Track Finding in the Velo at LHCb using Graph Neural Networks

Anthony Correia

15th December 2025

Protection zone

In collaboration with

1 **Beginner Introduction**

2 Neural Network Introduction

3 Problem Formulation

4 Experimental Setup

5 Exa.TrkX Pipeline

6 From Exa.TrkX to ETX4VELO

7 Implementation in Allen

8 Optimization

1 Beginner Introduction

a Particle Physics

b LHCb Detector

c Track Finding

d Summary

1 Beginner Introduction

a Particle Physics

Glass of Water

1 Beginner Introduction

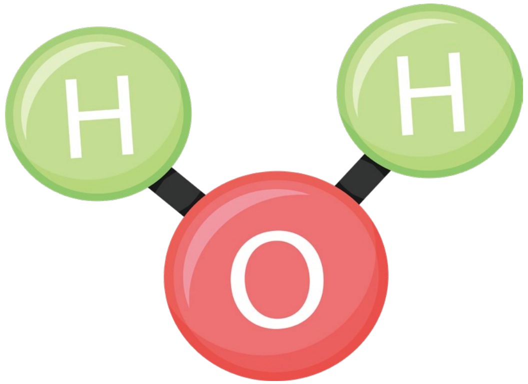
a Particle Physics

Glass of Water

Molecule H_20

1 atome O

+ 2 atomes H



1 Beginner Introduction

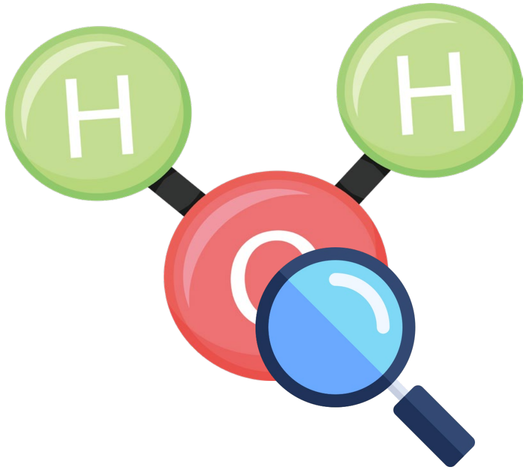
a Particle Physics

Glass of Water

Molecule H_20

1 atome O

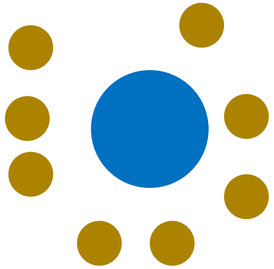
+ 2 atomes H



Atom O

Nucleus O

+ 8 **electrons e^-**

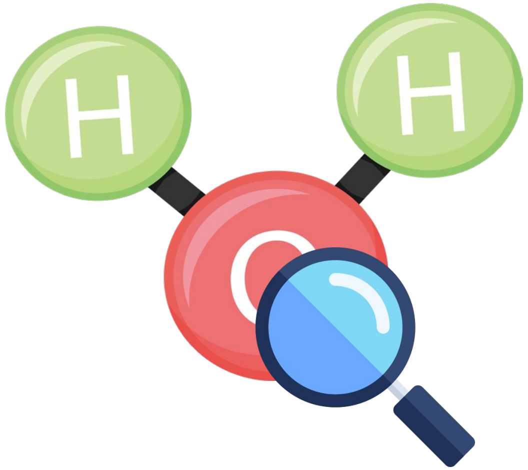


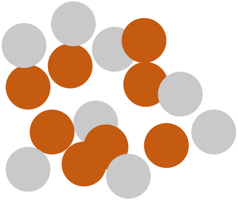
Not in scale!

1 Beginner Introduction

a Particle Physics

Glass of Water	Molecule H_20	Atom O	Nucleus O
	1 atome O + 2 atomes H	Nucleus O + 8 **electrons** e^-	8 **protons p** + 8 **neutrons n**

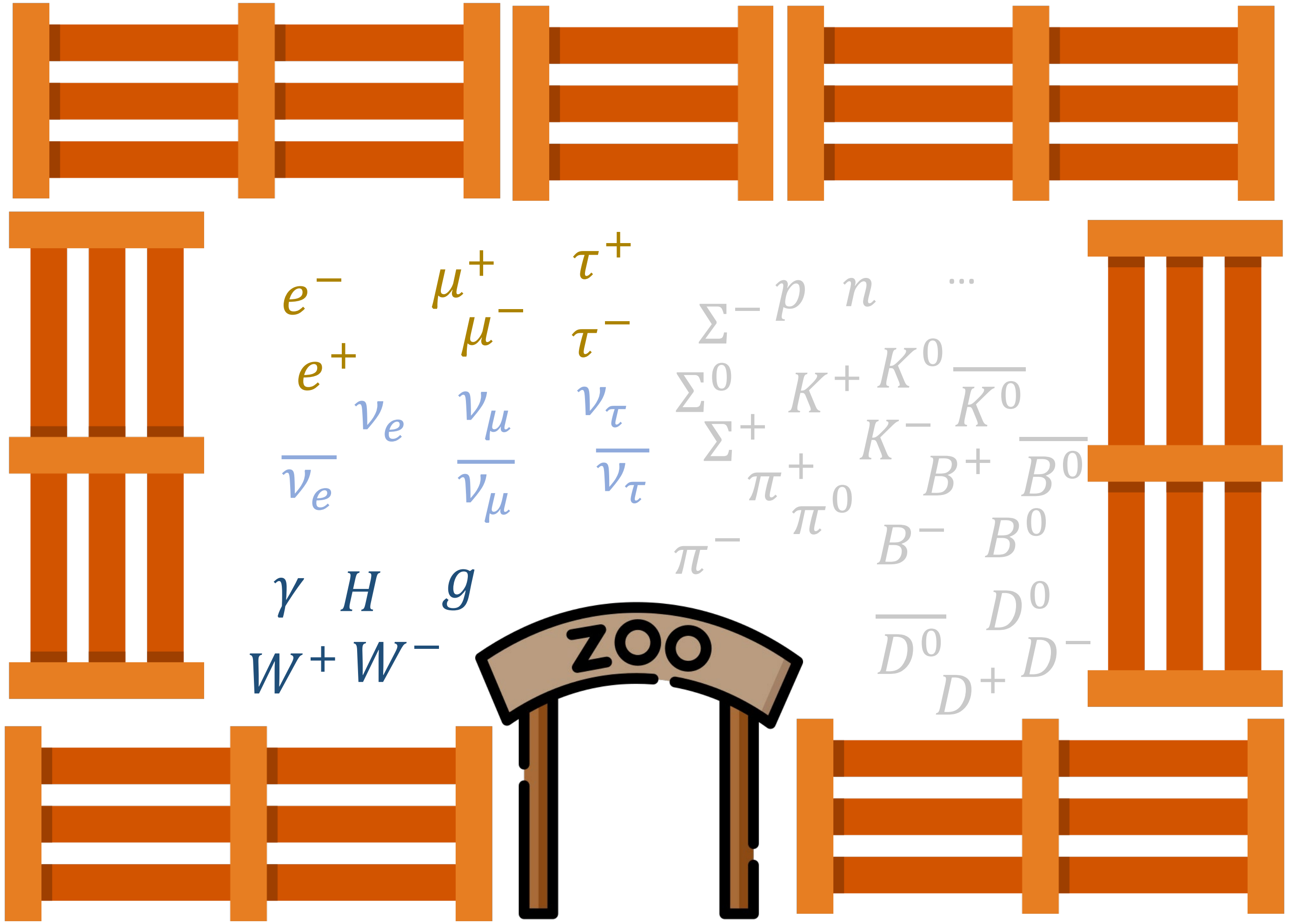




Not in scale!

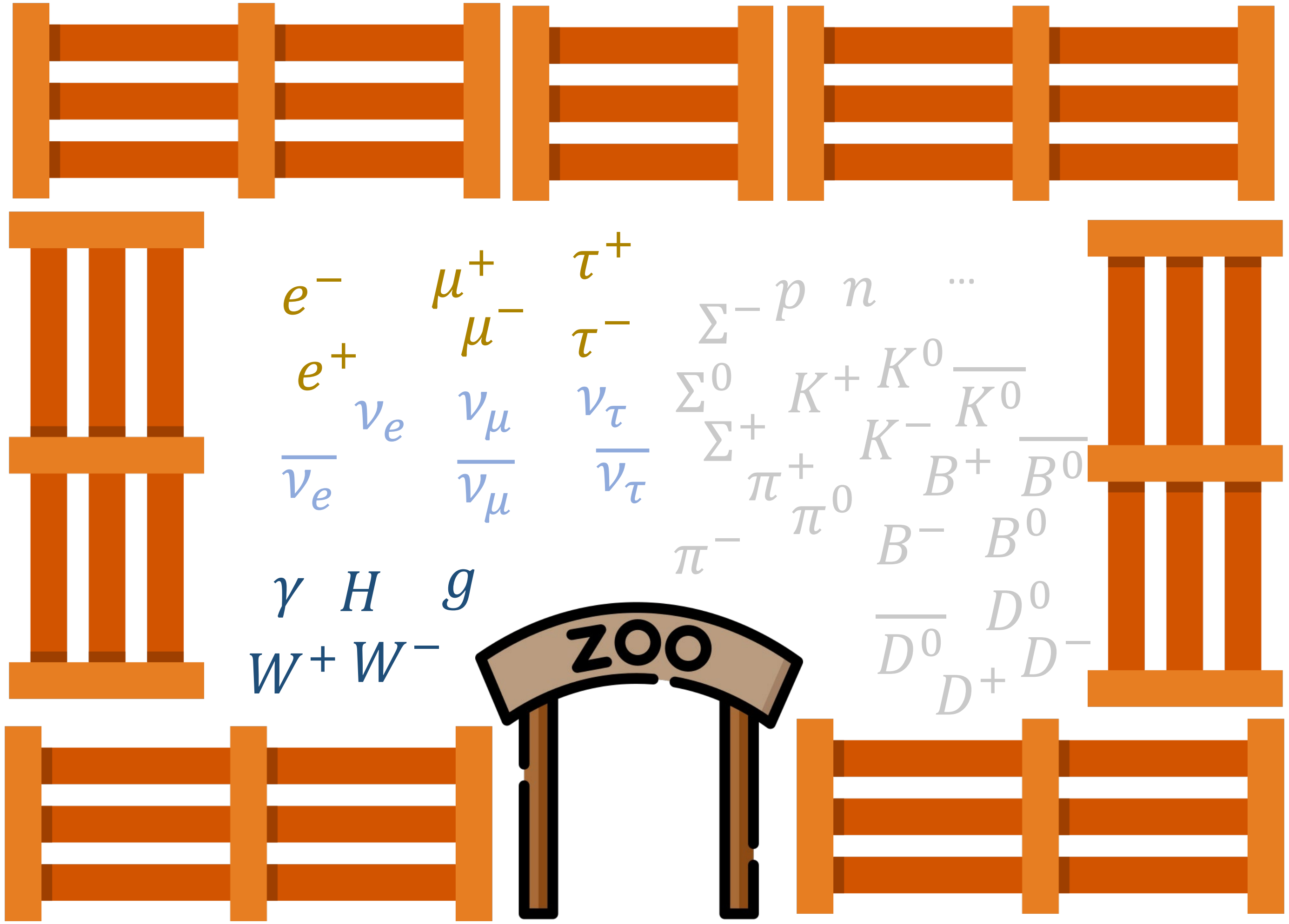
1 Beginner Introduction

a Particle Physics

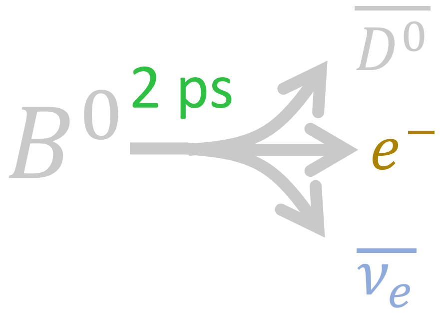


1 Beginner Introduction

a Particle Physics

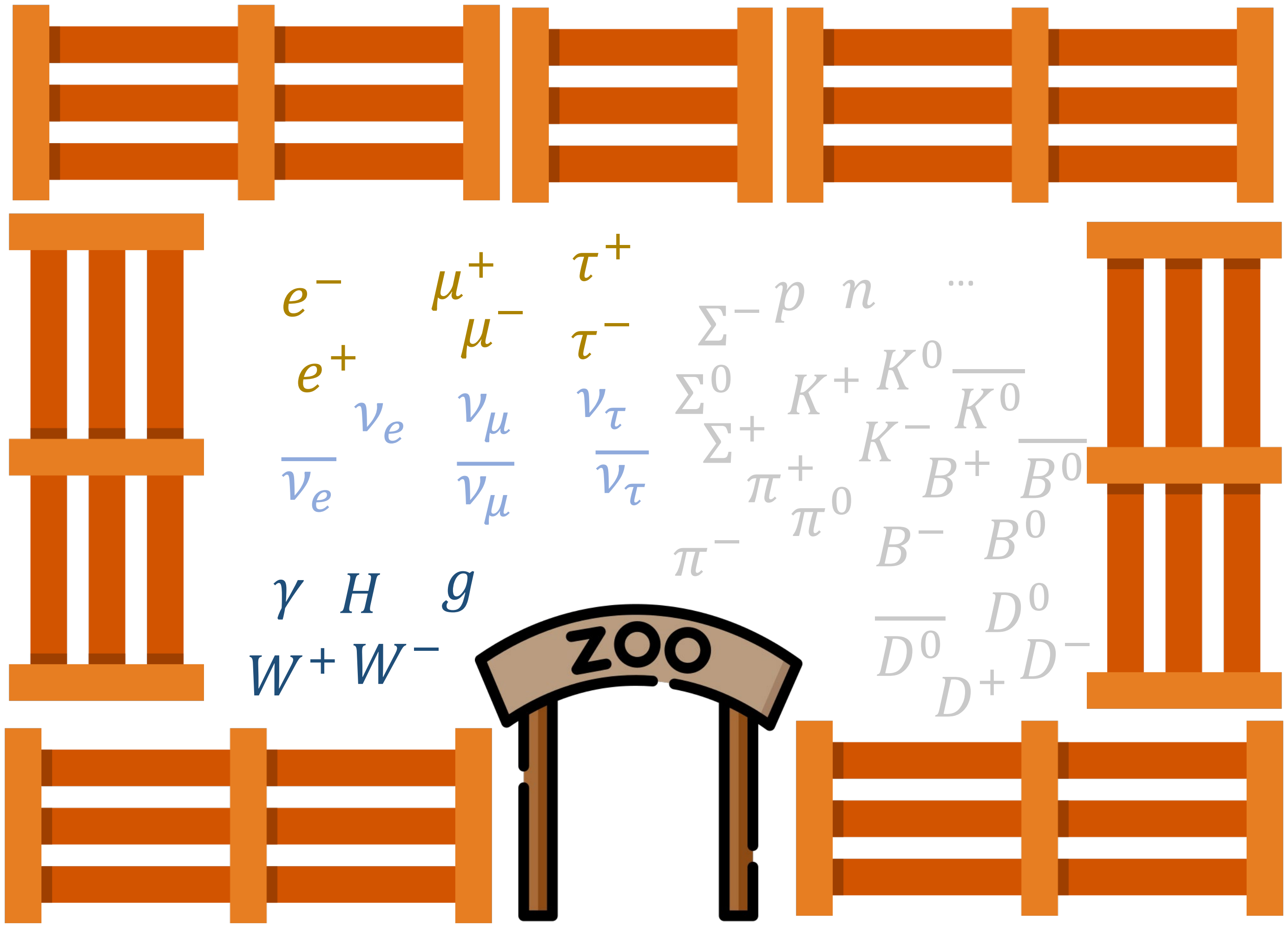


Somes particles very **unstable**.
They **decay** really fast.

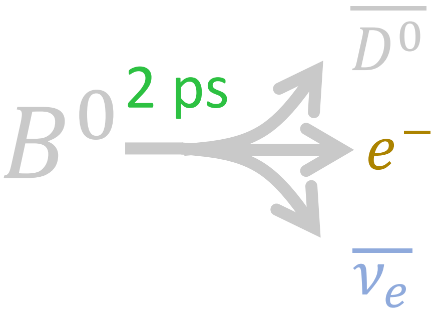


1 Beginner Introduction

a Particle Physics



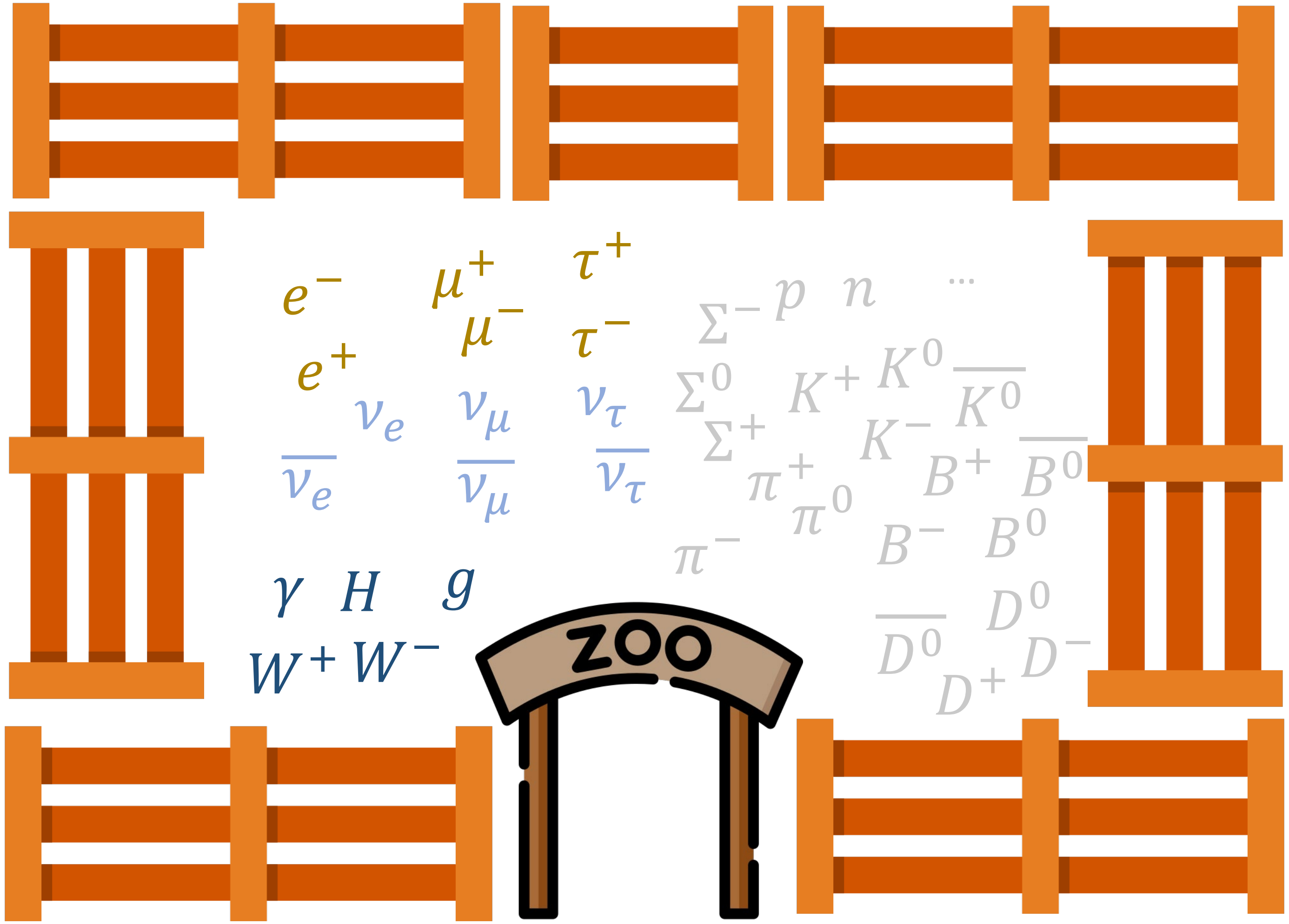
Somes particles very **unstable**.
They **decay** really fast.



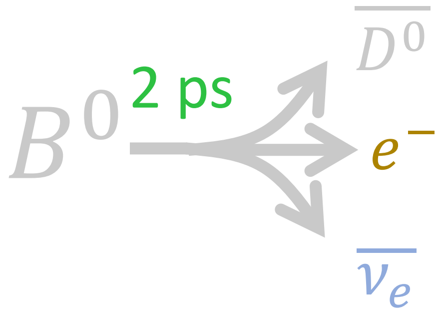
Standard Model: theory of particle physics

1 Beginner Introduction

a Particle Physics



Somes particles very **unstable**.
They **decay** really fast.



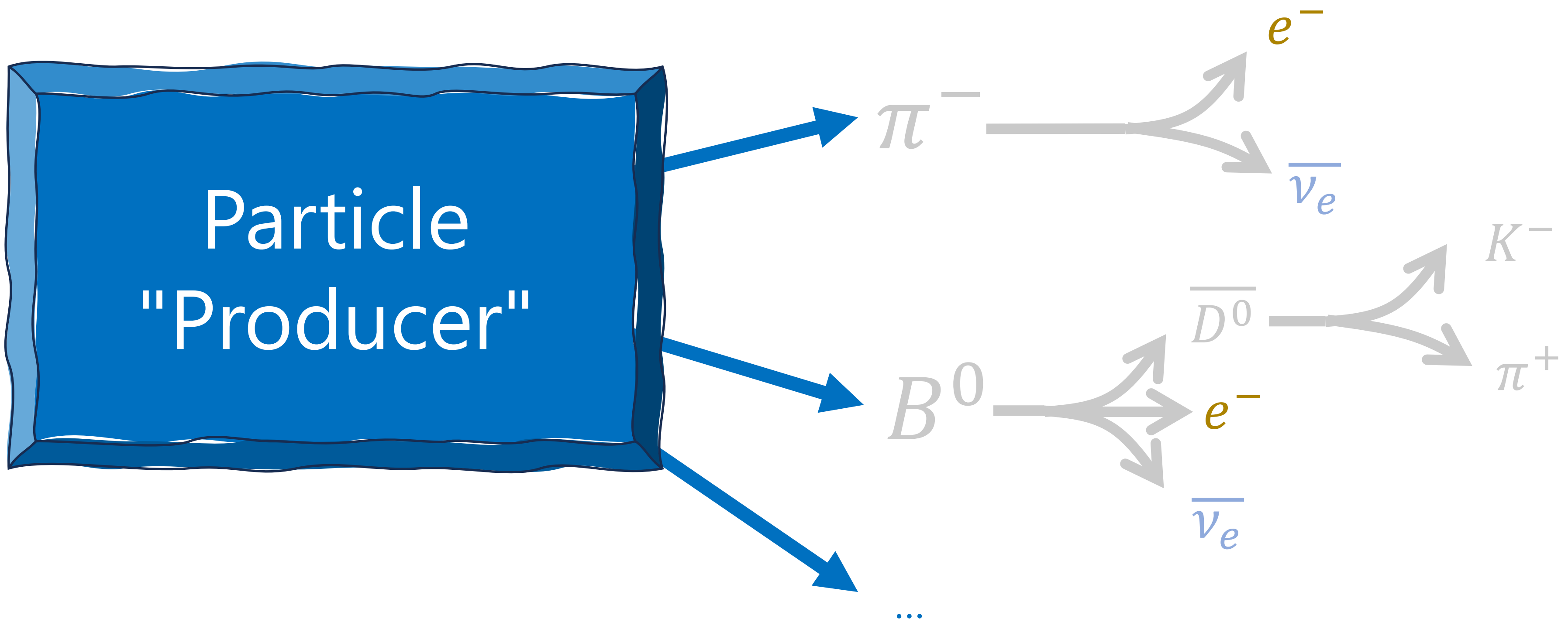
Standard Model: theory of particle physics

How to study them?

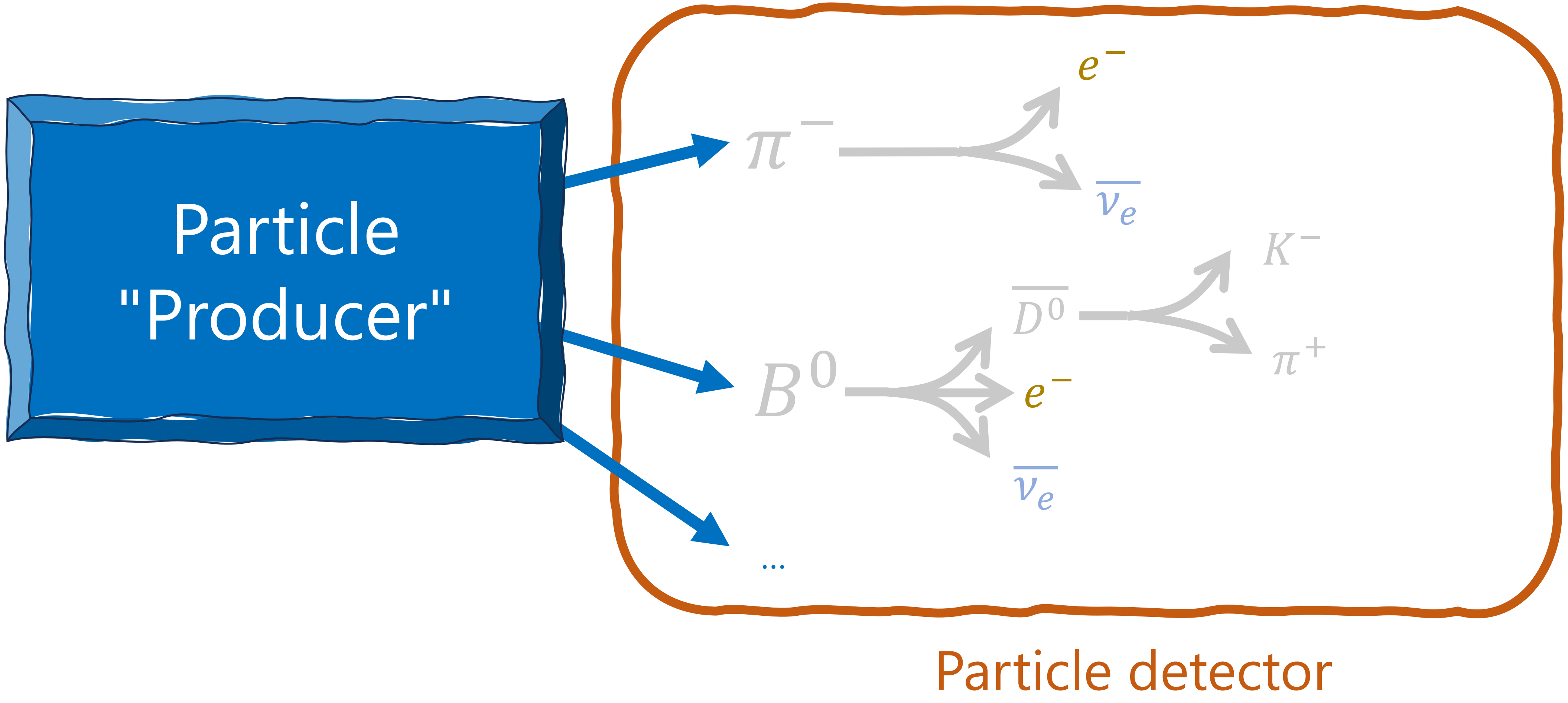
1 Beginner Introduction

a Particle Physics

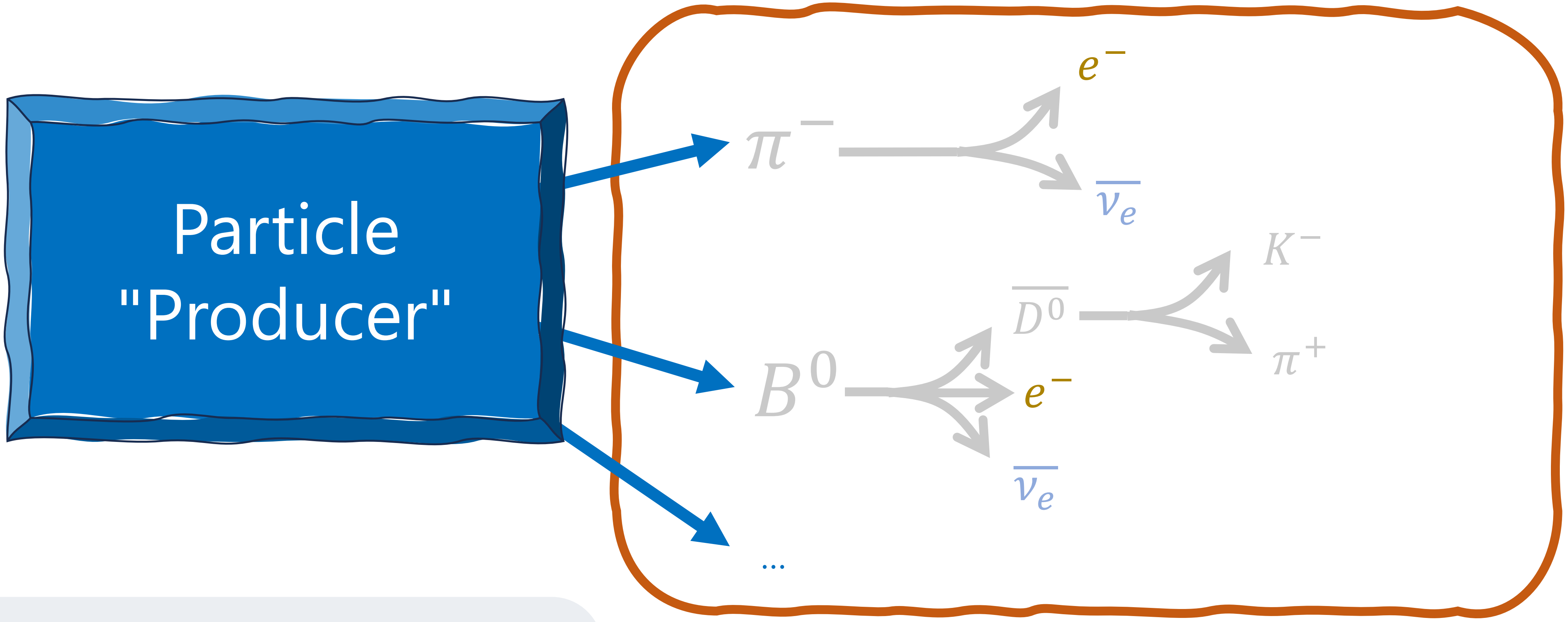
We produce them and detect them right away.



We produce them and detect them right away.



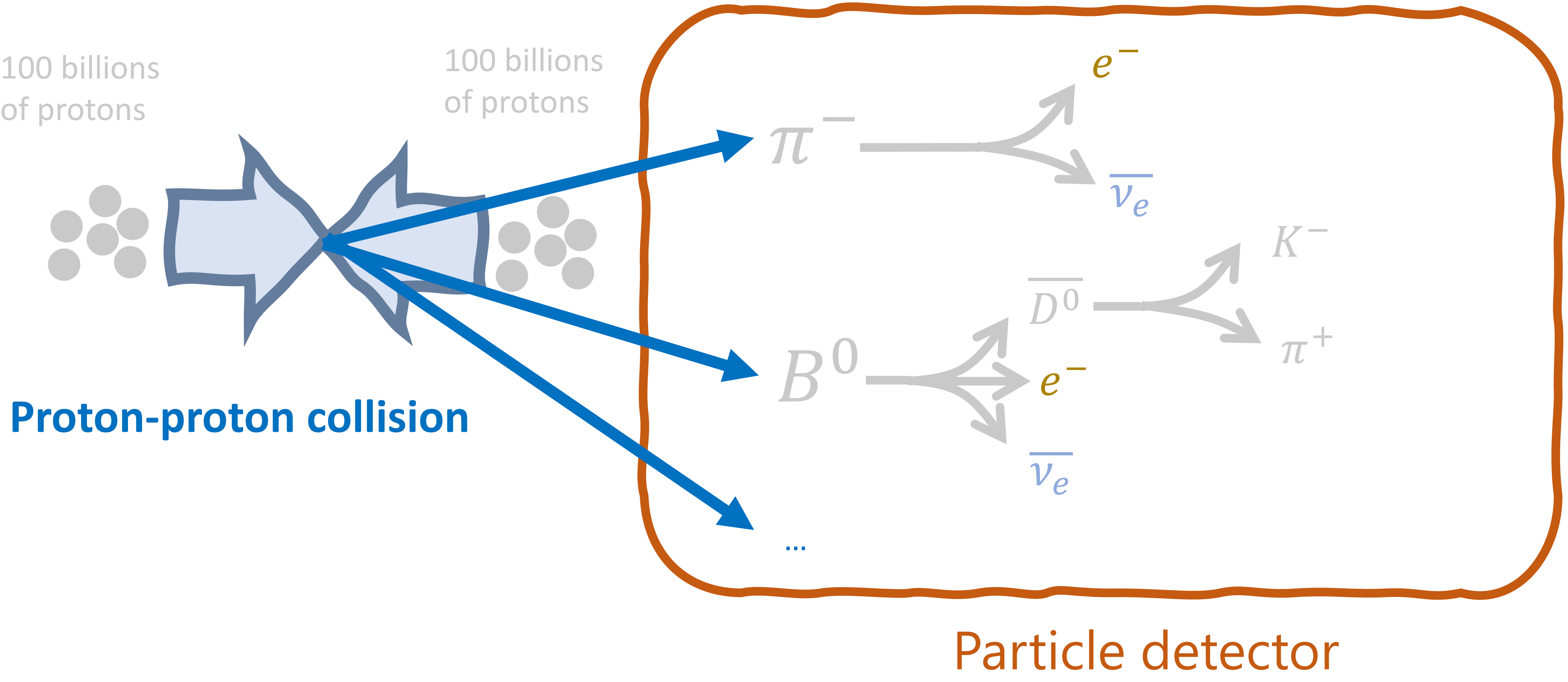
We produce them and detect them right away.



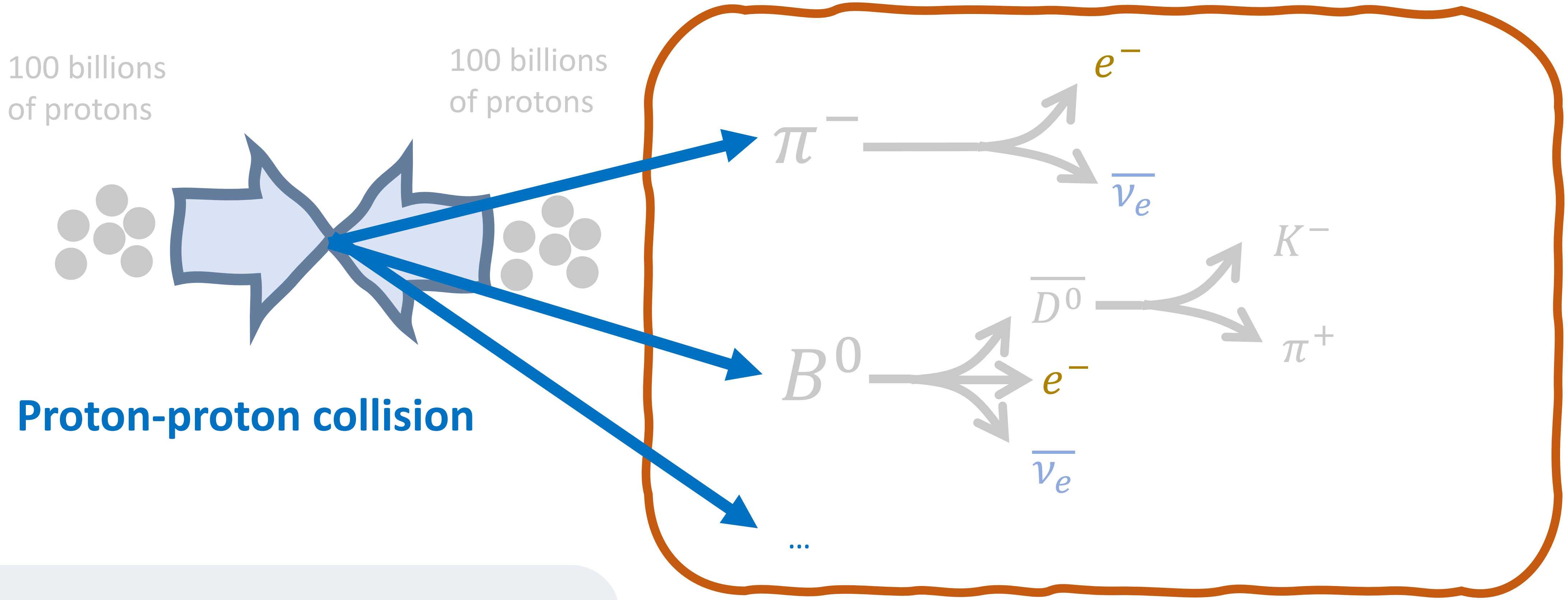
Particle detector

How to **produce them**?

We produce them and detect them right away.



We produce them and detect them right away.



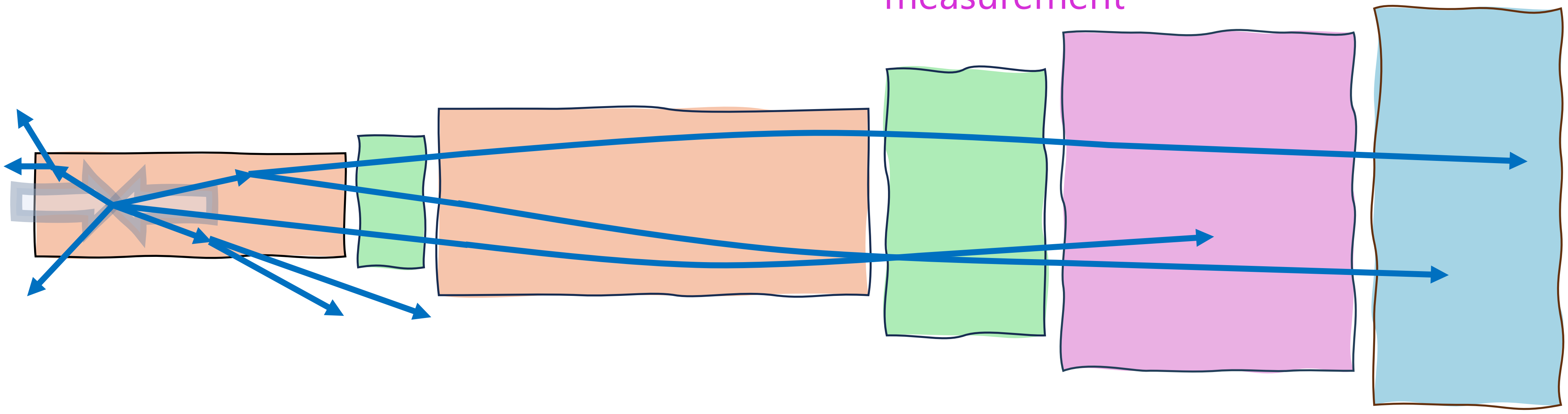
How to **detect them**?

Particle detector

1 Beginner Introduction

b LHCb Detector

Tracking detector	RICH detectors	Calorimeters	Muon Chamber
Measure trajectories	Identification	Energy measurement	Identification



