22 August 2025
lizzyjonesatx@gmail.com

Benchmarking Supernova Pointing using Machine Learning

Elizabeth Jones, Supervisors: Mathias El Baz, Soniya Samani
CERN, CH-1211 Geneva, Switzerland

Summary

Core-collapse supernovae are some of the most energetic events in the universe, but many aspects of the explosion mechanism and the associated neutrino emission remain poorly understood. Most of a supernova's energy is carried away by neutrinos, making their detection essential for probing the physics at the heart of the collapse and for constraining neutrino properties themselves. The next-generation Hyper-Kamiokande (HK) experiment in Japan will be the world's largest water Cherenkov detector and is expected to collect orders of magnitude more neutrinos than the current Super-Kamiokande (SK) detector during the next nearby core collapse supernova. In anticipation of this event, we simulate supernova events in HK and apply machine-learning algorithms to reconstruct the supernova direction and differentiate between different theoretical emission models. In this report, we compare three architectures—-convolutional neural networks (CNNs), graph neural networks (GNNs), and sparse CNNs—-and evaluate their ability to infer the supernova direction from simulated photomultiplier tube hit patterns. Our results show that the sparse CNN significantly outperforms the other architectures: for a supernova at 8 kpc, it reconstructs the direction to within 2.4° on average assuming no contamination from inverse beta decay events.

Contents

1 Background **3**
1.1 Supernova Neutrinos 3
1.2 Hyper Kamiokande Detector 5
1.3 Objectives for this Project 6

2 Project Pipeline **8**

3 ML Architectures **10**
3.1 Convolutional Neural Network 10
3.2 Graph Neural Network 12
3.3 Sparse Convolutional Neural Network 14
3.4 Global Hyperparameters 15

4 Hyperparameter Optimization **16**
4.1 Results and Analysis 18

5 Conclusion **22**

A Optuna Contour Plot Results **23**

1 Background

Core-collapse supernovae remain one of the most important open problems in astrophysics, with many aspects of their explosion dynamics and neutrino emission still uncertain. In a collapse, the forming neutron star releases nearly all of its gravitational binding energy as neutrinos within a few seconds. Because neutrinos interact only weakly with matter, they escape from deep within the collapsing star and carry information that photons cannot provide. Detecting these neutrinos offers a unique probe of both the explosion mechanism and neutrino behavior under these extreme conditions.

Hyper-Kamiokande (HK), currently under construction in Japan, will soon become the largest water-Cherenkov detector ever built. Its large fiducial volume and dense array of photomultiplier tubes will enable the detection of thousands of neutrinos from an intragalactic core collapse supernova, far surpassing the handful detected from SN 1987A. To take full advantage of HK's capabilities, this project is utilizing a simulation framework for supernova neutrino interactions in the detector and applying machine learning methods to the resulting data. Our goal is to prepare for two complementary tasks: determining the direction to the supernova (supernova pointing) and distinguishing between competing theoretical models of the neutrino emission. By training and benchmarking these algorithms on simulated events, we aim to ensure that, when the next supernova occurs, the HK collaboration will be ready to extract maximal information from the signal.

1.1 Supernova Neutrinos

Core-collapse supernovae are major astrophysical sources of neutrinos. In fact, 95% of the energy released during a supernova is emitted in the form of neutrinos [13].

verify the following is correct The process of a core collapse supernova is illustrated in Fig. 1 and is briefly outlined as follows [13, 10]. In massive stars, successive steps of nuclear fusion within the core creates an onion-like structure with increasingly heavy elements concentrated at the center. During this process, the star remains stable by balancing an inward force from gravity and outward force from electron degeneracy pressure. However, as the core becomes dominated by iron, photodissociation steadily erodes the electron degeneracy pressure and simultaneously creates neutrinos within the core. Once the core mass approaches the Chandrasekhar limit it begins to collapse under its own gravity. Initially, neutrinos are able to stream out freely, but eventually the core becomes so dense that they become trapped, forming a "neutrinosphere." The inner core continues to contract until the strong force halts the implosion; it rebounds, driving an outward shock and leaving behind a hot proto-neutron star.

Mass accretion behind the shockwave causes it to lose energy and stall. For the explosion to succeed, a fraction of the neutrinos emitted from the proto-neutron star must be reabsorbed in so-called "heating" reactions, where charged-current interactions deposit energy and can revive the shock. If there are sufficient amounts of heating reactions, the shockwave is revived and propels outward, resulting in a supernova explosion. If the heating fails, the proto-neutron star collapses to form a black hole.

The supernova neutrino emission from this process can thus be summarized in three steps [13]:

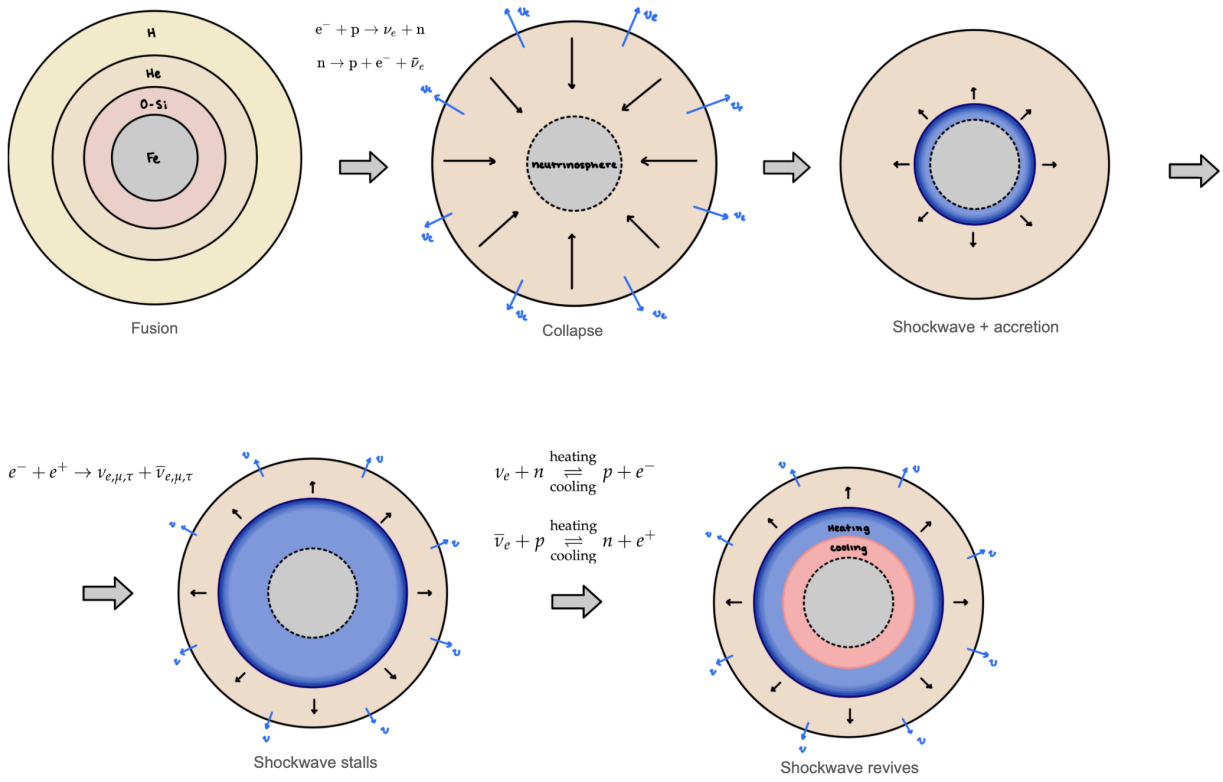


Figure 1: Schematic of the core–collapse supernova mechanism. Top row (left→right): an evolved massive star with an onion-shell structure (H, He, O/Si, Fe); gravitational collapse with rapid electron capture ($e^- + p \rightarrow \nu_e + n$) and neutrino trapping that forms a neutrinosphere; core bounce on the neutrinosphere launches an outward shockwave. Bottom row: matter accretion behind the shockwave causes the shockwave to stall while thermal neutrinos are emitted; neutrino absorption behind the shockwave, $\nu_e + n \rightarrow p + e^-$ and $\bar{\nu}_e + p \rightarrow n + e^+$, deposits energy ("neutrino heating") that can revive the shockwave. The revived shockwave propagates outward, producing a supernova explosion that ultimately leaves behind a cooling protoneutron star.

1. Neutronization burst: A brief, intense flash of electron neutrinos produced when electrons are rapidly captured on protons during the early stages of collapse, converting protons into neutrons.
2. Accretion phase: A period of intense neutrino emission while material continues to fall onto and accumulate around the nascent proto-neutron star.
3. Cooling phase: A longer stage during which the proto-neutron star gradually contracts and cools, emitting neutrinos over a timescale of tens of seconds.

The last supernova that we detected was SN1987A in 1987, and from this even our earth-bound detectors reconstructed only 25 neutrinos [17]. Fig 2 illustrates the energy profile of the neutrinos detected throughout this event, and as is visible from this plot, this information is to sparse to differentiate between many theoretical models.

The next nearby supernova will provide a critical opportunity to test these models, provided modern detectors can capture the signal in sufficient detail.

