Project Group F: Measuring Chirps and Energy

M. Binns
Department of Physics and Astronomy, University College London, Gower Street, WC1E 6BT

March 2024
Word Count (excluding references): 4872

Contents

Abstract –**This study explores the applications of deep learning techniques to ascertain Cyclotron Radiation Emission Spectroscopy (CRES) data for measuring neutrino mass, a fundamental particle. Neutrinos, despite being among the most abundant particles, have small masses that are difficult to measure directly. The CRES method, which uses the energy of electrons resulting from beta decay in a magnetic field, is a promising method of measuring the mass. Within this project multiple deep learning models were developed and evaluated, including Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) for spectrogram image and time series analysis, to detect the start time and frequency of chirps indicative of neutrino interactions. Our results demonstrate the potential of deep learning in significantly enhancing the accuracy and efficiency of neutrino mass measurements from CRES data. This work not only contributes to the field of neutrino physics but also showcases the versatility and power of deep learning in tackling complex and high-dimensional data in scientific research.**

I Introduction

I.1 Neutrinos

Neutrinos, often regarded as the ghosts of the particle world due to their weak interactions with matter, hold critical insights into fundamental physics and the early universe. Despite being immensely abundant, their absolute mass remains one of the most significant unanswered questions in particle physics. Understanding the mass of neutrinos is not merely an academic pursuit but bears implications for the standard model of particle physics, the dynamics of the early universe, and the mechanisms underlying supernova explosions.

The existence of neutrino oscillations, a phenomenon where neutrinos change flavors, has proven that neutrinos possess mass. However, the challenge in measuring the neutrino mass stems from its weak interaction with matter. Traditionally, experiments have relied on observing the beta decay of tritium, where the kinematics of the emitted electron are influenced by the neutrino mass but these techniques face limitations in sensitivity and precision.

Project F introduces a new approach by using Cyclotron Radiation Emission Spectroscopy (CRES). This technique involves situating an electron, having a specific kinetic energy T, within a magnetic field, B. Under the influence of this field, the electron undergoes cyclotron motion along the magnetic field lines, emitting cyclotron radiation during this motion.

The angular cyclotron frequency, ω_C, of an electron, considering its kinetic energy T and mass m_e, in a magnetic field B is given by:

$$\omega_C = \frac{eB}{\gamma m_e} = \frac{eB}{m_e + \frac{T}{c^2}} \tag{1}$$

Where e represents the charge of the electron, γ denotes the Lorentz factor, and c is the speed of light [1]. The cyclotron frequency, f_C, is subsequently related to the angular frequency by:

$$f_C = \frac{\omega_C}{2\pi} \tag{2}$$

As the electron spirals, its acceleration leads to the radiation's emission. By measuring the frequency of this emitted radiation, the electron's energy can be inferred. This methodology paves a novel way to ascertain the neutrino mass by scrutinising the energy spectrum of electrons emitted from beta decay, focusing on the nuanced effects exerted by the neutrino's mass.

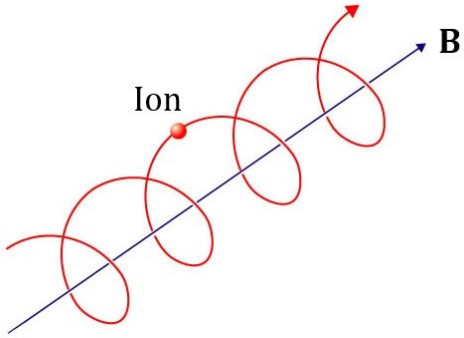


Figure 1: A diagram of cyclotron motion [1]

The advantage of using CRES lies in its unparalleled resolution and sensitivity, allowing for the detailed mapping of the electron's energy spectrum with minimal disturbance to the system. This technique has the potential to significantly lower the uncertainty in neutrino mass measurements, pushing the boundaries of our current understanding.

By focusing on data from cyclotron radiation emitted by electrons in a magnetic field, Project F aims to measure the electron energy spectrum near the beta decay endpoint, where the neutrino mass's influence is most notable.

I.2 Neural Networks

Neural networks form the backbone of modern machine learning applications, drawing inspiration from the structure and function of the human brain. A typical neural network consists of an input layer, several hidden layers, and an output layer. Each layer is made up of units, or neurons, which process incoming data, apply a transformation, and pass the result to the next layer.

In essence, a neural network learns by adjusting the weights of connections between neurons across different layers [3] . A weight determines how much influence one neuron's output has on another neuron's input. Additionally, each neuron may have a bias term, which allows the model to better fit the data by shifting the activation function to the left or right.

When data enters a neural network, it undergoes a series of transformations. At each neuron, the network calculates a weighted sum of inputs, adds a bias, and then applies an activation function. The activation function introduces non-linearity, enabling the network to learn complex patterns. The process continues layer by layer until reaching the output layer, which produces the final prediction.

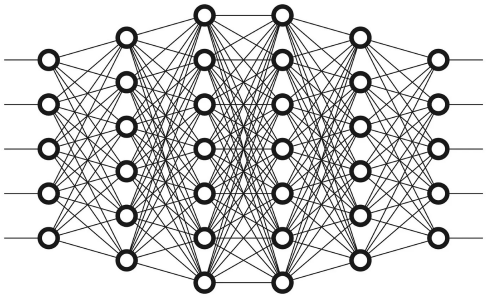


Figure 2: A diagram of a Neural Network with four hidden layers [4]

I.3 Deep Learning Algorithms (Project Specific)

I.3.1 Convolutional Neural Networks (CNN)

Convolutional Neural Networks (CNNs) are a type of neural network specifically built to interpret structured grid inputs like photographs [5]. Tasks such as image recognition, segmentation and classification have shown remarkable efficacy for CNNs.

Layers for convolution, pooling and full connectivity are commonly seen in CNN architectures. Spatial hierarchies of features (such as edges and textures) are captured by the convolutional layer by applying different filters to the input. By swiping over the input image, each filter calculates dot products and creates a feature map that highlights characteristics that it has identified.

