Department of Physics

Physics MSci

Matched Filter Autoencoder: A Physically inspired Neural Network for Waveform Analysis at LUX-ZEPLIN

Author:

Emil Haines

Supervisor:

Henning Flaecher, Ben Krikler

Word Count:

8419

For submission in 2020

Certificate of Ownership

Project Report presented as part of, and in accordance with, the requirements for the Final Degree of MSci at the University of Bristol, Faculty of Science.

I hereby assert that I own exclusive copyright in the item named below. I give permission to the University of Bristol Library to add this item to its stock and to make it available for consultation in the library, and for inter-library lending for use in another library. It may be copied in full or in part for any bona fide library or research worked, on the understanding that users are made aware of their obligations under copyright legislation, i.e. that no quotation and no information derived from it may be published without the author's prior consent.

Signed: Emil Haines

Title: *Matched Filter Autoencoder: A Physically inspired Neural Network for Waveform Analysis at LZ*

Date: 14/04/2020

Declaration

The simulated LZ DER data used in Section 4.2 was created by Ben Krikler, with the help of Sam Eriksen, based on code by Theresa Fruth (UCL). The results obtained by applying the LZ analysis package to this data were supplied by Chris Wright; these are presented in Section 4.2.2. Many of the important ideas behind the architecture of the model presented here, along with its connection to the matched filter technique, were provided by Ben. Throughout the project, guidance and ideas were provided by Henning Flaecher and Ben.

With the exception of the contributions noted above, all the code used during the project was written by myself and my partner; this includes the implementation of the neural network model, the production of results and plots, and the generation of the toy data set used in Section 4.1. We worked on a shared code base, to which we both made significant contributions. Decisions regarding which analyses to conduct were made together, along with Ben. Programming work was often split between us, such as early in the project when my partner wrote the code to implement the model while I wrote code to process the data into a useable form, in order to increase the work rate. However, important decisions regarding the general direction of the project were made together.

Industrial Action and COVID-19

As a result of the recent Industrial Action, my partner and I missed several meetings with our supervisor throughout the time allocated to complete the project, which slowed our progress. As the project was entirely computational and did not require access to a laboratory, the COVID-19 pandemic had a minimal effect on our ability to complete the project.

Abstract

Distinguishing and extracting features of pulses in noisy waveforms is vital in ensuring high sensitivity can be achieved at the next generation dark matter detection experiment LUX-ZEPLIN. In this work, a novel algorithm for waveform analysis at LUX-ZEPLIN, that combines the matched filter technique from traditional signal processing with the versatility of modern deep neural networks, is proposed: the Matched Filter Autoencoder. The algorithm is tested on its ability to detect pulses in waveforms, and recover their timings and amplitudes. The performance of the model is found to compare favourably to benchmarks set using alternative methods.

Contents

List of Figures

1 Introduction

A large fraction of the matter in the universe is thought to consist of a non-luminous, massive component, known as dark matter (DM) [1]. While the existence of DM has been indirectly evidenced by decades of astronomical experiments [2, 3], its true nature represents a major unsolved problem within several areas of physics. Weakly interacting massive particles (WIMPs) are an attractive and well-motivated DM canditate that have become the target of many current experimental searches. This is chiefly due to WIMPs having detection rates realisable using current and planned detectors, as well as the correspondence between the thermal production of WIMPs and observed relic DM abundance, known as the 'WIMP miracle' [4].

WIMP detection generally falls under one of three main methods: particle accelerators, indirect detection, direct direction; this project is primarily concerned with the latter. Direct detection experiments rely on detecting nuclear recoil (NR) due to interactions between DM particles from the halo of the Milky Way and target nuclei in terrestrial deep underground low-background detectors [1]. In recent years, several high profile direct detection experiments have searched for WIMPs in different mass ranges and at increasing sensitivities, with more planned for the future [5].

The focus of this project is the LUX-ZEPLIN (LZ) experiment [6], a direct detection experiment located nearly 1.5km underground at the Sanford Underground Research Facility in South Dakota, that is due to to be commissioned in late 2020. LZ employs a two-phase xenon (Xe) time projection chamber (TPC) with an active mass of 7 tonnes. Photomultiplier tubes (PMT) located in arrays above and below the TPC detect signals caused by scattering events.

Identifying the signals that have resulted from WIMP interactions is a considerable challenge, especially given the presence of problematic background events whose signatures can mimic those of WIMPs [7] and the elusiveness of WIMPs themselves [8]. It is therefore vital to develop sophisticated analysis tools capable of efficiently extracting relevant features from raw signal data, which can aid in identifying and classifying events

with a high degree of accuracy. This task is largely the domain of deep learning (DL), a branch of the broader machine learning class of statistical methods, that has received considerable attention in recent years [9], concerned with learning representations of raw data using multi-layered neural networks (NNs). In particular, DL has been successfully applied to processing and predicting features of time series like signals [10–12]. This is perhaps unsurprising given the close connection between the basic operations of traditional signal processing techniques and those found in modern deep NNs, namely the convolution.

In this report, a DL algorithm for unlabelled waveform analysis, termed the Matched Filter Autoencoder (MF-AE), that favourably exploits this connection by combining a deep NN with the matched filter technique (MFT) is proposed. The performance of the model in reconstructing raw input signals and predicting useful quantities, such as the number of pulses present in a time series signal as well as the timings and amplitudes of those individual pulses, is explored. Two different data sets are used, a toy data set and a set of fake events processed using the LZ Detector Electronics Response (DER); each attempt to emulate the response of a single LZ channel to a low energy event.

The structure of this report is as follows. First, some relevant theory regarding the functionality of and physical processes in LZ (and previous detectors of its type), the MFT and DL is provided in Sections 2.1, 2.2 and 2.3, respectively, in order to motivate the architecture of the model, of which the key features are described in Section 3. The experimental results when applying the model to the toy and LZ DER data sets are contained in Sections 4.1 and 4.2, respectively; the latter are contextualised through a comparison with results generated using the current LZ analysis package. An understanding of the results and their implication, along with some ideas for future work, is discussed in Section 5. Conclusions are drawn in Section 6.

2 Background and Theory

2.1 Direct Detection and LUX-ZEPLIN

The history and projected sensitivities of direct detection experiments for 50GeV WIMPs is shown in Figure 2.1. The >10GeV mass region is considered most probable for a WIMP, as other weakly interacting particles have masses of this order [13].

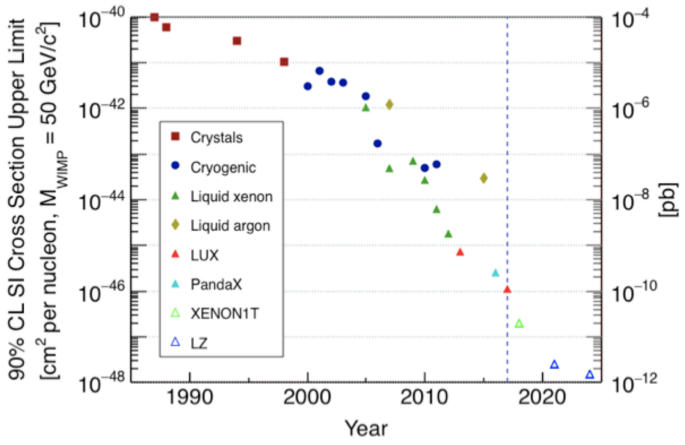


Figure 2.1: The history and projected sensitivities of direction detection experiments for WIMPs with mass of 50GeV. LZ (denoted by the dark blue triangle) is expected to become the most sensitive experiment in this mass range. Reproduced from [13].

Early detectors achieved cross sections of 10^{-41}cm^2 using NaI and Ge crystals, but, unlike later and more successful applications of cryogenic Ge detectors, they were constrained by an inability to distinguish NRs from electron recoils (ERs). Recently, detectors using noble liquids have increased sensitivity at a tremendous rate, allowing for deeper probing into WIMP parameter space. The LUX experiment, the precursor to LZ from which the latter derives much of its experimental strategy, achieved sensitivities of 1.1×10^{-46}cm^2 [14]. Boasting a larger active mass and exposure time, XENON1T reported sensitivities on the order of 10^{-47}cm^2 the year after [15]. LZ is hoped to achieve sensitivities of 10^{-48}cm^2, which would make it the most sensitive experiment in this mass region [13]. One should consult [5] for a comprehensive review of the sensitivities of direct detection experiments.

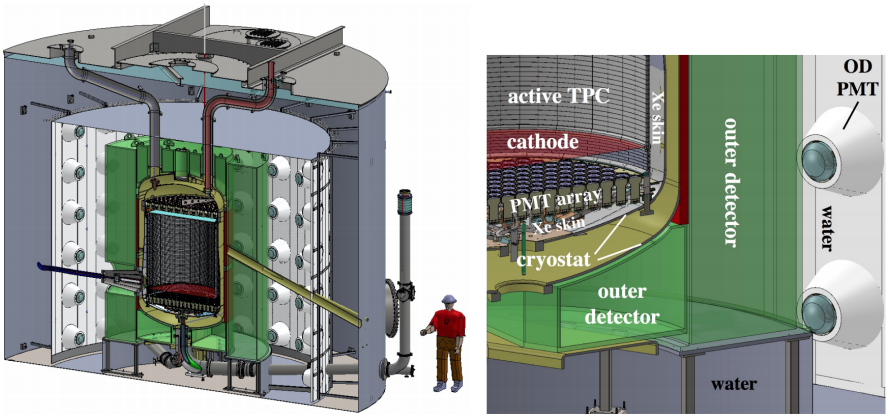


Figure 2.2: Schematic of the LZ detector system. Reproduced from [6].

Figure 2.2 shows a schematic of the LZ detector system. Of the 10 tonnes of Xe contained within the cryostat, 7 tonnes are in the two-phase (liquid and gas) Xe TPC (5.6 tonnes of which form the fiducial region), with the rest in the surrounding skin veto. A room temperature liquid scintillator outer detector (OD) maintains a constant temperature within the cryostat. Both parts of the detector are enclosed by a large water tank. The design of LZ is described in detail in the Technical Design Report [13].

Within the TPC, particles scatter off Xe atoms and induce NRs or ERs, causing excitation or ionisation of the Xe atoms, triggering the production of two signals [8]. Xe atoms that are excited through impact with a recoiling nucleus or electron promptly emit scintillation light; this signal is referred to as S1. An electric field extracts electrons released through ionisation from the interaction site to a gaseous layer above the surface of the detector where a second scintillation signal, referred to as S2, is generated by electroluminescence. Electrons can also recombine with a Xe ion, producing more S1. A schematic showing the energy deposition processes involved in LZ is shown in Figure 2.3. Both signals are detected by PMTs, which are organised into arrays above and below the detector [16]. Most of the S1 signal is spread uniformly across the bottom array, due to the reflection of light at the liquid-gas surface. In LZ, between 3 and 30 photons are expected to be detected for an S1 signal [13]. S2 signal is mostly concentrated around a single PMT in the top array, with the remainder spread uniformly across the bottom array. Several hundred photons are expected to be detected as a result of an S2 signal [13].

The ability to detect and distinguish S1 and S2 signals is an integral part of the back-

ground suppression strategy at LZ. Precise 3D event location is achieved using the time difference between the recorded S1 and S2 signals and the pattern of S2 signal on the top array, which allows for fiducialisation. Different particles deposit varying amounts of energy as scintillation or ionisation, therefore the S2/S1-ratio can be used to discriminate NR events and ER backgrounds [6]. This is generally achieved by looking at S2/S1 versus S1 for each event; the two types of recoil form two distinct bands in this phase space.

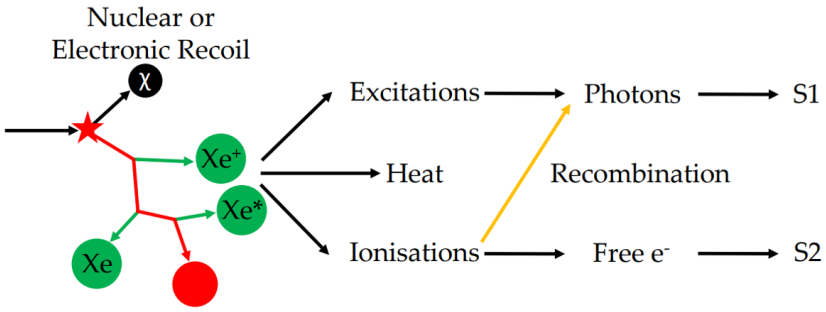


Figure 2.3: Deposition of energy due to electron and nuclear recoils in liquid Xe. The recoiling particle (red) is a Xe atom for a nuclear recoil and an electron for an electron recoil. Reproduced from [17].

