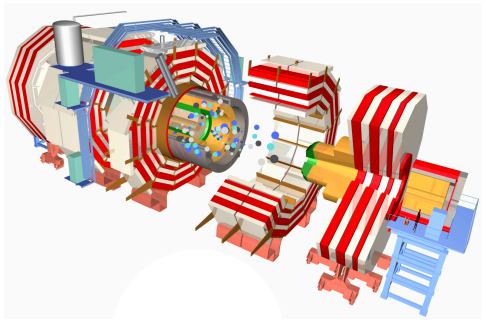
Fast Simulation for HEP Experiments

- Deep Learning Techniques -

Ioana Ifrim

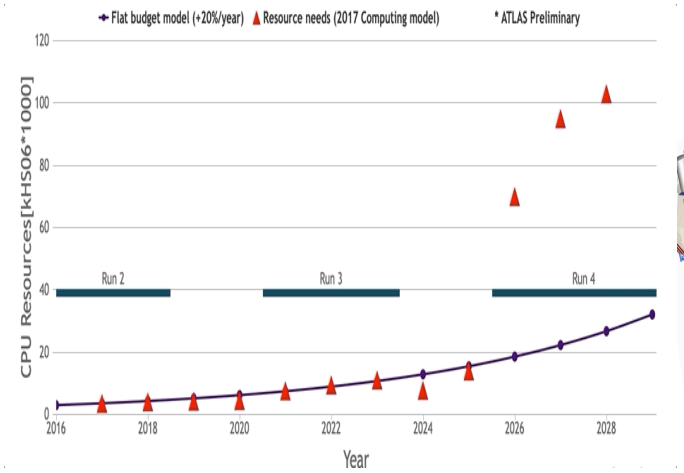
What? Why? How?

Full Simulation of Physics Detectors



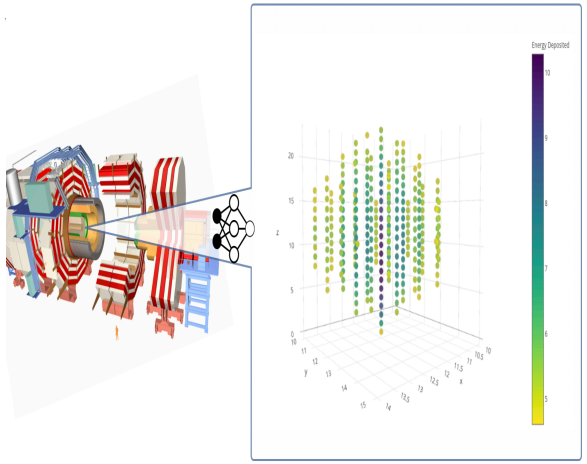
One particle at a time propagates through the physics detector and energy
depositions in (x, y, z) are recorded at every step. This then corresponds to the experimental output

