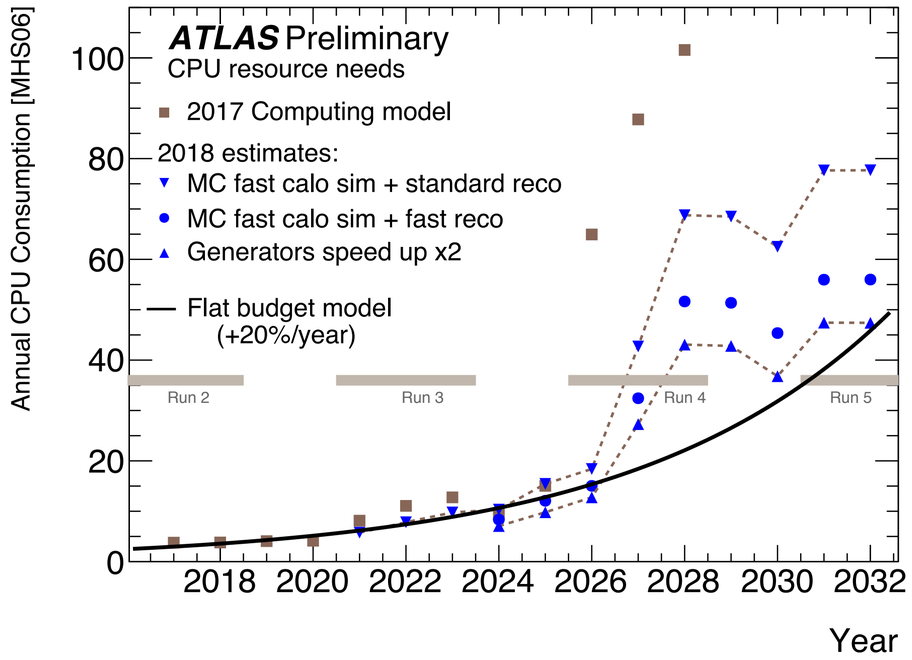
Novel Deep Autoregressive Networks For Fast Simulation

Ioana Ifrim
EP-SFT

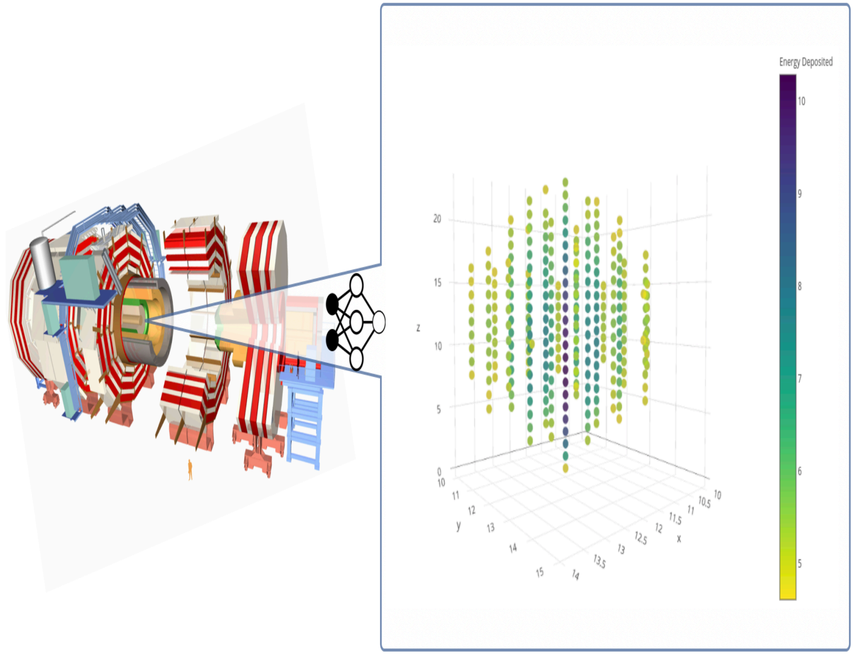
Motivation

Computational Requirements



Increasing luminosity and energy of particle accelerators pose greater challenges - large MC statistics to model experimental data - more collisions = more data = more computing resources required

