CCR - 14 June 2018

Machine Learning in HEP

D. Bonacorsi (University of Bologna)

Credits: all co-authors of the nature review on ML@HEP (out soon): A Radovic, D Rousseau, M Kagan,M Williams, A Himmel, A Aurisano, K Terao, T Wongjirad. Special thanks to M Pierini, S Glazer, V Kuznetsov, JR Vlimant (and more)

Where HEP stands about ML, and where ML itself stands

Pioneers:

- pioneering studies employing ML at previous-generation experiments laid the groundwork for the emergence of ML as an essential tool (e.g. at the LHC, but not only)

Past two <u>decades</u> (*HEP timescales..*):

- HEP has been migrating towards the use of ML methods in collection and analysis of large data samples
- ML algorithms made important contributions (e.g. to the discovery of the Higgs boson). Nowadays, plenty of data-analysis tasks (and not only) benefit from the use of ML

Past few <u>years</u> (*non-HEP timescales..*):

- in parallel, the ML field has developed at a rapid pace, and in particular the "new" subfield of DL has delivered genuinely superhuman performance in a number of domains

Cross-discipline (cultural and technical) bridging

- Incorporating these new tools while maintaining the scientific rigour required in particle physics analyses presents some unique (not only technical!) challenges and opportunities

"Which" ML for HEP?

Very wide field of **supervised ML** (mostly), e.g. training algorithms to classify data as signal or background by studying existing labeled (possibly Monte Carlo) data.

Typical ML workflow in HEP? (simplified..)

- problem statement and data preparation: variables relevant to the physics problem are selected, data are cleansed, etc
- training: e.g. a ML model is trained for *classification* using signal and background events (the most human- and CPU- time consuming)
- inference: relatively inexpensive

Typical ML algorithm for HEP?

- a large plethora of categories of algorithms to even attempt to list them here
- mostly: Boosted Decision Trees (**BDT**s) and Artificial Neural Networks (**ANN**s)
- then, expanding from these to more..

More on ML algorithms in HEP

BDTs/**ANN**s typically used to classify particles and events

- they are also used for regression, e.g. to obtain the best estimate of particle's energy based on the measurements from several detectors

ANNs being used for a while in HEP, then.. → rise of **DNN**s

- particularly promising when there is a large amount of data and features, as well as symmetries and complex non-linear dependencies between inputs and outputs

Different types of NNs used in HEP:

- fully-connected (**FCN**), convolutional (**CNN**), recurrent (**RNN**) network
- additionally, NNs are used in the context of Generative Models, when a NN is trained to mimic multidimensional distributions to generate any number of new instances. Variational AutoEncoders (**VAE**s) and more recent Generative Adversarial Networks (**GAN**s) are two examples of such generative models used in HEP.

Plus, ML algorithms devoted to time-series analysis and prediction

- in general not relevant for HEP where events are independent from each other
- however, growing interest in these algorithms for HEP-related sequential non-collision data, e.g. for Data Quality and Computing Infrastructure monitoring (as well as those physics processes and event reconstruction tasks where time is an important dimension)

HEP as ML consumers, not producers

At a first approximation, most ML usage in HEP is <u>not</u> ML research

- HEP community is being building domain-specific applications on top of existing toolkits and ML algorithms developed by computer scientists, data scientists, and scientific software developers from outside the HEP world

Work is also being done to understand where HEP problems do not map well onto existing ML paradigms and how these problems can be recast into abstract formulations of more general interest

Frameworks and tools

HEP-physicists nowadays mostly use TMVA in ROOT.

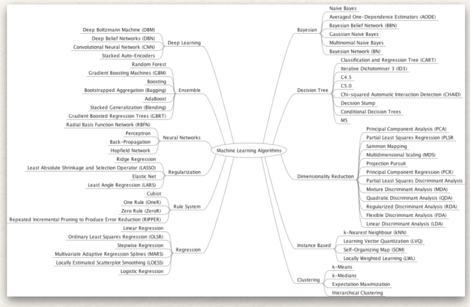
Non-HEP scientists (and not only!) use e.g.:

PYTORCH TensorFlow scikits learn machine learning in Python R Caffe
mxnet Caffe2
K Keras theano APACHE Spark Microsoft CNTK DL4J

HEP not at all closed to the worldwide approach to ML: **quite the opposite!**

- from Adaboost DT or Multilayer Perceptron NN (MLP), to state-of-the-art XGBoost DT or Deep NN (e.g. Tensorflow) or GANs or …

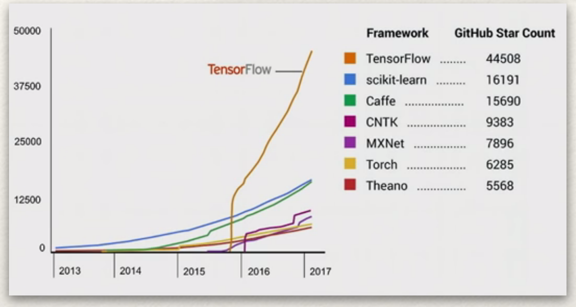
Opportunities and challenges



Machine Learning Algorithms
Deep Learning
Deep Boltzmann Machine (DBM)
Deep Belief Networks (DBN)
Convolutional Neural Network (CNN)
Stacked Auto-Encoders
Ensemble
Random Forest
Gradient Boosting Machines (GBM)
Boosting
Bootstrapped Aggregation (Bagging)
AdaBoost
Stacked Generalization (Blending)
Gradient Boosted Regression Trees (GBRT)
Neural Networks
Radial Basis Function Network (RBFN)
Perceptron
Back-Propagation
Hopfield Network
Regularization
Ridge Regression
Least Absolute Shrinkage and Selection Operator (LASSO)
Elastic Net
Least Angle Regression (LARS)
Rule System
Cubist
One Rule (OneR)
Zero Rule (ZeroR)
Repeated Incremental Pruning to Produce Error Reduction (RIPPER)
Regression
Linear Regression
Ordinary Least Squares Regression (OLSR)
Stepwise Regression
Multivariate Adaptive Regression Splines (MARS)
Locally Estimated Scatterplot Smoothing (LOESS)
Logistic Regression
Bayesian
Naive Bayes
Averaged One-Dependence Estimators (AODE)
Bayesian Belief Network (BBN)
Gaussian Naive Bayes
Multinomial Naive Bayes
Bayesian Network (BN)
Decision Tree
Classification and Regression Tree (CART)
Iterative Dichotomiser 3 (ID3)
C4.5
C5.0
Chi-squared Automatic Interaction Detection (CHAID)
Decision Stump
Conditional Decision Trees
M5
Dimensionality Reduction
Principal Component Analysis (PCA)
Partial Least Squares Regression (PLSR
Sammon Mapping
Multidimensional Scaling (MDS)
Projection Pursuit
Principal Component Regression (PCR)
Partial Least Squares Discriminant Analysis
Mixture Discriminant Analysis (MDA)
Quadratic Discriminant Analysis (QDA)
Regularized Discriminant Analysis (RDA)
Flexible Discriminant Analysis (FDA)
Linear Discriminant Analysis (LDA)
Instance Based
k-Nearest Neighbour (kNN)
Learning Vector Quantization (LVQ)
Self-Organizing Map (SOM)
Locally Weighted Learning (LWL)
Clustering
k-Means
k-Medians
Expectation Maximization
Hierarchical Clustering

Abundance: the number of ML algos and implementations in a growing variety of frameworks and libraries

- drawback: difficult and time-consuming to evaluate tradeoffs for using one ML "tool" compared to another, and also tradeoffs for ML vs non-ML solutions



Advancement: extremely quick

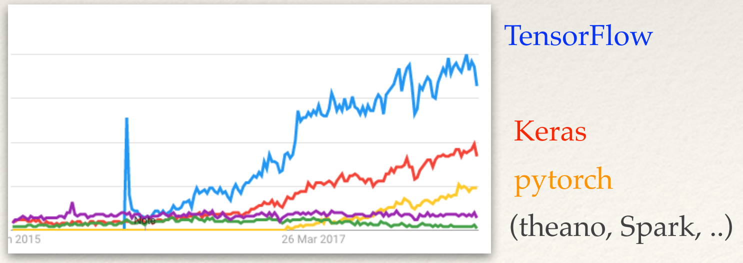
- drawback: HEP research teams need to investigate the numerous approaches at hand, adequate skills are needed to follow up, complexity requires not-best-effort engagements

