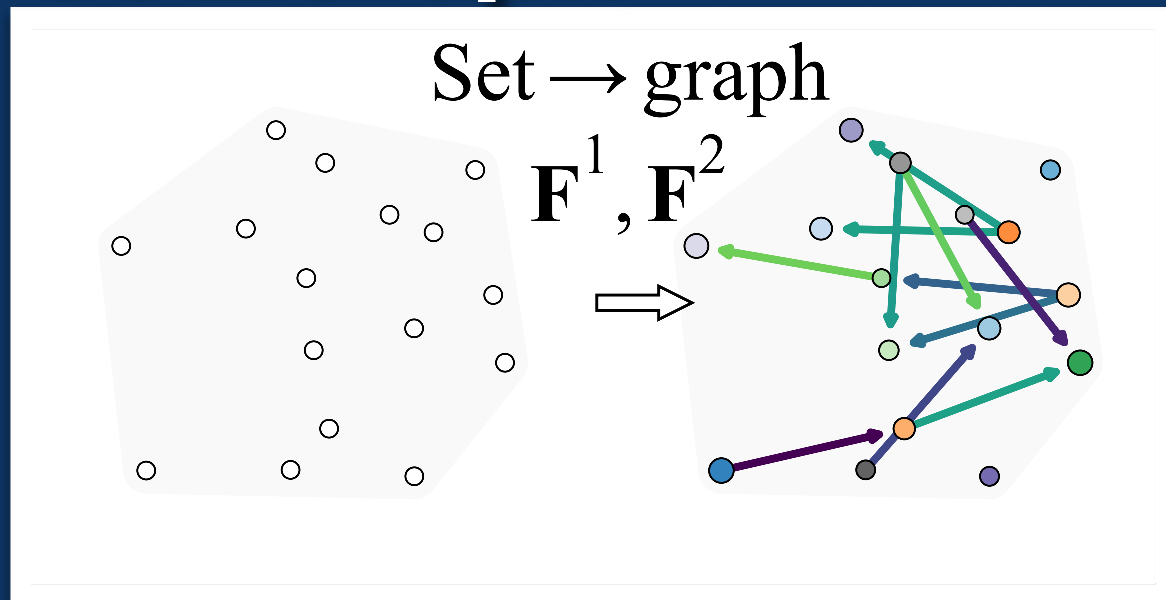
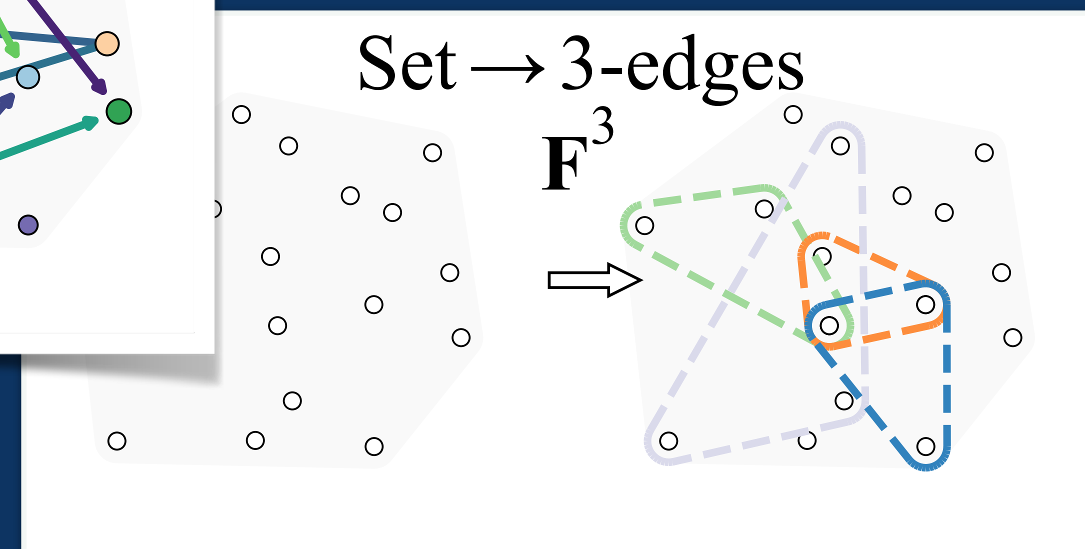
# Set2Graph




# Secondary vertex finding in jets with neural networks

4th IML Machine Learning Workshop
Wednesday, 21 October

Jonathan Shlomi, Sanmay Ganguly, Eilam Gross, Kyle Cranmer, Yaron Lipman,
Hadar Serviansky, Haggai Maron, Nimrod Segol

NYU
מכון ויצמן למדע
WEIZMANN INSTITUTE OF SCIENCE
NVIDIA
ATLAS EXPERIMENT

# Outline

Introduction

The Task - secondary vertex finding

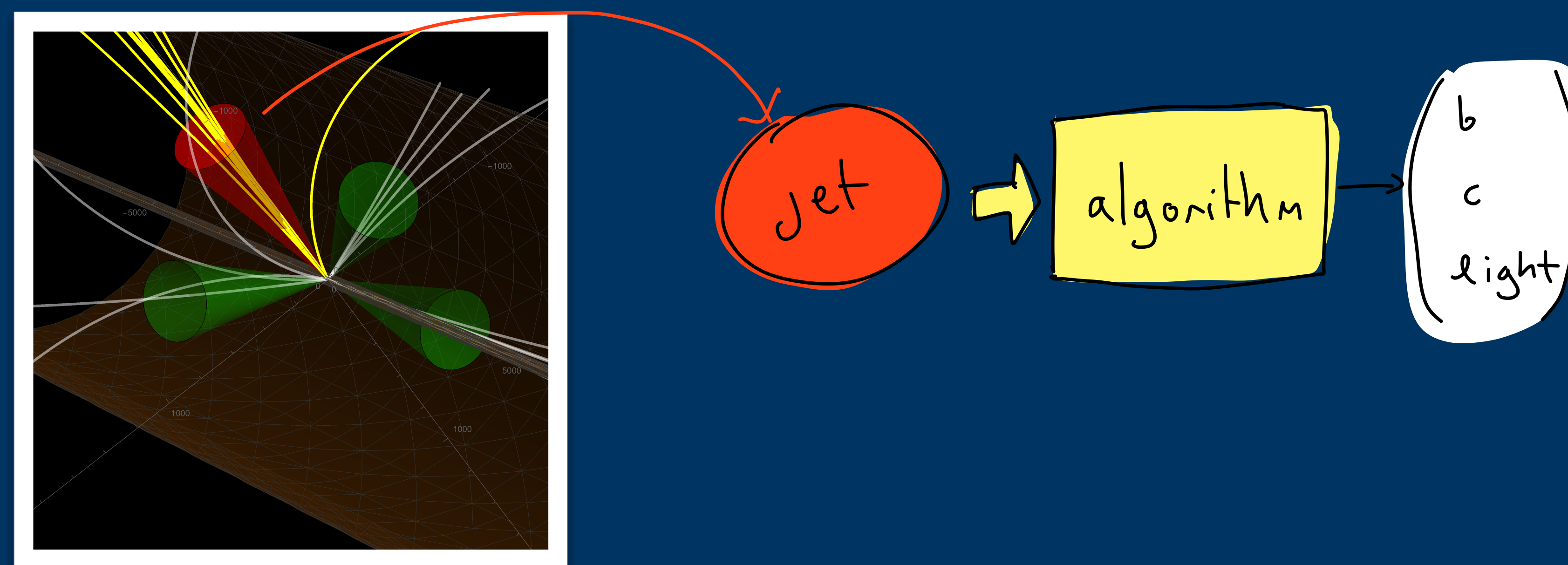
The model

Experiment

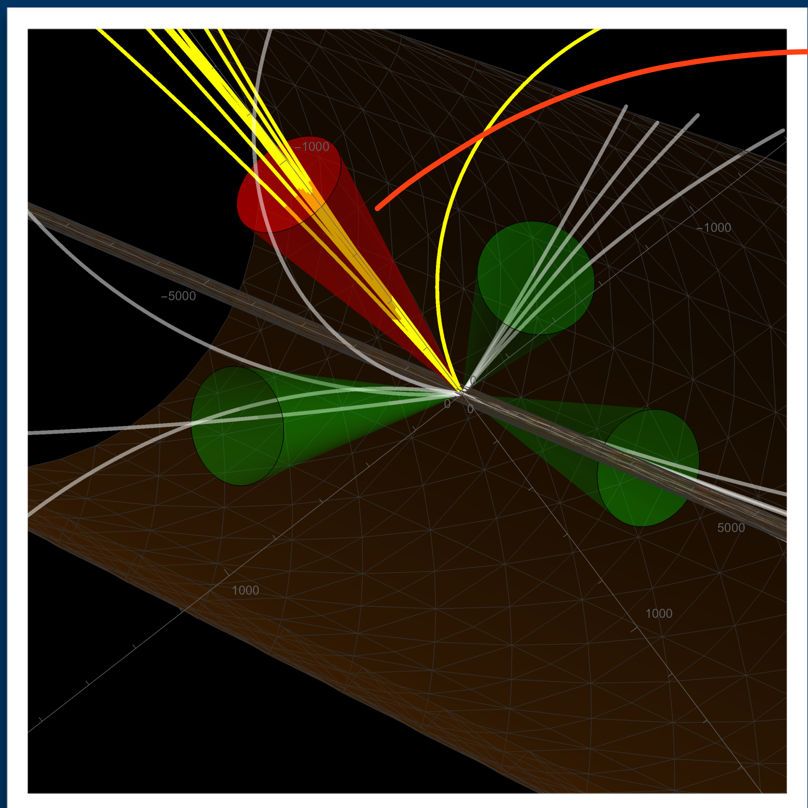
- The Dataset
- Baseline algorithms for comparison
- Performance metrics - 3 perspectives
  - + Results

