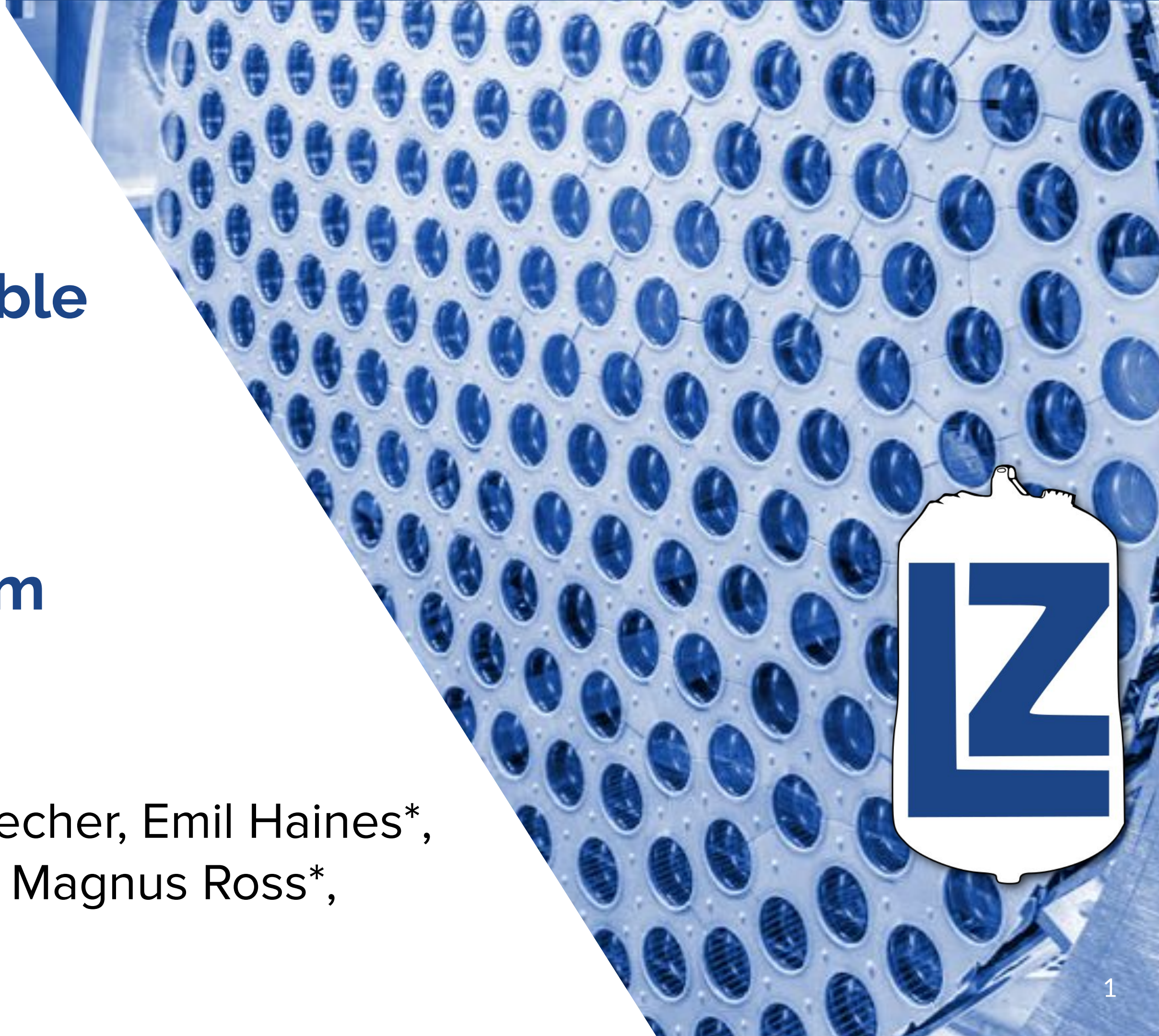
A readily-interpretable fully-convolutional autoencoder-like algorithm for unlabelled waveform analysis

23rd October 2020

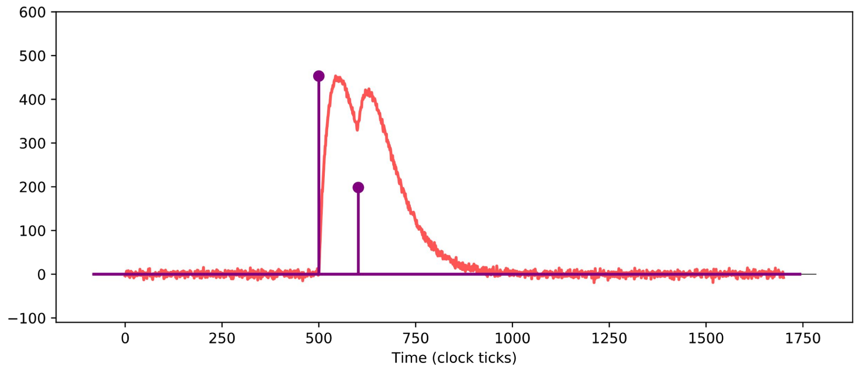
Sam Eriksen, Henning Flaecher, Emil Haines*, Ben Krikler, Jo Orpwood*, Magnus Ross*, Chris Wright



Waveform analysis

Goals of puse reconstruction:

- What energy, when, what type?



Methods:

- Simple sum or maxima ⇒ Poor resolution
- Filter noise: ⇒ better but sub-optimal resolution, and parameters to tune
- Template fitting ⇒ optimal resolution but need a template

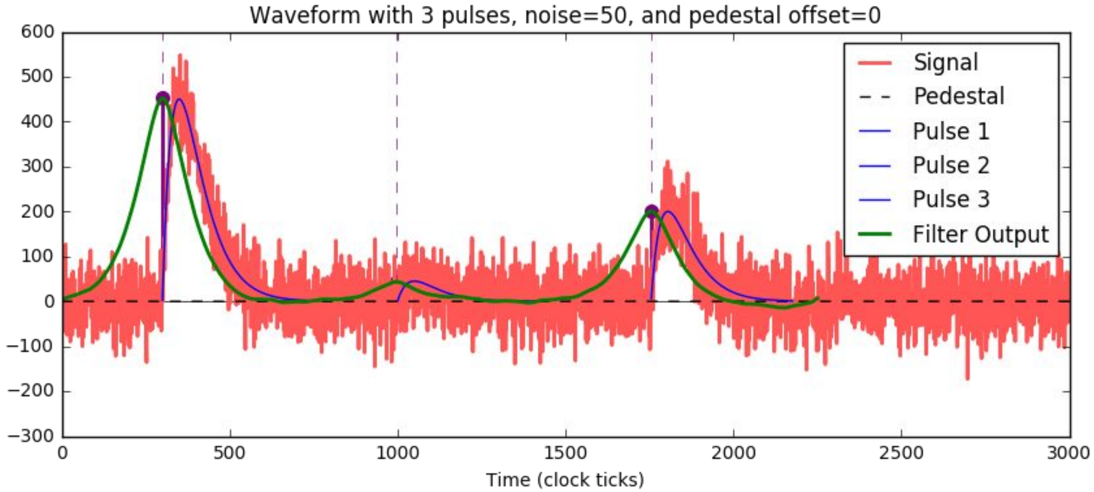
Learn the optimal template without labelling the data ?

