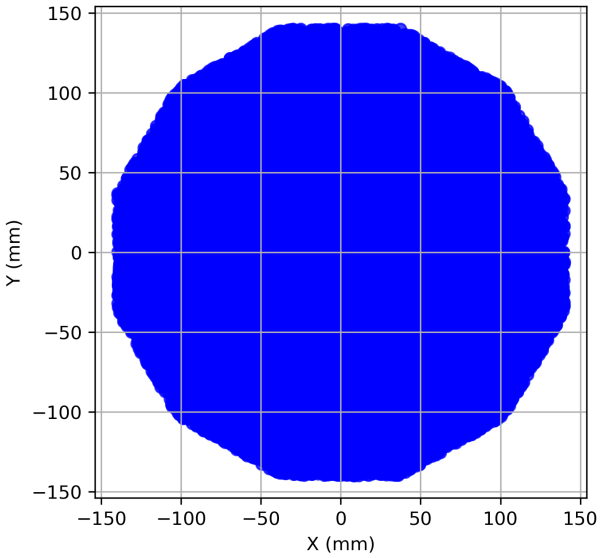
2 Computational Methods

2.1 Data

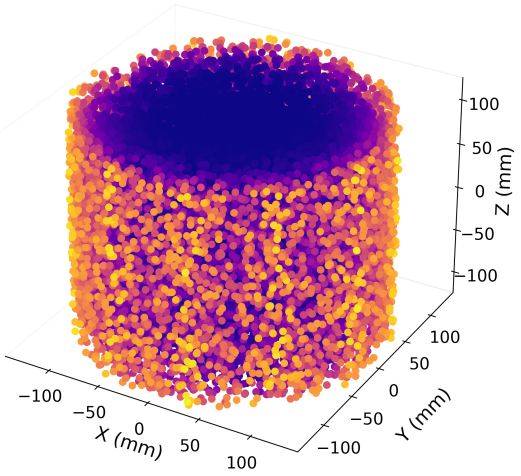
The data used consists of simulated events represented as 54 parallel detector readouts, each sampled over a 64 millisecond time window at a frequency of 256 Hz. This gives 16384 time steps per channel or a sequence length of 16384. The 54 channels correspond to individual MMCs, of which 9 are positioned above the superfluid surface. These upper detectors are the only ones that can detect the quasiparticle channel via helium atom evaporation. The remaining 45 detectors are submerged and primarily capture UV scintillation and triplet excimer signals. Because of this geometric configuration, and the differing propagation times and coupling mechanisms of these signal channels, the waveforms contain distinct spatial and temporal signatures. These signals are of either NR or ER events, covering a range of discrete energies –- 50 eV, 100 eV, 500 eV, 1000 eV, 10000 eV and 100000 eV. Each file contains 128 such events of each type and has the types specified for each event, so that the data is arranged into tensors of shape (128, 54, 16384) per event type (256 events per file). During loading, the events were assigned class values of 0 and 1 for ER and NR respectively.

The files also contain the positions and energies of the events. The event positions are confined to a volumetric cell defined by an x-y range of ± 140 mm (280 mm total span) and a z range of -1953 to -1713 mm (240 mm total span). The xy cross-section is geometrically constrained by a dodecagonal prismatic structure, reflecting the theoretical arrangement of detectors – 36 positioned along the sides of the cell, with 9 distributed across the bottom of the cell and 9 above the LHe. This configuration results in a non-cylindrical, polygonal volume that extends uniformly along the z-axis. A plot of the xy cross-section of the positions and a 3D plot of the positions can be seen in Figure 5.

Two main datasets were used: noiseless (idealised) and noised (realistic), with the latter generated by adding Gaussian white noise to the waveforms to simulate detector backgrounds. Figure 6 displays example waveforms. Preliminary tests revealed that Fourier-transformations of the data offered no significant improvement in training efficiency or model accuracy. Thus, the models used raw data with only two preprocessing steps: centring the z-coordinate at zero and log-normalising energies to handle their wide range spanning multiple orders of magnitude.



(a) XY cross-section of simulated event positions.



(b) 3D plot of simulated event positions, with radial distance encoded by a skewed colour map (scaling as r^7). Yellow indicates maximal distance from the centre, while purple corresponds to points near the origin. The non-linear colour map accentuates the edges of the volume.

