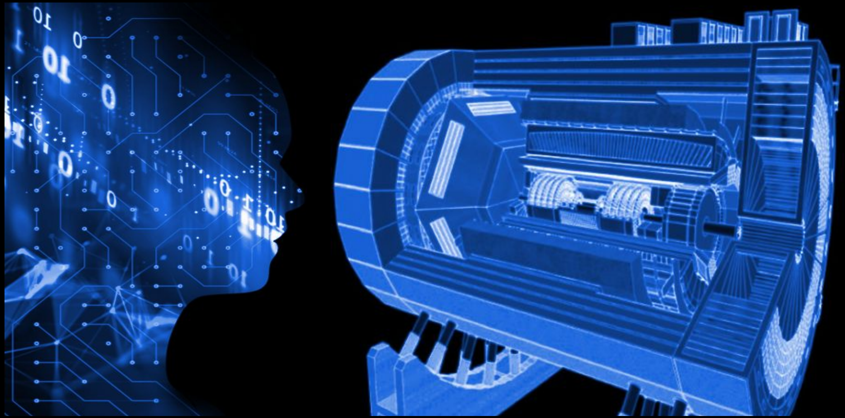


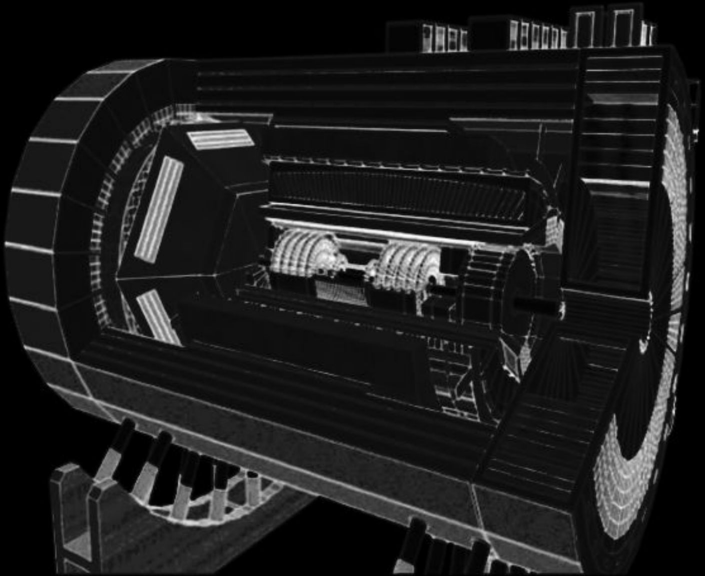
AI for Detector Design

National Nuclear Physics Summer School
MIT, 2022

Cristiano Fanelli

Lecture 3

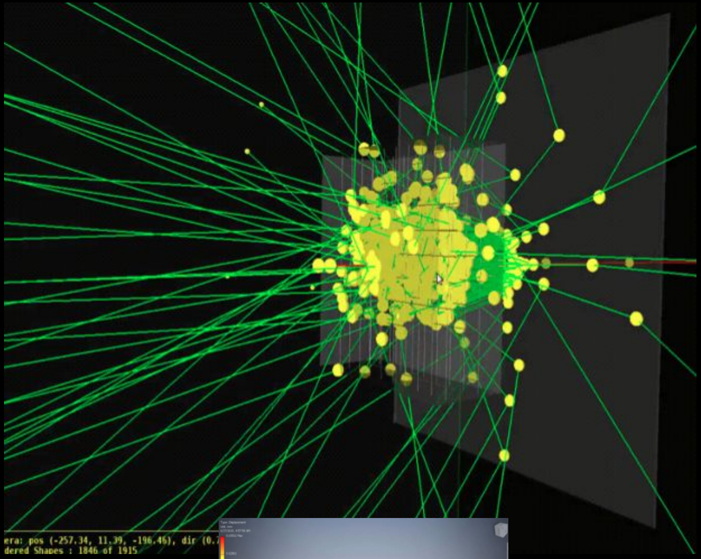
MOO for HEP/NP Applications

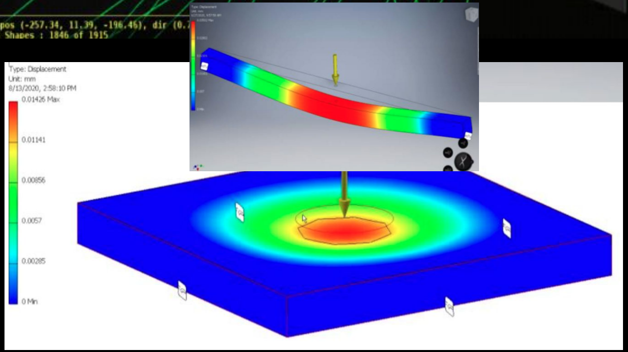


Obviously MOGA is widely utilized for many applications and in different fields

Novel aerogel material

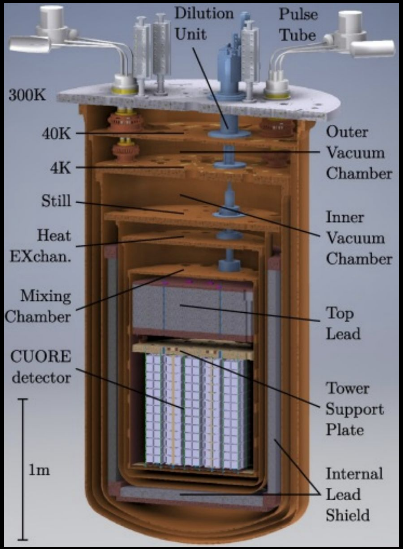
- Aerogels with low refractive indices are very fragile - tiles break during production and handling, and their installation in detectors.
- To improve the mechanical strength of aerogels, Scintilex is introducing fibers into the aerogel that increase mechanical strength, but do not affect the optical properties.
- We are designing the aerogel+fibers optimizing mechanical stability and resolution.
- Paper in preparation.


era: pos (-257.34, 11.39, -196.46), dir (0.
dered Shapes : 1846 of 1915

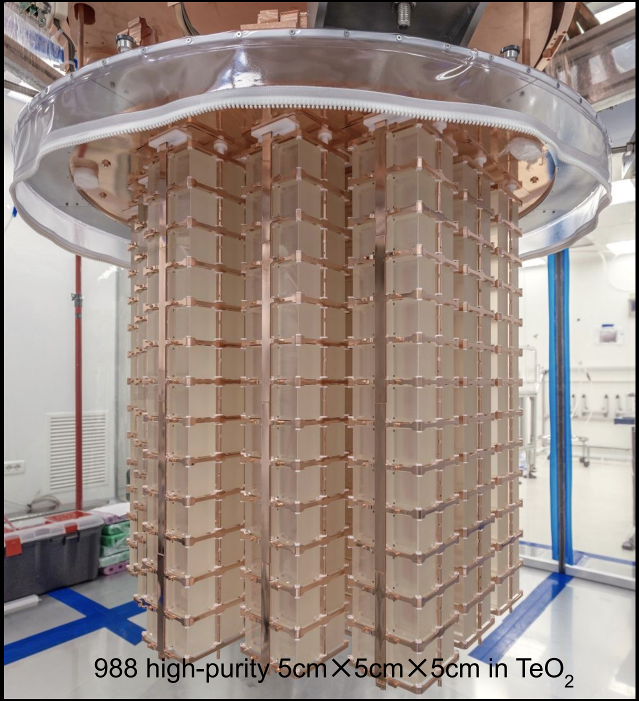


Muon Track Reconstruction at CUORE

Cryogenic Underground Observatory for Rare Events A ton-scale detector searching for 0νββ decay in ^{130}Te

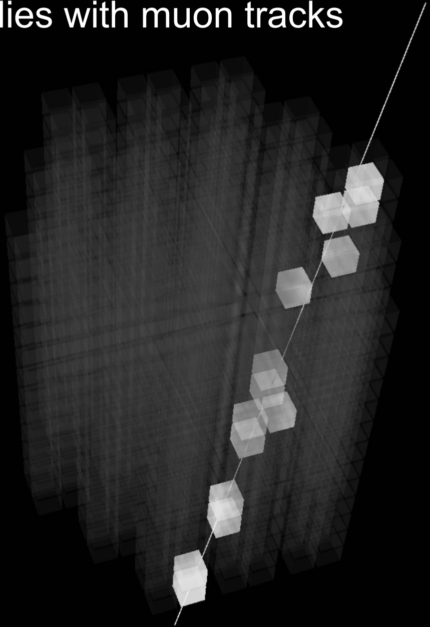


Schematic of the CUORE detector inside the cryostat held at T~12mK



Crystal absorbers suspended in copper frames in a 19-tower, 13-floor array

- Segmented bolometric detector array
  - 0νββ, direct DM, exotic track-like signatures
- Studies with muon tracks



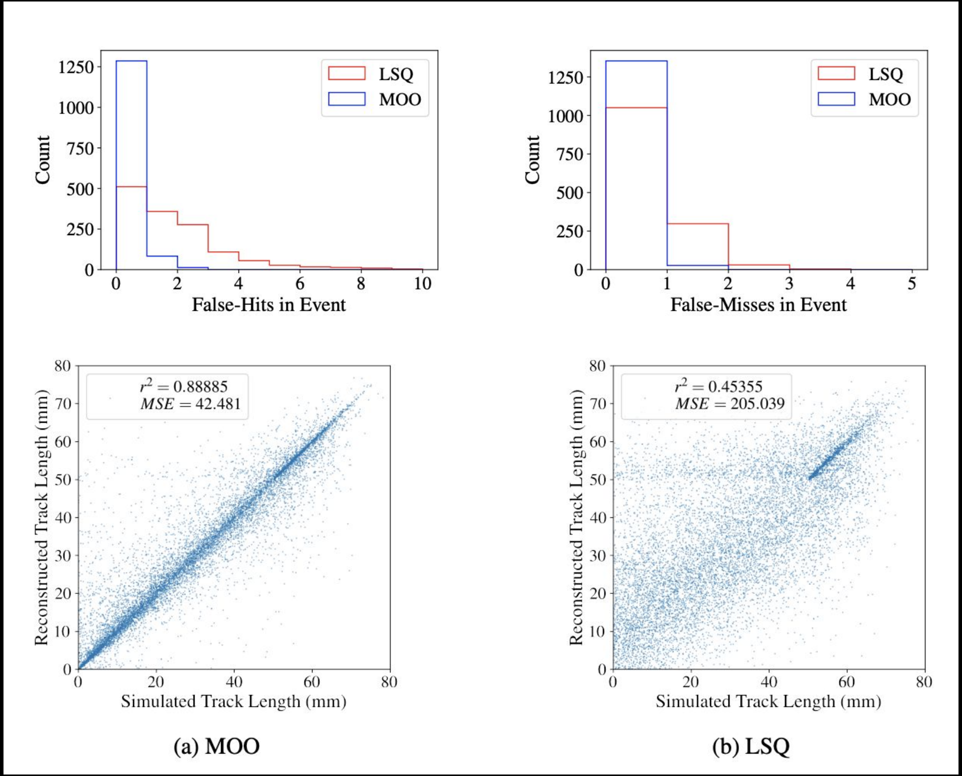
J. Yocum, D. Mayer, J. Ouellet, L. Winslow, 2202.03194 (2022)

Muon Track Reconstruction at CUORE

- Reconstruction of tracks based on MOO of three objectives; two-steps reconstruction (path-lengths and dE/dx); NSGA-II+NSGA-III
- Candidate trajectory:
  - Intersects as many lit crystals as possible
  - Minimizes intersection with unlit crystals
  - Encourages dE/dx values with high probability

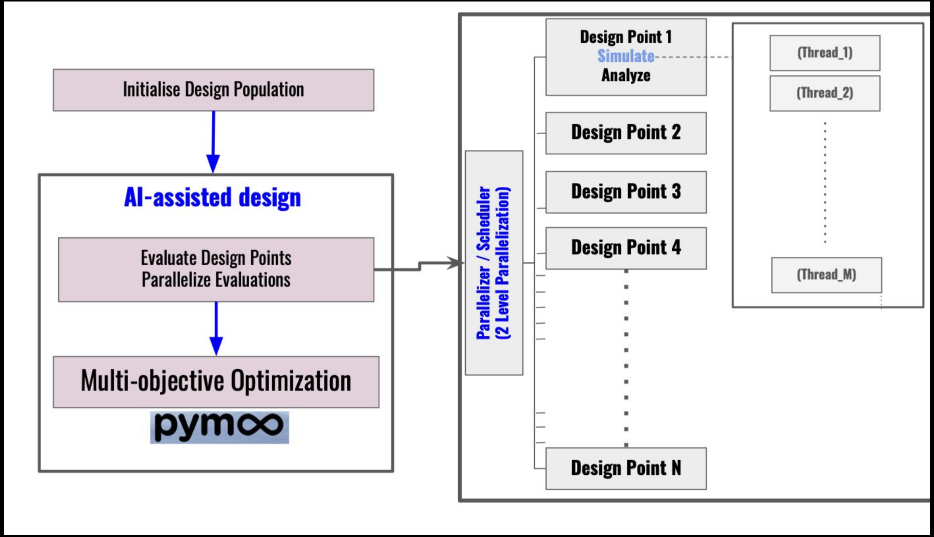
$$f_{1,2}(x) = \sum_{i \in \Omega_{1,2}} \frac{1}{1 + e^{\lambda_{1,2}(x_i - \alpha_{1,2})}}$$

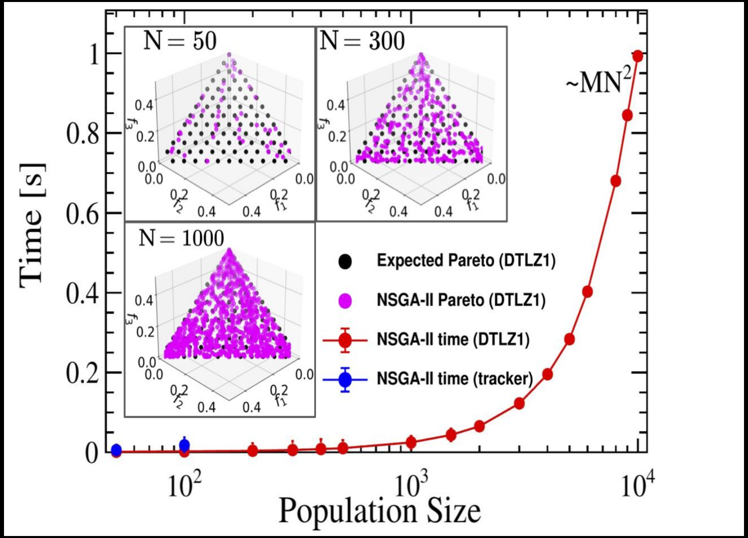
$$f_3(x) = -\sum_i \log(p(\beta_i))$$

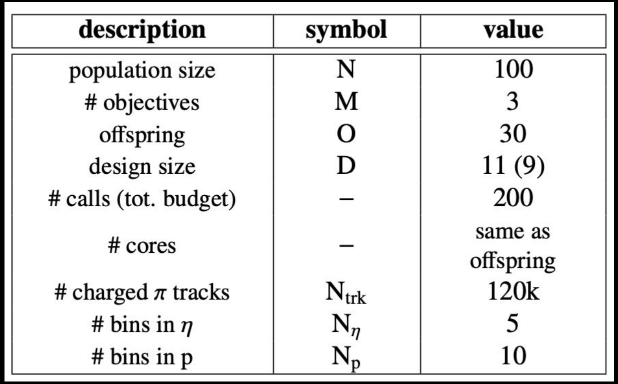


MOGA Parallelization

time taken by GA + sorting

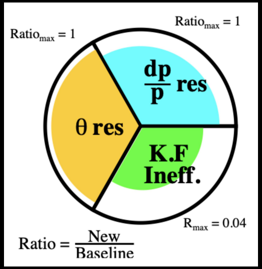






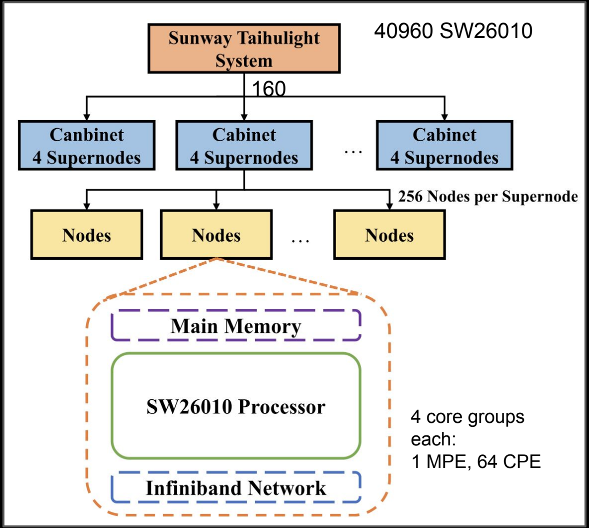
description	symbol	value
population size	N	100
# objectives	M	3
offspring	O	30
design size	D	11 (9)
# calls (tot. budget)	–	200
# cores	–	same as offspring
# charged π tracks	N_{trk}	120k
# bins in η	N_η	5
# bins in p	N_p	10

- Used a test problem DTLZ1
- Verified scaling following MN^2 and convergence to true front
- ~1s/call with 10^4 size!
- For 11 variables and 3 objectives needs ~ 10000 evaluations to converge
  ~10k CPUhours / pipeline



MOGA Parallelization on Supercomputers

- Well known that NSGA-II increase in computational complexity as **O(MN^2)**.
- A recent trend in MOEA is distributed NSGA-II and implementation on supercomputers. This is useful when large populations are needed (*e.g.*, 10^5), due to complexity and/or to approximate the Pareto front with high accuracy.
- A custom optimized parallel NSGA-II called swNSGA-II has been designed for Sunway TaihuLight [1] supercomputer.



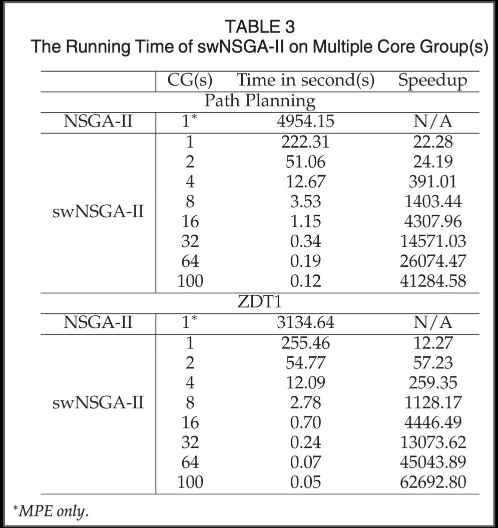


TABLE 3
The Running Time of swNSGA-II on Multiple Core Group(s)

	CG(s)	Time in second(s)	Speedup
	Path Planning		
NSGA-II	1*	4954.15	N/A
swNSGA-II	1	222.31	22.28
	2	51.06	24.19
	4	12.67	391.01
	8	3.53	1403.44
	16	1.15	4307.96
	32	0.34	14571.03
	64	0.19	26074.47
	100	0.12	41284.58
	ZDT1		
NSGA-II	1*	3134.64	N/A
swNSGA-II	1	255.46	12.27
	2	54.77	57.23
	4	12.09	259.35
	8	2.78	1128.17
	16	0.70	4446.49
	32	0.24	13073.62
	64	0.07	45043.89
	100	0.05	62692.80

*MPE only.

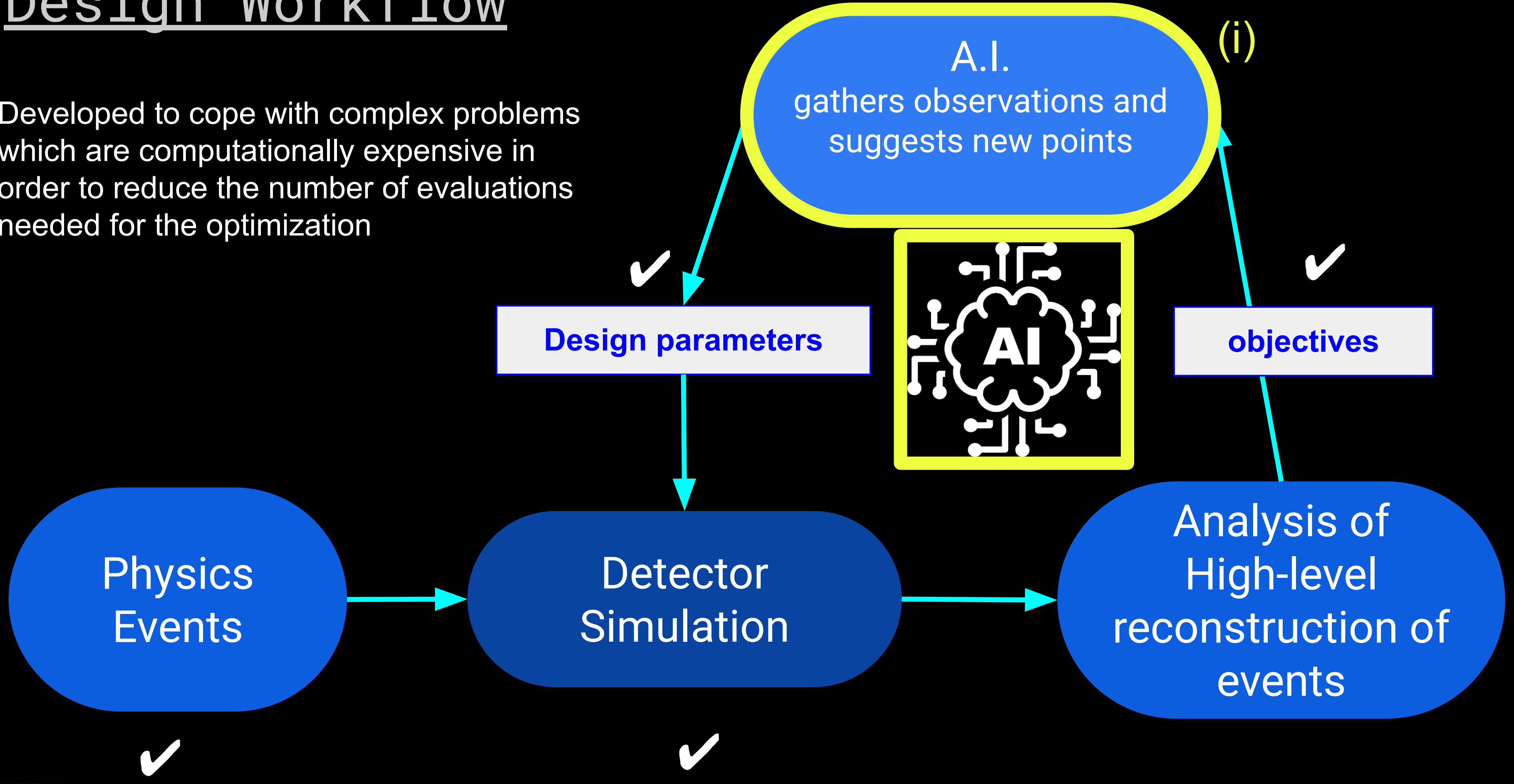
- swNSGA-II utilizes process and thread level parallelism based on an improved island master-puppet model.
- Performance have been benchmarked against conventional NSGA-II with a speedup of ~$5 \cdot 10^4$ for standard optimization problems.
- Comparisons with GPU (GeForce GT 630) -based NSGA-II done using 1 core group only (64 CPE), obtaining a speedup of ~10 with large populations.

