

高エネルギー物理実験におけるニューラルネットワーク応用について



Masahiro Morinaga

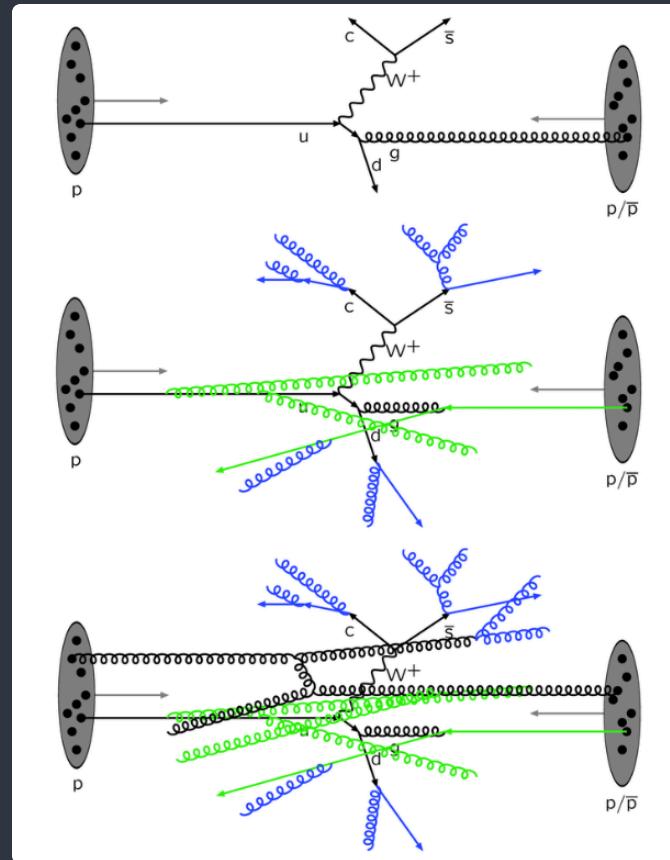
The University of Tokyo
(ICEPP, Beyond AI, ATLAS)

自己紹介

- Masahiro Morinaga (森永真央), 特任助教@ICEPP(東大)
 - 徳島県出身
 - 神戸大学(首席排出)から東大
 - 趣味: キーボード(PCの方)
- LHC-ATLAS実験で物理解析をしている(最近は若い人を鼓舞するだけ)
 - SUSY chargino search: 長寿命なチャージーノを短いトラックで探す
 - MSSM Higgs search: タウ粒子対へと崩壊する重いBSMヒッグス粒子探す



そもそも LHC とは



LHC

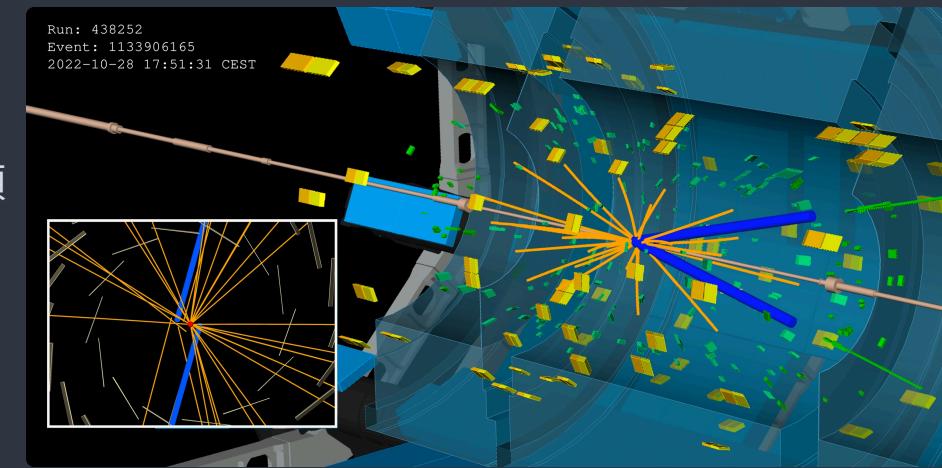
- スイスのジュネーブにあるCERN研究所で行われている実験
- 陽子同士をぶつけて出てくる粒子を検出する
- 陽子はバンチとよばれる集団で加速され同時に多数の衝突
- 40MHzで衝突が起こるように設計されている
- ヒッグス粒子や標準模型を超える粒子を探したい
 - もちろん標準模型の粒子の精密測定もしたい

要するに

- 多数の反応をざっくりすばやく処理したあとに、
- じっくり高精度に検出器の反応を読んで解析する
- どちらの要素でも機械学習は有望で研究されている

なぜLHCで機械学習か（実験屋の前提）

- 信号が希少・バックグラウンドが巨大 → 最適な分離が必要
- 再構成 (tracking/vertex/PF/jet/flavour/ τ /e/ γ /MET) に ML が入り込む
- HL-LHC : pileup増大+計算資源制約 → 精度とスループットの両立が必須
- (古いけど) レビューとコミュニティ展望
 - [Machine learning at the energy and intensity frontiers of particle physics](#)
 - [Deep Learning and its Application to LHC Physics](#)
 - [ML in HEP Community White Paper](#)
- この講演（前半）の狙い：30年史 → 代表手法（SoTAの系譜）→ 実験屋が嫌う点
 - 後半は、わたしの研究の紹介をします



ざっくり年表：LHC中心の流れ

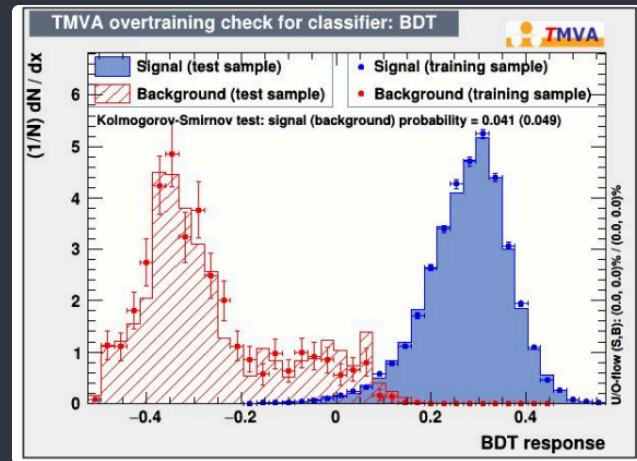
- 1988頃：NN導入（当初は懐疑→徐々に定着）
- 2000年代：BDT+ツール化（TMVA）でMVAが標準運用
 - [Neural Networks in High Energy Physics \(1992\)](#)
 - [BDT as an Alternative to ANN \(2004\)](#)
 - [TMVA \(2007\)](#)
 - [Artificial neural networks in high-energy physics \(2008\)](#)
- 2010年代後半：「入力表現」を低レベルへ押し下げ
 - [Jet-Images \(2015\)](#)
 - [DeepCSV/Heavy-flavour jets in CMS \(2017\)](#)
 - [CaloGAN \(2017\)](#)
- キーワード：性能向上の源泉は「表現（representation）」と「対称性・幾何の取り込み」、ただし導入コストは「検証と系統」
 - 2020年代：GNN/Transformer（幾何・対称性）+Generative（FastSim）が二大潮流
 - [ParticleNet \(2019\)](#)
 - [ParT \(2022\)](#)
 - [EFN/PFN \(Komiske/Metodiev/Thaler, 2018\)](#)
 - [CaloDiffusion \(2023\)](#)
 - [CaloChallenge結果 \(2024\)](#)

物理変数+尤度／線形判別（古典的な分離）

高エネルギー物理屋の限界

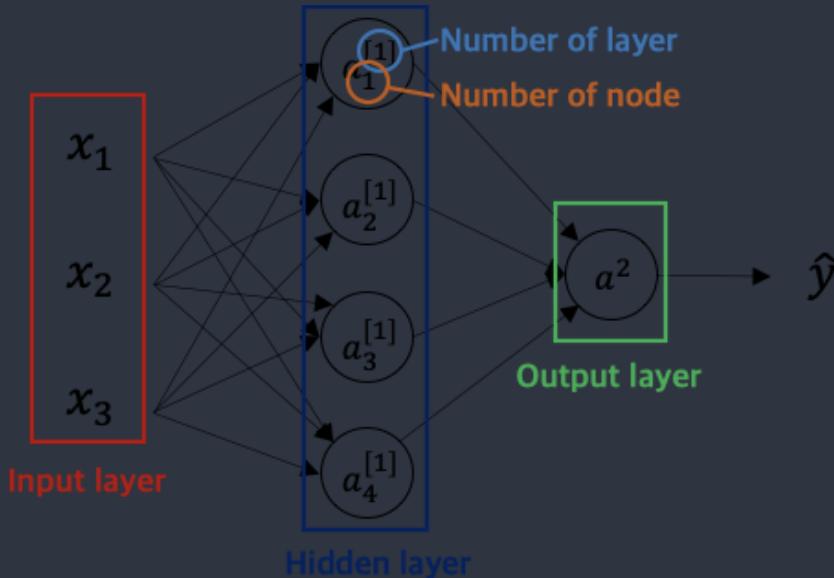
- 入力：人間が設計した高レベル変数(実験屋の限界)
 - 質量・角度・substructure・IP/SV要約など
- 強み：解釈・検証・系統の追跡が容易（実験屋の「安心感」）
- 弱み：高次元相関を取り切れない／変数設計が上限を決める
- 典型的にBoosted Decision Tree(BDT)が人気だった
 - 入力を適当に分けて信号と背景事象を分けるだけなのでわかりやすい
 - モンテカルロ(MC)をたくさん作れば精度出る
- 他にも、SVMなどの古き良き手法も使われていた

多変調解析ツール(TMVA)



- [TMVA \(Höcker et al., 2007\)](#)
- [TMVA overview \(Voss et al., 2009\)](#)

浅いNN (MLP) – 90年代の「最初のNNブーム」



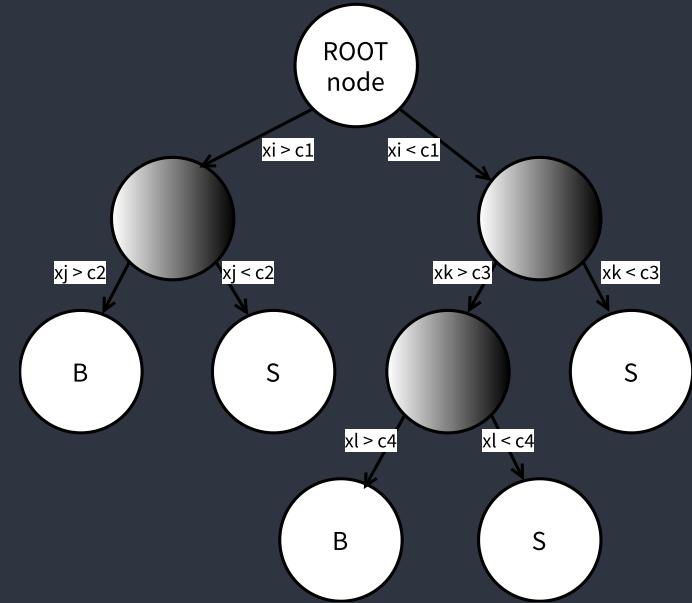
- 入力：高レベル変数（tabular）を中心
- 論点：ブラックボックス性／過学習／MC依存（教師ありの宿命）
- 位置づけ：NNは「昔からある」→ただし当時は表現力と運用が制約
- この規模のNNは意外と過学習しなかったので、サンプル数が少ないときに
- 代表レビュー／チュートリアル
 - [Neural Networks in High Energy Physics \(Peterson, 1992\)](#)
 - [Artificial neural networks in high-energy physics \(Teodorescu, 2008\)](#)
- 実験屋向け強調点：NN導入期から「トリガー応用」と「確率解釈」が意識されていた（ただし実装・検証が重い）

