# Introduction to Deep Learning with NumPy and PyTorch 4/4

## Part 4

## This and that

### 1st Terascale Alliance Machine Learning School, DESY, Hamburg

Dirk Krücker
DESY - Hamburg, 23.10.2018

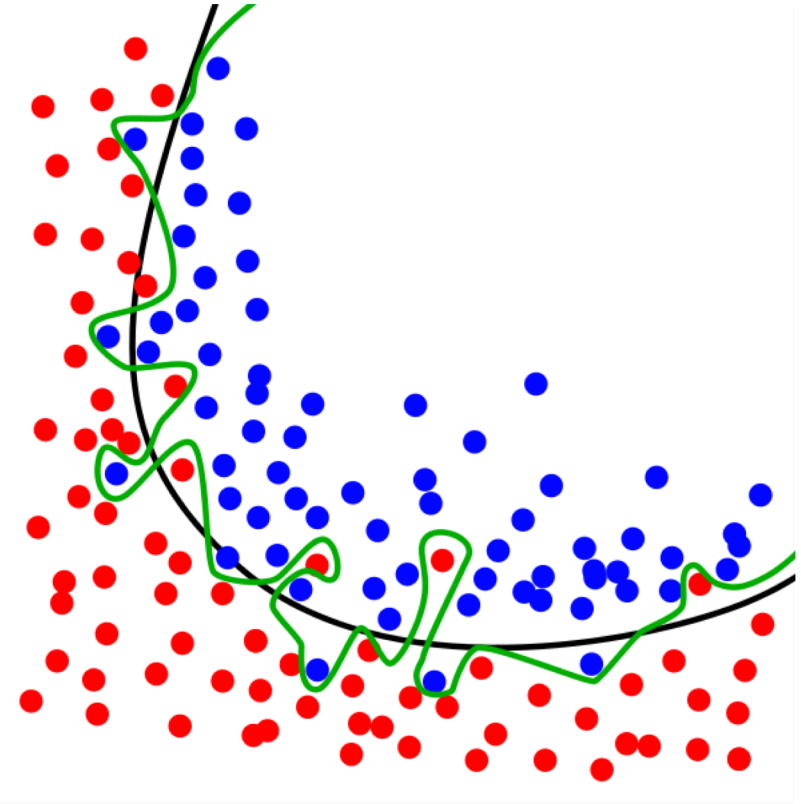
# Content IV

- Miscellaneous
  - Overtraining
  - Training best practice to avoid bias
    - splitting
    - Cross validation
  - Installation and environment
    - Anaconda
    - ROOT
    - Other software
  - Resources
    - Google Collab
  - What next
    - More PyTorch
    - More topologies

# Overtraining

## Every DL tutorial must cover overtraining

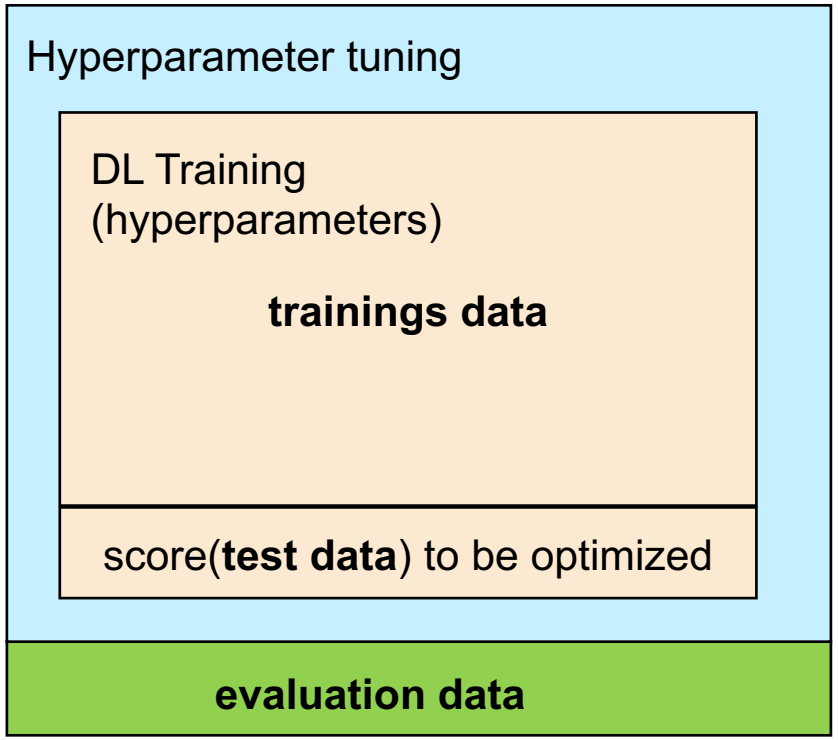
- With enough parameters, you can fit any data set
- Deep Learning models have a large number of parameters
- Overfitting refers to a model that models random details in the **training data**
- For a classifier the accuracy will be to high and the aleged quality cannot be reproduced on an independent dataset
  - Saturation in accuracy
  - Difference between **loss vs. epochs** trainings and test data, see tutorials



