Karlsruhe Institute of Technology

ETP-Bachelor-KA/2026-03

One-Shot Neural Inference for Latent Space Clustering in GNN-Based Track Finding at Belle II

Daniel Micha Grossmann

Bachelor's Thesis

at the Department of Physics
Institute of Experimental Particle Physics (ETP)

Reviewer:	Prof. Dr. Torben Ferber
Second Reviewer:	Prof. Dr. Achim Streit
Advisor:	Dr. Lea Reuter

1 December 2025 – 1 April 2026

Karlsruher Institut für Technologie
Fakultät für Physik
D-76128 Karlsruhe

I declare that I have developed and written the enclosed thesis completely by myself, and have not used sources or means without declaration in the text.

Karlsruhe, 1 April 2026

. . . . . .

(Daniel Micha Grossmann)

Abstract

This thesis proposes a one-shot neural clustering algorithm for object condensation pipelines. The algorithm groups an unordered input point cloud of variable count with latent space representations of the input hits into a variable number of clusters and differentiates between signal and noise. Implementation and evaluation is done in the graph neural network based track-finding pipeline for the Belle II central drift chamber. In a first step, the clustering pipeline and machine learning architecture are presented, combining key solutions from different computer vision pipelines. Next, the required hyperparameters are fitted using simulated detector samples, showing a good generalization across vastly differing event topologies. Then, the clustering is evaluated on a diverse set of challenging events, leveraging a realistic detector simulation combined with high levels of detector noise measured in actual collisions. Additional validation is performed directly on measured collision data with high levels of detector noise, showing real-world applicability and robustness. The evaluation shows a significant improvement in track charge efficiency and clone rate, while retaining a similar fake rate. When benchmarked against state of the art classical clustering algorithms, the one-shot neural inference provides a significant speedup without sacrificing clustering quality.

Zusammenfassung

Diese Arbeit stellt einen One-Shot Clustering-Algorithmus auf Basis neuronaler Netze für Object-Condensation Pipelines vor. Der Algorithmus teilt eine ungeordnete Punktwolke mit variabler Größe in eine variable Anzahl an Cluster auf und unterscheidet zwischen Signal und Hintergrund. Die Implementierung und Auswertung erfolgt in der auf einem Graph-Neuronalen Netzwerk basierenden Spurfindungs Pipeline für die zentrale Driftkammer im Belle II Experiment. Zuerst wird die Clustering Pipeline und die verwendete Machine Learning Architektur präsentiert, welche Lösungen aus verschiedenen Computer-Vision Pipelines verbindet. Danach werden die nötigen Hyperparameter auf simulierten Detektor-Ereignissen angepasst und eine gute Generalisierungsfähigkeit für stark unterschiedliche Topologien festgestellt. Dann wird das Clustering mithilfe einer realistischen Detektor-Simulation und hohem Hintergrundanteil von echten Kollisionen auf einer diversen Menge an herausfordernden Ereignissen getestet. Zusätzliche Validierungsexperimente werden direkt auf gemessenen Kollisionsdaten mit hohem Hintergrundanteil durchgeführt und die Robustheit und Anwendbarkeit in realen Umständen gezeigt. Die Auswertung zeigt eine signifikante Verbesserung in der Effizienz, sowie in der Anzahl der Duplikate, wobei eine vergleichbare Rate an unechten Spuren gehalten wird. Im direkten Vergleich mit einem klassischen State-of-the-Art Clustering-Algorithmus wird sichtbar, dass die One-Shot Neuronale Inferenz eine signifikante Beschleunigung bewirkt, ohne die Qualität zu beeinträchtigen.

Disclaimer

Data analyses in high-energy physics such as the measurement presented in this bachelor thesis are a collaborative effort. The SuperKEKB particle accelerator which provides the particle beams essential for all studies at Belle II was built and is operated and maintained by the SuperKEKB accelerator group. The Belle II detector was built and is maintained and operated by the Belle II collaboration. The Belle II collaboration also creates the simulated and recorded data sets and maintains the computing infrastructure necessary to process them. The software environment necessary for studies with Belle II data plays an important role and is maintained by the collaboration.
The underlying pipeline (as described in Section 2.2) is the work of Lea Reuter [1, 2], utilizing her previous results and building on the existing code base for model training and validation. I have performed all studies in this thesis except for the $D^{\star+} \to D^0(\to K^0_s(\to \pi^+\pi^-)\pi^+\pi^-)\pi^+_S$ and $e^+e^- \to \mu^+\mu^-$ candidate selections on measured data, that are implemented as validation modes in VIBE[1] and the following validation, implemented in [2].

This thesis incorporates the use of Artificial Intelligence (AI) tools to help with grammatical and stylistic improvement of text, and program code creation.
ChatGPT[2] is used for spell and grammar checks, as well as for paraphrasing individual, selected sentences to improve clarity and precision in academic writing. I have approved all suggested changes.
ChatGPT[2] and GitHub Copilot[3] are used to aid the development of program code, in particular code restructuring and optimisation that do not constitute the core scientific work of this thesis. I have approved and tested all suggestions to provide robust and reliable results.

[1] VIBE: Validation Interface for the Belle II Experiment, `https://vibe.belle2.org/` (accessed: March 31, 2026)
[2] ChatGPT: Large Language Model, `https://chat.openai.com/` (accessed: March 31, 2026).
[3] GitHub Copilot: AI-powered code completion tool, `https://github.com/features/copilot` (accessed: March 31, 2026).

Contents

1. Introduction

Many high energy physics (HEP) experiments rely on reconstructing single particles from characteristic sets of hits in a detector. In the Belle II experiment, this can be tracks formed by hits from charged particles in the Central Drift Chamber (CDC), clusters formed by cells in the Electromagnetic Calorimeter (ECL) or cherenkov rings made up of pixels in the Aerogel Ring Imaging Cherenkov Counter (ARICH). In each of these sub-detectors, the hits of actual particles are sparsely distributed across the full detector space. The track finding in the CDC is chosen as a representative for these tasks in the evaluation shown in this thesis.
A key challenge for track finding in the CDC is the high beam background, which increases the number of hits per event and thus necessitates scalable and robust reconstruction algorithms. As the SuperKEKB accelerator is designed to further increase its instantaneous luminosity considerably to search for ever more rare processes, steadily increasing background levels are expected [3]. These challenges can be met by the development of machine learning based approaches, which can learn to separate signal from background hits.
A common framework for machine learning based detector-hit instance segmentation is the object condensation approach [4]. This is a loss framework for training models which map detector hits into a latent space, where a clustering step combines the hits into physical objects. Current object-condensation implementations either rely on a fast, but not very robust *spherical clustering* or they employ iterative clustering algorithms. As the Belle II data acquisition rate lies significantly below the collision rate, a trigger system is required to scan for interesting event signatures [3]. To use object condensation pipelines within the High Level Trigger (HLT), non-iterative algorithms are important, as the latency should be minimized. But the computational cost is an important factor for offline reconstruction algorithms as well.
The *CATFinder* [1], a Graph Neural Network (GNN)-based track finding algorithm for the CDC is based on object condensation. Even though this algorithm outperforms the currently used *Baseline* tracking [5], it still struggles on challenging event topologies. This is in part due to the limited performance of the employed *spherical clustering*.

In this thesis, a new clustering algorithm is proposed, which leverages a one-shot neural inference to achieve a clustering quality comparable to state of the art iterative algorithms with a fraction of the computational cost.
In Chapter 2, an overview of the experimental context and the current implementation of the *CATFinder* algorithm is provided. Chapter 3 provides the data sets and evaluation setup used for development and evaluation of the clustering performance. The new clustering algorithm is proposed in Chapter 4, with the training and implementation setup described in Chapter 5. Chapter 6 provides performance comparisons to the current *CATFinder*

implementation and the *Baseline* algorithm across several different event topologies. Further experiments and ablation studies indicating future research directions are performed in Chapter 7. A validation of assumptions made during the development, as well as a performance study on real collision data is performed in Chapter 8. The thesis is concluded with Chapter 9, reporting future research directions and a summary of the key findings.

2. Foundations

The clustering algorithm algorithm presented in this thesis is implemented for and evaluated on the task of finding tracks in the Belle II experiment. This chapter provides an overview of the experimental context as well as the GNN-based track finding algorithm that is to be improved through integration of the *neural clustering*.
First, this chapter describes the Belle II experiment with a focus on the CDC which is its main tracking detector and the currently used *Baseline* tracking algorithm. Then, the *CDC AI Tracking* (*CATFinder*) pipeline proposed in [1] will be shown, highlighting the clustering algorithm used in the post-processing of the GNN output. Finally, existing clustering approaches and machine learning architectures related to the proposed *neural clustering* are introduced.

2.1. Belle II

Detector Design The Belle II detector [6] is located at the SuperKEKB accelerator [7]. SuperKEKB produces collisions of 7 GeV electrons with 4 GeV positrons at the Interaction Point (IP) of the Belle II detector. This results in a center-of-mass collision energy at $\sqrt{s} = 10.58\,\text{GeV}$, which matches the $\Upsilon(4S)$ resonance, allowing for decays into B-meson pairs [3].
The Belle II detector consists of a tracking detector that is surrounded by particle identification detectors, an electromagnetic calorimeter, a 1.5 T solenoid and a K_L and μ detector. To minimize material budget, the tracking detector uses silicon detectors only in the innermost section to reconstruct the decay vertex position [3]. The main track finding and determination of the particle momentum is performed using the CDC.
An overview of the detector structure is provided in Figure 2.1.

Central Drift Chamber The CDC is the main tracking detector of the Belle II experiment, providing momentum measurements for charged particles as well as particle identification for particles that do not reach the outer detector systems due to low transverse momentum. The CDC is build as a cylindrical chamber with $14,336$ sensor wires and $> 40,000$ field wires. Charged particles flying through the helium-ethane gas-mixture in the CDC ionize the gas, resulting in electrons drifting towards the sensor wires, triggering a townsend avalanche due to the high electric field. This charge deposit is measured as the signal for each hit. The measurement is digitized, resulting in timing and amplitude signals [3].
The timing is measured using a *time-to-digital converter*, providing a digital signal *tdc* representing the time between collision and measurement of the hit. This time is dominated by the drift of the electrons from the ionization point to the signal wire. It is thus correlated to the distance between the particle trajectory and the signal wire. The charge deposit

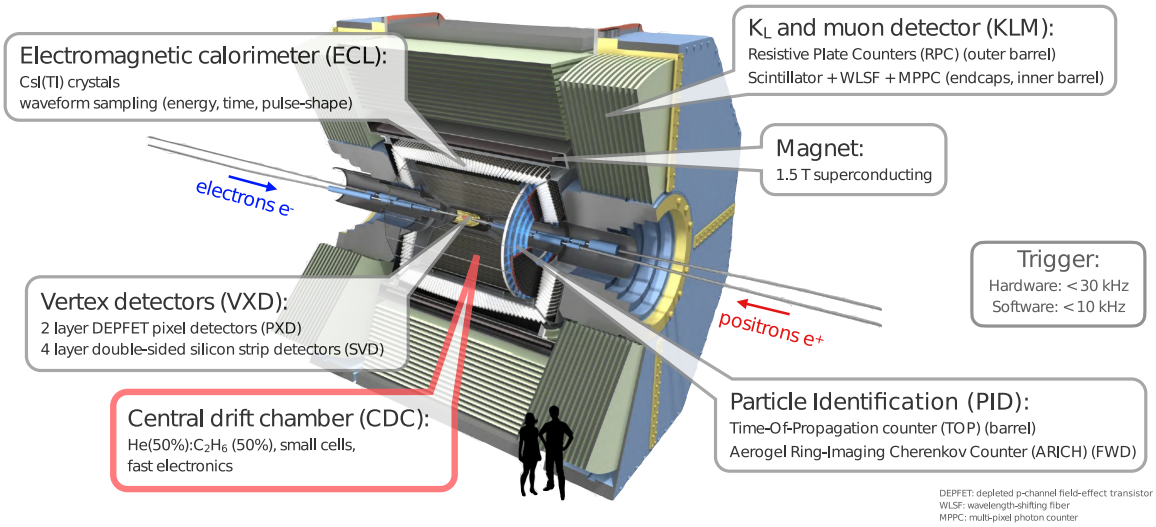
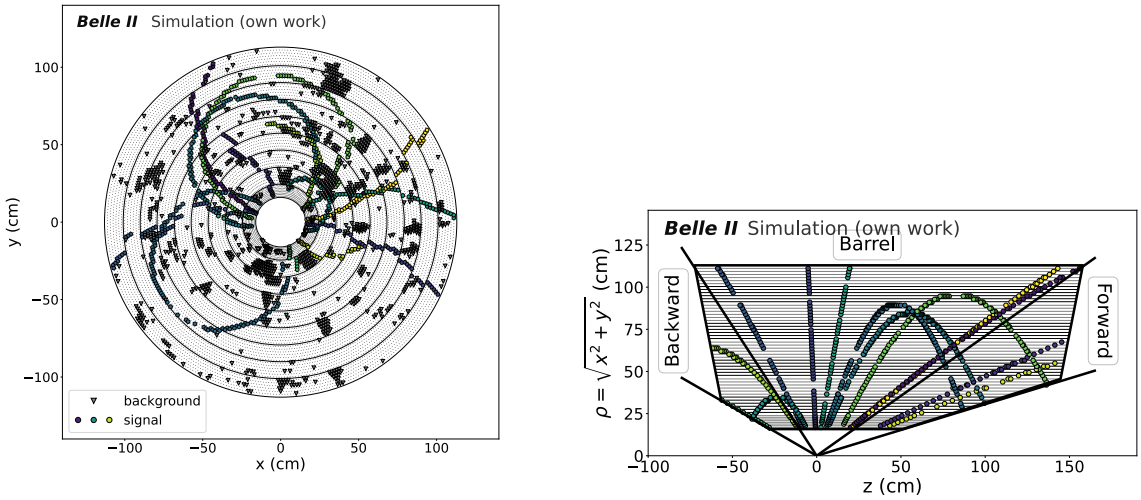


Figure 2.1.: The sub-detectors of the Belle II detector. The CDC is highlighted in red. Taken from [8]

is digitized as the integrated waveform amplitude using an analog-to-digital converter, resulting in a value *adc* correlated to the energy deposited by the particle [3].
As the wires are only read out on one end, no direct z coordinate (along the beam axis) is provided per hit. Instead the wires are arranged in 56 layers, grouped into 9 superlayers with the wires of every second superlayer not completely aligned to the z-axis. This results in shifts between segments of a track when the track crosses the layer between two superlayers at $z \neq 0$, which can be used to reconstruct z-information. The CDC wraps around the full azimuthal angle and covers a polar acceptance of $\theta \in [17°, 150°]$. Tracks with $\theta \in [35.4°, 123°]$ traverse the full CDC and exit in the *barrel* region, while particles with lower θ exit through the *forward endcap* and particles with higher θ exit through the *backward endcap* [3].
Figure 2.2 shows the hits in the CDC in a typical $\Upsilon(4S) \longrightarrow B\bar{B}$ event, where in the $x-y$-view the position of the middle of the wire is shown and the $\rho - z$-view shows the different regions and superlayers.

Baseline Track Finding Currently the Belle II track finding employs a two step algorithm based on a global Legendre transformation and a cellular automaton. The transformation maps prompt tracks into straight lines which are used to combine corresponding hits into tracks. The track finding algorithm is described in more detail in [5].
As described in [2], the *Baseline* fails on tracks which do not start at the interaction point, as these tracks are no longer mapped to a straight line by the transformation.
After track finding, a fitting algorithm implemented in GENFIT2 [9] is used to determine the track parameters. This track fitting requires an initial estimate of the track starting point, momentum and charge as well as an estimate of their covariance matrix [2]. The fit is then performed on a set of ordered hits starting at the decay vertex [2].



(a) $x - y$-view.

(b) $\rho - z$-view.

Figure 2.2.: CDC hits left by the particles in a simulated $\Upsilon(4S) \longrightarrow B\bar{B}$ event with *high data beam background*. Colored circular markers represent signal hits, gray triangular markers in the $x - y$-view show background hits.

2.2. GNN-based Track Finding

The following section describes the *CATFinder* track finding pipeline introduced in [1]. This algorithm leverages a GNN trained with object condensation (see [4]) to map detector hits into a latent space where a post-processing step finds clusters of hits corresponding to tracks in the real space. The *CATFinder*-GNN is based on GravNet [10] blocks, with graph-building based on a nearest-neighbor search in a learned representation space combined with an additional feature space. These blocks are connected via linear- and batchnorm-layers applied on all hits separately in parallel. Five regression heads are added to the final linear layer, learning clustering-specific and track-specific parameters described in more detail in Subsection 2.2.1, a schematic overview is provided in Figure 2.3. A detailed description of the GNN architecture is provided in [2].
In this section, first the object condensation loss and its application on the *CATFinder*-GNN are described. Then a detailed explanation of the *spherical clustering*, which is the clustering algorithm currently in use for *CATFinder* is provided. Finally, the track topologies which prove challenging to the current *CATFinder*-pipeline are reported.

2.2.1. Object Condensation

This section describes the object condensation loss used in the training of the *CATFinder*-GNN, closely following [2], which is a slight variation of the original object condensation pipeline proposed in [4].
In order to train a machine learning model for instance segmentation, a loss is needed

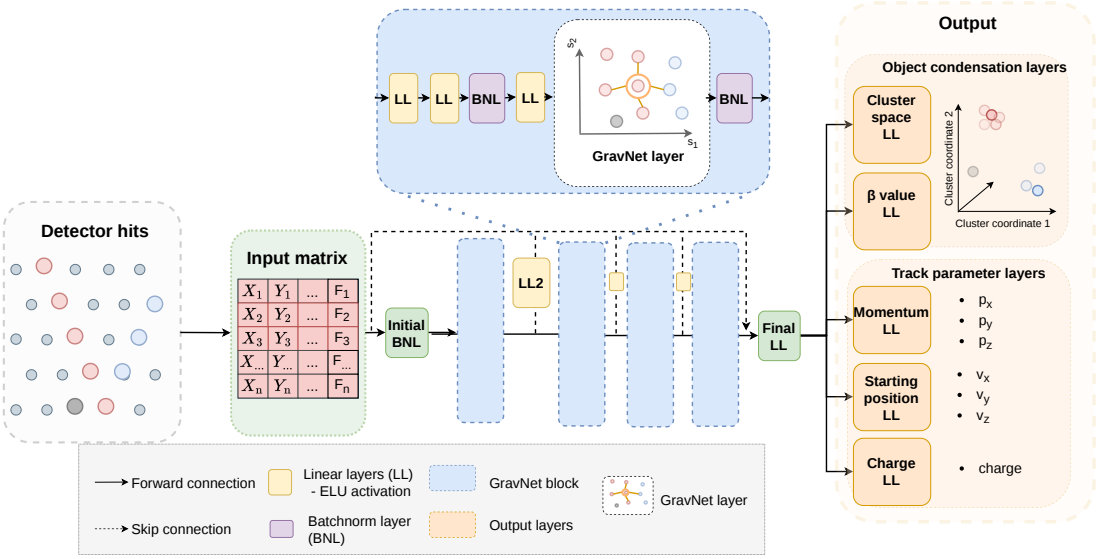


Figure 2.3.: A schematic overview of the *CATFinder*-GNN architecture. Taken from [1].

that can handle the varying number of instances to be found. In particle physics contexts, additionally to finding instances, often physical parameters of these instances are required as an input to subsequent fitting steps. The object condensation pipeline combines these two tasks into one loss framework.
A core concept in object condensation are *condensation points*. These are points which are good cluster center candidates. The idea behind object condensation is training the model to find the cluster centers, predicting the track parameters for them and predicting hits corresponding to the track of the *condensation point* into a close proximity in the latent space. For a given set of input hits $h_i \in H \, \forall i \in \{1, \ldots, N\}$, the model predicts latent space coordinates, a β value and track parameters $f_\theta(H)_i = (x_i, \beta_i, p_i)$. β_i predicts how likely the hit h_i is a good *condensation point* candidate for its cluster. The truth information for an event consists of a segmentation into clusters and background

$$\{1, \ldots, N\} = B \cup \bigcup_{k=1,\ldots,K} C_k.$$

For each cluster the *condensation point* is selected by requiring

$$\beta_{\alpha k} = \max_{i \in C_k} \beta_i.$$

In order to get one hit with high β per cluster, a loss term is added that maximizes $\beta_{\alpha k}$ for each cluster and minimizes β_i for background hits:

$$L_\beta = \frac{1}{K} \sum_k (1 - \beta_{\alpha k}) + s_B \frac{1}{N_B} \sum_i^N n_i \beta_i.$$

For the attraction-repulsion loss, hits are weighted with

$$q_i = \operatorname{artanh}(\beta_i) + q_{\min},$$

with β_i capped at $1 - \epsilon$ ($\epsilon = 10^{-7}$) and $q_{\text{min}} > 0$, to avoid divergence to 0 or ∞. Nodes from the same cluster are pulled together with the attractive potential

$$V_{\text{attractive},k}(x) = ||x - x_{\alpha k}||^2 q_{\alpha k},$$

while hits from different clusters are pushed apart with

$$V_{\text{repulsive},k}(x) = \max(0, 1 - ||x - x_{\alpha_k}||) q_{\alpha_k}.$$

The repulsive potential is set to 0 if hits are far enough apart to prevent the attractive potential from being overpowered if many clusters exist. Introducing the matrix element $M_{jk} = \mathbb{1}_{j \in C_k}$ that is 1 if hit j belongs to cluster k and 0 otherwise allows for writing the attraction loss as

$$L_{\text{attractive}} = \frac{1}{N} \sum_{j=1}^{N} q_j \sum_{k=1}^{K} M_{jk} V_{\text{attractive},k}(x_j)$$

and the repulsion loss as

$$L_{\text{repulsive}} = \frac{1}{N} \sum_{j=1}^{N} q_j \sum_{k=1}^{K} (1 - M_{jk}) V_{\text{repulsive},k}(x_j).$$

The track parameters are trained with their separate loss terms $L_{\text{parameter}}(p_i)$, weighted by the hit weights $\hat{q}_i$ (similar to q_i with a different $\hat{q}_{\text{min}}$), resulting in:

$$L'_{\text{parameter}} = \frac{\sum_{j=1}^{N} \sum_{k=1}^{K} \hat{q}_j M_{jk} L_{\text{parameter}}(p_k)}{\sum_{j=1}^{N} \sum_{k=1}^{K} \hat{q}_j M_{jk} + \epsilon}.$$

The parameter losses and chosen hyperparameters are described in more detail in [2].

2.2.2. Current Clustering Algorithm

The *CATFinder* uses a *spherical clustering* in its event post-processing. The *spherical clustering* works by finding *condensation points* and assigning one cluster prediction per *condensation point*. This is achieved by applying a $\beta > \tau_\beta$ cut, filtering for possible candidates. These candidates are further filtered by requiring a minimum distance of τ_{d}, which prevents overlapping clusters. τ_{d} is fitted such that an optimal trade-off between fusing different clusters (too large τ_{d}) and splitting up single larger clusters into duplicates (too low τ_{d}) is achieved. All hits within a sphere of radius τ_h of a *condensation point* are assigned to its cluster and clusters with less than 7 hits are rejected as the fitting stage requires at least 7 hits per track.
As the track fitting requires the hits to be ordered according to the flight-path of the particle, a hit-ordering algorithm is employed. If the decay vertex prediction lies in the CDC, the hit closest to it is chosen as a starting point. Otherwise the starting point is extrapolated into the CDC using the momentum prediction. Once the starting point is defined, all other hits are attached by a greedy nearest-neighbor walk. This works well for most cases, but fails for curling tracks, which leave and reenter the CDC. At the time of

writing of this thesis, this hit ordering is still an open problem, workarounds and solution approaches for this challenge are discussed in [2].
Hits that do not lie within τ_h of a *condensation point* are classified as background hits and ignored in the subsequent fitting. The parameters for each track are defined as the parameters of the *condensation point*. As no covariance matrix is predicted, all its elements are set to 0.1.
Figure 2.4 provides a schematic overview of the clustering process, Table 2.1 provides the hyperparameters chosen for the *spherical clustering* according to [2]. If not explicitly stated otherwise, these threshold values are used for comparisons with the *spherical clustering*.

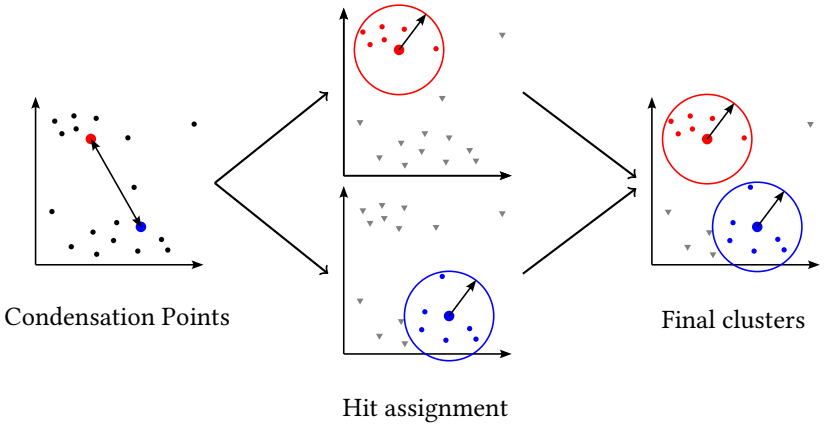


Figure 2.4.: A schematic overview of the *spherical clustering* pipeline. Colored hits are assigned to the cluster belonging to the *condensation point* marked with a larger circle. Grey triangles mark hits classified as background, black circles mark yet unassigned hits.

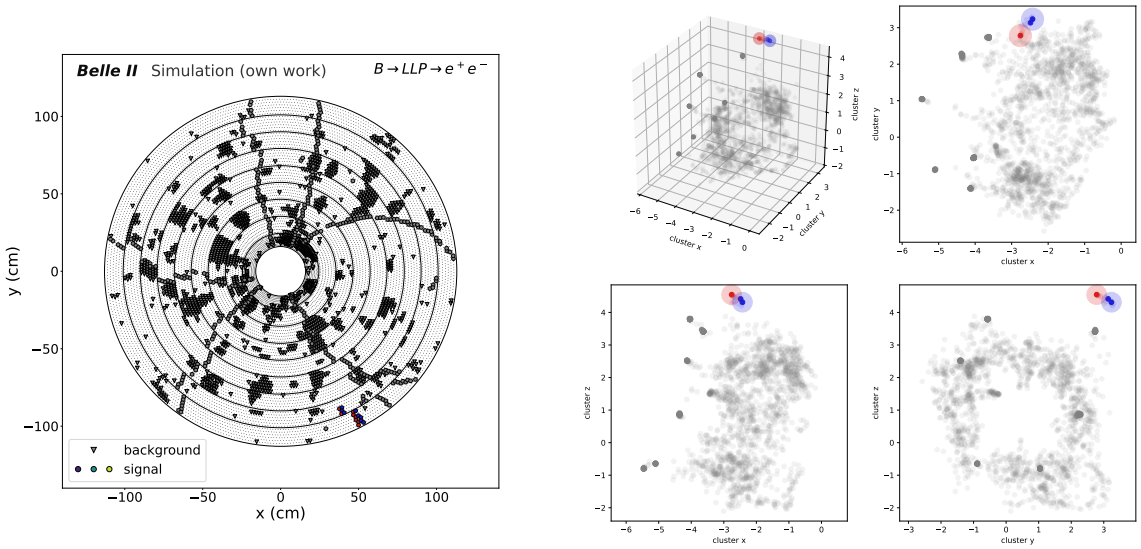
Table 2.1.: Hyperparameters for the *spherical clustering* and working point chosen in [2].

Hyperparameter	Notation	Chosen Value
Condensation point selection	τ_β	0.3
Minimum *condensation point* separation	τ_d	0.3
Cluster radius	τ_h	0.15

2.2.3. Challenging Signatures

The *spherical clustering* works well when the latent space is separated into small clusters with large distances between them. This requirement is fulfilled on events where the GNN is very confident in its prediction and thus clearly separates clusters.
If two tracks lie very close together in the real space, it can be hard to clearly distinguish between them. This results in the GNN predicting two clusters very close together, so these clusters are merged due to the minimum distance requirement of *spherical clustering*.

For samples which contain these kinds of tracks, the track finding performance drops for *CATFinder*. This can happen for tracks with a very low angle between them or displaced tracks with a very high opening angle, as in both cases, the hit-signature is hard to distinguish from that of a single track. An exemplary event where this happens is shown in Figure 2.5. This shows the real and latent space produced by a simulated $\Upsilon(4S) \longrightarrow B\bar{B}$ decay where one B decays to an e^+e^- pair through a Long Lived Particle (LLP). This results in two tracks starting at a very displaced vertex and having a small opening angle. The GNN is able to discern between both tracks, but it does not separate the clusters good enough to be resolved with *spherical clustering*. At least one of the these tracks would be rejected, as the *condensation points* are not far enough apart.



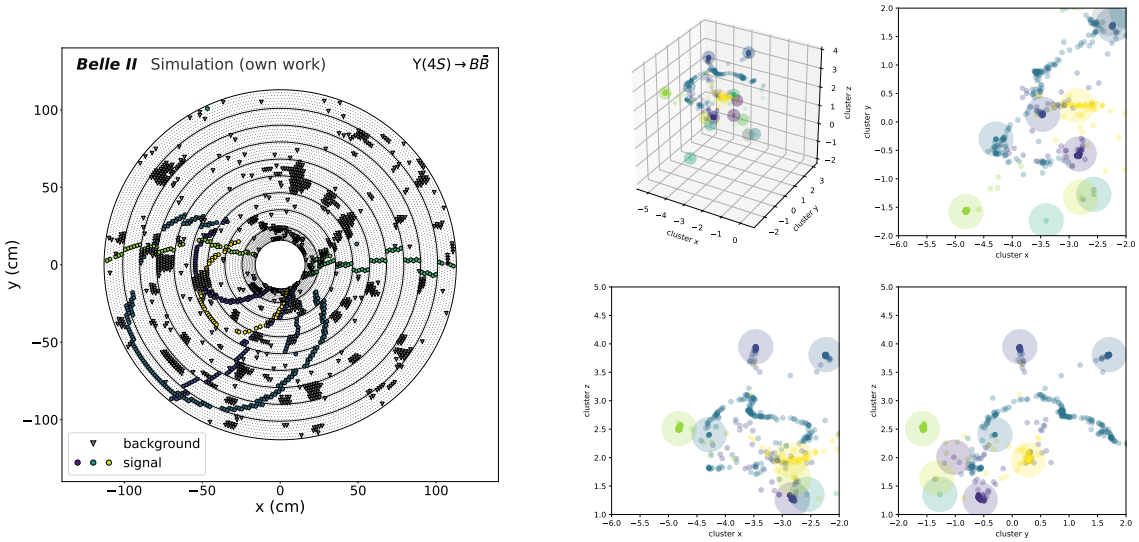
(a) Simulated CDC hits. Triangular markers represent background hits, circular markers show hits matched to a simulated particle. The colored markers represent the hits left by the $e^{\pm}$ produced in the decay of the *LLP*.

(b) The latent space with the hits from the event in Figure 2.5a. Markers represent the latent space coordinates of detector hits with transparency scaled by β. The larger circles represent the spheres with radius τ_h around the *condensation points* of the $e^{\pm}$.

Figure 2.5.: Simulated CDC hits (Figure 2.5a) and the corresponding latent space representation (Figure 2.5b) of the *CATFinder* GNN for a selected $B \longrightarrow LLP \longrightarrow e^+e^-$ event with *high data beam background*.

Additionally, the *spherical clustering* struggles with clusters that are spread out over a larger space, as the fixed radius sphere rejects hits from these tracks or forms duplicate predictions. This happens when the GNN has a low confidence in predicting whether hits in a cluster are caused from signal or background. This has been observed on events with high track multiplicity or low-momentum curling tracks and on events where the training on simulated events does not generalize perfectly to actual collision data (see section 8.2 in [2]). An exemplary event is shown in Figure 2.6, containing the CDC hits of a selected simulated $\Upsilon(4S) \longrightarrow B\bar{B}$ decay under *high data beam background*. This event

contains several tracks which are not clustered perfectly by the *CATFinder*-GNN. Even in the hypothetical best case scenario for the β values which would be getting exactly one *condensation point* per track, the *spherical clustering* looses many hits per track. Depending on the distribution of β values, the tails formed in the latent space are likely to be split into several duplicate clusters.
Efficiently clustering these latent space instances thus requires a more sophisticated clustering algorithm.



(a) Simulated CDC hits. Triangular markers represent background hits, circular markers show hits matched to a simulated particle. The circular markers are colored according to the simulated particle they are matched to.

(b) The latent space with the hits from the event in Figure 2.6a. Markers represent the latent space coordinates of signal detector hits, background hits are excluded. In the 3D plot (top left), the transparency of each hit is scaled by its β value. The larger circles represent the spheres with radius τ_h around a single *condensation point* of each particle.

Figure 2.6.: Simulated CDC hits (Figure 2.6a) and the corresponding latent space representation (Figure 2.6b) of the *CATFinder* GNN for a selected $\Upsilon(4S) \longrightarrow B\bar{B}$ event with *high data beam background*.

2.3. Classical Clustering Algorithms

For the task of segmenting the point cloud created by the *CATFinder*-GNN, a flexible clustering algorithm is needed. As the number of hits in an event varies and there is no defined ordering, this algorithm needs to handle an *unordered* set X of input points with *variable size*. All points $x \in X$ have to be assigned to tracks C_k or be labeled as background noise B. The number of tracks in each event also *varies*, so the algorithm has to be able to handle a varying number of clusters.

This section will introduce several clustering algorithms that were proposed for different object condensation tasks and are able to handle these requirements.

DBSCAN DBSCAN [11] identifies clusters as regions of high point density separated by regions of lower density. Given the parameters ϵ and MinPts, a point is called a *core point* if at least MinPts points lie within its ϵ-neighborhood. Starting from an unvisited core point, DBSCAN iteratively expands a cluster by adding all points that are density-reachable from it. Points that are not core points but lie within the ϵ-neighborhood of a core point can be assigned to the cluster as border points, while points that are not assigned to any cluster are labeled as noise. Clusters are then defined as the maximal sets of density-connected points.
Due to its popularity, highly efficient implementations of DBSCAN exist which are suitable for HEP tasks. DBSCAN is used in several proposed object condensation pipelines like for charged particle tracking at High Luminosity LHC [12]. But due to its iterative approach, DBSCAN is not optimal for tasks with strict low-latency requirements like the HLT at Belle II, where *CATFinder* aims at being applied.

Density Peak Clustering Similar to DBSCAN, Density Peak Clustering (DPC) [13] finds clusters based on the distribution of point densities. This algorithm is also used in a proposed collider agnostic object condensation pipeline for end-to-end event reconstruction [14]. The local density ρ of a point is defined as the number of points within a d_c-neighborhood, with neighbors optionally weighted by a point feature. Additionally, the minimum distance δ to a point with higher density is introduced. *Condensation points* are found by introducing two thresholds and selecting points with $\rho > \tau_\rho$ and $\delta > \delta_{\min}$. All remaining points are assigned to the cluster of the nearest *condensation point*. In the variant used in [14], points that are not within a distance d_p of a *condensation point* are labeled as noise.
This approach maps closely to the idea behind *spherical clustering*, only changing the *condensation point* filtering from a cut on β to a minimum local density. It is thus not able to solve all of the challenges identified in Subsection 2.2.3, as it still enforces a minimum distance $\delta_{\min}$ between *condensation points* and a maximum cluster radius d_p.

Variational Density Peak Clustering A more flexible variant of DPC is Variational Density Peak Clustering (VDPC) [15], which removes the global density threshold τ_ρ and groups the points into density levels. DBSCAN with heuristic thresholds is used for high-density regions, removing the maximum cluster radius. Shared Nearest Neighbor Clustering (SNNC) discriminates low-density clusters from points at the border of high-density clusters, segmenting what DPC would classify as noise into additional clusters. Noise clusters can then be rejected using a lower threshold on τ_β or using a point-wise foreground confidence predicted by the *CATFinder*-GNN.
Due to its reliance on DBSCAN, VDPC also includes iterative steps, resulting in lower parallelizability, without removing the minimum *condensation point* separation distance enforced by $\delta_{\min}$.

2.4. Machine Learning Approaches

To develop a more flexible and non-iterative clustering algorithm, a machine learning approach is developed. While classical algorithms directly operate on sets of points and infer the number of clusters automatically, standard machine learning architectures rely on a fixed number of inputs and/or outputs. The following section describes two architectures that each partially solve these challenges and are combined and refined into the *neural clustering* algorithm proposed in Section 4.1. First, the architectural design behind PointNet is described, showing how this enables efficient operation on point clouds without the computational overhead of graph-building in GNNs. Then, an overview of the You Only Look Once (YOLO)-pipeline is provided, showing how YOLO handles the task of predicting a varying number of outputs without requiring iterative steps.