The matched filter

If you know the expected pulse shape:

- Convolution ⇒ Best possible signal to noise ratio for any linear filter
- Maxima indicate:
    - When pulse happens
    - What energy it has

Convolutional layers in CNNs are matched filters

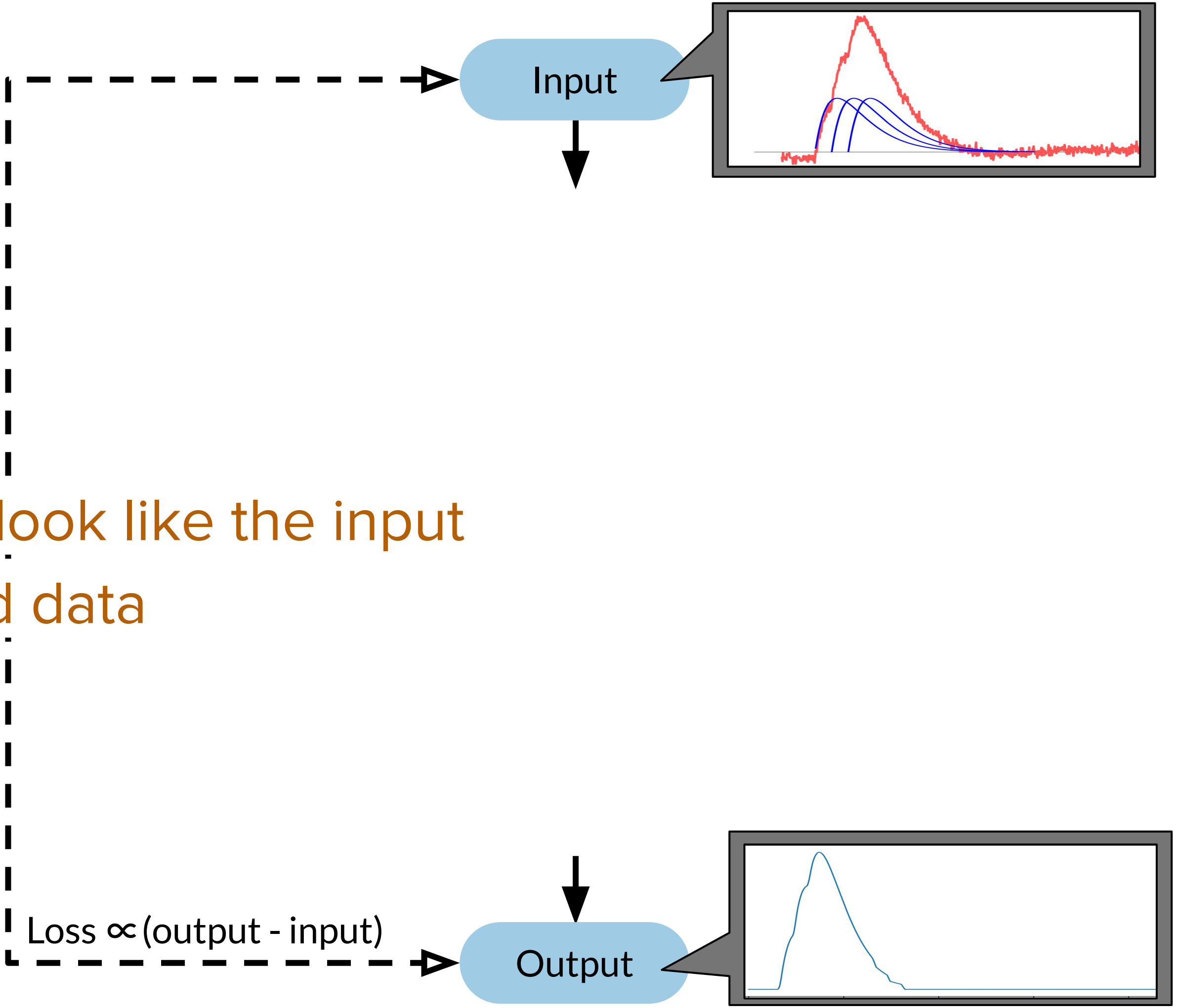


Our model

Autoencoder-like:

⇒ want the output to look like the input

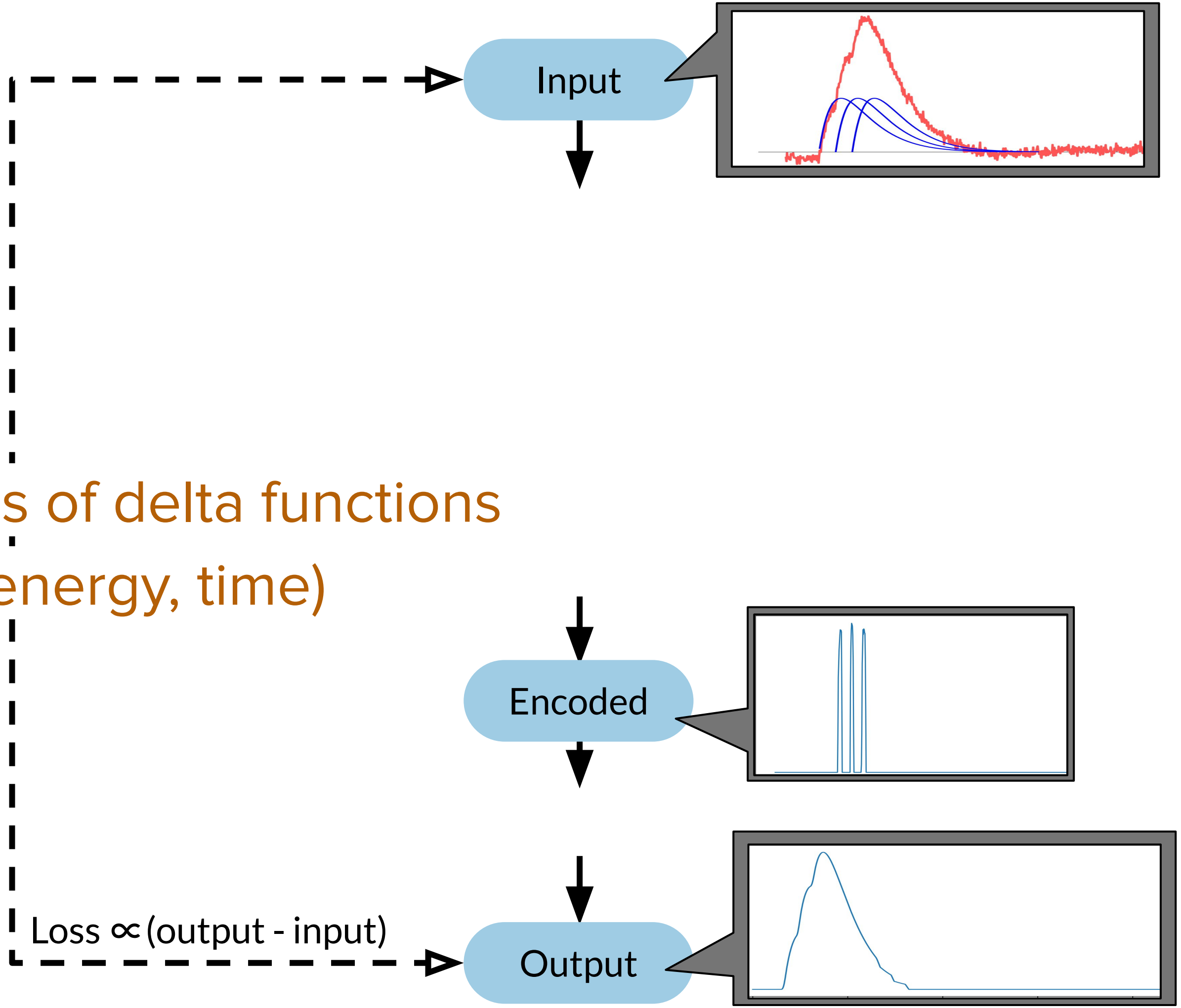
⇒ works on unlabeled data



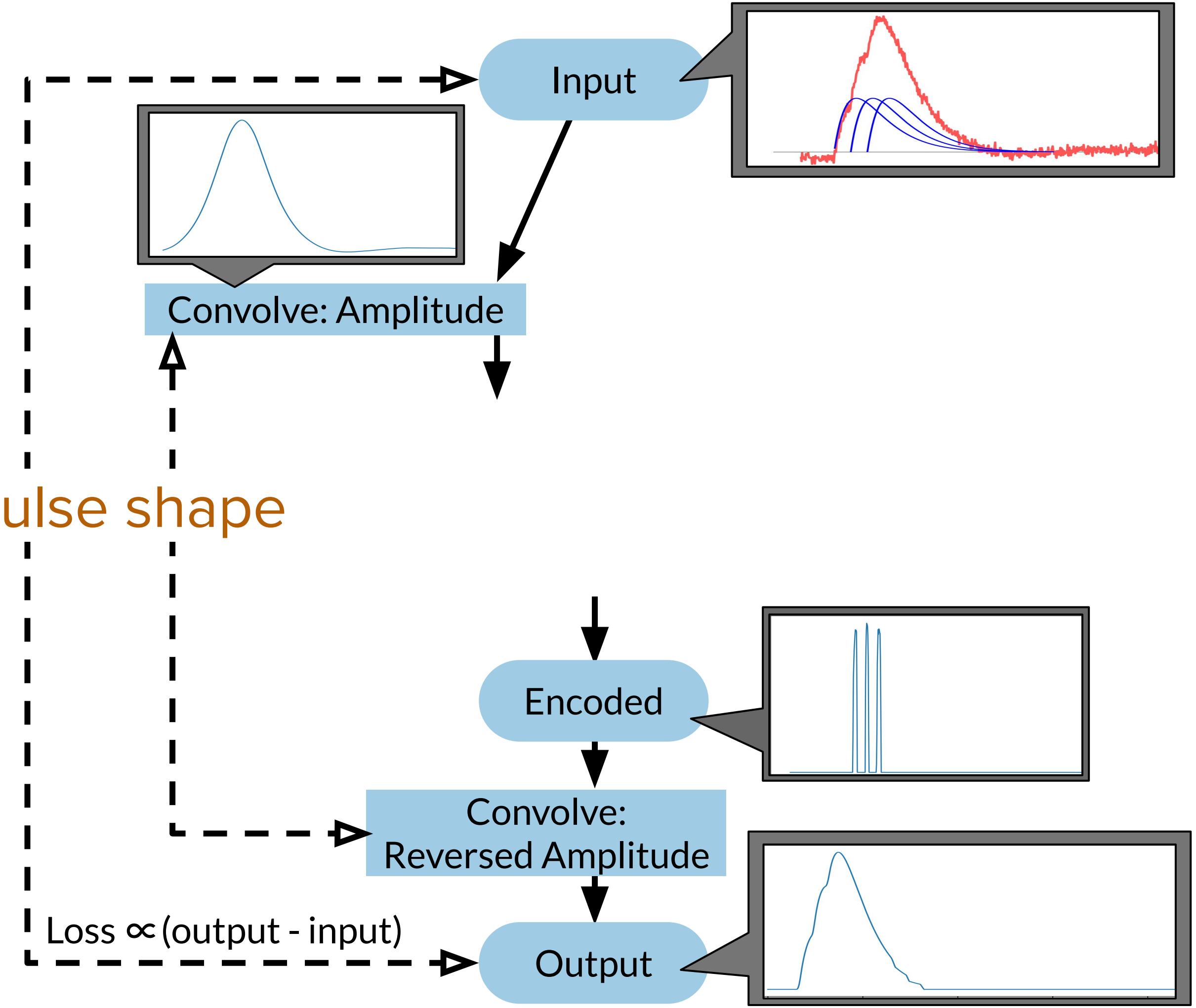
Our model

Encoded space: series of delta functions

⇒ Pulse parameters (energy, time)