Convolutional layers are followed by pooling layers, which serve to decrease the spatial dimensions of the feature maps. This reduces computing cost and makes feature identification scale and orientation invariant. One of the most popular pooling methods, max pooling, aggregates the maximum value for each feature map region.

Fully linked layers are the last stages of a CNN, in which every neuron in the layer above is connected to every other neuron. To accomplish categorisation, these layers aggregate all features that

the network has learned. The feature maps are flattened into a single vector prior to accessing the fully connected layers, guaranteeing that the data is in the proper format for processing. An example of a CNN is shown in figure 3 below.

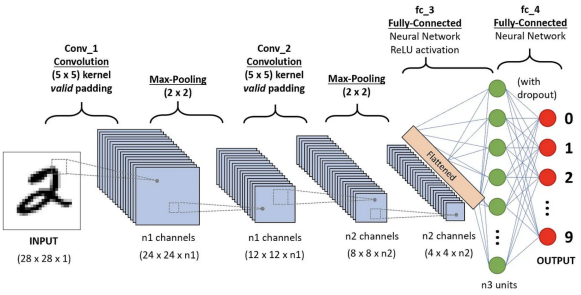


Figure 3: A diagram of a CNN sequence to classify handwritten digits [6]

CNNs harness the power of local receptive fields, shared weights, and pooling to efficiently handle image data. By focusing on local input patterns and reducing the number of parameters, CNNs can effectively learn from vast amounts of image data with relatively lower computational resources compared to fully connected networks.

In summary, while traditional neural networks apply a generalised approach to data processing, CNNs leverage structured network layers to excel in tasks involving grid-like topology data, notably images. This specialisation enables CNNs to achieve state-of-the-art performance in a wide range of visual tasks.

I.3.2 Recurrent Neural Networks (RNN)

Recurrent Neural Networks (RNNs) are another type of neural network that processes data sequences. They excel at jobs where understanding the present point in the sequence requires context from earlier in the sequence. Unlike feed-forward neural networks, which process each input separately, RNNs process sequences of inputs using their internal state, or memory.

Long Short-Term Memory Networks (LSTMs) and Gated Recurrent Units (GRUs) are advanced RNN architectures. By targeting the vanishing gradient issue that normal RNNs run into, LSTMs are able to learn on longer sequences. They achieve this through using a sophisticated architecture of information-regulating gates, such as memory gates that have the ability to add or remove data from a cell state. One type of LSTM called a GRU aims to dial down the model complexity whilst not sacrificing performance. GRUs do this by combining the forget and input gates into a single 'update gate' and by combining the cell state and hidden state.

LSTMs and GRUs are utilised in tasks requiring data order, such as language translation, text production, and speech recognition. They can capture dependencies and relationships in data that span long time periods, making them useful tools for any activity requiring sequential data.

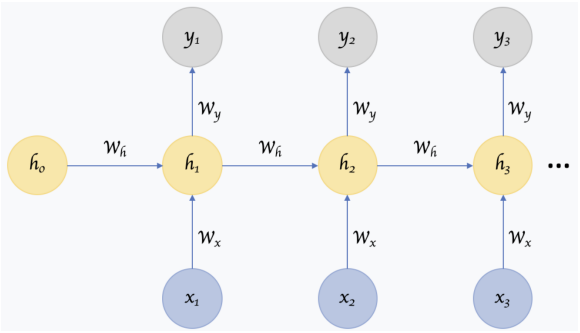


Figure 4: A diagram of a RNN with a hidden state that will carry information from one input to the next in the series [7]

II Tasks

The core objective of this project was to complete three tasks, leveraging various machine learning methods, centered around ascertaining the start frequency and time of a chirp from different sources of information. The project's data consisted of two datasets that reflect the behavior of electrons in a magnetic field. The first dataset contained spectrogram images that visually represent the change in frequency over time, which can be derived from the following voltage time series data. The second dataset includes voltage time series, akin to what would be observed on an oscilloscope connected to an antenna picking up cyclotron radiation emissions. Additionally, metadata was available for each waveform-spectrogram pairing, detailing simulated chirp

characteristics such as start time, initial frequency, frequency slope, noise temperature, and signal collection efficiency. The machine learning algorithms used this data to predict chirp characteristics.

II.1 Task 1: Spectrogram Data

II.1.1 Data Pre-Processing

The pre-processing approach detailed involves several crucial steps to prepare the spectrogram images, show in Figure 5, for deep learning analysis. Firstly, normalisation of the images is conducted to scale the pixel values between zero and one, enhancing model training efficiency. Subsequently, labels for start times and frequencies are standardised using 'StandardScaler', aligning them on a consistent scale and improving the model's learning capability. Data is then split into training and testing sets, ensuring a balanced representation for accurate model evaluation.

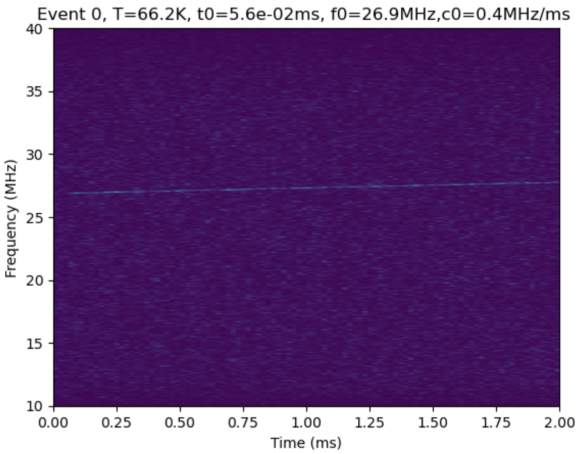


Figure 5: Spectrogram example (Event 0)

II.1.2 Model Design

The Convolutional Neural Network (CNN) architecture implemented for predicting chirp start times and frequencies from spectrogram data utilises a series of carefully chosen layers, each serving a specific purpose in processing the input data. CNNs excel at analysing images so were chosen below, in Figure 6, the model summary.