BDT+TMVA – 2000年代の‘標準兵器’と運用標準化

- BDT：性能が強く、ハイパラが直感的、過学習管理がしやすい
- TMVA：同一枠組みで学習・比較・適用 → 解析運用のコストを下げた
- 影響：MVAが「一部の職人技」から「広く共有可能な標準手順」へ
- 代表文献

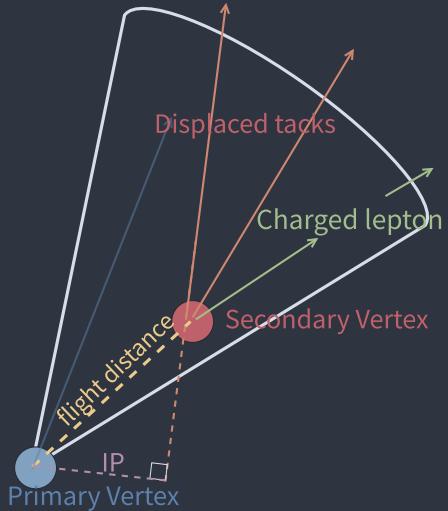
- [BDT as an Alternative to ANN \(Roe et al., 2004\)](#)
- [BDT studies \(Yang/Roe/Zhu, 2005\)](#)
- [TMVA \(Höcker et al., 2007\)](#)
- [TMVA overview \(Voss et al., 2009\)](#)

- TMVAでは、BDT以外にもいろいろな多変量解析が可能
 - よくあるのは、同じサンプル、同じ変数で、色々なアルゴリズムで訓練した結果を比較して最も良いものを採用
 - だいたいBDTが良い場合が多い、かつ、実験屋の脳みそできりぎり
 - TMVAを作ったアンドレアスホッカーはATLAS実験のボス



Deep b-tag (DeepCSV/DeepJet、ATLAS DL1/DL1r)

Bottom Quark Decay



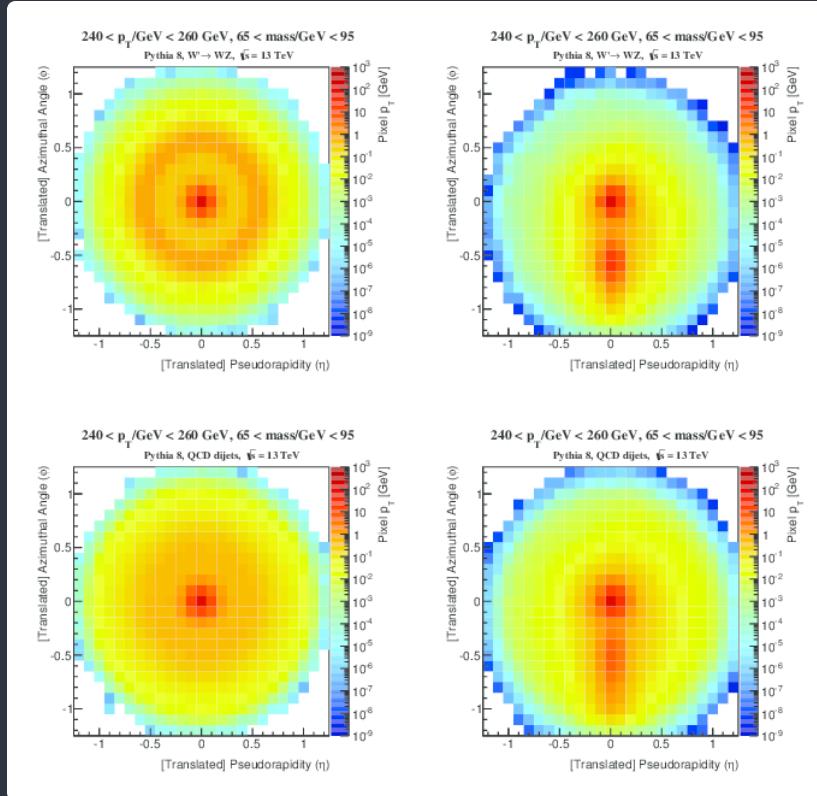
特徴

- Displaced vertex (secondary vertex) from primary vertex due to its long life ($\sim 1.5\text{ps}$)
- Large B-hadron mass
- Large impact parameters (d_0)
- Semi-leptonic e/μ decay of B-hadron ($\sim 40\%$ total B hadron decays)

- 入力：track/vertex/neutral候補などの低レベル情報を大量投入
- 強み：表現を学習し、人間の要約ボトルネックを突破（性能の段差）
- 実験屋の要点：校正（scale factor）とData/MC差の管理が核心

- CMS/ATLASの代表
 - [CMS heavy-flavour jets \(DeepCSV含む, 2017\)](#)
 - [CMS BTV-20-001 公開ページ \(DeepCSV/DeepJet定義\)](#)
 - [ATLAS flavour-tagging Run 2 \(DL1/DL1r, 2022\)](#)
- 実験屋向けメモ：性能だけでなく「校正可能」「安定運用」「系統の閉じ方」が採用の条件

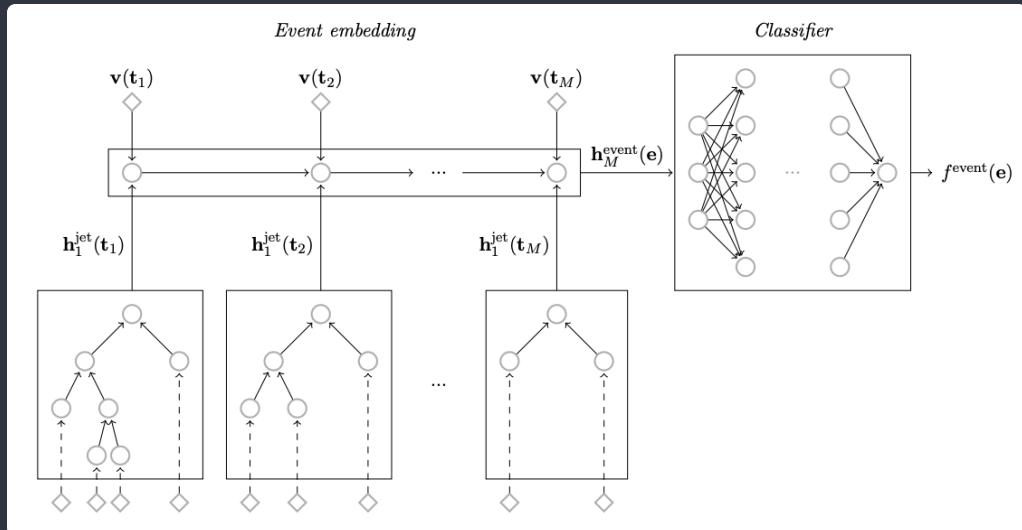
Jet Images + CNN (検出器=カメラの比喩)



- jetを2D画像（ η - ϕ 上のエネルギー堆積など）として扱う
 - RGBの値として、荷電粒子トラックやクラスターのエネルギー
 - 最も高いピクセルの x を 0 とするように回転すると精度が上がる
- 強み：CV資産を転用しやすい／可視化しやすい
- 弱み：幾何・分解能・pileup・前処理に敏感（表現が恣意的になりがち）
 - ジェットの構成要素(トラック or クラスター)はスペースなので疎な画像になる
 - あんまりできること少ない → モデルアーキテクチャの探索になりがち
- 代表論文
 - [Jet-Images — Deep Learning Edition \(2015/2016\)](#)

- CVの発展に伴って良くなっていたが、画像に一度変換する冗長性と情報損失で最近は使われない
- 次への橋渡し：「順序なし集合」「幾何そのもの」を扱う方向 (Sets/GNN/Transformer) へ自然に接続

系列・木構造(RNN/LSTM)構成要素を言語化する前段



- constituentsの列 (ordering必要) やイベント中の粒子を直接モデル化
- 強み: 可変長を自然に扱える／低レベル情報を圧縮しやすい
- 弱み: 順序付けの恣意性 (列) ／計算コスト (深い木)
 - 再起モデルなので学習安定性や記憶力やばい
- 代表論文 (jet構造を“言語/構文木”に見立てる)
 - [QCD-aware Recursive Neural Networks \(Louppe et al., 2017\)](#)
 - [LSTM with jet constituents for boosted top tagging \(Egan et al., 2017\)](#)

- 少し研究されたけど、扱いづらいのですたれた
- 実験屋への一言: この流れが「集合 (permutation)」「注意機構 (global相互作用)」へ発展する

Deep Sets / EFN・PFN(集合として扱う、IRC安全も議論)

DeepSets

Invariantの例: 総和は?

$$f[1 \ 2 \ 3 \ 4] = [10]$$

$$f[4 \ 3 \ 2 \ 1] = [10]$$

$$f[4 \ 2 \ 1 \ 3] = [10]$$

近い値でなく、全く同じ値が保証

Equivariantの例: 仲間はずれは?

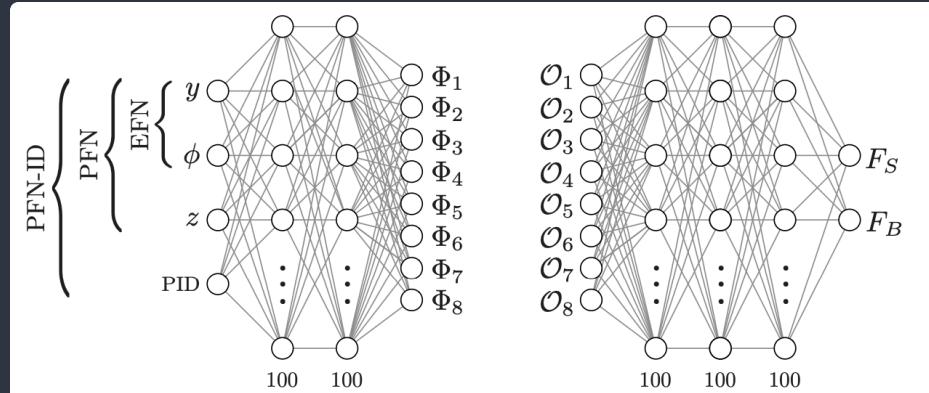
$$f[\smiley \ \smiley \ \frowny \ \frowny] = [0.15 \ 0.1 \ 0.05 \ \mathbf{0.8}]$$

$$f[\smiley \ \smiley \ \frowny \ \smile] = [0.15 \ 0.1 \ \mathbf{0.8} \ 0.05]$$

$$f[\frowny \ \frowny \ \smiley \ \smiley] = [\mathbf{0.8} \ 0.05 \ 0.1 \ 0.15]$$

