Optimization of the z-Vertex Neural Network Trigger for the Belle II Experiment

Sara McCarney

Munich 2019

Optimization of the z-Vertex Neural Network Trigger for the Belle II Experiment

Sara McCarney

Master Thesis
at the Faculty of Physics
of Ludwig–Maximilians–Universität
Munich

submitted by
Sara McCarney
from Belfast, Northern Ireland

Munich, 08.08.2019

Primary reviewer: Prof. Dr. Christian Kiesling

Optimierung des Neuronalen z-Vertex Trigger für das Belle II Experiment

Sara McCarney

Masterarbeit
an der Fakultät für Physik
der Ludwig–Maximilians–Universität
München

vorgelegt von
Sara McCarney
aus Belfast, Nordirland

München, den 08.08.2019

Erstgutachter: Prof. Dr. Christian Kiesling

Declaration:

I hereby declare that this thesis is my own work, and that I have not used any sources and aids other than those stated in the thesis.

München, 08.08.2019

Contents

List of Figures

List of Tables

Abstract

For the Belle II experiment at the SuperKEKB asymmetric electron-positron (e^+/e^-) collider (KEK, Japan) the concept of a first level (L1) track trigger, realized by neural networks, is presented. Using the input from a traditional Hough-based 2D track finder, the stereo wire layers of the Belle II Central Drift Chamber are used to reconstruct by neural methods the origin of the tracks along the beam (z) direction. A z-trigger for Belle II is required to suppress the dominating background of tracks from outside of the collision point. This so-called Neurotrigger is based on a Multi-Layer Perceptron (MLP) Architecture and is implemented in FPGA hardware to trigger on events in real-time, satisfying a fixed latency budget of 300 ns. The Neural Networks are trained offline in a supervised learning process using Monte Carlo (MC) particles as targets. The full L1 track trigger can be simulated in software to obtain resolutions, comparing the 'true' MC values to the predicted values of the network. By means of these software simulations, one can find optimal parameters for the preprocessing and training of the Neurotrigger. This thesis presents the results of such software simulations. Resolutions of $\sim$2 cm in the high particle transverse momentum (p_t) region, and $\sim$5 cm in the low p_t region are determined, sufficient for efficient background rejection. The importance of the selected drift time input algorithm on the optimal spatial resolution of the z-trigger and trainings for a preliminary z-cut of 40 cm are discussed.

Introduction

The Belle II Experiment at the SuperKEKB asymmetric electron-positron (e^+/e^-) collider (KEK, Japan) - a second-generation B-factory - aims to precisely measure CP-violation in the B-meson sector. Colliding e^- and e^+ at respective beam energies of 7 GeV and 4 GeV, which corresponds to the $\Upsilon(4S)$ resonance, large quantities of B-meson pairs can be produced, allowing for the parameter space of the unitarity triangle to be further confined. With unprecedented sensitivity measurements, Belle II also aims to search for new physics (NP) beyond the Standard Model at the intensity frontier.

Following the success of its predecessor experiment Belle, the Belle II detector has been upgraded to include a new Pixel Detector (PXD) based on DEPFET technology and a larger Central Drift Chamber (CDC). KEKB is upgraded to SuperKEKB to reach peak luminosities of up to $8 \times 10^{35}\,\mathrm{cm}^{-2}\,\mathrm{s}^{-1}$ and aims to achieve a total integrated luminosity sample of $50\,\mathrm{ab}^{-1}$. With increasing beam currents and the new nanobeam scheme, the planned luminosity upgrade is expected to be accompanied by a strong increase in beam-induced background compared to Belle. In order to identify, for example, low charged track multiplicity events such as τ pairs, this background should be suppressed as much a possible.

Belle II's trigger system aims then to discard a significant amount of this background at the first level trigger. By using the input from a traditional Hough-based 2D track finder, the stereo wire layers of the Belle II Central Drift Chamber are used to reconstruct in three dimensions and by neural methods the origin of the tracks along the beam (z) direction. A z-trigger for Belle II is required to suppress the dominating background of tracks from outside of the collision point. Belle did not employ a z-trigger, and as such their track trigger did not sufficiently reject an overwhelming quantity of background that originated outside the Interaction Point (IP), for example Bhabha tracks that hit accelerator structures, or Toushek effects from beam-induced background. A distribution of this background can be seen in Figure 1.

Since traditional track finding methods cannot be executed within the latency of the pipelined L1 trigger, a trigger based on neural networks (Neurotrigger) has been developed [2, 3, 4, 5] for implementation on FPGA hardware to operate in real time as part of the modular L1 track trigger. The Neurotrigger outputs z-Vertex and polar angle θ predictions. These outputs are sent to a Global Decision Logic (GDL) which makes a final decision on which events to keep. The goal of the Neurotrigger then is to discard events originiating outside a pre-determined 'z-cut', and to correctly identify those originating from the IP, which correspond to 'interesting' physics events.

The neural networks are trained offline in a supervised learning process with Monte Carlo (MC) particles used as training targets. The network weights are uploaded onto the hardware to trigger on events in realtime, with input values calculated in preprocess-

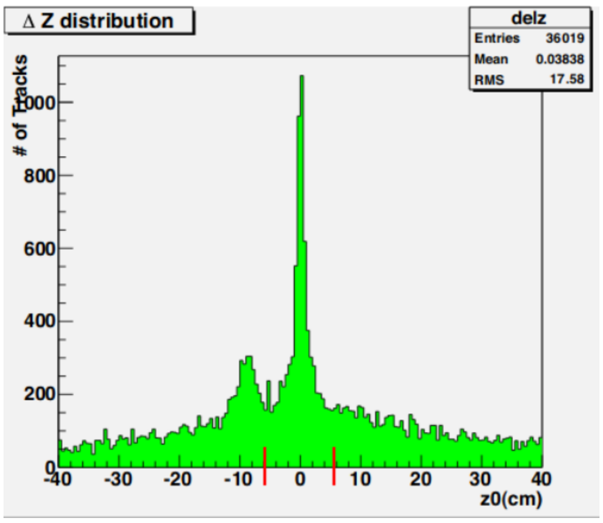


Figure 1.: Background distribution along z-axis in Belle [1]

ing steps on the FPGA. The full L1 track trigger up to and including the Neurotrigger may also be simulated in the software to determine resolutions and efficiencies, by comparing to the 'true' MC values to those predicted by the network. This is also used to debug the hardware, finding any discrepancies in results of the pipelined L1 track trigger algorithm and hardware implementations. This thesis presents the results of such software simulations. Resolutions were found to be dependent on particle transverse momentum (p_t) as expected due to a multiple scattering of particles. The networks achieve $\sim$2 cm resolutions in the high p_t region, and $\sim$5 cm in the low p_t region, sufficient for efficient background rejection. The importance of the selected drift time input algorithm on the optimal spatial resolution of the z-trigger and trainings for a preliminary z-cut of 40 cm are discussed.

As of April 2019, Belle II entered Phase 3, with first runs showing agreement with the simulation expectations. Improvement on z-cuts are expected to be made upon further optimization of the Neurotrigger.

This thesis will present some of the physics motivations behind the Belle II experiment in Chapter 1, including CP violation and rare τ-decays, before giving an overview of the SuperKEKB collider and the Belle II detector in Chapter 2. An overview of Belle II's trigger system will be given in Chapter 3 , with focus on the L1 track trigger. Since the z-trigger component of the L1 track trigger is deployed with artificial neural networks, Chapter 4 provides an overview of the neural architecture and algorithms used to train the networks. The results of the software simulations of the Neurotrigger are presented in Chapter 5, with recommendations on optimal training parameters. Details of the software simulation methods, parameters and error calculations can be found in the Appendix.

1. Physics Motivations

Belle II operates at the $\Upsilon(4S)$ resonance -just above the threshold for B-meson pair production - producing large quantities of B-mesons in order to precisely measure CP-violation in the B-meson system. This is enabled by the relatively 'clean' environments at SuperKEKB which also provides a boost to the centre-of-mass (COM) system, enabling better resolution of decay parameters. Belle II will perform precision measurements aimed to test the standard model parameters, which could be indicative of new physics (NP) beyond the standard model, as well as providing constraints, and discriminating between, NP models [6]. This chapter gives an overview of the physics motivations for the Belle II upgrade, including CP-violation in the B-meson system and rare τ-decays.

1.1. The Standard Model

The standard model of particle physics (SM) is currently the best tested theory of subatomic physics that describes the elementary particles and the fundamental forces that govern their interactions. The elementary particles can be distinguished by their spin quantum number: fermions are half-integer spin particles while the bosons carry integer spin. The gauge bosons are spin-1 particles that mediate a force - the W and Z bosons mediate the Weak Interaction, the photon γ mediates the Electromagenetic Interaction and the gluons mediate the Strong Interaction. The Higgs boson, a spin-0 boson, is responsible for the mass generation of heavy elementary particles and was discovered in 2012 at the Large Hadron Collider (LHC) by both the ATLAS and CMS experiments [7, 8].

The SM contains twelve fermions split into three generations of quarks (fermions that participate in the Strong Interaction) and leptons (fermions that do not participate in the Strong Interaction). Each generation consists of two particles leading to a total of six quark flavours (up, down, charm, strange, top and bottom) and six lepton flavours (electron, muon, tau and their corresponding neutrinos). Flavour physics concerns itself with the mixing of these so-called flavours. A summary of the SM can be seen in Figure 1.1.

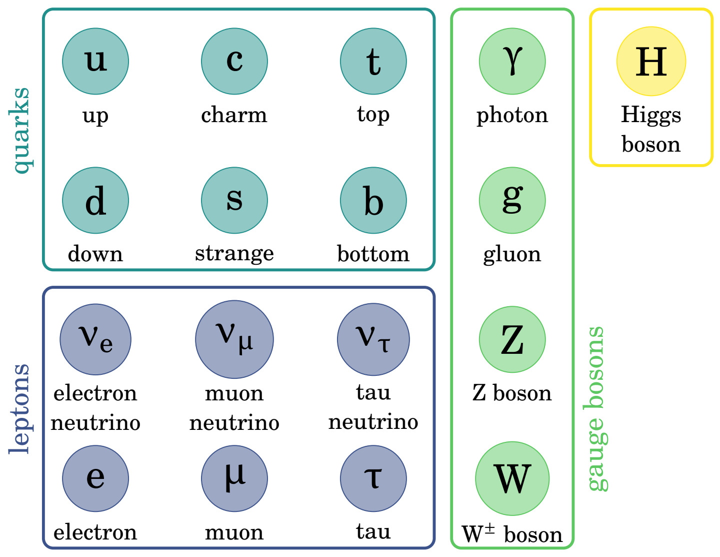


Figure 1.1.: The elementary particles in the standard model of particle physics [9]

Although the SM is currently our best phenomenological theory of subatomic physics, many phenomena remain unexplained: the question of why there are only three generations of fermions, the origin of the hierarchy of mass, the absence of a dark matter candidate and the origin of neutrino mass are among the many open-ended questions the SM fails to answer. The Belle II Experiment aims to investigate some of these phenomena. In particular, Belle II will investigate the nature of CP-violation, which could hint to the unexplained asymmetry of matter and antimatter - that there is more matter observed in the universe and almost no antimatter. CP-violation is a necessary condition for this asymmetry, but the measured levels of CP-violation in the quark sector are far too small to compensate for the imbalance.

Symmetries and conservation laws play an important role in particle physics. According to the famous Noether Theorem [10], conservation laws in physics are derived from an underlying symmetry. In the following we will discuss three fundamental (and discrete) symmetries, giving rise to surprising experimental findings and theoretical consequences. Violations of these symmetries are an essential ingredient to understand the working of the Weak Interaction, responsible for the transformation of flavour states.

1.1.1. Parity Transformation

Parity transformations are inversions at the origin of a coordinate system. Figure 1.2 shows an example of such an inversion, which is equivalent to a mirror reflection at the plane followed by a 180° rotation around the axis orthogonal to the plane. Parity violation has not been observed in Strong and Electromagnetic interactions. The Weak Interaction, on the other hand, violates parity maximally, as demonstrated in the famous Wu Experiment conducted by C.S. Wu. Lee and Yang who proposed the violation of parity in the Weak Interaction, were awarded the Nobel prize in 1957 [11].

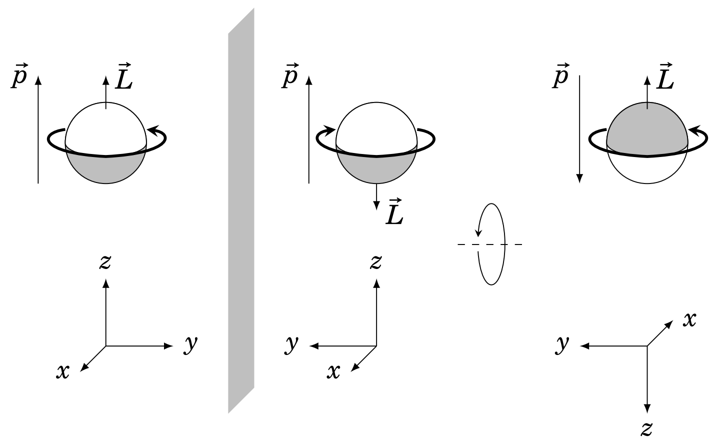


Figure 1.2.: The effect of parity transformation, which is equivalent to a mirror reflection followed by a 180° rotation about the mirror axis [9]

1.1.2. Charge Conjugation

Charge parity (C-parity), or charge conjugation, is also conserved in the Strong and Electromagnetic interactions. The C-parity operator transforms particles into their antiparticles with a change of sign of a generalised charge. The Electromagnetic charge, baryon number and the lepton number are among the charges which undergo a sign reversal. C-parity is observationally maximally violated in the Weak Interaction - we see no left-handed anti-neutrinos and no right-handed neutrinos. What we do see, however, are left-handed neutrinos and right-handed anti-neutrinos, indicating the conservation of the combined Parity and Charge conjugation operators, (CP-conservation). Later we will see that this is also violated.

1.1.3. Cabibbo Angle & Discovery of new quarks

The left-handed states can transform into one another via the Weak Interaction and so are grouped into generation-wise doublets. In 1963, when only the up (u), down (d) and strange (s) quarks were known to exist, it was observed that the u-quark could transition, with different strength, to either a d-quark or an s-quark. Therefore, first proposed by N. Cabibbo [12], the eigenstates of the Weak Interaction were:

$$d' = d\cos\theta_c + s\sin\theta_c \tag{1.1}$$

where θ_c is the Cabibbo angle. On comparing the lifetimes of the charged pions and kaons, the Cabibbo angle was found to be $\theta_c \approx 13.1°$. However, the branching ratio of ${K_L}^0 \longrightarrow \mu^+\mu^-$ was found to be much lower than expected. To explain this effect, the GIM Mechanism, proposed by Glashow, Iliopoulos and Maiani in 1970, introduced a fourth quark, named charm (c) [13]. There were now two left-handed interactions, defining now a relation analogous to Equation 1.1, namely:

$$s' = s\cos\theta_c - d\sin\theta_c \tag{1.2}$$

and the new weak doublet $\begin{pmatrix} c \\ s \end{pmatrix}$, the decay $\mathrm{K_L}^0 \longrightarrow \mu^+\mu^-$ could be suppressed with the addition of a second loop diagram as shown in Figure 1.3. The c-quark was experimentally verified several years later.

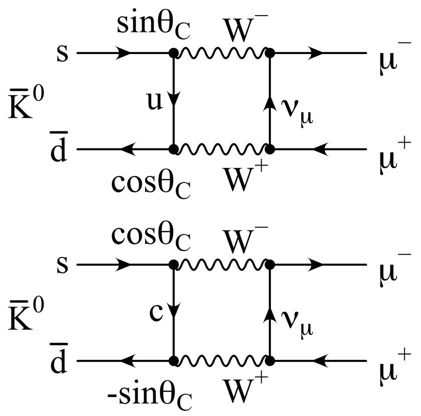


Figure 1.3.: The GIM Mechanism proposed a fourth quark, c, to achieve the required flavour-changing neutral current (FCNC) suppression (note the '-' sign in the lower diagram in front of $\sin(\theta_c)$, coupling the c-quark to the d-quark)

Cabibbo's mixing angle was thus extended to a matrix which described this coupling of u and c-quarks to mixed eigenstates of the d and s-quark (of the Weak interaction). For two generations, the unitary 2×2 matrix was:

$$\mathrm{V_C} = \begin{pmatrix} \cos\theta_c & \sin\theta_c \\ -\sin\theta_c & \cos\theta_c \end{pmatrix} \tag{1.3}$$

where the matrix $\mathrm{V_C}$ operates on the d and s-quark states:

$$\begin{pmatrix} \mathrm{d'} \\ \mathrm{s'} \end{pmatrix} = \mathrm{V_C} \begin{pmatrix} \mathrm{d} \\ \mathrm{s} \end{pmatrix} \tag{1.4}$$

In 1964, Christenson, Cronin, Fitch and Turlay proved CP-violation in the neutral kaon system, with their observation of two-pion decays in neutral kaon mixing of the $\mathrm{K_L}$ meson; to conserve CP, the $\mathrm{K_L}$ meson should only decay to three pions [14]. To explain this CP-violation within the SM, Kobayashi and Masakawa predicted a third generation of quarks, extending the Cabibbo matrix to the Cabibbo-Kobayashi-Masakawa (CKM) matrix, with a non-vanishing complex phase. The new quarks predicted by Kobayashi and Masakawa were discovered in 1975 (charm or c-quark), in 1977 (bottom or b-quark), and in 1995 the top (t-quark) was discovered at Fermilab [15, 16, 17, 18]. Kobayashi and Masakawa were awarded the Nobel prize for their bold conjecture in 2008, which was proven correct by the Belle and BaBar experiments [19, 20].

1.1.4. Cabibbo-Kobayashi-Masakawa Matrix

The CKM matrix elements can be labelled to represent the quark flavours involved in the charged current interactions:

$$\mathrm{V_{CKM}} = \begin{pmatrix} \mathrm{V_{ud}} & \mathrm{V_{us}} & \mathrm{V_{ub}} \\ \mathrm{V_{cd}} & \mathrm{V_{cs}} & \mathrm{V_{cb}} \\ \mathrm{V_{td}} & \mathrm{V_{ts}} & \mathrm{V_{tb}} \end{pmatrix} \tag{1.5}$$

where $\mathrm{V_{CKM}}$ acts on the d, s and b mass eigenstates and d', s' and b' are the flavour eigenstates which actually decay in the Weak Interaction:

$$\begin{pmatrix} \mathrm{d'} \\ \mathrm{s'} \\ \mathrm{b'} \end{pmatrix} = \mathrm{V_{CKM}} \begin{pmatrix} \mathrm{d} \\ \mathrm{s} \\ \mathrm{b} \end{pmatrix} \tag{1.6}$$

As example, the element $\mathrm{V_{ub}}$ appear in the coupling of a b and u-quark to the W boson. $\mathrm{V_{CKM}}$ is a unitary, 3×3 matrix with three real parameters and an irreducible complex phase. Several unitarity relations therefore hold, one of which is the B-triangle, where the first column is multiplied with the complex conjugated last column. For CP-violation the complex phase must be non-vanishing. As can be seen in the Wolfenstein representation of the CMK matrix (see below), $\mathrm{V_{ub}}$ and $\mathrm{V_{td}}$ contain the complex phase.

However, the observed CP-violation within the quark sector that originates from the complex phase of the CKM matrix is many orders of magnitude too small to explain the dominance of matter in the universe. There must therefore be undiscovered sources of CP-violation.

1.1.5. Wolfenstein Parametrization & Unitarity triangle

In general, a unitary n×n matrix has n^2 free parameters. An orthogonal n×n matrix can be constructed from $\frac{1}{2}$n(n-1) real parameters usually written as angles describing the rotation. The remaining $\frac{1}{2}$n(n-1) free parameters are the complex phases, however only $\frac{1}{2}$(n-1)(n-2) are physically observable phases that could lead to CP-violation. For a 3×3 unitary matrix, this leads to exactly one non-vanishing complex phase. A ‘standard’ parametrization of the CKM matrix is given by:

$$\mathrm{V_{CKM}} = \begin{pmatrix} \mathrm{c_{12}c_{13}} & \mathrm{s_{12}c_{13}} & \mathrm{s_{13}}e^{-i\delta} \\ -\mathrm{s_{12}c_{23}} - \mathrm{c_{12}s_{23}s_{13}}e^{i\delta} & \mathrm{c_{12}c_{23}} - \mathrm{s_{12}s_{23}s_{13}}e^{i\delta} & \mathrm{s_{23}c_{13}} \\ \mathrm{s_{12}s_{23}} - \mathrm{c_{12}c_{23}s_{13}}e^{i\delta} & -\mathrm{c_{12}c_{23}} - \mathrm{s_{12}c_{23}s_{13}}e^{i\delta} & \mathrm{c_{23}c_{13}} \end{pmatrix} \tag{1.7}$$

where $\mathrm{s_{ij}} := \sin\theta_{ij}$, $\mathrm{c_{ij}} := \cos\theta_{ij}$ and θ_{12}, θ_{23} and θ_{13} are the rotation angles where θ_{12} is the Cabibbo angle and is the largest mixing angle in the CKM matrix. $\delta := \delta_{13}$ is the complex phase leading to CP-violation.

Another common parametrization is the Wolfenstein Parametrization, which expands the CKM matrix in termns of the parameter $\lambda = \sin\theta_{12} \approx 0.22$ [21]. Wolfenstein then defined four parameters (λ, A, ρ, η):

$$\lambda = \sin\theta_{12} \qquad \mathrm{A}\lambda^2 = \sin\theta_{23} \qquad \mathrm{A}\lambda^3(\rho - i\eta) = \sin\theta_{12} e^{-i\delta}$$

so that up to $\mathcal{O}(\lambda^3)$, the CKM matrix can be written as:

$$\mathrm{V_{CKM}} = \begin{pmatrix} 1 - \frac{1}{2}\lambda^2 & \lambda & \mathrm{A}\lambda^3(\rho - i\eta) \\ -\lambda & 1 - \frac{1}{2}\lambda^2 & \mathrm{A}\lambda^2 \\ \mathrm{A}\lambda^3(1 - \rho - i\eta) & -\mathrm{A}\lambda^2 & 1 \end{pmatrix} \tag{1.8}$$

The advantage of this parametrization is that one can clearly see the hierarchy of the couplings - the largest couplings ($\mathcal{O}(1)$ belong to the same generation (along the diagonal) and the smallest couplings ($\mathcal{O}(\lambda^3)$ are between the first and third generations of quarks. Note that the complex phase appears only in couplings of the order of $\mathcal{O}(\lambda^3)$, indicating that CP symmetry is violated weakly in the SM [9].

One can expand the unitarity conditions $\mathrm{V_{CKM}}\mathrm{V_{CKM}}^\dagger = \mathbb{1}$ up to $\mathcal{O}(\lambda^3)$ which can be represented geometrically as a triangle on the complex plane. One particular orthogonality condition is interesting:

$$\mathrm{V_{ud}}\mathrm{V_{ub}}^* + \mathrm{V_{cd}}\mathrm{V_{cb}}^* + \mathrm{V_{td}}\mathrm{V_{tb}}^* = 0 = \mathrm{A}\lambda^3(\rho - i\eta) - \mathrm{A}\lambda^3 + \mathrm{A}\lambda^3(1 - \rho - i\eta) + \mathcal{O}(\lambda^5) \tag{1.9}$$

which contains three complex terms of $\mathcal{O}(\lambda^3)$ that must form a closed triangle in the complex plane, as shown in Figure 1.4. It is convenient to normalize these terms so that one side is purely real with length 1. By measuring the internal angles of the triangle, one can test the unitarity of the CKM matrix, which is only unitary if the triangle closes. This triangle is known at the B-triangle, since it includes the b-quark. An important property of this triangle is that the sides are all of the same order, which entails large CP-violating effects in the B-meson system.

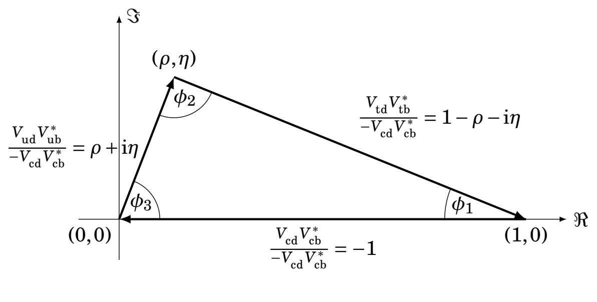

$\Im$
(ρ,η)
ϕ_2
$\frac{V_{td}V_{tb}^*}{-V_{cd}V_{cb}^*} = 1-\rho-i\eta$
$\frac{V_{ud}V_{ub}^*}{-V_{cd}V_{cb}^*} = \rho+i\eta$
ϕ_3
ϕ_1
$\Re$
(0,0)
(1,0)
$\frac{V_{cd}V_{cb}^*}{-V_{cd}V_{cb}^*} = -1$

Figure 1.4.: The unitarity condition can be represented as a triangle in the complex plane [9]

Mesons		Antimesons	
K	$\overline{\mathrm{s}}\mathrm{d}$	$\overline{\mathrm{K}}$	$\mathrm{s}\overline{\mathrm{d}}$
$\mathrm{B^0}_\mathrm{d}$	$\overline{\mathrm{b}}\mathrm{d}$	$\overline{\mathrm{B^0}_\mathrm{d}}$	$\mathrm{b}\overline{\mathrm{d}}$
$\mathrm{B^0}_\mathrm{s}$	$\overline{\mathrm{b}}\mathrm{s}$	$\overline{\mathrm{B^0}_\mathrm{s}}$	$\mathrm{b}\overline{\mathrm{s}}$
D	$\mathrm{c}\overline{\mathrm{u}}$	$\overline{\mathrm{D}}$	$\overline{\mathrm{c}}\mathrm{u}$

Table 1.1.: The mesons and their antiparticle constituents which exhibit mixing

1.2. CP Violation

A CP transformation combines the operators C and P successively, therefore interchanging a particle with its antiparticle and reversing the handedness of the particle. Therefore a CP transformation acts on a left-handed quark to produce a right-handed antiquark ($\mathrm{q_L} \rightarrow \overline{\mathrm{q_R}}$). Mesons then, which are bound states of a quark and an anti-quark, $\mathrm{M} = \mathrm{q_1}\overline{\mathrm{q_2}}$, are transformed to $\mathrm{M} = \overline{\mathrm{q_1}}\mathrm{q_2}$. Some meson constituents can be seen in Table 1.1. CP is conserved in Strong and Electromagnetic interactions but violated in the Weak Interaction; CP-violation is a necessary condition to explain the observed asymmetry of matter and antimatter quantities in the universe [22]. The hadrons which exhibit mixing with their antiparticles are shown in Table 1.1. This section outlines CP-violation effects and ways to measure it.

1.2.1. Direct & Indirect CP Violation

Indirect CP-violation is related to the mixing of particles and antiparticles, and occurs when different probabilities or phases for the transitions $\langle\overline{\mathrm{M}}|\mathrm{M}\rangle \neq \langle\mathrm{M}|\overline{\mathrm{M}}\rangle$ arises. For the neutral B-meson system, we have indirect CP-violation when $\langle\mathrm{B_0}|\overline{\mathrm{B_0}}\rangle \neq \langle\overline{\mathrm{B_0}}|\mathrm{B_0}\rangle$.

Direct CP-violation occurs in decays if $|\langle\mathrm{f}|B\rangle|^2 \neq |\langle\overline{\mathrm{f}}|\overline{\mathrm{B}}\rangle|^2$ - that is when the transition probability of a meson decaying to a fermion it not equal to reverse process (an anti-meson decaying to an anti-fermion). This can happen for charged and neutral B-mesons.

A hybrid form of direct and indirect CP-violation can also occur whenever there is CP-violation in the interference between mixing and decay. All these observables will be measured at Belle II. The effect is small in the kaon sector but turned out to be much larger in the B-meson sector. Belle successfully measured this, however the statistics at Belle are not sufficient to see significant deviations from the SM, which must be there, albeit at very small levels.

1.2.2. CP Violation at Belle II

Running at the $\Upsilon(4S)$ resonance, the main decay mode at Belle II is $\mathrm{B_d}^0\overline{\mathrm{B_d}^0}$. The boost in the COM frame, provided by the asymmetric COM energies of electron and positron collision, is transmitted to the B mesons. This is important in measuring asymmetries as a function of their decay length and allows for increased sensitivity to CP violating effects. A typical time-dependent CP-violation decay at Belle II of a $\mathrm{B^0}\overline{\mathrm{B^0}}$ can be seen in Figure 1.5 and time differences may be measured via the boosted decay points of the two B mesons to detect CP-violating effects.

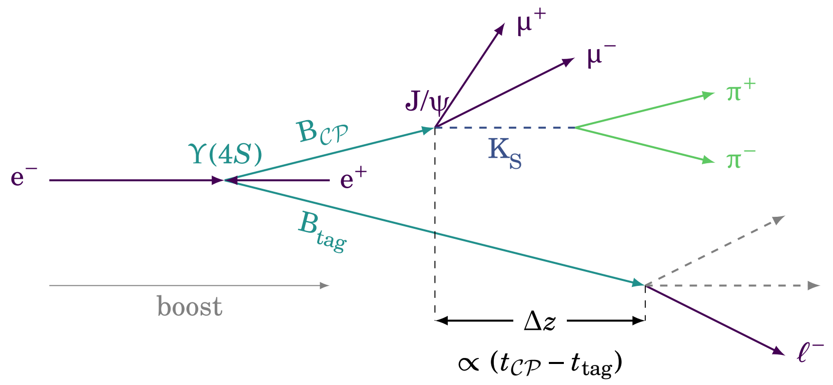


Figure 1.5.: Typical $B^0\overline{B^0}$ decay for CP-violation measurements at Belle II [9]

For decays such as $B_{CP} \rightarrow J/\Psi K_S$, where B_{CP} can be a matter or antimatter particle, a flavour tagging procedure needs to be performed, where the algorithm looks for flavour-specific signatures, for example charged leptons in final state particles, in order to identify the flavour of B_{tag} at the time of its decay. From this the flavour of B_{CP} can also be determined, since the two B-mesons are in a coherent state of exactly one B^0 and one $\overline{B^0}$ until one of them decays, at which point the remaining B-meson is free to oscillate between the flavours with a characteristic frequency. Thus flavour-tagging algorithms are very important for studying mixing and decay of neutral B-meson systems.

1.3. New Physics

Hadron colliders, such as the Large Hadron Collider (LHC) at CERN perform physics at the *Energy Frontier*, colliding particles at ever-increasing energies. SuperKEKB, on the other hand, will perform measurements at the *Intensity Frontier*, colliding particles at high luminosities and gathering high statistics. In addition to searching for CP-violation in the B-meson sector and other B-physics analyses, Belle II has an extenstive New Physics (NP) program that aims to over-confine SM parameters and search for hints of Physics Beyond the Standard Model (BSM). Such measurements include rare decays, strongly suppressed or forbidden in the SM, Electroweak processes and τ-physics. The following subsection gives an overview of the τ-physics program at Belle II which is tightly connected to the ability to construct an efficient track trigger for low charged multiplicity final states.

1.3.1. Lepton Flavour Violation

The discovery of neutrino oscillations was observed in 1998 at Super Kamiokande with the observation of $\nu_\mu \rightarrow \nu_e$. Neutrino oscillations occur when the flavour of a neutrino changes over large propagating distances. Neutrino oscillations necessitate a non-zero mass for the neutrinos and are therefore an extension to the Standard Model, where they are assumed to have zero mass. Neutrino oscillations can then be described in an analogous framework to CP-violation in the quark sector, with a Pontecorvo-Maki-Nakagawa-Sakata (PMNS) matrix to describe the mixing angles [23].

However, evidence for Lepton Flavour Violation (LFV) in the neutrino sector is highly suppressed because of the tiny masses of the neutrinos. Physicists then search for charged LFV, for example in the reaction $\tau \to \mu\gamma$. Belle II is ideal for such searches, since in the clean environments of $\mathrm{e}^+/\mathrm{e}^-$ colliders the initial state is well known.

Luckily, $\tau^+\tau^-$ pairs have a cross-section of about 0.9 nb - the same order of magnitude as $\sigma(\mathrm{B\overline{B}})$. Enough τ-pairs should therefore be generated to hopefully observe LFV at Belle II. Since different BSM theories predict different branching fractions for the τ-decays, it would be possible to discriminate between some of these theories. In the case that no LFV is observed, experimental limits on the branching ratios can be determined. The current branching fraction experimental limits [24] stand at

$$\tau \to \mu\gamma \qquad\qquad 4.4 \times 10^{-8} \tag{1.10}$$

$$\tau \to e\gamma \qquad\qquad 3.3 \times 10^{-8} \tag{1.11}$$

$\mathrm{B\overline{B}}$ events typically produce ≈10 charged tracks, with approximately 3 to 9 of those visible in the L1 track trigger. Since rare τ-events have a low-multiplicity of tracks, the trigger threshold must be lowered below 3 tracks in order to observe these events. Consequently, the L1 track trigger will see more background tracks, which should be suppressed at the first trigger level [3].

2. The Belle II Experiment

The Belle II Experiment is an international high-energy physics experiment located at the High Energy Accelerator Research Organization (KEK) in Tsukuba, Japan. The Belle II detector surrounds the collision point of the asymmetric SuperKEKB collider. Traversing a 3 km circumference, SuperKEKB collides electron-positron (e^+/e^-) pairs at respective beam energies of 7 GeV and 4 GeV. Belle II, following its predeccessors Belle and BaBar, is a second-generation B-factory; large quantities of B mesons and their antiparticles are produced at the $\Upsilon(4S)$ resonance, enabling a precise measurement of CP-violation in the B meson system. This is enabled by the relatively 'clean' environments at SuperKEKB, as opposed to hadron colliders like LHCb [25]. Belle II will perform precision measurements aimed to find deviations from the Standard Model (SM), which would be indicative of new physics [6]. To achieve the target $50\,\mathrm{ab}^{-1}$ of total integrated luminosity, SuperKEKB and Belle II have been upgraded and designed accordingly [26]. This section will outline SuperKEKB's upgrade and design features, the various Belle II subdetectors and their principle of operation, and some of the expected sources of background, important to understand for an optimized design of experimental triggers.

2.1. SuperKEKB

SuperKEKB is an asymmetric e^+/e^- collider, where asymmetric refers to the electron and positron energies of 7 GeV and 4 GeV, respectively, corresponding to a COM of the $\Upsilon(4S)$ mass. The enrgies can be tuned also reach higher and lower COMs. The double ring collider consists of an electron High Energy Ring (HER) and a positron Low Energy Ring (LER). The linear accelerator (LINAC) and 1 GeV positron Damping Ring (DR) provide the particles to be injected into the main rings. The LINAC accelerates bunches of e^+/e^- to their target energies before they are injected into bunches in their respective storage rings. Positrons are passed through the DR before injection into the LER since they are produced with too large an emittance for the nanobeam scheme. The electron and positron bunches collide at the Interaction Point (IP) at an angle of 83 mrad, allowing for ease of beam separation before and after collisions [9, 1]. A schematic of the SuperKEKB collider can be seen in Figure 2.1.

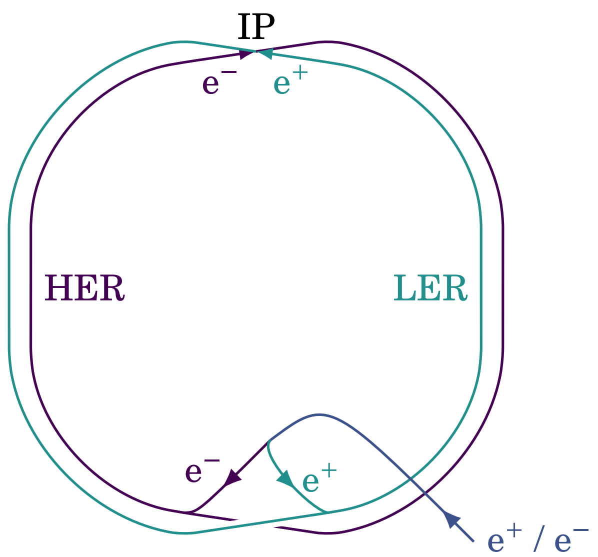


Figure 2.1.: Schematic of SuperKEKB collider located at KEK, Tsukuba - The Belle II detector surrounds the collision point of SuperKEKB

2.1.1. Beam Energies

SuperKEKB will operate primarily at the $\Upsilon(4S)$ resonance, which is the fourth radially-excited s-wave state of the Υ meson. Υ is the $\mathrm{b\bar{b}}$ quarkonium, and at the $\Upsilon(4S)$ resonance, is just above the production threshold for B-meson pair production ($\mathrm{m_B} = 5.28\,\mathrm{GeV}\,c_0{}^{-2}$, $\mathrm{m}_{\Upsilon(4S)} = 10.58\,\mathrm{GeV}\,c_0{}^{-2}$). $\Upsilon(4S)$ therefore has a 96% chance of decaying to B meson pairs. B-meson pairs are produced almost at rest in the COM frame, meaning that the contributing momentum is almost entirely due to the boost of the asymmetric colliding energies. This nicely allows one to directly relate the distance Δz between the decay vertices to the decay time difference Δt:

$$\Delta t = \frac{\Delta z}{c\beta\gamma} \tag{2.1}$$

where c is the speed of light, γ is the Lorentz factor and β is the boost, which for a head-on collision is given by:

$$\beta = \frac{E_{HER} - E_{LER}}{E_{HER} + E_{LER}} R_{\phi_c} \tag{2.2}$$

where R_{ϕ_c} is a correction factor, accounting for the fact that the boost is actually slightly larger because of the finite crossing angle, ϕ_c of the $\mathrm{e}^+/\mathrm{e}^-$ beams. In order to achieve the design luminosity, the boost at SuperKEKB is reduced by a factor of $\frac{2}{3}$ relative to KEKB. The average spatial distance Δz between the B-meson decay vertices will consequently become smaller, but this is expected to be compensated, or even improved compared to KEKB, by the improved resolution of the vertex detector system (VXD, see below), as well as providing advantage for those decays with neutrinos in the final state which require good detector hermiticity [6].

2.1.2. Luminosity

SuperKEKB is designed to reach peak instantaneous luminosities of $8 \times 10^{35}\,\mathrm{cm}^{-2}\,\mathrm{s}^{-1}$ - up to 40 times higher than its predecessor KEKB - and aims to collect over its lifetime over $50\,\mathrm{ab}^{-1}$ of data. For two colliding beams the luminosity may be written as

$$\mathcal{L} = \frac{N_+ N_- f}{4\pi\sigma_x\sigma_y}.R_L \propto \frac{I_+ I_-}{\sigma_x\sigma_y} \tag{2.3}$$

where the two transverse beam profiles are modelled as Gaussians with horizontal size σ_x and vertical size σ_y, N are the number of particles in the respective $\mathrm{e}^+/\mathrm{e}^-$ bunch, f is the crossing frequency of the bunches and R_L is a crossing factor which takes into account geometrical effects of the finite crossing angle and bunch length. One can see that the luminosity can therefore be increased by increasing the beam currents and by reducing the beam size.

Ideal particle trajectories perform closed orbits, returning to their initial start points after one revolution in the accelerator ring. In reality, however, these particles undergo oscillations around this closed orbit, so-called *betatron oscillations.* The horizontal and vertical beam sizes can then be written as

$$\sigma_{x,y} = \sqrt{\epsilon_{x,y}\beta_{x,y}(s)} \tag{2.4}$$

where ϵ is the emittance which measures how much the particles deviate from the ideal trajectory related to the beam divergence at initial injection and $\beta_{x,y}(s)$ is the so-called β-function which varies according to the position s of the ideal orbit, and can be thought of as an envelope around all possible particle trajectories inside the beam. This β-function depends strongly on the guide-field of the magnets around the ring and most importantly on the quadrupole magnets before and after the IP used to squeeze the beam.

The luminosity can be written in terms of the vertical beam-beam parameters ξ_y, which have the following scaling behaviour:

$$\mathcal{L} \propto \frac{I_\pm \xi_{y\pm}}{\beta_y} \qquad \xi_{y\pm} \propto \frac{N_\mp \beta_y}{\sigma_y\sigma_x} \tag{2.5}$$

ξ_y describes the focusing force exerted on a bunch by the EM field of the opposite bunch. At low beam currents, ξ_y increases with the number of particles in a bunch. As currents increase, however, the beams influence each other to increase the emittance, which in turn increases the beam size. The ξ_y is said to saturate at this *beam-beam limit.* At this limit, the luminosity will depend only on the beam currents.

The novel nanobeam scheme, new to Belle II, avoids the problem of the *hourglass effect* that arises with the increase of our vertical β-function parameter by maximally reducing the overlap d of the beams down to 0.3 mm, 20 times smaller than KEKB. This is achieved by the use of superconducting quadrupole magnets close to the IP. The

	KEKB	**SuperKEKB**
E_{LER}	3.5 GeV	4 GeV
E_{HER}	8 GeV	7 GeV
$\mathcal{L}$	$2.11 \times 10^{34}\,\mathrm{cm}^{-2}\,\mathrm{s}^{-1}$	$8 \times 10^{35}\,\mathrm{cm}^{-2}\,\mathrm{s}^{-1}$
σ_x	103/123 µm	10.2/7.5 µm
σ_y	2.9 µm	59 nm
ϕ_c	22 mrad	83 mrad
boost $\beta\gamma$	0.425	0.287

Table 2.1.: Summary of main differences between KEKB and SuperKEKB machine parameters [1]

hourglass effect arises when the bunch length $d > \beta_y$, effectively reducing the luminosity. The nanobeam scheme requires the beams to cross at a crossing angle ϕ_c, leading to an effective bunch length of

$$d = \frac{\sigma_x}{sin\phi_c} \tag{2.6}$$

and with the horizontal beam size $\sigma_x \approx 10\,\mu\mathrm{m}$, $\phi_c = 83\,\mathrm{mrad}$, the hourglass requirement is satisfied with $d \leqslant \beta_y$. Table 2.1 summarises the main differences between beam parameters of KEKB and SuperKEKB. A schematic of the nanobeam scheme can be seen in Figure 2.2. The first e^+/e^- collisions were in March 2018, and March 2019 saw the first collisions with full detector geometry [27, 28].

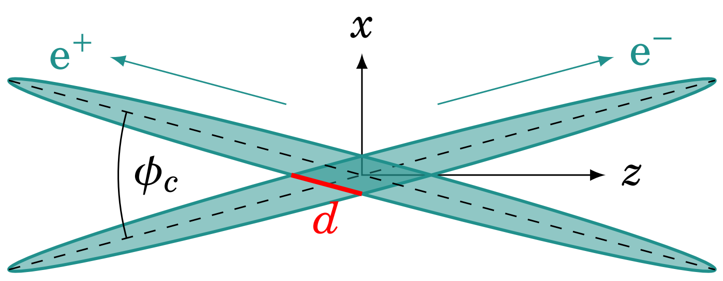


Figure 2.2.: The nanobeam scheme new to Belle II

2.2. Belle II Detector

The Belle II Detector is built to surround the interaction region of SuperKEKB and consists of several sub-detectors, all playing a role in particle tracking and identification. The Belle II detector has a similar principle design to Belle, but has been upgraded in consideration of the higher currents, decreased beam sizes, modified interaction region and the expected sizeable increase in background [1]. Figure 2.3 shows a schematic of the Belle II detector.

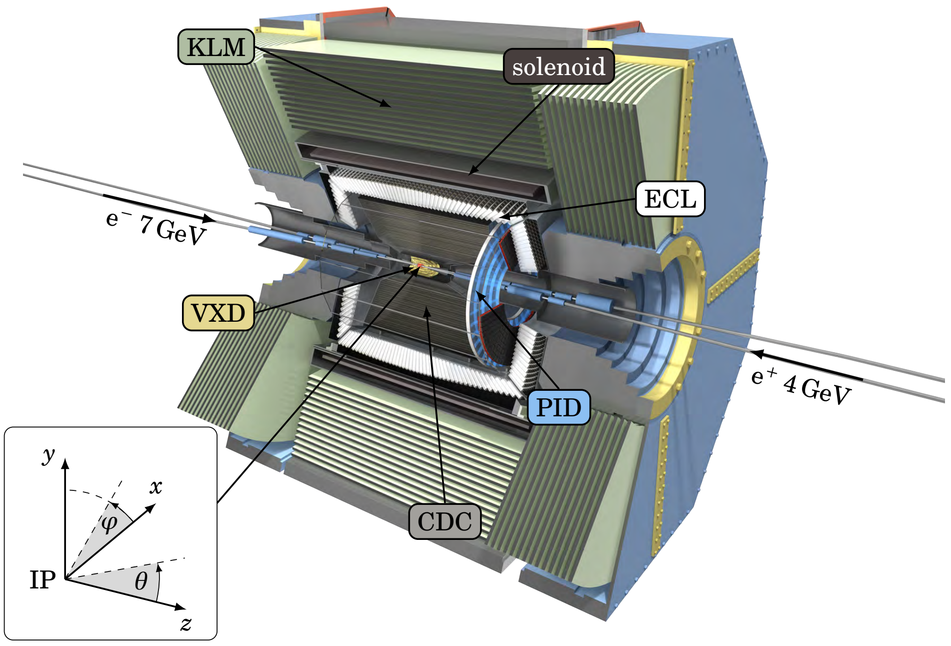


Figure 2.3.: THe Belle II detector and the coordinate system of Belle II

2.2.1. Beampipe

The beampipe follows the same principle design of Belle, consisting of two cylindrical Beryllium layers, separated by a gap for coolant and lined internally with gold plate to shield the detector from low energy X-rays. Beampipe radii are summarised in Table 2.2 [1].

Gold plate	Thickness	10 µm
Inner Be pipe	Inner radius	10.0 mm
	Thickness	0.6 mm
Gap for coolant	Thickness	1.0 mm
Outer Be pipe	Outer radius	12.0 mm
	Thickness	0.4 mm

Table 2.2.: Beampipe design parameters

2.2.2. Vertex Detector

The Vertex Detector (VXD) has been upgraded from Belle to consist of a novel Pixel Detector (PXD) in addition to the Silicon Strip Detector (SVD). The PXD, based on DEPFET (DEPleted Field Effect Transistor) technology, is now the innermost sub-detector and directly surrounds the Interaction Point (IP). The VXD consists of 2 layers of PXD surrounded by 4 layers of SVD. Silicon strip layers alone were no longer sufficient for the IP due to the large occupancy caused by the luminosity upgrade [1]. Because of the reduced beam pipe radius, the first two detector layers are closer to the IP, and the outermost layer is also placed at a considerably larger radius. Subsequently, Belle II expects to see significant improvement in the vertex resolution, as well as in the reconstruction efficiency for $K_S^0 \to \pi^+\pi^-$ decays that have hits in the VXD [6].

2.2.3. Central Drift Chamber

The Central Drift Chamber (CDC) is the main tracking detector in the Belle II Experiment. The main purpose of the CDC is to precisely measure the momenta of charged particles by reconstructing charged tracks which curve in the presence of the 1.5 T field provided by the Belle superconducting solenoid.. In addition, it provides particle identification by measuring energy loss in its gas volume (the CDC can identify low momentum tracks that do not reach other sub-detectors). It is the sole input to the first level (L1) track trigger [1].

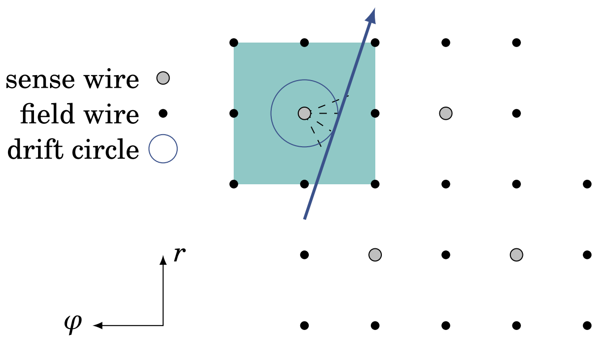


Figure 2.4.: Each sense wire is surrounded by 8 field wires to make up a ‘drift cell’

The CDC is a wire chamber consisting of 42240 field wires and 14366 sense wires containing a special gas mixture (50% C_2H_6 and 50% He). The wires are arranged radially in rectangular cells, with 8 field wires surrounding each sense wire and a high voltage applied between them. Charged particles passing through the chamber ionize the gas, as illustrated in Figure 2.4. Free electrons drift towards the sense wires, ionizing more gas atoms in the high electrical field surrounding the wire. This electron avalanche is registered as a hit whenever it reaches the sense wire, and a drift time can be determined (the time taken for the charged particle to reach the sense wire). In fact, the wire voltages and the gas mixture were specially selected so that the drift velocity remains

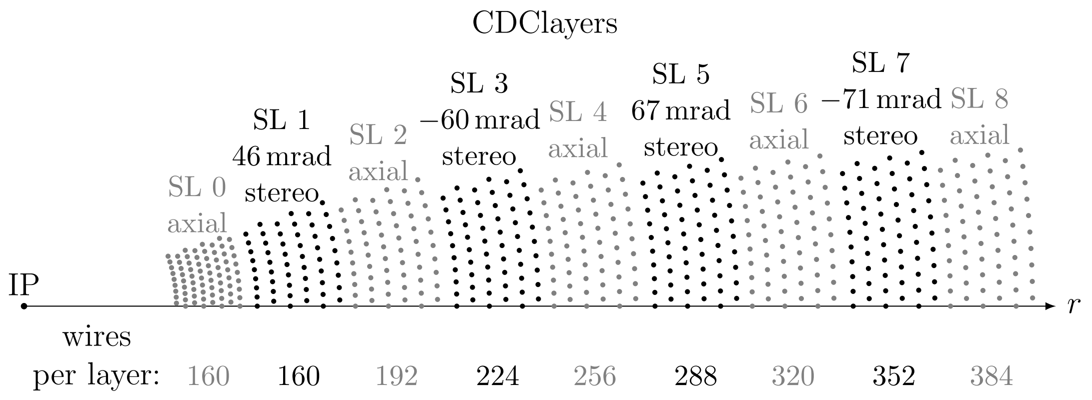


Figure 2.5.: Layer configuration of the CDC with 9 superlayers - stereo angles within a superlayer vary by a few mrad; the given numbers are average values [9]

almost constant at 40 µm ns^{-1} for a wide range of locations inside the drift cells. Each sense wire in the chamber is assigned a unique ID number.

The 56 layers of wires in the chamber are arranged into 9 Superlayers (SL) totalling a cylindrical volume with an outer radius of 113 cm and an inner radius of 16 cm. The innermost SL consists of 8 layers of wires to cope with the higher background near the IP. The remaining SL have 6 layers of drift cells each. SL alternate between axial and stereo orientations as shown in Figure 2.5 and Figure 2.6. Axial wires are parallel to the z-axis, whilst stereo wires are inclined/skewed with respect to the beamline, allowing for a 3D reconstruction of tracks. The sign of the stereo wires alternate according to the so-called U,V orientations, giving the total configuration of all SL as AUAVAUAVA, with A the axial SL. Stereo SLs are skewed between 45.0 mrad to 74.0 mrad.

The measured CDC spatial resolution is ≈100 µm. Viewed from the IP, the CDC covers a flat polar angle range of [17°, 150°]. The outer SL, again viewed from the IP, covers a polar angle of only [35°, 123°].

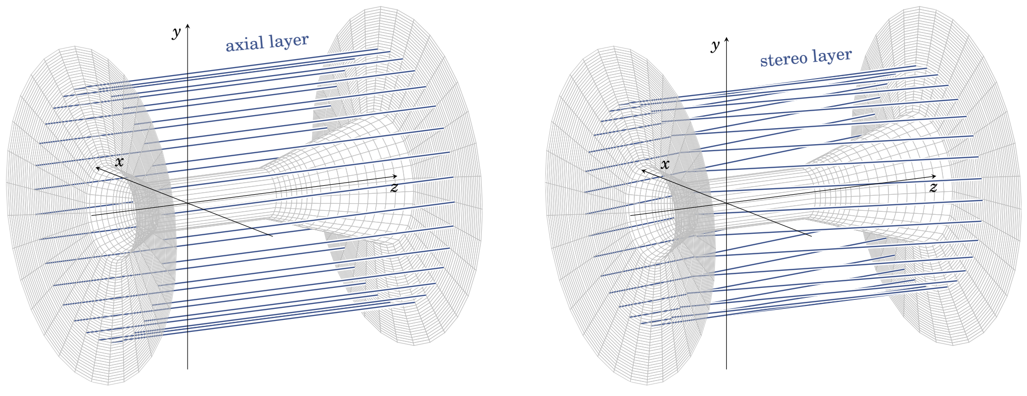


Figure 2.6.: Wire orientations of CDC: (a) Axial wires are parallel to beamline (z-axis); (b) Stereo wires are skewed with respect to beamline

2.2.4. Particle Identification System

In B factory detectors such as Belle II, particle identification (PID) is necessary for B-meson flavour tagging and to suppress background in precision measurements of B and D decays [29]. The newly developed PID system for Belle II consists of two independent Cherenkov detectors: a Time-of-Propogation (TOP) counter in the barrel region and an aerogel ring-imaging Cherenkov (ARICH) counter in the forward endcap region. Their main task is to improve kaon and pion identification capabilities of the CDC by additionally imaging the Cherenkov rings produced [1] .