Layer (type)	Output Shape	Param #	Connected to
input_15 (InputLayer)	[(None, 600, 100, 1)]	0	[]
conv2d_37 (Conv2D)	(None, 600, 100, 16)	160	['input_15[0][0]']
max_pooling2d_37 (MaxPooling2D)	(None, 300, 50, 16)	0	['conv2d_37[0][0]']
batch_normalization_15 (BatchNormalization)	(None, 300, 50, 16)	64	['max_pooling2d_37[0][0]']
dropout_41 (Dropout)	(None, 300, 50, 16)	0	['batch_normalization_15[0][0]']
conv2d_38 (Conv2D)	(None, 300, 50, 32)	4640	['dropout_41[0][0]']
max_pooling2d_38 (MaxPooling2D)	(None, 150, 25, 32)	0	['conv2d_38[0][0]']
batch_normalization_16 (BatchNormalization)	(None, 150, 25, 32)	128	['max_pooling2d_38[0][0]']
dropout_42 (Dropout)	(None, 150, 25, 32)	0	['batch_normalization_16[0][0]']
conv2d_39 (Conv2D)	(None, 150, 25, 64)	18496	['dropout_42[0][0]']
max_pooling2d_39 (MaxPooling2D)	(None, 75, 12, 64)	0	['conv2d_39[0][0]']
batch_normalization_17 (BatchNormalization)	(None, 75, 12, 64)	256	['max_pooling2d_39[0][0]']
dropout_43 (Dropout)	(None, 75, 12, 64)	0	['batch_normalization_17[0][0]']
flatten_14 (Flatten)	(None, 57600)	0	['dropout_43[0][0]']
dense_14 (Dense)	(None, 128)	7372928	['flatten_14[0][0]']
dropout_44 (Dropout)	(None, 128)	0	['dense_14[0][0]']
start_time (Dense)	(None, 1)	129	['dropout_44[0][0]']
start_freq (Dense)	(None, 1)	129	['dropout_44[0][0]']

Total params: 7396930 (28.22 MB)
Trainable params: 7396706 (28.22 MB)
Non-trainable params: 224 (896.00 Byte)

Figure 6: Task 1's machine learning model architecture

Input Layer: Accepts the spectrogram data with a shape of (600, 100, 1), indicating 600 time steps, 100 frequency bins, and 1 channel for the grayscale spectrogram images.

First Conv2D Layer: Applies 16 filters with a kernel size of (3, 3) to extract low-level features from the input spectrogram. The choice of 16 filters is a balance between complexity and computational efficiency, allowing the model to begin understanding basic aspects of the spectrogram without overwhelming computational resources.

First MaxPooling2D Layer: Reduces the output of the first convolutions by half, consolidating features into a smaller, more manageable format while preserving the most essential information.

Batch Normalisation: Stabilises the learning process by normalising the output of the previous layers, accelerating convergence and improving the training phase's overall effectiveness.

First Dropout Mechanism: Interspersed after

convolutional blocks, dropout layers help in preventing over-fitting by randomly omitting a portion of the feature detectors during training, ensuring the model generalises well.

Second Conv2D Layer: Doubles the number of filters to 32, delving deeper into the spectrogram to uncover more intricate features necessary for accurate predictions.

Second MaxPooling2D Layer: This layer further reduces the dimensions to (148, 23), compacting the feature maps and emphasising the most significant features by pooling over (2, 2) regions again.

Second Normalisation & Dropout Layers: As previously stated, these aim to improve effectiveness and generalisation.

Third Conv2D Layer: With 64 filters, this layer aims to extract even more complex and high-level features from the spectrogram. A higher number of filters at this stage allows the network to understand intricate patterns within the spectrogram, essential for accurate time and frequency prediction of chirps.

Third MaxPooling2D Layer: To ensure that only the most prominent characteristics are carried forward, this layer completes the spatial reduction trend by giving the feature maps their final shape before they are flattened.

Third Normalisation & Dropout Layers: Effectiveness and generalisation improvements.

Flatten Layer: Converts the 3D feature maps into 1D feature vectors, preparing the data for the fully connected dense layer.

Dense Layer: A fully connected layer with 128 units that integrates features learned by the network over the spectrogram. It serves as a classification mechanism, interpreting the features extracted by the convolutions and pooling layers to make predictions.

Output Layers: Following a final dropout layer, two separate dense layers each output a single value, the chirp's start time and the its initial frequency. These layers are endpoints of the model, where final predictions are made.

This architecture was specifically designed for task one, analysing spectrogram data, with each component selected to optimise performance for chirp characteristic predictions. The use of convolutional layers captures spatial hierarchies in the data, pooling layers that reduce dimensions and computational complexity, and dropout layers to address over-fitting, together creates a model that generalises well to unseen data while retaining the capacity to learn from complex patterns.

II.1.3 Training & Validation

The training and validation loss graphs for task one illustrate that we have achieved a solid model for a task that involves image analysis and the prediction of two distinct properties, 't0s' and 'f0s'. Initially, the model displays a significant decrease in both training and validation losses, shown in Figure 7, indicating an effective learning rate during the early epochs, with the steepest improvements around epochs 3-7. Beyond this point, the losses begin to level off, suggesting that the model is starting to reach the optimal learning capacity given its current structure and parameters. The trend shows the model is generalising well, as evidenced by the parallel decline of both training and validation losses, implying that it is achieving a balance between learning from the training data and maintaining accuracy on unseen data, without any substantial over-fitting.

