Generation of PXD background using generative models

Fabio Novissimo, Nikolai Hartmann, Thomas Kuhr

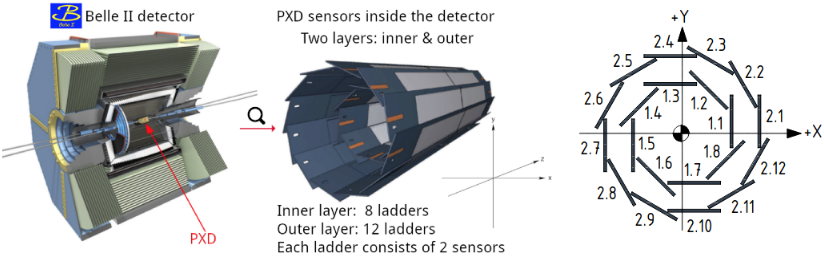
LMU München

KISS meeting, Munich, March 12th 2025

Bundesministerium
für Bildung
und Forschung

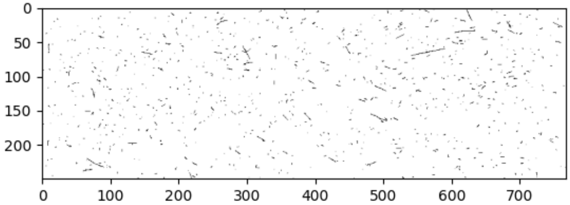
The Pixel Vertex Detector (PXD)

- The **Pixel Vertex Detector (PXD)** is the innermost semi-conductor sub-detector of Belle II, at 1.4 cm from the collision point.
- The sensitive area of the PXD is made up by 40 modules. Each module consists of a 250×758 pixel matrix.
- **Inner layer**: 16 modules implemented into 8 ladders.
- **Outer layer**: 24 modules implemented into 12 ladders.



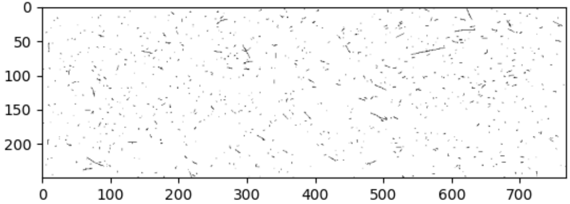
Background

- PXD hits come mainly from background processes.



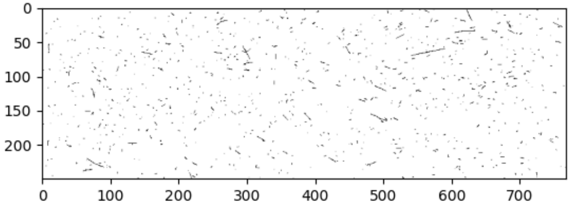
Background

- PXD hits come mainly from background processes.
- Two ways to include background processes:
  - Monte Carlo generation $\longrightarrow$ shows sizeable discrepancies with measurements.
  - Taking random trigger events.



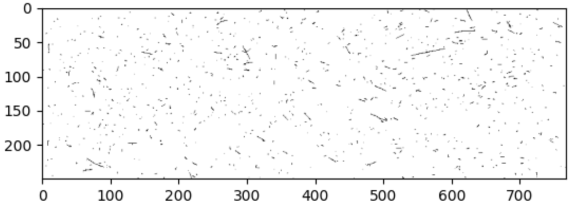
Background

- PXD hits come mainly from background processes.
- Two ways to include background processes:
  - Monte Carlo generation ⟶ shows sizeable discrepancies with measurements.
  - Taking random trigger events.
- **Problem: large amount of resources required for storage and distribution of the background data.**