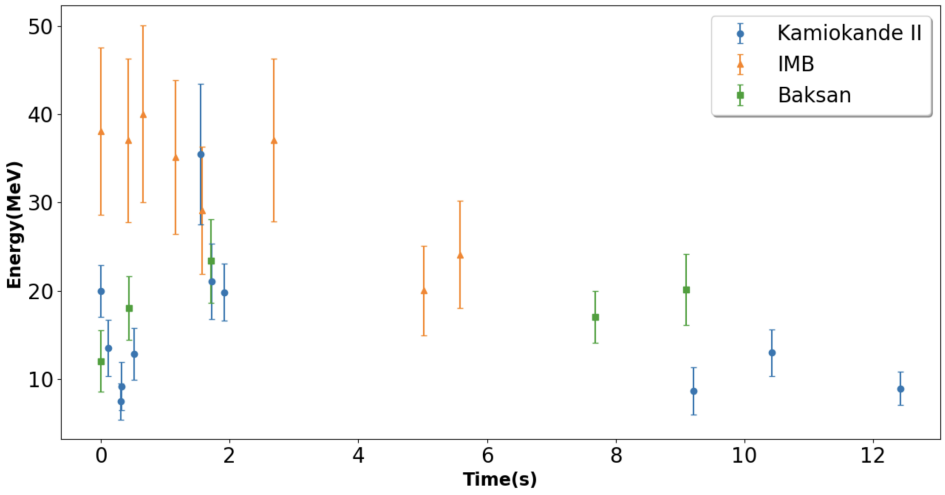


Figure 2: Measured energy of the 25 neutrino candidates from SN 1987A as a function of time. Points show measurements from Kamiokande II (blue), IMB (orange), and Baksan (green). The large uncertainties and lack of data highlight why SN1987A data provides only weak constraints on supernova–neutrino emission models. Image credit: [17].

1.2 Hyper Kamiokande Detector

This study uses simulated data from the Hyper-Kamiokande (HK) detector, a next-generation water Cherenkov detector under construction in Japan. As of this report, the access tunnel and dome excavation have been completed, and the project is projected to finish by 2028 [7].

HK is the successor to the Kamiokande (1983–1996) and Super-Kamiokande (1996–present; see Fig 3) detectors, which were used to detect 11 of the 25 neutrinos from the 1987 supernova event [8]. Like its predecessors, HK uses the water Cherenkov radiation principle to detect neutrinos.

The detector consists of a cylindrical cavity 68 m in diameter and 71m high, buried 650m beneath the Earth's surface. It encloses a fiducial volume of 188 kilotons – eight times that of Super-Kamiokande – thereby greatly enhancing its neutrino detection capabilities [7]. The tank is filled with water, which acts as the crux of the neutrino detection mechanism via Cherenkov radiation.

When neutrinos pass through the tank, most travel unimpeded, but a small fraction will interact with the water molecules. Two primary interaction channels are important for HK: electron scattering and inverse beta decay (IBD). IBD interactions dominate, accounting for roughly 90% of events, while electron scattering comprises the remaining 10% [13]]. Feynman diagrams for these processes are shown in Fig 4.

HK detects these interactions indirectly via Cherenkov radiation. When a charged particle moves through a dielectric medium such as water at a velocity exceeding the speed of light in that medium, its electric field 'disturbs' the polarization field of the surrounding matter. As the polarization field relaxes, it emits photons coherently, forming a shockwave of radiation that propagates as a cone at the characteristic angle θ_C [13, 9].

In HK, neutrino interactions including IBD, electron scattering, and charged current

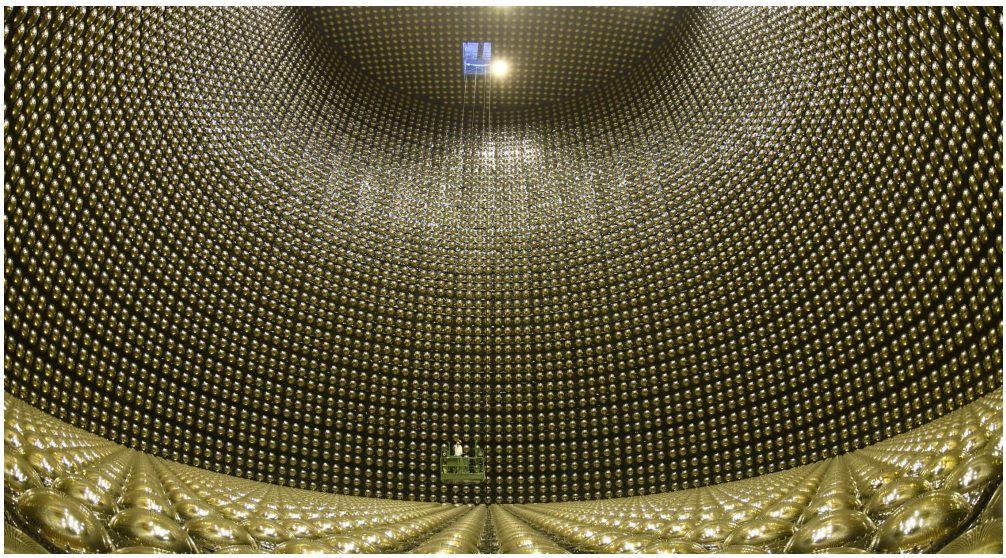


Figure 3: Interior view of the Super-Kamiokande detector in Japan. The cylindrical water tank is lined with thousands of photomultiplier tubes (PMTs), which detect Cherenkov radiation emitted by charged particles produced in neutrino interactions. This detector, operational since 1996, has played a central role in neutrino physics, including the study of solar, atmospheric, and supernova neutrinos. Image credit: [8]

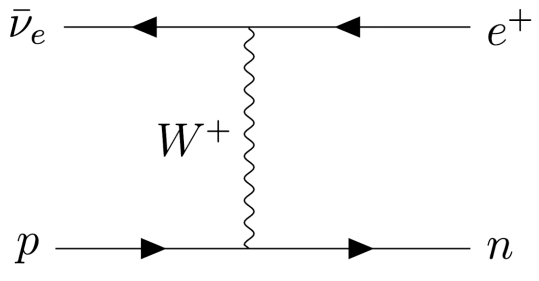
quasi elastic interactions produce charged leptons (electrons or muons). These particles travel faster than light in water and emit a cone of Cherenkov light at the characteristic angle 42° [9, 3]. This cone of emitted photons forms ring-shaped pattern on the walls of the detector, which are lined with thousands of photomultiplier tubes (PMTs). By analyzing the timing and geometry of these rings, we can reconstruct the energy, direction, and type of the original neutrino event.

1.3 Objectives for this Project

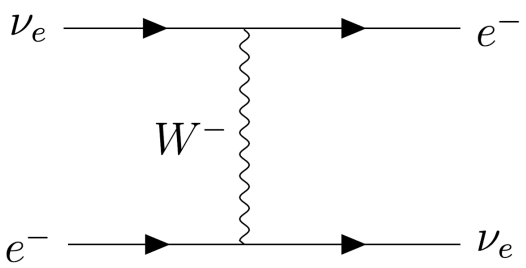
As noted in the introduction, our work targets two complementary goals: determining the direction of the detected supernova (supernova pointing) and distinguishing between different theoretical models of supernova neutrino emission (model discrimination). The pointing task must be performed as an online analysis in real time, as it is intended to quickly provide guidance for telescope observations, whereas model discrimination is an offline analysis that can be undertaken once the full dataset has been recorded.

Supernova Pointing

When a massive star collapses, the neutrino burst precedes the electromagnetic signal by several hours because neutrinos interact only weakly with matter and escape the dense core more easily. If we can quickly infer the direction of this neutrino burst, astronomers can direct their telescopes to the correct region of the sky and capture early photons from the explosion. For this task, the most useful interactions are elastic scattering of neutrinos on



(a) Inverse beta decay (IBD): an electron anti-neutrino interacts with a proton, producing a neutron and a positron ($\bar{\nu}_e + p \to n + e^+$).



(b) Elastic scattering charged current interaction: an electron neutrino scatters off an electron via W^- exchange ($\nu_e + e^- \to \nu_e + e^-$).

Figure 4

electrons. This is because the differential cross section $\frac{d\sigma}{d\Omega}$ for this process depends strongly on the direction of the incoming neutrino:

$$\frac{d\sigma}{d\Omega} = \frac{G_F^2 s}{16\pi^2}\left[g_L^2(1+\cos\theta)^2 + g_R^2(1-\cos\theta)^2\right] \tag{1}$$

where θ is the polar angle from the direction of the incoming neutrino, G_F is the Fermi coupling constant, and s is the mass of the electron [**?**]. The values of g_f and g_r depend on whether the interaction occurs with an electron neutrino ν_e or a tau or muon neutrino $\nu_{\mu,\tau}$:

$$\nu_e e: \quad g_L = \tfrac{1}{2} + \sin^2\theta_W, \qquad g_R = \sin^2\theta_W, \tag{2}$$

$$\nu_{\mu,\tau} e: \quad g_L = -\tfrac{1}{2} + \sin^2\theta_W, \qquad g_R = \sin^2\theta_W. \tag{3}$$

where θ_W is the Weinberg mixing angle [20]. The distribution for each of these cross sections peaks at $\theta = 0$ (see Fig 5), but it is worth noting that the electron neutrino interactions far outnumber the ν_x interactions.

