Attention, Transformers, and Symmetries for Inverse Problems in High Energy Physics

Alexander Shmakov
August 13, 2024

Problem Overviews

Where do we need attention?

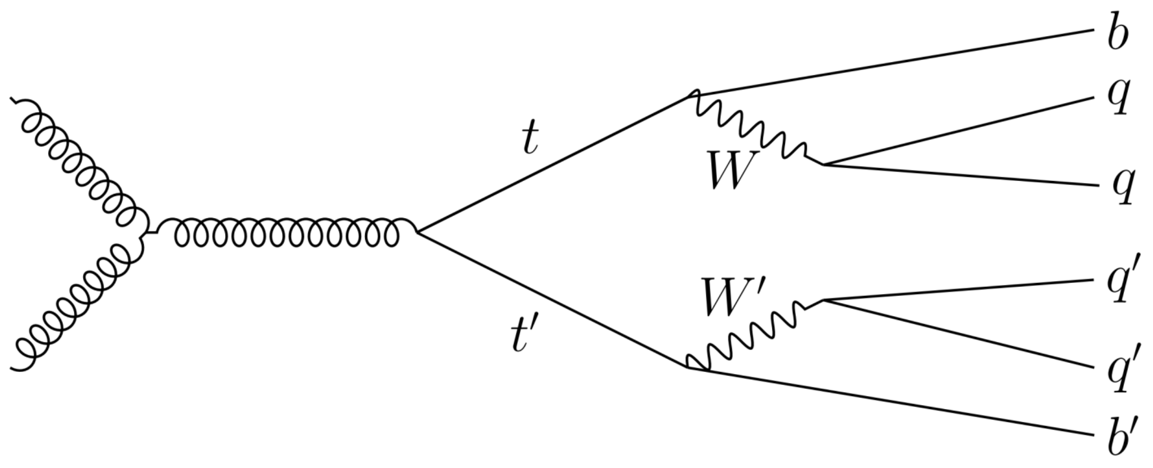
Problem Overview Reconstruction

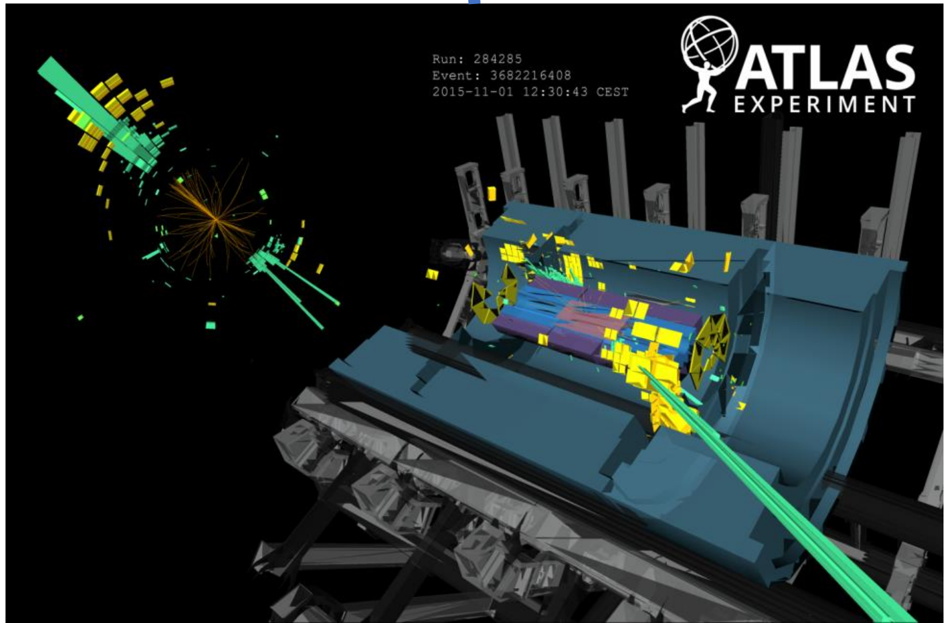
- One of the most common inverse problem: identify produced particles.
- Particles are detected as collimated streams (jets)
- Must be mapped to a more fundamental, theoretical identity.

Reconstruction

Theory
Partons

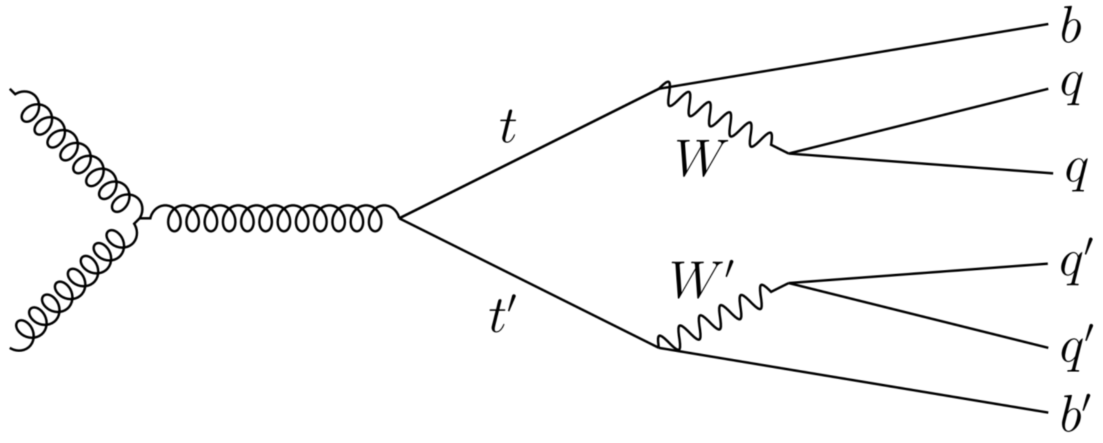
Detection
Jets





Problem Overview Jet-Parton Matching

- Many simultaneous decays in each event.
- Many more jets than partons.
- Initial cuts and requirements eliminate most of the garbage jets
    - Most events have at least 2 extra jets which must be removed.
- **Complications**
    - Inputs are unordered collections (Sets) of jets.
    - Outputs are not unique!



Set of Jets

$$\{j_1, j_2, j_3, j_4, j_5, j_6, j_7, j_8\}$$

$$\Big\downarrow$$

Match Jets to Parton Labels

$$\{b, q', \emptyset, q', b', \emptyset, q, q\}$$

Targets

Garbage Jets

Jet-Parton Matching Set Assignment

This modeling task reduces to a unique set assignment problem.

Input is a set of size N

$$\{j_1, j_2, \dots, j_N\}$$

Possible Targets are a set of size $C \leq N$ and a special null target $\emptyset$

$$\{\emptyset, t_1, t_2, \dots, t_C\}$$

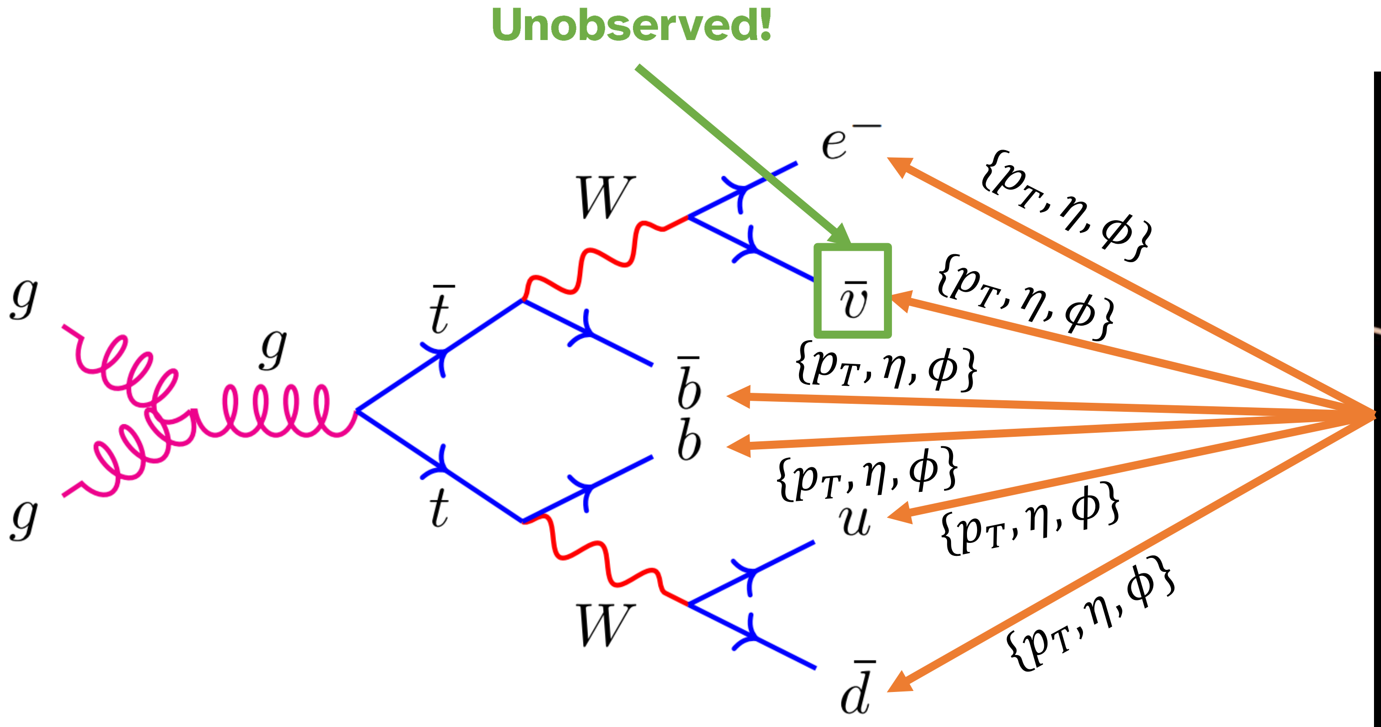
Output is another set of size N

with each $\boldsymbol{p} \in \{\emptyset, \boldsymbol{t}_1, \boldsymbol{t}_2, \dots, \boldsymbol{j}_C\}$ $\boldsymbol{s.t.}$ $\boldsymbol{p}_i \neq \boldsymbol{p}_j$ $\boldsymbol{or}$ $\boldsymbol{p}_i = \emptyset$

$$\{p_1, p_2, \dots, p_N\}$$

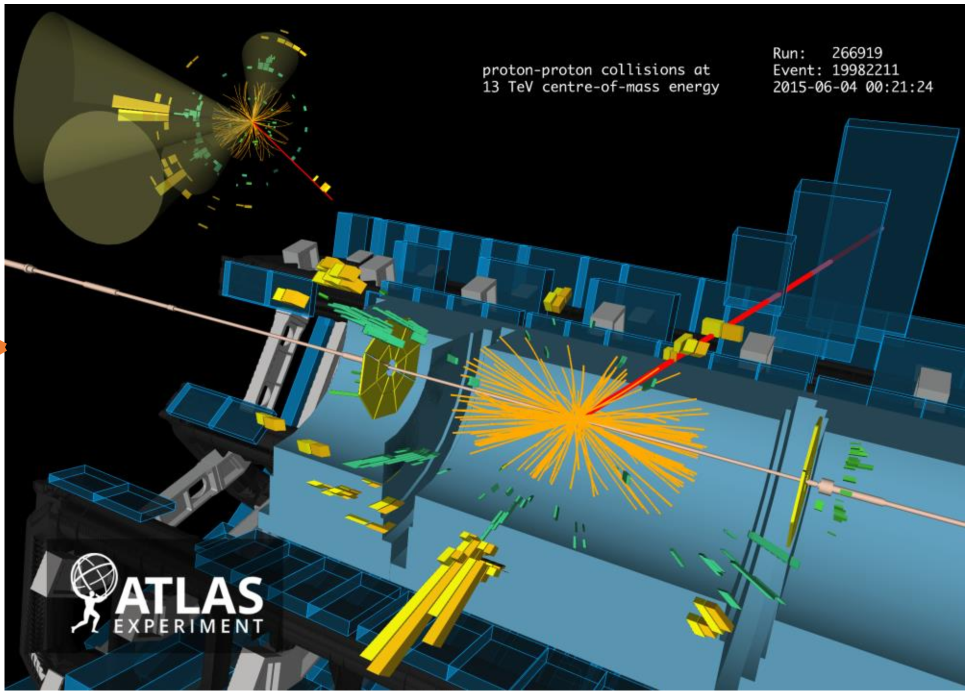
Problem Overview Parton Unfolding

- **A larger reconstruction problem:** Recover kinematic description of more fundamental particles.
- **Challenge**: Map set of jets to set of Parton kinematics.
- Need top effective **Summarize** a set of jets into a completed Parton description.



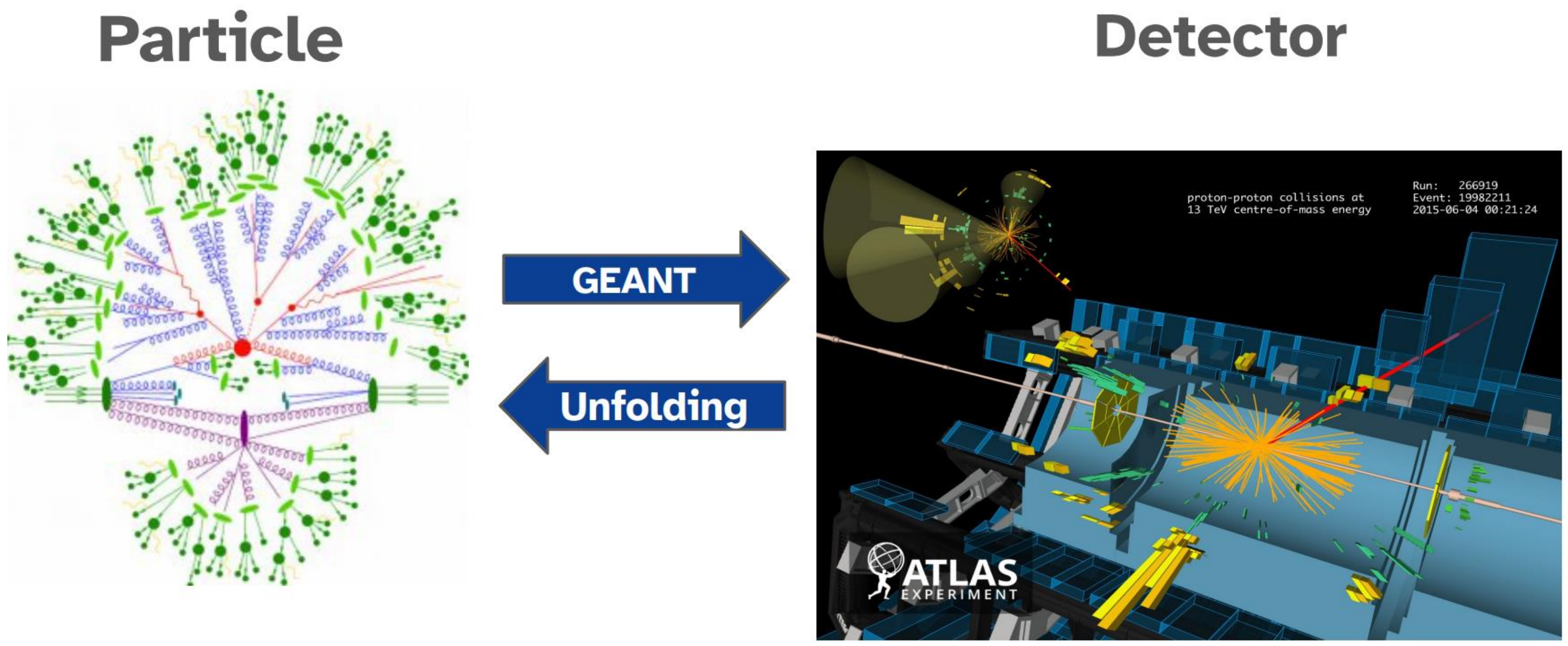
Set of Jets

$$\{j_1, j_2, j_3, j_4, j_5, j_6, j_7, j_8\}$$



Problem Overview Particle-level Unfolding

- **Invert just the detector response**
- Map a set of jets to a different set of particles
- Set cardinality may not be equal!