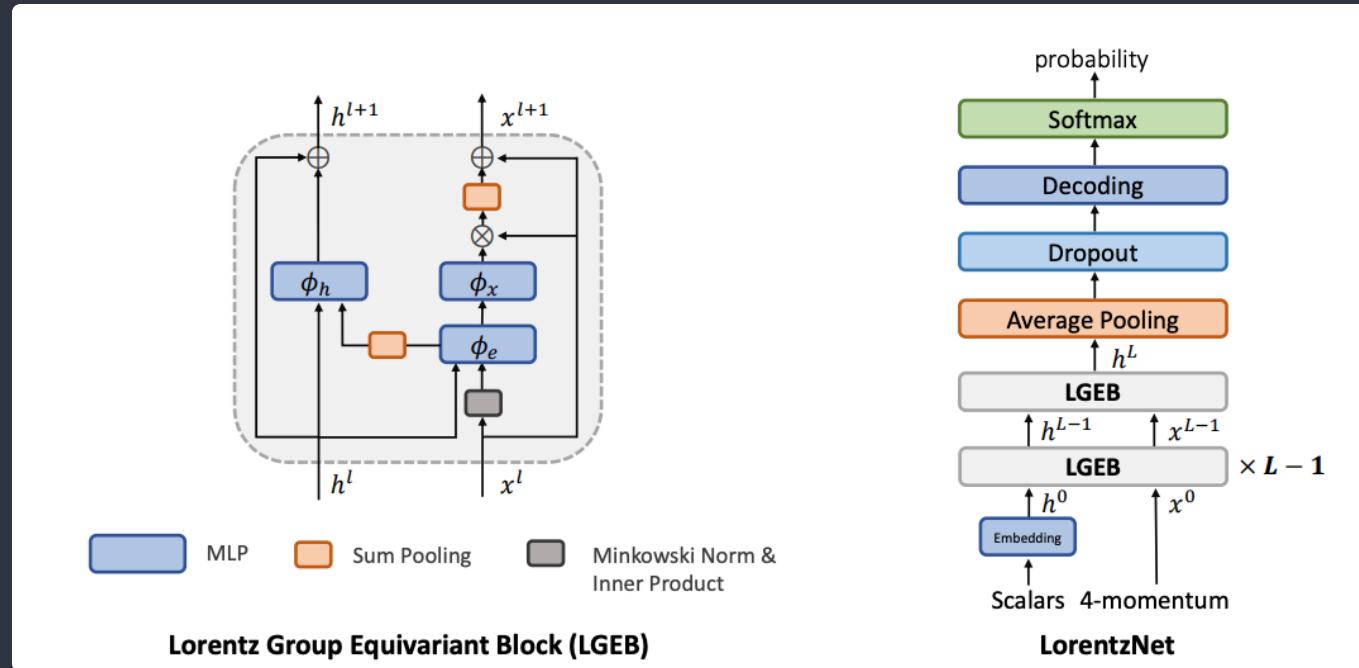
近い値でなく、全く同じ値が保証

Energy Flow Network



- 順序に関する演算を使わない → 和とか最大値とかだけ使う
 - equivariantとinvariantの2種類がある
- event/jetは本質的に「可変長・順序なしの集合」
- permutation invariance を構造で担保（設計思想がHEP向き）
- EFN：赤外・共線安全（IRC safe）を“構造で”満たす方向性
- 代表論文
 - Deep Sets (Zaheer et al., 2017)
 - Energy Flow Networks / Particle Flow Networks (Komiske et al., 2018)
- 典型的な説明図：粒子ごとの埋め込み→総和→イベント表現（「対称性を壊さない」）

LorentzNet (Lorentz群等価GNNによるJet Tagging)

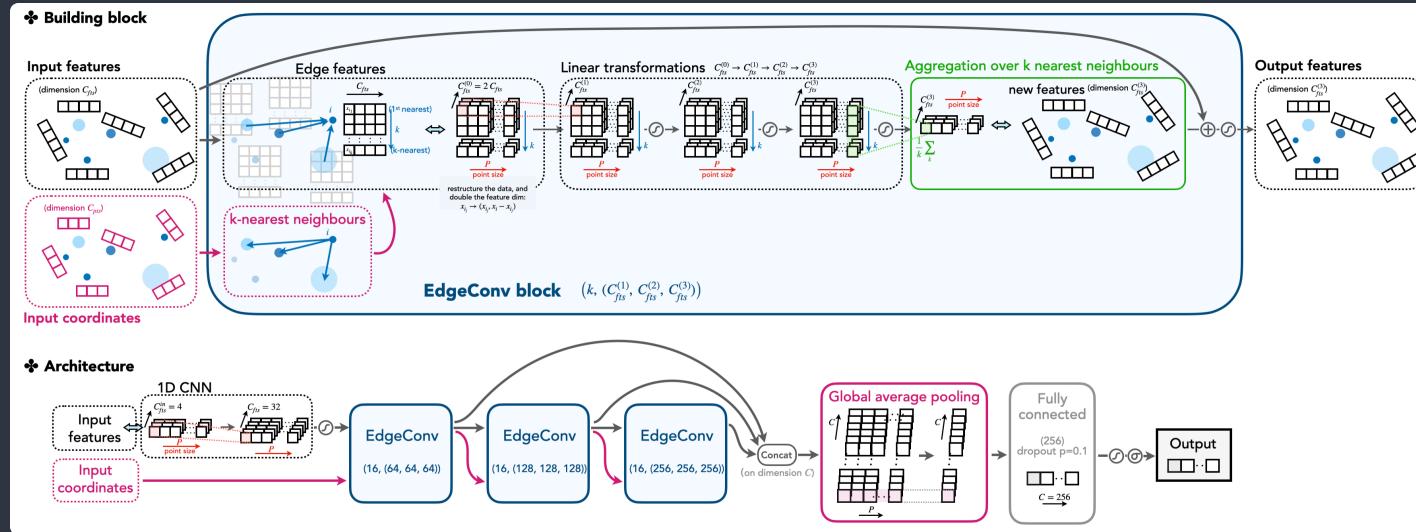


- Lorentz変換に依らない分類をモデルとして保証
- 入力：各粒子の4運動量+粒子ID等のスカラー特徴
- 形式：粒子集合をグラフとして粒子ペア情報を集約
- 典型タスク：top tagging / quark-gluon tagging
- [LorentzNet \(arXiv\)](#)
- [比較：LGN \(Lorentz Group Network\)](#)
- [比較：ParticleNet \(particle cloud\)](#)

- 既存のLorentz等価モデルは「高次テンソルの解析的構成」が重くなりがち
- LorentzNetは Lorentz不变量（Minkowski内積）を用いた attention を中核にして効率化
- 実験屋メリット：対称性の帰納バイアスで、分布シフト（ブースト）や少数データに強くなる可能性

GNN / Particle Clouds (ParticleNet、Trackingにも波及)

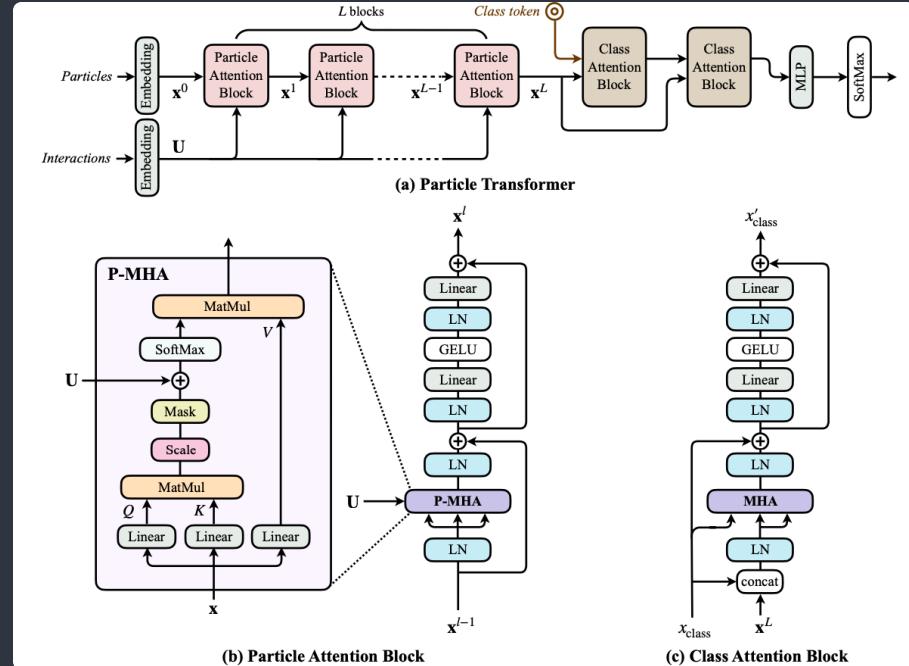
実験屋さんはグラフNNが大好き



- jet (SoTAの代表例)
 - [ParticleNet \(Qu/Gouskos, 2019\)](#)
- tracking (パイプラインの代表例)
 - [Novel deep learning methods for track reconstruction \(Farrell et al., 2018\)](#)
 - [GNNs for particle reconstruction in HEP detectors \(Ju et al., 2020\)](#)
- 実験屋向け強調点：粒子の相互作用はそのままグラフなので表現として素直、われわれのデータに

- jet : 点群やグラフとして、近傍関係も学習
- tracking : hitsがnode,結合をedge,edge分類
- 強み : 幾何・局所性・可変サイズを自然に表現
- 弱み : スケールしない, 計算効率が非常に悪い

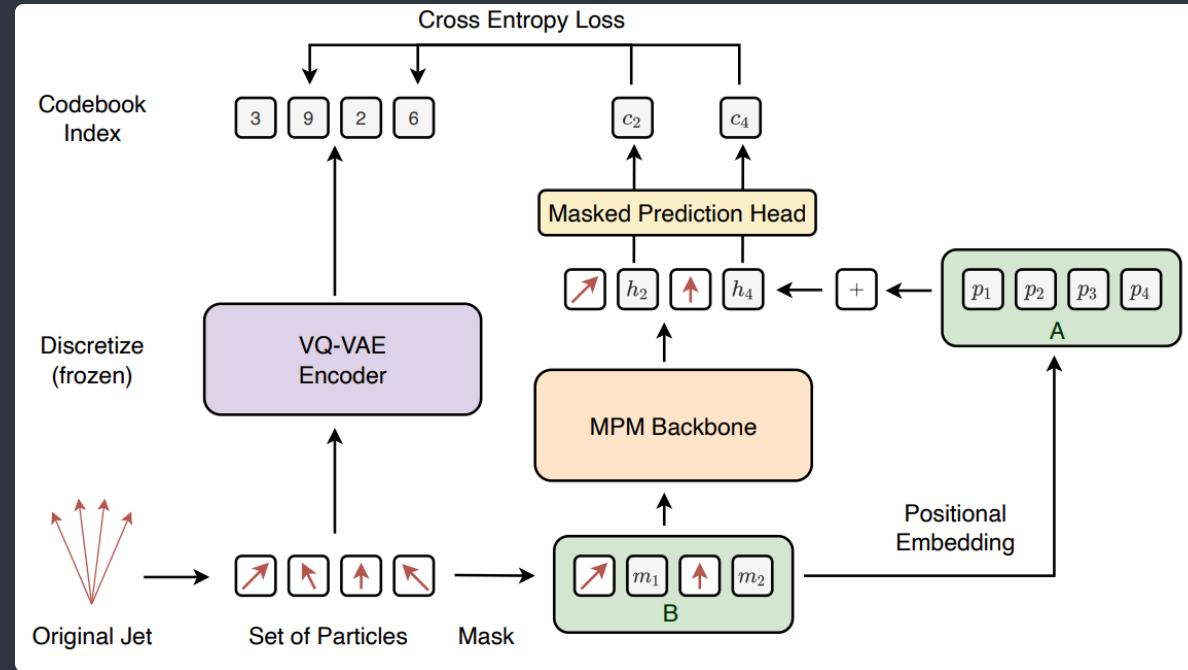
Transformer (Part) – attentionで全体相互作用



- constituents間の長距離相関を attention で扱う（順序付け問題を緩和）
 - ジェットで重要な2点相関変数などを'attention bias'としている
- pairwise相互作用を attention に注入する設計で性能を押し上げ
- 強み：注意機構による柔軟な表現と精度向上
- 弱み：計算量・レイテンシ→実装（オンライン）では軽量化が鍵
 - 生成をするわけではなくて, Encoder型(BERT)同様
- 代表論文
 - [Particle Transformer for Jet Tagging \(ParT, 2022\)](#)
 - [ParticleNet \(比較対象, 2019\)](#)

