Belle II Starterkit MVA Tutorial

Belle II Starterkit

Wehle, Simon
BNL, 01.08.2019

Disclaimer

Personal Experience

- The presented recommendations originate purely from personal experiences, you might be of different opinion and are encouraged to share and discuss them

Context of this talk

- Focus on machine learning for classification
- Presentation of various methods
- Presentation of available tools
- The Belle II MVA package
- Personal recommendations

- **Additional resources**
- General lecture on multivariate analysis by Thomas Keck MultivariateClassificationLecture.pdf
  - (A lot of distribution plots in this talk are taken from this reference)

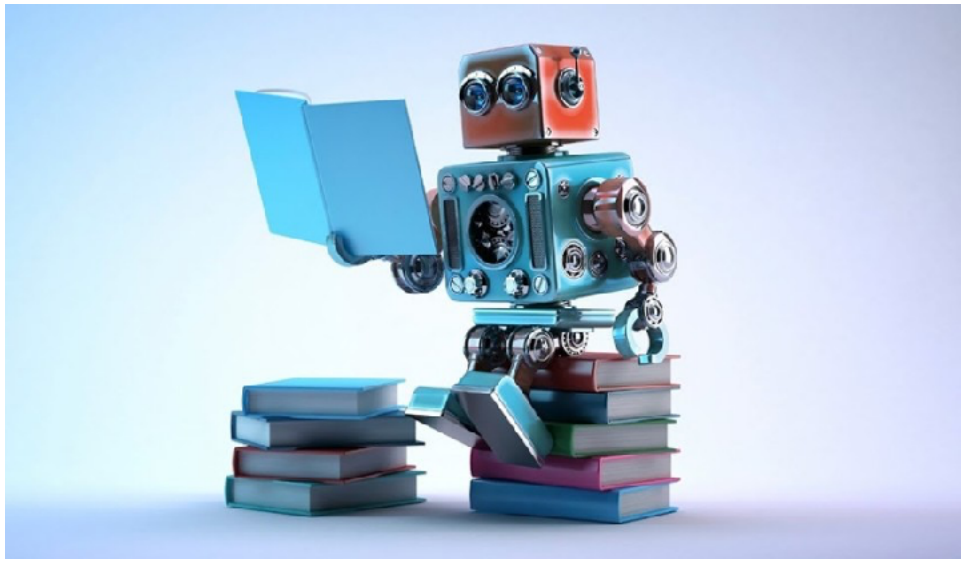
- Confluence, MVA package

- More on the packages to come

Impact of Machine Learning

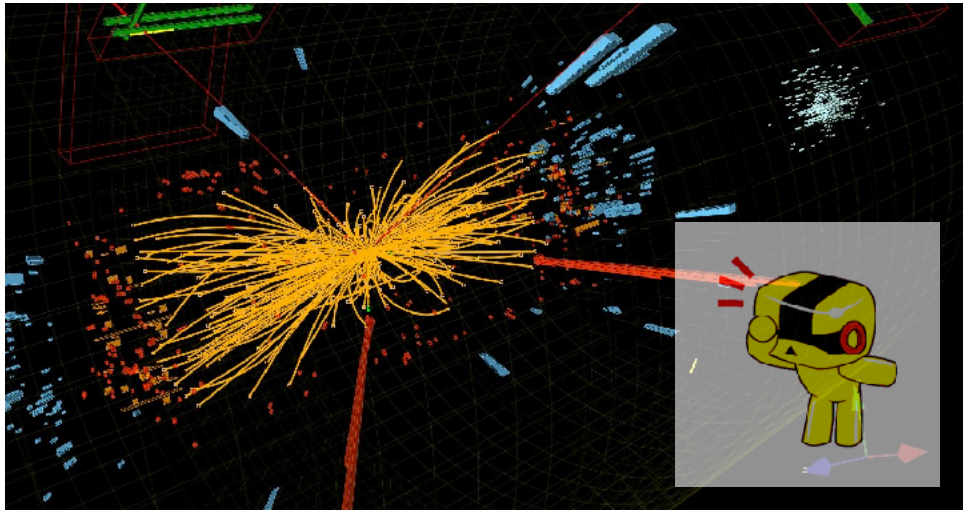
Era of data driven methods

- Huge boost in efficiency
- Rapid developments due to efforts of billion dollar companies
- Many open source machine learning platforms forming



Prospects for Physics

- Improvements due to machine learning can correspond to years of running an experiment
- In my opinion we might be just scratch at the beginning of what is possible for our field

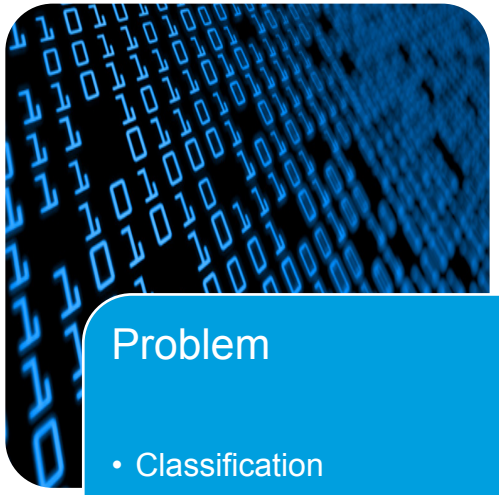


Multivariate Methods

Which and when to use

What is the Problem?

Where can we use machine learning?

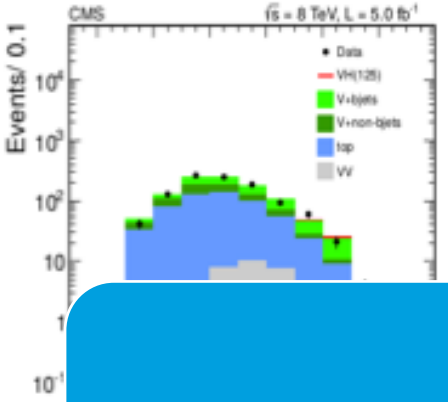


Problem

- Classification
- Regression
- Clustering

Solution

- Statistical methods
- Machine learning
  - Supervised (We have labeled data like a MC truth)
  - Unsupervised (Can be directly applied to data)
  - Reinforced Learning (Self learned strategy based on rewards)



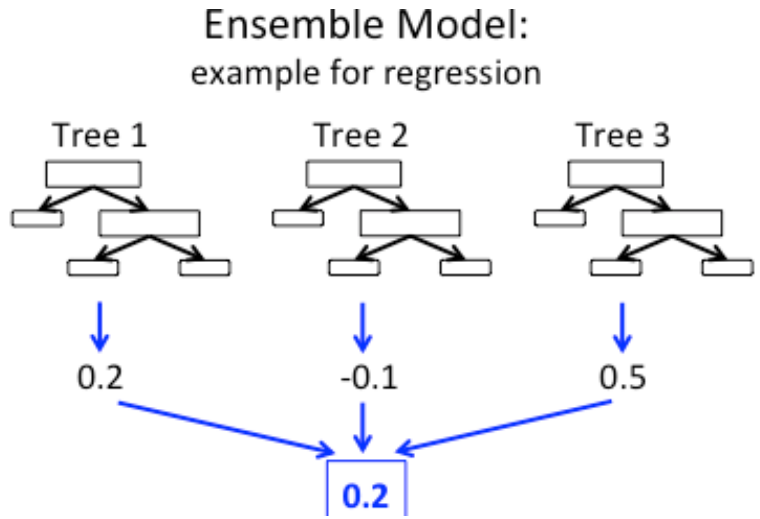
Result

Available Methods

Focusing on classification mainly

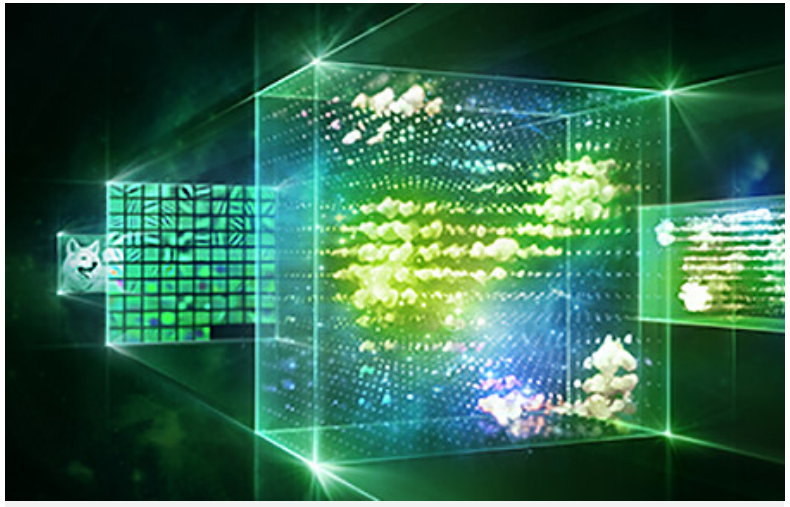
Conventional Machine Learning / Statistical Evaluation

- Many different methods available, I will list a few one which were/are used in physics
- LDA, SVM, Neural Networks, Bayesian Methods, Nearest Neighbors, Decision Trees, Boosting... (but there are of course many more)



Deep Learning

- Very popular machine learning method right now
- Has proven to be extremely efficient and powerful
- Many different techniques, Feed forward DNN, Convolutional Neural Network, Recurrent Neural Networks, LSTMs …

