Learning Symmetry-Independent Jet Representation via Jet Joint Embedding Predictive Architecture

Haoyang "Billy" Li[†], Subash Katel[†], Zihan Zhao[†], Farouk Mokhtar, Javier Duarte (UCSD)
Raghav Kansal (Caltech)

ML4Jets 2024
Nov 5

†: equal contribution

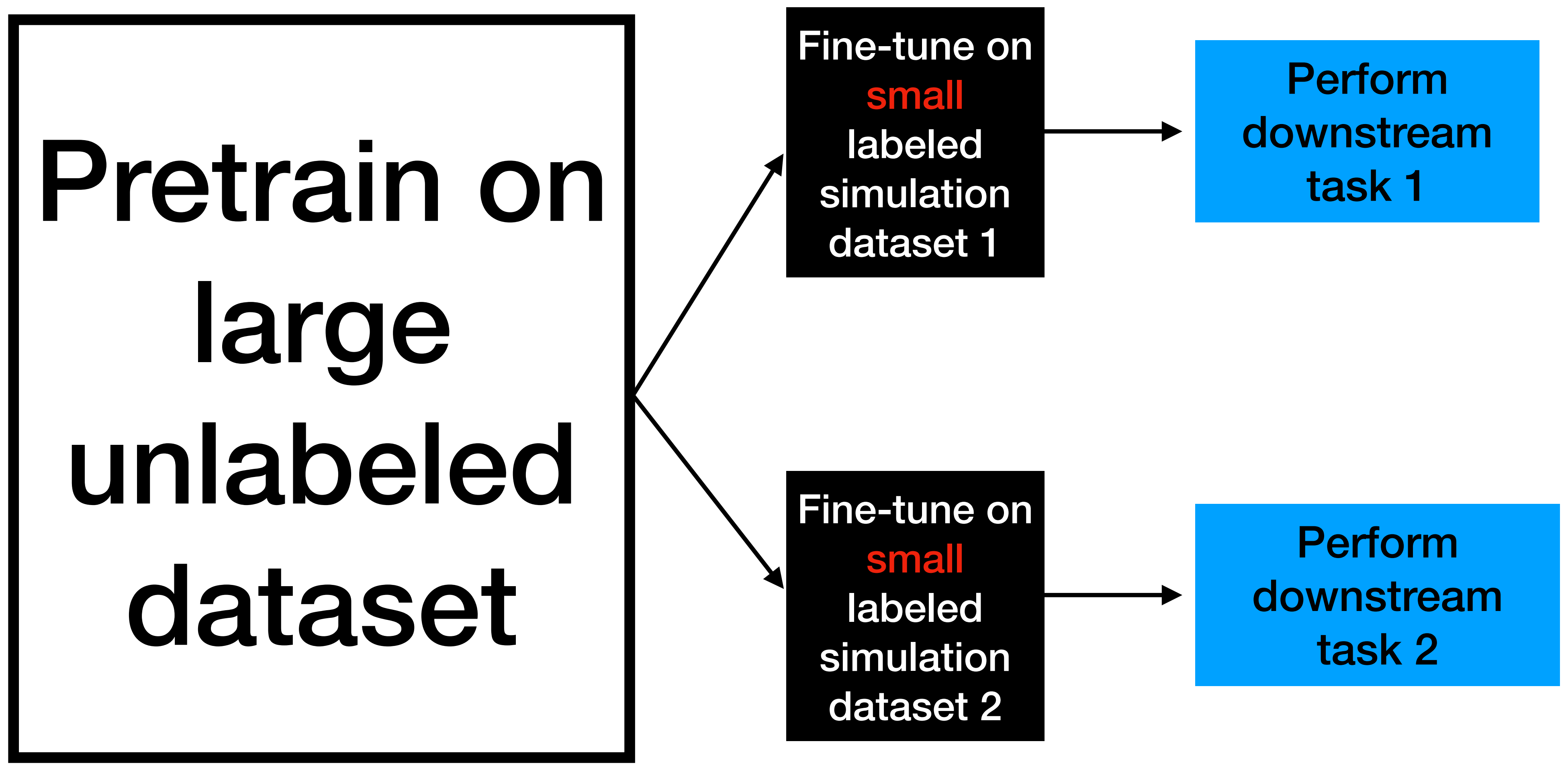
Outline

- Motivation
- Introduction to JEPA
- Our J-JEPA approach
- Dataset
- Pretraining + fintuning setup
- Pretraining result
- Pretraining + fine-tuning result
- Ongoing and Future work

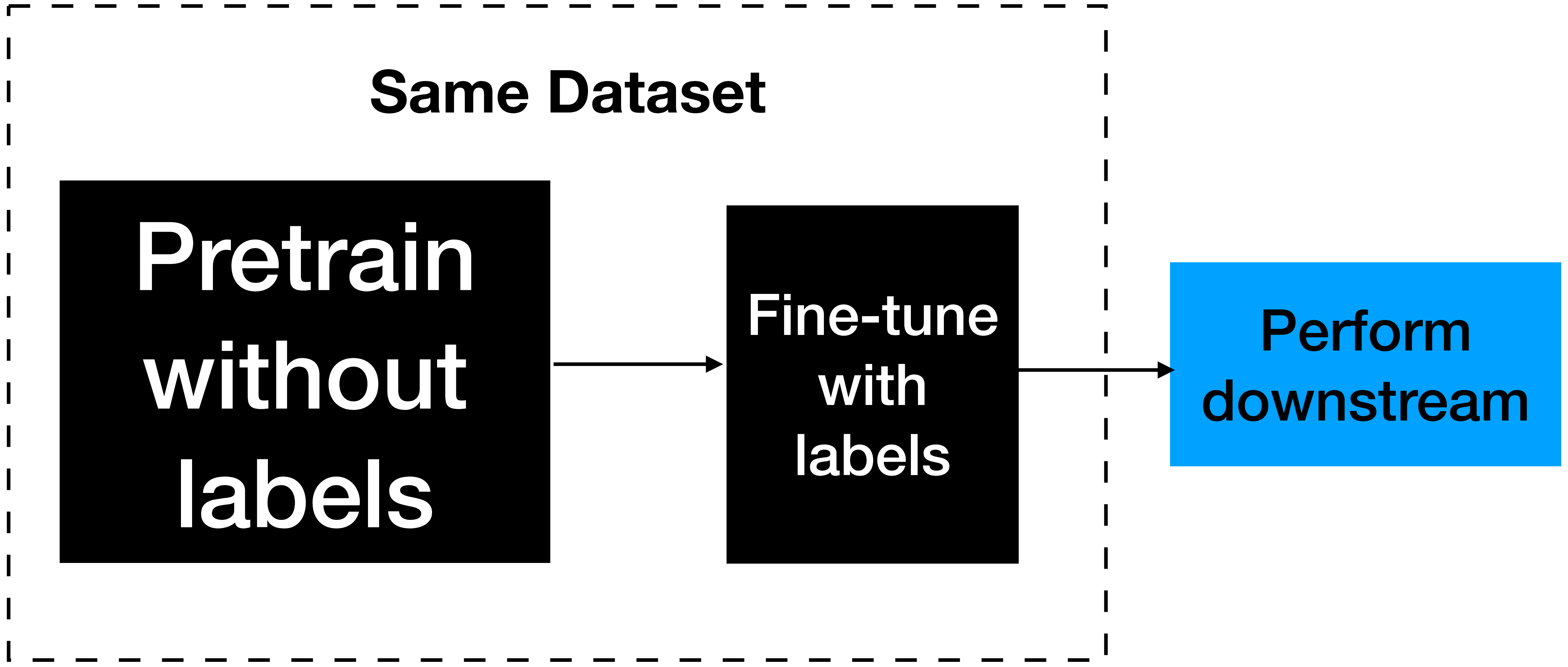
Motivations for Self-Supervised Learning (SSL)

Learning without labels

- Self-Supervised Learning: A type of machine learning where models learn useful features and representations from unlabeled data
- To learn effectively (like human), system must learn these representations directly from unlabeled data such as images or sounds, rather than from manually assembled labeled datasets.
- With the HL-LHC upgrade [1] in the near future, we will need to simulate an order of magnitude more events with a more complicated detector geometry to keep up with the recorded data [2].



SSL for foundation model

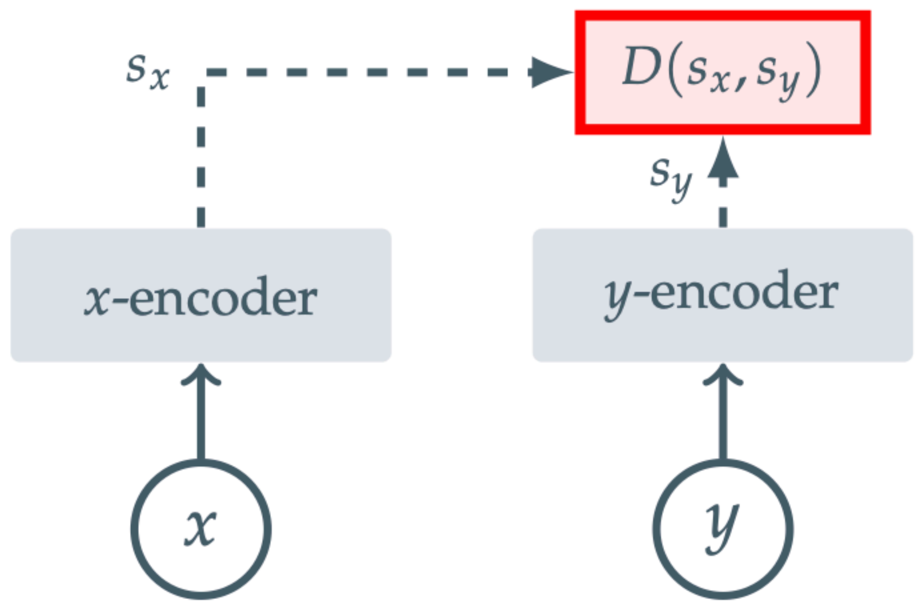


SSL stage in a mixed training

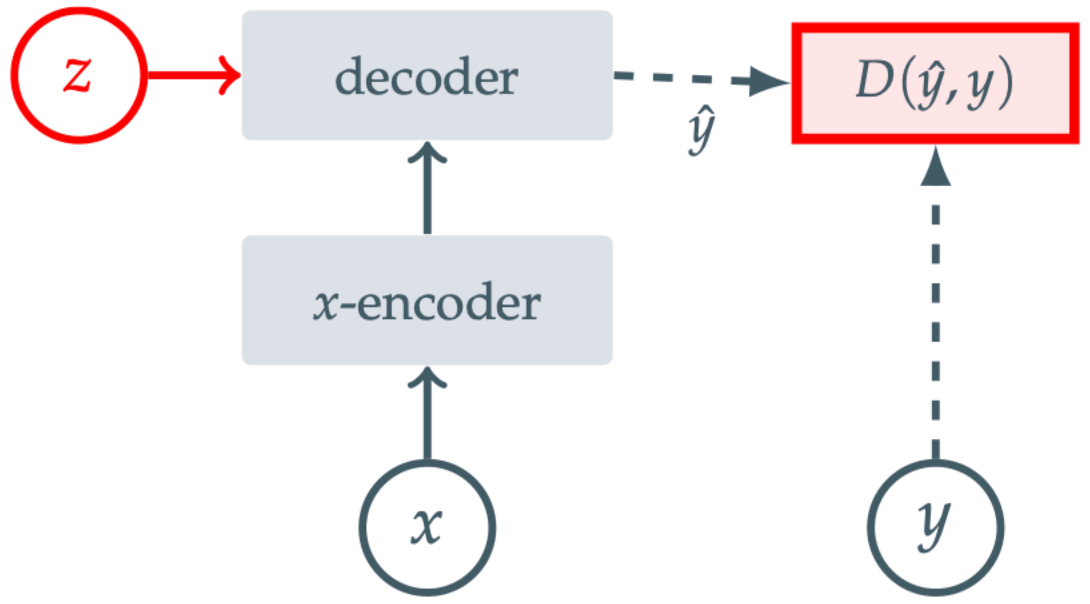
1. [HL-LHC] https://arxiv.org/abs/1705.08830
2. [Computing for HL LHC] https://doi.org/10.1051/epjconf/201921402036

JEPA: Different SSL Architectures

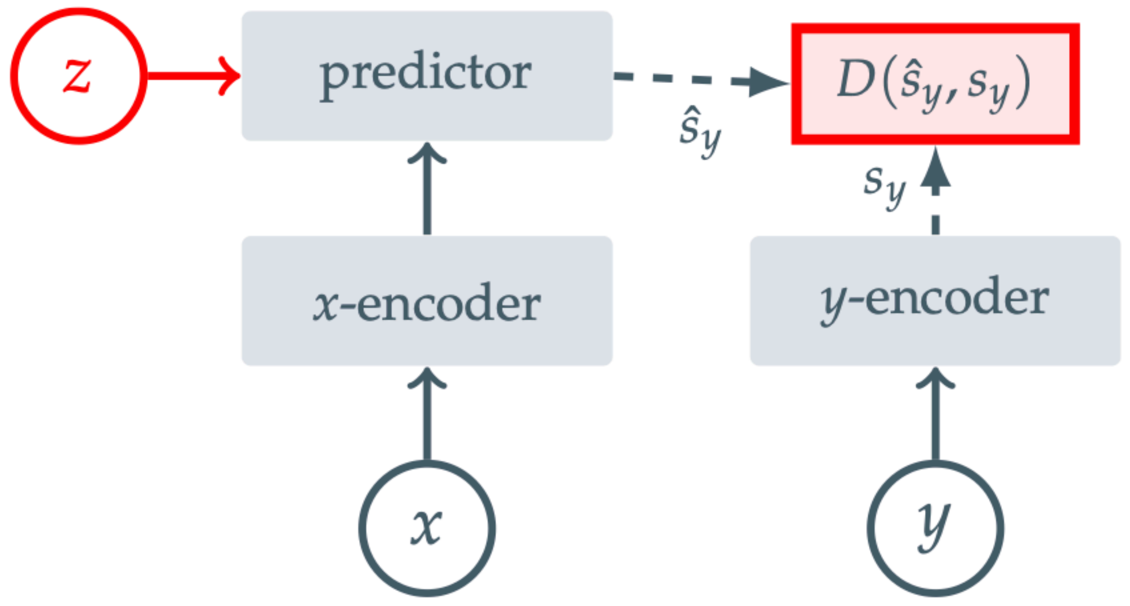
From the perspective of computer vision



(a) **Joint-Embedding Architecture**



(b) **Generative Architecture**

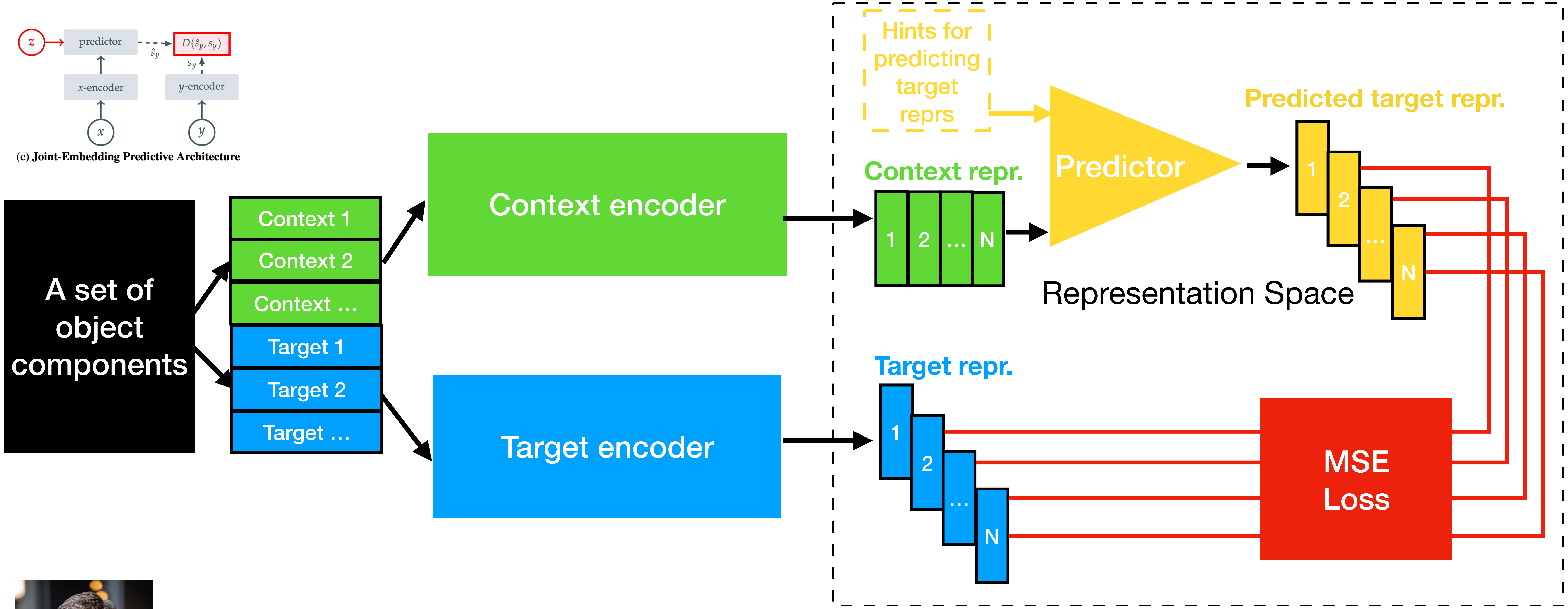


(c) **Joint-Embedding Predictive Architecture**

- Difference between JEPA and (a): JEPA is augmentation free and predictive
- Difference between JEPA and (b): JEPA predicts in the latent space