# Best practice

## The obvious rules that are not always obeyed

- Split your data:
  - *train, test* (e.g. 70/30)
    - Disadvantage: you can not use all your data for training
    - Cross validation may help
  - Better *train, test* and *evaluate*
    - Often we tune our models. The are many hyperparameters to be defines, e.g. learning rates, topology, algorithmic choices
    - This may adapt the procedure to the test data
    - Think about limit setting for a search: The data are blinded but the signal MC scan is often used for optimizing
  - There are tools for automatic hyperparameter tuning, e.g skopt



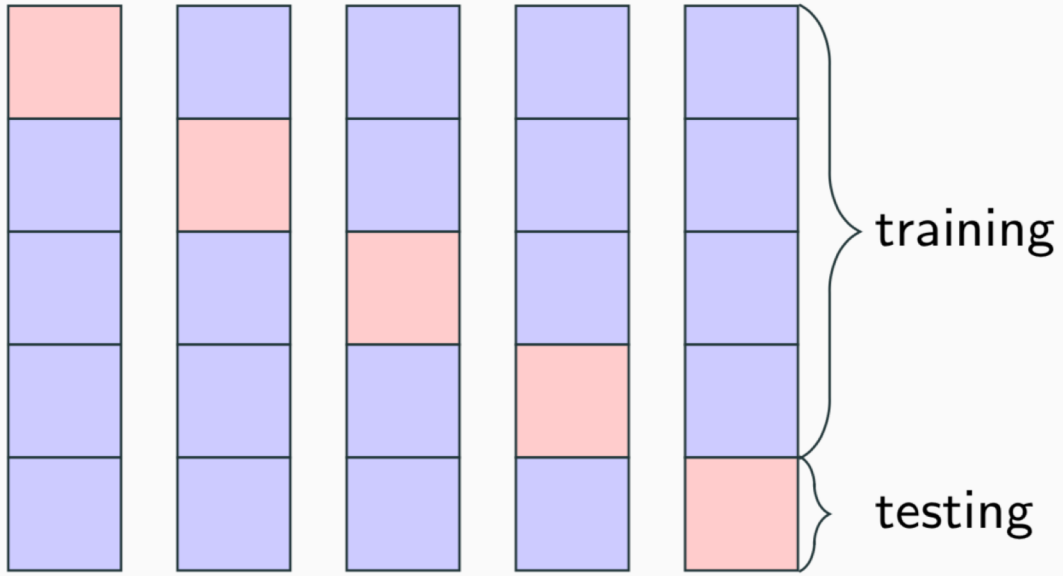

BTW normalize your data!

# Cross Validation

## Trading data for time

E.g. 5-fold cross validation

1. Split sample in 5 sub-samples
2. Train on 4/5th of the sample
3. Evaluate on remaining 5th
4. Repeat with changing roles
5. Evaluate final accuracy on $5 \times 1/5$