Computational Efficiency



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

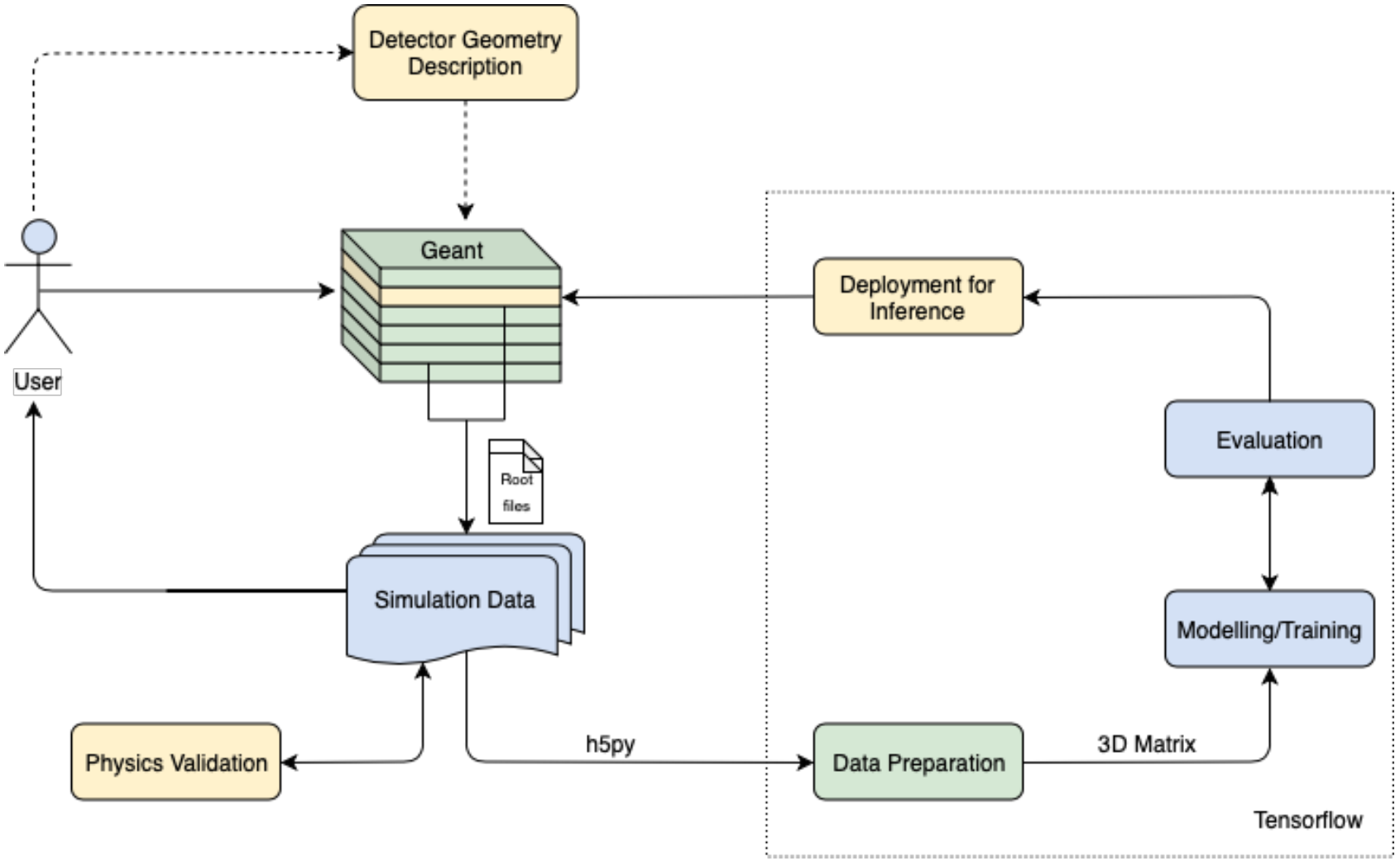
Fast Simulation



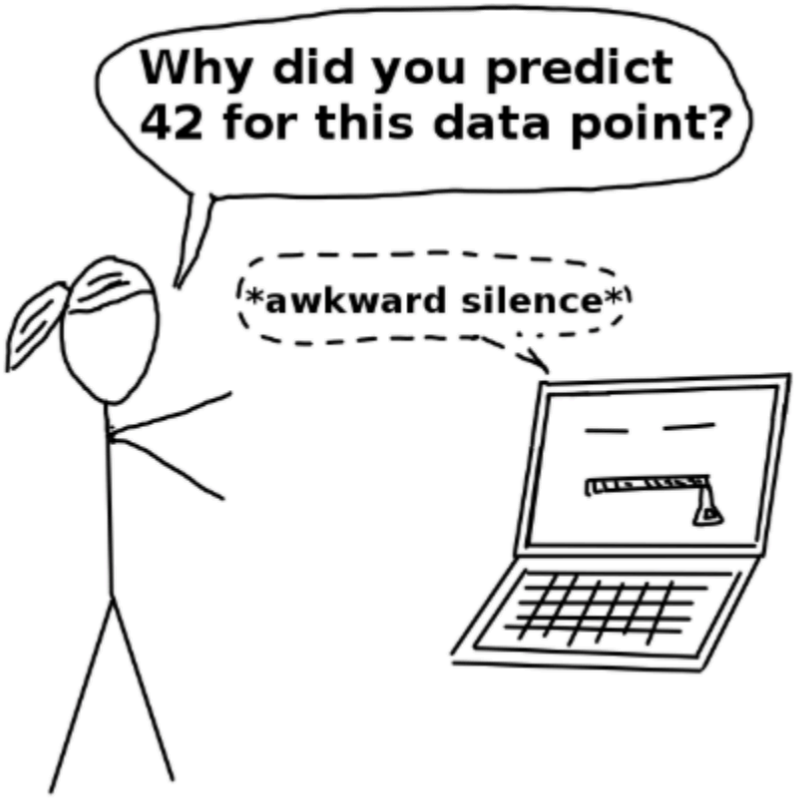
Using Deep Learning principles in treating physics data we can generate the simulation output (energy depositions) in a fast manner (2-3 orders of magnitude faster than full simulation) resembling the Machine Learning task of image generation

Pictures retrieved from [1] https://cds.cern.ch/record/1074317/files/gen-2007-004.jpg [2] https://cds.cern.ch/record/1433717/files/CMSnc.jpg

DL FastSim Development Plan



Deep Learning

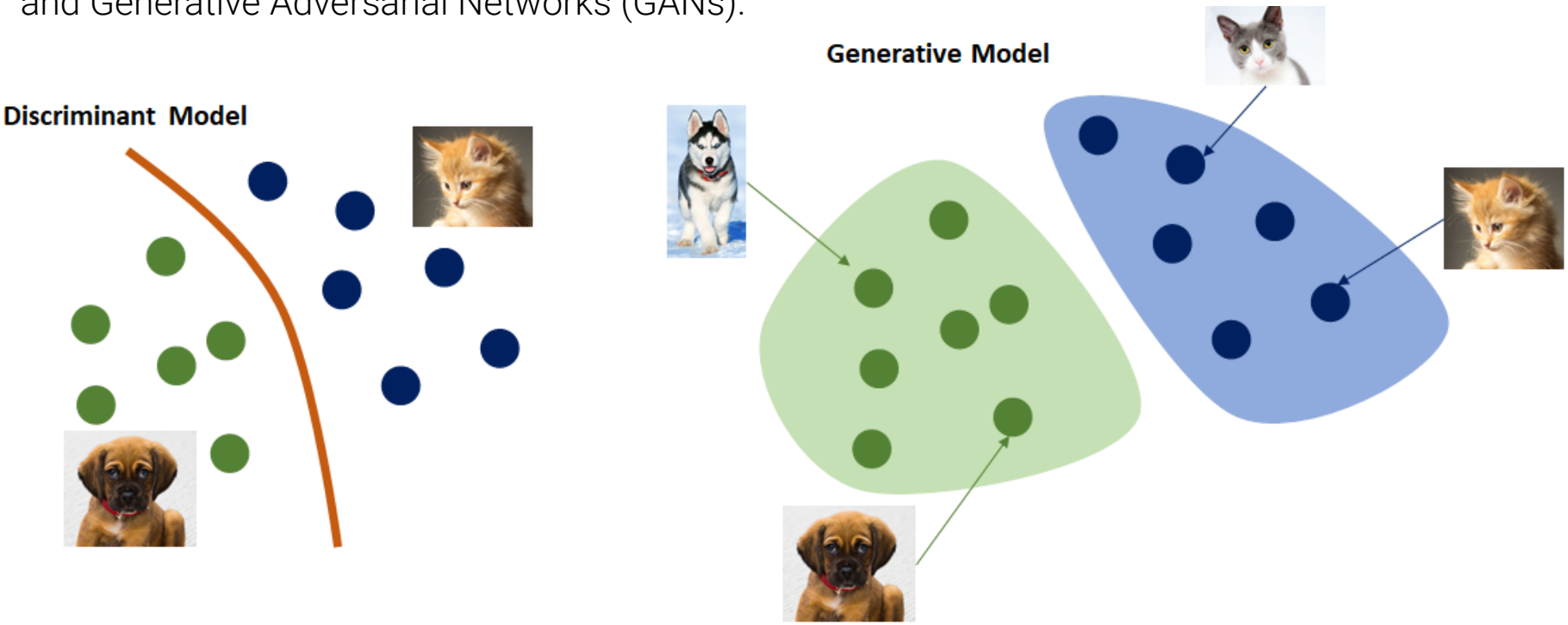


Source: interpretable-ml-book

Generative Models

- Generative models are a subset of deep learning networks which for a given training data-set, they generate new samples/data from the same distribution.

