Machine Learning for Particle Physics Experiments

Kevin Pedro
(Fermilab)
November 6, 2023

Prefaces

Bias alert

- This talk leans toward collider physics examples
  - And CMS in particular
  - Some connections to other subfields
- Also contains plenty of my personal opinions

Content limitations

- In my last ML plenary, tried to include all relevant conference contributions (CHEP23)
- No way to do that at ML4Jets with over 100 talks!

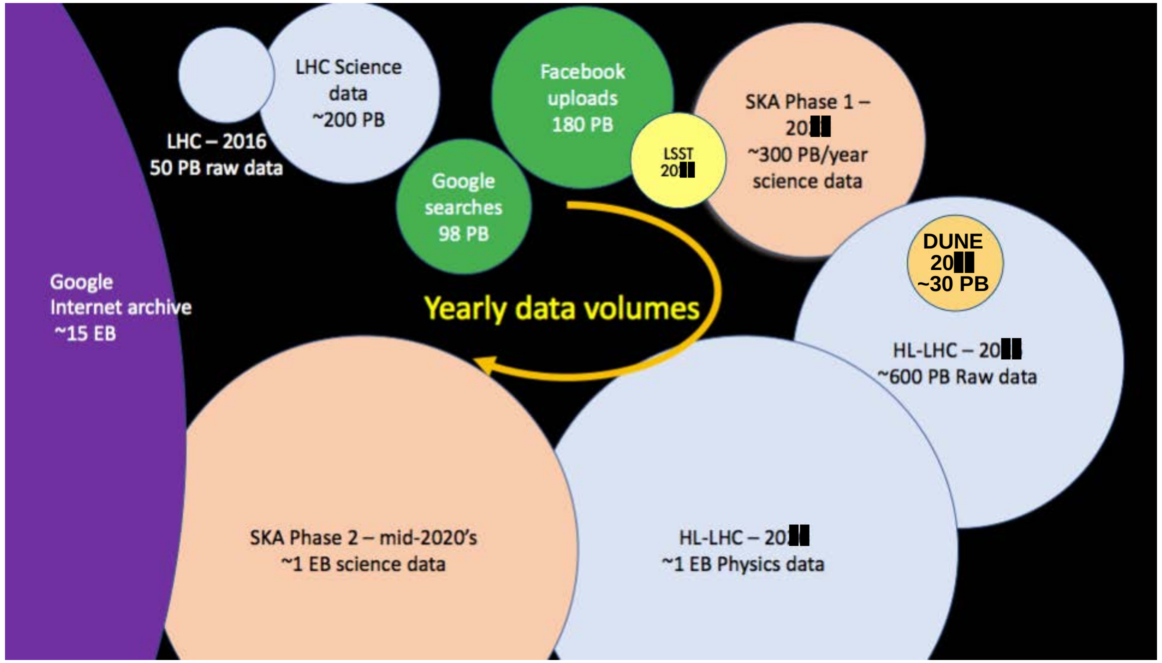
General disclaimer

- ML is a very fast-moving field
- Sorry if I overlooked (or misstated) your work!
  - Just think of it as an opportunity to correct the speaker…

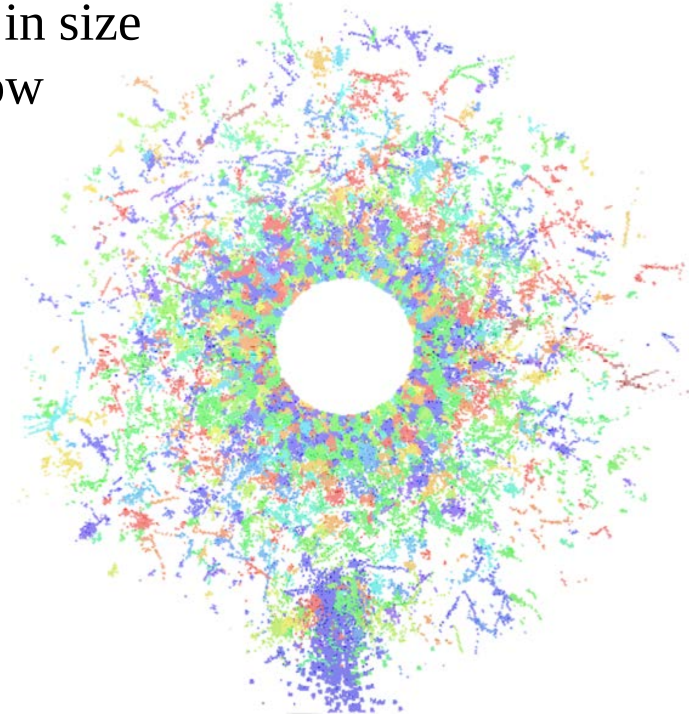
What is our goal?

- My goals:
  1. To learn more about particle physics
  2. Hopefully, to make discoveries!
- ML is a very useful *tool* for these goals → *ML for physics*
  - If applied correctly and efficiently
  - It can also be an unlimited time sink…
- Particle physics data and problems can be very different from industry
  - We naturally refine existing ML techniques and develop new ones → *physics for ML*
    - Among other things, this is a great sales pitch to funding agencies
- Role of ML expert community: not just to develop new cutting-edge tools, but to make them *usable*
  - Requires a balance of phenomenological studies vs. experimental integration
    - Integration is often thankless, but has long-term impact, and helps to develop best practices
  - Also strongly related to *robustness* and *interpretability*
- Ultimate goal: any physicist can extract the best physics from their data *without* being an ML expert

What's stopping us?



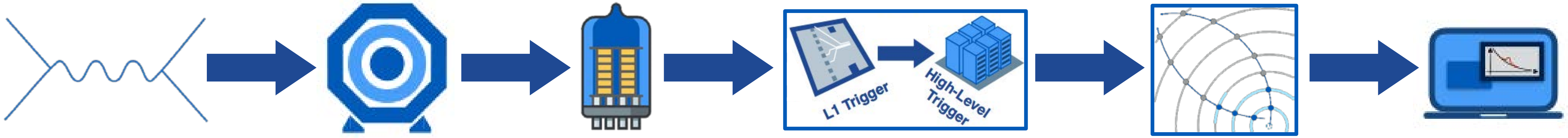
- More data: both a solution and a problem
  - o Opportunities for most precise measurements and discovery of rare processes
  - o Challenges to process, store, and analyze this upcoming flood of data
- Can we do more physics, *more efficiently*?
- As experiments grow in size and intensity, data grow in complexity



- Humans struggle to reason at high dimensionality

➢ Classical algorithms are *fundamentally limited*
  - o Unacceptable errors from simplifying assumptions as precision increases
- ML can take us *much further*
  - o Can we execute ML algorithms within our computing budget?
  - o Can we learn what the machine learns?

Where can we use ML?



Generation	Simulation	Digitization	Trigger	Reconstruction	Analysis
Integration Sampling Showering/ hadronization Tuning Uncertainties …	Generative Refinement Tuning End-to-end …	Pileup Electronics modeling …	Anomaly detection Low-latency On-detector …	Calibration Tracking Clustering …	Classification Regression Simulation-based inference …

Real-time operations for accelerators, telescopes…

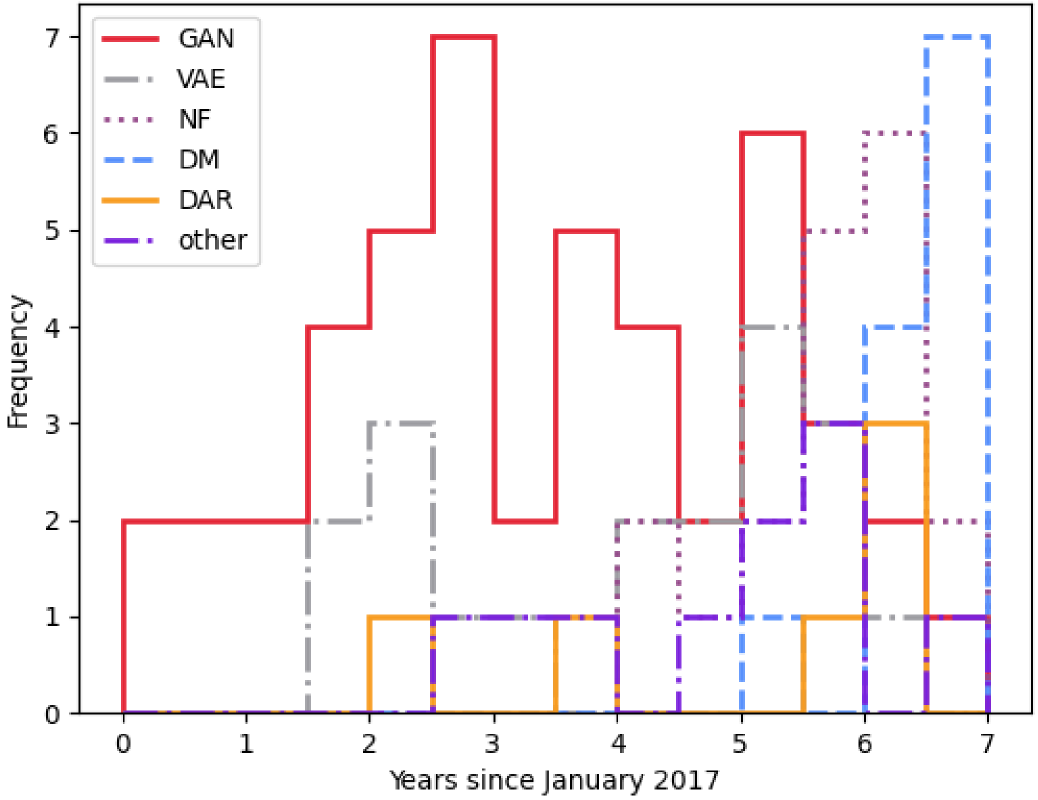
Compression
Resource allocation
Code generation
…

Everywhere!

