixel in the input layout. For rebinning, the input data is finely sampled, turned into a 2D histogram, and eventually rebinned to the desired output dimensions.

For bilinear and bicubic interpolation we select, for each output pixel, the collection of input pixels that will be involved in the interpolation by means of Delaunay triangulation (the three closest input pixels for bilinear, the 12 closest input pixels for bicubic). Additionally, the normalization over the mapping tables allows us to approximately preserve the input image charge after the transformation (oversampling is the only method that exactly preserves the input image charge). For all methods other than oversampling, the dimension of the output lattice is arbitrary. Fig. 1 contains explanatory diagrams for each of the mapping methods mentioned above and Fig. 2 shows those transformations applied over the image of a simulated gamma-ray initiated shower.

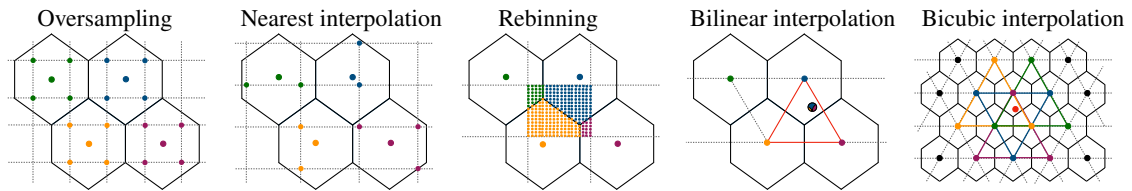


Figure 1: Diagrams depicting all the explored mapping methods.

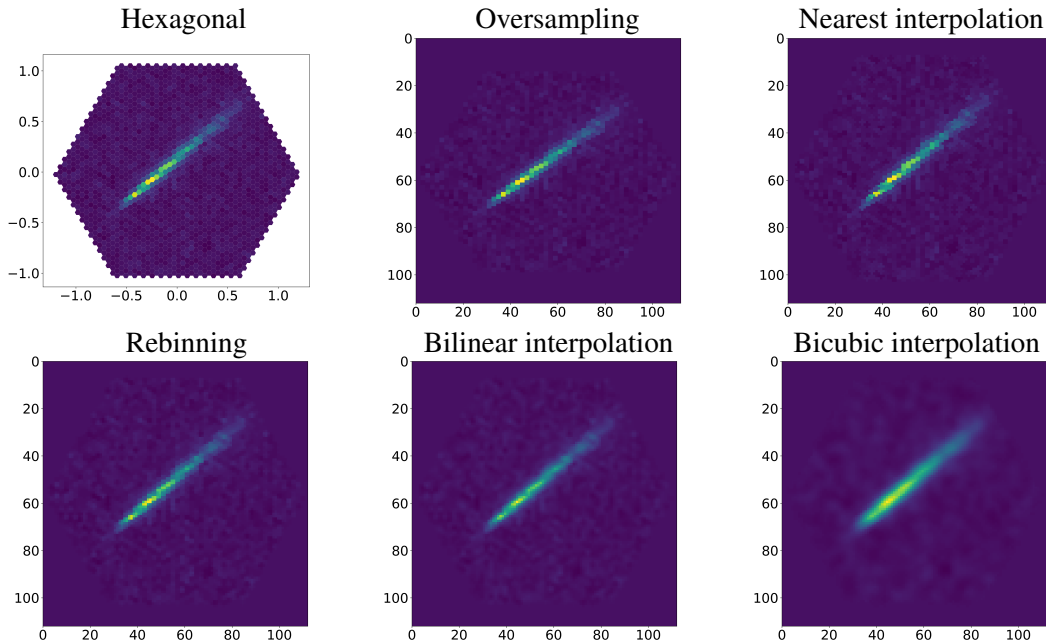


Figure 2: Image from a simulated gamma-ray event as seen by a camera with pixels arranged in a hexagonal lattice (top left), and the result of applying each of the explored mapping methods on that image.

## 2.2 Indexed convolution

Another approach to solve the hexagonal lattice challenge is to apply specific convolution operations to this pixel organization. In order to do so, we use the `IndexedConv` package [15], [16] that is based on PyTorch. This package allows the user to apply convolution to any pixel organization given that the matrix of pixel neighbours is provided. The necessary functions to deal with hexagonal grid images are also included in the package.

