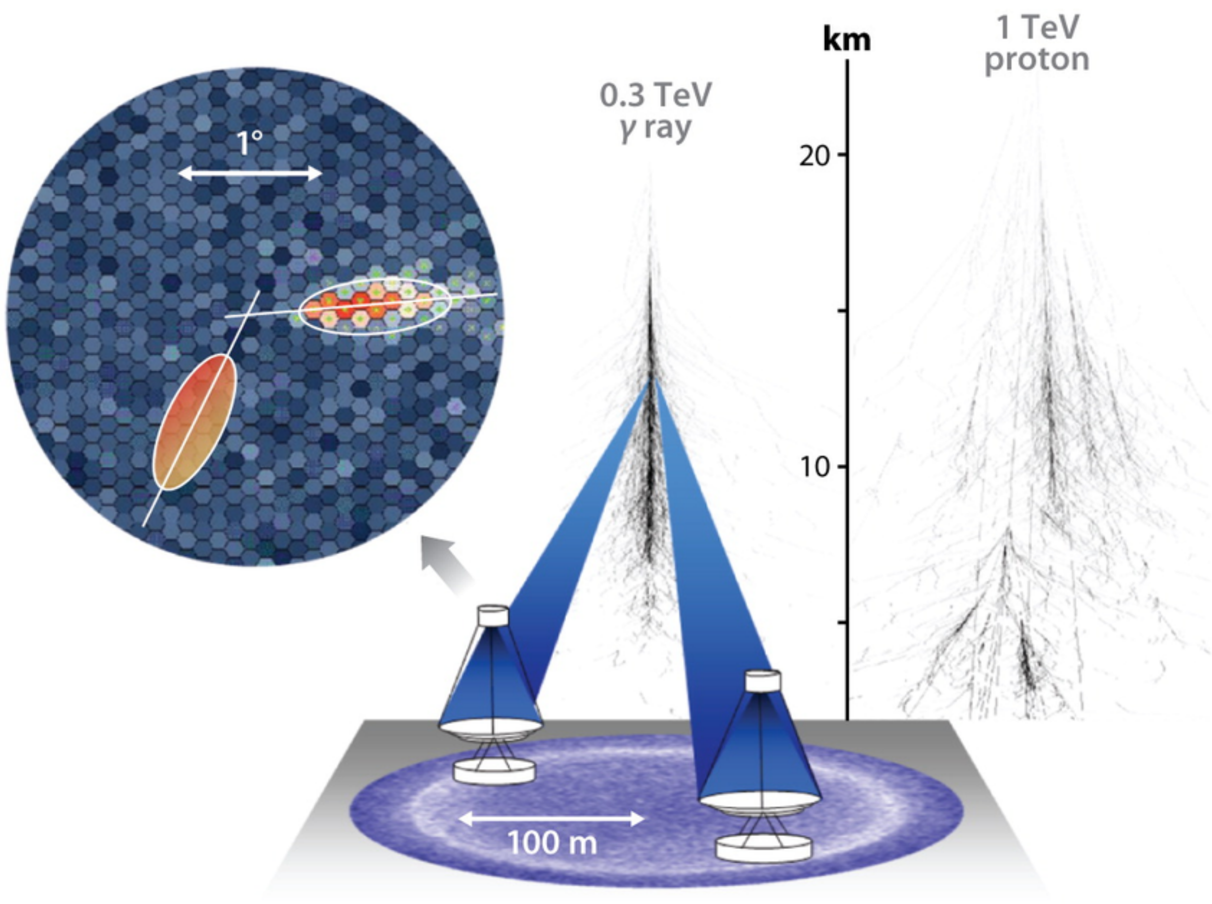
Air Shower Reconstruction With Hexagonal CNNs

