Pre-Learning Geometry on GPUs

Accelerating Geant4 Simulations

Vangelis Kourlitis, Walter Hopkins, Doug Benjamin

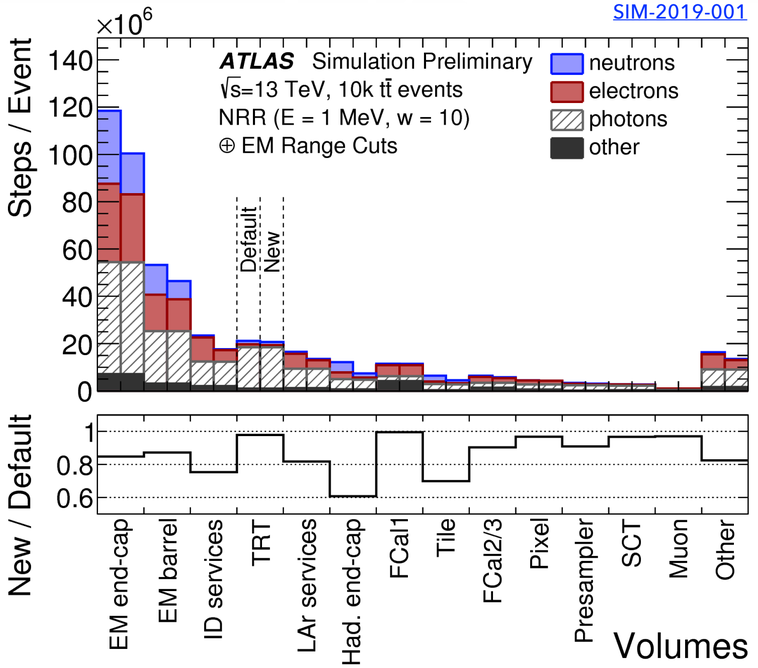
June 10th 2020

Present Situation

Facts

1. ATLAS EM calorimeters dominate the simulation load (steps).
2. Methods exploring the geometry* are taking significant amount of the simulation time

   * Locate position into geometry tree and calculate distance to next boundary in order to limit step.



Target

Accelerate the execution of expensive algorithms

Photons on EM End-cap

Callees	CPU Time: Total
G4SteppingManager::Stepping	100.0%
G4SteppingManager::DefinePhysicalStepLength	66.7%
G4VProcess::AlongStepGPIL	58.2%
G4Transportation::AlongStepGetPhysicalInteractionLength	51.7%
G4Navigator::ComputeStep	34.1%
G4NormalNavigation::ComputeStep	20.9%
G4VoxelNavigation::ComputeStep	10.8%

The Idea

Could we speed-up the geometry exploration by using a pre-defined/ learned map instead of algorithmic calculations *in each step?*