2.2.5. Electromagnetic Calorimeter

The goal of the Electromagnetic Calorimeter (ECL) is to detect photons and to identify electrons in order to discriminate them from hadrons, and in particular pions. High energy resolution and detection efficiency of photons is important in Belle II since one third of B-decay products are π^0's and other neutral products that produce photons [30]. It consists of a highly-segmented array of thallium doped caesium iodide CsI(TI) crystals in the barrel, forward and backward end-caps. CsI(TI) was chosen because of its ability to provide high light output via the thallium doping and because it has short radiation length of the CsI crystal [30].

2.2.6. K$_L$-Muon Detector

Based on Resistive Plate Chambers (RPCs) and newly installed scintillator planes for the innermost layers, the new K$_L$-Muon (KLM) Detector is used in the barrel and endcap regions of Belle II and is the outermost detector. Designed to detect long-lived K_L^0's and muons and positioned outside of the solenoid coil, it consists of 4.7 cm thick iron plates alternating with active detector components, where the muons are visible as tracks. Muon tracks in KLM can then be associated to a track in the CDC. The iron plates act as an absorber for hadronic particles and as the magnetic flux return yoke of the magnet. The ECL is surrounded by a superconducting coil, providing a field of 1.5 T parallel to the axial wires of the CDC (”the z-axis” in the Belle II coordinate system).

2.3. Beam-induced Background

Background rates at Belle II are expected to increase dramatically due to the luminosity upgrade. In order to discriminate background from real physics events, it is important to understand the different sources of background and their respective rates. Background sources at Belle II can in general be classified into two categories - collision-induced background (non-interesting physics plus secondary particles from non-relevant QED events such as Bhabha scattering) and beam-induced background (background originating from collisions of beam particles with gas molecules in the evacuated beam pipe or with components of the accelerator structures (beam pipe and magnet components)). Beam-induced sources of background are discussed in this section.

2.3.1. Beam-gas Scattering

Beam-gas scattering is caused by the scattering of beam particles by residual gas molecules in the beam pipe [6]. Bremsstrahlung and Coulomb radiations are the dominating sources of beam-induced gas background. Bremsstrahlung radiation refers to the emitted electromagnetic energy of a charged particle as it decelerates in the proximity of another charged high Z gas nucleus particle. Coulomb scattering refers to the scattering of charged particles as they come into contact with one of the gas nuclei at larger distances. Both these processes change the momenta of the beam particles - Bremsstrahlung decreases the energy of the beam particles whereas Coulomb scattering changes their direction. This enables vacuum chamber and magnet collisions which in turn produce particle showers. The size of these backgrounds depend on the beam current, the vacuum pressure in the rings and the material surrounding the magnets. Due to the very small radius of the beampipe at IP, the vacuum level around the IP is expected to be 100 to 1000 times worse than at KEKB [1]. The beam-gas Coulomb scattering rate is also expected to be a factor of 100 higher than at KEKB [6].

2.3.2. Touschek scattering

Another source of background is Touschek scattering [1]. Touschek scattering is intra-bunch scattering which changes the momenta of beam particles so that they can collide with the pipes and magnets, producing particle showers. This source of background is enhanced at SuperKEKB due to the new Nanobeam scheme [6]. This background is proportional to the beam bunch current, the number of bunches, and the inverse of the beam size and is studied by varying the HER or LER beam size [1]. Touschek background at Belle II is expected to be 20 times higher than at KEKB [6].

2.3.3. Bhabha scattering

Bhabha scattering refers to elastic scattering of electrons on positrons ($e^+e^- \rightarrow e^+e^-$), and was the dominant source of all backgrounds in Belle. Photons from these radiative Bhabha events propagate along the beam axis direction and interact with the iron in the magnets, which produces a large amount of neutrons via the giant photo-nuclear resonance mechanism (the main background source for the KLM). Belle II's Bhabha rate is expected to be much lower than at KEKB because two separate quadrapole magnets are used. The Bhabha scattering rate is in fact used to measure luminosity since it is easy to identify and is a well understood QED process [1]. There are two Bhabha (further QED) channels that contribute in leading order:

2.3.4. Synchrotron Radiation

Synchrotron Radiation (SR) emitted from the beam is proportional to the beam energy squared and the magnetic field strength squared. Therefore the HER is the main source of SR. The inner surface of the beryllium beam pipe is coated with a gold layer to absorb

SR photons before they reach the VXD, since the SVD in Belle was severely damaged by SR photons with energies on the order of a few keV [6].

2.3.5. Two-photon Processes

A fifth contribution to beam background is the low momentum pair production $e^+\mathrm{e}^- \to \mathrm{e}^+e^-e^+e^-$ via the two-photon process. Due to the very large cross section for low secondary e^+e^- pairs, this process can inject a large particle flux and consequently leave many hits in the inner Belle II detectors, and is the dominant source of background for the VXD [31].

2.3.6. Vacuum Scrubbing

Ultra-high vacuum is needed in particle accelerator rings in order to minimise the rate of beam-gas collisions. Vacuum scrubbing is a process by which the vacuum is improved over time, often months are needed.. Vacuum scrubbing works by simply allowing ‘wide’, unfocused beams with high currents to circulate in the rings in order to ‘knock out’ adsorbed molecules from the beampipe. Due to the upgraded LER, background is especially high there.

2.3.7. Background mixing

For this thesis, it is important to distinguish between two types of background: background *hits* and background *tracks*. Background tracks originate from processes described above and therefore do not come from the IP like the tracks from real physics events. If no information on the origin of these tracks is available at the trigger level, they will be considered as ‘real physics event’ and only recognized afterwards on the reconstruction level. The elimination of these background tracks already at the trigger level is the subject of this thesis. Single background hits, on the other hand, do not amount to tracks. Belle II expects to see a high number of these additional background hits, which may decrease the efficiency of the track finding in the trigger. Because of this problem the effects induced by the background hits need to be investigated carefully. Simulation studies allow for a *mixing* of these background hits before CDC Digitization (the response of the CDC electronics) overlain with physics and real background events[9].

In this thesis, the terms ‘Phase 2’ and ‘Phase 3’ backgrounds are used to indicate the background campaigns simulated for expected levels of background at full luminosity of the respective Phase, which essentially amounts to more background hits mixed corresponding to the increased luminosity. Phase 3 is the first Physics run of the Belle II Experiment which began in March 2019 with full detector geometry [28]. Since Phase 3 has not yet reached full luminosity, Phase 2 is expected to suffice for the present studies. See section 5.2 for related studies.

3. Belle II Trigger System

In order to deal with the copious amount of beam-induced background and select events for physics analysis with high efficiency, the Belle II trigger system is partitioned into two consecutive levels. The first level (L1) trigger, implemented in deadtime-free pipelined hardware, performs a partial online event reconstruction and sends a signal to the Data Acquisition system (DAQ) for full event readout whenever certain criteria are satisfied. The High Level Trigger (HLT) is implemented in software and performs a more detailed selection on the events triggered by the L1 trigger in order to further reduce the background among the events that are written to disk. A summary of the differences in the trigger levels are shown in Table 3.1. This chapter gives an overview of the L1 track trigger pipeline modules and requirements. Details on the HLT can be found in [32].

L1 Trigger	**HLT**
Hardware Max. avg trigger rate 30 k Event selection on based on incomplete data provided by hardware trigger processors	Software Max. avg rate written to disk 10 kHz Event selection based on the complete event data from L1 Trigger

Table 3.1.: Tasks and properties of the L1 and HLT trigger systems

3.1. L1 Track Trigger Pipeline

Since the L1 trigger operates online, it is important that it fulfils latency requirements in order to process the events in a deadtime-free way. The full L1 trigger is designed for a maximum average trigger rate of 30 kHz and a fixed latency budget of 5 µs. The L1 hardware trigger is divided into four sub-trigger components. The sub-triggers send information about an event in a given clock cycle to the Global Decision Logic (GDL), which makes a decision based on the outputs of the various sub-detector trigger processors on whether or not to stop the pipeline and send the event to the HLT within the 5 µs window. The main L1 trigger of Belle II is the CDC track trigger. In addition to the CDC trigger, the ECL trigger provides information about energy deposition in the calorimeter, the TOP trigger provides precise timing information and the KLM trigger gives muon track information. The vertex detector readout is too slow to provide data for the L1 trigger.

The CDC trigger, or track trigger, is the most complicated subtrigger system in Belle II and defines the latency of the L1 trigger due to the long drift times of the wire signals.

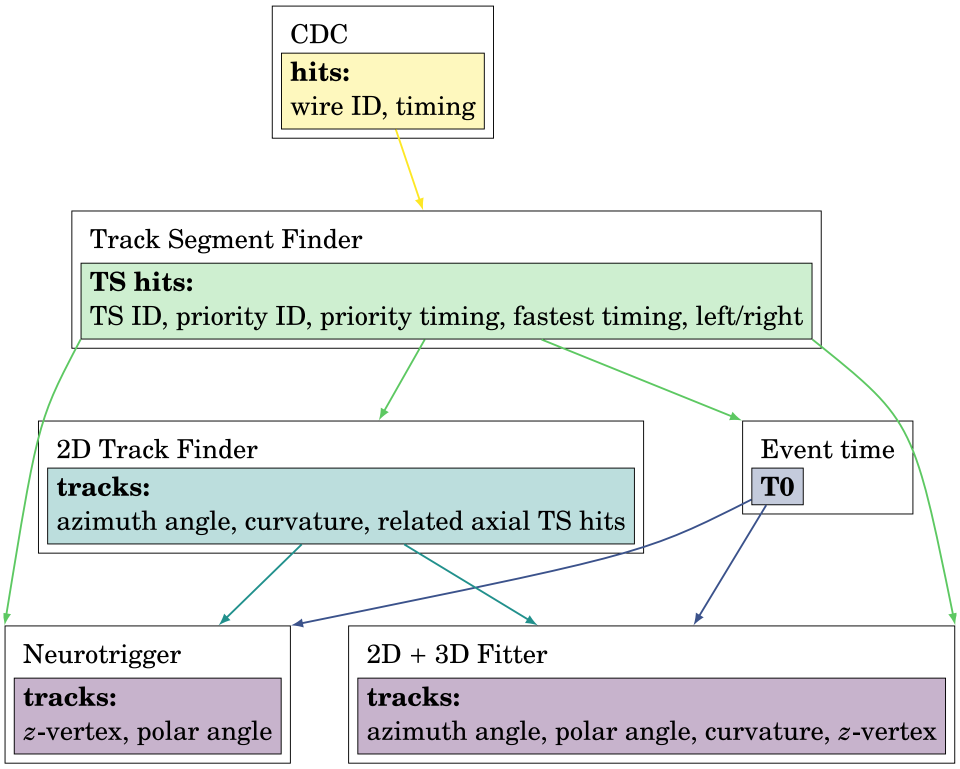

CDC
hits:
wire ID, timing
Track Segment Finder
TS hits:
TS ID, priority ID, priority timing, fastest timing, left/right
2D Track Finder
tracks:
azimuth angle, curvature, related axial TS hits
Event time
T0
Neurotrigger
tracks:
z-vertex, polar angle
2D + 3D Fitter
tracks:
azimuth angle, polar angle, curvature, z-vertex

Figure 3.1.: The first level track trigger modular pipeline

The track trigger processes four modules in its pipeline. The *Track Segment Finder* (TSF) combines hits in a superlayer to wire patterns characteristic for a short track in a given SL and thus minimizes the amount of data sent to the next module. The *2D Finder* then combines track segments from the axial SLs to tracks in the transverse plane using a Hough transformation. The *Event Time Finder* module determines the event time in parallel to the 2D Finder to calculate drift times for precise spatial information from the hits. Finally, the *Neurotrigger* module estimates the 3D track parameters based on a neural network approach, which is the focus of this thesis. A schematic of the L1 track trigger sub-components is shown in Figure 3.1. This section outlines the modules of the pipelined L1 track trigger, which takes CDC hits as input, and outputs low-level tracking information to the GDL.

3.1.1. Track Segment Finder

The Track Segment Finder (TSF) is the first module in the L1 track trigger pipeline. The TSF takes as input raw CDC hits and outputs so-called *track segments* by way of data reduction. Any hits not satisfying the criteria for a track segment (TS) hit are neglected already at this first processing step and not output to the next module. The TSF then simultaneously suppresses noise from isolated hits whilst additionally compressing CDC raw data for input to subsequent modules in the L1 trigger pipeline.

A TS is a collection of certain pre-defined wire patterns. A TS hit is then defined as a hit pattern which fulfils the TS wire pattern. TS hit pattern shapes can be seen in Figure 3.2. All TS hit patterns have the same hourglass shape (except for the first SL (SL0), which has a different pattern since it has 8 wires). A TS hit is registered when there are at least 4 hits found in 5 of the different TS layers and the hit pattern fulfils a straight track pattern within the hourglass shape. Each TS is assigned a so-called priority wire. The priority wire ID combined with the SL ID will later be used as a reference to the particular TS pattern. For the outer SL, a TS hit is defined as the first priority wire ID when the central wire in the hourglass pattern has a hit. Otherwise, a second priority wire is defined in the case that this hit is not present, as shown in Figure 3.2. A TS hit is then the first or second priority wire hit in a TS. But from then on a specific wire in the TS is used for further processing.

Additionally, the hit pattern may allow one to determine on which side of the priority wire the track has passed. The track orientation is labelled as left, right or undecided (left/right information). Possible hit pattern arrangements and their determined left/right information are stored in a TS Lookup Table (LUT) and loaded to the hardware. Figure 3.3 shows examples of track segments with left, right or undeceided left/right information.

The TSF then outputs priority wire IDs, drift times of a reference wire in the TS (priority time) with a resolution of 2 ns and the position of the track relative to the priority wire [33]. Precise spatial information is obtained from the drift time of a priority wire in the TS (see Figure 2.4). This drift time is a distance measure of the track to the wire and is used as input to the neural network.

The TSF efficiency inevitably depends on the crossing angle α of the track, as shown in [3]. The TSF is not sensitive to $\alpha < 30°$.

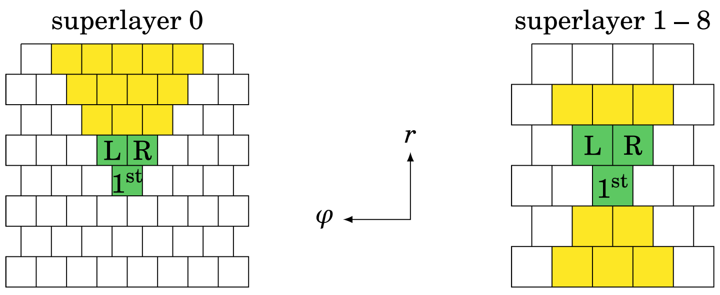


Figure 3.2.: Track Segment shapes for the SLs [9]

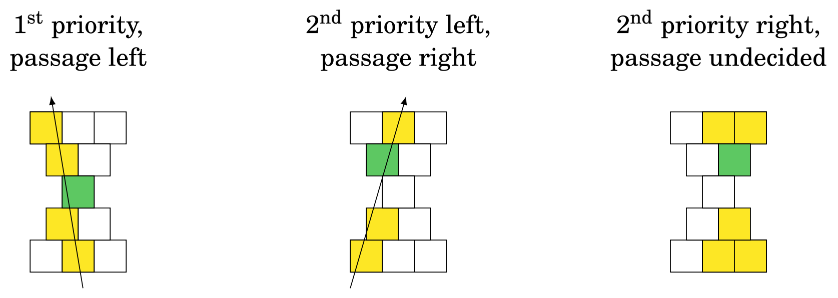

1st priority, passage left
2nd priority left, passage right
2nd priority right, passage undecided

Figure 3.3.: TS left/right information is determined from TS hit patterns and can be read from the TS LUT [9]

3.1.2. 2D Finder

The next step is to combine track segment hits to 2D tracks in the transverse $x-y$ plane. Therefore only the axial TS hits are used for this step. The track finding algorithm used for this step is the Hough transformation procedure. Hits on axial wires coming from curved tracks are transformed into curves in the Hough Plane. Tracks are then found as the crossing point of several curves in $\rho-\phi$ space [9], where ϕ is the azimuthal angle and ρ is the track curvature which is inversely proportional to the transverse momentum p_t. This track-finding step determines the number of tracks and track parameters, given by the crossing point coordinates in Hough space.

Tracks in the CDC are curved due to the presence of a magnetic field. The bending radius r can be calculated as:

$$r = \frac{p_t}{cB} \tag{3.1}$$

where p_t is the transverse momentum, c is the speed of light and B is the magnetic field. Tracks in Hough space can then be modelled as sinusoidal curves.

The transformation is then given by

$$\rho(\phi) = \frac{2}{r_{TS}} sin(\phi - \phi_{TS}) \tag{3.2}$$

where (r_{TS}, ϕ_{TS}) are the polar coordinates of the priority wire in the TS. Two crossing points are found for a given track, with one crossing point corresponding to a track moving in the clockwise direction, the other in the anti-clockwise direction. This crossing point ambiguity can be solved by eliminating the half of the Hough curve that corresponds to tracks curling back through the points by removing the parts of the sine curve that have negative gradient. The condition then for the remaining outgoing half is

$$\phi \in [\phi_{TS} - 90, \phi_{TS} + 90] \tag{3.3}$$

The charge of the track is obtained from the crossing point as the sign(ρ). In order to find the crossing point for a track, a grid of 160 (ϕ) × 34 (ρ) cells is defined in the parameter space. The number of crossing Hough curves in each cell from different SL is counted in parallel, and the crossing point candidates are obtained as grid cells with at least four curves contained. This corresponds to the 2D Finder imposing a condition of at least four SL TS hits for a given 2D track. The neighbouring grid cell candidates are clustered and the crossing point is then selected as the cluster centre. The cluster centre coordinates give the track parameters as ϕ and ρ.

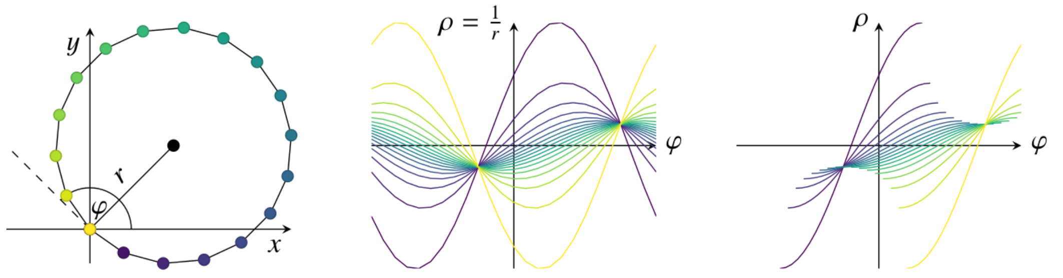


Figure 3.4.: 2D Hough transformation in the transverse plane: points in geometrical space are transformed to curves in the parameter space [9]

The number of grid cells has been optimized in order to get a high track parameter resolution whilst maintaining a low clone rate. Clones occur when gaps between clusters emerge and give two track candidate clusters rather than one. With the selected setup, the clone rate is kept to only 0.16 % [9].

The polar angle acceptance for the 2D tracks is $\theta \in [31°, 126°]$, which is related to the requirement of at least 4 present axial SL hits. The dependence on p_t is related to the crossing angle and TS Finder efficiency. Note that due to the limited TS acceptance, curling tracks cannot be found in L1 trigger.

A 3D Hough transformation step is being investigated as means to improve the track finding algorithm. Details of the algorithm can be found in [34].

3.1.3. Event Time Finder

The event time for a given event starts, in principle, the clock for determining the drift times in the individual CDC drift cells. This so-called 'T_0' can be derived from the fast timing detector in Belle II, such as the TOP counter or the electromagnetic calorimeter. However, for the pipelined trigger operation at the first trigger level, this 'global T_0' information from other subsystems is not available presently to the track trigger itself. Therefore the 'event time' is derived from the CDC alone. The most precise T_0 in the CDC is given by the first signal on any of the sense wires of the chamber. T_0 is an important step in determining drift times which are input to the Neurotrigger (see below). Drift times (t_{drift}) for each of the 9 SL are determined (at the Neurotrigger preprocessing) by

$$\text{drift time}_{\text{drift,SL}} = \text{priority time}_{\text{SL}} - T_0 \tag{3.4}$$

where the priority time$_{\text{SL}}$ is the timing of the priority wire in a SL TS.

The Event Time Finder (ETF) module operates in parallel to the 2D Finder to determine an event time (T_0) by comparing the fastest drift times of all TS fastest timings. The TS fastest timing is the shortest drift time, or the first hit, of all hits in a TS. The ETF outputs a T_0 only if there are a certain number of track segments *active* in a given time window (currently 32 ns).

The number of track segments active in this time window is defined by a threshold value. A threshold value of 0 requires 1 TS active in the timing window, a value of 1 requires 2 TS active in the timing window and so on. Whenever the ETF does not output an event time, a fallback option may be specified, such as using the fastest priority time (timing of the first priority wire hit in a 2D track). The selected event time input and threshold values were studied and found to have a significant impact on the resolution of the Neurotrigger output. Performance studies of the ETF are outlined in Section 5.3.

3.1.4. Neurotrigger

The final step in the L1 track trigger is a full 3D event reconstruction for each 2D Finder track. In order to sufficiently reject background events which do not originate from the IP, a z-vertex L1 Neural Network trigger (Neurotrigger), realized by neural methods, is implemented in the hardware. Using information from 2D tracks and related stereo hits as input, the Neurotrigger estimates estimates a z-vertex position and polar angle θ for the track. This output is sent to the GDL.

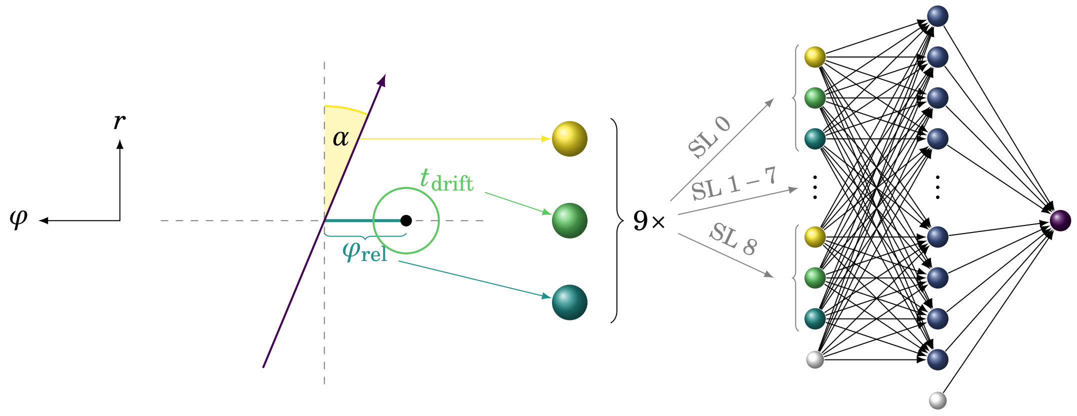


Figure 3.5.: Three inputs to the Neurotrigger for each of the 9 SL are determined from the spatial information related to the track [9]

To estimate the origin of the track along the z-axis (z-vertex), the Neurotrigger takes a tuple of inputs for each SL. For each TS hit, 3 input values are defined:

1. φ_{rel}

All angles are measured relative to the 2D track parameters so that the reconstruction is symmetric in φ, the azimuthal angle. φ_{rel} is then the azimuthal angle of the wire φ_{wire}

relative to the crossing point φ_{cross} (see Figure 3.6) of the 2D track and is calculated by:

$$\varphi_{\text{cross}} = (\varphi_{\text{track}} - \alpha) \cdot \frac{N^{\text{SL}_{\text{wires}}}}{2\pi} \tag{3.5}$$

$$\varphi_{\text{rel}} = \varphi_{\text{wire}} - \varphi_{\text{cross}} \tag{3.6}$$

φ_{cross} is the φ-position where the track crosses the wire layer. φ_{wire} is the φ-position of the reference wire of the TS hit. φ_{rel} is required to calculate the z-vertex position for stereo wires; it contains the z-position of the hit at the sense wire.

2. $\pm \text{t}_{\text{drift}}$

The drift time of the track to the wire is in principle calculated from the ETF input. It is hoped that the Neurotrigger learns the non-linear correction of the drift time to the drift length relations. The left/right information is input to the Neurotrigger as a sign (+ for right, − for left). If there was no left/right information for the TS, the drift time input is set to 0, i.e. the track is assumed to pass very close to the wire.

3. $\alpha = \frac{s}{2r}$

The crossing angle α (another crossing angle, see Figure 3.5) is required for extrapolating the track to the z-vertex and for precisely calculating the hit position. α can be calculated as the ratio of the curvature s of the track to its radius of curvature r (equivalent to Equation 3.7). The input calculation to the Neurotrigger is equivalent and calculated by:

$$\alpha^{\text{SL, pr}} = \arcsin \frac{r^{\text{SL, pr}} \cdot \omega}{2} \tag{3.7}$$

where $\alpha^{\text{SL, pr}}$ is the crossing angle for the SL and priority layer (changes if we have a second priority hit), $r^{\text{SL, pr}}$ is the radius depending on the SL and priority layer, $\text{ID}_{\text{ref}}^{\text{SL, pr}}$ is the ID of the reference wire related to φ_{rel}, φ_{track} is the ϕ-value of the track predicted by the 2D finder and $N^{\text{SL}_{\text{wires}}}$ are the number of wires in a SL [9]. α is related to the p_t provided by the 2D Finder.

Of the 5 µs latency budget of the L1 trigger, 1 µs is assigned to the Neurotrigger, of which ∼600 ns is dominated by I/O. The Neurotrigger is implemented with neural networks on FPGAs and the architecture is outlined in Chapter 4. Chapter 5 details software simulation studies of the Neurotrigger.

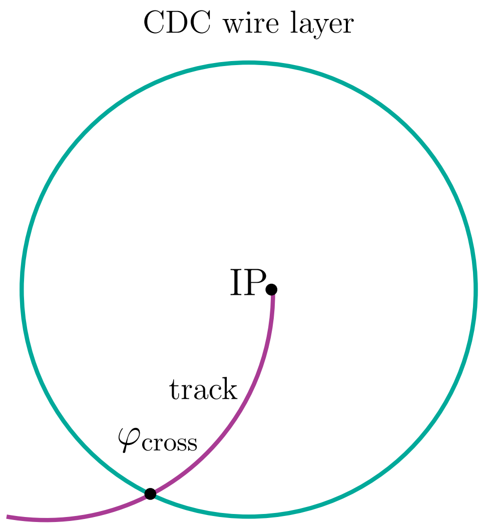


Figure 3.6.: The crossing point of the CDC wire layer with a 2D track in the transverse plane

3.1.5. Global Decision Logic

The GDL receive the output from all the L1 subtrigger systems and makes a final trigger decision on an event. Several independent trigger logics are defined - if one is fulfilled, a trigger signal will be sent. In addition, veto logics are defined. For example, an event with low track multiplicity can be combined with z-vertex veto to reject background events. The GDL will also be responsible for implementing a z-cut, deciding which events will be labeled as background by the veto.

3.2. Hardware Implementation

The pipeline modules are implemented on FPGA configurable hardware in a pipelined way to allow a dead-time free system with a total latency of 5 µs and a maximum average trigger rate of 30 kHz. FPGAs were chosen for their high parallelization ability and deterministic runtime. The input values for the neurotrigger are calculated in a preprocessing step on FPGA. The neural network is however trained offline, with weights uploaded to the FPGA for triggering on.

4. Artificial Neural Networks

The Neurotrigger described in Section 3.1.4 is the 3D track finding step of the L1 track trigger. The Neurotrigger takes a neural approach to estimate 3D track parameters with artificial neural networks (herein referred to as neural networks) loaded onto FPGA hardware and operating in realtime. The neural network approach to the z-Vertex trigger is new to Belle II and was not realized in the trigger system of Belle, which only had a 2D version of the track trigger. The neural approach was selected for Belle II because of the special ability of neural networks to learn non-linear correlations in data. Furthermore, owing to their massively parallel architecture, complex neural networks can be executed in a very short time. The section provides an introduction to neural networks, the Neurotrigger architecture and the training procedure by which weights are determined.

4.1. Multi Layer Perceptron

Neural networks (NN) are a class of machine learning algorithms inspired by the neural structure of the human brain. Machine learning methods can be broadly classed into *supervised* or *unsupervised* methods. The Neurotrigger uses a supervised learning approach - the program *learns* a function using a sufficient amount of labelled data (in a process called *training*) in order to make predictions on unlabelled data.

MLP feature	**Neurotrigger feature**
Feed-forward	Deterministic runtime (maintain latency budget)
Fixed number of nodes	Want input for all SL and want 2 outputs
Fully-connected nodes & weights	All nodes computed in parallel
1 or mode hidden layer	1 hidden layer is sufficient

Table 4.1.: MLP and Neurotrigger features

The selected architecture for the Neurotrigger was the Multi Layer Perceptron (MLP) architecture. A neural network with this architecture is frequently called a *Perceptron.* MLPs are feed-forward networks - information is always fed to the subsequent layer and not backwards (like recurrent neural networks); they have a fixed number of input and output nodes; they have at least one hidden node; and they are fully-connected - every layer is fully-connected to the next layer and only the next layer, where a fully-connected layer means that every node in a layer is connected to every node in the subsequent layer. Table 4.1 summarises the features of an MLP that are beneficial for a Neurotrigger architecture and Figure 4.1 provides a schematic of the MLP structure.

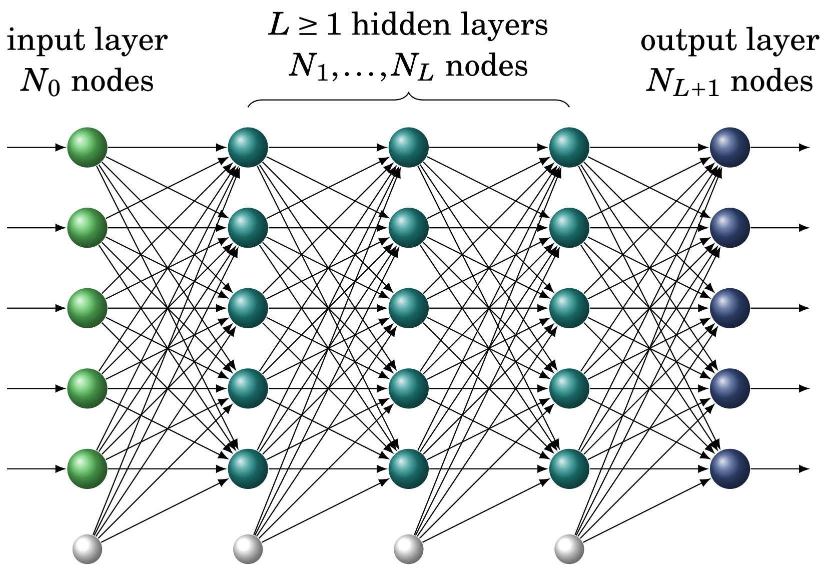


Figure 4.1.: Multi Layer Perceptron (MLP) architecture

4.1.1. Training Algorithm

The algorithm for such an MLP typically involves a series of forward and backward propagation steps. A combination of a forward propagation and backward propagation pass is called an *epoch*. Let's say we have an array of input vectors X:

$$X = \begin{bmatrix} \ldots & \ldots & \ldots & \ldots \\ x_1^{(1)} & x_1^{(2)} & \ldots & x_1^{(m)} \\ \ldots & \ldots & \ldots & \ldots \end{bmatrix}$$

where X is an (n_x, m) matrix, where m is the number of training samples and n_x is the number of input nodes or input features for each training sample. An additional bias $x_0^{(i)}$ is included for every i^{th} training sample, which is a vector of offsets for each layer in the neural network. The so-called bias nodes are determined by the network in the training algorithm. Bias nodes allow the output of an activation function to be shifted. Let's first examine the case of one training example, $X[:, 1] = x$, with only 2 layers (an input and output layer).

We want our network to predict the values Z, a $(n_z, 1)$ column vector, where n_z is the number of outputs nodes. The first step is to make what is called a *forward propagation* pass, where we make an initial prediction on Z. Let's call our prediction $\hat{Z}$, where $\hat{Z}$ is again a $(n_z, 1)$ column vector. For fully-connected ANNs, we then generate a weight matrix $W_{n_z,nx}$ where $i = n_z$ and $j = n_x$, with bias weights W_0^j. The weights are generally initialized as some random values. We compute the dot product of this weight matrix and the input vector, $W \cdot X$, a $(n_z, 1)$ column vector:

$$W_{n_z,nx}.X_{n_x,1} = \hat{Z}_{n_z,1} \tag{4.1}$$

This dot product is passed through a non-linear *activation function*, which typically scales the value between specific bounds. For example, we could use the sigmoid function σ, where $\sigma(x) = \frac{1}{1+e^{-x}}$, which will scale the output between [0,1]. Typically our the input values are not fed through an activation function, they are linearly mapped, typically between -1 and +1. target values are also scaled accordingly.** Our output vector $\hat{Z}$ is then compared to its true value Z. This is repeated for the m training examples in X. Next we want to do the *back propagation* step, where the weights are updated.

The principle behind back propagation is to compute the error on the prediction and update the weights appropriately. The error function, also known as the cost function, which depends on the weights and the bias b, may be written as a sum over all m training examples:

$$E(w,b) = \sum_{i=0}^{m} (Z_i - \hat{Z}_i)^2 \tag{4.2}$$

The weights are then updated according to the gradient of this function:

$$W_{ij} := W_{ij} - \alpha . \frac{\partial E(w,b)}{\partial Wij} \tag{4.3}$$

where α is the learning rate. The learning rate should be carefully selected - a very small learning rate will lead to a very slow convergence, whilst a larger one may lead to oscillations or skipping over some minima. The error should be minimised with each epoch.

To avoid something called overfitting, it is necessary to have a second, independent set, known as the validation set. Overfitting occurs when our network 'overlearns' a function so that it fits the training set very well, but may not be able to generalise to other data. Therefore, at every epoch, a validation set is passed through the epoch's weights and the cost function is calculated. Whenever the error on the validation set increases, we say that overfitting is occurring.

This algorithm can be extended to any n-layer MLP architecture, where the outputs are fed into subsequent hidden layers, before being output. An n-layer MLP therefore has $n-2$ hidden layers, $n-1$ weight matrices (and bias nodes), where the weight matrices are of size (L, L-1), where L is the current layer and L-1 is the previous layer.

4.1.2. z-Vertex NN Architecture

The Multilayer Perceptron (MLP) architecture used in the z-Vertex NN Trigger (Neurotrigger) is outlined as follows. The NN has 27 inputs; 3 inputs for each of the 9 SLs. These inputs are α, the crossing angle between the track and the priority wire, drift time t of the electrons from the track to the wire, and φ_{rel}, the distance between the track and the wire (see Section 3.1.4). The input layer is succeeded by one hidden layer with 81 nodes, three times the number of values of the input layer, which is the limit of the hardware. An academic study which examines the impact on the resolution with more nodes in the hidden layer is outlined in Section 5.5. The outer layer has two output nodes, which predict the z-vertex value of the track (the point along the z-axis the track originates from) and the polar angle θ. Note that an ANN architecture requires more than one hidden layer to be considered *deep learning*.

The Neurotrigger MLP uses the hyperbolic tan activation function $y(v_i) = tanh(v_i)$ for the hidden and output layers. This is convenient since the hyperbolic tan is differentiable and easy to calculate:

$$\frac{d}{dx}tanh(\frac{x}{2}) = \frac{1}{2}.(1 - tanh^2(\frac{x}{2})) \tag{4.4}$$

For the weight updates, the Resilient backPROPagation algorithm (RPROP) is used, which replaces a global learning rate with an adaptive learning rate, changing the magnitude of the step size whenever the sign of the gradient of the previous epoch does not match the sign of the gradient of the current epoch. This allows for slower learning rates over minima, and avoids the issues surrounding large learning rates, whilst avoiding slow convergence rates with smaller learning rates.

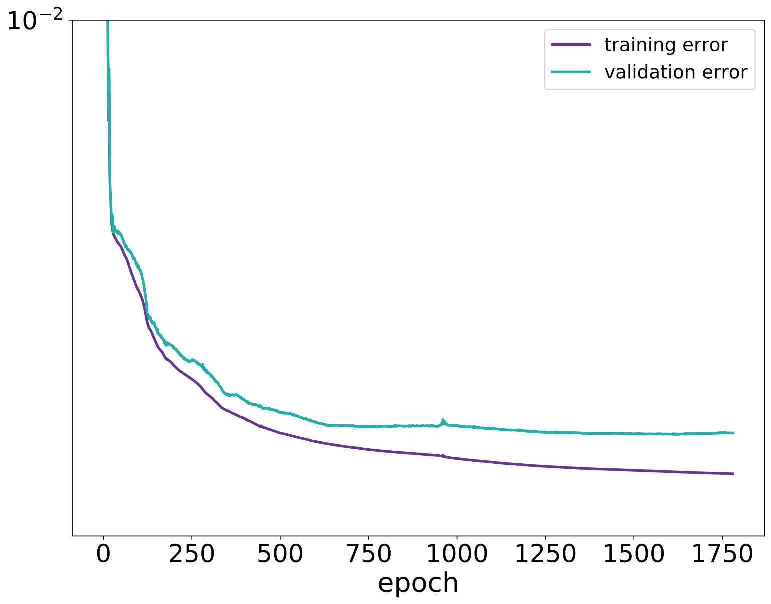


Figure 4.2.: Training and validation errors

In addition to the training and validation sets, the Neurotrigger utilises two additional datasets: an ID range preparation dataset and a test dataset. The ID range preparation dataset makes a histogram of the relative ID of a hit to the track, and makes a cut on this histogram to eliminate hits very far away, which are suspected to be background hits. The test dataset tests the network weights to produce an RMS value - whenever we train our network several times, the network weights with the best RMS value is selected as output from the Neurotrigger module (see Appendix A for details). Figure 4.2 shows example training and validation error curves. Whenever the validation error increases, the training is stopped. These error curves should be better understood, however, as there is quite some variation, as shown in Appendix B.

Software simulation studies of the Neurotrigger are outlined in Chapter 5.

4.1.3. FANN Library

The Fast Artificial Neural Network (FANN) Library, an open-source library, is used for implementation of the Neurotrigger training. FANN is an Open-source library that provides a C++ wrapper to implement a feed-forward neural network with predetermined number of layers and nodes.

5. Results

The Neurotrigger, outlined in Section 3.1.4, is trained in a supervised way with Monte Carlo (MC) generated particles which undergo a GEANT4 simulation [35]. The full first level (L1) track trigger may be simulated in the software (SW) up to and including the Neurotrigger. The output of the Neurotrigger simulation, i.e. the predicted z-vertex (found z) and polar angle θ values can then be compared to their 'true' MC z-vertex values (true z) to obtain resolutions.

This chapter details the results of several SW simulation studies of the Neurotrigger by investigating different preprocessing and training parameter effects on resolutions the neural networks can achieve. These studies focus on z-vertex resolutions, since the goal of the Neurotrigger is to suppress background outside a suitable 'z-cut' on the z-axis.

It should be noted that the Neurotrigger results presented here are in fact the combined resolutions of 5 so-called 'expert' neural networks. Each specialized expert is trained and tested according to the hit patterns of the input 2D tracks - one expert is trained for full stereo layer hits (4 TS hits to a 2D track) and the other 4 are trained for 3 hits per 2D track, one expert for each missing stereo layer hit. For further details on sectorization and particle generation parameters, see respectively Appendix A and Appendix B.

5.1. Standard training

The training presented in this section will be a so-called 'Standard training' - the resolutions of which will be used as 'benchmark resolutions' to compare the following studies to. The parameters for this training, which achieve good resolutions, were determined following the results of some of the below studies. Nevertheless, it is helpful to present this training as a standard before proceeding. Further plots and training parameters for all the triggered networks can be found in Appendix B and details of the training and simulating of the Neurotrigger can be found in Appendix A.

Figure 5.1 shows the distribution of z-vertex Neurotrigger predictions (we will call these found z) which overlay the MC z-vertex values (true z). The distributions are fairly uniform however there is some pile up in found z-vertex values near the $z = \pm\ 50\,\mathrm{cm}$ limits, where we set the boundaries for the training. This can be seen as the non-linear behaviour shown in Figure 5.2 towards the $z = \pm\ 50\,\mathrm{cm}$ boundaries again. This is an artefact of the training algorithm which scales the output values (for this network) within the $z = \pm\ 50\,\mathrm{cm}$ range using the hyperbolic tangent function as activation function, which has a similar shape (see Section 4.1.2). It should also be noted that more true z-vertex values and therefore more found z-vertex values are found in the positive z direction, due to the asymmetric geometry of the detector.

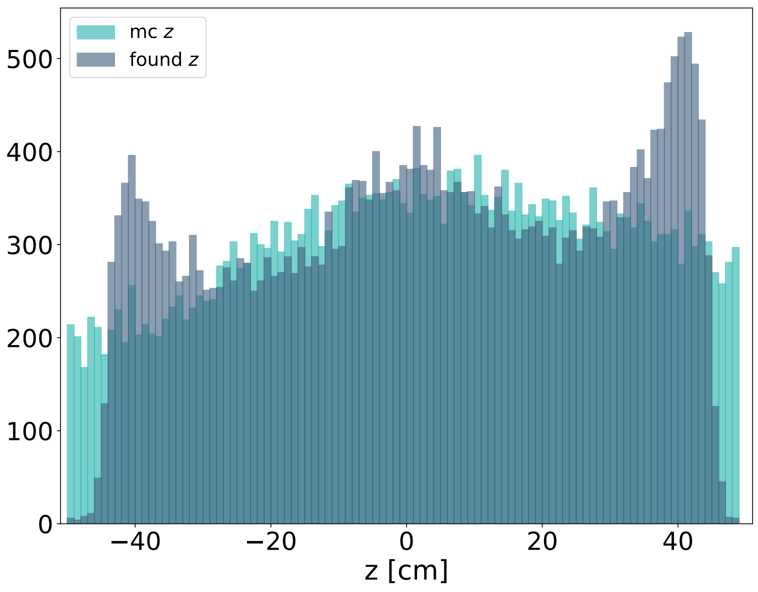


Figure 5.1.: Distribution of z-vertex values found by the neural network (found z) and the MC z-vertex values (true z), particles generated along ±50 cm

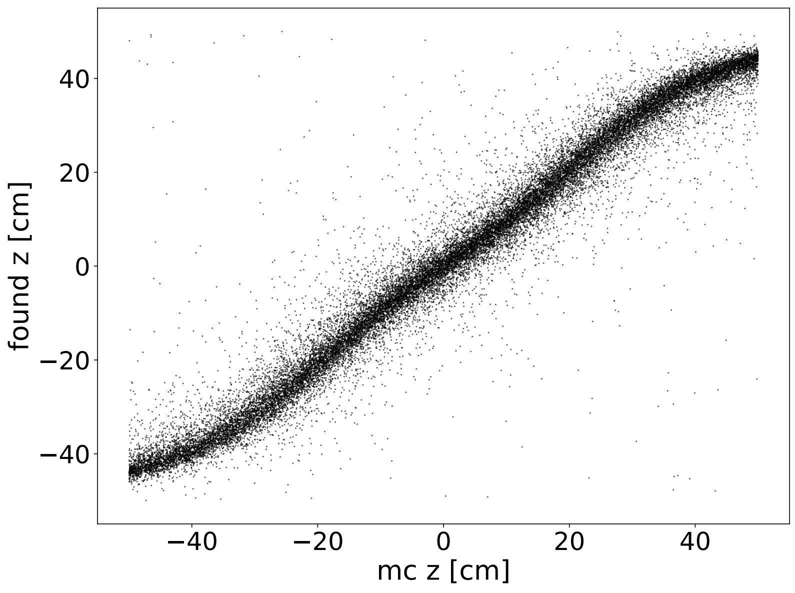
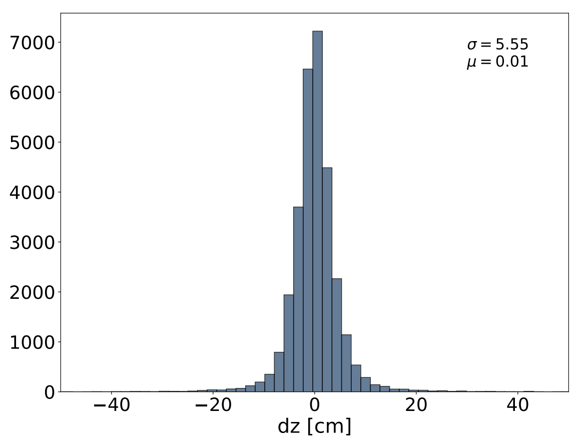
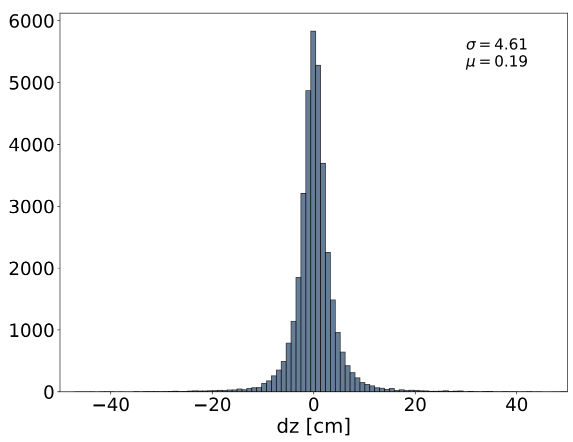


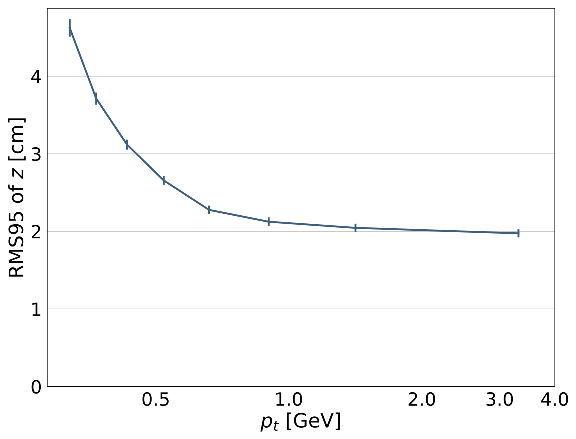
Figure 5.2.: Distribution of found z and true z-vertex values along the ±50 cm range



(a)



(b)



(c)

Figure 5.3.: The dz resolution is the difference between the z-vertex found by the ANN and the true MC Particle z-vertex value: (a) network tested with MC Particles generated uniformly along $z = \pm\ 50\,\text{cm}$; (b) network tested with MC particles generated at the IP (z=0); (c) dz resolution at the IP plotted as a function of p_t

Figure 5.3 shows the resolutions obtained by the Standard network, where we define the resolution dz as the difference between the found z and the true z values (dz= true$z$-found$z$). Figure 5.3(a) plots the dz resolution when the network is tested with particles generated along $z = \pm\ 50\,\text{cm}$ and Figure 5.3(b) plots the dz resolution when the network is tested with particles generated at the IP (in this case, dz is just a distribution of found z values, since true z in this case is 0). Figure 5.3(c) plots the RMS95 dz resolutions from (b) according to their 'true' particle transverse momentum (p_t) values. The RMS95 is the trimmed standard deviation which throws out 5% of the outliers from the calculation. The error bars are calculated by taking the variance of this RMS95 value - for details on the error calculations, see Appendix C. The resolution is worse in the low p_t region as expected, due to a multiple scattering of particles. The p_t-dependent resolution will be plotted with particles generated at the IP from here on.

Figure 5.4 plots the dz resolution depending on p_t and and true z value. One can see again the dependence of the resolution on p_t. Additionally, there is a dependence on the resolution on the origin of the track along the z-axis; resolution deteriorates with increasing displacement from the IP. This is due to the asymmetry of the detector. Additionally, one can see that in the positive z-direction the network is more likely to predict a negative z-vertex value (since dz = mc z - found z); this is again due to edge effects of the network seen in Figure 5.1. For further plots see Appendix B.0.1.

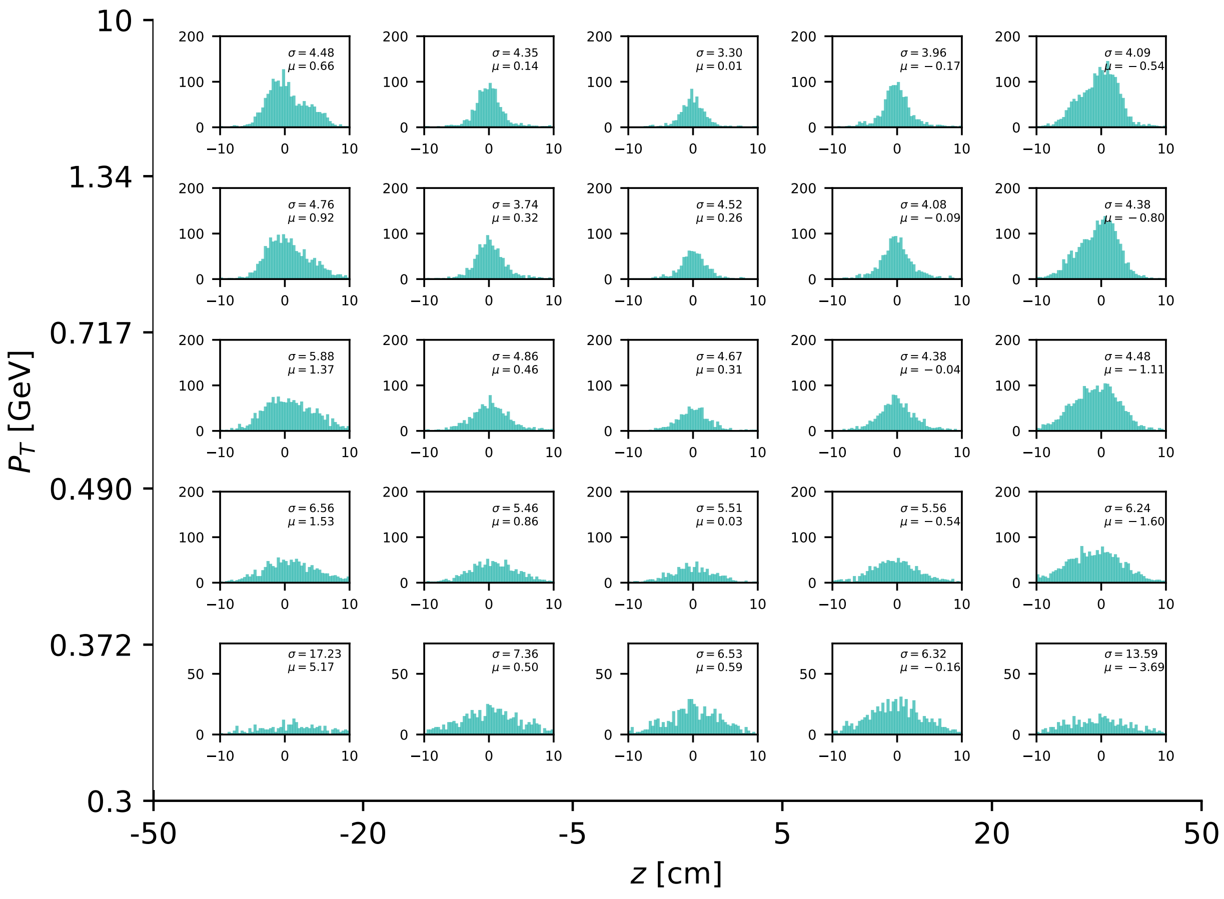


Figure 5.4.: The dz resolution as a function of the particle transverse momentum p_t (y-axis) and as a function of the true z-vertex value (x-axis)

5.2. Background studies

It is interesting to see how random background hits mixed with generated events (see Section 2.3.7) influences the resolutions of our networks. Figure 5.5 compares the p_t-dependent resolutions in two cases: the Standard network, which is trained with Phase 2 background, and a new network which is trained without background. Both networks are tested with here with Phase 2 background. Interestingly, the network that is trained without background performs slightly better in the low p_t region and approximately the same in the high p_t region, which might indicate that the network trained with background has not learnt to identify tracks with additional background hits.