# Ranking the Importance of Your Features

## A simple method

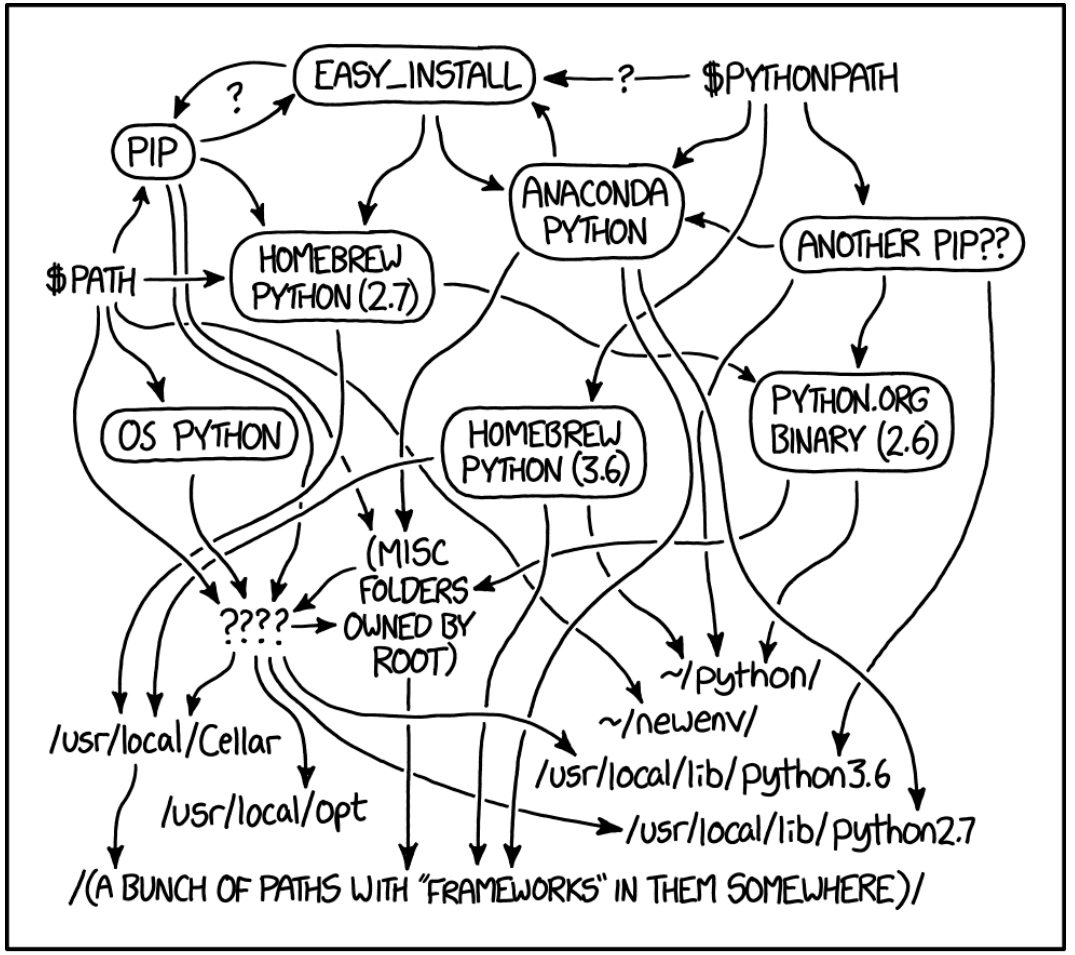
- NN can work with hundreds of input features
- While the networks often a tolerant against useless features it may improve the performance or at least simplifies the network if unimportant features are removed
- Just setting one feature to zero will have a negative impact on the performance
- Retraining with less features is time consuming
- A **simple strategy** to test the feature importance is
  - Randomize one column (feature) after the other in your test data
    - Apply the trained model to the modified data and calculate some appropriate score
- See tutorial

```
acc w/o   Pt1 0.9629
acc w/o    Pz1 0.5400
acc w/o     E1 0.9587
acc w/o   Phi1 0.9385
acc w/o  Eta1 0.5411
acc w/o   Pt2 0.9715
acc w/o    Pz2 0.5652
acc w/o     E2 0.9801
acc w/o   Phi2 0.9390
acc w/o  Eta2 0.5589
```

# How to install all that

## How to get things running

- If you start with deep learning you will need a large number of different interdependent python libraries. This often becomes messy.
- My advise is Anaconda
  - **Anaconda** is a free/open source distribution of Python (and R) software for scientific computing and machine learning
  - It comes with a package manager **conda**
    - **conda search blah, conda install blah, ...**
  - It allows to manage different **environments** in case your different projects have conflicting needs for software versions
  - It interacts with **pip** if the Anaconda repositories does not provide a certain software
  - It can be downloaded for Linux, Windows and Mac




MY PYTHON ENVIRONMENT HAS BECOME SO DEGRADED THAT MY LAPTOP HAS BEEN DECLARED A SUPERFUND SITE.