Symmetry Input Permutation Equivariance

- Input has not inherent order, just a collection of observations.
- Outputs match the order of input.
- Any approach must work for **any** initial ordering inputs.

$$\{j_1, j_2, j_3, j_4, j_5, j_6, j_7, j_8\} \cong \{j_3, j_7, j_1, j_2, j_8, j_4, j_6, j_5\}$$
$$\{b, q', \emptyset, q', b', \emptyset, q, q\} \cong \{\emptyset, q, b, q', q, q', \emptyset, b'\}$$

- *Enforce* an arbitrary consistent ordering? **Hard to justify.**
- Process every possible permutation? **Very expensive**.
- Use a permutation equivariant architecture from the start!

Attention

Overview

Need a method to process sets, maintaining permutation symmetry.

Several different problems:

1. Assign labels to each element of a set (Jet-Parton Assignment)
2. Summarize a set into a fixed-length description (Parton Unfolding)
3. *Generate an entirely new set conditioned on a set (Particle Unfolding)*

Attention

Overview and interpretations.

Attention Overview

Continuous, differentiable
key-value database

Vectors

$$\begin{aligned} Q &= \{q_1, q_2, \ldots, q_m\} \quad \textsc{Queries} \\ K &= \{k_1, k_2, \ldots, k_n\} \quad \textsc{Keys} \\ V &= \{v_1, v_2, \ldots, v_n\} \quad \textsc{Values} \end{aligned}$$

Pick a **SIMILARITY** function. Compute and normalize similarity between query-key pairs.

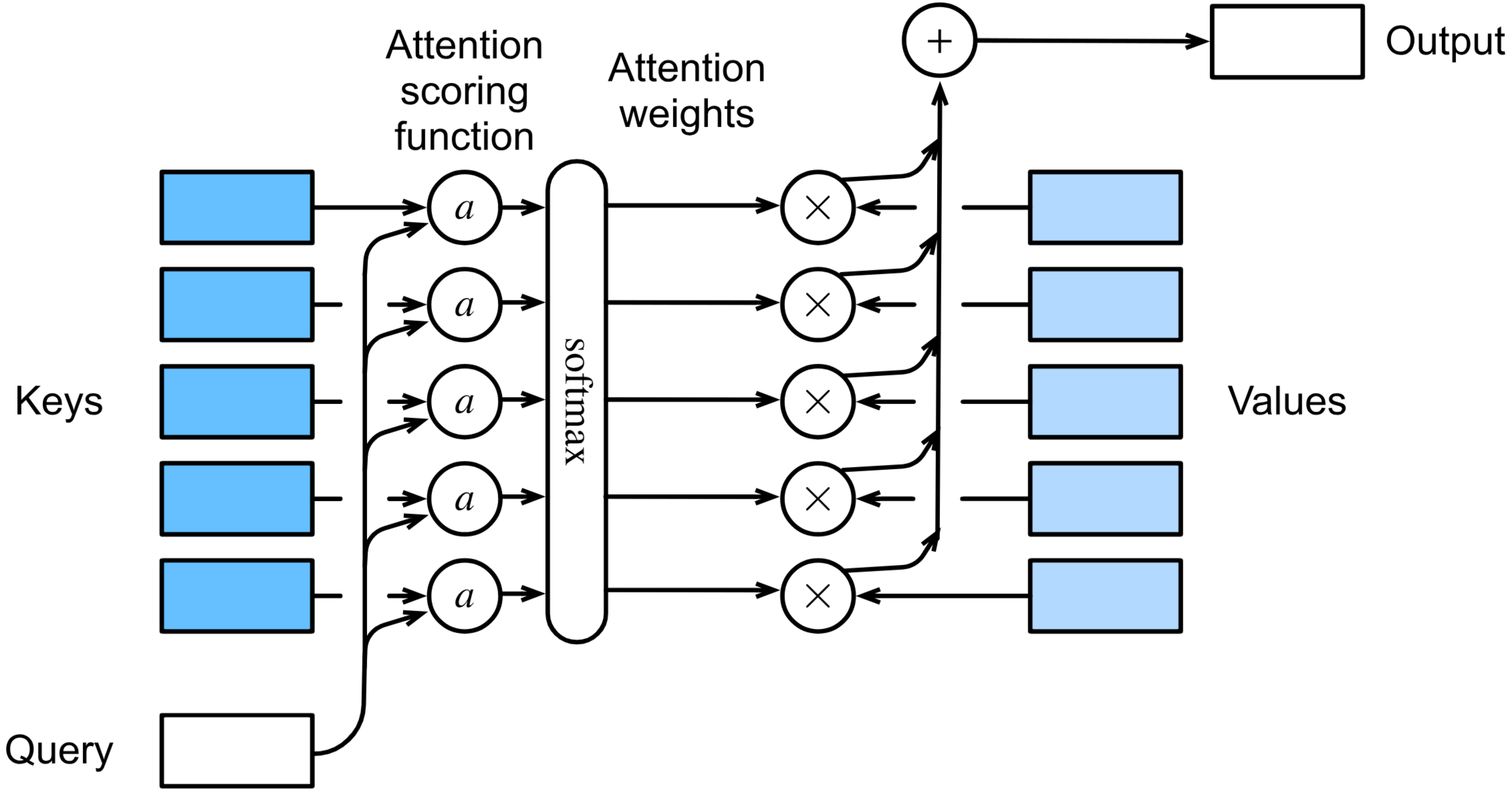
$$S_{ij} = \textsc{Similarity}(q_i, k_j)$$

$$A_{ij} = \textsc{Normalize}(S_{ij}) = \frac{e^{S_{ij}}}{\sum_{l=1}^{n} e^{S_{il}}}$$

Output a weighted average of all values based on similarity.

$$O_i = A_{ij} V^j$$

Attention Overview



Attention Self-Attention

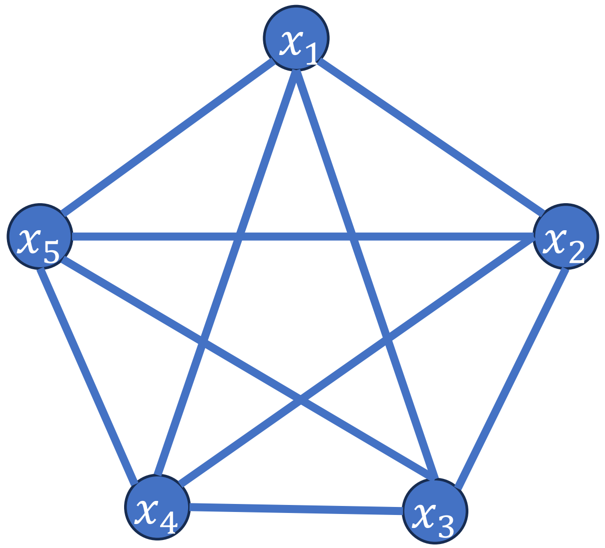
Special Case of attention where we use the same input stream, X, as the queries, keys, and values.

Used to add **context** to collections of objects. Make every element aware of the other elements and learn the relationship between them.

$$Q = f_\theta^Q(X)$$
$$K = f_\theta^K(X)$$
$$V = f_\theta^V(X)$$

Attention Self-Attention

Another Interpretation: A fully connected graph neural network.
Every node sends a message to every other node.
Update equation with a dot-product similarity:



$$x_k \leftarrow \frac{\sum_{i\leftarrow 1}^{N} \left[f_\theta^Q(x_k)^T f_\theta^K(x_i) \right] f_\theta^V(x_i)}{\sum_{i\leftarrow 1}^{N} \left[f_\theta^Q(x_k)^T f_\theta^K(x_i) \right]}$$

Attention Self-Attention

Yet Another Interpretation: A dynamic fully-connect layer.

- Treat all vectors as a single stacked vector of values.
- Construct a dynamics weights matrix from the inputs.

$$o = Av$$

$$v = \left[f_\theta^V(x_1), f_\theta^V(x_2), \ldots, f_\theta^V(x_N)\right]$$

$$A = \text{Attention Matrix from before}$$

Attention Multi-Head Attention

A simple extension: Learn multiple parallel, but smaller, attention layers.

Learn H mappings into a smaller space $\frac{D}{H}$ and recombine (simple concatentate)

$$f_{\theta_i}^{Q,K,V} : \mathbb{R}^D - > \mathbb{R}^{\frac{D}{H}}$$

$$Q_i = f_{\theta_i}^{Q}(X)$$

$$K_i = f_{\theta_i}^{K}(X)$$

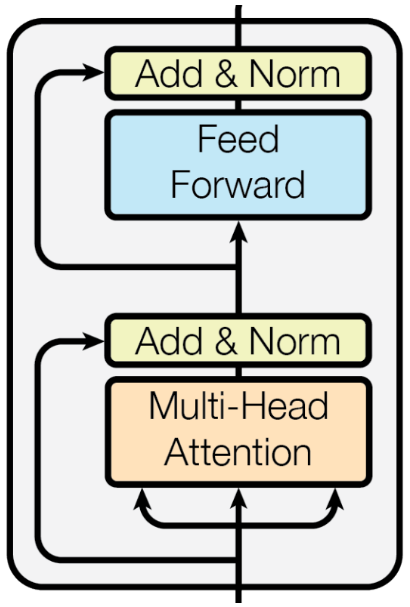
$$V_i = f_{\theta_i}^{V}(X)$$

$$O = [A_1V_1, A_2V_2, \ldots, A_HV_H]$$

Attention Transformers

$$\textsc{Similarity}(q_i, k_j) = \frac{q_i \cdot k_j}{\sqrt{D}}$$

Self-attention[1] with ***scaled dot-product*** as the similarity measure.

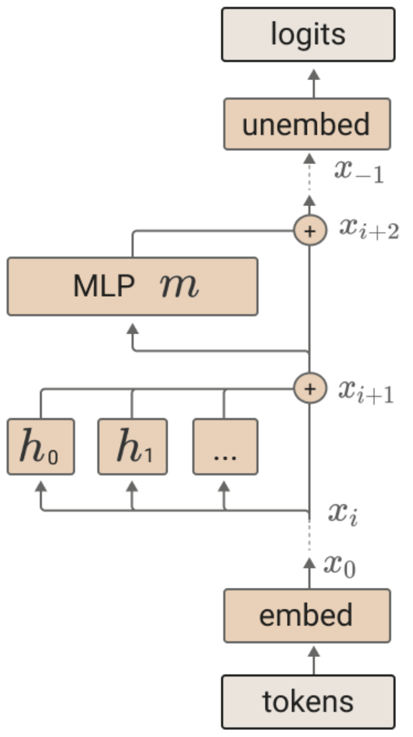


The **transformer encoder** combines

- Scaled dot-product attention
- Skip-connections
- Layer Normalization
- Position-independent feed-forward layers.

1.Vaswani, Ashish, et al. "Attention Is All You Need." Dec. 2017.

Attention Transformers Interpretation



The final logits are produced by applying the unembedding.

$$T(t) = W_U x_{-1}$$

An MLP layer, m, is run and added to the residual stream.

$$x_{i+2} = x_{i+1} + m(x_{i+1})$$

Each attention head, h, is run and added to the residual stream.

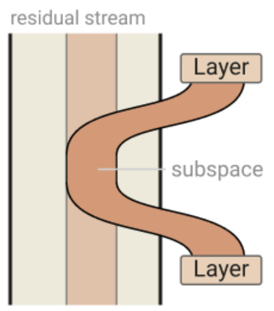
$$x_{i+1} = x_i + \sum_{h \in H_i} h(x_i)$$

One residual block

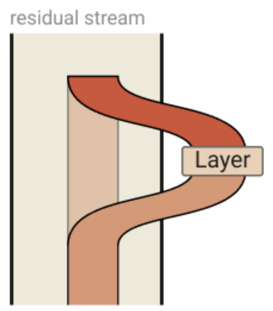
Token embedding.

$$x_0 = W_E t$$

The residual stream is high dimensional, and can be divided into different subspaces.



Layers can interact by writing to and reading from the same or overlapping subspaces. If they write to and read from disjoint subspaces, they won't interact. Typically the spaces only partially overlap.



Layers can delete information from the residual stream by reading in a subspace and then writing the negative verison.

Elhage, et al., "A Mathematical Framework for Transformer Circuits", Transformer Circuits Thread, 2021

Attention Permutation Equivariance

Scaled dot-product is permutation equivariant!

$$P_\pi \in \mathbb{P}^{N \times N}$$

$$\begin{aligned} A_\pi &= \textsc{Normalize}\left(\frac{(P_\pi Q)(P_\pi K)^T}{\sqrt{D}}\right) \\ &= \textsc{Normalize}\left(\frac{P_\pi (QK^T) P_\pi^T}{\sqrt{D}}\right) \\ &= P_\pi \textsc{Normalize}\left(\frac{(QK^T)}{\sqrt{D}}\right) P_\pi^T \\ &= P_\pi A P_\pi^T \end{aligned}$$

$$O_\pi = A_\pi (P_\pi V) = P_\pi A P_\pi^T P_\pi V = P_\pi O$$

Symmetric Attention

An extension of attention for jet-parton matching.

Shmakov A., Fenton M.J., Ho T., et al. *SPANet: Generalized Permutationless Set Assignment for Particle Physics using Symmetry Preserving Attention*. SciPost Phys. 12, 178 (2022)

https://scipost.org/SciPostPhys.12.5.178

https://github.com/Alexanders101/SPANet

Jet-Parton Matching Set Assignment

This modeling task reduces to a unique set assignment problem.

Input is a set of size N

$$\{j_1, j_2, \dots, j_N\}$$

Possible Targets are a set of size $C \leq N$ and a special null target $\emptyset$

$$\{\emptyset, t_1, t_2, \dots, t_C\}$$

Output is another set of size N

with each $\boldsymbol{p} \in \{\emptyset, \boldsymbol{t}_1, \boldsymbol{t}_2, \dots, \boldsymbol{j}_C\}$ $\boldsymbol{s.t.}$ $\boldsymbol{p}_i \neq \boldsymbol{p}_j$ $\boldsymbol{or}$ $\boldsymbol{p}_i = \emptyset$

$$\{p_1, p_2, \dots, p_N\}$$

Set Assignment Itemized Approach

The simplest approach to unique set classification would be **independent classification.**

Train a jet classifier f_θ which treats each jet as a separate object.

Postprocess your predictions and select the highest probability assignment.

$$\begin{tikzcd}
\{j_1, j_2, \dots, j_N\} \arrow[d, "f_\theta"] \\
\{\vec{d}_1, \vec{d}_2, \dots, \vec{d}_N\} \arrow[d, "max"] \\
\{p_1, p_2, \dots, p_N\}
\end{tikzcd}$$

Big Problems!