Figure 5: Visualisation of the spatial distribution of simulated events.

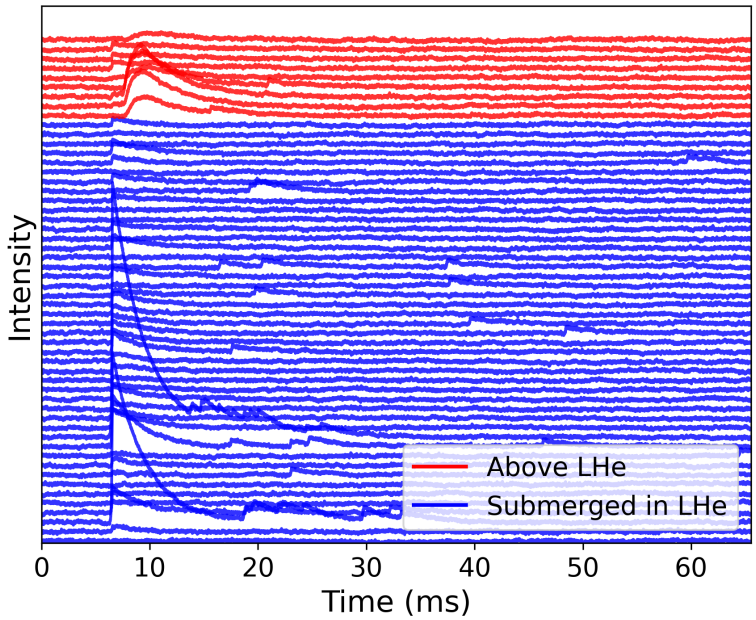


Figure 6: Plot displaying simulated waveforms from the 54-detector array for a 10 keV NR event, including noise contributions. The traces are colour-coded by detector location: red represents the 9 detectors positioned above the LHe volume, while blue corresponds to the 45 submerged detectors. The early waveform double peaks in the above-LHe detectors originate from UV/IR and quasiparticle detection, with the more rounded peak resulting from quasiparticle detection. Subsequent peaks beyond 20 ms are from triplet detection.

2.2 Model Architecture

Three models were constructed: PR (Position Reconstruction), PRC (Position Reconstruction and Classification), and PRCE (Position Reconstruction, Classification and Energy regression). This progressive approach first validated the most challenging timing-based position reconstruction, then tested classification capability, and finally integrated energy estimation to examine how multi-task learning improves performance while isolating each component's behaviour.

To learn from this waveform data, neural networks were employed to create the models. A neural network is a computational model made up of interconnected layers of artificial "neurons." Each neuron performs a simple mathematical operation – it takes one or more inputs, applies a set of learned weights to them, sums the result, adds a bias term, and passes it through a non-linear activation function [33]. Mathematically, the operation performed by a single neuron is expressed as

$$y = f\left(\sum_{i=1}^{n} w_i x_i + b\right)$$

where x_i are the inputs, w_i are the weights, b is the bias, and is f the activation function. The purpose of the activation function is to introduce non-linearity, allowing the network to model complex patterns [33]. In the models made in this project, the ReLU (Rectified Linear Unit) and GELU (Gaussian Error Linear Unit) activations were the main ones used. The ReLU activation function outputs the input directly if it is positive; otherwise, it outputs zero [34]. It is expressed as

$$ReLU(x) = max(0, x).$$

It introduces non-linearity in neural networks, helping them learn complex patterns while being computationally efficient. This can cause issues when trying to predict negative values such as for position reconstruction. For this reason, it was only implemented for energy regression. The GELU activation function weights inputs by their percentile under a Gaussian distribution, smoothly approximating ReLU [34]. It is expressed as

$$GELU(x) = x\Phi(x),$$

where $\Phi(x)$ is the cumulative Gaussian distribution. The GELU activation function was chosen for both position reconstruction and classification tasks because, unlike ReLU, it permits negative values – an important feature for modeling signed spatial coordinates. For the binary classification task specifically, GELU's non-zero gradient across all inputs helps prevent dying neurons during training, which is particularly valuable when learning from binary targets (0/1 labels for ER and NR respectively).