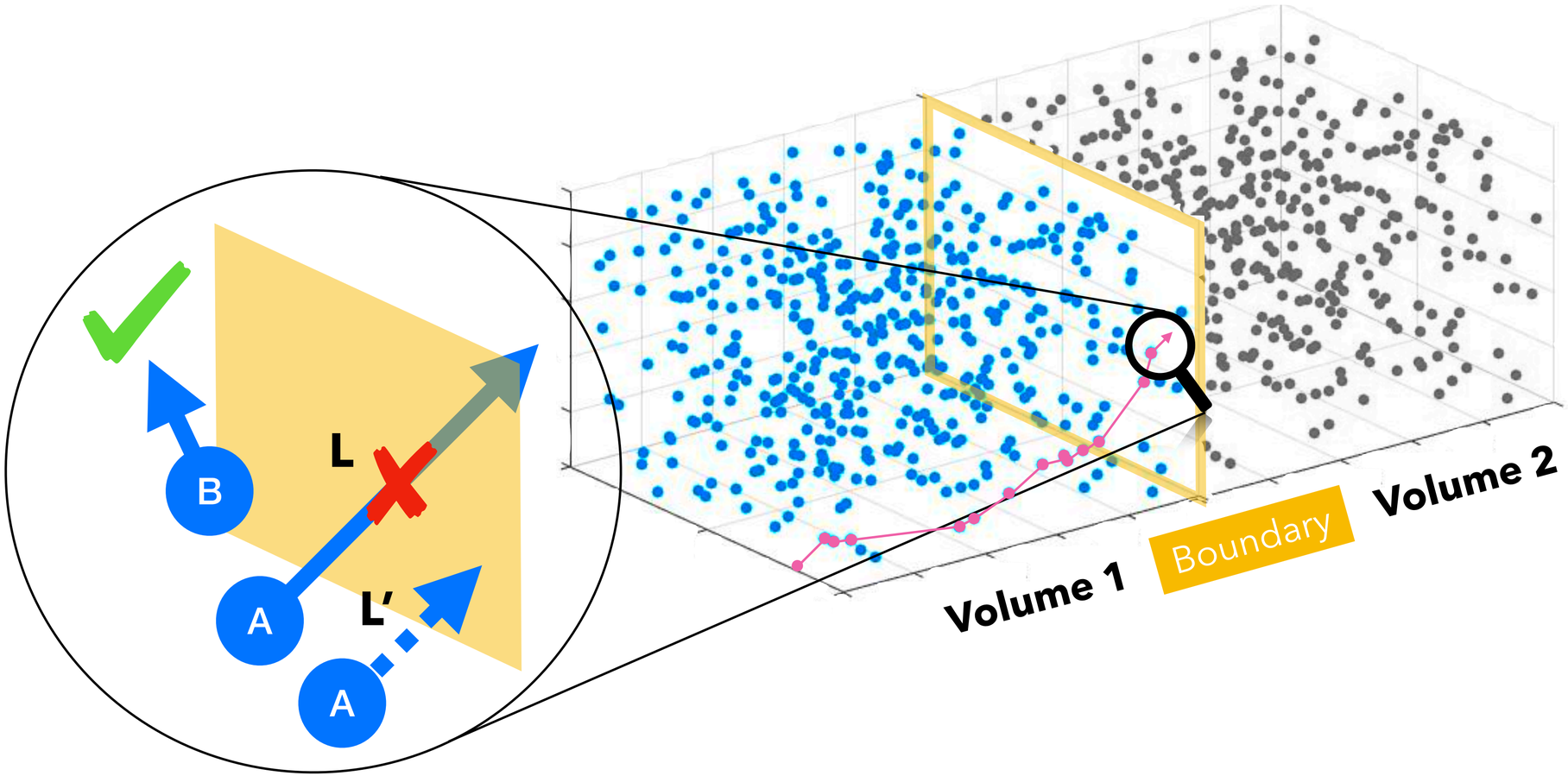
Machine learning regression technique trained for a particular geometry (e.g. ATLAS EMEC)

Hardware transparent industrial libraries optimized for different hardware architectures (CPU or **GPU**)

Much easier & assured future portability

The Idea

Could we speed-up the geometry exploration by using a pre-defined/ learned map instead of algorithmic calculations *in each step?*



Inputs:

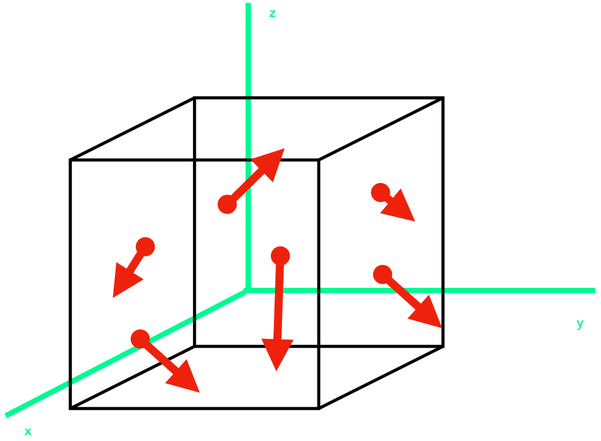
- Position (x, y, z)
- Step direction (x', y', z')

Output:

- Geometrically safe step length (L')

