



Track Finding in the Velo at LHCb using Graph Neural Networks

Anthony Correia



In collaboration with



1 Track Finding at LHCb

a Particle Collider

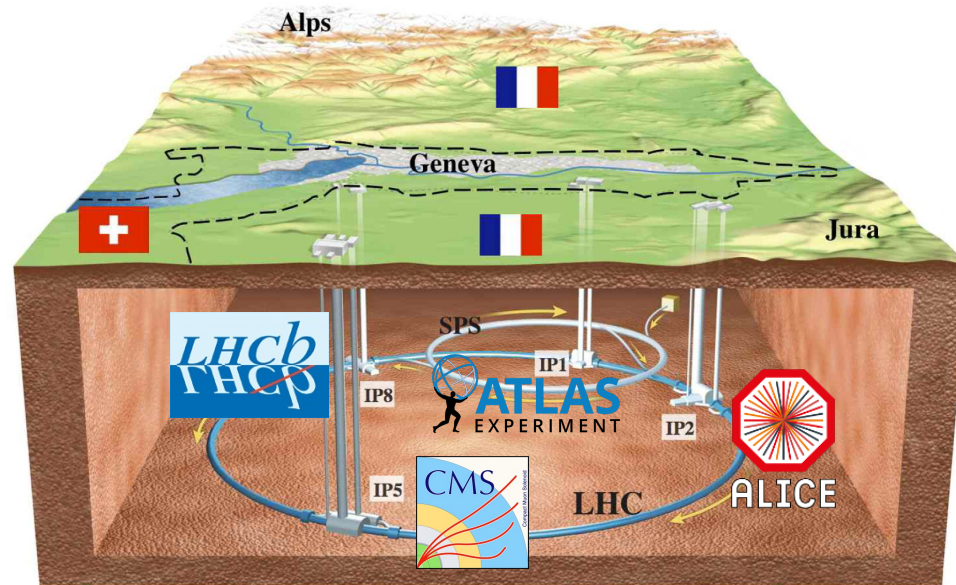


Large Hadron Collider at CERN

Largest particle accelerator (27 km)

Largest collision energy (13.6 TeV)

Experiments with **largest** author lists



<https://doi.org/10.1016/j.pnpnp.2012.03.001>

1 Track Finding at LHCb

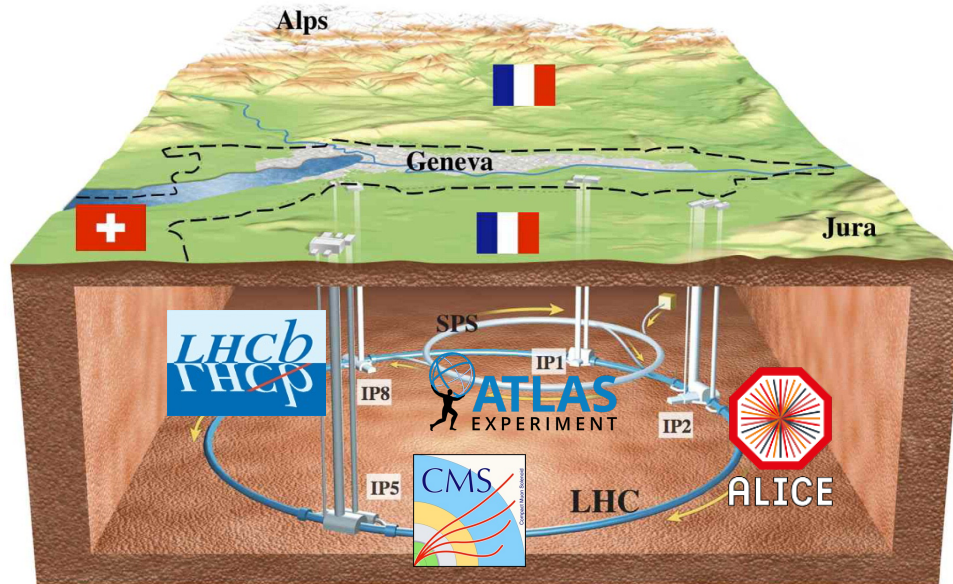
a Particle Collider



LHCb

Large Hadron Collider at CERN
Largest particle accelerator (27 km)
Largest collision energy (13.6 TeV)
 Experiments with **largest** author lists

Focus on decays of particles with *b* and *c* quarks.



<https://doi.org/10.1016/j.pnpnp.2012.03.001>

LHCb cavern in 2016



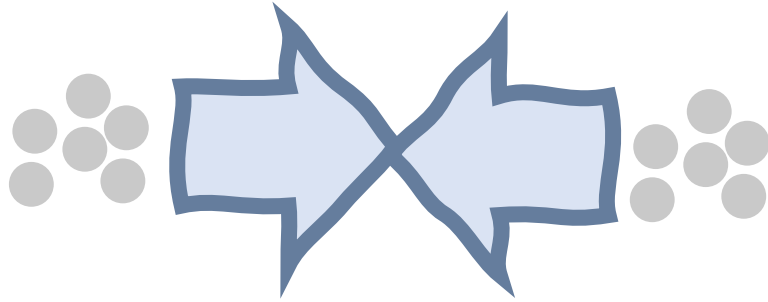
<https://home.cern/fr/node/5155>

1 Track Finding at LHCb

a Particle Collider

100 billions
of protons

100 billions
of protons



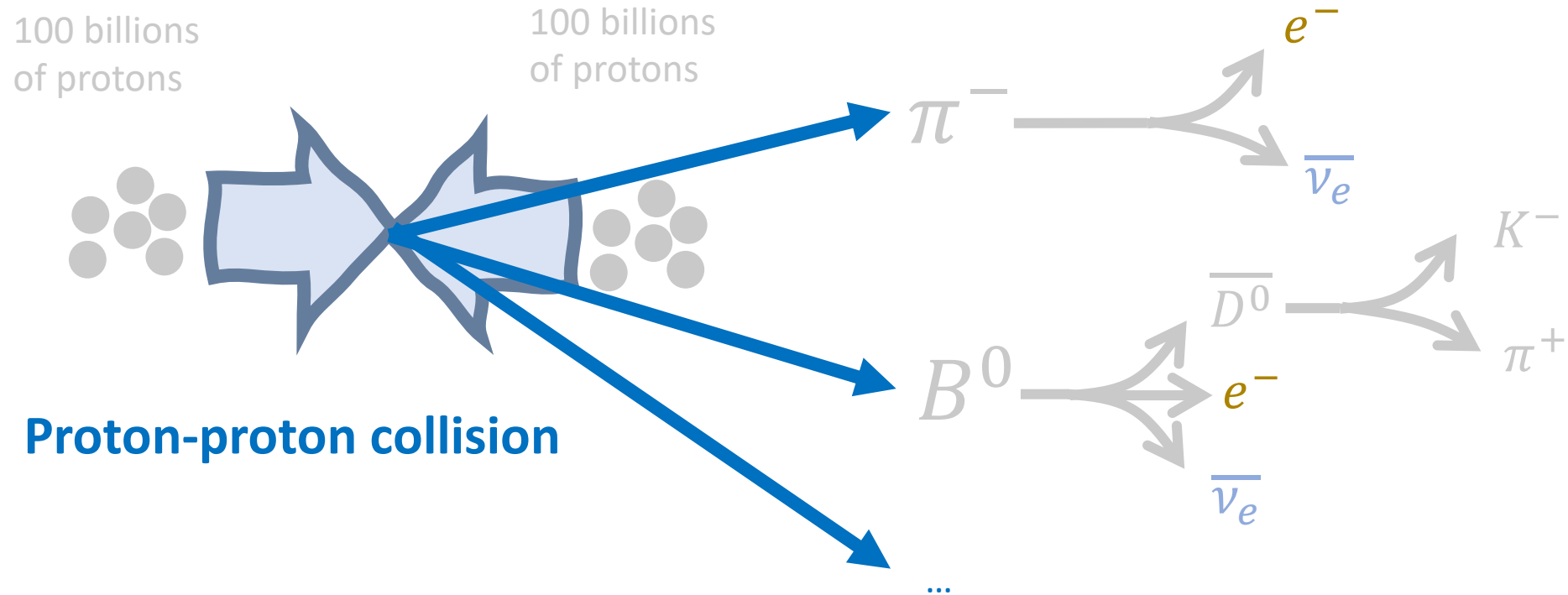
Proton-proton collision

At LHCb (run 3)

- $\sqrt{s} = 13.6 \text{ TeV}$
- **Bunch crossing rate:** 40 MHz (= 1 every 25 ns)
- ~5 proton-proton collisions / bunch crossing

1 Track Finding at LHCb

a Particle Collider

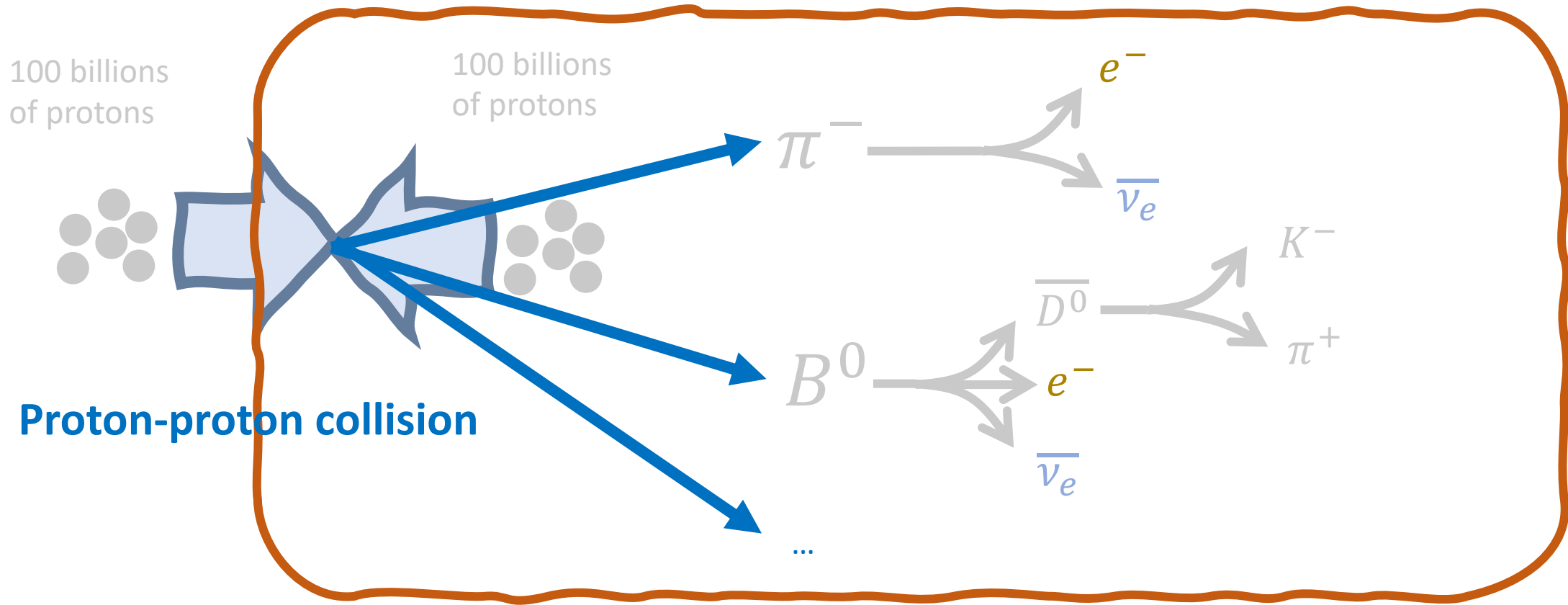


At LHCb (run 3)

- $\sqrt{s} = 13.6 \text{ TeV}$
- **Bunch crossing rate:** 40 MHz (= 1 every 25 ns)
- ~5 proton-proton collisions / bunch crossing

1 Track Finding at LHCb

a Particle Collider



At LHCb (run 3)

- $\sqrt{s} = 13.6 \text{ TeV}$
- **Bunch crossing rate:** 40 MHz (= 1 every 25 ns)
- ~5 proton-proton collisions / bunch crossing