1 Beginner Introduction

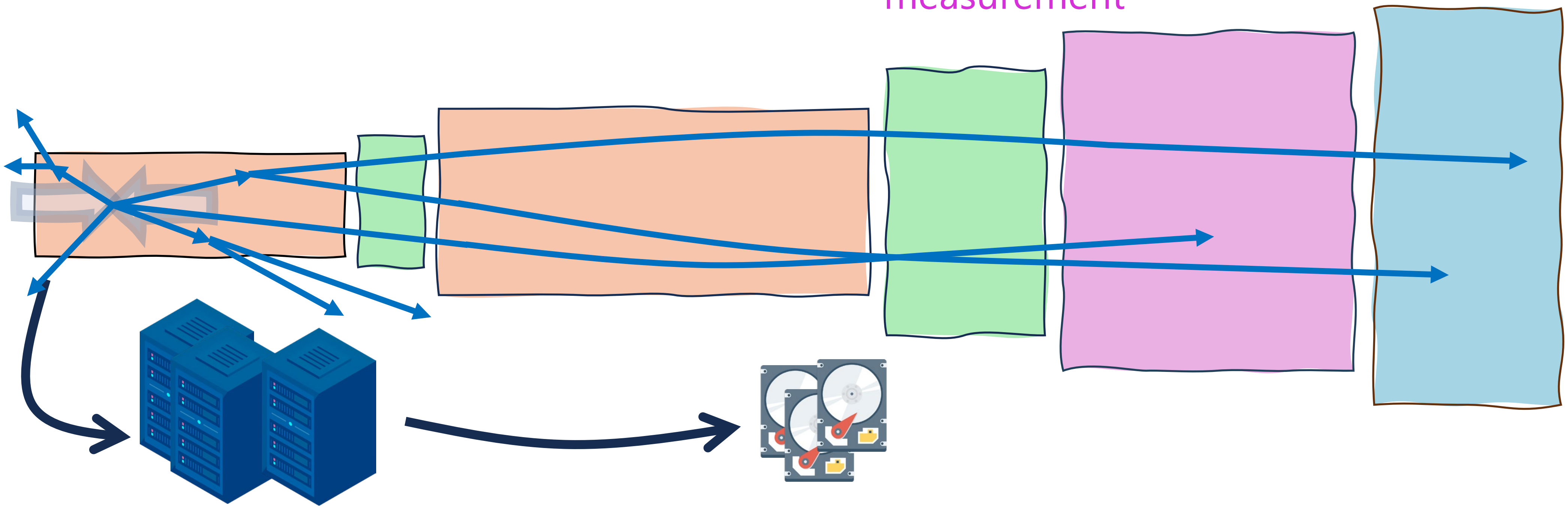
b LHCb Detector

Tracking detector — **Measure trajectories**

RICH detectors — Identification

Calorimeters — Energy measurement

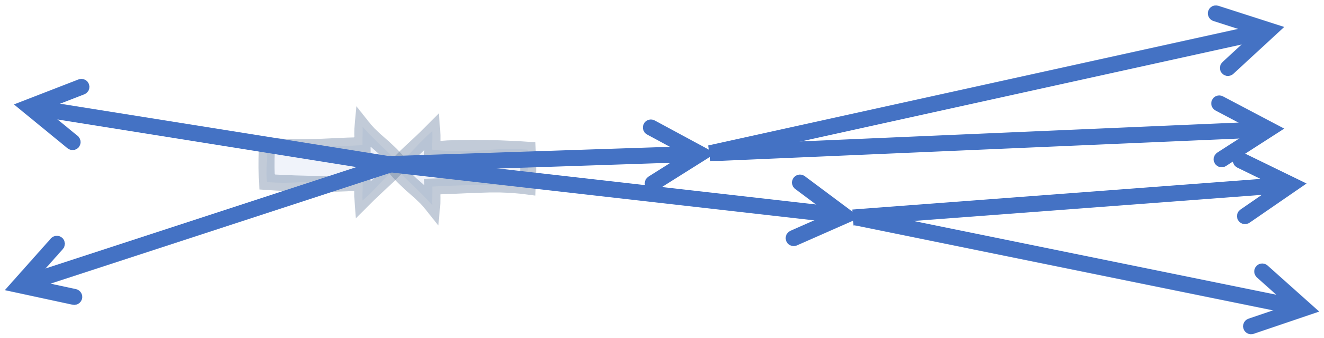
Muon Chamber — Identification



Collision **reconstructed** using computers, and saved to disk

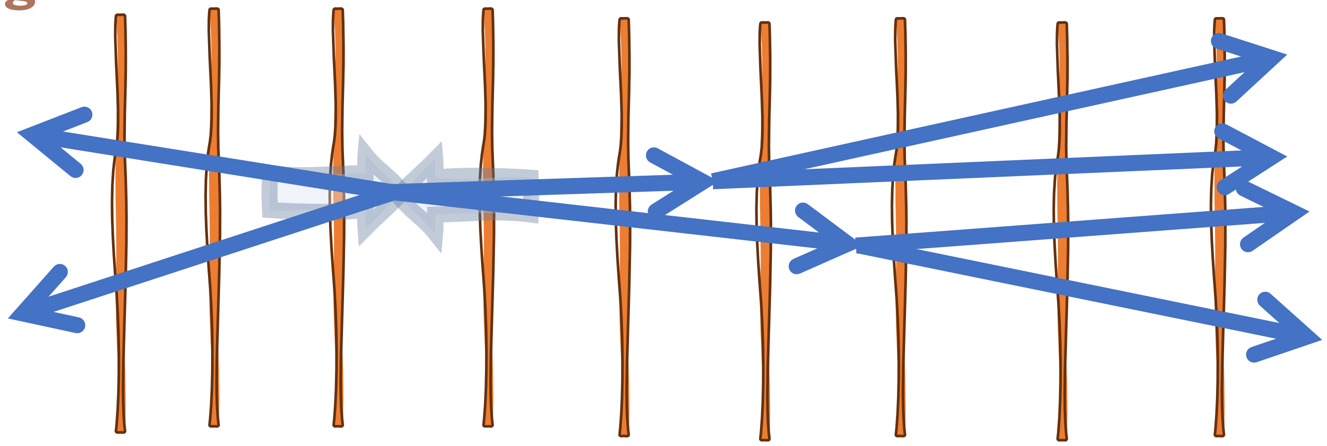
1 Beginner Introduction

C Track Finding



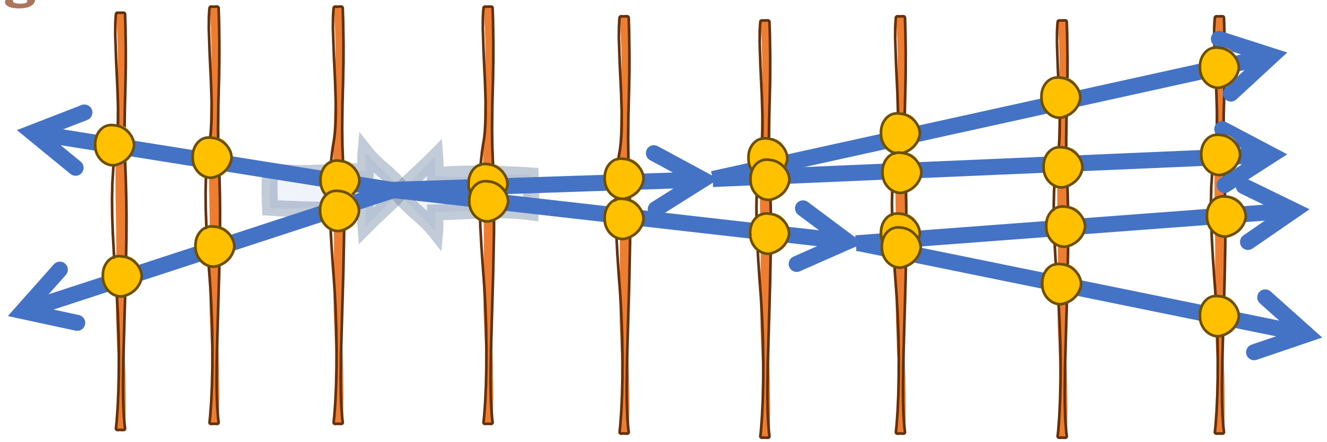
1 Beginner Introduction

C Track Finding



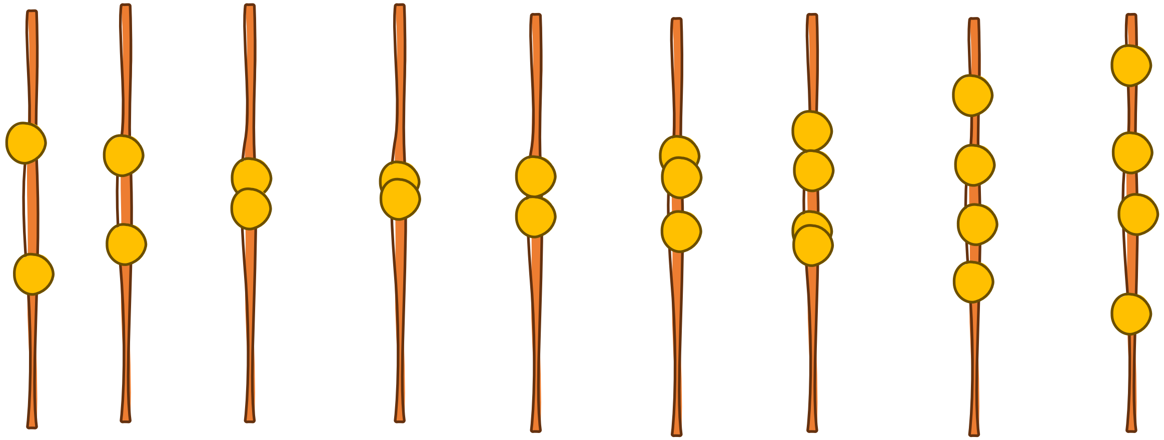
1 Beginner Introduction

C Track Finding



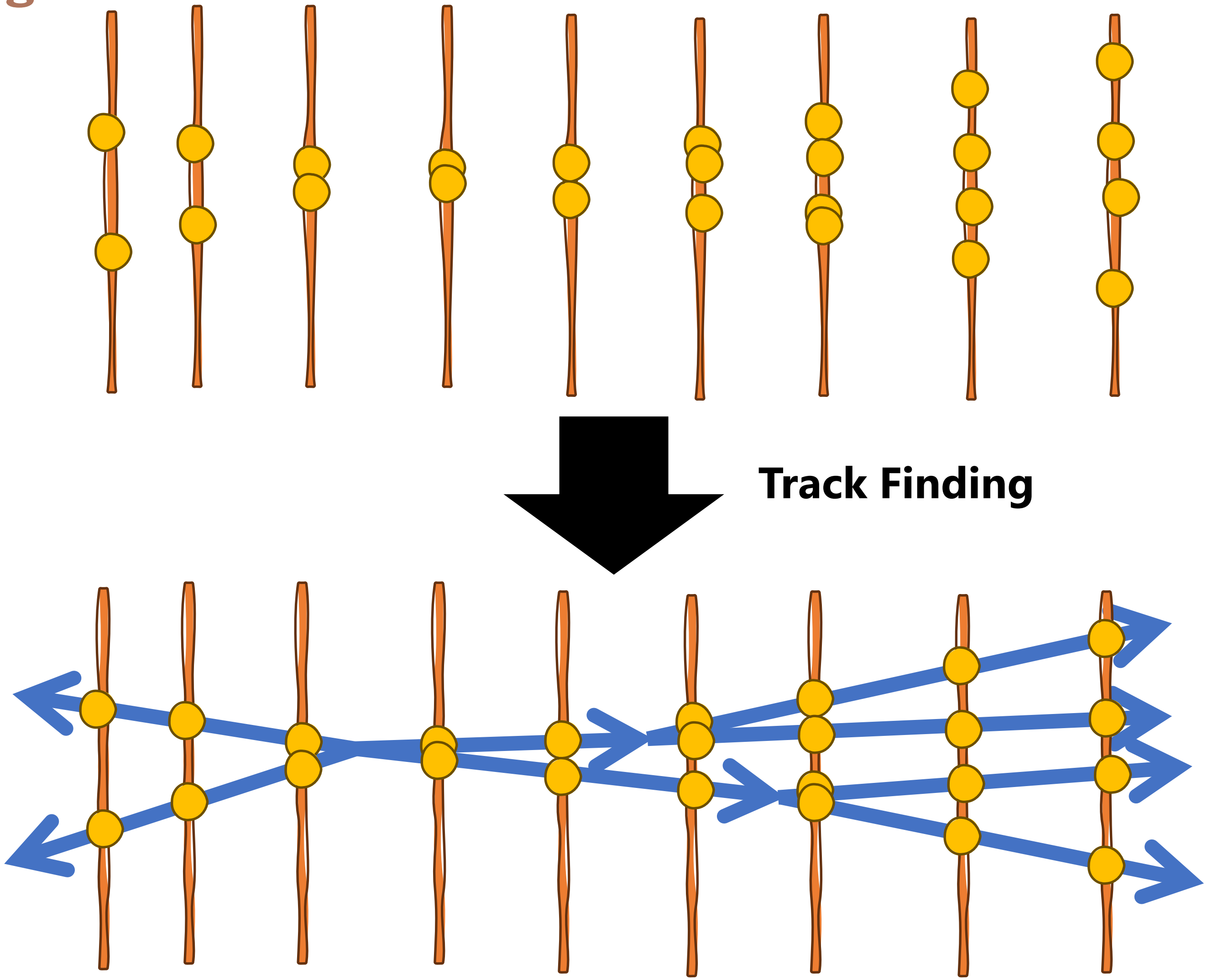
1 Beginner Introduction

C Track Finding



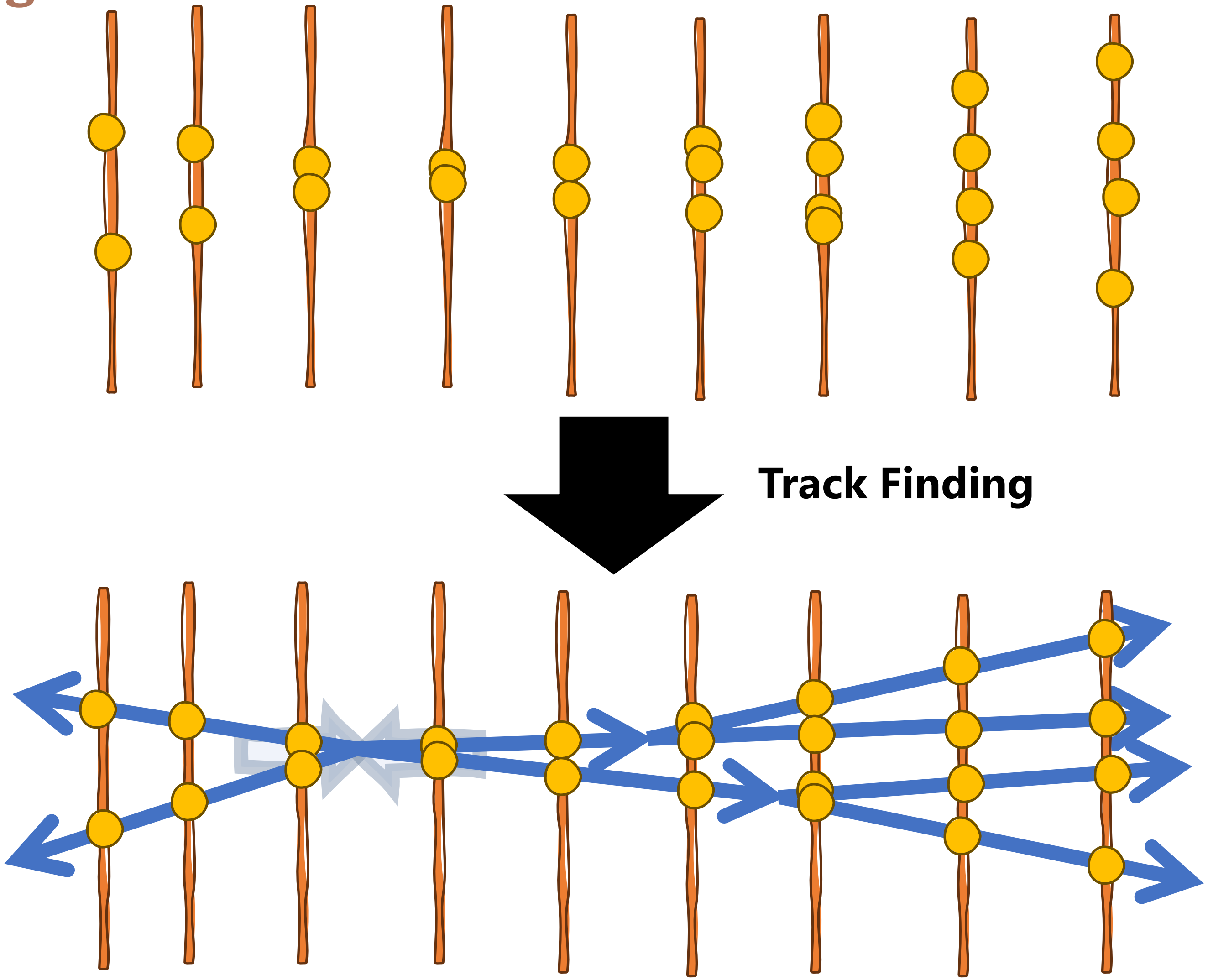
1 Beginner Introduction

C Track Finding

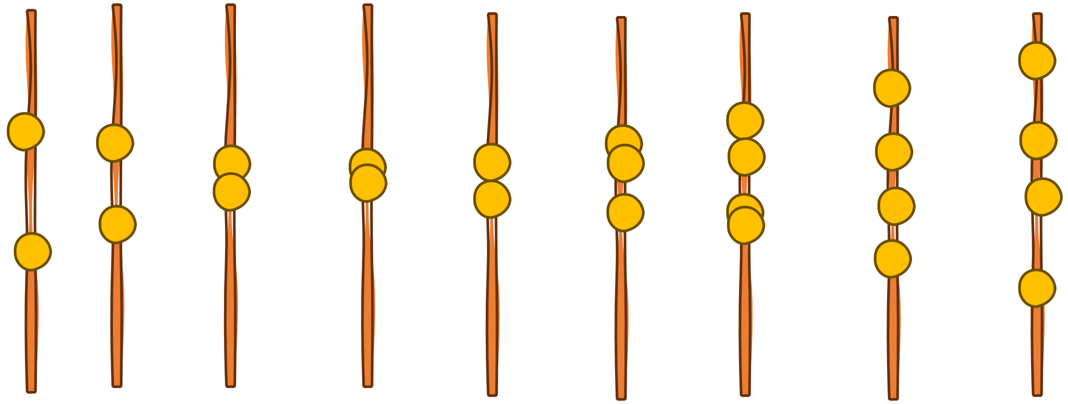


1 Beginner Introduction

c Track Finding

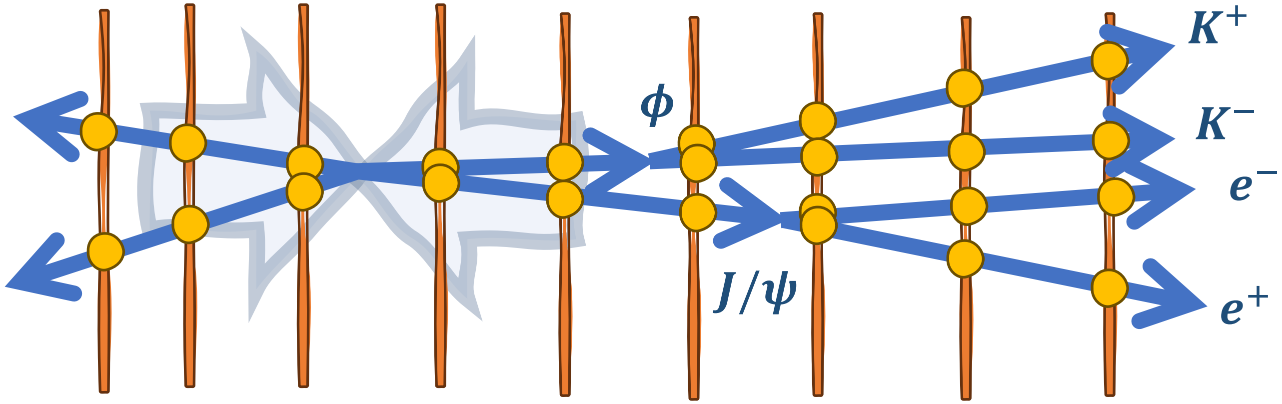


Proton-proton collision



Tracking detector

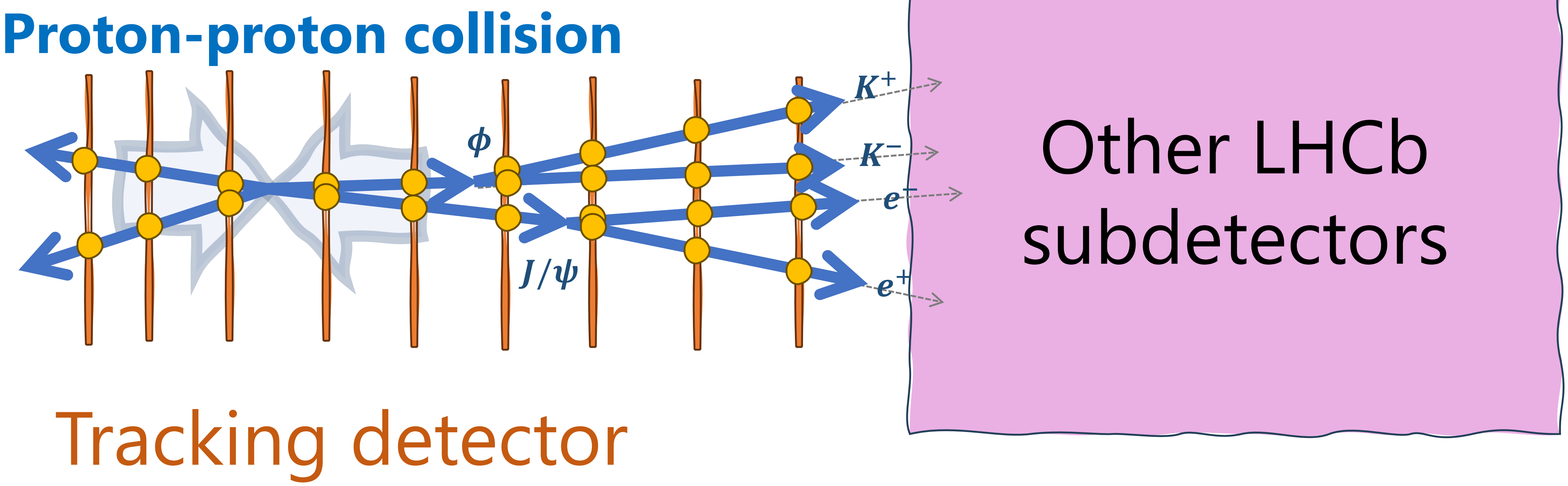
Proton-proton collision



Tracking detector

1 Beginner Introduction

d Summary



1 Beginner Introduction

2 Neural Network Introduction

3 Problem Formulation

4 Experimental Setup

5 Exa.TrkX Pipeline

6 From Exa.TrkX to ETX4VELO

7 Implementation in Allen

8 Optimization

2 Neural Network Introduction

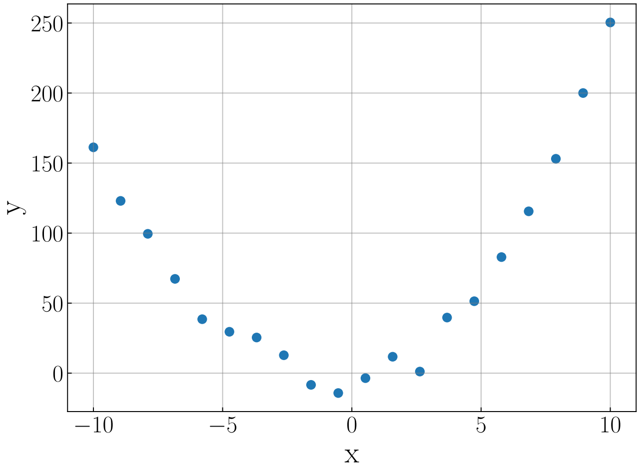
- a **Fitting a One-Dimensional Function**
- b **Multi-Layer Perceptron**
- c **Different Types of Neural Networks**

2 Neural Network Introduction

a Fitting a One-Dimensional Function

- **Regression problem**: Predict $y \in \mathbb{R}$ from $x \in \mathbb{R}$

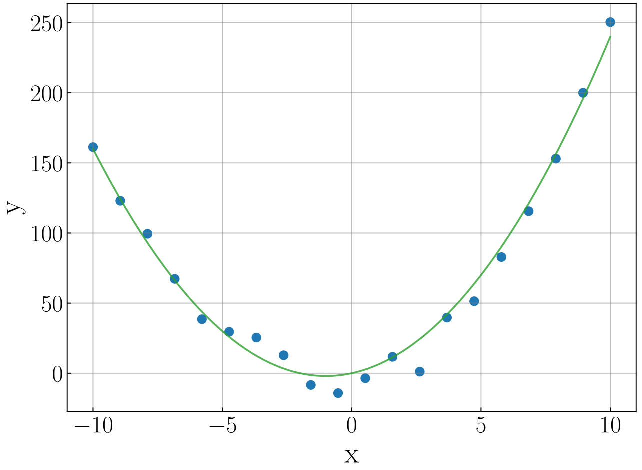
Target y



Input x

→ ? →

Predicted target $\hat{y}$



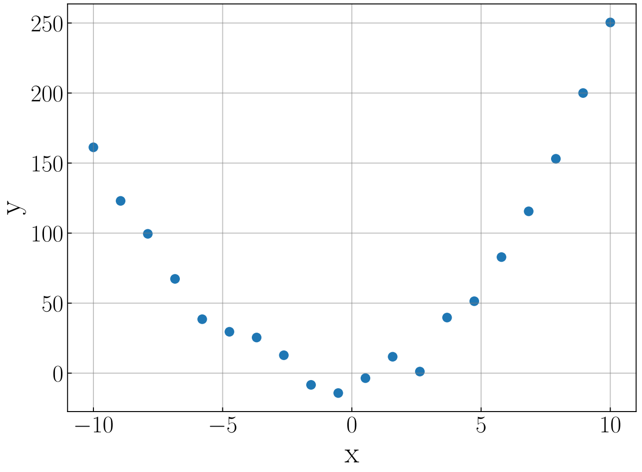
Input x

2 Neural Network Introduction

a Fitting a One-Dimensional Function

- **Regression problem**: Predict $y \in \mathbb{R}$ from $x \in \mathbb{R}$

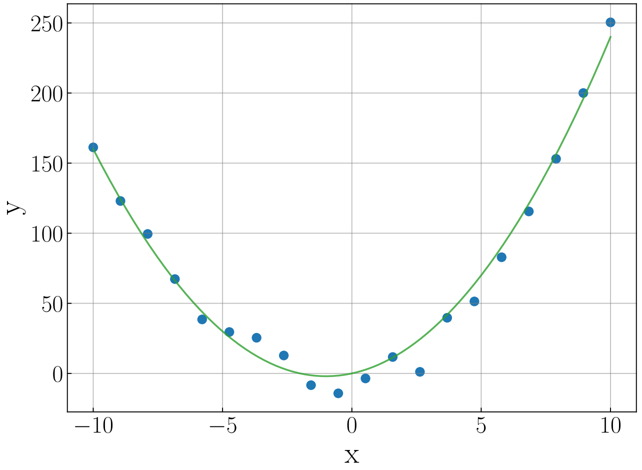
Target y



Input x

?

Predicted target $\hat{y}$



Input x

1. Model selection

Choose best model f_θ

2. Training

Find optimal parameters $\theta^\star$

3. Inference

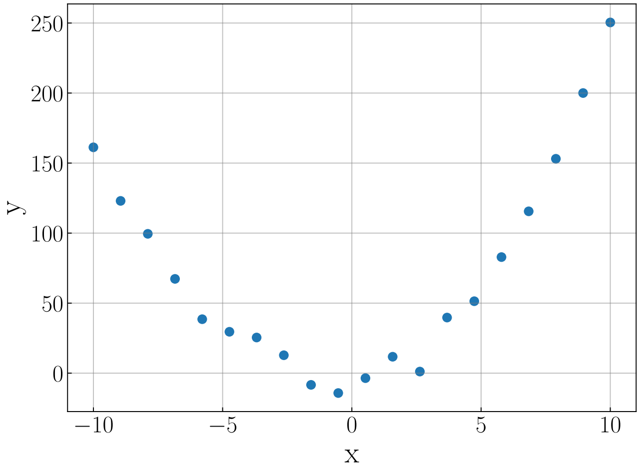
Use the model for other examples

2 Neural Network Introduction

a Fitting a One-Dimensional Function

- **Regression problem**: Predict $y \in \mathbb{R}$ from $x \in \mathbb{R}$

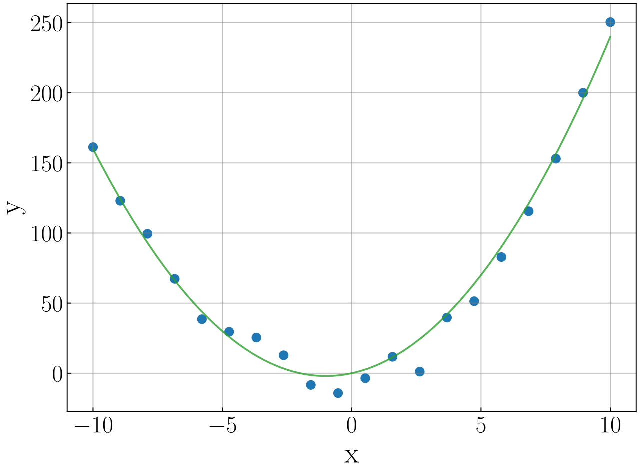
Target y



Input x

Polynomial model
3 parameters
$\hat{y} = ax^2 + bx + c$

Predicted target $\hat{y}$



Input x

1. Model selection
Choose best model f_θ

2. Training
Find optimal parameters $\theta^\star$

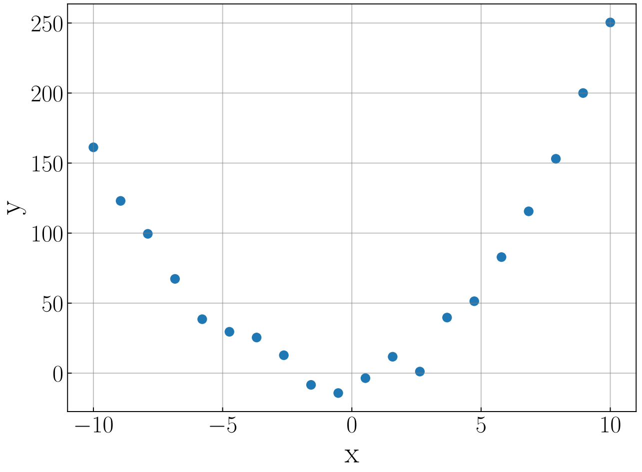
3. Inference
Use the model for other examples

2 Neural Network Introduction

a Fitting a One-Dimensional Function

- **Regression problem**: Predict $y \in \mathbb{R}$ from $x \in \mathbb{R}$

Target y

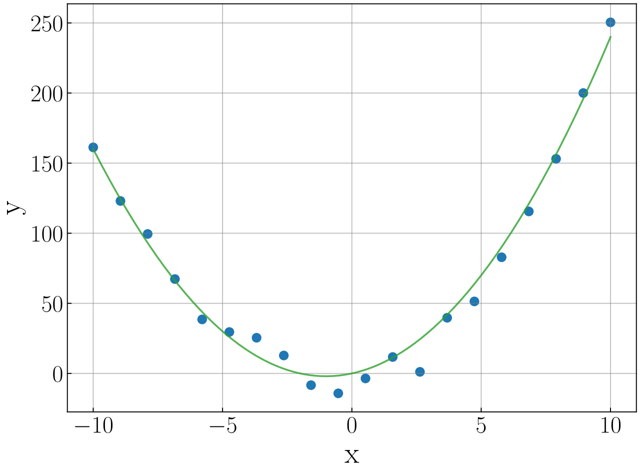


Input x

Polynomial model
3 parameters
$\hat{y} = ax^2 + bx + c$

$$\boldsymbol{\theta}^* = \begin{pmatrix} \boldsymbol{a} \\ \boldsymbol{b} \\ \boldsymbol{c} \end{pmatrix} = \begin{pmatrix} \mathbf{2} \\ \mathbf{4} \\ \mathbf{0} \end{pmatrix}$$

Predicted target $\hat{y}$



Input x

1. Model selection
Choose best model f_θ

2. Training
Find optimal parameters $\theta^\star$

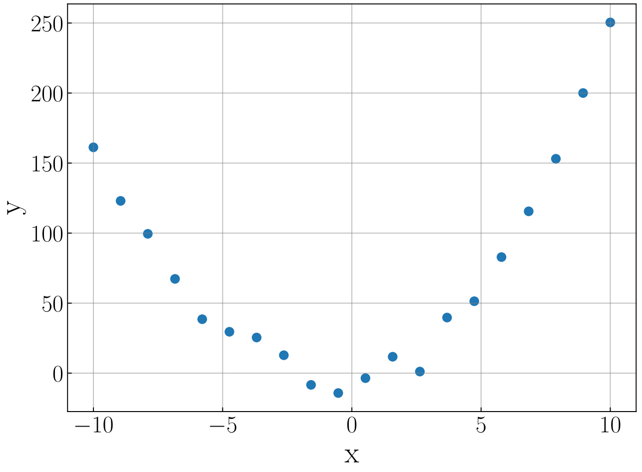
3. Inference
Use the model for other examples

2 Neural Network Introduction

a Fitting a One-Dimensional Function

- **Regression problem**: Predict $y \in \mathbb{R}$ from $x \in \mathbb{R}$

Target y

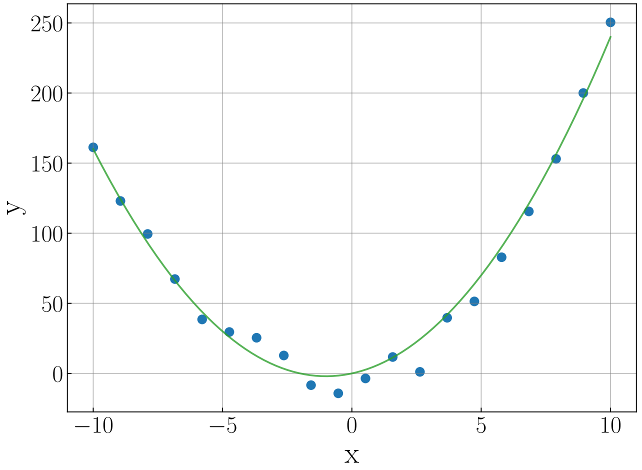


Input x

Polynomial model
3 parameters
$\hat{y} = ax^2 + bx + c$

$$\boldsymbol{\theta}^* = \begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} 2 \\ 4 \\ 0 \end{pmatrix}$$

Predicted target $\hat{y}$



Input x

1. Model selection
Choose best model f_θ

2. Training
Find optimal parameters θ^*

3. Inference
Use the model for other examples

What if multi-dimensional data?

2 Neural Network Introduction

b Multi-Layer Perceptron

- **Binary classification problem**: predict picture → is this a cat
  - **Input**: vectors made of the pixel values of the picture
  - **Output**: Probability of being a cat, between 0 (no) and 1 (yes)



→ ? → 0.4 0.9 0.2

2 Neural Network Introduction

b Multi-Layer Perceptron

- **Binary classification problem**: predict picture → is this a cat
  - **Input**: vectors made of the pixel values of the picture
  - **Output**: Probability of being a cat, between 0 (no) and 1 (yes)



→

Multi-Layer Perceptron (MLP)
Many parameters
$\hat{\boldsymbol{y}} = \mathbf{Model}(\boldsymbol{x})$

→

0.4 0.9 0.2

1. **Model Choice**: **Multi-Layer Perceptron** (**MLP**) *[many parameters]*

2 Neural Network Introduction

b Multi-Layer Perceptron

- **Binary classification problem**: predict picture → is this a cat
  - **Input**: vectors made of the pixel values of the picture
  - **Output**: Probability of being a cat, between 0 (no) and 1 (yes)

Multi-Layer Perceptron (MLP)
Many parameters
$\hat{y} = \mathbf{Model}(\boldsymbol{x})$

0.4 0.9 0.2

1. **Model Choice**: **Multi-Layer Perceptron** (**MLP**) *[many parameters]*

2. **Training:** Needs for:
   - Much more **data**
   - Much more **computing resource**

2 Neural Network Introduction

b Multi-Layer Perceptron

- **Binary classification problem**: predict picture → is this a cat
  - **Input**: vectors made of the pixel values of the picture
  - **Output**: Probability of being a cat, between 0 (no) and 1 (yes)



Multi-Layer Perceptron (MLP)
Many parameters
$\hat{y} = \text{Model}(x)$

0.4 0.9 0.2

$s > s_{\min} = 0.5$

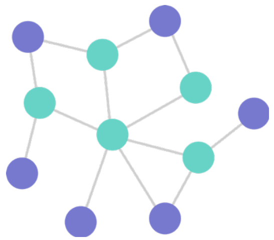
1. **Model Choice**: **Multi-Layer Perceptron** (**MLP**) *[many parameters]*

2. **Training:** Needs for:
   - Much more **data**
   - Much more **computing resource**

3. **Inference:** Apply a **minimum score threshold**: $s > s_{\min} = 0.5$

2 Neural Network Introduction

C Different Types of Neural Networks

Input		Neural Network Model Family
Pictures	→	Convolutional Neural Network (CNN)
Sequences: text, audio, …	→	Recursive Neural Network (RNN)
Graph 	→	Graph Neural Network (GNN)

- **Deep Learning Library**: PyTorch

1 Beginner Introduction

2 Neural Network Introduction

3 Problem Formulation

4 Experimental Setup

5 Exa.TrkX Pipeline

6 From Exa.TrkX to ETX4VELO

7 Implementation in Allen

8 Optimization

3 Problem Formulation

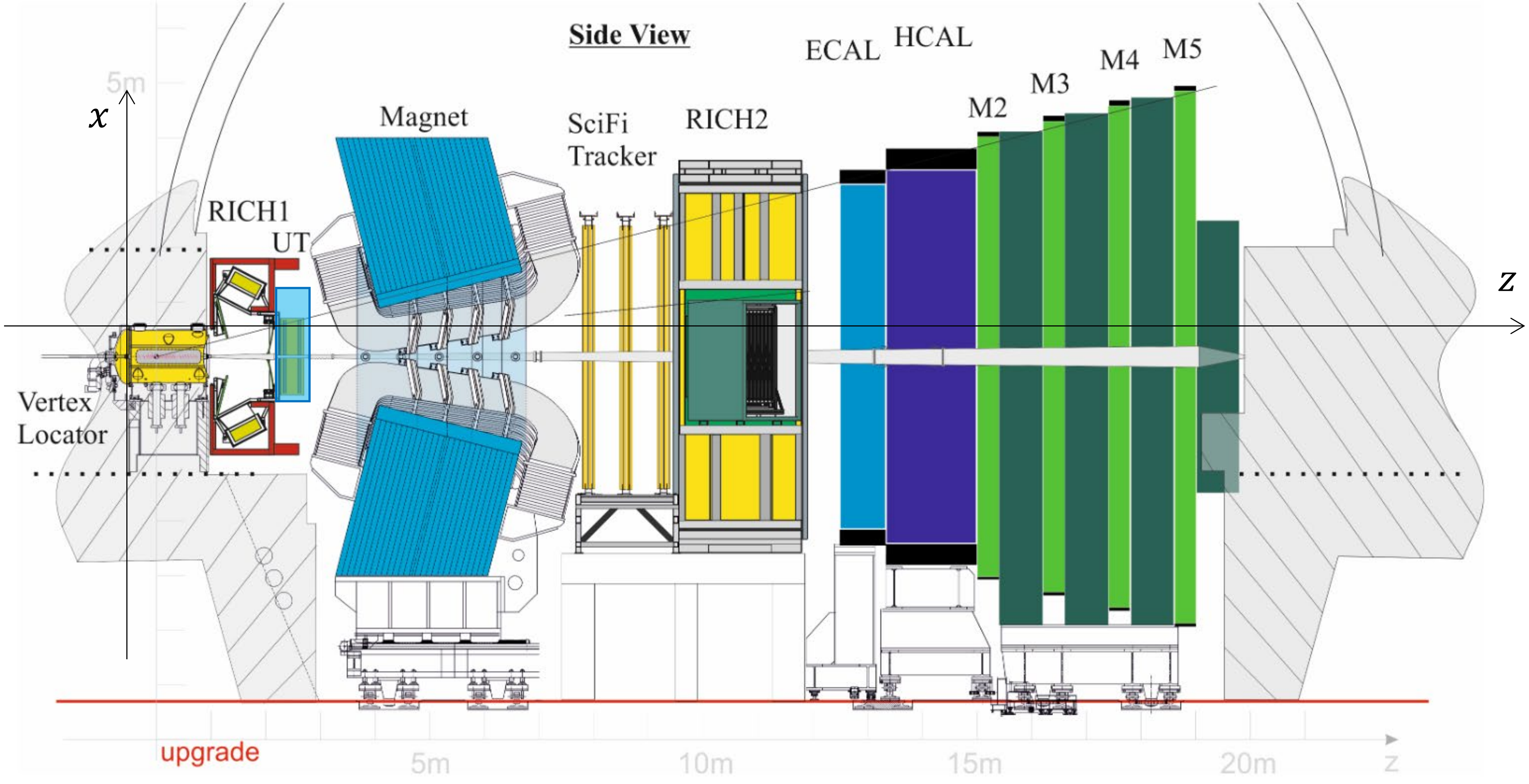
a Track Finding in the Velo

b Allen: a Fully GPU-Based Trigger

c Motivations

3 Problem Formulation

a Track Finding in the Velo



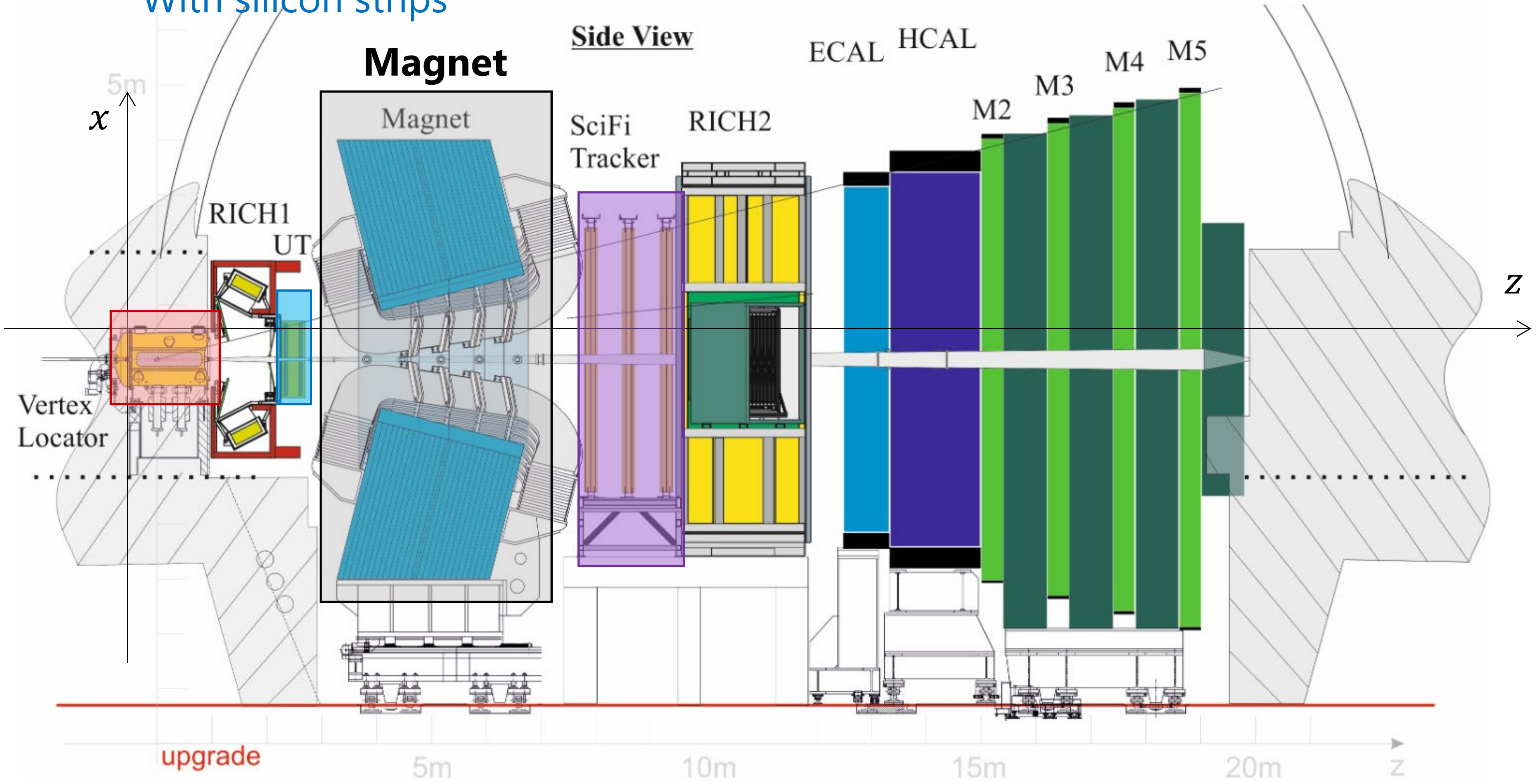
3 Problem Formulation

a Track Finding in the Velo

Velo
Vertex **Lo**cator
With silicon pixels
No magnetic field

UT
Upstream **T**racker
With silicon strips

SciFi
With **Sci**ntillating **Fi**bres



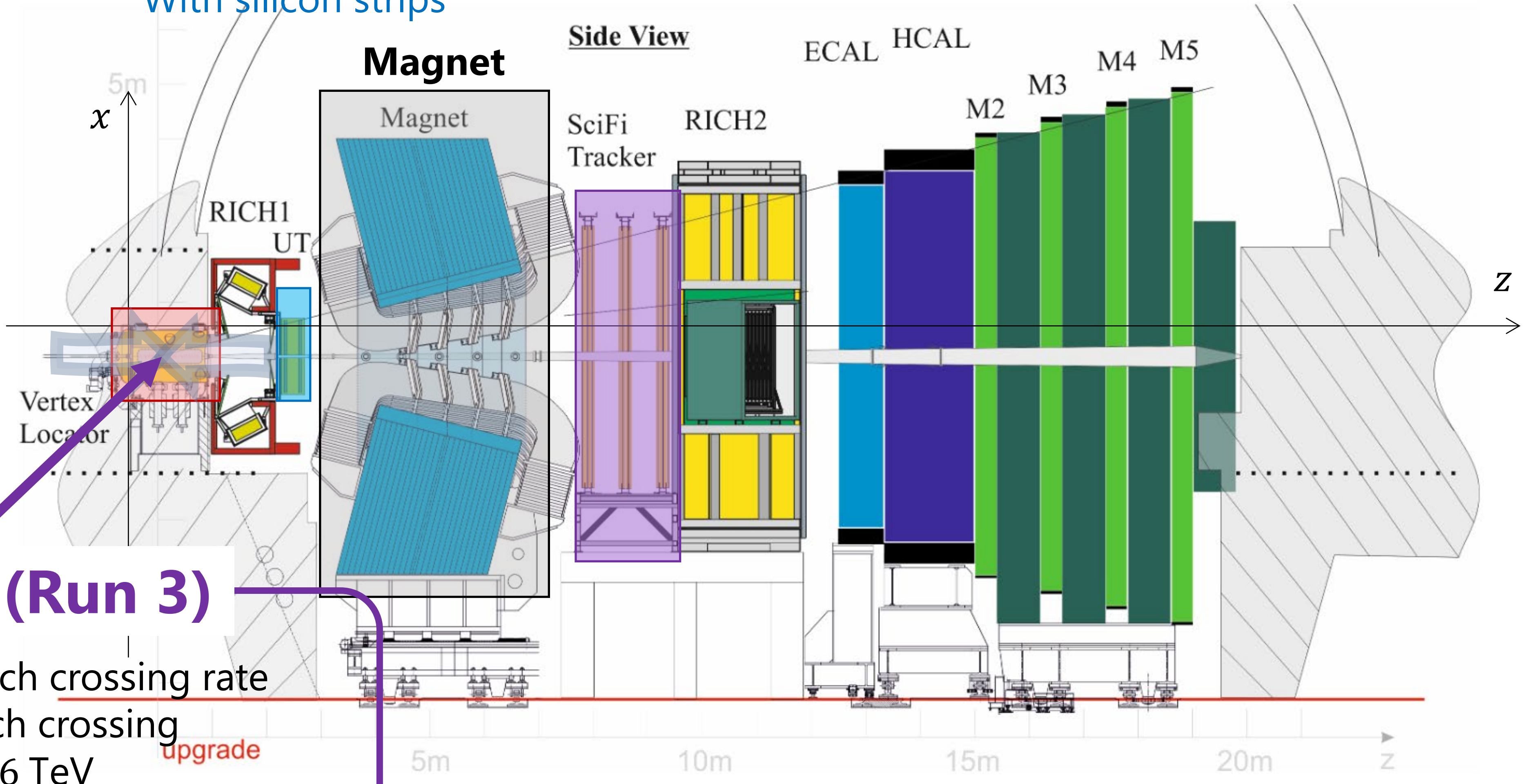
3 Problem Formulation

a Track Finding in the Velo

Velo
Vertex **Lo**cator
With silicon pixels
No magnetic field

UT
Upstream **T**racker
With silicon strips

SciFi
With **Sci**ntillating **Fi**bres



Collisions (Run 3)

- 25 MHz non-empty bunch crossing rate
- ~ 5 p-p collisions / bunch crossing
- p-p collision at $\sqrt{s} = 13.6$ TeV

3 Problem Formulation

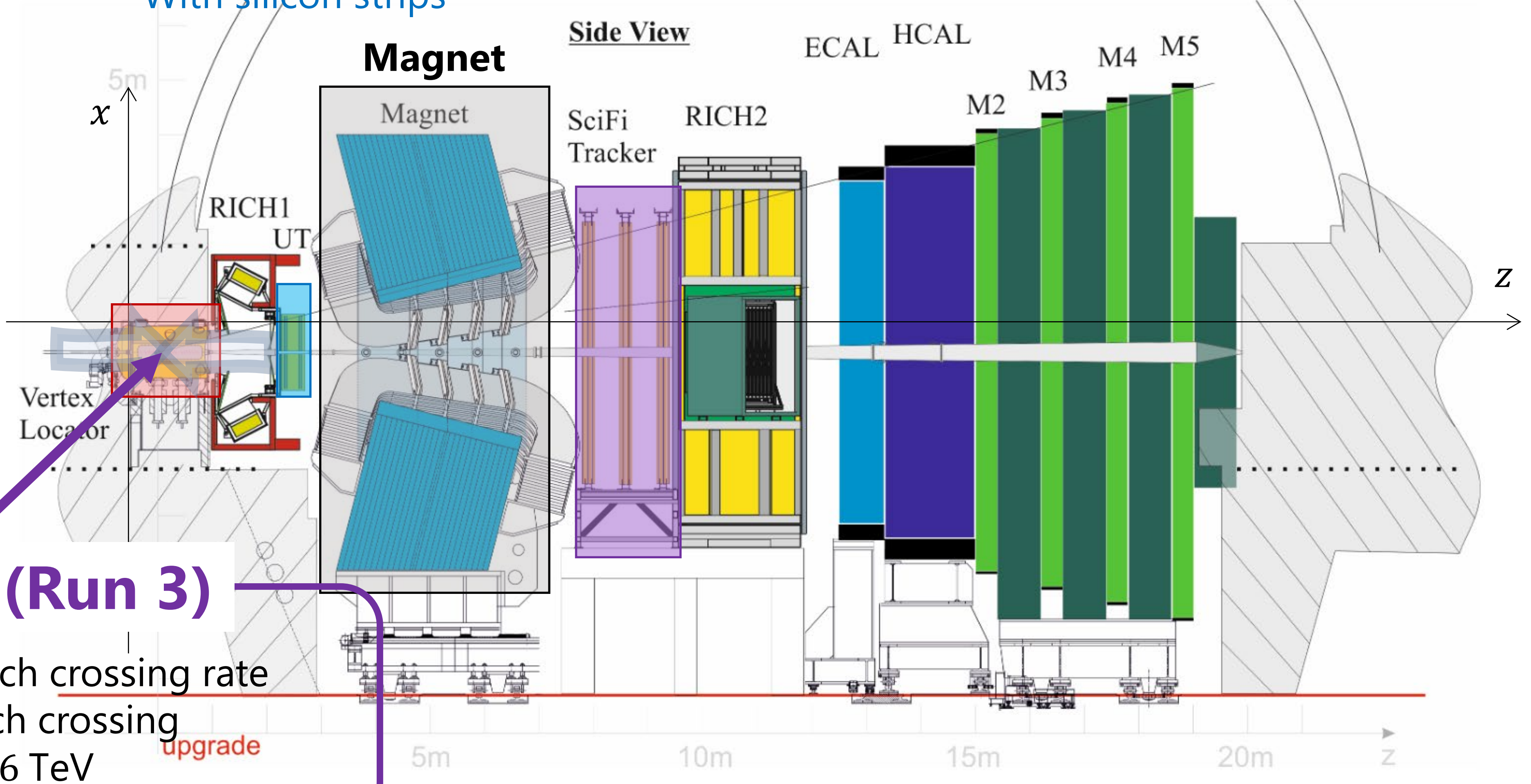
a Track Finding in the Velo

Velo
Vertex **Lo**cator
With silicon pixels
No magnetic field

UT
Upstream **T**racker
With silicon strips

SciFi
With **Sci**ntillating **Fi**bres

- Primary vertices
- Secondary Vertices
- Seeds

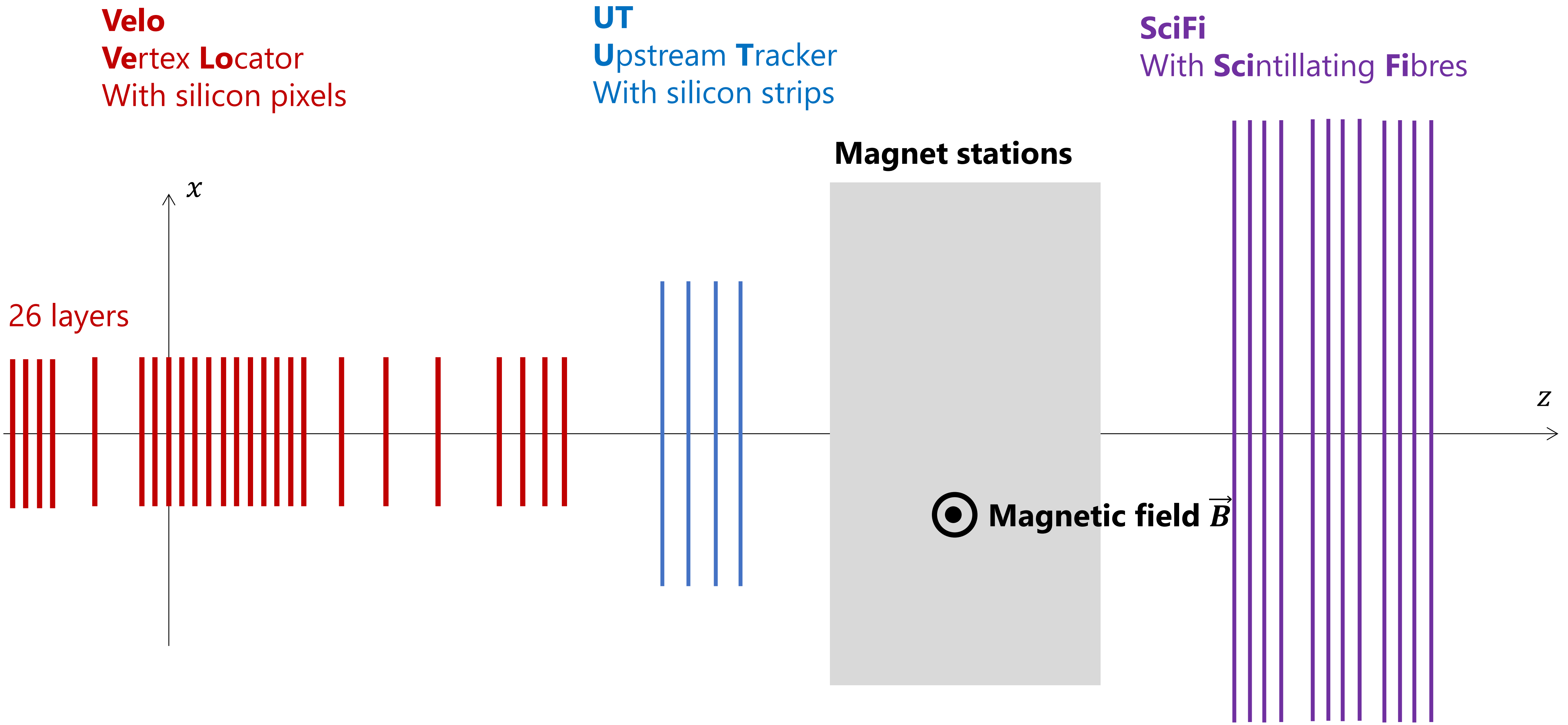


Collisions (Run 3)

- 25 MHz non-empty bunch crossing rate
- ~ 5 p-p collisions / bunch crossing
- p-p collision at $\sqrt{s} = 13.6$ TeV

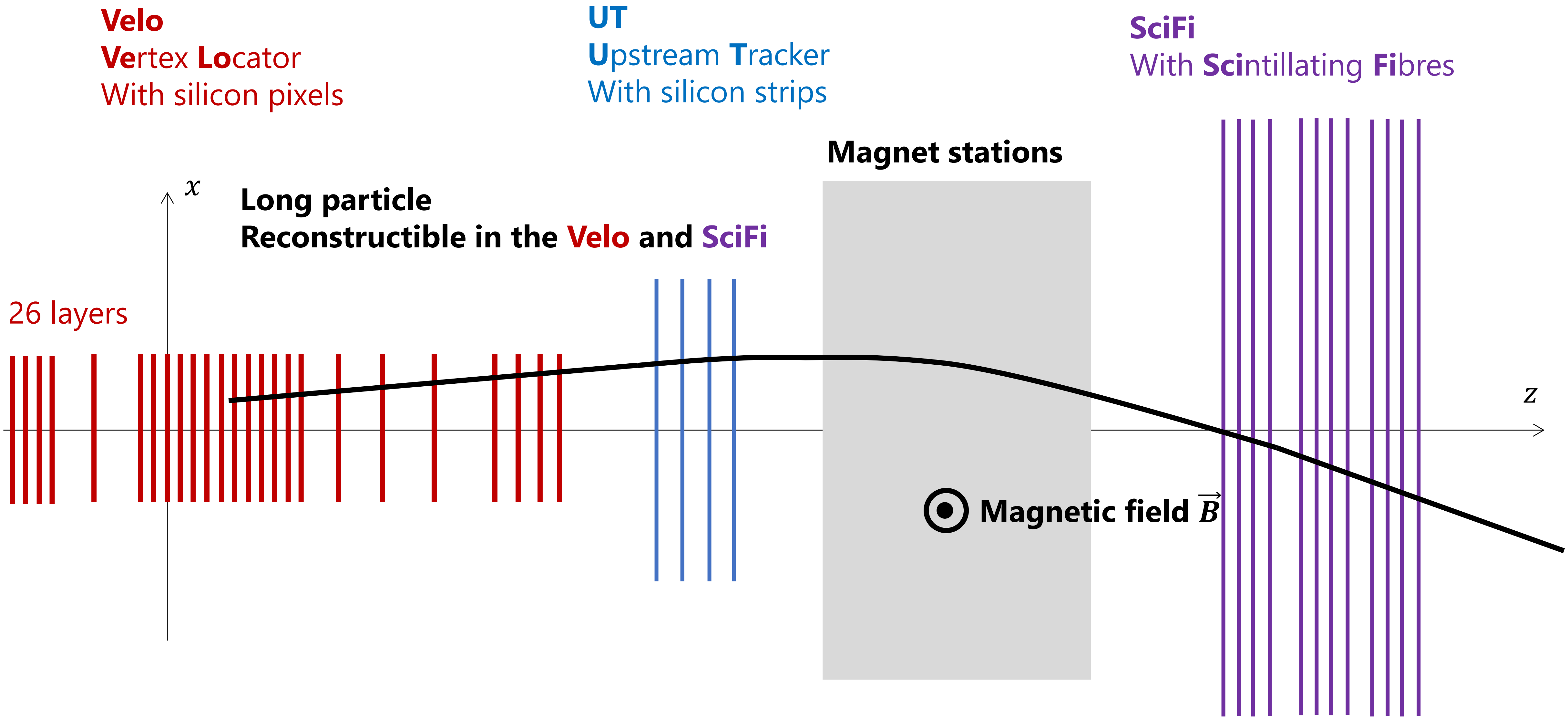
3 Problem Formulation

a Track Finding in the Velo



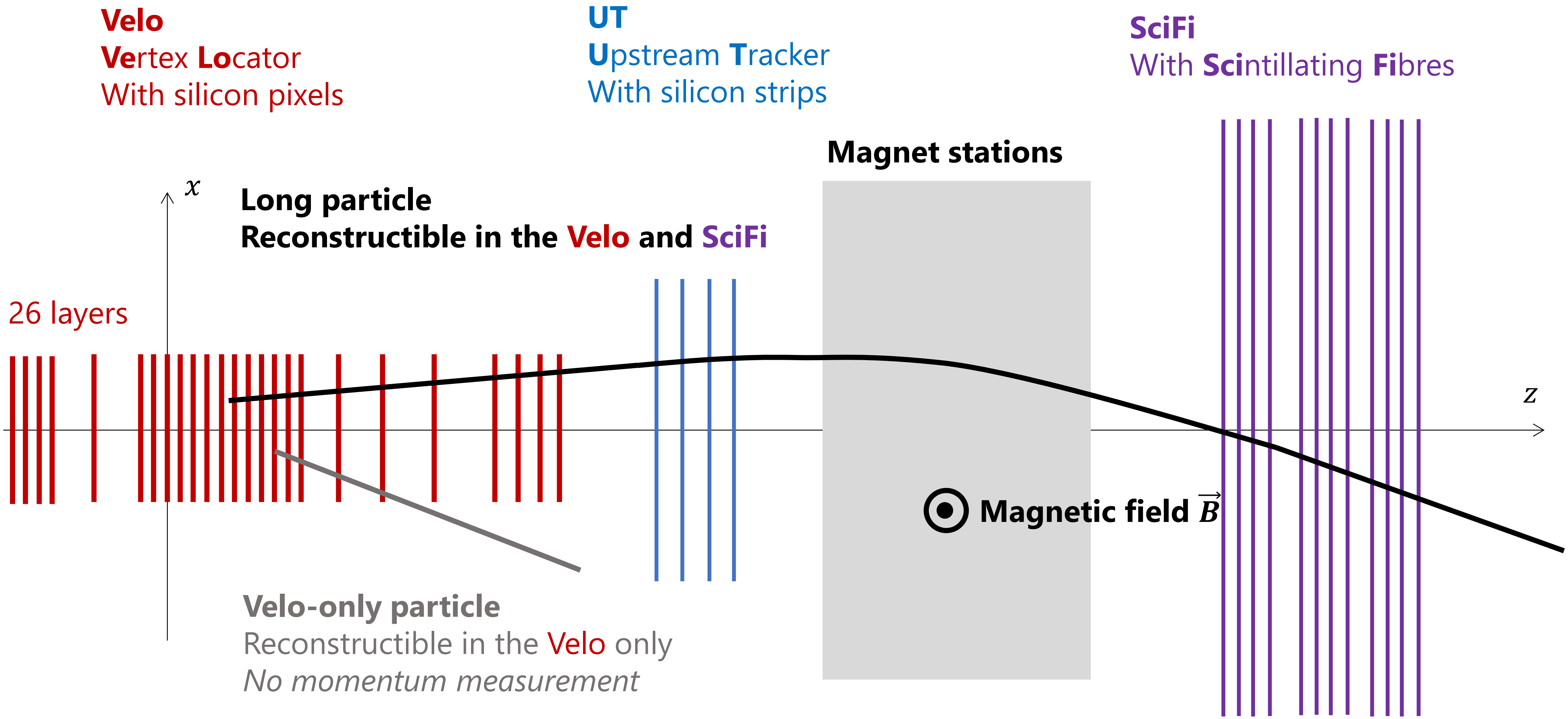
3 Problem Formulation

a Track Finding in the Velo



3 Problem Formulation

a Track Finding in the Velo



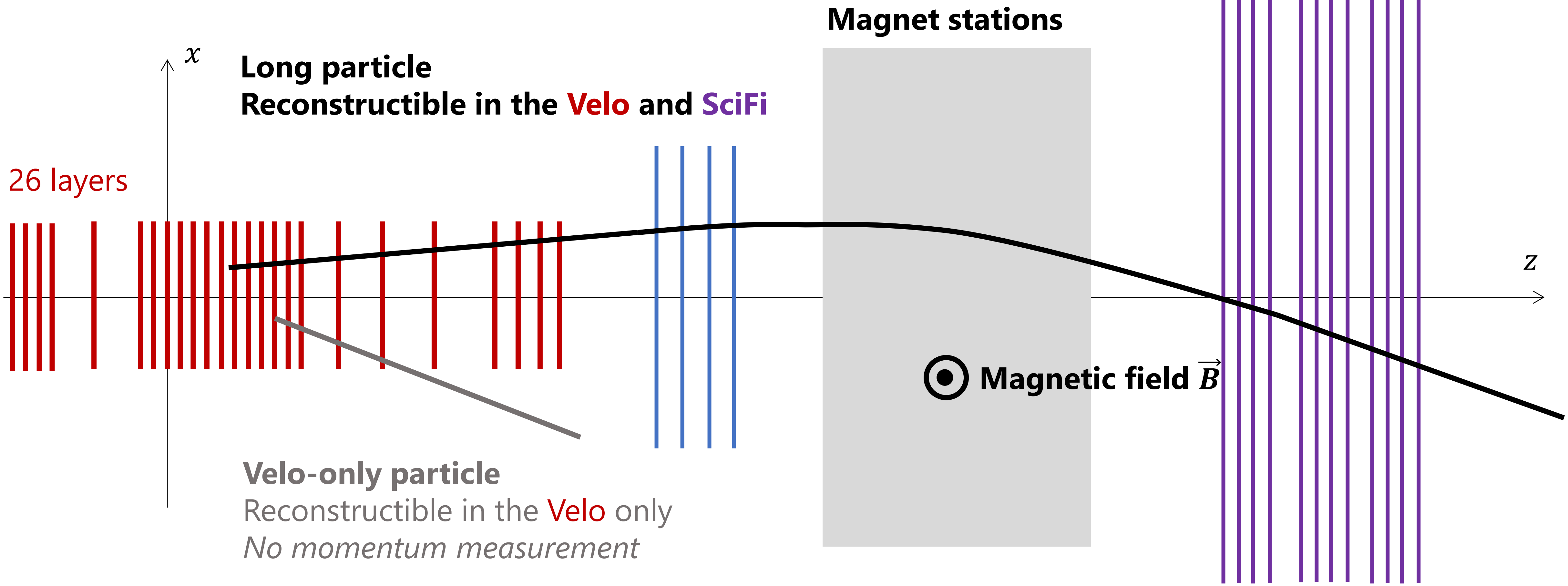
3 Problem Formulation

a Track Finding in the Velo

Velo
Vertex **Lo**cator
With silicon pixels

UT
Upstream **T**racker
With silicon strips

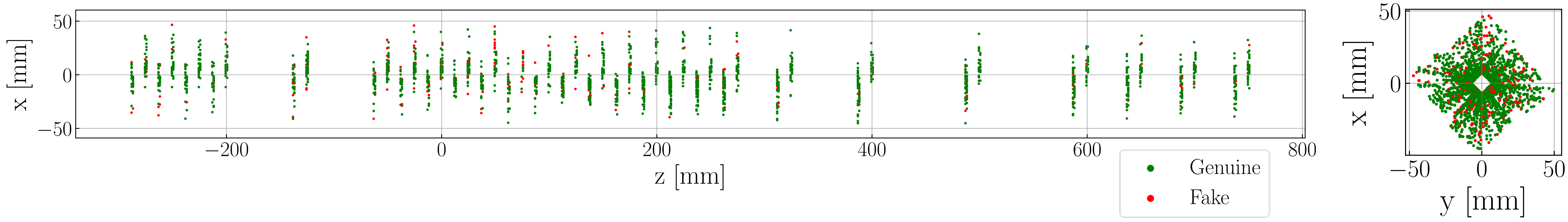
SciFi
With **Sci**ntillating **Fi**bres



3 Problem Formulation

a Track Finding in the Velo

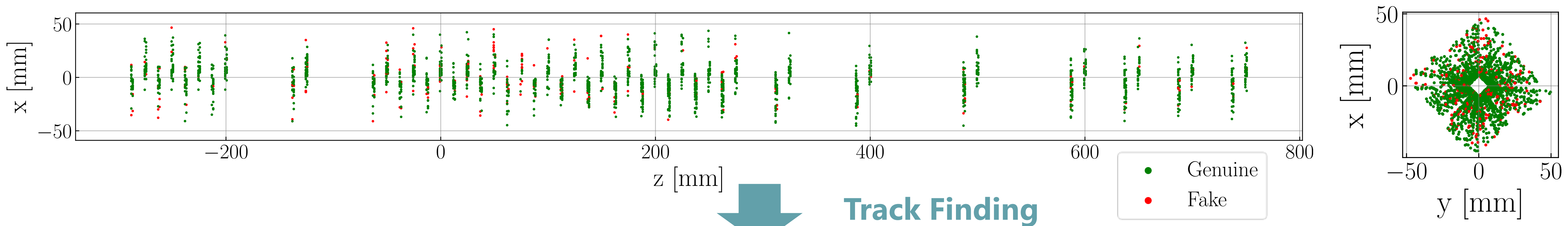
- 2000 hits
- 13% fake hits



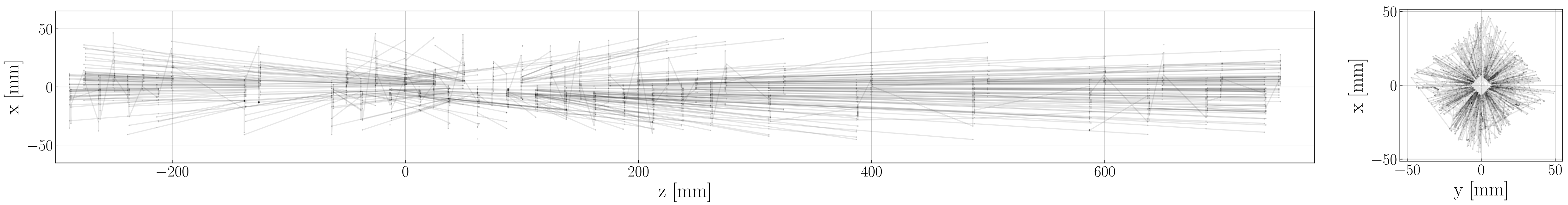
3 Problem Formulation

a Track Finding in the Velo

- 2000 hits
- 13% fake hits



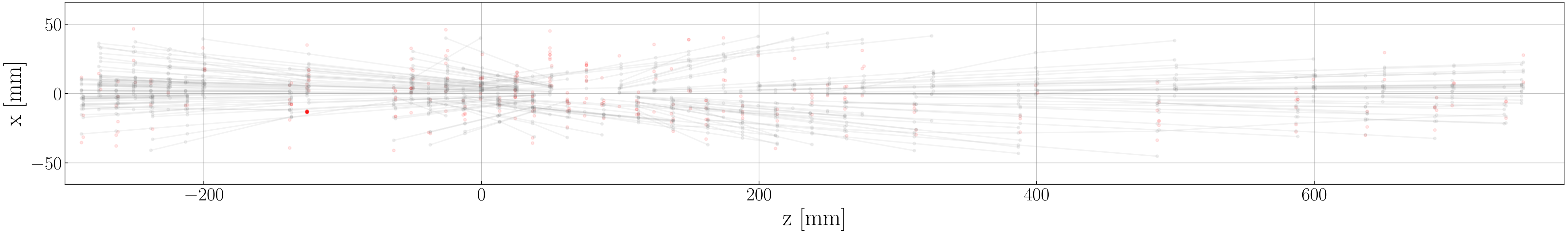
Track Finding

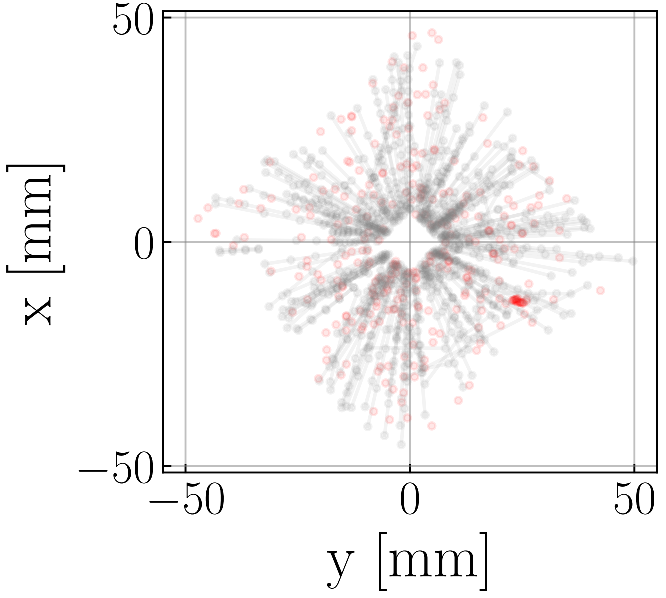


- **Velo-reconstructible**: at least 3 hits
- ~ 235 Velo particles

3 Problem Formulation

a Track Finding in the Velo

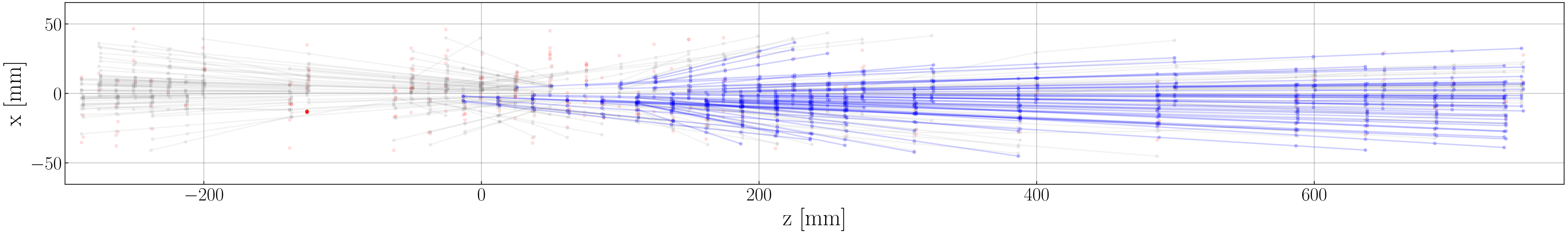


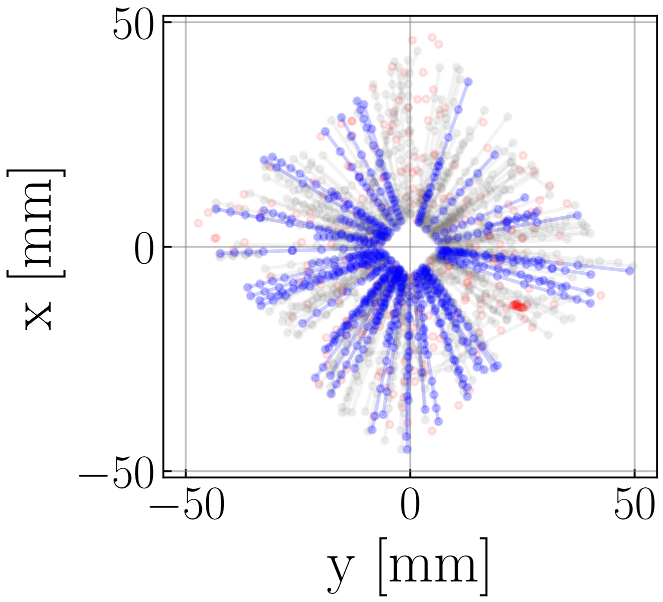


Category	Proportion	# particles / event
Velo	100%	235
Velo-only	**73.50%**	**172**

