Machine Learning Mini-Project report

Student Number: 21022300

Project F: Measuring chirps and energy

Submission date: 21/03/2024

Abstract:

This paper demonstrates machine learning methods to measure the start time and frequency of a neutrino mass (chirp) event using both spectrograph images and voltage time series data from cyclotron radiation emission spectroscopy (CRES). Various models are utilised, including dense networks, convolutional neural networks (CNNs) and deep, dense networks. A de-noising method is also applied to decrease the loss on the spectrogram data. The de-noising algorithm proves to be robust and facilitates the CNNs to prove the best results.

I. Introduction:

While the neutrino may be the most abundant particle in the universe [1], its mass is not yet known with high certainty [2]. CRES is a useful, sensitive method to measure the neutrino mass [3], where electrons go through cyclotron motion within a magnetic field, producing radiation as they do due to their centripetal acceleration [3].

The data analysed in this project was CRES data, specifically that of electrons in a magnetic field; spectrogram image data (Figure 1), and wave data (Figure 2). These were analysed and trained on by various machine learning models to determine the start time and frequency of each event. Neural networks were trained on these data both separately, and in conjunction, in a combined model. These models were created using TensorFlow [4], an open-source library for machine learning.

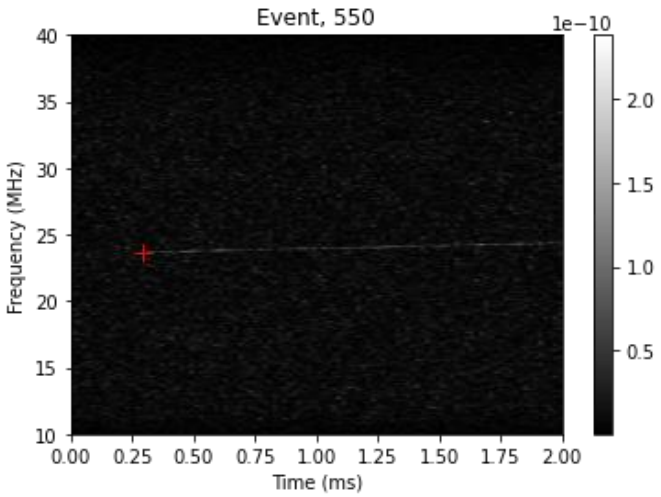


Figure 1: An example of a raw spectrograph image, showing a low signal to noise ratio. The red plus marks the start time and frequency of the chirp event.

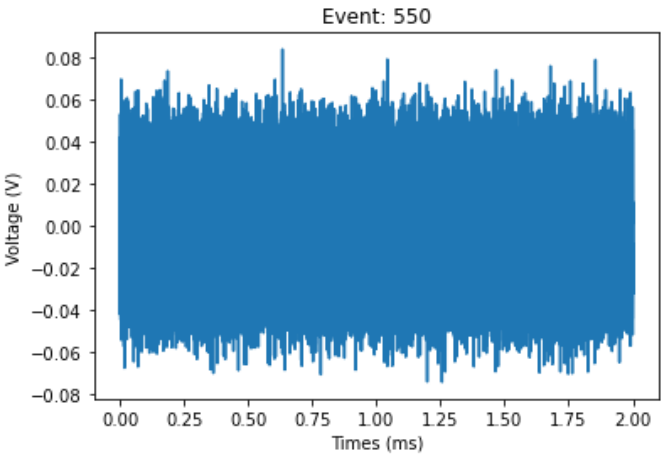


Figure 2: An example of a set of raw wave data.

Several different types of networks were applied, based on the concept of computer vision, a field of Artificial Intelligence (AI)

that trains models to extract relevant data from images [5]. The methods applied to the image data were CNNs, and purely dense networks.

In the case of image data, CNNs work by applying 2D filter kernels to the image data, calculating a dot product between the filter and a part of the image and sweeping this across the entire image [6] (Figure 3). In the case of 1D data, such as the wave data that was analysed, a 1D filter kernel can be used to the same effect.