Assran et al., "Self-supervised learning from images with a joint-embedding predictive architecture", 2023.

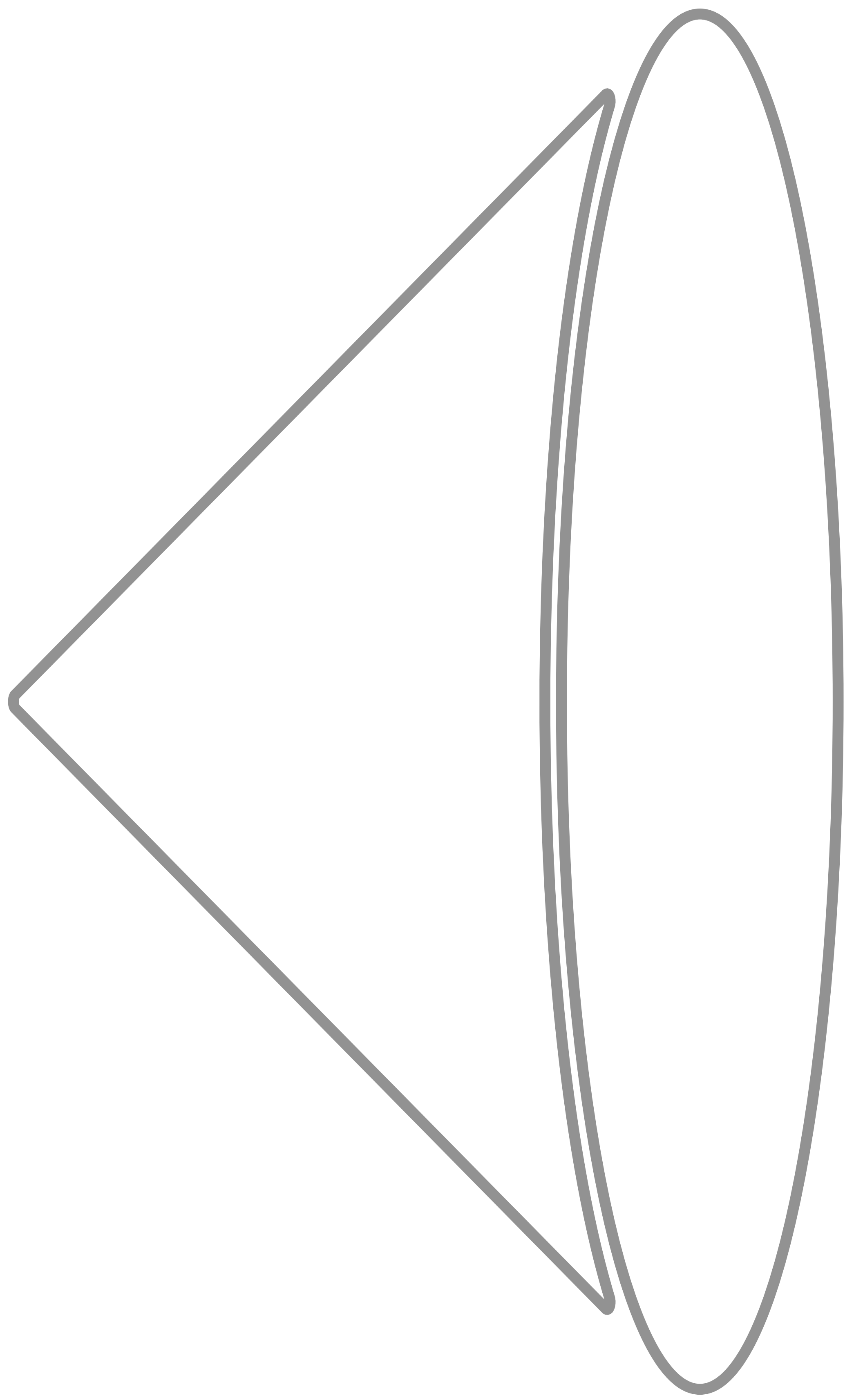
JEPA: Joint Embedding Predictive Architecture



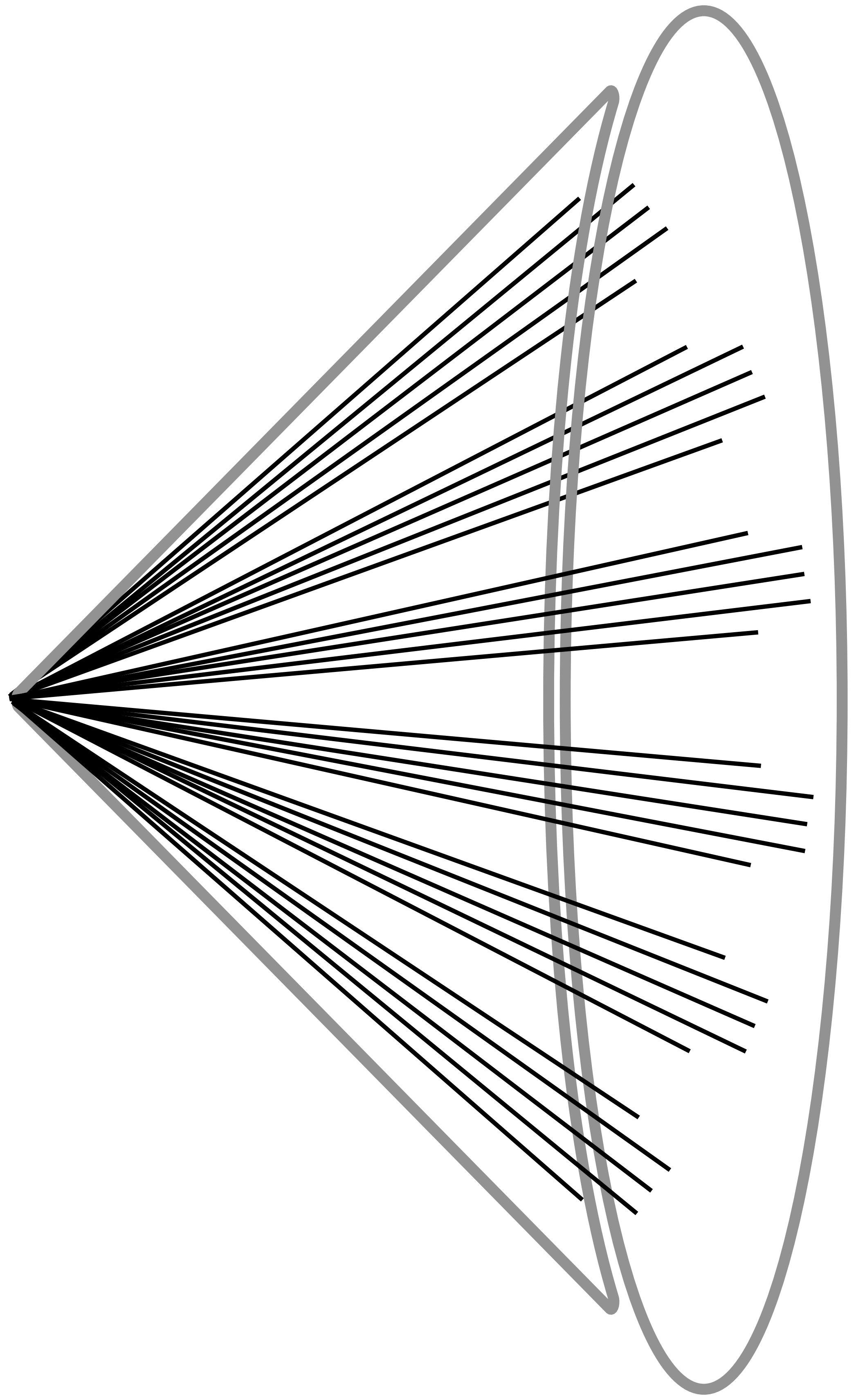
- Predict the masked parts in the representation space
- Augmentation free to minimize bias

