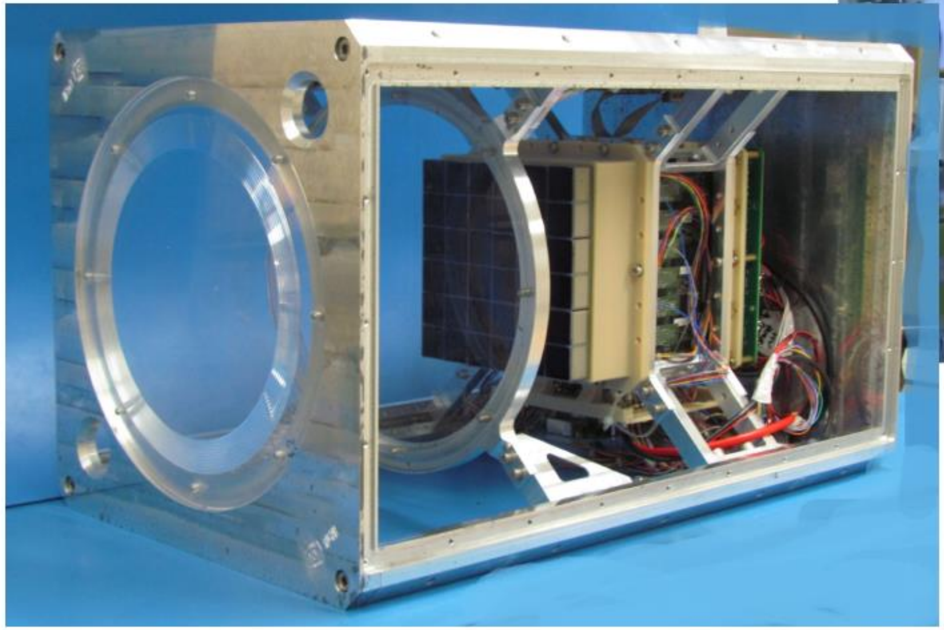
Hunting for ultraviolet transients with a neural network

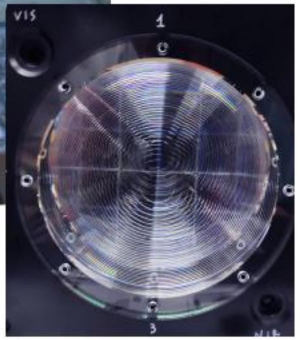
M. Zotov, D. Anzhiganov, A. Kryazhenkov

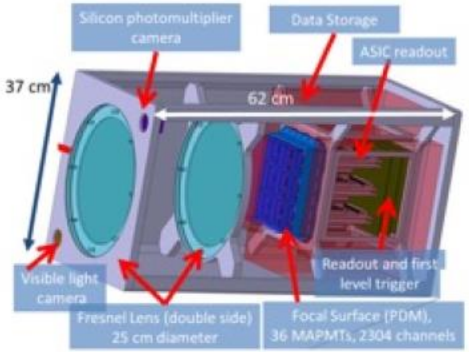
Lomonosov Moscow State University, Russia

