Introduction to Machine Learning in high-energy physics

Introduction and activities in heavy-ion physics

Rüdiger Haake (Yale)
(22.06.2018, LPC Clermont-Ferrand)

Outline

1) Basic concepts and selected advanced ideas

- What is machine learning?
- Techniques: Classification, regression…
- Models, software and tools

2) Applications in high-energy physics

- Focus on heavy-ion physics

3) Example – How to start a ML-assisted analysis?

Take part

Inter-Experimental LHC
Machine Learning Working Group

- Forum for machine learning community at CERN
- Focus on LHC – but everybody welcome!
- We organize monthly meetings, annual huge workshop
- Interface between experiments on ML
  Each LHC experiment is represented by an IML coordinator
- Contact point for everything ML-related, from software to hardware
  - Connects to data science community
  - Foster common solutions
  - Provides training and benchmarks

Take part

Inter-Experimental LHC
Machine Learning Working Group

Get in touch:

- Drop us a mail: iml.coordinators@cern.ch
- Subscribe to CERN egroup: lhc-machinelearning-wg@cern.ch
- No need to have a full CERN account → lightweight account
- Further information: http://iml.web.cern.ch/

For ALICE:

- Group dedicated to machine learning: alice-machine-learning@cern.ch

Concepts of machine learning

Motivation

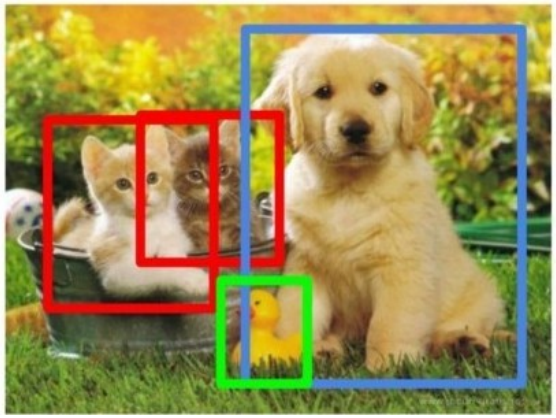
Motivation

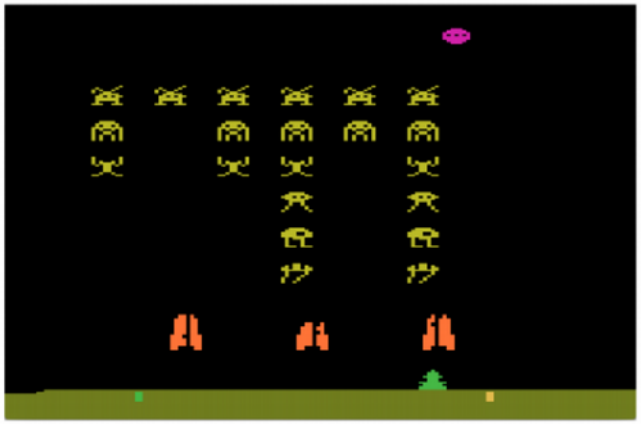
We want to use Machine Learning techniques to

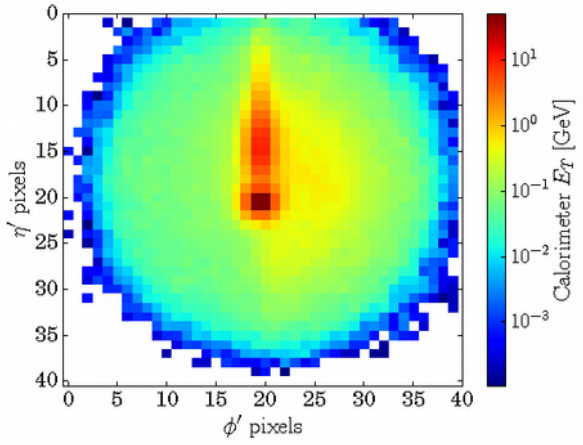
- solve a problem by learning from examples,
- and to be robust against obstacles.

Motivation

Amazing progress in the last years!
Partly reached ‘superhuman’ abilities







Interest rising in physics (and industry)

Introduction

What is Machine Learning?

- Wide field with increasing applications in research and industry
- First ideas already since 1950's
- Machine Learning not new to physics
- In HEP, many 'classic' ML methods are already in use.
  - Boosted decision trees (BDTs) in Higgs search
  - In ALICE, e.g. BDTs for signal extraction for charmed baryons
- Relatively new: Deep learning in physics
  → Huge progress done in last years
  → More and more analyses upcoming in all experiments

Development of Machine Learning techniques is incredibly fast

- A lot of progress from tech companies and industry
- Boost from big data, most progress in deep learning
- Many advanced solutions for "human problems" (e.g. image recognition, text understanding) can also be adapted to HEP problems!

Introduction

Important: Machine Learning is not the solution for all our problems

- Usually, one cannot just throw in data and expect the algorithm to perform better than human even in a well-defined task
  - ML no replacement for domain knowledge
  - “Garbage in → Garbage out”
  - Still: algorithms, classifiers, and training data need to be selected carefully
- Also interesting:
  Deep learning is not necessarily better than classic ML methods

Introduction

Basic ansatz of ML algorithms:

The algorithm improves its own performance on a task by gaining more experience

- In the simplest cases, these are more involved fitting algorithms
- More complex cases include ensembles of several methods including huge neural networks

In a simplified view, one can divide two types of algorithms:

- **Supervised learning** algorithms need labeled training data and learn the correct mapping of input data and desired output
- **Unsupervised learning** tries to find a structure in the data without a priori knowledge of desired outcome

Supervised learning

Supervised learning helps with either
classification or **regression** tasks

Classification: Group samples into predefined classes
Tagging b-jets, classifying neutrino candidates, classify photos of cats & dogs, etc.

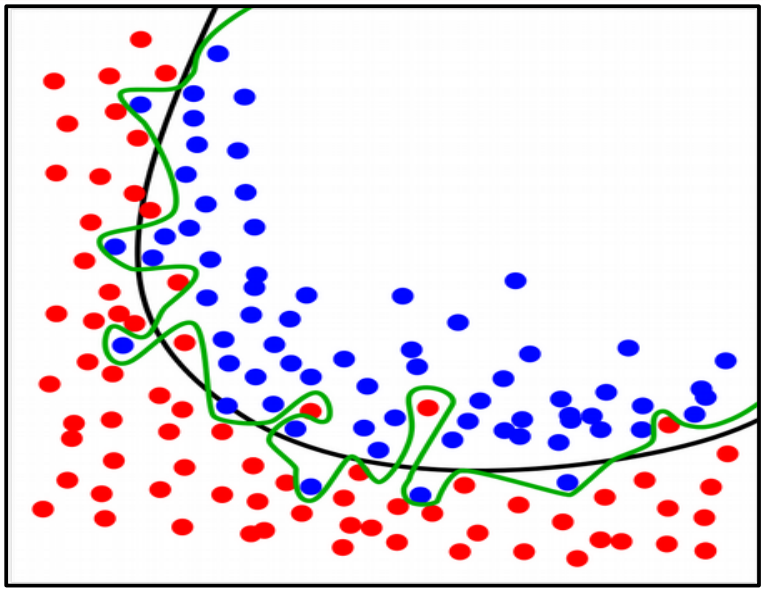
Regression: Assign value to each sample
Calculate jet-underlying background in Pb-Pb, predict estate worth depending on location & properties

General approach:

- Give training dataset to model where truth is known (labeled data)
- Model is optimized according to a loss function to produce the correct output. In a neural network: Weights are adapted
- Idea: Model learns general features and gives correct output for unknown samples
- Output can be a class (e.g. b-jet or not) or a value (e.g. background)

Supervised learning

- Important: Performance of the model must not be evaluated on the training dataset
- Model “has seen” training data and could have learned dataset-specific features
- This won’t help on general data
- This is called overfitting and can happen e.g. for too complex models
  Here the model works amazingly well on the training data, but fails on other data



Green line = overfitting

Solution: Strictly split training, validation, and testing data

- Training data: Used only for training, exclusively
- Validation data: Used for performance evaluation during training
- Testing data: Dataset used for “real physics”

Supervised learning

Crucial: Labeled training samples need to be as accurate as possible

- Classes must be well-defined
- Regression parameters must be precisely known
- Train with garbage = Find garbage
- Usually this means we need a good Monte Carlo description
- There are techniques to improve the MC to better fit the data

Good idea might always be:

- Define a model that is as general as possible, not taking into account details which are in fact no general features
- Do not use features poorly described in MC
- Adjust number of training samples to model complexity, e.g. for neural network:

  $O(\text{samples}) = O(\text{free parameters})$

Unsupervised learning

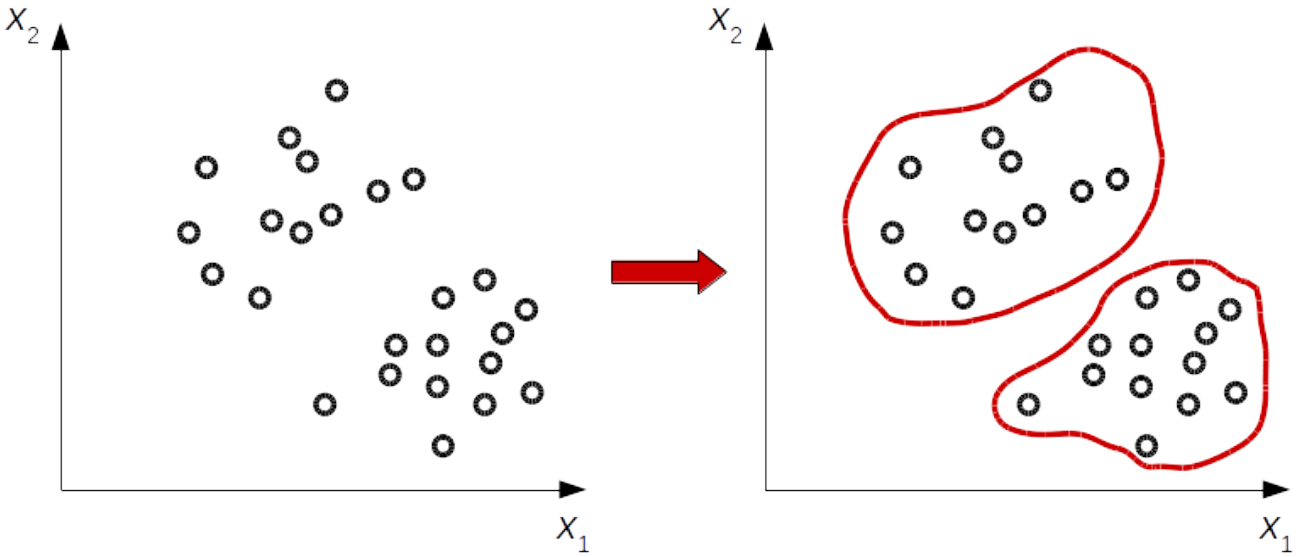
Unsupervised learning works on unlabeled data

Applications: Find structures in data

- Data clustering
- Represent data with reduced dimensionality

- Also: Generative adversarial network (later)

Example for clustering in 2-parameter space:



- Ambiguous task, performance strongly depends on problem
- Not so many applications yet – but one very well known in HEP

Unsupervised learning

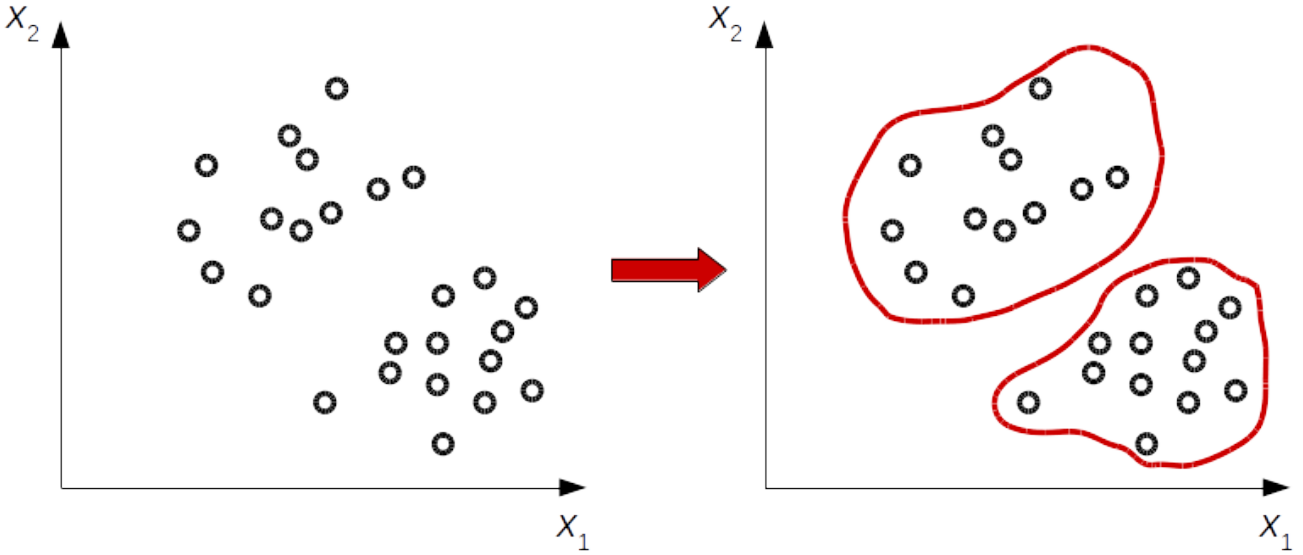
Unsupervised learning works on unlabeled data

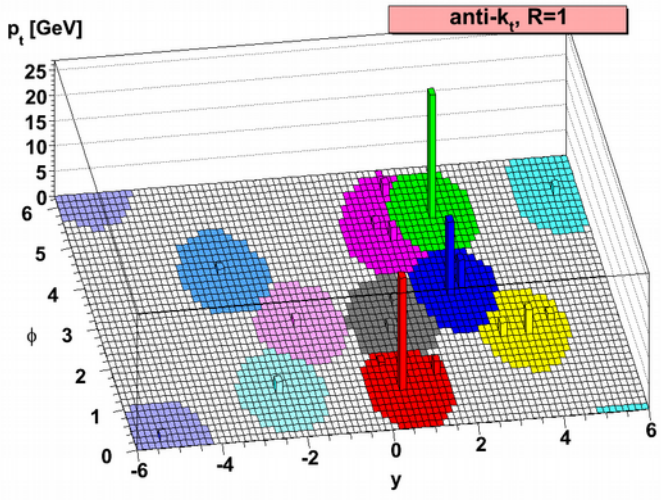
Applications: Find structures in data

- Data clustering
- Represent data with reduced dimensionality

- Also: Generative adversarial network (later)

Example for clustering in 2-parameter space:





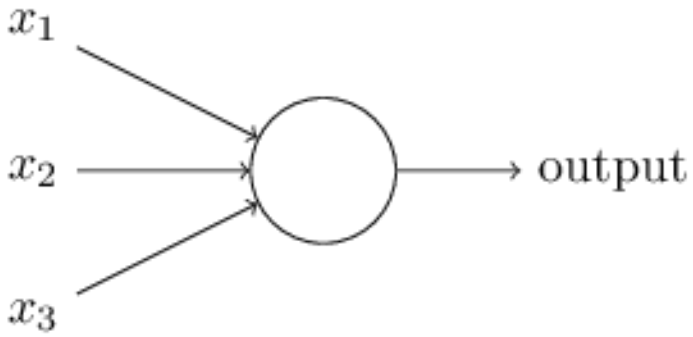
- Ambiguous task, performance strongly depends on problem
- Not so many applications yet – but one very well known in HEP

Neural networks

A lot of models available: Optimum model depends on task!

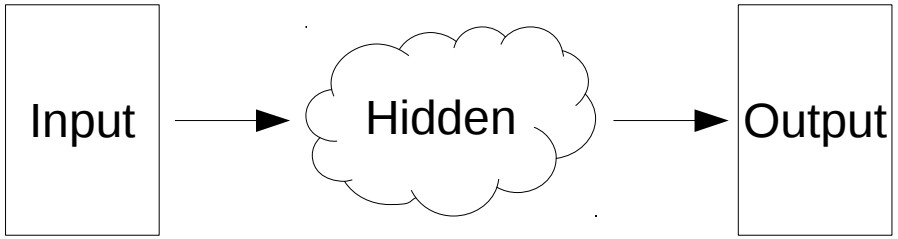
- Here focus on neural networks → allow shallow & deep learning
- Neural networks = Loosely inspired by biological neural networks
- Connected systems of nodes

Neuron:



- Several inputs
- Activation function that weights inputs
- Triggers output according to weights

Neural network:



- Multiple neuron layers, chain of tensor operations
- Multidim. input → multidim. output
- Massively parallelized → use GPUs

Neural networks

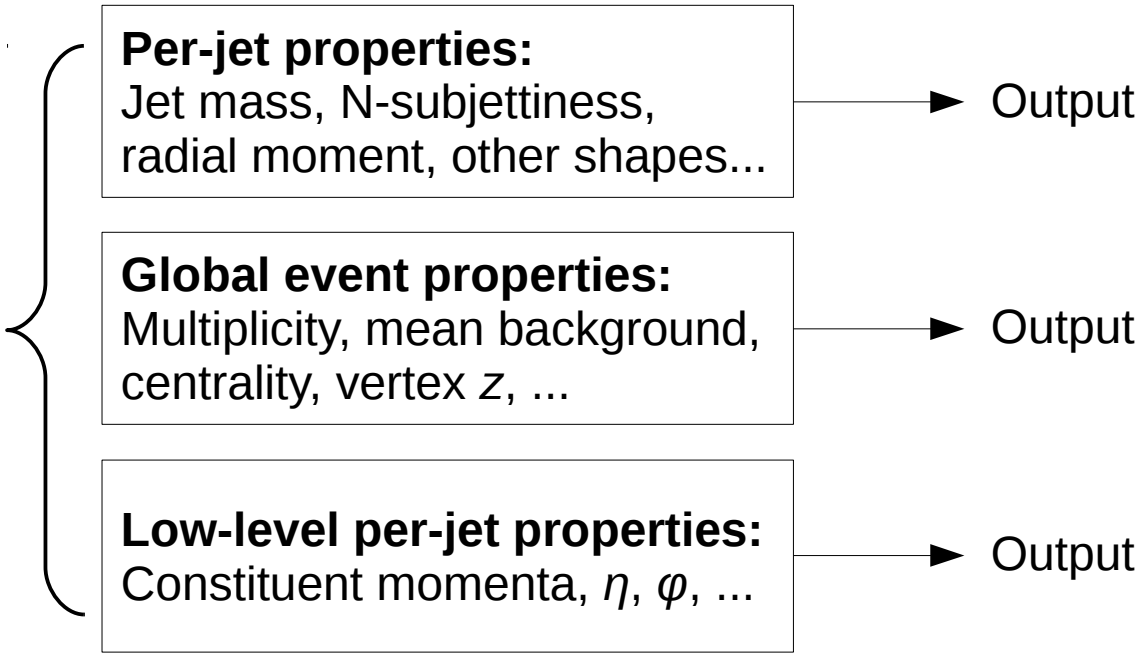
A lot of models available: Optimum model depends on task!