PointNet PointNet [16] is designed for the tasks of point cloud classification and semantic segmentation. In order to segment large point clouds efficiently, PointNet restricts its architecture to operations that scale (sub-)linear with the number of input points and can efficiently be performed on GPUs. The inference has to be invariant under permutations of the inputs, as no stable ordering for points in higher dimension exists (see [16]). Thus all operators are additionally required to be invariant under input order.
To achieve this, PointNet restricts itself to three operators:

- *Matrix multiplications* are used to align the point cloud and feature vectors to a learned canonical representation in order to provide invariance under affine transformations of the real space.
- *Multi-Layer Perceptrons (MLPs)* are only used point-wise, mapping one input point into one feature vector. For the MLP stages, the same MLP is applied to all points in parallel, mapping n independent inputs into n independent embeddings.
- *Pooling steps* are the only steps that aggregate information between points. A simple maximum pooling of feature embeddings is performed to generate a global feature, which is concatenated to the point-wise features.

This results in an architecture similar to a GNN, but without graph-building steps or nearest-neighbor searches. A more detailed architecture overview is provided in [16].
The shared MLP-stages and feature aggregation via max-pooling are adopted for the *neural clustering*, enabling inference on unordered inputs.

YOLO YOLO [17] is a computer vision pipeline for the task of predicting bounding-boxes for multiple objects in an image in one shot. To achieve this, YOLO predicts several bounding-boxes, confidence scores and corresponding class labels at each anchor point in a fixed grid. This results in a fixed, but large count of bounding-box predictions, where overlapping bounding-boxes are filtered using Non-Maximum Suppression (NMS): First, all bounding-boxes with a confidence score below a fixed threshold are rejected. Then, the remaining bounding-boxes are compared pairwise, with the Intersection over Union (IOU) ratio of their areas as a similarity measure. Bounding-boxes with an IOU higher

than a fixed threshold τ_{iou} are identified as the same object. For each object prediction, only the bounding-box with highest confidence score is kept and all other (non-maximum) bounding-boxes are discarded (suppressed).
The *neural clustering* proposed in this work adopts the idea of predicting a bounding-box per anchor point (similar to the sphere per *condensation point* in *spherical clustering*), and the removal of duplicate predictions detected by high IOU. This enables automatic inference of the cluster count.

3. Evaluation Setup

The training, fitting and evaluation of the *neural clustering* requires a diverse set of data sets to provide a clear differentiation between training and validation data, as well as topology-specific data sets for a fine-grained evaluation. This chapter describes the simulated data sets used in this thesis and introduces the metrics used for a robust evaluation.
First, the main data sets required for training, optimization and evaluation of the clustering model and pipeline are described. Then, the metrics determining the quality of the assignment of hits to tracks are defined.

3.1. Data Set

The development and validation of a clustering algorithm requires simulated data sets, which are described in this section. First, the general simulation of detector hits for training and evaluation is described. Then, the generation of the cluster space predictions that are used as input for the clustering is explained. Finally, several distinct event topologies used for different tasks are discussed.

3.1.1. Simulation

Detector Output In data from real particle collisions, the truth matching between hits and actual particles and background is not known and has to be inferred using the track finding algorithm. Measured data is thus not able to provide the information needed to train the clustering model and evaluate the impact of different hyperparameters. Unless explicitly labeled as *data events/samples*, all samples in this thesis are thus generated using a simulation. This simulation is performed using basf2, which simulates the interaction of generated particles with the detector material and the influence of the complete detector geometry using GEANT4 [18]. The response of the detector is simulated and digitized by basf2, resulting in simulations that closely match actual collision data from Belle II.
The full development from training to evaluation is performed on samples with a high level of beam background taken from randomly triggered events that have a low probability to contain data from actual collisions. If not specified otherwise, the simulations used in this work are overlayed with these *high data beam background* events. The background samples are described in more detail in subsubsection 3.1.1 in [2]. The clustering performance is also evaluated on simulated and on measured events with lower background levels, showing no significant difference in performance (see Section A.3).

Cluster Space In order to achieve a fair comparison between different clustering approaches, all clustering algorithms are evaluated on the same inputs. The cluster space

prediction of the *CATFinder*-GNN is precalculated and then distributed to the different clustering algorithms for all samples. As a backbone, the Graph Neural Network (GNN) used in the *CAT on B Finder* from [2] is utilized as this provides the best performance (see [2]). A different model is only used for the ablation study with 5D latent space (see Section 7.3) as this requires changes in the GNN architecture. This model relies on an additional input feature that does not generalize well from simulation to actual data, thus its results are not directly comparable.
Whenever clustering algorithms are compared with approaches that do not share the same *CATFinder*-GNN backbone, like the *Baseline* or the fine-tuned end-to-end model in Section 7.5, the approaches are compared on the same underlying detector outputs.

3.1.2. Sample Types

The development and evaluation of the *neural clustering* utilizes several distinct event topologies. These topologies and their simulation are described in more detail in this subsection. An overview of the samples is provided in Table 3.1, the measured collision events used for the validation are introduced in Section 8.3.

Table 3.1.: Overview of simulated samples used throughout this work.

category	**sample**	**generator**	**description**
prompt $\mu^\pm$	$\mu^\pm$ with low p_t	–	Single prompt track with low p_t
vertex $\mu^\pm$	displaced $\mu^\pm$	–	1–3 displaced vertices
radiative $\mu^+\mu^-$	$e^+e^- \to \mu^-\mu^+(\gamma)$	KKMC [19]	Two $\mu^\pm$ tracks with high p_t
radiative $\pi^+\pi^-$	$e^+e^- \to \pi^+\pi^-\gamma$	Phokhara [20]	Two $\pi^\pm$ tracks with low opening angle
$B\bar{B}$	$\Upsilon(4S) \to B\bar{B}$	EvtGen [21]	High number of low p_t $\pi^\pm$ tracks
B^+B^-	$\Upsilon(4S) \to B^+B^-$	EvtGen [21]	High number of low p_t $\pi^\pm$ tracks
B to LLP	$B \to LLP \to e^+e^-$	EvtGen [21]	Displaced vertex with low opening angle in high multiplicity event

Technical Muons As minimally ionizing particles, $\mu^\pm$ tracks are affected comparatively little by interactions with the detector material, making them good candidates for simpler track topologies. In this work, two data sets using technical $\mu^\pm$ are used.
First a prompt sample is generated, containing one $\mu^\pm$ track starting at the Interaction Point (IP) with randomly chosen charge and a low transverse momentum $p_t \in [0.039\,\text{GeV}, 0.3\,\text{GeV}]$. This contains $\mu^\pm$ which just barely enter the Central Drift Chamber (CDC), $\mu^\pm$ curling inside the CDC and $\mu^\pm$ which leave the CDC, loose energy in the Electromagnetic Calorimeter (ECL) and reenter the CDC. The polar angle of each track is

chosen randomly from the full CDC acceptance region $\theta \in [17°, 150°]$.
A second $\mu^{\pm}$ sample contains displaced vertex decays. These result in two $\mu^{\pm}$ tracks starting at the same point inside the CDC instead of starting at the IP. The decay vertex is sampled inside the CDC acceptance, with a radial distance to the IP $r \in [0\,\text{cm}, 100\,\text{cm}]$. Two opening angle distributions are combined with the same weight: a uniform base distribution $\theta \in [0°, 90°]$ and an exponential distribution focusing on low opening angles $\theta \in [0°, 25°]$. The vertex samples contain $1-3$ displaced vertices, resulting in $2-6$ $\mu^{\pm}$ tracks. These samples allow a direct study of the effect of vertex displacement and opening angle, eliminating correlations to other topological factors.

Low Multiplicity Events To evaluate the performance on physically consistent events, radiative dimuon events ($e^+e^- \to \mu^-\mu^+(\gamma)$) are generated using the KKMC [19] Generator. These events contain two $\mu^{\pm}$ tracks with mostly high transverse momentum and serve as the simplest event topology for basic validation. As this event topology is also used to calibrate the detector, it is an important signature for performance evaluation.
Decays with a high boost in one direction lead to tracks with a very low opening angle. Separating the hits from these tracks is a challenging task for track finding algorithms. To evaluate the corresponding performance on physical events, $e^+e^- \to \pi^+\pi^-\gamma$ decays are simulated using the Phokhara [20] Generator. In these decays the $\pi^{\pm}$ tracks often form a tight opening angle.

High Multiplicity Events As a primary focus of the Belle II experiment is the $\Upsilon(4S) \to B\bar{B}$ decay [3], this sample is included at all steps in the development pipeline. The decay of B-mesons contains several challenging signatures for track finding. In Belle II an average of 11 charged particles is expected in the CDC for $B\bar{B}$ events with several of these tracks formed by $\pi^{\pm}$ with low transverse momentum [5]. This results in *curling tracks* and tracks that overlap with each other due to the high track multiplicity. These decays are generated with the EvtGen [21] Generator.
A very similar sample is generated using EvtGen [21] to produce $\Upsilon(4S) \to B^+B^-$ events. To evaluate the performance on displaced tracks with a small opening angle in the challenging environment of $B\bar{B}$ decays, additional samples are generated with EvtGen [21], where one of the B-mesons decays normally while the other decays to a hypothetical long lived particle S with a mass of 0.22 GeV. After a randomly sampled lifetime with the maximum being equivalent to traveling to the outer edge of the CDC barrel region, the S decays into an e^+e^- pair. Due to the high mass difference between the B and S, the $e^{\pm}$ pair is highly boosted, resulting in most pairs having a small opening angle.

3.2. Metrics

The optimization and evaluation of the clustering is performed using metrics that apply after track finding or fitting. In order to directly optimize the performance gain, not the per-hit clustering metrics are reported, instead the metrics are reported track-wise or aggregated over tracks from several events. This section aims to provide an overview of the metrics utilized in this work, closely following the definitions provided in [1].

Per Track Metrics An ideal track prediction contains exactly the hits belonging to the particle that created the track. But track predictions are not perfect and thus not all true hits are found while additional fake hits stemming from other particles or beam background are included. To quantify the quality of a given track prediction, two metrics are introduced.
The *hit efficiency* ε_{hit} is defined as the number of CDC hits *matched* to a simulated particle and included in a track prediction divided by the total number of CDC hits left by this particle in the complete event:

$$\varepsilon_{\text{hit}} = \frac{n_{\text{hits}}(\text{matched and} \in \text{track})}{n_{\text{hits}}(\text{matched})}. \tag{3.1}$$

A *hit efficiency* of $\varepsilon_{\text{hit}} = 1.0$ indicates that a track prediction contains all CDC hits of a *matched* particle.
Complementary, the *hit purity* $\mathfrak{p}_{\text{hit}}$ is defined as the proportion of CDC hits in a track prediction stemming from the particle matched to this track:

$$\mathfrak{p}_{\text{hit}} = \frac{n_{\text{hits}}(\text{matched and} \in \text{track})}{n_{\text{hits}}(\in \text{track})}. \tag{3.2}$$

A *hit purity* of $\mathfrak{p}_{\text{hit}} = 1.0$ indicates that a track prediction contains no CDC hits that belong to other particles or background.

Particle Matching The *hit efficiency* and *hit purity* are used to match track predictions to particles of the simulation. First, tracks are defined as *related* to a particle if $\varepsilon_{\text{hit}} > 0.05$, $\mathfrak{p}_{\text{hit}} > 0.66$ and $n_{\text{hits}}(\in \text{track}) > 7$. This allows for a matching even in the case of *curling tracks* which leave many CDC hits in the detector but can be fitted using only a smaller fraction of these hits. Of all track predictions *related* to a particle, the one with highest $\mathfrak{p}_{\text{hit}}$ is *matched* to the particle, all others are denoted as *clone tracks*. Tracks that are not *related* to any particle are defined as *fake tracks*.

Aggregated Metrics The performance across particles from a larger set of events is aggregated by filtering all tracks according to the particle that they are *related* to and additionally keeping *fake tracks*. Once the targeted track predictions are selected, they can be evaluated with the following metrics.
The *track efficiency* is defined as the number of *matched tracks* divided by the number of simulated particles that are matched to at least one CDC hit

$$\varepsilon_{\text{trk}} = \frac{n_{\text{trks}}(\text{matched to a particle})}{n_{\text{simulated}}(\geq 1\text{matched hit})}. \tag{3.3}$$

Similarly the *track charge efficiency* is defined as the number of *matched tracks* with a correctly reconstructed charge divided by the number of simulated particles matched to at least one CDC hit

$$\varepsilon_{\text{trk,ch}} = \frac{n_{\text{trks}}(\text{matched to a particle, correct charge})}{n_{\text{simulated}}(\geq 1\text{matched hit})}. \tag{3.4}$$

The *track purity* is defined as the ratio of *matched tracks* to the number of all *found tracks*

$$\mathfrak{p}_{\text{trk}} = \frac{n_{\text{trks}}(\text{matched to a particle})}{n_{\text{trks}}}. \tag{3.5}$$

The *clone rate* is the ratio of *clone tracks* to the number of all tracks that are *related* to a particle

$$\mathfrak{r}_{\text{clone}} = \frac{n_{\text{clone trks}}}{n_{\text{trks}}(\text{related to a particle})}. \tag{3.6}$$

The *fake rate* is the ratio of *fake tracks* to the number of all *found tracks*

$$\mathfrak{r}_{\text{fake}} = \frac{n_{\text{fake trks}}}{n_{\text{trks}}}. \tag{3.7}$$

The *wrong charge rate* is the ratio of *matched tracks* reconstructed with the wrong charge divided by the number of *matched tracks*

$$\mathfrak{r}_{\text{wrong ch.}} = \frac{n_{\text{trks}}(\text{matched to a particle, wrong charge})}{n_{\text{trks}}(\text{matched to a particle})}. \tag{3.8}$$

The resolution of the predicted or fitted momentum is reported separated into the axial (p_z) and radial (p_t) components. The resolution for a *matched track* is defined as the relative error to the simulated values

$$\eta(p_{t,z}) = \frac{p_{t,z,\text{reconstructed}} - p_{t,z,\text{simulated}}}{p_{t,z,\text{simulated}}}. \tag{3.9}$$

The resolution for a set of tracks is then defined as the 68% coverage

$$r(p_{t,z}) = P_{68\%}(|\eta(p_{t,z}) - P_{50\%(\eta(p_{t,z}))}), \tag{3.10}$$

with P_q being the qth quantile of the distribution.
Found tracks can be unable to pass the fitting step, so the metrics are discriminated between *finding* and *fitting* versions which apply *before* and *after* the fitting step. If not stated explicitly as *finding*, track metrics in this thesis refer to the state post-fitting.
The definitions of particle matching, clone and fake tracks, as well as the aggregated tracking metrics used in this section closely follow those introduced in [1].

4. Proposed Clustering Algorithm

The main development proposed in this thesis is the new clustering algorithm for the object condensation latent space. This task requires an algorithm that can handle unordered input point clouds with varying length and arbitrary latent space dimension. The clustering should result in a varying number of output clusters as well as a background discrimination. The following chapter describes how the core ideas behind two different machine learning approaches are combined to solve this task. Clusters vary in shape size and location and different inputs may contain a different number of clusters. This motivates the use of several ideas of the You Only Look Once (YOLO)[17] bounding-box prediction pipeline, employing parallels to computer vision tasks. The architecture of PointNet[16] is adapted to handle the unordered point cloud.
This chapter first introduces the proposed clustering pipeline and then describes the neural network architecture used in the cluster assignment in more detail.

4.1. Clustering Pipeline

Predicting bounding-boxes per anchor point and discarding redundant predictions enables YOLO to infer the number of objects contained in the image and handle the varying object counts with a fixed architecture (see Paragraph 2.4). But YOLO operates on grids of pixels (or voxels if applied to 3D input) while the cluster space consists of an unordered, sparsely occupied point cloud. This thesis focuses on adapting the YOLO pipeline for application on point clouds instead of voxelizing the cluster space for direct use with YOLO. Voxelization would reduce the information content of the cluster space and reintroduce a small but still fixed minimum distance between clusters due to its finite resolution. Increasing the resolution would directly affect the computational effort required per inference. Working directly on the point cloud enables the pipeline to efficiently use the sparsity of the cluster space as only the actual points are used as input, instead of the full cluster space.
The proposed clustering algorithm works by selecting candidate points as cluster centers, predicting which of the nearest neighbors of each center belong to the cluster of this center and finally merging overlapping clusters. How this is realized is shown in more detail in the following sections.
Figure 4.1 shows a graphical overview of the clustering pipeline. A full algorithmic overview is provided in Appendix A as Algorithm 1.

Condensation Point Selection For YOLO, the fixed grid of anchor points limits the maximum object count and assumes that objects are distributed throughout the whole image [17].
To solve these limitations the β-parameter prediction of the *CATFinder*-GNN is used. This

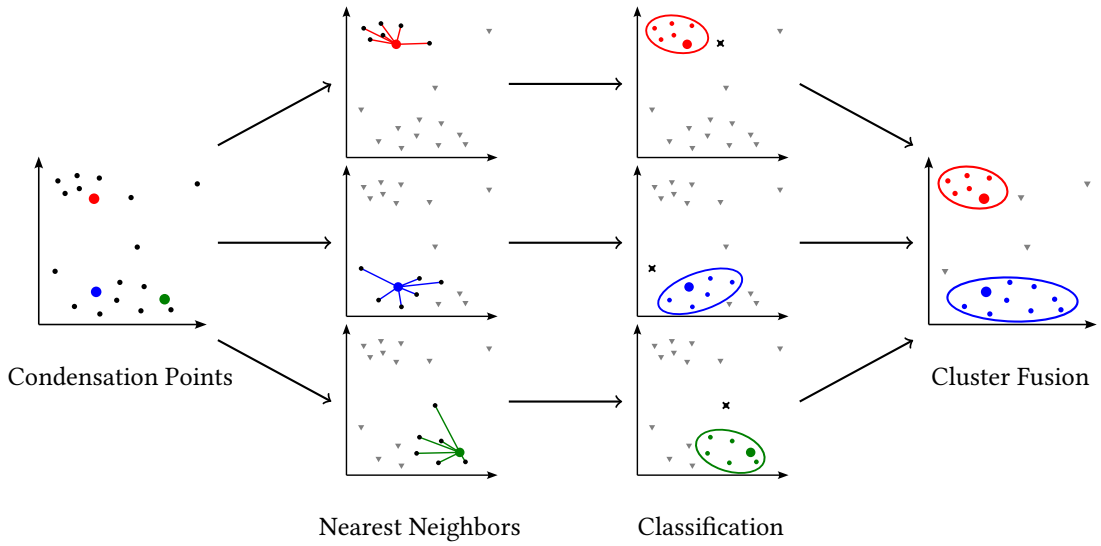
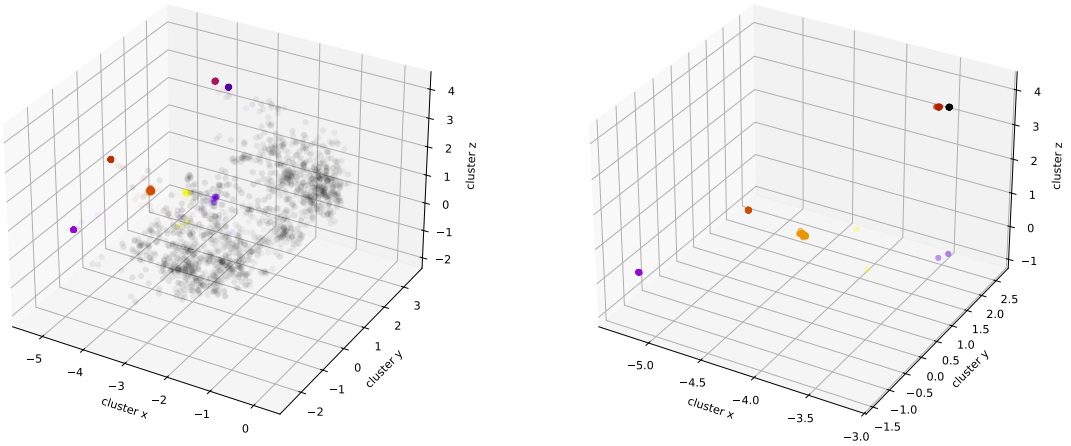


Figure 4.1.: The *neural clustering* pipeline. Colored hits are assigned to the cluster belonging to the *condensation point* marked with a larger circle. Grey triangles mark hits classified as background, black circles mark yet unassigned hits.

parameter predicts per-hit, how confident the *CATFinder*-GNN is that the hit is a good cluster-center candidate [4]. All input points with $\beta > \tau_\beta$ are selected as anchor points (for consistency with the naming convention used for object condensation, these will be called *condensation points* from now on) and predicting one cluster per *condensation point*. This corresponds to the first step in Figure 4.1 with β encoded as the marker size. If no points with high enough β exist, the pipeline returns no cluster predictions.
This is similar to the currently used *spherical clustering* but instead of enforcing a minimum separation distance, all *condensation points* are kept. In contrast to the *spherical clustering*, several *condensation points* can belong to the same cluster with duplicates being resolved in the final *cluster fusion* step. Consequently, almost no findable clusters are rejected at this stage, as this keeps all hits where the *CATFinder*-GNN is confident in its track-parameter prediction. Clusters where the *CATFinder*-GNN labels no point with non-zero β are likely not findable in the *CATFinder*-GNN latent space, as the *CATFinder*-GNN only clusters hits together where it assumes the existence of a cluster. Additionally, no clusters are fused yet, as nearby *condensation points* are not rejected. The result of this step is a list of *condensation points*, which are processed into cluster predictions. This resulting latent space is shown on the event produced by a simulated $\Upsilon(4S) \longrightarrow B\bar{B}$ decay in Figure 4.2.

Nearest Neighbor Selection Each *condensation point* is treated as the seed for a new cluster and all further processing happens for all *condensation points* in parallel. The neural network (see Section 4.2) is used to classify which points belong to the cluster of a given *condensation point*. As clusters are locally connected, only points that are close to the *condensation point* are considered as possible candidates for its cluster. To reduce the computational cost and to get a constant number of inputs for the neural network, only the k nearest neighbors of the *condensation point* are classified by the neural network,



(a) All hits.

(b) Hits with $\beta \geq 0.3$.

Figure 4.2.: Latent space of the on a simulated $\Upsilon(4S) \longrightarrow B\bar{B}$ decay before and after selecting *condensation points* with a threshold of $\tau_\beta = 0.3$. Hits are colored according to which simulated particle they belong to with gray markers for background hits. β is encoded as marker opacity.

while points that are further away are defined as not belonging to this cluster. k is chosen such that most clusters contain less than k points. This requirement is not strict, as larger clusters can still be found by merging several cluster predictions. An ablation study is performed in Section 7.1, showing that decreasing k is a viable option for reducing computational cost without sacrificing performance.
If the complete event contains less than k hits, all hits are provided to the neural network as the architecture does not prescribe a fixed number of inputs. This can result in a lower classification quality but events with less than 100 hits are not expected under current background conditions at Belle II [1]. If an additional hit-cleanup step is employed, this should be addressed by including corresponding samples in the training of the neural network. As the architecture does not require a fixed input count, it is likely that the model can learn to successfully predict clusters even in events with a low number of hits.

Neighbor Classification For each *condensation point* x_c, the relative spatial coordinates $x_i - x_c$ of the k nearest neighbors of x_c are provided to the neural network described in Section 4.2. The neural network outputs a score $p_{i,c}$ per input point which is interpreted as the confidence that point x_i lies in the same cluster as the *condensation point* x_c. All points with $p_{i,c} > \tau_\mathrm{p}$ are considered to belong to the cluster of the *condensation point* x_c. Varying τ_p affects how many points are added to each cluster, resulting in a trade-off between hit efficiency and hit purity. For this thesis, τ_p is fixed at $\tau_\mathrm{p} = 0.5$ to reduce the size of the working point search and prevent overfitting to the sample set used for hyperparameter optimization. Testing different values of τ_p can still be helpful to provide a more detailed optimization in future studies.

Cluster Fusion As no minimum separation of *condensation points* is enforced, it is possible that several selected *condensation points* belong to the same cluster.
This is resolved by adapting the filtering of overlapping bounding-boxes in YOLO to point clouds. Cluster predictions are interpreted as sets of points similar to the sets of pixels inside bounding-boxes in YOLO. The prediction confidence is replaced by the β-value of the *condensation point*, as this correlates with the prediction confidence of the *CATFinder*-GNN [4]. The Intersection over Union (IOU) is calculated with the number of points in the intersection and union of both sets, which can efficiently be implemented with the cluster predictions being represented by bitvectors.
As sets of points are not bound to form any geometric shape, overlapping sets can be fused instead of discarding duplicates. This results in clusters of arbitrary size, enabling the cluster algorithm to find clusters with more than k points.
The cluster fusion step provides the final clusters, in general preventing duplicates. If a hit would be assigned to more than one cluster, it is only added to the cluster with highest cluster confidence (by β), as the subsequent fitting stages currently do not handle double assignments.

4.2. Neural Network Architecture

A central distinction between point clouds and other machine learning inputs like images is the fact that the input points have no constant ordering. Not only is ordering the input points inefficient, it is mathematically impossible to find a stable ordering in higher dimensions [16]. The architecture is thus required to be invariant across input orderings. A possible approach for this would be the usage of a Graph Neural Network (GNN), but the architectural complexity can further be reduced by employing the architecture proposed by PointNet [16]. PointNet simplifies the GNN approach by replacing the message passing with a global pooling step. Instead of messages being passed to the neighbors in a graph, all messages are combined into a global feature vector through max pooling and then distributed to the nodes. Thus, PointNet does not require the graph-building step, which is time-consuming and hard to port to application-specific hardware like Field Programmable Gate Arrays (FPGAs). Replacing message passing with a global feature aggregation suffices for this clustering task, as the locality of cluster space is already utilized through the nearest neighbor selection and does not need to be included via a local graph structure. For this thesis, the architecture of PointNet is scaled down to the requirements of a single clustering step.
This approach enables the model to scale efficiently with the cluster space dimension, making use of the cluster space sparsity. Additionally, the input features can be enhanced with other GNN-outputs or analog detector outputs without changing the architecture, by varying the input dimension.
In this section, the architecture of the neural network is described in detail.

Model Architecture The clustering model takes a list of points as input and starts by mapping each of these to a feature embedding with dimension h_2. The input consists of the cluster space coordinates of each point relative to the selected *condensation point*.

Thus, for each inference, the *condensation point* is located at the cluster space origin and it is not necessary to provide its location to the model. The effect of adding additional features is tested in the ablation study in Section 7.4. The input embedding is performed by an Multi-Layer Perceptron (MLP) via one hidden layer h_1 with Rectified Linear Unit (ReLU)-activation. The MLP is shared across all input hits, making this step independent of the input order. The feature embeddings are then combined into one global feature vector. This pooling is the only step in the inference that aggregates features across inputs, so it is critical to make this step input order invariant, which is achieved with a single max pooling operation over all feature vectors. The global vector is then concatenated to each of the pointwise feature vectors, combining local and global information. The concatenated feature vectors are then mapped to a single output per point with a second shared MLP with one hidden layer h_3 with ReLU-activation and a sigmoid output activation, resulting in a confidence vector $\hat{y} \in [0, 1]^k$. The dimensions of the feature embedding and hidden layers are chosen by evaluating models trained with different settings (see Section 5.1). A schematic overview over the architectural design is provided in Figure 4.3.

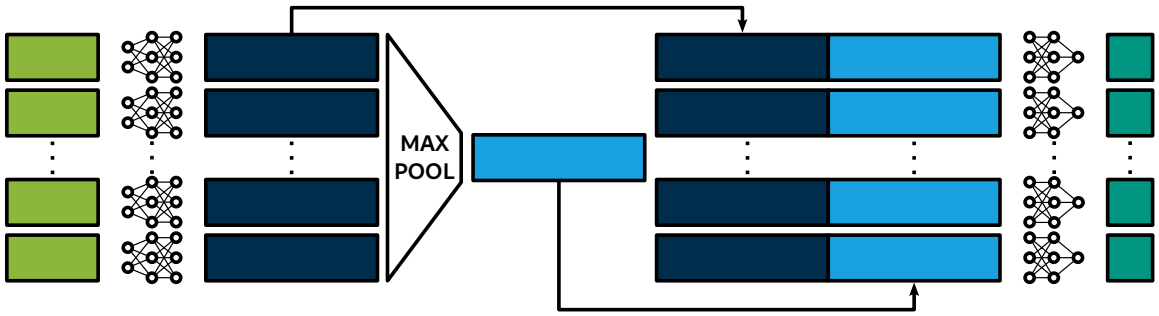


Figure 4.3.: The neural network architecture used for hit classification. Inputs are the relative coordinates of the k nearest neighbors of a *condensation point* optionally including additional features. Each input vector is mapped to a confidence score in $[0, 1]$, indicating whether it belongs to the cluster of the *condensation point*. Green boxes indicate inputs and outputs per hit, blue boxes represent local and global feature embeddings.

Loss The model is trained supervised on a set of *CATFinder*-GNN latent space predictions for simulated events. A Binary Cross Entropy loss is chosen, comparing the predictions $\hat{y}$ with the ground truth $y \in \{0, 1\}^k$. y indicates for each hit, whether it is generated by the same simulated particle as the hit chosen as *condensation point*. If the *condensation point* itself is caused by induced background instead of a simulated particle, the nearest neighbors are defined as not belonging to its cluster, even if they are also caused by background.
The actual training data and scheduling- as well as early-stopping-pipeline is described in Section 5.1.

5. Implementation

To evaluate the new clustering algorithm in a real-world application and simultaneously improve the track finding at the Belle II experiment, the pipeline proposed in Chapter 4 is fine-tuned for this use-case. This requires training the neural network and optimizing the working point of the pipeline specific to the actual application. This chapter describes the process of implementing the *neural clustering* for the Graph Neural Network (GNN)-based track finding at Belle II.
First, the process of training the neural network and selecting the optimal model size is illustrated. Then, the procedure of determining an optimal set of thresholds for the clustering pipeline is presented.

5.1. Training

The clustering model is trained on the latent space output of the *CATFinder*-GNN combined with truth information from the underlying simulation.

Data Set The training data set is comprised of samples from the *displaced* $\mu^{\pm}$, $B\bar{B}$ and $B^{+}B^{-}$ samples (see Section 3.1). This combines the high-multiplicity B decays with low multiplicity, but displaced $\mu^{\pm}$-pairs with an additional focus on pairs with low opening angle. The data set focuses on challenging track signatures, as these samples still contain enough clusters which are easier to locate, so no additional inclusion of simpler topologies is necessary for the model to learn to generalize.
The data set is generated by running the *CATFinder*-GNN-inference on the input samples, for each event randomly selecting one *condensation point* with $\beta > 0.5$ and labeling its k nearest neighbors. If the sampled *condensation point* references a background hit, all nearest neighbors are defined as not belonging to its cluster ($y = 0^k$).
Table 5.1 provides an overview of the sample distribution.

Table 5.1.: Composition of the training data set.

Sample type	# events	# charged particles
vertex $\mu^{\pm}$	20 000	80 000
$B\bar{B}$	2 000	~22 000
$B^{+}B^{-}$	2 000	~22 000
total	24 000	~124 000

Parameters The model is trained with an automatic learning rate scheduler and early stopping. When the validation loss does not improve for several epochs, the learning rate is reduced by a fixed factor. If a reduction of the learning rate does also not lead to improvement in validation loss, the training is stopped and the epoch with best validation loss is saved. This introduces several training-specific hyperparameters: The start-off learning rate lr, the learning rate reduction factor lr_{reduce}, the scheduling patience p_{lr} and the early stopping patience p_{es}.
These parameters influence the training convergence and thus the required training time as well as the model performance.
To assess the influence on model performance, a small training gridsearch is run and the clustering is evaluated by comparing $\mathcal{F} = \varepsilon_{\text{trk,ch}} + \mathfrak{p}_{\text{trk}}$ from [1] on a sample with 2000 $B\bar{B}$ decays. A test on the coarse grid of

$$\begin{aligned} lr &\in \{10^{-2}, 5{\times}10^{-3}, 10^{-3}, 5{\times}10^{-4}\} \\ lr_{\text{reduce}} &\in \{0.5, 0.75\} \\ p_{\text{lr}} &\in \{3\} \\ p_{\text{es}} &\in \{5, 10\} \end{aligned}$$

results in no significant variation in performance. Neither $\varepsilon_{\text{trk,ch}}$ nor $\mathfrak{p}_{\text{trk}}$ vary significantly above the statistical uncertainty. The best model achieves $\varepsilon_{\text{trk,ch}} = 0.541 \pm 0.003$ and $\mathfrak{p}_{\text{trk}} = 0.965 \pm 0.001$ while the deviation between models lies at ± 0.0012 for $\varepsilon_{\text{trk,ch}}$ and ± 0.0007 for $\mathfrak{p}_{\text{trk}}$.
With the small model and data set size, the total difference between training times lies below the computational effort to perform a more detailed gridsearch for the training parameters. As no difference in model performance is detected, all further training will use the parameters with the slowest but most stable convergence in the above gridsearch. Thus the training parameters are set to

$$\begin{aligned} lr &= 5{\times}10^{-4} \\ lr_{\text{reduce}} &= 0.5 \\ p_{\text{lr}} &= 3 \\ p_{\text{es}} &= 10 \end{aligned}$$

if not explicitly stated otherwise.

5.2. Hyperparameter Optimization

The clustering algorithm depends on a set of hyperparameters which have to be fitted on task-specific data. These parameters belong to two different groups: training-specific and inference-specific parameters. Training-specific parameters include the dimensions of the hidden layers and feature vectors in the model, as well as learning rate and scheduling parameters. At inference, the algorithm requires thresholds for *condensation point* selection and the fusion of clusters. Other parameters like the number of nearest neighbors, the cluster space dimension and the in- or exclusion of additional input features are fixed for

model selection and varied later in the ablation studies (see Chapter 7). The following section shows in detail how the hyperparameter values are attained. An overview of all hyperparameters is provided in Table 5.2.

Table 5.2.: Search space for fixed-, training- and inference-hyperparameters

Hyperparameter	Examined range	Result
Number of nearest neighbors k	150	150
Input cluster space dimension	3	3
Model parameter count	{0.7k, 10k, 165k}	10k
Condensation point selection threshold τ_β	$[10^{-20}, 0.9]$	0.3
Cluster fusion threshold τ_{iou}	[0.05, 0.95]	0.05

5.2.1. Fixed Parameters

Number of Nearest Neighbors One of the central hyperparameters in the clustering algorithm is the choice of how many nearest neighbors of each *condensation point* are forwarded to the clustering model. Increasing k provides more context to the model, resulting in a better classification. But this also increases computational cost, as both the search for nearest neighbors and the model inference are slower if k is increased. As these two steps are performed in parallel for all *condensation points*, unnecessarily increasing their computational cost should be avoided.

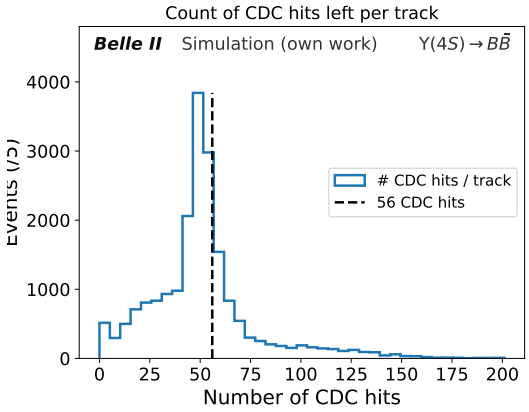


Figure 5.1.: Distribution of the number of matched hits per track in $\Upsilon(4S) \longrightarrow B\bar{B}$ decays.

Figure 5.1 shows that the distribution of how many points belong to each cluster peaks little below 56 points, corresponding to particles that traverse the whole Central Drift Chamber (CDC) and leave a hit in almost all of its 56 wire layers. Almost all ($> 99\%$) clusters contain less than 150 points, thus choosing $k = 150$ covers most clusters. Clusters with more points can still be found, as overlapping clusters are merged in the fusion step

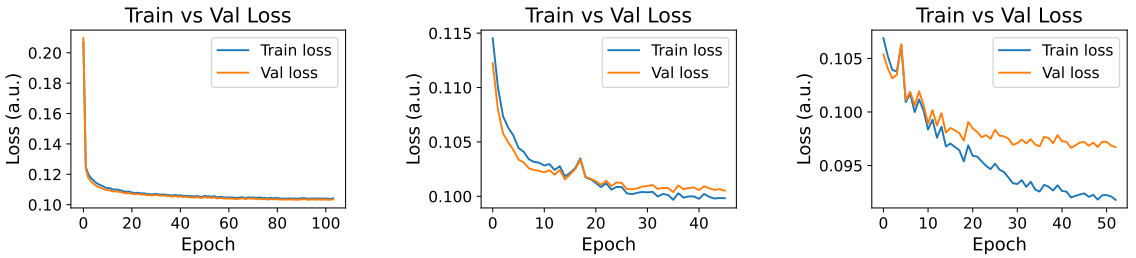
of the clustering pipeline.
An ablation study (see Section 7.1) evaluates the impact of reducing k, showing that the cluster fusion enables the algorithm to keep most of its performance under tighter latency constraints.