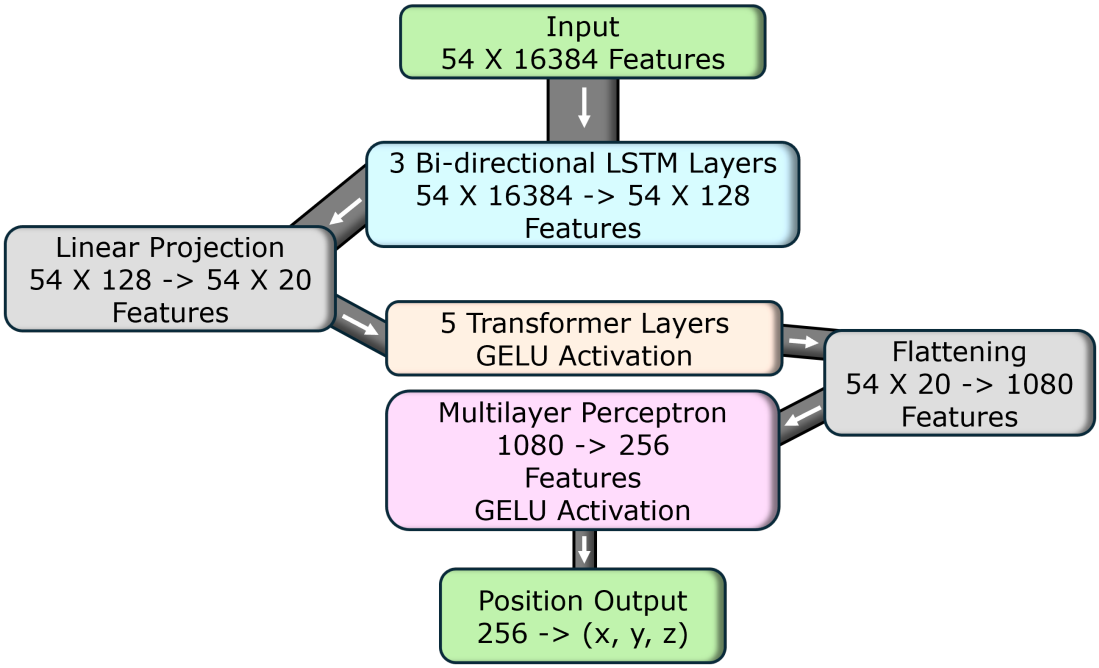


Figure 7: Schematic of the PR model. It shows the flow of a single input (one event). 54 is the number of detectors, 16384 is the sample length (features).

