Warsaw University of Technology

ALICE

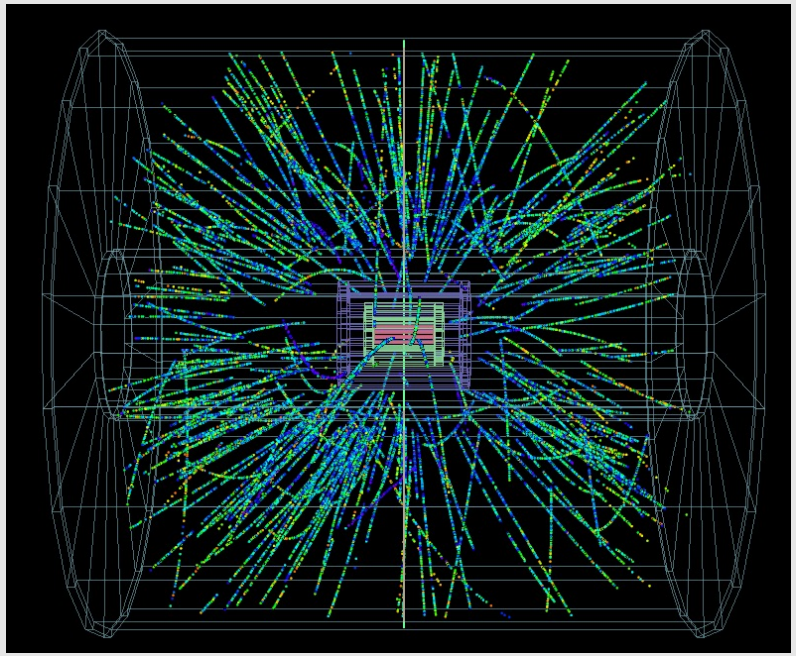
Machine Learning methods for simulating particle response in the Zero Degree Calorimeter at the ALICE experiment, CERN

16/09/2022

Jan Dubiński, Kamil Deja, Sandro Wenzel
Przemysław Rokita, Tomasz Trzciński

The need for faster simulations

- CERN has the biggest computional grid in the world ~800,000 CPU cores
- The majority of the compution power is used for HEP simulations
- Standard simulation procedures are extremally costly (Monte Carlo)
- Machine learning offers an alternative cost-efficient approach to the problem

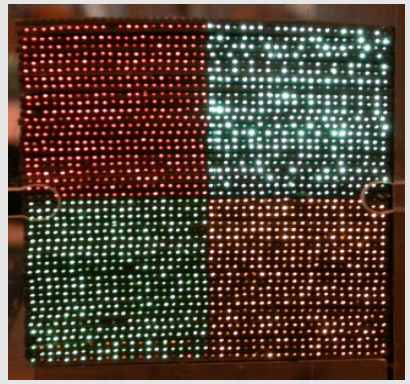


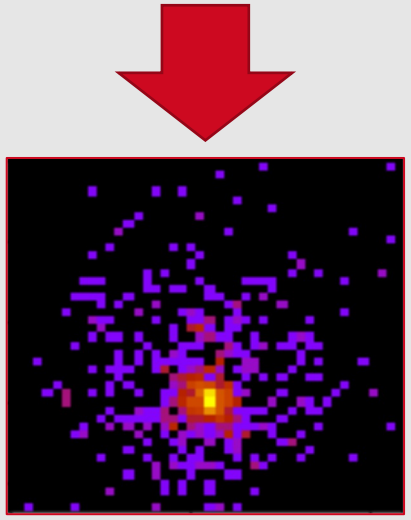
Zero Degree Calorimeter

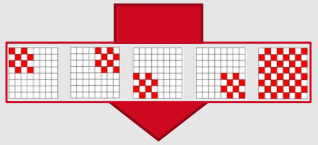
- The most costly simulations are calorimeter response simulations
- The Zero Degree Calorimeter ZDC is located in the ALICE experiment
- It detects the energy of the spectator nucleons in order to determine the overlap region in nucleus-nucleus collisions
- Consists of 44x44 optic fibers arranged in a grid
- Principle of operation is based on the detection of Cherenkov light produced by the charged particles of the shower in the fibers.

Fast simulation of the ZDC

- We treat the response of the ZDC as a 44x44 1-channel image
- The image is produced in reponse to a particle described by 9 conditonal variables (Energy, mass, charge, Pxyz, Vxyz)
- We have gathered 10 milion pairs of ZDC responses and corresponding particle attributes
- The distribution of „channels” in generated data should be simillar to the distribution in original simulation

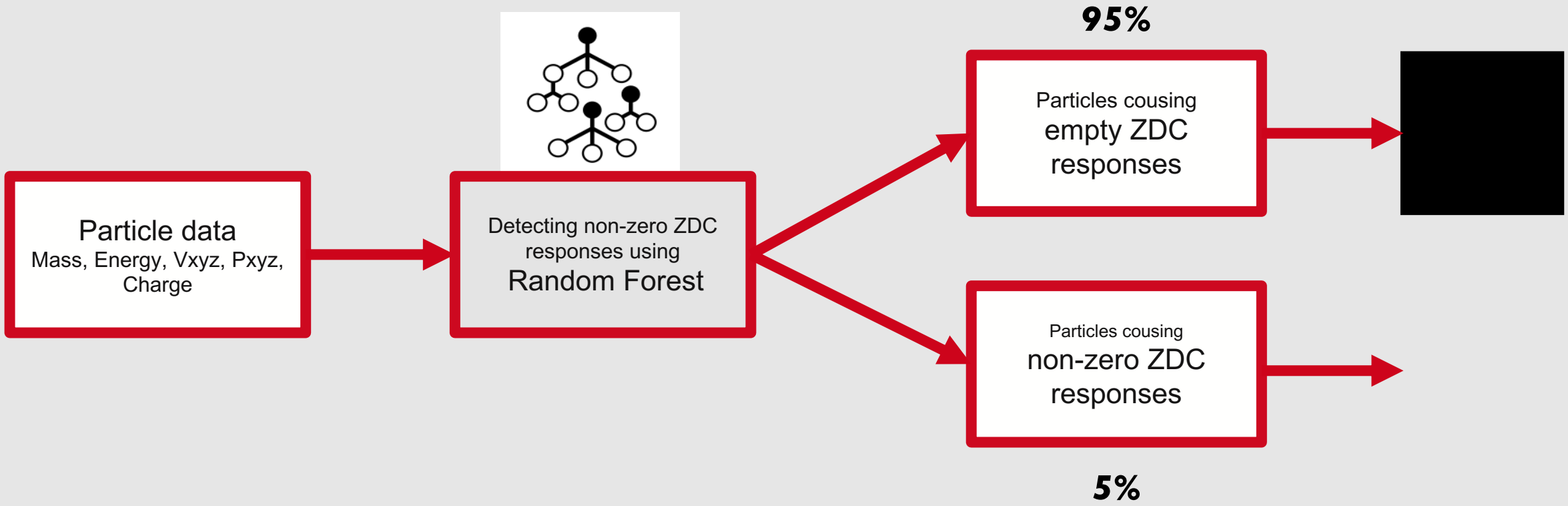






8, 22, 180, 197, 391

Detecting non-zero ZDC responses



Detecting non-zero ZDC responses

Random Forest

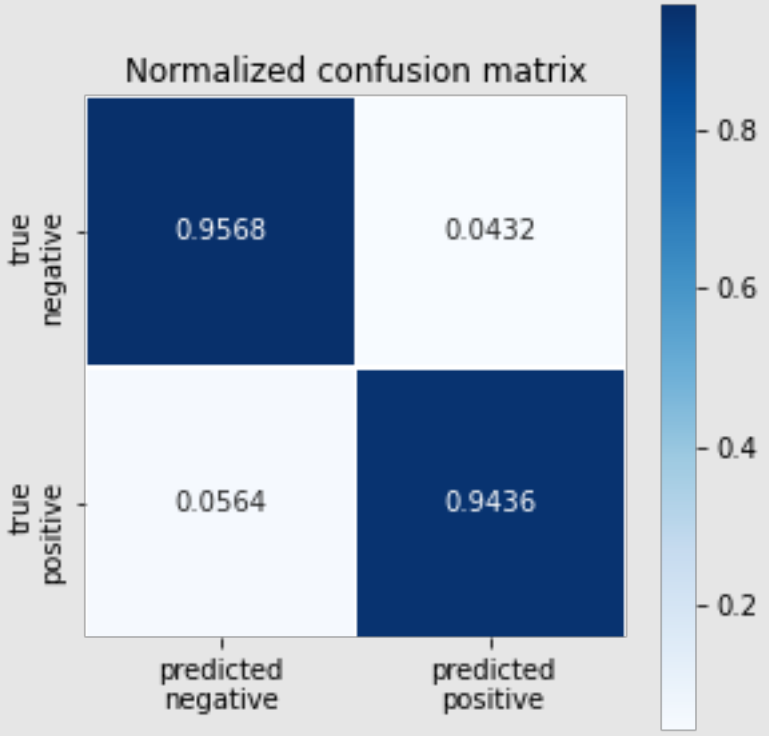
Input data:

9 particle features - mass, energy, charge, Vxyz, Pxyz

Output data:

particle will produce empty OR non-zero response

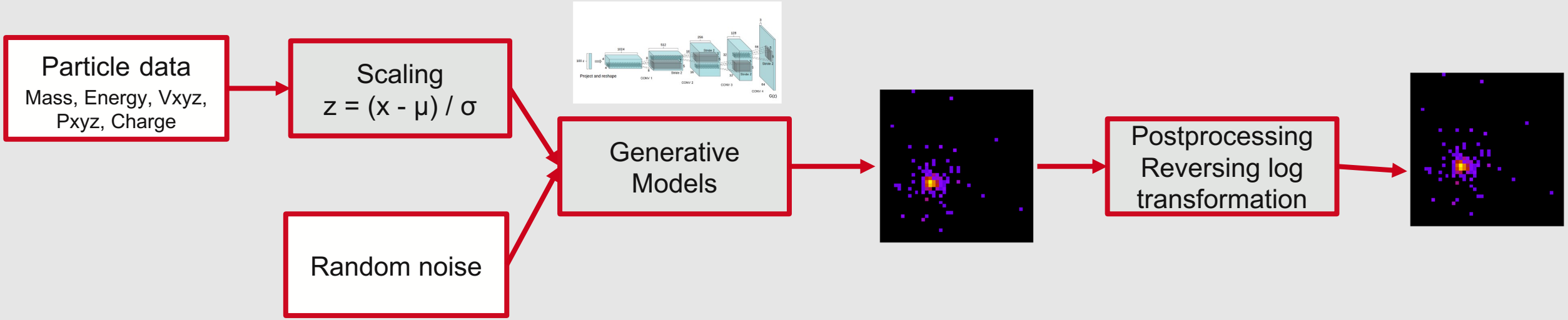
	precision	recall	f1-score
zero	0.96	0.95	0.96
nonzero	0.93	0.95	0.94
accuracy			0.95



Postive: nonzero response
Negative: zero response

Generating non-zero responses

- The input of the simulation is random noise and conditional parameters
  (Energy, primary vertex position (x, y, z), momenta (x, y, z), mass, charge)
- We scale the conditional input with standard scaler
- We transform the ZDC response images with logarithm before using them as real training data

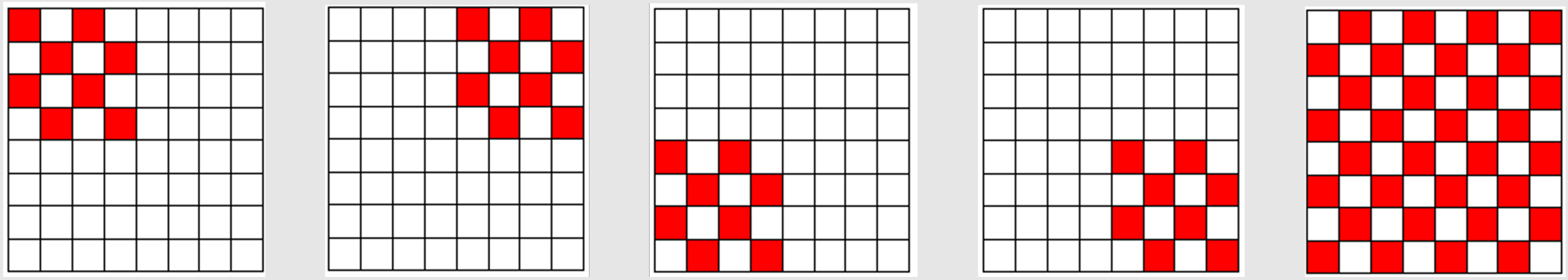


Calculation of channels

We started with validation on the basis of standard metrics (MSE, differences between placement of central hit for generated and original simulations), but simulations are too random

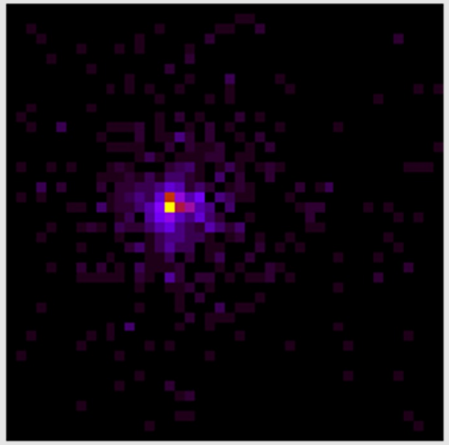
Statistical comparison of output values:

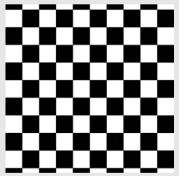
- 5 Channel values are calculated by summing pixels (photons) that are located at the specific fields of a checkerboard grid



Validation

Original Simulations

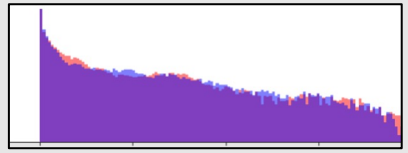




Channel count

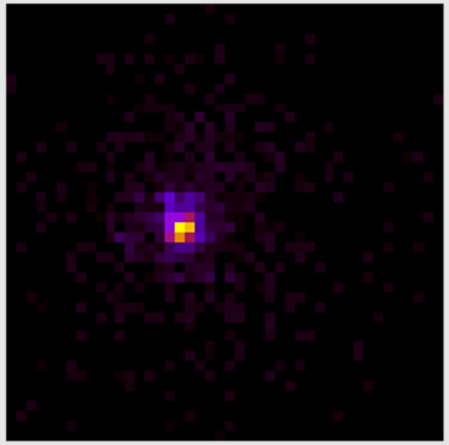
272. 44. 95. 21. 397

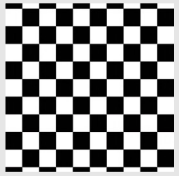
Gather distribution



Compare channels distributions (e.g. Wasserstein distance)