If DM is formed of WIMPs, a significant number should be passing through the Earth at any time. They should scatter off nuclei in the detector via a weak interaction and induce a NR. NR events due to WIMPs are incredibly rare, as a result, a WIMP signal must be consistent with a single energy deposit, which, for a WIMP with mass $10-1000$GeV travelling at galactic speeds $\mathcal{O}(10^{-3}c)$, is <100keV [18]. This makes unambiguously identifying a WIMP very difficult. The addition of backgrounds whose signals are readily mistaken for WIMP induced NRs, such as beta emission from radon daughter nuclei and neutrons [7, 8], as well as the finite accuracy in discriminating ERs, further limit the sensitivity.

Several important aspects of the design of LZ help mitigate background events and reduce uncertainty in identifying WIMP signals. These include: carefully chosen detector materials, an active veto system, the self-shielding properties of XE, NR/ER discrimination and the deep underground location [8, 13]. However, as background signals are never fully suppressed, the need for effective signal processing techniques that can help classify events directly from raw PMT output waveforms is highly necessary.

2.2 The Matched Filter

The MFT is used to search for and estimate parameters of pulses of a known shape, called a template, **T**, within a signal containing noise [19].

2.2.1 Single Pulse Waveforms

The raw output signal of a PMT channel in LZ is a time series waveform. The i-th sample S_i of such a signal containing a single pulse (i.e. the result of one photon being absorbed by the PMT) can be modelled by

$$S_i = aT_{i-k} + \epsilon_i, \tag{1}$$

where T_{i-k} is the contribution from a pulse of known shape (that forms part of **T**) that occurred at some time k, a is a scaling parameter, and ϵ_i is additional stochastic noise. The function of the MFT is to find the two optimum parameters a and k. The MFT provides the filtered output with the maximised signal-to-noise ratio, from which these parameters can be derived [19, 20]. This is found by normalising **T**, then correlating it with the signal. The i-th sample of the filtered output **F** is therefore given by

$$\begin{aligned} F_i &= \sum_j S_j T_{j-i} \\ &= a \sum_j T_{j-k} T_{j-i} + \sum_j \epsilon_j T_{j-i}, \end{aligned} \tag{2}$$

after substituting equation 1. The two terms in the filtered output correspond to the autocorrelation function of **T** and the noise of the output itself. At $i = k$, i.e. when the offset between the two templates in the correlation is zero, the autocorrelation $\sum_j T_{j-k} T_{j-i}$ is maximised, and therefore so is **F**. At this point, F_i is equal to a (given the normalisation of **T** and ignoring the noise term). The optimal shift k and scaling a (to which the true

amplitude of the pulse is proportional) can therefore be inferred as [20]

$$\begin{aligned} k &= \underset{i}{\operatorname{argmax}}(F_i) \\ a &= \max_i(F_i). \end{aligned} \tag{3}$$

Correlating the signal with the template in this manner to obtain the matched filter output can be motivated by trying to find the values of a and k that scale and shift $\mathbf{T}$ such as to minimise the residual sum of squares (RSS) between it and the waveform; this is described in Appendix A.

The problem of extracting information about the timing and amplitude of a pulse in a noisy waveform is integral to particle physics experiments such as LZ; the MFT provides a method to do so. However, several factors can make this more challenging. For example, the addition of a constant pedestal to the signal requires the use of a template that has been orthogonalised with respect to the pedestal.

2.2.2 Pile-up Waveforms

A further case to consider is pile-up, which occurs when multiple photons are detected by a single channel within its response time, producing waveforms containing overlapping pulses. This is particularly relevant to LZ, as pile-up would be expected in certain signals, such as S2, where energy deposits are concentrated around a single PMT channel [16].

The i-th sample of a waveform containing two pulses of the same shape occurring at times k and l can be modelled by

$$S_i = aT_{i-k} + bT_{i-l} + \epsilon_i. \tag{4}$$

The i-th sample of the output of the matched filter then becomes

$$F_i = a\sum_j T_{j-k}T_{j-i} + b\sum_j T_{j-l}T_{j-i} + \sum_j \epsilon_j T_{j-i}. \tag{5}$$

Ignoring the error terms, as they are assumed to sum to zero, the output of the matched filter at the true pulse times is given by

$$\begin{pmatrix} F_k \\ F_l \end{pmatrix} = \begin{pmatrix} |\mathbf{T}|^2 & \sum_j T_{j-l} T_{j-k} \\ \sum_j T_{j-k} T_{j-l} & |\mathbf{T}|^2 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix}. \tag{6}$$

Solving this pair of equations for a and b requires values for k and l.

For a waveform containing a single pulse, the correct timing and amplitude scaling could be found by identifying the position and value of the maximum of the filtered output. The same approach can not be taken for waveforms containing a pile-up. At the timings at which the pulses occurred, the output of the filter now not only receives contributions from the pulse at that point, but also from any overlapping pulses, by an amount dependent on the separation between the pulses. The values F_k and F_l can therefore not be equated with the optimum amplitude scaling for pulses at k and l, unlike the single pulse case. This can be resolved by correlating the signal with a short filter, that that mimics the sharp rising edge of $\mathbf{T}$, able to separate pulses within the waveform, and inferring the values of k and l from the maxima of this output [20]. (This approach assumes the pulse being detected has a steep rising edge.) Equation 6 can then be solved by inverting and pre-multiplying by the matrix on the right hand side, which unfolds the two overlapping pulses and gives the values that optimally scale their individual contributions to the signal, a and b. This process can be extended for pile-ups containing > 2 pulses by increasing the rank of the matrix, however this would make the calculation more costly.

It has been shown that by using multiple correlation filters, a rising edge pulse finding filter whose output recovers the timings and a matched filter whose output helps recover the the amplitude scalings, it is possible to extract useful features of pulses in pile-up signals. The technique presented here is fast and well suited for real time use, however, is not without its limitations. The most apparent rests on the assumption that the true shape of the underlying pulses, $\mathbf{T}$, is known; in real life particle physics experiments, this is unlikely to be the case and the template would have to be explicitly determined.

2.3 Deep Learning

Machine learning has become a popular tool for statistical analysis within particle physics, prompted by a growing need to reduce the dimensionality of data without the loss of information [21, 22]. Event reconstruction and classification, trigger systems and simulations have all benefited from the use of machine learning.

2.3.1 Deep Neural Networks

DL is a branch of machine learning that employs the use of NNs with multiple layers to gain useful representations of data [23]. The number of layers in a NN is often referred to as its depth, hence deep NNs. From here onwards, a complete NN structure is referred to as a model, and the organisation of the layers in a NN as its architecture.

The layers in a NN are parameterised by a set of weights, $\mathbf{w}$, and each possess an associated operation, such as a matrix multiplication or a non-linear activation function. Each intermediate operation defines in part a function mapping some input, $\mathbf{x}$, to an output layer, that returns $\mathbf{y} = f(\mathbf{x}; \mathbf{w})$. The NN must learn the values of $\mathbf{w}$ for which $f(\mathbf{x}; \mathbf{w})$ best approximates a target function, $f^*(\mathbf{x})$, that is capable of performing a specific task.

The process by which a NN learns $\mathbf{w}$ is called training. Training data provides each input sample $\mathbf{x}$ with a label $\mathbf{y^*} = f^*(\mathbf{x})$ that specifies the intended result of the output layer. For example, if the task was to classify a particular input, $\mathbf{y^*}$ would denote the true class that the sample belongs to. As their desired behaviour is not explicitly dictated by the training data, layers beside the output layer are called hidden layers [23]. The error between the intended output $\mathbf{y^*}$ and actual output $\mathbf{y}$ of the output layer for a particular set of $\mathbf{w}$ can be quantified by a loss function $\mathcal{L}(\mathbf{y}, \mathbf{y^*}; \mathbf{w})$.

During training $f(\mathbf{x}; \mathbf{w})$ is driven towards $f^*(\mathbf{x})$ by updating the weights such as to minimise the loss function. This can be achieved using gradient descent optimisation [23]. Taking the gradient of the loss function with respect to $\mathbf{w}$, $\nabla_{\mathbf{w}}\mathcal{L}(\mathbf{y}, \mathbf{y^*}; \mathbf{w})$, gives the direction in which the loss increases fastest with respect to changes in the weights, the weights are therefore updated in the opposite direction. Often, including in this work,

weights are updated iteratively using subsets of the training data, called mini-batches. This has the advantage of inducing less variance in parameter updates and therefore a more stable convergence than gradient descent methods for which the loss is calculated entirely stochastically. A complete cycle through these mini-batches is called an epoch. Multi-layer models use the back-propagation algorithm to ensure gradients are efficiently calculated through multi-layered networks [24].

Modern machine learning packages tend to use more complex weight optimisation methods that described above, such as the ADAM optimiser [25] (used in this work), which often incorporate gradient descent in some manner. Readers are advised to consult [25] for a more in depth look at the specific optimisation procedure used here.

2.3.2 L_1 and L_2 Regularisation

It is hoped that once a model has been trained, its aptitude for a particular task will be high when tested on data that was not part of the training set, this is known as generalisation [23]. Alterations to the learning process of a model can help improve generalisation or encourage the model to behave in a certain way, this process is called regularisation. Regularisation strategies vary, and can impact, for example, the type of data used, the model architecture or the loss term. Two methods of regularisation integral to the model introduced in Section 3 are L_1 and L_2 regularisation.

L_1 and L_2 regularisation are related in that they both penalise the performance of a model during training by adding additional terms to the loss function. However, their effects are distinct. An appreciation of these effects is all that is required for this work, hence a full mathematical description is omitted; this can be found in [23].

L_2 regularisation incorporates a penalty into the loss function that encourages the sum of the squared absolute values of a parameter to be small. In this case, the loss function becomes

$$\mathcal{L}_{L_2}(\mathbf{y}, \mathbf{y^*}, \mathbf{w}) = \mathcal{L}(\mathbf{y}, \mathbf{y^*}, \mathbf{w}) + \lambda \sum_i |p_i|^2, \tag{7}$$

where $\mathcal{L}(\mathbf{y}, \mathbf{y^*}, \mathbf{w})$ is the loss prior to the addition of regularisation, p_i are the parameters

being regularised (which can be, for example, the output of any intermediate layer or its weight vector) and λ provides the strength of the regularisation (larger λ produces greater changes to $\mathbf{p}$).

L_1 regularisation incorporates a penalty into the loss function that encourages the sum of the absolute values of a parameter to be small. The loss function to be minimised becomes

$$\mathcal{L}_{L_1}(\mathbf{y}, \mathbf{y}^*, \mathbf{w}) = \mathcal{L}(\mathbf{y}, \mathbf{y}^*, \mathbf{w}) + \lambda \sum_i |p_i|, \tag{8}$$

where symbols are defined in the same manner as in equation 7. While both L_1 and L_2 help to shrink the values of the parameters, L_1 regularisation also causes the parameter vector to become sparse. This makes it a suitable option for extracting useful features from data, such as waveforms, as less important features are suppressed.

The parameter λ in equations 7 and 8 is referred to as a hyperparameter. Hyperparameters are chosen independently of the training process and are used to control the behaviour of a model [23]. Other hyperparameters include the size of the mini-batches used during optimisation and the number of epochs completed before training is stopped.

2.3.3 Convolutional Neural Networks and Autoencoders

As they are fundamental to the architecture described below in Section 3, it is sensible to briefly describe two specific types of NN, convolutional neural networks (CNNs) and autoencoders (AEs), as well as explain their applicability to waveform analysis.

CNNs replace general matrix multiplications in at least one of their layers with a convolution. They are effective when tasked with extracting high-level features from grid-like data, such as a time-series waveform [23]. They are therefore well suited for signal-versus-background detection, reconstruction and classification problems, such as those involving image data from particle physics experiments [26].

A typical CNN structure consists of stacked convolutional, pooling and fully-connected layers and additional non-linear activation functions. A convolutional layer, in the context of this work, comprises a vector (or kernel) of trainable weights, whose length is a

hyperparameter of the model, that is convolved with the input of the layer to produce an output. A trainable bias is added to the output, before it is fed into the next layer. Modern CNNs can have significant depth and an incredibly high number of trainable parameters. They are able to achieve near human performance in classification and detection tasks [27]. However, the interpretability of these models can suffer as a result. When designing a model for physical analysis, interpretability is an important factor to consider. Not only is it desirable for a model to accurately produce a certain output, an intuitive explanation as to why the model has learnt the parameters that generate this output can provide further evidence that a model is working as intended or help when fine-tuning its architecture to ensure it does.