Open-source and code accessibility, documentation, training: the portfolio of ML techniques and tools is in constant evolution, many have well-documented open source software implementations, often supported by MOOCs, etc

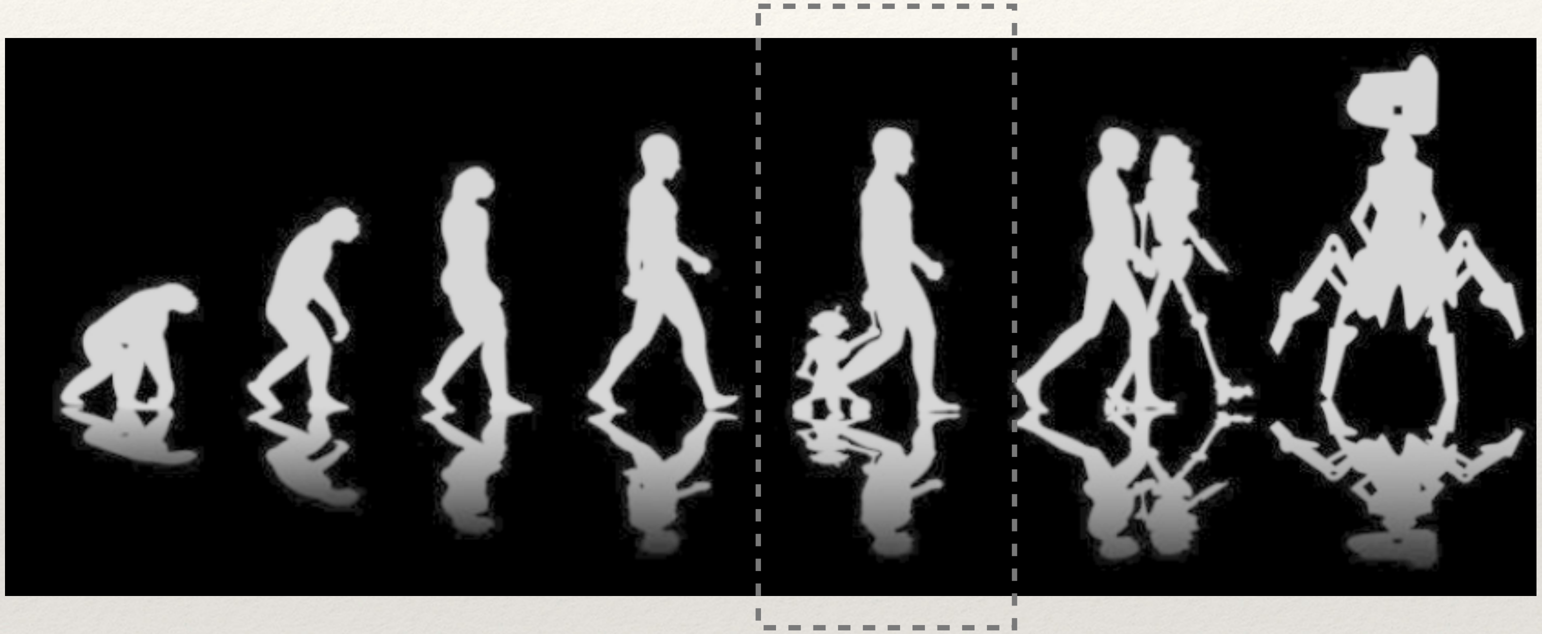
- drawback: acquired expertise and lessons learned by few people risks to get lost before being adequately disseminated to a wider community + issues in adequate training of young HEP collaborators

Hype: guarantees attention and investments

- drawback: overhyped?!



Up to now..



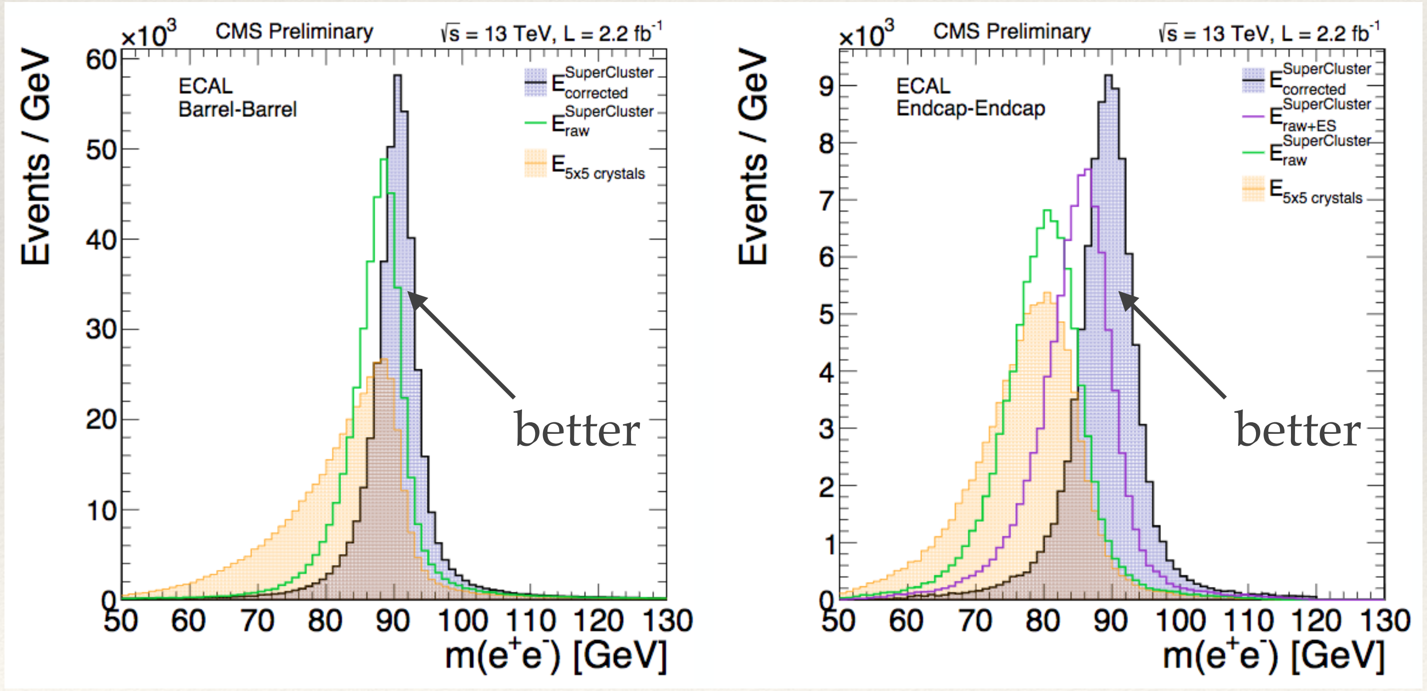
ML in HEP as the use of field-specific knowledge for feature engineering

i.e. use physicist-designed high-level features as input to shallow algorithms

Particle properties: energy resolution

Using ML to improve the determination of particle properties is now commonplace in **all LHC experiments**

- E.g. energy deposited in calorimeters is recorded by many sensors, which are clustered to reconstruct the original particle energy. **CMS** is training **BDT**s to learn corrections using all information available in the various calorimeter sensors - thus resulting in a sizeable improvement in resolution



