|  |  |  |
| --- | --- | --- |
| **Audio Classification** | Banded FFT | * Train a multi-layer perceptron on the temperature-, pressure-, and position-corrected frequency domain audio information which was used to train a similar network in the original PICO-60 paper. * Results closely replicated those observed in the PICO-60 paper. A strong correlation was seen between the acoustic parameter and the neural network’s score. It was generally unsuccessful in discriminating between alpha particles and unusually loud neutron outliers. |
|  | Raw Waveform | * Train a neural network on the audio waveform recorded at the time of the event. This includes multiple architectures; a few convolutional layers followed by a dense network, as well as a fully convolutional architecture with only a single small dense layer at the end. * The goal is to discriminate between low background data and neutron calibration runs (starting with AmBe and later including other sources). * Modify the number of layers and the hyperparameters of the individual layers to improve the performance of the network. * Significant success was obtained using a fully convolutional network with only AmBe source data. Inclusion of other data resulted in mediocre validation performance and significant overfitting. |
|  | Chirp Prediction | * Unknown audio events known as “chirps” are present in parts of the training data. Discover a meaningful predictor of these events in the available data, and use it to train a neural network (potentially using semi-supervised learning) to predict their presence. Analyze their characteristics and ideally determine their source. |
|  | Wall Events | * The acoustics of events located near walls and events located in the middle of the vessel are notably different. Analyze their waveforms and Fourier transforms to determine how they differ, and potentially train a neural network to distinguish between them or predict their distance from the wall in a continuous way. |
| **Image Classification** | Convolutional Neural Network | * Train a 2D convolutional neural network to distinguish between alpha particles and neutrons based on small windows cut out at the approximate locations of the bubbles in the images. * The idea is to cue on the shape of the bubble and the visible shockwaves around the bubbles, which may be correlated with certain aspects of the audio and potentially the identity of the source. * Initial testing was notably unsuccessful; the network was able to obtain high training accuracy, but had a validation accuracy of 50%. |
| **Semi-Supervised Learning** | Iterative Cluster Nucleation | * Start with a small set of training examples for each category that we are very sure about. Train a neural network on this data (preferably a small one including lots of regularization so it won't overfit). * Then, run inference on the remainder of the dataset which is presently unclassified. Take the ones which the network is most confident about and add them to the corresponding training datasets. * Train another network (possibly with less regularization or a more complex architecture because the dataset is larger and provides more information) on the new labeled dataset. * Run inference again, and repeat until the entire dataset has been classified. |
|  | Gravitational and Distribution-Biased Gradients, and Probabilistic Ground Truths | * Start by training a conventional model with a very low learning rate. However, modify the learning rate for each example depending on how confident we are that its label is correct. This means the most-confident ones result in large steps to the weights, while the most questionable examples result in very small changes. * Later on in the training process, introduce a "gravity" term to the gradient calculations which pulls examples towards either 0 or 1, whichever they are closest to, and with the strength of this pull depending on how close they are. * In addition, it is practical to add a new term which slightly offsets the gradient for the output layer such that it will push all examples to equalize the overall distribution to the expected distribution. |