# ROOT

## I still believe that we as HEP etc. physicists need ROOT

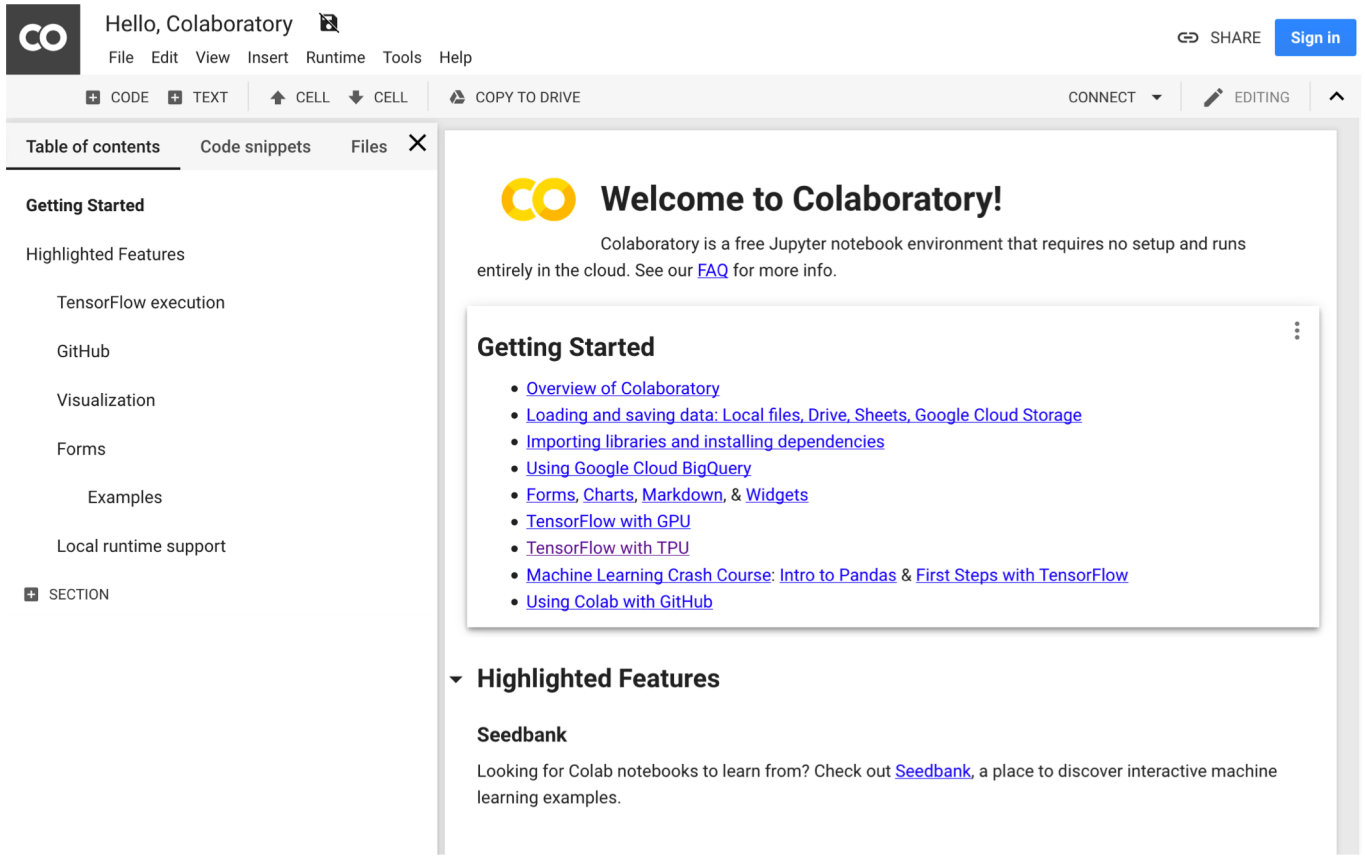

- **ROOT** is not well integrated into the Anaconda world.
  - There had been an effort from the **NLESC** for the XENON1T dark matter experiment which is a bit outdated but still working
- On Centos7 ROOT6 and Anaconda5 coexist well
  - But not on a Mac
- If you have a CERN account
  - and access to /**cvmfs** you can use the lcg software stacks e.g.
    source /cvmfs/sft.cern.ch/lcg/views/LCG_93c/x86_64-centos7-gcc62-opt/setup.sh
  - or use the same lcg package with **Swan** - a Jupyter server from CERN "Interactive analysis in the cloud"