- Here focus on neural networks → allow shallow & deep learning
- Neural networks = Loosely inspired by biological neural networks
- Connected systems of nodes

Network of networks:

Neural networks allow to combine useful networks
→ Powerful ansatz!

- Train several networks
- Inputs can be very different!



Neural networks

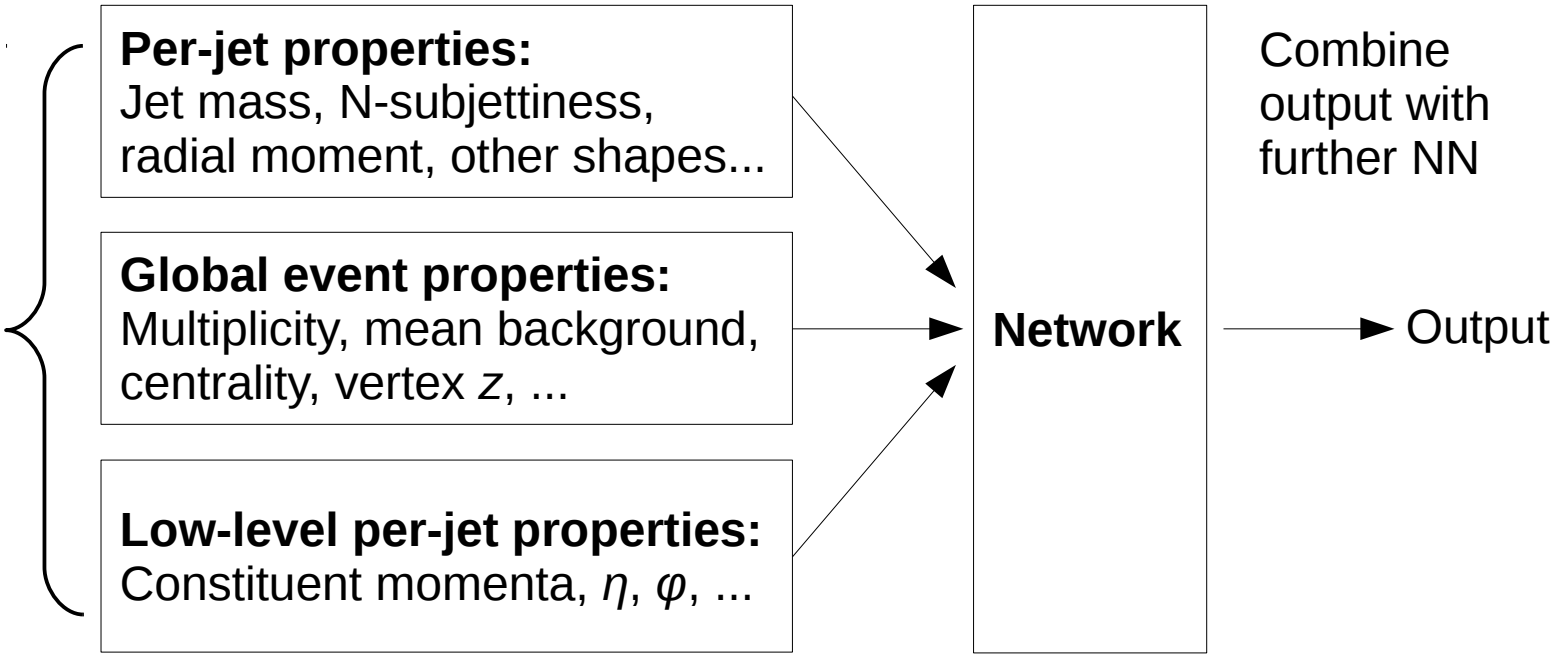
A lot of models available: Optimum model depends on task!

- Here focus on neural networks → allow shallow & deep learning
- Neural networks = Loosely inspired by biological neural networks
- Connected systems of nodes

Network of networks:

Neural networks allow to combine useful networks
→ Powerful ansatz!

- Train several networks
- Inputs can be very different!



Software and frameworks

No need to write everything from scratch: Open frameworks exist
Huge data science community → large knowledge base!

Keras:
- Fast-growing easy-to-use Python lib
- Allows application of deep-learning models (CNNs, LSTMs, …)
- Tensorflow, Theano backends → GPU support

Scikit-learn:
- Python lib that implements many (non-deep) techniques
- A lot of data preprocessing & statistics tools:

TMVA:
- Implemented in ROOT
- Pro: Allows direct integration in HEP codebase
- Has many (non-deep) ML techniques implemented
- Interface to Keras (deep neural networks) in development

Clear advantage of data analysis with Python: Very quickly set up

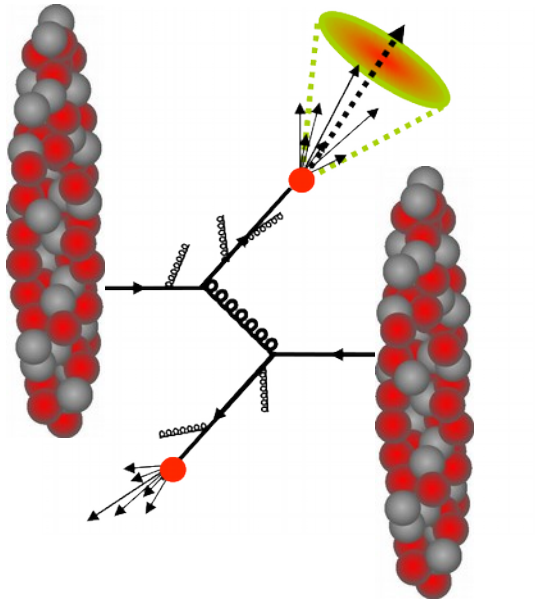
A few words on hardware

- Depending on task, many ML-analyses can be done on laptop
- But: Deep neural networks usually need additional computation power
  - Large data samples (*O*(100k))
  - Huge amount of free parameters (*O*(100k))
- University/lab GPU farms: Good solution for long-term training jobs
  In the next months, CERN might also come up with a GPU farm
- For testing and medium scale analysis: Local GPUs

Example:

- I use an NVIDIA GTX 970 Ti for my analyses (few hundred Euros)
- Several orders of magnitude faster than CPU-only
- Make sure GPU is supported by ML-backend of choice!

Applications in high energy physics: Particle jets



Jets: Machine learning applications

Depending on the problem, jets can serve as input to

- shallow learning algorithms (e.g. BDTs)
- deep learning (neural networks)

High-level parameters	**Low-level parameters**
Per-jet properties: Jet mass, N-subjettiness, radial moment, other shapes...	**Low-level per-jet properties:** Constituent momenta, η, φ, ...
Global event properties: Multiplicity, mean background, centrality, vertex z, ...	**Other low-level properties:** Reconstructed secondary vertices, ...

- Features need to have discrimination power on problem
- Need good MC description of features

Typical applications for jets

Jet tagging/classification

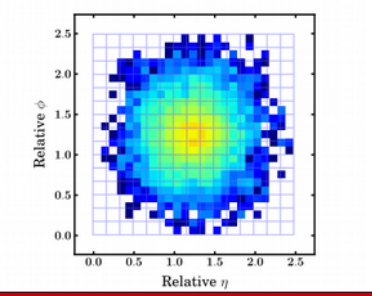
- q/g-jet tagging
- b/c-jet tagging
- W-jets vs. QCD jets
- Multiclass jet classification

Regression of jet parameters

- Background in heavy-ion jets

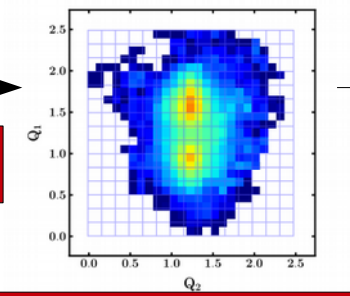
Jets: Jet images

- Motivation: Huge progress with convolutional neural networks in image recognition/classification
- Classify jets according to their pattern they leave in detector
  - *… in calorimeter cells*
  - *… as charged particle tracks*
- In 1407.5675, jet images are used for W-jet tagging

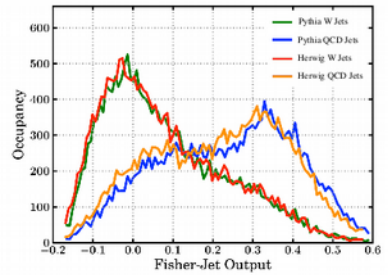


Average jet image before preprocessing

Preprocessing



Average jet image after preprocessing

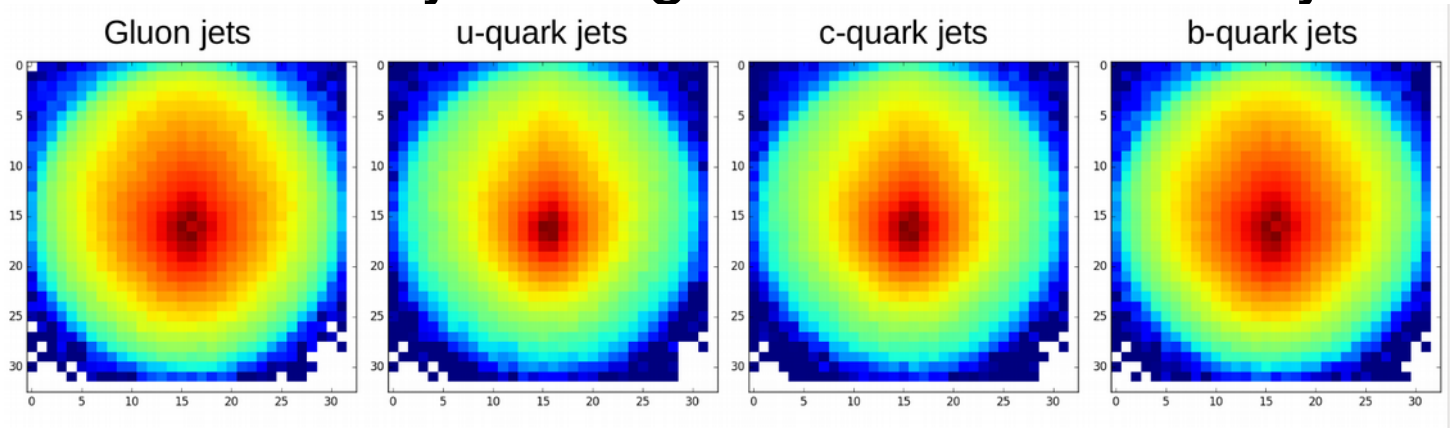


Discrimination between the two populations

- Several approaches on jet images: CNNs, Locally-connected networks…
- Works, but have in mind: “Jets are no cats”

