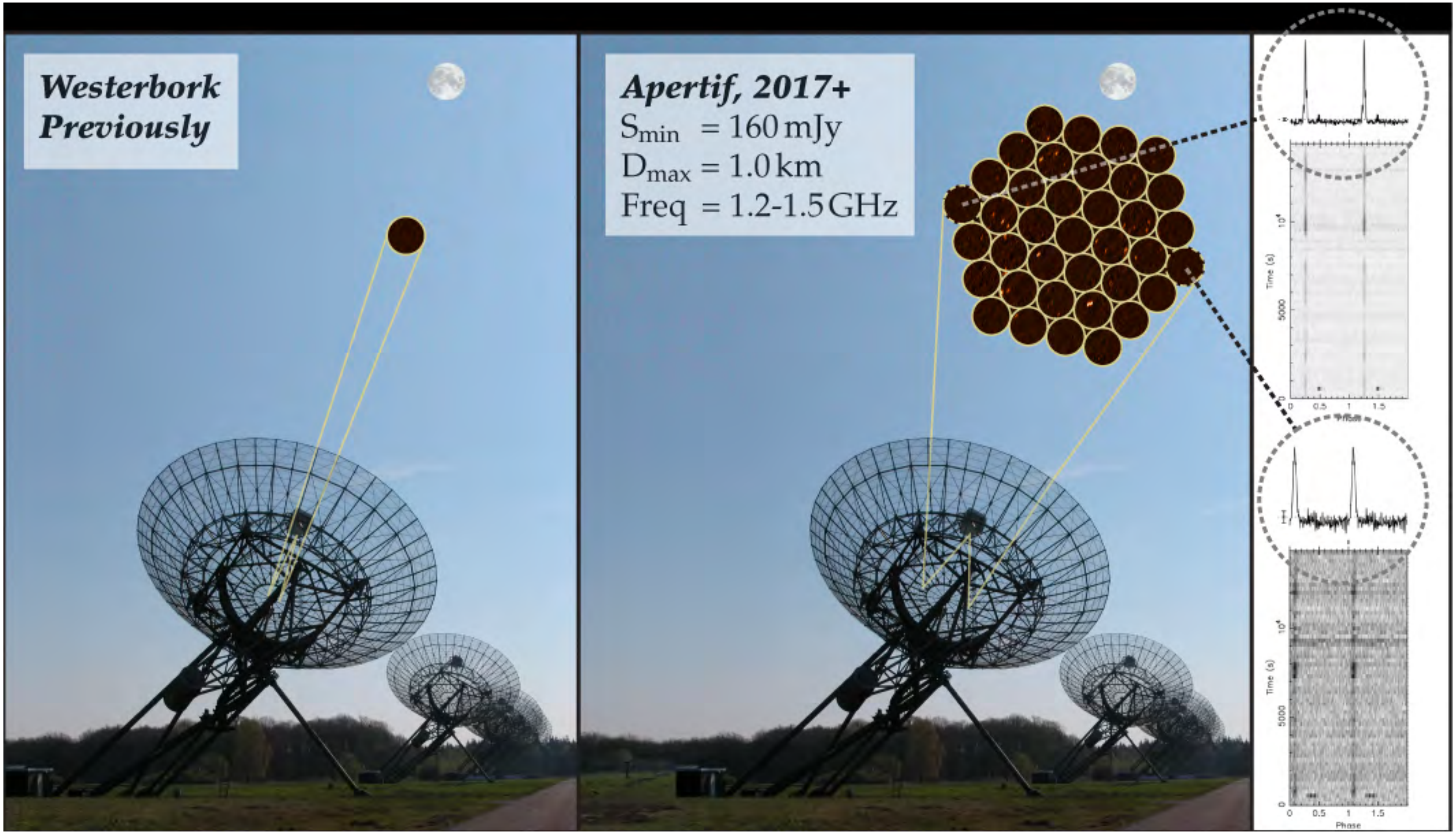
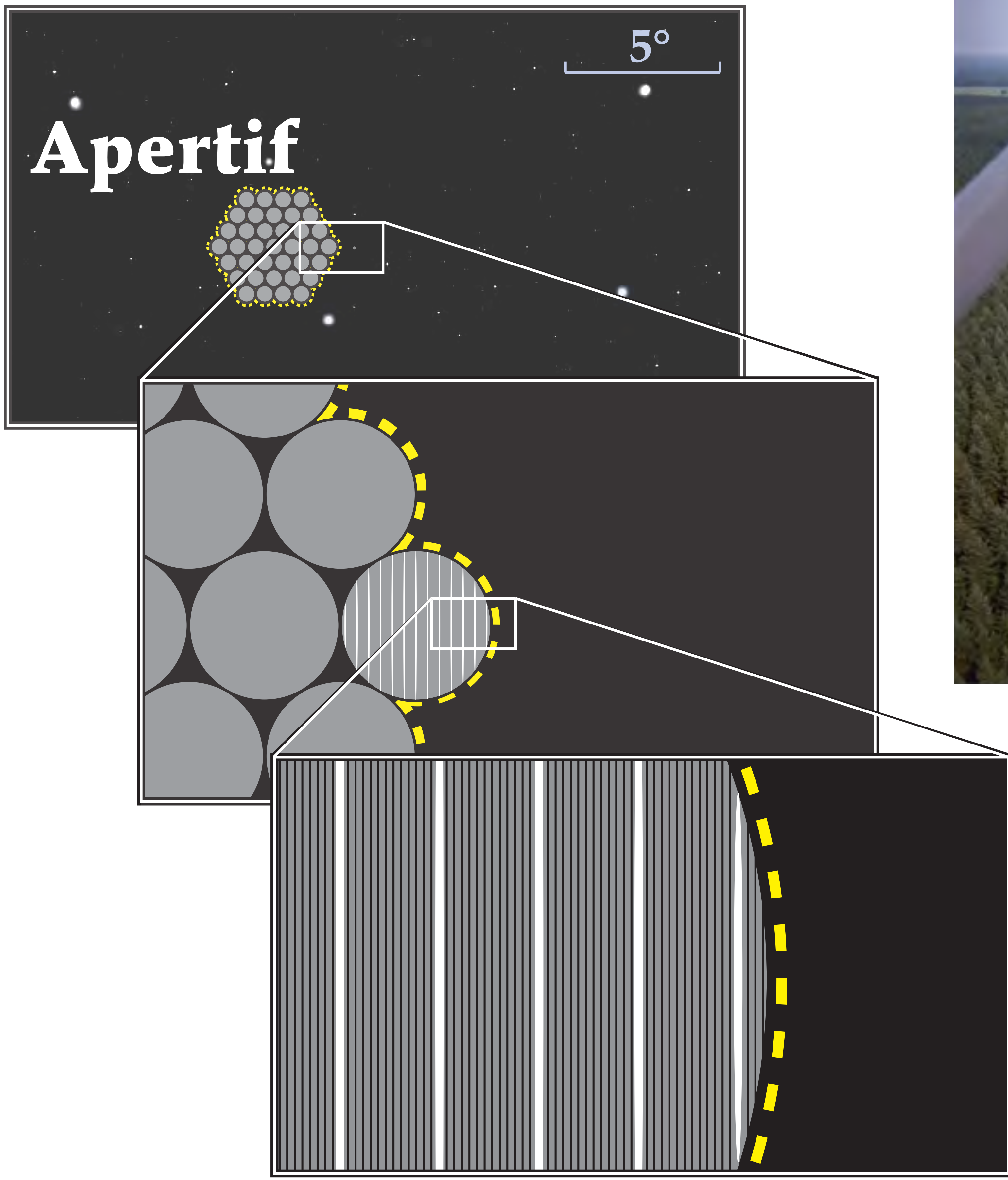
Applying deep learning to FRB classification