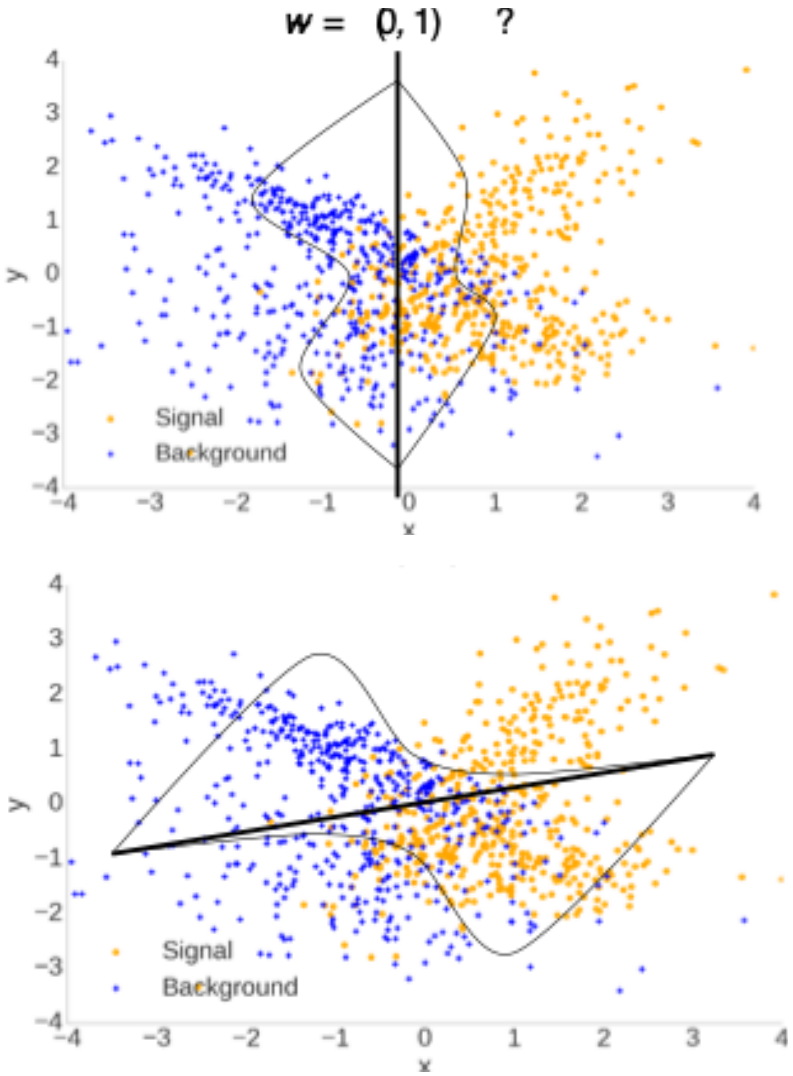


LDA – Linear Discriminant Analysis

Apparently do not call it Fisher's discriminant anymore

Linear Discriminant Analysis

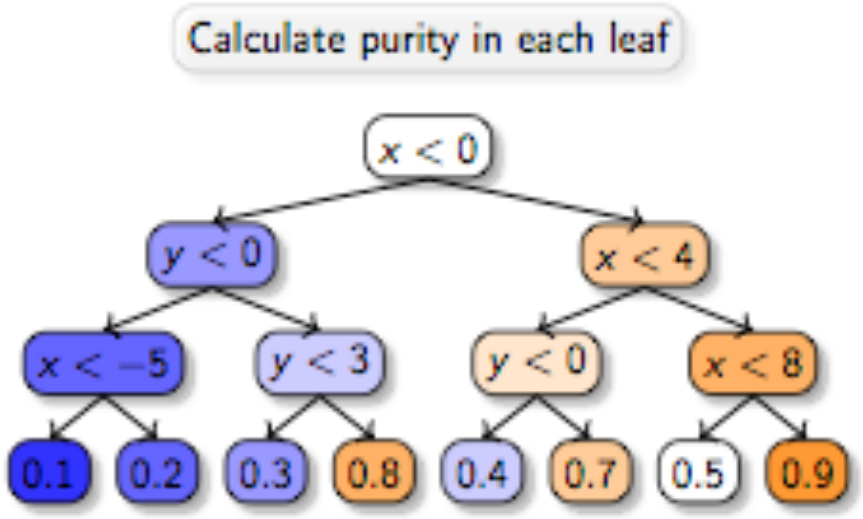
- Commonly used under the name Fisher's Discriminant
- Only requires mean and covariance of the sample
- Can be expanded to Quadratic Discriminant Analysis

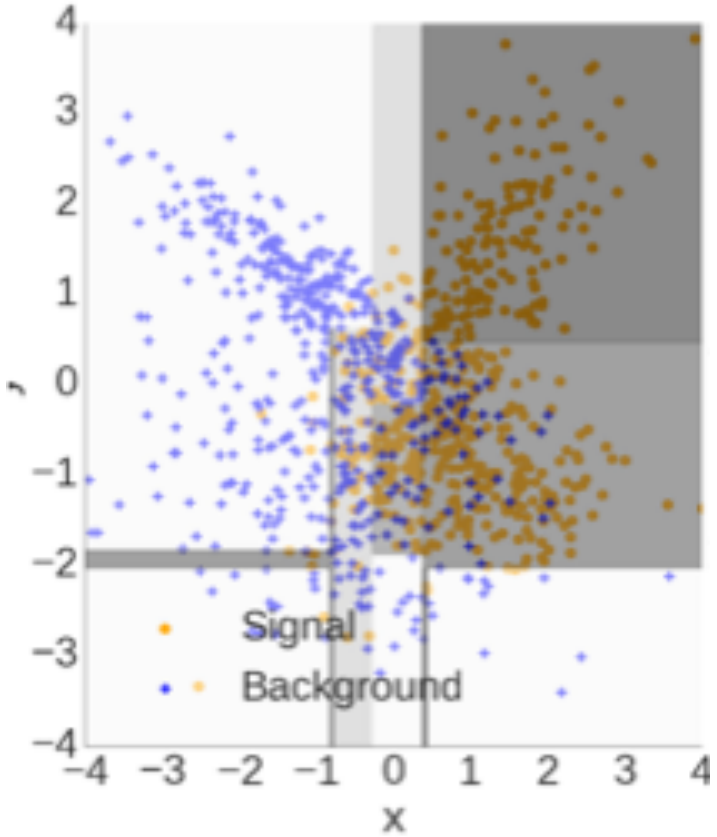


Decision Trees

Decision Trees

- Simple to comprehend
- Easy to train and implement
- Reasonably powerful





Boosted Decision Trees

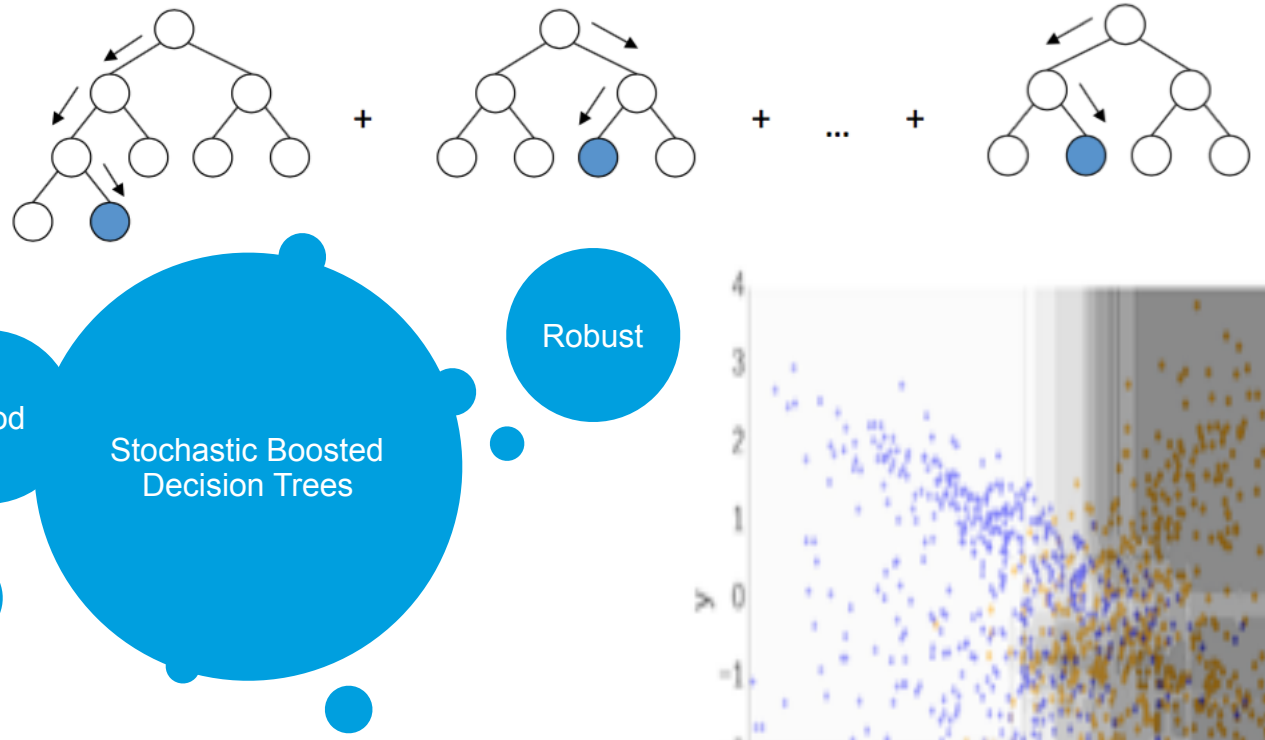
Very good

Boosting

- Use may ”weak” classifiers
- Reweight events according to prediction of previous tree(s) for the next tree to focus on falsely classified events

Bagging

- Use only a fraction of events in each tree
- Robustness against statistical fluctuations



Stochastic Gradient Decision Trees

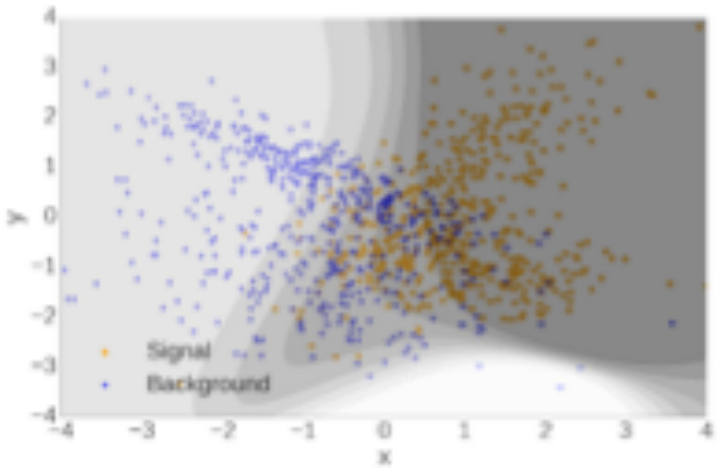
- Widely used in HEP
- Robustness against overfitting
- Out of the box very good performance

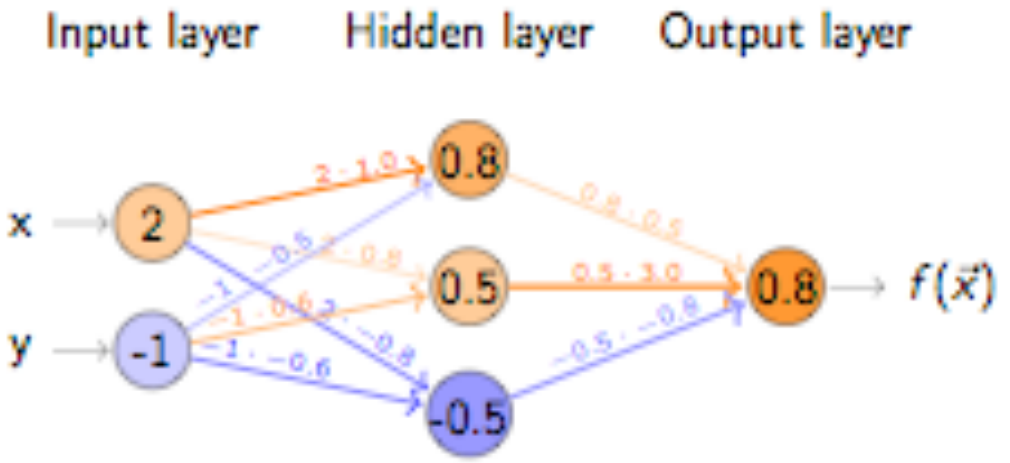
Artificial Neural Networks (ANN)

Artificial Neural Networks

- Attempt to “simulate” neurons in a brain
- Usually a feed-forward approach, where data flows from input over hidden layers to the output layer
- Nice visualizations made by google to play around with neural networks in the browser:
  - http://playground.tensorflow.org

- ANN without hidden layer is similar to a LDA
- A neural network with one hidden layer can can approximate continuous functions on compact subsets of R^n