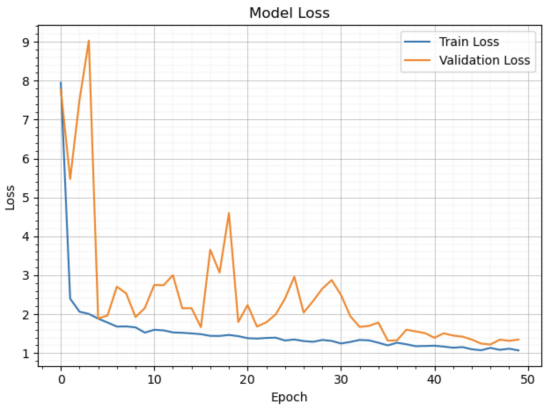


Figure 7: Task 1's Model Loss vs Epoch

Both mean squared error (MSE) plots for 'start time' and 'start frequency', shown below, reinforce this observation as they exhibit a similar pattern of reduction, pointing to the model's ability to hone in on the nuances of both temporal and frequency aspects of the input images. The

alignment between the MSE for training and validation sets shows the consistency of the model's performance across both datasets. However, the observable plateau in validation loss alongside a marginal but continuing decrease in training loss calls for a cautious approach moving forward. This pattern could be an early indicator of overfitting and that the model might be starting to pick up idiosyncrasies in the training data that do not generalise well.

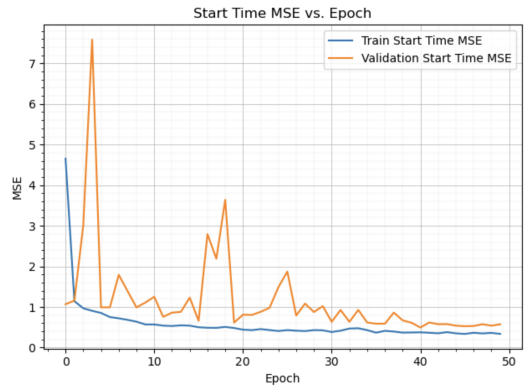


Figure 8: Task 1's Start Time Mean Squared Error vs Epoch

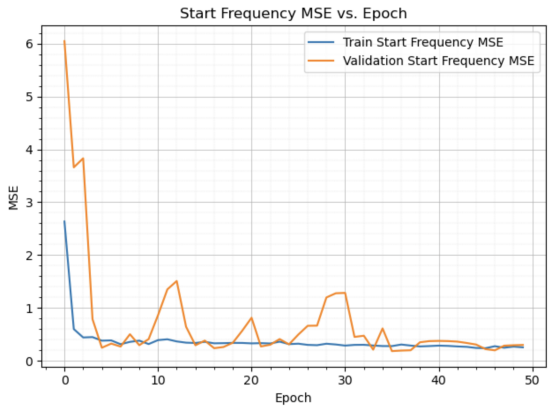


Figure 9: Task 1's Start Frequency Mean Squared Error vs Epoch

Considering the multi-faceted nature of the predictions—both temporal and frequency elements—and the similarity of the learning trends across these dimensions, it's clear the model is well-structured for the task. Despite the robust performance up to the last observed epoch, it is essential to keep a watchful eye on the progression of the validation loss. A rise in validation loss in future epochs would be a sign that the model's generalisation capabilities are decreasing and that it's beginning to memorise the training data rather than learning to discern the general patterns that apply to both the training and unseen data.

II.1.4 Model Predictions

The outcomes shown in the spectrogram pictures below show the predictive power of the model with temporal predictions having noteworthy accuracy ($\approx$97.5%). However, the accuracy for the frequency predictions have shown some variability. This variability suggests that certain aspects of the input spectrograms, potentially noise artifacts or singular striations, might be impacting the model's ability to pinpoint the exact start times of the events. Such discrepancies could stem from the model interpreting these features (horizontal lines) as relevant signals, thereby introducing a degree of confusion in the frequency predictions. In order to strengthen the model's resistance in relation to these input data striations and produce more precise temporal predictions, additional model refinement may involve the use of improved feature extraction layers or noise reduction techniques.

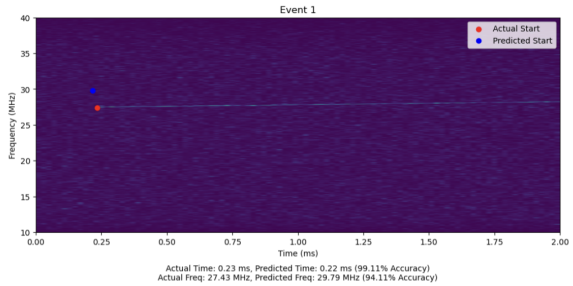


Figure 10: Task 1's (Event 1) resulting prediction vs actual spectrogram

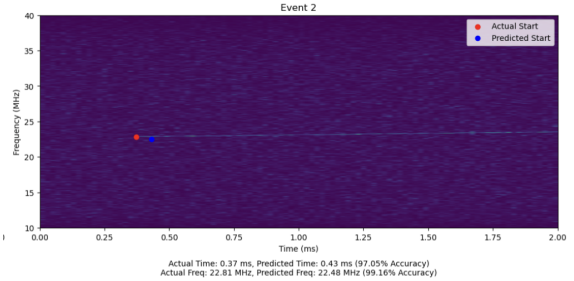


Figure 11: Task 1's (Event 2) resulting prediction vs actual spectrogram

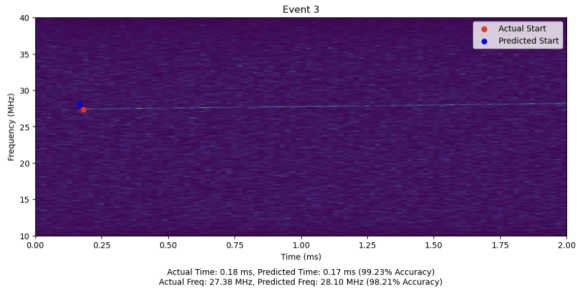


Figure 12: Task 1's (Event 3) resulting prediction vs actual spectrogram

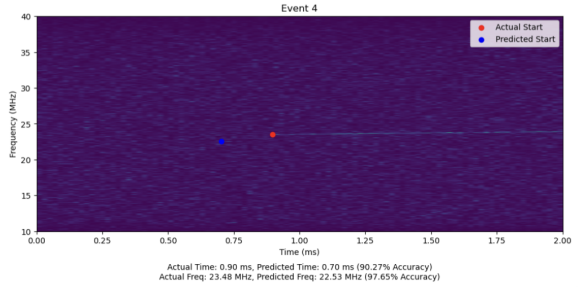


Figure 13: Task 1's (Event 4) resulting prediction vs actual spectrogram

However, it is clear that the model functions well over a range of events, demonstrating its effectiveness and potential use in time-frequency analysis applications requiring admirable precision.