Cluster Space Dimension The dimension of the cluster space can be varied but this requires the retraining or fine-tuning of the underlying *CATFinder*-GNN. As this requires a high amount of computing resources, this experiment was skipped in favor of fine-tuning the *CATFinder*-GNN end-to-end together with the *neural clustering* model (see Section 7.5). A proof of concept is implemented in Section 7.3, showing that integrating the *neural clustering* with a *CATFinder*-GNN with higher cluster space dimension is possible, but due to differences in the inputs provided to this *CATFinder*-GNN during its pre-training, the results are not directly comparable. For this thesis, the cluster space dimension is thus fixed at 3 unless explicitly stated otherwise.

5.2.2. Model Size

The training parameters are fitted by training differently sized models on the training dataset and evaluating the validation loss to prevent overfitting.
The model size is defined by the two shared Multi-Layer Perceptron (MLP) sizes, which depend on their hidden sizes h_1 and h_3 as well as the embedding dimension h_2. Three different model configurations are evaluated:

$$\begin{aligned} s : (h_1, h_2, h_3) &= (8, 16, 16) \longrightarrow 0.7\text{k params} \\ m : (h_1, h_2, h_3) &= (32, 64, 64) \longrightarrow 10\text{k params} \\ l : (h_1, h_2, h_3) &= (128, 256, 256) \longrightarrow 165\text{k params}. \end{aligned}$$



(a) Model size: s (b) Model size: m (c) Model size: l

Figure 5.2.: Train and validation loss for different model sizes.

As can be seen in Figure 5.2, the large model overfits on the training data, resulting in a high difference between the training and validation loss. This model size is thus larger than required for the clustering task. The small model does not reach the same low validation loss as the larger models, showing that it does not have the capacity required to consistently find the clusters. All further evaluation is done with the medium sized model, as this shows neither significant overfitting, nor underfitting.

A more detailed study on how small the model could be made without sacrificing prediction performance could further speed up the inference, but lies beyond the scope of this thesis.

5.2.3. Thresholds

The clustering inference requires three distinct thresholds. They are introduced in the *condensation point* selection, neural network output binning and cluster fusion steps.

5.2.3.1. Search Space

Condensation Point Selection *Condensation points* are selected by choosing all points with $\beta > \tau_\beta$. With this cut, *condensation points* belong to the points that are classified as good candidates for cluster seeds by the object condensation loss. Increasing this threshold results in less *condensation points* and thus less predicted clusters. Accordingly, fake clusters are suppressed (track purity increases) but this may also lead to the loss of real clusters (track efficiency decreases).
As β is a `float32` value, its value lies in $\{0\} \cup [\approx 10^{-38}, 1]$. The distribution of predicted β-values is shown in Figure 5.3.

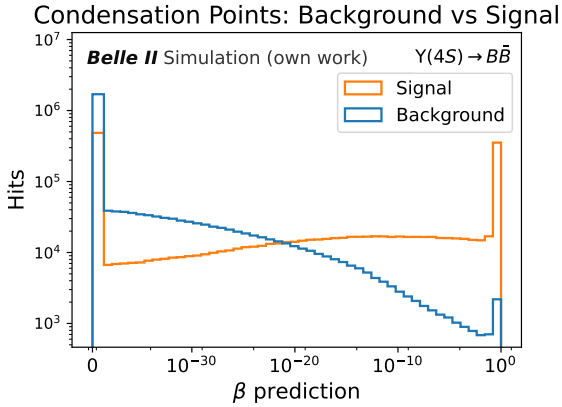


Figure 5.3.: Distribution of β-values in GNN-predictions for signal and background hits on $\Upsilon(4S) \longrightarrow B\bar{B}$ events.

Most signal hits have a predicted β-value of 0 or very close to 1. To validate that hits in the tail in between mostly are not the ideal *condensation point* candidates, values for τ_β are not only sampled close to 1, but from this tail as well. This results in the following set of possible thresholds for the hyperparameter gridsearch:

$$\tau_\beta \in \{10^{-20},\ 10^{-10},\ 0.1,\ 0.2,\ 0.3, \ldots, 0.9\}. \tag{5.1}$$

Neural Network Output Binning In each inference, the cluster model predicts an affinity score for each input point. This score p lies in $[0, 1]$ and stems from the sigmoid activation function. All points with $p > \tau_\sigma$ are labeled as belonging to the predicted cluster. As the activation function is symmetric ($\sigma(-z) = 1 - \sigma(z)$), the threshold $\tau_\sigma = 0.5$ is chosen. Varying this threshold directly influences hit efficiency and hit purity, as a lower τ_σ would

include more hits in each cluster prediction, adding both fake and true hits. For the hyperparameter search, other values for τ_σ are not tested, as this would increase the search space exponentially, while introducing more ways to overfit the hyperparameters to the selection sample. Additionally, τ_σ is strongly dependent on the training of the clustering model, as it directly depends on the prediction confidence. The gridsearch of this thesis focuses on the other parameters, which are less dependent on the actual clustering model. Thus the threshold is set to:

$$\tau_\sigma = 0.5. \tag{5.2}$$

A careful optimization of τ_σ could result in further improvements but is left for future experiments.

Cluster Fusion In the last pipeline step, overlapping clusters are fused, removing duplicates and combining partial cluster predictions into the final clusters. This requires a definition of when two clusters are counted as *overlapping*. As described in Section 4.1, the ratio of the intersection and the union of both clusters is compared with the threshold τ_{iou}. Two clusters A and B are fused if $\text{IOU}(A, B) = \frac{|A \cap B|}{|A \cup B|} > \tau_{\text{iou}}$.
If τ_{iou} is too big, two parts of the same truth cluster may not be combined, resulting in duplicate clusters. If τ_{iou} is too small, two different truth clusters may be combined into one cluster if enough points are predicted to belong to both clusters. Thus τ_{iou} has to be fitted on a validation data set. As $\text{IOU} \in [0, 1]$, τ_{iou} is sampled from $(0, 1)$, resulting in the following set of thresholds for the gridsearch:

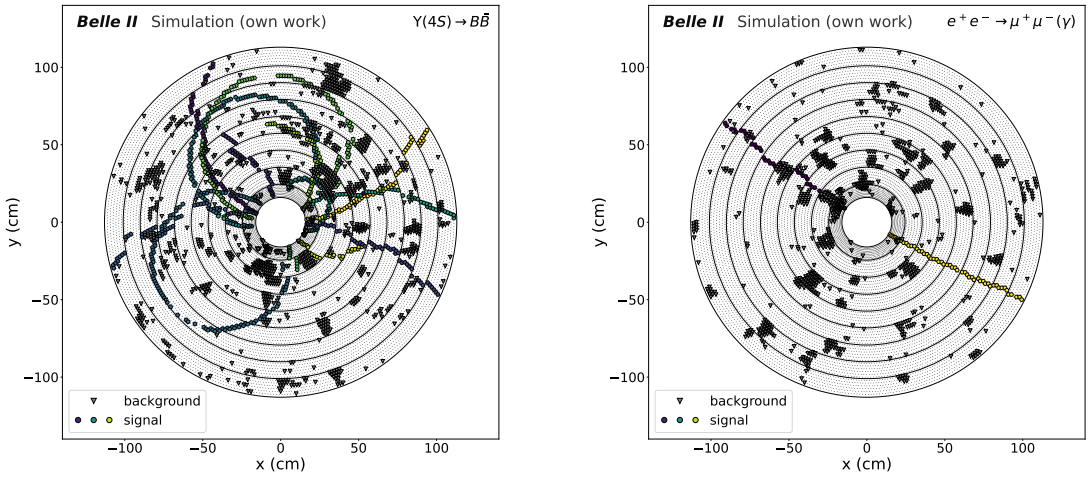
$$\tau_{\text{iou}} \in \{0.05,\ 0.1,\ 0.15, \ldots 0.95\}. \tag{5.3}$$

5.2.3.2. Working Point Selection

The working point is selected by evaluating the performance of each set of thresholds in the full search space grid and picking the parameter set with best performance. This section will show how the performance is assessed, describing the chosen data set and figure of merit.

Data Set A key challenge in selecting the thresholds is ensuring that they generalize across different event topologies and do not overfit on one sample type. Instead of solving this by combining different sample types into one data set, fitting is performed on one specific sample type and validated on other topologies.
For fitting, the $B\bar{B}$ dataset is chosen, which contains all particles from decays of a $B\bar{B}$-pair stemming from an $\Upsilon(4S)$-resonance for 2000 events. These events belong to the most important but still challenging topologies of the physics samples common in Belle II collisions, as they have a comparatively high track multiplicity, contain low-momentum tracks and particles that interact with the detector material. This data set choice is motivated by the intuition that both τ_β and τ_{iou} provide a lower bound for the sensitivity with lower values for both increasing the rate of fake and duplicate cluster predictions as well as the cluster finding efficiency. Thus, the combination of high background levels with hard to find tracks should provide the strongest trade-off between track finding efficiency

and purity. A parameter set fitted on these events should ideally lie in the broader range of optimal values for less complex event topologies. The differences between the two extreme topologies are shown in Figure 5.4.



(a) $B\bar{B}$ event.

(b) $e^+e^- \rightarrow \mu^-\mu^+(\gamma)$ event.

Figure 5.4.: Comparison of the CDC hits from a $B\bar{B}$ (a) and a $e^+e^- \rightarrow \mu^-\mu^+(\gamma)$ (b) decay in the $x-y$-plane for *high data beam background*. Filled circular markers indicate signal hits. Filled gray triangular markers show background hits.

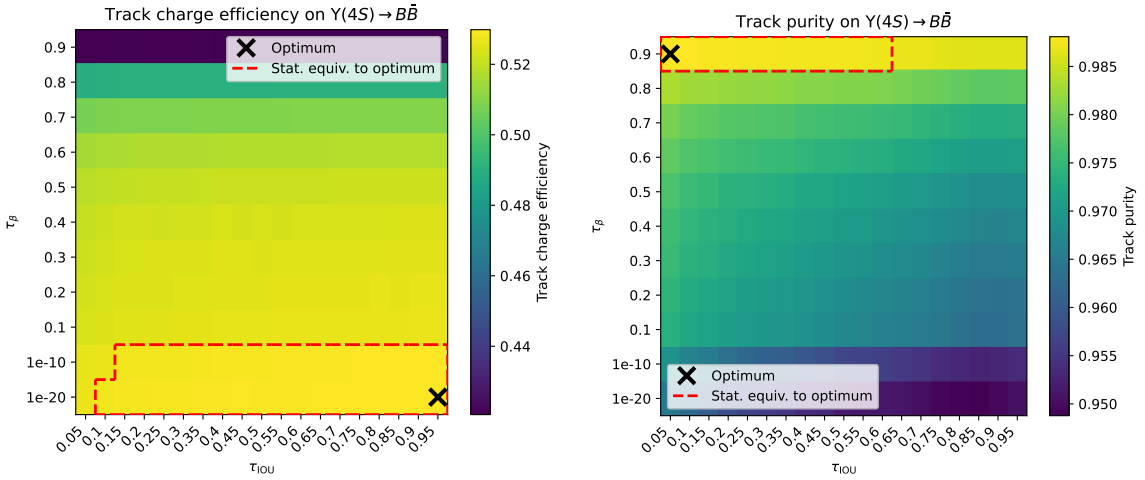
In Section 8.1, the thresholds fitted on this data set are validated on $\mu^\pm$ pairs from the $e^+e^- \rightarrow \mu^-\mu^+(\gamma)$ dataset. The validation experiment shows that the hyperparameters generalize from the high multiplicity B decays with many low-momentum tracks to events containing only a high-energy μ-pair.

Performance Metric The primary metrics for evaluation of the clustering performance are *track charge fitting efficiency* $\varepsilon_{\text{trk,ch}}$ and *track fitting purity* $\mathfrak{p}_{\text{trk}}$, as these are the metrics directly relevant for the physics analysis after track finding (see Section 3.2). Even though track finding metrics would provide a sharper efficiency-purity trade-off, evaluating after fitting is important, as the inclusion of fake hits into true tracks can cause the fitting step to fail, which would not be covered by pre-fitting metrics.
As in [1], both metrics are combined into one figure of merit

$$\mathcal{F} = \varepsilon_{\text{trk,ch}} + \mathfrak{p}_{\text{trk}} \tag{5.4}$$

and the parameter set $(\tau_\beta, \tau_{\text{iou}}) = \arg\max_{\tau_\beta,\, \tau_{\text{iou}}} \mathcal{F}(\tau_\beta, \tau_{\text{iou}})$ is chosen for all further evaluation.

Figure 5.5 shows the direct trade-off between $\varepsilon_{\text{trk,ch}}$ and $\mathfrak{p}_{\text{trk}}$ as a low τ_β includes more tracks, which raises $\varepsilon_{\text{trk,ch}}$ through the inclusion of previously not found tracks at the cost of $\mathfrak{p}_{\text{trk}}$ through fake and clone track predictions. Combining these metrics into $\mathcal{F}$



(a) Track charge efficiency gridsearch.

(b) Track purity gridsearch.

Figure 5.5.: *Track charge efficiency* and *track purity* for all different combinations of τ_β and τ_{iou} in the gridsearch evaluated on $\Upsilon(4S) \to B\bar{B}$. The dashed line marks the region of threshold combinations with performance within statistical uncertainty of the optimum.

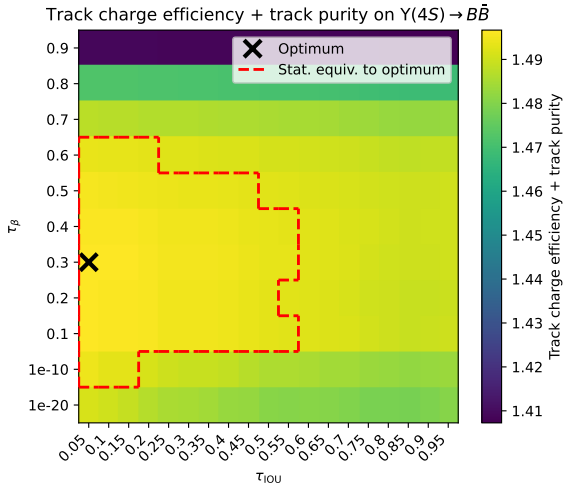


Figure 5.6.: Figure of merit after fitting for all threshold combinations in the gridsearch evaluated on $\Upsilon(4S) \to B\bar{B}$. The dashed line marks the region of threshold combinations with performance within statistical uncertainty of the optimum.

suggests an intermediate value for the fitted thresholds. As can be seen in Figure 5.6, the best performing set of thresholds is $\tau_\beta = 0.3$, $\tau_{\text{iou}} = 0.05$, with a large part of the parameter space performing comparably. These parameters will be used for all further evaluation, if not explicitly stated.
A comparison to the thresholds attained on the dimuon sample can be found in Section 8.1.

6. Results

This chapter evaluates the track finding performance of the *neural clustering* approach applied on the *CATFinder*-GNN. The *neural clustering* is compared with the *spherical clustering* approach with both approaches relying on the same *CATFinder*-GNN, trained on a dataset combining technical $\mu^{\pm}$ tracks with $\Upsilon(4S) \rightarrow B\bar{B}$ decays. Additionally, the currently deployed track finding algorithm [5] is included in the comparison as a *Baseline*.
First, the performance on simple event topologies is evaluated to ensure that the optimization on complex topologies does not reduce performance on the samples where the current clustering algorithm already performs well. Next, the performance on challenging signatures outlined in Subsection 2.2.3 is evaluated. Finally, the computational effort of the *neural clustering* is compared to a clustering algorithm with similar performance.
Results for *track charge efficiency*, *fake rate* and *clone rate* are shown before and after the track fitting step. Metrics before fitting show how many true and fake tracks are found by the clustering. The track fitting rejects track predictions which do not form a coherent track, these can either be total fake tracks or true tracks with too few assigned signal hits (low *hit efficiency*) or with too many assigned background hits (low *hit purity*) or track-like background features stemming from cross-talk or interactions with the detector material. This results in the fitting step deflating *fake rate* and *clone rate*, while restricting *track charge efficiency* to the tracks that are provided as final output. Thus, from a physics analysis point-of-view, the metrics after fitting are the most important. While a high *fake rate* before fitting is not problematic in itself, it results in more computational effort for the fitting procedure, as fake tracks are fitted and rejected if necessary.

6.1. Standard Signatures

A large part of the events recorded by the Belle II experiment consist of two oppositely charged particles with both tracks starting at the Interaction Point (IP). These prompt tracks are (especially at high transverse momentum) found consistently by both the *Baseline* and the *CATFinder with spherical clustering*. To validate that the proposed clustering does not improve on challenging signatures by sacrificing performance on prompt tracks, this section evaluates the performance on two less challenging event topologies.
First, a physics-based dataset containing $e^+e^- \rightarrow \mu^-\mu^+(\gamma)$ events is evaluated, as these events are used in the calibration of Belle II. Secondly, a technical prompt sample containing $1-3$ μ tracks per event with a higher focus on low-momentum tracks is evaluated.

6.1.1. High Transverse Momentum

In the $e^+e^- \rightarrow \mu^-\mu^+(\gamma)$ sample, almost all events contain two prompt tracks with high momentum. These samples are used in the Belle II calibration, thus a maximum *track charge efficiency* in the barrel region is important.

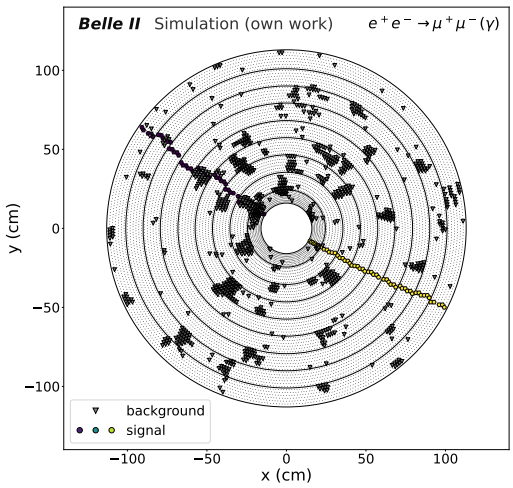
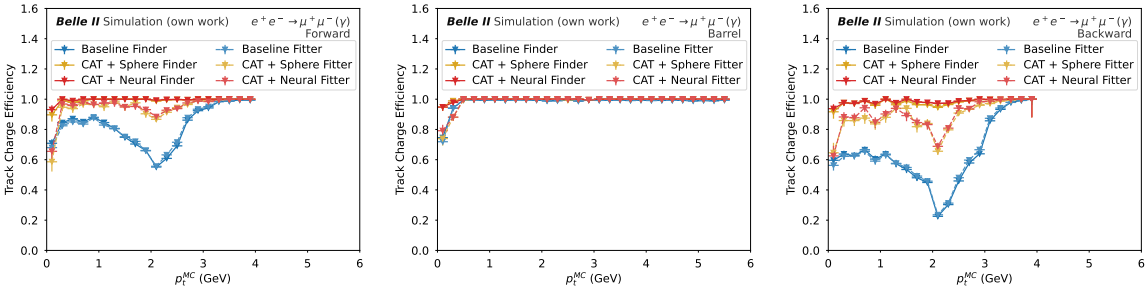


Figure 6.1.: Hits from an $e^+e^- \rightarrow \mu^-\mu^+(\gamma)$ decay in the $x-y$-plane for *high data beam background*. Colored circular markers indicate signal hits. Filled gray triangular markers show background hits.

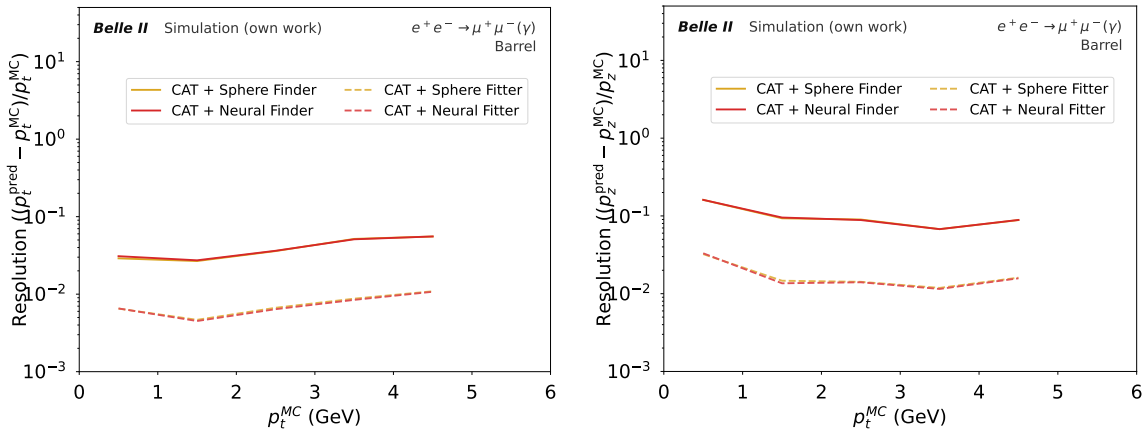
Figure 6.2 shows the *track charge efficiency* before and after fitting for the *Baseline*, *CATFinder with spherical clustering* and *CATFinder with neural clustering* algorithms. The *neural clustering* does not significantly change *track charge efficiency* across all detector regions compared to the *spherical clustering*. Both *CATFinder* variants provide a significantly higher *efficiency* than the *Baseline* algorithm.
The strong decrease in *efficiency* for low transverse momentum is attributed to the fact that particles with $p_t < 0.255\,\text{GeV}$ do not reach the end of the Central Drift Chamber (CDC), resulting in tracks which leave several overlapping loops inside the CDC. These tracks are harder to fit, as a successful fit requires the hits to be ordered according to which loop they belong to. Prompt tracks with $p_t < 0.039\,\text{GeV}$ do not reach the CDC and cannot be found by CDC track finding algorithms, instead they are found in the Silicon Vertex Detector (SVD) and are thus not taken into account for this evaluation. The performance on tracks with low transverse momentum is evaluated in Subsection 6.1.2.
A detailed overview of all finding metrics integrated over the whole momentum range is provided in Table 6.1 and shows that the *neural clustering* performs similar or better in *track efficiency* and *clone rate* in all detector regions, at the cost of an increase in *fake rate*. These differences are similar before and after fitting.



(a) Forward endcap. (b) Barrel. (c) Backward endcap.

Figure 6.2.: *Track charge efficiency* before (solid lines) and after (dashed lines) track fitting for $\mu^\pm$ in $e^+e^- \to \mu^-\mu^+(\gamma)$ samples with *high data beam background* for different algorithms and detector regions. Horizontal bars indicate bin width, vertical error bars indicate uncertainties (correlated between algorithms).

Figure 6.3 shows that in the barrel region, the post-fitting momentum resolution does not differ significantly between the *spherical* and *neural clustering*. The hit metrics shown in Figure 6.4 indicate that the *neural clustering* finds more hits for each track, resulting in a higher *hit efficiency*. This comes at the cost of a slight decrease in *hit purity* for low-momentum tracks, as more fake hits are added in these cases. These differences do not significantly impact the post-fitting momentum resolution, but they result in more tracks being successfully fitted, increasing post-fitting *track charge efficiency*.



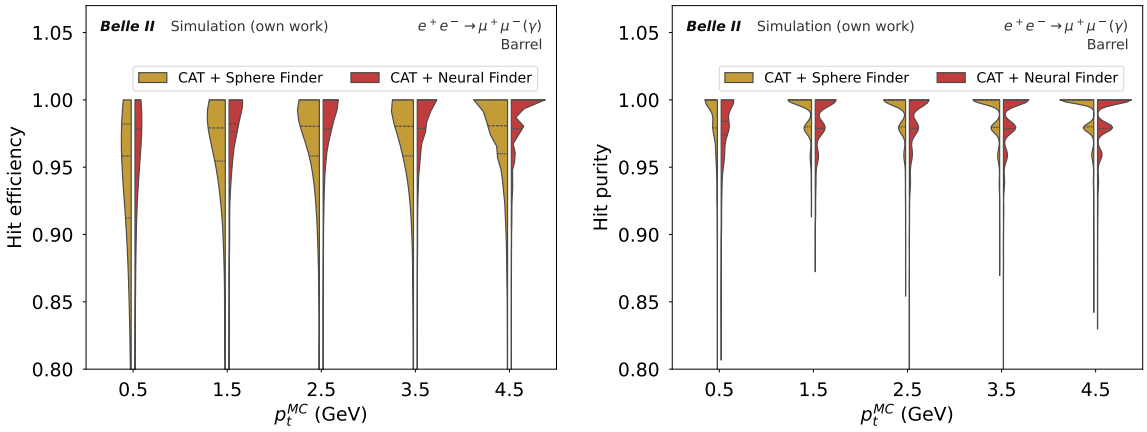
(a) Transverse momentum resolution. (b) Axial momentum resolution.

Figure 6.3.: Momentum resolution before (solid lines) and after (dashed lines) track fitting for $\mu^\pm$ in $e^+e^- \to \mu^-\mu^+(\gamma)$ samples with *high data beam background* for different algorithms limited to the intersecting sample in the barrel region.

Importantly, the post-fitting performance in the barrel region does not degrade with the introduction of the *neural clustering*.

Table 6.1.: The performance metrics for $\mu^{\pm}$ in $e^+e^- \to \mu^-\mu^+(\gamma)$ samples for different track finding algorithms in different detector regions for *high data beam background*. Uncertainties below 0.01% are not shown in the table, uncertainties are correlated between different methods.

Method	$\varepsilon_{\text{trk}}[\%]$	$\mathfrak{r}_{\text{fake}}[\%]$	$\mathfrak{r}_{\text{clone}}[\%]$	$\varepsilon_{\text{trk,ch}}[\%]$	$\mathfrak{r}_{\text{wrong ch.}}[\%]$
			FWD		
Baseline Finder	$85.7^{+0.2}_{-0.2}$	$1.58^{+0.08}_{-0.08}$	$0.01_{-0.01}$	$83.4^{+0.2}_{-0.2}$	$2.6^{+0.1}_{-0.1}$
CAT + Sphere Finder	$99.68^{+0.08}_{-0.07}$	$9.0^{+0.4}_{-0.4}$	$0.59^{+0.1}_{-0.11}$	$99.62^{+0.09}_{-0.08}$	$0.06^{+0.03}_{-0.04}$
CAT + Neural Finder	$99.85^{+0.06}_{-0.05}$	$10.6^{+0.4}_{-0.4}$	$0.21^{+0.06}_{-0.07}$	$99.77^{+0.07}_{-0.06}$	$0.08^{+0.03}_{-0.05}$
Baseline Fitter	$85.2^{+0.2}_{-0.2}$	$1.55^{+0.08}_{-0.08}$	$0.0_{-0.01}$	$84.1^{+0.2}_{-0.2}$	$1.25^{+0.07}_{-0.08}$
CAT + Sphere Fitter	$98.6^{+0.2}_{-0.2}$	$2.3^{+0.2}_{-0.2}$	$0.33^{+0.07}_{-0.09}$	$95.9^{+0.3}_{-0.3}$	$2.8^{+0.2}_{-0.2}$
CAT + Neural Fitter	$98.9^{+0.2}_{-0.1}$	$2.6^{+0.2}_{-0.2}$	$0.08^{+0.03}_{-0.05}$	$96.3^{+0.3}_{-0.3}$	$2.6^{+0.2}_{-0.2}$
			BRL		
Baseline Finder	$99.15^{+0.03}_{-0.03}$	$2.77^{+0.05}_{-0.05}$	$0.03^{+0.01}_{-0.01}$	$99.06^{+0.03}_{-0.03}$	$0.1^{+0.01}_{-0.01}$
CAT + Sphere Finder	$99.99^{+0.01}_{-0.01}$	$9.5^{+0.2}_{-0.2}$	$0.21^{+0.03}_{-0.03}$	$99.95^{+0.02}_{-0.01}$	$0.03^{+0.01}_{-0.01}$
CAT + Neural Finder	$99.99^{+0.01}_{-0.01}$	$10.9^{+0.2}_{-0.2}$	$0.06^{+0.02}_{-0.02}$	$99.95^{+0.02}_{-0.01}$	$0.04^{+0.01}_{-0.02}$
Baseline Fitter	$99.11^{+0.03}_{-0.03}$	$2.2^{+0.04}_{-0.05}$	0.02	$99.05^{+0.03}_{-0.03}$	$0.06^{+0.01}_{-0.01}$
CAT + Sphere Fitter	$99.78^{+0.03}_{-0.03}$	$2.4^{+0.1}_{-0.1}$	$0.11^{+0.02}_{-0.02}$	$99.74^{+0.04}_{-0.03}$	$0.04^{+0.01}_{-0.02}$
CAT + Neural Fitter	$99.81^{+0.03}_{-0.03}$	$2.7^{+0.1}_{-0.1}$	$0.03^{+0.01}_{-0.01}$	$99.79^{+0.03}_{-0.03}$	$0.02^{+0.01}_{-0.01}$
			BWD		
Baseline Finder	$68.9^{+0.3}_{-0.3}$	$3.5^{+0.1}_{-0.2}$	$0.02^{+0.01}_{-0.01}$	$64.5^{+0.3}_{-0.3}$	$6.4^{+0.2}_{-0.2}$
CAT + Sphere Finder	$98.8^{+0.2}_{-0.1}$	$13.8^{+0.5}_{-0.5}$	$0.3^{+0.07}_{-0.09}$	$98.4^{+0.2}_{-0.2}$	$0.39^{+0.08}_{-0.1}$
CAT + Neural Finder	$99.4^{+0.1}_{-0.1}$	$19.0^{+0.5}_{-0.5}$	$0.14^{+0.05}_{-0.06}$	$98.9^{+0.2}_{-0.1}$	$0.44^{+0.09}_{-0.1}$
Baseline Fitter	$68.3^{+0.3}_{-0.3}$	$3.2^{+0.1}_{-0.1}$	$0.01^{+0.01}_{-0.01}$	$65.6^{+0.3}_{-0.3}$	$4.0^{+0.1}_{-0.2}$
CAT + Sphere Fitter	$96.0^{+0.3}_{-0.3}$	$4.9^{+0.3}_{-0.3}$	$0.15^{+0.05}_{-0.06}$	$90.5^{+0.4}_{-0.4}$	$5.8^{+0.3}_{-0.3}$
CAT + Neural Fitter	$97.3^{+0.2}_{-0.2}$	$6.1^{+0.3}_{-0.4}$	$0.12^{+0.04}_{-0.06}$	$91.9^{+0.4}_{-0.4}$	$5.5^{+0.3}_{-0.3}$



(a) *Hit efficiency.*

(b) *Hit purity.*

Figure 6.4.: *Hit efficiency* and *purity* before track fitting for $\mu^{\pm}$ in $e^+e^- \to \mu^-\mu^+(\gamma)$ samples with *high data beam background* for different algorithms limited to the intersecting sample in the barrel region.

6.1.2. Low Transverse Momentum

As indicated in the $e^+e^- \rightarrow \mu^-\mu^+(\gamma)$ evaluation (see Subsection 6.1.1), tracks with low transverse momentum pose a challenge for track finding, as they contain either very few hits (particles with $p_t \in [0.039\,\text{GeV}, 0.055\,\text{GeV}]$ only reach the first superlayer) or a very high number of hits in overlapping loops (particles with $p_t \in [0.039\,\text{GeV}, 0.255\,\text{GeV}]$ do not reach the end of the CDC).
To evaluate performance on these samples, a set of events containing single technical $\mu^\pm$ tracks with low momentum is used.

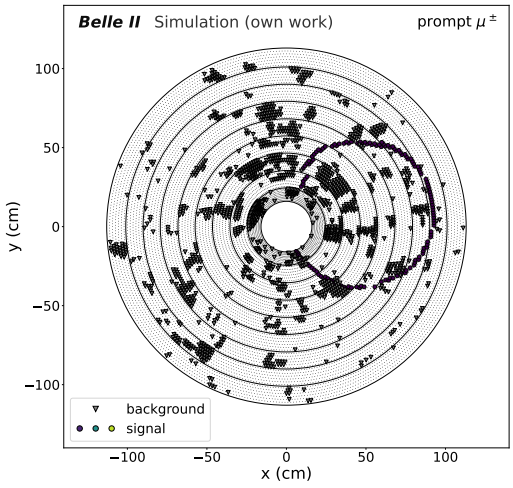
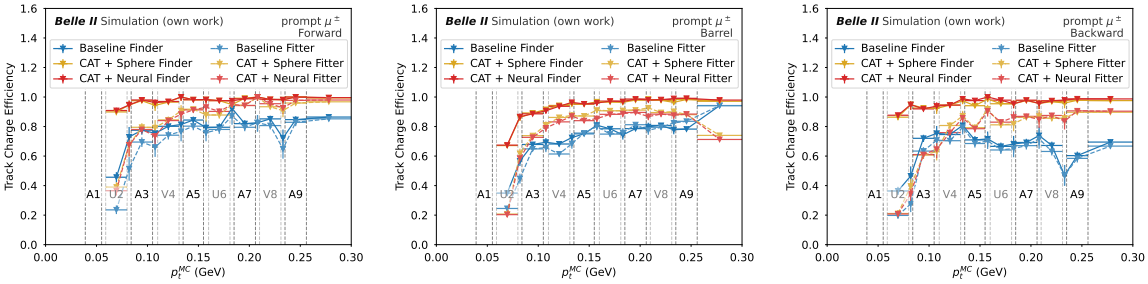


Figure 6.5.: $\mu^\pm$ with low p_t event display in the $x-y$-plane for *high data beam background*. Colored circular markers indicate signal hits. Filled gray triangular markers show background hits.

Figure 6.6 shows that *track charge efficiency* is similar for *spherical* and *neural clustering*. The *neural clustering* improves performance on the short tracks left by particles leaving the CDC in the endcaps, while reducing performance on the tracks in the barrel region. Tracks with $p_t \in [0.255\,\text{GeV},\ 0.3\,\text{GeV}]$ reach the outer edge of the CDC and reenter the CDC after loosing a significant amount of energy in the material between the CDC and the Electromagnetic Calorimeter (ECL) . These outer-curlers are hard to fit as a complete track, as the current fitting algorithm does not take the energy loss in the ECL into account. The *Baseline* algorithm only finds the first section of these tracks, resulting in a fittable track prediction. In contrast, the *CATFinder* algorithms also assign the outer loops, which results in an unstable fit, reducing the *fitting efficiency*, while the *finding efficiency* is higher than the *Baseline* performance.
All regions show a large difference between pre- and post-fitting *efficiency* for the *CATFinder* algorithms. The fitting algorithm requires the hits in a track to be ordered, which is not handled stably by the current implementation of the *CATFinder* algorithm for tracks which curl inside the tracking volume, resulting in overlapping loops. This

results in post-fitting *efficiencies* lower than the *Baseline* performance, even though the pre-fitting performance is significantly better. With the higher *hit efficiency* (see Figure 6.7), *neural clustering* includes even more hits from multiple loops per curler, resulting in more difficulties for hit ordering and thus lower *fitting efficiency* in the barrel region even though the *finding efficiency* improves.
The detailed integrated metrics on this sample are found in Table A.1, showing a similar trend as in Subsection 6.1.1 with an increase in *fake rate* while reducing *clone rate* when employing the *neural clustering*. Tracks leaving the CDC through the endcaps mostly reach the end of the CDC before completing more than one loop thus the fitting performance is less strongly affected for these cases.



(a) Forward endcap. (b) Barrel. (c) Backward endcap.

