Generative Models for Fast Calorimeter Simulation: LHCb Case

Viktoria Chekalina, Egor Zakharov, Fedor Ratnikov, Elena Orlova

NRU Higher School of Economics, Yandex School of Data Analysis

July 09 2018
Simulation in LHC

- The significant part of the computing resources are used for MC simulation in High Energy Physics experiments in LHC.
- About 53% of the simulations resources are spent to simulation processes in calorimeters.
- In Run 3 a significant increase in luminosity is planned.
- We need to speed up the simulation.
Simulating the Calorimeter Responses

GEANT
- Simulation of the particle passing through the material now is provided by GEANT application.
- GEANT simulation is very detailed
- Calorimeter has less granularity, than GEANT simulation step
- We can simulate detector’s response by using simpler model

Formulation of the Simulation Problem
- Input: particle parameters (i.e. 3D momentum + 2D coordinate)
- Output: calorimeter response
Approaches

Shower Library

https://indico.cern.ch/event/740959/

- Store showers, simulated by GEANT
- For input parameters choose the the most suitable shower and, respectively, the detector’s response

Generative Model: Variational Auto Encoders (VAE)

- Model samples the energy value in cells of response from the set of distributions
- Parameters of distributions is tuned by training neural network

Generative Model: Generative Adversarial Network (GAN)

- Model consists of two parts: generator tries to create objects similar to real, discriminator tries to distinguish real object from generated
- Training ends when the discriminator stops seeing the differences between real and generated
From GAN to WGAN

Classical GAN objective function

- $P_{real}$ - the distribution over real data, $P_{gen}$ - the distribution over generated data, $x$ - real object, $\hat{x}$ - generated object, $z$ - input noise
- $\max_D \mathbb{E}_{x \sim P_{real}} [\log D(x)] + \mathbb{E}_{\hat{x} \sim P_{gen}} [\log(1 - D(\hat{x}))]
- $\min_G \mathbb{E}_{z \sim P_z} [-\log(D(G(z)))]$

We can choose the measure by which we want to match the distributions.
The Wasserstein distance can provide a meaningful and smooth representation of the divergence between two distributions

Wasserstein GAN objective function

- $\max_D \mathbb{E}_{y \sim p(y)} D(y) + \mathbb{E}_{\hat{y} \sim p(\hat{y})} D(\hat{y}) + \lambda \mathbb{E}_{\hat{y} \sim p(\hat{y})} (||\nabla \hat{y} p_{\hat{y}}|| - 1)^2$
- $\min_G \mathbb{E}_{z \sim p_z(z)} [-D(G(z))]
- Wasserstein GAN decreases Wasserstein measure between real and generated samples
Wasserstein Distance

Wasserstein distance

- $W(P_{real}, P_{gen})$ can be informally interpreted as a cheapest transportation plan to move sand from first pile (distribution) to second
- $\Pi(P_{real}, P_{gen})$ is the set of all possible joint probability distributions ("all possible way to move sand") between $P_{real}$ and $P_{gen}$
- $\gamma \in \Pi(P_{real}, P_{gen})$ - one joint distribution ("one possible transport act"),
  $\sum_{x, \hat{x}} \gamma(x, \hat{x}) = P_{real}(x)$, $\sum_{x} \gamma(x, \hat{x}) = P_{gen}(\hat{x})$
- $W(P_{real}, P_{gen}) = \inf_{\gamma \in \Pi(P_{real}, P_{gen})} \mathbb{E}_{(x, \hat{x}) \sim \gamma} [\|x - \hat{x}\|]$
- $\sum_{x, \hat{x}} \gamma(x, \hat{x})\|x - \hat{x}\| = \mathbb{E}_{(x, \hat{x}) \sim \gamma} \|x - \hat{x}\|$
Use stand-alone LHCb-like calorimeter GEANT4 setup to produce reference train and test samples

Consider calorimeter response as a figure of 30*30 calorimeter cells to fit any possible granularity in LHCb calorimeter

Deep Convolutional Neural Network (DCNN) as a generator and a discriminator

Generator converts 5 initial particle parameters ($3D$ momentum + $2D$ coordinate) and the Gaussian noise to response

We reconstruct 5 initial parameters on every generated images and try to minimize divergence between predicted and input particle parameters (add this term to generator loss)
Training scheme

\[ \min_G \mathbb{E}_{\hat{x} \sim P_{\text{fake}}} [-D(\hat{x})] + ||y - \hat{y}||_1 \]

**Generator**

- Input: \(5x1: px, py, pz, \ldots\)
- Output tensor size (w/o batch size): \(CxHxW\)
- Noise: \(Nx1\)
- Training scheme: FC + reshape + concat
- Conv 2x + Upsampling + BN + ReLU
- Output: \(\hat{x}\)

**Discriminator**

- \(D(x)\) + score
- \(D(\hat{x})\)
- Input: \(5x1\)
- Output tensor size: \(30x30\)
- Output: \(\hat{y}\)

**Regressor (pretrained)**

- \(\hat{y}\)
- \(\hat{x}\)

\[ \min_D \mathbb{E}_{\hat{x} \sim P_{\text{fake}}} [D(\hat{x})] - \mathbb{E}_{x \sim P_{\text{real}}} [D(x)] + \lambda \mathbb{E}_{\hat{x} \sim P_{\text{fake}}} [||\nabla_{\hat{x}} D(\hat{x})||_2 - 1]^2 \]
5D: real and generated responses

- E 63.7 GeV
  - px/pz 0.005
  - py/pz 0.154

- E 6.5 GeV
  - px/pz 0.046
  - py/pz 0.108

- E 15.6 GeV
  - px/pz -0.196
  - py/pz -0.036

- E 15.6 GeV
  - px/pz -0.019
  - py/pz 0.181
1D case

Energy fraction

- $E_{\text{init}}$ - the energy of particle
- $E_1 = \sum_{i=1}^{2} \sum_{j=1}^{2} E_{14+i,14+j}$
- $E_2 = \sum_{i=1}^{4} \sum_{j=1}^{4} E_{13+i,13+j} - E_1$
- $E_3 = \sum_{i=1}^{6} \sum_{j=1}^{6} E_{12+i,12+j} - E_2$
- $E_4 = \sum_{i=1}^{8} \sum_{j=1}^{8} E_{11+i,11+j} - E_3$
5D case

- Some chosen distributions are reproduced pretty well, some - not quite. The definition of quality metric is an issue. We can’t observe all possible distributions.
Time of generation

- 0.04 ms per sample on GPU
- 4.7 ms per sample on CPU

Conclusion

- We developed generative models to generate calorimeter responses.
- Generated responses look similar to real hits.
- Described shape’s property of response and statistical property of samples’ set distributions matches in real and generated data.