機械学習を活用した高計数率ドリフトチェンバーのヒット再構成

High-rate drift chamber hit reconstruction with machine learning technique

東京大学
THE UNIVERSITY OF TOKYO

ICEPP
The University of Tokyo

研究拠点形成事業
Core-to-Core Program

内山 雄祐 (The University of Tokyo)
on behalf of MEG II collaboration

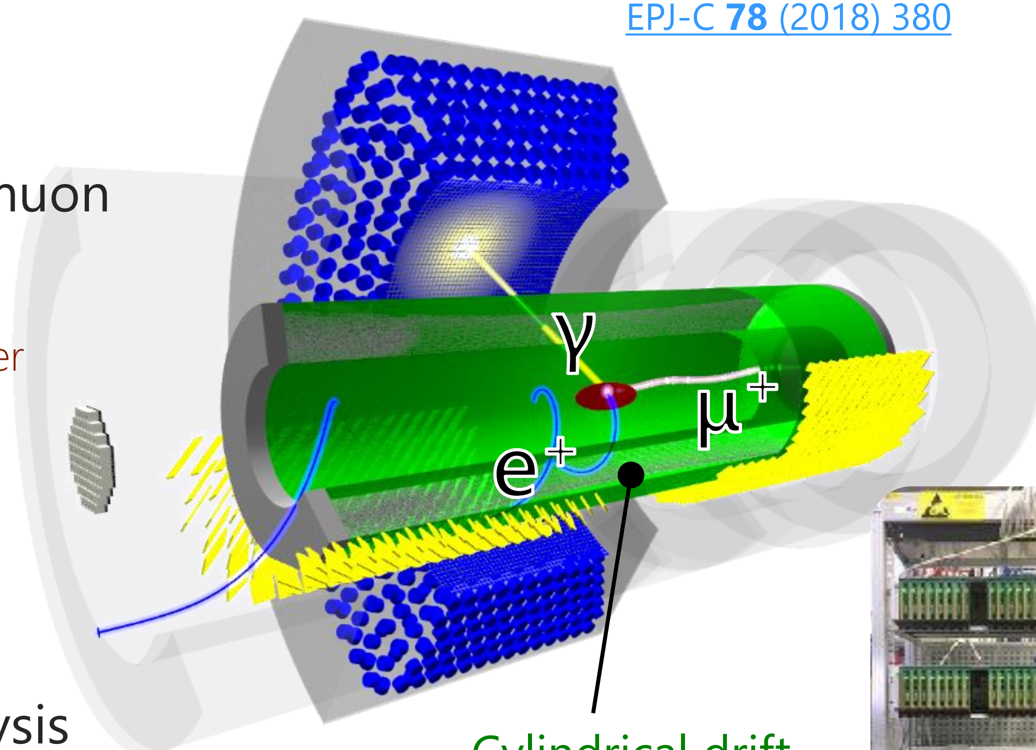
日本物理学会2021年年次大会
令和3年3月14日

14aT3-7

MEG II experiment

- Search for rare muon decays
  - to find definitive evidence for BSM
- Use world's most intense DC muon beam
  - continuously emits 7×10^7 e^+/s
  - detected by a cylindrical drift chamber
- The detector signals are read out as waveform
  - by DRS4 waveform digitizer
  - 1024 points @ 1.2 – 1.8 GSPS
- All the detectors as well as computing resource and analysis framework have been prepared.
- Starting physics data taking in 2021
  - Engineering data were taken in previous years.
  - In this study, use 2020 data.

EPJ-C **78** (2018) 380



Cylindrical drift chamber (CDCH)
($\sim 1.6 \times 10^{-3} X_0$, $\sigma_p \sim 100$ keV)

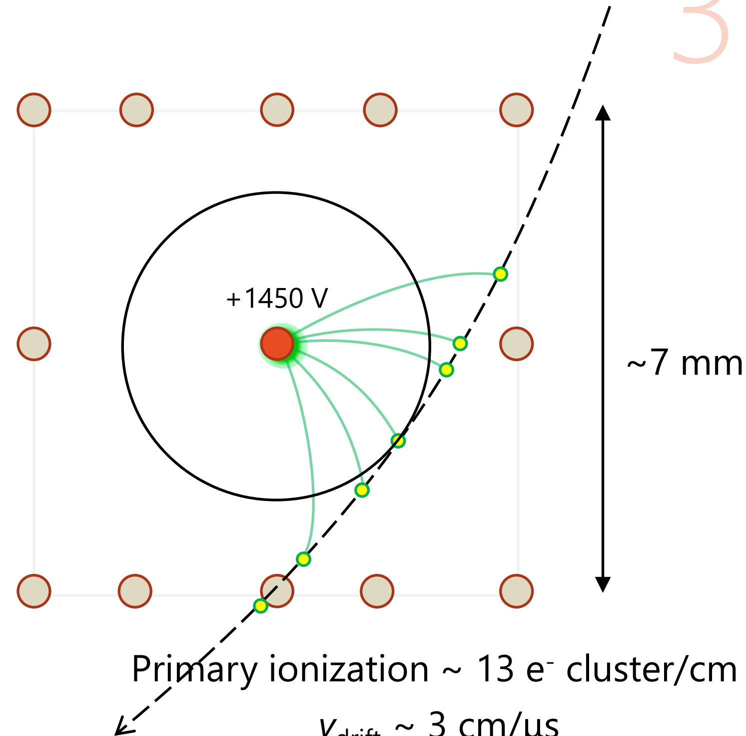
Drift chamber: a nutshell

Signal formation

1. Charged particle generates primary ionization clusters discretely in gas
2. The ionized e⁻s drift to an anode wire and form avalanche near the wire

Reconstruction

1. Measure the timing of the 1st cluster
2. Draw a drift circle
3. Fit a track to the drift circles



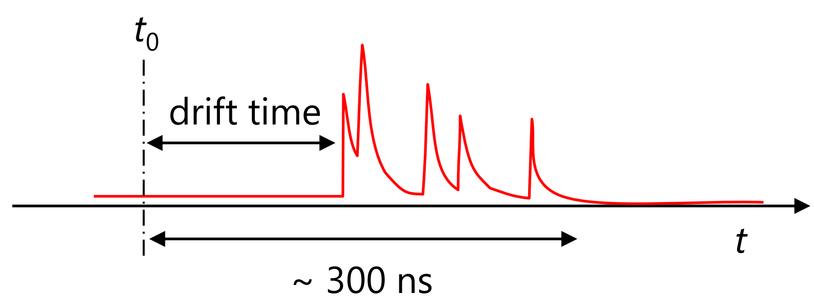
Primary ionization ~ 13 e^- cluster/cm

v_{drift} ~ 3 cm/μs

Gas gain ~ 5×10^5

MEG II CDCH: an ultra low-mass chamber

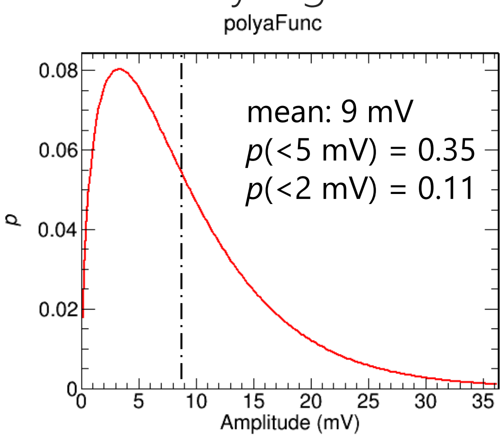
Gas: He:iC_4H_{10} = 90:10
Wires: 20 μm W anode
+ 40/50 μm Al cathode
2 m long, 9 layers,
1152 readout cells in total