- 次章（後半）への接続：Transformerを「ジェット言語化」「生成」「不確かさ推定」へ拡張する流れ

Masked Particle Modeling



- Track(or cluster)を量子化(VQ-VAE), それをBERT
- 強み: 生成が可能, 自己教師学習
- 弱み: ベクトル量子化がネック, 学習不安定
- 代表論文
 - [Masked Particle Modeling on Sets](#)

- 自己教師学習が原理的に可能であって, それらをファインチューニングして個別のタスクに転移
- 人間のラベルによる学習よりも改善する傾向がある(しないものもある)
- 言語モデルのように自己回帰学習だけど, 文法や語順は本質的には不定で位置エンコードは???論文でも言及あり
 - 論文タイトルにもあるように, **Sets**に対するTransformerの応用 → 順序による予測性が低い

FastSim-LHC シミュレーションのつらいところ

material height:250pt

- LHCのコンピューティングの7-8割がシミュレーションで消費
 - FullSim (Geant4) : 高精度だがCPUが重い
 - FastSim : ボトルネックを置き換えて統計を稼ぐ
 - 再構成チェーンに接続し、FullSim/データと比較してチューニング
- 代表（実験の枠組み）
 - [ATLAS: AtlFast3 \(arXiv:2109.02551\)](#)
 - [CMS: Fast Simulation of the CMS Experiment \(Giammanco, 2014\)](#)
 - [CMS: Recent Developments in CMS Fast Simulation \(arXiv:1701.03850\)](#)
- FastSimは「生成できる」だけでは不十分で、解析物理量・系統・運用（スループット）まで含めて合格が必要
- ここから先は「GAN→Flow→Diffusion→Benchmark→production」の流れで整理する

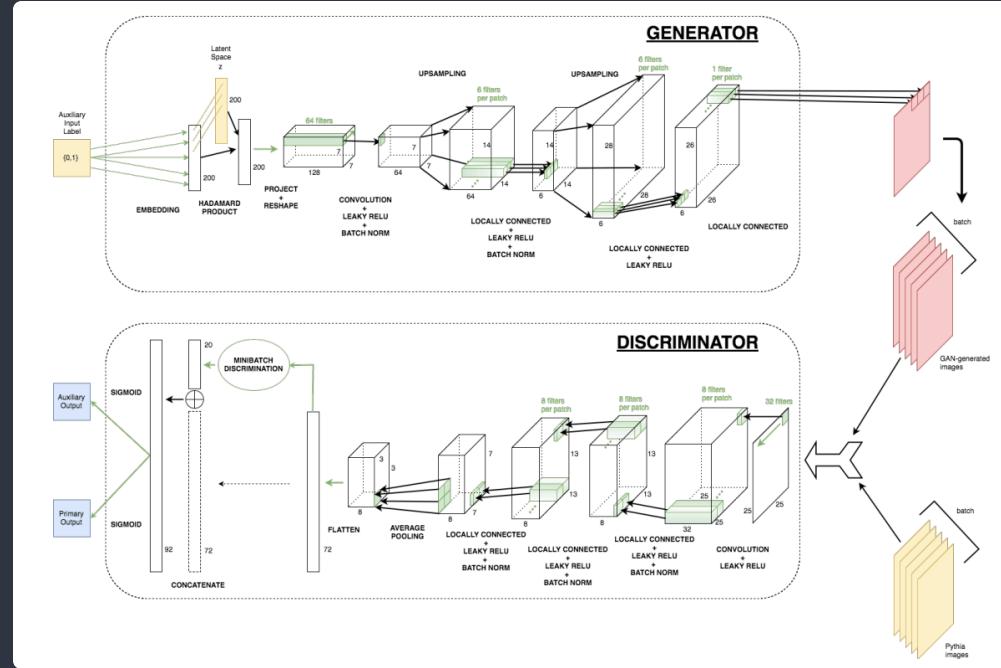
ATLAS- AtlFast3 (FastCaloSim × FastCaloGAN の production 設計)

	Inner Detector	Calorimeters		Muon Spectrometer	
Electrons Photons	Geant4	FastCaloGAN V2 $E_{kin} < 8 \text{ GeV} \& \eta < 2.4,$ Except [0.9 < \eta < 1.1, 1.35 < \eta < 1.5]	FastCaloSim V2 $E_{kin} > 16 \text{ GeV} \& \eta < 2.4,$ All E_{kin} & [0.9 < \eta < 1.1, 1.35 < \eta < 1.5, \eta > 2.4]		
Charged Pions Kaons		Geant4 Pions: $E_{kin} < 200 \text{ MeV}$ Other hadrons: $E_{kin} < 400 \text{ MeV}$	FastCaloSim V2 $E_{kin} < 4 \text{ GeV} \& \eta < 1.4,$ $E_{kin} < 1 \text{ GeV} \& \eta < 3.15$	FastCaloGAN V2 $E_{kin} > 8 \text{ GeV} \& \eta < 1.4,$ $E_{kin} > 2 \text{ GeV} \& 1.4 < \eta < 3.15,$ All E_{kin} & $ \eta > 3.15$	Muon Punchthrough + Geant4
Baryons			FastCaloGAN V2		
Muons		Geant4			

- AtlFast3 : パラメトリック手法とML (GAN等) を組み合わせた fast simulation
- Run 3 production での使い分け (粒子種・エネルギー・ η 領域で FastCaloSim / FastCaloGAN を切替)
- HL-LHCに向けて、Fast chain (中間フォーマット削減・CPU/ディスク最適化) まで含めた議論
- 代表
 - [AtlFast3: the next generation of fast simulation in ATLAS \(arXiv:2109.02551\)](#)
 - [AtlFast3: Fast Simulation in ATLAS for LHC Run 3 and Beyond \(ATL-SOFT-PROC-2025-034\)](#)
 - [ATLAS Fast Chain for HL-LHC \(ATL-SOFT-PROC-2025-042\)](#)

- 研究モデルをそのまま入れるのではなく「どの相空間で何を使うか」を設計しているのが実験的に重要
- 速度・精度に加えて、validation と systematic の閉じ方が採用条件

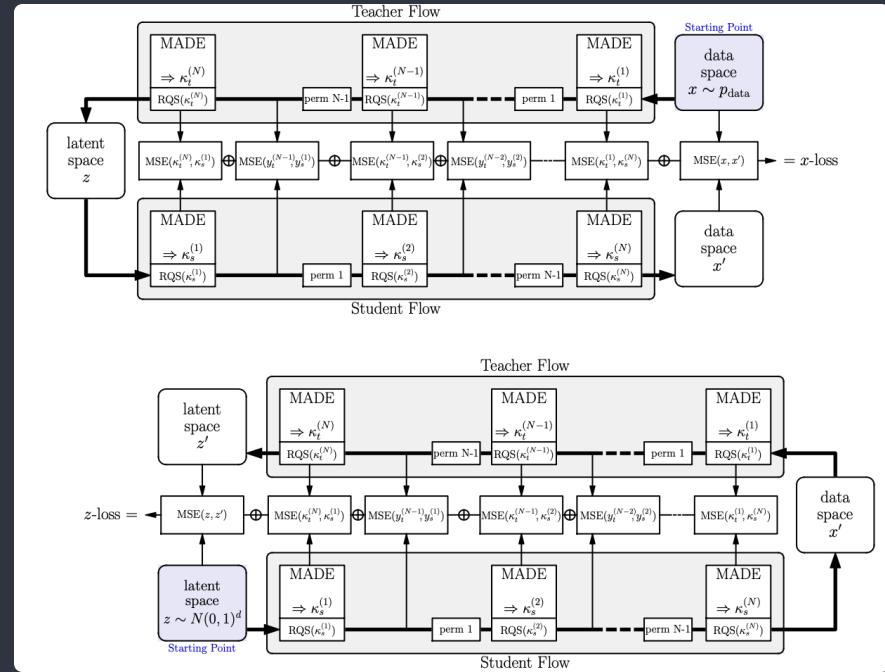
GAN 時代：CaloGAN → 実験内 FastCaloGAN 系へ



- GANは「最初の成功例」として重要だが、LHCで勝つには評価・外挿・系統まで含めて設計が必要
- ここから「評価可能性 (likelihood/metrics)」を強めるFlowや「高忠実度」へ寄るDiffusionに分岐

- GAN：サンプリングが高速、「桁で速くする」インパクト
- 課題：学習安定性、特徴崩壊、評価指標
- 実験側：FastCaloGANのように、AtlFast3に統合
- 代表
 - [CaloGAN \(arXiv:1705.02355\)](#)
 - [ATLAS: AtlFast3 \(arXiv:2109.02551\)](#)
 - [ATLAS Run 3: FastCaloGAN V2を含む構成](#)
 - [LHCb: Fast Calo Simulation](#)