Liam Connor

16 February 2017
Swinburne FRB

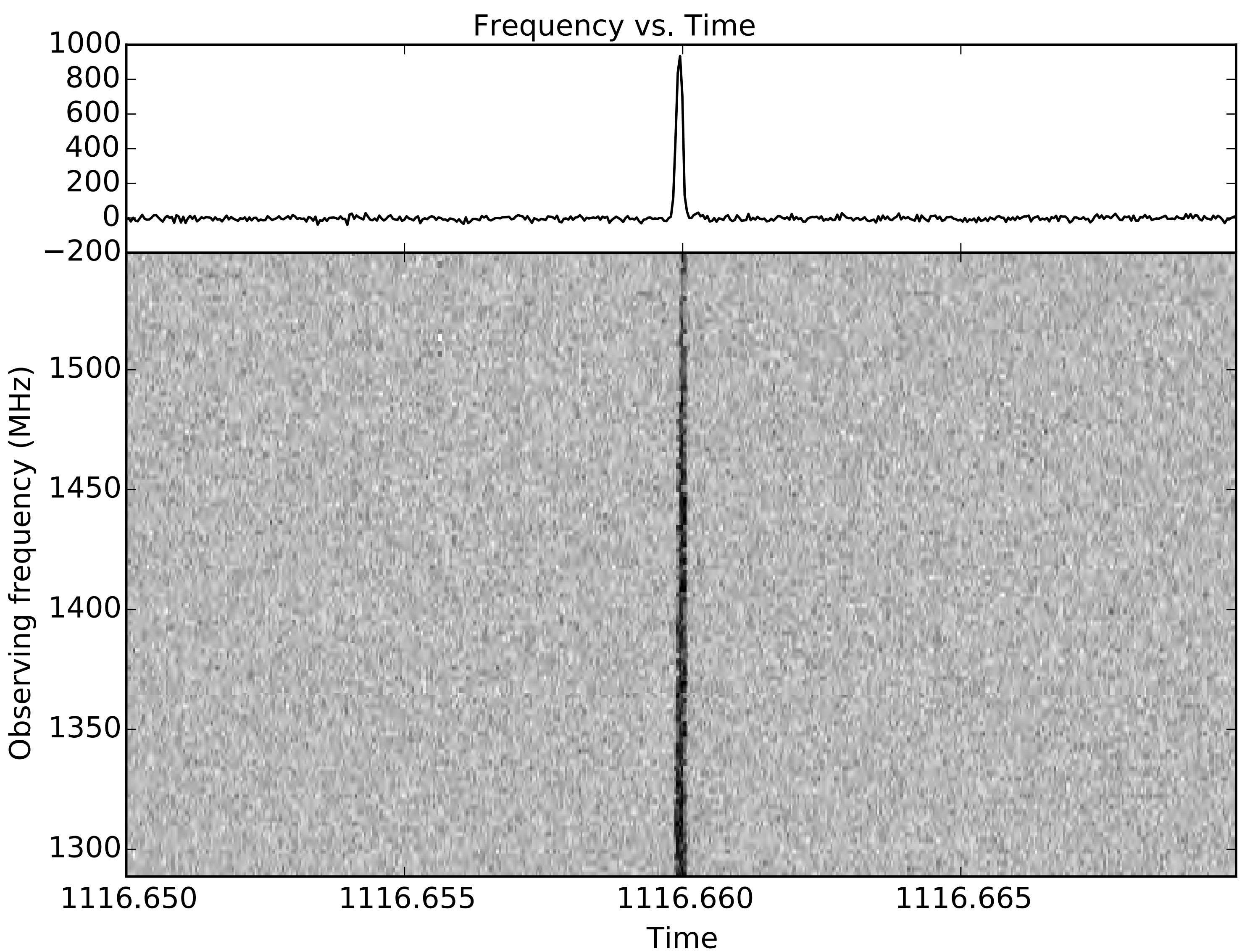


Apertif PAF increases FoV by ~30

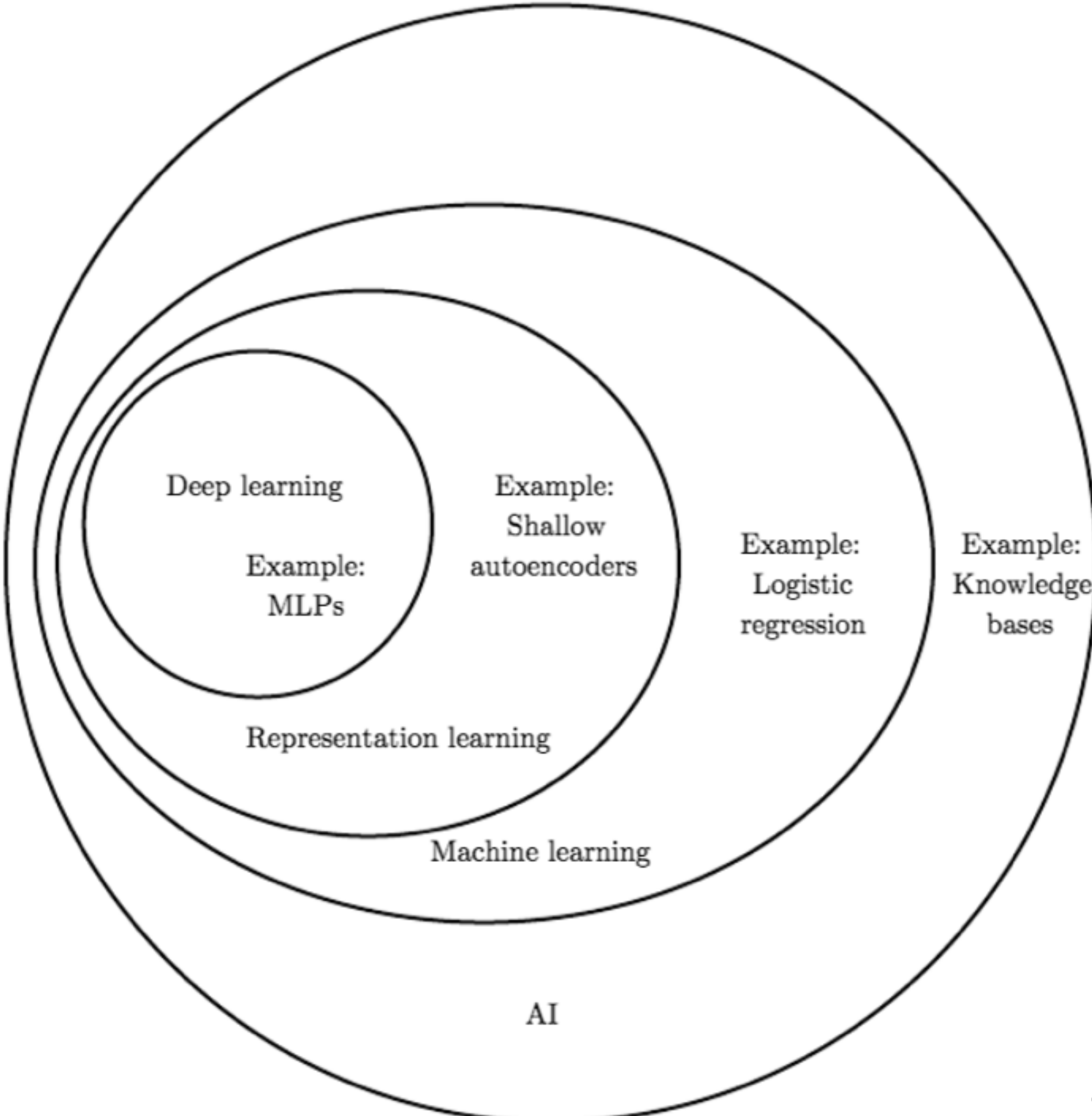

5°
Apertif


CIAT

Search pipeline tested



- Dozens of Crab GPs detected by pipeline
- Single pulses from B0329+54
- Clean band

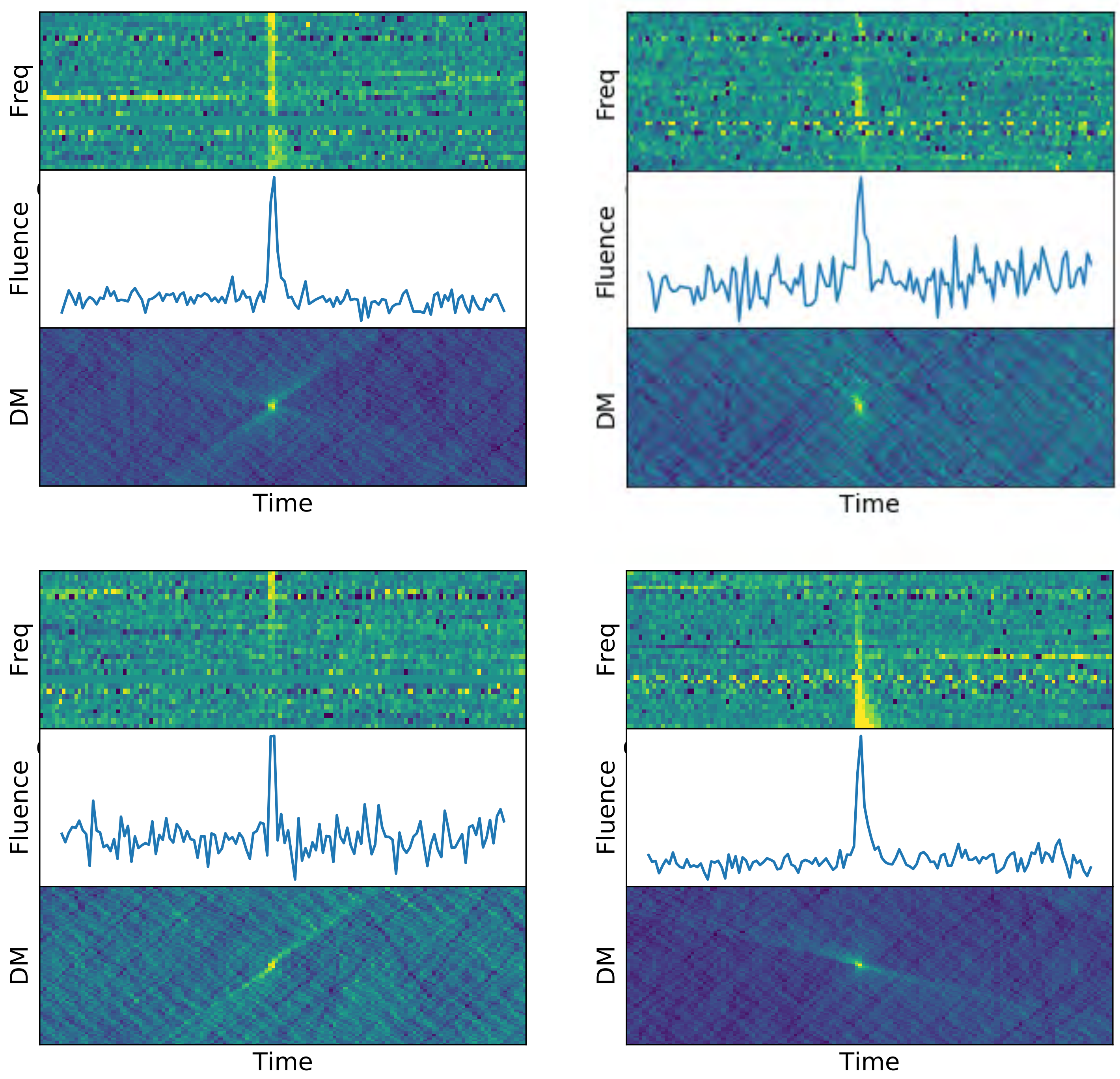


Goodfellow et al. 2016
http://www.deeplearningbook.org/contents/intro.html

TensorFlow

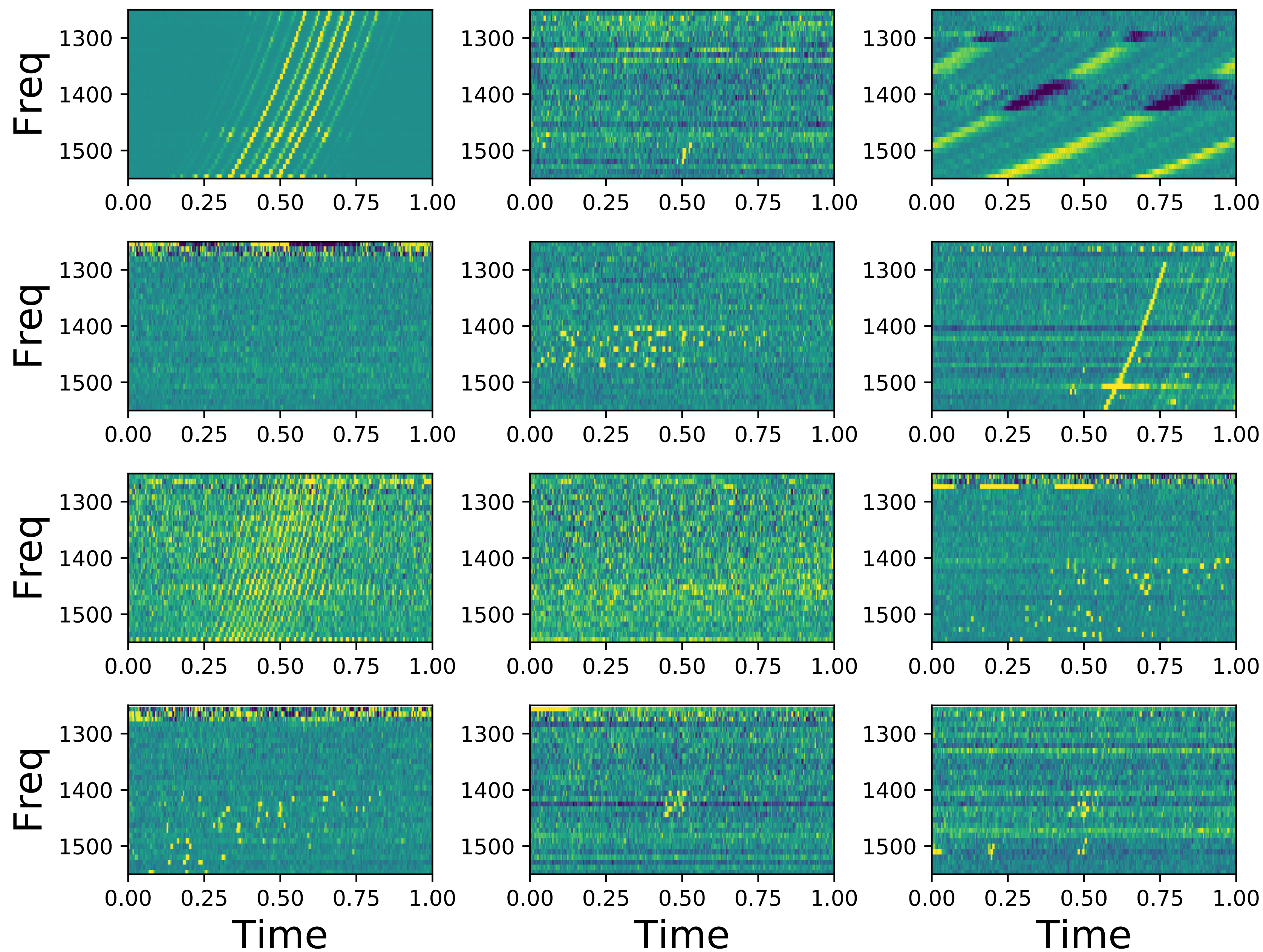
- Opensource software developed by Google Brain for internal use
- Builds / runs computational graph representing NNs
- Provides ML visualisation toolkit "TensorBoard"
- Easily run on CPUs, GPUs, or TPUs
- Makes use of pre-existing highly-optimised numerical libraries