3 Problem Formulation

a Track Finding in the Velo

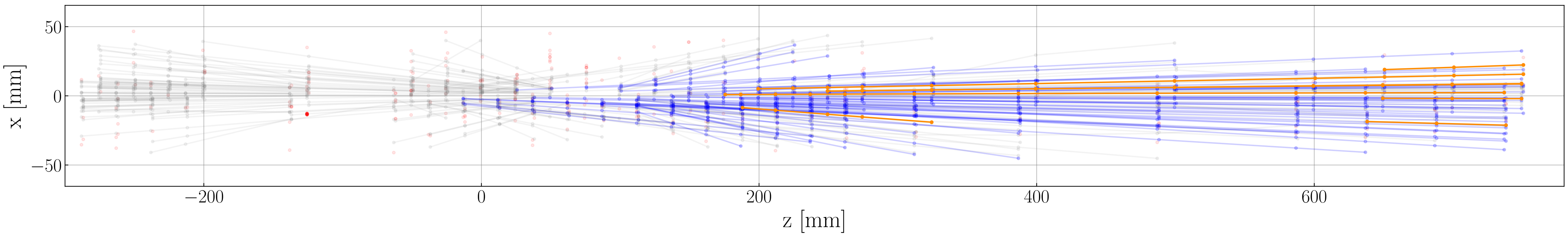


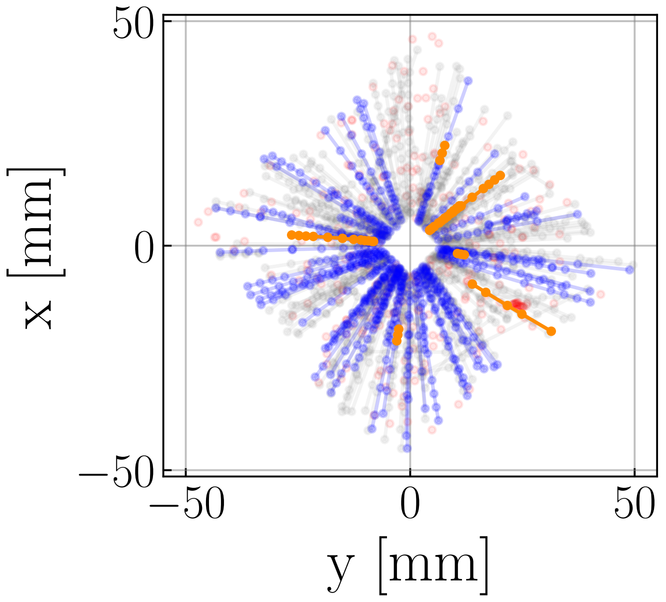


Category	Proportion	# particles / event
Velo	100%	235
Velo-only	73.50%	172
Long no electrons	**24.74%**	**58**

3 Problem Formulation

a Track Finding in the Velo

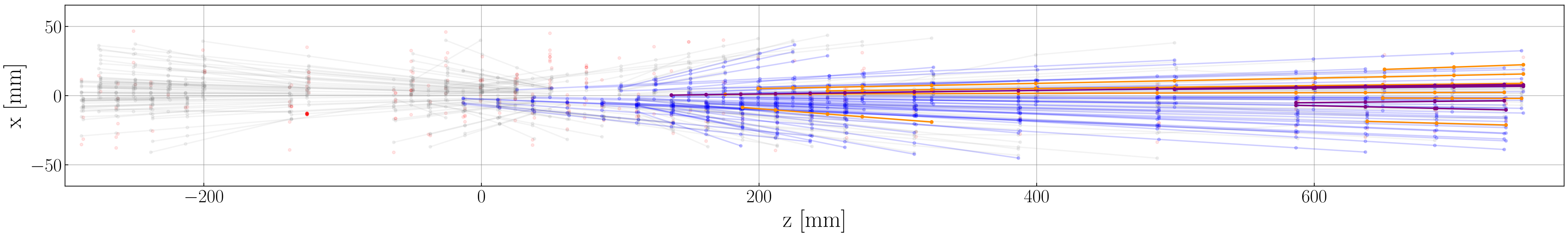


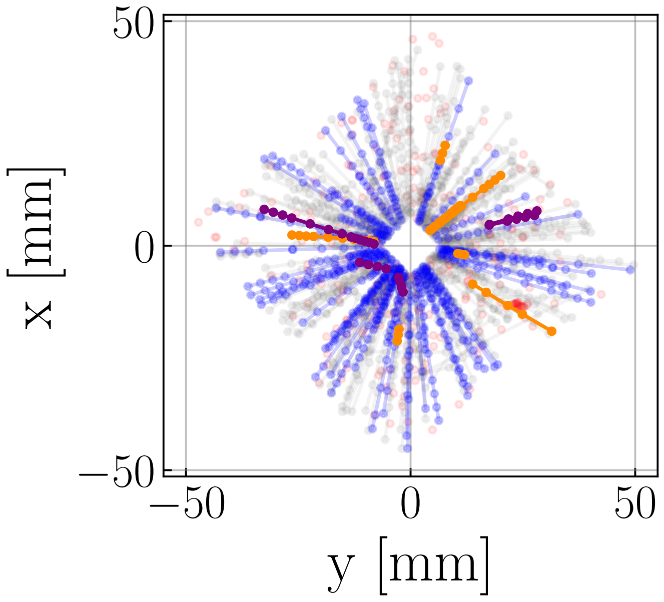


Category	Proportion	# particles / event
Velo	100%	235
Velo-only	73.50%	172
Long no electrons	**24.74%**	**58**
Long electrons	**1.58%**	**4**

3 Problem Formulation

a Track Finding in the Velo





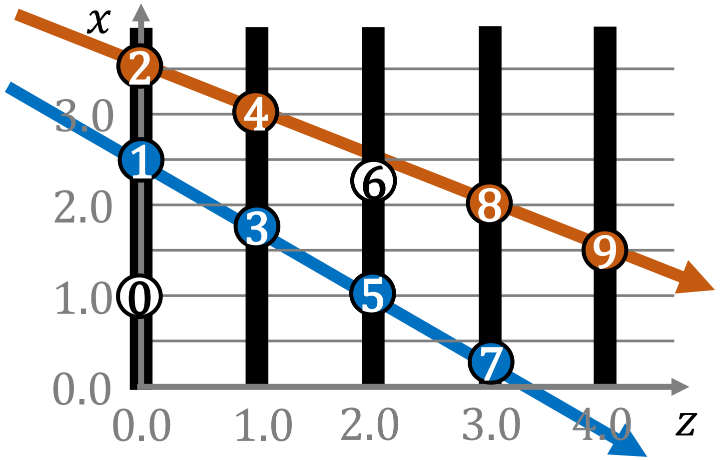
Category	Proportion	# particles / event
Velo	100%	235
Velo-only	73.50%	172
Long no electrons	**24.74%**	**58**
Long electrons	**1.58%**	**4**
Long from strange	**1.11%**	**3**

3 Problem Formulation

a Track Finding in the Velo

Particle Trajectories

- **Straight lines** (magnetic field negligible)
- **Skipped layers**: 5% of Velo-reconstructible particles miss at least 1 layer

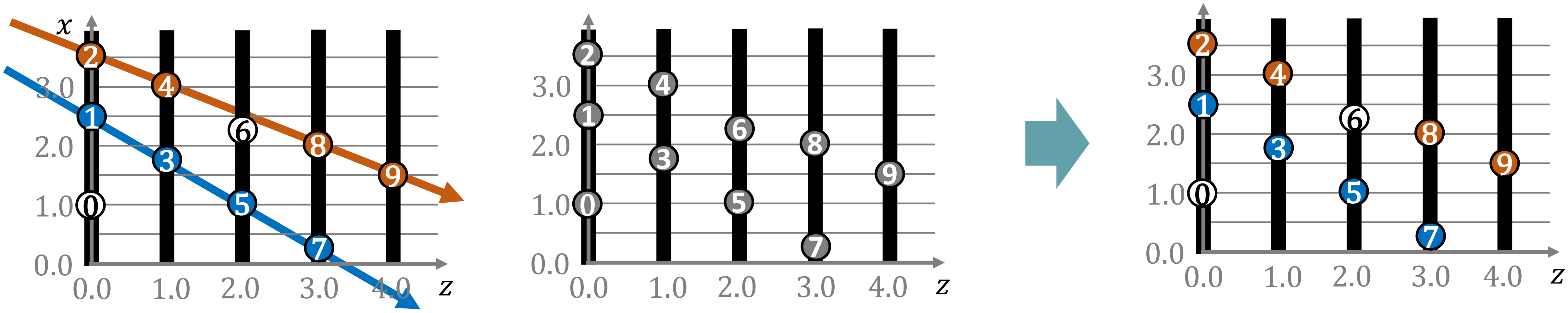


3 Problem Formulation

a Track Finding in the Velo

Particle Trajectories

- **Straight lines** (magnetic field negligible)
- **Skipped layers**: 5% of Velo-reconstructible particles miss at least 1 layer



3 Problem Formulation

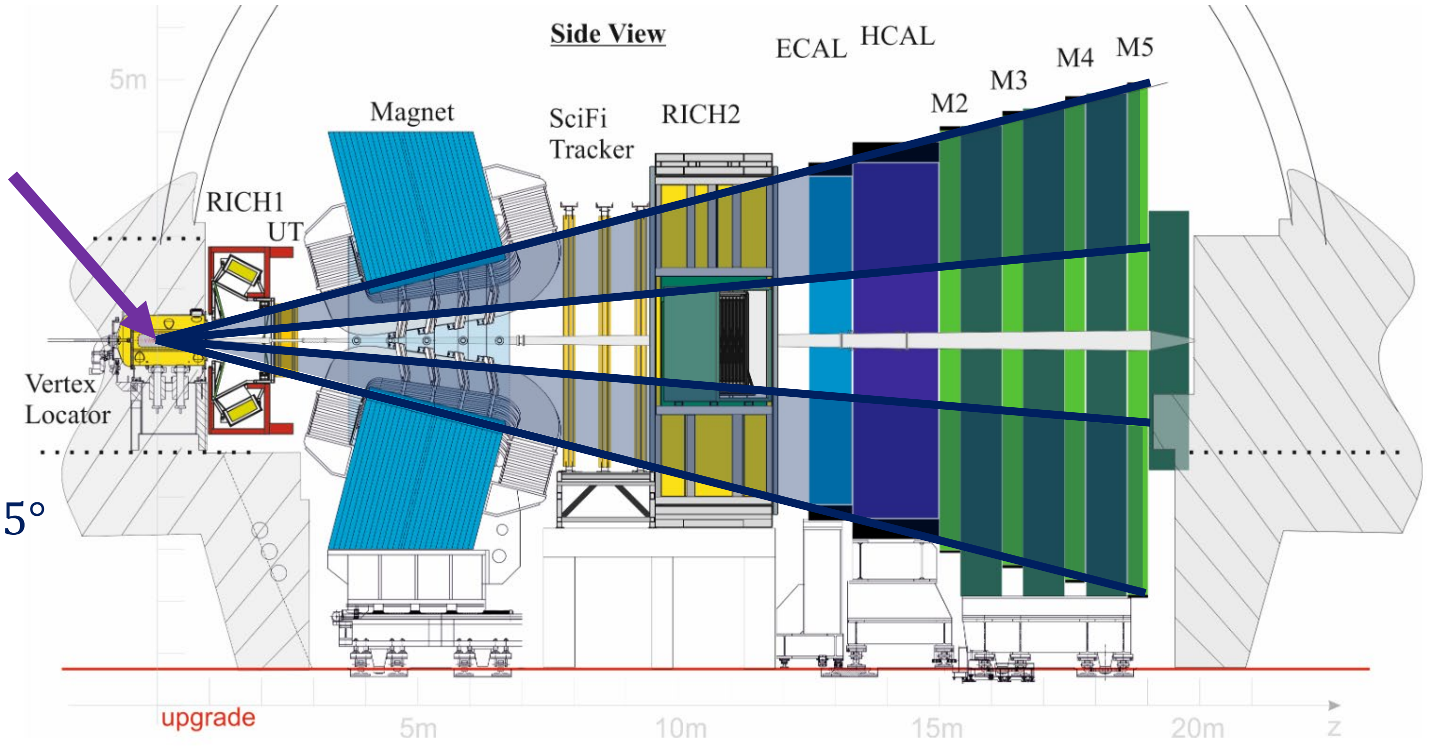
b Allen: a Fully GPU-based trigger

Collisions (Run 3)

- 25 MHz non-empty bunch crossing rate
- ~ 5 p-p collisions / bunch crossing
- p-p collision at $\sqrt{s} = 13.6$ TeV

LHCb Subdetectors

Acceptance
$2 < \eta < 5$
$1° < \theta < 15°$



3 Problem Formulation

b Allen: a Fully GPU-based trigger

Collisions (Run 3)

- 25 MHz non-empty bunch crossing rate
- ~ 5 p-p collisions / bunch crossing
- p-p collision at $\sqrt{s} = 13.6$ TeV

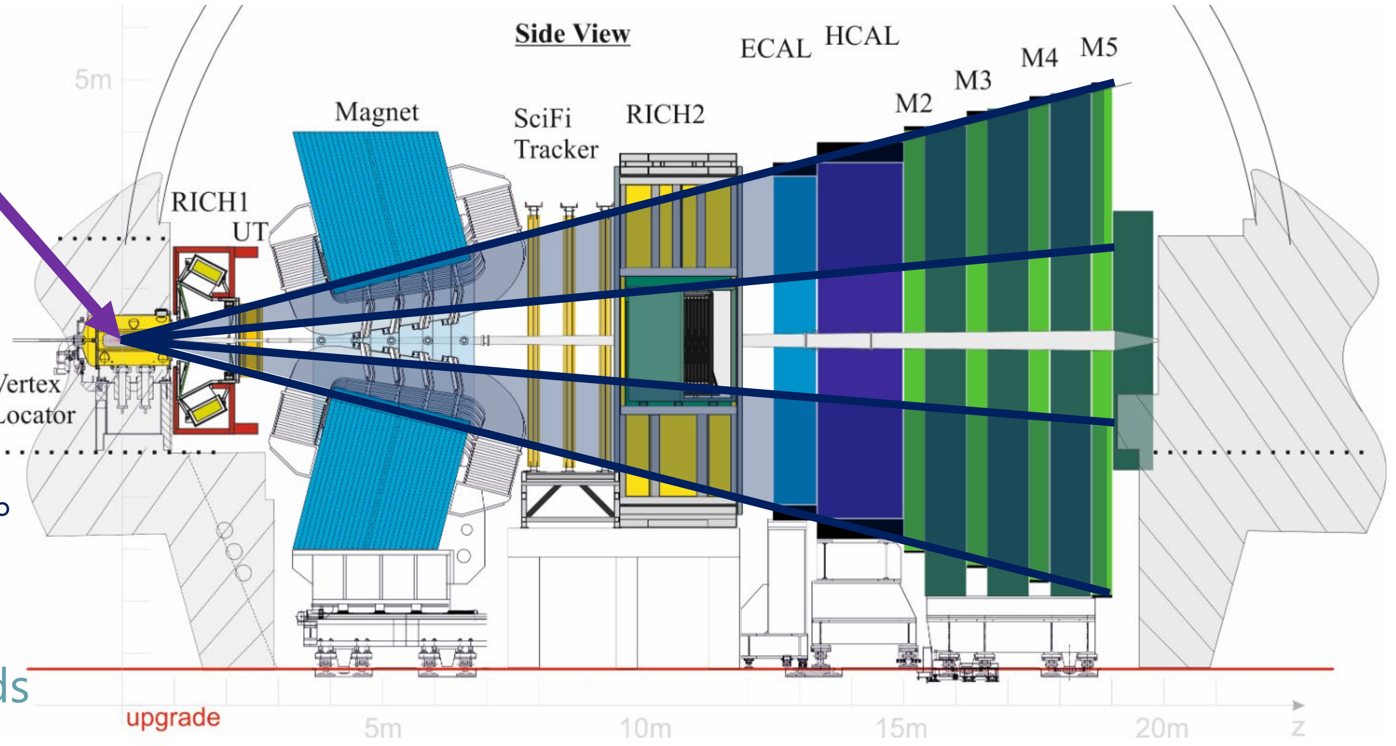
LHCb Subdetectors

Acceptance
$2 < \eta < 5$
$1° < \theta < 15°$

Digitization

PCIe40 boards

5 TB/s

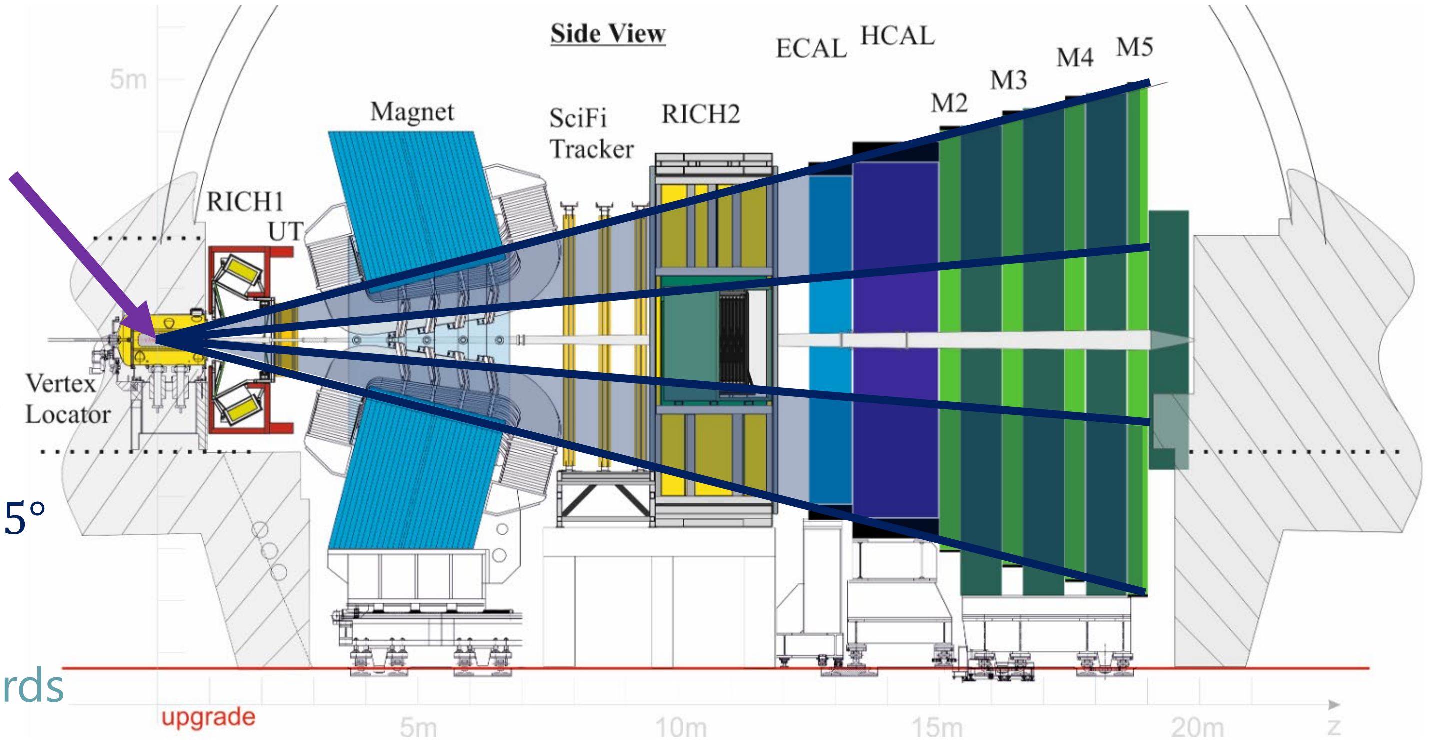


3 Problem Formulation

b Allen: a Fully GPU-based trigger

Collisions (Run 3)

- 25 MHz non-empty bunch crossing rate
- ~ 5 p-p collisions / bunch crossing
- p-p collision at $\sqrt{s} = 13.6$ TeV



LHCb Subdetectors

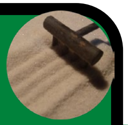
Acceptance
$2 < \eta < 5$
$1° < \theta < 15°$

Digitization

PCIe40 boards

5 TB/s

Allen [HLT1]
$\mathcal{O}(500)$ GPUs
Partial reconstruction
Partial Selection



70-200 GB/s

Storage buffer

HLT2
$\mathcal{O}(3000)$ CPU x86 servers
Full reconstruction
Full selection

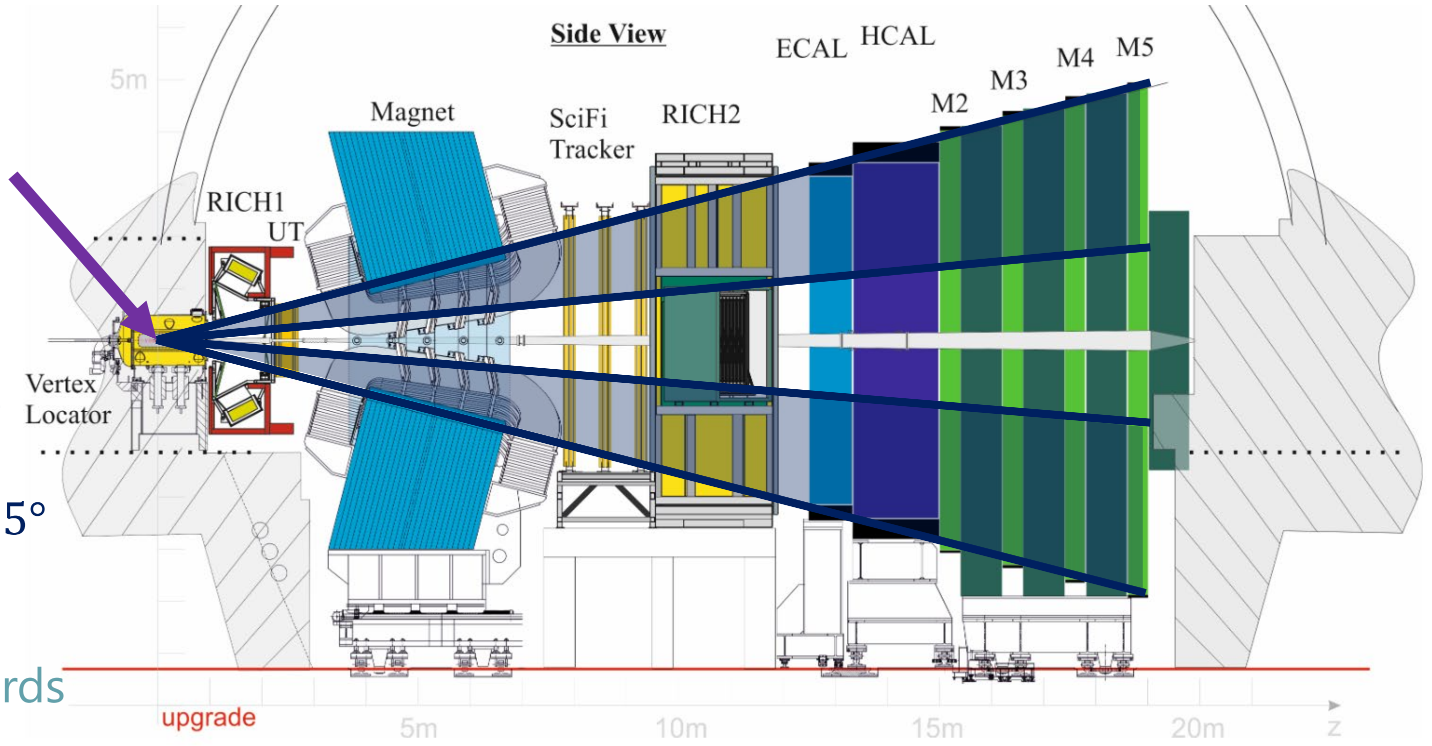
10 GB/s

3 Problem Formulation

b Allen: a Fully GPU-based trigger

Collisions (Run 3)

- 25 MHz non-empty bunch crossing rate
- ~ 5 p-p collisions / bunch crossing
- p-p collision at $\sqrt{s} = 13.6$ TeV



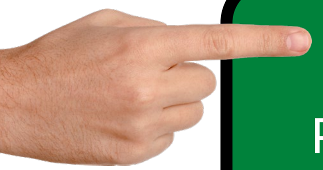
LHCb Subdetectors

Acceptance
$2 < \eta < 5$
$1° < \theta < 15°$

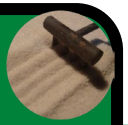
Digitization

PCIe40 boards

5 TB/s



Allen [HLT1]
$\mathcal{O}(500)$ GPUs
Partial reconstruction
Partial Selection



70-200 GB/s

Storage buffer

HLT2
$\mathcal{O}(3000)$ CPU x86 servers
Full reconstruction
Full selection

10 GB/s

3 Problem Formulation

b Allen: a Fully GPU-based trigger

CPU
Central Processor Unit

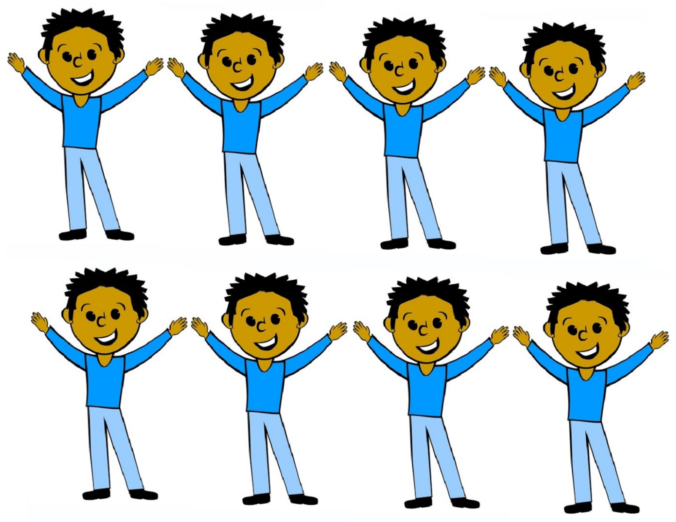
O(10-100) fast cores running in parallel

Optimised for **latency**
(≡ duration of a single operation)

GPU
Graphical Processor Unit

O(1000-10000) slow cores running in parallel

Optimised for **throughput**
(≡ # operations / second)



Programming Language for NVIDIA GPUs: CUDA

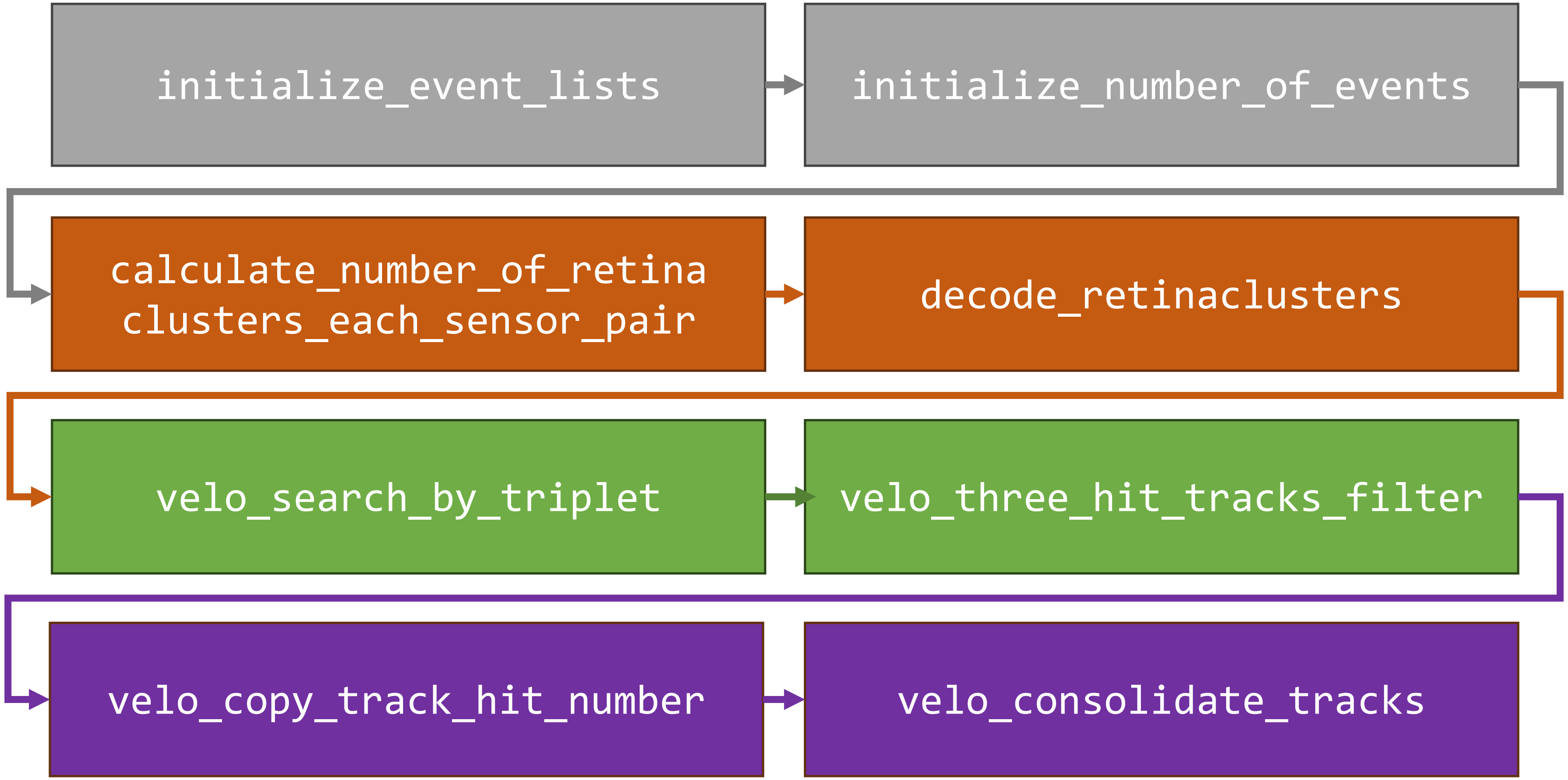
3 Problem Formulation

b Allen: a Fully GPU-based trigger

Allen = framework

- **Algorithm**: C++/CUDA
- **Sequence**: Python

Sequence for "Search by Triplet" algorithm for track-finding in the Velo



3 Problem Formulation

c Motivation

- **Search by Triplet:** $\mathcal{O}(10\%)$ total reconstruction time
- **High-Luminosity Phase of LHCb (Run 5)**: 42 visible pp collisions per bunch crossing
  - $\times 8$ particles
  - Higher detector occupancy
- **Classical algorithms** scale **worse than quadratically** with n_{hits}.

$\Rightarrow$ **Neural-Network Algorithms?**

3 Problem Formulation

C Motivation

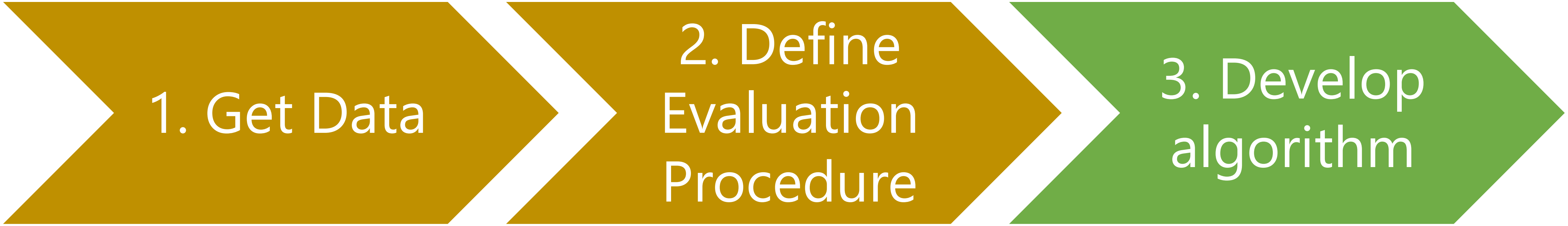
- **Search by Triplet:** $\mathcal{O}(10\%)$ total reconstruction time
- **High-Luminosity Phase of LHCb (Run 5)**: 42 visible pp collisions per bunch crossing
  - $\times$ 8 particles
  - Higher detector occupancy
- **Classical algorithms** scale **worse than quadratically** with n_{hits}.
  $\Rightarrow$ **Neural-Network Algorithms?**
- **Allen** = optimal **bench for developping neural network** solution
  - **GPU** inference
  - Reference algorithm **on GPU** (Search by Triplet)

Goal: **develop**, **optimize** and **evaluate** a **neural network-based pipeline** for **track-finding in the Velo**

1 Beginner Introduction
2 Neural Network Introduction
3 Problem Formulation
4 **Experimental Setup**
5 Exa.TrkX Pipeline
6 From Exa.TrkX to ETX4VELO
7 Implementation in Allen
8 Optimization

4 Experimental Setup

- a **Evaluation Setup**
- b **Data Acquisition**
- c **Performance of Search by Triplet**



4 Experimental Setup

a Evaluation Setup

Two kinds of **evaluation**:

- **Physics performance**: *how well particles are recovered*
- **Throughput**: *# events / second*

4 Experimental Setup

a Evaluation Setup

Two kinds of **evaluation**:

- **Physics performance**: *how well particles are recovered*
- **Throughput**: *# events / second*

In this presentation, **2 metrics** for **physics performance**:

$$\text{Efficiency} = \frac{\#\ \text{reconstructed particles}}{\#\ \text{particles}} \qquad \text{Fake rate} = \frac{\#\ \text{fake tracks}}{\#\ \text{tracks}}$$

4 Experimental Setup

a Evaluation Setup

Two kinds of **evaluation**:

- **Physics performance**: *how well particles are recovered*
- **Throughput**: *# events / second*

In this presentation, **2 metrics** for **physics performance**:

$$\text{Efficiency} = \frac{\text{\# reconstructed particles}}{\text{\# particles}} \qquad \text{Fake rate} = \frac{\text{\# fake tracks}}{\text{\# tracks}}$$

Allen implements both **physics performance** and **throughput** evaluation.

→ **Physics performance** in Python: I developed `MonteTracko` library.
Produce **performance report** (tables, histograms)

→ **Throughput** measured through **Allen**.

4 Experimental Setup

b Data Acquisition

CERN grid

Simulated files in the (X)DIGI format

digout library

"local"

Table of hits

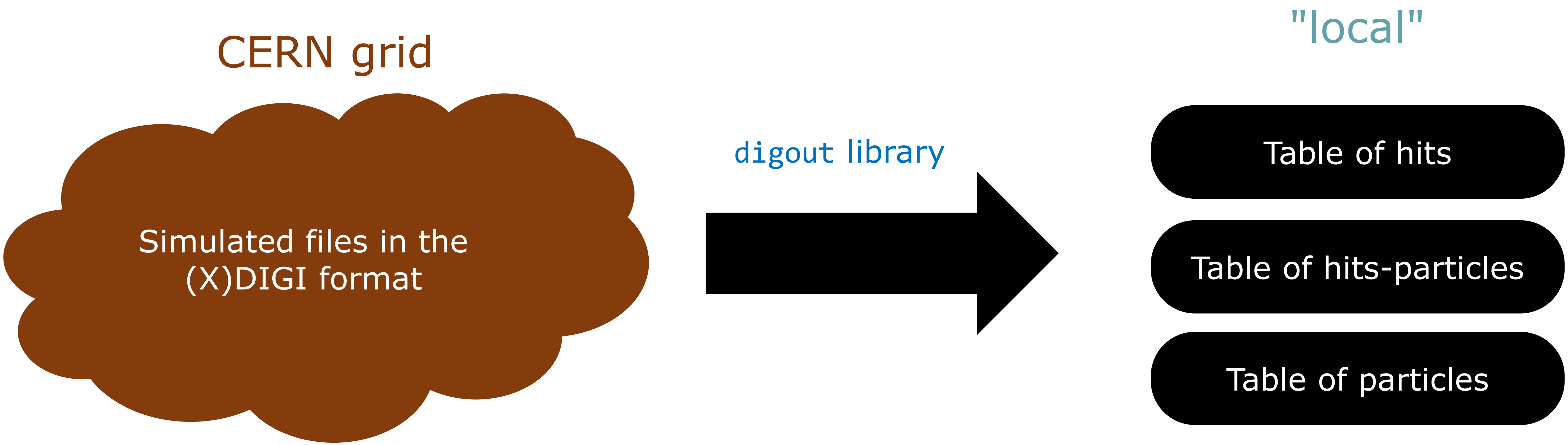
Table of hits-particles

Table of particles

In format **Parquet**
With **LZ4 compression**

4 Experimental Setup

b Data Acquisition



In format **Parquet**
With **LZ4 compression**

digout library: from a *bookkeeping path*

1. Find the corresponding (X)DIGI files
2. Batch submission (HTCONDOR) : conversion of each (X)DIGI file in parallel

4 Experimental Setup

C Performance of Search by Triplet

Performance of Search by Triplet on 1000 simulated events (with spillover)

Physics performance

Metric	Category	Search By Triplet
Efficiency	**Long, no electrons**	99.35%
	Long electrons	97.53%
	Long from strange	95.21%
Fake rate		2.19%
Throughput (# events / s) *NVIDIA RTX 2080Ti*		595 kHz

1 Beginner Introduction

2 Neural Network Introduction

3 Problem Formulation

4 Experimental Setup

5 Exa.TrkX Pipeline

6 From Exa.TrkX to ETX4VELO

7 Implementation in Allen

8 Optimization

5 Exa.TrkX Pipeline

a **Why Graphs?**

b **Pipeline Overview**

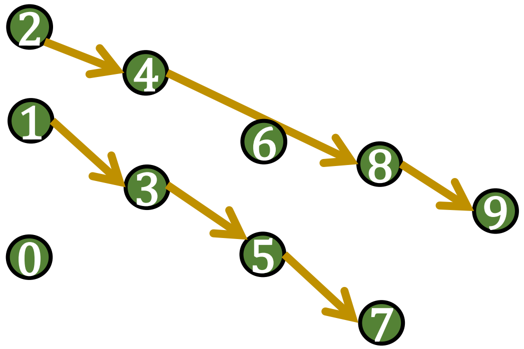
c **Step 1: Embedding + FRNN**

d **Step 2: GNN Edge Classifier**

e **Results**

5 Exa.TrkX Pipeline

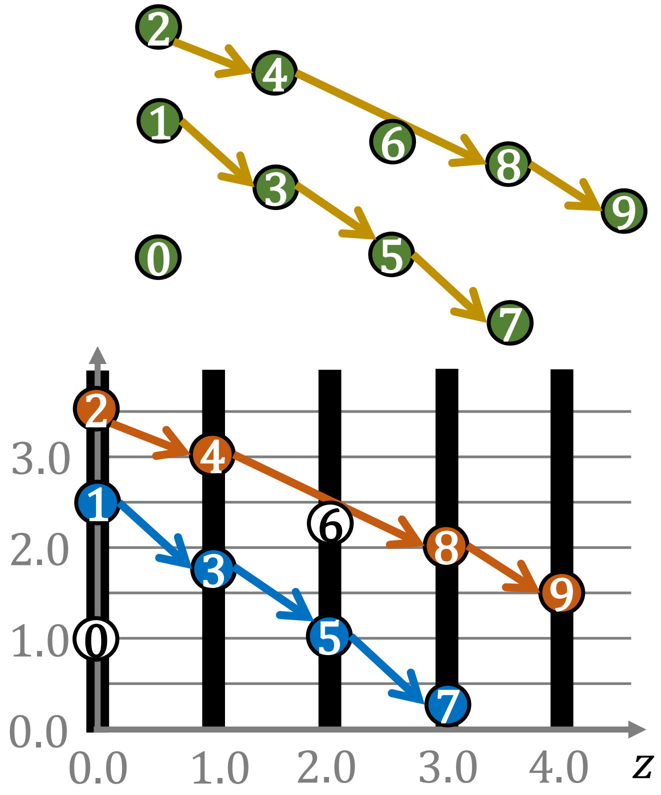
a Why Graphs?



- **Graph** $\mathcal{G}$ is defined as
  - Set of **nodes** $\mathcal{V}$: indexed from 0 to 9
  - Set of **edges** $\mathcal{E}$ ≡ connection between nodes

5 Exa.TrkX Pipeline

a Why Graphs?

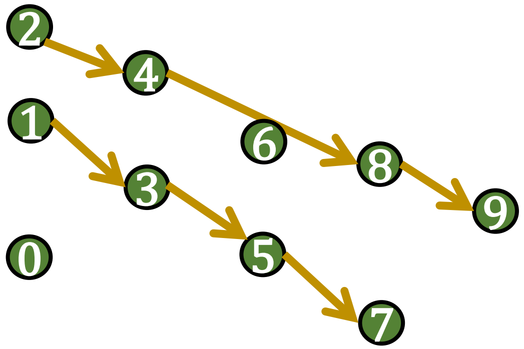


- **Graph** $\mathcal{G}$ is defined as
  - Set of **nodes** $\mathcal{V}$: indexed from 0 to 9
  - Set of **edges** $\mathcal{E}$ ≡ connection between nodes

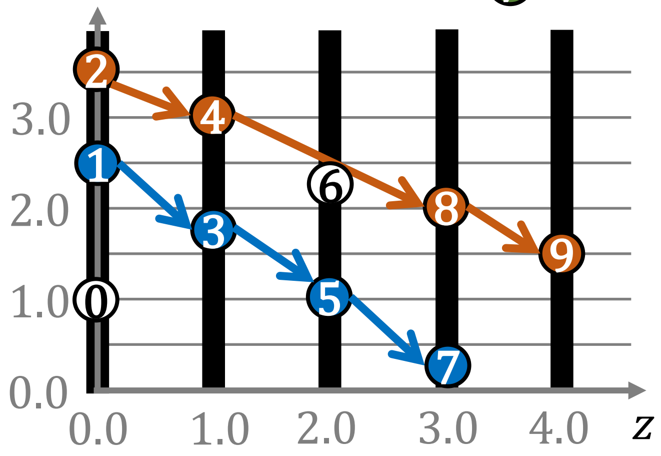
- **Graph** ≡ Natural way of representing an event in a tracking detector
  - Set of **nodes**: hits
  - Set of **edges**: adjacent hits belonging to the same particle

5 Exa.TrkX Pipeline

a Why Graphs?