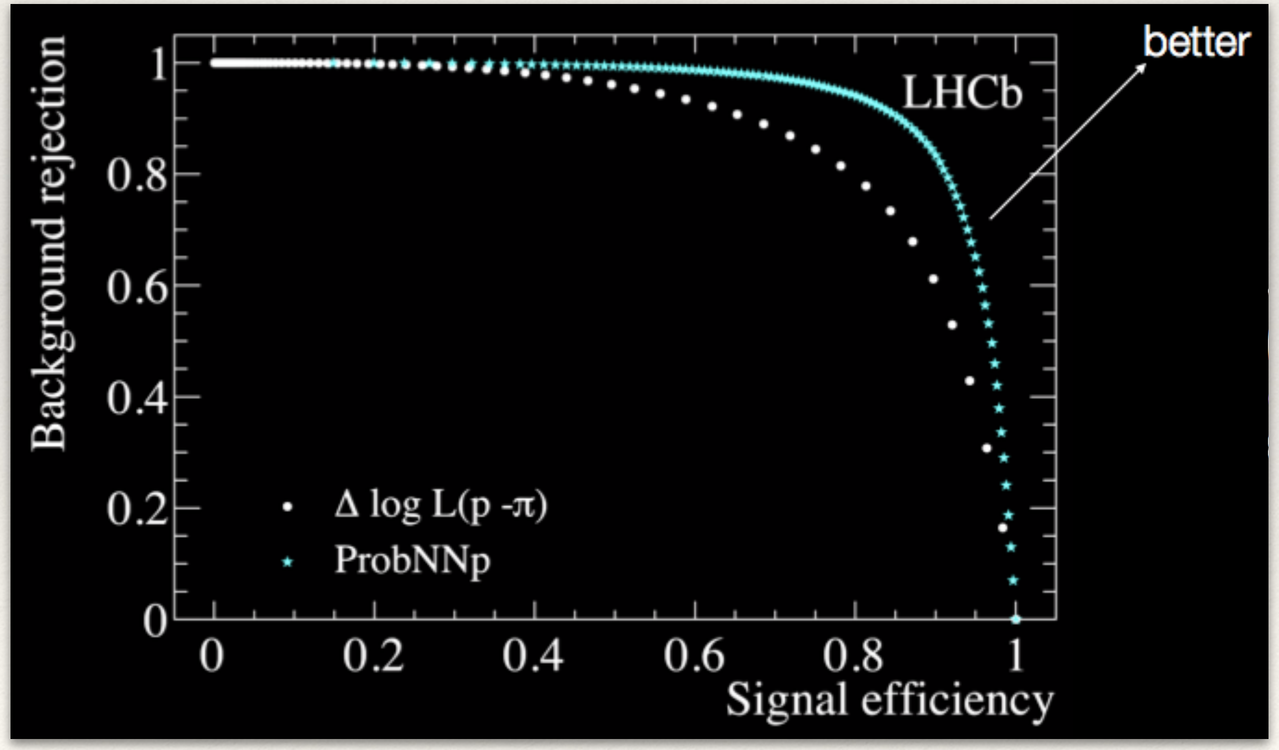
Improvements to the Z→e+e- energy scale and resolution from the incorporation of more sophisticated clustering and cluster correction algorithms (energy sum over the seed 5x5 crystal matrix, bremsstrahlung recovery using supercluster, inclusion of pre-shower energy, **energy correction using a multivariate algorithm**)

[2015 ECAL detector performance plots, CMS-DP-2015-057. Copyright CERN, reused with permission]

Particle ID

Similarly, ML is commonly used to identify particle types

- e.g. **LHCb** uses **NN**s trained on O(30) features from all its subsystems, each of which is trained to identify a specific particle type
- ~3x less mis-ID bkg /particle. Estimates indicate that more advanced algorithms may reduce bkg by another ~50%

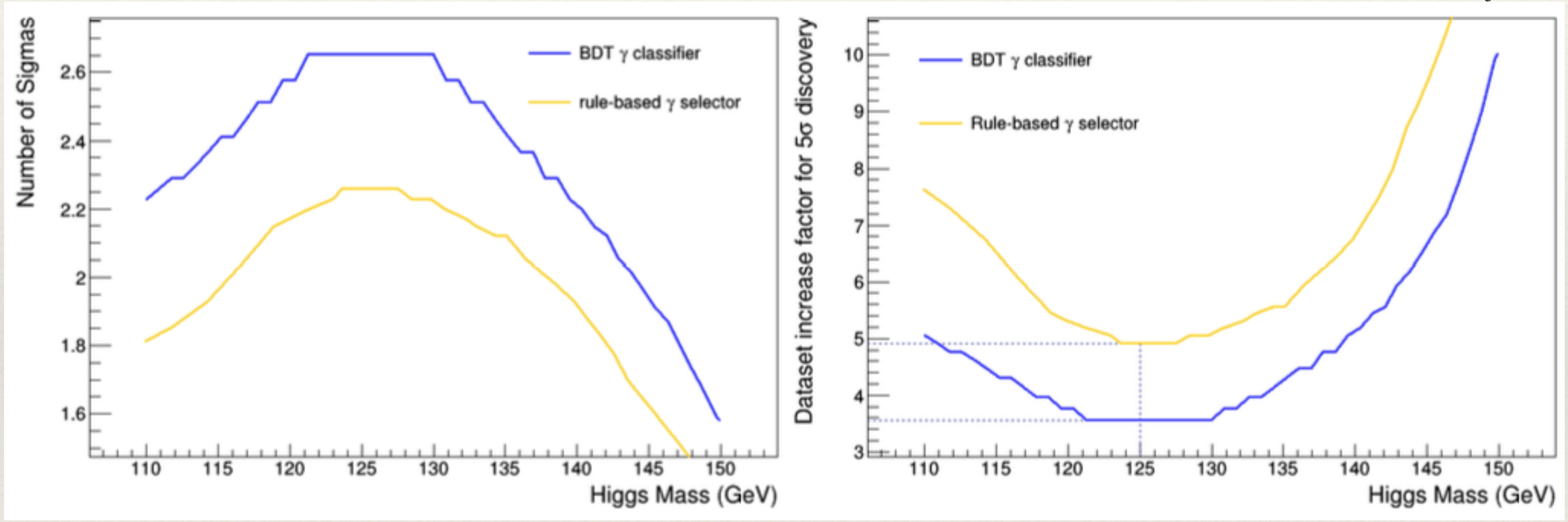


[courtesy: M.Williams]

Discovery of the Higgs boson

ML played a key role in the discovery of the Higgs boson, especially in the diphoton analysis by **CMS** where ML (used to improve the resolution and to select/categorize events) <u>increased the sensitivity by roughly the equivalent of collecting ~50% more data</u>.

[courtesy M.Pierini]



We were not supposed to discover the Higgs boson as early as 2012

- Given how machine progressed, we expected discovery by end 2015 / mid 2016

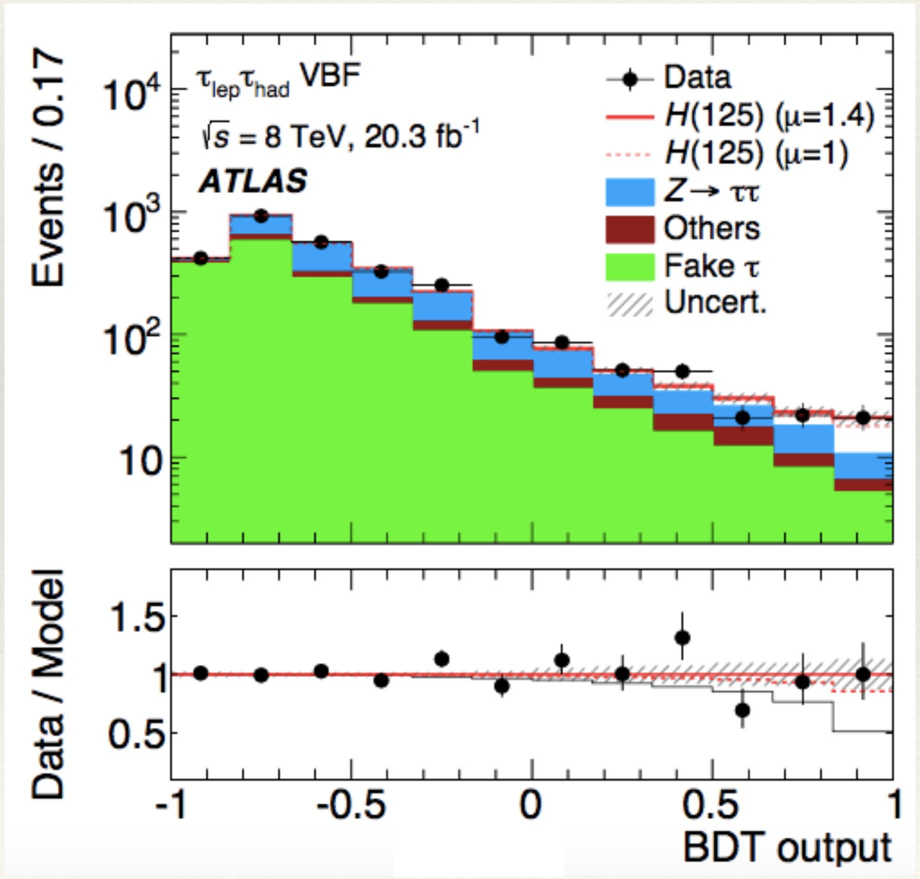
We made it earlier thanks (also) to ML

Study of Higgs properties

E.g. analysis of τ leptons at LHC complicated as they decay before being detected + loss of subsequently produced neutrinos + bkg from Z decays

