

# Ariadne: PyTorch Library for Particle Track Reconstruction Using Deep Learning

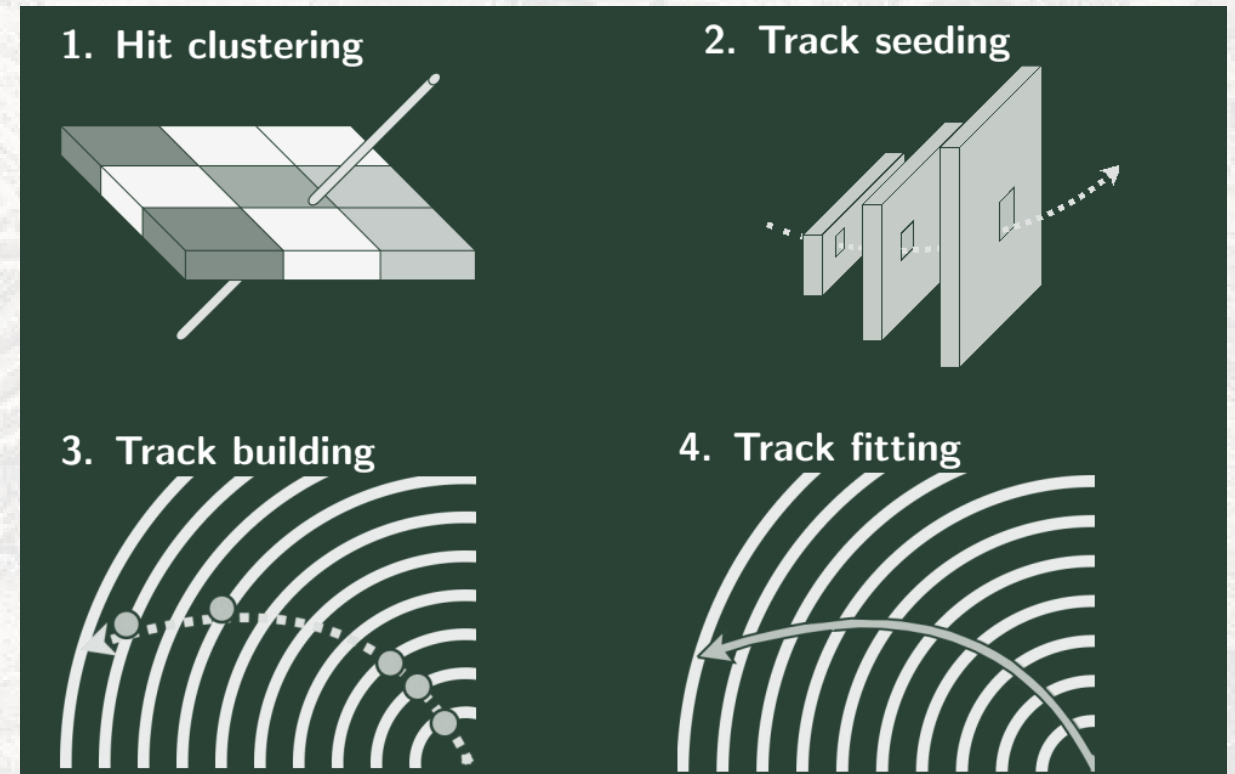
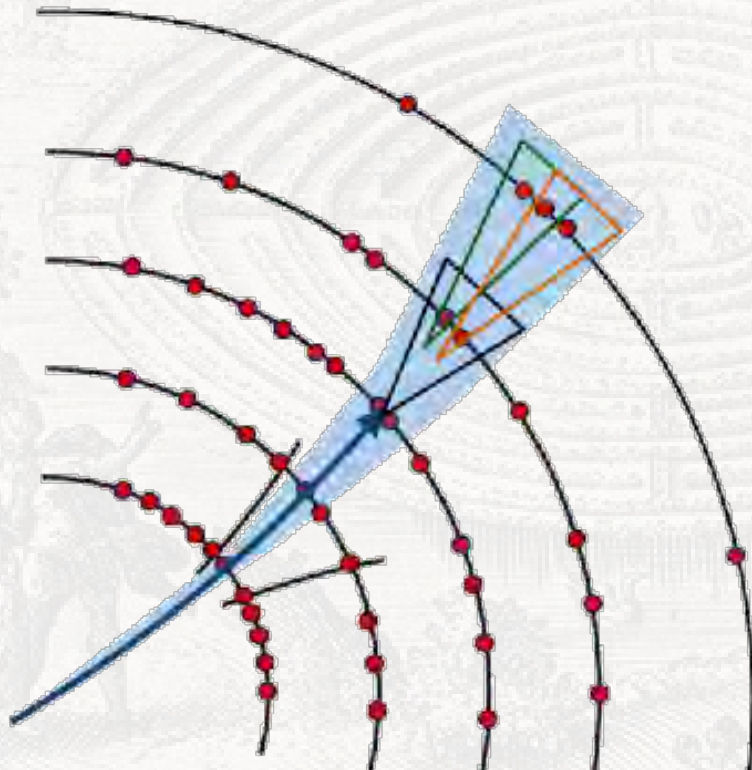
**Pavel Goncharov**, Egor Schavelev, Anastasia Nikolskaya, Gennady