Generative Simulations

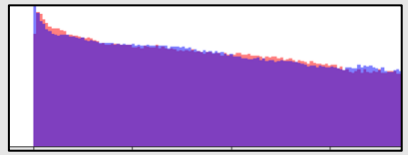




Channel count

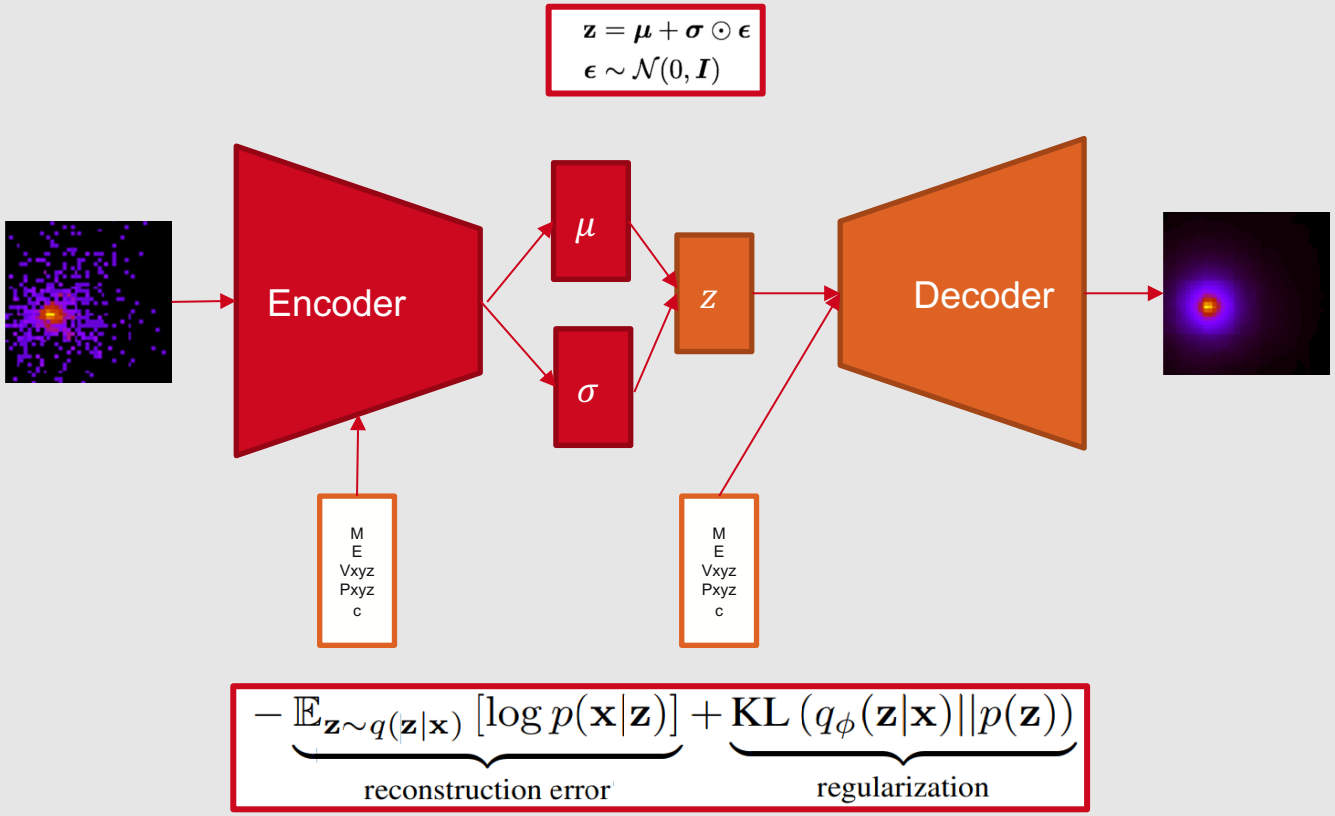
186. 43. 125. 44. 436.

Gather distribution



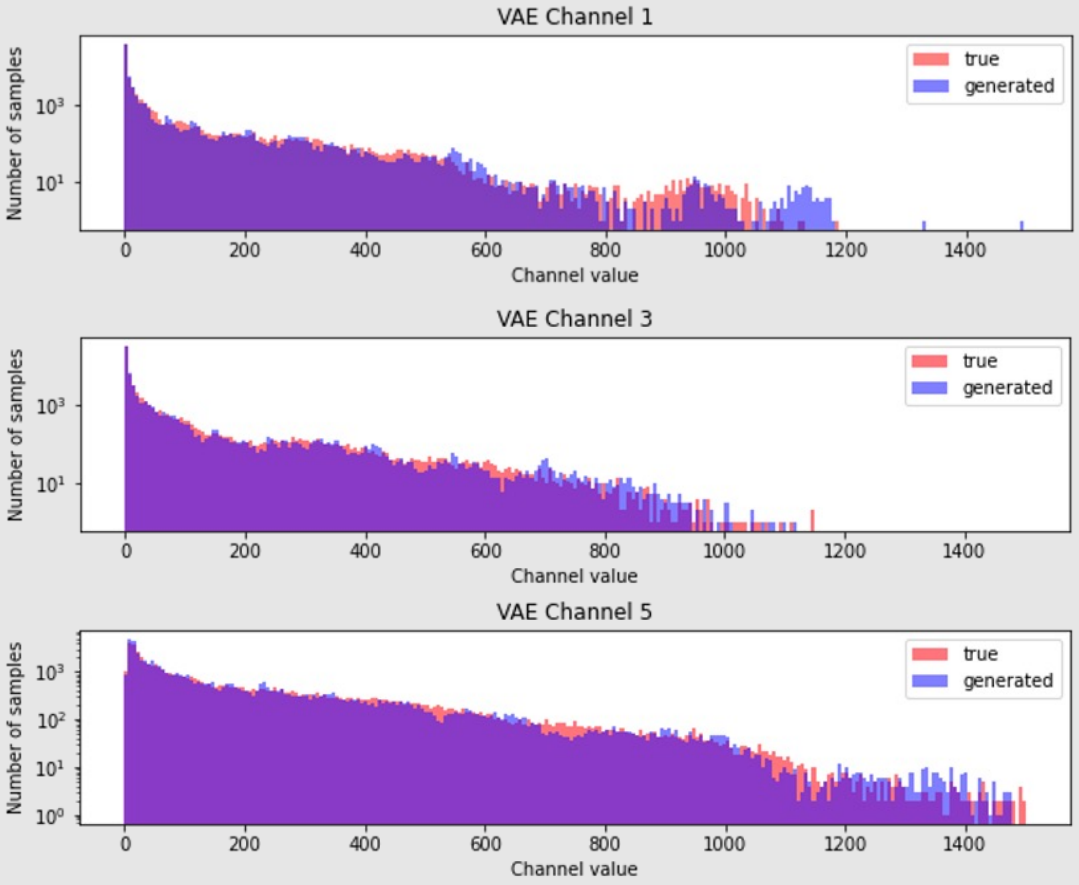
Conditonial Variational Autencoder

- **The encoder** compresses the data into a multivariate normal distribution of the latent variables
- **The decoder** attempts to decompress the data and reconstruct the input
- After training, the encoder is discarded and the decoder can be used to generate new data

