Unsupervised Anomaly Detection in Particle Physics

Greg Landsberg

Brown AI Winter Workshop

January 17, 2024

BROWN

Outline

- General Search Methods
-

BROWN

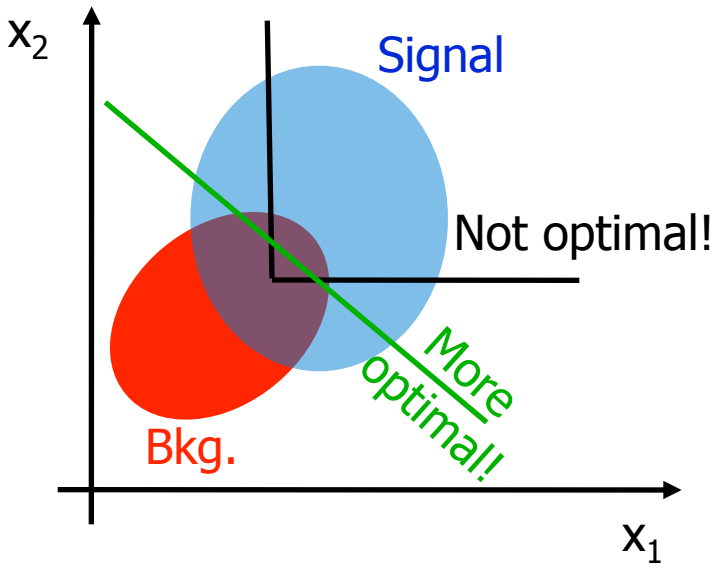
Search Methods

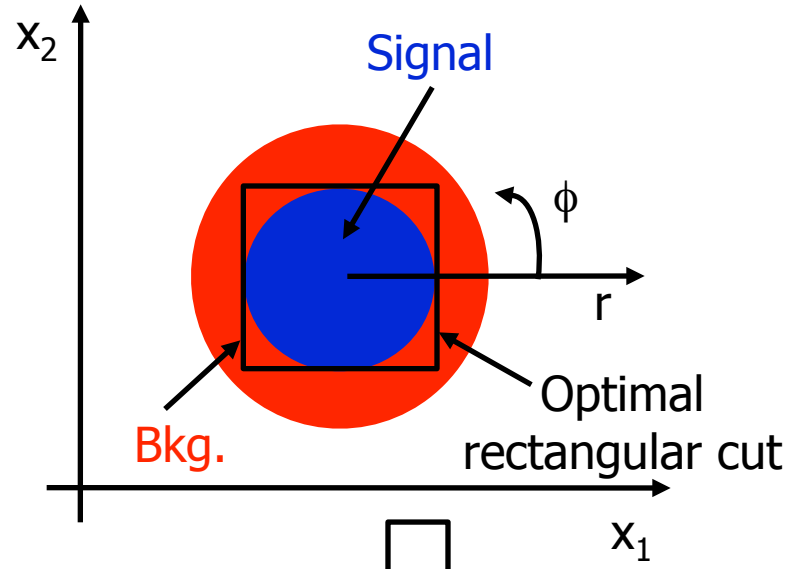
Search Concept

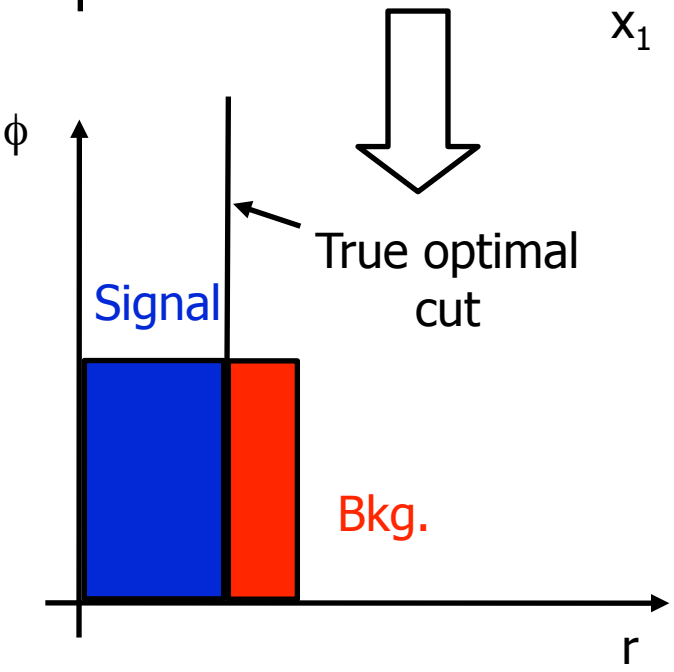
- Generally, one searches for a specific signal S in a sample composed of a potential signal and background B
- Classical hypothesis testing statistical problem, which is usually solved using the maximum likelihood method
- Various possible treatment of systematic uncertainties: profiling, integration, etc.
- Search analyses typically deal with the $S \ll B$ situation, which requires analysis *optimization*
  - ★ Various optimization figures of merit (FOM) are used, e.g., Gaussian significance S/sqrt(S+B), modified Gaussian significance taking into account systematics $S/\sqrt{S+B+\delta B^2}$, Punzi significance, etc.
  - ★ For low-background cases, important to use Poisson significance, which is usually done by maximizing the expected signal significance
  - ★ N.B. Beware optimizing for the best limit: what's the point of doing a search when you are venturing to set a limit *a priori*?
    - ✤ While in the Gaussian case, the discovery and limit based optimizations give the same optimal cuts, this is no longer true for the Poisson case!