- Two ways to model the distribution (explicit and implicit density). The most efficient and popular of generative models are: Auto-Regressive models, Variational Auto-Encoders (VAEs) and Generative Adversarial Networks (GANs).



Generative Models - Taxonomy

Models learn the probability density function either:

1. Explicitly:

   - Approximate density: Variational Auto-Encoders (VAEs)

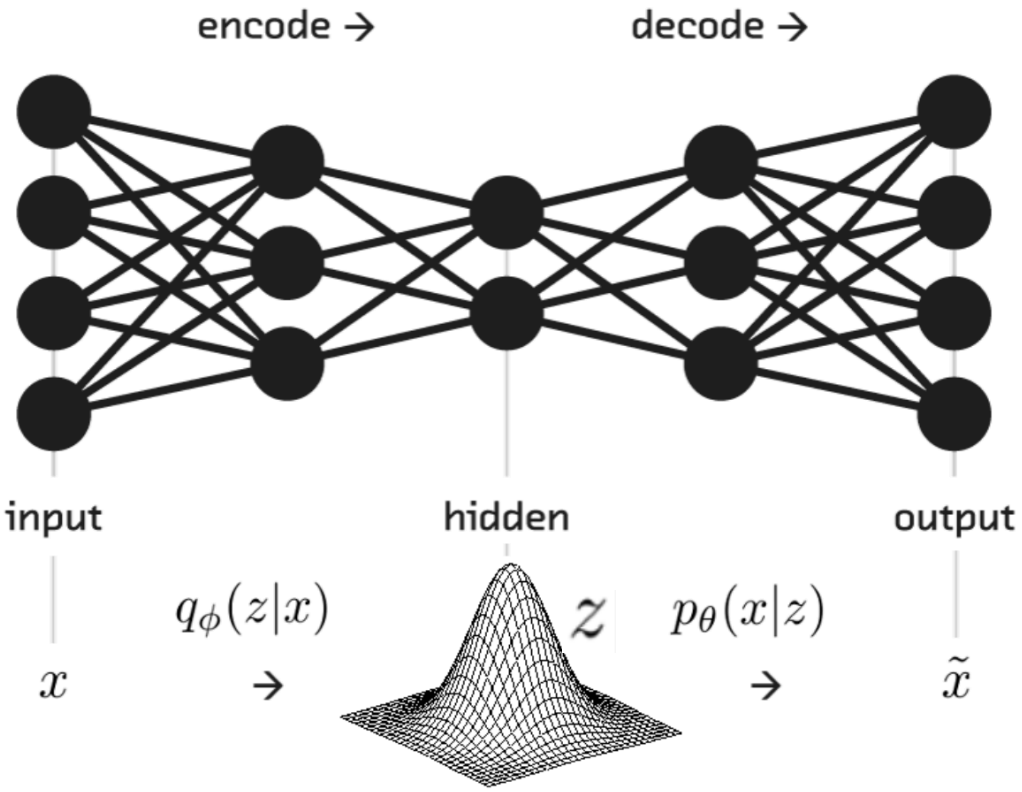
   - Tractable density: PixelCNN, Wavenet

2. Implicitly:

   - Direct: Generative Adversarial Networks (GANs)