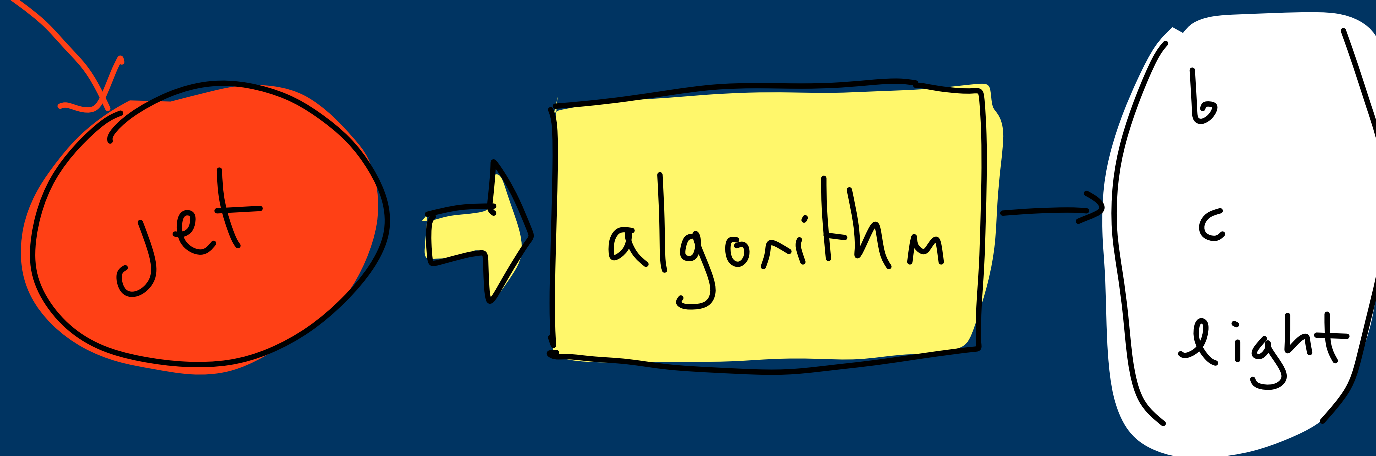
Summary + Future Directions

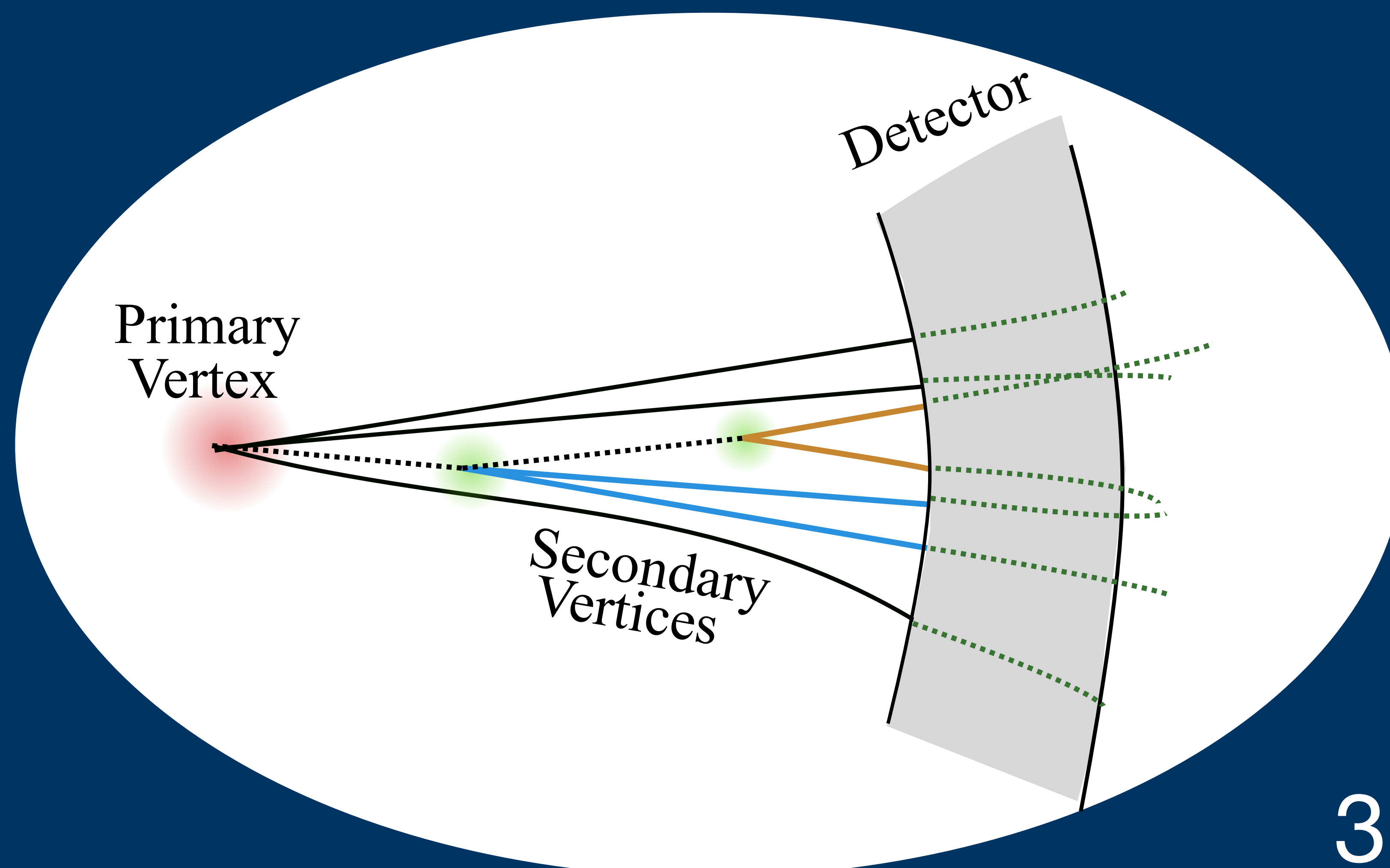
## Jet flavour tagging: identifying the quark flavour at the origin of the jet



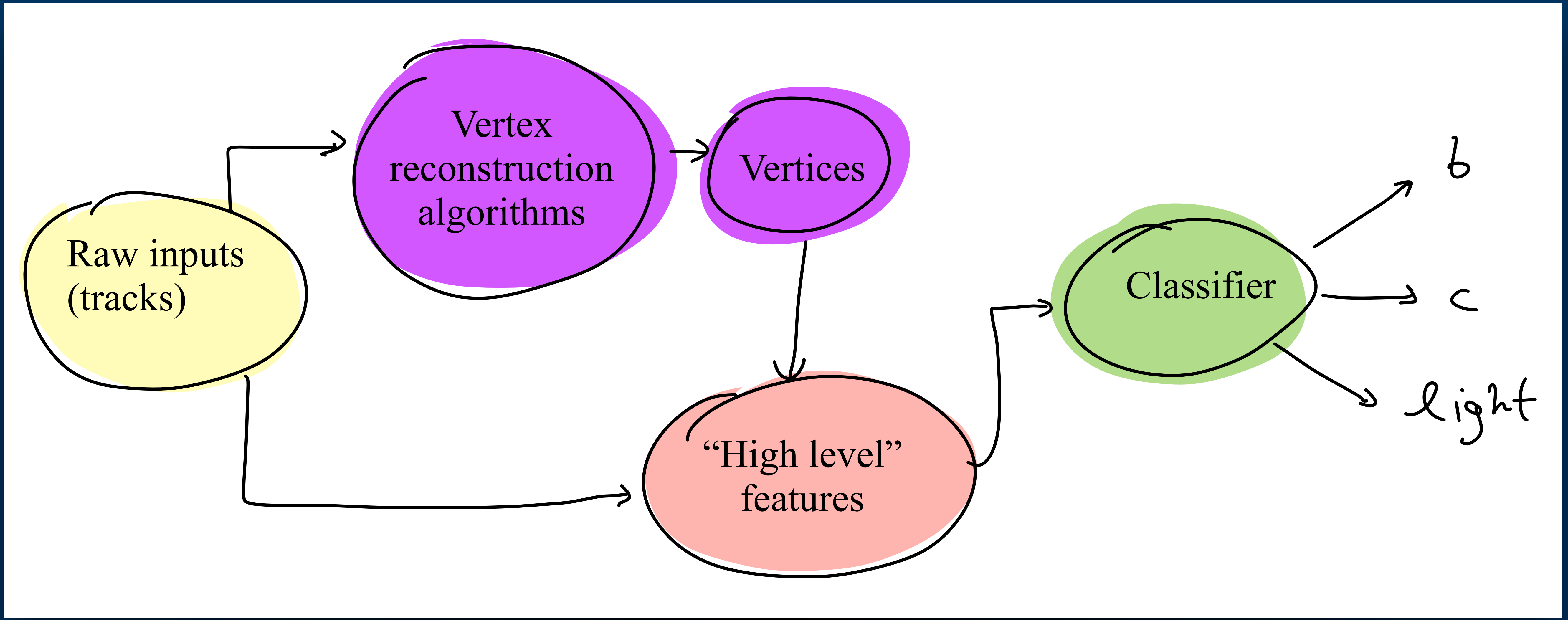
# Jet flavour tagging:
# identifying the quark flavour at the origin of the jet



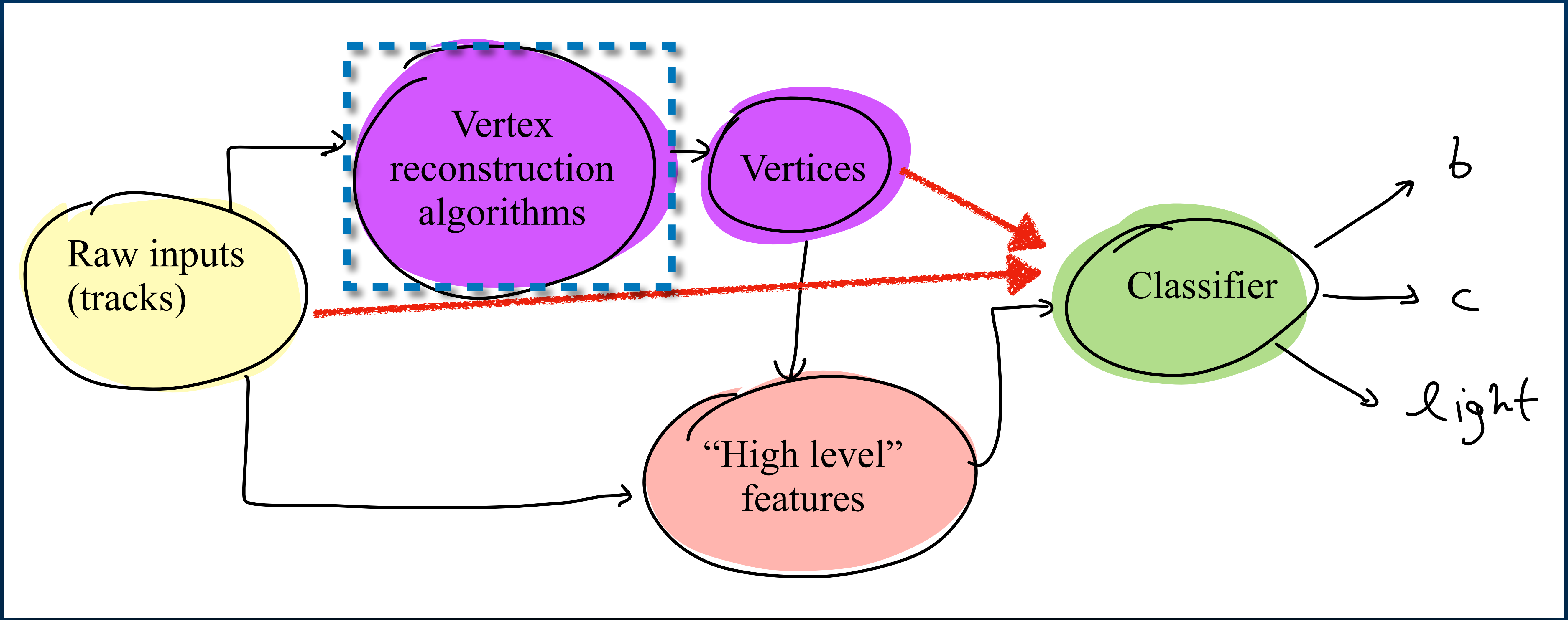


- Classifiers were built on human-designed discriminating “high level” features.

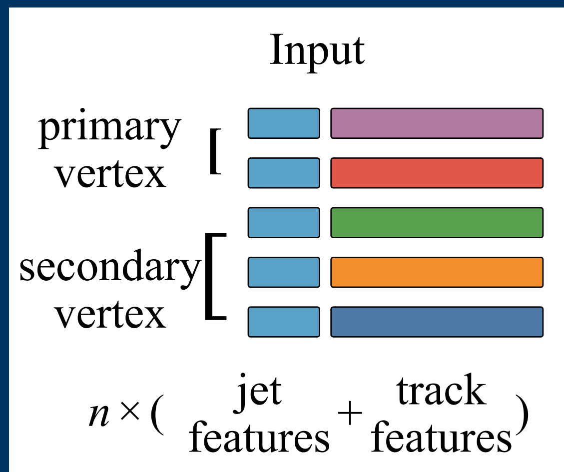


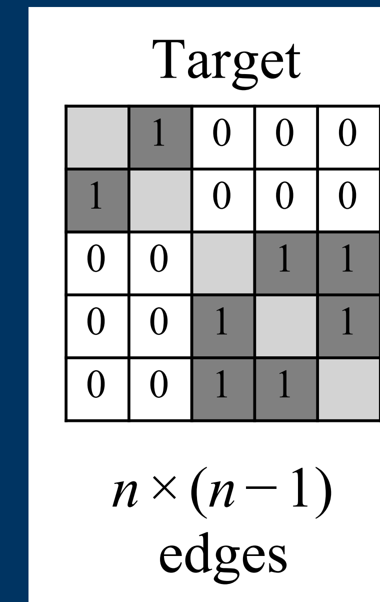
- In recent years, classifiers are using the raw reconstructed tracks/ vertices in the jet - in addition to the high level features.
- This talk is about using machine learning for performing the intermediate step of vertex reconstruction