- e.g. **ATLAS** divided the data sample into 6 distinct kinematic regions, and in each a **BDT** was trained using 12 weakly discriminating features → improved sensitivity by ~40% vs a non-ML approach

Also part of the 2014 Higgs ML **Kaggle** challenge

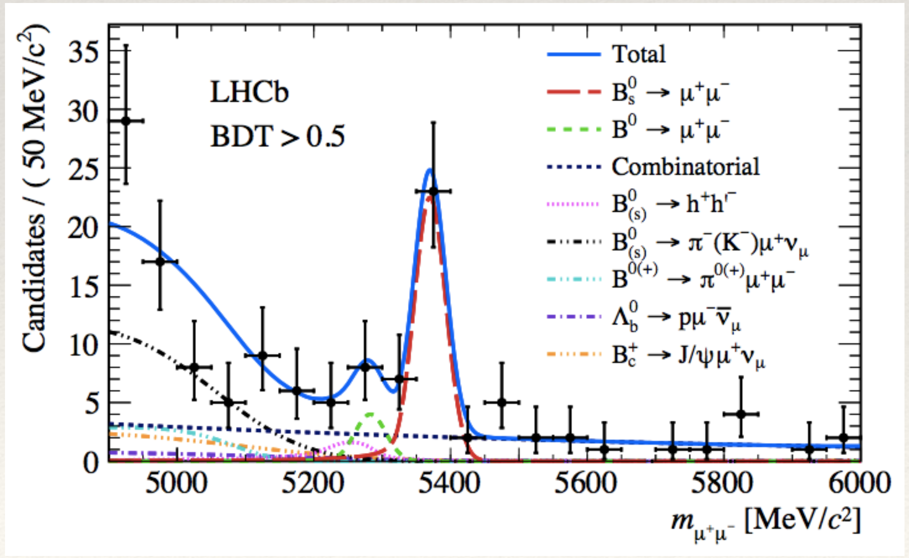


[arXiv:1501.04943]

High-precision tests of the SM

CMS and **LHCb** were the first to find evidence for the $B^0_s \rightarrow \mu^+\mu^-$ decay with a combined analysis (as rare as ~ 1 / 300 billion pp collisions..)

- **BDT**s used to reduce the dimensionality of the feature space - excluding the mass - to 1 dimension, then an analysis was performed of the mass spectra across bins of BDT response
- decay rate observed is consistent with SM prediction with a precision of ~25%, placing stringent constraints on many proposed extensions to the SM
- To obtain the same sensitivity without ML by LHCb as a single experiment would have required ~4x more data. Just one of many examples of high-precision tests of the SM at the LHC where ML can dramatically increase the power of the measurement



Mass distribution of the selected $B^0 \rightarrow \mu^+\mu^-$ candidates with BDT > 0.5.

[arXiv: 1703.05747]

Trigger

Crucial trade-off in algorithm complexity and performance under strict inference time constraints

E.g. **ATLAS**/**CMS** each only keep about 1 in every 100 000 events, and yet their data samples are each still about 20 PB/yr

- ML algorithms have already been used very successfully for rapid event characterisation
- adoption depth vary across experiments, but the increasing event complexity at HL-LHC will require more sophisticated ML solutions and its expansion to more trigger levels

A critical part of this work will be to understand which ML techniques allow us to maximally exploit future computing architectures

Trigger (cont'd)

E.g. **CMS** employs ML in its trigger hardware to better estimate the momentum of muons

- inputs to the algorithm are discretised to permit encoding the ML response in a large look-up table that is programmed into FPGAs

E.g. **LHCb**, many of the reactions of greatest interest do not produce striking signatures in the detector, making it necessary to thoroughly analyse high-dimensional feature spaces in real time to efficiently classify events

- LHCb used a **BDT** for 2 years, then a MatrixNet algorithm
- ML now almost ubiquitous in LHCb Trigger. 70% of all persisted data is classified by ML algorithms. All charged-particle tracks are vetted by **NN**s.
- LHCb estimated that reaching the same sensitivity as a recent LHCb search for the dark matter on 2016 data, would have required collecting data for 10 yrs without the use of ML

Tracking

Pattern recognition has always been a computationally challenging step

- e.g. the HL-LHC environment makes it an extremely challenging task

Adequate ML techniques may provide a solution that scales linearly with LHC intensity.

Several efforts in the HEP community have started to investigate sophisticated ML algorithms for track pattern recognition on many-core processors.

No time to cover all this area in details..

Computing resource optimisations

Industrial-scale data samples collected by e.g. LHC experiments produce non-collisions metadata from which actionable insights can be extracted

- results of logging while running Run-1/2 operations of complex WM and DM systems

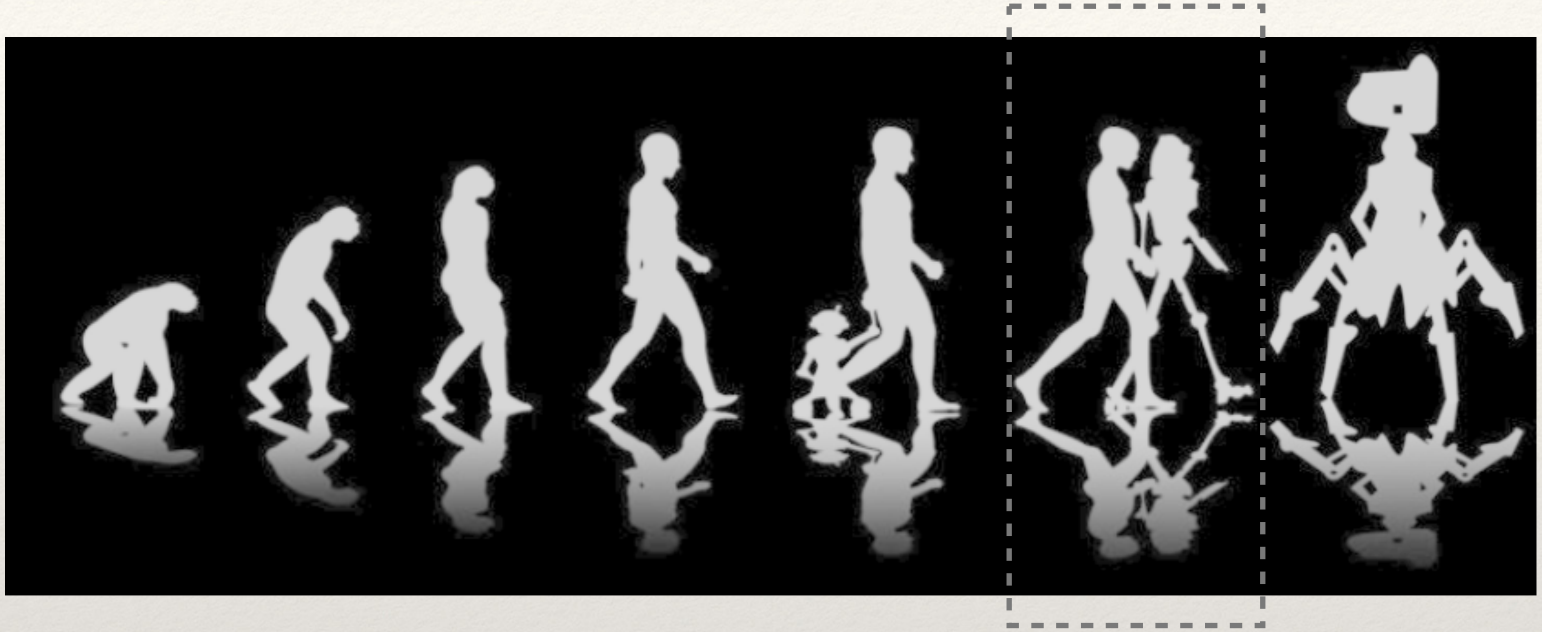
ML techniques have begun to play a crucial role in increasing the efficiency of computing resource usage for LHC experiments since few years

- e.g. predicting which data will be accessed the most to a-priori optimise data storage at Grid computing centres via pre-placement, or perform WAN path optimisation based on user access historical patterns (done/in-progress primarily, but not only, in **LHCb** and **CMS**)
- e.g. monitoring data transfer latencies over complex network topologies, using ML to identify problematic nodes and predict likely congestions (in progress by **CMS**)

Current approach is that ML informs the choices of the computing operations teams

- this might be the basis of <u>fully adaptive models</u> in the next future

Now to ~2020/22?