- There a several libraries to bridge ROOT into the numpy/pandas data science world
- They *depend* on a ROOT installation
  - ROOT itself as pyROOT
  - **root_numpy**
  - **root_pandas**
- *Without* any ROOT installation **ROOT files** can be read by **uproot**

# Computing resources

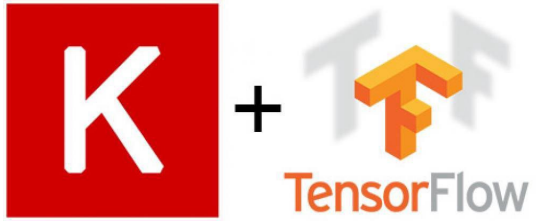
## Resources not related to a specific experiment or from your home institute

- Swan - if you have a CERN account
- **Goolgle Colab**
  - 1 **GPU** for free (12h)
  - **TPUs** with Tensorflow
  - Jupyter-style notebooks (slightly modified)
    - can be **shared** if you want to collaborate on a project
    - **pyTorch** can be installed from within the Colab notebook
- Amazon etc. you have to pay - we did this for you ( from your fees ☺ )

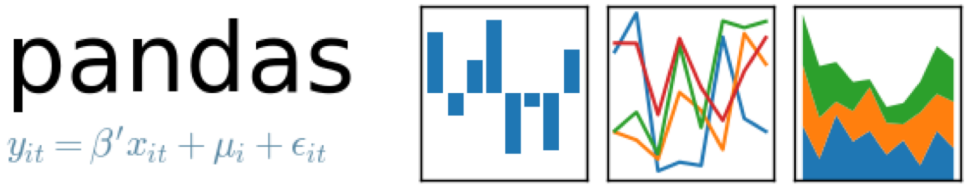


# Other useful software

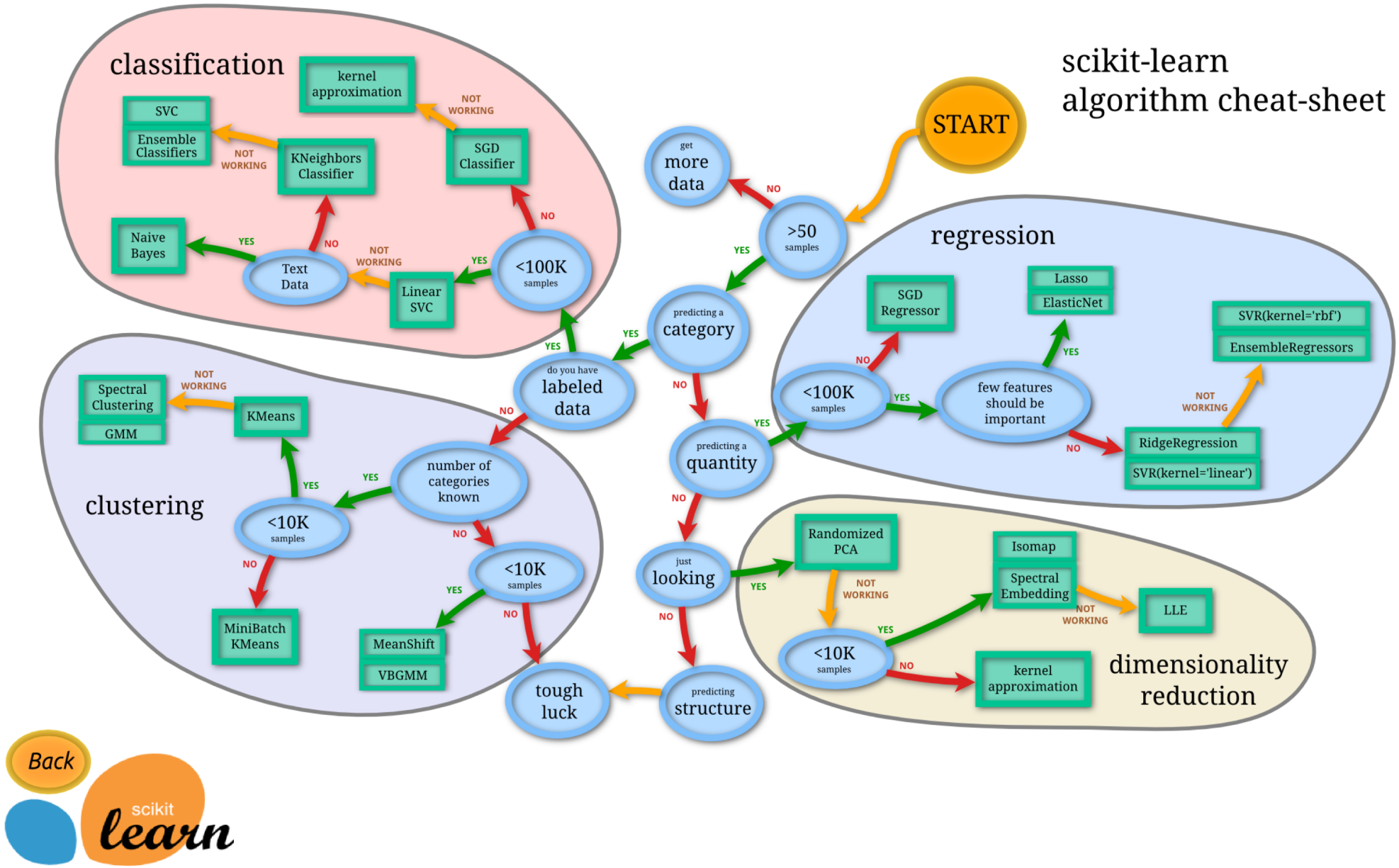
- Besides of Keras and Tensorflow
  - See tomorrow
  - Google has started to somewhat copy the PyTorch approach checkout the eager execution API
- Matplotlib & Seaborn
  - Plotting and visualizing
- Pandas
  - Convenient data mangling
- Sklearn aka Scikit-learn
  - Large collection of Machine Learning tools and algorithms