- How to prevent two identical targets being predicted? Maybe removing elements?
- How to pick order to go through targets? Different ordering could change the prediction!
- The network has no information about the uniqueness. No context for each input!

Set Assignment Permutation Approach

A more invariant approach to this would be the **permutation score function**.

Generate every C-permutation of your set.

Score each permutation with DNN $f_\theta(S) \in \mathbb{R}$.

Predict the highest scoring permutation.

$$\begin{tikzcd}
\{j_1, j_2, \dots, j_k\} \arrow[d, "\textbf{Permute}"] \\
\{S_1, S_2, \dots, S_{k\mathbf{P}C}\} \arrow[d, "\textbf{Score}"] \\
\{f_\theta(S_1), f_\theta(S_2), \dots, f_\theta(S_{k\mathbf{P}T})\}
\end{tikzcd}$$

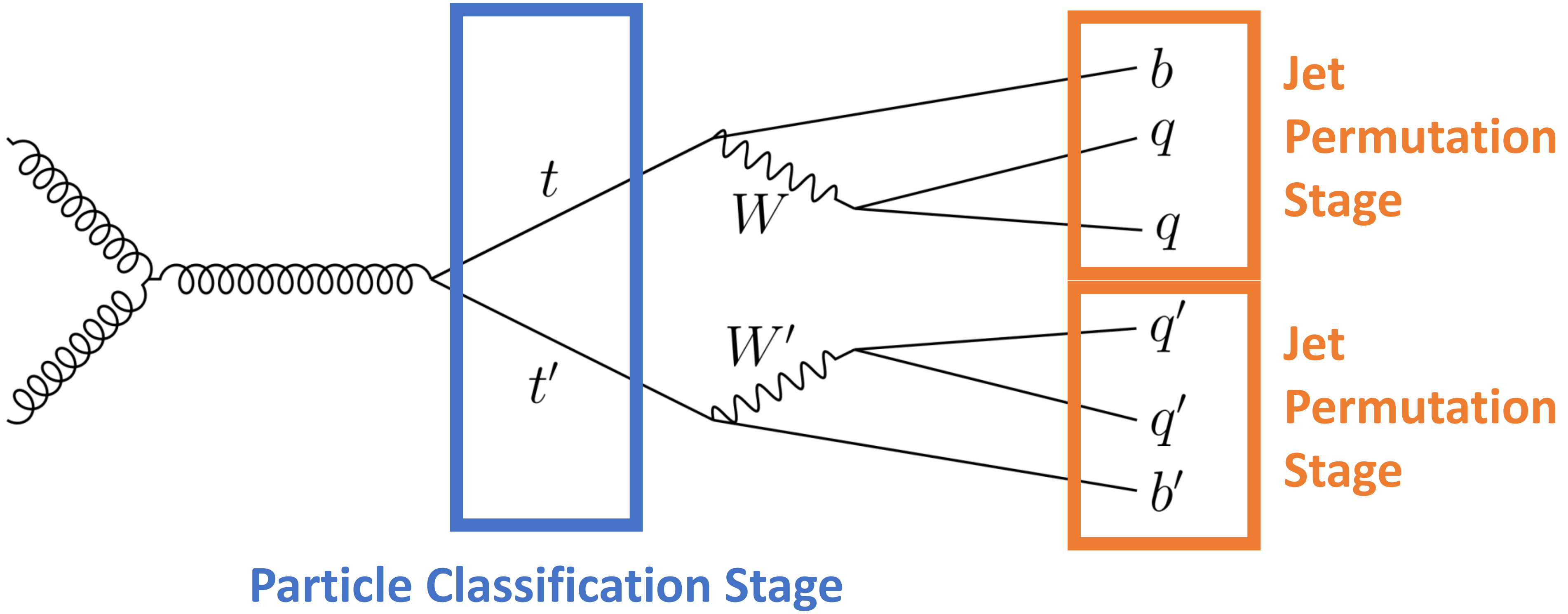
Good approach but terrible run-time

- Currently used in many baseline methods
- Need to generate every permutation! Runtime is $O(N^C)$.

SPANet A Combined Approach

Merge these two approaches to get the best of both **plus symmetry**!

Output **independent sub-permutations scores** for each top-level particle and learn to **differentiate sub-permutations with classification**.



Symmetry Target Symmetries

One very interesting property of Feynman Diagram matching is the presence of symmetries.
The following target sets are equivalent due to charge symmetry.

$$\boldsymbol{q_1 q_2} b q_1' q_2' b' \leftrightarrow \boldsymbol{q_2 q_1} b q_1' q_2' b'$$

$$q_1 q_2 b \boldsymbol{q_1' q_2'} b' \leftrightarrow q_1 q_2 b \boldsymbol{q_2' q_1'} b'$$

We call these **jet symmetries.**
We will handle this with **attention**.

$$\underbrace{q_1 q_2 b}_{}\overset{\mathcal{T}_1}{} \; \overset{\mathcal{T}_2}{\boldsymbol{q_1' q_2' b'}} \leftrightarrow \overset{\mathcal{T}_2}{\boldsymbol{q_1' q_2' b'}} \; \overset{\mathcal{T}_1}{q_1 q_2 b}$$

We call this **particle symmetries.**
We will handle this with a **special loss function**.

Note: this is not the same as allowing duplicate targets because the target groupings **must remain together**.

$$\boldsymbol{q_1 q_2 b q_1'} q_2' b' \neq \boldsymbol{q_1' q_2 b q_1} q_2' b'$$

Tensor Attention Overview

We can also use attention to produce **joint distributions over N dimensions**.
Generalization of dot-product attention: **Tensor Attention**

Suppose $\boldsymbol{X}$ is our list of vectors. This can be viewed as a $(1,1)$-tensor with ranks (N, D).

Suppose $\boldsymbol{\Theta}$ is a $(0, K)$-tensor of **learnable weights** with rank $(D, D, \ldots, D)$.

1. Perform generalized dot-product self-attention on $\boldsymbol{X}$ with the mixing weights $\boldsymbol{\Theta}$.

$$O^{j_1 j_2 \cdots j_N} = X^{j_1}_{n_1} X^{j_2}_{n_2} \ldots X^{j_N}_{n_N} \Theta^{n_1 n_2 \ldots n_N}$$

2. Create a $\boldsymbol{K}$-joint distribution $\boldsymbol{P}$ by normalizing $\boldsymbol{O}$.

$$\mathcal{P}^{j_1 j_2 \cdots j_N} = \frac{\exp O^{j_1 j_2 \cdots j_N}}{\sum \exp O}$$

Tensor Attention Symmetric Attention

Suppose we want our joint distribution to obey **permutation symmetries**.
For example: $p(j_1, j_2, \ldots) = p(j_2, j_1, \ldots)$.

We encode this as a symmetry group on the indices of $\boldsymbol{\Theta}$

Suppose $\boldsymbol{G_P} \subseteq \boldsymbol{S_K}$ is a permutation group acting on the indices $\{j_1, j_2, \ldots, j_K\}$.

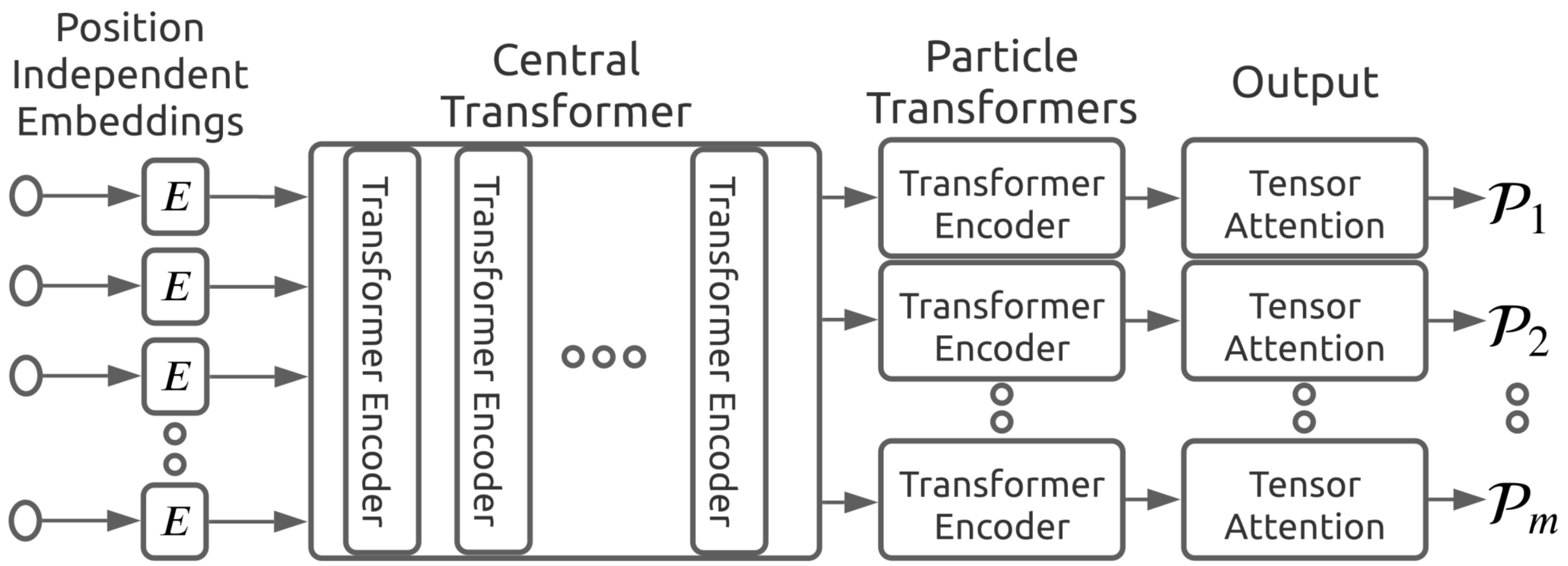
1. Create a symmetric weights tensor $\boldsymbol{S}$ by summing over the symmetric indices of $\boldsymbol{\theta}$ according to $\boldsymbol{G_P}$.
2. Perform generalized dot-product on the list of input vectors $\boldsymbol{X}$ with a **symmetric** weights tensor $\boldsymbol{S}$.
3. Create a **symmetric** joint distribution $\boldsymbol{P}$ by normalizing $\boldsymbol{O}$ just like before.

$$S^{i_1 i_2 \ldots i_K} = \sum_{\sigma \in G_P} \Theta^{i_{\sigma(1)} i_{\sigma(2)} \cdots i_{\sigma(K)}}$$

$$O^{j_1 j_2 \cdots j_K} = X_{i_1}^{j_1} X_{i_2}^{j_2} \ldots X_{i_K}^{j_K} S^{i_1 i_2 \ldots i_K}$$

$$\mathcal{P}^{j_1 j_2 \cdots j_K} = \frac{\exp\left(O^{j_1 j_2 \cdots j_K}\right)}{\sum_{j_1, j_2, \ldots, j_K} \exp\left(O^{j_1 j_2 \cdots j_K}\right)}$$

SPANet Architecture



SPANet Training the Permutation Ranker

- Train symmetric joint-distributions using a simple categorical cross-entropy.
- **One special difference**
  - The target $\mathcal{T}$ is not a delta distribution!
  - $\mathcal{T}$ will be non-zero for every valid symmetric assignment.

$$\mathcal{L}_P(\mathcal{P}, \mathcal{T}) = \sum_{j_1, j_2, \cdots, j_N} -\mathcal{T}^{j_1 j_2 \cdots j_N} \log \mathcal{P}^{j_1 j_2 \cdots j_N}$$

SPANet Symmetric Training

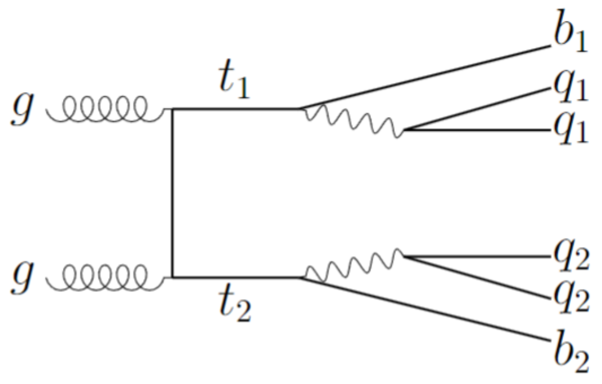
- Handle event symmetries using a symmetric loss function.
- Define an **event-level symmetry group** $G_E \subseteq \boldsymbol{S}_m$ acting on particles $\{P_1, P_2, \dots, P_m\}$
- Loss function simply takes the **minimum** achievable loss over valid permutations.
- Also tried other methods such as *sum* of *soft-min*. Simple *min* works the best.

$$\mathcal{L} = \min_{\sigma \in G_E} \sum_{i=1}^{m} \mathcal{L}_P \left(\mathcal{P}_{\sigma(i)}, \mathcal{T}_{\sigma(i)} \right)$$

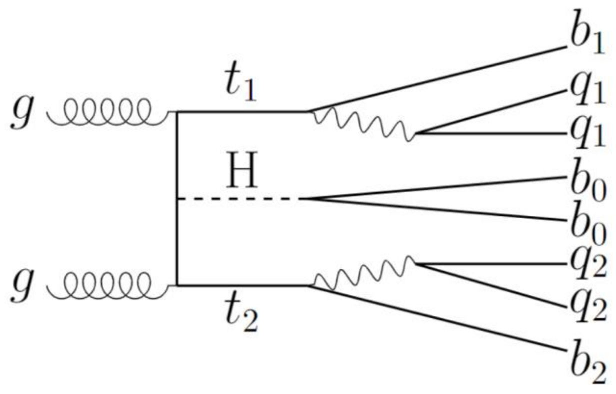
SPANet Results

- Compare to a common baseline used in CERN analyses – A permutation-based χ^2 approach.
- Greatly improve accuracy over baseline methods. On average around ~25% improvement.
- Drastically increase runtime performance. Baseline method cannot tractably evaluate $tttt$!

