

Probing the nature of the universe with machine learning at the ATLAS experiment

Steven Schramm

CEA Saclay
December 2, 2019



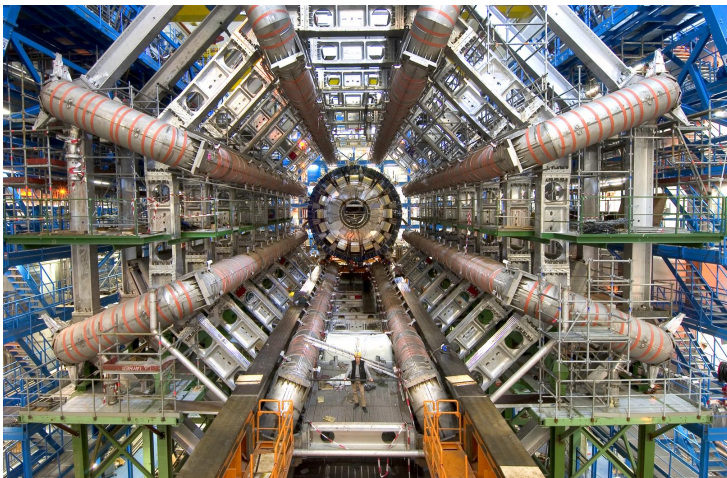
Probing the nature of the universe using machine learning at the ATLAS experiment

- Objective: probe the nature of the universe
 - Search for new physical phenomena or measure known phenomena
- Experimental apparatus: ATLAS experiment (at the LHC)
 - The device that we will use to test hypotheses about the universe
- Methodology: machine learning
 - Machine learning is increasingly used to exploit the enormous dataset
- Let's start with the experiment, as it defines what we can study

What is the ATLAS experiment?

Image: CERN  UNIVERSITÉ DE GENÈVE

- The ATLAS experiment is an enormous particle physics “detector”
 - Essentially a really big, fast, and complicated camera



Where is ATLAS?

Image: CERN  UNIVERSITÉ DE GENÈVE



What is the LHC?

Image: CERN  UNIVERSITÉ DE GENÈVE

- The world's largest and most powerful particle accelerator
 - 27 km of **superconducting magnets**, cooled to 1.9 K ($-271.25\text{ }^{\circ}\text{C}$)
 - Harder vacuum **and** colder than space!
 - Protons are accelerated to 6.5 TeV (99.999999% the speed of light)

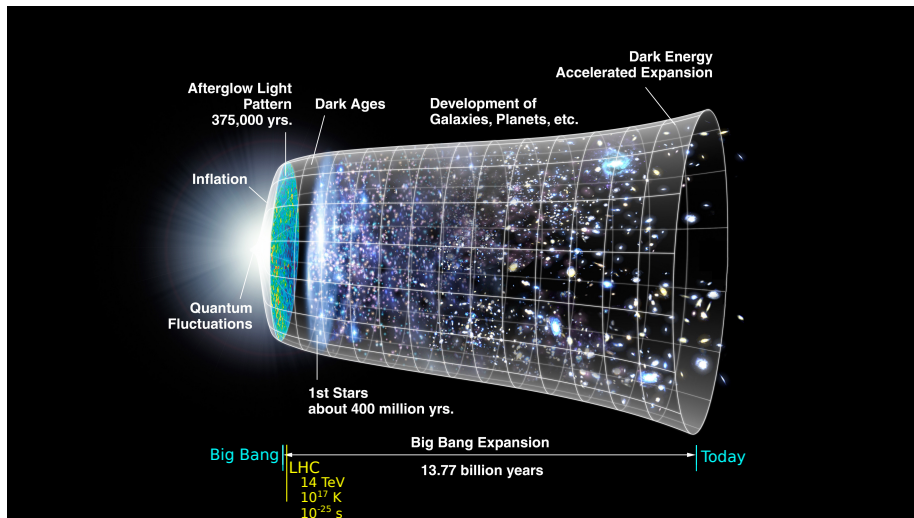


Probing the nature of the universe using machine learning at the ATLAS experiment

- Objective: probe the nature of the universe
 - Search for new physical phenomena or measure known phenomena
- Experimental apparatus: ATLAS experiment at the LHC
 - The device that we will use to test hypotheses about the universe
- Methodology: machine learning
 - Machine learning is increasingly used to exploit the enormous dataset
- Now, why did we build such a huge device? Why is it useful?

Why is the LHC interesting?

Image: NASA  UNIVERSITÉ DE GENÈVE



The LHC is the best way to probe the earliest moments of the universe

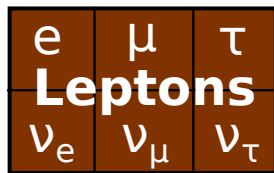
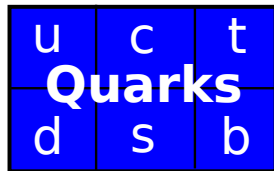
The Standard Model of particle physics

Medal image:

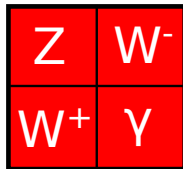
UNIVERSITÉ
DE GENÈVE

Nobel Media

Matter



Forces (Bosons)



Scalar



Discovered 2012
(ATLAS+CMS)



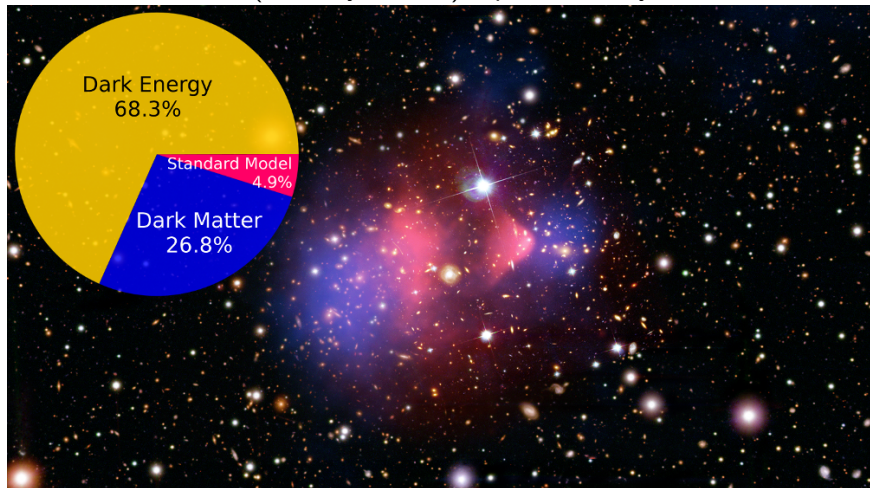
Nobel prize 2013

The Standard Model is complete, but does not explain most of the universe

Dark matter

Image: NASA  UNIVERSITÉ
DE GENÈVE

The Standard Model (ordinary matter) represents only 5% of the universe

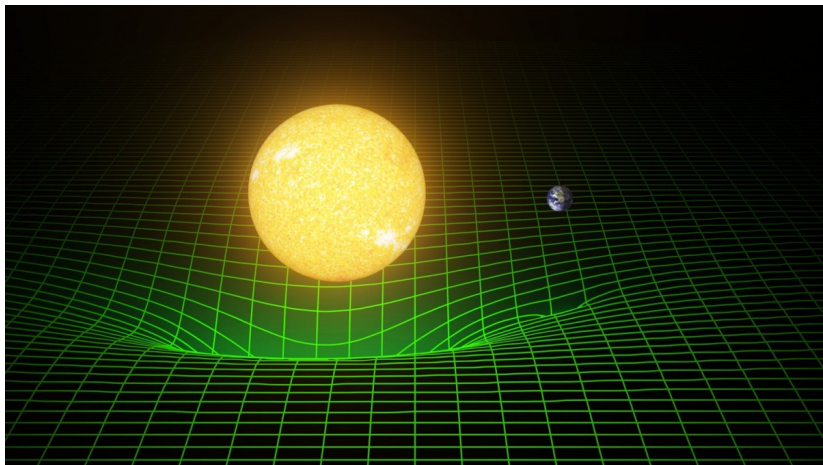


The LHC is sensitive to many well-motivated Dark Matter candidates

Gravity

Image: T. Pyle/Caltech/MIT/LIGO Lab

The Standard Model doesn't explain the first force humanity "discovered"



LHC is sensitive to some models of gravitons, with implications on gravity

Naturalness and the hierarchy problem

Why is gravity so weak compared to EM force / why is the Higgs so light?

$$\text{Higgs Boson}^2 \quad (125 \text{ GeV})^2 = \text{Raw value} \quad 10^{36} \text{ GeV}^2 - \text{Quantum corrections} \quad 10^{36} \text{ GeV}^2$$

$$\begin{array}{r} 123456789012345678901234567890123456 \\ - \\ 123456789012345678901234567890107831 \\ = \\ 15625 = (125)^2 \end{array}$$