ML in HEP as **the use of full high-dimensional feature space to train cutting-edge ML algorithms (e.g. DNNs)**

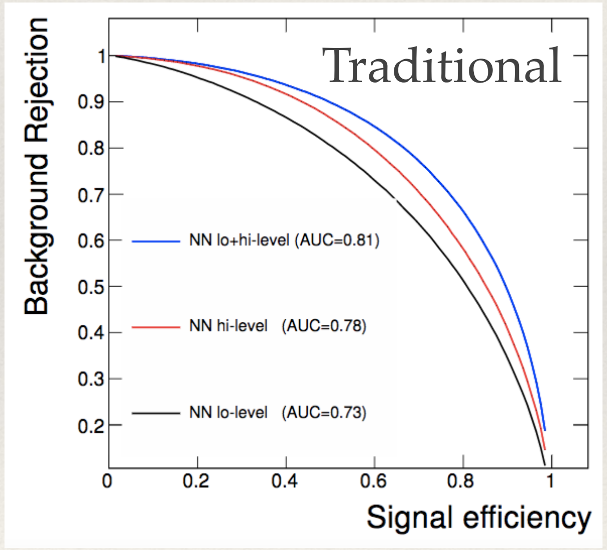
As in computer vision and NLP, growing effort in HEP too to skip the feature-engineering step. How well can we do using deeper networks and/or special architectures?

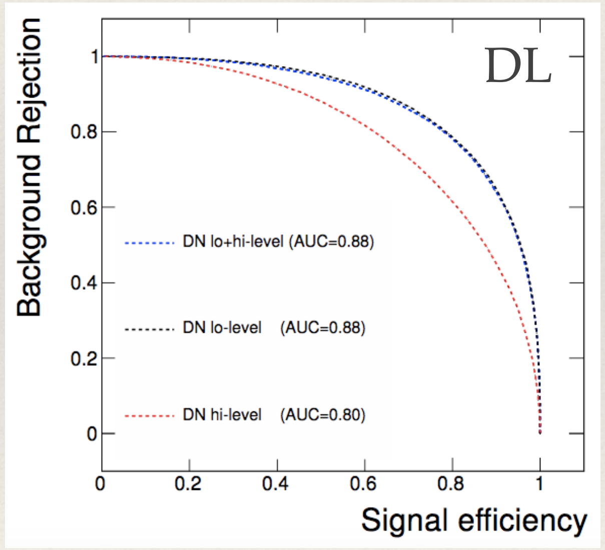
Do DNNs need us?

Does a DNN need high-level features like invariant masses, or can it just learn the physics by itself from the 4-vectors (once it is given examples)?

- If a DNN using low-level features outperforms any selection based only on high-level features..

ML models with limited capacity to learn complex non-linear functions of the inputs rely on painful manual construction of helpful non-linear feature combinations to guide the shallow networks. But recent DL advancements allow to automatically discover powerful non-linear feature combinations, thus providing better discrimination power.





Higgs benchmark: comparison of bkg rejection vs signal efficiency for the traditional learning method (left) and the DL method (right) using the low-level features, the high-level features and the complete set of features.

[arXiv:1402.4735]

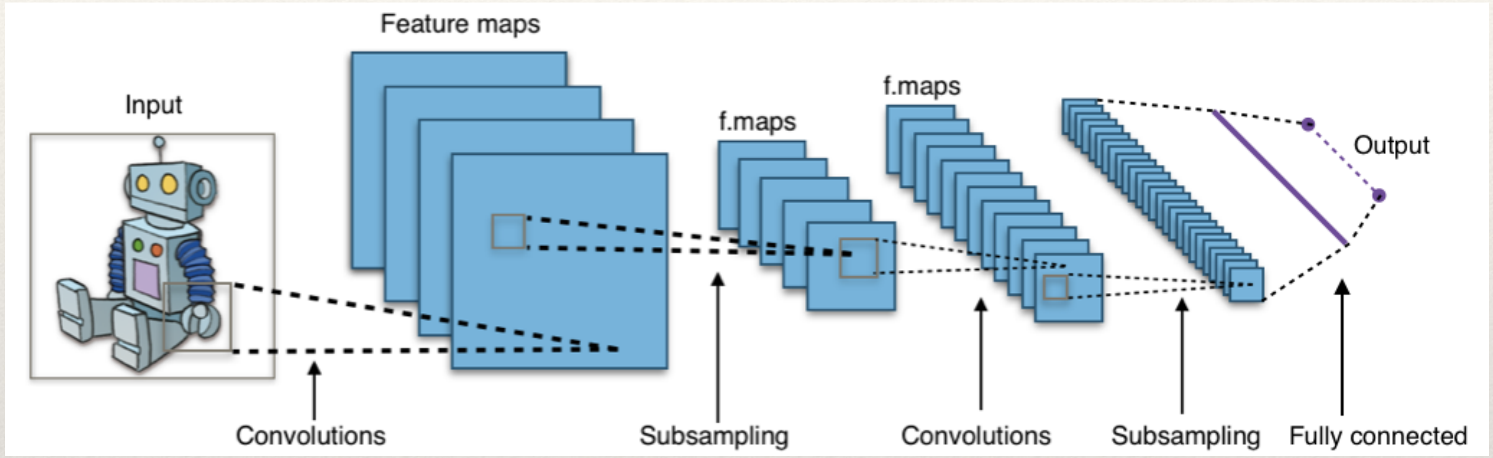
Demonstrated improvements O(~10%) over the best current approaches

- DL techniques can provide powerful boosts to searches for exotic particle

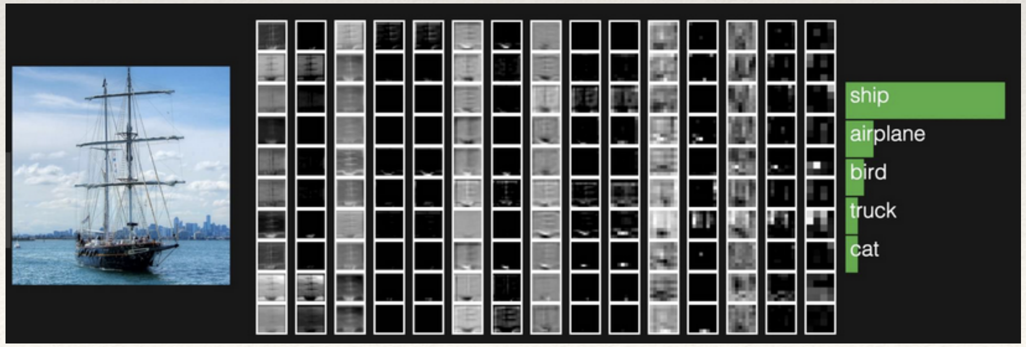
CNNs

CNNs are deep FFNNs with architecture inspired by the visual cortex

- CNN neurons seek local examples of translationally invariant features. Convolutional filters locate patterns producing maps of simple features. Complex features are built using many layers of simple feature maps.



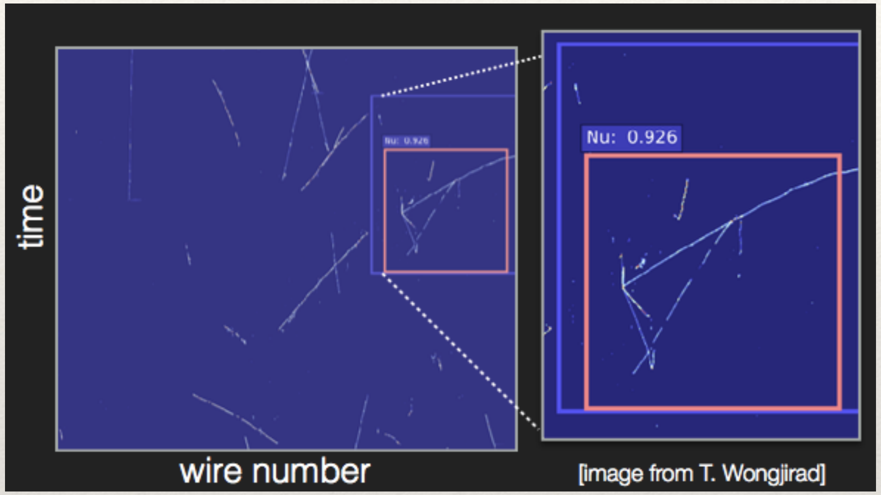
Used to solve a large variety of problems, including many in image recognition