[1] Liu, Xin, et al. IEEE Trans Parallel Distrib Syst 32.4 (2020): 975-987.

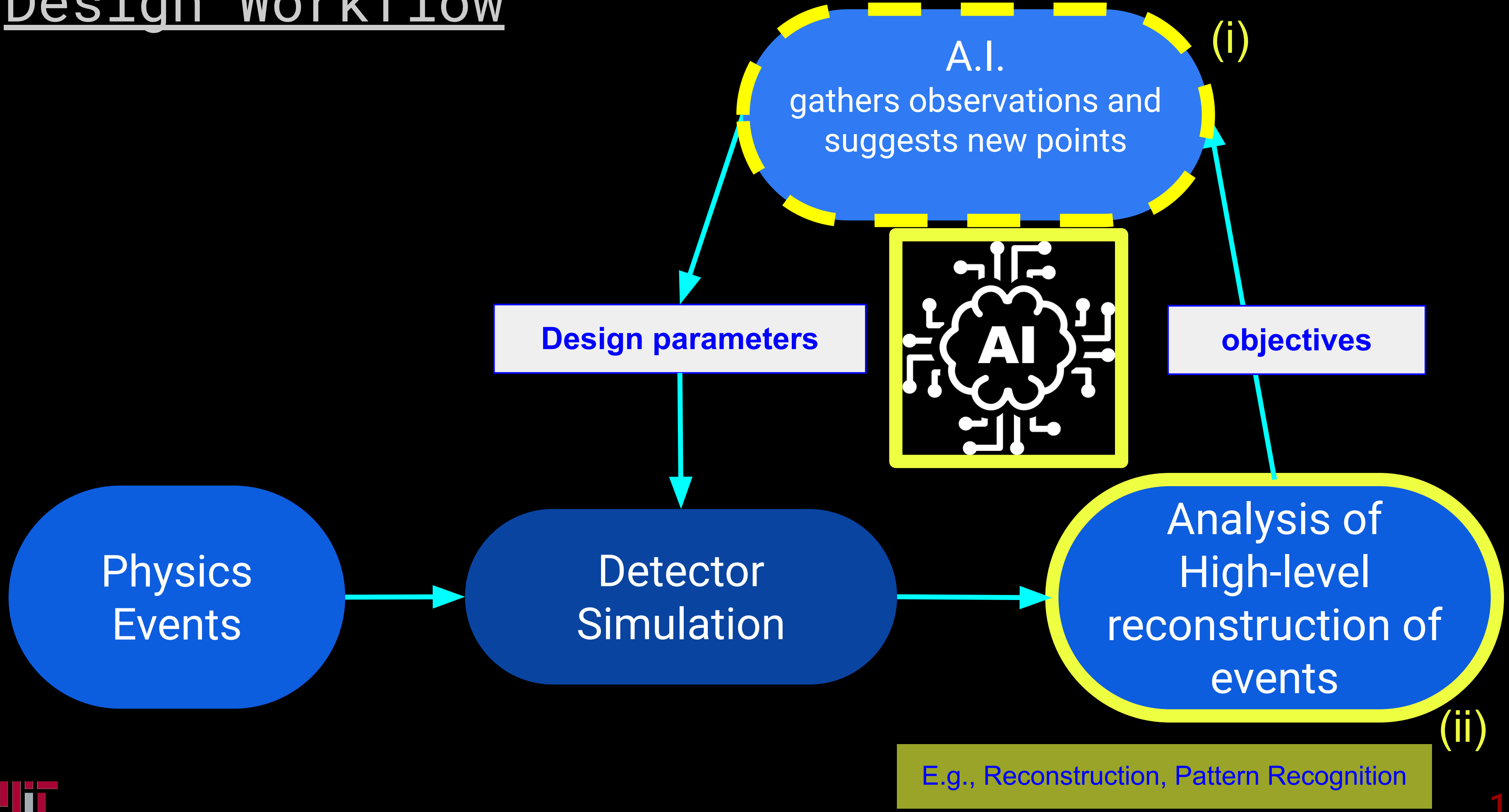
Improving the workflow

Design Workflow

Developed to cope with complex problems which are computationally expensive in order to reduce the number of evaluations needed for the optimization

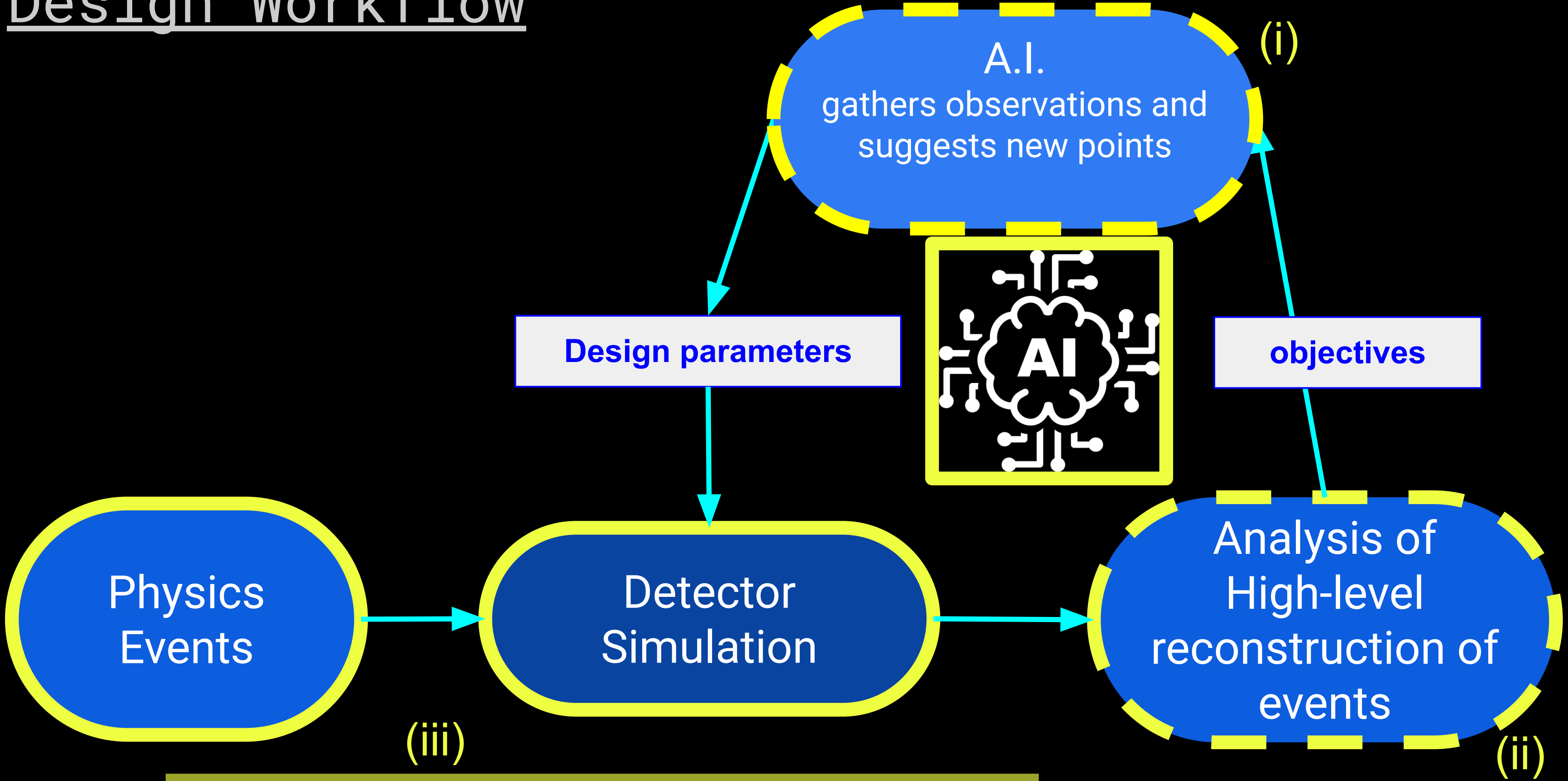

A.I.
gathers observations and suggests new points
(i)
Design parameters
objectives
AI
Physics Events
Detector Simulation
Analysis of High-level reconstruction of events
✔
✔
✔
✔

Design Workflow



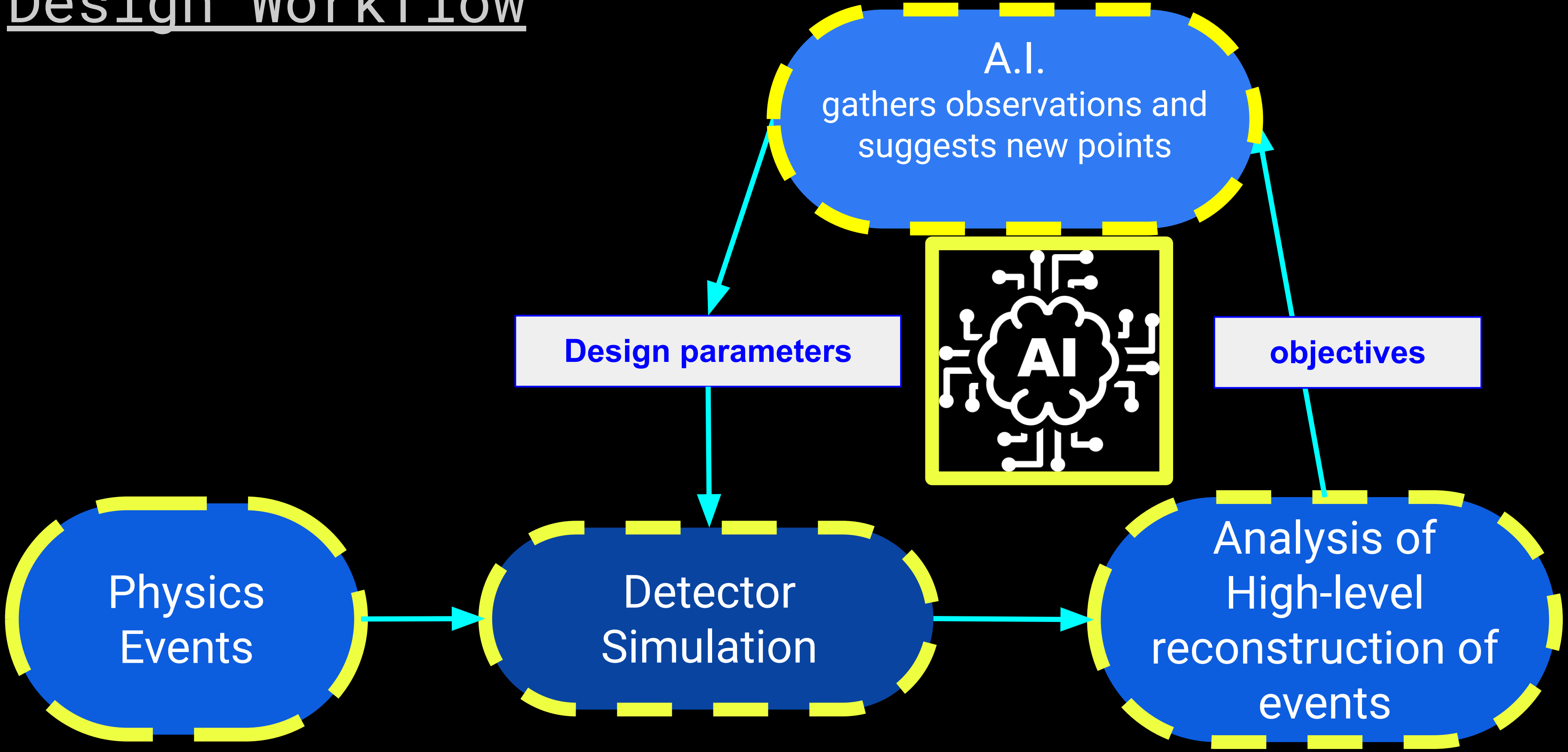
A.I.
gathers observations and suggests new points
(i)
Design parameters
objectives
AI
Physics Events
Detector Simulation
Analysis of High-level reconstruction of events
(ii)
E.g., Reconstruction, Pattern Recognition

Design Workflow



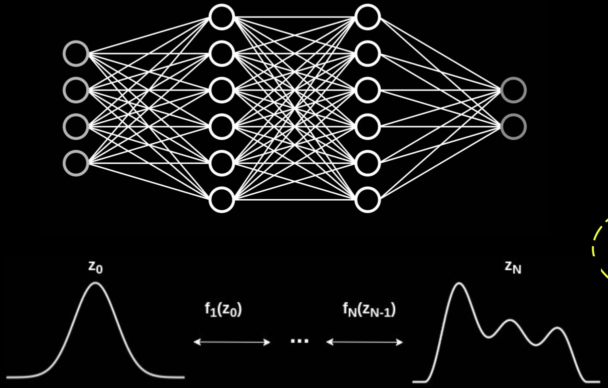
Interactions of simulated particle with matter / detector response

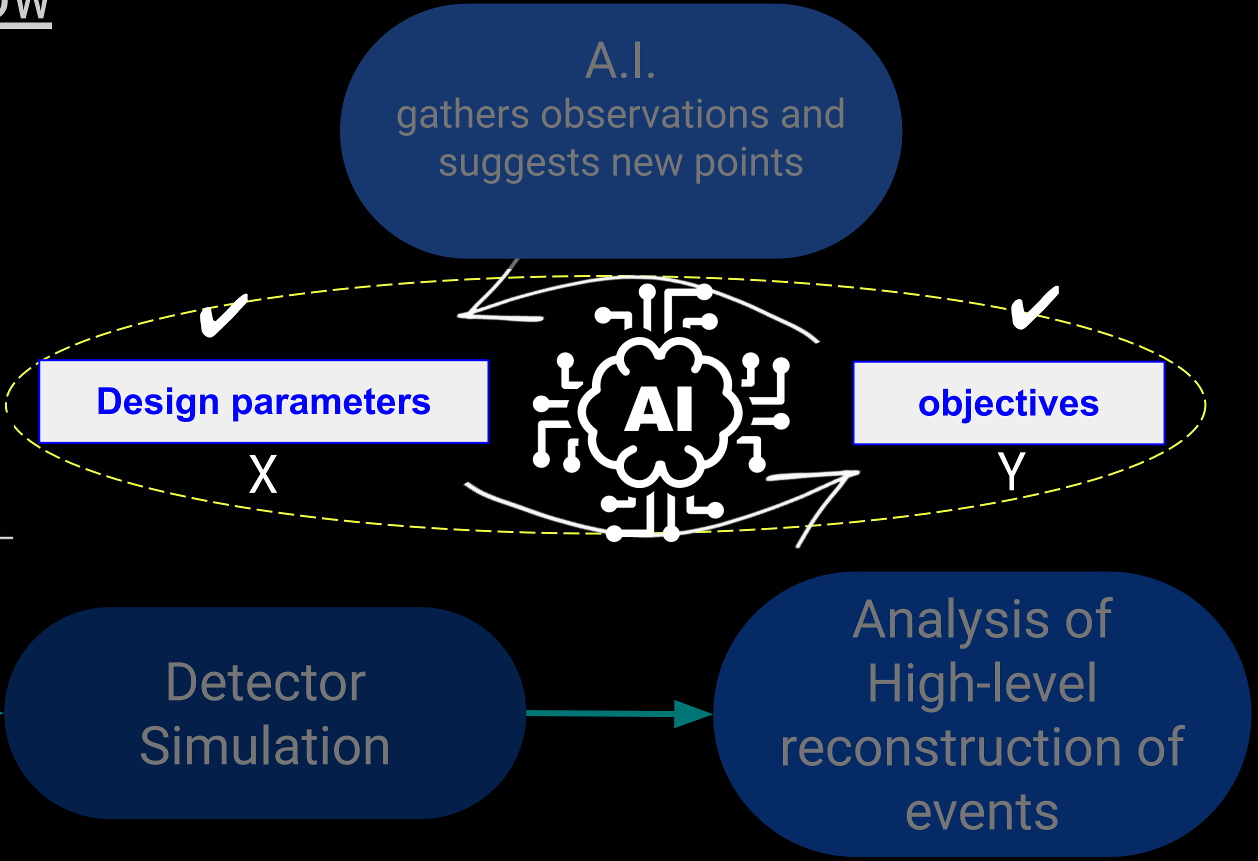
Design Workflow


A.I.
gathers observations and suggests new points
Design parameters
objectives
AI
Physics Events
Detector Simulation
Analysis of High-level reconstruction of events

AI/ML can potentially enter in all the steps of the design pipeline

Design Workflow





A.I.
gathers observations and suggests new points
✔
Design parameters
X
AI
✔
objectives
Y
Physics Events
Detector Simulation
Analysis of High-level reconstruction of events

…with large datasets…

Speed-up

- In general speed-up is reached by:
  - Hardware-based solutions to accelerate traditional algorithms
  - Hybrid ML/traditional techniques
  - End-to-end ML methods
- Of course the overall performance improvement gained by optimizing a single part of a system is limited by the fraction of time that the improved part is actually used (Amdhal's law)

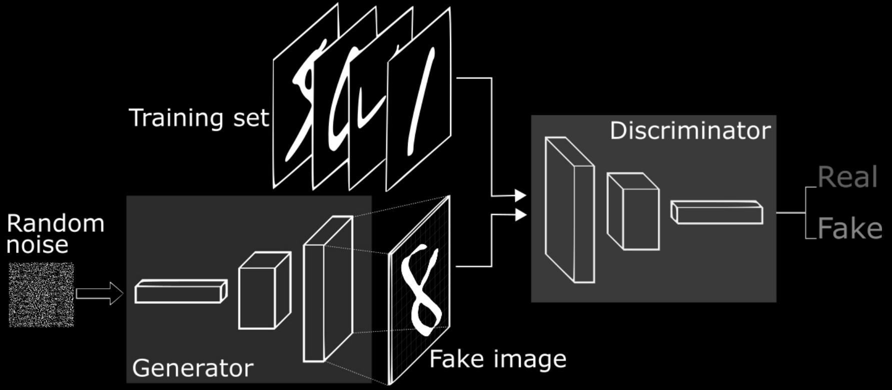
$$\frac{1}{(1-P)+\frac{P}{S}}$$

P: fraction of execution time that the part benefiting from improved resources originally occupied

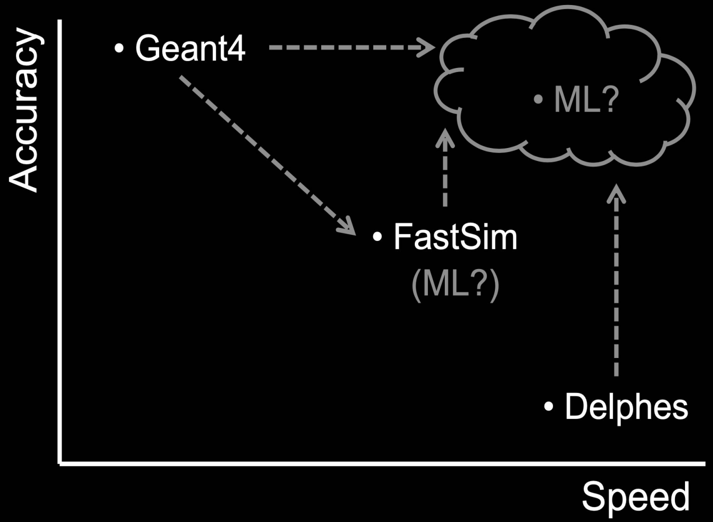
s: speed-up of the part benefiting

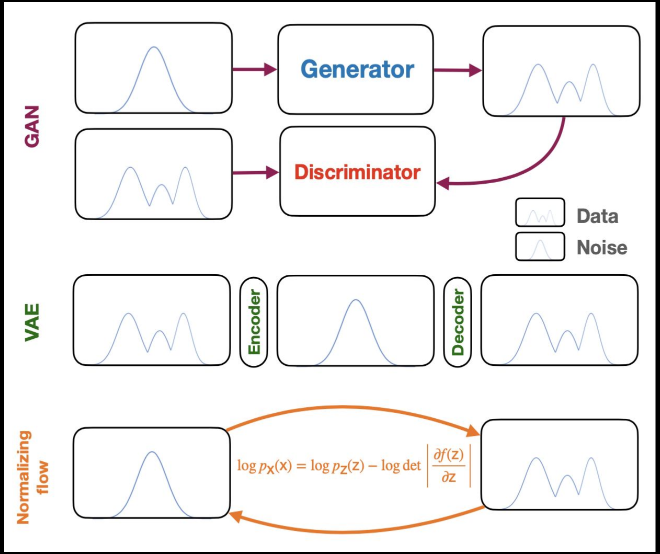
- What follows will show some ML/DL example and is not meant to be exhaustive — argument for another talk; see P. Harris' lectures

Learning Interactions of Simulated Particle with Matter



ML-accelerated "Simulations"



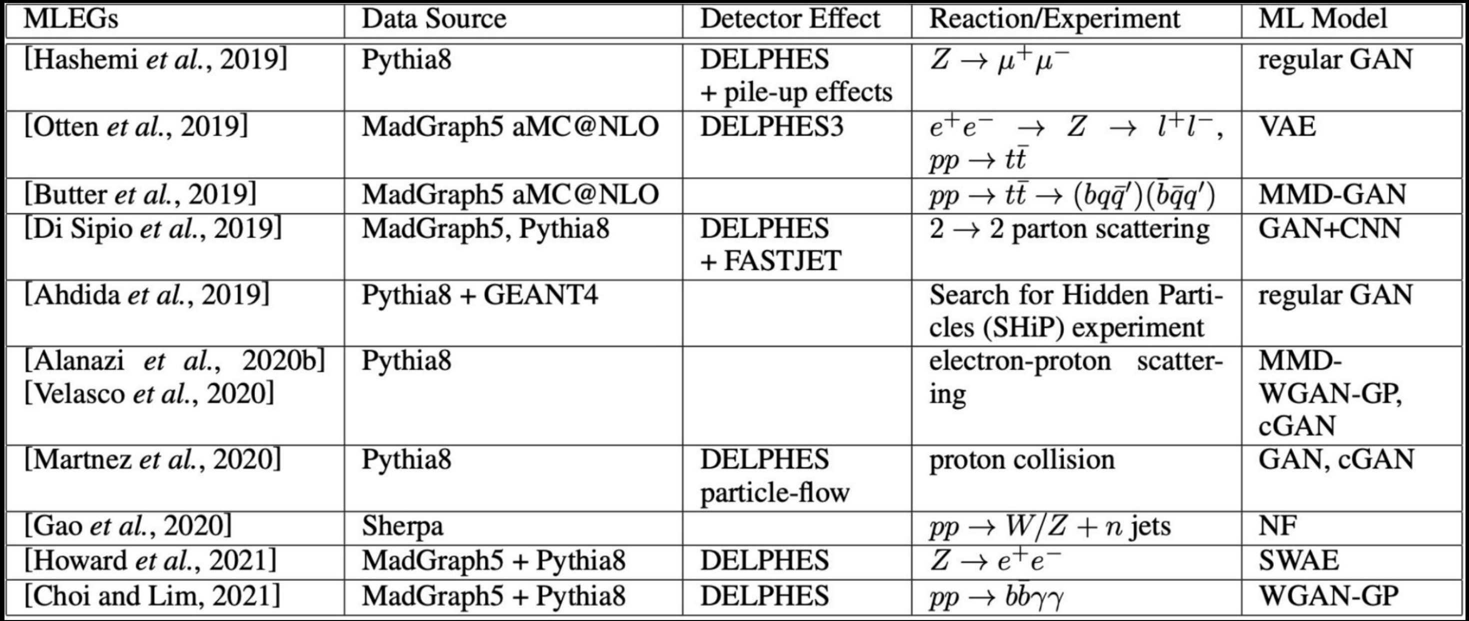

GAN
Generator
Discriminator
Data
Noise
VAE
Encoder
Decoder
Normalizing flow
$\log p_X(x) = \log p_Z(z) - \log \det \left| \frac{\partial f(z)}{\partial z} \right|$

- Computational demands for simulation of current and next generation HEP experiments inspired investigation of surrogates using deep generative models (GAN, VAE, NF based) to decrease simulation time while maintaining fidelity — "real" and "fake" harder to distinguish with NF
- Complex detectors require many fully simulated events as a dataset for the ML architecture
- Notice that a new detector design requires a new dataset…

A. Adelmann et al., New directions for surrogate models and differentiable programming for HEP detector simulation, arXiv:2203.08806v1 and references therein

Survey of ML-based Event Generators

Courtesy of M. Kuchera,
AI4EIC — What can AI offer for Simulation at the EIC?