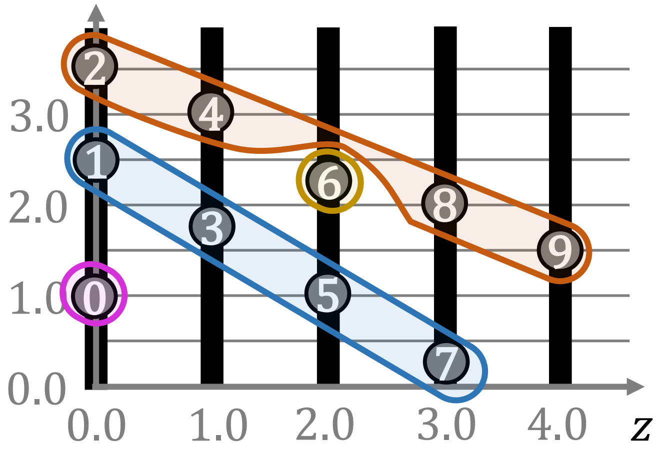


- **Graph** $\mathcal{G}$ is defined as
  - Set of **nodes** $\mathcal{V}$: indexed from 0 to 9
  - Set of **edges** $\mathcal{E}$ ≡ connection between nodes



- **Graph** ≡ Natural way of representing an event in a tracking detector
  - Set of **nodes**: hits
  - Set of **edges**: adjacent hits belonging to the same particle

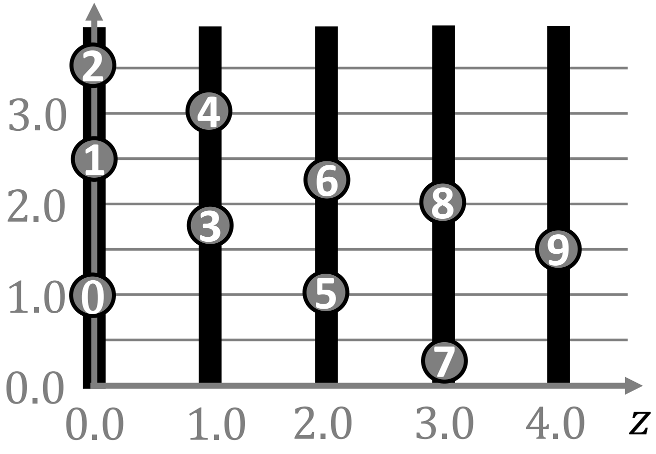
WCC algorithm

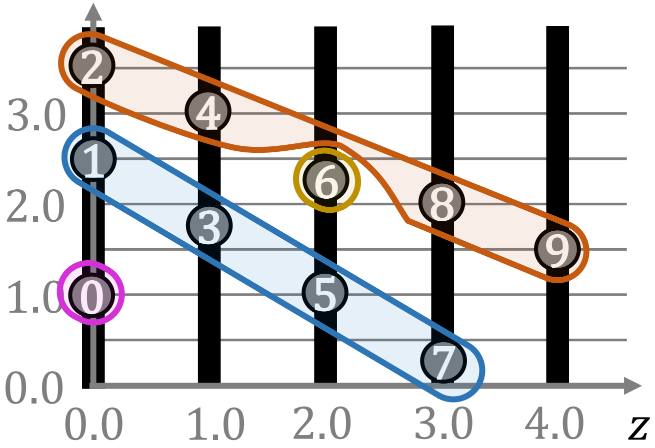


- **Weakly Connected Component** (**WCC**) **algorithm**:
  WCC = Nodes indirectly connected to each other
  → **Tracks** = WCC with at least 3 nodes

5 Exa.TrkX Pipeline

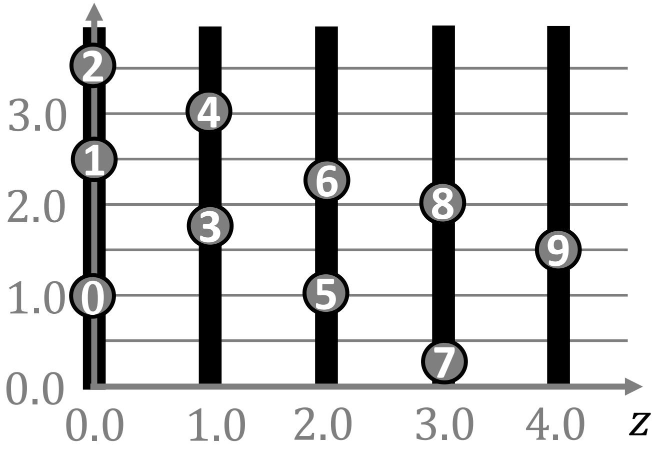
b Pipeline Overview

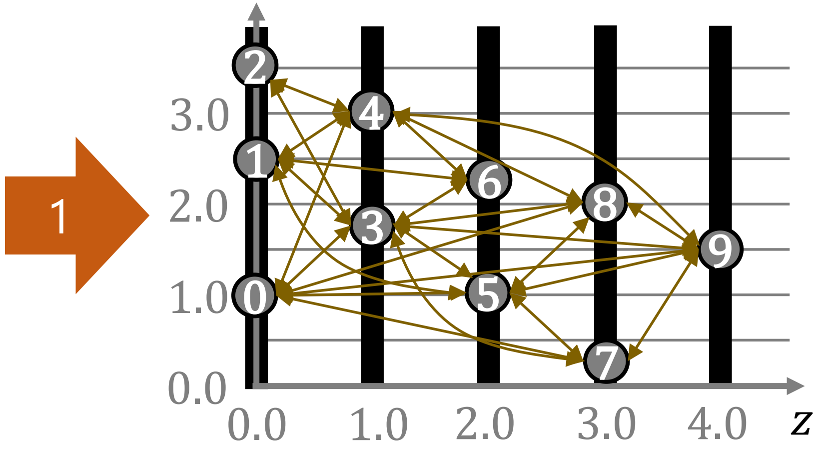


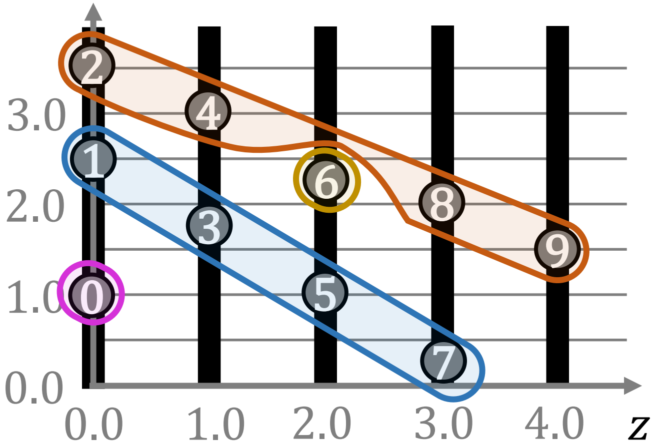


5 Exa.TrkX Pipeline

b Pipeline Overview





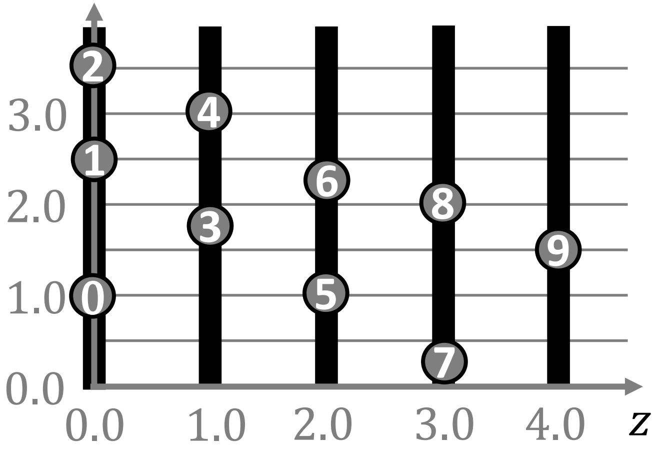


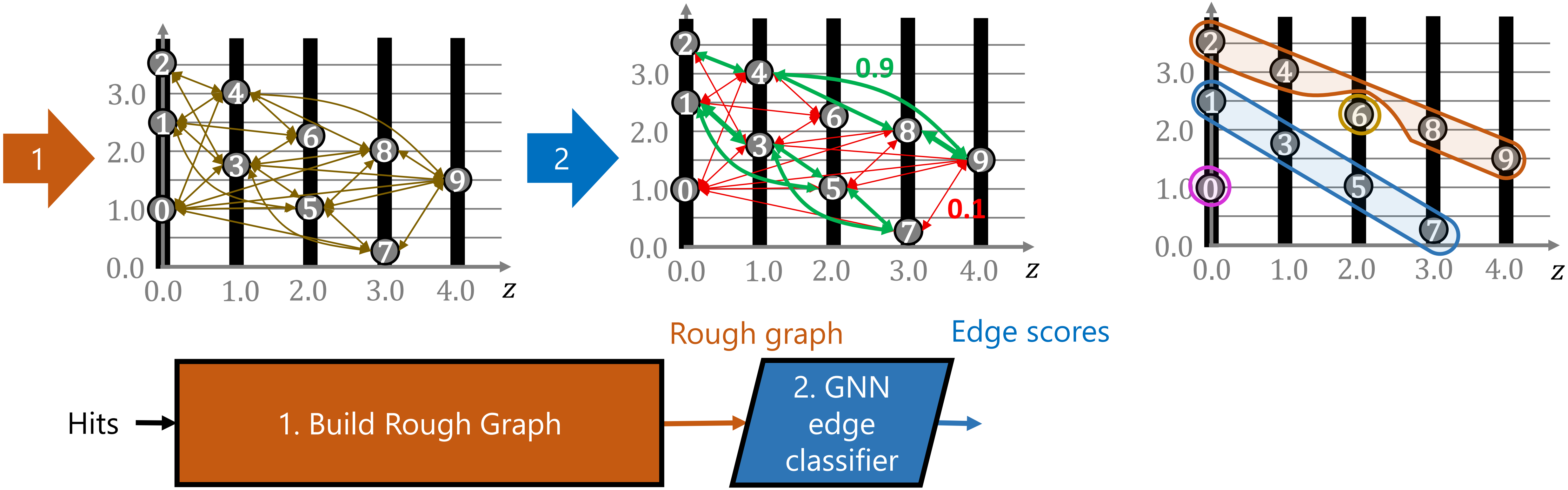
Rough graph

Hits → 1. Build Rough Graph →

5 Exa.TrkX Pipeline

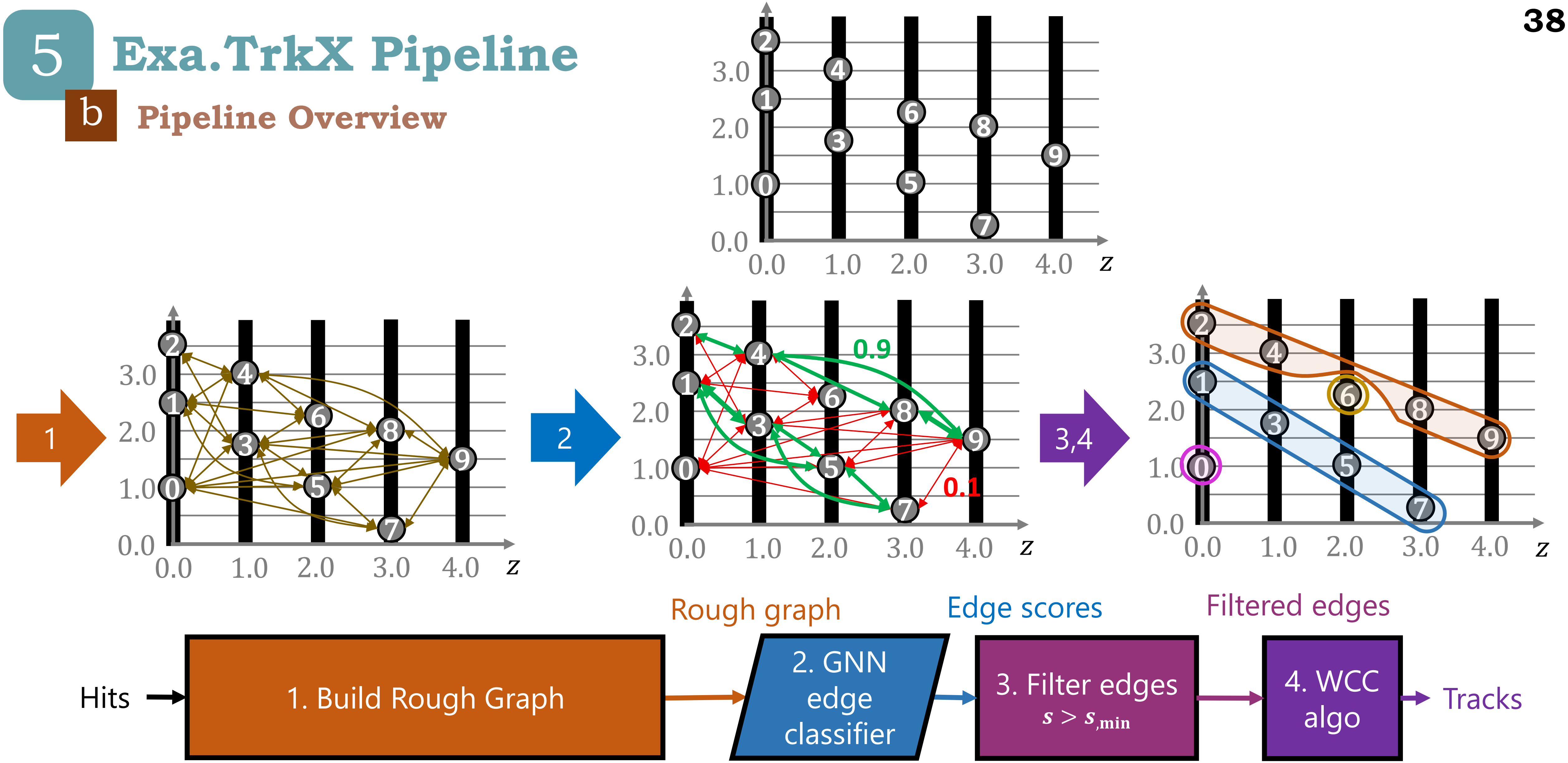
b Pipeline Overview





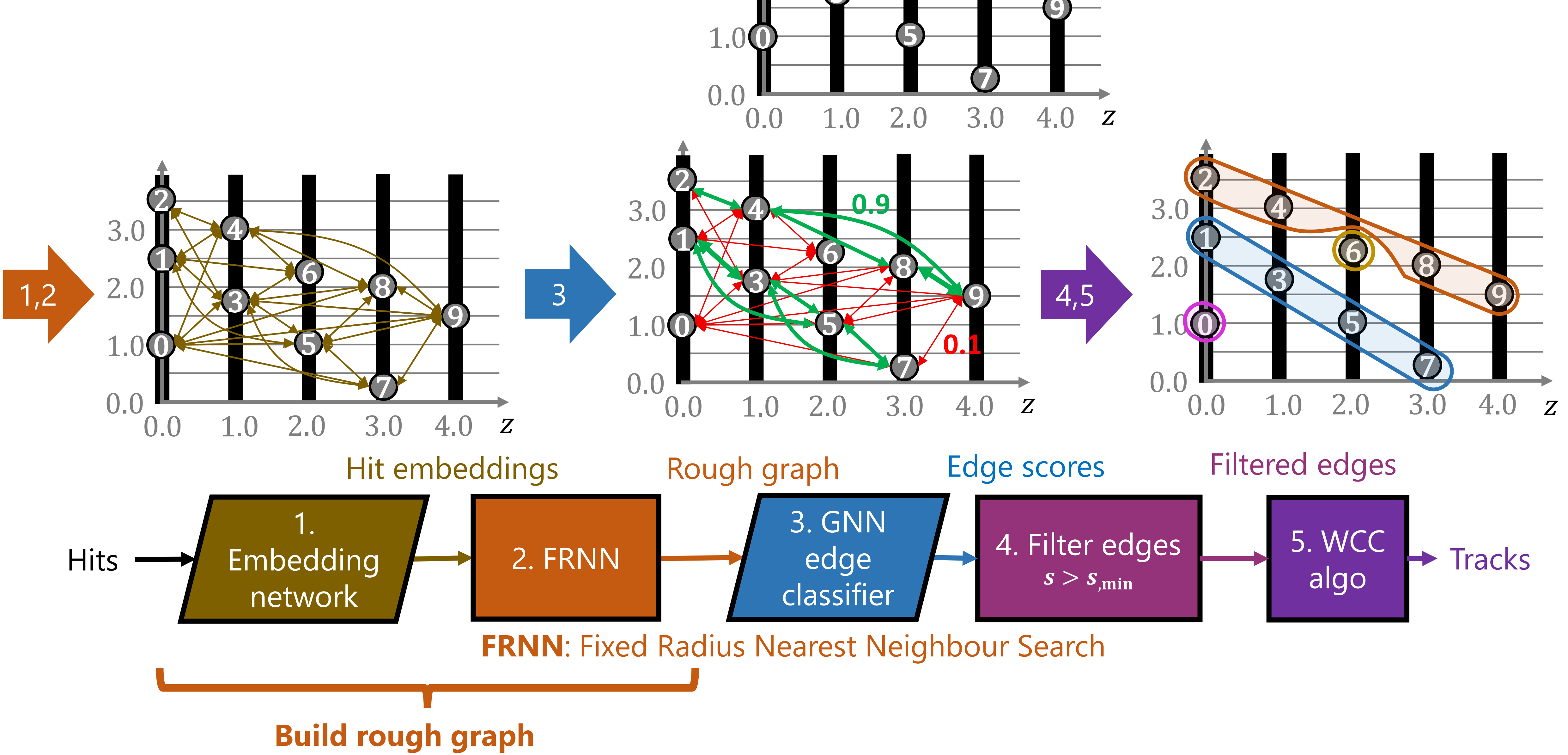
5 Exa.TrkX Pipeline

b Pipeline Overview



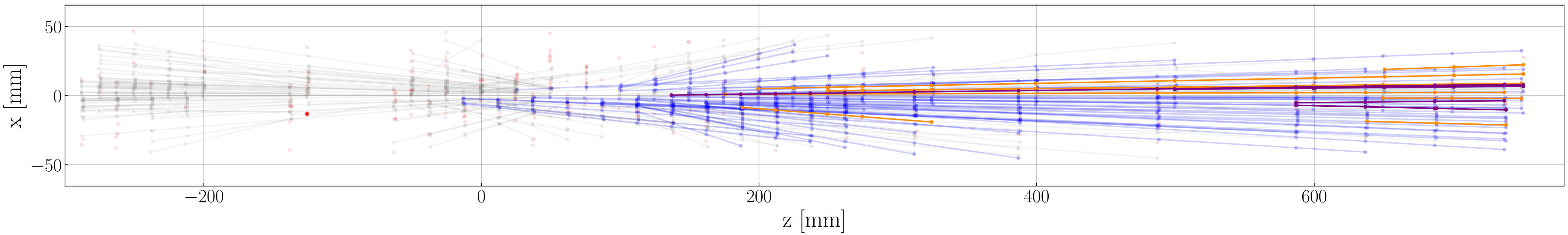
5 Exa.TrkX Pipeline

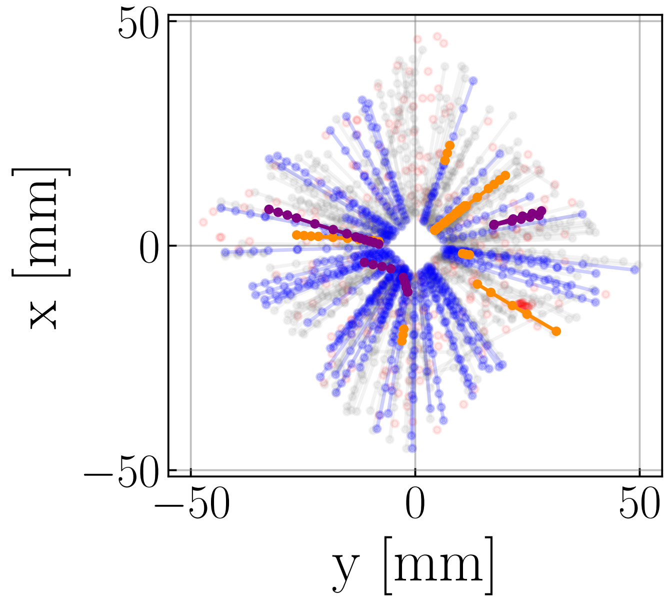
b Pipeline Overview



5 Exa.TrkX Pipeline

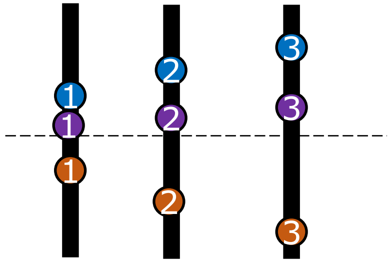
c Graph Building





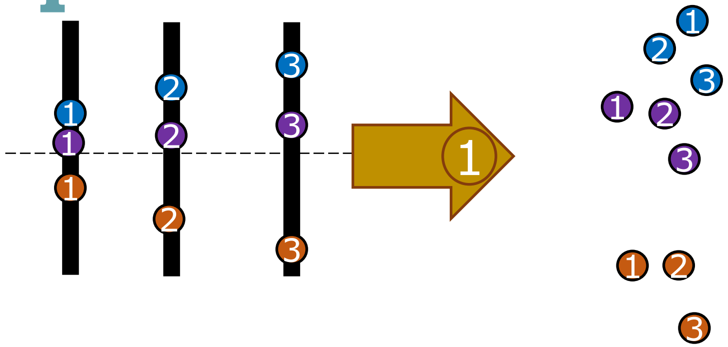
5 Exa.TrkX Pipeline

C Graph Building



5 Exa.TrkX Pipeline

C Graph Building



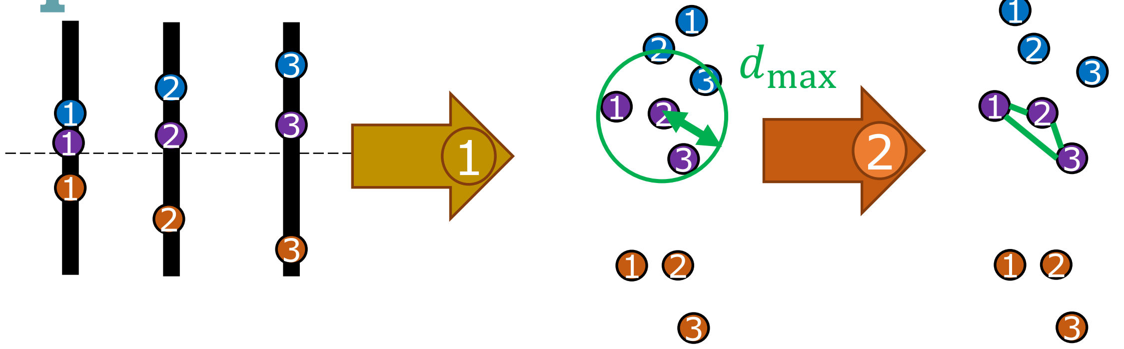
1 Learn a d_{emb}-dimensional embedding

$$\vec{x} = \begin{pmatrix} r \\ \phi \\ z \\ \text{layer} \end{pmatrix} \longrightarrow \text{Embedding Network (MLP)} \longrightarrow \vec{e}$$

- MLP maps each hit → vector $\vec{e}$ in $\mathbb{R}^7$
- Network trained so that:
  - **Connected** hits → **close** in embedding space
  - **Non-connected** hits → **far apart** in embedding space

5 Exa.TrkX Pipeline

C Graph Building



1 Learn a $\mathrm{d_{emb}}$-dimensional embedding

$$\vec{x} = \begin{pmatrix} r \\ \phi \\ z \\ \text{layer} \end{pmatrix} \longrightarrow \boxed{\text{Embedding Network (MLP)}} \longrightarrow \vec{e}$$

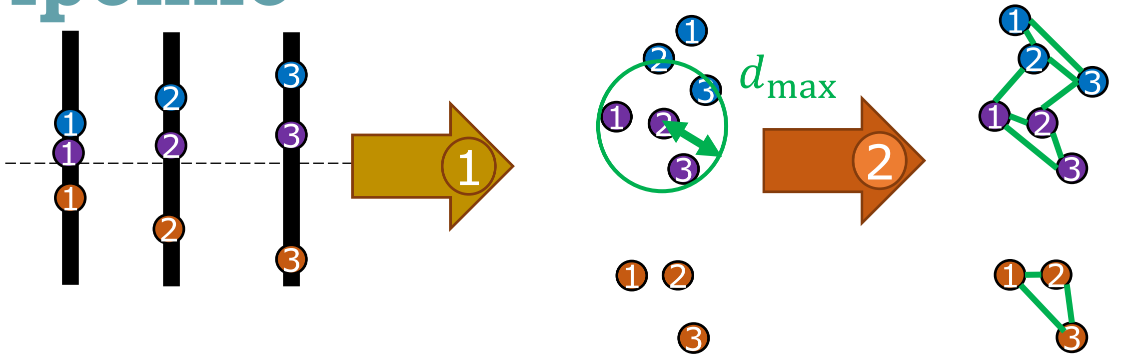
- MLP maps each hit → vector $\vec{e}$ in $\mathbb{R}^7$
- Network trained so that:
  - **Connected** hits → **close** in embedding space
  - **Non-connected** hits → **far apart** in embedding space

2 Build candidate edges with Fixed Radius Nearest Neighbours (FRNN)

1. For each hit i, find **neighbours within radius $\boldsymbol{d_{\max}}$** in embedding space:
$$\mathrm{FRNN}(i) = \{\text{hits j} \,|\, \|\vec{e_i} - \vec{e_j}\| < d_{\max}\}$$
2. Limit to $\boldsymbol{k_{\max}} = \mathbf{50}$ neighbours / hit

5 Exa.TrkX Pipeline

C Graph Building



1 Learn a d_{emb}-dimensional embedding

$$\vec{x} = \begin{pmatrix} r \\ \phi \\ z \\ \text{layer} \end{pmatrix} \longrightarrow \boxed{\text{Embedding Network (MLP)}} \longrightarrow \vec{e}$$

- MLP maps each hit → vector $\vec{e}$ in $\mathbb{R}^7$
- Network trained so that:
  - **Connected** hits → **close** in embedding space
  - **Non-connected** hits → **far apart** in embedding space

2 Build candidate edges with Fixed Radius Nearest Neighbours (FRNN)

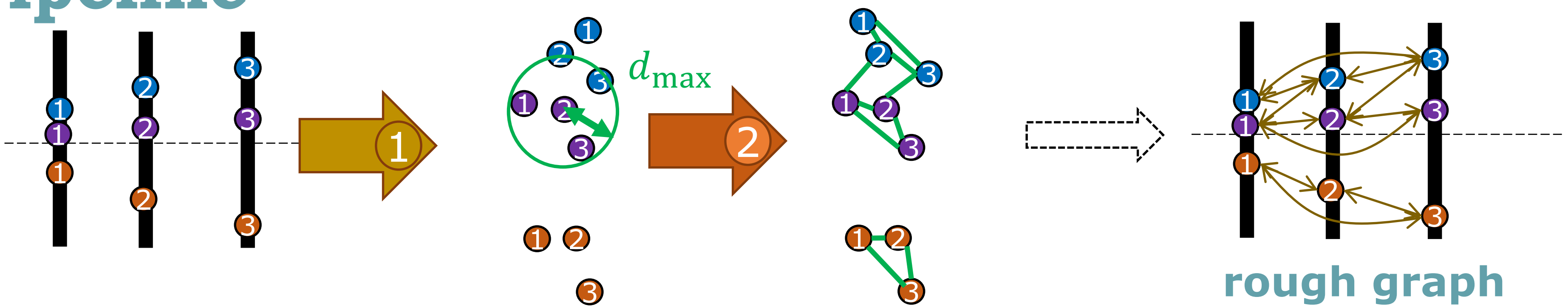
1. For each hit i, find **neighbours within radius $\boldsymbol{d_{max}}$** in embedding space:

$$\text{FRNN}(i) = \{\text{hits j} \mid \|\vec{e_i} - \vec{e_j}\| < d_{max}\}$$

2. Limit to $\boldsymbol{k_{max}} = \mathbf{50}$ neighbours / hit

5 Exa.TrkX Pipeline

C Graph Building



1 Learn a d_{emb}-dimensional embedding

$$\vec{x} = \begin{pmatrix} r \\ \phi \\ z \\ \text{layer} \end{pmatrix} \longrightarrow \boxed{\text{Embedding Network (MLP)}} \longrightarrow \vec{e}$$

- MLP maps each hit → vector $\vec{e}$ in $\mathbb{R}^7$
- Network trained so that:
  - **Connected** hits → **close** in embedding space
  - **Non-connected** hits → **far apart** in embedding space

2 Build candidate edges with Fixed Radius Nearest Neighbours (FRNN)

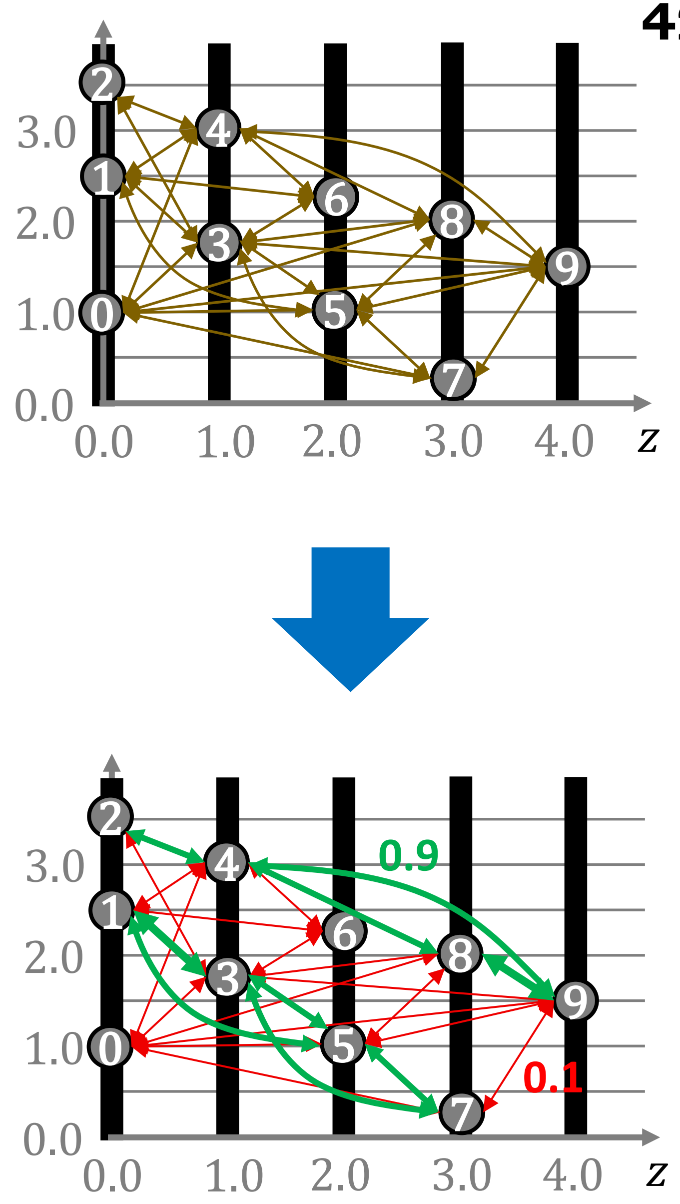
1. For each hit i, find **neighbours within radius** $\boldsymbol{d_{\max}}$ in embedding space:

$$\mathrm{FRNN}(i) = \left\{\text{hits j} \middle| \left\|\vec{e_i} - \vec{e_j}\right\| < d_{\max}\right\}$$

2. Limit to $\boldsymbol{k_{\max}} = \mathbf{50}$ neighbours / hit
3. Add an **edge** to each neighbour

5 Exa.TrkX Pipeline

d Edge Classification



5 Exa.TrkX Pipeline

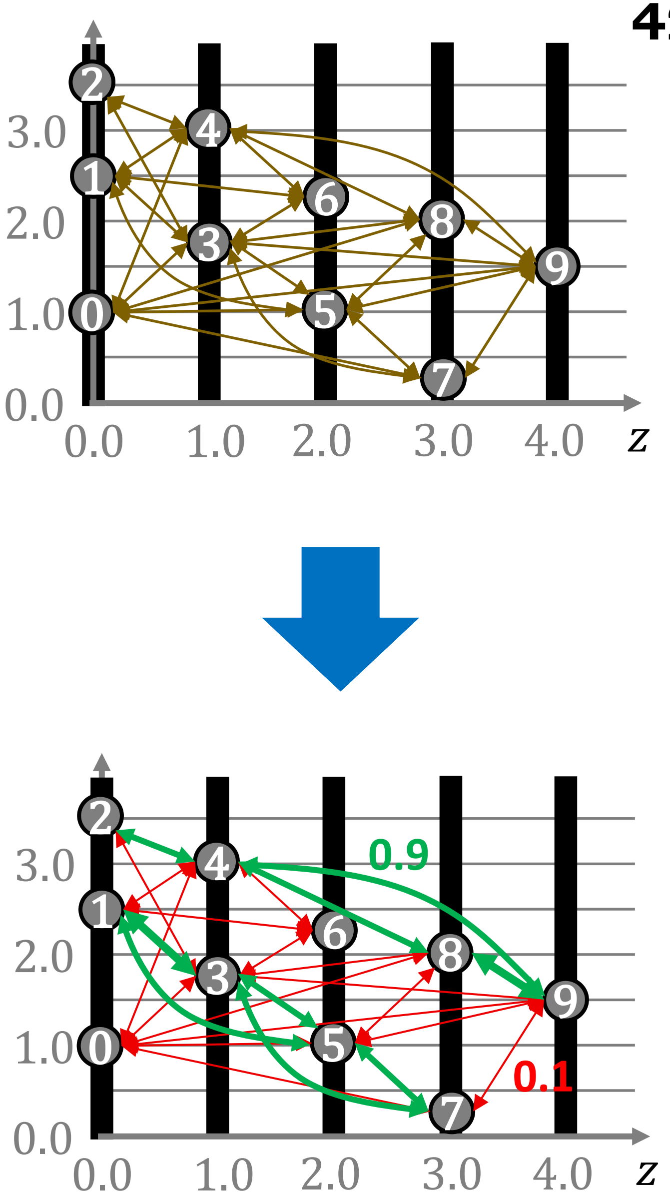
d Edge Classification

3 **GNN Edge Classifier**: outputs edge score ∈ [0, 1]

- 5 MLPs
- `scatter_add`
- `scatter_max`

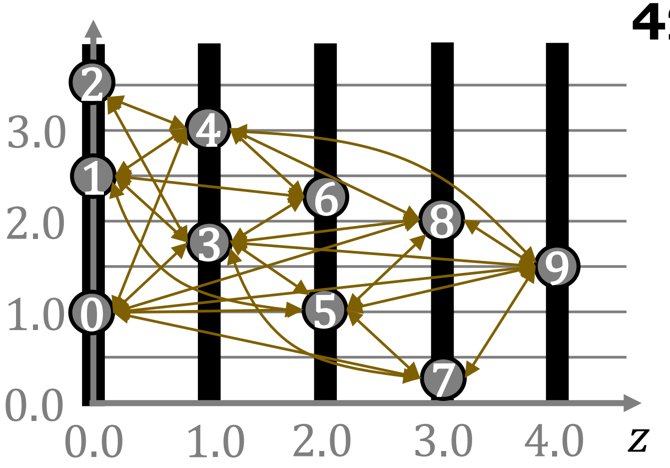
Cylindrical Hit coordinates → **GNN Edge Classifier** → Edge scores S

Edges →



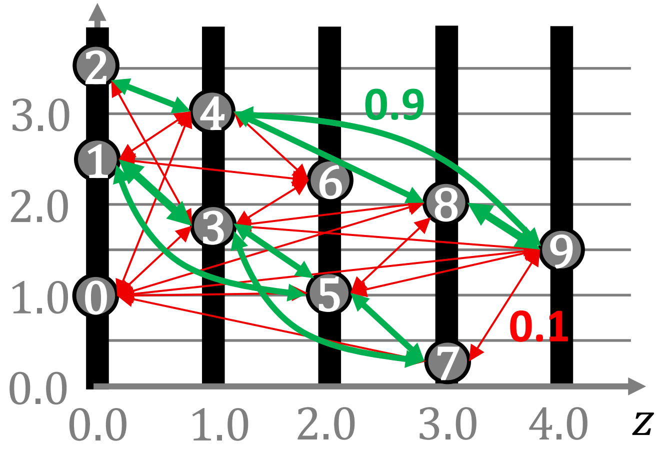
5 Exa.TrkX Pipeline

d Edge Classification



3 **GNN Edge Classifier**: outputs edge score ∈ [0, 1]

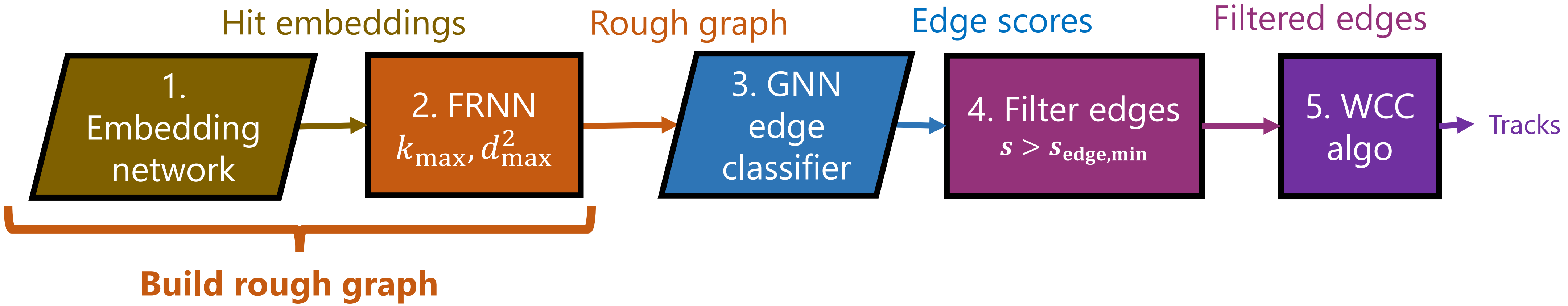
- 5 MLPs
- `scatter_add`
- `scatter_max`



4 **Edge Filtering**: $s > s_{\min} = 0.5$

5 Exa.TrkX Pipeline

e Results



- Adaption of the Exa.TrkX pipeline to LHCb data by Fotis Giasemis.
- Training/testing with **1000** simulated events **without spillover**

Metric	Category	Allen	Exa.TrkX
Efficiency	Velo no electrons	98.25%	93.72%
	Long electrons	96.55%	64.14%
	Long from strange	97.74%	88.89%
Fake rate		0.88%	7.50%

1. Beginner Introduction
2. Neural Network Introduction
3. Problem Formulation
4. Experimental Setup
5. Exa.TrkX Pipeline
6. **From Exa.TrkX to ETX4VELO**
7. Implementation in Allen
8. Optimization

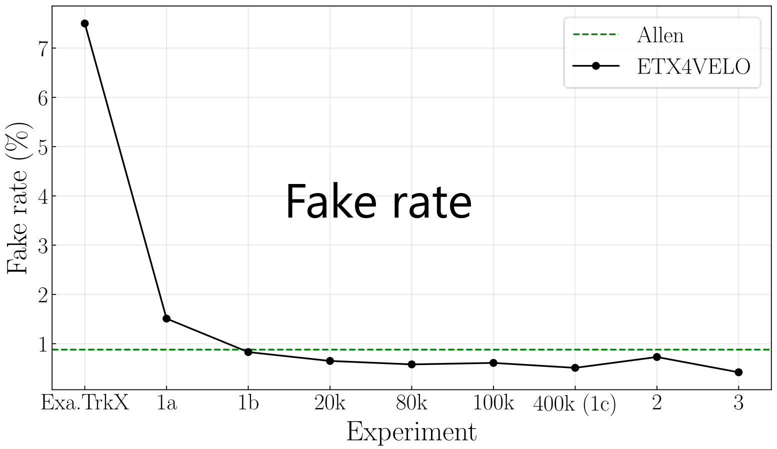
6 ETX4VELO Pipeline

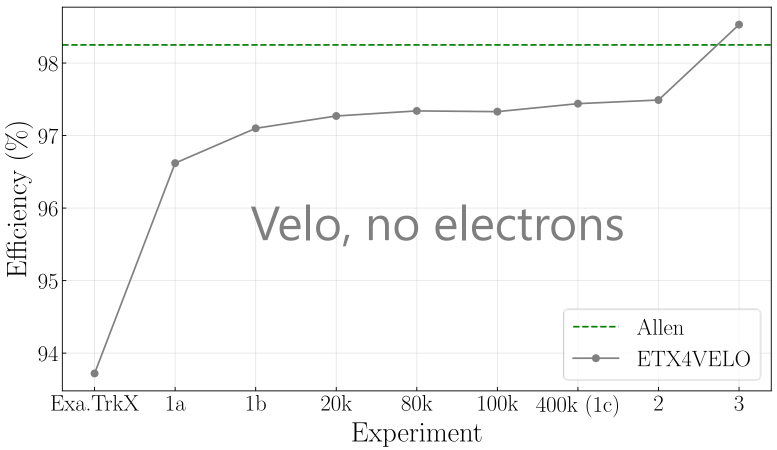
a Round 1: First Improvements

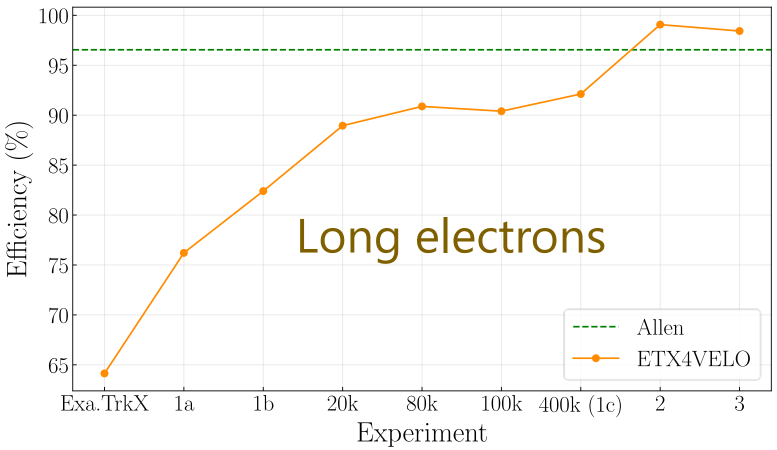
b Round 2: Improving Long Electrons

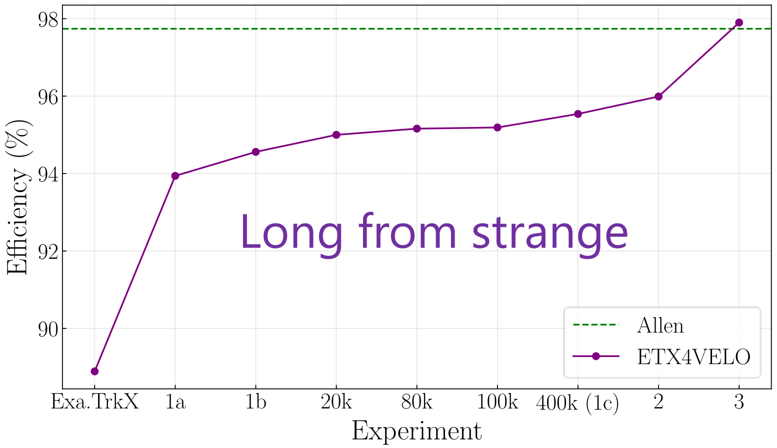
c Round 3: Improving long from strange

d Results Without Spillover







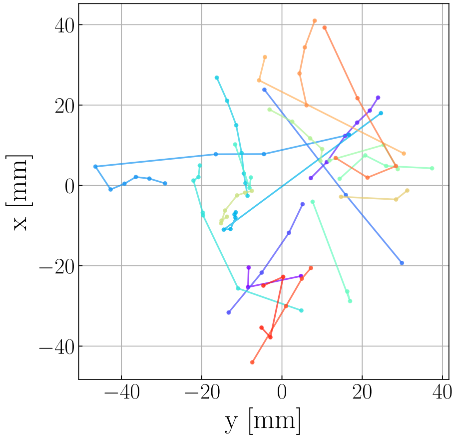


6 From Exa.TrkX to ETX4VELO

a Round 1a: Remove Scattered Trajectories

- Remove **scattered trajectories** from training set.
    1. Fit a straight 3D line to each particle
    2. Remove particles whose Root Mean Square (RMS) distance between hits and line > 0.8 mm *[extremely conservative]*

- Training with **10,000 events**

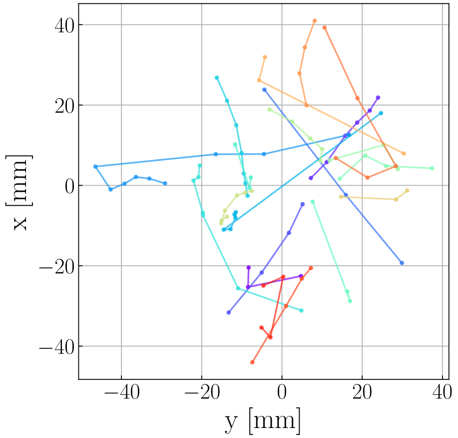


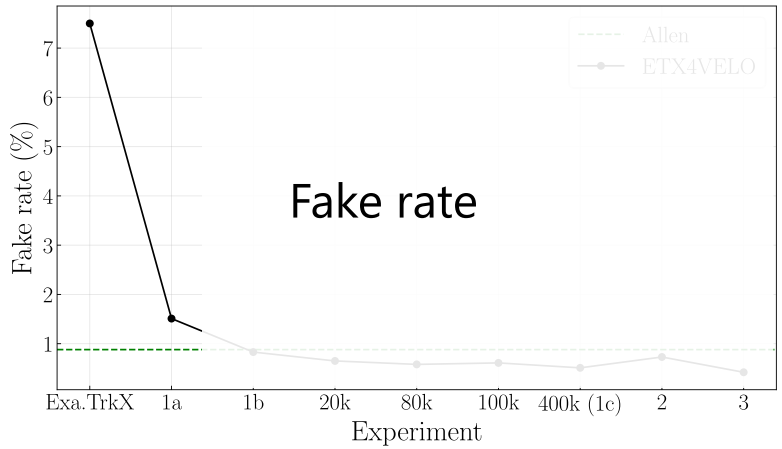
6 From Exa.TrkX to ETX4VELO

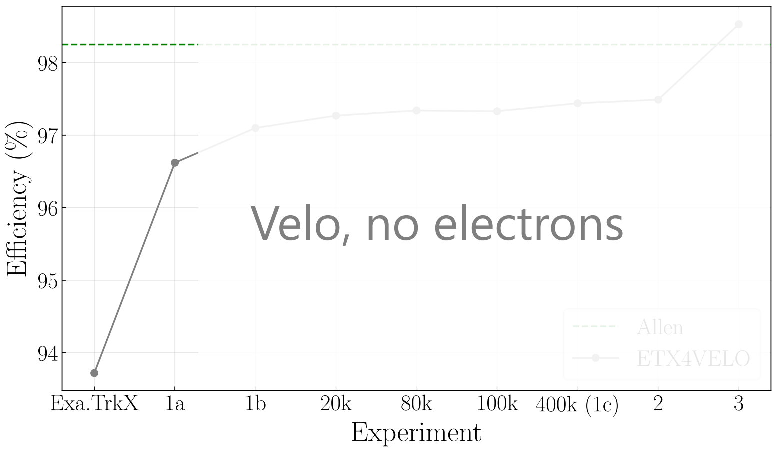
a Round 1a: Remove Scattered Trajectories

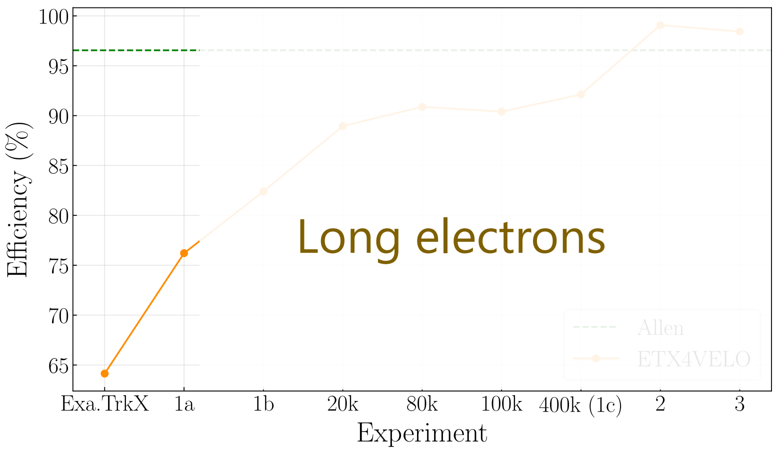
- Remove **scattered trajectories** from training set.
  1. Fit a straight 3D line to each particle
  2. Remove particles whose Root Mean Square (RMS) distance between hits and line > 0.8 mm *[extremely conservative]*

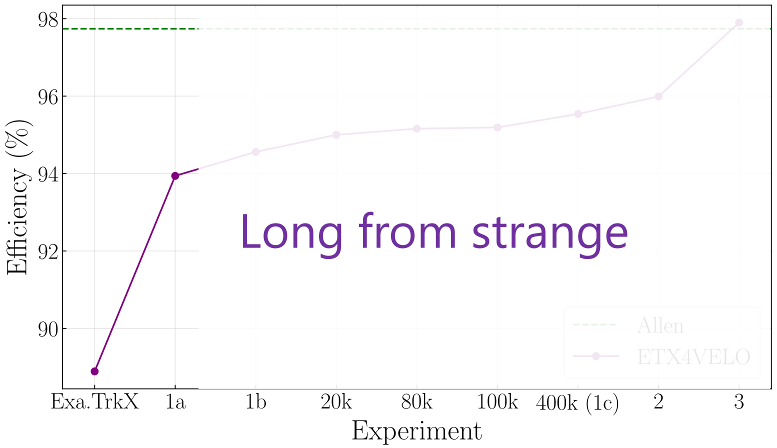
- Training with **10,000 events**









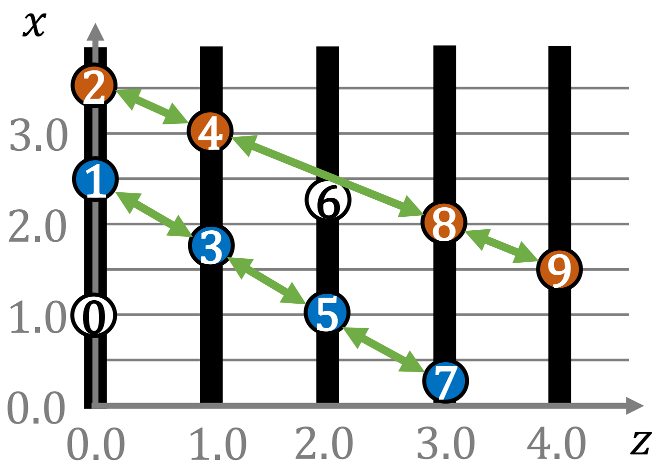


6 From Exa.TrkX to ETX4VELO

a Round 1b: True Edges

- **True edges** used as **target** during **training** of the **embedding network** and **GNN**.

- **Exa.TrkX definition**:
  - Connect consecutive hits ordered from origin vertex
  - “Bidirectional”: edges duplicated in the other direction



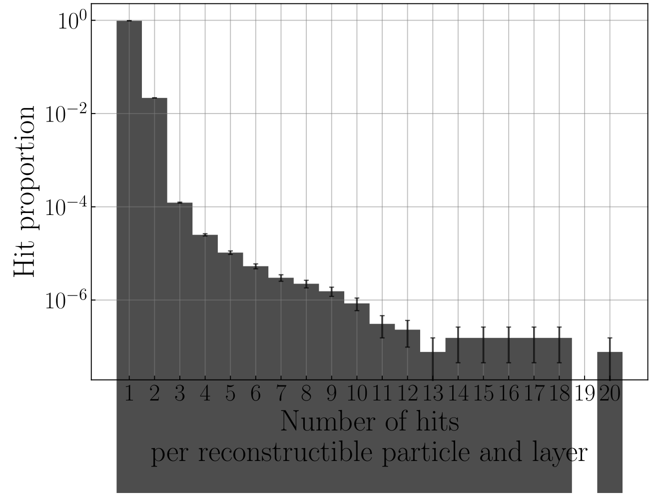
6 From Exa.TrkX to ETX4VELO

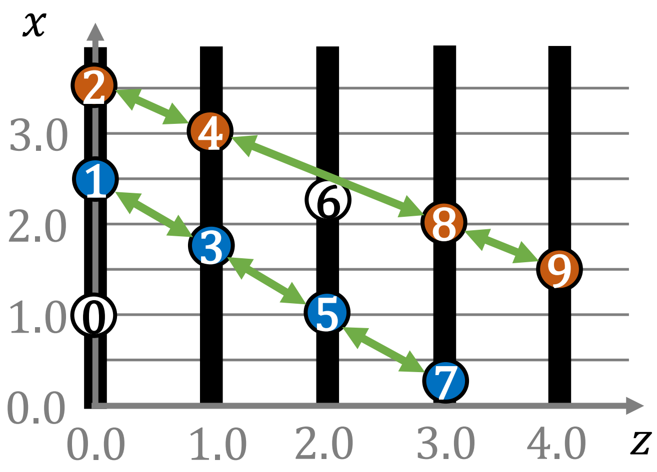
a Round 1b: True Edges

- **True edges** used as **target** during **training** of the **embedding network** and **GNN**.

- **Exa.TrkX definition**:
  - Connect consecutive hits ordered from origin vertex
  - “Bidirectional”: edges duplicated in the other direction

- **Problem**: 13% of reconstructible particles have at least 2 hits in a layer
  ⇒ edges within the same layer





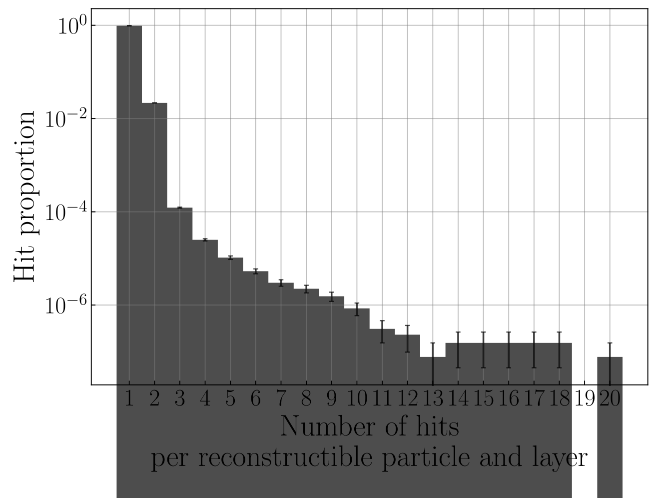
6 From Exa.TrkX to ETX4VELO

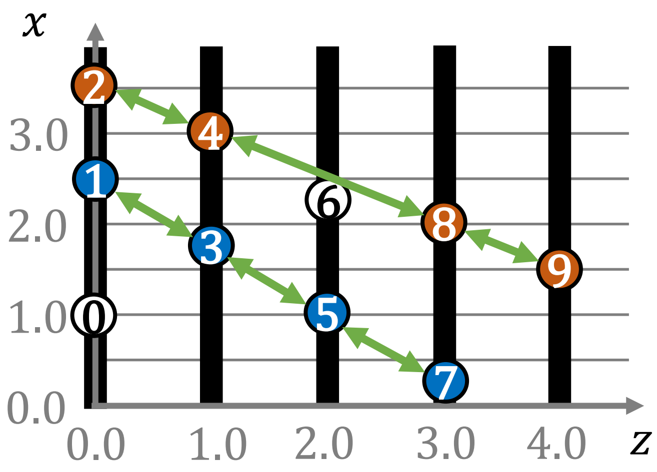
a Round 1b: True Edges

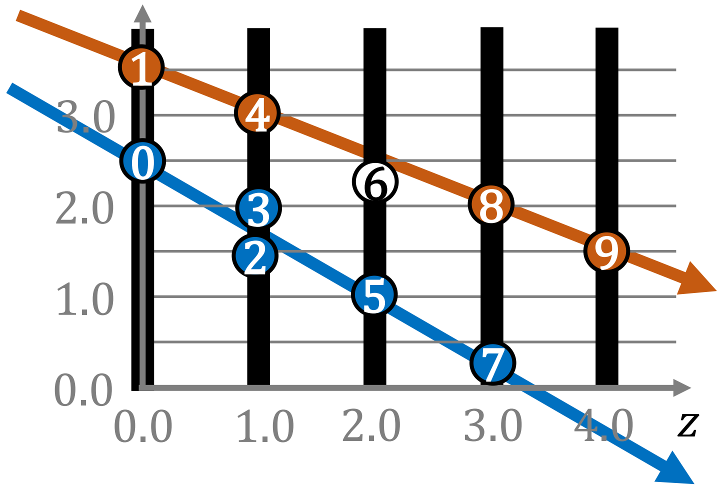
- **True edges** used as **target** during **training** of the **embedding network** and **GNN**.

