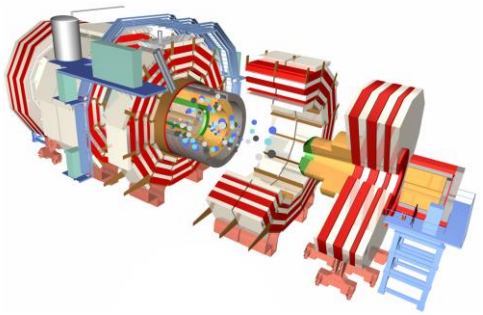
Fast Simulation for HEP Experiments

- Deep Learning Techniques -

Ioana Ifrim

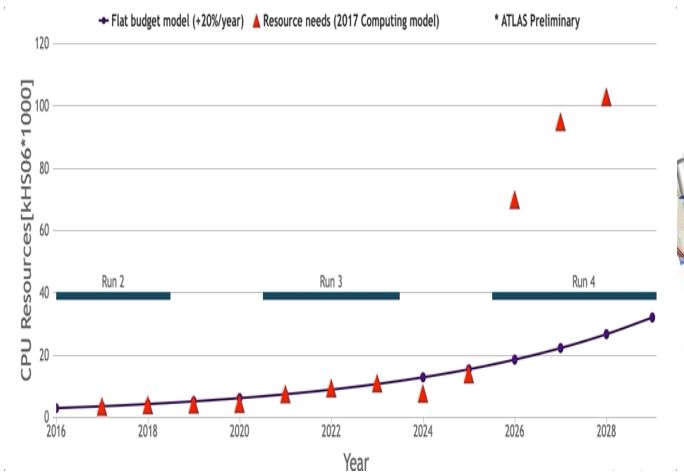
What? Why? How?

Full Simulation of Physics Detectors



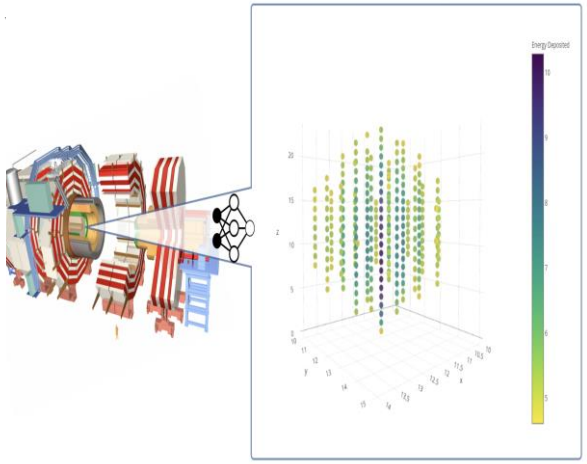
One particle at a time propagates through the physics detector and energy
depositions in (x, y, z) are recorded at every step. This then corresponds to the experimental output