An AE is a type of neural network that chiefly consists of two parts: an encoder that maps the input $\mathbf{x}$ into a lower dimensional hidden 'encoded' layer, $\mathbf{e} = f(\mathbf{x})$, and a decoder that attempts to produce a reconstruction of the input, $\mathbf{r} = g(\mathbf{e})$, using the encoded representation [23]. The outputted reconstruction has the same dimension as the input. The model is trained to produce accurate reconstructions of the input; as a result, the encoded representation is forced to contain the most useful features of the data. An AE can be trained using the mean squared error

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(x_i - r_i)^2, \tag{9}$$

where n is the length of the input vector, between its inputs and reconstructions as its loss function. The training is therefore unsupervised, as it requires no labels for the data (as opposed to supervised training, which does).

3 Matched Filter Autoencoder

Sections 2.2 and 2.3 highlight clear parallels between the operations of the traditional MFT and CNNs. Each exhibit the use of multiple layers of convolutional[1] transformations to

[1]The correlations of the MFT are equivalent to convolutions with time reversed templates.

elicit useful features from raw data. The kernels of the convolutional layers of a CNN can be interpreted as filters like those in the MFT, the values of which can be learnt during training. It is by this connection that the architecture of the model presented in this section, the MF-AE, is inspired.

The architecture for the MF-AE is shown in Figure 3.1. The model was developed using Tensorflow [28] and Keras [29]. The MF-AE is a fully-convolutional autoencoder that takes as its input a waveform $\mathbf{x} \in \mathbb{R}^n$, and outputs a reconstruction $\mathbf{r} \in \mathbb{R}^n$. Each of the three convolutional layers in the MF-AE are parameterised by a kernel of weights and a bias vector, which are both optimised during training. In order to compare $\mathbf{x}$ and $\mathbf{r}$ and calculate the loss, convolutions in the MF-AE must produce outputs that are the same length as their inputs. This is achieved by padding the end of the input to a convolutional layer with kernel size m with $m-1$ zeros. The convolution between some input to a convolutional layer $\mathbf{y} \in \mathbb{R}^n$ and its kernel $\mathbf{w} \in \mathbb{R}^m$, $\mathbf{y} * \mathbf{w}$, is therefore defined such that the i-th element of its output is

$$(\mathbf{y} * \mathbf{w})_i = \sum_{j=0}^{m-1} y'_{i+j} w_j. \tag{10}$$

where y'_i is an element of the padded input $\mathbf{y}' \in \mathbb{R}^{n+m-1}$. The bias is added to the output of the convolution.

The MFAE, and its connection to the MFT, is best understood by appreciating the operations contained therein (see Figure 3.1) and their intended functions.

First, the waveform $\mathbf{x}$ is fed into two convolutional layers, that run in parallel. The 'Amplitude' layer has a kernel of weights $\mathbf{w}^A \in \mathbb{R}^{m^A}$ which are intended to replicate the matched filter template $\mathbf{T}$, the true pulse shape as defined in Section 2.2.1, after training. L2 regularisation is applied, using equation 7, to the derivatives of $\mathbf{w}^A$; the derivatives are approximated such that $p_i = w^A_{i+1} - w^A_{i-1}$. The L2 parameter is defined as λ_{L2}. The regularisation encourages the derivatives to take on small values, ensuring $\mathbf{w}^A$ becomes smooth during training (as one would expect of a pulse that had not been corrupted by

noise). The convolution of $\mathbf{x}$ with $\mathbf{w}^A$ produces the output

$$\mathbf{h}_1 = \mathbf{x} * \mathbf{w}^A + \mathbf{b}^A, \tag{11}$$

(where $\mathbf{b}^A \in \mathbb{R}^n$ is the bias vector for this layer) which can be associated with the output of the matched filter $\mathbf{F}$ (see Section 2.2.2).

The 'Pulse Finder' layer has a kernel of weights $\mathbf{w}^{PF} \in \mathbb{R}^{m^{PF}}$. In general, $m^{PF} < m^A$. This layer performs a similar function to the pulse finding filter in the MFT (see Section 2.2.2). The output of the layer

$$\mathbf{h}_2 = \sigma(\mathbf{x} * \mathbf{w}^{PF} + \mathbf{b}^{PF}) \tag{12}$$

(where $\mathbf{b}^{PF} \in \mathbb{R}^n$ is the bias vector for this layer) should consist of large values at points where pulses have occurred within the waveform, and be small everywhere else. Sigmoid activation is used to extremise the output, makings the maxima more pronounced.

Element wise multiplication between $\mathbf{h}_1$ and $\mathbf{h}_2$ followed by a leaky ReLU function [30] produces

$$\mathbf{h}_3 = \mathrm{ReLU}(\mathbf{h}_1 \odot \mathbf{h}_2). \tag{13}$$

This step uses the maxima in the pulse finding output to identify the correct amplitude related samples from the filtered output to use in the unfolding, akin to identifying the elements F_k and F_l of $\mathbf{F}$ in the example in Section 2.2.2.

A third convolutional layer, named 'Unfold', has a kernel of weights $\mathbf{w}^U \in \mathbb{R}^{m^U}$, where $m^U = m^A$, which are convolved with $\mathbf{h}_3$ to give

$$\mathbf{h}_4 = \mathbf{h}_3 * \mathbf{w}^U + \mathbf{b}^U, \tag{14}$$

where $\mathbf{b}^U \in \mathbb{R}^n$ is the bias vector for this layer. This step is intended to reproduce the matrix inversion described in Section 2.2.2, and therefore separate the contributions from overlapping pulses in the filtered output, such that the amplitude scalings for each pulse

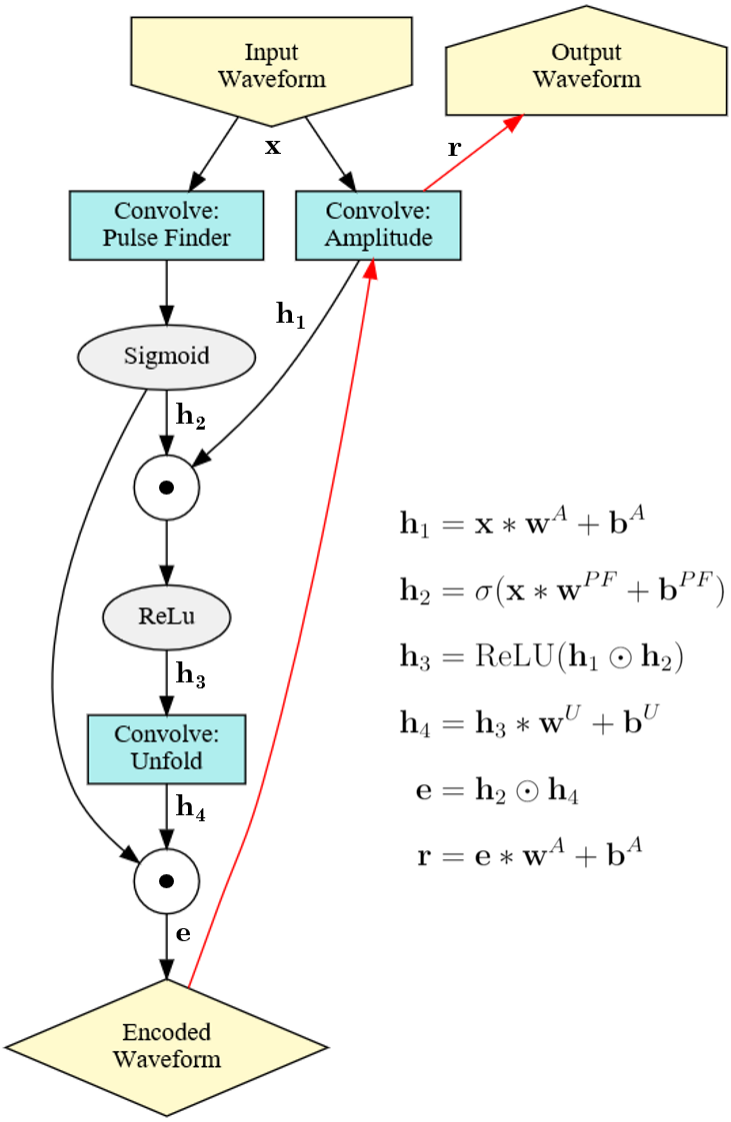


Figure 3.1: Architecture of the MF-AE, along with the operations that define the network. Circles containing black dots indicate element-wise multiplication of the inputs. The black arrows form part of the encoder while the red arrows form part of the decoder.

can be recovered.

The encoded representation is then given by

$$\mathbf{e} = \mathbf{h}_2 \odot \mathbf{h}_4. \tag{15}$$

This step selects the amplitude scalings from $\mathbf{h}_4$ at the correct timings, using the maxima of $\mathbf{h}_2$. L1 regularisation is applied to $\mathbf{e}$. The L1 parameter is defined as λ_{L1}. By encouraging sparsity, less important features should be removed from $\mathbf{e}$. The resulting vector should

consist of peaks in an otherwise sparse vector. The number of peaks should correspond to the number of pulses in the waveform, while the timings and amplitudes of these peaks should be proportional to the timings and amplitude scalings of the true underlying pulses in the waveform. Therefore, through accessing $\mathbf{e}$, one is able to extract relevant features of the underlying pulses in the waveform, as predicted by the MF-AE. Unlike regular AEs, $\mathbf{e}$ is the same dimension as $\mathbf{x}$ in this instance. However, by imposing sparsity in $\mathbf{e}$, its dimensionality has in effect been reduced.

A final convolution between $\mathbf{e}$ and $\mathbf{w}^A$ produces the reconstructed waveform

$$\mathbf{r} = \mathbf{e} * \mathbf{w}^A + \mathbf{b}^A. \tag{16}$$

This can be interpreted as generating the pulses encoded in $\mathbf{e}$ by scaling and shifting the learnt true underlying pulse shape (contained in $\mathbf{w}^A$) and summing each of their contributions, to recover the input.

4 Experiments

In this section, the efficacy of the MF-AE for event detection and feature extraction is studied. Two data sets are used, a toy data set and data from the LZ DER. Details of how each data set is generated, along with their respective results, are divided into Sections 4.1 and 4.2.

In LZ, S1 signal should spread uniformly across a PMT array, therefore, since $\mathcal{O}(10)$ photons are detected as a result of an S1 signal, one might expect individual PMTs to detect only a small number of photons, producing signals with as many pulses. As the detection of S1 signals plays an important role in background suppression (see Section 2.1), pulse finding tools used at LZ must therefore be able to detect and disambiguate these types of signals. Hence, each data set used here to test the MFAE consists of waveforms containing either one or two pulses, as well as additional noise. This is intended to approximate the response of a single LZ PMT channel to an S1 signal triggered by a single scatter WIMP

interaction. The main characteristic of an S1 signal is its steep appearance, resulting from the promptness of the scintillation.

4.1 Toy Data

4.1.1 Data Processing

The toy data set consists of waveforms generated by convolving a sparse vector, populated only at samples where pulses occur, with a template approximating the shape of a prompt S1 signal. A single photon (or pulse) event would have one non zero sample, while a double photon event would have two. The values of these samples are A_{true} and are what scale the template the produce the pulses in the waveform (synonymous with a and b in equations 1 and 4). They are normally distributed around some chosen value. In waveforms containing two pulses, the second pulse is offset uniformly with respect to the second (by between zero and twenty samples). Normally distributed noise was added to each sample both before and after the convolution. This more realistically models the noise in the detector, as the result is no longer Gaussian. The post-convolution noise is denoted by ϵ. A more in depth look at the algorithm used to generate the toy data can be found in Appendix B, along with details of the specific parameters used here.

4.1.2 Toy Data Results

The model in this section used $m^A = m^U = 90$ and $m^{PF} = 12$. Bayesian optimisation [31, 32] was used to determine the optimal hyperparameters λ_{L1} and λ_{L2}, which were found to be $\lambda_{L1} = 0.72$ and $\lambda_{L2} = 15$. The weights $\mathbf{w}^A$, $\mathbf{w}^{PF}$ and $\mathbf{w}^U$ were initialised using Glorot initialisation [33]. The MF-AE was trained using the ADAM optimiser with 4375 mini-batches of 32 waveforms, with post-convolution noise $\epsilon = 2$, for 10 epochs. The performance of the model was validated using 1249 mini-batches, to produce the results presented here.