7 Years of ML4Sim

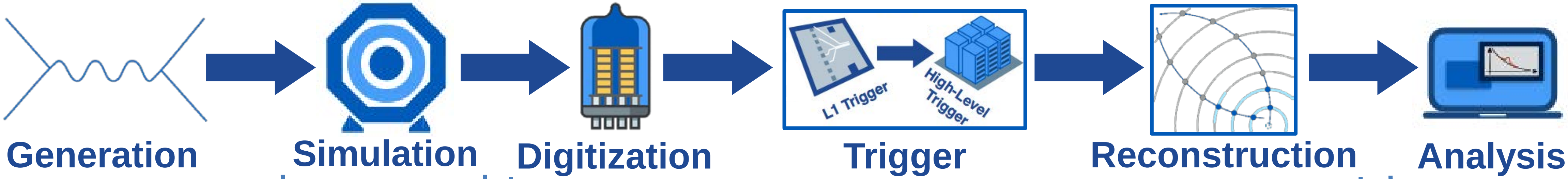
Simulation

- From my database of 92 ML4Sim-related papers
- Normalizing flows and diffusion models supplanting traditional GANs and VAEs
- Diffusion model takeoff in particular looks almost exponential…
  - An *entire session* of ML4Jets dedicated just to these models!
- Some growing interest in autoregressive models
  - Perhaps motivated by success in industry (GPT)
- Common datasets and metrics from CaloChallenge are a big step forward to be able to compare different approaches → summary on Thursday!
- *Experiential knowledge* from ATLAS Run 2/3 deployment of FastCaloGAN also very valuable



“Other” = non-generative models (FCNs, CNNs, GNNs), typically regression-based approaches

One Step vs. End-to-End



Changes O(1 year)
(geometry, physics models)

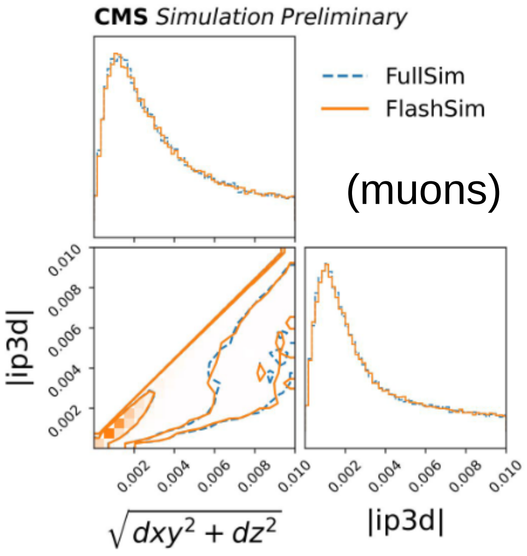
Changes O(1 second–1 month)
(calibrations, radiation damage, algorithms, …)

Changes O(1 year)
(mostly performed after data-taking finishes)

- End-to-end models like [FlashSim] that produce analysis-level observables from generator input have massive utility: essentially eliminate statistical fluctuations
  - …*for end-stage analysis*, where nothing is rapidly varying
- But accurate simulation is needed *throughout* the lifecycle of an experiment

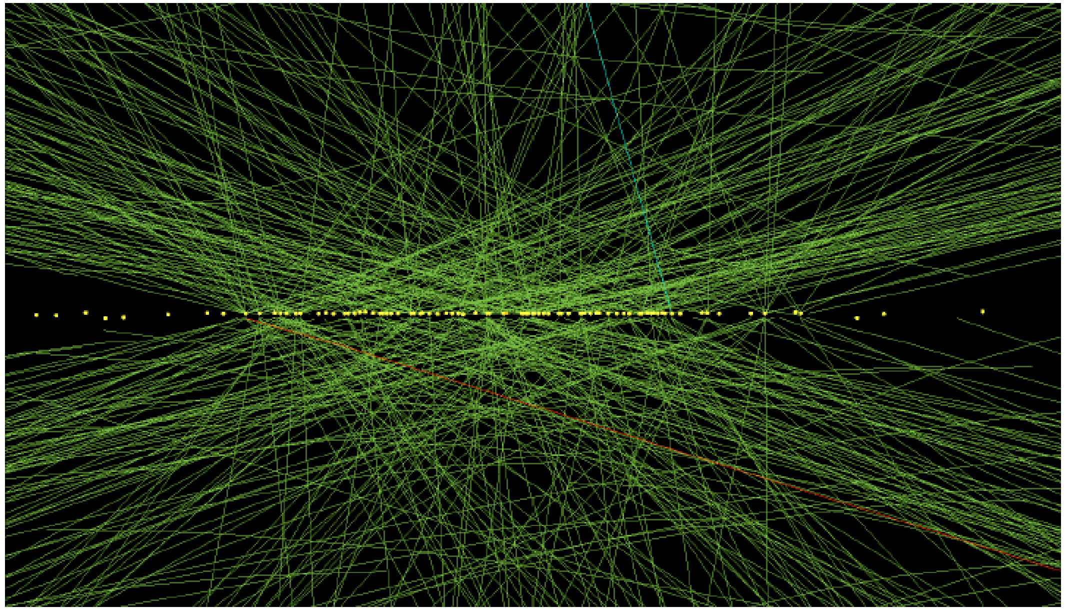
➢ Models that target simulated hits are *more broadly* applicable

  - Complementary use cases for both approaches



Pileup: An Overlooked Case

Digitization



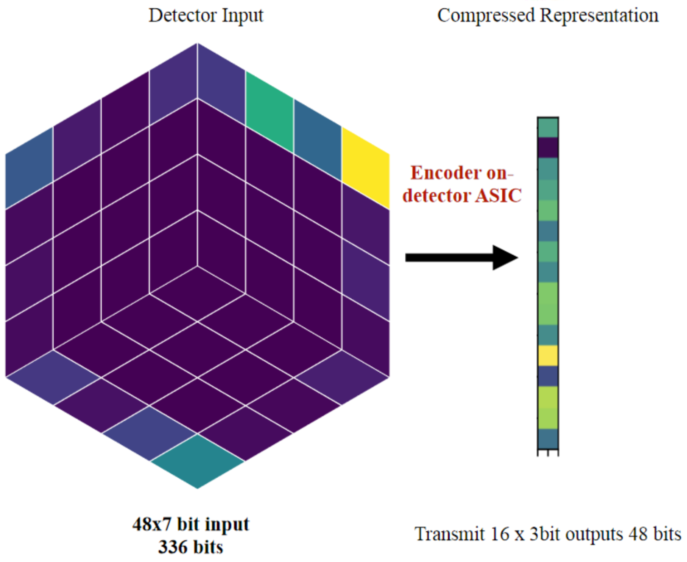
- “Classical” mixing: overlay n_{PU} *distinct* simulated minimum bias events *per bunch crossing* on top of signal event → massively I/O intensive
- “Premixing”: perform overlay in advance, save hits after aggregation (digitized format)
    - Leads to O(PB) samples that have to be served throughout the grid with very high availability
    - Better than classical mixing, but still disk- and network-intensive
- Viewed as a solved problem… but substantial room for improvement
    - Generative ML could compress O(PB) samples into O(MB) model + RNG & conditioning info → *completely eliminate* premixing resource usage, in exchange for training
- Straightforward to repurpose detector simulation surrogates, but also possible improvements here
    - Train on data and realize long-awaited data mixing?
- Stay tuned for DeGeSim talk on Friday!