The reported study was funded by <sup>Osofkov</sup> RFBR according to the research project Joint Institute for Nuclear Research, Saint Petersburg State University № 18-02-40101



# Tracking is our main focus

**Tracking or track finding** is a process of **reconstruction the particle's trajectories** in high-energy physics detector by connecting the points – hits – that each particle leaves passing through detector's planes. Tracking includes track seeding and track building phases.



# Dream team of tracking

## Data Science team

- Gennady Ososkov
- Pavel Goncharov
- Egor Schavelev
- Anastasia Nikolskaya
- Ekaterina Rezvaya

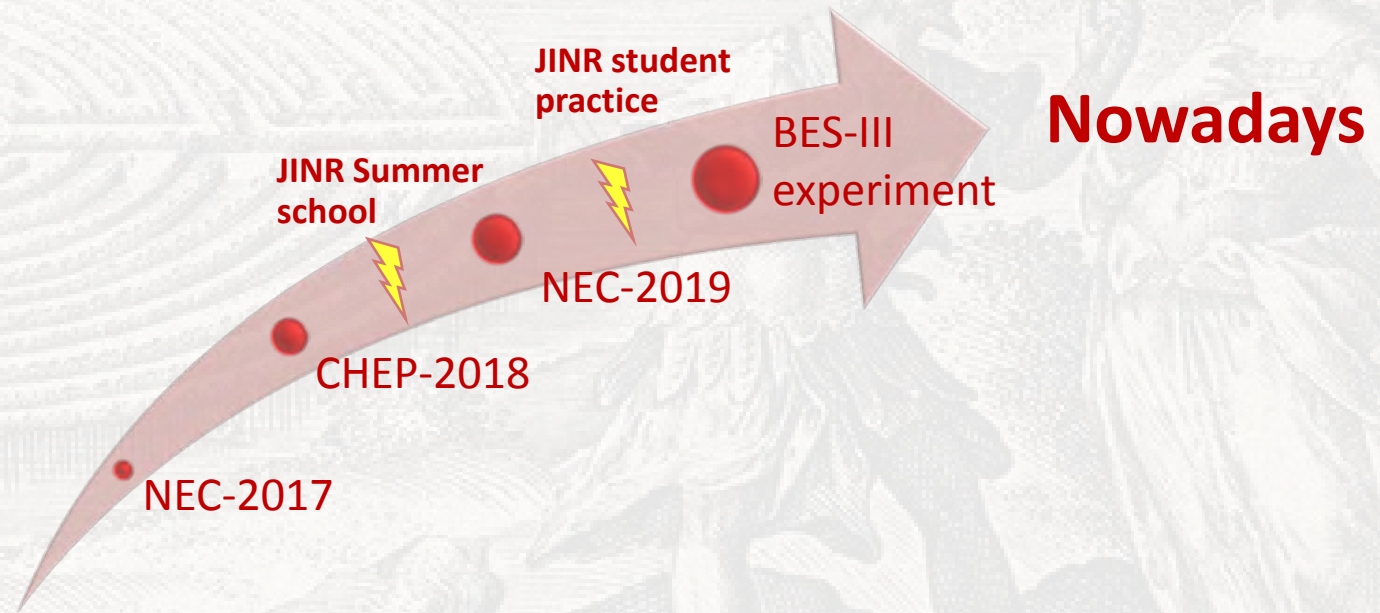
**Team is growing!**

## Physicists kernel

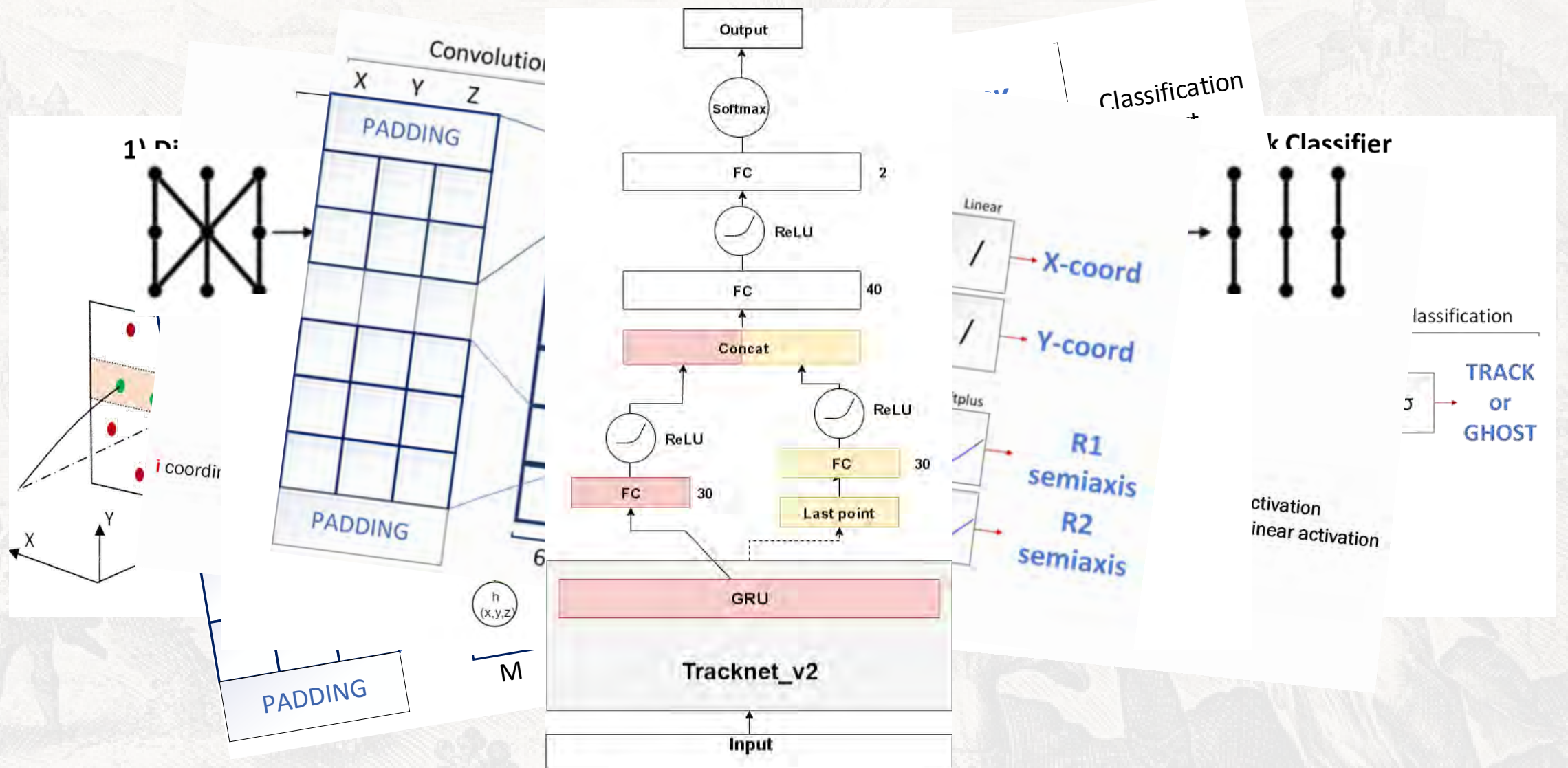
- Dmitry Baranov
- Aleksey Zhemchugov
- Igor Denisenko
- Yuri Nefedov