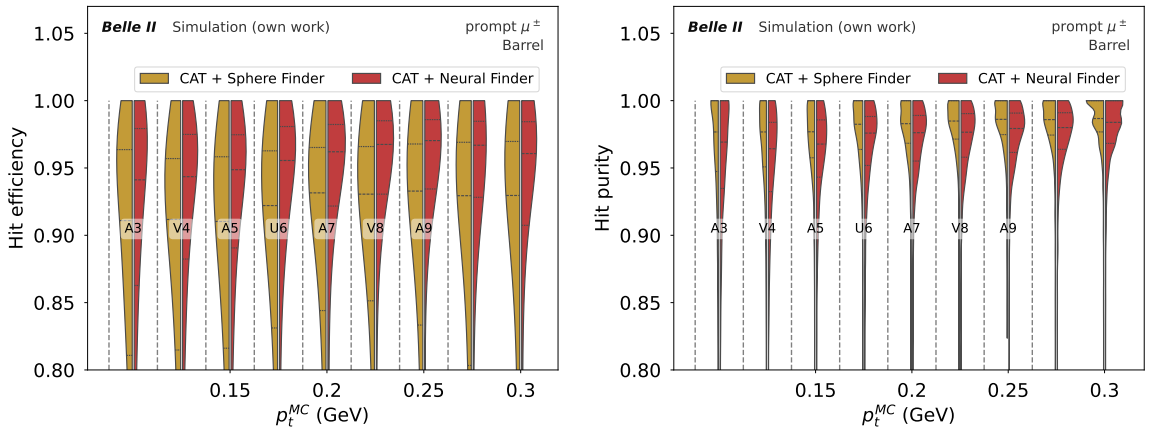
Figure 6.6.: *Track charge efficiency* before (solid lines) and after (dashed lines) track fitting in $\mu^{\pm}$ with low p_t samples with *high data beam background* for different algorithms and detector regions. Horizontal bars indicate bin width, vertical error bars indicate uncertainties (correlated between algorithms). Bins are aligned to the outermost superlayer reached by a particle with given momentum.

Table 6.2.: The performance metrics for $\mu^{\pm}$ with low p_t samples for different track finding algorithms with *high data beam background*. Uncertainties below 0.01% are not shown in the table, uncertainties are correlated between different methods.

Algorithm	$\varepsilon_{\text{trk,ch}}$	$\mathfrak{p}_{\text{trk}}$
Baseline Finder	$80.97^{+0.29}_{-0.28}$	$91.58^{+0.22}_{-0.21}$
CAT + Sphere Finder	$97.81^{+0.11}_{-0.10}$	$78.53^{+0.27}_{-0.27}$
CAT + Neural Finder	$98.19^{+0.10}_{-0.10}$	$76.27^{+0.27}_{-0.27}$
Baseline Fitter	$75.80^{+0.31}_{-0.31}$	$93.11^{+0.21}_{-0.20}$
CAT + Sphere Fitter	$79.64^{+0.29}_{-0.29}$	$94.14^{+0.19}_{-0.18}$
CAT + Neural Fitter	$78.44^{+0.30}_{-0.30}$	$94.18^{+0.19}_{-0.18}$

Similar to the performance on $e^+e^- \to \mu^-\mu^+(\gamma)$ samples, on $\mu^{\pm}$ with low p_t samples, an increase in *hit efficiency* at the cost of *hit purity* is observed in Figure 6.7. This does not affect the momentum resolution shown in Figure A.1, indicating that the decreased *hit purity* does not reduce the momentum resolution. Instead, it increases the rate of unfittable track predictions, as the *neural clustering* outperforms the *spherical clustering* in

pre-fitting *efficiency* while underperforming after fitting.



(a) Hit efficiency.

(b) Hit purity.

Figure 6.7.: *Hit efficiency* and *purity* before track fitting for $\mu^\pm$ with low p_t samples with *high data beam background* for different algorithms limited to the intersecting sample in the barrel region.

6.2. Challenging Signatures

While the *CATFinder* provides a significant improvement upon the currently used track finding algorithm, there are event topologies on which both approaches struggle to achieve a high *efficiency*. This section evaluates the improvements on these challenging signatures gained by combining the *CATFinder* with the *neural clustering* algorithm.
As described in Subsection 2.2.3, displaced decays pose a challenge to current track finding algorithms, as they often result in tracks with few hits or in overlapping tracks if the opening angle is small. The *Baseline* algorithm is currently not able to reliably find these displaced tracks. Complementary, the *CATFinder* with *spherical clustering* struggles with events that contain a high number of tracks.
This section thus evaluates the performance on displaced decays in low and high multiplicity events separately. First, a technical sample of displaced $\mu^\pm$ tracks is evaluated, which does not contain additional tracks. Then, a sample of displaced tracks from B-events is evaluated, which contains significantly more background tracks. Next, the performance on low-momentum prompt tracks from B-events is evaluated, as these also provide challenges for the current tracking algorithms. Finally, the performance on a physics sample with non-displaced tracks but low opening angle between tracks is studied on $e^+e^- \longrightarrow \pi^+\pi^-\gamma$ events.

6.2.1. Low Multiplicity Displaced

In order to separate the influence of decay vertex displacement from other factors, the technical sample of displaced $\mu^{\pm}$ tracks is chosen. A detailed description of this sample is provided in Section 3.1. Tracks with high displacement only leave few hits before they leave the CDC, which complicates track finding. Tracks with low opening angle lie close together, making it difficult to correctly assign the hits to their track. In the following section, both effects are studied.
Figure 6.9 provides an overview of the effect of vertex displacement on the *finding efficiency*. The *CATFinder* already improves significantly upon the *Baseline* performance. An additional improvement on the tracks with very high displacement or extreme opening angle is provided by implementing the *neural clustering*. Both low and high opening angles pose a challenge to the track finding, as low opening angles result in overlapping tracks, while two tracks with high opening angle are hard to distinguish from one curved track. In both cases, the *neural clustering* significantly improves upon the *spherical clustering*.
Table 6.3 contains the metrics integrated over the whole displacement range. It shows that the *neural clustering* increases the *track charge efficiency* both before and after fitting, albeit at the cost of reducing *track purity* but after fitting the *track purity* still exceeds the performance of the *Baseline* algorithm.
The *hit efficiency* and *hit purity* for different displacement regimes are shown in Figure A.3. The *neural clustering* significantly increases *hit efficiency* for tracks with high displacement but does so at the cost of *hit purity*. This decrease in *hit purity* does not reach a scale where it poses a problem to the track fitting as can be seen in the unchanged momentum resolution shown in Figure A.2 and in the fact that the increase in $\varepsilon_{\text{trk,ch}}$ is preserved after fitting.

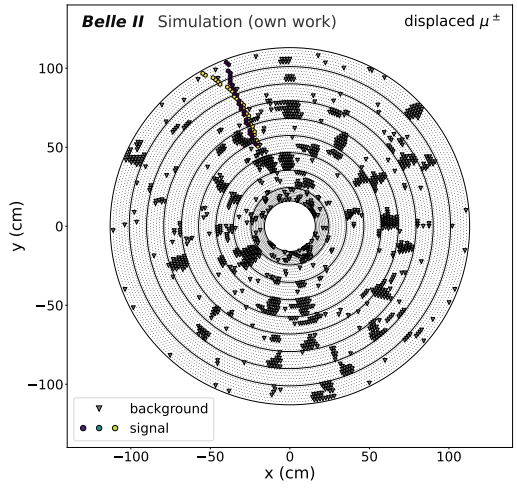
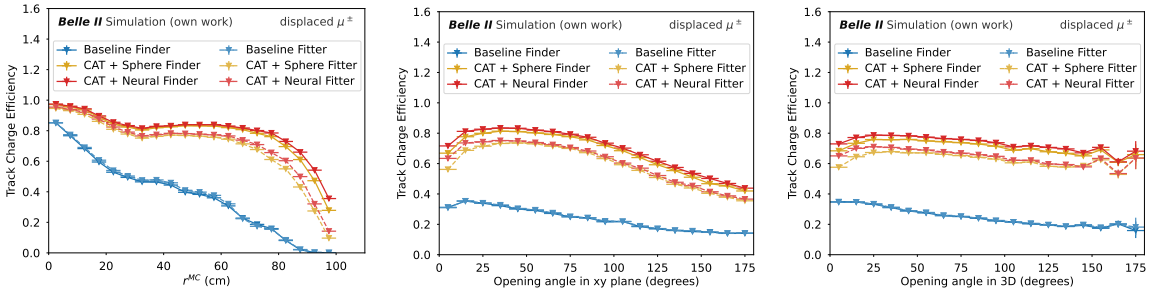


Figure 6.8.: Event display of displaced $\mu^{\pm}$in the $x-y$-plane for *high data beam background*. Colored circular markers indicate signal hits. Filled gray triangular markers show background hits.



(a) By displacement. (b) By radial opening angle. (c) By 3D opening angle.

Figure 6.9.: *Track charge efficiency* before (solid lines) and after (dashed lines) track fitting for $\mu^{\pm}$ in displaced $\mu^{\pm}$ samples with *high data beam background* for different algorithms. Horizontal bars indicate bin width, vertical error bars indicate uncertainties (correlated between algorithms).

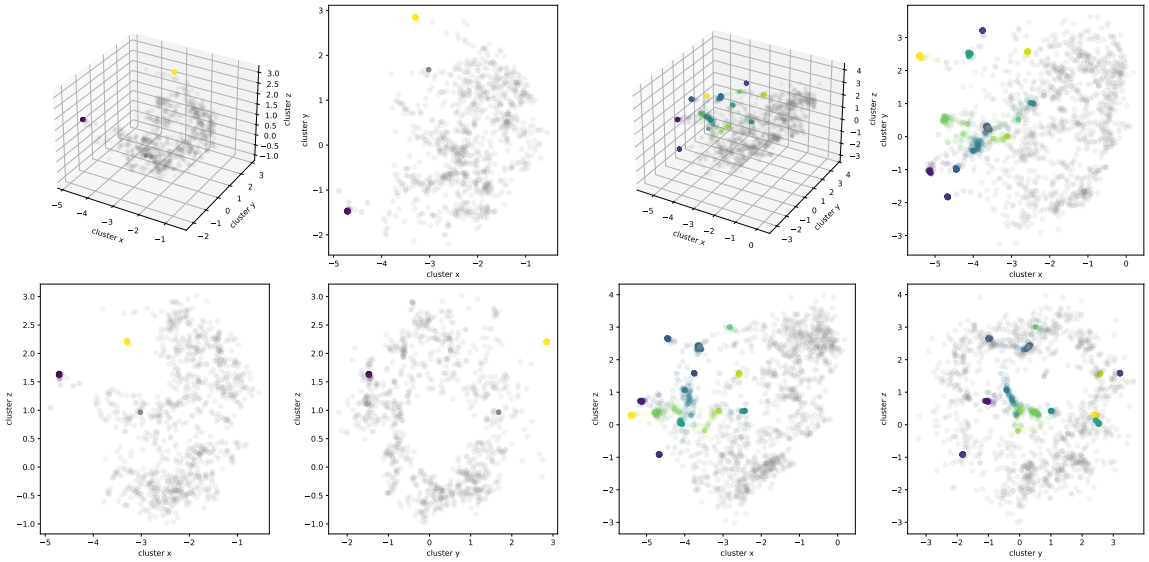
Table 6.3.: The performance metrics for displaced $\mu^{\pm}$ samples for different track finding algorithms *high data beam background*. Uncertainties below 0.01% are not shown in the table, uncertainties are correlated between different methods.

Algorithm	$\varepsilon_{\text{trk,ch}}$	$\mathfrak{p}_{\text{trk}}$
Baseline Finder	$48.84^{+0.07}_{-0.07}$	$96.59^{+0.04}_{-0.04}$
CAT + Sphere Finder	$87.27^{+0.05}_{-0.05}$	$94.16^{+0.04}_{-0.04}$
CAT + Neural Finder	$89.15^{+0.05}_{-0.05}$	$91.61^{+0.04}_{-0.04}$
Baseline Fitter	$48.05^{+0.07}_{-0.07}$	$97.15^{+0.04}_{-0.03}$
CAT + Sphere Fitter	$80.20^{+0.06}_{-0.06}$	$98.43^{+0.02}_{-0.02}$
CAT + Neural Fitter	$82.59^{+0.06}_{-0.06}$	$97.32^{+0.03}_{-0.03}$

6.2.2. High Multiplicity Displaced

As one of the main research areas of the Belle II experiment is B-physics, performance on the event topologies created by B-meson decays is important. Events containing B-mesons result on average in 11 tracks in the CDC. As many of these tracks have a low transverse momentum, many signal hits are left in the CDC, with tracks overlapping in the $x-y$ view provided by the CDC sense wires.
This results in a challenging event topology for track finding, especially if combined with the displaced tracks from Subsection 6.2.1. In the following section, the performance on displaced tracks in B-events is evaluated. To gain enough tracks with high displacement, B decays are simulated, where one of the B-mesons decays into a Long Lived Particle (LLP) S with a mass of 0.22 GeV, which decays into an electron-positron pair. The lifetime of S is varied statistically to get electron pairs at all levels of displacement. The high mass of the LLP results in a high boost of the e^+e^- pair and thus a small opening angle between their tracks.

Similar to the analysis for displaced $\mu^{\pm}$ tracks, Figure 6.12 shows that the *neural clustering* improves *track charge efficiency* both before and after fitting on tracks with high



(a) The latent space with the hits from the $e^+e^- \rightarrow \mu^-\mu^+(\gamma)$ event in Figure 6.1. Colored markers represent the latent space coordinates corresponding to detector hits left by simulated particles, gray markers represent background hits. The marker transparency is scaled by β.

(b) The latent space with the hits from the $\Upsilon(4S) \rightarrow B\bar{B}$ event in Figure 6.13. Colored markers represent the latent space coordinates corresponding to detector hits left by simulated particles, gray markers represent background hits. The marker transparency is scaled by β.

Figure 6.10.: Latent space of the *CATFinder* GNN for an $e^+e^- \rightarrow \mu^-\mu^+(\gamma)$ event with low (Figure 6.10a) and an $\Upsilon(4S) \rightarrow B\bar{B}$ event with high (Figure 6.10b) track multiplicity.

displacement and across all opening angles. The improvement is more significant in the high-multiplicity scenario of B-decays, compared to the $\mu^\pm$ events, because clearly separating all tracks in the latent space is harder if more tracks are to be separated. This results in less clearly separated clusters, where a more sophisticated clustering algorithm significantly improves performance.
Both *CATFinder* approaches find tracks with very high opening angle, which fail the fitting step. Due to the small mass of the e^+e^- pair, high opening angles require a very low momentum of the mother particle. In this case, the LLP decays close the the IP even for large lifetimes, resulting in daughter tracks that just barely enter the CDC. These tracks contain very few signal hits, and are thus hard to fit without information from the SVD.
The integrated metrics are shown in Table 6.4. The *neural clustering* improves *track charge fitting efficiency* by several percentage points, significantly increasing the sensitivity. *Purity* is low for all algorithms, as on events with very low or very high opening angle, both tracks are combined into one prediction, which fails the matching requirements. Additionally, found tracks may have only very few assigned hits due to displacement and clustering inefficiencies, failing the minimum hit count requirement. This results in a comparatively large number of fake tracks.

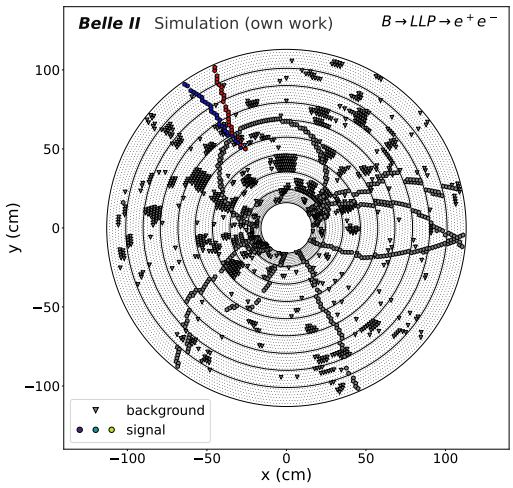
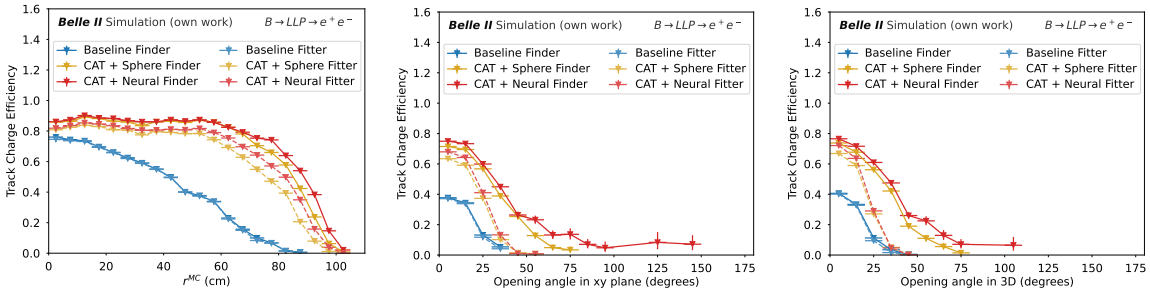


Figure 6.11.: $B \longrightarrow LLP \longrightarrow e^+e^-$ event display in the $x-y$-plane for *high data beam background*. Colored circular markers indicate signal hits from the e^+e^--pair. Gray circular markers indicate signal hits from other particles in the decay. Filled gray triangular markers show background hits.



(a) By displacement. (b) By radial opening angle. (c) By 3D opening angle.

Figure 6.12.: *Track charge efficiency* before (solid lines) and after (dashed lines) track fitting in $B \longrightarrow LLP \longrightarrow e^+e^-$ samples with *high data beam background* for different algorithms. Horizontal bars indicate bin width, vertical error bars indicate uncertainties (correlated between algorithms).

Figure A.4 shows the momentum resolution by vertex displacement before and after fitting. Even though *hit efficiency* is increased at the expense of *hit purity* as seen in Figure A.5, the momentum resolution on the tracks found by both *CATFinder* algorithms is not affected. Thus, the *neural clustering* results in more found tracks, without sacrificing fitting performance on the tracks already found with *spherical clustering*.

Table 6.4.: The performance metrics for $B \longrightarrow LLP \longrightarrow e^+e^-$ samples for different track finding algorithms *high data beam background*. Uncertainties below 0.01% are not shown in the table, uncertainties are correlated between different methods.

Algorithm	$\varepsilon_{\text{trk,ch}}$	$\mathfrak{p}_{\text{trk}}$
Baseline Finder	$48.83^{+0.25}_{-0.25}$	$57.65^{+0.26}_{-0.26}$
CAT + Sphere Finder	$79.09^{+0.20}_{-0.20}$	$49.53^{+0.19}_{-0.19}$
CAT + Neural Finder	$81.52^{+0.19}_{-0.19}$	$47.44^{+0.19}_{-0.19}$
Baseline Fitter	$47.93^{+0.25}_{-0.25}$	$59.97^{+0.27}_{-0.27}$
CAT + Sphere Fitter	$70.49^{+0.22}_{-0.22}$	$61.26^{+0.22}_{-0.22}$
CAT + Neural Fitter	$73.98^{+0.22}_{-0.21}$	$60.16^{+0.22}_{-0.22}$

6.2.3. High Multiplicity Low Momentum

While displaced tracks are especially challenging for track finding algorithms and important for many hypothetical dark sector decays, decays with extremely high displacement are not very frequent. For this reason, the mostly low-momentum $\pi^{\pm}$ from B-meson decays are evaluated to asses the performance of the *neural clustering* in a comparatively common track signature. These tracks are challenging for track finding, as they form loops inside the CDC, which results in many overlapping tracks in the high track-multiplicity of B decays. Thus, this is a more difficult sibling of the tracks studied in Subsection 6.1.2.

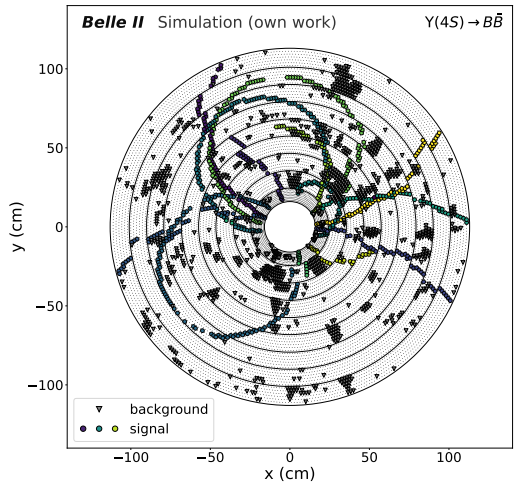
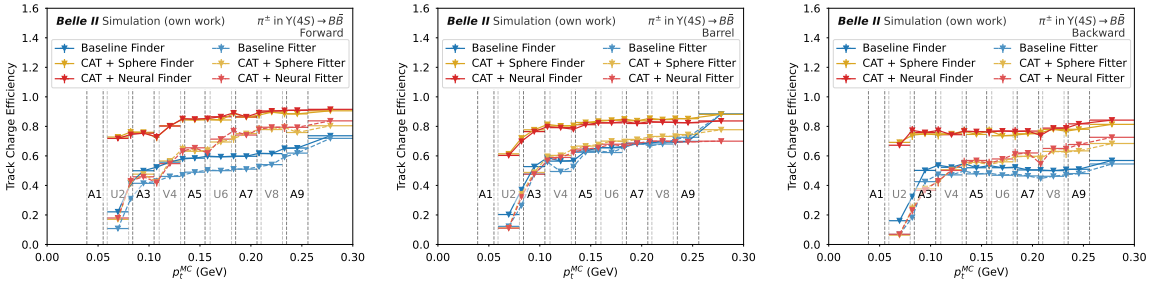


Figure 6.13.: $\Upsilon(4S) \longrightarrow B\bar{B}$ event display in the $x-y$-plane for *high data beam background*. Colored circular markers indicate signal hits. Filled gray triangular markers show background hits.

The *track charge efficiency* by transverse momentum shown in Figure 6.14 is similar to the results on $\mu^{\pm}$ with low p_t. The *neural clustering* improves *efficiency* in the endcap regions at the cost of *efficiency* in the barrel region, with large differences between pre-



(a) Forward endcap. (b) Barrel. (c) Backward endcap.

Figure 6.14.: *Track charge efficiency* before (solid lines) and after (dashed lines) track fitting for $\pi^{\pm}$ in $\Upsilon(4S) \longrightarrow B\bar{B}$ samples with *high data beam background* for different algorithms and detector regions. Horizontal bars indicate bin width, vertical error bars indicate uncertainties (correlated between algorithms).

Table 6.5.: The performance metrics for $\Upsilon(4S) \longrightarrow B\bar{B}$ samples for different track finding algorithms for *high data beam background*. Uncertainties below 0.01% are not shown in the table, uncertainties are correlated between different methods.

Algorithm	$\varepsilon_{\text{trk,ch}}$	$\mathfrak{p}_{\text{trk}}$
Baseline Finder	$78.21^{+0.05}_{-0.05}$	$83.27^{+0.04}_{-0.04}$
CAT + Sphere Finder	$88.94^{+0.08}_{-0.08}$	$70.30^{+0.11}_{-0.11}$
CAT + Neural Finder	$88.10^{+0.08}_{-0.08}$	$69.25^{+0.11}_{-0.11}$
Baseline Fitter	$75.59^{+0.05}_{-0.05}$	$84.67^{+0.04}_{-0.04}$
CAT + Sphere Fitter	$77.99^{+0.11}_{-0.11}$	$80.67^{+0.10}_{-0.10}$
CAT + Neural Fitter	$77.03^{+0.11}_{-0.11}$	$79.69^{+0.11}_{-0.11}$

and post-fitting *efficiency* due to the hit ordering. The comparatively high *track charge efficiency* of the *Baseline* algorithm in the $p_t \in [0.255\,\text{GeV},\ 0.3\,\text{GeV}]$ bin is due to the fact that these particles leave the CDC and re-enter after loosing a large part of their momentum in the detector material between the CDC and the ECL. The secondary loops of these *outer curlers* are not adequately handled by the fitting algorithm and result in failed fitting attempts on these track predictions. The *Baseline* algorithm does not assign these secondary loops to their tracks, resulting in fittable predictions while the *CATFinder* often includes them, resulting in failing fits. How much of the barrel *track charge efficiency* reduction due to *neural clustering* can be attributed to the higher *hit efficiency* and hit ordering problems, remains to be studied by employing a more sophisticated hit ordering algorithm. Figure 6.15 provides a granular comparison between *neural* and *spherical clustering* with tracks binned by polar angle and transverse momentum. The direct comparison clearly shows the improvements in the endcaps and similar performance for high-momentum barrel tracks, with a clear performance drop on the *outer curlers*.

As shown in Figure A.4 and Figure A.5, *hit efficiency* is improved at the cost of *hit purity* with no direct impact on the momentum resolution. A detailed overview of the integrated metrics by detector region is provided in Table A.2, showing a *fake rate* increase but *clone rate decrease* with *efficiency* improvements in the endcaps but lower performance in the

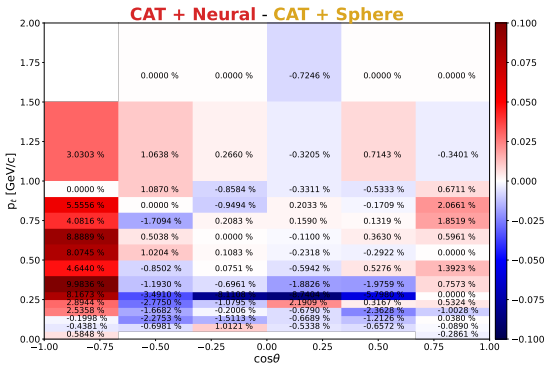


Figure 6.15.: Comparison between *neural* and *spherical clustering* by *track fitting charge efficiency* by polar angle and transverse momentum on $\Upsilon(4S) \longrightarrow B\bar{B}$ samples with *high data beam background*. Positive values indicate that the *neural clustering* performs better than *spherical clustering*.

barrel region. The similarities between the $\mu^{\pm}$ with low p_t and $B \longrightarrow LLP \longrightarrow e^+e^-$ analysis indicate that the *neural clustering* generalizes across different track multiplicities, with similar performance changes in both cases.

6.2.4. Low Multiplicity Small Opening Angle

In this subsection, the performance is evaluated on a sample of $e^+e^- \longrightarrow \pi^+\pi^-\gamma$ events. These events contain two prompt $\pi^{\pm}$ tracks and one γ, which is not visible in the CDC. Depending on the momentum of the γ, the $\pi^{\pm}$ can have a very small opening angle, resulting in hard to distinguish tracks. These tracks are often predicted close together in the *CATFinder*-GNN latent space and the *spherical clustering* fails to separate them (similar to Figure 2.5b).

Figure 6.17 and Table A.3 show that the *neural clustering* again improves *efficiency* in the endcap regions with a slight reduction in the barrel region for very low transverse momentum. There is no significant difference between *neural* and *spherical clustering* for large opening angles (see Figure 6.18), but at small opening angles, *neural clustering* significantly improves the *efficiency*.
Neural clustering increases *hit efficiency* at the cost of *hit purity*, without influencing the post-fitting momentum resolution as can be seen in Figure A.8 and Figure A.9.

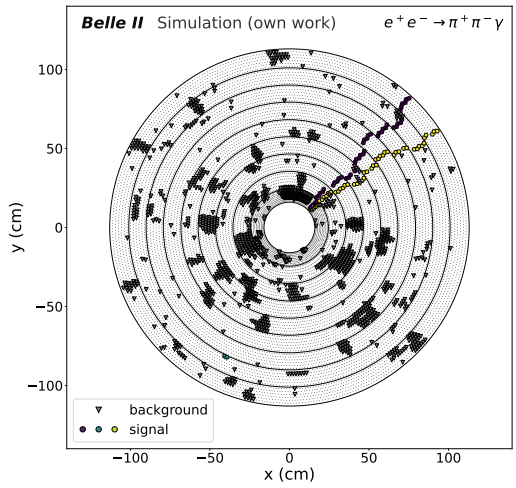
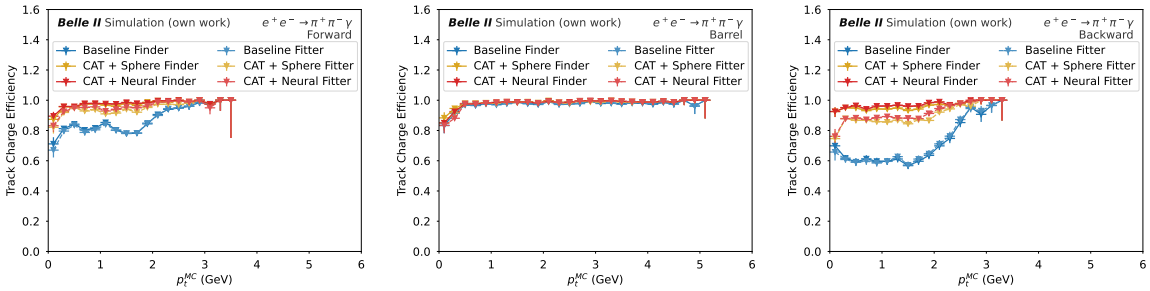


Figure 6.16.: $e^+e^- \longrightarrow \pi^+\pi^-\gamma$ event display in the $x-y$-plane for *high data beam background*. Colored circular markers indicate signal hits. Filled gray triangular markers show background hits.

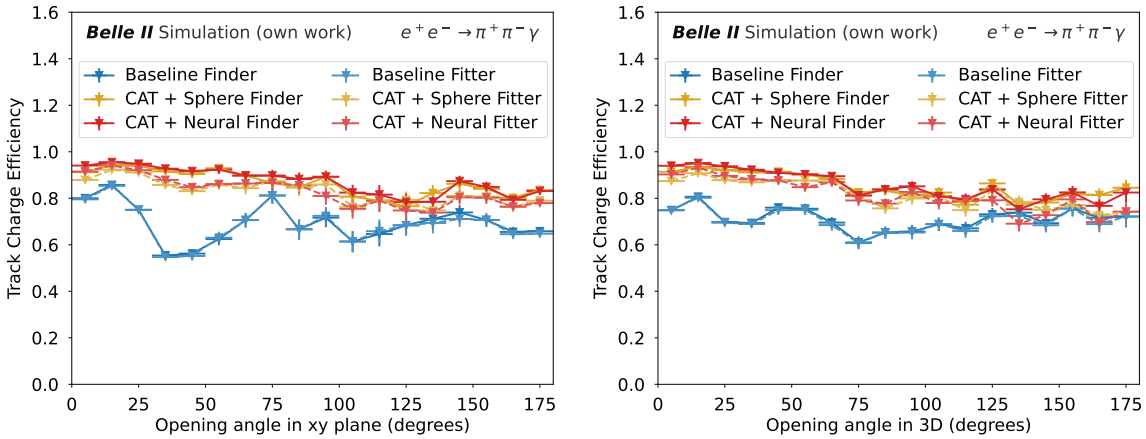


(a) Forward endcap. (b) Barrel. (c) Backward endcap.

Figure 6.17.: *Track charge efficiency* before (solid lines) and after (dashed lines) track fitting for $\pi^\pm$ in $e^+e^- \longrightarrow \pi^+\pi^-\gamma$ samples with *high data beam background* for different algorithms and detector regions. Horizontal bars indicate bin width, vertical error bars indicate uncertainties (correlated between algorithms).

Table 6.6.: The performance metrics for $e^+e^- \longrightarrow \pi^+\pi^-\gamma$ samples for different track finding algorithms *high data beam background*. Uncertainties below 0.01% are not shown in the table, uncertainties are correlated between different methods.

Algorithm	$\varepsilon_{\text{trk,ch}}$	$\mathfrak{p}_{\text{trk}}$
Baseline Finder	$87.76^{+0.21}_{-0.21}$	$89.52^{+0.20}_{-0.20}$
CAT + Sphere Finder	$97.48^{+0.11}_{-0.10}$	$78.89^{+0.24}_{-0.24}$
CAT + Neural Finder	$97.80^{+0.10}_{-0.10}$	$76.65^{+0.25}_{-0.25}$
Baseline Fitter	$87.27^{+0.22}_{-0.22}$	$90.06^{+0.20}_{-0.20}$
CAT + Sphere Fitter	$95.84^{+0.13}_{-0.13}$	$87.38^{+0.21}_{-0.21}$
CAT + Neural Fitter	$96.57^{+0.12}_{-0.12}$	$86.70^{+0.21}_{-0.21}$



(a) By radial opening angle.

(b) By 3D opening angle.

Figure 6.18.: *Track charge efficiency* before (solid lines) and after (dashed lines) track fitting in $B \longrightarrow LLP \longrightarrow e^+e^-$ samples with *high data beam background* for different algorithms. Horizontal bars indicate bin width, vertical error bars indicate uncertainties (correlated between algorithms).

6.3. Inference Timing

At SuperKEKB $\Upsilon(4S)$ are produced at a rate of 0.67 kHz at target luminosity with a data acquisition rate of 10 kHz for all triggered events [3]. Thus a low inference latency per event is critical, as a small computational overhead per event leads to a high absolute increase in computational cost.
This section compares the time required for clustering per event for different algorithms. First the process of finding a good set of hyperparameters for Variational Density Peak Clustering (VDPC) is described. Then, *spherical* and *neural clustering* are compared with VDPC, a density-based clustering algorithm [15] described in more detail in Section 2.3. Finally the time required for clustering is evaluated across these algorithms.

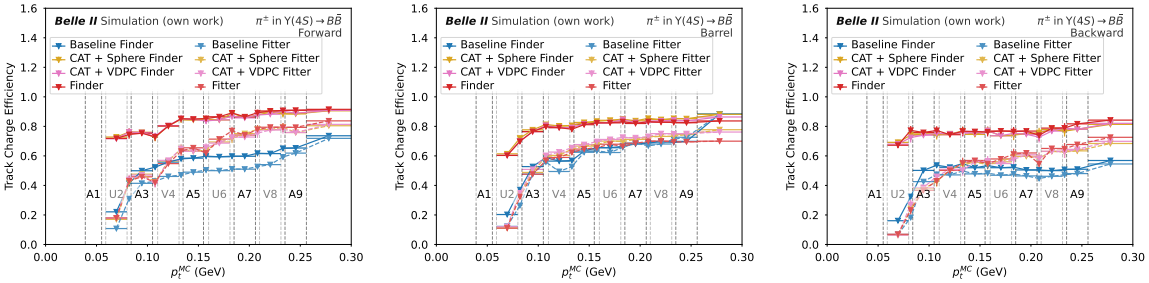
VDPC Working Point Search VDPC depends on three hyperparameters (d_c, $\delta_{\min}$, $\rho_{\min}$) which can take any positive real number. This makes a full gridsearch of the parameter space impossible, so a genetic algorithm is employed to find the optimum. A population of configurations is generated randomly and each configuration is evaluated on the $\Upsilon(4S) \to B\bar{B}$ dataset used for all working point searches. The parameter sets with highest *track charge efficiency* and *track purity* after fitting are then combined and mutated for the next generation. The best configurations in each run are defined as all configurations where no other entry performs better in both *track charge efficiency* and *track purity*. Once the search is converged, the best parameter set is selected with the figure of merit introduced in Subsubsection 5.2.3.2. This results in the hyperparameter set

$$d_c = 0.4,\ \delta_{\min} = 0.1,\ \rho_{\min} = 0.7.$$

VDPC Clustering Performance As the hyperparameters for VDPC are fitted on $\Upsilon(4S) \to B\bar{B}$ decays, the algorithm is evaluated on the same event topology to assess its best-case performance. Across all detector regions VDPC performs similar to the *spherical clustering* (see Figure 6.19) with a slight increase in *track fitting charge efficiency* at the cost of *track fitting purity* shown in the integrated metrics in Table 6.7. As this best-case performance of VDPC does not improve upon the other clustering algorithms, no large improvements are expected if the working point search is constrained to hyperparameters with good generalization to other samples.

The current set of parameters does not generalize well to other topologies as shown in Figure 6.20, where the performance is shown on displaced tracks in high-multiplicity events. On these tracks VDPC does not provide a significant improvement over *spherical clustering* while *neural clustering* provides a higher *track fitting charge efficiency*.

Inference Timing The time required for the clustering step is measured four times on the same sample of 2000 $\Upsilon(4S) \to B\bar{B}$ events. The cluster space prediction of the *CATFinder*-GNN is precalculated for each event and distributed to all clustering algorithms. This is done by iterating over all events and for each event iterating over all clustering algorithms. The order of clustering algorithms is randomized for each event to avoid systematical errors due to memory access. To provide a fair comparison between algorithms, all inferences are limited to a single thread and no GPU usage although especially the *neural clustering*



(a) Forward endcap. (b) Barrel. (c) Backward endcap.

Figure 6.19.: *Track charge efficiency* before (solid lines) and after (dashed lines) track fitting for $\pi^{\pm}$ in $\Upsilon(4S) \longrightarrow B\bar{B}$ samples with *high data beam background* for different clustering algorithms and detector regions. Horizontal bars indicate bin width, vertical error bars indicate uncertainties (correlated between algorithms).