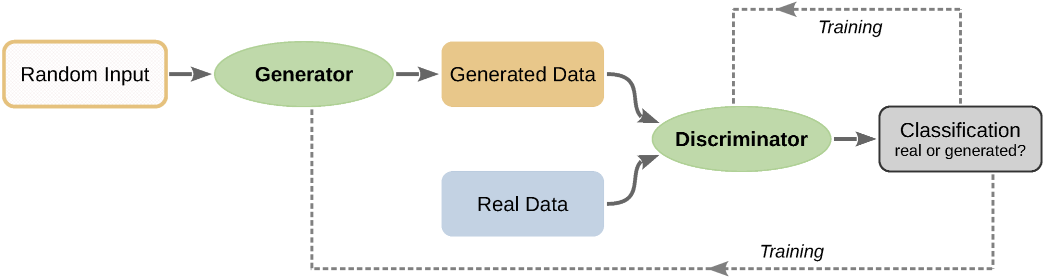


Background

- PXD hits come mainly from background processes.
- Two ways to include background processes:
  - Monte Carlo generation $\longrightarrow$ shows sizeable discrepancies with measurements.
  - Taking random trigger events.
- **Problem: large amount of resources required for storage and distribution of the background data.**
- Solution: generate background hits on the fly for each sensor.



Generative Adversarial Network



Generating pixels with GAN

Previous approach:

- GAN conditioned on sensor number with a transformer-based relational reasoning module to reproduce the correlations between sensors(IEA-GAN).

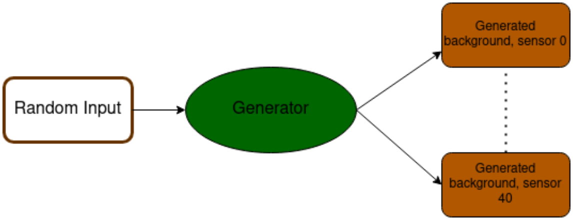
Generating pixels with GAN

Previous approach:

- GAN conditioned on sensor number with a transformer-based relational reasoning module to reproduce the correlations between sensors(IEA-GAN).

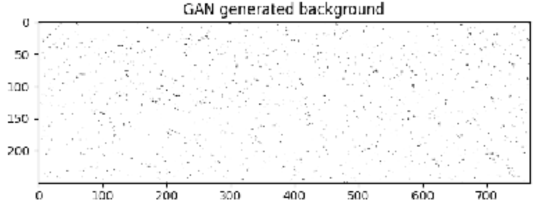
New approach: generate the background using a GAN without conditioning on the sensor number.

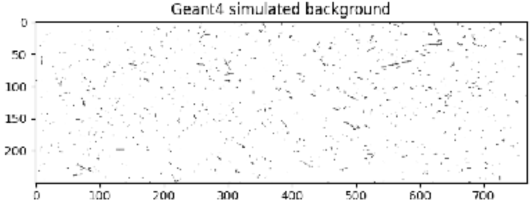
- Generate instances of background for all sensors at once.
- Wasserstein GAN with CNN layers used in the Generator and Discriminator.



Generated background

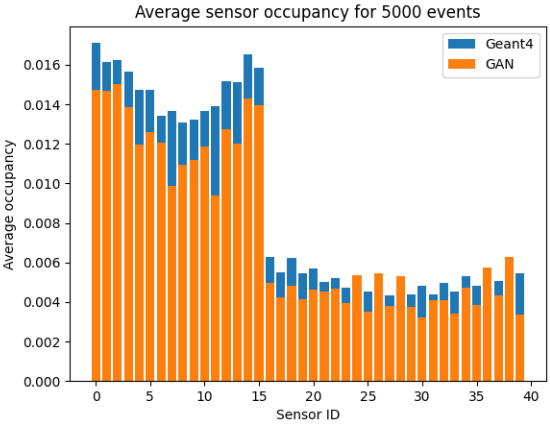
The generated images are visually very similar, but with some subtle differences.

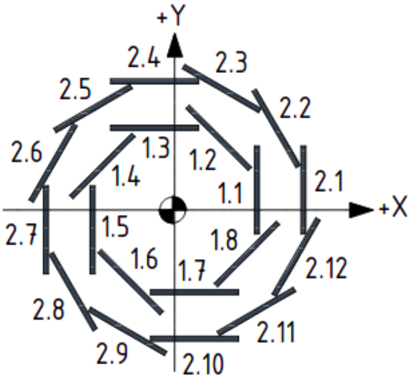




Evaluation - Occupancy per sensor

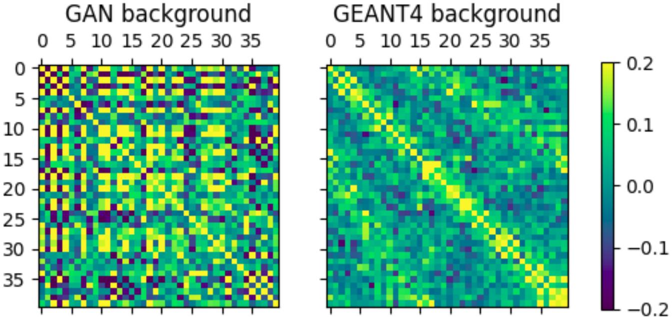
The model seems to reproduce quite well the sensor occupancy, aside from some minor details probably due to some fluctuations in the weights of the model.