Simple Analysis Methods

- **Straight (rectangular) cuts are not always a good idea, despite simplicity**
  - ★ **Requires careful choice of variables, often hard to find a priori**
  - ★ **May result in non-optimal cuts for correlated variables!**



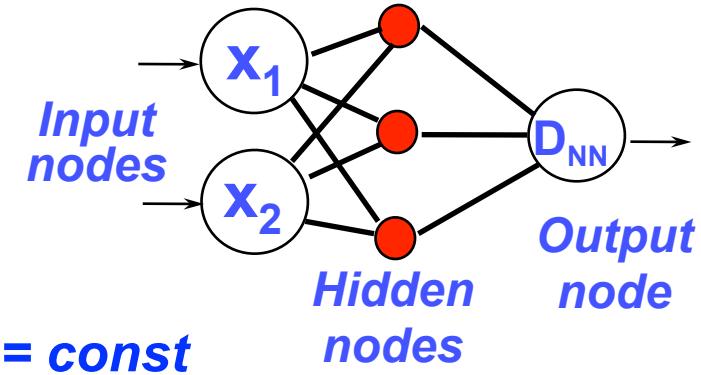


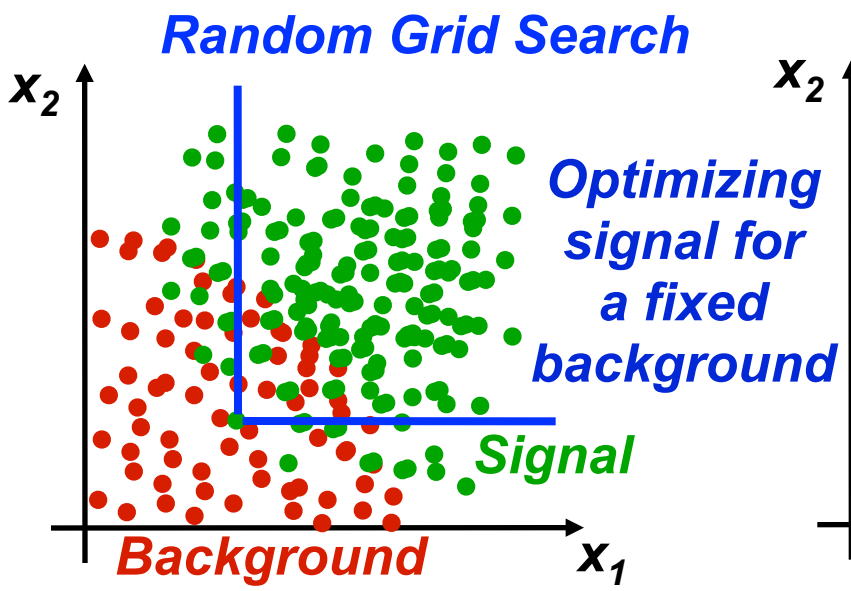


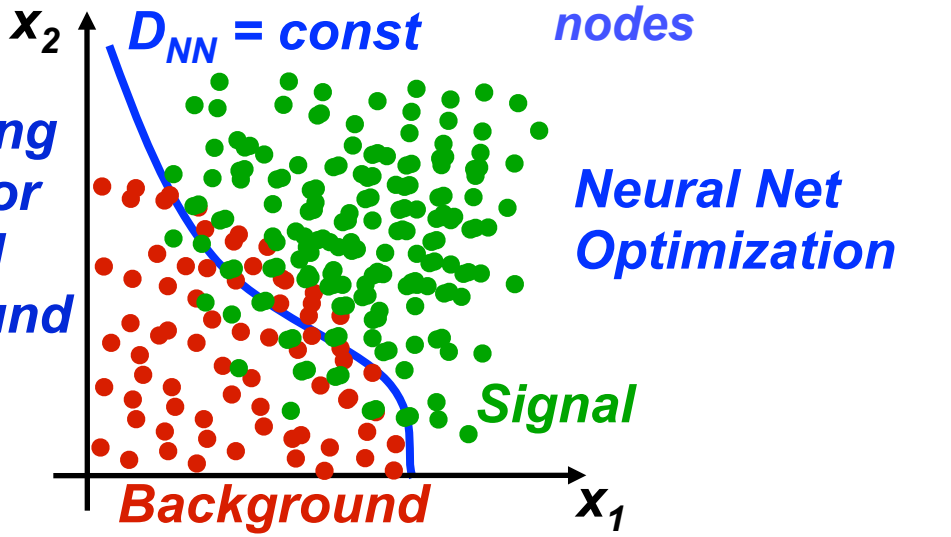
Multivariate Analysis Techniques (MVA)

- **Random Grid Search:**
  - Use kinematic parameters of MC events for signal, and MC or data events for backgrounds in order to find the optimal cuts, given chosen optimization criterion
- **Neural Nets/Boosted Decision Trees:**
  - Train a net on a mixture of signal MC and background, so that the output $D \approx 1$ for signal and $D \approx 0$ for background

- **Artificial neural nets approximate the way brain works via numerically solving set of non-linear ODE's**