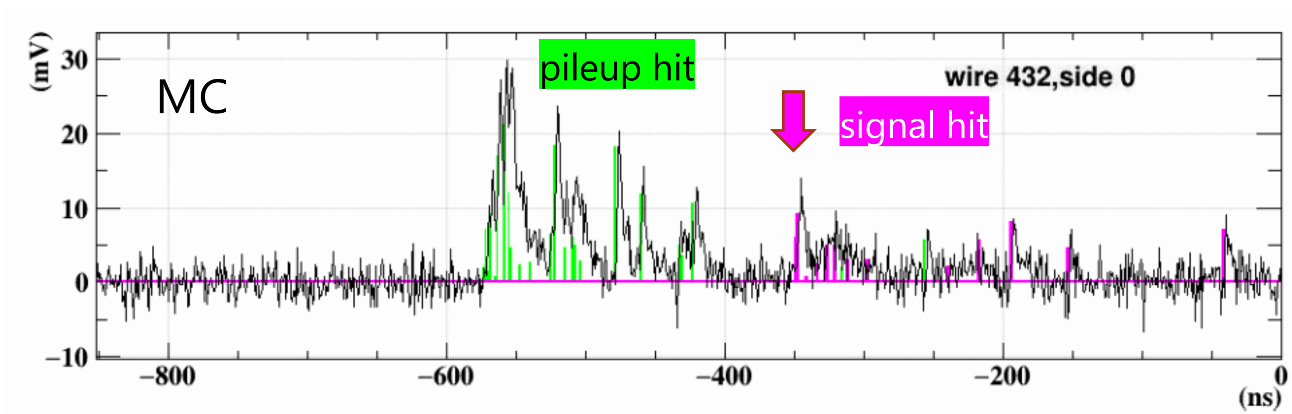


Challenges

- Detecting the 1st cluster signal is essential for the experiment
  - The efficiency is directly connected to the e^+ reconstruction efficiency, and thus, search sensitivity.
- Two difficulties:

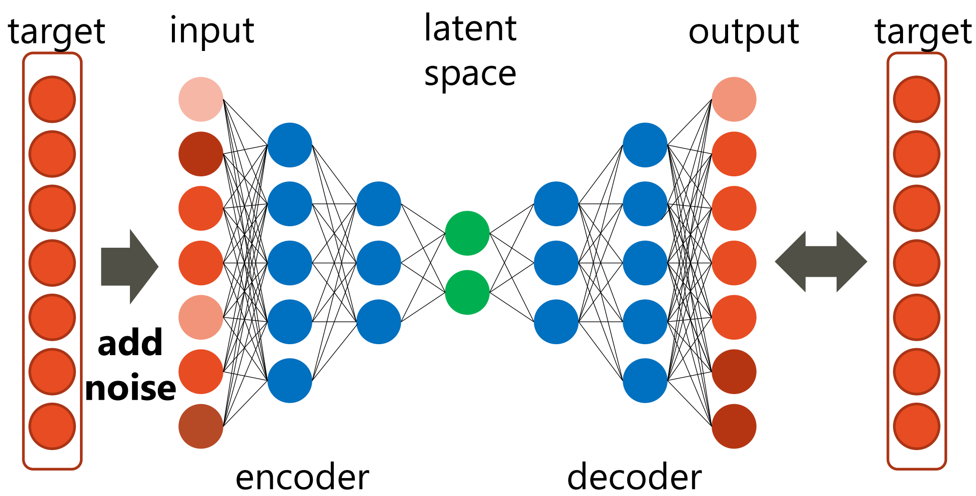
1. S/N
   - The amplification in avalanche process (gas gain) has large fluctuation obeying a Polya distribution. The 1st cluster signal can be very small.
2. Pileup
   - Very high hit rate in MEG II: up to 1.7 MHz per cell, 35% occupancy in 250ns.





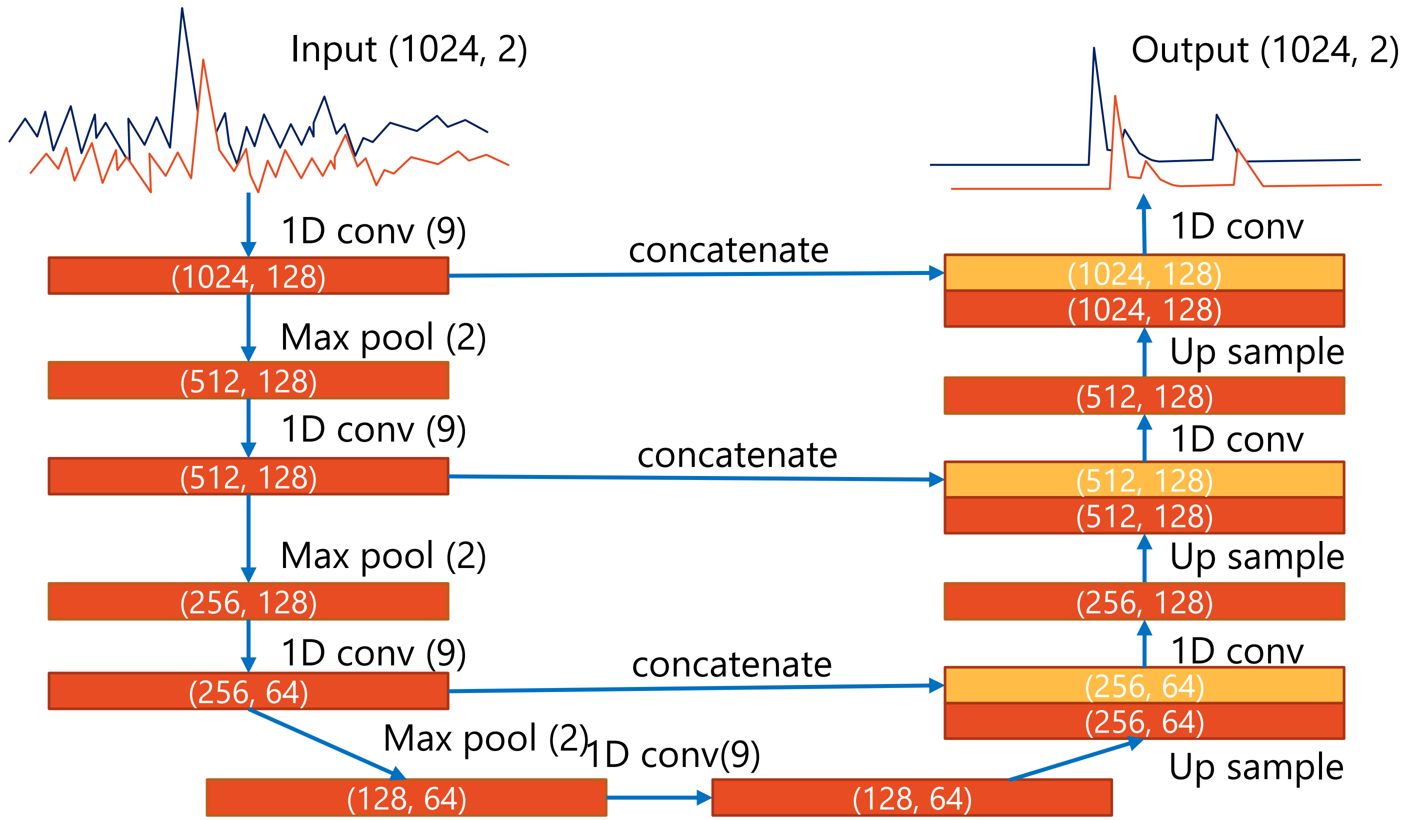
Apply ML to the complicated waveform analysis.
As the first step, apply to noise reduction to improve S/N

