CERN openlab

Deep generative models meet GEANT4

Alexandru-Mihai Hau

Sofia Vallecorsa

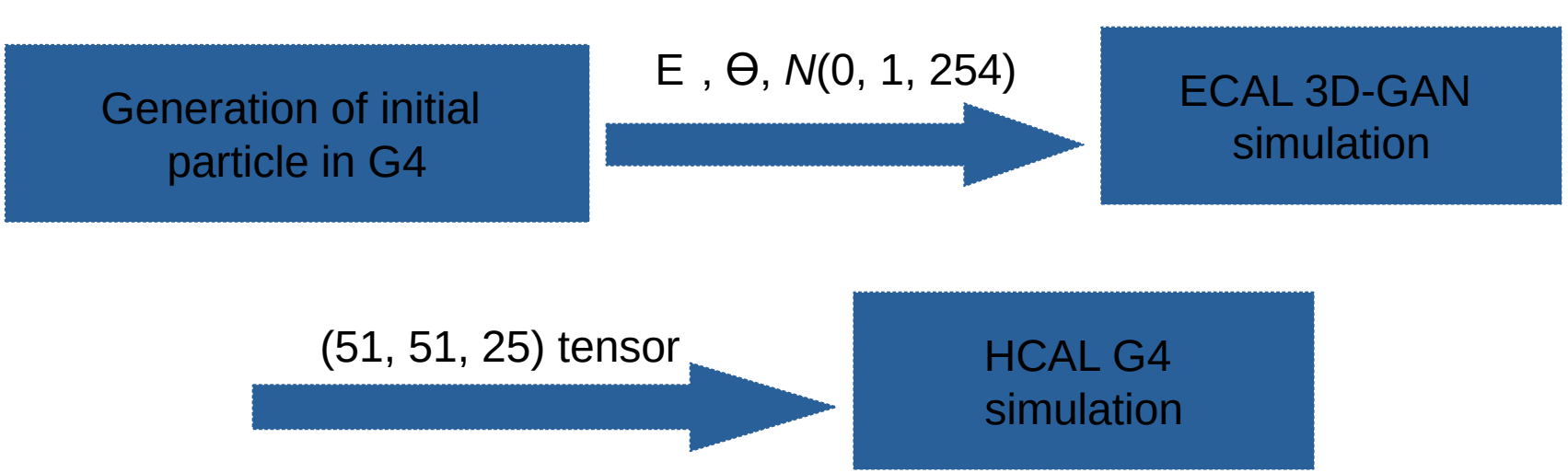
Kristina Jaruskova

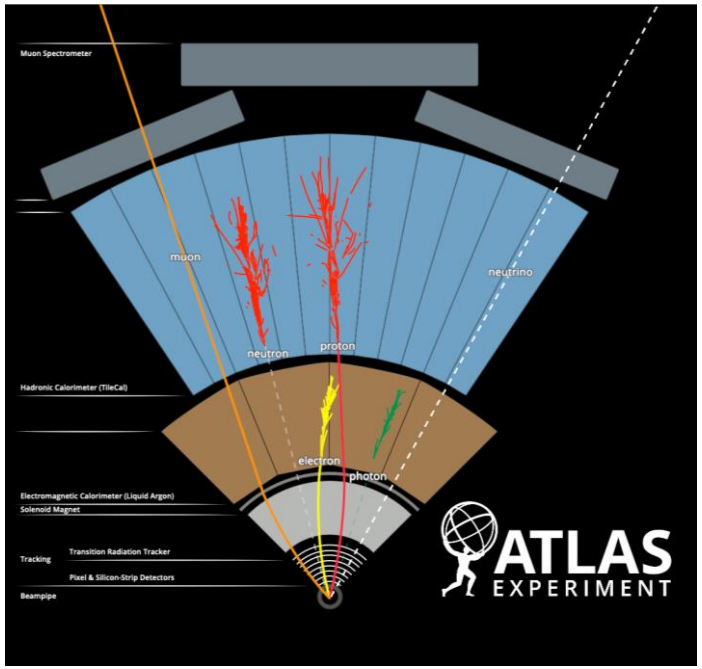
Motivation

- Particle collisions and interactions are currently best simulated on computer using Monte Carlo simulations on GEANT4
- However, the MC processes are computationally expensive – interactions of particles with the detector material in the electromagnetic calorimeter (ECAL) are costly in terms of time
- An alternative is required – development of deep learning methods

Solution

- Integrate a pre-trained 3D-Generative Adversarial Network (3D-GAN) into GEANT4 simulation chain