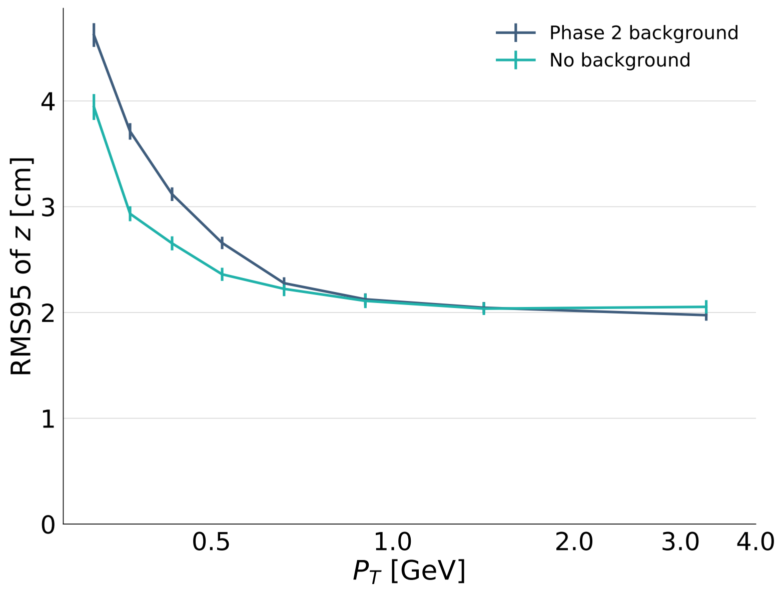


Figure 5.5.: p_t-dependent resolution of two networks - the Standard network (trained on Phase 2 background) and a new network trained without background; both are tested with MC Particles with Phase 2 simulated background mixed

This study was repeated for Phase 3 simulated background; a new network was trained with Phase 3 background, and all three networks were now tested with Phase 3 background hits mixed, shown in Figure 5.6. The resolution deteriorates significantly for Phase 3 background; evidently the presence of more background hits influences one or more of the preprocessing steps. For example, the TS information will deteriorate, meaning left/right information can be wrong which can affect the drift times, or wrong TS could be assigned to a 2D track.

One can see quite clearly from Figure 5.6 that it is important, at least in the presence of more background, that the network should be trained with background in order to identify background-like events. Moreover, with expected Phase 3 background levels, the network should be trained with the appropriate background conditions. However, at the beginning of Phase 3, while beam currents are still low, the Standard network can be used, but with the increase in luminosity a Phase 3 network should eventually replace this. For further plots see Appendices B.0.2–B.0.3.

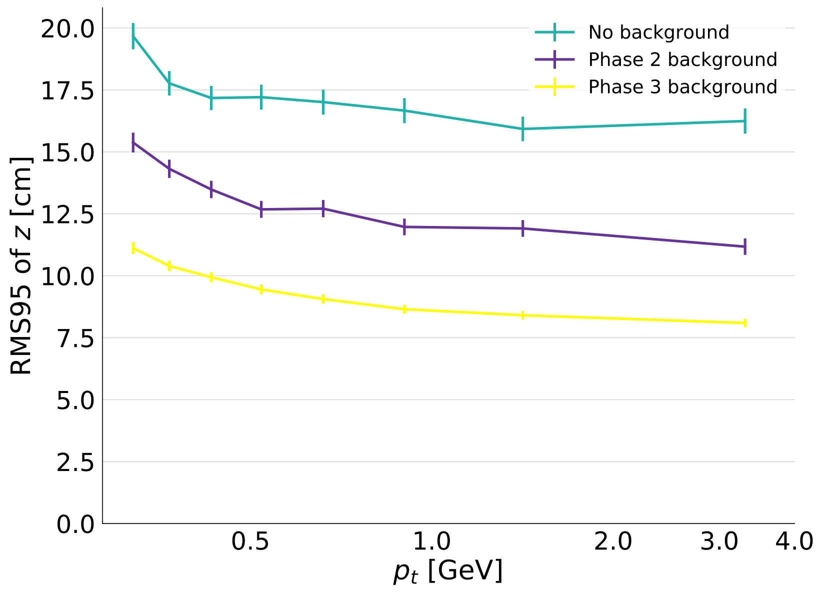


Figure 5.6.: Three networks are trained - one is trained on events without background mixed, one is trained with Phase 2 simulated background mixed and one on Phase 3 background; all are tested here with MC Particles generated with Phase 3 simulated background mixed

5.3. Event Time Option

SW studies have found a non-negligible influence of the selected event time finding algorithm, used to compute the drift time for the CDC wires, on the z-resolution. In particular, when our ANN is not trained with a reliable event time algorithm derived from the CDC system, resolutions deteriorate significantly. Therefore, it is crucial that the ANN used in the experiment is trained with an optimal event time finder algorithm that is also available at the hardware level to preprocess the input in the hardware trigger.

There are currently two options for determining the event time that can be input to the Neurotrigger - the Event Time Finder (ETF) selects the fastest wire hit if a certain number of TS are active in a given time window, defined by the threshold value (see Section 3.1.3 for details). Alternatively, the fastest priority time may be used, which provides the timing of the fastest (first) hit priority wire in a 2D track. In principle, one would naïvely expect the fastest priority time to provide the best resolution, since

Threshold 0	100%
Threshold 1	99%
Threshold 2	77%
Threshold 3	35%

Table 5.1.: Efficiency of ETF to find an event time for different threshold values when trained with Phase 2 background - explain what these efficiencies mean earlier in text, when introducing thresholds

the priority wires are not necessarily the first wires that are hit in a TS. The Standard network uses this fastest priority time. The following subsections study the performance of the ETF compared to the fastest priority time for Phase 2 and Phase 3 backgrounds.

5.3.1. Phase 2 background

The p_t-dependent resolutions for the fastest priority time (Standard network) and the ETF at 4 different threshold levels which are trained and tested with Phase 2 background are compared in Figure 5.7. One sees that as the threshold increases - i.e. the number of active track segments in the timing window required to output an event time - the resolution deteriorates significantly.

Since the ETF does not always output an event time, a fallback option must be selected in the trainer. Here the fallback option is to input no drift times (i.e. set all drift times to 0) for an event. The resolution of a 'no drift time' network (inputting all drift times for all events in a training to 0) is shown as a comparison. It then makes sense to consider the efficiency of the ETF to output an event time, shown in Table 5.1. The ETF efficiency for each threshold corresponds to the deterioration. Evidently, at high thresholds, the ETF does not find enough active TS in a timing window with Phase 2 background.

Since Phase 2 background hits are much lower than in Phase 3, and since these simulations study only single tracks, these studies may not be realistic. Phase 3 background, studied in the next subsection, would then be expected to provide better resolutions. At the early stages of Phase 3, before full luminosity is reached, it is expected that background levels will be comparable to Phase 2 simulated background. Therefore, it has been decided to use the fastest priority time as event time input until luminosity increases. For further plots see Appendices B.0.4–B.0.7.

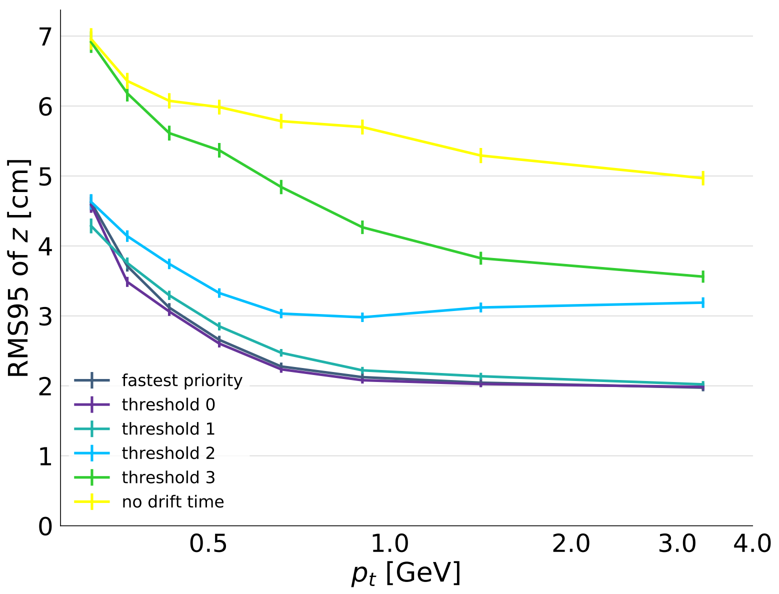


Figure 5.7.: p_t-dependent resolutions of the ETF at different threshold levels and FP time

5.3.2. Phase 3 background

The above study was repeated for Phase 3 background (trained and tested) and indeed a different trend can be seen; the resolution improves as the threshold values increase, shown in Figure 5.7. This can be attributed to more TS hits present in a given time window, improving the efficiency of the ETF. In fact, the ETF provides much better resolution than the fastest priority time with a threshold value of 4.

Therefore, when design luminosity is reached, it is recommended to use the ETF algorithm. The deterioration in resolution with a threshold value of 5 (6 active TS hits in a timing window required) can be attributed to 'too many' TS hits required in a given time window; the efficiency decreases. For further plots see Appendices B.0.8–B.0.13.

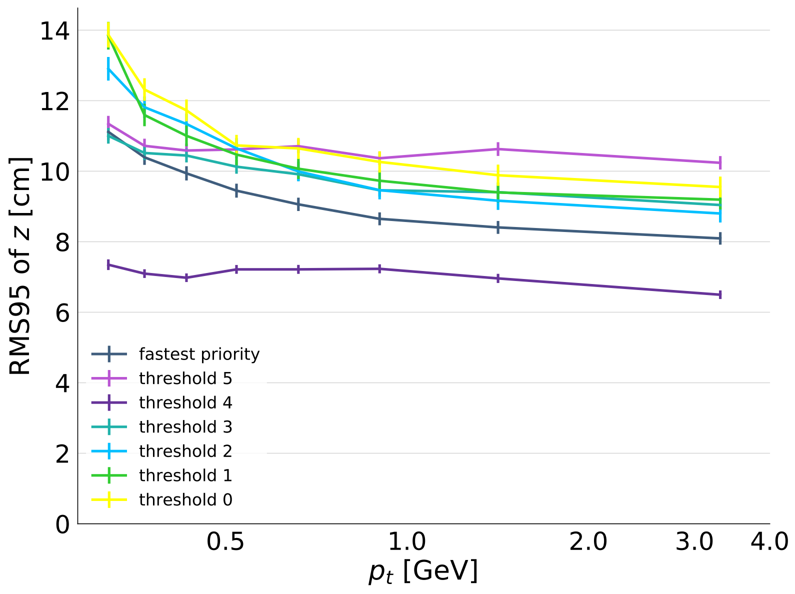


Figure 5.8.: p_t-dependent resolutions of the ETF at different threshold levels and FP time with Phase 3 background

5.4. Left/Right Information

As described in Section 3.1.4, our networks take 27 input nodes, nine of which are drift times, one for each SL. In addition, a 2 bit left/right information is provided, informing on which side of the wire the track passed. The drift times are then input to our NN with a ± sign, which tells the NN on which side of the wire the track passed (left or right). However, whenever this left/right information is undetermined for the 2D track, the drift time inputs for the event are set to 0.

Figure 5.9 shows the inefficiency of the left/right determination; approximately 1/3 of TS hits have no left/right information, which means no drift time is used at all. This study aimed to investigate the impact on the resolution if we ignore the left/right information and instead input the modulus of the drift time. The motivation behind this study was that the network might somehow *learn* to determine the sign of the drift time. Again no drift time (inputting all event drift times to 0) is used as a comparison.

Phase 2 background was used again for training and testing and will be used from here on for the following studies. The results can be seen in Figure 5.10. It is clear from the resolution that the network relies on the left/right information. In fact, taking the modulus of the drift time actually achieves worse resolution than neglecting the drift time totally. To avoid the problem of undecided left/right information, the TS LUTs could be improved, or a new neural architecture could be considered. For further plots see Appendices B.0.14–B.0.15.

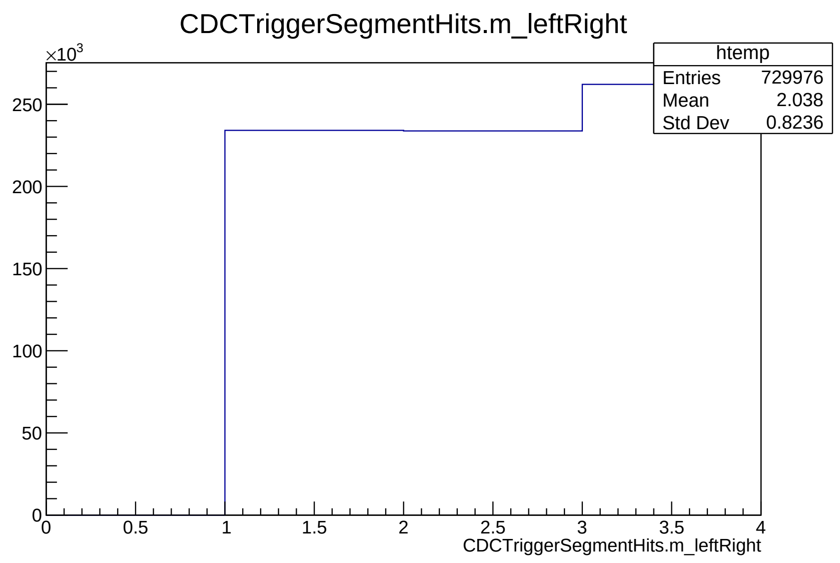


Figure 5.9.: Left/right determination for TS hits - from left to right: (1) Number of TS hits found to pass on left-side of priority wire of TS; (2) Number of TS hits found to pass on right-side of priority wire of TS; (3) Number of TS hits with undetermined left/right information

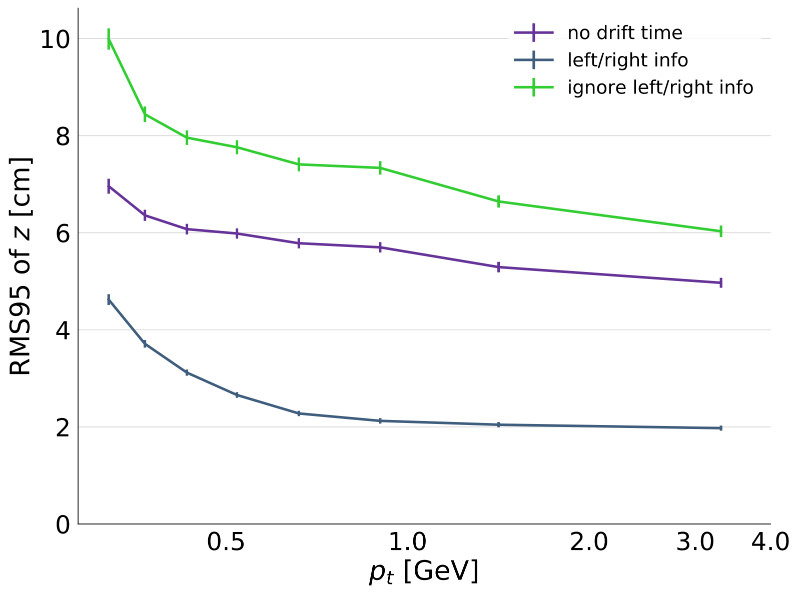


Figure 5.10.: p_t-resolution deteriorates if we ignore left/right drift time information

5.5. Increasing hidden nodes

As an academic study (or to possibly motivate more powerful hardware), the number of nodes in the hidden layer of the network were increased to see if this had any effect of resolution. Figure 5.11 shows that increasing the number of hidden nodes has negligible effect on the resolution. However, more hidden layers could be investigated, and may also improve resolution in the above study. For further plots see Appendix B.0.16.

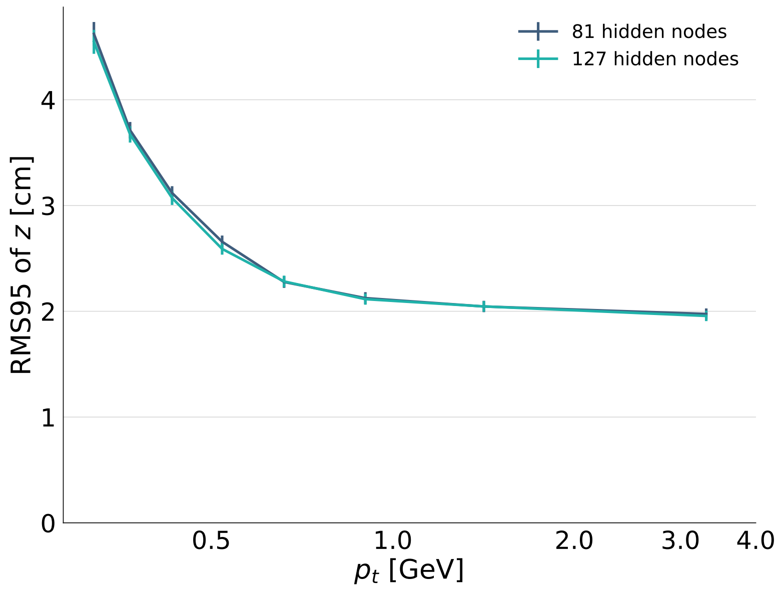


Figure 5.11.: p_t-dependent resolution with 127 hidden nodes compared to 81 (Standard network); more hidden nodes has a negligible effect on resolution

5.6. Enlarged training region

So far, all the trainings presented have been trained with MC particles generated in the ±50 cm region with the network outputs scaled accordingly. As seen in Figure 5.1, there is a pile up of events at the ±50 cm regions, which are the boundaries of the training. This is an artefact of training our network in a bounded region with the tanh activation function.

This was deemed a sufficient training region with a small enough z-cut ($\mathcal{O}(10\,\text{cm})$, for example, 6 cm. However, as of July 2019, a preliminary z-cut of 40 cm has been suggested by the collaboration. Such a z-cut would be very close to the training boundaries, where events are not accurately predicted by the network. This can be seen more evidently in Figure 5.2, where we lose linearity outside the ±30 cm region. The non-linear behaviour at the edges tells us that here, the network is not predicting the z values well. As such, it was decided to train in a larger z region of ± 100 cm, in order to avoid this non-linearity so close to a z-cut of 40 cm.

Figure 5.12 show the distribution of found z values in the ± 100 cm region. Again, there are some pile ups of z-vertex values predicted by the NN at the boundaries of the training regions, but these are now extended further out in z. As such, we now have linear correlation in the ± 50 cm region as seen in Figure 5.13. Figure 5.14 shows the effect on the resolution of the training in the extended region. There is some deterioration in this extended region, but whilst the GDL maintains a z-cut of 40 cm, training in the extended region will be adopted. For further plots see Appendix B.0.17.

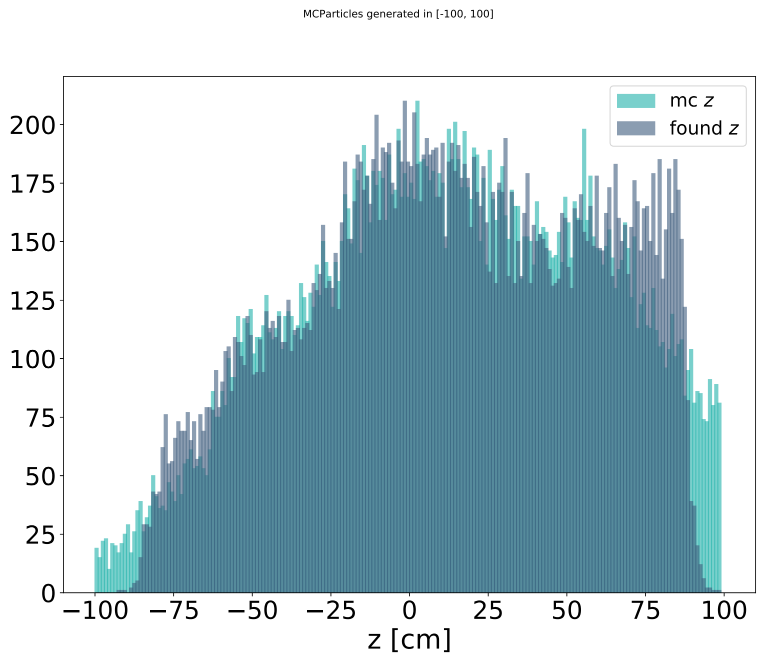


Figure 5.12.: Distribution of z-vertex values along the ± 100 cm range

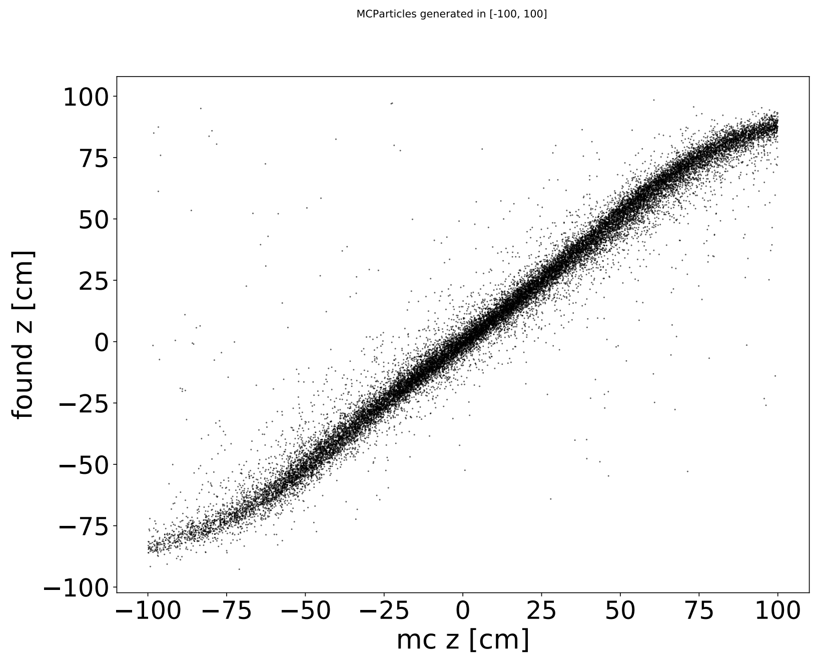


Figure 5.13.: Scatter plot of found z and true z-vertex values along the $\pm 100\,\mathrm{cm}$ range

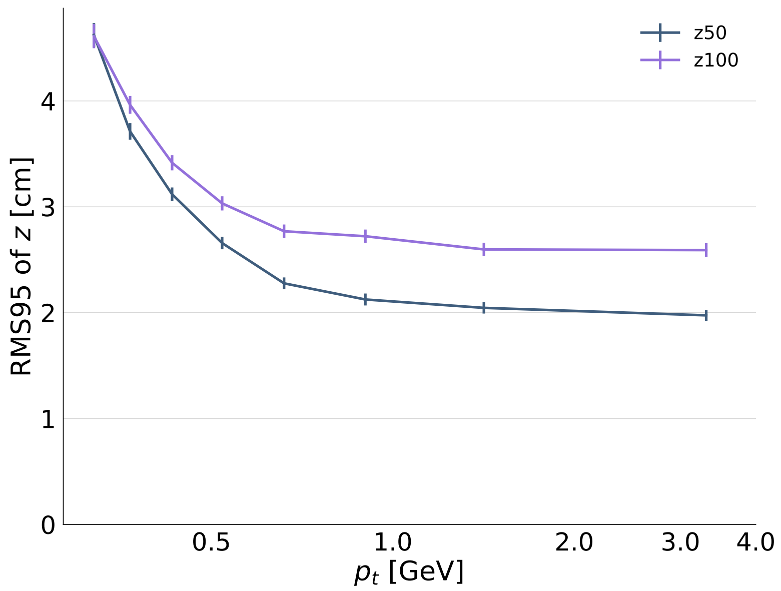


Figure 5.14.: Distribution of z-vertices along the $\pm 100\,\mathrm{cm}$ range

5.7. Reconstructed Tracks

In order to train the networks with realistic background, it is foreseen to train on Reconstructed Tracks (RecoTracks) from real data rather than MC Particles. The High Level tracking produces an object called RecoTracks which contain the results of the fit, as well as the hits used in the fitting. The procedure of training for RecoTracks is somewhat more complicated than four-vectors from MCParticles, however. The RecoTracks are not extrapolated to the z-axis and this must be somehow undertaken by the trainer. This then retrieves the position and momentum of the track at the Point of Closest Approach (POCA) which is used for the extrapolation.

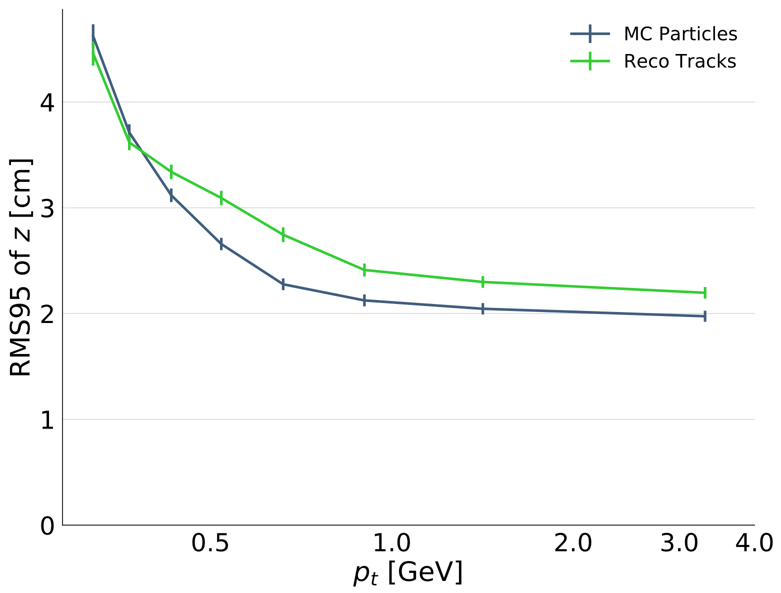


Figure 5.15.: p_t-dependent resolution when using Reco Tracks as training targets, compared to using MC Particles as training targets (Standard network)

Figure 5.15 shows the results when training (and testing) with Reco Tracks from the fully reconstructed MC four-vectors, as opposed to training and testing with MC particles. The resolution is slightly worse in the Reco Track case which is not expected: MC particles scatter and secondary particles may be triggered on, which would have a different track in the detector than what is simulated at the vertex. With Reco Tracks we have hits from a 4-vector which propagate through the detector, so we expect the resolution here to be better. Ultimately, training with Reco Tracks should replace MC particle trainings, since in the end we want to train on Reco Tracks from real data, with the expectation that this will improve resolution further. Further studies should be undertaken to investigate training on Reco Tracks. For further plots see Appendix B.0.18.

Conclusion

The Belle II Experiment at the e^+e^- SuperKEKB collider located at KEK, Japan aims to perform precise measurements of CP-violation in the B-meson system, in addition to an extensive extensive NP program. SuperKEKB has been upgraded from its predecessor Belle to reach luminosities of up to $8 \times 10^{35}\,\mathrm{cm}^{-2}\,\mathrm{s}^{-1}$ and the Belle II detector has been upgraded accordingly. The physical motivations and an outline of the SuperKEKB upgrade and Belle II detector were outlined in Chapters 1 and 2 respectively.

This luminosity upgrade is expected to be accompanied by much higher rates of background compared to Belle. Belle II's trigger system aims to reduce a large proportion of this background at the first trigger level (L1). The L1 track trigger is implemented in hardware in a pipelined fashion. The Neurotrigger component of the L1 track trigger aims to reconstruct by neural methods the origin of a track along the beamline (z-axis) in order to reject tracks coming from outside the IP. The Neurotrigger is based on an MLP architecture and calculates inputs from the preprocessed CDC hits. The Neurotrigger is trained in the SW to output a z-vertex and polar angle θ values. The weights from the trained networks are uploaded on FPGAs to trigger on events online to reject events outside a dedicated z-cut. A summary of Belle II's trigger system was outlined in Chapter 3 and an overview of the Neurotrigger architecture and training algorithm was given in Chapter 4.

The full L1 track trigger may be simulated in the SW up to and including the Neurotrigger. This thesis presented the results of SW simulation studies in Chapter 5. MC Particles were primarily used as targets in the supervised training process (a training with Reco Tracks as targets is outlined in Section 5.7) and for testing the Neurotrigger. Tests on real data can be found in Appendix B. The studies showed that with Phase 2 background, the Neurotrigger could achieve as low at 2 cm in the high p_t region (Section 5.1). A preliminary z-cut of 40 cm has been suggested for early Phase 3, motivating training in the enlarged $z = \pm 100$ cm region (Section 5.6).

The studies however show that although the resolution is fine for Phase 2 simulated background, with increased luminosity and therefore increased background (Phase 3 simulated background), resolutions deteriorate significantly (Section 5.2). It is worth noting that the influence of the selected event time (used to determine the drift time input to the Neurotrigger) has a considerable effect on the resolution, shown in Section 5.3. Further studies on increased background levels should be taken and more tests should be performed with real data to better understand the performance of the Neurotrigger under these high-background conditions. The ETF algorithm could also be better understood and perhaps improved to enhance resolution. If the hardware is upgraded it could accommodate further sectorization or more hidden layers, which could already be investigated at the level of SW.

As of March 2019, Belle II entered Phase 3 of the experiment and data taking is under way with full detector geometry [28]. First runs show that results agree with expectations, and debugging and optimization of the hardware performance and their correspondence with the SW simulations are being investigated. Neural networks trained in the $z = \pm 100\,\mathrm{cm}$ range are expected to be uploaded to the FPGAs in the near future, allowing for implementation of the planned z-cut of $40\,\mathrm{cm}$. Training with Reco Tracks and with real data are being investigated as these are expected to provide better resolutions still.

A. Training & Simulating the Neurotrigger

A.1. Dependencies

The Belle II software framework, `basf2`, provides a virtual environment with all necessary dependencies for a specific release or branch to work on. Where possible, all networks were trained, tested and analyzed on the below commit number.

- **Commit number:** `f7f69586c40aa1667e6024f4d2d2c298cab070ab`
- **Python version:** 3.6.6
- **ROOT version:** 6.14/06
- **Belle2 Externals version:** v01-07-01

A.2. Definitions

- **MCParticles** - GEANT4 simulated particles - the z-vertex and θ are taken from the MC four-vector when training on MCParticles
- **2DTrack** - 2D Track found by the 2D Finder at L1 trigger (every 2D track appears to produce a NeuroTrack)
- **NeuroTracks** - 3D tracks reconstructed at level of L1 trigger
- **RecoTracks** - 3D tracks reconstructed at level of HLT and offline - the extrapolated offline z-vertex and θ are taken from the track itself when training on RecoTracks
- **TS hit** - Track segment hits are Priority hits from a TS (related to a track). Each TS has an ID and the drift time belonging to the priority wire
- **Event time** time of passage through detector for a given event, determined from TS drift times
- **Sector, MLP** and **Expert** - used synonymously for each neural network that comprises the Neurotrigger (by default, the Neurotrigger has 5 networks loaded)

A.3. Generating MC Particles

The `ParticleGun` module in `basf2` generates single tracks/events that can be used for training and testing the network. Various input parameters define the type and number of events we wish to simulate. Their trajectory is simulated through the detector using GEANT4. Events may also generate secondary particles on their simulated passage through the detector. It is also possible to 'reconstruct' the particle tracks using the offline algorithm by calling the function `add_reconstruction()`. This is called after the `add_simulation()` function. It is also possible to mix or overlay different backgrounds. The particles can be generated on the fly, but it is recommended to generate large samples for training and testing and outputting these to a ROOT file - this saves a lot of time during the training and testing steps and also keeps this as a constant during training/testing. Typically muons are produced in uniform inverse Pt, uniform costheta and uniform in z. The ranges of these values are typically altered in order to observe their effect on the training. Standard and variable parameters for generation can be found in Appendix B. Path modular order for generating particles:

```
import basf2
from simulation import add_simulation
from reconstruction import add_reconstruction

main.basf2.create_path()
main.add_module('Gearbox')
main.add_module('Geometry')
main.add_module('ParticleGun') #Or other physics generator
main.add_module('BeamBkgMixer') #If want to mix Bkg
add_simulation(main) #Includes CDC Digitizer, or use Full Sim
add_reconstruction(main) #Optional
main.add_module('RootOutput')
basf2.process(main)
```

A.4. Training the Neural Networks

`CDCTriggerNeuroTrainer`[1] is the neural network (NN) trainer module for the Neurotrigger. It takes track segment hits, 2D track estimates and an event time to prepare input data for the training of the NN. The NNs are then trained after the event preparation loop and saved. The data preparation is done in two steps:

1. The MLP uses hits from a limited range around the 2D track.
2. Input data is calculated from the hits, the 2D tracks and the ID ranges. Target data is collected from a MCParticle or RecoTrack related to the 2D track.

[1] basf2/software/trg/cdc/modules/neurotrigger/

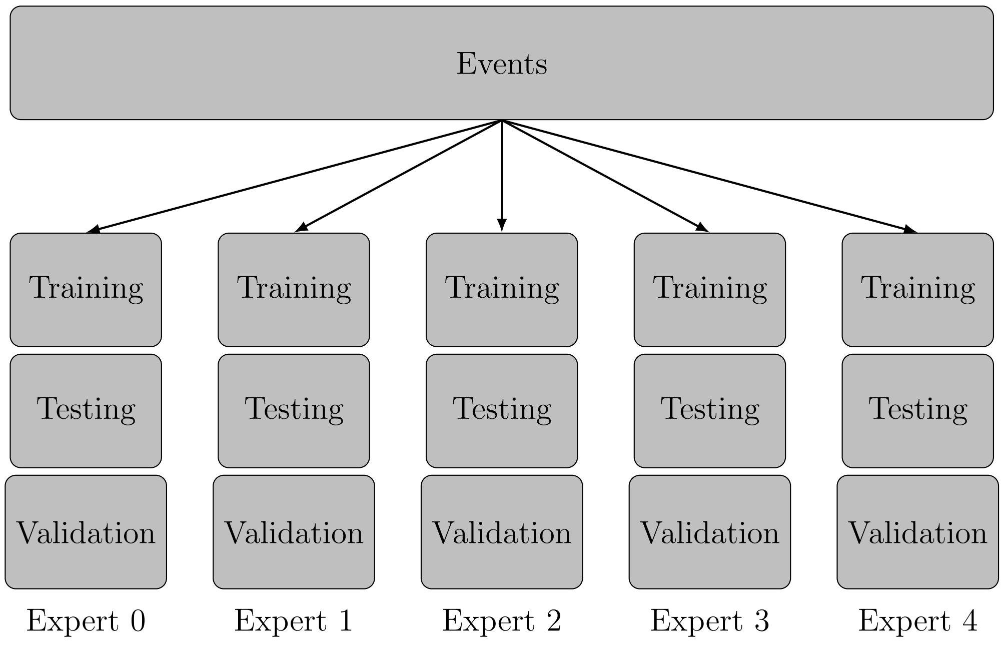


Figure A.1.: Training, testing and validation events are collected together in one vector for each expert in the `CDCTriggerNeuroTrainer` module, and are used according to their position in the vector

The `CDCTriggerNeuroTrainer` module takes as input one large dataset. In the event preparation step, a dataset for each expert is prepared (events may be reused for experts). Each dataset for each expert is divided into 4 subsets depending on their position in the list. The first `nTrainPrepare` events is used to compute relevant ID ranges, the next `nTrain` events are used for training the NN and updating the weights, the next `nValid` events are used as a validation set and the remaining `nTest` events are used to test the weights after training to calculate an RMS and select the best training run (if `repeatTrain` is greater than 1). See Figure A.1 for a schematic of the data organisaton in the trainer module. A fifth, independent dataset is generated to test the NNs to produce the plots in Chapter 5.

A.4.1. Simulate L1 Pipeline

To prepare input for `CDCTriggerNeuroTrainer` module, the sequence of hardware modules in the L1 pipeline pipeline must be simulated. The modular order of the path (when training on MCParticles) follows:

```
import basf2

main.basf2.create_path()
main.add_module('RootInput') #Input the generated particles
main.add_module('Gearbox')
main.add_module('Geometry')
main.add_module('CDCTriggerTSF')
main.add_module('CDCTrigger2DFinder')
main.add_module('CDCTriggerETF')
main.add_module('CDCTriggerMCMatcher)
main.add_module('CDCTriggerNeuroTrainer)
add_simulation(main) #Includes CDC Digitizer, or use Full Sim
add_reconstruction(main) #Optional
main.add_module('RootOutput')
basf2.process(main)
```

A.4.2. Definitions

Clarification of some important parameters:

- **nTrainPrepare** Number of samples for preparation of relevant ID ranges
- **nTrain** Number of samples for training an expert
- **nValid** Number of samples for validation
- **nTest** Number of samples to test expert weights
- **hitCollectionName** Name of the input StoreArray of TS hits
- **inputCollectionName** Name of the StoreArray holding the 2D input tracks, also called **m_tracks**
- **targetCollectionName** Name of the MCParticle/RecoTrack collection used as target values
- **trainSets** Set of training data for all sectors (one set for every sector)
- **outputScale** Output scales scaled from [-1,1] to specified scales (in z and θ)
- **phiRange, invptRange** and **thetaRange** - sectorize experts where `nPhi * nPt * nTheta * nPattern = nMLP`, for example 1 value pair for phi e.g. [0, 360] (`nPhi` = 1) gives one sector in phi
- **phiRangeTrain, invptRangeTrain** and **thetaRangeTrain** - the range from which training events are taking

- **maxHitsPerSL** Maximum number of hits in a single SL (default=1)
- **SLpatternMask** Checks/ignores input for sector selection for example just check stereos for sector selection
- **SLpattern** (or sector pattern) chooses which input is used and which input is not used, axials are used if present
- **rescaleTarget** True: set target values greater than outputScale to 1, else skip them
- **wMax** weights are limited to [-`wMax`, `wMax`] after each training epoch for convenience of FPGA implementation, default=63
- **checkInterval** Training stopped if validation error higher than the validation error `checkInterval` epochs ago, or the gain is less than the fluctuations, default=500
- **repeatTrain** If set to > 1, the training is repeated `repeatTrain` times with different start weights. The final weights are triggered on with `nTest` events and the run with the best resolution on the test samples is kept

A.4.3. Pseudocode

What happens in pseudocode for event preparation:

```
for itrack in 2Dtracks:
    get target values
    update 2D tracks
    scale targets
    if event number < nTrainPrepare:
        get relative ids for all hits
    else:
        count relative ids to find relevant id range
        get event time
        check hitPattern == sectorPattern
        save dataset
```

A.4.4. Code snippets

After initialization, every 2D track is related to a RecoTrack or MCParticle hypothesis:

```
for (int itrack = 0; itrack < m_tracks.getEntries(); ++itrack) {
  // get related MCParticle/RecoTrack for target
  // and retrieve track parameters
  float phi0Target = 0;
  float invptTarget = 0;
  float thetaTarget = 0;
  float zTarget = 0;
  if (m_trainOnRecoTracks) {
    RecoTrack* recoTrack =
      m_tracks[itrack]->getRelatedTo<RecoTrack>(m_targetCollectionName);
    if (!recoTrack) {
      B2DEBUG(150, "Skipping CDCTriggerTrack without relation to
      RecoTrack.");
      continue;
```

The trainer takes the first RecoTrack hypothesis that does not give errors. If no valid representation is found, the 2D track is skipped (the trainer tries to get the state of the RecoTrack using the hit from closest to the IP - when an exception occurs, the next representation is tried). The 2D track variables are updated accordingly. Every 2D track is then matched to every MLP that fulfils phase space sector criteria (however there is an option to select sector based on MC information). If no sector is found for the track the event is skipped. First its `phi0`, `invpt` and `theta` values are fetched:

```
  // find all matching sectors
  float phi0 = m_tracks[itrack]->getPhi0();
  float invpt = m_tracks[itrack]->getKappa(1.5);
  float theta = atan2(1., m_tracks[itrack]->getCotTheta());
```

The MLPs are then selected according to their `phi0`, `invpt` and `theta`:

```
  vector<int> sectors = m_NeuroTrigger.selectMLPs(phi0, invpt, theta);
  if (sectors.size() == 0) continue;
```

The number of available sectors is given by `nMLP = nPhi * nPt * nTheta * nPattern`. In our case, we do not sectorize in `phi0`, `invpt` or `theta`, so the above step is redundant and therefore we have `nMLP = nPattern`, where `nPattern` is typically 5. At this stage, every 2D track is still assigned to every expert.

Targets are then rescaled if necessary and then saved. Relevant ID ranges are then found for each sector using the first `m_nTrainPrepare` events in each sector's trainSet by selecting hits from a limited range around the 2D tracks: histograms of the *relative* IDs of all hits related to the RecoTrack are produced before making a cut selection on *relevant* ids by discarding a `cutSum`% of hits. The aim of this step is to further eliminate background hits which are far from the track.

```
if (m_nTrainPrepare > 0 &&
    m_trainSets[isector].getTrackCounter() < m_nTrainPrepare) {
  // get relative ids for all hits related to  MCParticle/RecoTrack
  // and count them to find relevant id range
  // using only related hits suppresses background
  // EXCEPT for curling tracks
  if (m_trainOnRecoTracks) {
    RecoTrack* recoTrack =
      m_tracks[itrack]->getRelatedTo<RecoTrack>(m_targetCollectionName);
    for (const CDCTriggerSegmentHit& hit :
         recoTrack->getRelationsTo<CDCTriggerSegmentHit>
         (m_hitCollectionName))
      // get relative id
      double relId = m_NeuroTrigger.getRelId(hit);
      m_trainSets[isector].addHit(hit.getISuperLayer(), round(relId));
```

Once the loop has finished, the relevant IDs for the sector can then be updated:

```
  m_trainSets[isector].countTrack();
  // if required hit number is reached, get relevant ids
  if (m_trainSets[isector].getTrackCounter() >= m_nTrainPrepare) {
    updateRelevantID(isector);
```

The number of samples is calculated by `nSamples = nTrain + nValid + nTest`. `nTrainMax` is typically `nWeights` multiplied by a factor, but can be explicitly specified. A check is made if this number has already been processed:

```
  else {
  // check whether we already have enough samples
  float nTrainMax = m_multiplyNTrain ? m_nTrainMax *
  m_NeuroTrigger[isector].nWeights(): m_nTrainMax;
  if (m_trainSets[isector].nSamples() > (nTrainMax + m_nValid +
  m_nTest)) {
    continue;
```

Now a check is made to check if the 2D hit pattern matches the sector pattern:

```
  if (sectorPattern > 0 && (sectorPattern & hitPattern) !=
  sectorPattern) {
        B2DEBUG(250, "hitPattern not matching " <<
        (sectorPattern & hitPattern));
        continue;
```

Here the logic says that hit patterns with hits in every SL are appended to every expert dataset (their input nodes are later set to 0) and hit patterns with missing SL hits are appended to the appropriate expert. The selected event time is then determined. The first `nTrain` events in the dataset for each sector are taken for the training of the experts, the next `nValid` events are taken for the validation, and the remainder is used for testing the experts:

```
unsigned nTrain = m_trainSets[isector].nSamples() - m_nValid - m_nTest;

struct fann_train_data* train_data =
  fann_create_train(nTrain, nNodes[0], nNodes[nLayers - 1]);
for (unsigned i = 0; i < nTrain; ++i)

struct fann_train_data* valid_data =
  fann_create_train(m_nValid, nNodes[0], nNodes[nLayers - 1]);
for (unsigned i = nTrain; i < nTrain + m_nValid; ++i)
```

Input data is prepared by calling `getInputVector`. The training is then performed with random initialized weights in [-0.1, 0.1] `repeatTrain` times. The best run is selected which outputs the best RMS when the expert is triggered on using the `nTest` samples. The network weights are saved to a ROOT file which can be triggered on in the SW, and can then be converted to a format suitable for FPGA upload for the HW trigger.

A.5. Simulating the SW Trigger

```
  import basf2

  main.basf2.create_path()
  main.add_module('RootInput') #Input the generated particles
  main.add_module('Gearbox')
  main.add_module('Geometry')
  add_cdc_trigger(main) #Simulate full L1 and trigger on events
  basf2.process(main)
```

Using the generated dataset for testing the NN, one can use Python steering in the `basf2` framework. First, the trigger is added by calling `add_cdc_trigger`. This simulates the preprocessing steps and the events are passed through the neural network trigger to predict z and theta values, which are compared to their true MC values. This is not the same test set as used in the trainer, but is a fifth and independent dataset. This means that the events may not be preprocessed the same way, which might be useful if we want to, for example, see the effect of different event time inputs when the network has been processed with a different one (which can have dramatic effects on the resolution). This dataset is typically larger but need not be. Various analyzer scripts can be produced to make studies of resolution and efficiencies.

A.6. Format conversion

The `basf2` module `NeuroTriggerMLPToTextfile` can be used convert the network weights to a text file, which is a more appropriate format for loading to FPGA. The weights are saved with fixed point precision. This value should be set to 10.