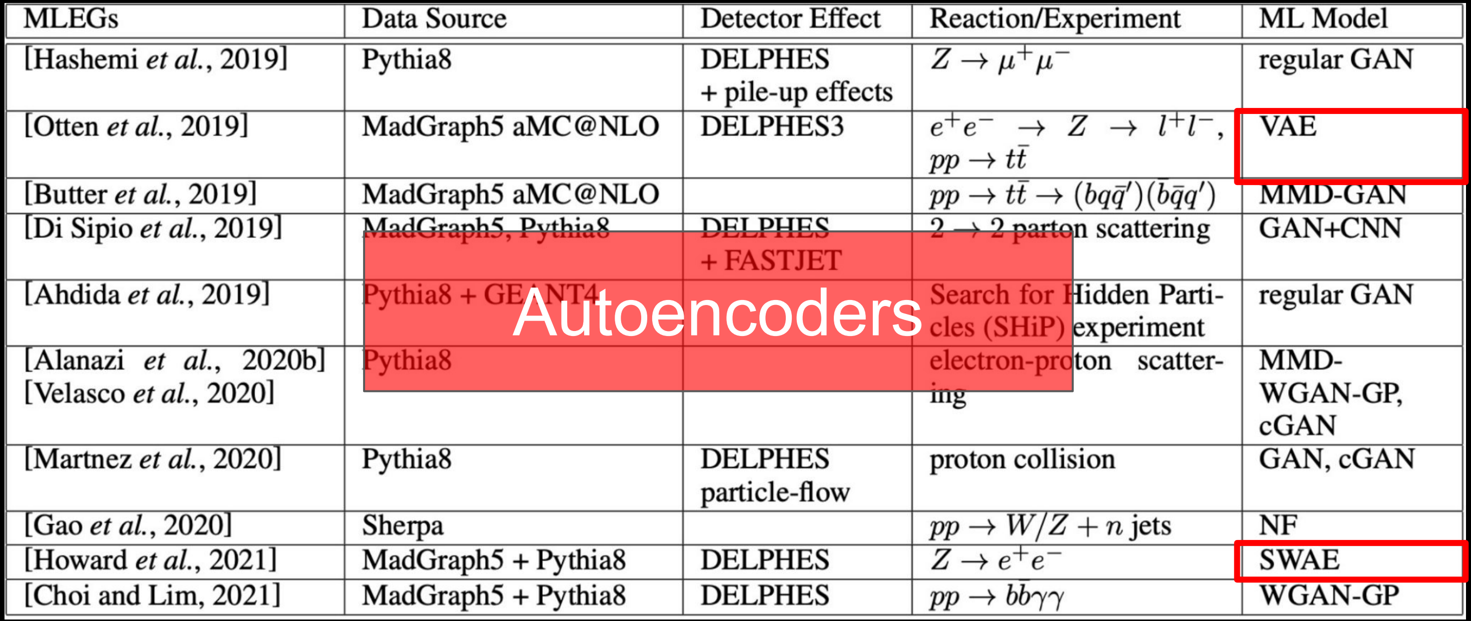


MLEGs	Data Source	Detector Effect	Reaction/Experiment	ML Model
[Hashemi *et al.*, 2019]	Pythia8	DELPHES + pile-up effects	$Z \to \mu^+\mu^-$	regular GAN
[Otten *et al.*, 2019]	MadGraph5 aMC@NLO	DELPHES3	$e^+e^- \to Z \to l^+l^-$, $pp \to t\bar{t}$	VAE
[Butter *et al.*, 2019]	MadGraph5 aMC@NLO		$pp \to t\bar{t} \to (bq\bar{q}')(\bar{b}\bar{q}q')$	MMD-GAN
[Di Sipio *et al.*, 2019]	MadGraph5, Pythia8	DELPHES + FASTJET	$2 \to 2$ parton scattering	GAN+CNN
[Ahdida *et al.*, 2019]	Pythia8 + GEANT4		Search for Hidden Particles (SHiP) experiment	regular GAN
[Alanazi *et al.*, 2020b] [Velasco *et al.*, 2020]	Pythia8		electron-proton scattering	MMD-WGAN-GP, cGAN
[Martnez *et al.*, 2020]	Pythia8	DELPHES particle-flow	proton collision	GAN, cGAN
[Gao *et al.*, 2020]	Sherpa		$pp \to W/Z + n$ jets	NF
[Howard *et al.*, 2021]	MadGraph5 + Pythia8	DELPHES	$Z \to e^+e^-$	SWAE
[Choi and Lim, 2021]	MadGraph5 + Pythia8	DELPHES	$pp \to b\bar{b}\gamma\gamma$	WGAN-GP

There is a rich literature
Generally aiming at significantly faster simulations without sacrificing physics accuracy

Y. Alanazi et al, A Survey of ML-Based Physics Event Generation, Proceedings of the XXX IJCAI-21

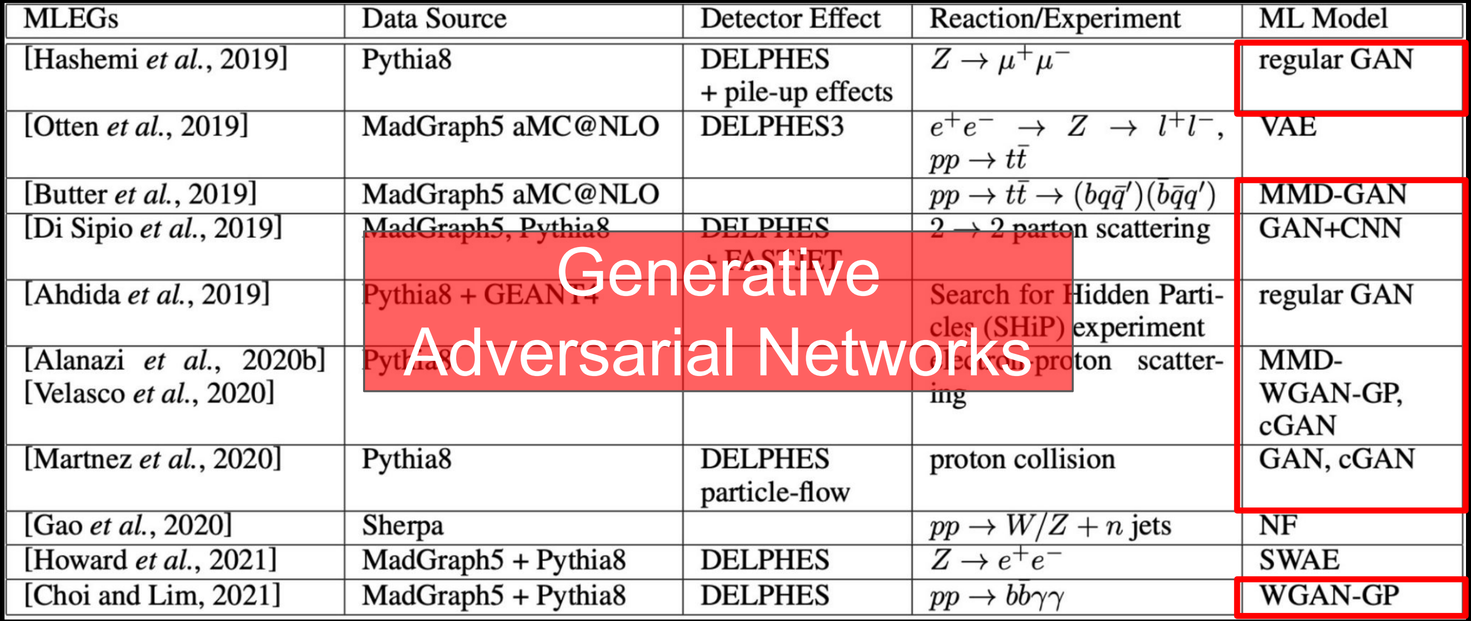
Survey of ML-based Event Generators



MLEGs	Data Source	Detector Effect	Reaction/Experiment	ML Model
[Hashemi *et al.*, 2019]	Pythia8	DELPHES + pile-up effects	$Z \rightarrow \mu^+\mu^-$	regular GAN
[Otten *et al.*, 2019]	MadGraph5 aMC@NLO	DELPHES3	$e^+e^- \rightarrow Z \rightarrow l^+l^-$, $pp \rightarrow t\bar{t}$	VAE
[Butter *et al.*, 2019]	MadGraph5 aMC@NLO		$pp \rightarrow t\bar{t} \rightarrow (bq\bar{q}')(\bar{b}\bar{q}q')$	MMD-GAN
[Di Sipio *et al.*, 2019]	MadGraph5, Pythia8	DELPHES + FASTJET	$2 \rightarrow 2$ parton scattering	GAN+CNN
[Ahdida *et al.*, 2019]	Pythia8 + GEANT4		Search for Hidden Particles (SHiP) experiment	regular GAN
[Alanazi *et al.*, 2020b] [Velasco *et al.*, 2020]	Pythia8		electron-proton scattering	MMD-WGAN-GP, cGAN
[Martnez *et al.*, 2020]	Pythia8	DELPHES particle-flow	proton collision	GAN, cGAN
[Gao *et al.*, 2020]	Sherpa		$pp \rightarrow W/Z + n$ jets	NF
[Howard *et al.*, 2021]	MadGraph5 + Pythia8	DELPHES	$Z \rightarrow e^+e^-$	SWAE
[Choi and Lim, 2021]	MadGraph5 + Pythia8	DELPHES	$pp \rightarrow b\bar{b}\gamma\gamma$	WGAN-GP

Autoencoders

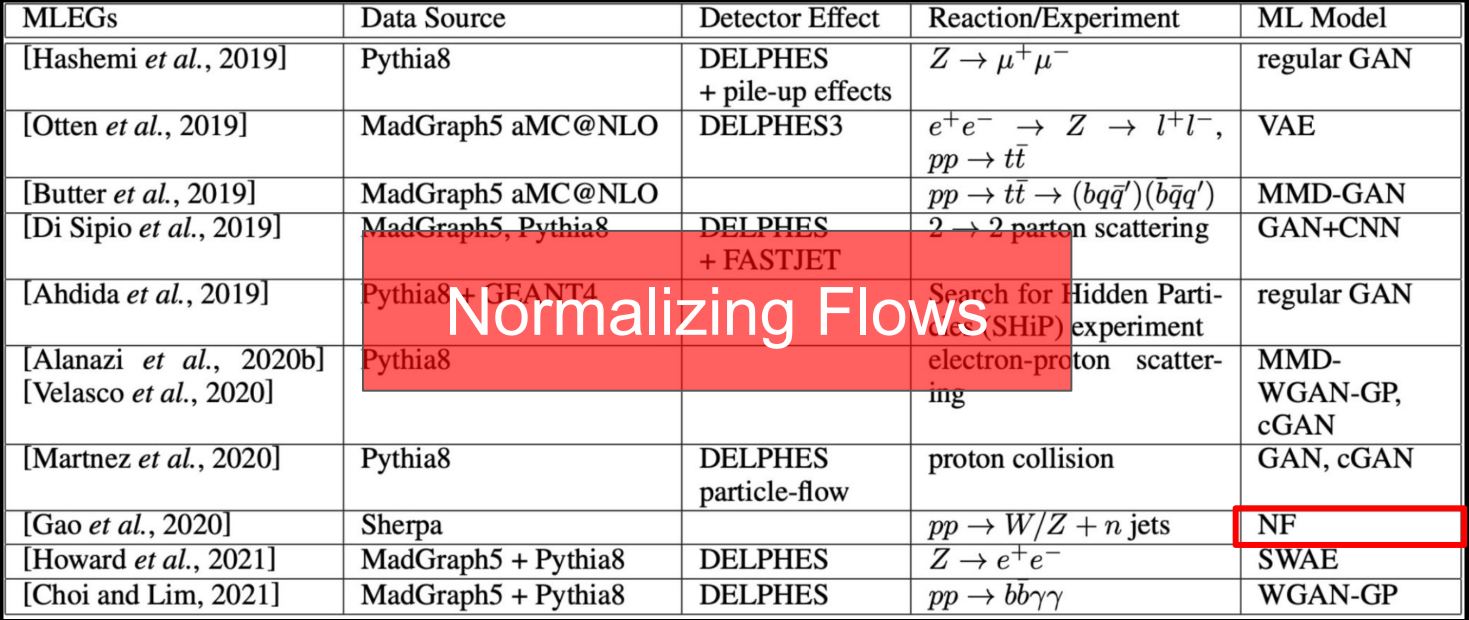
Survey of ML-based Event Generators



MLEGs	Data Source	Detector Effect	Reaction/Experiment	ML Model
[Hashemi *et al.*, 2019]	Pythia8	DELPHES + pile-up effects	$Z \to \mu^+\mu^-$	regular GAN
[Otten *et al.*, 2019]	MadGraph5 aMC@NLO	DELPHES3	$e^+e^- \to Z \to l^+l^-$, $pp \to t\bar{t}$	VAE
[Butter *et al.*, 2019]	MadGraph5 aMC@NLO		$pp \to t\bar{t} \to (bq\bar{q}')(\bar{b}\bar{q}q')$	MMD-GAN
[Di Sipio *et al.*, 2019]	MadGraph5, Pythia8	DELPHES + FASTJET	2 → 2 parton scattering	GAN+CNN
[Ahdida *et al.*, 2019]	Pythia8 + GEANT4		Search for Hidden Particles (SHiP) experiment	regular GAN
[Alanazi *et al.*, 2020b] [Velasco *et al.*, 2020]	Pythia8		electron-proton scattering	MMD-WGAN-GP, cGAN
[Martnez *et al.*, 2020]	Pythia8	DELPHES particle-flow	proton collision	GAN, cGAN
[Gao *et al.*, 2020]	Sherpa		$pp \to W/Z + n$ jets	NF
[Howard *et al.*, 2021]	MadGraph5 + Pythia8	DELPHES	$Z \to e^+e^-$	SWAE
[Choi and Lim, 2021]	MadGraph5 + Pythia8	DELPHES	$pp \to b\bar{b}\gamma\gamma$	WGAN-GP

Generative Adversarial Networks

Survey of ML-based Event Generators



MLEGs	Data Source	Detector Effect	Reaction/Experiment	ML Model
[Hashemi *et al.*, 2019]	Pythia8	DELPHES + pile-up effects	$Z \to \mu^+\mu^-$	regular GAN
[Otten *et al.*, 2019]	MadGraph5 aMC@NLO	DELPHES3	$e^+e^- \to Z \to l^+l^-$, $pp \to t\bar{t}$	VAE
[Butter *et al.*, 2019]	MadGraph5 aMC@NLO		$pp \to t\bar{t} \to (bq\bar{q}')(\bar{b}\bar{q}q')$	MMD-GAN
[Di Sipio *et al.*, 2019]	MadGraph5, Pythia8	DELPHES + FASTJET	$2 \to 2$ parton scattering	GAN+CNN
[Ahdida *et al.*, 2019]	Pythia8 + GEANT4		Search for Hidden Particles (SHiP) experiment	regular GAN
[Alanazi *et al.*, 2020b] [Velasco *et al.*, 2020]	Pythia8		electron-proton scattering	MMD-WGAN-GP, cGAN
[Martnez *et al.*, 2020]	Pythia8	DELPHES particle-flow	proton collision	GAN, cGAN
[Gao *et al.*, 2020]	Sherpa		$pp \to W/Z + n$ jets	NF
[Howard *et al.*, 2021]	MadGraph5 + Pythia8	DELPHES	$Z \to e^+e^-$	SWAE
[Choi and Lim, 2021]	MadGraph5 + Pythia8	DELPHES	$pp \to b\bar{b}\gamma\gamma$	WGAN-GP

Normalizing Flows

Variational Autoencoders

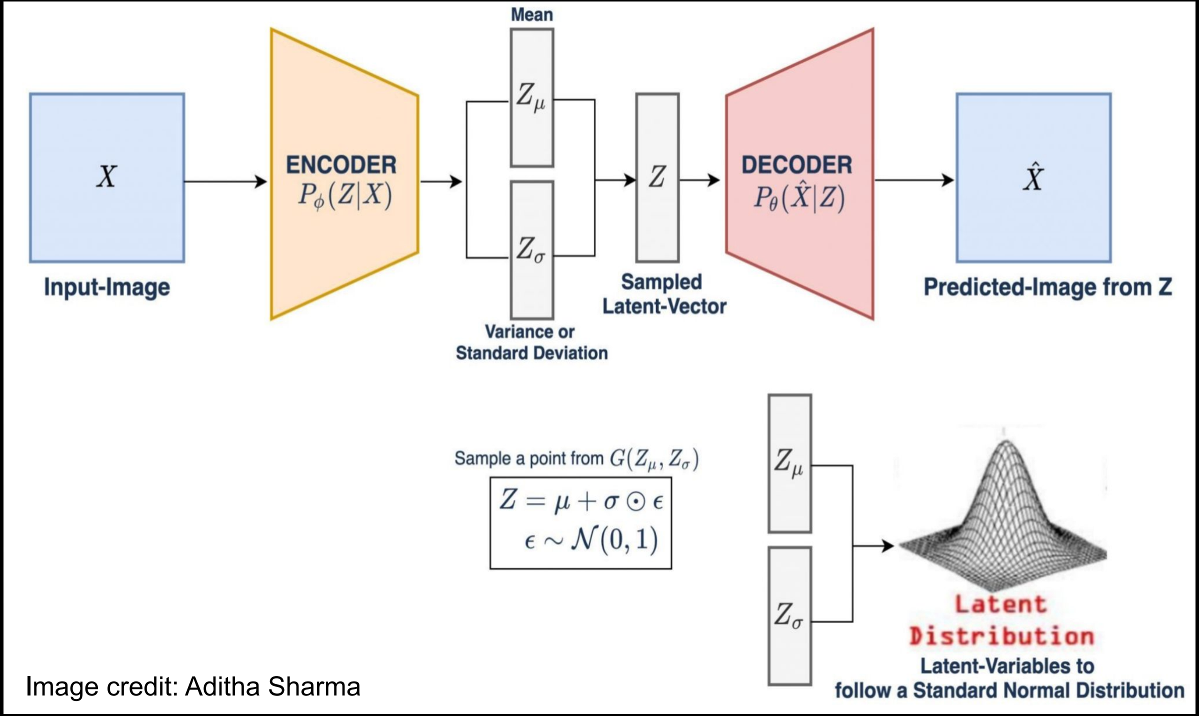
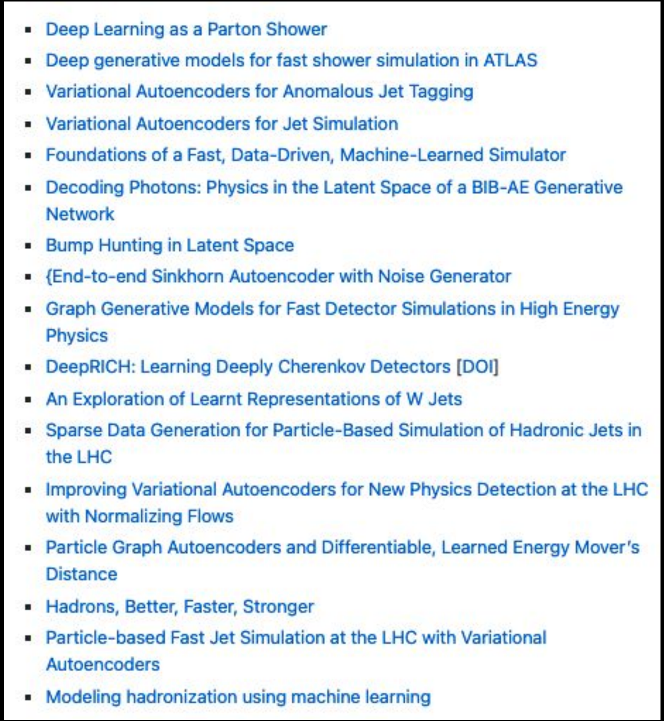