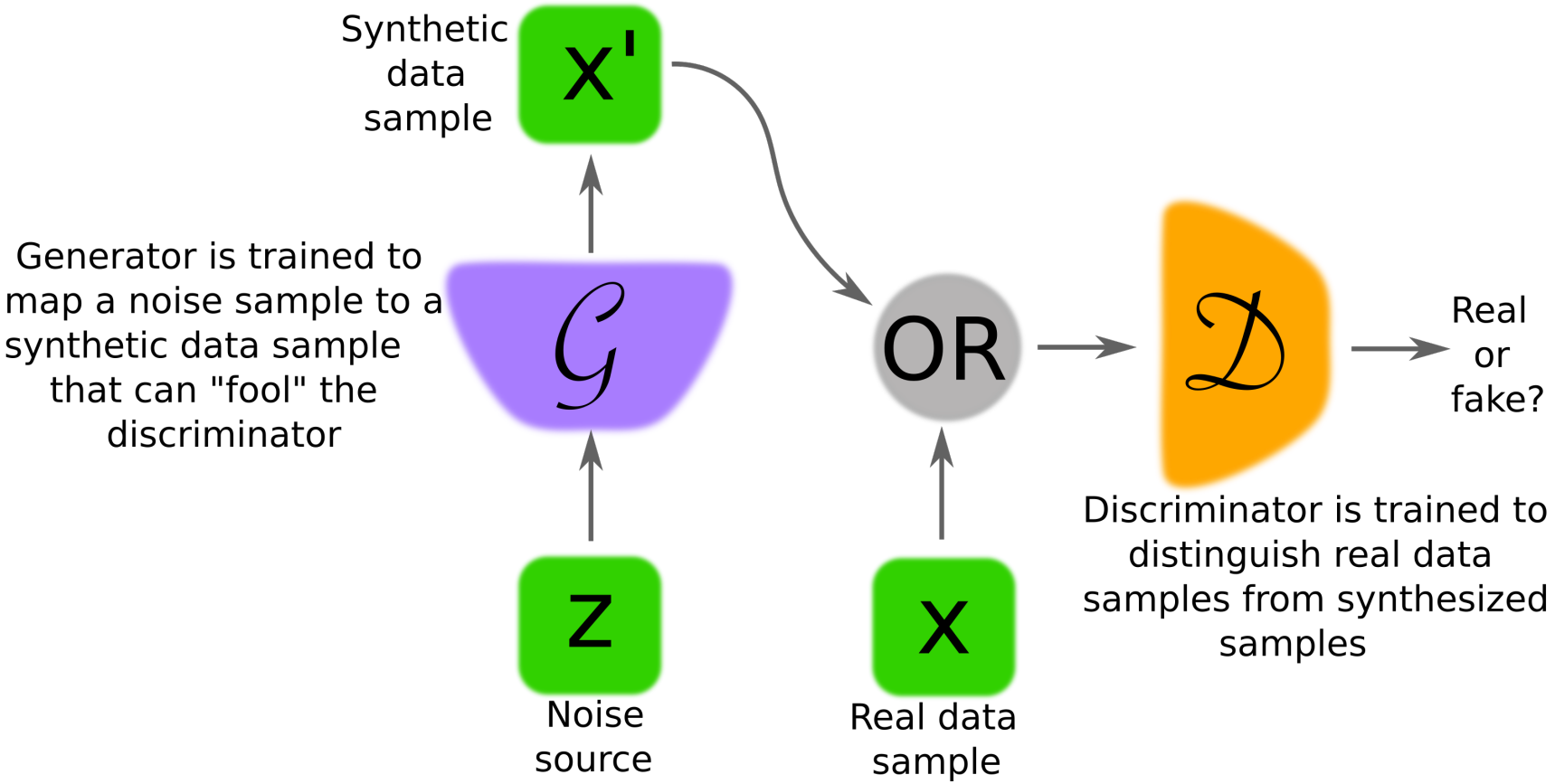
Variational Auto-Encoders Networks

- Intuitively: we introduce a restriction on z such that our data point x is distributed in a latent space (manifold) following a specified probability density function Z (N(0,1)).



https://blog.fastforwardlabs.com/2016/08/22/under-the-hood-of-the-variational-autoencoder-in.html

Generative Adversarial Networks

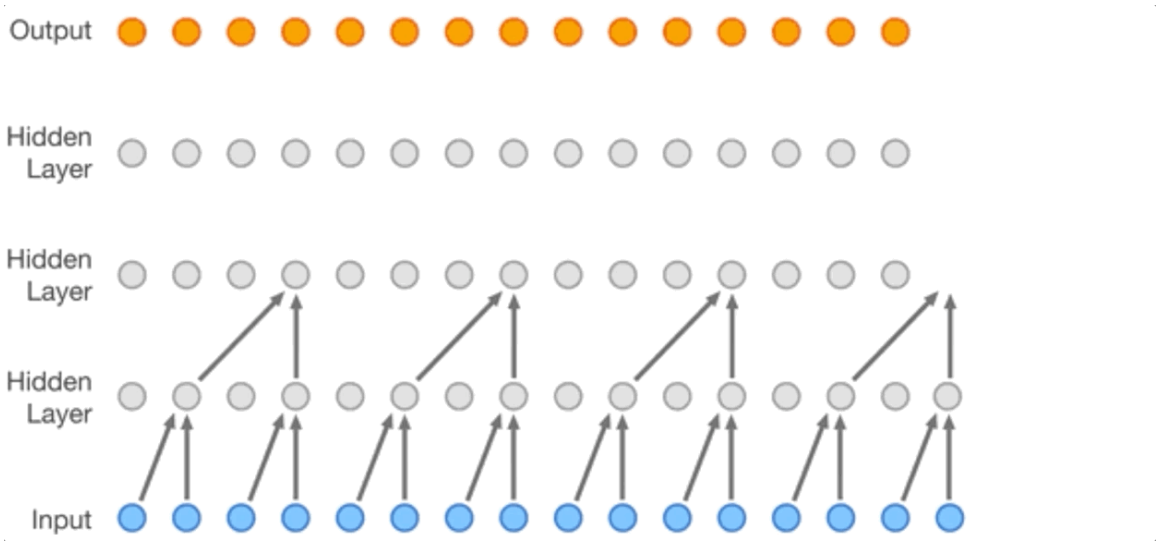


https://tinyurl.com/y2vl6ulm

Auto-regressive Networks

- The basic difference between Generative Adversarial Networks (GANs) and Auto-regressive models is that GANs learn implicit data distribution whereas the latter learns an explicit distribution governed by a prior imposed by model structure.

- Deep autoregressive models are:
    - sequence models, yet feed-forward (i.e. not recurrent);
    - generative models, yet supervised.
    - a compelling alternative to RNNs for sequential data, and GANs for generation tasks.

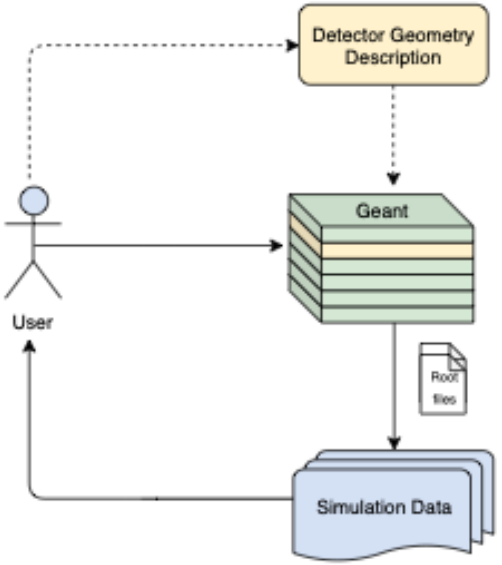


https://deepmind.com/blog/article/wavenet-generative-model-raw-audio

Data Production

- Based on Geant4 validation tools

- One event (or data point) consists of N x N x M cells
  - N on (xy) plane, M along z axis
  - Final data point size is 24x24x24



- Each cell can be built of K absorbers
  - TestEm3 inspired
  - K = 1 for homogeneous calorimeters (PbWO4)
  - Other geometries:Pb/LAr, Pb/Sci,W/Si