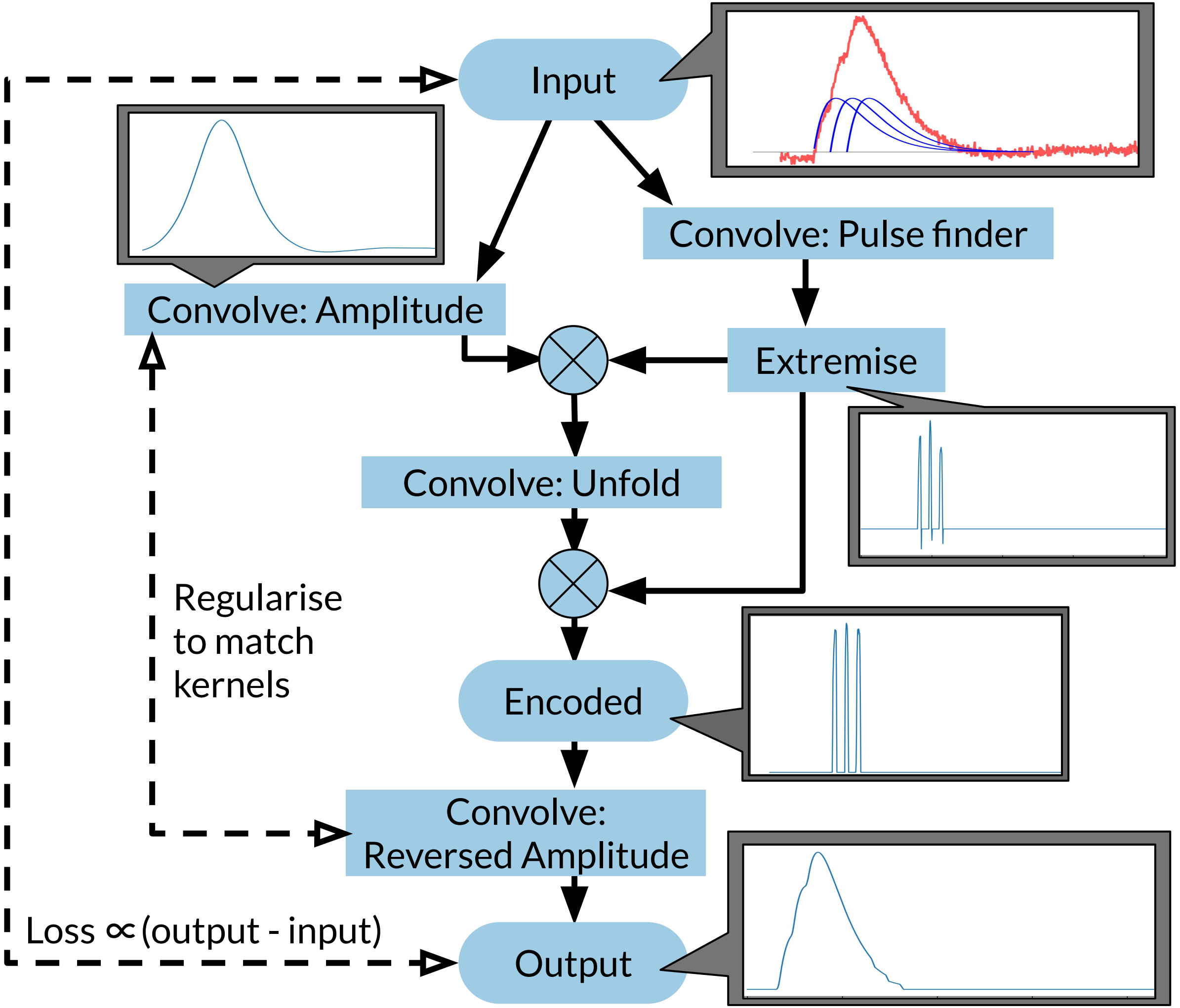
Our model



Convolve with learnt pulse shape

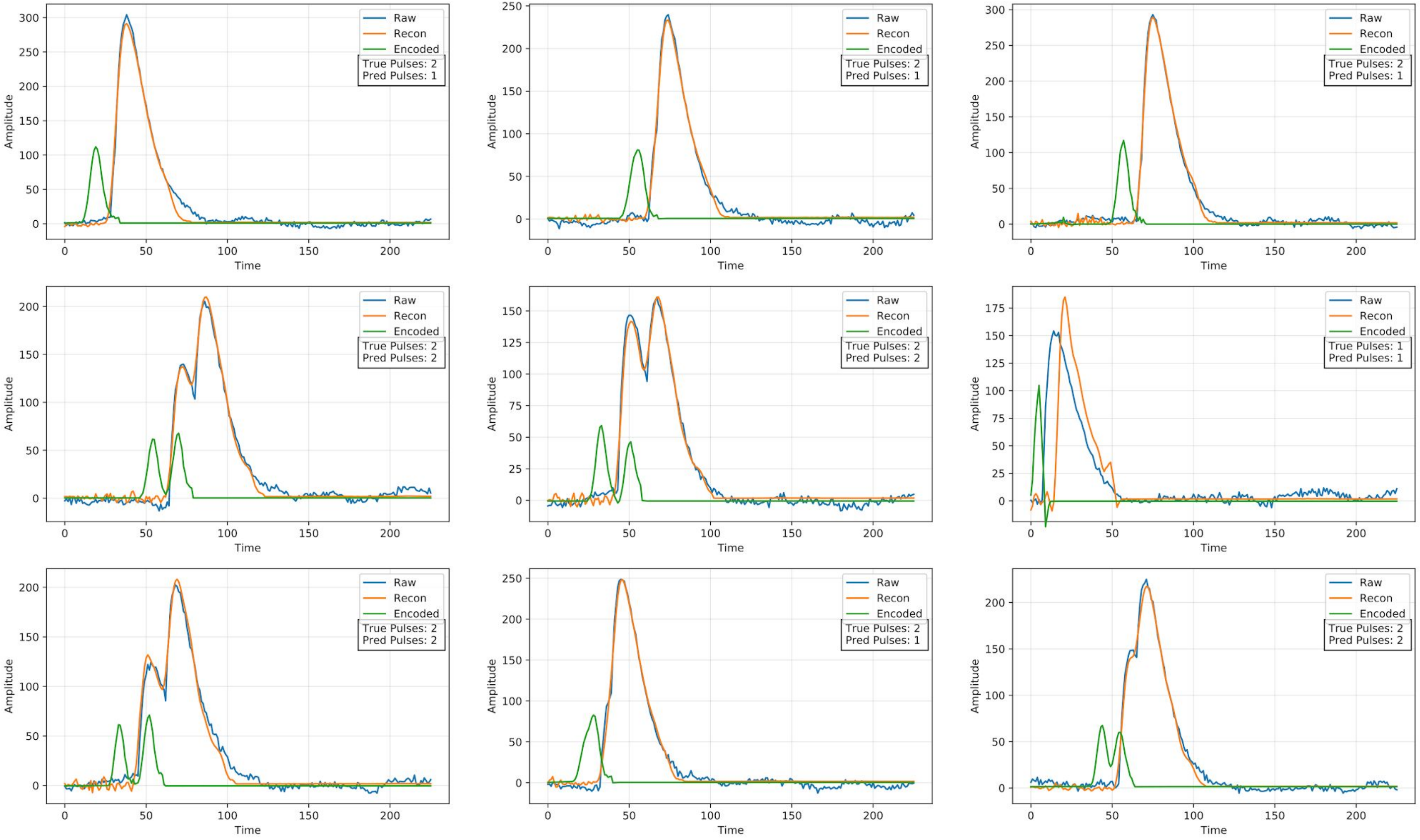
Our model

Detect pulses and disentangle contributions to observed amplitudes

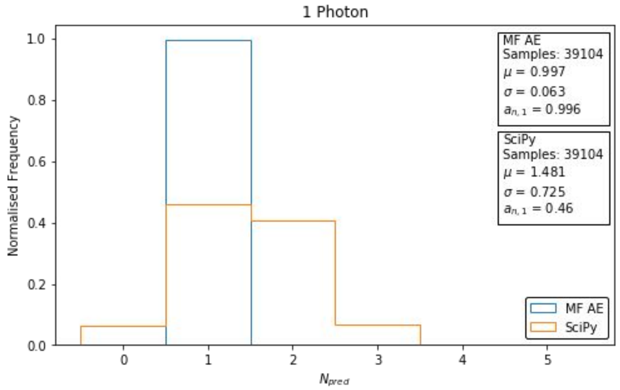


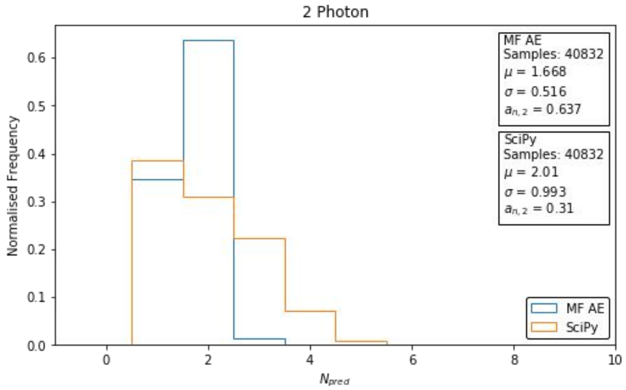
Does it work?