Denoising autoencoder



- **Autoencoder**: train network so that output = input
  - Latent space holds the features of signal
- **Denoising autoencoder**: add noise to the target for the input
  - More effectively learns the feature and becomes more robust
  - Can be used to denoise noisy data
- Apply to waveform data
  - Use MC signal w/o noise + noise data (random trigger data w/o beam)
    (mix events randomly in time at 7×10^{7} s^{-1}) (non-Gaussian non-white noise)
  - Tried in two directions: estimating signal or estimating noise

The model: signal estimation

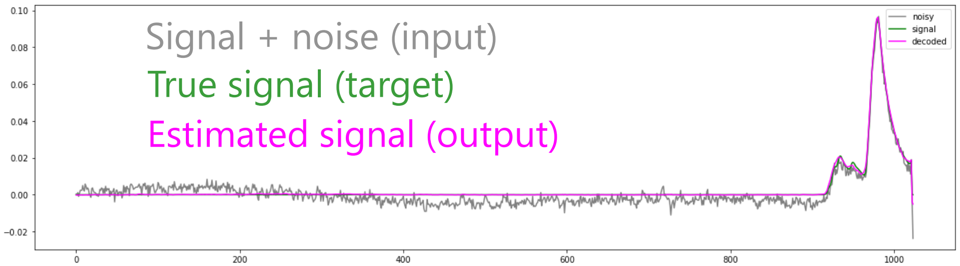


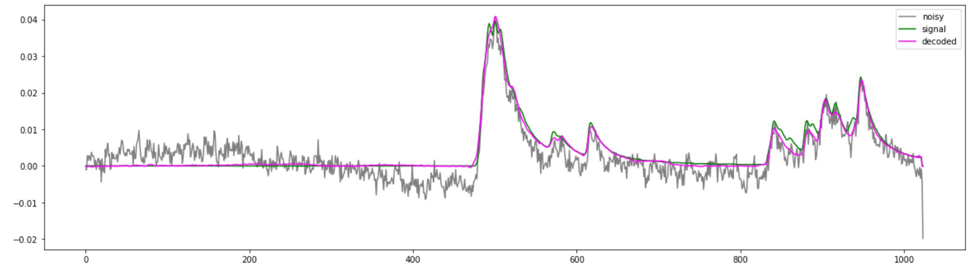
Adam, ReLU,
512 batch size,
100 epochs,
373k parameters

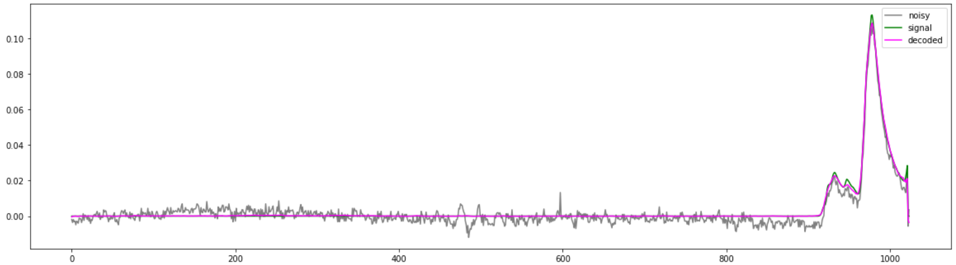
Extend the denoising autoencoder with:

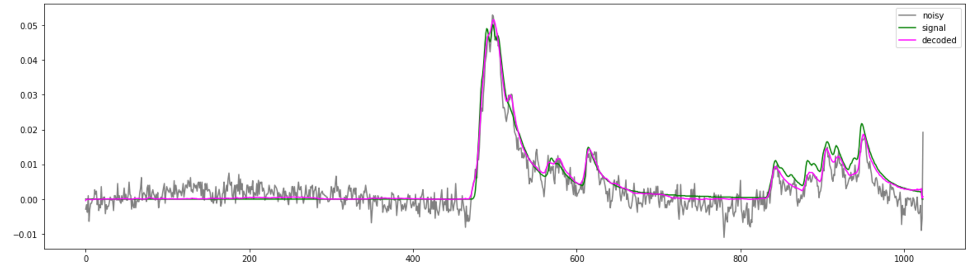
- 1D convolutional network
- 'UNet'-like structure with skip connections https://arxiv.org/abs/1505.04597 (image segmentation)
- 2-channel input with 2-end waveforms from a wire
- Use 'mean squared logarithmic error (msle)' loss function.
  - with 1 mV offset to avoid 0-division.