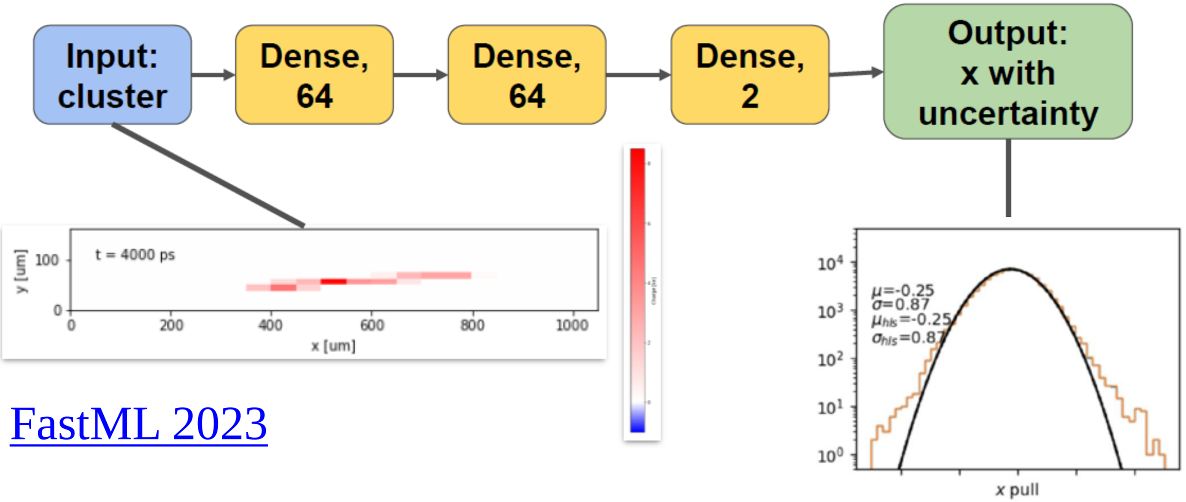
Anomaly Triggers

- ~99.999% of LHC data is discarded (can’t write 600 TB/s to disk)
  - Most of it is uninteresting… but how do we know we’re picking the most interesting 0.001%?
- CMS approach: train a (variational) autoencoder on zero bias data
  - CICADA: Calorimeter Image Convolutional Anomaly Detection Algorithm
    - Uses calorimeter trigger inputs
  - AXOL1TL: Anomaly eXtraction Online Level-1 Trigger Lightweight
    - Uses global trigger objects (jets, MET, leptons)
- Deploy at L1 trigger on FPGA using hls4ml
  - Achieve latencies as low as 50 ns!
  - How do they do it? Check out Tuesday’s talk!
- These triggers will operate for *next 2 years* of LHC Run 3
- Looking forward to very interesting data…

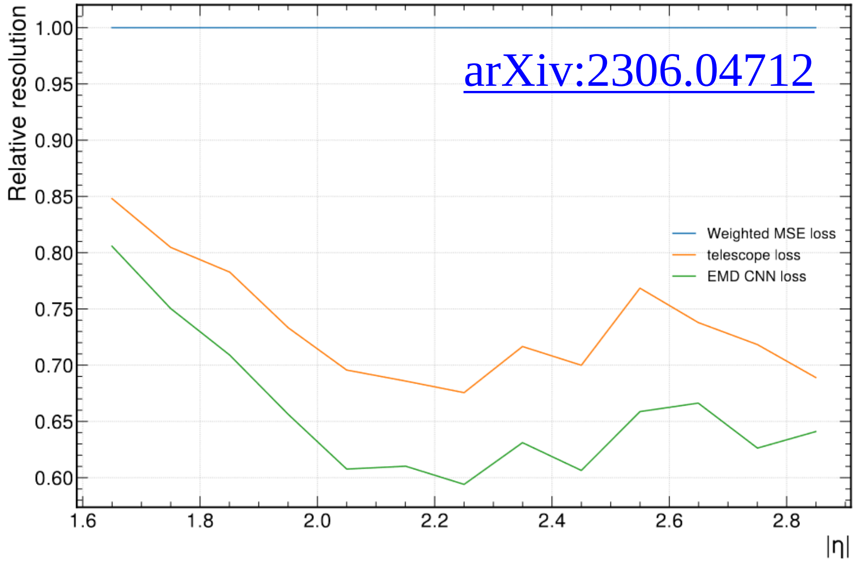
Detector Intelligence

- CMS High Granularity Calorimeter will have 6 million channels
  - No way to read all of them in 12 μs latency
  - *Compress* using on-detector ASIC running CNN encoder!
- Latest advance: train encoder using differentiable Earth Mover's Distance loss (implemented as CNN surrogate)
  - Substantial improvement in electron resolution
- ❖ Into the future: *smart pixels* for single-layer tracking
  - Use Mixture Density Network to predict parameters and errors





[FastML 2023](#)



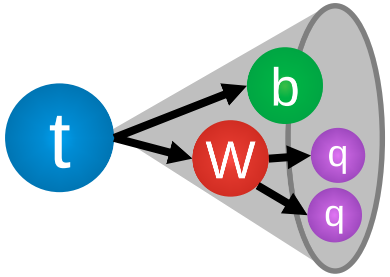
Classification

arXiv:2202.03772

Analysis

	All classes		$H \to b\bar{b}$	$H \to c\bar{c}$	$H \to gg$	$H \to 4q$	$H \to \ell\nu qq'$	$t \to bqq'$	$t \to b\ell\nu$	$W \to qq'$	$Z \to q\bar{q}$
	Accuracy	AUC	$\mathrm{Rej}_{50\%}$	$\mathrm{Rej}_{50\%}$	$\mathrm{Rej}_{50\%}$	$\mathrm{Rej}_{50\%}$	$\mathrm{Rej}_{99\%}$	$\mathrm{Rej}_{50\%}$	$\mathrm{Rej}_{99.5\%}$	$\mathrm{Rej}_{50\%}$	$\mathrm{Rej}_{50\%}$
PFN	0.772	0.9714	2924	841	75	198	265	797	721	189	159
P-CNN	0.809	0.9789	4890	1276	88	474	947	2907	2304	241	204
ParticleNet	0.844	0.9849	7634	2475	104	954	3339	10526	11173	347	283
ParT	**0.861**	**0.9877**	**10638**	**4149**	**123**	**1864**	**5479**	**32787**	**15873**	**543**	**402**
ParT (plain)	0.849	0.9859	9569	2911	112	1185	3868	17699	12987	384	311

- ML4Jets 2018: comparison studies eventually led to the Greatest Of All Taggers (GOAT)
  - o Performance slightly exceeding ParticleNet
- Particle Transformer (ParT) is a *massive step forward*
  - o Many more parameters, but fewer operations (→ faster!)
  - o Uses pairwise features from 4-vectors → *domain knowledge*
- Other transformers like GN2X also being explored: H → $b\bar{b}$ $\mathrm{Rej}_{50\%}$ = 300



Architecture	Accuracy	AUC	$1/\epsilon_B$	# Params
$\mathrm{LorentzNet}_{n_{\mathrm{hidden}}=3}$	0.9056	0.9623	136	120
$\mathrm{nPELICAN}_{C_{\mathrm{hidden}}=10}$	0.9208	0.9747	332	101
$\mathrm{nPELICAN}_{C_{\mathrm{hidden}}=3}$	0.9179	0.9722	272	31
$\mathrm{nPELICAN}_{C_{\mathrm{hidden}}=2}$	0.9169	0.9716	255	21
$\mathrm{nPELICAN}_{C_{\mathrm{hidden}}=1}$	0.8951	0.9480	70	11

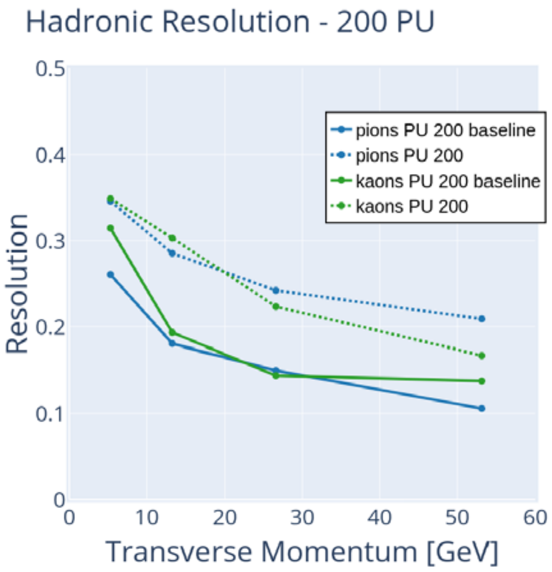
arXiv:2310.16121

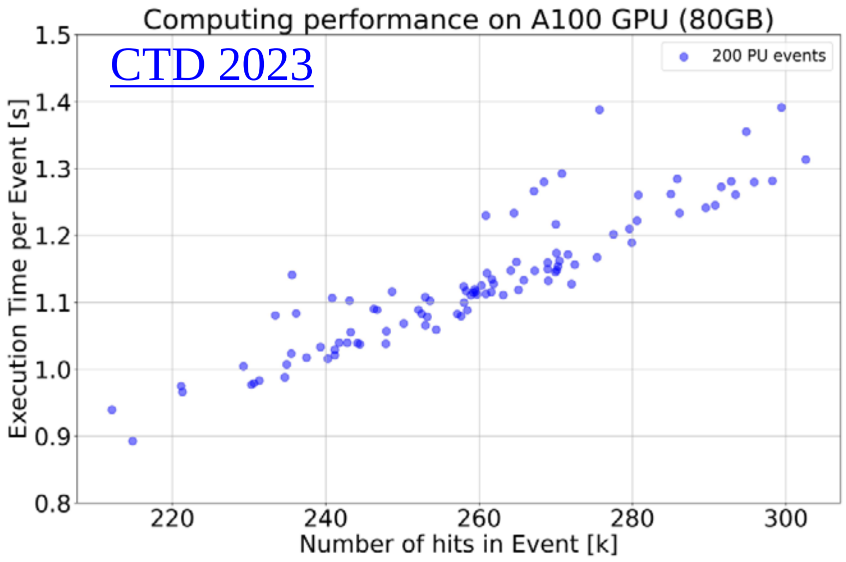
- Opposite side of the spectrum: PELICAN achieves competitive performance with only O(100) parameters!
  - o Incorporates even more domain knowledge
  - o $n_{\mathrm{param}} = 10C_{\mathrm{hidden}} + 1$