J (Jet) - JEPA

An AK8 Jet



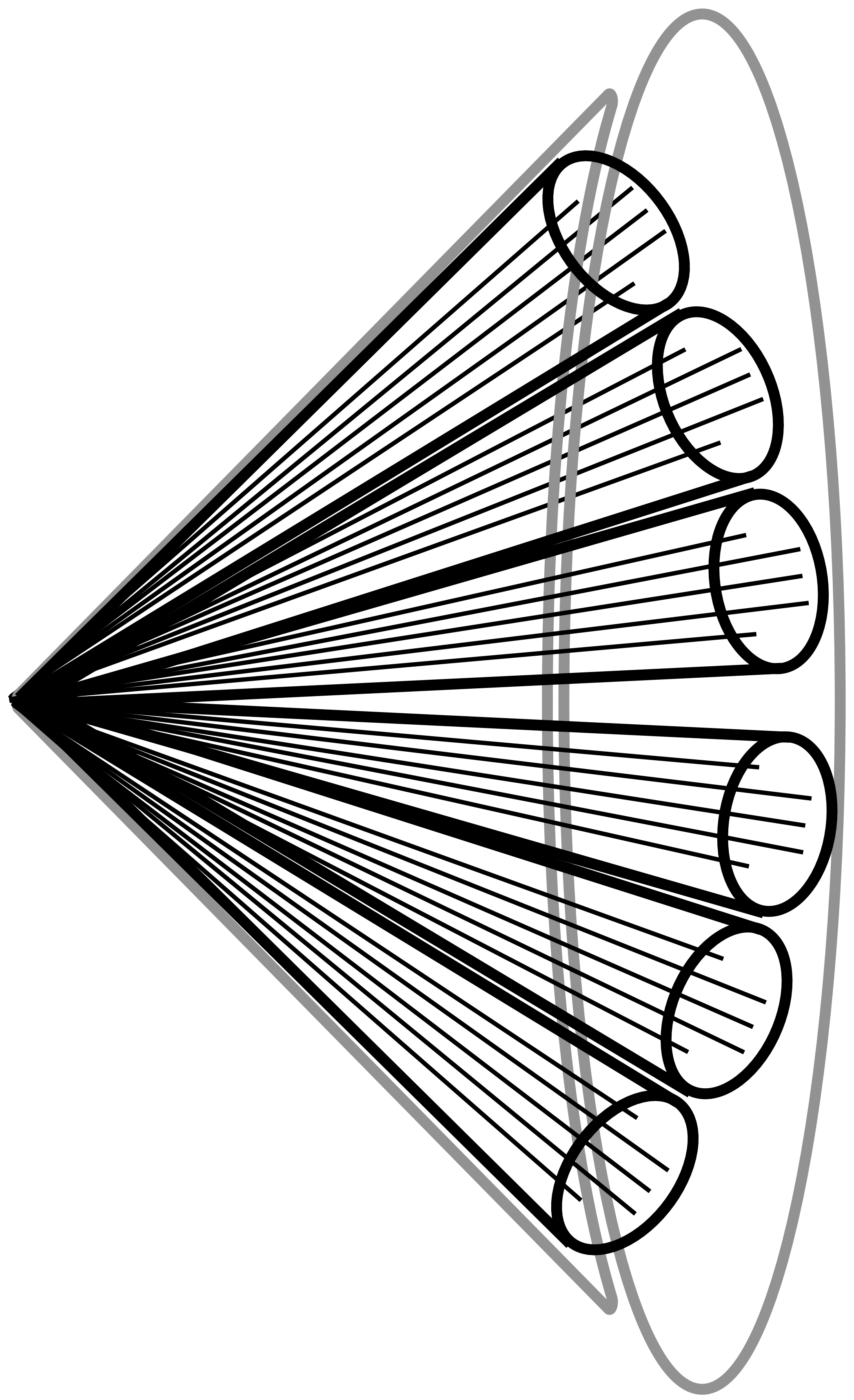
An AK8 Jet



J-JEPA

Cluster subjets with radius 0.2

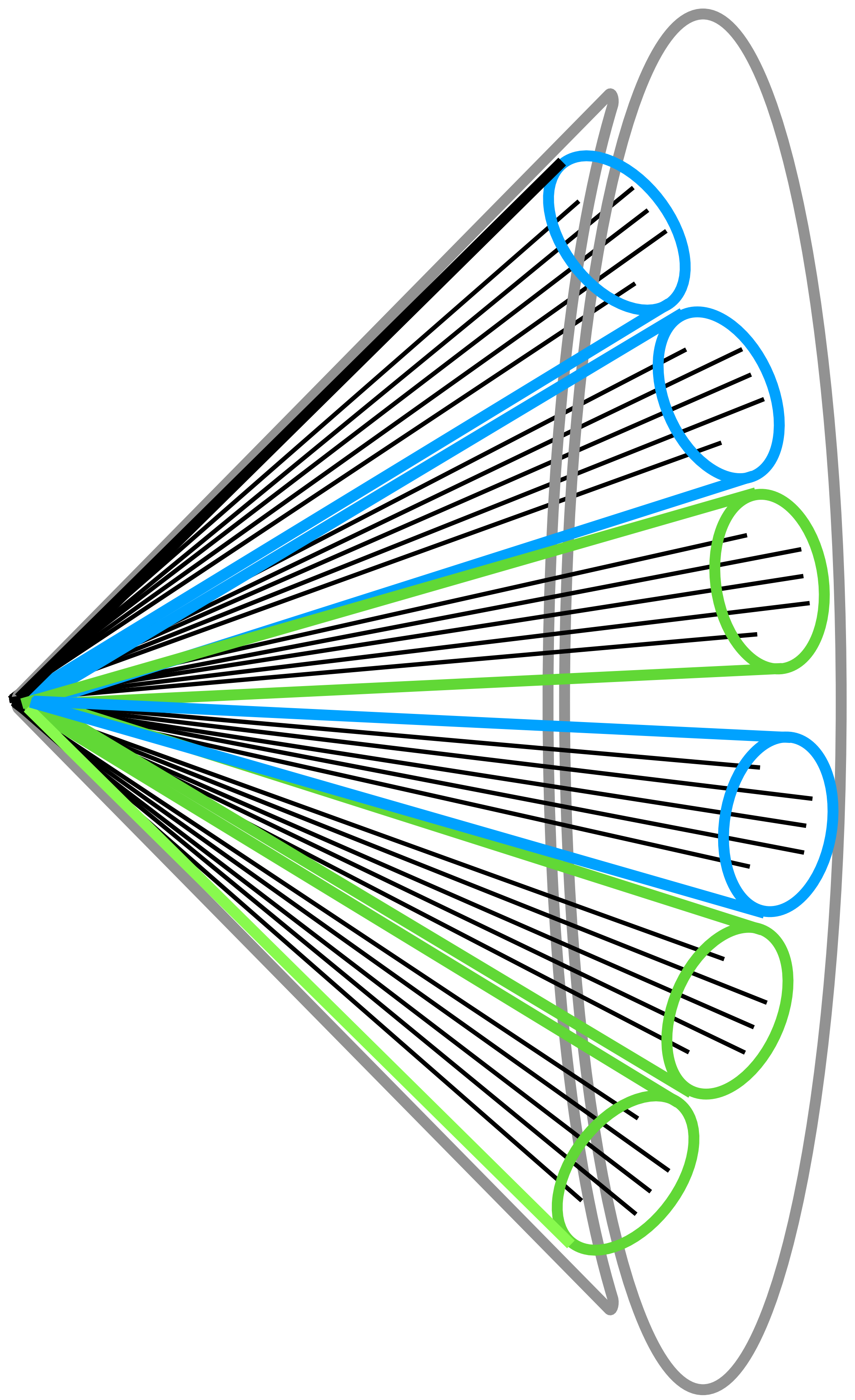
An AK8 Jet



J-JEPA: Define Target and Context Subjets

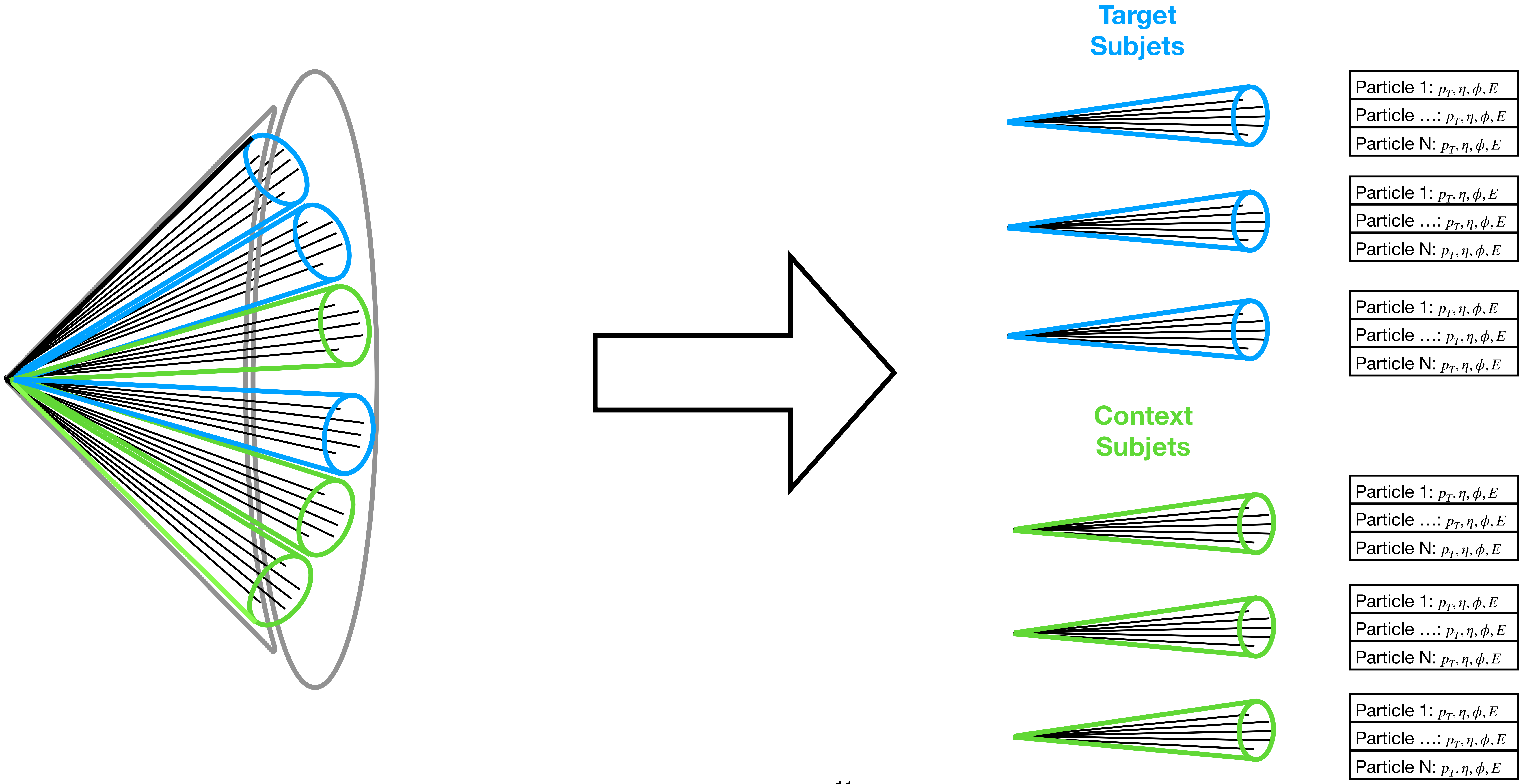
Randomly divide subjets into target/context categories

An AK8 Jet



J-JEPA: Define Target and Context Subjets

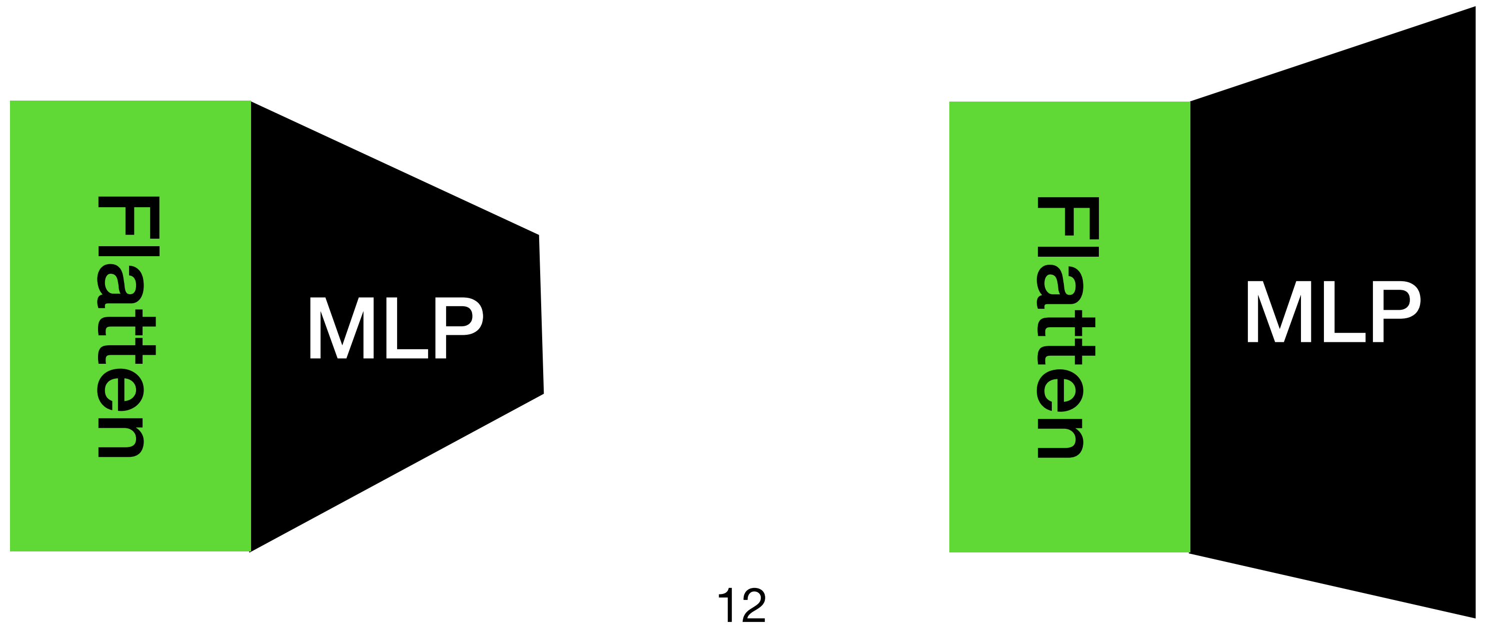
Randomly divide subjets into target/context categories



J-JEPA: Subjet Embedding Layer (SEL)

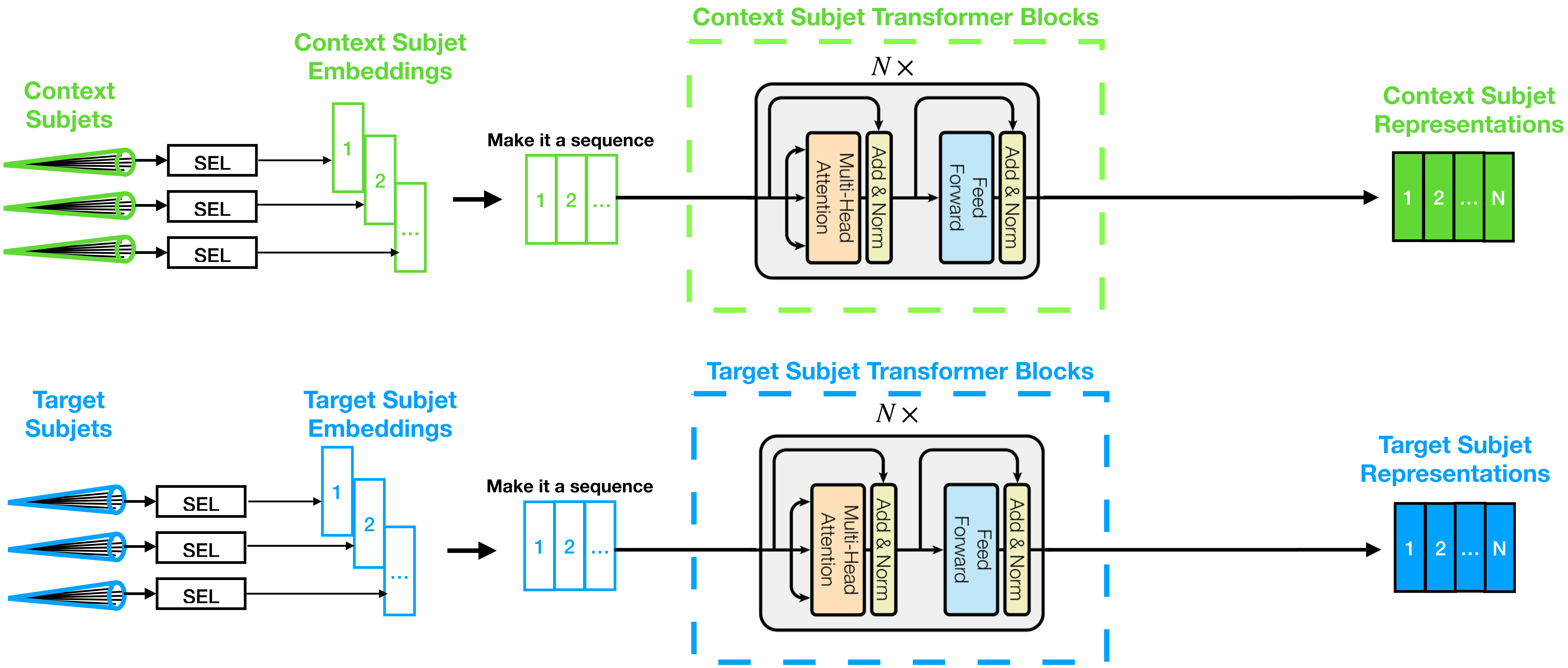
Each subjet creates its embedding independently

Other options:



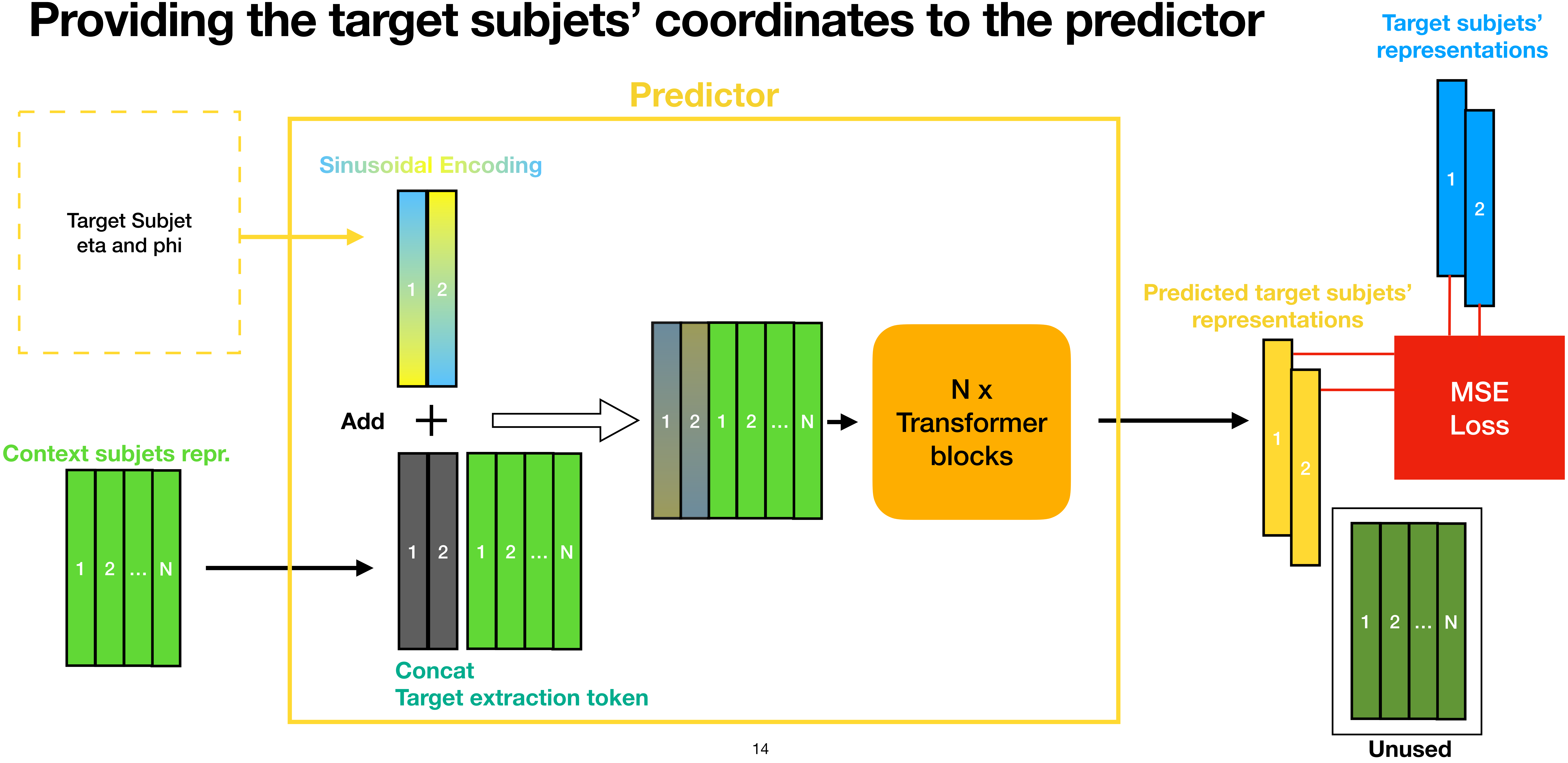
J-JEPA: Calculate Subjet Representations