Training set: true-positives

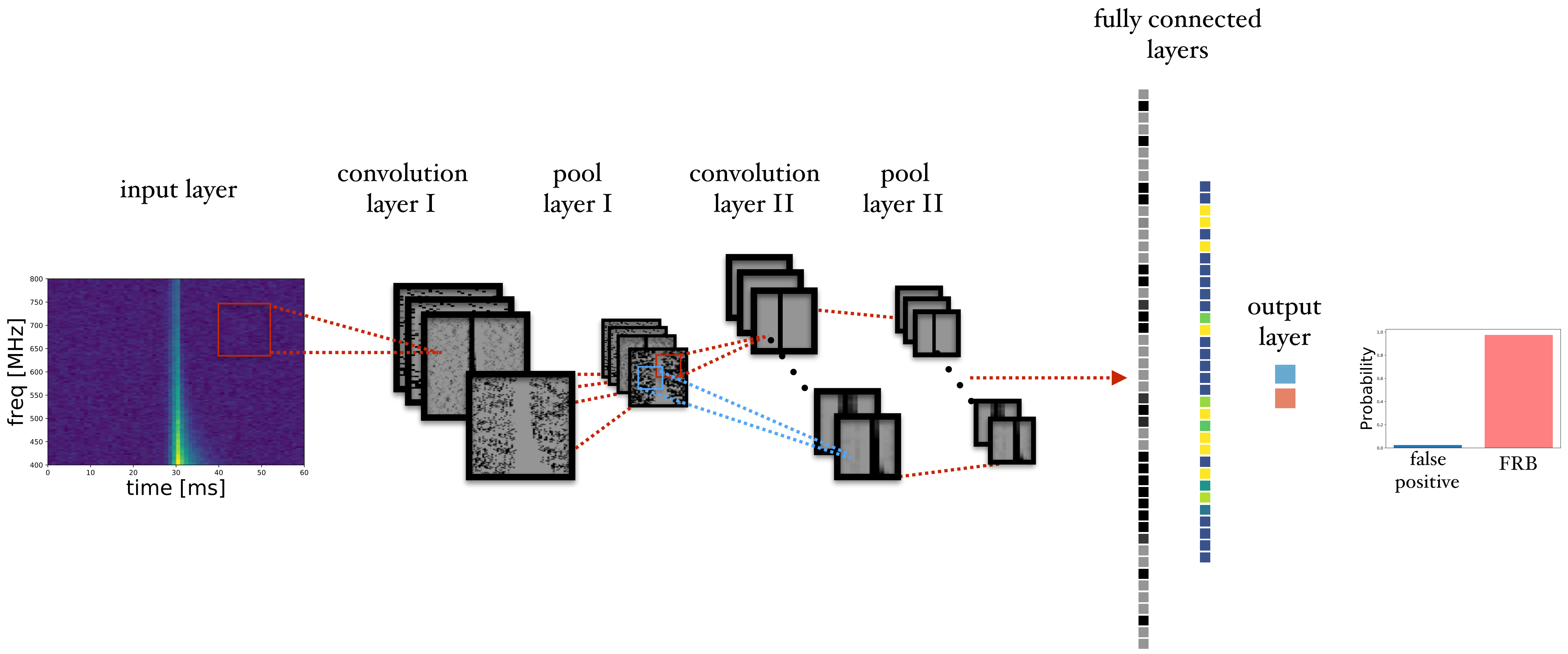


- Inject simulated bursts into real data
- Draw from broad distribution of width, scattering, scintillation, etc.

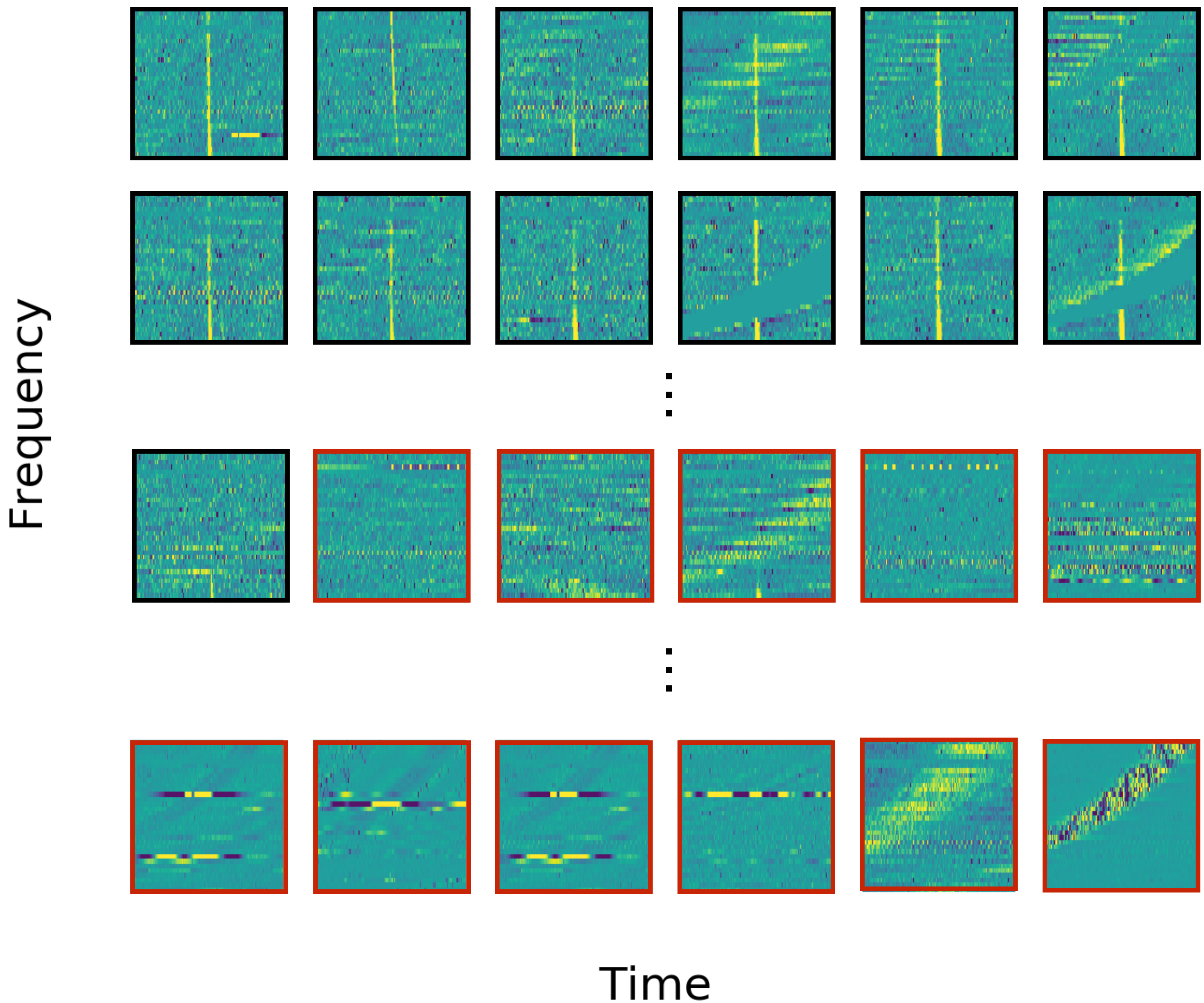
Training set: false-positives

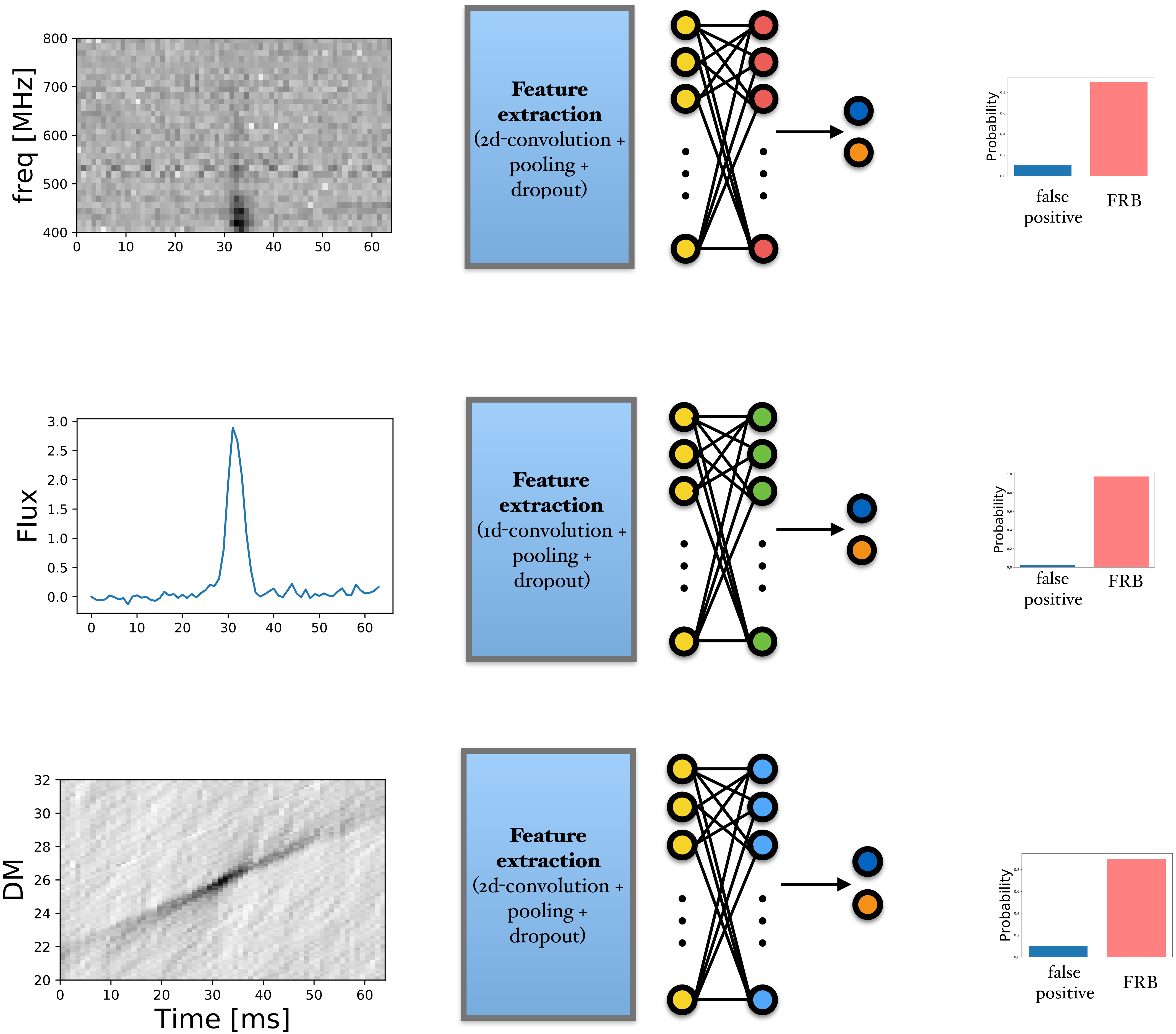


- Use real false positives
- RFI / dropped packets / thermal triggers too hard to simulate