Particle detector

1 Track Finding at LHCb

b LHCb Detector at Run 3

**Tracking
detector**

Measure trajectories

RICH detectors

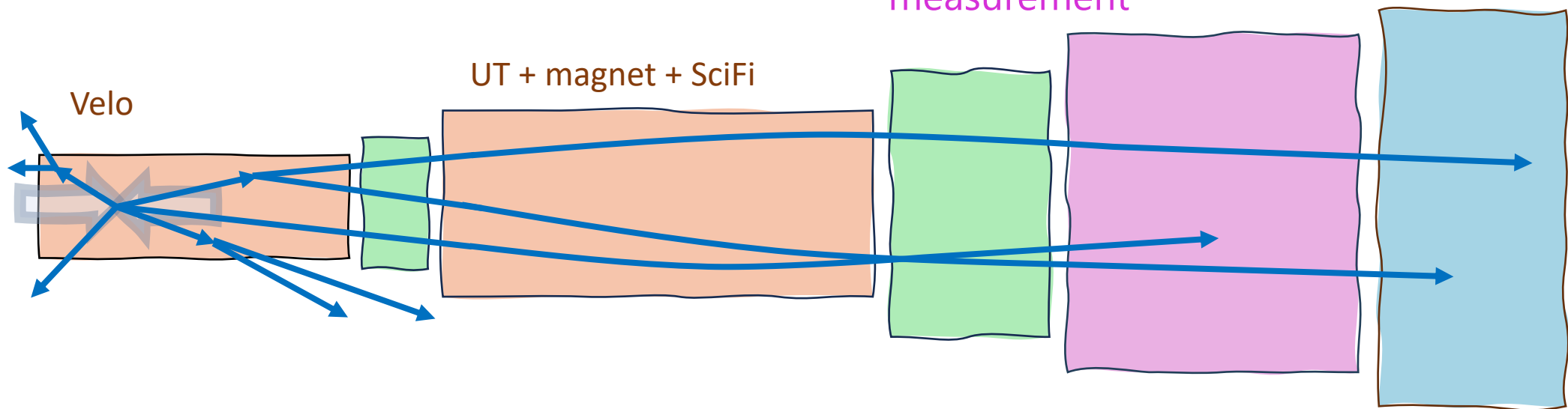
Identification

Calorimeters

Energy
measurement

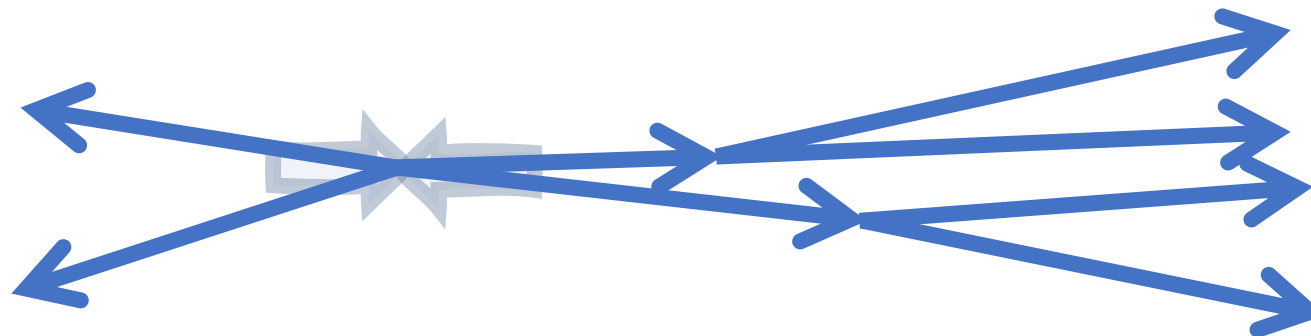
Muon Chamber

Identification



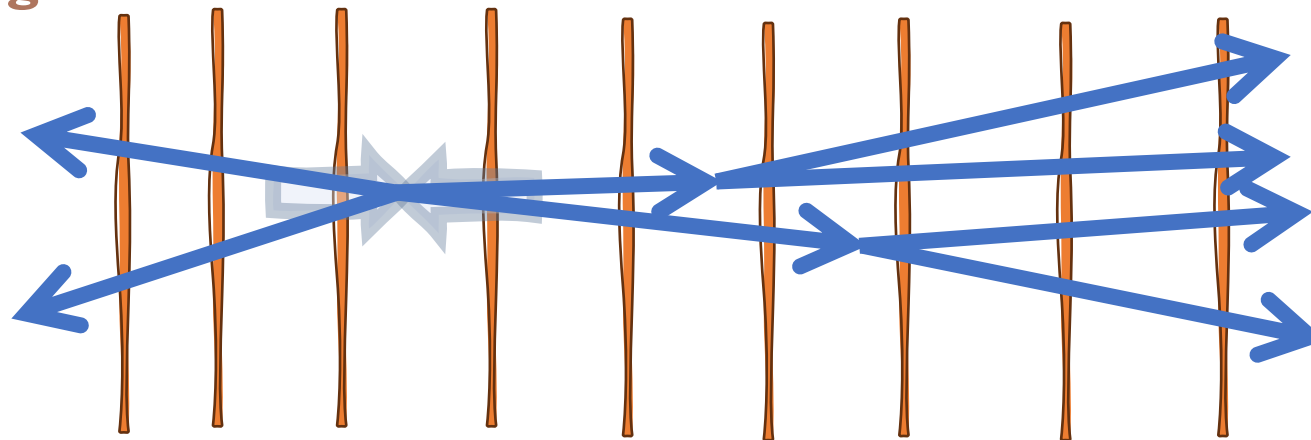
1 Track Finding at LHCb

c Track Finding



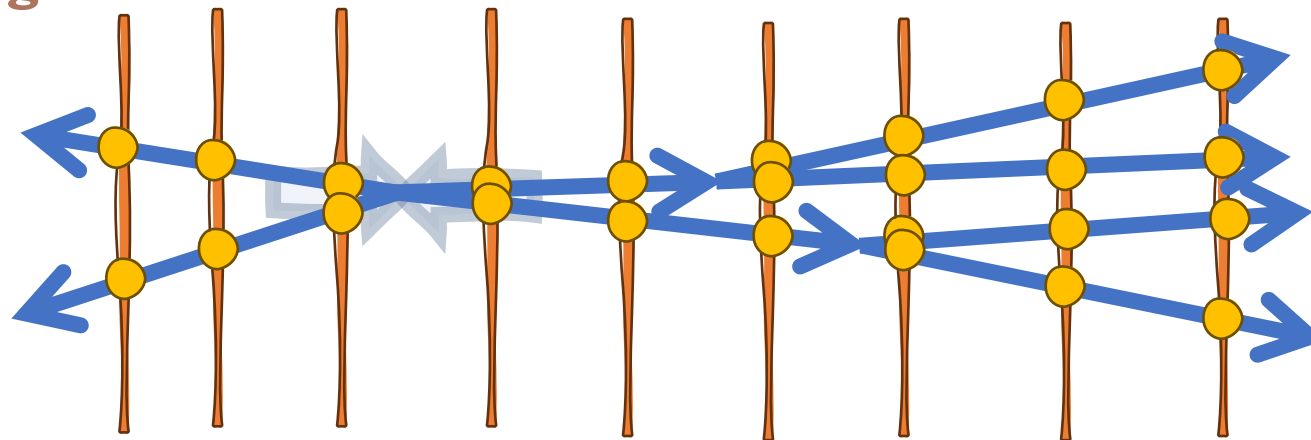
1 Track Finding at LHCb

C Track Finding



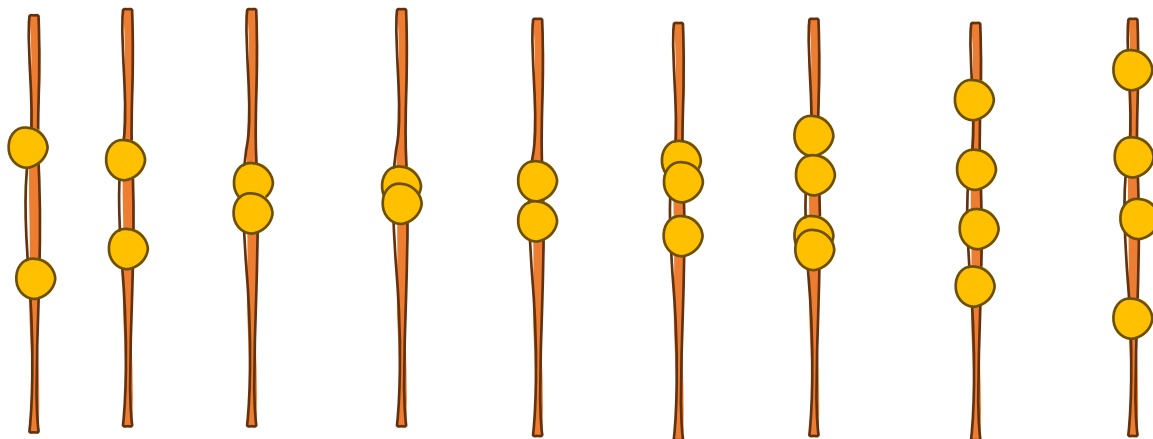
1 Track Finding at LHCb

c Track Finding



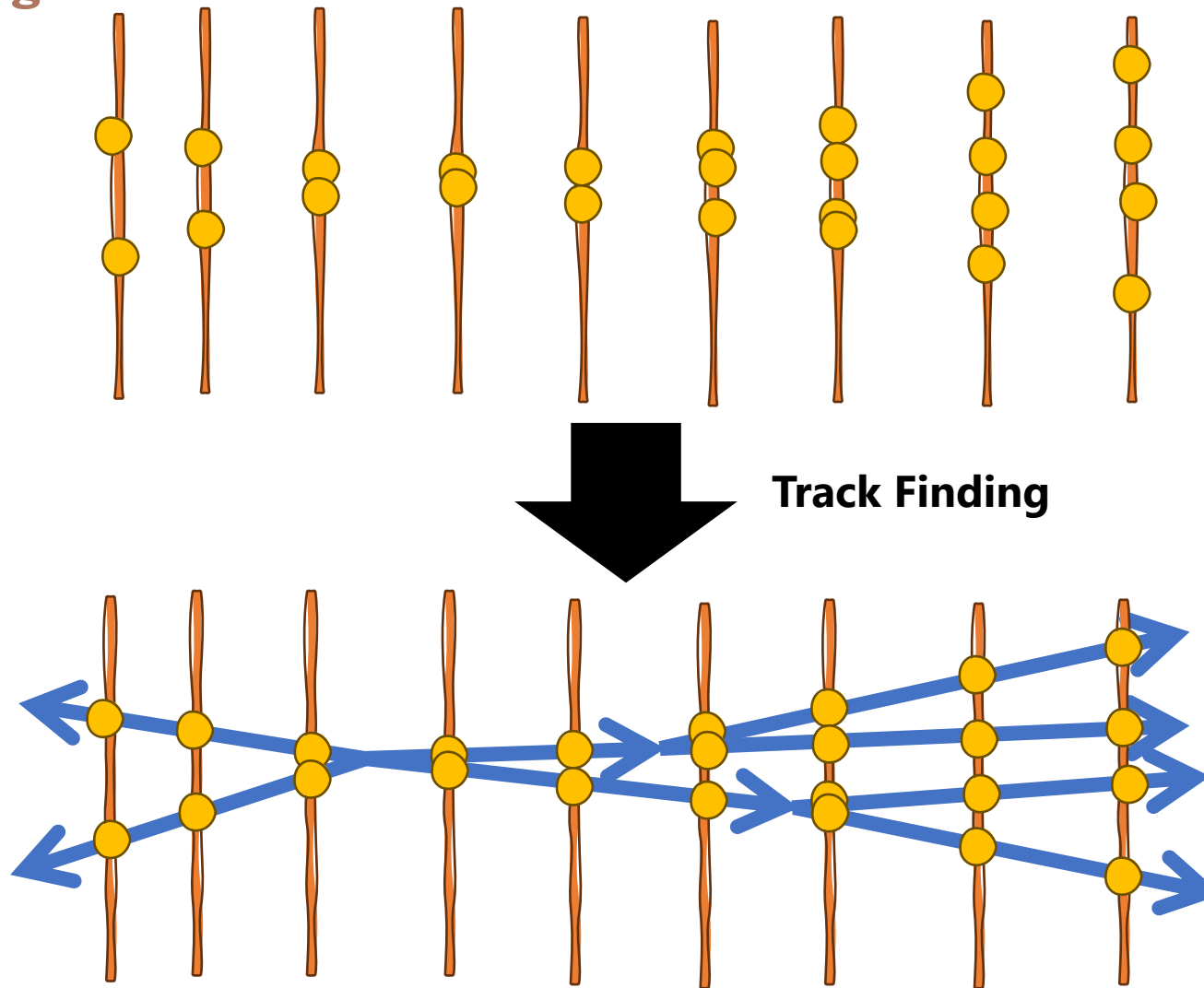
1 Track Finding at LHCb

C Track Finding



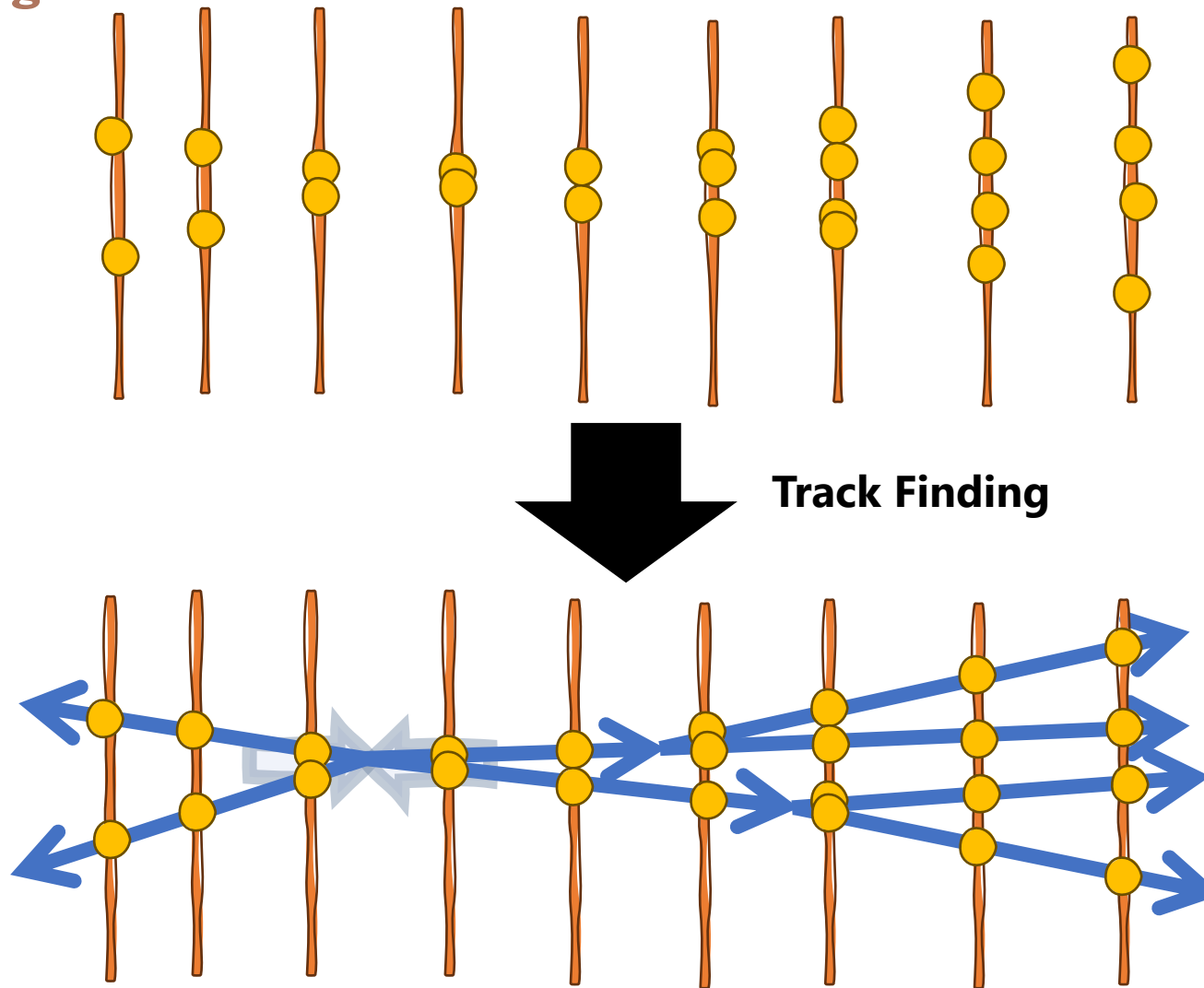
1 Track Finding at LHCb

c Track Finding



1 Track Finding at LHCb

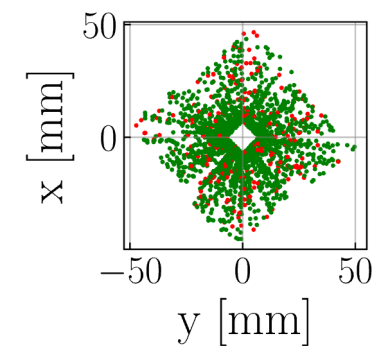
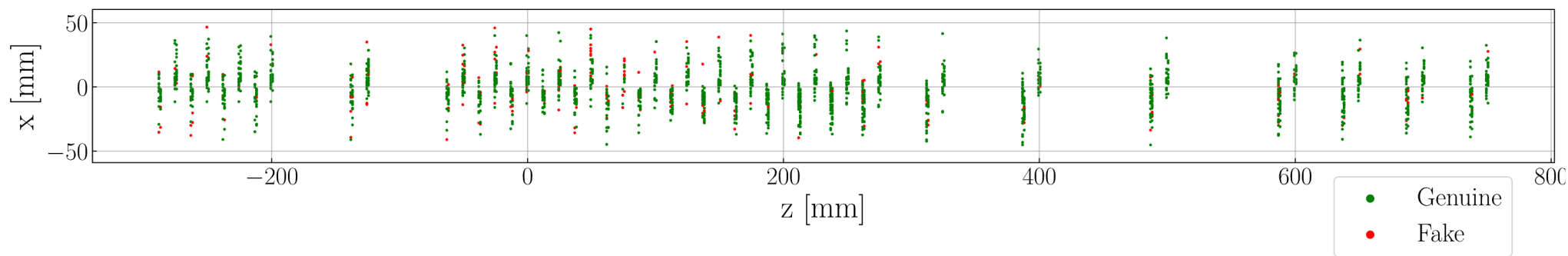
C Track Finding



1 Track Finding at LHCb

C Track Finding

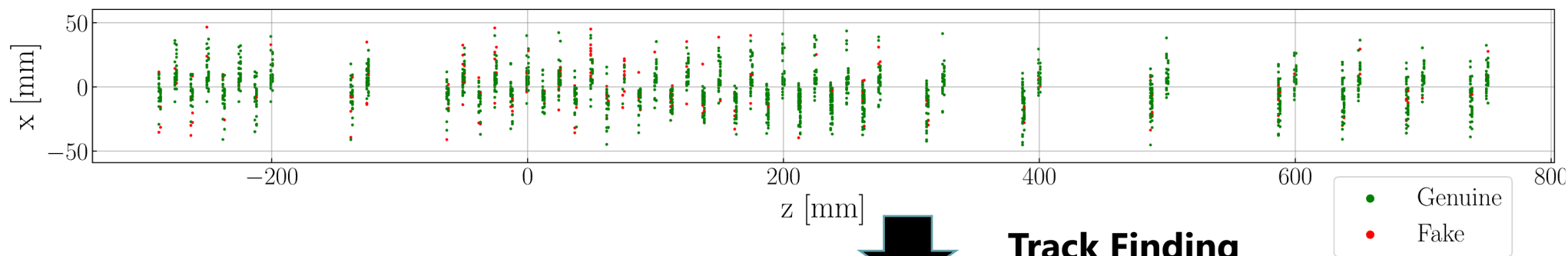
- 26 layers
- 2000 hits
- 13% fake hits



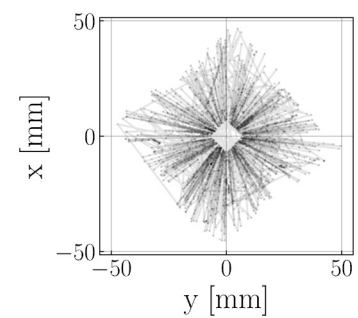
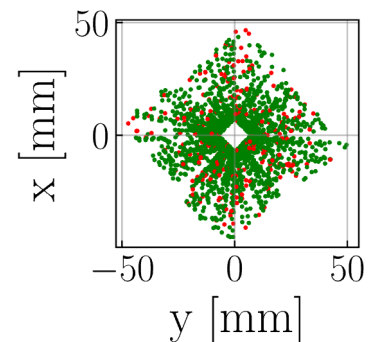
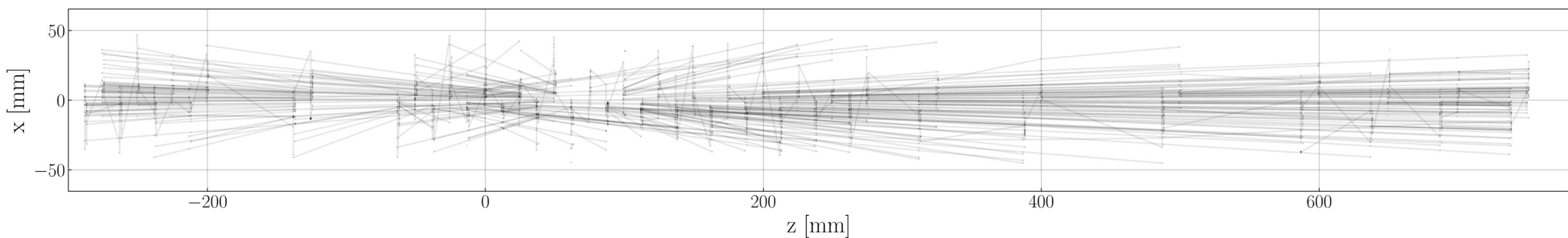
1 Track Finding at LHCb

C Track Finding

- 26 layers
- 2000 hits
- 13% fake hits



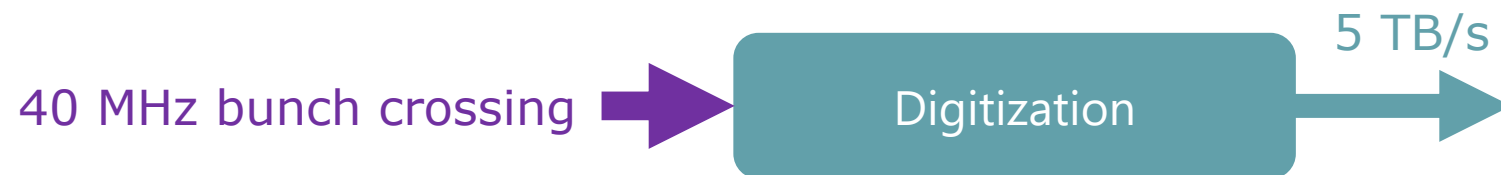
Track Finding



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

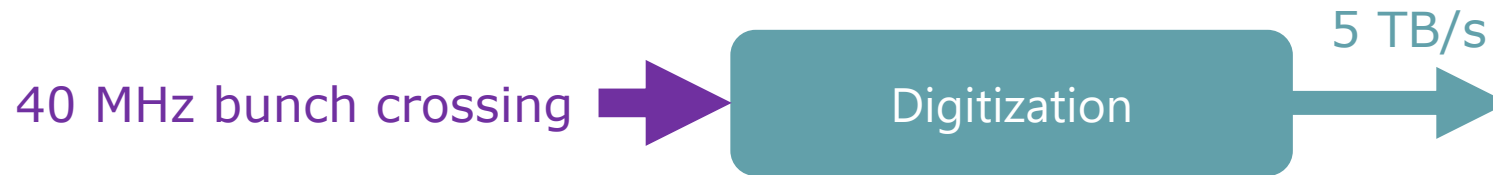
1 Track Finding at LHCb

d Trigger



1 Track Finding at LHCb

d Trigger

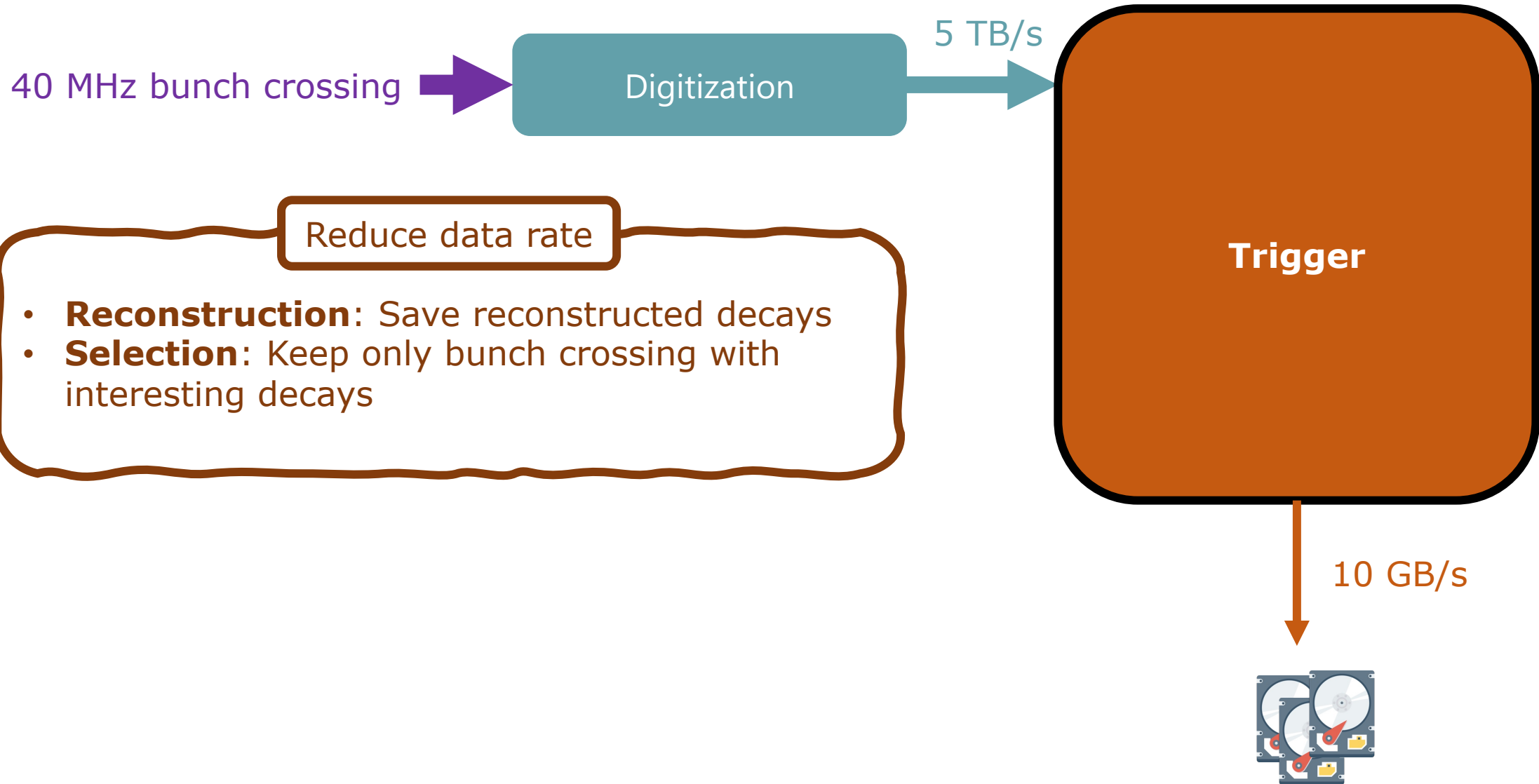


Reduce data rate

- **Reconstruction:** Save reconstructed decays
- **Selection:** Keep only bunch crossing with interesting decays

