AutoML to tune VAEs

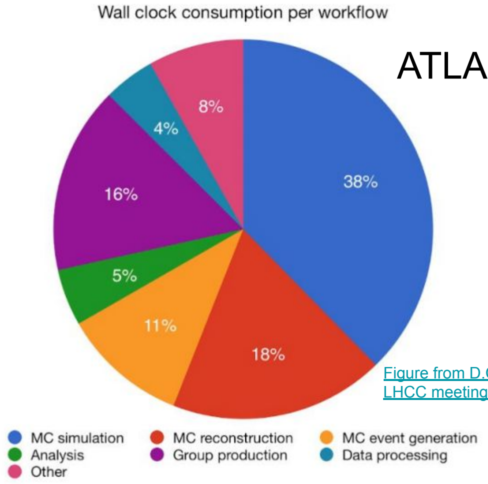
Poliana Nascimento Ferreira
Supervisor: Dalila Salamani

Summer Student Project 2021 - AutoML for Fast Simulation
ML4Sim Topical Meeting - 28/10/2021

Motivation

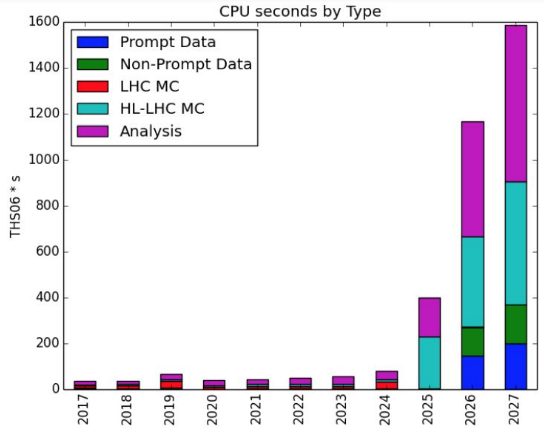
- Successful physics programs depend on the availability of Monte Carlo simulated events;
- Simulations, and shower simulation in the calorimeter in particular, are a large part of CPU consumption in the experiments;
- An alternative: fast simulation approach using Machine Learning;
- **Challenge**: How to optimize the hyperparameters of these models automatically.

Reference



ATLAS

Figure from D.Costanzo, J.Catmore, LHCC meeting



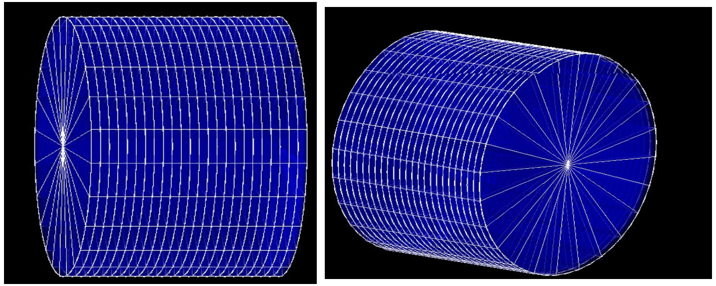
CMS

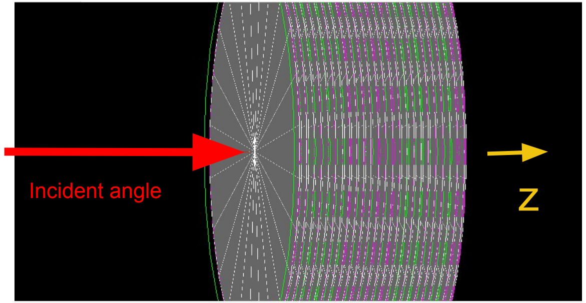
Figure from J.Phys.:Conf.Ser.1085 022006

Context - Shower Simulations

- The **calorimeter** is segmented into layers (z), and in radial (r) and azimuthal angle (phi);
- Incoming particle hits the calorimeter and generates **secondary particles**;
- Showering process: Cascade of **energy deposition** along the calorimeter layers;
- For the simulation, one shower in a layer can be seen as an **image**;
- Currently, the main method used is the Geant4 Monte Carlo simulation.

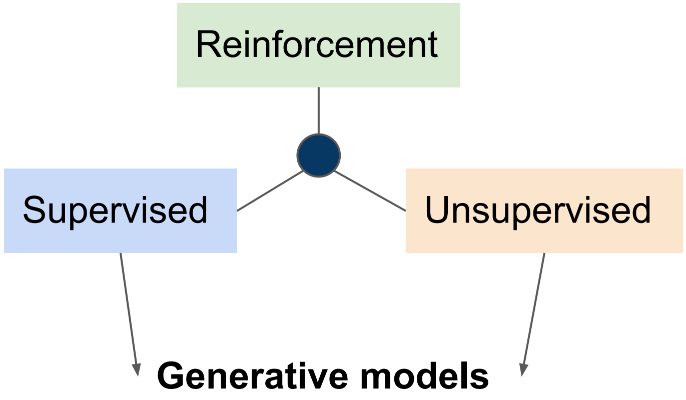
PBWO4 Geometry with 24x24x24 cell segmentation