Figure 4.1 contains examples of raw, encoded and reconstructed waveforms, along with the true and predicted number of pulses they contain. The reconstructions waveforms

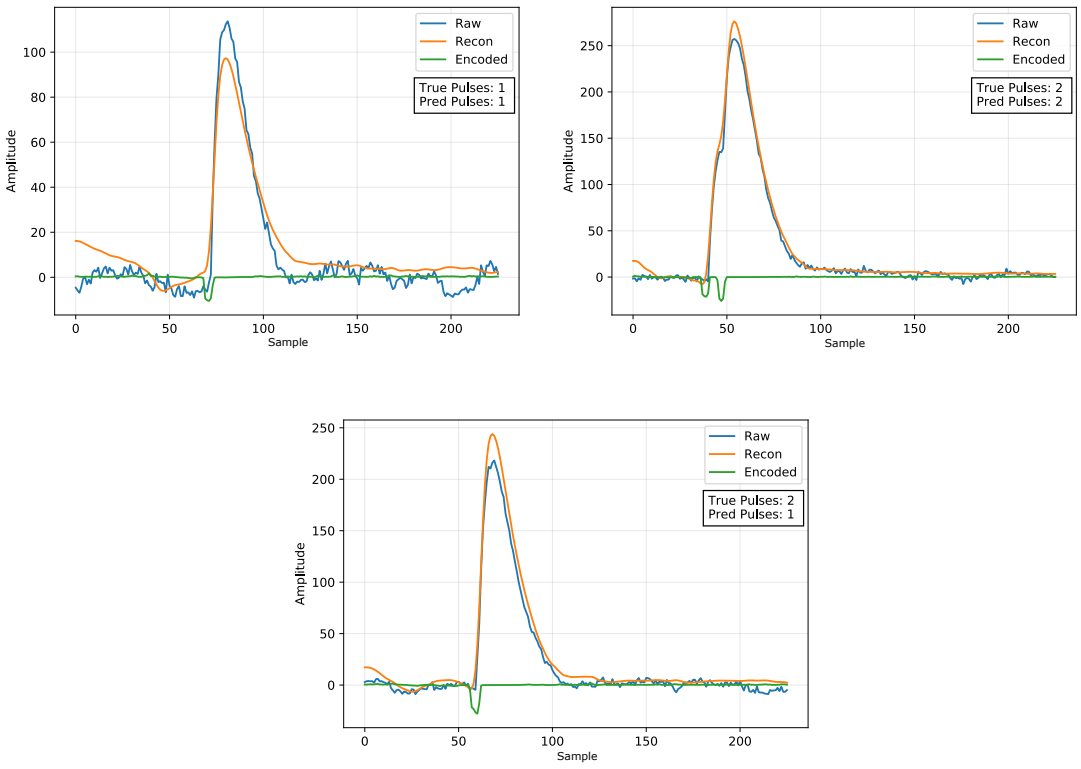


Figure 4.1: Examples of raw (blue), encoded (green) and reconstructed (orange) waveforms. The encoded representation is of the desired form: sharp peaks where pulses are predicted to have occurred, but otherwise sparse. The true and predicted number of pulses contained in the waveforms are given. The prediction of the MF-AE is correct for the top two plots, but incorrect for the bottom.

appear to closely approximate the raw waveform, but with less noise. The output of the encoded layer is of the intended form: a sparse vector with delta function like peaks corresponding to pulses. The top left example demonstrates the mapping of a single pulse waveform, to a single peak encoded representation. The ability of the MF-AE to extract two separate pulses from a raw waveform with only a single discernible peak is demonstrated in the top-right example. However, as can be seen in the bottom example, this is not always the case. Nonetheless, these examples do indicate that, without the use of any truth information during training, the model is able to learn useful representations of the data.

Figure 4.2 shows the learnt weights for the 'Amplitude', 'Pulse Finder' and 'Unfold' filters, for this particular model. As intended, $\mathbf{w}^A$ has clearly taken on the shape of a

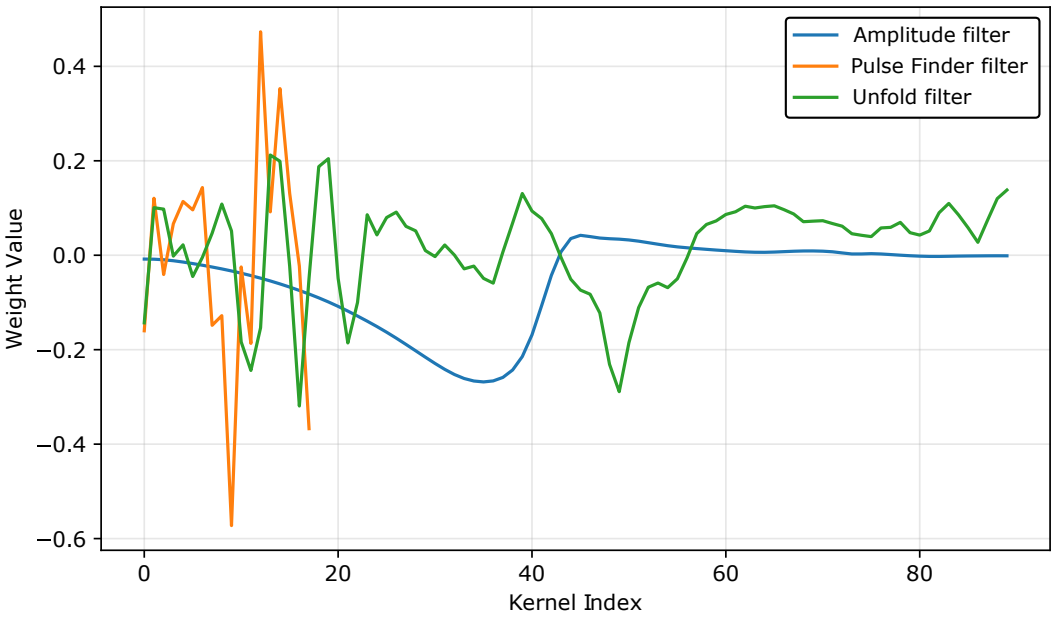


Figure 4.2: The learnt weights for the 'Amplitude', $\mathbf{w}^A$ (blue), 'Pulse Finder', $\mathbf{w}^{PF}$ (orange) and 'Unfold', $\mathbf{w}^U$ (green) filters, for the model trained using toy data.

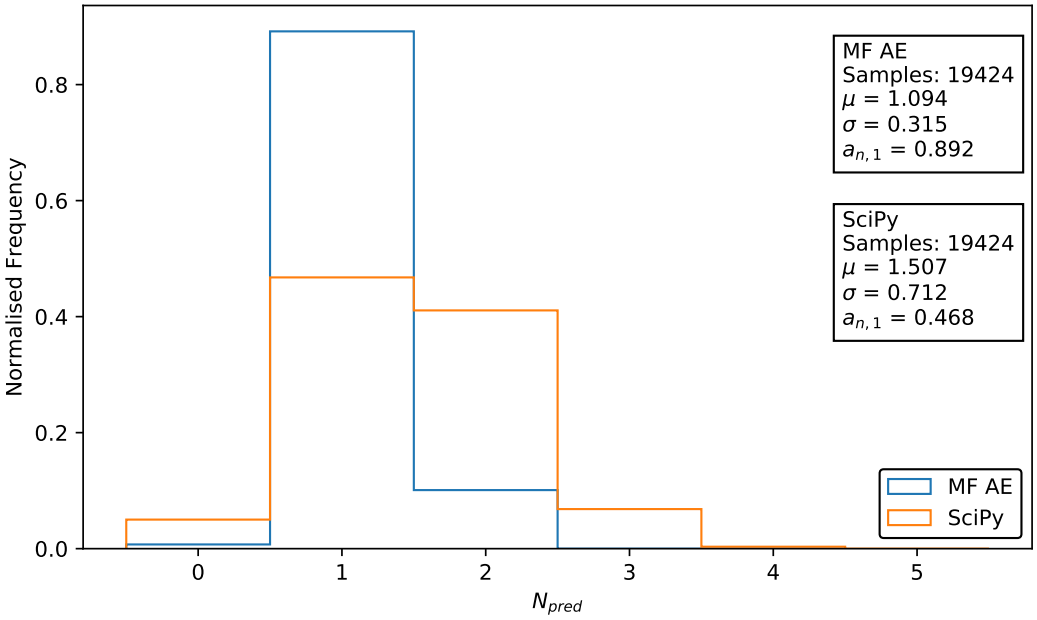
smooth pulse with a steep rising edge, accounting for the time reversal introduced by the convolution. The sharp characteristics of $\mathbf{w}^{PF}$ are perhaps related to the rising edge filter used in the MFT, and are what would be expected of a filter being used for peak finding; the slightly jagged appearance could be the result of noise. It is, however, less obvious what is being shown by the learnt weights of $\mathbf{w}^U$. This is perhaps unsurprising as this layer was intended to reproduce the matrix inversion and pulse separation step in the MFT, for which which their is no corresponding filter to compare, unlike $\mathbf{w}^A$ and $\mathbf{w}^{PF}$. There are, however, clear features of $\mathbf{w}^U$, such as the symmetrical peak near the base of $\mathbf{w}^A$, which could be further understood.

To ascertain whether the encoded space contains physically meaningful information, one must measure how closely parameters extracted from the encoded space match the underlying truth. The MF-AE is being tested on its ability to predict: the number of pulses, N_{pred}, the timings of the pulses, T_{pred}, and the amplitudes of the pulses, A_{pred}, in a waveform. These predictions are compared with the the truth information N_{true}, T_{true} and A_{true}.

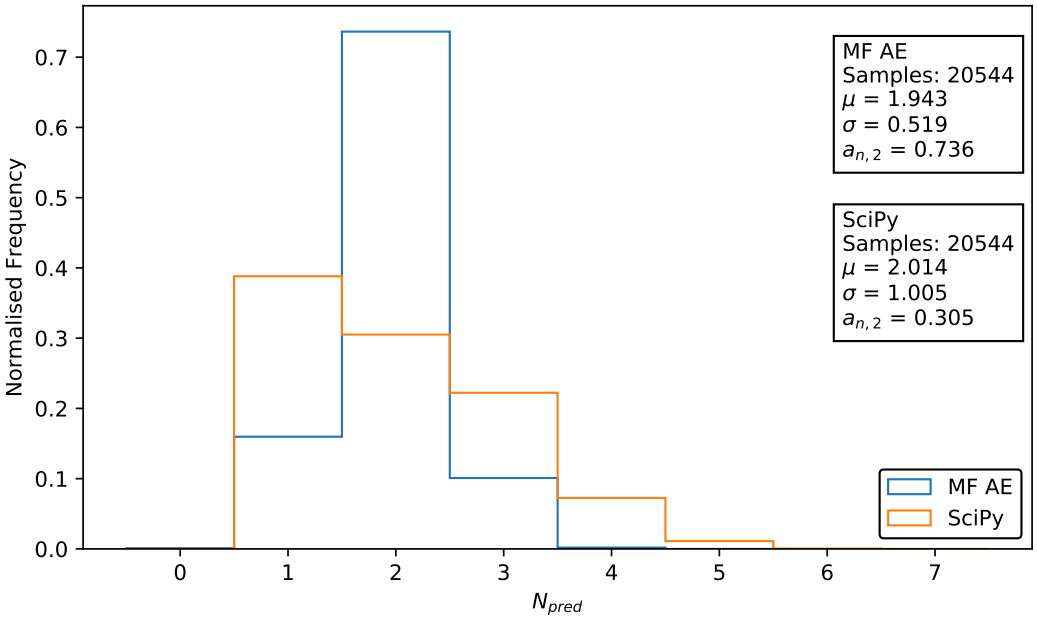
To determine N_{pred}, T_{pred} and A_{pred} for a waveform, the SciPy [34] `find_peaks` function, which finds local maxima above a certain height threshold in a signal by comparing each sample with its neighbours, was applied to the encoded layer. This method was calibrated by finding the integer-valued height threshold that optimises the numerical accuracy, a_n, the proportion of waveforms in a data set for which N_{pred} is correct, using a subset, ~ 300 mini-batches, of the training data. This method of calibration introduces some supervision into the learning process. However, it is possible that sensible estimations for the height threshold could be derived using a smaller quantity of labelled data, however this would need to be explored. To set a benchmark for the performance of the MF-AE, the same SciPy function was applied directly to the raw data; these results are labelled 'SciPy' in the figures. The same calibration method was used to find its optimal height threshold.

Figures 4.3a and 4.3b show the distribution of N_{pred} for single and double pulse waveforms using both the MF-AE and the SciPy benchmark. The numerical accuracy for single pulse waveforms, $a_{n,1}$ and double pulse waveforms, $a_{n,2}$ was calculated to be 89.2% and 73.6%, respectively, using the MF-AE. This represents a vast improvement on simply using the peak finding algorithm on the raw waveforms, which resulted in $a_{n,1} = 46.8\%$ and $a_{n,2} = 30.5\%$. The overall numerical accuracy for this model was $a_n = 81.2\%$. The higher proportion of false predictions for waveforms containing two pulses can largely be understood by looking at Figure 4.4.