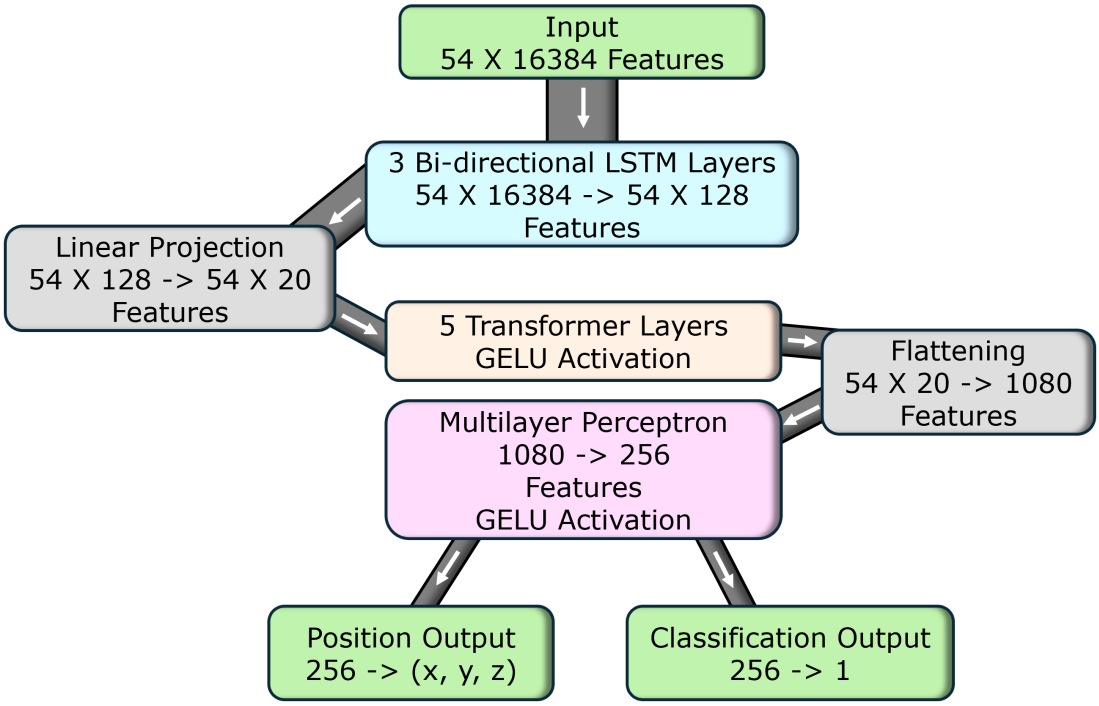


Figure 8: Schematic of the PRC model.

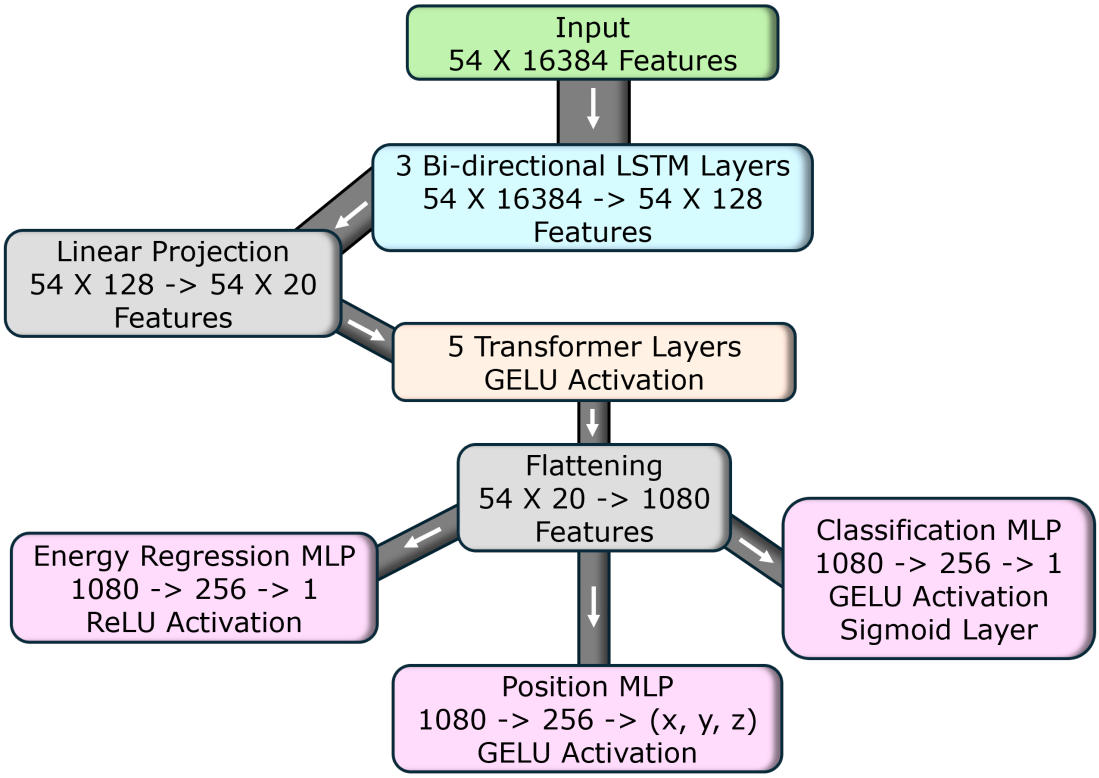


Figure 9: Schematic of the PRCE model.

All models were implemented in Python [35] using the PyTorch framework [36]. The architecture used across all tasks shares a consistent backbone combining recurrent and attention-based components – specifically, a bidirectional Long Short-Term Memory (LSTM) network followed by a Transformer block. Diagrams of each model's architecture can be seen in Figures 7, 8 and 9.

Each model begins with a three-layer bidirectional LSTM. LSTMs are recognised as a specialised type of recurrent neural network which have been designed to model sequential data effectively through their distinctive gating architecture [37]. In contrast to standard RNNs, the vanishing gradient problem is addressed by LSTMs via their cell state mechanism and three varieties of gates: input gates by which new information is regulated, forget gates through which unnecessary information is discarded, and output gates that determine what is passed to the subsequent time step [37]. This architecture renders them particularly suitable for processing the detector's high-resolution waveforms, which extend across 16,384 time steps.