Constantin Steppa, Tim L. Holch, Kathrin Egberts, Mark Olchanski

Imaging Atmospheric Cherenkov Technique

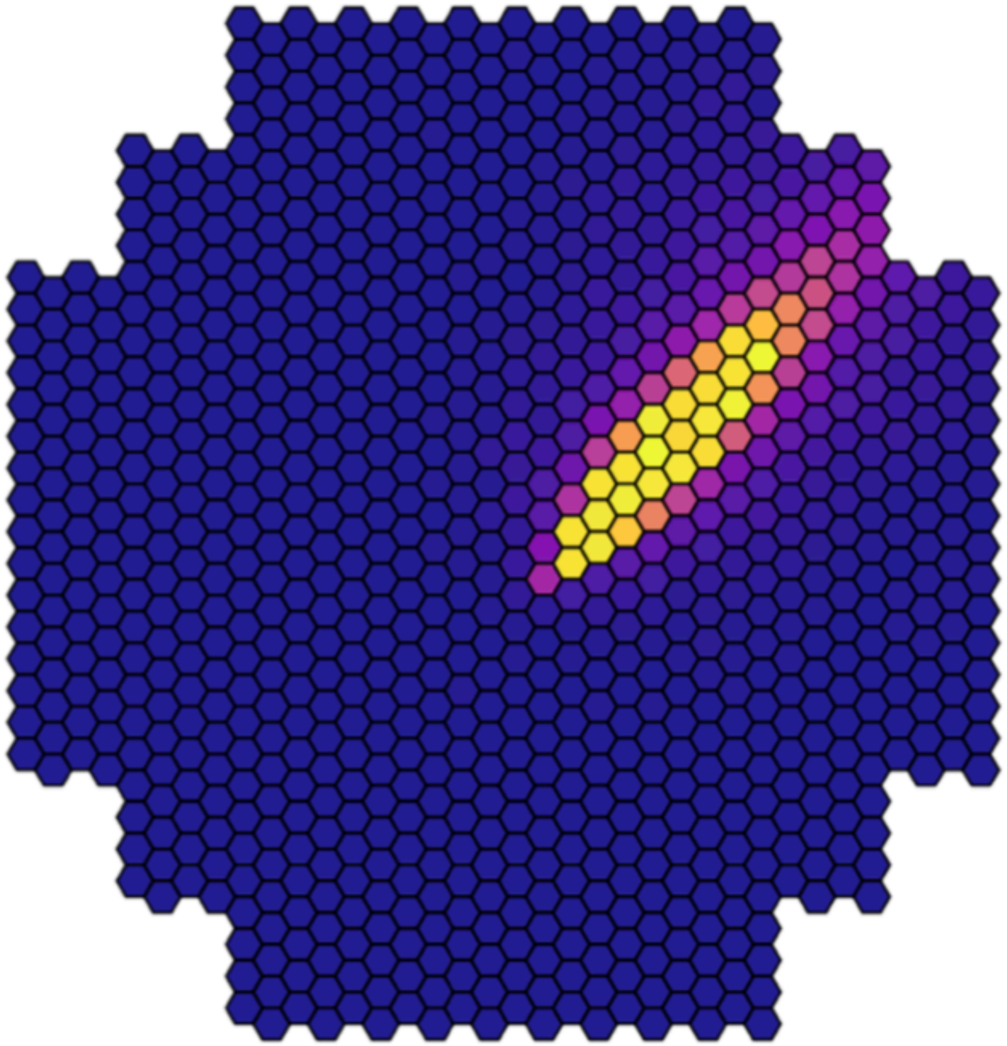
- Measure air showers to deduce properties of high-energy photons
- Classification & regression tasks
  - Gamma-hadron separation
  - Direction reconstruction
  - Energy reconstruction



Hinton JA, Hofmann W. 2009. Annu. Rev. Astron. Astrophys. 47:523–65

Hexagonally Sampled Data

- Hexagonal sampling is common with IACTs
- Advantages:
  - Densest tiling of 2D-Euclidean plane
  - Optimal sampling of circularly band-limited signals
- Issue for the application of CNNs
  - Format is not supported by any open-source deep learning framework



Processing Hexagonally Sampled Data

- Solution 1: Transform to Cartesian grid
  - \+ Many tools available to process data
  - \- Transformation can introduce distortions and artefacts
  - \- Increase of computer storage and computational costs
- Solution 2: Process hexagonal data directly
  - \- Write custom tool
  - \+ None of the cons above
  - \+ Potentially higher angular resolution

    (consistent connectivity, higher rotational symmetry)

Processing Hexagonally Sampled Data

- Aim:
  - Convolution on hexagonal grids
  - Convenience of available DL frameworks
- Result:
  - HexagDLy - an extension to PyTorch
  - Easy-to-use and flexible implementation of hexagonal convolution and pooling operations
  - Arbitrary kernel size, stride and input size ⇒ Fast prototyping of models
  - See: Steppa, C. & Holch, T. L. (2019)

https://github.com/ai4iacts/hexagdly

HexagDLy - Processing Hexagonal Data with PyTorch

HexagDLy provides convolution and pooling methods for hexagonally sampled data within the deep learning framework PyTorch.

- Getting Started
- Preparing the Data
- How to use HexagDLy
- General Concept
- Disclaimer
- Citing HexagDLy

Getting Started

There are different ways to get HexagDLy up and running on your system as shown below. Basic examples for the application of HexagDLy are given as notebooks. Additionally unit tests are provided in tests.

Pip Installation

The suggested way to install HexagDLy is to set up a clean virtual python environment (e.g. with conda, see https://www.anaconda.com/) and use the provided pip installer. To install basic functionalities only use:

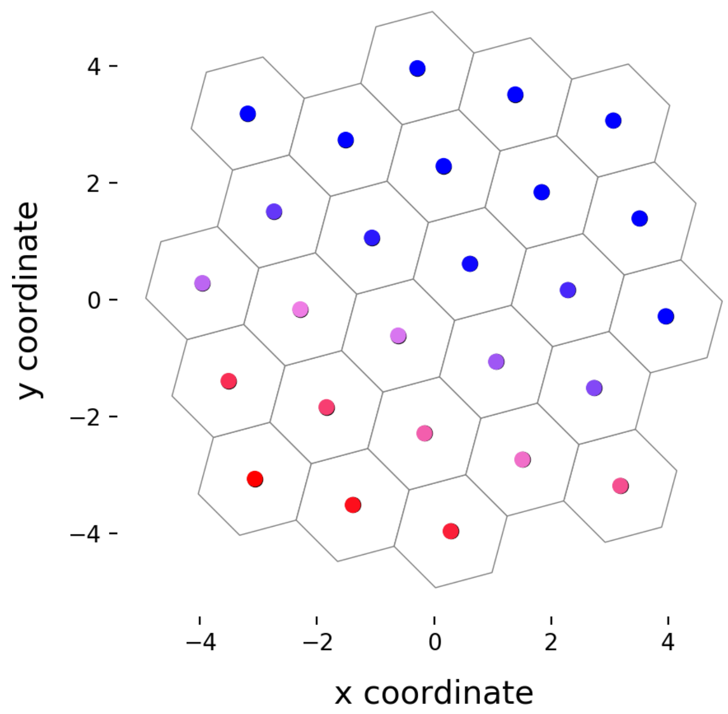
```
pip install hexagdly
```

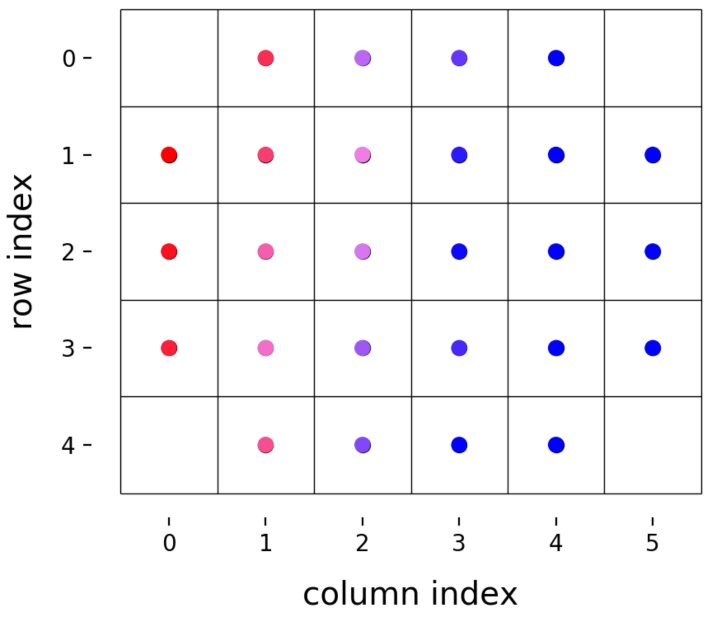
To get all necessary dependencies to run the provided unit tests and notebooks, add the `dev` option:

```
pip install hexagdly[dev]
```

Short Introduction To HexagDLy

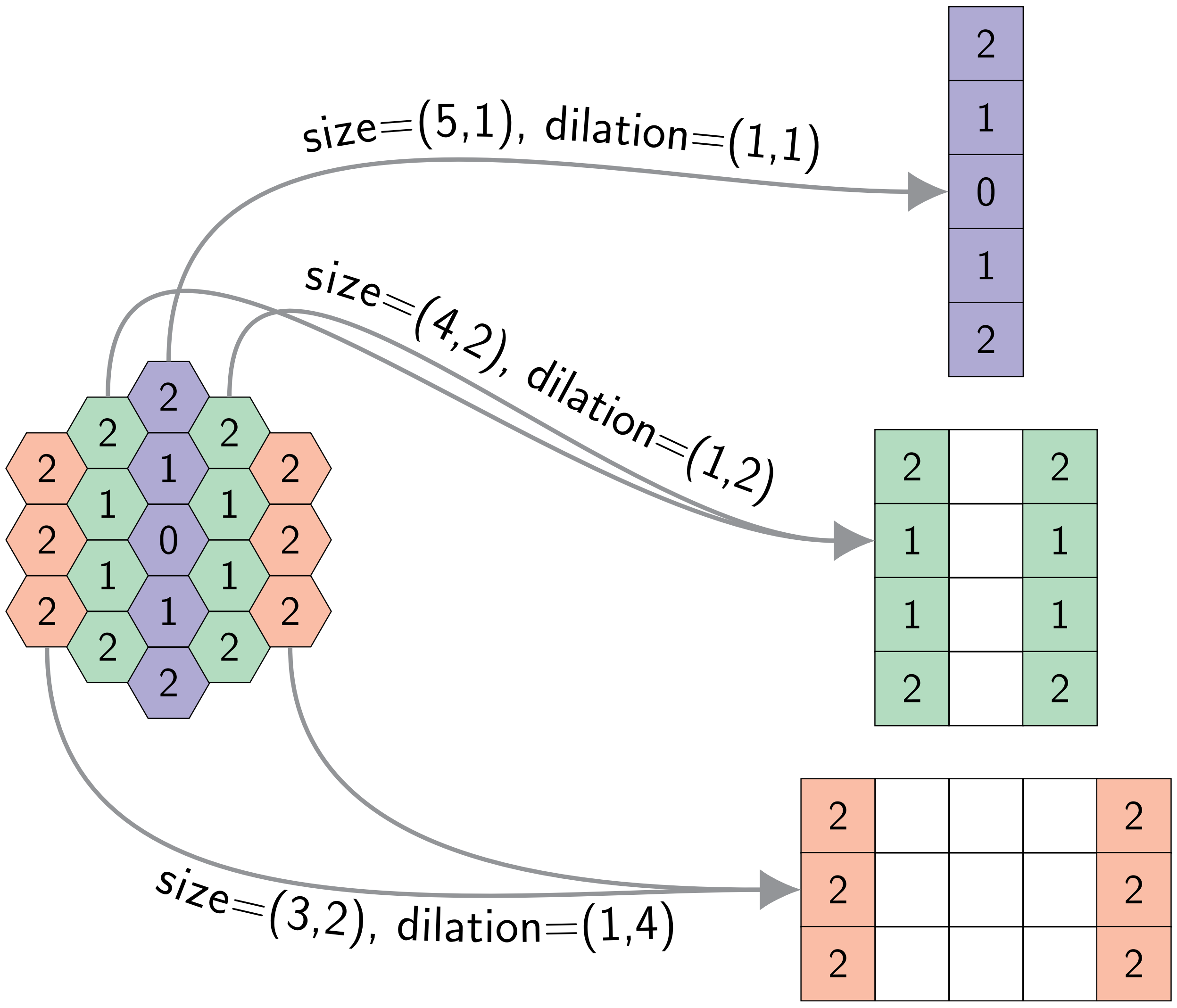
- **Addressing scheme**
- Kernels
- Convolution
- Example





