Signal recognition and background suppression by matched filters and neural networks for Tunka-Rex

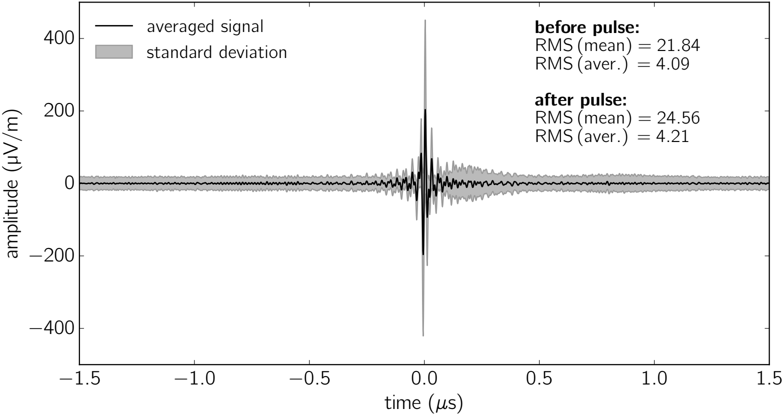
Dmitry Shipilov for the Tunka-Rex Collaboration

Irkutsk State University

June 12, 2018

Motivation

- Using pulse-shape information for lowering the threshold of signal detection
- Machine learning on traces with noise to extract features of background
- We develop and compare matched filtering and autoencoder based on CNN

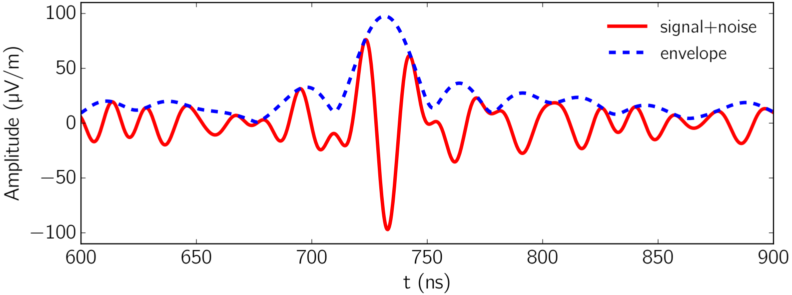


Average of 400 events; RMS should reduce by factor of 20 $\Rightarrow$ Noise not white

Standard method of signal reconstruction

Analytic signal $u(t) = s(t) + i\mathcal{H}[s(t)]$, where $\mathcal{H}$ is Hilbert transformation

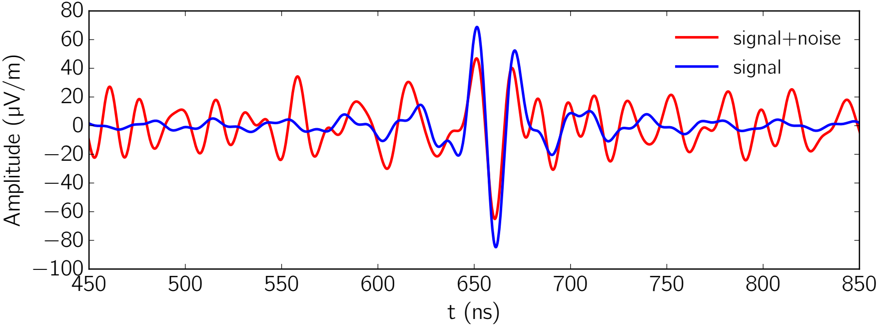
Envelope = Absolute value of analytic signal



Simulation set

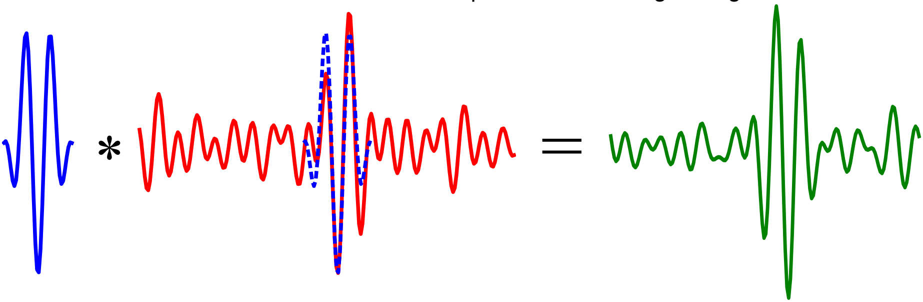
- 650k samples of Tunka background recorded in 2014-2017
- CoREAS simulations of Tunka-Rex signals (25k samples)
- Pulse is randomly located inside signal window (200 ns)
- Using single polarization ($v \times B$)
- Folded with Tunka-Rex hardware response
- Upsampling:
  - Factor 64 for matched filtering
  - Factor 16 for machine learning