## 3. Methodology

We assess the performance of the methods proposed in Sec. 2 through the classification of single IACT images as gamma-ray or proton-induced events, using a labeled collection of simulated events for both classes. After selecting a simple DCN model, we read the images, stored as unidimensional arrays in HDF5 format, and transform them following the methods described in Sec. 2 by means of the `DL1-Data-Handler` package; the transformed images are then fed to two independent packages, `GammaLearn` and `CTLearn`, used to train an identical predefined model. These packages have been developed independently and are based on different libraries, respectively, PyTorch and TensorFlow. This allows us to cross-check the obtained results. The performances of the proposed methods are then compared in terms of the accuracy and AUC<sup>1</sup> metrics. Brief descriptions of the `DL1-Data-Handler`, `GammaLearn`, and `CTLearn` packages follow below.

**DL1-Data-Handler** The datasets used for training models in this work are stored in a custom HDF5-based file format defined by the `DL1-Data-Handler` (`DL1DH`) package [17]. This format, built on the `PyTables` library, is database-like in structure, consisting of tables of structured data which include information about telescope properties, the IACT array configuration, and the complete list of simulated events and their properties, as well as the images from the detected showers. A system of index-based mapping allows easy lookup of the variable number of telescope images associated with each shower event (and vice versa), while maintaining the efficient storage properties of the table-based `PyTables` HDF5 format.

Large datasets of IACT data simulated with `CORSIKA` and `sim_telarray` [18] can be processed efficiently into this HDF5 format using the `DL1DataWriter` module from `DL1DH`. `DL1DataWriter` is built upon `ctapipe` [19] data containers<sup>2</sup> and is designed to flexibly support a variety of other input file formats besides the ones described above, allowing the possibility of processing IACT data from sources other than `sim_telarray`. `DL1 Data Writer` implements several `PyTables` HDF5 storage optimizations, including indexing for faster lookup and chunking/compression to minimize file sizes.

`DL1DH` offers the `DL1DataReader` module to read the HDF5 files it produces. Internally, `DL1DataReader` can either map the camera images from the 1D arrays contained in the HDF5 files to 2D NumPy arrays, implementing all the mapping methods described in Sec. 2.1, or calculate the pixel neighbor matrix needed for indexed convolution, described in Sec. 2.2. Additionally, `DL1DataReader` implements efficient selection cuts on data and supports arbitrary event and

<sup>1</sup>Area under the receiver-operating-characteristic curve.

<sup>2</sup>The calibration of the raw `sim_telarray` data is performed using the available `ctapipe` calibration tools.

image filtering and user-defined transformations. Data loading and pre-processing for both GammaLearn and CTLearn are performed using `DL1DataReader`.

**GammaLearn** GammaLearn is a high-level Python package providing a framework to apply deep learning methods to IACT data using PyTorch. In particular, it solves some of the main challenges scientists are facing when applying DCNs to the reconstruction of IACT data: the application of convolutions on hexagonal images, the combination of stereoscopic images, and the reproducibility of the experiments via configuration files. `IndexedConv` is fully integrated in the GammaLearn framework. A more detailed description of this package can be found in [20, 21].

**CTLearn** CTLearn is a high-level Python package providing a backend for training deep learning models for IACT event reconstruction using TensorFlow. CTLearn allows its user to focus on developing and applying new models while making use of functionality specifically designed for IACT event reconstruction. CTLearn offers reproducible training and prediction, ensuring that settings used to train a model are explicitly and automatically recorded. Further details on CTLearn can be found in [22, 23].

**DCN model** We decided to work with simple models of proven particle classification capabilities, taking as a reference the *single-tel* model in CTLearn for the image mapping strategy and a slightly modified version of this model for the indexed convolution strategy. The *single-tel* model consists of four convolutional layers with 32, 32, 64, and 128 filters and a kernel size of 3 in each layer, interspaced by a ReLU activation layer followed by a max-pooling layer with a kernel size (and stride) of 2; the output of the last convolutional layer is flattened and fed to a fully connected layer with an output dimensionality of 2, the number of classes. The indexed convolution version of the model has hexagonal convolution kernels of size 7, corresponding to the first neighbours of the pixel to process. To fairly compare the presented mapping methods with unmapped (i.e. hexagonal) images, the first pooling layer is removed. The mapping methods increase the size of the images. Thus, the idea behind this adaptation is to have roughly the same number of pixels of interest (i.e. without taking into account the artificial pixels added by the mapping methods) in the feature maps in order to, except for the first convolution layer, apply convolution at the same level of fineness with respect to the original pixel size. No dropout or batch normalization were set for any of the models. We set the loss function to categorical cross-entropy and, as for the optimizer, we chose Adam with a learning rate of  $5 \cdot 10^{-5}$ .