Using Transformer Encoder Blocks

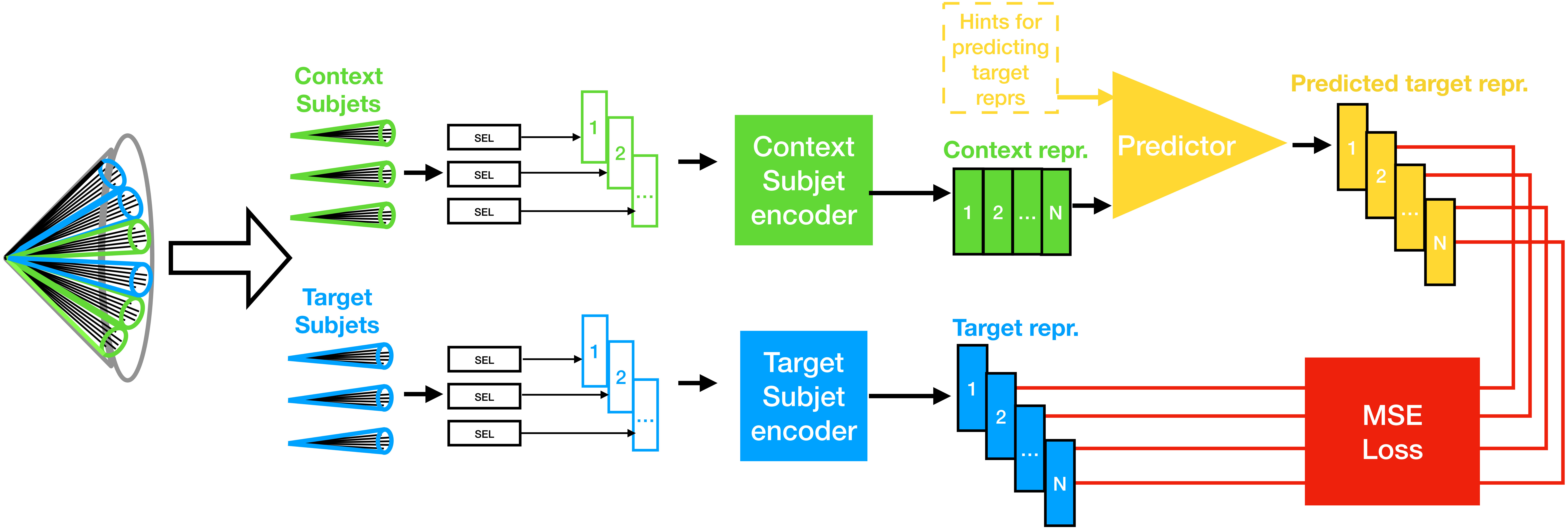


J-JEPA: Predict in the Representation Space

Providing the target subjets' coordinates to the predictor



J-JEPA: Pretraining

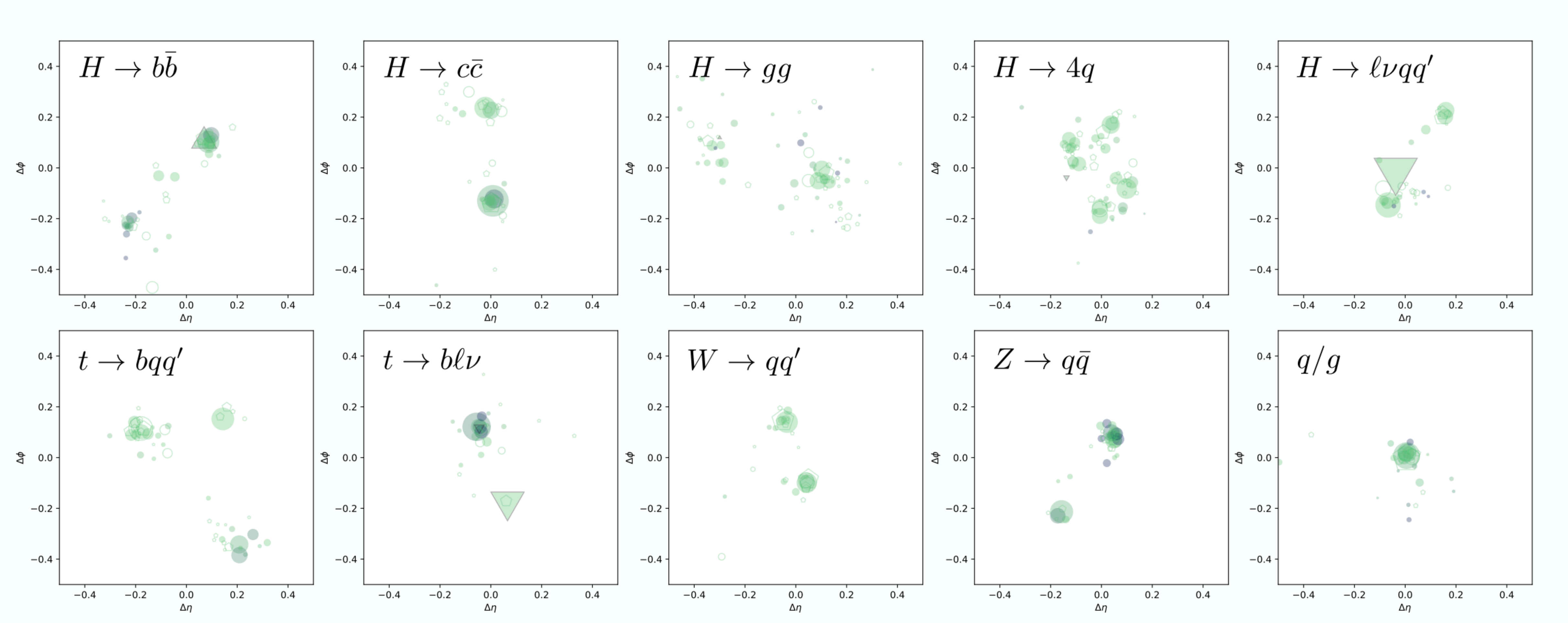


Datasets

We use JetClass for pretraining and TopTagging for finetuning

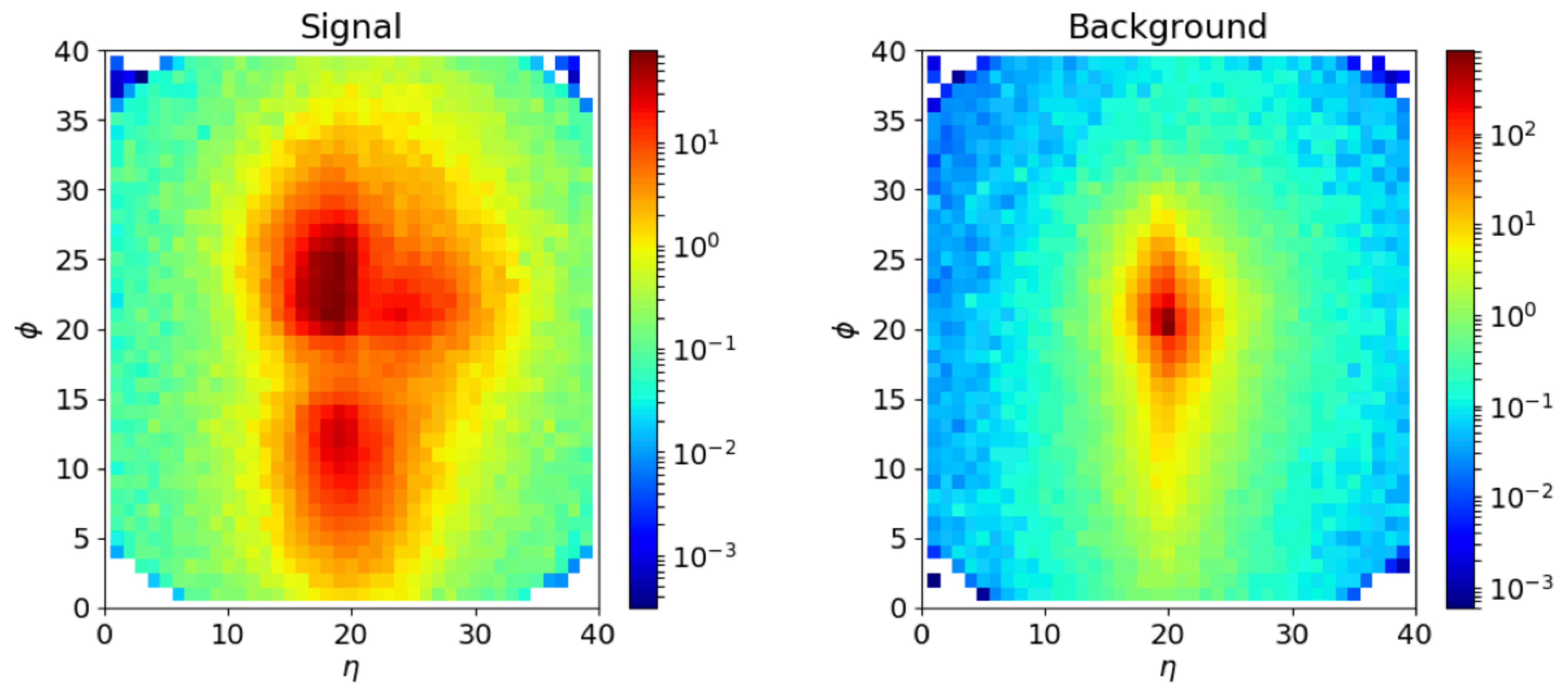
Dataset name	Size	Description	Portions we used	Role in transfer learning
JetClass	100 Million AK8 Jets	Contains 10 classes of jets	500K Top jets 500k q/g jets	Stand in for the large pretaining unlabeled dataset
Top Tagging	1.2 Million AK8 Jets	Only Top and QCD jets	760K mixed jets*	Stand in for the small fine-tuning dataset