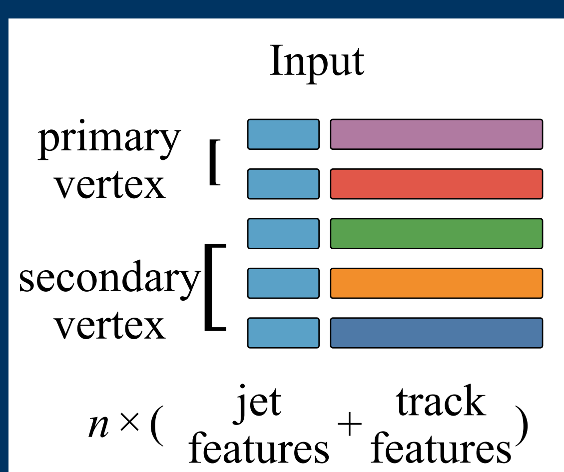
## The Task - secondary vertex finding

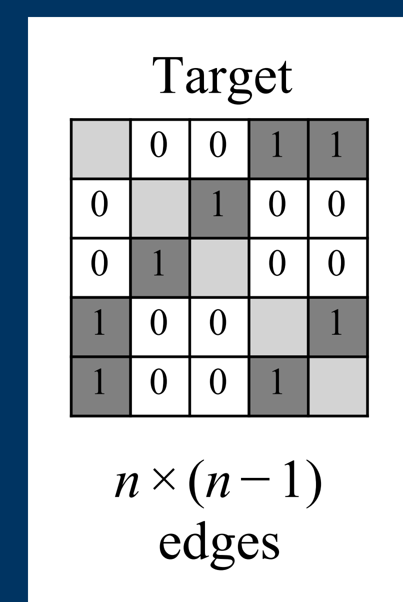
- We want to learn a function from $\mathbb{R}^{n \times d_{in}} \rightarrow \mathbb{R}^{n \times n \times 1}$




- The function from $\mathbb{R}^{n \times d_{in}} \rightarrow \mathbb{R}^{n \times n \times 1}$ is equivariant - If we permeate the inputs the output undergoes a similar permutation



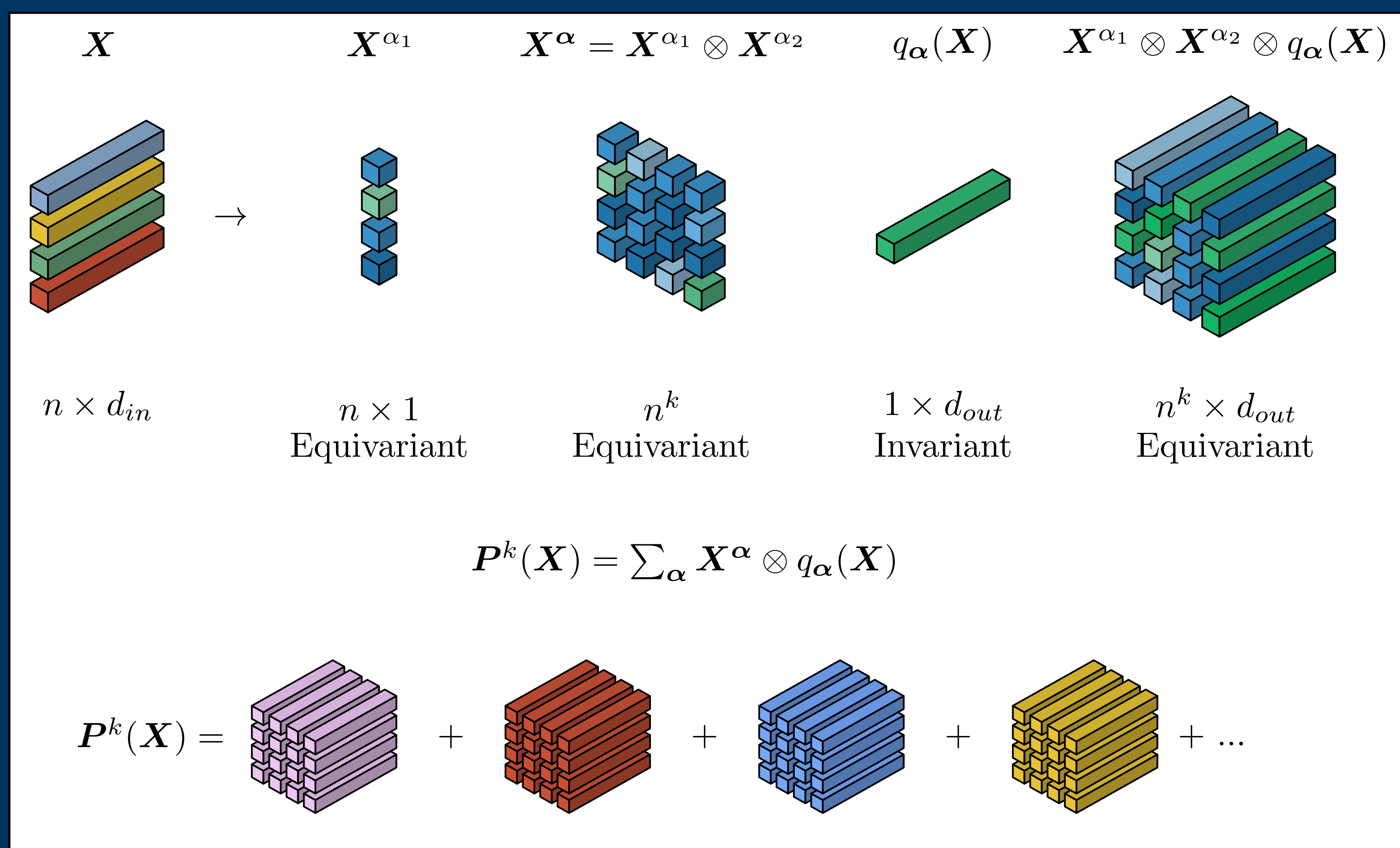



## The model - a universal model for any task that takes as input a set, and learns a graph structure (edges, hyper edges)

# The idea of the proof of universality of the model:

- Any continuous equivariant function G from set to k-edges can be approximated by an equivariant polynomial $P^k(X)$

- This polynomial has a very specific structure because it is equivariant
- We can build our neural network model to match this structure of $P^k(X)$

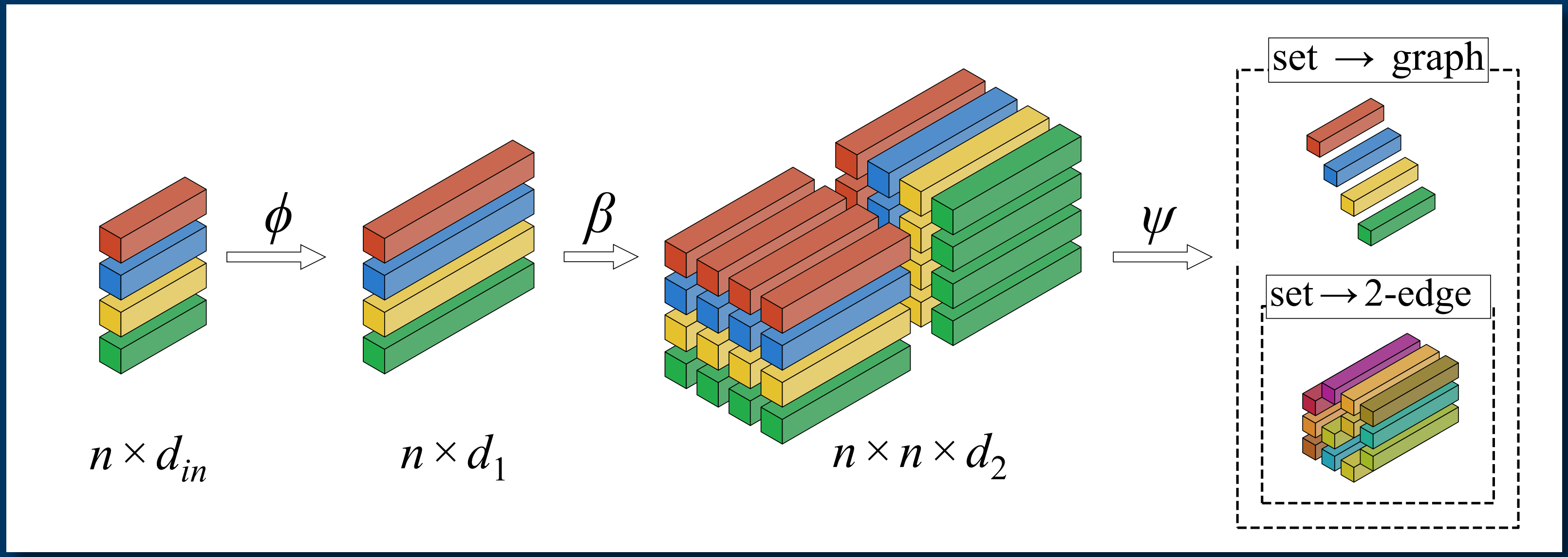


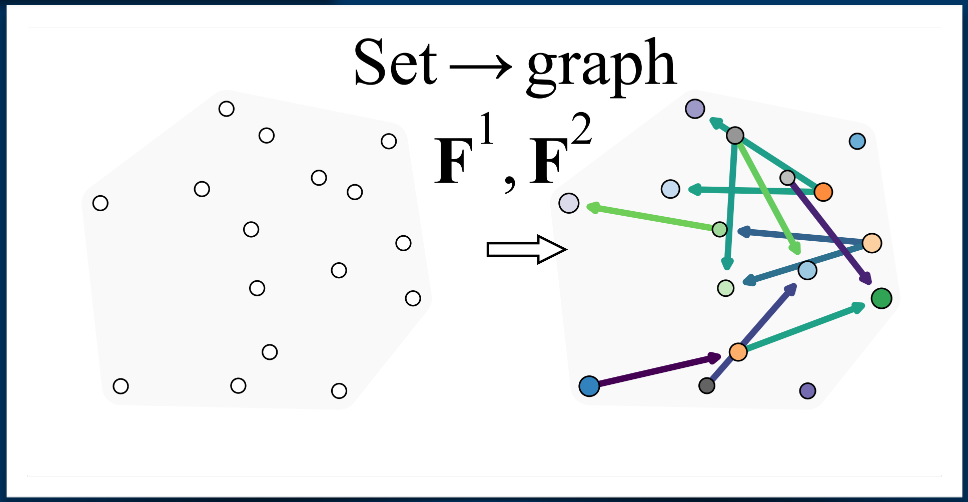
The model has the form $\psi(\beta(\phi(X)))$

$\phi$ is an equivariant set to set function

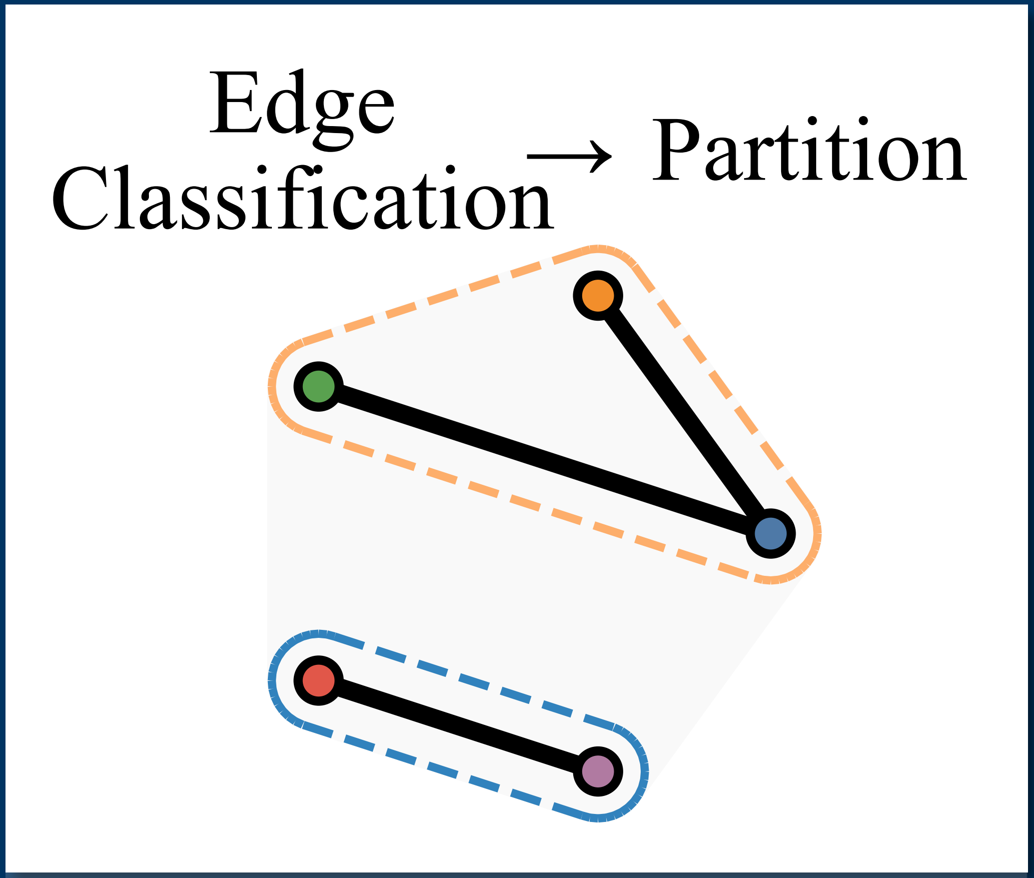
$\beta$ is a broadcasting layer, it forms all the possible k-tuples of nodes