#### 4. Dataset

The dataset that was utilized in this work is made of Monte Carlo simulated events for the Cherenkov Telescope Array (CTA)<sup>3</sup>. CTA is the next generation ground-based observatory for gamma-ray astronomy at very high energies, aiming to improve on the sensitivity of current-generation experiments by an order of magnitude and provide energy coverage from 20 GeV to more than 300 TeV. CTA will access the full sky thanks to an installation in the Northern Hemisphere (La Palma, Spain) and another installation in the Southern Hemisphere (Cerro Paranal, Chile), featuring more than 100 IACTs in total. Events from the third large-scale Monte Carlo production [24] were reduced, from raw to calibrated images, on the EGI<sup>4</sup> by means of the DL1 Data

<sup>3</sup>[www.cta-observatory.org](http://www.cta-observatory.org)

<sup>4</sup>[www.egi.eu](http://www.egi.eu)

LST		Oversampling	Rebinning	Nearest interp.	Bilinear interp.	Bicubic interp.	Indexed conv.
Validation	ACC	0.703±0.004	0.702±0.005	0.689±0.003	0.704±0.004	0.697±0.004	0.703±0.004
	AUC	0.788±0.004	0.789±0.005	0.773±0.004	0.790±0.004	0.782±0.004	0.786±0.003
Test	ACC	0.709±0.003	0.708±0.004	0.695±0.003	0.709±0.003	0.701±0.004	0.709±0.001
	AUC	0.795±0.002	0.795±0.003	0.779±0.003	0.796±0.003	0.787±0.003	0.790±0.002
MST-F		Oversampling	Rebinning	Nearest interp.	Bilinear interp.	Bicubic interp.	Indexed conv.
Validation	ACC	0.752±0.004	0.752±0.004	0.738±0.005	0.754±0.004	0.738±0.004	0.754±0.002
	AUC	0.838±0.003	0.836±0.004	0.825±0.005	0.839±0.004	0.824±0.004	0.840±0.002
Test	ACC	0.750±0.006	0.751±0.004	0.740±0.004	0.752±0.004	0.735±0.003	0.754±0.002
	AUC	0.835±0.005	0.835±0.003	0.824±0.004	0.837±0.003	0.822±0.003	0.840±0.002
SST-1M		Oversampling	Rebinning	Nearest interp.	Bilinear interp.	Bicubic interp.	Indexed conv.
Validation	ACC	0.772±0.005	0.774±0.004	0.760±0.003	0.777±0.005	0.769±0.002	0.777±0.003
	AUC	0.853±0.004	0.854±0.003	0.839±0.003	0.858±0.003	0.853±0.002	0.857±0.003
Test	ACC	0.772±0.003	0.773±0.003	0.760±0.003	0.777±0.003	0.771±0.002	0.776±0.003
	AUC	0.852±0.002	0.853±0.003	0.838±0.003	0.858±0.002	0.853±0.002	0.857±0.002

Table 1: Average and standard deviation of the learning metrics obtained from all the training runs, for both the validation and the test set.

Writer in DL1DH. We selected simulated data for the Southern installation, with a Zenith angle of  $20^\circ$  and an Azimuth angle of  $0^\circ$  (North pointing), and the "S8" layout (according to the notation in [24]). Such layout consists of four large-size telescopes (LSTs), 25 medium-size telescopes (MSTs), and 70 small-size telescopes (SSTs). Three models of MST and three models of SST were originally simulated, but we restricted ourselves to the single-mirror MST featuring FlashCam as its camera (MST-F) and the single-mirror SST-1M respectively. The dataset contains both diffuse gamma-ray and proton-initiated showers with balanced statistics, accounting for nearly 400 thousand events (1.4 million images, since most events trigger more than one telescope). The selected events were randomly drawn from the source dataset and then split following a 8/2 ratio into a train dataset and a test dataset.

## 5. Results

We trained the models in Sec. 3 with batches of 64 individual images (samples) randomly drawn from the training dataset described in Sec. 4, and stopped after 50,000 batches were seen by the model. We trained on each type of telescope independently. The training was monitored by periodically validating the evolution of the loss function, and the accuracy and AUC metrics. The samples used for the validation were randomly drawn from the training dataset (amounting to 10% of it) and never used during actual training. Image mapping methods were explored with CTLearn, while the experiments with indexed convolution were conducted with GammaLearn. Fig. 3 (left panel) exemplifies the evolution of the AUC during training for the three types of telescopes and all the tested methods. We performed independent training runs with different random seeds for each explored method (10 runs for CTLearn, 5 runs for GammaLearn) to assess if the differences in performance between methods were due to stochastic fluctuations alone. The obtained results are summarized in Table 1 and Fig. 3 (right panel).