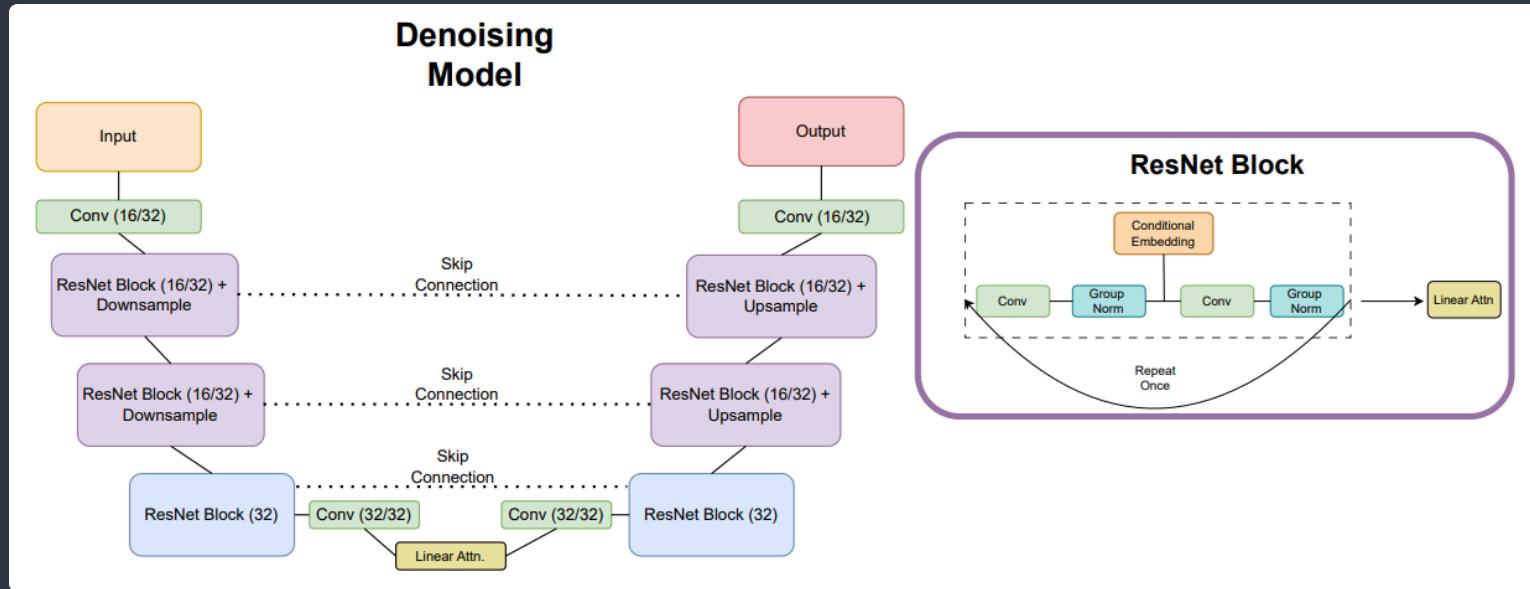
Normalizing Flow : CaloFlow / 蒸留 / 高粒度への拡張



- Flow : 尤度が扱える, 学習が安定, モデル選択が容易
- ただし, サンプリングが遅い設計もある → 蒸留で加速する流れ
- 高粒度化 : メモリ制約を避ける工夫でスケールさせる
- 代表
 - [CaloFlow \(arXiv:2106.05285\)](#)
 - [CaloFlow II \(arXiv:2110.11377\)](#)
 - [iCaloFlow \(Inductive CaloFlow, arXiv:2305.11934\)](#)
 - [CaloChallenge Dataset 1向けCaloFlow \(arXiv:2210.14245\)](#)

- Flow系は「評価可能性」と「学習の工学的安定性」で実験屋が扱いやすい
- 一方で、最終的に欲しいのは“解析物理量の再現”なので、ベンチマーク+下流タスク評価がセット

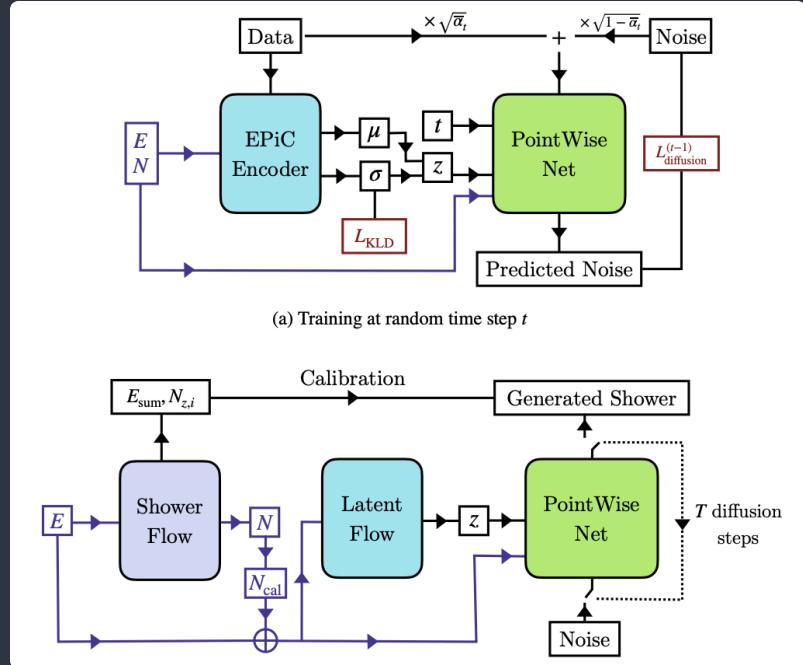
Diffusion/Score : 高忠実度→蒸留で単発生成へ



- 高次元で高精度だがサンプリングが重い
- 対応策：蒸留, consistency, 構造設計で高速化
- 幾何対応：irregular geometry を扱う
- 代表
 - [CaloScore](#)
 - [CaloScore v2](#)
 - [CaloDiffusion \(GLaM\)](#)
 - [Graph-based diffusion model](#)

- 「Diffusionは遅い」を蒸留で潰していくのが2023以降の主戦場
- 実験導入では、速度だけでなく“外挿・系統・更新”がセットで問われる

幾何非依存・点群・Transformer (高粒度/将来を意識)

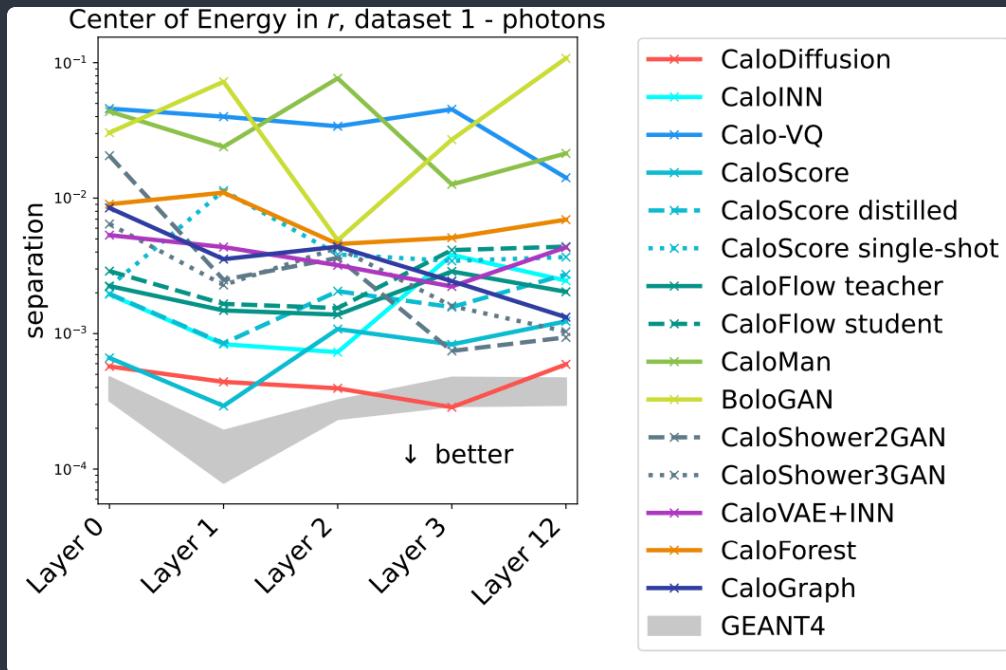


- 高粒度では、voxel表現が巨大・疎になる → 点群が自然
- 幾何非依存：detector geometry の変更に強い設計を狙う
- 可変長系列として生成し、基盤モデル方向の議論にも接続
- 代表

- [CaloClouds \(arXiv:2305.04847\)](#)
- [CaloClouds II \(arXiv:2309.05704\)](#)
- [CaloClouds3 \(arXiv:2511.01460\)](#)
- [OmniJet- \$\alpha\$ C \(arXiv:2501.05534\)](#)

- LHC本番で効くのは「HL-LHCの高pileup+高粒度」なので、この方向性は今後の伸びしろ
- ただし production には、検出器条件変動（劣化・校正）まで含めた設計が必要

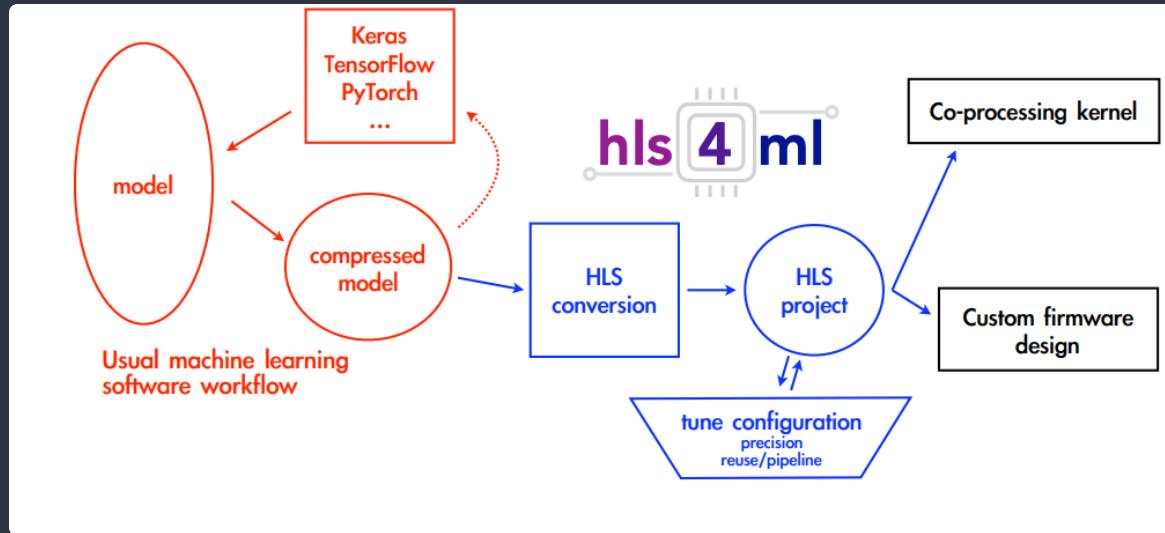
CaloChallenge 2022 (評価指標・速度・モデルサイズ)



- VAE/GAN/Flow/Diffusion/Conditional Flow Matchingなど幅広い提出を比較
- 指標：1D分布差だけでなく、分類器AUC,KPD/FPD、生成時間、モデルサイズ等
- データ：公開データで再現可能性が高い
- 一貫して、CaloDiffusionやCaloINN(Flow系)が高いパフォーマンス
- 代表
 - [CaloChallenge結果 \(arXiv:2410.21611\)](#)
 - [CaloChallenge公式ページ \(データ/提出物リンク\)](#)
 - [CaloChallenge Dataset 1 \(Zenodo\)](#)
 - [CaloChallenge Dataset 3 \(Zenodo\)](#)

- 研究が「何でもあり」になった結果、評価が本質的な研究テーマになった (CaloChallengeはその整理)
- 実験屋的には「下流解析での差」「外挿」「系統」をどこまでベンチに取り入れるかが次の焦点

オンライン実装 (FPGA・低レイテンシ推論)



- 実験中に動作するオンラインのモデルをどう実装するか
- L1/HLT：レイテンシ・資源制約の中でMLを動かす必要
- hls4ml：学習済みモデル→HLS→FPGA実装のワークフロー
- BDTもFPGAで実装（100 ns級レイテンシを志向）
- 代表
 - [hls4ml \(DNN推論, Duarte et al., 2018\)](#)
 - [BDTのFPGA推論 \(Summers et al., 2020\)](#)