Jets: Jet images

- Motivation: Huge progress with convolutional neural networks in image recognition/classification
- Classify jets according to their pattern they leave in detector
  - *… in calorimeter cells*
  - *… as charged particle tracks*
- In 1407.5675, jet images are used for W-jet tagging



Might also work for QCD jet classification

- Several approaches on jet images: CNNs, Locally-connected networks…
- Works, but have in mind: “Jets are no cats”

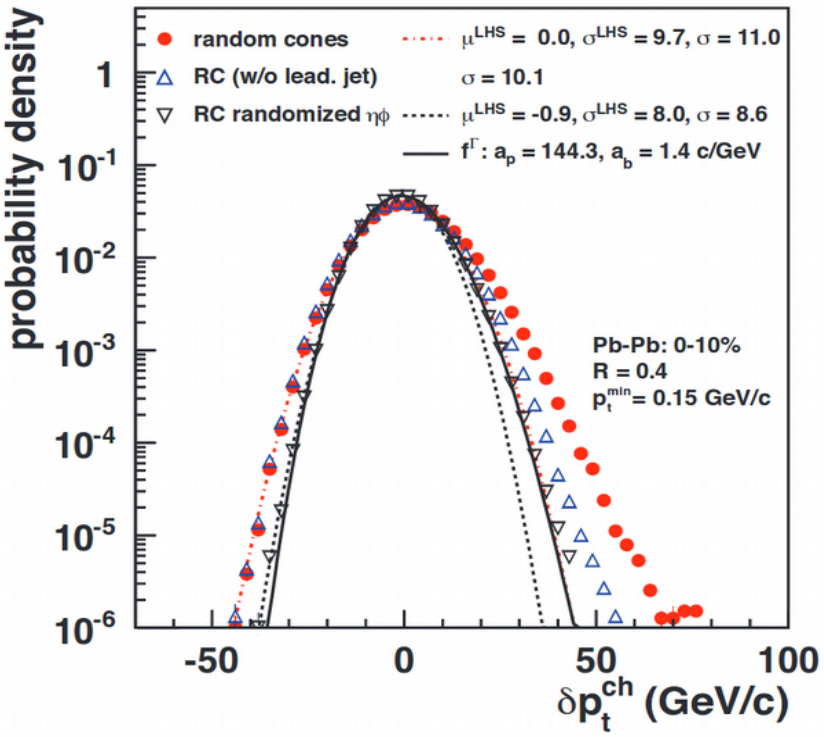
Jets: Recurrent/recursive approaches

- Jet images exploit analogy to image classification
- What about analogy to speech recognition?
- Exploit:
  Sentence = sequence of words
  Jet = sequence of constituents
- Good analogies are useful: We can build on progress in computer science
- A lot of research on text classification/understanding

- **Like for text classification, recurrent networks promising**
- Interesting in this context:
  Recursive networks whose topology changes event-by-event depending on jet finder combination history (1702.00748)

Jets: Background approximation

- In heavy-ion collisions, jets strongly affected by background
  - → We are interested in the hard jet process
  - → But jet is overlapped by many soft processes which are not correlated to jet
- Ideal example for regression task: Approximation of background



Old ansatz:

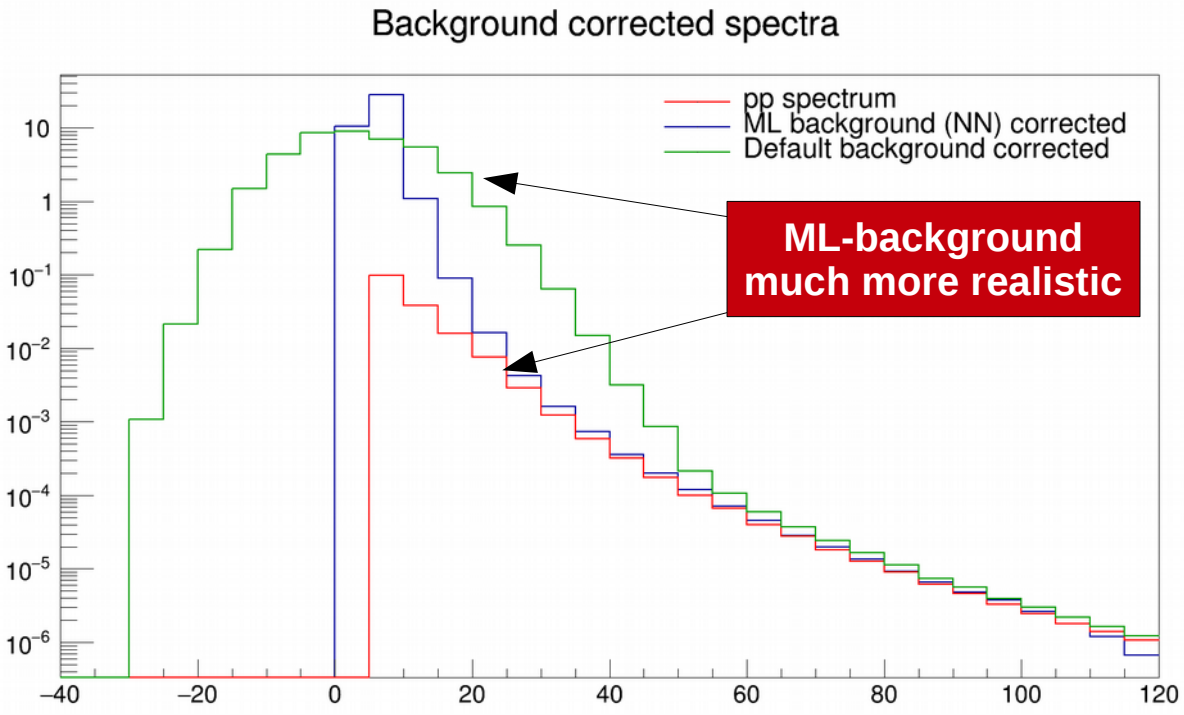
- Calculate background **event-by-event** and subtract from each jet
- Correct for fluctuations in unfolding

New idea:

- Approximate background **jet-by-jet** using ML methods

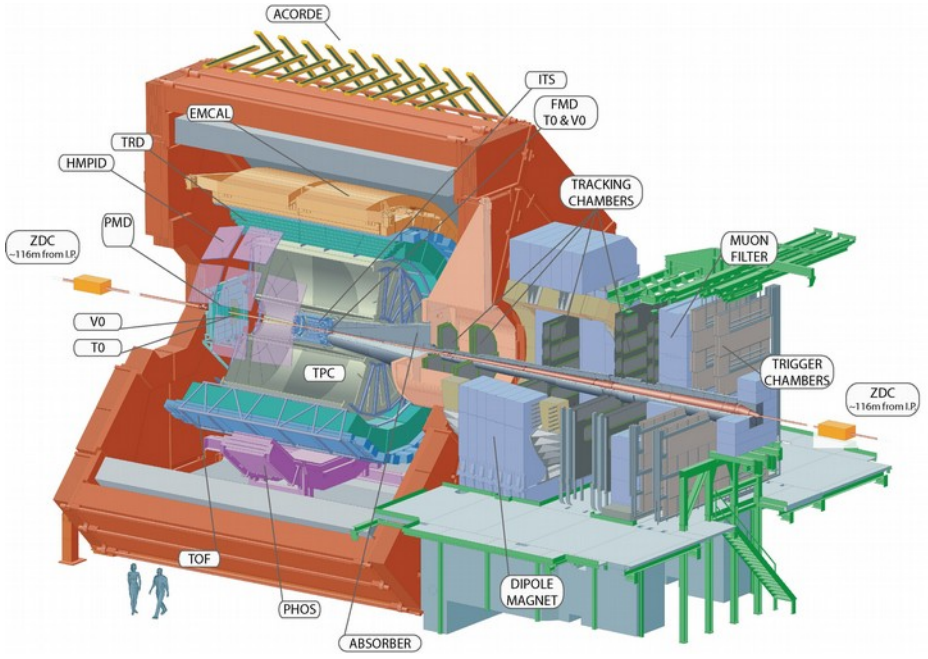
Jets: Background approximation

- In contrast to jet constituents, background is uncorrelated from jet
  → Neural network might be able to estimate background
- Major concern is the need for good training data
- Here, this means jets & background must be realistic:
  - Monte Carlo jets from PYTHIA (real physics)
  - Background from toy, parameters taken from real data



- Might eventually allow measurement of jets at lower transverse momentum
- First results promising with simple neural networks