$\psi$ is an MLP that operates on each edge/hyperedge to produce the final output



In the end we “manually” convert the edge classification to a valid partition of the set



# Experiment

The Dataset

Baseline algorithms for comparison

Performance metrics - 3 perspectives

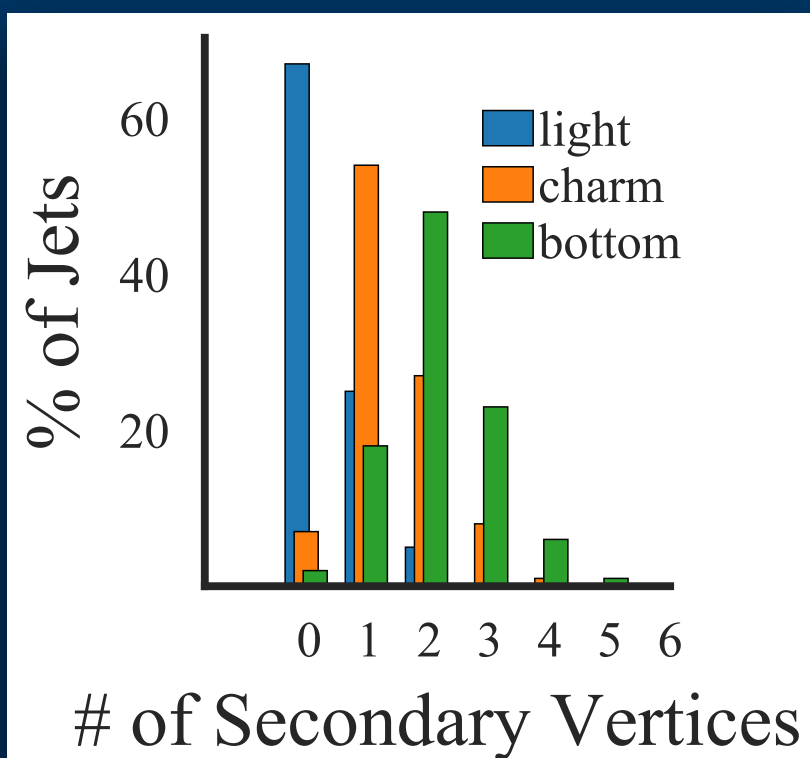
+ Results

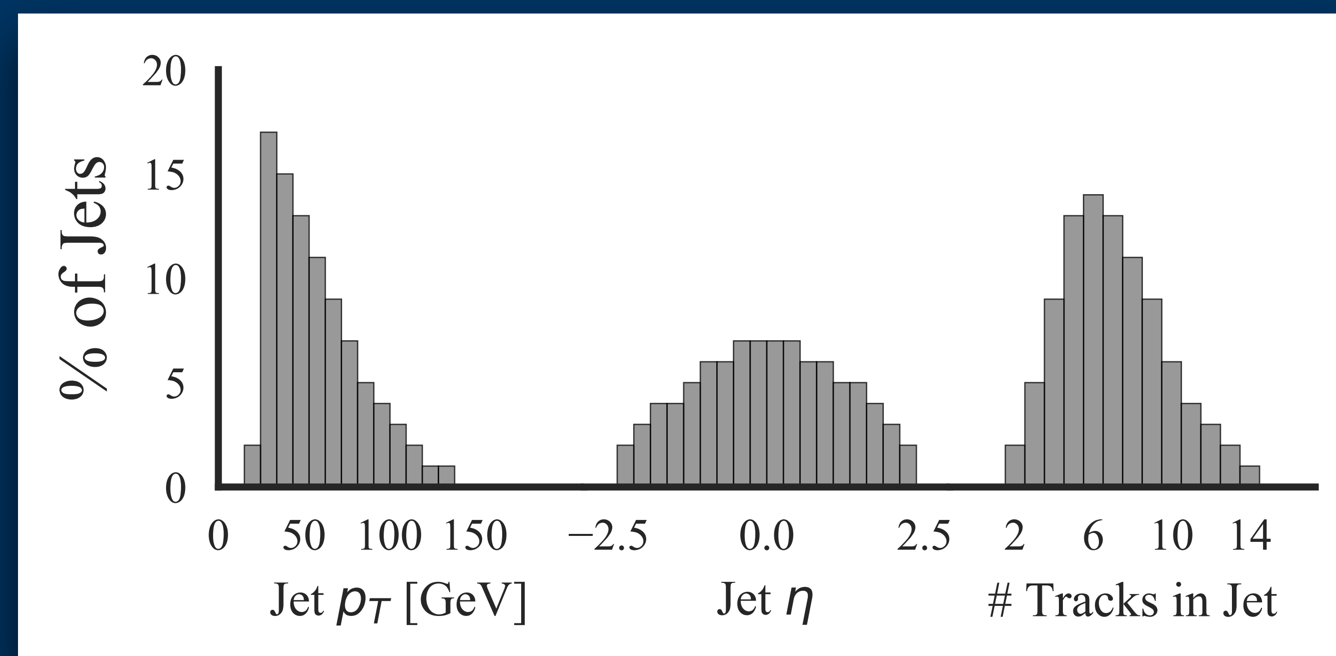
# The Dataset

https://zenodo.org/record/4044628

DOI 10.5281/zenodo.4044628

A $t\bar{t}$ sample, but with the distributions of b/c/light jets adjusted to have the same number of jets for each $p_T, \eta, n_{tracks}$ bin




PYTHIA

DELPHES
fast simulation

http://home.thep.lu.se/Pythia/ https://github.com/delphes/delphes

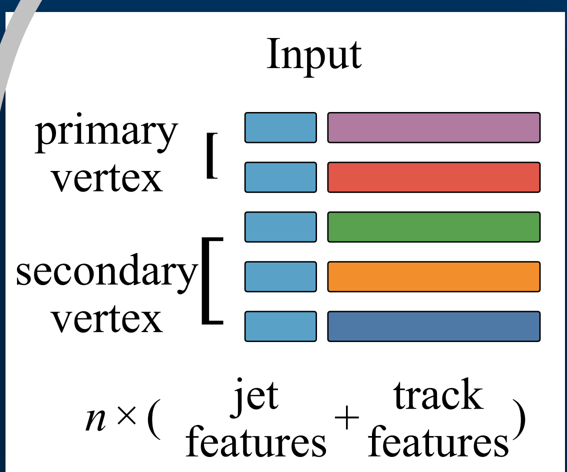
# The Dataset

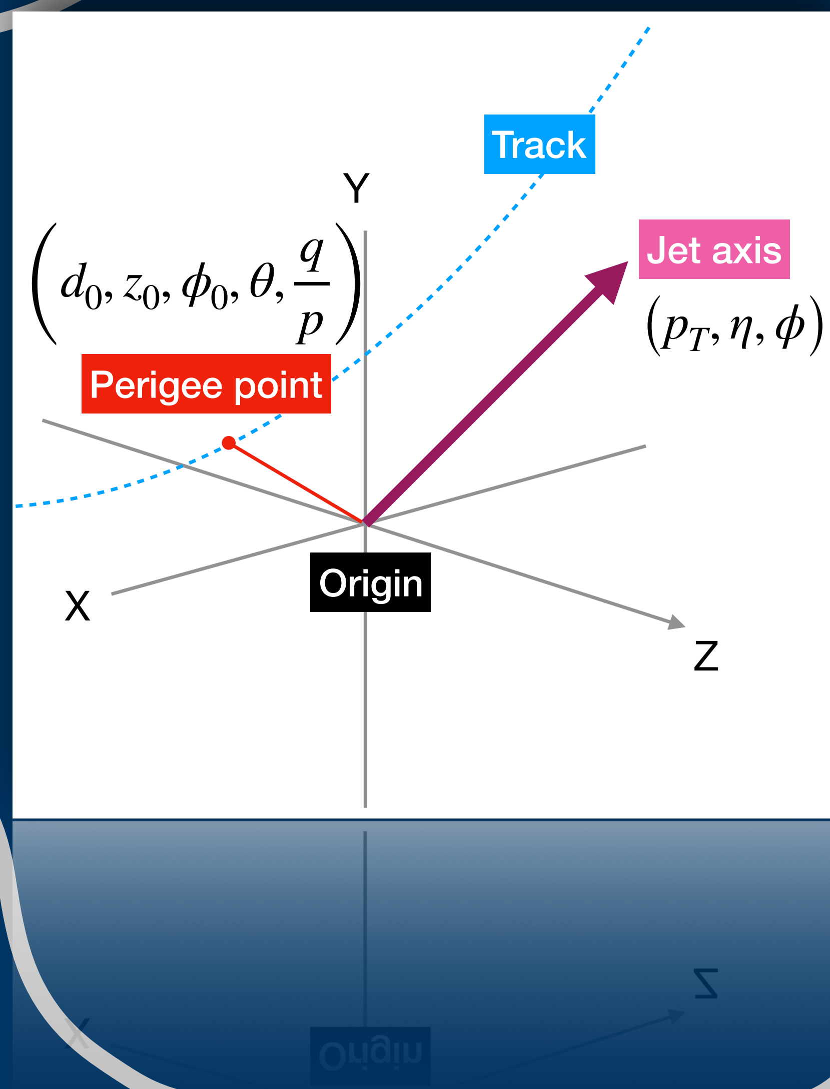
https://zenodo.org/record/4044628

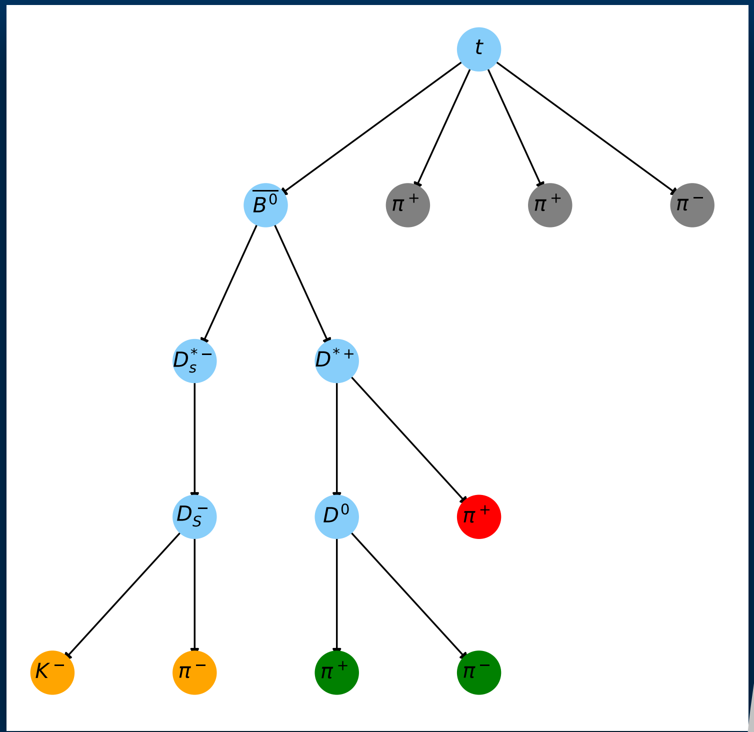
DOI 10.5281/zenodo.4044628

Track perigee parameters

Jet 4-vector




Target

| | | | | |
|---|---|---|---|---|
| | 1 | 0 | 0 | 0 |
| 1 | | 0 | 0 | 0 |
| 0 | 0 | | 1 | 1 |
| 0 | 0 | 1 | | 1 |
| 0 | 0 | 1 | 1 | |

$n \times (n-1)$
edges

PYTHIA

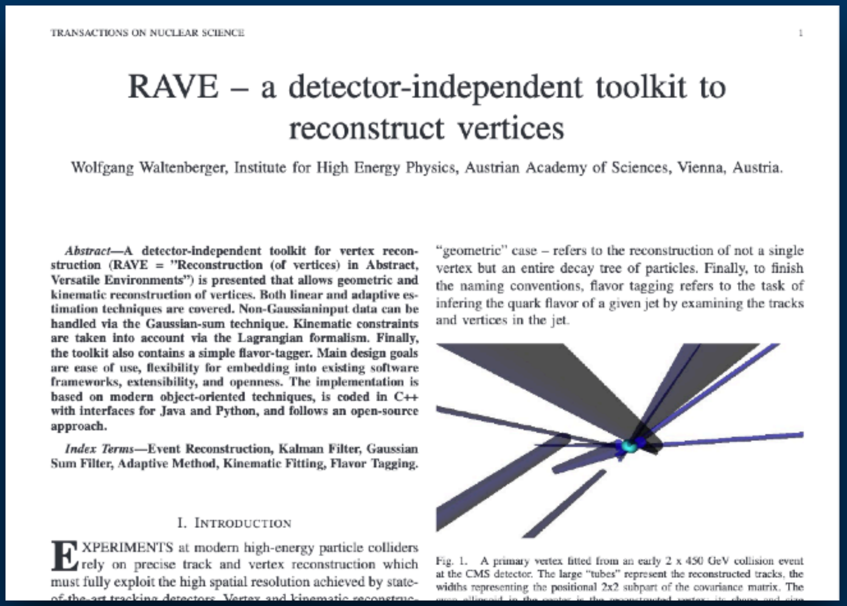
DELPHES fast simulation

http://home.thep.lu.se/Pythia/ https://github.com/delphes/delphes

# Two baseline algorithms:

https://ieeexplore.ieee.org/document/5734880

Adaptive Vertex Reconstruction (AVR)


TRANSACTIONS ON NUCLEAR SCIENCE 1

RAVE – a detector-independent toolkit to reconstruct vertices

Wolfgang Waltenberger, Institute for High Energy Physics, Austrian Academy of Sciences, Vienna, Austria.