Table 6.7.: The performance metrics for $\Upsilon(4S) \longrightarrow B\bar{B}$ samples for different track clustering algorithms on *high data beam background*. Uncertainties below 0.01% are not shown in the table, uncertainties are correlated between different methods.

Algorithm	$\varepsilon_{\text{trk,ch}}$	$\mathfrak{p}_{\text{trk}}$
Baseline Finder	$78.21^{+0.05}_{-0.05}$	$83.27^{+0.04}_{-0.04}$
CAT + Sphere Finder	$88.94^{+0.08}_{-0.08}$	$70.30^{+0.11}_{-0.11}$
CAT + VDPC Finder	$88.34^{+0.08}_{-0.08}$	$70.20^{+0.11}_{-0.11}$
CAT + Neural Finder	$88.10^{+0.08}_{-0.08}$	$69.25^{+0.11}_{-0.11}$
Baseline Fitter	$75.59^{+0.05}_{-0.05}$	$84.67^{+0.04}_{-0.04}$
CAT + Sphere Fitter	$77.99^{+0.11}_{-0.11}$	$80.67^{+0.10}_{-0.10}$
CAT + VDPC Fitter	$78.33^{+0.11}_{-0.11}$	$79.88^{+0.11}_{-0.10}$
CAT + Neural Fitter	$77.03^{+0.11}_{-0.11}$	$79.69^{+0.11}_{-0.11}$

would profit from a GPU inference.
The results for this timing evaluation are shown in Figure 6.21a showing a clear speed difference between the three clustering algorithms. *Spherical clustering* is the fastest at an inference time of (9.2 ± 8.9) ms, while *neural clustering* provides a better clustering quality with slower inference of (16.2 ± 5.3) ms. VDPC takes significantly longer ((58.1 ± 20.5) ms) which has two main reasons. It requires a full $n_{\text{hits}} \times n_{\text{hits}}$ distance matrix which introduces a large overhead for B-meson decays with $n_{\text{hits}} \approx 2000$. Additionally, all algorithms are currently fully implemented in `Python` with VDPC being implemented less efficiently than the other algorithms which mostly rely on standard `PyTorch` functions.

To indicate the possible room for improvement, the same experiment is performed with the *neural clustering* fully moved to a GPU without including the data transfer, as in an end-to-end integration into the *CATFinder*-GNN with GPU inference all tensors would already be on the correct device. An additional speedup for *spherical* and *neural clustering* is expected by compiling the complete end-to-end inference in `C++`. A very rough approximation for the time required by optimized state-of-the-art clustering algorithms is done by timing the implementation of HDBSCAN from [22]. This implementation already

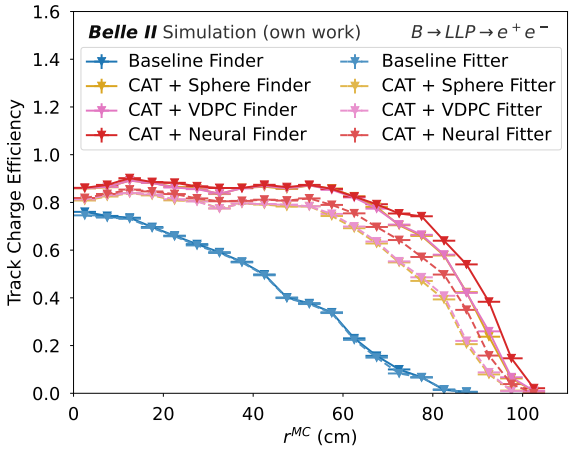
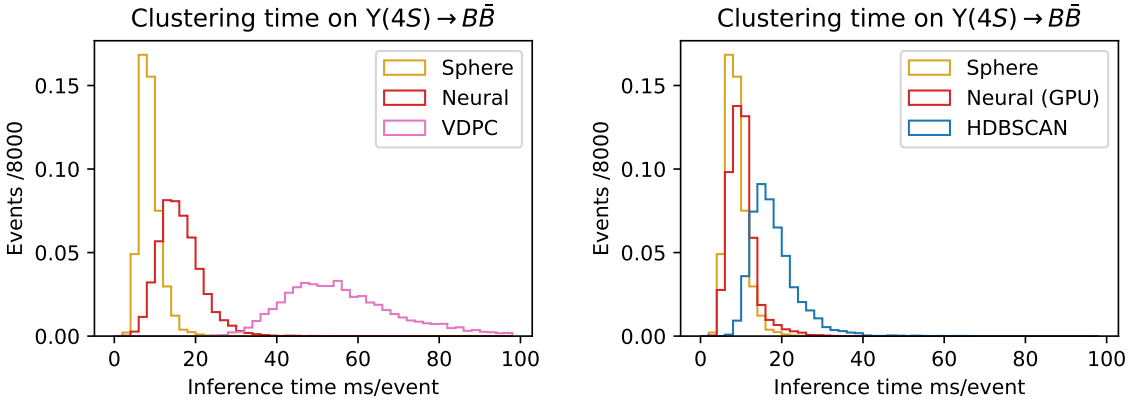


Figure 6.20.: *Track charge efficiency* by vertex displacement before (solid lines) and after (dashed lines) track fitting in $B \longrightarrow LLP \longrightarrow e^+e^-$ samples with *high data beam background* for different algorithms. Horizontal bars indicate bin width, vertical error bars indicate uncertainties (correlated between algorithms).



(a) Time required with all algorithms implemented in `Python` and limited to a single CPU.

(b) Time required with optimized algorithms and the *neural clustering* utilizing the GPU.

Figure 6.21.: Time required for the clustering step evaluated on $\Upsilon(4S) \longrightarrow B\bar{B}$ samples with *high data beam background* for the current implementation Figure 6.21a and with additional optimization Figure 6.21b.

employs efficient data structures like `KDTrees` implemented in `Cython`. The results of this experiment are shown in Figure 6.21b. Moving the neural inference to GPU reclaims most of the speed difference between *spherical* and *neural clustering*, reducing the inference time to (10.3 ± 6.0) ms. A more efficient implementation of VDPC could reach the performance of HDBSCAN, requiring (18.0 ± 5.7) ms, but this would not only require porting the algorithm to `C++`, it would also have to be restructured to utilize more efficient data structures.

A more significant evaluation of the real-world computational cost will be possible once all algorithms are compiled for the actual architecture.

7. Approach Variations

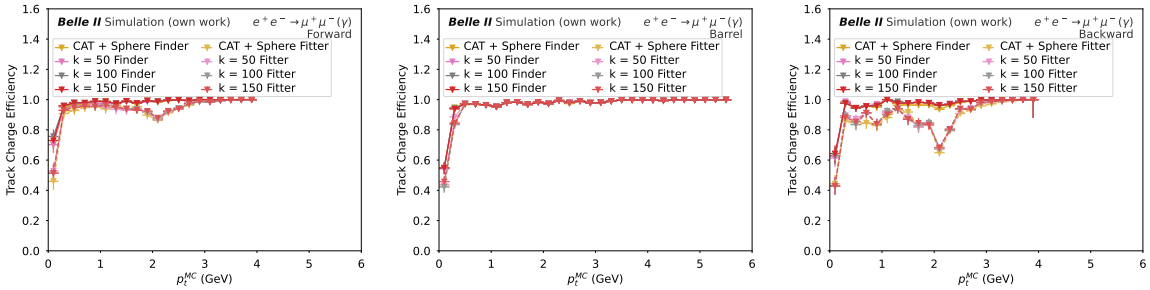
Several additional experiments and ablation studies were performed during the development of the *neural clustering*. These studies are presented in the following chapter. First, two ablation studies are performed to validate choices made during the design of the clustering algorithm. These experiments vary the hyperparameter k and test whether replacing the cluster fusion with a more straightforward cluster suppression affects clustering performance. Then, several experiments are carried out as a proof of concept for further developments. The *neural clustering* is applied to a different *CATFinder*-GNN model that maps into a five- instead of three-dimensional cluster space to validate the independence on the input space dimension. Then the *neural clustering* is performed including analog hit information to show the feasibility of integrating skip-connections to earlier GNN-layers or even to the direct input. Finally, the *neural clustering* model is integrated into the *CATFinder*-GNN and both models are fine-tuned end-to-end together, showing the advantages of a differentiable clustering algorithm.
All approaches are evaluated on the $B \rightarrow LLP \rightarrow e^+e^-$ sample, as this sample shows the clearest performance differences by varying the clustering method. Where the performance on other samples is significantly affected, these samples are shown as well.

7.1. Neighbor Count

A central hyperparameter required for the *neural clustering* is k, which represents the number of inputs for the neural network. This number defines how many nearest neighbors of each *condensation point* are labeled by the network, resulting in the maximum cluster size before cluster fusion. The amount of spacial context provided to the neural network would be maximized by increasing k to include as many points as possible in each prediction. But as most tracks have ~ 50 hits, this comes with diminishing returns for larger values of k, while significantly increasing the computational cost, as the inference cost increases linear with the number of inputs for our architecture.
In the following ablation study, the performance effect of reducing k is studied.

High Transverse Momentum On $e^+e^- \rightarrow \mu^-\mu^+(\gamma)$ samples, the performance does not differ significantly between different values of k. As can be seen in Figure 7.1 and Table 7.1, the *neural clustering* algorithms with different nearest neighbor counts all perform within statistical uncertainty of each other. This is expected on prompt tracks with high transverse momentum, as these tracks on average contain one hit per Central Drift Chamber (CDC)-wire layer, resulting in ≈ 56 hits per track. So for $k \in \{50, 100, 150\}$, the higher values of k do not add additional signal hits to the neural inference, thus not significantly

improving the prediction performance.



(a) Forward endcap. (b) Barrel. (c) Backward endcap.

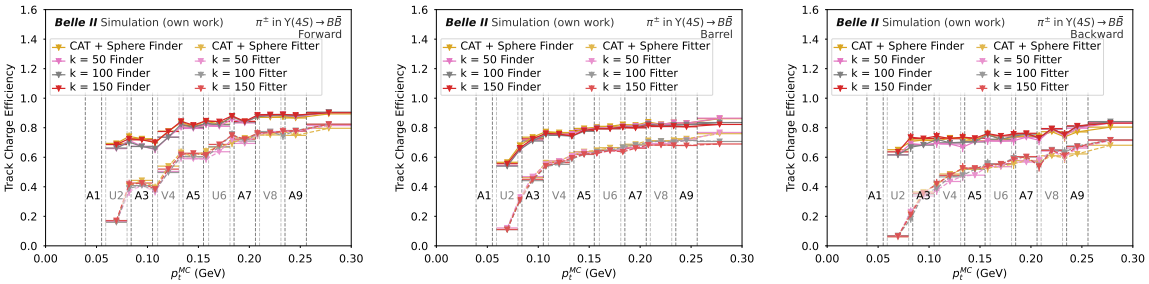
Figure 7.1.: *Track charge efficiency* before (solid lines) and after (dashed lines) track fitting for $e^+e^- \to \mu^-\mu^+(\gamma)$ samples with *high data beam background* for different nearest neighbor counts and detector regions. Horizontal bars indicate bin width, vertical error bars indicate uncertainties (correlated between algorithms).

Table 7.1.: The performance metrics for $e^+e^- \to \mu^-\mu^+(\gamma)$ samples for different track finding algorithms *high data beam background*. Uncertainties below 0.01% are not shown in the table, uncertainties are correlated between different methods.

Algorithm	$\varepsilon_{\text{trk,ch}}$	$\mathfrak{p}_{\text{trk}}$
CAT + Sphere Finder	$92.56^{+0.14}_{-0.14}$	$88.35^{+0.17}_{-0.17}$
CAT + Neural k = 50 Finder	$92.72^{+0.14}_{-0.14}$	$85.28^{+0.18}_{-0.18}$
CAT + Neural k = 100 Finder	$92.73^{+0.14}_{-0.14}$	$86.89^{+0.18}_{-0.18}$
CAT + Neural k = 150 Finder	$92.72^{+0.14}_{-0.14}$	$85.95^{+0.18}_{-0.18}$
CAT + Sphere Fitter	$91.65^{+0.15}_{-0.15}$	$96.91^{+0.10}_{-0.10}$
CAT + Neural k = 50 Fitter	$92.00^{+0.15}_{-0.15}$	$96.36^{+0.10}_{-0.10}$
CAT + Neural k = 100 Fitter	$91.97^{+0.15}_{-0.15}$	$96.54^{+0.10}_{-0.10}$
CAT + Neural k = 150 Fitter	$91.97^{+0.15}_{-0.15}$	$96.55^{+0.10}_{-0.10}$

Low Transverse Momentum A more significant difference in performance is expected on tracks with low transverse momentum, which curl inside the tracking volume, leaving a large number of hits. These tracks are evaluated on the $\Upsilon(4S) \to B\bar{B}$ sample, but due to the hit ordering problems with curling tracks, the *track efficiency* is biased in favor of algorithms which find less hits per track, resulting in a post-fitting *efficiency* drop for higher k (see Table 7.2). The performance on curlers is also visible in Figure 7.2, where $k = 150$ performs best in the endcap regions (where tracks are short), while performing worst in the barrel region, as most multi-loop curlers are found in the barrel, because endcap tracks often leave the CDC before they complete more than one loop.
A more significant statistic is thus the pre-fitting *hit efficiency*, which shows how consistently hits are assigned to tracks, indicating the post-fitting performance in the case of a more robust hit ordering algorithm. This is shown in Figure 7.3 and Table 7.3, focusing

on the curling tracks in the barrel region. A clear trade-off between *hit efficiency* and *hit purity* is visible, as increasing k results in a significant increase in *hit efficiency* at the cost of a smaller *hit purity* loss.



(a) Forward endcap. (b) Barrel. (c) Backward endcap.

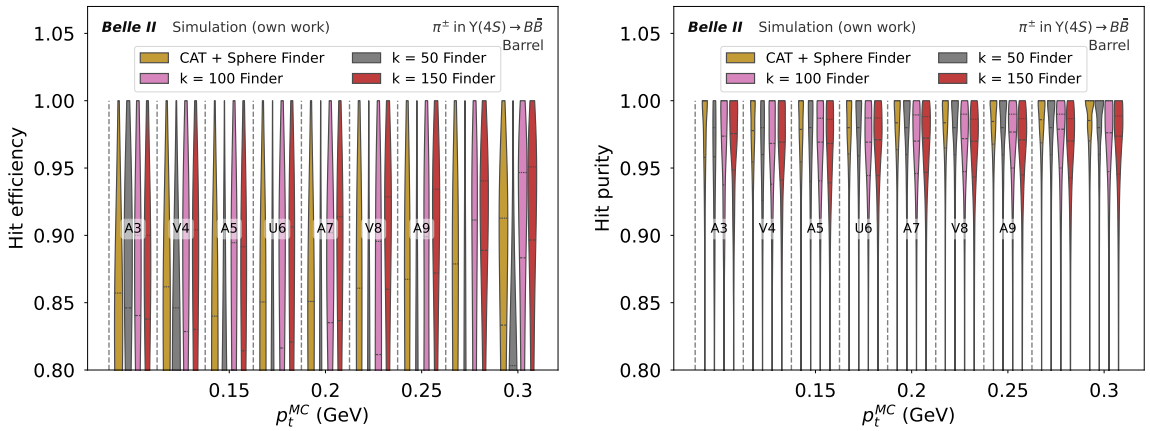
Figure 7.2.: *Track charge efficiency* before (solid lines) and after (dashed lines) track fitting for $\Upsilon(4S) \longrightarrow B\bar{B}$ samples with *high data beam background* for different nearest neighbor counts and detector regions. Horizontal bars indicate bin width, vertical error bars indicate uncertainties (correlated between algorithms).

Table 7.2.: The performance metrics for $\Upsilon(4S) \longrightarrow B\bar{B}$ samples for different values of k in *high data beam background*. Uncertainties below 0.01% are not shown in the table, uncertainties are correlated between different methods.

Algorithm	$\varepsilon_{\text{trk,ch}}$	$\mathfrak{p}_{\text{trk}}$
CAT + Sphere Finder	$85.81^{+0.08}_{-0.08}$	$75.55^{+0.09}_{-0.09}$
CAT + Neural k = 50 Finder	$85.51^{+0.08}_{-0.08}$	$73.96^{+0.09}_{-0.09}$
CAT + Neural k = 100 Finder	$85.12^{+0.08}_{-0.08}$	$75.28^{+0.09}_{-0.09}$
CAT + Neural k = 150 Finder	$85.23^{+0.08}_{-0.08}$	$74.81^{+0.09}_{-0.09}$
CAT + Sphere Fitter	$76.44^{+0.09}_{-0.09}$	$84.77^{+0.08}_{-0.08}$
CAT + Neural k = 50 Fitter	$76.56^{+0.09}_{-0.09}$	$83.85^{+0.09}_{-0.08}$
CAT + Neural k = 100 Fitter	$76.14^{+0.09}_{-0.09}$	$84.21^{+0.08}_{-0.08}$
CAT + Neural k = 150 Fitter	$75.84^{+0.09}_{-0.09}$	$84.10^{+0.09}_{-0.08}$

Table 7.3.: The hit-level performance metrics for $\Upsilon(4S) \longrightarrow B\bar{B}$ samples for different values of k on *high data beam background*. Uncertainties below 0.01% are not shown in the table, uncertainties are correlated between different methods.

Algorithm	ϵ_{hit} [%]	$\mathfrak{p}_{\text{hit}}$ [%]
CAT + Sphere	65.7 ± 23.9	96.7 ± 4.8
CAT + Neural k = 50	56.0 ± 20.3	96.0 ± 5.6
CAT + Neural k = 100	72.5 ± 22.6	95.2 ± 5.8
CAT + Neural k = 150	74.4 ± 22.7	94.9 ± 6.0



(a) *Hit efficiency.* (b) *Hit purity.*

Figure 7.3.: *Hit efficiency* and *purity* by transverse momentum before track fitting for $\Upsilon(4S) \to B\bar{B}$ samples with *high data beam background* for different values of k limited to the intersecting sample in the barrel region.

Displaced Tracks As the largest difference between the performance of *neural* and *spherical clustering* was observed on displaced tracks in high-multiplicity scenarios, the $B \to LLP \to e^+e^-$ sample is again used to compare the different values for k.
On this subset of challenging tracks, no significant trade-off between *efficiency* and computational cost can be seen. As shown in Figure 7.4 and in Table 7.4, decreasing k does not result in an *efficiency* loss. Table 7.5 shows that the *neural clustering* performs similar in *hit efficiency* and *hit purity* for all values of k. This indicates, that the computational cost of the clustering step can be reduced, loosing performance only on curling tracks, where the hit-ordering currently already does not handle tracks with many hits well. Overall, reducing the number of nearest neighbors is a promising candidate for future speedup if the *neural clustering* is slower than required. This will be evaluated once the *CATFinder* and the *neural clustering* are ported to C++ and the final inference latency can be measured.

Table 7.4.: The performance metrics for $B \to LLP \to e^+e^-$ samples for different values of k on *high data beam background*. Uncertainties below 0.01% are not shown in the table, uncertainties are correlated between different methods.

Algorithm	$\varepsilon_{\text{trk,ch}}$	$\mathfrak{p}_{\text{trk}}$
CAT + Sphere Finder	$77.36^{+0.22}_{-0.22}$	$44.18^{+0.20}_{-0.20}$
CAT + Neural k = 50 Finder	$79.34^{+0.21}_{-0.21}$	$41.02^{+0.18}_{-0.18}$
CAT + Neural k = 100 Finder	$80.01^{+0.21}_{-0.21}$	$43.33^{+0.19}_{-0.19}$
CAT + Neural k = 150 Finder	$80.10^{+0.21}_{-0.21}$	$43.10^{+0.19}_{-0.19}$
CAT + Sphere Fitter	$68.97^{+0.24}_{-0.24}$	$56.74^{+0.23}_{-0.23}$
CAT + Neural k = 50 Fitter	$72.42^{+0.23}_{-0.23}$	$55.28^{+0.23}_{-0.23}$
CAT + Neural k = 100 Fitter	$72.91^{+0.23}_{-0.23}$	$56.34^{+0.23}_{-0.23}$
CAT + Neural k = 150 Fitter	$72.96^{+0.23}_{-0.23}$	$56.42^{+0.23}_{-0.23}$

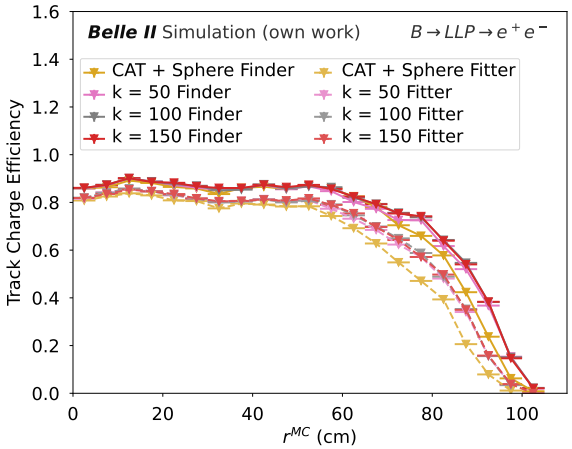


Figure 7.4.: *Track charge efficiency* before (solid lines) and after (dashed lines) track fitting for $B \longrightarrow LLP \longrightarrow e^+e^-$ samples with *high data beam background* for different nearest neighbor counts. Horizontal bars indicate bin width, vertical error bars indicate uncertainties (correlated between algorithms).

Table 7.5.: The hit-level performance metrics for $B \longrightarrow LLP \longrightarrow e^+e^-$ samples for different values of k on *high data beam background*. Uncertainties below 0.01% are not shown in the table, uncertainties are correlated between different methods.

Algorithm	*hit efficiency*	*hit purity*
CAT + Sphere	76.4 ± 18.4	97.9 ± 4.2
CAT + Neural k = 50	85.9 ± 15.7	93.9 ± 7.1
CAT + Neural k = 100	85.7 ± 15.5	93.6 ± 7.4
CAT + Neural k = 150	86.0 ± 14.8	93.6 ± 7.5

7.2. Non-Max Suppression

The final step in the clustering pipeline is the fusion of overlapping clusters to remove duplicates. In the You Only Look Once (YOLO)-pipeline, this step does not merge overlapping clusters but discards all except the one with highest prediction confidence. The following ablation study evaluates the impact of merging overlapping cluster predictions instead of suppressing duplicates.
On the $B \longrightarrow LLP \longrightarrow e^+e^-$ sample, no significant difference between *neural clustering* with and without cluster fusion (merging) is observed (see Table 7.6). This is due to the fact that almost all clusters have less than $k = 150$ hits, resulting in identical cluster predictions for all *condensation points* of the same cluster. Cluster fusion thus only adds very few hits to the clusters, as each cluster prediction already contains the full cluster before merging. Figure 7.6 shows that the cluster fusion does not result in adding more hits to each track prediction, as neither *hit efficiency*, nor *hit purity* are affected.
This also holds for tracks with a higher number of hits, as present in the $\Upsilon(4S) \longrightarrow B\bar{B}$ sample, shown in Figure A.10 and Figure A.11. This sample contains many tracks with low transverse momentum, which can contain more than k hits. The fact that cluster fusion has

no effect on this sample can be attributed to the *condensation point* selection. *Condensation points* are only points which the *CATFinder*-GNN predicts as being good candidates for cluster centers. These candidates are likely to be predicted very close together in the cluster space. This can result in them sharing the same 150 nearest neighbors, resulting in identical cluster predictions, while additional points of the same cluster might be further away and thus not provided to the clustering network.
A larger effect of the cluster fusion might be observed by lowering k below the average cluster size, as then the cluster predictions would not be able to completely cover the actual clusters.
This ablation study shows that the cluster fusion does not strongly change the clustering performance. As cluster fusion does not introduce additional complexity and does not reduce performance compared to cluster suppression, it is not removed from the pipeline.

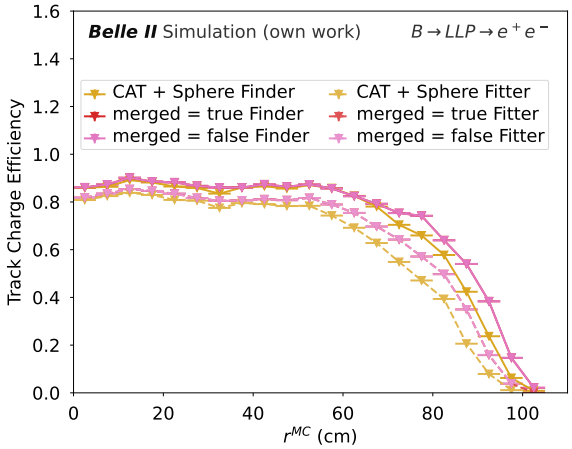
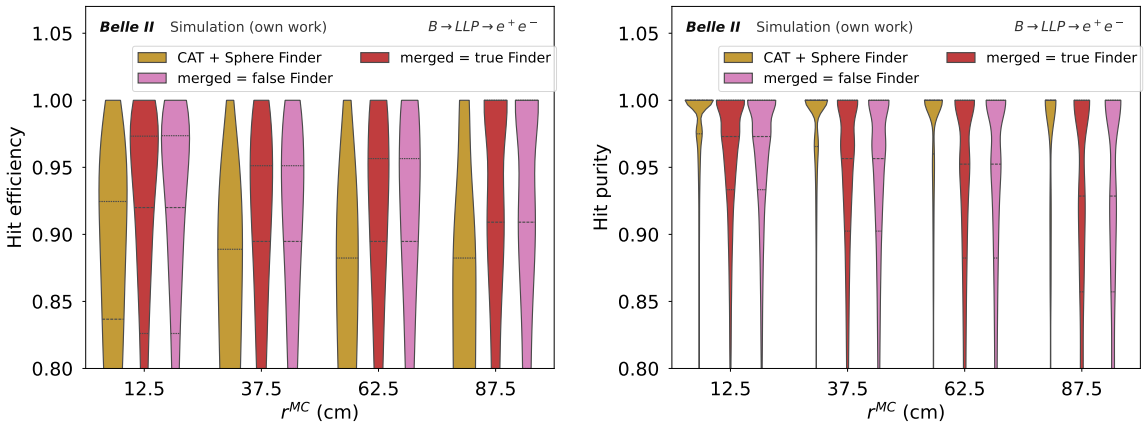


Figure 7.5.: *Track charge efficiency* before (solid lines) and after (dashed lines) track fitting for $B \longrightarrow LLP \longrightarrow e^+e^-$ samples with *high data beam background* with and without cluster merging. Horizontal bars indicate bin width, vertical error bars indicate uncertainties (correlated between algorithms). The series with and without cluster merging are identical and thus overlap.

Table 7.6.: The performance metrics for $B \longrightarrow LLP \longrightarrow e^+e^-$ samples with and without cluster merging on *high data beam background*. Uncertainties below 0.01% are not shown in the table, uncertainties are correlated between different methods.

Algorithm	$\varepsilon_{\text{trk,ch}}$	$\mathfrak{p}_{\text{trk}}$
CAT + Sphere Finder	$77.36^{+0.22}_{-0.22}$	$44.18^{+0.20}_{-0.20}$
merged = true Finder	$80.10^{+0.21}_{-0.21}$	$43.10^{+0.19}_{-0.19}$
merged = false Finder	$80.09^{+0.21}_{-0.21}$	$43.10^{+0.19}_{-0.19}$
CAT + Sphere Fitter	$68.97^{+0.24}_{-0.24}$	$56.74^{+0.23}_{-0.23}$
merged = true Fitter	$72.96^{+0.23}_{-0.23}$	$56.42^{+0.23}_{-0.23}$
merged = false Fitter	$72.95^{+0.23}_{-0.23}$	$56.42^{+0.23}_{-0.23}$



(a) *Hit efficiency.*

(b) *Hit purity.*

Figure 7.6.: *Hit efficiency* and *purity* before track fitting for $B \longrightarrow LLP \longrightarrow e^+e^-$ samples with *high data beam background* with and without cluster merging limited to the intersecting sample in the barrel region.

7.3. 5D Cluster Space

The computational cost of the *neural clustering* is not significantly dependent on the cluster space dimension, as increasing this dimension only increases the input dimension of the first shared Multi-Layer Perceptron (MLP). So increasing the dimension is a viable option to achieve a less crowded cluster space, with clusters being distributed across more dimensions. As indicated by Figure 2.5b, the three cluster space dimensions are already saturated by including information about the real space coordinates (the hits form a ring in the $y-z$-plane) and background suppression (signal is separated from background mostly along the x-axis). Performance improvements on a higher-dimensional cluster space are thus expected, as this would increase the information content of the latent space.
In order to asses the performance on a 5-dimensional cluster space, the underlying *CATFinder*-GNN-model is replaced with a version with 5-dimensional latent space. A new clustering network is trained on the output of this *CATFinder*-GNN, with the same underlying events as the 3D clustering network. The hyperparameters of the original clustering pipeline are not refitted, further improvements could thus be achieved, but as the two provided GNNs are not directly comparable in performance — the (experimental) 5D GNN utilizes an additional input feature, which does not generalize to collision data and is thus removed from the final 3D GNN — this section serves more as a proof-of-concept than a direct comparison.
On the $B \longrightarrow LLP \longrightarrow e^+e^-$ sample, the *neural clustering* shows a strong *track charge efficiency* improvement (see Figure 7.7) at a slight *track purity* loss. A detailed comparison of the integrated track metrics is provided in Table 7.7. The high *efficiencies* for the 5D CAT models is in part due to the additional input feature, so these values are not directly comparable to the 3D performance. A directly comparable *CATFinder*-GNN with 5D cluster space was not trained, as preliminary experiments showed no performance increase due to the *spherical clustering*. With the improvements due to *neural clustering* it could be

worthwhile to revisit this and train a *CATFinder*-GNN with higher dimensional latent space.

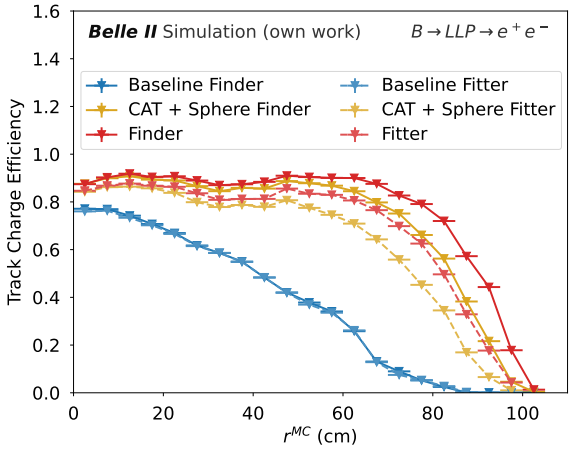


Figure 7.7.: *Track charge efficiency* before (solid lines) and after (dashed lines) track fitting for $B \longrightarrow LLP \longrightarrow e^+e^-$ samples with *high data beam background* with 5D cluster space. Horizontal bars indicate bin width, vertical error bars indicate uncertainties (correlated between algorithms).

Table 7.7.: The performance metrics for $B \longrightarrow LLP \longrightarrow e^+e^-$ samples for different track finding algorithms with 5D cluster space on *high data beam background*. Uncertainties below 0.01% are not shown in the table, uncertainties are correlated between different methods.

Algorithm	$\varepsilon_{\text{trk,ch}}$	$\mathfrak{p}_{\text{trk}}$
Baseline Finder	$44.57^{+0.26}_{-0.26}$	$51.56^{+0.28}_{-0.28}$
CAT (5D) + Sphere Finder	$77.60^{+0.22}_{-0.22}$	$47.67^{+0.20}_{-0.20}$
CAT (5D) + Neural Finder	$82.71^{+0.20}_{-0.20}$	$45.73^{+0.19}_{-0.19}$
Baseline Fitter	$43.96^{+0.26}_{-0.26}$	$54.12^{+0.29}_{-0.29}$
CAT (5D) + Sphere Fitter	$69.26^{+0.24}_{-0.24}$	$58.43^{+0.24}_{-0.24}$
CAT (5D) + Neural Fitter	$74.91^{+0.23}_{-0.23}$	$57.34^{+0.23}_{-0.23}$

7.4. Additional Input Features

A significant difference between *neural clustering* and traditional clustering algorithms is that the *neural clustering* is not limited to take cluster space coordinates as inputs. While most classical clustering algorithms strongly rely on distance metrics and cluster space densities which are not well-defined on additional input features, the neural inference can be enhanced with additional information.
This can be done by adding analog detector values, track parameter predictions or skip-connections to earlier GravNet layers, enabling the neural network to incorporate additional information which is not included in the cluster space prediction.

For this experiment, the *tdc* and *adc* (see Paragraph 2.1) values are appended to the cluster space coordinates of each hit. These parameters are chosen as they are also used in the *CATFinder*-GNN inference and have been shown to generalize to collision data [2]. The clustering network is trained with the same training setup described in Section 5.1 with the additional features concatenated to the input vector of each hit.
On the $B \longrightarrow LLP \longrightarrow e^+e^-$ sample, adding the additional input features increases the *track purity* significantly both before and after fitting, while keeping the same *track charge efficiency* within statistical uncertainty (see Table 7.8). This shows that the *neural clustering* is able to pick up differences in the analog values to discriminate between real and fake tracks more consistently. The GNN is unable to encode all available information into the limited 3D cluster space, thus adding more information either by scaling the cluster space dimension or by including skip-connections to earlier layers or directly to the input improves the clustering performance.

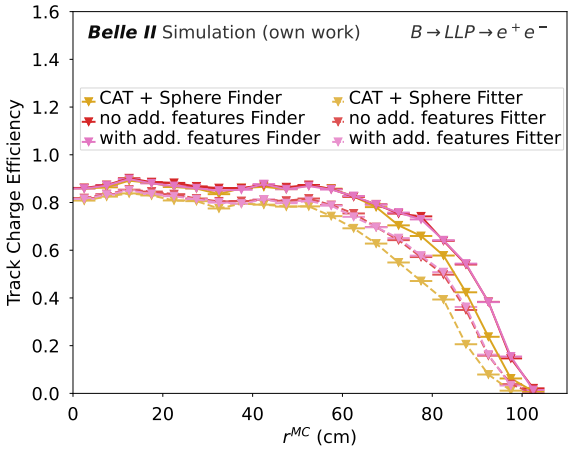


Figure 7.8.: *Track charge efficiency* before (solid lines) and after (dashed lines) track fitting for $B \longrightarrow LLP \longrightarrow e^+e^-$ samples with *high data beam background* with and without including *tdc* and *adc* as additional input features. Horizontal bars indicate bin width, vertical error bars indicate uncertainties (correlated between algorithms).

Table 7.8.: The performance metrics for $B \rightarrow LLP \rightarrow e^+e^-$ samples for *neural clustering* with and without including *tdc* and *adc* as additional input features on *high data beam background*. Uncertainties below 0.01% are not shown in the table, uncertainties are correlated between different methods.

Algorithm	$\varepsilon_{\text{trk,ch}}$	$\mathfrak{p}_{\text{trk}}$
CAT + Sphere Finder	$77.36^{+0.22}_{-0.22}$	$44.18^{+0.20}_{-0.20}$
CAT + Neural no add. features Finder	$80.10^{+0.21}_{-0.21}$	$43.10^{+0.19}_{-0.19}$
CAT + Neural with add. features Finder	$79.70^{+0.21}_{-0.21}$	$45.85^{+0.20}_{-0.20}$
CAT + Sphere Fitter	$68.97^{+0.24}_{-0.24}$	$56.74^{+0.23}_{-0.23}$
CAT + Neural no add. features Fitter	$72.96^{+0.23}_{-0.23}$	$56.42^{+0.23}_{-0.23}$
CAT + Neural with add. features Fitter	$72.63^{+0.23}_{-0.23}$	$58.11^{+0.23}_{-0.23}$

7.5. End-to-End Fine-Tuning

A key feature of the *neural clustering* is the fact that it is mostly differentiable. Gradients can be propagated from the cluster predictions through the neural network to the cluster space coordinates of each point, enabling a combined fine-tuning of the clustering network and the *CATFinder*-GNN. As a proof-of-concept, the *neural clustering* is integrated into the *CATFinder*-GNN, and both models are fine-tuned together. This subsection describes the development of this end-to-end model, starting with training and hyperparameter optimization, and then evaluating its performance on several sample topologies.

Training In this experiment the pre-trained *CATFinder*-GNN and the clustering model are combined into an end-to-end wrapper. The only two non-differentiable steps are the nearest-neighbor selection and the cluster fusion. In end-to-end fine-tuning, the object condensation loss is kept active to bridge the nearest-neighbor selection. As in the original training of the clustering network, the cluster fusion step is not performed during training and the loss is calculated directly on the cluster predictions. All GravNet layers of the GNN are freezed in order to reduce the computational effort, leaving only the final linear layer and the output linear layers for fine-tuning. The final linear layer is unfreezed as it combines the information of all GravNet layers and thus is expected to have a large influence upon the clustering performance. As this layer forms a bottleneck with the track parameter heads also relying on its output, the track parameter losses are kept during fine-tuning, to prevent performance loss on these predictions. To prevent the track parameter losses from overpowering the clustering loss, the track parameter heads are unfreezed and fine-tuned together with the clustering.
In order to compute the clustering loss, the clustering inference is performed as described in Chapter 4. For training, the cluster fusion step is skipped and the output of the neural network is compared with the actual truth cluster memberships.