* We only used jets with more than 10 subjets



JetClass Dataset

2202.03772

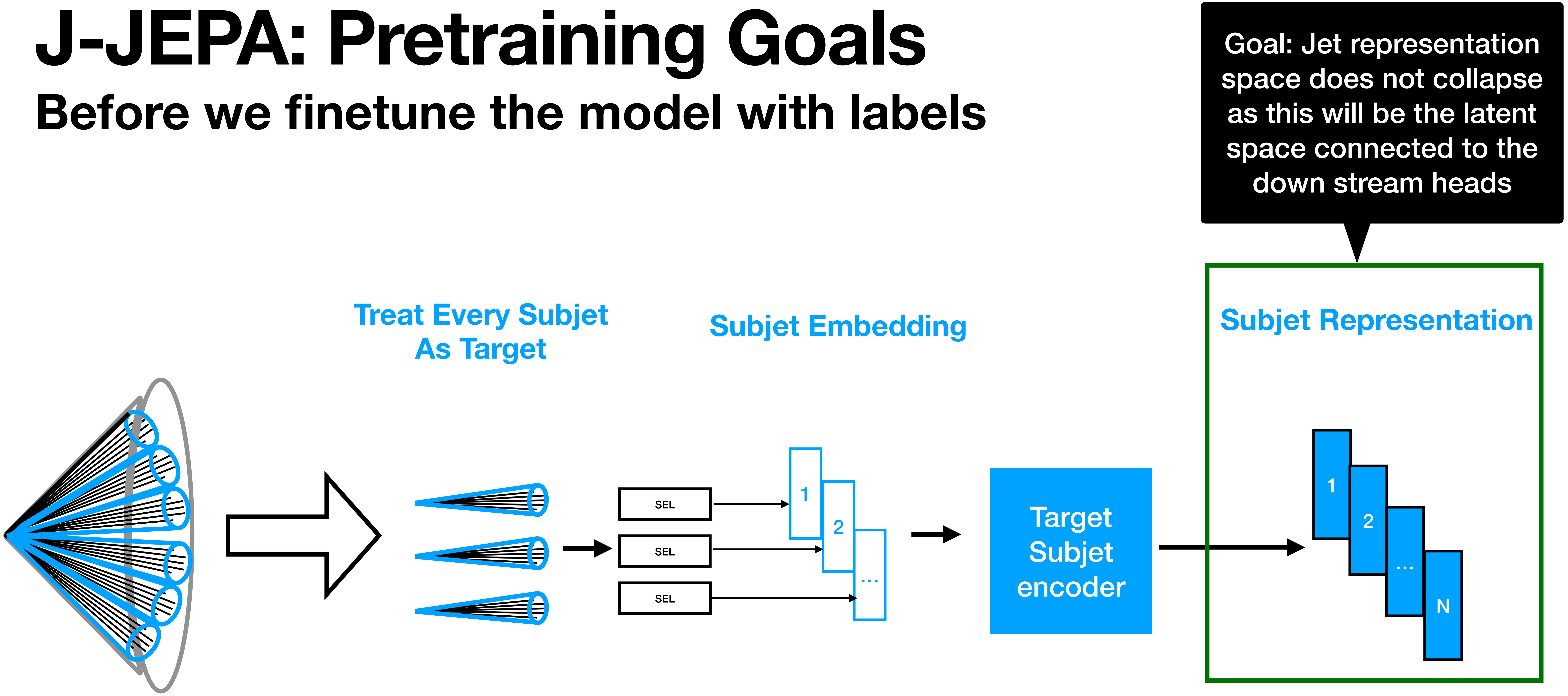


Top Tagging Dataset

1902.09914

J-JEPA: Pretraining Goals

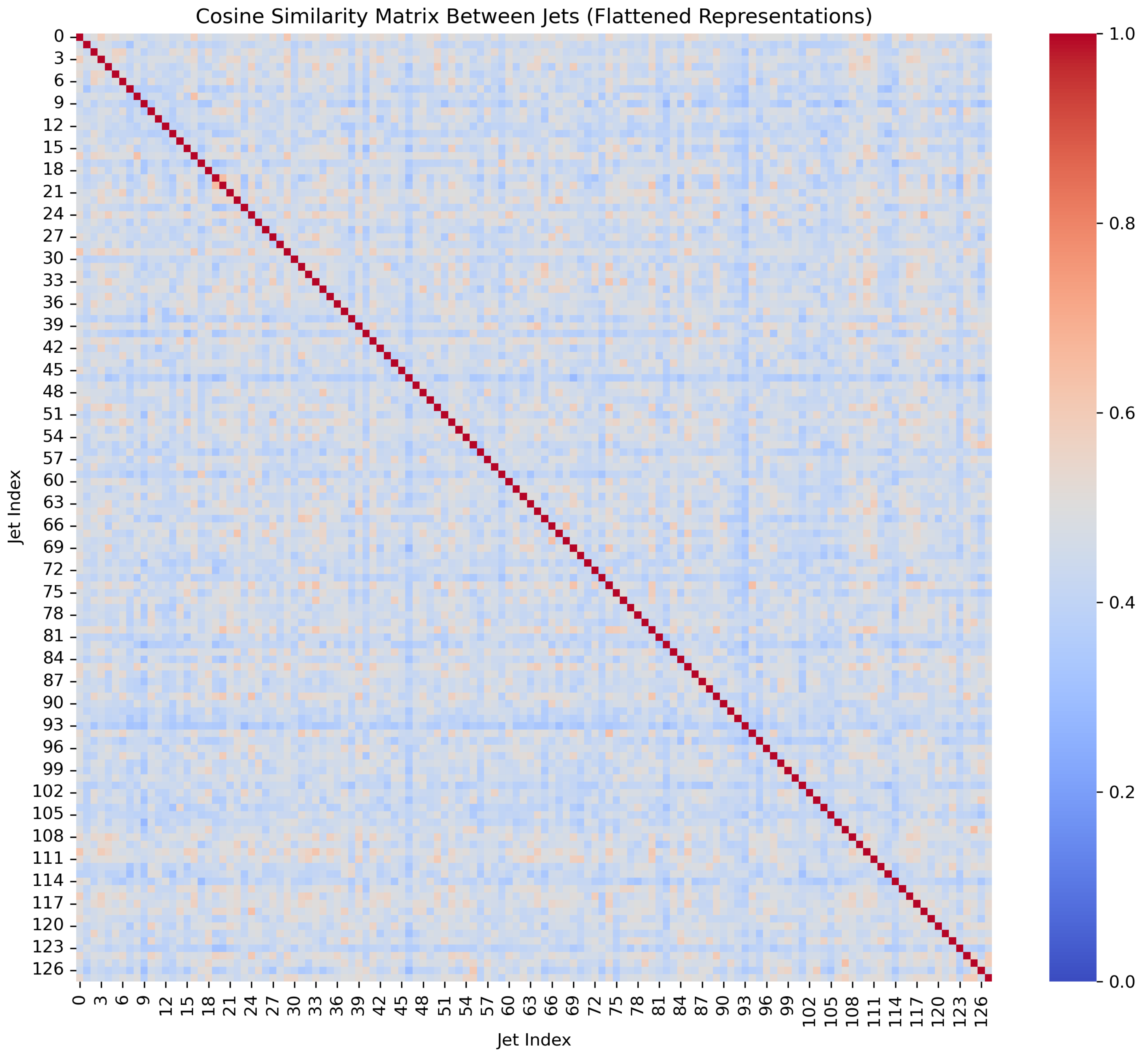
Before we finetune the model with labels



Information collapse: The model fails to capture the meaningful variations in the data, leading to poor performance in tasks like classification or regression.

Latent after Pre-training: Not Collapsing

J-JEPA model learned a diverse latent space



Let A be the features of Jet 1, and B be the features of Jet 2, then the cosine similarity is defined as

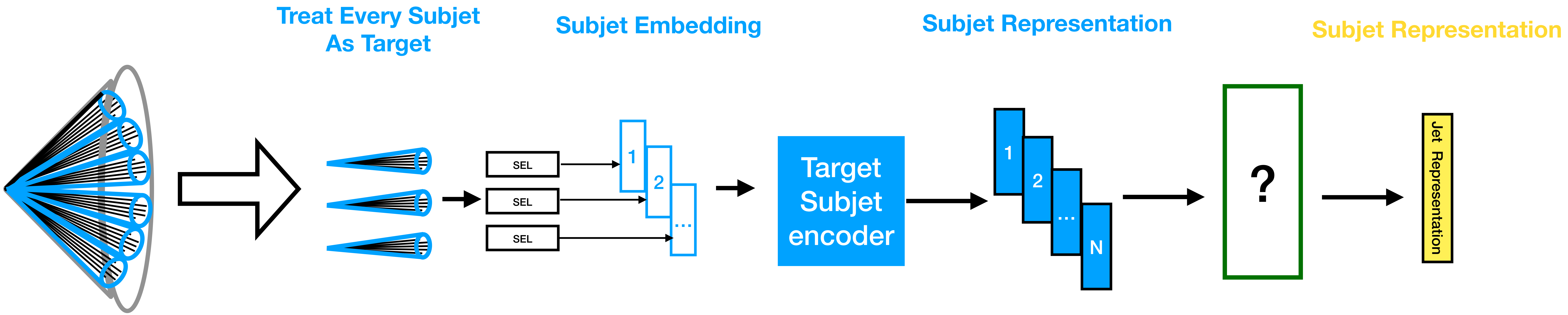
$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|}$$

1. Randomly select 128 Jets.
2. Represent each jet by their flattened subjet representations
3. Calculate cosine similarity between each pair of jets

Average Cosine Similarity: 0.457

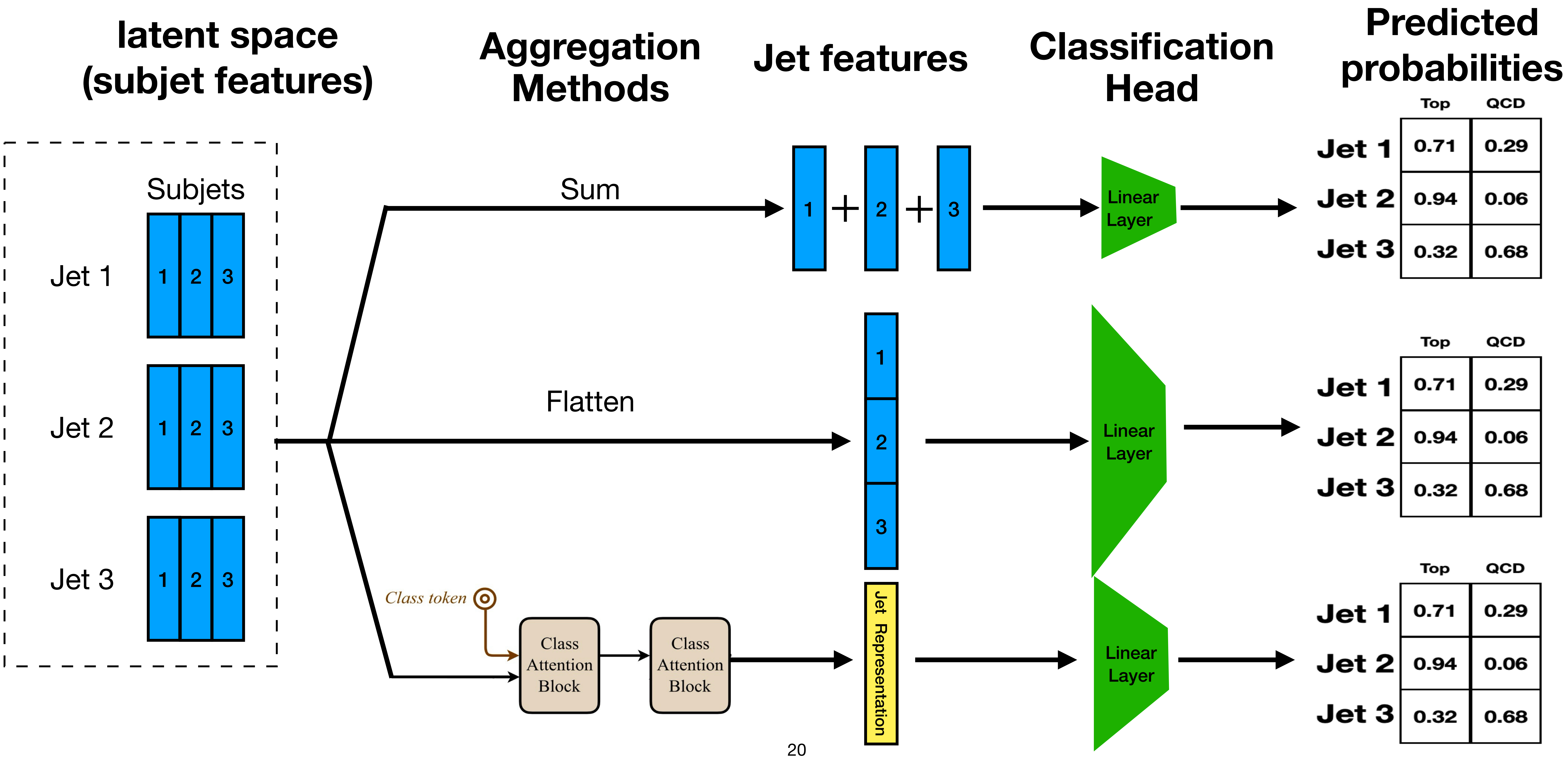
J-JEPA: Finetuning Setup

From subjet representation to jet representation



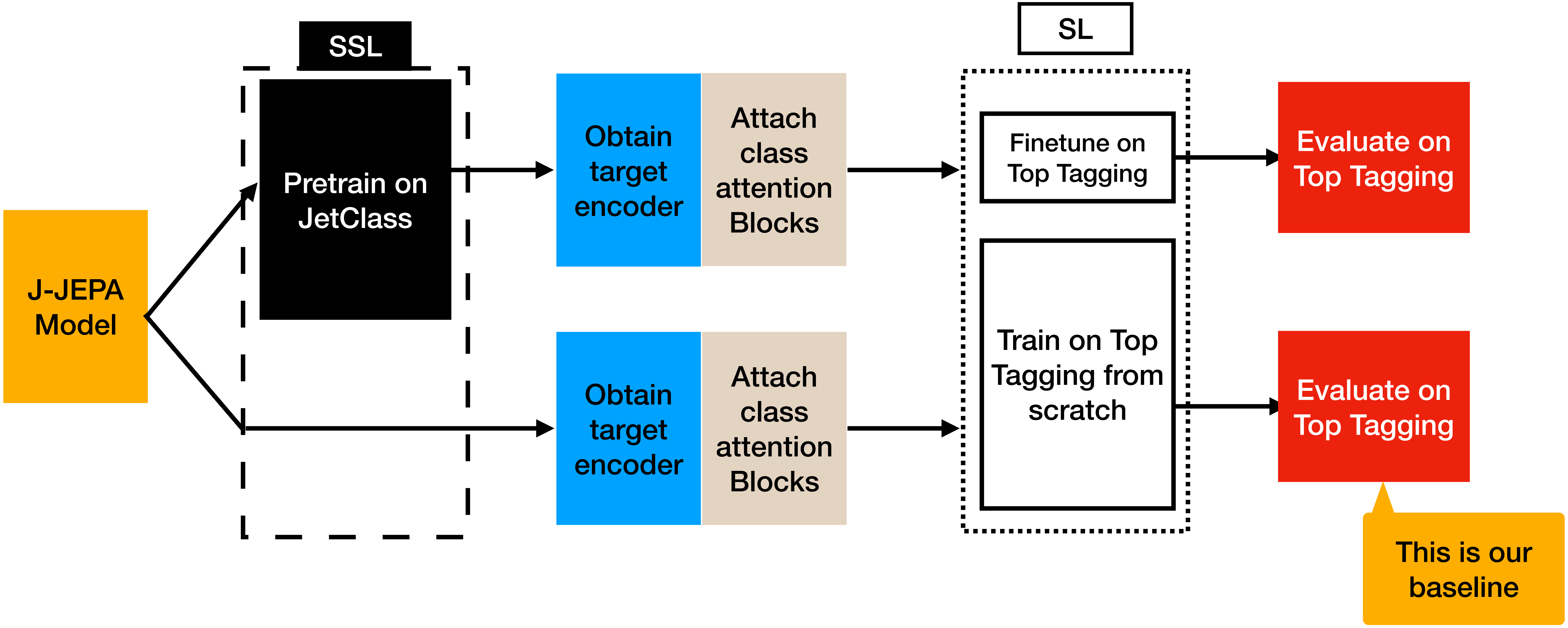
Aggregation Methods for Fine-tuning

3 Different methods of attaching the latent space to a classification head



Our training and evaluation setup

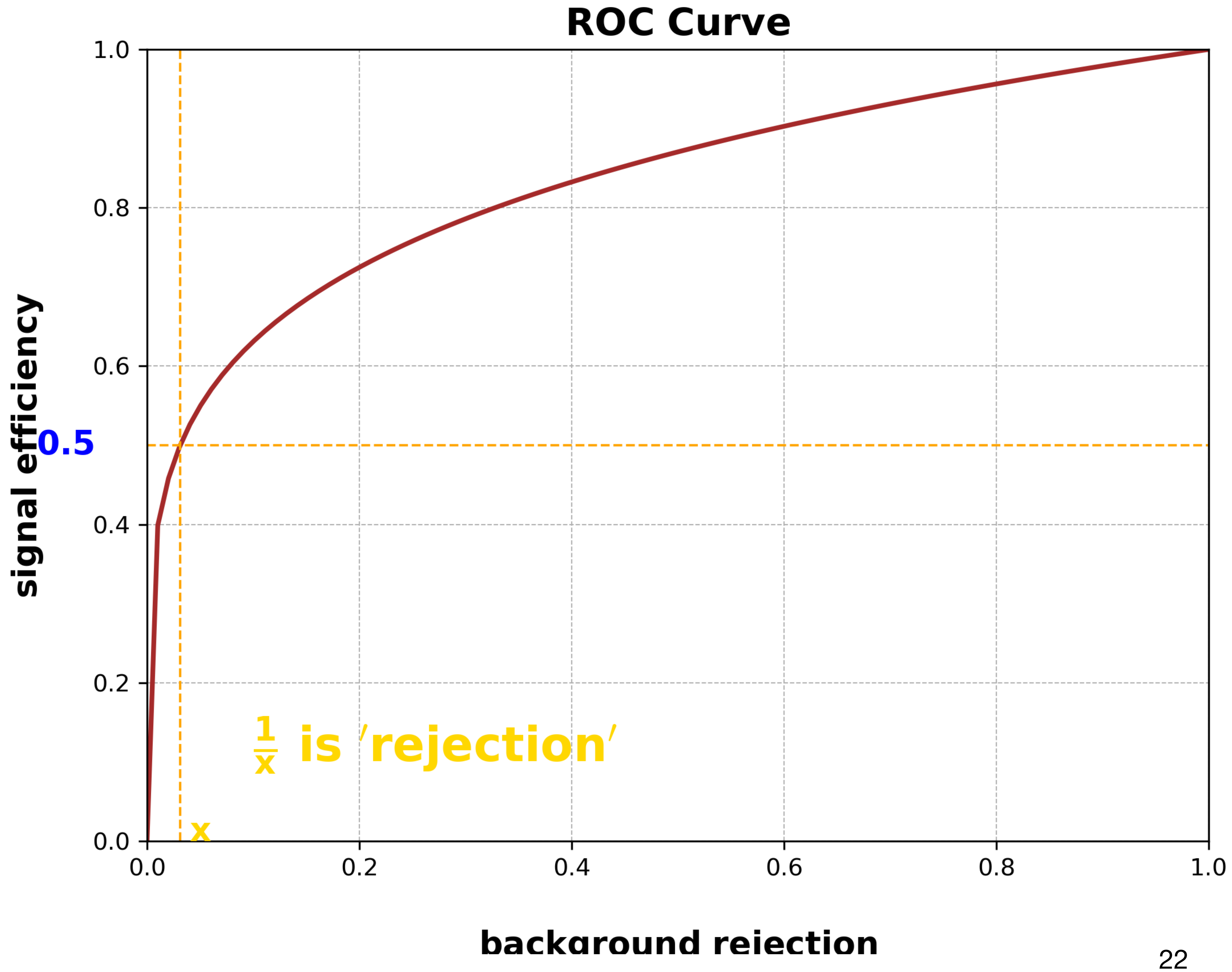
Baseline refers to the same model directly trained on the finetuning dataset without pretraining