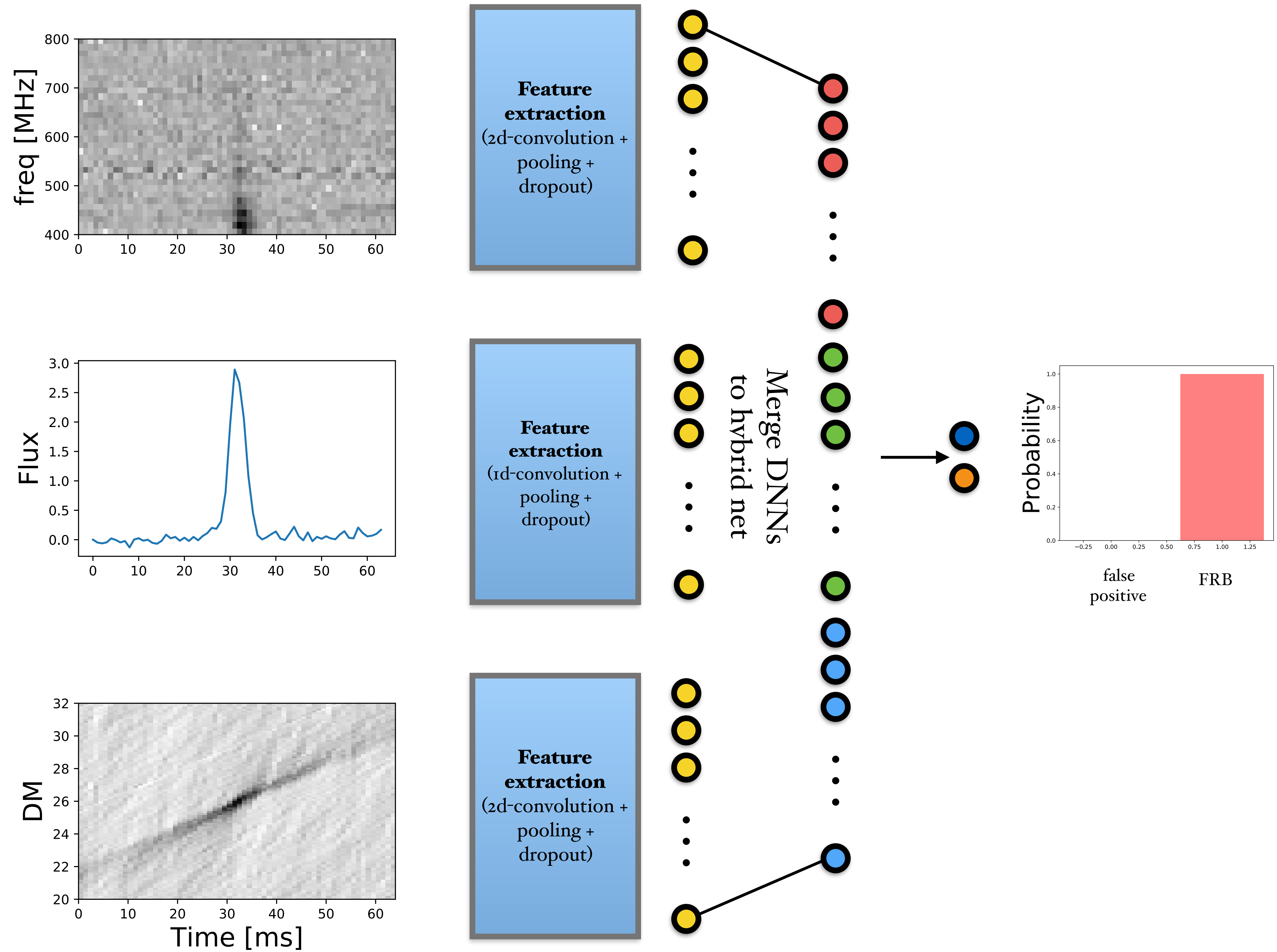


Dynamic spectrum CNN

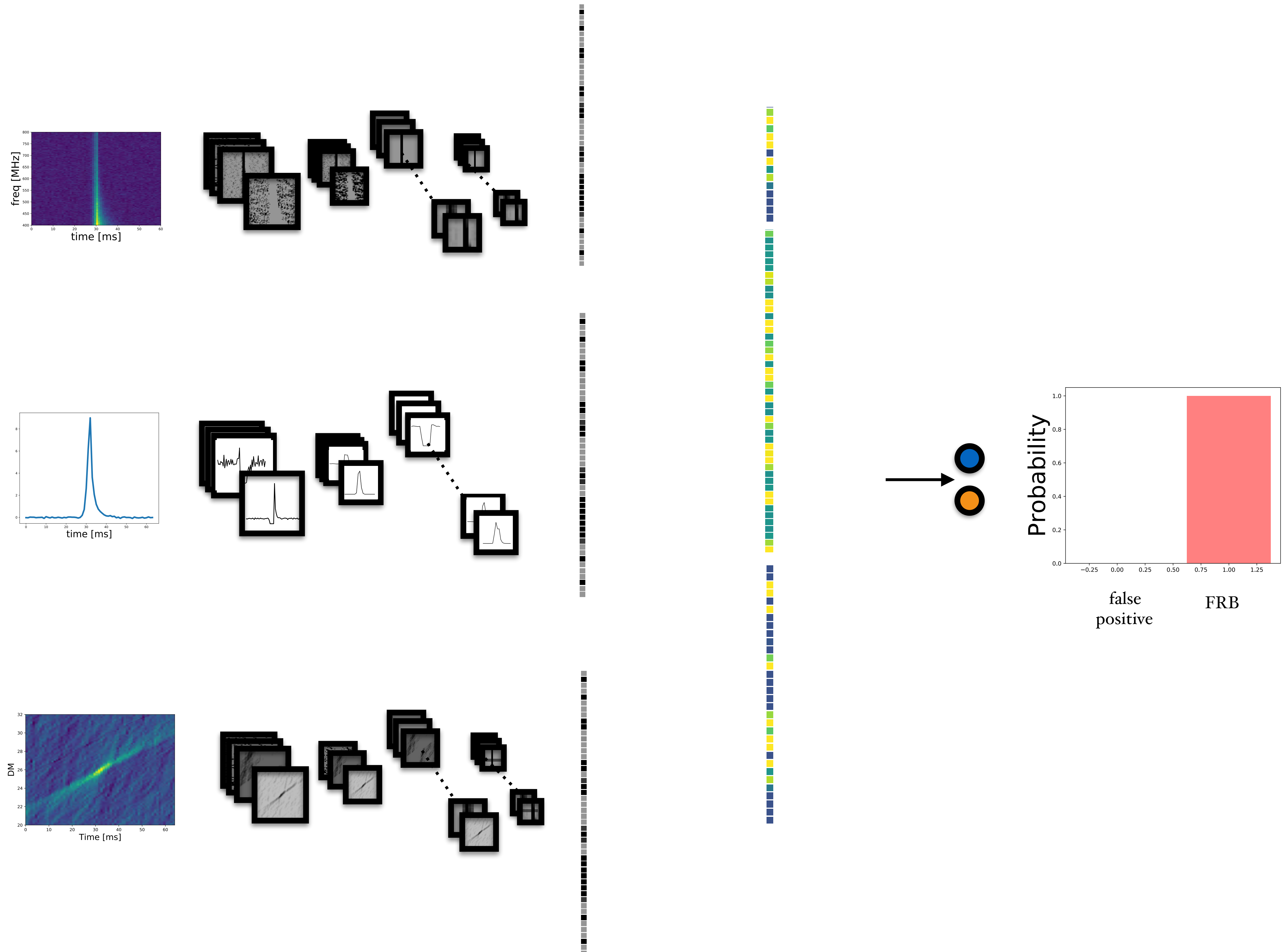

Frequency
Time



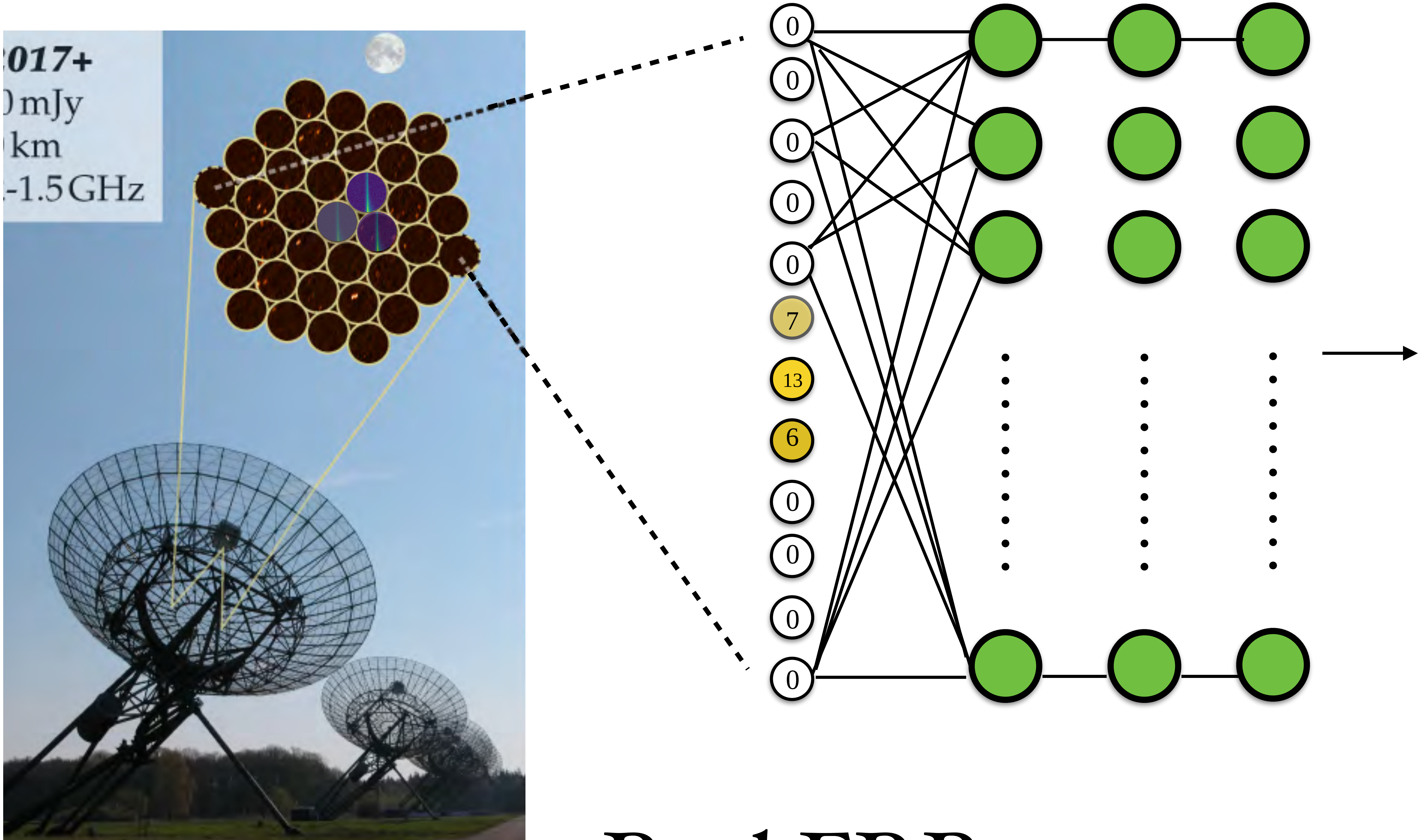
freq [MHz]
Flux
DM
Time [ms]
Feature extraction (2d-convolution + pooling + dropout)
Feature extraction (1d-convolution + pooling + dropout)
Feature extraction (2d-convolution + pooling + dropout)
Probability
false positive
FRB