## Grants management

- Andrey Nechaevskiy

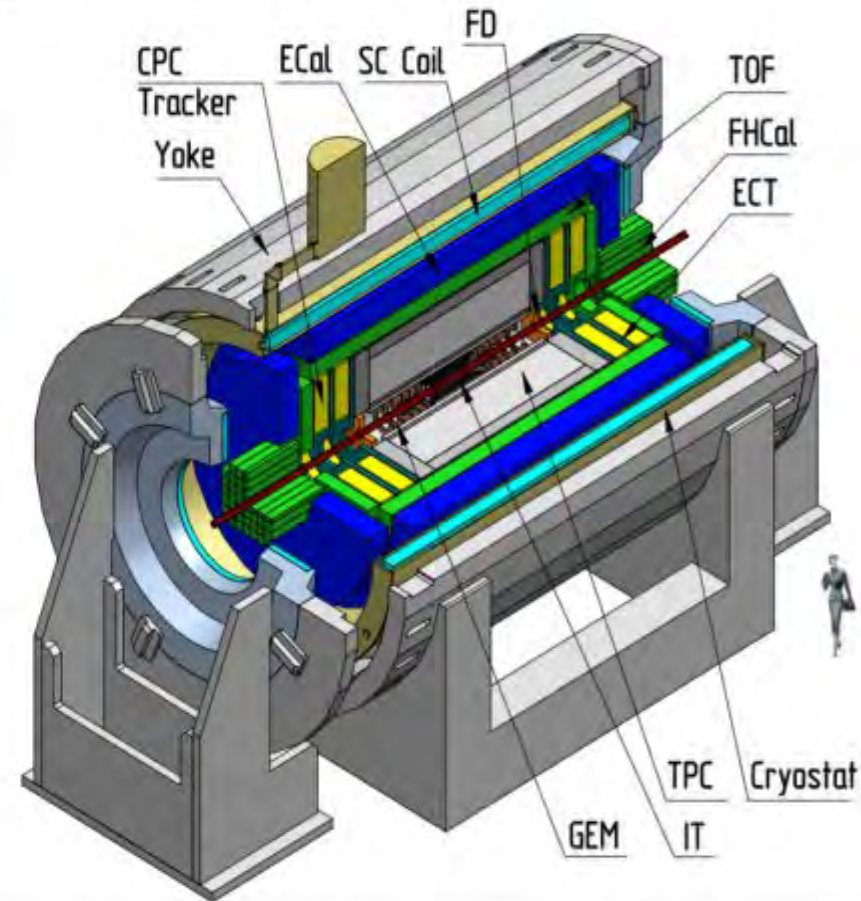
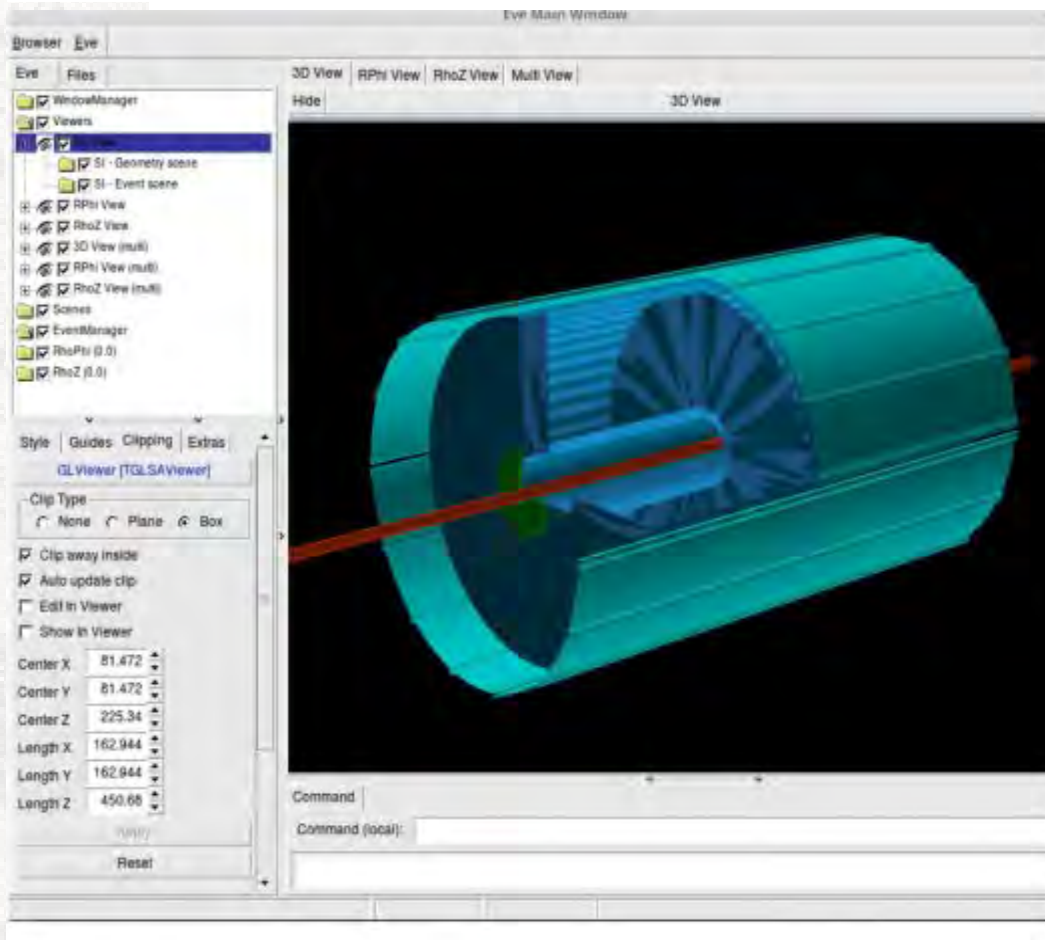