Metrics

Accuracy: correctly predicted / total number of samples

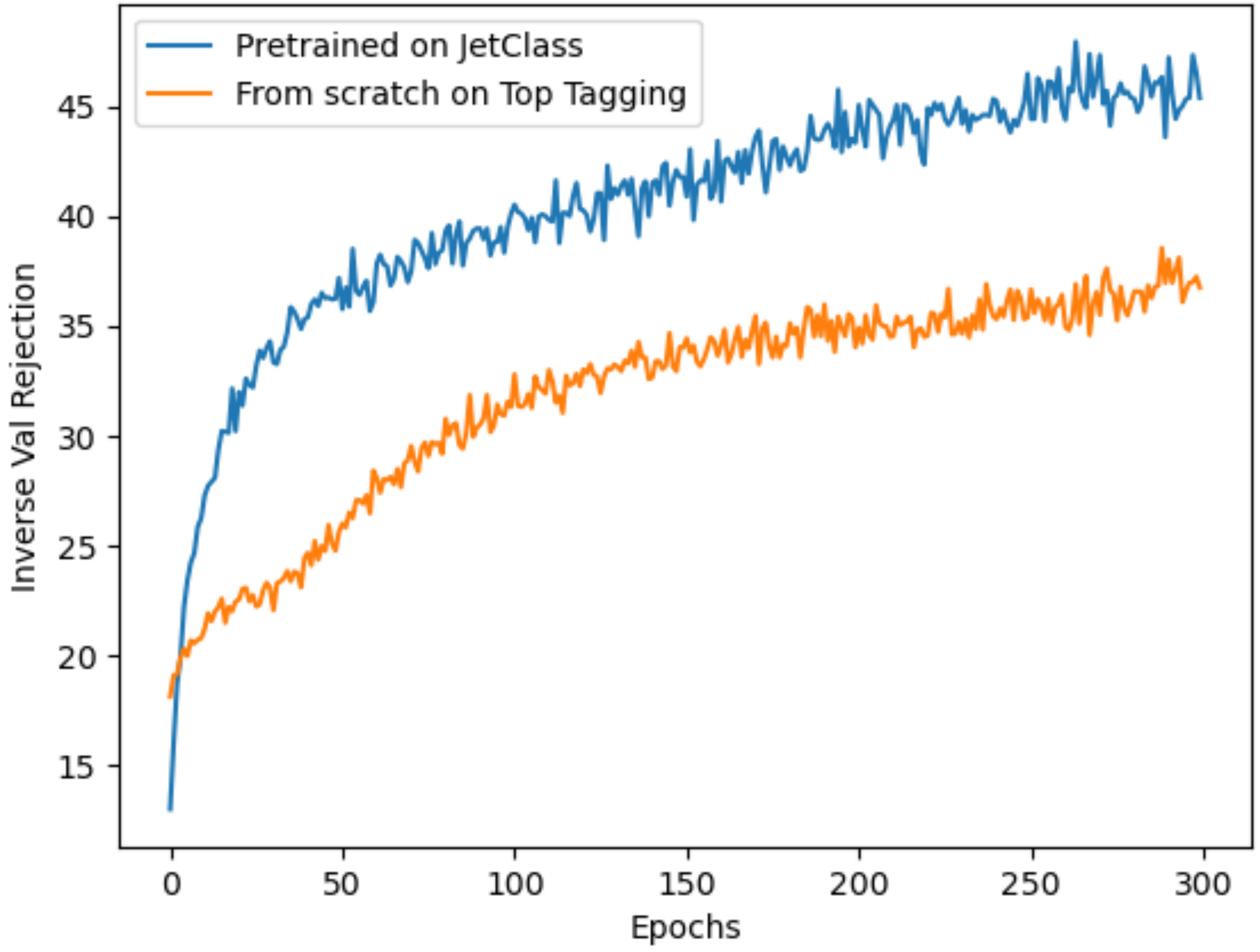
Rejection: inverse of background rejection (FPR) at 50% signal efficiency (TPR)

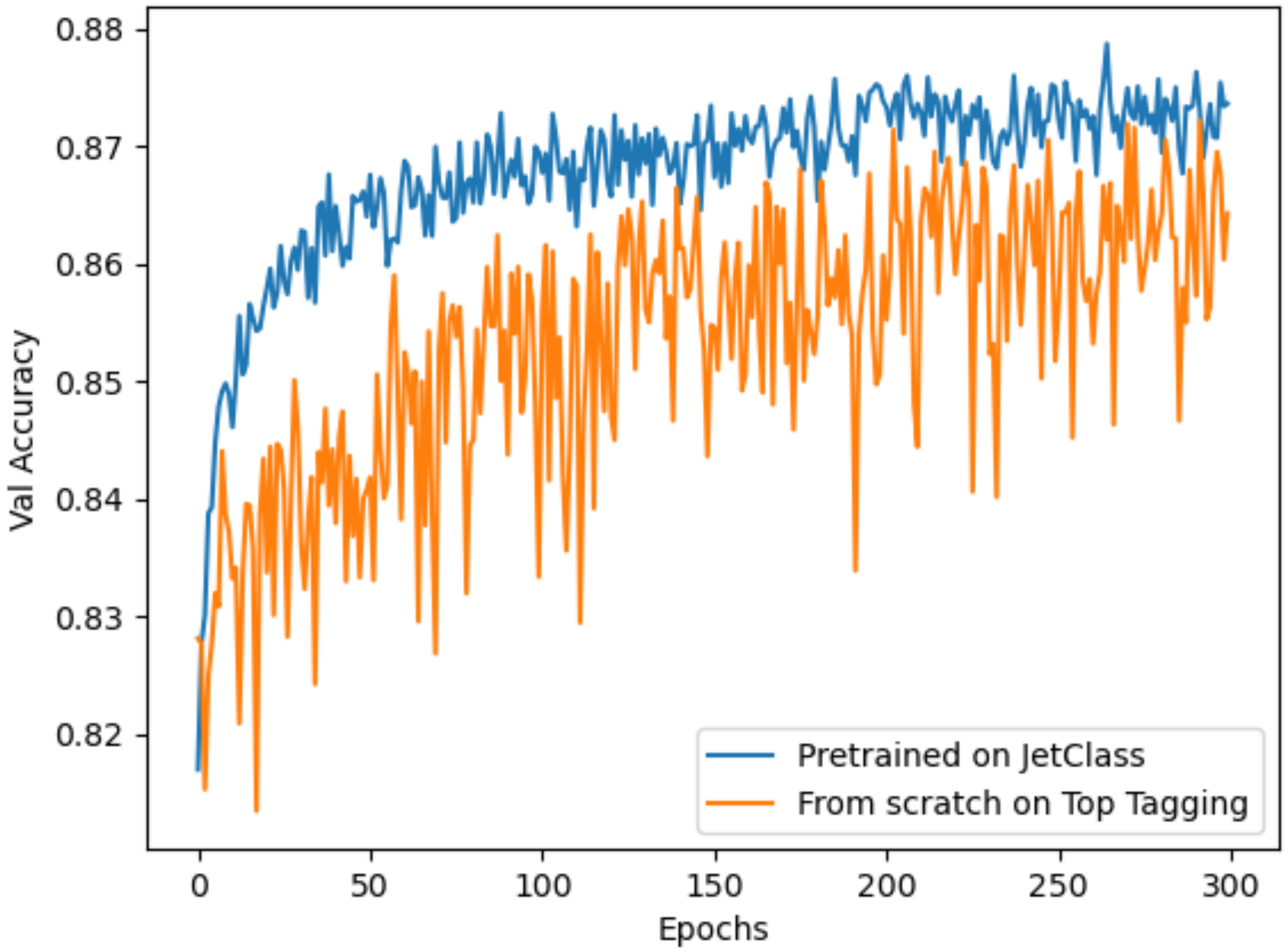


Significance: In a background dominant dataset, how much background can you reject while letting in a certain number of signal samples (the more the better)

J-JEPA Performance

Pretrain on JetClass and finetune on Top Tagging

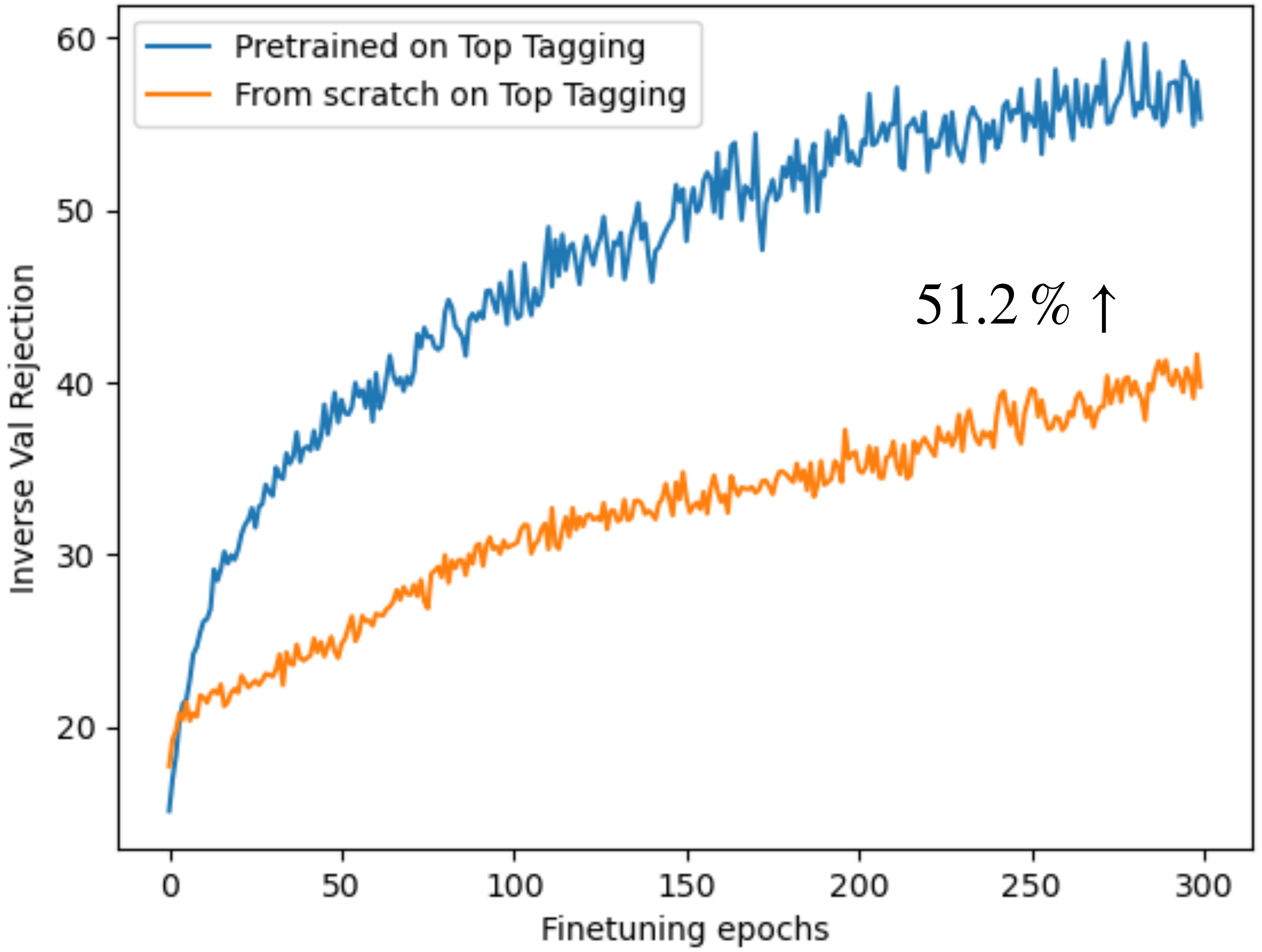


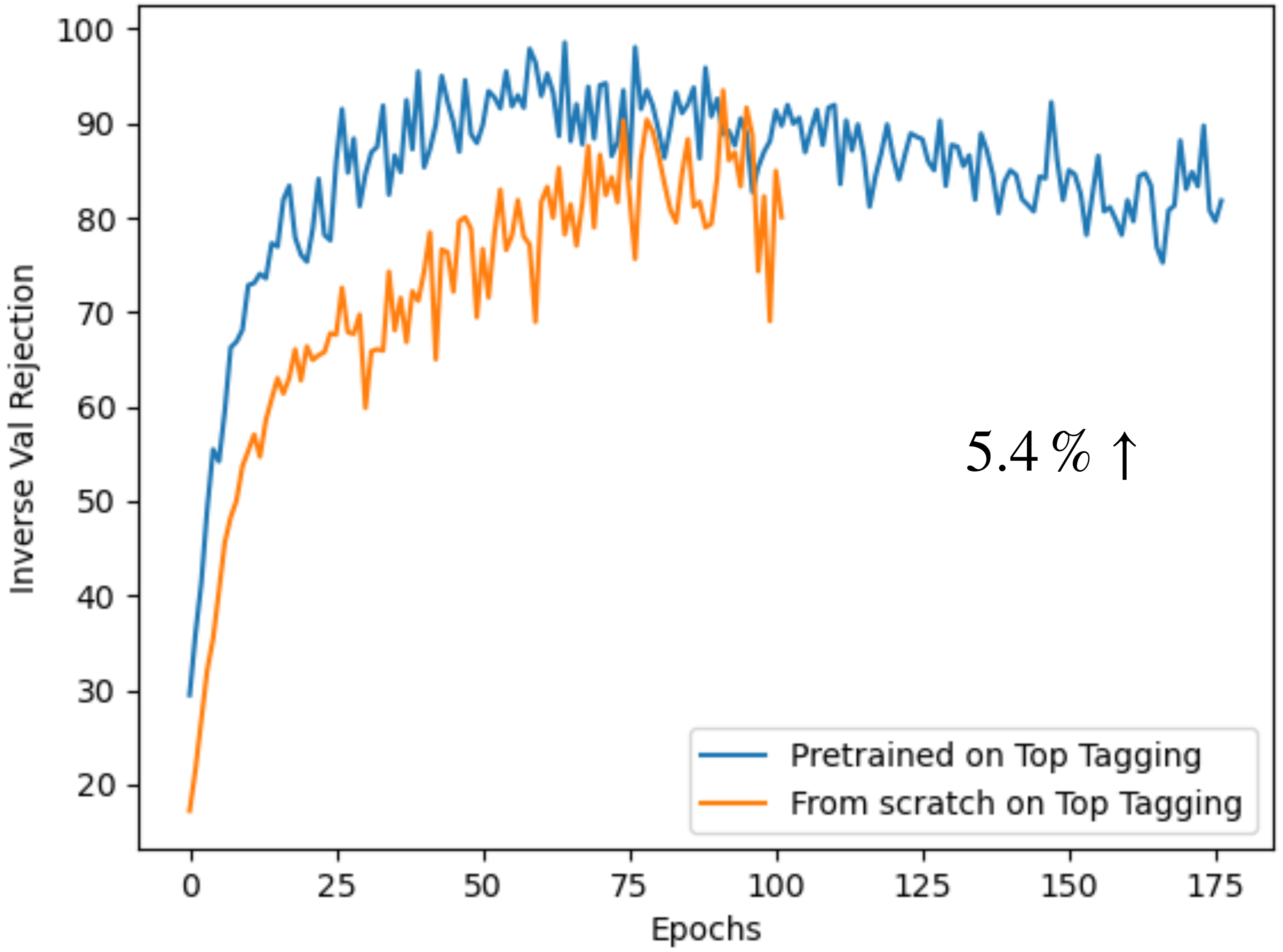


Models shown on this slide used MLP as SEL, flattened the subjet representations to represent each jet, and were fine-tuned with 10% validation set

J-JEPA Performance

Pre-train and fine-tune on TopTagging

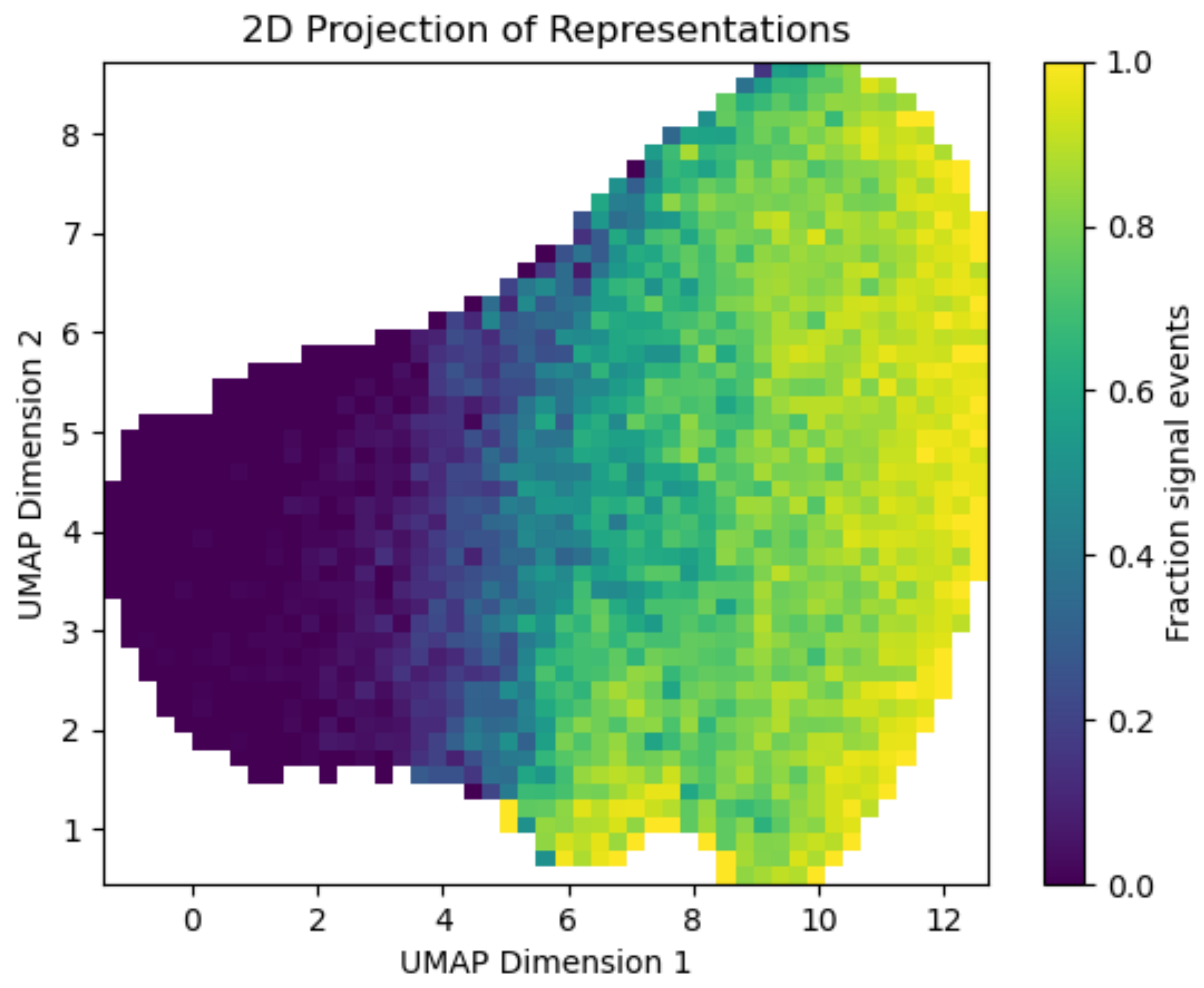


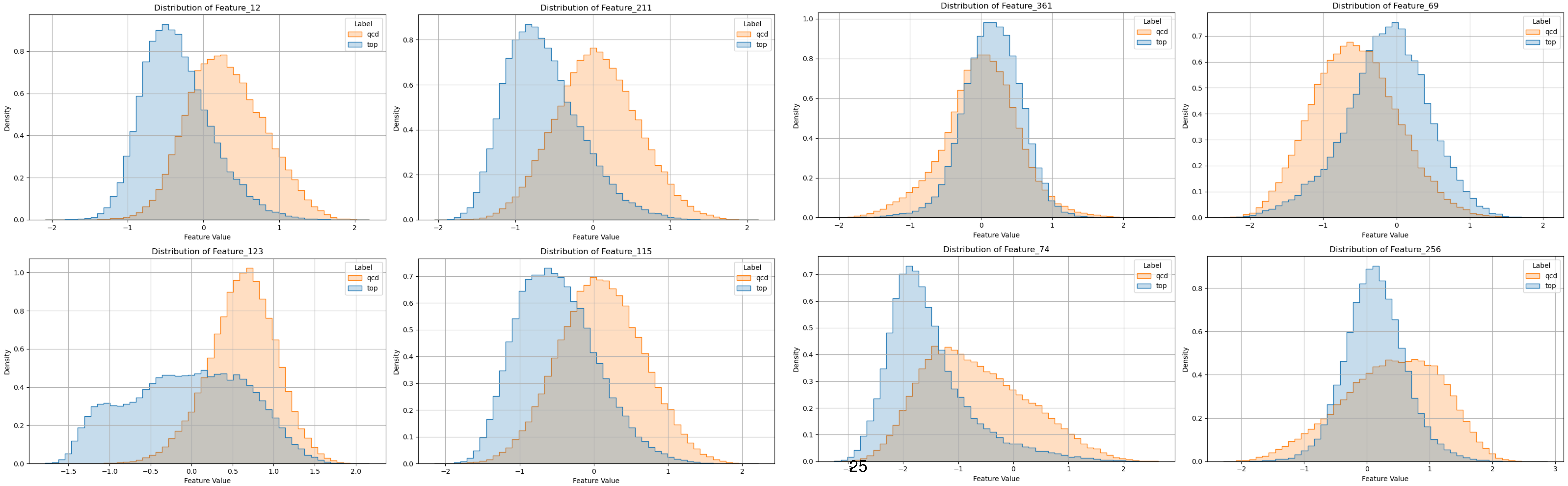


The left hand side used MLP as SEL, flattened the subjet representations to represent each jet, and were fine-tuned with 10% validation set
The right hand side used transformer SEL, used class attention blocks to aggregate jet representations, and were fine-tuned on the whole validation set

Visualizing learned features

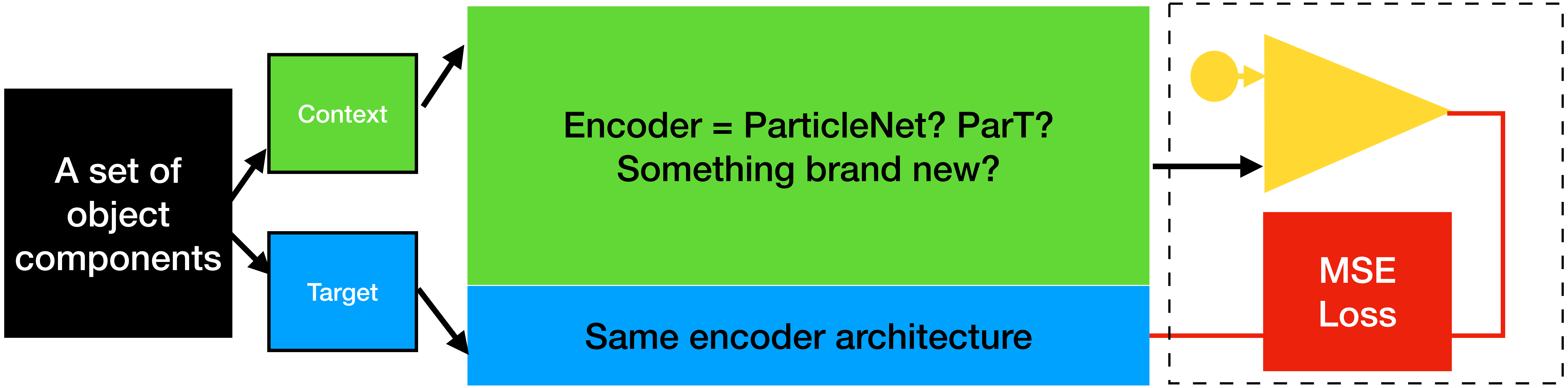
UMAP and direct comparison show that the features have good separation power





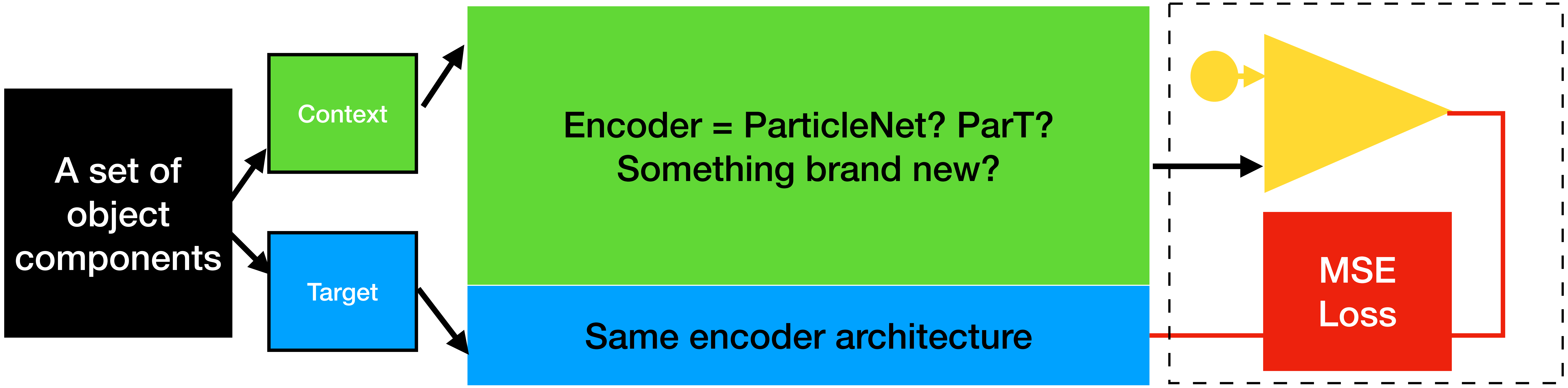
Summary

- J-JEPA: A subject-based Joint-Embedding Predictive Architecture
- Pre-train J-JEPA on a large dataset and finetune the target encoder on a small dataset achieves better performance than training the encoder from scratch,
- Pre-train J-JEPA + fine-tune on the same dataset achieves better performance faster than the baseline that learned from scratch in a supervised fashion.
- Different encoder architectures has different response to the J-JEPA pre-training, but overall positive.



Ongoing Work

- Implementing a particle-based JEPA
- Training shorter models to reduce overfitting
- Experiment different ways to provide information to the predictor
- Generalize the JEPA scheme to different physics objects: particles, events, detector readout, etc.



Support

Thank you for listening!

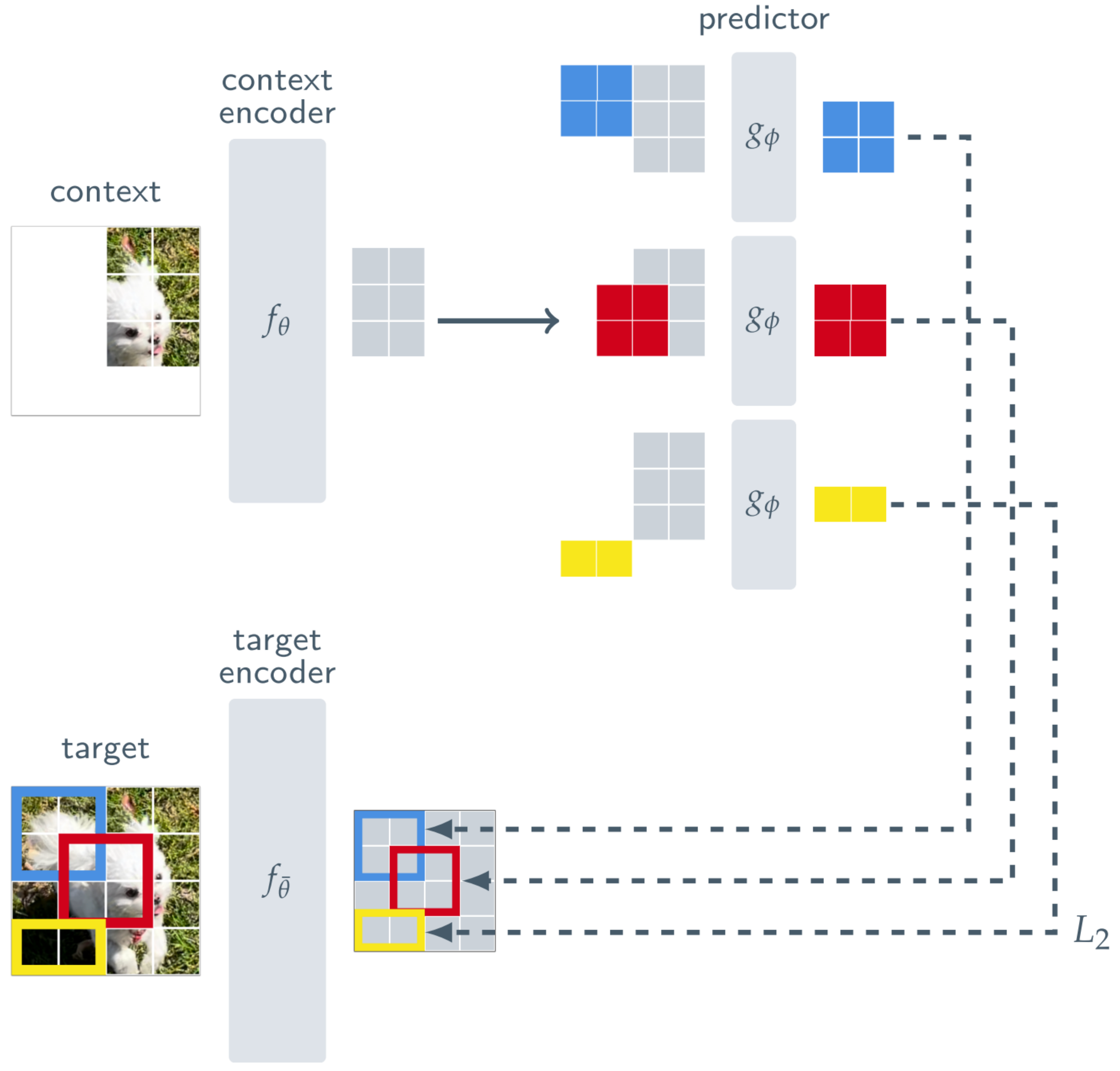
- This work is supported by the National Science Foundation under award number 2117997 (A3D3 Institute), Research Corporation For Science Advancement, the Alfred P. Sloan Foundation, and the U.S. Department of Energy

- This work was performed using the National Research Platform Nautilus HyperCluster supported by NSF

Backup

Example: The I-JEPA Architecture

I: Image



Details of the Top Tagging Dataset

The top signal and mixed quark-gluon background jets are produced with using Pythia8 [25] with its default tune for a center-of-mass energy of 14 TeV and ignoring multiple interactions and pile-up. For a simplified detector simulation we use Delphes [26] with the default ATLAS detector card. This accounts for the curved trajectory of the charged particles, assuming a magnetic field of 2 T and a radius of 1.15 m as well as how the tracking efficiency and momentum smearing changes with η. The fat jet is then defined through the anti-k_T algorithm [27] in FastJet [28] with $R = 0.8$. We only consider the leading jet in each event and require

$$p_{T,j} = 550 \; .... \; 650 \text{ GeV} \; . \tag{1}$$

For the signal only, we further require a matched parton-level top to be within $\Delta R = 0.8$, and all top decay partons to be within $\Delta R = 0.8$ of the jet axis as well. No matching is performed for the QCD jets. We also require the jet to have $|\eta_j| < 2$. The constituents are extracted through the Delphes energy-flow algorithm, and the 4-momenta of the leading 200 constituents are stored. For jets with less than 200 constituents we simply add zero-vectors.