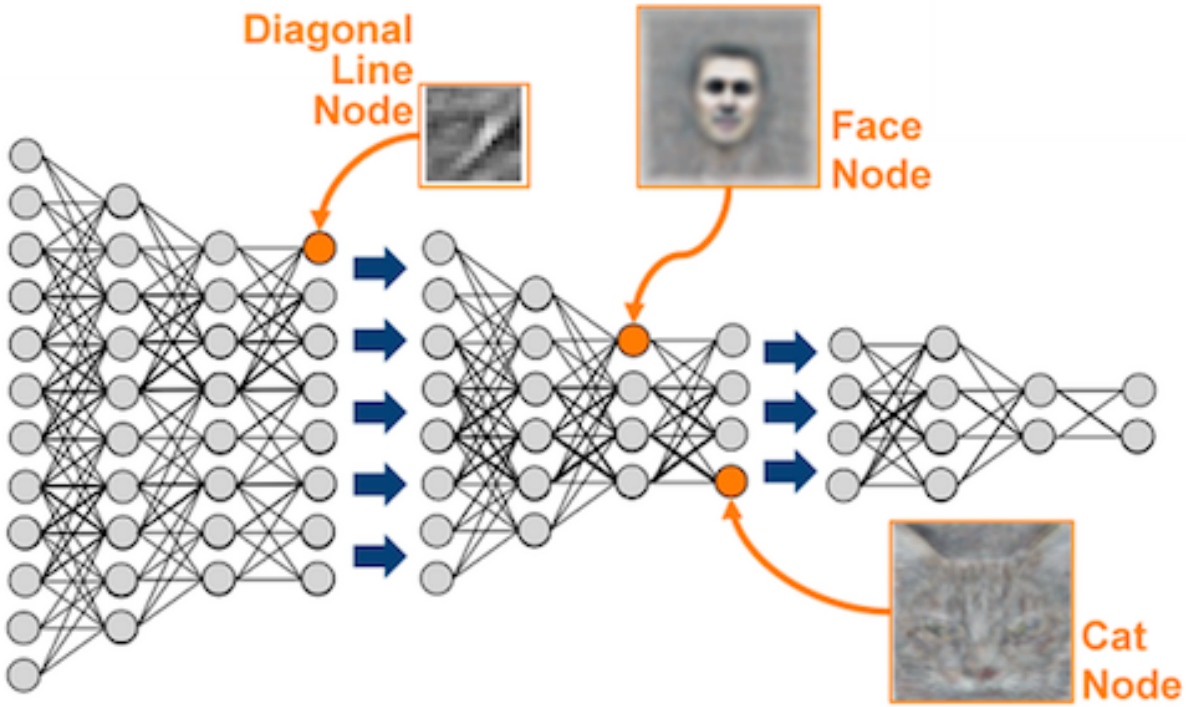


Deep Neural Networks (DNN)

The glorious future

Deep Learning

- Basically a large, mostly feed-forward ANN with many hidden layers
- Progress in computing power made backpropagation algorithm possible (mostly on gpus)
- Can/should be fed with “low level” features
- Benefit of DNNs: They can learn high level features themselves
- Idea: Give raw data, they figure out how to do a vertex fit or constrain invariant masses.. (okay it does unfortunately not work so easy for us)
- The input data has to be adequately complex or the danger of overfitting becomes imminent!

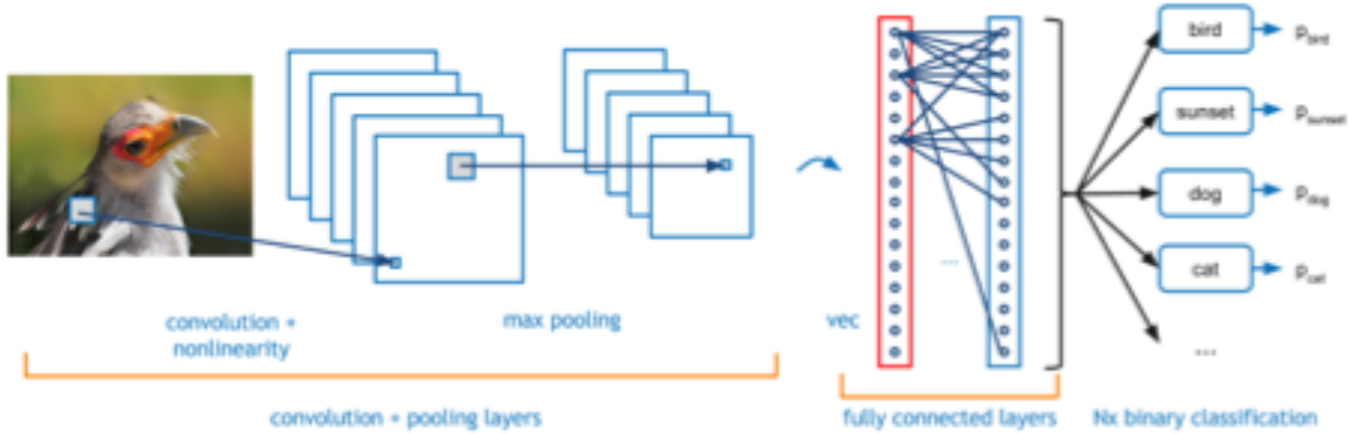


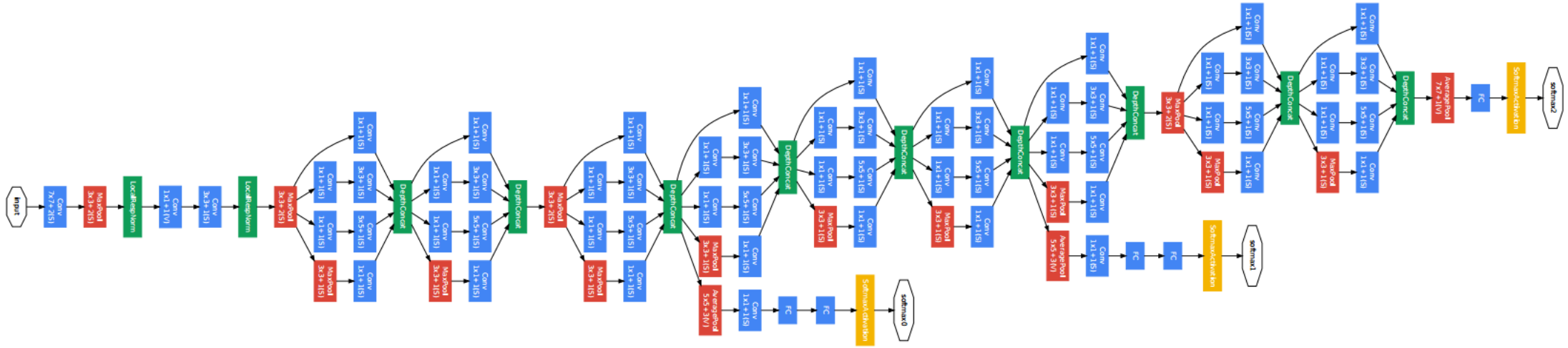
Convolutional Neural Networks (DNN)

The glorious future

Convolutional Neural Networks

- Used for image recognition
- Profit extremely from parallelization in gpus





Implementation Overview

A few notable implementations

ROOT TMVA

- Features many different classifiers
- Ability to perform preprocessing
- Nice overview and comparison of methods after training
- https://root.cern.ch/tmva

Scikit-Learn

- Python (numpy) based framework
- Many classification, regression, clustering and preprocessing tools are available
- http://scikit-learn.org/

Keras

- Python based “meta” framework for deep learning
- Can use various implementations as backend, like tensorflow or theano
- Very simple syntax
- keras.io

FastBDT

- Used often in the Belle II software
- Extremely(!) fast and robust
- https://github.com/thomaskeck/FastBDT

Other notable candidates

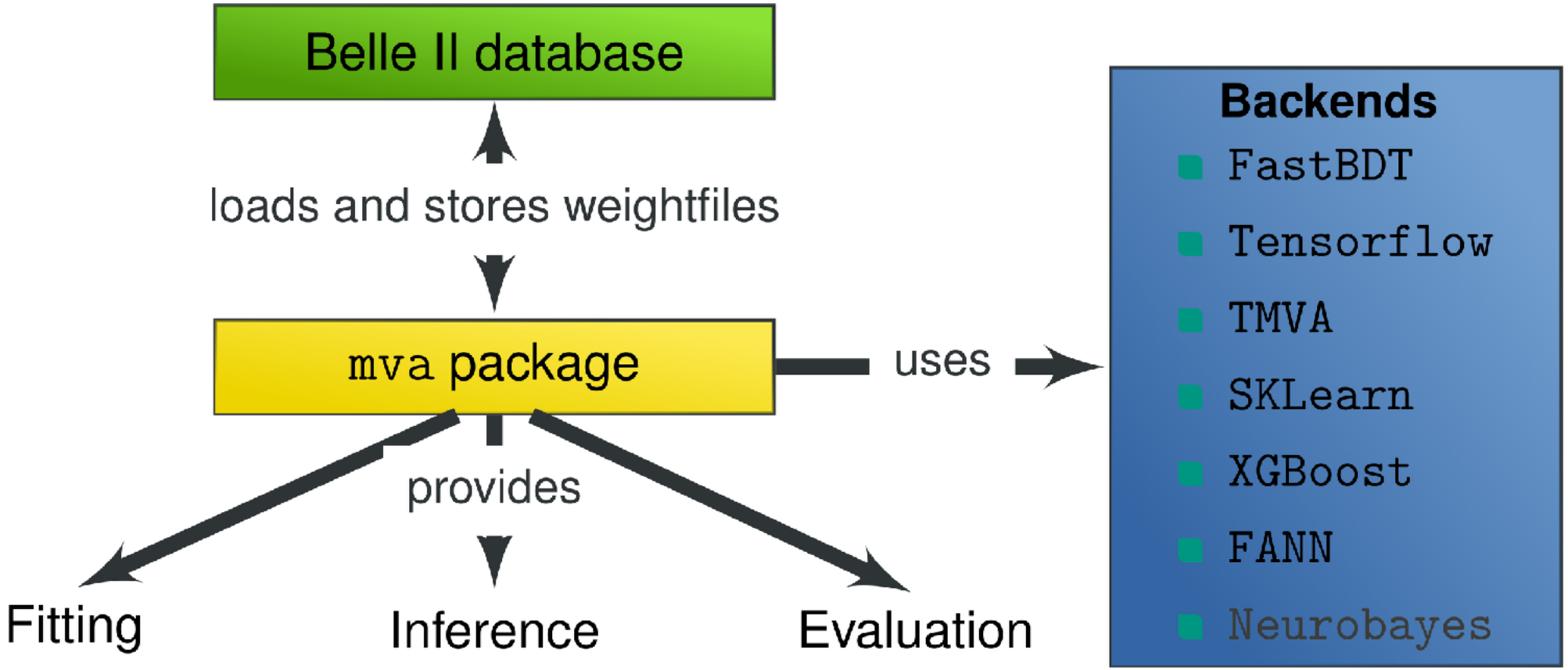
- **FANN**
  - Fast Artificial Neural Network
  - http://fann.sourceforge.net/fann.pdf
- **XGBoost**
  - Industry standard GBDT
- **Tensorflow**
  - Generic library for large matrix/ tensor operation
  - Optimized for gpu usage
  - https://www.tensorflow.org/
- **NeuroBayes**

Belle II MVA Package

Overview of the package

Belle II MVA Package Overview

Subheading, optional



Belle II MVA Package Overview

Goals and existing use cases

Main goals

- The mva package was introduced to provide:
  - Provides tools to integrate mva methods in BASF2
  - Collection of examples for basic and advanced mva usages
  - Backend independent evaluation and validation tools
- The mva package is NOT:
  - Yet another mva framework → Avoids reimplementing existing functionality
  - A wrapper around existing mva frameworks → Tries to avoid artificial restrictions

Use cases

- Analysis: Full Event Interpretation, over 100 classifiers have to be trained without user interaction and have to be loaded from the database automatically if a user wants to apply the FEI to MC or data
- Analysis: Flavor Tagger
- Analysis: Continuum Suppression
- Tracking: Track finding in CDC and VXD, classifiers should be retrained automatically using the newest MC, maybe even run-dependent, and automatically loaded and applied on MC and data during the reconstruction phase
- ECL: Classification and regression tasks
- BKLM: KLong ID