- Detector messenger to set size, number of cell, materials, sensitivity

- Current cell size: ~ $1X_0$ on z and ~0.5 R_M in xy

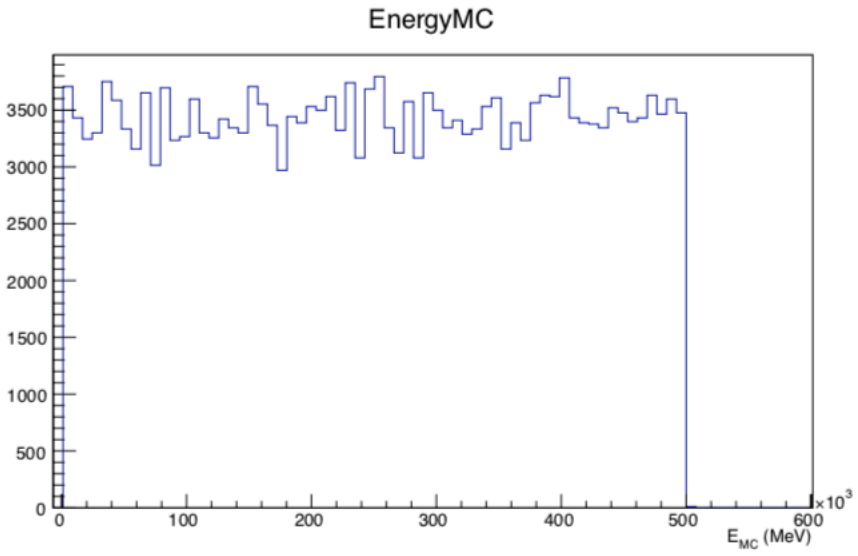
Data Production

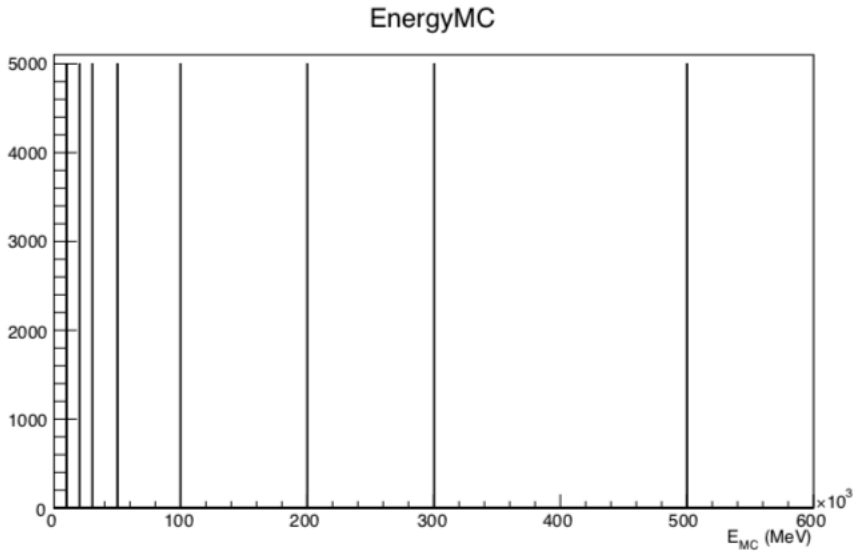
- For ML training:

- Flat energy spectrum (1-500 GeV) of particle gun along z axis

- For validation/ analysis/ comparison:

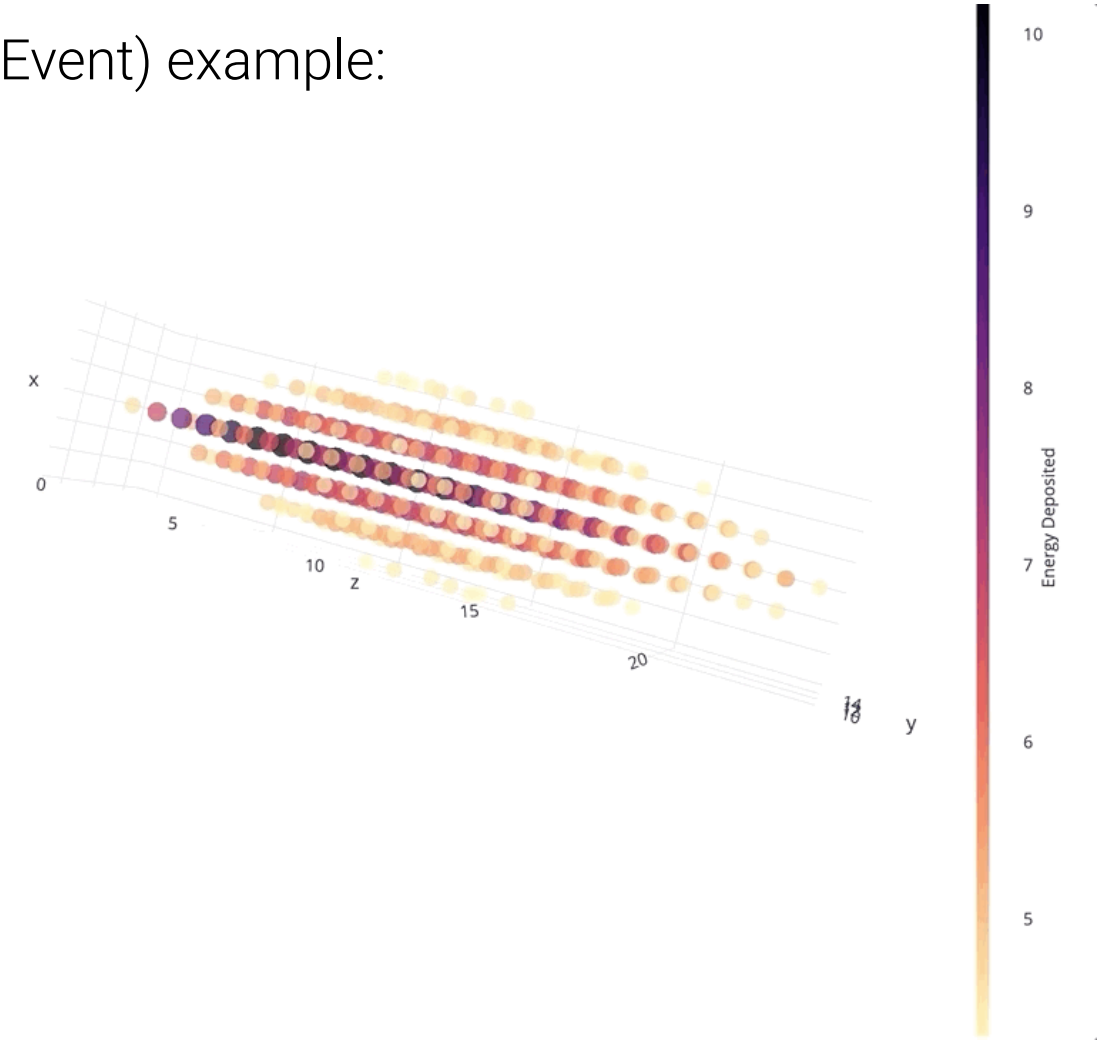
- Single energy gun along z axis





Data Production

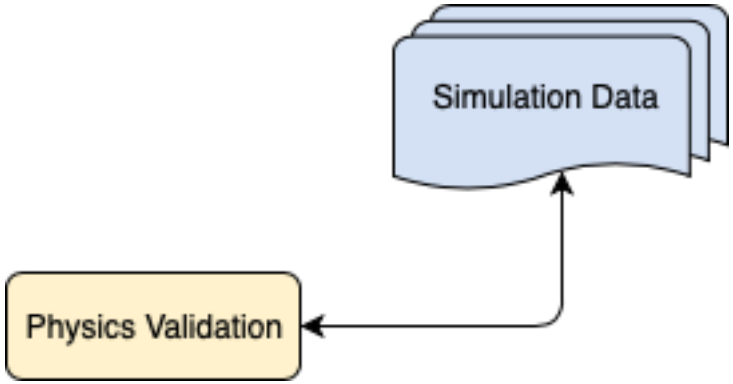
- Data Sample (Event) example:



Validation

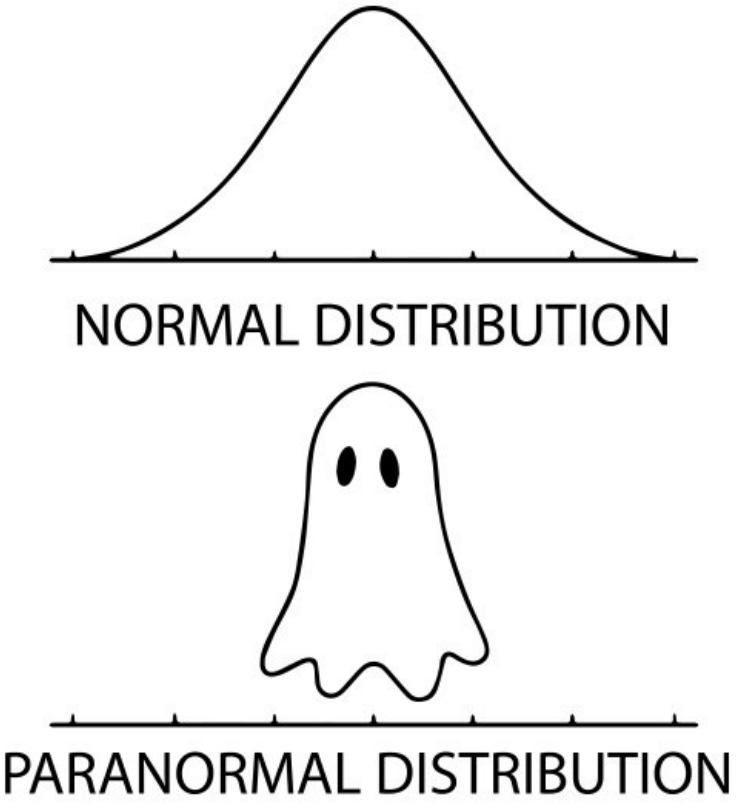
A set of general validation histograms is created:

- MC energy
- Deposited energy
- Number of cells above threshold (currently Ecell > 0.1 MeV)
- Cell Energy Distribution
- Longitudinal and transverse profiles (and first/second moments)
- Energy distribution layer-wise
- Transverse profile layer-wise
- Simulation time



Network Implementation - Overview

- We use an explicit model with tractable density



Network Implementation - Intro

- Employ the chain rule to decompose the likelihood of an event x into a product of 1-d distributions:

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

Likelihood of event x

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

- We then maximise likelihood of training data

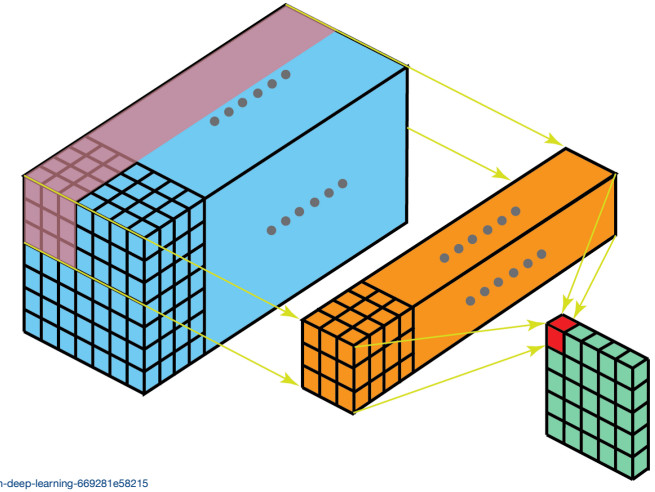
- We express the complex distribution over cell values using a neural network

Network Implementation - Intro

- For our case, cells are conditioned on all cells form the layers before (z0…z(i-1)) and subsequent data is blocked with the help of masks