B. Parameters & Additional Plots

Background Campaigns used: Campaign 15 - Phase 2 & Phase 3 (set0)
Tests on real data (see below): `e0008-r02249/dst.physics.0008.02249.1.f00001.root`
Constant parameters:

pdgCodes	[-13, 13]	**nTracks**	1
momentumGeneration	inversePt	**thetaGeneration**	uniformCos
phiGeneration	uniform	**phiParams**	[0, 360]
vertexGeneration	uniform	**xVertexParams**	[0, 0]
yVertexParams	[0, 0]		

Table B.1.: Particle Generation parameters that remained constant for all trainings

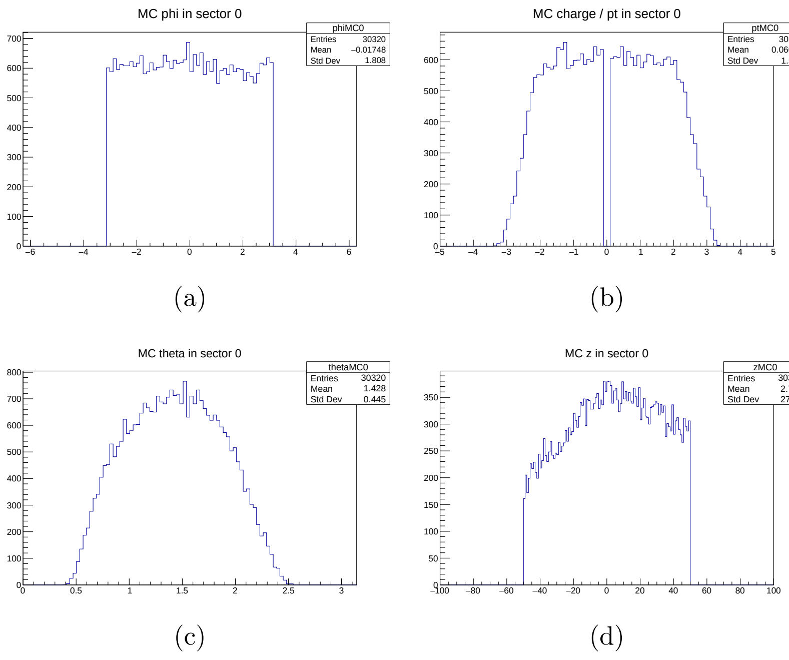


Figure B.1.: MC Particle training data used for Standard network, prepared by `CDCTriggerNeuroTrainer` (for expert 0): (a) distribution of uniformly generated φ; (b) charge/p_t distribution or inverse p_t since charge is 1; (c) uniform θ distribution; (d) distribution of uniformly generated z vertex values, asymmetry due to detector geometry

B.0.1. Standard network

Training parameters

thetaParams	[10, 170]	**zVertexParams**	[-50, 50]
Background	Phase 2	**et_option**	fastest priority
nTrainPrepare	1000	**nValid**	1000
nTrainMax	nWeights*10	**nTrainMin**	nWeights*10
nTest	5000	**nHidden**	81
Training target	MC Particles		

Table B.2.: Training parameters for the Standard network

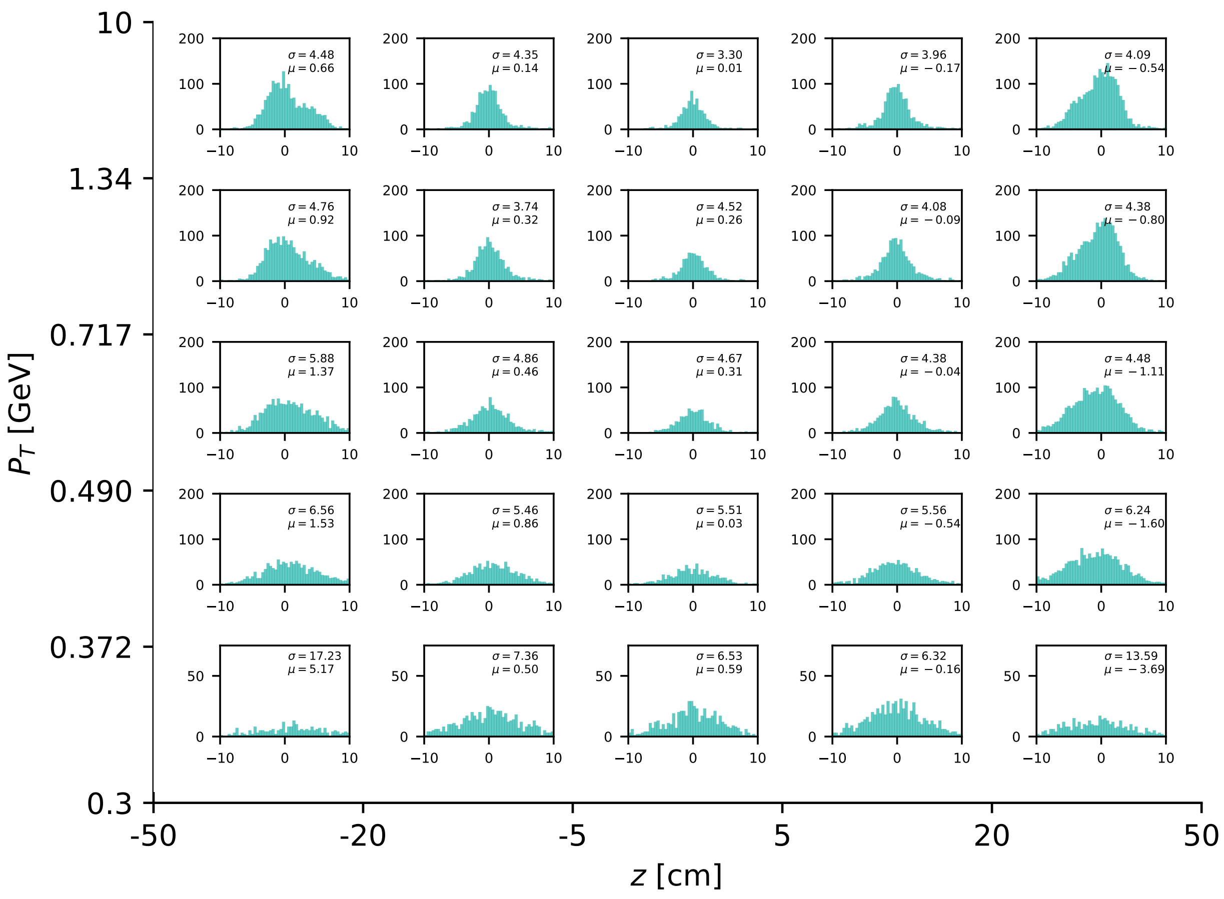
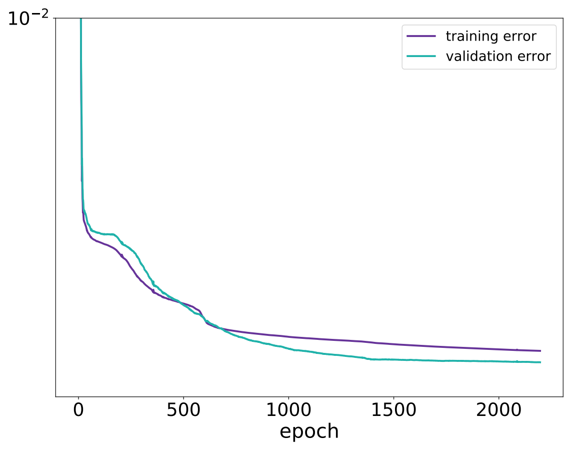
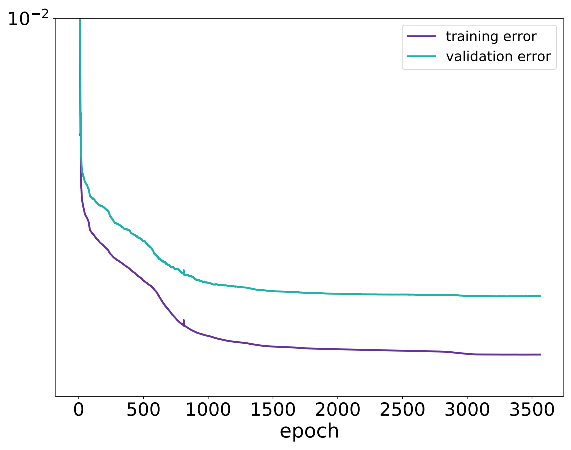


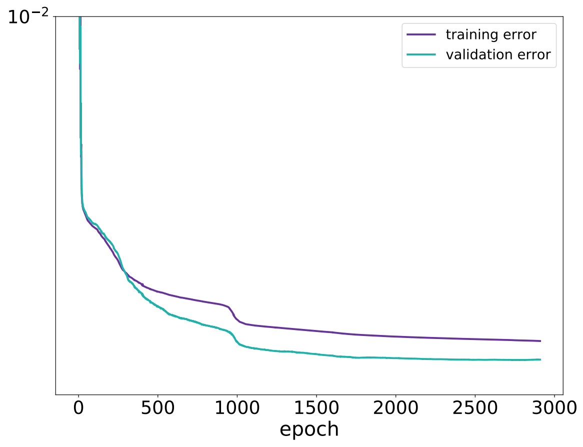
Figure B.2.: The dz resolution as a function of the particle transverse momentum p_t (y-axis) and as a function of the true z-vertex value (x-axis)



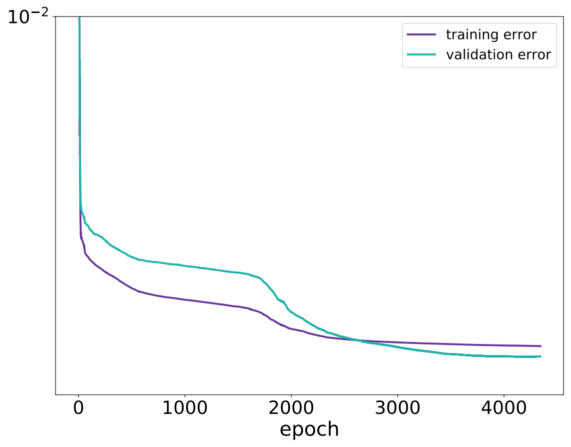
(a)



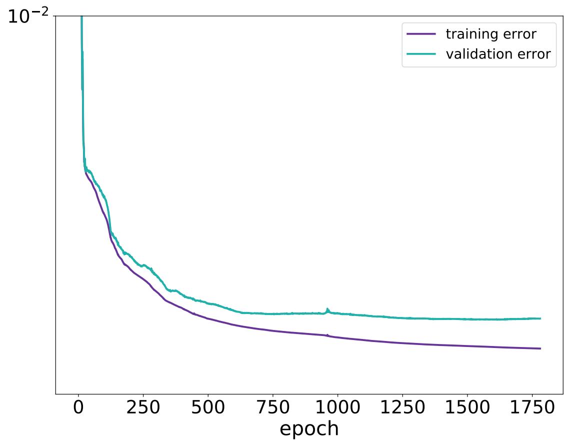
(b)



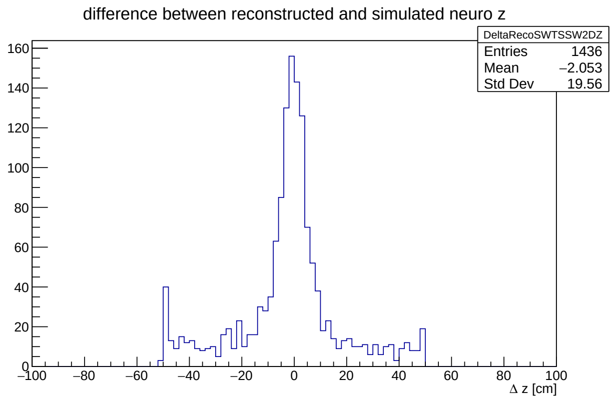
(c)



(d)



(e)



(f)

Figure B.3.: Error curves for each expert network in the Standard training: (a) Expert 0 (full hits); (b) Expert 1 (SL7 missing); (c) Expert 2 (SL5 missing); (d) Expert 3 (SL3 missing); (e) Expert 4 (SL1 missing), details of the training methods can be found in Chapter 4; (f) network tested with real data from early Phase 3

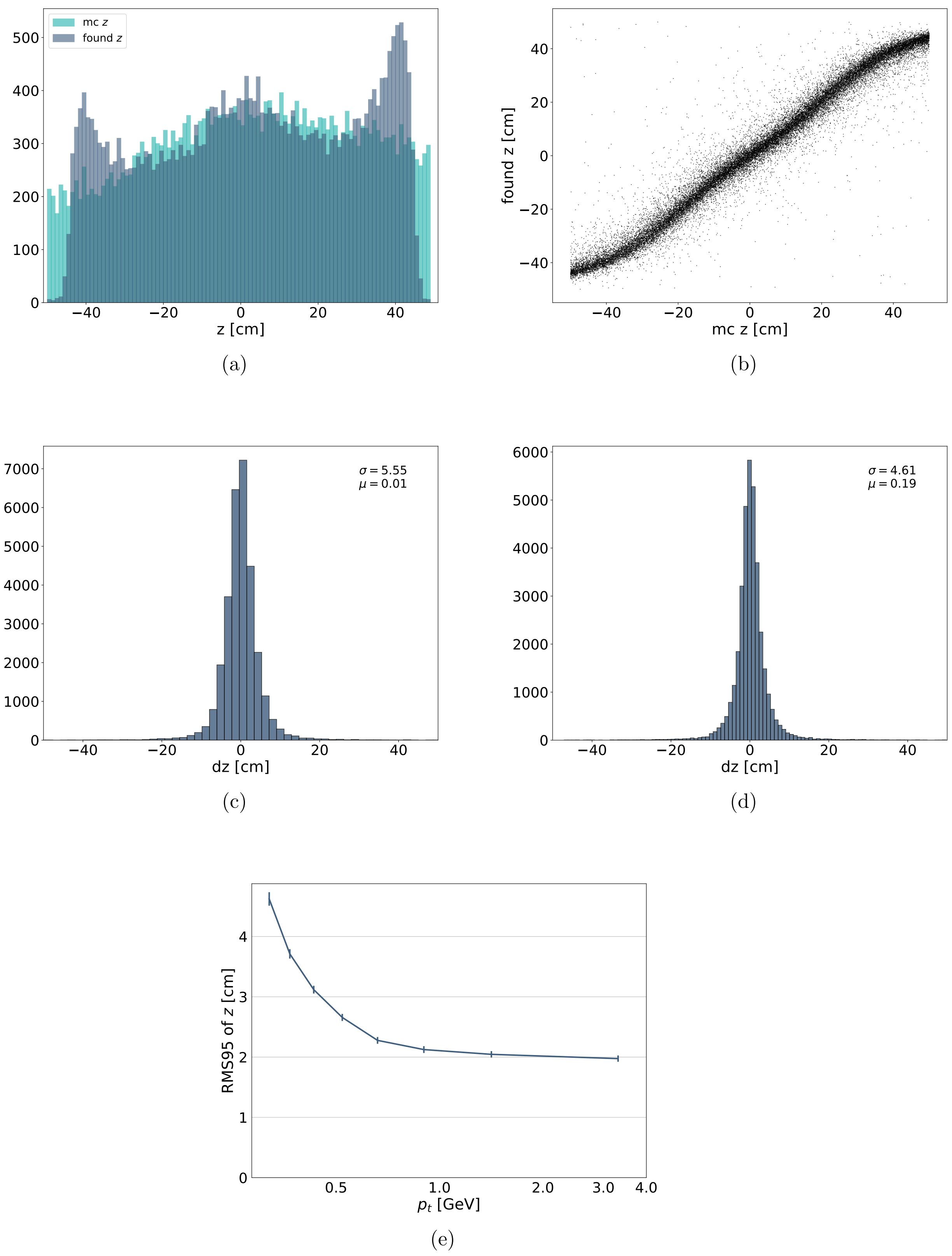
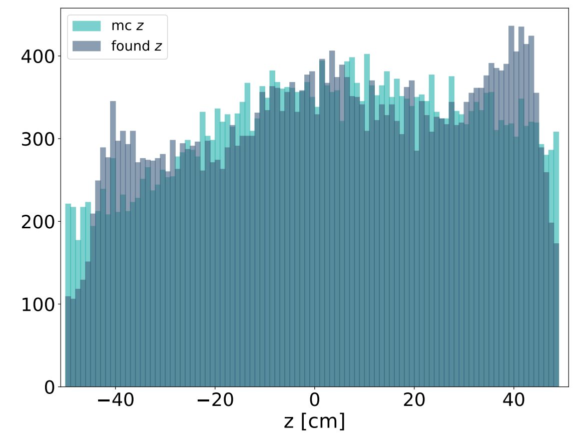
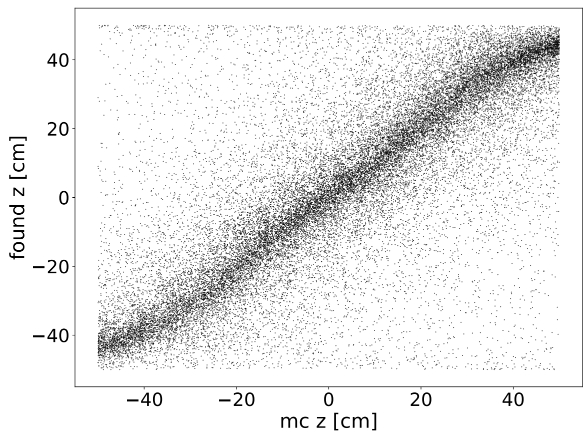


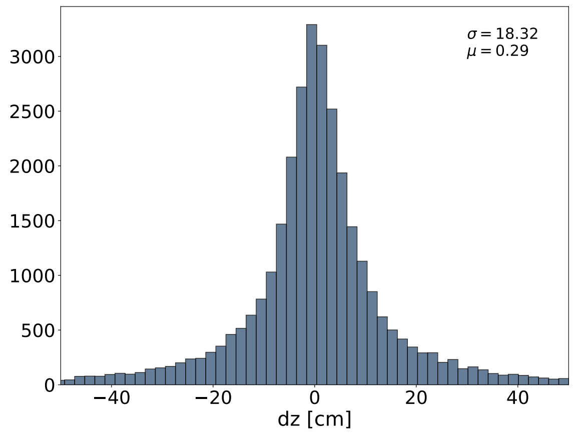
Figure B.4.: **Tested with Phase 2 Background:** Distribution & Resolution plots for Standard network : (a) Distribution of true z and found z-vertex values; (b) Scatter plot of the true z and found z-vertex values; (c) dz resolution (dz = mc z - found z), all (a)–(c) tested with particles generated along z=±50 cm, (d)–(e) tested with particles generated at IP (z=0); (d) dz resolution at IP; (e) p_t-dependent resolution at IP



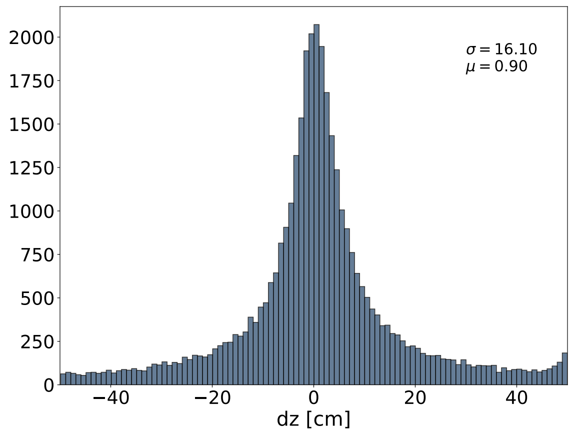
(a)



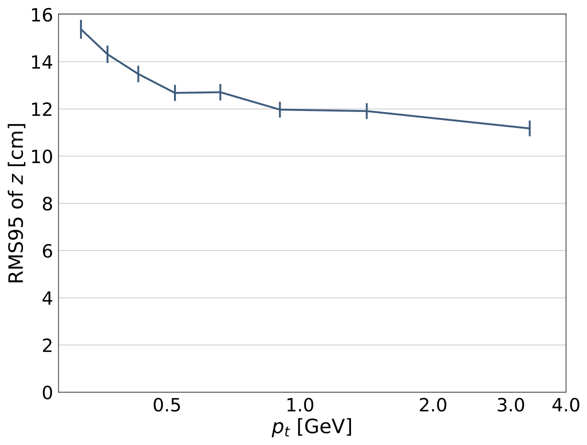
(b)



(c)



(d)



(e)

Figure B.5.: **Tested with Phase 3 Background:** Distribution & Resolution plots for Standard network : (a) Distribution of true z and found z-vertex values; (b) Scatter plot of the true z and found z-vertex values; (c) dz resolution (dz = mc z - found z), all (a)–(c) tested with particles generated along z=±50 cm, (d)–(e) tested with particles generated at IP (z=0); (d) dz resolution at IP; (e) p_t-dependent resolution at IP

B.0.2. No Background network

thetaParams	[10, 170]	**zVertexParams**	[-50, 50]
Background	-	**et_option**	fastest priority
nTrainPrepare	1000	**nValid**	1000
nTrainMax	nWeights*10	**nTrainMin**	nWeights*10
nTest	5000	**nHidden**	81
Training target	MC Particles		

Table B.3.: Training parameters for No Background network

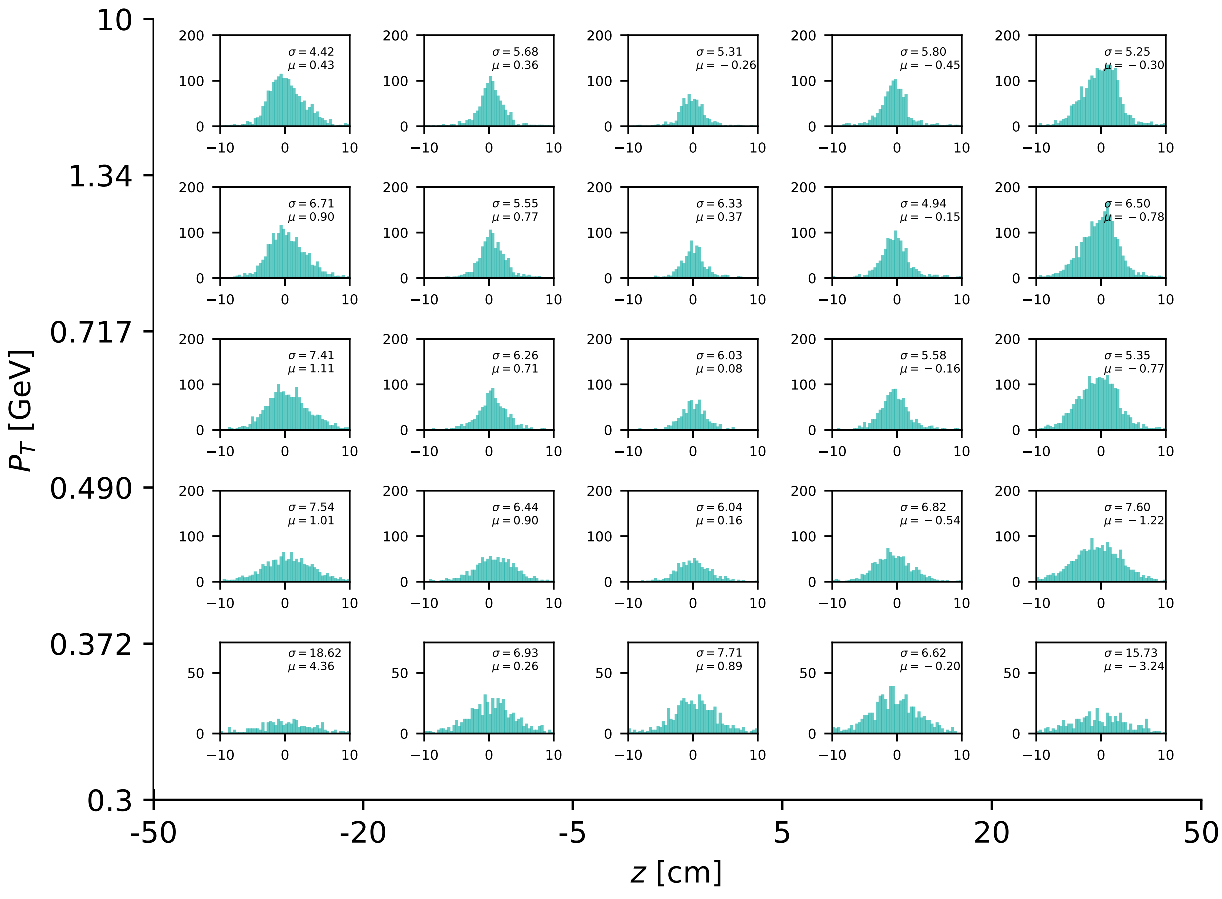
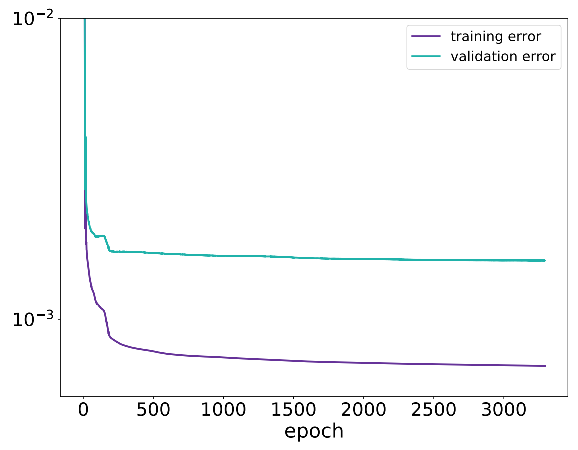
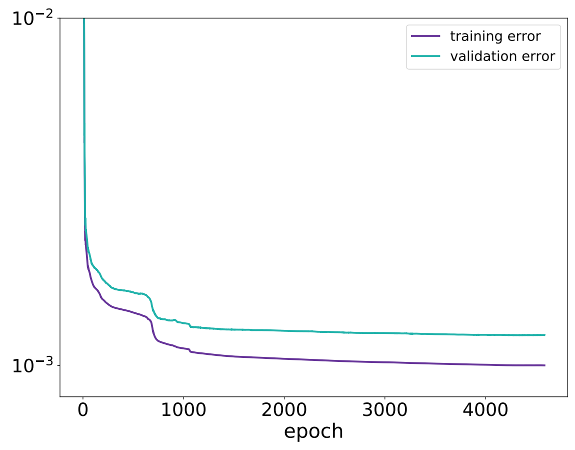


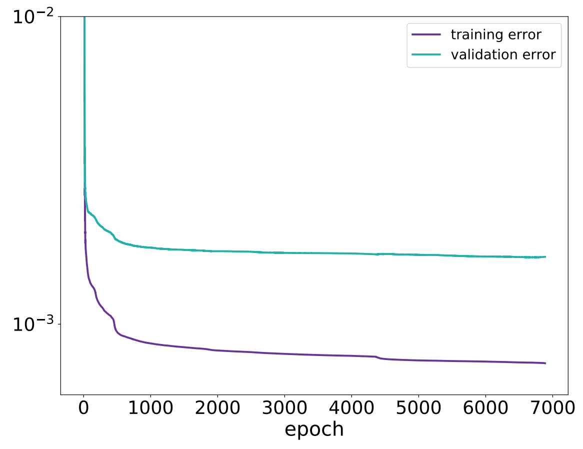
Figure B.6.: **Tested with Phase 2 background:**The dz resolution as a function of the particle transverse momentum p_t (y-axis) and as a function of the true z-vertex value (x-axis)



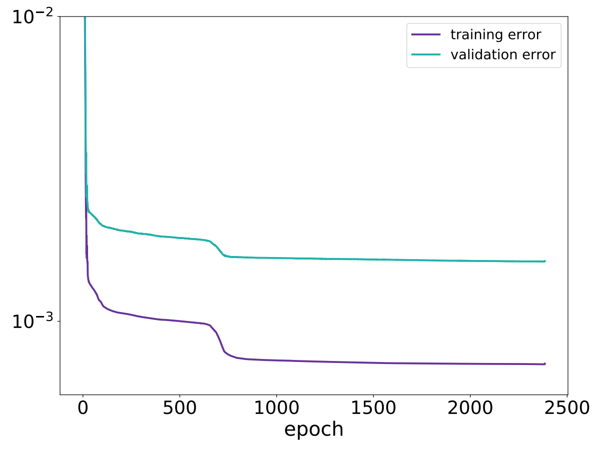
(a)



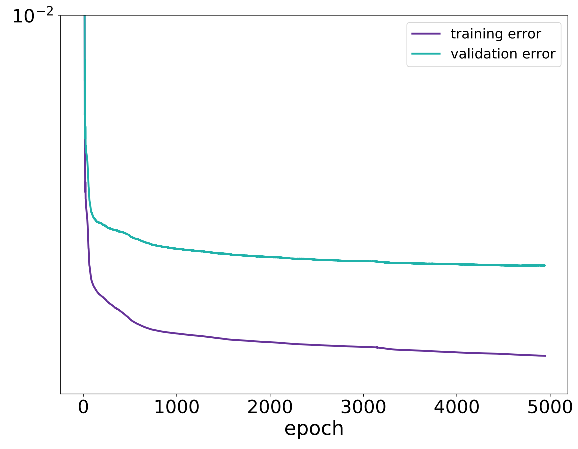
(b)



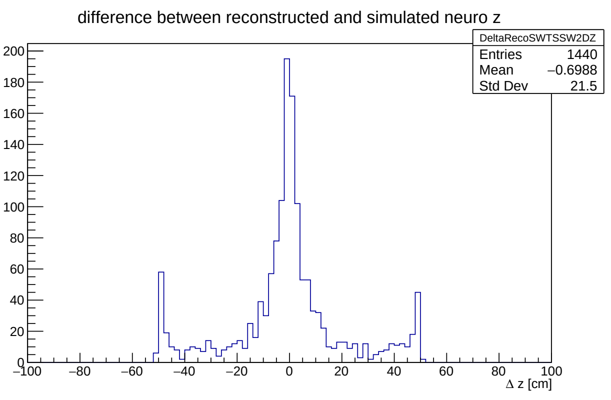
(c)



(d)

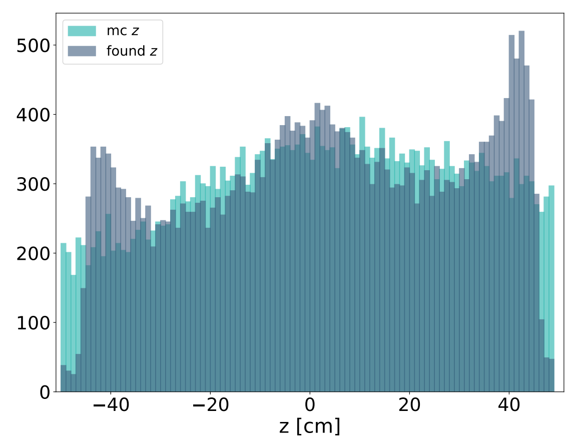


(e)

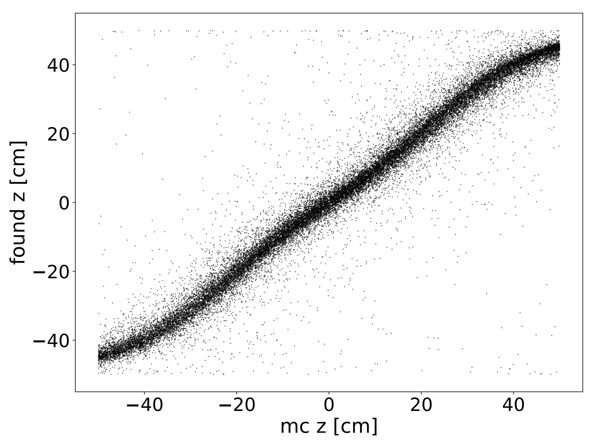


(f)

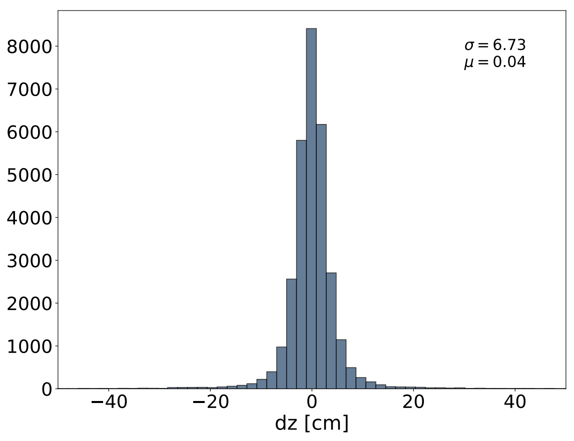
Figure B.7.: Error curves for each expert network in the No Background training: (a) Expert 0 (full hits); (b) Expert 1 (SL7 missing); (c) Expert 2 (SL5 missing); (d) Expert 3 (SL3 missing); (e) Expert 4 (SL1 missing), details of the training methods can be found in Chapter 4; (f) network tested with real data from early Phase 3; (f) network tested with real data from early Phase 3



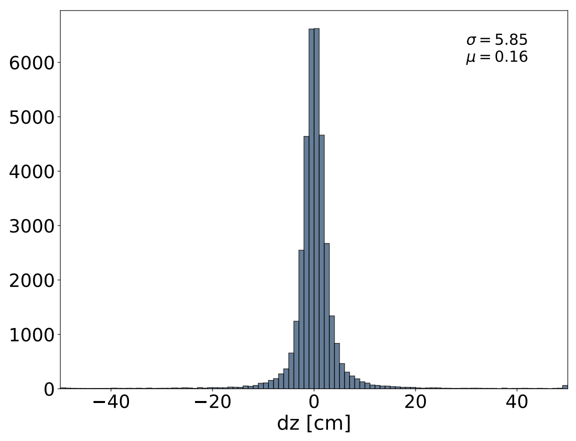
(a)



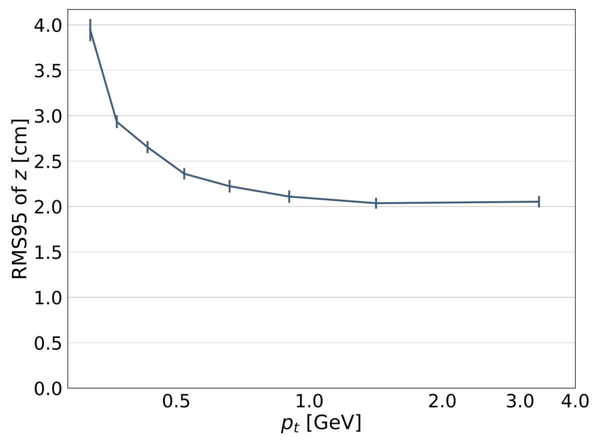
(b)



(c)

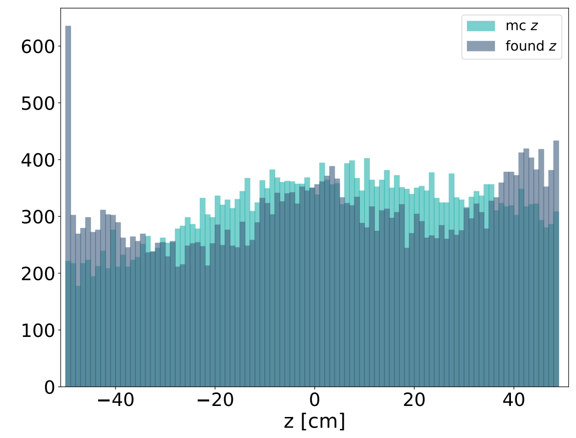


(d)

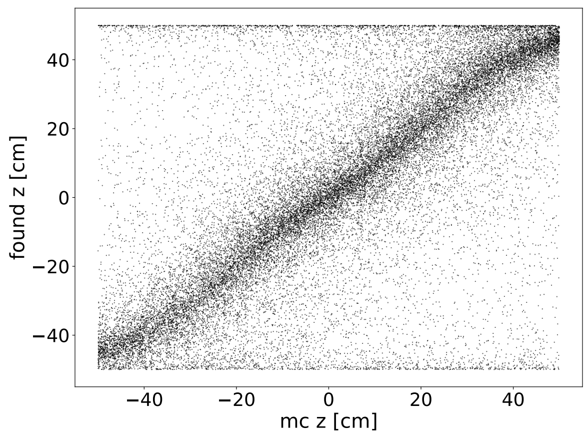


(e)

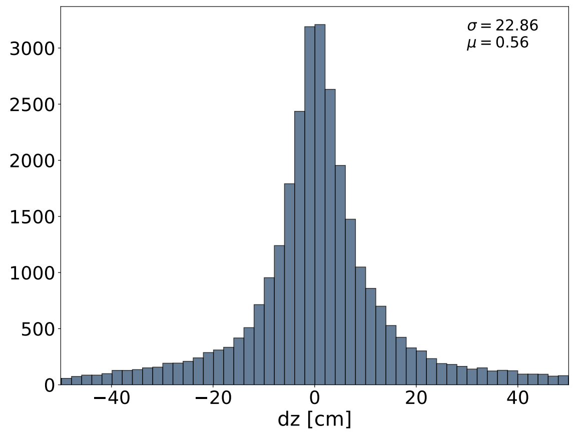
Figure B.8.: **Tested with Phase 2 Background:** Distribution & Resolution plots for No background network: (a) Distribution of true z and found z-vertex values; (b) Scatter plot of the true z and found z-vertex values; (c) dz resolution (dz = mc z - found z), all (a)–(c) tested with particles generated along z=±50 cm, (d)–(e) tested with particles generated at IP (z=0); (d) dz resolution at IP; (e) p_t-dependent resolution at IP; (f) network tested with real data from early Phase 3



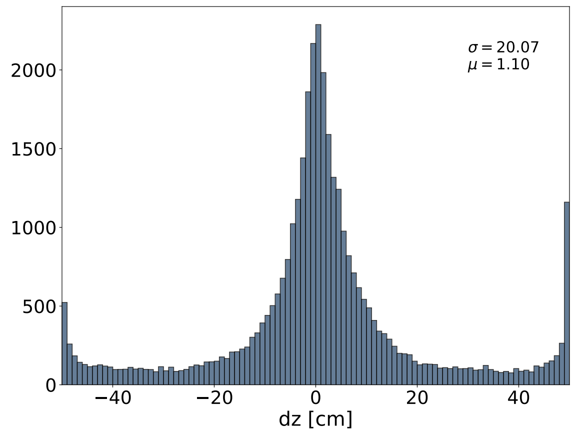
(a)



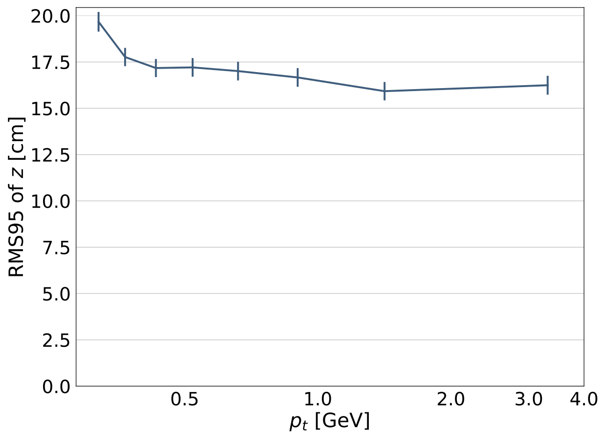
(b)



(c)



(d)



(e)

Figure B.9.: **Tested with Phase 3 Background:** Distribution & Resolution plots for No background network: (a) Distribution of true z and found z-vertex values; (b) Scatter plot of the true z and found z-vertex values; (c) dz resolution (dz = mc z - found z), all (a)–(c) tested with particles generated along z=±50 cm, (d)–(e) tested with particles generated at IP (z=0); (d) dz resolution at IP; (e) p_t-dependent resolution at IP; (f) network tested with real data from early Phase 3

B.0.3. Phase 3 network

thetaParams	[10, 170]	**zVertexParams**	[-50, 50]
Background	Phase 3	**et_option**	fastest priority
nTrainPrepare	1000	**nValid**	1000
nTrainMax	nWeights*10	**nTrainMin**	nWeights*10
nTest	5000	**nHidden**	81
Training target	MC Particles		

Table B.4.: Training parameters for the Phase 3 network

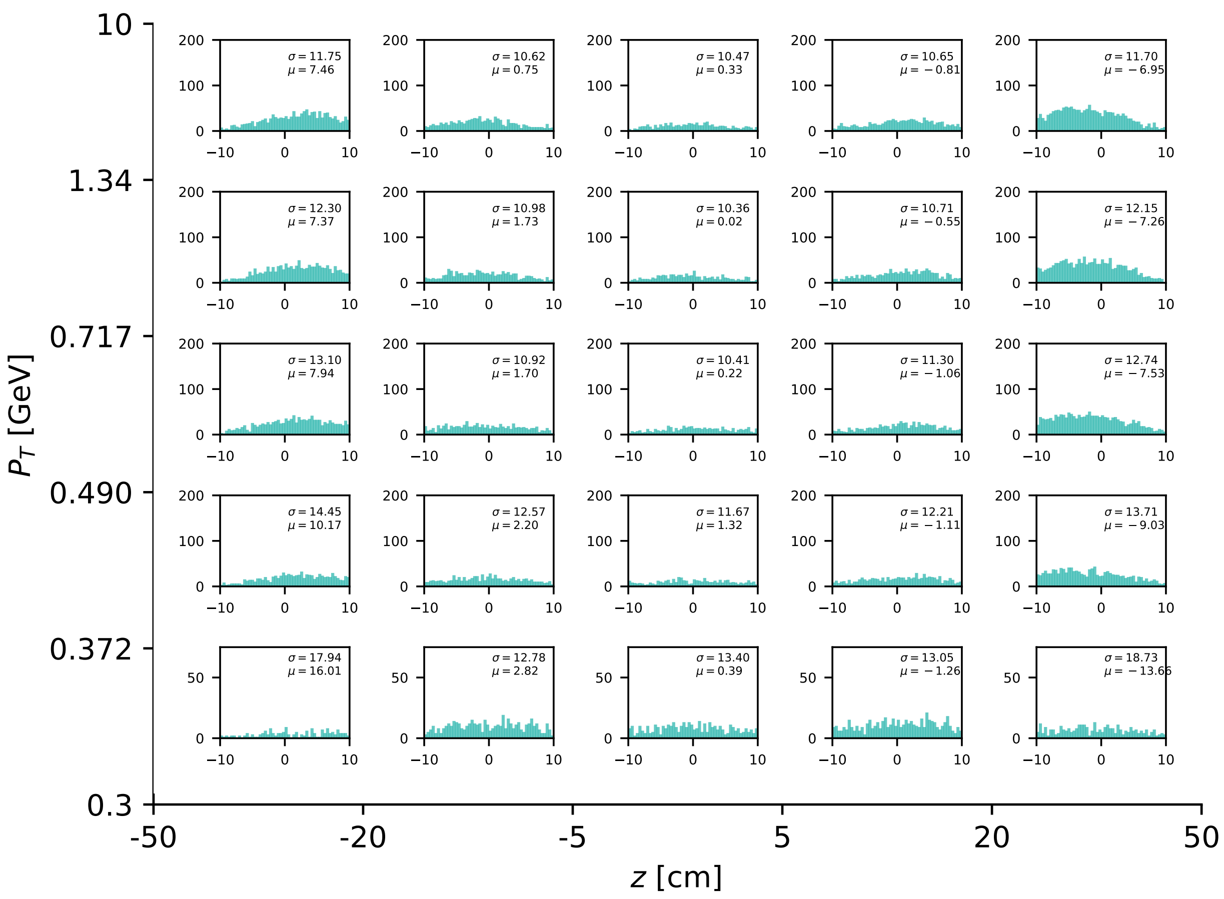
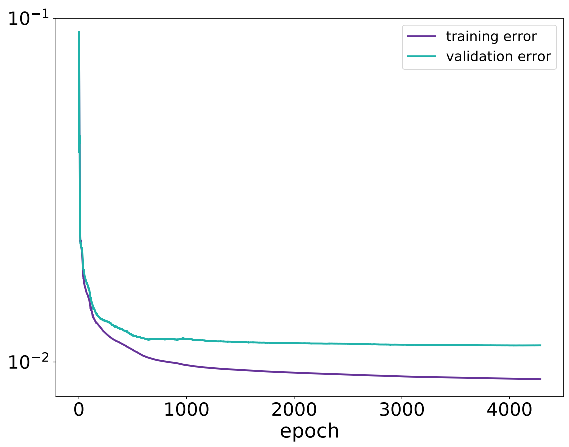
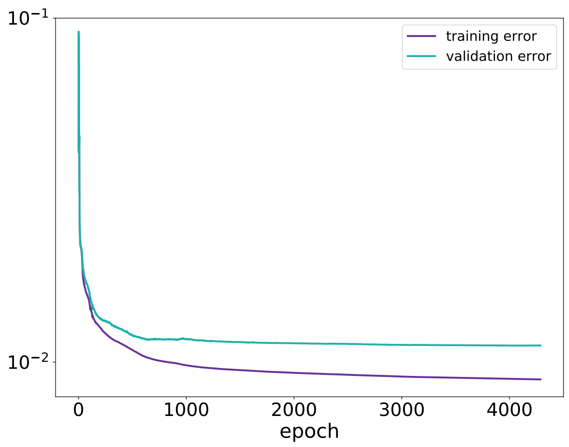


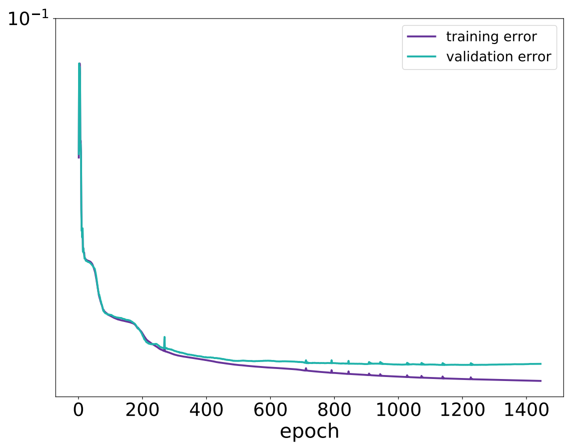
Figure B.10.: The dz resolution as a function of the particle transverse momentum p_t (y-axis) and as a function of the true z-vertex value (x-axis)



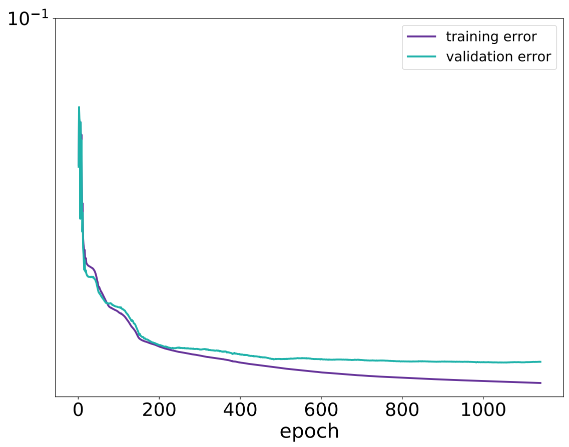
(a)



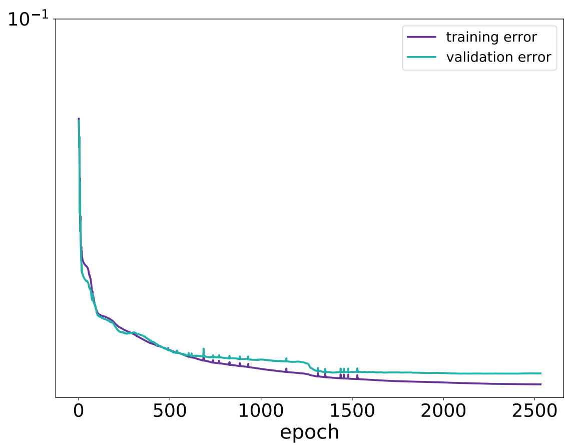
(b)



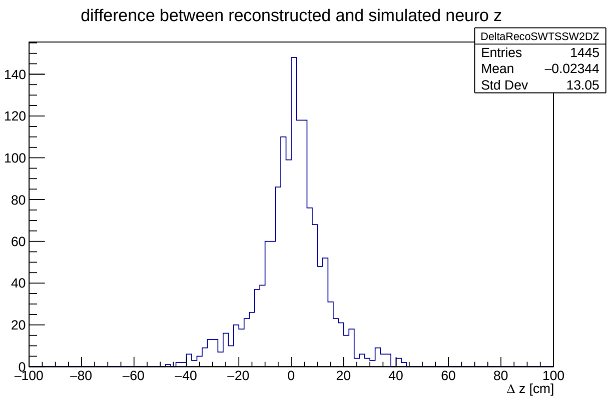
(c)



(d)

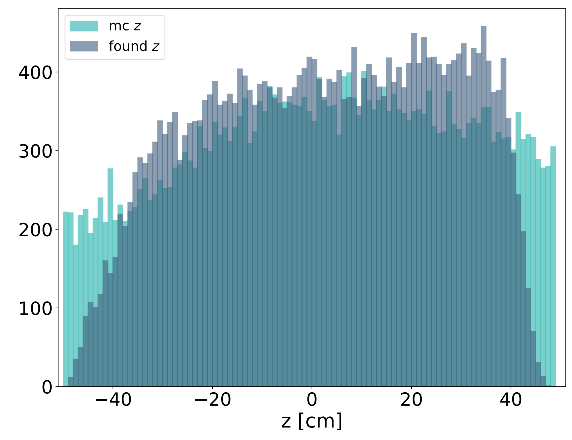


(e)

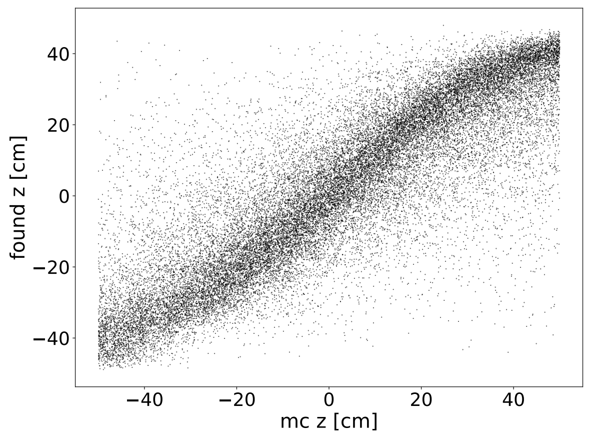


(f)

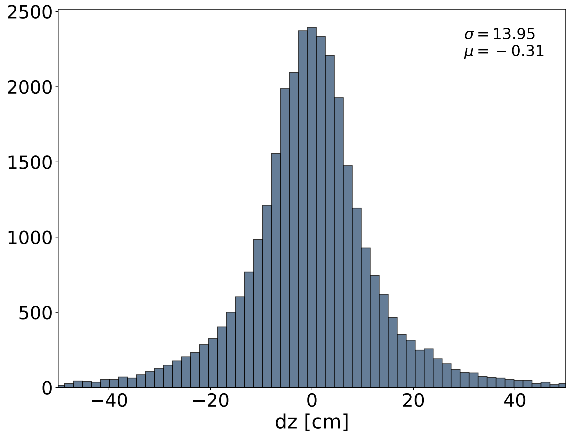
Figure B.11.: Error curves for each expert network in the Phase 3 training: (a) Expert 0 (full hits); (b) Expert 1 (SL7 missing); (c) Expert 2 (SL5 missing); (d) Expert 3 (SL3 missing); (e) Expert 4 (SL1 missing), details of the training methods can be found in Chapter 4; (f) network tested with real data from early Phase 3; (f) network tested with real data from early Phase 3



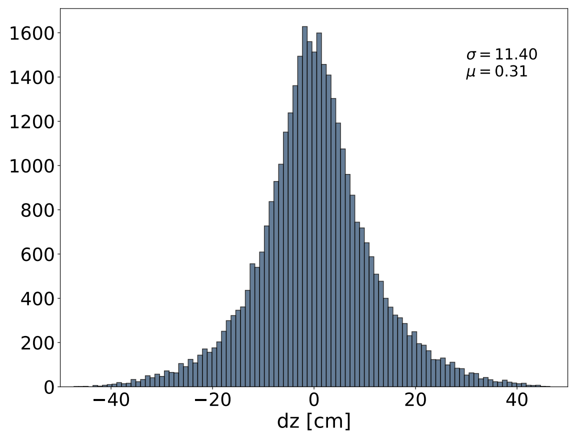
(a)



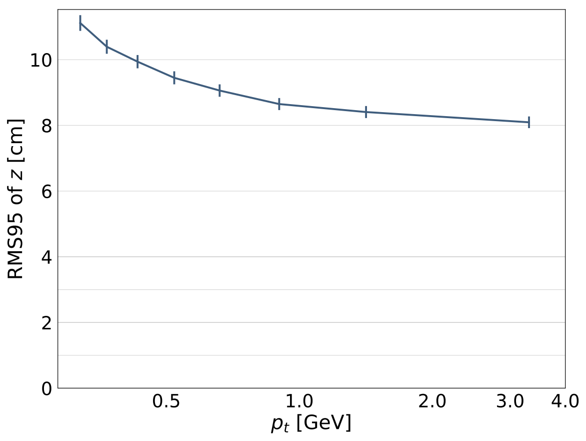
(b)



(c)



(d)



(e)

Figure B.12.: **Tested with Phase 3 Background:** Distribution & Resolution plots for Phase 3 network: (a) Distribution of true z and found z-vertex values; (b) Scatter plot of the true z and found z-vertex values; (c) dz resolution (dz = mc z - found z), all (a)–(c) tested with particles generated along z=±50 cm, (d)–(e) tested with particles generated at IP (z=0); (d) dz resolution at IP; (e) p_t-dependent resolution at IP

B.0.4. ETF Phase 2 Threshold 0 network

thetaParams	[10, 170]	**zVertexParams**	[-50, 50]
Background	Phase 2	**et_option**	ETF (threshold=0)
nTrainPrepare	1000	**nValid**	1000
nTrainMax	nWeights*10	**nTrainMin**	nWeights*10
nTest	5000	**nHidden**	81
Training target	MC Particles		

Table B.5.: Training parameters for the ETF Phase 2 Threshold 0 network

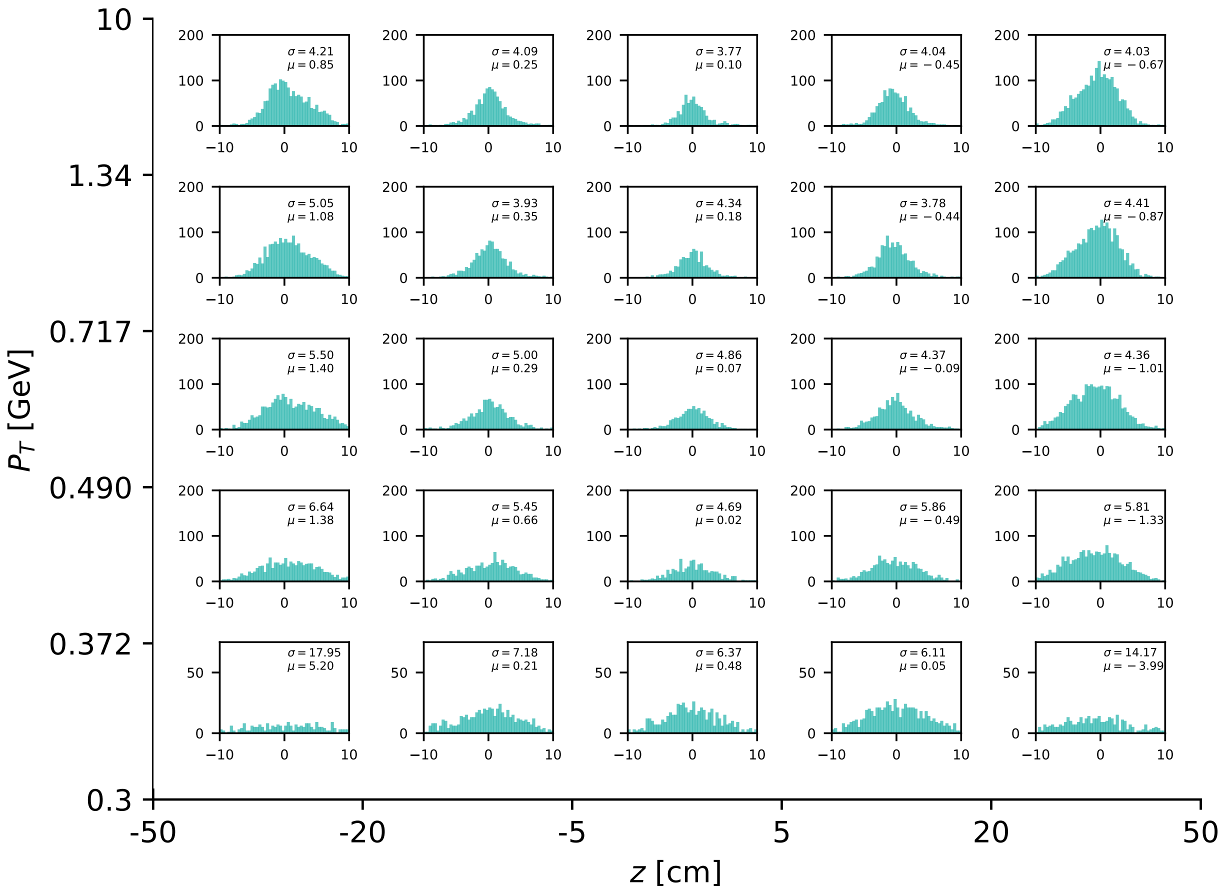
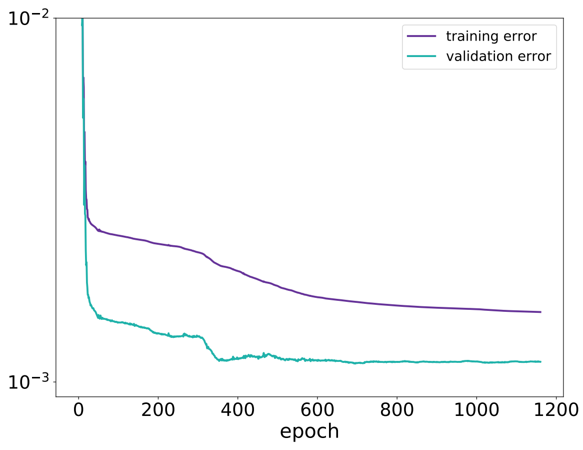
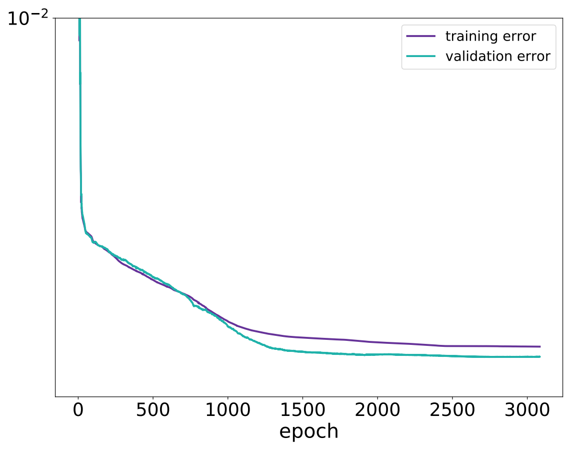