CNNs for neutrinos

MicroBooNE has managed to train **CNN**s that can locate neutrino interactions within an event in the LArTPC, identify objects and assign pixels to them

- CNN perfect to identify objects in an image (translational invariant feature learning), and sensitive volumes are large due characteristics of neutrino interaction with matter

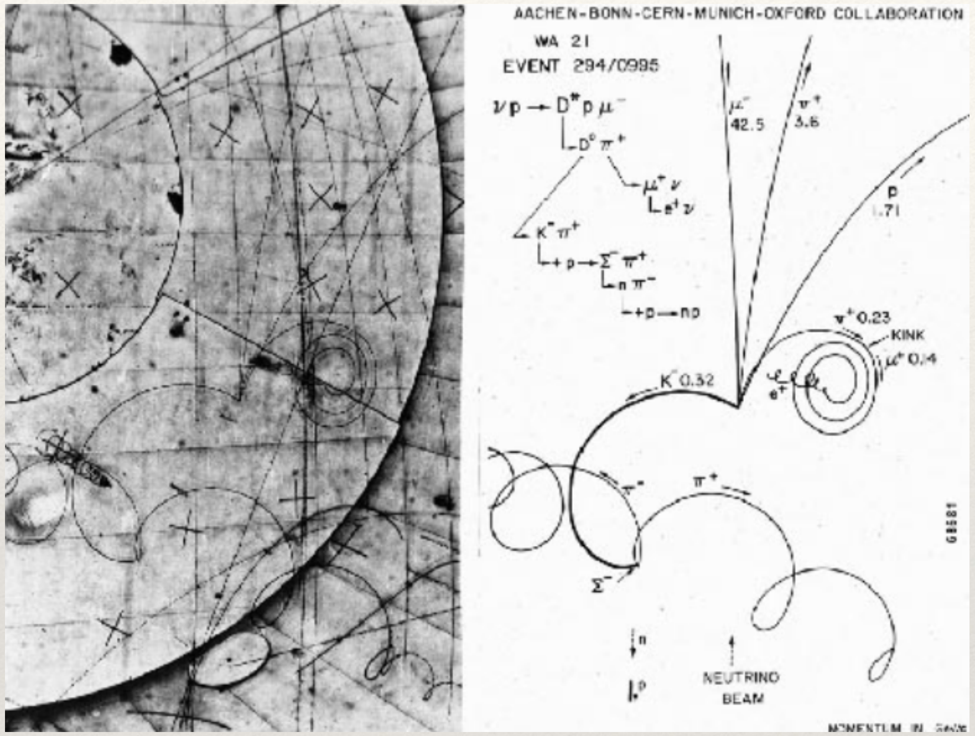


[more at arXiv:1611.05531]

Similar work ongoing at:

- other neutrino experiments - e.g. **NOvA** *[arXiV:1604.01444]*
    - inspired to GoogLeNet architecture. Improvement in the efficiency of selecting electron neutrinos by 40% with no loss in purity. Used as event classifier in both an electron neutrino appearance search, and in a search for sterile neutrinos
- collider experiments in the area of jet physics *[arXiv:1511.05190]* *[arXiv:1603.09349]*

Does this remind you of something?



Neutral currents in BEBC - WA21 CC Charm Event: Roll 204, Frame 995 [CERN]

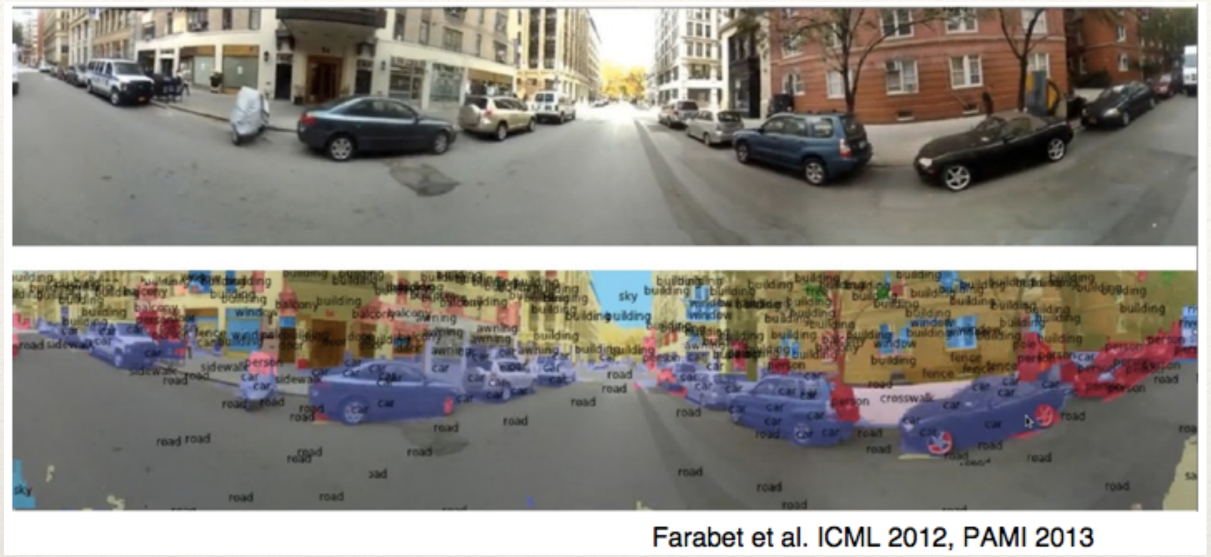
The data taking pace has changed

- e.g. BEBC in 1973-83 equals to 6 seconds of (e.g.) LHCb today
- e.g. LHC sensor arrays's 1 hr equals to ~ Facebook data in 1 year
- algorithms running on large computing farms took over long ago

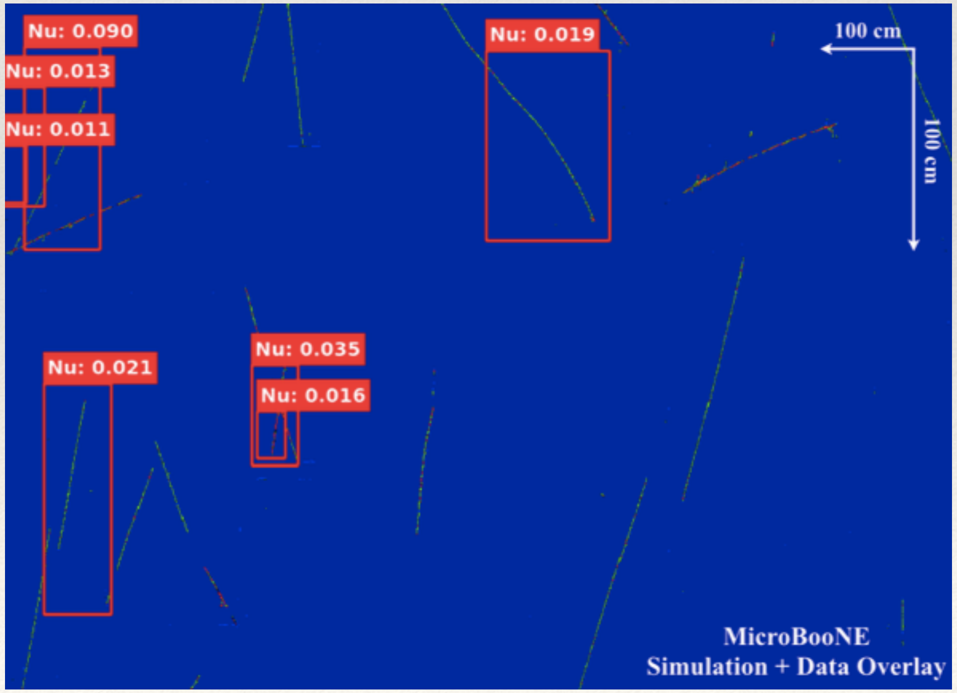
Still dealing with inability for humans to visually inspect vast amounts of data

- Indeed, inability "for humans"..

Arguing "HEP is different"..