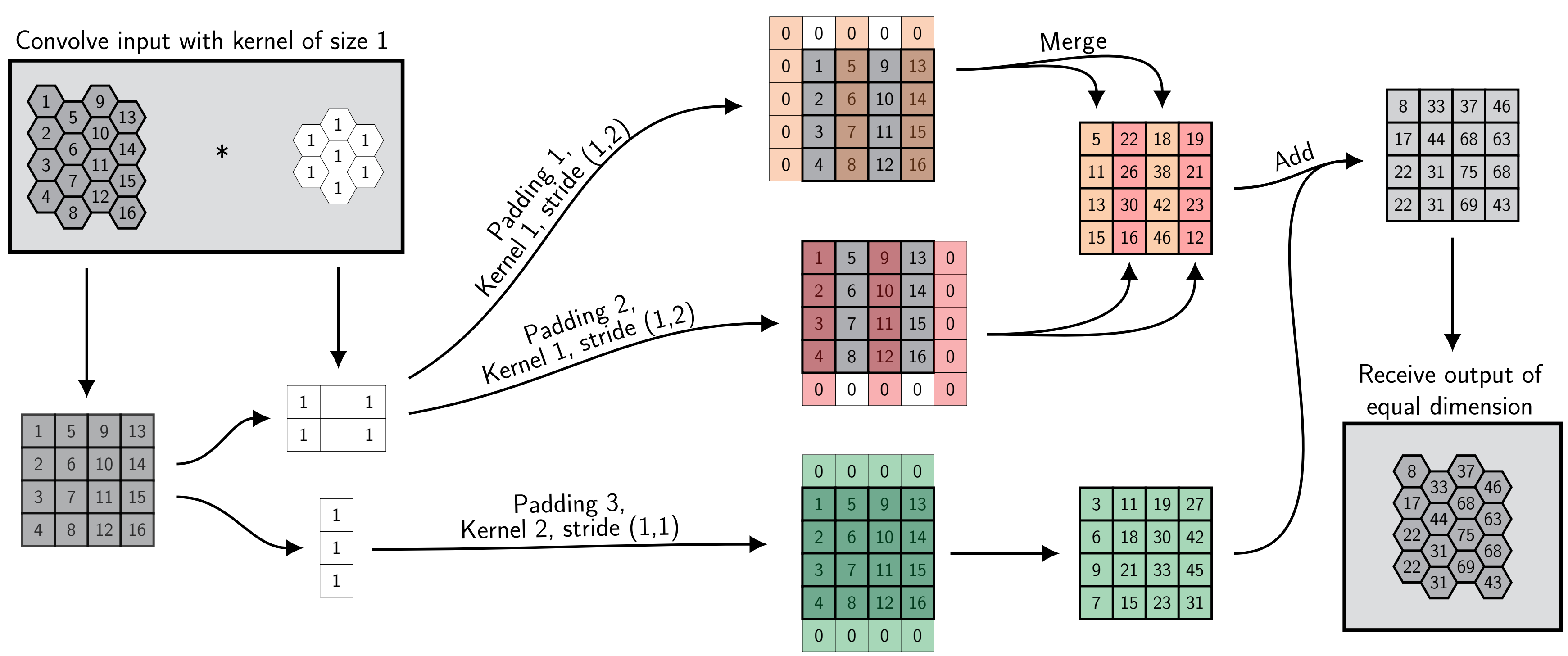
Short Introduction To HexagDLy

- Addressing scheme
- **Kernels**
- Convolution
- Example



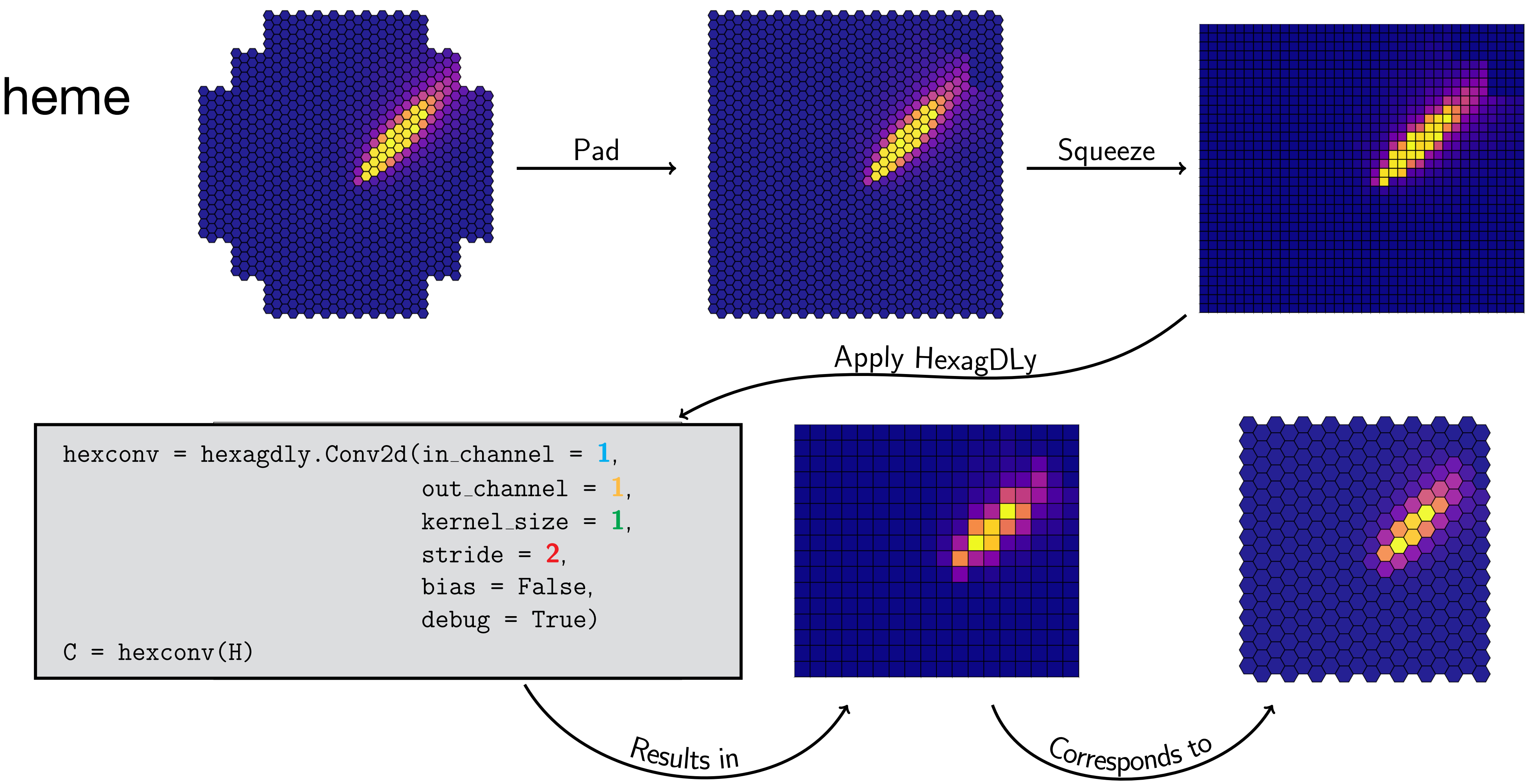
Short Introduction To HexagDLy

- Addressing scheme
- Kernels
- **Convolution**
- Example



Short Introduction To HexagDLy

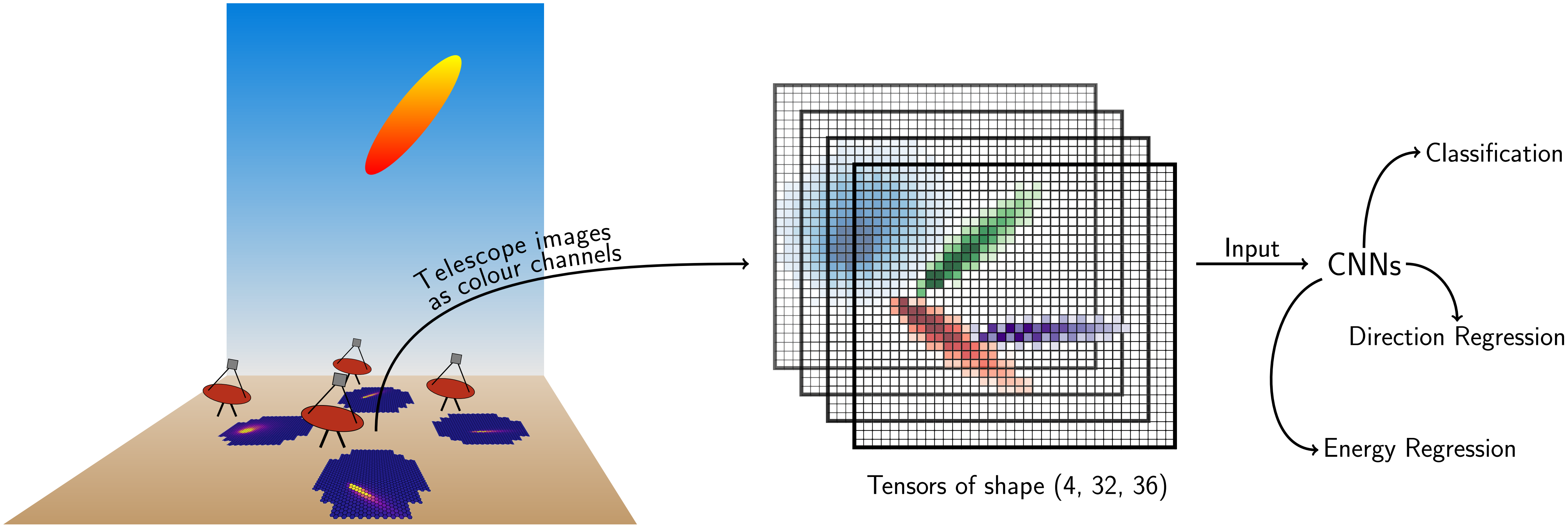
- Addressing scheme
- Kernels
- Convolution
- **Example**




```
hexconv = hexagdly.Conv2d(in_channel = 1,
                          out_channel = 1,
                          kernel_size = 1,
                          stride = 2,
                          bias = False,
                          debug = True)
C = hexconv(H)
```