# More and more models each time



# Same for experiments

MPD in the end ???



# How to become the master of chaos

The problem is that all these models have:

**SAME** data preprocessing  
training skeleton  
metrics

Moreover, they are:

- Separated in **different** github repositories
  - with **different** freshness code (two-step approach became legacy almost a hundred years ago)
  - Even with **different** deep learning backends (**Old TensorFlow** vs **Keras** vs **TF 2.0** vs **PyTorch**)
1. Therefore, it is very difficult to keep all the repositories fresh and compare methods for equal criteria.
  2. Besides, it seems almost impossible that any physicist will understand how to apply the proposed methods to his problem and data.

# Here Ariadne appears on the stage

**Our goal** is to create **the first open-source library for particle tracking** based on deep learning methods.

We named it Ariadne in fame of Ariadne's thread which means **the path leading to the goal in difficult conditions** as in the tracking process.

## Library should

- provide a simple interface that allows one to prepare his data
- implement all our best models with ability to train them on an arbitrary tracking task
- have a modular structure and abstract classes simplifying the process of developing custom model and data processing pipeline
- include the system of configuration files for fast reproducing of the experiments with 100% reproducibility
- be open-sourced to facilitate academic research in the field of particle tracking based on deep learning

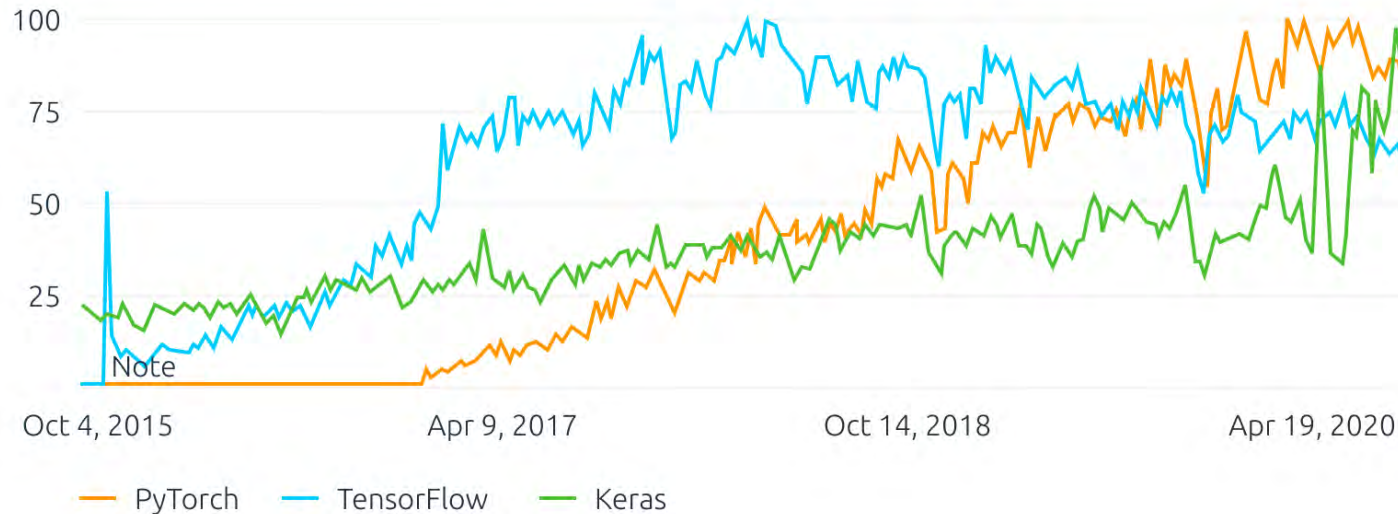
## Also, our wish list includes many other small points:

- multi-GPU training
- hyperparameters optimization
- metrics logging
- multiprocessing for data preparation
- etc.



# Ariadne is PyTorch-based

## TENSORFLOW VS KERAS VS PYTORCH



Data source: Google Trends

IFLE  ION

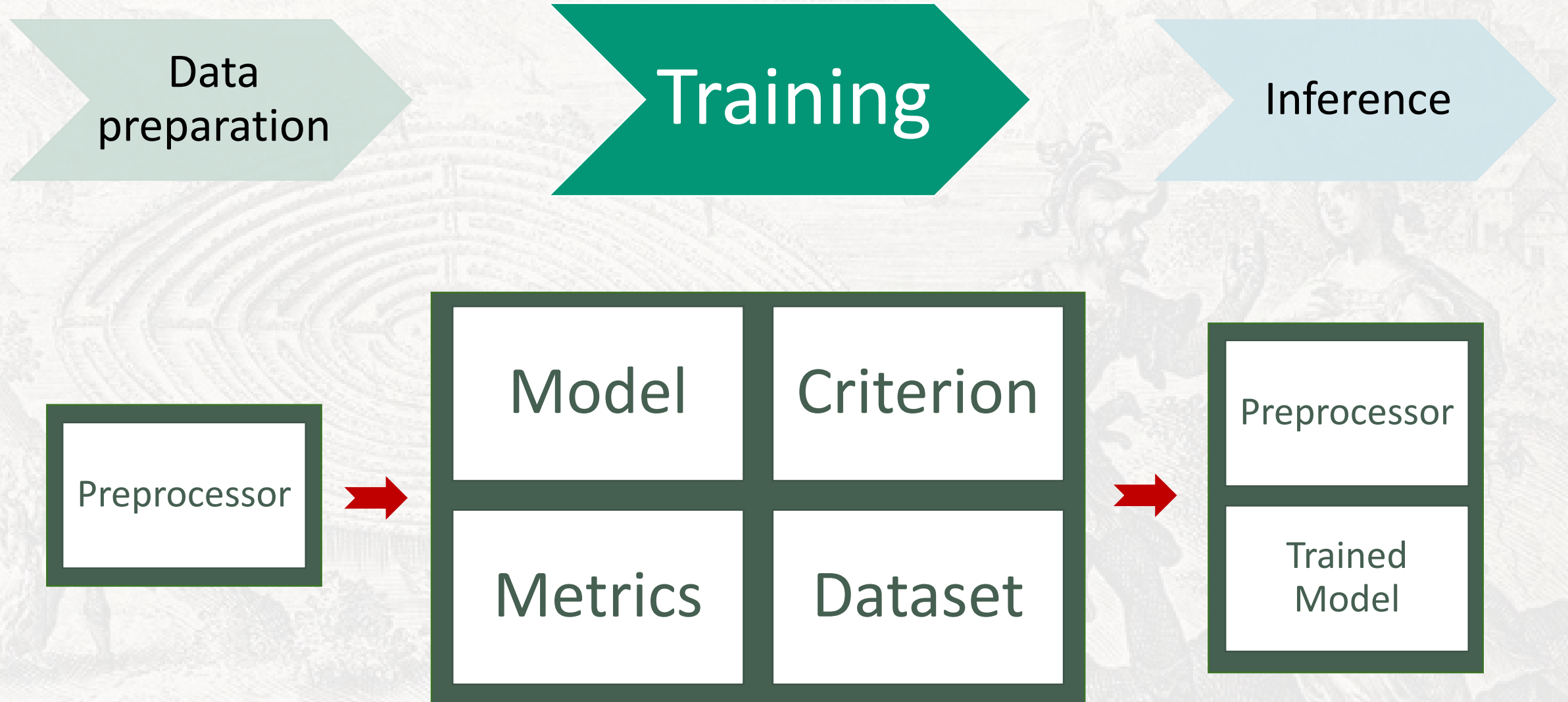
We have chosen PyTorch as a base deep learning framework for our library because