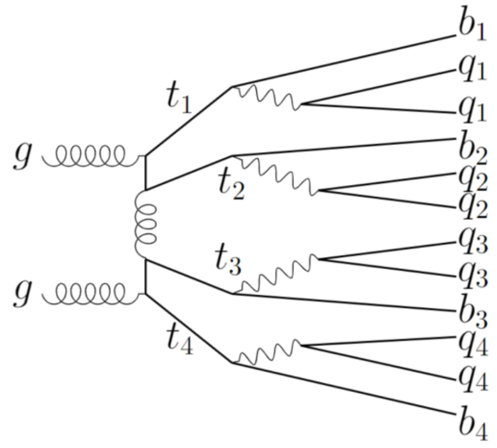
$t\bar{t}$

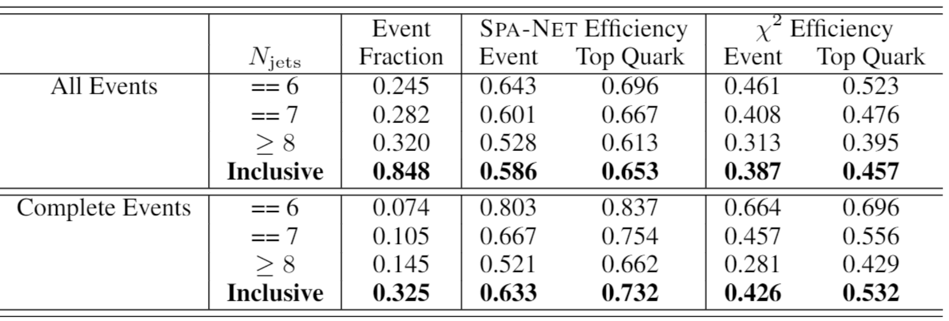


ttH

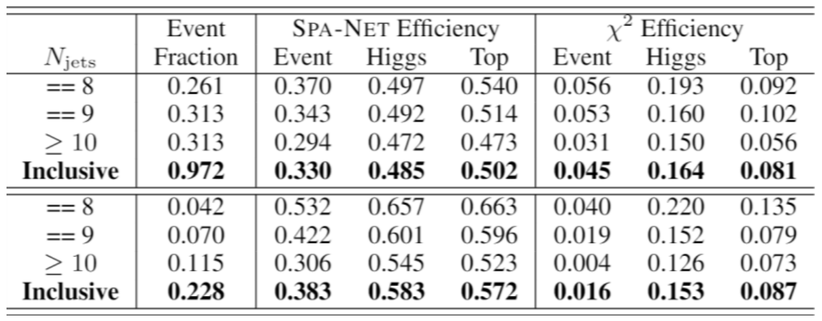


$tttt$

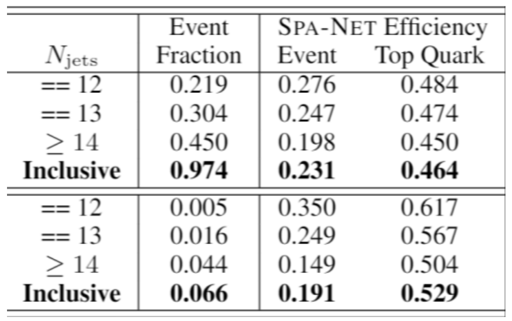




	N_{jets}	Event Fraction	SPA-NET Efficiency Event	SPA-NET Efficiency Top Quark	χ^2 Efficiency Event	χ^2 Efficiency Top Quark
All Events	== 6	0.245	0.643	0.696	0.461	0.523
	== 7	0.282	0.601	0.667	0.408	0.476
	≥ 8	0.320	0.528	0.613	0.313	0.395
	Inclusive	**0.848**	**0.586**	**0.653**	**0.387**	**0.457**
Complete Events	== 6	0.074	0.803	0.837	0.664	0.696
	== 7	0.105	0.667	0.754	0.457	0.556
	≥ 8	0.145	0.521	0.662	0.281	0.429
	Inclusive	**0.325**	**0.633**	**0.732**	**0.426**	**0.532**



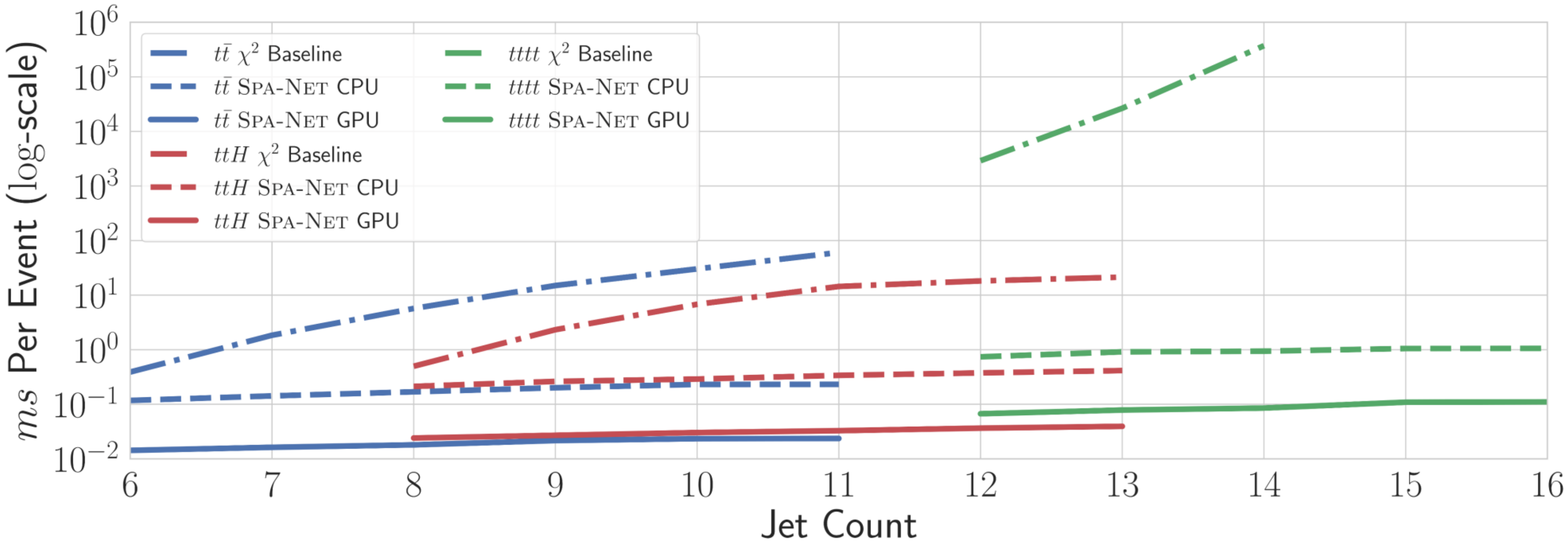
N_{jets}	Event Fraction	SPA-NET Efficiency Event	SPA-NET Efficiency Higgs	SPA-NET Efficiency Top	χ^2 Efficiency Event	χ^2 Efficiency Higgs	χ^2 Efficiency Top
== 8	0.261	0.370	0.497	0.540	0.056	0.193	0.092
== 9	0.313	0.343	0.492	0.514	0.053	0.160	0.102
≥ 10	0.313	0.294	0.472	0.473	0.031	0.150	0.056
Inclusive	**0.972**	**0.330**	**0.485**	**0.502**	**0.045**	**0.164**	**0.081**
== 8	0.042	0.532	0.657	0.663	0.040	0.220	0.135
== 9	0.070	0.422	0.601	0.596	0.019	0.152	0.079
≥ 10	0.115	0.306	0.545	0.523	0.004	0.126	0.073
Inclusive	**0.228**	**0.383**	**0.583**	**0.572**	**0.016**	**0.153**	**0.087**



N_{jets}	Event Fraction	SPA-NET Efficiency Event	SPA-NET Efficiency Top Quark
== 12	0.219	0.276	0.484
== 13	0.304	0.247	0.474
≥ 14	0.450	0.198	0.450
Inclusive	**0.974**	**0.231**	**0.464**
== 12	0.005	0.350	0.617
== 13	0.016	0.249	0.567
≥ 14	0.044	0.149	0.504
Inclusive	**0.066**	**0.191**	**0.529**

SPANet Results

- Compare to a common baseline used in CERN analyses – A permutation-based $\mathcal{X}^2$ approach.
- Greatly improve accuracy over baseline methods. On average around ~25% improvement.
- Drastically increase runtime performance. Baseline method cannot tractably evaluate $tttt$!



Extended SPANet

Further extensions of attention for spatial symmetry and regression.

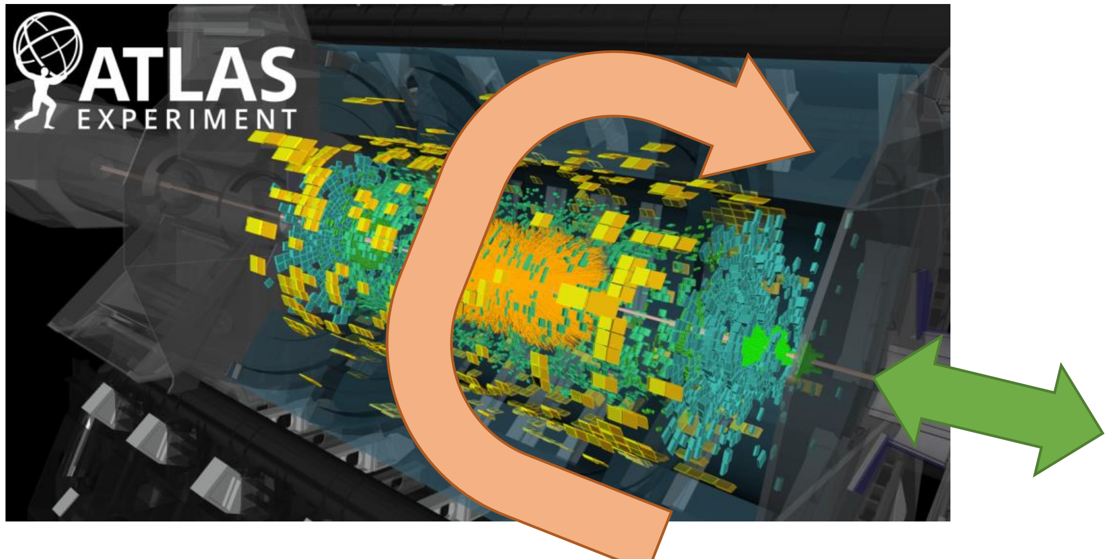
Fenton, M.J., Shmakov, A., Okawa, H. et al. *Reconstruction of unstable heavy particles using deep symmetry-preserving attention networks*. Communications Physics 7, 139 (2024).

https://www.nature.com/articles/s42005-024-01627-4

Extended *SPANet* Continuous Input Symmetries

There are really three types symmetries in reconstruction.

- Mathematical permutation symmetry of sets
  - Handled with transformer attention.
- Discrete Physics symmetries such as CPT
  - Handled charge and parity symmetries with tensor attention and symmetric loss.
- Continuous Physics symmetries – **Lorentz symmetries** of space.
  - We can **rotate** or **flip** the entire detector and get the exact same event.
  - **Still present!**



Extended *SPANet* Continuous Input Symmetries

A **Reference Frame** is a conditional mapping $\phi(u; v)$ over the input vectors. In HEP, our transform is the **Lorentz boost,** which is preserved under a global Lorentz transform.

$$\phi(u; v) = \frac{1}{1 - v \cdot u}\left(\frac{u}{\gamma_v} - v + \frac{\gamma_v}{\gamma_v + 1}(u \cdot v)v\right)$$

Keys and values become **matrix** collections of vectors. Queries remain as lists of vectors.

$$\begin{array}{ll} Q \in \mathbb{R}^{N \times D} & \textsc{Queries} \\ K \in \mathbb{R}^{N \times N \times D} & \textsc{Keys} \\ V \in \mathbb{R}^{N \times N \times D} & \textsc{Values} \end{array}$$

$K_{i,j} = \phi(k_j; k_i)$ represents the $j'th$ key **from i'th reference frame.**

$V_{i,j} = \phi(v_j; v_i)$ represents the $j'th$ value **from i'th reference frame.**

$Q_i = \phi(q_i; q_i)$ represents the $i'th$ query **from i'th reference frame.**

Extended *SPANet* Continuous Input Symmetries

- Modified attention operation is nearly identically to regular attention.
- Just need to add some extra indices
- Output vector, O, is invariant to any global changes w.r.t the perspective operation.

$$S_{i,j} = \frac{Q_i \cdot K_{i,j}}{\sqrt{D}} \quad \in \mathbb{R}$$

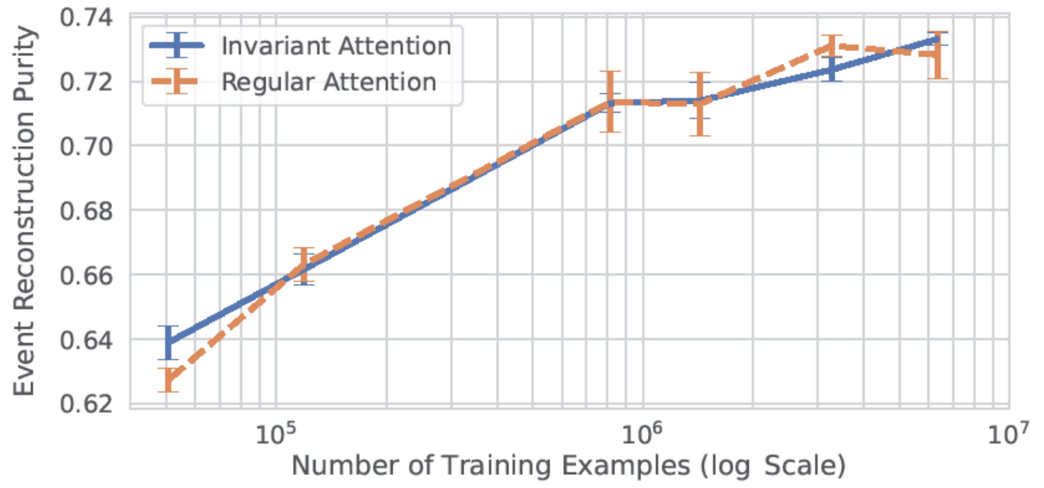
$$A_{i,:} = \textsc{Softmax}(S_{i,:}) \quad \in \mathbb{R}^N$$

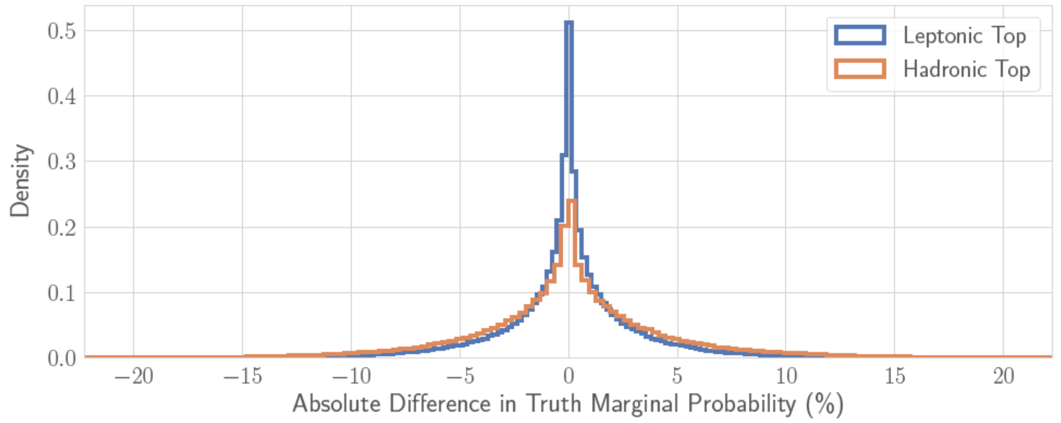
$$O_i = \sum_j (A_{i,j} \odot V_{i,j}) \quad \in \mathbb{R}^D$$

A trivial reference frame function, $\phi(u; v) = u$, reduces this operation regular scaled dot-product attention.

Extended *SPANet* Continuous Input Symmetries

Almost no improvement in performance!