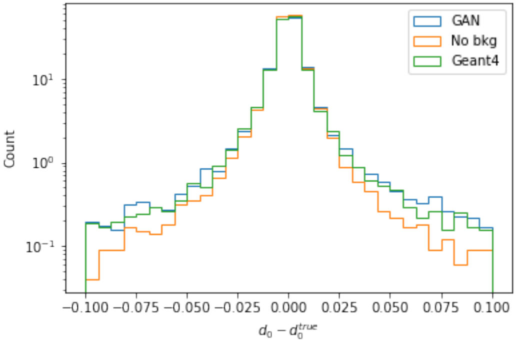
Evaluation - Correlation

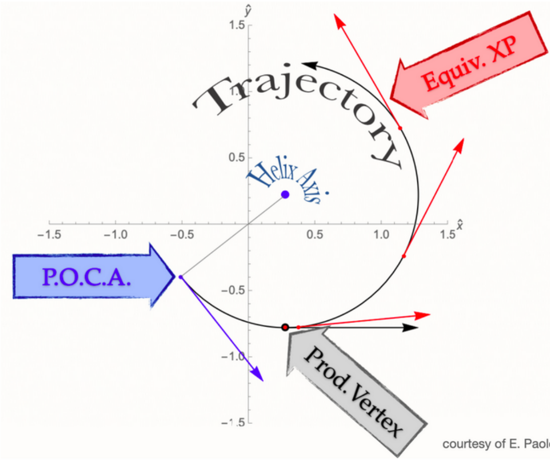
The model does not reproduce correctly the correlation between the sensor occupancy.



Evaluation - helix parameters resolution

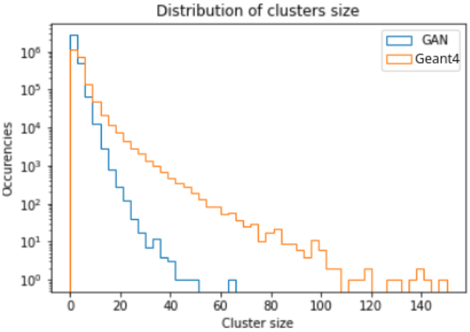
GAN background can be used to reproduce resolution of the helix parameters.

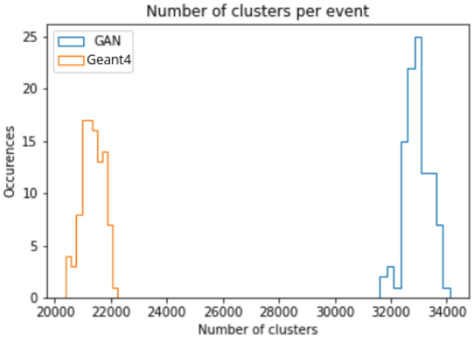




Evaluation: Clusters

The generated background images have different clusters distributions.





Cluster generation with GAN

- Train GAN to directly generate clusters instead of full sensor pixels.
- Trained using clusters of sizes from 1 to 30.
- Training dataset uniform in cluster size.