Applications in high energy physics: Particle identification



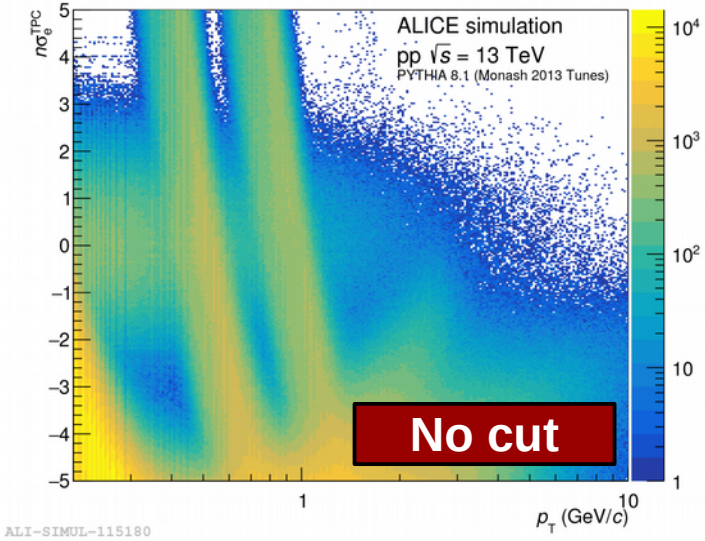
PID: Motivation

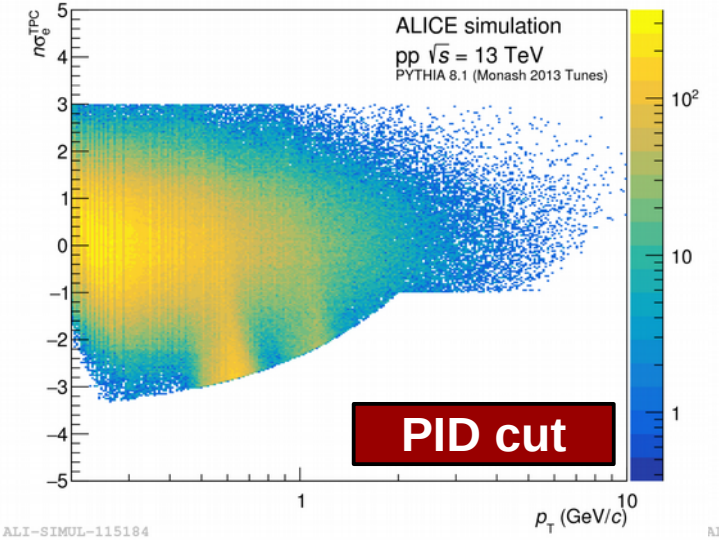
- In tracking, usually transverse momenta measured
- Particle identification needs further measurement
- Information on particle species often important for physics analysis:
  - Production of pions, kaons, protons and their modification in heavy-ion collisions
  - Heavy-flavor physics (D, B-mesons, b-jets)
  - Neutron pion production
  - Photon production
  - Particle composition in jets
  - …

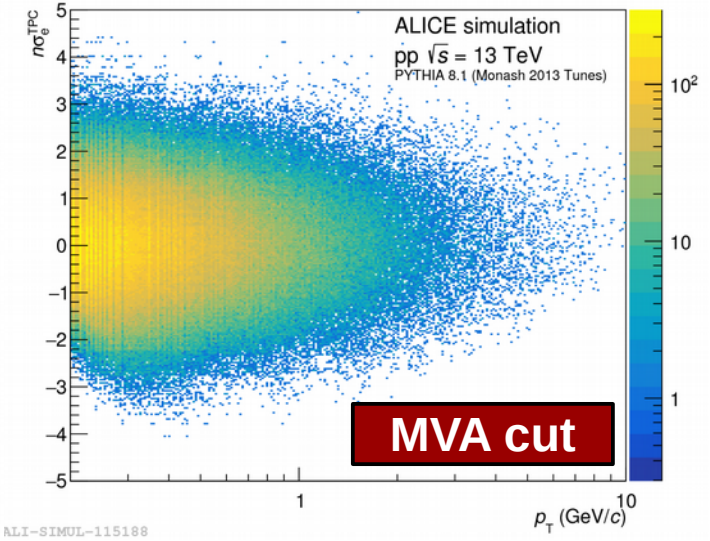
ALICE uses a variety of different detectors to gain complementary information on particles

PID: Electron identification

- Example for electron identification
- MVA approach:
  Use Boosted Decision Tree (BDT) on *nσ* values, track properties







nσ distribution for electrons TPC

PID: General identification task

Ultimate goal: General particle identification which exploits all available information

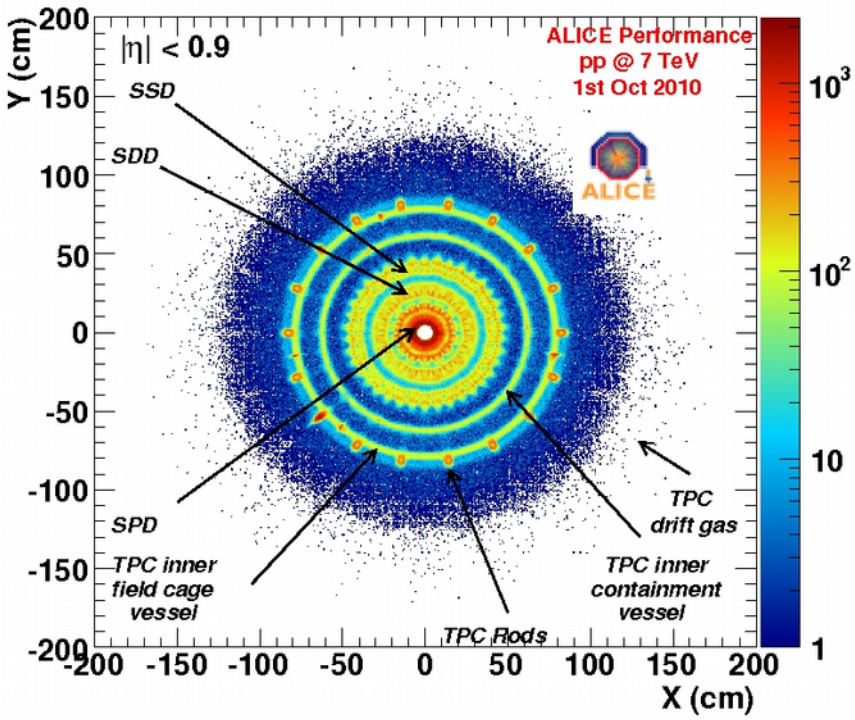
Stage 1: Classifier works on cleaned, calibrated distributions, e.g. on $n\sigma$ values
Stage 2: Classifier works on raw PID detector distributions

Both cases would be very helpful to raise efficiency and purity

Crucial point: Monte Carlo productions

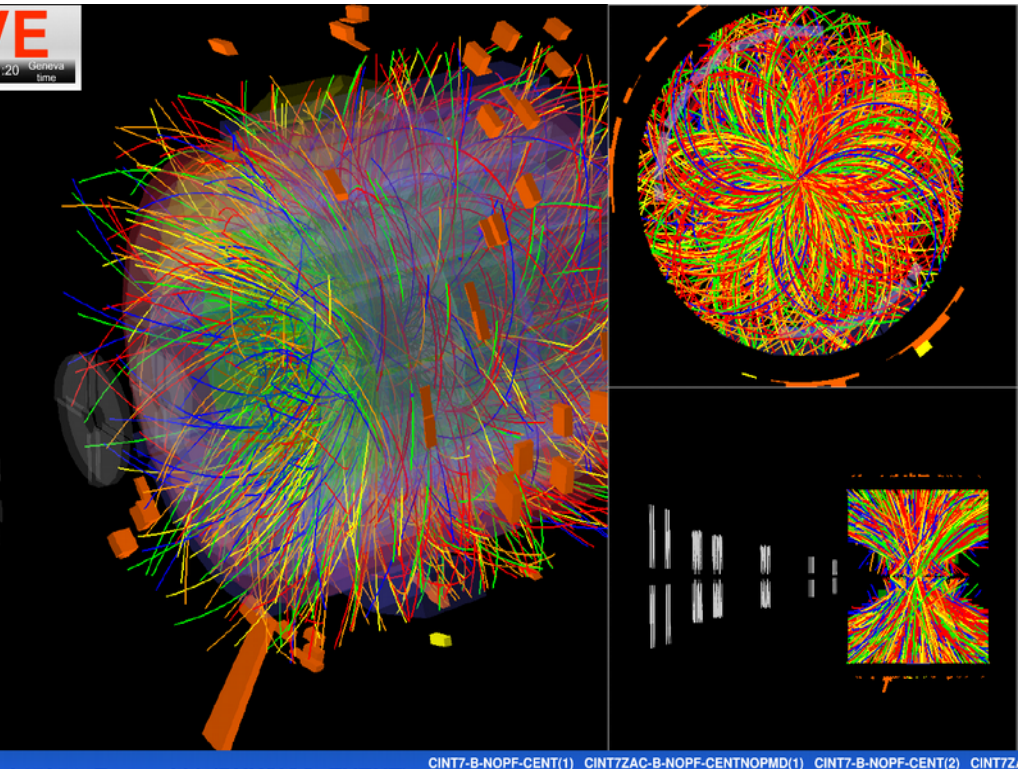
- Training data needs to be as precise as possible
- Monte Carlos often show a poor PID agreement, at least for some particle species

Applications in high energy physics: Fast simulation



Fast simulation: GANs

- Simulation of expected physics in detector crucial for interpretation
- Usually: Monte Carlo simulation
  1) Event generation on particle level (e.g. PYTHIA)
  2) Reconstruction on detector level (e.g. GEANT)



- Both steps can be computationally expensive, especially for heavy-ion collisions
- Currently, huge amount of computing resources (~50%) used for simulations
- Computational costs for LHC run 3 and HL-LHC much worse:
  Higher statistics in data
  → **need higher MC statistics!**