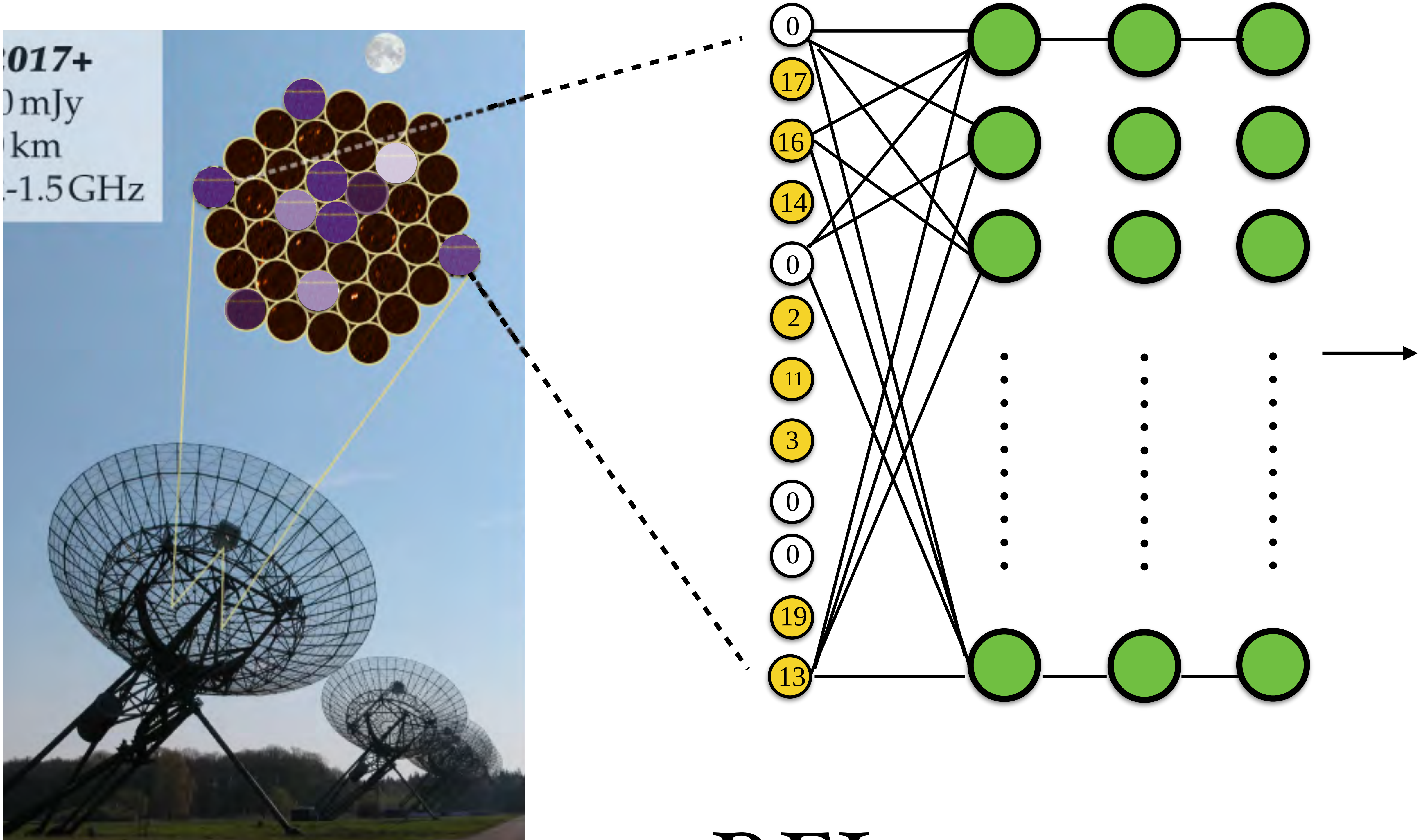
freq [MHz]
Flux
DM
Time [ms]
Feature extraction (2d-convolution + pooling + dropout)
Feature extraction (1d-convolution + pooling + dropout)
Feature extraction (2d-convolution + pooling + dropout)
Merge DNNs to hybrid net
Probability
false positive
FRB



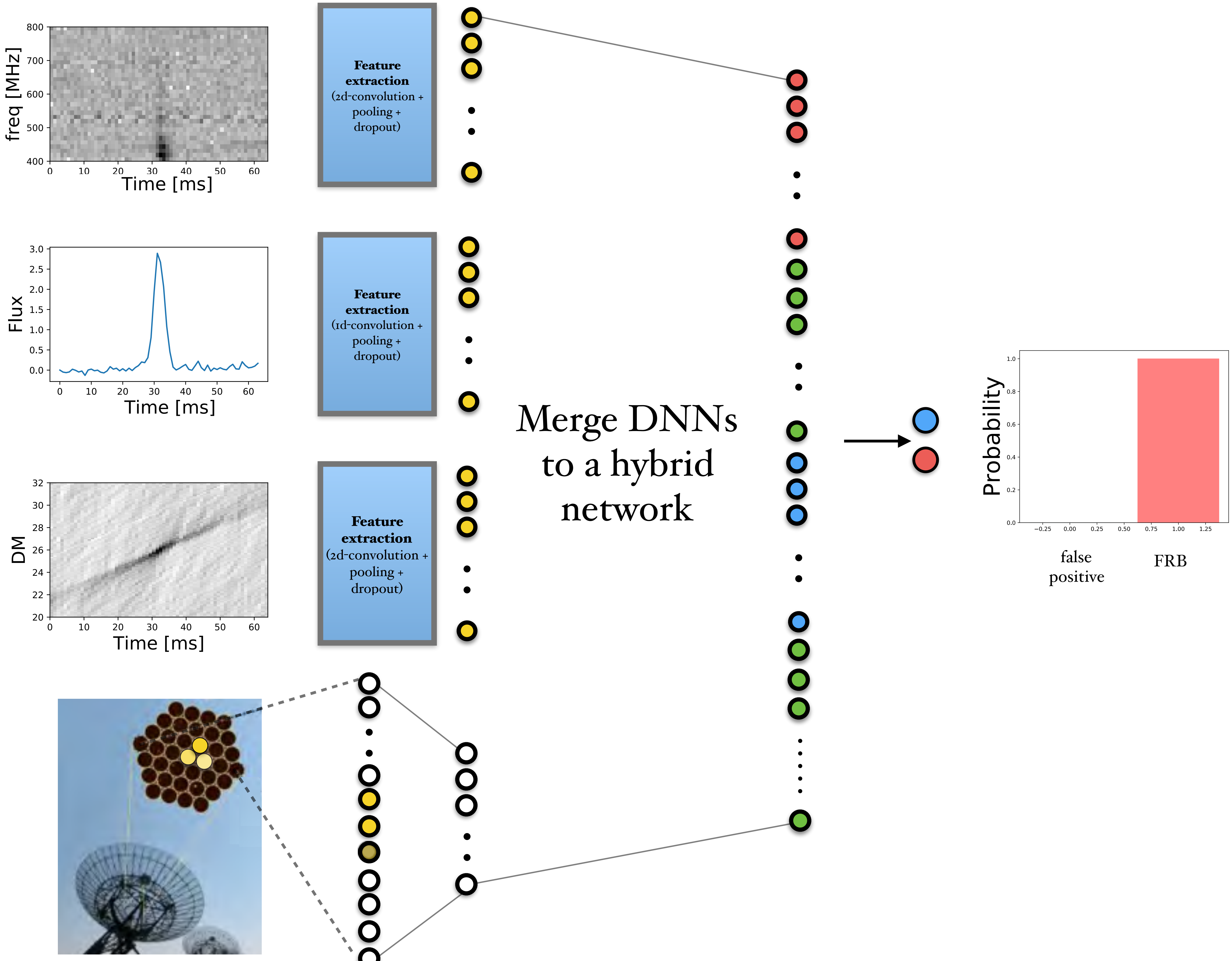
freq [MHz]
time [ms]
time [ms]
DM
Time [ms]
Probability
false positive
FRB