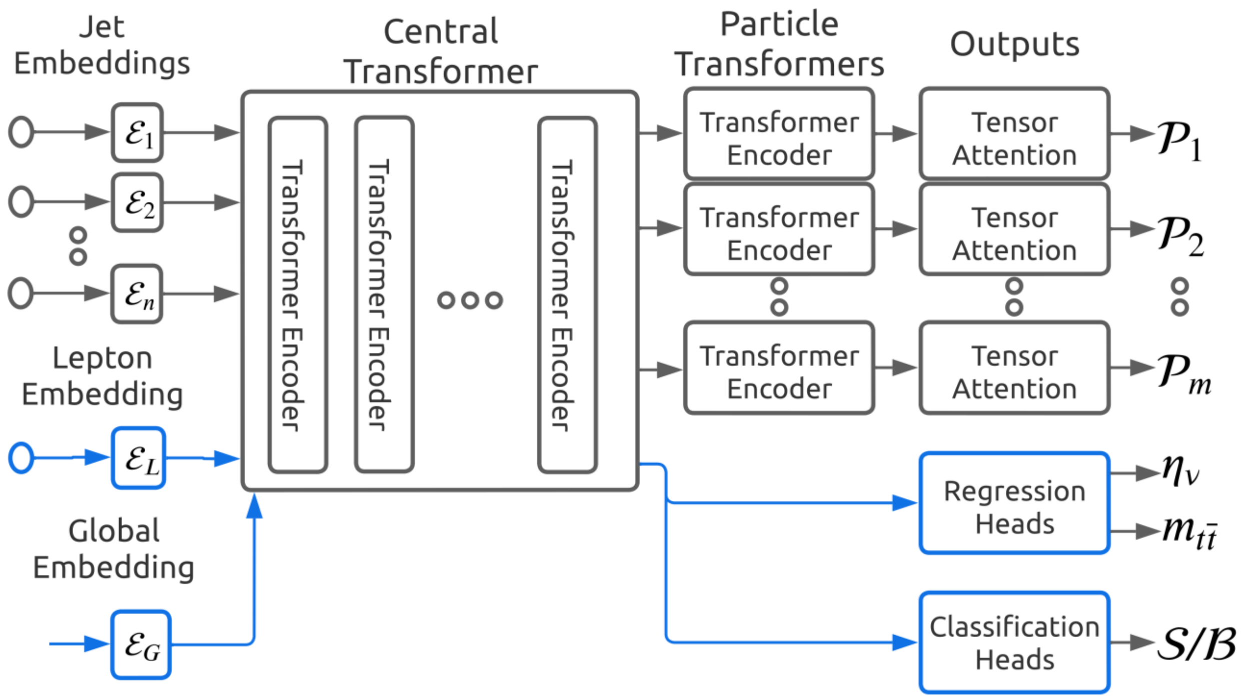
What's going on?

- The transformation function is just a *non-linear conditional function* of the original input.
- This is precisely a transformer!
- If Lorentz invariance is useful, then the network **should just learn it.**

Let's examine how much our assignment probability changes as you feed the **same event** rotated in several ways.

Even without Lorentz invariant attention, SPANet is **approximately invariant**!

Extended *SPANet* Event and Particle Level Outputs



Combined Training
Train SPANet on simulated data with the hypothetical particle present and not present. This will ensure reasonable performance regardless of what the real data holds.

Regress extra neutrino terms

Classify event into signal or background

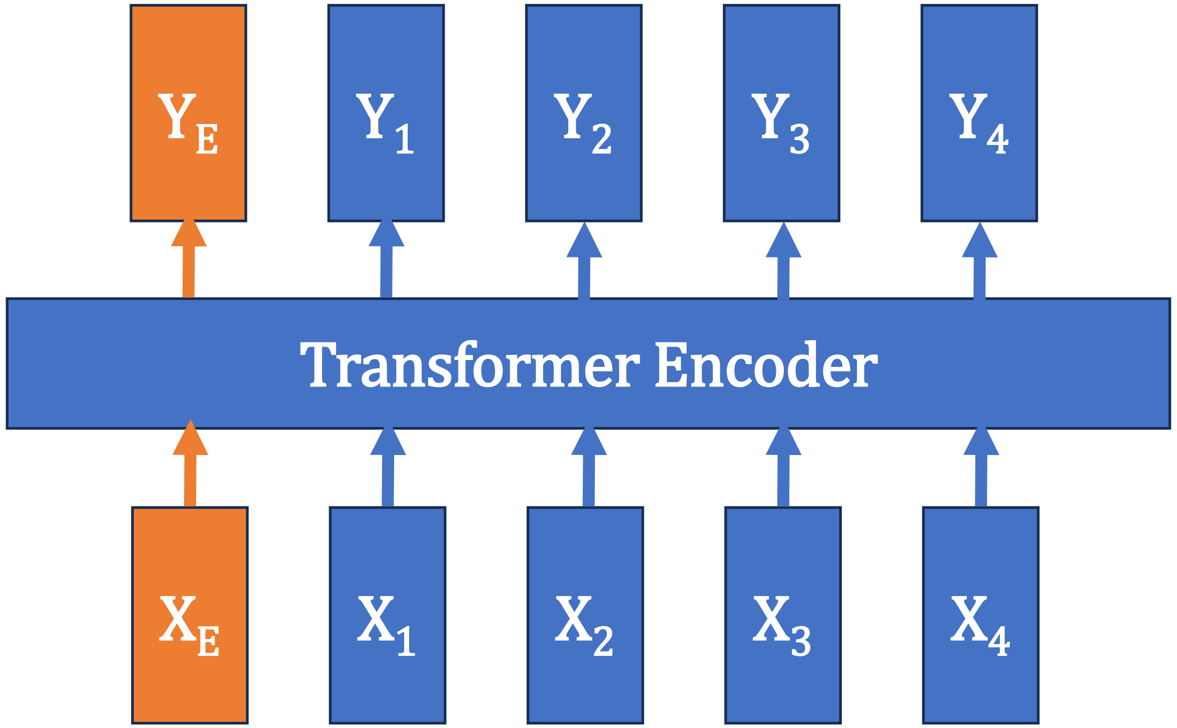
Trainable Global Vector
Will store information about the complete event.

Global Outputs
Use the encoded global vector to predict additional event-level terms.

Extended *SPANet* Event and Particle Level Outputs

We can add a special **Event-Level** output to attention with a neat trick.

Add a special *fixed* vector to our input stream, X_E, which will represent the *event* information. This special vector will have the same value for all events. The transformer's contextual learning will fill Y_E with relevant event information.



Extended *SPANet* Detection Probability

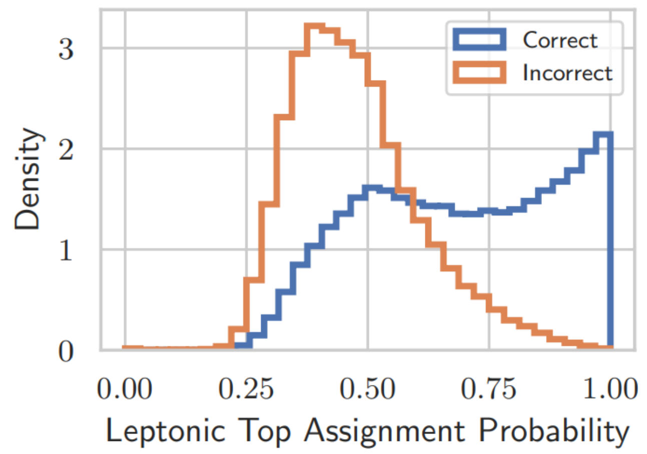
Original SPANet only produced assignment distributions, which we trained only on events where the particle was fully reconstructable. We add the other half of the distribution to detect when the reconstruction is even valid. Detection output is trained using binary cross-entropy on the particle mask.

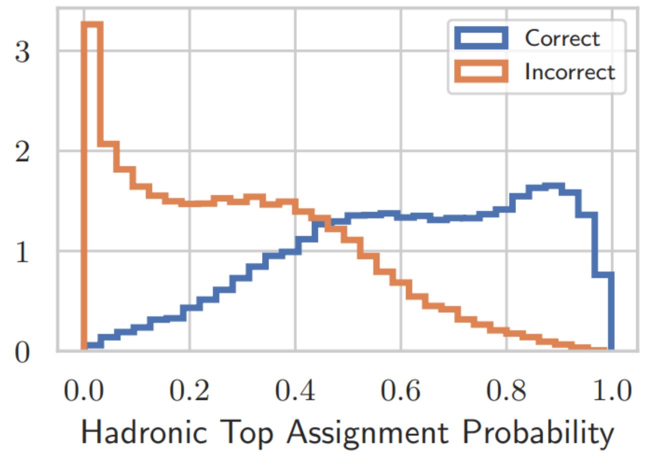
$$\begin{aligned}
\textsc{Assignment Probability:}\quad & P_a(j_1, j_2, \ldots, j_{k_p} \mid p \text{ reconstructable}) \\
\textsc{Detection Probability:}\quad & P_d(p \text{ reconstructable}) \\
\textsc{Marginal Probability:}\quad & P_m(j_1, j_2, \ldots, j_{k_p}) = P_a(j_1, j_2, \ldots, j_{k_p} \mid p) P_d(p)
\end{aligned}$$

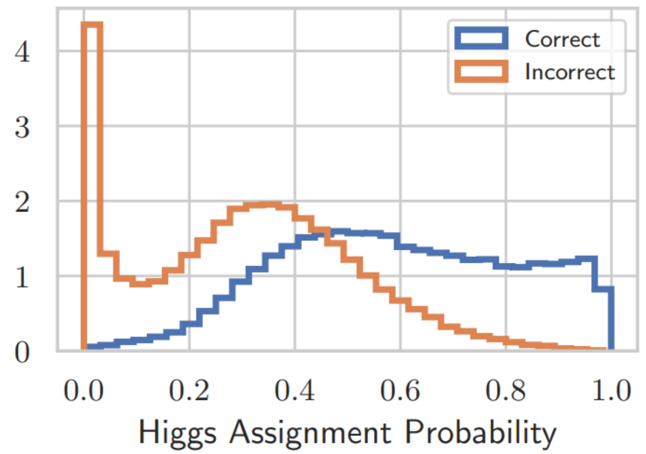
$$\mathcal{L}_{\text{detection}} = \min_{\sigma \in G_E} \left[\mathcal{M}_{\sigma(p)} \log P_d(p) + (1 - \mathcal{M}_{\sigma(p)}) \log (1 - P_d(p)) \right].$$

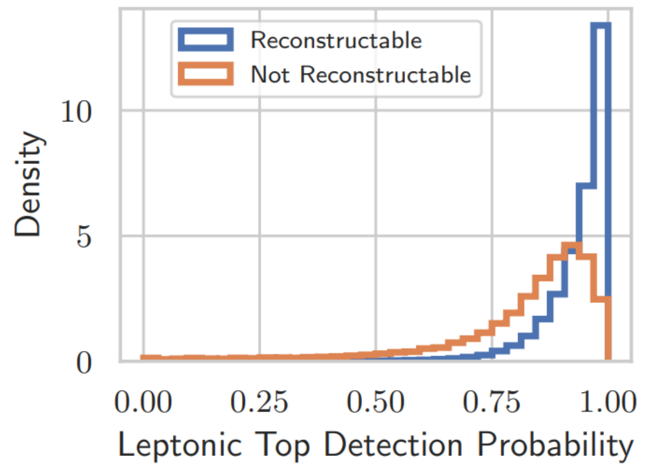
Extended *SPANet* Detection Probability

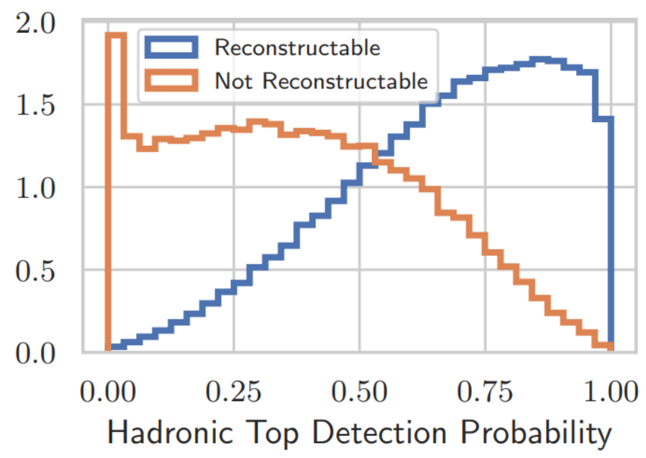
Evaluating this new output head on semi-leptonic ttH

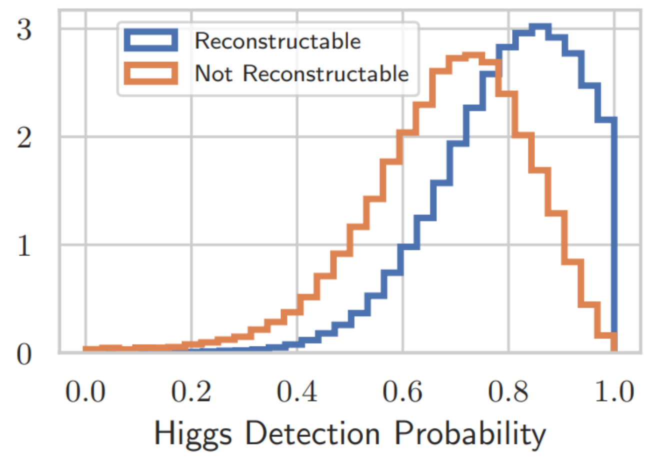




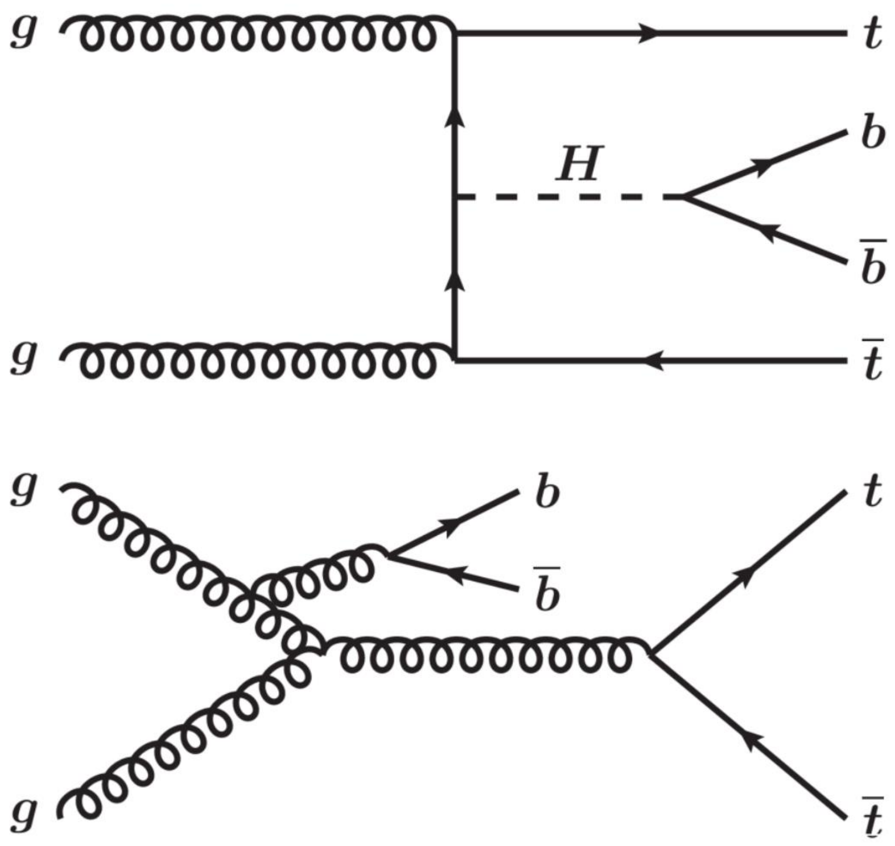








Extended *SPANet* Signal / Background Separation



- $ttbb$ present a significant background to ttH events with very similar kinematics and final states
- **Add auxiliary classification output to SPANet**
  - Feed both ttH and $ttbb$ into SPANet training, labeling events with either 0 or 1.
  - Do not perform reconstruction on $ttbb$ events.
- **Optionally fine-tune auxiliary output after training.**
- Compare to BDTs trained from model scores.
  - Feed assignment, detection, and marginal probabilities into BDT.