OpenTalks.AI, 6–7 March, 2023, Yerevan, Armenia

Intro



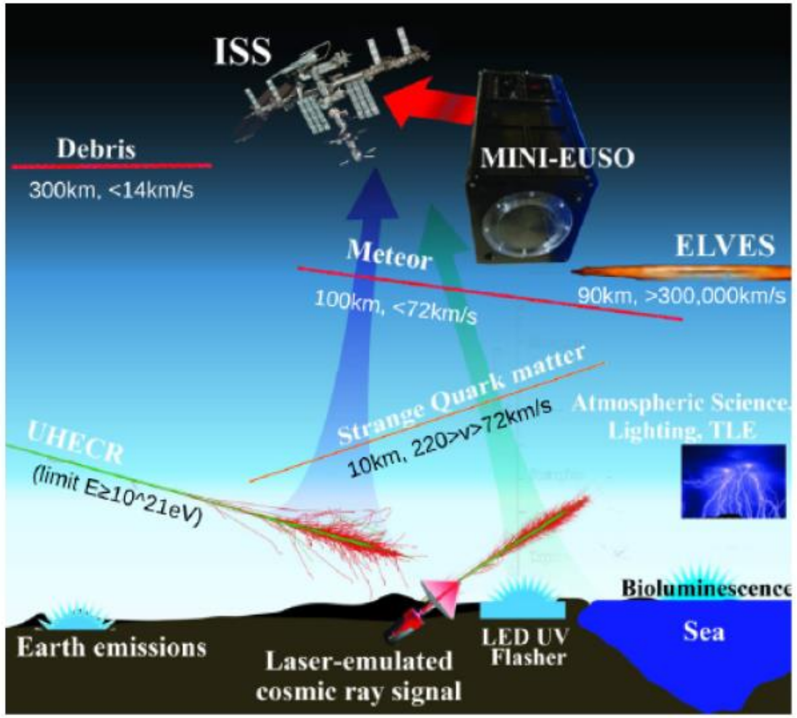


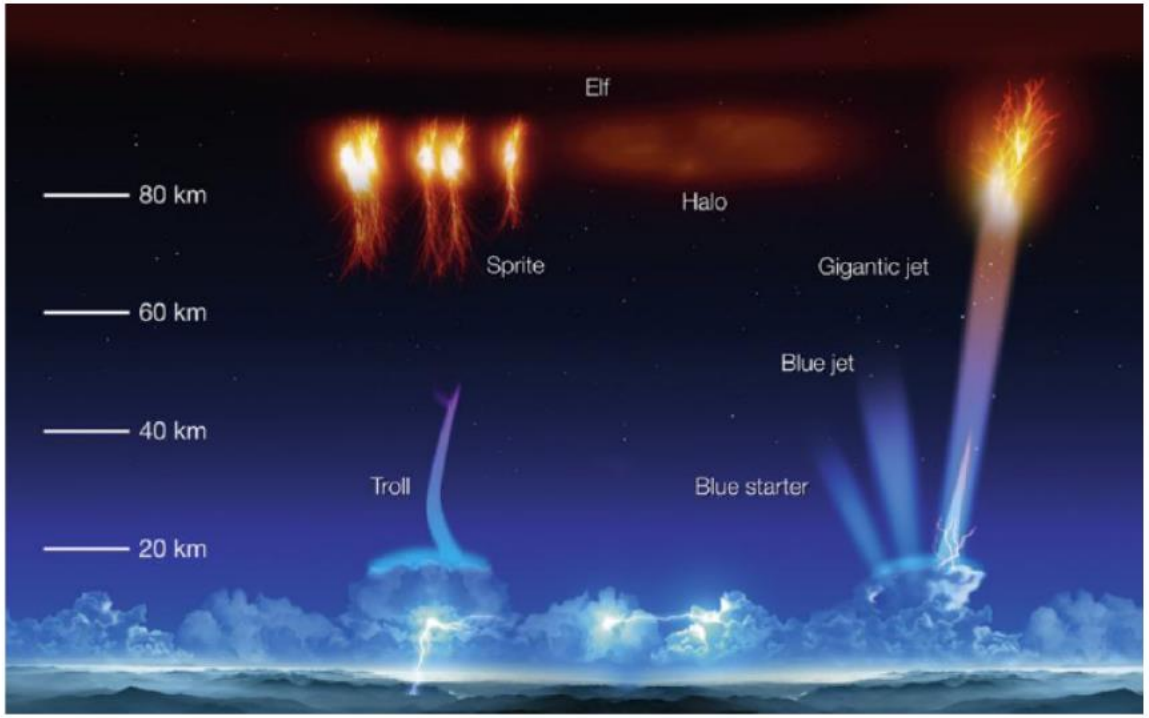


The blue checkered structure behind is the focal surface. In the top left and bottom right corners are present the Visible and Near Infrared cameras

Mini-EUSO telescope onboard the ISS since 2019. Field of view: 48 × 48 pixels 300 × 300 km2