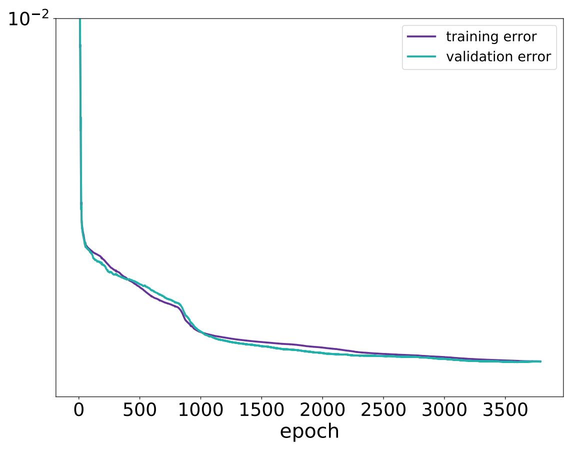
Figure B.13.: The dz resolution as a function of the particle transverse momentum p_t (y-axis) and as a function of the true z-vertex value (x-axis)



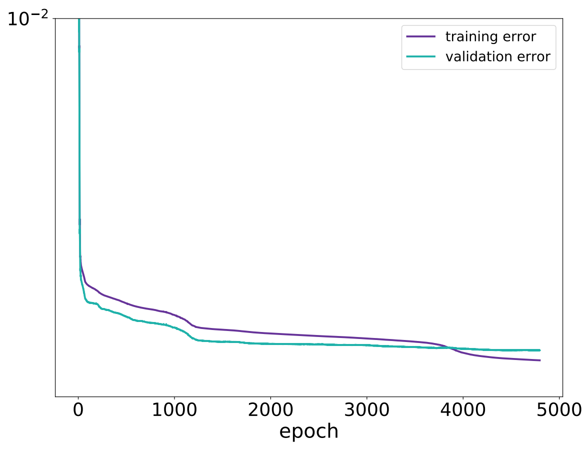
(a)



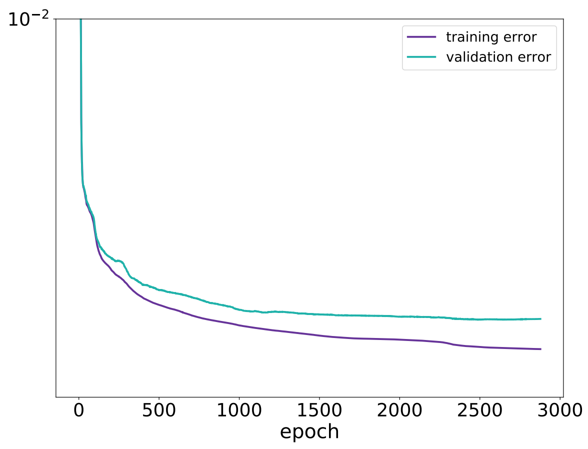
(b)



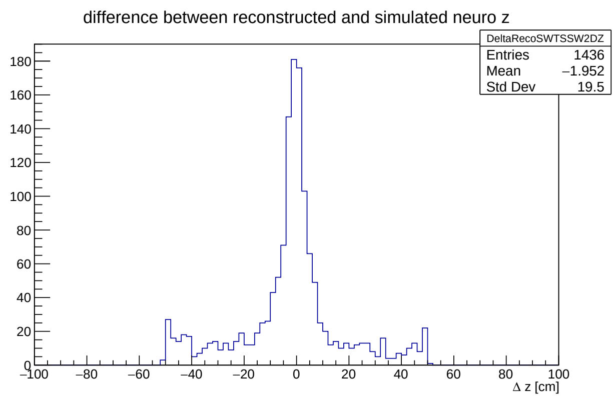
(c)



(d)

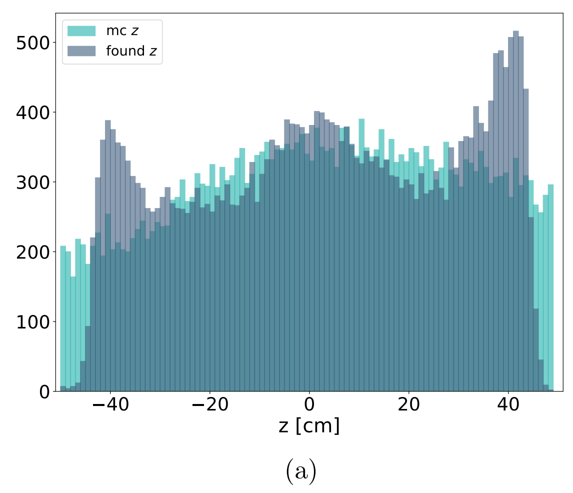


(e)

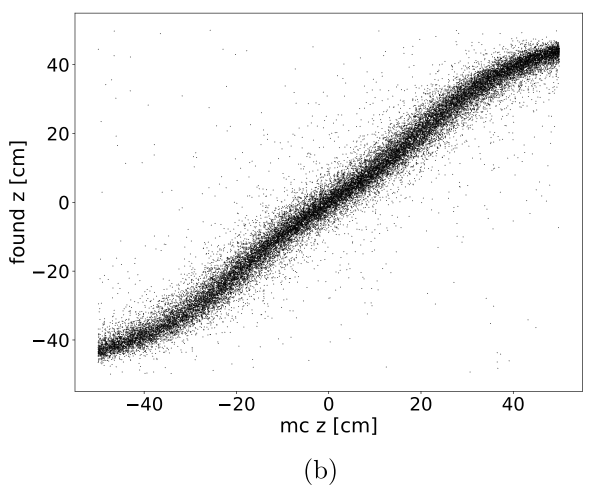


(f)

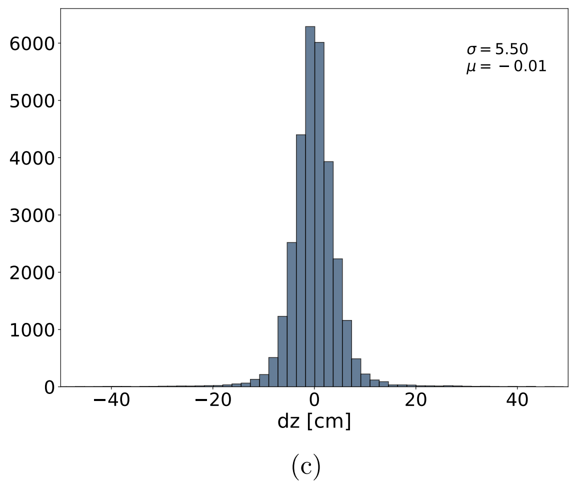
Figure B.14.: Error curves for each expert network in the ETF Phase 2 Threshold 0 network training: (a) Expert 0 (full hits); (b) Expert 1 (SL7 missing); (c) Expert 2 (SL5 missing); (d) Expert 3 (SL3 missing); (e) Expert 4 (SL1 missing), details of the training methods can be found in Chapter 4; (f) network tested with real data from early Phase 3



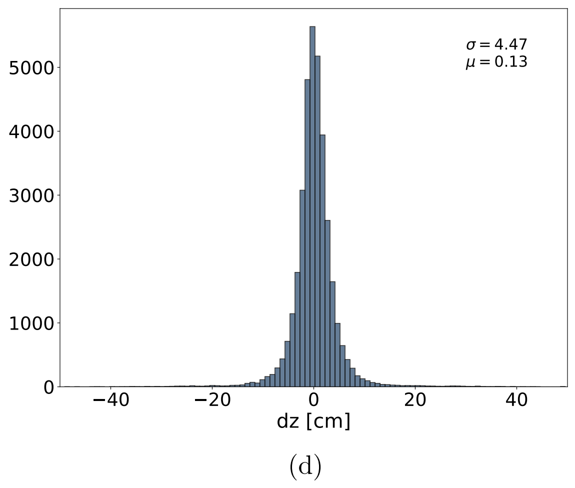
(a)



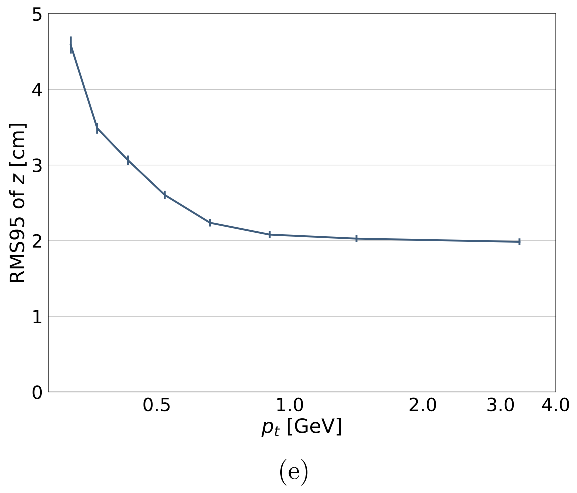
(b)



(c)



(d)



(e)

Figure B.15.: Distribution & Resolution plots for ETF Phase 2 Threshold 0 network: (a) Distribution of true z and found z-vertex values; (b) Scatter plot of the true z and found z-vertex values; (c) dz resolution (dz = mc z - found z), all (a)–(c) tested with particles generated along z=±50 cm, (d)–(e) tested with particles generated at IP (z=0); (d) dz resolution at IP; (e) p_t-dependent resolution at IP

B.0.5. ETF Phase 2 Threshold 1 network

thetaParams	[10, 170]	**zVertexParams**	[-50, 50]
Background	Phase 2	**et_option**	ETF (threshold=1)
nTrainPrepare	1000	**nValid**	1000
nTrainMax	nWeights*10	**nTrainMin**	nWeights*10
nTest	5000	**nHidden**	81
Training target	MC Particles		

Table B.6.: Training parameters for ETF Phase 2 Threshold 1 network

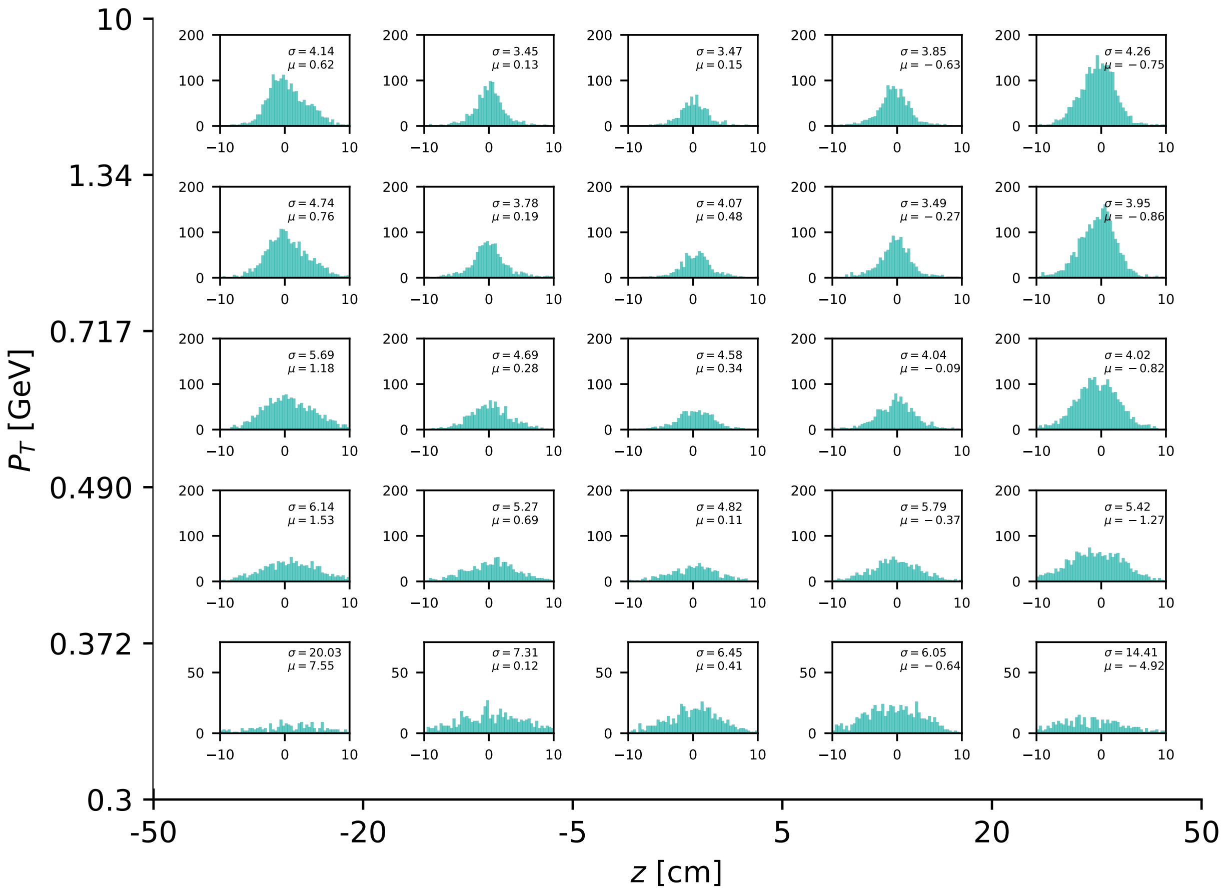


Figure B.16.: The dz resolution as a function of the particle transverse momentum p_t (y-axis) and as a function of the true z-vertex value (x-axis)

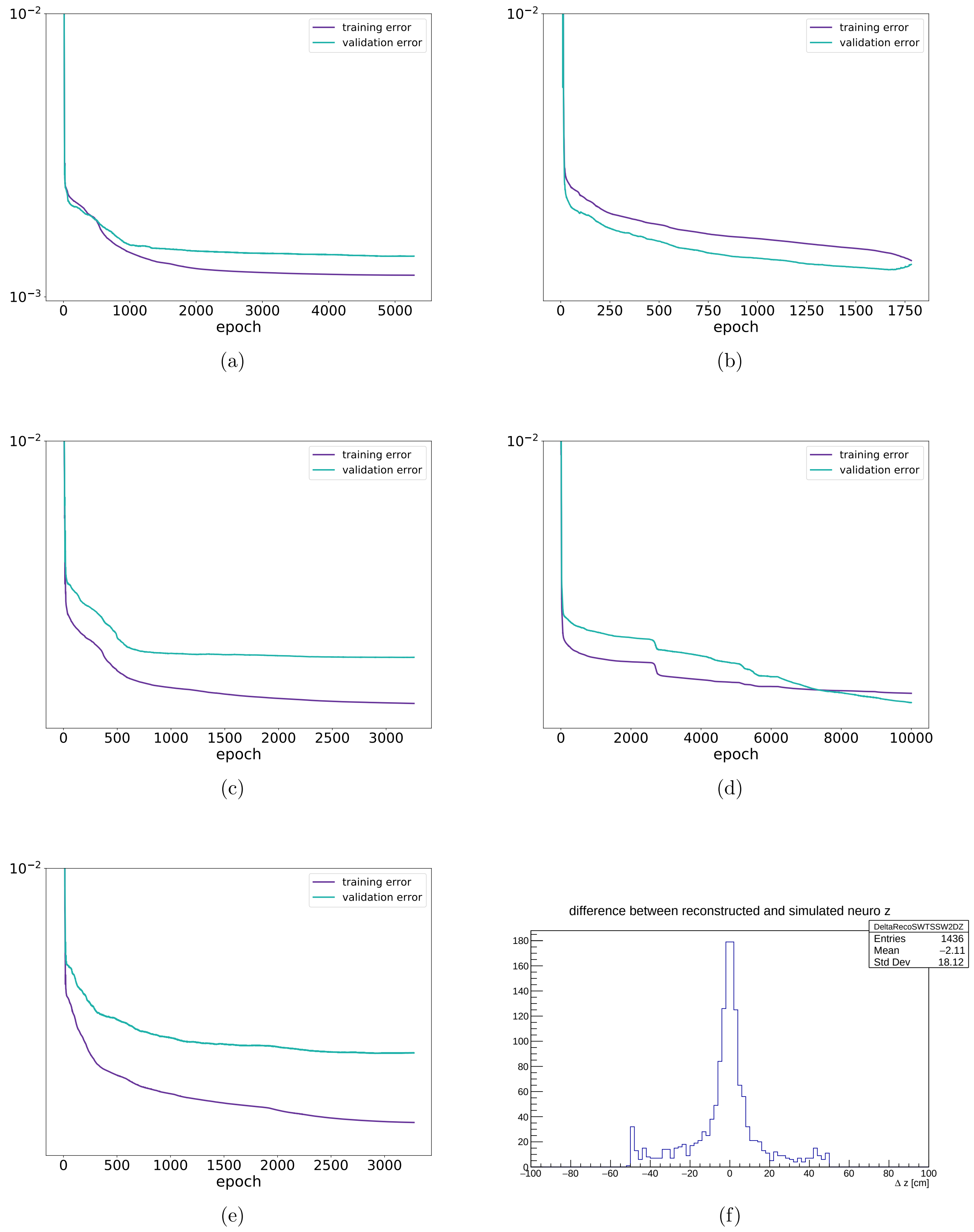
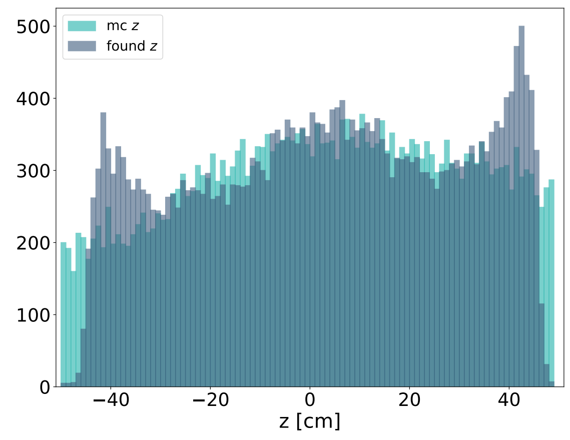
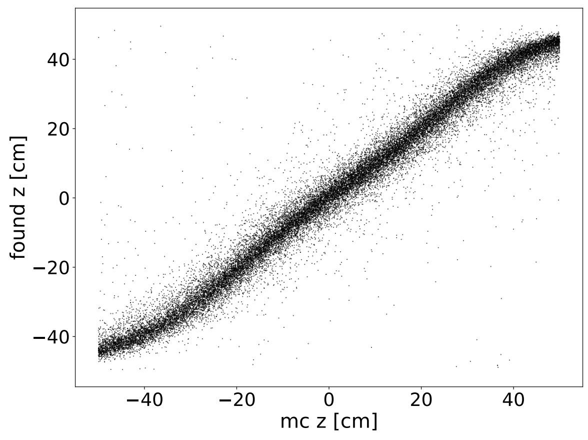


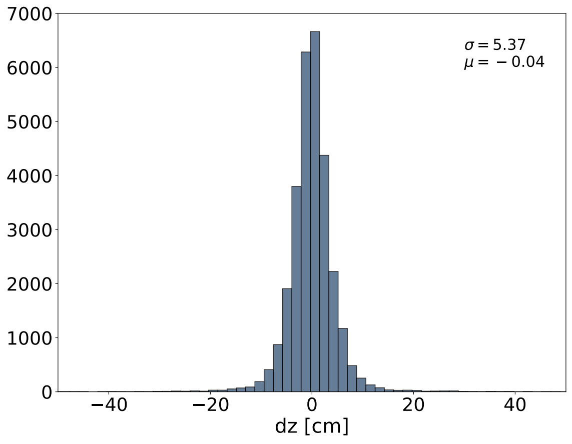
Figure B.17.: Error curves for each expert network in the ETF Phase 2 Threshold 1 network training: (a) Expert 0 (full hits); (b) Expert 1 (SL7 missing); (c) Expert 2 (SL5 missing); (d) Expert 3 (SL3 missing); (e) Expert 4 (SL1 missing), details of the training methods can be found in Chapter 4; (f) network tested with real data from early Phase 3



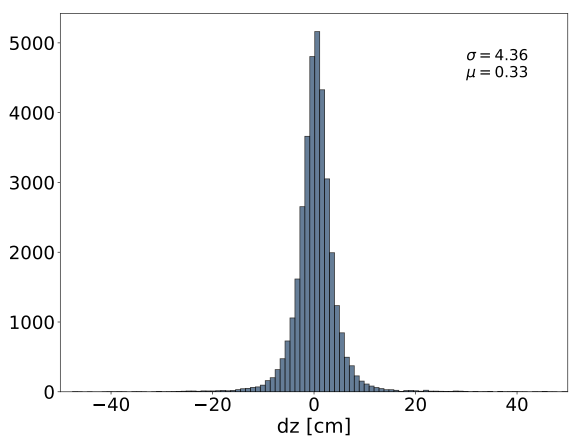
(a)



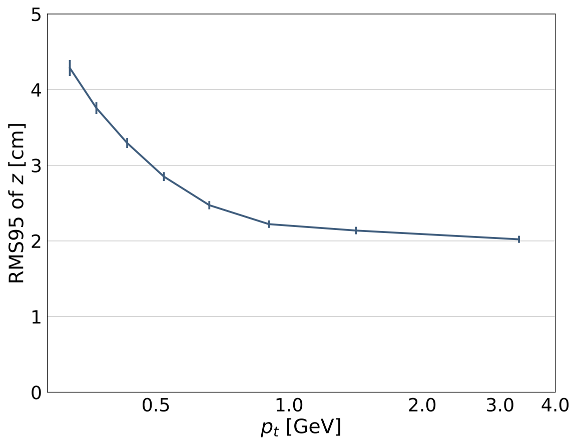
(b)



(c)



(d)



(e)

Figure B.18.: Distribution & Resolution plots for ETF Phase 2 Threshold 1 network: (a) Distribution of true z and found z-vertex values; (b) Scatter plot of the true z and found z-vertex values; (c) dz resolution (dz = mc z - found z), all (a)–(c) tested with particles generated along z=±50 cm, (d)–(e) tested with particles generated at IP (z=0); (d) dz resolution at IP; (e) p_t-dependent resolution at IP

B.0.6. ETF Phase 2 Threshold 2 network

thetaParams	[10, 170]	**zVertexParams**	[-50, 50]
Background	Phase 2	**et_option**	ETF (threshold=2)
nTrainPrepare	1000	**nValid**	1000
nTrainMax	nWeights*10	**nTrainMin**	nWeights*10
nTest	5000	**nHidden**	81
Training target	MC Particles		

Table B.7.: Training parameters for ETF Phase 2 Threshold 2 network

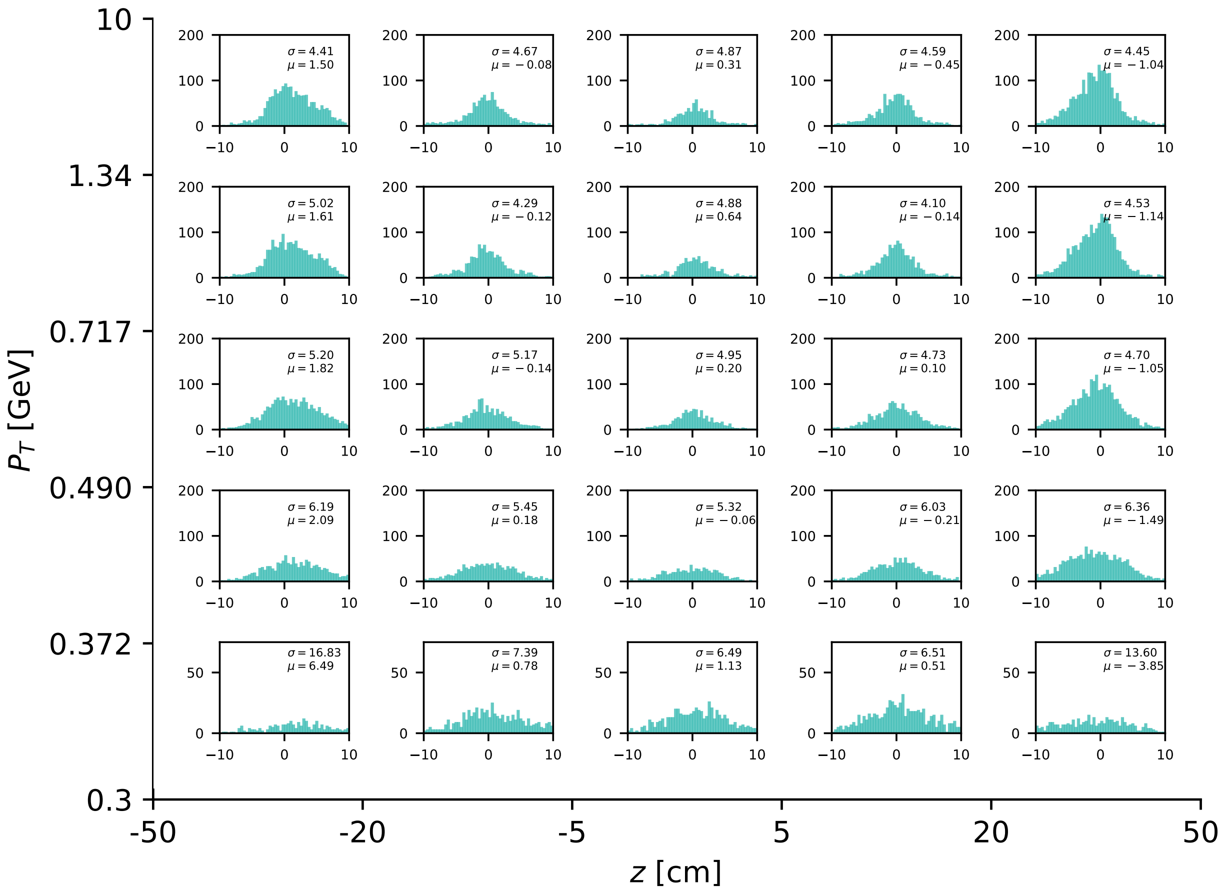
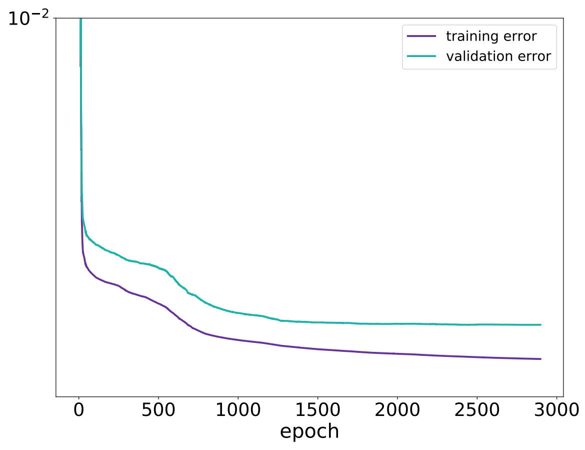
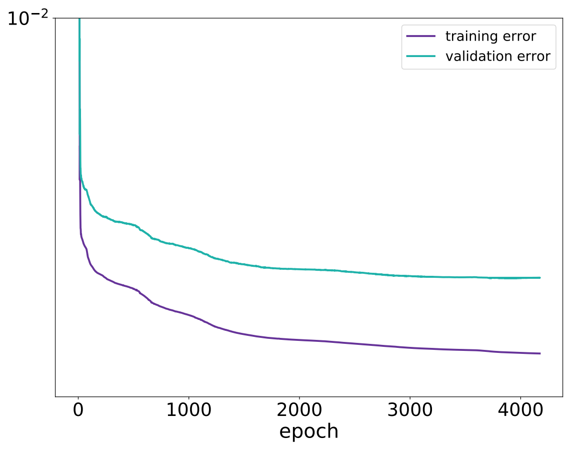


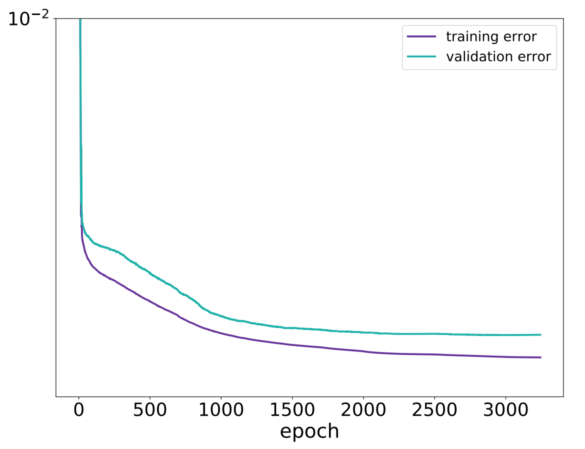
Figure B.19.: The dz resolution as a function of the particle transverse momentum p_t (y-axis) and as a function of the true z-vertex value (x-axis)



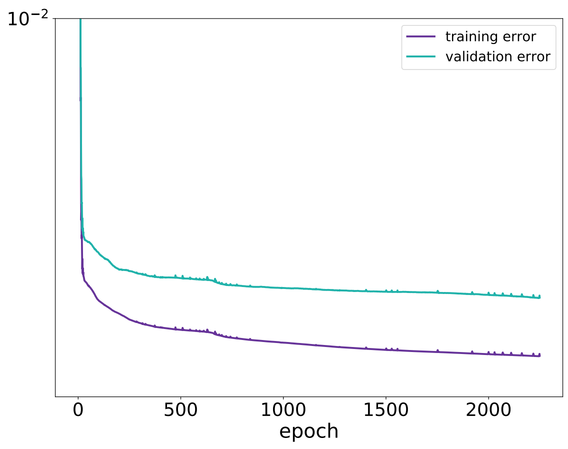
(a)



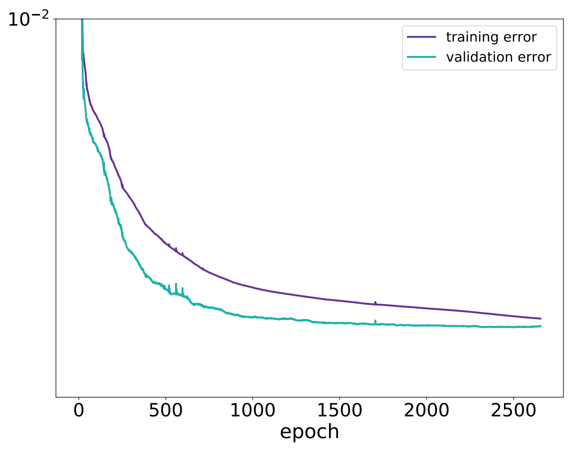
(b)



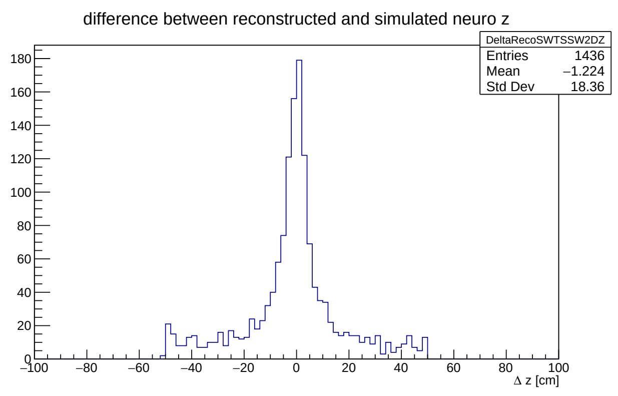
(c)



(d)

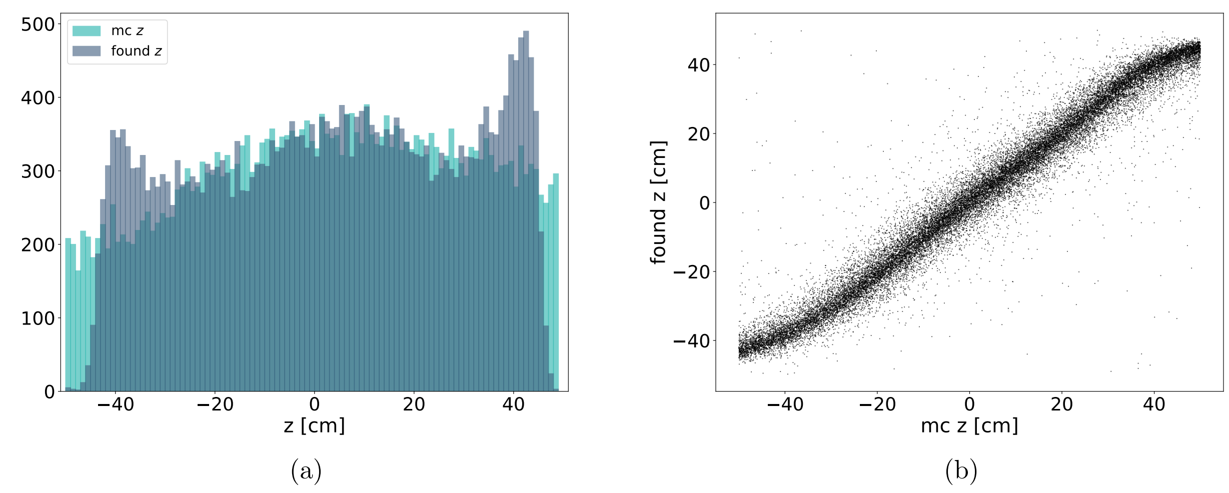


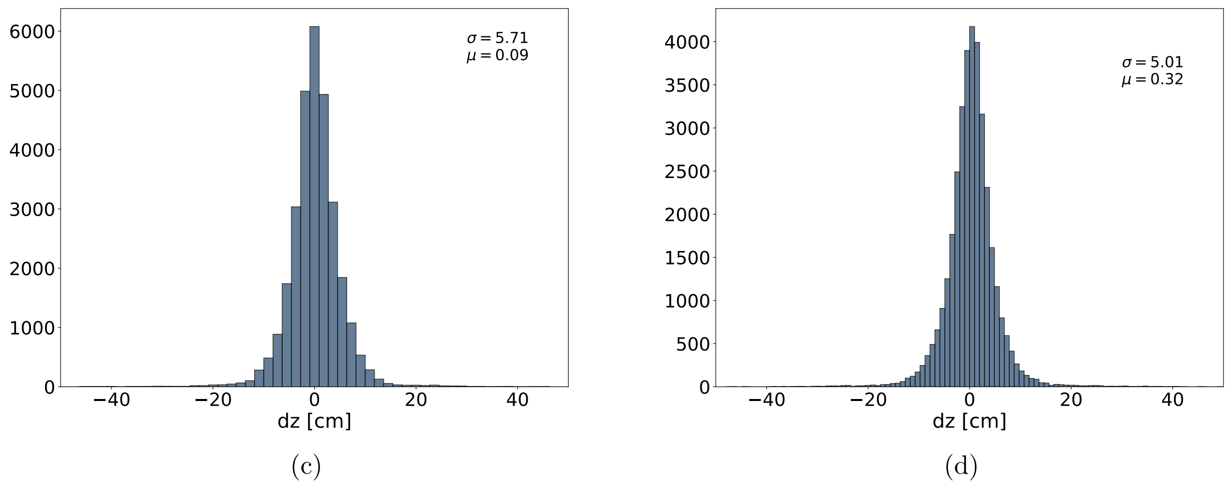
(e)



(f)

Figure B.20.: Error curves for each expert network in the ETF Phase 2 Threshold 1 network training: (a) Expert 0 (full hits); (b) Expert 1 (SL7 missing); (c) Expert 2 (SL5 missing); (d) Expert 3 (SL3 missing); (e) Expert 4 (SL1 missing), details of the training methods can be found in Chapter 4; (f) network tested with real data from early Phase 3





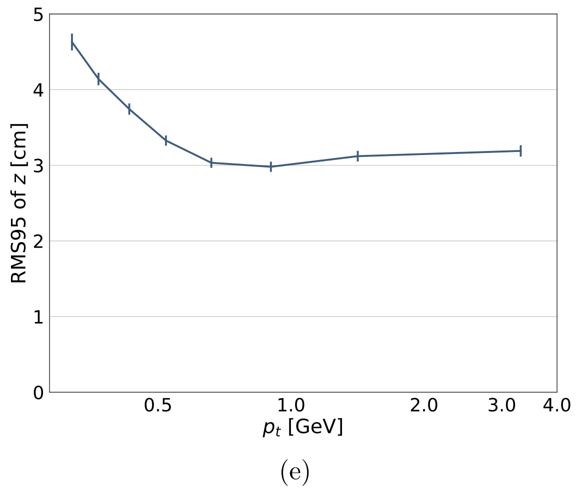


Figure B.21.: Distribution & Resolution plots for ETF Phase 2 Threshold 2 network: (a) Distribution of true z and found z-vertex values; (b) Scatter plot of the true z and found z-vertex values; (c) dz resolution (dz = mc z - found z), all (a)–(c) tested with particles generated along z=±50 cm, (d)–(e) tested with particles generated at IP (z=0); (d) dz resolution at IP; (e) p_t-dependent resolution at IP

B.0.7. ETF Phase 2 Threshold 3 network

thetaParams	[10, 170]	**zVertexParams**	[-50, 50]
Background	Phase 2	**et_option**	ETF (threshold=3)
nTrainPrepare	1000	**nValid**	1000
nTrainMax	nWeights*10	**nTrainMin**	nWeights*10
nTest	5000	**nHidden**	81
Training target	MC Particles		

Table B.8.: Training parameters for ETF Phase 2 Threshold 3 network

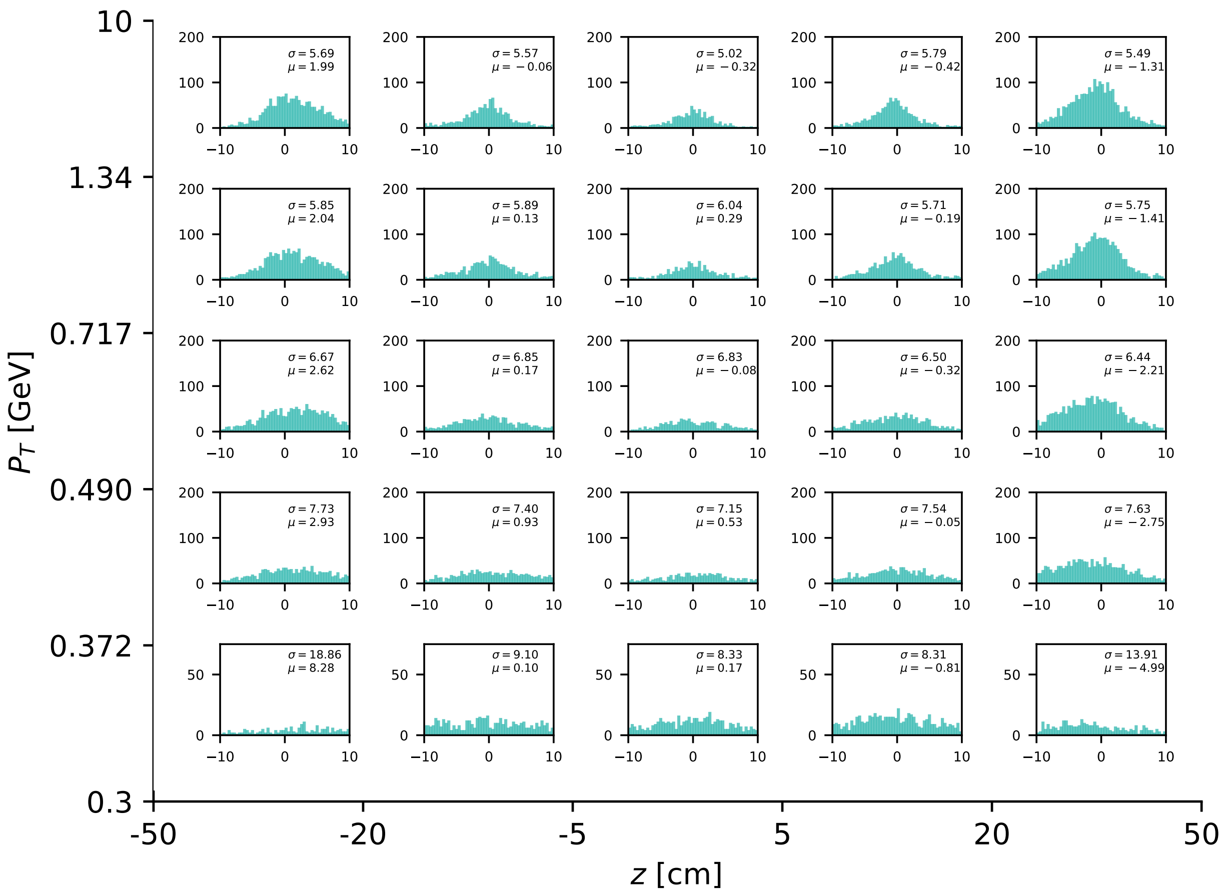
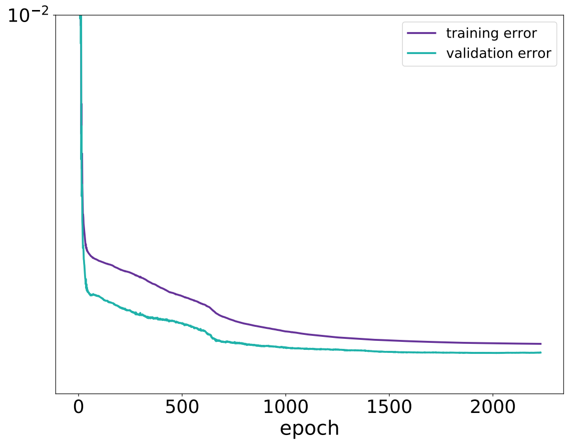
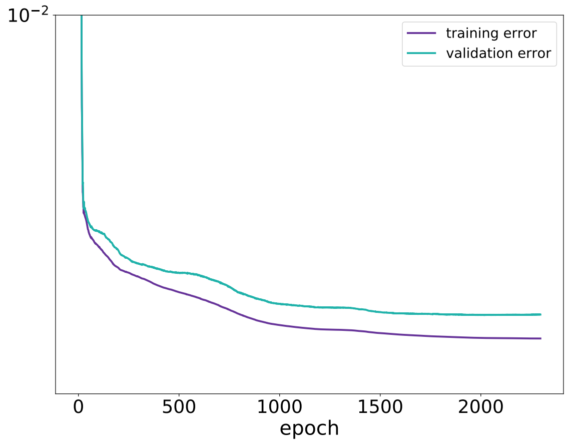


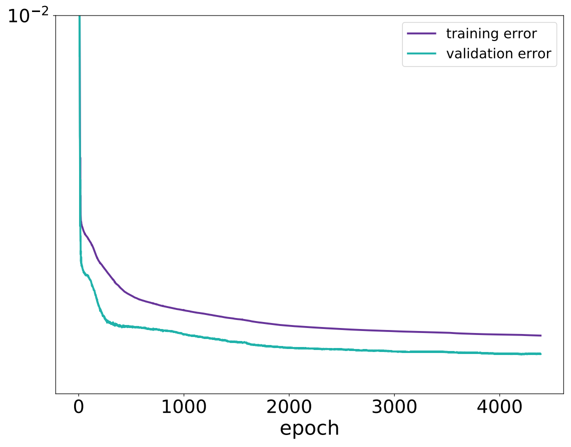
Figure B.22.: The dz resolution as a function of the particle transverse momentum p_t (y-axis) and as a function of the true z-vertex value (x-axis)



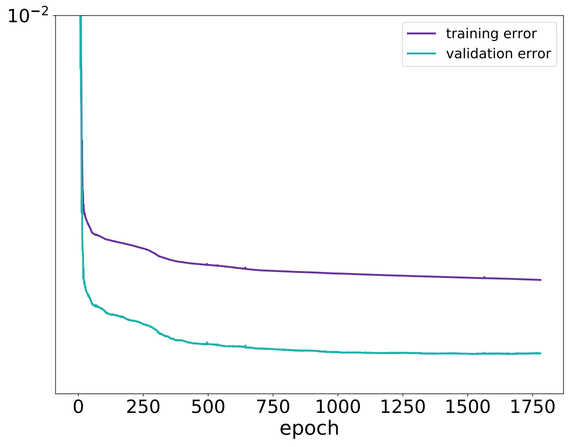
(a)



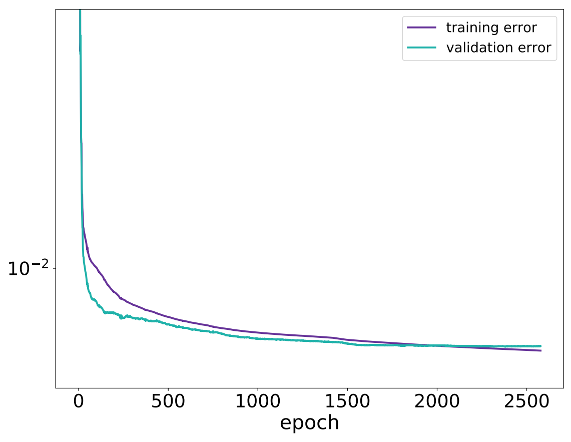
(b)



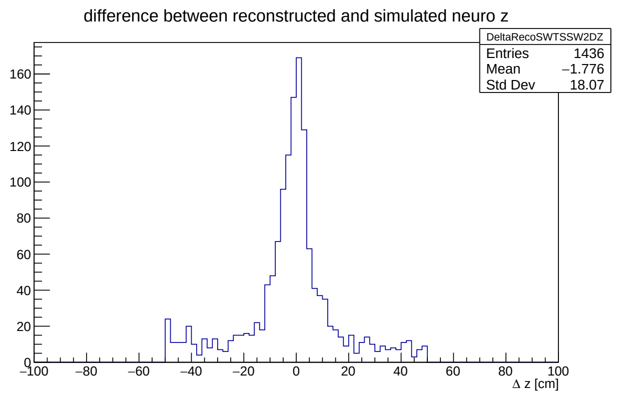
(c)



(d)

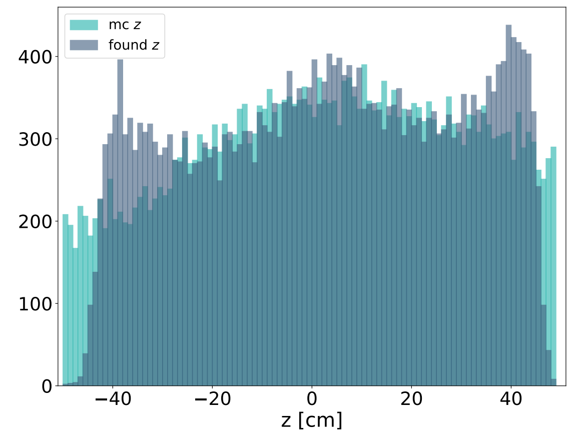


(e)

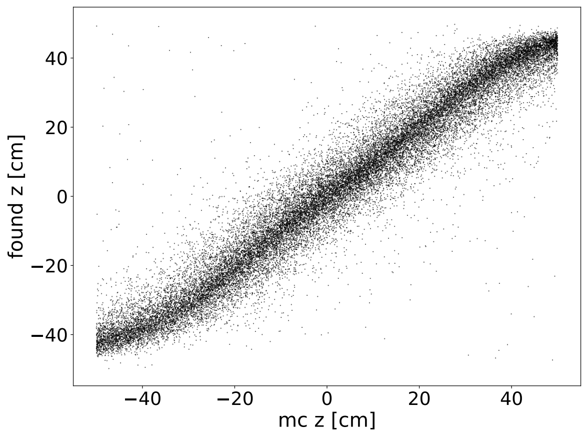


(f)

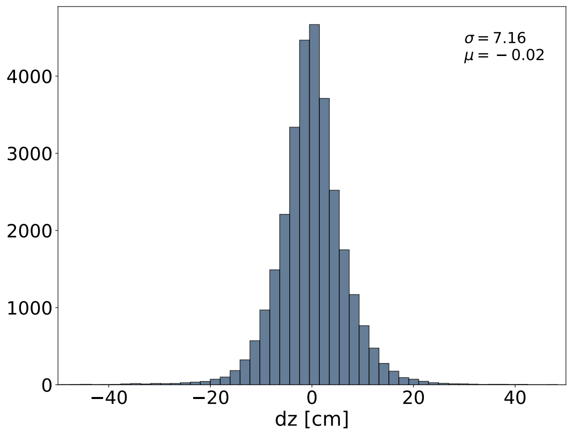
Figure B.23.: Error curves for each expert network in the ETF Phase 2 Threshold 3 network training: (a) Expert 0 (full hits); (b) Expert 1 (SL7 missing); (c) Expert 2 (SL5 missing); (d) Expert 3 (SL3 missing); (e) Expert 4 (SL1 missing), details of the training methods can be found in Chapter 4; (f) network tested with real data from early Phase 3



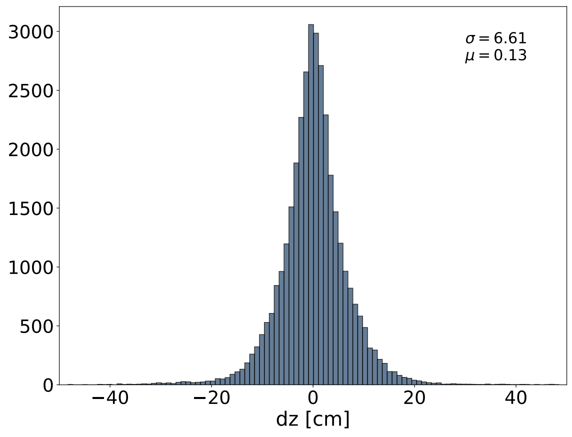
(a)



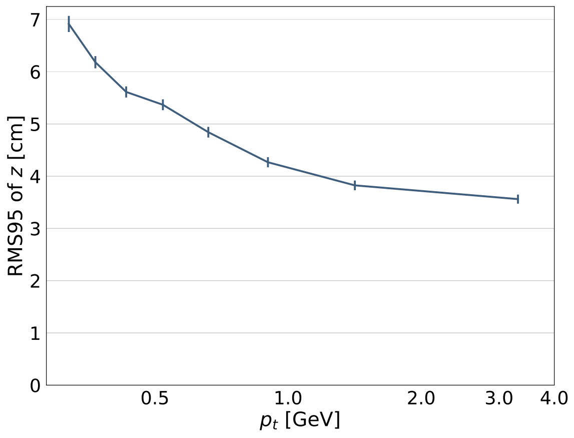
(b)



(c)



(d)



(e)

Figure B.24.: Distribution & Resolution plots for ETF Phase 2 Threshold 3 network: (a) Distribution of true z and found z-vertex values; (b) Scatter plot of the true z and found z-vertex values; (c) dz resolution (dz = mc z - found z), all (a)–(c) tested with particles generated along z=±50 cm, (d)–(e) tested with particles generated at IP (z=0); (d) dz resolution at IP; (e) p_t-dependent resolution at IP

B.0.8. ETF Phase 3 Threshold 0 network

thetaParams	[10, 170]	**zVertexParams**	[-50, 50]
Background	Phase 3	**et_option**	ETF (threshold=0)
nTrainPrepare	1000	**nValid**	1000
nTrainMax	nWeights*10	**nTrainMin**	nWeights*10
nTest	5000	**nHidden**	81
Training target	MC Particles		