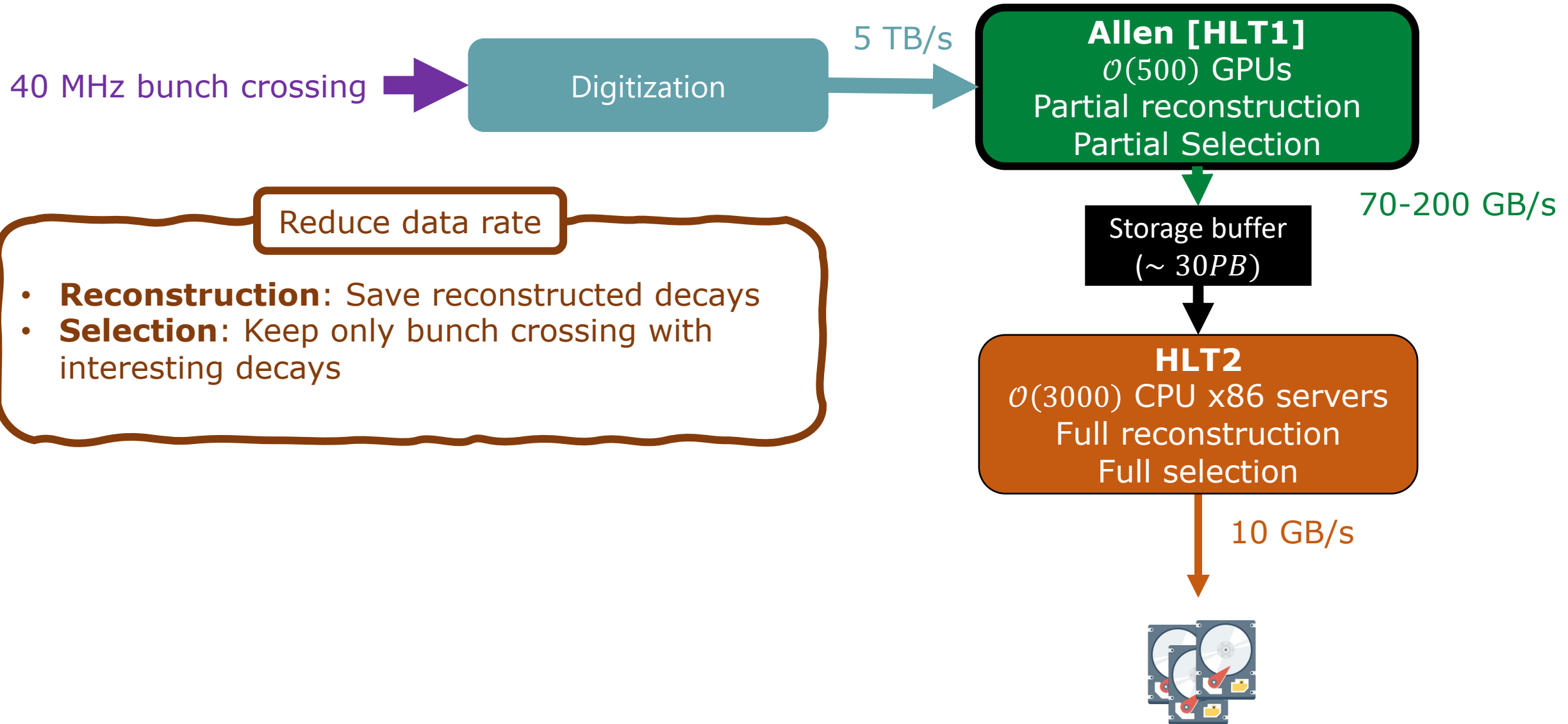
1 Track Finding at LHCb

d Trigger



1 Track Finding at LHCb

d Trigger



1 Track Finding at LHCb

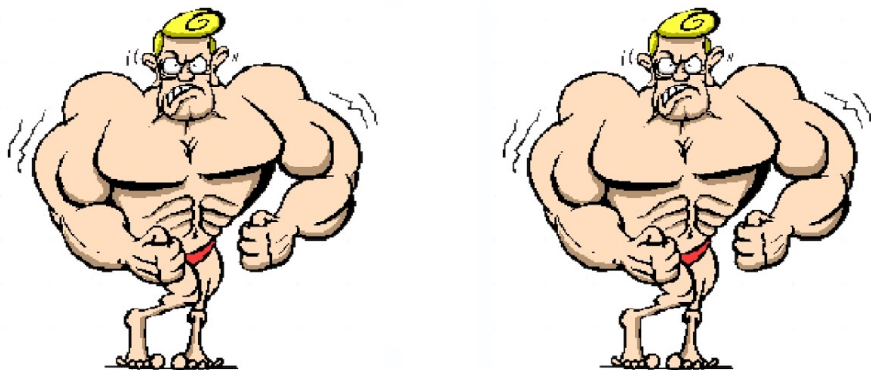
d Trigger

CPU

Central Processor Unit

O(10-100) fast cores running in parallel

Optimised for **latency**
($\equiv$ duration of a single operation)

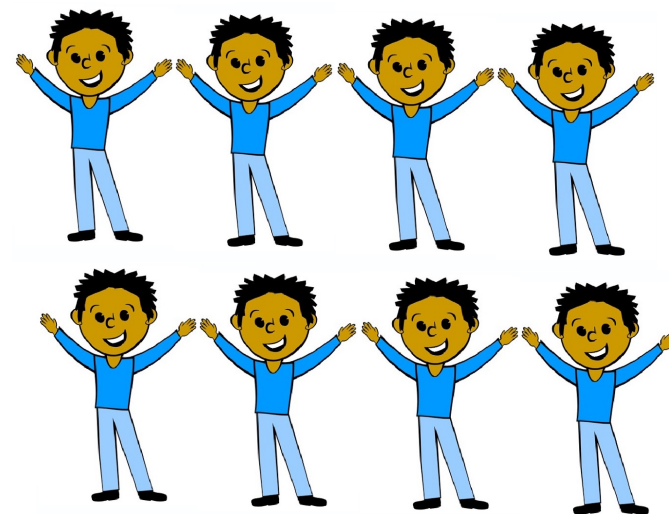


GPU

Graphical Processor Unit

O(1000-10000) slow cores running in parallel

Optimised for **throughput**
($\equiv$ # operations / second)



Programming Language for NVIDIA GPUs: CUDA

GPU allows to accomodate for high input rate

1 Track Finding at LHCb

e Motivation

- **Velo track finding in Allen:** $\mathcal{O}(10\%)$ 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} .
⇒ **Neural-Network Algorithms?**

1 Track Finding at LHCb

e Motivation

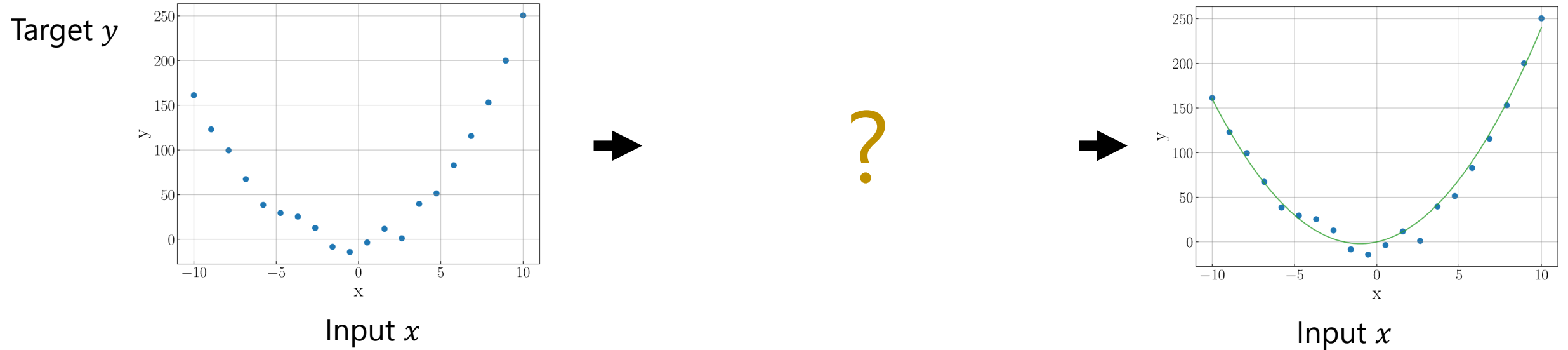
- **Velo track finding in Allen:** $\mathcal{O}(10\%)$ total reconstruction time
- **High-Luminosity Phase of LHCb (Run 5):** 42 visible pp collisions per bunch crossing
 - $\times 8$ particles
 - Higher detector occupancy
- **Classical algorithms** scale worse than quadratically with n_{hits} .
⇒ **Neural-Network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

2 Neural Network Introduction

a Fitting a One-Dimensional Function

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

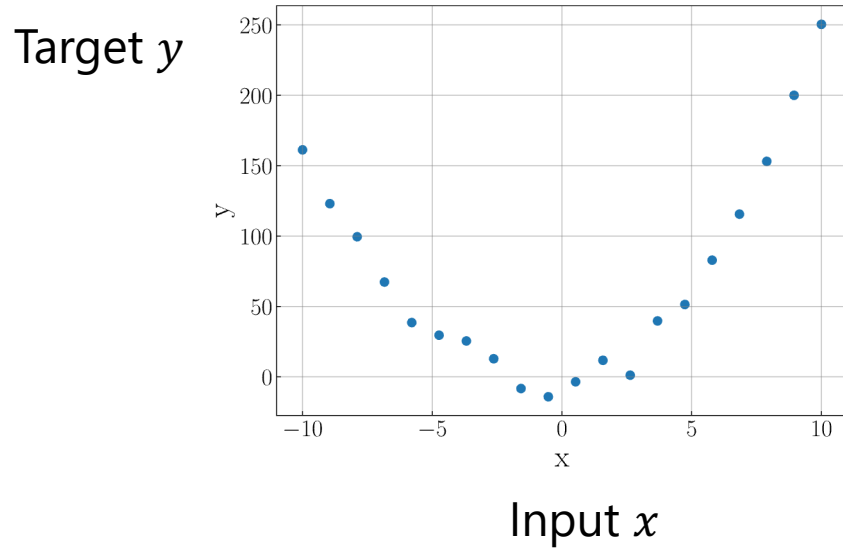


2 Neural Network Introduction

11

a Fitting a One-Dimensional Function

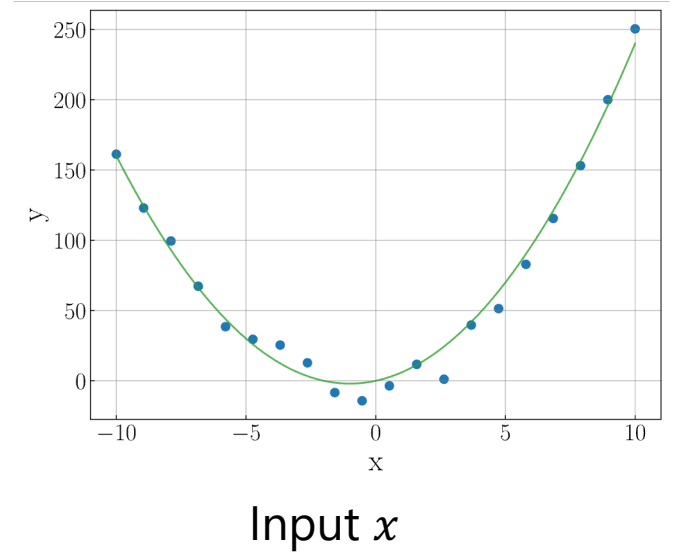
- **Regression problem:** Predict $y \in \mathbb{R}$ from $x \in \mathbb{R}$



?



Predicted target $\hat{y}$



1. Model selection

Choose best model f_{θ}

2. Training

Find optimal parameters θ^*

3. Inference

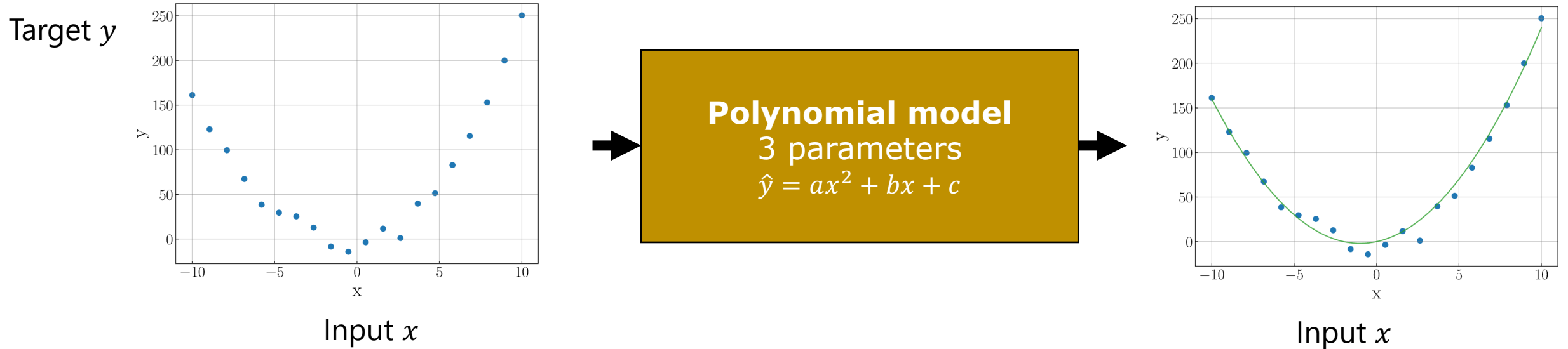
Use the model for other examples

2 Neural Network Introduction

11

a Fitting a One-Dimensional Function

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



1. Model selection

Choose best model f_{θ}

2. Training

Find optimal parameters θ^*

3. Inference

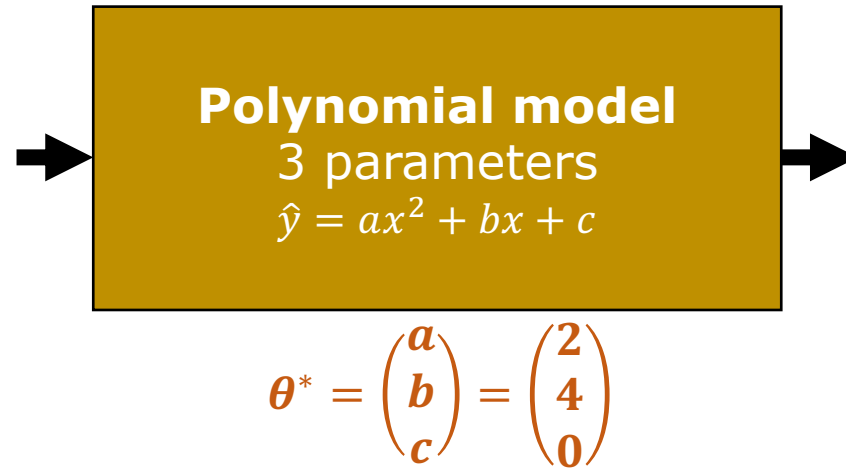
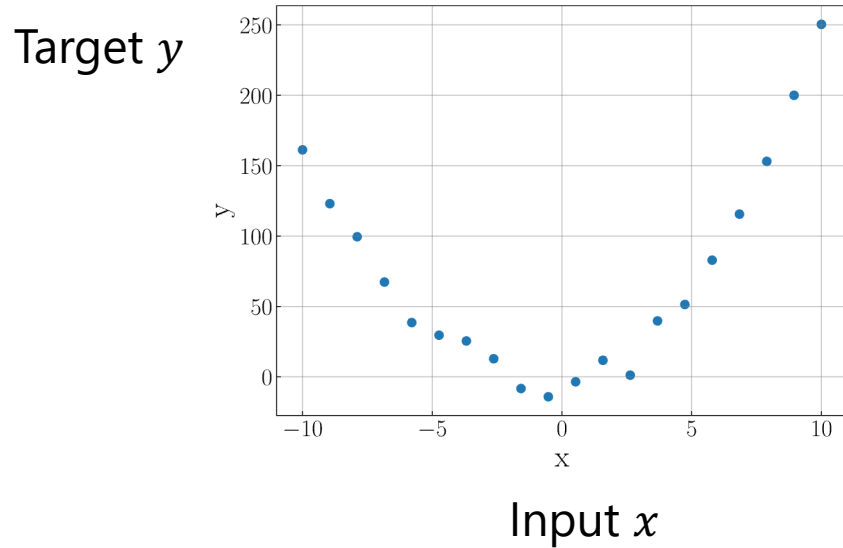
Use the model for other examples