Zoo of UV transients in the Earth atmosphere

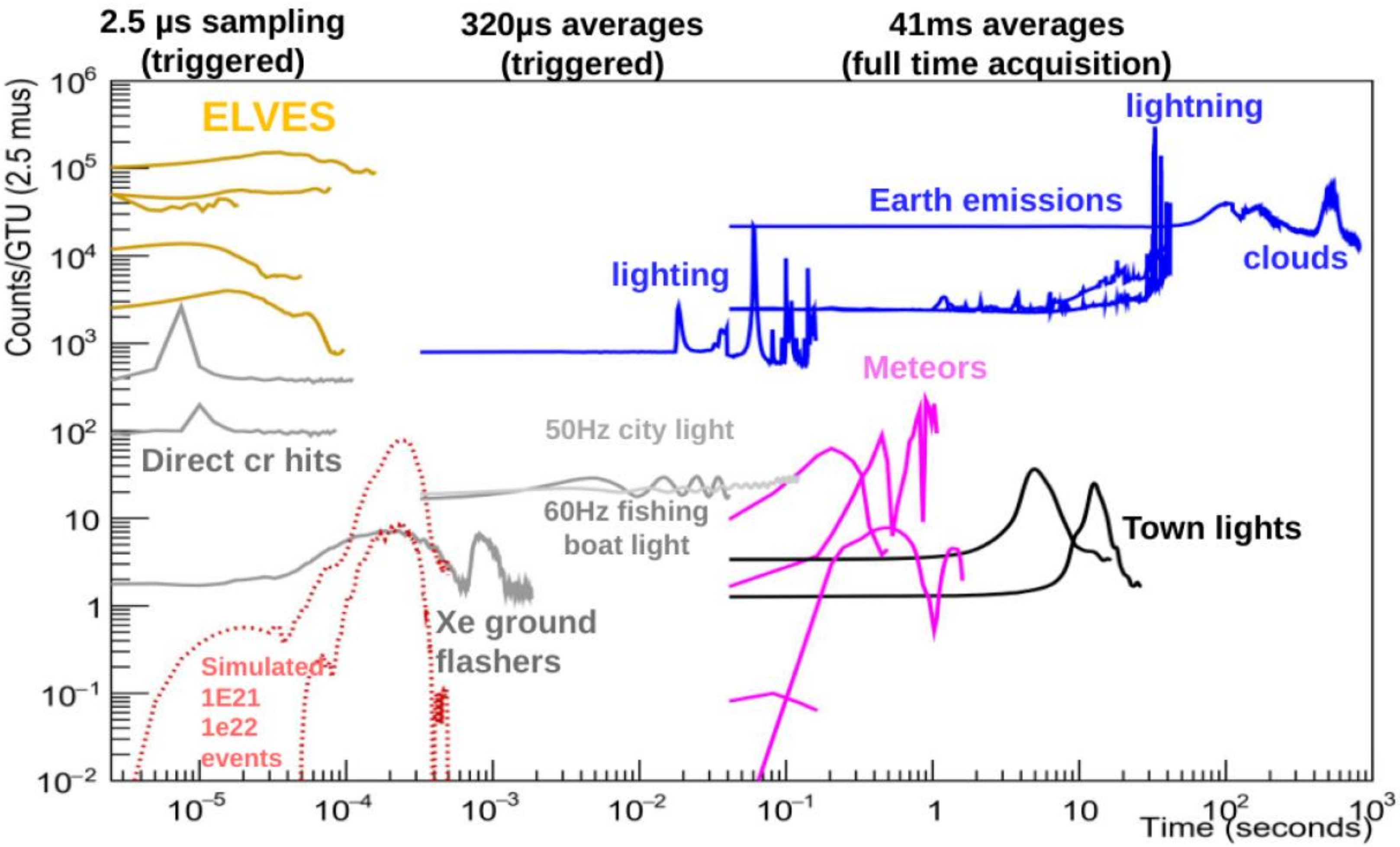




Sources. Left: Mini-EUSO collaboration. *Right:* Forbes

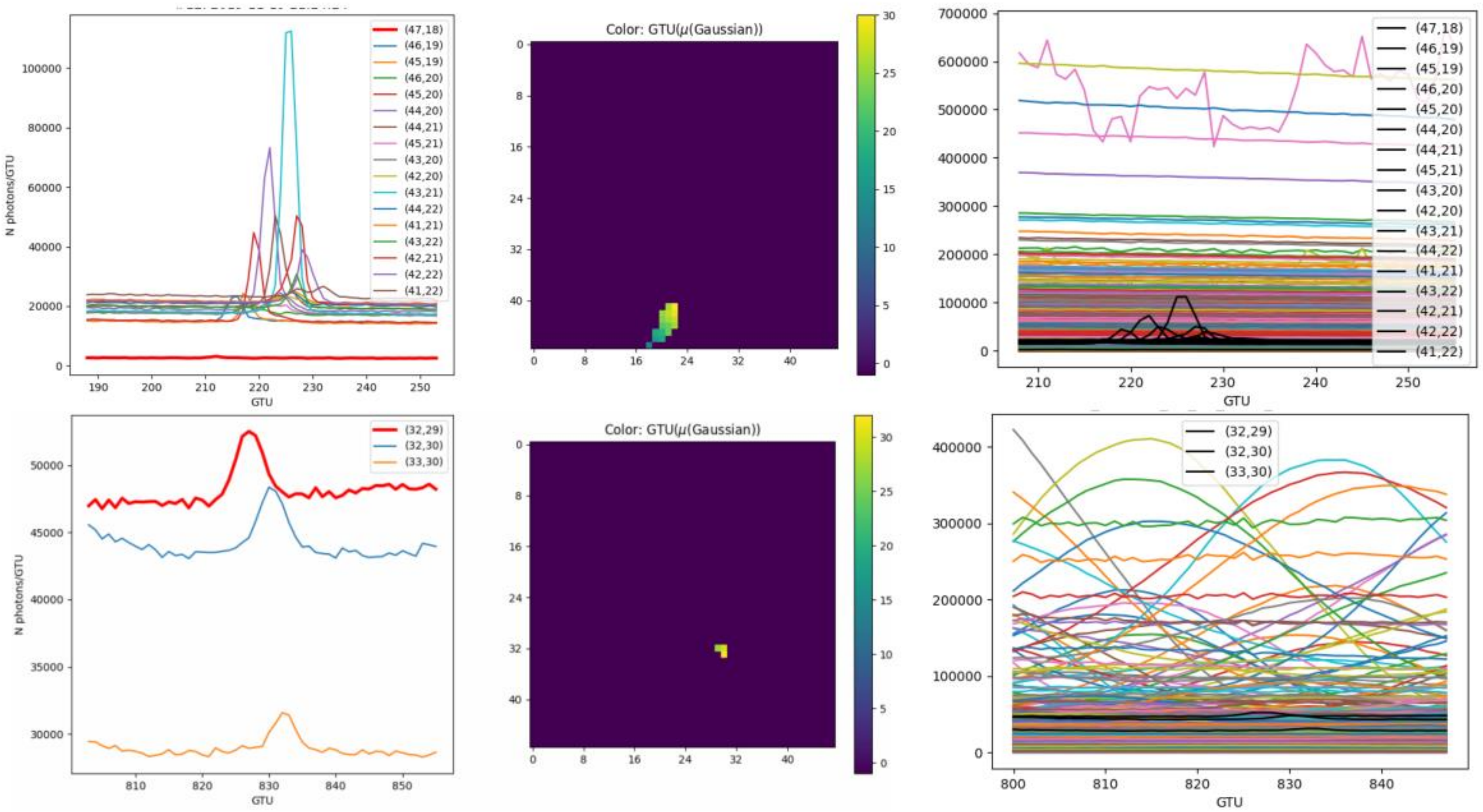
The variety of UV illumination is enormous!

Zoo of signals

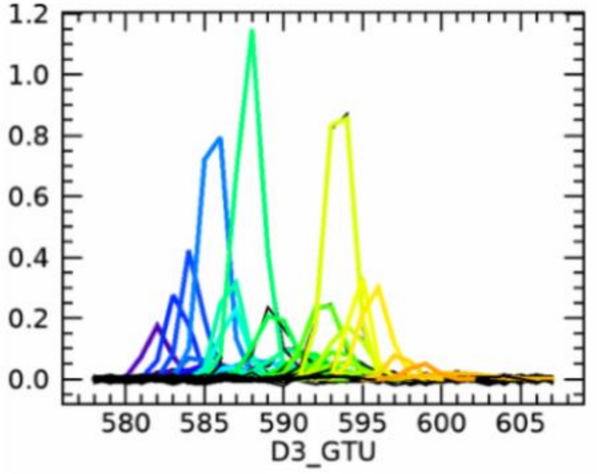


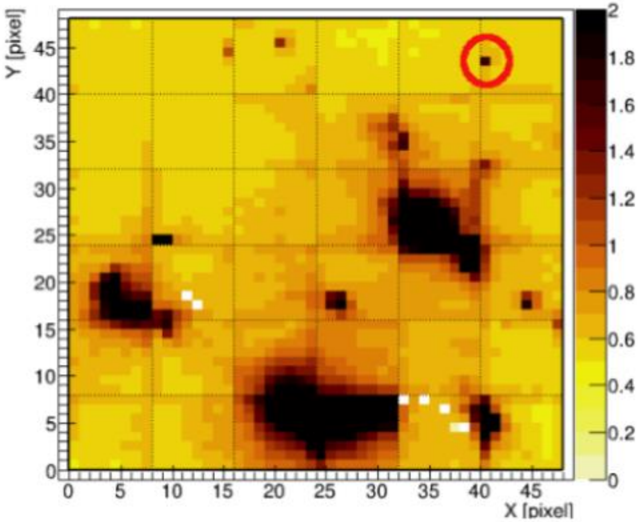
Sources: ICRC-2021

Meteors in Mini-EUSO



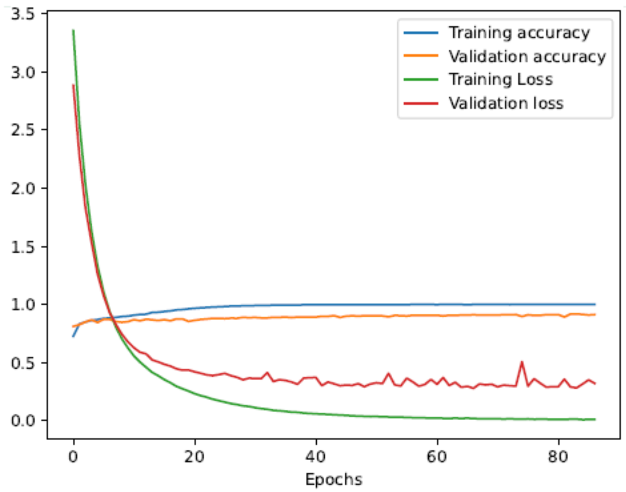
Hunting meteors × *Why it’s not so easy*