- **Implemented in major stat packages, e.g., TMVA in Root**

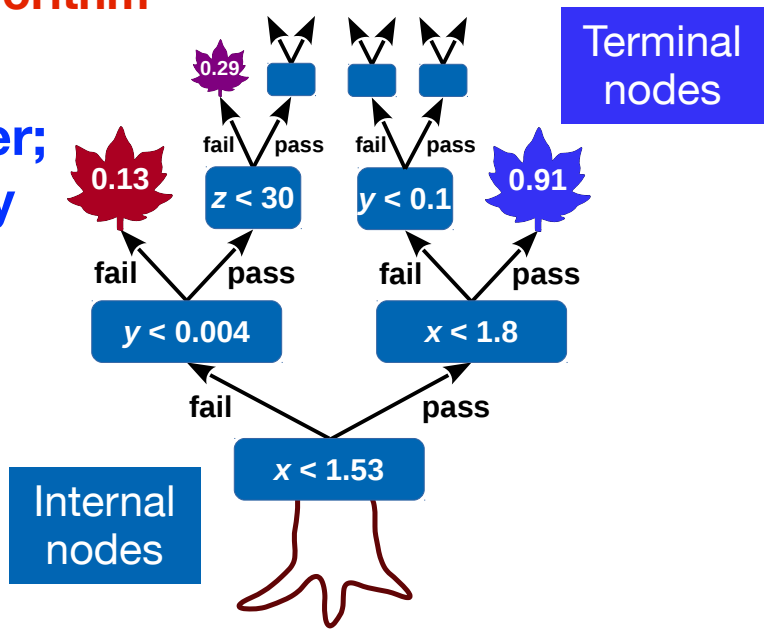






Boosted Decision Trees

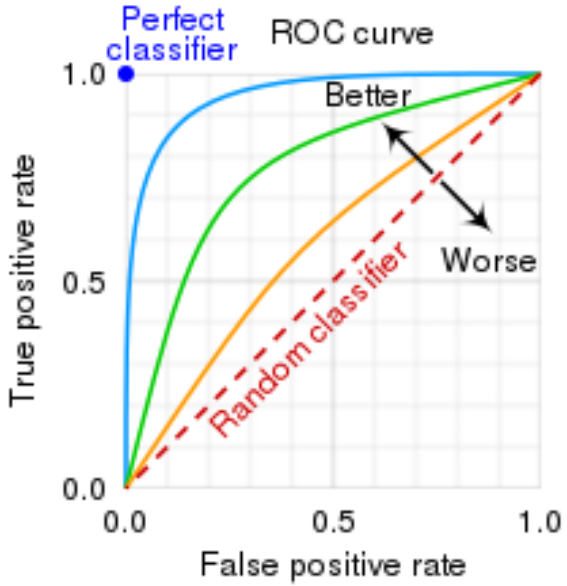
- BDTs are based on recursive binary trees:
  - ★ Sort data in each input variable
  - ★ For each variable find an optimal split value
  - ★ For the next node, select the variable and the split value that gives best separation
  - ★ If no improvement in separation can be achieved by further splitting, make the node a terminal node and exit the algorithm
  - ★ Apply the above steps recursively
- In reality, a single tree is a weak classifier; boost the performance by creating many trees (random forest) and weight them according to the separation power
- In the training of each tree use some new events and some events misclassified by a previous tree
- XGBoost is a powerful boosting library for decision trees