Beyond Classification: Clustering

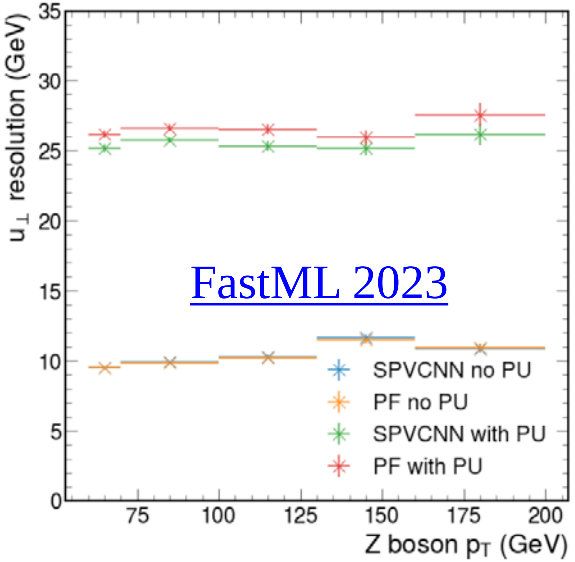
Reconstruction

- Upcoming high granularity calorimeters will measure particle showers with *unprecedented accuracy*
  - o *No known classical algorithm* can handle this level of information
- Leading approach: GravNet architecture and Object Condensation loss
  - o Graphs formed in latent space; each hit scored for seeding
- Latest tests on simplified geometry (still 3M channels):
  - o Good physics performance in high pileup, *linear* inference scaling!





CTD 2023



FastML 2023

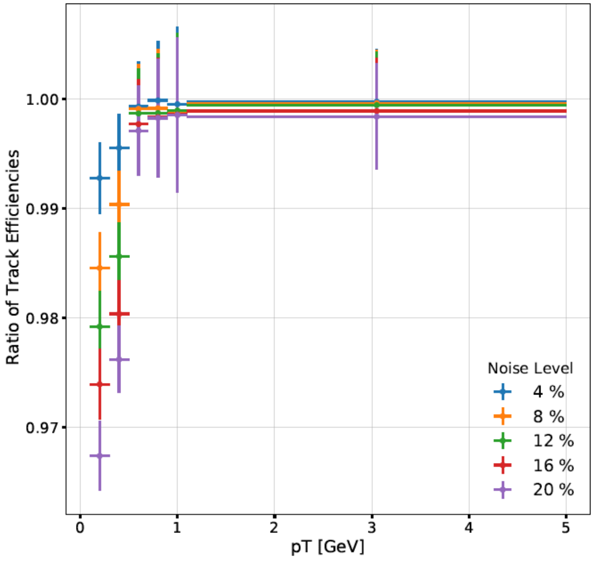
- Alternative architecture: SPVCNN
  - o Feature transforms in point space, convolutions in voxel space
  - o Fast and memory-efficient
- Easy to accelerate on GPU: competitive performance w/ domain algorithm and 16× faster

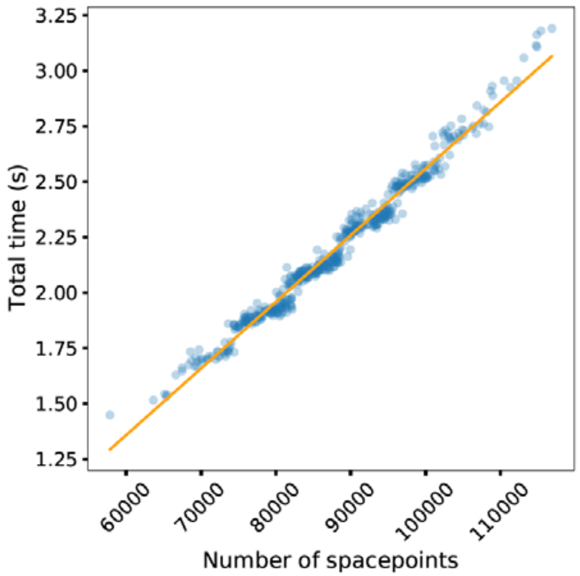
Beyond Classification: Tracking

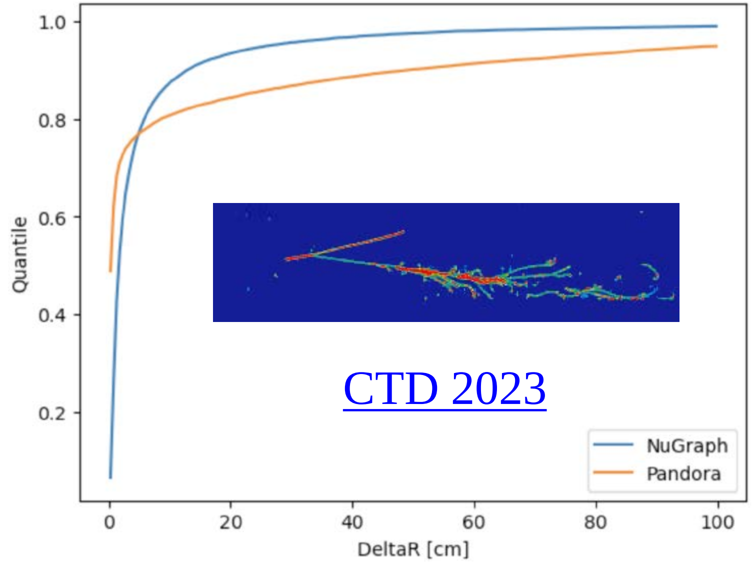
Reconstruction

- GNN architectures also work well for tracking
- ExaTrkX developing solutions for various experiments
  - o Robust to noise for track p_T > 1 GeV in collider setup
  - o Again, *linear* inference!
    - ▪ Remember, naïve expectation is O(n²) scaling (less naïve: n log n)