II.2 Task 2: Time Series Data

Task two focuses on leveraging time series data to predict the start times and frequencies of chirps. The challenge lies in extracting these signals accurately, necessitating a robust machine learning approach of which we would usually utilise an LSTM method. However, due to the constraints of this computer this could not be realised. Therefore, three alternative machine learning methods have been made for Task 2; a simple CNN with cutoff, a complex CNN with sub-sampling, and a GRU model. This allows us to come to a general consensus of which model performs best when the model of choice (LSTM) is not available.

II.2.1 Data Pre-Processing

Model 1: As part of the first model's pre-processing a cutoff is implemented at the 75th percentile to filter data, reducing noise by nullifying values below the threshold. This emphasises more significant signals in the time series. Following this the data is, like in Task 1, normalised and reshaped for the models input.

Model 2: The second model's pre-processing involved standardising the data to preserve feature variances, crucial for time-series analysis. This approach prevents larger variances from overshadowing smaller ones, enhancing the model's learning from data variability. After reshaping, the model was compiled. Sub-sampling then reduced X_{train} & Y_{train} sizes to decrease the computational demand and accelerate training, leading to the model being trained on this optimised dataset.

Model 3: For the last model's pre-processing, the data was simply normalised, reshaped, and fed into the model.

II.2.2 Model Designs

Model 1:

This model uses multiple 1D convolutional and max-pooling layers to process time-series data:

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1d (Conv1D)             (None, 199995, 16)        112

 max_pooling1d (MaxPooling1  (None, 99997, 16)         0
 D)

 conv1d_1 (Conv1D)           (None, 99992, 64)         6208

 max_pooling1d_1 (MaxPoolin  (None, 49996, 64)         0
 g1D)

 flatten (Flatten)           (None, 3199744)           0

 dense (Dense)               (None, 2)                 6399490

=================================================================
Total params: 6405810 (24.44 MB)
Trainable params: 6405810 (24.44 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Figure 14: Model 1 architecture for Task 2

Conv1D Layer: With 16 filters and a kernel size of 6, this layer begins to extract low-level features from the input data sequence, which is crucial for identifying trends and patterns in time-series data.

MaxPooling1D Layer: The subsequent pooling layer down-samples the input representation by taking the maximum value over a pool size of 2, reducing the computational load and extracting the dominant features.

Conv1D Layer: Increases the complexity of the model by applying 64 filters, in order to capture

more sophisticated aspects of the data.
MaxPooling1D Layer: Further reduces data dimensionality, this helps in preventing over-fitting and makes the detection of features invariant to small shifts and distortions.
Flatten Layer: This layer flattens and makes the data into a single long vector to prepare the data for the final dense layer.
Dense Layer (Output): A fully connected layer that integrates the learned features to predict two continuous values, the start time and frequency of a chirp.

Model 2:

This model is a more complex CNN, which uses sub-sampling to reduce the input data size, thus improving training efficiency:

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1d_2 (Conv1D)           (None, 99997, 32)         256

 max_pooling1d_2 (MaxPoolin  (None, 49998, 32)         0
 g1D)

 conv1d_3 (Conv1D)           (None, 24997, 64)         10304

 max_pooling1d_3 (MaxPoolin  (None, 12498, 64)         0
 g1D)

 flatten_1 (Flatten)         (None, 799872)            0

 dense_1 (Dense)             (None, 32)                25595936

 dense_2 (Dense)             (None, 2)                 66

=================================================================
Total params: 25606562 (97.68 MB)
Trainable params: 25606562 (97.68 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Figure 15: Model 2 architecture for Task 2

Conv1D Layer: Utilises a larger stride (2) for more aggressive down-sampling within the first convolutional layer. It allows for faster acquisition of the general context of the input data.
MaxPooling1D Layer: Complements the convolutional layer by summarising the features over a window of 2.
Conv1D Layer: With a stride of 2 and 64 filters, it builds a deeper understanding of the data by finding complex patterns.
MaxPooling1D Layer: Continues the trend of spatial reduction and emphasises important features.
Flatten Layer: Transforms the multi-dimensional features into a vector.
Dense Layer: Serves as a classification layer with a reduced size of 32 units to interpret the features for making predictions.
Dense Layer (Output): Outputs two values for the prediction, making use of sub-sampled training data for efficiency.
It is worth noting that for this model we sub-sample after the model.

Model 3:

This model incorporates a GRU layer, a variant of RNNs suitable for time-series data predictions:

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 gru (GRU)                   (None, 1, 128)            76849920

 dropout (Dropout)           (None, 1, 128)            0

 gru_1 (GRU)                 (None, 64)                37248

 dropout_1 (Dropout)         (None, 64)                0

 dense_3 (Dense)             (None, 2)                 130

=================================================================
Total params: 76887298 (293.30 MB)
Trainable params: 76887298 (293.30 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Figure 16: Model 3 architecture for Task 2

GRU Layer: Processes the input sequence in a recurrent manner, allowing the model to maintain information across longer sequences, which is essential for tasks like time-series forecasting.
Dropout Layer: Reduces over-fitting by randomly ignoring a fraction of the units during training, forcing the network to learn more robust features.
Dense Layer: After the sequential information has been processed by the GRU, the dense layer consolidates these features into predictions for the start time and frequency.

II.2.3 Training & Validation

Upon training, the validation loss was found to repeatedly converge to ≈ 0.08. This might indicate a consistent error margin that could be tied to noise within the data or perhaps some inherent complexity that the model is unable to capture.

Model 1:

A sharp decrease in training loss during the first epoch indicates effective initial learning. Subsequent epochs show both losses stabilising, suggesting the model quickly reached a good generalisation performance without over-fitting, as evidenced by the close and parallel loss values. This indicates a well-tuned model for the task at hand.