Extended *SPANet* Signal / Background Separation

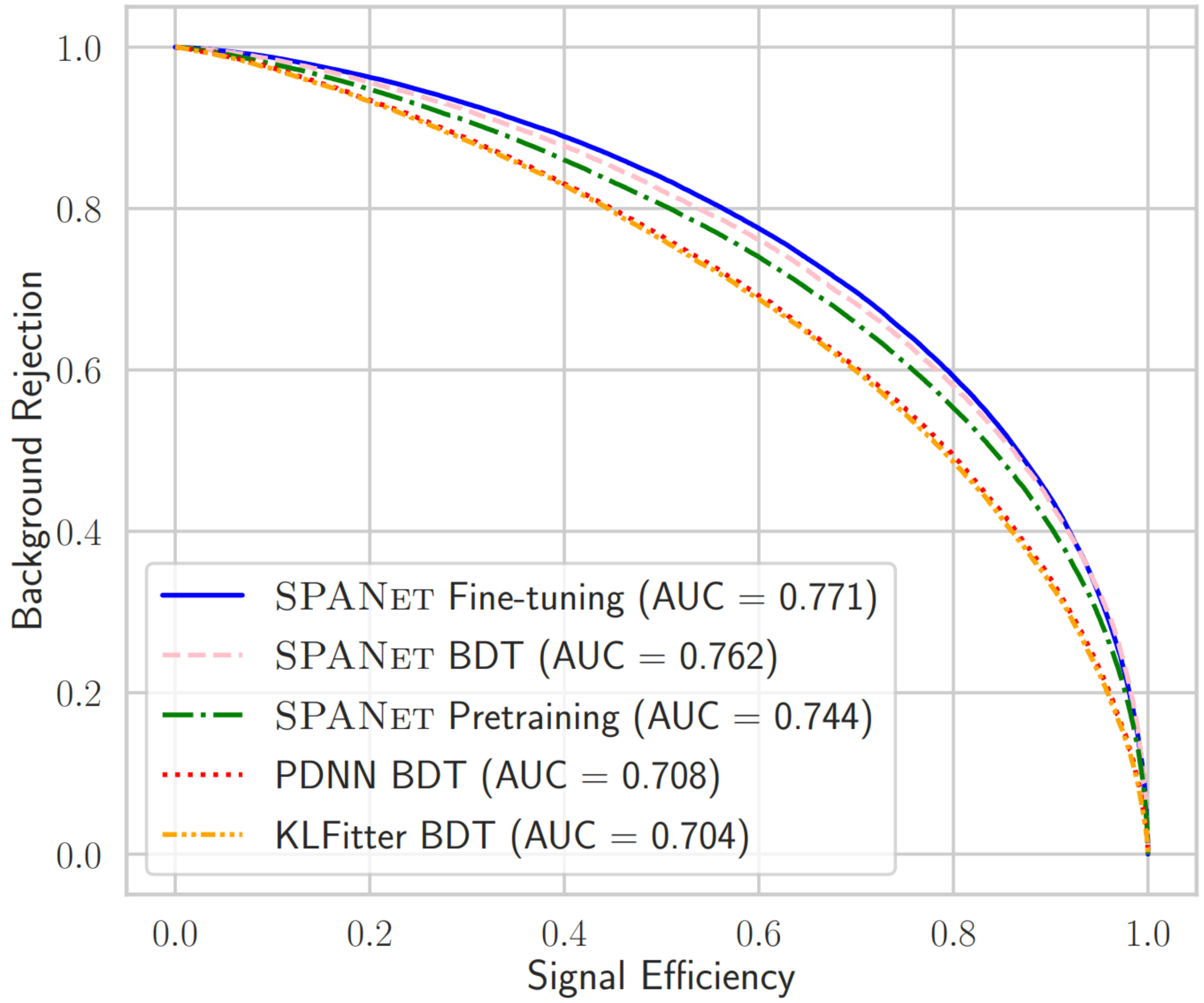


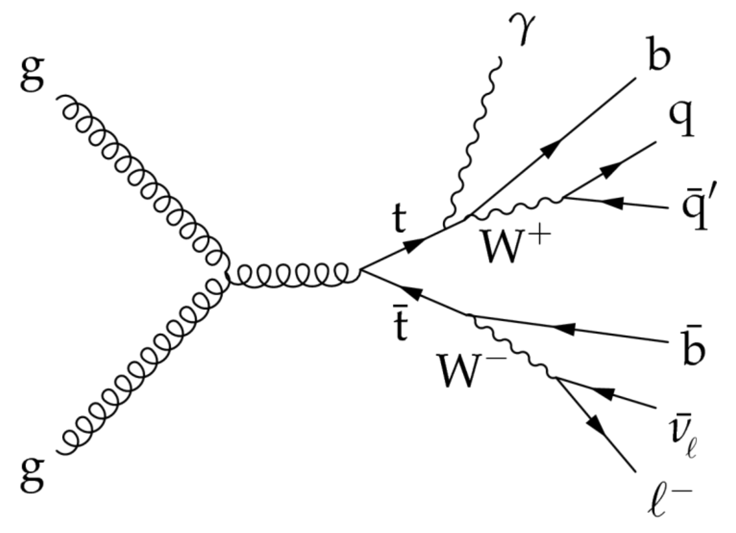
Table 5: Expected LHC Run 2 sensitivity to $t\bar{t}H$ as measured in a parameterized detector model described in the text. Shown is the expected statistical significance of the measurement as well as expected upper limits on cross section and signal strength using the output of classification networks trained on the products of various reconstruction algorithms.

	Signal significance	Upper cross section limit [pb]	Upper signal strength limit
KLFitter BDT	2.38 σ	0.847	0.84
PDNN BDT	2.40 σ	0.843	0.831
SPANET BDT	3.00 σ	0.773	0.671
SPANET pre-training	2.74 σ	0.799	0.732
SPANET fine-tuning	3.08 σ	0.765	0.655

Table 6: Expected LHC Run 3 sensitivity to $t\bar{t}H$ as measured in a parameterized detector model described in the text. Shown is the expected statistical significance of the measurement as well as expected upper limits on cross section and signal strength using the output of classification networks trained on the products of various reconstruction algorithms.

	Signal significance	Upper cross section limit [pb]	Upper signal strength limit
KLFitter BDT	4.11 σ	0.705	0.489
PDNN BDT	4.14 σ	0.703	0.486
SPANET BDT	5.21 σ	0.662	0.387
SPANET pre-training	4.76 σ	0.677	0.423
SPANET fine-tuning	5.72 σ	0.647	0.353

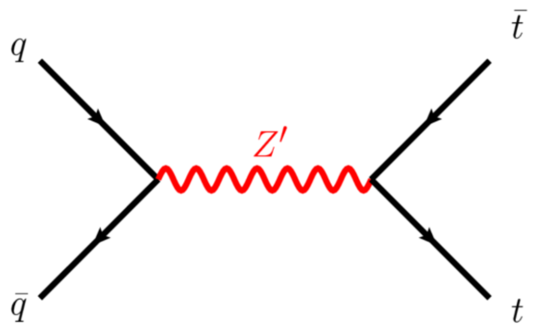
Extended *SPANet* Search for Z'



Background
Standard model semi-leptonic $t\bar{t}$ event.

Events also have an additional *neutrino* term which needs to be reconstructed separately.

Need to separate events into a *signal* and a *background* class depending on if the hypothetical particle was present in the event.



Signal
Hypothetical modified event with a non-standard-model Z' Boson before top decay.

Extended *SPANet* Search for Z’

- Use regression outputs to predict the missing neutrino kinematics to assists in search.
- Also employ signal-background classification to assist in cuts.

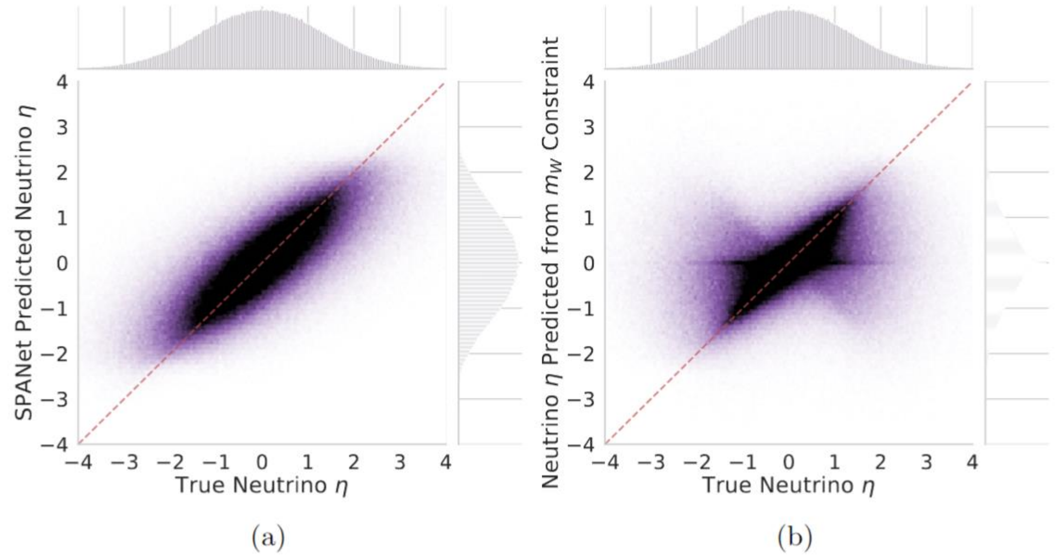


Fig. 2: True neutrino pseudo-rapidity (η) versus predicted values from (a) SPA-NET, and (b) the W-mass constraint.

Table 4: Expected global significance for a Z' signal with an integrated luminosity of 140 (300) fb^{-1}, for several choices of Z' mass and reconstruction algorithms.

	KLFitter	PDNN	SPA-NET	SPA-NET w/ η^ν
$m_{Z'} = 500$ GeV	1.2σ (2.5σ)	1.8σ (3.5σ)	2.8σ (5.5σ)	2.7σ (5.4σ)
$m_{Z'} = 700$ GeV	1.6σ (3.3σ)	2.5σ (4.9σ)	3.1σ (6.1σ)	2.9σ (5.7σ)
$m_{Z'} = 900$ GeV	1.9σ (3.9σ)	2.8σ (5.5σ)	4.3σ (8.5σ)	4.1σ (8.2σ)

Unfolding

Attention for learning kinematic distributions

Shmakov A, Greif K, Fenton M, Ghosh A, Baldi P, Whiteson D. End-To-End Latent Variational Diffusion Models for Inverse Problems in High Energy Physics. Neural Information Processing Systems. 2023

Huetsch et. al. The Landscape of Unfolding with Machine Learning. Preprint. 2024
https://arxiv.org/abs/2404.18807

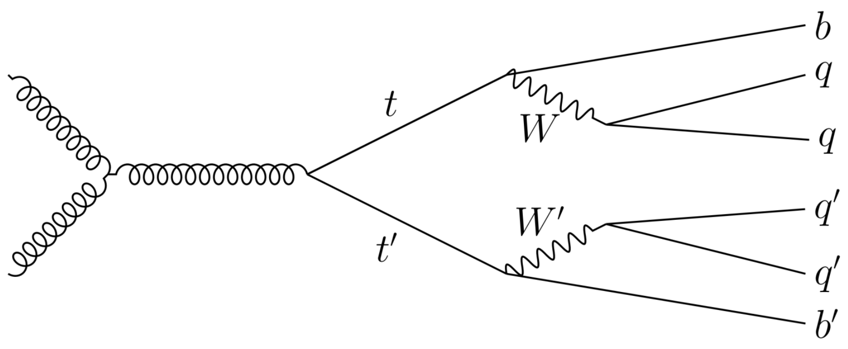
Unfolding Parton Unfolding

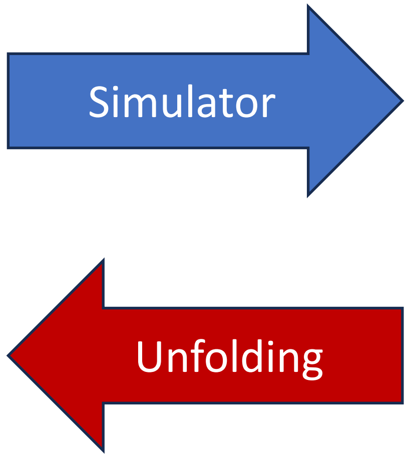
So far, we have been **assigning** jets to source partons. What if we could do more?

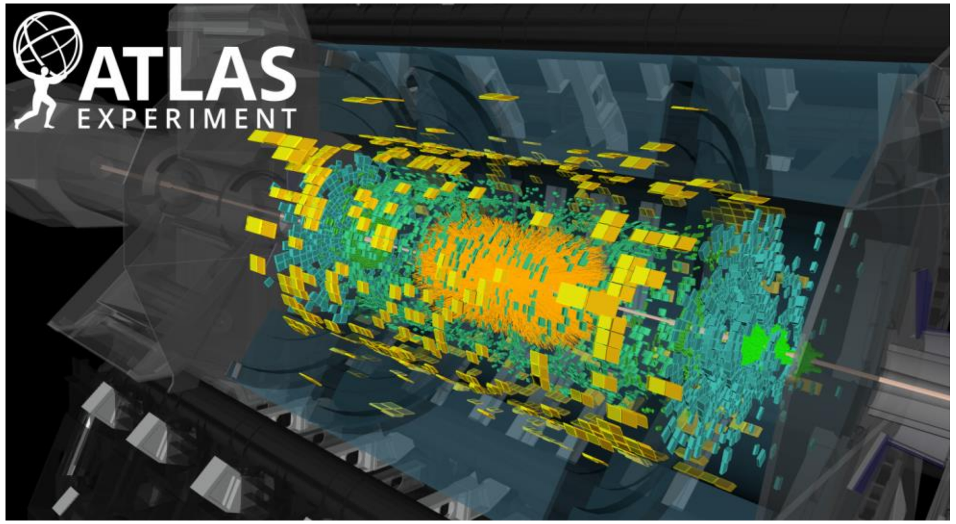
Unfolding Fully recovering the momenta of the source partons from observations.