Belle II MVA – Setting up a Training

Subheading, optional

Interface

- **Fitting and Inference**
  - basf2_mva_teacher
  - basf2_mva_expert
- **Condition database**
  - basf2_mva_upload
  - basf2_mva_download
  - basf2_mva_available
- **Evaluation**
  - basf2_mva_evaluate.py
  - basf2_mva_info
  - basf2_mva_extract

GlobalOptions

```
import basf2_mva
go = basf2_mva.GeneralOptions()
go.m_datafiles = basf2_mva.vector("train.root")
go.m_treename = "tree"
go.m_identifier = "Identifier"
go.m_variables = basf2_mva.vector('p', 'pz', 'M')
go.m_target_variable = "isSignal"
```

SpecificOptions

```
sp = basf2_mva.FastBDTOptions()
sp.m_nTrees = 100
sp.m_shrinkage = 0.2
fastbdt_options.m_nLevels = 3
```

Belle II MVA – How to Perform a Training

Perform a training using basf_mva

Training in Python

```
import basf2_mva

go = basf2_mva.GeneralOptions()
go.m_datafiles =
basf2_mva.vector("train.root")
go.m_treename = "tree"
go.m_identifier = "DatabaseIdentifier"
go.m_variables =
basf2_mva.vector('p', 'pz', 'M')
go.m_target_variable = "isSignal"

sp = basf2_mva.FastBDTOptions()

basf2_mva.teacher(go, sp)
```

Training in Shell

```
basf2_mva_teacher --datafiles train.root
                  --treename tree
                  --identifier \
                       DatabaseIdentifier
                  --variables p pz M
                  --target_variable
isSignal
                  --method FastBDT
```

```
basf2_mva_teacher –help

basf2_mva_teacher –method FastBDT –help
```

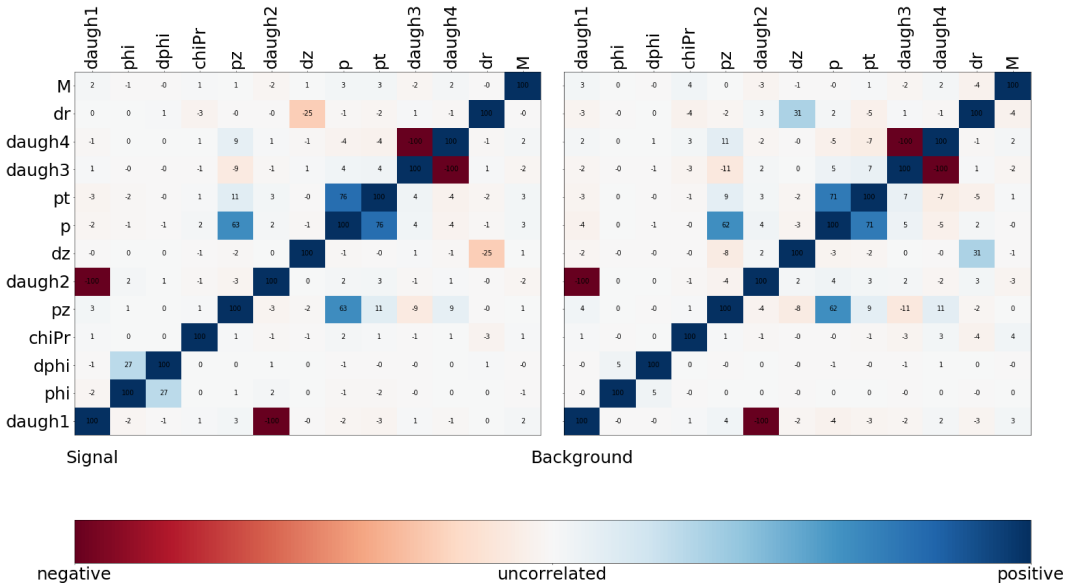
Belle II MVA – Evaluate the Training

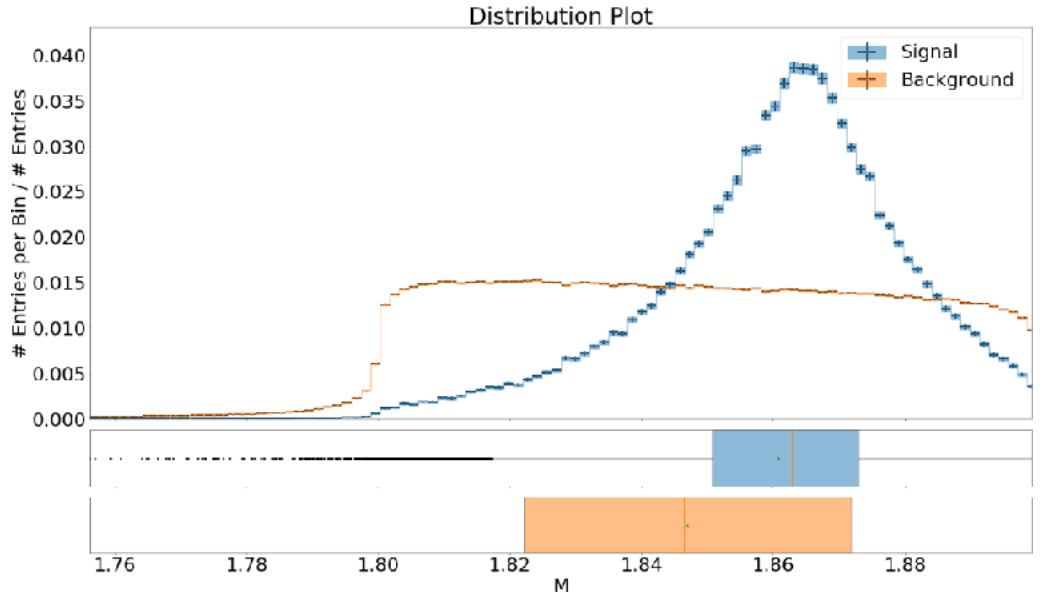
Apply method in basf2

Validate the training on command line

```
basf2_mva_evaluate.py -id
DatabaseIdentifier
                      -train train.root
                      -data test.root
                      -o validation.pdf
```

2.2 Correlation





Belle II MVA – How to apply a Training

Apply method in basf2

Apply method in basf2

```
path.add_module('MVAExpert',
                listNames=['D0'],
                extraInfoName='Test',
                identifier='DatabaseIdentifier')
```

Belle II MVA – Applying Expertise to Data

Apply method in shell

```
basf2_mva_expert --identifiers DatabaseIdentifier
                 --datafiles test.root
                 --treename tree
                 --outputfile expert.root
```

Apply method in python

```
import basf2_mva

basf2_mva.expert(basf2_mva.vector('DatabaseIdentifier'),
                 basf2_mva.vector('test.root'),
                'tree', 'expert.root')
```

Belle II MVA – Further Information

Please consider further information on confluence and the examples

$BELLE2_RELEASE_DIR/mva/examples/

```
[$ ls *
advanced:
bayesian_optimization_multicore.py  builtin_splot.py  performance_comparison.py  purity_transformation.py
bayesian_optimization.py            grid_search.py    plotting.py                variable_importance.py

basics:
application_in_basf2.py  create_data_sample.py  usage_in_python.py  usage_in_shell.sh

data_driven:
baseline.py  builtin_reweighting.py  builtin_sideband_substraction.py  builtin_splot.py  create_data_sample_example.py  custom_reweighting.py

keras:
adversary_network.py  keras_to_weightfile.py  preprocessing.py  relational_network.py  simple_deep.py

orthogonal_discriminators:
fastbdt_ugboost.py  hep_ml_ugboost.py  tensorflow_adversary.py

python:
hep_ml_uboost.py               howto_wrap_your_existing_training_into_a_weightfile.py  sklearn_mlpclassifier.py
howto_use_arbitrary_methods.py  sklearn_default.py                                     xgboost_default.py

tensorflow:
dplot.py  multithreaded.py  relations.py  simple_deep.py  simple.py  using_tfdeploy.py

tmva:
shell.sh  tmva_bdt.py  tmva_nn.py
```

MVA Package on Confluence

Machine Learning Good Practice

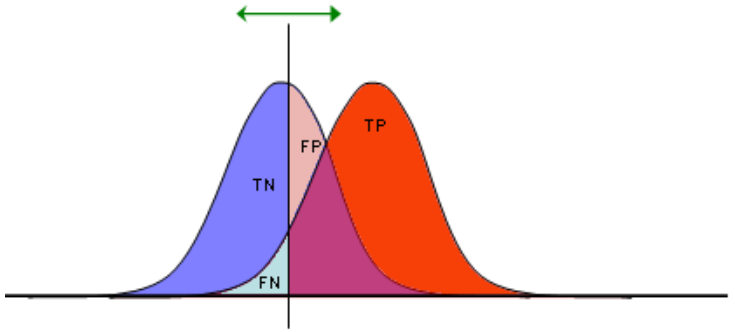
(Pure personal recommendations)

Evaluate your Training!

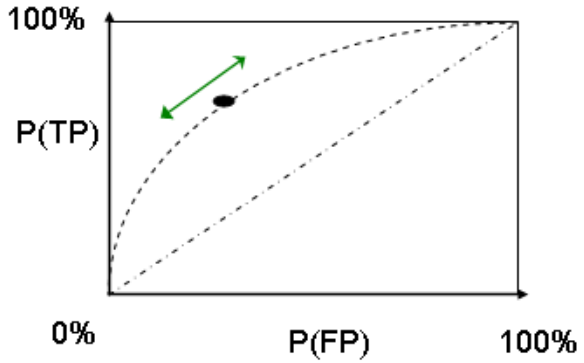
One or the most important aspects!

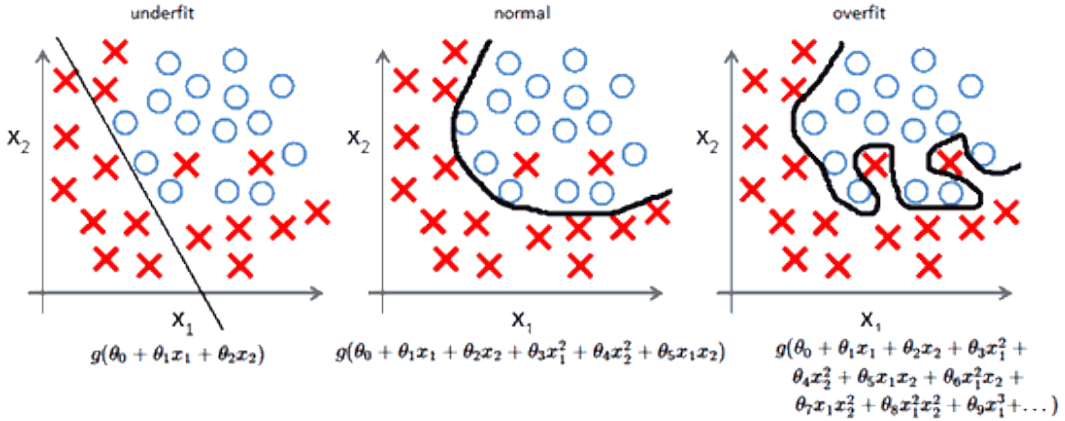
How to evaluate a training

- One can compare trainings and different methods on the same data with the ROC curve
- Often (but not always) a training with higher area under the curve (AUC) performs better
- Compare both training data and test data performance to spot overfitting



TP	FP
FN	TN
1	1





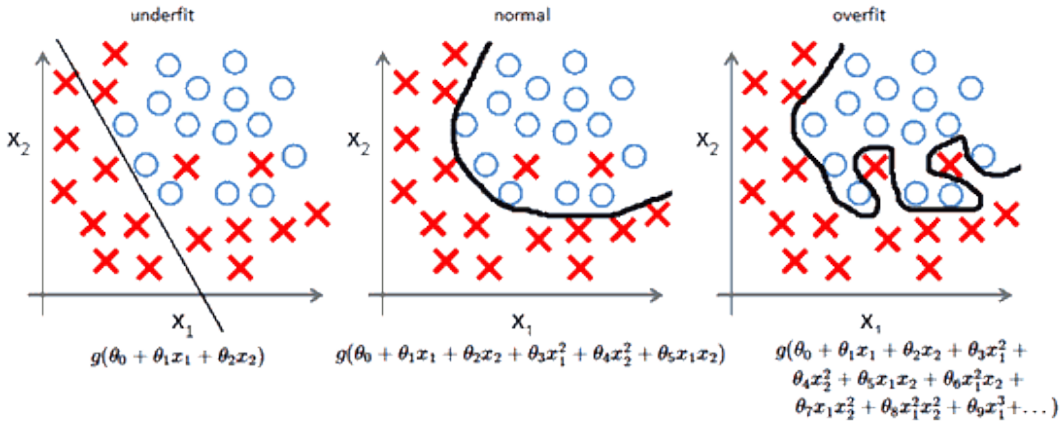
Always use an independent control sample !

