
HexagDLy - Hexagonal Convolutions with PyTorch

And Applications in CNN-based Analysis of H.E.S.S. Data

Tim Lukas Holch and Constantin Steppa
For the H.E.S.S. Deep Learning Task Group

Big Data Science in Astroparticle Research - HAP Workshop, Aachen 2018

20.02.2018

Outline

- 1 HexagDLy - Introduction and Implementation
- 2 Application - Processing H.E.S.S. Data
- 3 Performance on H.E.S.S. MC Data
- 4 Conclusion

Introducing HexagDLy

Motivation:

- 1 Problem of hexagonally sampled detector data - as discussed here in Aachen last year. . .
Our initial solution: Sample the data to a square grid \Rightarrow *decent and promising results!*
BUT: Sampling only approximates the original information while increasing the number of data points to process and can introduce sampling artefacts!
- 2 How can we reduce potential sampling artefacts? None at all would be even better. . .
 \Rightarrow We need true hexagonal convolutions
- 3 Hexagonal kernels + the convenience of available DL frameworks?

Resulted in **HexagDLy**:

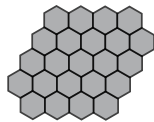
- Small piece of Python code to be used on top of PyTorch
- Easy-to-use and flexible implementation of hexagonal convolution and pooling operations
- Arbitrary kernel size, stride and input size \Rightarrow **Fast prototyping of models**
- Check it out on GitHub: <https://github.com/ai4iacts/hexagdly>

Hexagonal Kernels in common DL Frameworks?

Main steps in HexagDLy:

- 1 Rearrange hexagonal data-points to a square grid \Rightarrow Enable the application of available DL frameworks
- 2 Design custom convolution kernels consisting of different square-grid sub-kernels that, in combination, resemble a hexagonal kernel
- 3 Apply the sub-kernels to the rearranged data and combine the output in a way that conserves the 6-fold symmetry of the hexagonal grid

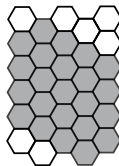
1. Squeezing the Data into Shape



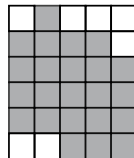
Original Shape



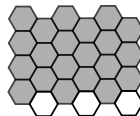
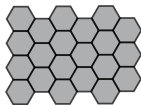
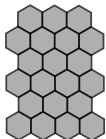
Rotate to align
arm-chair edge
horizontally



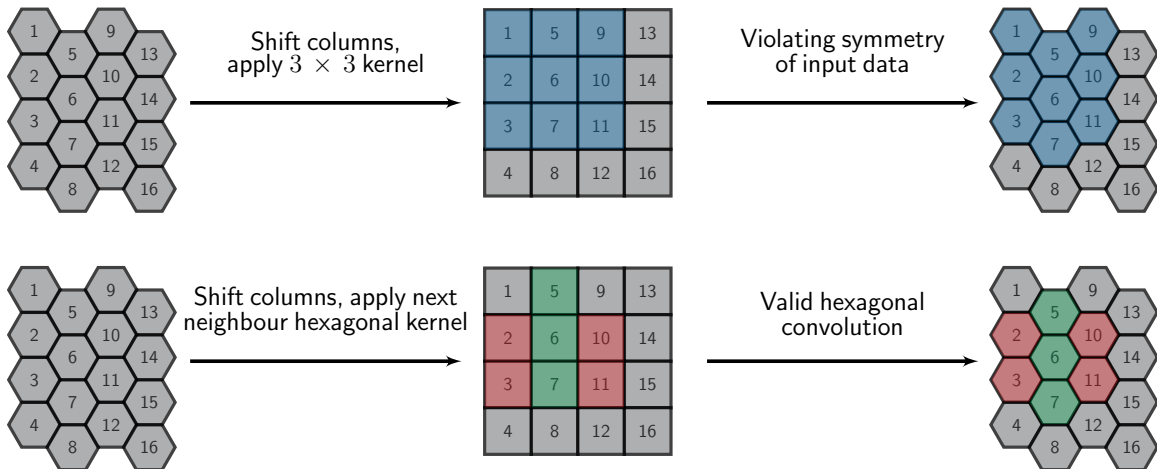
Zero-pad to get
protruding odd
columns on top
row and columns
of equal length



Shift columns
against each other
by $1/2$ of vertical
cell distance

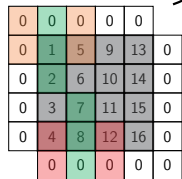
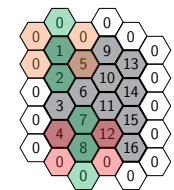


2. Design Custom Kernels



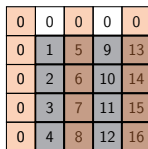
3. Combine Sub-Kernels to Conserve Symmetry

Padded Hexagonal Input

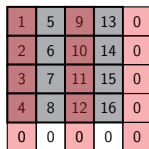


Input squeezed
into square array

+ Padding 1
+ Kernel 1, stride (2,1)



+ Padding 2
+ Kernel 1, stride (2,1)