arXiv:2103.06995





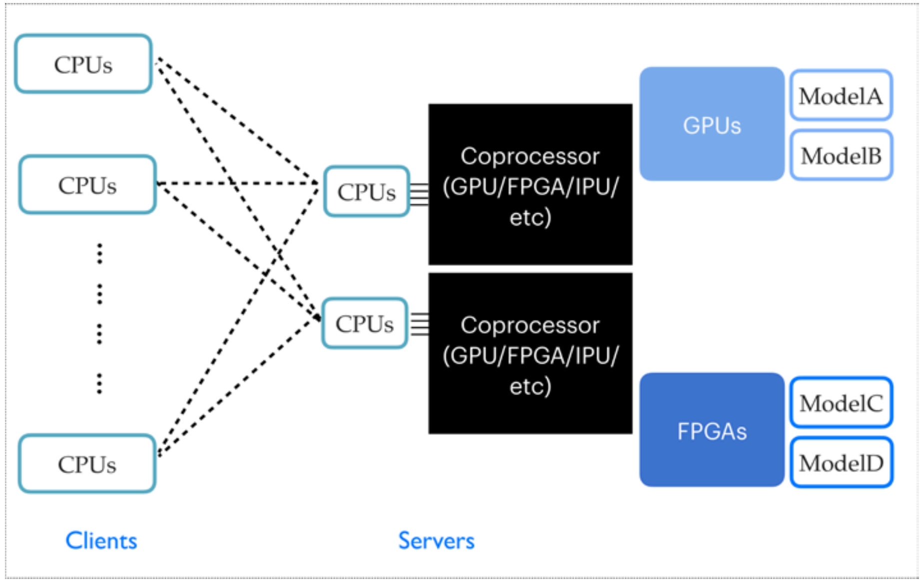


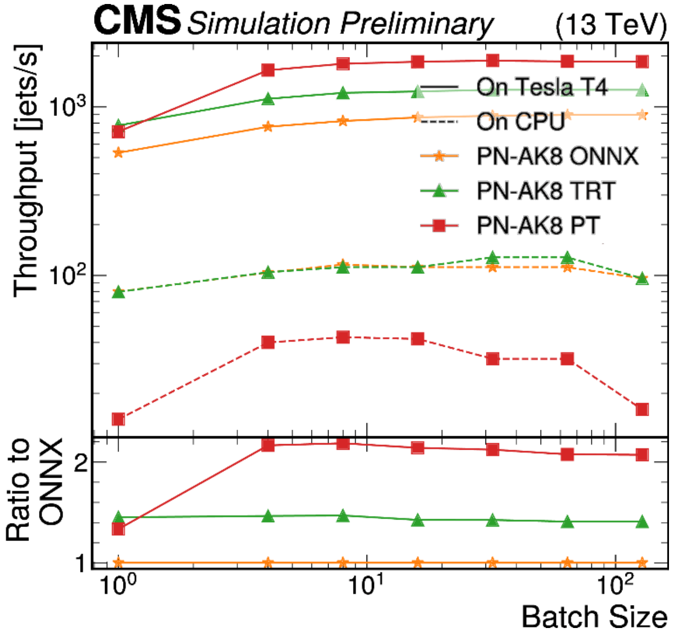
CTD 2023

- Similar architecture can be applied to LArTPC detectors in neutrino experiments
  - o Among first users of modern ML (CNNs) for classification!
- Robust performance vs. classical algorithm
  - o Also fast: 0.005 s/evt inference on GPU with batching

Computing for ML

- ML algorithms use a restricted set of operations (mostly matrix multiplications)
  - o Natural and easy to accelerate on specialized coprocessors
- *Most flexible* approach: inference as a service
  - o Abstract away specific computing elements: client makes request, server delivers
  - o Example: ~10× speedup in ParticleNet on GPU vs. CPU
    - ▪ Algorithm latency becomes essentially *invisible* with asynchronous calls in offline processing
    - ▪ Can batch *across events* for optimal GPU utilization → maximize throughput
- Demonstrated for [CMS](), [protoDUNE](), [LIGO]()
  - o Use CPUs, GPUs, FPGAs, IPUs… with zero code changes!





Constrained Optimization

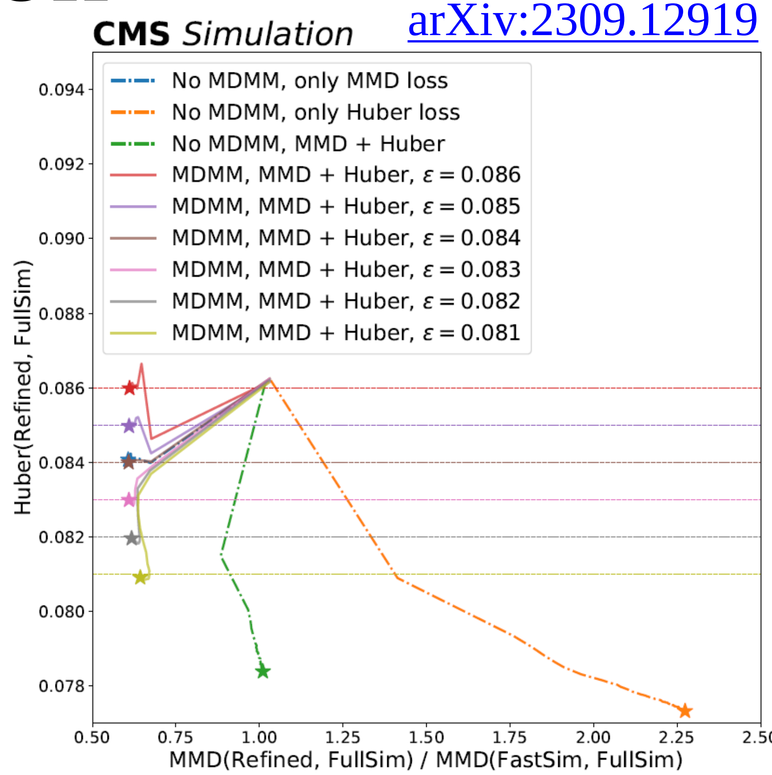
[arXiv:2309.12919]

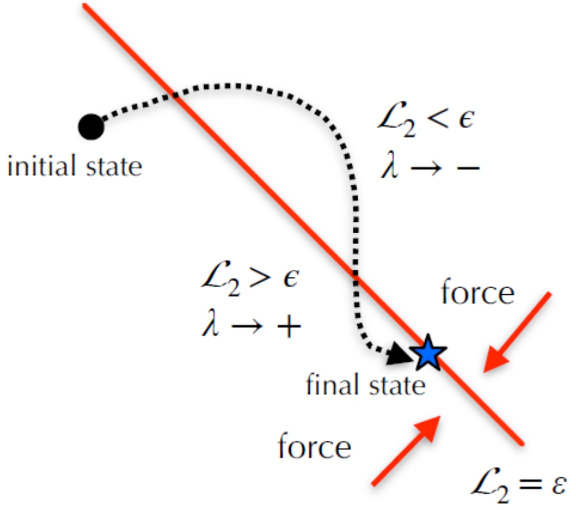
- *General principle*: you can't optimize for two things at once
  - o Instead, optimize for one thing with *constraints* on others (Lagrange)
- Multiple loss terms are one approach to encode domain knowledge

☠ $\mathcal{L} \to \mathcal{L}_1 + \lambda\mathcal{L}_2 + \cdots$; set λ by trial and error → *objectively suboptimal*

➢ modified differential method of multipliers (mdmm): [paper, blog, code]

$$\mathcal{L} \to \underset{\text{gradient \textit{ascent}}}{\mathcal{L}_1} - \overset{\textit{learnable}}{\lambda}(\underset{\text{constraint}}{\varepsilon} - \mathcal{L}_2) + \overset{\text{hyperparameter (convergence rate)}}{\delta}\underbrace{(\varepsilon - \mathcal{L}_2)^2}_{\text{damping to ensure convergence}}$$





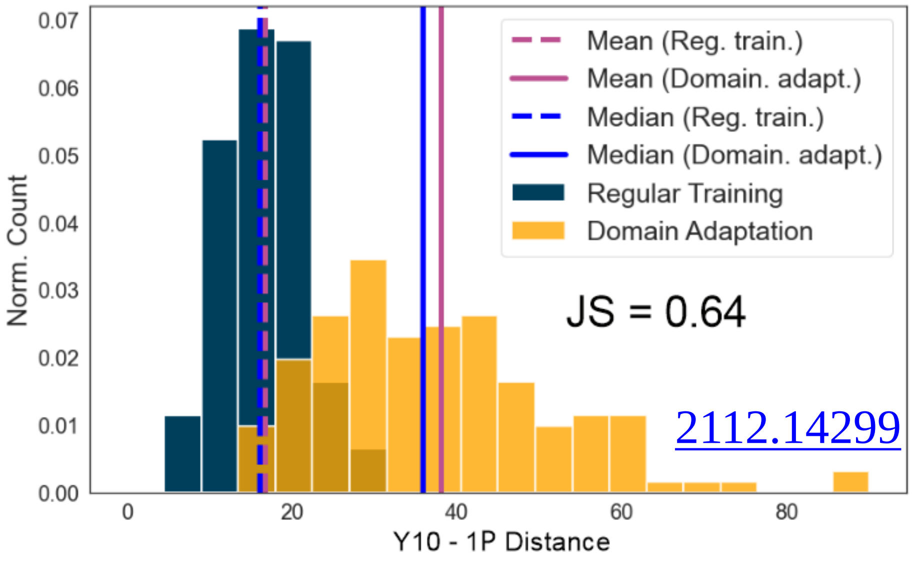
- First known usage in HEP: balance per-event and ensemble losses for ML-based refinement of classical FastSim
  - o Minimize per-event: bad ensemble value
  - o Minimize ensemble: per-event still good!
  - o Learn more later today!