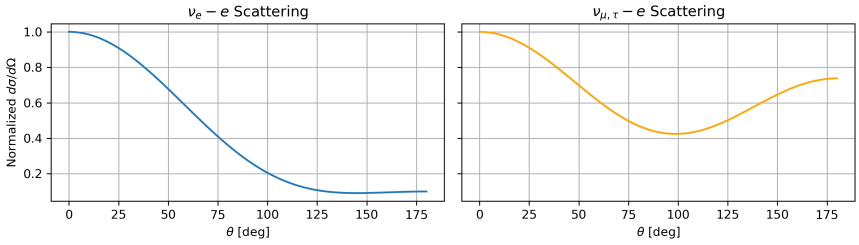


Figure 5: Normalized differential cross sections for elastic scattering of ν_e (left) and $\nu_{\mu,\tau}$ (right) on electrons as a function of scattering angle θ. Both distributions are strongly forward-peaked at $\theta = 0$, reflecting that the direction of the outgoing electron tends to align with the incoming neutrino direction. The overall interaction rate is much higher for ν_e.

In simple terms, the outgoing electron tends to travel in roughly the same direction as the incoming neutrino, as depicted in Fig. 6.

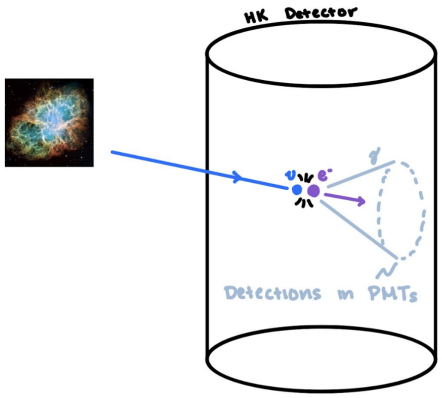


Figure 6: Illustration of supernova pointing in Hyper-Kamiokande. Elastic scattering of neutrinos on electrons produces Cherenkov cones that preserve directional information, enabling reconstruction of the supernova's location.

By analyzing the geometrical distribution of PMT hits on the detector wall, we can infer the directions of the incoming neutrinos and thus locate the (θ, ϕ) direction of the supernova in the sky.

For this project, we use machine-learning techniques to recognize the characteristic ring-shaped pattern on the detector walls. Our study compares the performance of convolutional neural networks (CNNs), graph neural networks (GNNs) and sparse CNNs, and tunes their hyperparameters to optimize the algorithm's accuracy.

Supernova Model Discrimination

Though not the focus of this report, the second objective of this project is an offline analysis that aims to distinguish among competing theoretical models of supernova neutrino emission. By studying the neutrino signal recorded in HK, we can assess which model is most consistent with the observed data. This aspect of the project was led by Miski Nopo under the supervision of Dr. Soniya Samani and Mathias El Baz, and a detailed account of the methodology and results is provided in a separate report.

2 Project Pipeline

The same general workflow can be used for both the model discrimination and pointing tasks with only small differences between the two. This workflow can be broken into two parts: dataset generation and machine learning as outlined in Fig 7. The analysis in this report focuses on the machine learning portion, but for broader context, it is helpful to also give a brief overview of how we generate datasets.

First, we simulate a flux on neutrinos from the first 10 seconds of a supernova given a particular supernova direction (θ and ϕ) and a model that describes the physics of the

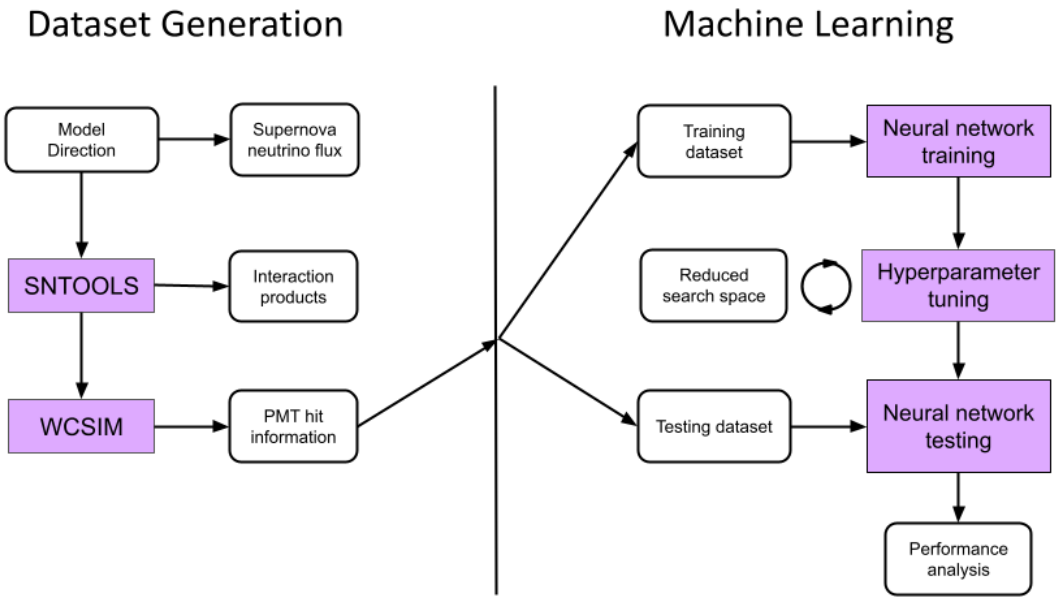


Figure 7: Workflow for the supernova pointing and model discrimination studies. On the left, the dataset generation pipeline begins with the choice of a supernova model and direction, which is passed through a sequence of simulation tools: `SNTOOLs` generates the interaction products from the neutrino flux, `WCSim` simulates the detector response to produce photomultiplier tube (PMT) hits [14, 21]. The output file of this process is in HDF5 format. On the right, the machine learning pipeline uses these datasets for training and testing neural networks. A training dataset is used to train and optimized models through hyperparameter tuning, while the testing dataset is used for evaluating model performance.

neutrino flux. Common models include those developed by Nakazato 2013, Tamborra 2013, Couch 2019, and more [12, 18, 4]. The ultimate goal of the model discrimination task is to differentiate between these models for a fixed direction, whereas for the supernova pointing project, we fix the model and vary the direction. For the pointing task, we choose to use the Nakazato model somewhat arbitrarily because it is a well-established model. Note that the impact of the supernova model on the accuracy is expected to be small because it only influences the energy and time distribution, not the geometrical distribution, of the PMT hits. The only impact that the model choice could have is indirect: if the energy of the neutrinos is higher on average, this results in more inverse beta decay (IBD) reactions. These IBD interactions produce an isotropic detection signature in HK, adding noise to the signal. However, at this stage in the project we do not yet consider IBD reactions, so our choice of model has no impact.

This information is then passed through a series of software (SNTools then WCSim) that converts this information into simulated events in the HK detector [14, 21]. The output of this process is an H5 file containing information about which PMTs detected events, the charge they detected, and the time of the detection.

In addition to the PMT hits simulated by the supernova model, we also add PMT dark

noise to the dataset, which provides an isotropic source of noise. For the supernova pointing task, we consider just the electron scattering channel and have not yet added the inverse beta decay (IBD) interactions, which create an additional isotropic signal. Adding the IBD interactions is a next step currently being implemented by other members of the lab.

In this study, we generate training datasets for 1200 supernovae at distances of both 8kpc and 15kpc. The 1200 supernovae directions follow a cosine distribution in polar angle θ to ensure equal distribution in solid angle. Our testing datasets consist of 400 supernovae, again at angles distributed equally in solid angle for both 8kpc and 15 kpc.

After generating training and testing datasets, we can proceed to training our datasets using a neural network. For the supernova pointing task, we aim to determine which type of neural network is best suited to reconstructing supernova direction, so we consider several types of neural networks as explained in the following section.

3 ML Architectures

This study evaluates three machine learning architectures for supernova pointing: convolutional neural networks (CNNs), graph neural networks (GNNs), and sparse convolutional neural networks (sparse CNNs). The following sections outline their main principles and their specific application in our study.

3.1 Convolutional Neural Network

Convolutional Neural Networks (CNNs) are a class of deep learning models designed to process data with a grid-like structure, such as images. The idea is to use a multi-layer system in order to automatically learn spatial hierarchies of features from the input image [11]. Each convolutional layer applies a set of learnable filters that scan across the input, detecting local patterns such as edges or textures. This step is called the convolution, as it is essentially applying the inner product of the relevant feature with the section of the image. The next layer (n1 channels in Fig 8) is created from these convolutions.

Successive convolutional layers capture increasingly complex patterns while pooling layers compress the spatial dimensions and emphasize important features. After repeating this process several times, the resulting two-dimensional feature maps are flattened into a one-dimensional vector. This vector is then processed by fully connected layers to produce the final predictions. In the pointing problem, this final vector contains two number which correspond to the inferred (θ,ϕ) coordinates of the supernova.

Dataset

To build the CNN dataset, we start by mapping the cylindrical geometry of the Hyper-Kamiokande detector onto a two-dimensional grid (see Fig 9). Each pixel in the resulting image corresponds to a physical location in the detector (i.e. a small group of PMTs). The mapping is constructed using a geometry file that provides the spatial coordinates of each PMT.

We create the dataset for a set of supernovae at various θ and ϕ directions (this is the direction of the incoming neutrino). On average, our simulations yield 700 electron

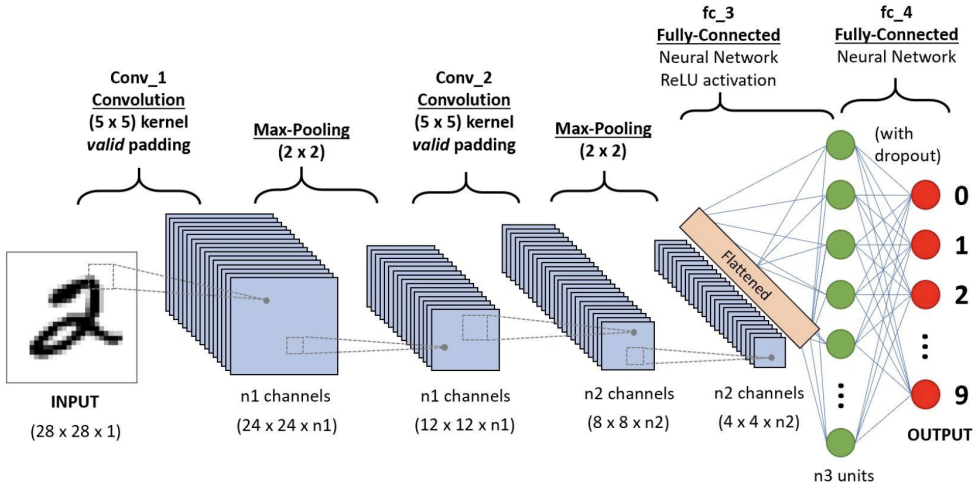


Figure 8: Schematic illustration of a typical convolutional neural network (CNN) architecture. The input image is processed through successive convolutional layers that extract spatial features, with pooling layers reducing dimensionality. The resulting feature maps are flattened and passed to fully connected layers that perform classification or regression. In this example, the CNN classifies handwritten digits, but in our study a similar architecture is adapted for supernova pointing, a regression task where the two outputs represent the supernova direction (θ, ϕ). Image credit: [1].

scattering events for supernovae at 15kpc and 2500 events for those at 8kpc. After creating the datasets using the process outlined in Section 2, we scan through the H5 event files for all the generated supernovae. Each file contains information about which PMTs were hit during a simulated supernova event, how much charge was recorded by each PMT, and the time of each hit.

For the CNN, the final input for each event is a multi-channel image: the height and width correspond to the unwrapped 2-dimensional detector grid, and the number of channels is set by the number of time bins specified (time binning is implemented by binning the charge per PMT/pixel in time).

Model

For this project, we use CNNs with ResNet backbones (ResNet18, ResNet34, ResNet50, ResNet101) to predict the direction of incoming supernova neutrinos from detector images. These models are loaded from the standard torchvision library.

The ResNet backbone consists of a series of convolutional and pooling layers organized into blocks [6]. The number (18, 34, 50, 101) indicates the number of convolutional layers used. These layers are responsible for learning hierarchical spatial features from the detector image, capturing both local and global patterns that are relevant for pointing prediction.

After the backbone has extracted features from the image, the resulting feature vector is flattened. This vector is then passed through a multi-layer perceptron (MLP) head, which consists of fully connected layers with ReLU activations. The final layer of the head produces the output, in our case a vector with two numbers representing the predicted direction (e.g.,

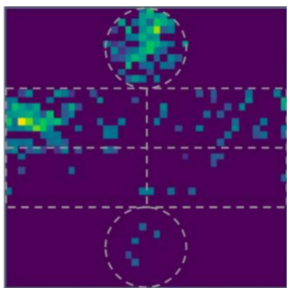


Figure 9: Pictoral representation of the CNN dataset for the HK detector. The detector's cylindrical shape is flattened to two dimensions as show in the image. Each pixel corresponds to a group of PMTs, and the color of the pixel corresponds to the amount of energy those PMTs detected throughout the supernova event.

θ and ϕ angles).

The architecture is highly configurable: we can choose which ResNet variant to use, set the number of input channels, adjust the size and number of hidden layers in the head, and decide whether to use pretrained weights.
Hyperparameters specific to the CNN on the model level:

- **Backbone**: Which ResNet model to use (e.g., resnet18, resnet34, resnet50, resnet101).
- **Hidden Dimensions**: List of the size of each layer in the MLP head.

3.2 Graph Neural Network

Graph neural networks (GNNs) extend convolutional methods to data structured as graphs rather than grids. For this project, when we use a CNN, we have to "unwrap" the cylindrical detector into a flat image. This process can distort the real geometry and create artificial edges, making it harder for the model to learn the true relationships between detector elements.

In contrast, with a GNN, we represent the dataset as a graph, where each node is a single PMT and edges connect nearby elements to each other. This graph representation enables us keep the real cylindrical shape and connections, so the model can learn from the actual layout of the detector.

Whereas a CNN extracts features from an image by detecting local patterns on a fixed grid of pixels, a GNN learns to recognize patterns in data represented as a graph. Instead of a regular grid, the input is a set of nodes connected by edges, and the model learns how information flows along these connections. In the feature-extraction stage of the ML pipeline, GNNs replace the standard convolution with a graph convolution: each node's feature are updated by combining its own information with the one of his neighbors through learnable weights, much like how a CNN aggregates information from nearby pixels.

The following sections outline the specific processes we use and parameters we adjust in the context of this project.

Dataset

For the GNN, the detector is represented directly in 3D using the PMT geometry file, which provides the coordinates of every PMT. Only PMTs that record a signal in an event are included as nodes. Edges are defined using a k-nearest neighbors algorithm, connecting each node to its k closest neighbors in space. The parameter k is tunable (see the list of hyperparameters below).

To reduce computational cost, we downsample the active PMTs by keeping only a fraction ('Keep Fraction') of them. Each PMT is given a score equal to the sum of the charges in its local neighborhood (set by the 'Number of Neighbors' parameter). PMTs are then sampled without replacement, with probability proportional to this score, until the desired fraction is reached. Without this step, including edges between all 20,000 PMTs would make training infeasible. The tradeoff is that downsampling reduces the information available to the model and introduces a bottleneck for the GNN approach.

Each node is assigned features including PMT ID, charge, time, and (x, y, z) position. If time binning is used, the charge is split into several bins, so each node feature becomes a vector over time.

Like with the CNN, the dataset is built from simulated supernova events at different (θ, ϕ) directions. We also include an optional preprocessing step to randomly rotate the azimuthal angle (ϕ) of the event to augment the data and encourage the model to learn rotational invariance. If time information is used, the charge can be split into several time bins, so each node's feature becomes a vector where each entry corresponds to the charge in a specific time window.

Hyperparameters specific to the GNN on the dataset level:

- **k**: The number of nearest neighbors used to construct the graph for each event.
- **Keep Fraction**: fraction of active nodes to keep for each event to reduce the size of the graph
- **Number of Neighbors**: Number of nearest neighbors used to compute local statistics before sub-sampling

Model

For this project, we use a Graph Attention Network (GATNet) as our model for the GNN. GATNet is a type of graph neural network that uses attention mechanisms to decide how much each node should be influenced by its neighbors when updating its features [19].

The model is built from several graph attention layers. Each layer has multiple attention heads, which means the model can learn to focus on different aspects of the local neighborhood at the same time. After each layer, the features of each node are updated by combining its own information with that of its neighbors, weighted by the learned attention scores.

Once the node features have been updated through all the attention layers after message passing (i.e. graph convolution), we use a pooling operation to combine all the node features into a single vector that represents the whole node. The pooling can be a sum, mean, or max.

After pooling, the vector passes through a multi-layer perceptron (MLP) head. Just as with the CNN, this head is a stack of fully connected layers. The final output is a vector (again, in our case two numbers representing the predicted (θ, ϕ) direction of the supernova). Hyperparameters specific to the GNN on the model level:

- **Hidden Dimensions**: Number of filters used in the graph convolution.
- **Number of Layers**: Number of times we repeat the graph convolution + pooling block.
- **Heads**: Number of attention heads per layer.
- **Concat**: Whether to concatenate or average the outputs from the heads.
- **Dropout**: Dropout rate after pooling.
- **Head Dimensions**: List of hidden layer sizes for the head.
- **Pool**: Pooling type ("add", "mean", or "max").

3.3 Sparse Convolutional Neural Network

Sparse convolutional neural networks (Sparse CNNs) are designed to process inputs where only a small fraction of elements are active. This is well suited to our project, where PMTs exist only on the surface of the cylindrical detector and not in the fiducial volume. The sparse dataset structure allows us to consider only the voxels on the cylinder's surface and ignore the interior.

Unlike standard CNNs, which require the cylindrical detector to be unwrapped into a 2D grid, Sparse CNNs operate directly in 3D. The detector volume is divided into small three-dimensional boxes called voxels, with each voxel storing the total charge from all PMTs inside it. A visual representation of this dataset is provided in Fig 10.

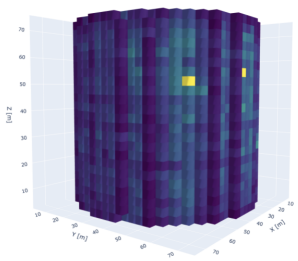


Figure 10: Pictoral representation of the sparse CNN dataset for the HK detector. Each voxel is represented by a small cube shape containing a group of PMTs. The color of the voxel indicates the amount of charge that the PMTs within the voxel receive over the entire supernova event.

What makes the model "sparse" is that it only stores and computes features for voxels that actually contain data (ie nonzero charge) [5]. Empty voxels are ignored, which reduces the memory and computation required. This efficiency means we can use deeper networks or larger detector volumes without running into resource limits.

Dataset

Like with the CNN and GNN architectures, the sparse CNN dataset is created by mapping PMT hits onto a 3D voxel grid using a detector geometry file. Each active voxel stores a vector of charge values across time bins, while empty voxels are omitted. This approach preserves the three-dimensional structure of the detector and allowing the model to extract physical information from supernova events without flattening or distorting the detector geometry.
Hyperparameters for the sparse CNN dataset on the dataset level:

- **Voxel Size**: The size of each voxel in millimeters (e.g., (5, 5, 5)). Controls the spatial resolution of the grid.
- **Coordinate Range**: The physical range of the detector volume covered by the voxel grid (e.g., (-50, -50, -50, 50, 50, 50)).
- **Max Points per Voxel**: Maximum number of PMT hits allowed in a single voxel.
- **Max voxels**: The maximum number of active voxels per event.

Model

The sparse CNN model processes detector data using a series of sparse 3D convolutional blocks. Each block applies submanifold convolutions, batch normalization, and ReLU activations [5]. As with the regular CNN, downsampling is performed between these blocks. This structure enables the model to capture both local and global features.

After the convolutional steps, features from all active voxels are aggregated using a global pooling operation (either average or max pooling) resulting in a single feature vector for each event. This vector is then passed through an MLP head just as with the regular CNN and GNN architectures.
Hyperparameters specific to Sparse CNN on the model level:

- **Channels**: The number and size of feature maps in each sparse convolutional layer.
- **Hidden dimensions**: The sizes of the hidden layers in the fully connected head.
- **Model**: The type of backbone used during feature extraction. Options included SparseUNet, which first increased layer size before subsequently decreasing back to the original value as well as SparseCNN, which simply increases the number of feature maps monotonically.

3.4 Global Hyperparameters

We also consider several aspects of the neural network that are common to all three architectures. These hyperparameters are as follows.
Dataset level:

- **Model**: supernova model used to generate simulated events (e.g. Nakazato, Tamborra, etc). We use Nakazato only in this study.

- **Distance**: The distance (in kiloparsecs) of the supernova from the detector.
- **Number of time bins**: The number of time bins used to split the detected charge at each PMT.

Model level:

- **In Dimensions**: Number of input features per node (e.g., charge for each time bin, position).
- **Out Dimensions**: Size of output vector (always 2 for us for θ and ϕ)
- **Activation**: Activation function (e.g., ReLU).

Training Level: For the model training, we employ a scheduled varying learning rate over the course of training. The values follow the OneCycleLR scheduling pattern from PyTorch, which starts at an initial value, ramps to a maximum value by a fixed fraction of the total epochs, and then gradually decreases [15]. This approach allowing faster convergence to the minimum loss while preventing overfitting [16]. These hyperparameters, along with other global parameters used in training, are summarized below.

- **Batch Size**: Number of training examples processed before the model parameters are updated.
- **Epochs**: Total number of epochs to run for.
- **Validation Split**: Fraction of the training data reserved for validation during training.
- **Maximum Learning Rate**: The maximum value reached by the learning rate during training.
- **Percent Start**: The percentage of the cycle (in number of steps) spent increasing the learning rate to its maximum value.
- **Weight Decay**: Regularization term applied to model parameters to prevent overfitting by penalizing large weights.

4 Hyperparameter Optimization

After a first round of manual tuning to narrow down reasonable ranges, we used the Optuna library to carry out automated hyperparameter searches [2]. We distributed trials on the batch system, with each job running a fixed number of Optuna trials. Each trial trained a model with a unique set of sampled hyperparameters and returned the validation loss, which Optuna used to update its sampling strategy. In practice, we ran 3–5 jobs in parallel per "stage," then merged the results before starting the next stage. This iterative process let us progressively narrow the search space around promising regions.

Each round produced an Optuna database containing the tried hyperparameter sets and their validation losses. From this, we extracted the best-performing configuration to use in final training. To guide decisions about how to adjust the search space between rounds, we generated two types of diagnostic plots: contour plots and hyperparameter importance plots.

Contour Plot

The contour plots show how pairs of hyperparameters affect the validation loss. For each pair, Optuna interpolates between trials to produce a 2D contour map, where the color of the contours indicates the validation loss. Fig 11 gives an example from the first CNN optimization round, showing the results for hyperparameters percent start vs. maximum learning rate. Each gray point corresponds to a trial, with the five best runs marked in red. Darker regions correspond to lower loss, making it easy to spot where the best configurations are concentrated.

Based on results from this plot, we can determine how to further restrict or shift the search space in future optimization rounds. For example, in the subsequent hyperparameter optimization for Fig 11, we decreased the search space for these parameters to encompass only the area with the most successful trials (i.e. the top left corner of the plot).

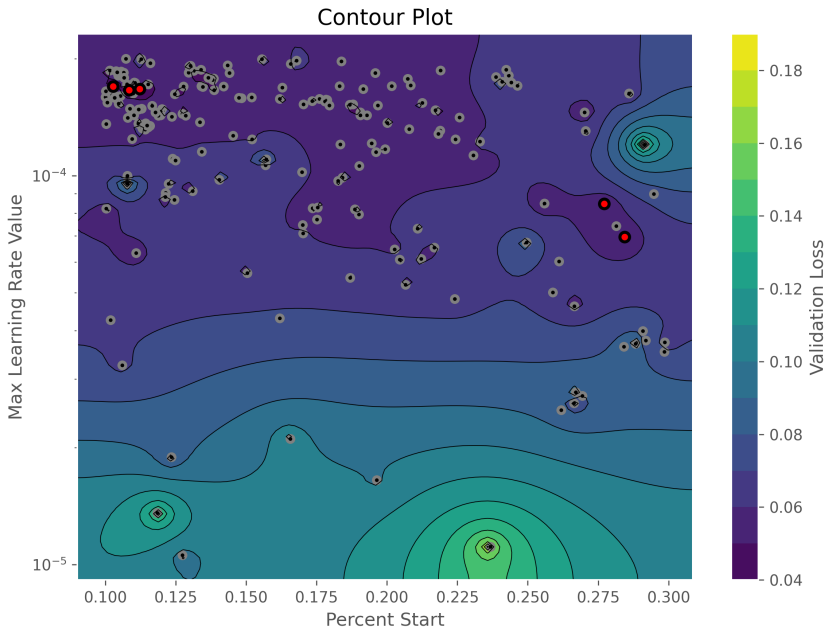


Figure 11: Example contour plot from hyperparameter optimization. The axes show two hyperparameters—percent start (x-axis) and maximum learning rate (y-axis)—with color indicating the associated validation loss from machine learning trials on the training dataset. Grey points represent the trials run, and red points mark the five best-performing configurations. Dark purple regions correspond to lower loss, highlighting promising areas of the search space for subsequent optimization.

At the end of each round of hyperparameter tuning, we generated a large plot including contour plots for each pair of hyperparameters. The final versions for each architecture are included in Appendix A.

Hyperparameter Importances Plot

To help narrow the search space, we also looked at hyperparameter importances. These plots measure how strongly each parameter correlates with validation loss across all completed trials. We calculate this using the Pearson correlation coefficient between each parameter and the validation loss. The importance scores are shown as a horizontal bar chart (Fig. 12). This makes it clear which hyperparameters are worth tuning and which have little effect, so we can focus on the on the most influential parameters in subsequent searches.

For the GNN runs, a useful example was the parameter k, the number of edges per node. Because our GAT model uses multiple attention heads, increasing k rapidly increases the number of connections and the memory required. This often caused our trainings to crash on the GPU. The importances plot (Fig 12) showed that k had little correlation with validation loss, so we reduced it to lower the computational cost without worrying that it would greatly affect model performance.

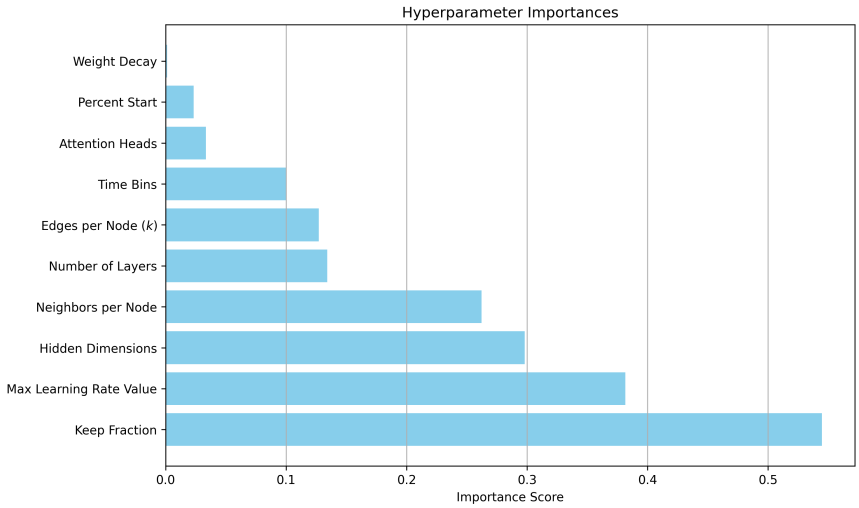


Figure 12: Hyperparameter importance scores from Optuna analysis for the GNN architecture. Each bar represents the relative influence of a given hyperparameter on validation loss across all trials. Parameters such as keep fraction, maximum learning rate, and hidden dimensions had the largest impact on performance, whereas weight decay and percent start contributed minimally.

4.1 Results and Analysis

We carried out two rounds of Optuna hyperparameter tuning for each of the three architectures (CNN, GNN, and Sparse CNN). At the end of this process, we selected the three best-performing trials for each architecture. The corresponding hyperparameter values are listed in Tables 1–3. Reported validation losses are based on the validation dataset from training; the test dataset is reserved for the analysis in the following sections.

Table 1: Best hyperparameter configurations for CNN

Parameter	**Best Trial**	**2nd Trial**	**3rd Trial**
Model backbone	resnet18	resnet18	resnet18
Hidden dimensions	[512]	[1024]	[1024]
Learning rate	1.67e-4	8.47e-5	6.97e-5
Weight decay	5.92e-4	6.51e-4	8.25e-4
Time bins	28	26	27
Percent start	0.112	0.277	0.284
Validation Loss	0.015132	0.017391	0.019888

Table 2: Best hyperparameter configurations for GNN

Parameter	**Best Trial**	**2nd Trial**	**3rd Trial**
Head dimensions	[98, 64]	[98, 64]	[98, 64]
Heads	3	4	3
Hidden dimensions	118	94	114
Learning rate	3.12e-4	4.90e-4	2.41e-4
Weight decay	2.28e-4	6.52e-5	3.76e-5
Time bins	3	4	3
Edges per node (k)	35	35	50
Keep fraction	0.286	0.298	0.311
Number of neighbors	26	16	12
Percent start	0.119	0.100	0.219
Validation Loss	0.012125	0.012785	0.015938

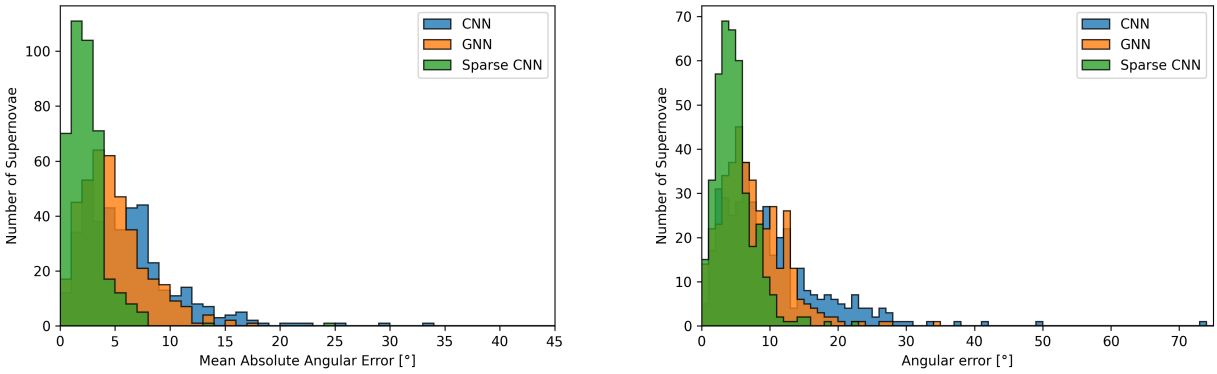
Table 3: Best hyperparameter configurations for Sparse CNN

Parameter	**Best Trial**	**2nd Trial**	**3rd Trial**
Model type	SparseUNet	SparseUNet	SparseUNet
Hidden dimensions	[512]	[512, 256]	[512, 256]
Learning rate	1.26e-3	9.19e-4	7.53e-4
Weight decay	2.21e-7	2.49e-6	1.93e-7
Time bins	3	3	3
Voxel size	[2.7,2.7,2.7]	[2.6,2.6,2.6]	[3.1,3.1,3.1]
Percent start	0.231	0.0816	0.0832
Validation Loss	0.002785	0.003472	0.003924

As is evident from the validation loss results of these trainings, the sparse CNN far outperforms the regular CNN and the GNN architectures. In order to more clearly see this trend and to analyze results, we fed these hyperparameters into a newly generated testing dataset too see how each model performed. These results are analyzed in the following sections.

Comparing Predicted Angular Error for each Architecture

To measure the performance of each architecture, we generated two datasets, one at 8 kpc and one at 15 kpc. Each dataset contains 400 supernova events with different θ and ϕ directions. Figure 13 compares the distributions of angular reconstruction errors for the three architectures at 8 kpc and 15 kpc.



(a) Distance 8kpc

(b) Distance 15kpc

Figure 13: Distribution of mean absolute angular errors for CNN, GNN, and sparse CNN. For both 8kpc and 15kpc, the sparse CNN achieves the lowest errors, while the standard CNN performs worst.

At 8 kpc (Fig. 13a), all three models reconstruct most supernovae within about 10°. The sparse CNN performs the best: most of its predictions are below 5°, with a mean error of 2.43°, and there are fewer large-error events compared to the CNN and GNN. The GNN does better than the CNN but still has a broader spread, especially in the 5–15° range.

At 15 kpc (Fig. 13b), the performance of all models is worse. This is expected, since fewer neutrino interactions are detected at larger distances. The sparse CNN remains the most accurate, with most events reconstructed within 10°. The CNN again shows the widest spread, with many events above 20° and some even larger than 50°. Values for the mean absolute angular errors in included in Table 4.

Table 4: Summary of Performance of each Architecture

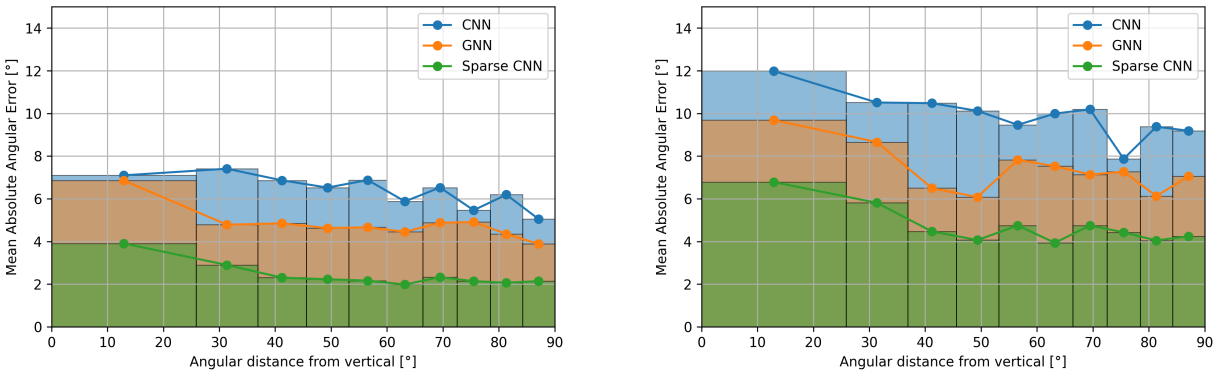
Architecture	Distance [kpc]	Mean Abs Angular Error [°]	Inference Time per SN (s)
CNN	8	6.39	0.08
	15	9.98	0.04
GNN	8	4.83	0.29
	15	7.47	0.23
Sparse CNN	8	2.43	0.07
Sparse CNN	15	4.81	0.06

Overall, the results show two trends. First, reconstruction accuracy decreases with distance. Second, the sparse CNN consistently outperforms the other models, followed by the GNN, with the CNN performing the worst.

For each of these architectures, we also determined the time taken to generate the prediction. This metric is important, as the initial neutrino burst during a supernova is very quick, and since supernova pointing is an online analysis, it must also be performed quickly. Results of mean absolute angular error and inference time are summarized in Table 4.

Dependence on Polar Angle θ

Because the HK detector is cylindrical and therefore not rotationally invariant in the polar angle θ (the angle from the vertical direction in the tank), performance should vary as a function of this angle θ. To understand this trend, we plotted the performance of the best three trials for each architecture as a function of θ in Fig 14.



(a) Distance 8kpc

(b) Distance 15kpc

Figure 14: Mean absolute angular error as a function of angular distance from vertical for supernovae at 8kpc and 15kpc. The downward trend indicates that the models perform worst when detections occur near the top/bottom of the tank and best when the detections occur on the barrel.

All three architectures show a downward trend of accuracy vs. angle θ from the tank's vertical direction. This indicates that the model is able to predict the direction of the supernova most accurately at larger angles θ when most of the hits are concentrated in the barrel of the detector tank. Geometrically this makes sense, as the tank has less area at its top and bottom surfaces than it does on the barrel section, and therefore the Cherenkov radiation cone is less likely to make a clean, easily detectable signature at the PMTs.

Dependence on Distance

Finally, because the 8kpc dataset is closesr to the detector, it includes more detections per supernova (the simulations yield, on average, 2500 electron scattering events for supernovae at 8kpc and 700 for supernovae at 15kpc). As a result, we expect the algorithm to more accurately reconstruct the direction of the supernova. In Figure 15, we have plotted the performance of each algorithm as a function of the distance.

As this figure illustrates, the 8kpc indeed outperforms the 15kpc dataset, as expected. To better understand the dependence of model performance on supernova distance, it would

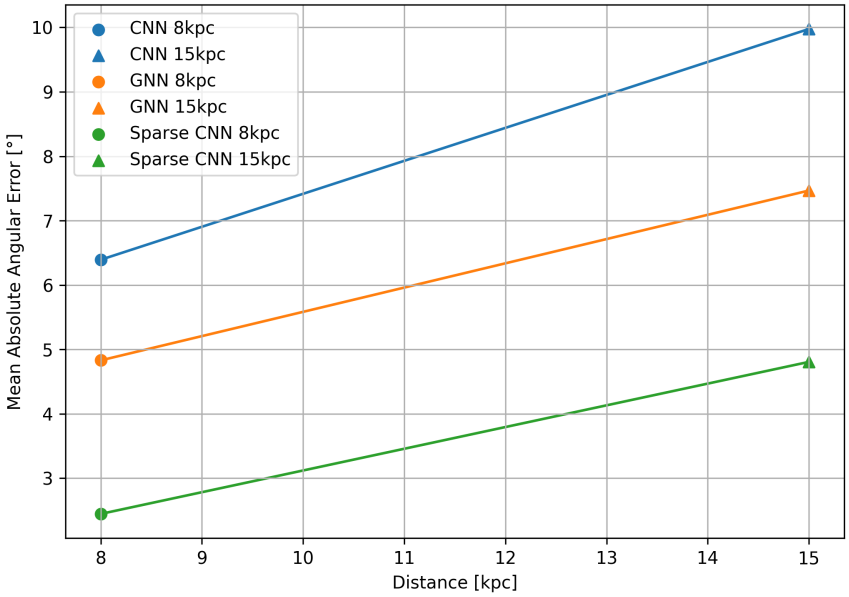


Figure 15: Mean absolute angular error as a function of supernova distance for CNN, GNN, and sparse CNN models. Performance degrades with increasing distance for all methods, but the sparse CNN consistently achieves the lowest errors, followed by the GNN, while the standard CNN performs worst.

be instructive to generate additional datasets at different distances and add their results to this plot.

5 Conclusion

In this project, we compared three machine-learning architectures for reconstructing the direction of core-collapse supernovae from simulated Hyper-Kamiokande neutrino data. Supernova events were simulated at distances of 8 kpc and 15 kpc, using only the electron-scattering channel with PMT dark noise included. CNN, GNN, and sparse CNN models were trained and tuned through two rounds of Optuna hyperparameter optimization. Among these, the sparse CNN achieved the best pointing accuracy, with mean angular errors of about 2.4° at 8 kpc and 4.8° at 15 kpc. The GNN gave intermediate performance ($\sim$ 4.8° and $\sim$ 7.5°), while the standard CNN performed worst ($\sim$ 6.4° and $\sim$ 10.0°). Performance was also found to improve for events at larger polar angles compared to small polar angles.

The next stage of this project will add inverse beta decay (IBD) interactions to the simulations. Although IBD dominates the events in Hyper-Kamiokande, its signal is isotropic and provides no directional information. Including IBD will increase statistics but also introduce an isotropic background that reduces pointing precision. To address this, we plan to develop a channel-tagging or event clustering algorithm to reduce the contamination from IBD events. Combining such tagging with the sparse CNN pointing model will yield a more realistic estimate of HK's sensitivity and help prepare the collaboration for the next supernova event.

A Optuna Contour Plot Results

This appendix provides pairwise contour and histogram plots from hyperparameter optimization for the second and final CNN, GNN, and sparse CNN hyperparameter optimization runs using Optuna. Each off-diagonal subplot shows the relationship between two sampled hyperparameters, with colors indicating validation loss values. Grey points mark individual trials, while red points highlight the five best-performing configurations. The diagonal panels display distributions for each hyperparameter, where the bars are color-coded to match the average validation loss values for all trials within the corresponding bin. These plots help visualize correlations between hyperparameters and regions of the hyperparameter space leading to improved performance.

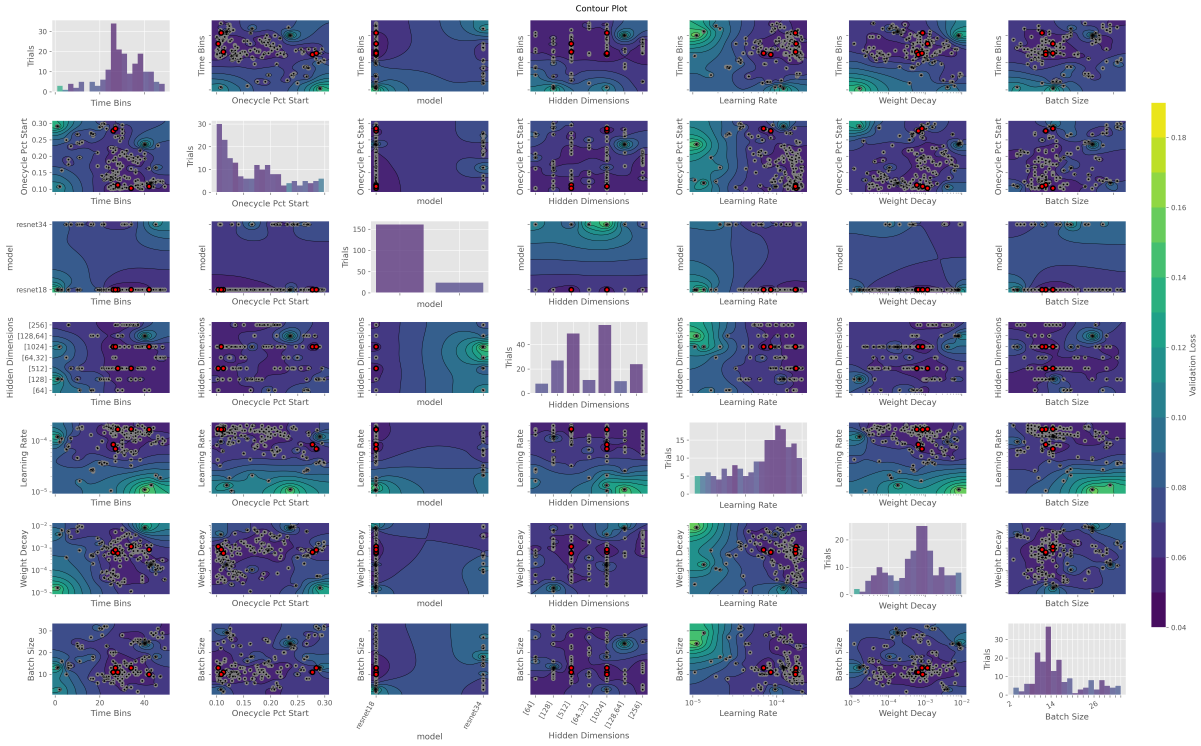


Figure 16: Pairwise contour and histogram plots from second Optuna tuning for the CNN architecture.

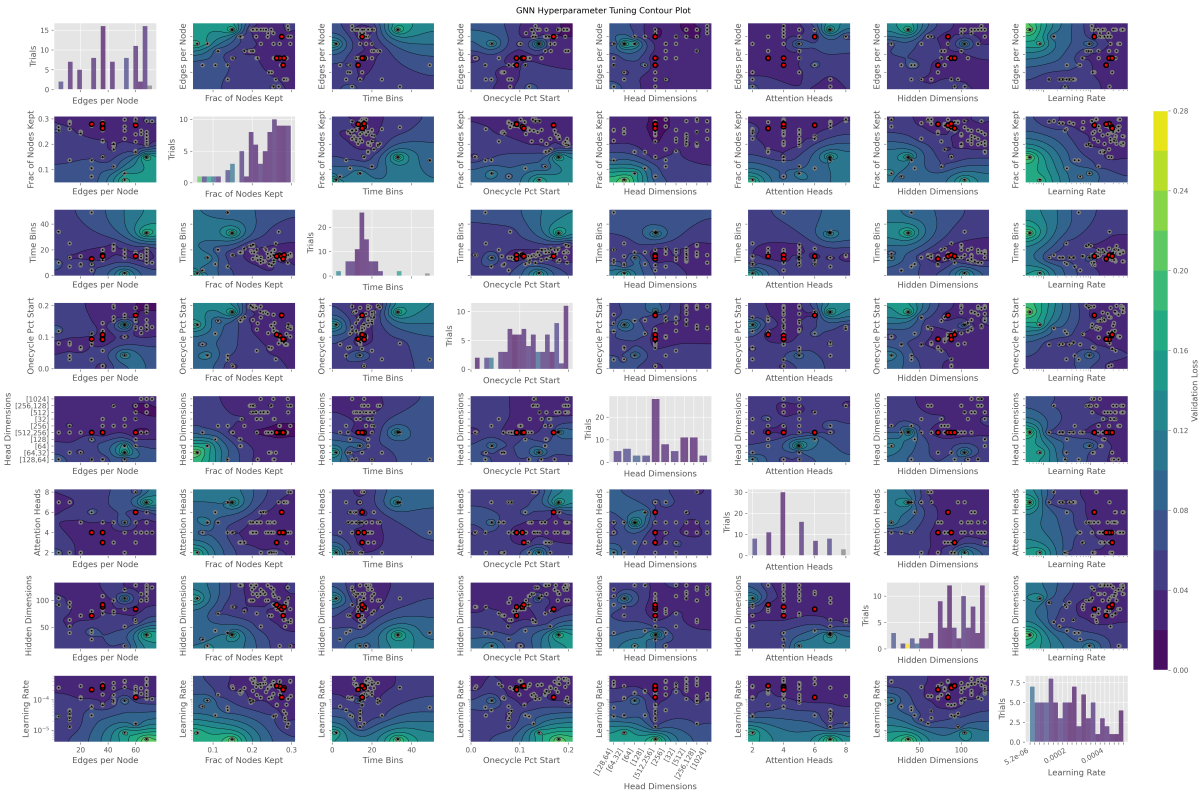


Figure 17: Pairwise contour and histogram plots from second Optuna tuning for the GNN architecture.

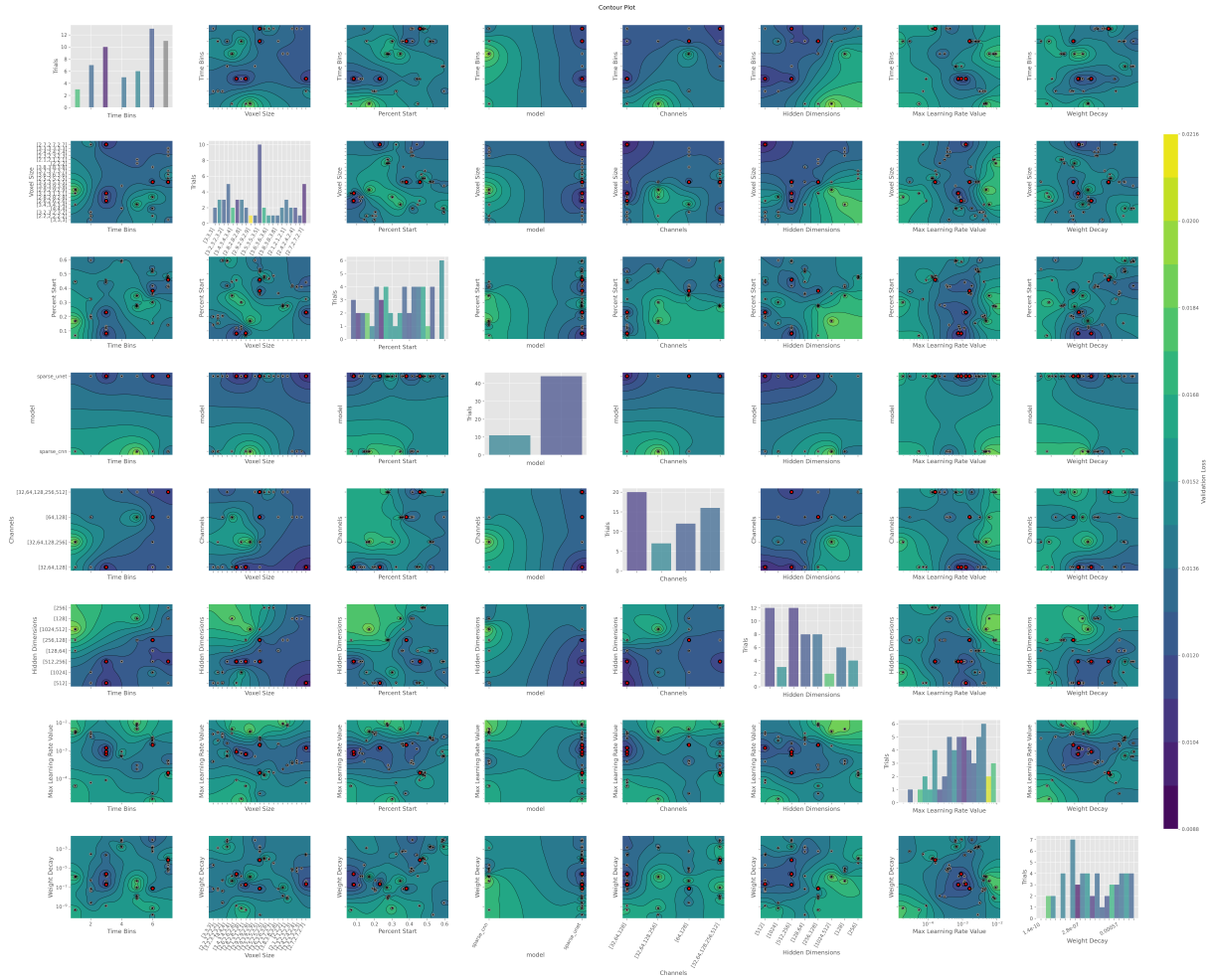


Figure 18: Pairwise contour and histogram plots from second Optuna tuning for the sparse CNN architecture.

References

[1] Nagnath Aherwadi, Usha Mittal, Jimmy Singla, N. Z. Jhanjhi, Abdulsalam Yassine, and M. Shamim Hossain. Prediction of fruit maturity, quality, and its life using deep learning algorithms. *Electronics*, 11(24), 2022. ISSN 2079-9292. doi: 10.3390/electronics11244100. URL https://www.mdpi.com/2079-9292/11/24/4100.

[2] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*, pages 2623–2631. ACM, 2019. doi: 10.1145/3292500.3330701.

[3] Kirk Bays, Takashi Iida, K. Abe, Y. Hayato, K. Iyogi, J. Kameda, Y. Koshio, Lluis Marti Magro, M. Miura, Shigetaka Moriyama, Masayuki Nakahata, S. Nakayama, Yoshihisa Obayashi, H. Sekiya, M. Shiozawa, Y. Suzuki, Atsushi Takeda, Y. Takenaga, K. Ueno, and R. Wilkes. Supernova relic neutrino search at super-kamiokande. *Physical Review D*, 85, 11 2011. doi: 10.1103/PhysRevD.85.052007.

[4] Couch. Simulating turbulence-aided neutrino-driven core-collapse supernova explosions in one dimension. *arXiv preprint arXiv:1902.01340*, 2019. URL https://arxiv.org/abs/1902.01340. Revised November 2019.

[5] Benjamin Graham and Laurens van der Maaten. Submanifold sparse convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9224–9232, 2017. URL https://arxiv.org/abs/1706.01307.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[7] The University of Tokyo Institute for Cosmic Ray Research. Overview of the hyper-kamiokande project. https://www-sk.icrr.u-tokyo.ac.jp/en/hk/about/outline/, . Accessed on August 21, 2025.

[8] The University of Tokyo Institute for Cosmic Ray Research. Overview of the super-kamiokande experiment. https://www-sk.icrr.u-tokyo.ac.jp/en/sk/about/outline/, . Accessed: 2025-08-18.

[9] John David Jackson. *Classical Electrodynamics*. John Wiley & Sons, New York, 3 edition, 1998. ISBN 978-0-471-30932-1.

[10] H.-Thomas Janka. Explosion mechanisms of core-collapse supernovae. *Annual Review of Nuclear and Particle Science*, 62(1):407–451, November 2012. doi: 10.1146/annurev-nucl-102711-094901.

[11] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[12] Nakazato. Supernova neutrino light curves and spectra for various progenitor stars: From core collapse to proto-neutron star cooling. *Astrophysical Journal Supplement Series*, 205(1):2, 2013. doi: 10.1088/0067-0049/205/1/2.

[13] Soniya Samani. *Constraining the diffuse supernova neutrino background in Super-Kamiokande with gadolinium and precision measurements of photosensors for Hyper-Kamiokande*. Phd thesis, University of Oxford, 2024. Available via Oxford Research Archive.

[14] Kate Scholberg. SNOwGLoBES / SNTools: Supernova Neutrino Event Rate Calculator, 2012. URL https://github.com/SNOwGLoBES/snowglobes. Accessed: 2025-08-21.

[15] Leslie N. Smith. A disciplined approach to neural network hyper-parameters: Part 1 – learning rate, batch size, momentum, and weight decay. *arXiv preprint arXiv:1803.09820*, 2018. URL https://arxiv.org/abs/1803.09820.

[16] Leslie N. Smith and Nicholay Topin. Super-convergence: Very fast training of neural networks using large learning rates. *arXiv preprint arXiv:1708.07120*, 2017. URL https://arxiv.org/abs/1708.07120. Revised May 17, 2018.

[17] Maurizio Spurio. *Probes of Multimessenger Astrophysics*. 2018. doi: 10.1007/978-3-319-96854-4.

[18] Tamborra. Self-sustained asymmetry of lepton-number emission: A new phenomenon during the supernova shock-accretion phase in three dimensions. *Physical Review Letters*, 111(12):121104, 2013. doi: 10.1103/PhysRevLett.111.121104.

[19] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations (ICLR)*, 2018. URL https://arxiv.org/abs/1710.10903.

[20] P. Vogel and J. Engel. Neutrino electromagnetic form-factors. *Phys. Rev. D*, 39:3378–3383, 1989. doi: 10.1103/PhysRevD.39.3378.

[21] Chris Walter. WCSim: A Geant4-based Water Cherenkov Detector Simulation, 2012. URL https://github.com/WCSim/WCSim. Accessed: 2025-08-21.