Coadou, arXiv:2206.09645

ROC Curves

- **Performance of an MVA algorithm is often characterized by a *ROC (Receiver Operating Characteristic) curve***
  - ★ **The term originated during WWII as a way to identify enemy objects on the battlefield using radars**
  - ★ **Used to compare the performance of various algorithms and to set working points, e.g., at a given true positive rate (efficiency) or a fixed false positive rate (misidentification)**
  - ★ **The algorithm with larger *area under curve* (AUC) performs better**
    - ✣ **AUC = 0.5 - random classifier**
    - ✣ **AUC = 1 - perfect classifier**
    - ✣ **Good algorithms typically have AUC > 0.9**

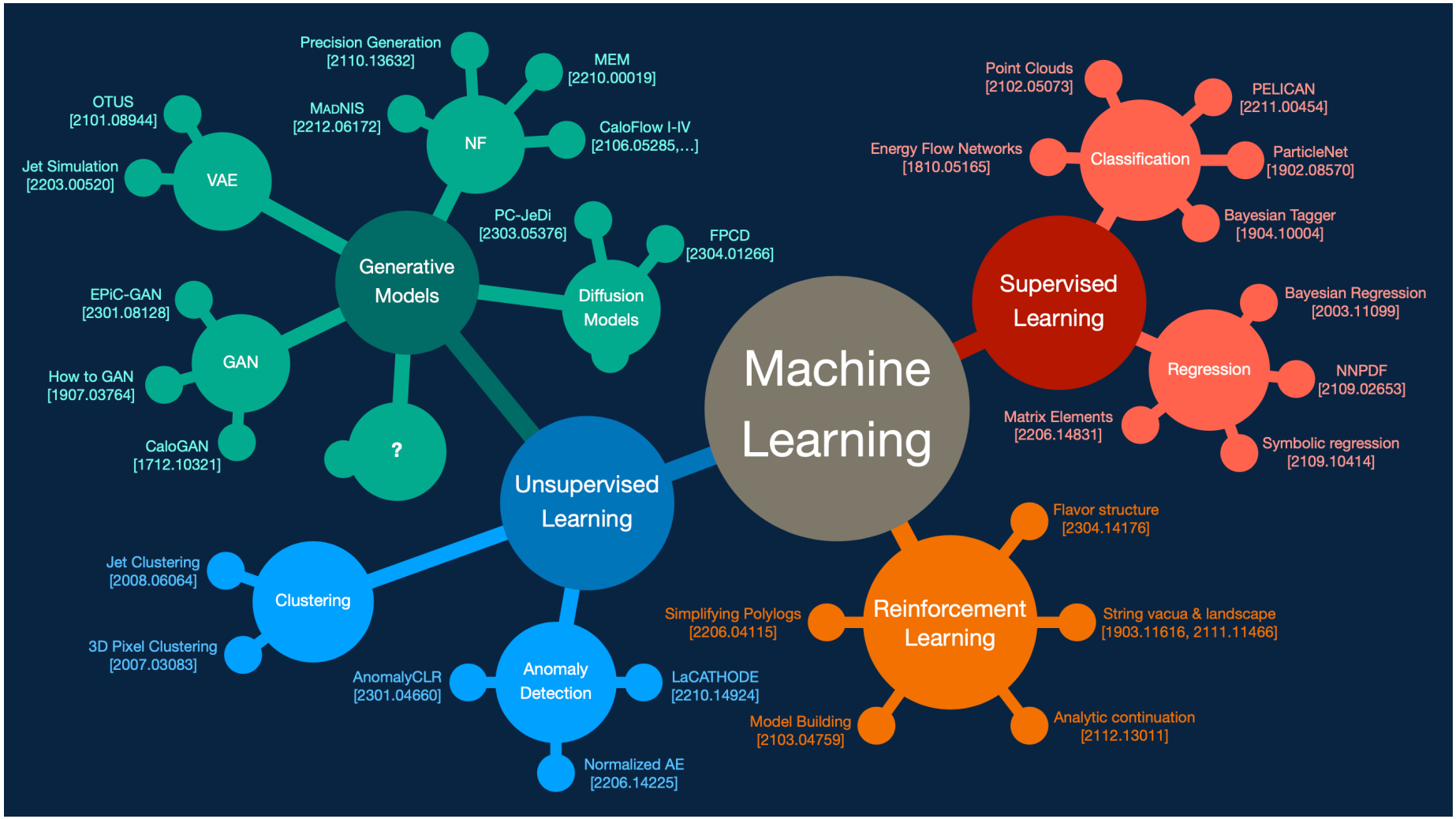


Deep Learning

- Advances in CPU and GPU allowed to take ANNs to a quantitatively new level, Deep Neural Networks
- They contain large number of hidden layers, resulting in billions (or sometimes trillions) nodes, thus allowing for a qualitative improvement in performance
  - ★ N.B. Human brain contains 80 billion neurons and 150 trillion synapses
- DNNs are a very popular tool these days, both in everyday life (ChatGPT) and particle physics
  - ★ ChatGPT 4 contains 170 trillion parameters - more than the human brain!
- Basic idea: minimize the so-called loss function over a set of parameters, using a non-linear function (NN)