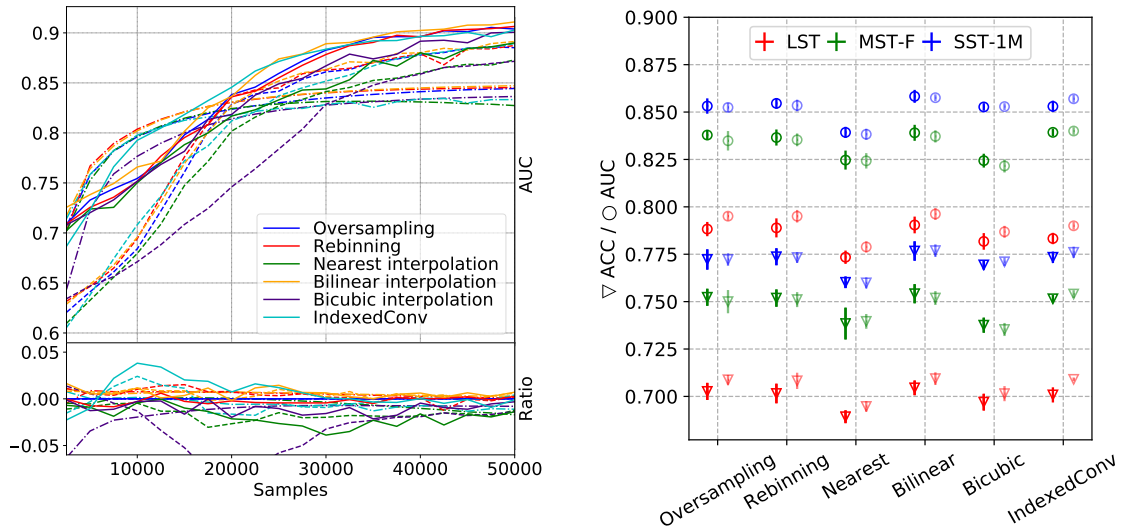


Figure 3: *Left*: Example of the evolution of the AUC during a given training run for the LST (dash dotted), the MST-F (dashed) and the SST-1M (solid). Ratio to oversampling method shown for comparison purposes. *Right*: Average and standard deviation of the learning metrics for all runs. Bright and pale markers depict train and test sets, respectively.

## 6. Conclusion and Outlook

The standard deviation of the distribution of learning metrics obtained from all the training ranges from 0.2% to 0.5% in the validation set, almost identical in the test set, without noticeable distinction between methods or telescope types. The values for accuracy and AUC from the validation and test datasets are compatible within errors for all methods and telescope types. Bilinear interpolation is consistently superior, although not significantly better than most of the other methods, being consistent with indexed convolution, oversampling, rebinning and bicubic interpolation within errors. Nearest interpolation consistently underperforms and there are strong hints that this method could be significantly worse than the rest.

Future studies should test whether these results generalize to further aspects of IACT event reconstruction, like the estimation of the energy and arrival direction of the simulated events. In addition, when performing model optimization, the mapping method or the usage of indexed convolution could be considered as an additional hyperparameter to be explored. Eventually, the final test would be to verify if all those results still hold when working with real data.

## 7. Acknowledgments

This work was conducted in the context of the CTA Analysis and Simulations Working Group. We thank Johan Bregeon for his support reducing our dataset on the EGI. DN and TM acknowledge support from the former *Spanish Ministry of Economy, Industry, and Competitiveness / European Regional Development Fund* grant FPA2015-73913-JIN. This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreements 653477 and 824064. This work has been done thanks to the facilities offered by the Univ. Savoie

Mont Blanc - CNRS/IN2P3 MUST computing center. AB acknowledges support from NSF award PHY-1229205. BK acknowledges support from NSF awards 1229792 and 1607491. DN acknowledges the support of NVIDIA Corporation with the donation of a Titan X Pascal GPU used for this research. MJ and TV gratefully acknowledge the support of the NVIDIA Corporation with the donation of one NVIDIA P6000 GPU for this research.

This paper has gone through internal review by the CTA Consortium.