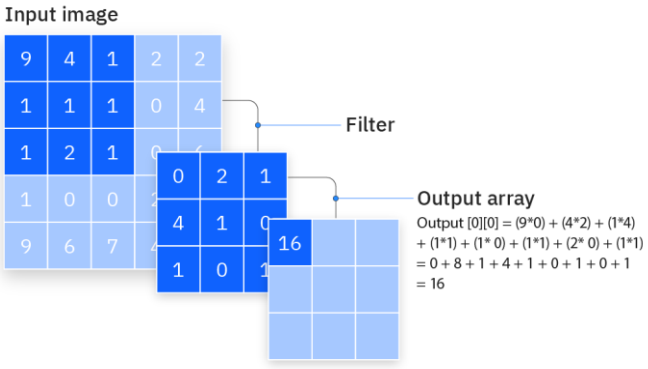


Figure 3: A visual example of 3D kernel filtering. [6]

Dense neural networks were also applied. Dense neural networks are a subset of Deep neural networks (Figure 4), where each neuron in a layer is connected to each neuron in the following layer [7].

Overall, the main aim of this project was to determine the start time and frequency of a chirp event in an efficient manner; large numbers of parameters were to be avoided to ensure fast runtimes and reduce the likelihood of overfitting. This was especially important due to the features in the data being incredibly fine, and thus easy to miss by computer vision (when compared to the background noise).

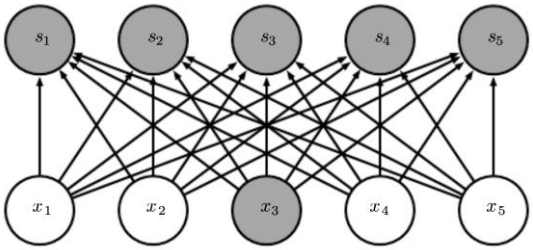


Figure 4: A visual example of 2 dense (fully connected) layers. [10]

Given the data types available, 3 main goals were set: Calculate the start time and frequency from the spectrogram data, from the wave data, and from a combination of the two. The image data objective was expanded into testing a de-noising algorithm to improve training and validation losses, as well as generating noiseless images (Figure 5), to investigate the extent to which the noise temperature affected the accuracy of the predictions on the realistic noisy data.

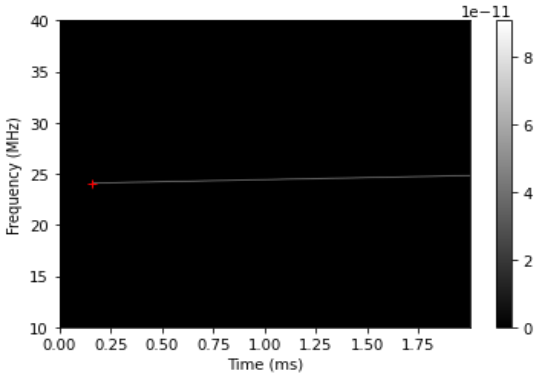


Figure 5: An example of a raw noiseless image with the start point marked.

II. Method:

IIa. Spectrogram data:

The project began by using the spectrogram data to predict the chirp start time and frequency. Since the spectrogram data is image based, a convolutional neural network was initially assumed to be the most well-suited model to predict the start time and frequency.

The data was prepared by importing 2000 images, each 600x100 pixels, from two separate datasets. These images were all normalised (such that the maximum value on each image was 1), and an array of labels was created for each image, containing the start time and frequency.

The labels were also normalised by dividing them by their maximum values, so all labels took values between 0 and 1, as the models were found to perform better when the two labels had similar values. The labels could easily be de-normalised after a prediction was made by multiplying them by their associated maximum values, to return a frequency in MHz and a time in ms.

The images and labels were then randomly shuffled (while preserving image to label correspondence, such that each image still corresponded to the correct label), as the data was taken from two separate datasets, and it is unknown how these data were generated, so shuffling ensured there would be no major differences between the first and last 1000 images.

A convolutional neural network (Model 1) was prepared and tested on the spectogram. The model is summarised in the Table 1.