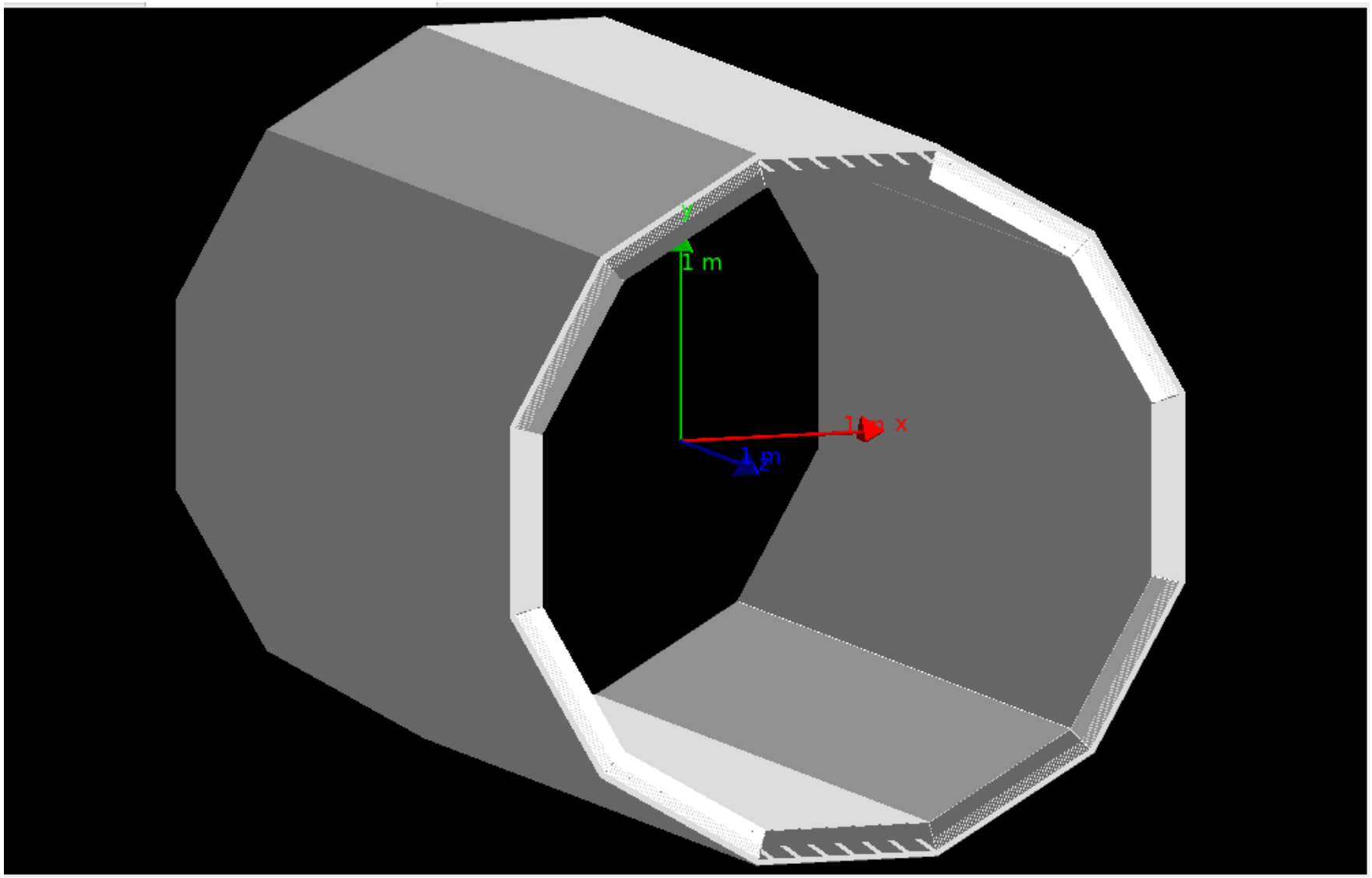


Melhase S., ATLAS detector slice (and particle visualisation), ATLAS Collaboration, 2021. https://cds.cern.ch/record/2770815

Steps

- Load a detector prototype into GEANT4 (the model is loaded from DD4HEP software – Detector Description for High Energy Physics)
- Process Geant4 output to 3DGAN input
- Integrate the 3D-GAN output in the detector frame