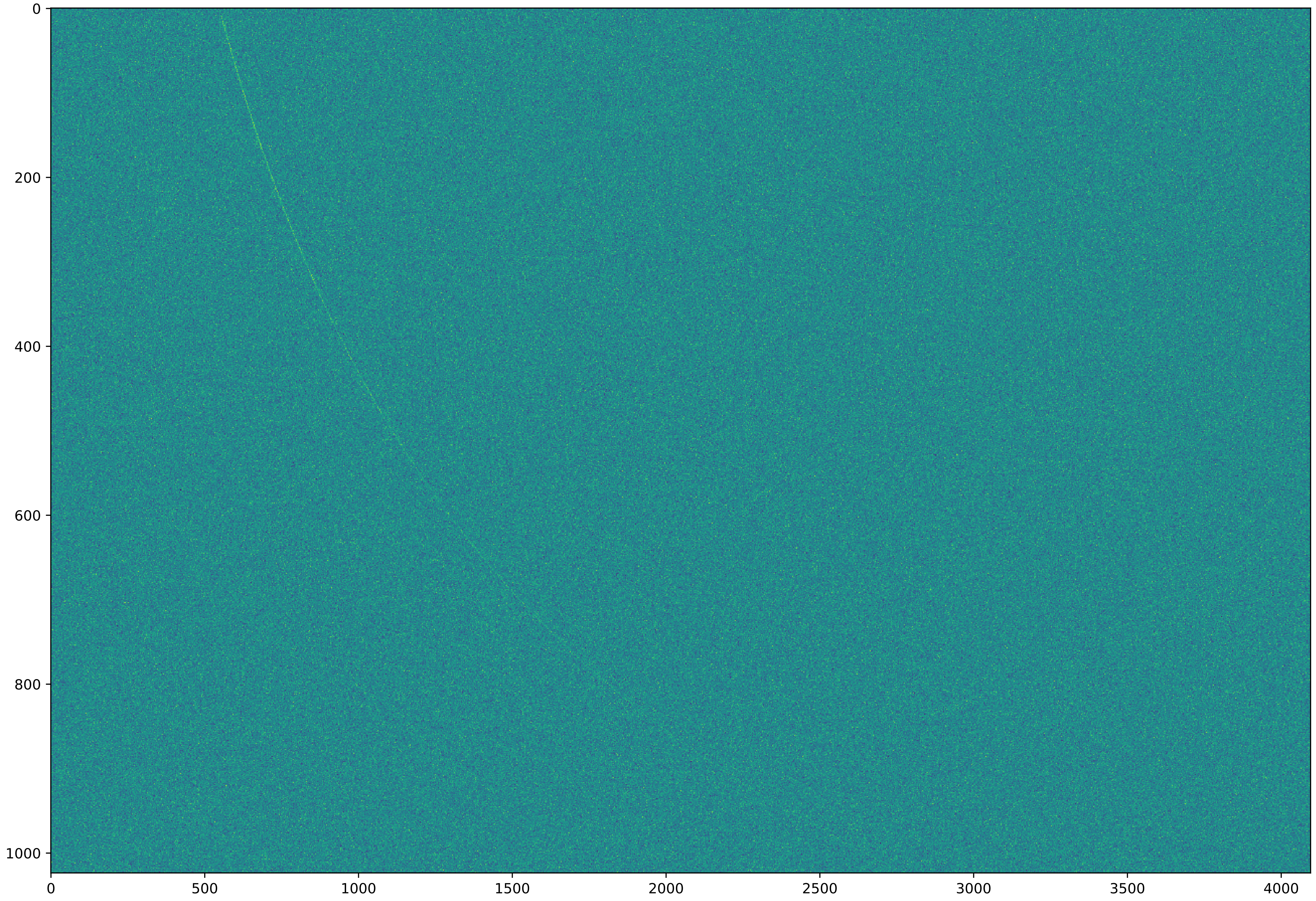
Real FRB



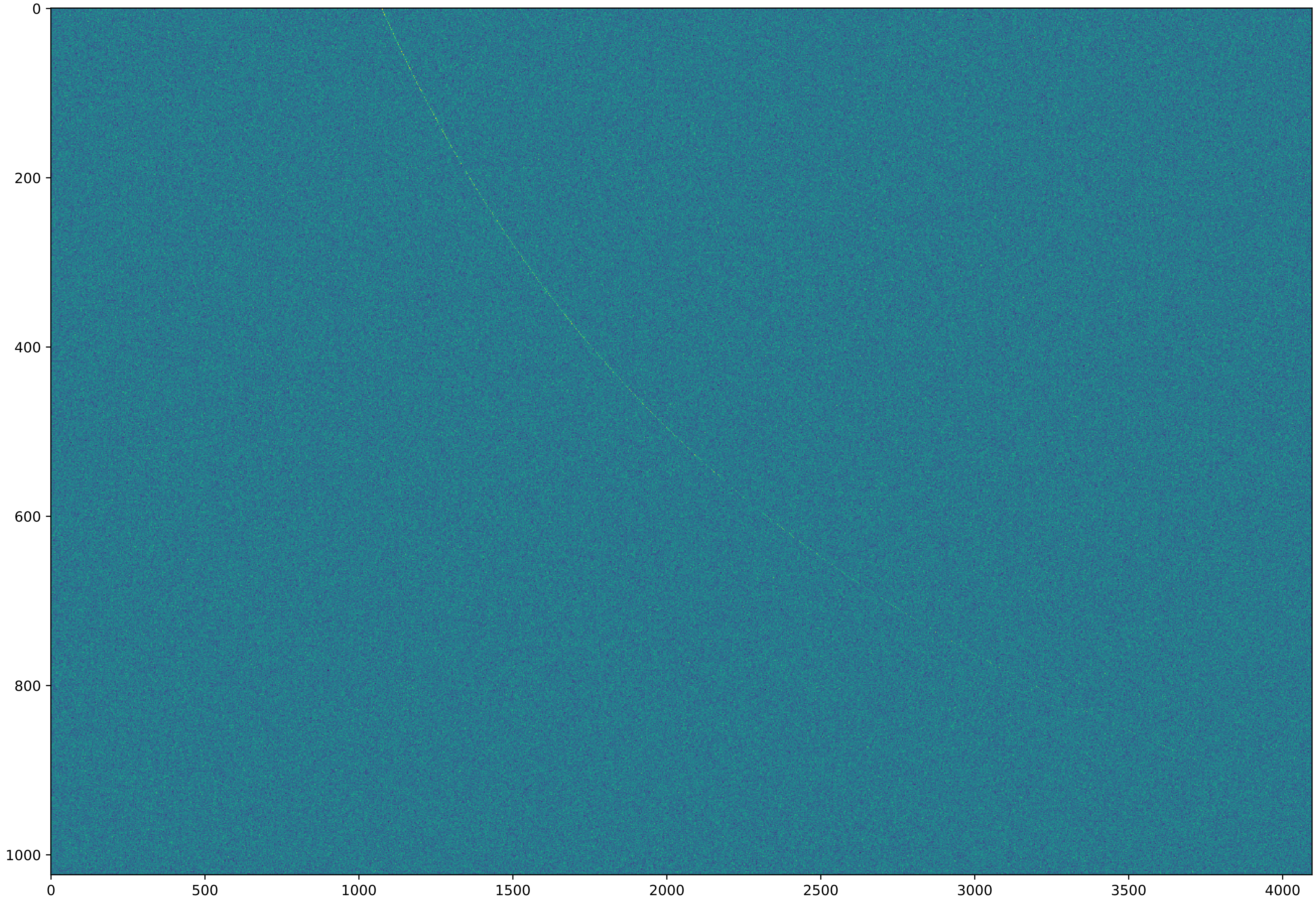
RFI


freq [MHz]
Time [ms]
Flux
Time [ms]
DM
Time [ms]
Feature extraction (2d-convolution + pooling + dropout)
Feature extraction (1d-convolution + pooling + dropout)
Feature extraction (2d-convolution + pooling + dropout)
Merge DNNs to a hybrid network
Probability
false positive
FRB

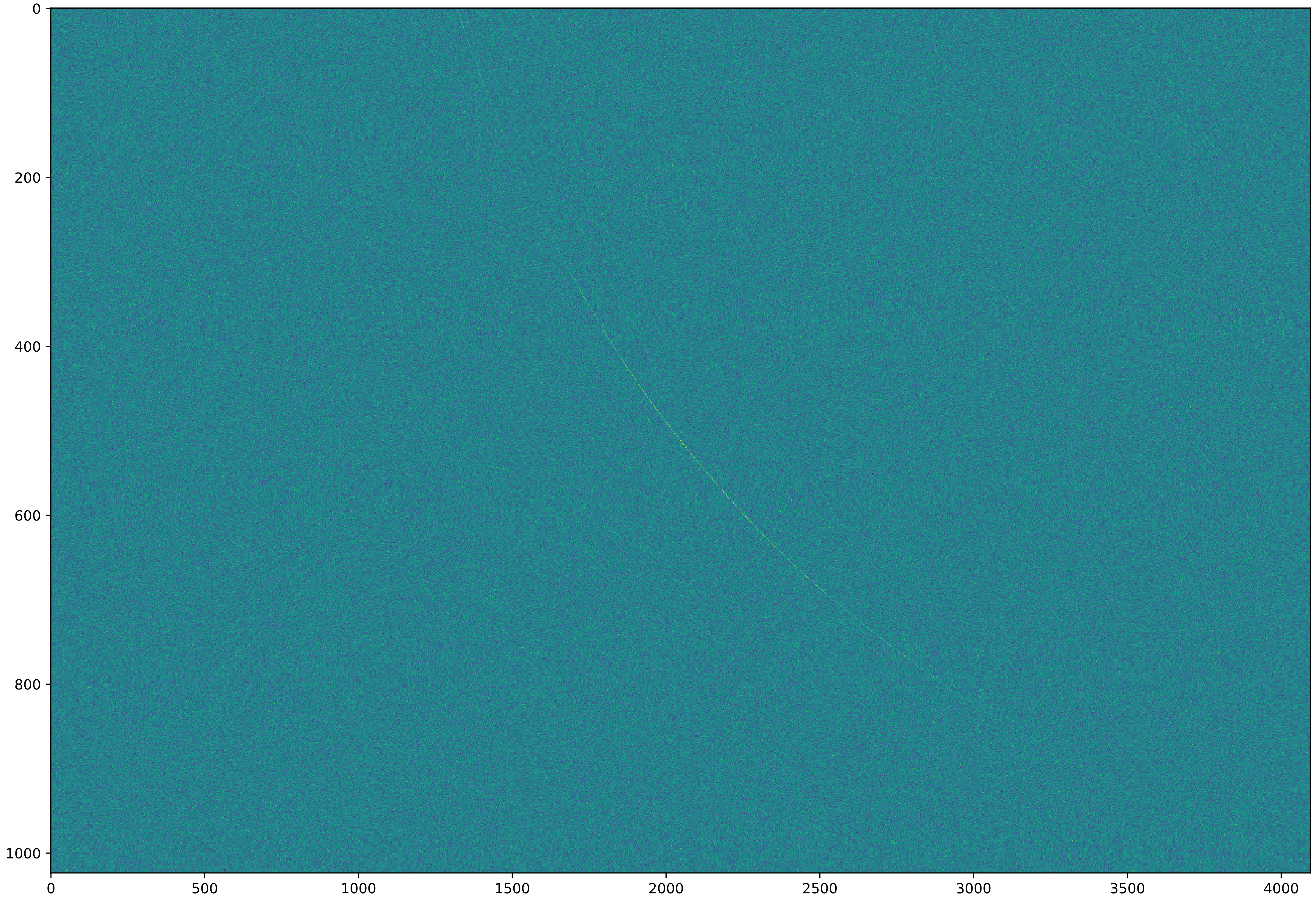
Real-time signal detection?



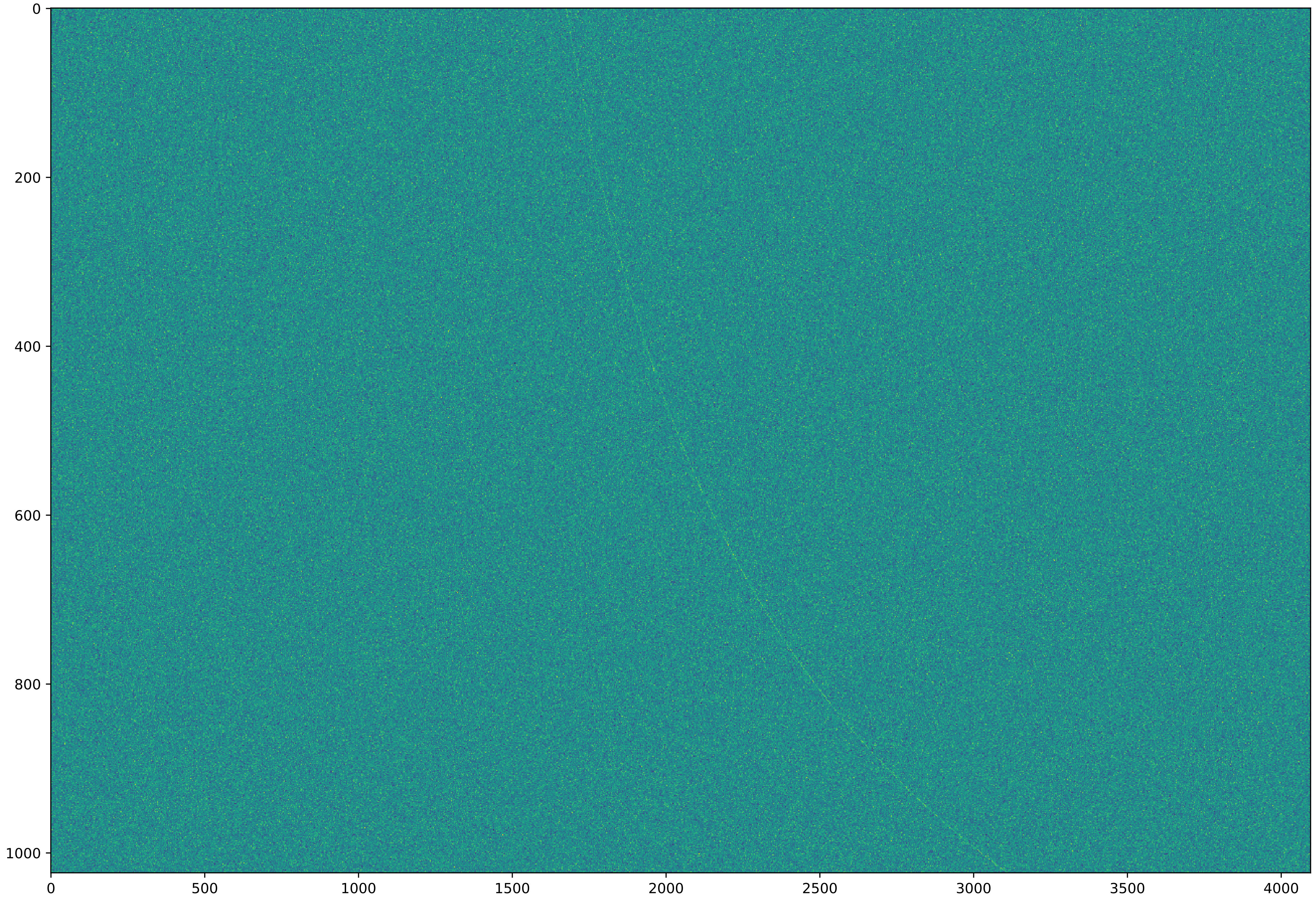
Real-time signal detection?