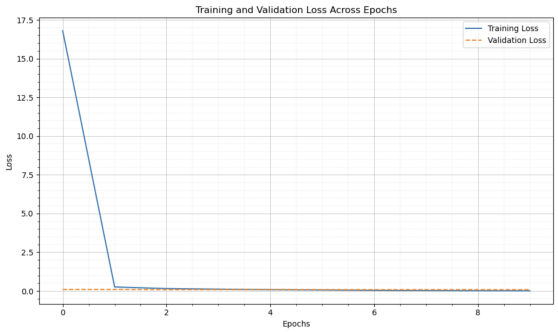


Figure 17: Model 1 Train & Val. Losses vs Epochs

Model 2:

The decline and convergence suggest effective model learning and good generalisation to unseen data. The training loss follows the validation loss closely demonstrating no over-fitting. The stable model performance across training and validation sets is indicative of a well-functioning model.

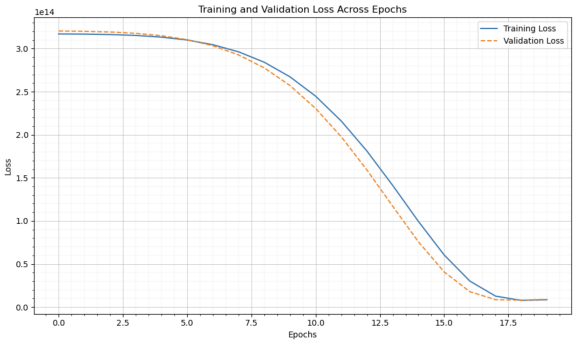


Figure 18: Model 2 Train & Val. Losses vs Epochs

Model 3:

Like the other two models, the training loss has a sharp initial decrease however levels off post epoch five whereby it converges close to the validation loss. This shows the models effectiveness in learning fast and generalising well to unseen data.

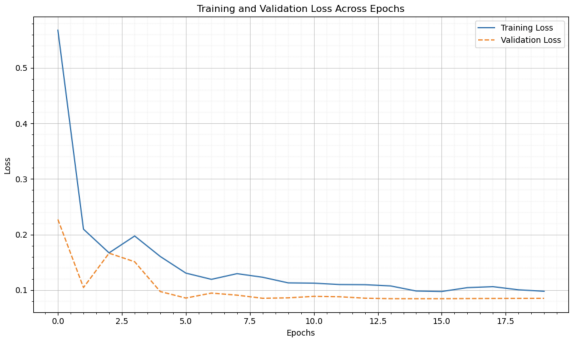


Figure 19: Model 3 Train & Val. Losses vs Epochs

II.2.4 Model Predictions

Model 1:

For Task 2, a different method of prediction visualisation was used for easier comparison between the different models. Fig. 20 shows Model 1's predictions capturing the general trends of the start frequencies and times, though not perfectly. It's positive to see the model is responsive to the changes in the actual values, suggesting that the underlying neural network architecture is picking up patterns from the data. Since the trends are generally followed, the model seems promising, and further fine-tuning, noise filtering, or feature engineering could potentially enhance its predictive performance.

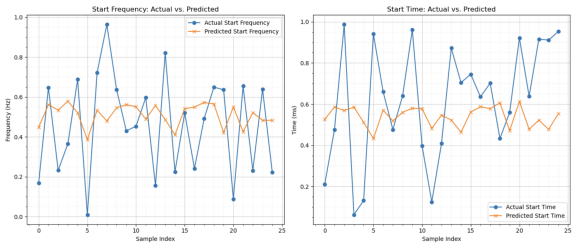


Figure 20: Model 1 Chirp start time and start frequency predictions

Model 2:

Model 2 demonstrates a slightly different performance. While it captures the pattern of start

times very well, the scale of the start frequencies is significantly different from the actual values, possibly due to scaling issues in the data preprocessing step or the presence of outliers affecting the model's output.

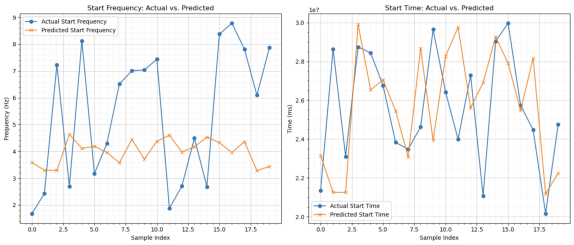


Figure 21: Model 2 Chirp start time and start frequency predictions

Model 3:

Model 3 under-performs due to its inability to accurately capture the start frequency and time variations in the dataset, exhibiting constant predictions that fail to reflect the actual varying data. The effectiveness of RNNs in time series analysis is heavily influenced by the quality of the input data. While RNNs are designed to process sequential data, their ability to yield accurate predictions is severely reduced when the data is of poor quality. Noise and irregularities within the data have prevented effective learning, suggesting that the model's architecture is not at fault. Instead, the focus should be on enhancing data quality and pre-processing to improve predictions, emphasising that the efficacy of models is reliant on the quality of data they are trained on.

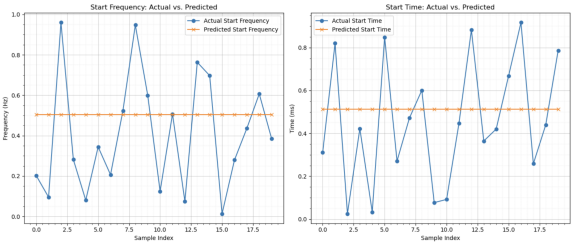


Figure 22: Model 3 Chirp start time and start frequency predictions

II.3 Task 3: Spectrogram Data & Time Series Data

Task three merges objectives from both task one and two into a combined aim of using the spectrogram and time-series data simultaneously to predict chirp times and start frequencies. Given the GRU network's poor performance and the CNN's commendable results, we utilised complex CNNs for the final task.