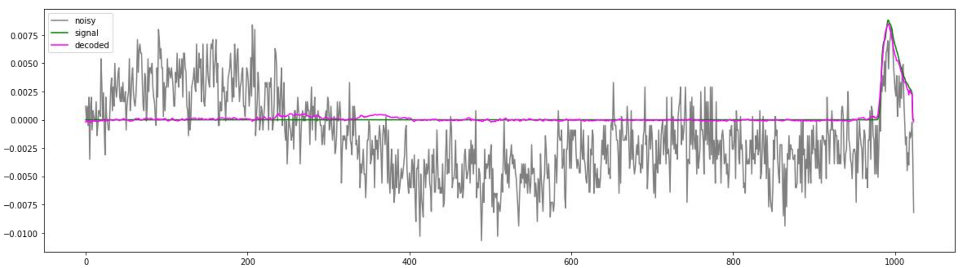
Signal estimation with 1D autoencoder

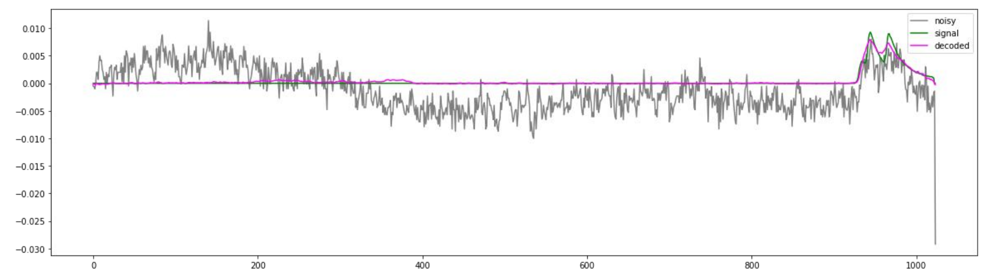


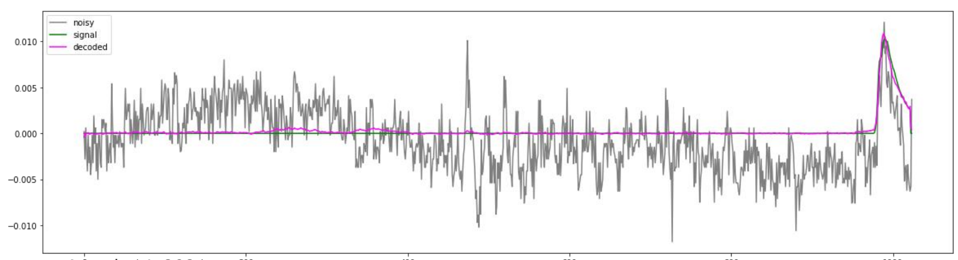


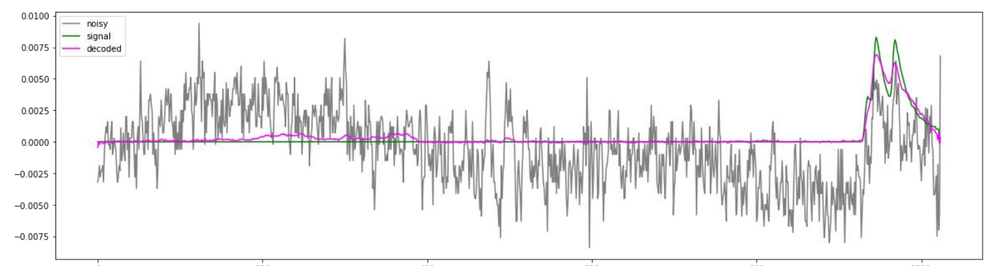






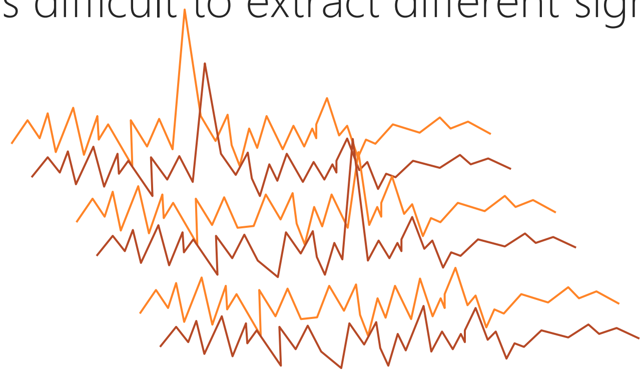


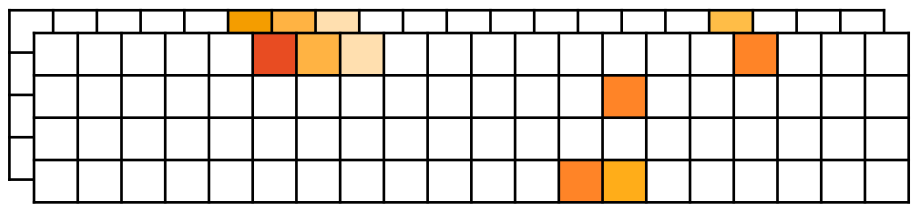




Noise estimation

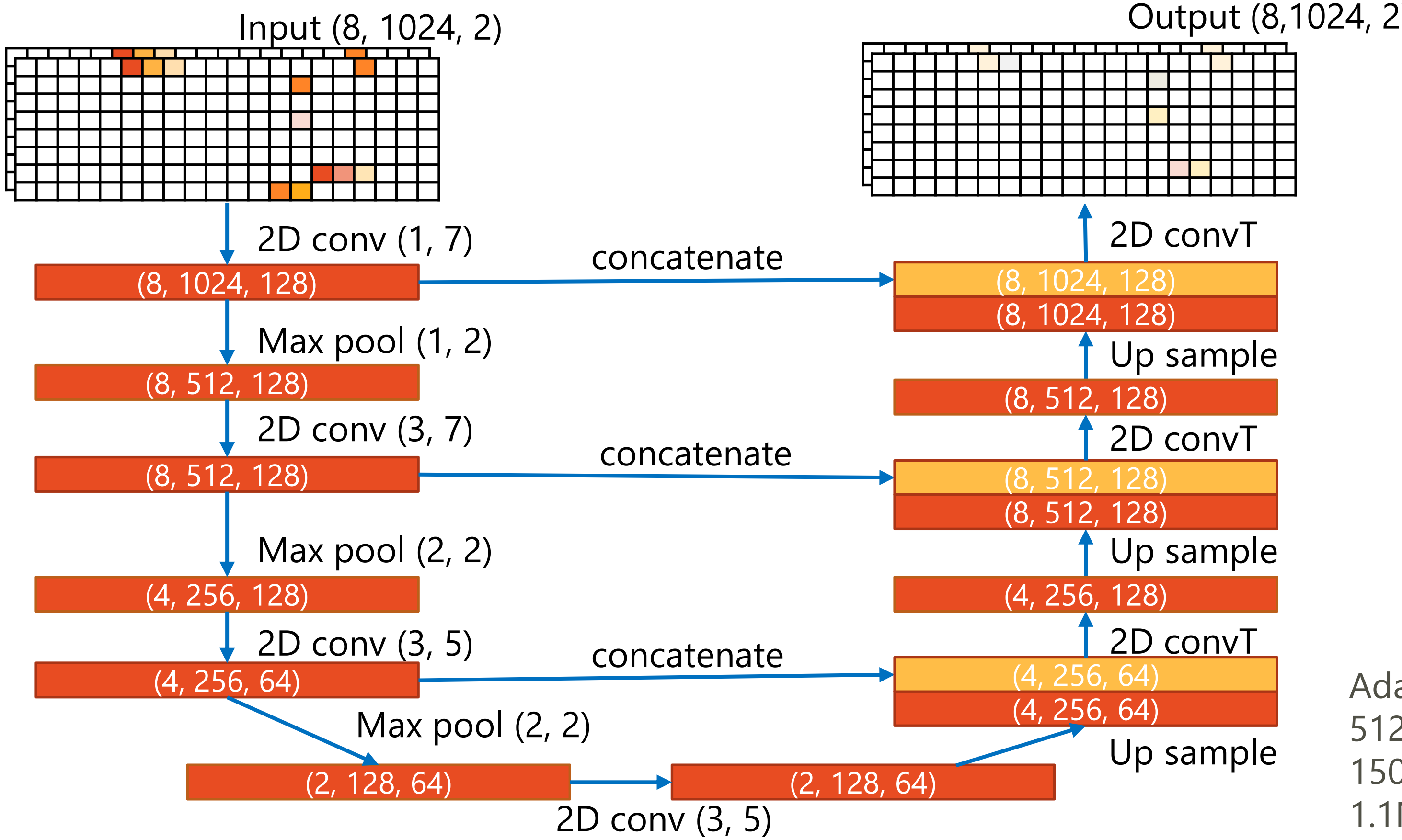
- Want to use other wires information together
  - which contains information for coherent noise.
- However, neither increasing input channels nor extending to 2D input works well.
  - It is difficult to extract different signal patterns in different wires with CNN.





- Change the view of the data → estimate noise instead of signal.
  - Coherent noise changes gradually over different wires. → 2D CNN can deal with it well.
  - Existence of signal masks the noise, but estimate it using other wires waveform.
  - Group 8 wires that connect to the same front-end cards into an input.