2 Neural Network Introduction

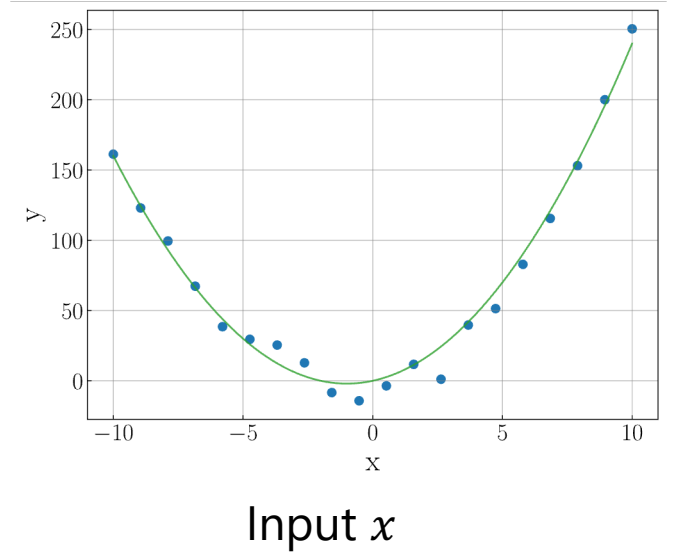
11

a Fitting a One-Dimensional Function

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



Predicted target $\hat{y}$



1. Model selection

Choose best model f_{θ}

2. Training

Find optimal parameters θ^*

3. Inference

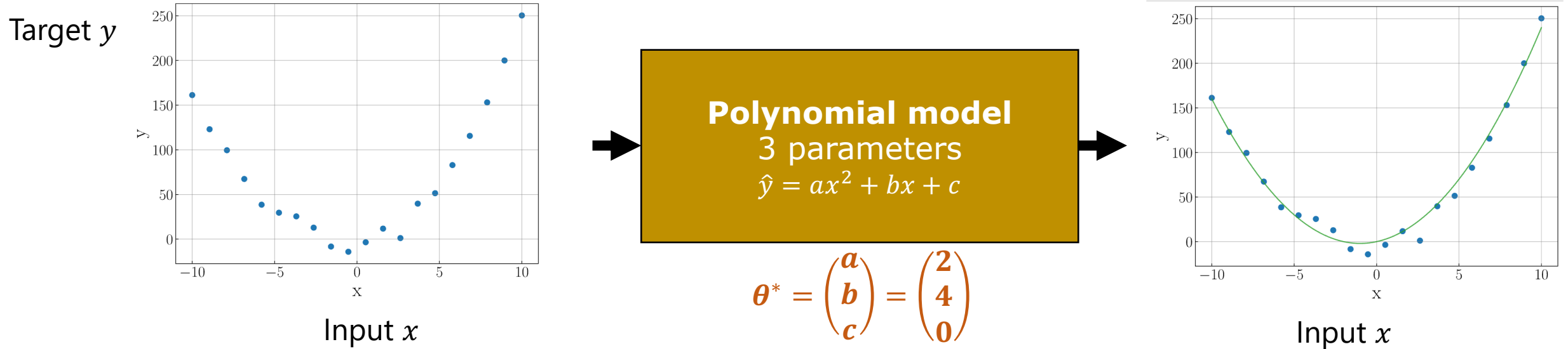
Use the model for other examples

2 Neural Network Introduction

11

a Fitting a One-Dimensional Function

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



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

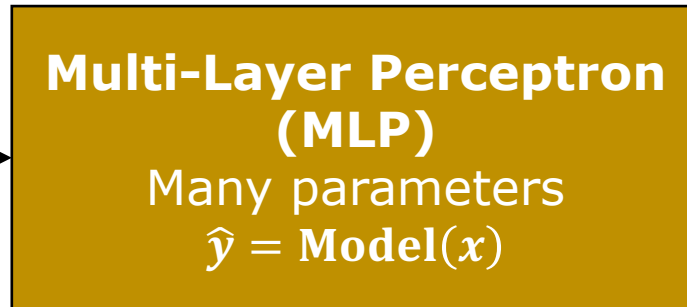
0.2

2 Neural Network Introduction

12

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

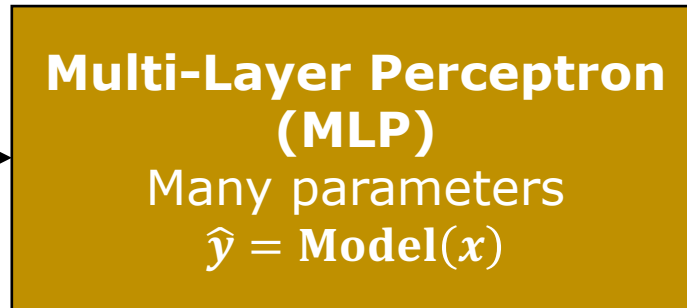
1. **Model Choice: Multi-Layer Perceptron (MLP)** [many parameters]

2 Neural Network Introduction

12

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

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

2. **Training:** Needs for:

- Much more **data**
- Much more **computing resource**

2 Neural Network Introduction

12

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

13

C Different Types of Neural Networks

Input

Pictures



Sequences: text

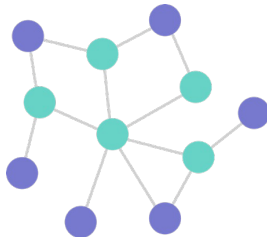


, audio



, ...

Graph



Neural Network Model Family



Convolutional Neural Network (CNN)



Recursive Neural Network (RNN)

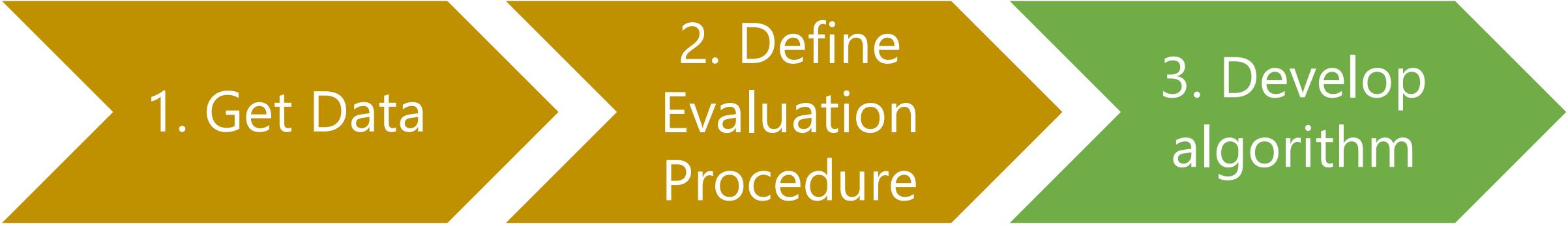


Graph Neural Network (GNN)

- **Deep Learning Library:**



PyTorch



```
graph LR; A[1. Get Data] --> B[2. Define Evaluation Procedure]; B --> C[3. Develop algorithm];
```

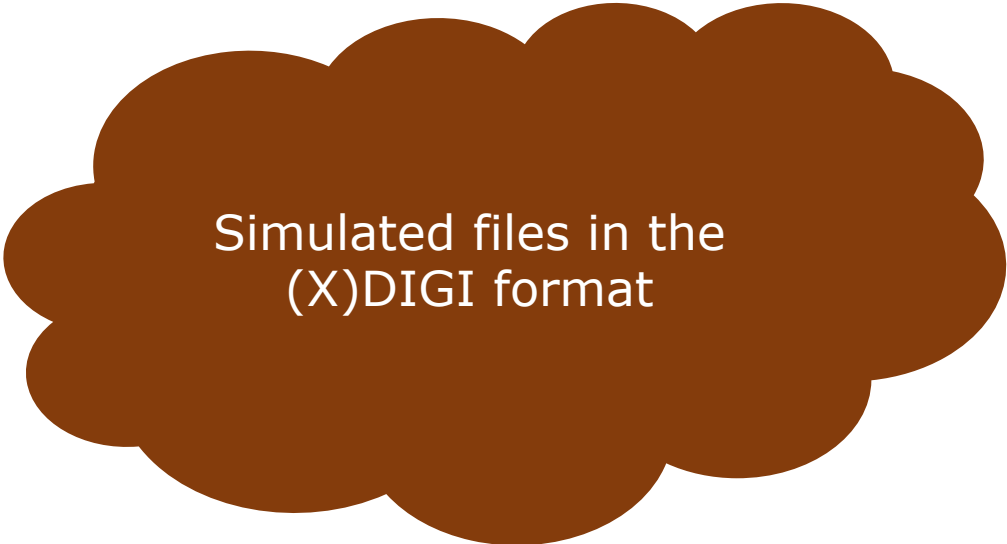
1. Get Data

2. Define
Evaluation
Procedure

3. Develop
algorithm

a Data, more data

CERN grid



Simulated files in the
(X)DIGI format

digout library



"local"

Table of hits

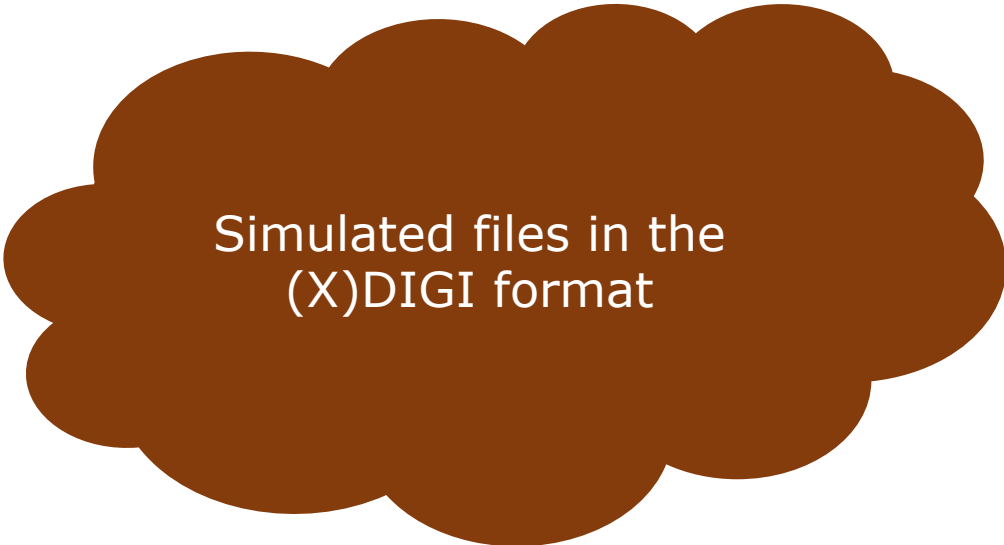
Table of hits-particles

Table of particles

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

a Data, more data

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**

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

⇒ **1.5M bunch crossing events available**

3 Before Training

b Performance evaluation

- Physics performance

$$\text{Efficiency} = \frac{\# \text{ reconstructed particles}}{\# \text{ reconstructible particles}}$$

$$\text{Fake rate} = \frac{\# \text{ fake tracks}}{\# \text{ tracks}}$$

- Computational performance

$$\text{Throughput} = \# \text{ bunch crossing events processed} / \text{unit of time}$$

Metric	Allen
Non-electron efficiency	99.35%
Electron efficiency	96.55%
Fake rate	2.19%
Throughput <i>NVIDIA GeForce RTX 5080 Ti</i>	595 kHz

evaluated on 1000 events

3 Before Training

b Performance evaluation

- Physics performance

$$\text{Efficiency} = \frac{\# \text{ reconstructed particles}}{\# \text{ reconstructible particles}}$$

$$\text{Fake rate} = \frac{\# \text{ fake tracks}}{\# \text{ tracks}}$$

- Computational performance

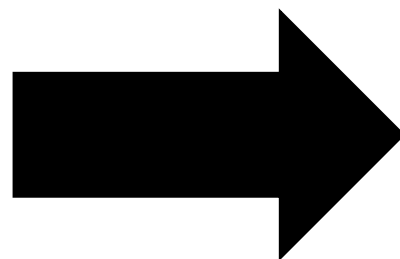
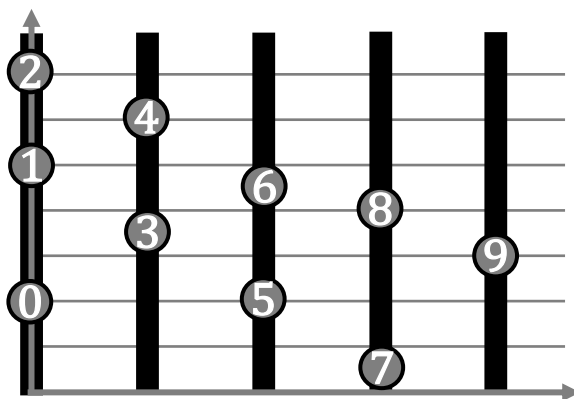
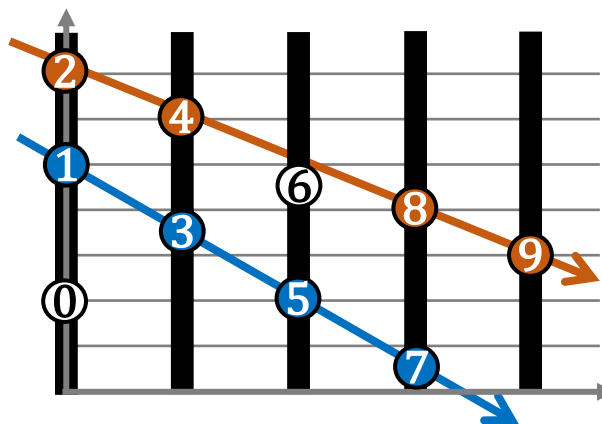
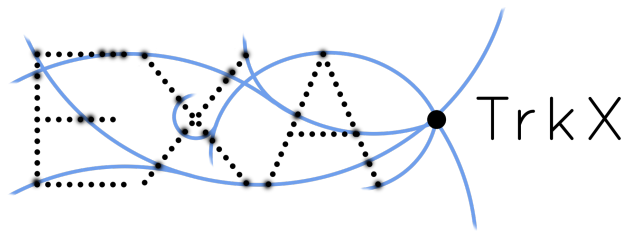
Throughput = # bunch crossing events processed / unit of time

Metric	Allen
Non-electron efficiency	99.35%
Electron efficiency	96.55%
Fake rate	2.19%
Throughput <i>NVIDIA GeForce RTX 5080 Ti</i>	595 kHz

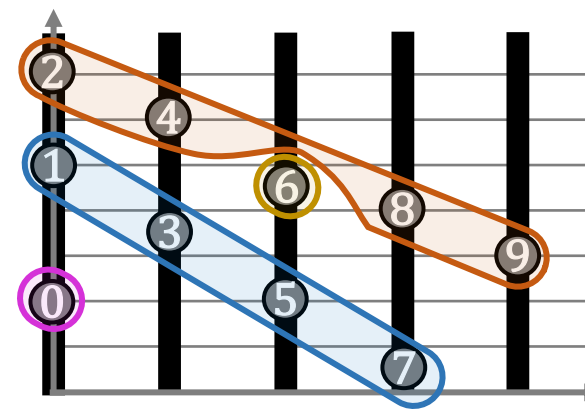
evaluated on 1000 events

Strategy

1. Can we reach the **physics performance**?
2. Given the physics performance, can we reach the **throughput**?



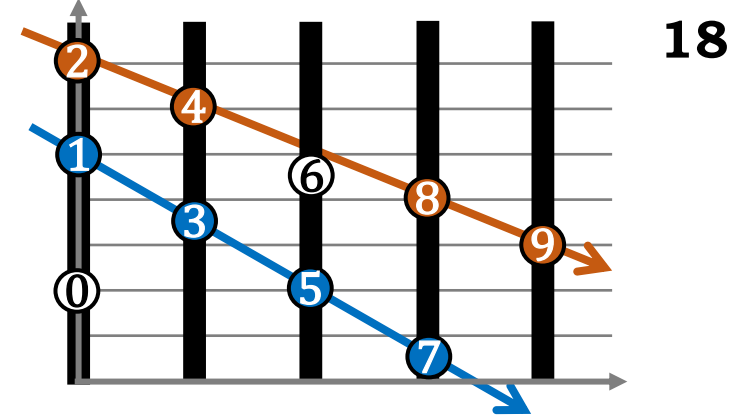
Track Finding



4 Neural Network Introduction

a Why GNNs

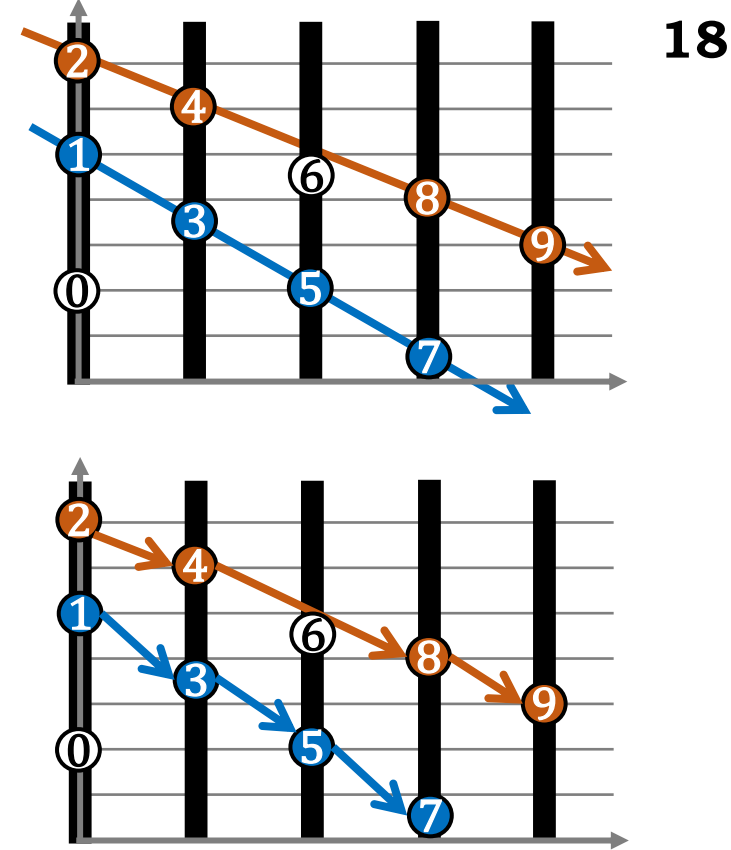
- **Graph $\mathcal{G}$** is defined as
 - Set of **nodes** $\mathcal{V}$: indexed from 0 to 9
 - Set of **edges** $\mathcal{E} \equiv$ connection between nodes



4 Neural Network Introduction

a Why GNNs

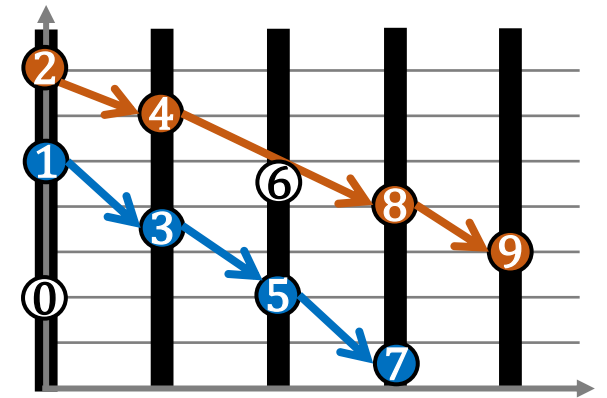
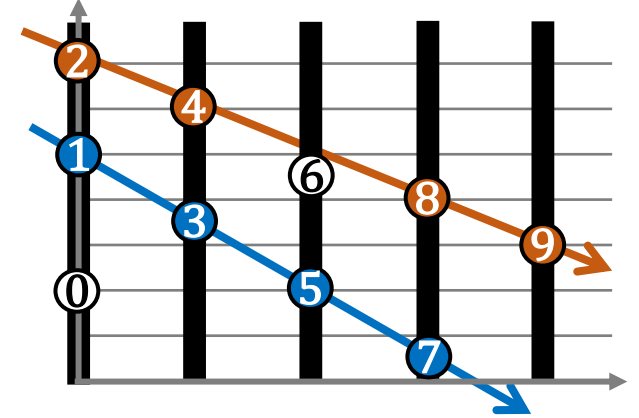
- **Graph $\mathcal{G}$** is defined as
 - Set of **nodes** $\mathcal{V}$: indexed from 0 to 9
 - Set of **edges** $\mathcal{E} \equiv$ connection between nodes
- **Graph** $\equiv$ Natural way of representing an event in a tracking detector
 - Set of **nodes**: hits
 - Set of **edges**: adjacent hits belonging to the same particle



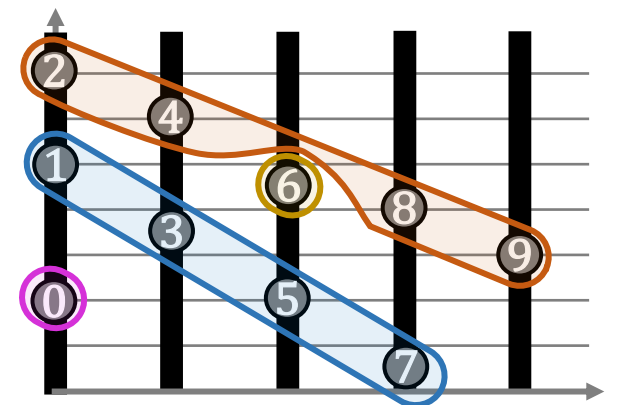
4 Neural Network Introduction

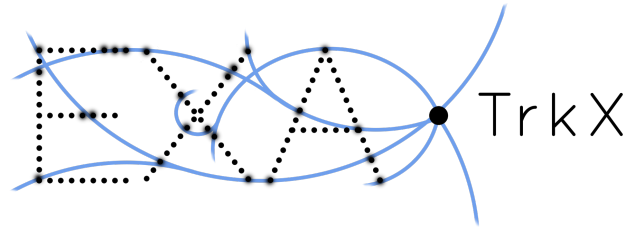
a Why GNNs

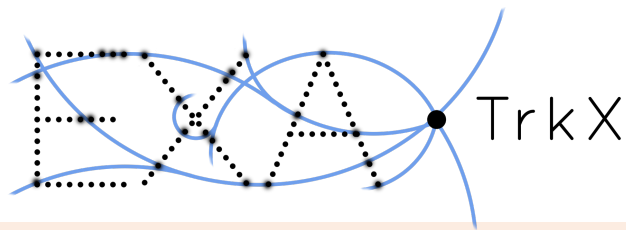
- **Graph $\mathcal{G}$** is defined as
 - Set of **nodes** $\mathcal{V}$: indexed from 0 to 9
 - Set of **edges** $\mathcal{E} \equiv$ connection between nodes
- **Graph** $\equiv$ Natural way of representing an event in a tracking detector
 - Set of **nodes**: hits
 - Set of **edges**: adjacent hits belonging to the same particle
- **Weakly Connected Component (WCC) algorithm**:
 - WCC** = Nodes indirectly connected to each other
 - $\rightarrow$ **Tracks** = WCC with at least 3 nodes



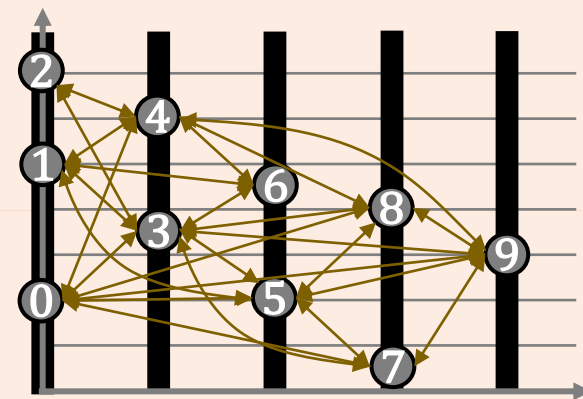
WCC algorithm 

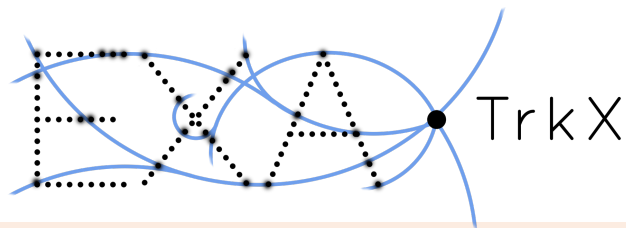




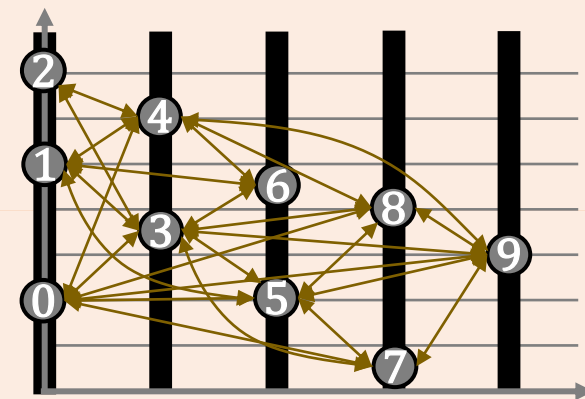
**Build a rough graph**

e.g., link every hits to hits on the next 2 planes.

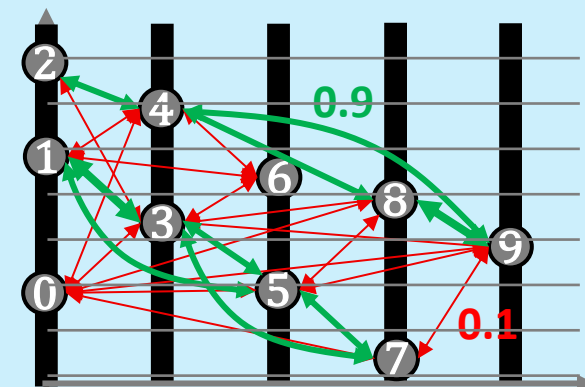


b Basic idea: Exa.TrkX Pipeline

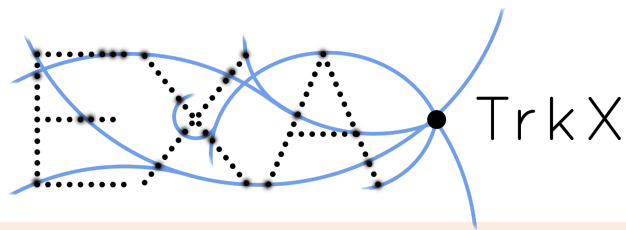
1 **Build a rough graph**
e.g., link every hits to hits on the next 2 planes.



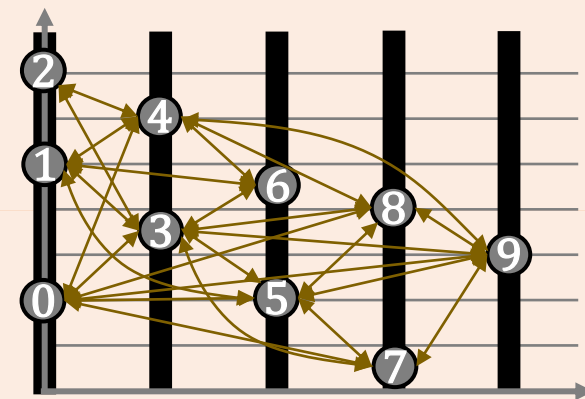
2a **Score every edge** between 0 (**fake**) and 1 (**genuine**)
using a Graph Neural Network (GNN)



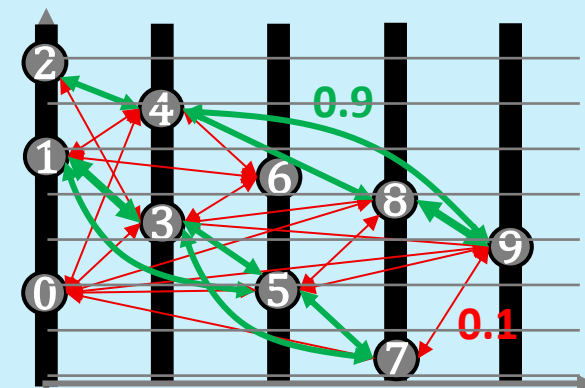
2b **Discard fake edges**
by requiring edge score $> s_{\text{edge,min}}$

b Basic idea: Exa.TrkX Pipeline

1 **Build a rough graph**
e.g., link every hits to hits on the next 2 planes.

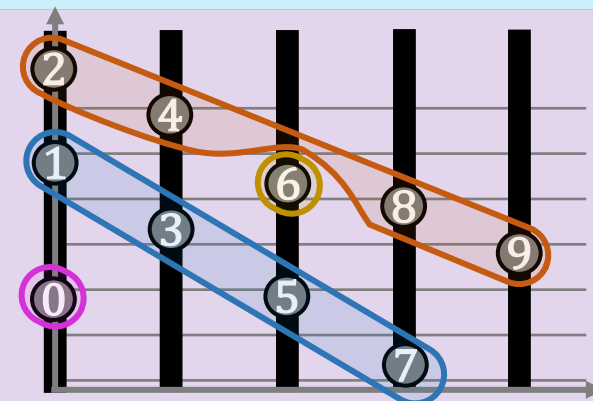


2a **Score every edge** between 0 (**fake**) and 1 (**genuine**)
using a Graph Neural Network (GNN)

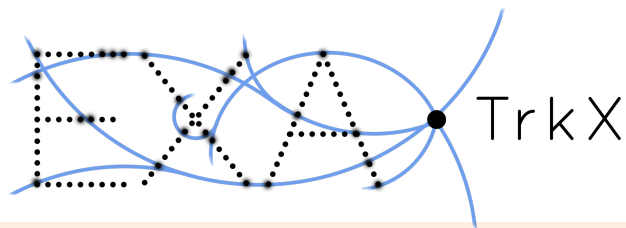


2b **Discard fake edges**
by requiring edge score $> s_{\text{edge,min}}$

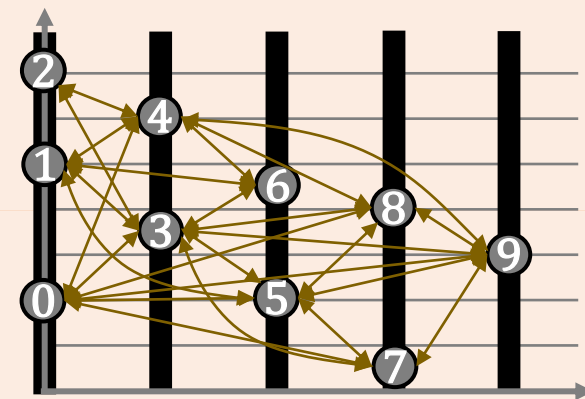
3 **Find connected components**
using a weakly connected component algorithm



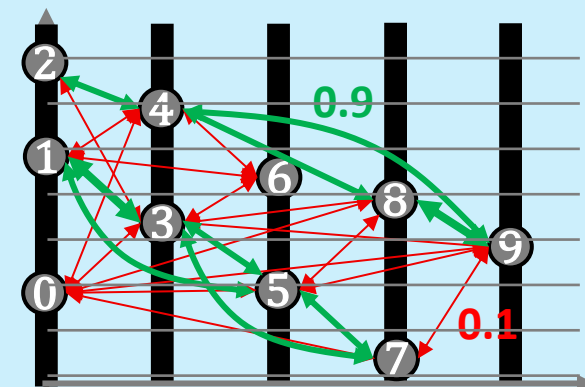
b Basic idea: Exa.TrkX Pipeline



1 **Build a rough graph**
e.g., link every hits to hits on the next 2 planes.

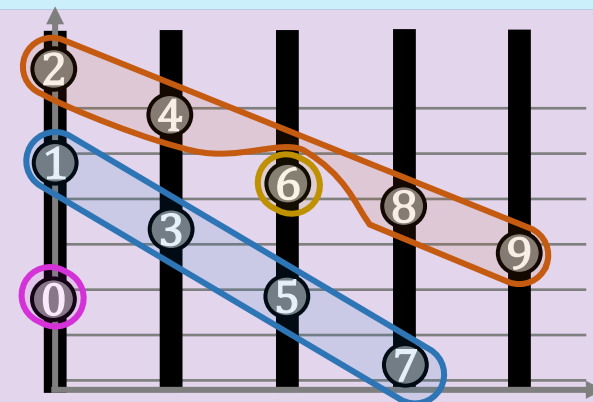


2a **Score every edge** between 0 (**fake**) and 1 (**genuine**)
using a Graph Neural Network (GNN)



2b **Discard fake edges**
by requiring edge score $> s_{\text{edge,min}}$

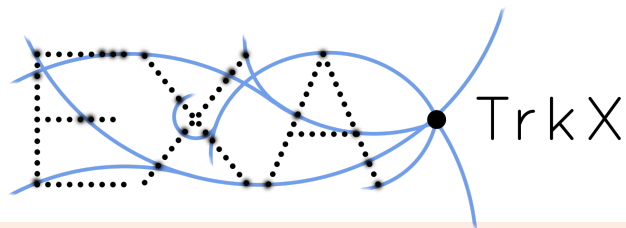
3 **Find connected components**
using a weakly connected component algorithm



Build graph

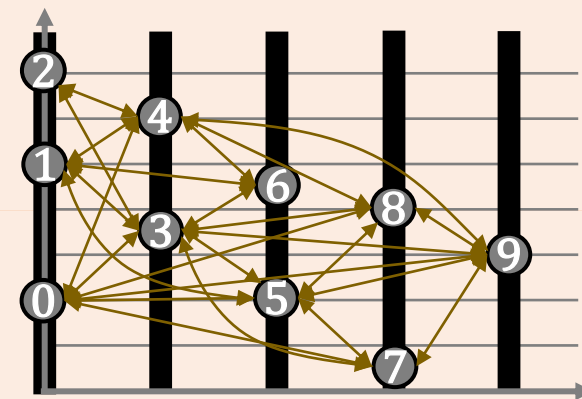
Filter edges with GNN

Find connected components

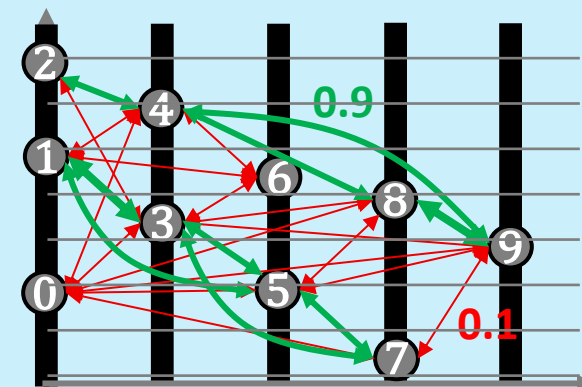
b Basic idea: Exa.TrkX Pipeline

1 **Build a rough graph**
e.g., link every hits to hits on the next 2 planes.

Actually: Neural Network + kNN

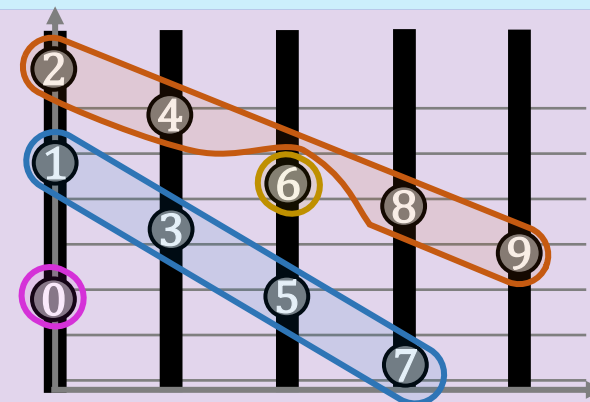


2a **Score every edge** between 0 (**fake**) and 1 (**genuine**)
using a Graph Neural Network (GNN)



2b **Discard fake edges**
by requiring edge score $> s_{\text{edge,min}}$

3 **Find connected components**
using a weakly connected component algorithm

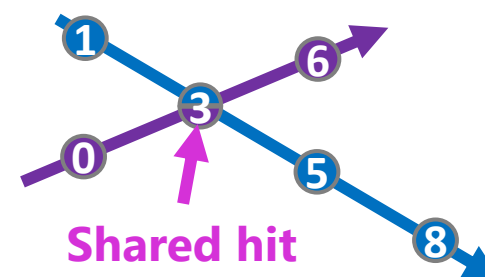


Build graph

Filter edges with GNN

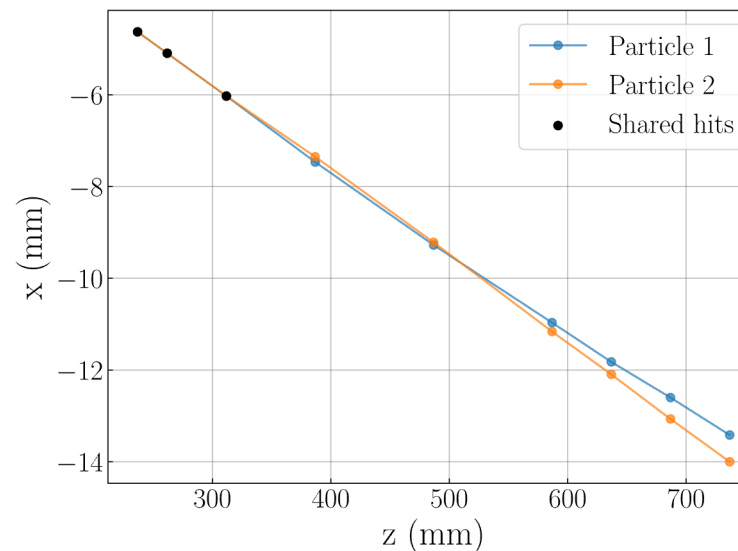
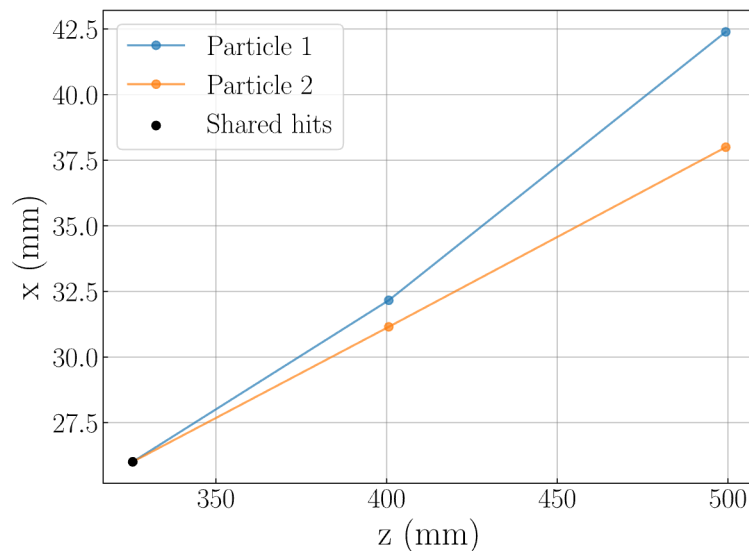
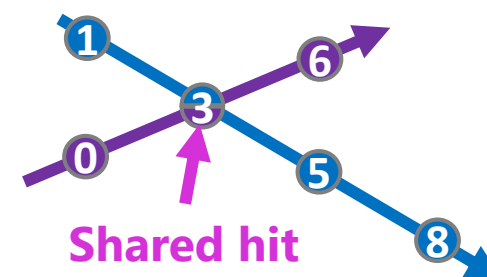
Find connected components

Shared hit: hit that belongs to > 1 reconstructible particle



C Issue with shared hits

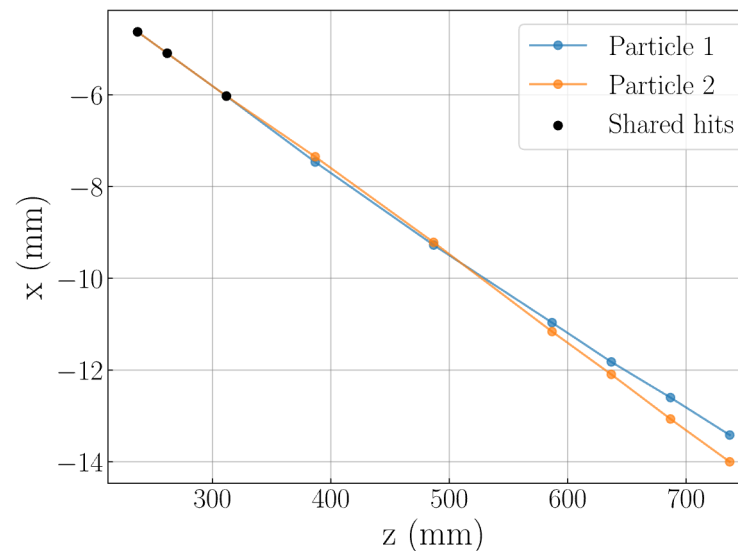
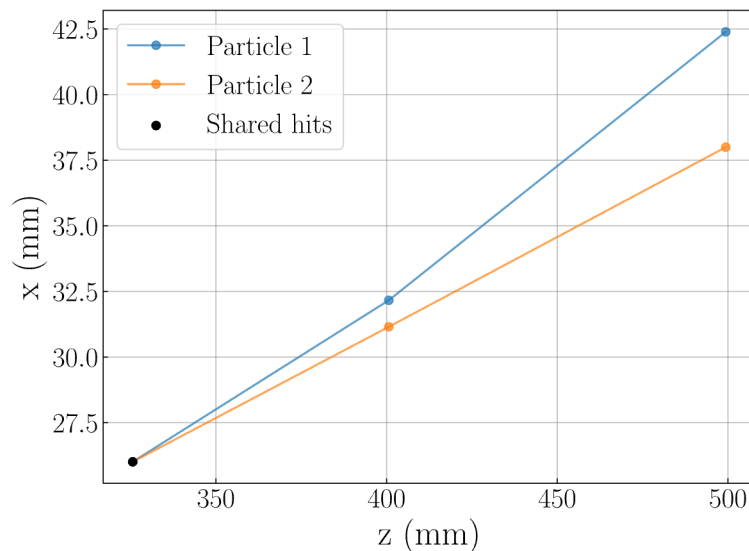
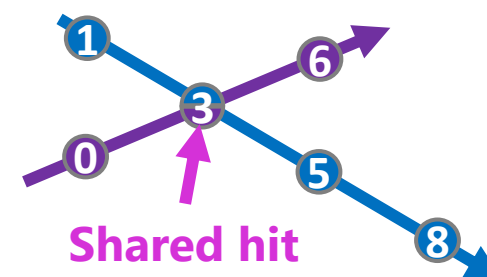
Shared hit: hit that belongs to > 1 reconstructible particle



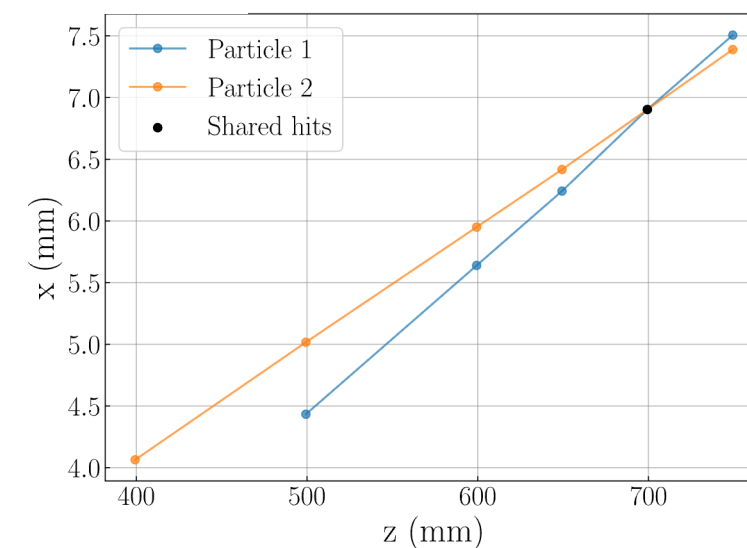
- Electrons have many **shared hits**
 - Typically e^-e^+ **pairs** sharing their **first hit(s)**
 - Correspond to **photon conversion in detector material** $\gamma \rightarrow e^-e^+$

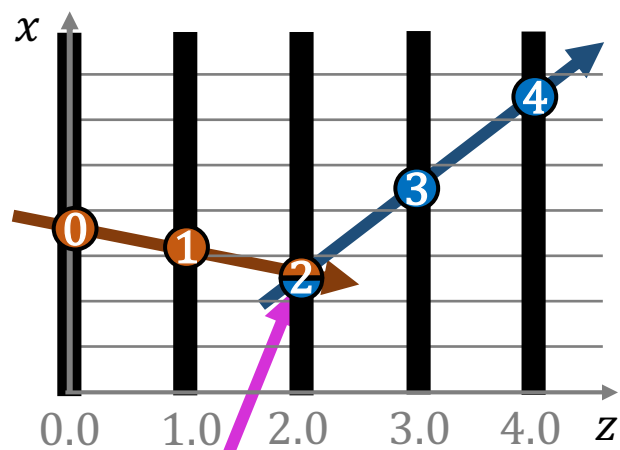
C Issue with shared hits

Shared hit: hit that belongs to > 1 reconstructible particle

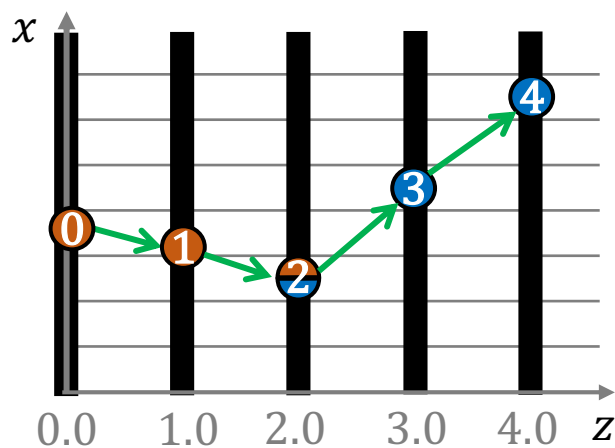


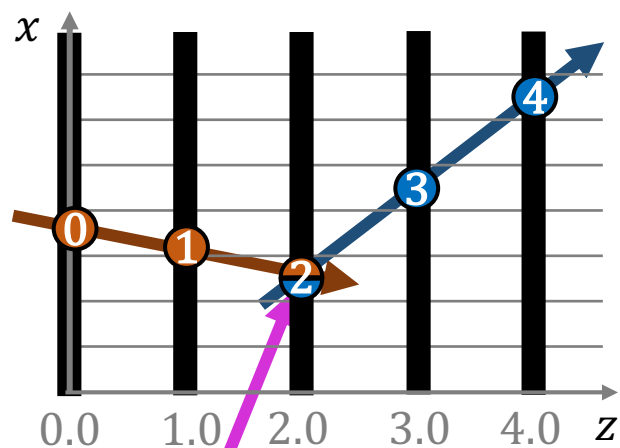
- Electrons have many **shared hits**
 - Typically e^-e^+ **pairs** sharing their **first hit(s)**
 - Correspond to **photon conversion in detector material** $\gamma \rightarrow e^-e^+$
- Other particle categories have **shared hits** too!
 - e.g., particle crossing



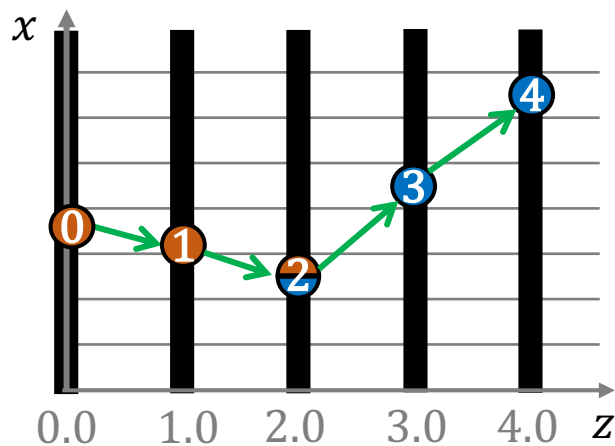
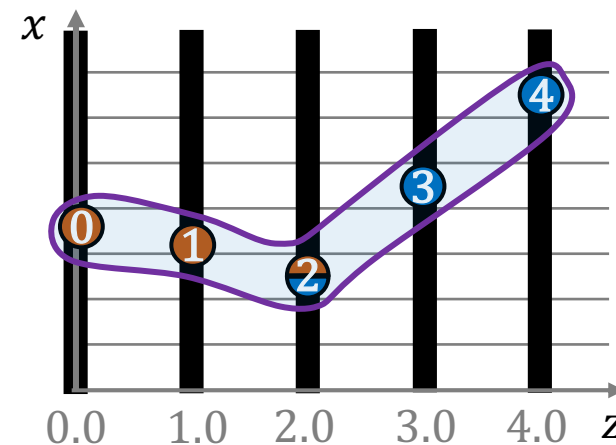
C Issue with shared hits

Shared hit



C Issue with shared hits

Shared hit

WCC
algo

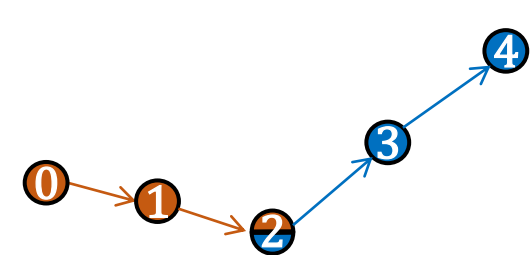
- Shared hits belong to same WCC → **two particles get merged**

4

ETX4VELO

C

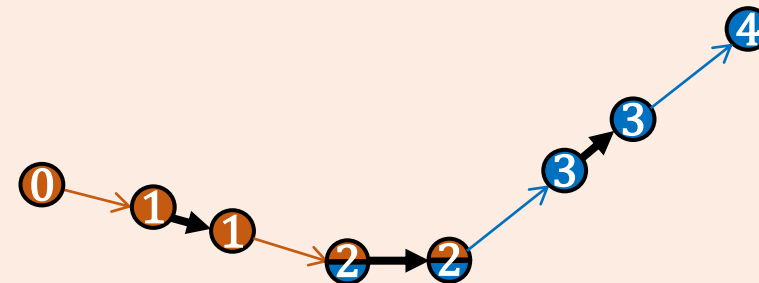
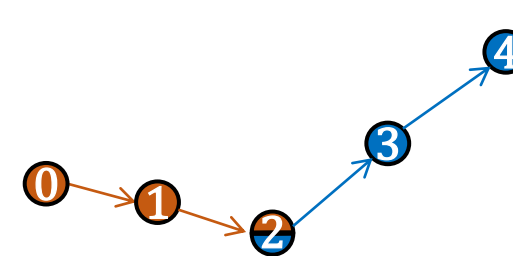
Issue with shared hits



22

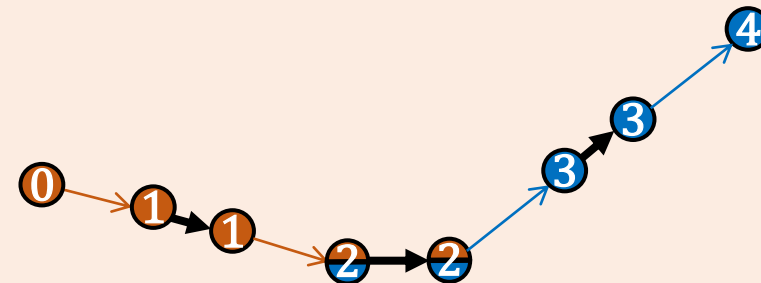
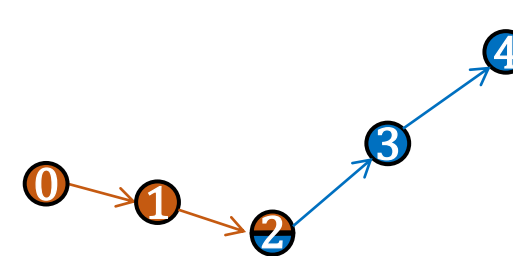
C Issue with shared hits**1. Go to line graph**

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

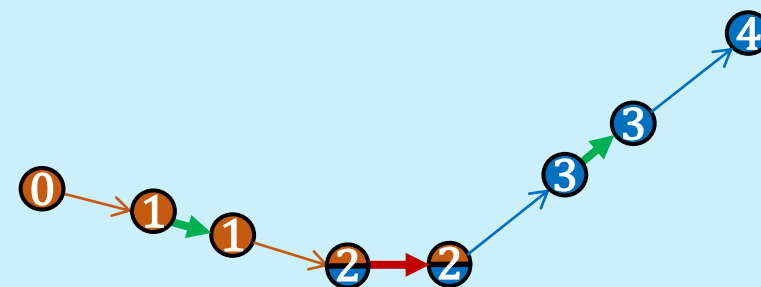


C Issue with shared hits**1. Go to line graph**

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

**2. Classify edge-edge connections using a GNN**

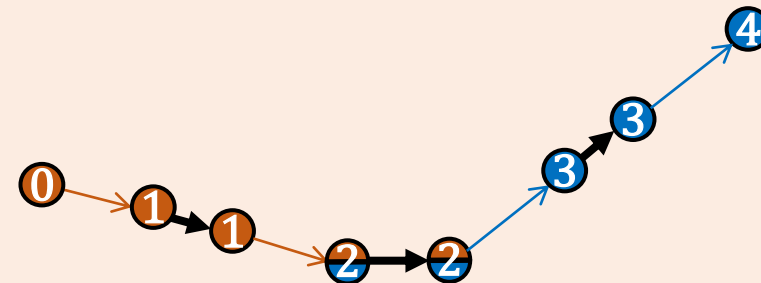
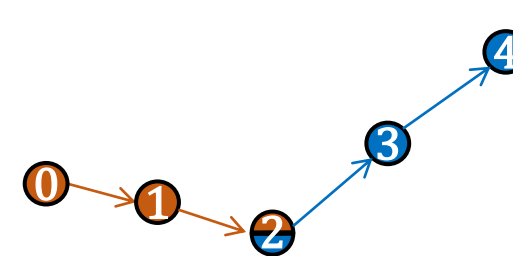
Use the same GNN trained with a double objective.



C Issue with shared hits

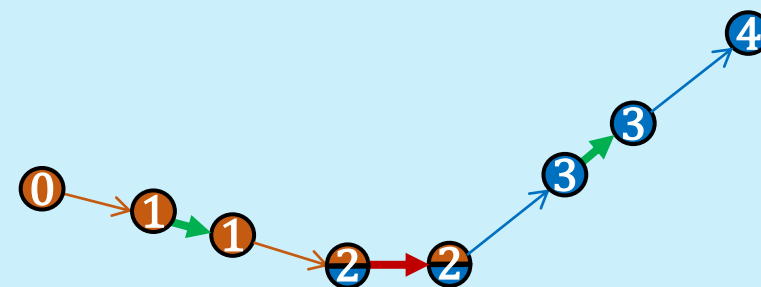
1. Go to line graph

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



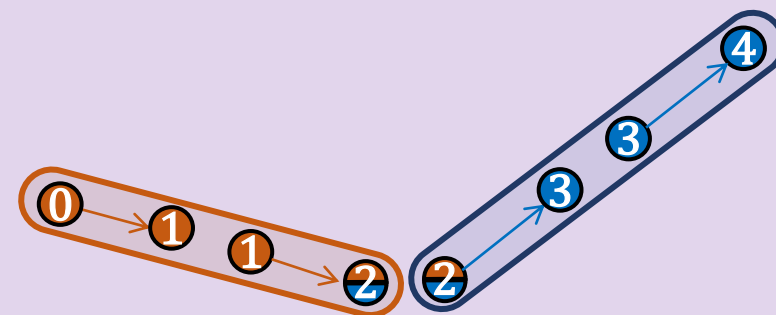
2. Classify edge-edge connections using a GNN

Use the same GNN trained with a double objective.



3. Apply WCC

More complex algorithm required to account for all cases.



d Other improvements

Metric	Allen	Exa.TrkX	1. Redefine edges	2. Increase dataset (400k)	3. Go to line graph	4. kNN layer by layer
No-electron efficiency	98.35%	93.72%	97.10%	97.44%	97.49%	98.53%
Electron efficiency	96.55%	64.14%	92.40%	92.12%	99.27%	98.43%
Fake rate	0.88%	7.50%	1.51%	0.50%	0.73%	0.42%

1000 simulated minimum-bias events without spillover

- Physics performance: **OK**
- Throughput: **?**

e Throughput

- **Goal:** implement GNN-based pipeline on C++/CUDA inside Allen.
- **Pipeline:** *line graph not implemented*



```
graph LR; A[Graph Building] --> B[GNN to filter edges]; B --> C[Build tracks];
```

Graph Building

GNN to filter edges

Build tracks

- **Detailed view:**



```
graph LR; A[Embedding Neural Network] --> B[kNN]; B --> C[GNN edge classifier]; C --> D[Filter edges]; D --> E[Weakly connected component algorithm];
```

Embedding Neural Network

kNN

GNN edge classifier

Filter edges

Weakly connected component
algorithm

e Throughput

- **Goal:** implement GNN-based pipeline on C++/CUDA inside Allen.
- **Pipeline:** *line graph not implemented*



Graph Building

GNN to filter edges

Build tracks

- **Detailed view:**



Embedding Neural Network

kNN

GNN edge classifier

Filter edges

Weakly connected component
algorithm

- There are **neural network model to export** and **algorithms to implement in C++/CUDA**.
- Deep learning model inference in C++:
 1. Export model in [ONNX open-source format](#).
 2. Inference in C++ using either [ONNXRuntime](#) or [NVIDIA TensorRT](#) libraries.

Metric	Allen	ETX4VELO
No-electron efficiency	98.35%	99.37%
Electron efficiency	96.55%	97.87%
Fake rate	2.19%	0.58%
Throughput	595 kHz	0.026 kHz

*1000 simulated minimum-bias events
with spillover*

e Throughput**Optimisations**

- Network size reduction
- Architectural changes
- Input coordinates changes
- etc.

Metric	Allen	ETX4VELO	ETX4VELO <i>optimised</i>
No-electron efficiency	98.35%	99.37%	99.35%
Electron efficiency	96.55%	97.87%	97.43%
Fake rate	2.19%	0.58%	1.01%
Throughput	595 kHz	0.026 kHz	0.996 kHz

*1000 simulated minimum-bias events
with spillover*

×38 in throughput

$\mathcal{O}(100 - 1000)$ slower!

e **Throughput****Optimisations**

- Network size reduction
- Architectural changes
- Input coordinates changes
- etc.

Metric	Allen	ETX4VELO	ETX4VELO <i>optimised</i>
No-electron efficiency	98.35%	99.37%	99.35%
Electron efficiency	96.55%	97.87%	97.43%
Fake rate	2.19%	0.58%	1.01%
Throughput	595 kHz	0.026 kHz	0.996 kHz

1000 simulated minimum-bias events
with spillover

×38 in throughput

$\mathcal{O}(100 - 1000)$ slower!

GNN is the bottleneck

- **scatter_add**: rely on AtomicAdd
- **# rough edges**: ×9 w.r.t. hits
- **# operations**

Step	Throughput
0. Decode velo hits	1100 kHz
1. Embedding Network	330 kHz
2. FRNN	110 kHz
3. GNN Edge Classifier	1.00 kHz
4. WCC	0.996 kHz
Allen	595 kHz

	ETX4VELO	Exa.TrkX as a Service On NVIDIA Triton server 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

- More in my thesis: [10.70675/ed529256zb1ddz41a7zb8a6z93eb17f16bb8](https://arxiv.org/abs/10.70675/ed529256zb1ddz41a7zb8a6z93eb17f16bb8)

	ETX4VELO	Exa.TrkX as a Service On NVIDIA Triton server 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

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

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

- More in my thesis: [10.70675/ed529256zb1ddz41a7zb8a6z93eb17f16bb8](https://arxiv.org/abs/10.70675/ed529256zb1ddz41a7zb8a6z93eb17f16bb8)

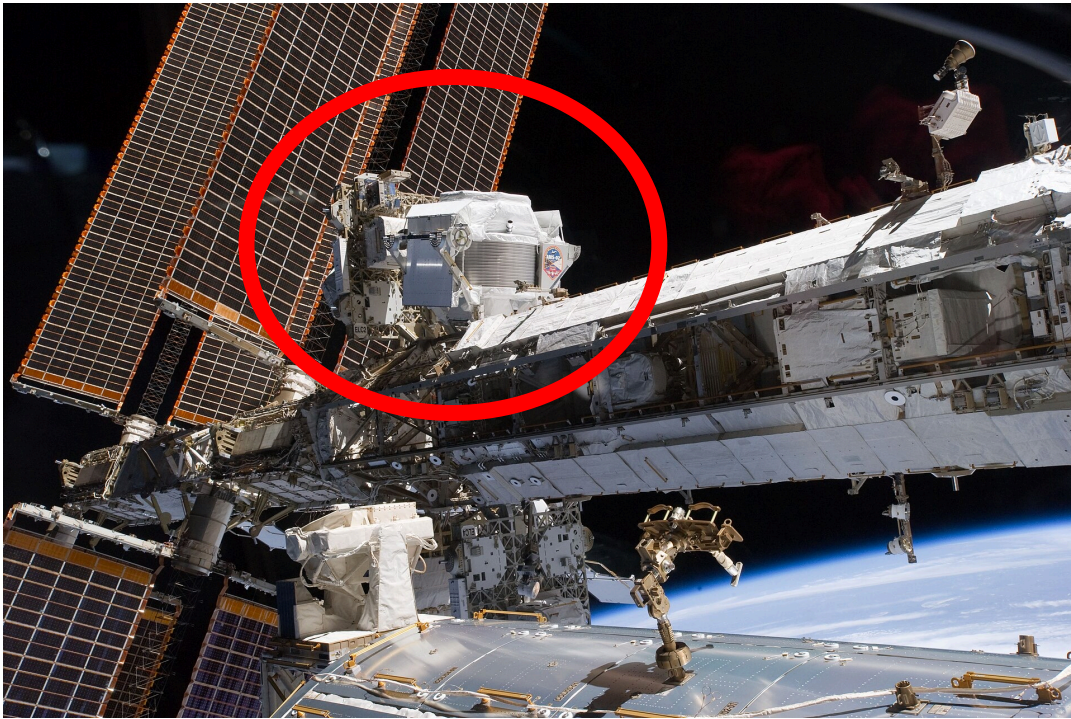
Model Nuclear Cross Sections for Cosmic Ray Propagation

Anthony Correia



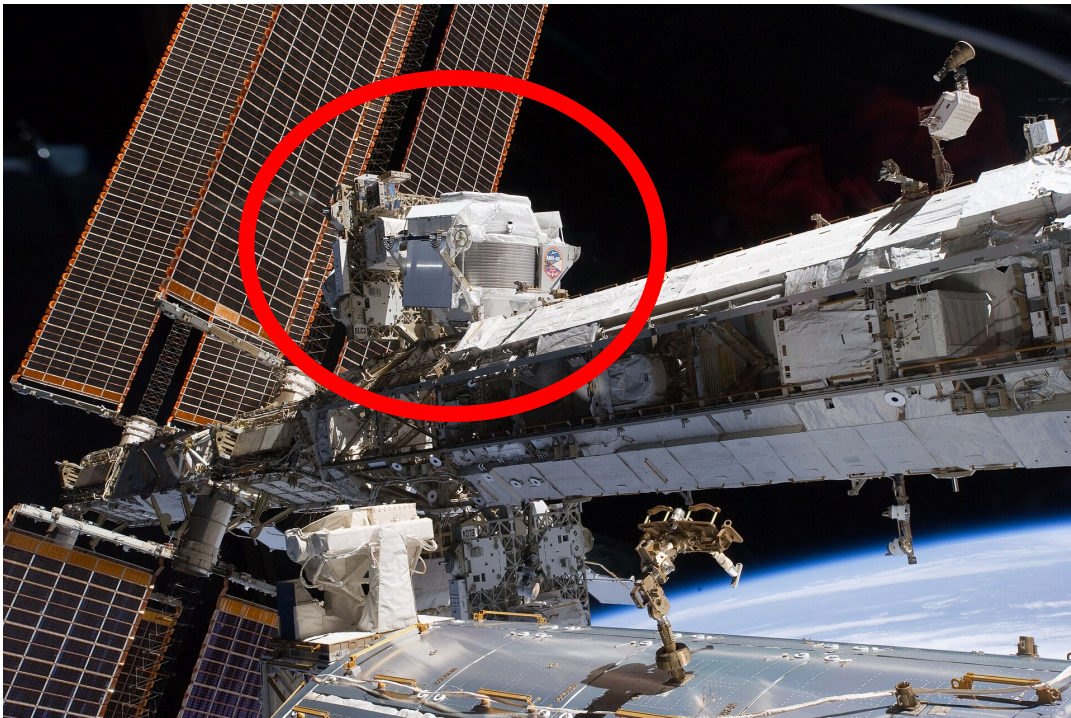
a Tension in $2H$ flux

AMS-02 (*Alpha Magnetic Spectrometer*)
Particle detector installed at the ISS

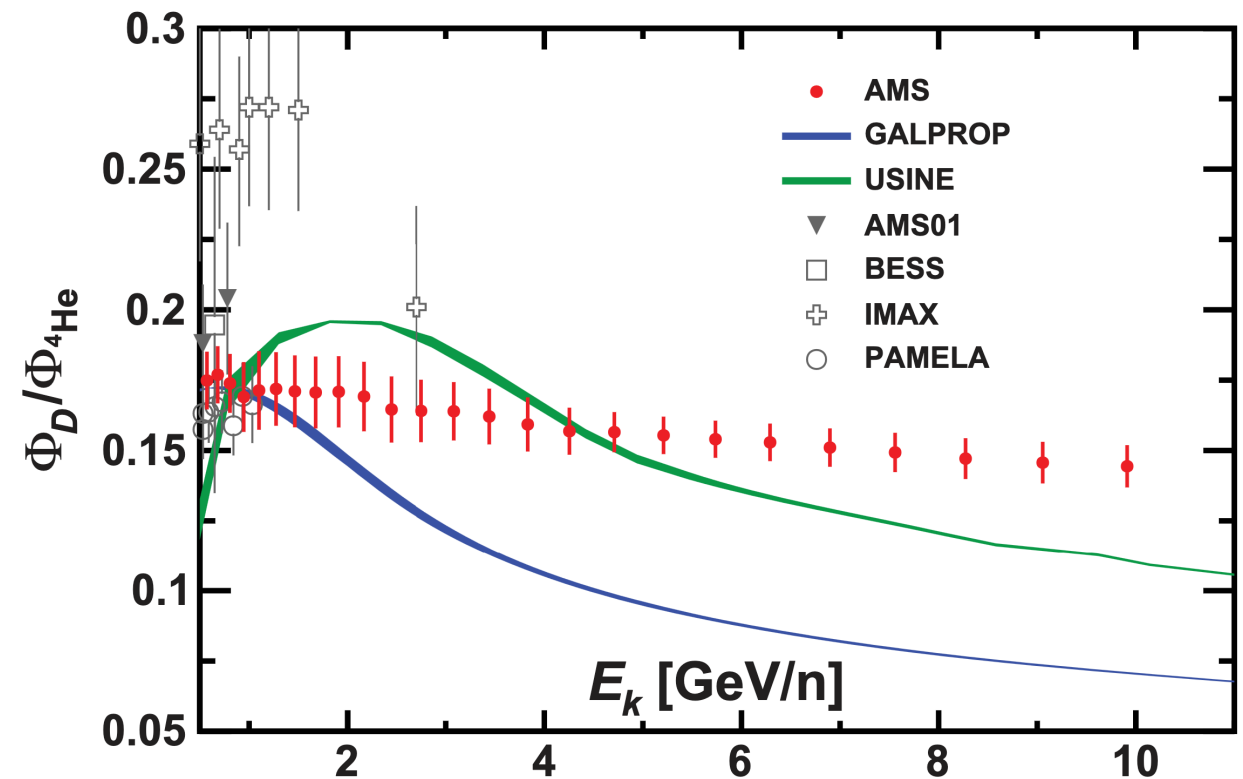


a Tension in 2H flux

AMS-02 (*Alpha Magnetic Spectrometer*)
Particle detector installed at the ISS

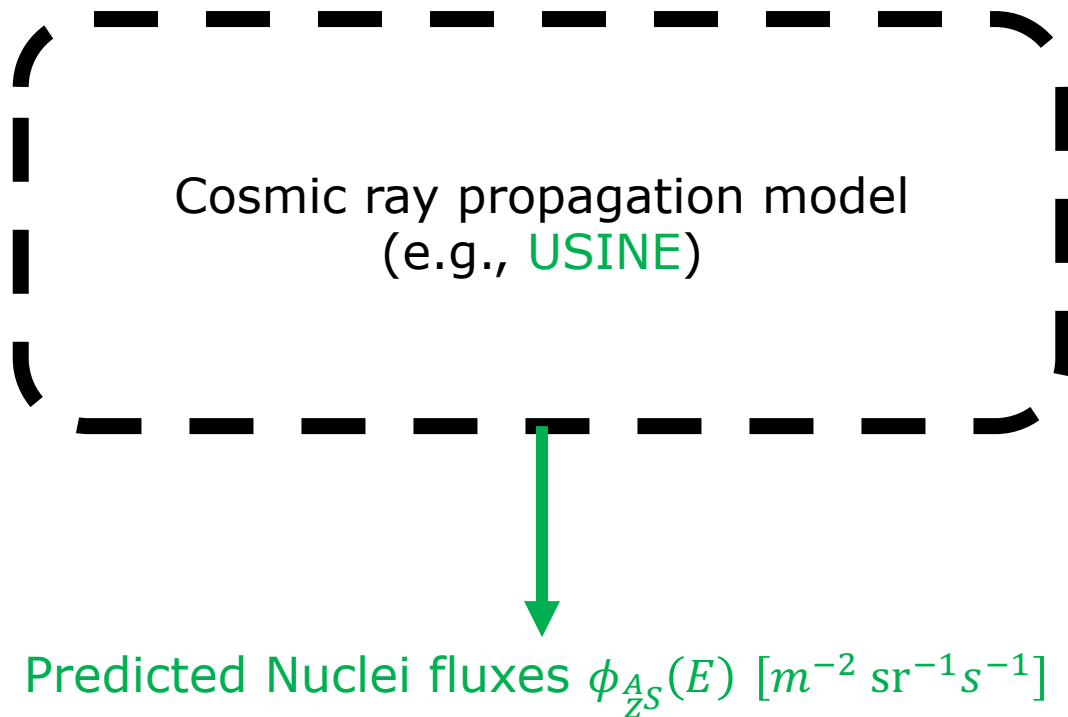


AMS-02 measurements

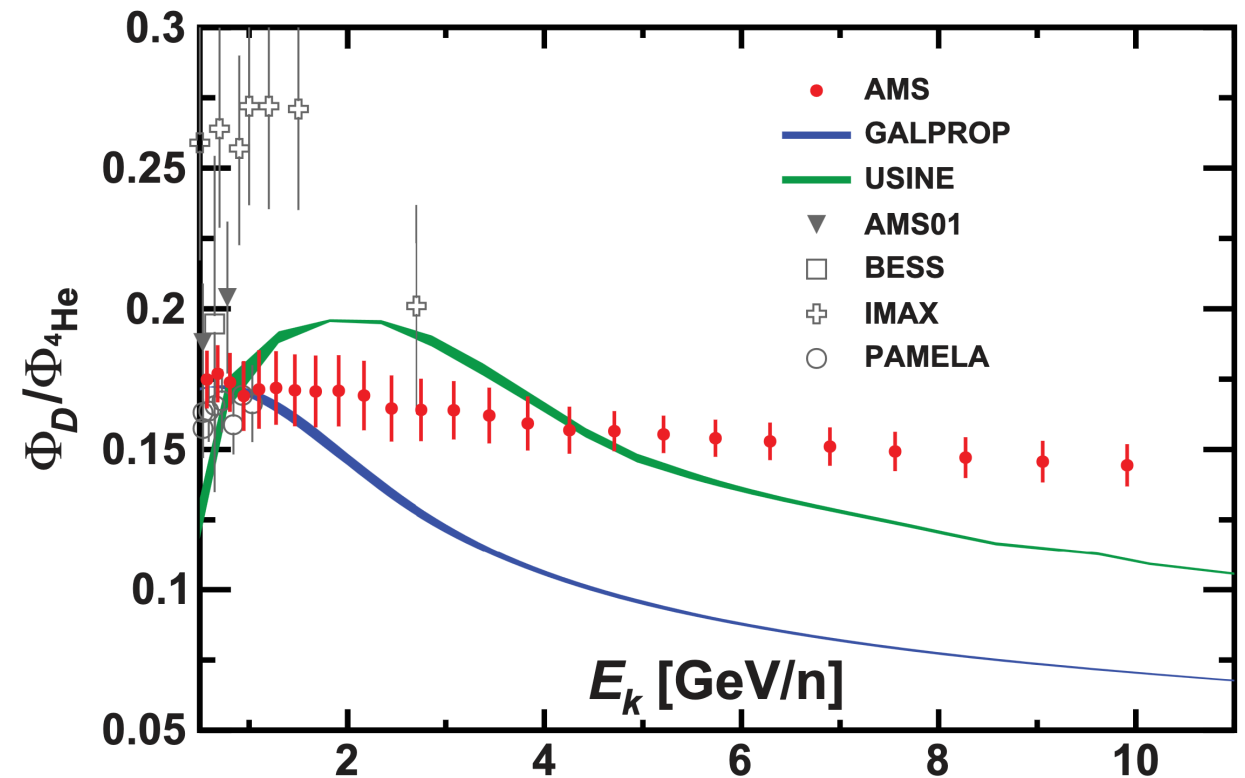


[10.1103/PhysRevLett.132.261001](https://arxiv.org/abs/10.1103/PhysRevLett.132.261001)

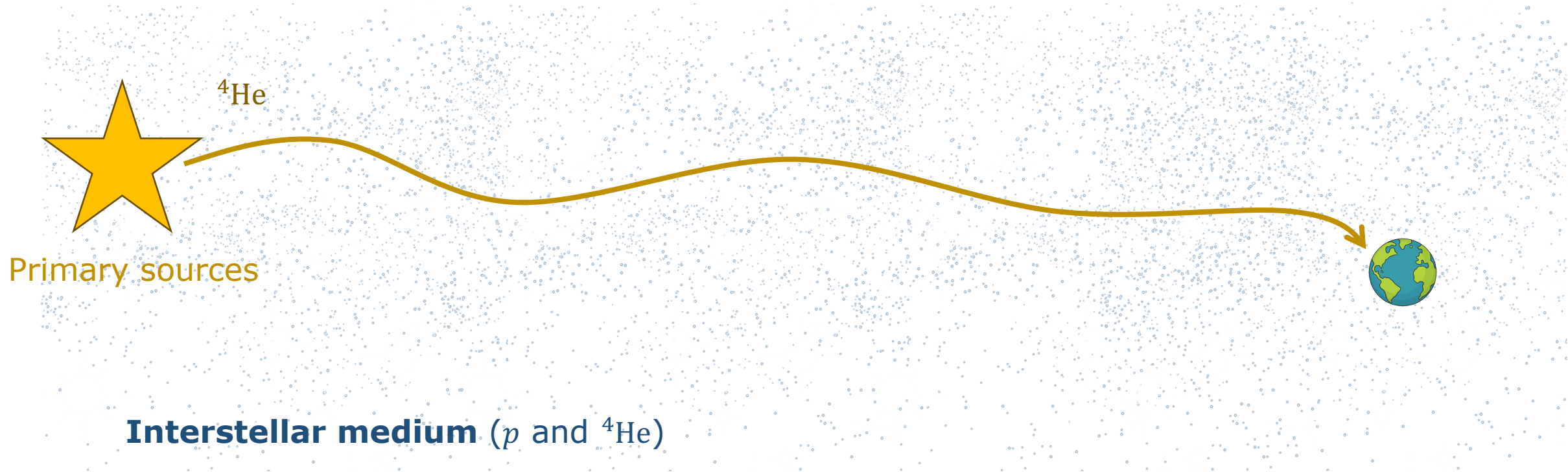
⇒ Tension in 2H between **observations** and **predictions**

a Tension in 2H flux

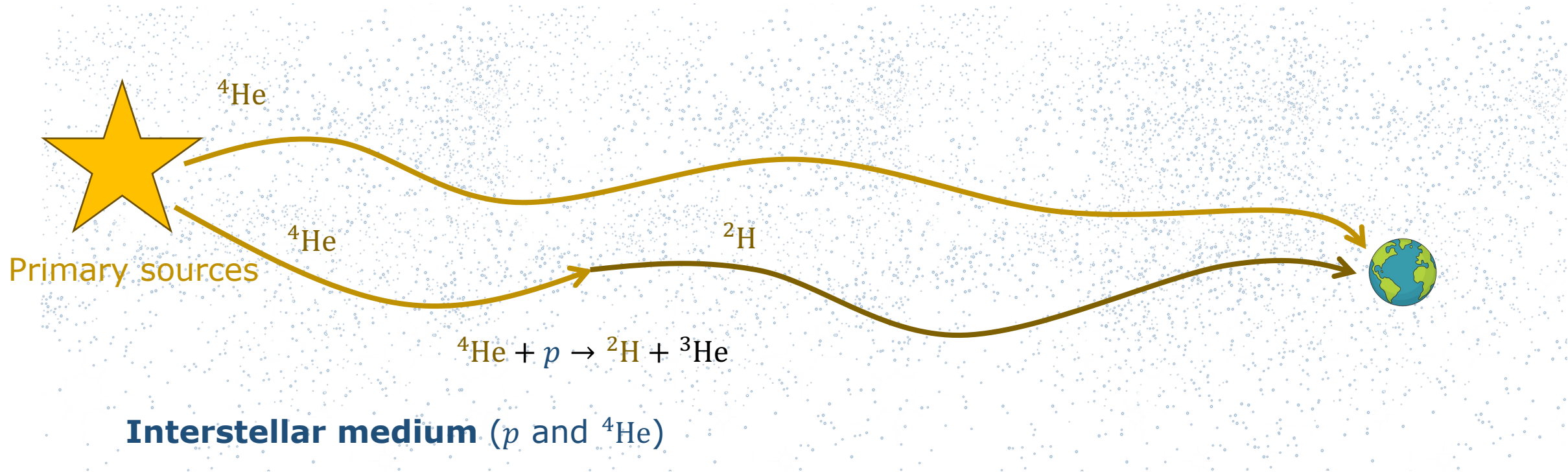
AMS-02 measurements


[10.1103/PhysRevLett.132.261001](https://arxiv.org/abs/10.1103/PhysRevLett.132.261001)

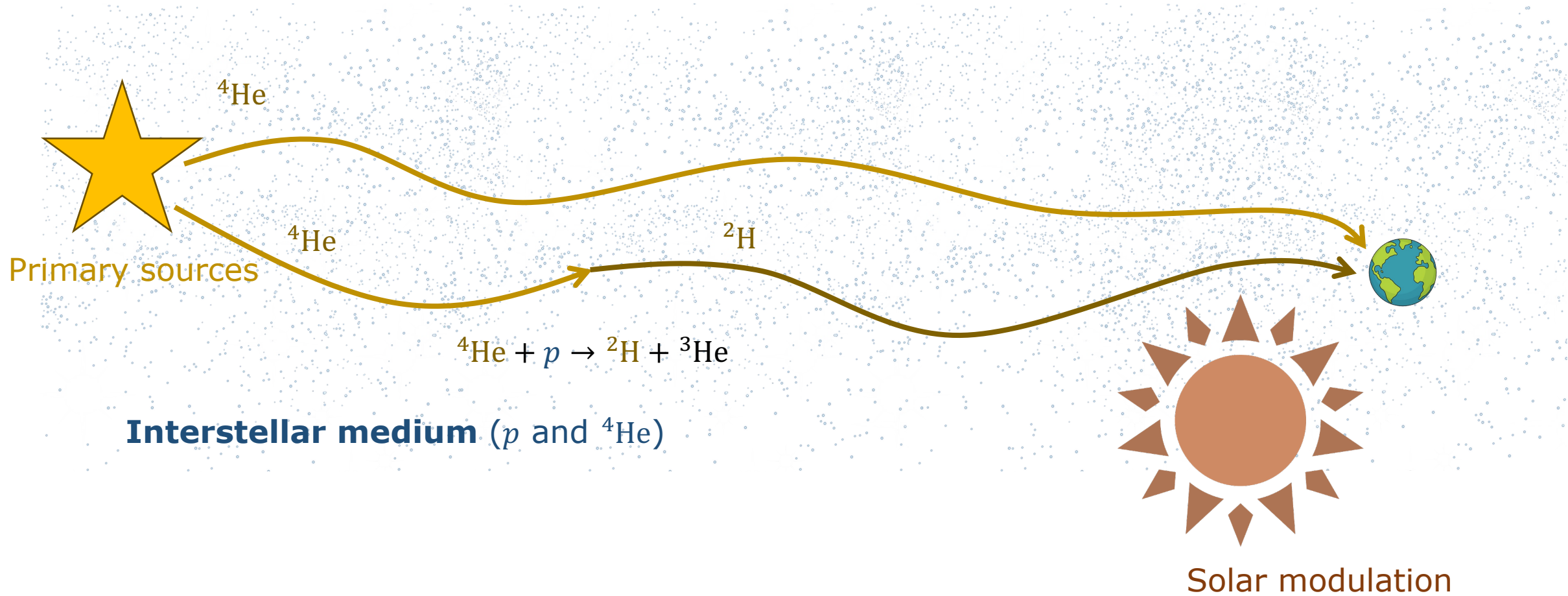
⇒ Tension in 2H between **observations** and **predictions**

b Cosmic Ray Propagation

b Cosmic Ray Propagation



b Cosmic Ray Propagation

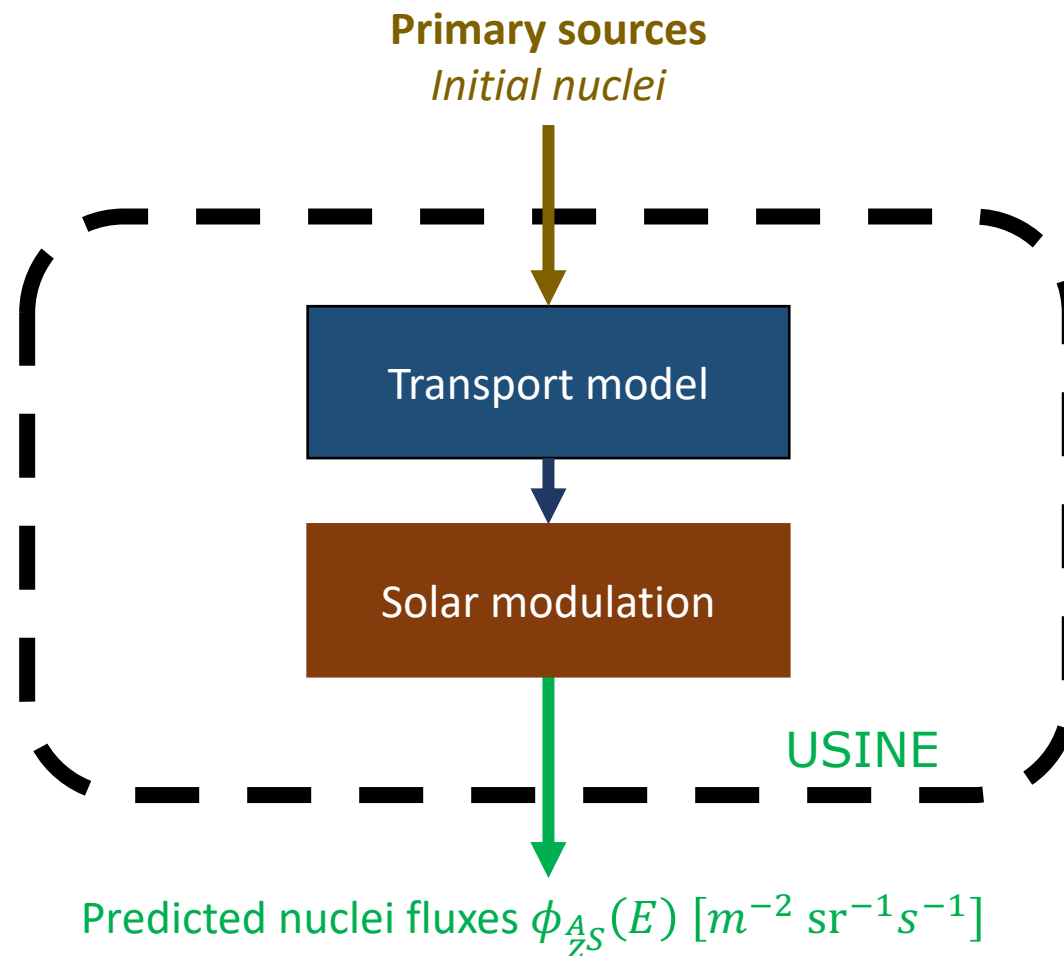


6

Cross Section Modeling for Cosmic Ray Propagation

b

Cosmic Ray Propagation

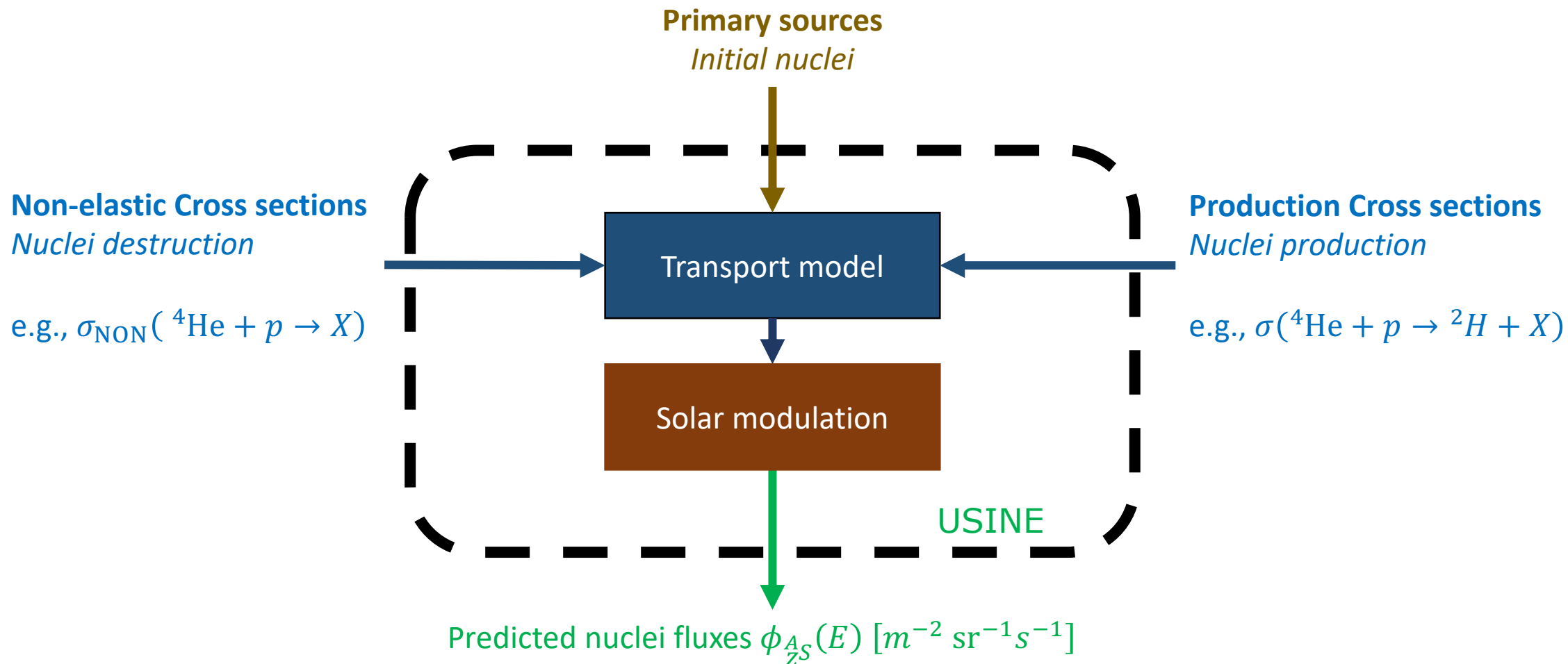


6

Cross Section Modeling for Cosmic Ray Propagation

b

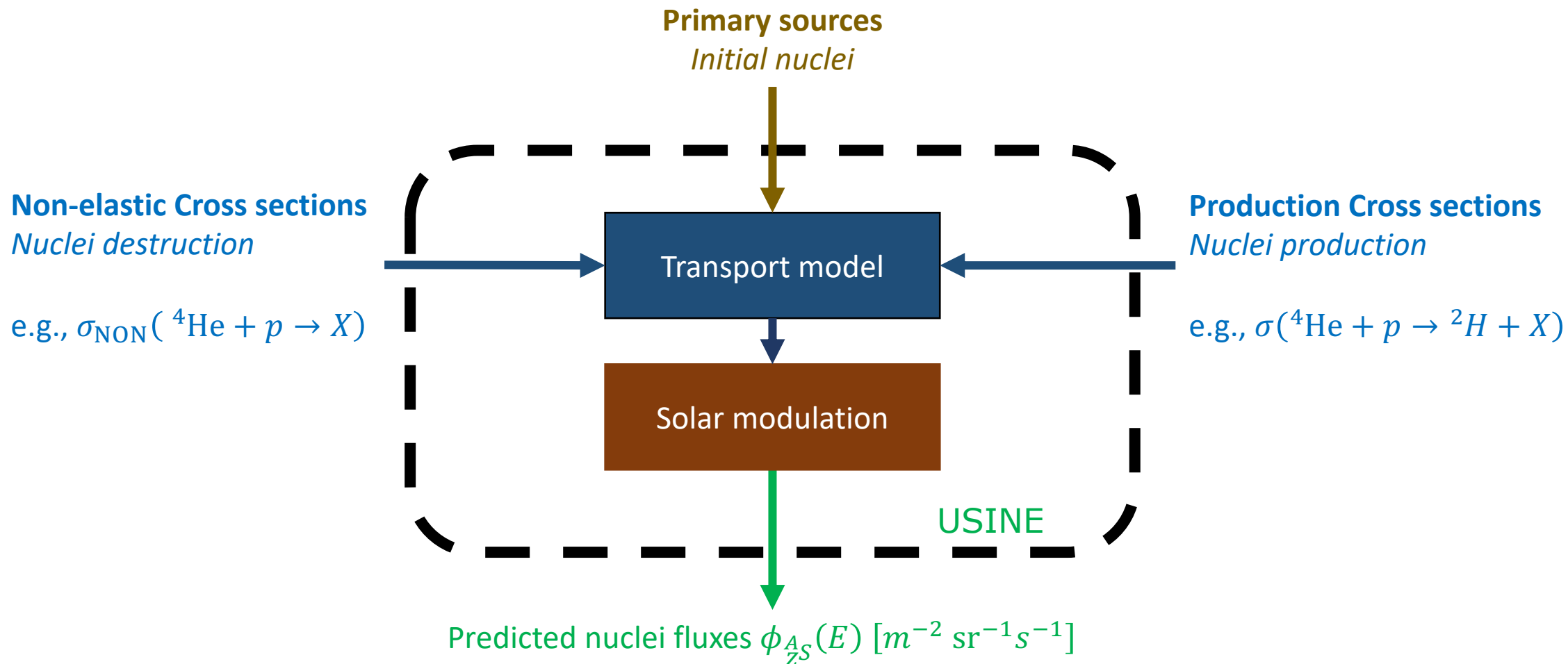
Cosmic Ray Propagation



6

Cross Section Modeling for Cosmic Ray Propagation

b Cosmic Ray Propagation



Tension could be resolved by:

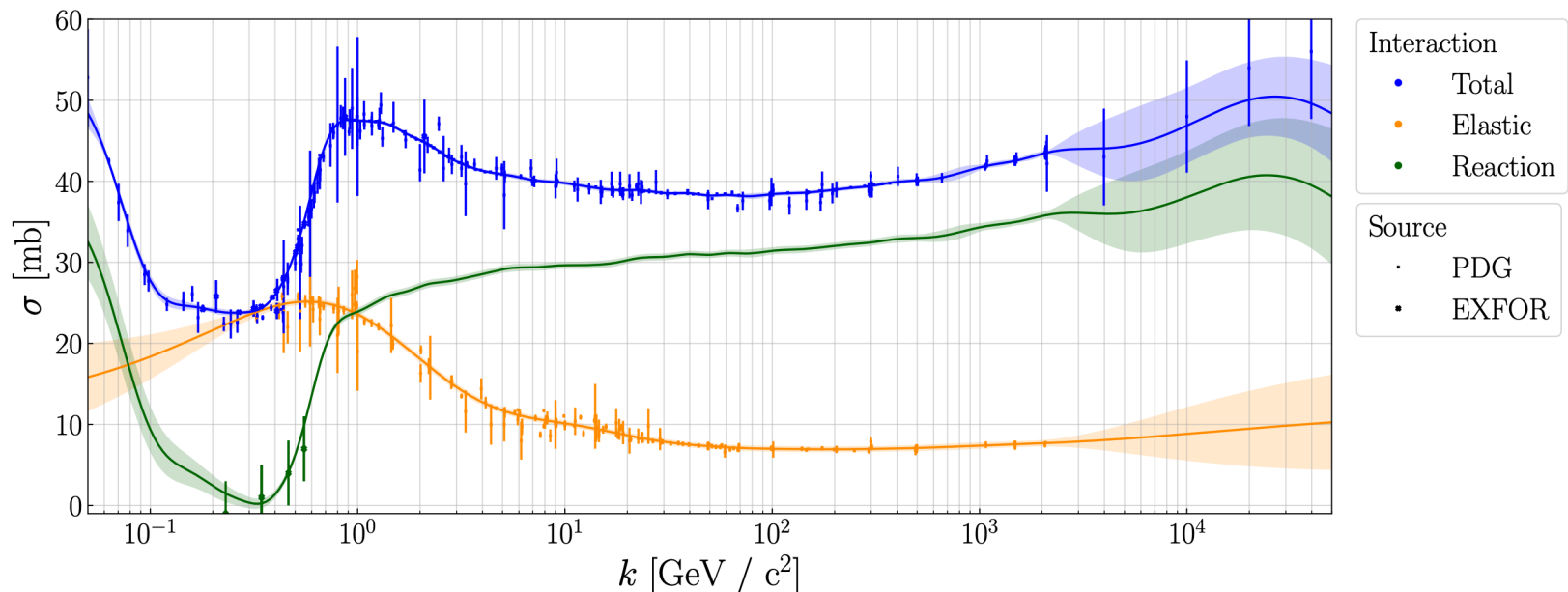
- ^2H primary source
- **Up-to-date cross sections**

C Example

- **Gather training data**
 - EXFOR
 - Particle Data Group (PDG)
 - Manual lookup
 - HEPData (*new!*)

Exemple using **Gaussian Processes** to get $\sigma_{NON}(pp \rightarrow X)$

1. Fit $\sigma_{TOT}(pp \rightarrow X)$
2. Fit $\sigma_{EL}(pp \rightarrow pp)$
3. Compute $\sigma_{NON} = \sigma_{TOT} - \sigma_{EL}$



Goal: model all **production** and **reaction cross sections**, **including their uncertainties**

- Gather as much data as possible (data is scarce)
- Benchmark a suite of ML methods (Gaussian processes, boosted decision trees, etc.)

⇒ Update cosmic ray propagation models

Goal: model all **production** and **reaction cross sections**, **including their uncertainties**

- Gather as much data as possible (data is scarce)
- Benchmark a suite of ML methods (Gaussian processes, boosted decision trees, etc.)

⇒ Update cosmic ray propagation models

Postdoc

- from **1st July 2026 to 31th December, 2027**
- funded by **MIAM**
- supervised by **David Maurin**

That's all folks!

Annexe

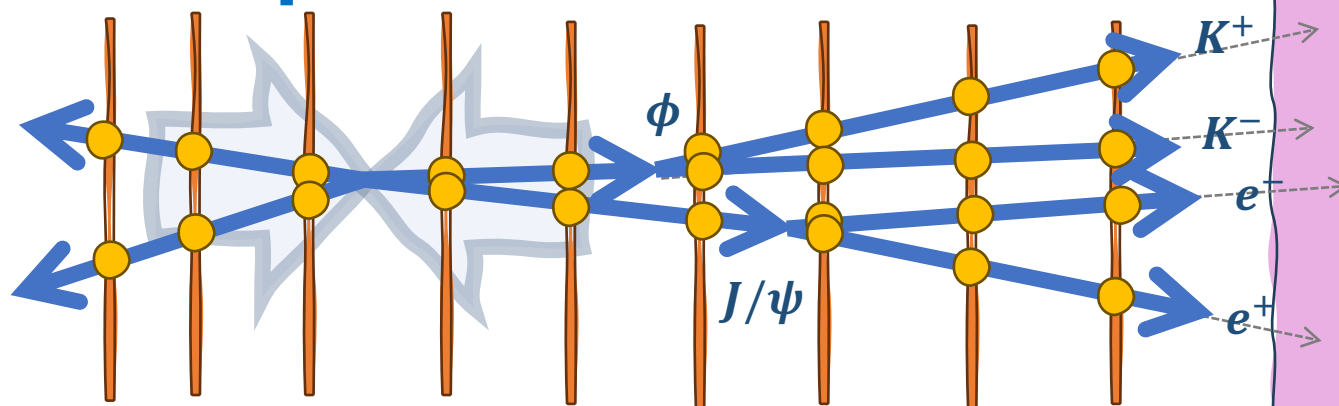
1 Track Finding at LHCb

C Track Finding

Challenges

- Reconstruct **pp collision points** (*primary vertices*)
- Reconstruct **decay points** (*secondary vertices*)
- **Seeds** for other algorithms

Proton-proton collision



Other LHCb
subdetectors

Tracking detector