Figure 4.4 shows numerical accuracy versus separation between pulses in two pulse waveforms. As the separation between the two pulses in the waveform decreases, both methods become less accurate in predicting the correct number of pulses in the waveform. This is understandable, as it is a significant challenge to distinguish waveforms containing two very closely overlapping pulses. This contributes to the lower values of $a_{n,2}$.



(a)



(b)

Figure 4.3: Normalised distributions of N_{pred} for: (a) waveforms containing one underlying pulse and (b) waveforms containing two underlying pulses, for the MF-AE (blue) and SciPy benchmark (orange). The numerical accuracy for single pulse events $a_{n,1}$ was calculated to be 89.2% for the MF-AE and 46.8% for the SciPy benchmark. The numerical accuracy for double pulse events $a_{n,2}$ was calculated to be 73.6% for the MF-AE and 30.5% for the SciPy benchmark. The values of μ and σ are the mean and standard deviations of the distributions.

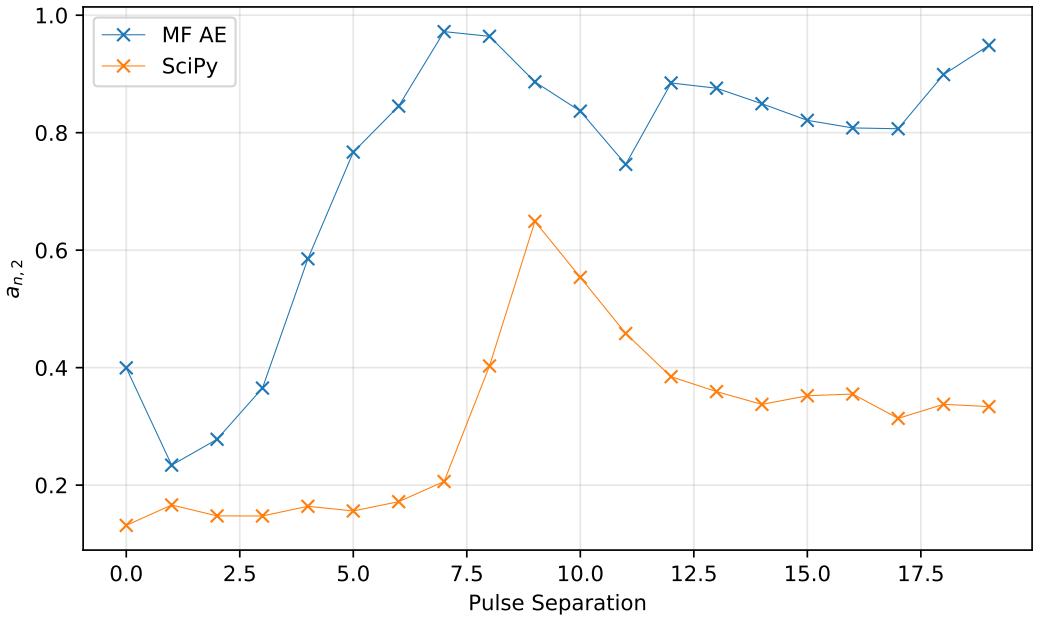


Figure 4.4: Numerical accuracy versus pulse separation (in number of samples) for waveforms containing two pulses for the MF-AE (blue) and the SciPy benchmark (orange).

To understand how well the MF-AE can recover the timings of each pulse in a waveform, histograms containing the distribution of the differences between the true timings T_{true} and predicted timings T_{pred} of pulses were made for both the MF-AE and SciPy benchmark. These can be seen in Figure 4.5. The distributions only include waveforms for which N_{pred} was correctly identified, therefore, in general, more waveforms are included in the distribution for the MF-AE than the SciPy benchmark. The primary interest of this distribution is the spread. The mean or mode, once determined through a small amount of labelled training data, can be used to calibrate the model (in this instance by adding either to T_{pred}). The spread gives an indication of how accurate the model is once this calibration has been performed. The timing accuracy, a_t, is defined as the proportion of the distribution occupied by the modal bin. As pulse timings are discretised on a sample basis, a_t gives the proportion of timings the model correctly predicts after a calibration using the mode is made. The timing error σ_t is defined as the standard deviation of the distribution. Good performance is indicated by a low value of σ_t and a value of a_t close to

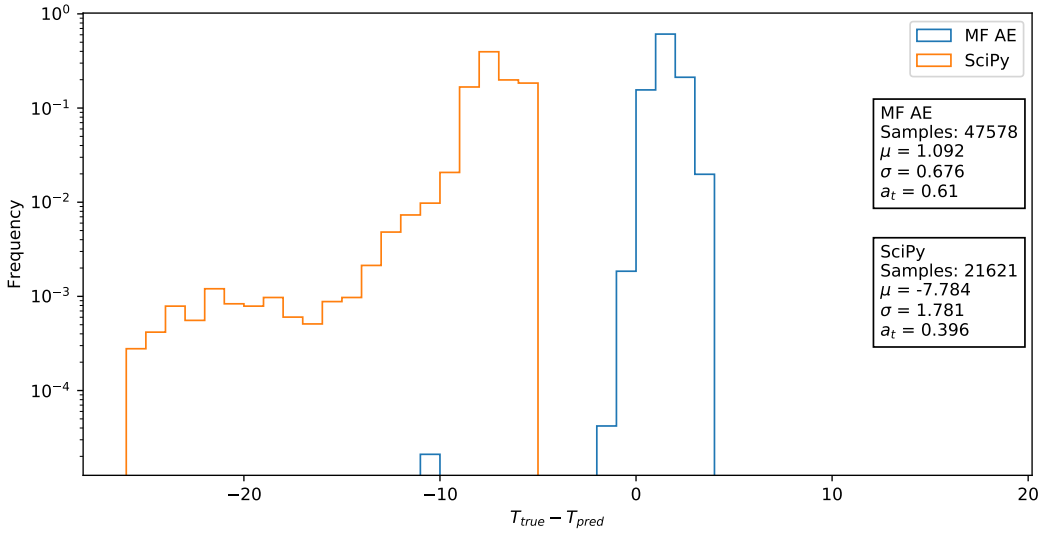


Figure 4.5: Normalised distributions of the differences between the true, T_{true}, and predicted timings, T_{pred}, for each pulse in a waveform, using the MF-AE (blue) and the SciPy benchmark (orange). A perfect model would contain a single bin. Only waveforms for which each method correctly predicted N_{pred} are included. The number of samples is the number of differences calculated. The values of μ and σ are the mean and standard deviations of the distributions. The y-axis is logarithmic. The timing accuracy a_t is found to be 61.0% for the MF-AE and 39.6% for the SciPy benchmark. The timing error σ_t was found to be 0.676 for the MF-AE and 1.781 for the SciPy benchmark.

one.

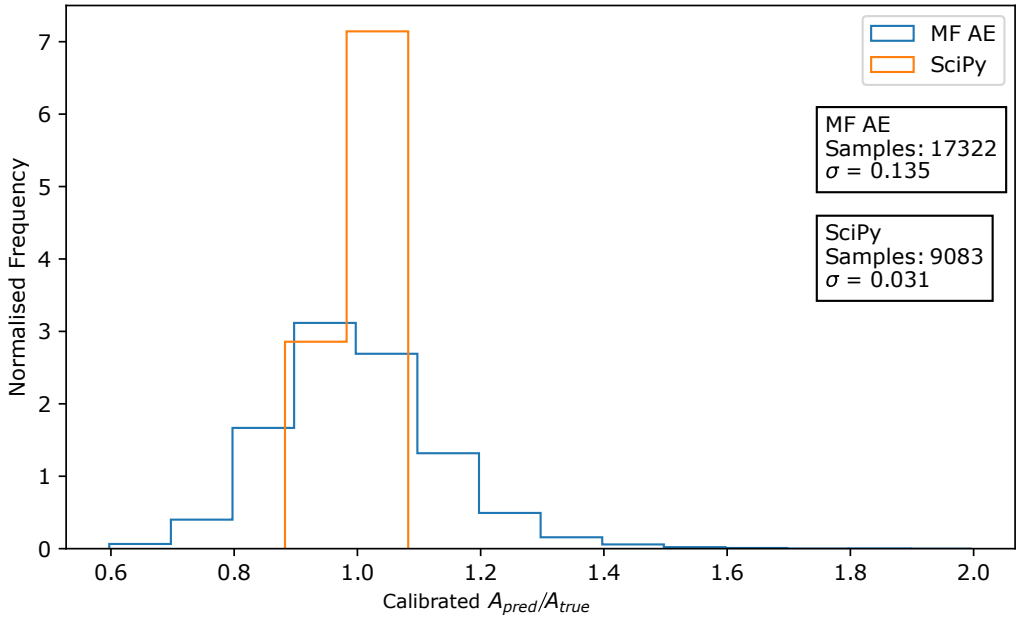
The timing accuracy, a_t, was calculated to be 61.0% for the MF-AE and 39.6% for the SciPy benchmark. The timing error, σ_t, was calculated to be 0.676 for the MF-AE and 1.781 for the SciPy benchmark. The results clearly show that, when the correct number of pulses in a waveform are identified, the MF-AE is far more accurately recovering the timings of those pulses than the simple peak finding approach applied to the raw waveforms. The MF-AE is extracting the majority of pulse timings correctly to within one sample. The tight, single peaked distribution also suggests that the MF-AE is recovering the timings of the second pulse in a double pulse event with a similar accuracy to the first.

The ability of the MF-AE to recover the amplitudes of pulses in waveform was investigated by comparing A_{pred} and A_{true}. Histograms containing the distribution of the ratios between A_{pred} and A_{true} were plotted for the MF-AE and the SciPy benchmark. These can be seen, separated into single and double pulse waveforms, in Figures 4.6a and 4.6b.

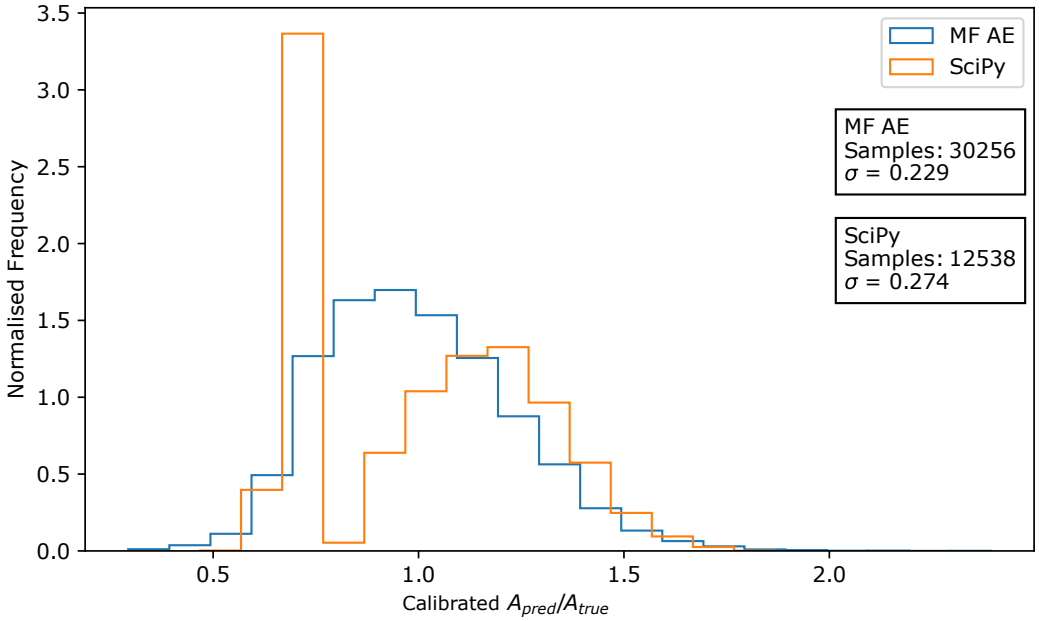
The distributions only include waveforms for which N_{pred} was correct using each method. Once more, it is the spread of this distribution that is of interest. As with the timing analysis, the mean of this distribution can be used to calibrate the model (this calibration has already been performed to produce the distributions in Figure 4.6, such that the two methods can be fairly compared), while its standard deviation, defined as the amplitude error σ_a, gives the error of the predictions once the calibration has been made. A low value of σ_a indicates good performance.