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

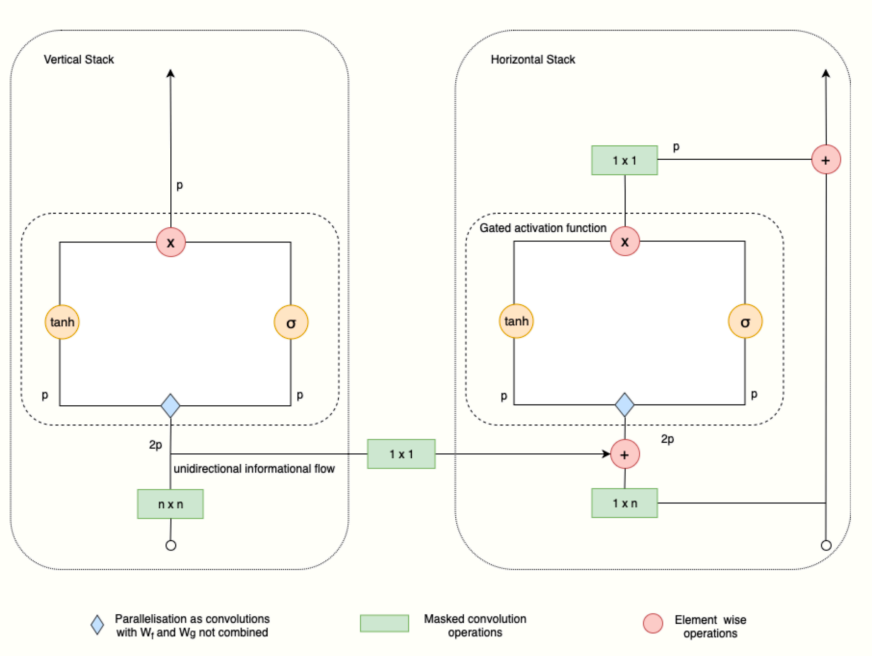
- The training is done by maximising the likelihood of training data - at each cell location we have the ground truth (training data point value at location)



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

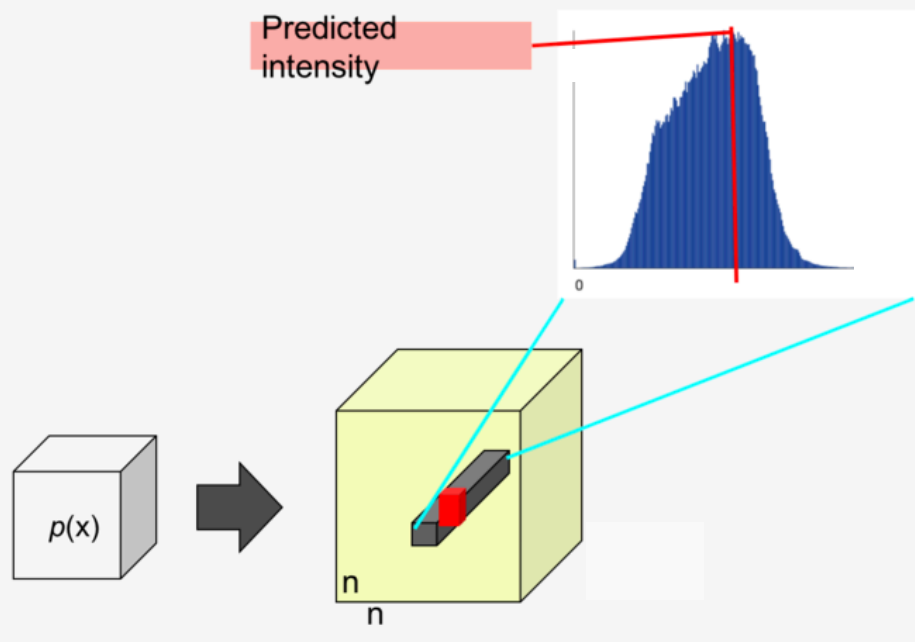
Network Implementation - Intro

Conditioning on incoming particle energy - During training we feed the cell depositions as well as the MC Energy to the network to ensure that the network learns to incorporate that information. During inference we specify which energy input the output event should generate.



Network Implementation - Intro

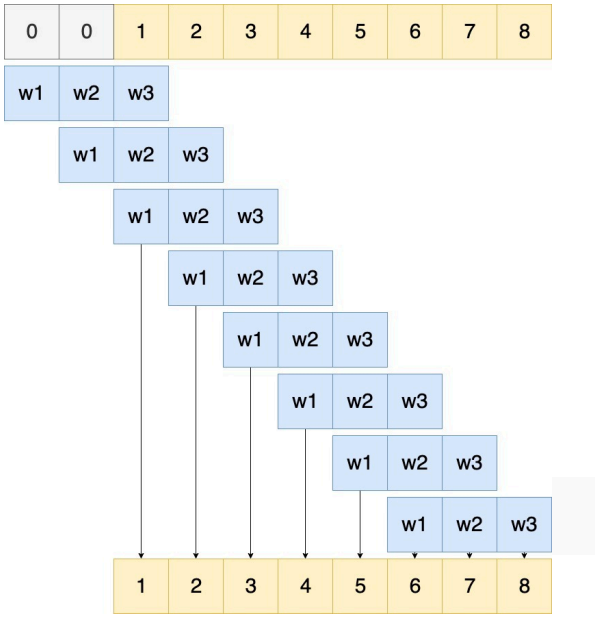
The convolution is split into two different operations: two separate stacks - vertical and horizontal. Where vertical stack has no access to horizontal stack information. Vertical feeding the information from previous z layers and horizontal the information on the current layer up until the interest cell.