1902.09914

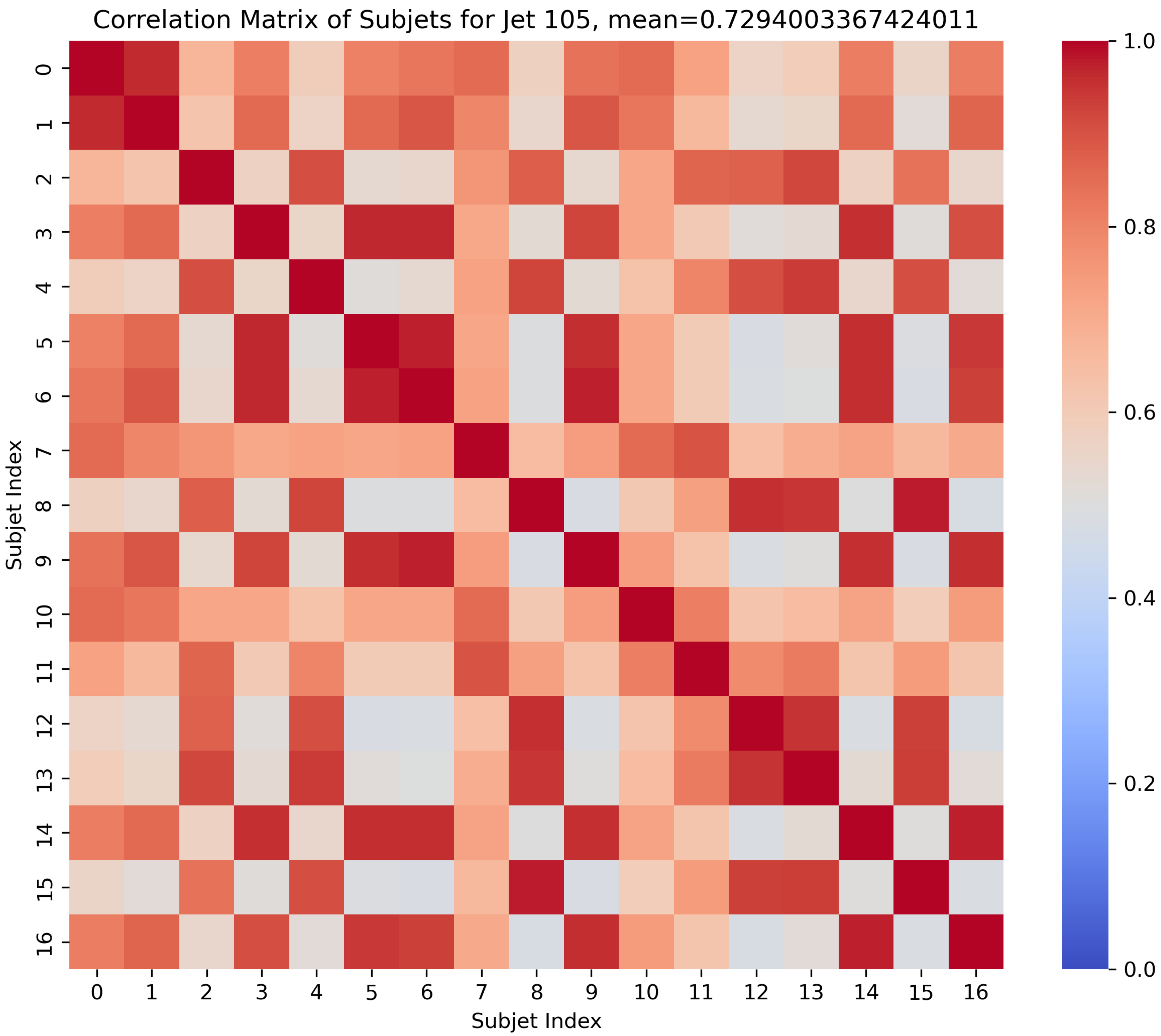
Details of the JetClass Dataset

Simulation setup. Jets in this dataset are simulated with standard Monte Carlo event generators used by LHC experiments. The production and decay of the top quarks and the W, Z and Higgs bosons are generated with MADGRAPH5_aMC@NLO (Alwall et al., 2014). We use PYTHIA (Sjöstrand et al., 2015) to evolve the produced particles, i.e., performing parton showering and hadronization, and produce the final outgoing particles[1]. To be close to realistic jets reconstructed at the ATLAS or CMS experiment, detector effects are simulated with DELPHES (de Favereau et al., 2014) using the CMS detector configuration provided in DELPHES. In addition, the impact parameters of electrically charged particles are smeared to match the resolution of the CMS tracking detector (CMS Collaboration, 2014). Jets are clustered from DELPHES E-Flow objects with the anti-k_{T} algorithm (Cacciari et al., 2008; 2012) using a distance parameter $R = 0.8$. Only jets with transverse momentum in 500–1000 GeV and pseudorapidity $|\eta| < 2$ are considered. For signal jets, only the "high-quality" ones that fully contain the decay products of initial particles are included[2].

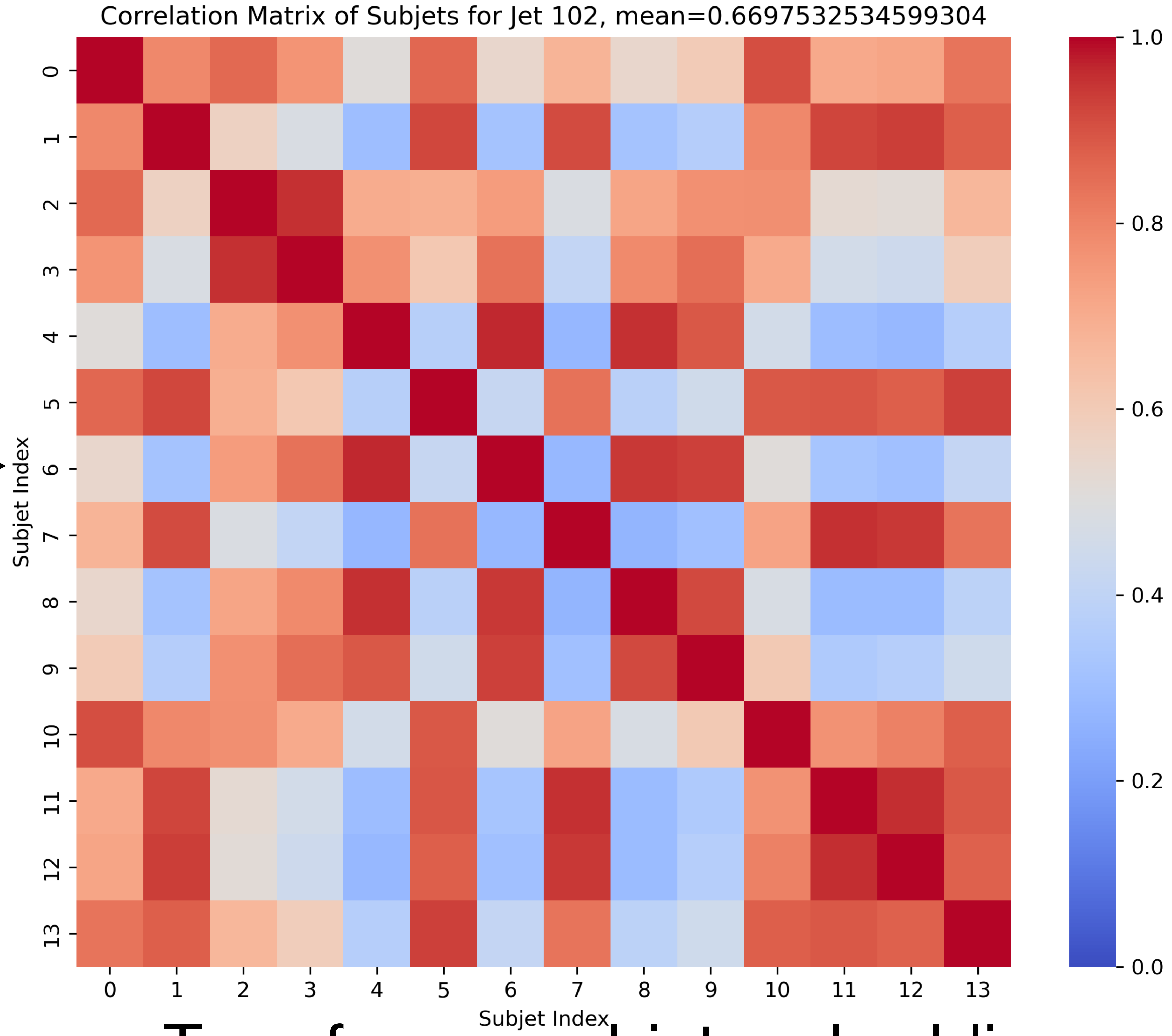
2202.03772

Transformer Embedding Layer Effects

Correlation between subjets is reduced



MLP subjet embedding



Transformer subjet embedding

WIP: Study of how to provide the additional info

Pre-train and fine-tune on Top Tagging

Experiments	Encode subjet coordinates at both (encoder and predictor)	Encode coordinates only at predictor	Encode pT ranking at both	Use a MLP to encode subjet coordinates
Inverse Rejection Power	63.99	45.33	45.02	Converging…

Study of subjet embedding

Pre-training and fine-tuning on Toptagging dataset

Inverse Rejection Power	Dimension Reduction	Dimension Expansion
Attention	**86.42**	73.81
MLP	73.55	63.99
Linear	44.31	

Strategies to prevent collapse

- Targets being padded subjets
- Most particles are padded so all subjets look the same to the model
- Information bottleneck in the predictor is too big
- Dataset was not normalized

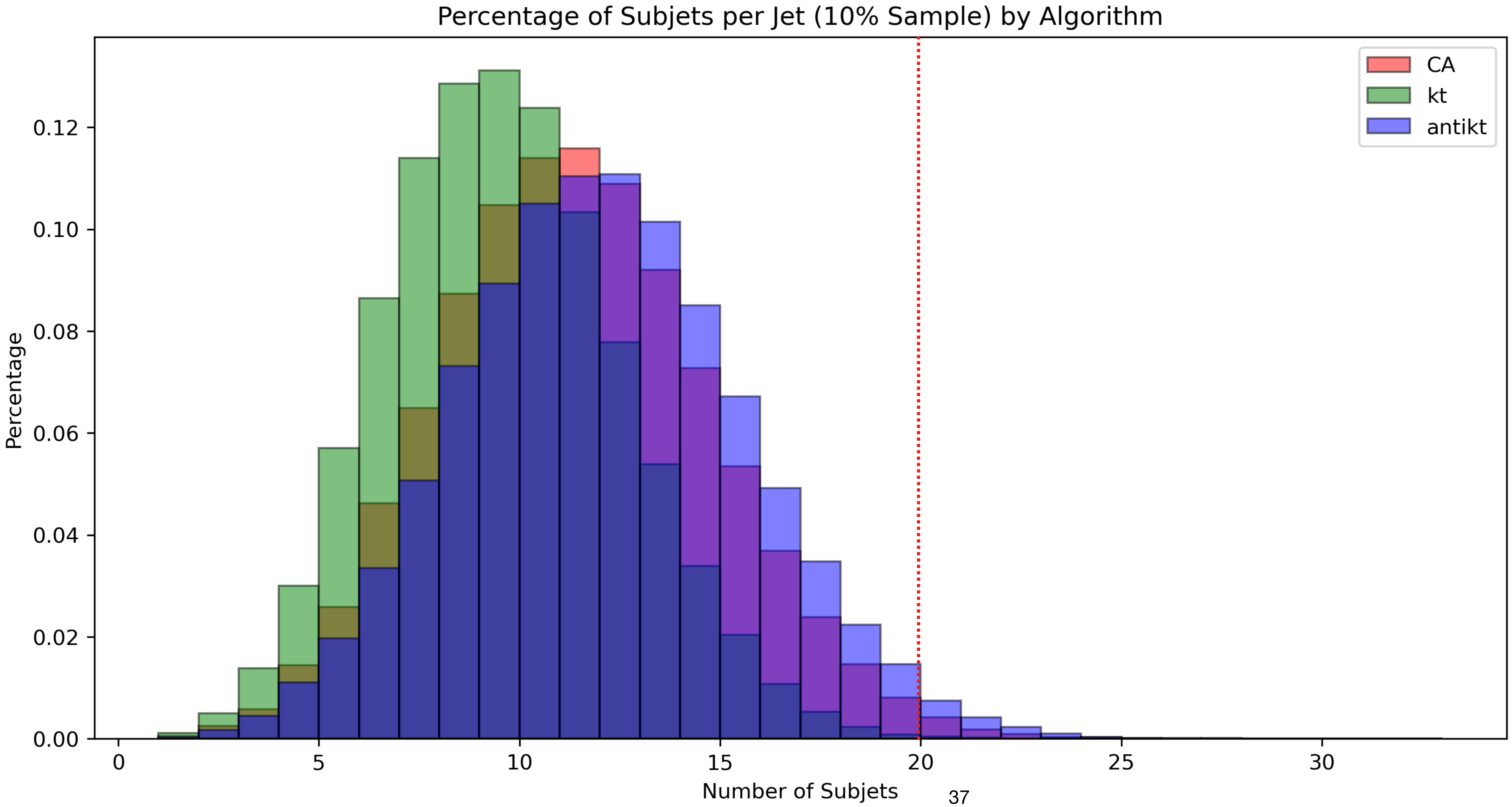
→

- We only select targets from non-empty subjets
- We implemented Attention-based embedding
- We decreased the size of the predictor dimension
- We normalized the dataset

Plus: EMA updating the Target Encoder

J-JEPA: Splitting jets into subjets

number of subjets per jet



J-JEPA: Splitting jets into subjets

number of particles per subjet