Fast simulation: GANs

- General approach: Use fast simulation
  Ex.: Mix fully-reconstructed with only simply reconstructed signals
- But to prepare for HL-LHC (~2023), we need to save more (x100)
- Promising ansatz: **Generative models**, realized as DNNs
  - Variational Autoencoders (VAEs)
  - Generative Adversarial Networks (GANs)
  - …

  → **Generation of realistic samples according to training samples**
- Unsupervised learning

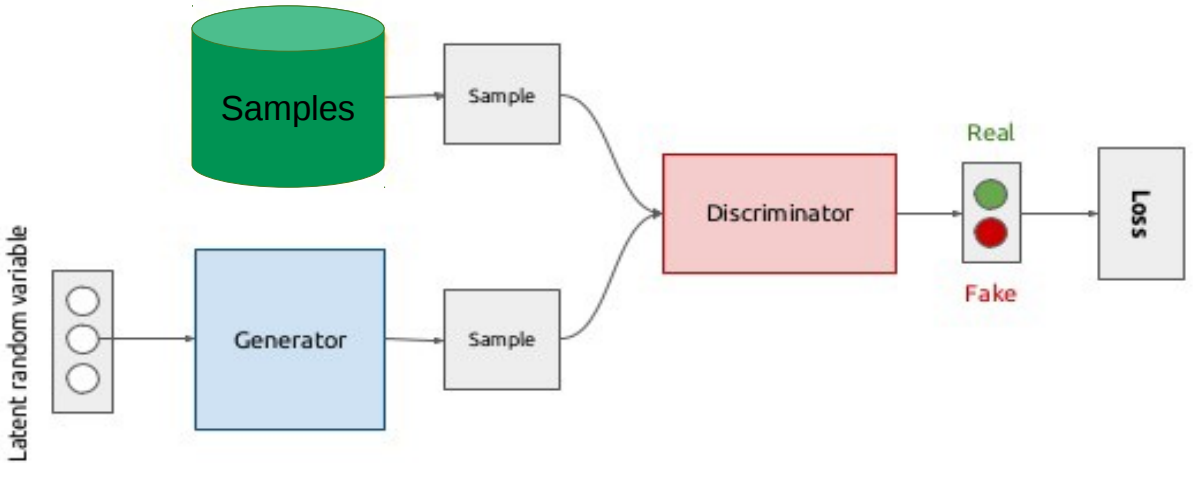
Advantages over classic MC:

- Neural network inference much faster than reconstruction (x 10^5)
- Parallel computing (GPU), not so much CPU-bound
- Can use commercial infrastructure: GPU clusters, cloud computing

Fast simulation: GANs

Generative Adversarial Network:

Two networks trained simultaneously: Generator and discriminator



- In competition & cooperation, generator learns to create more and more realistic samples
- Several studies show that deep GANs are able to reproduce a very large feature space

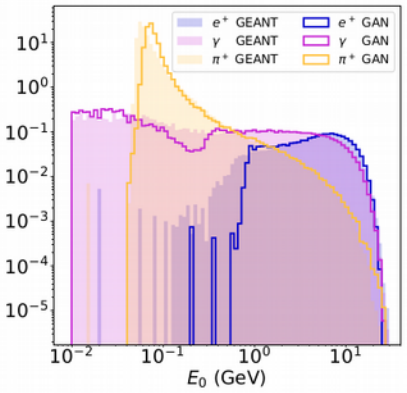
Fast simulation: CaloGAN

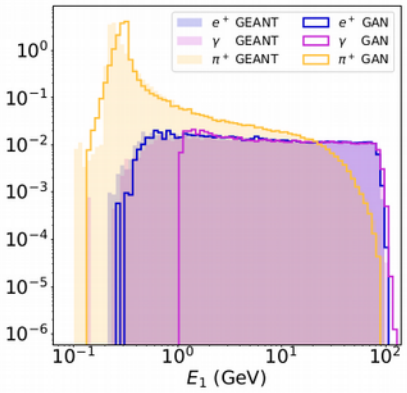
In ALICE, proof-of-concept work is ongoing to use GANs

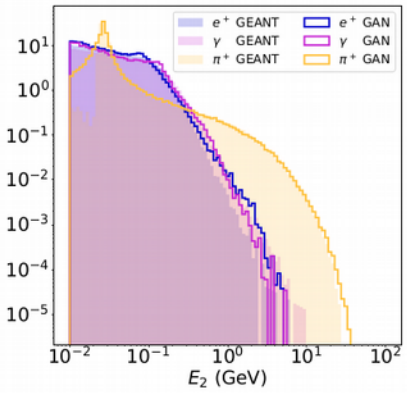
- to simulate fully-reconstructed tracks with TPC
- to perform detector reconstruction of particle-level data

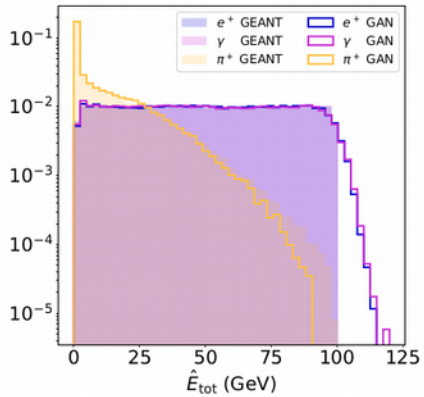
Outlook what is possible: **CaloGAN** (GAN for ATLAS LAr calorimeter)

Deposited energy

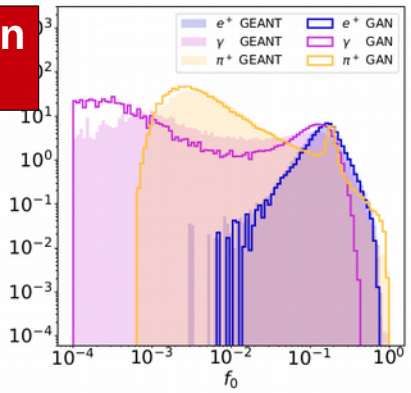


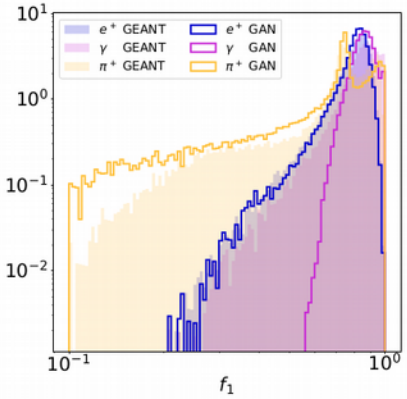


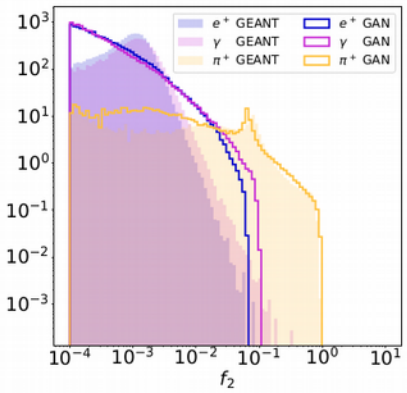


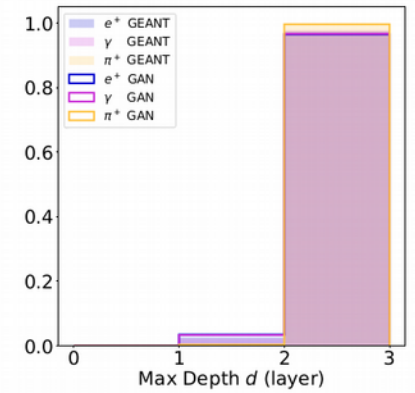


Energy fraction in i$^{\text{th}}$ layer









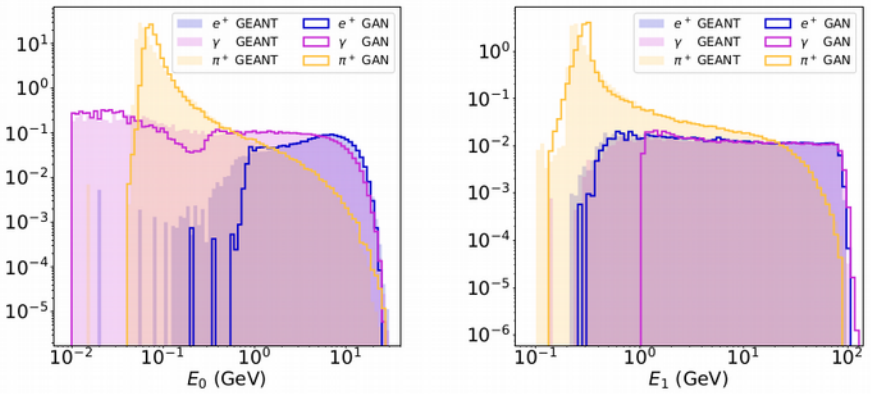
Fast simulation: CaloGAN

In ALICE, proof-of-concept work is ongoing to use GANs

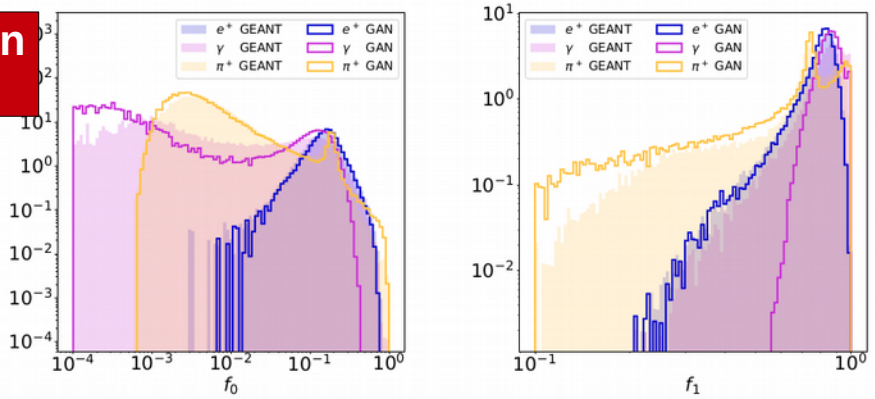
- to simulate fully-reconstructed tracks with TPC
- to perform detector reconstruction of particle-level data