Layer	Output shape	Parameters
Input	(600, 100, 1)	
Conv2d	(None, 598, 98, 32)	320
Dropout	(None, 598, 98, 32)	0
Flatten	(None, 1875328)	0
Dense	(None, 8)	15,002,632
Dense	(None, 16)	144
Dense	(None, 16)	272
Dense Output	(None, 2)	34
Total:		15,003,402

Table 1: Model 1, the convolutional neural network trained on the image data.

The dropout layers reduce overfitting in the model, by randomly discarding a fraction of the parameters in the previous layer based on the argument given to the dropout layer, ensuring the model does not become too reliant on any nodes. Various dropout values were experimented with, but 0.3 (discarding 30% of the parameters) was generally found to provide the best results. Dropout layers do, however, increase training times as more epochs are required to compensate for the loss of training information due to the dropout layers.

A dense neural network was also prepared and tested on the spectrogram data. The model is summarised in Table 2.

Layer	Output shape	Parameters
Input	(600, 100, 1)	
Flatten	(None, 60000)	0
Dense	(None, 128)	7,680,128
Dropout	(None, 128)	0
Dense	(None, 1024)	132,096
Dropout	(None, 1024)	0
Dense	(None, 1024)	1,049,600
Dropout	(None, 1024)	0
Dense Output	(None, 2)	2050
Total:		8,863,874

Table 2: Model 2, the dense neural network trained on the image data.

This model has fewer parameters, and thus can run significantly faster, which meant tuning the model was easier, and training over more epochs was possible.

However, the raw image data was very noisy, so some initial filtering to reduce this noise could clearly improve the model accuracy. For this, the chosen library was the cv2 library [8], an open-source image and video processing library. Since the important characteristic of the image data is a long horizontal edge, filtering could be used to separate this edge from the noise in the data (Figure 6). Hence, an edge enhancement Sobel filter was the first filter applied, to emphasize the edges within the data. This was followed by a Gaussian blur to make this edge more prominent by smearing it out slightly. Finally, a threshold was applied to separate the brighter line from the noise. This algorithm was applied to all the images.

After this, CNNs and dense neural networks could be applied to the images as before. The model utilised were the same as the previous two that were applied to the noisy data. The only difference was, for the dense model, the activation function on the output layer was linear, rather than sigmoid as in the previous model, as this gave much smaller losses.

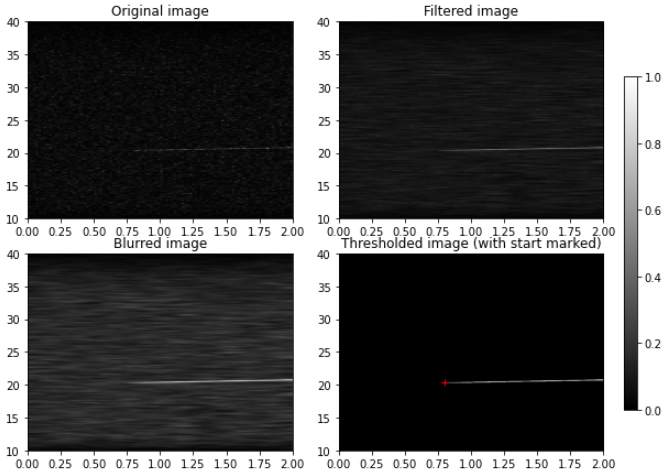


Figure 6: The results after each filtering stage. The start point is marked with a red plus on the final image.

IIb. Time series data:

The data were similarly prepared as in the Spectrogram case; the wave data were normalised by dividing each wave by the maximum absolute value (as the waves took both positive and negative values). This meant that each wave took values between -1 and 1. The same labels were used as before, such that the label values were also between 0 and 1.

The initial model considered for prediction on the time series data was a Long Short-Term Memory (LSTM) network. However, such a network is very computationally expensive, due to possessing large numbers of parameters, meaning the runtimes would be too long on the large wave datasets.

The next chosen model was a 1D CNN, summarised in Table 3.