- オフラインで特化モデルを学習 → 蒸留,量子化など軽量モデルをFPGA(HLS)でファームウェアにする
 - おそらくここで強力なモデルが動くならゲームチェンジャー
- 典型スライド案：オフラインSoTA → オンライン制約下SoTA（量子化・蒸留・構造最適化）という対比

実験屋が気にするポイント

- 検証：control region／データ駆動でクロスチェック可能か
- 系統：入力のData/MC差、tagger出力、再学習の影響をどう閉じるか
- 再現性：学習データ・コード・モデルの保存と追跡（モデル更新のガバナンス）
- 運用：推論速度、メモリ、オンライン制約（trigger/HLT）、保守性
- 次（後半）への接続：Transformer・拡散モデルを「実験屋の採用条件」に合わせてどう料理するか（不確かさ・制約・解釈）
- 参照（全体像）
 - [ML at the energy/intensity frontiers \(Radovic et al., 2018\)](#)
 - [Deep Learning and its Application to LHC Physics \(2018\)](#)
 - [ML in HEP Community White Paper \(2018\)](#)

One More Thing : My Recent Research

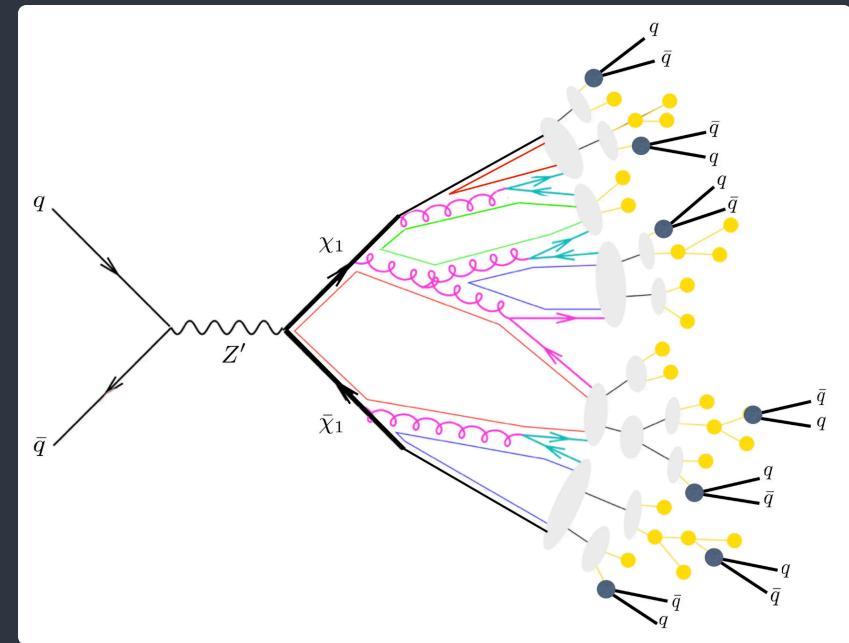
ジェットとは

ジェットはLHCでの主人公

- クォーク, グルーオンはシャワーを出してハドロン化し検出できるのはジェット
- LHC 検出器では荷電粒子のトラックと中性カロリメータクラスタとして検出
- これらをクラスタリングしてジェットを再構成する

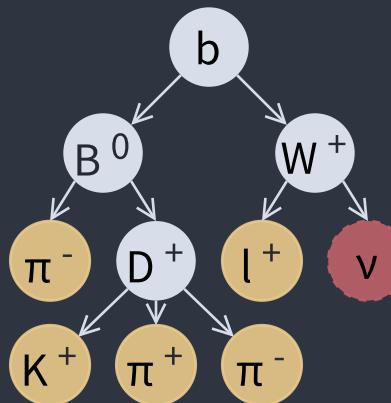
なぜジェットが重要なか

- LHC はハドロン衝突型 \Rightarrow 観測される量の多くがジェット
- 新粒子探索や精密測定では、基になる過程（例： $H \rightarrow b\bar{b}$ ）の同定が鍵。
 - ジェットの起源が分かれば物理過程の推定が変わる
 - ジェットエネルギーが分かれば測定精度が向上する
- ジェットの途中状態は重要だが、SoTAモデルの多くは最終的な要約だけを使う
 - 「出演者一覧から犯人を当てる」のではなく、経過と証拠を丸ごと使う



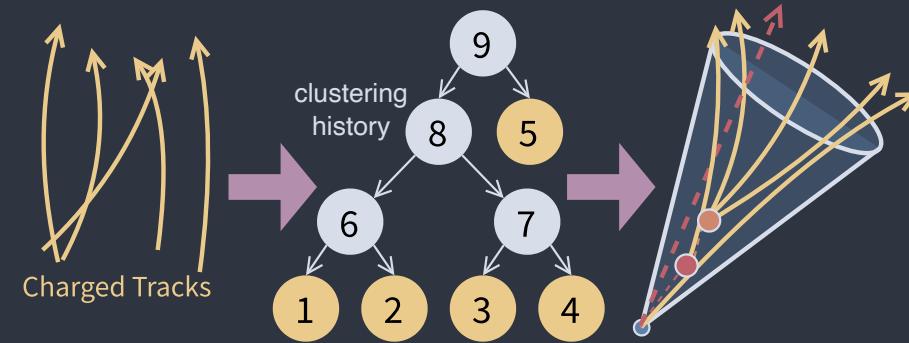
粒子崩壊とジェット再構成

粒子崩壊



- 粒子崩壊は木構造として表せる。
 - 親 \rightarrow 子1 + 子2。保存則と許可された崩壊モードを満たす
 - 簡単のため束縛状態の効果や結合は無視

ジェット再構成



- anti- k_t アルゴリズムでトラック（またはクラスター）を再構成
 - 最小の $d_{ij} = \Delta R_{ij} \min(p_{T,i}^\alpha, p_{T,j}^\alpha)$ の組をマージ
 - 何も残らないか最小距離がしきい値を超えるまで繰り返す
- 構造上、マージ履歴は完全二分木になる

共通点

- どちらも木構造。再構成ジェットの木を真の崩壊の木に翻訳できれば、真の崩壊を復元できる
- Transformerによるtree-to-tree翻訳を提案

言語モデルとは



- 言語モデルはトークン列 (w_1, w_2, \dots, w_T) に確率を与える:

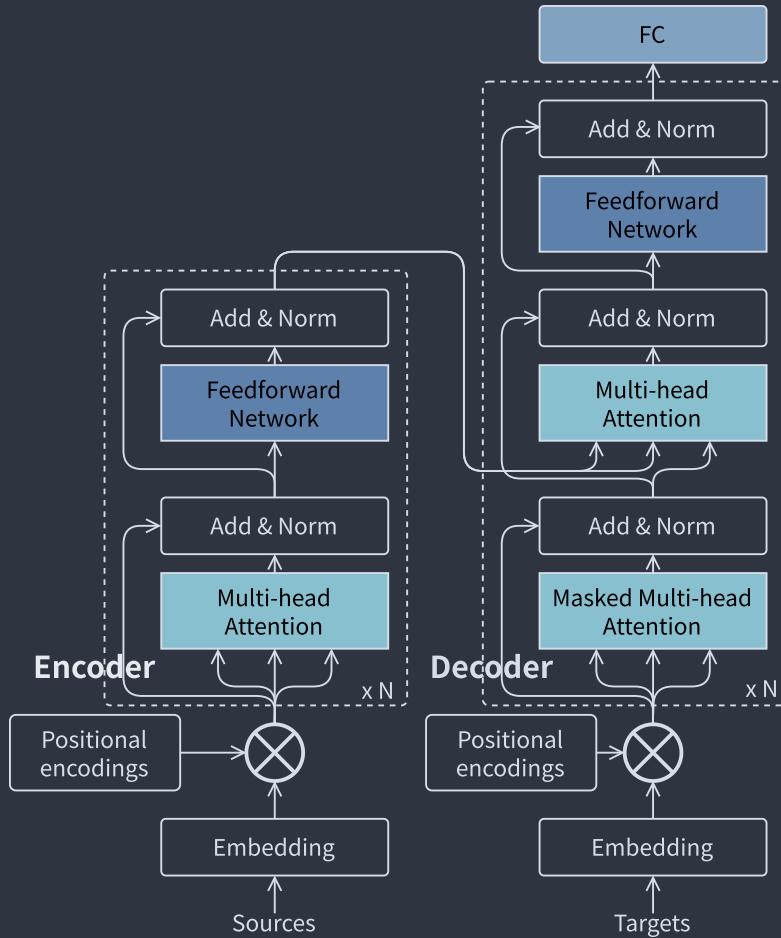
$$P(w_1, w_2, \dots, w_T)$$

- 連鎖律（因果分解）より:
- 因果分解（自己回帰）:**

$$P(w_1, w_2, \dots, w_T) = \prod_{t=1}^T P(w_t | w_1, \dots, w_{t-1})$$

- 次のトークンの条件付き確率を予測

Transformer

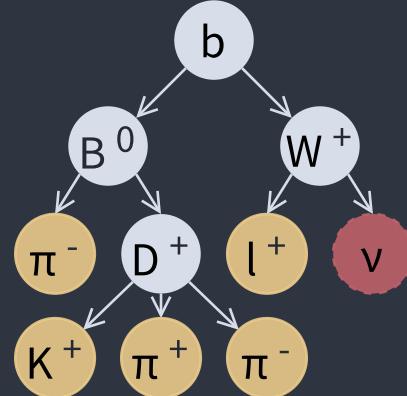


- 現在最も成功しているアーキテクチャ。
 - 入力はトークン（単語/記号の整数 id）。
 - Attention と FFN のブロックを積み重ね、attention により時間方向を並列化。
- Encoder/Decoder は別の役割（例：機械翻訳）
 - Encoder: ソーステキストを潜在表現に写像。
 - Decoder: ターゲットテキストを生成。
- 推論はソース + 過去に生成したトークンを逐次与えるため遅い。
 - KV-cache が過去状態を再利用して計算を軽くする。
- GPT 系はデコーダのみ。本研究もデコーダのみを使用。
 - 入力がトークン ⇒ 木構造をトークン化する必要。

木構造のトークン化

木を平坦化する

- よくある表現:
 - Bracketed:** A(B(D, E), C)
 - Preorder:** A B D # # E # # C # #
 - Edge list:** (A,B) (A,C) (B,D) (B,E)
 - Adjacency list:** A:[B,C]; B:[D,E]; C:[]; D:[]; E:[]
- ここではエッジリストを採用。 b クオーク崩壊の例:
 - $(b, B^0)(b, W^+)(B^0, \pi^-)(B^0, D^-)(W^+, \ell^+)(W^+, \nu)(D^-, K^+)(D^-, \pi^+)(D^-, \pi^-)$



トークン化

- 連続特徴 (float) は符号 + 仮数 + 指数に分解して符号化。
 - 例: $\phi = 0.123 \rightarrow \text{Phi@Sign_P } \text{Phi@M_123 } \text{Phi@E_01}$
- 粒子種 (PDGID) は PDG の規則に沿って桁ごとに分割してトークン化。
 - 例: PDGID = -323 ($K^*(892)^-$) $\rightarrow \text{PDG@ANTI } \text{PDG@FLV_SU } \text{PDG@J_3 } \text{PDG@END}$
- その他: **bos** (begin-of-sequence) などの特殊トークン。
- 語彙サイズは約 10k。