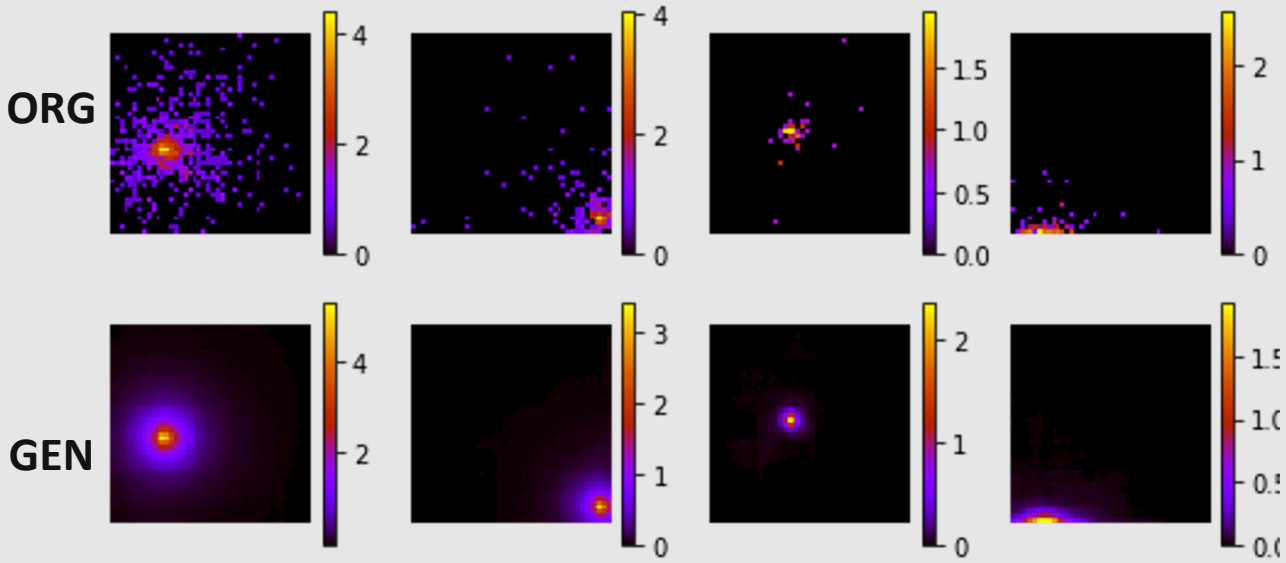


VAE results

Channel distribution comparision



Example responses

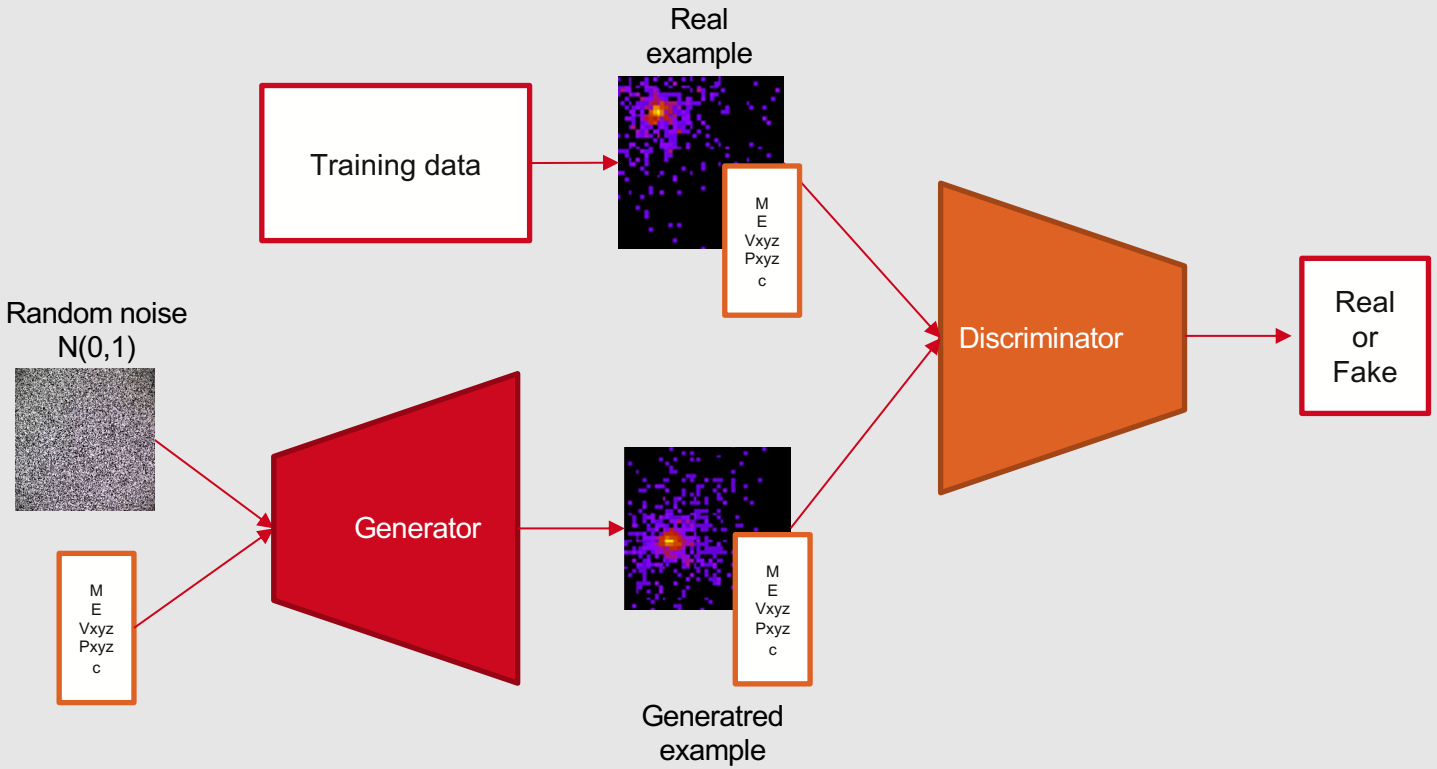


Wasserstein distance

model	MEAN	CH1	CH2	CH3	CH4	CH5
VAE	6.37	4.57	5.15	4.15	9.13	13.68

conditional DC-GAN

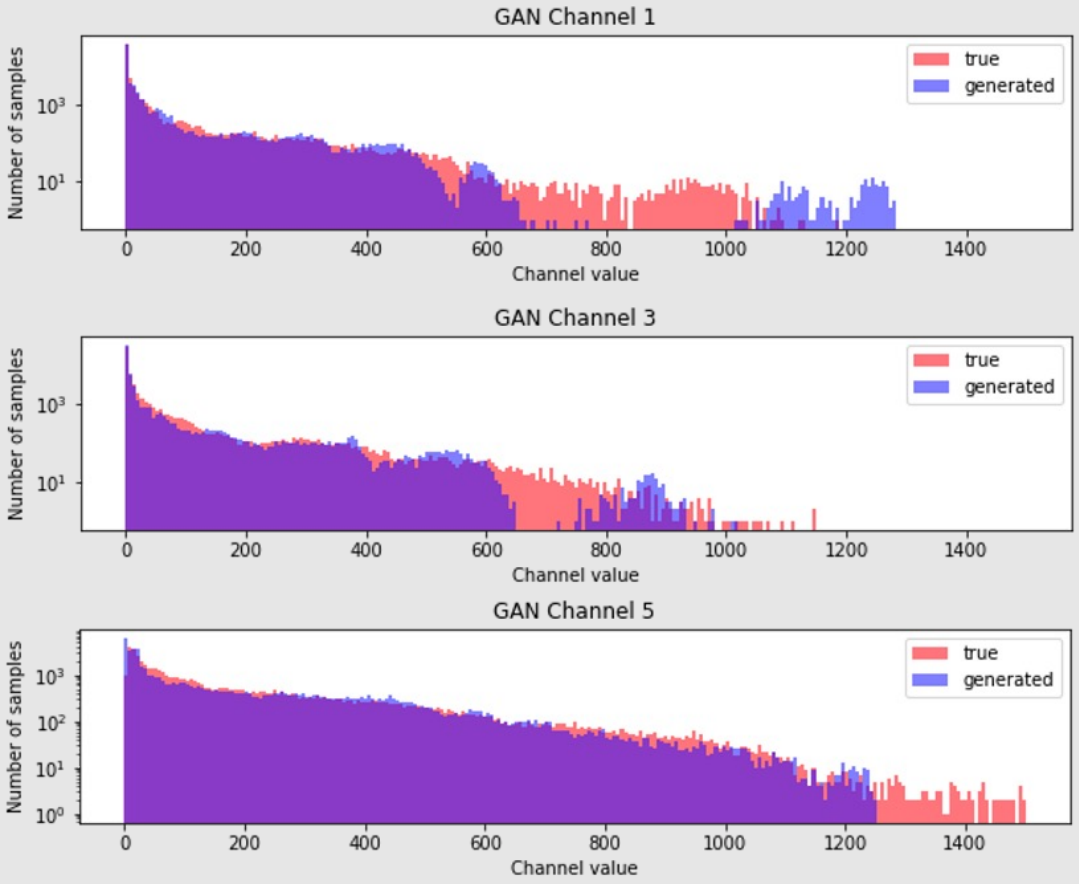
- **The generator** learns to transform random noise into realistic examples of data
- **The discriminator** learns to distinguish real data from generated data
- The two networks compete with each other during training. This process leads to a realistic generator that can be used to generate new data



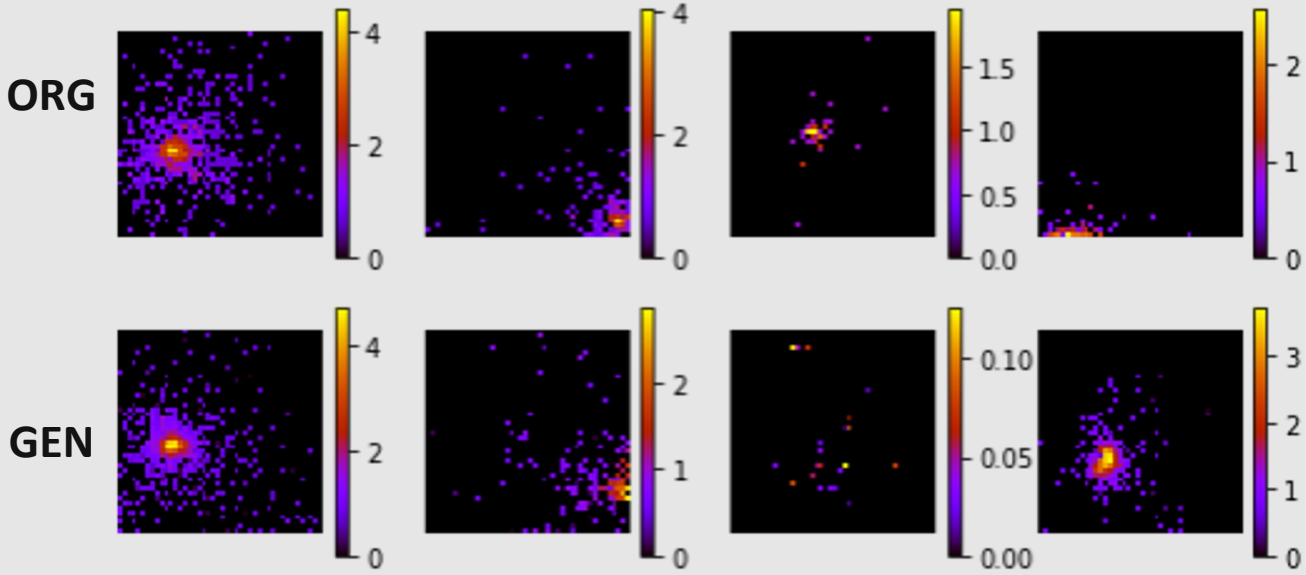
$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))].$$