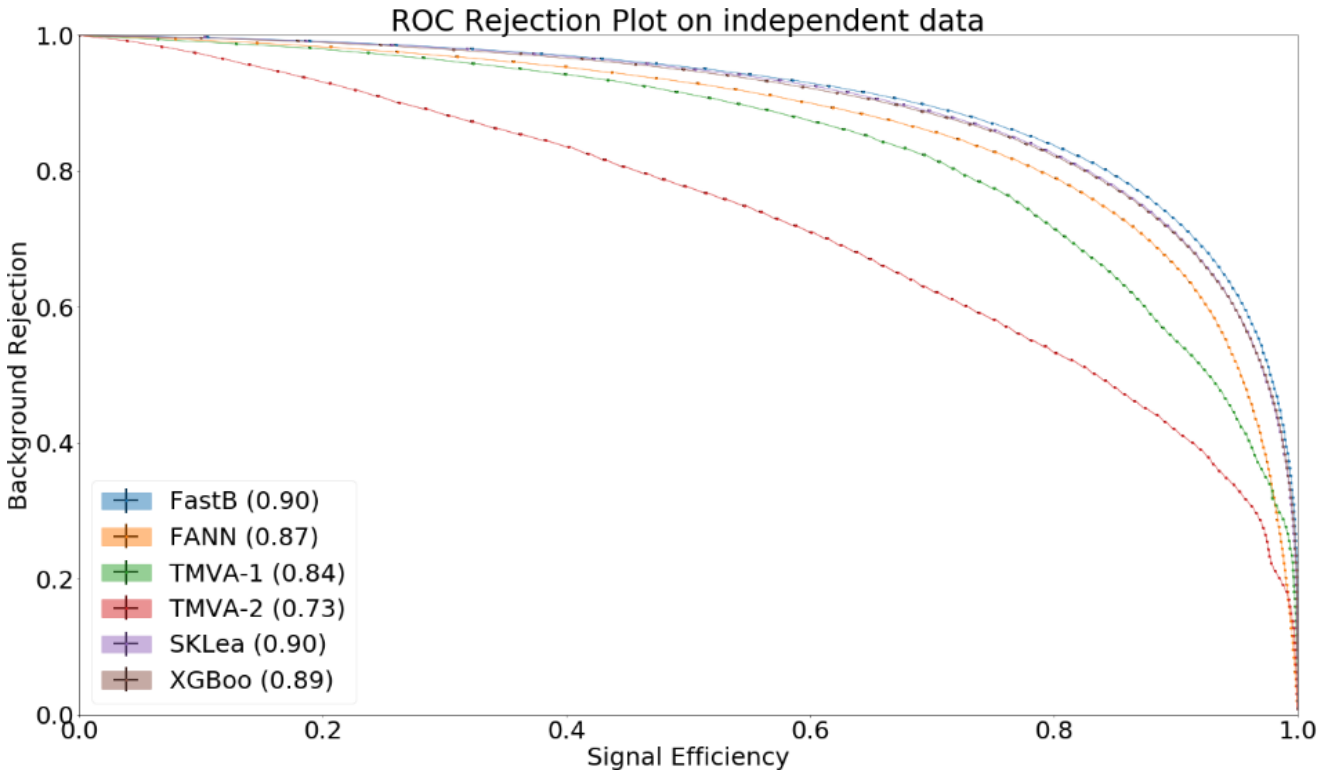
Evaluate your Training!

One or the most important aspects!

How to evaluate a training

- One can compare trainings and different methods on the same data with the ROC curve
- Often (but not always) a training with higher area under the curve (AUC) performs better
- Compare both training data and test data performance to spot overfitting





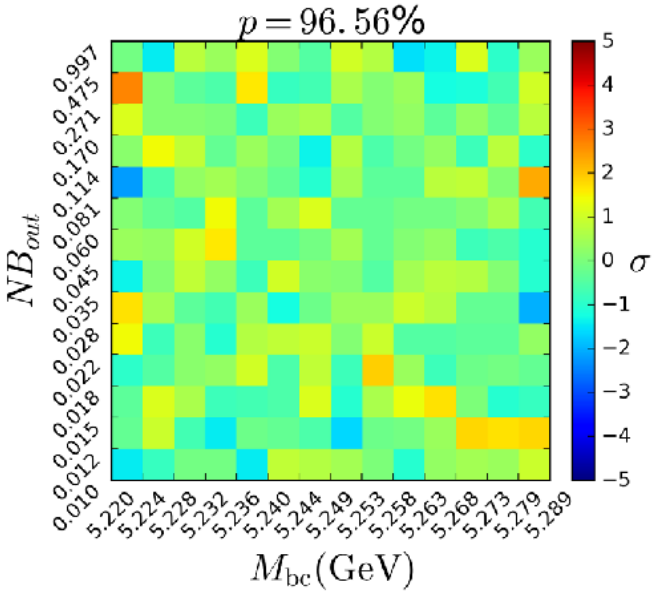
Always use an independent control sample !

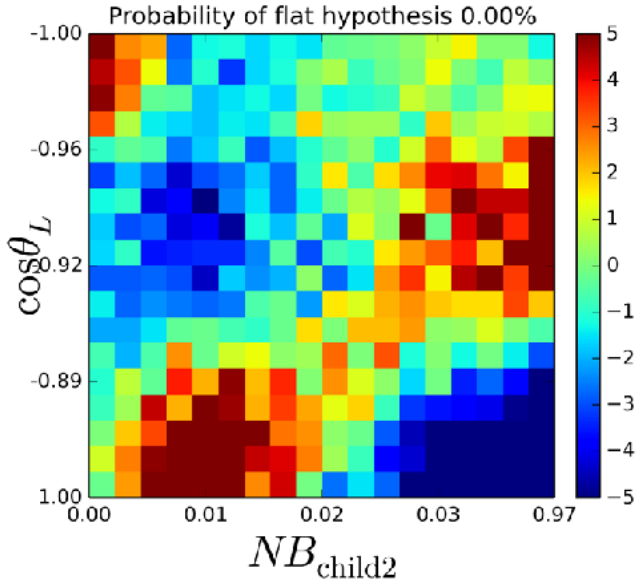
Monitor your Target

Also one or the most important aspects!

Monitor correlations to your target

- **Correlations can bias your selections**
  - Background in target variable can become peaking
- **Distributions can be biased towards the trained model**
  - Important for instance in angular analyses



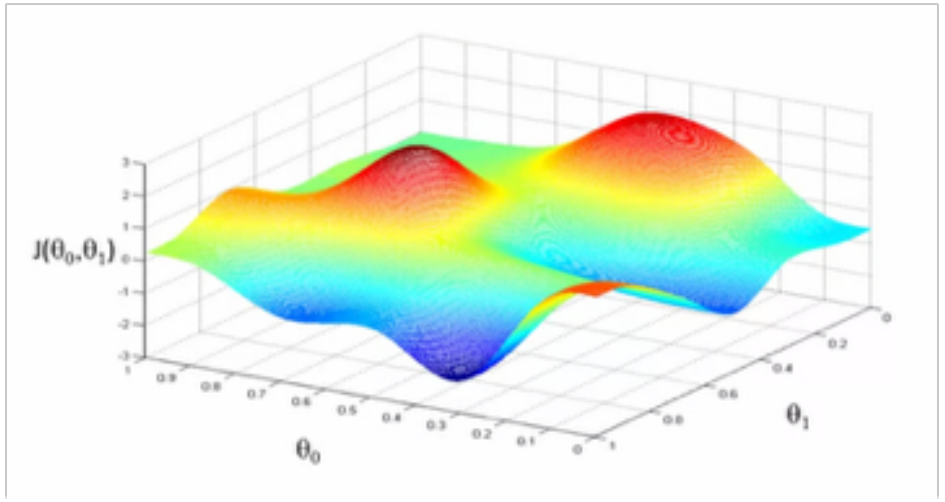


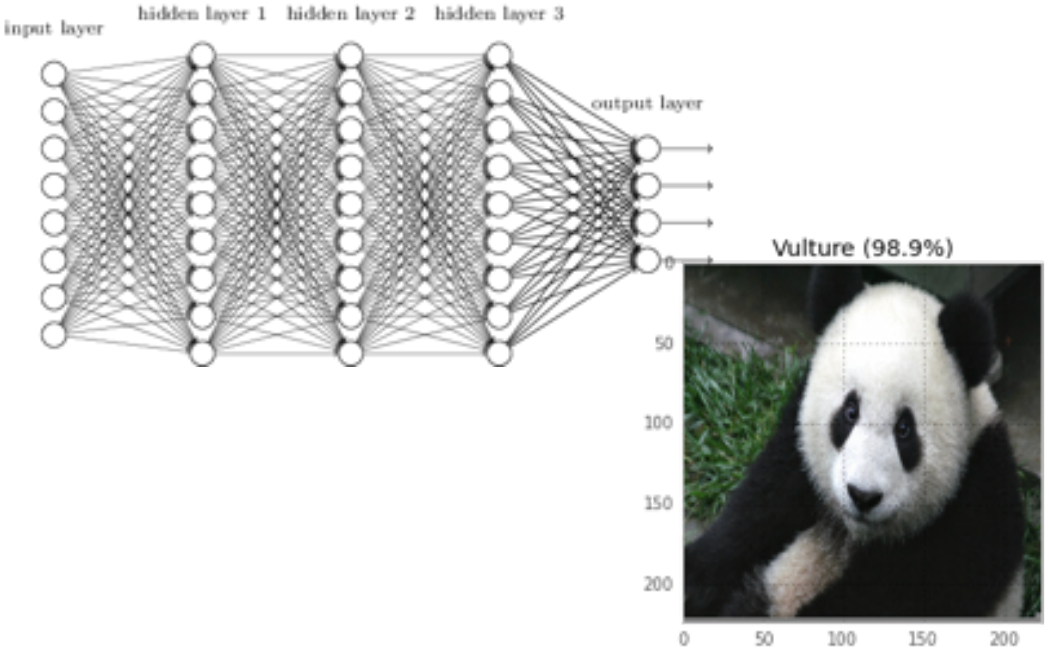
Do not use Neural Networks…

… if you don't know what you are doing.