1) Detector load



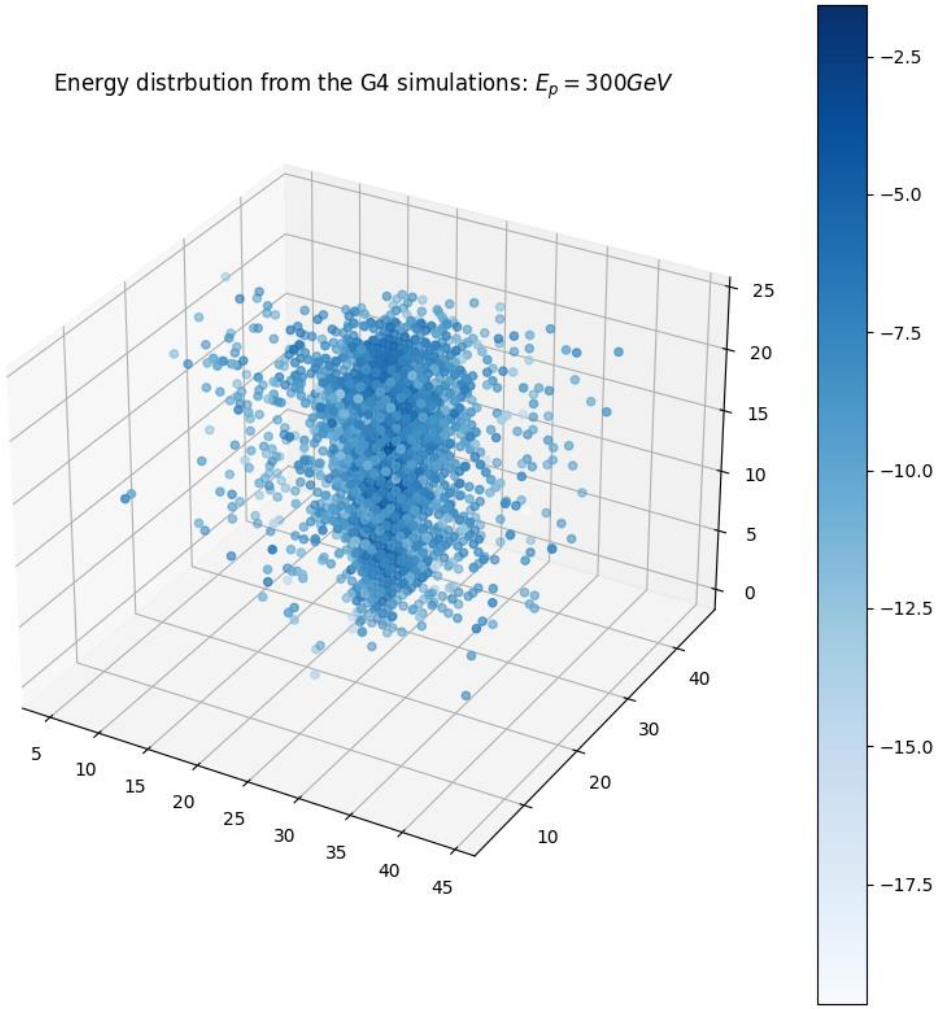
2) Integration of the 3D-GAN

```
std::vector<Ort::Value> ort_outputs =
  fSession->Run(Ort::RunOptions{ nullptr }, fInames.data(), ort_inputs.data(), ort_inputs.size(),
                output_node_names.data(), output_node_names.size());
// get pointer to output tensor float values
G4cout << "Tensor output: " << ort_outputs[0] << G4endl;
G4cout << "Tensor output: " << ort_outputs[90] << G4endl;

std::vector<int64_t> output_dims = ort_outputs[0].GetTensorTypeAndShapeInfo().GetShape();
for(int i = 0; i < output_dims.size(); i++)
{
  G4cout << "Output tensor dimension: " << output_dims[i] << G4endl;
}

G4cout << "Output test: " << ort_outputs.front().At<float>({0,30,30,23,0}) << G4endl;
assert(ort_outputs.front().IsTensor());

float* floatarr = ort_outputs.front().GetTensorMutableData<float>();
aEnergies.assign(aSize, 0);
G4int count = 0;
for(G4int x = 0; x < output_dims[1]; x++)
{
  for(G4int y = 0; y < output_dims[2]; y++)
  {
    for(G4int z = 0; z < output_dims[3]; z++)
    {
      // G4cout << "Energy from tensor output: " << ort_outputs.front().At<float>({0,x,y,z,0}) << G4endl;
      aEnergies[count] = ort_outputs.front().At<float>({0,x,y,z,0});
      // G4cout << "Difference: " << floatarr[count] - ort_outputs.front().At<float>({0,x,y,z,0});
      count++;
```