# Sklearn


scikit-learn
algorithm cheat-sheet
START
>50 samples
get more data
predicting a category
do you have labeled data
predicting a quantity
just looking
predicting structure
tough luck
<100K samples
Text Data
Naive Bayes
Linear SVC
SVC
Ensemble Classifiers
KNeighbors Classifier
kernel approximation
SGD Classifier
classification
<100K samples
SGD Regressor
Lasso
ElasticNet
few features should be important
SVR(kernel='rbf')
EnsembleRegressors
RidgeRegression
SVR(kernel='linear')
regression
number of categories known
<10K samples
KMeans
Spectral Clustering
GMM
MiniBatch KMeans
<10K samples
MeanShift
VBGMM
clustering
Randomized PCA
Isomap
Spectral Embedding
LLE
<10K samples
kernel approximation
dimensionality reduction
YES
NO
NOT WORKING
Back
scikit learn

# DL History

- **Deep Learning** break through
  - Several improvements in the last ~10 years,
    see Schmidthuber https://arxiv.org/abs/1404.7828
    - Better training of DNN (aka multilayer) NN
      - stochastic gradient descent -> RMSProp -> Adam
    - different topologies (CNN,RNN,LSTM,autoencoder)
    - new hardware - GPU/TPU
  - **DNN do not need feature engineering, they can work on raw data – at least as good as humans**
  - Nature com. by P. Baldi et al. (2014) "Searching for exotic particles in high-energy physics with deep learning"
  - Nature article by Yann LeCun et al. (2015) on Deep Learning
  - Nature article by Google people (2016), on AlphaGo
- Entering everyday applications: e.g.  Siri and Alexa
- The **Hype**: AlphaGo from Google beats leading human Go players

# AlphaGo

- GO – a game that cannot be solved by a search, combinatorial explosion
- AlphaGo Algorithm consist of 3 elements
  - Policy network, supervised learning,  13-Layer-CNN, finds the possible moves
  - Value network, reinforcement learning,  estimate position value
  - MC tree search,
    - intelligent, randomized brute force approach
- Hardware
  - At playtime: 1.920 CPUs,280 **GPUs** (2016, depending on game)
  - Training: Google custom made TPUs (Tensor Processing Unit)
    - High integer performance - first generation TPU is available on CoI
- Training
  - 30 million moves database of human games
  - playing against itself
- The 2017 version won 60:0 against human GO masters
  - One machine with specialized TPU
  - The project is now closed - problem solved
- Software: **Tensorflow**