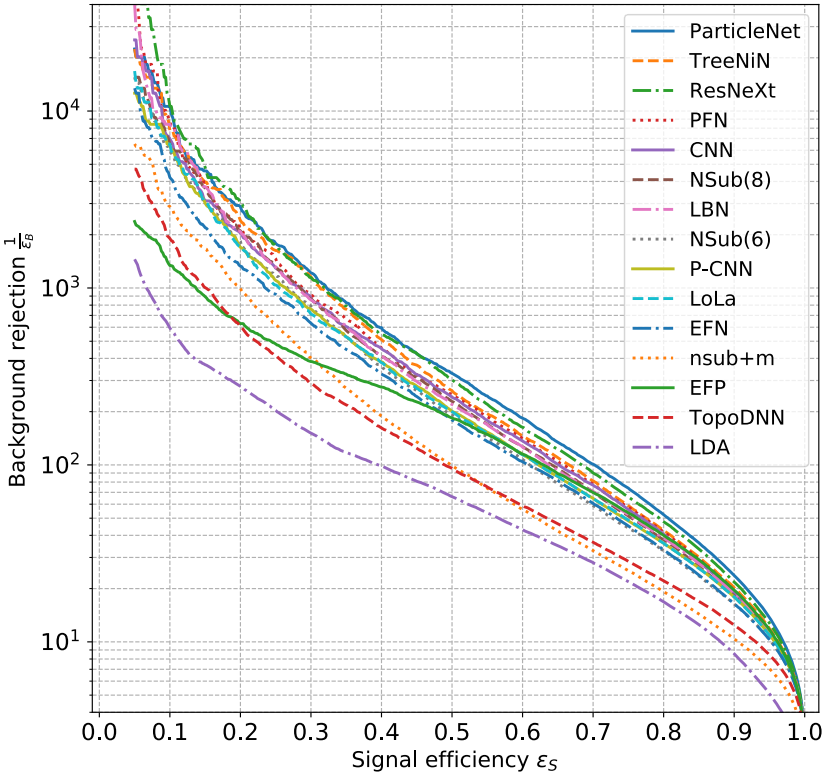
Landscape of ML for HEP

R. Winteralder



Amazing Tool for Particle ID

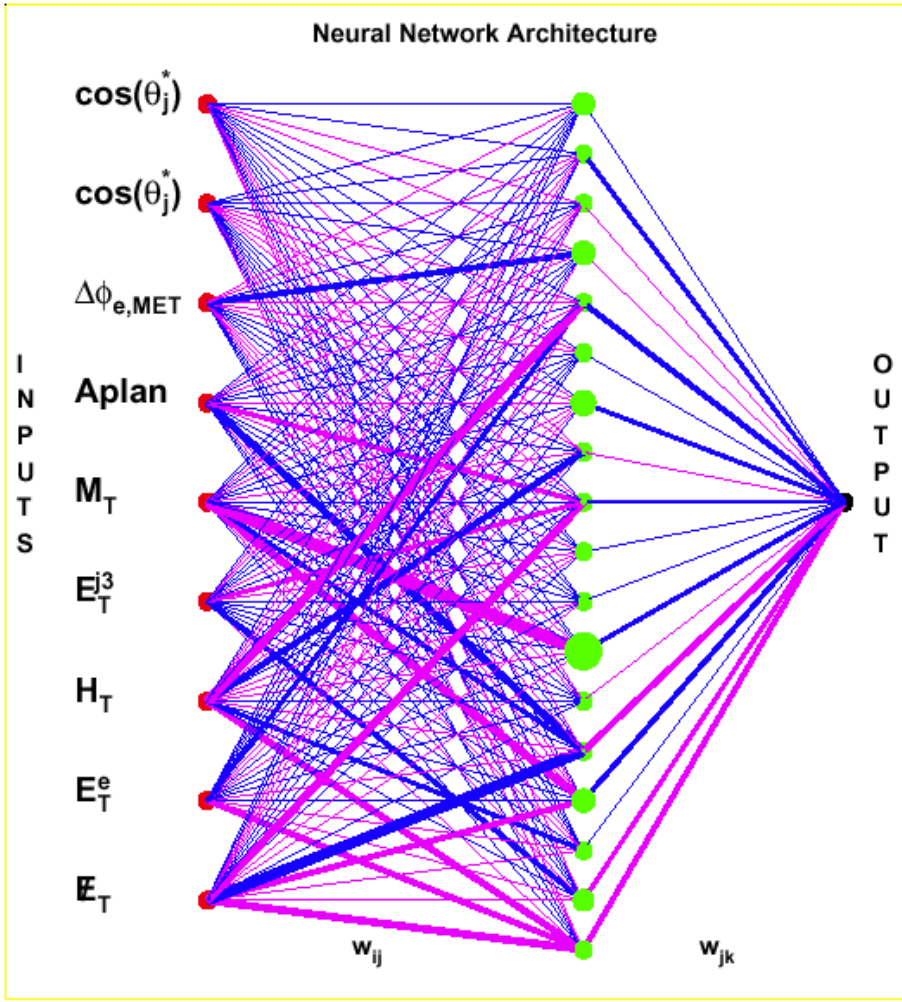
- e.g., ParticleNet - a deep graph network allowing to classify jets as t, b, c, q, g, etc.
- Deep learning based algorithms outperform dedicated taggers based on theoretical considerations
- Solved problem in particle physics; just requires further optimization and better training
- Well-understood performance, thanks to essentially infinite samples of different kinds of jets available both in MC and in data, allowing for efficient deep training and performance evaluation



Kasieczka et al., Sci. Post Phys. 7 (2019) 014

NN's: A Word of Caution

- It's very dangerous to use NNs (or other multivariate techniques) as black boxes
- Representative and large training samples are crucial, especially for deep neural networks