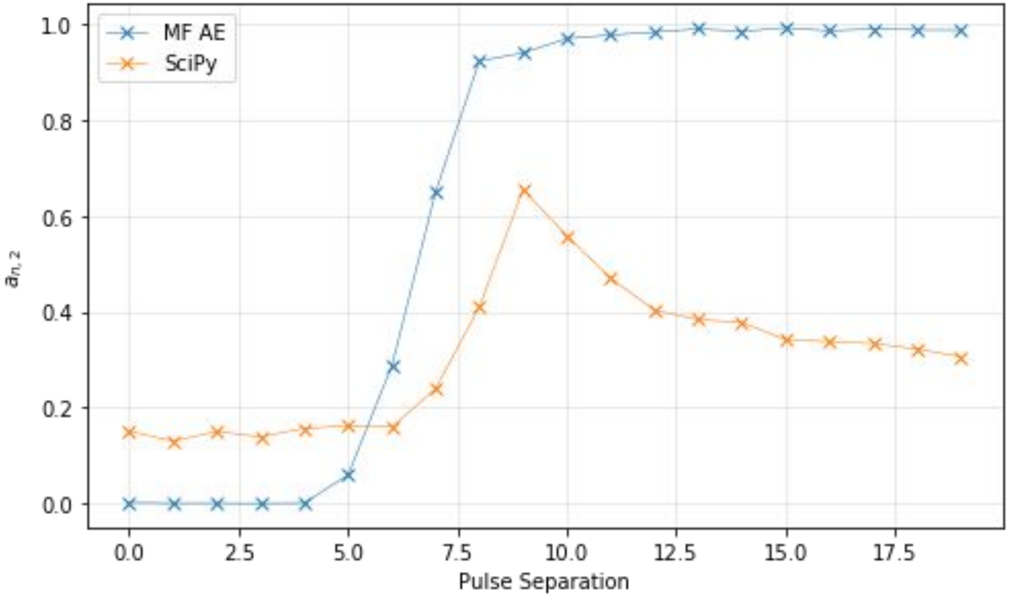
Examples on toy data



Handling overlapping pulses



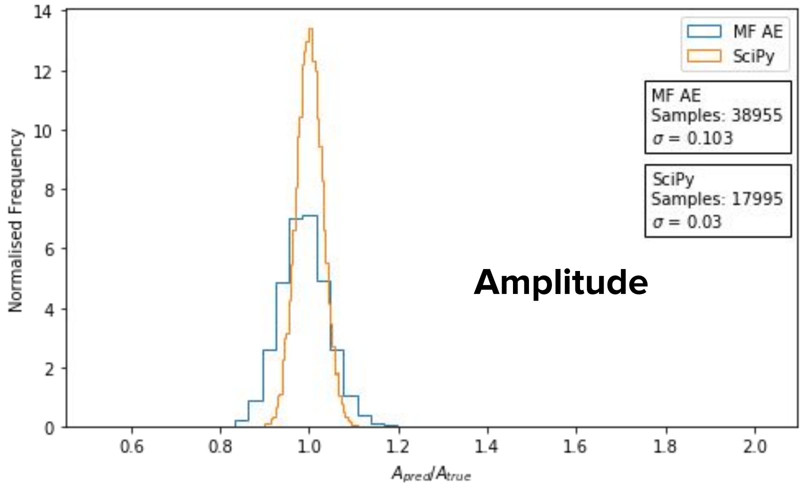


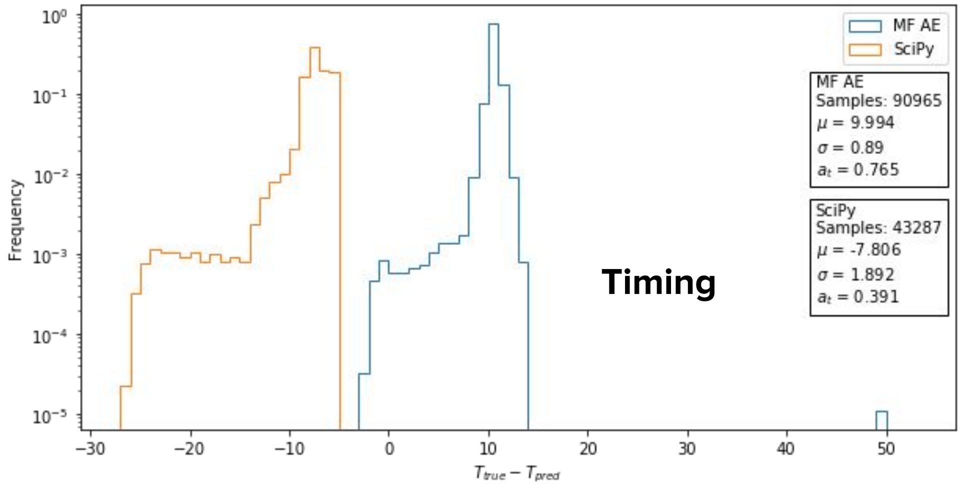


MF AE = Our model

SciPy = Scipy's peak finder

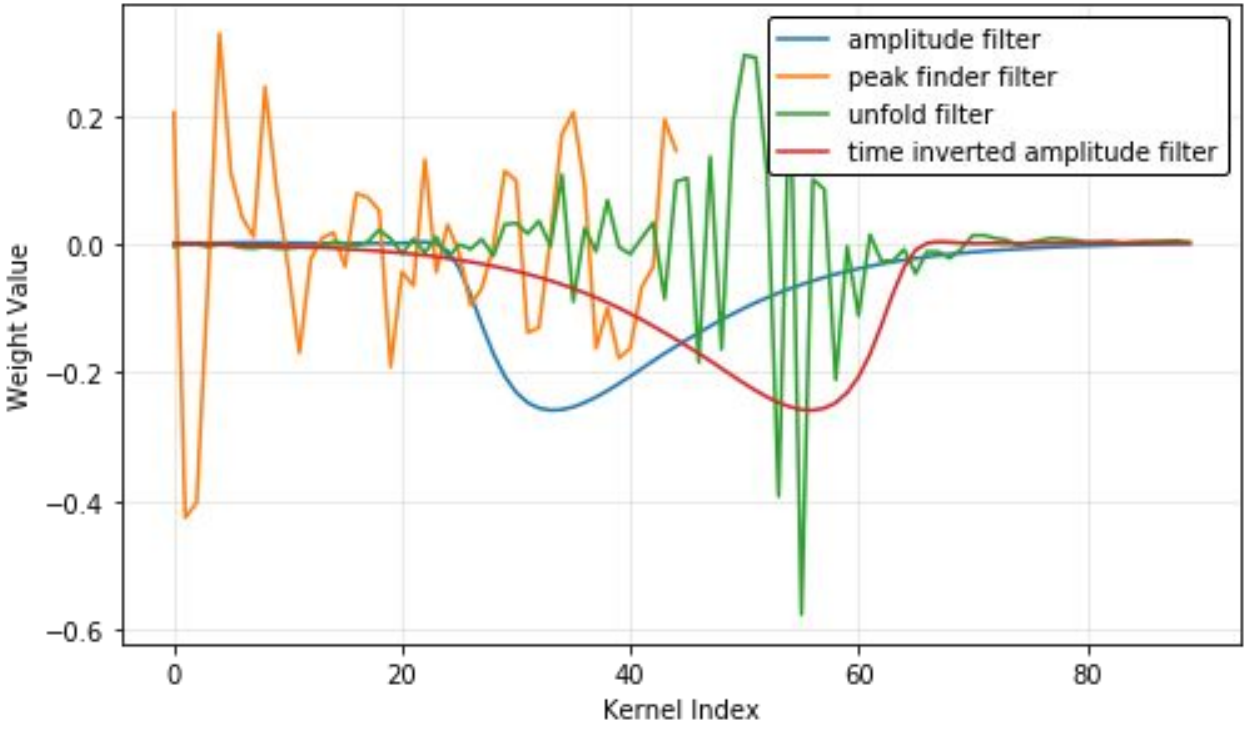
Parameter accuracy





	std. dev. amplitude	std. dev. timing
Our model	0.10	0.89
Scipy	0.03	1.89

Can we understand what it’s learnt?