Table B.9.: Training parameters for ETF Phase 3 Threshold 0 network

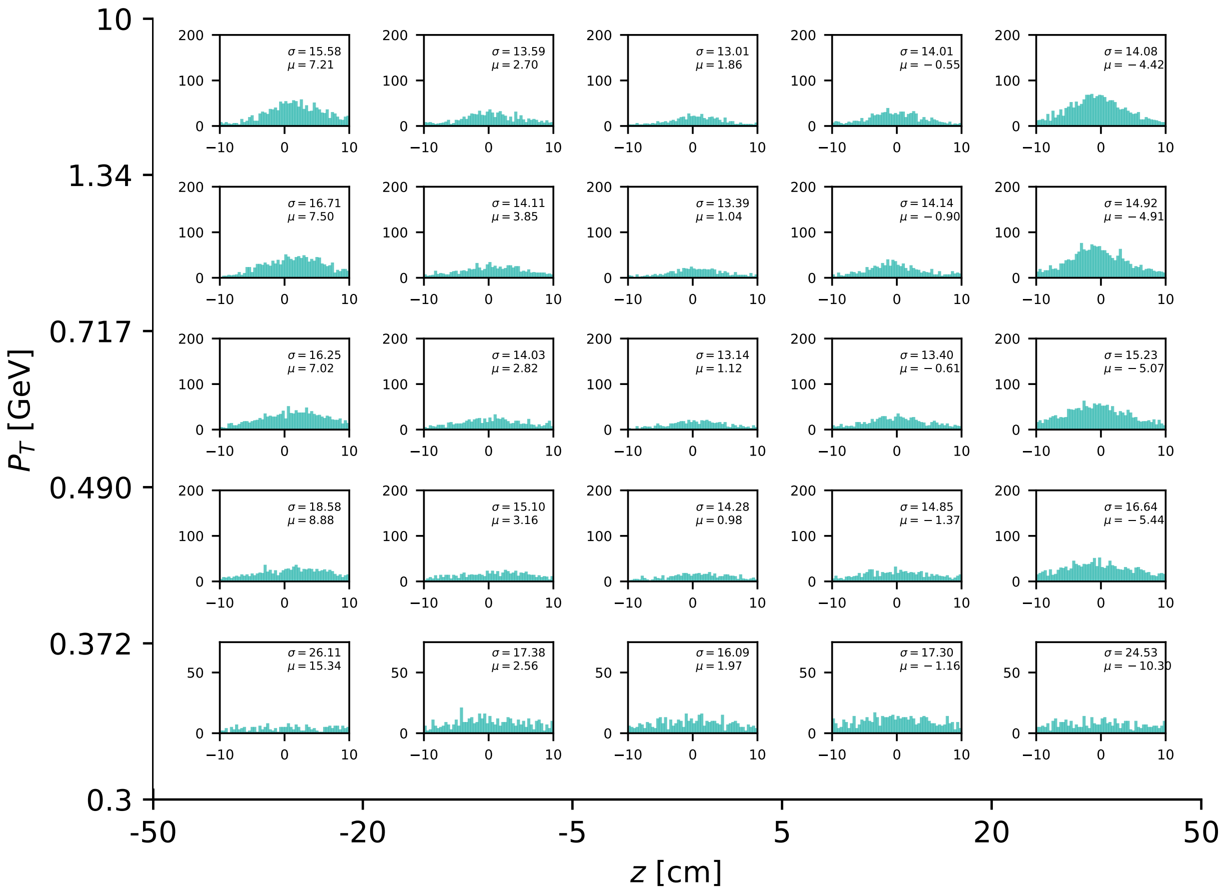


Figure B.25.: The dz resolution as a function of the particle transverse momentum p_t (y-axis) and as a function of the true z-vertex value (x-axis)

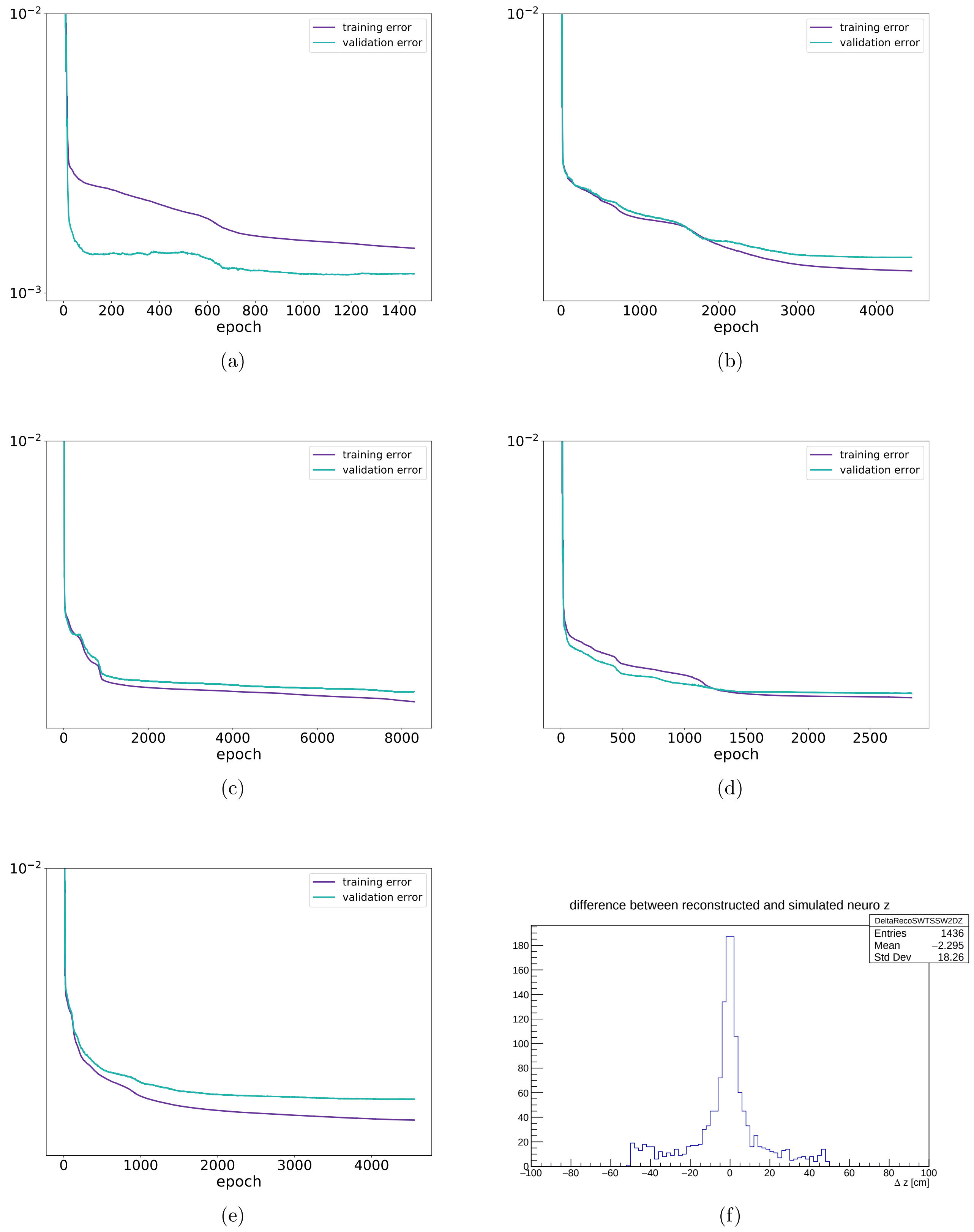
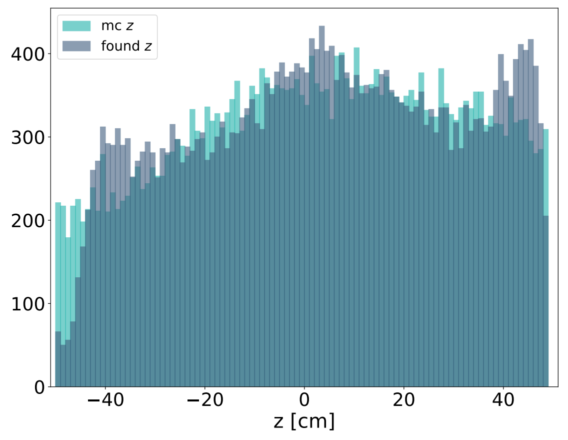
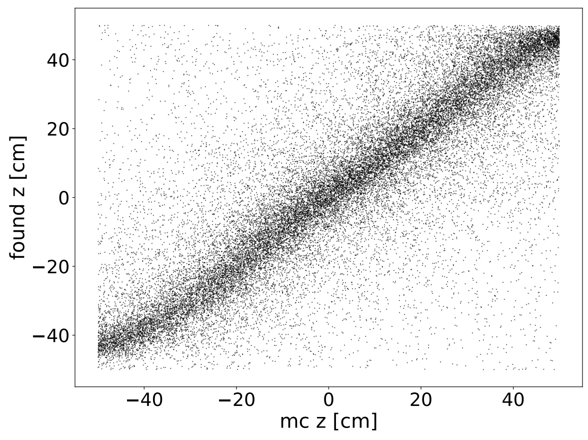


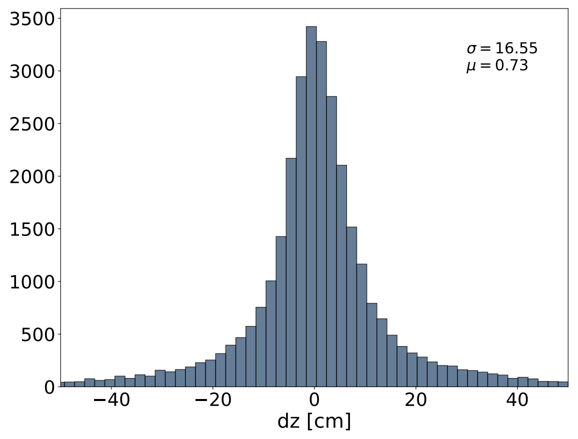
Figure B.26.: Error curves for each expert network in the ETF Phase 3 Threshold 0 network training: (a) Expert 0 (full hits); (b) Expert 1 (SL7 missing); (c) Expert 2 (SL5 missing); (d) Expert 3 (SL3 missing); (e) Expert 4 (SL1 missing), details of the training methods can be found in Chapter 4; (f) network tested with real data from early Phase 3



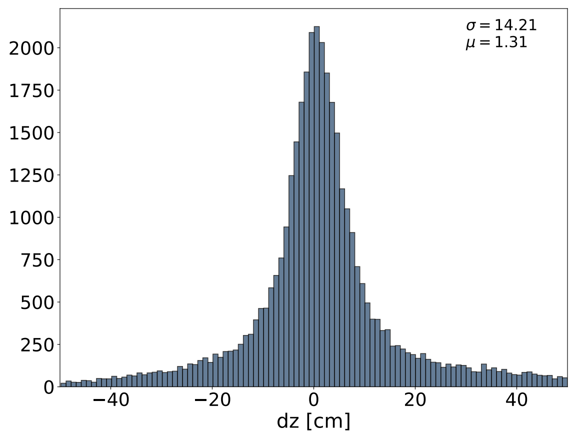
(a)



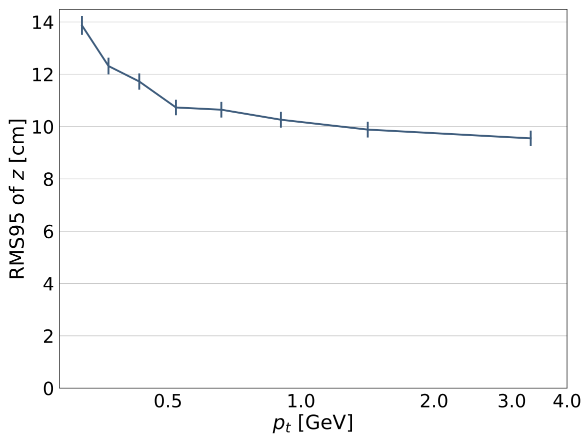
(b)



(c)



(d)



(e)

Figure B.27.: Distribution & Resolution plots for ETF Phase 3 Threshold 0 network: (a) Distribution of true z and found z-vertex values; (b) Scatter plot of the true z and found z-vertex values; (c) dz resolution (dz = mc z - found z), all (a)–(c) tested with particles generated along z=±50 cm, (d)–(e) tested with particles generated at IP (z=0); (d) dz resolution at IP; (e) p_t-dependent resolution at IP

B.0.9. ETF Phase 3 Threshold 1 network

thetaParams	[10, 170]	**zVertexParams**	[-50, 50]
Background	Phase 3	**et_option**	ETF (threshold=1)
nTrainPrepare	1000	**nValid**	1000
nTrainMax	nWeights*10	**nTrainMin**	nWeights*10
nTest	5000	**nHidden**	81
Training target	MC Particles		

Table B.10.: Training parameters for ETF Phase 3 Threshold 1 network

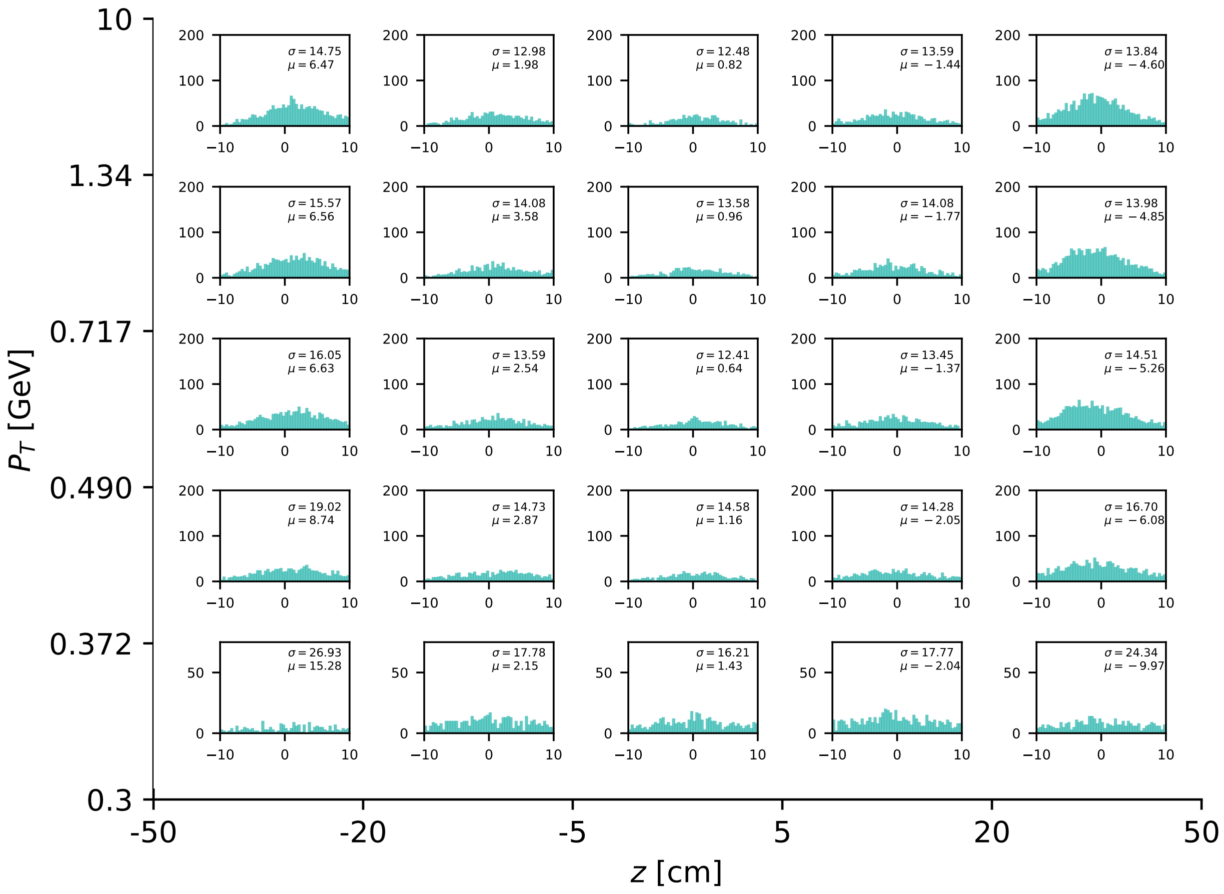
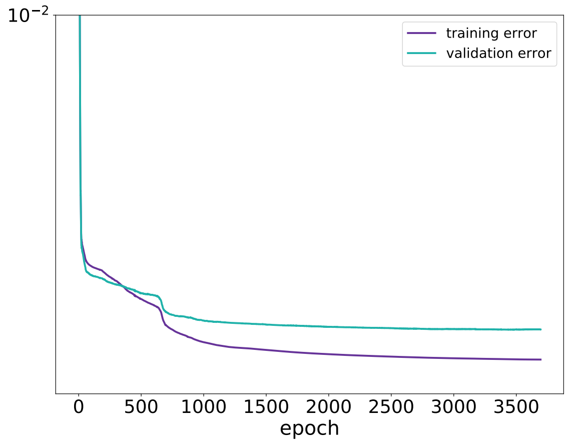
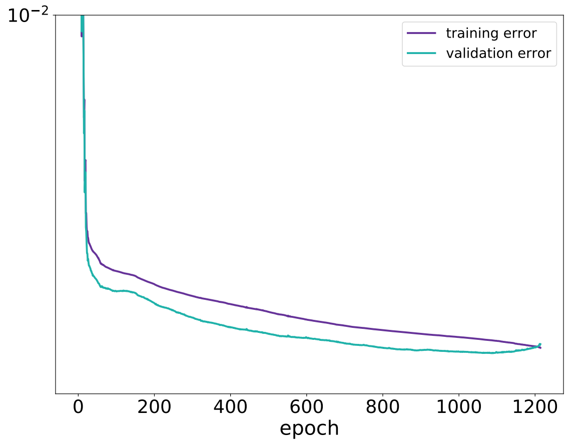


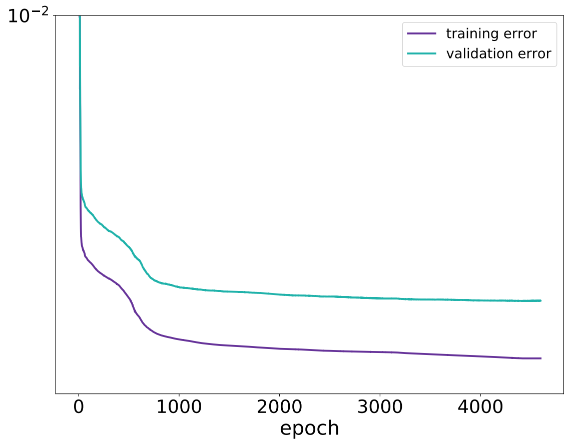
Figure B.28.: The dz resolution as a function of the particle transverse momentum p_t (y-axis) and as a function of the true z-vertex value (x-axis)



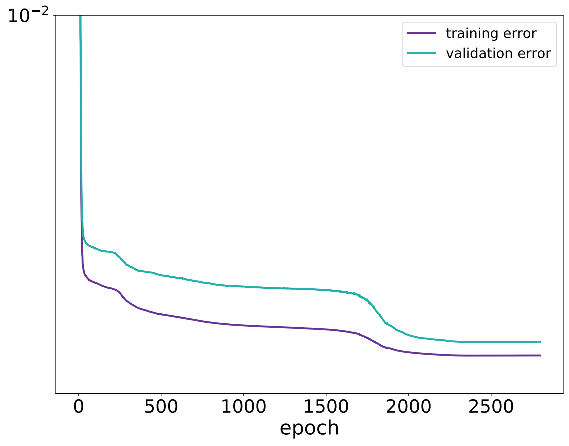
(a)



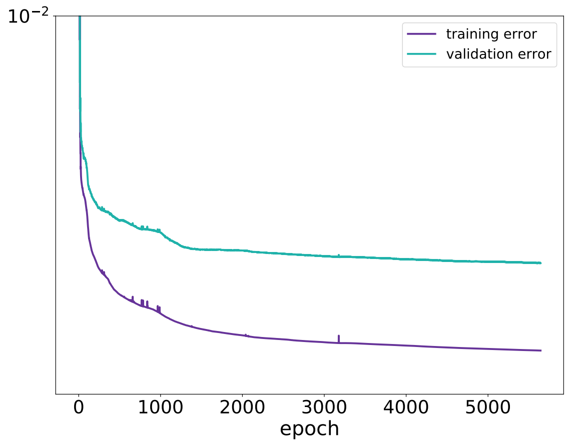
(b)



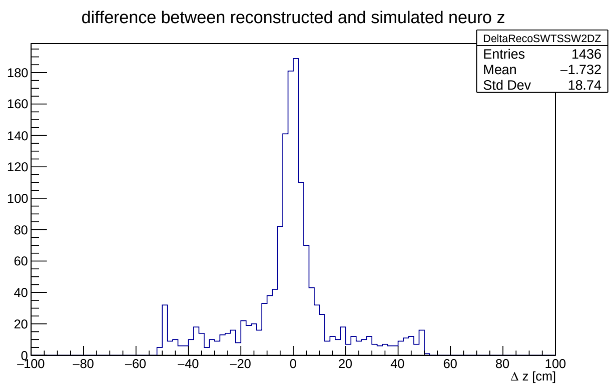
(c)



(d)

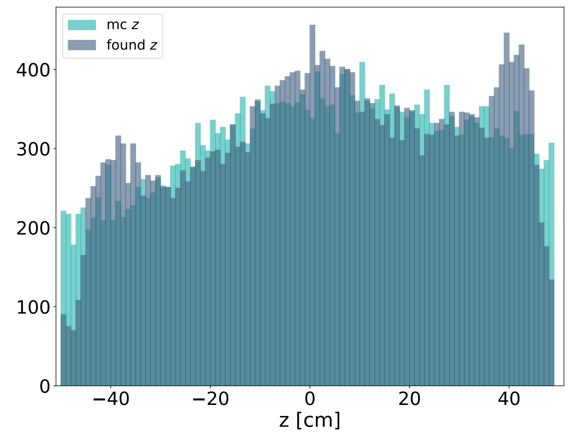


(e)

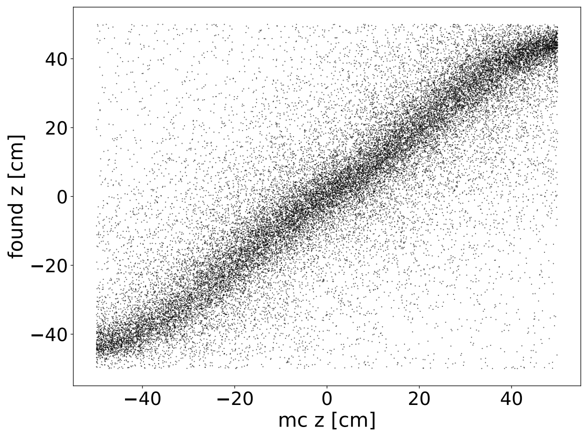


(f)

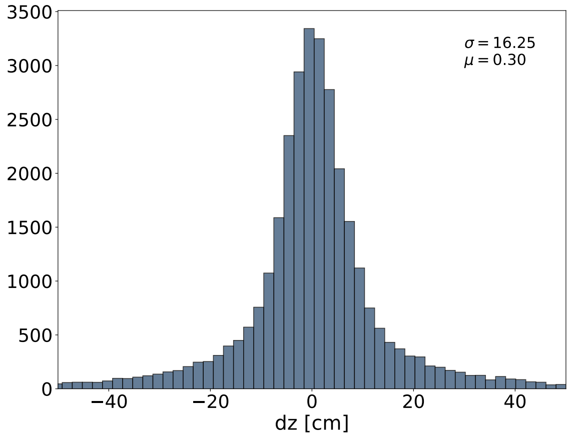
Figure B.29.: Error curves for each expert network in the ETF Phase 3 Threshold 1 network training: (a) Expert 0 (full hits); (b) Expert 1 (SL7 missing); (c) Expert 2 (SL5 missing); (d) Expert 3 (SL3 missing); (e) Expert 4 (SL1 missing), details of the training methods can be found in Chapter 4; (f) network tested with real data from early Phase 3



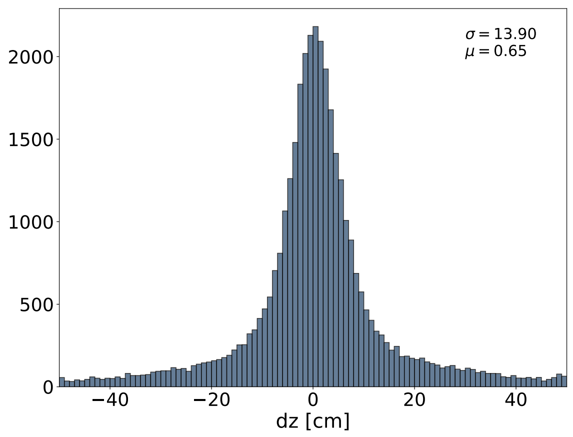
(a)



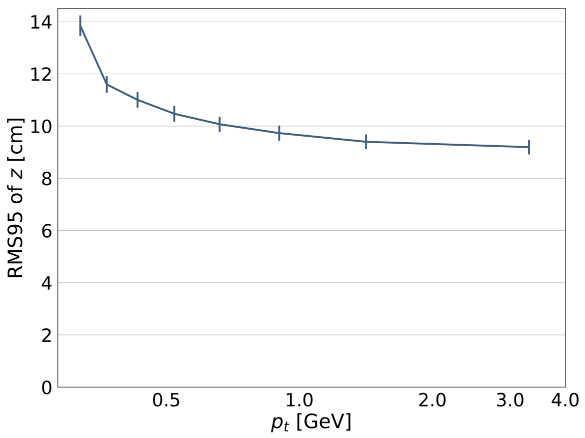
(b)



(c)



(d)



(e)

Figure B.30.: Distribution & Resolution plots for ETF Phase 3 Threshold 1 network: (a) Distribution of true z and found z-vertex values; (b) Scatter plot of the true z and found z-vertex values; (c) dz resolution (dz = mc z - found z), all (a)–(c) tested with particles generated along z=±50 cm, (d)–(e) tested with particles generated at IP (z=0); (d) dz resolution at IP; (e) p_t-dependent resolution at IP

B.0.10. ETF Phase 3 Threshold 2 network

thetaParams	[10, 170]	**zVertexParams**	[-50, 50]
Background	Phase 3	**et_option**	ETF (threshold=2)
nTrainPrepare	1000	**nValid**	1000
nTrainMax	nWeights*10	**nTrainMin**	nWeights*10
nTest	5000	**nHidden**	81
Training target	MC Particles		

Table B.11.: Training parameters for ETF Phase 3 Threshold 2 network

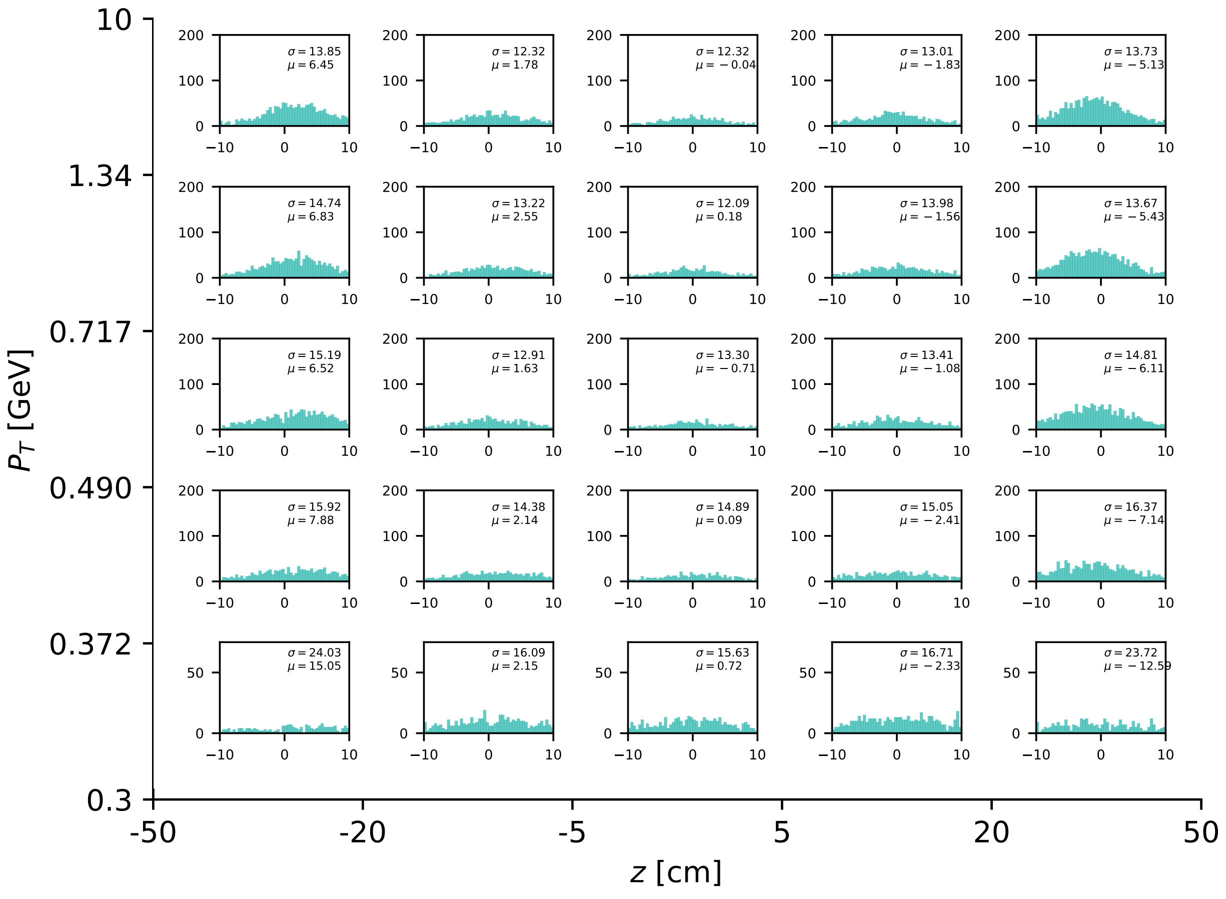
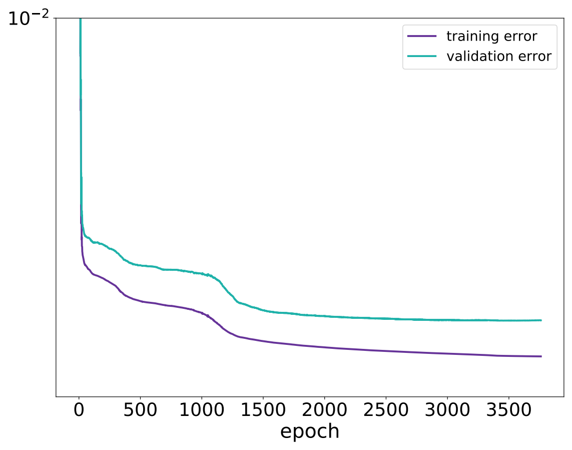
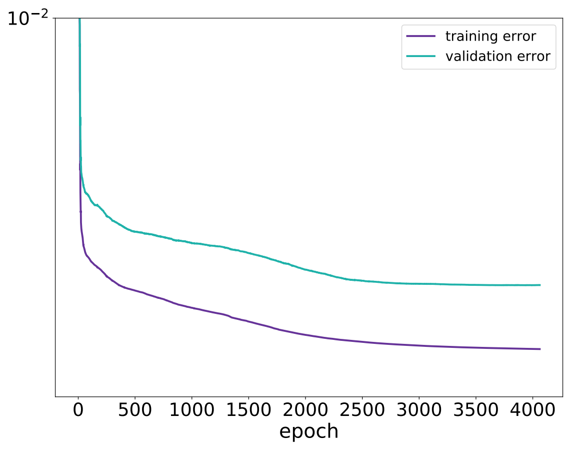


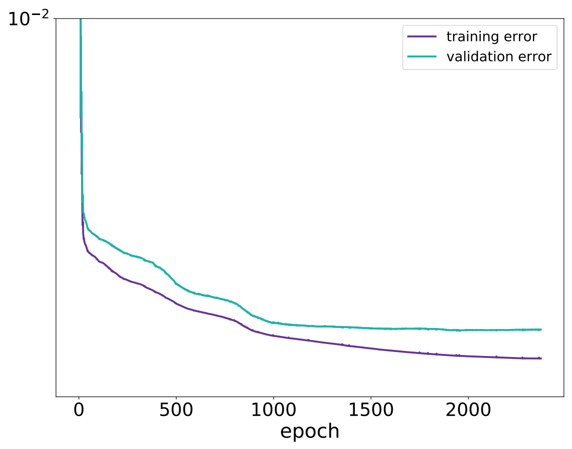
Figure B.31.: The dz resolution as a function of the particle transverse momentum p_t (y-axis) and as a function of the true z-vertex value (x-axis)



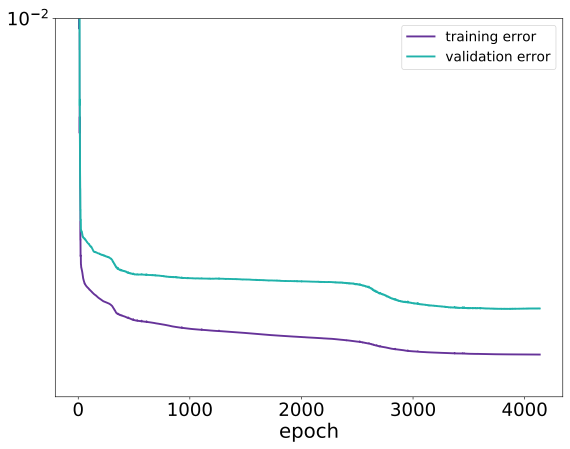
(a)



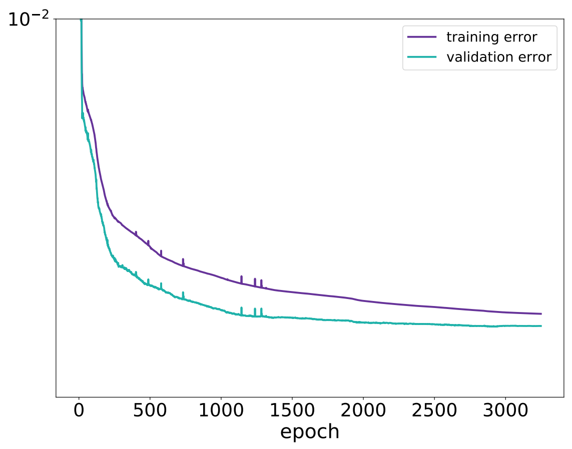
(b)



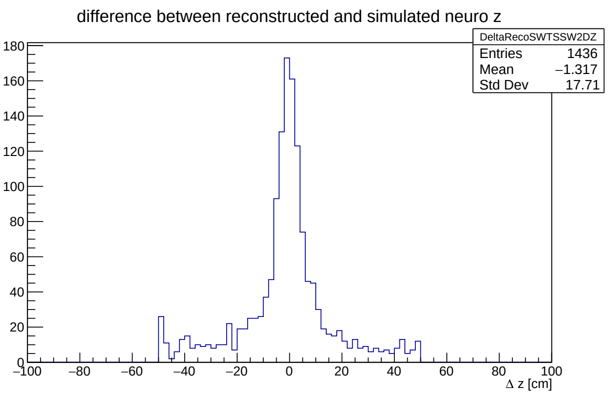
(c)



(d)

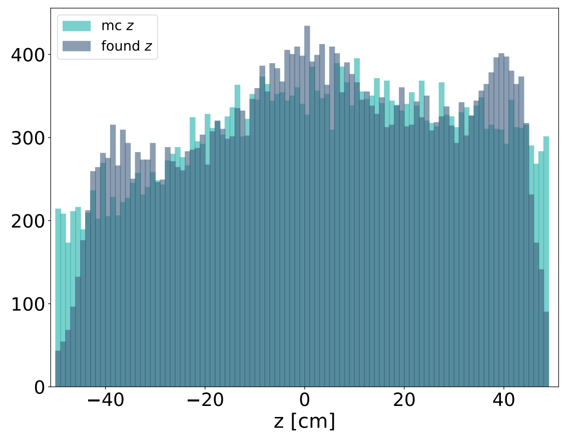


(e)

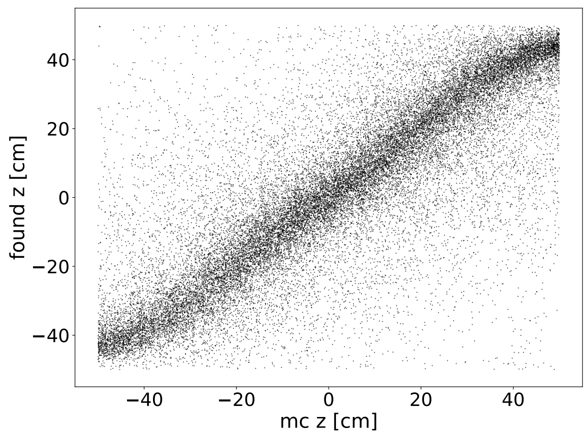


(f)

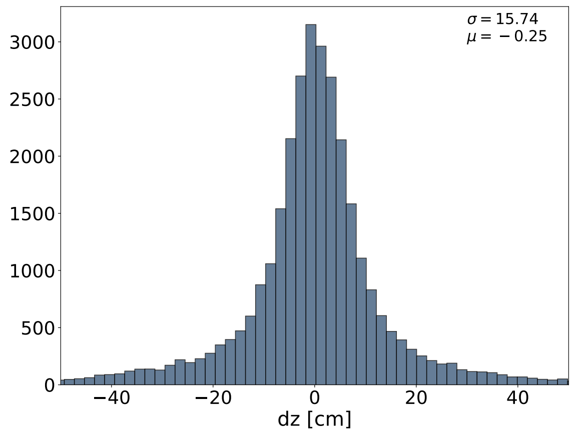
Figure B.32.: Error curves for each expert network in the ETF Phase 3 Threshold 2 network training: (a) Expert 0 (full hits); (b) Expert 1 (SL7 missing); (c) Expert 2 (SL5 missing); (d) Expert 3 (SL3 missing); (e) Expert 4 (SL1 missing), details of the training methods can be found in Chapter 4; (f) network tested with real data from early Phase 3



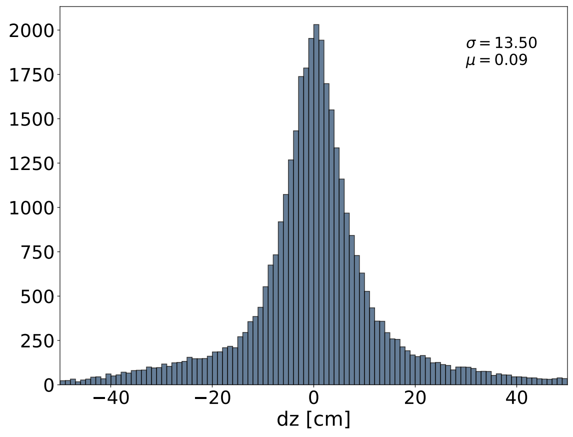
(a)



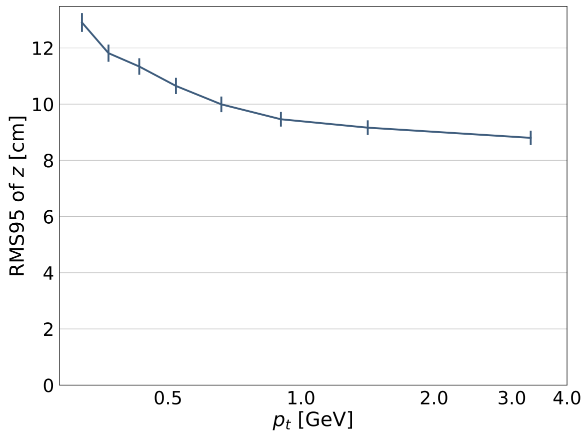
(b)



(c)



(d)



(e)

Figure B.33.: Distribution & Resolution plots for ETF Phase 3 Threshold 2 network: (a) Distribution of true z and found z-vertex values; (b) Scatter plot of the true z and found z-vertex values; (c) dz resolution (dz = mc z - found z), all (a)–(c) tested with particles generated along z=±50 cm, (d)–(e) tested with particles generated at IP (z=0); (d) dz resolution at IP; (e) p_t-dependent resolution at IP

B.0.11. ETF Phase 3 Threshold 3 network

thetaParams	[10, 170]	**zVertexParams**	[-50, 50]
Background	Phase 3	**et_option**	ETF (threshold=3)
nTrainPrepare	1000	**nValid**	1000
nTrainMax	nWeights*10	**nTrainMin**	nWeights*10
nTest	5000	**nHidden**	81
Training target	MC Particles		

Table B.12.: Training parameters for ETF Phase 3 Threshold 3 network

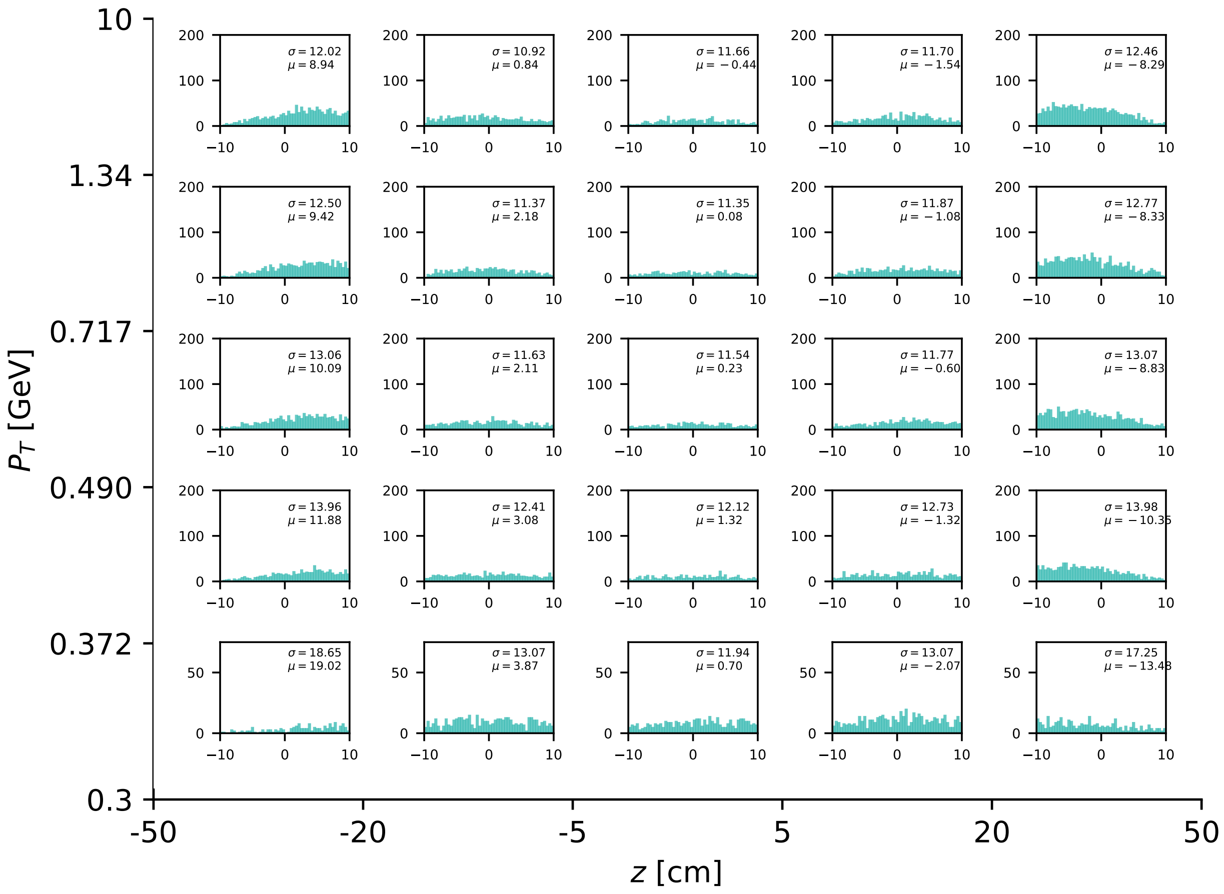


Figure B.34.: The dz resolution as a function of the particle transverse momentum p_t (y-axis) and as a function of the true z-vertex value (x-axis)

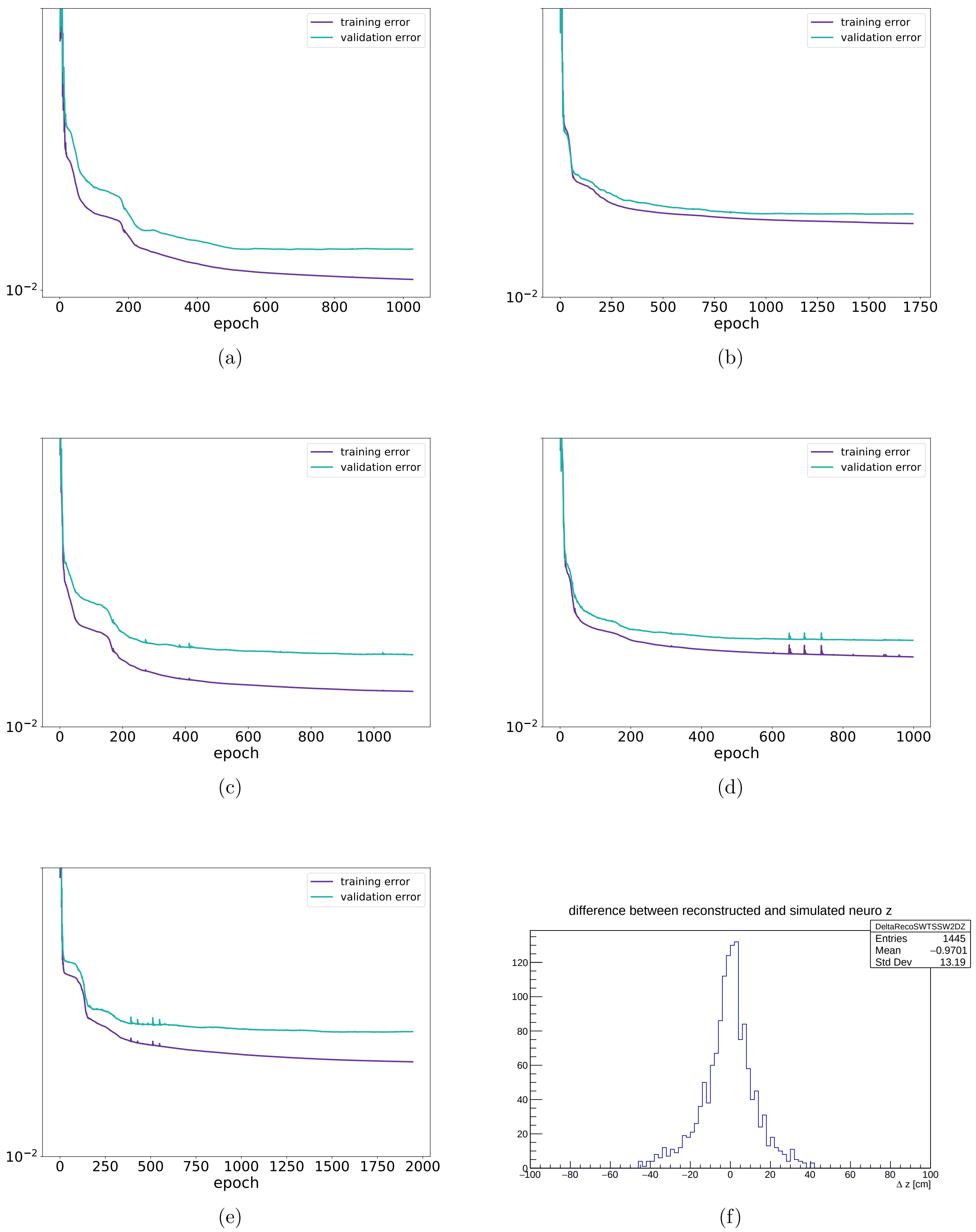
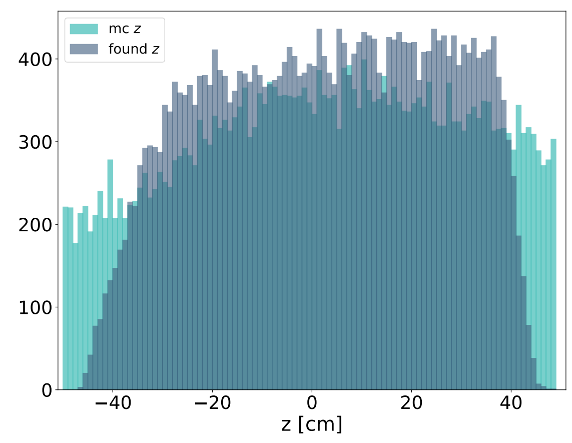
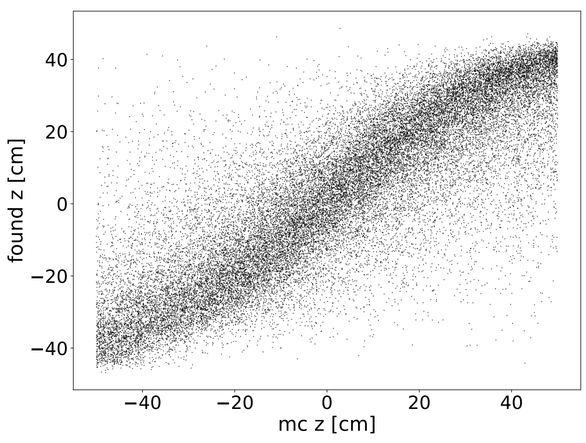


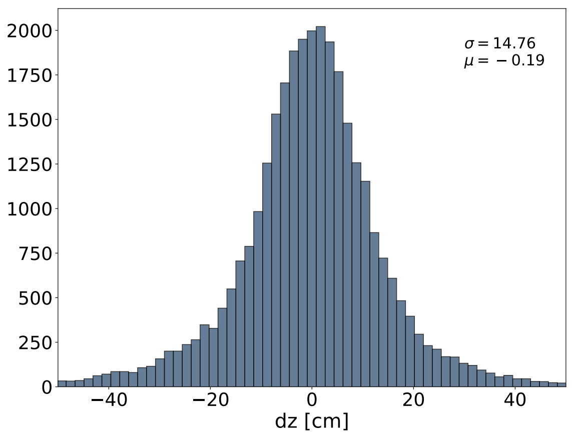
Figure B.35.: Error curves for each expert network in the ETF Phase 3 Threshold 3 network training: (a) Expert 0 (full hits); (b) Expert 1 (SL7 missing); (c) Expert 2 (SL5 missing); (d) Expert 3 (SL3 missing); (e) Expert 4 (SL1 missing), details of the training methods can be found in Chapter 4; (f) network tested with real data from early Phase 3



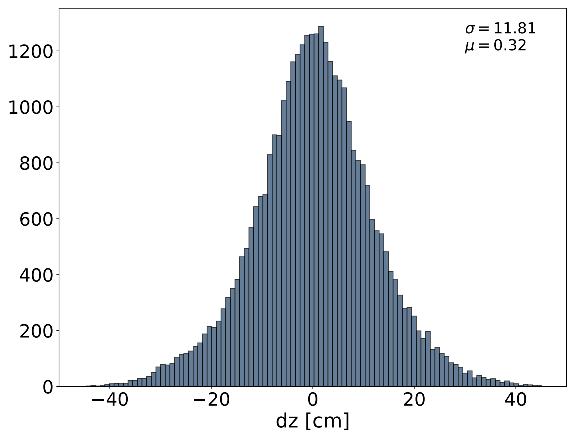
(a)



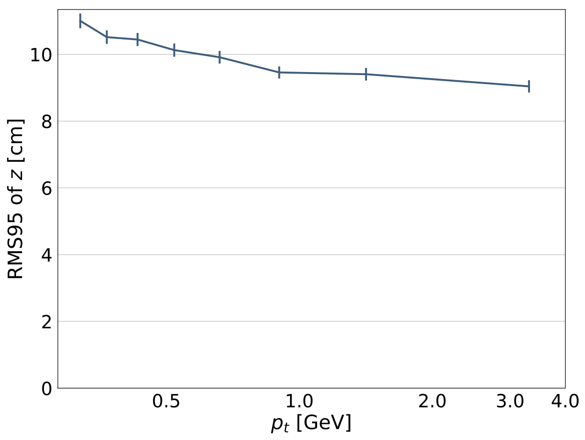
(b)



(c)



(d)



(e)

Figure B.36.: Distribution & Resolution plots for ETF Phase 3 Threshold 3 network: (a) Distribution of true z and found z-vertex values; (b) Scatter plot of the true z and found z-vertex values; (c) dz resolution (dz = mc z - found z), all (a)–(c) tested with particles generated along z=±50 cm, (d)–(e) tested with particles generated at IP (z=0); (d) dz resolution at IP; (e) p_t-dependent resolution at IP

B.0.12. ETF Phase 3 Threshold 4 network

thetaParams	[10, 170]	**zVertexParams**	[-50, 50]
Background	Phase 3	**et_option**	ETF (threshold=4)
nTrainPrepare	1000	**nValid**	1000
nTrainMax	nWeights*10	**nTrainMin**	nWeights*10
nTest	5000	**nHidden**	81
Training target	MC Particles		