Farabet et al. ICML 2012, PAMI 2013

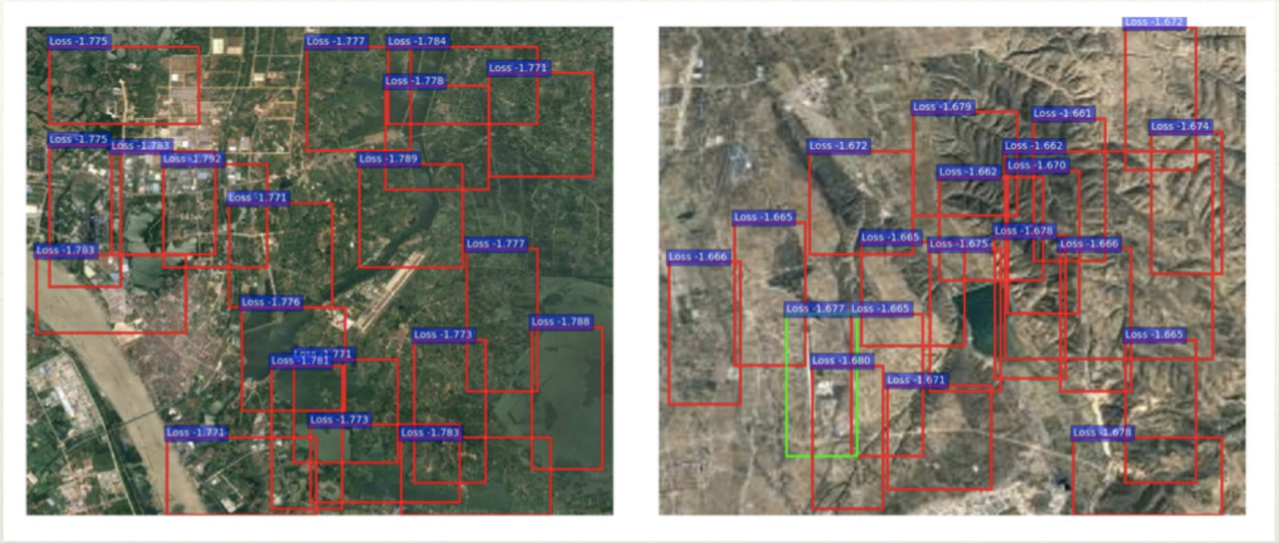
Scene labelling
in automotive applications



MicroBooNE examples of cosmic bkg events with detected neutrino bounding boxes with low scores.

[arXiv:1611.05531]

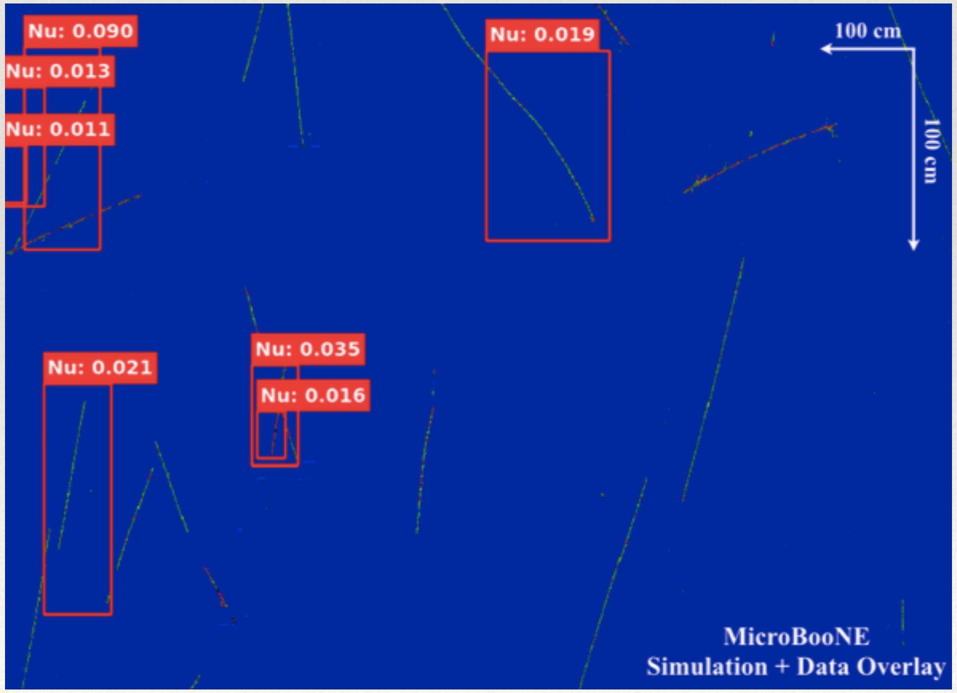
Arguing "HEP is different"..



[Remote Sens. **2017**, 9, 1198; doi:10.3390/rs9111198]

Airports detection from satellite images with CNNs

MicroBooNE examples of cosmic bkg events with detected neutrino bounding boxes with low scores.



[arXiv:1611.05531]

More on particle ID and particle properties

In CALOs or TPCs the data can be represented as a 2D or 3D image (even 4D, including timing information): the problem can be cast as a computer vision task.

DL techniques in which DNNs are used to reconstruct images from pixel intensities are good candidate to identify particles and extract many parameters

- promising DL architectures for these tasks include (at least) **CNN**, **RNN**
- e.g. LArTPCs is the chosen detection technology for **DUNE** (the new flagship experiment in the neutrino programme). A proof of concept and comparison of various DL architectures is expected to be finalised by 2020
- e.g. **b-tagging in collider experiments**. Techniques also from NLP are expected to be finalised by 2020

Simulation

Physics-based full simulation modelling in HEP (with GEANT 4 as the state of the art) is very computationally demanding

- e.g. for LHC, the large samples to be generated for future experimental runs and the increase in luminosity will exacerbate the problem, prohibitive also for GEANT

This already sparked the development of approximate, Fast Simulation solutions to mitigate this computational complexity - especially relevant in calorimeter showers simulations

Promising alternatives for Fast Simulation may be built on recent progress in high fidelity fast generative models

- e.g. Generative Adversarial Networks (**GAN**s) and Variational AutoEncoders (**VAE**s)
- ability to sample high dimensional feature distributions by learning from existing data samples

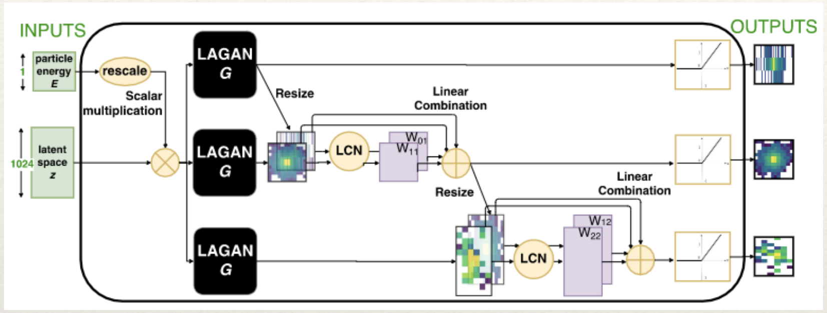
A simplified first attempt at using such techniques in simulation saw orders of magnitude increase in speed over existing Fast Simulation techniques, of which **all HEP experiments** would largely benefit

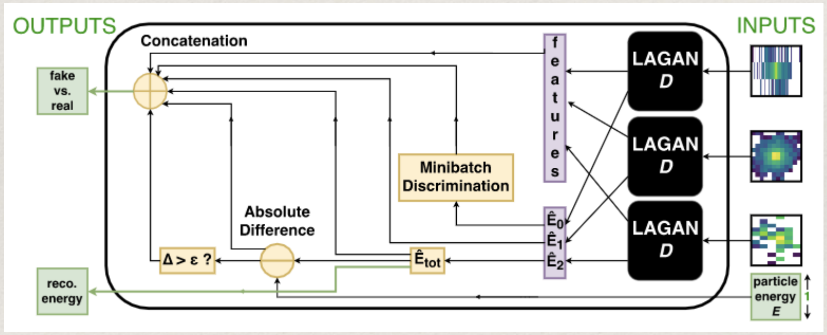
- not yet reached the required accuracy, though

Perhaps more towards >2020, but promising.

Simulation (cont'd)

E.g. exploit **GAN**s, a 2-NN game where one maps noise to images, and the other classifies the images as real vs fake (the best generator is the one that maximally confuses its adversary)





CaloGAN composite generator (up) and discriminator (down)

[arXiv:1712.10321]

E.g. **CaloGAN**, a new FastSimulation technique, to simulate 3D HEP showers in multi-layer ECAL systems with GANs

- basically, CaloGAN can generate the reconstructed Calo image using random noise, <u>skipping the GEANT and RECO steps - thus making it 10k faster than GEANT</u>..