Working Point Search The same working point search as in Subsubsection 5.2.3.2 is performed for the end-to-end model. The *track fitting charge efficiency* and *track fitting*

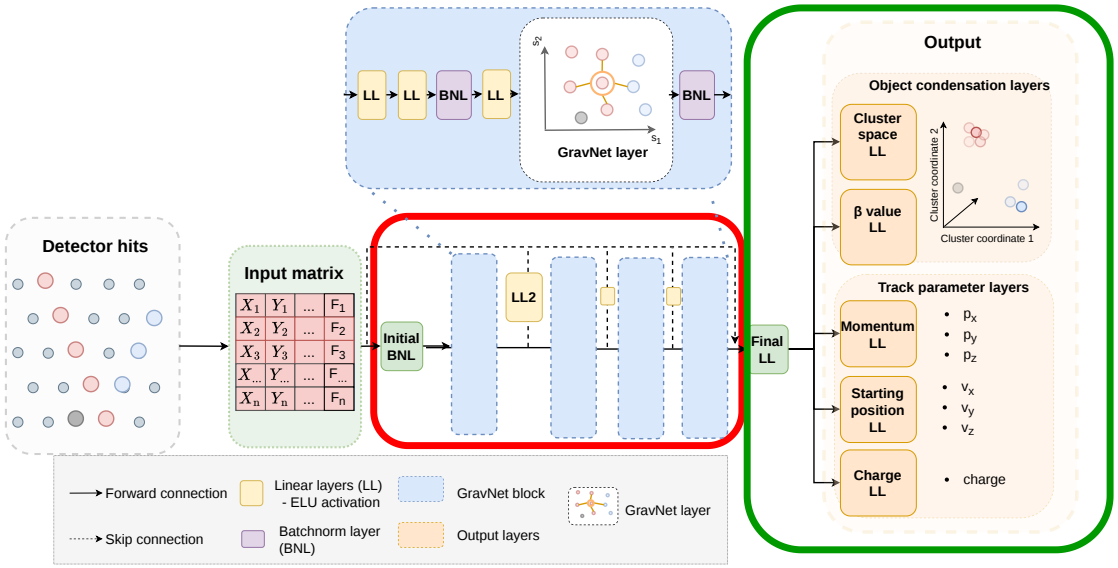
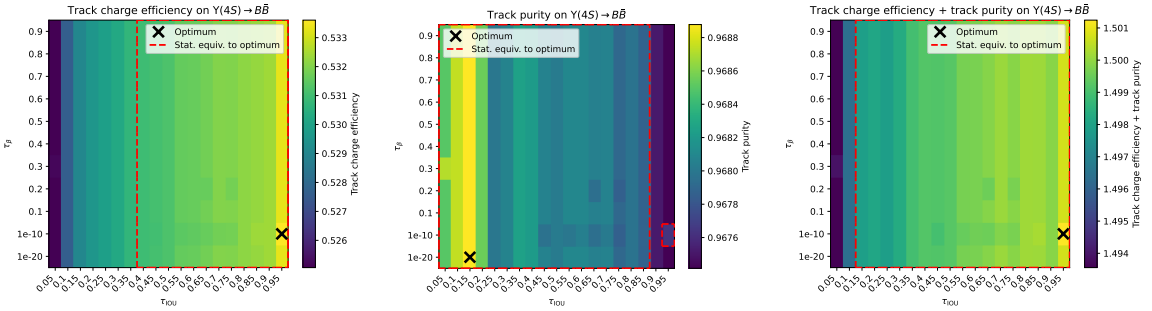


Figure 7.9.: A schematic overview of the CATFinder-GNN architecture showing the fine-tuned parts. The red box indicates the parts of the network, which were frozen during fine-tuning, the green box indicates the fine-tuned layers. Adapted from [1].

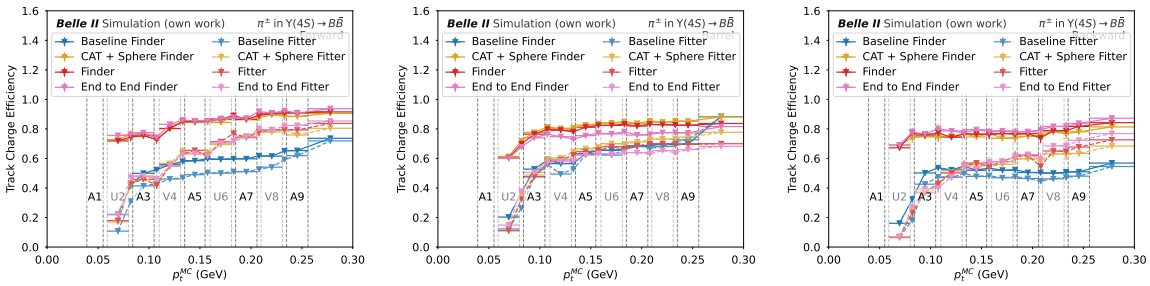
purity for all parameter pairs from the search space in Eqs. (5.2) to (5.3) is evaluated and compared using the figure of merit from Equation 5.4. The distributions shown in Figure 7.10 indicates that different to the evaluation in Subsubsection 5.2.3.2, the trade-off between *efficiency* and *purity* is now dominated by τ_{iou} instead of τ_β with higher values for τ_{iou} increasing *efficiency* at the cost of *purity*. The working point is selected as $\tau_\beta = 10^{-10}$ and $\tau_{\text{iou}} = 0.95$.



(a) *Track charge efficiency* gridsearch. (b) *Track purity* gridsearch. (c) $\mathcal{F}$ gridsearch.

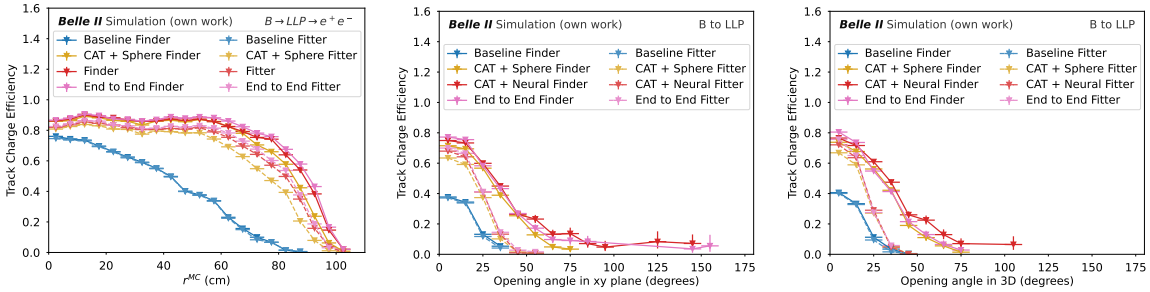
Figure 7.10.: *Track charge efficiency* (a), *track purity* (b) and *figure of merit* (c) for all different combinations of τ_β and τ_{iou} in the gridsearch evaluated on $\Upsilon(4S) \longrightarrow B\bar{B}$ with the end-to-end model. The dashed line marks the region of threshold combinations with performance within statistical uncertainty of the optimum.

Track Finding Performance The performance of the end-to-end integrated model is evaluated on $\pi^{\pm}$ in $\Upsilon(4S) \longrightarrow B\bar{B}$ decays and on $B \longrightarrow LLP \longrightarrow e^+e^-$ samples. Both samples show that the end-to-end fine-tuning follows the trend set by the *neural clustering*.
On $\Upsilon(4S) \longrightarrow B\bar{B}$ samples, the *track finding efficiency* improves in the endcap regions with a slight decline in barrel performance. This is shown in Figure 7.11. Table 7.9 shows that this decrease in integrated *efficiency* comes with an increase in *purity* compared to the non-fine-tuned CATFinder approaches.
On $B \longrightarrow LLP \longrightarrow e^+e^-$ samples, the end-to-end fine-tuning improves both *efficiency* and *purity* compared to non-fine-tuned *spherical* and *neural clustering* as shown in Table 7.10. Figure 7.12 shows that post-fitting *efficiency* improves across the full range of vertex displacement and especially in the case of small opening angles. This indicates that the fine-tuning yields an improved ability to discern between two hard to distinguish tracks.



(a) Forward endcap. (b) Barrel. (c) Backward endcap.

Figure 7.11.: *Track charge efficiency* before (solid lines) and after (dashed lines) track fitting for $\pi^{\pm}$ in $\Upsilon(4S) \longrightarrow B\bar{B}$ samples with *high data beam background* for different algorithms and detector regions. Horizontal bars indicate bin width, vertical error bars indicate uncertainties (correlated between algorithms).



(a) By displacement. (b) By radial opening angle. (c) By 3D opening angle.

Figure 7.12.: *Track charge efficiency* before (solid lines) and after (dashed lines) track fitting in $B \longrightarrow LLP \longrightarrow e^+e^-$ samples with *high data beam background* for different algorithms. Horizontal bars indicate bin width, vertical error bars indicate uncertainties (correlated between algorithms).

Table 7.9.: The performance metrics for $\Upsilon(4S) \to B\bar{B}$ samples for different track finding algorithms on *high data beam background*. Uncertainties below 0.01% are not shown in the table, uncertainties are correlated between different methods.

Algorithm	$\varepsilon_{\text{trk,ch}}$	$\mathfrak{p}_{\text{trk}}$
Baseline Finder	$78.21^{+0.05}_{-0.05}$	$83.27^{+0.04}_{-0.04}$
CAT + Sphere Finder	$88.94^{+0.08}_{-0.08}$	$70.30^{+0.11}_{-0.11}$
CAT + Neural Finder	$88.10^{+0.08}_{-0.08}$	$69.25^{+0.11}_{-0.11}$
End to End Finder	$86.94^{+0.09}_{-0.09}$	$71.01^{+0.11}_{-0.11}$
Baseline Fitter	$75.59^{+0.05}_{-0.05}$	$84.67^{+0.04}_{-0.04}$
CAT + Sphere Fitter	$77.99^{+0.11}_{-0.11}$	$80.67^{+0.10}_{-0.10}$
CAT + Neural Fitter	$77.03^{+0.11}_{-0.11}$	$79.69^{+0.11}_{-0.11}$
End to End Fitter	$76.93^{+0.11}_{-0.11}$	$79.93^{+0.11}_{-0.11}$

Table 7.10.: The performance metrics for $B \to LLP \to e^+e^-$ samples for different track finding algorithms on *high data beam background*. Uncertainties below 0.01% are not shown in the table, uncertainties are correlated between different methods.

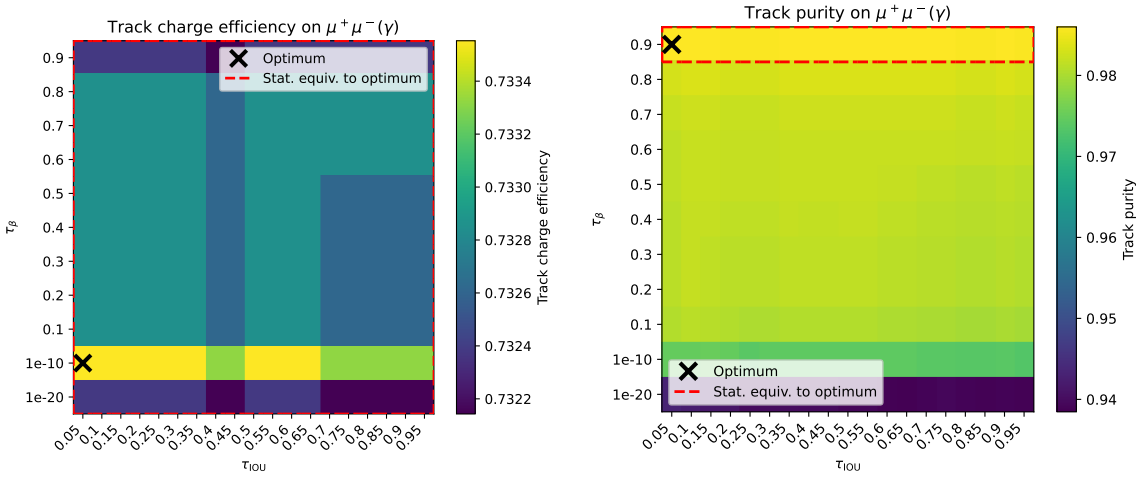
Algorithm	$\varepsilon_{\text{trk,ch}}$	$\mathfrak{p}_{\text{trk}}$
Baseline Finder	$44.54^{+0.26}_{-0.26}$	$52.26^{+0.28}_{-0.28}$
CAT + Sphere Finder	$77.36^{+0.22}_{-0.22}$	$45.84^{+0.20}_{-0.20}$
CAT + Neural Finder	$80.10^{+0.21}_{-0.21}$	$43.88^{+0.19}_{-0.19}$
End to End Finder	$81.41^{+0.20}_{-0.20}$	$45.17^{+0.19}_{-0.19}$
Baseline Fitter	$43.77^{+0.26}_{-0.26}$	$54.67^{+0.29}_{-0.29}$
CAT + Sphere Fitter	$68.97^{+0.24}_{-0.24}$	$57.70^{+0.24}_{-0.24}$
CAT + Neural Fitter	$72.96^{+0.23}_{-0.23}$	$56.76^{+0.23}_{-0.23}$
End to End Fitter	$74.43^{+0.23}_{-0.23}$	$56.98^{+0.23}_{-0.23}$

8. Validation

This chapter summarizes several validation experiments performed to ensure the applicability of the previous results for real-world usage.
First, the generalization of the fitted thresholds is evaluated by performing the same gridsearch on a different event topology and comparing the results. Then, the *neural clustering* is implemented in the full Belle II tracking chain and the full-chain performance is evaluated. Finally, the *CATFinder* with *neural clustering* is applied on real collision data, investigating whether the optimization on simulated data yields good results for actual usage.

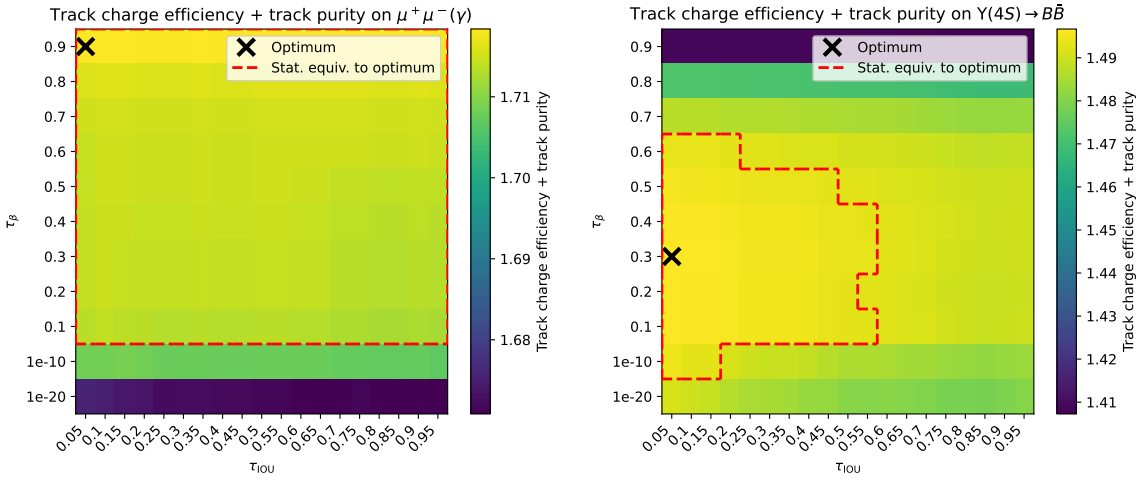
8.1. Hyperparameter Generalizability

In order to ensure that the *neural clustering* can be applied for all events present in Belle II, the hyperparameters chosen in the working point search have to generalize across different event topologies. In Subsubsection 5.2.3.2 the search for the optimal set of thresholds is performed on the $\Upsilon(4S) \rightarrow B\bar{B}$ sample with the intention that this sample should provide a sharp trade-off between *track charge efficiency* and *track purity*, as it combines a high number of low transverse momentum tracks with a high level of background hits. The working point search assumes that this sharp trade-off results in an optimum which lies in the larger range of optimal parameters on samples with easier event topologies.
In the following section, the validity of this assumption will be evaluated by performing the same gridsearch as in Subsubsection 5.2.3.2 on the $e^+e^- \rightarrow \mu^-\mu^+(\gamma)$ sample and comparing these results with the gridsearch on $\Upsilon(4S) \rightarrow B\bar{B}$ samples.
The assumption of a more broad optimum is valid, as shown in Figure 8.1. The *track charge efficiency* lies within statistical uncertainty for all threshold combinations, with the optimum at $\varepsilon_{\text{trk,ch}} = (73.35 \pm 0.68)\%$. The *track purity* also shows a broader optimum than on $\Upsilon(4S) \rightarrow B\bar{B}$ events. A direct comparison of $\mathcal{F}$ between the two gridsearches is shown in Figure 8.2. On the simple topology of $e^+e^- \rightarrow \mu^-\mu^+(\gamma)$ events, the optimal range spans almost the complete set of evaluated thresholds while on $\Upsilon(4S) \rightarrow B\bar{B}$ events there is a tighter constraint.
As the set of optimal threshold configurations on $\Upsilon(4S) \rightarrow B\bar{B}$ events approximately is a subset of the optimal range on $e^+e^- \rightarrow \mu^-\mu^+(\gamma)$ choosing the working point on $\Upsilon(4S) \rightarrow B\bar{B}$ generalizes well to simpler event topologies. The chosen working point thus approximately provides the best possible thresholds across vastly different event types.



(a) *Track charge efficiency* gridsearch. (b) *Track purity* gridsearch.

Figure 8.1.: *Track charge efficiency* and *track purity* after fitting for all different combinations of τ_β and τ_{iou} in the gridsearch evaluated on $e^+e^- \to \mu^-\mu^+(\gamma)$. The dashed line marks the region of threshold combinations with performance within statistical uncertainty of the optimum. For *track charge efficiency* this region contains the whole plot range.



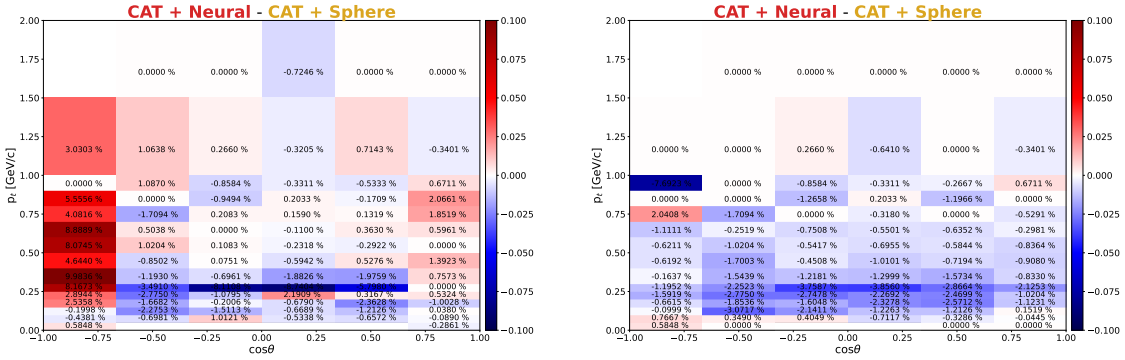
(a) $e^+e^- \to \mu^-\mu^+(\gamma)$ gridsearch. (b) $\Upsilon(4S) \to B\bar{B}$ gridsearch.

Figure 8.2.: Figure of merit after fitting for all threshold combinations in the gridsearch evaluated on $e^+e^- \to \mu^-\mu^+(\gamma)$ compared to $\Upsilon(4S) \to B\bar{B}$. The dashed line marks the region of threshold combinations with performance within statistical uncertainty of the optimum.

8.2. Integration into Full Tracking Chain

As all previous results focused on the track finding using only hits in the Central Drift Chamber (CDC), this section validates whether improving CDC-only performance also

translates to improvements when combined with the other subdetectors. This integration is necessary to validate performance on measured collision data (see Section 8.3). To analyze the real-world performance difference, 10 000 generic $\Upsilon(4S) \rightarrow B\bar{B}$ decays are simulated under realistic *high data beam background*. The performance is assessed by integrating the *neural clustering* into the full basf2 tracking chain and running this chain once with all subdetectors active and once only on the CDC.
Table 8.1 shows the integrated metrics for finding $\pi^{\pm}$ in these events. In CDC-only track finding, both *CATFinder*-approaches strongly improve the *track finding efficiency* compared to the *Baseline*. The *neural clustering* performs worse than the *spherical clustering* with lower *efficiency* and higher *fake rate*. This is mostly due to the *neural clustering* finding more hits on curling tracks, resulting in a failing fit due to the imperfect hit ordering (as described in Subsection 6.2.3). This is also shown in Figure 8.3a, where the improvements in the endcap regions are offset by a decline on curling tracks, especially on *outer curlers*. Figure 8.3b shows that with the inner tracking detectors the tracks found only in the *neural clustering* in the CDC are now also found with the *spherical clustering* in the Silicon Vertex Detector (SVD), reducing the lead. The decline in performance on curling tracks gets stronger, as the *neural clustering* finds more hits in the CDC and these predictions are removed from complementary finding steps. This also translates to a performance decline seen in the integrated metrics in Table 8.1.
This shows that in the complex pipeline of the current Belle II tracking with several feedback loops, improving a single step does not necessarily result in improvements across the full pipeline. A tracking algorithm currently under development aims to remove the feedback loops between CDC and SVD by integrating the responses from both detectors into a single multi-model GNN [23]. As this GNN is also trained with the object-condensation loss, it is expected that employing the *neural clustering* on this model will lead to direct improvements in the tracking pipeline, as a more holistical optimization will be made possible.



(a) Only CDC.

(b) Full detector.

Figure 8.3.: Difference in *track fitting charge efficiency* between *neural* and *spherical clustering* on $\pi^{\pm}$ in $\Upsilon(4S) \rightarrow B\bar{B}$ events with *high data beam background*. Red bins indicate where *neural clustering* performs better, blue bins show worse performance.

Table 8.1.: Full detector post-fitting track finding performance for different algorithms on $\pi^{\pm}$ from simulated $\Upsilon(4S) \rightarrow B\bar{B}$ decays with *high data beam background*. Uncertainties below 0.01% are not shown in the table.

Method	$\varepsilon_{\text{trk,ch}}[\%]$	$\mathfrak{r}_{\text{fake}}[\%]$	$\mathfrak{r}_{\text{clone}}[\%]$
	CDC only		
Baseline	71.27 ± 0.17	1.71 ± 0.04	0.64 ± 0.03
CAT + Sphere	75.07 ± 0.16	1.62 ± 0.04	1.6 ± 0.04
CAT + Neural	74.61 ± 0.16	2.7 ± 0.05	1.25 ± 0.04
	Full detector		
Baseline	92.03 ± 0.1	3.36 ± 0.05	3.17 ± 0.05
CAT + Sphere	92.23 ± 0.1	3.57 ± 0.05	3.9 ± 0.06
CAT + Neural	91.1 ± 0.1	4.17 ± 0.06	4.31 ± 0.06

8.3. Evaluation on Data

In all previous experiments, the clustering algorithm was optimized on simulated data, as truth information is required to evaluate the track finding performance. The following section describes validation experiments that test whether the *neural clustering* also works if applied to actual collision data.
First, it is verified if the improvements shown in Subsection 6.1.1 for prompt $\mu^{\pm}$ with high transverse momentum also hold on real collision data. Second, a short validation is performed on a more complex scenario with high multiplicity and displaced tracks.

8.3.1. Low Multiplicity Prompt Tracks

In this subsection, the performance on measured data is evaluated on $e^+e^- \rightarrow \mu^-\mu^+$ samples. These events contain only two back-to-back tracks with the same transverse momentum and are thus suited to evaluate the fitting resolution. The event selection is described in more detail in subsection 8.3.1 in [2], providing a tight restriction on events with two prompt tracks with opposite direction. This is achieved by requiring both $\mu^{\pm}$ tracks to start at the Interaction Point (IP), with a polar angle within the CDC acceptance region and both polar angles being close to 180° apart [2].

Signal Yield The signal and background yield on measured data for the different algorithms is compared using a DSCB [24] fit with no fixed parameters assuming a linear background distribution. The results are shown in Figure 8.4, indicating a slightly increased signal yield for the *CATFinder* with *neural clustering* compared to the *Baseline* and *CATFinder* with *spherical clustering*. The signal yield for all algorithms is very high, as these events are already pre-selected using a skim that employs the *Baseline* algorithm. The background yield for all algorithms is approximately zero, as very tight event selection

cuts are applied to ensure back-to-back tracks.

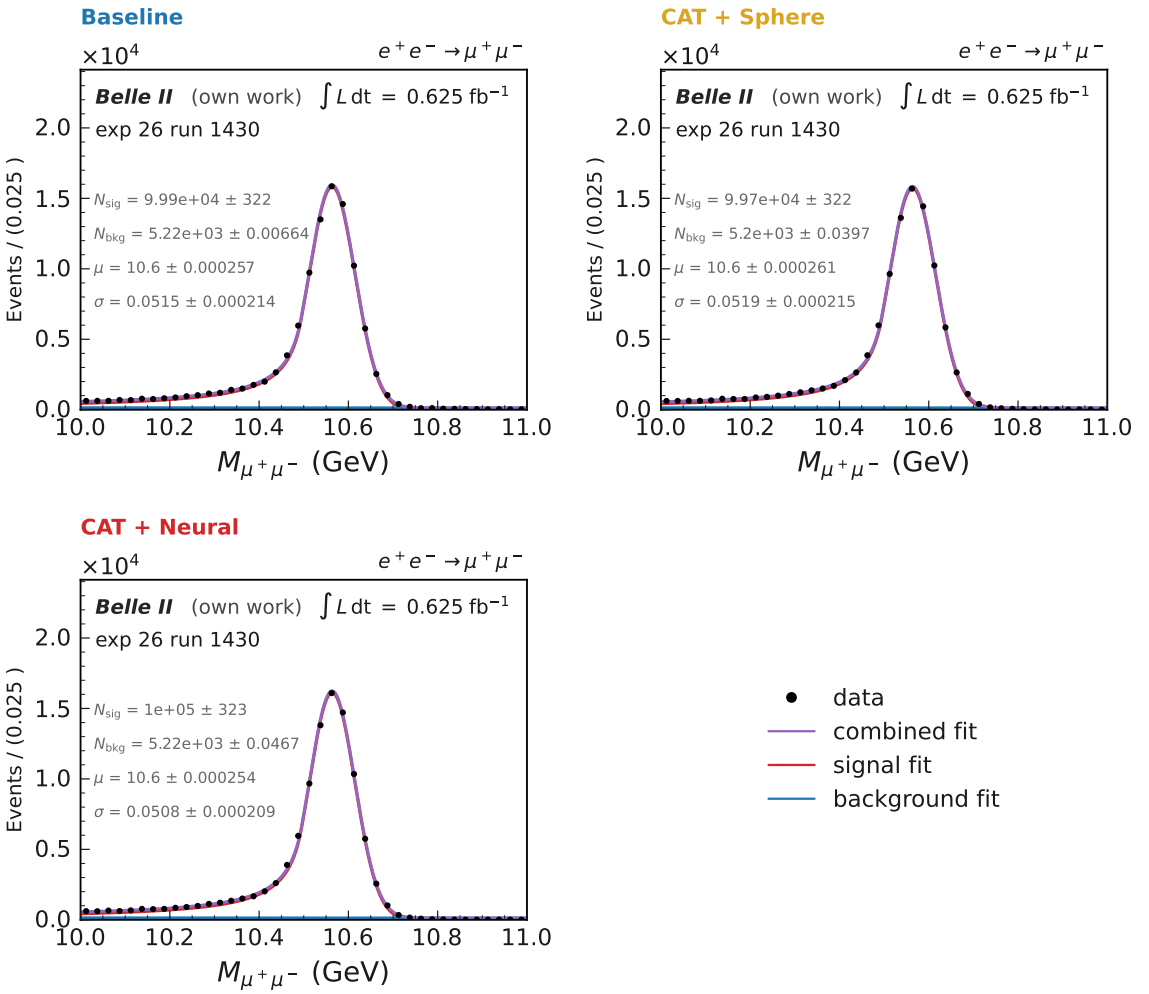
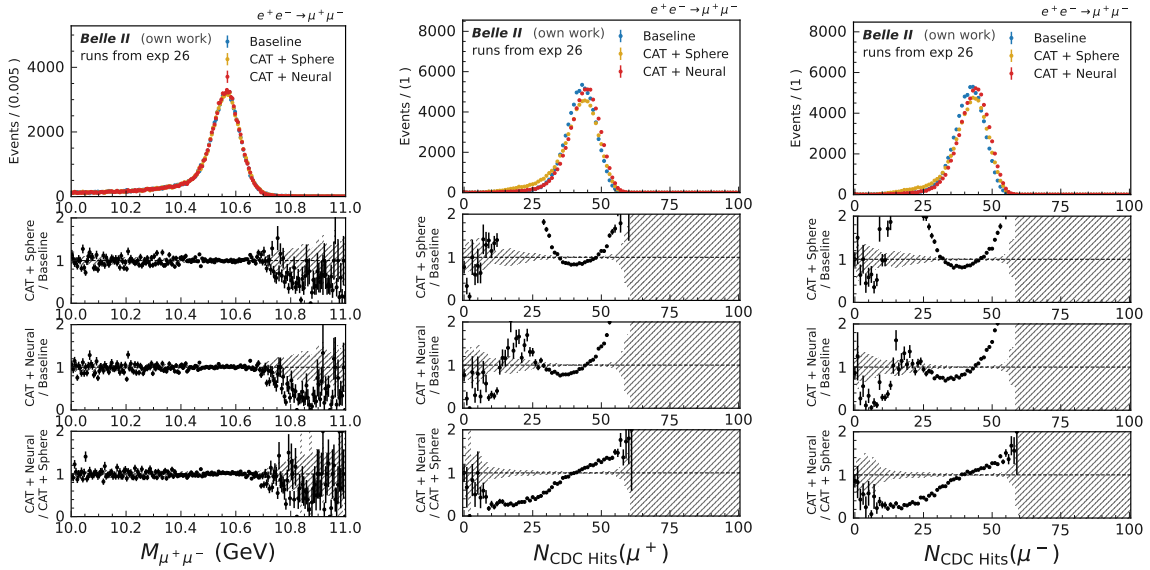


Figure 8.4.: DSCB [24]-fit for the reconstructed mass of the two $\mu^{\pm}$ in $e^+e^- \rightarrow \mu^-\mu^+$ data events with *high beam background* for different algorithms in the full reconstruction. Uncertainties are correlated between algorithms, as they are evaluated on the same events.

Hit Assignment and Momentum Resolution The number of CDC hits assigned to each track is shown in Figure 8.5. The *CATFinder* with *spherical clustering* assigns less hits to each track than the *Baseline*, while the *neural clustering* finds significantly more hits than the *spherical clustering* and also outperforms the *Baseline*. This aligns with the increased *hit efficiency* observed in Subsection 6.1.1, as the *neural clustering* also includes more hits on simulated data.
Figure 8.6 shows that this increased *hit efficiency* also leads to an improvement in the resolution of the fitted transverse momentum. The figure shows the distribution of the difference in transverse momentum for both muon tracks, which is expected to be zero.

The momentum resolution is extracted as the Full Width at Half Maximum (FWHM) of a DSCB [24] fitted to this distribution with no fixed parameters.



(a) Reconstructed mass of the two muons.

(b) Number of CDC hits in the μ^+-track.

(c) Number of CDC hits in the μ^--track.

Figure 8.5.: Comparison between different algorithms on data of $e^+e^- \to \mu^-\mu^+$ events under *high beam background* conditions for reconstructed mass (Figure 8.5a) and the number of CDC hits for the tracks of the μ^+ (Figure 8.5b) and the μ^- (Figure 8.5c).

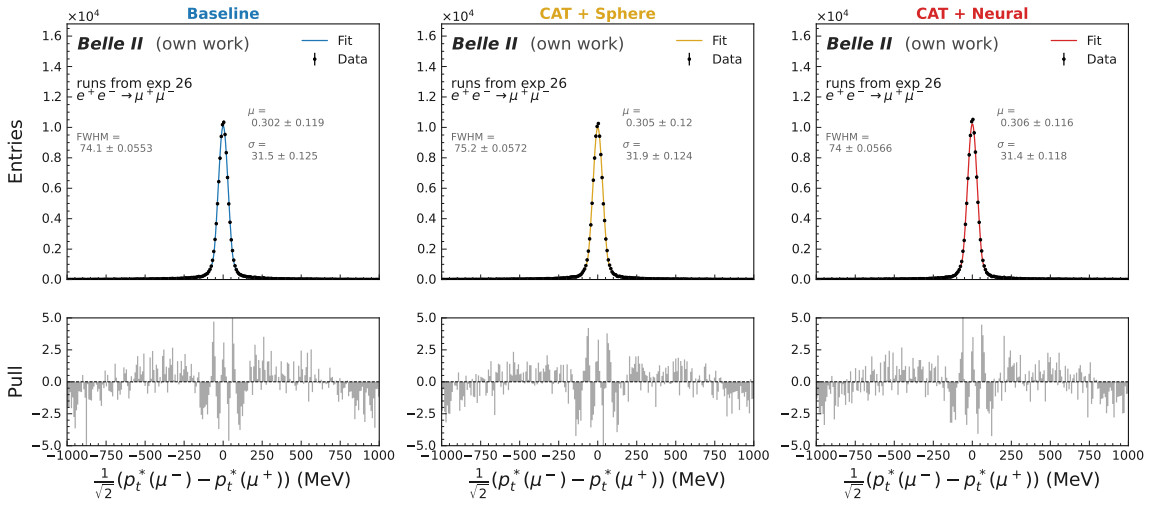


Figure 8.6.: DSCB [24]-fit for the difference in transverse momentum between the two $\mu^\pm$ in the center-of-mass frame in $e^+e^- \to \mu^-\mu^+$ data events with *high beam background* for different algorithms in the full reconstruction. Uncertainties are correlated between algorithms, as they are evaluated on the same events.

PXD and SVD Extrapolation The extrapolation of the found tracks into the inner tracking detectors is tested by comparing the resolution of the reconstructed $\mu^{\pm}$ origin. A good vertex resolution requires a successful extrapolation of the track from the CDC into the inner detectors as these provide the necessary spatial information. This resolution is evaluated using the difference between the radial (Δd_0) as well as the axial (Δz_0) vertex distances for the μ^+ and the μ^- track. As in the momentum resolution, the vertex resolution is defined as the FWHM of a DSCB [24] fit without fixed parameters on these reconstructed features.

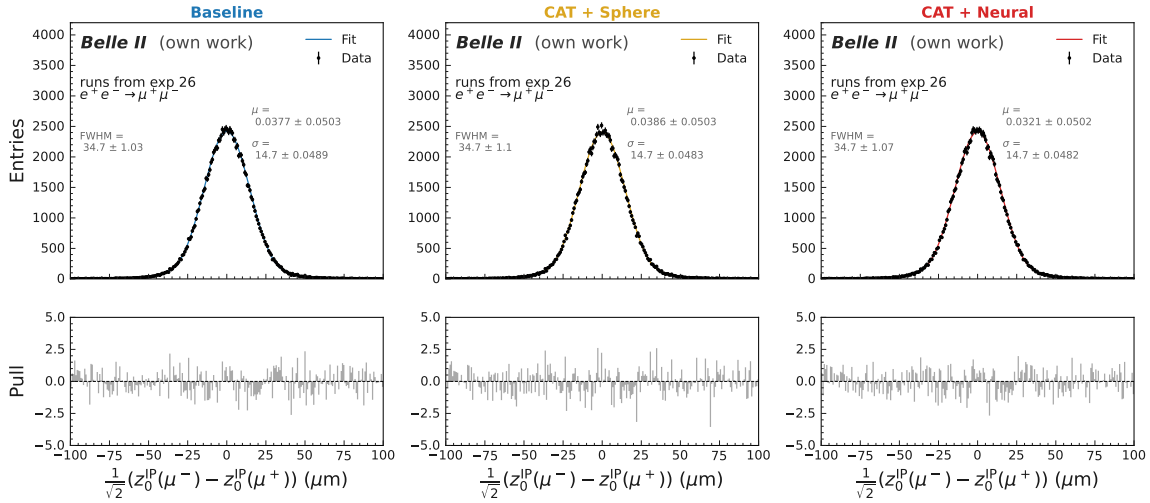


Figure 8.7.: DSCB [24]-fit for the difference in axial distance to the IP between the two $\mu^{\pm}$ in $e^+e^- \longrightarrow \mu^-\mu^+$ data events with *high beam background* for different algorithms in the full reconstruction. Uncertainties are correlated between algorithms, as they are evaluated on the same events.