Simulators (Madgraph, Delphes, etc) define the forward problem:
Parton -> Detector

We want to solve the inverse problem:
Detector -> Parton



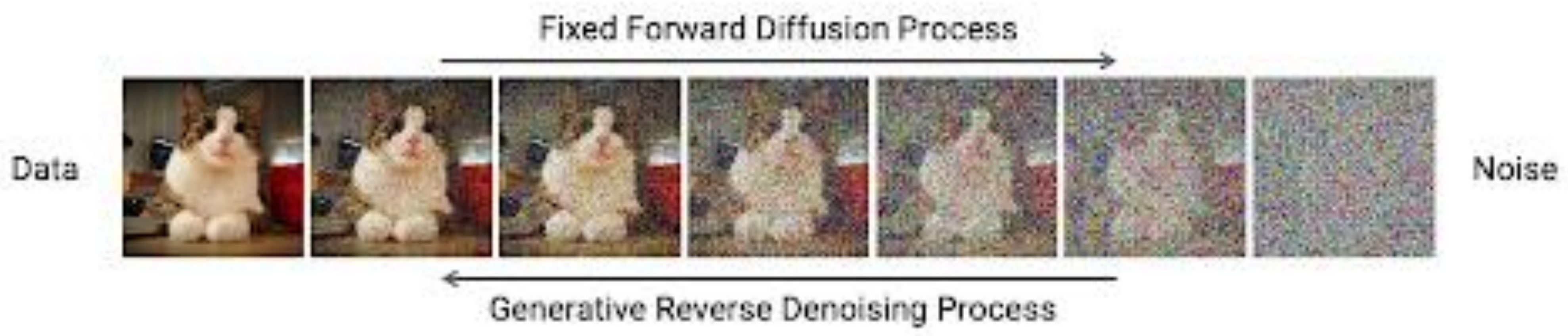




Parton Unfolding Diffusion

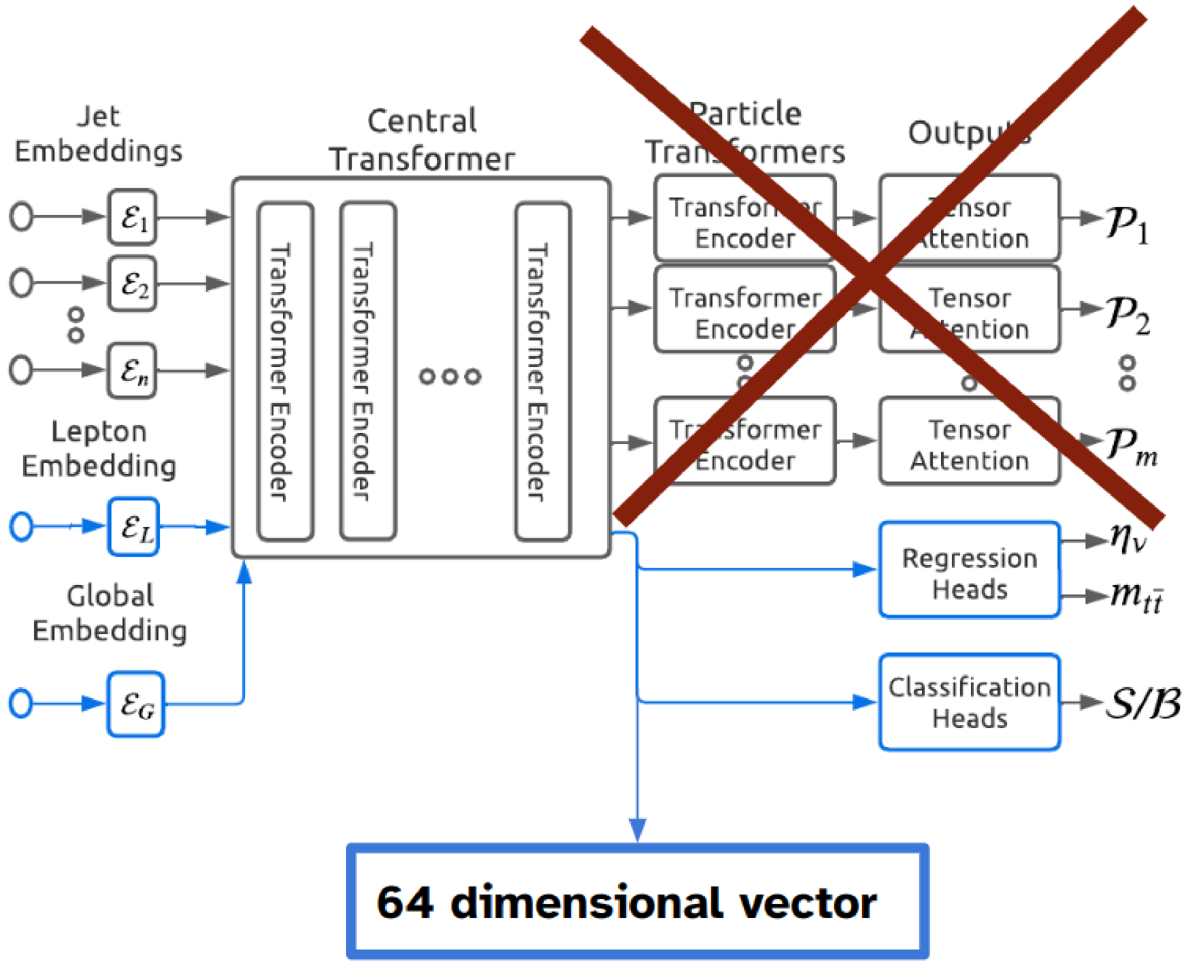
We borrow a popular idea from machine learning: **Diffusion.**
Generate complex distributions *conditioned* on another observation.

Based on a principle of *denoising:* generate new samples by reversing gradual noise.


Fixed Forward Diffusion Process
Data
Noise
Generative Reverse Denoising Process

Learning objective
Given conditioning c and noisy data $z = x + \epsilon$, predict $\hat{\epsilon} = f_\theta(z, c)$

Parton Unfolding Architecture



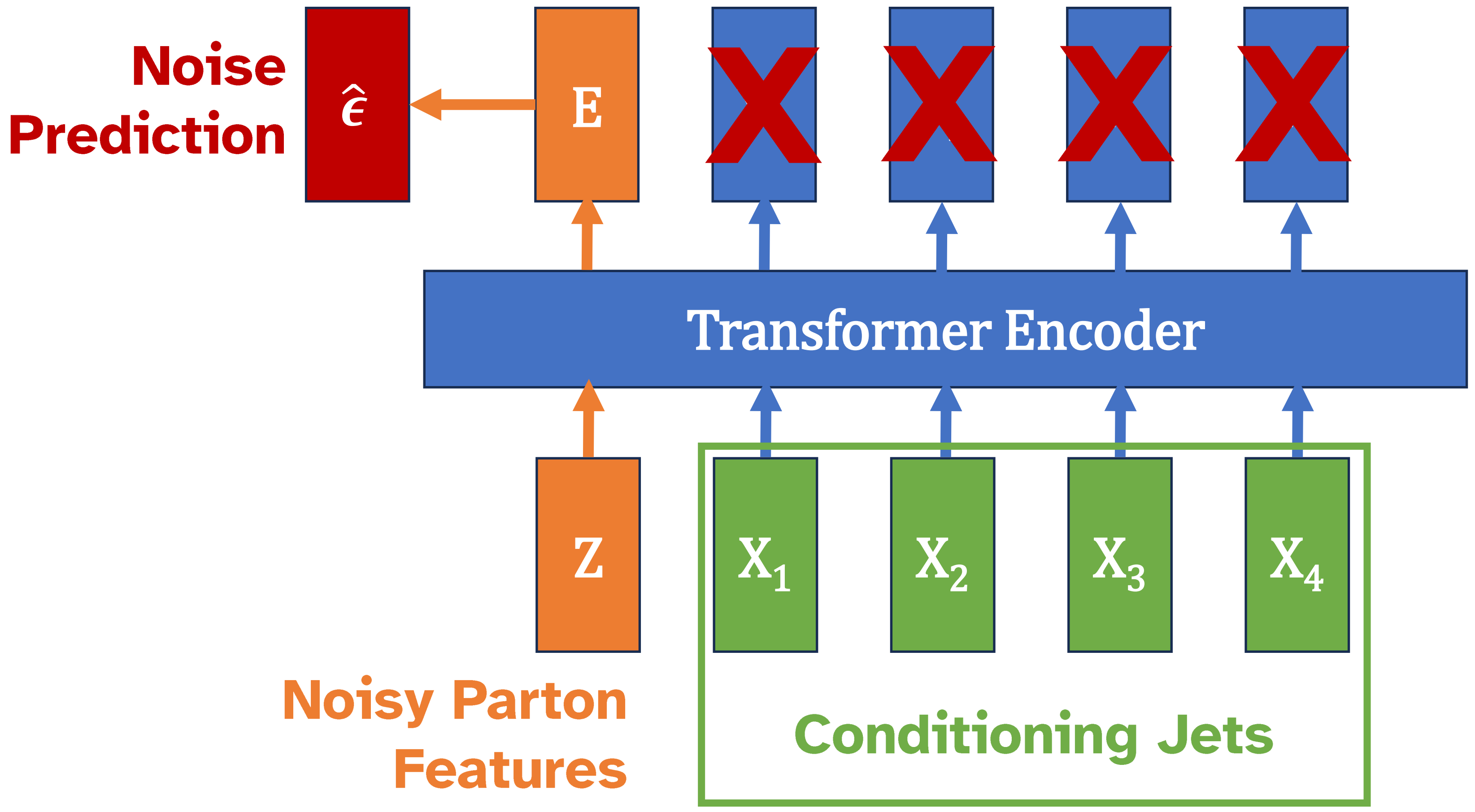
Detector Encoder

Use the special event-level output from SPANet to get an abstract conditional latent vector.

Convert a complex variable-length observation into a fixed-length vector to use for diffusion.

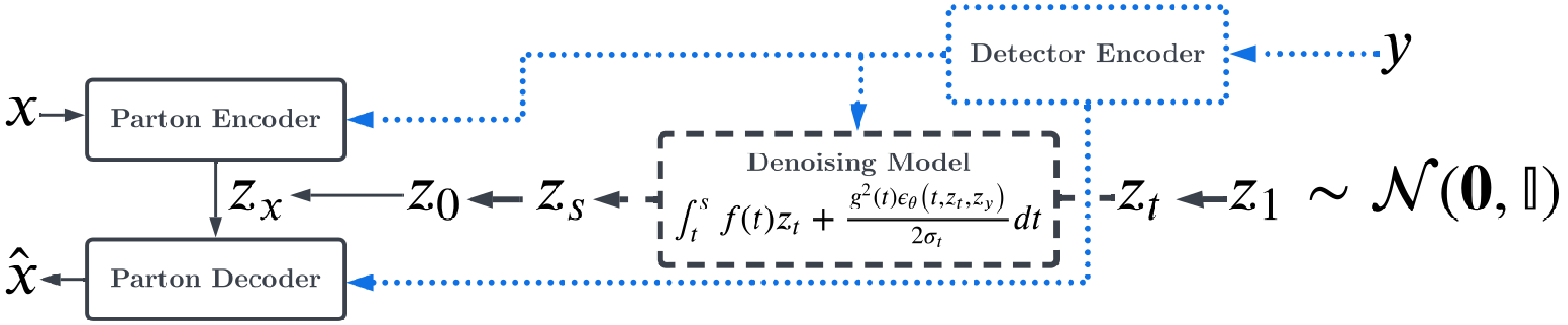
Parton Unfolding Event Vector

- Use same trick as the event-level SPANet outputs.
- Ignore the jet outputs. Don't need them for anything.



Parton Unfolding Diffusion

The whole unfolding framework will generate
Parton configurations conditioned on event vector.


x
Parton Encoder
z_x
$\hat{x}$
Parton Decoder
z_0
z_s
Denoising Model
$\int_t^s f(t)z_t + \frac{g^2(t)\epsilon_\theta(t,z_t,z_y)}{2\sigma_t}dt$
z_t
$z_1 \sim \mathcal{N}(\mathbf{0}, \mathbb{I})$
Detector Encoder
y

Parton Unfolding Results

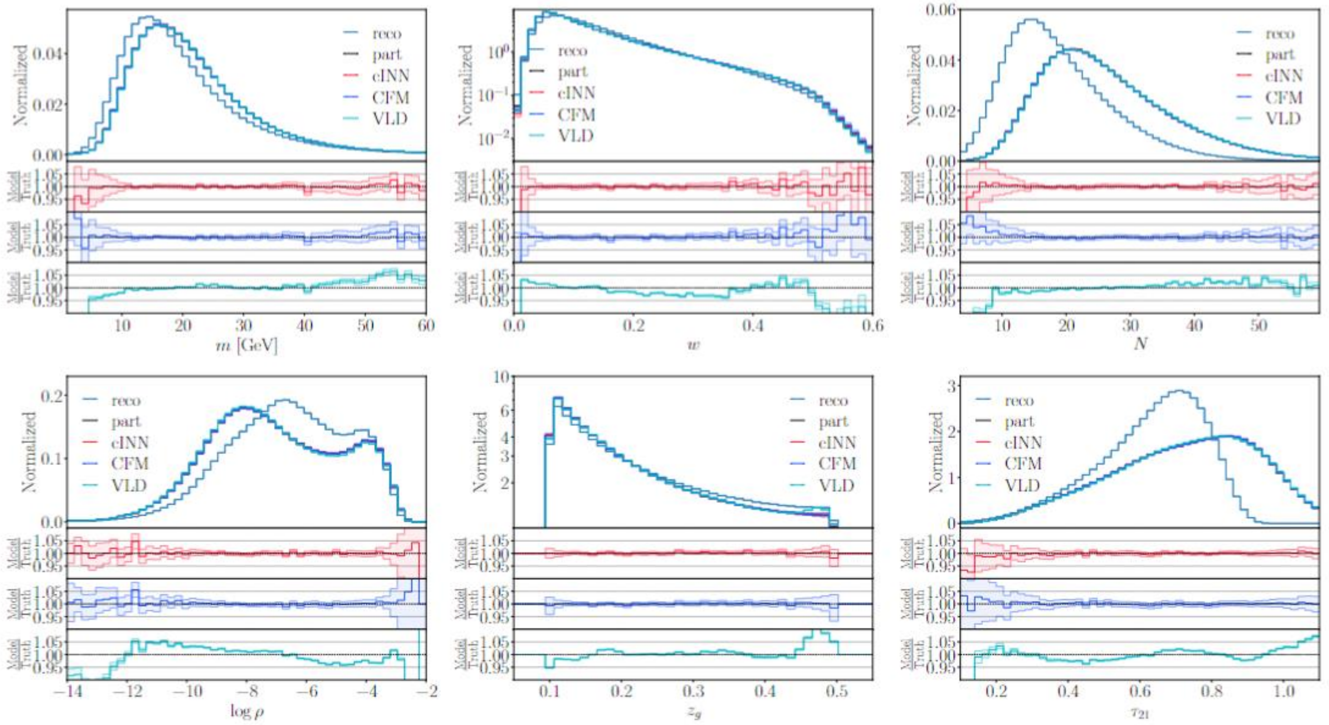
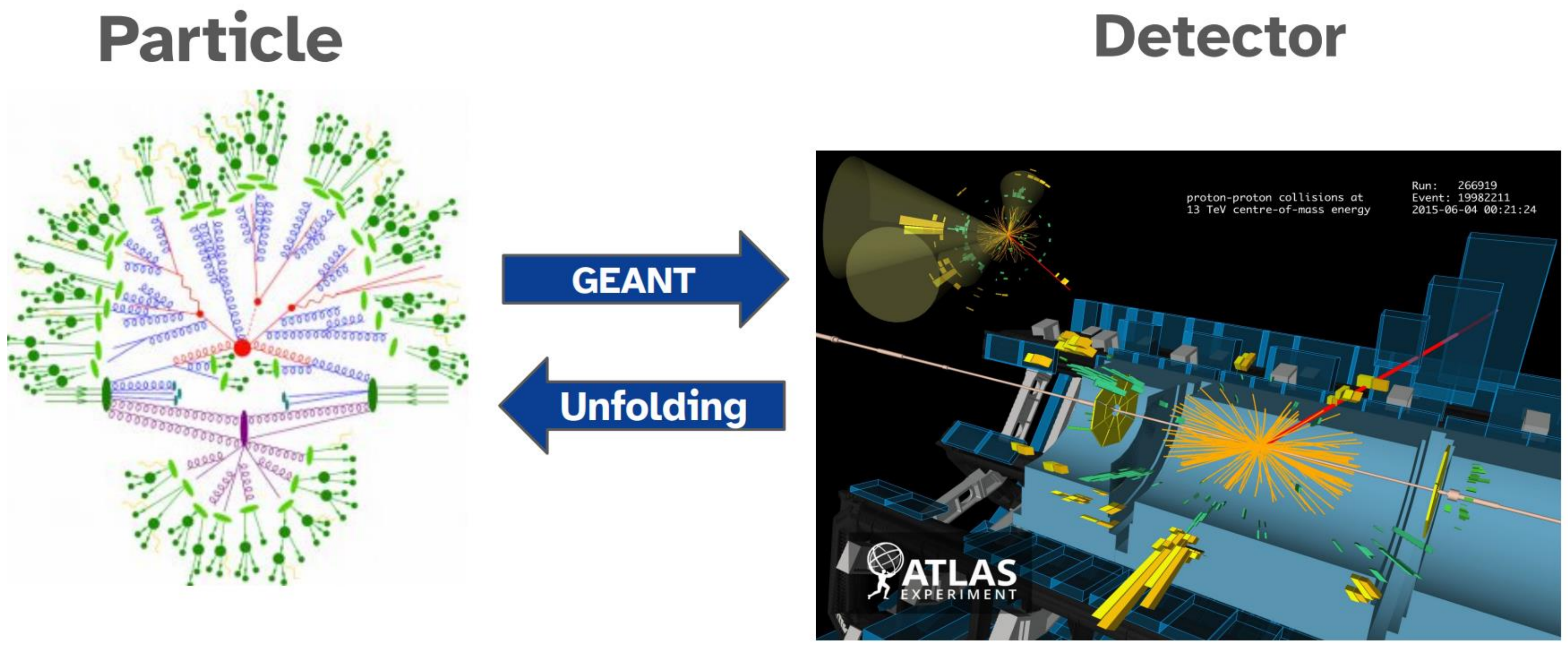


Figure 7: Unfolded distributions from conditional generation, using cINN, CFM and VLD. For cINN and CFM, the Bayesian errors are based on drawing 50 samples, and the MAP estimate is obtained by unfolding each event 30 times. For VLD, we show the bin-wise mean and standard deviation of 33 unfoldings.