The model: noise estimation

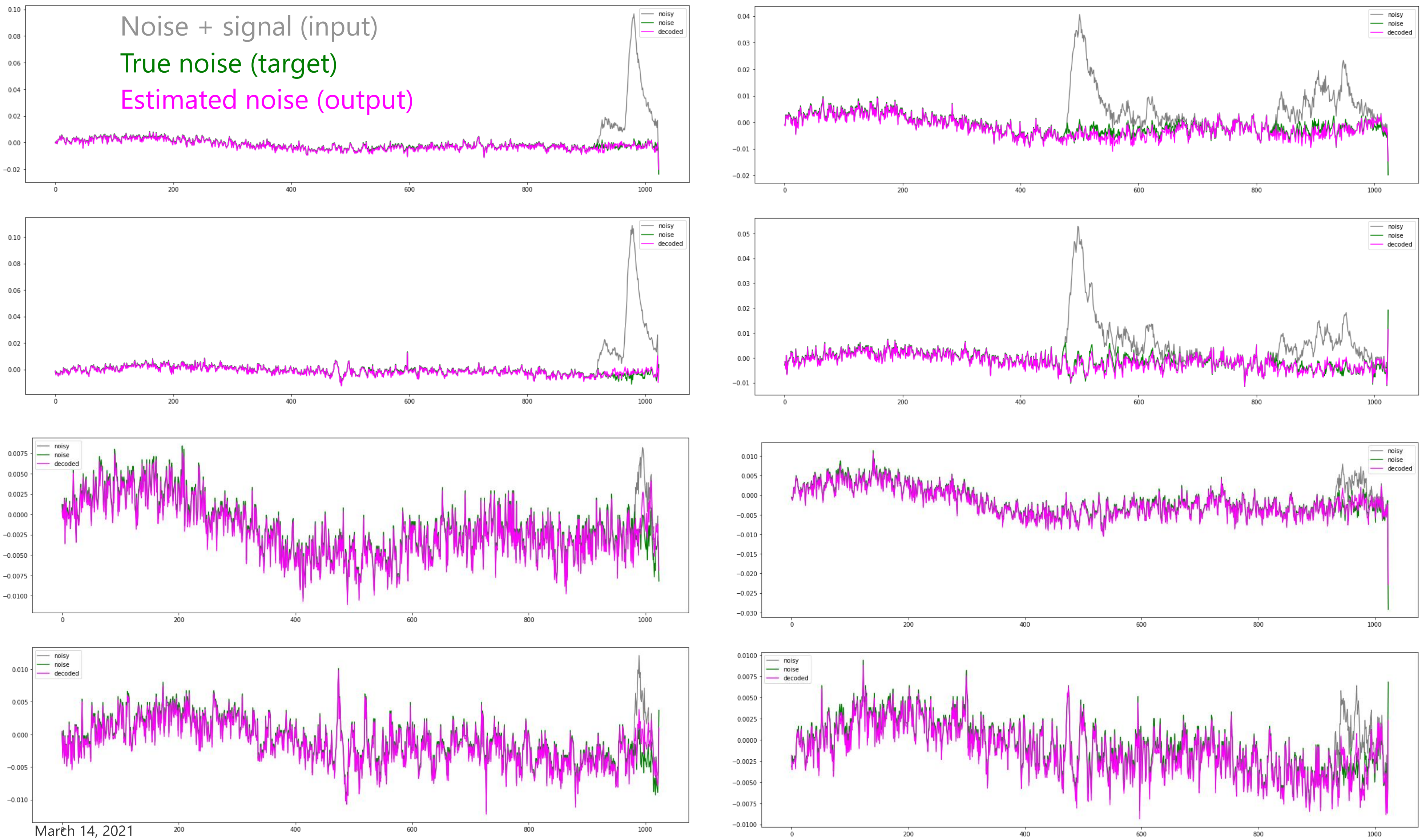


Adam, ReLU,
512 batch size,
150 epochs,
1.1M parameters

- 2D convolutional network
- 'UNet'-like structure with skip connections https://arxiv.org/abs/1505.04597 (image segmentation)
- 2-channel input with 2-end waveforms from 8 wires
- Use 'mean squared error (mse)' loss function.

This is equivalent to the residual learning

Noise estimation with 2D CNN autoencoder



Implementation

TRAINING

- Tensorflow 2.4 + Keras
- in Python3.7
- on Google Colab
- with Tensor Processing Unit (TPU)
- convert to ONNX format

INFERENCE

- ROOT based MEG II reconstruction framework
- in C++17
- ONNX Runtime C++ API
- with CPU single thread (Xeon Gold 6138 2.0 GHz)

High flexibility × Easy maintenance

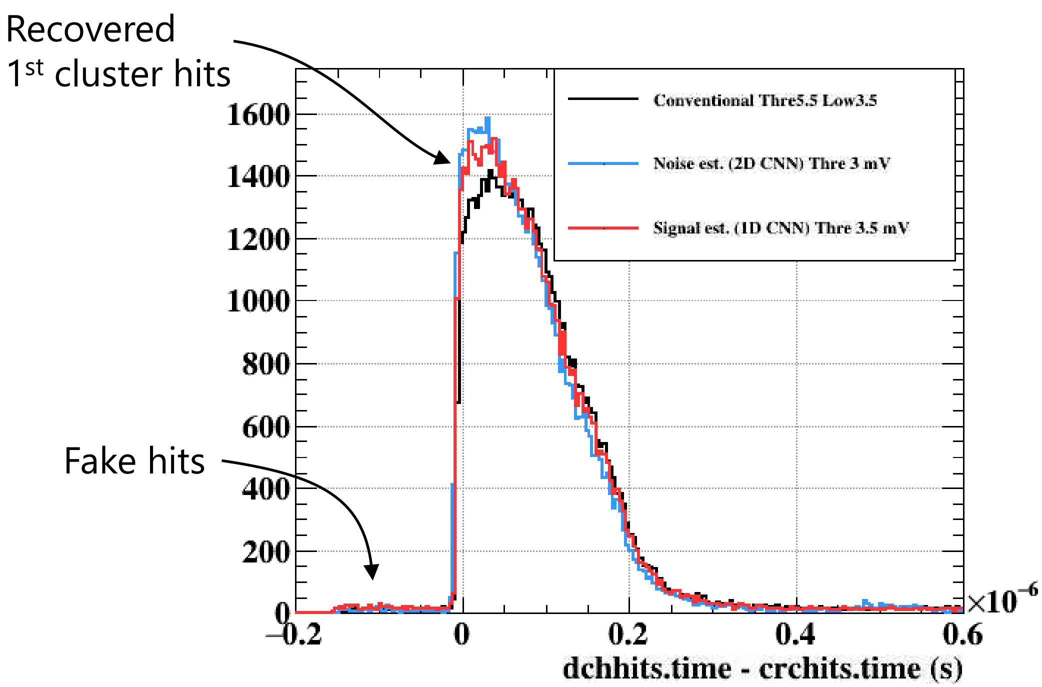
Use one's preferred package (one good at the problem under consideration) for model building & training.

Use a common interface in C++ to use the trained model in inference/prediction.

GPU/TPU in cloud are available for training, while only CPU (single thread) is available in the MEG II resource & framework.

Results

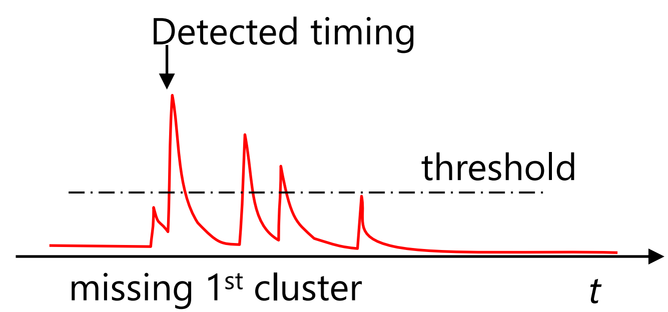
- Apply to cosmic-ray (low rate) data in 2020 run.
  - 128 wires were readout (only 1/5 of the whole).
  - Triggered by scintillation counters. → t_0
- Evaluate the performance from the hit time distribution



1st cluster detection efficiency improves.

- Thresholds are lowered from 5.5 mV to 3.5 mV with signal estimation, 3.0 mV with noise estimation.
- Signal estimation tends to generate fake pulses from noise fluctuation.

The number of hits matched with CR tracks increases by 17%.