Example of simulation



Matched filtering

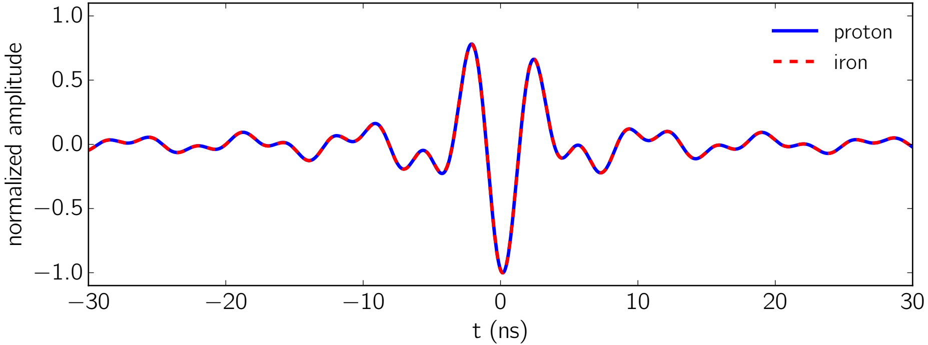
Matched filtering is based on the convolution of input trace with template, maximum of which defines the position of the original signal



Best performance of matched filtering is achieved in white noise conditions and is proportional to the power (length) of template

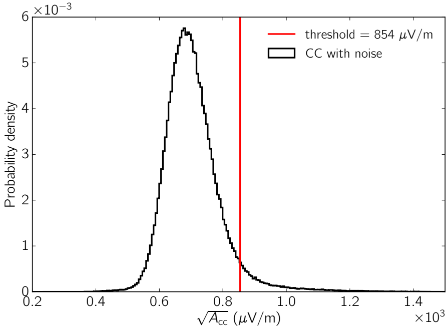
Templates for matched filtering

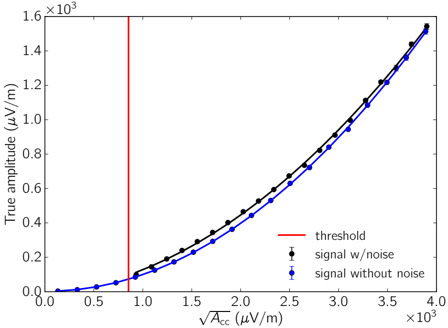
- Templates obtained from averaging of many CoREAS simulations
- Templates for proton and iron signals are the same
- In the present work we use single template with width of 60 ns



Threshold and amplitude reconstruction

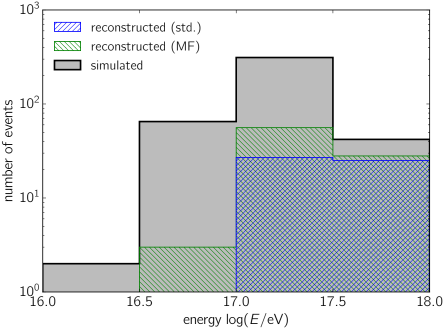
- Threshold is defined as 5% probability of false positives
- Amplitude is estimated as $f(\sqrt{A_{cc}})$ (amplitude of cross-correlation)

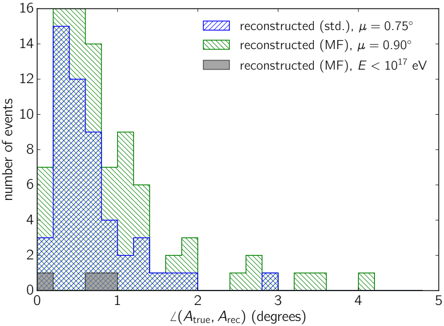




Full-pipeline reconstruction with matched filtering

Matched filtering is implemented in Tunka-Rex fork of Auger Offline
Reconstruction of CoREAS simulations (reproduction of 2012-2014 events)