Mean
X
Input-Image
ENCODER
$P_\phi(Z|X)$
Z_μ
Z_σ
Variance or Standard Deviation
Z
Sampled Latent-Vector
DECODER
$P_\theta(\hat{X}|Z)$
$\hat{X}$
Predicted-Image from Z
Sample a point from $G(Z_\mu, Z_\sigma)$
$Z = \mu + \sigma \odot \epsilon$
$\epsilon \sim \mathcal{N}(0,1)$
Z_μ
Z_σ
Latent Distribution
Latent-Variables to follow a Standard Normal Distribution

Image credit: Aditha Sharma

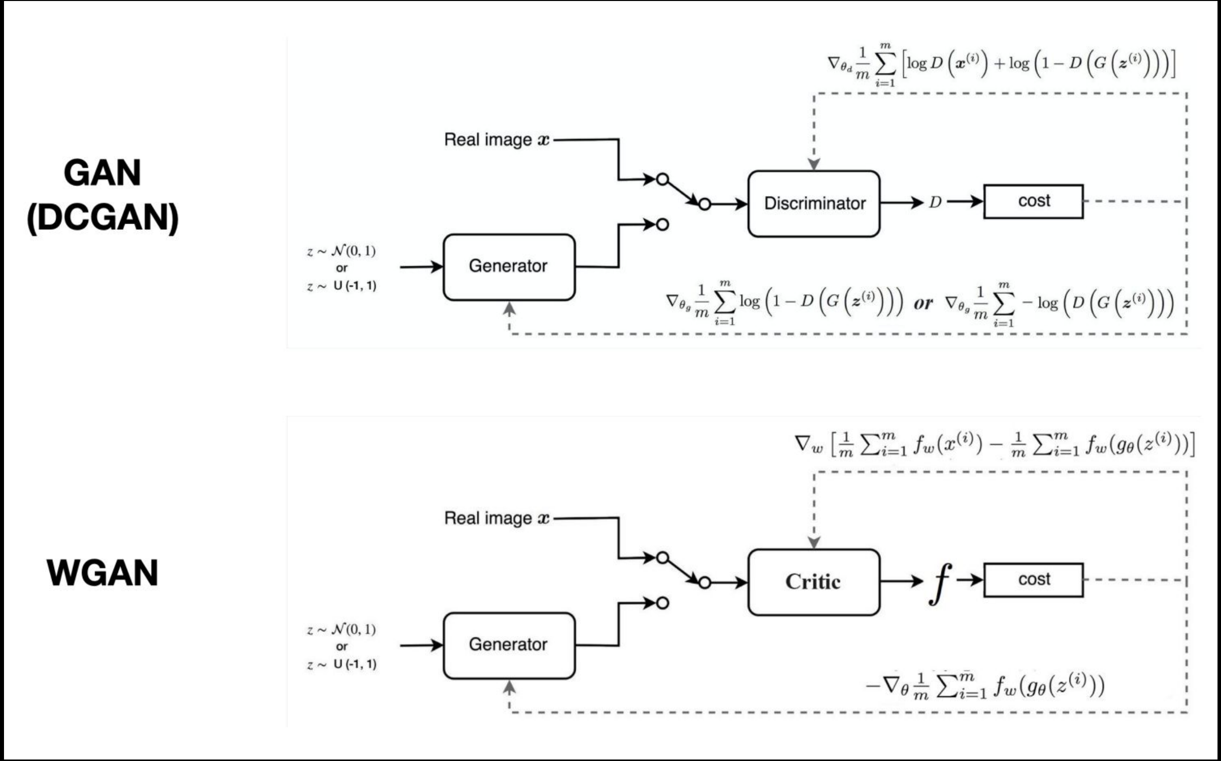
Examples in HEP/NP



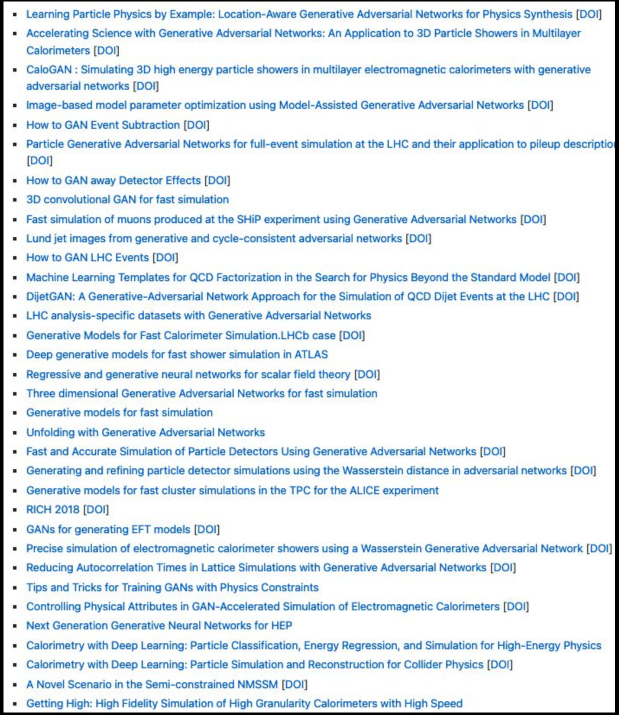
- Deep Learning as a Parton Shower
- Deep generative models for fast shower simulation in ATLAS
- Variational Autoencoders for Anomalous Jet Tagging
- Variational Autoencoders for Jet Simulation
- Foundations of a Fast, Data-Driven, Machine-Learned Simulator
- Decoding Photons: Physics in the Latent Space of a BIB-AE Generative Network
- Bump Hunting in Latent Space
- {End-to-end Sinkhorn Autoencoder with Noise Generator
- Graph Generative Models for Fast Detector Simulations in High Energy Physics
- DeepRICH: Learning Deeply Cherenkov Detectors [DOI]
- An Exploration of Learnt Representations of W Jets
- Sparse Data Generation for Particle-Based Simulation of Hadronic Jets in the LHC
- Improving Variational Autoencoders for New Physics Detection at the LHC with Normalizing Flows
- Particle Graph Autoencoders and Differentiable, Learned Energy Mover's Distance
- Hadrons, Better, Faster, Stronger
- Particle-based Fast Jet Simulation at the LHC with Variational Autoencoders
- Modeling hadronization using machine learning

More examples in HEPML-LivingReview: A Living Review of Machine Learning for Particle Physics

Generative Adversarial Networks


GAN (DCGAN)
$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(x^{(i)}\right) + \log\left(1 - D\left(G\left(z^{(i)}\right)\right)\right)\right]$
Real image x
Discriminator
D
cost
$z \sim \mathcal{N}(0,1)$ or $z \sim$ U (-1, 1)
Generator
$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(z^{(i)}\right)\right)\right)$ or $\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} -\log\left(D\left(G\left(z^{(i)}\right)\right)\right)$
WGAN
$\nabla_w \left[\frac{1}{m} \sum_{i=1}^{m} f_w(x^{(i)}) - \frac{1}{m} \sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))\right]$
Real image x
Critic
f
cost
$z \sim \mathcal{N}(0,1)$ or $z \sim$ U (-1, 1)
Generator
$-\nabla_\theta \frac{1}{m} \sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))$

Examples in HEP/NP

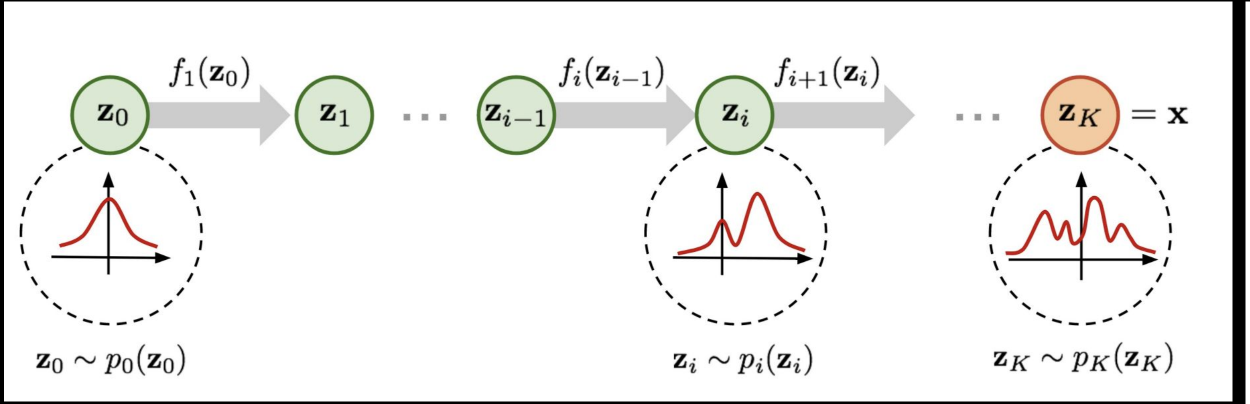


- Learning Particle Physics by Example: Location-Aware Generative Adversarial Networks for Physics Synthesis [DOI]
- Accelerating Science with Generative Adversarial Networks: An Application to 3D Particle Showers in Multilayer Calorimeters [DOI]
- CaloGAN : Simulating 3D high energy particle showers in multilayer electromagnetic calorimeters with generative adversarial networks [DOI]
- Image-based model parameter optimization using Model-Assisted Generative Adversarial Networks [DOI]
- How to GAN Event Subtraction [DOI]
- Particle Generative Adversarial Networks for full-event simulation at the LHC and their application to pileup descriptio [DOI]
- How to GAN away Detector Effects [DOI]
- 3D convolutional GAN for fast simulation
- Fast simulation of muons produced at the SHiP experiment using Generative Adversarial Networks [DOI]
- Lund jet images from generative and cycle-consistent adversarial networks [DOI]
- How to GAN LHC Events [DOI]
- Machine Learning Templates for QCD Factorization in the Search for Physics Beyond the Standard Model [DOI]
- DijetGAN: A Generative-Adversarial Network Approach for the Simulation of QCD Dijet Events at the LHC [DOI]
- LHC analysis-specific datasets with Generative Adversarial Networks
- Generative Models for Fast Calorimeter Simulation.LHCb case [DOI]
- Deep generative models for fast shower simulation in ATLAS
- Regressive and generative neural networks for scalar field theory [DOI]
- Three dimensional Generative Adversarial Networks for fast simulation
- Generative models for fast simulation
- Unfolding with Generative Adversarial Networks
- Fast and Accurate Simulation of Particle Detectors Using Generative Adversarial Networks [DOI]
- Generating and refining particle detector simulations using the Wasserstein distance in adversarial networks [DOI]
- Generative models for fast cluster simulations in the TPC for the ALICE experiment
- RICH 2018 [DOI]
- GANs for generating EFT models [DOI]
- Precise simulation of electromagnetic calorimeter showers using a Wasserstein Generative Adversarial Network [DOI]
- Reducing Autocorrelation Times in Lattice Simulations with Generative Adversarial Networks [DOI]
- Tips and Tricks for Training GANs with Physics Constraints
- Controlling Physical Attributes in GAN-Accelerated Simulation of Electromagnetic Calorimeters [DOI]
- Next Generation Generative Neural Networks for HEP
- Calorimetry with Deep Learning: Particle Classification, Energy Regression, and Simulation for High-Energy Physics
- Calorimetry with Deep Learning: Particle Simulation and Reconstruction for Collider Physics [DOI]
- A Novel Scenario in the Semi-constrained NMSSM [DOI]
- Getting High: High Fidelity Simulation of High Granularity Calorimeters with High Speed

etc

More examples in HEPML-LivingReview: A Living Review of Machine Learning for Particle Physics

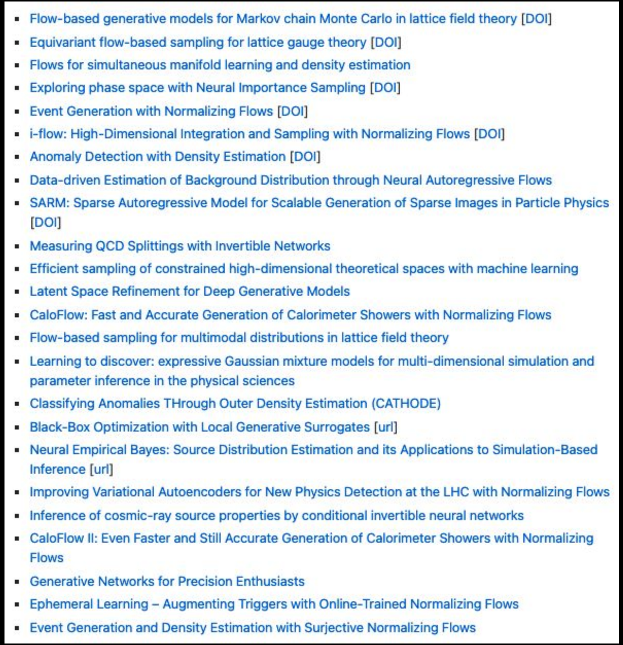
Normalizing Flows



$\mathbf{z}_0$
$f_1(\mathbf{z}_0)$
$\mathbf{z}_1$
$\mathbf{z}_{i-1}$
$f_i(\mathbf{z}_{i-1})$
$\mathbf{z}_i$
$f_{i+1}(\mathbf{z}_i)$
$\mathbf{z}_K = \mathbf{x}$
$\mathbf{z}_0 \sim p_0(\mathbf{z}_0)$
$\mathbf{z}_i \sim p_i(\mathbf{z}_i)$
$\mathbf{z}_K \sim p_K(\mathbf{z}_K)$

Maps complex distributions by transforming a probability density through a series of invertible mappings.

Examples in HEP/NP

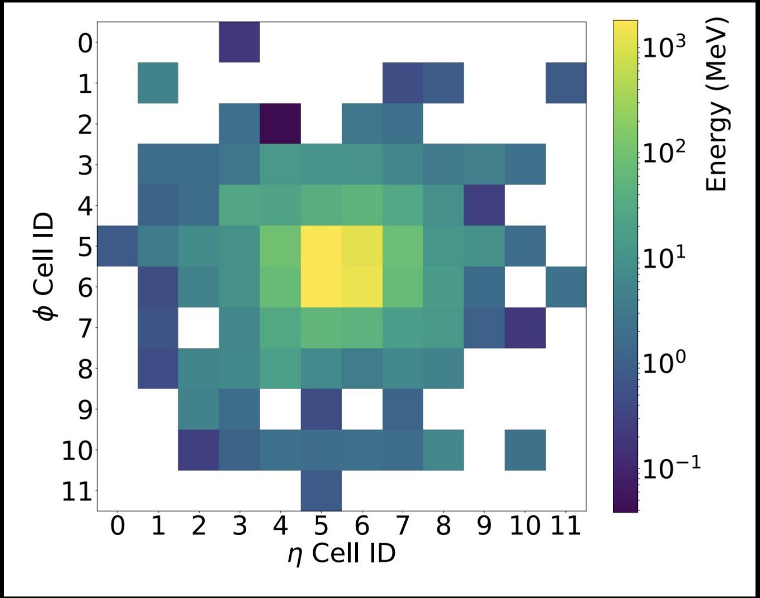


- Flow-based generative models for Markov chain Monte Carlo in lattice field theory [DOI]
- Equivariant flow-based sampling for lattice gauge theory [DOI]
- Flows for simultaneous manifold learning and density estimation
- Exploring phase space with Neural Importance Sampling [DOI]
- Event Generation with Normalizing Flows [DOI]
- i-flow: High-Dimensional Integration and Sampling with Normalizing Flows [DOI]
- Anomaly Detection with Density Estimation [DOI]
- Data-driven Estimation of Background Distribution through Neural Autoregressive Flows
- SARM: Sparse Autoregressive Model for Scalable Generation of Sparse Images in Particle Physics [DOI]
- Measuring QCD Splittings with Invertible Networks
- Efficient sampling of constrained high-dimensional theoretical spaces with machine learning
- Latent Space Refinement for Deep Generative Models
- CaloFlow: Fast and Accurate Generation of Calorimeter Showers with Normalizing Flows
- Flow-based sampling for multimodal distributions in lattice field theory
- Learning to discover: expressive Gaussian mixture models for multi-dimensional simulation and parameter inference in the physical sciences
- Classifying Anomalies THrough Outer Density Estimation (CATHODE)
- Black-Box Optimization with Local Generative Surrogates [url]
- Neural Empirical Bayes: Source Distribution Estimation and its Applications to Simulation-Based Inference [url]
- Improving Variational Autoencoders for New Physics Detection at the LHC with Normalizing Flows
- Inference of cosmic-ray source properties by conditional invertible neural networks
- CaloFlow II: Even Faster and Still Accurate Generation of Calorimeter Showers with Normalizing Flows
- Generative Networks for Precision Enthusiasts
- Ephemeral Learning – Augmenting Triggers with Online-Trained Normalizing Flows
- Event Generation and Density Estimation with Surjective Normalizing Flows

More examples in HEPML-LivingReview: A Living Review of Machine Learning for Particle Physics

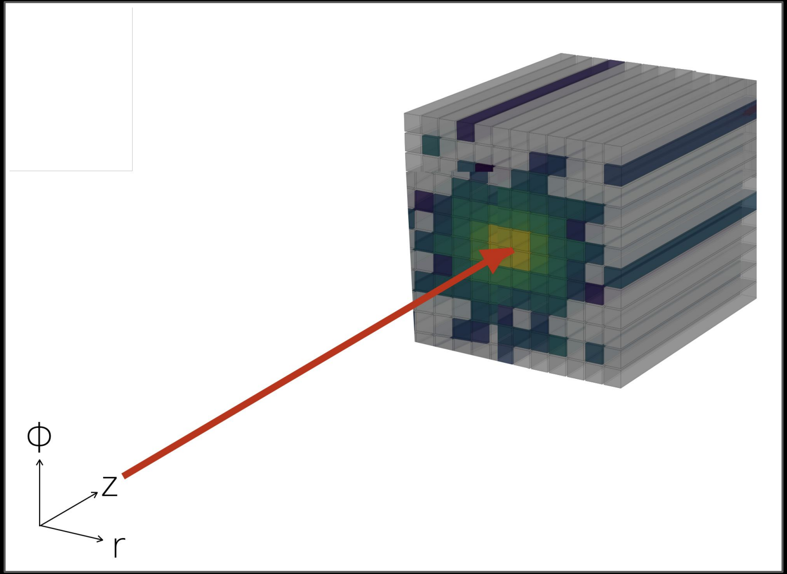
Accelerating Detector Simulations

Courtesy of B. Nachman, AI4EIC — Generative ML applications for simulations in colliders



Calorimeters are often the slowest to simulate

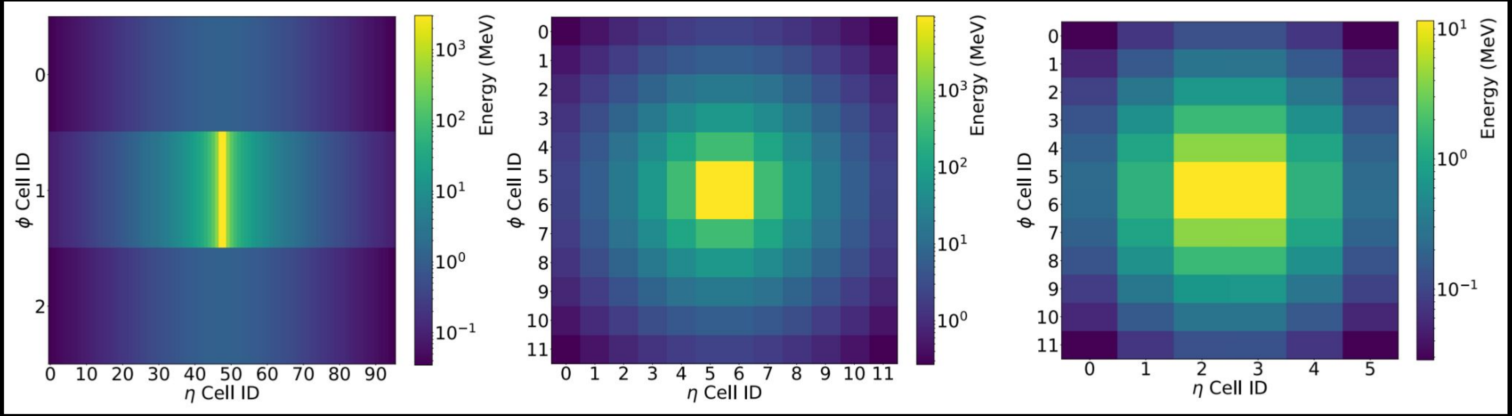
Stopping particles requires simulating interactions of all energies