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

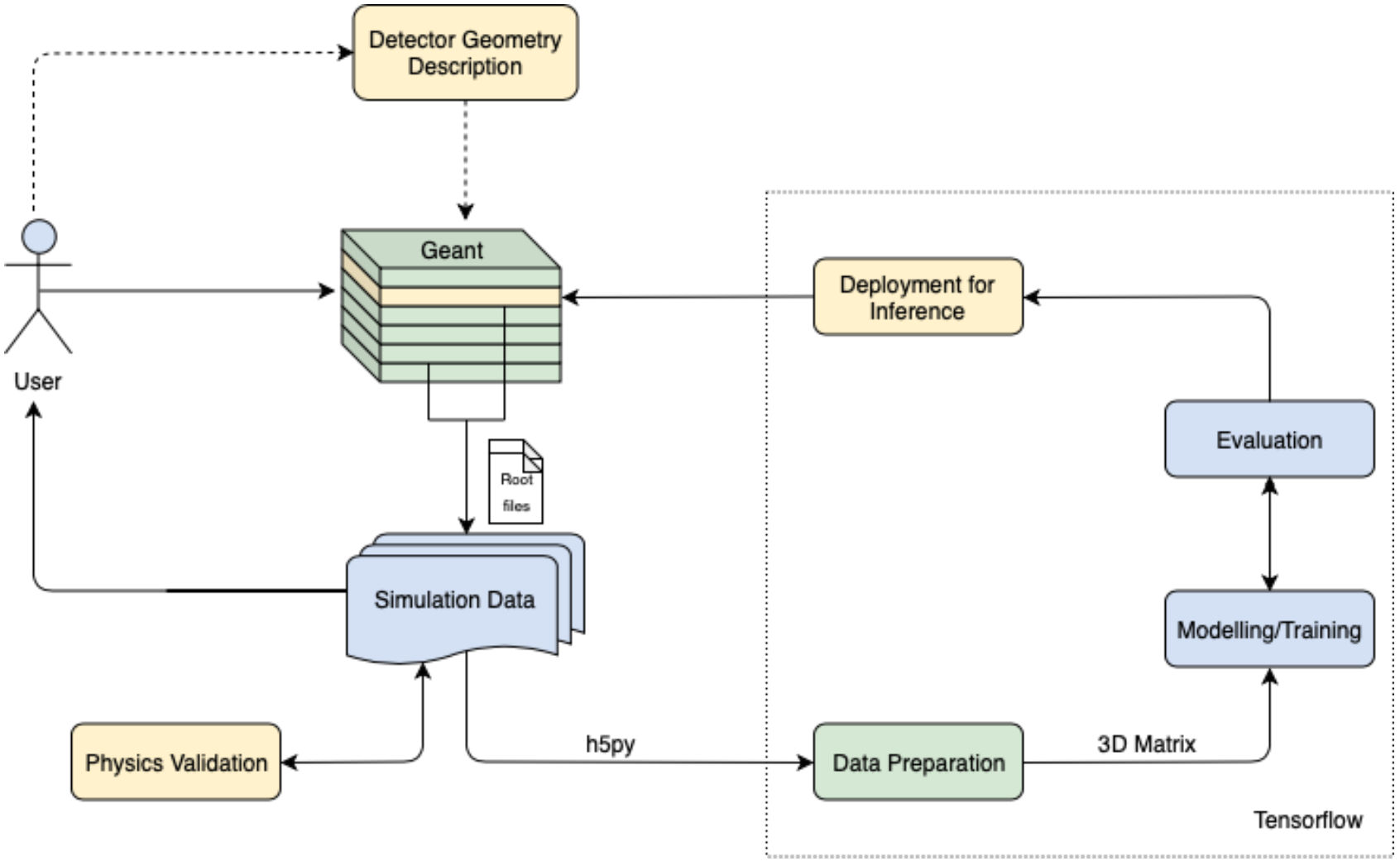
Fast Simulation



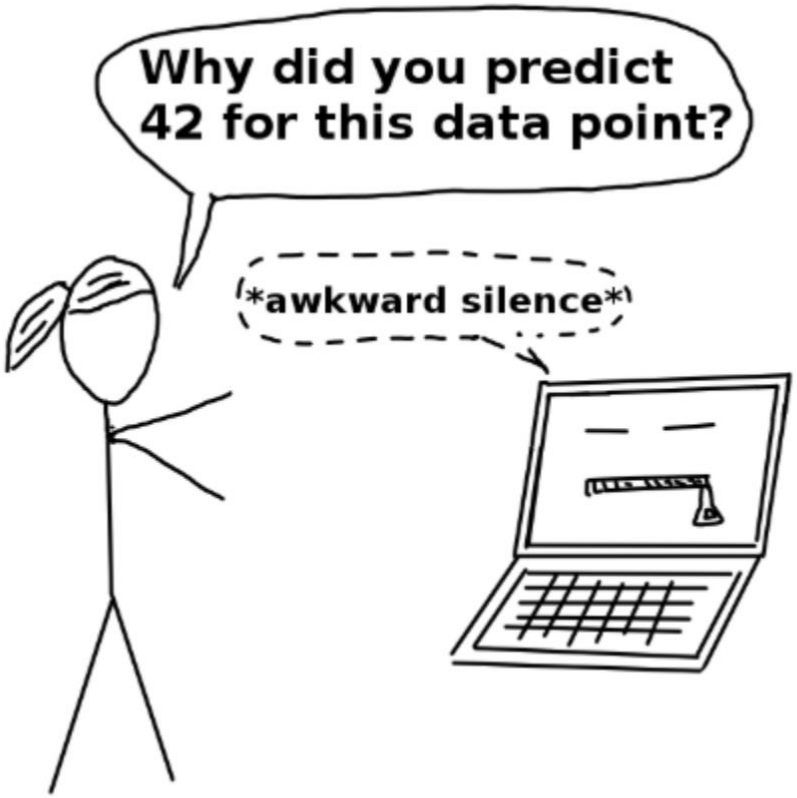
Using Deep Learning principles in treating physics data we can generate the simulation output (energy depositions) in a fast manner (2-3 orders of magnitude faster than full simulation) resembling the Machine Learning task of image generation