- until recently (TF 2.3) tensorflow didn't have full reproducibility on GPUs
- pytorch Datasets much more convenient and flexible in our opinion than tf.data.Dataset
- at this time researches prefer pytorch, so most of the novel models appear in pytorch at first



# Ariadne's concept

*Quint. Metam. libro 8.*



# Data Preparation

For the data preparation step we introduce **transformations** that allow us to:

- translate coordinates into another system, e.g. cartesian to cylindrical
- filter out inapplicable tracks or events
- normalize and rescale features to be in an unit variance

At the moment, we have a plenty of transformations:

- StandardScale
- MinMaxScale
- Normalize
- ConstraintsNormalize
- DropShort
- DropSpinningTracks
- DropFakes
- ToCylindrical
- ToCartesian
- ToBuckets

Also we **can compose multiple transformations** in one block for convenience

```
self.transformer = Compose([
    DropFakes(),
    DropSpinningTracks(),
    DropShort(),
    ToCylindrical(),
    ConstraintsNormalize(
        use_global_constraints=False,
        constraints=radial_stations_constraints,
        columns=('r', 'phi', 'z')
    ),
])
```




# PyTorch Lightning as a core of training

PyTorch Lightning is a lightweight PyTorch wrapper for high-performance AI research as stated on its github. Lightning helps you to disentangle code to make it more readable, flexible and suitable for testing.


We created a special class **TrainModel** which is actually a Lightning module to extend the functionality of simple pytorch code with Lightning features:

- full reproducibility
- checkpointing
- callbacks
- metrics logging
- multi-GPU training
- TPU training
- learning rate schedulers
- batch size optimization
- etc.



PyTorch Lightning

```
dm = MNISTDataModule(batch_size=32)
model = MLPClassifier()
trainer = Trainer(callbacks=SSLOnlineEvaluator(), ConfusedLogitCallback())
trainer.fit(model, dm)
```



# System of configuration files

To allow the users to reproduce our experiments and setup their own easily we introduce a structure of configuration files, consisting of:

- train configs
- data preparation configs
- inference/evaluation configs

Standard Python ArgumentParser is not very flexible so we decided to utilize **gin-configs**.



Gin-config opens an opportunity to specify the whole classes with their arguments inside a configuration file, so in future all the users of our library will be able to “program” any training from scratch using Ariadne modules and such approach requires no knowledge of Python.

## Example

```
### experiment setup ###
experiment.model = @TrackNETv2
experiment.criterion = @TrackNetLoss
experiment.metrics = [@ellipse_area, @efficiency]
experiment.optimizer = @Adam
experiment.data_loader = @TrackNetV2DataLoader
experiment.epochs = 20
experiment.fp16_training = False
experiment.random_seed = 42
```

```
### model ###
TrackNETv2.input_features = 3
TrackNETv2.conv_features = 32
TrackNETv2.rnn_type = 'gru'
TrackNETv2.batch_first = True
```

```
### data ###
TrackNetV2DataLoader.batch_size = 16
TrackNetV2DataLoader.dataset = @TrackNetV2Dataset
TrackNetV2DataLoader.max_size = 10000
TrackNetV2DataLoader.valid_size = 0.3
```



TrackNETv2 is a class inherited from pytorch.Module

# Run training in the HybriLIT

As we are working on the base of Laboratory of Information Technologies we want to be able to run all our scripts in the HybriLIT's environment and especially on the GOVORUN supercomputer.

We added several scripts in the 'scripts/hydra' path for that purpose. So now you don't need to build an environment, download dependencies and manage conflicts between libraries versions. We created two configurations of conda environments – for gpu and cpu usage.

For example, to execute training script on the GPU queue of hydra cluster:

1. Verify that the miniconda has installed in the `~/miniconda3` or manually change the path in the script you want to execute. Run with `source ~/miniconda3/etc/profile.d/conda.sh` command
2. Run `scripts/hydra/hydra_gpu.sh` script:

```
sbatch scripts/hydra/hydra_gpu.sh python train.py --config resources/gin/tracknet_v2_train.cfg
```

3. The `slurm-jobid.out` file with stdout will appear in the root directory.

For more details see the README in the <https://github.com/t3hseus/ariadne> repository.