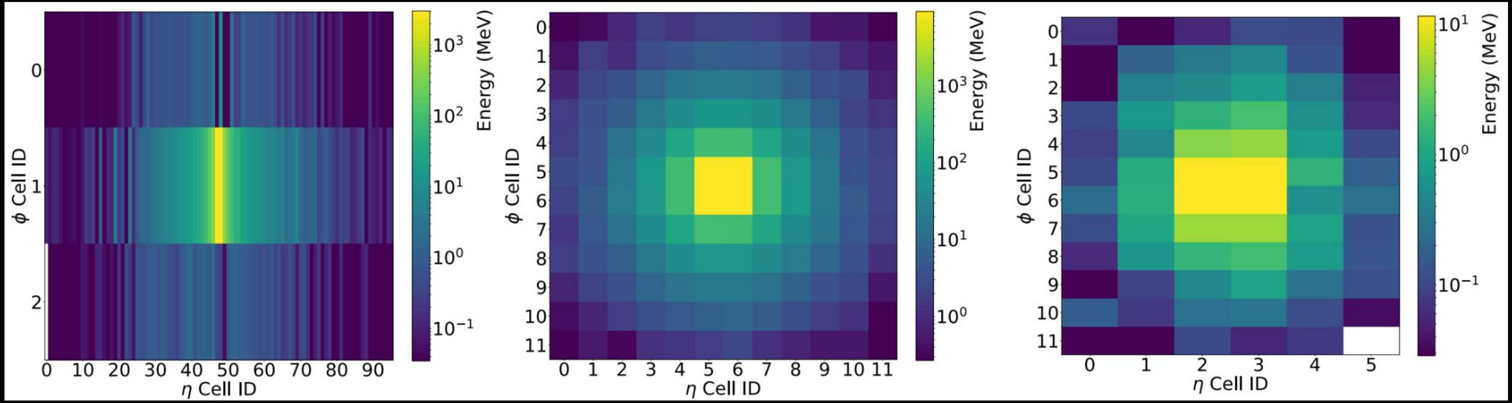


Pixel intensity = energy deposited

Performance: average images

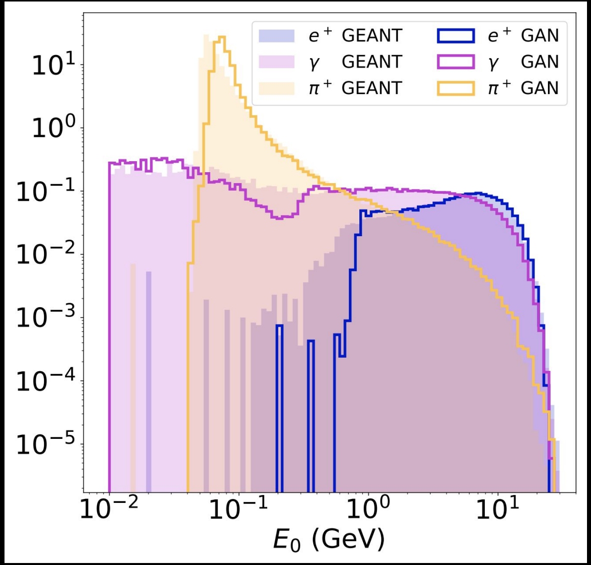
Geant4

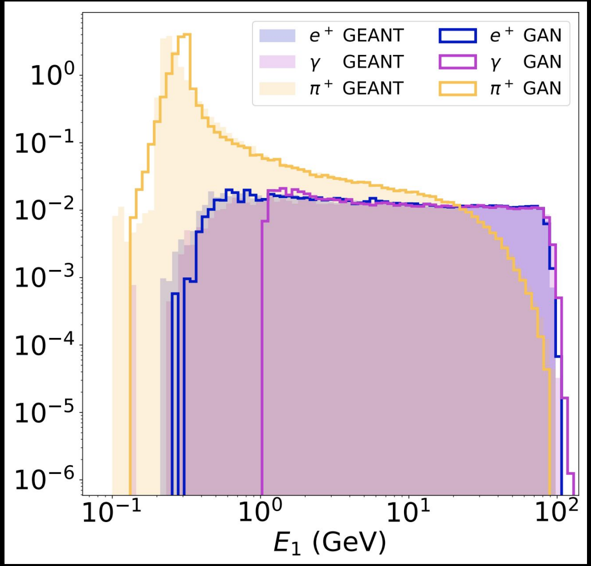


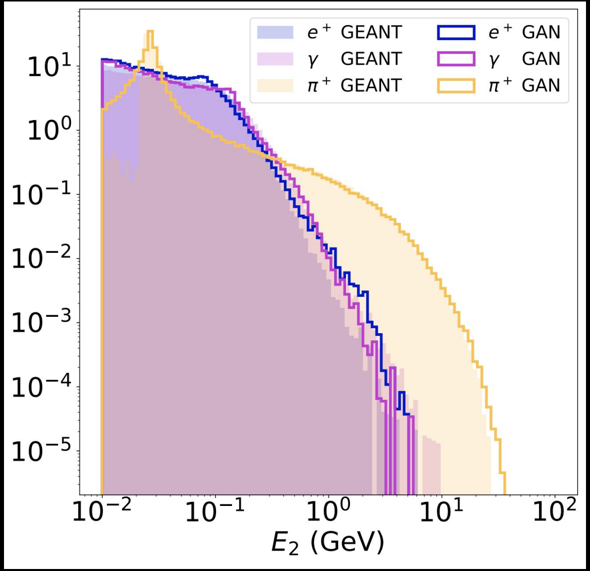


CaloGAN

Performance: Energy per Layer





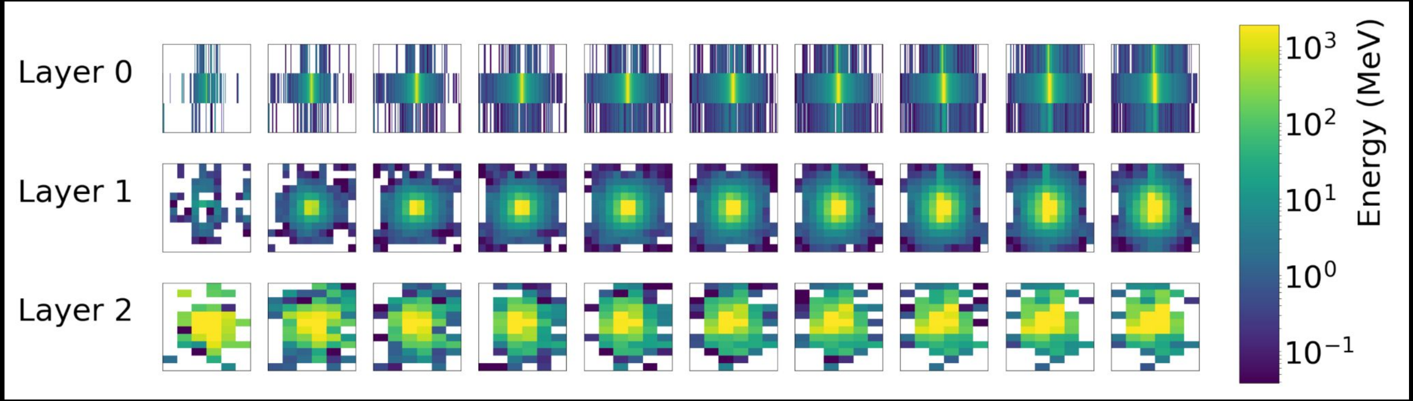


- Comparison of shower shape variables
- Pions deposit much less energy in the first layers; leave the calorimeter with significant energy

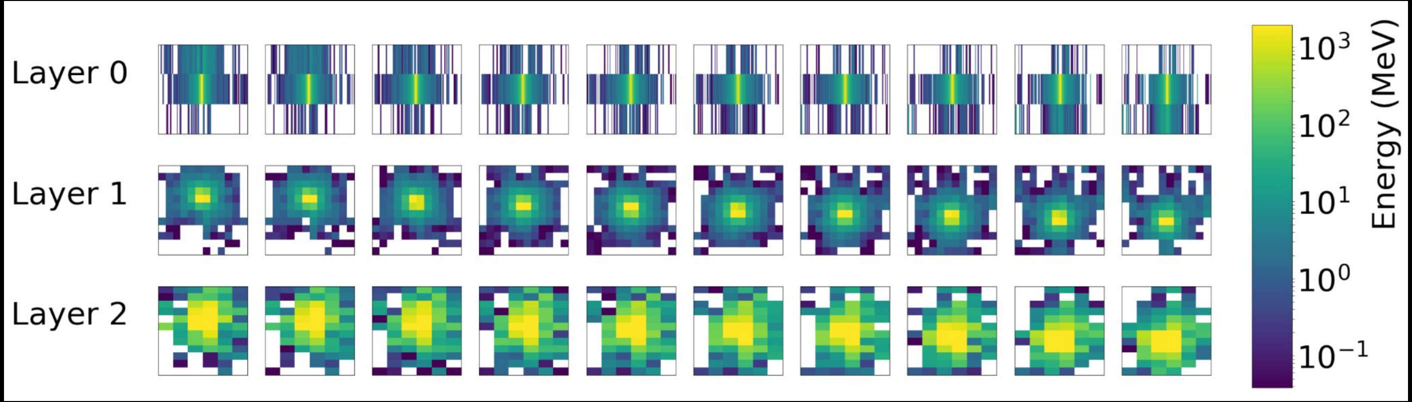
M. Paganini, L. de Oliveira, B. Nachman, 1712.10321 (2017)

Conditioning

Fix noise, scan latent variable corresponding to energy



Fix noise, scan latent variable corresponding to x-position



L. de Oliveira, M. Paganini, B. Nachman, 1711.08813 (2017)

Timing Performance / Speedup

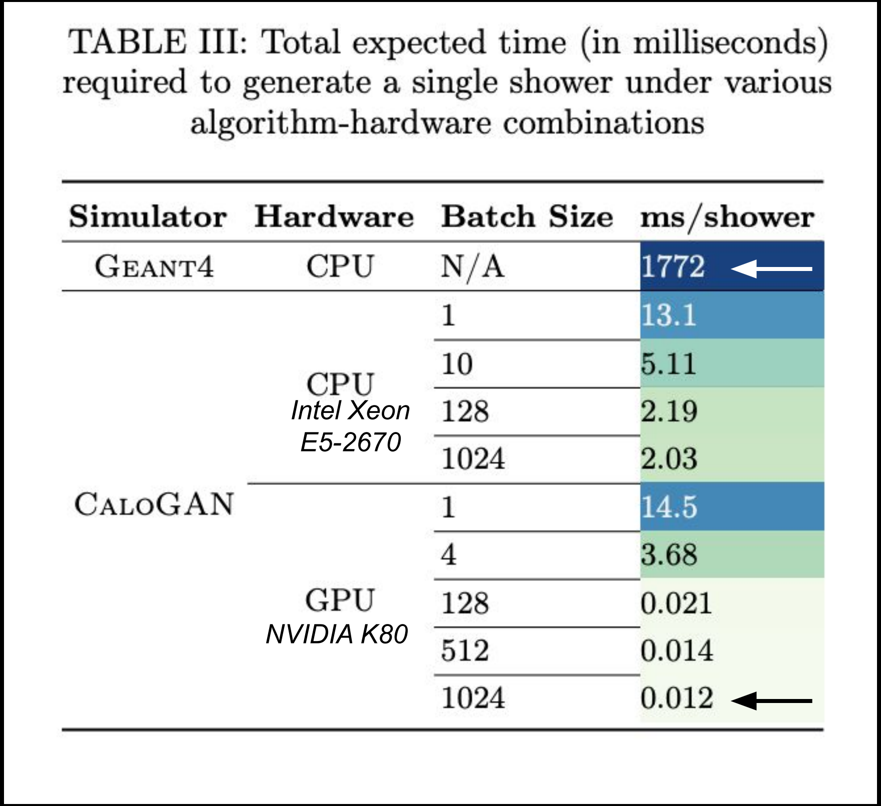


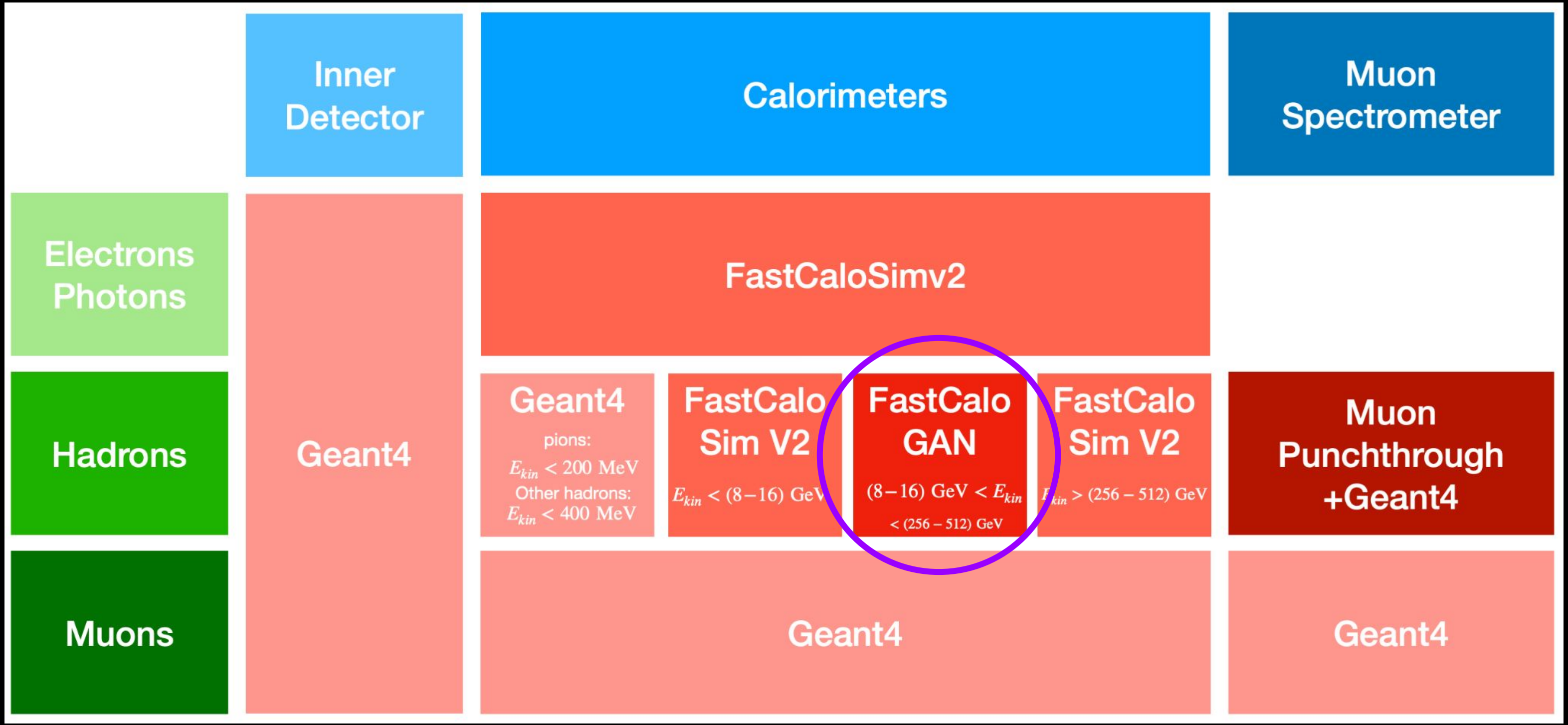
TABLE III: Total expected time (in milliseconds) required to generate a single shower under various algorithm-hardware combinations

Simulator	Hardware	Batch Size	ms/shower
GEANT4	CPU	N/A	1772 ←
CALOGAN	CPU *Intel Xeon E5-2670*	1	13.1
		10	5.11
		128	2.19
		1024	2.03
	GPU *NVIDIA K80*	1	14.5
		4	3.68
		128	0.021
		512	0.014
		1024	0.012 ←

These numbers have changed as both technologies have improved

This is simply meant to be qualitative & motivating!

Integration into real detector simulations

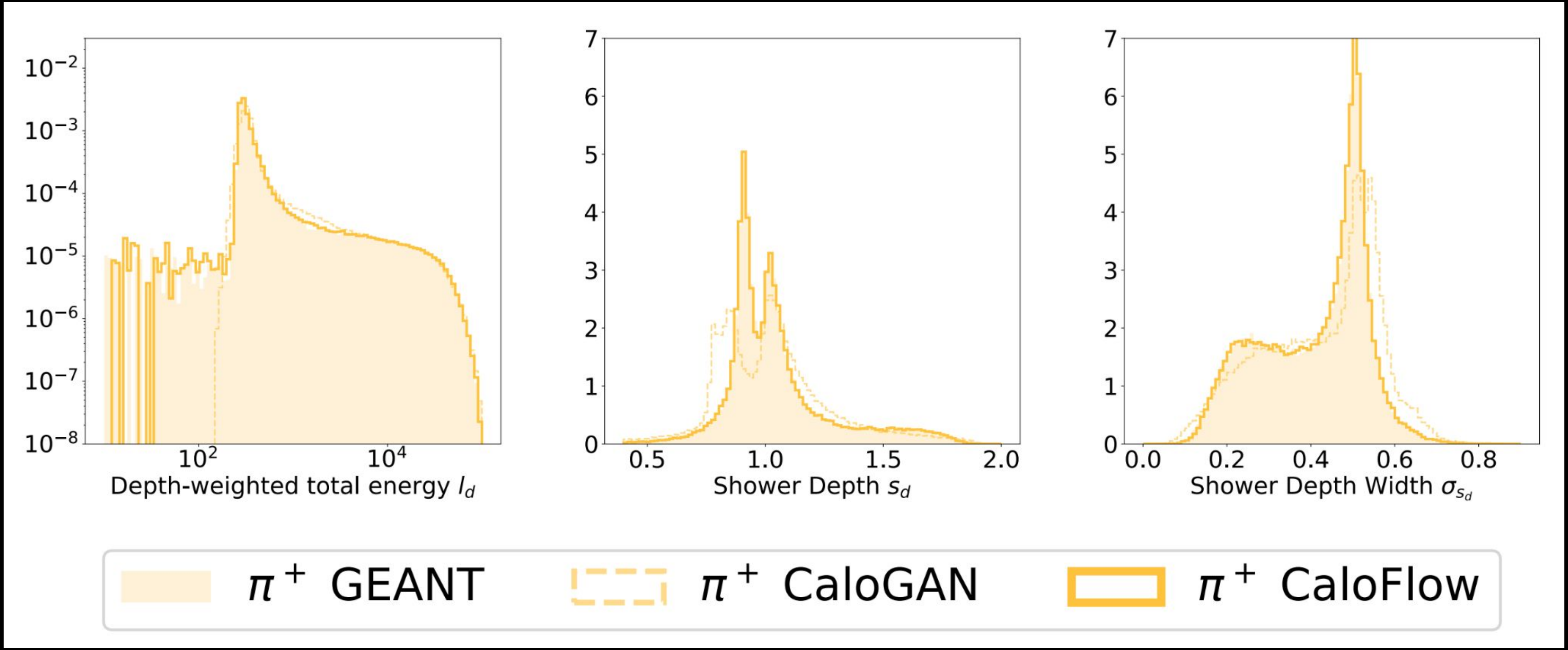


The ATLAS Collaboration fast simulation (AF3) includes a GAN at intermediate energies for pions

AtlFast3: the next generation of fast simulation in ATLAS (ATLAS Collaboration), 2109.02551 (2021)

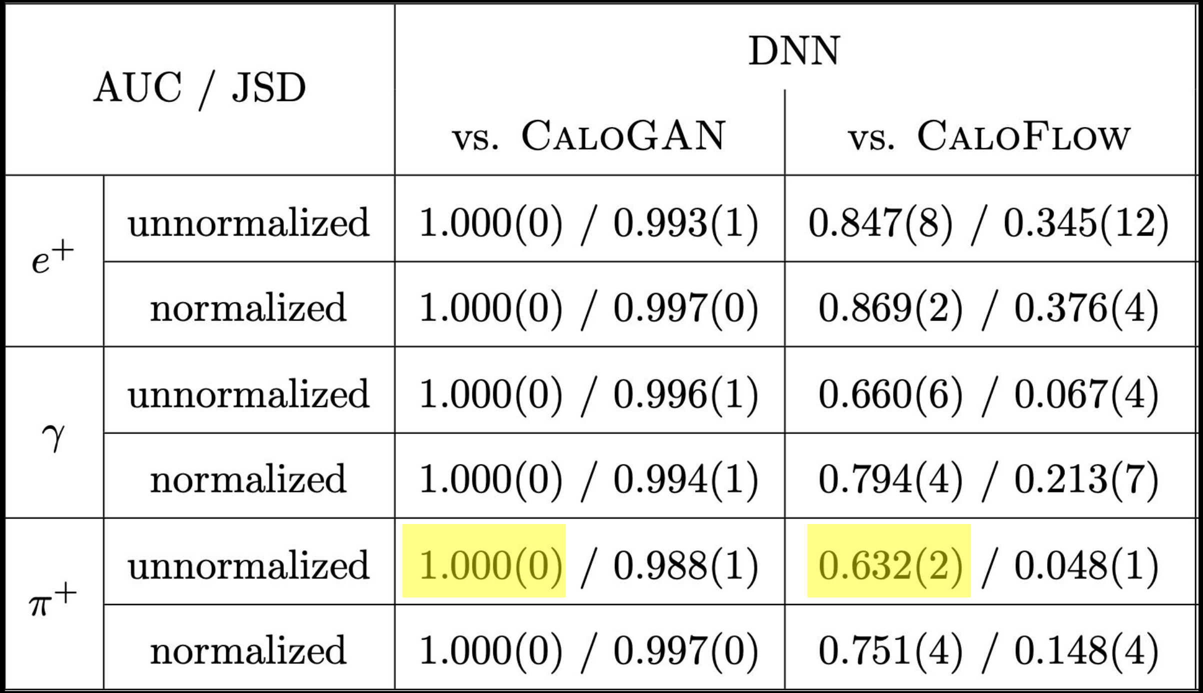
Calorimeter Showers Generation: SOTA

Generative models have gotten much better; flow models are particularly promising. Added bonus: have an explicit density.