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

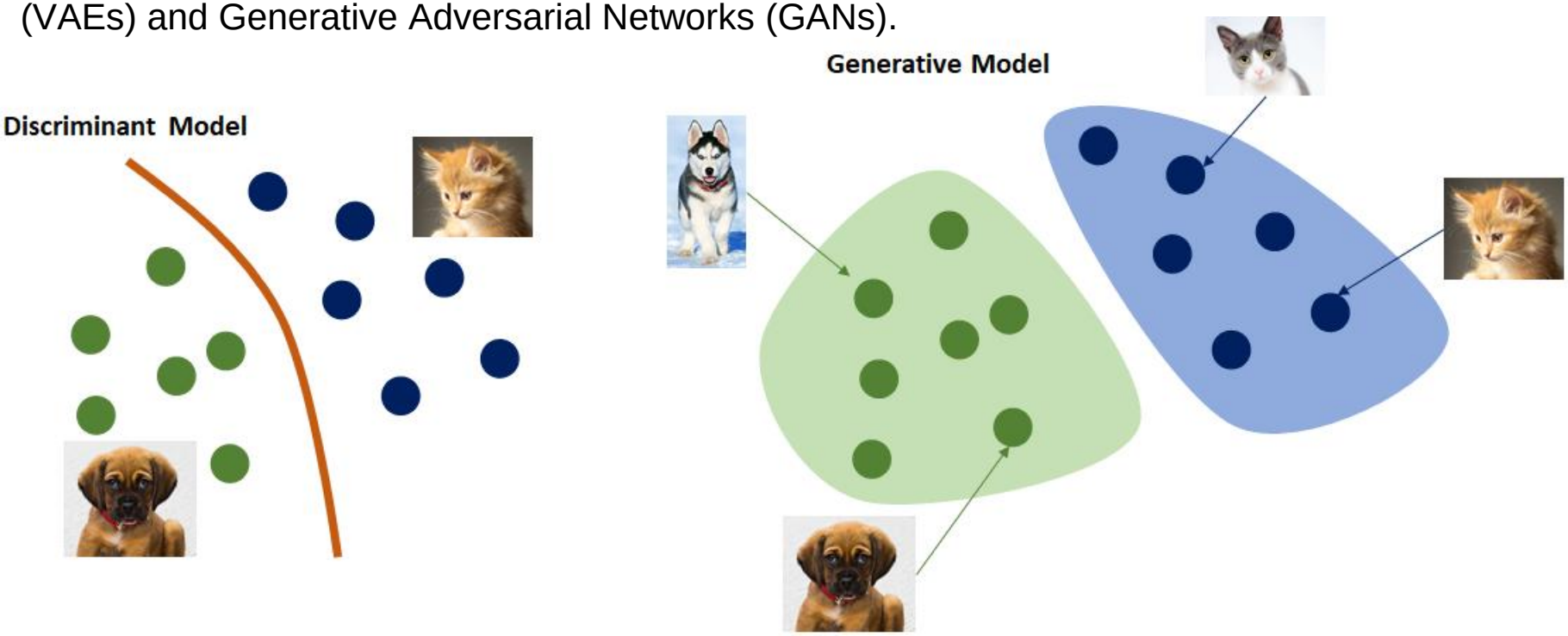


Source: interpretable-ml-book

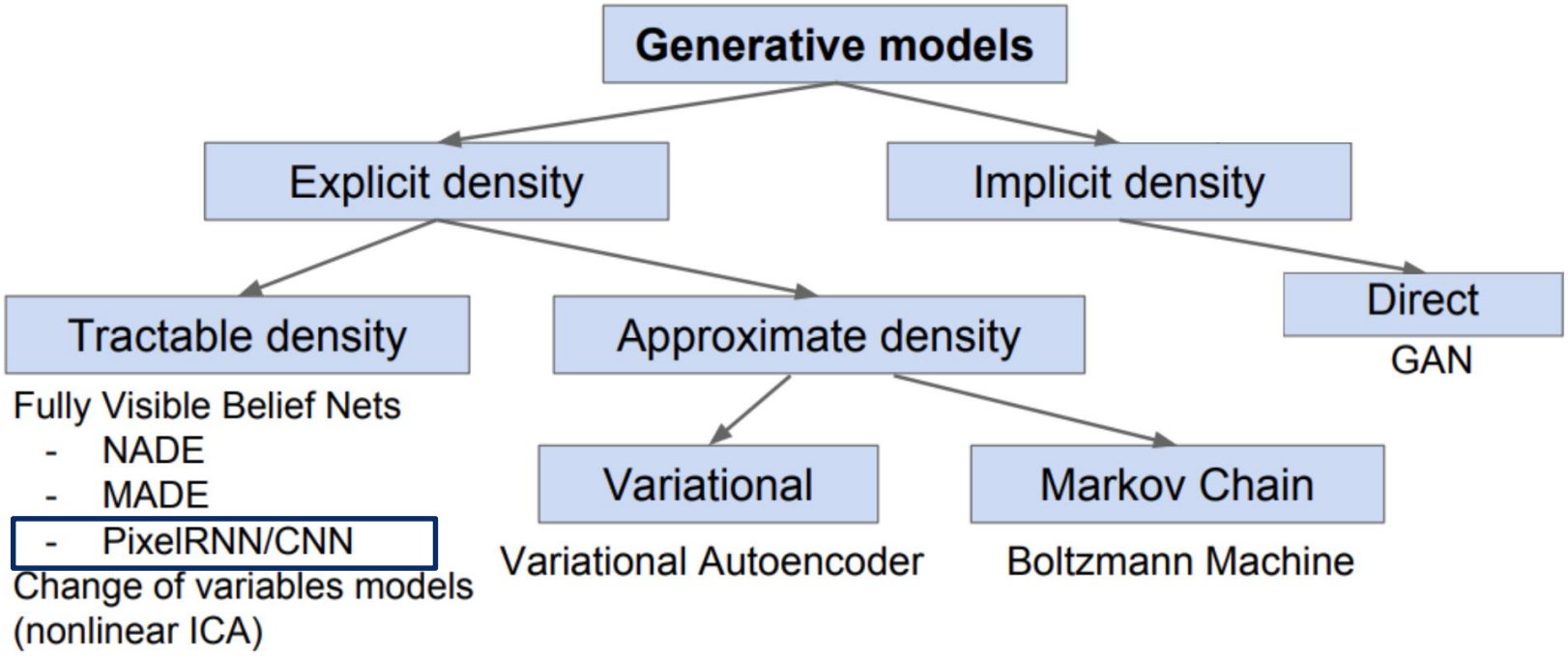
Generative Models

- 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.