GAN results

Channel distribution comparision



Example responses



Wasserstein distance

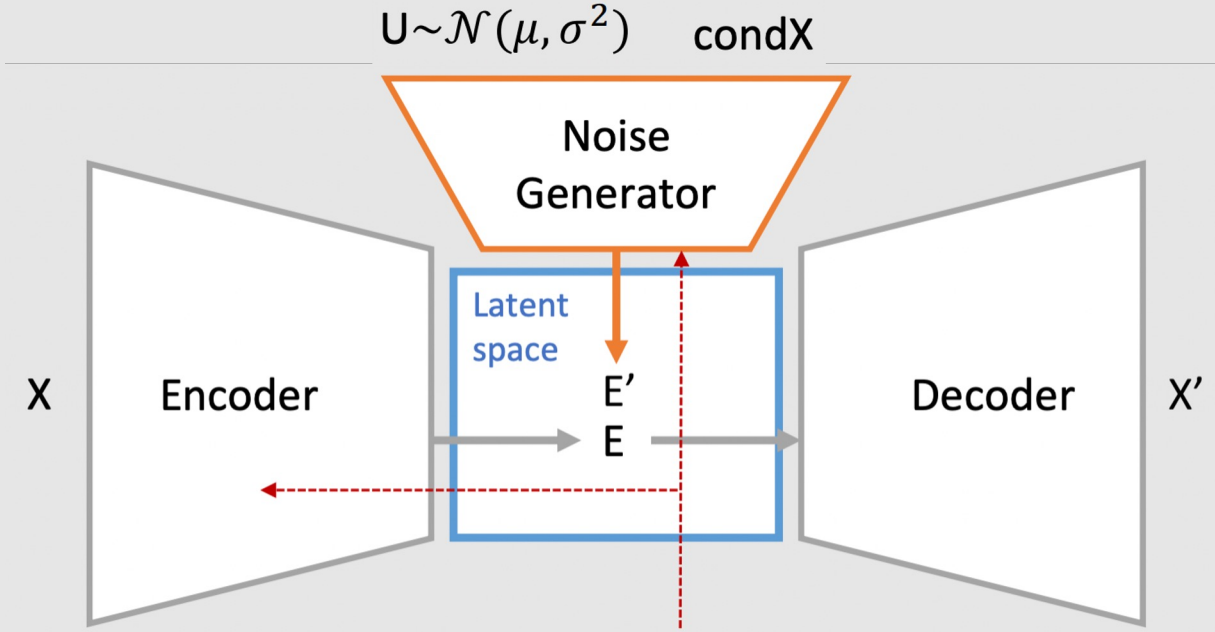
model	MEAN	CH1	CH2	CH3	CH4	CH5
GAN	8.25	4.36	5.46	7.28	9.13	14.99

End-to-end Sinkhorn autoencoder

Problem: "blurry" output of VAE due to latent space regularization

Solution:

- No implicit regularisation of autoencoder's latent space
- Approximation of original data embeddings with deterministic neural network
- Joint optimisation of both neural networks
- Conditional information added to noise generator
- Wasserstein distance between embeddings concatenated with conditional values
- Original data distribution on latent space



$$\mathcal{L}_1 = S_{c,\epsilon}([E, condX], [E'(condX), condX])$$

Encoded data

E1	E2	CondX
0.1	0.2	0
0.9	0.8	1
0.15	0.1	0

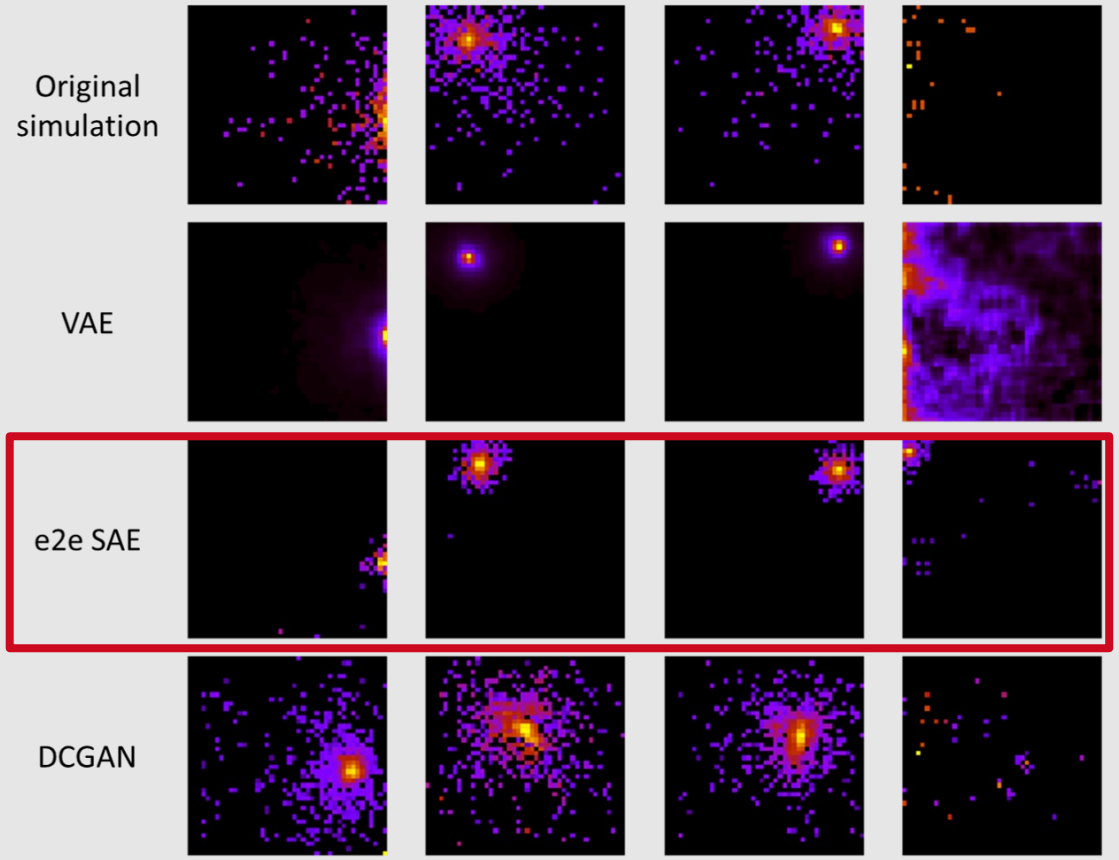
Wasserstein dist.

Encoded noise

E'1	E'2	CondX
0.16	0.12	0
0.88	0.82	1
0.11	0.18	0

e2e SAE results

Example responses



Wasserstein distance

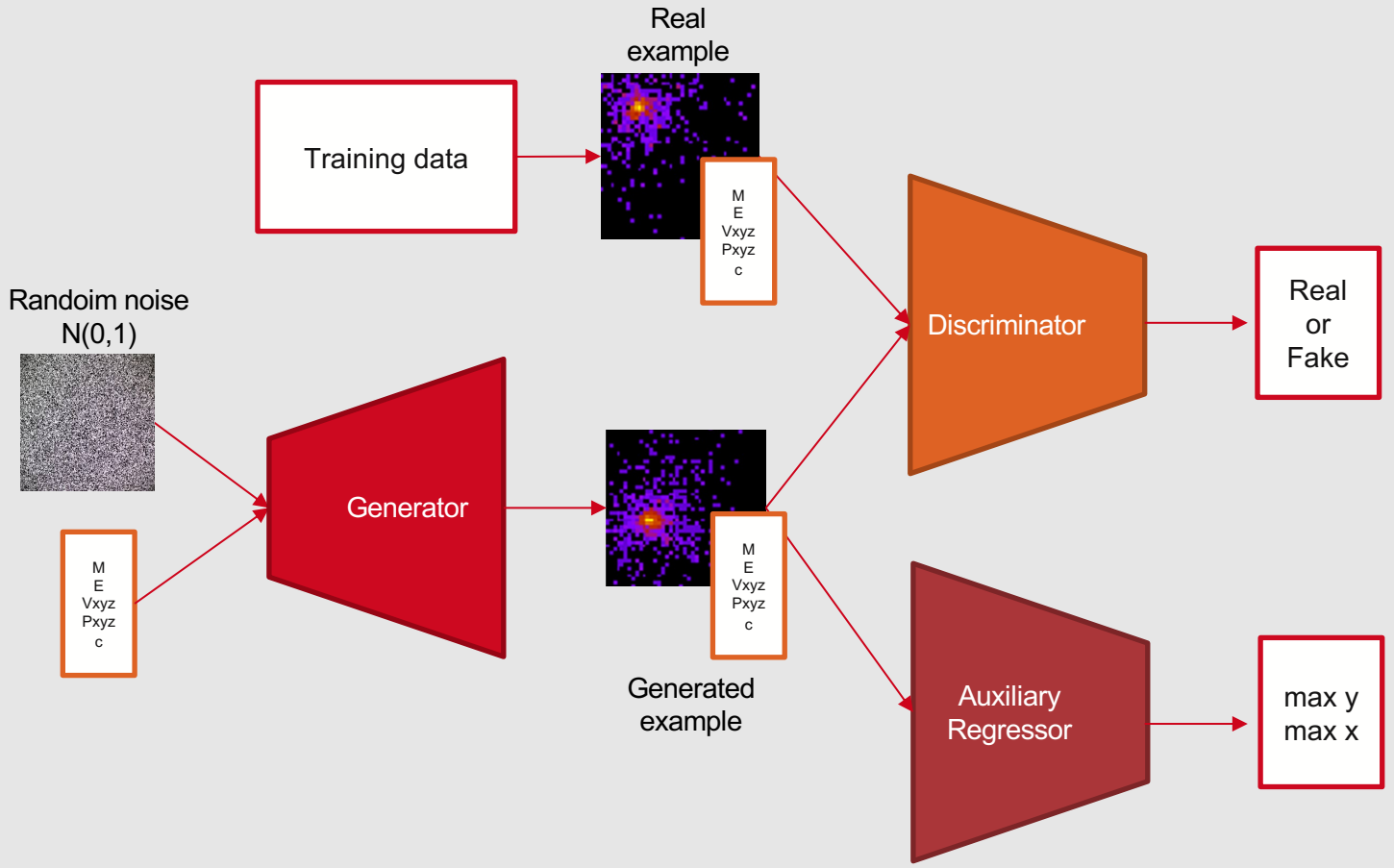
model	MEAN	CH1	CH2	CH3	CH4	CH5
VAE	6.45	4.75	5.03	4.23	4.23	13.72
GAN	8.25	4.36	5.46	7.28	9.13	14.99
E2e SAE	**6.27**	**4.17**	**5.05**	**4.05**	**4.03**	**13.58**
Upper bound	1.25	1.72	0.41	1.46	1.05	1.68

Adding an auxiliary regressor to GAN

Idea: We can control geometric properties of the generated output

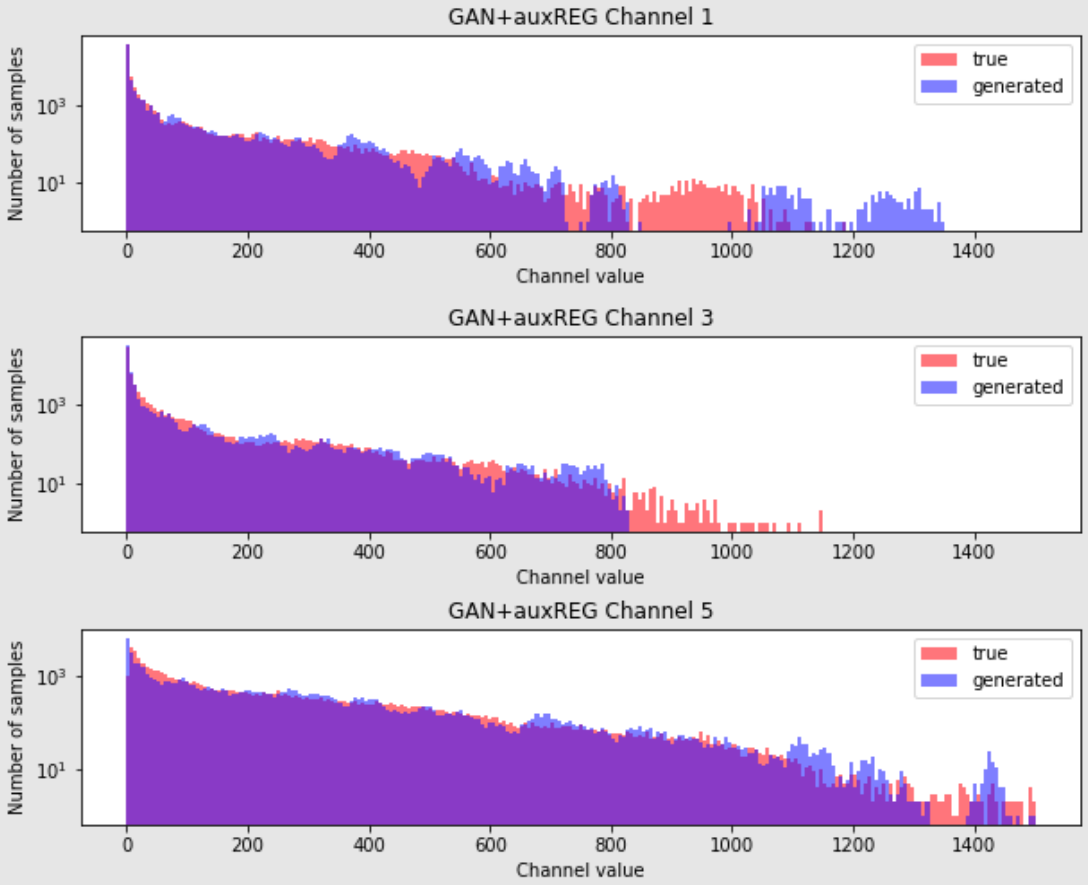
Soluton:

- **An auxiliary regressor** trained to output the position coordinates of the maximum number of photons in the input image.
- The regressor provides an additional source of loss to the generator by comparing the coordinates of the maximum of the generated examples with the maximum coordinates of corresponding sample in the training set.

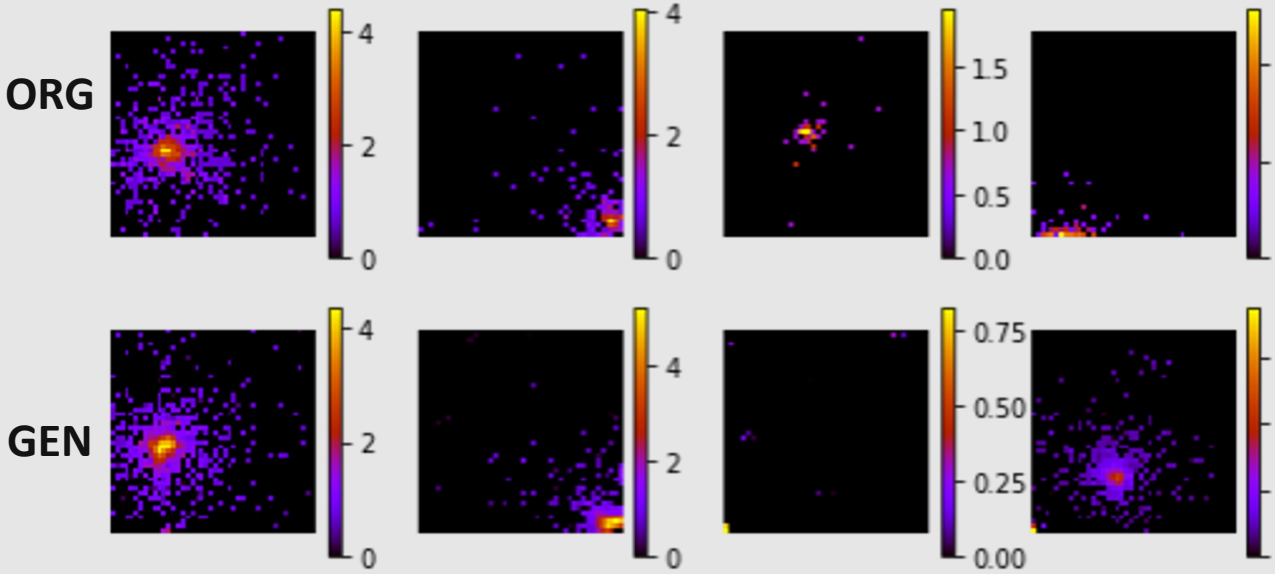


GAN + aux REG results

Channel distribution comparision



Example responses

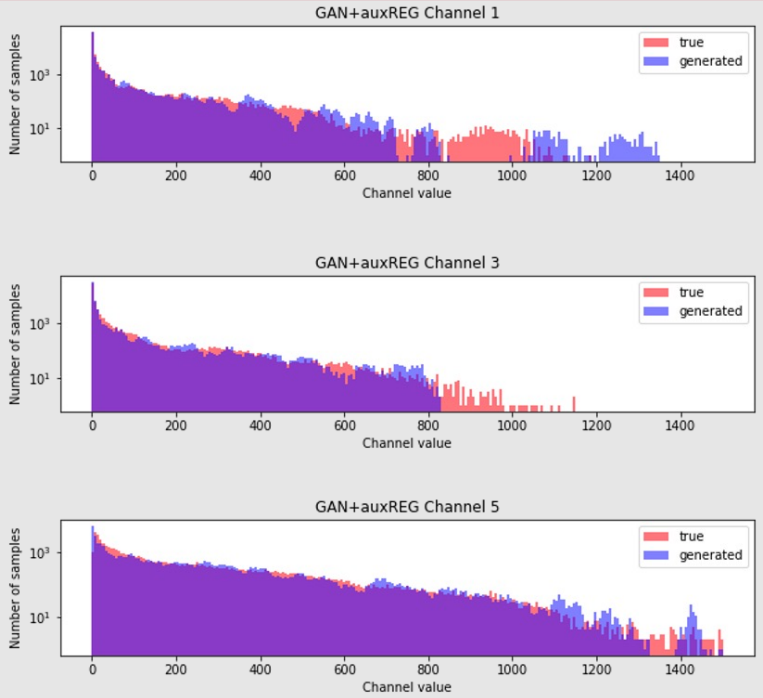


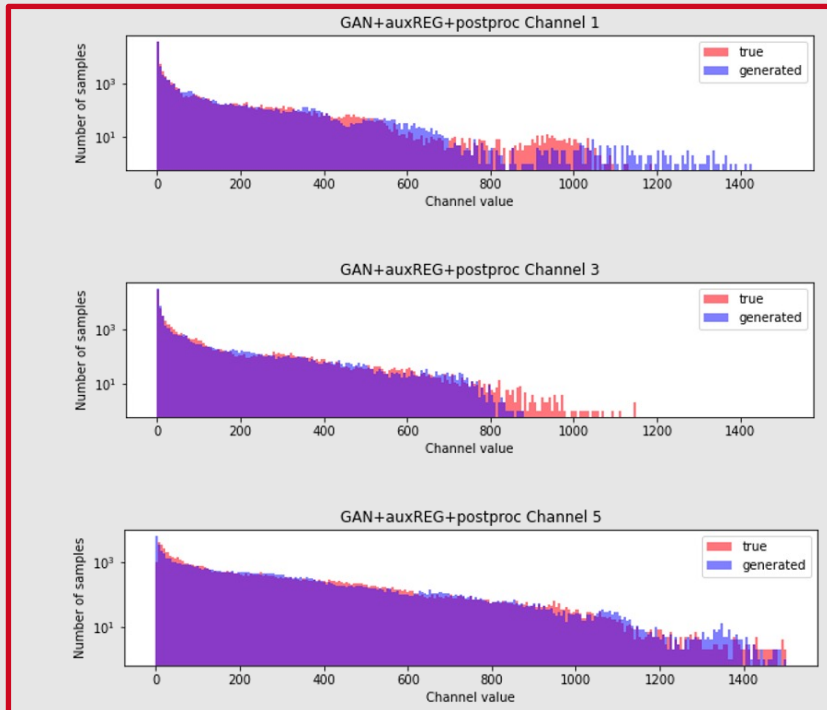
Wasserstein distance

model	MEAN	CH1	CH2	CH3	CH4	CH5
GAN + auxREG	7.20	4.24	8.42	3.54	4.56	15.19

Adding postprocessing - scaling generator response

GAN





GAN with scaling

Idea: Since we can explicitly evaluate the quality of generated results (e.g. using the Wasserstein distance) we can find an optimal value to scale the response of the generator

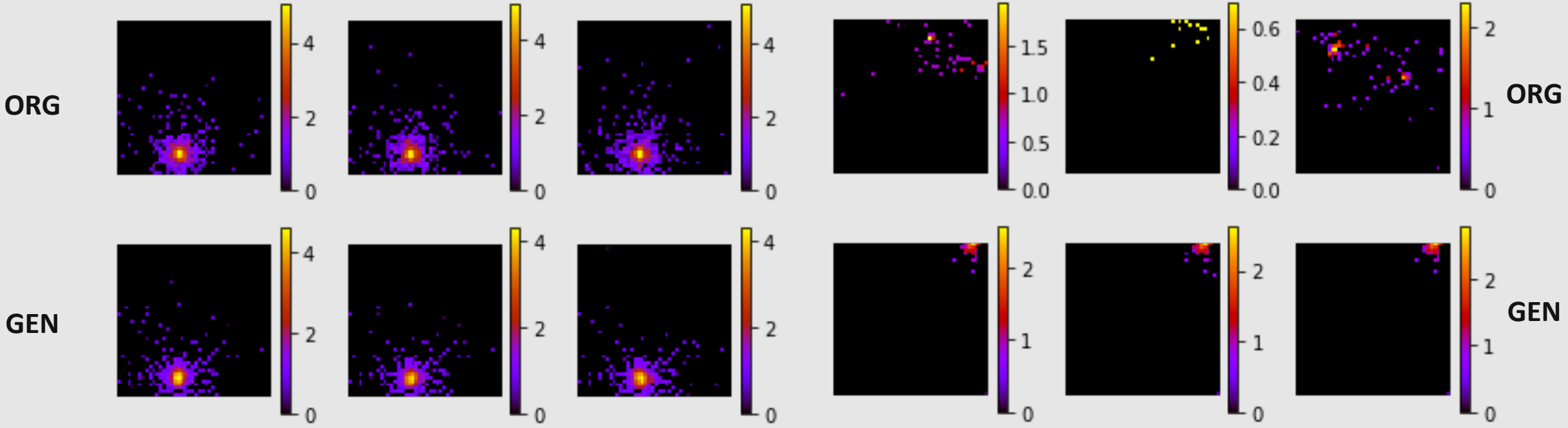
Here we see effects of multiplication of generator output values by *const =0.96*

Change in Wasserstein distance

model	MEAN	CH1	CH2	CH3	CH4	CH5
GAN + auxREG	7.20	4.24	8.42	3.54	4.56	15.19
GAN + auxREG + scaling	5.16	2.17	4.63	4.89	6.71	8.59

Problem: GANs work well for “consistent” showers …

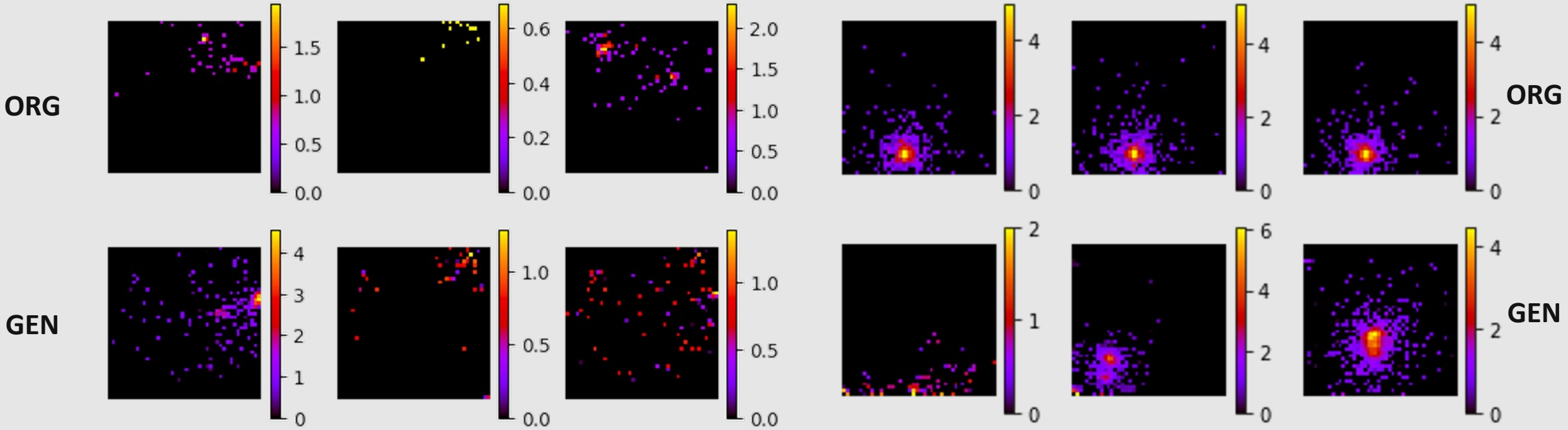
… but fail to simulate possible diverse outcomes