Deep Learning Fast Simulation



Using Deep Learning principles in treating physics data, we can generate the simulation output (energy depositions) in a fast manner - resembling the Deep Learning task of image generation

Retrieved from [1] http://inspirehep.net/record/1747546/files/josh-hlcpu.png [2] https://cds.cern.ch/record/1433717/files/CMSnc.jpg

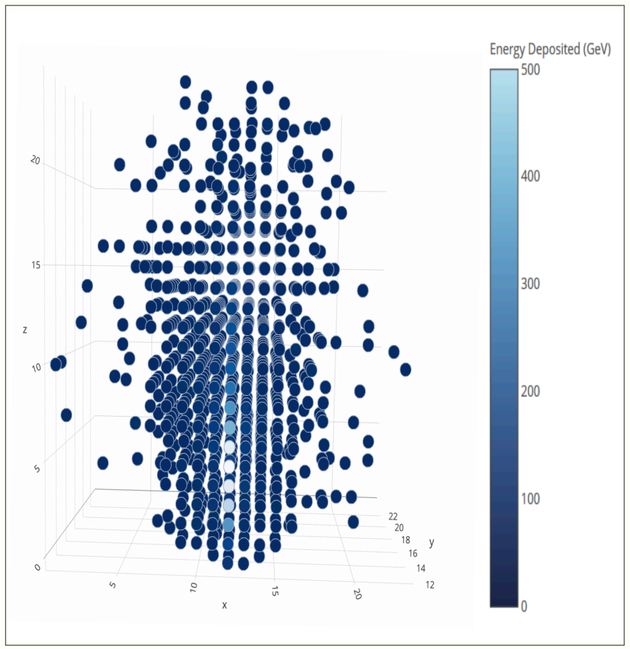
Training Data Production

- Using the Geant4 full simulation toolkit

- A single particle event for deep learning training is represented by:

    - the label = particle properties (energy, type, …)
    - energy depositions on (x,y,z)

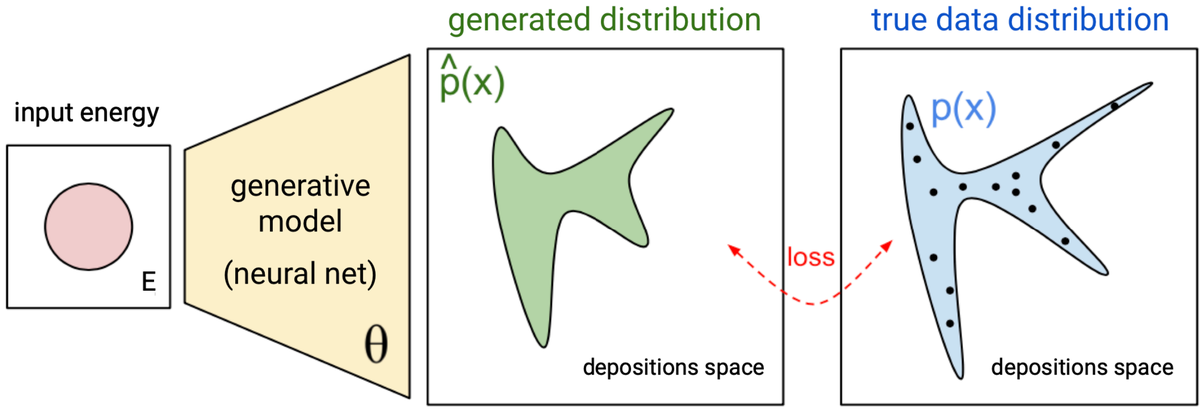
- Flat incoming energy spectrum (1-500 GeV) along z axis



EM sum shower of 3 300 GeV electrons

- For training we use 24x24x24 cells, each having the size of 10x10x10mm in $PbWO_4$

Generative Deep Learning



- The learning aim is for the parameters within a group of model distributions to minimise the distance between the model distribution p_θ and our calorimeter showers data distribution p_{data}:

$$\min_{\theta \in \mathcal{M}} d(p_{\text{data}}, p_\theta)$$

- Our choices of design revolve around:
  - which model representation is suitable?
  - what is the objective function of distance, d(·)?
  - which optimisation procedure should we use for minimising d(·)?