Context - Machine Learning Techniques

- Machine Learning: Learns to improve performance by experience;
- Generative Models
  - Learn the true data distribution of the training set to reproduce it;
  - From noise, generate new data;
- Variational Autoencoder (VAE)

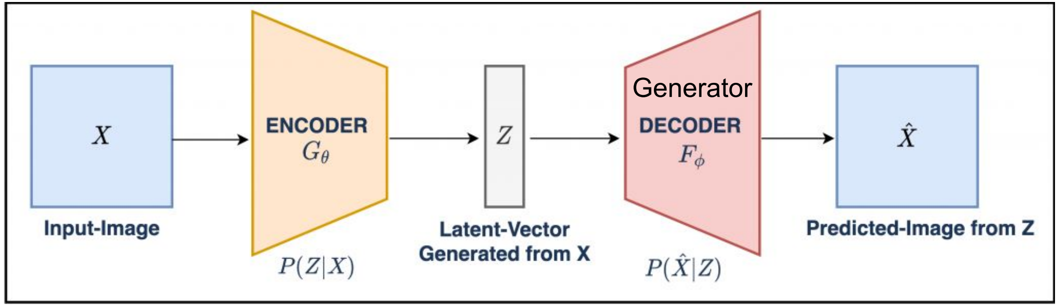


Reinforcement

Supervised

Unsupervised

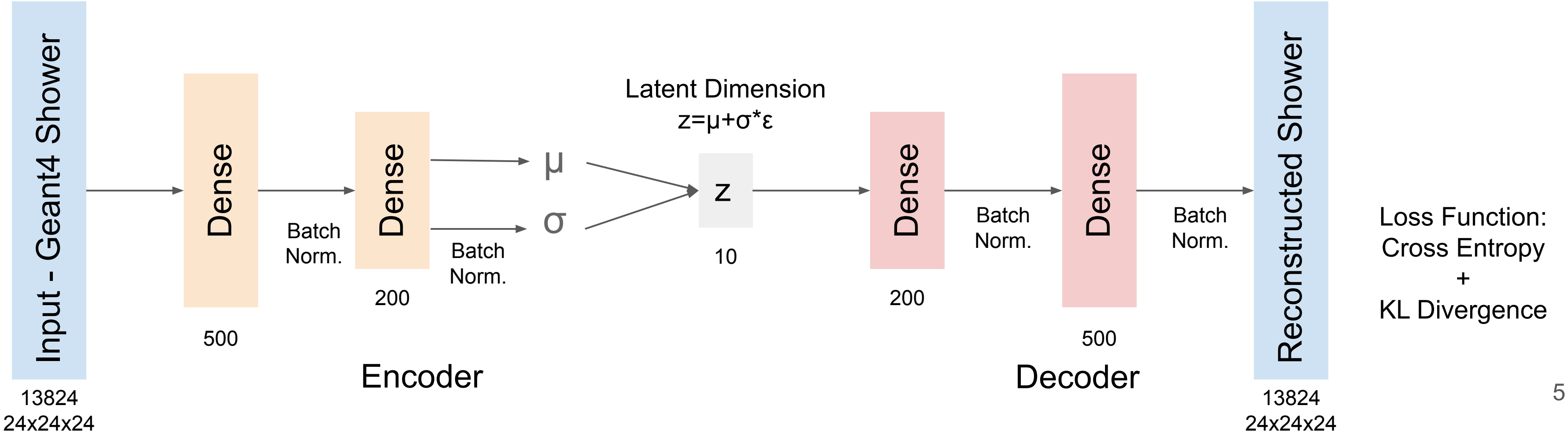
Generative models



Reference

Implementing VAE for shower simulation

- Training data (Geant4):
  - 10k events;
  - Incident particles
    - Energy: 60 GeV
    - Direction: perpendicular to the surface of the calorimeter
- Model: learns to simulate the energy deposited in the (24, 24, 24) calorimeter.

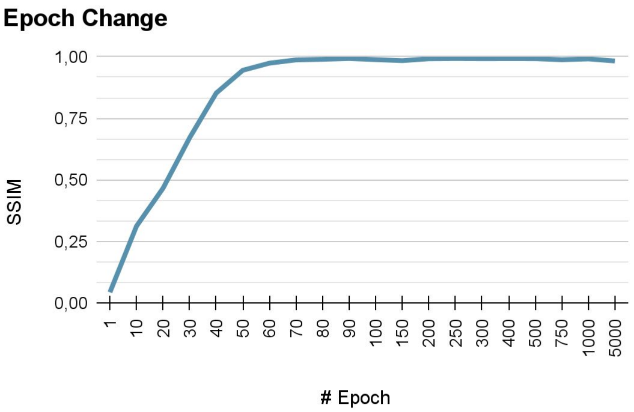


Loss Function:
Cross Entropy
+
KL Divergence

Tuning the hyperparameters

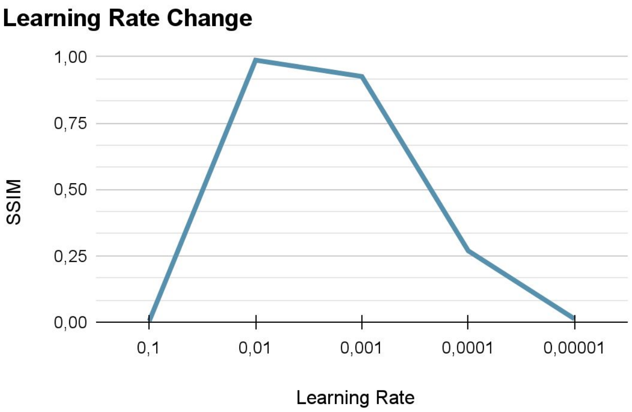
- Hyperparameters: parameters of the model that are used to control the learning process;
- We can try to tune it by changing one value at a time and seeing the impact in the model by hand;
- Metric: SSIM