Difficulties with training a neural network

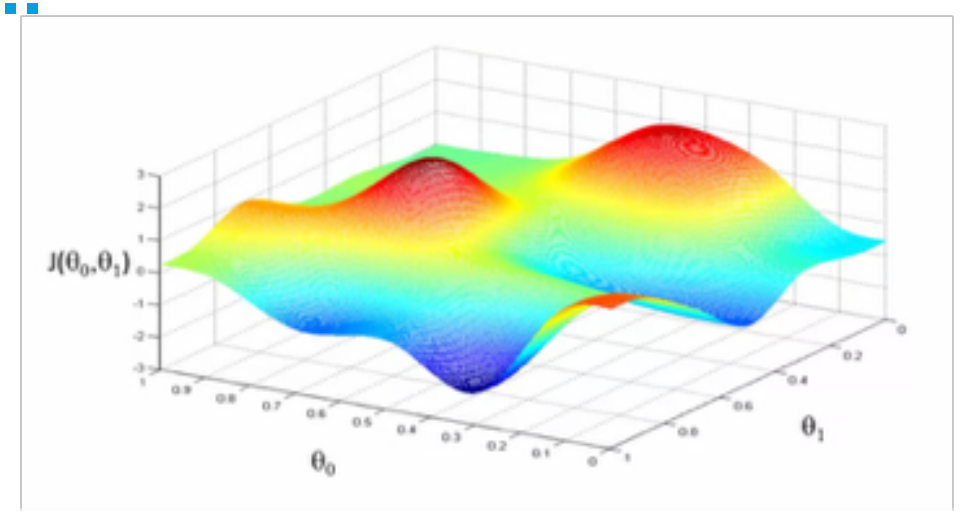
- For a robust training, the data should be preprocessed
- Normalized, Gaussian distributions as input help the network to converge faster
- Ideally, also correlation among the input variables can be a concern, some methods perform a de-correlation of the input vectors
- The layout of the network is not trivial, also the learning rate and activation functions can be tuned according to the problem
- Most of the time a Gradient Boosted Decision Tree with default parameters outperforms a neural network in a classification task out of the box..





Do not use Deep Learning Models..

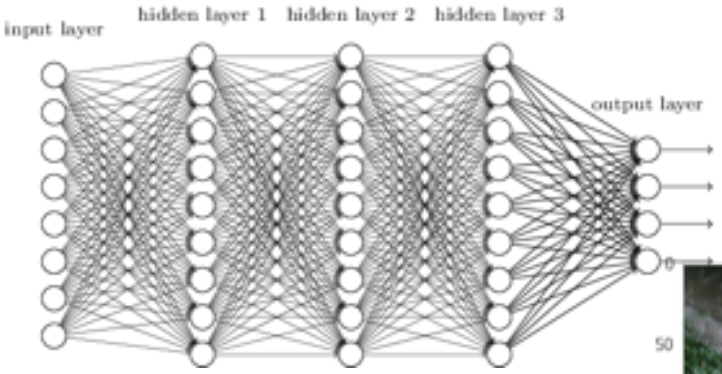
… if your data/problem is not appropriate.

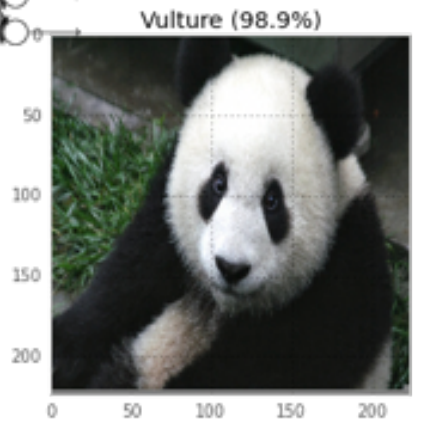


When to use deep leaning

- If the data is sufficiently complex
- If the dataset is large enough

- **Number of Degrees of freedom (NDF) of the model(≈number of parameters)**
  - DNN can have millions of free parameters
  - What should the model learn?
  - Can’t a decision tree with O(1000) NDF learn the essential properties?





- Use deep learning on low level features instead of high level features
- The real strength of DL is the ability to create features which are relevant internally

Dangers of Deep Learning

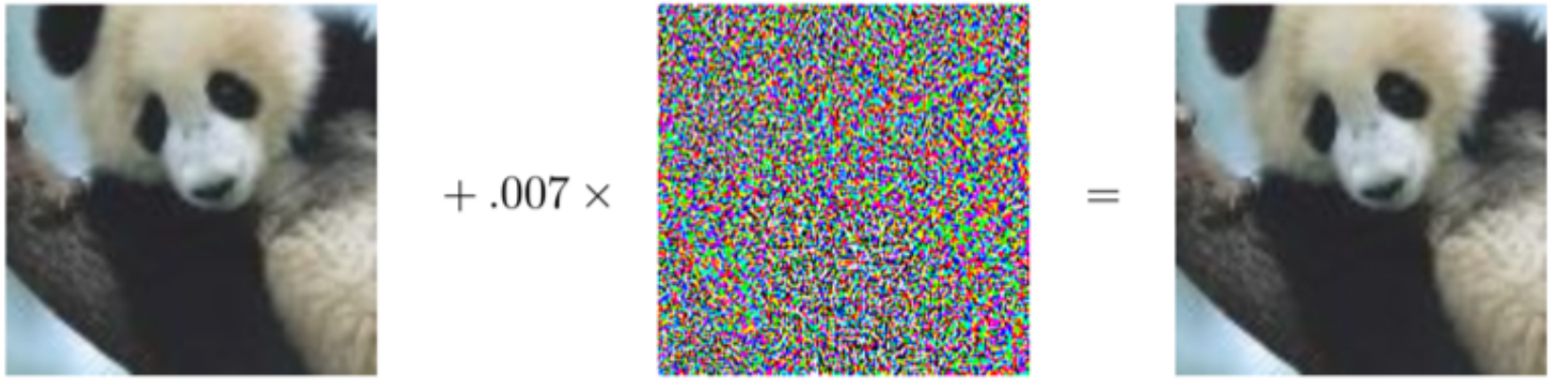
Difficult to validate what exactly is learned - outside of our comprehension

What is the danger?

- Methods mostly trained on simulated data
- Difficult to evaluate
- The method can learn structures we can't understand
- Suggestion: Use only with robust validation

EXPLAINING AND HARNESSING ADVERSARIAL EXAMPLES

Ian J. Goodfellow, Jonathon Shlens & Christian Szegedy
Google Inc., Mountain View, CA
{goodfellow,shlens,szegedy}@google.com



Dangers of Deep Learning

Difficult to validate what exactly is learned - outside of our comprehension

What is the danger?

- Methods mostly trained on simulated data
- Difficult to evaluate
- The method can learn structures we can't understand
- Suggestion: Use only with robust validation

(a) Husky classified as wolf (b) Explanation

Figure 11: Raw data and explanation of a bad model's prediction in the "Husky vs Wolf" task.

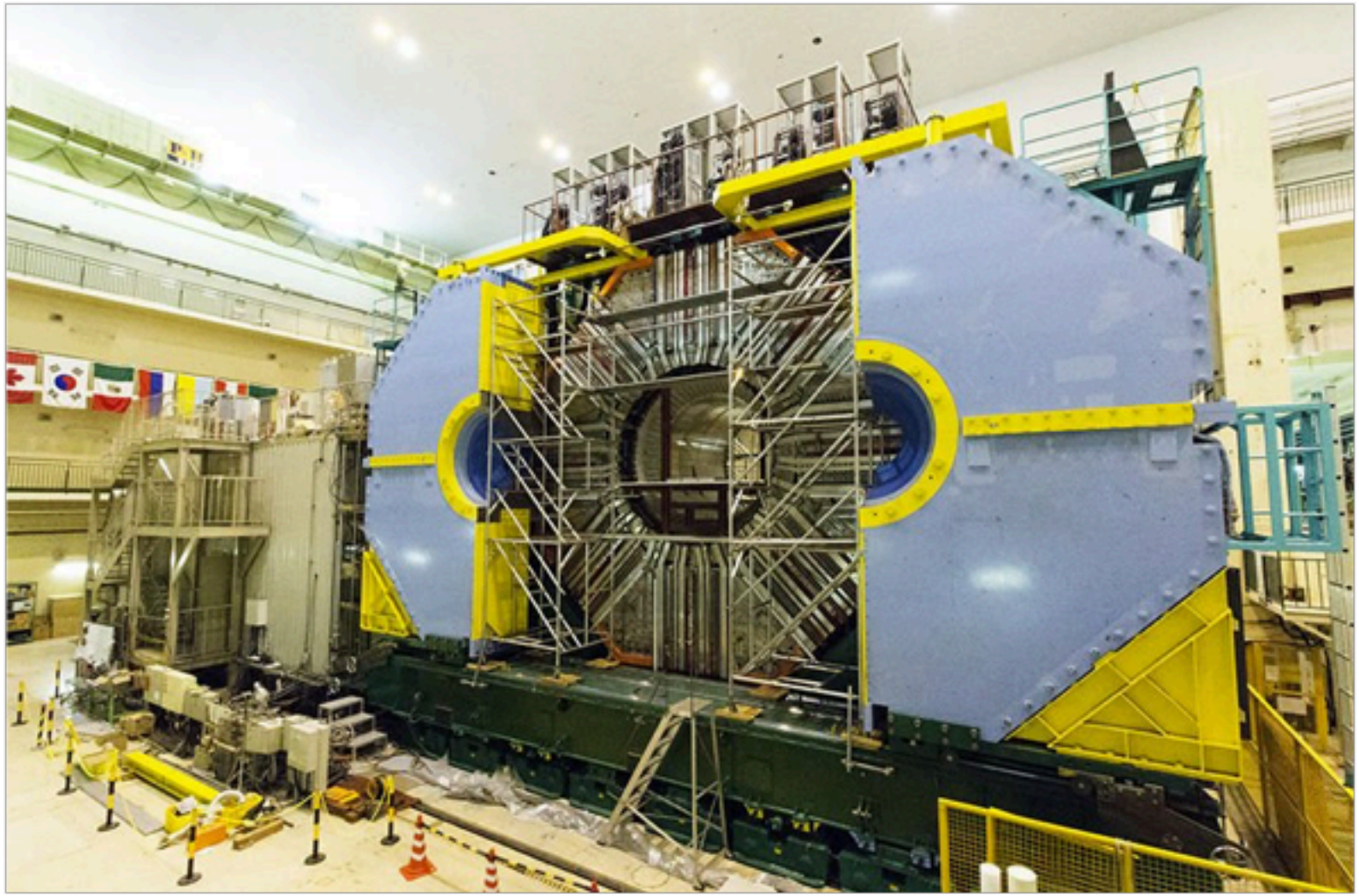
Re-Interfere model response on the input may help understand the expertise