Table B.13.: Training parameters for ETF Phase 3 Threshold 4 network

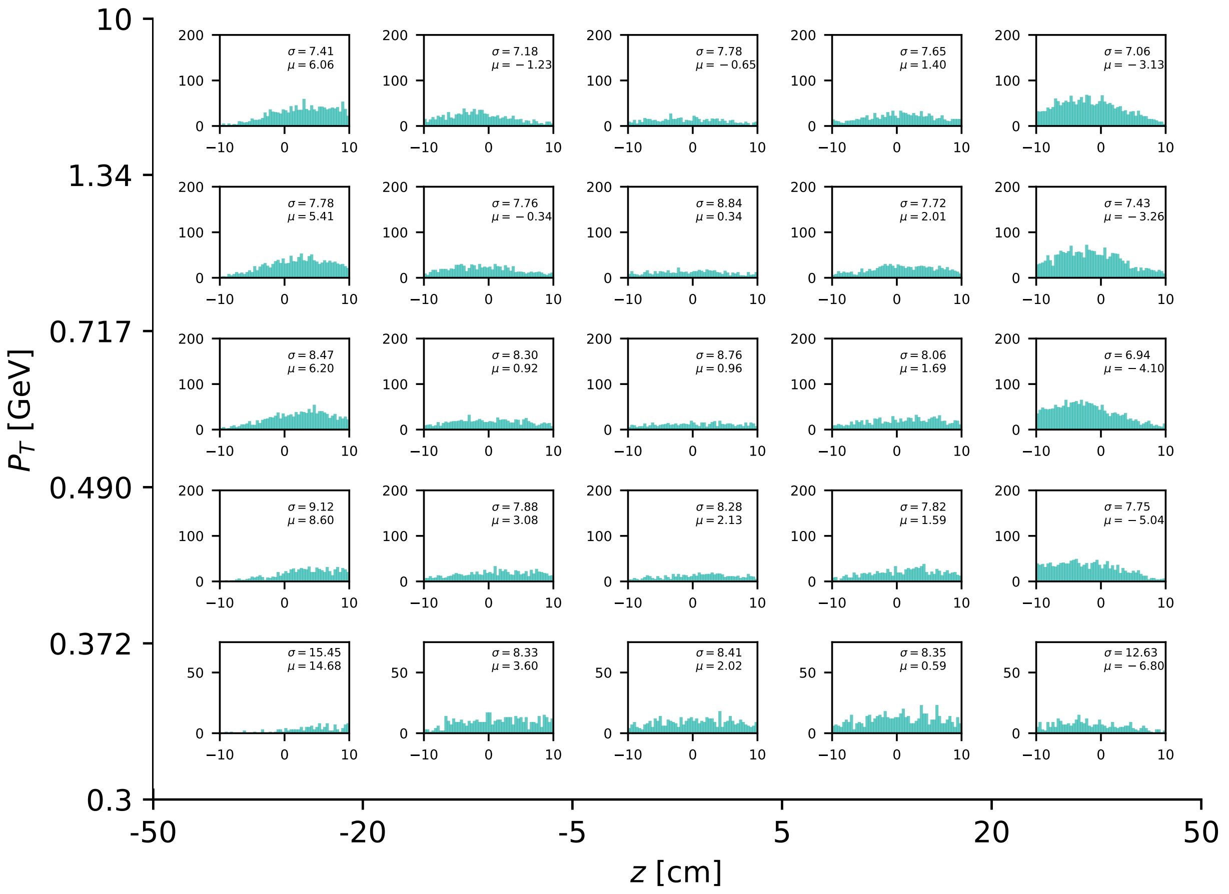
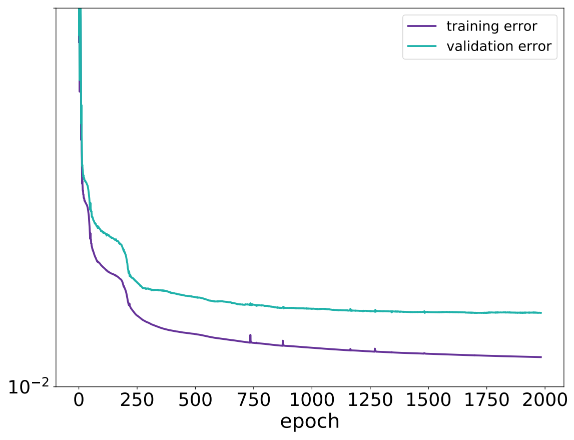
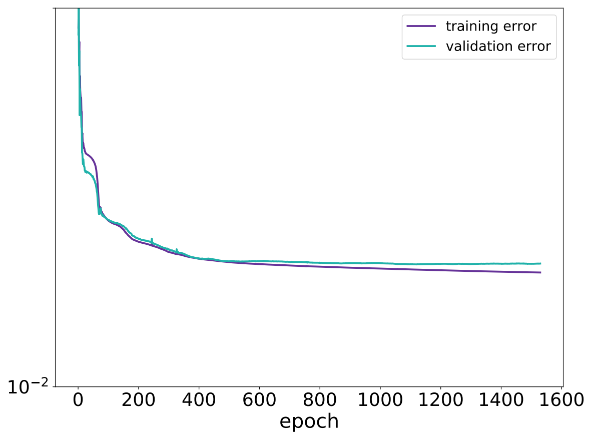


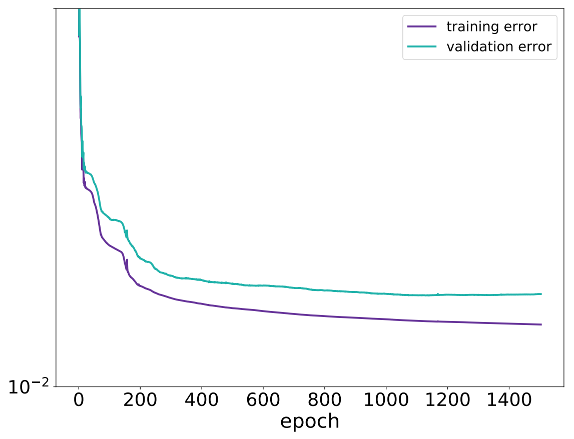
Figure B.37.: The dz resolution as a function of the particle transverse momentum p_t (y-axis) and as a function of the true z-vertex value (x-axis)



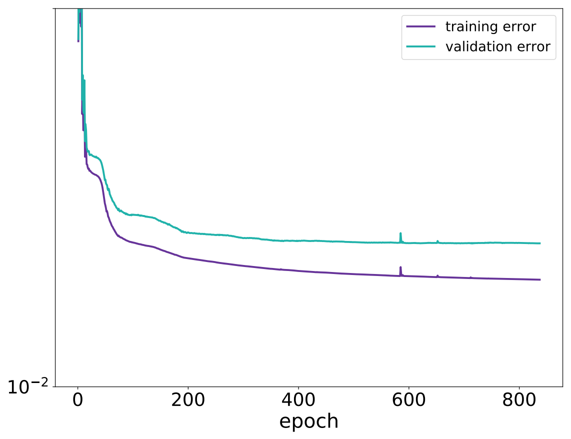
(a)



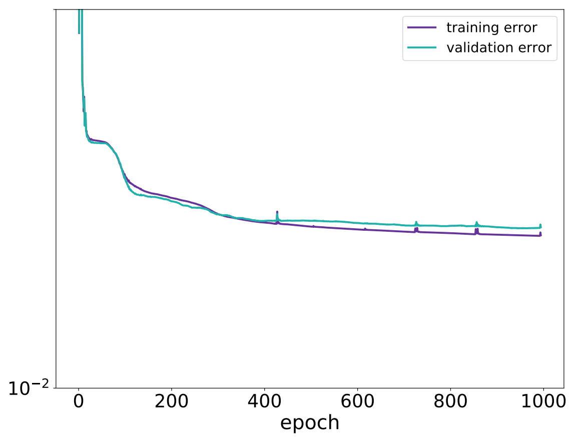
(b)



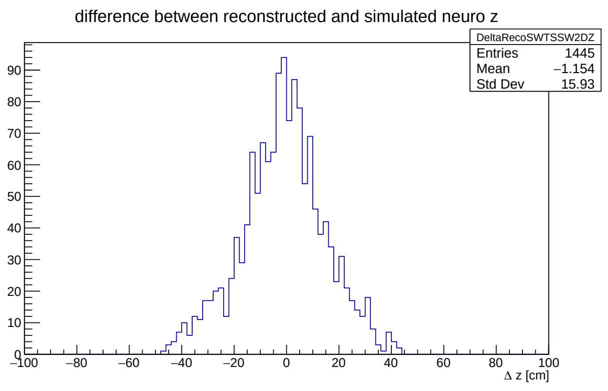
(c)



(d)

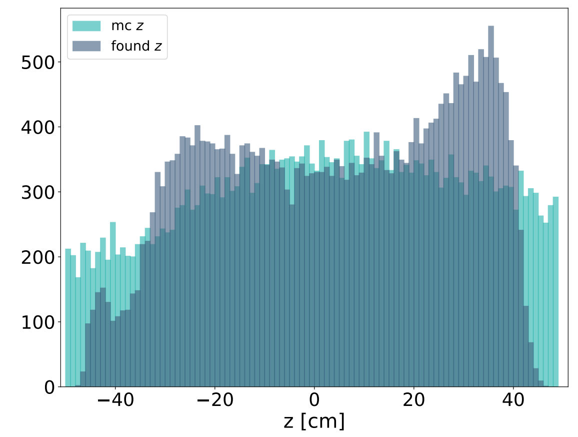


(e)

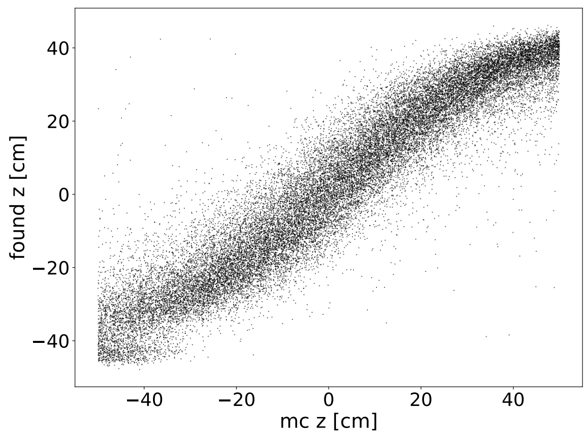


(f)

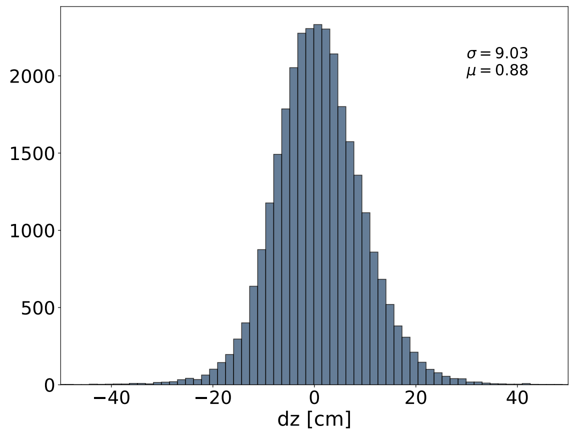
Figure B.38.: Error curves for each expert network in the ETF Phase 3 Threshold 4 network training: (a) Expert 0 (full hits); (b) Expert 1 (SL7 missing); (c) Expert 2 (SL5 missing); (d) Expert 3 (SL3 missing); (e) Expert 4 (SL1 missing), details of the training methods can be found in Chapter 4; (f) network tested with real data from early Phase 3



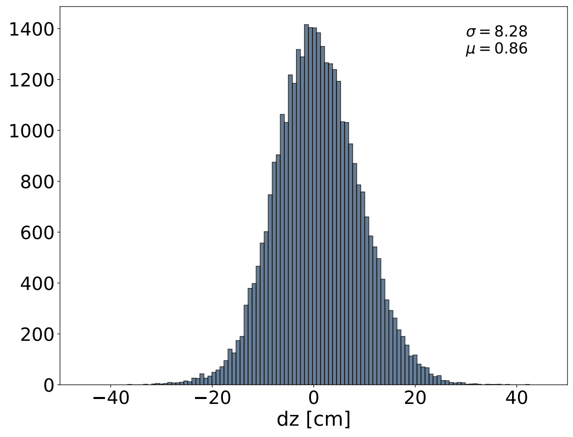
(a)



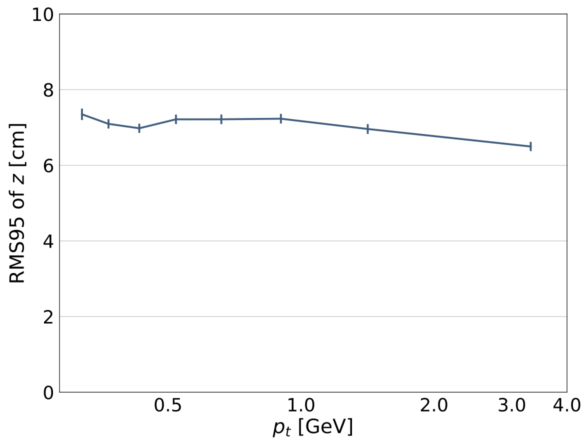
(b)



(c)



(d)



(e)

Figure B.39.: Distribution & Resolution plots for ETF Phase 3 Threshold 4 network: (a) Distribution of true z and found z-vertex values; (b) Scatter plot of the true z and found z-vertex values; (c) dz resolution (dz = mc z - found z), all (a)–(c) tested with particles generated along z=±50 cm, (d)–(e) tested with particles generated at IP (z=0); (d) dz resolution at IP; (e) p_t-dependent resolution at IP

B.0.13. ETF Phase 3 Threshold 5 network

thetaParams	[10, 170]	**zVertexParams**	[-50, 50]
Background	Phase 3	**et_option**	ETF (threshold=5)
nTrainPrepare	1000	**nValid**	1000
nTrainMax	nWeights*10	**nTrainMin**	nWeights*10
nTest	5000	**nHidden**	81
Training target	MC Particles		

Table B.14.: Training parameters for ETF Phase 3 Threshold 5 network

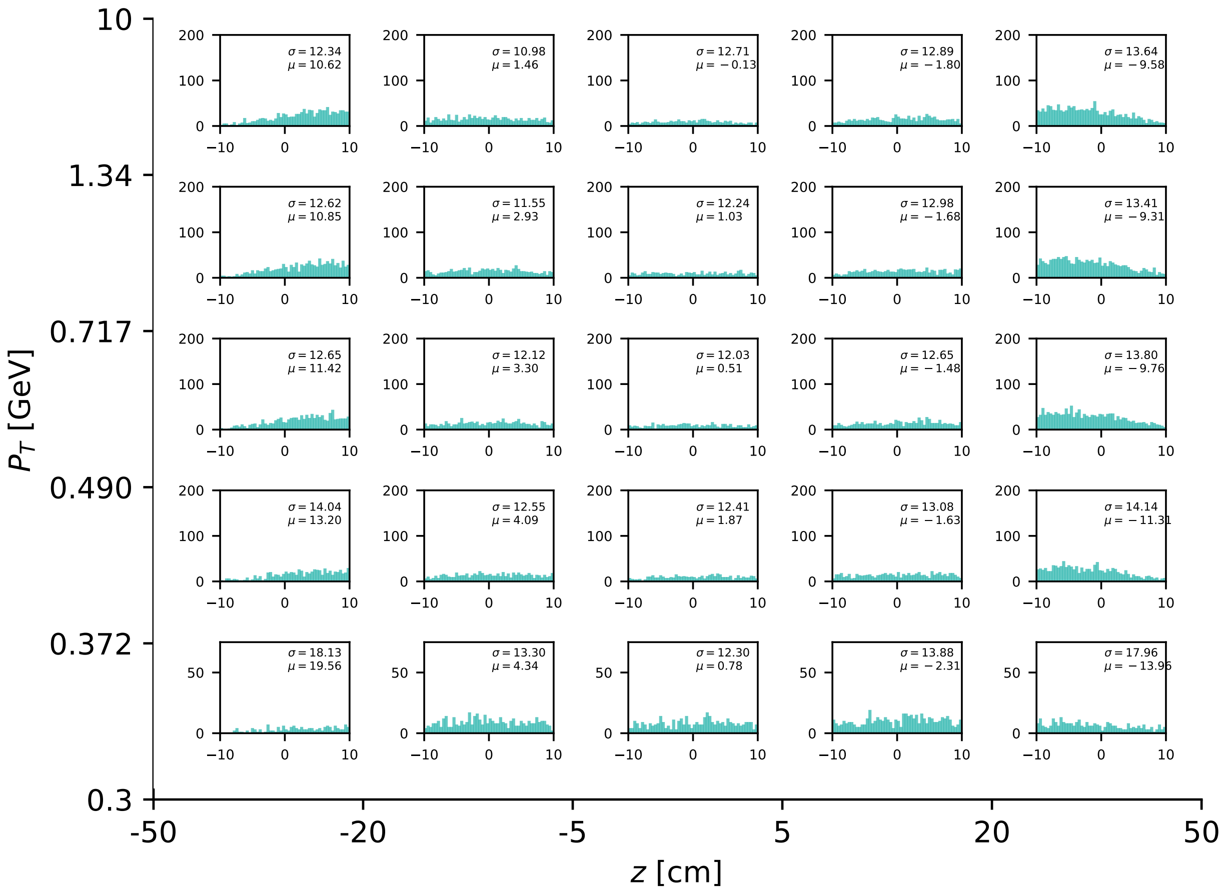
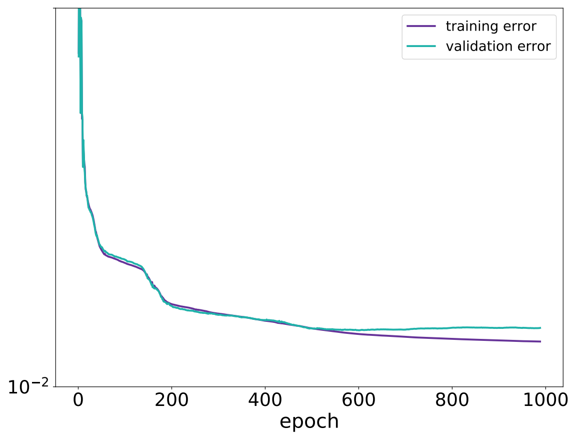
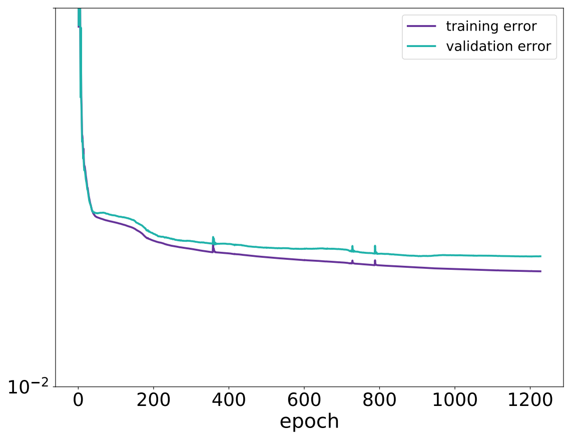


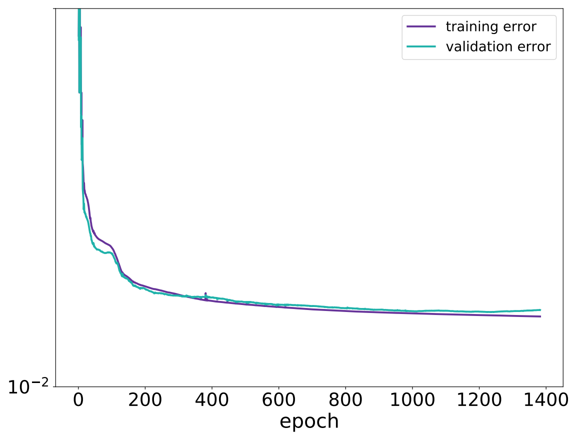
Figure B.40.: The dz resolution as a function of the particle transverse momentum p_t (y-axis) and as a function of the true z-vertex value (x-axis)



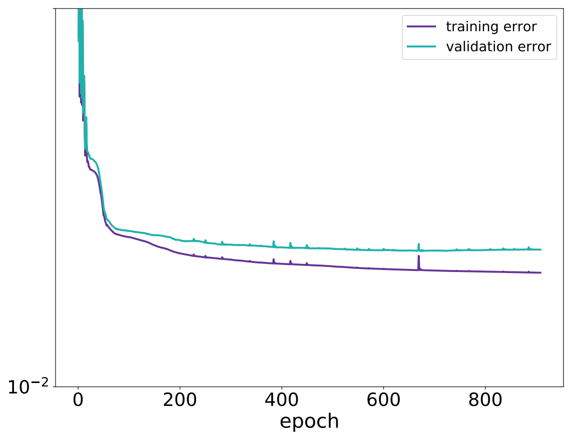
(a)



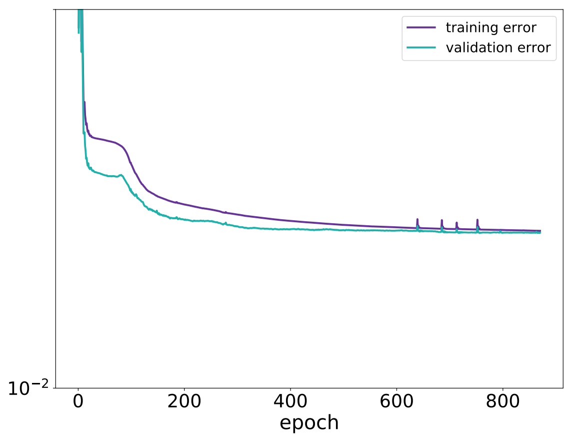
(b)



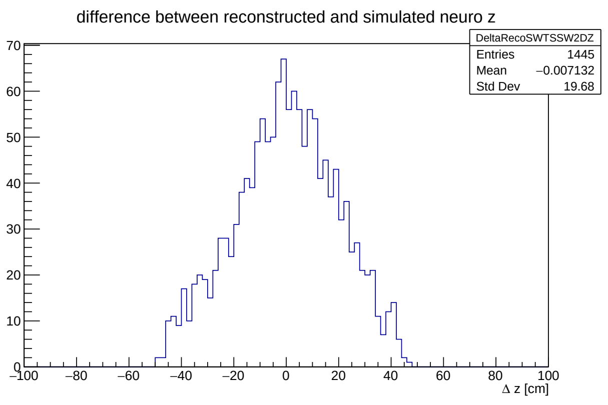
(c)



(d)

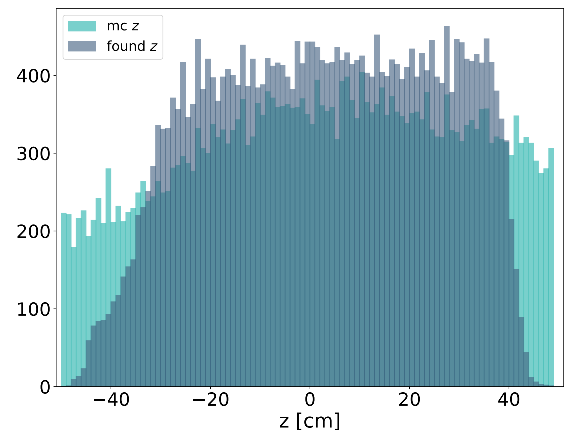


(e)

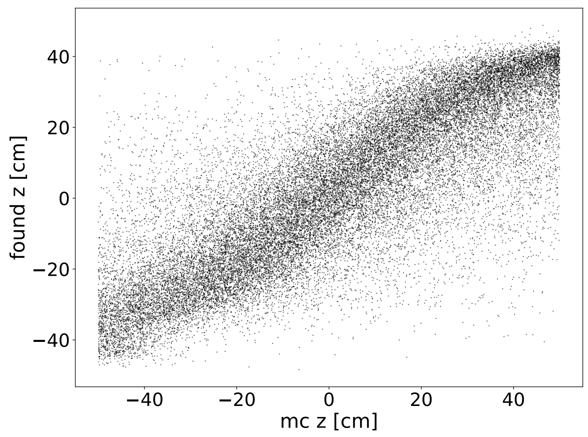


(f)

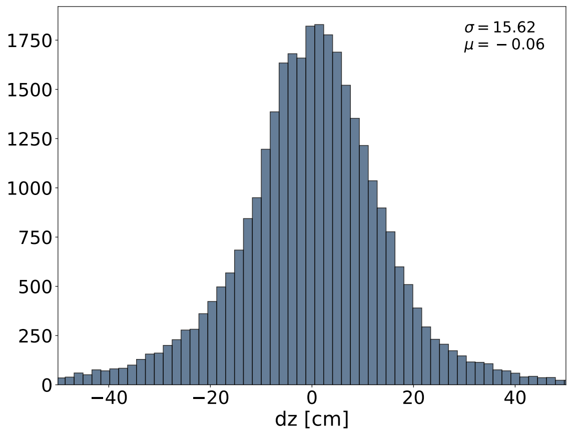
Figure B.41.: Error curves for each expert network in the ETF Phase 3 Threshold 5 network training: (a) Expert 0 (full hits); (b) Expert 1 (SL7 missing); (c) Expert 2 (SL5 missing); (d) Expert 3 (SL3 missing); (e) Expert 4 (SL1 missing), details of the training methods can be found in Chapter 4; (f) network tested with real data from early Phase 3



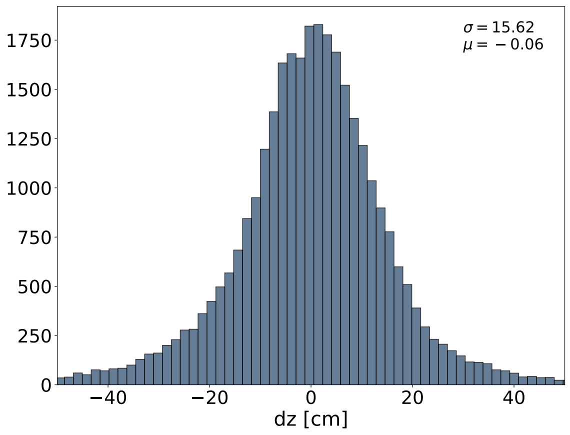
(a)



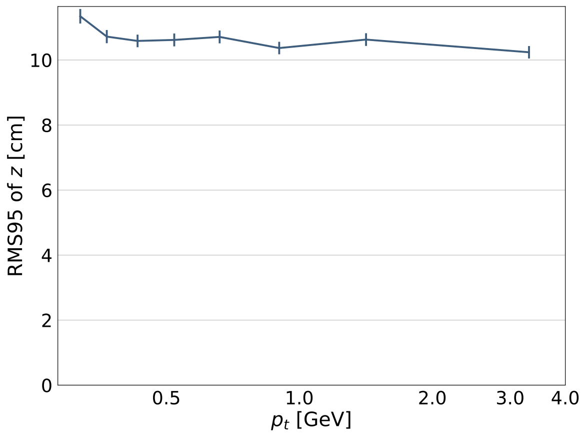
(b)



(c)



(d)



(e)

Figure B.42.: Distribution & Resolution plots for ETF Phase 3 Threshold 5 network: (a) Distribution of true z and found z-vertex values; (b) Scatter plot of the true z and found z-vertex values; (c) dz resolution (dz = mc z - found z), all (a)–(c) tested with particles generated along z=±50 cm, (d)–(e) tested with particles generated at IP (z=0); (d) dz resolution at IP; (e) p_t-dependent resolution at IP; (f) network tested with real data from early Phase 3

B.0.14. No Drift time

thetaParams	[10, 170]	**zVertexParams**	[-50, 50]
Background	Phase 2	**et_option**	-
nTrainPrepare	1000	**nValid**	1000
nTrainMax	nWeights*10	**nTrainMin**	nWeights*10
nTest	5000	**nHidden**	81
Training target	MC Particles		

Table B.15.: Training parameters for No Drift time network

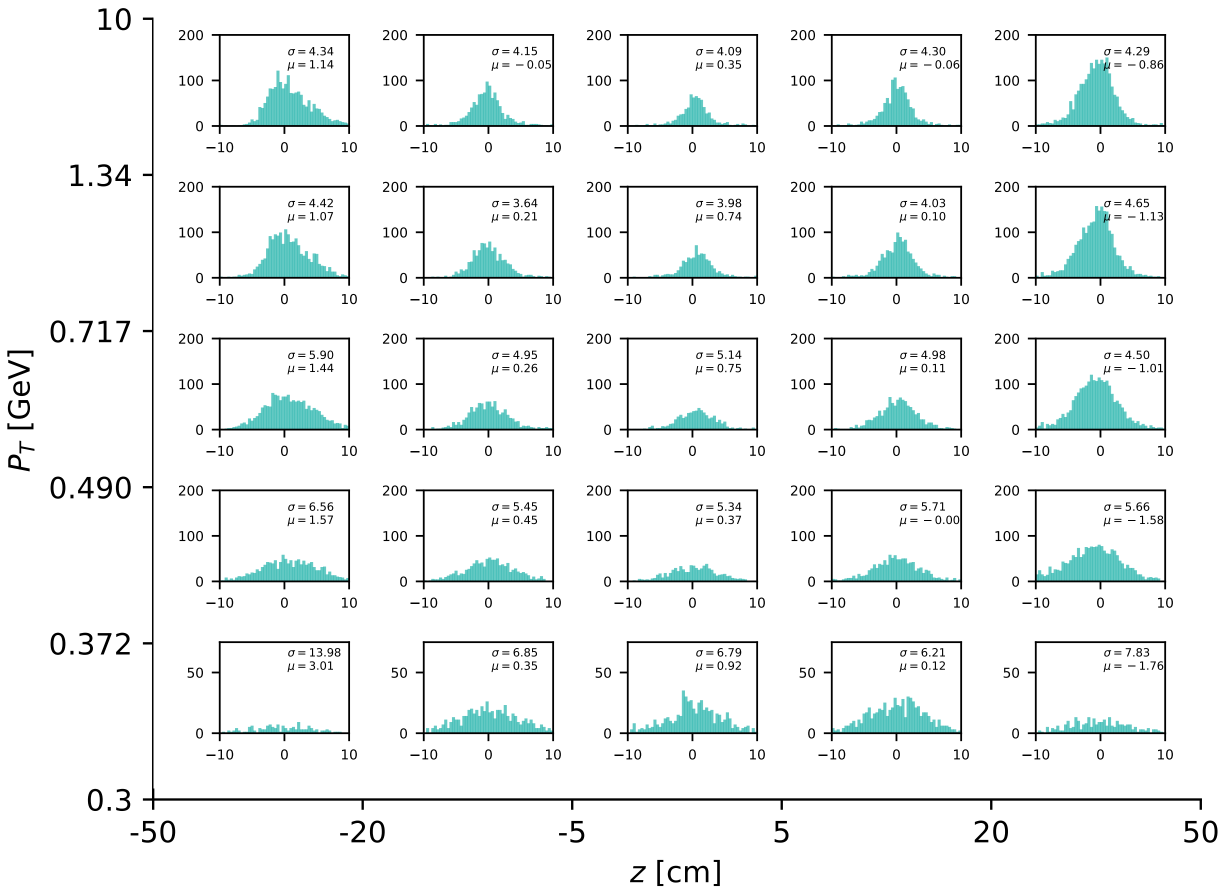
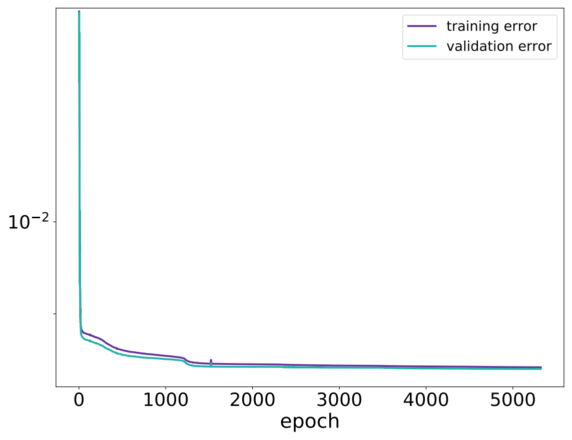
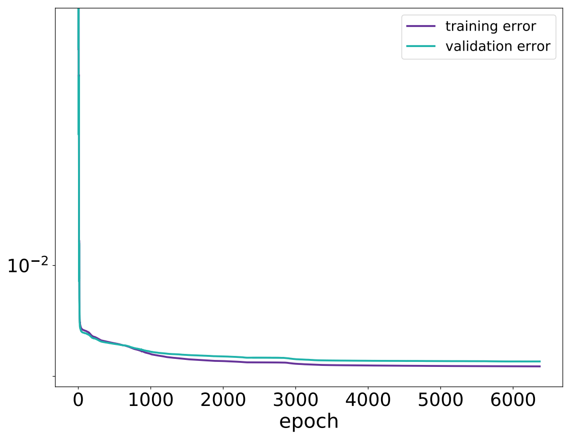


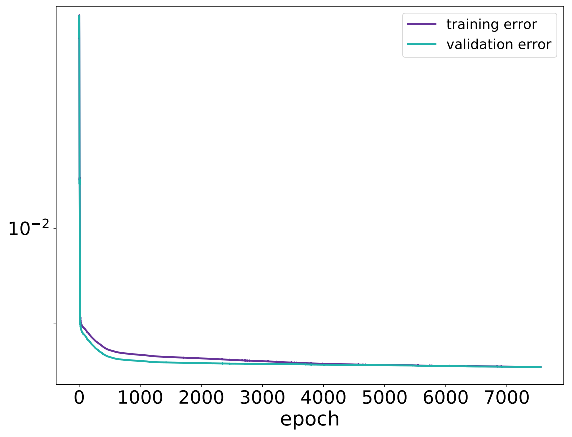
Figure B.43.: The dz resolution as a function of the particle transverse momentum p_t (y-axis) and as a function of the true z-vertex value (x-axis)



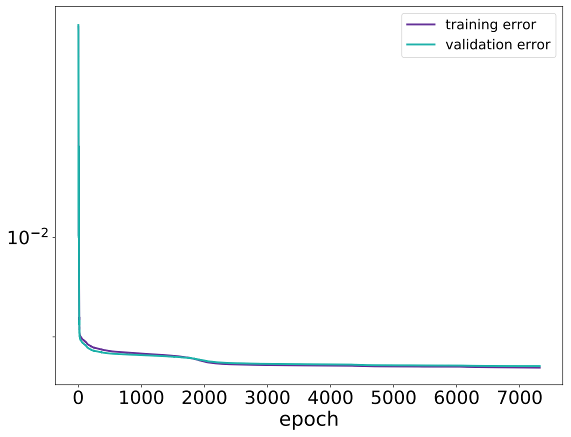
(a)



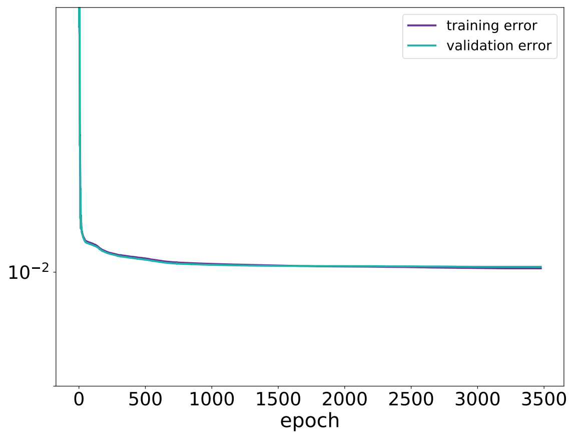
(b)



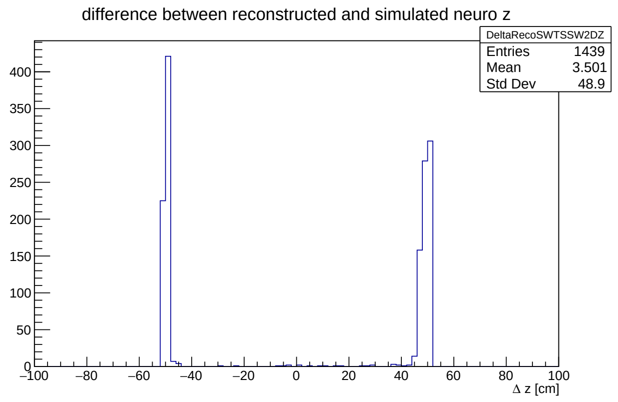
(c)



(d)

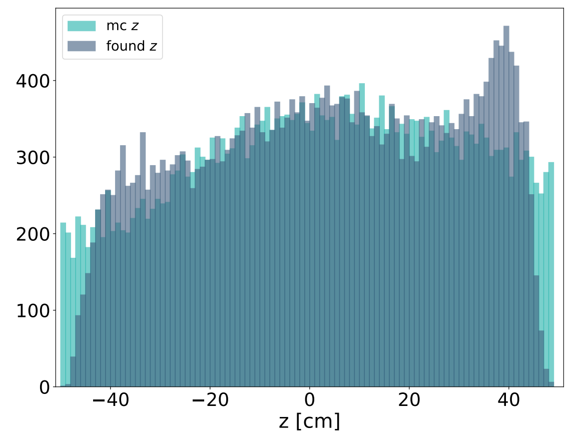


(e)

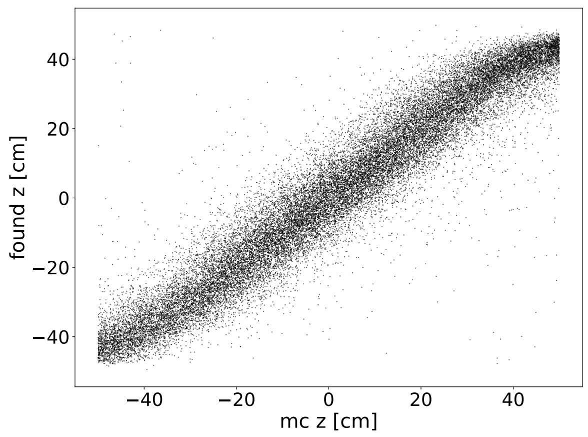


(f)

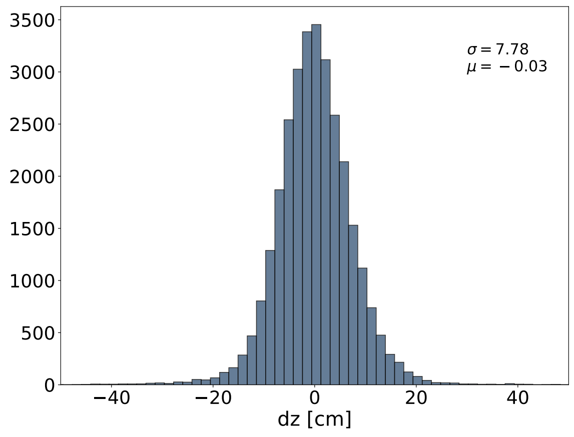
Figure B.44.: Error curves for each expert network in the No Drift time network training: (a) Expert 0 (full hits); (b) Expert 1 (SL7 missing); (c) Expert 2 (SL5 missing); (d) Expert 3 (SL3 missing); (e) Expert 4 (SL1 missing), details of the training methods can be found in Chapter 4; (f) network tested with real data from early Phase 3



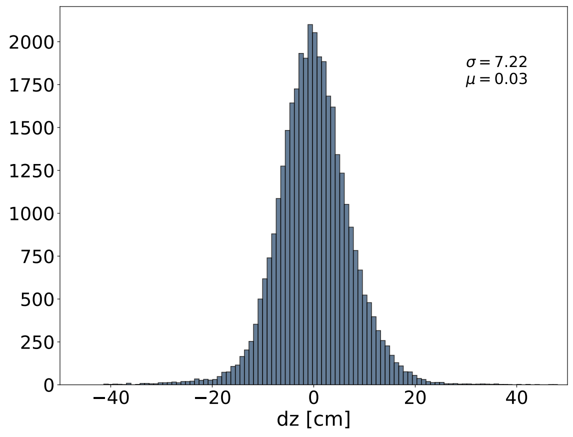
(a)



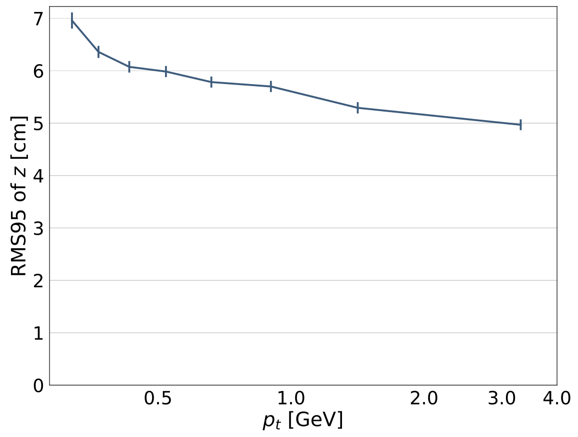
(b)



(c)



(d)



(e)

Figure B.45.: Distribution & Resolution plots for No Drift time network: (a) Distribution of true z and found z-vertex values; (b) Scatter plot of the true z and found z-vertex values; (c) dz resolution (dz = mc z - found z), all (a)–(c) tested with particles generated along z=±50 cm, (d)–(e) tested with particles generated at IP (z=0); (d) dz resolution at IP; (e) p_t-dependent resolution at IP; (f) network tested with real data from early Phase 3

B.0.15. Ignore Left/right

thetaParams	[10, 170]	**zVertexParams**	[-50, 50]
Background	Phase 2	**et_option**	fastest priority
nTrainPrepare	1000	**nValid**	1000
nTrainMax	nWeights*10	**nTrainMin**	nWeights*10
nTest	5000	**nHidden**	81
Training target	MC Particles		

Table B.16.: Training parameters for Ignore Left/right network

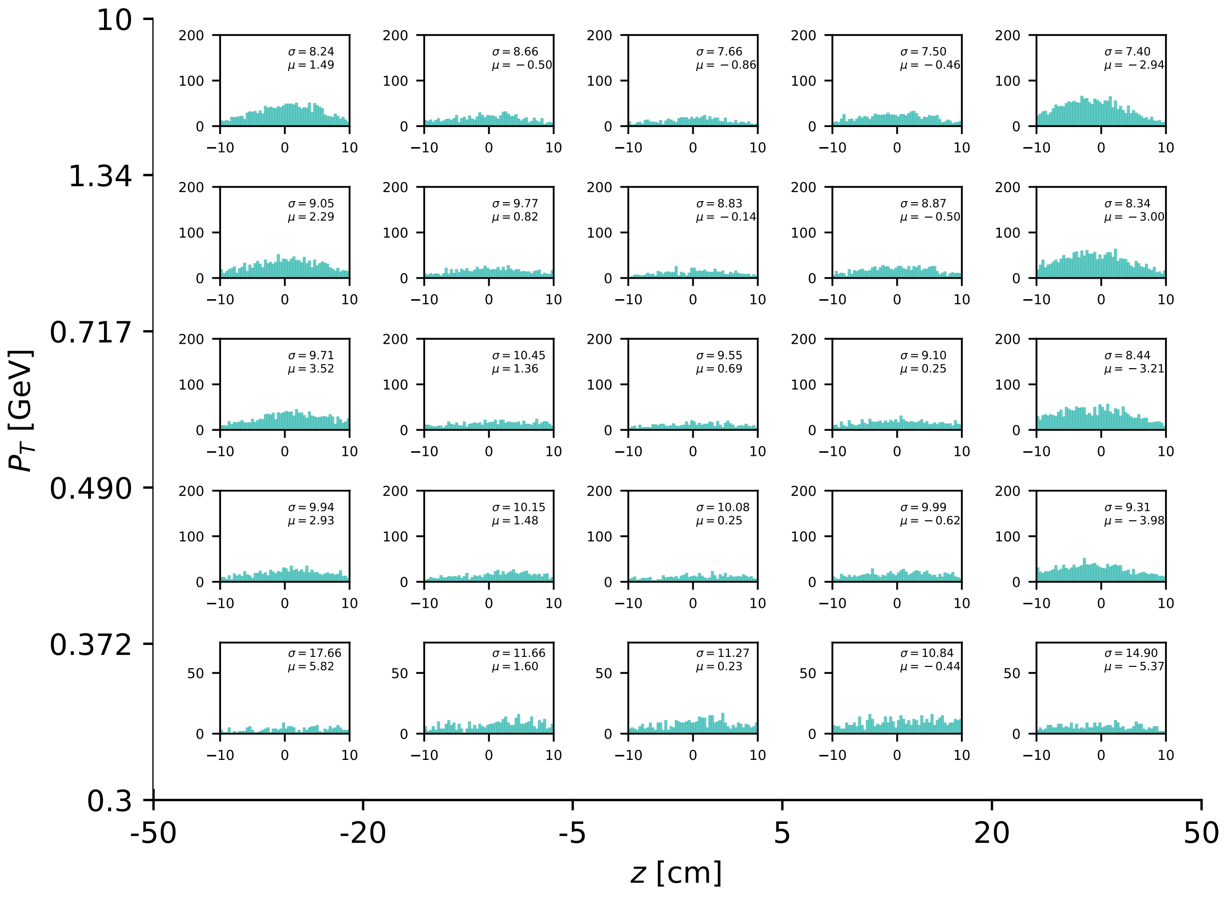


Figure B.46.: The dz resolution as a function of the particle transverse momentum p_t (y-axis) and as a function of the true z-vertex value (x-axis)

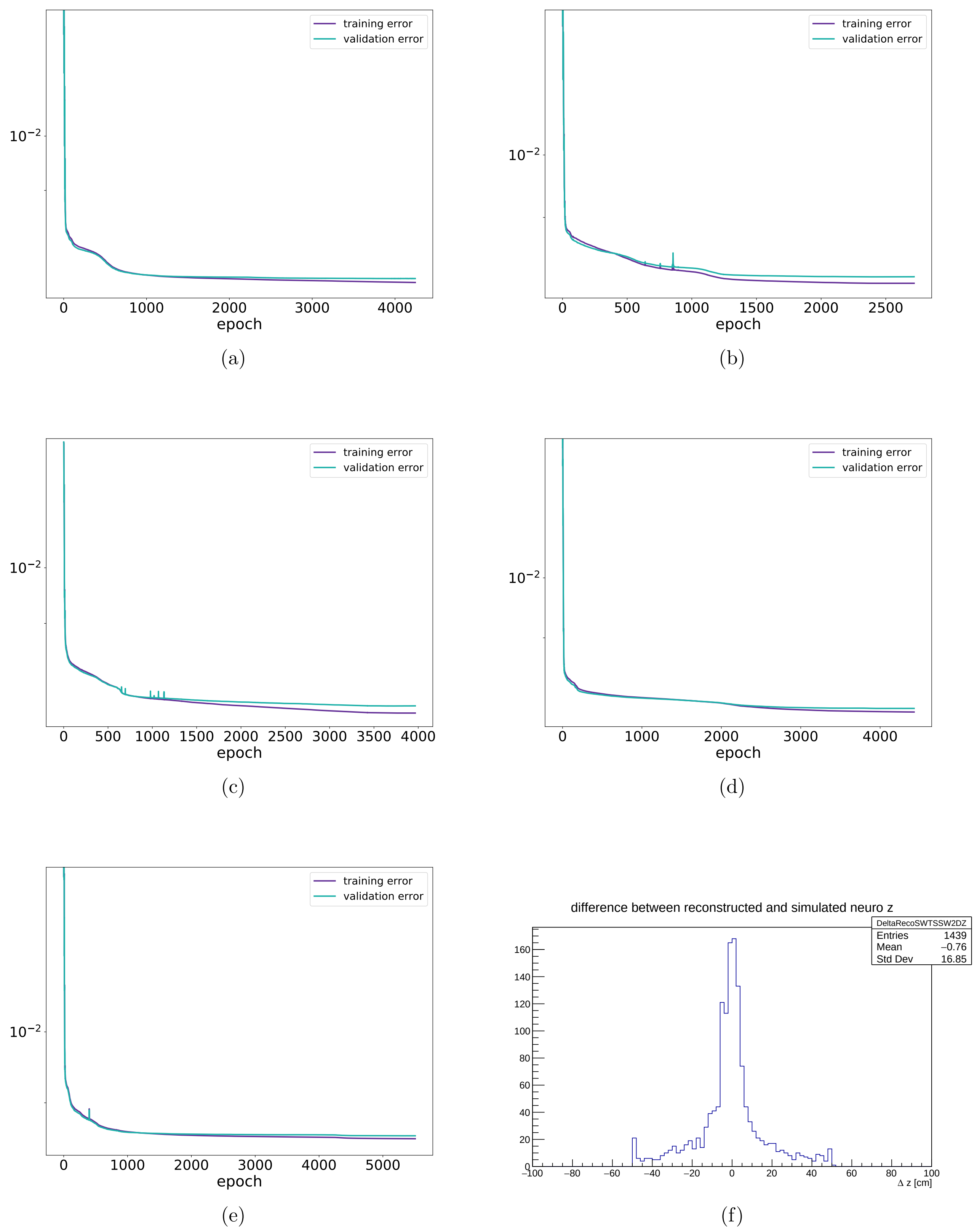
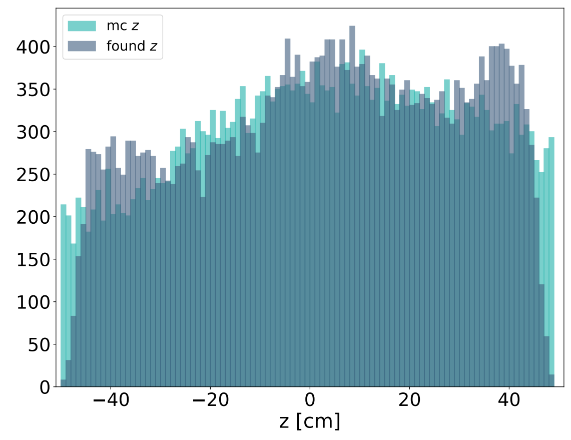
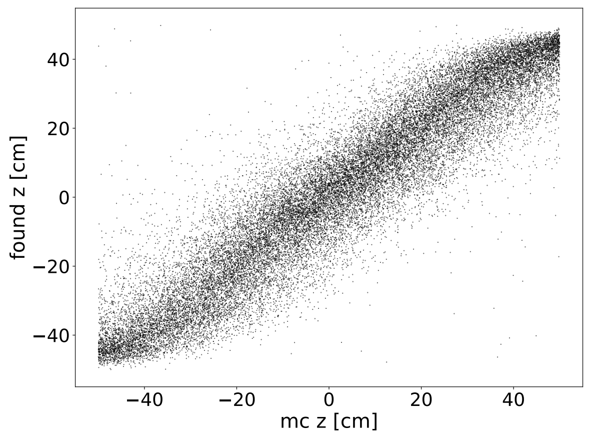


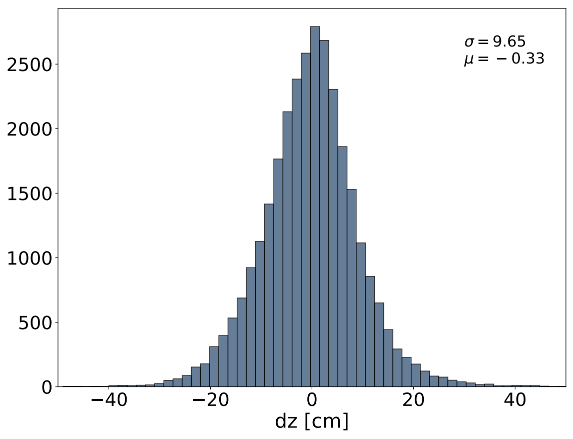
Figure B.47.: Error curves for each expert network in the Ignore Left/right network training: (a) Expert 0 (full hits); (b) Expert 1 (SL7 missing); (c) Expert 2 (SL5 missing); (d) Expert 3 (SL3 missing); (e) Expert 4 (SL1 missing), details of the training methods can be found in Chapter 4; (f) network tested with real data from early Phase 3



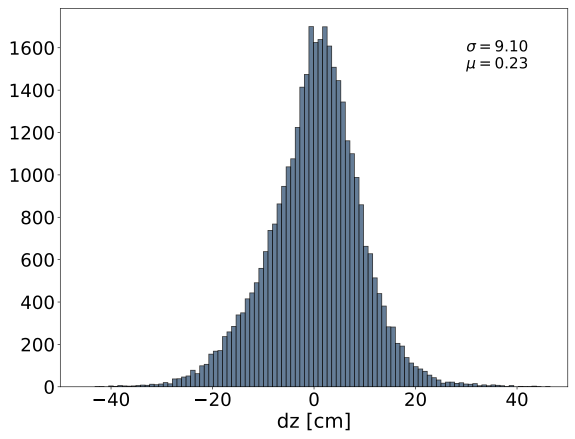
(a)