➢ Find Pareto front (concave or convex) and pick tradeoff

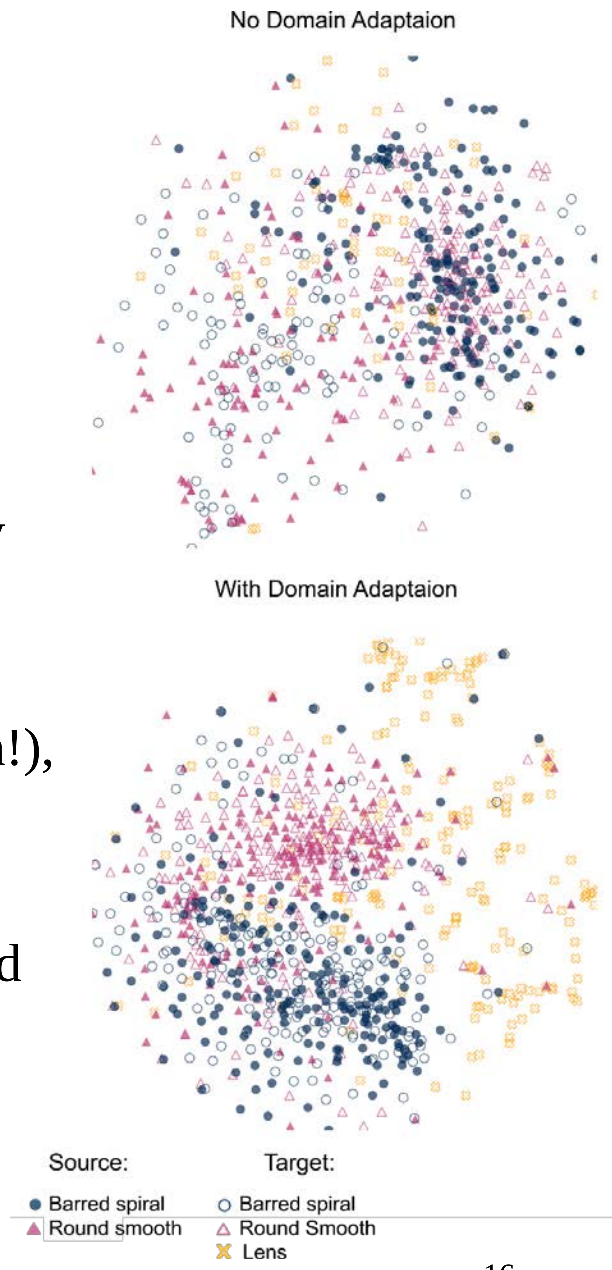
Domain Adaptation

arXiv:2302.02005

- Various techniques: adversarial training, gradient reversal, MMD, DisCo, etc.
  - Frequently explored in methods papers; occasionally by experiments
- Recent results in astrophysics:
  (simulations less reliable, many data sources to compare)
  - DA (using MMD) increases robustness against adversarial attacks
  - DA using semi-supervised *contrastive learning* (adaptive clustering + entropy separation) has multiple benefits:
    - Alignment of source and target classes: better performance (on both!), *physically meaningful* latent space
    - *Anomaly detection* capabilities: known & unknown classes separated



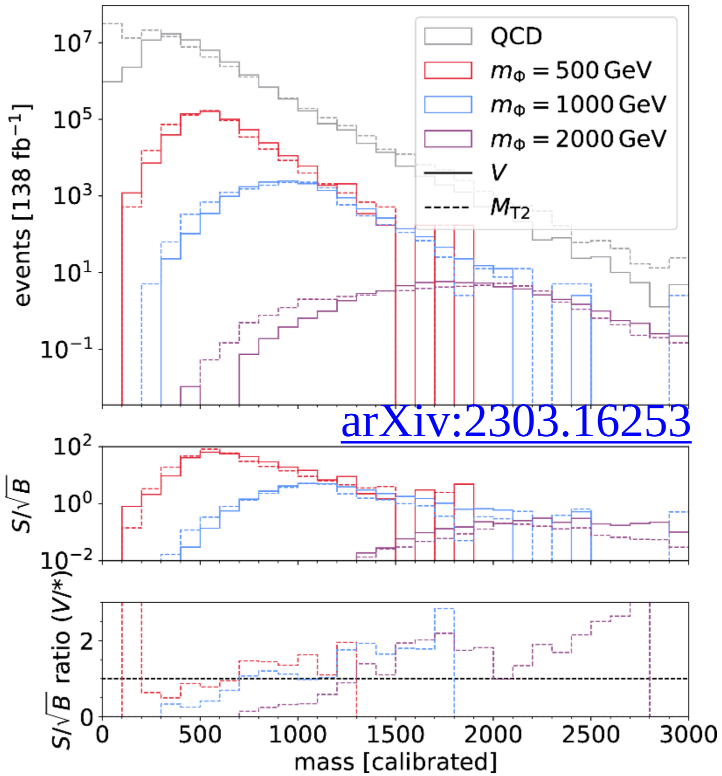
2112.14299



Interpretability

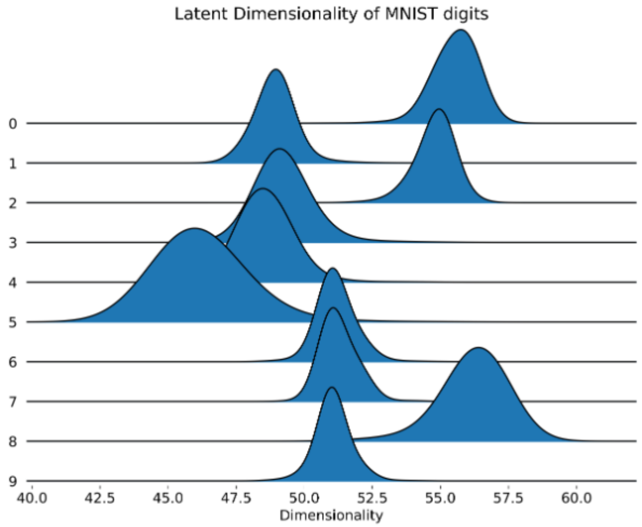
- From last year: semi-supervised information bottleneck to learn optimal mass variables (beats M_{T2} in semivisible final state)

➢ How to scale up to higher dimensions, tougher questions?



[arXiv:2303.16253](https://arxiv.org/abs/2303.16253)

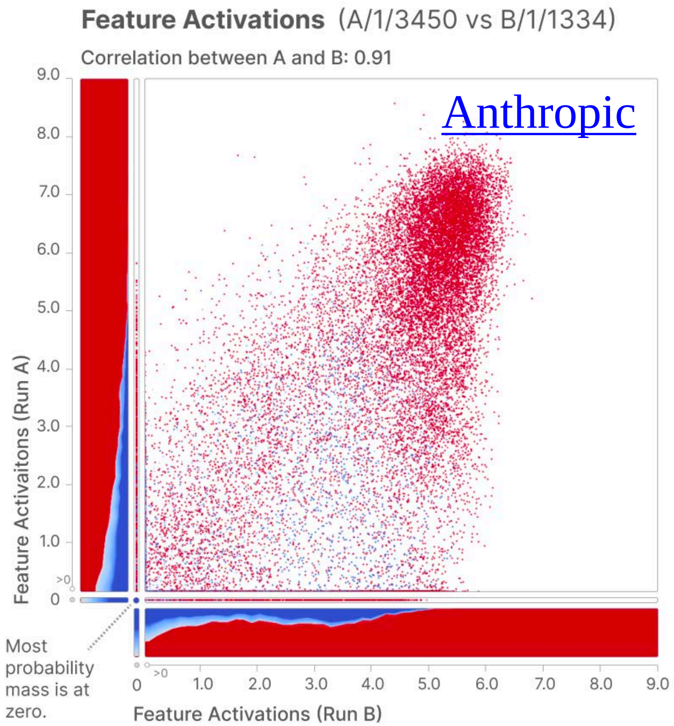
- Sparsity-inducing categorical prior can learn optimal latent dimension, improve both performance and robustness



[arXiv:2203.02592](https://arxiv.org/abs/2203.02592)

- Still a long way from learning what the machine is learning…
- But *new techniques* are bringing us closer than ever before!

- New approach to decompose superposition/polysemanticity in LLM neurons: finds consistent features between different networks