The bidirectional implementation is characterised by each sequence being processed in both temporal directions – one pass moving forwards through time and another moving backwards, enabling contextual information from the complete sequence to be incorporated when any given time point is analysed. Each LSTM layer is composed of 64 units, though the bidirectional architecture results in 128 features per detector being produced, as the forwards and backwards passes generate separate outputs that are concatenated. These features are designed to capture the most significant temporal patterns from the raw waveforms while preserving information about long-range dependencies.

To prevent overfitting during training, dropout regularisation is applied with a rate of 0.3. Dropout operates by randomly deactivating a fraction of neurons during each training iteration, which compels the network to develop redundant representations and prevents excessive dependence on any individual neuron [38]. Finally, the LSTM outputs are passed through a linear projection layer where the dimensionality is reduced to 20 features per detector, thereby matching the required input size for the subsequent Transformer block.

A Transformer is a powerful architecture for sequential data that leverages self-attention mechanisms to dynamically assess relationships across the entire input sequence. Unlike traditional recurrent networks, Transformers process all sequence elements in parallel, with self-attention weights determining the relevance of each element to others—enabling the model to focus on the most informative features while preserving global context [39].

The Transformer block used in the models is composed of five stacked encoder layers. Each encoder layer consists of multi-head attention (using three parallel attention heads), followed by a position-wise feed-forward network with 128 hidden dimensions and GELU activation. Residual connections and layer normalisation stabilise training across the stacked layers, while dropout (applied at 0.2 rate) regularises activations to prevent overfitting.

The Transformer serves as a relational reasoning module, processing the LSTM-extracted waveforms from all 54 detectors. Through self-attention, it dynamically models inter-detector relationships, weighing their feature-based importance to uncover spatial patterns. Unlike isolated processing, this explicit linking enables position inference via "triangulation" of collective waveform features. The Transformer's relational reasoning can be also leveraged for energy regression, as inter-detector correlations are weighted according to their predictive relevance for deposition magnitude. The attention mechanism can highlight detector-level patterns diagnostic of recoil type (e.g., UV-dominated vs. quasiparticle-dominated signals), allowing ER/NR discrimination.

The quadratic complexity of self-attention necessitates sequence length reduction before the Transformer layer [39], hence the reduction after the LSTM. This computational optimisation improves efficiency and scalability while focusing on salient features, enhancing generalisation. Inputs are permuted to match the Transformer's expected (sequence length, batch size, detector dimension) format, then restored for downstream processing. Batch size corresponds to the number of events (in the context of this work) processed simultaneously.

The output of the Transformer is then dimensionally flattened and passed to one or more Multilayer Perceptrons (MLPs) depending on the task. An MLP is a class of feedforward artificial neural networks composed of multiple layers of interconnected neurons that use non-linear activation functions to learn complex patterns in data [40]. Each layer transforms the input through weighted connections and activations, enabling the network to approximate virtually any continuous function given sufficient hidden units and proper training.

In the PR model, the output is sent through a two-layer MLP with 256 hidden units and GELU activation, projecting to a final output of three values corresponding to the (x, y, z) coordinates of the interaction. The PRC model has a shared MLP with 256 GELU-activated units with two outputs: one for spatial coordinates and another that produces a single scalar classification logit. The PRCE model uses 3 separate MLPs (heads) – position, classification and energy regression. The position head follows the same GELU-activated structure as above. The classification head also uses GELU, followed by a sigmoid activation to convert the output into a probability. The energy regression head uses a ReLU activation to ensure that the predicted energy remains non-negative, followed by a linear projection to a scalar output. During initial training, the PRCE model would return NaN (Not a Number) loss values, which was likely caused by division by zero. Having a separate MLP for classification with a final sigmoid activation layer alleviated this issue.

2.3 Training and Evaluation

Training, validation, and testing are the three phases of evaluating a machine learning model's performance. Datasets follow typical splits of 70% for training, 15% for validation, and 15% for testing, though ratios may vary based on data size. The model learns patterns during training, the validation tunes hyperparameters and detects overfitting, while the testing provides a final,

unbiased performance assessment [41]. Hyperparameters are predefined configuration settings that control the learning process of a model (e.g., learning rate, batch size, or number of layers), which are set before training and influence how the model learns from the data. Overfitting occurs when a model learns the noise or specific details of the training data too closely, harming its ability to generalise to unseen data.