For single pulse waveforms, the amplitude error, $\sigma_{a,1}$, was calculated to be 0.135 for the MF-AE and 0.031 for the SciPy benchmark. In this instance, applying a simple peak finding algorithm to the raw data has performed better than the MF-AE. However, this is unsurprising. For a single pulse waveform, the peak finding algorithm is able to extract the true amplitude of the pulse directly from the raw data to a degree of accuracy given by the noise in the signal, whilst the additional operations of the MF-AE increase the margin for error.

For double pulse waveforms, the SciPy distribution consists of a large narrow peak followed by a wider peak. This might suggest that the simple peak finding algorithm is accurately extracting the amplitude of the first pulse, but overestimating the amplitude of the second, due to it being unable to separate the contributions of the first. As a result of this overestimation, the mean A_{pred}/A_{true} increases and the calibration causes the predictions for the first pulses to appear as underestimates, and the overall amplitude error to be $\sigma_{t,2} = 0.274$. The MF-AE, on the other hand, produces a far more uniform distribution with an amplitude error of $\sigma_{t,2} = 0.229$. This suggests that, when the correct number of pulses is identified, the MF-AE is better separating the contributions of each pulse, leading to more consistent predictions of their amplitudes. However, the error of amplitudes extracted from double pulse waveforms is still significantly higher than those from single pulse waveforms.



(a)



(b)

Figure 4.6: Normalised calibrated distributions of A_{pred}/A_{true} for waveforms containing (a) a single pulse and (b) two pulses, using the MF-AE (blue) and the SciPy benchmark (orange). Only waveforms for which each method correctly predicted N_{pred} are included. The number of samples is the total number of pulses contained in those waveforms. The calibration was performed by dividing by the mean amplitude ratio, to give distributions with a mean of 1. The amplitude error for single pulses events was calculated to be 0.135 and 0.031 for the MF-AE and SciPy benchmark, respectively. The amplitude error for double pulse events was calculated to be 0.229 and 0.274 for the MF-AE and SciPy benchmark, respectively.

In experiments such as LZ, the signal-to-noise ratio of output waveforms are likely to decrease over time, as detecting elements wear out. It is therefore important to test how robust analysis tools, such as the MF-AE, are to changes in noise, once they have already been trained and calibrated. To replicate this scenario, the model used in the following analysis has not been re-trained, therefore the weights remain unchanged, and the optimum height threshold is never re-calculated. The model is tested on data sets consisting of 200 mini-batches of 32 waveforms, in which varying post-convolution noise ϵ has been added. The pre-convolution noise remains constant. For each data set a_n, a_t, σ_t and σ_a are calculated. The same analysis is conducted using the SciPy benchmark.

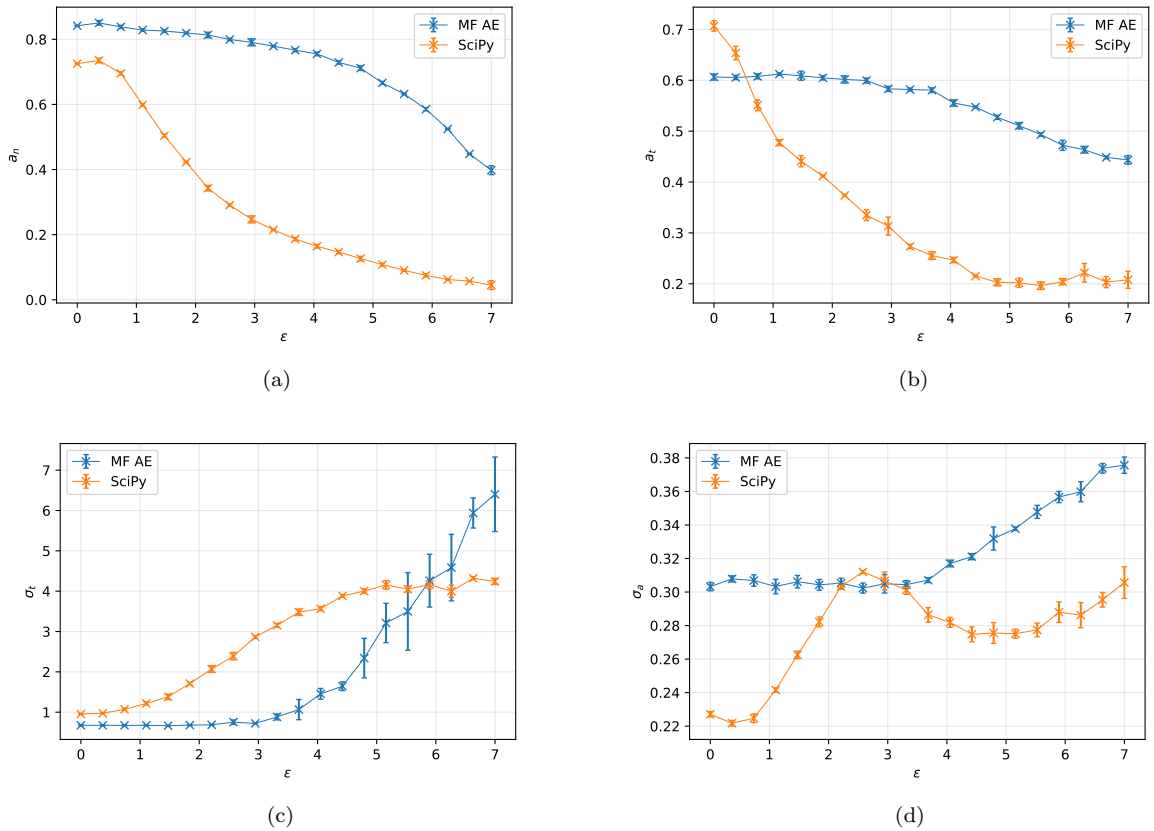


Figure 4.7: Changes in: (a) numerical accuracy, a_n, (b) timing accuracy, a_t, (c) timing error, σ_t and (d) amplitude error, σ_a, with increased post-convolution noise ϵ, for the MF-AE (blue) and SciPy benchmark (orange). Each data point is calculated using validation set consisting of 200 mini-batches of 32 waveforms. Three repeats at each ϵ are conducted; the bars represent the standard deviation across these repeats. Only waveforms for which N_{pred} was correct are used in plots (b), (c) and (d). Note: $\epsilon = 0$ does not indicate that there is no noise is the waveforms, pre-convolution noise is still added to the data (see Appendix B).

Like before, a_t, σ_t and σ_a are calculated using only waveforms for which N_{pred} is correct (meaning, in general, fewer waveforms are used in calculations for the SciPy benchmark). In this instance, a_n and σ_a incorporate both single and double pulses. Three repeats are conducted at each ϵ, and the results averaged. Results for both methods can be seen in Figure 4.7.

Figures 4.7a and 4.7b demonstrate how both a_n and a_t decrease for both methods as ϵ is increased, as one might expect. However, the MF-AE by and large significantly outperforms the benchmark. The MF-AE appears to be more robust to changes in noise for these two measures, as both a_n and a_t decrease at a lower rate, when compared to the benchmark. At $\epsilon = 7$, the MF-AE maintained values of around 40% for both a_n and a_t, while for the benchmark, they had been reduced to around 5% and 20%, respectively.

From Figure 4.7c and 4.7d, both the timing error σ_t and the amplitude error σ_a of the MF-AE are fairly robust to changes in ϵ, up to approximately $\epsilon = 4$, after which they begin to increase. This could be due to the increased noise shifting the apparent timings and amplitudes of peaks in a waveform, making the truth harder to recover. The increase in the error of σ_t with noise is due to there being fewer waveforms for which N_{pred} is correct. Furthermore, although the benchmark appears to outperform the MF-AE for ϵ greater than about 6, the SciPy method has a value of a_n much nearer to zero at this level of noise, while the MF-AE maintains a value around 40-50%, which effectively negates the comparison. The same is true of comparisons of σ_a at high noise; however, at lower noise the benchmark outperforms the MF-AE in this case. This is likely due to the high accuracy when recovering amplitudes from single pulse waveforms using the peak finding algorithm on raw data, as described previously and evident in Figure 4.6a.

4.2 DER Data

4.2.1 Data Processing

As LZ is yet to begin taking data, the functionality of the MF-AE was instead further explored using simulated LZ data. The DER, which forms a part of the wider simulation

framework used by LZ, replicates the PMT response to photon hits, and the ensuing signal processing of the read-out electronics, to produce waveforms formatted to match the intended output of the LZ data acquisition system [35]. The PMT response model uses a quantum efficiency parameter, that determines whether or not a photon interacting with the detector produces a signal, and applies Gaussianly-distributed gain and time delay to the resulting signals [35].

The waveforms used here simulate the response of a single low-gain channel located in the top PMT array, to single and double photon S1 signals. For single pulse events, photons fed into the DER had a Gaussian time profile with mean 20ns and variance 30ns. For double pulse events, the initial photon had a Gaussian time profile with mean 40ns and variance 30ns while the second photon had a time uniformly offset between 0 and 40ns with respect to the first. Here, 10ns equates to one sample, therefore approximately one fifth of double pulse events contain photons which would have been fed into the DER at the same time. The photons had a fixed wavelength of 275nm. The waveforms produced by the DER were translated to near zero, by subtracting their maximum, and padded to the nearest ten samples, then grouped into mini-batches of 32 waveforms containing the same number of samples.

4.2.2 DER Data Results

The model in this section used $m^A = m^U = 60$ and $m^{PF} = 12$. The hyperparameters $\lambda_{L1} = 0.6$ and $\lambda_{L2} = 15$ were used, which were found using Bayesian optimisation. The MF-AE was trained using the ADAM optimiser with 4754 mini-batches of 32 DER waveforms for 5 epochs. The weights $\mathbf{w}^A$, $\mathbf{w}^{PF}$ and $\mathbf{w}^U$ were initialised using Glorot initialisation. The performance of the model was validated using 1358 mini-batches, to produce the results presented here. Predictions were extracted from the encoded representation using the same SciPy peak finding function as described in Section 4.1.2. Both the training and validation sets contain data of variable lengths and an approximately even number of single and double pulse waveforms.

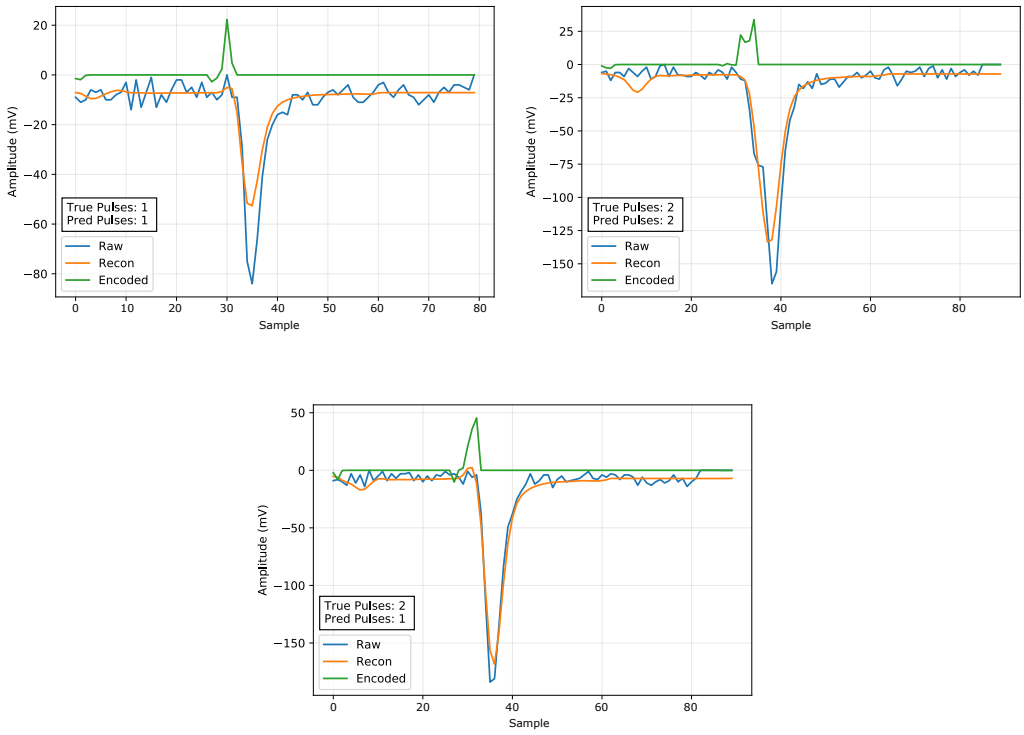


Figure 4.8: Examples of raw (blue), encoded (green) and reconstructed (orange) waveforms, using the DER data. Each sample is a 10ns step. As with Figure 4.1, the encoded representation is of the desired form: sharp peaks where pulses are predicted to have occurred, but otherwise sparse. The true and predicted number of pulses contained in the waveforms are given. The prediction of the MF-AE is correct for the top two plots, but incorrect for the bottom.