3) Data post-processing of the coordinates

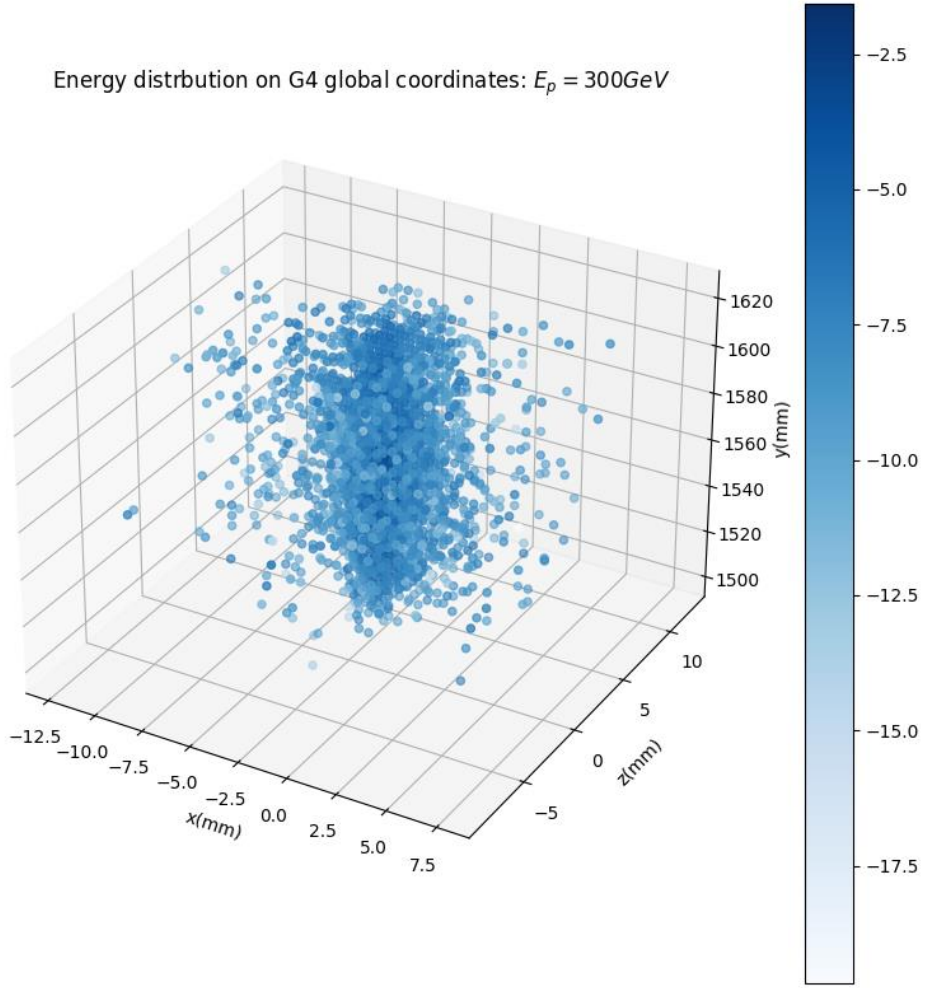
```
for(G4int x = 0; x < 51; x++){
  for(G4int y = 0; y < 51; y++){
    for(G4int z = 0; z < 25; z++){

      // Update the value for the corresponding coordinates of each vector
      // in the aPositions list
      G4float x_global = x_0 + (x - local_bary_x) * calibration;
      G4float z_global = z_0 + (y - local_bary_y) * calibration;
      G4float y_global;

      // Now take two different cases for the y-coordinate, due to the two
      // different values of the first 17 layers and the last 8 layers
      if(z < 17){
        y_global = pos0.getY() + z * thickness_17;
      }

      // For the latter case, add the thicknesses from the first 17 layers and
      // then put in the remaining part from the last 8 layers
      else{
        y_global = pos0.getY() + 17 * thickness_17 + (z - 17) * thickness_8;
      }

      // Update the counter loop and the aPositions list of vectors
      aPositions[count] = G4ThreeVector(x_global, y_global, z_global);
      count++;
    }
  }
}
```



Next steps

- Integrate the Tracker and the HCAL in the detector construction
- Work out the effect of the post-processing data on the computation time

QUESTIONS?

Alexandru-Mihai Hau

alexandru-mihai.hau@cern.ch

https://www.linkedin.com/in/alexandru-hau-71158119b/