The LHC is sensitive to several possible explanations

Machine learning at the LHC

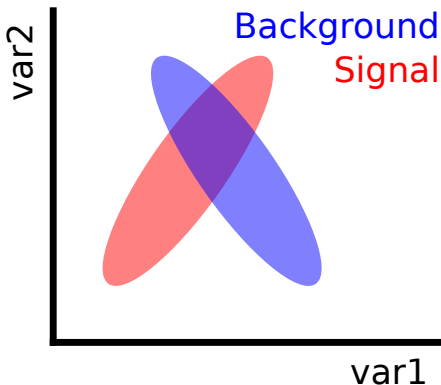
- The LHC collaborations have **a lot** of data
- Machine learning techniques are used in \sim every part of the process
 - Building physics objects from detector electrical signals (reconstruction)
 - Correcting physics objects for mis-measurements (calibration)
 - Quickly filtering the enormous amount of input data (triggering)
 - Determining the origin of a physics object (identification)
 - Searching for unexpected detector signals (generic searches)
 - Improving the final data analysis sensitivity (multivariate analysis)
 - Validation of the quality of the recorded data (monitoring)
 - Quickly generating large-scale simulated datasets (fast simulation)
- Machine Learning (ML) terminology: clustering + pattern matching, regression, classification, anomaly detection, and generative models

Brief introduction to ML (from a physicist)

- ML is a powerful technique to extract information from a dataset
- ML has both positives and negatives
 - Classification improvements using ML can be quite substantial
 - However, less understanding of what is being done
 - Need to ensure the ML is learning real features, not simulation artifacts
 - Need to have a way to quantify data vs simulation differences
- There are many, many types of ML techniques
 - Next slides are a brief intro to two common ones: BDTs and NNs

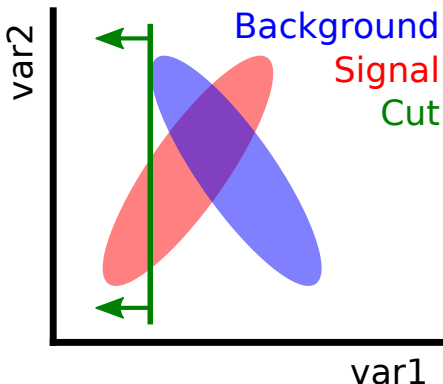
Classification example

- Typical HEP use case:
separate **signal** vs **background**
- Two discriminating variables
- What can we do?



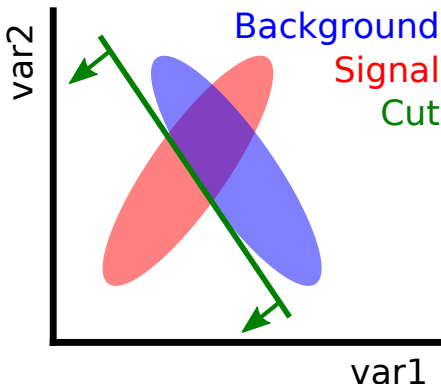
Classification example

- Typical HEP use case:
separate **signal** vs **background**
- Two discriminating variables
- What can we do?
 1. Independent variable **cut(s)**



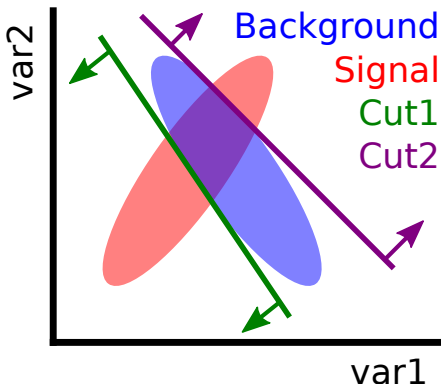
Classification example

- Typical HEP use case:
separate **signal** vs **background**
- Two discriminating variables
- What can we do?
 1. Independent variable **cut(s)**
 2. **Cut** on var1 and var2 simultaneously



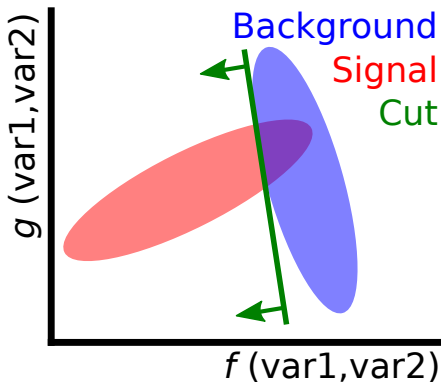
Classification example

- Typical HEP use case:
separate **signal** vs **background**
- Two discriminating variables
- What can we do?
 1. Independent variable **cut(s)**
 2. **Cut** on var1 and var2 simultaneously
 3. Partition the parameter space and simultaneously cut **multiple times**



Classification example

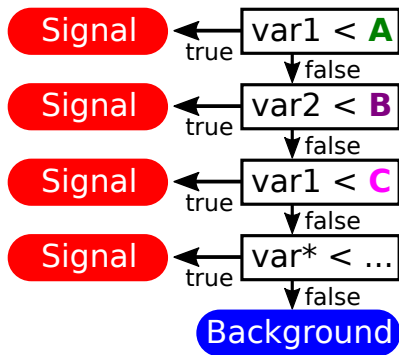
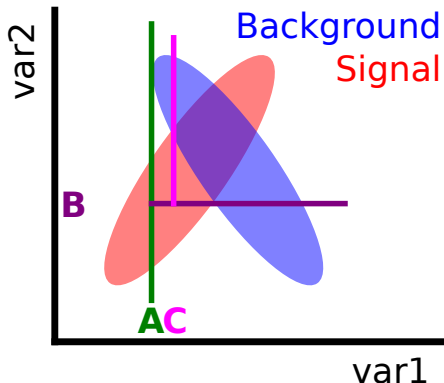
- Typical HEP use case:
separate **signal** vs **background**
- Two discriminating variables
- What can we do?
 1. Independent variable $cut(s)$
 2. **Cut** on var1 and var2 simultaneously
 3. Partition the parameter space and simultaneously cut **multiple times**
 4. Calculate new properties $f()$ and $g()$; **cut** simultaneously on them instead



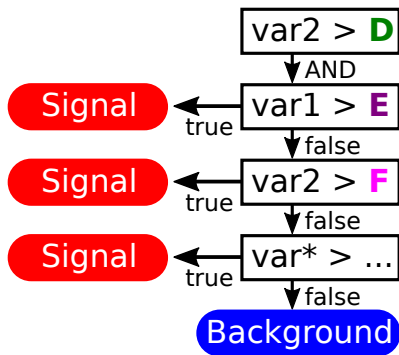
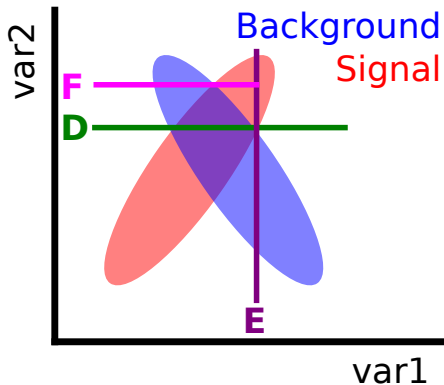
Classification example

- Typical HEP use case:
separate **signal** vs **background**
- Two discriminating variables
- What can we do?
 1. Independent variable **cut(s)**
 2. **Cut** on var1 and var2
simultaneously
 3. Partition the parameter
space and simultaneously
cut **multiple times**
 4. Calculate new properties $f()$
and $g()$; **cut** simultaneously
on them instead
- Rough conceptual analogy...
 - #3 \sim Boosted Decision Trees
 - #4 \sim Neural Networks

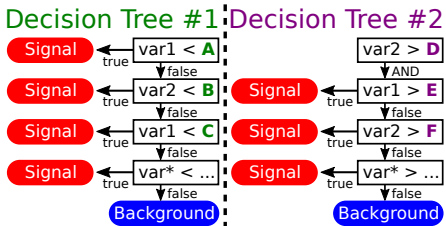
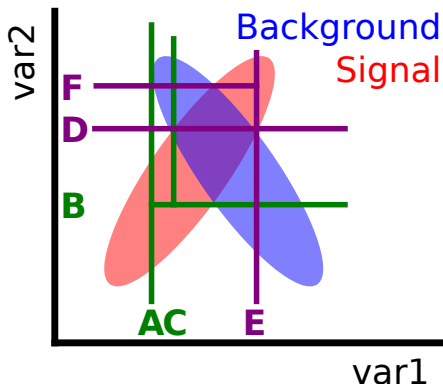
Classification example, BDT perspective