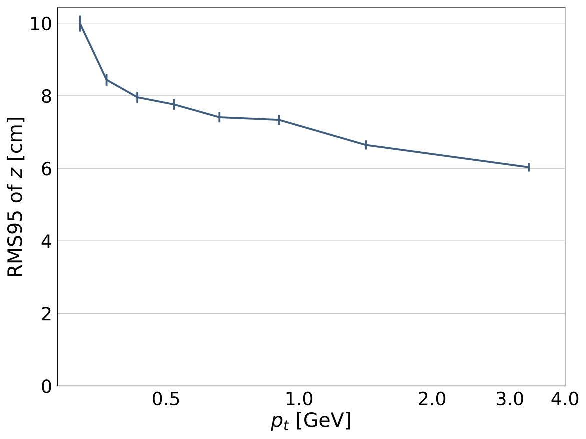
(b)



(c)



(d)



(e)

Figure B.48.: Distribution & Resolution plots for Ignore Left/right network: (a) Distribution of true z and found z-vertex values; (b) Scatter plot of the true z and found z-vertex values; (c) dz resolution (dz = mc z - found z), all (a)–(c) tested with particles generated along z=±50 cm, (d)–(e) tested with particles generated at IP (z=0); (d) dz resolution at IP; (e) p_t-dependent resolution at IP

B.0.16. 127 nodes

thetaParams	[10, 170]	**zVertexParams**	[-50, 50]
Background	Phase 2	**et_option**	fastest priority
nTrainPrepare	1000	**nValid**	1000
nTrainMax	nWeights*10	**nTrainMin**	nWeights*10
nTest	5000	**nHidden**	127
Training target	MC Particles		

Table B.17.: Training parameters for 127 nodes network

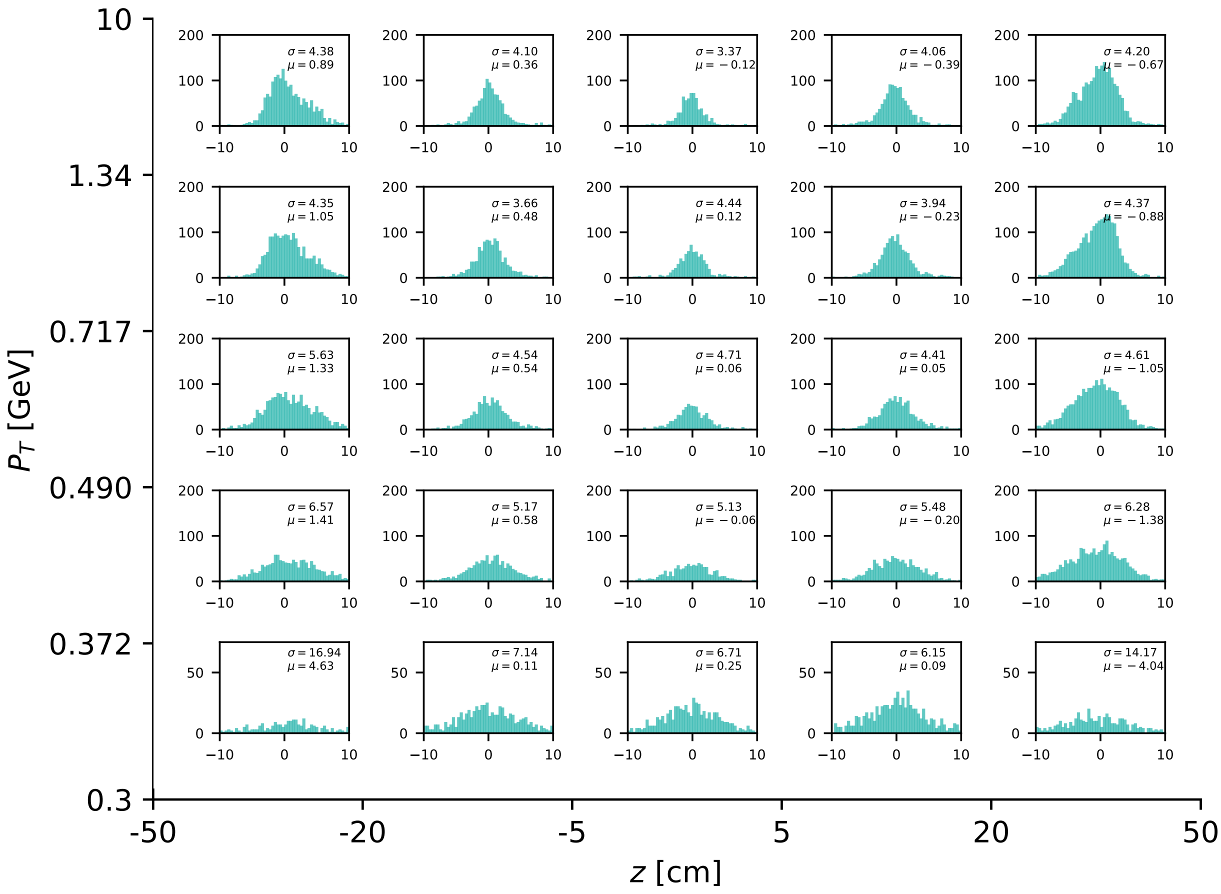


Figure B.49.: The dz resolution as a function of the particle transverse momentum p_t (y-axis) and as a function of the true z-vertex value (x-axis)

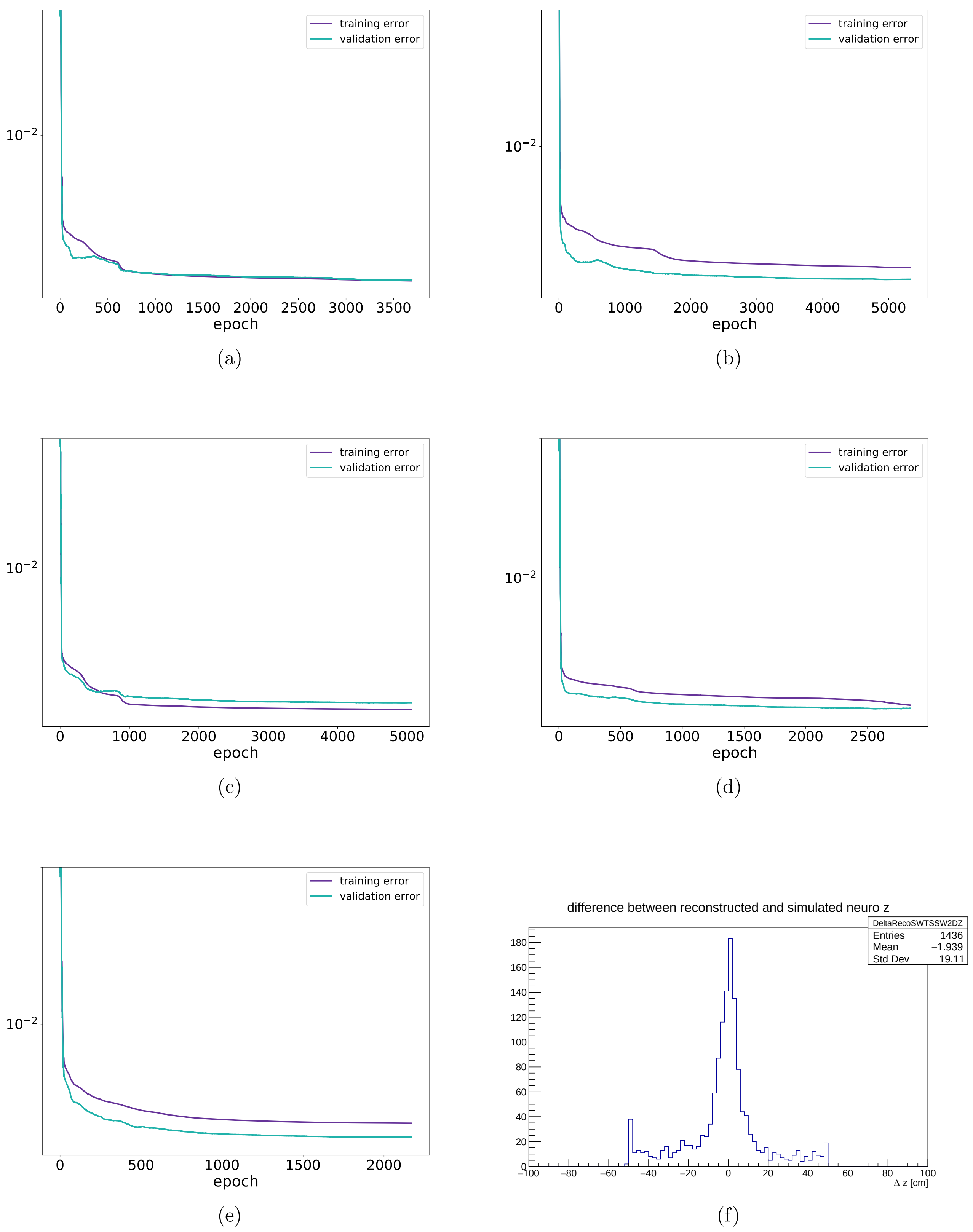
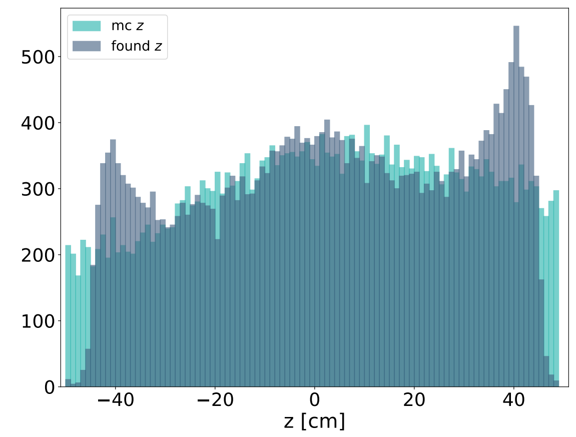
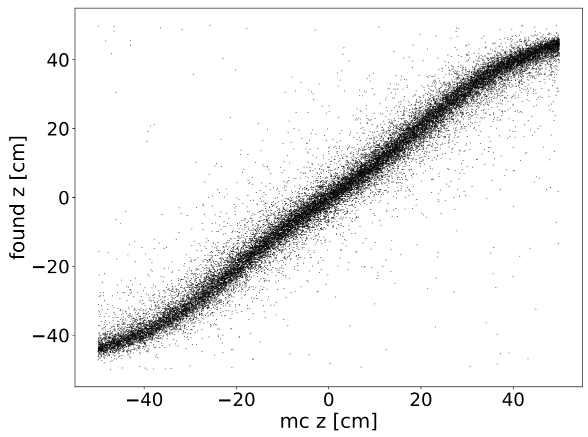


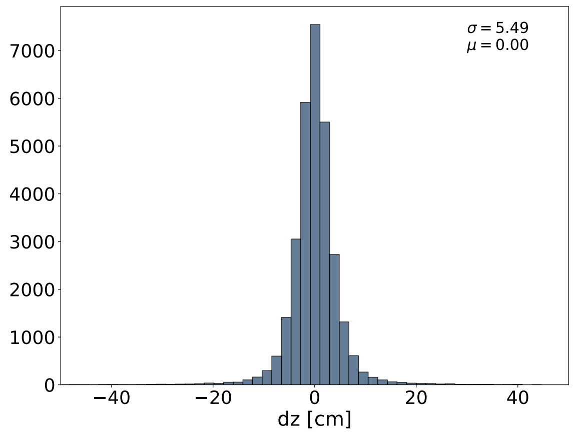
Figure B.50.: Error curves for each expert network in the 127 nodes network training: (a) Expert 0 (full hits); (b) Expert 1 (SL7 missing); (c) Expert 2 (SL5 missing); (d) Expert 3 (SL3 missing); (e) Expert 4 (SL1 missing), details of the training methods can be found in Chapter 4; (f) network tested with real data from early Phase 3



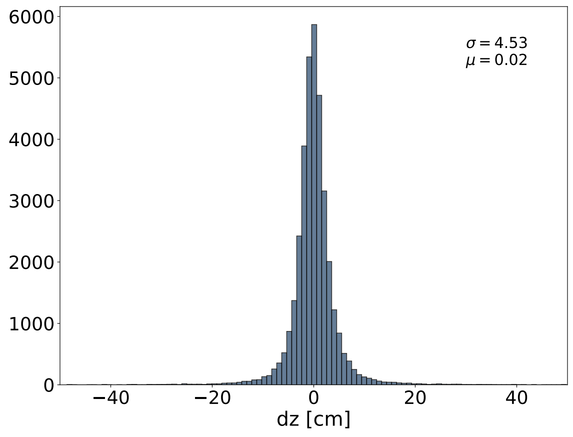
(a)



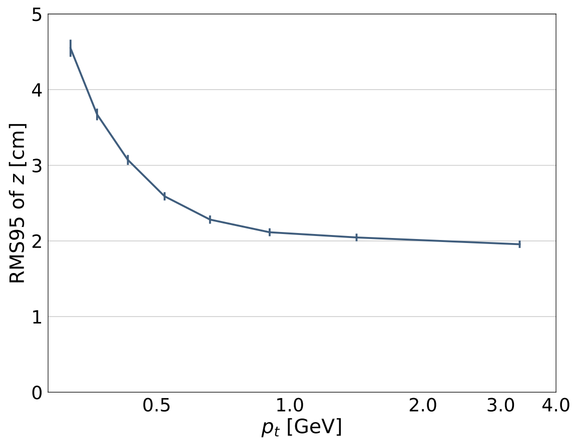
(b)



(c)



(d)



(e)

Figure B.51.: Distribution & Resolution plots for 127 nodes network: (a) Distribution of true z and found z-vertex values; (b) Scatter plot of the true z and found z-vertex values; (c) dz resolution (dz = mc z - found z), all (a)–(c) tested with particles generated along z=±50 cm, (d)–(e) tested with particles generated at IP (z=0); (d) dz resolution at IP; (e) p_t-dependent resolution at IP

B.0.17. z100

thetaParams	[10, 170]	**zVertexParams**	[-100, 100]
Background	Phase 2	**et_option**	fastest priority
nTrainPrepare	1000	**nValid**	1000
nTrainMax	nWeights*10	**nTrainMin**	nWeights*10
nTest	5000	**nHidden**	81
Training target	MC Particles		

Table B.18.: Training parameters for z100 network

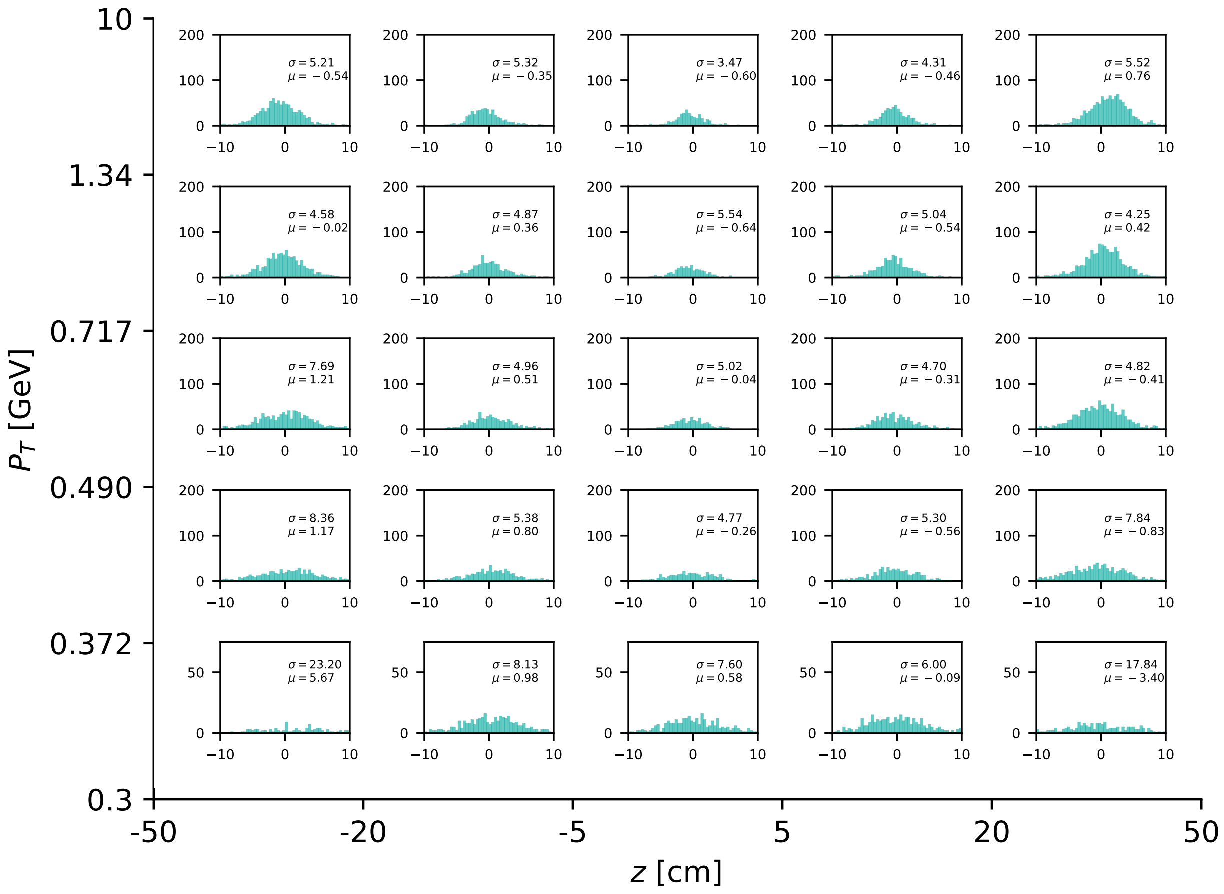
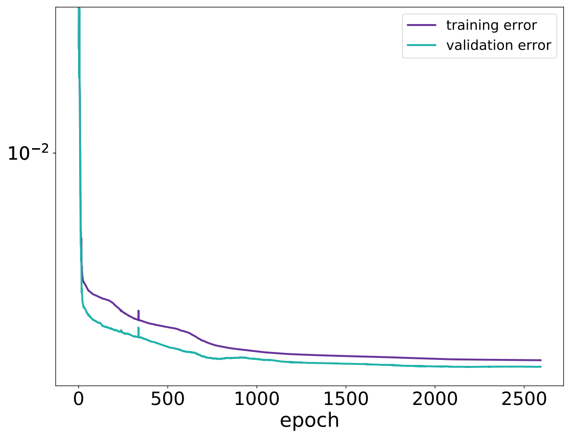
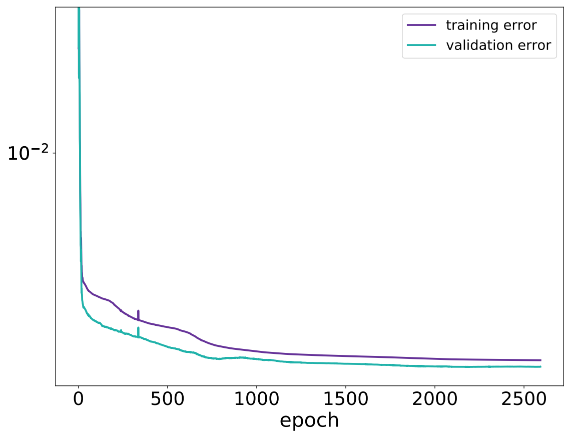


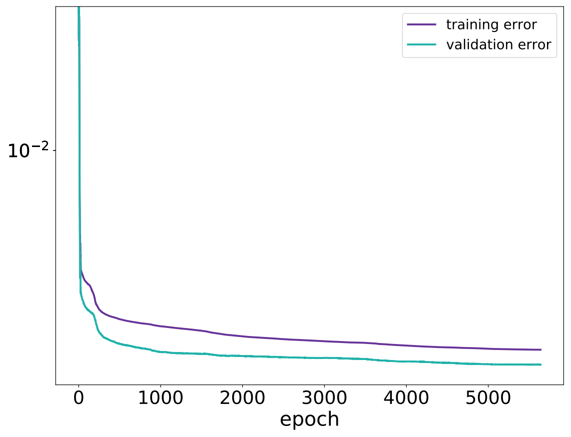
Figure B.52.: The dz resolution as a function of the particle transverse momentum p_t (y-axis) and as a function of the true z-vertex value (x-axis)



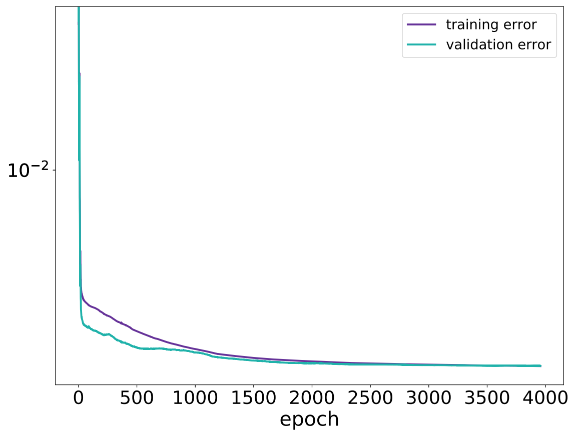
(a)



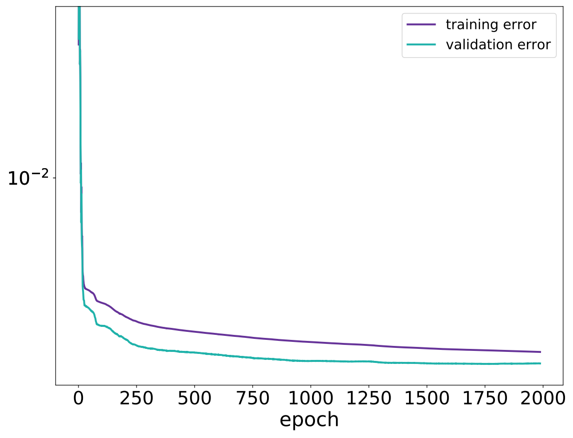
(b)



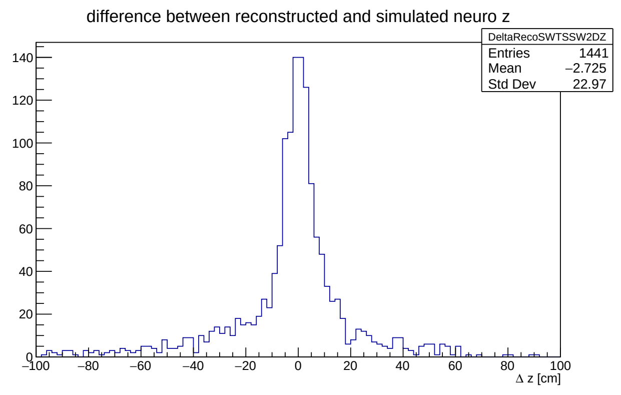
(c)



(d)

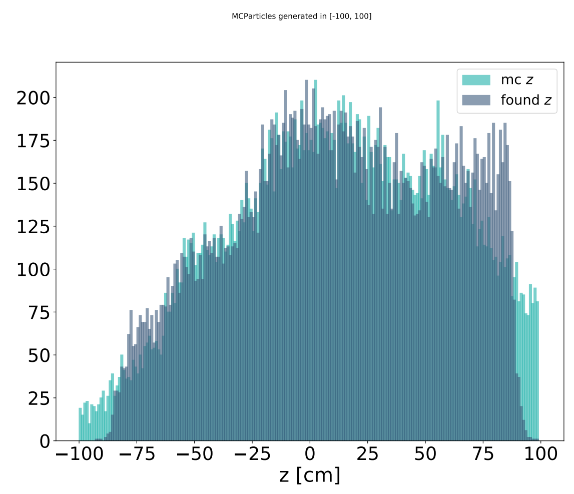


(e)

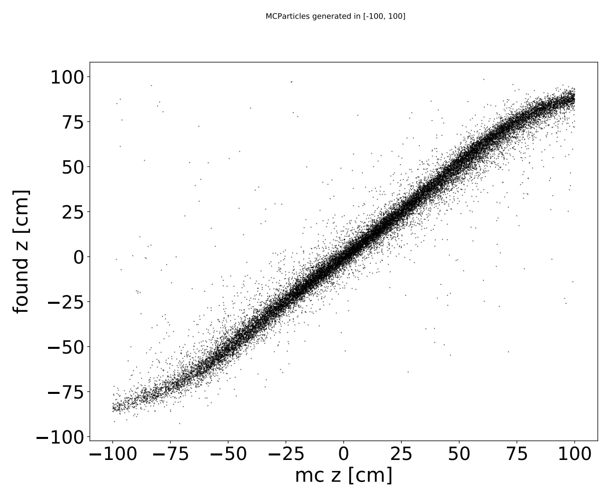


(f)

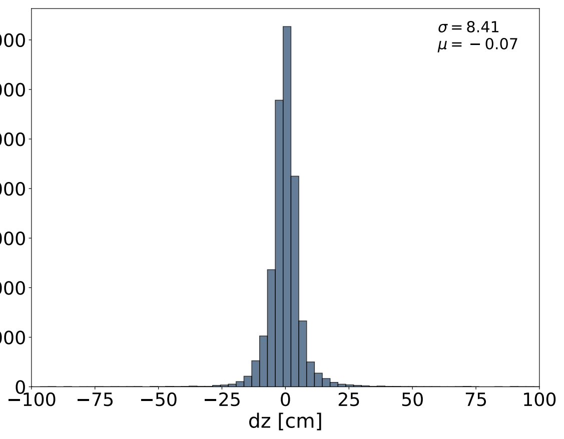
Figure B.53.: Error curves for each expert network in the $z100$ network training: (a) Expert 0 (full hits); (b) Expert 1 (SL7 missing); (c) Expert 2 (SL5 missing); (d) Expert 3 (SL3 missing); (e) Expert 4 (SL1 missing), details of the training methods can be found in Chapter 4; (f) network tested with real data from early Phase 3



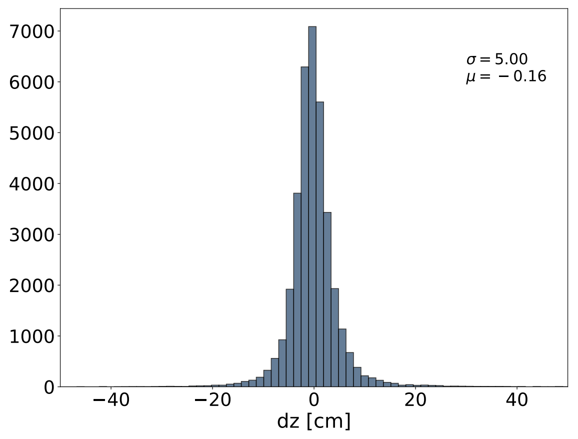
(a)



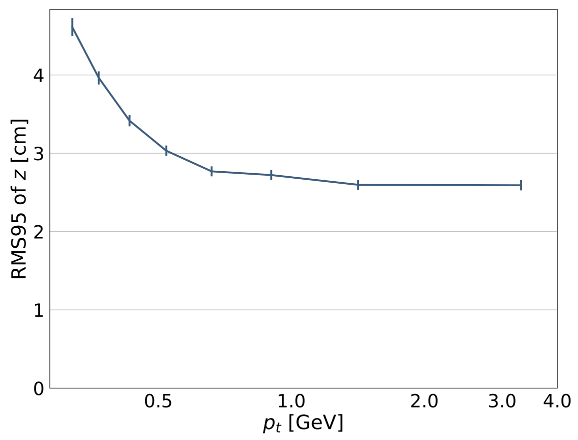
(b)



(c)



(d)



(e)

Figure B.54.: Distribution & Resolution plots for $z100$ network: (a) Distribution of true z and found z-vertex values; (b) Scatter plot of the true z and found z-vertex values; (c) dz resolution (dz = mc z - found z), all (a)–(c) tested with particles generated along z=±50 cm, (d)–(e) tested with particles generated at IP (z=0); (d) dz resolution at IP; (e) p_t-dependent resolution at IP

B.0.18. Reco Tracks

thetaParams	[10, 170]	**zVertexParams**	[-50, 50]
Background	Phase 3	**et_option**	fastest priority
nTrainPrepare	1000	**nValid**	1000
nTrainMax	nWeights*10	**nTrainMin**	nWeights*10
nTest	5000	**nHidden**	81
Training target	Reco Tracks		

Table B.19.: Training parameters for Reco Tracks network

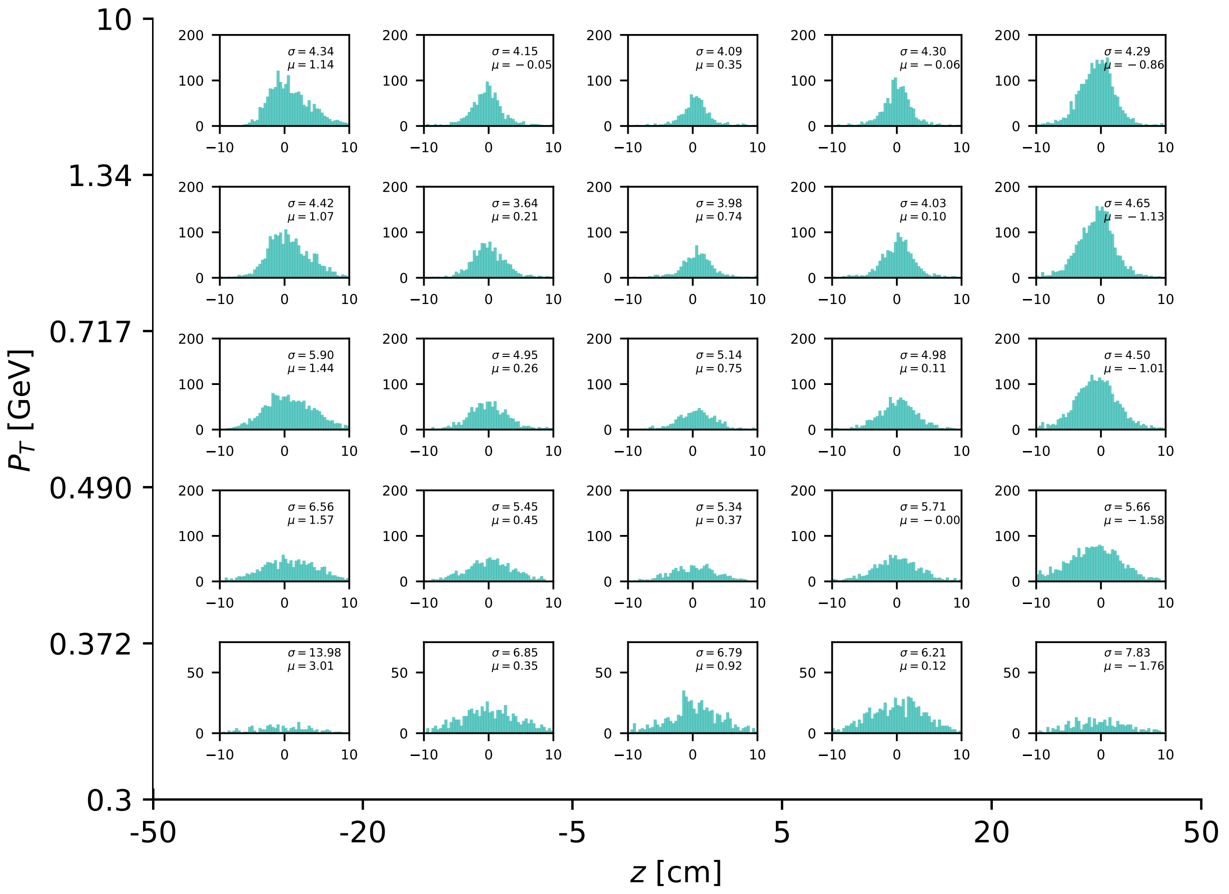
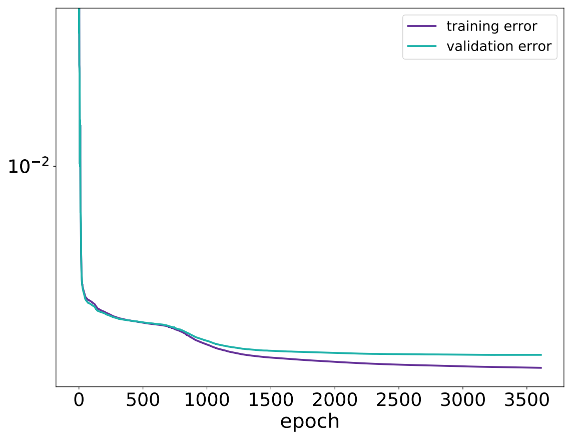
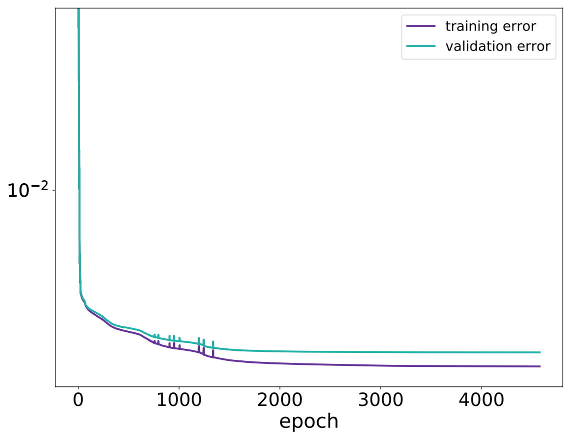


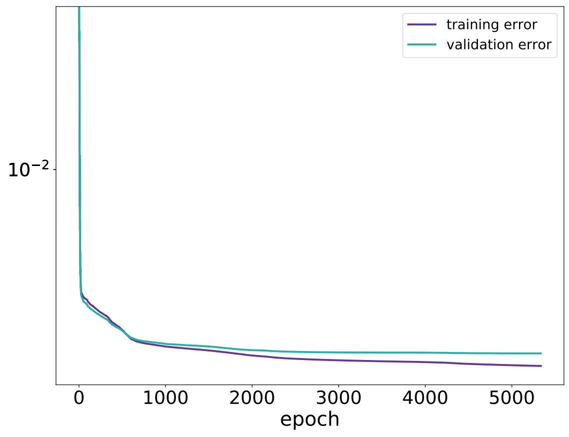
Figure B.55.: The dz resolution as a function of the particle transverse momentum p_t (y-axis) and as a function of the true z-vertex value (x-axis)



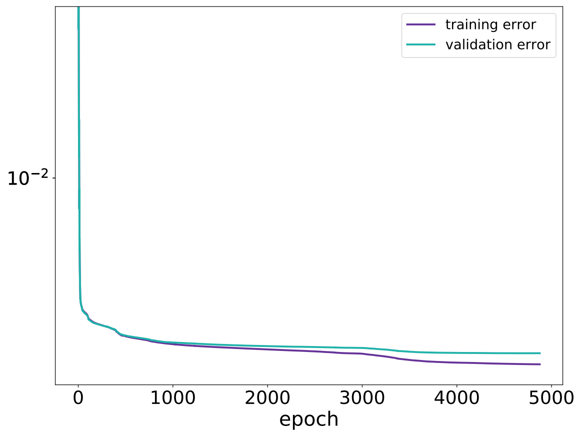
(a)



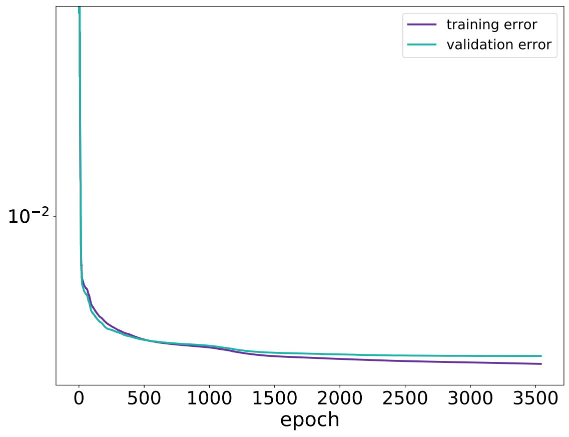
(b)



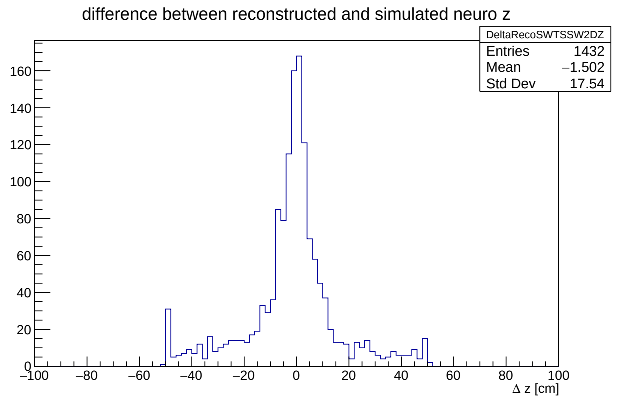
(c)



(d)



(e)



(f)

Figure B.56.: Error curves for each expert network in the Reco Tracks training: (a) Expert 0 (full hits); (b) Expert 1 (SL7 missing); (c) Expert 2 (SL5 missing); (d) Expert 3 (SL3 missing); (e) Expert 4 (SL1 missing), details of the training methods can be found in Chapter 4; (f) network tested with real data from early Phase 3

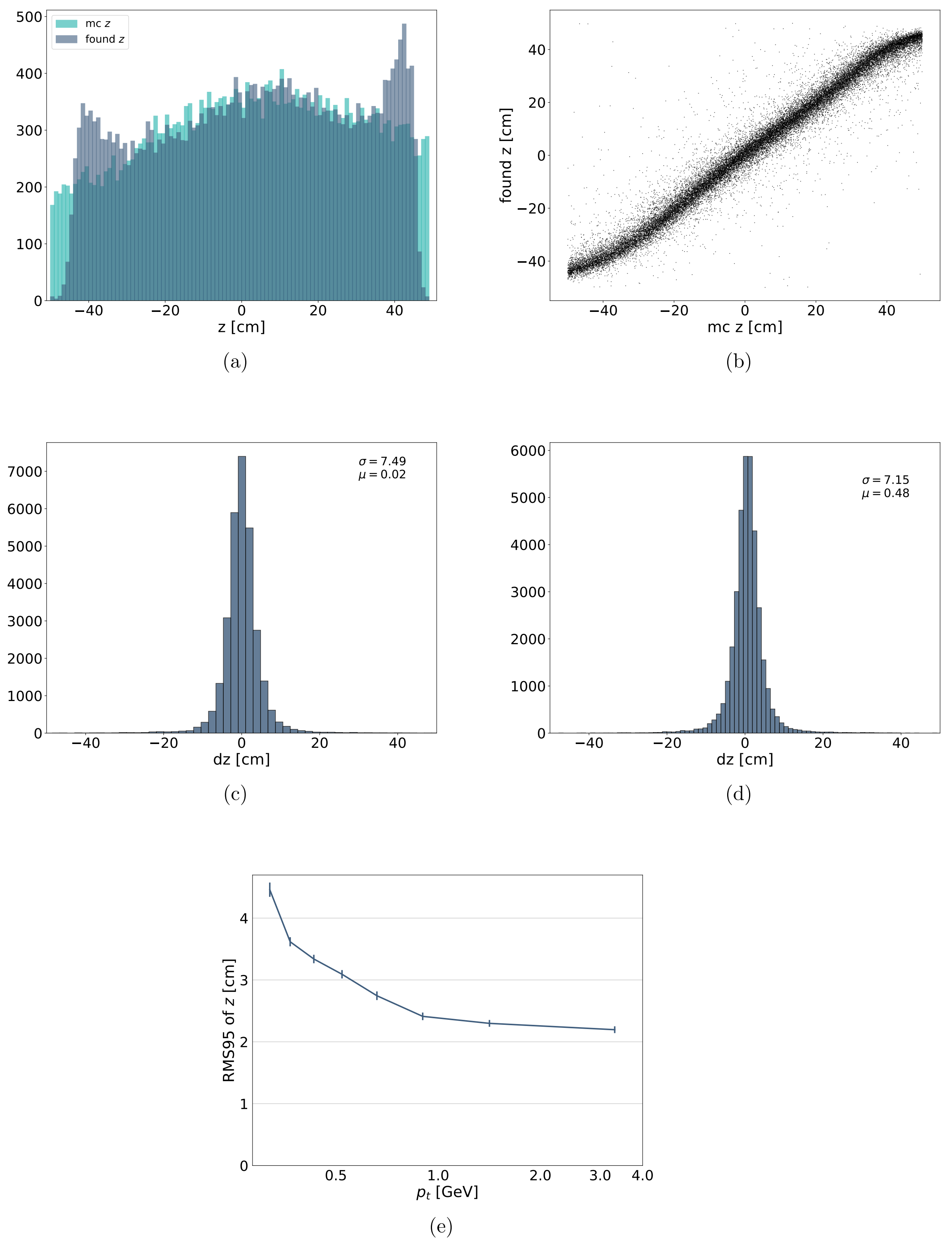


Figure B.57.: Distribution & Resolution plots for ETF Phase 3 Threshold 5 network: (a) Distribution of true z and found z-vertex values; (b) Scatter plot of the true z and found z-vertex values; (c) dz resolution (dz = mc z - found z), all (a)–(c) tested with particles generated along z=±50 cm, (d)–(e) tested with particles generated at IP (z=0); (d) dz resolution at IP; (e) p_t-dependent resolution at IP

C. Error Estimation

The distribution of a random variable X can be described by its moments, with the n^{th} moments μ_n defined by

$$\mu_n = \langle (x - \langle x \rangle)^n \rangle \tag{C.1}$$

where $\langle x \rangle$ is the expectation value of X. The variance of a random variable X is the 2^{nd} moment μ_2 and can be written as

$$Var[x] = \mu_2 = \langle (x - \langle x \rangle)^2 \rangle = \langle x^2 - 2x\langle x \rangle + \langle x \rangle^{"} \rangle = \langle x^2 \rangle - \langle x \rangle^2 \tag{C.2}$$

the standard deviation σ is defined at the square root of the variance

$$\sigma = \sqrt{Var[x]} = \sqrt{\mu_2} \tag{C.3}$$

Typically only the first 2 moments are used since in Gaussian distributions all higher moments (n>2) vanish, and the distributions are described by its mean $\langle x \rangle$ and its variance. However if we want to define an error to the variance, we can use the same definition of variance and calculate the related variance. Substituting $Var[x] = \langle y \rangle$ we obtain

$$Var[y] = \langle y^2 \rangle - \langle y \rangle^2 = \langle (x - \langle x \rangle^4 \rangle - \langle (x - \langle x \rangle^2 \rangle^2 = \mu_4 - \mu_2^2 \tag{C.4}$$

where

$$\mu_4 = \langle (x - \langle x \rangle^4) \rangle = \langle x^4 \rangle - 4\mu \langle x^3 \rangle + 6\mu_2 \langle x^2 \rangle - 3\mu_4 \tag{C.5}$$

which is known as kurtosis. An error of standard deviation can then be propagated from the error of the variance [36]:

$$\Delta\sigma = \left| \frac{\partial \sigma}{\partial \mu_2} \Delta\mu_2 \right| = \left| \frac{-1}{2\sqrt{\mu_2}} \sqrt{\frac{1}{N}(\mu_4 - \mu_2^2)} \right| = \frac{1}{2\sigma} \sqrt{\frac{1}{N}(\mu_4 - \sigma^4)} \tag{C.6}$$

Bibliography

[1] T Abe, I Adachi, K Adamczyk, S Ahn, H Aihara, K Akai, M Aloi, L Andricek, K Aoki, Y Arai, et al. Belle II technical design report. *arXiv preprint arXiv:1011.0352*, 2010.

[2] Sara Neuhaus, Sebastian Skambraks, Fernando Abudinén, Yang Chen, Michael Feindt, Rudolf Frühwirth, Martin Heck, Christian Kiesling, Alois Knoll, Stephan Paul, et al. A neural network z-vertex trigger for Belle II. In *Journal of Physics: Conference Series*, volume 608, page 012052. IOP Publishing, 2015.

[3] Sara Neuhaus, Sebastian Skambraks, and Christian Kiesling. Track vertex reconstruction with neural networks at the first level trigger of Belle II. In *EPJ Web of Conferences*, volume 150, page 00009. EDP Sciences, 2017.

[4] Sebastian Skambraks, Sara Neuhaus, and Christian Kiesling. The Neuro-Z-Vertex Trigger of the Belle II Experiment. In *EPJ Web of Conferences*, volume 127, page 00016. EDP Sciences, 2016.

[5] Steffen Baehr, S Skambraks, S Neuhaus, C Kiesling, and J Becker. A neural network on FPGAs for the z-vertex track trigger in Belle II. *Journal of Instrumentation*, 12(03):C03065, 2017.

[6] Emi Kou, P Urquijo, W Altmannshofer, F Beaujean, G Bell, M Beneke, II Bigi, F Bishara, M Blanke, C Bobeth, et al. The Belle II physics book. 2018.

[7] Georges Aad, Tatevik Abajyan, B Abbott, J Abdallah, S Abdel Khalek, Ahmed Ali Abdelalim, O Abdinov, R Aben, B Abi, M Abolins, et al. Observation of a new particle in the search for the standard model higgs boson with the atlas detector at the lhc. *Physics Letters B*, 716(1):1–29, 2012.

[8] CMS Collaborations et al. Combined measurement of the higgs boson mass in pp collisions at $\backslash sqrt\{s\}=$ 7 and 8 tev with the atlas and cms experiments. *arXiv preprint arXiv:1503.07589*, 2015.

[9] Sara Pohl. *Belle II z-vertex trigger*. PhD thesis, Ludwig-Maximilians-Universität München, 2018.

[10] Emmy Noether. Invariant variation problems. *Transport Theory and Statistical Physics*, 1(3):186–207, 1971.

[11] Chien-Shiung Wu, Ernest Ambler, RW Hayward, DD Hoppes, and Ralph Percy Hudson. Experimental test of parity conservation in beta decay. *Physical review*, 105(4):1413, 1957.

[12] Nicola Cabibbo. Unitary symmetry and leptonic decays. *Physical Review Letters*, 10(12):531, 1963.

[13] Sheldon L Glashow, Jean Iliopoulos, and Luciano Maiani. Weak interactions with lepton-hadron symmetry. *Physical review D*, 2(7):1285, 1970.

[14] James H Christenson, Jeremiah W Cronin, Val L Fitch, and René Turlay. Evidence for the 2 π decay of the k 2 0 meson. *Physical Review Letters*, 13(4):138, 1964.

[15] Jean-Jacques Aubert, U Becker, PJ Biggs, J Burger, M Chen, G Everhart, P Goldhagen, J Leong, T McCorriston, TG Rhoades, et al. Experimental observation of a heavy particle j. *Physical Review Letters*, 33(23):1404, 1974.

[16] Daniel M Kaplan. The discovery of the upsilon family. In *History of Original Ideas and Basic Discoveries in Particle Physics*, pages 359–380. Springer, 1996.

[17] F Abe, H Akimoto, A Akopian, MG Albrow, SR Amendolia, D Amidei, J Antos, C Anway-Wiese, S Aota, G Apollinari, et al. Observation of top quark production in p p collisions with the collider detector at fermilab. *Physical review letters*, 74(14):2626, 1995.

[18] S Abachi, B Abbott, M Abolins, BS Acharya, I Adam, DL Adams, M Adams, S Ahn, H Aihara, G Alvarez, et al. Search for high mass top quark production in p p collisions at s= 1.8 tev. *Physical Review Letters*, 74(13):2422, 1995.

[19] Kazuo Abe, R Abe, I Adachi, Byoung Sup Ahn, H Aihara, M Akatsu, G Alimonti, K Asai, M Asai, Y Asano, et al. Observation of large cp violation in the neutral b meson system. *Physical Review Letters*, 87(9):091802, 2001.

[20] Jérôme Charles, A Höcker, Heiko Lacker, Sandrine Laplace, FR Le Diberder, Julie Malclès, José Ocariz, Muriel Pivk, and Lydia Roos. Cp violation and the ckm matrix: Assessing the impact of the asymmetric b factories. *The European Physical Journal C-Particles and Fields*, 41(1):1–131, 2005.

[21] Lincoln Wolfenstein. Parametrization of the kobayashi-maskawa matrix. *Physical Review Letters*, 51(21):1945, 1983.

[22] Andrej Dmitrievich Sakharov. Violation of cp invariance, c asymmetry, and baryon asymmetry of the universe. *JETP lett.*, 5:24–27, 1967.

[23] Ziro Maki, Masami Nakagawa, and Shoichi Sakata. Remarks on the unified model of elementary particles. *Progress of Theoretical Physics*, 28(5):870–880, 1962.

[24] M. et al Tanabashi. Review of particle physics. *Phys. Rev. D*, 98:030001, Aug 2018.

[25] K Nishimura et al. New Physics Prospects in Mixing and CP violation at Belle II. *arXiv preprint arXiv:1212.4112*, 2012.

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

[27] Sarah Lawhun. First collisions at belle ii.

[28] Yutaka Ushiroda and Kazunori Akai. Superkekb phase 3 (belle ii physics run) starts.

[29] S. Sandilya. Particle Identification with the TOP and ARICH Detectors at Belle II. *Springer Proc. Phys.*, 203:563–565, 2018.

[30] Belle-ECL, V Aulchenko, A Bobrov, A Bondar, B G Cheon, S Eidelman, D Epifanov, Yu Garmash, Y M Goh, S H Kim, P Krokovny, A Kuzmin, I S Lee, D Matvienko, K Miyabayashi, I Nakamura, V Shebalin, B Shwartz, Y Unno, Yu Usov, A Vinokurova, V Vorobjev, V Zhilich, and V Zhulanov. Electromagnetic calorimeter for belle II. *Journal of Physics: Conference Series*, 587:012045, feb 2015.

[31] A. Moll. Comprehensive study of the background for the pixel vertex detector at belle ii. PhD Thesis, 2015.

[32] R Itoh, T Higuchi, M Nakao, SY Suzuki, and S Lee. Data flow and high level trigger of belle ii daq system. *IEEE Transactions on Nuclear Science*, 60(5):3720–3724, 2013.

[33] Sara Neuhaus, Sebastian Skambraks, Fernando Abudinn, Yang Chen, Michael Feindt, Rudolf Frhwirth, Martin Heck, Christian Kiesling, Alois Knoll, Stephan Paul, and Jochen Schieck. A neural network z-vertex trigger for Belle II. arXiv:1410.1395 [physics.ins-det], 2015.

[34] Sebastian Skambraks. A 3d track finder for the belle ii cdc l1 trigger. *ACAT*, 2019.

[35] Sea Agostinelli, John Allison, K al Amako, John Apostolakis, H Araujo, P Arce, M Asai, D Axen, S Banerjee, G 2 Barrand, et al. GEANT4a simulation toolkit. *Nuclear instruments and methods in physics research section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 506(3):250–303, 2003.

[36] Jens Zimmerman. *Statistical Learning in High Energy and Astrophysics.* PhD thesis, Diploma Thesis, Ludwig-Maximilians-Universität München, MPP-2013-339, 2013.

Acknowledgements

I firstly wish to express my thanks to my supervisor Prof. Dr. Christian Kiesling whose expertise and guidance allowed me to explore many corners of this project and whose enthusiasm inspired me to test new ideas. Thank you for answering my emails late into the night and for your constant encouragement.

Special thanks to my Neurotrigger colleagues Sebastian Skambraks and Felix Meggendorfer for sharing their knowledge, teaching me more about programming than I would have imagined possible. Thanks especially for your patience and willingness to help, it is truly appreciated.

Thanks to the Belle II group including Dr. Hans-Günther Moser at Max Planck Institute for Physics for the great time I had during my Master thesis.

Thanks also to my Neurotrigger colleagues at Karlsruhe - Steffen Bähr, Raynette van Tonder and Kai Lucas Unger and to the Belle II trigger group for the insightful discussions and meetings.

Thanks also to Keti Parlagashvili, Meghranjana Chatterjee, Markus Reif & Kejal Patel for their constant support. Thanks to non-physics friends Aina Vidal Lorente, Vasiliki Georgaki and Ross Braniff for reminding me that there is life outside of physics.

Last but not least thanks to my parents and three brothers for their unwavering support and encouragement during my studies.