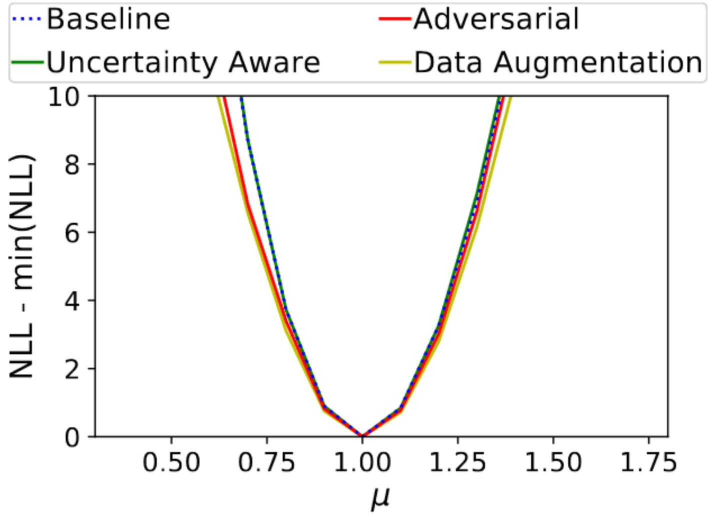


Anthropic

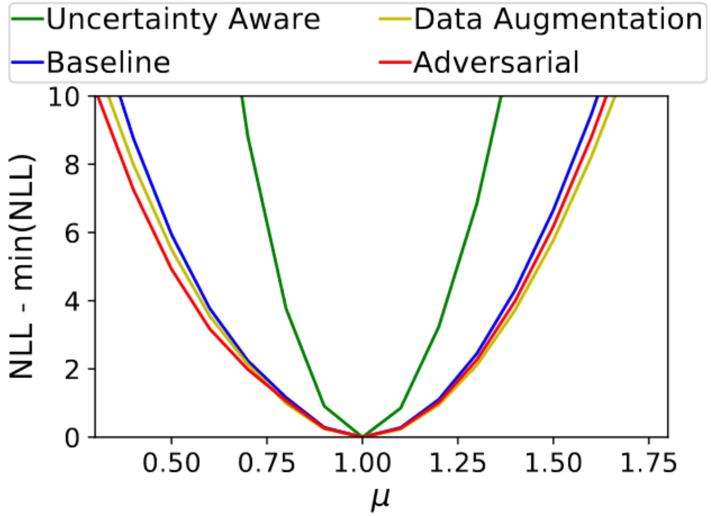
Uncertainty

- Uncertainty *quantification*: never convinced of its importance
  - Classifiers are just functions: propagate input feature uncertainties and call it a day
- But a different story for generative models
  - LLMs are known to “hallucinate” (really *confabulation*)
  - Would we know if our models do the same?
- Also important for parameter estimation tasks
  - Mixture Density Networks are useful there
- Uncertainty *reduction*, on the other hand: the name of the game
  - Uncertainty-aware training handles in-domain and out-of-domain data equally well for H → ττ
- Looking forward to more discussion throughout the week!

arXiv:2105.08742



(a) Data generated with $z = \frac{\pi}{4}$.



(b) Data generated with $z = \frac{\pi}{2}$.

Foundation Models

- A growing trend in industry:
  - LLMs: finetuning or simply tokenizing additional data
    - Apply already-learned relationships to new information
  - Image generation: textual inversion, low-rank adaptation
- Back to ParT: not just >2M parameters, but trained on *100M jets*
  - *2 orders of magnitude* higher than previous datasets
  - Fine-tuned version performs *better* on those previous datasets than fully retrained version!
  - ➢ Learns *generalized physics* that can apply to many datasets
- Maybe the *first glimmer* of foundation models for physics
  - Build on techniques mentioned here (and others) to improve generalization and physics learning
  - Diffusion models could be foundation for generative tasks…

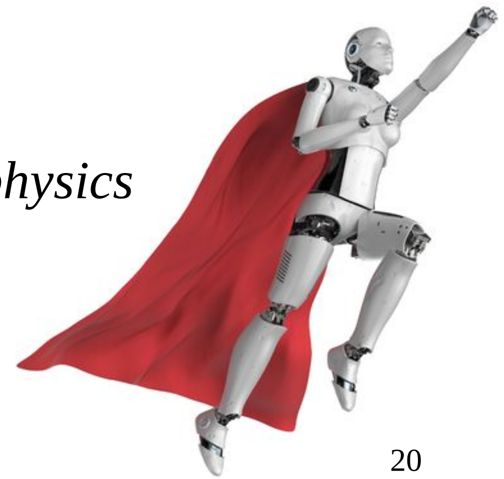
arXiv:2202.03772

	Accuracy	AUC	$\mathrm{Rej}_{50\%}$	$\mathrm{Rej}_{30\%}$
P-CNN	0.930	0.9803	201 ± 4	759 ± 24
PFN	—	0.9819	247 ± 3	888 ± 17
ParticleNet	0.940	0.9858	397 ± 7	1615 ± 93
JEDI-net (w/ $\sum O$)	0.930	0.9807	—	774.6
PCT	0.940	0.9855	392 ± 7	1533 ± 101
LGN	0.929	0.964	—	435 ± 95
rPCN	—	0.9845	364 ± 9	1642 ± 93
LorentzNet	0.942	0.9868	498 ± 18	2195 ± 173
ParT	0.940	0.9858	413 ± 16	1602 ± 81
ParticleNet-f.t.	0.942	0.9866	487 ± 9	1771 ± 80
ParT-f.t.	**0.944**	**0.9877**	$\mathbf{691 \pm 15}$	$\mathbf{2766 \pm 130}$

Conclusion

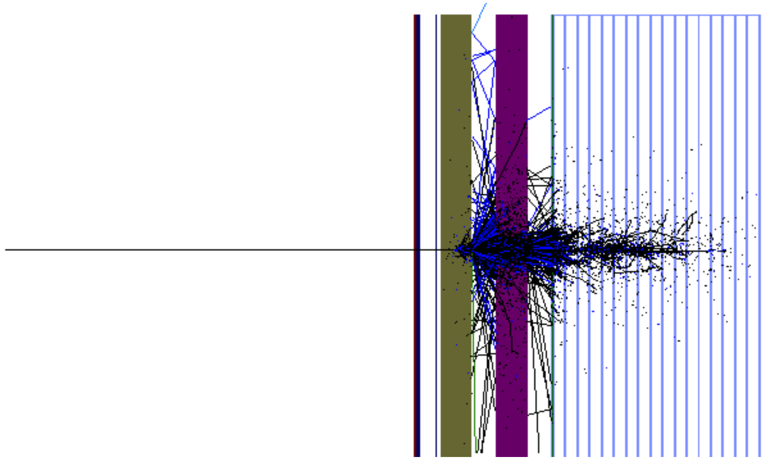
- ML has impacts *throughout* particle physics
  - Every subfield: collider, neutrino, astro, accelerator, computing, and beyond
  - Every step: simulation, digitization, triggering, reconstruction, analysis, and more
- The field is maturing:
  - Converging on the “right ways” to perform (at least some) tasks
  - Deploying ML algorithms at larger scales and for more types of problems
  - Building a new toolkit—including mixture density networks, mdmm, contrastive learning, and more—to make our ML more *reliable* and more *physically meaningful*
- *Interpretability* and *uncertainty* are two big outstanding (and related) questions
  - Hopefully the new toolkit helps us make more progress
- Foundation models will help achieve our ultimate goal: for *everyone* to do the *best physics*

➢ The future is bright!



Backup

Simulation Landscape

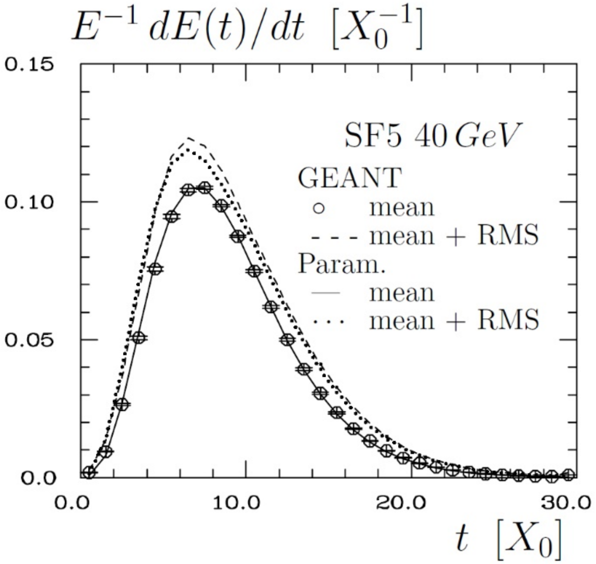


"FullSim"

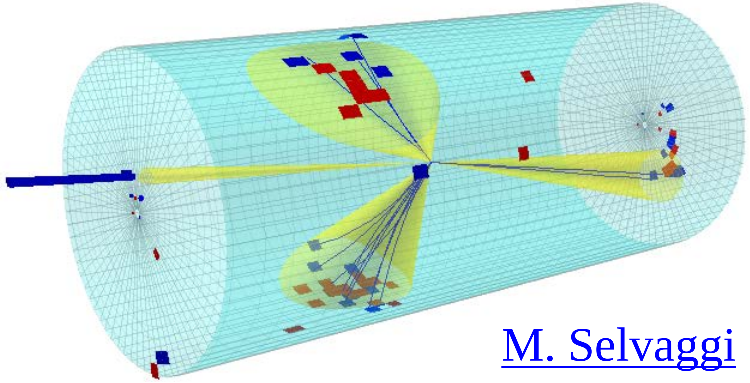
- Common software framework (i.e. Geant4)
  - Experiments can provide additional code via user actions
- Explicit modeling of detector geometry, materials, interactions w/ particles

"FastSim"

- Usually experiment-specific framework
- Implement approximations: analytical shower shapes (e.g. GFLASH), truth-assisted track reconstruction, etc.