Classification example, BDT perspective



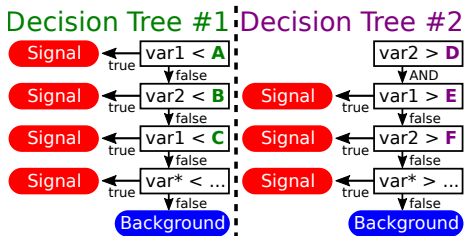
Classification example, BDT perspective



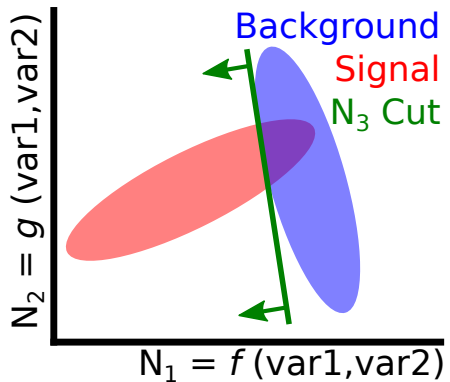
- In this simple example, the two decision trees can be combined into a single tree
- In reality, not easy to cleanly separate signal and background

Classification example, BDT perspective

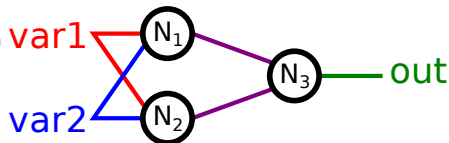
- In reality, partitions are not perfect
 - The background is non-zero in the signal region
 - Having a reasonable amount of signal \implies keeping some background
- Each tree has a misclassification rate
 - Define the final discriminant as a combination of individual trees, weighted by their respective misclassification rates
- Discriminant = $c_1 \cdot \text{DT1} + c_2 \cdot \text{DT2} + \dots$



Classification example, NN perspective



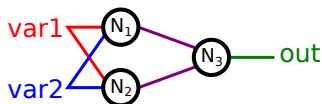
- Neural networks: more complex
 - Cut on combinations of input variables
- In analogy to the example:
 - Nodes N_1 and N_2 are the functions $f()$ and $g()$
 - Cut on N_3 , which is the convolution of $f()$ and $g()$



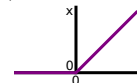
Classification example, NN perspective

How this works (in very brief):

- Nodes 1 and 2:
 - Inputs: {var1, var2}, {var1, var2}
 - Parameters: $\{c_1, c_2, b_1\}$, $\{d_1, d_2, b_2\}$
 - Activation: ReLU, for non-linearity
 - $N_1 = \max(0, c_1 \cdot \text{var1} + c_2 \cdot \text{var2} + b_1)$
 - $N_2 = \max(0, d_1 \cdot \text{var1} + d_2 \cdot \text{var2} + b_2)$
- Node 3 (the final discriminant):
 - Inputs: N_1 and N_2
 - Parameters: a_1, a_2, b_3
 - Activation: Sigmoid, for a probability
 - $N_3 = 1 / \left[1 + e^{-(a_1 \cdot N_1 + a_2 \cdot N_2 + b_3)} \right]$



ReLU
(Rectified Linear Unit)



$$f(x) = \max(0, x)$$

Sigmoid

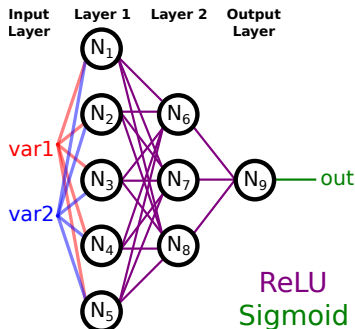


$$f(x) = \frac{1}{1 + e^{-x}}$$

We will look at this in
more detail later

Classification example, NN perspective

- The last slide was a bit simplistic
 - This is slightly more realistic
- Layers 1 and 2 are “hidden layers”
 - Hidden = neither inputs nor outputs
 - If there are at least two hidden layers, then the network is “deep”; a DNN
- Output layer could have multiple nodes
 - One output = binary discriminant (signal vs background)
 - 2+ outputs = multi-class discriminant (signal vs BG1 vs BG2 or similar)
- An *infinitely deep* network can represent *any* non-linear function
 - That said, in many cases we aren't so close to infinity



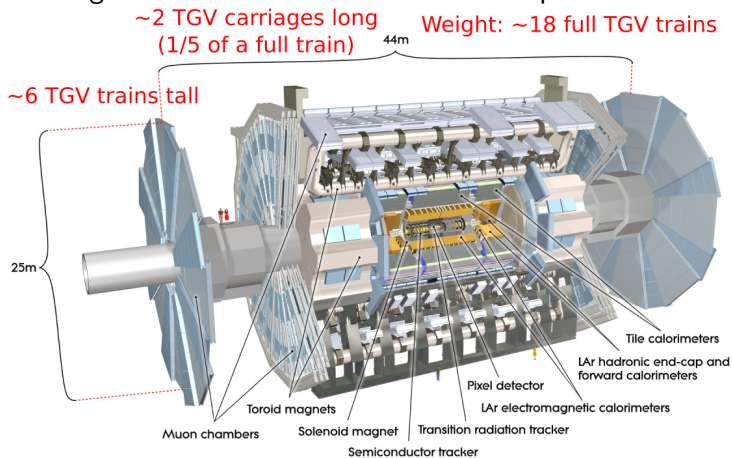
Machine learning in ATLAS

- That is a very brief introduction to BDTs and NNs
 - BDTs define a series of partitions and weight different such series
 - NNs form non-linear combinations and convolutions of variables
- BDTs and NNs are not the only types of ML!
 - They are, however, the dominant use type in HEP
- Now, how does ATLAS make use of these ML tools?
 - To answer that, we need to understand what the data looks like

The ATLAS detector

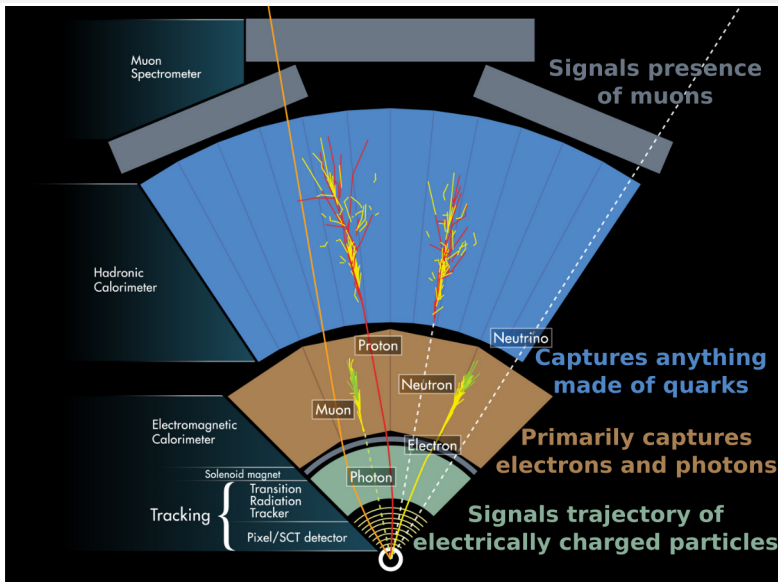
Image: CERN  UNIVERSITÉ DE GENÈVE

- Large detectors are built around each interaction point
 - The ATLAS experiment is the largest detector (by size)
- Combining the different sub-detectors enables particle identification



Particle identification in ATLAS

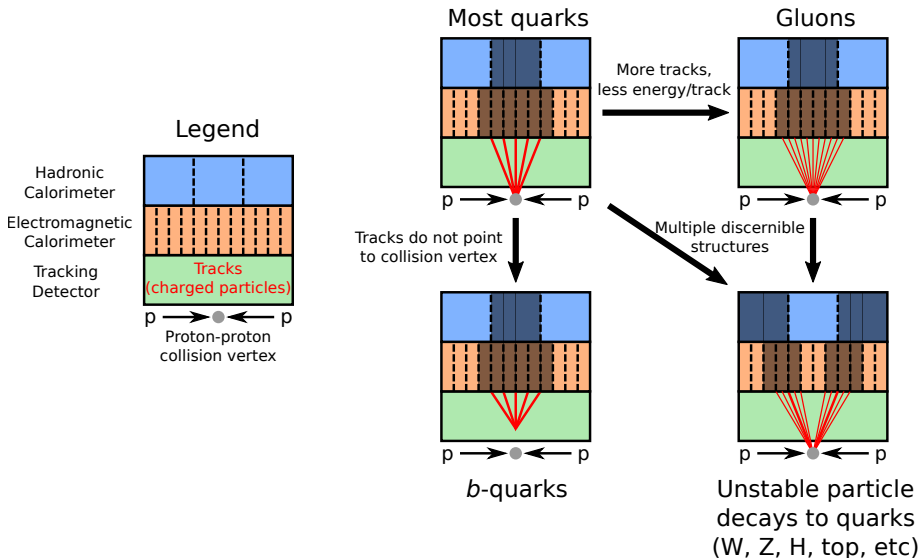
Image: CERN  UNIVERSITÉ DE GENÈVE



Telling apart different particles involving hadrons

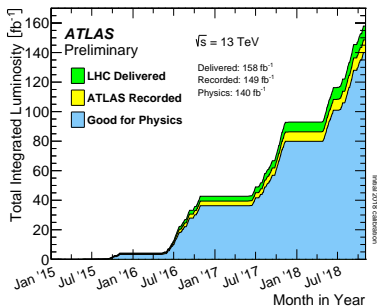
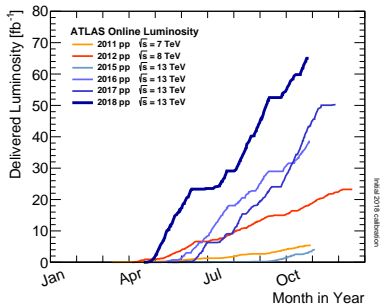
- The last slide shows how to identify particles **except** quarks
 - However, particles involving quarks are much more complex
 - Quarks cannot exist in isolation: they immediately “shower”
 - We call this collimated stream of particles a “jet”
- There are many different quarks and related particles
 - How can we tell them apart?
 - Requires a much more complex approach, and the whole detector!
 - Arguably the most active area of development for ML in particle physics

Differentiating between jets, simplified



How much data do we have?