+ Padding 3
+ Kernel 2, stride (1,1)



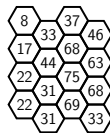
Merge



Add



Hexagonal Output
of equal dimension



3. Combine Sub-Kernels to Conserve Symmetry

Padded Hexagonal Input



0	0	0	0	0
0	1	5	9	13
0	2	6	10	14

Merge

Following the same concept:

- **Larger kernels** via additional sub-kernels + different padding and slicing of input for each sub-convolution
 ⇒ *The size of a kernel always corresponds to the number of included neighbours along the symmetry axes!*
- **Pooling operations** by replacing 2D convolution with nested 2D pooling methods and set larger strides

0	0				
0	1				
0	2	6	10	14	0
0	3	7	11	15	0
0	4	8	12	16	0
0	0	0	0	0	0

Input squeezed into square array

Padding 3
+ Kernel 2, stride (1,1)

0	0	0	0
1	5	9	13
2	6	10	14
3	7	11	15
4	8	12	16
0	0	0	0

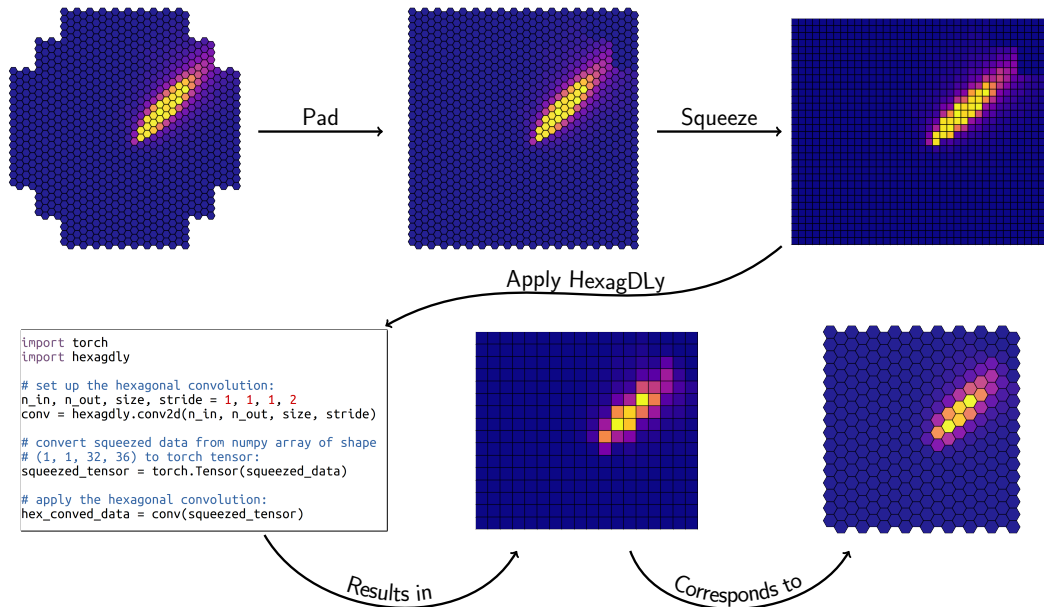
3	11	19	27
6	18	30	42
9	21	33	45
7	15	23	31

8	33	37	46
	44	68	63
	31	75	68
	31	69	33



Hexagonal Output of equal dimension

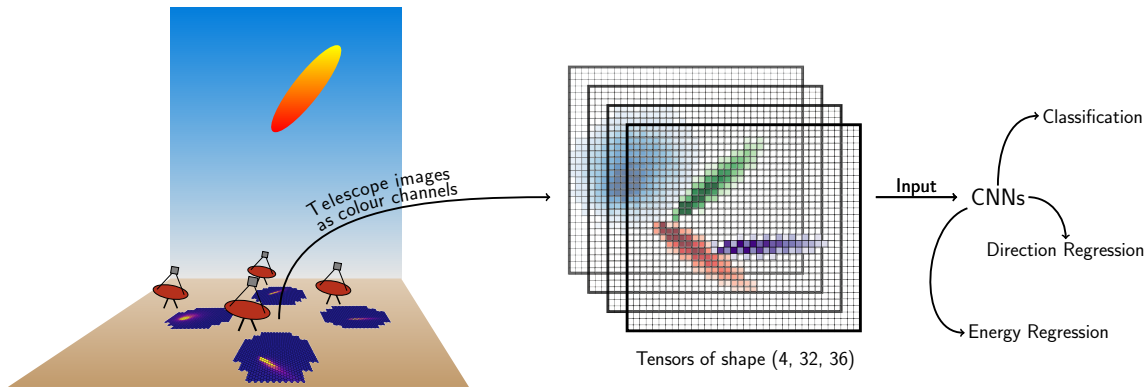
Applying HexagDLy to H.E.S.S. Camera Data