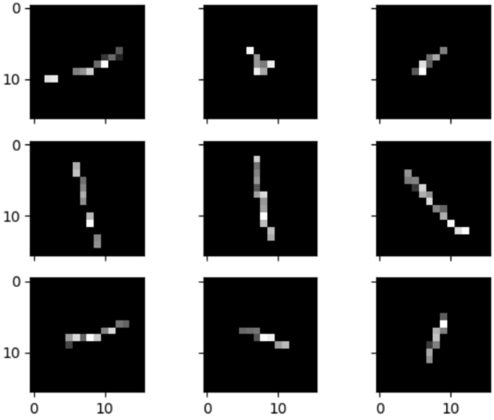
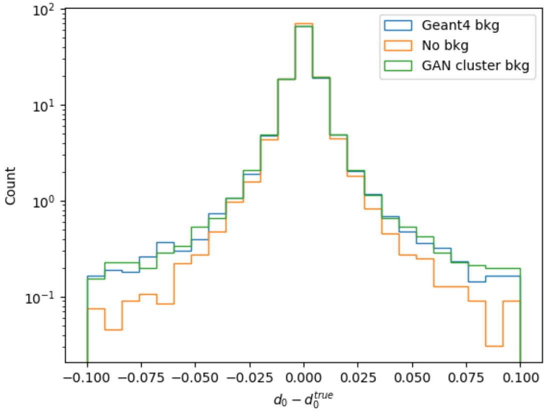
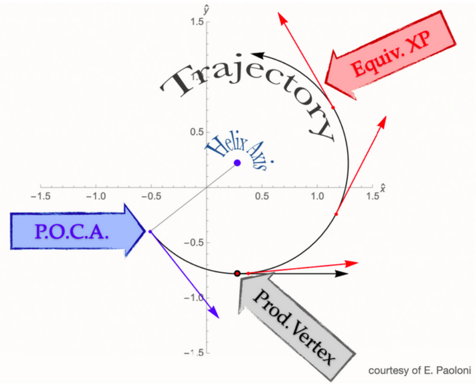


Figure: Example of generated clusters

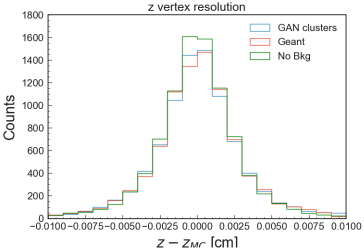
Evaluation - helix parameters resolution

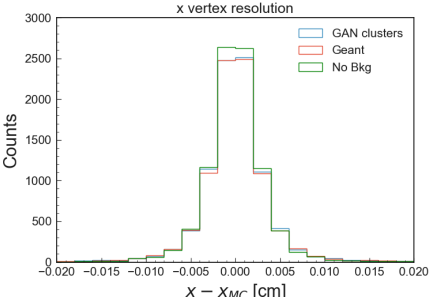


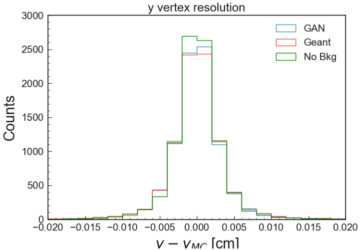


Evaluation - Vertex reconstruction

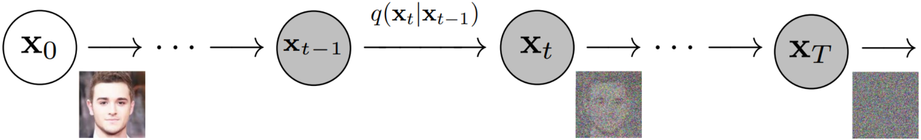
- Vertex resolution of D^0 in the decay $D^0 \to K^- \pi^+$
- Drop in efficiency when taking into account the background.







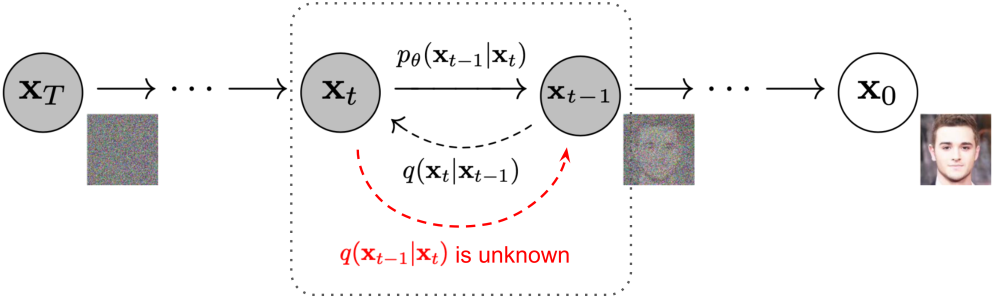
Diffusion model: forward process



$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t \mathbb{I}) \qquad \{\beta_t \in (0,1)\}_{t=1}^{T}$$