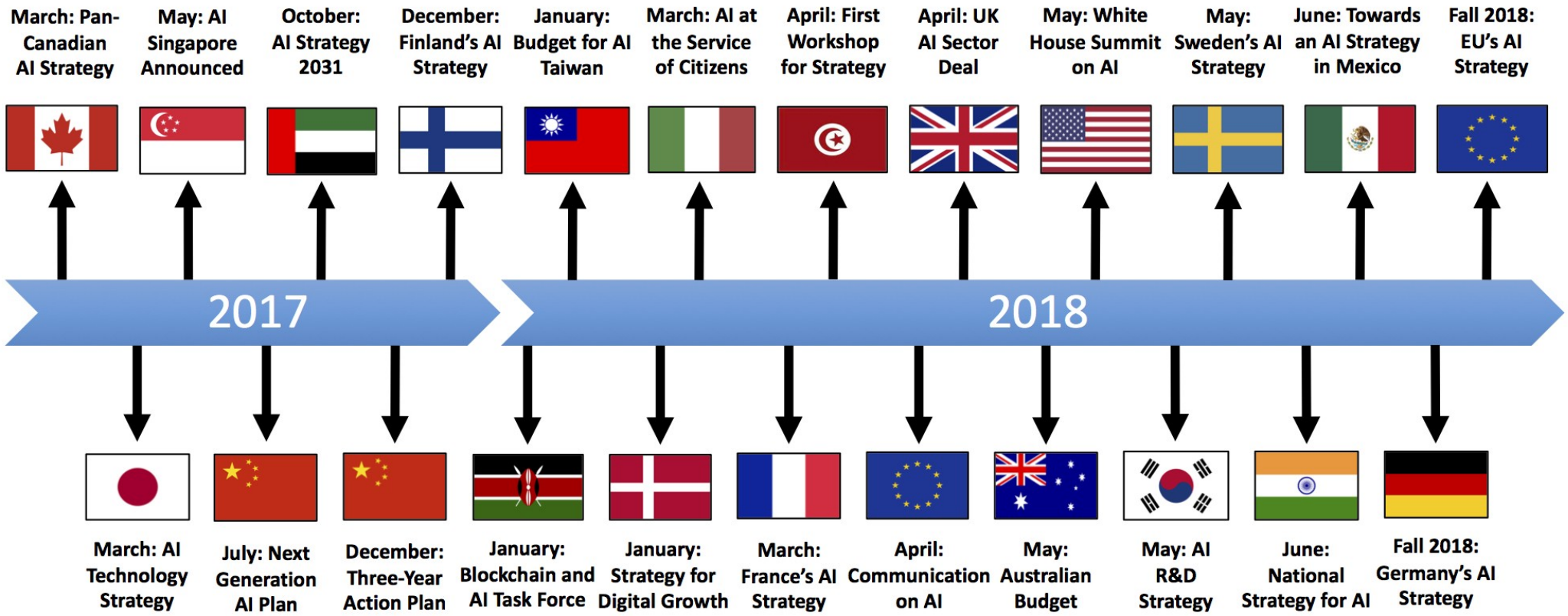
# AlphaGo

- Combination of different techniques,
  Google profits from
  - Improvements in techniques for deep NN training
    - Available (but algorithms are not invented by Google) -> Tensorflow
  - Specialized hardware
    - For us, with a limited budget, GPU
    - Or FPGA for trigger applications
  - Domain (playing Go) specific approach (tree search)
    - That we have to find out for our problems
      - That needs expertise – not a black box approach
    - AlphaGo would not have been the winner without this element
      - Deep Learning **does not solve** all problem
      - My guess, tracking in HEP will be solved by specialized algorithms, like the one we have always used
        Kaggle TrackML Particle Tracking Challenge

# Hype brings funding

- There is a big interest to support this new technologies by funding bodies relevant for Germany and beyond
  - Helmholtz
  - BMBF
  - DFG
  - EU
- Almost all academic call, proposal just now is talking about AI
  - It seems that any simple 2 parameter fit is sold as AI ☺
- In France Macron realized this early and decided to spent 1.5 billion euros
  https://www.wired.com/story/emmanuel-macron-talks-to-wired-about-frances-ai-strategy/