Outlook what is possible: **CaloGAN** (GAN for ATLAS LAr calorimeter)

Deposited energy



Energy fraction in ith layer



- Works for complex segmented calorimeter
- Up to five orders of magnitude faster
- More work to be done until ready for production

Some pragmatic hints:
How to start an ML-analysis

How to design a good model

Define your problem

- Is it a regression or classification task?
  Optimizer, loss function, activation function, etc. depend on this choice
- In a classification task, do you need multiple classes or does binary classification suffice?
  Binary classification might be easier to learn for the network than multi-class classification
- Will the problem only rely on high-level parameters?
  If yes, also different technique (like BDTs) can be considered
  High-level parameters are e.g. jet mass, jet shapes. Low-level parameters e.g. constituents

Define your dataset

- In case of classification, clearly define signal and background
- In case of regression, be sure your regression parameter is well defined and represents what you want
  Crucial step, better put more effort here than less
- Which input features could potentially have discrimination power for your problem? Implement them

How to design a good model

Define your model(s)

- Get inspired by similar problems, experiment with different designs
- Once you found a suitable design → Perform grid search
  Clever “brute force” trial of possible hyper parameters
  - Number of layers, neurons per layers …
  - Activation function, loss function, …

If suitable, combine several models on features

- Useful, if the models work on distinct input features
- Example: Combination of PID classifier models on TPC and TOF might be useful

Systematic uncertainties

In general, calculation of systematic uncertainties very similar to standard analysis

1) Method commissioning & validation
2) Uncertainty on method
3) Uncertainty from analysis parameter variations

Systematic uncertainties

In general, calculation of systematic uncertainties very similar to standard analysis

1) Method commissioning & validation
2) Uncertainty on method
3) Uncertainty from analysis parameter variations

Commissioning & validation:

Test that ML technique brings expected results, e.g. in Monte Carlo

- Not always easy: Check cross cross correlations with other analysis parameters to guarantee that measurement has not strong bias
- Same to be done for standard analysis, but more effort to be done for ML techniques
- Many experts still need to be convinced

Systematic uncertainties

In general, calculation of systematic uncertainties very similar to standard analysis

1) Method commissioning & validation
2) Uncertainty on method
3) Uncertainty from analysis parameter variations

Uncertainty of Machine Learning method:

Test sensitivity to method configuration

- Slightly change neural network architecture/ model parameters
- Even possible: Compare to results from other models
  → might overestimate uncertainty

Systematic uncertainties

In general, calculation of systematic uncertainties very similar to standard analysis

1) Method commissioning & validation
2) Uncertainty on method
3) Uncertainty from analysis parameter variations

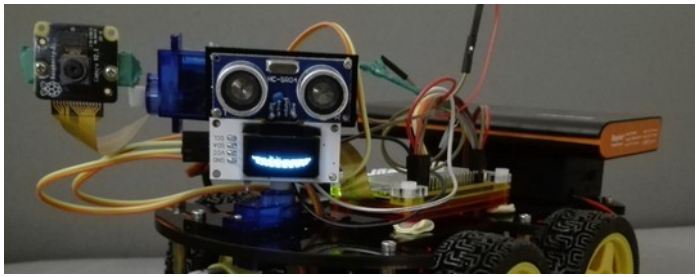
Uncertainty of analysis parameter variations:

Test sensitivity to certain analysis parameters

- Similar to the default analysis, but have in mind:
  → Depending on input data, result can implicitly depend on more parameters
- Might need huge amount of resources: Model training done several times

Conclusions

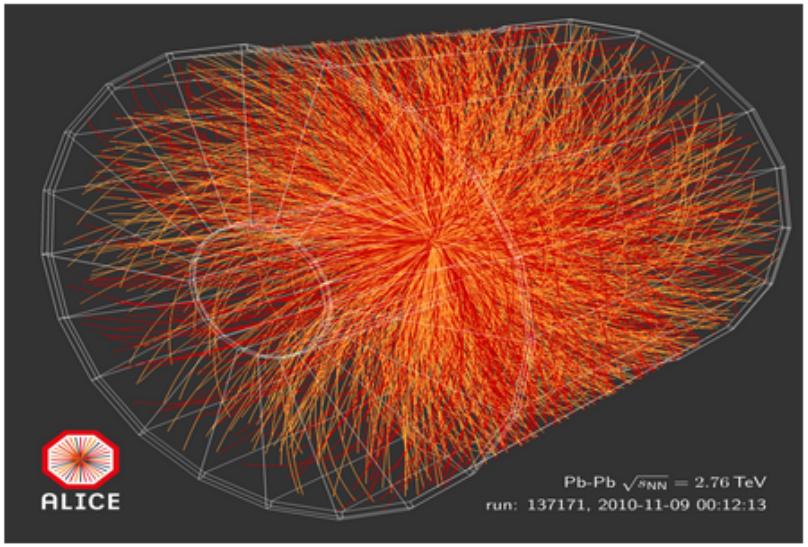
- This talk only presented some highlights
  → Machine Learning is much more!
- Fast growing field, a lot of research being done
- In physics, already used in many applications
- Might help us with striking problems
  - More precise analyses …
  - Fast simulation
  - Reliable QA



Take part:

- IML meetings
- Tutorials, further reading & getting started:
  https://github.com/iml-wg/HEP-ML-Resources

Result for Visual Question Answering



Was there a Quark-Gluon plasma formed in this collision?

Submit

Predicted top-5 answers with confidence:

yes	93.967%
no	6.033%
sun	0.000%
light	0.000%
many	0.000%

vqa.cloudcv.org

Thank you for your attention!

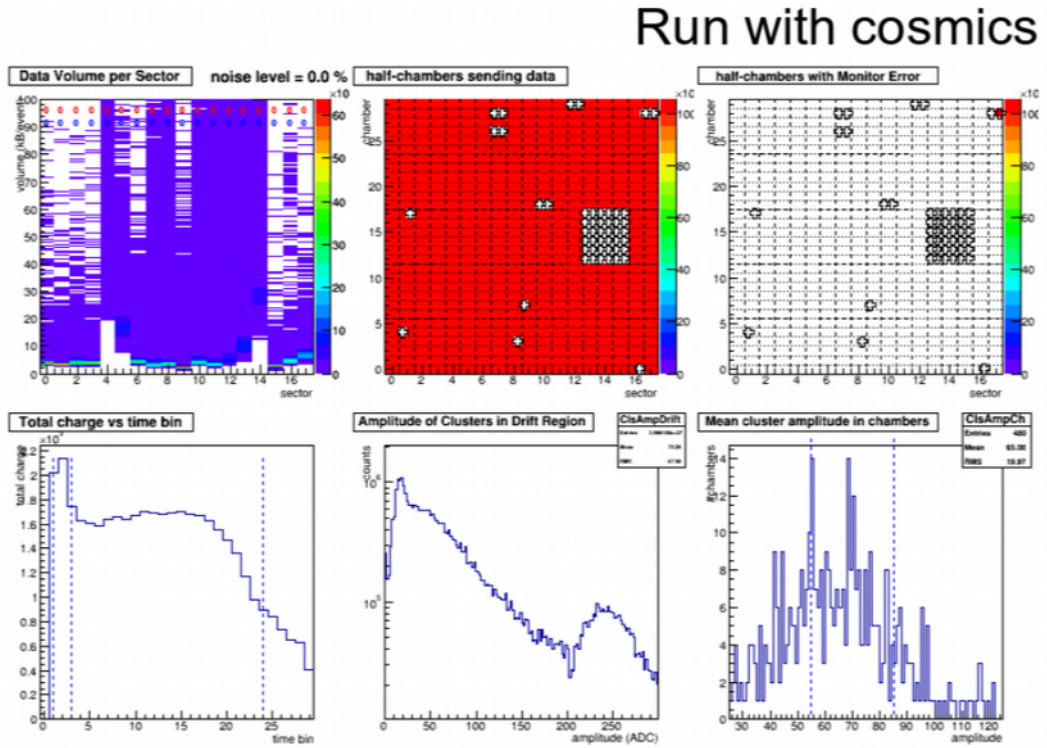
Backup

Applications in high energy physics: DQM/QA

DQM/QA

- Data Quality Management (DQM) and Quality Assurance (QA) still huge amount of work from experts

- Takes place during data taking (DQM) and shortly after (QA)
- Needs a lot of resources
- Even worse for LHC Run III, when much more data will be recorded
- Usual approach: Experts give flags to recorded runs using DQM/QA histograms