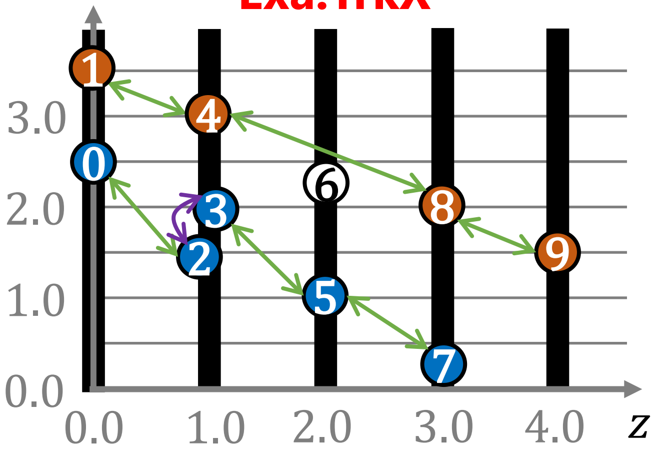
- **Exa.TrkX definition**:
    - Connect consecutive hits ordered from origin vertex
    - “Bidirectional”: edges duplicated in the other direction

- **Problem**: 13% of reconstructible particles have at least 2 hits in a layer
  ⇒ edges within the same layer





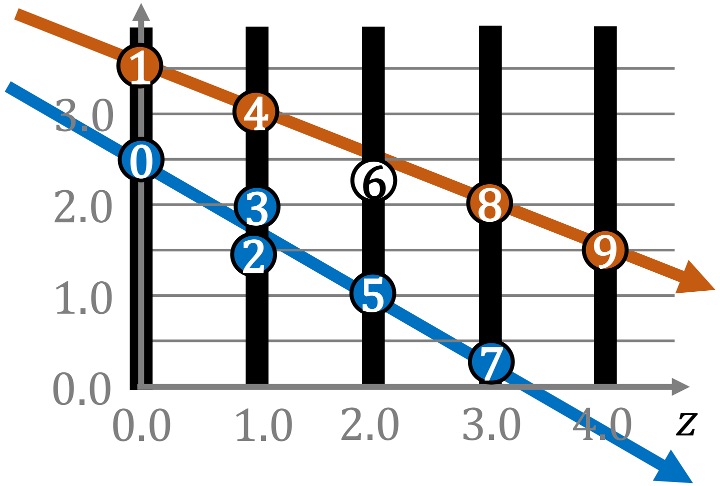


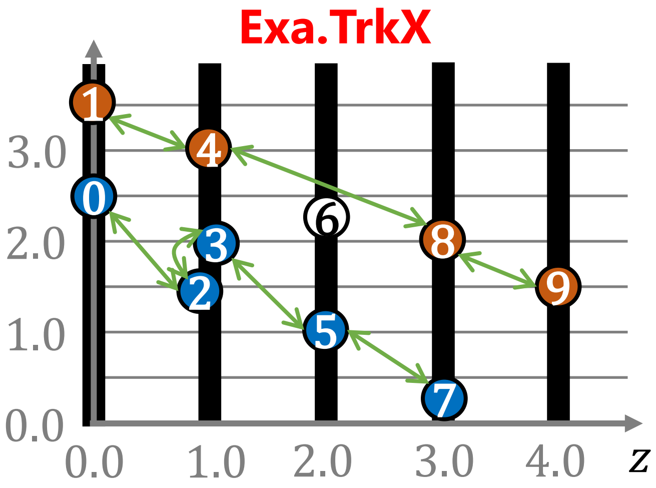


6 From Exa.TrkX to ETX4VELO

a Round 1b: True Edges

- **Problem**:
  - edges within the same layer
  - Bidirectional = # edges × 2

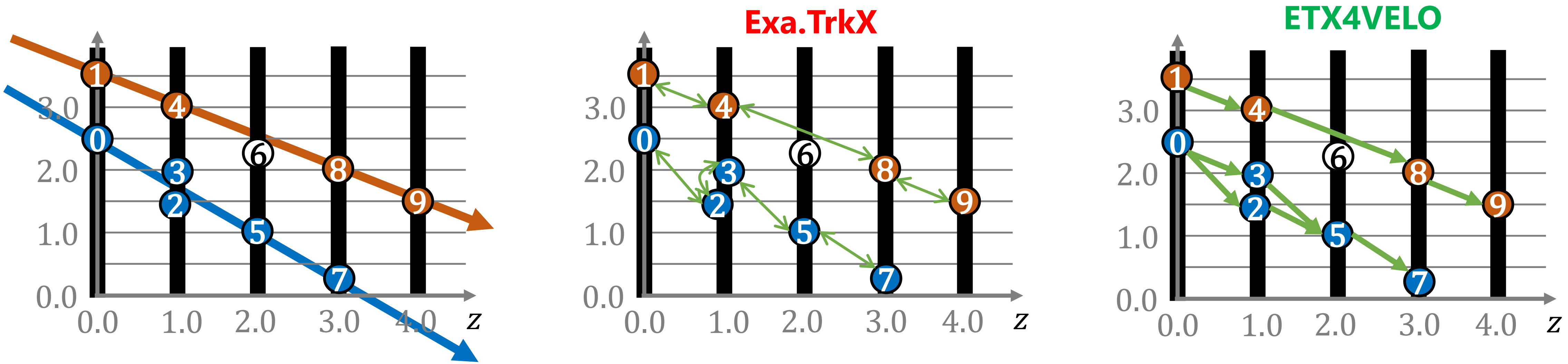




6 From Exa.TrkX to ETX4VELO

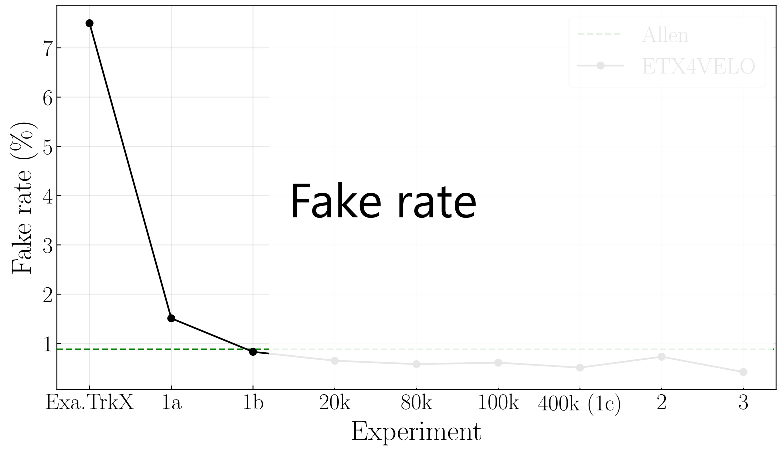
a Round 1b: True Edges

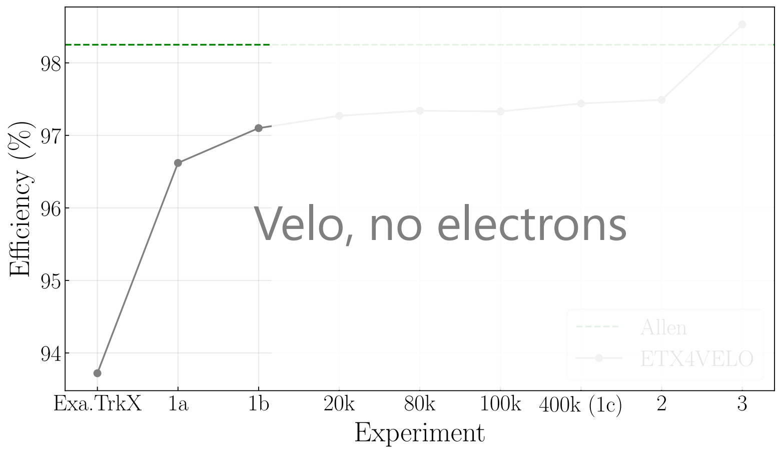
- **Problem**:
  - edges within the same layer
  - Bidirectional = # edges × 2
- **Solution**:
  - Connect hits in **adjacent layers**
  - Only consider edges in $+z$ direction

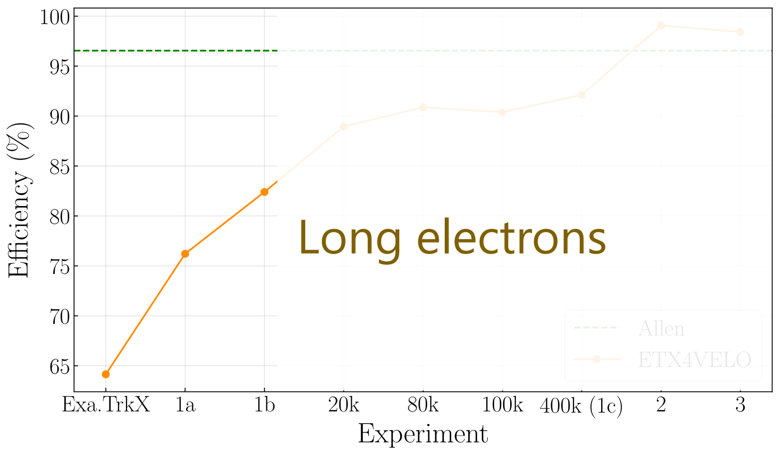


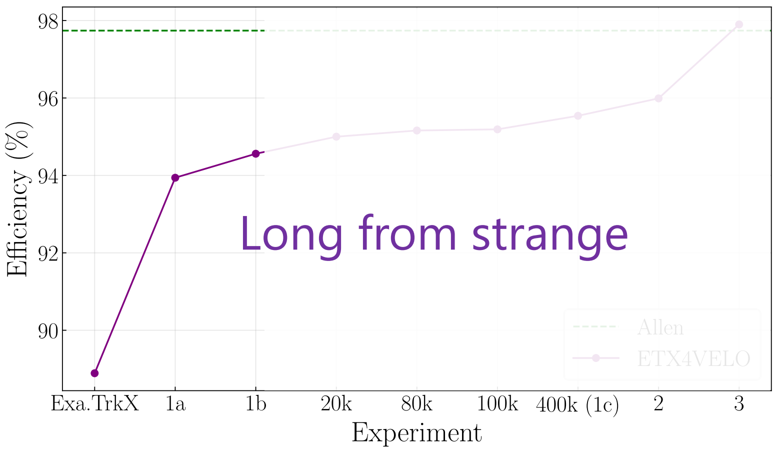
6 From Exa.TrkX to ETX4VELO

a Round 1b: True Edges





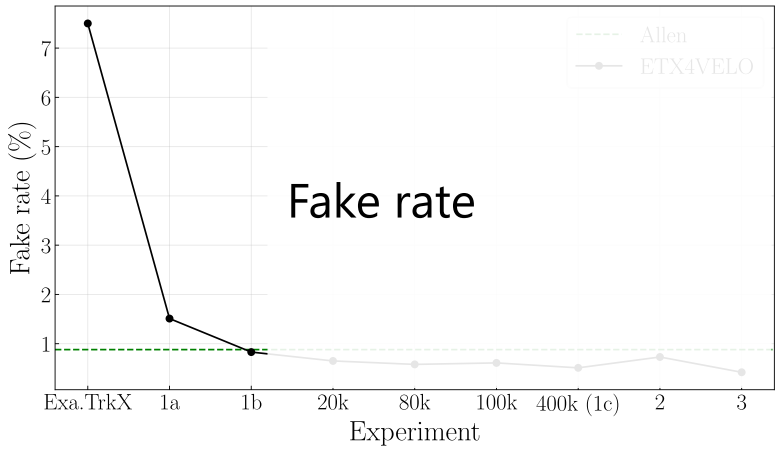


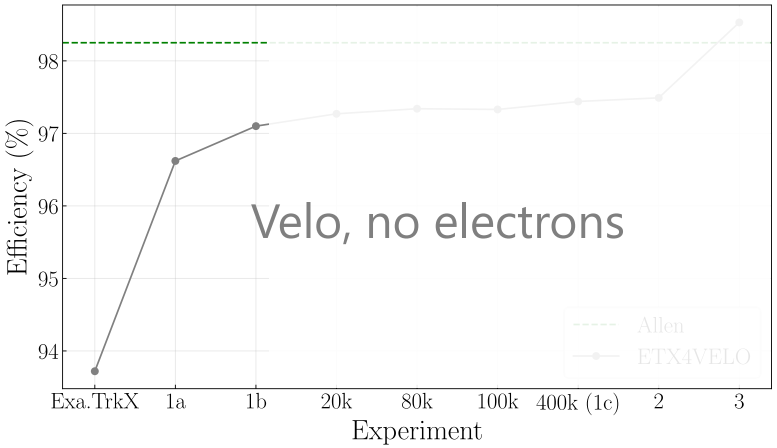


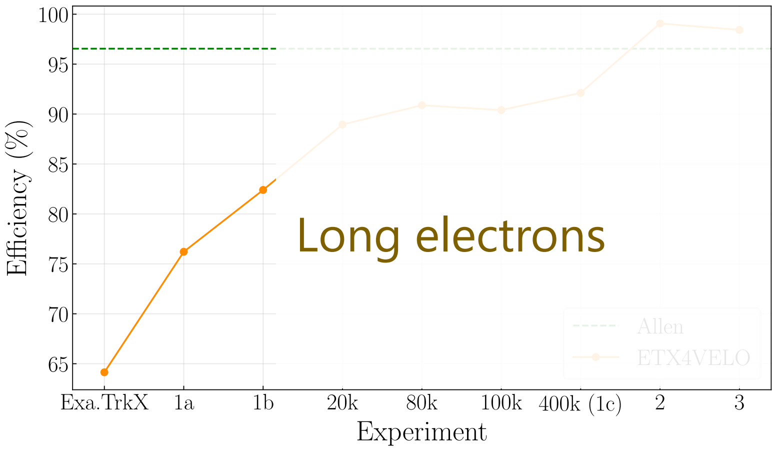
6 From Exa.TrkX to ETX4VELO

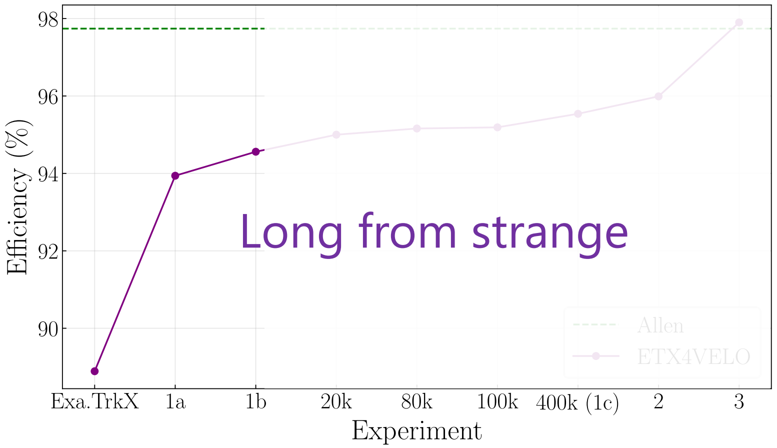
a Round 1b: True Edges

Fake rate will **never be a problem from now on**.









6 From Exa.TrkX to ETX4VELO

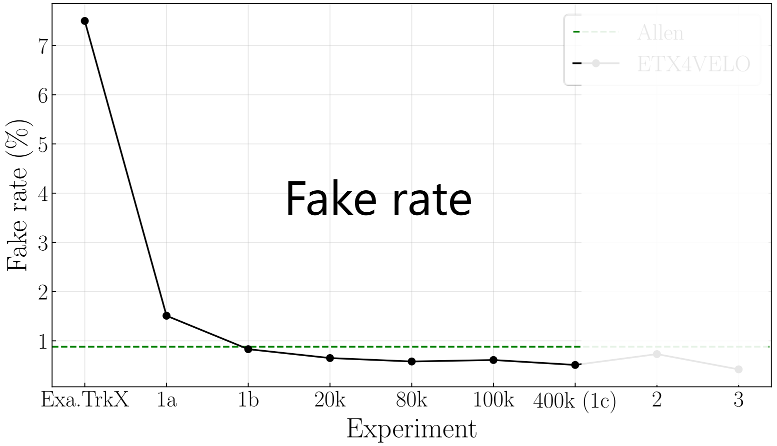
a Round 1c: Increase Training Dataset Size

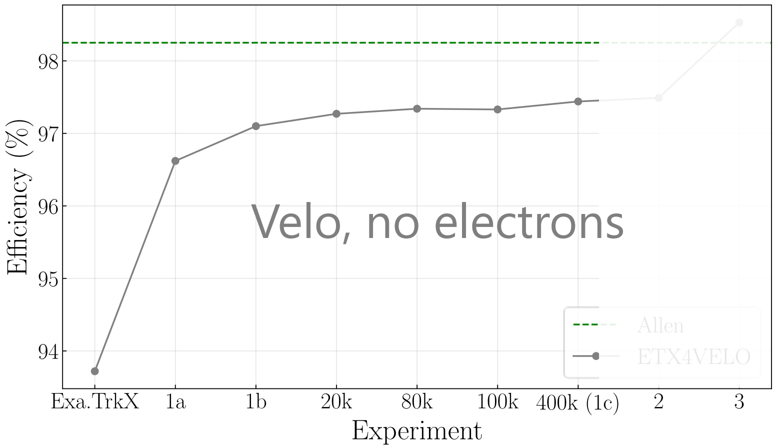
- Data available: **500k**
- Use as much data as possible! 10k → 400k

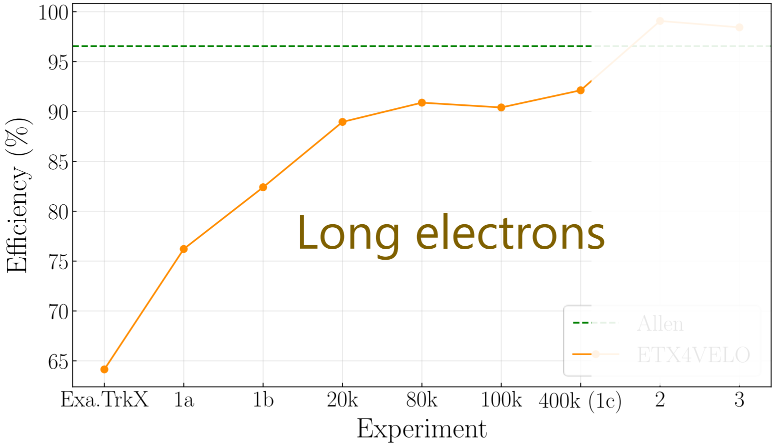
6 From Exa.TrkX to ETX4VELO

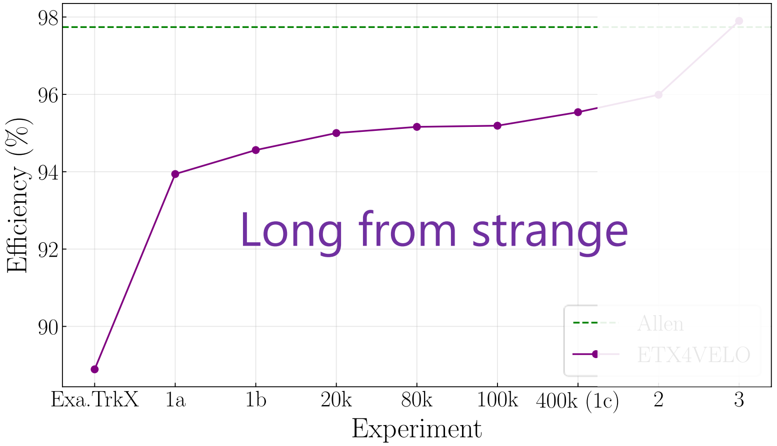
a Round 1c: Increase Training Dataset Size

- Data available: **500k**
- Use as much data as possible! 10k → 400k





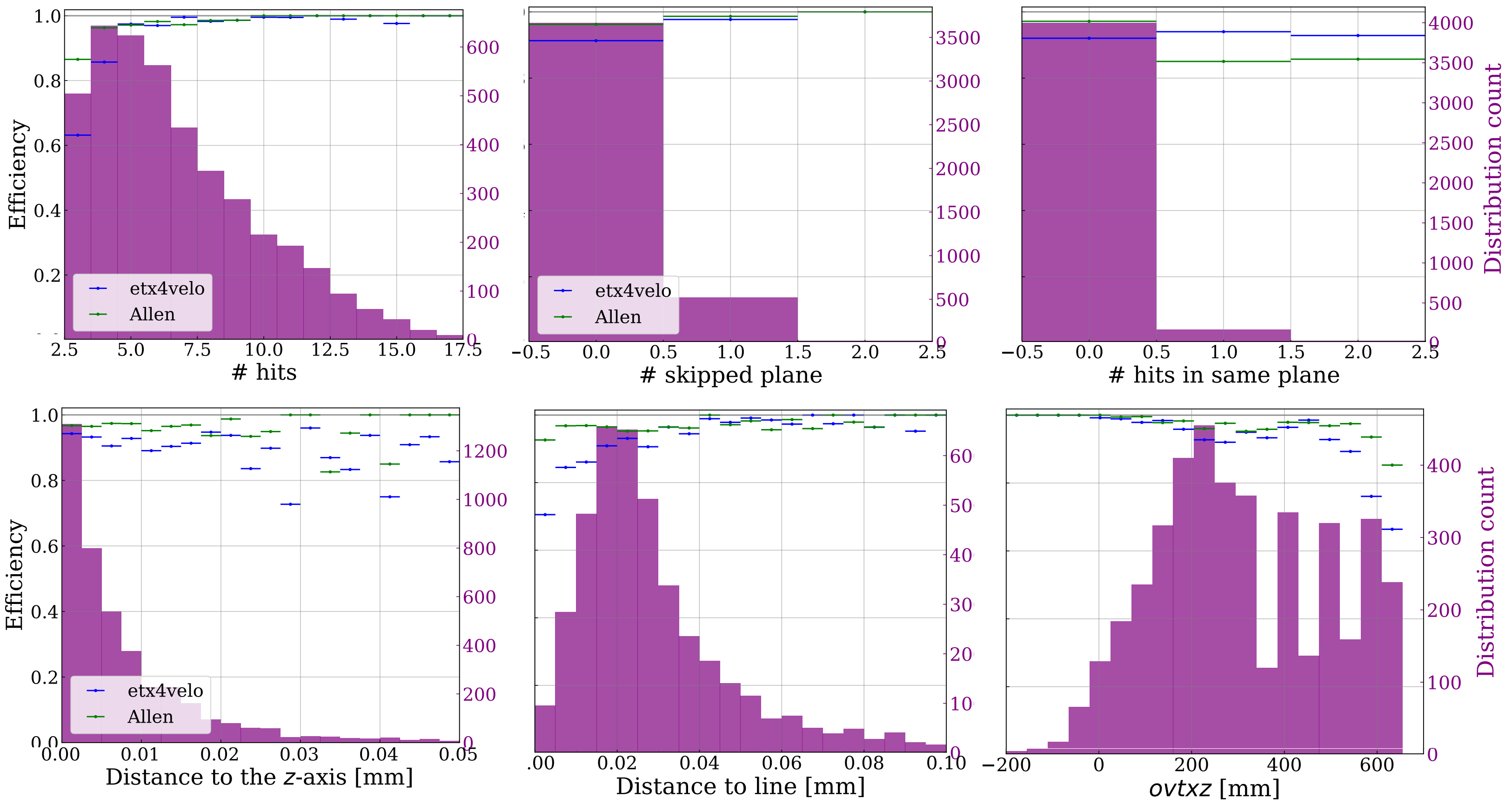




6 From Exa.TrkX to ETX4VELO

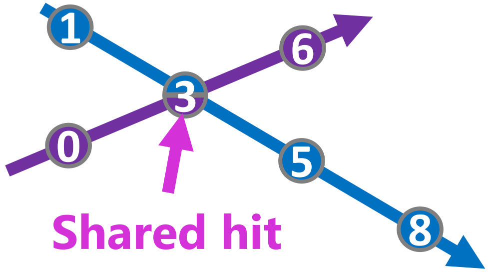
b Electron Problem

- Low performance on **long electrons**.
- Pipeline does not know about “electrons” → problem either
  - **Geometric**: angle, production vertex
  - **Hit-related**: # hits, skipped planes, etc.



6 From Exa.TrkX to ETX4VELO

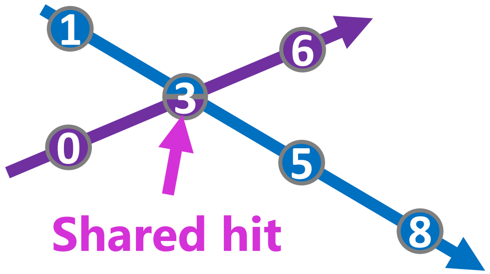
b Electron Problem



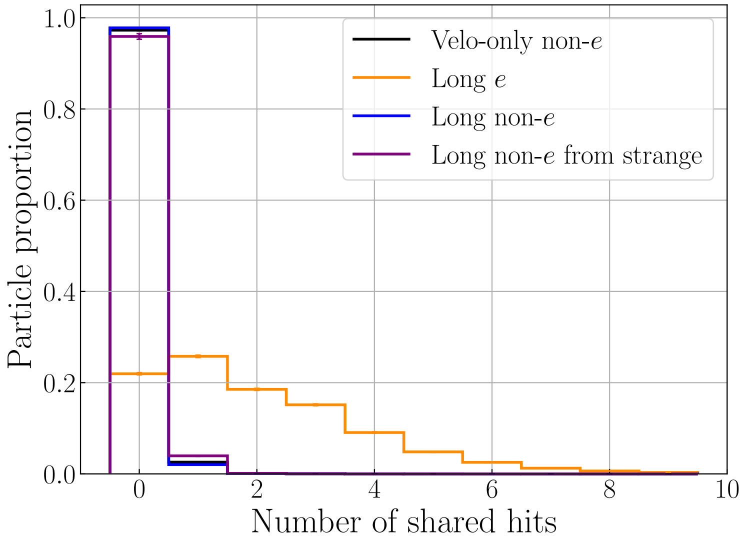
Shared hit: hit that belongs to > 1 reconstructible particle

6 From Exa.TrkX to ETX4VELO

b Electron Problem

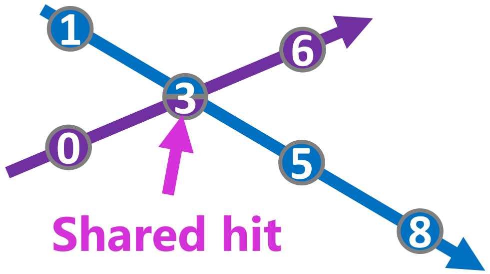


Shared hit: hit that belongs to > 1 reconstructible particle

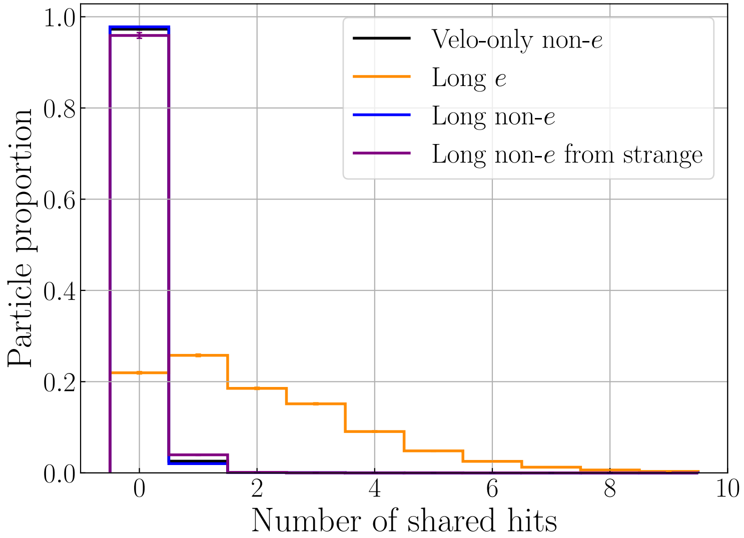


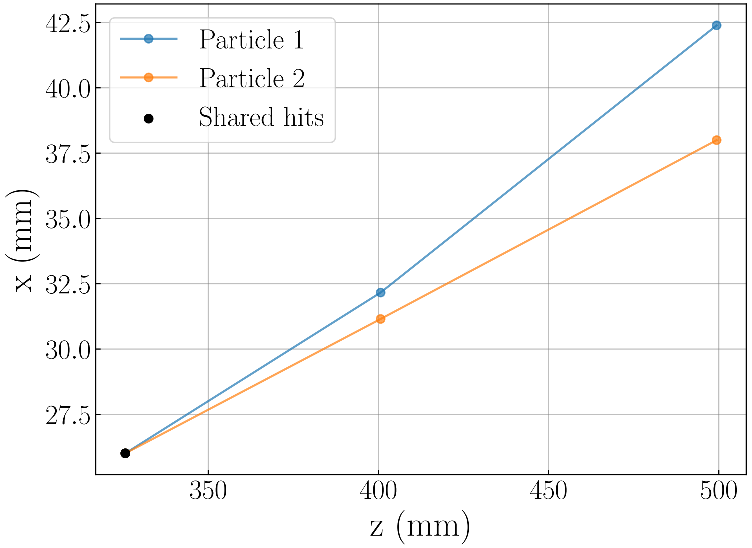
6 From Exa.TrkX to ETX4VELO

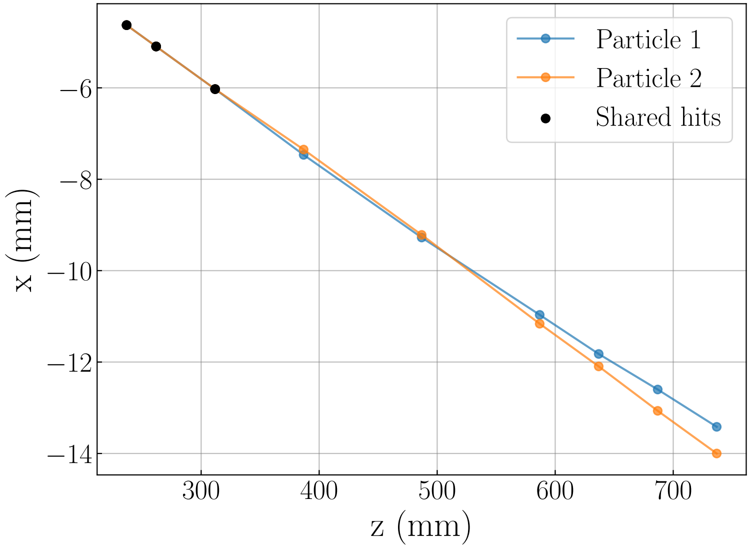
b Electron Problem



Shared hit: hit that belongs to > 1 reconstructible particle



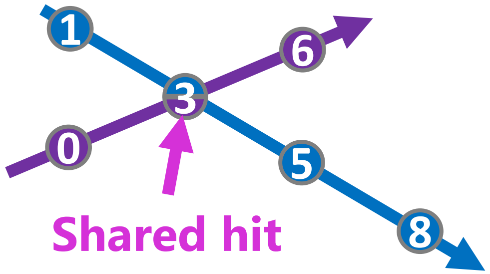




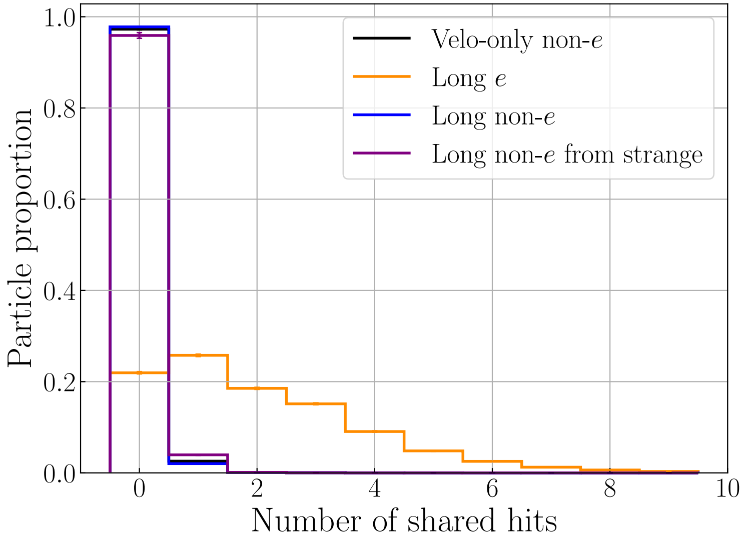
- Electrons have many **shared hits**
  - Typically $e^- e^+$ **pairs** sharing their **first hit(s)**
  - Correspond to **photon conversion in detector material $\gamma \rightarrow e^- e^+$**

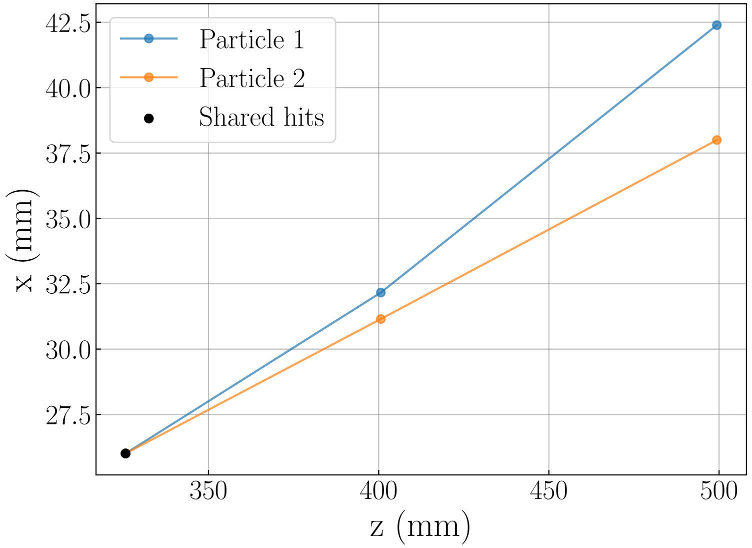
6 From Exa.TrkX to ETX4VELO

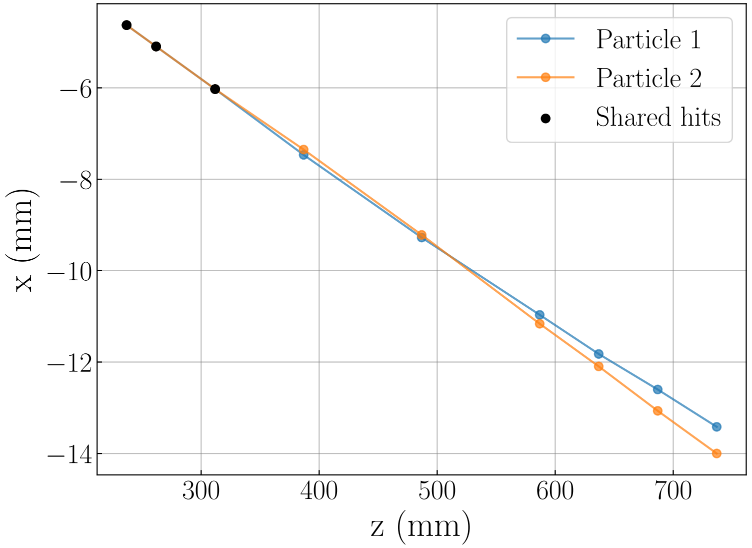
b Electron Problem



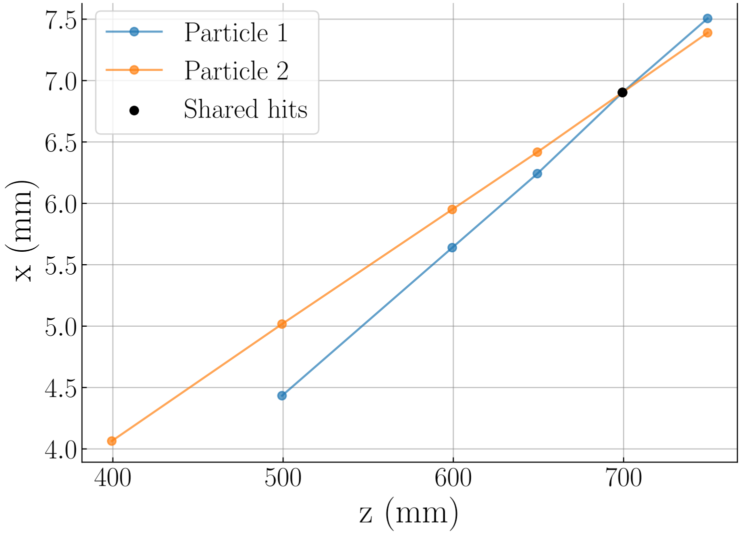
Shared hit: hit that belongs to > 1 reconstructible particle







- Electrons have many **shared hits**
  - Typically e^-e^+ **pairs** sharing their **first hit(s)**
  - Correspond to **photon conversion in detector material** $\gamma \rightarrow e^-e^+$

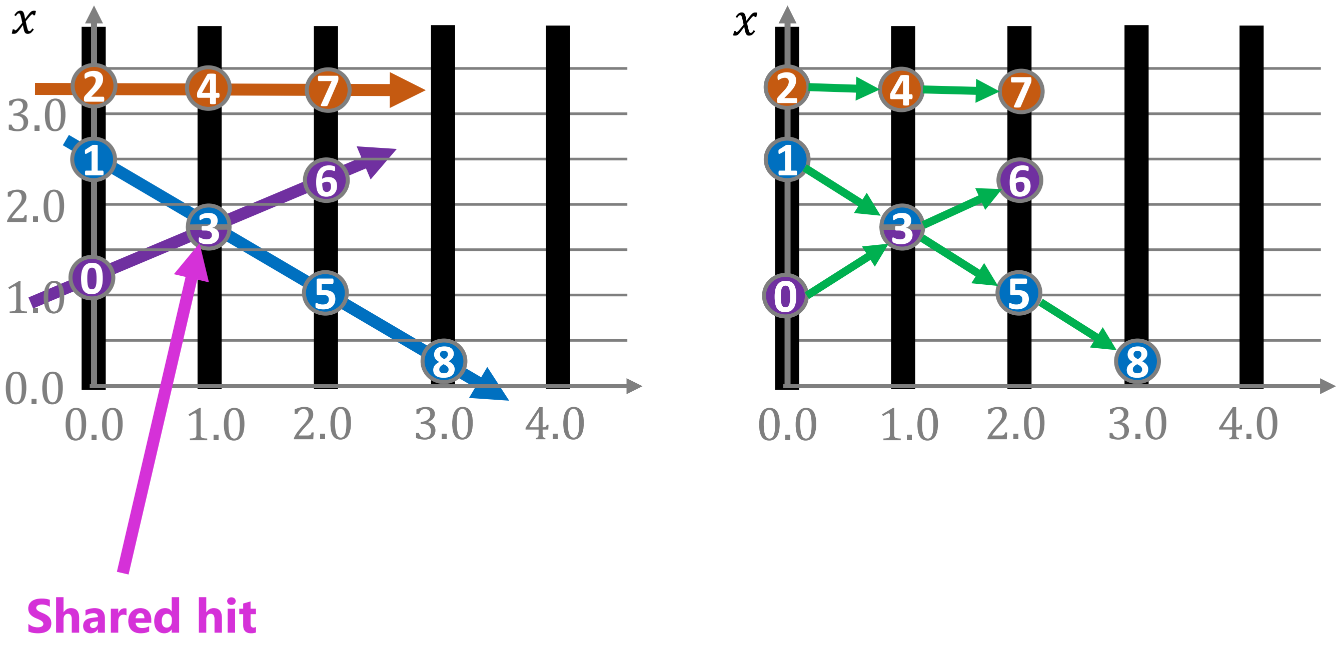


- Other particle categories have shared hits too!
  - e.g., particle crossing

6 From Exa.TrkX to ETX4VELO

b Shared Hit Problem

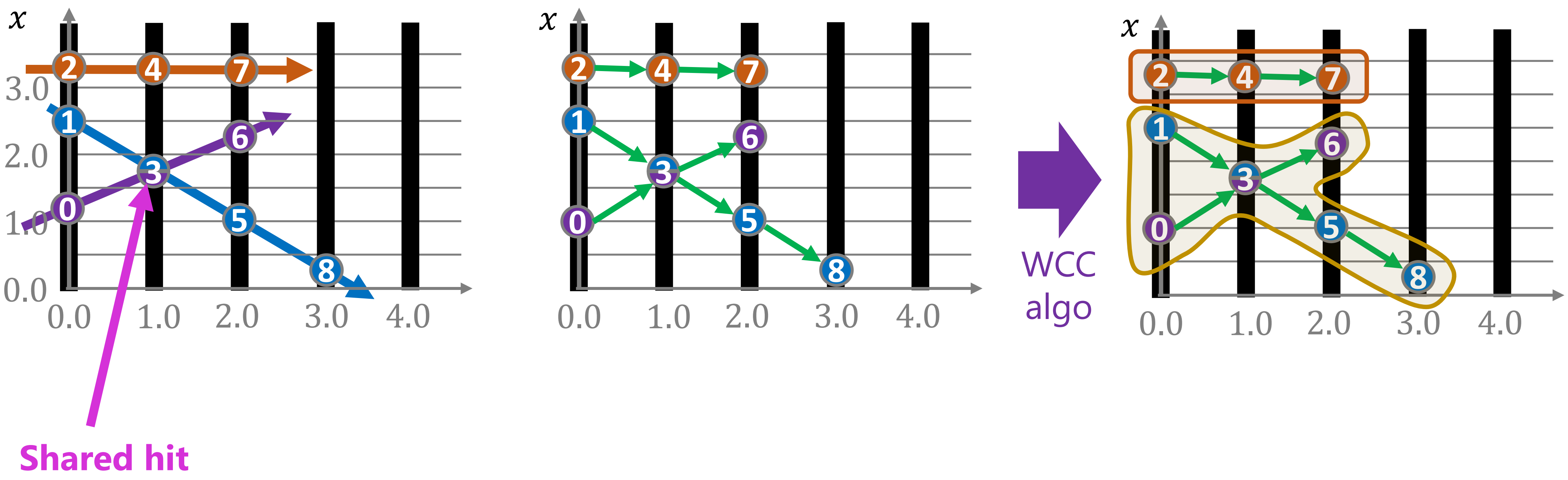
- Example of two particles crossing



6 From Exa.TrkX to ETX4VELO

b Shared Hit Problem

- Example of two particles crossing



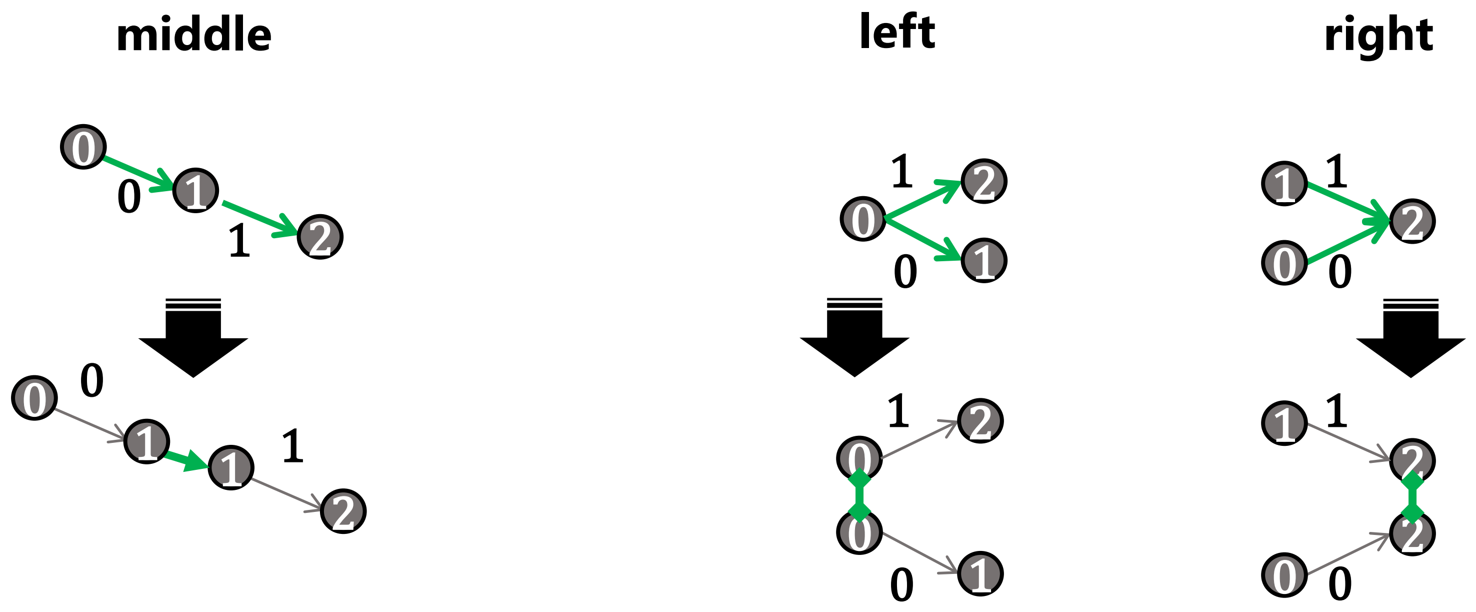
- Shared hits belong to same WCC → **two particles get merged**

6 From Exa.TrkX to ETX4VELO

b Round 2: Handle Shared Hits

- **Line graph**:
  - **Nodes** = edges of the hit graph
  - **Edges** = edge-edge connection of the hit graph

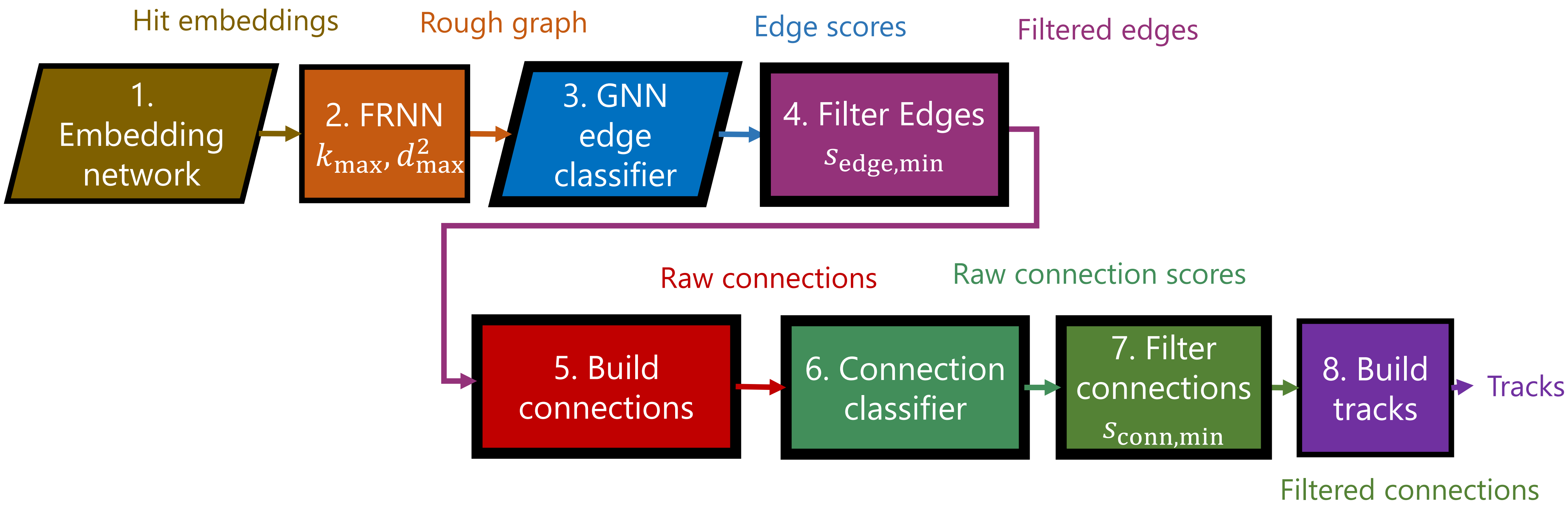
- 3 kind of **edge-edge connections**



- **Allow to solve all kinds of shared hits**

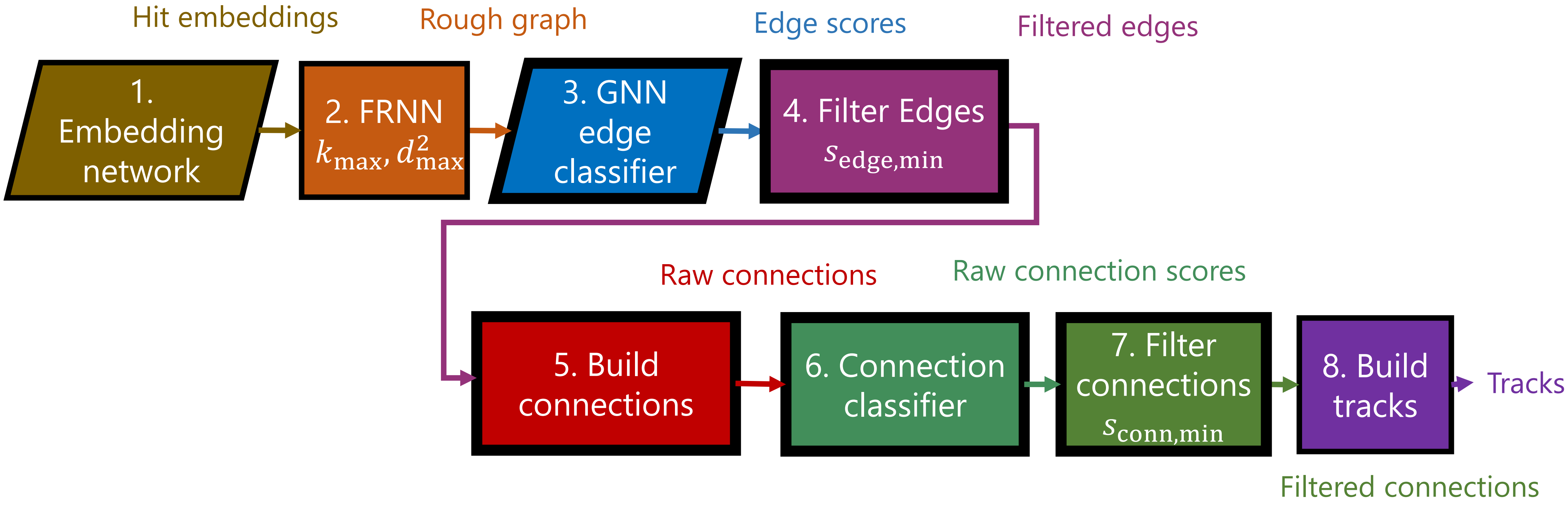
6 From Exa.TrkX to ETX4VELO

b Round 2: Handle Shared Hits



6 From Exa.TrkX to ETX4VELO

b Round 2: Handle Shared Hits

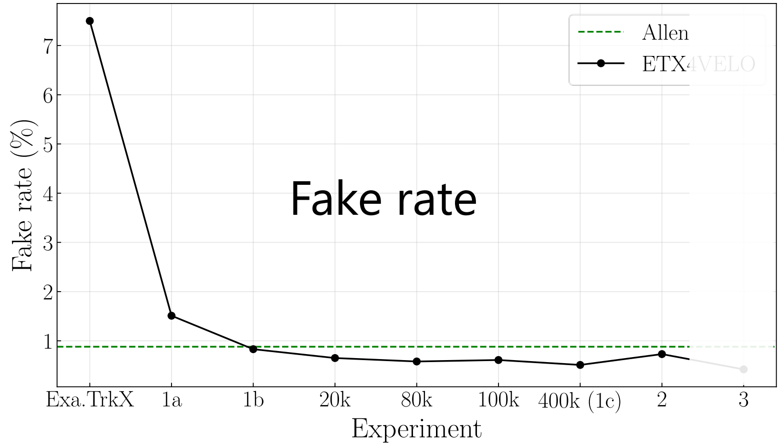


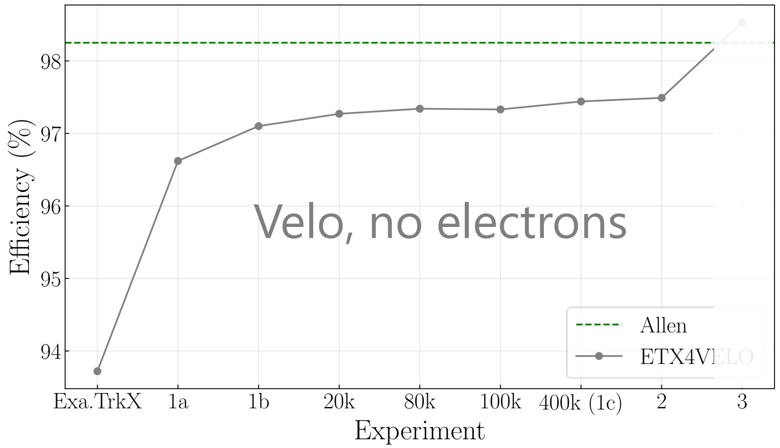
Classify edge-edge connections using **same GNN**!

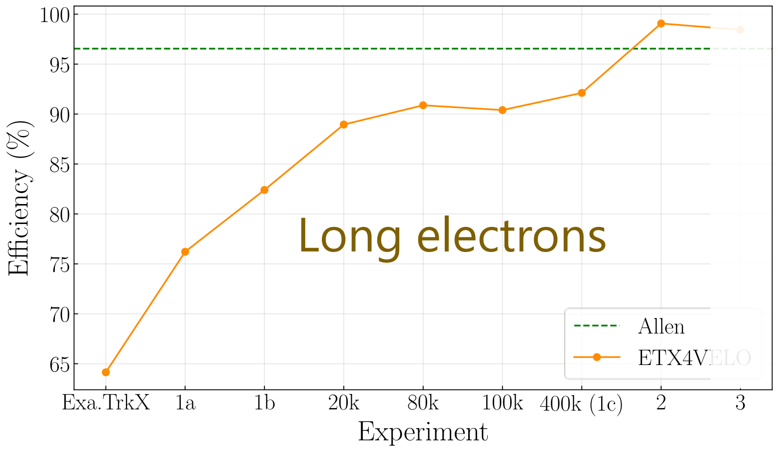
6 From Exa.TrkX to ETX4VELO

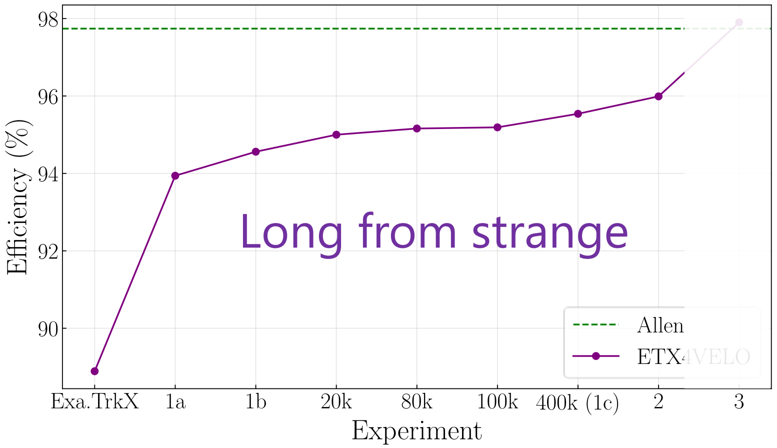
b Round 2: Handle Shared Hits

- Problem of "long electrons" **immediately solved**!
- From there on, **long electrons will never be a problem**.