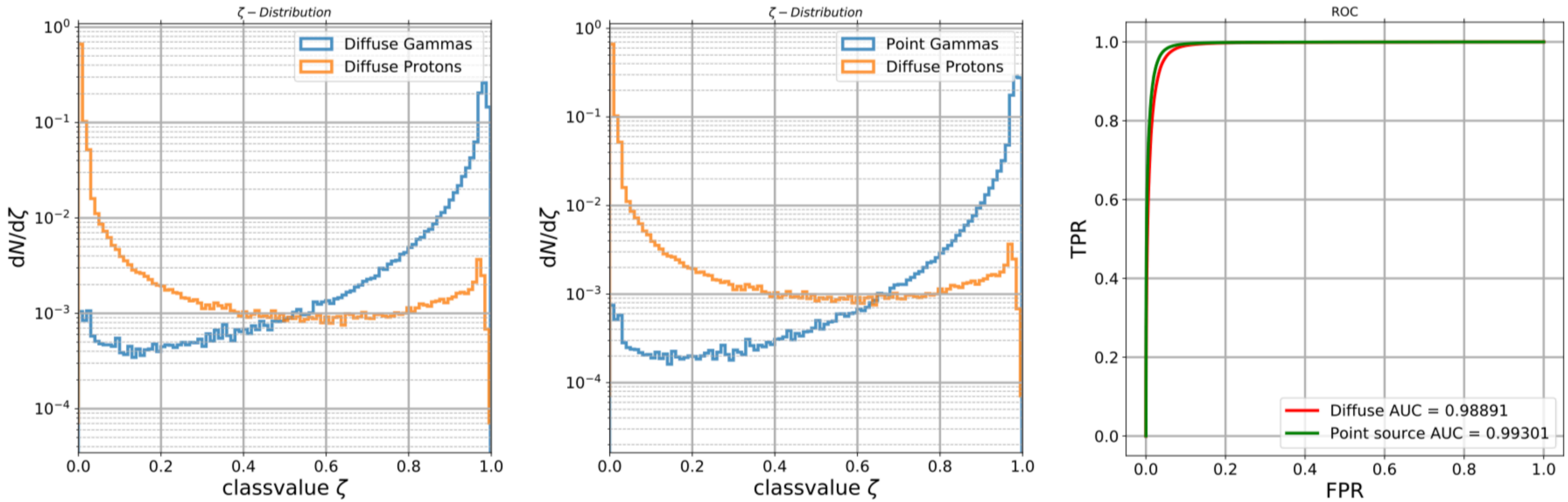
Applying HexagDLy To H.E.S.S. MC Data

- Four cameras that see the same air shower
- Norm individual images
- Treat single images like colour channels



Classification - MC

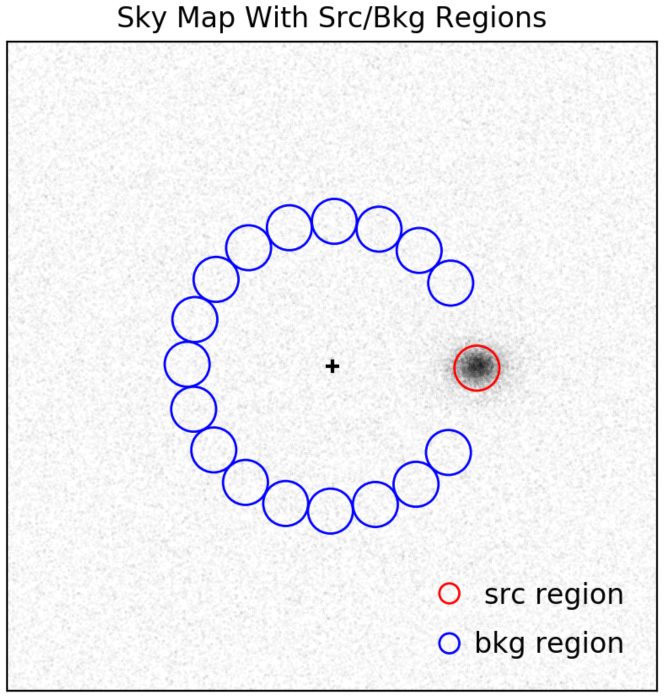
- Trained on a total of 2.8M simulated diffuse γ-ray and proton showers at 20° zenith
- No parameter cuts applied → all events triggering the system are included
- 4 conv. layers, 3 pooling layers, 3 fully connected + 1 dropout of 0.5