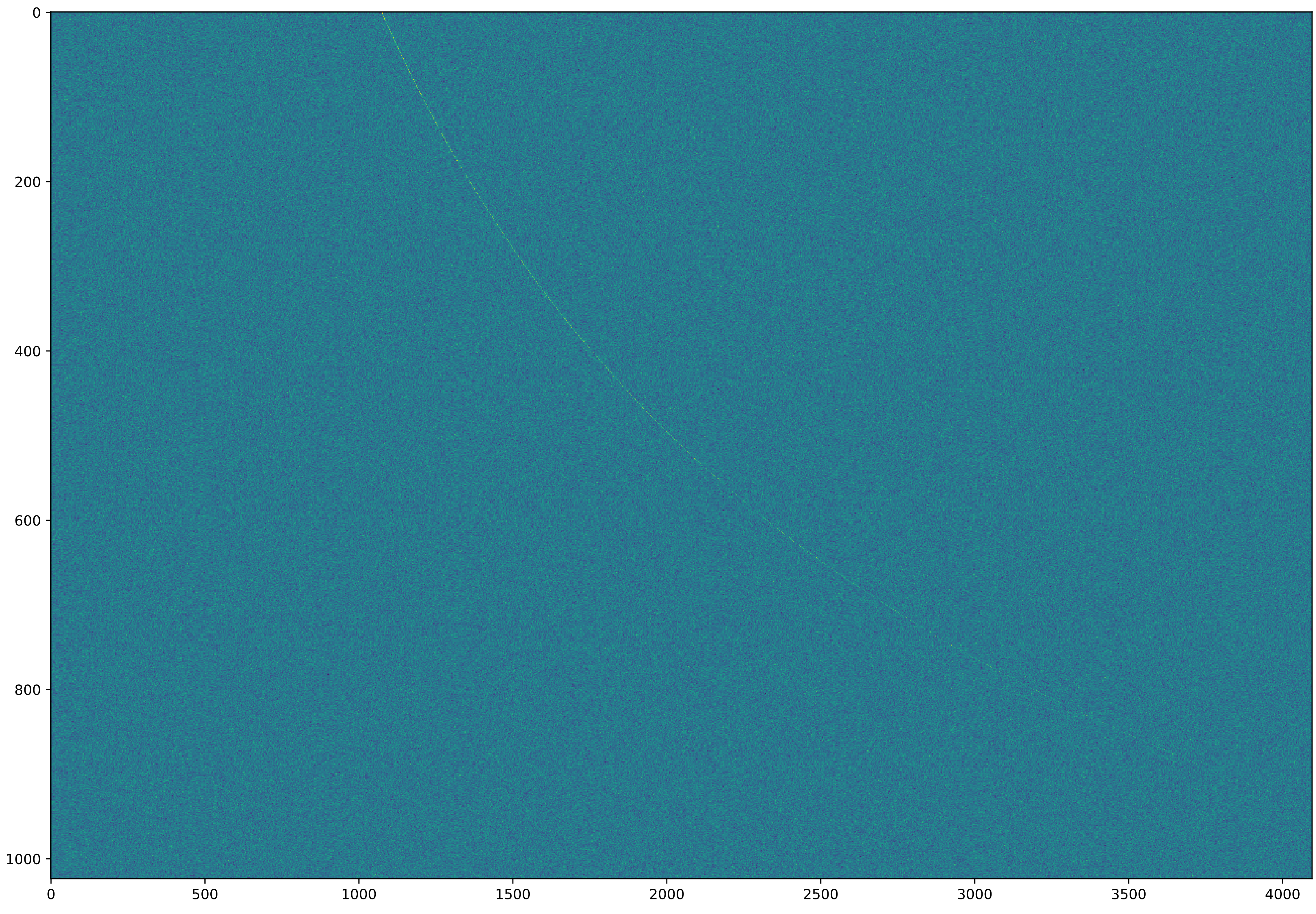
Real-time signal detection?



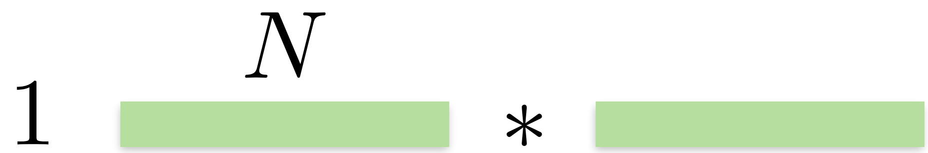
Real-time signal detection?



Real-time signal detection?

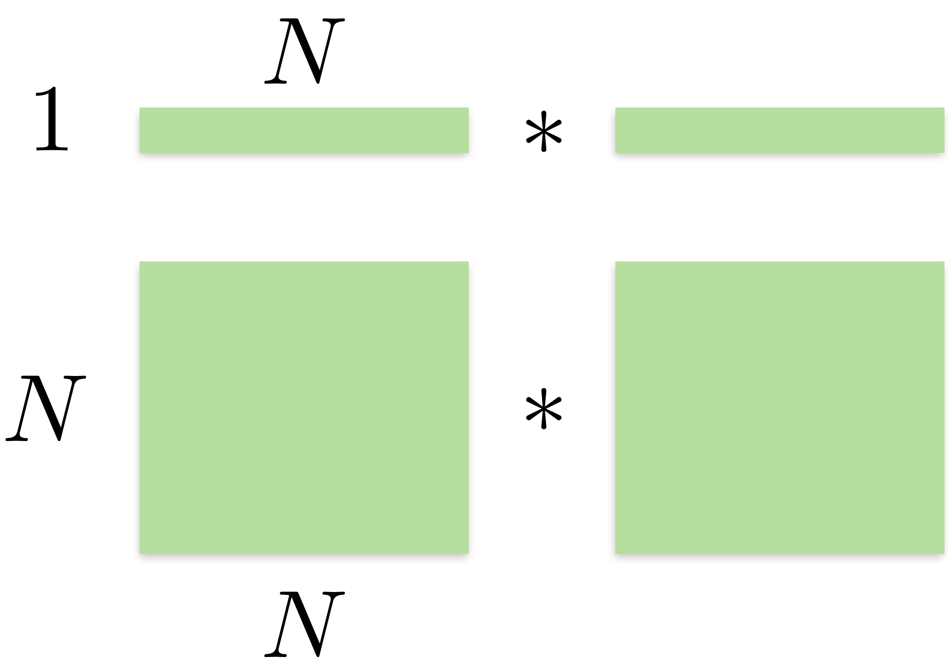


Convolution



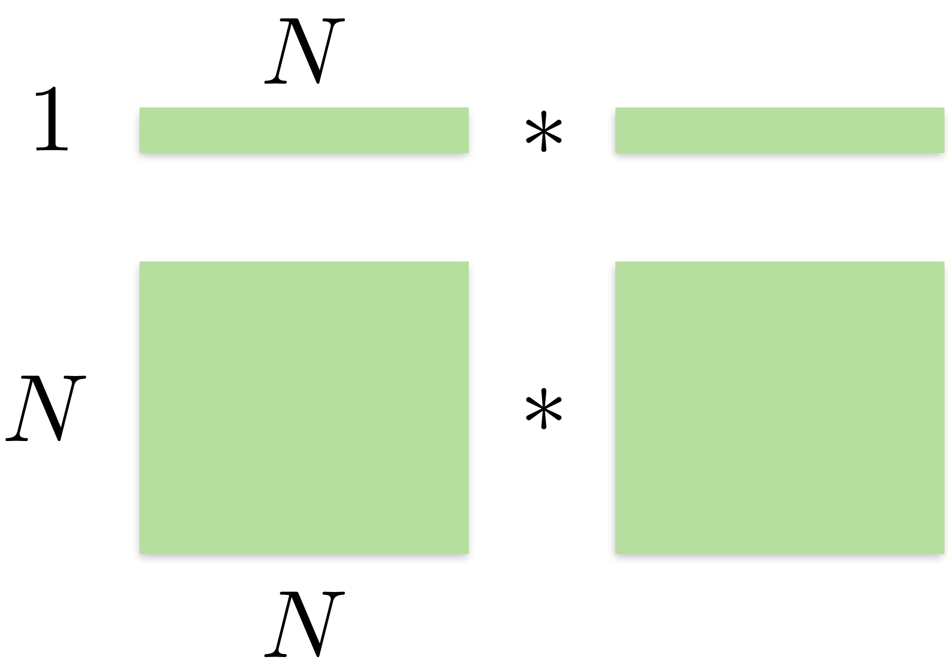
	brute-force	FFT
	$\mathcal{O}(N^2)$	$\mathcal{O}(Nlog_2N)$

Convolution

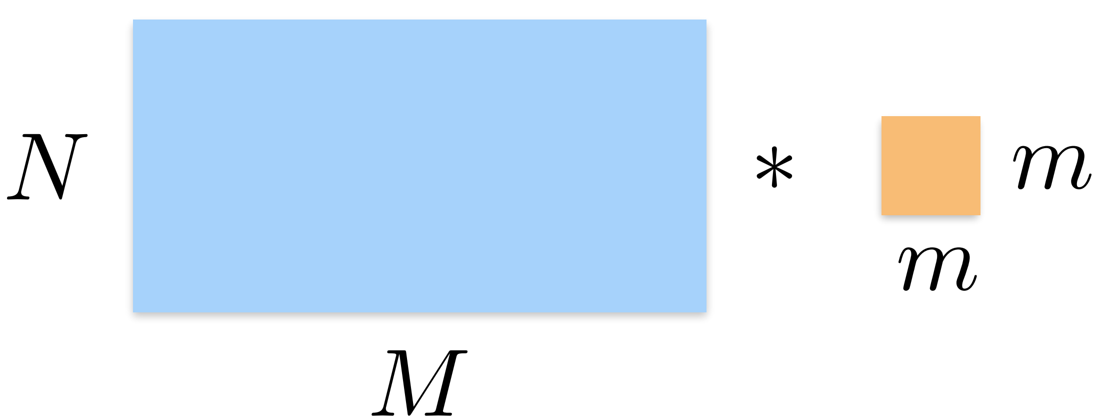


	brute-force	FFT
1D	$\mathcal{O}(N^2)$	$\mathcal{O}(N log_2 N)$
2D	$\mathcal{O}(N^4)$	$\mathcal{O}(N^2 log_2^2 N)$

Convolution



	brute-force	FFT
	$\mathcal{O}(N^2)$	$\mathcal{O}(N log_2 N)$
	$\mathcal{O}(N^4)$	$\mathcal{O}(N^2 log_2^2 N)$