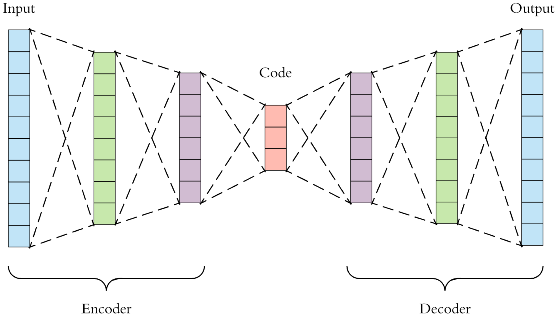




- Matched filtering has shown ability of detection of low-energy events
- Reconstruction of signal properties is under development

Chosen architecture (autoencoder)

- Unsupervised neural network with compressed representation
- Use Keras and Tensorflow with GPU support
- Based of 1D convolution layers
- ReLu $(\max(0, x))$ activation function
- Max pooling (and upsampling) after convolutional layers
- Binary crossentopy loss function and RMSprop optimizer
- Train networks via uDocker on SCC ForHLR II cluster



Learning strategy and training pipeline

Datasets:

- 25k samples for training

Subsets grouped by amplitudes:

- 10 – 100 µV/m (used in present work)
- 100 – 200 µV/m
- 200 – 300 µV/m

Training and evaluation:

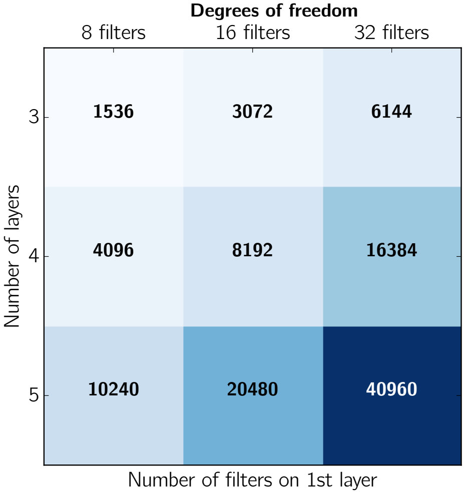
- Depth (D) and number of filters per layer as free parameters
- Primary evaluate by loss metrics
- Blind test with full-pipeline Offline reconstruction

i-th encoding layer is described by the following ($i = 1, ..., D$):

$$S_i = S_{\min} \times 2^{D-i}$$
$$n_i = 2^{i+N-1}$$

where S_i is a size of the i-th filter, n_i is a number of filters per layer
D and N are free parameters; $S_{\min} = 16$ is minimal size of layer (corresp. to few ns)

Degrees of freedom

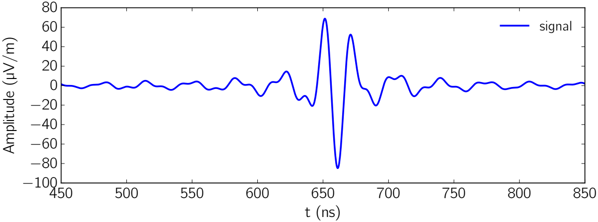


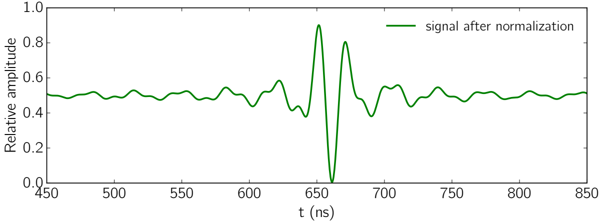
Filter used in MF consists of 786 points

Traces normalization

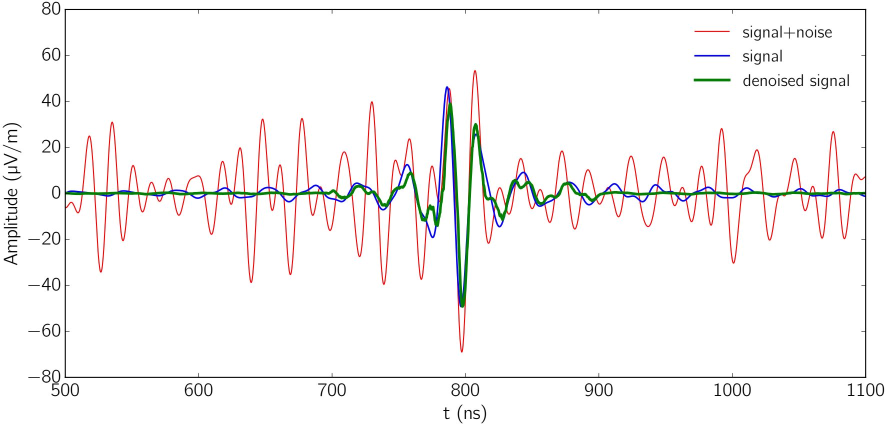
Traces should be normalized to 0–1 values, baseline should be located at 0.5 level

$$s_i' = \frac{s_i}{\max(u_i)} + 0.5, \text{ where } u_i \text{ is envelope of trace}$$