Fixed Hyperparameters:
latent_dim:10;
ki:RandomNormal;
bi:Zeros;
batch_size:100;
dense_layers:2;
int_dim1:500;
int_dim2:200;
epochs:varies;
activ:relu;
outActiv:sigmoid;
val_split:0.1;
wkl:0.5;
opt:Adam;
lr:0.001;



Epoch Change

Fixed Hyperparameters:
latent_dim:10;
ki:RandomNormal;
bi:Zeros;
batch_size:100;
dense_layers:2;
int_dim1:500;
int_dim2:200;
epochs:50;
activ:relu;
outActiv:sigmoid;
val_split:0.1;
wkl:0.5;
opt:Adam;
lr:varies;

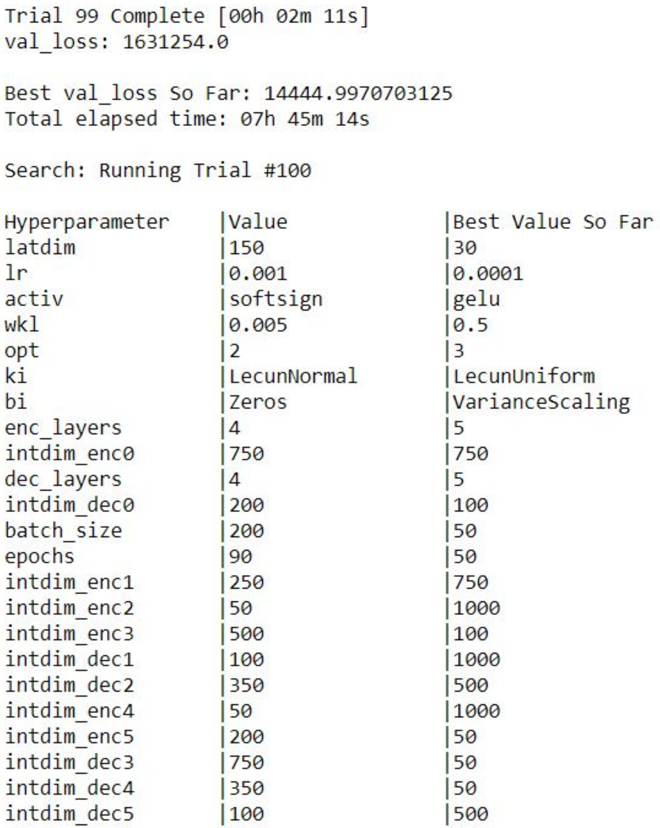


Learning Rate Change

AutoML

- Automatically search the best hyperparameters according to a certain metric;
- Has the advantage of changing more than one at the same time;
- Multiple ways of tuning: Random Search, Bayesian Optimization, Hyperband Algorithm;
- AutoKeras and Keras Tuner;

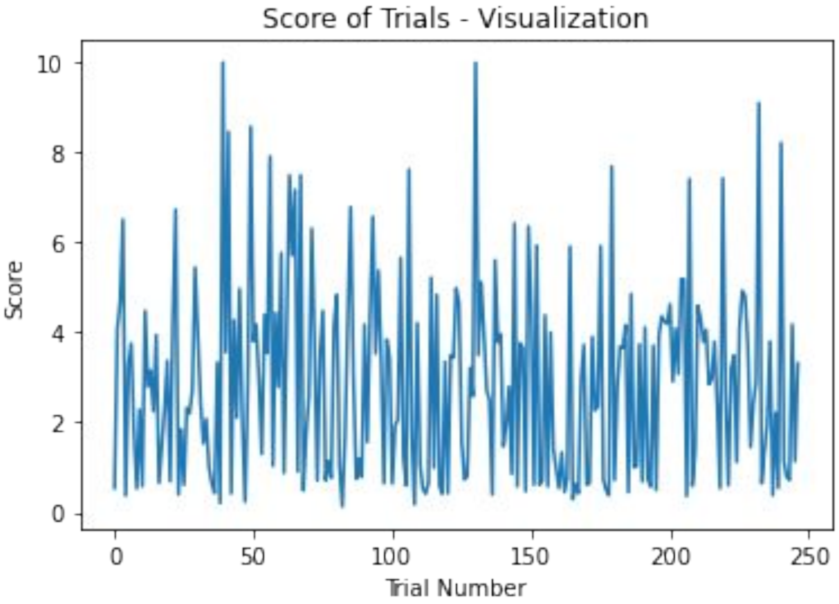
AutoKeras



```
Trial 99 Complete [00h 02m 11s]
val_loss: 1631254.0

Best val_loss So Far: 14444.9970703125
Total elapsed time: 07h 45m 14s

Search: Running Trial #100

Hyperparameter    |Value             |Best Value So Far
latdim            |150               |30
lr                |0.001             |0.0001
activ             |softsign          |gelu
wkl               |0.005             |0.5
opt               |2                 |3
ki                |LecunNormal       |LecunUniform
bi                |Zeros             |VarianceScaling
enc_layers        |4                 |5
intdim_enc0       |750               |750
dec_layers        |4                 |5
intdim_dec0       |200               |100
batch_size        |200               |50
epochs            |90                |50
intdim_enc1       |250               |750
intdim_enc2       |50                |1000
intdim_enc3       |500               |100
intdim_dec1       |100               |1000
intdim_dec2       |350               |500
intdim_enc4       |50                |1000
intdim_enc5       |200               |50
intdim_dec3       |750               |50
intdim_dec4       |350               |50
intdim_dec5       |100               |500
```

AutoML - RandomSearch

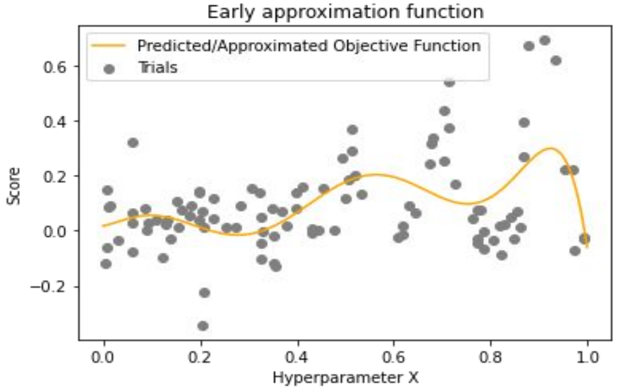
- Input
    - Model;
    - Number of trials;
    - Range of each hyperparameter;
    - Metric (for scoring the trials);
- Randomly pick a new set of hyperparameters (hp) at each trial;
- Train the model using these hp;
- Calculate a score using the input metric;
- Compare the score to previous trials;

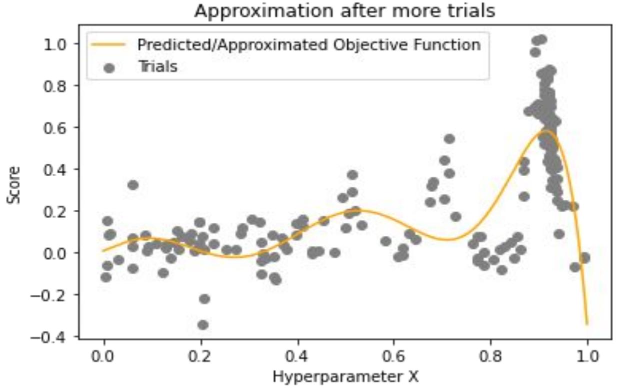


AutoML - Bayesian Optimization

It optimizes the tuning process by trying to calculate an approximation of the objective function for the tuning (score as a function of the hyperparameters):

- Pick a random set of hp, calculate the score;
- Estimate the objective function with a Gaussian process from the values of previous trials;
- Predict the score of N random sets of hp with this approximated objective function;
- Get the best set and train the model with it;
- Use this trained trial to better estimate the objective function.





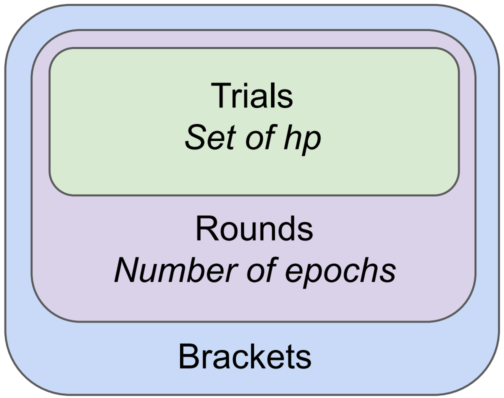
Illustrative Example with one parameter:
Reference

AutoML - Hyperband

Improves the Random Search by exploring a bigger space in less time (running on fewer epochs) and **keeping the best trials to develop** further:

- Input
    - Maximum number of epochs to train (m)
    - Factor (n) for which to increase the number of epochs;
- For each round i, train the m/n^i of sets on n^i epoch.
    - Choose the best trials to run for more epochs.
- To explore more of the space, we have brackets (black lines).

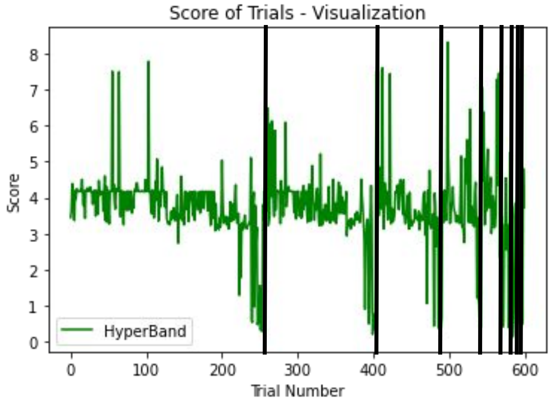
Hyperband Scheme



Trials
Set of hp

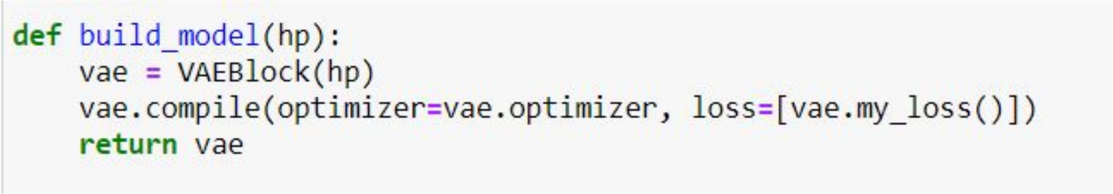
Rounds
Number of epochs

Brackets

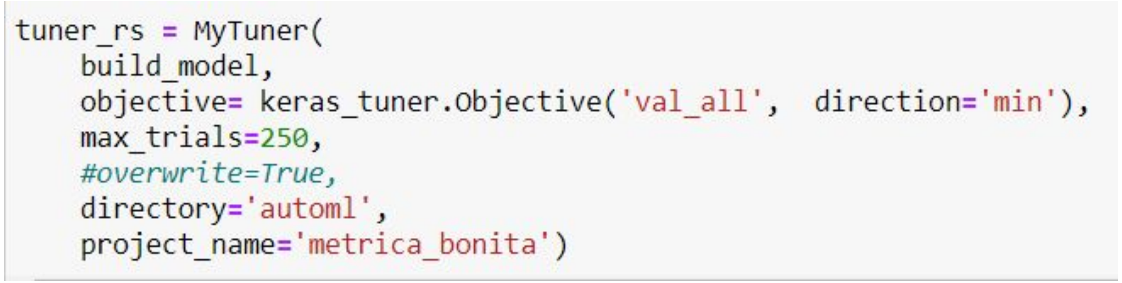


AutoML - Code Details

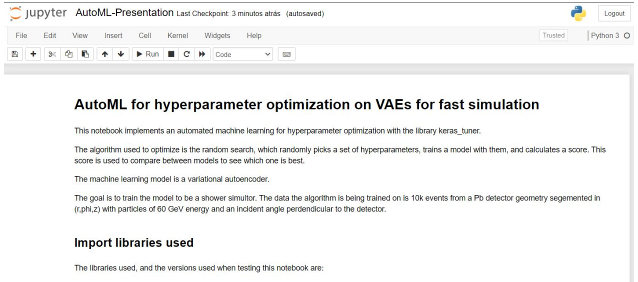
- Implementation of the VAE for each method using Keras Tuner.
- Jupyter Notebook - [Link]



```
def build_model(hp):
    vae = VAEBlock(hp)
    vae.compile(optimizer=vae.optimizer, loss=[vae.my_loss()])
    return vae
```



```
tuner_rs = MyTuner(
    build_model,
    objective= keras_tuner.Objective('val_all',  direction='min'),
    max_trials=250,
    #overwrite=True,
    directory='automl',
    project_name='metrica_bonita')
```



AutoML for hyperparameter optimization on VAEs for fast simulation

This notebook implements an automated machine learning for hyperparameter optimization with the library keras_tuner.

The algorithm used to optimize is the random search, which randomly picks a set of hyperparameters, trains a model with them, and calculates a score. This score is used to compare between models to see which one is best.

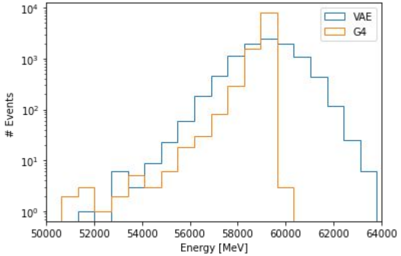
The machine learning model is a variational autoencoder.

The goal is to train the model to be a shower simultor. The data the algorithm is being trained on is 10k events from a Pb detector geometry segemented in (r,phi,z) with particles of 60 GeV energy and an incident angle perdendicular to the detector.

Import libraries used

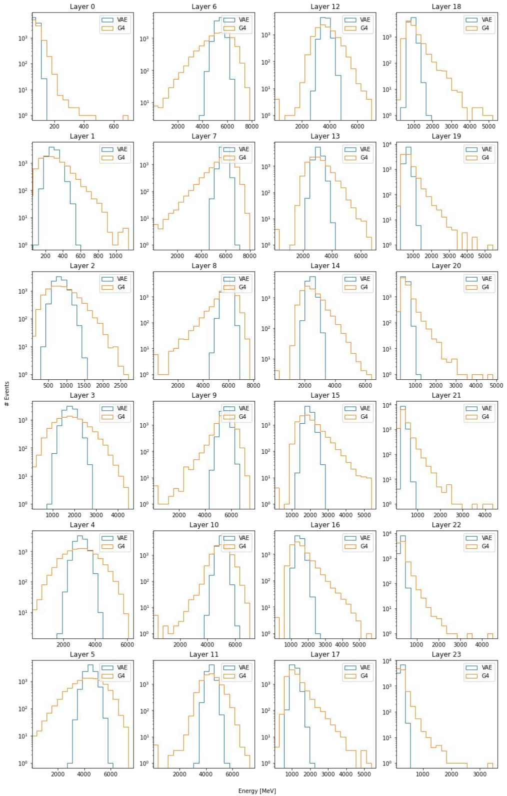
The libraries used, and the versions used when testing this notebook are:

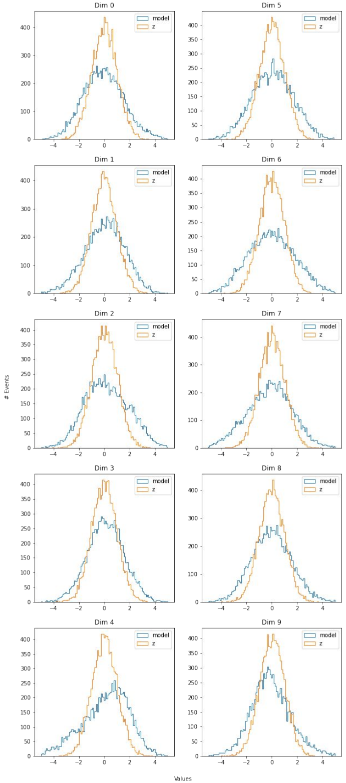
AutoML - What we want the model to be good at



Total Energy

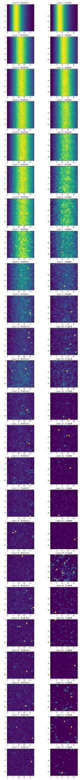
Energy per layer





How gaussian the latent representation is

Representation



AutoML - Metrics to choose the best model

- Loss?
  - Problem - different weights for the reconstruction part and the gaussianity of the latent space;
- Loss, but with same weight and order of magnitude for both parts?
  - Problem - the value for the cross entropy wasn't a good measure to look at similarities capturing the high and low energetic parts in the reconstruction;
- SSIM for the reconstruction?
  - Problem - Didn't take into account the gaussianity of the latent space;
- MSE as a metric to compute the distance between the gaussian distribution and the learned latent space distribution?
  - Problem - Didn't take into account the reconstruction part;
- Combine MSE and SSIM?
  - Worked well for the results on the evaluations!!
  - **But....**

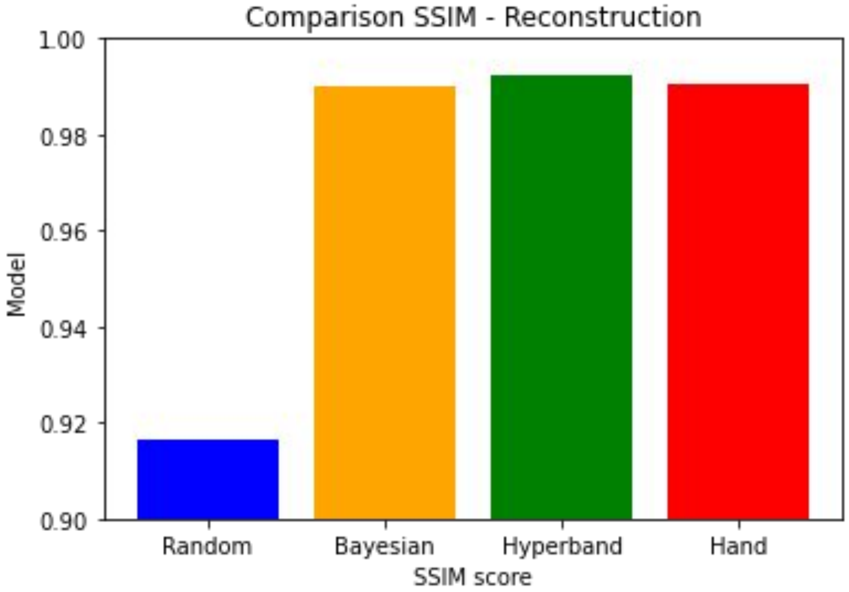
AutoML - Metrics to choose the best model

- What about the generation, that takes into account the physics properties of the simulation?
  - Problem: The best models scored from those metrics didn’t do well with the generation
- Solution: Combined metric - the Machine Learning part and the Physics part.
  - The SSIM for the reconstruction;
  - MSE as a metric to compute the distance between:
    - gaussian distribution and the learned latent space distribution
    - total energy deposited in the calorimeter, comparing the Geant4 and the generation with VAE from random values;
  - Mean of the MSE of the energy deposited in each layer, comparing the Geant4 and the generation with VAE;

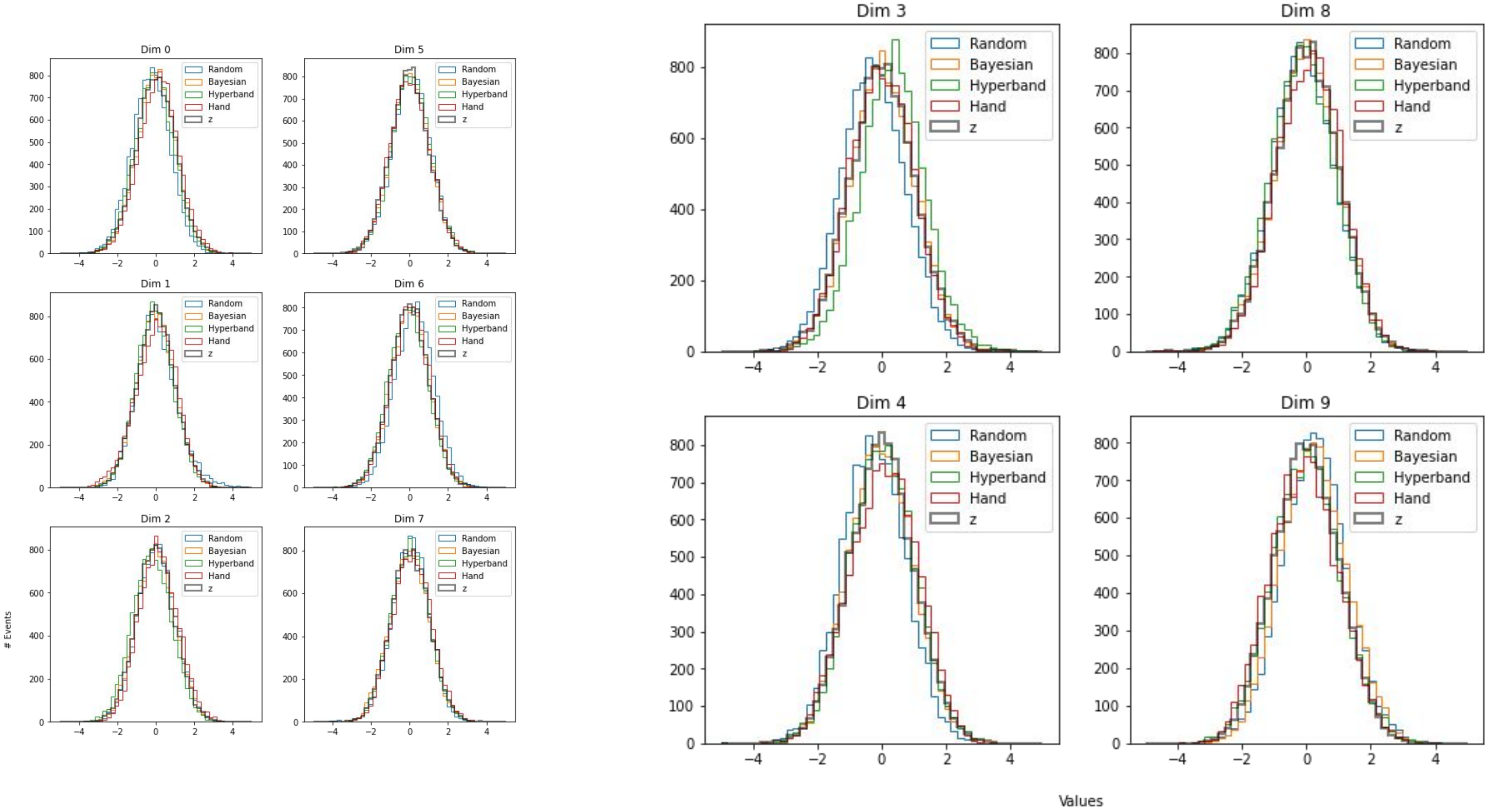
AutoML - Comparison

- Random - Best model using the Random Search.
  - 250 trials - 10h50min
  - The best model took 13 minutes to train (150 epochs);
- Bayesian - Best model using the Bayesian Optimization.
  - 250 trials - 13h14min
  - The best model took 9 minutes to train (90 epochs);
- Hyperband - Best model using the Hyperband.
  - 610 trials - 7h34min
  - The best model took 6 minutes to train (64 epochs);
- Hand - Best model considering the 4 metrics when using the hand tuning.
  - The whole hand tuning process took 3-4 days - 130 models evaluated;
  - The best model took 34 minutes to train (1000 epochs);