Layer	Output shape	Parameters
Input	(200000, 1)	
Conv1D	(None, 199998, 16)	64
Dropout	(None, 199998, 16)	0
Maxpooling	(None, 99999, 16)	0
Conv1D	(None, 99997, 32)	1,568
Dropout	(None, 99997, 32)	0
Maxpooling	(None, 49998, 32)	0
Flatten	(None, 1599936)	0
Dense Output	(None, 2)	3,199,874
Total:		3,201,506

Table 3: Model 3, the convolutional neural network trained on the wave data.

Another model utilised was a dense neural network, summarised in Table 4.

Layer	Output shape	Parameters
Input	(200000, 1)	
Flatten	(None, 200000)	0
Dense	(None, 64)	12,800,064
Dropout	(None, 64)	0
Dense	(None, 1024)	66,560
Dropout	(None, 1024)	0
Dense	(None, 1024)	1,049,600
Droupout	(None, 1024)	0
Dense Output	(None, 2)	2,050
Total:		13,918,274

Table 4: Model 4, the dense neural network trained on the wave data.

IIc. Image combined with time series data:

The most successful and computationally inexpensive models were combined into a single model. Since so much data was being processed at one time, it was critical that the models have a reasonably low number of parameters. Hence, the chosen models were two dense neural networks, summarised in Table 5.

Layer	Output shape	Parameters
Input (I)	(600, 100, 1)	
Input (W)	(200000, 1)	
Flatten (I)	(None, 60000)	0
Flatten (W)	(None, 200000)	0
Dense (I)	(None, 128)	7,680,128
Dense (W)	(None, 64)	12,800,064
Dense (I)	(None, 1024)	132,096
Dense (W)	(None, 1024)	66,560
Dense (I)	(None, 1024)	1,049,600
Dense (W)	(None, 1024)	1,049,600
Concatenation Layer	(None, 2048)	0
Dense Output	(None, 2)	4098
Total:		22,782,146

Table 5: Model 5, the dense neural network trained on the combined data. A dropout layer was placed between each dense layer.

IId. Generated noiseless data:

To investigate the effect of the noise on the prediction capabilities of the models, spectrogram data was manually generated using the MockCresGenerator function. This was used to generate 10000 images,

each of size (600, 100), and the noise temperature parameter was set to 0, such that only the feature of interest (the long straight line) remained, with no background noise. Random time and frequency values were generated from a uniform distribution. These images and their labels were then normalised, such that all values were between 0 and 1.

This data was then provided to the model summarised in Table 6.

Layer	Output shape	Parameters
Input	(600, 100, 1)	
Flatten	(None, 60000)	0
Dense	(None, 128)	7,680,128
Dropout	(None, 128)	0
Dense	(None, 1024)	132,096
Dropout	(None, 1024)	0
Dense	(None, 1024)	1,049,600
Dropout	(None, 1024)	0
Dense Output	(None, 2)	2050
Total:		8,863,874

Table 6: Model 6, the dense neural network trained on the noiseless image data.

III. Results:

Each model was trained on a training dataset, and evaluated on the same training dataset, along with an unseen validation dataset. The models were trained until their losses converged to a stable value. In each model, the mean squared error loss was used, as this provided the quickest convergence to very small values.

Overall, the image data was analysed most successfully, providing the best predictions on both seen and unseen data, especially after de-noising. This is due to the image data possessing unique, easily identifiable features which could easily be augmented with image processing. In contrast, the wave data possessed no immediately identifiable features, and the noise could not be eliminated due to the frequency characteristics rather than the amplitude of the data being the feature of interest.

Performance metrics were calculated ro evaluate each model. The main calculation utilised was Euclidean distance. For a characteristic truth point, (x_n, y_n), and its associated predicted value, (p_{x_n}, p_{y_n}), the Euclidean distance, D_n, (calculated in arbitrary units, as the labels were normalised) between the points is given by:

$$D_n = \sqrt{\left(p_{x_n} - x_n\right)^2 + \left(p_{y_n} - y_n\right)^2} \quad \text{(Eq. 1)}$$

A mean value, $\overline{D}$, was also calculated for D_n. From these values, the standard deviation, σ, could be calculated:

$$\sigma = \sqrt{\frac{\sum(D_n - \overline{D})^2}{n-1}} \quad \text{(Eq. 2)}$$

These values were calculated for each model. The smaller $\overline{D}$, the closer the predictions were on average to the true start points (i.e. the greater the accuracy), and the smaller σ, the less spread out these distances were (i.e. the greater the precision).

Using Equation 1 and Equation 2, the performance metrics of each model are listed in Appendix I. The epoch against loss plots are provided in Appendix II.

Clearly, the noiseless wave data provided the best results, demonstrating directly that the difference between the measured (predicted) and true times/frequencies is directly proportional to the simulated noise temperature (i.e. the difference decreases as the noise temperature decreases). The filtered de-noised images provide a middle ground between the original spectrogram images, and the truly noiseless images.

The simulated noiseless images represent an upper limit of the performance attainable by the filtered images, as the filtered images seek to replicate the behaviour of the noiseless images.

The combined method was prone to overfitting, as shown by the validation results being much poorer than the training results. This is likely due to the model being unable to pick up the features in the wave data, and instead focusing on the idiosyncrasies of the wave training dataset to make correct guesses. Hence, de-noising/feature augmentation of the wave data is an object of further research, as this will significantly improve the results of not only the wave models, but the combined model. However, despite this overfitting, the model still obtained the 3rd smallest validation $\overline{D}$ value. Hence, this model could easily outperform the others if the wave data can be augmented.

The de-noising algorithm clearly performed well, as demonstrated by the models trained on the de-noised images exhibiting superior performance to their noisy counterparts, further demonstrating the direct proportionality between the noise temperature and the average difference between predictions and true values.

The models trained on the wave data alone performed poorly in comparison to all the other methods. This is due to the aforementioned lack to strong features but may also be strongly dependent on the filter size in the case of the convolutional model. The features of interest may be spread out over very large distances. Hence, an object of further research is into the length of the features of interest of the wave data.

IV. Conclusion

Overall, the applicable model (i.e. excluding models trained on idealised noiseless data) with the best performance in terms of accuracy is the convolutional model trained on de-noised data, as this model attained the smallest $\overline{D}$ value. This model should be applied in cases where high accuracy is desired in the determination of the start time and frequency. The data should be prepared by using the same de-noising algorithm if this model is to be applied.

In cases where large numbers of images are to be tested, the dense model trained on the de noised images is the best model to apply, as this model runs much faster than the convolutional model and obtained the next smallest validation $\overline{D}$. The data should be prepared by using the same de-noising algorithm if this model is to be applied.

It was also noted that models would likely perform much worse for data where the chirp slope was steeper, as the edge enhancement Sobel filter is much better at augmenting horizontal edges. This was demonstrated in some of the de-noised images where the chirp line displayed prominent gaps (Figure 7).

This disruption of the chirp line would certainly affect the model predictions if it were trained on data with no gaps (i.e., very horizontal chirp lines). Hence, to ensure the model remains robust, it should be trained on data that includes these gaps.

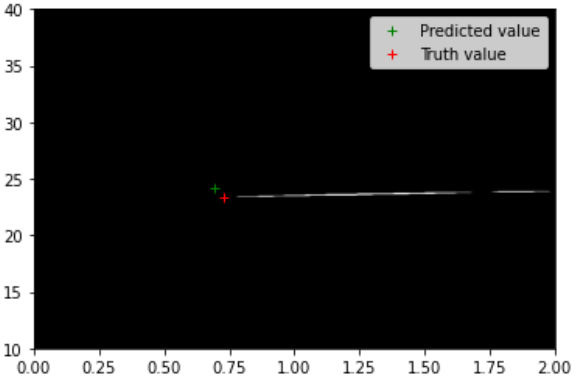


Figure 7: A de-noised wave (with predictions and truth values labelled) displaying gaps in the chirp line due to the slope being too steep for the Sobel filter. The y axis is in units of MHz, the x axis in units of ms.

While the models trained on the wave data performed poorly, there are several objects of further research that can significantly improve performance, such as feature augmentation through de-noising, and larger filter kernels. The combined model also shows promise if the wave data can be augmented. All models could be improved by training them on larger datasets. However, due to the long runtimes of some of the models, this may require significant computational resources. Models could also be improved by introducing regularisation penalties to reduce overfitting, such as Ridge Regression [9].

Overall, the main goals were reached; the other models were trained to suitable losses

on all data types, and the de-noising algorithm provided significant improvements on this.

V. References

[1] J. G. Learned and K. Mannheim, "High-Energy Neutrino Astrophysics," *Annual Review of Nuclear and Particle Science,* vol. 50, no. 200, pp. 679-749, 2000.

[2] E. Giusarma, M. Gerbino, O. Mena, S. Vagnozzi, S. Ho and K. Freese, "Improvement of cosmological neutrino mass bounds," *PHYSICAL REVIEW D,* vol. 94, no. 8, 2016.

[3] A. A. Esfahani, D. M. Asner, S. Böser, R. Cervantes, C. Claessens, L. de Viveiros, P. J. Doe, S. Doeleman, J. L. Fernandes and M. Fertl, "Determining the neutrino mass with cyclotron radiation emission spectroscopy—Project 8," *Journal of Physics G: Nuclear and Particle Physics,* vol. 44, no. 5, 2017.

[4] "Get started with TensorFlow," 2024. [Online]. Available: https://www.tensorflow.org/. [Accessed 16 March 2024].

[5] IBM, "What is computer vision?," 2024. [Online]. Available: https://www.ibm.com/topics/computer-vision. [Accessed 16 March 2024].

[6] IBM, "What are convolutional neural networks?," 2024. [Online]. Available: https://www.ibm.com/topics/convolutional-neural-networks. [Accessed 16 March 2024].

[7] S. Mudadla, "Deep Neural Networks vs Dense Neural Networks," Medium, 3 December 2023. [Online]. Available: https://medium.com/@sujathamudadla1213/deep-neural-networks-vs-dense-neural-networks-bad5918a5b2a. [Accessed 16 March 2024].

[8] "opencv-python 4.9.0.80," 2024. [Online]. Available: https://pypi.org/project/opencv-python/. [Accessed 16 March 2024].

[9] E. Kavlakoglu and J. Murel, "What is regularization?," IBM, 16 November 2023. [Online]. Available: https://www.ibm.com/topics/regularization. [Accessed 21 March 2024].

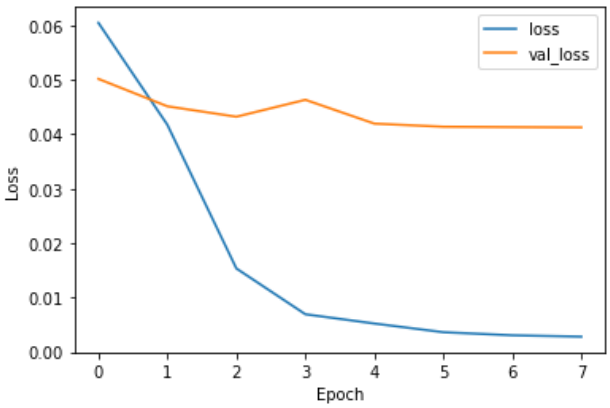
[10] Papers with Code, "Dense Connections," [Online]. Available: https://paperswithcode.com/method/dense-connections. [Accessed 16 March 2024].

Appendix I:

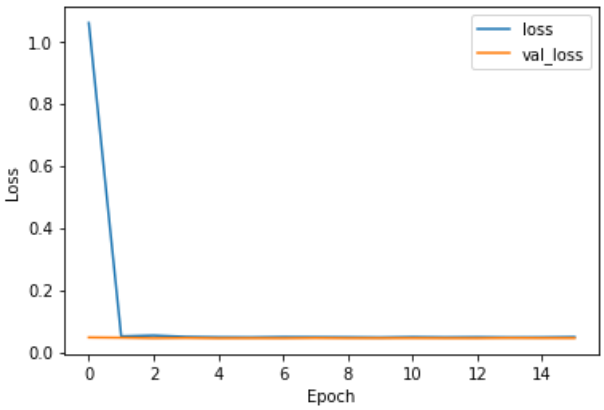
Model	Epochs	Mean time per epoch (s)	Training loss	Training $\bar{D}$	Training σ	Validation loss	Validation $\bar{D}$	Validation σ
Image convolutional	8	31.375	0.0028	0.0453	0.0420	0.0412	0.2486	0.1438
Image dense	16	4.0625	0.0485	0.2722	0.1335	0.0448	0.2662	0.1367
Image* convolutional	8	32.25	0.0034	0.0575	0.0394	0.0054	0.0856	0.0593
Image* dense	16	4.375	0.0030	0.0575	0.0320	0.0090	0.1117	0.0739
Noiseless dense	12	18.75	0.0017	0.0329	0.0201	0.0022	0.0335	0.0209
Wave conv	8	83	0.0855	0.5981	0.2106	0.1838	0.5827	0.1678
Wave dense	16	4	0.0093	0.0846	0.0415	0.0885	0.3996	0.1312
Combined* dense	10	9.3	0.0026	0.0487	0.0269	0.0215	0.1753	0.1105

*Appendix I: The table showing the performance of each model. The most important metric is the Validation $\bar{D}$ parameter (mean distance), as this represents the mean distance from the truth value over the unseen validation data. Models marked with a * represent models trained on de-noised data, using the de-noising algorithm specified in the paper. Hence, the convolutional model trained on the de-noised the data is the model with the greatest accuracy (excluding the model trained on the noiseless data, as this represents idealised data that cannot be replicated in a lab environment). The distances are given in arbitrary units, as they are calculated from the normalised labels. All values have been rounded to 4 decimal places. Note that the noiseless dense model was trained on 9000 images, and the wave only models were trained on 900 waves.*

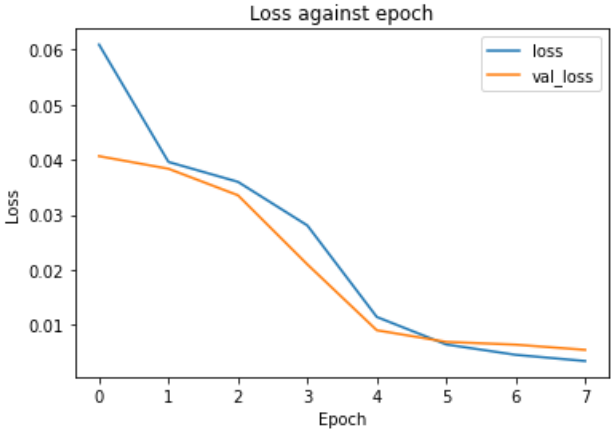
Appendix II:



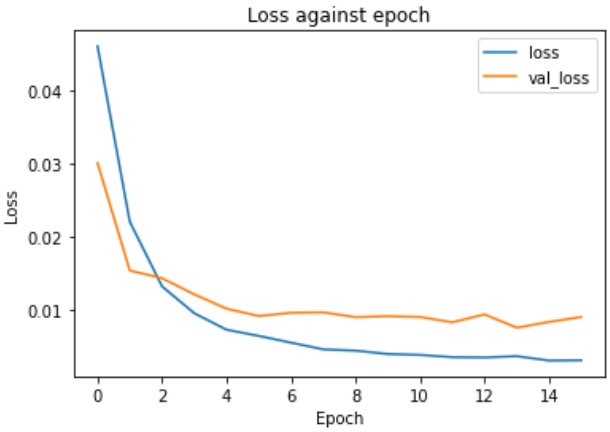
The epoch against loss graph for the convolutional model trained on the noisy data.