*Abstract*—A detector-independent toolkit for vertex reconstruction (RAVE = "Reconstruction (of vertices) in Abstract, Versatile Environments") is presented that allows geometric and kinematic reconstruction of vertices. Both linear and adaptive estimation techniques are covered. Non-Gaussianinput data can be handled via the Gaussian-sum technique. Kinematic constraints are taken into account via the Lagrangian formalism. Finally, the toolkit also contains a simple flavor-tagger. Main design goals are ease of use, flexibility for embedding into existing software frameworks, extensibility, and openness. The implementation is based on modern object-oriented techniques, is coded in C++ with interfaces for Java and Python, and follows an open-source approach.

*Index Terms*—Event Reconstruction, Kalman Filter, Gaussian Sum Filter, Adaptive Method, Kinematic Fitting, Flavor Tagging.

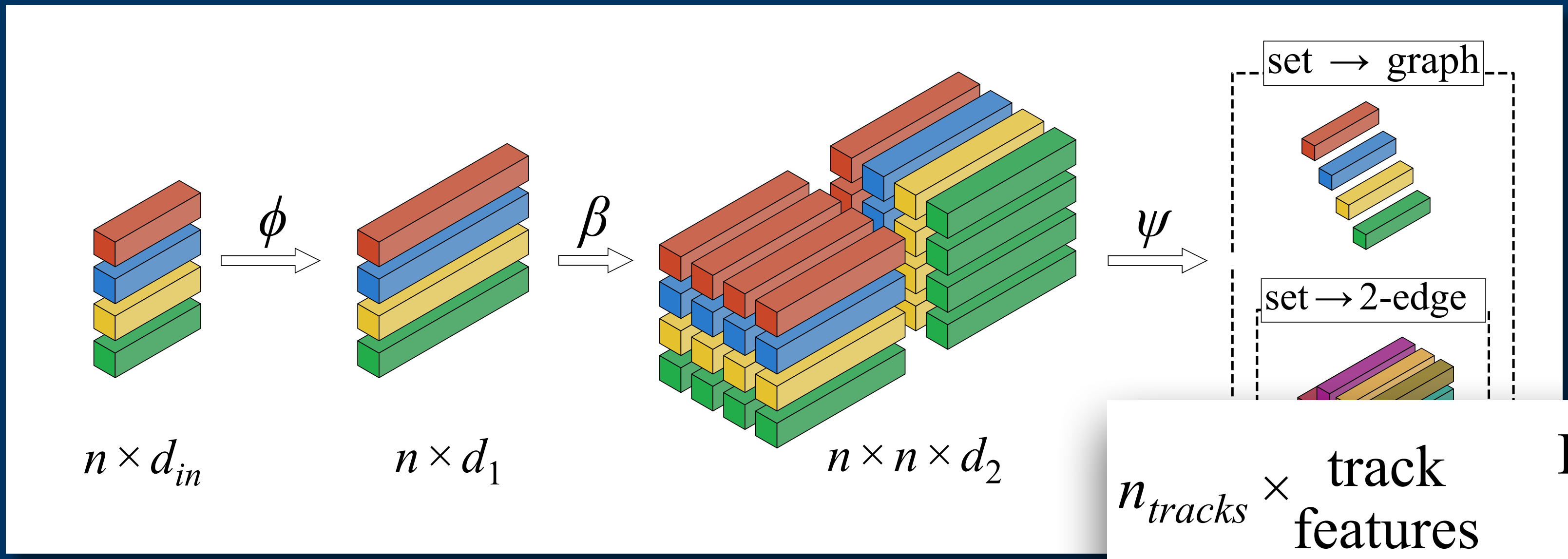
I. Introduction

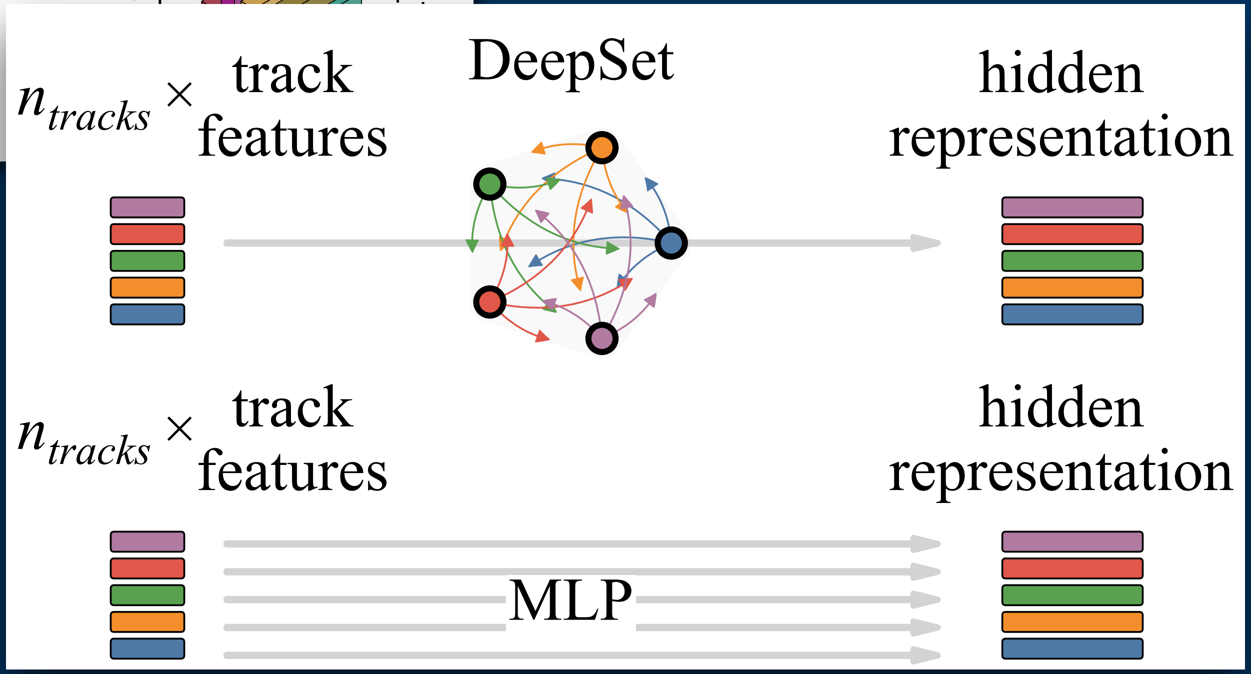
EXPERIMENTS at modern high-energy particle colliders rely on precise track and vertex reconstruction which must fully exploit the high spatial resolution achieved by state-

"geometric" case – refers to the reconstruction of not a single vertex but an entire decay tree of particles. Finally, to finish the naming conventions, flavor tagging refers to the task of infering the quark flavor of a given jet by examining the tracks and vertices in the jet.

Fig. 1. A primary vertex fitted from an early 2 x 450 GeV collision event at the CMS detector. The large "tubes" represent the reconstructed tracks, the widths representing the positional 2x2 subpart of the covariance matrix. The

Neural network "track pair classifier"



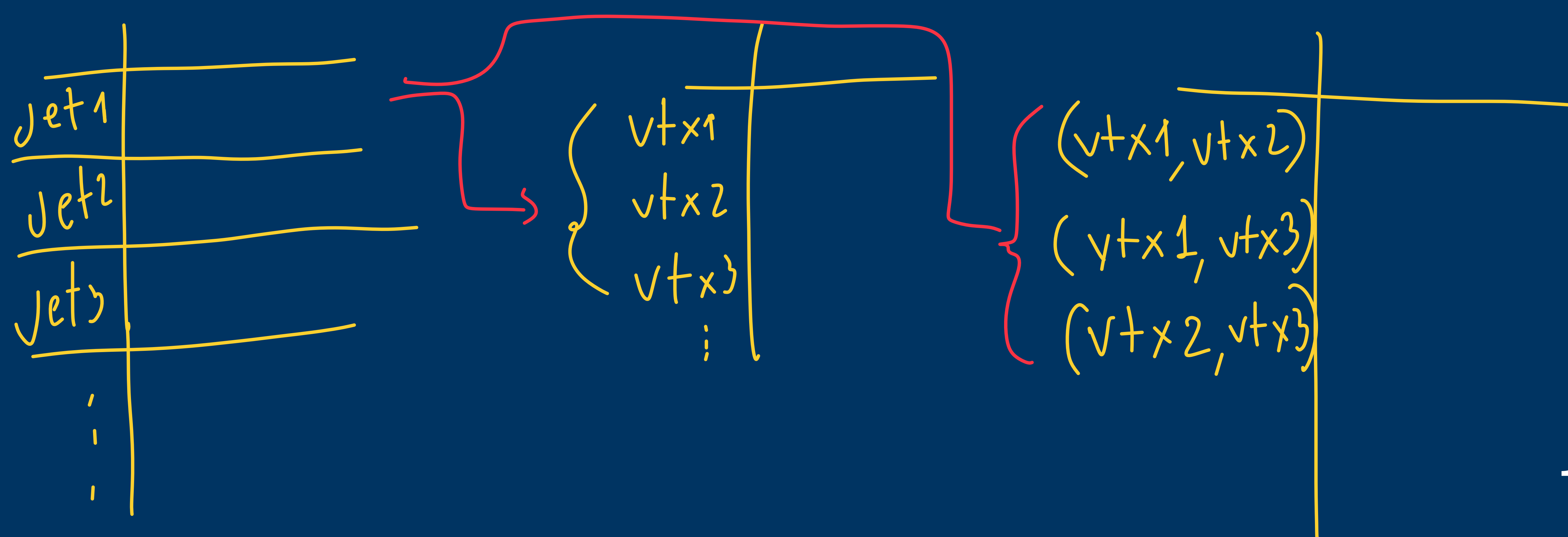




Answers the question - how important is the "big picture" of the other tracks in the jet in contrast to the pair of tracks in question.