- **Amplitude**: Pulse shape at the right hand edge. Zero-mean ringing ?
- **Reversed Amplitude**: matching input
- **Pulse finder**: second derivative of rising edge of pulse
- **Unfold layer**: Looks roughly like I’d expect but shifted to edge rather than centred

What now and next

Now:

- Improve stability and sparsity
- Test against more realistic scenarios
  - real data
  - real algorithms for comparison
- Finalise built-in pulse classification (see backups)

Next:

- Extend to target multiple types of pulse shapes (layers with multiple kernels)
- Multiple simultaneous waveforms (e.g. output of many PMTs)

b.krikler@bristol.ac.uk

Thank You

Analytical fit

b.krikler@bristol.ac.uk

Define chi-square for goodness of fit

$$\chi^2 = \sum (S_i - a_{\mathrm{fit}} T_i)^2$$

Minimal chi-square value when

$$\frac{d\chi^2}{da} = -\sum_i T_i (S_i - a T_i) = 0$$

So the least-squares fitted amplitude is given (semi-analytically) by:

$$a = \frac{\sum_i T_i S_i}{\sum_j T_j^2}$$

Which is just a weighted sum over samples in the waveform

Pulse separation

Can invert the output of the matched filter and disentangle the contributions of each pulse

A two-pulse signal would be formed by:

$$S_i = aT_{i-k} + bT_{i-m} + \epsilon_i$$

So the matched filter output becomes:

$$F_j = \sum T_i S_{i-j} = aT^2_{k-j} + bT^2_{m-j} + E_j$$

Where for simplicity we define:

$$T^2_k = \sum_i T_i T_{i-k} \quad \text{(the auto-correlation function)}$$

The response of the filter at the two pulse times is given by

$$\begin{pmatrix} F_k \\ F_m \end{pmatrix} = \begin{pmatrix} T^2_0 & T^2_{k-m} \\ T^2_{k-m} & T^2_0 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix}$$

Now invert and solve for *a* and *b*

b.krikler@bristol.ac.uk

Pulse Classification

- The Matched Filter gives the best signal-to-noise ratio
  - If the template used is the same as the underlying signal pulse shape
- Return to the chi-square, which is a measure of the noise:

$$\chi^2 = \sum_i (S_i - a_{\text{fit}} T_i)^2$$

So minimising the chi-square leads to the matched filter output

$$a = \frac{\sum_i T_i y_i}{\sum_j T_j^2} \implies F_j = \sum_i T_i' S_{i-j}$$

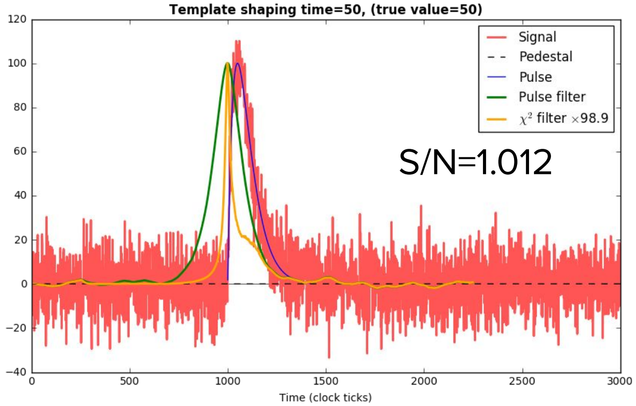
Expand the chi-square formula, and replace 'a' with the matched filter output

$$C_i = \frac{1}{N-1}\left(\sum_{j=0}^{N} y_{j-i}^2 - F_i^2 \sum_{j=0}^{N} T_j^2\right)$$

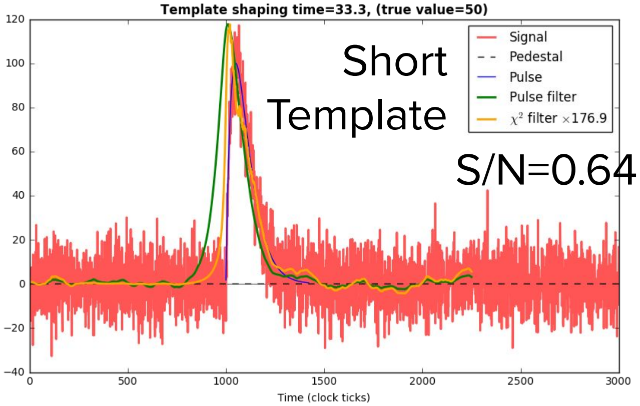
And finally define a signal-to-noise filter

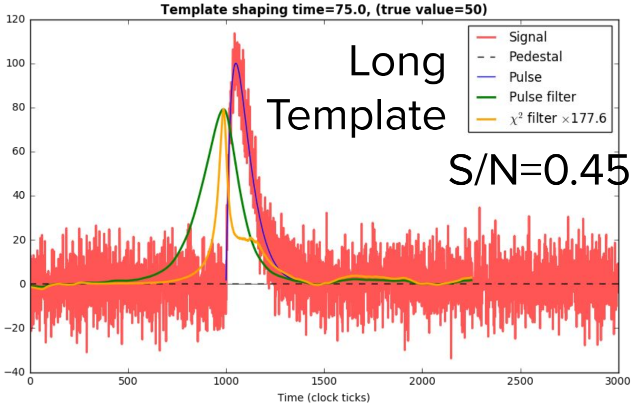
$$SN_i = \frac{F_i(N-1)}{\sum_{j=0}^{N} y_{j-i}^2 - F_i^2 T_0^2}$$

Pulse classification demo

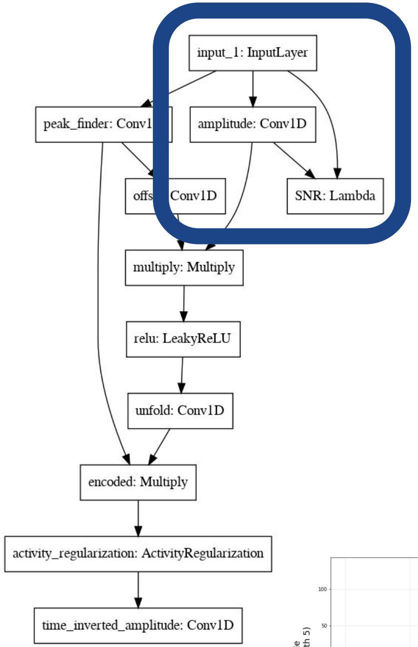


Correct Template



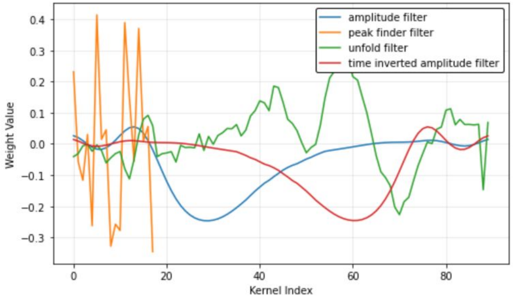


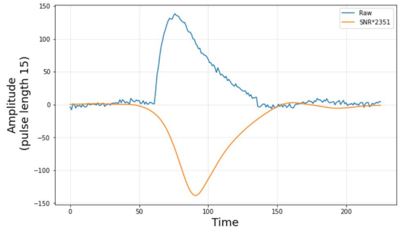
SNR activation in the model

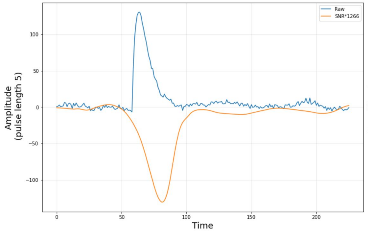


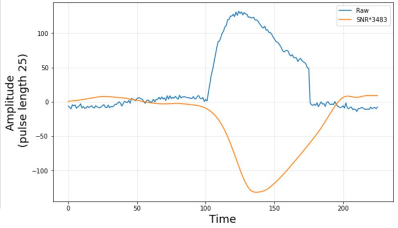
Train on pulses with shaping time = 15

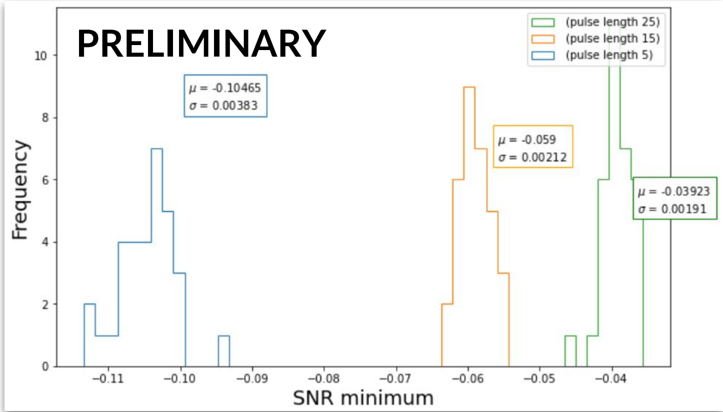
How does the SNR filter respond to different shaping times?