**Slow Progress**



**Is Still Progress**

# Inference

*Quin Metam Libro 8*

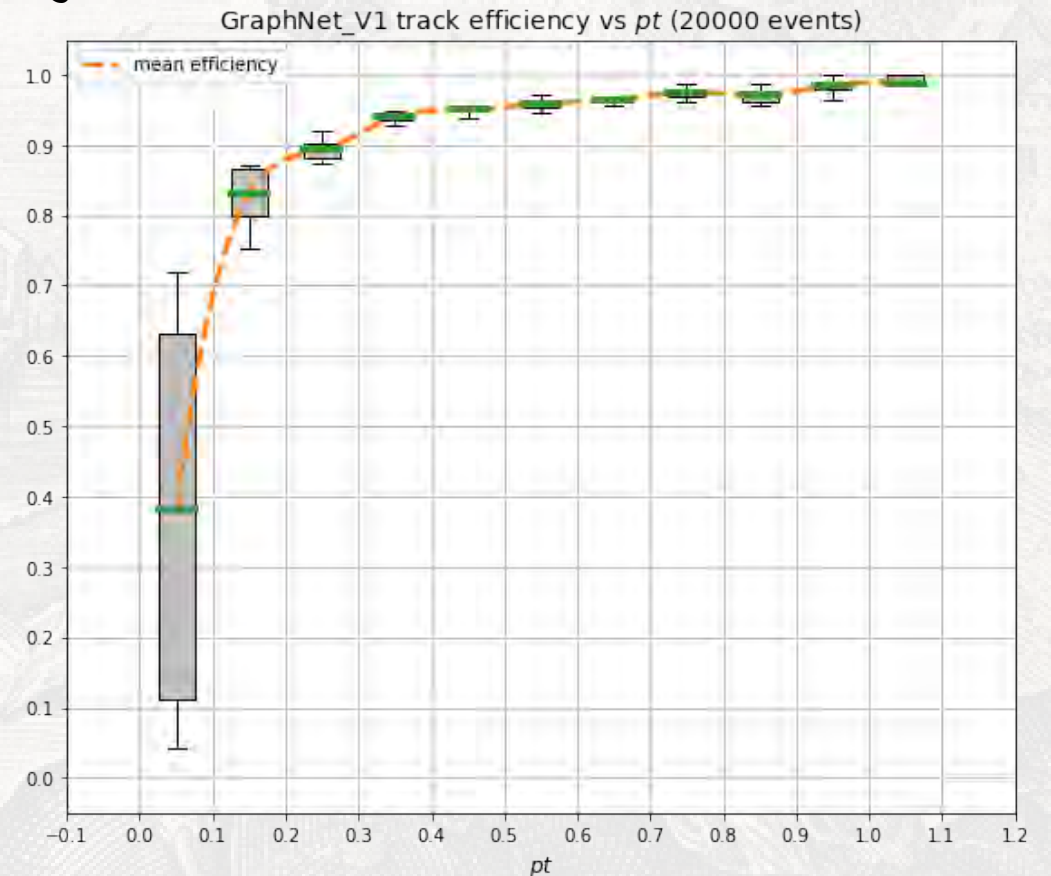


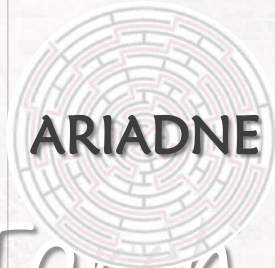
# Intermediate results

This is not the last chapter of our Ariadne's history, but we have already achieved some results:

- most of the code for training is completed, however it requires a refactoring
- data preparation pipeline for the TrackNETv2 and GraphNet models on the BESIII data was made
- two models – TrackNETv2 and GraphNet have already been trained using Ariadne (watch tomorrow's presentations of *Egor Schavelev* and *Anastasia Nikolskaya*)
- models were trained using HybriLIT cluster

**SPOILER  
ALERT!**





# Ariadne: PyTorch Library for Particle Track Reconstruction Using Deep Learning

**Pavel Goncharov**, Egor Schavelev, Anastasia Nikolskaya, Gennady

The reported study was funded by <sup>Ososkov</sup> RFBR according to the research project Joint Institute for Nuclear Research, Saint Petersburg State University № 18-02-40101

Ariadne Github:

