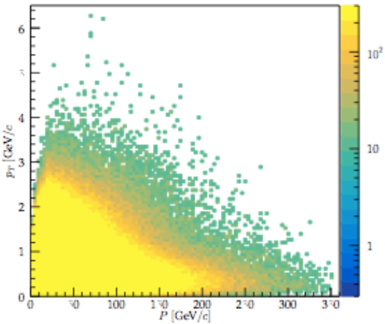
# GANs for the Muon Shield Optimisation

A. Marshall, K. Petridis

April 28, 2020

HH Wills Physics Laboratory
University of Bristol

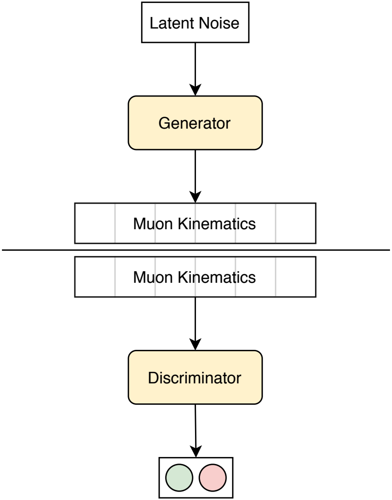
# Enhanced Muon Distribution




(Source: Oliver's thesis)

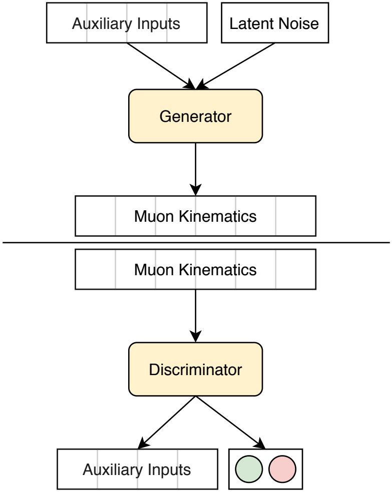
- Limited time to run muons during optimisation, need to enhance the tails of the distribution.
- Initially this was done with a small muon sample, capping the height of the distribution and applying phi rotations to events in the tails.
- **This library is able to quickly generate similar distributions based a much larger muon background sample.**

# Vanilla GAN




- Traditional GAN:
  - Generator has single input: latent noise, and outputs a generated muon kinematic vector.
  - Discriminator takes a muon kinematic vector as an input, and has a single output: a prediction at whether the sample was from the generator or the training sample.

# Auxiliary GAN




- Auxiliary GAN:
  - Generator now has **two** inputs: noise and an additional auxiliary vector. The generator then outputs as before a generated muon kinematic vector.
  - Discriminator still takes a single kinematic vector as an input, however now has **two** outputs: a prediction at real/fake along with predictions for the values of the auxiliary inputs of the kinematic vector.

# Properties of an Auxiliary GAN

- Before training the GAN the auxiliary values are calculated for the each vector in the training sample:
  - 4 auxiliary values are defined based on local density in the directions [x/y, z, P_t, P_z]
  - The auxiliary values are mapped to easy to sample distributions, in this case a single tailed normal.
  - In this case a low auxiliary value encodes means the muon sample is from the core of the distribution.

Some advantages over the Vanilla GAN:

- Training with the extra auxiliary information can add stability to the training progress.
- After training we can tune the auxiliary distribution to mould the generated output distribution as required.

**The next slides will demonstrate how to use these GANs...**

# Loading the library

To make full use of this library an installation of both Keras and uproot are required.

To begin load the library with the following:

```
import numpy as np
''' Load the muon GAN module from the SHiP_GAN_module
    package. '''
from SHiP_GAN_module import muGAN

''' Initialise the muGAN class.'''
muGAN = muGAN()
```

# Generating and Plotting

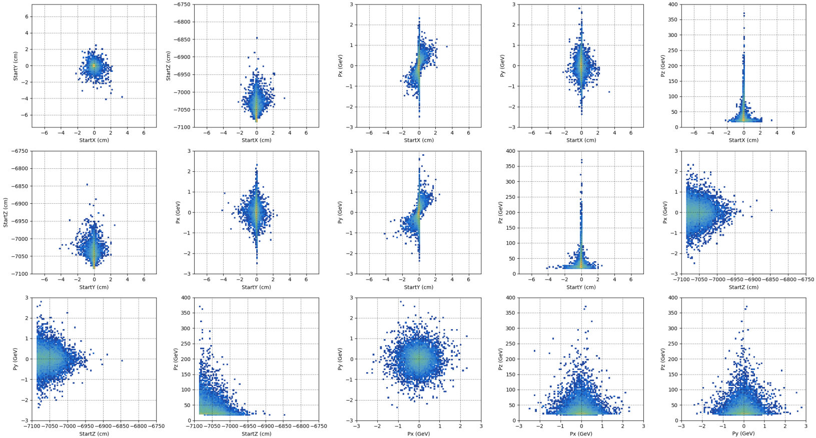
To generate from the GAN using normally distributed auxiliary values, the following `generate()` function may be used. This will produce the GANs effort at generating physical distribution of muons.

```
''' Generate muon vectors with normally distributed
    auxiliary distributions... '''
muon_kinematic_vectors = muGAN.generate(size=10000,
    tuned_aux=True)
''' The columns are as follows: Pdg, StartX, StartY, StartZ,
     Px, Py, Pz. '''
```

Then, to plot the output:

```
''' Plot kinematics of this generated vector'''
muGAN.plot_kinematics(data=muon_kinematic_vectors)
''' Plot momentum vs transverse momentum'''
muGAN.plot_p_pt(data=muon_kinematic_vectors)
```

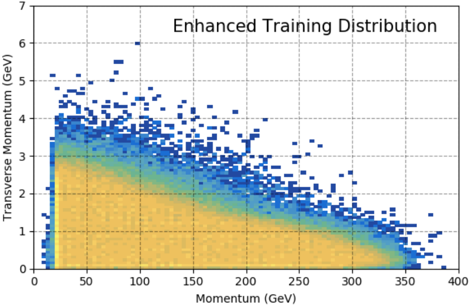
# GAN output



# Generating Enhanced Distributions

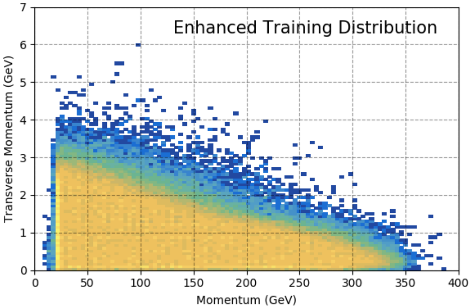
As mentioned, for the Muon Shield Optimisation procedure we will need to generate enhanced distributions. Here are the steps taken in order to do this:

- Firstly, from the GAN training data we select out an enhanced distribution.
    - The script `Create_seed_distribution.py` does this from an selection of GAN training files in my `EOS` directory.



# Generating Enhanced Distributions

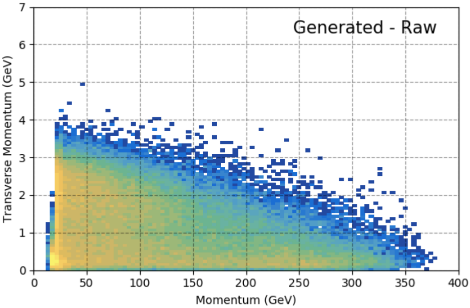
- From this enhanced distribution separate out the **auxiliary values**.
- This creates a **seed auxiliary distribution** which will be used in the generation step.
  - I have created an example stored in `SHiP_GAN_module/data_files/` which is automatically used if the user does not create their own.



# Generating Enhanced Distributions

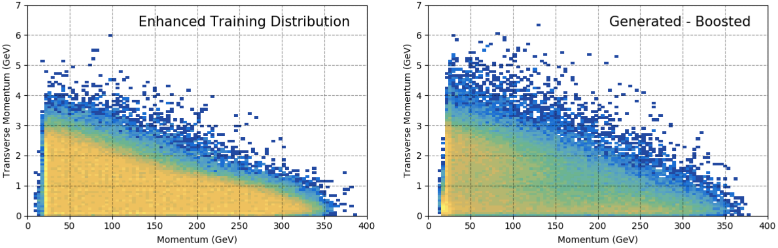
- From this seed of auxiliary values we can generate values with the `generate_enhanced()` function.

```
boosted_muon_kinematic_vectors = muGAN.generate_enhanced(
    size=int(1E5))
```



# Generating Enhanced Distributions

- This enhanced distribution can be boosted further by slightly manipulating the seed auxiliary distribution.
  - This is required as, even in using a seed distribution, we haven't addressed the issue of the GAN underestimating the tails.
- The code to generate this kind of distribution is available in the script `Enhanced_example.py`.



# Creating ROOT files

- The final step for integration with FairShip is the production of ROOT files.
- This is done with the `save_to_ROOT()` function which uses uproot.

```
muGAN.save_to_ROOT(data=muon_kinematic_vectors,filename='
    example.root')
```

The generated ROOT file is compatible with the master `MuonBackGenerator.cxx`. However I struggled to get `run_simScript.py` to quit correctly on reaching the end of the file. So there is a buffer event created at the end of the files and `run_simScript.py` must be run with the option `- n` $N$. Where $N$ is the number of generated muons.