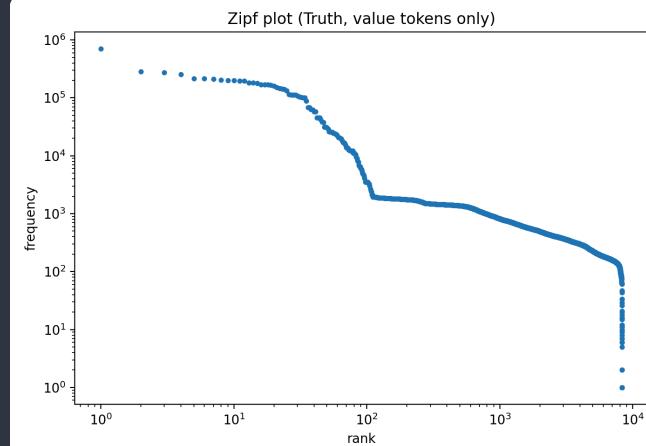
トークン分布

```
BOS RECO ROOT pt@M341 pt@E+1 eta@S_N eta@M576 eta@E-1 phi@S_N phi@M530 phi@E-1 energy@M404 energy@E+1 EOE
IDX@I+0 IDX@I+1 pid@I-1 charge@I+0 pt@M333 pt@E+1 eta@S_N eta@M566 eta@E-1 phi@S_N phi@M527 phi@E-1 energy@M392 energy@E+1 z
IDX@I+0 IDX@I+16 pid@I+3 charge@I+0 pt@I851 pt@E-1 eta@S_N eta@M927 eta@E-1 phi@S_N phi@M638 phi@E-1 energy@M124 energy@E+0 z
IDX@I+1 IDX@I+2 pid@I-1 charge@I+0 pt@M311 pt@E+1 eta@S_N eta@I541 eta@E-1 phi@S_N phi@M520 phi@E-1 energy@M361 energy@E+1 z
IDX@I+1 IDX@I+15 pid@I+5 charge@I+0 pt@M219 pt@E+0 eta@S_N eta@M882 eta@E-1 phi@S_N phi@M638 phi@E-1 energy@M310 energy@E+0 z
IDX@I+2 IDX@I+3 pid@I-1 charge@I+0 pt@M298 pt@E+1 eta@S_N eta@M526 eta@E-1 phi@S_N phi@M513 phi@E-1 energy@M343 energy@E+1 z
IDX@I+2 IDX@I+14 pid@I+2 charge@I+0 pt@I131 pt@E+0 eta@S_N eta@M841 eta@E-1 phi@S_N phi@M667 phi@E-1 energy@M181 energy@E+0 z
IDX@I+3 IDX@I+4 pid@I-1 charge@I-1 pt@M258 pt@E+1 eta@S_N eta@M488 eta@E-1 phi@S_N phi@M530 phi@E-1 energy@M290 energy@E+1 z
IDX@I+3 IDX@I+13 pid@I+4 charge@I+1 pt@I402 pt@E+0 eta@S_N eta@M754 eta@E-1 phi@S_N phi@M405 phi@E-1 energy@M255 energy@E+0 z
IDX@I+4 IDX@I+5 pid@I-1 charge@I-1 pt@M211 pt@E+1 eta@S_N eta@M497 eta@E-1 phi@S_N phi@M555 phi@E-1 energy@M238 energy@E+1 z
IDX@I+4 IDX@I+12 pid@I+5 charge@I+0 pt@I471 pt@E+0 eta@S_N eta@M441 eta@E-1 phi@S_N phi@M418 phi@E-1 energy@M518 energy@E+0 z
IDX@I+5 IDX@I+6 pid@I-1 charge@I+0 pt@M188 pt@E+1 eta@S_N eta@I487 eta@E-1 phi@S_N phi@M566 phi@E-1 energy@M211 energy@E+1 z
IDX@I+5 IDX@I+11 pid@I+4 charge@I-1 pt@I229 pt@E+0 eta@S_N eta@M580 eta@E-1 phi@S_N phi@M461 phi@E-1 energy@M269 energy@E+0 z
IDX@I+6 IDX@I+7 pid@I-1 charge@I+0 pt@M179 pt@E+1 eta@S_N eta@M486 eta@E-1 phi@S_N phi@M561 phi@E-1 energy@M201 energy@E+1 z
IDX@I+6 IDX@I+10 pid@I+3 charge@I+0 pt@I950 pt@E-1 eta@S_N eta@M503 eta@E-1 phi@S_N phi@M669 phi@E-1 energy@M107 energy@E+0 z
IDX@I+7 IDX@I+8 pid@I+5 charge@I+0 pt@M124 pt@E+1 eta@S_N eta@M495 eta@E-1 phi@S_N phi@M538 phi@E-1 energy@M140 energy@E+1 z
IDX@I+7 IDX@I+9 pid@I+3 charge@I+0 pt@M549 pt@E+0 eta@S_N eta@M465 eta@E-1 phi@S_N phi@M613 phi@E-1 energy@M610 energy@E+0 z
SEP TRUTH ROOT pt@I288 pt@E+1 eta@S_N eta@M445 eta@E-1 phi@S_N phi@M563 phi@E-1 energy@M318 energy@E+1 EOE
IDX@I+0 IDX@I+1 PDG@ANTI PDG@FLV_CS PDG@END PDG@ANTI PDG@FLV_SU PDG@J_3 PDG@END charge@I-1 pt@I104 pt@E+1 eta@S_N eta@M418 e
IDX@I+0 IDX@I+6 PDG@ANTI PDG@FLV_CS PDG@END PDG@ANTI PDG@FLV_SD PDG@END charge@I+0 pt@I184 pt@E+1 eta@S_N eta@M460 eta@E-1 pl
IDX@I+1 IDX@I+2 PDG@ANTI PDG@FLV_SU PDG@J_3 PDG@END PDG@ANTI PDG@FLV_SU PDG@END charge@I-1 pt@I378 pt@E+0 eta@S_N eta@M414 e
IDX@I+1 IDX@I+10 PDG@FLV_SU PDG@J_3 PDG@END PDG@FLV_DD PDG@END charge@I+0 pt@E-1 eta@S_N eta@M420 eta@E-1 phi@S_N phi@M667 pt@E+0 eta@S_N eta@M420 eta@E-1 phi@S_N phi@M667
IDX@I+3 IDX@I+4 PDG@FLV_DD PDG@END PDG@P_PH PDG@END charged@I+0 pt@M954 pt@E-1 eta@S_N eta@M440 eta@E-1 phi@S_N phi@M675 phi@I
IDX@I+3 IDX@I+5 PDG@FLV_DD PDG@END PDG@P_PH PDG@END charged@I+0 pt@M573 pt@E+0 eta@S_N eta@M416 eta@E-1 phi@S_N phi@M622 phi@I
IDX@I+6 IDX@I+7 PDG@ANTI PDG@FLV_SD PDG@END PDG@P_K0L PDG@END charge@I+0 pt@M184 pt@E+1 eta@S_N eta@M460 eta@E-1 phi@S_N phi@M622
ROOT pt@M397 pt@E+0 eta@S_N eta@M525 eta@E-1 phi@S_N phi@M504 phi@E-1 energy@M456 energy@E+0 EOE
IDX@I+0 IDX@I+1 PDG@FLV_DD PDG@J_3 PDG@END PDG@ANTI PDG@FLV_UD PDG@END charge@I-1 pt@M323 pt@E+0 eta@S_N eta@M529 eta@E-1 phi@S_N phi@M529
IDX@I+0 IDX@I+1 PDG@FLV_DD PDG@J_3 PDG@END PDG@FLV_UD PDG@END charge@I+1 pt@M749 pt@E-1 eta@S_N eta@M498 eta@E-1 phi@S_N phi@M498
```

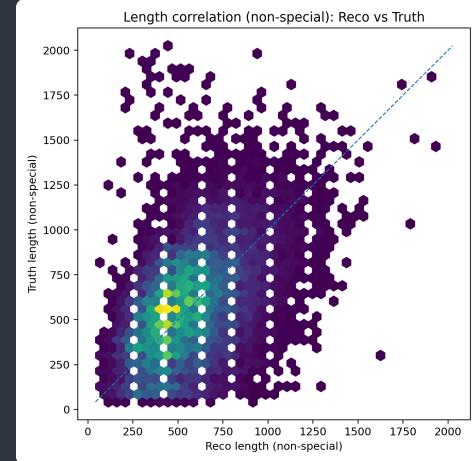
典型的なトークン化例

実務的な指標

- ジェットのトークン列:** 典型的に 500 トークン程度 (uint16 で保存)。
- Zipf 則:** 人工的な言語なので理想的な $1/k$ からずれる → より良いトークン化の余地。
- 系列長:** Truth の方がやや長い (構造+PDG トークン)。



Zipfプロット



文字列長分布 (2D)

学習手順とセットアップ

LM の作法にならう

- Pretraining → Translation → Grammar 制約付き finetuning → Physics RL
- Pretraining: reco/truth 混在列での次トークン予測。
- Translation: reco をプロンプトし, truth を生成。
- Grammar finetuning: 構文制約を満たすよう学習。
- Physics RL: 保存則と崩壊を教える。
 - GRPO: 値値ネット不要の新手法。

強化学習

- モデルが環境と対話し, 出力に基づき報酬を計算。
- 重要なのは 報酬設計 (ここでは物理法則) :
 - 運動量保存: 親と子が保存則を満たせば報酬。
 - 許可された崩壊: PDG に沿う崩壊パターンなら報酬。

データ

- MG5+MC@NLO + Pythia8 + Delphes (CMS 設定)
 - $pp \rightarrow q\bar{q}, gg, Z \rightarrow \tau\tau$
 - ジェットの reco-truth マッチング。あいまいな事例は除外。
 - 2 億ジェット生成。利用可能なのは約 1.2 億。

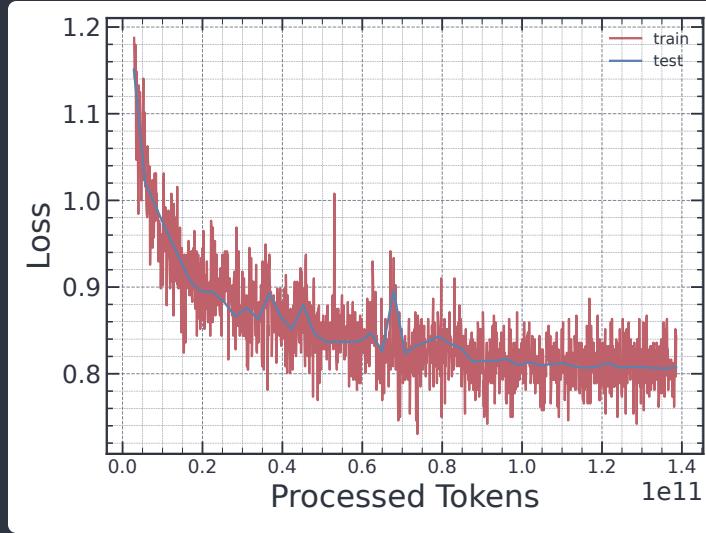
モデル

- LLaMA 系 Transformer
 - 効率的な attention: Grouped-Query Attention
 - ReLU² のスパース FFN
 - u-muP スケーリングでハイパラを転移可能に
- 本研究: 幅 512, 深さ 8 (約 3600 万パラメータ)