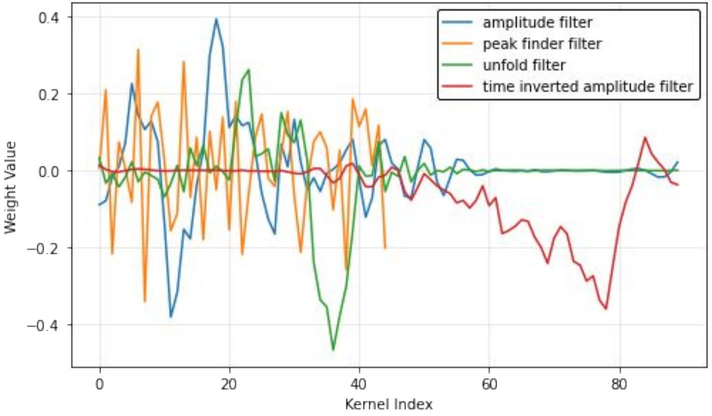


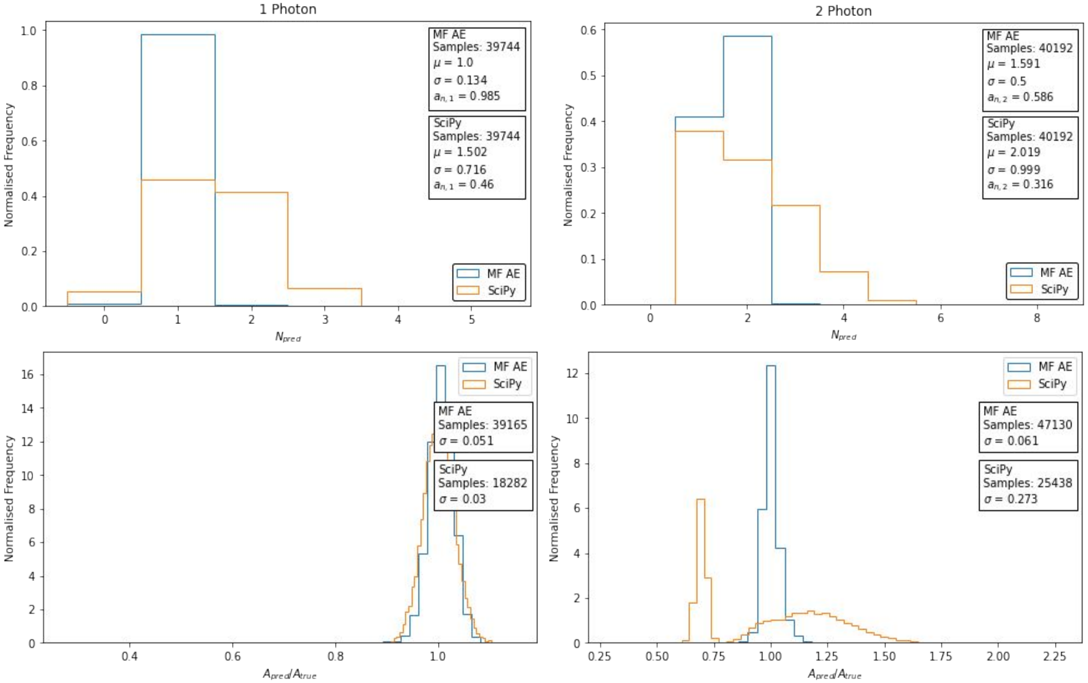


Issue in our input pulse normalisation: don't have the same "signal strength" for pulses with different shape...

⇒ "Does SNR activation of model trained on different pulse shapes correctly identify if a pulse matches its training set?

Interpretability vs performance



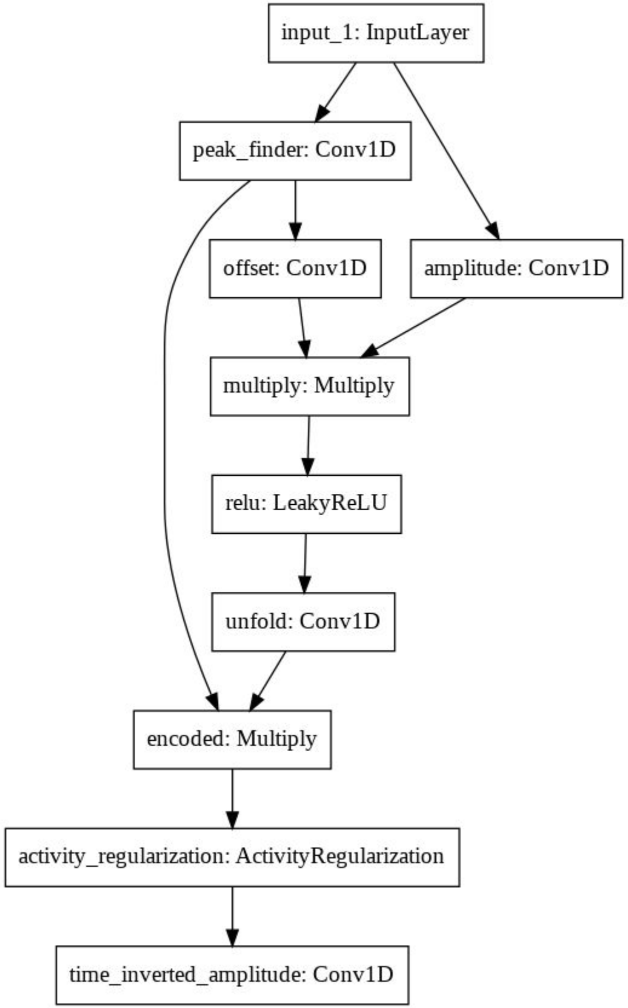


Relaxing desire for obvious interpretation can improve performance

Which makes for a heavily constrained model

```
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_1 (InputLayer)            [(None, None, 1)]    0

peak_finder (Conv1D)            (None, None, 1)      19          input_1[0][0]

amplitude (Conv1D)              (None, None, 1)      91          input_1[0][0]

offset (Conv1D)                 (None, None, 1)      91          peak_finder[0][0]

multiply (Multiply)             (None, None, 1)      0           amplitude[0][0]
                                                                 offset[0][0]

relu (LeakyReLU)                (None, None, 1)      0           multiply[0][0]

unfold (Conv1D)                 (None, None, 1)      91          relu[0][0]

encoded (Multiply)              (None, None, 1)      0           peak_finder[0][0]
                                                                 unfold[0][0]

activity_regularization (Activi (None, None, 1)      0           encoded[0][0]

time_inverted_amplitude (Conv1D (None, None, 1)      91          activity_regularization[0][0]
==================================================================================================
Total params: 383
Trainable params: 292
Non-trainable params: 91
```