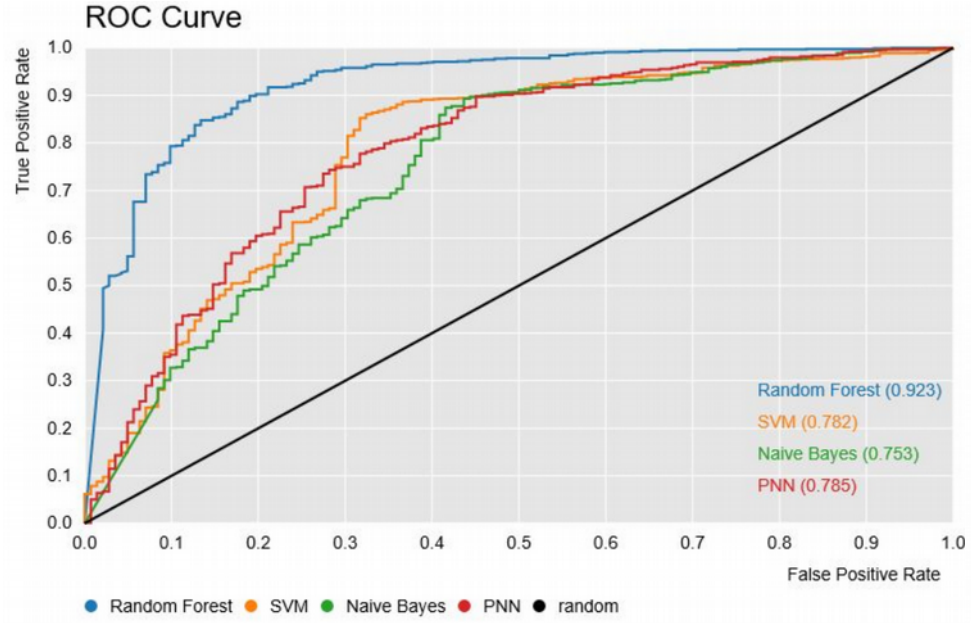


- Machine Learning can help with several aspects, e.g. anomaly detection & automatic data classification

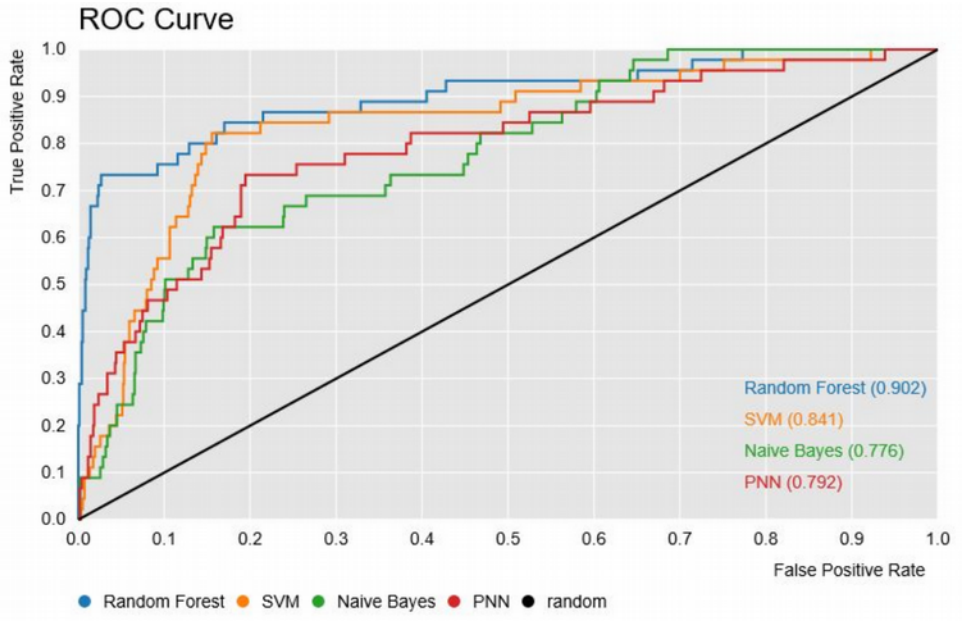
DQM/QA

- Current research approach uses more than 200 physics parameters from available QA parameters

Classification: Good



Classification: Bad



- First tests show that automatic/assisted classification is possible
- Also other approaches tested: GANs for anomaly detection, LSTM autoencoders for time series prediction (inspired by ECG anomaly detection)

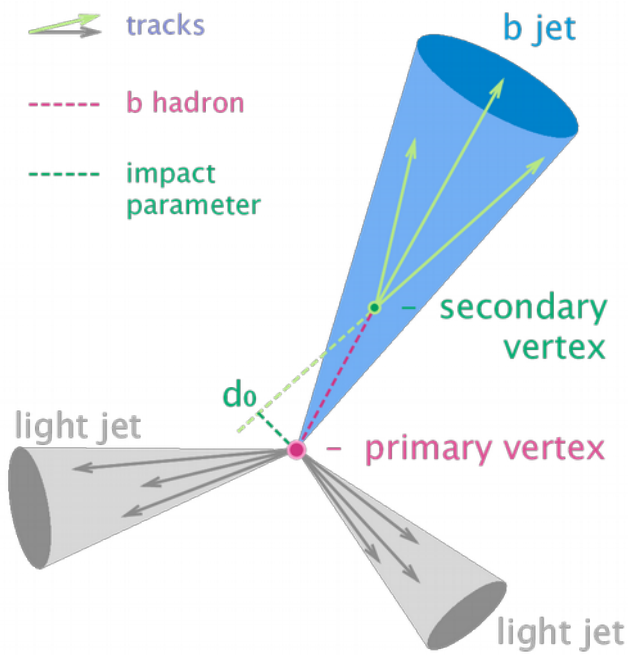
b-jet tagging in ALICE

Jets: Heavy-flavor jet tagging

- Jets from b-/c-quarks interesting probes in heavy-ion collisions
- In-medium modification of b-jets different to udsg-jets
  - Larger energy loss for gluons than quarks (color charge)
  - “Dead cone effect”: For massive quarks, gluon bremsstrahlung suppressed at smaller angles w.r.t. parton direction
- Approach in ALICE: deep learning tagger
- Exploit that B-hadrons decay in the (sub-)millimeter range
  - → displaced from primary vertex
  - → reconstruct secondary vertices

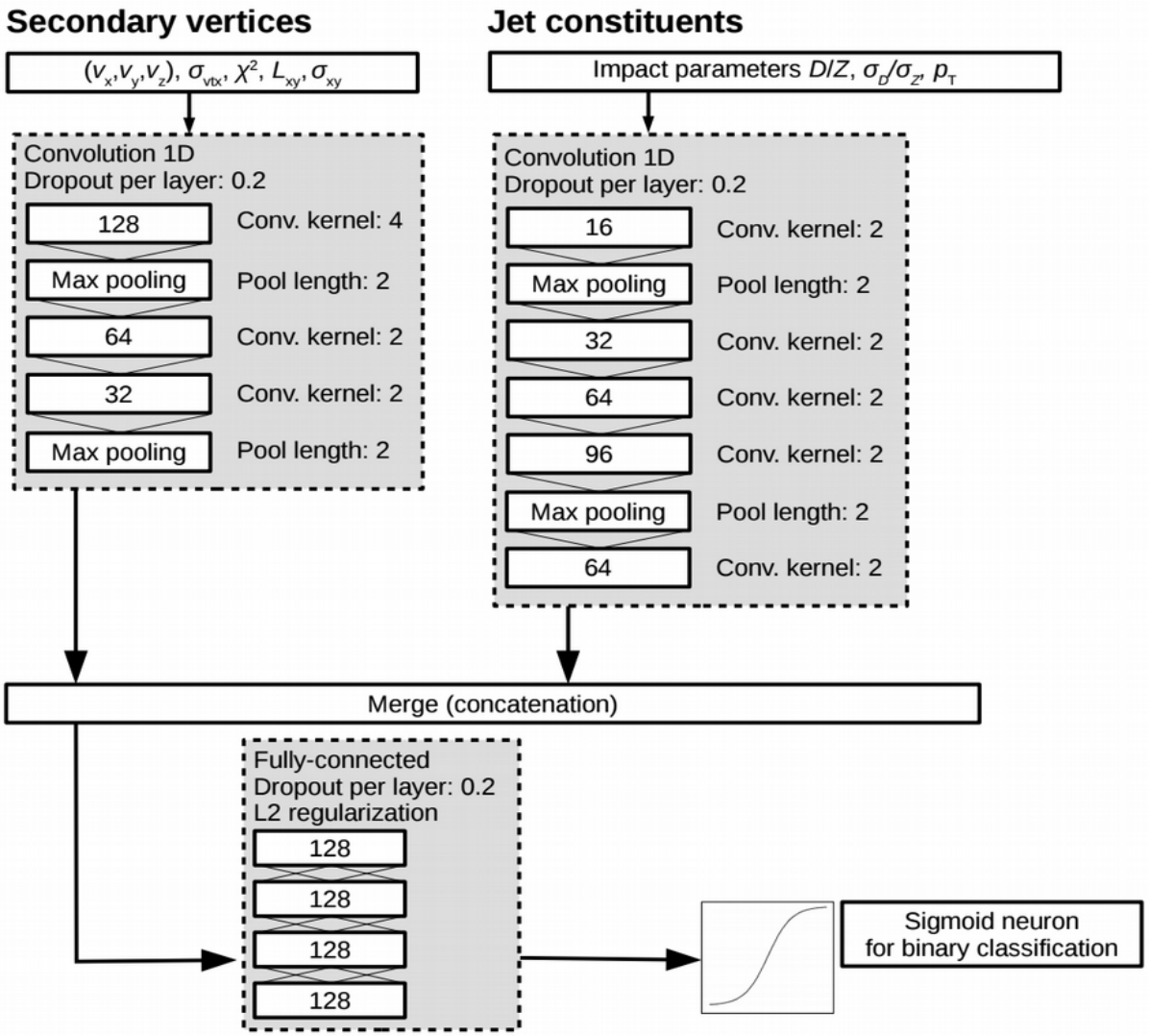
“Conventional” approach: Rectangular cuts on properties of most displaced vertices

Our ansatz: Apply neural network to several low-level input parameters



http://bartosik.pp.ua/hep_sketches/btagging

Jets: Heavy-flavor jet tagging



- Tagger uses two subnets which are merged
  - Secondary vertices
  - Track impact parameters
- 1D convolutional networks exploiting vertex or constituent relations
- Each subnet optimized via grid search, separately:
  "Clever brute force"
- **Powerful concept:**
  - Find suitable designs for available discriminators
  - Optimize separately
  - Merge with neural net