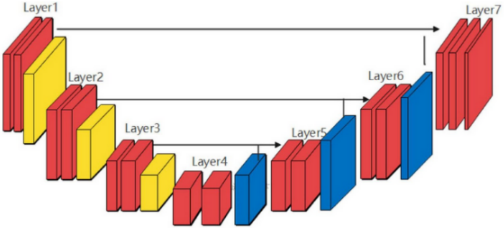
Diffusion model: inverse process

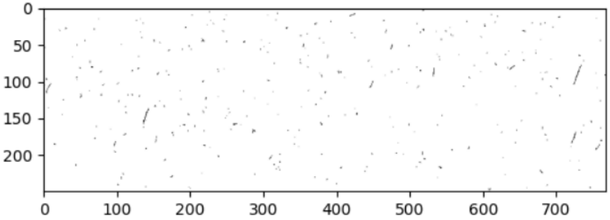
Use variational lower bound



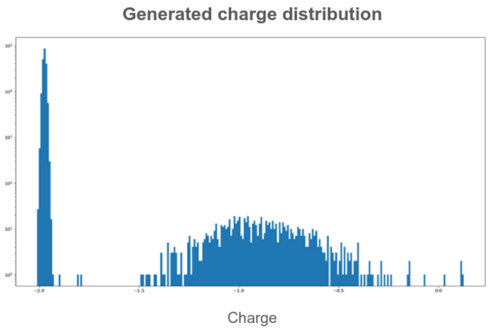
$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \Sigma_\theta(\mathbf{x}_t, t))$$

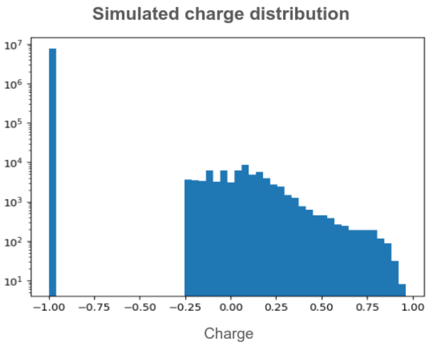
Example image of generated background





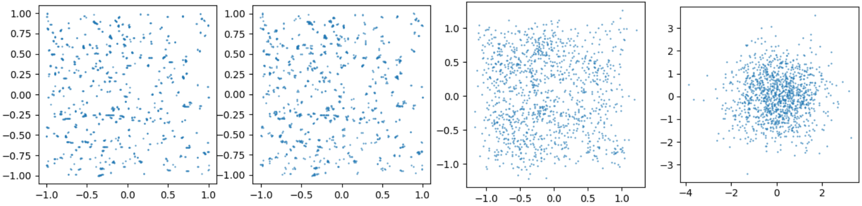
Preliminary results: charge distribution



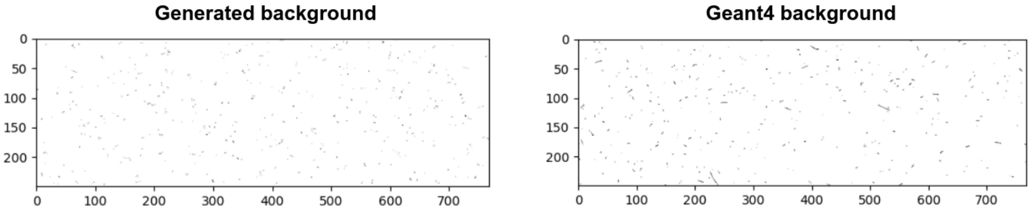


Graph Neural Network

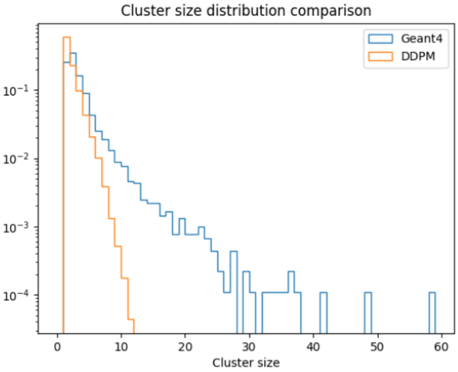
- IDEA: consider a point cloud representation of the BKG image and use a Graph Neural Network to predict the noise.
- In the GNN edge convolutional layers are employed.