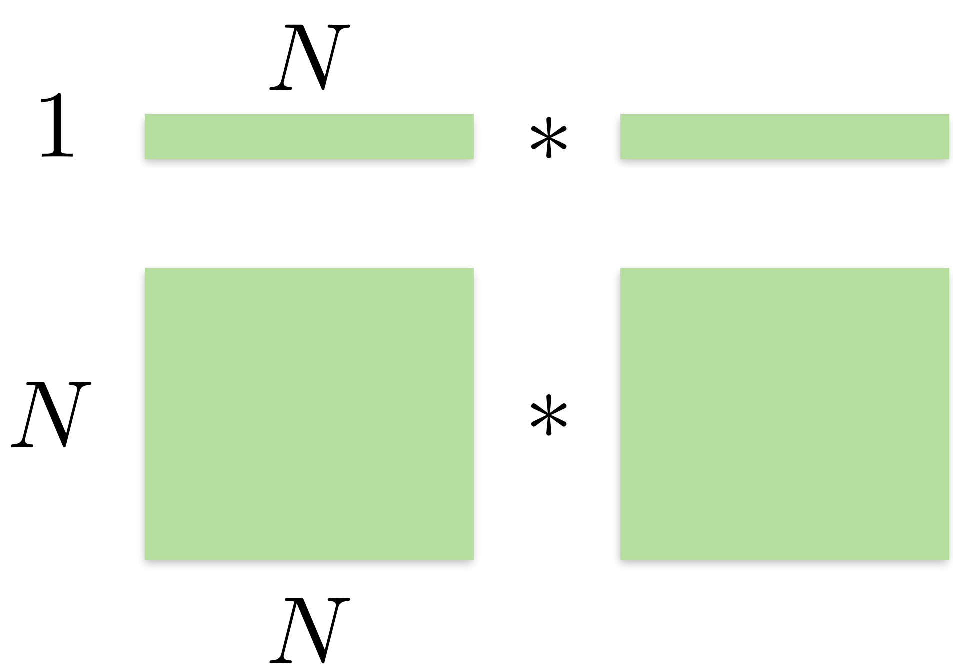


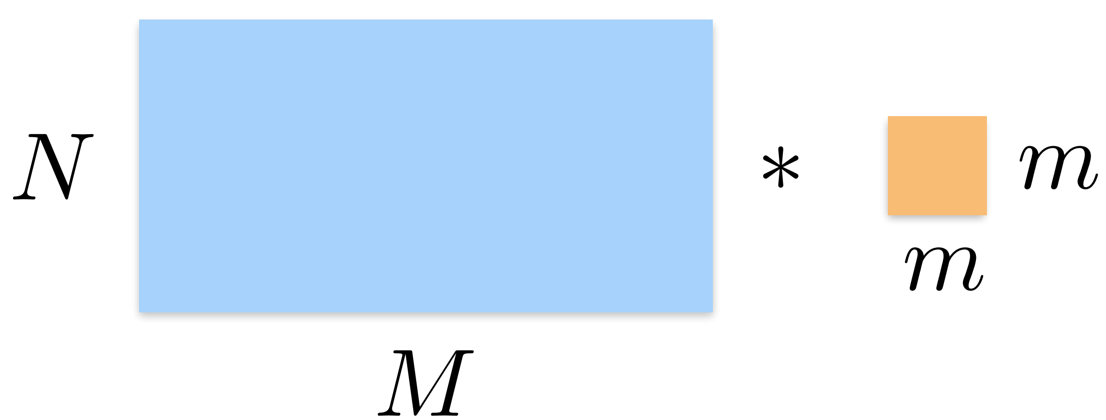
$\mathcal{O}(MN \times m^2)$ $\qquad$ $\mathcal{O}(MN \log_2 M)$

highly optimised for GPUs

overlap-add method

Convolution

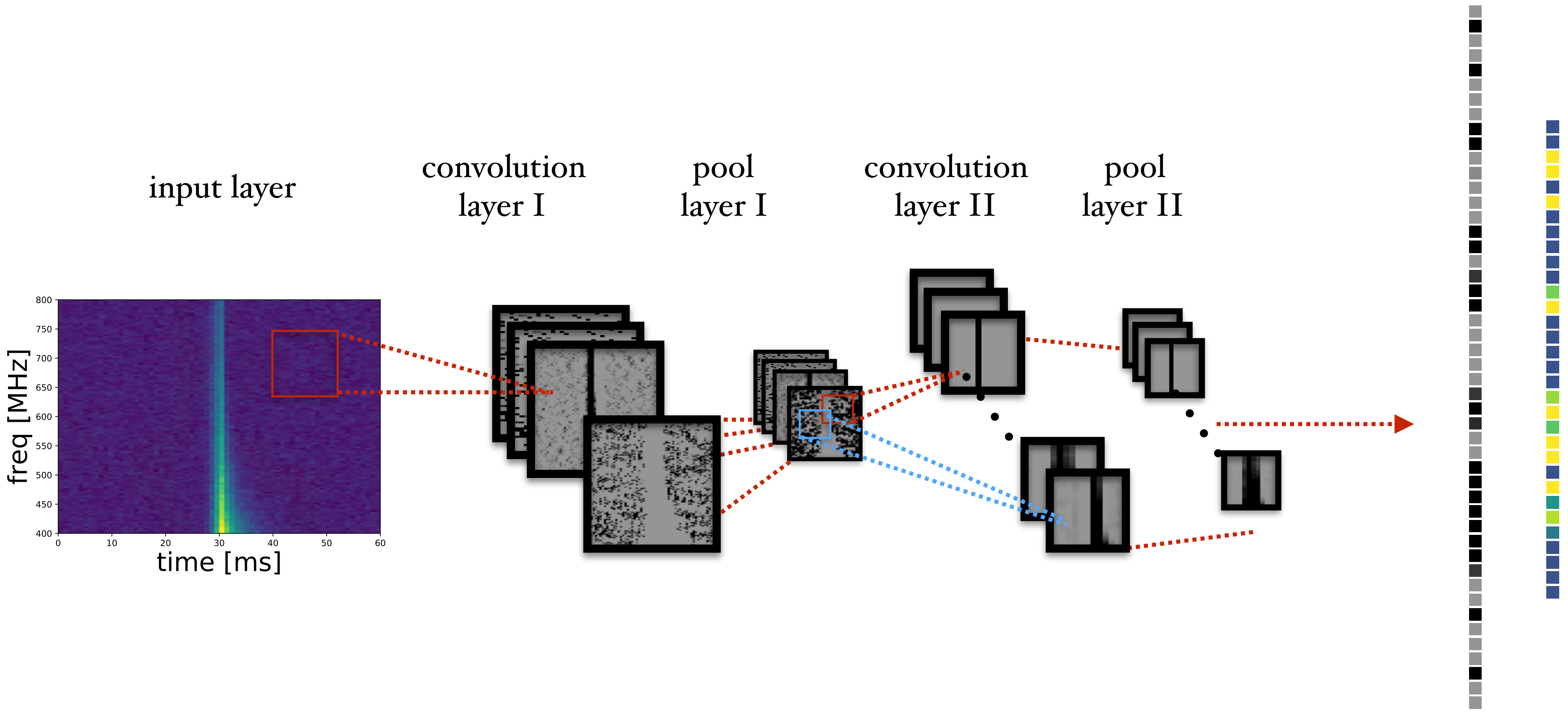




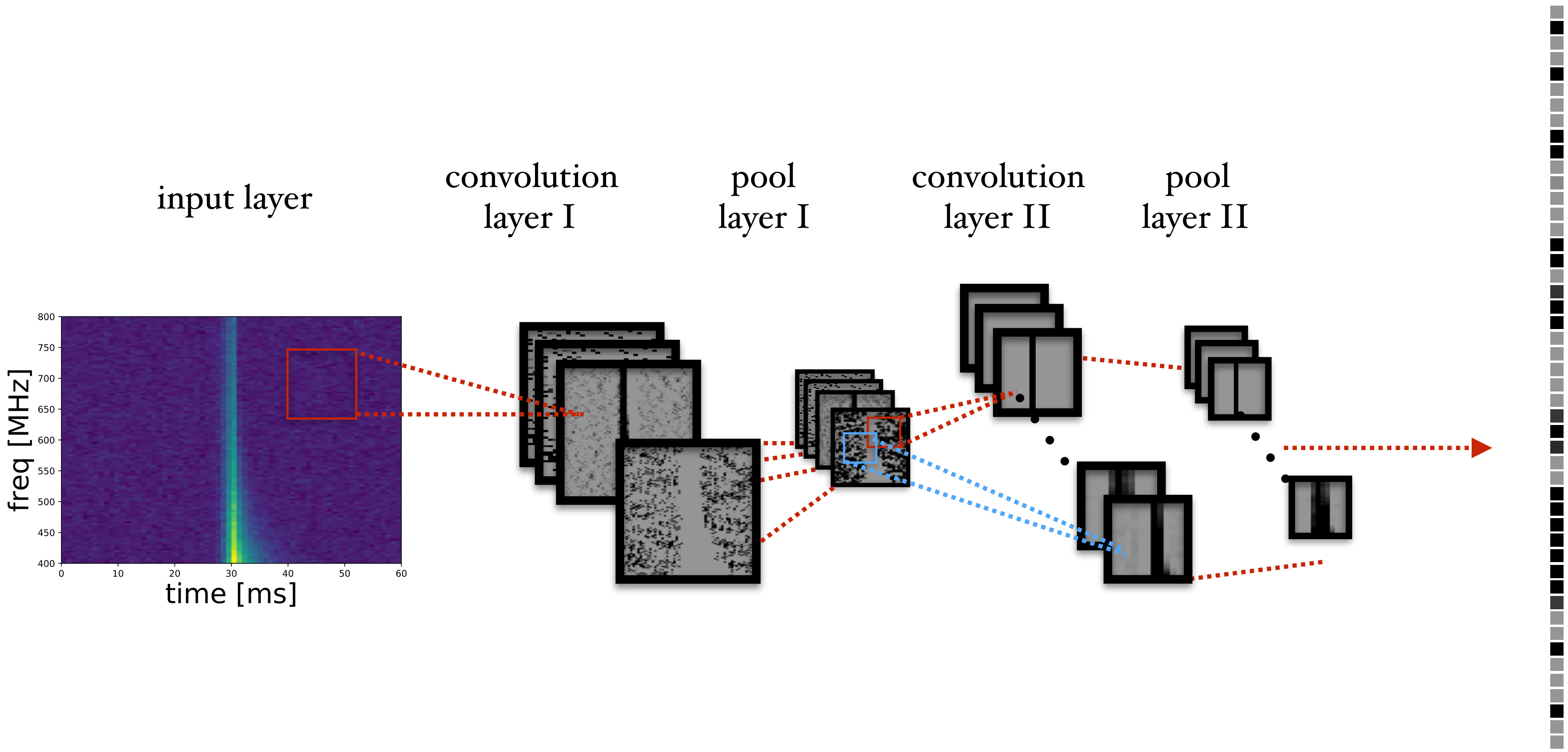
brute-force	FFT
$\mathcal{O}(N^2)$	$\mathcal{O}(N log_2 N)$
$\mathcal{O}(N^4)$	$\mathcal{O}(N^2 log_2^2 N)$
$\mathcal{O}(MN \times m^2)$	$\mathcal{O}(MN \log_2 M)$

highly optimised for GPUs

overlap-add method



Convolution: $\mathcal{O}(n_k N_t N_f \log_2(N_t))$



Convolution: $\mathcal{O}(n_k N_t N_f \log_2(N_t))$

DNN parameters

brute force

$$\mathcal{O}(N_t N_f N_{dm})$$

tree de-dispersion

$$\mathcal{O}(N_t N_f \log_2 N_f)$$

- Working code to train / apply a hierarchical DNN to FRB candidates
- High recall / accuracy can be attained with a few thousand labelled triggers
- Let me know if you'd like to try it!
- Real-time classification could be faster than traditional dedispersion, but not ideal for FRBs