II.3.1 Data Pre-Processing

For Model 3, data pre-processing was similar to the other tasks, it involved normalising both spectrogram images and time series voltage data to a [0, 1] range using MinMaxScaler, ensuring compatibility with the hybrid neural network's inputs. This normalisation process is critical for pattern recognition in spectrograms by the CNN layers and for effective sequential data analysis by the Conv1D & Conv2D layers. Additionally, target variables for chirp start times and frequencies were normalised to improve prediction accuracy. The dataset was then split into training and testing sets to carry out model training and validate its performance on unseen data. This pre-processing method is essential for optimising data for analysis, enabling precise chirp predictions.

II.3.2 Model Design

As described above, a complex CNN has been made utilising Convo2D layers for two-dimensional spectrogram analysis and Convo1D layers for the singular dimension time-series data analysis. The two resulting outputs are then merged via a concatenate layer to create a single combined vector for further analysis.

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, 600, 100, 1)]	0	[]
input_2 (InputLayer)	[(None, 200000, 1)]	0	[]
conv2d (Conv2D)	(None, 600, 100, 24)	240	['input_1[0][0]']
conv1d (Conv1D)	(None, 200000, 24)	96	['input_2[0][0]']
max_pooling2d (MaxPooling2D)	(None, 300, 50, 24)	0	['conv2d[0][0]']
max_pooling1d (MaxPooling1D)	(None, 100000, 24)	0	['conv1d[0][0]']
batch_normalization (BatchNormalization)	(None, 300, 50, 24)	96	['max_pooling2d[0][0]']
batch_normalization_1 (BatchNormalization)	(None, 100000, 24)	96	['max_pooling1d[0][0]']
global_average_pooling2d (GlobalAveragePooling2D)	(None, 24)	0	['batch_normalization[0][0]']
global_average_pooling1d (GlobalAveragePooling1D)	(None, 24)	0	['batch_normalization_1[0][0]']
concatenate (Concatenate)	(None, 48)	0	['global_average_pooling2d[0][0]', 'global_average_pooling1d[0][0]']
dense (Dense)	(None, 32)	1568	['concatenate[0][0]']
dropout (Dropout)	(None, 32)	0	['dense[0][0]']
batch_normalization_2 (BatchNormalization)	(None, 32)	128	['dropout[0][0]']
dense_1 (Dense)	(None, 2)	66	['batch_normalization_2[0][0]']

Total params: 2290 (8.95 KB)
Trainable params: 2130 (8.32 KB)
Non-trainable params: 160 (640.00 Byte)

Figure 23: Model architecture for Task 3

Conv2D Layer: With 24 filters and a kernel size of 3x3, this layer processes the spectrogram input, extracting spatial features.
MaxPooling2D Layer: Reduces the feature map dimensions by half, emphasises most relevant features while decreasing computational load.
BatchNormalization: Stabilises learning by normalising the output of the previous layer, which can accelerate convergence and improve overall performance.
GlobalAveragePooling2D: Transforms the two-dimensional feature map into a one-dimensional vector, reducing the number of features and minimising over-fitting possibility.
Conv1D Layer: Uses 24 filters with a kernel size of 3 to extract temporal features from the one-dimensional time-series data.
MaxPooling1D Layer: Halves the sequence length, allowing the model to focus on the most significant temporal features.
BatchNormalization: Similar to the 2D version, it normalises the features from the Conv1D layer, aiding in model training stability.
GlobalAveragePooling1D: Reduces the feature dimensions, resulting in a fixed-size output that ensures model robustness and generalisation.
Concatenate Layer: Joins the aquired spatial and temporal features into a single vector, leveraging the strengths of both CNN and Conv1D branches for feature representation.
Dense Layer: A fully connected layer with 32 neurons that integrates the combined features, needed for the prediction task.
Dropout Layer: Introduces regularisation by randomly dropping 30% of the neurons, encouraging the model to learn more robust features and prevent over-fitting.
BatchNormalization: Ensures the outputs from the dropout layer are scaled appropriately, contributing to faster and more stable learning.
Output Dense Layer: The final fully connected layer with two neurons, using "linear" as the activation function to predict the chirp start times and frequencies.
The model's architecture capitalises on CNN's spatial feature extraction capabilities and Conv1D's temporal pattern recognition, while also emphasising efficiency and generalisation through global pooling layers and regularisation layers. These elements are strategically combined to predict the characteristics of chirps from complex datasets without over-fitting.

II.3.3 Training & Validation

The training and validation loss of task three's model converge to ≈ 0.08, a pattern previously seen in task two, suggesting a potential noise floor in the data that stops further loss reduction.

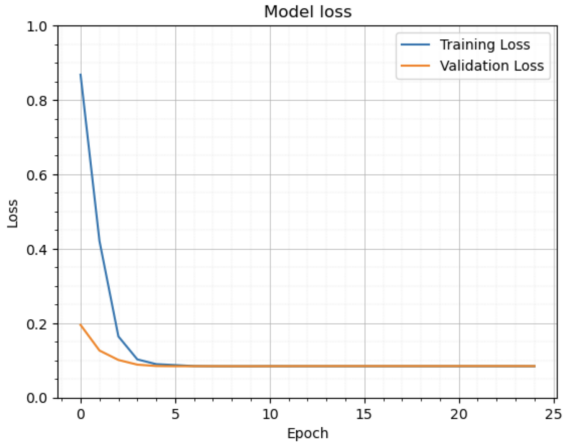


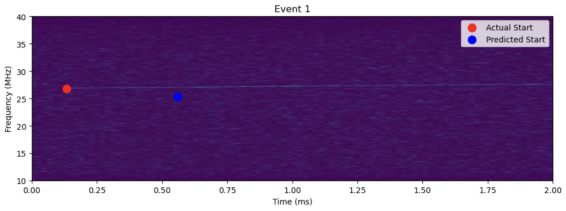
Figure 24: Model 3 Train & Val. Losses vs Epochs

However, the initial significant reduction in loss indicates that the model learns quickly, and it does not show over-fitting because the validation loss

stabilises with the training loss. On the other hand, the reduced validation loss is unusual and could point to problems such as data leaking, underfitting, or a too simplistic model. The fact that multiple models plateau at the same threshold suggests that, besides adjusting the model's complexity, addressing data quality issues such as noise reduction, advanced feature engineering, or data augmentation may be necessary to improve the model's performance even with the good generalisation indicated by the low validation loss. Therefore, additional investigation into the subtleties of the data should be carried out to increase accuracy and get past the recurrent 0.08 loss constraint.