Calorimeter Showers Generation: SOTA

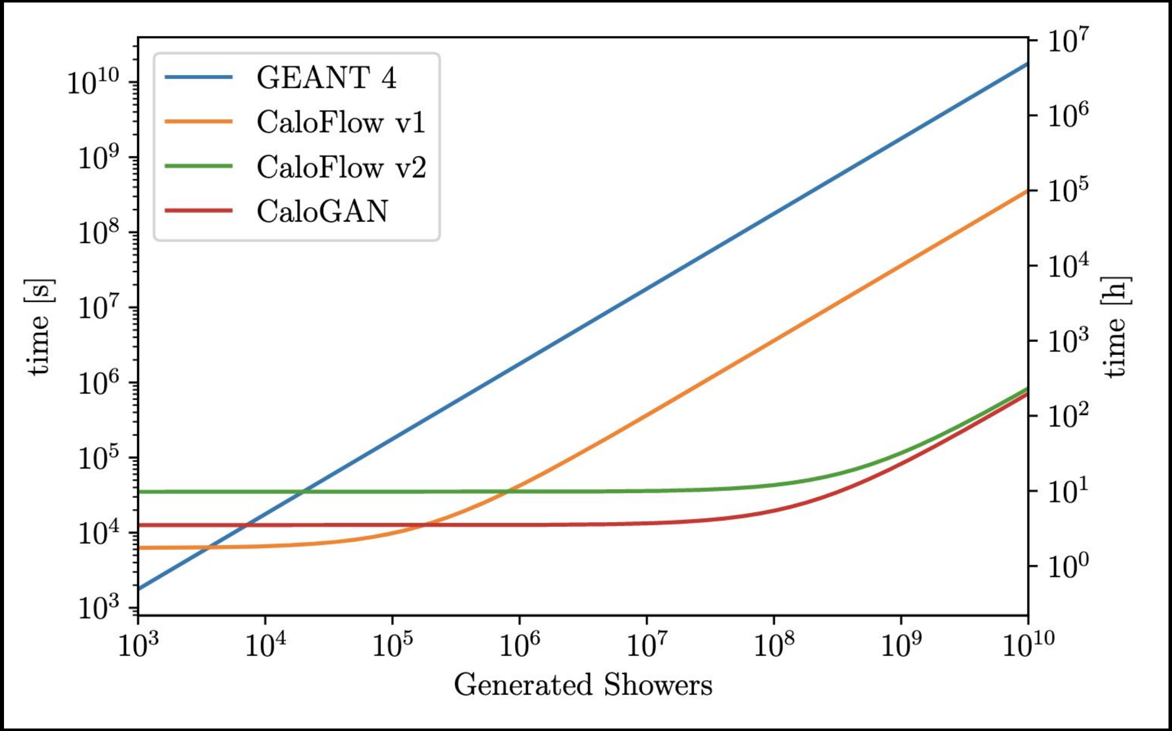
Geant4 vs. Simulations



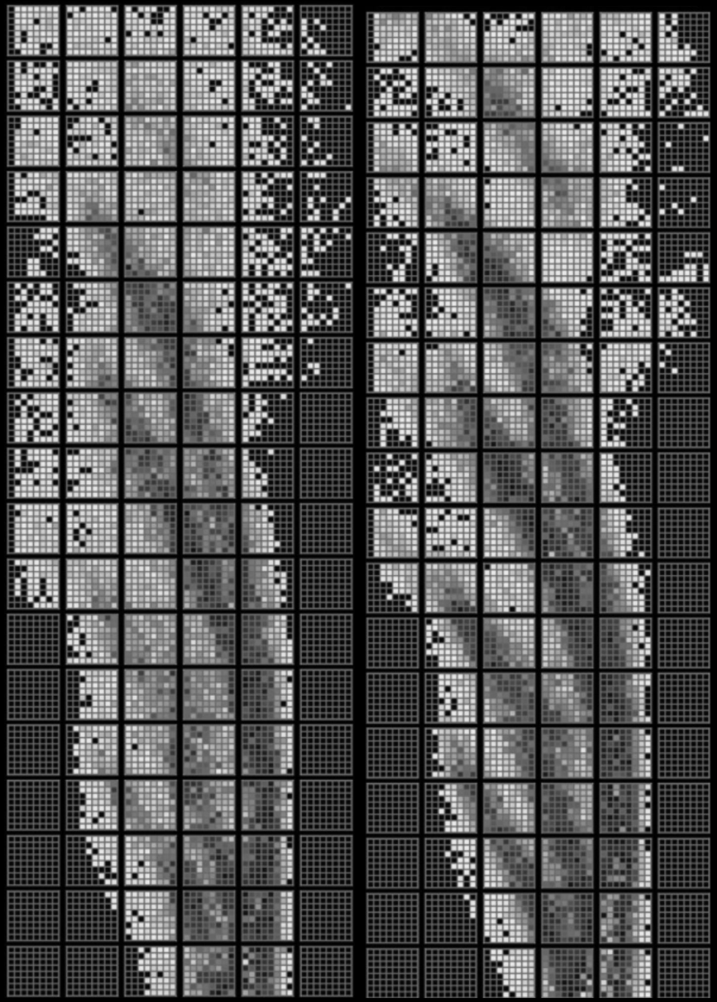
AUC / JSD		DNN vs. CaloGAN	DNN vs. CaloFlow
e^+	unnormalized	1.000(0) / 0.993(1)	0.847(8) / 0.345(12)
	normalized	1.000(0) / 0.997(0)	0.869(2) / 0.376(4)
γ	unnormalized	1.000(0) / 0.996(1)	0.660(6) / 0.067(4)
	normalized	1.000(0) / 0.994(1)	0.794(4) / 0.213(7)
π^+	unnormalized	1.000(0) / 0.988(1)	0.632(2) / 0.048(1)
	normalized	1.000(0) / 0.997(0)	0.751(4) / 0.148(4)

- AUC = 1 means easily distinguishable, AUC = 0.5 means not distinguishable
- JSD ~ 0 means labels are similarly distributed; JSD ~ 1 largest divergence

Calorimeter Showers Generation: SOTA



Learning
Event Reconstruction,
Pattern Recognition



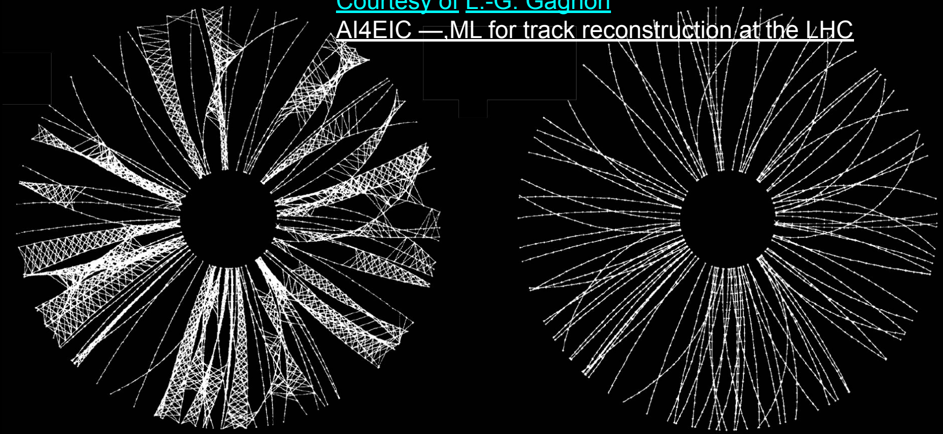
Considerations

- In general in the context of detector design is good to have algorithms that are geometry agnostic (and possibly differentiable, see later).
- Machine Learning algorithms can offer this flexibility (and are differentiable by construction)
- Some of them can easily adapt to changing conditions
- Reconstruction procedure must generalize easily if the detector design changes.
  - Otherwise we introduce a bias towards some detector design…

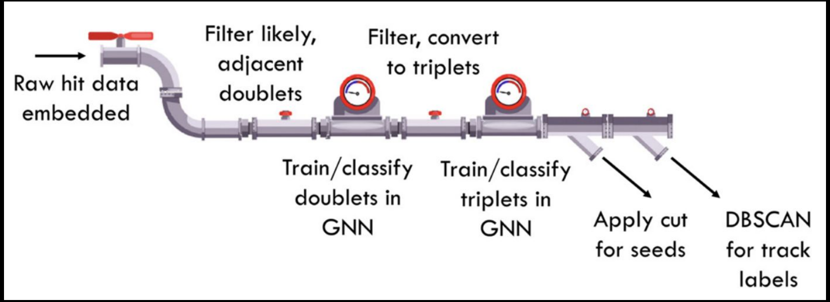
It’s not only about “speeding-up”, but also about “reducing” biases in the pipeline during its optimization.

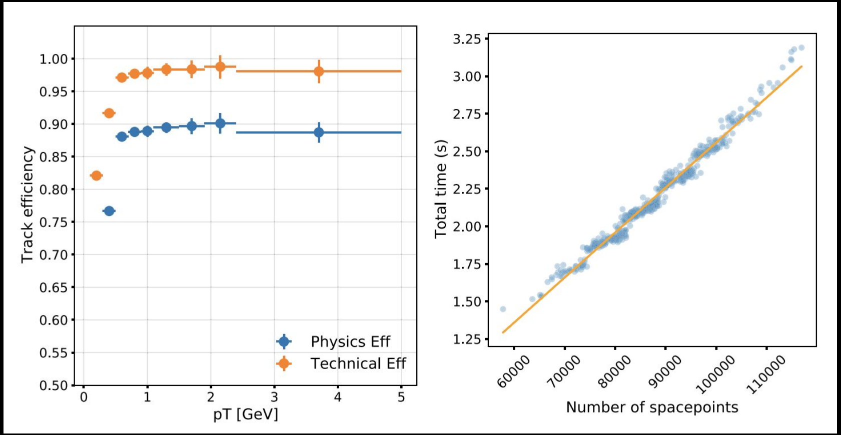
Tracking example

Courtesy of L.-G. Gagnon
AI4EIC — ML for track reconstruction at the LHC



- Kalman Filter typically boasts excellent performance for track finding and fitting; this comes at runtime cost since the time it scales quadratically or worse with # spacepoints in the event
- High-luminosity experiments like HL-LHC for example will lead to a combinatorial explosion in KF-based algorithm
- Many projects and initiatives on ML, in the last few years, e.g.,
  - HEP.trkX (ML + HPC) + TrackML data Challenge


Raw hit data embedded
Filter likely, adjacent doublets
Filter, convert to triplets
Train/classify doublets in GNN
Train/classify triplets in GNN
Apply cut for seeds
DBSCAN for track labels



(1) Construct a metric space and define a graph linking pairs of space-points together;

(2) Use dense neural network to filter edges of this graph to increase its purity and sparsity;

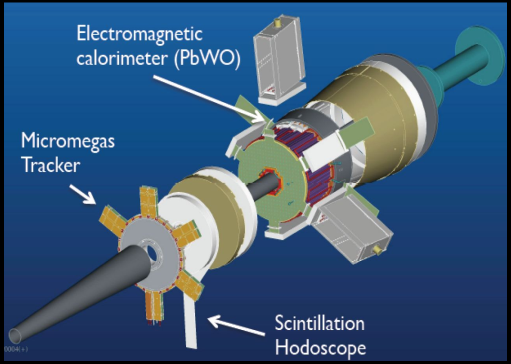
(3) Core step: use GNN ("interaction network" to classify edges as being part of a true track or not;

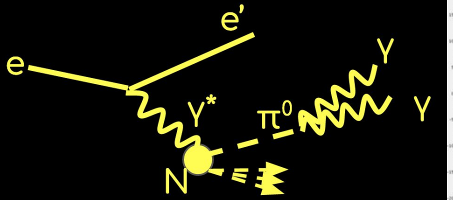
(4) Post-processing: walk the resulting graph to build tracks from which parameters can be estimated

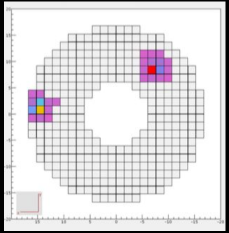
Performance of a geometric deep learning pipeline for HL-LHC particle tracking, EPJC 81, 876 (2021)

Clustering example

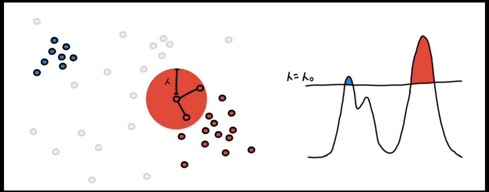
- Hierarchical clustering VS traditional clustering of energy deposited by photons; AI robust against variations in experimental conditions
- Example of Streaming Readout tests in CLAS12 Forward Tagger (with uncalibrated data in SRO)



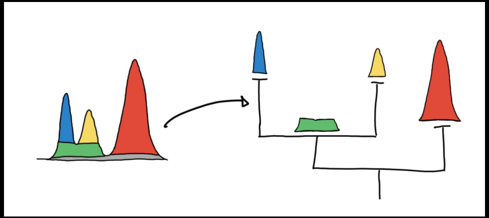




Core distance (defined by a required # of neighbors) as estimate of density

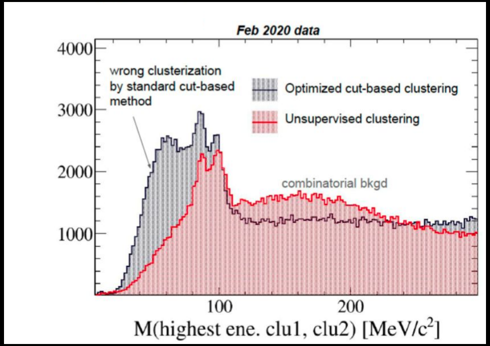


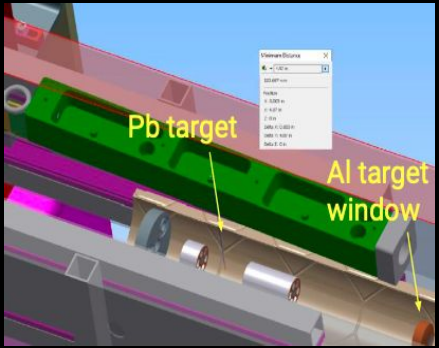
Points have to be in a high density region and close to each other ("mutual reachability")



The area of the regions is the measure of "persistence".

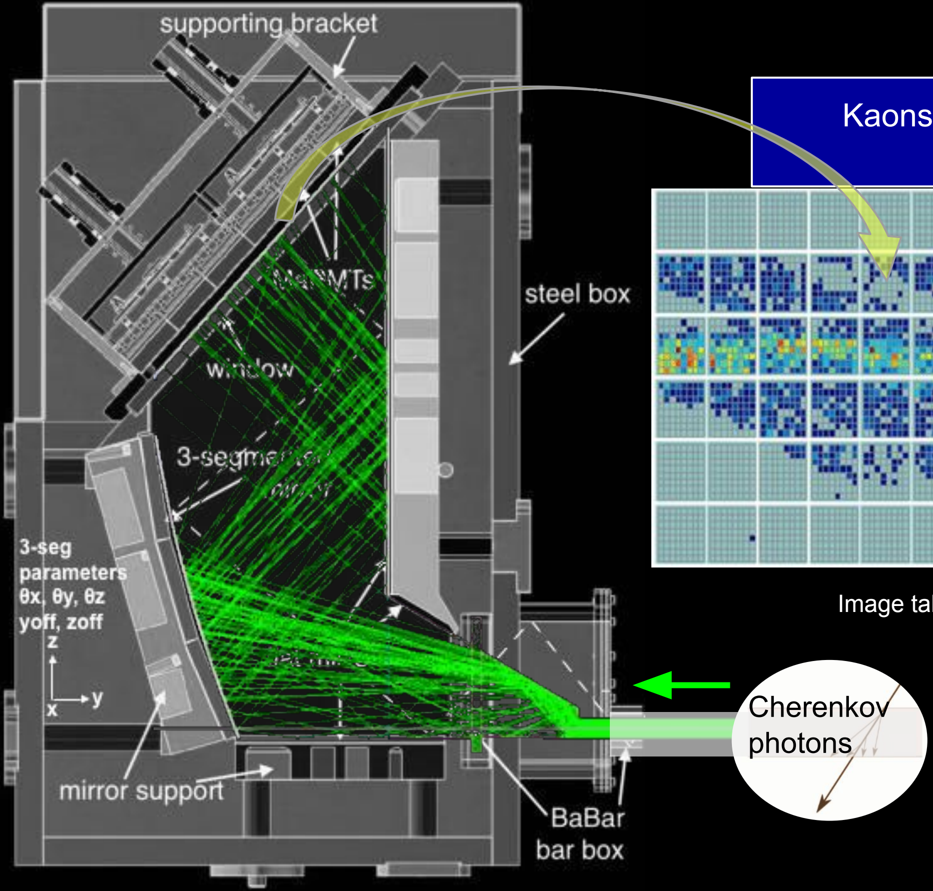
Maximize the persistence of the clusters under the constraint that they do not overlap.





F. Ameli, CF, et al. ,Streaming readout for next generation electron scattering experiments, arXiv:2202.03085, 2022

Pattern Recognition: Imaging Cherenkov Detectors example



Kaons @ 4 GeV/c for different polar and azimuthal angle

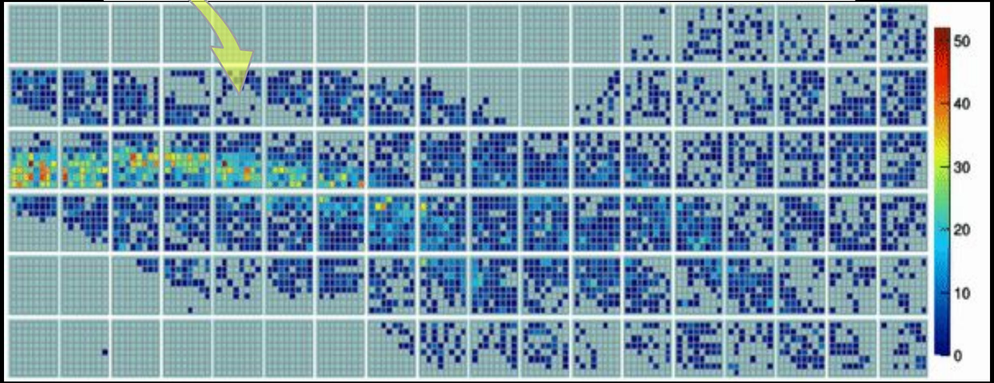
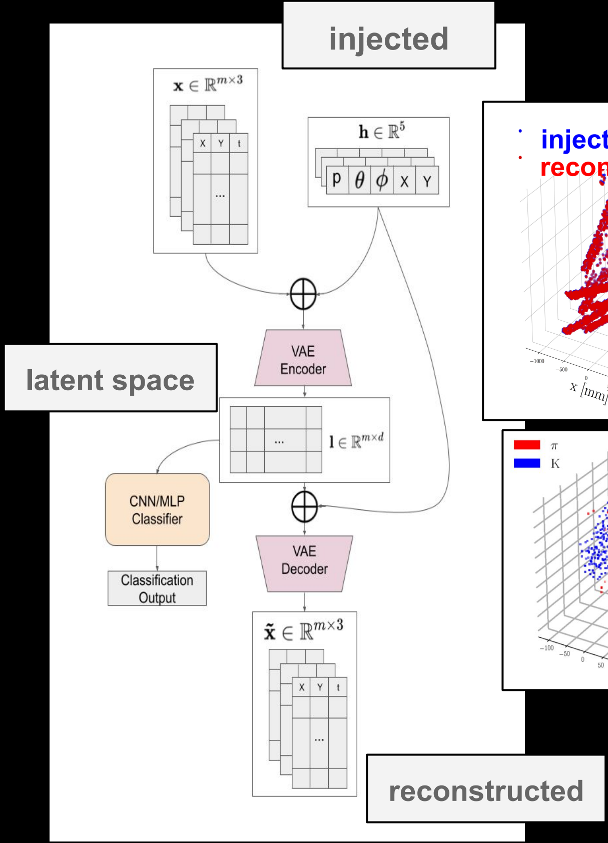


Image taken from https://web-docs.gsi.de/~rdzhigad/

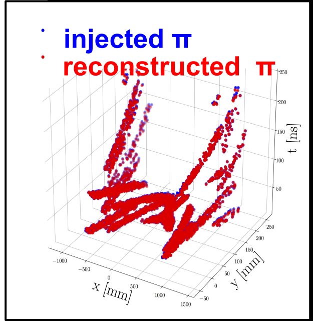
Dependence on charged particle kinematics $(p,(\theta,\varphi)^*,X,Y)$

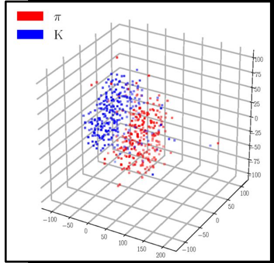
1

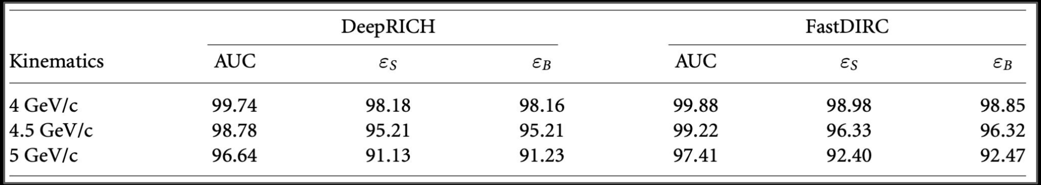
Pattern Recognition: Imaging Cherenkov Detectors example



injected
$\mathbf{x} \in \mathbb{R}^{m\times 3}$
X Y t
...
$\mathbf{h} \in \mathbb{R}^5$
p θ ϕ X Y
VAE Encoder
latent space
...
$\mathbf{l} \in \mathbb{R}^{m\times d}$
CNN/MLP Classifier
Classification Output
VAE Decoder
$\tilde{\mathbf{x}} \in \mathbb{R}^{m\times 3}$
X Y t
...
reconstructed

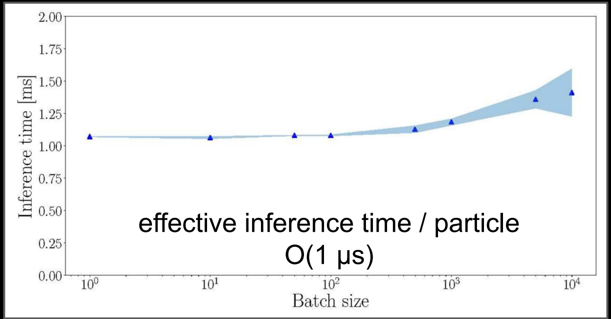






	DeepRICH			FastDIRC		
Kinematics	AUC	ε_S	ε_B	AUC	ε_S	ε_B
4 GeV/c	99.74	98.18	98.16	99.88	98.98	98.85
4.5 GeV/c	98.78	95.21	95.21	99.22	96.33	96.32
5 GeV/c	96.64	91.13	91.23	97.41	92.40	92.47

AUC(DeepRICH) ≳ 0.99 AUC(FastDIRC)



Combines both great reconstruction performance and computing time:

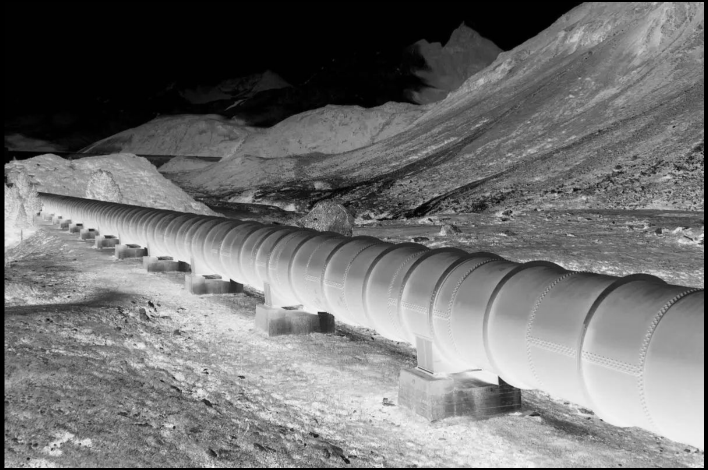
- ~ same accuracy of established method with best reconstruction performance (FastDIRC[2])
- O(1μs) on GPU VS O(1ms) on CPU as compared to fastest established method (geometric w look-up table)

[1] C. Fanelli, J. Pomponi, DeepRICH: Learning Deeply Cherenkov Detectors, *Mach. Learn.: Sci. Technol.* 1 015010 (2020)
[2] J. Hardin, M. Williams. FastDIRC: a fast Monte Carlo and reconstruction algorithm for DIRC detectors. *JINST* 11.10 (2016): P10007.