Next

- Improve
  - Tune hyperparameters
  - Increase training samples or augmentation
  - Develop a better model
- Speedup inference
  - Compress the model with pruning
  - Use a simpler or more efficient model with distillation

	Signal estimation	Noise estimation
Training (TPU)	2.6 s/epoch	1.3 s/epoch
Inference (CPU)	1.2 s/events	1.5 s/events

* only 1/5 of full readout wires
* 60k waveforms used in training

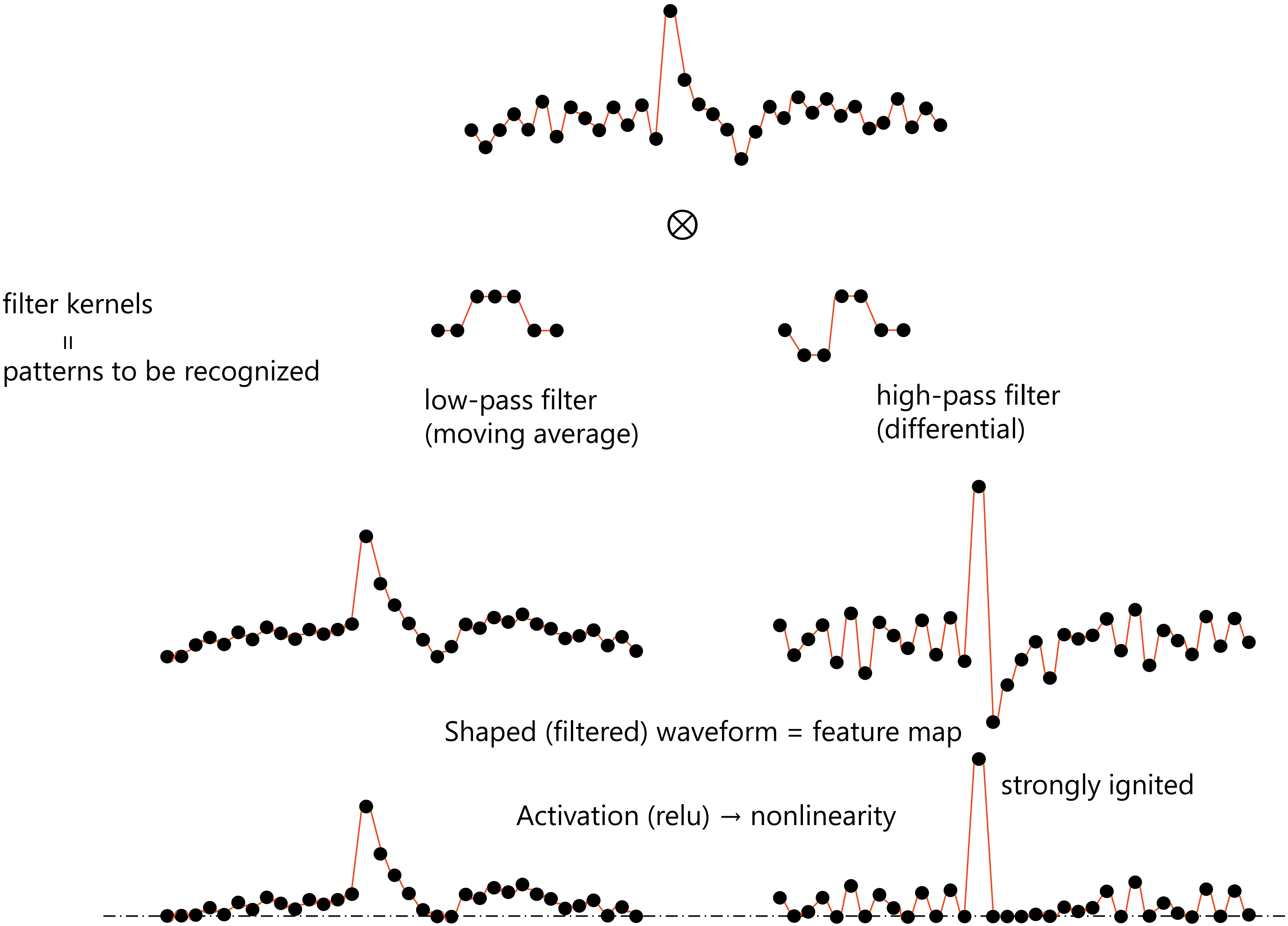
- Apply to muon beam data
- Extend to directly detecting hits (times and amplitudes) from the input waveforms
  - Combine the noise & signal networks with transfer learning.
  - Disentangle clusters from different hits (pileup).
  - Require delicate MC tuning and precise data calibration.

Conclusions

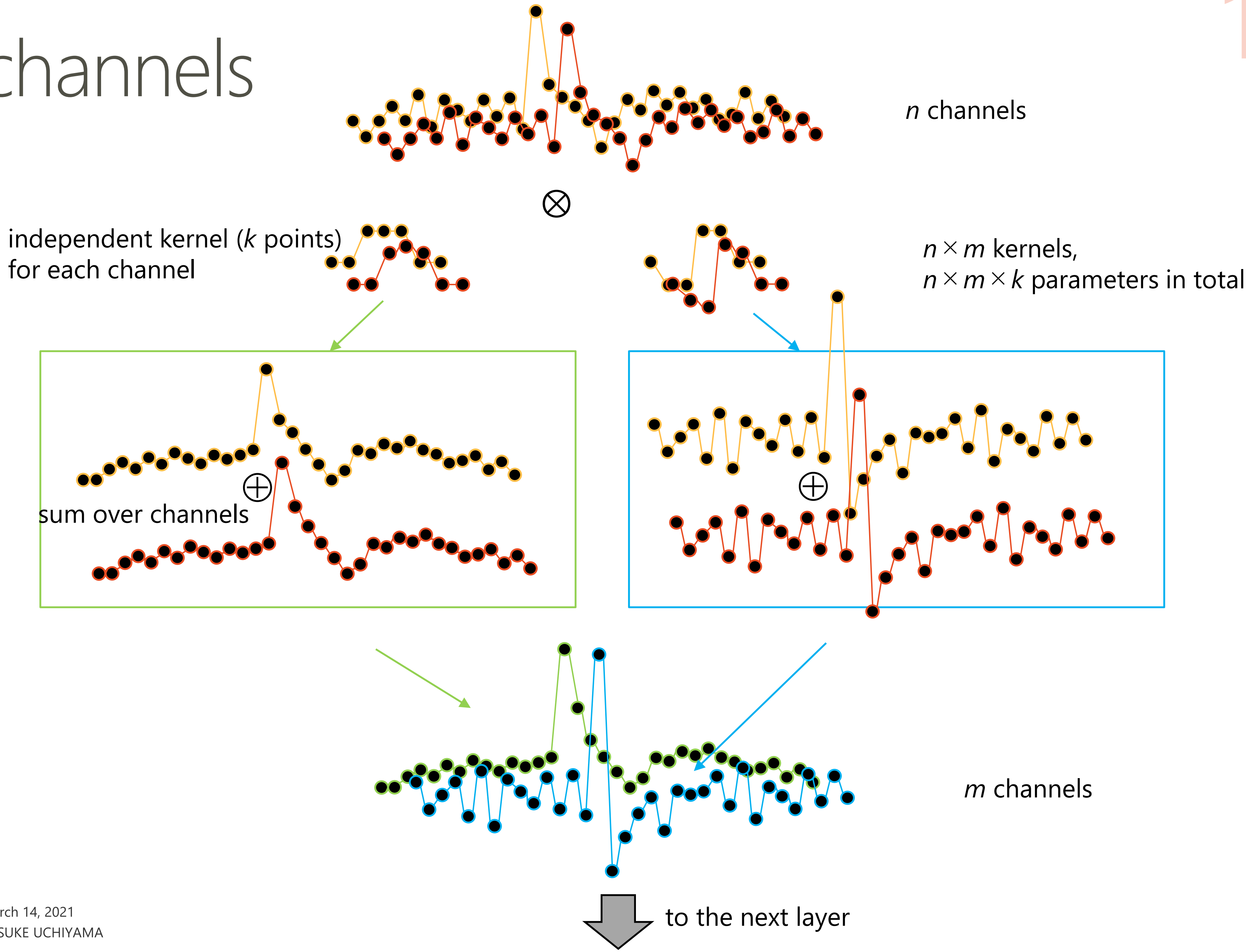
- Applied denoising autoencoders to MEG II CDCH waveform data.
- The models certainly learn the features of signal and noise.
- Denoising enables lowering hit detection threshold and improves the detection efficiency of the 1st cluster signal.
  - Superior to conventional waveform analysis with digital filters.
  - A promising technique to improve the experiment sensitivity.

