Deep Autoregressive Networks For Fast Simulation

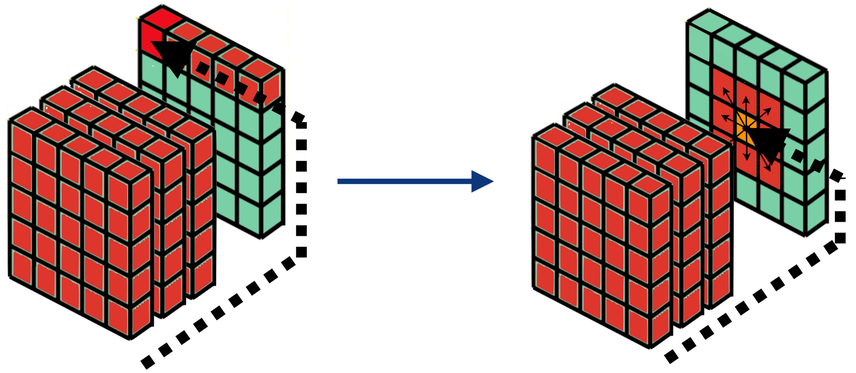
Ioana Ifrim
EP-SFT

Machine Learning Based Fast Simulation Tools - Status

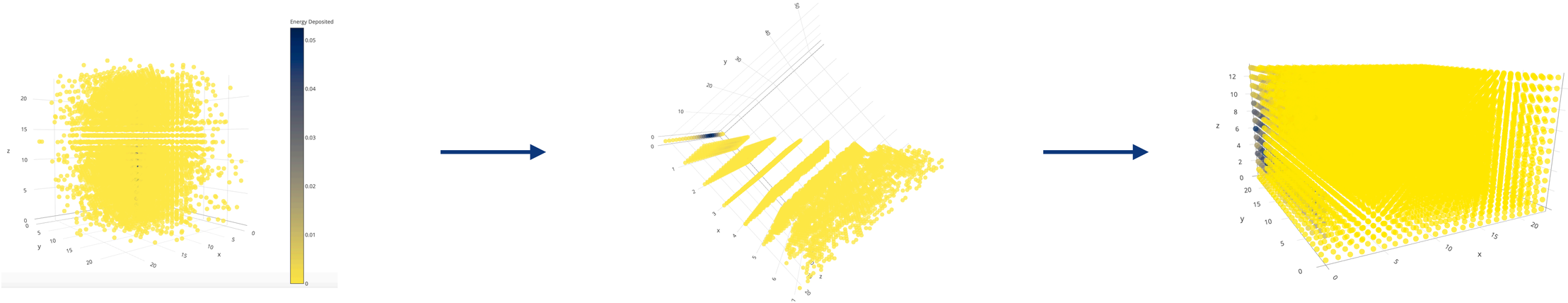
- Developed a number of testing procedures for model evaluation
- Cylindrical representation of data brought great improvements of network results
- The latest results to be presented follow a specific test on the autoregressive model based on train data of shape [24,24,14] and input energies of 10-20 GeV
- The input energy label for the inference in this test has a sensitivity of 1 GeV
- The response of the network is highly influenced by the nature of our data (real values with an increment of min 1e-6 VS typical RGB integers)
- Performance tests are still ongoing for finding the best approach in data representation to lead to generalisation (signal processing // graph networks)