arXiv:hep-ex/0001020



M. Selvaggi

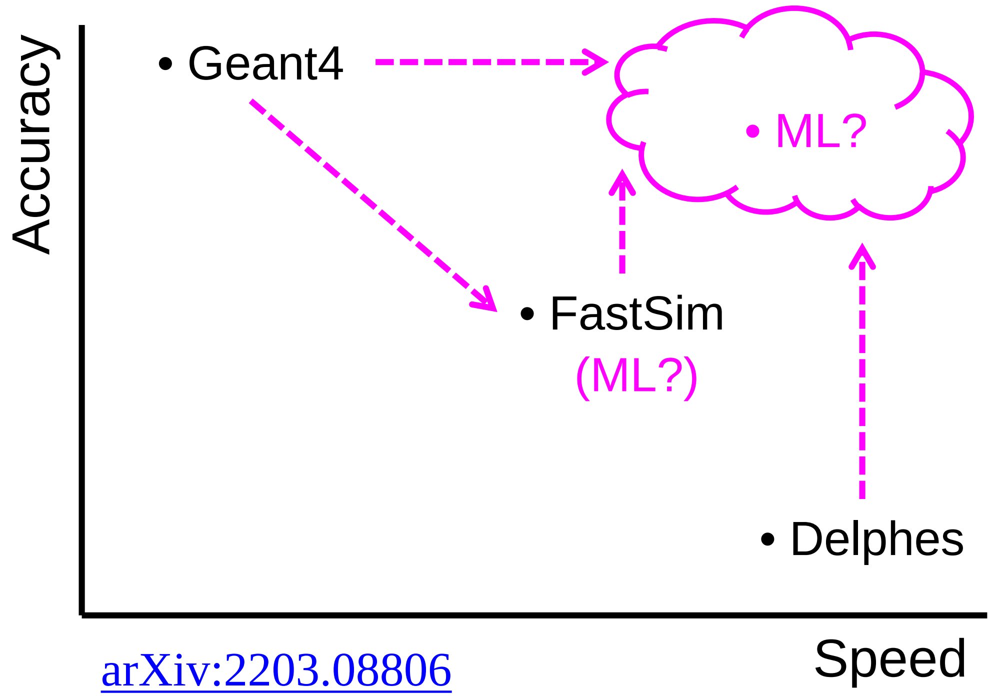
Delphes

- Ultra-fast parametric simulation
- Used for phenomenological studies, future projections, etc.

Simulation is crucial in HEP!

ML4Sim Landscape

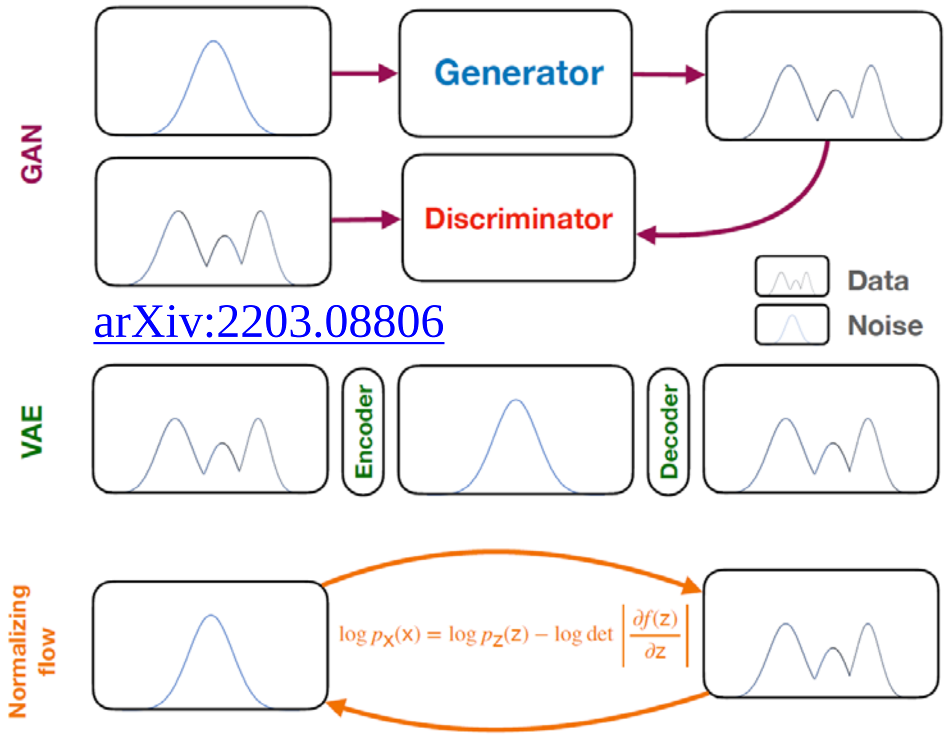
- Options to use ML for sim:
  1. Replace or augment (part or all of) Geant4
  2. Replace or augment (part or all of) FastSim
- Goals:
  1. Increase speed while preserving accuracy
  2. Preserve speed while increasing accuracy
- ML can also create faster, but less accurate simulation
  - à la existing classical FastSim
    - then augment w/ more ML to improve accuracy
- Another option: replace entire chain (“end-to-end”)
  - Exciting prospect, potentially complements other cases



[arXiv:2203.08806](https://arxiv.org/abs/2203.08806)

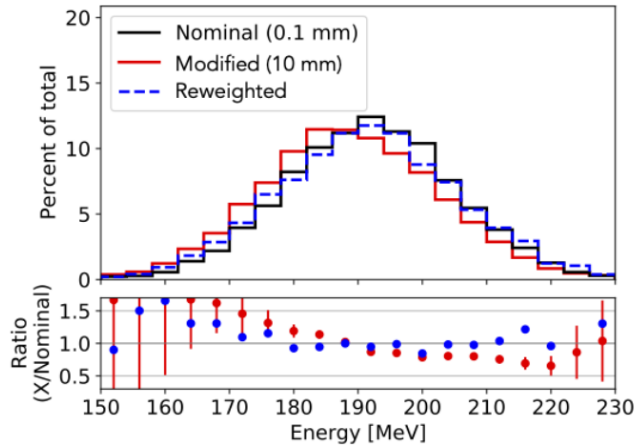
Taxonomy

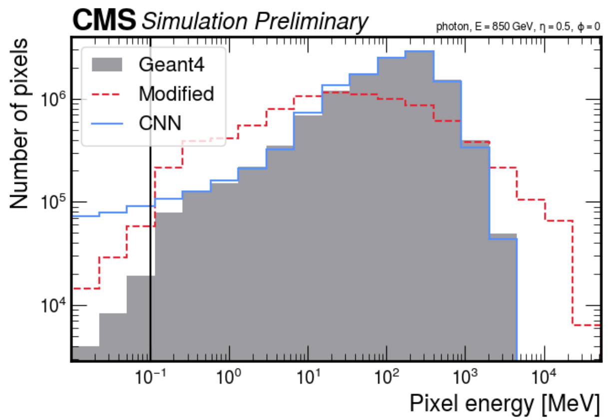
- Generative models ("replace"):
  - Usually *stochastic*
  - Generative Adversarial Networks (GANs)
  - Variational Autoencoders (VAEs)
  - Normalizing Flows (NFs)



arXiv:2203.08806

- Refinement techniques ("augment"):
  - Usually *deterministic*
  - Classification-based (reweighting)
  - Regression-based (correcting)





Metrics

- Speed only matters if needed accuracy is achieved
  - Wrong answers can be obtained infinitely fast
- Looking at 1D histograms: not good enough!
  - Can miss high-dimensional correlations
- Best category: **integral probability metrics**

$$D_{\mathcal{F}}(p_{\mathrm{real}}, p_{\mathrm{gen}}) = \sup_{f \in \mathcal{F}} \left| \mathbb{E}_{\mathbf{x} \sim p_{\mathrm{real}}} f(\mathbf{x}) - \mathbb{E}_{\mathbf{y} \sim p_{\mathrm{gen}}} f(\mathbf{y}) \right|$$

  - *Wasserstein distance* W_1: $\mathcal{F}$ is set of all K-Lipschitz functions
    - Only works well in 1D, biased in high-D
  - *Maximum mean discrepancy* (MMD): $\mathcal{F}$ is unit ball in reproducing kernel Hilbert space
    - Depends on choice of kernel
  - *Fréchet distance*: W_2 distance between Gaussian fits to (high-D) feature space
    - Features can be hand-engineered or obtained from NN activations
- Another interesting category: *classifier scores*
  - Train NN to distinguish real vs. generated
  - AUC score ranges from 0.5 to 1.0
- *Fréchet Particle Distance* most clearly distinguishes between two similar approaches (message passing GAN and generative adversarial particle transformer)

	FPD $\times 10^3$	KPD $\times 10^3$	W_1^M $\times 10^3$
Truth	0.08 ± 0.03	-0.006 ± 0.005	0.28 ± 0.05
MPGAN	$\mathbf{0.30 \pm 0.06}$	$\mathbf{-0.001 \pm 0.004}$	$\mathbf{0.54 \pm 0.06}$
GAPT	0.66 ± 0.09	0.001 ± 0.005	0.56 ± 0.08

arXiv:2211.10295