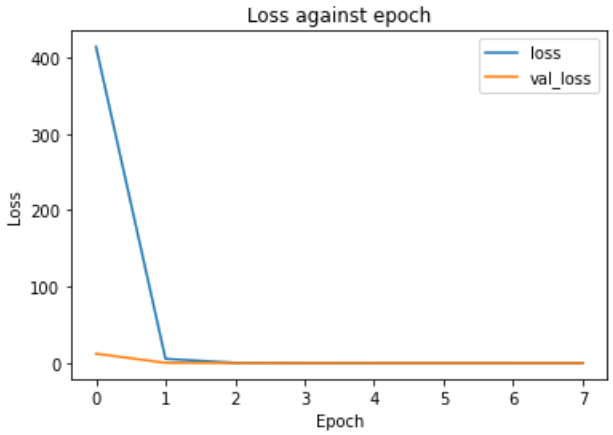
The epoch against loss graph for the dense model trained on the noisy data.



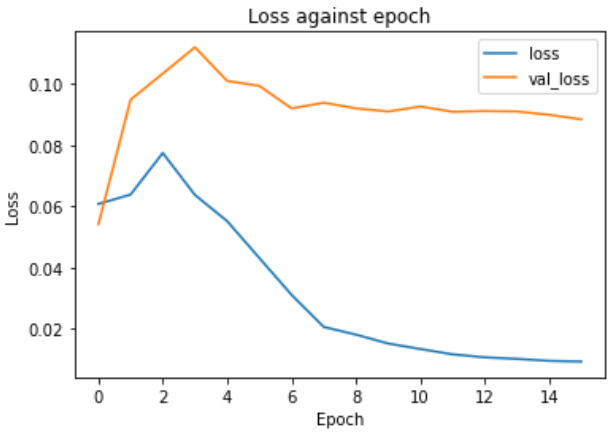
The epoch against loss graph for the convolutional model trained on the de-noised data.



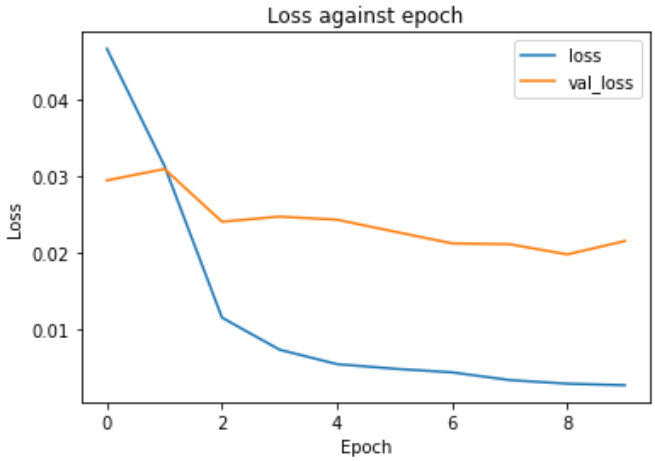
The epoch against loss graph for the dense model trained on the de-noised data.



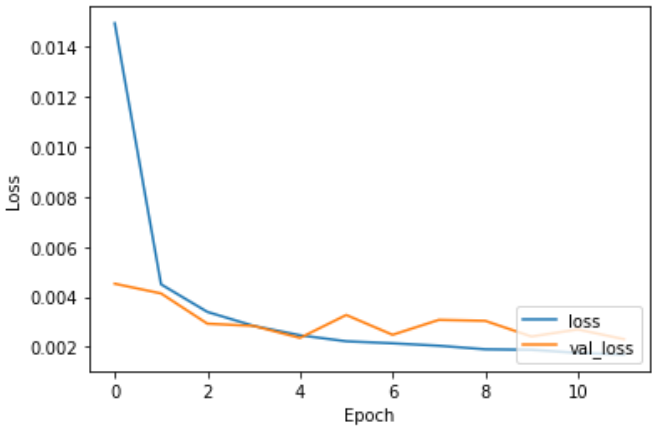
The epoch against loss graph for the convolutional model trained on the wave data.



The epoch against loss graph for the dense model trained on the wave data.



The epoch against loss graph for the dense model trained on the combined data.



The epoch against loss graph for the dense model trained on the idealised noiseless data.