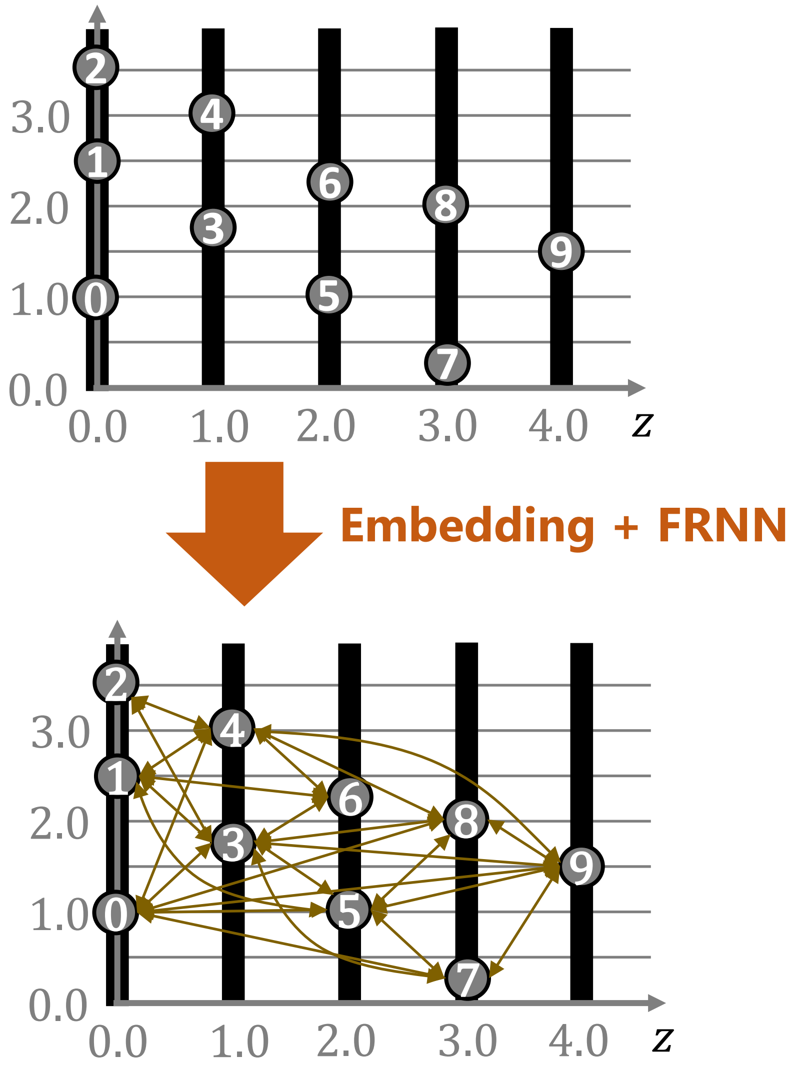




6 From Exa.TrkX to ETX4VELO

C Round 3: FRNN Layer by Layer

- FRNN applied to the whole space
  - Edges within the same layer
  - 20% edges with > 2-plane gap
  - **We always ask “is hit in layer 0 connected to hit in layer 25?”**



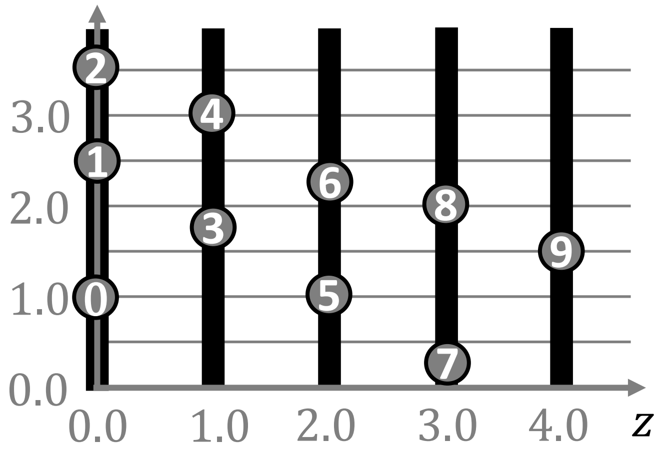
6 From Exa.TrkX to ETX4VELO

C Round 3: FRNN Layer by Layer

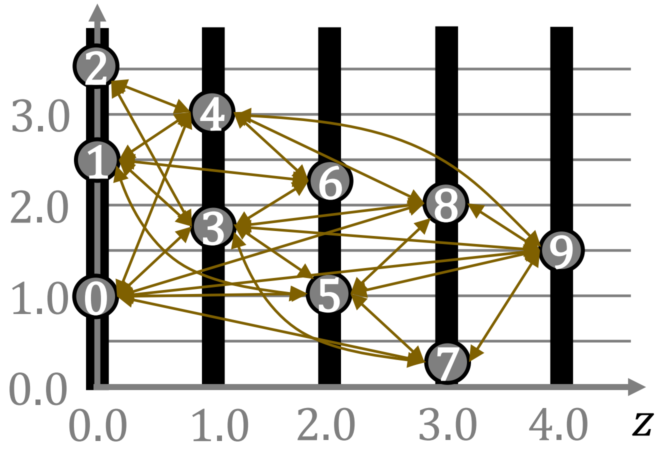
- FRNN applied to the whole space
  - Edges within the same layer
  - 20% edges with > 2-plane gap
  - **We always ask “is hit in layer 0 connected to hit in layer 25?”**
- New approach: **FRNN Layer by Layer**

Apply FRNN from layer i to layer $i+1$ and $i+2$
for $i \in \{0, \dots, n_{\text{layers}} - 1\}$

- Parallelizable over layers



Embedding + FRNN

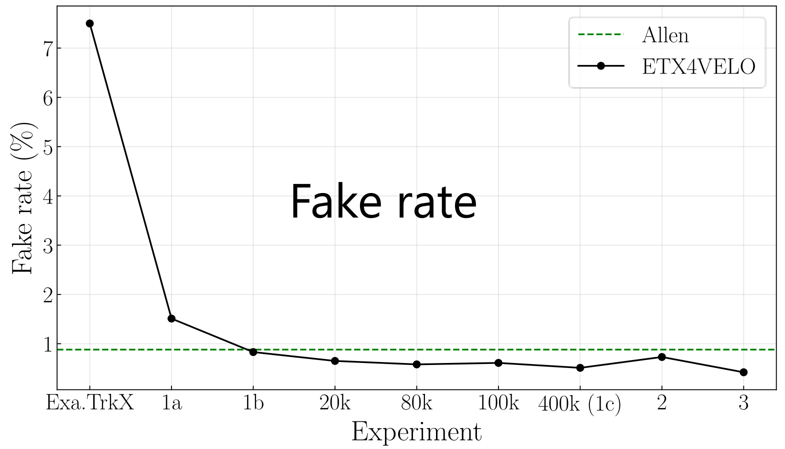


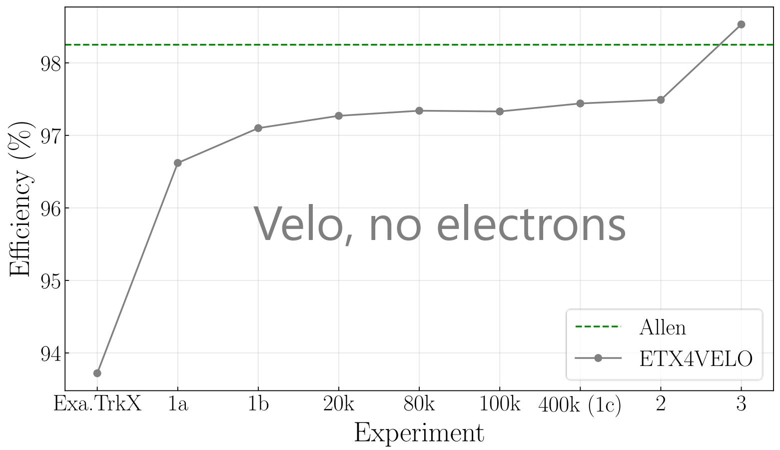
6 From Exa.TrkX to ETX4VELO

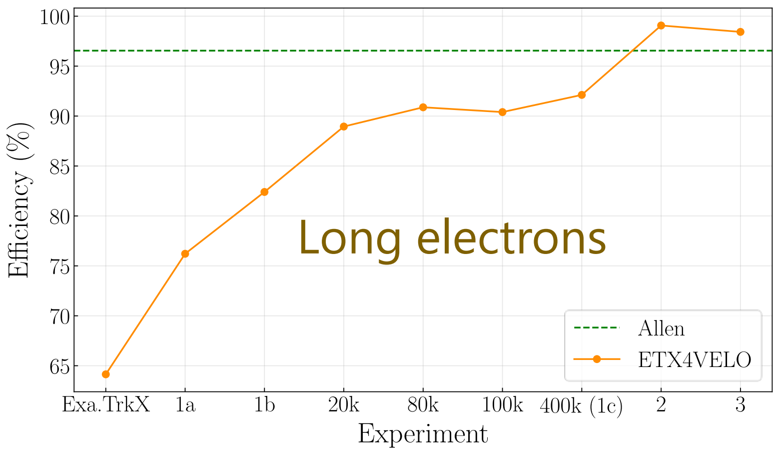
c Round 3: FRNN Layer by Layer

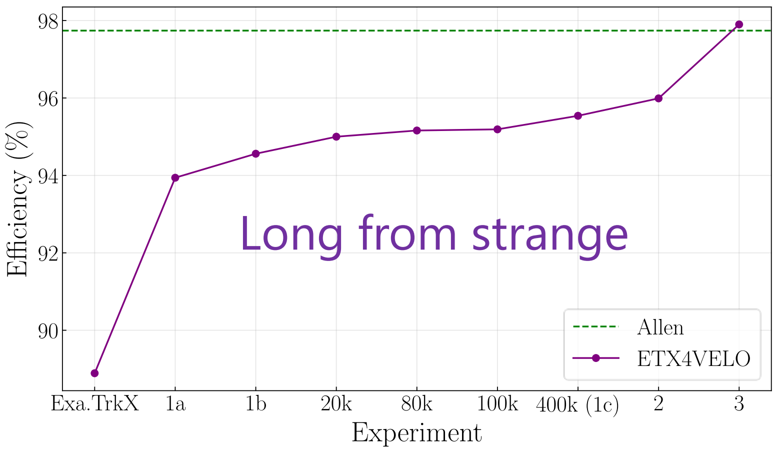
- **Performance better *everywhere***

Metric	Category	Allen	ETX4VELO
Efficiency	Velo no electrons	98.25%	98.53%
	Long electrons	96.55%	98.43%
	Long from strange	97.74%	97.90%
Fake rate		0.88%	0.42%







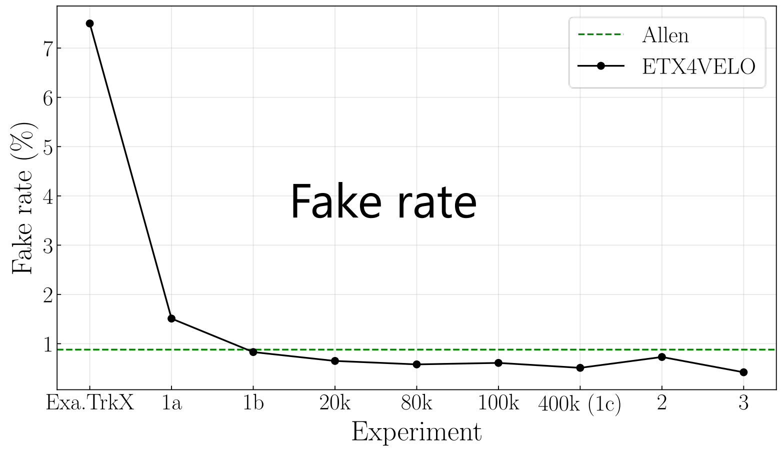


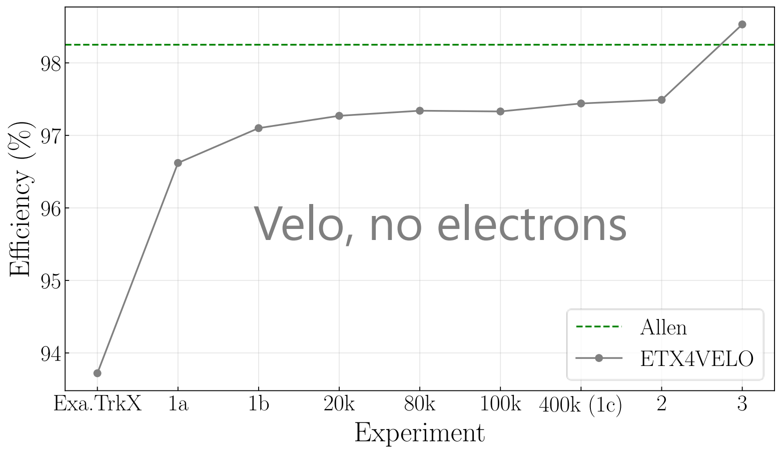
6 From Exa.TrkX to ETX4VELO

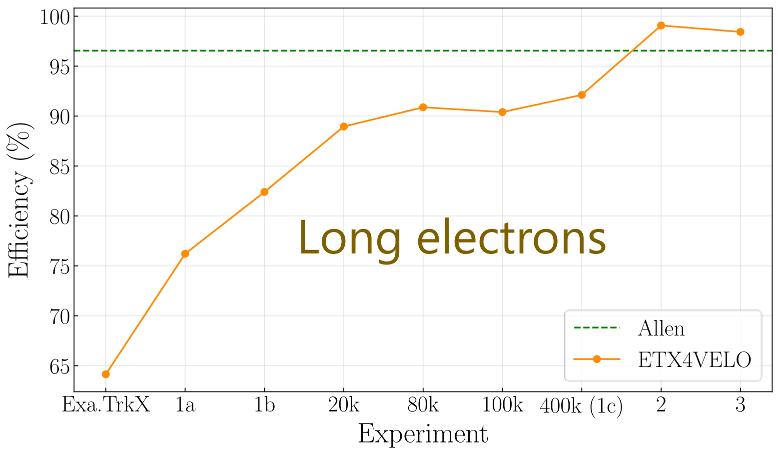
C Round 3: FRNN Layer by Layer

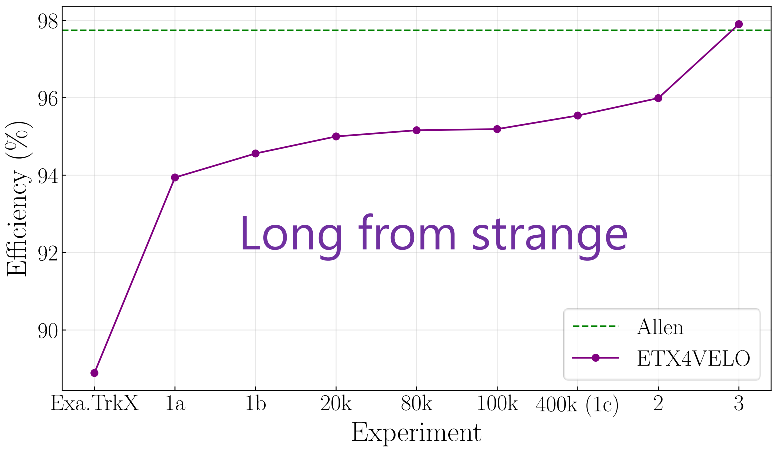
- **Performance better *everywhere***

Metric	Category	Allen	ETX4VELO
Efficiency	Velo no electrons	98.25%	98.53%
	Long electrons	96.55%	98.43%
	Long from strange	97.74%	97.90%
Fake rate		0.88%	0.42%
Throughput		**595 kHz**	?









1 Beginner Introduction

2 Neural Network Introduction

3 Problem Formulation

4 Experimental Setup

5 Exa.TrkX Pipeline

6 From Exa.TrkX to ETX4VELO

7 Implementation in Allen

8 Optimization

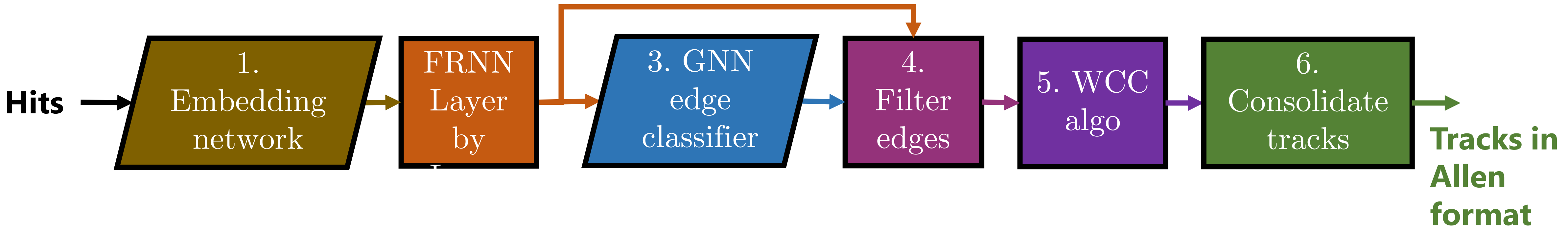
7 Implementation in Allen

- a **Pipeline**
- b **Inference Engine**
- c **Results**

7 Implementation in Allen

a Pipeline

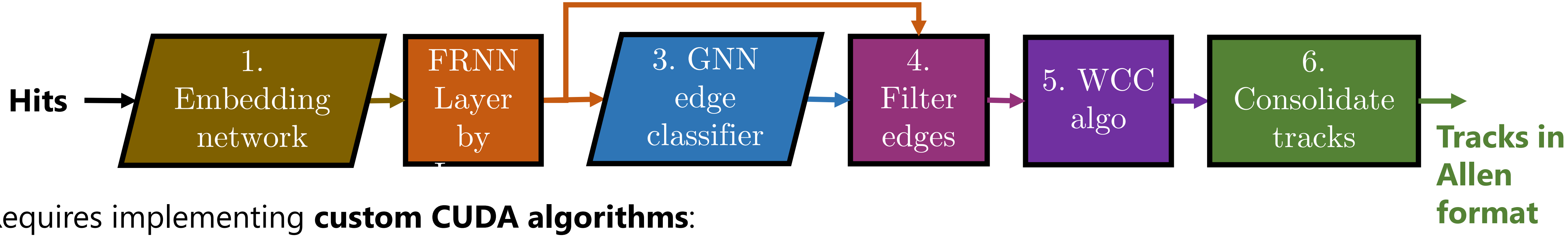
- Edge-edge connections not yet implemented in Allen
  ⇒ **Evaluate throughput up to WCC on hit graph**
  → Only **upperbound** throughput measurement
- Supplementary "Consolidate tracks" step



7 Implementation in Allen

a Pipeline

- Edge-edge connections not yet implemented in Allen
  ⇒ **Evaluate throughput up to WCC on hit graph**
  → Only **upperbound** throughput measurement
- Supplementary "Consolidate tracks" step



- Requires implementing **custom CUDA algorithms**:
  - **FRNN Layer by Layer**: `apply_frnn_plane_by_plane, consolidate_target_edges, build_source_edges`
  - **Filter Edges**: `mask_edges, filter_edge_offsets, filter_edges, build_edge_sources`
  - **WCC algorithm**: `count_edges_per_target_hit, build_invert_edge_targets, apply_wcc`
  - **Consolidate tracks**: `cound_hits_per_label, compute_track_offsets, build_tracks, consolidate_tracks`

- **Require inferring Neural Networks** (in C++/CUDA)
  - **Embedding**
  - **GNN**

7 Implementation in Allen

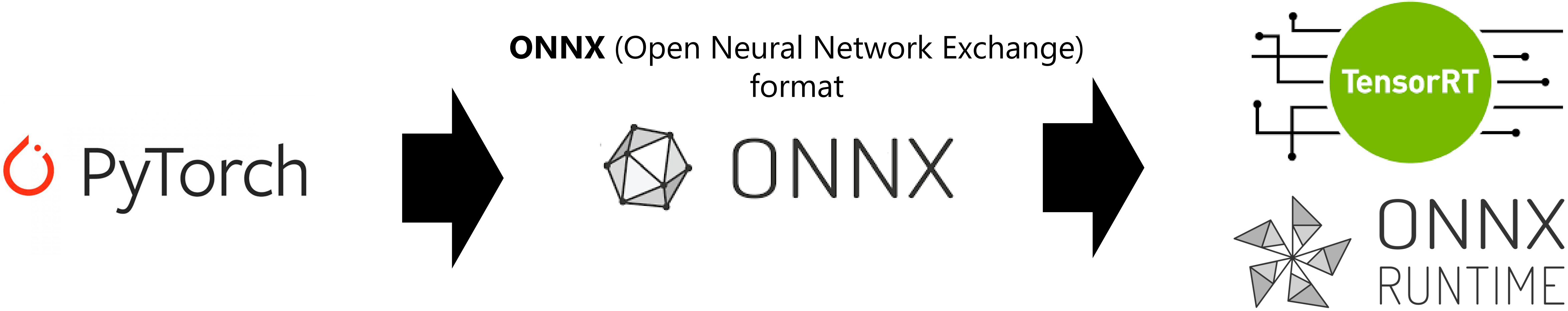
b Inference Engine

How to infer a neural network in C++/CUDA?

- Re-implement everything from scratch
- Use **LibTorch** (C++ API of Torch)
- Use an **inference engine**

Inference Engine on GPU: **ONNX Runtime** and **TensorRT (NVIDIA)**

ONNX (Open Neural Network Exchange) format



`torch.onnx.export`

7 Implementation in Allen

b Inference Engine

- **TensorRT** > **ONNX Runtime** for deployment on **NVIDIA GPU**

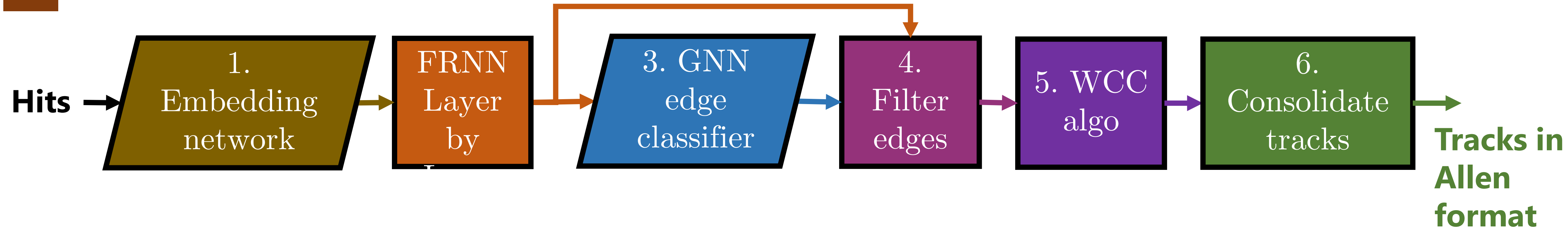
	ONNX Runtime	TensorRT
Open source	Yes	Only a small subset
CPU Support	Yes Different *execution providers*: CPU, CUDA, TensorRT, ROCm, etc.	No
Memory manageable by Allen	No	Yes, a pointer can be passed to TensorRT
Memory released after each inference	Too slow to release	Yes → can be re-used by other algorithms
Documentation	Worse	Better

- Implementation of **`scatter_add` operation**:
  - **ONNX Runtime**: already implemented
  - **TensorRT**: I implemented a TensorRT plugin

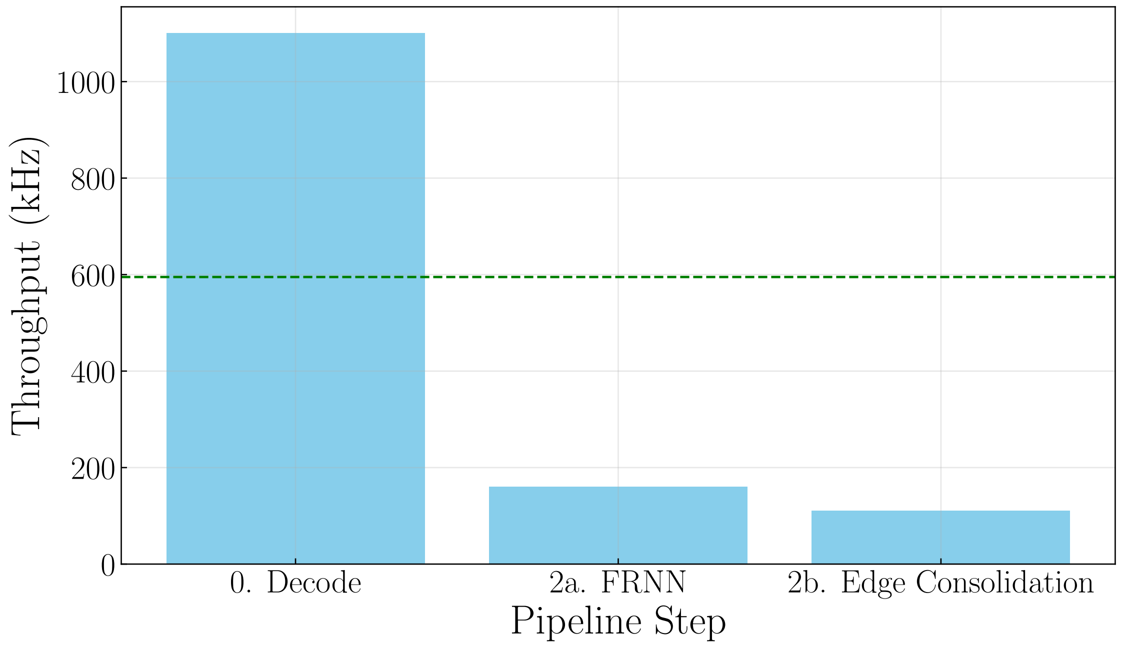
- Throughput reported **with TensorRT by default**

7 Implementation in Allen

C Results

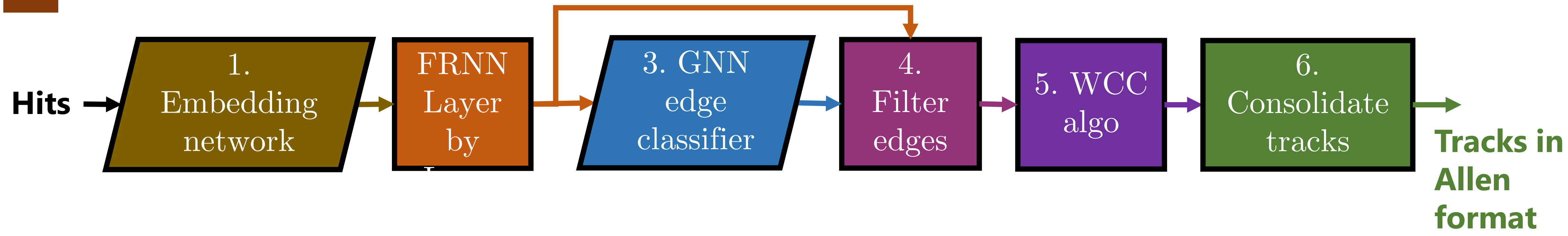


Step	Throughput
0. Decode velo hits	1100 kHz
1. Embedding Network	?
2a. FRNN	160 kHz
2b. Edge Consolidation	110 kHz
3. GNN Edge Classifier	?
4. Filter Edges 5. WCC Algo 6. Track Consolidation	*Able to follow*
Allen	***595 kHz***

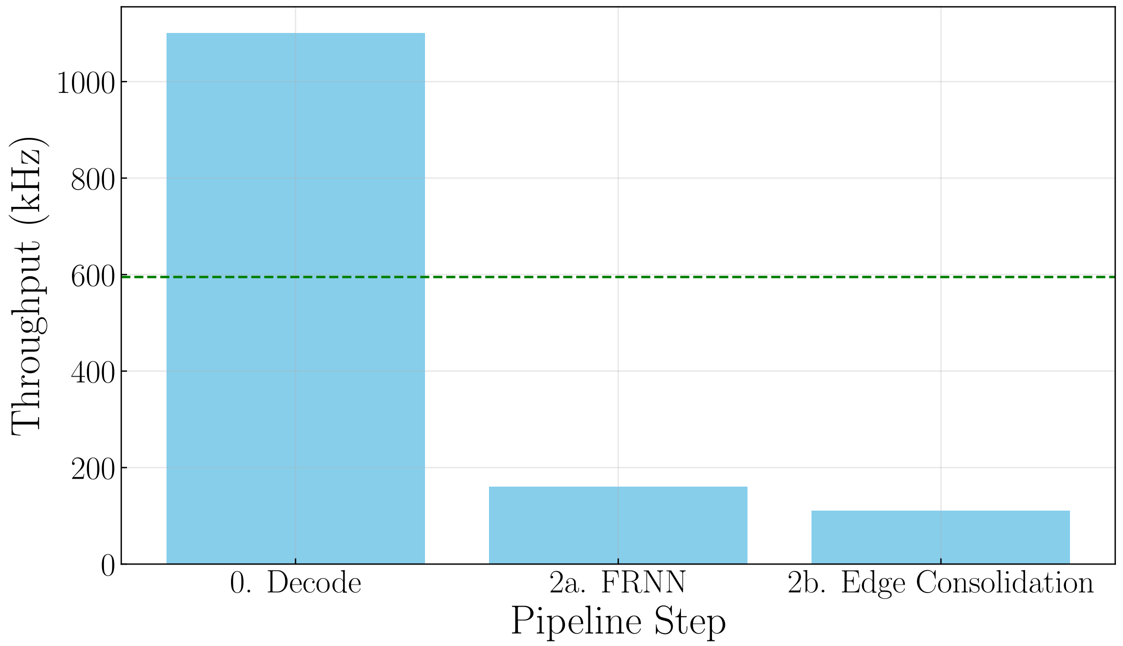


7 Implementation in Allen

C Results



Step	Throughput
0. Decode velo hits	1100 kHz
1. Embedding Network	? **< 38 kHz**
2a. FRNN	160 kHz
2b. Edge Consolidation	110 kHz
3. GNN Edge Classifier	? **< 0.026 kHz**
4. Filter Edges 5. WCC Algo 6. Track Consolidation	*Able to follow*
Allen	***595 kHz***



1 Beginner Introduction

2 Neural Network Introduction

3 Problem Formulation

4 Experimental Setup

5 Exa.TrkX Pipeline

6 From Exa.TrkX to ETX4VELO

7 Implementation in Allen

8 Optimization

8 Optimisation

a **Optimising the Embedding Network**

b **Optimising the GNN**

c **TensorRT vs ONNX Runtime**

d **Final Performance**

8 Optimisation

a Embedding Network

Changes:

- Reduce # parameters of **embedding network** from **35k** down to **251 parameters**
- Train on **reconstructible particles with $|\eta| \in [2, 5]$**

Metric	Allen	Before	Now
Embedding throughput (events/second)	595k	**< 38k**	**330k**

Better physics performance and **better throughput**

8 Optimisation

a Embedding Network

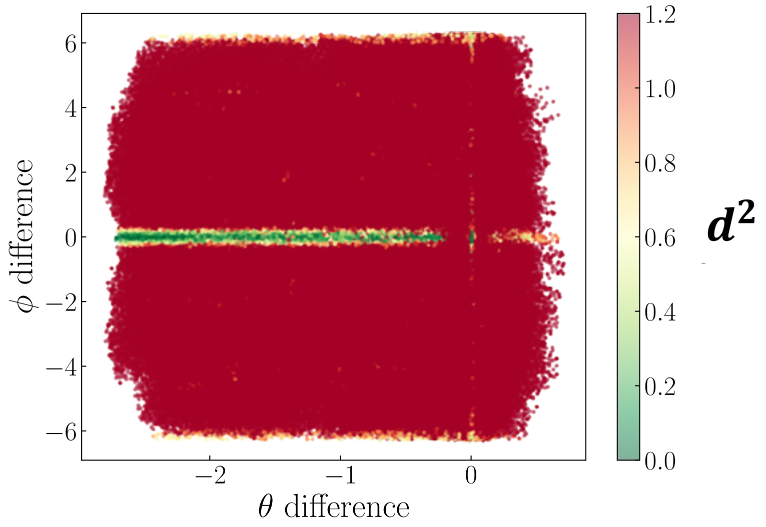
- Most tracks originate towards (0,0,0) ⇒ small angles:
  - **Polar angle $\boldsymbol{\theta}$**: angle w.r.t. z-axis
  - **Azymuthal angle $\boldsymbol{\phi}$**: angle around z-axis

Build all edge-edge candidates up to 2 planes apart and compare $(\Delta\theta, \Delta\phi)$ to

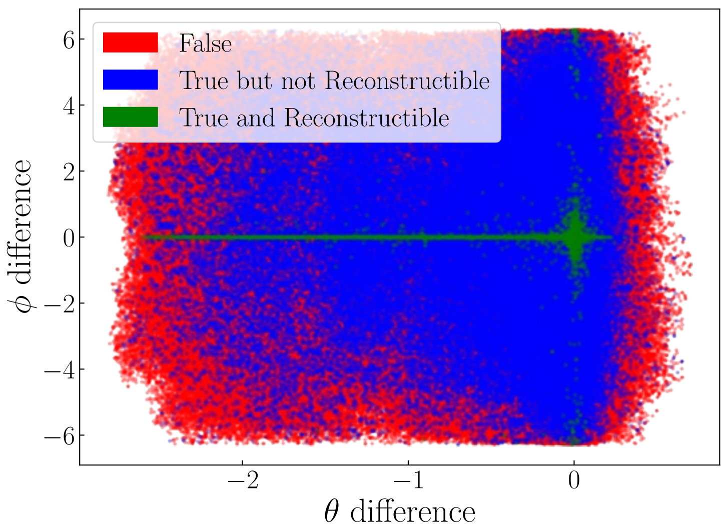
- d^2 from embedding network
- Truth

⇒ **clear correlation**

Embedding Network



Truth



8 Optimisation

b GNN

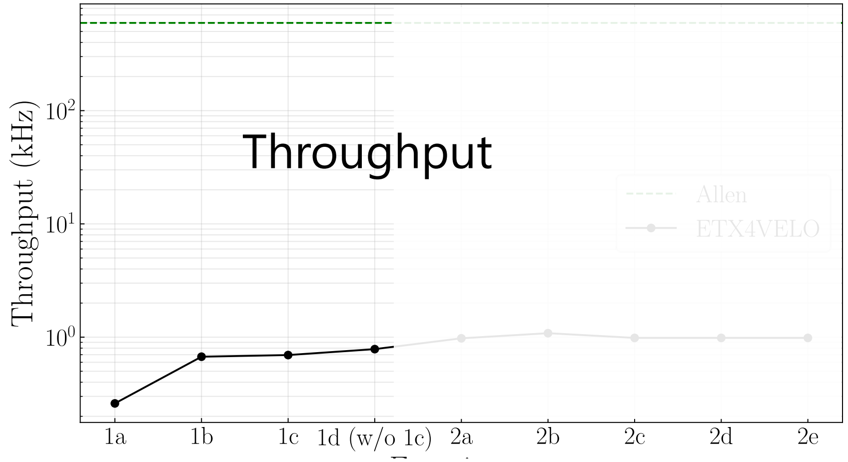
Step 1: Increase throughput by decreasing GNN size

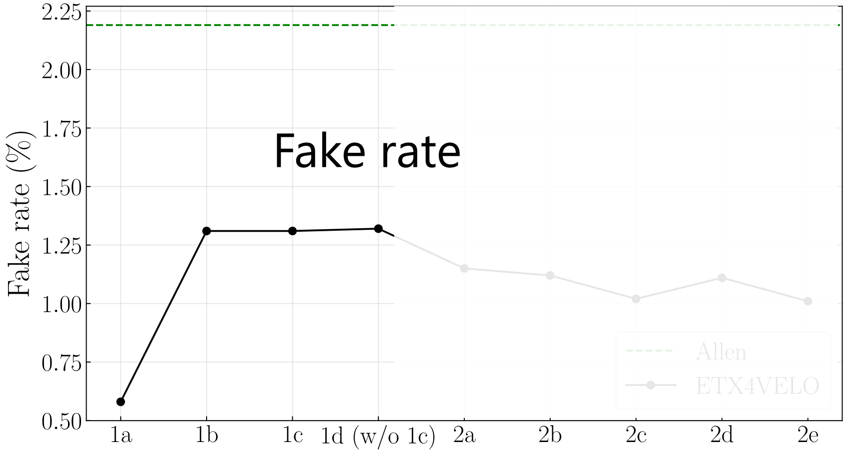
a) Removing `scatter_max`, **only use** `scatter_add`,

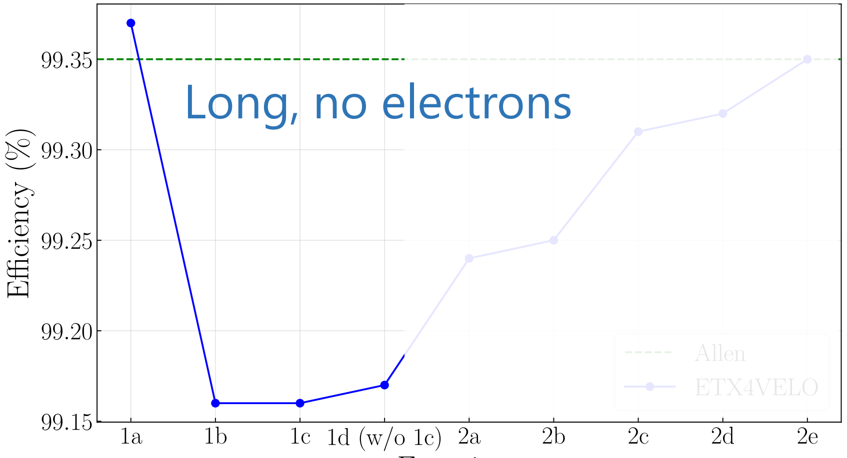
b) Decreasing **hit and edge encoding dimensions** from $h = 256$ to $h = 32$

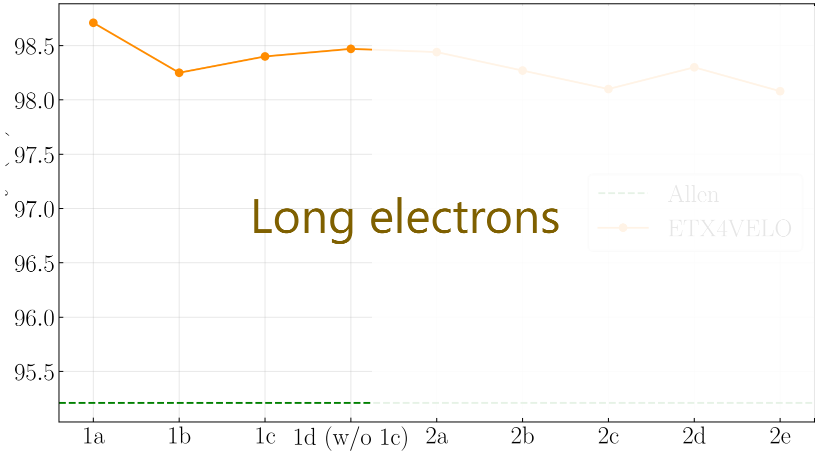
c) Use only **edge encodings** for classifications

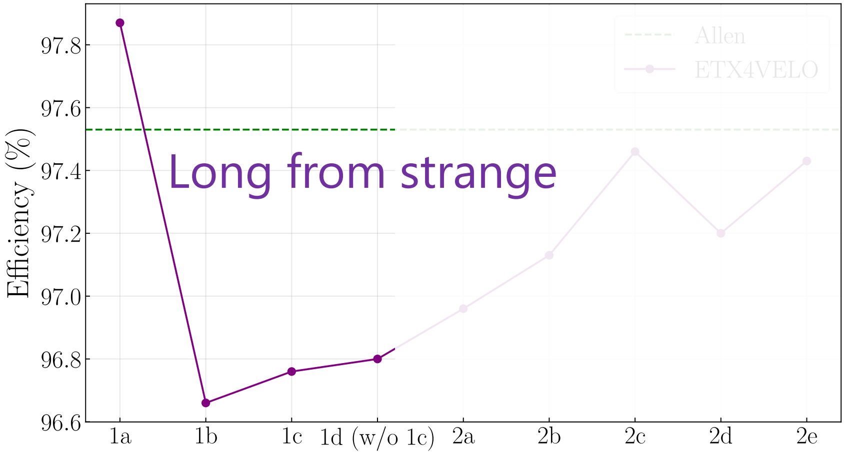
d) Decreasing **# graph iterations** from 6 to 5











8 Optimisation

b GNN

Step 1: Increase throughput by decreasing GNN size

a) Removing **scatter_max**, **only use scatter_add**,

b) Decreasing **hit and edge encoding dimensions** from $\boldsymbol{h = 256}$ to $\boldsymbol{h = 32}$

c) Use only **edge encodings** for classifications

d) Decreasing **# graph iterations** from **6** to **5**

Step 2: recover lost performance

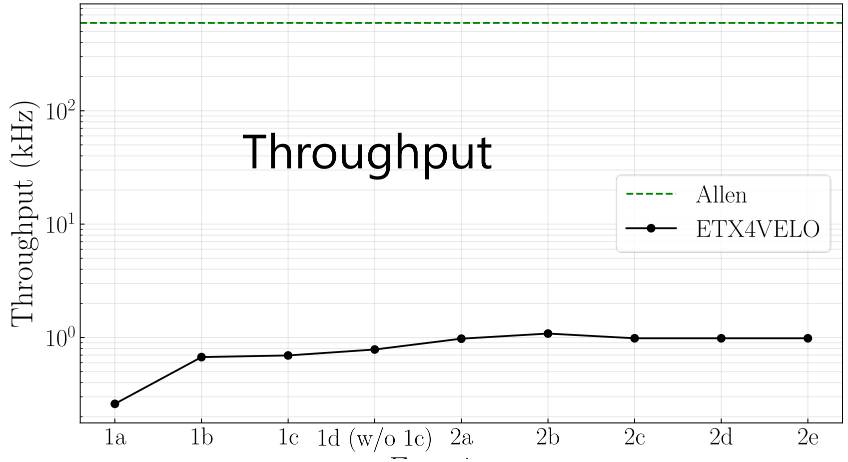
a) GNN **non-recursive**

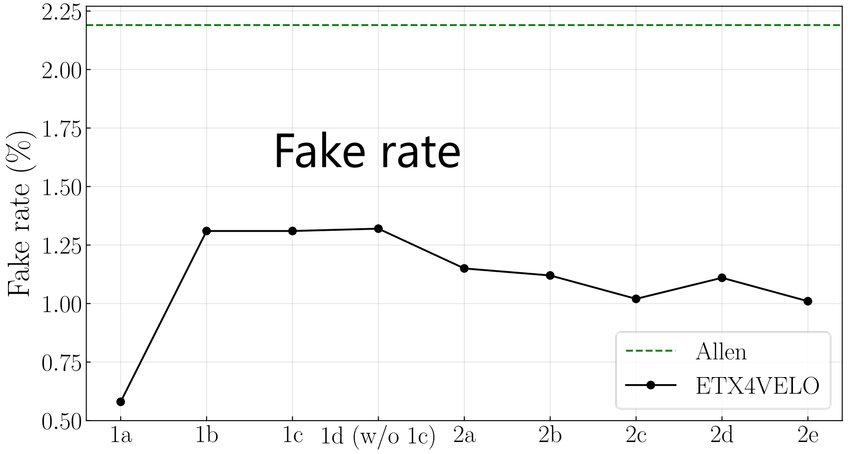
b) Use **cartesian coordinates** for input node features instead of **cylindrical**

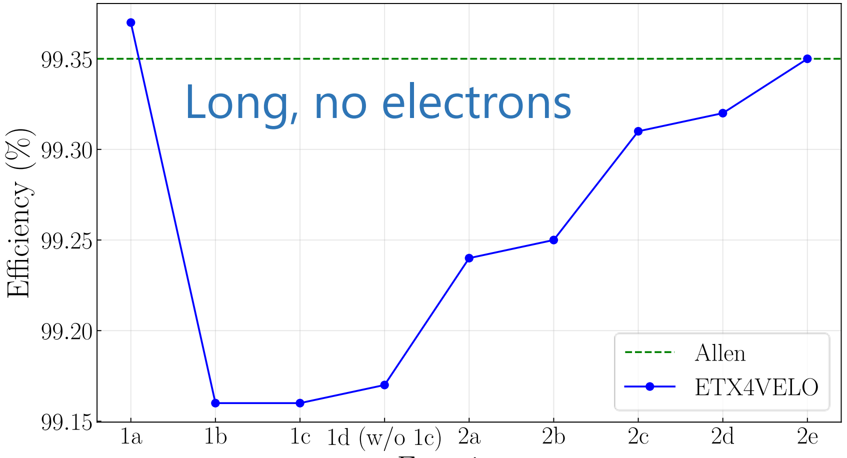
c) Use the **new embedding network** from previous slide

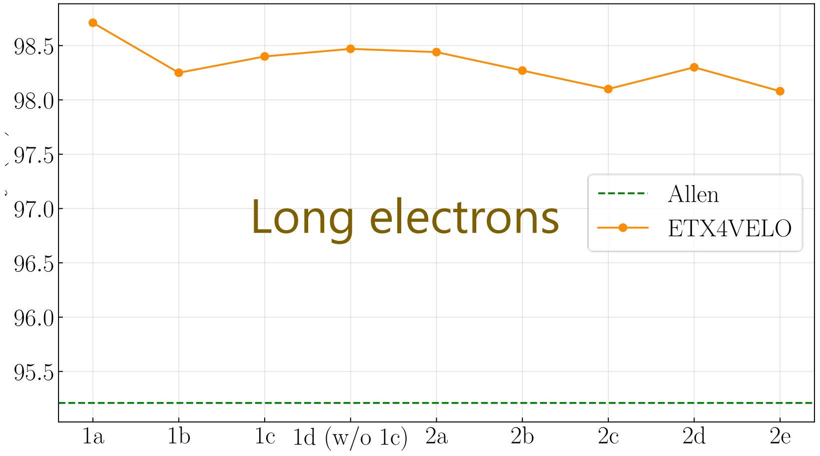
d) Do not **remove curved particles from training set**, but only **from the loss**;
Consider isolated edges as fake.

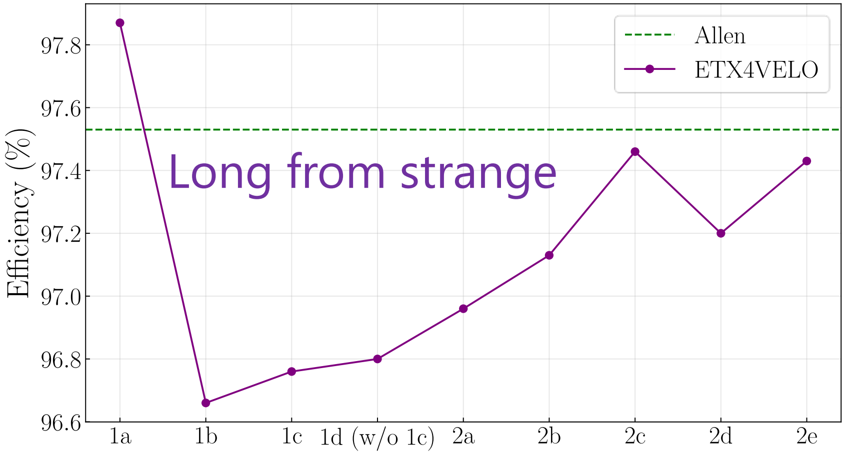
e) Use a **different classifier** for middle connections, & left/right connections











8 Optimisation

b GNN

Metric	**Category**	**Allen**	1a	**2e**
Efficiency	**Long**	99.35%	99.37%	**99.35%**
	Long from strange	97.53%	97.87%	**97.43%**
	Long electrons	95.21%	98.71%	**98.08%**
Fake rate		2.19%	0.58%	**1.01%**
GNN throughput (kHz)		**595**	0.026	0.985

× 38 in throughput

8 Optimisation

C TensorRT vs ONNX Runtime

TensorRT significantly faster than **ONNX Runtime** for both the embedding network and GNN.

Embedding network

Inference Engine	Throughput
ONNX Runtime	50 kHz
TensorRT	330 kHz

6.1 times faster

GNN

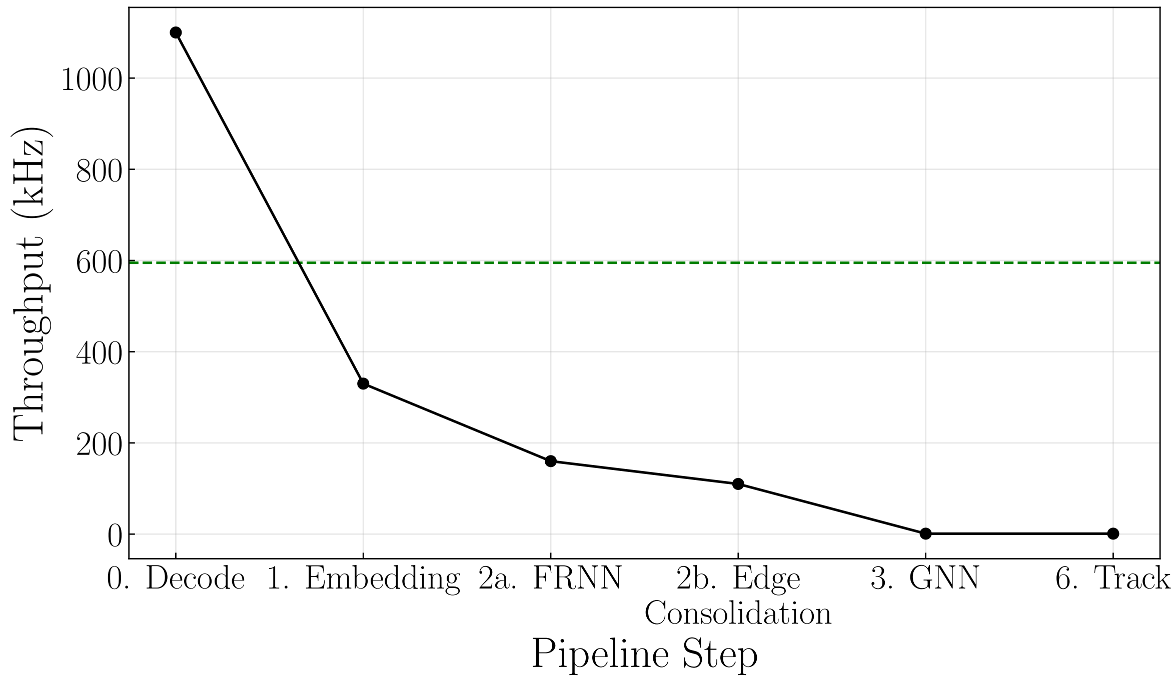
Inference Engine	Throughput
ONNX Runtime	0.307 kHz
TensorRT	1.004 kHz

3.3 times faster

8 Optimisation

d Final Performance

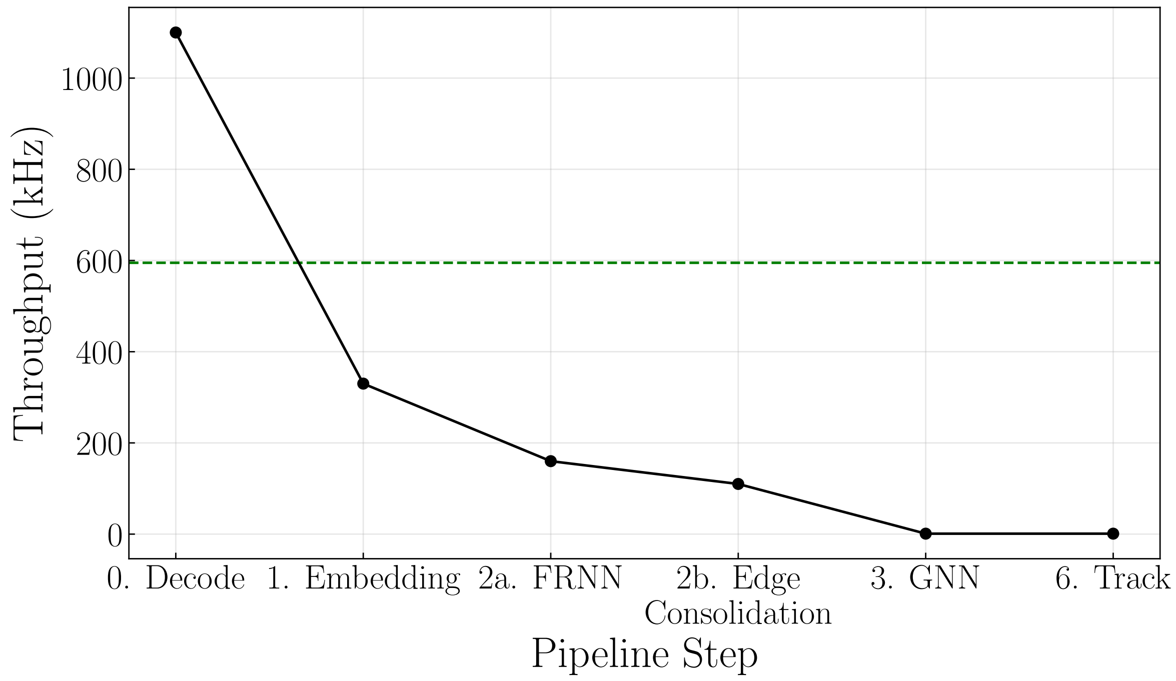
Step	Throughput
0. Decode velo hits	1100 kHz
1. Embedding Network	330 kHz
2a. FRNN	160 kHz
2b. Edge Consolidation	110 kHz
3. GNN Edge Classifier	1.00 kHz
6. Track Consolidation	0.996 kHz
Allen	**595 kHz**



8 Optimisation

d Final Performance

Step	Throughput
0. Decode velo hits	1100 kHz
1. Embedding Network	330 kHz
2a. FRNN	160 kHz
2b. Edge Consolidation	110 kHz
3. GNN Edge Classifier	1.00 kHz
6. Track Consolidation	0.996 kHz
Allen	**595 kHz**



- **GNN** is the blottleneck of the pipeline

- GNN is slow because of
  - **`scatter_add`**: rely on `AtomicAdd`
  - **# rough edges**: $\times$ 9 w.r.t. hits
  - **# operations**

Conclusion & Opening

1. Get Data
2. Define Evaluation Procedure
3. From Exa.TrkX to ETX4VELO
4. Implementation in Allen
5. Optimization

Conclusion & Opening

1. Get Data
2. Define Evaluation Procedure
3. From Exa.TrkX to ETX4VELO
4. Implementation in Allen
5. Optimization

	ETX4VELO	**Exa.TrkX as a Service On NVIDIA Triton server** [10.1088/1748-0221/20/06/P06002](https://doi.org/10.1088/1748-0221/20/06/P06002)
Throughput	**1000 events/s**	**1.75 events/s**
GPU	NVIDIA RTX 2080Ti	NVIDIA A100
# hits / event	2k	350k

Conclusion & Opening

1. Get Data → 2. Define Evaluation Procedure → 3. From Exa.TrkX to ETX4VELO → 4. Implementation in Allen → 5. Optimization

Approach	Results	Potential
Filter edges **inside GNN**	$\times$ 3 throughput Loss in physics performance	Up to $\times$ 14 *(upper bound)*
Quantization	12% throughput gain	$\times$ 4 to $\times$ 16
Reconstruction approaches **without edge-edge connections**	Limited loss in physics performance performance	?