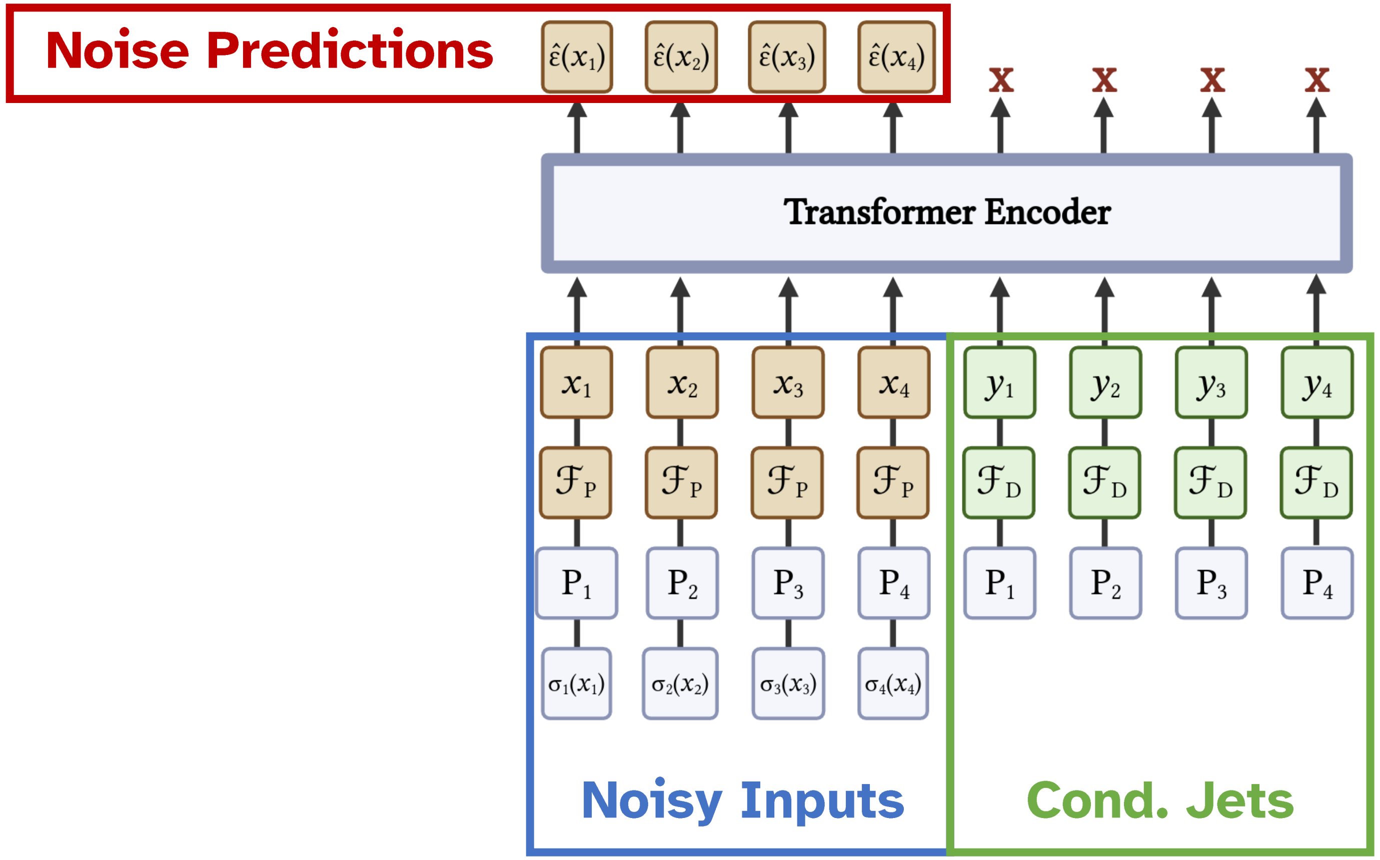
Unfolding Particle Unfolding

- **Invert just the detector response**
- Map a set of jets to a different set of particles
- Set cardinality may not be equal!
- N particles, M observed Jets at detector level, $M \neq N$



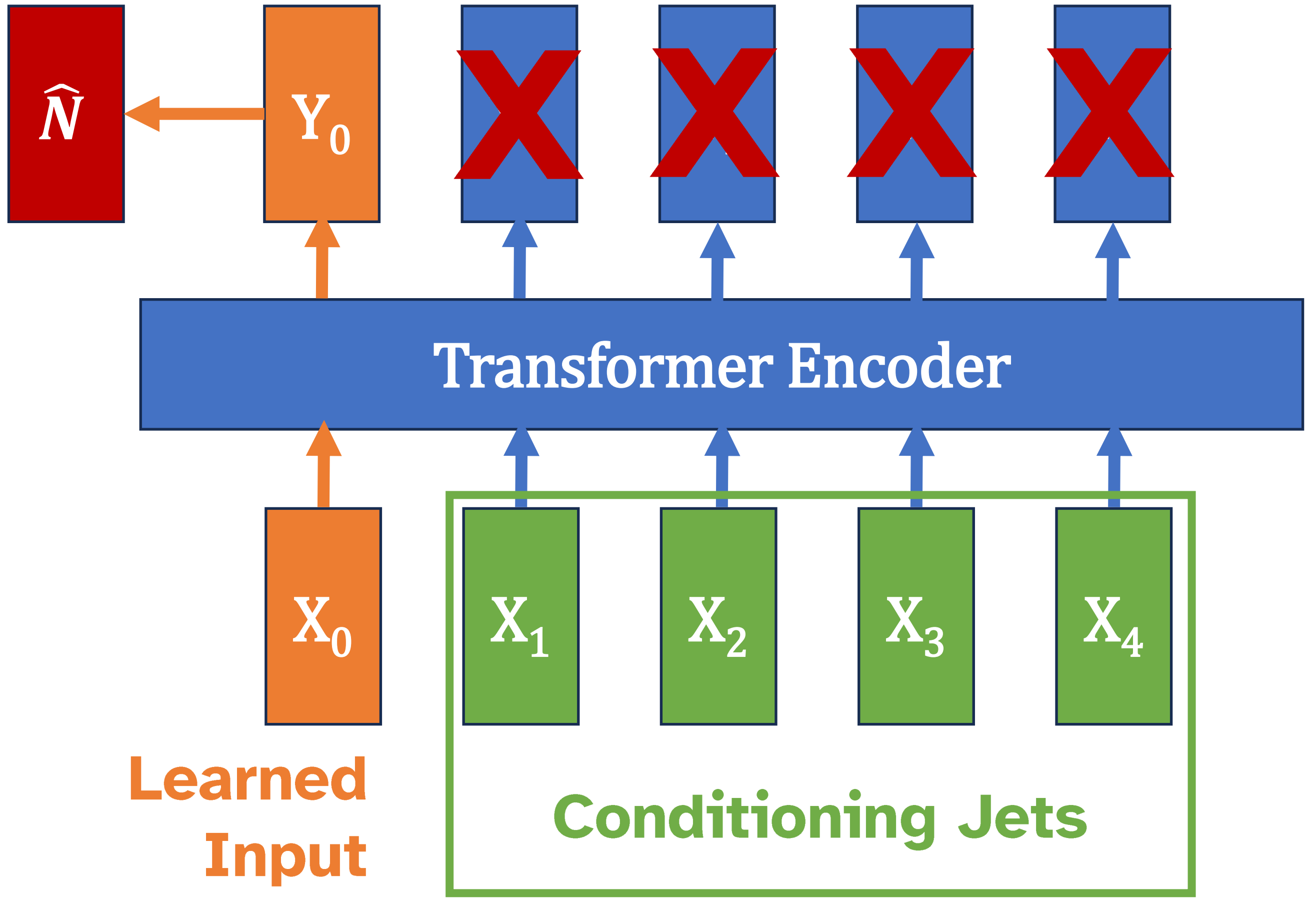
Unfolding Particle Unfolding

Denoising network will now take in and output sets.
Simple trick: Feed everything into the transformer, only use what you need.

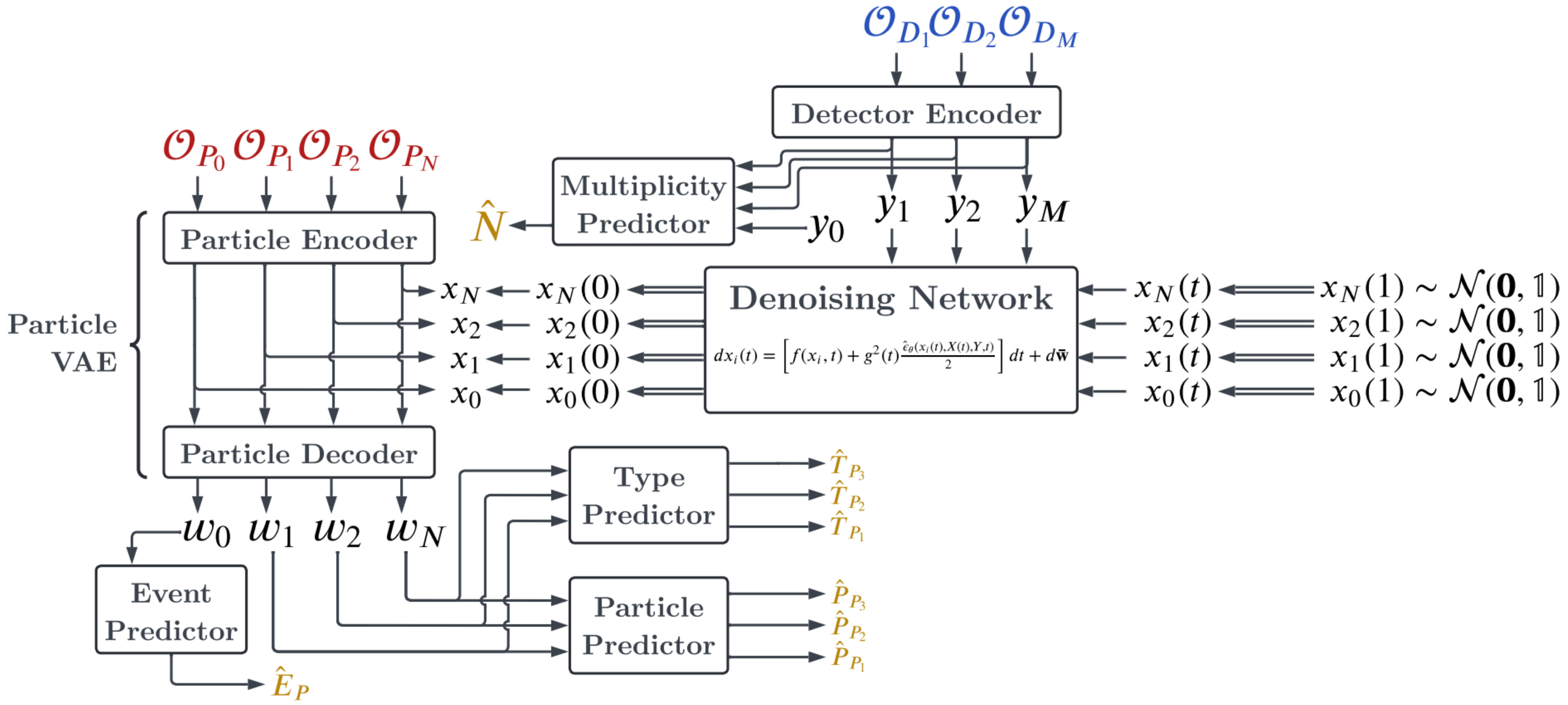


Unfolding Particle Unfolding

How many particles to generate? Learn a multiplicity predictor!



Unfolding Particle Unfolding



$\mathcal{O}_{D_1}$ $\mathcal{O}_{D_2}$ $\mathcal{O}_{D_M}$
Detector Encoder
$\mathcal{O}_{P_0}$ $\mathcal{O}_{P_1}$ $\mathcal{O}_{P_2}$ $\mathcal{O}_{P_N}$
Multiplicity Predictor
$\hat{N}$
y_0 y_1 y_2 y_M
Particle Encoder
Particle VAE
Denoising Network
$dx_i(t) = \left[f(x_i,t) + g^2(t)\frac{\hat{\epsilon}_\theta(x_i(t),X(t),Y,t)}{2}\right]dt + d\bar{\mathbf{w}}$
x_N $x_N(0)$ $x_N(t)$ $x_N(1) \sim \mathcal{N}(\mathbf{0}, \mathbb{1})$
x_2 $x_2(0)$ $x_2(t)$ $x_2(1) \sim \mathcal{N}(\mathbf{0}, \mathbb{1})$
x_1 $x_1(0)$ $x_1(t)$ $x_1(1) \sim \mathcal{N}(\mathbf{0}, \mathbb{1})$
x_0 $x_0(0)$ $x_0(t)$ $x_0(1) \sim \mathcal{N}(\mathbf{0}, \mathbb{1})$
Particle Decoder
w_0 w_1 w_2 w_N
Type Predictor
$\hat{T}_{P_3}$ $\hat{T}_{P_2}$ $\hat{T}_{P_1}$
Event Predictor
$\hat{E}_P$
Particle Predictor
$\hat{P}_{P_3}$ $\hat{P}_{P_2}$ $\hat{P}_{P_1}$