Machine Learning Based Fast Simulation Tools

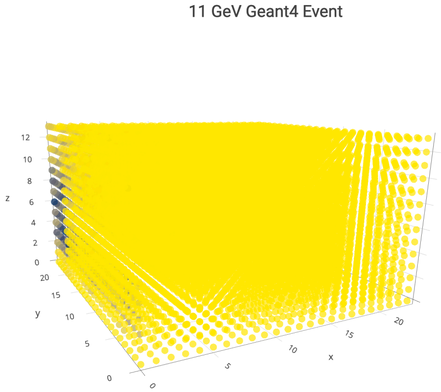
Information Propagation Changes

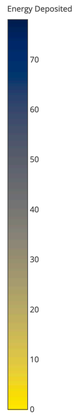


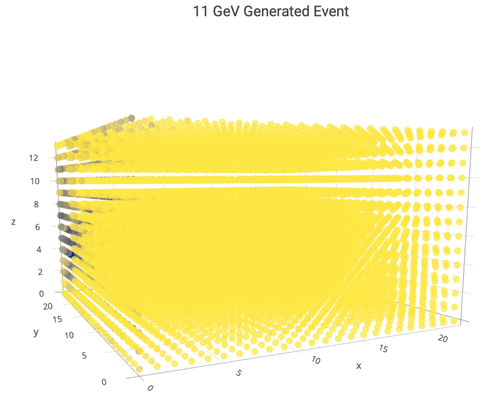
Shower Representation Changes

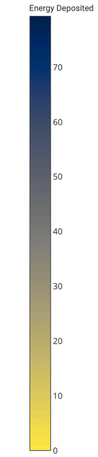


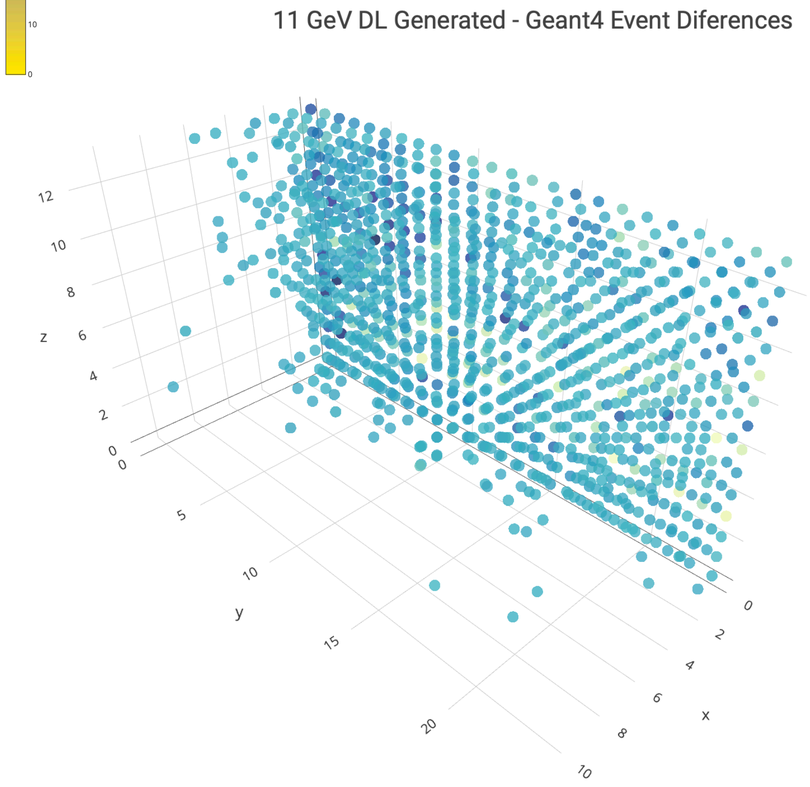
Machine Learning Based Fast Simulation Tools - Status

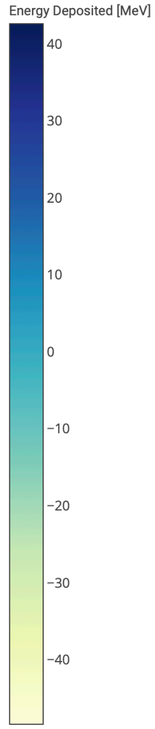




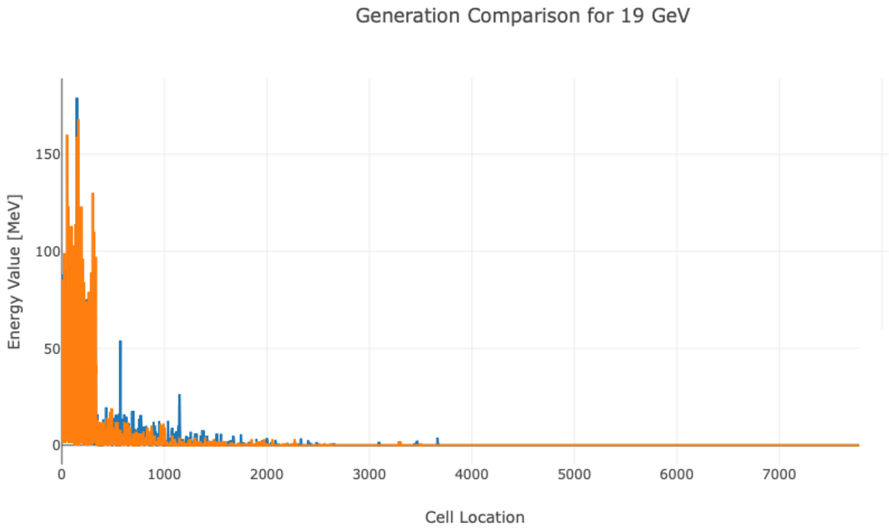


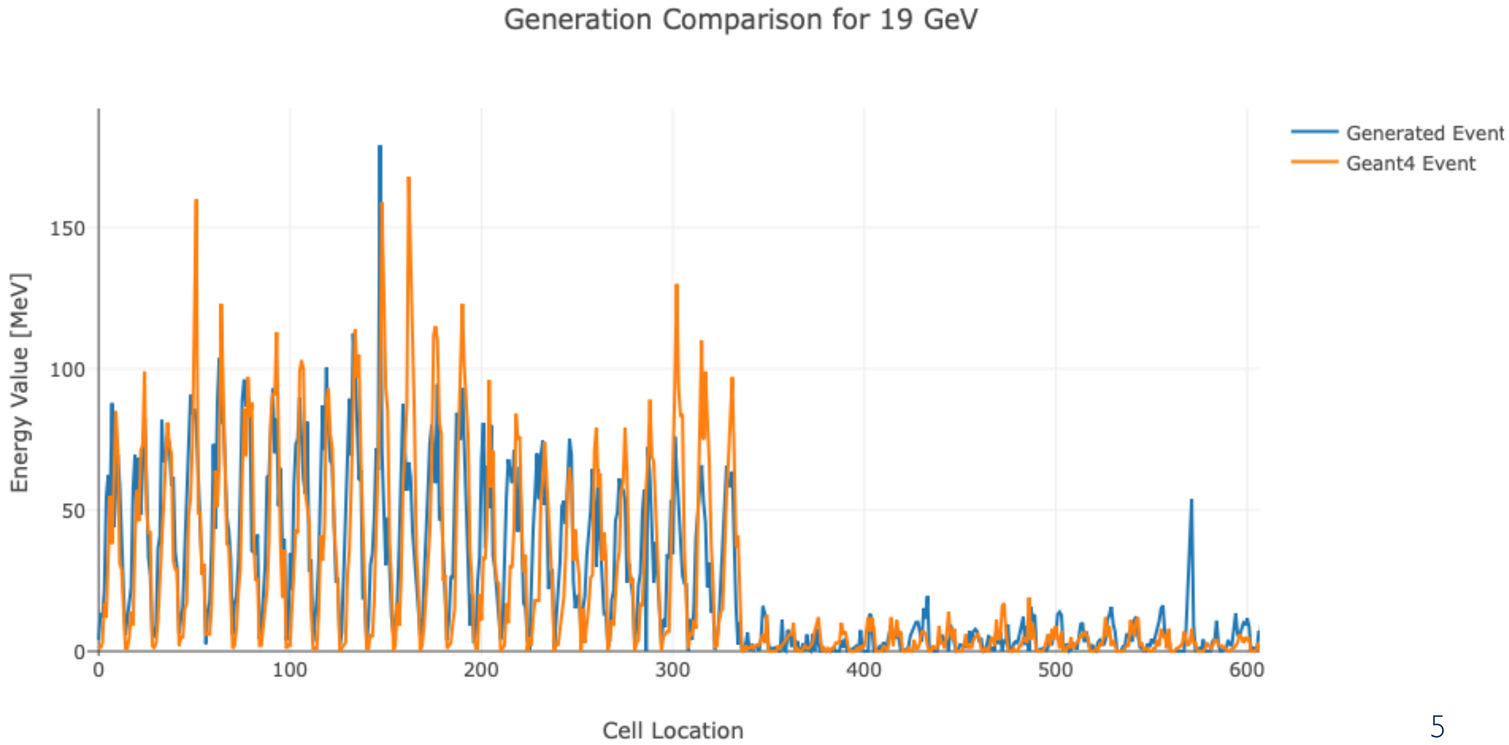




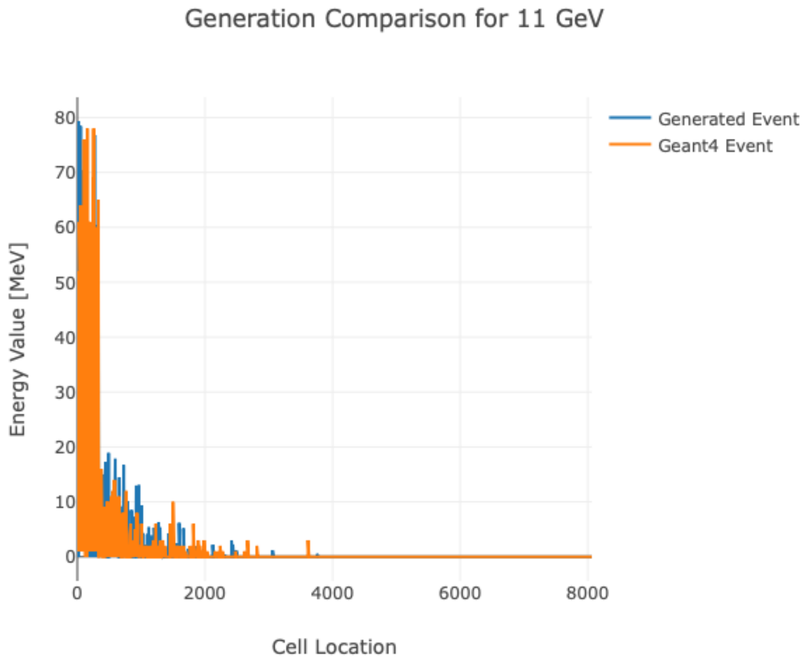


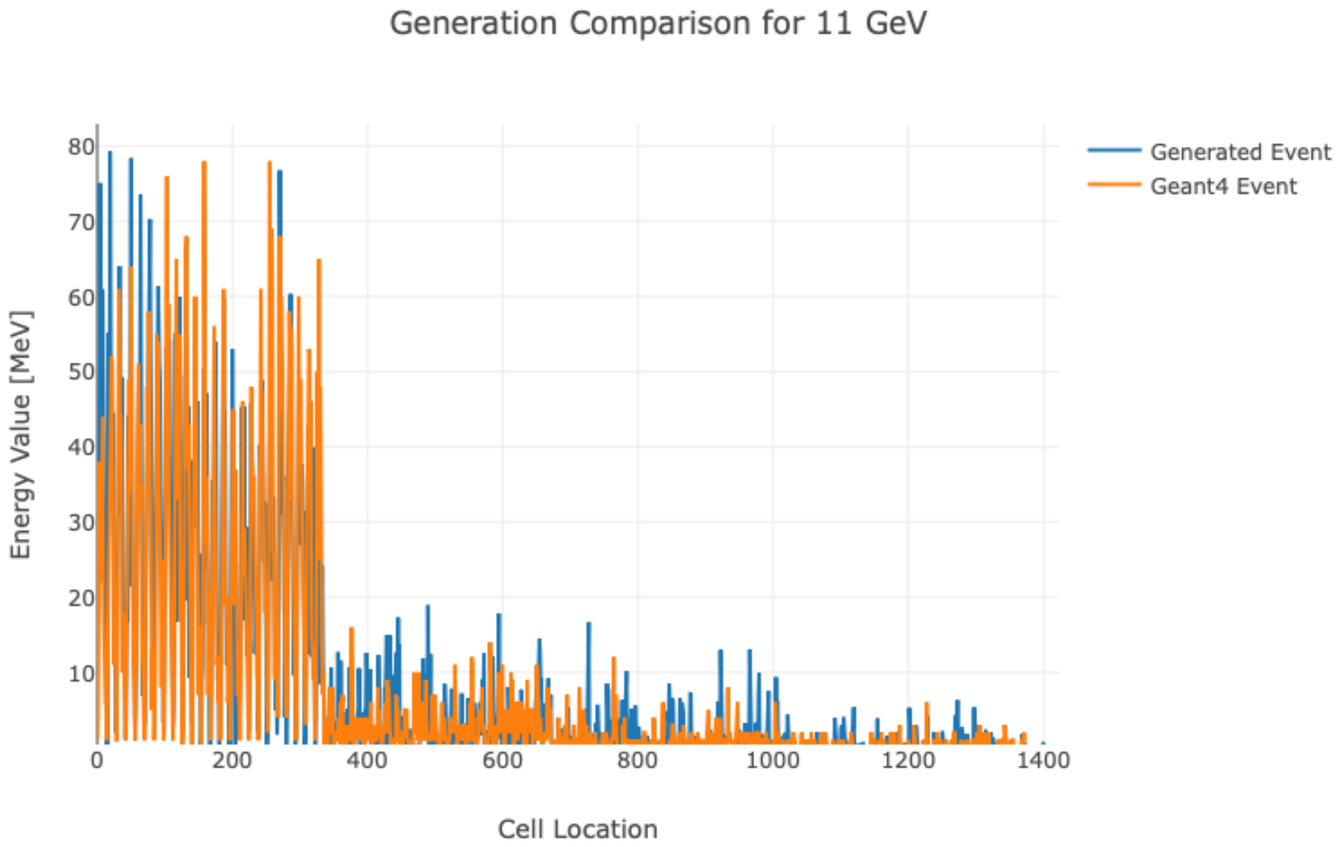
Machine Learning Based Fast Simulation Tools - Status





Machine Learning Based Fast Simulation Tools - Status





Machine Learning Based Fast Simulation Tools - Inference Status

Deep Learning Inference for Fast Simulation Applications

C++ Inference module for Generative TensorFlow Models

How To Run Inference

Download the Tensorflow C API (https://www.tensorflow.org/install/lang_c) and extract its `./lib/` contents to `./modules/all/`

Moreover, you can run inference for your choice of model and energy input:

```
cd module/ensemble/
mkdir build
cd build
cmake ..
make .
./dlinf modelChoice energyValue
```

where `modelChoice` can be either `dcgan` , `cvae` , `ar`

How To Integrate Your Model

1. Save your input/output node names. For example, given a Python model:

```
# Event Data Innputs

x_sample = tf.placeholder(tf.float32, shape=xs, name='input_cells')
y_sample = tf.placeholder(tf.float32, shape=xs[0], name="input_labels")

# Generated Result

generation = tf.add(a, b, name='output_result')
```

2. Store the graph definition in a `.pb` file as well as the latest checkpoint in `.ckpt` files (`.data` , `.index` , `.meta`)
3. Note your input data shape information (both for samples and labels).

Model Integration Info File Example

```
modelType       =  "dcgan"
modelGraph      = "../dcgan.pb"
modelRestore    = "../model.b32.ckpt"
inputNode       = "input_cells"
labelNode       = "input_labels"
outputNode      = "output_result"
inputShape      = {64,8,8,24}
labelShape      = {64,100}
```