Conclusion & Opening

1. Get Data → 2. Define Evaluation Procedure → 3. From Exa.TrkX to ETX4VELO → 4. Implementation in Allen → 5. Optimization

Approach	Results	Potential
Filter edges **inside GNN**	$\times$ 3 throughput Loss in physics performance	Up to $\times$ 14 *(upper bound)*
Quantization	12% throughput gain	$\times$ 4 to $\times$ 16
Reconstruction approaches **without edge-edge connections**	Limited loss in physics performance performance	?

$\mathcal{O}(10-100)$ speed-up?

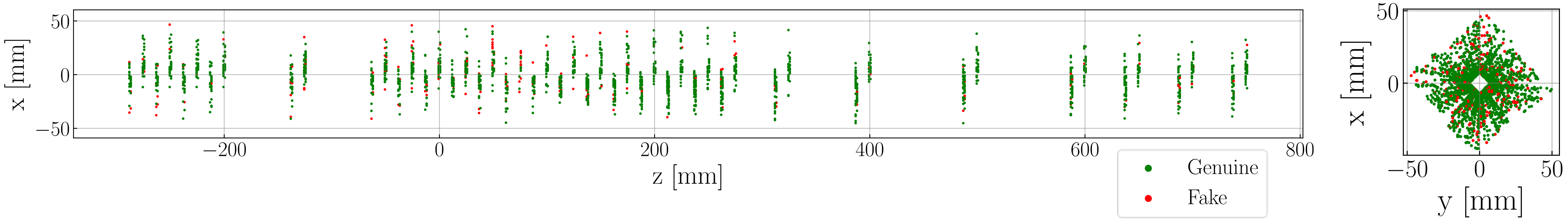
Thank you

3 Problem Formulation

a Track Finding in the Velo

2000 hits

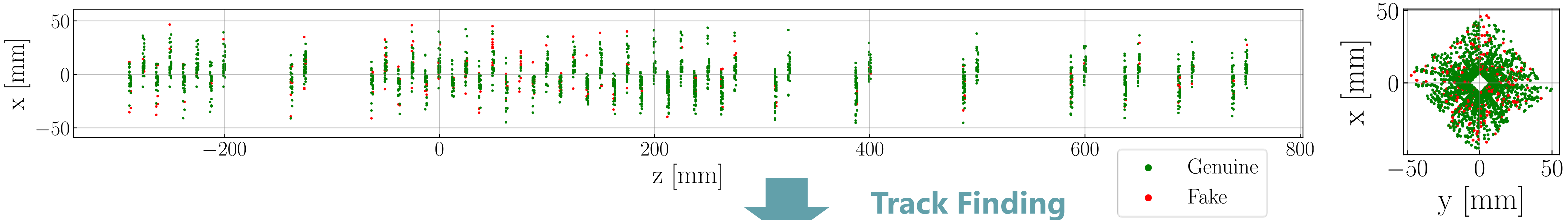
13% fake hits from *spillover* (*residuals from previous events*)



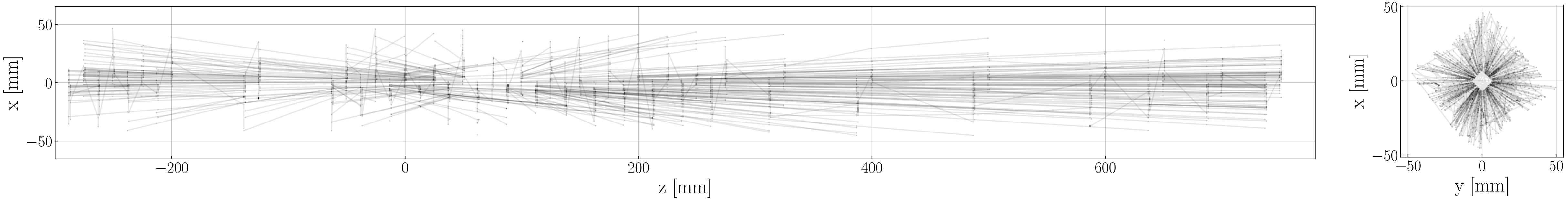
3 Problem Formulation

a Track Finding in the Velo

2000 hits
13% fake hits from *spillover* (*residuals from previous events*)



Track Finding



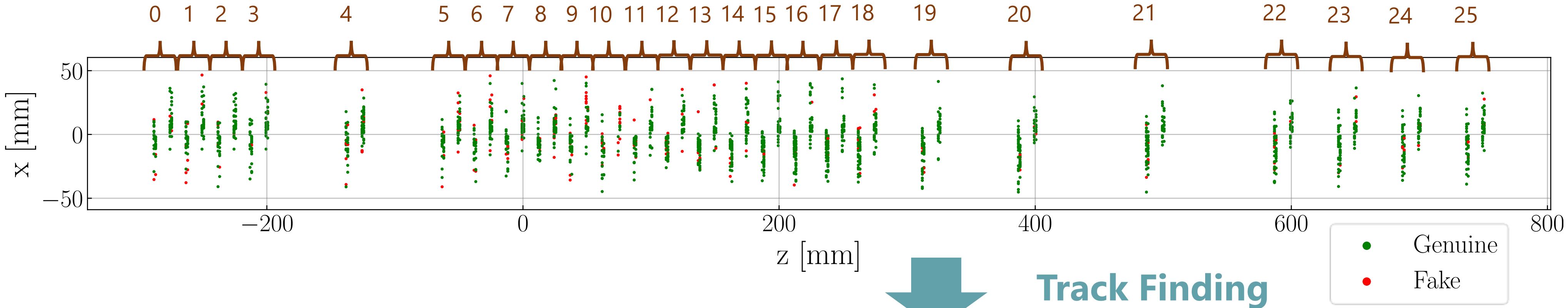
3 Problem Formulation

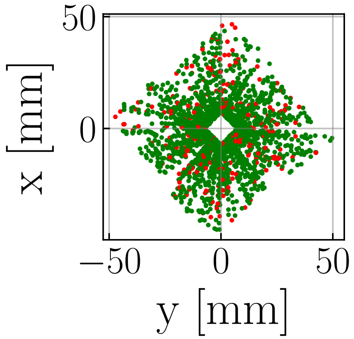
a Track Finding in the Velo

2000 hits

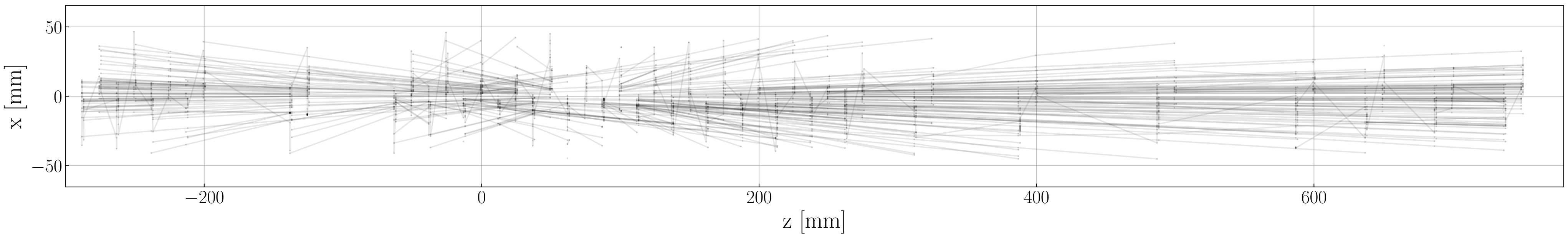
13% fake hits from *spillover* (*residuals from previous events*)

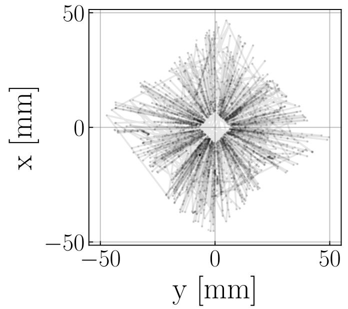
Layer number (0 to 25)



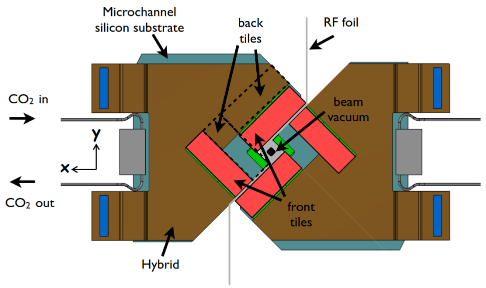


Track Finding





1 layer = **4 sensor planes**



1 layer

4 sensor planes

[P. C. Tsopelas, 'A Silicon Pixel Detector for LHCb', PhD Thesis, Vrije U., Amsterdam, 2016.]

3 Problem Formulation

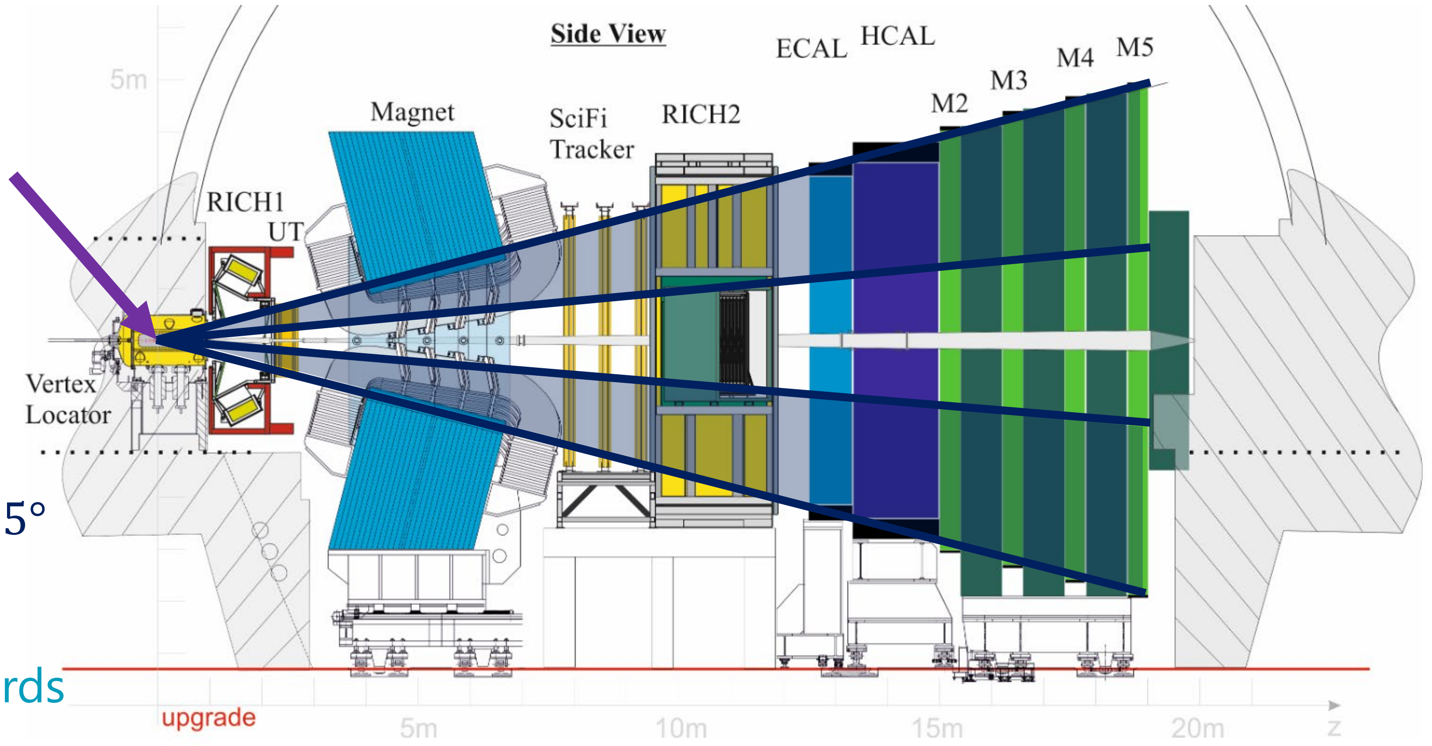
b Allen: a Fully GPU-based trigger

Collisions (Run 3)

- 20 MHz non-empty bunch crossing rate
- ~ 5 p-p collisions / bunch crossing
- p-p collision at $\sqrt{s} = 13.6$ TeV

LHCb Subdetectors

Acceptance
$2 < \eta < 5$
$1° < \theta < 15°$



Digitization

PCIe40 boards

5 TB/s

Trigger

Rate that can be saved to disk: 10 GB/s

Trigger: reduce the data rate to save to disk.

- Choose which **events to discard**,
- Only **save reconstructed objects**.

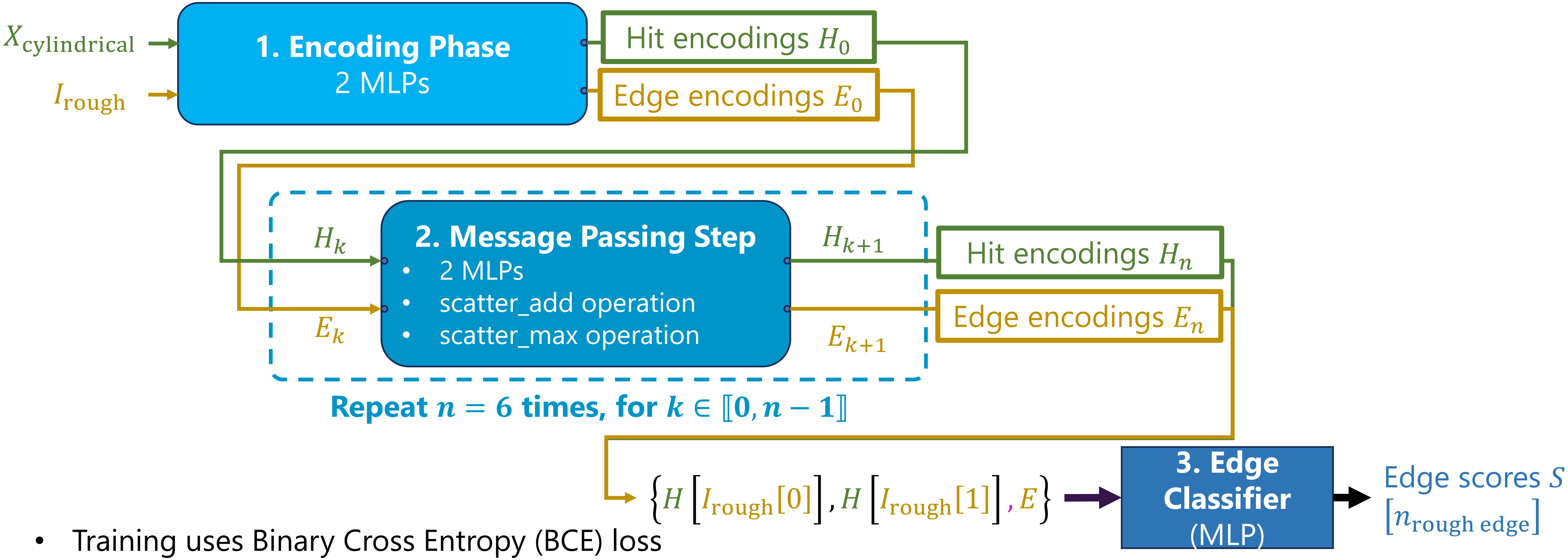
$\Rightarrow$

1. **Reconstruct** the events,
2. **Choose** which ones to keep given the reconstruction.

5 Exa.TrkX Pipeline

C GNN Edge Classifier

- **GNN** = 5 MLPs + an operation called `scatter_add`
  1. **Encoding phase:** encode hits and edges in **high-dimensional encoding space** $\boldsymbol{h} = \mathbf{256}$
  2. **Message passing phase:** $\boldsymbol{n}_{\text{iters}} = \mathbf{6}$ times, aggregate neighbour encodings and update encodings
  3. **Classification phase:** Final MLP to classify edges from their hit and edge encodings

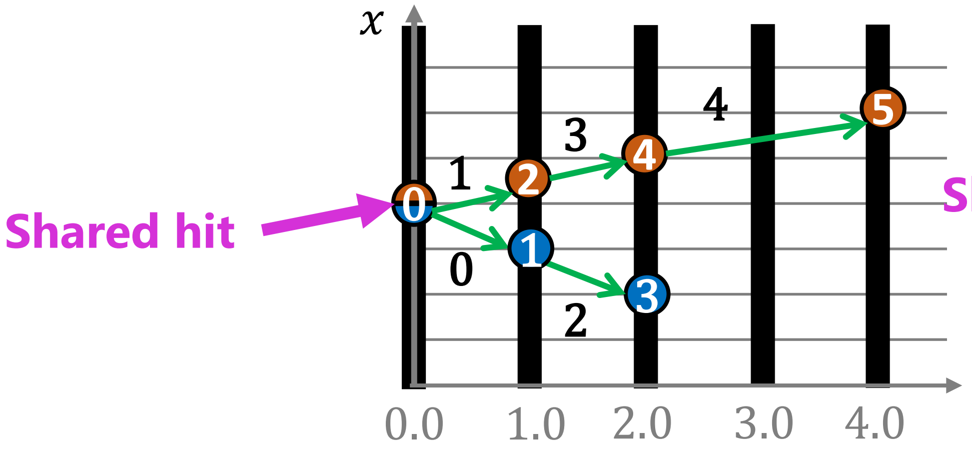


- Training uses Binary Cross Entropy (BCE) loss
- Only keep edges with score $s > s_{\text{edge,min}}$

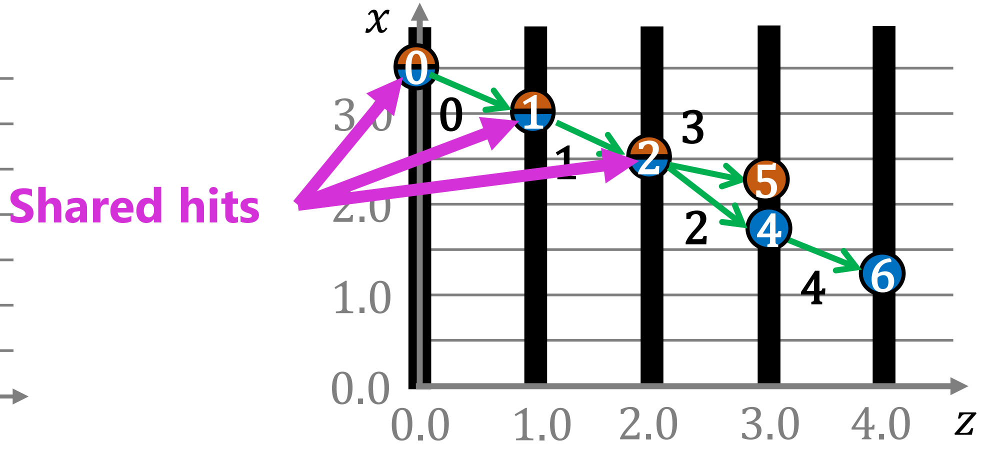
6 From Exa.TrkX to ETX4VELO

b Shared Hit Problem

A. First hit shared: 6.5/event



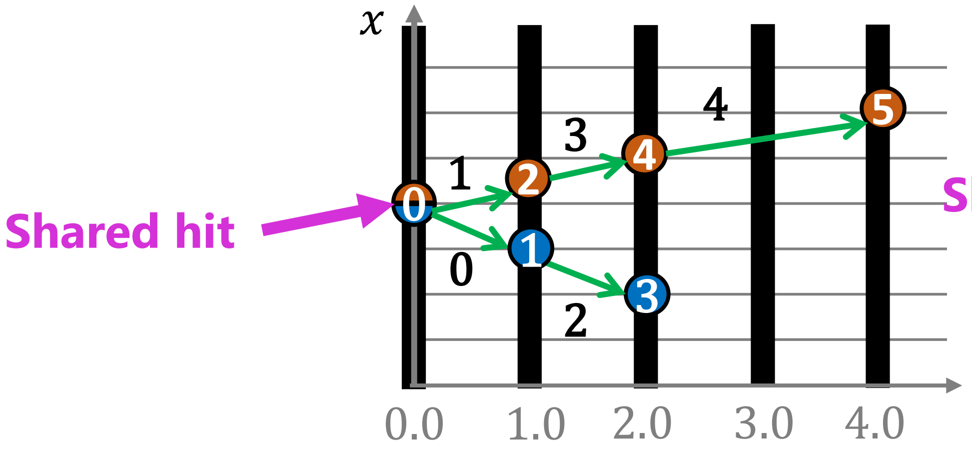
B. First hits shared: 2.8/event



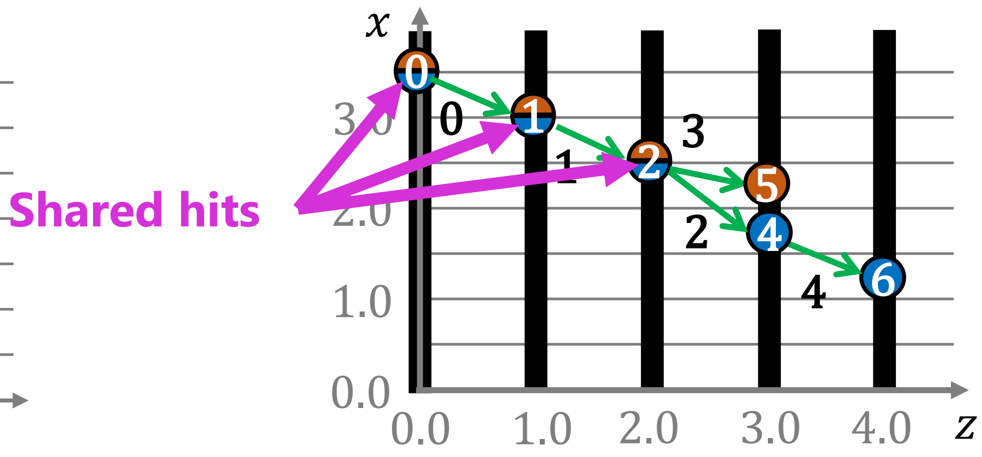
6 From Exa.TrkX to ETX4VELO

b Shared Hit Problem

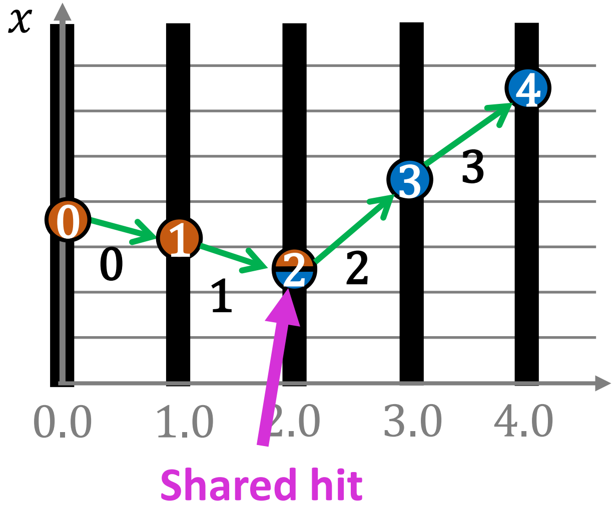
A. First hit shared: 6.5/event



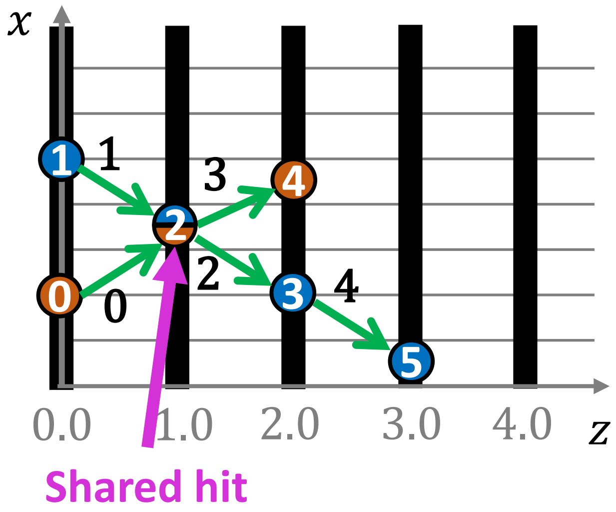
B. First hits shared: 2.8/event



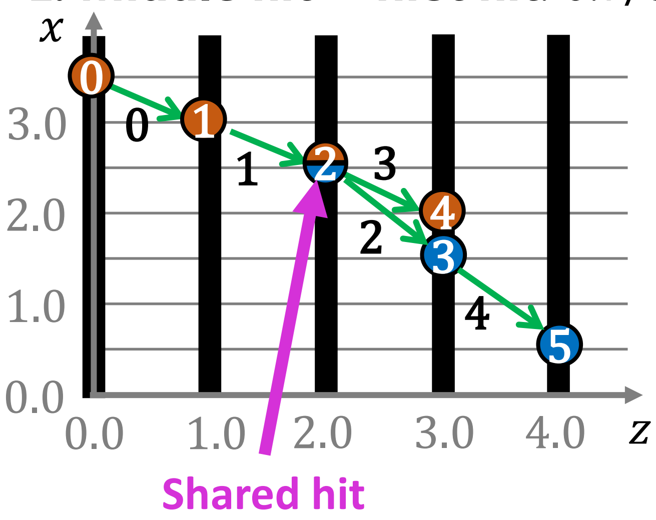
C. Last hit = first hit: 0.5/event



D. Middle hit shared: 0.4/event

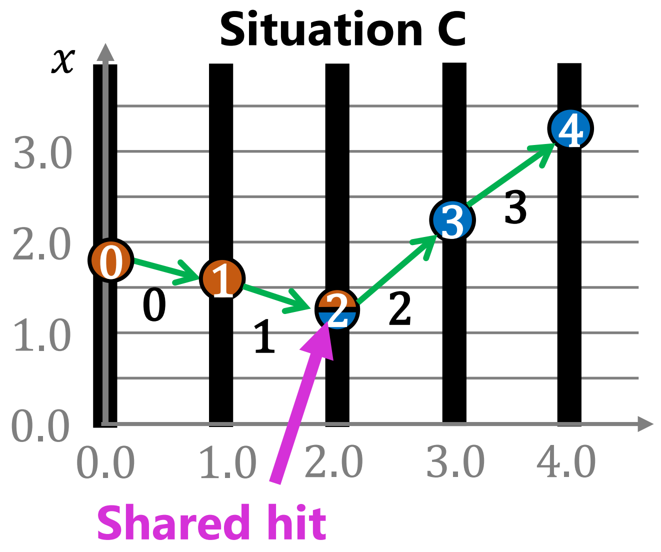


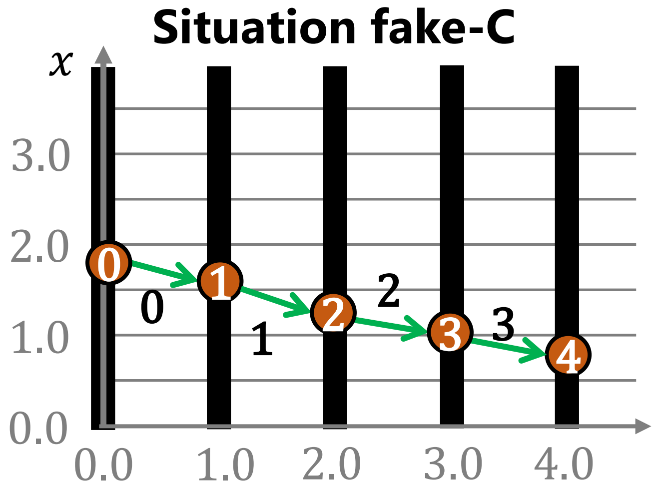
E. Middle hit = first hit: 0.7/event



6 From Exa.TrkX to ETX4VELO

b Round 2: Handle Shared Hits

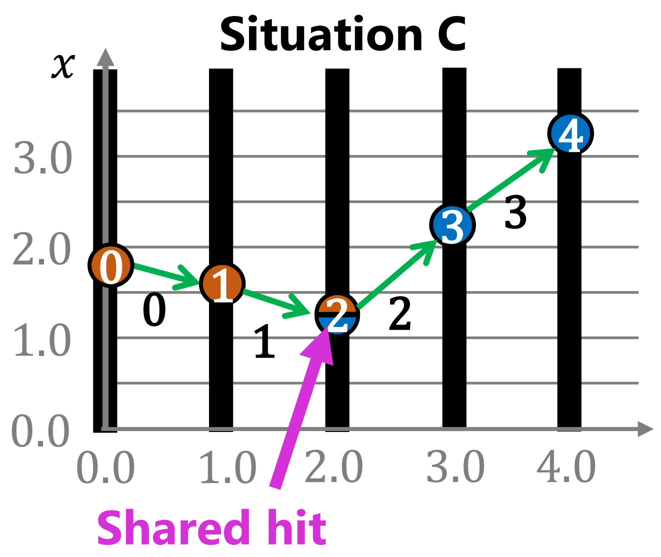


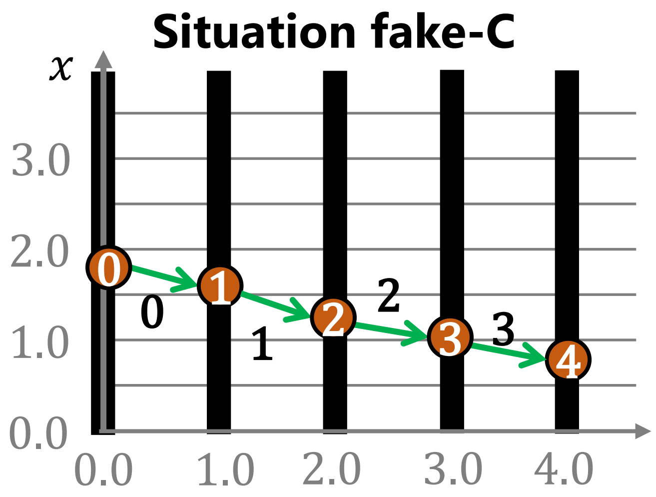


- **Problem**: Same graph!

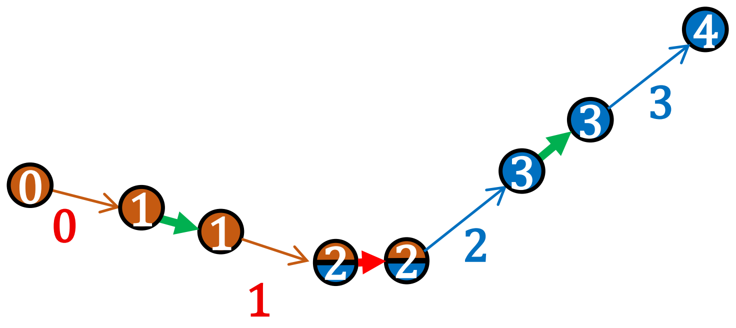
6 From Exa.TrkX to ETX4VELO

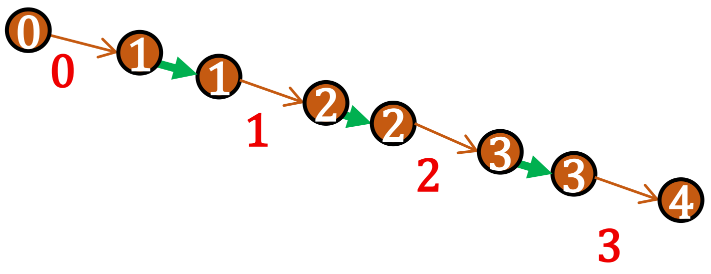
b Round 2: Handle Shared Hits





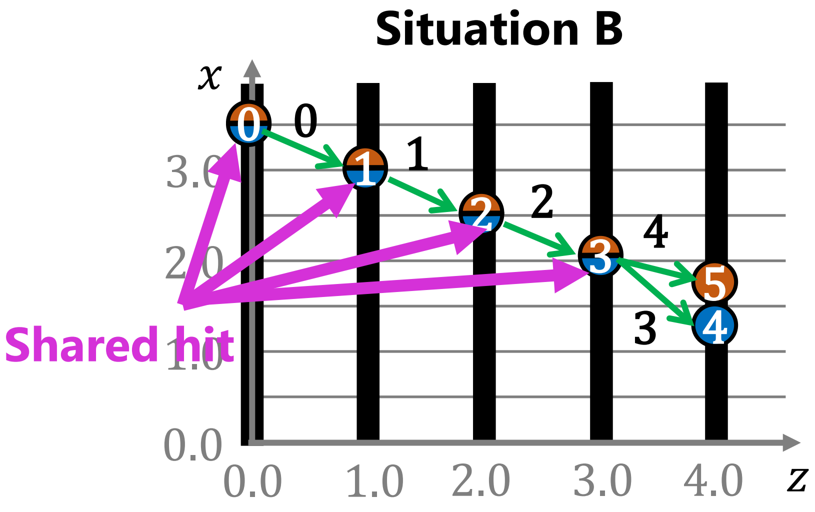
- **Problem**: Same graph!
- **Solution**: Middle edge-edge connections

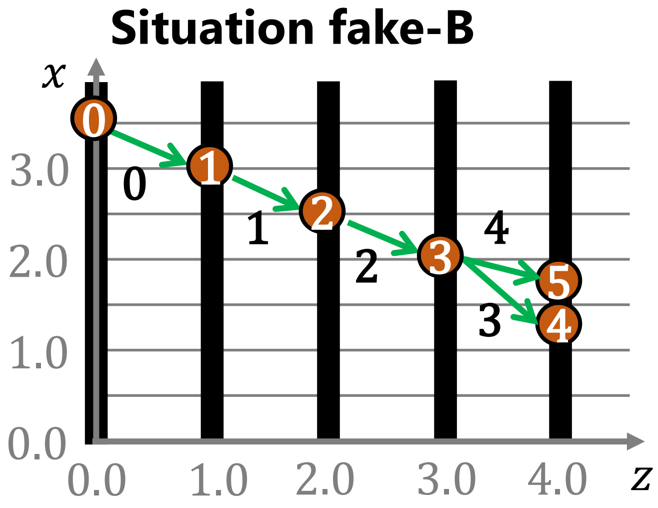




6 From Exa.TrkX to ETX4VELO

b Round 2: Handle Shared Hits

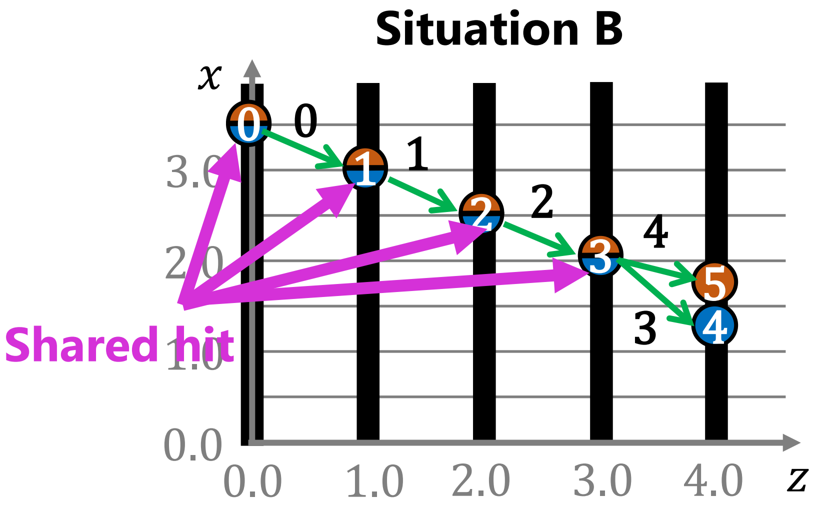


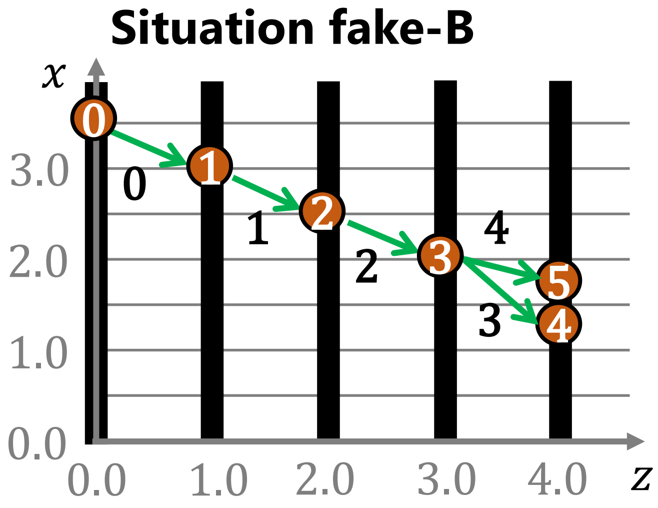


- **Problem**: Same graph!

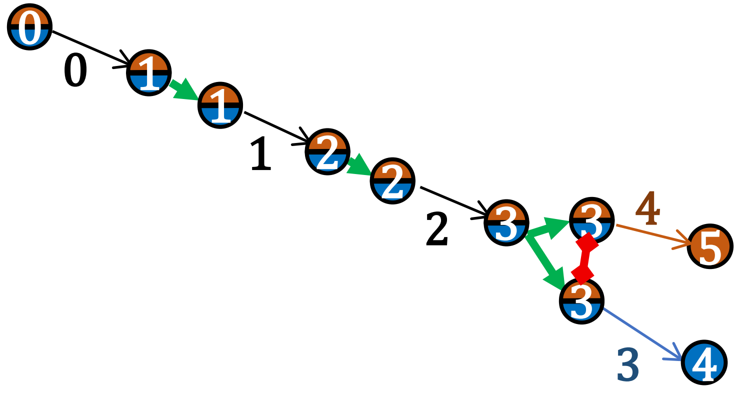
6 From Exa.TrkX to ETX4VELO

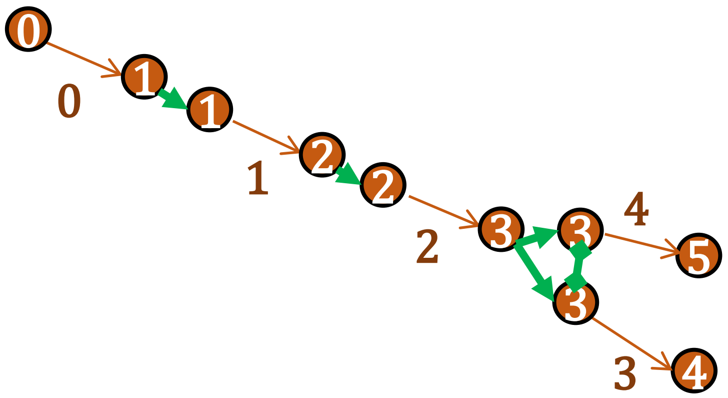
b Round 2: Handle Shared Hits





- **Problem**: Same graph!
- **Solution**: Left edge-edge connections

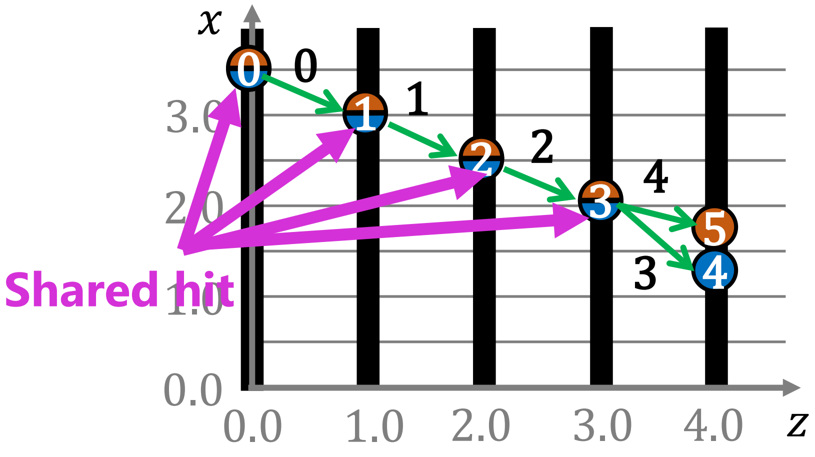




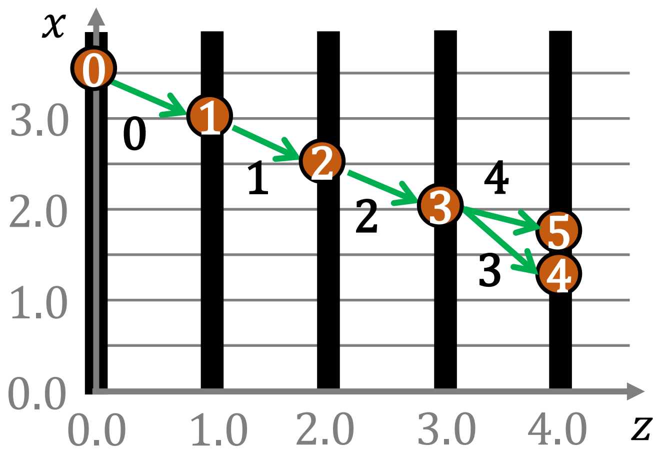
6 From Exa.TrkX to ETX4VELO

b Round 2: Handle Shared Hits

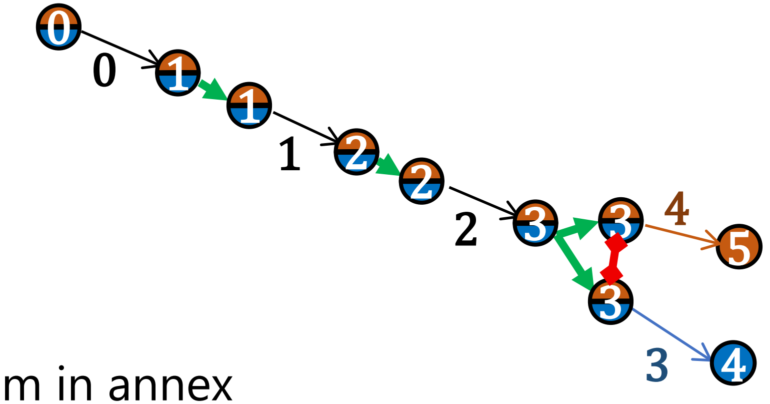
Situation B

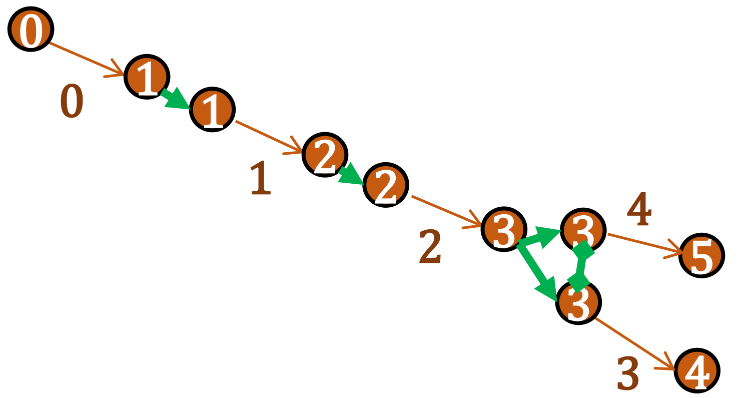


Situation fake-B



- **Problem**: Same graph!
- **Solution**: Left edge-edge connections





- Algorithm in annex

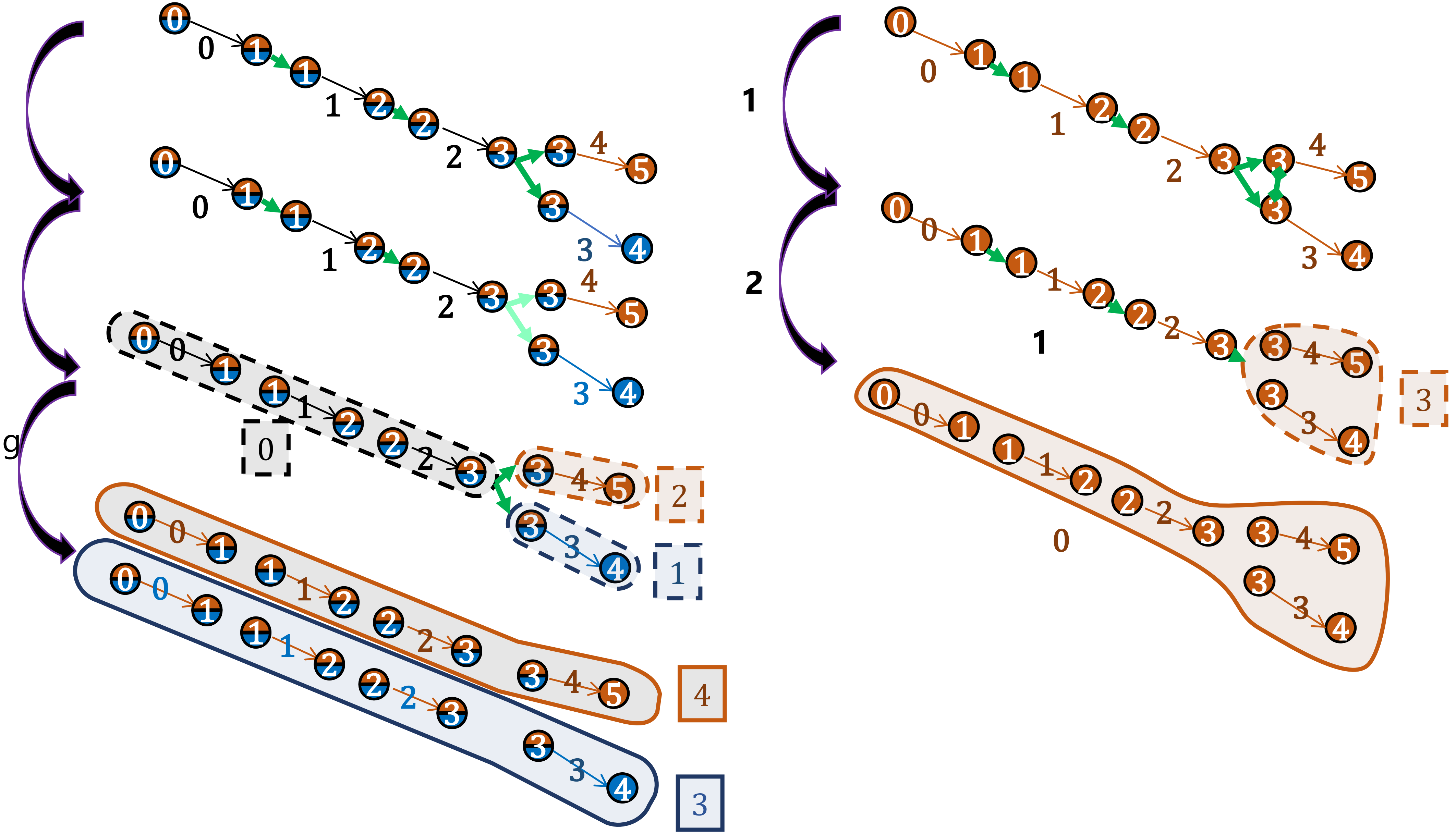
6 From Exa.TrkX to ETX4VELO

b Round 2: Handle Shared Hits

1. Connect **left and right connections**

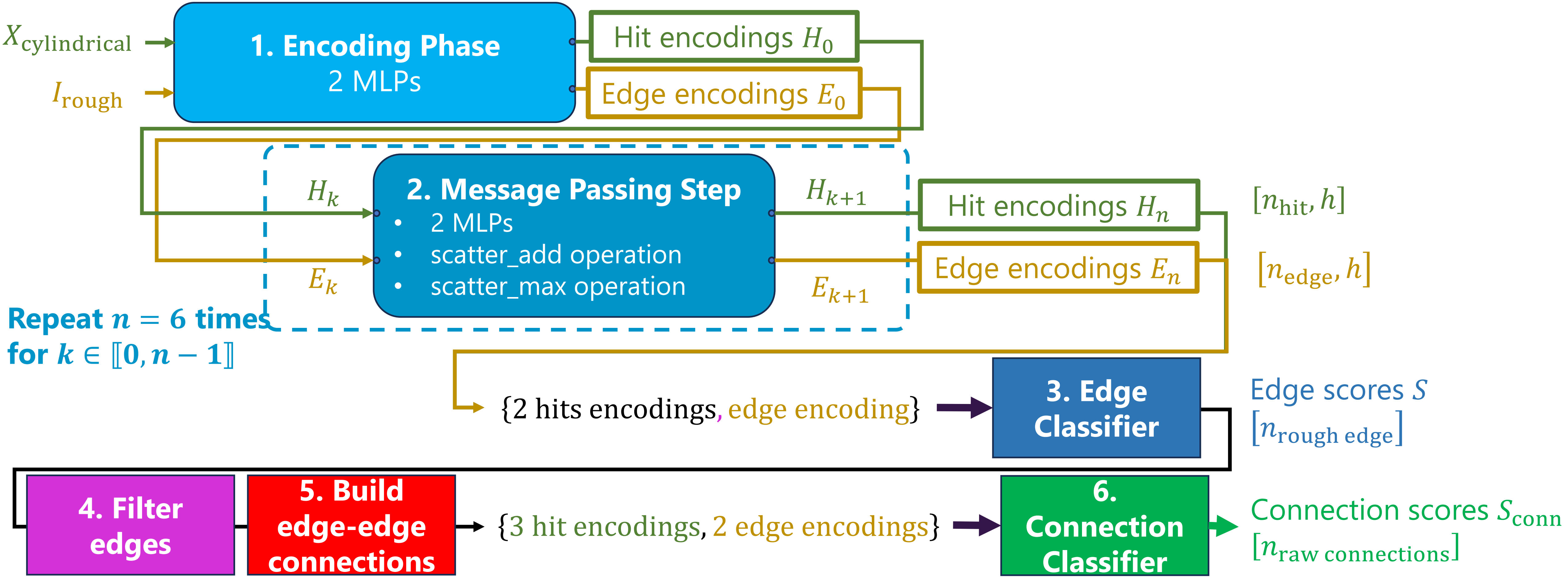
2. Connect **middle connections w.o. fork**

3. Each remaining middle connnection forms a track



6 From Exa.TrkX to ETX4VELO

b Round 2: Handle Shared Hits



GNN trained with **double objective**: Loss

$$\mathcal{L}_{\text{GNN}} = \mathcal{L}_{\text{edges}} + \mathcal{L}_{\text{connections}}$$

6 From Exa.TrkX to ETX4VELO

d With Spillover

- **Training with 700k events**

Metric	Category	Allen	ETX4VELO
Efficiency	Velo no electrons	98.25%	98.53%
	Long electrons	96.55%	98.43%
	Long from strange	97.74%	97.90%
Fake rate		0.88%	0.42%

Use simulation with spillover

Metric	Category	Allen	ETX4VELO
Efficiency	Velo no electrons	98.27%	98.38%
	Long electrons	96.90%	99.31%
	Long from strange	97.23%	97.01%
Fake rate		2.29%	1.56%
Throughput		**595 kHz**	**?**

6 From Exa.TrkX to ETX4VELO

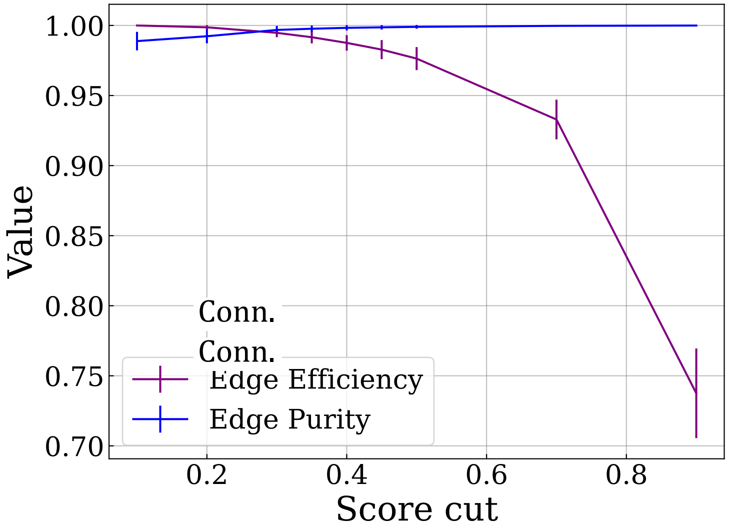
e Choice of min connection score $s_{\text{conn, min}}$

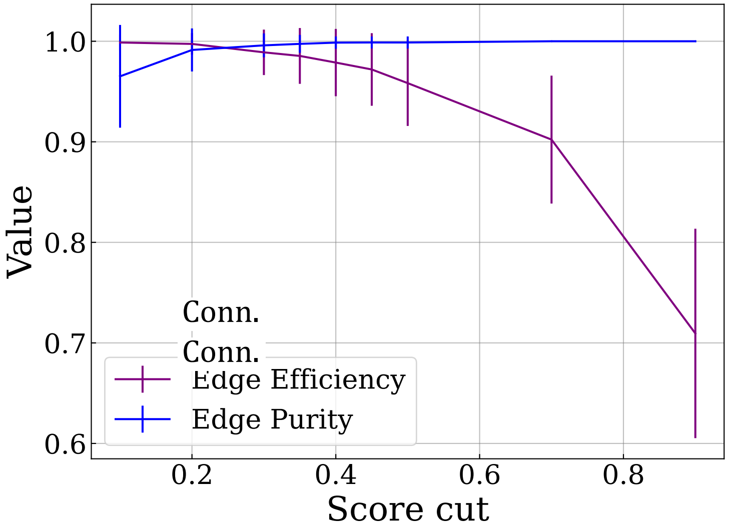
- Minimum edge score $s_{\text{min,edge}}$ mainly to **filter edges before building edge-edge connections**
  Set at $s_{\text{min,edge}} = 0.4$ after quick scan
- **Minimum connection score $s_{\text{conn,score}}$ most important**

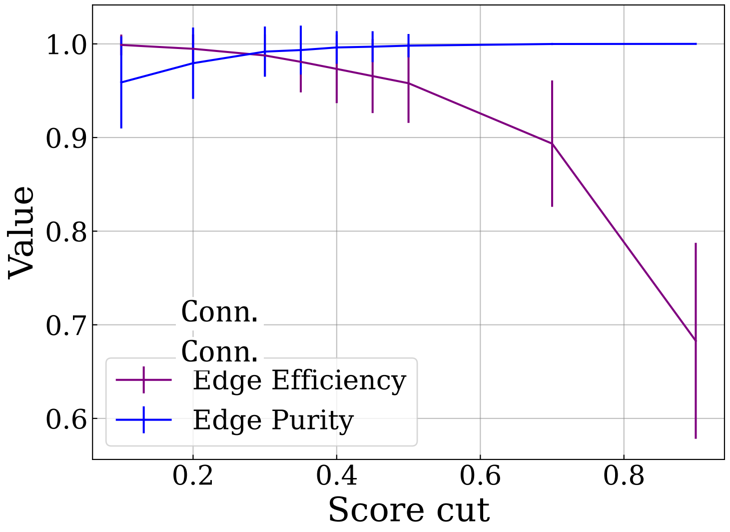
How to choose $s_{\text{conn,min}}$?

- Look at connection efficiency and purity?

$$\text{Efficiency} = \frac{\#\text{ selected true}}{\#\text{ true}} \qquad \text{Purity} = \frac{\#\text{ selected true}}{\#\text{ selected}}$$







No idea about final performance!

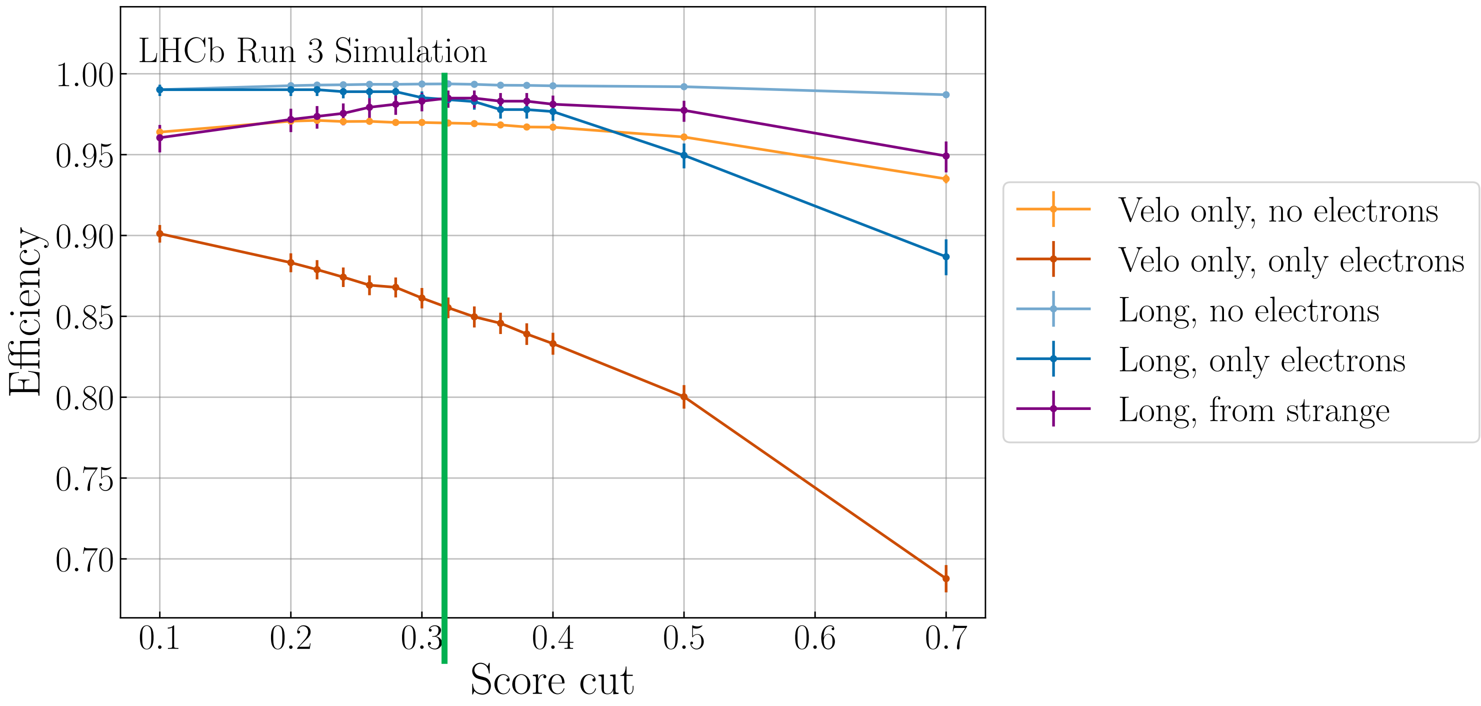
6 From Exa.TrkX to ETX4VELO

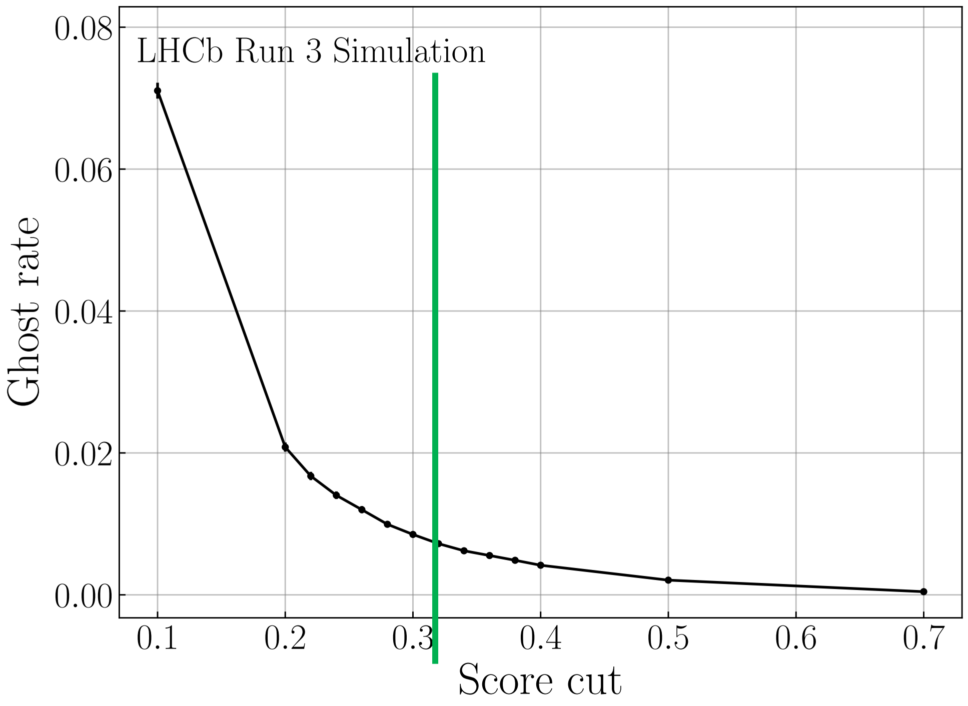
e Choice of min connection score $s_{\text{conn,min}}$

How to choose $s_{\text{conn,min}}$?

Look at efficiency and fake rate **after reconstruction**

→ Build tracks for each choice of $s_{\text{conn,min}}$





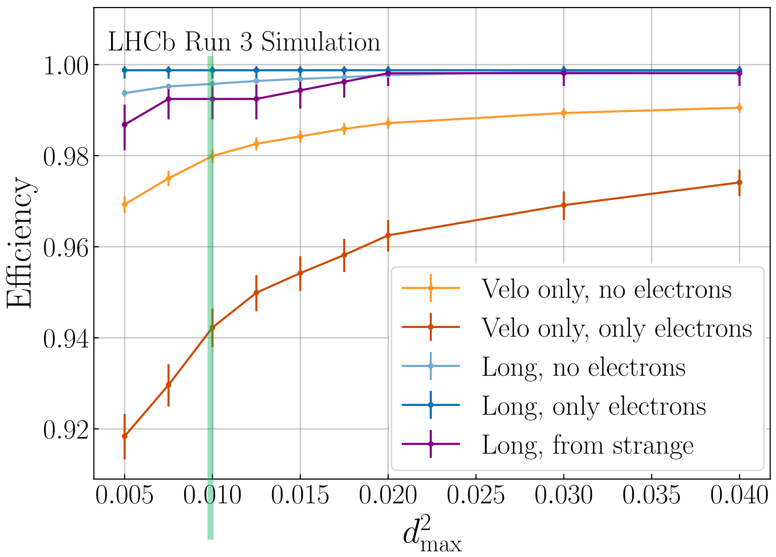
$s_{\text{conn,min}} = 0.32$ where long, from strange efficiency maximised.

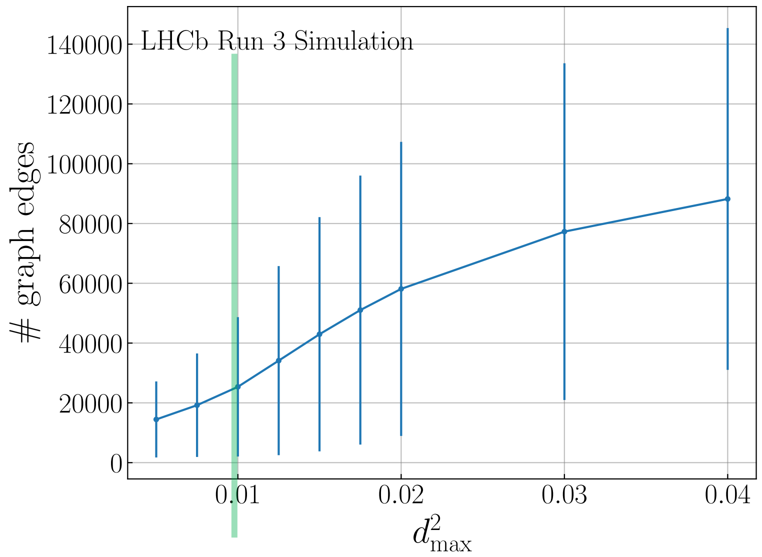
6 From Exa.TrkX to ETX4VELO

f Choice of *d* max 2

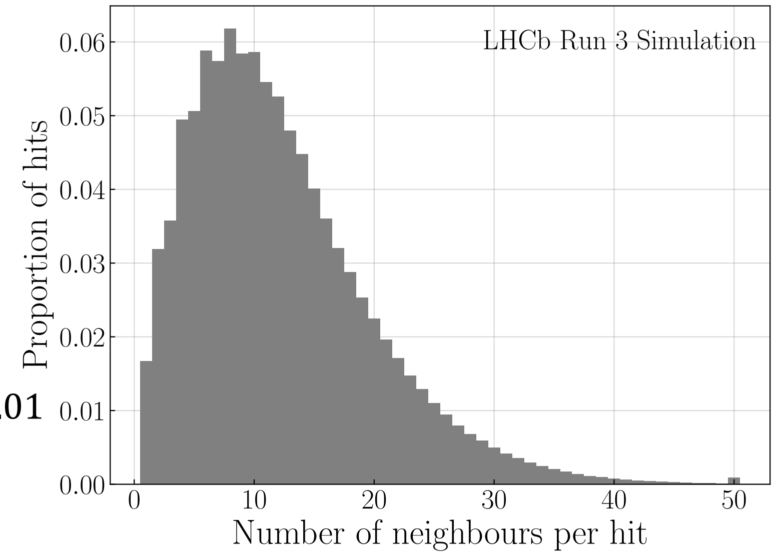
- **$k_{\mathbf{max}}$ set to 50**
- **How to choose $d^2_{\mathbf{max}}$?**
  - Same idea as $s_{\text{conn,min}}$? But there is the GNN after.
  - ⇒ Replace **GNN edge classifier** and **connection classifier** by **perfect classification**

- Allow to estimate **best efficiency** and **fake rate** obtainable if GNN perfect.
- Also look at **graph size**: large graph ⇒ small throughput



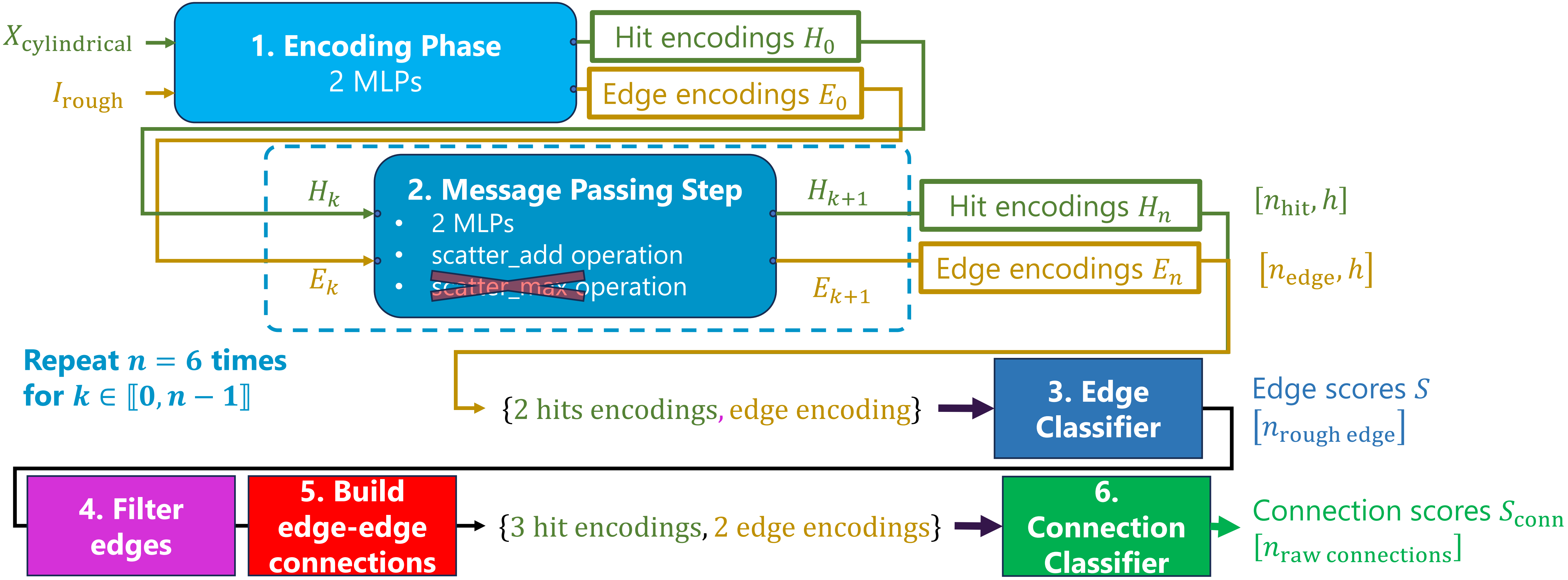


$d^2_{\text{max}} = 0.01$



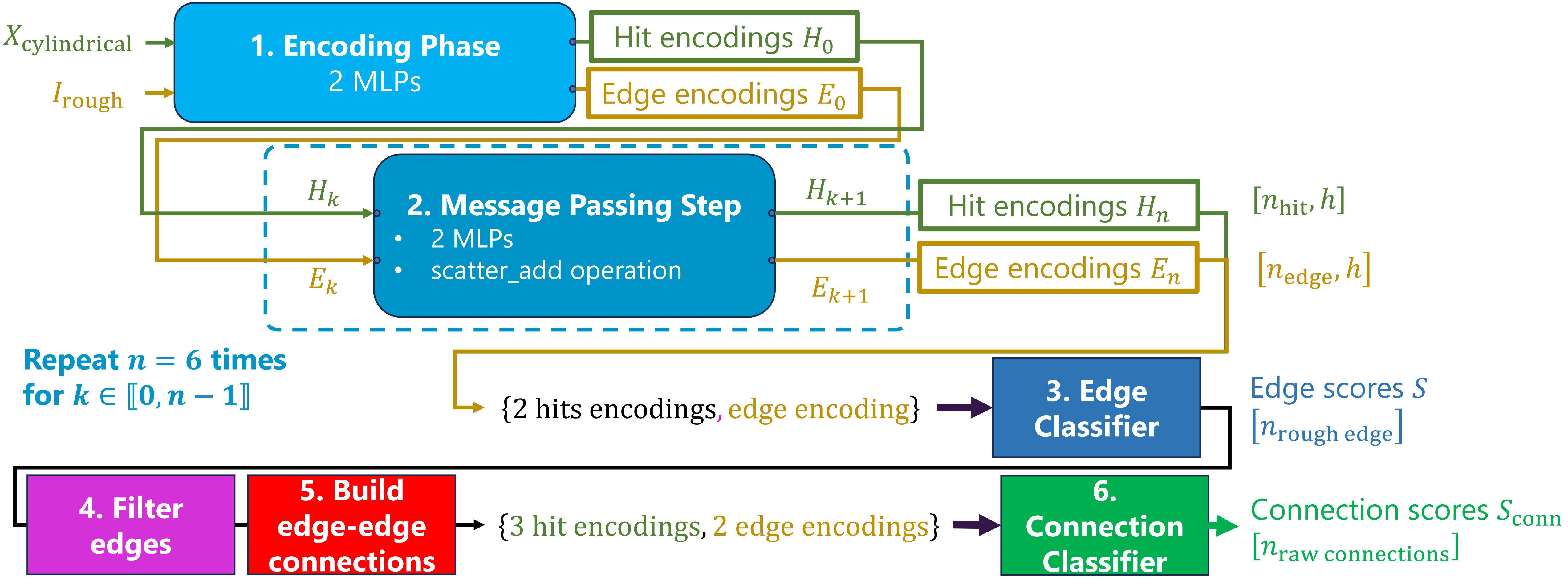
Step 1: Increase throughput

1. Removing `scatter_max`, **only use `scatter_add`**,



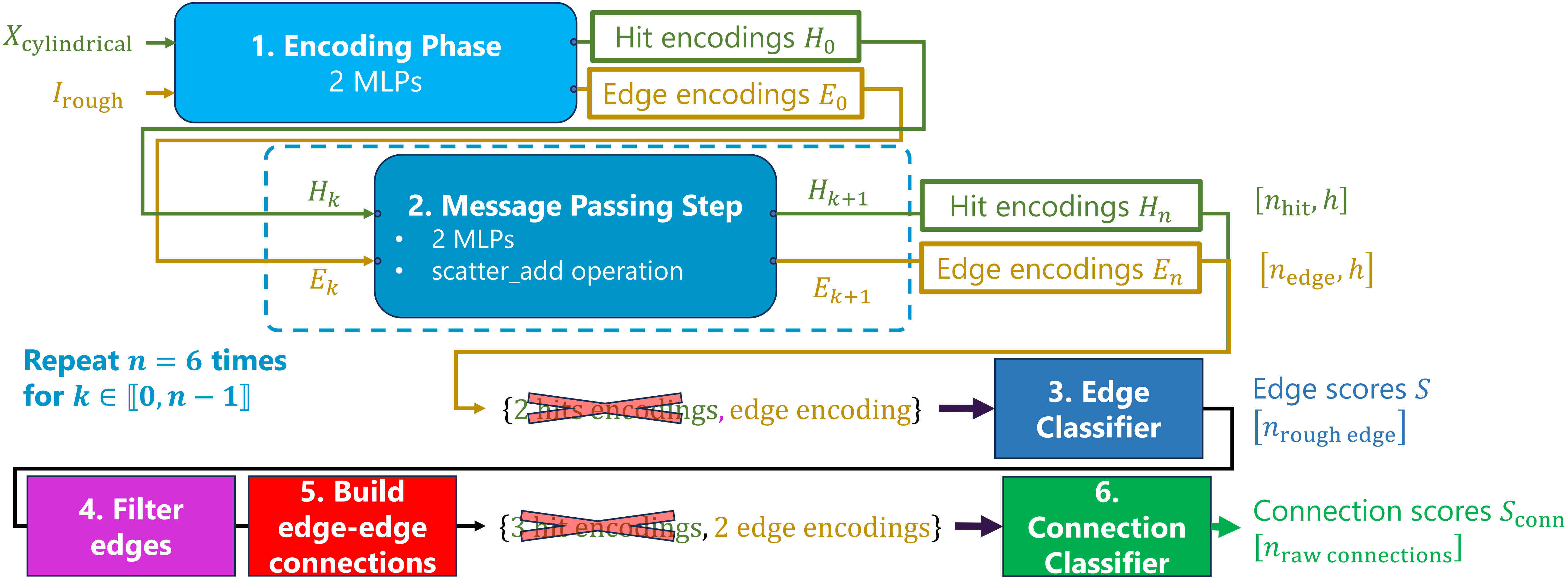
Step 1: Increase throughput

1. Removing `scatter_max`, **only use `scatter_add`**,
2. Decreasing **hit and edge encoding dimensions** from $h = 256$ to $h = 32$



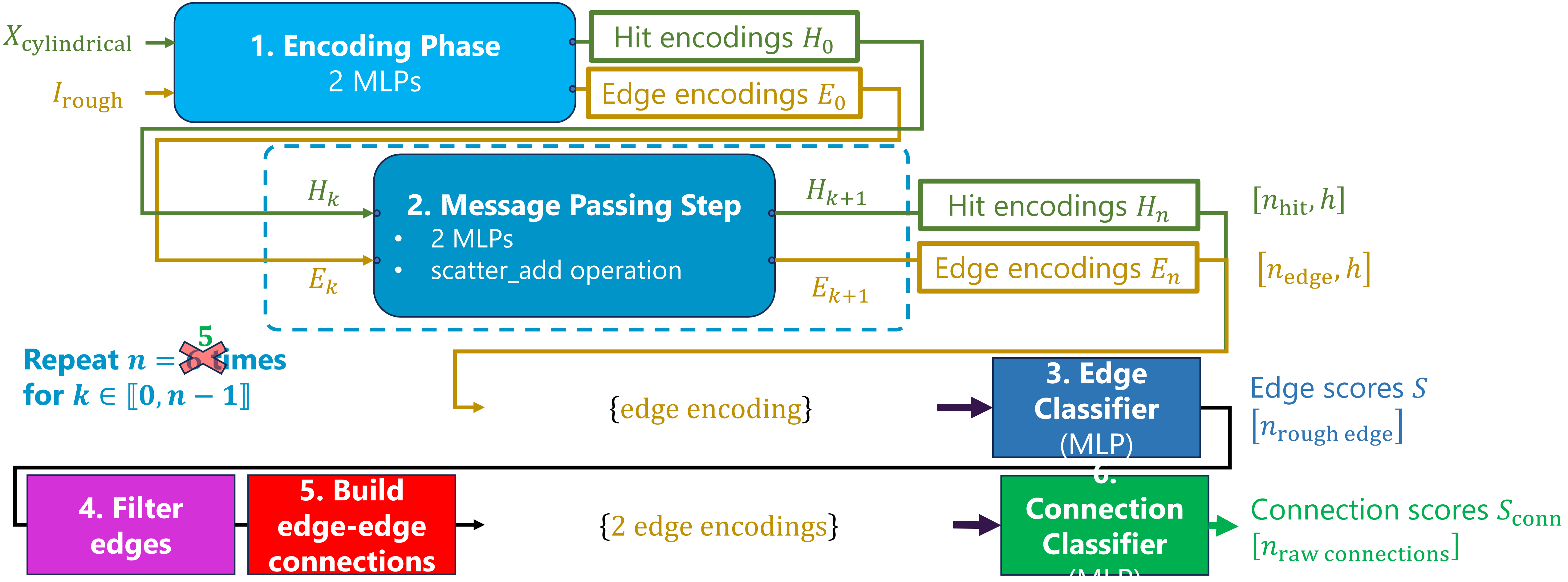
Step 1: Increase throughput

1. Removing `scatter_max`, **only use `scatter_add`**,
2. Decreasing **hit and edge encoding dimensions** from $\boldsymbol{h} = \mathbf{256}$ to $\boldsymbol{h} = \mathbf{32}$
3. Use only **edge encodings** for classifications



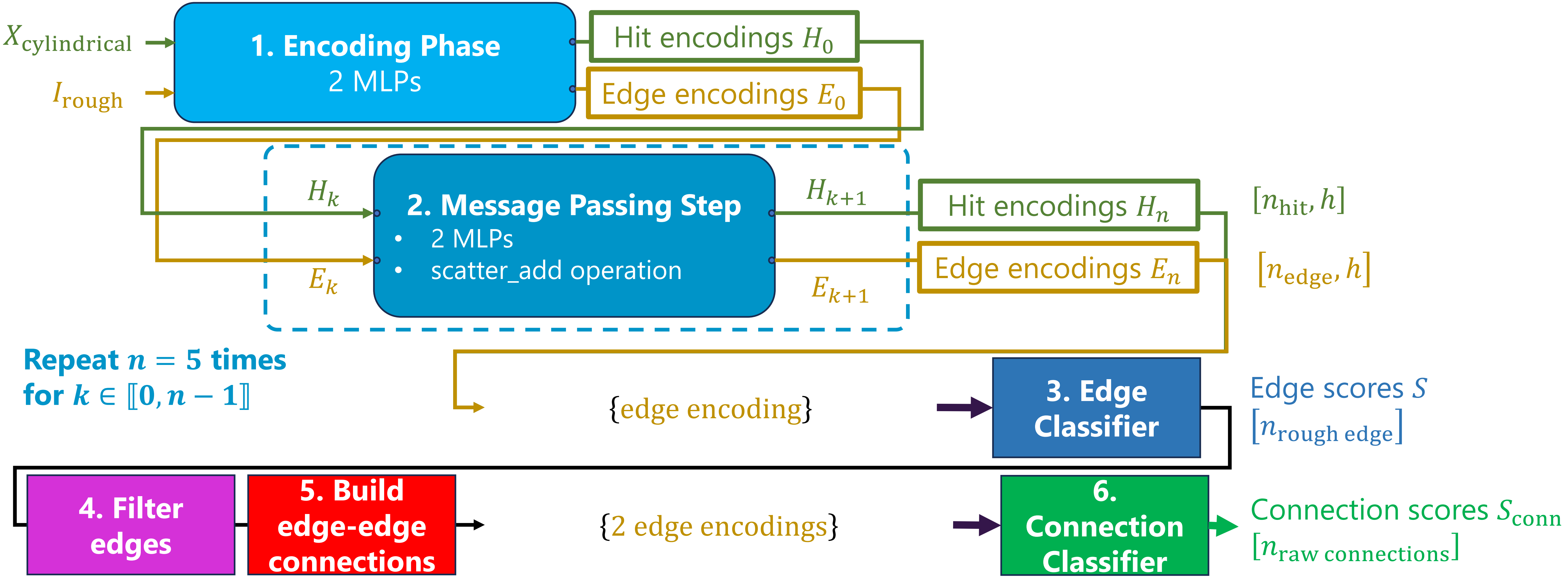
Step 1: Increase throughput

1. Removing `scatter_max`, **only use `scatter_add`**,
2. Decreasing **hit and edge encoding dimensions** from $\boldsymbol{h} = \mathbf{256}$ to $\boldsymbol{h} = \mathbf{32}$
3. Use only **edge encodings** for classifications
4. Decreasing **# graph iterations** from **6** to **5**



Step 1: Increase throughput

1. Removing `scatter_max`, **only use `scatter_add`**,
2. Decreasing **hit and edge encoding dimensions** from $\boldsymbol{h} = \mathbf{256}$ to $\boldsymbol{h} = \mathbf{32}$
3. Use only **edge encodings** for classifications
4. Decreasing **# graph iterations** from **6** to **5**



Step 1: Increase throughput

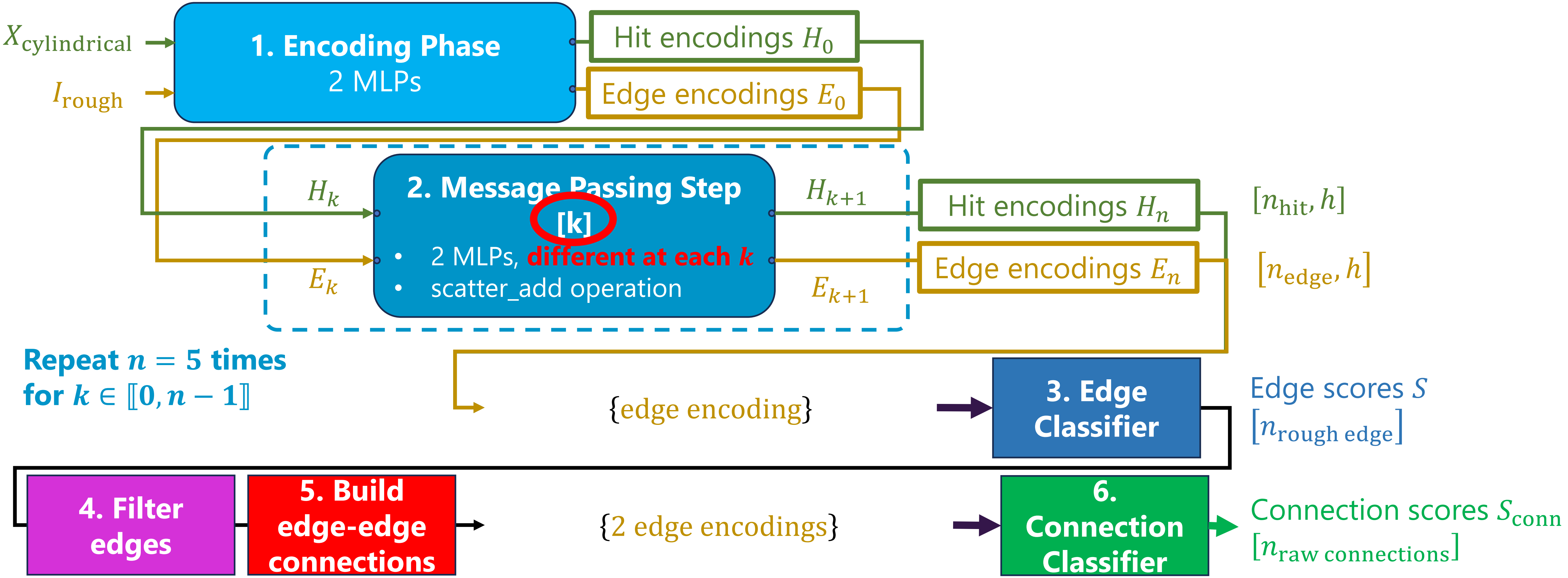
a) Removing `scatter_max`, **only use `scatter_add`**,
b) Decreasing **hit and edge encoding dimensions** from $\boldsymbol{h} = 256$ to $\boldsymbol{h} = 32$
c) Use only **edge encodings** for classifications
d) Decreasing **# graph iterations** from **6** to **5**

Metric	Category	Allen	(1a) $h = 256$	$h = 128$	$h = 64$	(1b) $h = 32$	(1c) Only E	(1d) $n_{\text{iter}} = 5$ (not 1c)
Efficiency	**Long**	99.35%	99.37%	99.32%	99.28%	99.16%	99.16%	99.17%
	Long from strange	97.53%	97.87%	97.30%	97.30%	96.66%	96.76%	96.80%
	Long electrons	95.21%	98.71%	98.47%	98.61%	98.25%	98.40%	98.47%
Fake rate		2.19%	0.58%	0.77%	1.02%	1.31%	1.31%	1.32%
GNN throughput (events/second)		**595k**	0.26	0.134	0.333	0.672	0.695	0.784

- Throughput $\mathcal{O}(10^3)$ **below**
- Some performance lost but:
  - Low fake rate
  - High long electron efficiency

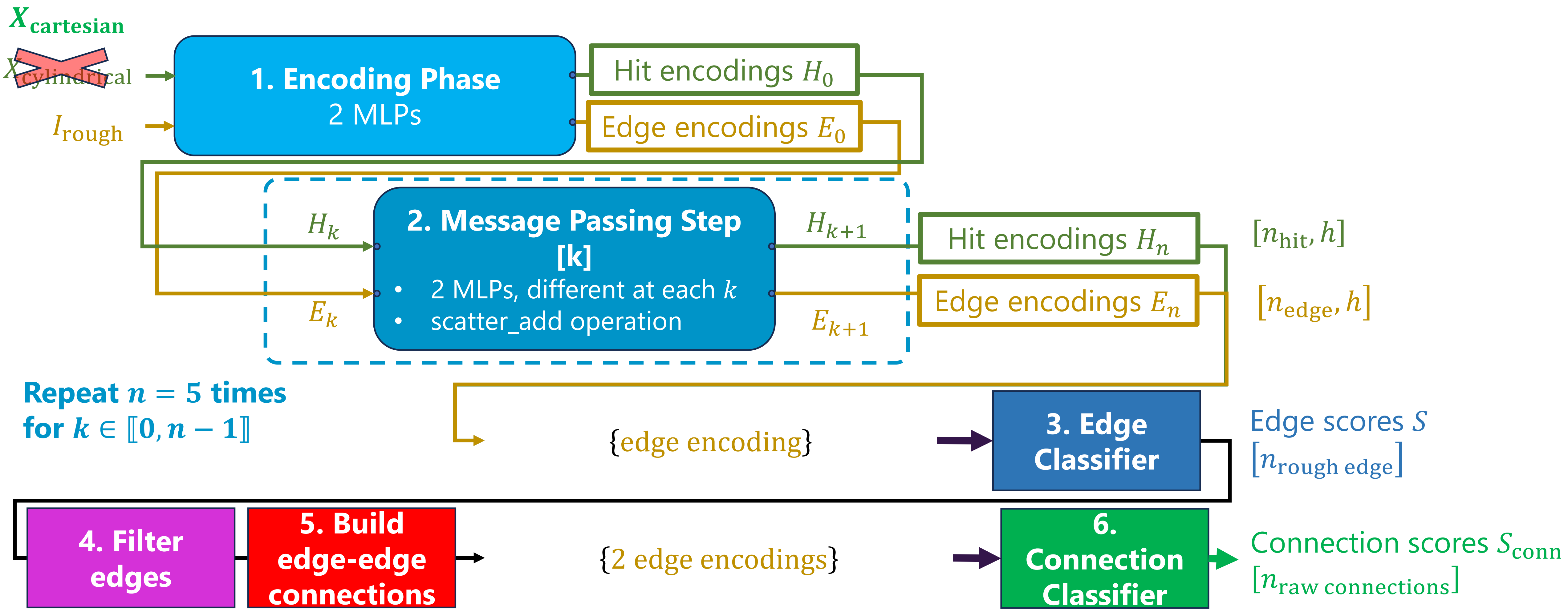
Step 2: recover lost performance

5. GNN **non-recursive**



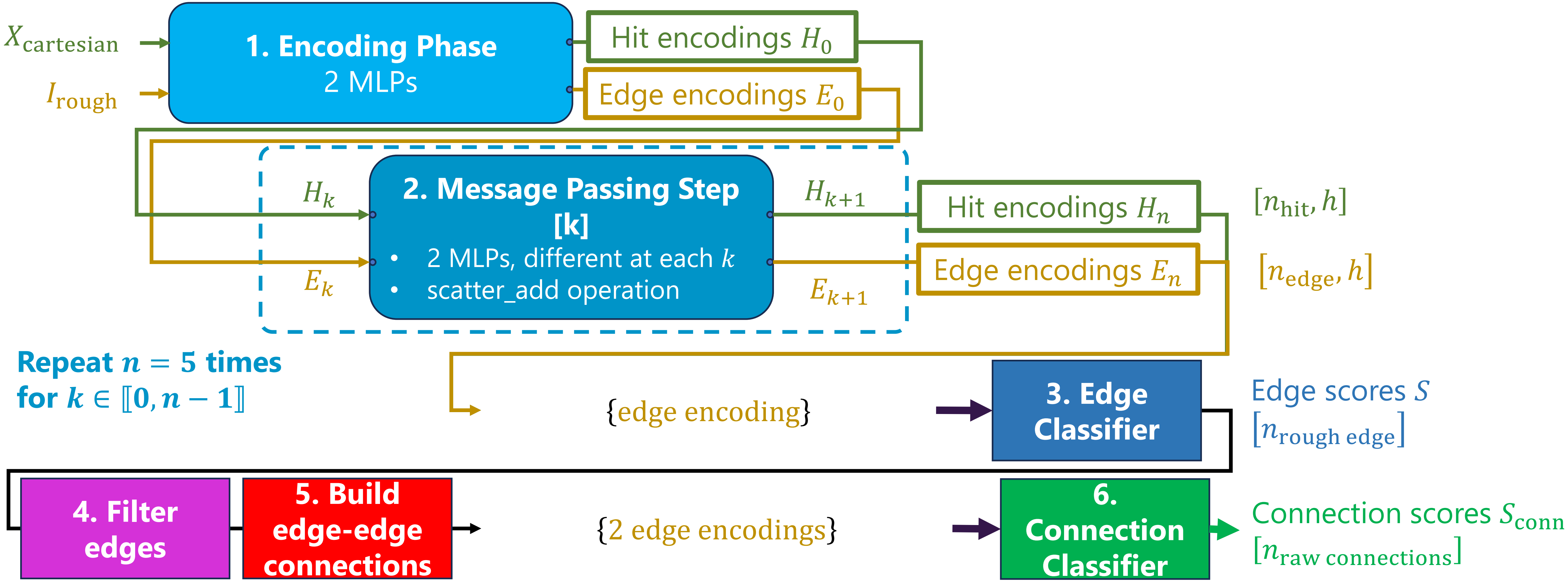
Step 2: recover lost performance

5. GNN **non-recursive**
6. Use **cartesian coordinates** for input node features instead of **cylindrical**



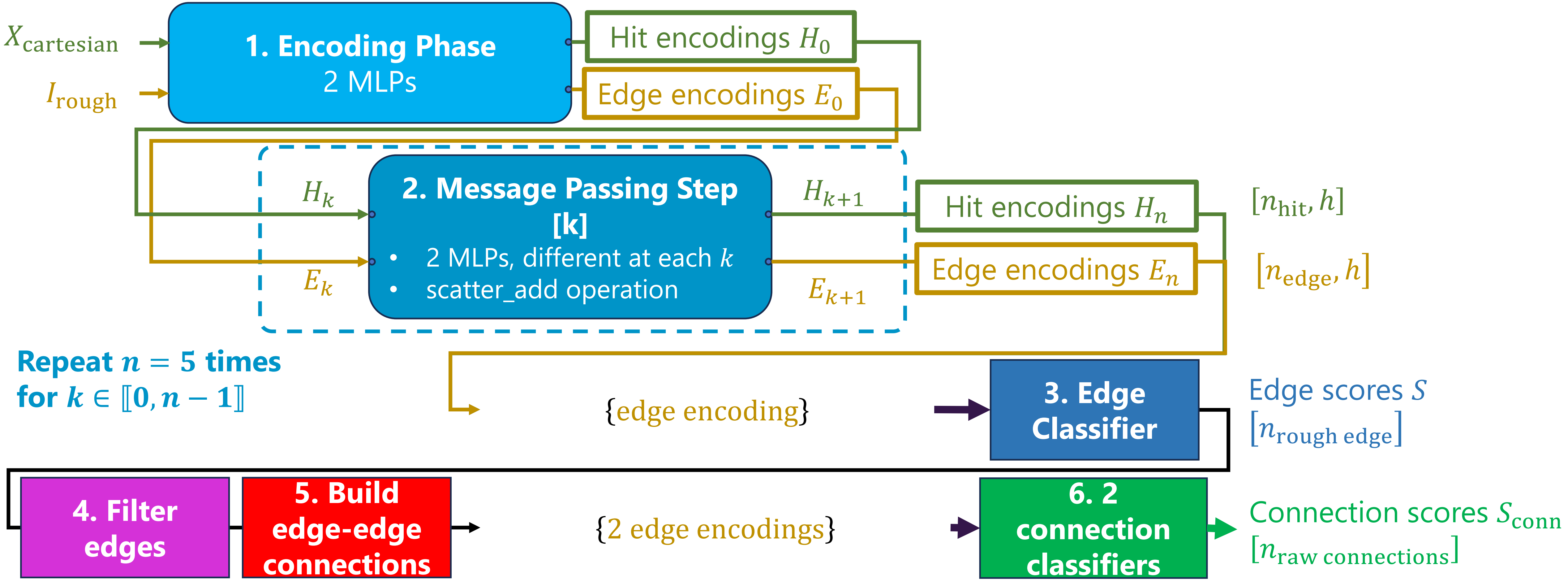
Step 2: recover lost performance

5. GNN **non-recursive**
6. Use **cartesian coordinates** for input node features instead of **cylindrical**
7. Use the **new embedding network** from previous slide
8. Do not **remove curved particles from training set**, but only **from the loss**



Step 2: recover lost performance

5. GNN **non-recursive**
6. Use **cartesian coordinates** for input node features instead of **cylindrical**
7. Use the **new embedding network** from previous slide
8. Do not **remove curved particles from training set**, but only **from the loss**; Consider isolated edges as fake.
9. Use a **different classifier** for middle connections, & left/right connections



Step 2: recover lost performance

a. GNN **non-recursive**
b. Use **cartesian coordinates** for input node features instead of **cylindrical**
c. Use the **new embedding network** from previous slide
d. Do not **remove curved particles from training set**, but only **from the loss**; Consider isolated edges as fake.
e. Use a **different classifier** for middle connections, & left/right connections

Metric	Category	Allen	(4) $n_{\text{iter}} = 5$ (not only E)	(5) Non-recursive	(6) Cartesian coords.	(7) New embed.	(8) Mask curved	(9) Diff. classifier
Efficiency	**Long**	99.35%	99.17%	99.24%	99.25%	99.31%	99.32%	**99.35%**
	Long from strange	97.53%	96.80%	96.96%	97.13%	97.46%	97.20%	**97.43%**
	Long electrons	95.21%	98.47%	98.44%	98.27%	98.10%	98.30%	**98.08%**
Fake rate		2.19%	1.32%	1.15%	1.12%	1.02%	1.11%	**1.01%**
GNN throughput (events/second)		**595k**	0.784	0.977	1.084	0.985	0.985	0.985

- Physics performance recovered.
- Other change to explore: reduce n_{iter} **to 4**

9 Opening

a Pre Edge-Filtering Approach

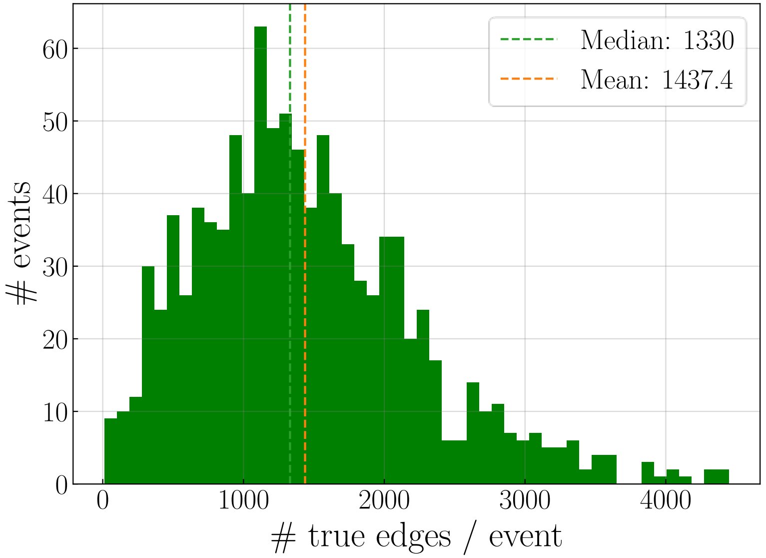
- # edges is a bottleneck

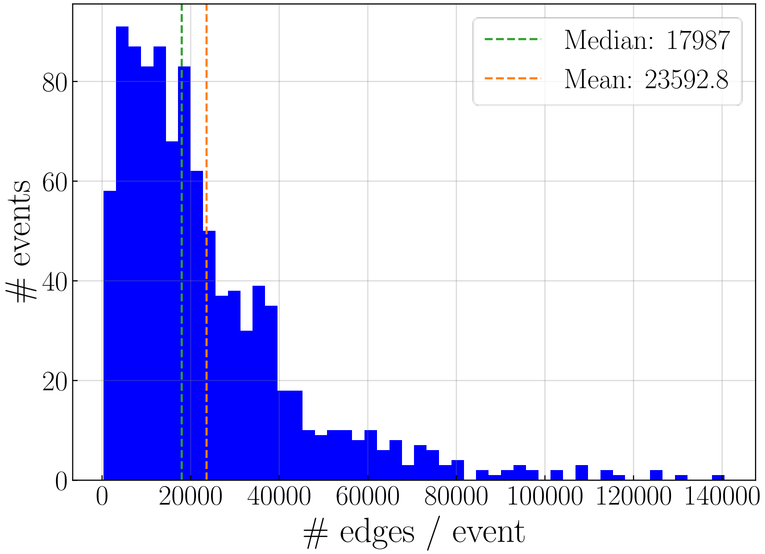
- **Idea 1**: 2 GNNS
  - 1st small shallow GNN to **remove most of obvious fake edges**
  - 2nd normal GNN
- **Idea 2**: Filter edges **within the GNN**
- Results:
  - Throughput × 3
  - Lost in performance

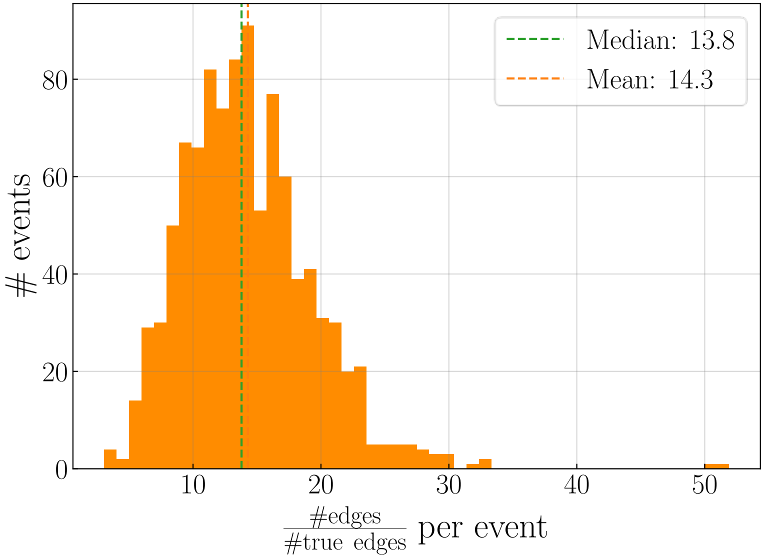
Metric	Category	Allen	ETX4VELO	Pre Edge-filtering
Efficiency	**Long no electrons**	99.35%	99.35%	**99.20%**
	Long electrons	95.21%	98.08%	98.15%
	Long from strange	97.53%	97.43%	**96.60%**
Fake rate		2.19%	1.01%	1.06%
GNN throughput (kHz)		595	0.985	**2.97**

9 Opening

a Pre Edge-Filtering Approach







9 Opening

b Quantization

- **Tensors** and **parameters** in **FLOAT32** (4 bits)
- **Quantization**: Convert them to **INT8** (1 bit)

- Expected throughput gain:
  - **4** × **gain** since memory / 4
  - **16** × **gain** for **matrix multiplications** thanks to **Tensor cores**.

- If operation not quantizable ⇒ Need to Dequantize/Quantize

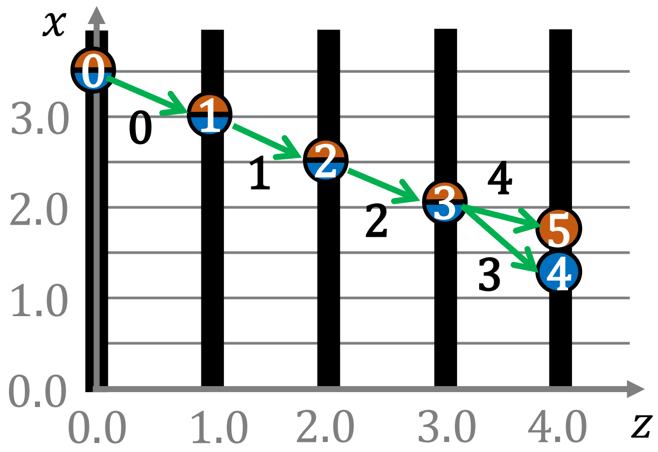
Naive quantization of the GNN: 1.00 kHz → 1.127 kHz

9 Opening

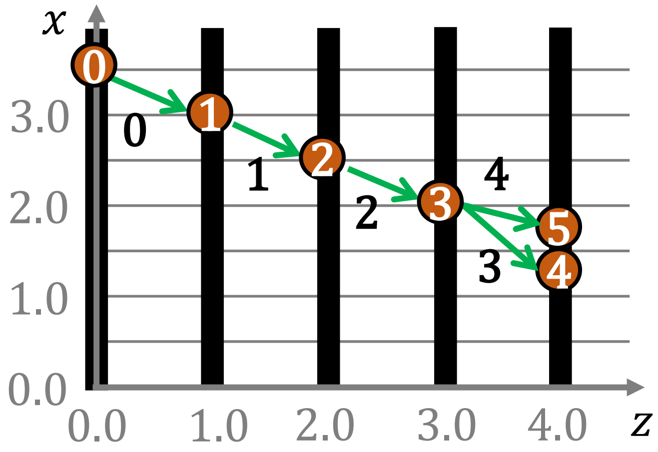
C Other Reconstruction Approaches

- **Left** and **right edge-edge connections** used to distinguish these two cases

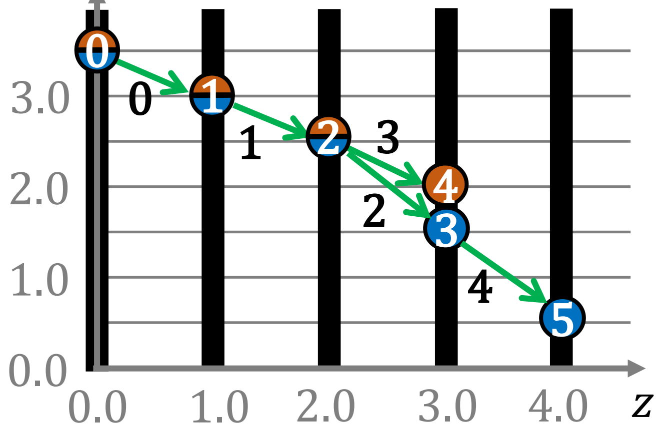
Situation B

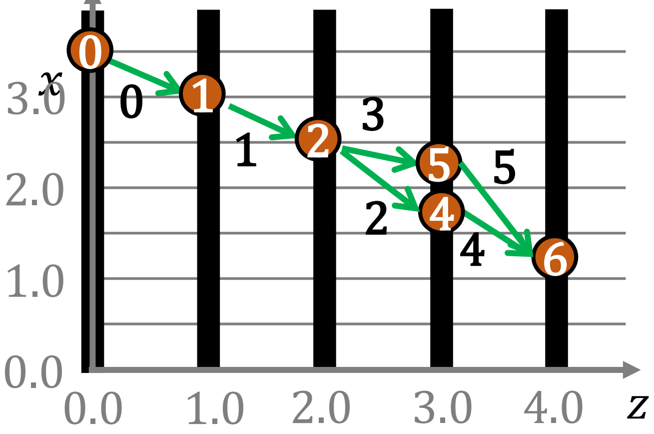


Situation fake-B



- Split at the last layer ⇒ probably **indistinguishable in practice**
- If **more than one layer**: no need for **left** & **right edge-edge connections**





- **Middle connections** handle rare cases

2 new reconstructions algorithms

- **Without any connections**: Only classify edges
- **Without left and right connections**: classify edges and middle connections

Metric	Category	Allen	ETX4VELO	No connections	Left & right connections
Efficiency	**Long no-electrons**	99.35%	99.35%	99.17%	99.34%
	Long no-electrons No shared hits	99.45%	99.46%	99.34%	99.45%
	Long no-electrons With shared hits	94.46%	97.10%	96.53%	96.82%
	Long electrons	95.21%	98.08%	98.27%	97.76%
	Long from strange	97.53%	97.43%	96.53%	97.40%
Fake rate		2.19%	1.01%	1.13%	0.88%

Results

- Middle connections **helpful** even for non-shared hit situations
- Left & right connections could be discarded