- Important to avoid overtraining and sculpting!
- Even complicated neural nets can be "opened" up
- Use of two-dimensional projections and other graphic tools to make sure that the NN acts reasonably



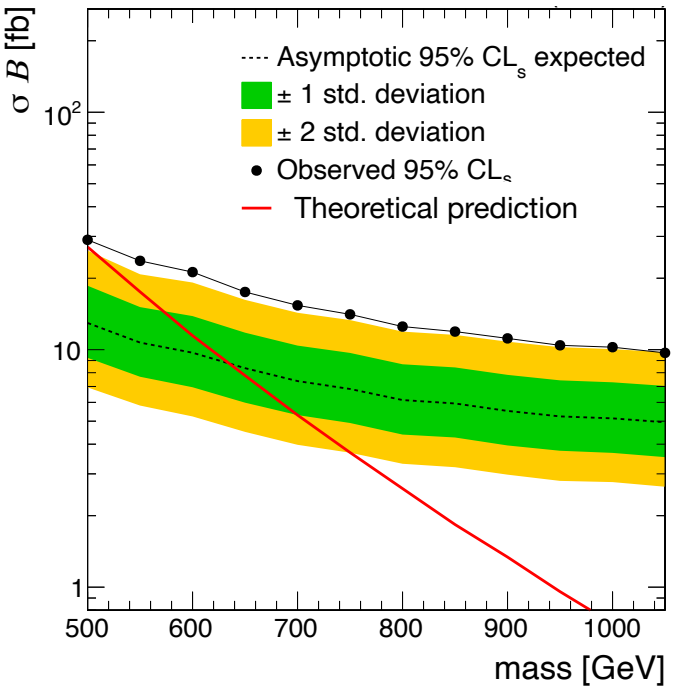
Use of DNNs in Analysis

- This is a much more debatable issue
- Unlike large training and test samples for particle ID purposes, there is only one reincarnation of the analysis
  - ★ How do you know that a DNN hasn't picked on some artificial feature of the signal MC sample, e.g., due to a lack of higher-order corrections?
  - ★ How do we know that the DNN is stable against small changes in the architecture?
  - ★ Not so easy to prove that the performance is correct with simulation, especially if very large background rejection is required
- Shall be used with a lot of caution and cross checks to avoid false "discoveries"