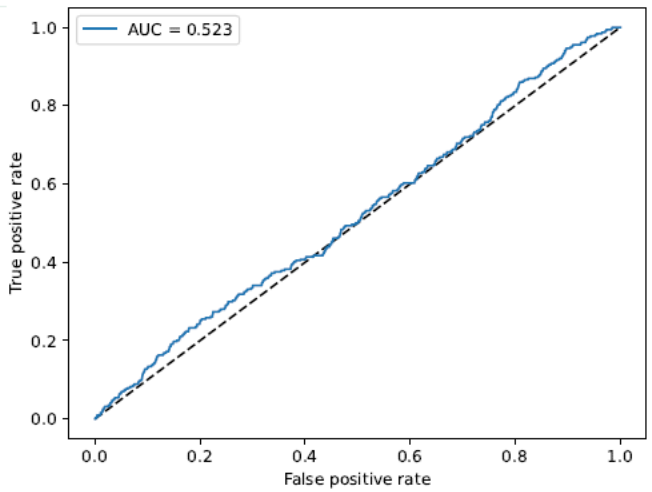


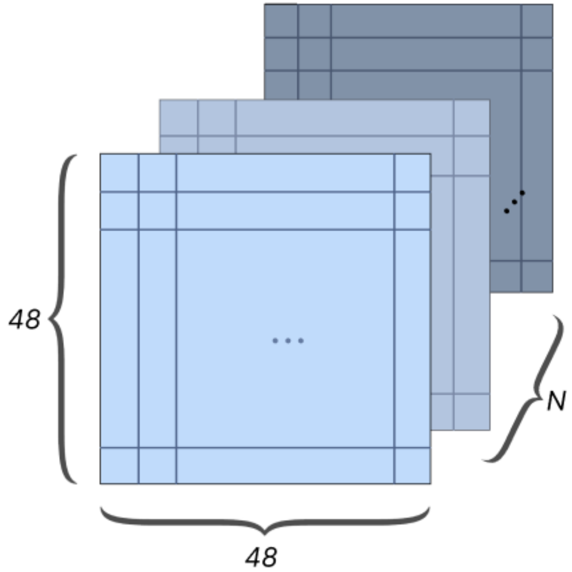


- Training data set is small (1100 meteors)
- Majority of signals occupy only 3–4 pixels out of 48 × 48
- “non-meteor” signals are often much brighter and larger than meteor ones
- Background shifts due to the ISS movement

Classification







CNN with one convolutional layer, one pooling layer, two fully connected layers

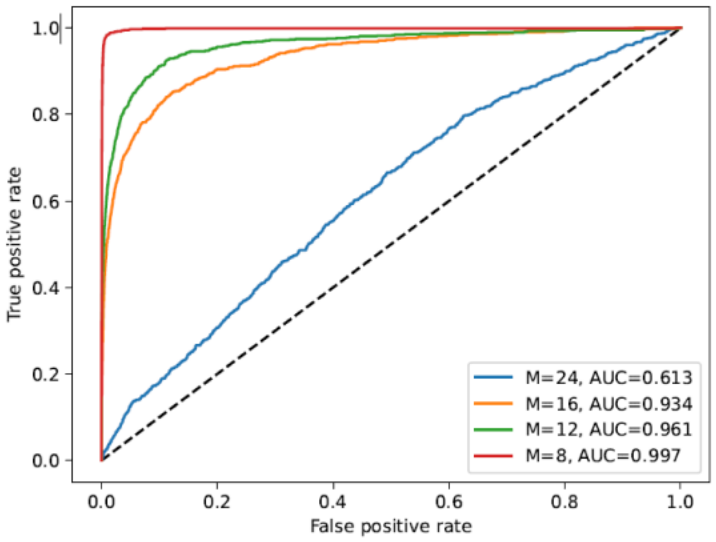
Input data shape: 48 × 48 × N, where N = 8 . . . 64

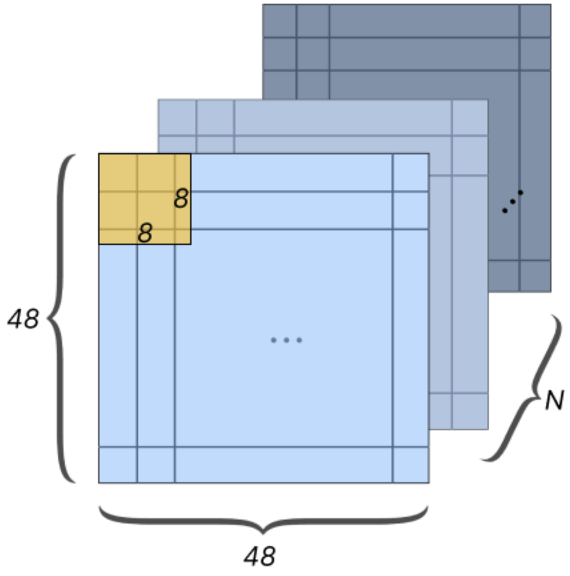
Training (validation) *accuracy:* 0.99 (0.89). However: AUC on a testing set: 0.523

Other hyperparameters/more convolutional layers/LSTM: AUC~0.75

Classification

Split the focal surface into small pieces (for the same trivial CNN)!

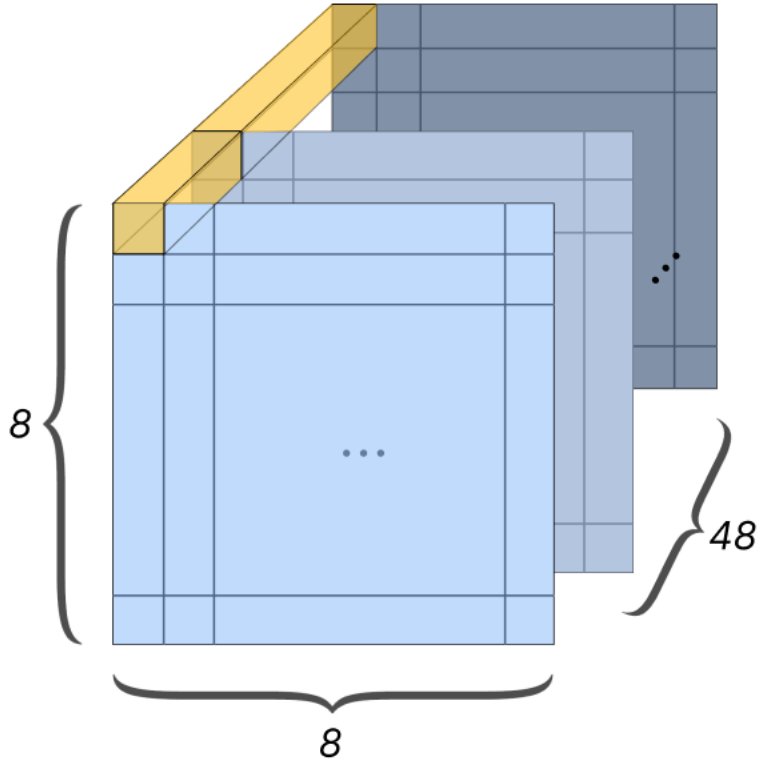


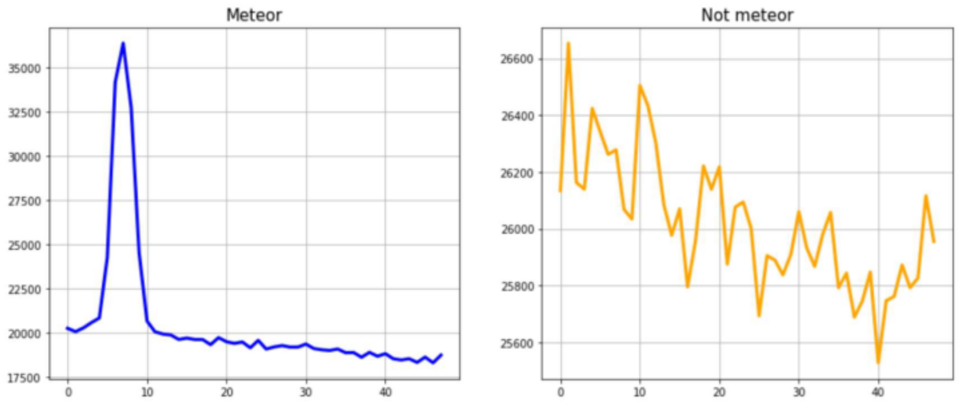


Input data shape: 8 × 8 × N, where N = 8 . . . 64 (good: 48)

It's not enough to find chunks of data that contain meteor signals.
We want to know exactly the pixels where the signal is located.

Data × *Transform*





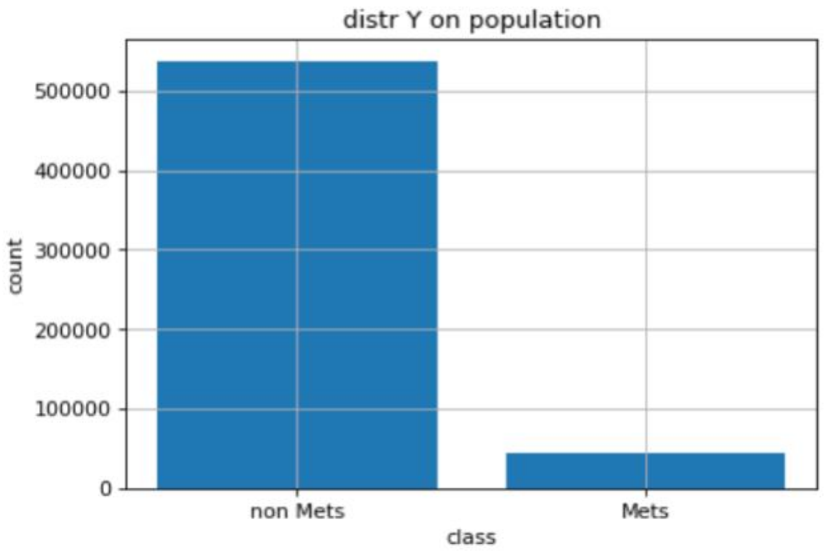
Input data shape: vector with size N = 48

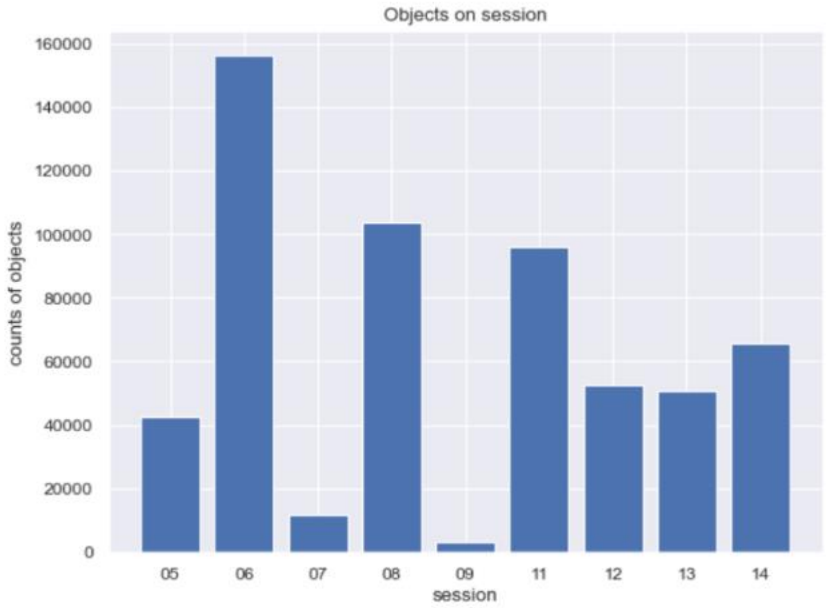
Meteor set size 537864 and non Meteor set size 43896