# Evaluating the performance

Three perspectives:

**Jet** **Vertex** **Vertex-pair**



# Evaluating the performance

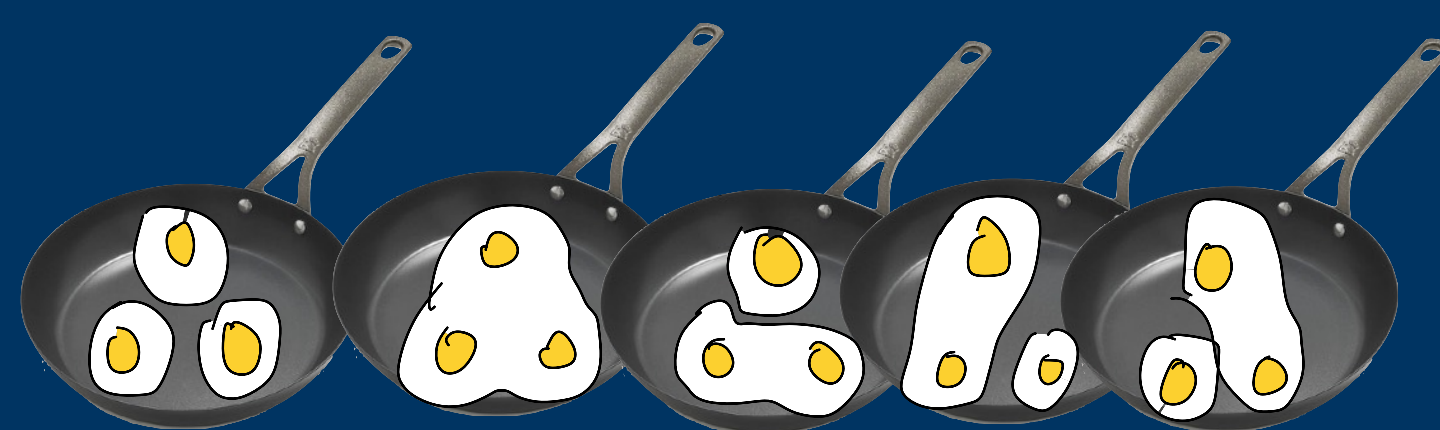
## Jet

Rand Index = $\dfrac{\text{True positives + true negatives}}{n \cdot (n-1)/2}$

## Adjusted Rand Index

What if we were just randomly guessing?

True (target) partition

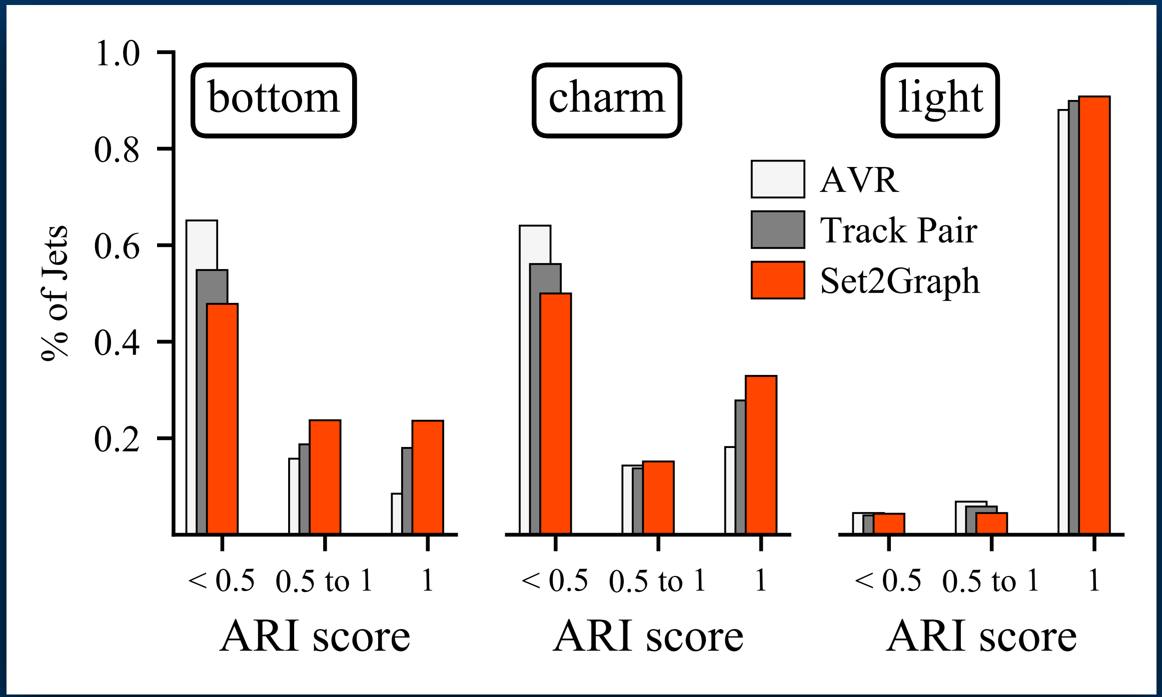


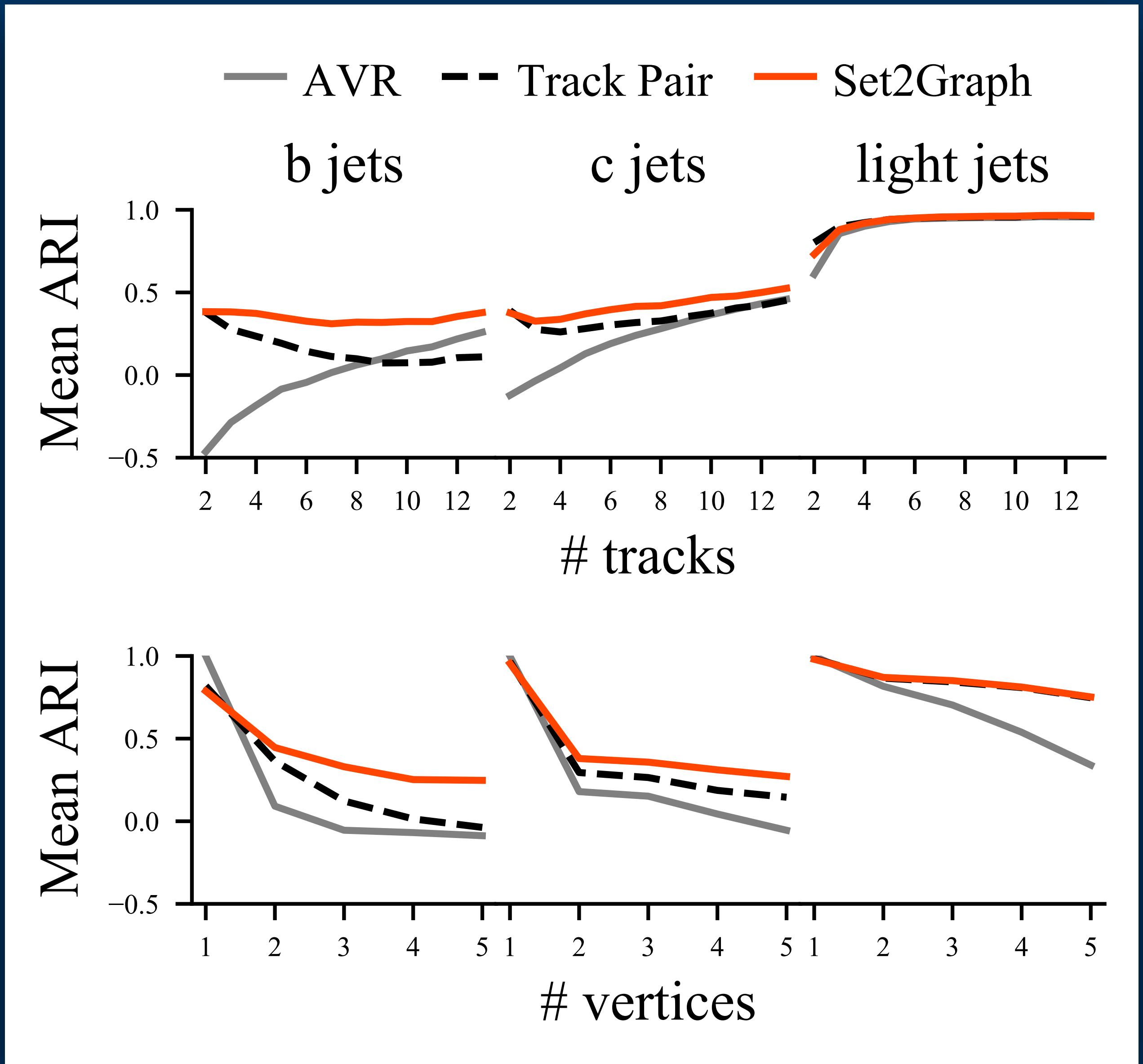
Compute the expectation value of the RI over all the possible partitions and normalise the RI

$$\text{ARI} = \frac{\text{RI} - \mathbb{E}[\text{RI}]}{1 - \mathbb{E}[\text{RI}]}$$

# Evaluating the performance

# Jet

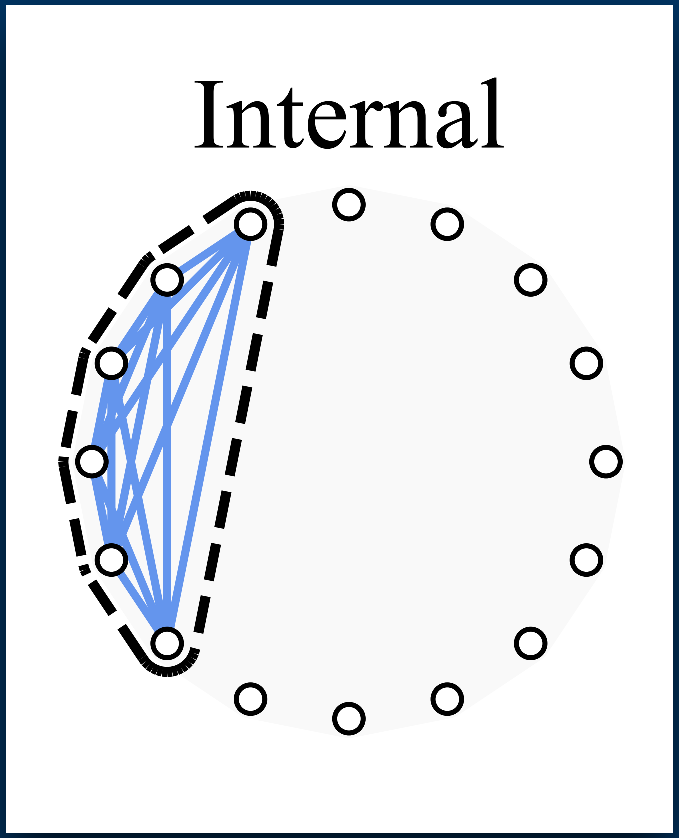


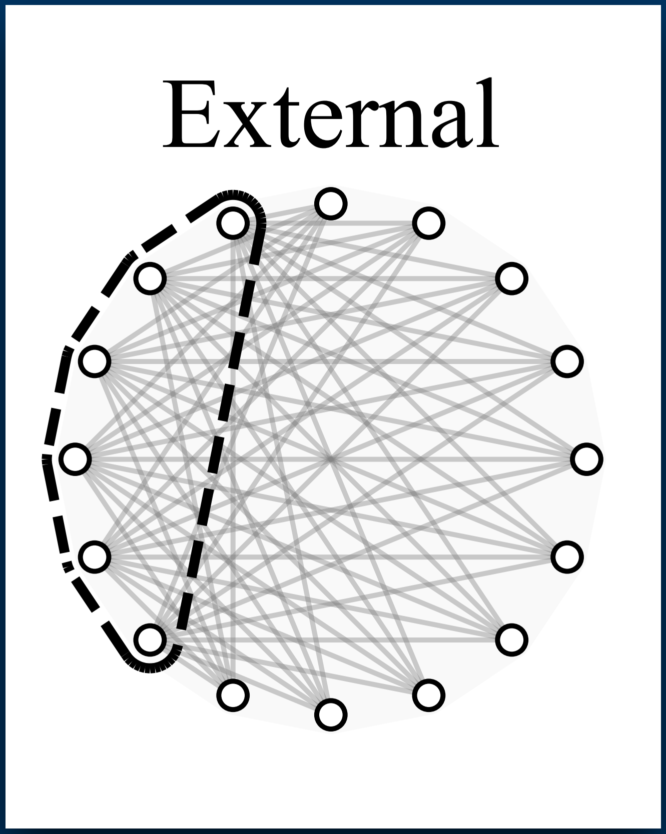



# Evaluating the performance

## Vertex

We can look at different kinds of edges and compute their accuracy - what percentage of them were predicted correctly

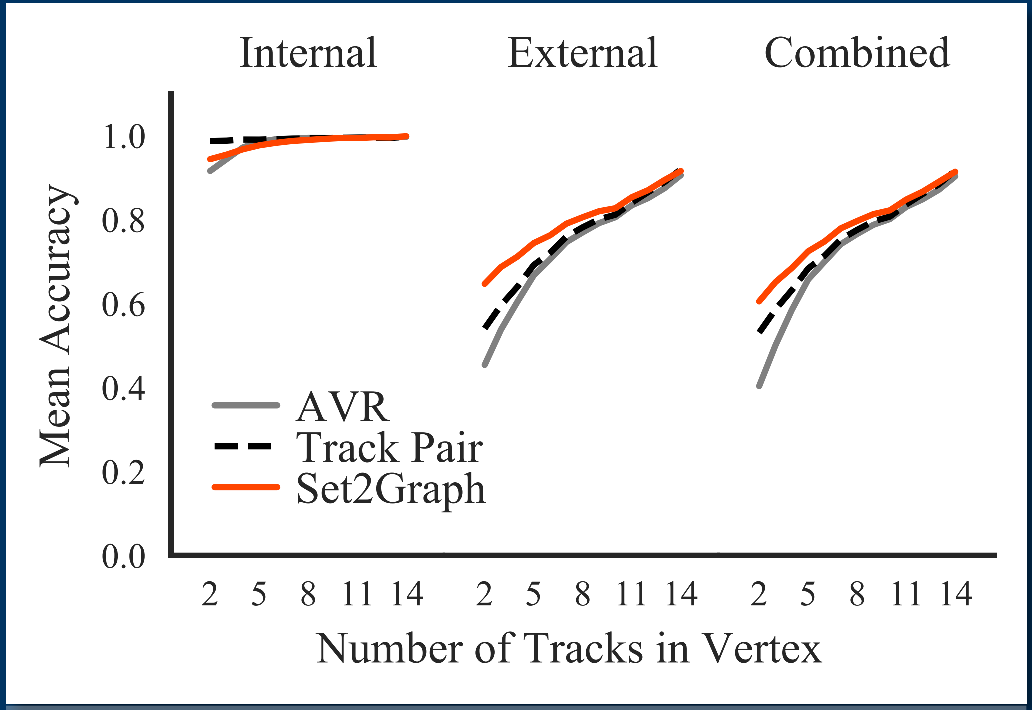


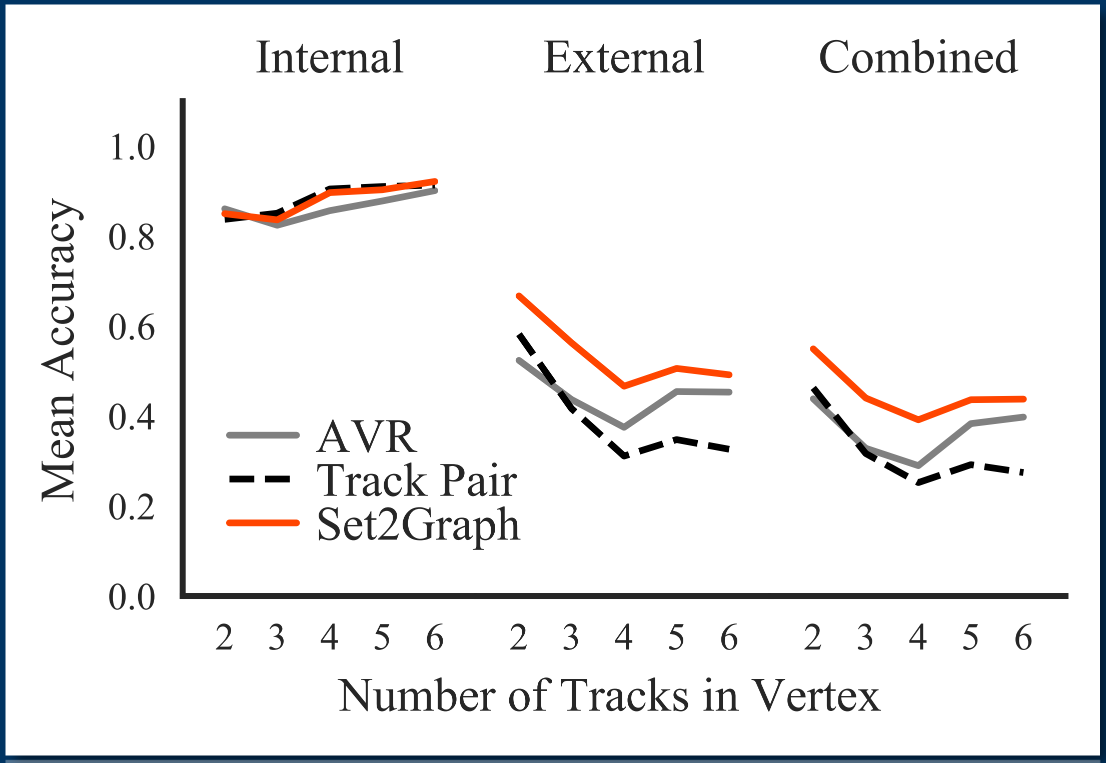
# Evaluating the performance

## Vertex

### Primary vertices



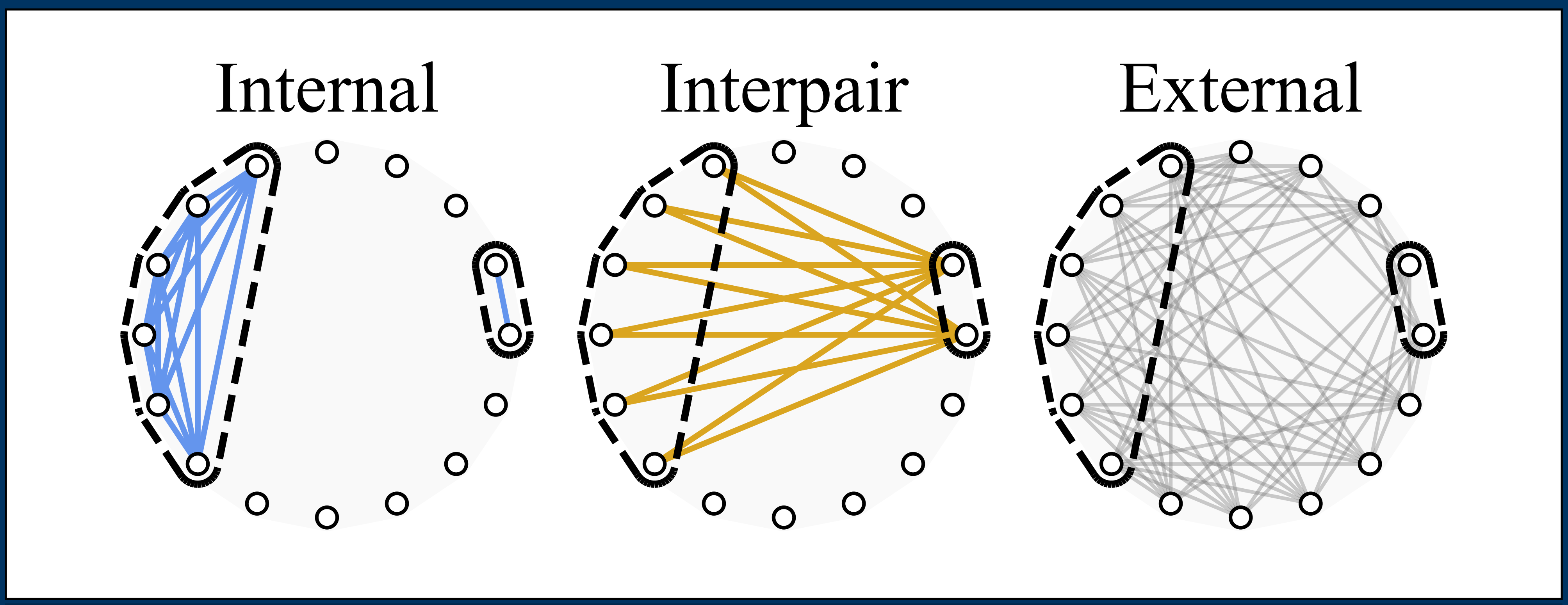

### Secondary vertices



# Evaluating the performance

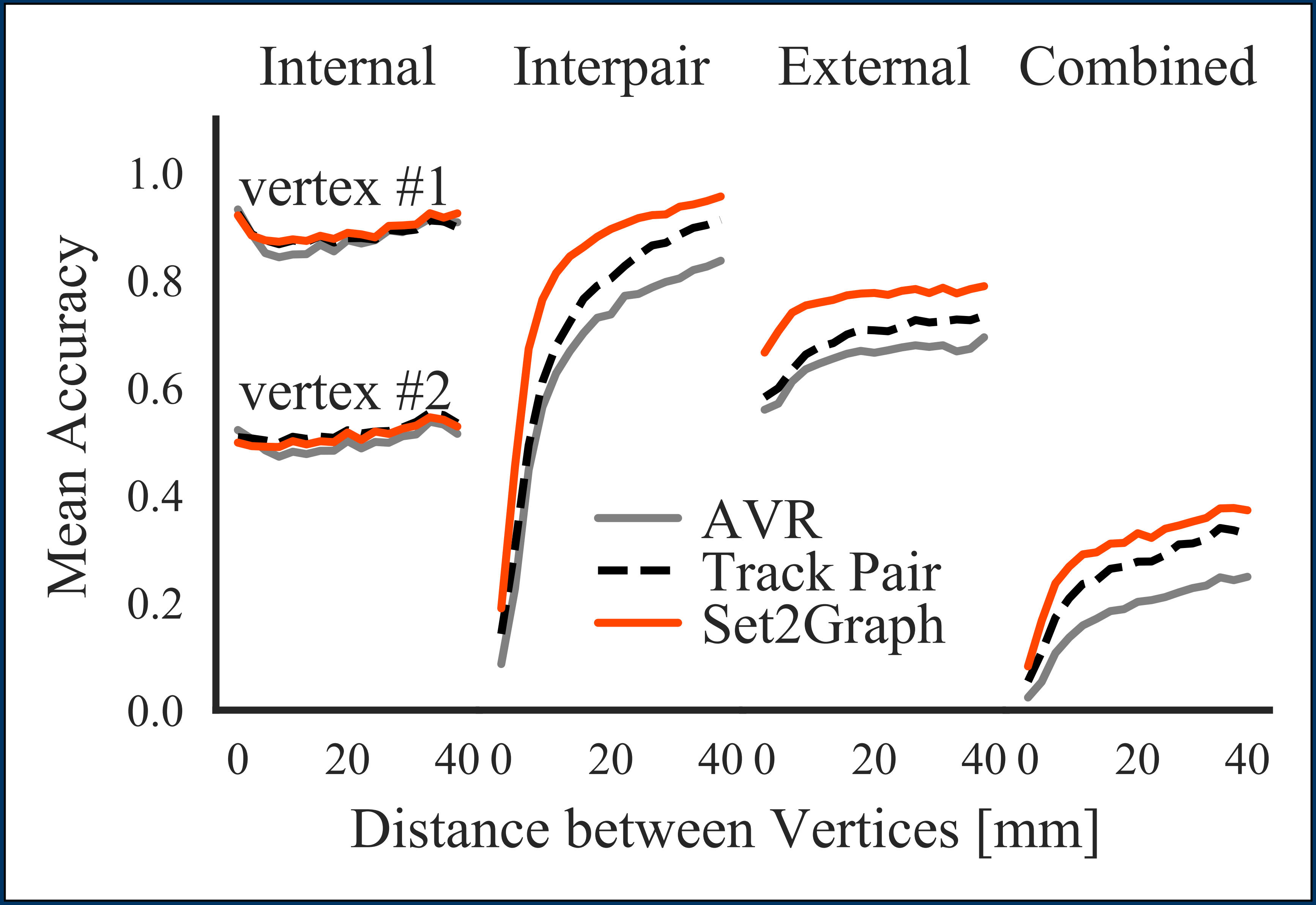
## Vertex-pair

We can look at different kinds of edges and compute their accuracy - what percentage of them were predicted correctly



# Evaluating the performance

## Vertex-pair



“math” paper

https://arxiv.org/abs/2002.08772

“physics” paper

https://arxiv.org/abs/2008.02831

arXiv:2002.08772v2 [cs.LG] 24 Jun 2020

# Set2Graph: Learning Graphs From Sets

Hadar Serviansky
Weizmann Institute of Science
Rehovot, Israel
hadar.serviansky@weizmann.ac.il

Nimrod Segol
Weizmann Institute of Science
Rehovot, Israel

Jonathan Shlomi
Weizmann Institute of Science
Rehovot, Israel

Kyle Cranmer
New York University
New-York, USA

Eilam Gross
Weizmann Institute of Science
Rehovot, Israel

Haggai Maron
NVIDIA Research

Yaron Lipman
Weizmann Institute of Science
Rehovot, Israel

## Abstract

Many problems in machine learning can be cast as learning functions from sets to graphs, or more generally to hypergraphs; in short, Set2Graph functions. Examples include clustering, learning vertex and edge features on graphs, and learning features on triplets in a collection. A natural approach for building Set2Graph models is to characterize all linear equivariant set-to-hypergraph layers and stack them with non-linear activations. This poses two challenges: (i) the expressive power of these networks is not well understood; and (ii) these models would suffer from high, often intractable computational and memory complexity, as their dimension grows exponentially. This paper advocates a family of neural network models for learning Set2Graph functions that is both practical and of maximal expressive power (universal), that is, can approximate arbitrary continuous Set2Graph functions over compact sets. Testing these models on different machine learning tasks, mainly an application to particle physics, we find them favorable to existing baselines.