Network Implementation - Intro

- we enforce a causal convolution by padding our cells sequence to the left with (the width of the convolution kernel - 1) zeros

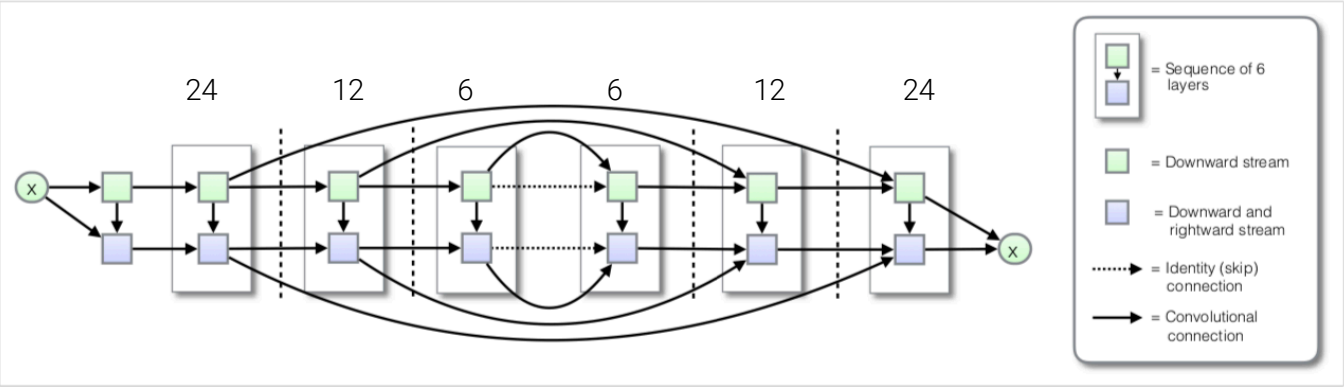
- we thus emulate the behaviour of a recurrent neural network with a convolutional neural network in order to parallelise the computations and control the access to past information



Network Implementation - Intro

- we allow the network to capture long range structure by downsampling with convolutions of stride 2 (we retrieve lost information with the help of short-cut connections)

- we follow a convolutional architecture with residual connections with downsampling and upsampling and long-range skip-connections with each downsampling layer providing direct input to the corresponding layer in the upsampling module



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

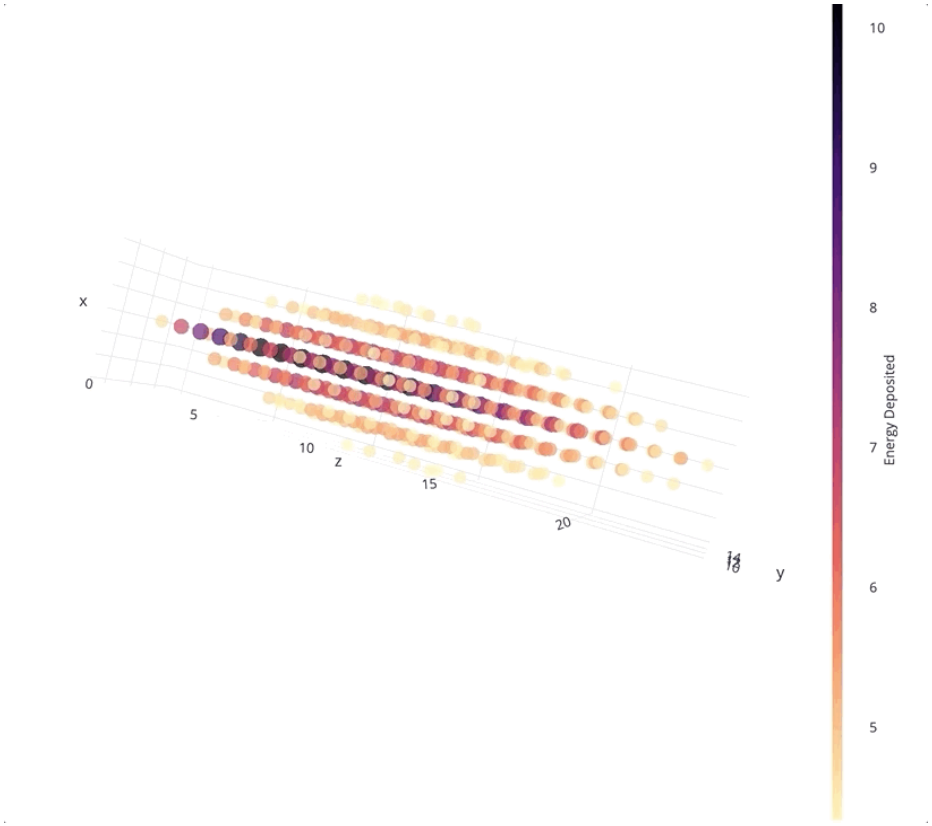
Network Implementation - Overview

- High level data (energy depositions, incident particle energy) description as a latent vector h

- The latent vector models the conditional distribution p(x|h) of energy depositions

- Conditioning is dependent on the coordinates of the cell

- Increase the global structure performance with Autoregressive Channel: skip connections between sets of channels via 1x1 convolution gated ResNet block*



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

Auto-regressive Networks - Advantages

1. Provide a way to calculate likelihood : these models have the advantage of returning explicit probability densities

2. Training is more stable than GANs : training a GAN requires finding the Nash equilibrium, making it unstable as compared to auto-regressive networks

3. Employed for both discrete and continuous data

https://towardsdatascience.com/auto-regressive-generative-models-pixelrnn-pixelcnn-32d192911173

Existing Applications

- PixelCNN by Google DeepMind - probably the first deep autoregressive model. The authors discuss a recurrent model, PixelRNN, and consider PixelCNN as a "workaround" to avoid excessive computation.
- PixelCNN++ by OpenAI is essentially PixelCNN but with various improvements.
- WaveNet by Google DeepMind is heavily inspired by PixelCNN, and models raw audio, not just encoded music. Telecommunications/signals processing tools are used for coping with the sheer size of audio (high-quality audio involves at least 16-bit precision samples = 65,536-way-softmax per time step)
- Transformer, the "attention is all you need" model by Google Brain is now a mainstay of NLP, performing very well at many NLP tasks and being incorporated into subsequent models like BERT (Bidirectional Encoder Representations from Transformers).
- Google DeepMind's ByteNet can perform neural machine translation (in linear time!) and Google DeepMind's Video Pixel Network can model video.

Work in progress and Observations

- Work in progress on both existing models testing and customised network development

- Exploring different geometries and understanding their influence on the results

- Auto-Regressive training is supervised which means that training is stable and highly parallelizable, it is straightforward to tune hyper-parameters, and that inference is computationally inexpensive. Also, all principles from ML-101: train-valid-test splits, cross validation, loss metrics, etc. can be employed (things that we lose when we resort to other models)

- 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