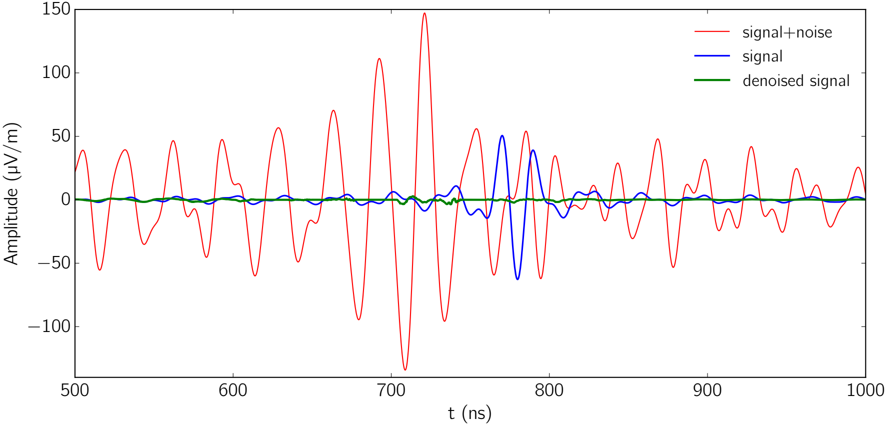


Example: correct identification



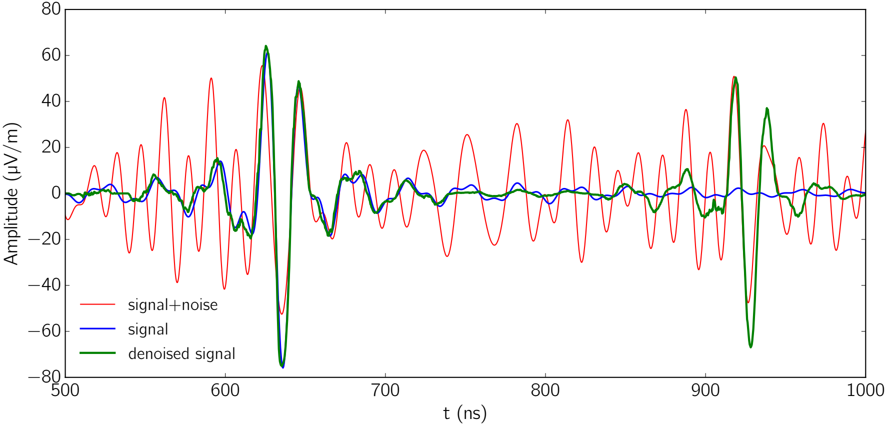
True signal and noise are identified correctly, noise is removed

Example: no identification



True signal is heavily distorted by noise, and removed as background

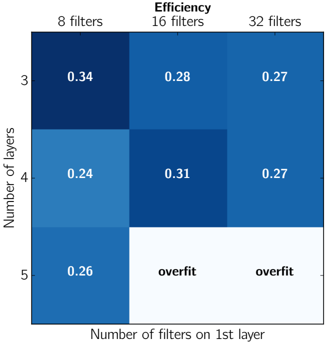
Example: double identification

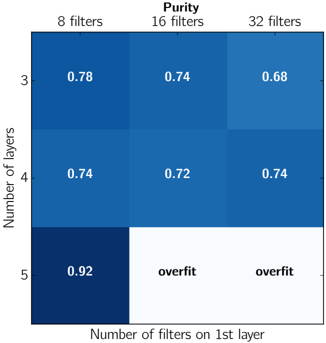


Signal-like RFI is identified as signal

Threshold and metrics

- Threshold amplitude of denoised signal is defined as 5% tolerance to false positives
- Efficiency: $N_{\mathrm{rec.}}/N_{\mathrm{tot.}}$, fraction of events passed the threshold
- Purity: $N_{\mathrm{hit}}/N_{\mathrm{rec.}}$, fraction of events with reconstructed position of the peak: $|t_{\mathrm{rec.}} - t_{\mathrm{true}}| < 5$ ns