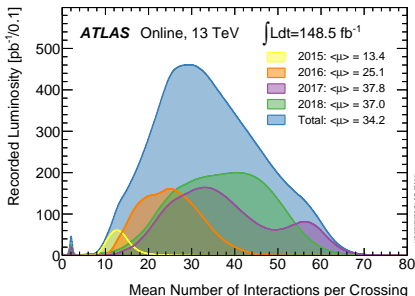
- Particle physics uses an unusual unit: the inverse femtobarn, fb^{-1}
 - One inverse femtobarn ≈ 100 trillion proton-proton collisions
- ATLAS has recorded 140 fb^{-1} of good data from 2015–2018
 - Therefore, $140 \text{ fb}^{-1} \approx 14$ quadrillion = 14×10^{15} collisions
- That is a **big dataset**



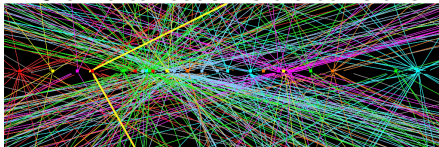
Pileup

Left: ATLAS  UNIVERSITÉ DE GENÈVE
 Right: ATLAS  UNIVERSITÉ DE GENÈVE

- The last slides are a very simplified view of what happens
- In reality, averaged 34 collisions per crossing in Run 2 (2015–2018)
 - In-time pileup: other collisions that happened at the same time
 - Out-of-time pileup: overlapping collisions due to detector read-out time
 - We only want to look at **one collision**, so we need to separate pileup
- Dealing well with pileup is the key to huge datasets



25 simultaneous visible collisions



The tracking detector can resolve individual vertices

Reconstruction and identification

The multiple stages of reconstruction

- Reconstruction: turning detector signals into physics objects
- Four main stages of reconstruction:
 1. Convert analog electrical signals to digital signals to energy values
 2. Group (cluster) regions of energy values into a single object
 3. **Calibrate** the resulting object
 4. **Identify** the origin of that object
- The third and fourth steps might be inverted, or might be iterative
- These are also two steps where ML is very promising
 - **Regression** is the ML name for a calibration-like algorithm
 - **Classification** if the ML name for an identification algorithm

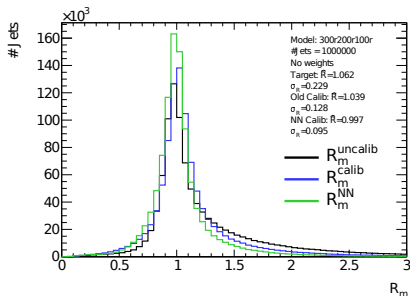
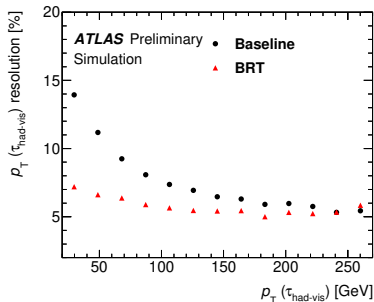
Object calibration (regressions)

Left: CONF-2017-029

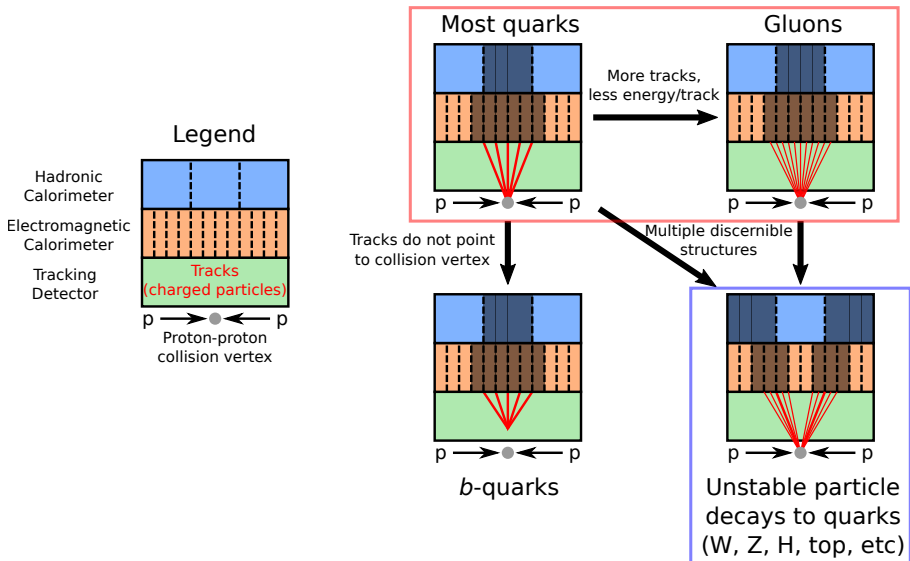
Right: Ennis (2017)

UNIVERSITÉ
DE GENÈVE

- ATLAS is increasingly making use of regressions
 - Boosted Regression Tree (BRT) significantly improves p_T^{τ} resolution
 - Resolution = measurement precision, want it to be 0 (perfect)
- NNs are also used for regressions, here for the jet mass
 - Comparing a normal calibration vs neural network regression
 - NN calibration is a sharper peak (better resolution) \implies many benefits



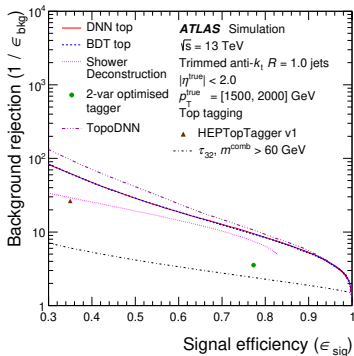
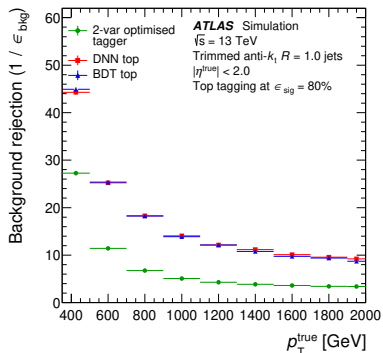
Jets, revisited: a second stage classification



Hadronic decay tagging

Both: JETM-2018-03

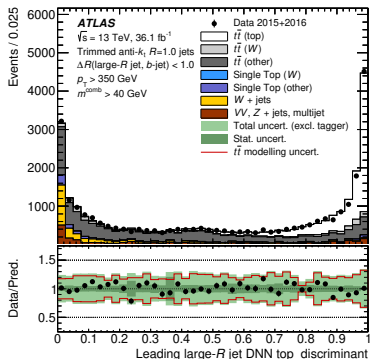
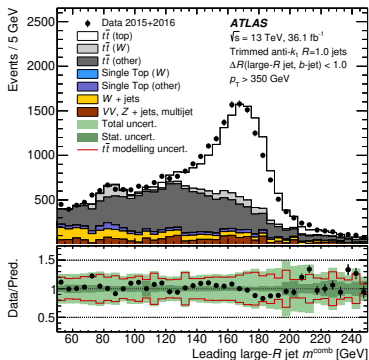
- Goal: identify hadronic decays (signal) vs quarks/gluons (background)
 - Even two-variable simple selections give impressive results
- Combined $\mathcal{O}(10)$ variables into a BDT or DNN
 - Huge improvements, 2-3 \times increased background rejection
- However, BDTs and DNNs are still performing similarly
 - They both use “high-level” variables (properties of jets)
 - TopoDNN adds low-level info (properties of inputs to jets)



Comparing data and simulation

Both: JETM-2018-03

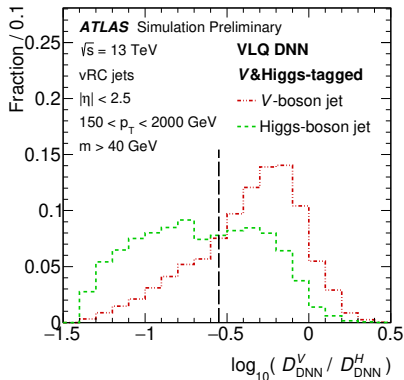
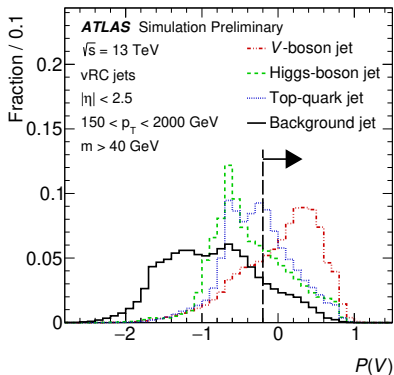
- Next, define high-purity “control regions”
 - Allows for comparing data and simulation
 - This can be used to control and cross-check our ML tools
- DNN discriminant agrees well between data and simulation
 - They are identical within the physical uncertainties