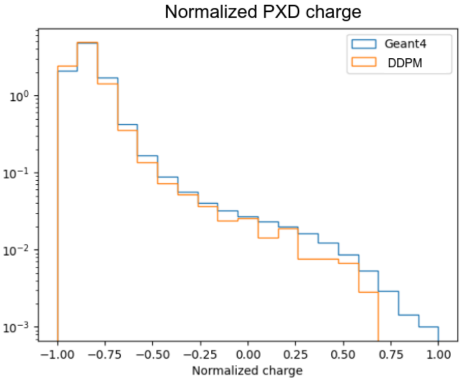


GNN: result


Generated background
0
50
100
150
200
0
100
200
300
400
500
600
700
Geant4 background
0
50
100
150
200
0
100
200
300
400
500
600
700

GNN: evaluation





Summary and conclusions

- Successfully trained a GAN to generate PXD hitmaps.
- Differences between simulated and generated images, especially regarding sensor occupancy correlation and clusters.
- Generated background reproduces helix parameters resolution well.
- Successfully trained a GAN to generate clusters.
- Generated clusters correctly reproduce the helix parameter resolution.
- Trained a diffusion model to generate background images.

Conclusions and outlook:

- Demonstrated that generative models can produce PXD background with the required accuracy.
- Possibility to use other methods which do not involve ML i.e. constructing BKG files by sampling techniques.

Summary and conclusions

- Successfully trained a GAN to generate PXD hitmaps.
- Differences between simulated and generated images, especially regarding sensor occupancy correlation and clusters.
- Generated background reproduces helix parameters resolution well.
- Successfully trained a GAN to generate clusters.
- Generated clusters correctly reproduce the helix parameter resolution.
- Trained a diffusion model to generate background images.

Conclusions and outlook:

- Demonstrated that generative models can produce PXD background with the required accuracy.
- Possibility to use other methods which do not involve ML i.e. constructing BKG files by sampling techniques.

Thanks for your attention!

Backup - Generator

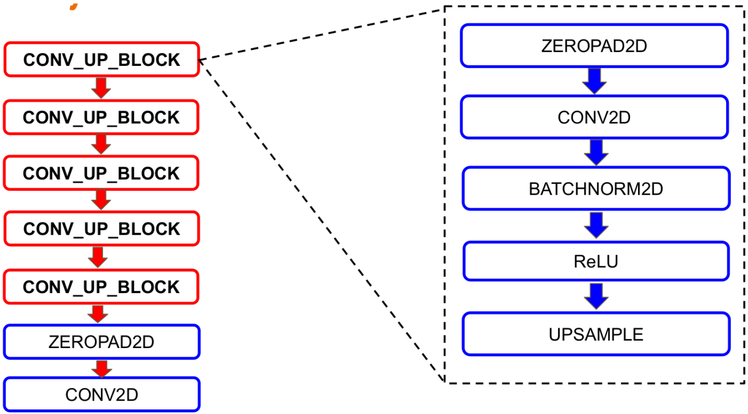


Figure: Generator architecture

Backup - Discriminator

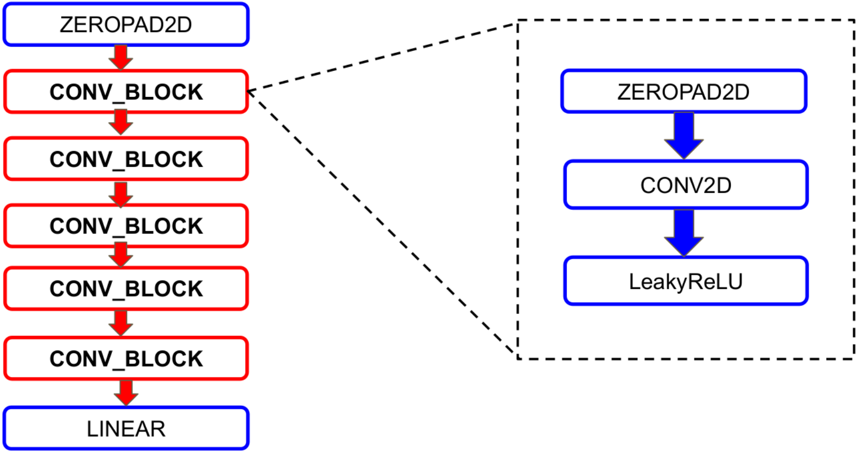


Figure: Discriminator architecture

Evaluation: cluster size