HEP data format for ML

(might look ML-unrelated, but it deeply is)

HEP relies on the ROOT format for its data, whereas the ML worldwide community has developed several other formats (often associated with specific ML tools)

A desirable data format for offline usage with ML world-class applications and frameworks should have the following attributes:

- high read-write speed for efficient training
- sparse readability without loading the entire dataset into RAM
- high compressibility
- widespread adoption by the ML community

The thorough evaluation of the different data formats and their impact on ML performance for **all HEP experiments** is in progress.

- Strategy for bridging/migrating HEP formats to chosen ML format(s), or vice-versa, are being envisioned.

ML as-a-Service (MLaaS)

ML in production at scale is not only matter of software algorithms. Actually, it is mostly matter of infrastructure.

MLaaS emerging also in HEP as a possible range of services that offer ML tools as part of cloud computing services

- no need to install software or provision owned servers: the provider's data centres handle the actual computation

Not at all widely used in HEP, but first interesting attempts by pioneering experiments are appearing

- e.g. **CMS** has a working prototype of **TensorFlow-as-a-service (TFaaS)**, demonstrated for S/B discrimination in full hadronic top analyses, for event classification, etc.

Range of potential benefits:

- a plethora of trained models loadable and servable upon request
- optimal for prototyping, use of checkpoints, etc
- *an explorable "work model" for HEP: outsource the CS (ML) part of the work in a physics analysis team to a skilled sub-set of members + cloud resources*

Detector anomaly detection

WARNING: just one example of unsupervised..

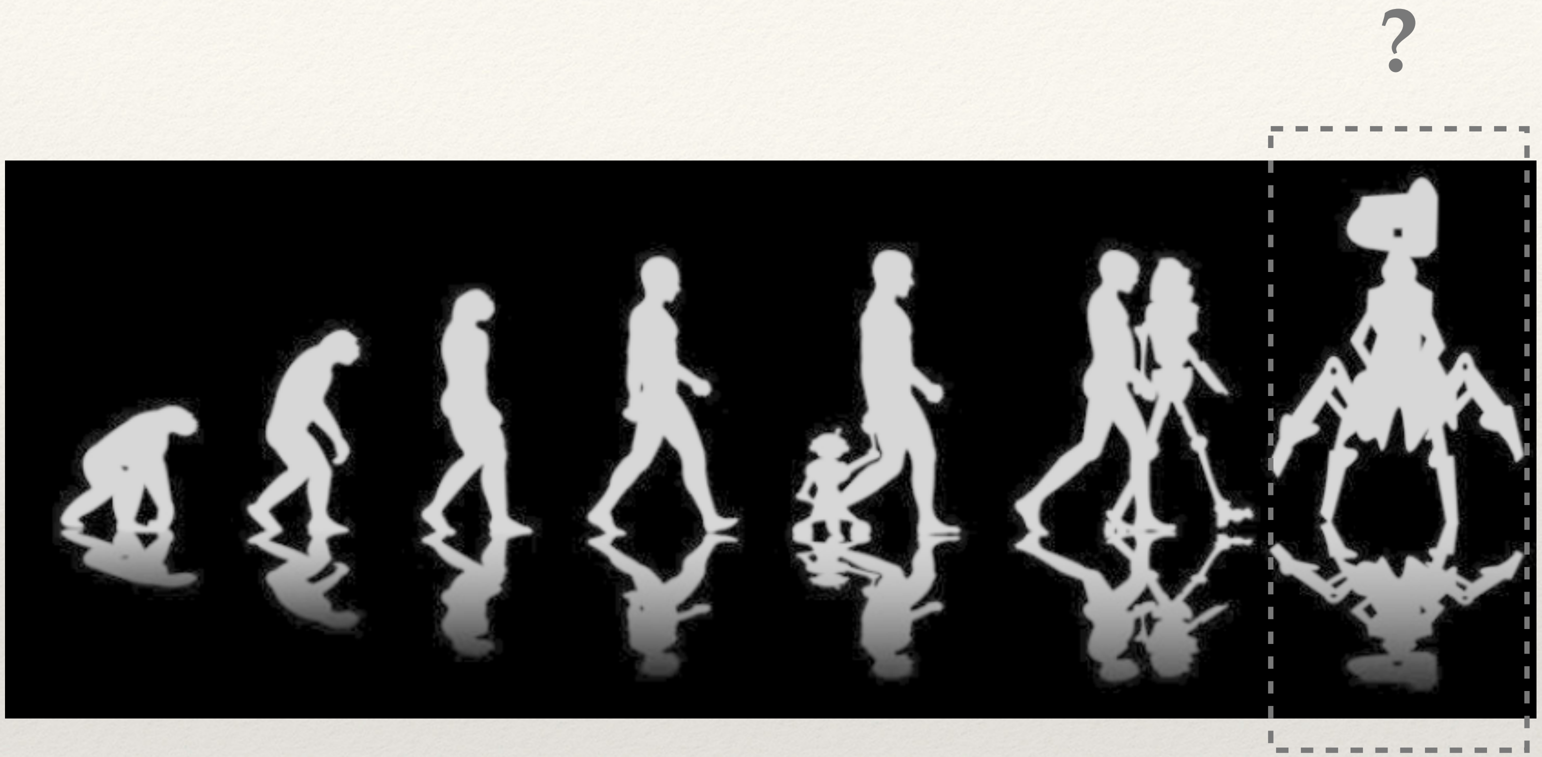
Data taking continuously monitored by physicists taking shifts to monitor and assess the quality of the incoming data

- largely using reference histograms produced by experts

Automation may come from the whole class of "**anomaly detection**" ML algorithms

- **unsupervised algorithms** able to monitoring many variables at the same time, learn from data and produce an alert when deviations are observed
    - synergy with **predictive maintenance** in industry: algorithms sensitive to subtle signs forewarning of imminent failure, so that pre-emptive actions can be scheduled

Work in progress by **various LHC experiments**



HEP AI?

Aka *"send RAW data straight to some HEP AI and forget"*?

Agnostically, check the requirements we need to maintain:

- ability to reformulate the problem (we do not know questions a-priori)
- modularity, i.e. also reusability
- interpretability
- easy validation
- ..

IMO: a omni-comprehensive HEP AI is improbable, but few modular "intelligent" adaptive systems based on advanced ML/DL able to focus on some HEP tasks for everyone (i.e. cross-experiment synergy) is not unthinkable.

One open (key) aspect are e.g. the systematic uncertainties..

Systematic uncertainties

We often do <u>not</u> know a-priori sizes and sources of systematic uncertainty…

- How can a ML algorithm be robust against systematic effects in the training samples, if we do not know how to transfer to it a knowledge we do not have?

Several approaches developed within HEP so far:

- define a physics-specific loss function that explicitly drives the ML optimisation to a solution that is invariant under changes in some (possibly completely unknown) features ***[arXiV:1305.7248] [arXiV:1410.4140]***
- enforce invariance using the adversarial network approach, where the adversary now tries to guess the value of the latent parameter ***[arXiV:1611.01046]***
- parametrise latent parameters such that the NN learns to smoothly interpolate itself as a function of the latent parameters ***[arXiV:1601.07913]***

Summary

The **use of ML** is becoming **ubiquitous in HEP**

- a rapidly evolving approach in HEP to characterising and describing data with the potential to radically change how data is reduced and analysed

Applications domain varies:

- Some will qualitatively (directly) improve the *physics reach of datasets.* Others will allow more efficient use of computing resources, thus (indirectly) extending the *physics reach of experiments*

DL is starting to make a visible impact in HEP

- firstly, with HEP problems that are closely related to those commonly solved using DL

Collaboration with **CS** and synergy with the **world-class ML community** is vital for HEP, and a challenge in itself for both sides!

- HEP has interesting features from a CS perspective (sparse data, irregular detector geometries, heterogeneous information, systematics, ..)
- HEP should be open to other communities, and improve in how to formulate problems in a way CS can understand and be attracted to

Plenty of cutting-edge **very** interesting work and R&D that I did not cover…

- (not exhaustive list!) hardware-side of choices, deployed computing infrastructures for ML in HEP, tracking challenges, jet tagging with RNNs, deep NNs on FPGAs, Deep Kalman Filters, compression using autoencoders, sustainable MEM, and more…