Multi-class tagging

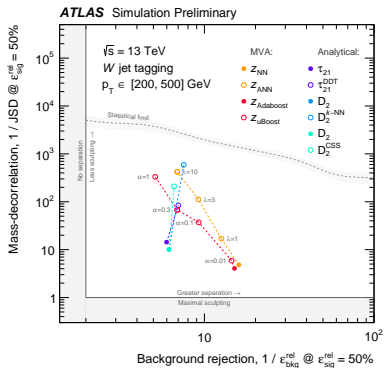
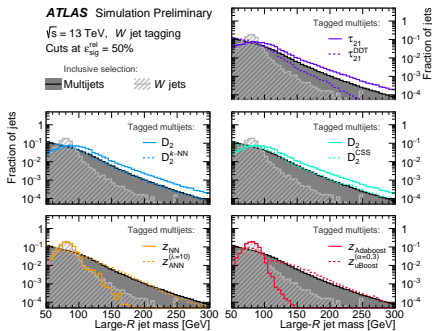
Both: EXOT-2017-14

- Sometimes there are multiple similar objects to differentiate
 - Different hadronic decays can appear similar: **W/Z** vs **H** vs **top** vs QCD
- Multi-class DNN trained on a mix of low-level and high-level variables
 - First stage: discriminate against non-decaying (QCD) jet background
 - Second step: use discriminant likelihoods for signal ambiguity resolution



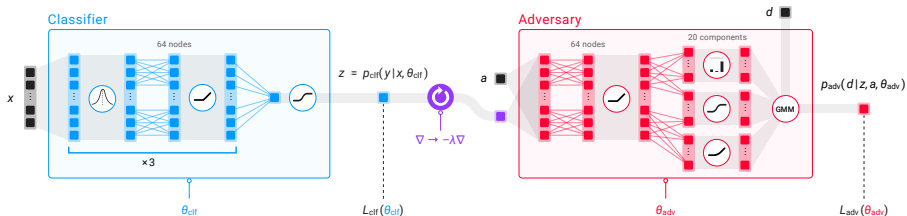
Decorrelated taggers

- ML exploits correlations, but this may not be desired
 - When measuring a variable, you don't want correlations to change it
- Studied a variety of decorrelation techniques
 - Two ML-based: **uniform boosted BDTs** and **adversarial neural networks**
 - Adversarial training** parametrizes the trade off: shaping vs classification

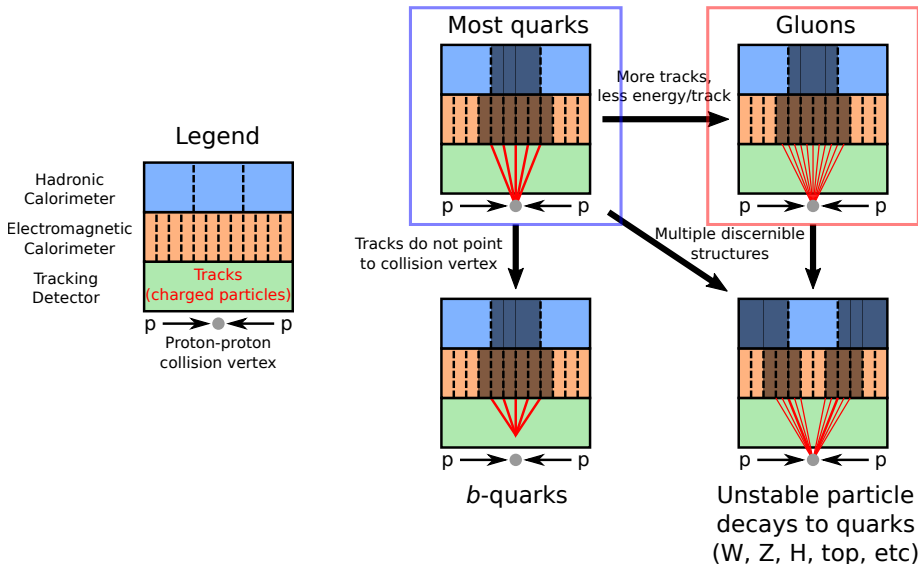


Adversarial neural networks

- Idea: put two networks competing against each other
 - First network: perform the classification, signal vs background
 - Second network: insist that the shape remains unchanged
 - Change the loss function to make the networks compete
 - $L_{\text{tot}} = L_{\text{classifier}} - \lambda \cdot L_{\text{adversary}}$, where λ is a hyperparameter
- Result: Adversarial Neural Network (ANN, \neq Artificial NN)
- This is generic: can be applied to any variable(s) of interest
 - However, adversarial networks can be **very hard to train**

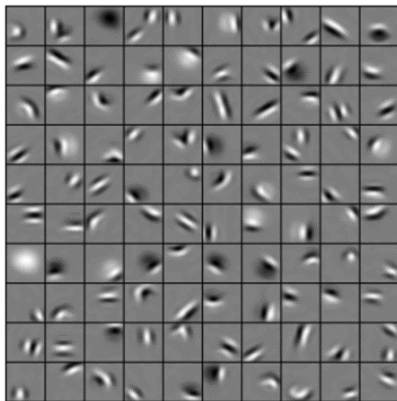


Jet identification: quark/gluon tagging



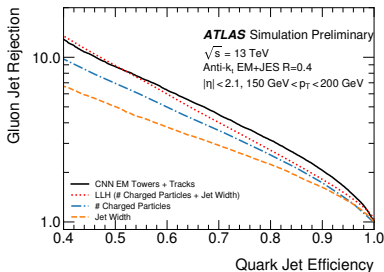
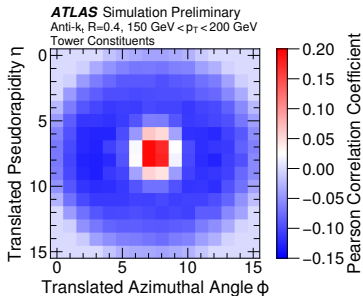
Convolutional neural networks (CNNs)

- CNNs are a dominant form of image recognition
- Convolutions are essentially mini-patterns; CNNs have several levels
 - Large patterns: large parts of faces
 - Medium patterns: individual parts of faces
 - Small patterns: edges and gradients (structural transitions) of faces

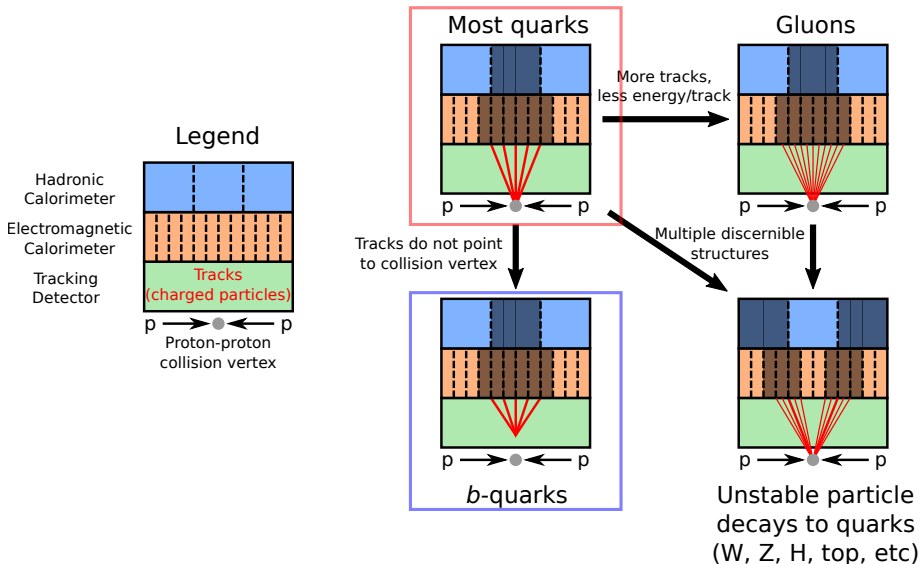


CNNs and jet images

- “Jet images” and CNNs have been tested for quark/gluon tagging
 - Convolutional NN uses tracks and calorimeter towers as inputs
 - Different inputs are analogous to different “colours” in RGB pictures
- Moderate gain over **two-variable likelihood** for high quark efficiency
 - However, ultimately little gain beyond using the number of tracks
 - Not enough information for the CNN to exploit in this context