Figs. 8.7 to 8.8 show no significant differences between the different track finding algorithms. This indicates that both *CATFinder* approaches allow for successful extrapolation of tracks into the inner detectors, which suggests a successful integration into the tracking chain.

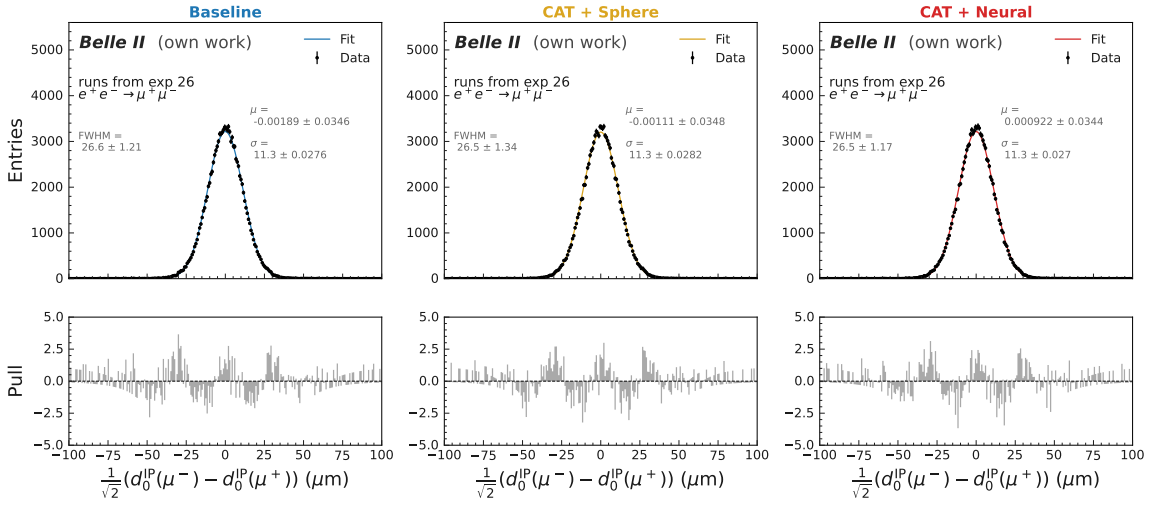


Figure 8.8.: DSCB [24]-fit for the difference in radial distance to the IP between the two $\mu^\pm$ in $e^+e^- \to \mu^-\mu^+$ data events with *high beam background* for different algorithms in the full reconstruction. Uncertainties are correlated between algorithms, as they are evaluated on the same events.

8.3.2. High Multiplicity Displaced Tracks

In the following validation experiment, the performance of the *neural clustering* is assessed on prompt and displaced pions in high multiplicity events. The decay $D^{\star+} \to D^0(\to K_s^0(\to \pi^+\pi^-)\pi^+\pi^-)\pi_S^+$ is chosen, as it poses several challenges to the track finding. The reconstruction of this decay heavily relies on the SVD and includes many curlers due to the π_S^+ from $D^{\star+} \to D^0\pi_S^+$ having a very low momentum. The event selection and decay reconstruction is described in more detail in section 8.2 of [2]. It ensures high selection purity and results in the largest contribution stemming from $e^+e^- \to c\bar{c}$ events.

Reconstructed K_S^0 Mass The mass of the K_S^0 is reconstructed for each event and to this distribution, a DSCB [24] function is fitted for the signal contribution, while a first order Chebyshev polynomial is used for background. For each algorithm, the approximate significance is calculated as $N_{\text{sig}}/\sqrt{N_{\text{bkg}}}$.
Figure 8.10 shows the mass distribution of K_S^0-candidates in the signal region. The comparison indicates an increase in the count of candidates for the *CATFinder* algorithms compared to the *Baseline*.
The fit results are shown in Figure 8.9. The *Baseline* provides both the lowest signal and background yield, reaching a significance of 48.9 ± 2.5. The *CATFinder* with *spherical clustering* performs best with a significance of 52.3 ± 2.8 due to its high signal yield. In this evaluation, the *CATFinder* with *neural clustering* performs worst with a signal yield close to the *Baseline* but significantly higher background yield, reaching a significance of 45.6 ± 2.8. This is expected, as the analysis on simulated data already showed worse performance on events containing curling tracks due to hit-ordering issues (see Subsection 6.1.2).

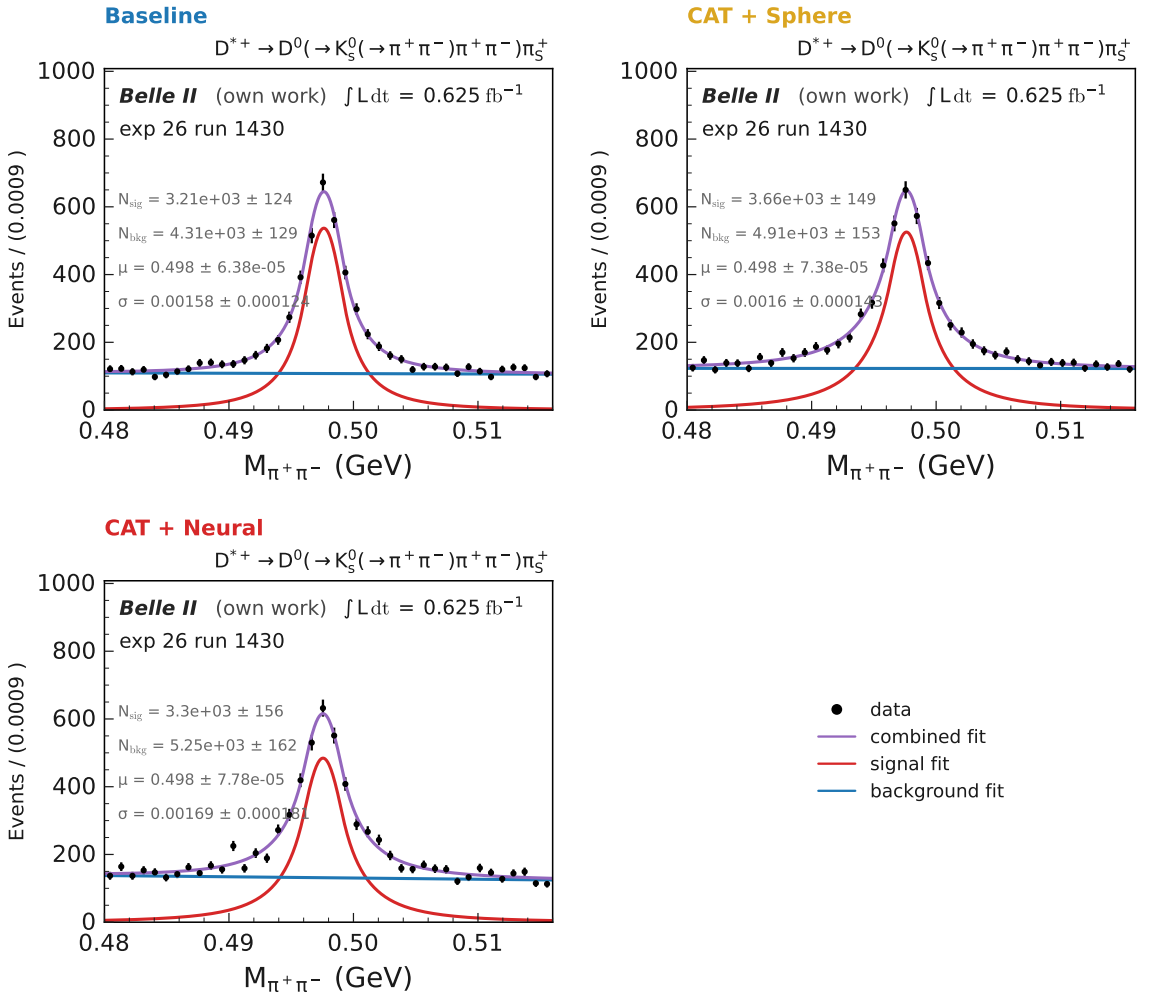


Figure 8.9.: DSCB [24]-fit for the reconstructed mass of the K_S^+ in $D^{\star+} \to D^0(\to K^0_s(\to \pi^+\pi^-)\pi^+\pi^-)\pi^+_S$ data events with *high beam background* for different algorithms in the full reconstruction. Uncertainties are correlated between algorithms, as they are evaluated on the same events.

Hit Assignment Figs. 8.11 to 8.12 show that the *CATFinder* with *neural clustering* assigns similar numbers of hits to each track as the *Baseline*, while the *CATFinder* with *spherical clustering* yields significantly lower numbers of CDC-hits per track. This indicates that the *neural clustering* proves more flexible than the *spherical clustering*, adapting to a latent space that is less clearly separated into clusters as on simulated data. The *spherical clustering* outperforms the *neural clustering* only for the slow pion, which again indicates that the *neural clustering* struggles with curling tracks, expressing the need for a more sophisticated hit ordering.

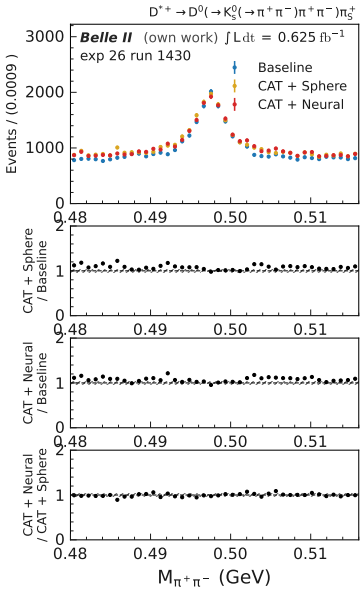


Figure 8.10.: Distribution of the reconstructed mass of the K^+_S-candidates in $D^{\star+} \to D^0(\to K^0_s(\to \pi^+\pi^-)\pi^+\pi^-)\pi^+_S$ data events with *high beam background* for different algorithms in the full reconstruction. Uncertainties are correlated between algorithms, as they are evaluated on the same events.

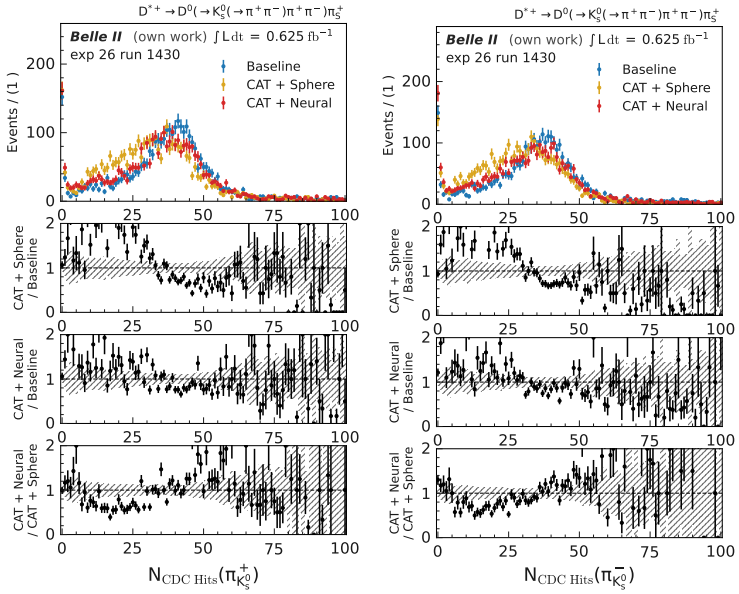


Figure 8.11.: Comparison of the number of hits assigned to the track of the displaced pions in data $D^{\star+} \to D^0(\to K^0_s(\to \pi^+\pi^-)\pi^+\pi^-)\pi^+_S$ decays by different track finding algorithms. Evaluated on data from experiment 26 with *high beam background*. Uncertainties are correlated between algorithms, as they are evaluated on the same events.

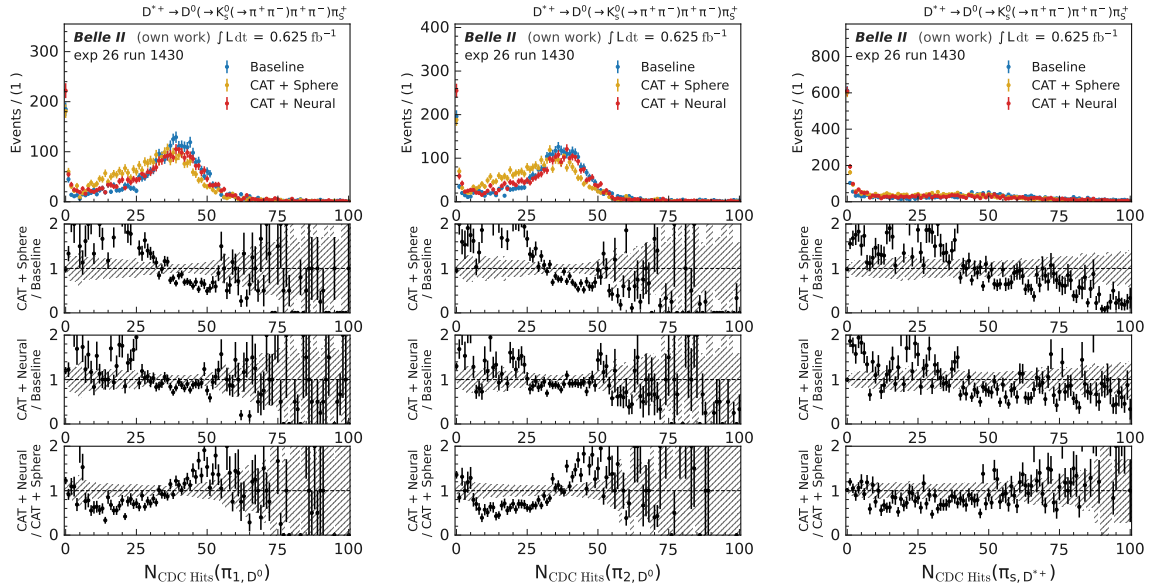


Figure 8.12.: Comparison of the number of hits assigned to the track of the prompt pions in data $D^{\star+} \longrightarrow D^0(\longrightarrow K^0_s(\longrightarrow \pi^+\pi^-)\pi^+\pi^-)\pi^+_S$ decays by different track finding algorithms. Evaluated on data from experiment 26 with *high beam background*. Uncertainties are correlated between algorithms, as they are evaluated on the same events.

Overall, the evaluation on measured data shows that the *neural clustering* generalizes well from simulated to real data. The evaluations described in Chapter 6 are thus expected to be similar to the performance on actual collision data.

9. Conclusion

This thesis presents the successful implementation, evaluation and validation of a novel latent space clustering algorithm that leverages a neural inference for non-iterative finding of a variable number of clusters.
The proposed clustering pipeline is built around a small neural network, which is trained on the *CATFinder*-GNN latent space predictions from diverse simulated events with high detector background. The architectural challenge of handling unordered, variable-length inputs and finding a variable number of clusters is addressed by combining key features from You Only Look Once (YOLO) [17] and PointNet [16].
Two complementary samples are evaluated to determine the optimal working point, demonstrating good hyperparameter generalization. The range of hyperparameters performing within statistical uncertainty of the optimum for challenging events form a subset of those for simpler event signatures.
First, the evaluation for tracks which leave the Belle II Central Drift Chamber (CDC) through the backward endcap, shows that the *neural clustering* improves *track fitting charge efficiency* from 90.5% to 91.9%, when the *spherical clustering* is replaced with *neural clustering*. For events with high track multiplicity containing the decay of a hypothetical long-lived massive particle, which decays at a random point in the CDC, *track fitting charge efficiency* increases by 3.5 percentage points. This comes at the cost of a decrease of 1.1 percentage points in *track fitting purity*.
Second, for curling tracks, *neural clustering* improves *hit efficiency* from 85.4% (*spherical*) to 90.7% (*neural*) at the cost of a 1 percentage point drop in *hit purity*. However, this better assignment results in a lower *track charge efficiency* after fitting, because the fitting step requires a sorted list of hits and the currently implemented hit-ordering algorithm does not generalize to curling tracks.
Integration into the full tracking chain and evaluation on real collision data demonstrates the potential of the proposed algorithm. On data, the number of hits assigned to each track is increased, resulting in an enhanced momentum resolution of 74.0 MeV with *neural clustering* compared to the 75.2 MeV when using *spherical clustering*. The *neural clustering* is shown to generalize to the more complex latent space topologies left by events on which the *CATFinder*-GNN does not match the simulated performance. This indicates a more robust hits-to-tracks assignment, significantly improving momentum resolution compared to *CATFinder* with *spherical clustering* and outperforming the currently used *Baseline* [5].

Because the proposed clustering algorithm is neural-network based, it allows gradients to propagate through the clustering step, enabling joint fine-tuning of the clustering network and the *CATFinder*-GNN. This is already implemented as a proof-of-concept, yielding additional increases in *efficiency* and *purity* on high-multiplicity events for both low-momentum and displaced tracks.

Further improvements are expected by combining the *neural clustering* with the BATFinder-GNN [23], which includes the hits from the Silicon Vertex Detector (SVD) into the *CATFinder*-pipeline.
Due to its scalable design, the *neural clustering* can also handle high-dimensional input spaces formed by including additional input features or by increasing the cluster space dimension. The feasibility of both directions is already probed in this work, indicating that this is a promising area for further research. As the information content in a low-dimensional cluster space is limited, adding skip-connections to earlier *CATFinder*-GNN layers could enhance the information available for clustering decisions, and enable a more fine-grained hit assignment.
An important focus of further research should be the development of a new hit-ordering algorithm that is able to handle curling tracks, because improving the hit assignment currently leads to a worse ordering performance on these tracks.
Inference latency can be significantly reduced by implementing the full algorithm in `C++` and porting the neural network to `ONNX`. This will be important for including the *neural clustering* in the official basf2 tracking pipeline and enhancing the actual event reconstruction in Belle II.

Overall, this new clustering algorithm advances track finding at Belle II, while providing several promising starting points for further improvements.

Bibliography

[1] L. Reuter et al. End-to-End Multi-track Reconstruction Using Graph Neural Networks at Belle II. *Computing and Software for Big Science*, 9(1):6, December 2025.

[2] Lea Reuter. *Track Finding with Graph Neural Networks in the Belle II Drift Chamber*. PhD thesis, Karlsruher Institut für Technologie (KIT), 2026.

[3] E Kou et al. The Belle II Physics Book. *Progress of Theoretical and Experimental Physics*, 2019(12):123C01, December 2019.

[4] Jan Kieseler. Object condensation: One-stage grid-free multi-object reconstruction in physics detectors, graph, and image data. *The European Physical Journal C*, 80(9):886, September 2020.

[5] Valerio Bertacchi et al. Track finding at Belle II. *Computer Physics Communications*, 259:107610, February 2021.

[6] T. Abe et al. Belle II Technical Design Report, November 2010.

[7] Kazunori Akai, Kazuro Furukawa, and Haruyo Koiso. SuperKEKB collider. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 907:188–199, November 2018.

[8] Dmitry Matvienko. The Belle II experiment: Status and physics program. *EPJ Web of Conferences*, 191:02010, 2018.

[9] C. Höppner, S. Neubert, B. Ketzer, and S. Paul. A novel generic framework for track fitting in complex detector systems. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 620(2-3):518–525, August 2010.

[10] Shah Rukh Qasim, Jan Kieseler, Yutaro Iiyama, and Maurizio Pierini. Learning representations of irregular particle-detector geometry with distance-weighted graph networks. *The European Physical Journal C*, 79(7):608, July 2019.

[11] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96)*, pages 226–231, 1996.

[12] Kilian Lieret and Gage DeZoort. An Object Condensation Pipeline for Charged Particle Tracking at the High Luminosity LHC. *EPJ Web of Conferences*, 295:09004, 2024.

[13] Alex Rodriguez and Alessandro Laio. Clustering by fast search and find of density peaks. *Science*, 344(6191):1492–1496, June 2014.

[14] Dolores Garcia, Lena Herrmann, Gregor Krzmanc, and Michele Selvaggi. End-to-end event reconstruction for precision physics at future colliders, 2026.

[15] Yizhang Wang, Di Wang, You Zhou, Xiaofeng Zhang, and Chai Quek. VDPC: Variational density peak clustering algorithm. *Information Sciences*, 621:627–651, April 2023.

[16] R. Qi Charles, Hao Su, Mo Kaichun, and Leonidas J. Guibas. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 77–85, Honolulu, HI, July 2017. IEEE.

[17] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You Only Look Once: Unified, Real-Time Object Detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, Las Vegas, NV, USA, June 2016. IEEE.

[18] J. Allison et al. Recent developments in Geant4. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 835:186–225, November 2016.

[19] S. Jadach, B.F.L. Ward, and Z. Wąs. The precision Monte Carlo event generator for two-fermion final states in collisions. *Computer Physics Communications*, 130(3):260–325, August 2000.

[20] G. Rodrigo, H. Czyż, J.H. Kühn, and M. Szopa. Radiative return at NLO and the measurement of the hadronic cross-section in electron–positron annihilation. *The European Physical Journal C*, 24(1):71–82, May 2002.

[21] David J. Lange. The EvtGen particle decay simulation package. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 462(1-2):152–155, April 2001.

[22] Lars Buitinck et al. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122, 2013.

[23] Lea Reuter, Tristan Brandes, Giacomo De Pietro, and Torben Ferber. Multi-Modal Track Reconstruction using Graph Neural Networks at Belle II, 2026.

[24] J. E. Gaiser et al. Charmonium spectroscopy from inclusive ψ' and $\mathcal{J}\psi$ radiative decays. *Physical Review D*, 34(3):711–721, August 1986.

[25] The Belle II Collaboration. Belle II Analysis Software Framework (basf2). Zenodo, July 2025.

[26] T. Kuhr, C. Pulvermacher, M. Ritter, T. Hauth, and N. Braun. The Belle II Core Software: Belle II Framework Software Group. *Computing and Software for Big Science*, 3(1):1, December 2019.

[27] J. Allison et al. Geant4 developments and applications. *IEEE Transactions on Nuclear Science*, 53(1):270–278, February 2006.

[28] S. Agostinelli et al. Geant4—a simulation toolkit. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 506(3):250–303, July 2003.

A. Appendix

A.1. Clustering Algorithm

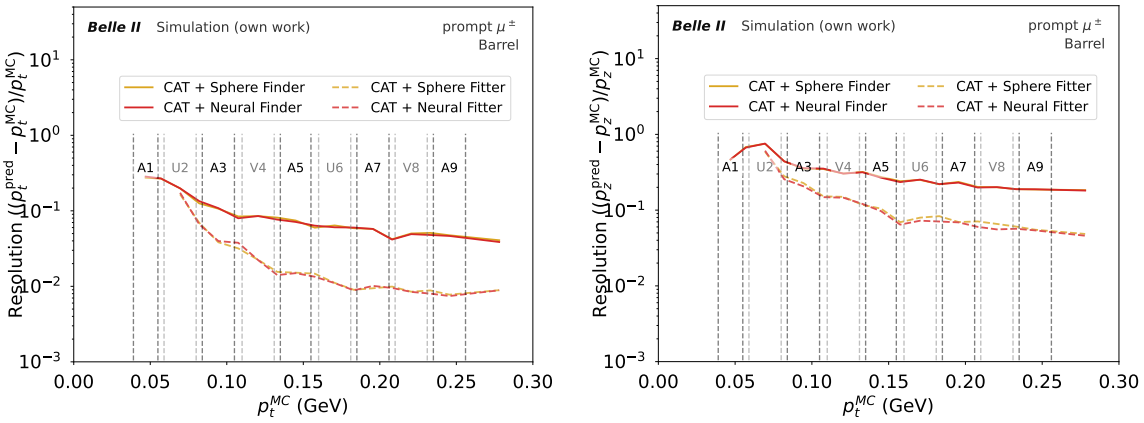
Algorithm 1 Point-cloud clustering pipeline

Require: Point cloud $P = \{(\vec{r}_i, \beta_i)\}_{i=1}^{N}$, condensation threshold τ_β, number of neighbors k, neighbor score threshold τ_{p}, fusion threshold τ_{IoU}
Ensure: Final set of clusters C_{final}

// Step 1: Select condensation points
$\mathcal{S} \leftarrow \{\, i \in \{1, \ldots, N\} \mid \beta_i > \tau_\beta \,\}$
// Step 2: Predict one candidate cluster per condensation point
$C_{\mathrm{cand}} \leftarrow \emptyset$
for all $s \in \mathcal{S}$ **in parallel do**
    $N_s \leftarrow k$ nearest neighbors of point s in P
    $\tilde{X}_s \leftarrow \{\, \vec{r}_j - \vec{r}_s \mid j \in N_s \,\}$ ▷ relative coordinates
    $\{p_j\}_{j \in N_s} \leftarrow f_\theta(\tilde{X}_s)$ ▷ neighbor classifier
    $C_s \leftarrow \{\, j \in N_s \mid p_j > \tau_{\mathrm{p}} \,\}$
    $C_s \leftarrow C_s \cup \{s\}$ ▷ ensure seed point is contained
    store candidate cluster (C_s, β_s) in C_{cand}
end for
// Step 3: Sort candidate clusters by seed confidence
Sort C_{cand} in descending order of seed score β_s
// Step 4: Fuse overlapping candidate clusters
$C_{\mathrm{final}} \leftarrow \emptyset$
for all (C, β) in C_{cand} **do**
    merged $\leftarrow$ **false**
    for all $C' \in C_{\mathrm{final}}$ **do**
        $\mathrm{IoU}(C, C') \leftarrow \dfrac{|C \cap C'|}{|C \cup C'|}$
        if $\mathrm{IoU}(C, C') > \tau_{\mathrm{IoU}}$ **then**
            $C' \leftarrow C' \cup C$ ▷ fuse overlapping predictions
            merged $\leftarrow$ **true**
            break
        end if
    end for
    if not merged **then**
        $C_{\mathrm{final}} \leftarrow C_{\mathrm{final}} \cup \{C\}$
    end if
end for
return C_{final}

A.2. Detailed Performance Metrics

A.2.1. Low Transverse Momentum



(a) Transverse momentum resolution.

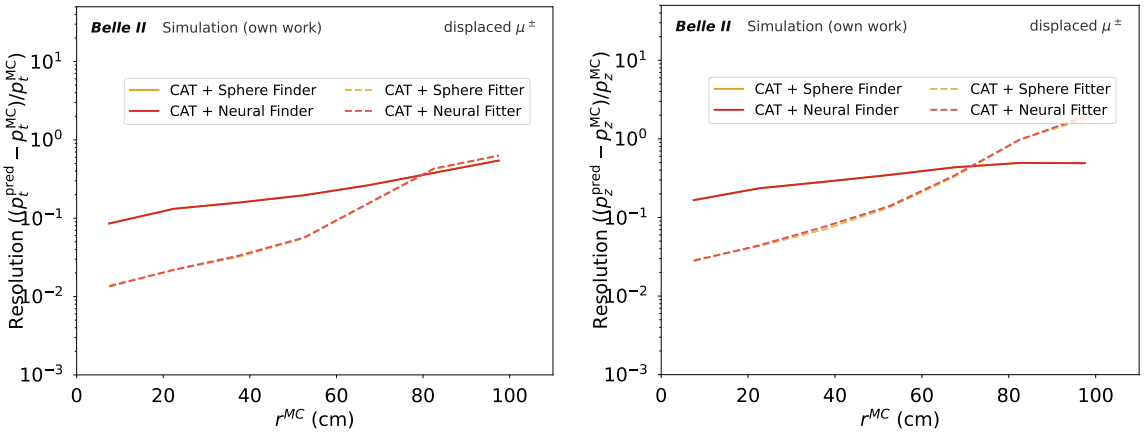
(b) Axial momentum resolution.

Figure A.1.: Momentum resolution before (solid lines) and after (dashed lines) track fitting for $\mu^\pm$ with low p_t samples with *high data beam background* for different algorithms limited to the intersecting sample in the barrel region.

Table A.1.: The performance metrics for $\mu^{\pm}$ with low p_t samples for different track finding algorithms in different detector regions for *high data beam background*. Uncertainties below 0.01% are not shown in the table, uncertainties are correlated between different methods.

Method	$\varepsilon_{\text{trk}}[\%]$	$\mathfrak{r}_{\text{fake}}[\%]$	$\mathfrak{r}_{\text{clone}}[\%]$	$\varepsilon_{\text{trk,ch}}[\%]$	$\mathfrak{r}_{\text{wrong ch.}}[\%]$
			FWD		
Baseline Finder	$74.8^{+0.8}_{-0.8}$	$2.8^{+0.4}_{-0.4}$	$0.05^{+0.04}_{-0.07}$	$74.8^{+0.8}_{-0.8}$	$0.0_{-0.06}$
CAT + Sphere Finder	$96.4^{+0.4}_{-0.3}$	$21.4^{+0.8}_{-0.8}$	$1.0^{+0.2}_{-0.2}$	$95.8^{+0.4}_{-0.4}$	$0.5^{+0.1}_{-0.2}$
CAT + Neural Finder	$96.8^{+0.3}_{-0.3}$	$25.5^{+0.9}_{-0.9}$	$0.7^{+0.1}_{-0.2}$	$96.3^{+0.4}_{-0.3}$	$0.6^{+0.1}_{-0.2}$
Baseline Fitter	$70.9^{+0.9}_{-0.9}$	$2.7^{+0.4}_{-0.4}$	$0.05^{+0.04}_{-0.08}$	$69.8^{+0.9}_{-0.9}$	$1.6^{+0.3}_{-0.3}$
CAT + Sphere Fitter	$82.8^{+0.7}_{-0.7}$	$2.2^{+0.3}_{-0.3}$	$0.17^{+0.07}_{-0.1}$	$81.5^{+0.7}_{-0.7}$	$1.7^{+0.3}_{-0.3}$
CAT + Neural Fitter	$83.1^{+0.7}_{-0.7}$	$2.7^{+0.3}_{-0.4}$	$0.09^{+0.05}_{-0.08}$	$81.8^{+0.7}_{-0.7}$	$1.5^{+0.2}_{-0.3}$
			BRL		
Baseline Finder	$87.7^{+0.3}_{-0.3}$	$5.7^{+0.2}_{-0.2}$	$4.4^{+0.2}_{-0.2}$	$73.9^{+0.4}_{-0.4}$	$15.8^{+0.3}_{-0.4}$
CAT + Sphere Finder	$99.12^{+0.09}_{-0.08}$	$13.5^{+0.3}_{-0.3}$	$7.8^{+0.2}_{-0.2}$	$92.5^{+0.2}_{-0.2}$	$6.7^{+0.2}_{-0.2}$
CAT + Neural Finder	$99.26^{+0.08}_{-0.07}$	$16.2^{+0.3}_{-0.3}$	$4.6^{+0.2}_{-0.2}$	$92.8^{+0.2}_{-0.2}$	$6.5^{+0.2}_{-0.2}$
Baseline Fitter	$81.6^{+0.4}_{-0.3}$	$4.3^{+0.2}_{-0.2}$	$3.7^{+0.2}_{-0.2}$	$72.1^{+0.4}_{-0.4}$	$11.7^{+0.3}_{-0.3}$
CAT + Sphere Fitter	$79.9^{+0.4}_{-0.4}$	$3.4^{+0.2}_{-0.2}$	$2.5^{+0.2}_{-0.2}$	$77.2^{+0.4}_{-0.4}$	$3.3^{+0.2}_{-0.2}$
CAT + Neural Fitter	$77.8^{+0.4}_{-0.4}$	$4.4^{+0.2}_{-0.2}$	$0.96^{+0.1}_{-0.1}$	$75.3^{+0.4}_{-0.4}$	$3.2^{+0.2}_{-0.2}$
			BWD		
Baseline Finder	$63.7^{+0.8}_{-0.8}$	$5.5^{+0.5}_{-0.5}$	$0.0_{-0.05}$	$63.6^{+0.8}_{-0.8}$	$0.21^{+0.08}_{-0.11}$
CAT + Sphere Finder	$94.6^{+0.4}_{-0.4}$	$15.9^{+0.6}_{-0.6}$	$0.8^{+0.1}_{-0.2}$	$93.9^{+0.4}_{-0.4}$	$0.8^{+0.1}_{-0.2}$
CAT + Neural Finder	$95.7^{+0.3}_{-0.3}$	$22.2^{+0.6}_{-0.6}$	$0.5^{+0.1}_{-0.1}$	$94.9^{+0.4}_{-0.4}$	$0.9^{+0.1}_{-0.2}$
Baseline Fitter	$60.4^{+0.8}_{-0.8}$	$4.8^{+0.4}_{-0.5}$	$0.0_{-0.05}$	$59.1^{+0.8}_{-0.8}$	$2.3^{+0.3}_{-0.3}$
CAT + Sphere Fitter	$76.6^{+0.7}_{-0.7}$	$6.3^{+0.5}_{-0.5}$	$0.24^{+0.08}_{-0.1}$	$74.0^{+0.7}_{-0.7}$	$3.4^{+0.3}_{-0.3}$
CAT + Neural Fitter	$77.1^{+0.7}_{-0.7}$	$7.7^{+0.5}_{-0.5}$	$0.07^{+0.04}_{-0.06}$	$74.1^{+0.7}_{-0.7}$	$3.8^{+0.3}_{-0.4}$

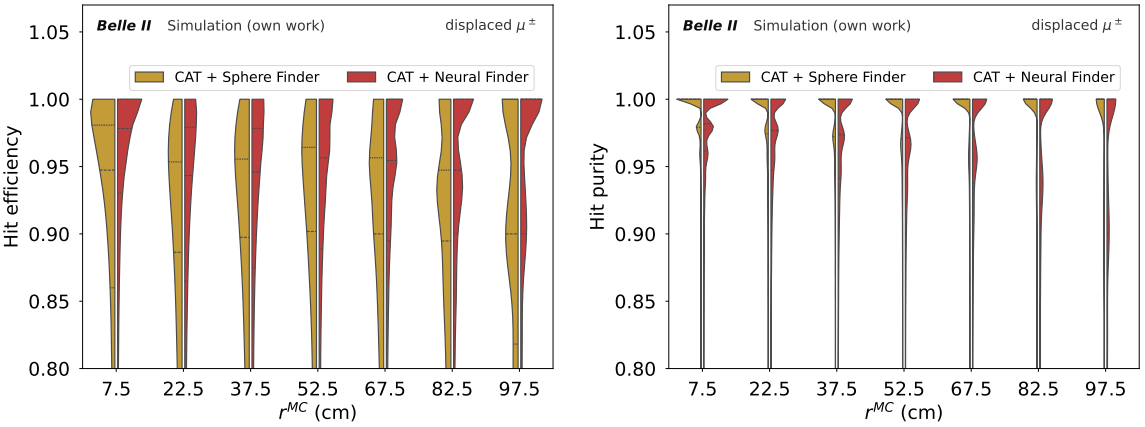
A.2.2. Low Multiplicity Displaced



(a) Transverse momentum resolution.

(b) Axial momentum resolution.

Figure A.2.: Momentum resolution before (solid lines) and after (dashed lines) track fitting for $\mu^\pm$ in displaced $\mu^\pm$ samples with *high data beam background* for different algorithms limited to the intersecting sample.

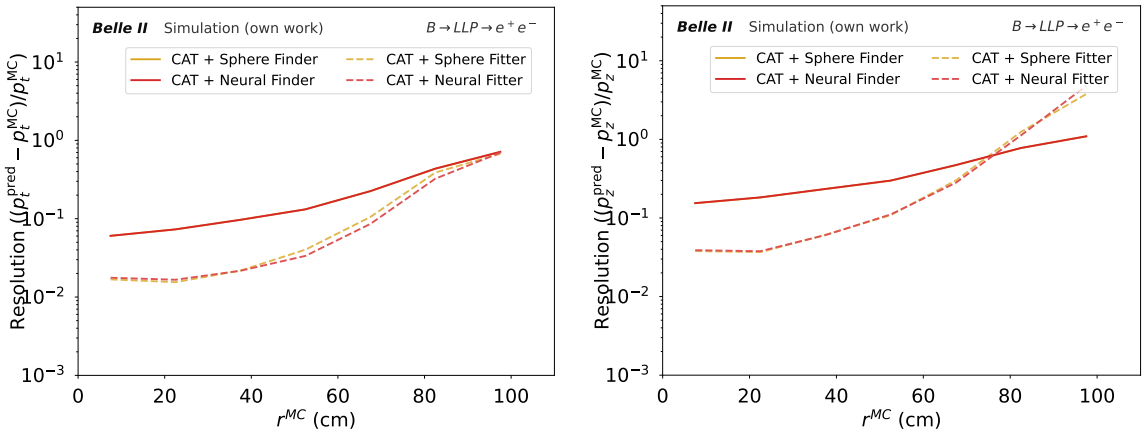


(a) *Hit efficiency*.