3 ZDC responses generated for the same particle (same input conditonal data)

Methods to increase diversity of GAN samples exist …

… but introduce problems for "consistent" showers



3 ZDC responses generated for the same particle (same input conditonal data)

Selective increase of diversity

$\propto_c$ - measure of training samples diversity for this particular set of conditonals c

We use mean standard deviation of pixels normalized from 0 to 1

d_g - measure of distance between 2 images generated from different noise vectors z_1, z_2

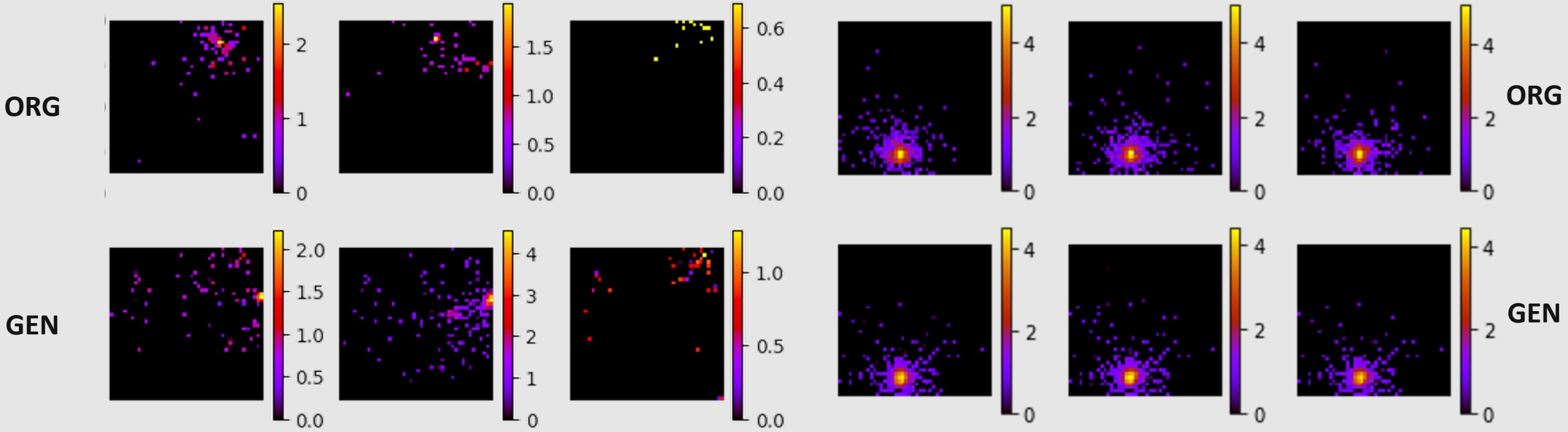
We use L_1 norm between features extracted from the discrimiantor penultimate layer

$$L = L_{gen} + \lambda L_{div} \qquad div = \propto_c \times \left(\frac{d_g\left(G((z_1, c)), G((z_2, c))\right)}{d_z(z_1, z_2)} \right)^{-1}$$

d_z - measure of distance between 2 input noise vectors
We use $L_1(z_1, z_2)$

SDI-GAN generates diverse results …

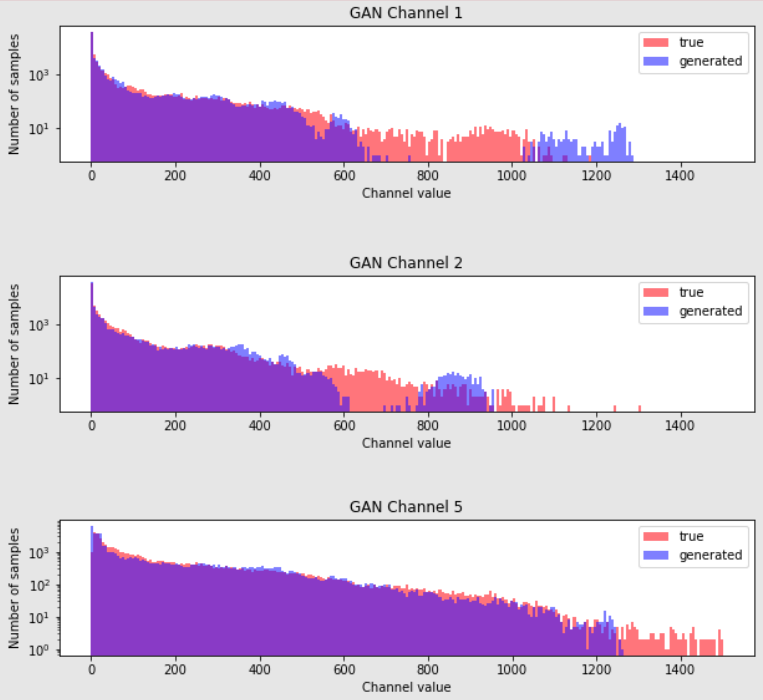
… and keeps consistency where needed

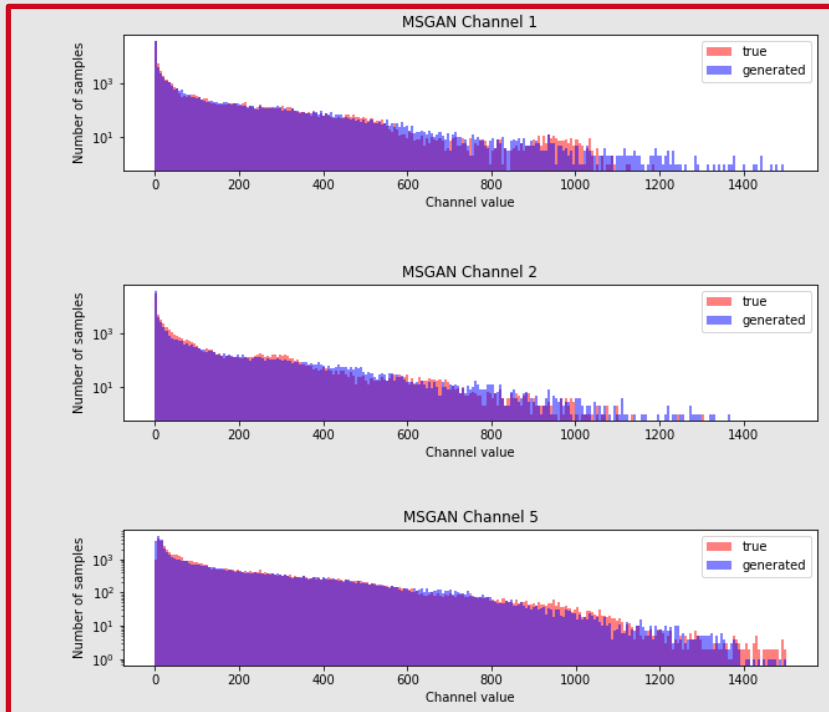


3 ZDC responses generated for the same particle (same input conditonal data)

Improvement of simulation quality

GAN





GAN with selective diversity increase

Our method:

- increases the diversity of generated samples for a selected subset of input data
- leads to higher simulation fidelity by:
  - decreasing the differences between the distribution of original and fast simulation
  - smoothing the distribution of the generated results.

Summary

model	WS MEAN	WS CH1	WS CH2	WS CH3	WS CH4	WS CH5
cond VAE	6.45	4.75	5.03	4.23	4.34	13.72
cond DCGAN	8.25	4.35	5.46	7.28	9.13	14.98
cond end2end SAE	**6.27**	**4.17**	**5.05**	**4.05**	**4.03**	**13.56**
cond DCGAN + auxREG	7.20	4.24	8.42	3.54	4.55	15.19
cond DCGAN + postproc	5.71	2.53	3.92	3.64	5.93	12.55
cond DCGAN + auxREG + postproc	5.16	2.71	4.63	4.89	6.71	8.59
cond DCGAN + selectiv div increase	**4.51**	**2.21**	**4.03**	**4.38**	**6.17**	**8.04**

The fast simulation model results in a ~100x speedup compared to the Monte Carlo-based approach.

Main takeaways:

- Generative machine learning models offer a cost-efficient alternative to Monte-Carlo based simulations
- Simulating HEP processes requires the generative models to follow strict physciall properties of the simulation
- However, this domain also offers new possibilities to improve the performance of the generative models and provides objective evaluation metrics