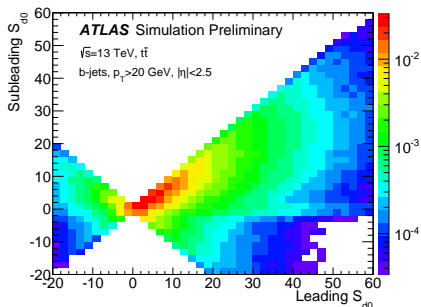
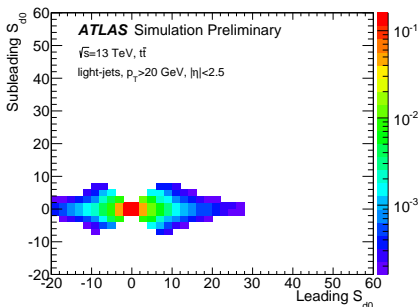


Jet identification: *b*-tagging



Ordering in b-tagging

- When there is an ordering expectation, it should be taken into account
- Recall that b-jets are primarily identified by displaced vertices
 - Sort all tracks by their displacement significance
 - Non-b-jet: random, uncorrelated track displacements, no real structure
 - b-jet: correlated displacements, multiple tracks from same b-vertex
- Cuts on displacement work, but neglect the track-track correlations



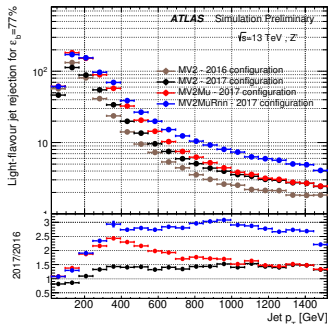
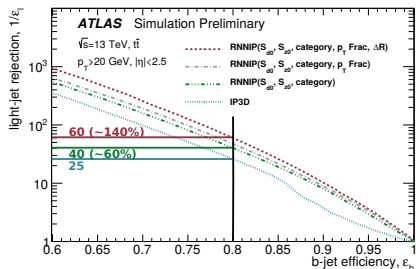
RNNs and b -tagging

Left: PUB-2017-003

UNIVERSITÉ
DE GENÈVE

Right: PUB-2017-013

- Use of tracks as input to RNN for b -tagging
 - Recurrent NN considers up to 15 tracks for each jet (for training speed)
 - $\sim 60\%$ gain using **same variables** vs likelihood, $\sim 140\%$ with **extra vars**
- RNN output is an input to the higher-level b -tagging BDT
 - Joint improvements from **physics** and **ML** developments
 - Important to push forward in both directions for the best possible result



ML in analysis

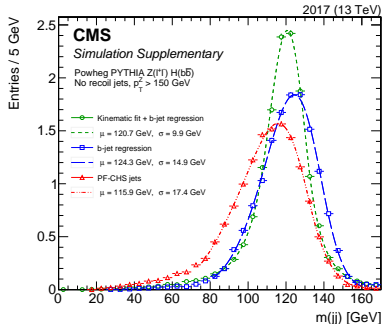
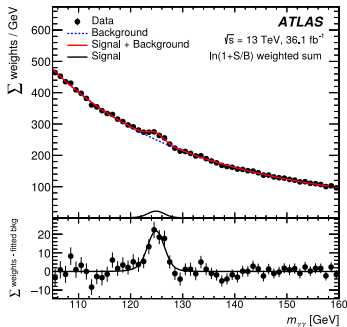
Preparing for a physics analysis

Left: HIGG-2016-33


Right: CMS

UNIVERSITÉ
DE GENÈVE

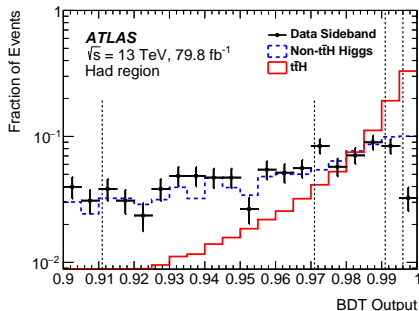
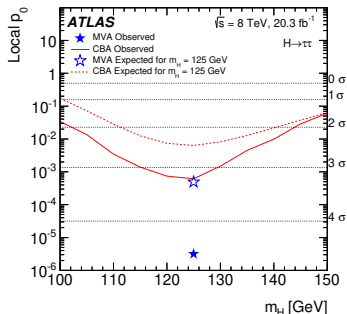
- So far we have discussed building, calibrating, and identifying objects
- These ML improvements propagate to searches and measurements
- Canonical example: measuring Higgs boson properties
 - Clear “spike” in the data for the two-photon or two-b-jet mass
 - This is the easiest way to show the existence of a new particle
 - ML can be used to increase the prominence of this population of events



ML in the final results

Left: HIGG-2013-32  UNIVERSITÉ DE GENÈVE
 Right: HIGG-2018-13  UNIVERSITÉ DE GENÈVE

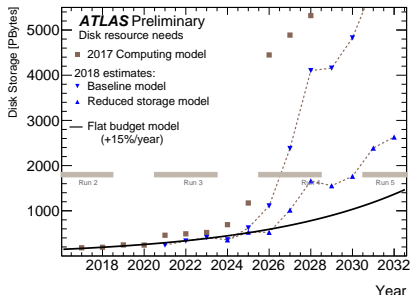
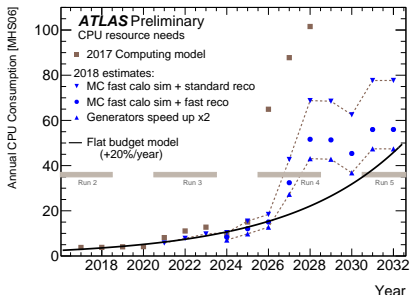
- BDTs and NNs are used throughout many high-profile ATLAS results
- BDTs played a key role in the first evidence for $H \rightarrow \tau\tau$
 - Note that the τ particle is identified using a BDT (another type of jet)
 - At the analysis level, another BDT improves compared to cut-based
- BDTs are also key to last year's $t\bar{t}H$ observation
 - Two separate BDTs are used, with multiple channels and categories



Looking toward the future

Looking toward the future

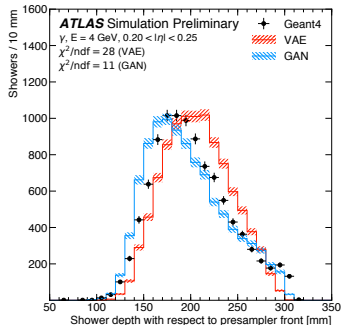
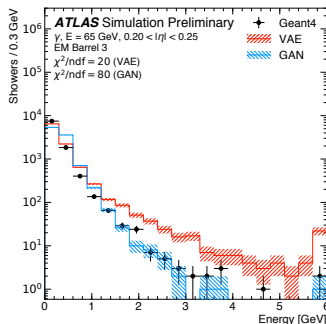
- Currently the LHC delivers $\sim 8.5 \times 10^9$ collisions per second
 - This is set to increase to $\sim 50 \times 10^9$ collisions per second
 - There will also be much more information in every event
- Below is the foreseen associated computing resource costs
 - Something needs to be done to reduce computing requirements
 - Storage is hard, but CPU can be addressed more directly



Simulation

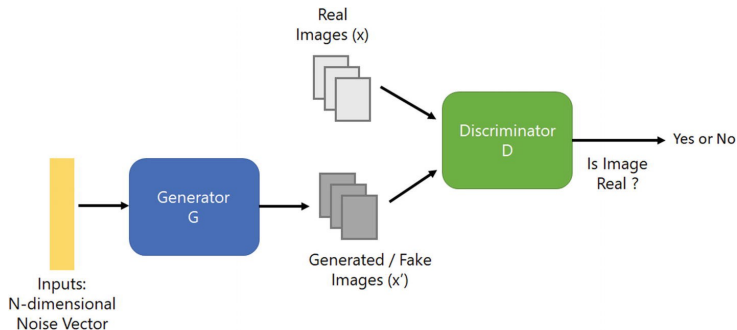
Both: PUB-2018-001

- Currently, roughly half of all CPU is dedicated to simulation
 - This is going to continue, we need more simulation than data
- The problem is the fully detailed ATLAS simulation is **slow**
 - Enormous number of interactions to propagate through the detector
- Two main types of generative models under study
 - **Generative Adversarial Networks** and **Variational AutoEncoders**



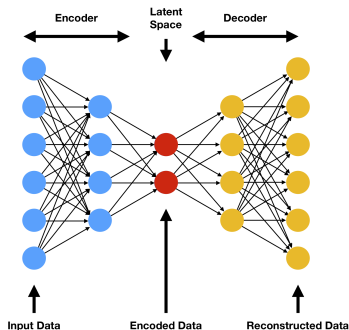
Generative Adversarial Networks (GANs)

- Similar to our previous discussion on adversarial networks
- Puts a generator against a discriminator
 - Generator: produce images from random noise inputs
 - Discriminator: tell “fake” images apart from real ones
- If training succeeds, the discriminator can no longer tell what is real
 - At this point, the generator can be used to produce realistic data



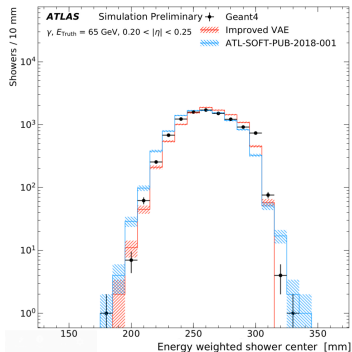
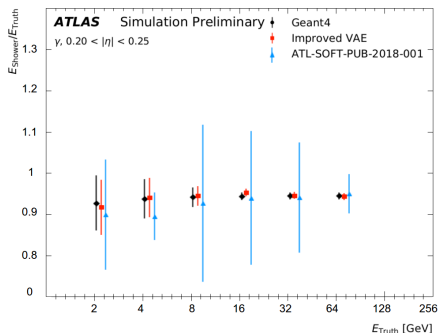
Variational AutoEncoders (VAEs)

- VAEs compress information to a “latent space”, then reconstruct
 - Essentially learning an identity matrix with a small dimensionality
 - Training: minimize difference between inputs and outputs
- Once trained, the decoder has learned how to create input-like images
 - Cut the network, starting from latent space, and use noise as inputs
 - Will then produce realistic images similar to the training inputs



What is the current status?

- Significant update to the VAE approach this month
- Making enormous improvements, but still room for improvement
 - Main focus so far is on electrons, later extend to hadrons
- Speeds not listed, but typically orders of magnitude faster with VAE/GAN than full simulation when running with GPUs

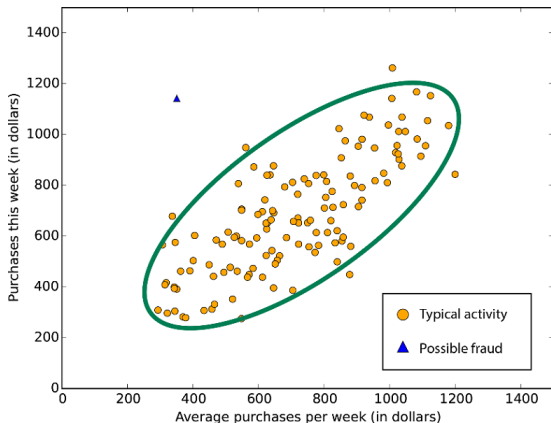


Anomaly detection

- Another use of machine learning is to detect unusual situations
 - Train ML to understand the expectation
 - Look for deviations from that expectation
- Two main use cases:
 1. Unexpected features in a time series
 2. Statistically significant outliers in N -dimensional space
- Both of these are common in industry applications
- They are increasingly under study also in HEP
 - In particular, the topic of statistical outliers is a major focus right now

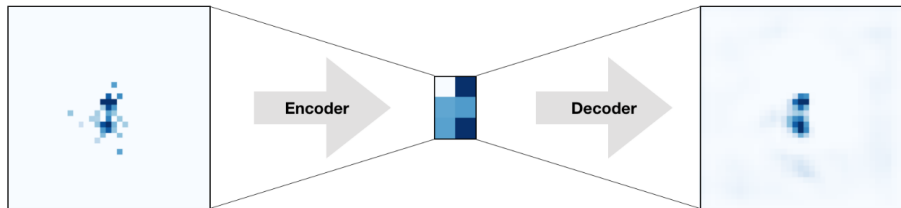
Anomaly detection: statistical outliers

- Anomaly detection is used a lot in monitoring for fraud
 - Most likely your credit card purchases are being watched this way
 - Many parameters to such methods, looking for unusual behaviour



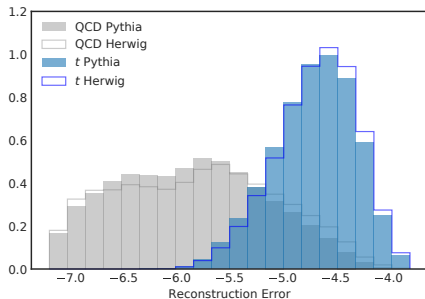
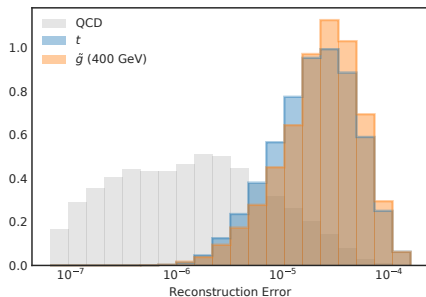
Anomaly detection in HEP

- Lack of discovery of new physics so far prompts questions
 - Are we looking in the wrong place for new physics?
- Increasing interest in more generic searches for new particles
- Example of jet identification: consider an autoencoder
 - Train the autoencoder on normal Standard Model jets
 - The ML is then able to reconstruct normal jets, but not abnormal jets
 - Look for jets where the autoencoder output is far from the input



Anomalous jet identification

- Autoencoder was trained on QCD jets (light quarks)
 - QCD jets are thus reconstructed similarly (low reconstruction error)
 - Both **hadronic top quark decays** and **gluinos** show up as anomalous jets
 - Stable also vs MC type, so it's not learning MC features
- This principle can be expanded
 - Train on all known jet types and look for anomalies
 - Iteratively add detector features/etc to the training as encountered
- May eventually encounter an unexpected type of new physics



Summary

A few last thoughts

- ML can be a powerful tool, but it is not magic
 - Inputs matter a lot: better inputs = better performance
 - Having sufficient statistics is important: it needs to be able to “learn”
 - You need to pre-process training samples to avoid undesired features
- Don't leave everything to the machine
 - With infinite statistics and complexity, machines *could* learn “anything”
 - We do not live in such a world, so we need to help the machine
 - ML + domain knowledge is a winning combination
- ML algorithms do not necessarily mean increased MC dependence
 - It is not uncommon for ML to reduce MC modelling uncertainties
 - You can even use adversaries to not learn data/MC or MC/MC diffs
 - That said, you should still check this, as bad things can happen
- Recall that ML is an approximation algorithm
 - If you can calculate the exact likelihood or analytic solution, do it!
ML can at best converge on this exact solution
 - ML is most useful in situations where exact solutions are intractible
 - Most commonly situations with many useful (distinct) dimensions

Summary

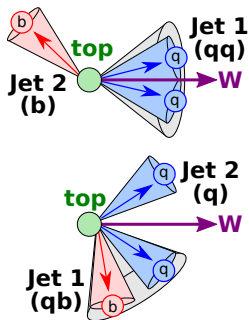
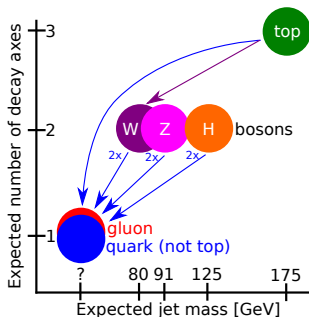
- ATLAS has a **big dataset**, and this is only going to grow
- Machine learning is increasingly used to exploit this data
 - Used in all corners of the experiment
 - Triggering, reconstruction, analysis, simulation, and much more
- ATLAS is using a variety of three-letter ML acronyms to do this
 - BDTs, DNNs, CNNs, RNNs, ANNs, BRTs, GANs, VAEs, ...
 - There are also many other approaches under study but not yet public; the field of ML is evolving at an extremely fast rate
- HEP benefits a lot from watching external community developments
 - At the same time, we sometimes have our own contributions as we care about some topics more than most external people

- In the coming years, ATLAS dataset will approach the exabyte scale
 - It's important to start thinking now about how we can benefit
 - Machine learning usage is only going to increase!

Backup Material

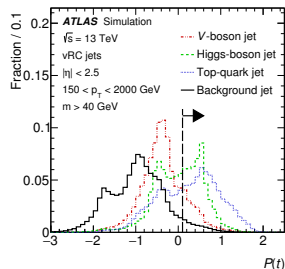
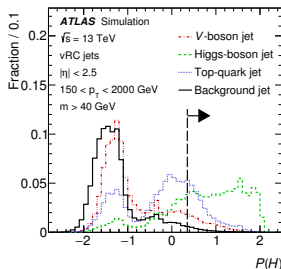
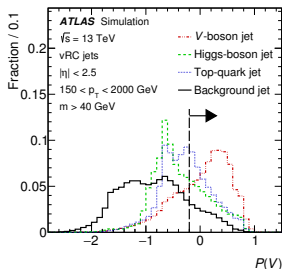
Overlapping signal definitions

- It is hard to unambiguously define a **top quark**
 - Natural confusion with **W boson** as it's part of the decay
- The W is also easy to confuse with **Z** and **H** bosons
 - **W** and **Z** are \sim identical except for small mass shift
 - **Higgs** is a bit more different, but still similar substructure
- What if our analysis involves **top**, **W**, **Z**, and **Higgs**?
 - Need a tagger that can tell them apart from each other AND **quarks**



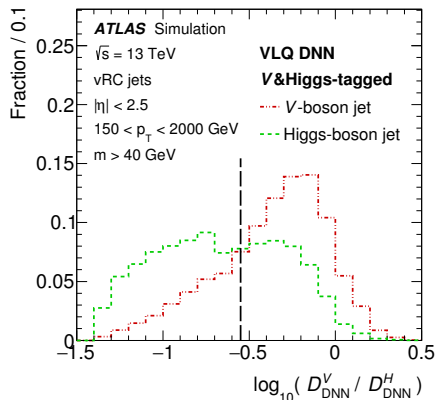
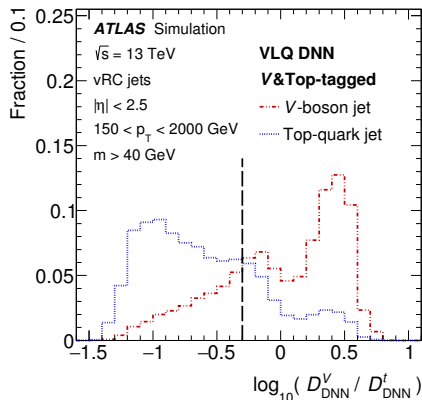
Multi-class jet identification

- We would have five labels, but we will combine W and Z into **V**
- Design a network with four outputs: **V**, **H**, **top**, and **quarks/gluons**
 - Last layer uses sigmoid activation \implies each output is between 0 and 1
- Define “probabilities” (not between 0 and 1) as:
 - $P(V) = \log_{10} \left(\frac{D_V}{0.9 \cdot D_{\text{QCD}} + 0.05 \cdot (D_H + D_{\text{top}})} \right)$
 - Same for H and top, just switch labels as appropriate
- However, these are not exclusive: a jet can be multi-tagged



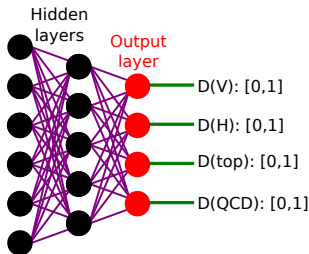
Second-stage identification

- In case a jet is tagged with multiple origins, how can we resolve this?
 - Simple approach: log ratios of single-tag discriminants
- End result is unique identification as **V**, **H**, **top**, or **QCD**



An alternative: SoftMax

- This approach works, but it is complex and has three stages
 - Raw discriminant $D(X)$, probability $P(X)$, and multi-tag decision $T(X)$
- SoftMax is an alternative output layer choice
 - All values are in the range $[0,1]$ and are normalized such that $\sum X_i = 1$
 - Outputs are now probabilities, use most probable label as final decision



$$P(V) = D(V) / [D(QCD) + D(H) + D(top)]$$

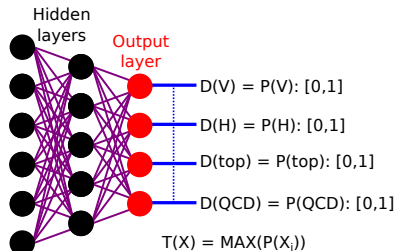
$$P(H) = D(H) / [D(QCD) + D(V) + D(top)]$$

$$P(top) = D(top) / [D(QCD) + D(V) + D(H)]$$

$$T(V) = P(V), \log[D(V)/D(H)], \log[D(V)/D(top)]$$

$$T(H) = P(H), \log[D(H)/D(V)], \log[D(H)/D(top)]$$

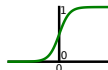
$$T(top) = P(top), \log[D(top)/D(V)], \log[D(top)/D(H)]$$



ReLU
(Rectified Linear Unit)



Sigmoid



SoftMax



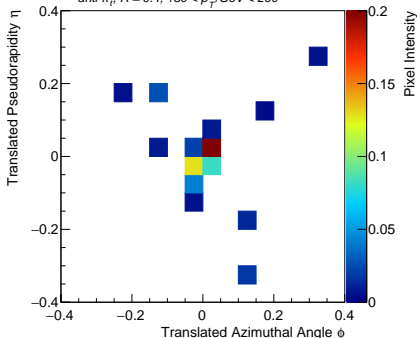
A closer look at jet images for q/g tagging

- Individual jet images are very sparse
- This is challenging for CNNs to handle
 - Convolutions are looking for consistent features in a given grid size

ATLAS Simulation Preliminary

Glueon Jets, Track Constituents

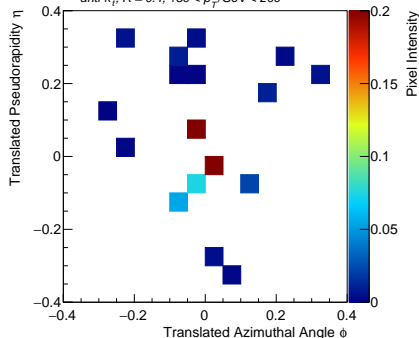
anti- k_r , $R = 0.4$, $150 < p_T/\text{GeV} < 200$



ATLAS Simulation Preliminary

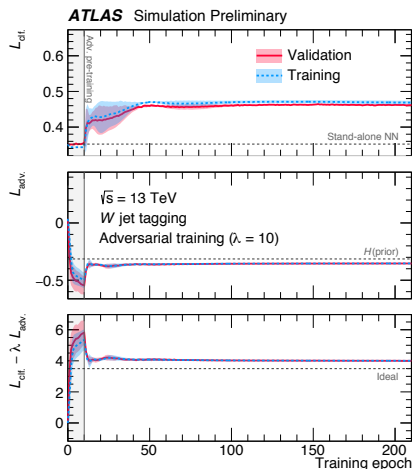
Glueon Jets, Topocluster Constituents

anti- k_r , $R = 0.4$, $150 < p_T/\text{GeV} < 200$



Training adversarial networks

- Training proceeds in stages
 1. Pre-train the classifier
 2. Pre-train the adversary on the trained classifier
 3. Simultaneously train the classifier and adversary
- Necessary for convergence
 - Need to start from a reasonable classifier
 - Adversary then learns the desired pre-tag distribution and how the classifier differs
 - Then the two are allowed to evolve together to converge



Expanding to unsupervised learning

- So far, we have always been talking about *supervised learning*
 - When training, you *know* the correct result (truth information)
- Anomaly detection is a place where you may want to stop doing this
 - Train **directly on the data** to avoid MC limitations
 - The data will have features not in MC (detector structure, etc)
- In this case, the training has no truth information
 - This is known as *unsupervised learning*
- Multiple ways to consider doing this
 1. Define control and signal regions in data, train in control regions
 2. Rely on any anomalies being rare enough to not impact training
- The first option is the normal HEP approach, the second is unique

Contamination in anomaly detection (arXiv:1808.08992)

- When training the autoencoder, it will focus on the bulk of the inputs
 - Rare inputs will be neglected as they have \sim no weight
- As such, you can train and use anomaly detection on the same data
 - Only works for truly anomalous signals (low contamination)
- Still remarkably powerful: 10% signal contamination in training leads to 10% signal efficiency at 1% background efficiency

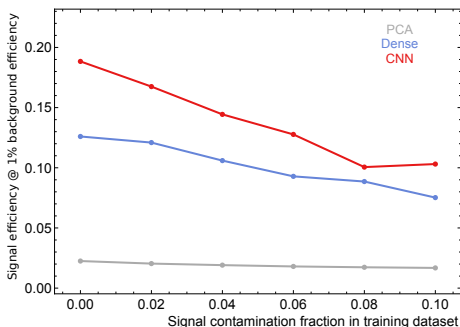


Image denoising for pileup mitigation (PUB-2019-028)

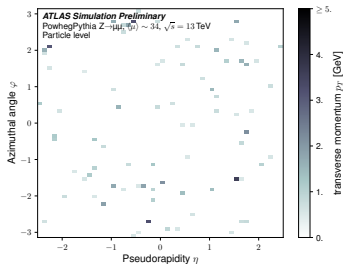
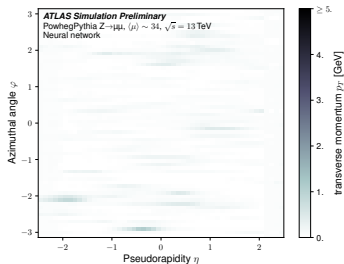
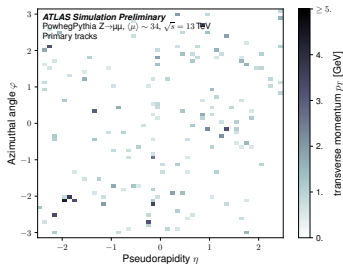
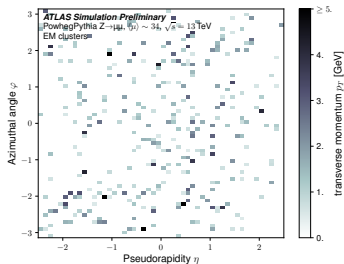


Image denoising for pileup mitigation (PUB-2019-028)

- Missing transverse momentum, E_T^{miss} , is a full event balance
- Does a good job suppressing resolution pileup dependence
 - However, has some moderate implications on the scale
- Trained to optimize resolution, not scale
 - An area that may improve later with a different optimization