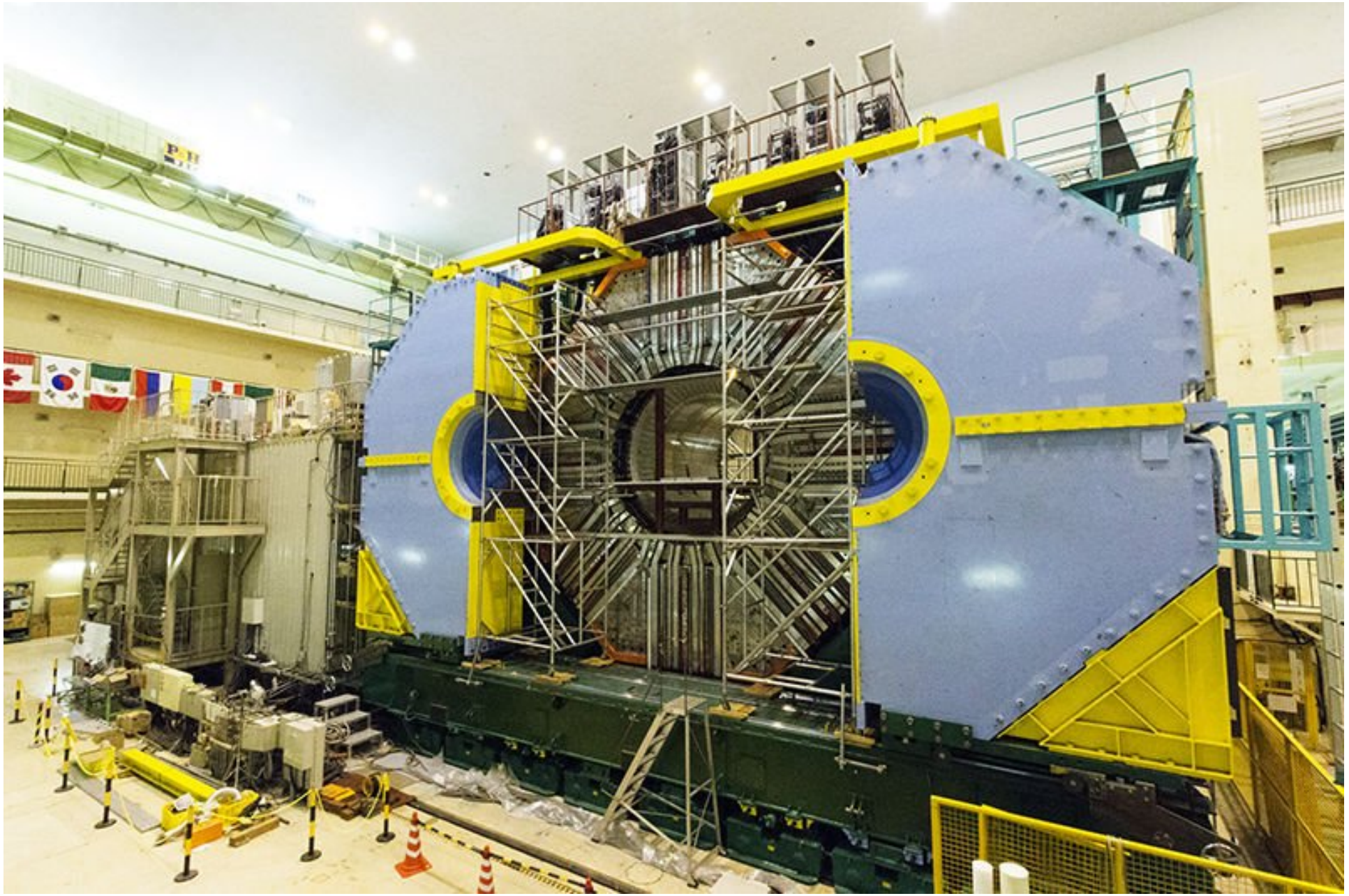
What do Deep Neural Networks think about Belle?