(b) *Hit purity*.

Figure A.3.: *Hit efficiency* and *purity* before track fitting for $\mu^\pm$ in displaced $\mu^\pm$ samples with *high data beam background* for different algorithms limited to the intersecting sample. Values are binned by displacement, the x-axis shows the bin centers.

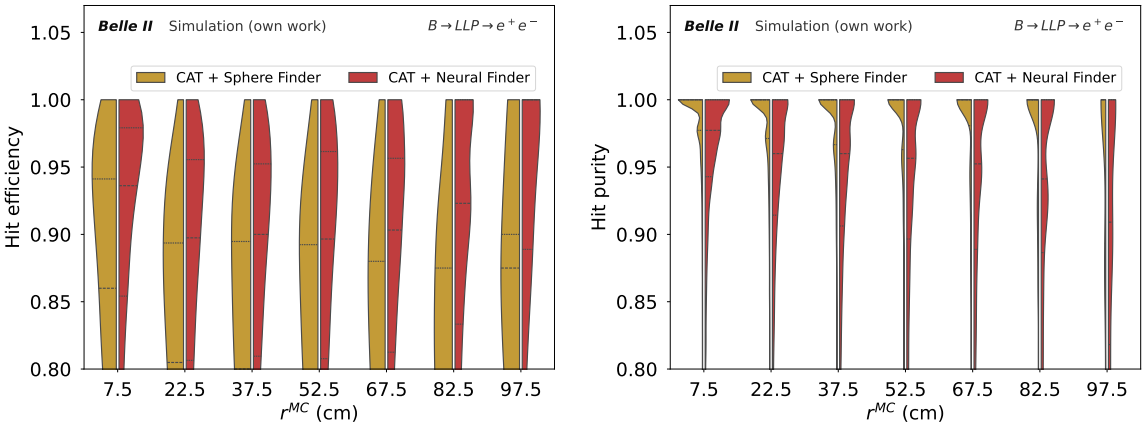
A.2.3. High Multiplicity Displaced



(a) Transverse momentum resolution.

(b) Axial momentum resolution.

Figure A.4.: Momentum resolution by decay vertex displacement before (solid lines) and after (dashed lines) track fitting for $B \to LLP \to e^+e^-$ samples with *high data beam background* for different algorithms limited to the intersecting sample.



(a) *Hit efficiency*.

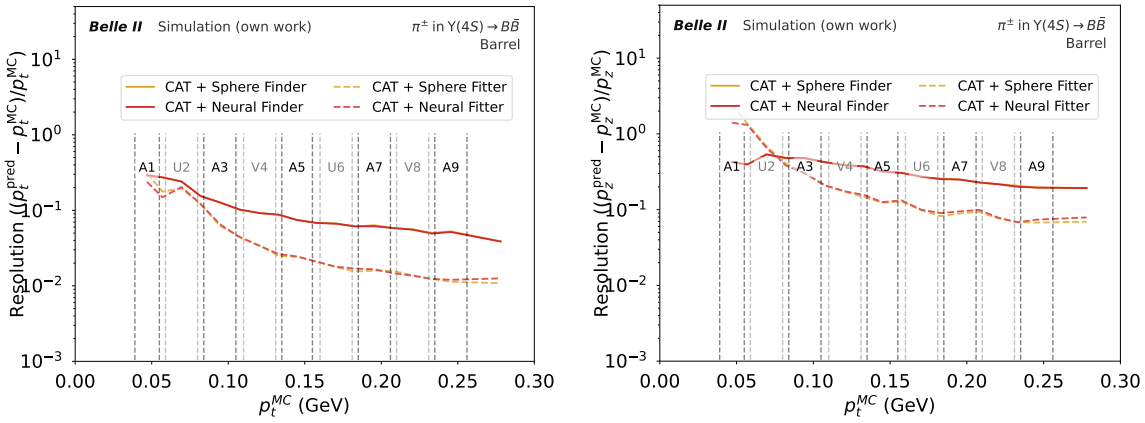
(b) *Hit purity*.

Figure A.5.: *Hit efficiency* and *purity* before track fitting for $B \to LLP \to e^+e^-$ samples with *high data beam background* for different algorithms limited to the intersecting sample. Values are binned by displacement, the x-axis shows the bin centers.

A.2.4. High Multiplicity Low Momentum

Table A.2.: The performance metrics for $\Upsilon(4S) \longrightarrow B\bar{B}$ samples for different track finding algorithms in different detector regions for *high data beam background*. Uncertainties below 0.01% are not shown in the table, uncertainties are correlated between different methods.

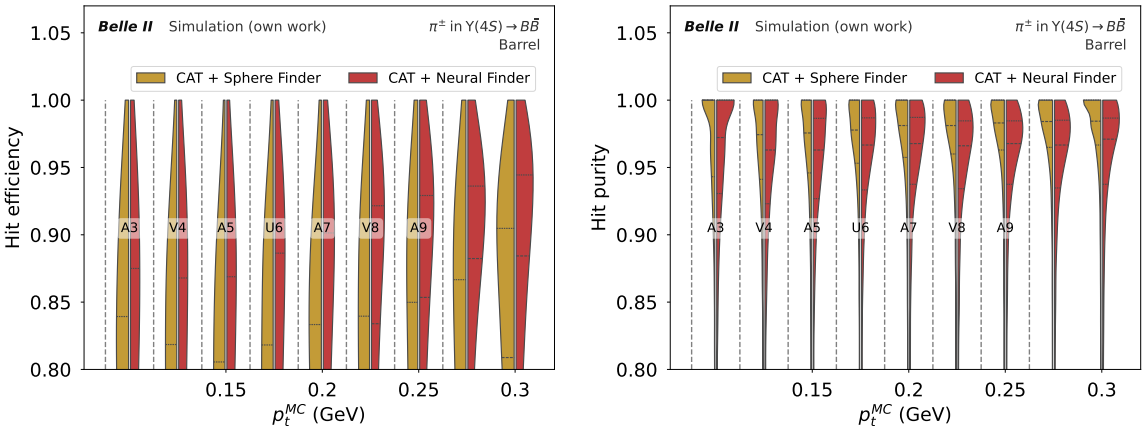
Method	$\varepsilon_{\text{trk}}[\%]$	$\mathfrak{r}_{\text{fake}}[\%]$	$\mathfrak{r}_{\text{clone}}[\%]$	$\varepsilon_{\text{trk,ch}}[\%]$	$\mathfrak{r}_{\text{wrong ch.}}[\%]$
			FWD		
Baseline Finder	$61.0^{+0.2}_{-0.2}$	$13.3^{+0.1}_{-0.1}$	$0.05^{+0.01}_{-0.01}$	$60.9^{+0.2}_{-0.2}$	$0.11^{+0.01}_{-0.01}$
CAT + Sphere Finder	$85.3^{+0.3}_{-0.3}$	$24.4^{+0.4}_{-0.4}$	$1.47^{+0.09}_{-0.1}$	$84.7^{+0.3}_{-0.3}$	$0.68^{+0.06}_{-0.07}$
CAT + Neural Finder	$85.9^{+0.3}_{-0.3}$	$27.0^{+0.4}_{-0.4}$	$1.41^{+0.09}_{-0.09}$	$85.3^{+0.3}_{-0.3}$	$0.73^{+0.06}_{-0.07}$
Baseline Fitter	$56.4^{+0.2}_{-0.2}$	$12.2^{+0.1}_{-0.1}$	$0.03^{+0.01}_{-0.01}$	$55.4^{+0.2}_{-0.2}$	$1.78^{+0.06}_{-0.06}$
CAT + Sphere Fitter	$68.2^{+0.3}_{-0.3}$	$15.7^{+0.3}_{-0.3}$	$0.36^{+0.05}_{-0.06}$	$66.5^{+0.3}_{-0.3}$	$2.5^{+0.1}_{-0.1}$
CAT + Neural Fitter	$69.5^{+0.3}_{-0.3}$	$17.4^{+0.3}_{-0.3}$	$0.32^{+0.05}_{-0.05}$	$67.7^{+0.3}_{-0.3}$	$2.6^{+0.1}_{-0.1}$
			BRL		
Baseline Finder	$86.02^{+0.05}_{-0.05}$	$15.98^{+0.05}_{-0.05}$	$0.81^{+0.01}_{-0.01}$	$81.87^{+0.05}_{-0.05}$	$4.82^{+0.03}_{-0.03}$
CAT + Sphere Finder	$92.46^{+0.08}_{-0.08}$	$27.6^{+0.1}_{-0.1}$	$3.84^{+0.06}_{-0.06}$	$90.34^{+0.09}_{-0.09}$	$2.29^{+0.05}_{-0.05}$
CAT + Neural Finder	$90.94^{+0.09}_{-0.09}$	$29.4^{+0.1}_{-0.1}$	$2.4^{+0.05}_{-0.05}$	$88.9^{+0.09}_{-0.09}$	$2.24^{+0.05}_{-0.05}$
Baseline Fitter	$83.78^{+0.05}_{-0.05}$	$14.72^{+0.05}_{-0.05}$	$0.62^{+0.01}_{-0.01}$	$80.96^{+0.05}_{-0.05}$	$3.36^{+0.03}_{-0.03}$
CAT + Sphere Fitter	$83.2^{+0.1}_{-0.1}$	$18.3^{+0.1}_{-0.1}$	$0.98^{+0.03}_{-0.03}$	$82.2^{+0.1}_{-0.1}$	$1.31^{+0.04}_{-0.04}$
CAT + Neural Fitter	$81.4^{+0.1}_{-0.1}$	$19.4^{+0.1}_{-0.1}$	$0.49^{+0.02}_{-0.02}$	$80.4^{+0.1}_{-0.1}$	$1.26^{+0.04}_{-0.04}$
			BWD		
Baseline Finder	$54.6^{+0.2}_{-0.2}$	$20.5^{+0.2}_{-0.2}$	$0.04^{+0.01}_{-0.01}$	$54.5^{+0.2}_{-0.2}$	$0.27^{+0.03}_{-0.03}$
CAT + Sphere Finder	$79.9^{+0.3}_{-0.3}$	$24.4^{+0.4}_{-0.4}$	$1.03^{+0.09}_{-0.1}$	$79.2^{+0.3}_{-0.3}$	$0.86^{+0.08}_{-0.09}$
CAT + Neural Finder	$81.9^{+0.3}_{-0.3}$	$27.6^{+0.4}_{-0.4}$	$0.87^{+0.08}_{-0.09}$	$81.1^{+0.3}_{-0.3}$	$0.89^{+0.08}_{-0.09}$
Baseline Fitter	$51.5^{+0.2}_{-0.2}$	$17.9^{+0.2}_{-0.2}$	$0.02^{+0.01}_{-0.01}$	$50.7^{+0.2}_{-0.2}$	$1.6^{+0.06}_{-0.07}$
CAT + Sphere Fitter	$63.2^{+0.4}_{-0.4}$	$23.2^{+0.4}_{-0.4}$	$0.26^{+0.05}_{-0.06}$	$61.0^{+0.4}_{-0.4}$	$3.6^{+0.2}_{-0.2}$
CAT + Neural Fitter	$66.0^{+0.4}_{-0.4}$	$25.0^{+0.4}_{-0.4}$	$0.17^{+0.04}_{-0.05}$	$63.6^{+0.4}_{-0.4}$	$3.7^{+0.2}_{-0.2}$



(a) Transverse momentum resolution.

(b) Axial momentum resolution.

Figure A.6.: Momentum resolution before (solid lines) and after (dashed lines) track fitting for $\pi^\pm$ in $\Upsilon(4S) \to B\bar{B}$ samples with *high data beam background* for different algorithms limited to the intersecting sample in the barrel region.



(a) *Hit efficiency*.

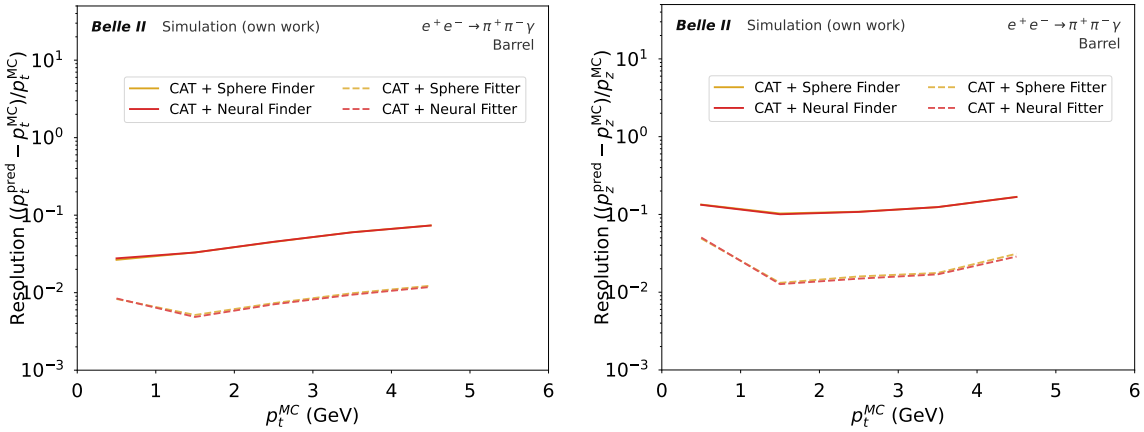
(b) *Hit purity*.

Figure A.7.: *Hit efficiency* and *purity* before track fitting for $\pi^\pm$ in $\Upsilon(4S) \to B\bar{B}$ samples with *high data beam background* for different algorithms limited to the intersecting sample in the barrel region.

A.2.5. Low Multiplicity Small Opening Angle

Table A.3.: The performance metrics for $e^+e^- \to \pi^+\pi^-\gamma$ samples for different track finding algorithms in different detector regions for *high data beam background*. Uncertainties below 0.01% are not shown in the table, uncertainties are correlated between different methods.

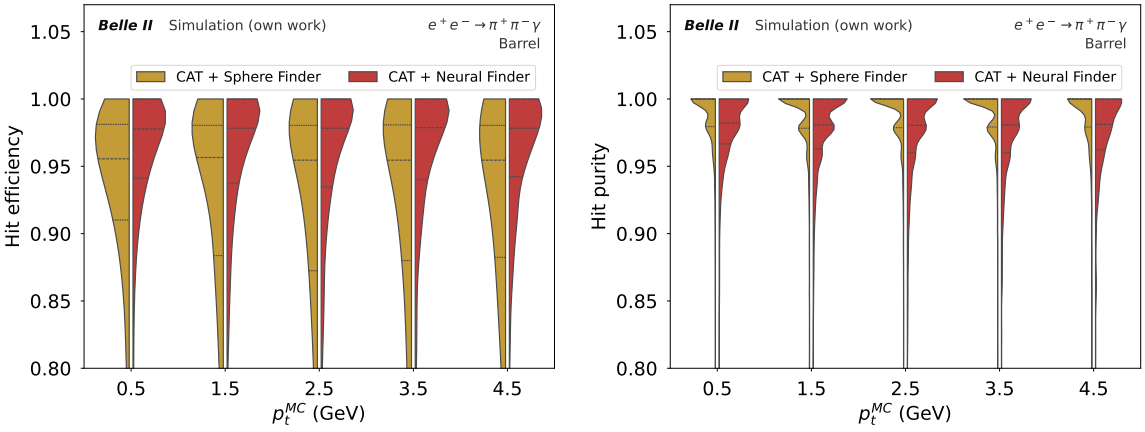
Method	$\varepsilon_{\text{trk}}[\%]$	$\mathfrak{r}_{\text{fake}}[\%]$	$\mathfrak{r}_{\text{clone}}[\%]$	$\varepsilon_{\text{trk,ch}}[\%]$	$\mathfrak{r}_{\text{wrong ch.}}[\%]$
			FWD		
Baseline Finder	$84.2^{+0.5}_{-0.5}$	$7.4^{+0.4}_{-0.4}$	$0.02^{+0.02}_{-0.03}$	$83.4^{+0.5}_{-0.5}$	$0.9^{+0.1}_{-0.1}$
CAT + Sphere Finder	$97.2^{+0.2}_{-0.2}$	$17.0^{+0.5}_{-0.5}$	$0.52^{+0.09}_{-0.11}$	$97.1^{+0.2}_{-0.2}$	$0.08^{+0.03}_{-0.05}$
CAT + Neural Finder	$97.9^{+0.2}_{-0.2}$	$18.7^{+0.5}_{-0.5}$	$0.23^{+0.06}_{-0.07}$	$97.7^{+0.2}_{-0.2}$	$0.21^{+0.06}_{-0.07}$
Baseline Fitter	$83.2^{+0.5}_{-0.5}$	$7.3^{+0.4}_{-0.4}$	$0.02^{+0.02}_{-0.03}$	$82.7^{+0.5}_{-0.5}$	$0.6^{+0.1}_{-0.1}$
CAT + Sphere Fitter	$95.1^{+0.3}_{-0.3}$	$9.0^{+0.4}_{-0.4}$	$0.22^{+0.06}_{-0.07}$	$93.5^{+0.3}_{-0.3}$	$1.7^{+0.2}_{-0.2}$
CAT + Neural Fitter	$96.7^{+0.3}_{-0.2}$	$9.5^{+0.4}_{-0.4}$	$0.08^{+0.03}_{-0.05}$	$95.1^{+0.3}_{-0.3}$	$1.6^{+0.2}_{-0.2}$
			BRL		
Baseline Finder	$97.3^{+0.1}_{-0.1}$	$11.7^{+0.3}_{-0.3}$	$0.08^{+0.02}_{-0.03}$	$97.1^{+0.1}_{-0.1}$	$0.23^{+0.04}_{-0.04}$
CAT + Sphere Finder	$98.5^{+0.1}_{-0.1}$	$21.1^{+0.3}_{-0.3}$	$0.89^{+0.08}_{-0.09}$	$98.3^{+0.1}_{-0.1}$	$0.12^{+0.03}_{-0.03}$
CAT + Neural Finder	$98.1^{+0.1}_{-0.1}$	$22.9^{+0.3}_{-0.3}$	$0.27^{+0.04}_{-0.05}$	$98.1^{+0.1}_{-0.1}$	$0.06^{+0.02}_{-0.03}$
Baseline Fitter	$97.2^{+0.1}_{-0.1}$	$11.0^{+0.3}_{-0.3}$	$0.08^{+0.02}_{-0.03}$	$97.1^{+0.1}_{-0.1}$	$0.14^{+0.03}_{-0.04}$
CAT + Sphere Fitter	$97.9^{+0.1}_{-0.1}$	$13.5^{+0.3}_{-0.3}$	$0.39^{+0.05}_{-0.06}$	$97.7^{+0.1}_{-0.1}$	$0.18^{+0.04}_{-0.04}$
CAT + Neural Fitter	$97.7^{+0.1}_{-0.1}$	$14.2^{+0.3}_{-0.3}$	$0.11^{+0.03}_{-0.03}$	$97.5^{+0.1}_{-0.1}$	$0.14^{+0.03}_{-0.04}$
			BWD		
Baseline Finder	$66.1^{+0.7}_{-0.7}$	$8.5^{+0.5}_{-0.5}$	$0.0_{-0.04}$	$63.6^{+0.7}_{-0.7}$	$3.7^{+0.3}_{-0.3}$
CAT + Sphere Finder	$95.2^{+0.3}_{-0.3}$	$17.8^{+0.5}_{-0.5}$	$0.35^{+0.08}_{-0.1}$	$95.0^{+0.3}_{-0.3}$	$0.2^{+0.06}_{-0.08}$
CAT + Neural Finder	$96.8^{+0.3}_{-0.2}$	$21.7^{+0.6}_{-0.6}$	$0.07^{+0.03}_{-0.05}$	$96.5^{+0.3}_{-0.3}$	$0.35^{+0.08}_{-0.1}$
Baseline Fitter	$65.0^{+0.7}_{-0.7}$	$8.4^{+0.5}_{-0.5}$	$0.0_{-0.04}$	$63.8^{+0.7}_{-0.7}$	$1.8^{+0.2}_{-0.2}$
CAT + Sphere Fitter	$91.1^{+0.4}_{-0.4}$	$11.6^{+0.5}_{-0.5}$	$0.19^{+0.06}_{-0.07}$	$87.7^{+0.5}_{-0.5}$	$3.7^{+0.3}_{-0.3}$
CAT + Neural Fitter	$93.5^{+0.4}_{-0.4}$	$12.8^{+0.5}_{-0.5}$	$0.05^{+0.03}_{-0.04}$	$89.7^{+0.4}_{-0.4}$	$4.1^{+0.3}_{-0.3}$



(a) Transverse momentum resolution.

(b) Axial momentum resolution.

Figure A.8.: Momentum resolution before (solid lines) and after (dashed lines) track fitting for $e^+e^- \to \pi^+\pi^-\gamma$ samples with *high data beam background* for different algorithms limited to the intersecting sample in the barrel region.

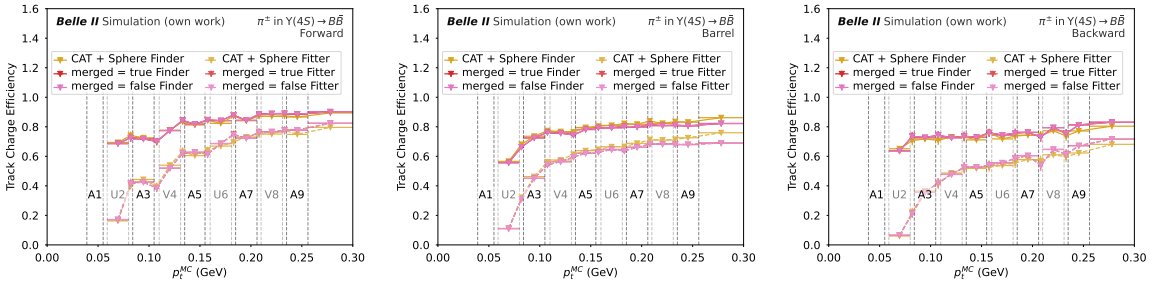


(a) *Hit efficiency.*

(b) *Hit purity.*

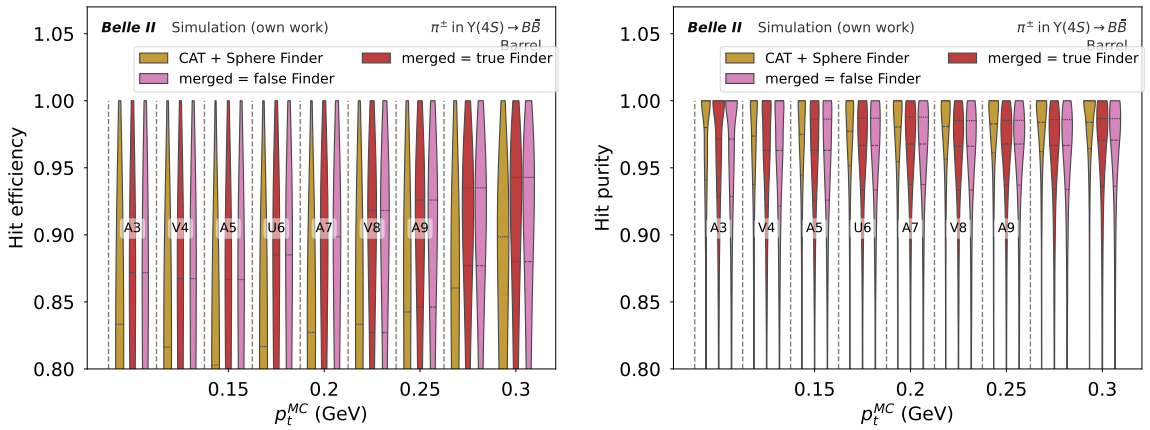
Figure A.9.: *Hit efficiency* and *purity* before track fitting for $e^+e^- \to \pi^+\pi^-\gamma$ samples with *high data beam background* for different algorithms limited to the intersecting sample in the barrel region.

A.2.6. Non-Max Suppression



(a) Forward endcap. (b) Barrel. (c) Backward endcap.

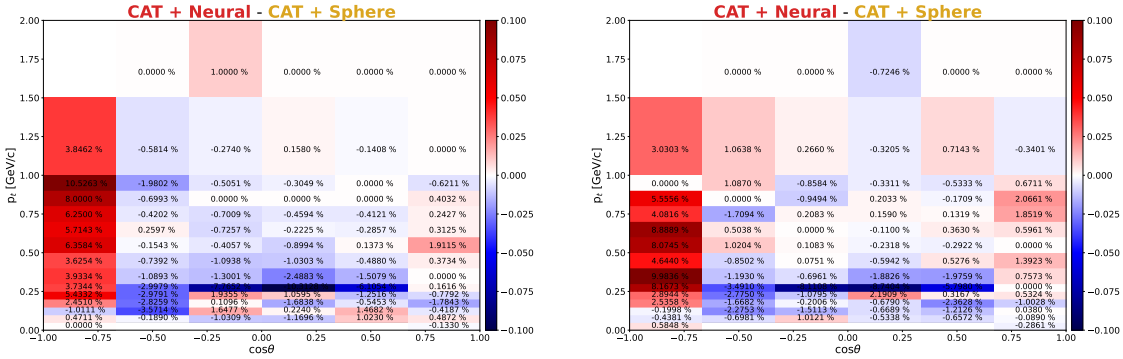
Figure A.10.: *Track charge efficiency* before (solid lines) and after (dashed lines) track fitting for $\pi^{\pm}$ in $\Upsilon(4S) \to B\bar{B}$ samples with *high data beam background* for different algorithms and detector regions. Horizontal bars indicate bin width, vertical error bars indicate uncertainties (correlated between algorithms).



(a) *Hit efficiency.* (b) *Hit purity.*

Figure A.11.: *Hit efficiency* and *purity* before track fitting for $\Upsilon(4S) \to B\bar{B}$ samples with *high data beam background* for different algorithms limited to the intersecting sample in the barrel region.

A.3. Generalization to Low Background



(a) *Low simulated beam background.* (b) *High data beam background.*

Figure A.12.: Difference in *track fitting charge efficiency* in the CDC between neural and spherical clustering on $\pi^{\pm}$ in $\Upsilon(4S) \rightarrow B\bar{B}$ events with low background compared to *high data beam background.* Red bins indicate where neural clustering performs better, blue bins show worse performance.

Table A.4.: Full detector post-fitting track finding performance for different algorithms on $\pi^{\pm}$ from simulated $\Upsilon(4S) \rightarrow B\bar{B}$ decays with *low simulated beam background.* Uncertainties below 0.01% are not shown in the table.

Method	$\varepsilon_{\mathrm{trk,ch}}[\%]$	$\mathfrak{r}_{\mathrm{fake}}[\%]$	$\mathfrak{r}_{\mathrm{clone}}[\%]$
		CDC only	
Baseline	74.14 ± 0.16	1.59 ± 0.04	0.85 ± 0.03
CAT + Sphere	79.65 ± 0.15	1.44 ± 0.04	1.62 ± 0.04
CAT + Neural	78.93 ± 0.15	2.37 ± 0.05	1.27 ± 0.04
		Full detector	
Baseline	92.07 ± 0.1	3.34 ± 0.05	3.34 ± 0.05
CAT + Sphere	92.78 ± 0.1	3.35 ± 0.05	3.53 ± 0.06
CAT + Neural	91.55 ± 0.1	3.92 ± 0.06	3.95 ± 0.06

.

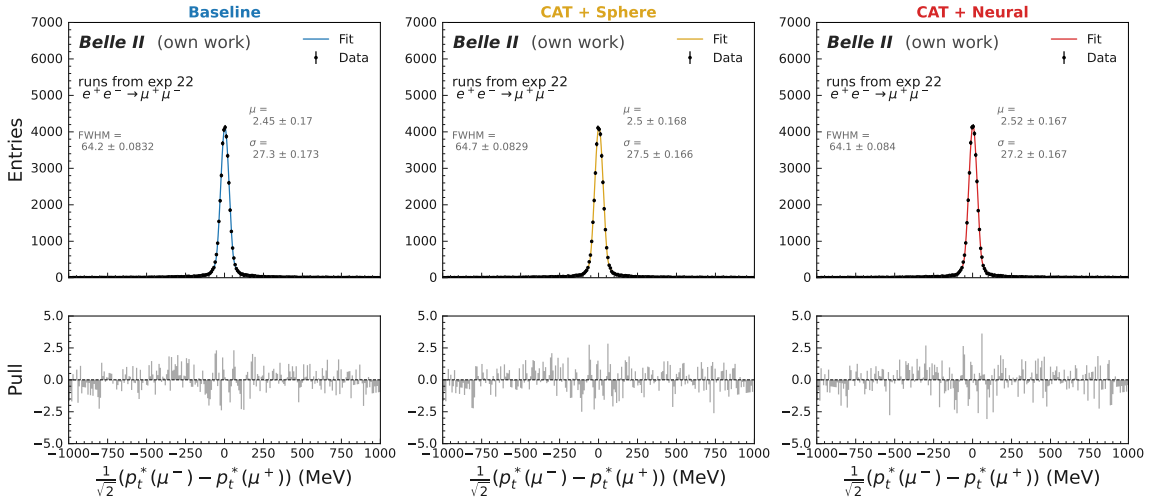


Figure A.13.: DSCB [24]-fit for the difference in transverse momentum between the two $\mu^\pm$ in the center-of-mass frame in $e^+e^- \to \mu^-\mu^+$ data events with low beam background for different algorithms in the full reconstruction. Uncertainties are correlated between algorithms, as they are evaluated on the same events.

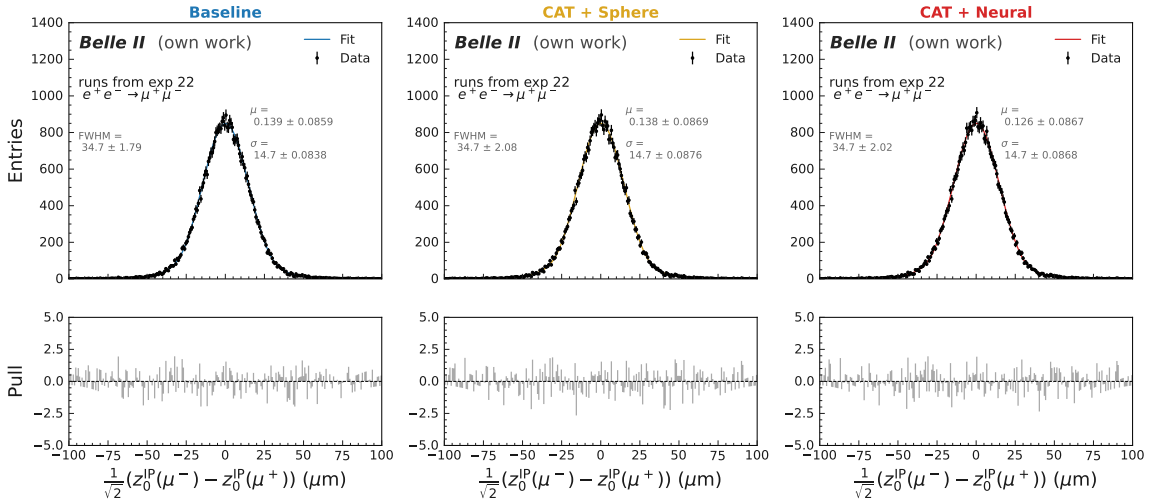


Figure A.14.: DSCB [24]-fit for the difference in axial distance to the IP between the two $\mu^\pm$ in $e^+e^- \to \mu^-\mu^+$ data events with low beam background for different algorithms in the full reconstruction. Uncertainties are correlated between algorithms, as they are evaluated on the same events.

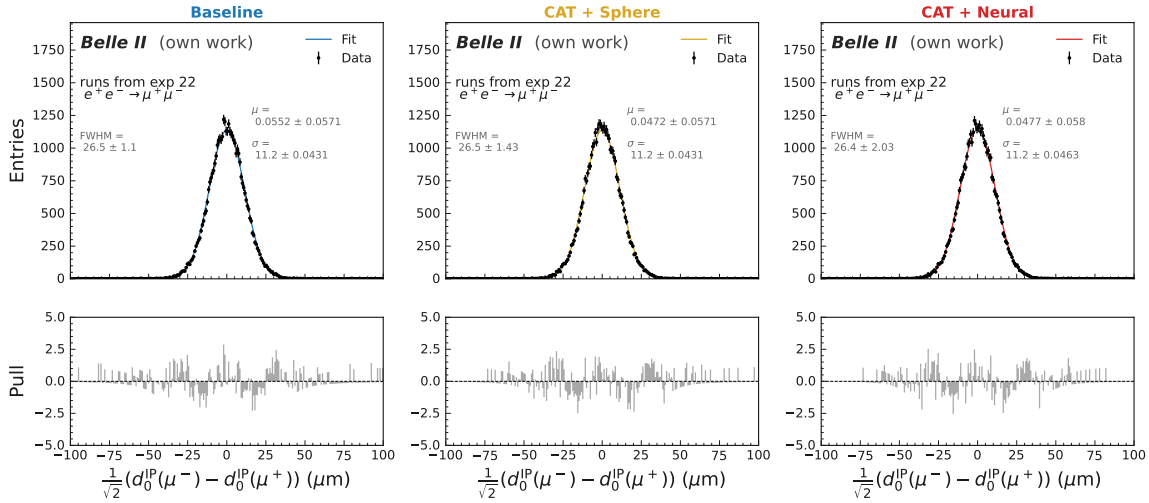


Figure A.15.: DSCB [24]-fit for the difference in radial distance to the IP between the two $\mu^{\pm}$ in $e^+e^- \to \mu^-\mu^+$ data events with low beam background for different algorithms in the full reconstruction. Uncertainties are correlated between algorithms, as they are evaluated on the same events.

List of Acronyms

***CATFinder*-GNN** *CATFinder* Graph Neural Network. 5, 10, 11, 16, 21–25, 27, 30, 35, 49, 52, 53, 55, 60–64, 81, 82

ARICH Aerogel Ring Imaging Cherenkov Counter. 1

CDC Central Drift Chamber. 1, 3, 4, 7, 9, 16–18, 29, 36, 40, 41, 43–45, 47–49, 55, 56, 70–75, 77, 81

DPC Density Peak Clustering. 11

ECL Electromagnetic Calorimeter. 1, 16, 40, 48

FPGA Field Programmable Gate Array. 24

FWHM Full Width at Half Maximum. 74, 75

GNN Graph Neural Network. 1, 3, 5, 8, 9, 12, 16, 24, 27

HEP high energy physics. 1, 11

HLT High Level Trigger. 1, 11

IOU Intersection over Union. 12, 13, 24

IP Interaction Point. 3, 16, 17, 35, 45, 72

LLP Long Lived Particle. 9, 44, 45

MLP Multi-Layer Perceptron. 12, 25, 30, 61

NMS Non-Maximum Suppression. 12

ReLU Rectified Linear Unit. 25

SNNC Shared Nearest Neighbor Clustering. 11

SVD Silicon Vertex Detector. 36, 45, 71, 76, 82

VDPC Variational Density Peak Clustering. 11, 52–54

YOLO You Only Look Once. 12, 21, 24, 59, 81

Glossary

basf2 Belle II Analysis Software Framework [25, 26]. 15, 71, 82

Belle II A second-generation B factory and the successor of the Belle experiment.. 3, 4

DBSCAN Density-Based Spatial Clustering of Applications with Noise. 11

DSCB [24] Double sided crystal ball fit. 72–77, 98, 99

EvtGen [21] A Monte Carlo Generator suited for the decay of heavy flavour particles. See [21]. 16, 17

GEANT4 [18] Simulation framework for the propagation of particles through matter using phenomenological models. See [18, 27, 28]. 15

GENFIT2 [9] Track Fitting Framework. 4

GravNet [10] GNN architecture for distance-weighted graph networks. 5

HDBSCAN Hierarchical Density-Based Spatial Clustering of Applications with Noise. 53, 54

KKMC [19] The precision Monte Carlo event generator for two-fermion final states in collisions. See [19]. 16, 17

Phokhara [20] Radiative return at NLO and the measurement of the hadronic cross-section in electron–positron annihilation. See [20]. 16, 17

PointNet Machine learning architecture focused on applicability on point clouds. 12, 21, 24, 81

SuperKEKB An upgrade of the KEKB electron-positron collider and the accelerator at which the Belle II experiment is located.. 1, 3, 52