Giving it some Colour - The Multi-Channel Approach

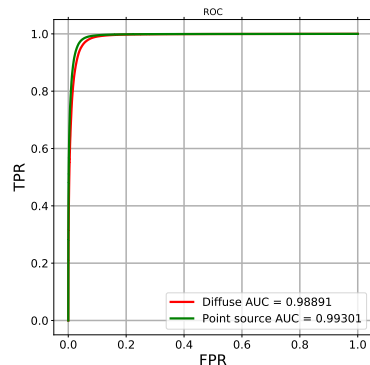
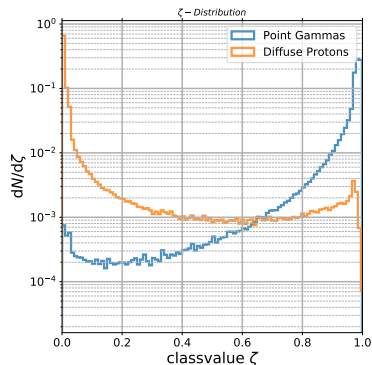
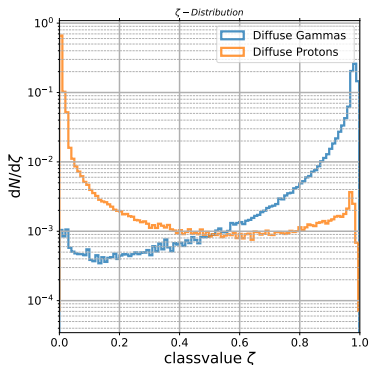
Only regarding small 12 m H.E.S.S. telescopes:

- Four cameras that see the same air shower
- Sort images by size (sum over pixel intensities)
- Norm individual images
- Treat single images like colour channels



Performance - Classification

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

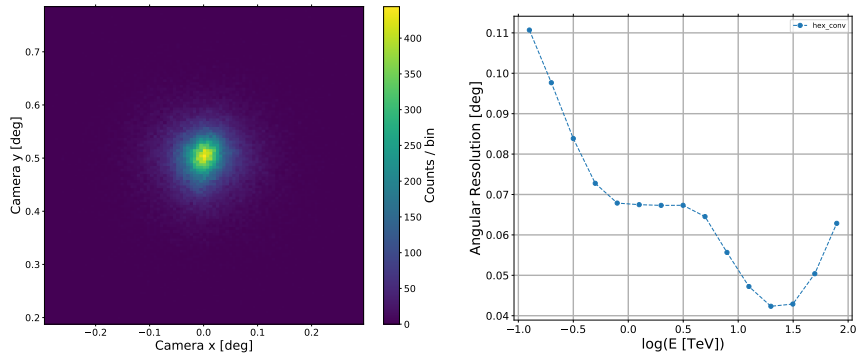


Shown results incl. only events with size > 40 and local distance $< 0.525^\circ$!

Point gammas refer to a simulated point source at 0.5° offset.

Performance - Direction Regression

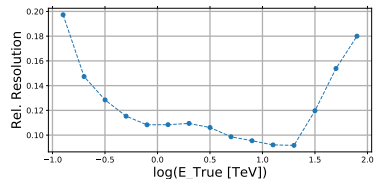
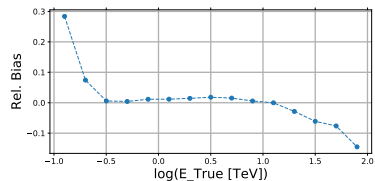
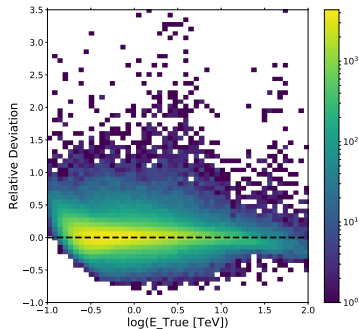
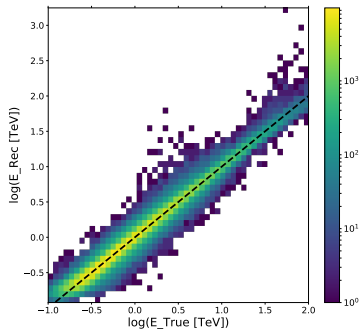
- Trained on 360k simulated diffuse γ -ray showers at 20° zenith
- Model with 8 conv. layers, 3 pooling layers, 3 fully connected + 1 dropout of 0.5
- Plots for simulated point-source at 0.5° offset from camera center



Results only incl. events with size > 40 and local distance < 0.525 and a classvalue < 0.95.

Performance - Energy Regression

- Model consists of a CNN **and** an MLP with size vector as input with concatenation to produce one output
- Trained on 2.82M simulated diffuse γ -ray showers at 20° zenith
- Plots for simulated point-source
- Problems for $E \gtrsim 30$ TeV \rightarrow probably statistics. . .



Results only incl. events with size > 40, local distance < 0.525 and a classvalue < 0.95.

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, available on <https://github.com/ai4iacts/hexagdly>
- Easy-to-use and flexible implementation aiming for fast prototyping
- Transferable to other DL frameworks ⇒ *Feel free to contribute!*

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 Monte Carlo data!**