An Example of a Poor Optimization

- A complicated search for hypothetical particles in a high-multiplicity final states in a model with several types of new particles
- Chose DNN to optimize signal against the dominant QCD background, with a lot of attention given to avoid the mass sculpting
  - ★ Clearly optimized for a limit, not a discovery!
  - ★ Surprise after the unblinding: a nearly 3σ excess, but no clue at what mass!
- Lesson learned: poor optimization of the analysis, not allowing to even design a dedicated analysis with future data
  - ★ Excess is likely driven by the DNN features, but the way the analysis is done makes it very hard to cross-check the result by using simpler means

Plot from a real publication with identifiable information removed

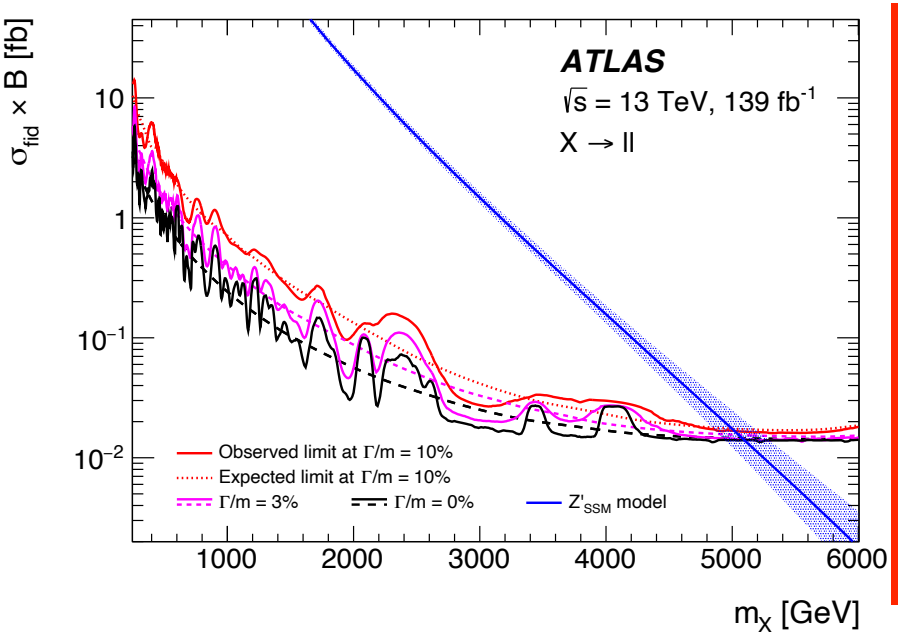


Look-Elsewhere Effect

- This brings us to a concept of the look-elsewhere effect (LEE)
  - ★ If one rolls a dice, a probability to get six is 1/6 ≈ 0.17
  - ★ If one rolls three dices, a probability that one of them will have six is $1 - (5/6)^3 = 0.42$
  - ★ The ratio r = 0.42/0.17 ≈ 2.5 is the *trial factor*
- Similarly, if one looks for a signal in multiple search regions, the local p-value of the largest excess among them needs to be multiplied by a trial factor in order to get the *global p-value* of an excess
  - ★ The trial factor is usually estimated with pseudo-experiments, but there are also analytical asymptotic methods [Gross and Vitells, Eur. Phys. J. C 70 (2010) 525]
- Therefore, the more SRs one has, the larger the trial factor is, and the lower the global p-value is, compared to the local one
- Consequently these model-independent searches [e.g., D0's Sleuth, PRL 86 (2001) 3712, ; ATLAS, Eur. Phys. J. C 79 (2019) 120; CMS' MUSiC, EPJC 81 (2021) 629] are inherently less powerful than model-specific searches that can be properly optimized without paying the trial factor price
  - ★ They are good as a grand finale kind of thing: e.g., have we missed anything interesting in the LHC Run 2 data, despite all the model-specific searches we published?
  - ★ While fun to conduct, they are unlikely to become a discovery tool