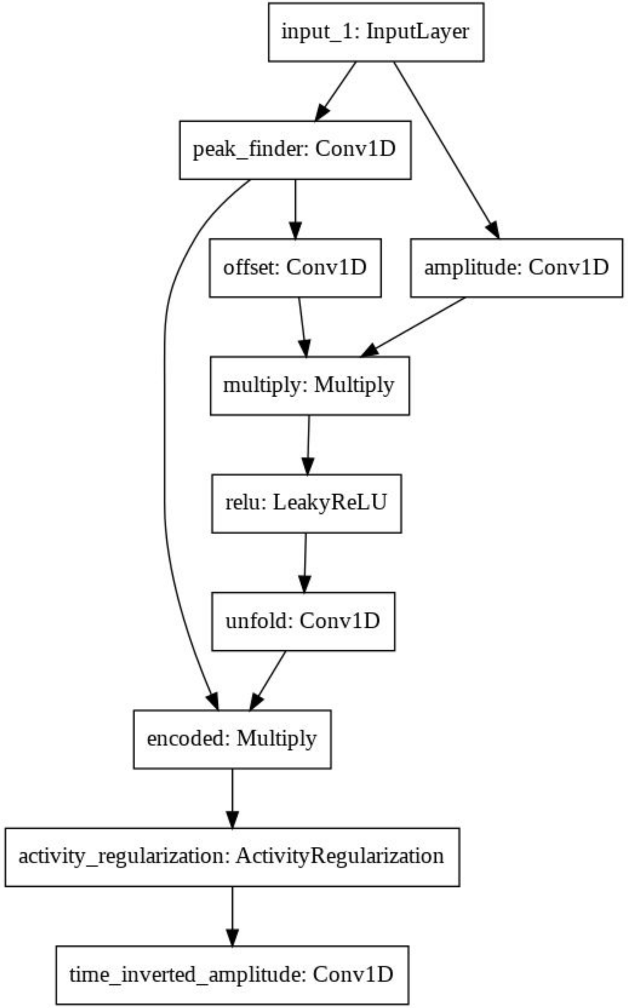
<300 trainable weights



Optimising the performance

There are many hyperparameters to be tuned:

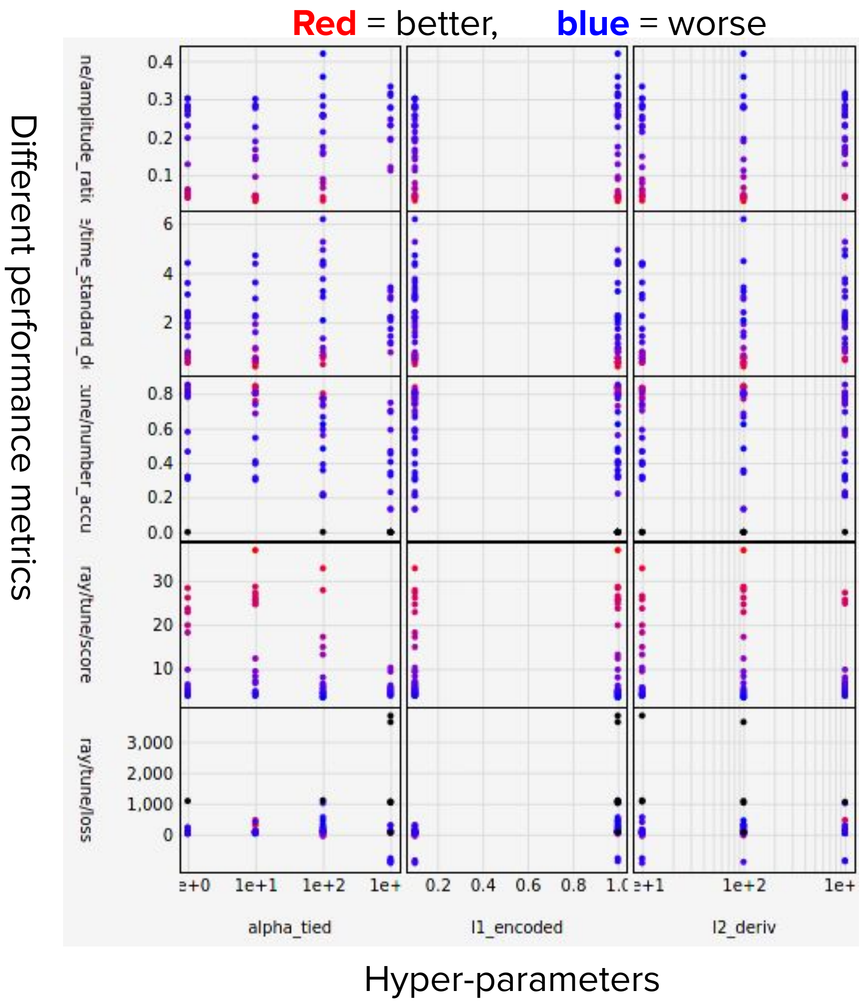
- Regularisation strengths:
  - L1 on the encoded space activation
    ➔ Makes it “sparse”
  - L2 on the derivative of the amplitude layer weights ➔ make this smooth
  - Mirror symmetry between input / output amplitude layers
- Architecture:
  - Lengths of the layers (3 params)
  - Activation functions for “extremising” the pulse finding layer
  - Relative alignment of filter outputs (2 params)
- Learning parameters:
  - Batch sizes, learning rate, early stopping, etc



Optimising the performance

Some preliminary conclusions:

- No single parameter studied strongly correlated to performance
  - Need to study learning parameters
- Stability is difficult:
  - For some points, model often collapses to predict nothing (vanishing gradients…?)



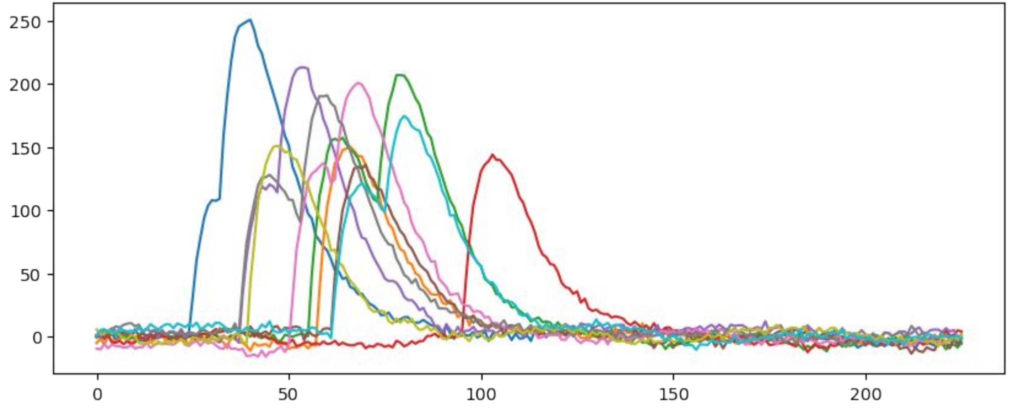
Generating input data

Toy detector sim for development:

- Input = impulse delta functions + white noise
  - Final waveform =
    input * convolve w. pulse shape + white noise
- Pulse shape: $T_i = x_i A e^{\left(1 - \frac{x_i}{\tau}\right)}$
- Shaping time, tau = 8 samples

Underlying waveform truth (not for training):

- 1 or 2 pulses per waveform
- Randomised pulse times



b.krikler@bristol.ac.uk