Feasibility Study

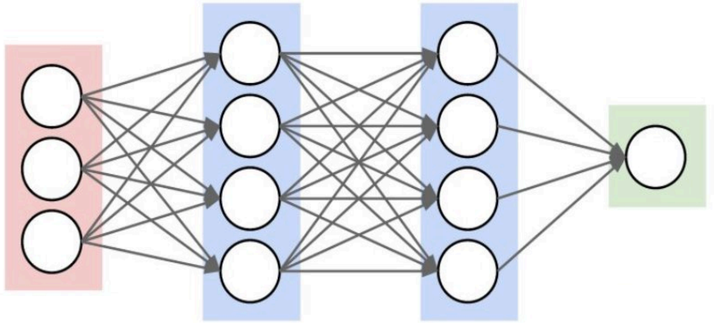
Data Collection



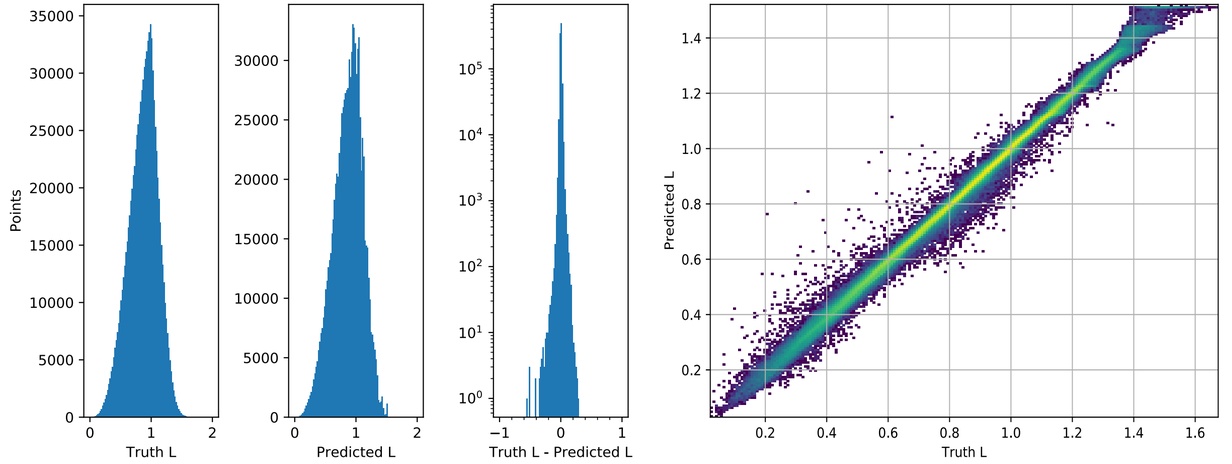
1. Sample simple geometry in random points & directions.
2. Geant4 application shooting *geantinos* and calculating the geometry-limited step length.
3. Write-out the position, direction and calculated length.

Regression DNN

Inputs: position & direction



Output: Geometric safe step length



Considerations

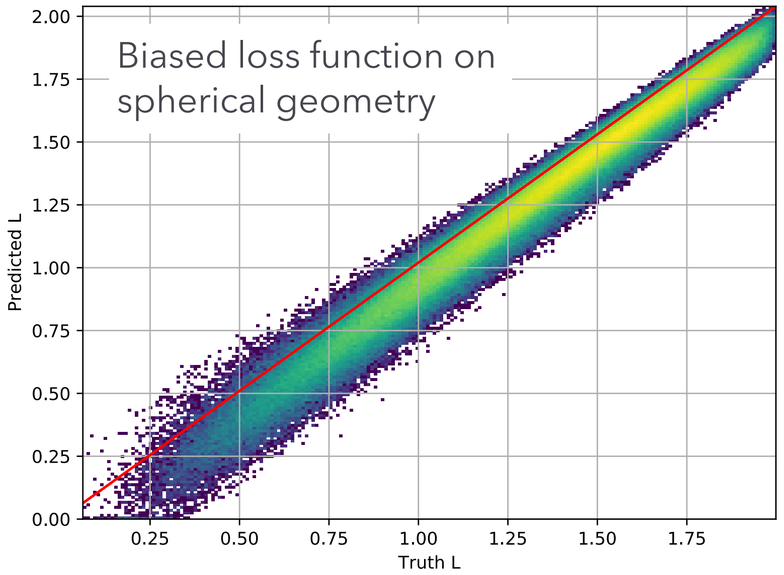
DNN geometry regression

What's the level of **accuracy** we can achieve?

Train in order **not to overestimate** the geometrically limited step.

Could we even **safely skip the distance calculation** completely?

(e.g. in the middle of geometry)



Scale to more complex example (e.g. G4Polycone)

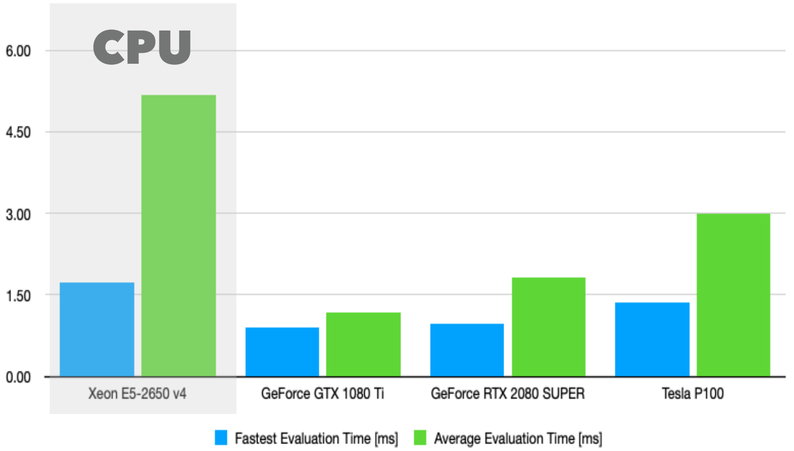
How network architecture evolves with **geometry complexity**?

Could we actually learn a geometry representation?

Evaluation time on GPU

Will the map evaluation be **faster** than Geant4 computation on CPU?

Great benefit: **Parallelism** (stepping on multiple tracks)



NB: G4Box shape calculations take ~0.3 ms