Slicing the focal surface into pixels (channels)

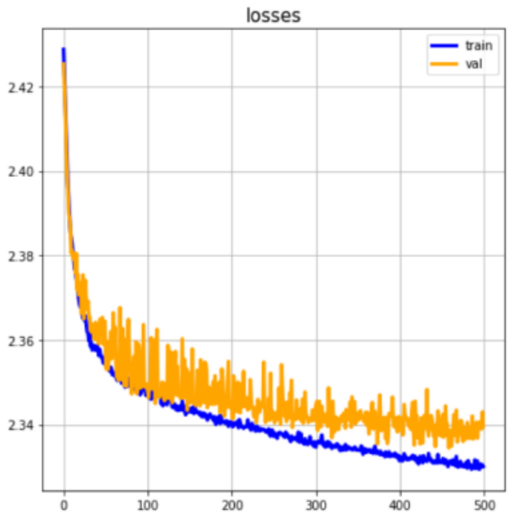
Pure data × *Sessions*

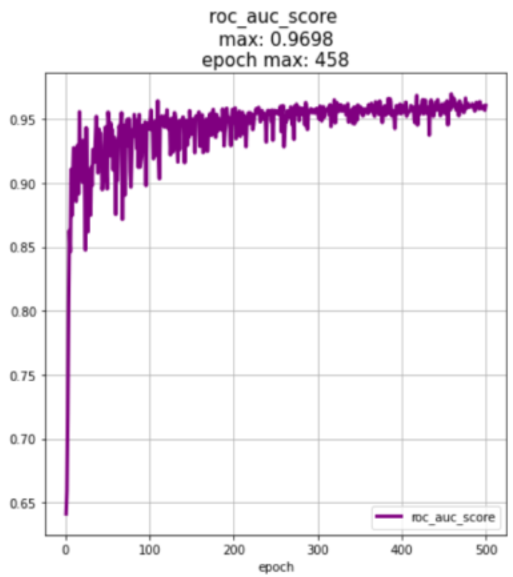
537864 43896





NN × *Pure data* × *Example*





Train: Session №5

Test: Session №6

Arch: (48, 64) ReLu

(64, 32) Adam

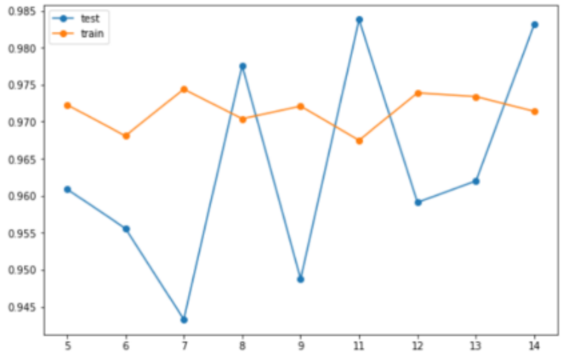
(32, 2)

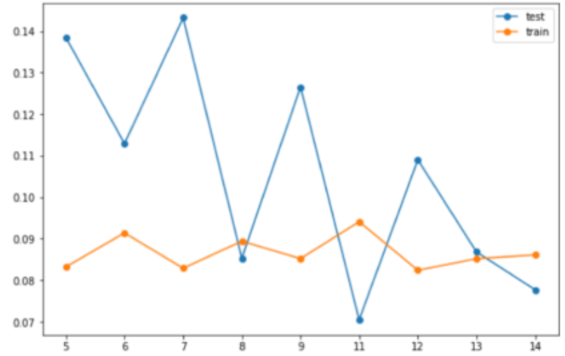
Max Auc: 0.9698

* Increase the number of layers - not so good

NN × *Pure data*

Params = {Exponential Decay, 'Sigmoid', Binary Crossentropy, Batch = 128, Adam}





Units on first hidden layer	Units on second hidden layer	Mean AUC on Sessions
32	32	0.9562
32	64	0.9584
32	96	0.9501
32	128	0.9438
64	32	0.9589
64	64	0.9559
64	96	0.9588
64	128	0.9428
96	32	0.9638
96	64	0.9637
96	96	0.9602
96	128	0.9619
128	32	0.9619
128	64	0.9606
128	96	0.9613
128	128	0.9614

Max roc_auc_score: 0.9638 (0.0137)

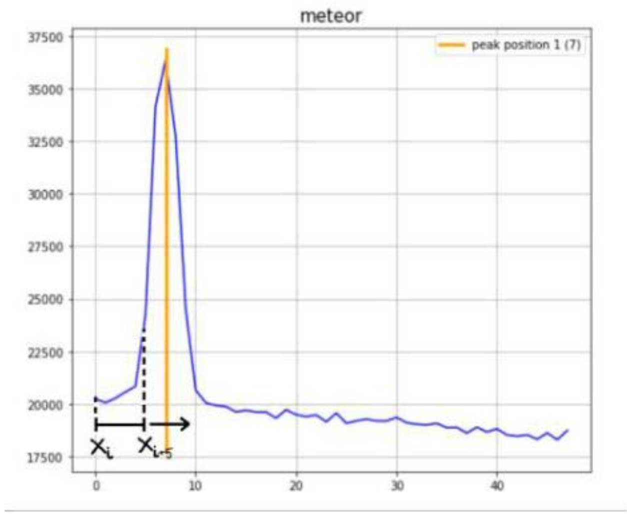
ML based methods × *Pure data*

Pure data

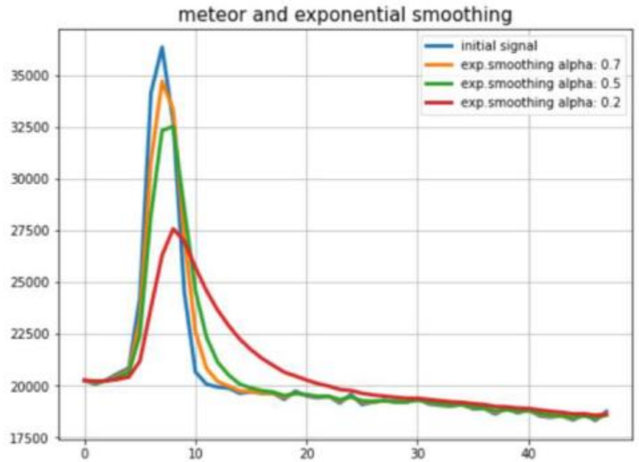
	5	6	7	8	9	11	12	13	14	mean ROC AUC:
KNN	0.977	0.984	0.946	0.986	0.933	0.990	0.980	0.964	0.988	**0.972**
Random Forest	0.968	0.983	0.9017	0.985	0.925	0.990	0.972	0.966	0.988	**0.964**
XGBoost	0.720	0.764	0.480	0.812	0.443	0.778	0.745	0.794	0.770	0.701
Logistic Regression	0.725	0.733	0.734	0.752	0.707	0.736	0.739	0.730	0.770	0.737