Remarks

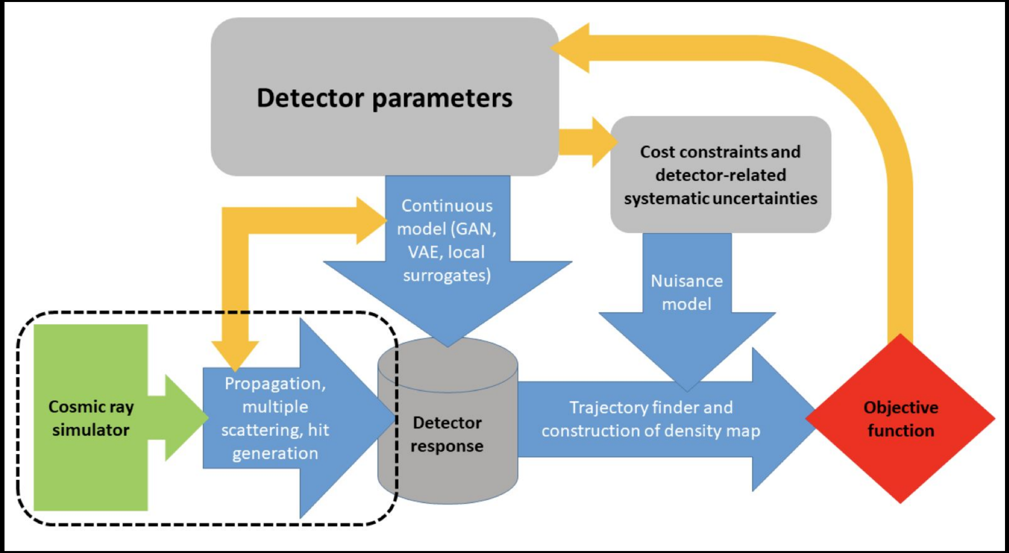
- Full simulation can involve complete physics responses and digitization, including but not limited to Cherenkov photon production and electron cascades.
- Accelerated simulations aims at speeding up full simulations
- Faster reconstruction allows for an additional speed-up in reconstructing and analysing high-level observables
- The exploration of the design space can be driven by AI as discussed (with multiple driving criteria/objectives).
- What happens when you change your design point to the AI/ML in the simulation and reconstruction steps? Different scenarios/options:
    - Training was done on multiple design points (difficult); no change in the sim/reco
    - Re-training needed — this may be time consuming and lose all advantage of accelerated steps
    - Some algorithms may undergo a simple “retuning“: e.g., unsupervised approaches
    - etc

Combining all together: End-to-end Optimization Pipelines



MODE-Machine Learning Optimized Design of Experiments

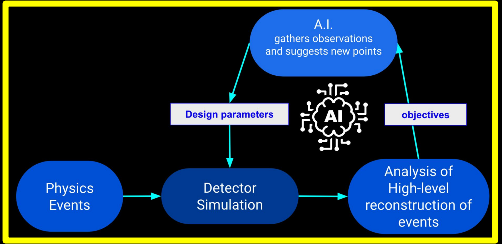
- Detectors design with AI is gaining a lot of interest.
- MODE is a recently formed collaboration of physicists and computer scientists who target the use of differentiable programming in design optimization of detectors for particle physics applications A. G. Baydin et al. Nuclear Physics News 31.1 (Mar 30, 2021): 25-28.
- Ambitious project: develop a modular, customizable, and scalable, fully differentiable pipeline for the end-to-end optimization of articulated objective functions that model in full the true goals of experimental particle physics endeavours, to ensure optimal detector performance, analysis potential, and cost-effectiveness.


Detector parameters
Cost constraints and detector-related systematic uncertainties
Continuous model (GAN, VAE, local surrogates)
Nuisance model
Cosmic ray simulator
Propagation, multiple scattering, hit generation
Detector response
Trajectory finder and construction of density map
Objective function

Conceptual layout of an optimization pipeline taken from a muon radiography apparatus.

An end to end optimization requires modeling of simulations. Requires collect reference data to train the surrogate models ML implementations.

Design Optimization: Problem Description



A.I.
gathers observations and suggests new points
Design parameters
objectives
AI
Physics Events
Detector Simulation
Analysis of High-level reconstruction of events

SENSOR READOUT

PHYSICS PROCESSES

DETECTOR PHYSICAL PROPERTIES AND GEOMETRY

1 $$p(z|x,\theta)$$

PDF OF DETECTION GIVEN PHYSICS, DETECTOR

$z \sim p(z|x,\theta)$ $x \sim f(x)$ (PDF PHYSICS)

HIGH-LEVEL FEATURES

NUISANCE PARAMETERS

2 $$\zeta(\theta) = R[z,\theta,\nu(\theta)]$$

RECONSTRUCTION MODEL

3 $$s = A[\zeta(\theta)]$$

DATA ANALYSIS: CLASSIFIER OR REGRESSOR (NN)

The design problem becomes:

4

PERFORMANCE

COST

$$\hat{\theta} = arg\ min_\theta \int L[A(\zeta), c(\theta)] p(z|x,\theta) f(x) dx dz$$

LOSS FUNCTION

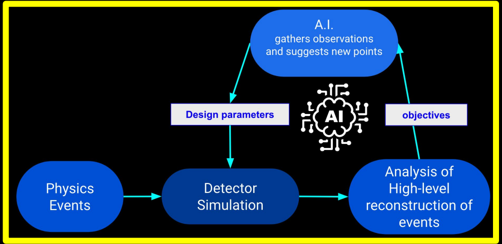
Since typically the PDF p(z|x,θ) is not available in closed form we use forward simulations to sample from:

5

$$\Rightarrow \hat{\theta}_a = arg\ min_\theta \frac{1}{n} \sum_{i=1}^{n} L[A(R(z_i)), c(\theta)]$$

approx

Design Optimization: Problem Description


A.I. gathers observations and suggests new points
Design parameters
AI
objectives
Physics Events
Detector Simulation
Analysis of High-level reconstruction of events

6 Useful to approximate the non-differentiable stochastic simulator with a local surrogate model that depends on a parameter y for the stochastic variation of the approximated distribution: [arXiv:2002.04632]

$$z = S(y, x, \theta)$$

SURROGATE

The design problem becomes:

4

PERFORMANCE COST

$$\hat{\theta} = arg\ min_{\theta} \int L[A(\zeta), c(\theta)] p(z|x,\theta) f(x) dx dz$$

LOSS FUNCTION

Since typically the PDF p(z|x,θ) is not available in closed form we use forward simulations to sample from:

5

$$\hat{\theta}_a = arg\ min_{\theta} \frac{1}{n} \sum_{i=1}^{n} L[A(R(z_i)), c(\theta)]$$

approx

7 Descend the minimum of the approximated loss by following the surrogate gradient

$$\boxed{\nabla_{\theta}(L(\hat{z})) = \frac{1}{n} \sum_{i=1}^{n} \nabla_{\theta} L[A(R(S(y_i, x_i, \theta))), c(\theta)]}$$

GLOBAL OPTIMIZATION TASK

Auto-differentiation

- Automatic-differentiation, aka algorithmic differentiation or autodiff
  - *“a set of techniques to evaluate the derivative of a function specified by a computer program, which, no matter how complicated, executes a sequence of elementary operations, and a repeated usage of chain rule.”*
- Differentiable programming captures the essence of DL practice; differentiable code is realized to solve various tasks; their optimization is done via gradient-based optimization of an objective based on training data
- Neural networks are member of the family of differentiable programs, in that they can be seen a series of non-linear transformations
- There are two ways to make a simulation differentiable:
  - Using AD directly in the simulation code
  - Using DL to produce a differentiable surrogate model

[1] MODE, Toward the End-to-End Optimization of Particle Physics Instruments with Differentiable Programming: arXiv:2203.13818, 2022
[2] Y. LeCun. Facebook post on differentiable programming, 2018 https://facebook.com/yann.lecun/posts/10155003011462143

Surrogate Models

- With a surrogate based on a deep-learning architecture, AD is immediately available within the machine learning framework used to train the surrogate
- Note that the surrogate can be differentiable even if the original function is not
  - Also for this reason is important to come up with a pipeline that can address systematic uncertainties [2,3,4]
- Evaluation of the surrogate (and its derivatives) is orders of magnitude faster than the computation of the "true" model (due to vectorization and hardware parallelism of GPUs and TPUs available in ML libraries)
- Nonetheless training the surrogate requires a substantial number of evaluations of the original function
- And again poorly trained surrogates can introduce bias in the analysis…

[1] MODE, Toward the End-to-End Optimization of Particle Physics Instruments with Differentiable Programming: arXiv:2203.13818, 2022
[2] P. de Castro and T. Dorigo, INFERNO: Inference-Aware Neural Optimisation. Comput. Phys. Commun. 244:170-179, 2019
[3] A. Ghosh and B. Nachman, and Daniel Whiteson, Phys. Rev. D **104**, 056026, 2021
[4] AI4EIC, topic-oriented meeting on uncertainty quantification https://indico.bnl.gov/event/16073/

Modeling the Cost of Detectors

- It is possible to compute the effect of construction costs on the loss function in two main steps:
  - Local cost parameters φ are specific to the technology used (e.g., active components material, light transport techniques etc)
  - Global cost c(φ,θ) can be expressed as a function of local cost parameters φ and a set of parameters θ describing the overall detector concept, like number and size of detector modules, their positions, etc.

$$L_{cost} = c(\theta, \phi)$$

- More complicated to factor in the parametrization costs like labor, etc.

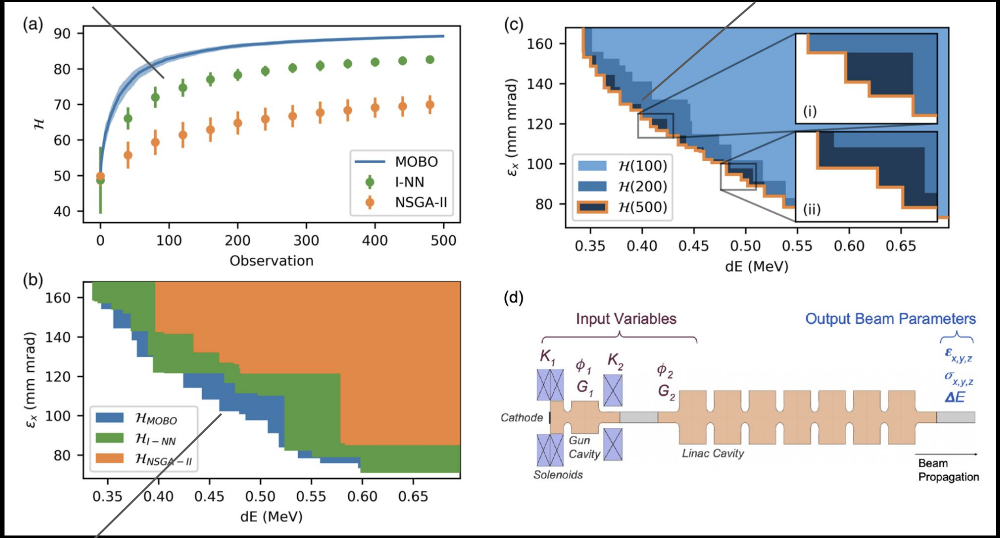
Example use cases: Accelerator Optimization

- The challenge and interest in optimizing these systems both during the design and the operation phases increase as we push the energy and intensity frontiers of beam physics. The design process and online optimization are segmented at present: this produces sub-optimal results… [1]

- MOBO has been successfully utilized for online accelerator tuning with 7D objective space for the Argonne Wakefield Accelerator [2]

MOBO is more efficient

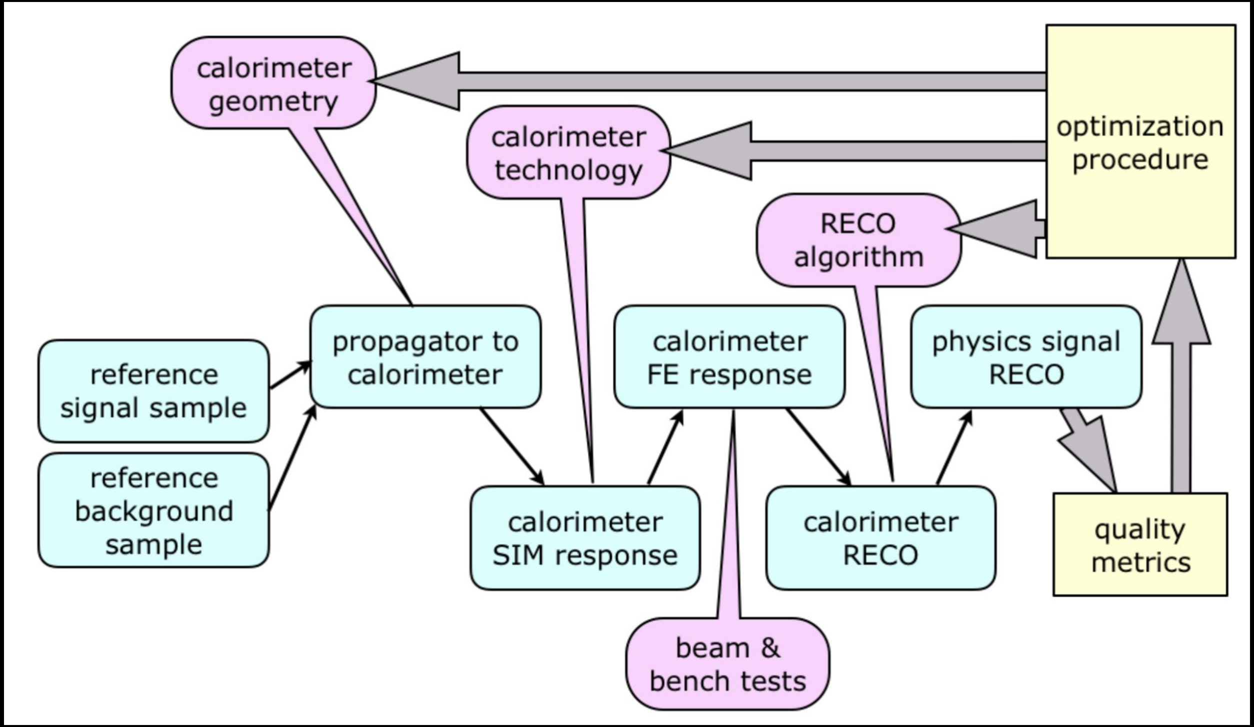
More evaluations brings refinement



MOBO font has higher resolution

[1] MODE, Toward the End-to-End Optimization of Particle Physics Instruments with Differentiable Programming: arXiv:2203.13818, 2022
[2] R. Roussel, A. Hanukkah and A. Edelen. MOBO for online accelerator tuning, Phys. Rev. Accel. Beams, 24:062801, 2021

Example use cases: Calorimeter Optimization

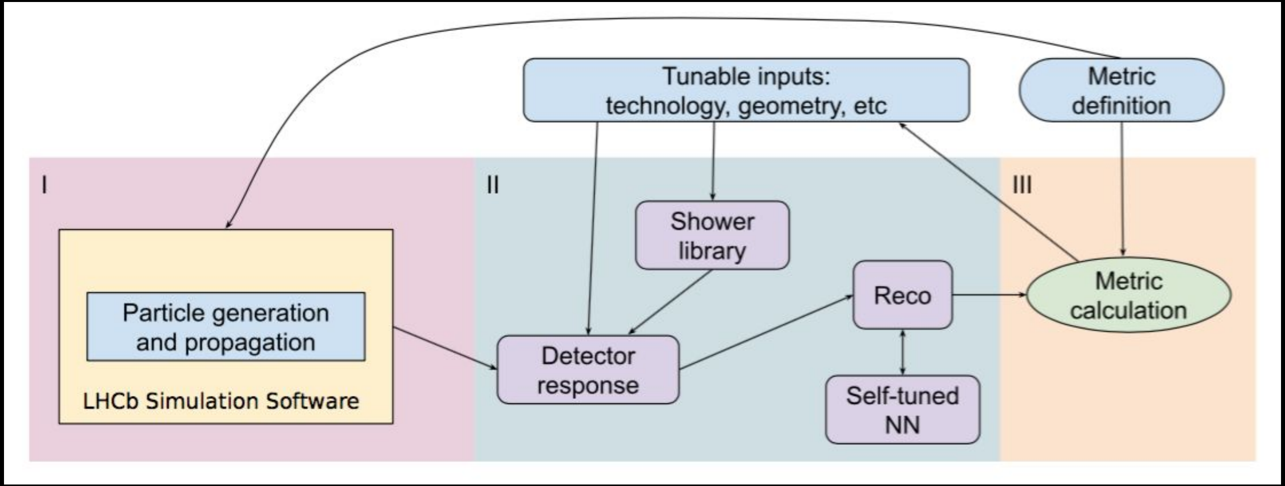


- Fine-tuning of the individual blocks important to propagate the properties of the configuration under study
- Surrogate models may be built and trained on labelled data using regular ML approaches
- Speed up model building for different pipeline steps.
- Necessity of fine-tuning reconstruction algorithms for every new calorimeter technology and geometry

The big slowdown factor for running an optimization workflow is the need to fine-tune the reconstruction algorithm for every new calorimeter technology (R&D) and geometry (design) configuration.
ML may help tuning the reconstruction in an "automatic" way (regression).

ML-assisted approach to calorimeter R&D

- In [1] ML can substitute the most computationally intensive steps while retaining the GEANT4 accuracy to details.
- Focus on the Phase II Upgrade of the LHCb Calorimeter under the requirements on operation at high luminosity. The optimization pipeline looks like the following:



ML (decision trees) intervenes in tuning the reconstruction.

(optimizers can be used to tune hyper-parameters).

[1] A. Boldyrev et al (Yandex), arXiv:2005.07700v1 [physics.ins-det] 2020

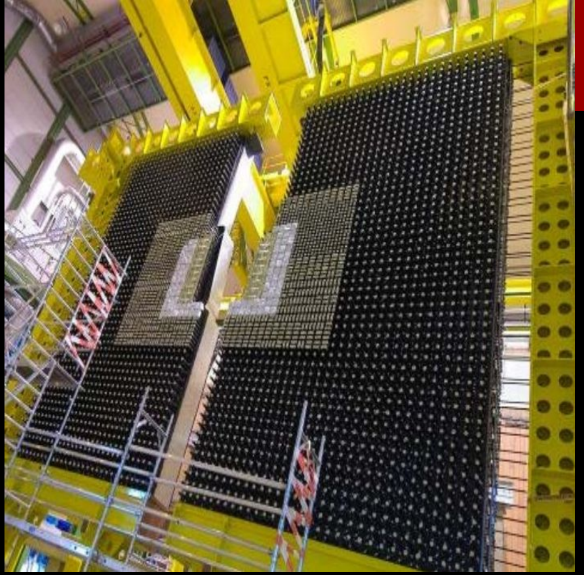
ECAL LHCb

$$\frac{\sigma_E}{E} = \frac{(8 \div 10)\%}{\sqrt{E(\mathrm{GeV})}} \oplus 0.9\%$$

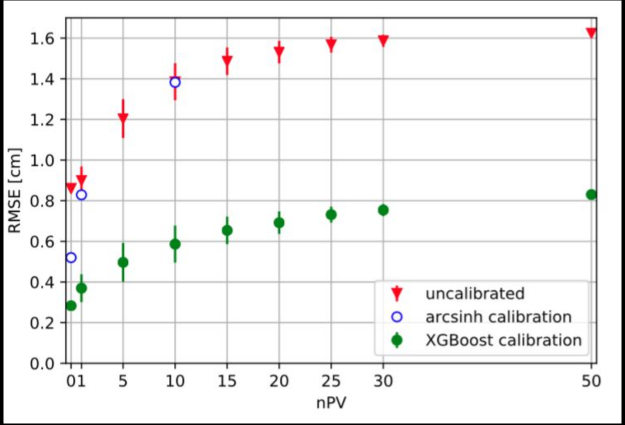
4 mm thick scintillator tiles and 2 mm thick lead plates (Shashlik technology) ~25 X_0 (1.1 λ_I);
Moliere radius ~ 36 mm;

Modules 121.2 x 121.2 mm^2, 66 Pb +67 scintillator tiles;

Segmentation: 3 zones 3 module types, Inner (9 cells per module), Middle (4), Outer (1).
Total of 3312 modules, 6016 cells, (7.7 x 6.3) m^2, ~100 tons.



- Take advantage of segmentation / modularity (see discussion on characterization of the detector design problem) and create a Geant4 standalone simulation for 30x30 cells of size 20.2x20.2 mm^2 which can be rearranged in the inner, middle and outer ECAL modules.
- Used a signal sample $B^0_s \to J/\psi(\mu^+\mu^-)\pi^0(\gamma\gamma)$ and the LHCb minimum bias sample as background.
- Studies as a function of pile-up (PU) and number of primary vertices (nPV).
- Calibration (spatial and energy) optimized using XGBoost and BO for fine-tuning of parameters at the simulation and reconstruction steps.