- Flexible & sustainable framework matching HEP analysis was established.
- Computation time in inference is an issue for practical application,
  - in which only single thread CPU is available.
  - Speeding up by a factor 5 is desirable.

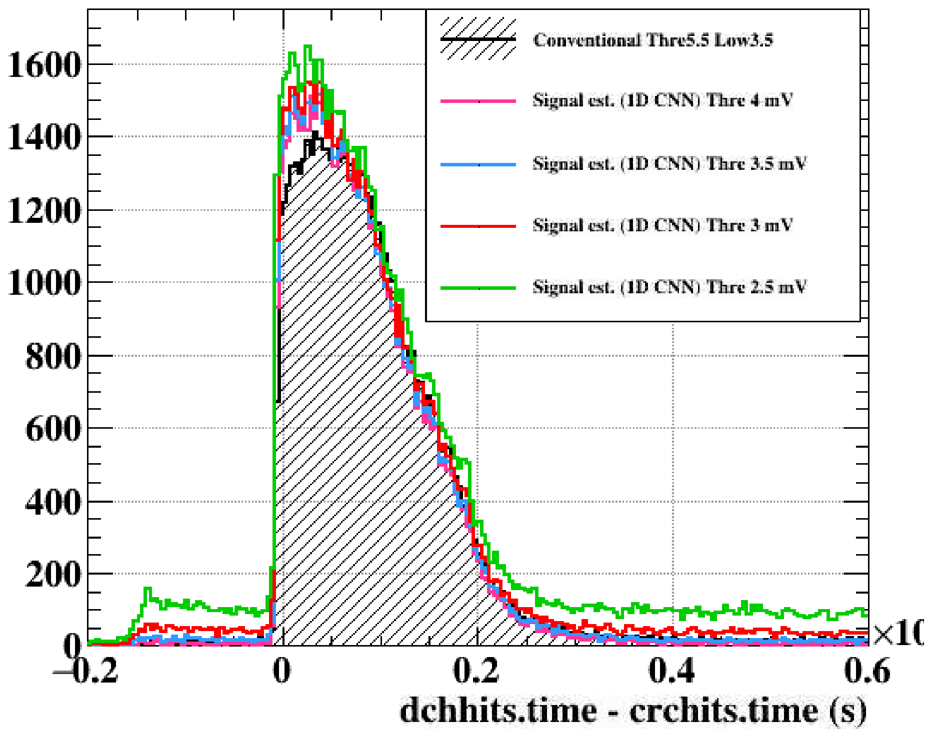
- 1D conv ⇔ FIR digital filter. Apply multiple filters to catch different patterns.
- Activation → nonlinear response.
- CNN → position invariant signal detection,
  but not scale invariant → learn from data. ←Augmentation will help it.
- Pooling → allow timing variation, good for local pattern recognition but loose global timing information
- U-net skip connection → recover global timing information


⊗
filter kernels
=
patterns to be recognized
low-pass filter
(moving average)
high-pass filter
(differential)
Shaped (filtered) waveform = feature map
strongly ignited
Activation (relu) → nonlinearity

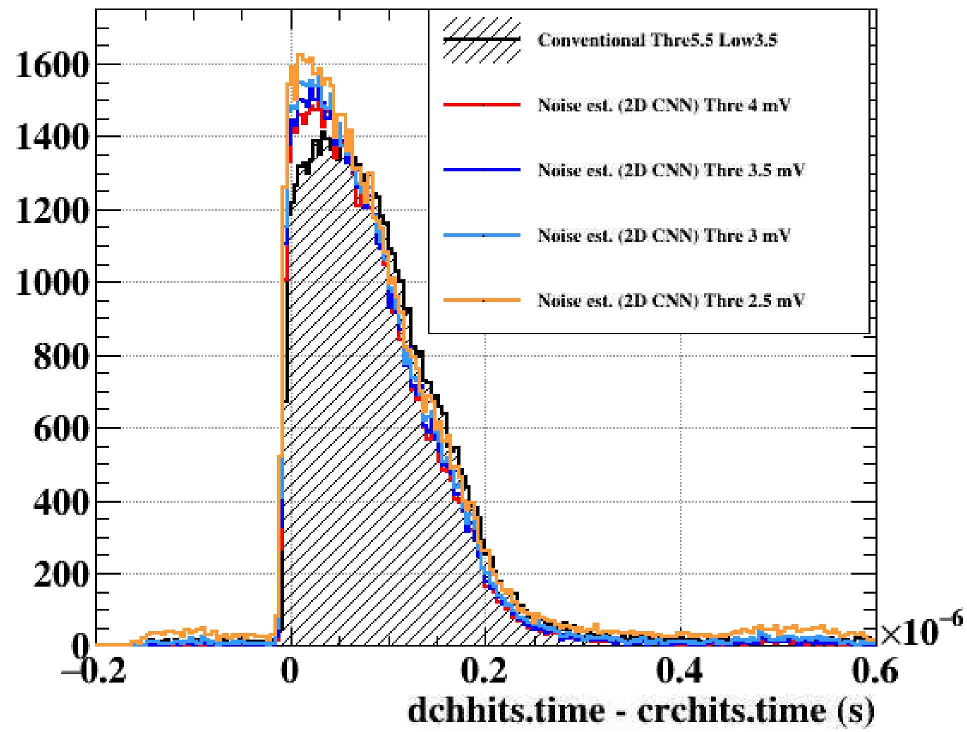
channels


n channels
$\otimes$
independent kernel (k points)
for each channel
$n \times m$ kernels,
$n \times m \times k$ parameters in total
$\oplus$
sum over channels
$\oplus$
m channels
to the next layer

Signal estimation



Noise estimation



ONNX

- The best solution as of today, we concluded, is using ONNX.
- Open Neural Network Exchange (ONNX) is an open standard format for representing machine learning models.
  - Able to exchange the models built by different frameworks.