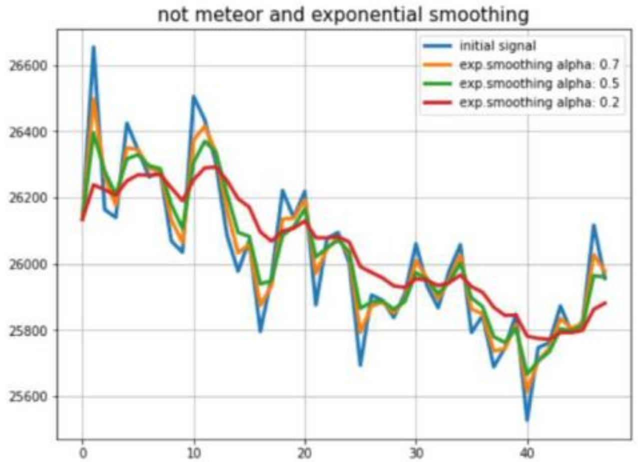
Adding new features

Set fixed size window



Smoothing

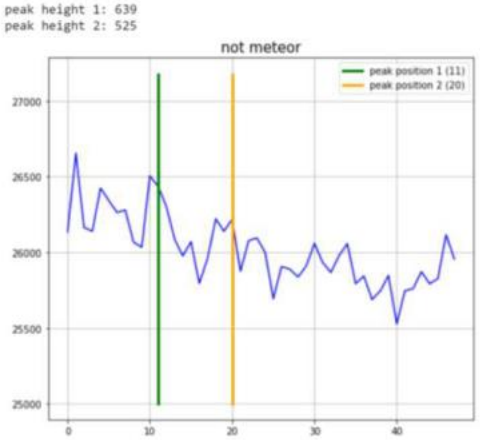


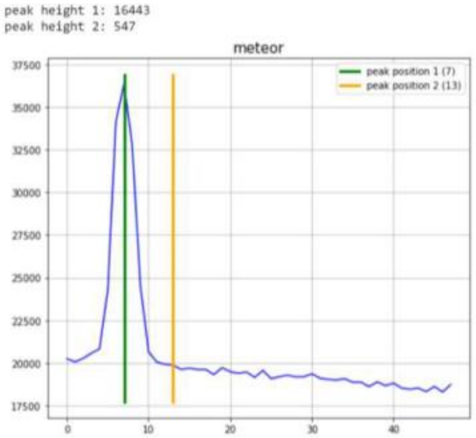


Lets create new features!

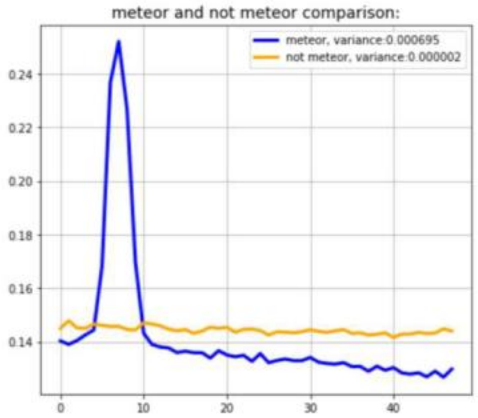
Adding new features

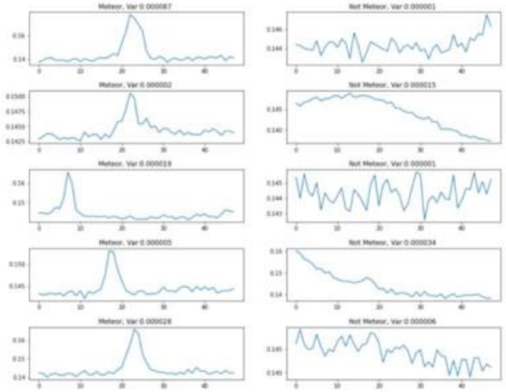
Two peaks
and thier position





Variance





Input data shape:
vector with size = 5

Models × *New data + Pure data*

New data

	5	6	7	8	9	11	12	13	14	mean ROC AUC:
KNN	0.964	0.978	0.908	0.979	0.919	0.986	0.967	0.958	0.982	**0.96**
Random Forest	0.967	0.979	0.926	0.981	0.866	0.986	0.968	0.959	0.981	**0.958**
XGBoost	0.973	0.983	0.944	0.985	0.88	0.989	0.977	0.964	0.988	**0.966**
Logistic Regression	0.963	0.977	0.923	0.979	0.920	0.985	0.966	0.961	0.982	**0.956**