# Artificial Intelligence Strategies




https://medium.com/politics-ai/an-overview-of-national-ai-strategies-2a70ec6edfd

# In case you need more PyTorch

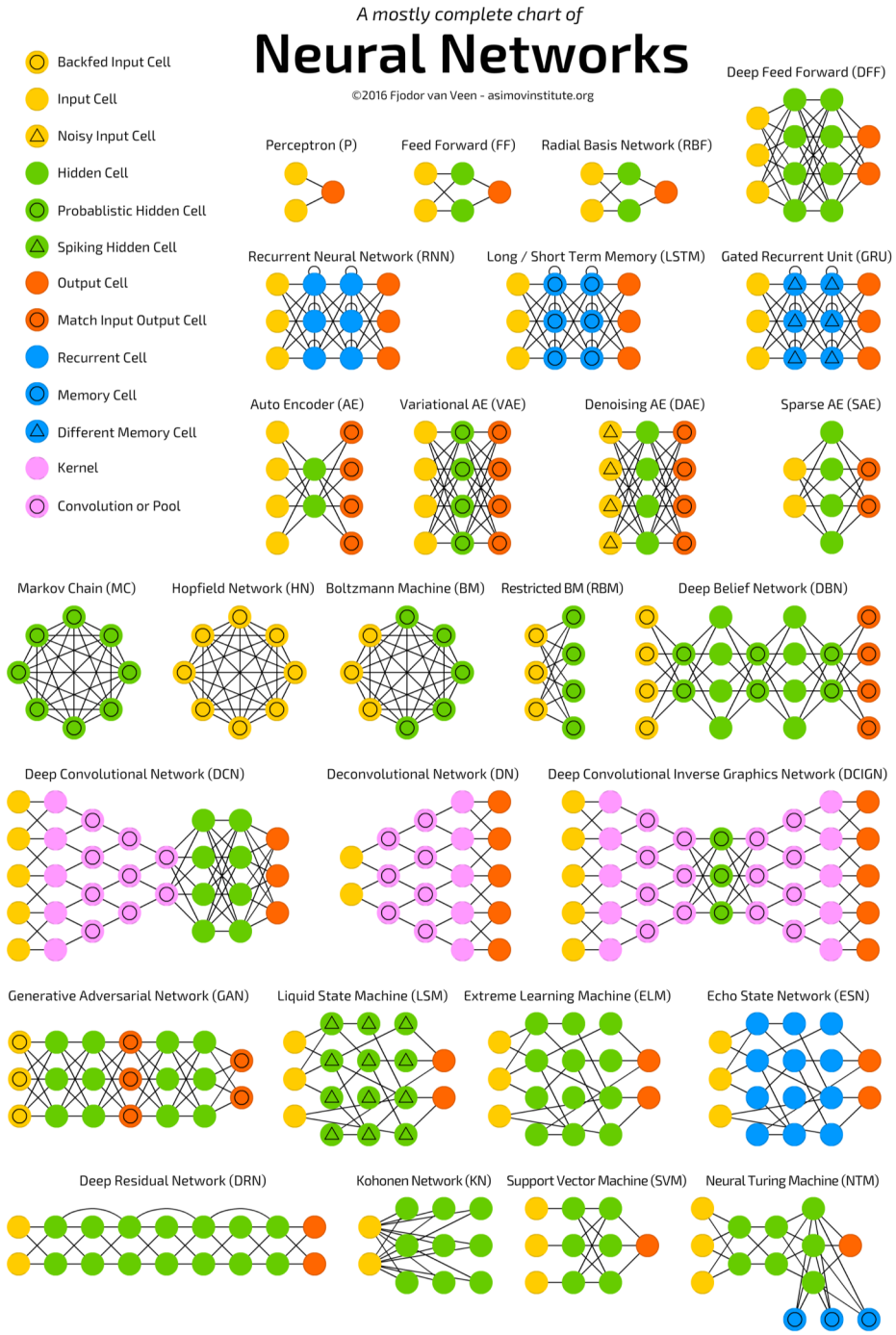
## PyTorch has several tutorials

- PyTorch tutorials and the more involved examples
- There is plenty material out there e.g. model-zoo
- We haven't discussed the details but you can checkout the style transfer tutorial from PyTorch to create *paintings similar to our poster.*
  - Style transfer tutorial and here
- The tutorials used here will be updated to PyTorch 1.0 when available and will go to a public github

# What we did not talk about

## See the following days

- I covered only simple dense networks
- This is just the starting point
  - There are dozens of different topologies/ideas
- I hope we can give you an idea about all this in the following days.
- https://www.deeplearningbook.org/



# Thank you

## Contact

DESY. Deutsches
Elektronen-Synchrotron

www.desy.de

Dirk Krücker
CMS
dirk.kruecker@desy.de