Figure 4.8 contains examples of the raw, encoded and reconstructed waveforms after applying the MF-AE to the DER data. As with the toy data, the encoded space is of the intended form, with one example demonstrating the ability of the MF-AE to separate two closely overlapping pulses in a waveform. Figure 4.9 shows the learnt weights of the three convolutional kernels for this model, which exhibit similar features to those observed in Figure 4.9. The 'Amplitude' layer has once again learnt a smooth pulse. The 'Pulse Finder' filter has steep characteristics and contains features that resemble the sharp rising edge of a pulse, however it is not strictly increasing as a rising edge filter would be. The interpretation of the 'Unfold' layer is once again less obvious. Both Figures 4.8 and 4.9 provide further evidence that, to a significant degree, the MF-AE is capable of learning what is intended of it without explicitly being guided in what to learn. It is also promising to see that the MF-AE is achieving this in a manner that allows it to adapt to different

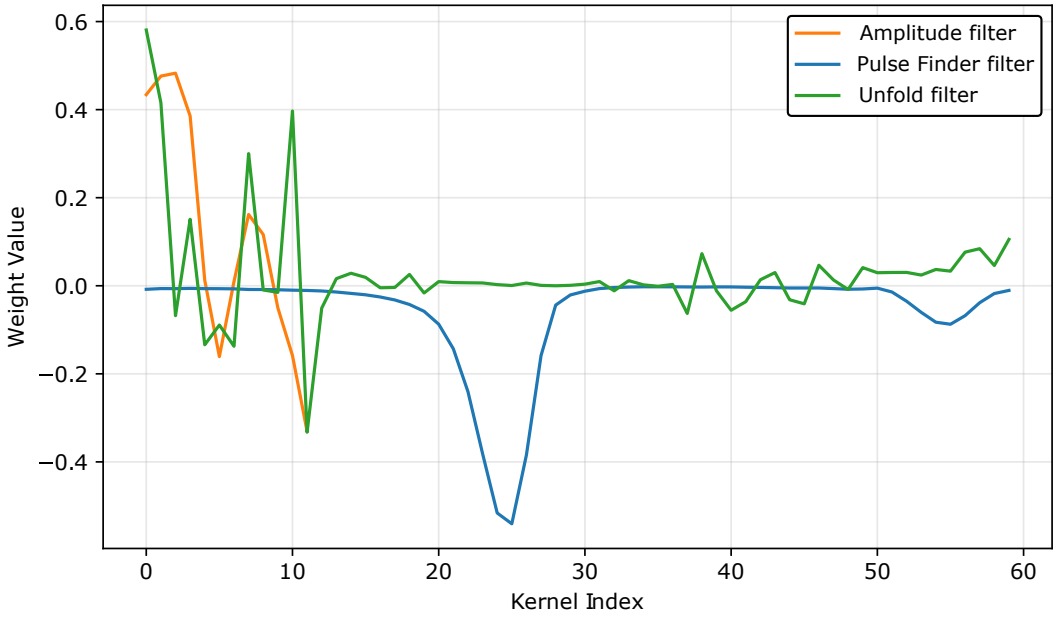
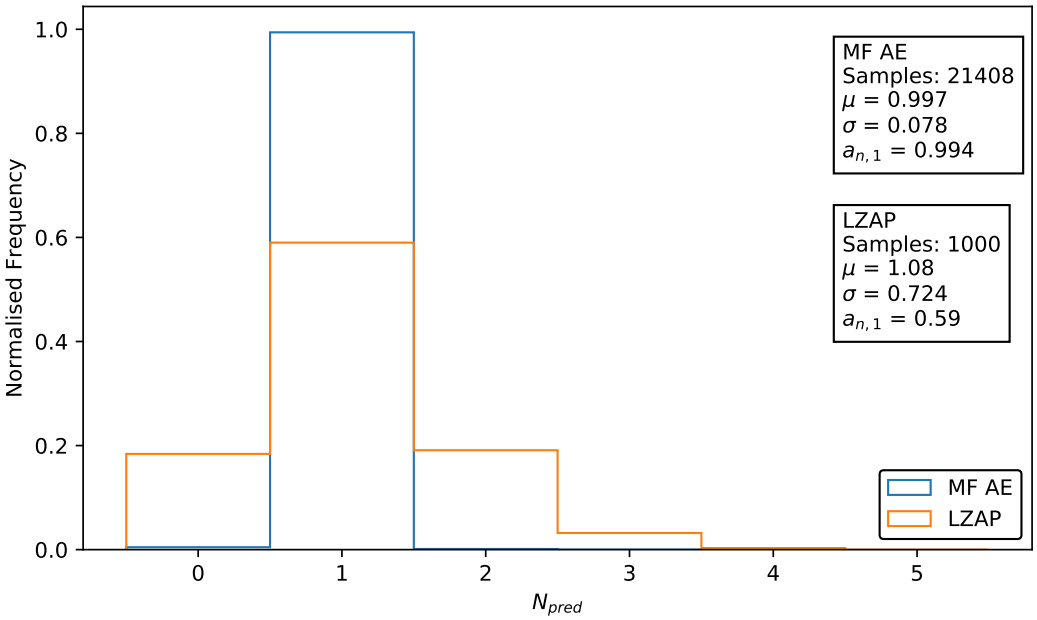


Figure 4.9: The learnt weights for the 'Amplitude' $\mathbf{w}^A$ (blue), 'Pulse Finder' $\mathbf{w}^{PF}$ (orange) and 'Unfold' $\mathbf{w}^U$ (green) filters, for the model trained using DER data.

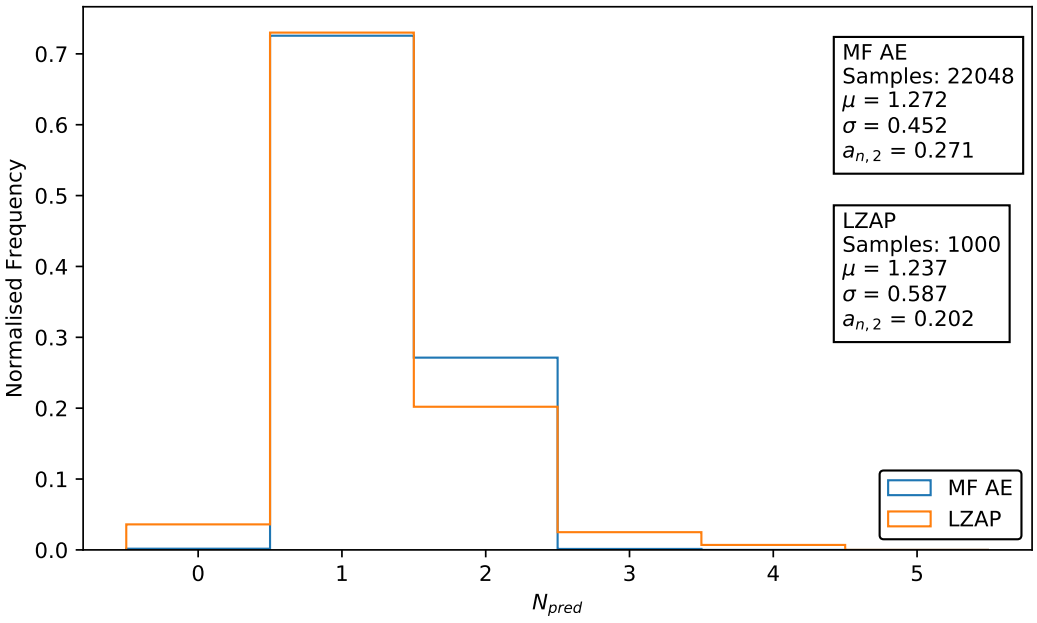
data sets.

To contextualise the performance of the MF-AE when applied to the DER data, predictions of the number of pulses, N_{pred}, contained in DER waveforms were made using both the MF-AE and the current LZ analysis package (LZAP). LZAP has a broad range of functions including pulse finding, event and interaction classifcation and position reconstruction. Here, the main interest is pulse finding. LZAP uses the Difference of Gaussians (DoG) method to detects pulses in a waveform [36]. The DoG method involves convolving two Gaussian filters with the waveform and subtracting the results. The filtered output preserves the frequencies of the signal in a range specified by the widths of the Gaussian filters; these are chosen to match the frequency characteristics of S1 and S2 signals. Pulse features are identified using the local extrema of the filtered output.

Figures 4.10a and 4.10b show the distribution of N_{pred} using the two methods for single and double pulse waveforms. The numerical accuracy for single pulse waveforms $\mathrm{a}_{n,1}$ was calculated to be 99.4% and 59.0% using the MF-AE and LZAP, repectively. The numerical accuracy for double pulse waveforms $\mathrm{a}_{n,2}$ was calculated to be 27.1% and 20.2% using the MF-AE and LZAP, respectively.



(a)



(b)

Figure 4.10: Normalised distributions of N_{pred} for: (a) waveforms containing one underlying pulse and (b) waveforms containing two underlying pulses, for the MF-AE (blue) and LZAP (orange). The numerical accuracy for single pulse events $a_{n,1}$ was calculated to be 99.4% for the MF-AE and 59.0% for LZAP. The numerical accuracy for double pulse events $a_{n,2}$ was calculated to be 27.1% for the MF-AE and 20.2% for LZAP. The values of μ and σ are the mean and standard deviations of the distributions. Only 1000 waveforms of each type were used to test LZAP, compared to the $\sim$ 20000 for the MF-AE.

The MF-AE has therefore produced a substantial improvement in accuracy for single pulse events, having made a minute fraction of false predictions, and a more modest improvement for double pulse events. The lower accuracy for double pulse events using DER data when compared to the toy data can be attributed to the former containing waveforms with less separated pulses, resulting in more false predictions (see Figure 4.4). The overall numerical accuracy was 62.7% for the MF-AE and 39.6% for LZAP. It should be noted that these results by no means offer a comprehensive comparison between MF-AE and LZAP. However, they certainly demonstrate the potential of the MF-AE for use in waveform analysis at LZ, motivating further investigations beyond this initial project.

5 Discussion and Future Work

The work presented here instigates good initial progress in developing a DL model for waveform analysis at LZ. By designing its architecture to approximate the MFT, the MF-AE becomes a tool for waveform analysis that effectively leverages the benefits of DL, while remaining interpretable and highly constrained. The MF-AE has $\mathcal{O}(100)$ trainable parameters, far fewer than regular CNNs. The MF-AE can produce the intended results, without explicitly enforcing how the weights should be learnt. The ability to learn the filters that optimise results for a given set of waveforms, makes it a more robust method than the traditional MFT, where the template needed to be known in advance. This, coupled with the fact that the model accepts as its input waveforms containing any number of samples, makes the MF-AE highly applicable to real-life scenarios; although, a calibration using a small labelled training set would be required.

Results presented in Section 4.1.2 demonstrate the effectiveness of the MF-AE for extracting features from waveforms, particularly the number of pulses they contain and their timings. Improvements could be made to the resolution of the recovered amplitudes, however. Promising results were observed in Section 4.2.2, in which the performance of the MF-AE in identifying pulses compared favourably to that of the current pulse finding method used in LZAP, when using data simulated by the LZ DER. It would have been

helpful to have also extracted the pulse timing and amplitude information from the DER waveforms, as with the toy data, and compared these predictions with those made by LZAP. This would have provided a more grounded source for comparison between the MF-AE and LZAP. However, the relevant truth information was not available on this occasion. This would be interesting to see, should it become available in the future.

Interpretabilty would benefit from further understanding the learnt weights $\mathbf{w}^U$. It is hard to know how the optimal solution for a weight kernel that emulates the matrix inversion step in the MFT should look, especially without a corresponding filter from the MFT with which to compare, and it is possible that it may not have an easily interpretable structure. It is still fair to say, however, that the interpretability of the model is high, despite the current ambiguity of $\mathbf{w}^U$.