Validation and testing employ identical technical procedures [41]. The uniform energy distributions and theoretically constrained signal patterns of the datasets can allow validation metrics to reliably approximate test performance without requiring separate evaluation. Preliminary checks showed negligible validation-test performance differences, which motivated the choice of a final data split of 80% training and 20% validation without a dedicated test set. Cross-energy generalisation (e.g., training on 50 eV and testing on 100 eV) was not attempted, as energy-specific training aligns with the intended use case of matching experimental energy bins.

The PR model was trained and evaluated exclusively on NR data, with separate runs performed for each energy (50–100000 eV). In contrast, the PRC and PRCE models were trained on both ER and NR data to enable classification. While PRC was evaluated per energy, PRCE was trained jointly across all energies to optimize cross-energy regression performance. A preliminary PRCE evaluation restricted to energies $\geq$ 500 eV was conducted following the PR model's poor low-energy performance (Section 3.2). This revealed significant classification differences despite unchanged energy and position reconstruction results (Section 3.3). All models were first evaluated on noiseless data, then on noised data. This allowed a direct comparison between idealistic and realistic conditions.

All models were trained using supervised learning, a paradigm where models learn from input-output pairs to approximate the mapping between them [42]. During training, the models iteratively minimised their prediction errors through optimisation of a loss function – a mathematical measure of the discrepancy between predicted and true values [41]. This optimisation process systematically adjusted model parameters to improve performance on the three target tasks. A mean squared error (MSE) loss was applied to the predicted spatial coordinates (x, y, z) to quantify position reconstruction error. It is calculated using the formula

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n} \|\mathbf{r}_i^{\mathrm{true}} - \mathbf{r}_i^{\mathrm{pred}}\|^2,$$

where $\mathbf{r}_i = (x_i, y_i, z_i)$ denotes spatial coordinates and n is the number of events in each batch. For classification, a binary cross-entropy (BCE) loss with logits was used to evaluate the PRC and PRCE models' ER/NR classification performance. This formulation allowed for numerical stability by operating directly on the model's raw logits rather than probability scores. The numerically stable formulation used in PyTorch is

$$\mathcal{L}_{\mathrm{BCE}} = \frac{1}{n}\sum_{i=1}^{n} \left[\max(\hat{y}_i, 0) - \hat{y}_i y_i + \log(1 + e^{-|\hat{y}_i|})\right],$$

where $\hat{y}_i$ are the model's raw logits, $y_i \in \{0, 1\}$ are labels. Energy regression was also supervised using MSE loss, applied to the predicted energy values. For the PRCE model these three losses were computed independently during each training iteration and then summed without explicit weighting, reflecting an assumption of equal task importance and simplifying the training objective. The PR model only used MSE loss, while the PRC model used the sum of both MSE and BCE loss.

The network weights were initialised randomly and optimised through backpropagation, computing gradients via backward loss propagation. The data structure (128 events/file) naturally aligned with batch processing – the PR and PRC models used single-file batches (128)

while the PRCE model employed double-file batches (256) for enhanced parallel computation on GPU hardware. This batch sizing balanced computational efficiency with gradient accuracy, as larger batches leverage parallel processing but may oversmooth gradients, while smaller batches preserve finer data patterns through more frequent updates [43]. Preliminary testing confirmed these sizes optimally negotiated the trade-off between efficient hardware utilisation (favoured by larger batches) and precise gradient estimation (better with smaller batches).

The calculated gradients were used by an optimizer to adjust the weights in a direction that reduced the loss. For this purpose, the AdamW optimizer was employed. AdamW combines adaptive learning rate adjustments with momentum-based updates and a decoupled weight decay mechanism, which improves generalisation and mitigates overfitting [44]. The learning rate determines the size of the steps the optimizer takes when updating the model's weights during training, where a larger rate speeds up convergence but risks overshooting optimal solutions, while a smaller rate ensures stability but may slow progress. Weight decay is a regularisation technique that discourages overly large weights in a neural network by adding a penalty term (proportional to the squared magnitude of the weights) to the loss function, helping to prevent overfitting and improve generalisation.

A relatively high weight decay coefficient of 0.04 was applied to penalise large weights and promote sparsity in parameter values. To further refine the learning process, a cosine annealing scheduler with warm restarts was applied to the learning rate to modulate it in a cyclic manner, decreasing it according to a cosine schedule over 30 epochs before resetting it to its initial value [45]. The initial value was chosen to be 3×10^{-4} after preliminary testing. The minimum learning rate was constrained to 1×10^{-8}. This approach introduced periodic variability in the learning rate, which encouraged exploration of different regions of the loss landscape and helped the optimizer to avoid shallow local minima.

To mitigate the issue of exploding gradients, a gradient clipping threshold of 1.0 was introduced. This restricted the norm of the gradients to remain below the specified value, thereby ensuring stable updates, especially in architectures containing recurrent elements. LSTM layers are particularly susceptible to unstable gradients due to their recurrent structure. Even modest clipping like the one used here protects against instability when backpropagating through long sequences (16,384 samples) or cross-detector attention, without neutering the model's ability to learn [46].

Mixed precision training is a technique that uses both 16-bit and 32-bit floating-point numbers during neural network training (16-bit for faster computation and memory savings), while critical operations (like weight updates) retain 32-bit precision to maintain stability and accuracy [47]. It was implemented using automatic casting of operations to 16-bit floating point where appropriate, combined with dynamic gradient scaling to prevent numerical underflow during backpropagation, reduce memory usage and improve computational speed. A gradient scaler was used to manage this process, scaling the loss before computing gradients and unscaling them prior to gradient clipping and optimizer updates.

At the conclusion of each training epoch, the models were switched to evaluation mode. In this mode, dropout and other stochastic layers were disabled to ensure deterministic outputs. Validation was then conducted using a separate portion of the dataset, and losses were computed without computing gradients. Validation predictions for position, classification, and energy were recorded and saved alongside the corresponding true values for subsequent analysis. One training and one validation epoch together form a full epoch. Due to the time constraints and limited availability of GPU resources the PR model was trained for 600 epochs while the PRC and PRCE models were trained for 300 epochs. If a previous training session had been interrupted, the script was designed to resume from the last completed epoch by reloading model weights, optimizer states, and previously logged loss data. Table 1 summarises the training differences between the models.

The average training and validation losses were recorded per epoch in a structured CSV file. Total losses were recorded for the three models. However, MSE and BCE losses were only recorded for the PRC model's trainings on noised data. Section 3 explains how task performances were assessed in detail.

Model	Batch Size	Loss Funtions	Epochs	NR/ER	Energies
PR	128	MSE	600	NR	One at a time
PRC	128	MSE + BCE	300	Both	One at a time
PRCE	256	MSE + BCE + MSE	300	Both	All at once

Table 1: Training differences between the models.

3 Results

The models' performances were evaluated using performance metrics, which measure how well a model generalises, balancing errors (e.g., false positives/negatives) to avoid overfitting, or in other words they asses predictive reliability and model confidence. Commonly used ones are accuracy, precision, and ROC curves. Sections 3.2, 3.3 and 3.4 include more information on the specific metrics used and task performances, while Section 3.1 focuses on examining the loss plots, which were made as a preliminary study on the models' performances. The position reconstruction and energy regression results from the PRCE limited energy range training on noiseless data were unremarkable and as such are omitted.

3.1 Loss

The losses observed in the figures shown in this section are MSE for position reconstruction, BCE for classification and total losses (sum of two or more losses depending on the model). Minimum values for training and validation losses are shown in red and blue respectively. Loss plots for the PR model were analysed but are omitted here to avoid redundancy, as the PRC model's performance inherently encompasses position reconstruction.

In all figures presented here, minimum values for training and validation losses are shown in red and blue respectively. Figure 10 shows that the unbalanced weighing of the separate loss functions allowed position loss to dominate. Significant overfitting was observed during training, particularly at low energies (50 and 100 eV), with even more pronounced effects in noised data (Figures 11 and 12). Notably, while position loss stagnated for noised 50 eV data, classification loss continued to decrease. Performance improved markedly at $\geq$ 500 eV as seen in Figures 13 and 14, where regularisation effects increased training loss, resulting in a lower validation loss than training loss, particularly at higher energies. Model performance generally scaled with energy. Figure 15 displays the PRCE model's loss trajectories across noiseless and noised data.