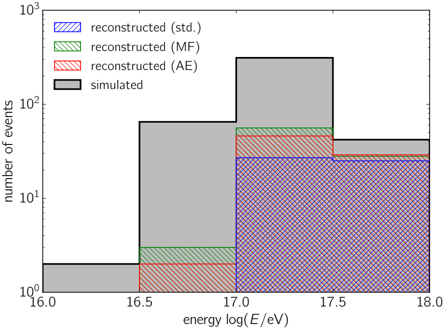


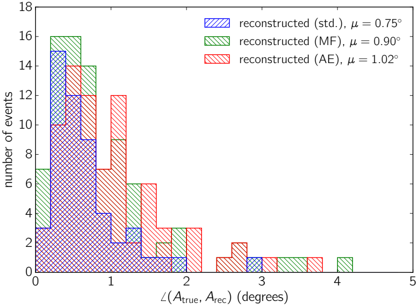


Best architecture contains $N_{\mathrm{dof}} = 10240$

Full-pipeline reconstruction with autoencoder

Autoencoder is binded with Tunka-Rex fork of Auger Offline
Reconstruction of CoREAS simulations (reproduction of 2012-2014 events)





- Autoencoder shows performance similar to matched filtering
- Reconstruction of signal position (`TunRaC` + `Offline`) and properties is under development

Conclusion

- The signal reconstruction of Tunka-Rex is improved with matched filtering and denoiser
- Classical (MF) and modern (AE) approaches show the similar performance, which is better than standard method.
- Software is ready and almost implemented in standard reconstruction

Few remarks on machine learning[1].

- "Stack more layers" rule works, but requires larger training sets
- Signal properties of denoised traces are under investigation
- We plan to try different architectures of neural networks

[1] The work was funded by the Russian Science Foundation (the grant No. 18-41-06003)

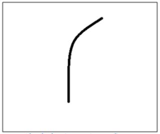
Working environment for neural networks

- We create complete set of necessary tools for neural network's training
- Converter from ADST Root to NumPy Binary format
- Tools for creating datasets, training networks and evaluating them
- We train networks via uDocker on ForHLR II cluster
- Binding with Offline

Example feature extraction

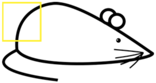
0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter



Visualization of a curve detector filter

Original image



Visualization of the filter on the image

Visualization of the receptive field

0	0	0	0	0	0	30
0	0	0	0	50	50	50
0	0	0	20	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0

Pixel representation of the receptive field

*

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter

Multiplication and Summation = (50*30)+(50*30)+(50*30)+(20*30)+(50*30) = 6600 (A large number!)

Visualization of the filter on the image

0	0	0	0	0	0	0
0	40	0	0	0	0	0
40	0	40	0	0	0	0
40	20	0	0	0	0	0
0	50	0	0	0	0	0
0	0	50	0	0	0	0
25	25	0	50	0	0	0

Pixel representation of receptive field

*

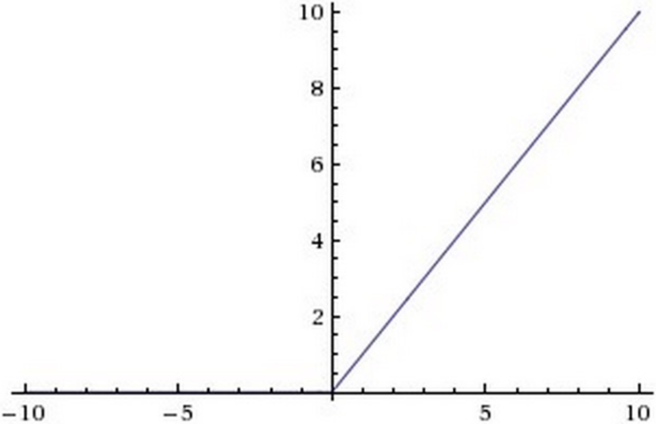
0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter

Multiplication and Summation = 0

ReLu

- ReLu (rectified linear unit) activation



- f(x) = max(0,x)

Max pooling

- MaxPooling

Acknowledgements

- MaxPooling