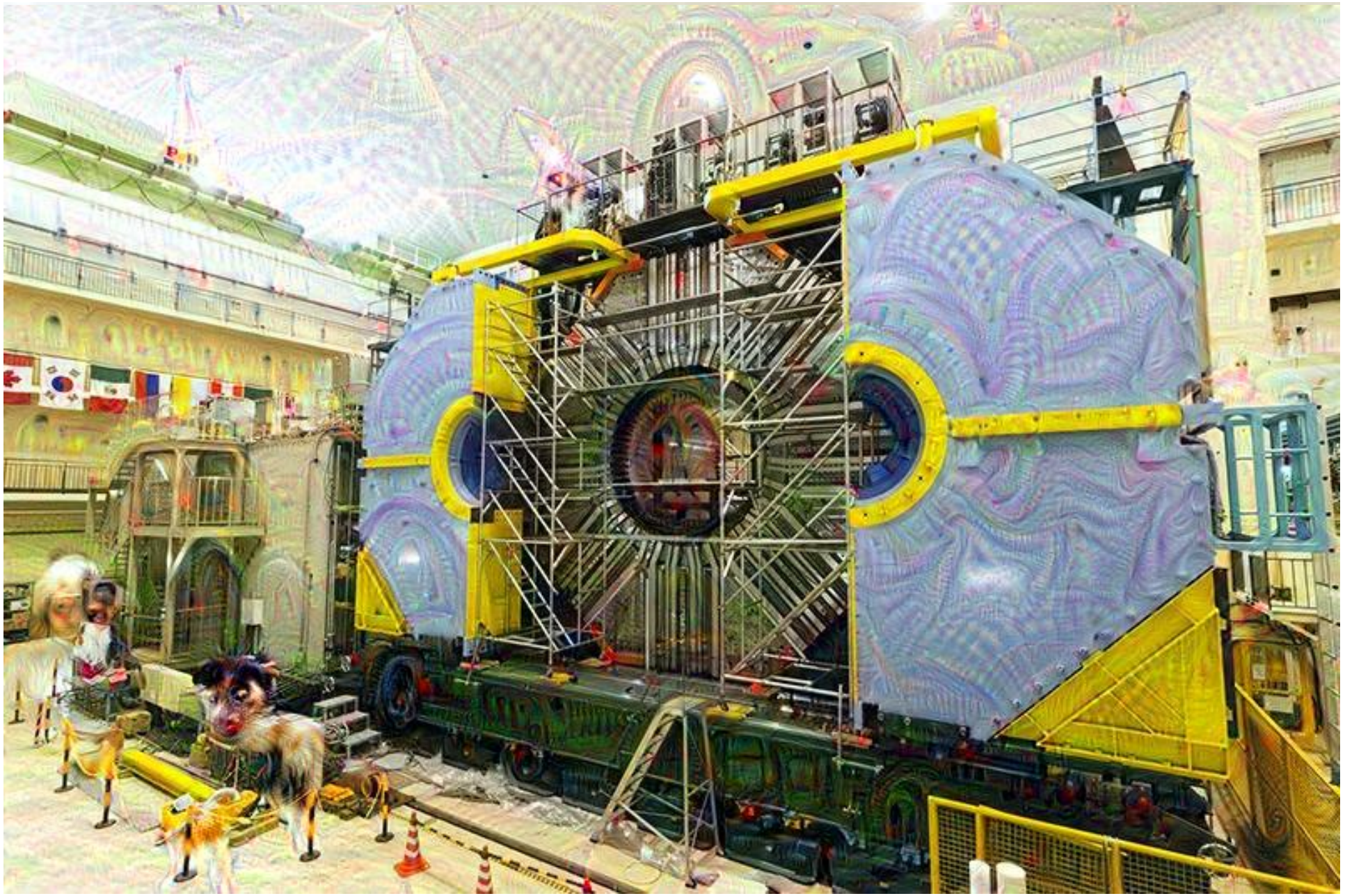
Using DeepDream

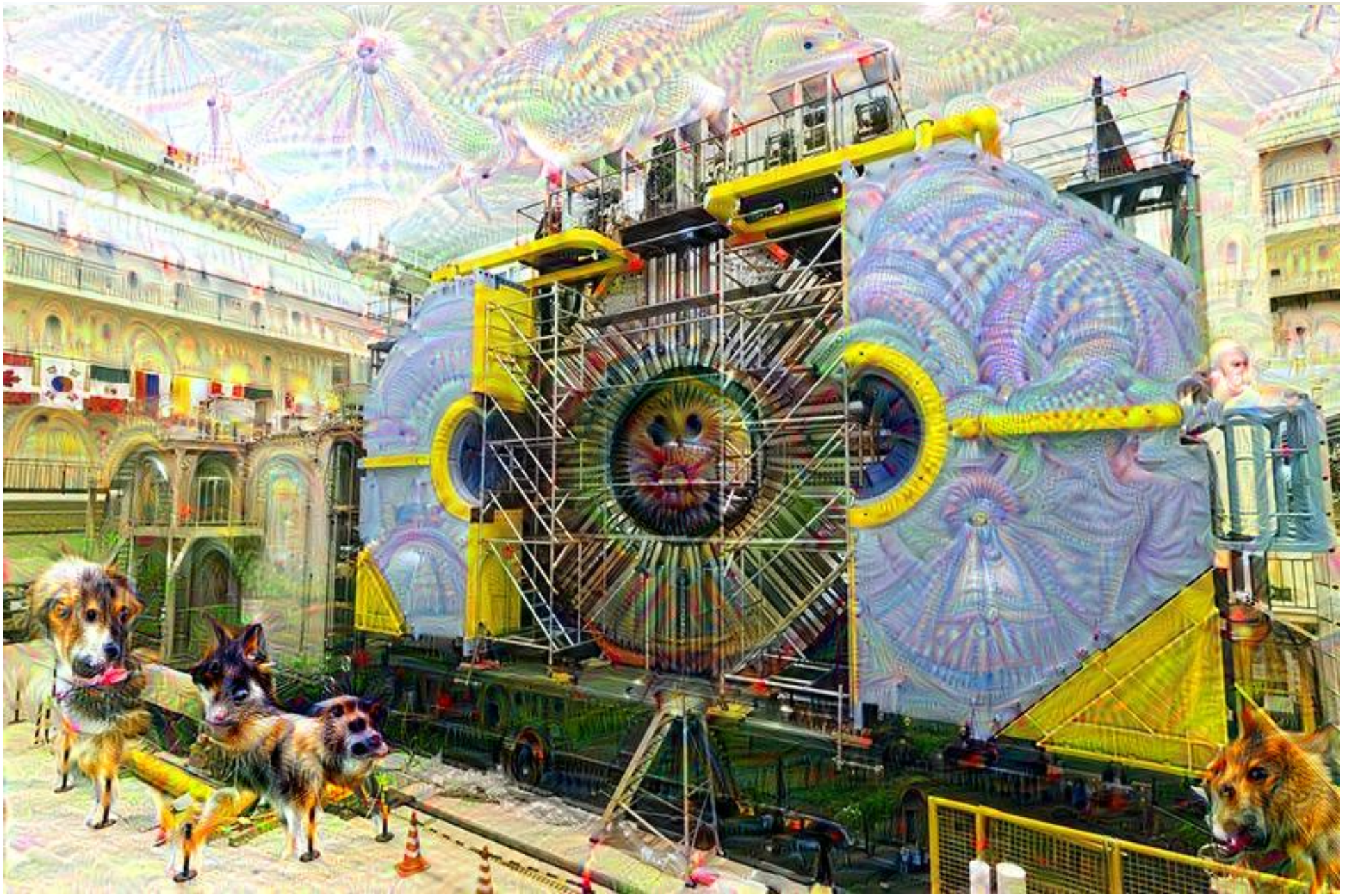
Method

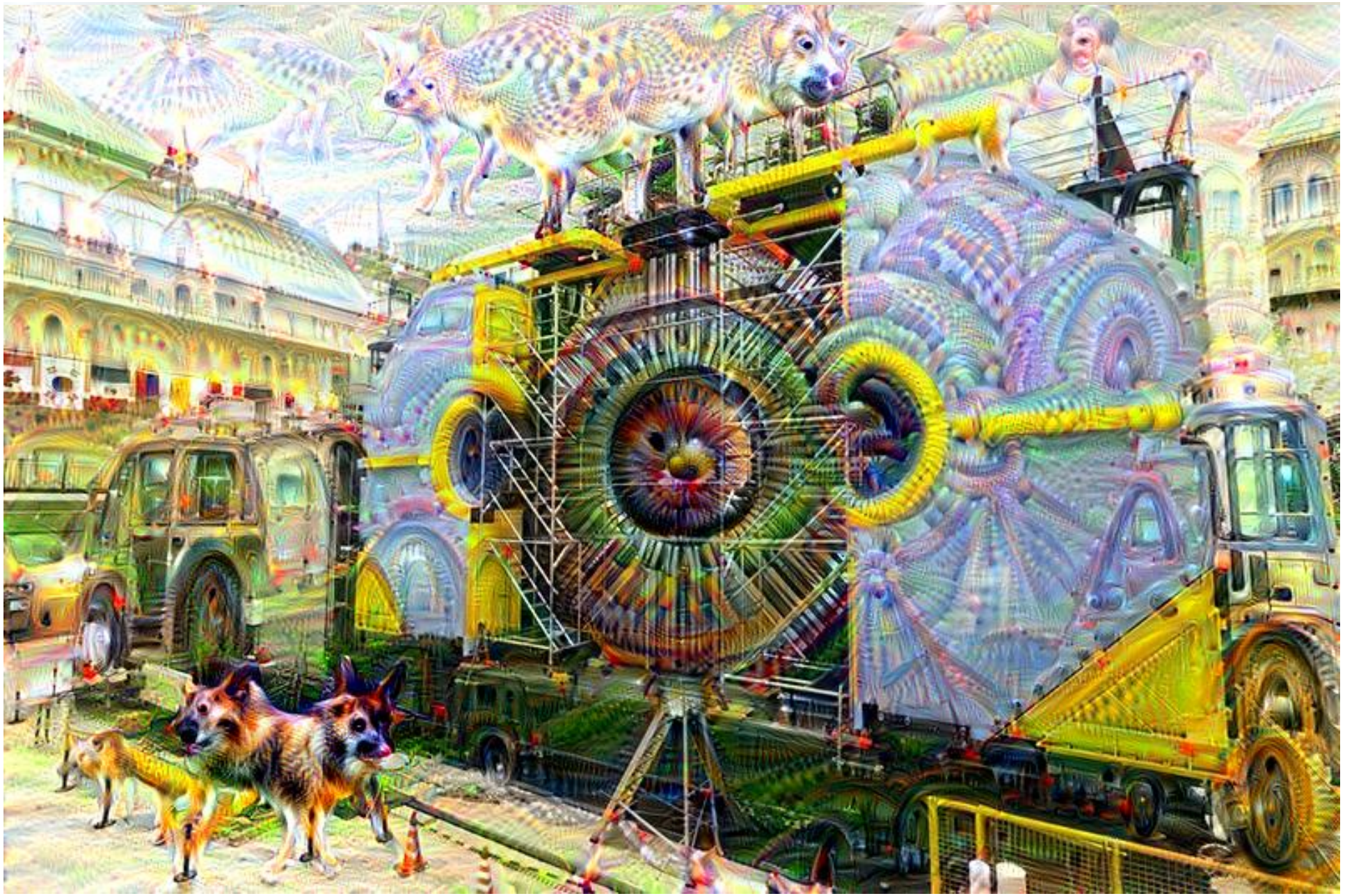
- Take pre-trained deep neural network for objects classification
- DeepDream
- “Reverse” classification by presenting an image to the network and overlaying the original image with what the activated layers “see”

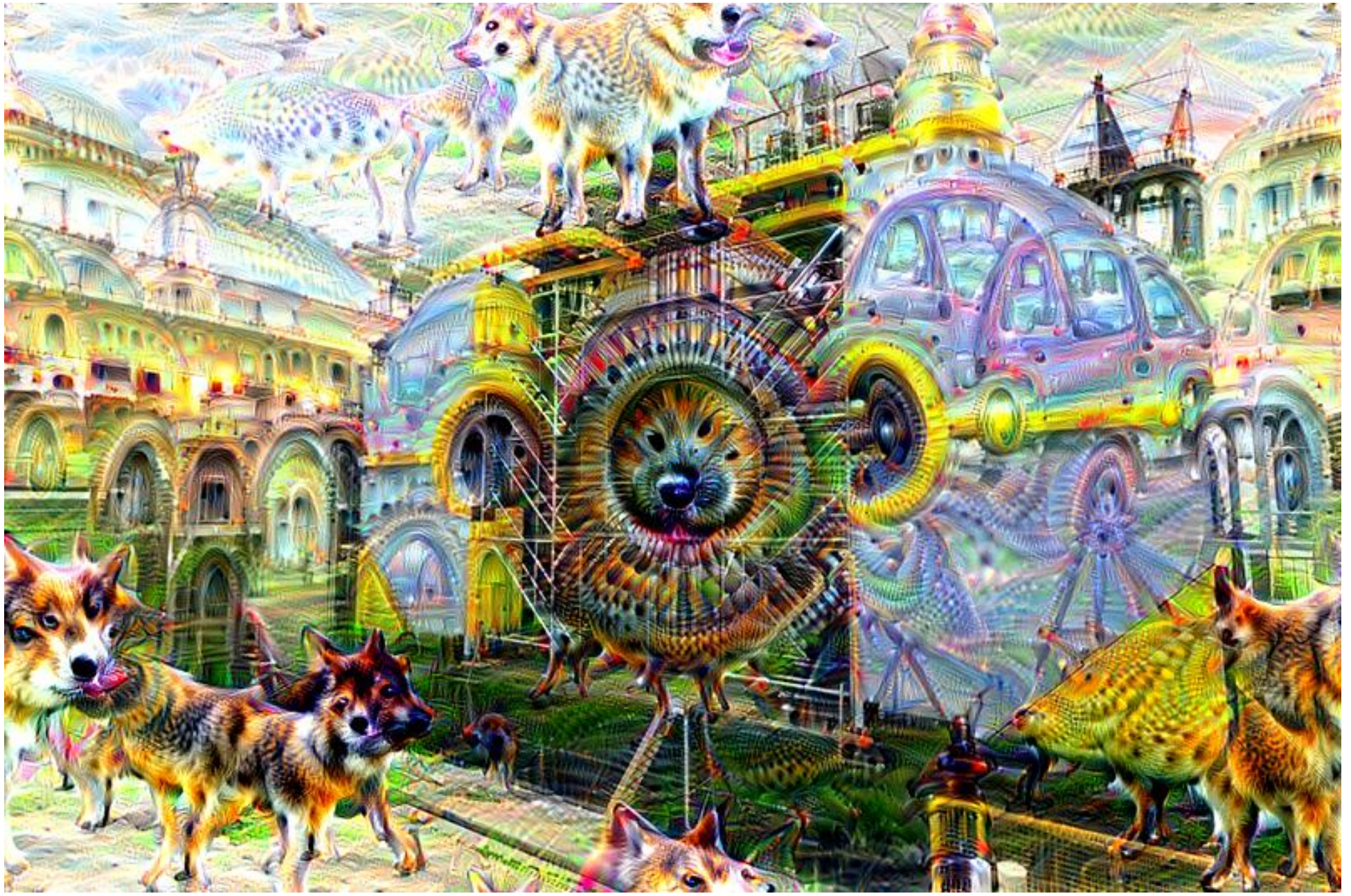


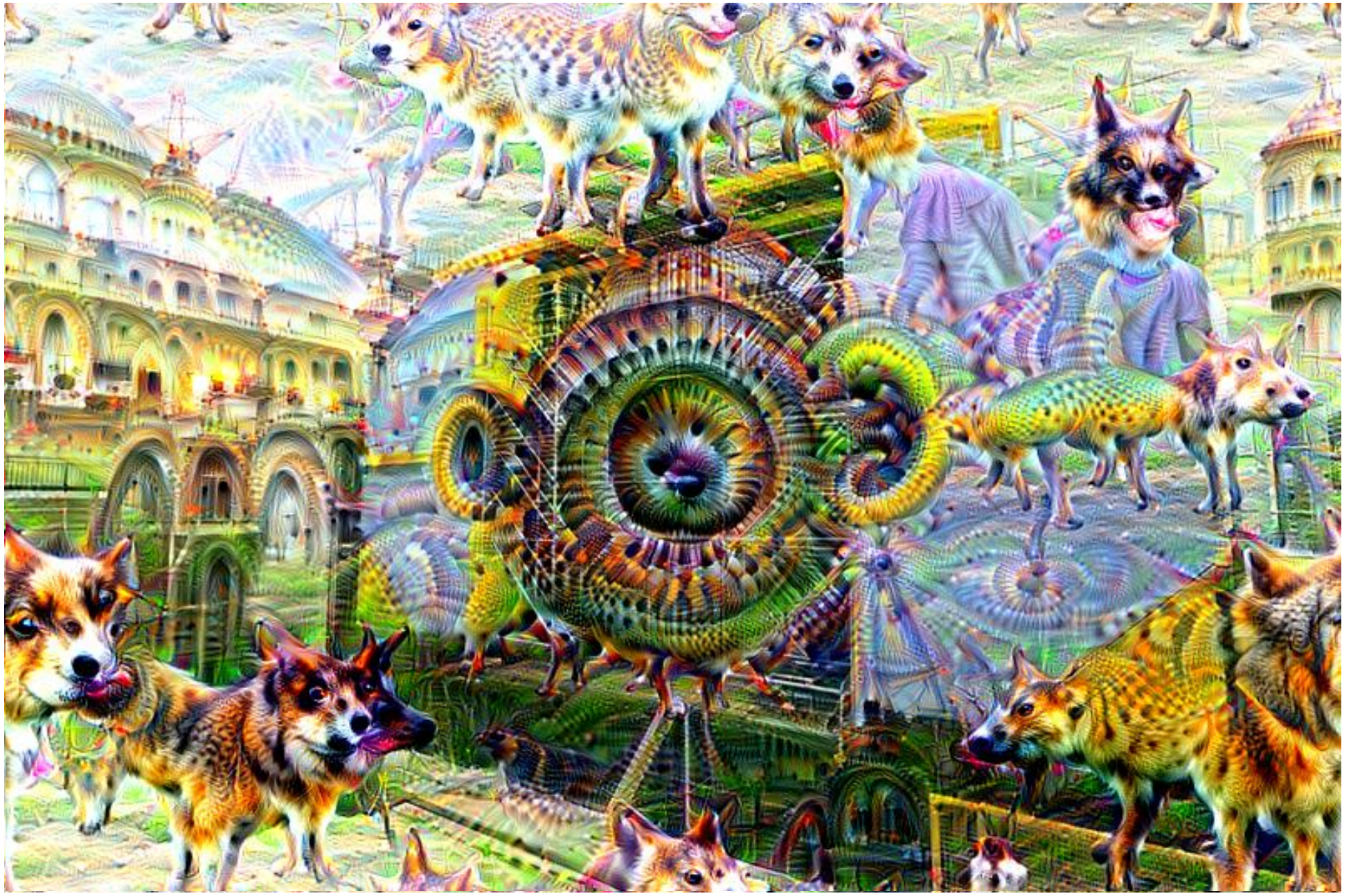













Now
Hands
On
Tutori
l

Hands-on Tutorial

- https://jupyterhub.belle2.org
- https://jupyter.sdcc.bnl.gov