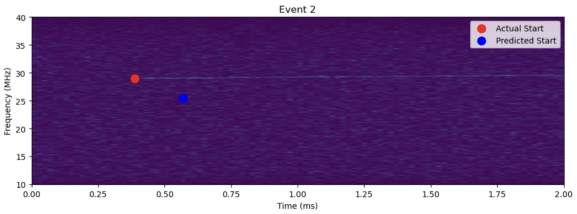
II.3.4 Model Predictions

The model performs well with frequency related predictions, but falls sub par when executing temporal predictions, most likely due to poor input data.



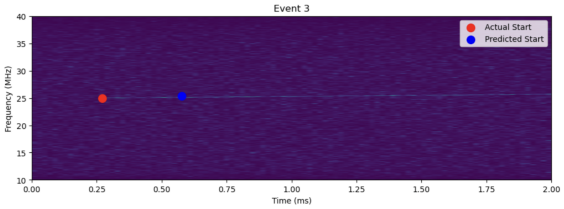
Actual Time: 0.1332 ms, Predicted Time: 0.5588 ms
Actual Freq: 26.8314 MHz, Predicted Freq: 25.3375 MHz

Figure 25: Model 3 Predictions vs Actual (Event 1)



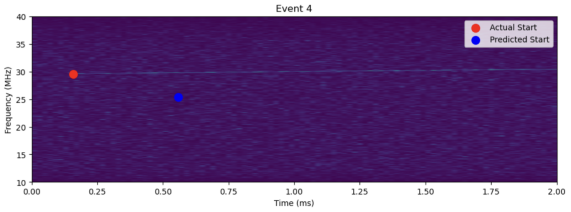
Actual Time: 0.3869 ms, Predicted Time: 0.5688 ms
Actual Freq: 28.9532 MHz, Predicted Freq: 25.3856 MHz

Figure 26: Model 3 Predictions vs Actual (Event 2)



Actual Time: 0.2719 ms, Predicted Time: 0.5763 ms
Actual Freq: 24.9217 MHz, Predicted Freq: 25.4223 MHz

Figure 27: Model 3 Predictions vs Actual (Event 3)



Actual Time: 0.1589 ms, Predicted Time: 0.5576 ms
Actual Freq: 29.5332 MHz, Predicted Freq: 25.3312 MHz

Figure 28: Model 3 Predictions vs Actual (Event 4)

The model's predictions across the four events show a consistent trend that while the predicted frequencies lie close to the actual frequencies, there is a noticeable discrepancy in the timing predictions. The similar mean squared error (MSE) values for time and frequency across these events (Time MSE: 0.0835, Frequency MSE: 0.0859), as well as the convergence around an MSE of approximately 0.08 seen in previous tasks, suggest that the model's precision is constrained by factors in the input data.

Given the continuing gap in temporal prediction accuracy, it is likely that noise or complex signal characteristics within the dataset are stopping the model from making precise temporal measurements. The consistency of this pattern across different models and events reinforces the idea that addressing data quality, perhaps through noise reduction techniques and improved feature extraction, could improve model performance.

Overall, the model shows potential in frequency identification but requires improvements in the temporal analyses to provide a more accurate and well-rounded prediction in the tasks.

III Conclusion

In this project, we analysed Cyclotron Radiation Emission Spectroscopy (CRES) data using deep learning models in an attempt to precisely model the frequencies and start times of energy chirps, a sign of neutrino interactions. We explored the subtleties of Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) in processing high-dimensional spectrogram and time series data through three different tasks.

Our results showed that the task one model performed admirably at predicting both frequency and time, frequently coming close to the real values. Conversely, task two's models did well in temporal forecasting but fell short with the frequency predictions. Lastly, task three's model showed the opposite behaviour and forecasted the frequency well, but the temporal not. This mismatch, along with a persistent plateau in the loss values around 0.08 for both tasks and models, indicates that the data itself likely represents the source of the performance ceiling we observed rather than just the models' architectures. The assumption that there is an intrinsic noise floor in the data, which prevents the models from improving their temporal accuracy, is supported by the convergence of mean squared error (MSE) values for both time and frequency forecasts.

Our investigation, in summary, highlights the delicate balance that must be struck between model complexity, data quality, and the difficulties that arise with data analysis. Deep learning clearly has the potential to make a significant contribution to neutrino physics and the larger scientific community. However, the issues will arise when overcoming the challenges presented by data noise and complexity.

IV Bibliography

[1] Prof. Ryan Nichol, "cresWeb", Project Workbook.

[2] Solar Wind-Magnetosphere Interactions: A Statistical Analysis of Spacecraft Measurements - Scientific Figure on ResearchGate. [accessed 18 Mar, 2024]

[3] "TensorFlow," TensorFlow. [Online]. Available: https://www.tensorflow.org/guide. [Accessed: Mar.18, 2024].

[4] Mahapatra, S. (2019). Why Deep Learning over Traditional Machine Learning? [online] Medium. Available at: https://towardsdatascience.com/why-deep-learning-is-needed-over-traditional-machine-learning-1b6a99177063.

[5] Taye, M.M. Theoretical Understanding of Convolutional Neural Network: Concepts, Architectures, Applications, Future Directions. Computation 2023, 11, 52.

[6] Saha, S. (2018). A Comprehensive Guide to Convolutional Neural Networks—the ELI5 way. [online] Towards Data Science.

[7] gotensor. (2019). Recurrent Neural Networks - Remembering what's important. [online] Available at: https://gotensor.com/2019/02/28/recurrent-neural-networks-remembering-whats-important/.