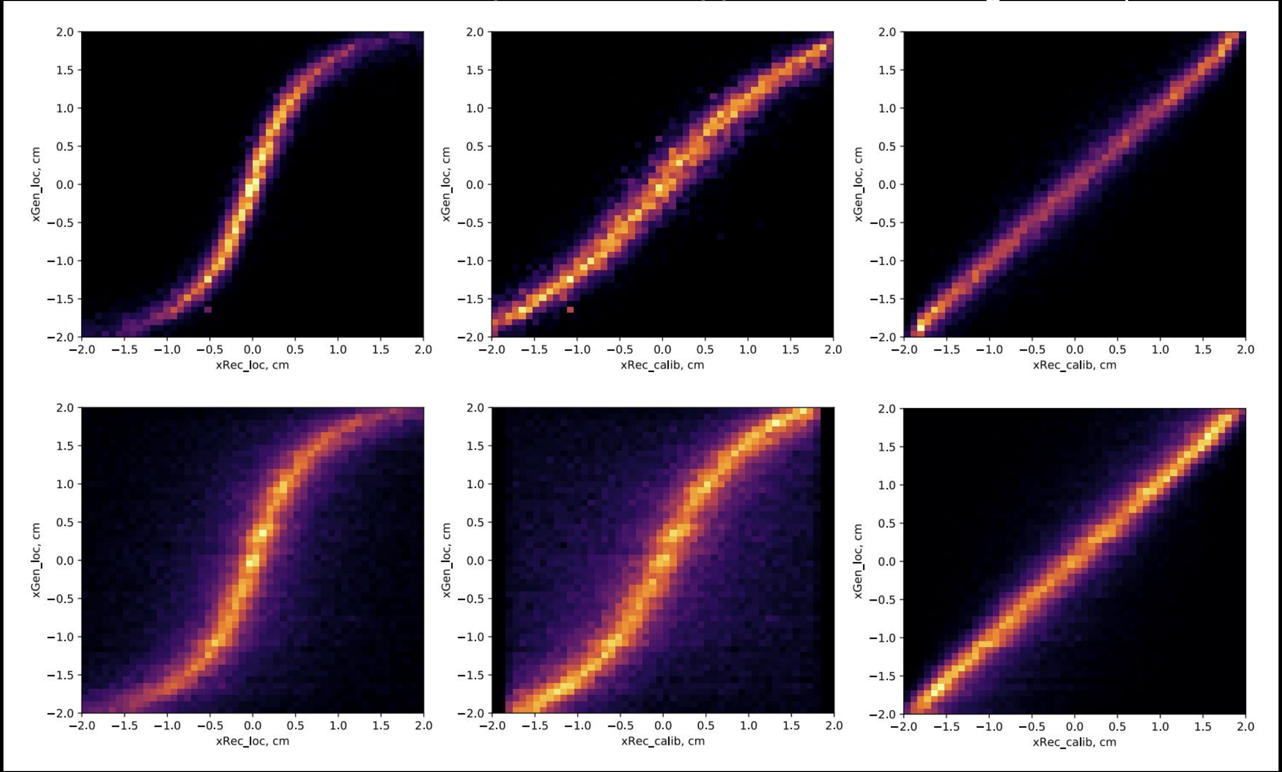
A. Boldyrev et al (Yandex), arXiv:2005.07700v1 [physics.ins-det] 2020

ECAL LHCb

uncalibrated | parametric approach | ML-regressor (XGBoost)

Signal (no background)

nPV/event = 10



ML-based reconstruction of the calorimeter cluster position provides good spatial resolution without a priori knowledge about the spatial properties of the calorimeter under study

A. Boldyrev et al (Yandex), arXiv:2005.07700v1 [physics.ins-det] 2020

Other Examples

- EM Calorimeter of a Muon Collider
- Optimization of the MUoNE Detector
- Searches for Milli-charged Particles
- Astro-particle Physics and Neutrino Experiments
- High-Energy Gamma-Ray Astronomy
- Interferometric Gravitational-Wave Detectors
- Cosmic-Ray Muon Imaging
- Portable Modular Detectors for Flexible Muography
- Proton Computed Tomography
- Low-Energy Particle Physics

MODE, Toward the End-to-End Optimization of Particle Physics Instruments with Differentiable Programming: arXiv:2203.13818, 2022

System architecture and requirements

- Support variety of simulation packages (GEANT4, Pythia, etc)
- Flexibility
- Distributed execution and scalability
- Interruption-friendly execution
- User ask independence and result sharing

"Excerpts" from MODE workshop

Main SW components:

- Structural storage (database) management
- Volume storage management
- Cloud compute management
- Simulation package connection and software environment configuration
- User task management
- Optimization monitoring/benchmarking interface
- Black-box optimization runtime
- Differentiable optimization runtime

[1] MODE, Toward the End-to-End Optimization of Particle Physics Instruments with Differentiable Programming: a White Paper arXiv:2203.13818

Final Remarks

- Moving towards Physics-driven optimization pipelines: i.e., from FoM encoding the detector intrinsic response to physics observables and implementation of full analysis pipeline — need identify main processes for the experiment
- For each design point, we are utilizing the same analysis pipelines and validation procedures that everyone else has used in the past to design detectors
- Differently from the past, for the exploration of the design space, instead of relying on manual optimizations done by a team of PhD students, we can do this more efficiently using modern AI-assisted pipelines (eventually developed and maintained by PhD students :)
- Next steps: physics-driven design, where FoM are physics observables measured by different EIC detector design configurations

Sociology

- Utility of new approaches to design increases with complexity of experiment
- Possibility of concrete realization (embracing/deploying these new technologies), inversely related to the complexity of the experiment / collaboration

Workshops/Schools on Detector Design

- AI4EIC, https://eic.ai/workshops, https://eic.ai/events

  Special Meeting on Design: July 20

  Next workshop: October 10-14, 2022
- MODE, https://mode-collaboration.github.io/workshop/index.html

  Next: September 12-16, 2022

“Review” Articles on Detector Design/Sim.

- T. Dorigo et al., Toward the End-to-End Optimization of Particle Physics Instruments with Differentiable Programming: a White Paper, arXiv:2203.13818
- A. Adelmann et al., New directions for surrogate models and differentiable programming for HEP detector simulation, arXiv:2203.08806
- C. Fanelli, Design of Detectors at the Electron Ion Collider with AI, arXiv:2203.04530

Conclusions

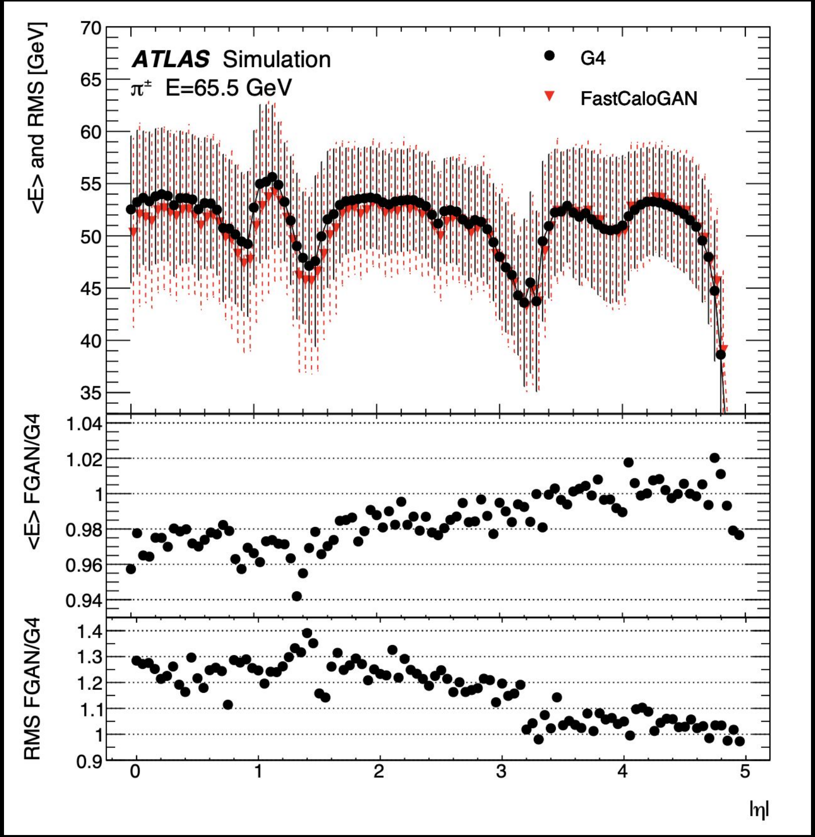
- These lectures covered a set of tools/technologies that can empower our community to tackle larger scale and more complex detector design problems: [1]
    - AI-assisted optimization strategies and ML-based surrogate-models (with possibility of autodiff)
    - The "*hand and intuition*" of the expert cannot be substituted at present, but these automated pipelines cannot be any longer ignored as valuable tools to assist the design problems
    - In the (near) future their utilization can bring to "*groundbreaking solutions to century-old problems*"
- The optimization of an entire detector or accelerator is a daunting task that is probably beyond present-day capabilities; nonetheless we already saw successful applications to portion of these machines and their automated control.
- AI can indeed assist the design and R&D of complex experimental systems by providing more efficient design (considering multiple objectives) and optimizing the computing budget needed to achieve that. EIC is one of the first experiments to be designed with the support of AI. [2]
- One of the conclusions from the DOE Town Halls on AI for Science on 2019 was that "*AI techniques that can optimize the design of complex, large-scale experiments have the potential to revolutionize the way experimental nuclear physics is currently done*". [3]

[1] MODE, Toward the End-to-End Optimization of Particle Physics Instruments with Differentiable Programming: a White Paper arXiv:2203.13818
[2] C. Fanelli et al (ECCE consortium) AI-assisted Optimization of the ECCE Tracking System at the Electron Ion Collider, arXiv:2205.09185
[3] R. Stevens et al., AI for Science: Report on the Department of Energy (DOE) Town Halls on Artificial Intelligence (AI) for Science

AI-optimization strategies are much more than just fine-tuning the detector design…

Backup

Integration into real detector simulations



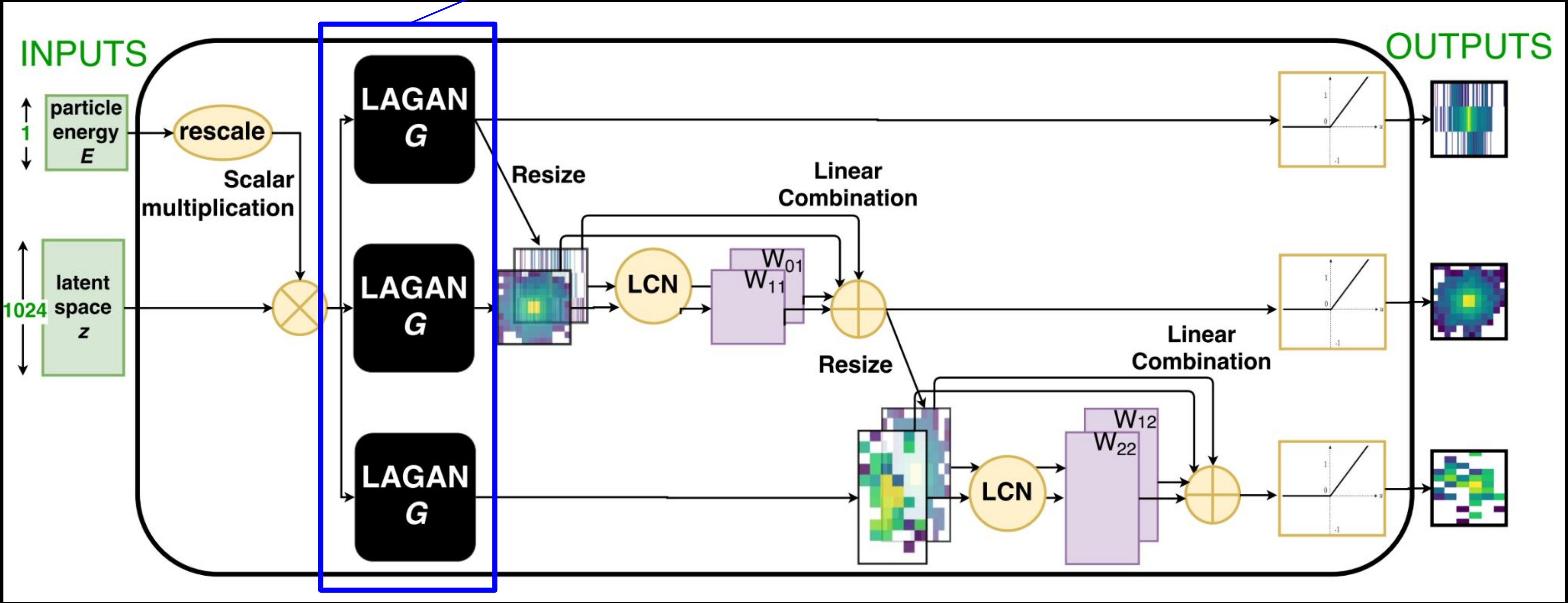
The GAN architecture is relatively simple, but it is able to match the energy scale and resolution well.

There is one GAN per η slice

CaloGAN

One image per calo layer

One network per particle type;
Input particle energy



LA:
Locally Aware,
Like a CNN

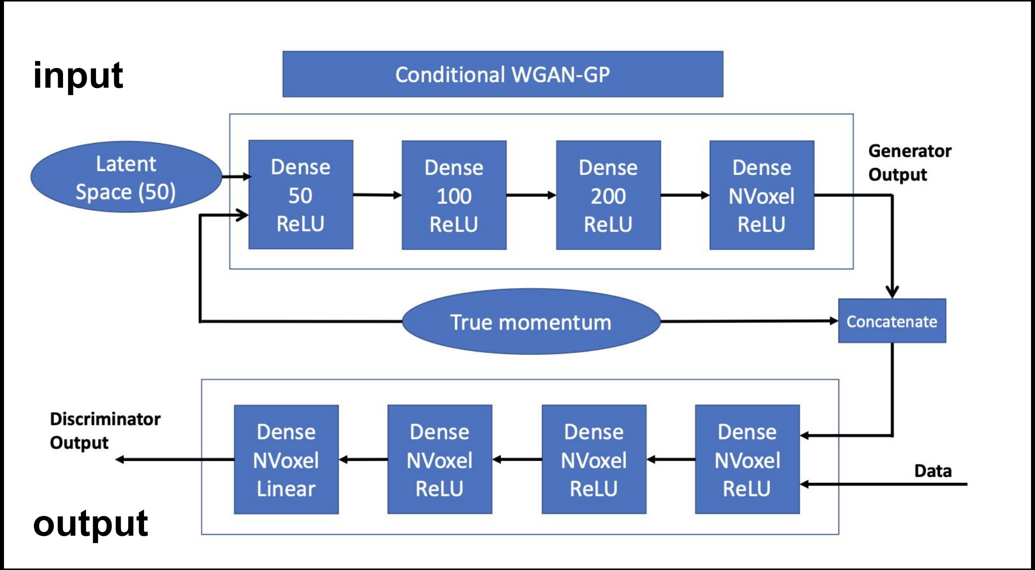
Use layer i as input to layer i+1

ReLU to encourage sparsity

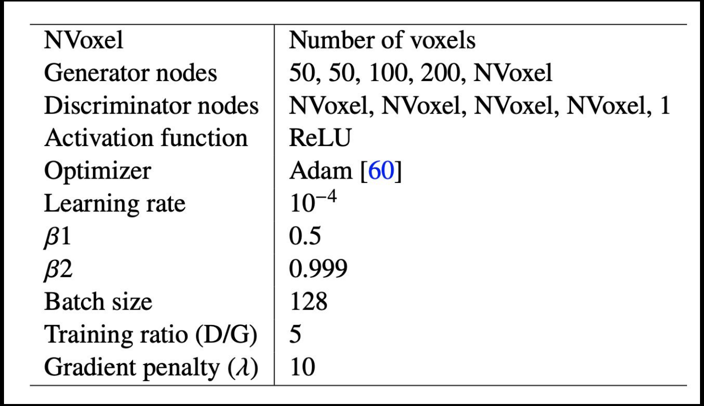
M. Paganini, L. de Oliveira, B. Nachman, 1705.02355, 1712.10321 (2017)

FastCaloGAN

ATL-SOFT-PUB-2018-001



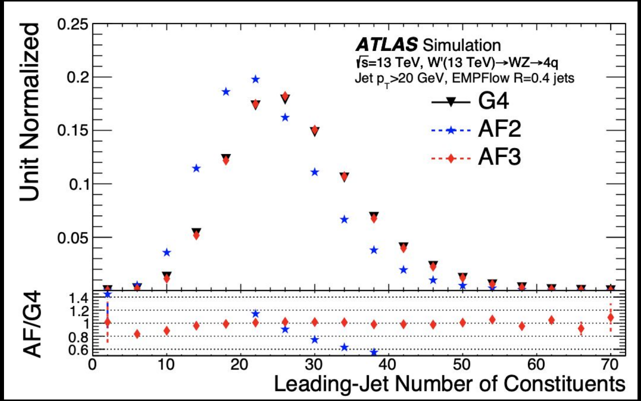
A Wasserstein GAN with gradient penalty is used

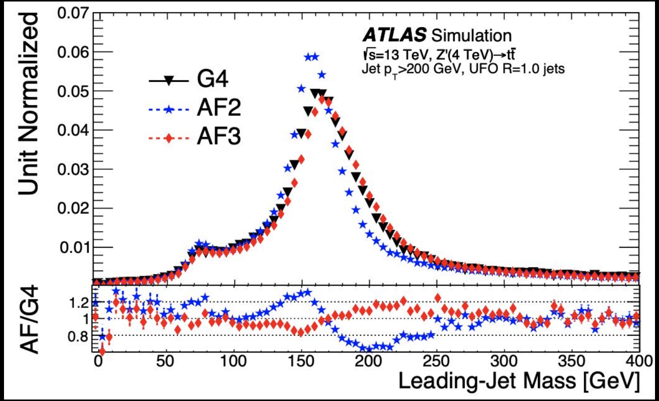


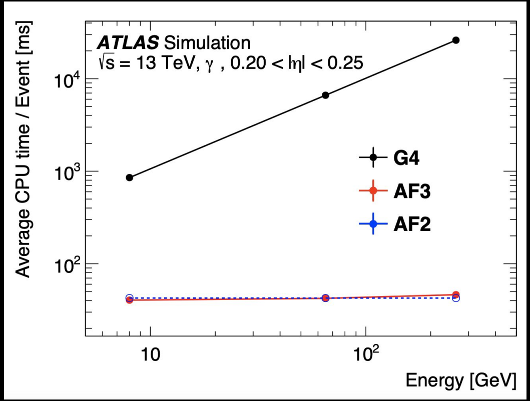
NVoxel	Number of voxels
Generator nodes	50, 50, 100, 200, NVoxel
Discriminator nodes	NVoxel, NVoxel, NVoxel, NVoxel, 1
Activation function	ReLU
Optimizer	Adam [60]
Learning rate	10^{-4}
$\beta 1$	0.5
$\beta 2$	0.999
Batch size	128
Training ratio (D/G)	5
Gradient penalty (λ)	10

The Rectified Linear Unit (ReLU) activation function is used in all layers of the discriminator with the exception of the last.

Integration into real detector simulations







The new fast simulation (AF3) significantly improves jet substructure with respect to the older one (AF2)

Ideally, the same calibrations derived for full sim. (Geant4-based) can be applied to the fast sim.

As expected, the fast sim. timing is independent of energy, while Geant4 requires more time for higher energy.

AtlFast3: the next generation of fast simulation in ATLAS (ATLAS Collaboration), 2109.02551 (2021)

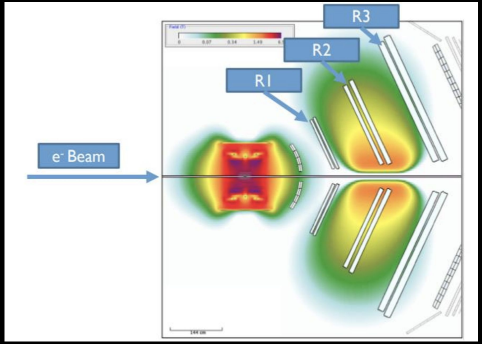
More details on AD

- Gradient-based optimization
  - Gradient descent; (damped) Newton's method; quasi-Newton methods (BFGS, L-BFGS-B, etc)
  - Can require user-defined stopping-criteria
  - For applications, solutions that do not perfectly reach the global minimum, typically still present a valuable improvement over previous designs
  - Popular optimization algorithms are implemented in open-source packages such as SciPy, with the gradient of the objective supplied by the PyTorch's AD module
- Computing derivatives
  - Numerical differentiation; symbolic differentiation; automatic differentiation
- Automatic differentiation
  - Forward Mode: extends each variable a by a variable a', for its partial derivative in some direction, also called "tangent". Time complexity to compute the full gradient is proportional to the time complexity of calculating the function f times the number of input variables
  - Reverse Mode: all statements are recorded usually in computational graph or stack=like data called tape; after this primal run, each primal variable is extended by an adjoint variable; the adjoint variables are successively updated while revisiting the statements in reverse; the time complexity relative to the primal is independent of the number of input variables, making it faster than the forward mode; however recording a tape requires a significant amount of memory

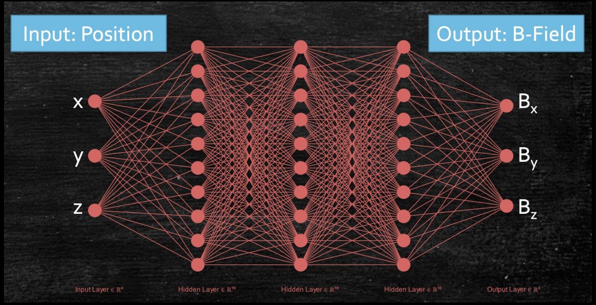
MODE, Toward the End-to-End Optimization of Particle Physics Instruments with Differentiable Programming: arXiv:2203.13818, 2022

Magnetic Field with DNN

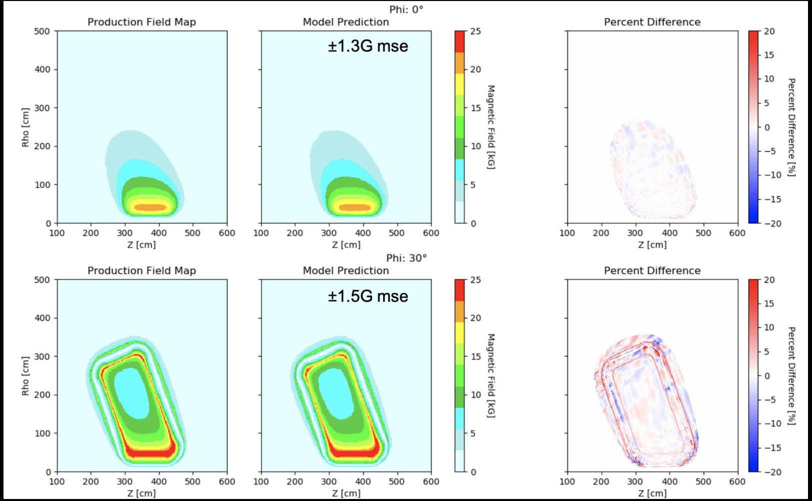
Courtesy of A. Quiroga, W. Phelps
AI Town Hall@JLab 2021 — Magnetic Fields with DNN

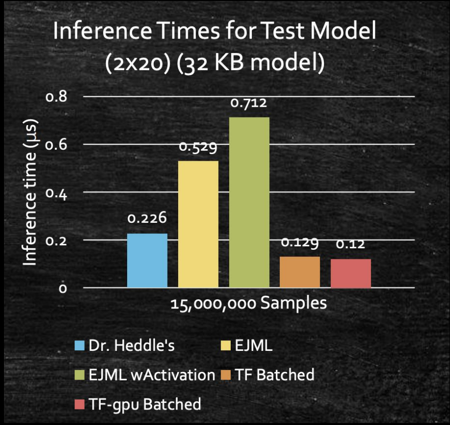


- The production magnetic field was ~1.5 GB (2019) for both solenoid and torus fields combined.
- Can a neural network model be faster?
- It could also be used for OSG transfers to save bandwidth



Input: Position
Output: B-Field
x
y
z
B_x
B_y
B_z


Phi: 0°
Production Field Map
Model Prediction
±1.3G mse
Percent Difference
Phi: 30°
±1.5G mse
Rho [cm]
Z [cm]
Magnetic Field [kG]
Percent Difference [%]


Inference Times for Test Model (2x20) (32 KB model)
Inference time (µs)
15,000,000 Samples
Dr. Heddle's
EJML
EJML wActivation
TF Batched
TF-gpu Batched