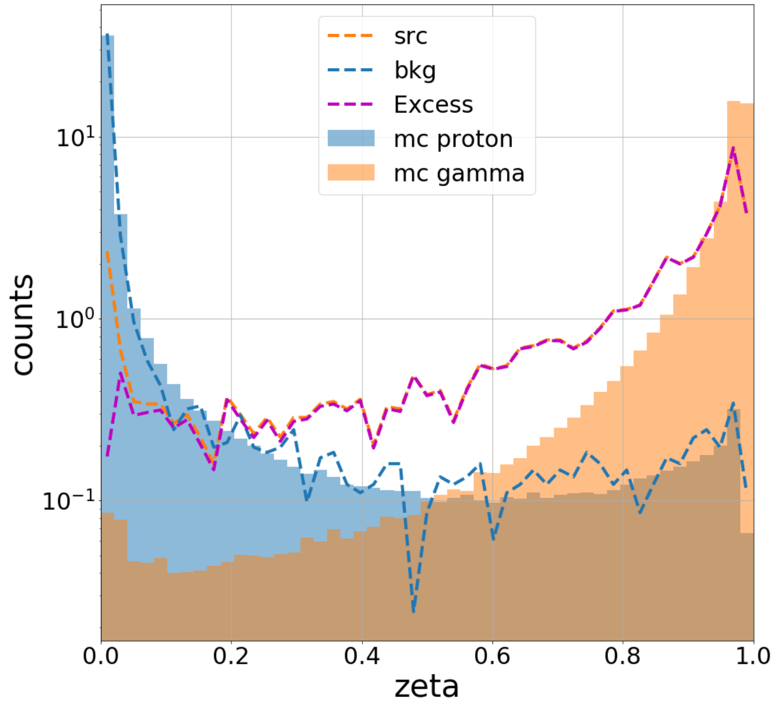


Shown results incl. only events with size > 40 and local distance < 0.525!
Point gammas refer to a simulated point source at 0.5° offset.

Classification - Data

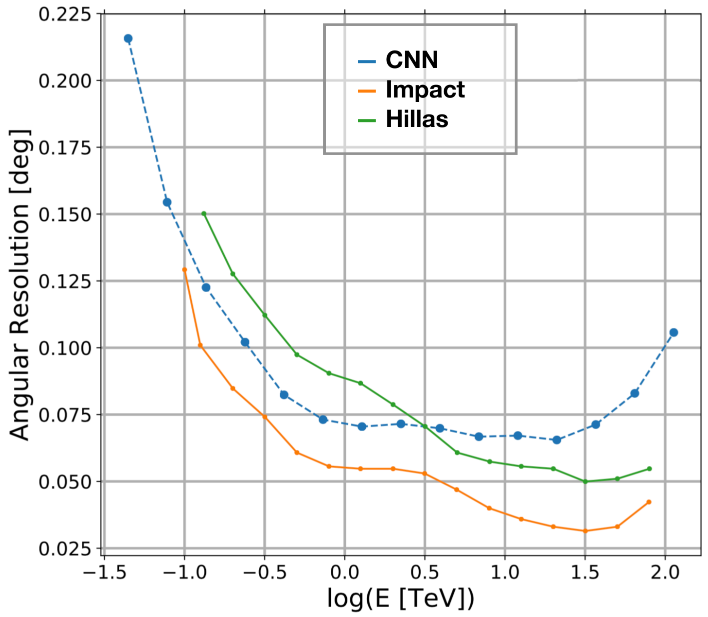
- Background distribution matches MC predictions
- Discrepancy between MC and real data distributions of signal events
- Potential solutions that will be explored
  - Data augmentation
  - Domain adversarial NN
  - Run-wise simulations