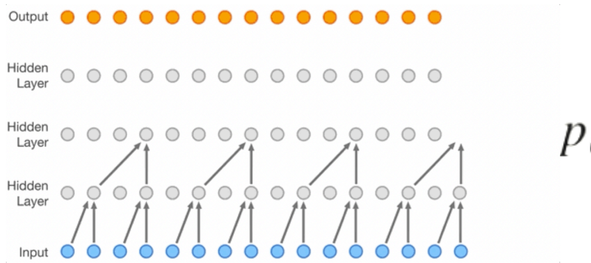
https://openai.com/blog/generative-models/

Generative Models

- Models learn the probability density function differently:

The choice of our study

Explicitly - tractable: **Autoregressive Models**

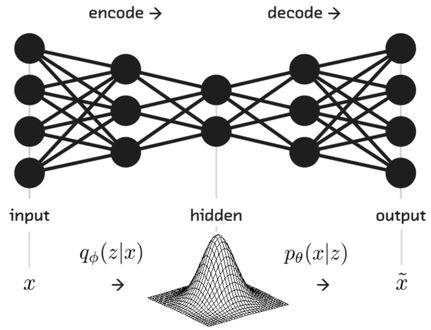


$$p_\theta(\mathbf{x}) = \prod_{i=1}^{N} p_\theta(x_i|\mathbf{x}_{<i})$$

- Tractable and scalable model
- Complex modelling of the data distribution
- Able to encode long term dependencies between energy cells
- Training is highly parallelizable given the type of operations
- The training is more stable than GANs

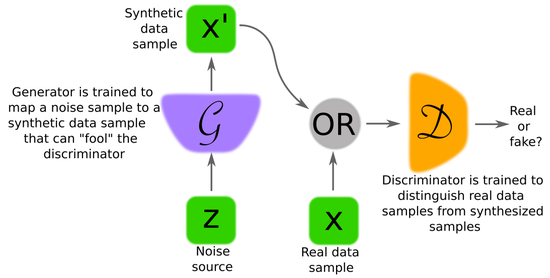
Commonly studied

Explicitly - approximation: Variational Autoencoders (**VAEs**)



$$p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{x}, \mathbf{z})\mathrm{d}\mathbf{z}$$

Implicitly: Generative Adversarial Networks (**GANs**)



$$\min_\theta \max_\phi V(G_\theta, D_\phi) = \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}[\log D_\phi(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})}[\log(1 - D_\phi(G_\theta(\mathbf{z})))]$$

https://blog.fastforwardlabs.com/2016/08/22/under-the-hood-of-the-variational-autoencoder-in.html https://tinyurl.com/y2vl6ulm https://deepmind.com/blog/article/wavenet-generative-model-raw-audio

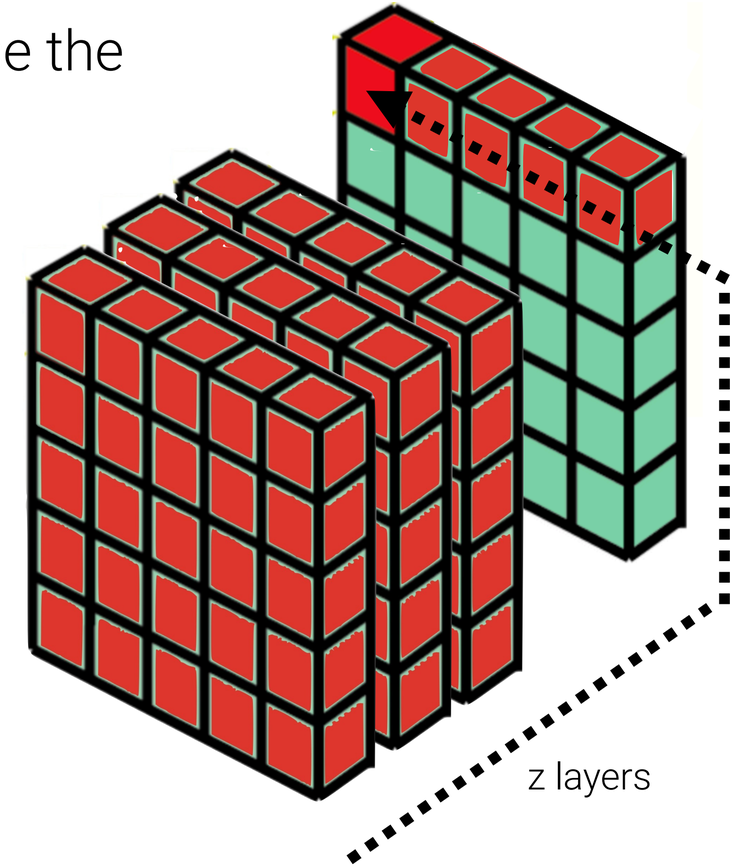
Model Representation - Autoregressive

- Given our dataset D of n-dimensional data points x, we can factorise the joint distribution over the n-dimensions as:

$$p_{(x)} = \prod_{i=1}^{n} p_{(x_i | x_1, \dots \ x_{i-1})}$$

Likelihood of event x

Probability of i'th energy deposition value given all previous depositions



- Correlations between layers are kept given that cell x_i is dependant on preceding z layers cells

Modelling of Data Distribution

- How do we generate energy depositions values based on incoming particle properties

    - we predict the distribution of energy depositions, thus any value will have a probability to be represented (regardless if it is present or not in the training set, as per softmax)

- How to model the complex data distribution?

    - use a linear combination of distributions

$$p(x) = \pi_1 \times p_1(x) + \pi_2 \times p_2(x) + \ldots + \pi_n \times p_n(x)$$

where π_i is the probability of the distribution to be picked

Distribution Type

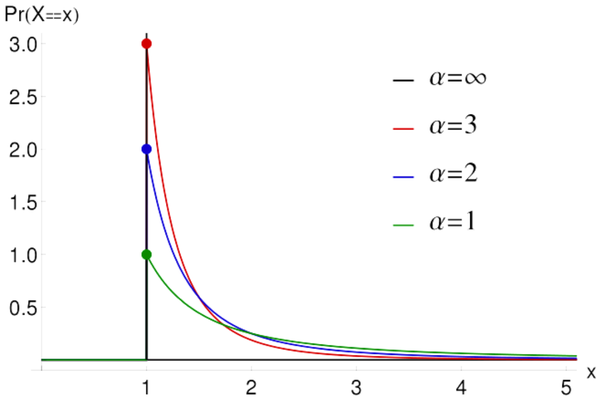
- Our data consists of a large number of low energies entries while high energy ones have a much lower occurrence rate

- We have chosen a Pareto distribution, where if X is a random variable with such distribution, then the probability that X is greater than a number x is given by:

$$\overline{F}(x) = \Pr(X > x) = \begin{cases} \left(\frac{x_{\mathrm{m}}}{x}\right)^{\alpha} & x \geq x_{\mathrm{m}}, \\ 1 & x < x_{\mathrm{m}}, \end{cases}$$

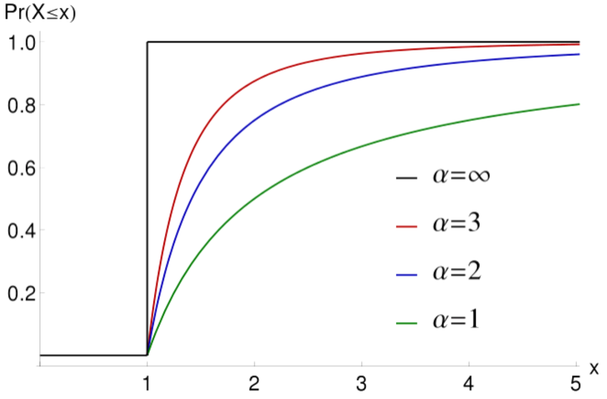
- The distribution is characterised by a scale parameter x_m and a shape parameter α, which is known as the tail index

Probability density function



Pareto Type I probability density functions for various α with $x_{\mathrm{m}} = 1$. As $\alpha \to \infty$, the distribution approaches $\delta(x - x_{\mathrm{m}})$, where δ is the Dirac delta function.

Cumulative distribution function



Pareto Type I cumulative distribution functions for various α with $x_{\mathrm{m}} = 1$.

Retrieved from [1] https://en.wikipedia.org/wiki/Pareto_distribution

Sampling

- Select which distribution from the linear combination should be used by applying a softmax-like sampling (Gumbel-max trick for sampling from a discrete distribution parametrized by unnormalized log-probabilities)

$$s(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}}$$

- Sample from the chosen Pareto distribution:

$$1 - \left(\frac{x_m}{x}\right)^{\alpha} => x = \frac{x_m}{(1-y)^{\frac{1}{\alpha}}}$$

Loss Functions

- To train sampling layers in the neural network, the objective is to have some output distribution parameters that maximise the likelihood of a target y to be sampled from such distribution

- Our loss is comprised of:

    - The loss on the distribution itself (how likely is the distribution to hold the target y)

    - The loss on the distribution selection

Distribution Loss

$$cdf_{min} = 1 - \left(\frac{x_m}{x - \varepsilon}\right)^{\alpha} \qquad cdf_{plus} = 1 - \left(\frac{x_m}{x + \varepsilon}\right)^{\alpha}$$

- The challenge is to maximise the derivative at the point of y (cdf(x)) -> max(cdf_{plus} - cdf_{min})

- Transforming this to a log likelihood loss function:

$$\log\left(\frac{\alpha \times x_m^{\alpha}}{x^{\alpha+1}}\right)$$

Distribution Selection Loss

- The model should decide on the best Pareto distribution (from the linear combination) to be used depending on the situation

- The distribution selection loss (probs) is given by the product between cdf_{delta} and a simple softmax over the distributions probabilities:

$$cdf_{delta} = cdf_{plus} - cdf_{min} \qquad probs = cdf_{delta} \times softmax(\pi)$$

Discretisation Of Energy Depositions

- The continuous univariate distribution is described as a linear combination of distributions from which we compute the discretised value of energy deposition x

$$P(x|\pi, x_m, \alpha) = \sum_{i=1}^{K} \pi_i \left[\left(\frac{x_m}{x+\varepsilon} \right)^{\alpha} - \left(\frac{x_m}{x-\varepsilon} \right)^{\alpha} \right]$$

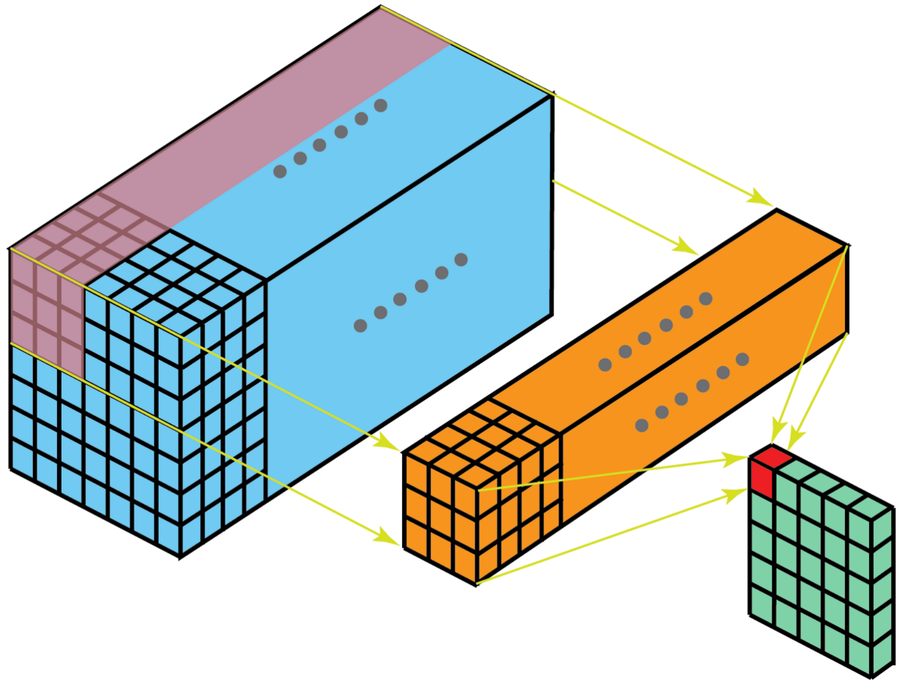
where ε is half the smallest difference between two energy depositions

- Thus, the discrete deposition x is represented as the range between [x- ε, x+ ε]

Network Implementation

- Cells are conditioned along the shower development axis while subsequent information is blocked

- The dependency on previous layers is modelled using a Convolutional Neural Network over a context region

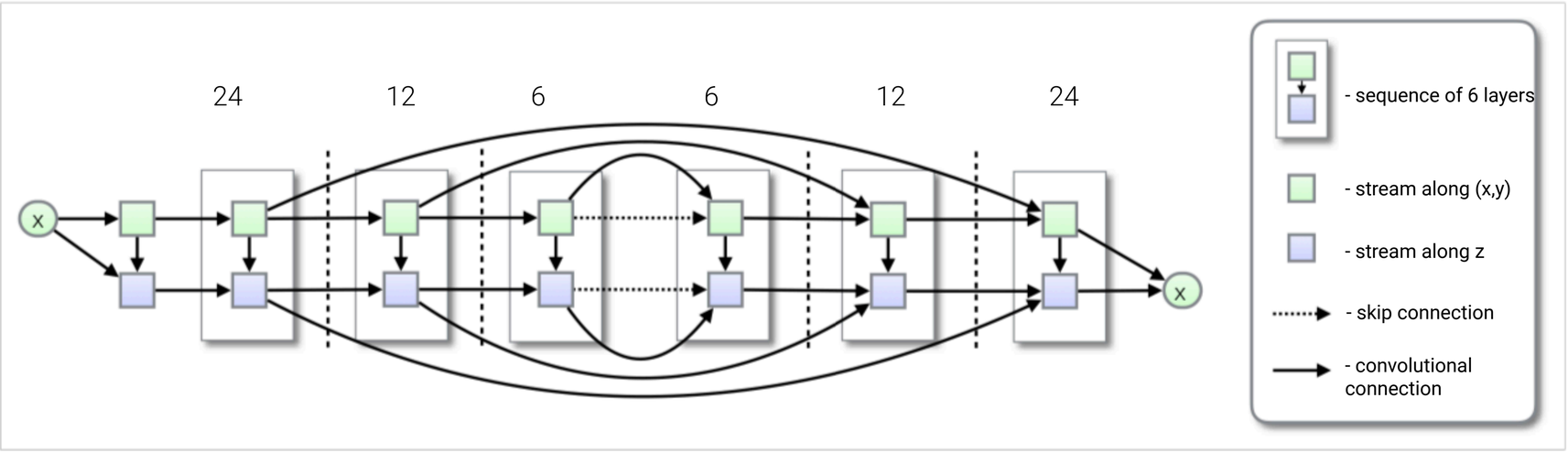


https://towardsdatascience.com/a-comprehensive-introduction-to-different-types-of-convolutions-in-deep-learning-669281e58215

Network Implementation

- The network captures the long range structure by downsampling with convolutions of stride 2 (thus improving the relative size of receptive field); the loss in information is accounted for by adding extra short-cut connections

- The behaviour of a Recurrent Neural Network is emulated with a Convolutional Neural Network in order to parallelise the computations and control the access to past information

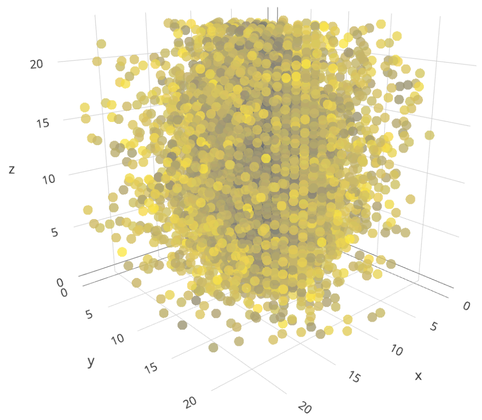


https://openreview.net/pdf?id=BJrFC6ceg

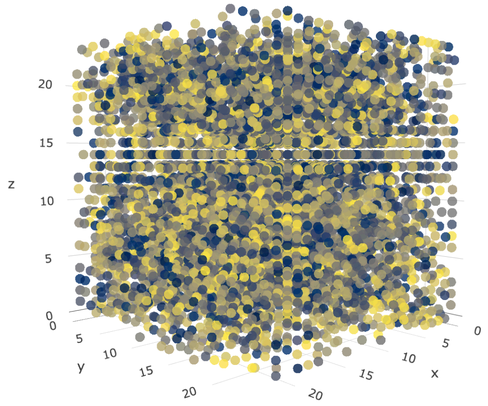
First Tests

- The network shows a learning trend along the shower development axis

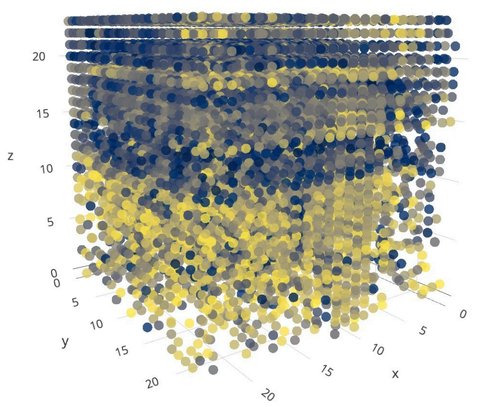
- Development is ongoing according to validation and optimisation procedures



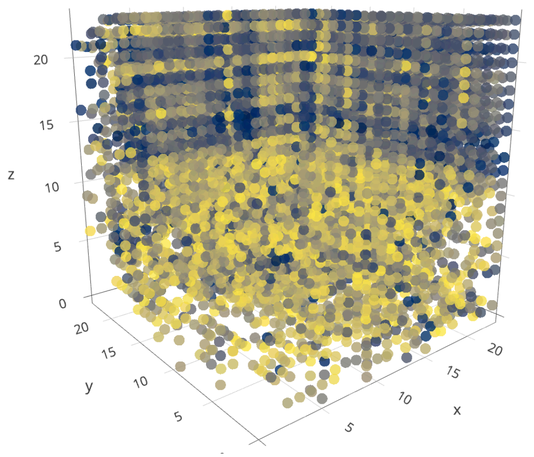
Shower Example



Iteration 2



Iteration 65



Iteration 87

Training iteration time is approximately 174 seconds

Conclusions

- We are running comprehensive tests on modelling the complexity of HEP data with linear combinations of distributions
- The model goes beyond the widely used Gaussian spaces for feature sampling (VAEs)
- The chosen training procedure is highly parallelizable
- We exploit the causal structure of the generative process and preserve layer wise correlations
- Training is more stable than GANs: training a GAN requires finding the Nash equilibrium, making it unstable when compared to autoregressive networks
- In contrast with VAEs, the autoregressive model provides tractable likelihoods
- Autoregressive sequential models have worked for audio (WaveNet), images (PixelCNN++) and text (Transformer): these models are very flexible in the kind of data that they can model

Thank you!