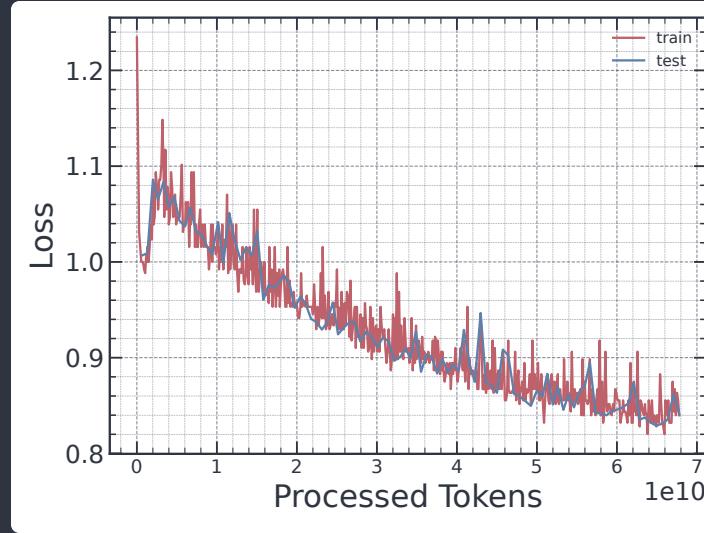
計算資源

- A100 × 8 台でモデル並列+データ並列

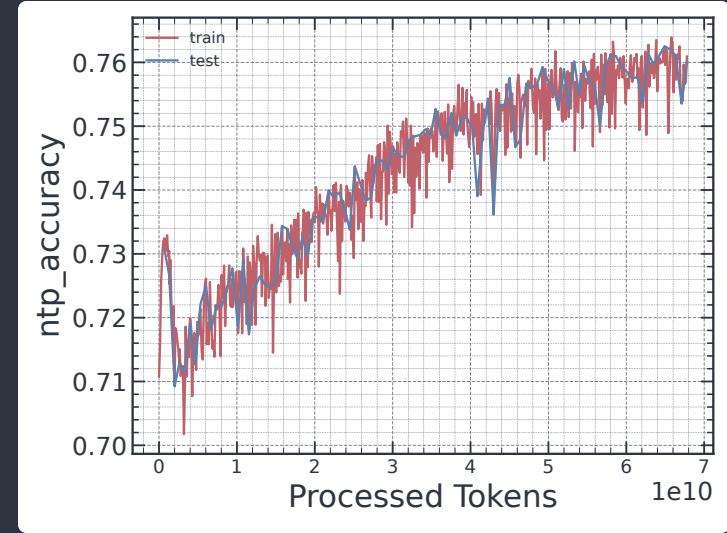
学習の進捗



事前学習の損失



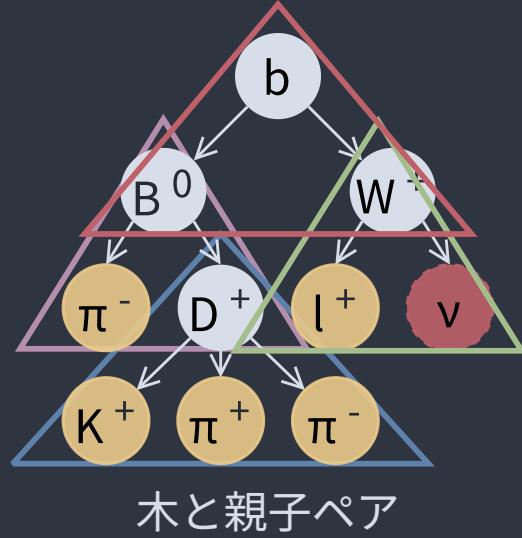
reco→truth 翻訳損失



reco→truth 翻訳精度

- Pretraining と Translation は綺麗に学習しており、モデルがジェット構造を捉えている。
- Translation はまだ伸びそう。継続学習すれば loss がさらに下がるはず。
 - 1 epoch \approx 1 日 \rightarrow それなりに重い。
- 次は強化学習へ。

物理制約を課す強化学習



- まずは「物理を教える」段階：モデルの親子出力を評価。
 - 親子の PDG の組み合わせが正しい崩壊なら報酬。
 - 親子で運動量保存を満たせば報酬。
 - truth と reco の運動量を比較し、近さに応じて報酬。
 - hard (厳密) と soft (部分点) 両方を使い、soft で hard を導く。

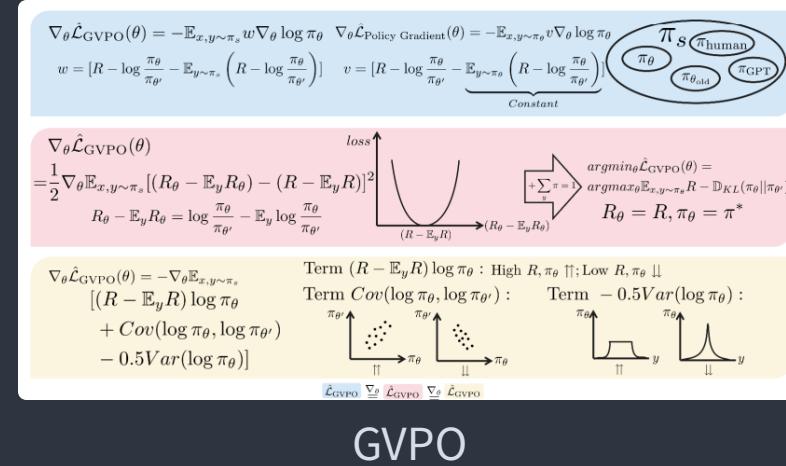
• Grouped Variance Policy Optimization (GVPO):

- 各プロンプトで複数出力をサンプリングし、報酬を計算してグループ内で順位付け。
- 価値ネットワーク不要で、LM と相性が良い。

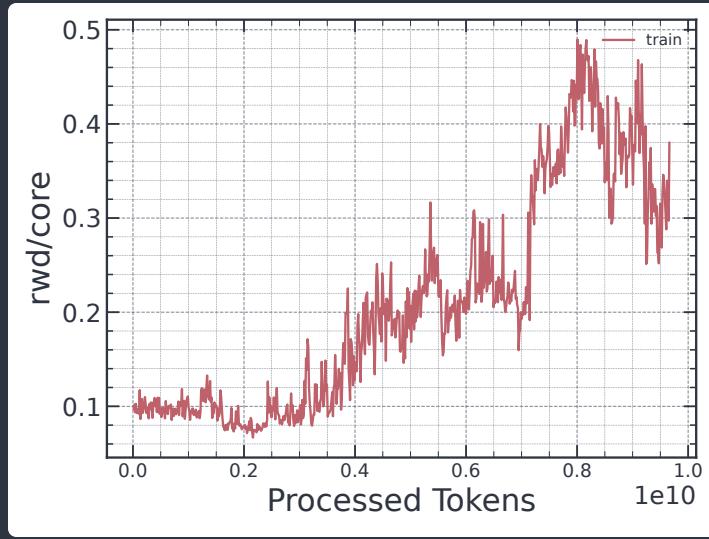
$$\max_{\pi_\theta} \mathbb{E}_{a \sim \pi_\theta}[r(a)] \quad \text{s.t. } \text{KL}(\pi_\theta || \pi_{\text{ref}}) \leq \epsilon$$

$$\pi^*(a|s) \propto \pi_{\text{ref}}(a|s) \exp(\beta r(a))$$

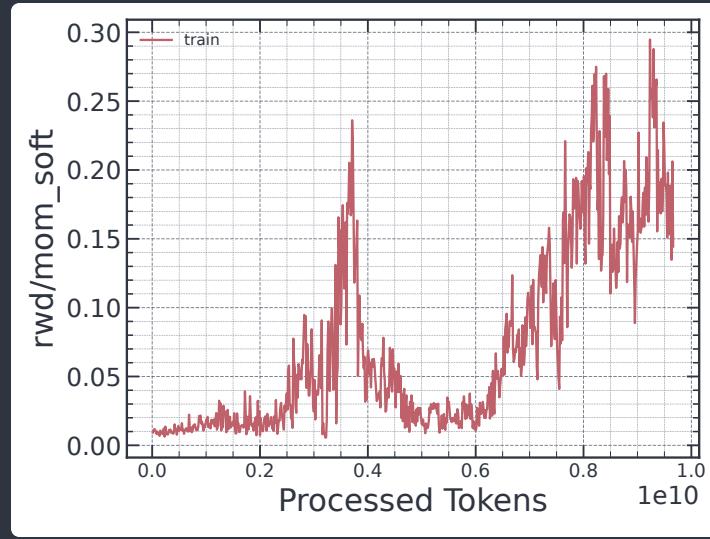
$$\mathcal{L} = -\langle \min(rA, \text{clip}(r, 1-\epsilon, 1+\epsilon), A) \rangle + \beta, \text{KL}$$



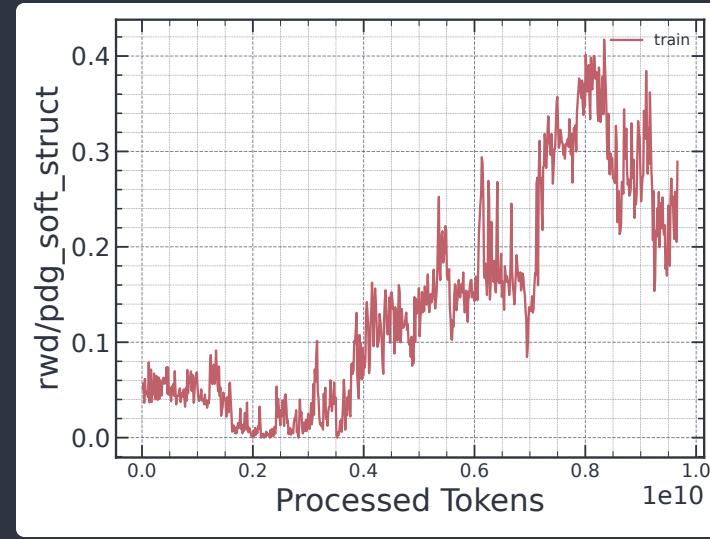
RL 結果



コア報酬



運動量項



崩壊項

- 報酬は親子ペアごとに「崩壊の妥当性」と「運動量保存」を評価。
- 各報酬は崩壊項と運動量項で構成。
 - Decay:** モデル出力が PDG 上正しい崩壊なら加点。
 - Momentum:** モデル出力が運動量保存を満たせば加点。
- 全ペアの報酬を統合。
- この手法は機能しており、近日 arXiv 2601.XXXXXX に投稿予定。

Summary

Summary

- 高エネルギー実験(LHC)での機械学習を紹介
- さまざまなPoCが存在 → しかし実際に実用されるものは精度と拡張性などが重要
- おもに, ジェット関連やシミュレーションの置き換えがあつい
 - ジェット: よくわからない箇所の推測や分類はNNが強い
 - シミュレーション: 精度と生成速度の両立が可能のように思える
- 現状は単純なモデル(分類とかシミュレーションの置換)が多い
- 今後は, 生成モデルや基盤モデルに研究と応用が進む(と思っている)

Backup