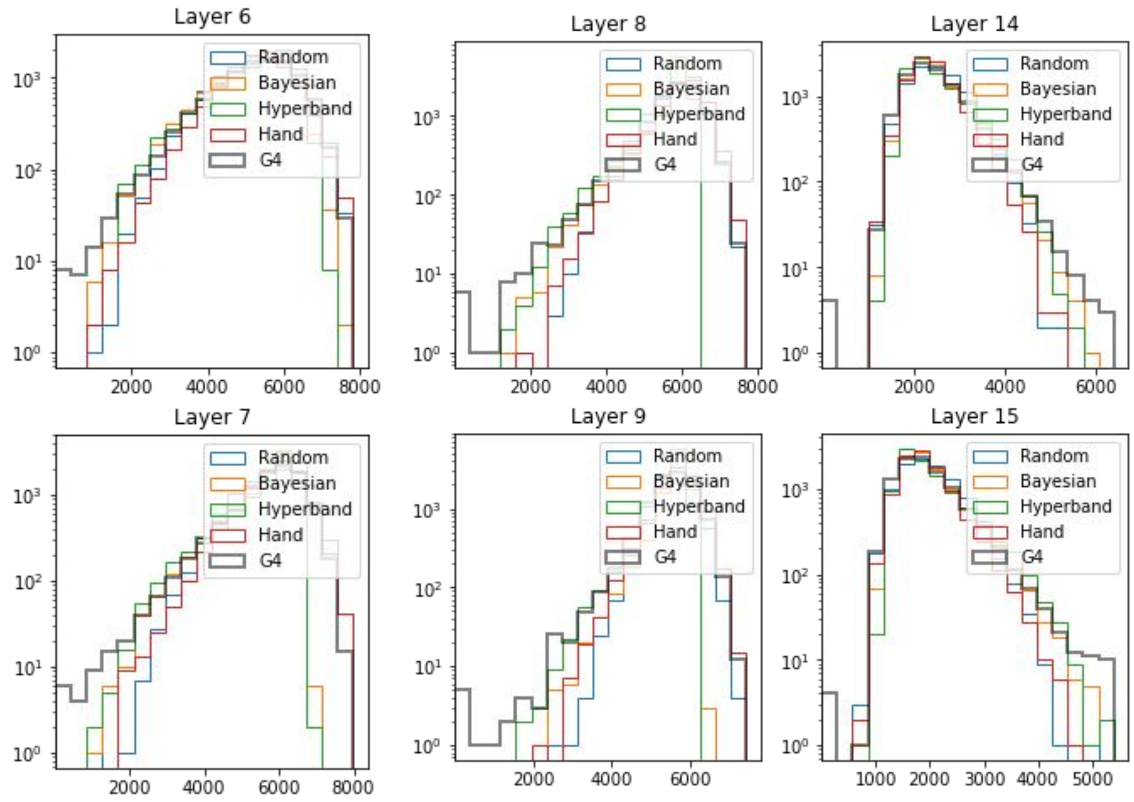
AutoML - Results - Reconstruction SSIM



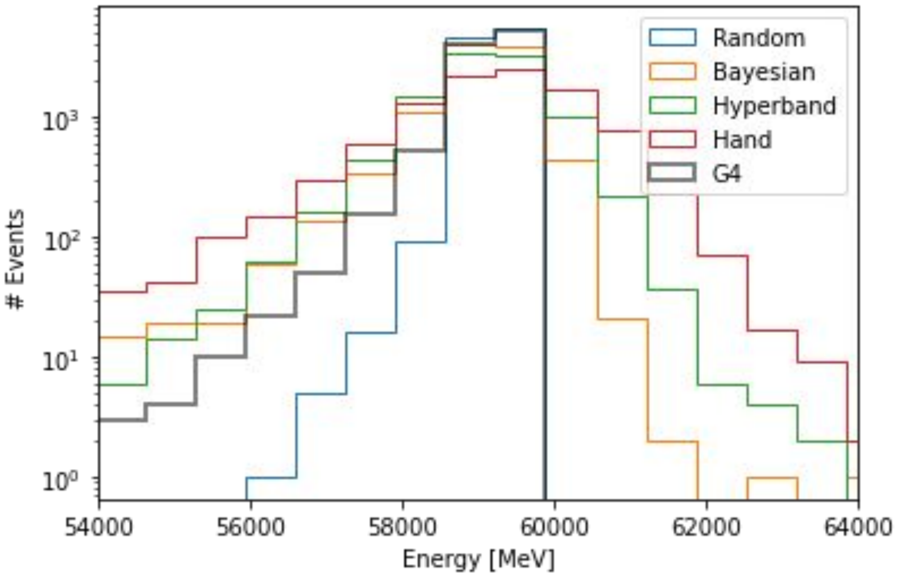
AutoML - Results - Gaussianity of the Latent Space



AutoML - Results - Energy per Layer



AutoML - Results - Total Energy



AutoML - Comparison

	Random Search	**Bayesian Optimization**	**Hyperband**
+	Simple	Tracking of the tuning process Approximate objective function for fast scoring	Fast Explore larger space Tracking of the tuning process
-	No control of the tuning process	Too long to approximate a good enough function	Discard some trials too fast

Summary and Conclusion

- Monte Carlo simulated events are a large part of **CPU consumption**;
- An alternative is to use **fast simulation** with Machine Learning;
- To improve the model, we have to **hand tune the hyperparameters**;
- **AutoML** can help to optimize those parameters automatically.
- To select the best model with the AutoML, it is important to have the right metric;
    - **Combined metric** (ML and physics) allow the tuner to choose the best considering all aspects of the problem

Summary and Conclusion

Future

- Possibility to expand this algorithm to more complex problems!
- Never tried before in shower simulation context, and can help in different areas, besides using VAEs.