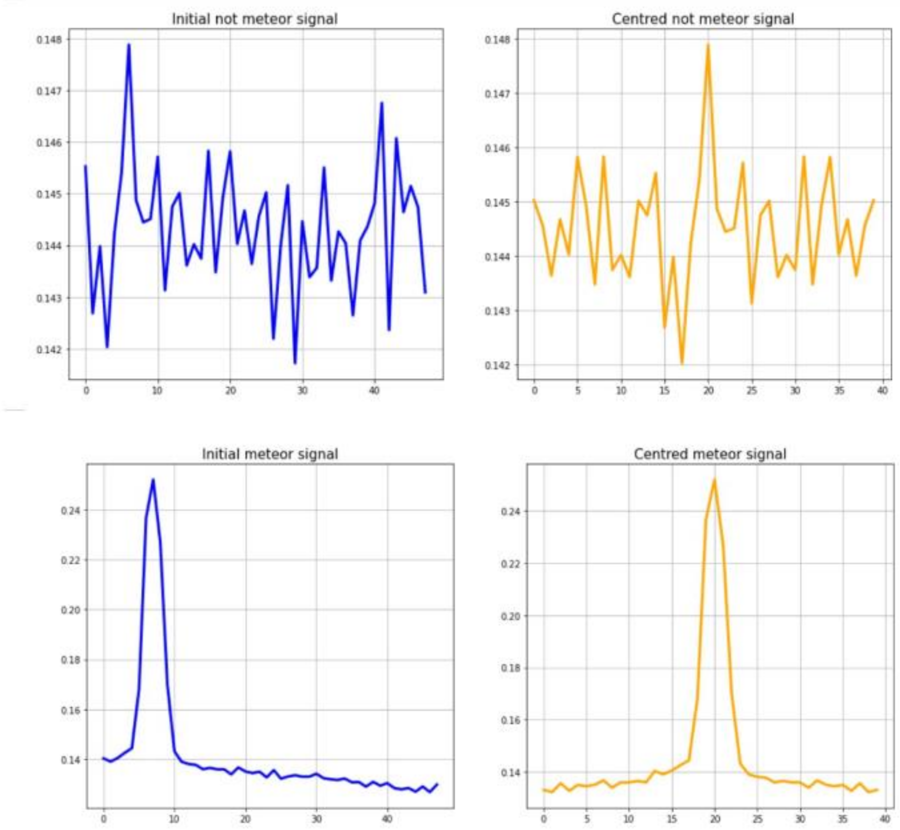
New data + Pure data

	5	6	7	8	9	11	12	13	14	mean ROC AUC:
KNN (pure data)	0.9772	0.9848	0.9468	0.9869	0.9333	0.9907	0.9808	0.9647	0.9901	**0.9728**
KNN (pure & features)	0.9774	0.9853	0.9468	0.9874	0.9427	0.9917	0.9806	0.9664	0.9902	**0.9742**

NN New data

	fully connected neural networks with ReLU() activations:									
	5	**6**	**7**	**8**	**9**	**11**	**12**	**13**	**14**	**mean ROC AUC:**
nn.sequential 5->16->2 (1 hidden layer)	0.961	0.975	0.948	0.978	0.819	0.985	0.959	0.953	0.978	**0.951**
nn.sequential 5->8->8->2 (2 hidden layers)	0.961	0.975	0.950	0.978	0.825	0.985	0.962	0.956	0.980	**0.953**
nn.sequential 5->8->8->8->2 (3 hidden layers)	0.9609	0.974	0.940	0.977	0.908	0.985	0.964	0.955	0.978	0.960
	note: neural networks with 2 and 3 hidden layers are very prone to overfitting on these features									

NN × *Centering*



Let's perform the centering process (Up: example for Non Mets and Down: for Non Mets)

Models × *New data* × *Centering*

	fully connected neural networks with ReLU() activations:									
	5	6	7	8	9	11	12	13	14	mean ROC AUC:
nn.sequential 5->16->2 (1 hidden layer)	0.961	0.975	0.948	0.978	0.819	0.985	0.959	0.953	0.978	0.951
nn.sequential 5->8->8->2 (2 hidden layers)	0.961	0.975	0.950	0.978	0.825	0.985	0.962	0.956	0.980	0.953
nn.sequential 5->8->8->8->2 (3 hidden layers)	0.9609	0.974	0.940	0.977	0.908	0.985	0.964	0.955	0.978	0.960
	note: neural networks with 2 and 3 hidden layers are very prone to overfitting on these features									

	5	6	7	8	9	11	12	13	14	mean ROC AUC:
KNN	0.977	0.984	0.946	0.986	0.933	0.990	0.980	0.964	0.990	0.972
Random Forest	0.968	0.983	0.9017	0.985	0.925	0.990	0.972	0.966	0.988	0.964
XGBoost	0.720	0.764	0.480	0.812	0.443	0.778	0.745	0.794	0.770	0.701
Logistic Regression	0.725	0.733	0.734	0.752	0.707	0.736	0.739	0.730	0.770	0.737
SVM	for 6 hours of calculations, the method did not give results for at least one session									

	5	6	7	8	9	11	12	13	14	mean ROC AUC:
KNN (pure data)	0.9772	0.9848	0.9468	0.9869	0.9333	0.9907	0.9808	0.9647	0.9901	0.9728
KNN (pure & features)	0.9774	0.9853	0.9468	0.9874	0.9427	0.9917	0.9806	0.9664	0.9902	0.9742

	centred peaks:										
	5	6	7	8	9	11	12	13	14	mean ROC AUC:	
KNN (pure data)	0.9772	0.9848	0.9468	0.9869	0.9333	0.9907	0.9808	0.9647	0.9901	0.9728	
KNN (pure & features)	0.9774	0.9853	0.9468	0.9874	0.9427	0.9917	0.9806	0.9664	0.9902	0.9742	
KNN (centred peaks 10)	0.9787	0.9844	0.9522	0.9883	0.9371	0.9912	0.9812	0.9670	0.9906	0.9745	preproc.normalize
KNN (centred peaks 12)	0.9795	0.9846	0.9613	0.9888	0.9368	0.9917	0.9817	0.9695	0.9914	0.9761	preproc.normalize
KNN (centred peaks 16)	0.9810	0.9849	0.9662	0.9890	0.9377	0.9916	0.9832	0.9701	0.9924	0.9773	preproc.normalize
KNN (centred peaks 16)	0.9794	0.9854	0.9789	0.9878	0.9368	0.9906	0.9818	0.9677	0.9922	0.9778	min_max scale
KNN (centred peaks 20)	0.9811	0.9859	0.9686	0.9894	0.9447	0.9910	0.9829	0.9711	0.9919	0.9785	preproc.normalize

Conclusions

Neural networks work fine in recognizing meteors in the Mini-EUSO UV telescope data

The ML-based and neural-network-based approach excels conventional algorithms in accuracy and speed (10 times!). *But this requires transforming the data and creating new features*

The NNs trained to find meteors can find signals with similar shape and kinematics but with a completely different nature (extensive air showers from cosmic rays!)

NNs can be implemented in onboard electronics in the future orbital experiments

Thank you for your attention!