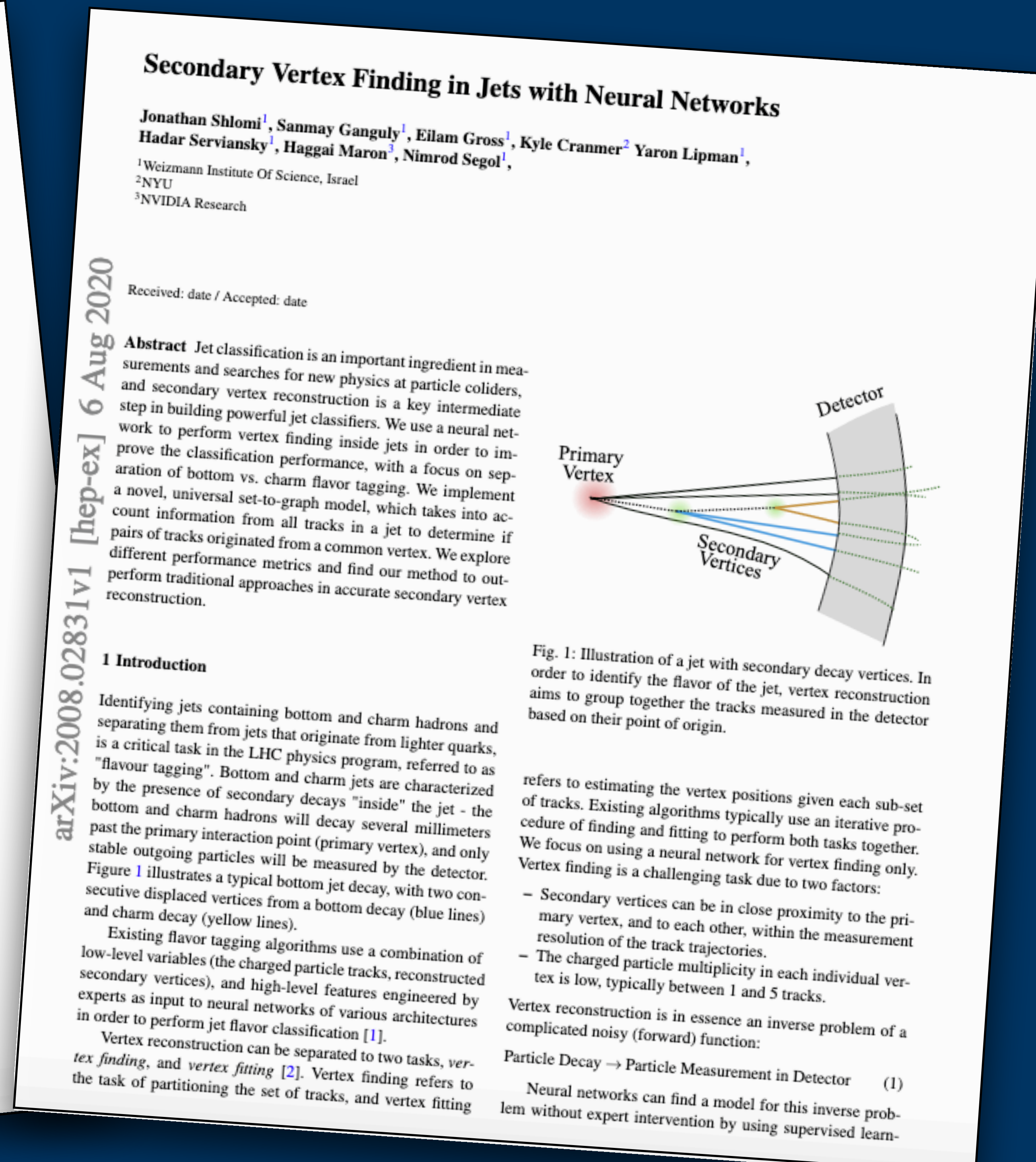
## 1 Introduction

We consider the problem of learning functions taking sets of vectors in $\mathbb{R}^{d_{in}}$ to graphs, or more generally hypergraphs; we name this problem Set2Graph, or set-to-graph. Set-to-graph functions appear in machine-learning applications such as clustering, predicting features on edges and nodes in graphs, and learning $k$-edge information in sets.

Mathematically, we represent each set-to-graph function as a collection of set-to-$k$-edge functions, where each set-to-$k$-edge function learns features on $k$-edges. That is, given an input set $\mathcal{X} = \{x_1, \ldots, x_n\} \subset \mathbb{R}^{d_{in}}$ we consider functions $\mathbf{F}^k$ attaching feature vectors to $k$-edges: each $k$-tuple $(x_{i_1}, \ldots, x_{i_k})$ is assigned with an output vector $\mathbf{F}^k(\mathcal{X})_{i_1,i_2,\ldots,i_k,:} \in \mathbb{R}^{d_{out}}$. Now, functions mapping sets to hypergraphs with hyper-edges of size



Figure 1: Set-to-graph functions are represented as collections of set-to-k-edge functions.



arXiv:2008.02831v1 [hep-ex] 6 Aug 2020

# Secondary Vertex Finding in Jets with Neural Networks

Jonathan Shlomi[1], Sanmay Ganguly[1], Eilam Gross[1], Kyle Cranmer[2] Yaron Lipman[1], Hadar Serviansky[1], Haggai Maron[3], Nimrod Segol[1],

[1]Weizmann Institute Of Science, Israel
[2]NYU
[3]NVIDIA Research

Received: date / Accepted: date

**Abstract** Jet classification is an important ingredient in measurements and searches for new physics at particle coliders, and secondary vertex reconstruction is a key intermediate step in building powerful jet classifiers. We use a neural network to perform vertex finding inside jets in order to improve the classification performance, with a focus on separation of bottom vs. charm flavor tagging. We implement a novel, universal set-to-graph model, which takes into account information from all tracks in a jet to determine if pairs of tracks originated from a common vertex. We explore different performance metrics and find our method to outperform traditional approaches in accurate secondary vertex reconstruction.



Fig. 1: Illustration of a jet with secondary decay vertices. In order to identify the flavor of the jet, vertex reconstruction aims to group together the tracks measured in the detector based on their point of origin.

### 1 Introduction

Identifying jets containing bottom and charm hadrons and separating them from jets that originate from lighter quarks, is a critical task in the LHC physics program, referred to as "flavour tagging". Bottom and charm jets are characterized by the presence of secondary decays "inside" the jet - the bottom and charm hadrons will decay several millimeters past the primary interaction point (primary vertex), and only stable outgoing particles will be measured by the detector. Figure 1 illustrates a typical bottom jet decay, with two consecutive displaced vertices from a bottom decay (blue lines) and charm decay (yellow lines).

Existing flavor tagging algorithms use a combination of low-level variables (the charged particle tracks, reconstructed secondary vertices), and high-level features engineered by experts as input to neural networks of various architectures in order to perform jet flavor classification [1].

Vertex reconstruction can be separated to two tasks, *vertex finding*, and *vertex fitting* [2]. Vertex finding refers to the task of partitioning the set of tracks, and vertex fitting refers to estimating the vertex positions given each sub-set of tracks. Existing algorithms typically use an iterative procedure of finding and fitting to perform both tasks together. We focus on using a neural network for vertex finding only. Vertex finding is a challenging task due to two factors:

– Secondary vertices can be in close proximity to the primary vertex, and to each other, within the measurement resolution of the track trajectories.
– The charged particle multiplicity in each individual vertex is low, typically between 1 and 5 tracks.

Vertex reconstruction is in essence an inverse problem of a complicated noisy (forward) function:

Particle Decay → Particle Measurement in Detector (1)

Neural networks can find a model for this inverse problem without expert intervention by using supervised learn-

Code and Dataset:

https://github.com/hadarser/SetToGraphPaper/

https://zenodo.org/record/4044628

DOI 10.5281/zenodo.4044628

# Summary

- Neural networks are useful for secondary vertex finding
- Set2Graph model is universal
- S2G model outperforms traditional approach in a variety of performance metrics

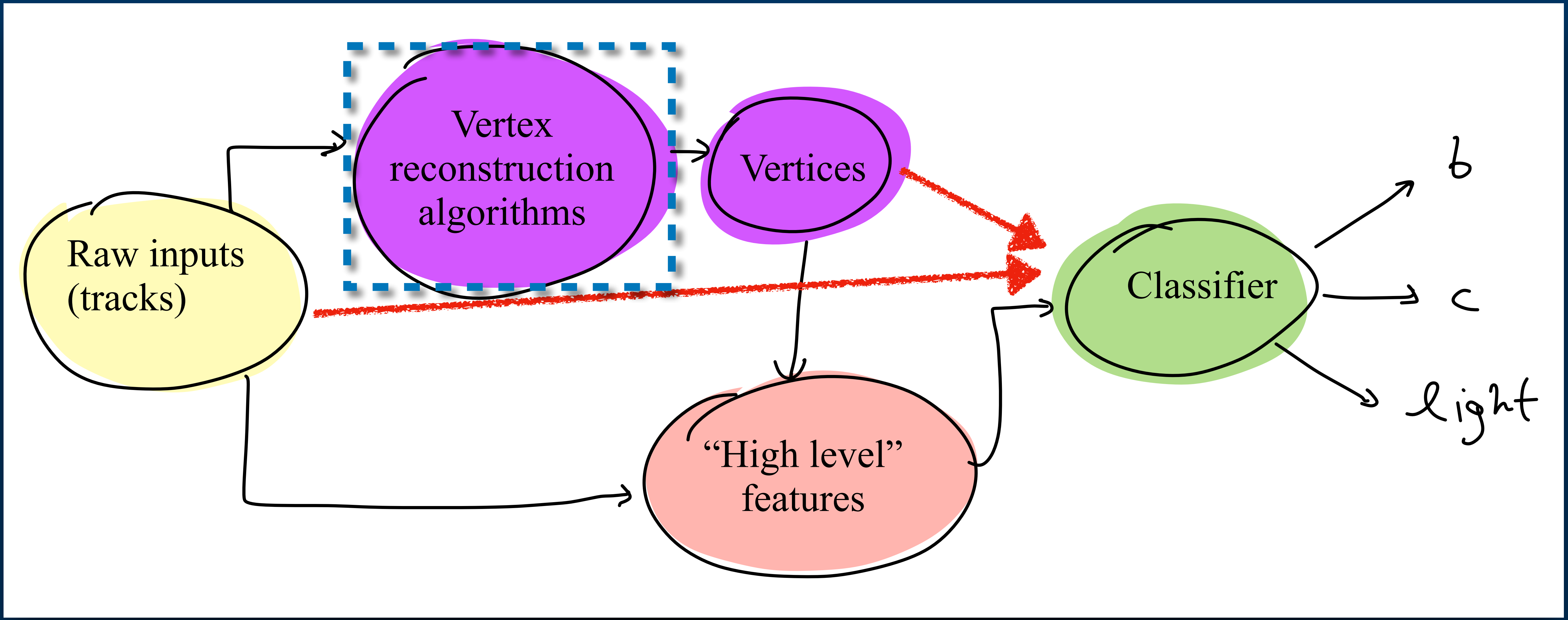
# Future directions

Train the model on full simulation / apply on real data - how does it perform compared to existing algorithms?

How does it impact performance for downstream tasks?

Use neural networks to learn vertex fitting

The underlying question is, does using ML to do better reconstruction help downstream?



Thank you for your attention!