Supported by

Supported frameworks

Frameworks & Converters

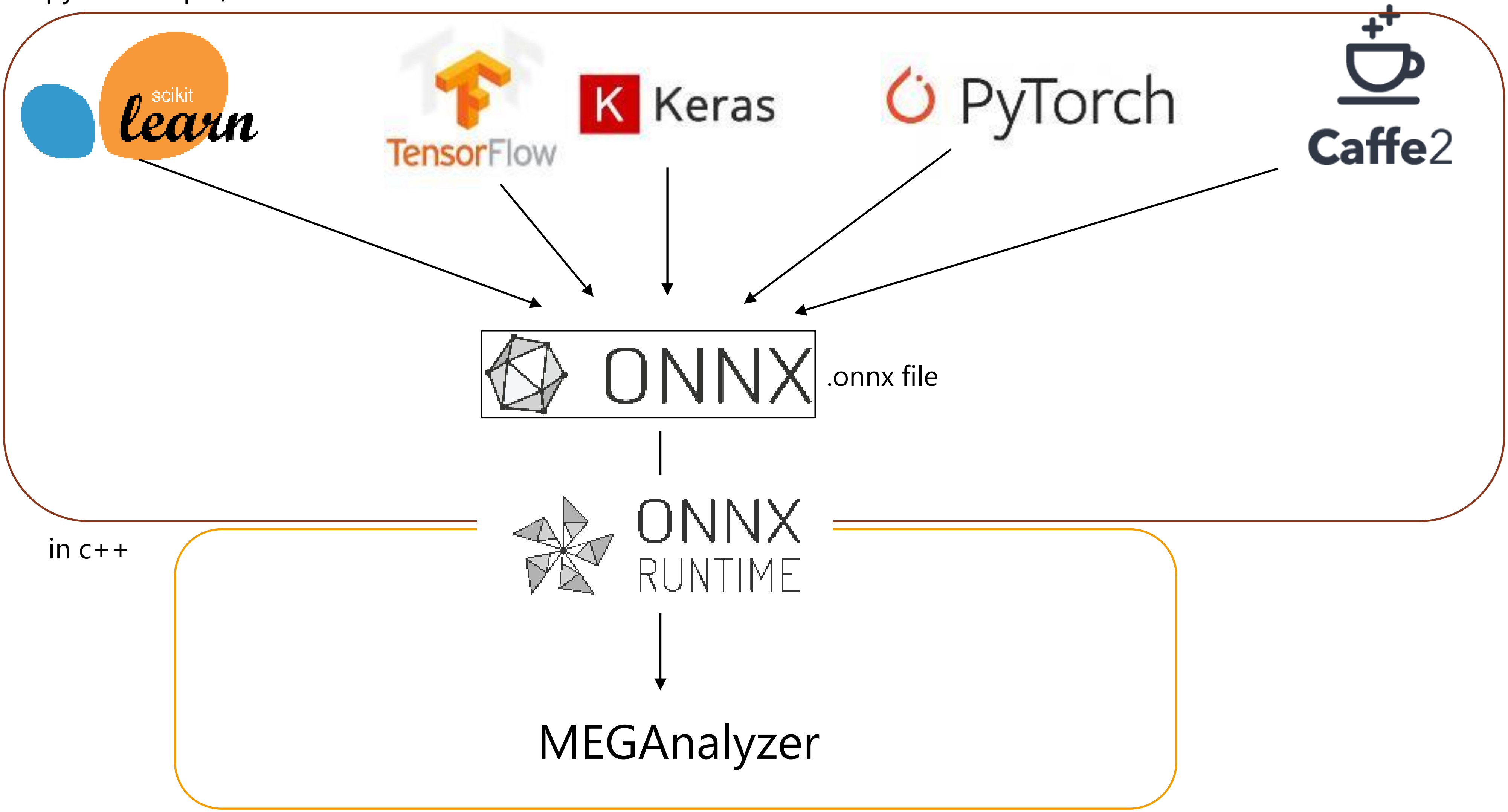
Use the frameworks you already know and love.

For example, following exchange is possible:

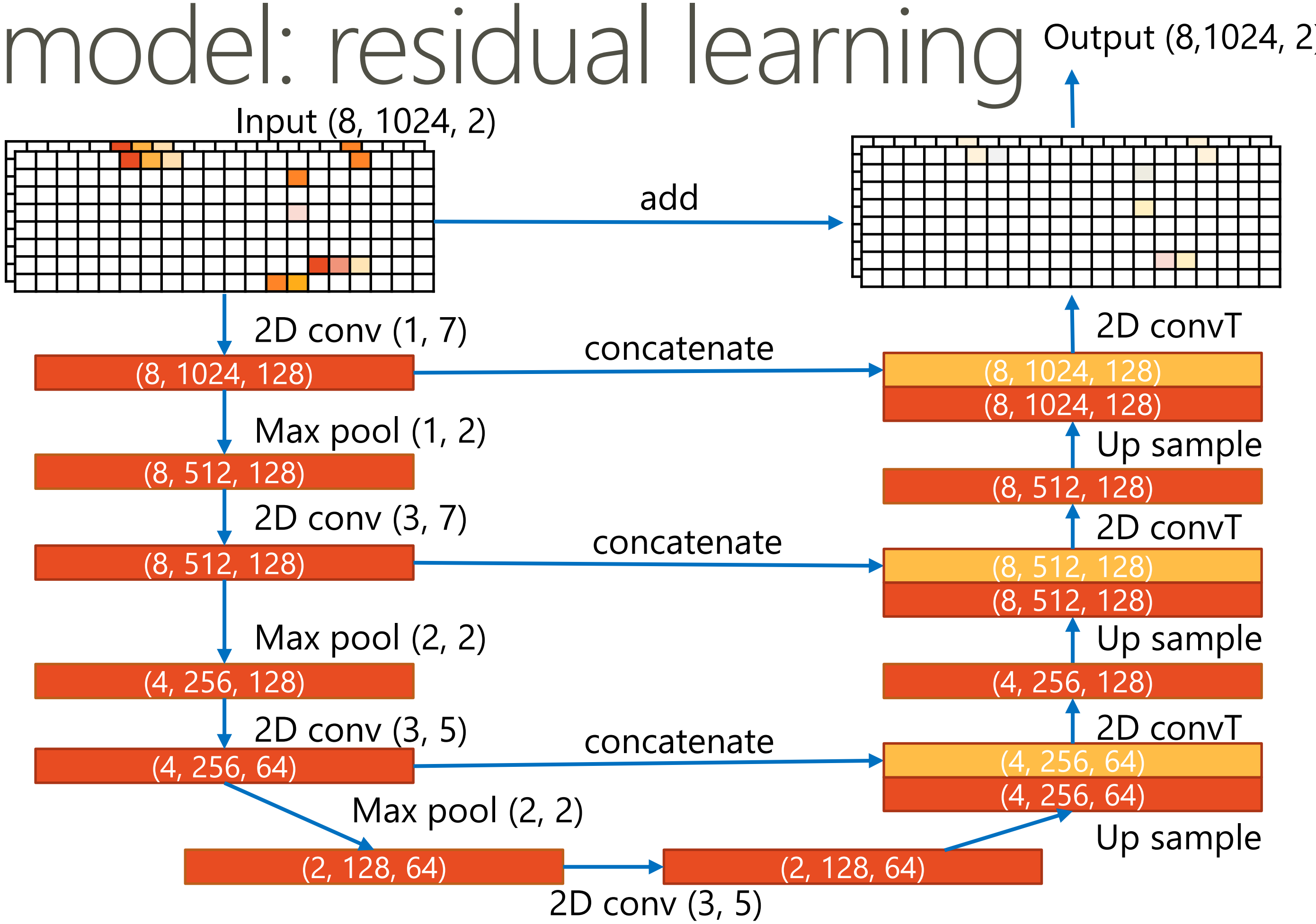
Building & training model in PyTorh → ONNX file → Inference in Tensorflow

- Note that not all the features may be supported.

In python scripts,



The model: residual learning



Adam, ReLU,
512 batch size,
150 epochs,
1.1M parameters

- 2D convolutional network
- 'UNet'-like structure with skip connections https://arxiv.org/abs/1505.04597 (image segmentation)
- 2-channel input with 2-end waveforms from 8 wires
- Use 'mean squared error (mse)' loss function.