## References

- [1] Hillas, A. M., *Cerenkov light images of EAS produced by primary gamma*, *International Cosmic Ray Conference* **3** (1985)
- [2] Albert, J. *et al.*, *Implementation of the Random Forest method for the Imaging Atmospheric Cherenkov Telescope MAGIC*, *Nuclear Instruments and Methods in Physics Research A* **588** 424–432 (2008), doi:10.1016/j.nima.2007.11.068
- [3] Ohm, S., van Eldik, C., Egberts, K.,  *$\gamma$ /hadron separation in very-high-energy  $\gamma$ -ray astronomy using a multivariate analysis method*, *Astroparticle Physics* **31**, 383 (2019)
- [4] Becherini, Y., Djannati-Ataï, A., Marandon, V., Punch, M., and Pita, S., *A new analysis strategy for detection of faint  $\gamma$ -ray sources with Imaging Atmospheric Cherenkov Telescopes*, *Astroparticle Physics* **34** 858–870 (2011), [arXiv:1104.5359]
- [5] Krause, M., Pueschel, E., and Maier, G., *Improved  $\gamma$ /hadron separation for the detection of faint  $\gamma$ -ray sources using boosted decision trees*, *Astroparticle Physics* **89** 1–9 (2017)
- [6] Lemoine-Goumard, M., Degrange, B., Tluczykont, M., *Selection and 3D-reconstruction of gamma-ray-induced air showers with a stereoscopic system of atmospheric Cherenkov telescopes*, *Astroparticle Physics* **25**, 195 (2006)
- [7] de Naurois, M., Rolland, L., *A high performance likelihood reconstruction of  $\gamma$ -rays for imaging atmospheric Cherenkov telescopes*, *Astroparticle Physics* **32**, 231.
- [8] Parsons, R. D., Hinton, J. A., *A Monte Carlo template based analysis for air-Cherenkov arrays*, *Astroparticle Physics* **56**, 26 (2014)
- [9] Goodfellow, I., Bengio, Y. and Courville, A., *Deep Learning*, MIT Press (2016), [www.deeplearningbook.org](http://www.deeplearningbook.org)
- [10] Nieto, D., Brill, A., Kim, B., Humensky, T. B., *et al.*, *Exploring deep learning as an event classification method for the Cherenkov Telescope Array*, 35th International Cosmic Ray Conference (2017), doi:10.22323/1.301.0809
- [11] Mangano, S., *et al.*, *Extracting gamma-ray information from images with convolutional neural network methods on simulated Cherenkov Telescope Array data*, ANNPR (2018), LNAI 11081, p.243-254, doi:10.1007/978-3-319-99978-4
- [12] Shilon, I. *et al.*, *Application of deep learning methods to analysis of imaging atmospheric Cherenkov telescopes data*, *Astroparticle Physics* **105**, 44 (2019)
- [13] Abadi, M., *et al.*, *TensorFlow: Large-scale machine learning on heterogeneous systems*, 2015. Software available from [tensorflow.org](http://tensorflow.org)
- [14] Paszke, A. *et al.*, *Automatic differentiation in PyTorch*, NIPS-W (2017)
- [15] Jacquemont, M., Antiga, L., Vuillaume, T. *et al.*, *Indexed Operations for Non-rectangular Lattices Applied to Convolutional Neural Networks*, in *Proceedings of the 14th VISIGRAPP - Volume 5: VISAPP*, p. 362-371 (2019), doi:10.5220/0007364303620371
- [16] Jacquemont, M., and Vuillaume, T., *IndexedConv/IndexedConv (Version v1.1)*, Zenodo (2019), doi:10.5281/zenodo.2542664
- [17] *DL1 Data Handler: HDF5 format IACT data writer, reader and processor*, Zenodo (2019) doi:10.5281/zenodo.3336560, <https://github.com/cta-observatory/dl1-data-handler>.
- [18] Bernlöhr, K., *Simulation of imaging atmospheric Cherenkov telescopes with CORSIKA and sim\_telarray*, *Astroparticle Physics* **30** 149–158 (2008), [arXiv:0808.2253]
- [19] Kosak, K. *et al.* *ctapipe: Low-level data processing prototype for the Cherenkov Telescope Array*, 36th International Cosmic Ray Conference (2019), <https://github.com/cta-observatory/ctapipe>.
- [20] Jacquemont, M. *et al.*, *GammaLearn: a Deep Learning framework for IACT data*, 36th International Cosmic Ray Conference (2019)
- [21] Vuillaume, T., *et al.*, *GammaLearn - first steps to apply Deep Learning to the Cherenkov Telescope Array data*, 23rd International Conference on Computing in High Energy and Nuclear Physics (2018)
- [22] *CTLearn: Deep Learning for IACT Event Reconstruction*, Zenodo (2019), doi:10.5281/zenodo.3342952, [github.com/ctlearn-project/ctlearn](https://github.com/ctlearn-project/ctlearn)
- [23] Nieto, D., *et al.*, *CTLearn: Deep Learning for Gamma-ray Astronomy*, 36th International Cosmic Ray Conference (2019)
- [24] Acharyya, A. *et al.*, *Monte Carlo studies for the optimisation of the Cherenkov Telescope Array layout*, *Astropart. Phys.* **111**, 35 (2019), doi:10.1016/j.astropartphys.2019.04.001