One area of future work should focus on improving the stability and repeatability of the model. A total of 100 models were trained using the identical parameters to those given in 4.1.2, each with a different set of training and validation data. Of those, 61 produced flat encoded representations. When applied to their respective validation sets, the average numerical accuracy, and its standard deviation, of the remaining 39 models was calculated to be $\mathrm{a}_n = 0.5 \pm 0.2$. Therefore, although ingrained in the model architecture presented here is the ability to perform feature extractions tasks well, this clearly depends on the appropriate weights being learnt. A large proportion of the trained models produced encoded representations of no value, while for those that did, the performance varied significantly. In theory, one could keep training models until a particular weight configuration performs as expected, and then not worry about further training, however this is decidedly impractical. The large number of flat encoded representations could be due to the L1 regularisation that is applied to the encoded layer, and its effect on the loss. It is possible that, during training, reducing the contribution of the L1 term could sometimes be more beneficial in minimising the loss than producing accurate reconstructions, in which case a flat encoded representation could be learnt in favour of one containing information about the waveform. Stability may therefore improve by adjusting the hyperparameter that governs this regularisation. Adjustments to the model architecture and learning process could also be

explored. The high variation in the numerical accuracy of the models could be due to the fact that the model is not being guided in what to learn. It is therefore conceivable that there are certain configurations of the encoded space that, while able to effectively minimise the loss by producing accurate reconstructions, are not of the desired delta-function like form and therefore not well suited for feature extraction. If this were the case, it is less clear how this could be resolved, as the unsupervised nature of the learning process is one of the fundamental aspects of the design of the model.

Future work should also focus on developing and testing the model using data that better approximates the more varied types of event LZ could be detecting. This work tested the MF-AE using waveforms of one only shape and signal type (S1) at a time. It would be informative to test its performance using data sets containing different shaped waveforms, and multiple types of signals (such as including S2 signals). A model that could effectively discriminate multiple types of signals and pulse shapes is vital for minimising background and would be important when conducting multi-channel analysis. Different PMTs are expected to detect different types of signal more often than others dependent on, for example, their location within the detector, and are likely to have varying noise profiles. If every PMT channel had attached to it a model responsible for processing its signal responses, then this model should be able to accommodate this expected variation in responses. Some work was done producing a multi-layer AE model, with the intention of having each layer identify a different pulse shape, although this was not presented here due it being in the primitive stages of development. Producing a deeper model in this way, with a larger number of trainable parameters, would likely require a trade off between the performance and interpretability. Future work could continue with this idea.

6 Conclusion

In this project, a new architecture for a fully convolutional neural network for waveform analysis at LZ has been proposed: the MF-AE. Inspired by the MFT, the model transforms its inputs in a way that is highly interpretable and possesses several other attractive

qualities, including: having no constraint on the length of its input, having relatively few trainable parameters and the ability to learn useful features of data without supervision. The model was trained and tested using toy waveforms that simulated the response of an LZ PMT channel to an S1 signal and was found to accurately predict the number of pulses in a waveform and recover the timings of the pulses. The model is currently less effective at recovering pulse amplitudes. Preliminary comparisons of the MF-AE with LZAP proved promising, with this former correctly predicting the number of pulses in a higher proportion of simulated LZ DER waveforms than the latter. The work presented here demonstrates the potential of the MF-AE for use in waveform analysis at LZ and has opened up several avenues for future work that could seek to develop and improve it.

A Minimising the RSS with the Matched Filter

Consider modelling a waveform $\mathbf{S}$ using a template offset by i and scaled by μ_i. The RSS between the waveform and a template shifted and scaled as such is

$$\mathrm{RSS}_i = \sum_j (S_j - \mu_i T_{j-i})^2. \tag{17}$$

The scaling μ_i that minimises the RSS for a template with this offset can be found by differentiating with respect to μ_i and equating with zero:

$$\frac{d\mathrm{RSS}_i}{d\mu_i} = -\sum_j T_{j-i}(S_j - \mu_i T_{j-i}) = 0, \tag{18}$$

which gives

$$\mu_i = \frac{\sum_j S_j T_{j-i}}{|\mathbf{T}|^2}. \tag{19}$$

Therefore, by comparing equations 19 and 2, it is clear that the elements of the filtered output F_i are the scaling parameters that minimise the RSS between the waveform and a template offset by i. The scaling parameter μ_i (or similarly F_i) is then equal the optimum

value a when the template has been offset by the optimal shift (i.e. the argument of the maximum value of F_i, as seen previously).

B Toy Data Algorithm

Algorithm 1 shows pseudocode for the algorithm used to generate single pulses waveforms that formed part of the toy data set.

Algorithm 1: Generate single pulse waveform

Initialise $\mathbf{S}$, an array of zeros length 300
Initialise $\mathbf{T}$, template function length 75
i $\leftarrow$ Sample $\mathcal{N}(\frac{3}{7} \times 300, 15)$ `// Position pulse on average` $\frac{3}{7}$ `along sparse array`
a $\leftarrow$ Sample $\mathcal{N}(400, 50)$ `// Generate amplitude scaling`
$\mathbf{S}_i \leftarrow$ a
for $i = 1, 2, ..., 300$ **do**
    $\delta \leftarrow \mathcal{N}(0, 3)$
    $\mathbf{S}_i \leftarrow \mathbf{S}_i + \delta$ `// Add pre conv noise`
$\mathbf{X} \leftarrow \mathbf{S} * \mathbf{T}$ `// 'Valid' conv, X shorter than S by one less than length T`
for $i = 1, 2, ..., 226$ **do**
    $\delta \leftarrow \mathcal{N}(0, \epsilon)$
    $\mathbf{X}_i \leftarrow \mathbf{X}_i + \delta$ `// Add post conv noise`
t $\leftarrow$ i - (75 - 1) `// Pulse timing shifted back by one less than length T after conv`
Save $\mathbf{X}$, t, a `// Waveform, pulse timing, pulse amplitude scaling`

A second pulse can be added to the waveform generated using Algorithm 1 by generating a second amplitude scaling and positioning it in the sparse matrix by an amount offset uniformly from the first, before the convolution. In this work, the offset used was between 0 and 20 samples. The template $\mathbf{T}$ used in this work had a maximum value of ~ 0.34

meaning the pulses had an average amplitude of $0.34 * 400 \approx 135$. Using a template whose maximum value is 1 would result in pulses whose mean amplitude is equal to the mean amplitude scaling. Using a 'valid' convolution for which the dot product between **T** and **S** is only calculated when **T** is overlapping **S** in its entirety (without padding) resulted in waveforms containing fewer samples than the initial sparse matrix by one less than the length of **T**. This also shifts the timings of the pulses backwards by the same number of samples. The sparse matrix used here contained 300 samples, resulting in waveforms containing 226 samples.

References

[1] G. Bertone, D. Hooper, and J. Silk. "Particle dark matter: Evidence, candidates and constraints". *Phys. Rept.* **405** (2005), 279–390. DOI: 10.1016/j.physrep.2004.08.031. arXiv: hep-ph/0404175.

[2] V. C. Rubin, N. Thonnard, and W. K. Ford Jr. "Extended rotation curves of high-luminosity spiral galaxies. IV - Systematic dynamical properties, SA through SC". *Astrophys. J.* **225** (1978), L107. DOI: 10.1086/182804.

[3] M. Persic, P. Salucci, and F. Stel. "The universal rotation curve of spiral galaxies — I. The dark matter connection". *Mon. Not. R. Astron. Soc.* **281.1** (1996), 27–47. DOI: 10.1093/mnras/278.1.27.

[4] M. Schumann. "Direct detection of WIMP dark matter: concepts and status". *J. Phys. G* **46.10** (2019), 103003. DOI: 10.1088/1361-6471/ab2ea5.

[5] J. Liu, X. Chen, and X. Ji. "Current status of direct dark matter detection experiments". *Nat. Phys.* **13.3** (2017), 212–216. DOI: 10.1038/nphys4039.

[6] D.S. Akerib et al. "The LUX-ZEPLIN (LZ) experiment". *Nucl. Instrum. Methods Phys. Res. A.* **953** (Feb. 2020), 163047. DOI: 10.1016/j.nima.2019.163047.

[7] E. H. Miller et al. "Constraining radon backgrounds in LZ" (2018). DOI: 10.1063/1.5018996.

[8] D. S. Akerib et al. "Projected WIMP sensitivity of the LUX-ZEPLIN dark matter experiment". *Phys. Rev. D.* **101.5** (2020). DOI: 10.1103/physrevd.101.052002.

[9] Y. Bengio, A. Courville, and P. Vincent. "Representation Learning: A Review and New Perspectives". *IEEE Trans. Pattern Anal. Mach. Intell* **35.8** (2013), 1798–1828.

[10] D. George and E.A. Huerta. "Deep Learning for real-time gravitational wave detection and parameter estimation: Results with Advanced LIGO data". *Phys. Lett. B.* **778** (2018), 64–70. DOI: 10.1016/j.physletb.2017.12.053.

[11] P. Holl et al. "Deep learning based pulse shape discrimination for germanium detectors". *Eur. Phys. J. C* **79.6** (2019). DOI: 10.1140/epjc/s10052-019-6869-2.

[12] J. L. Flores et al. "Application of neural networks to digital pulse shape analysis for an array of silicon strip detectors". *Nucl. Instrum. Methods Phys. Res. A.* **830** (2016), 287–293. DOI: 10.1016/j.nima.2016.05.107.

[13] B.J. Mount et al. "LUX-ZEPLIN (LZ) Technical Design Report" (2017). eprint: arXiv:1703.09144.

[14] D. S. Akerib et al. "Results from a Search for Dark Matter in the Complete LUX Exposure". *Phys. Rev. Lett* **118.2** (2017). DOI: 10.1103/physrevlett.118.021303.

[15] E. Aprile et al. "Dark Matter Search Results from a One Ton-Year Exposure of XENON1T". *Phys. Rev. Lett* **121.11** (2018). DOI: 10.1103/physrevlett.121.111302.

[16] J. Yin. "The dynamic range of LZ". *J. Instrum* **11.02** (2016), C02054–C02054. DOI: 10.1088/1748-0221/11/02/c02054.

[17] A. Bailey. "Dark Matter Searches and Study of Electrode Design in LUX and LZ". PhD Thesis (2016), Imperial College London.

[18] J.D. Lewin and P.F. Smith. "Review of mathematics, numerical factors, and corrections for dark matter experiments based on elastic nuclear recoil". *Astropart. Phys* **6.1** (1996), 87–112. DOI: 10.1016/s0927-6505(96)00047-3.

[19] G. Turin. "An introduction to matched filters". *IEEE Trans. Inf. Theory* **6.3** (1960), 311–329.

[20] B. Krikler. Private communication (2019).

[21] D. Guest, K. Cranmer, and D. Whiteson. "Deep Learning and Its Application to LHC Physics". *Annu. Rev. Nucl. Part. S.* **68.1** (Oct. 2018), 161–181. DOI: 10.1146/annurev-nucl-101917-021019.

[22] A. Radovic et al. "Machine learning at the energy and intensity frontiers of particle physics". *Nature* **560.7716** (2018), 41–48. DOI: 10.1038/s41586-018-0361-2.

[23] I. Goodfellow, Y. Bengio, and A. Courville. "Deep Learning" (2016). http://www.deeplearningbook.org.

[24] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. "Learning representations by back-propagating errors". *Nature* **323.6088** (1986), 533–536. DOI: 10.1038/323533a0.

[25] D. Kingma and J. Ba. "Adam: A Method for Stochastic Optimization". *ICLR* (2014).

[26] C. Madrazo et al. "Application of a Convolutional Neural Network for image classification to the analysis of collisions in High Energy Physics". *EPJ Web Conf.* **214** (2017). DOI: 10.1051/epjconf/201921406017.

[27] Y. LeCun, Y. Bengio, and G. Hinton. "Deep learning". *Nature* **521.7553** (2015), 436–444. DOI: 10.1038/nature14539.

[28] M. Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. (2015).

[29] F. Chollet et al. *Keras*. https://keras.io. (2015).

[30] A. L. Maas. "Rectifier Nonlinearities Improve Neural Network Acoustic Models" (2013).

[31] J. Snoek, H. Larochelle, and R. P. Adams. "Practical Bayesian Optimization of Machine Learning Algorithms". *NIPS'12: Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 2* (2012), 2951–2959.

[32] F. Nogueira. *Bayesian Optimization: Open source constrained global optimization tool for Python*. `https://github.com/fmfn/BayesianOptimization`. 2014–.

[33] X. Glorot and Y. Bengio. "Understanding the difficulty of training deep feedforward neural networks". *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics* **9** (2010), 249–256.

[34] P. Virtanen et al. "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python". *Nat. Methods* **17** (2020), 261–272. DOI: `https://doi.org/10.1038/s41592-019-0686-2`.

[35] The LUX-ZEPLIN Collaboration et al. "Simulations of Events for the LUX-ZEPLIN (LZ) Dark Matter Experiment" (2020). eprint: `arXiv:2001.09363`.

[36] Paper under preparation.