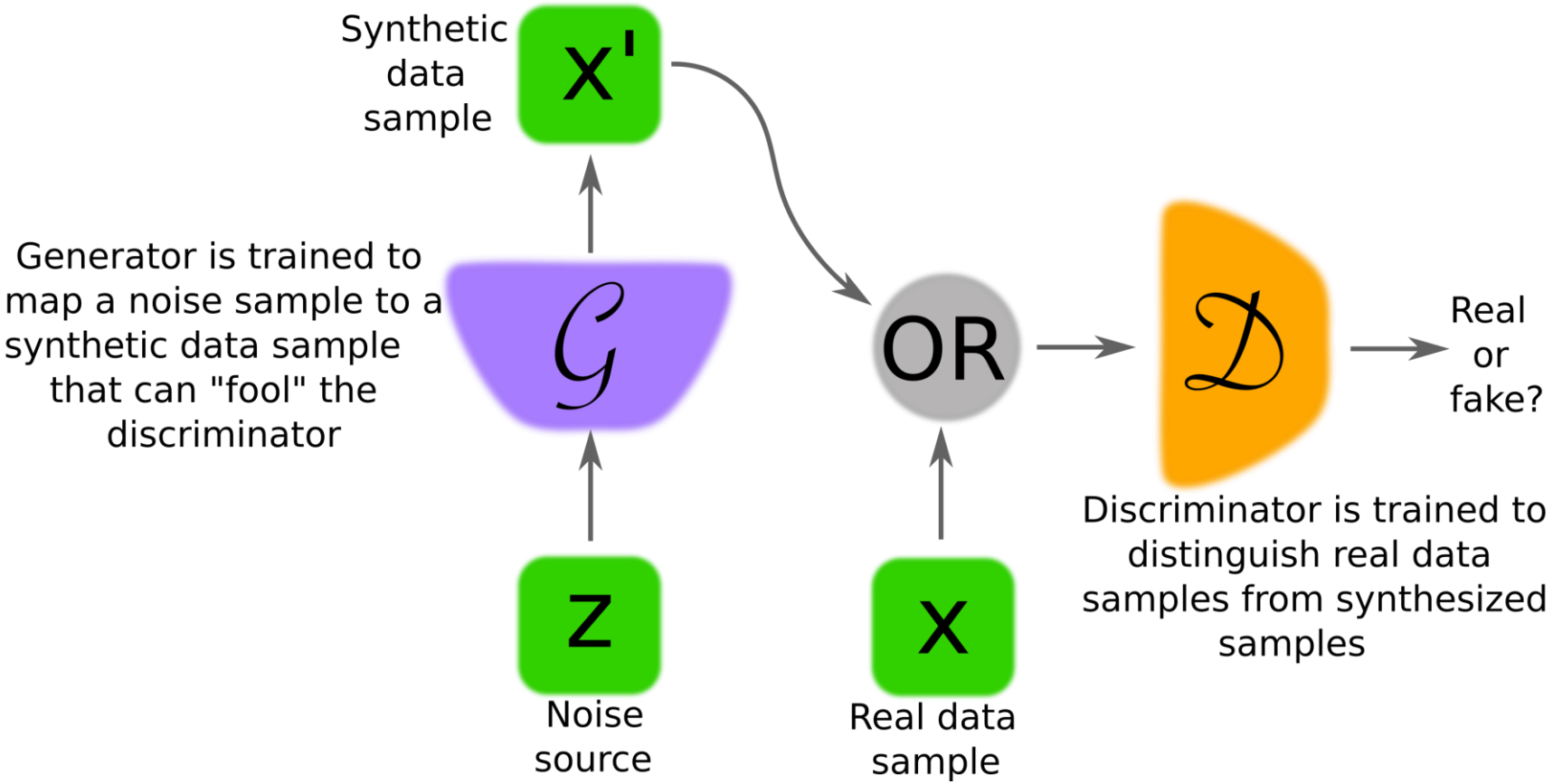
- Two ways to model the distribution (explicit and implicit density). The most efficient and popular of generative models are: Auto-Regressive models, Variational Auto-Encoders (VAEs) and Generative Adversarial Networks (GANs).



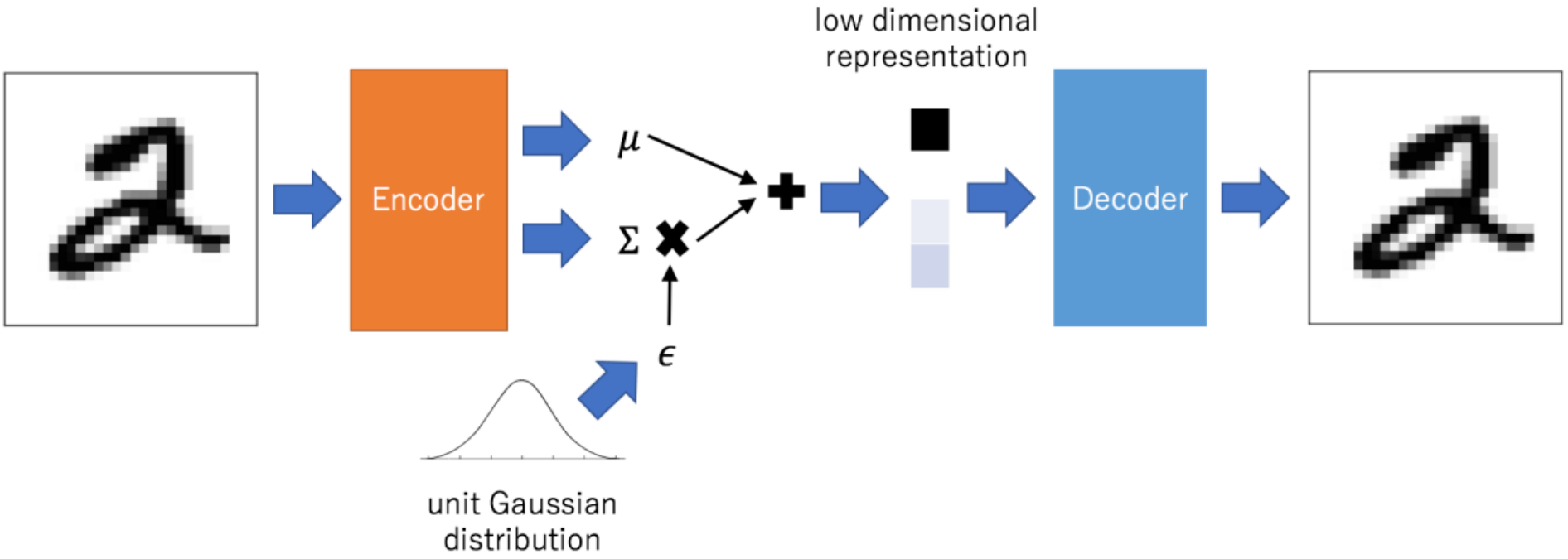
Generative Models



Generative Adversarial Networks



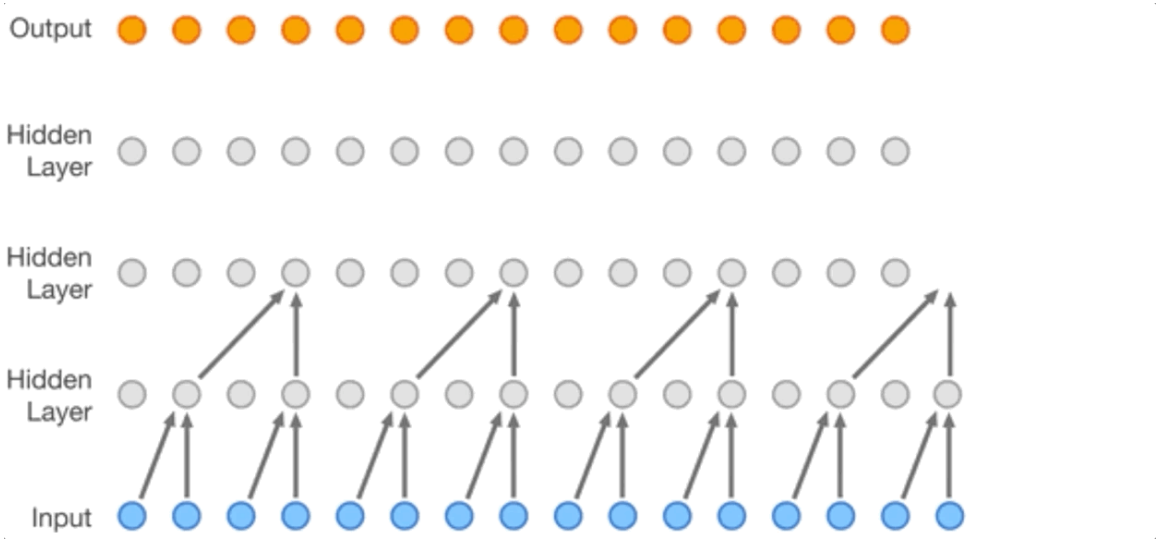
Variational Auto-Encoders Networks



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.

- tldr: 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.

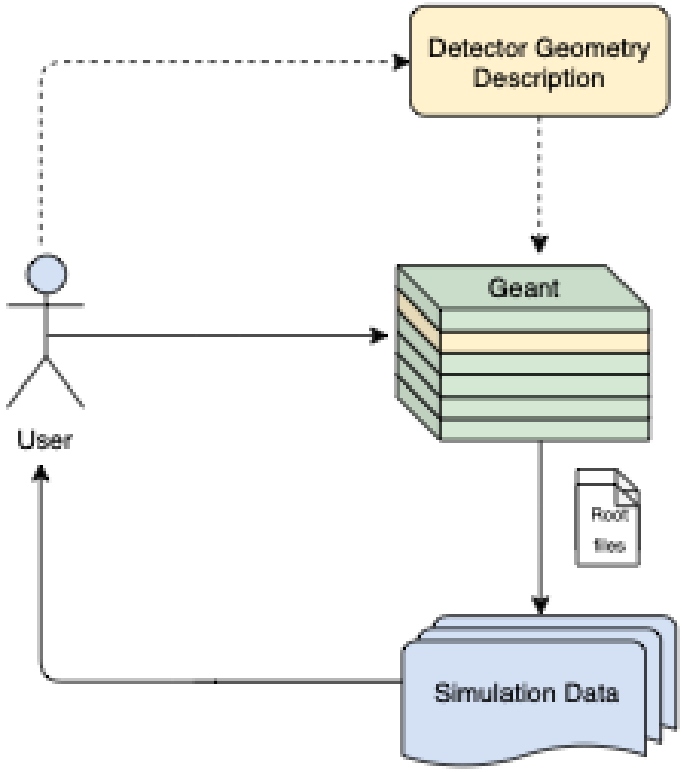


Data Production

- Based on Geant4 validation tools

- Different simplified calorimeters available
  - Pb/LAr (ATLAS like)
  - $PbWO_4$ (CMS like)
  - Pb/Sci (LHCb like)
  - W/Si (ILD, CMS HGCal like)

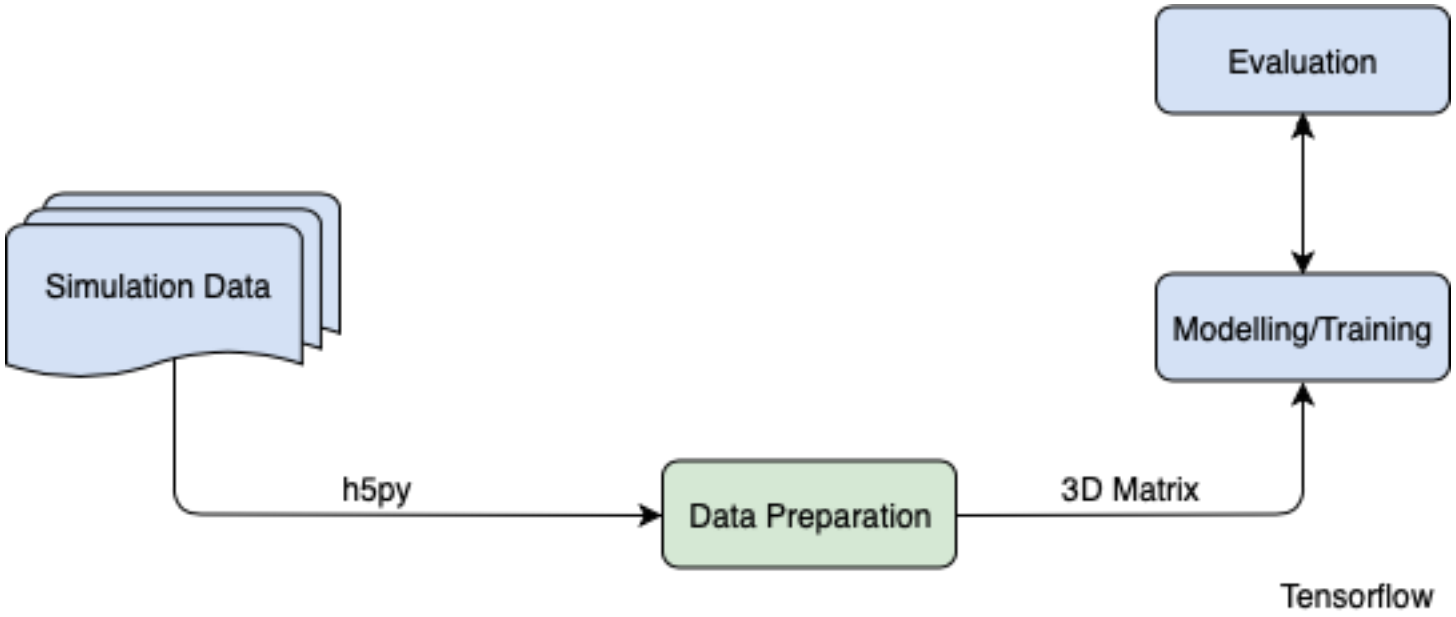
- Started with production of data sets
  - >200k events for first tests



Network Testing

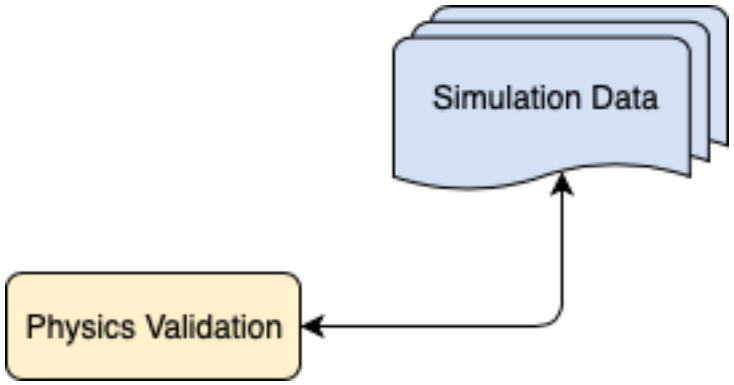
- Working on testing GANs, VAEs and Auto-Regressive Networks implementations for the different datasets

- Focusing on developing a proper Auto-Regressive network for HEP domain data



Validation

- Based on existing validation tools and plots used by ML FastSim community
    - Visible Energy
    - Energy/cell (layer)
    - Shower profiles (+ mean, second moment)
    - …



Auto-regressive Networks - Advantages

1. **Provide a way to calculate likelihood** : These models have the advantage of returning explicit probability densities (unlike GANs), making it straightforward to apply in domains such as compression and probabilistic planning and exploration

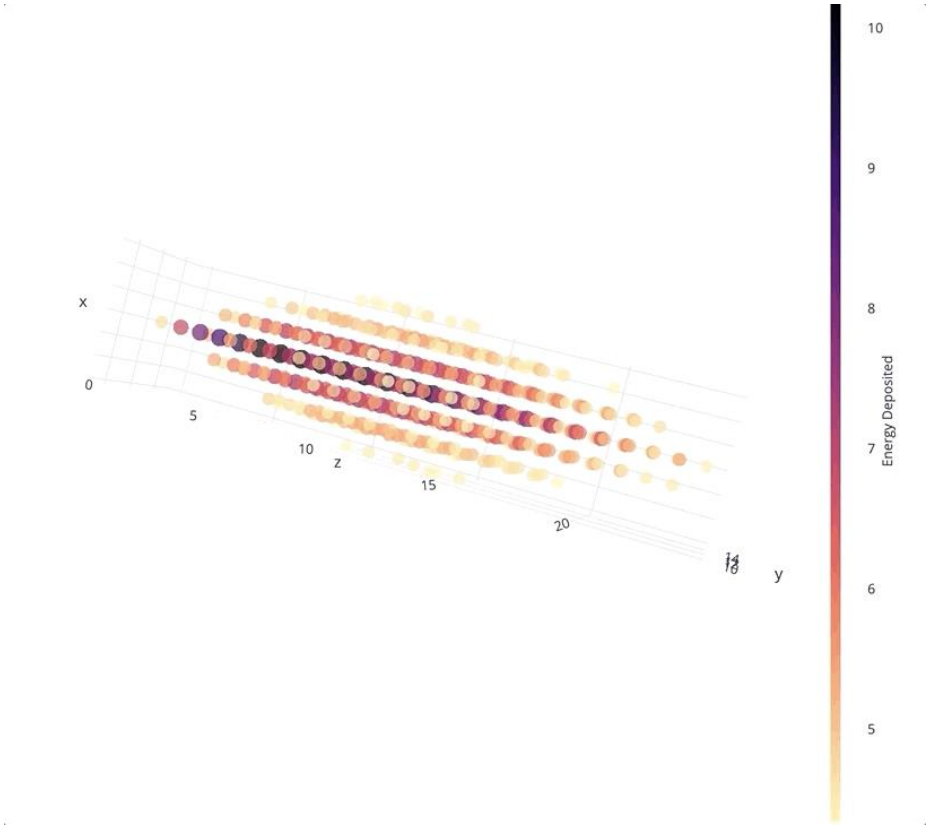
2. **Training is more stable than GANs** : Training a GAN requires finding the Nash equilibrium. Since, there is no algorithm present which does this, training a GAN is unstable as compared to auto-regressive networks.

3. **It works for both discrete and continuous data** : It is hard to learn to generate discrete data for GAN.

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

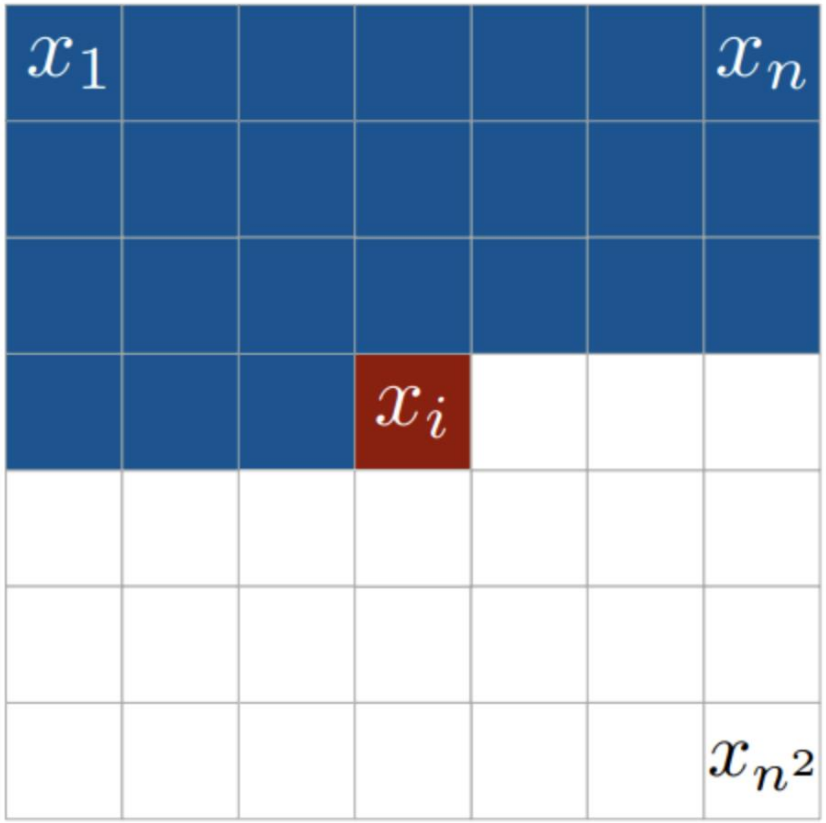
Network Implementation - Overview

- 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 pixel
- Architecture built upon PixelCNN
- Gates provide context memory and help model more complex interactions
- The vertical stack accounts for blind spots during convolution



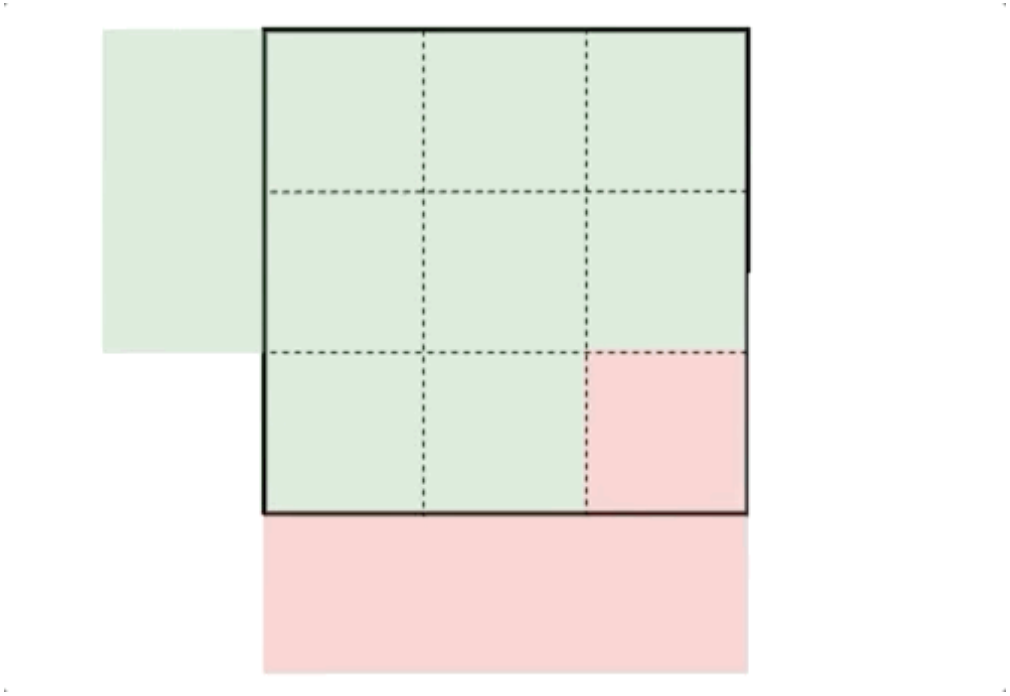
Network Implementation - Intro

Pixel xi depends on pixels x1…x(i-1). To ensure that this is the case, the use of masks is employed in order to block subsequent information.



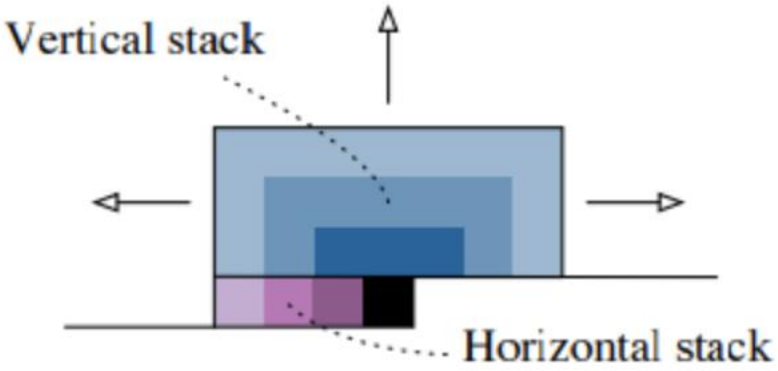
Network Implementation - Intro

The convolutional layers are unable to completely process the receptive fields thus leading to a slight miscalculation of pixel values. The pixels left out constitute the blind spot.



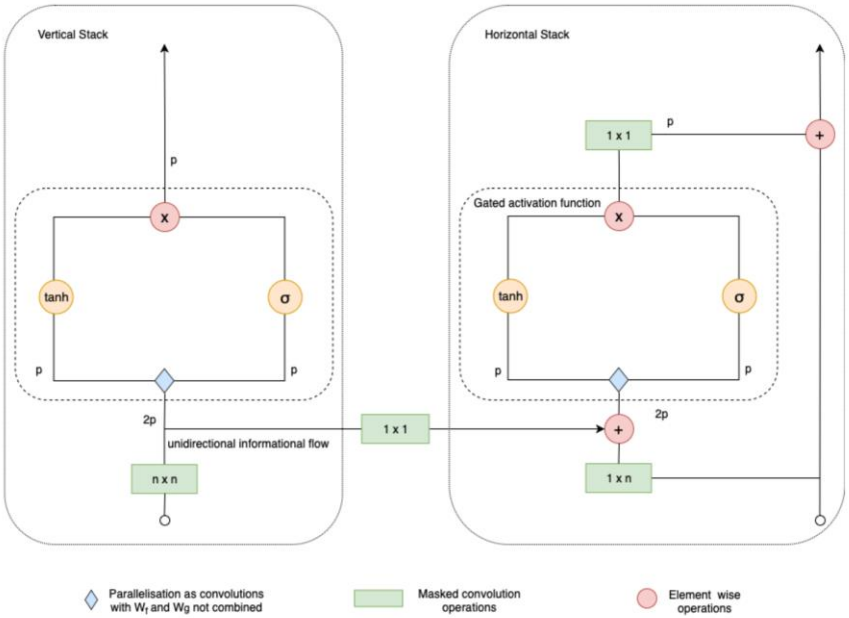
Network Implementation - Intro

The convolution is split into two different operations: two separate stacks - vertical and horizontal. Where vertical stack has no access to horizontal stack information



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

We are modelling the distribution p(x|h) of outputs following the description given by the latent vector h (the high-level image description):

$$p(\mathbf{x}|\mathbf{h}) = \prod_{i=1}^{n^2} p(x_i|x_1, ..., x_{i-1}, \mathbf{h}).$$

With the following gated, conditioned activation unit:

$$\mathbf{y} = \tanh(W_{k,f} * \mathbf{x} + V_{k,f}^T \mathbf{h}) \odot \sigma(W_{k,g} * \mathbf{x} + V_{k,g}^T \mathbf{h})$$

where k is the layer number, W the learned weights matrix, f,g the feature maps, V the matrix of shape [nr of classes, nr of filters], h a one-hot classes vector.

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.
- 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 e.g. GANs.)

- 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