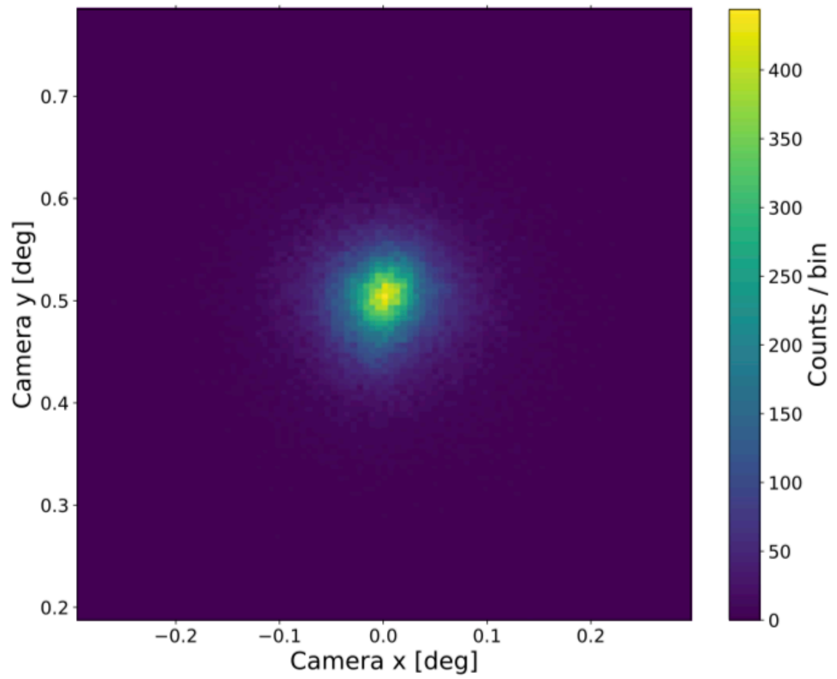




Direction Reconstruction - MC & Data

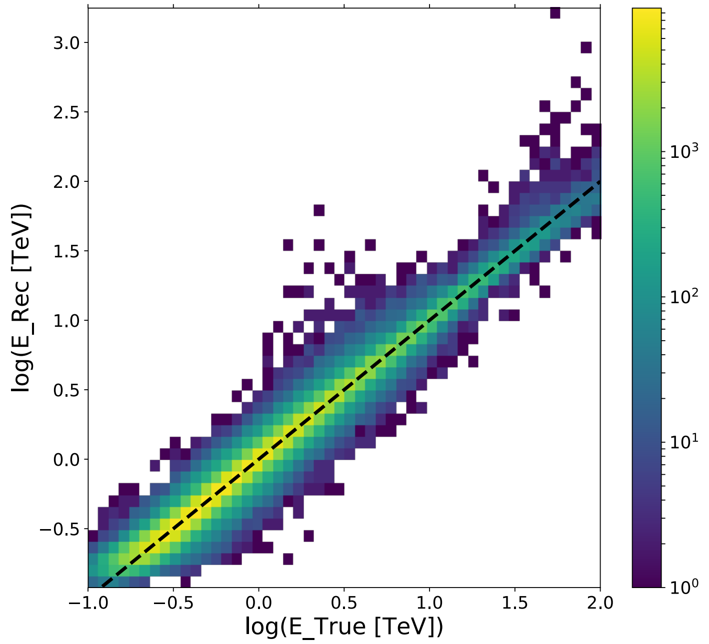
- Trained on 360k simulated diffuse γ-ray showers at 20° zenith
- Model with 8 conv. layers, 3 pooling layers, 3 fully connected + 1 dropout of 0.5
- Plots for simulated point-source at 0.5° offset from camera center
- On MC: decent performance for E < 3 TeV
- On real data: missing pointing correction

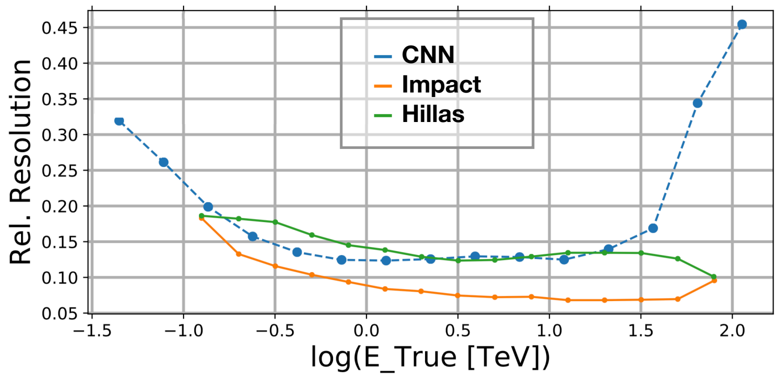


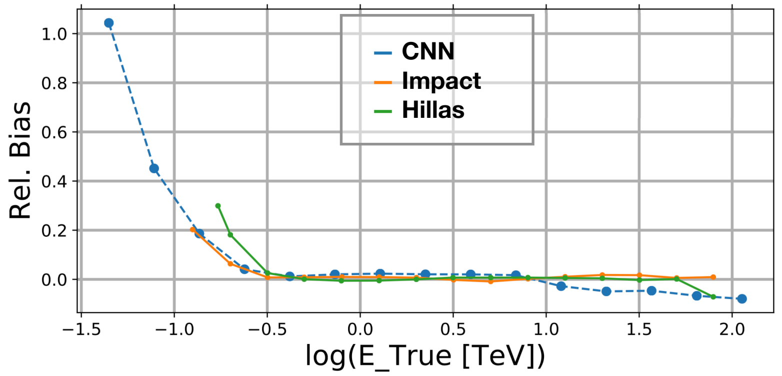


Energy Reconstruction - MC

- Model consists of a CNN and an NN with size vector as input with concatenation to produce one output
- Trained on 2.8M simulated diffuse γ-ray showers at 20○ zenith
- Plots for simulated point-source
- Problems for E > 30 TeV
  → probably statistics. . .







Conclusion

- Hexagonal convolutions:
  - Remove necessity to sample data to higher resolutions → less data points to process + no sampling artefacts
  - Provide straight forward integration of additional data channels like timing information
- HexagDLy:
  - PyTorch extension to process hexagonal data
  - Easy-to-use and flexible implementation aiming for fast prototyping
  - Detailed description: Steppa, C. & Holch, T. L. (2019)
  - Download from: GitHub or PyPI
- Air shower reconstruction for H.E.S.S.:
  - Application to MC data shows promising results for full shower reconstruction
  - Currently investigating application to real data

We thank the H.E.S.S. Collaboration for supporting this research and granting access to data!

Backup

Comparing Hex-Conv To Square-Conv

- Datasets
  - 4 classes, 128 images of each class
  - Original hexagonal sampling
  - Square re-sampling, same resolution (small)
  - Square re-sampling, 4x resolution (large)
- Models
  - Small hexagonal CNN (h-CNN), ~ 13 k parameters
  - Small square CNN (s-CNN), ~ 13 k parameters
  - Large square CNN (s-CNN), ~ 1.2 M parameters
- Training
  - 100 epochs
  - Repeated 150 times with new datasets & re-initialised models