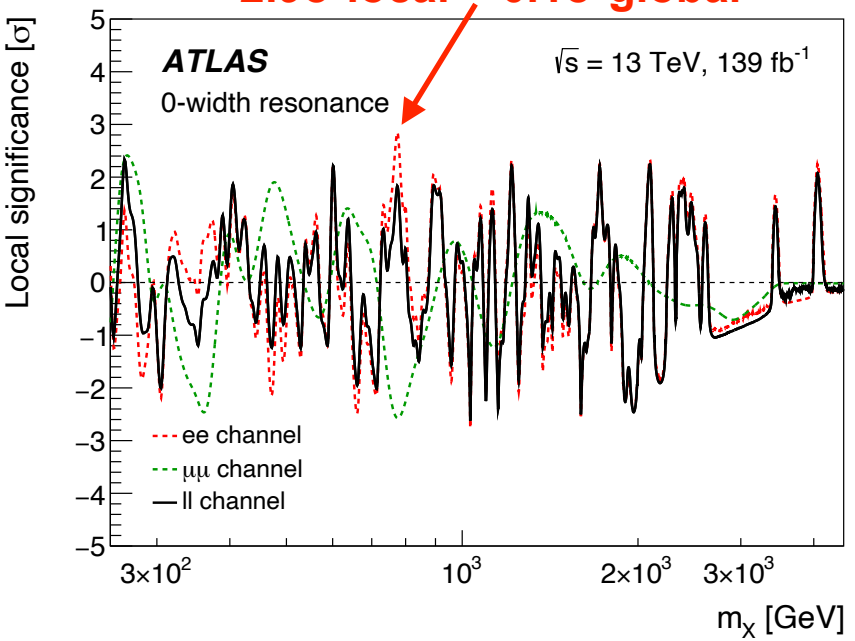
An LEE Example

- Typically, LEE is very pronounced in searches in a high-resolution channel
  - ★ An infamous X(750) was a classic example of people underestimating LEE
- Roughly speaking the trial factor is approximately equal to the search range divided by the average resolution, i.e., how many non-overlapping signals one could "fit" in the mass spectrum probed
- In the case of ATLAS dielectron X(ee) search, the trial factor is ~50



ATLAS PLB 796 (2019) 68

2.9σ local = 0.1σ global



Supervising the Machine

- Typical problem in particle physics (as well as many other areas) is to uncover a subtle signal hidden in large background
  - ★ This is a classical anomaly detection problem: find "anomalous" (signal-like) events in a sea of "non-anomalous" (background-like) events
- The most powerful method is fully supervised machine learning algorithm
  - ★ Supervision essentially means labeling
  - ★ One exposes an ML algorithm to a set of training data with *labels*, which could be "signal" (S), "bkg1", "bkg2", etc.
- The most powerful discriminant based on a set of observables $\vec{x}$ according to the Neyman-Pearson lemma is the likelihood ratio:

$$L(\vec{x}) = \frac{p(\vec{x}\,|\,\mathbf{S})}{p(\vec{x}\,|\,\mathbf{B})}$$

  - ★ A fully supervised ML classifier is built to approximate this formula by using the known labels ("bkg." (B) = {"bkg1","bkg2",...})
- ML is just a fancy version of any other optimization technique, where it is used to find a classifier $h(\vec{x})$ that is monotonically proportional to $L(\vec{x})$

Supervising the Machine (cont'd)

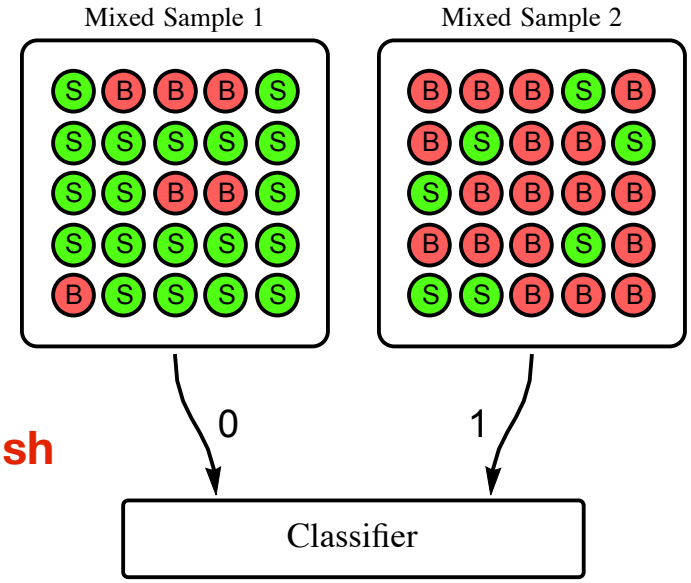
- In practice, one defines a loss function, which is numerically minimized
- An example of a loss function MSE (mean squared error; an analogue of the χ^2):
  - ★ $L_{\text{MSE}} = \frac{1}{N}\sum_{i=1}^{N}\left(h(\vec{x}_i) - \mathbb{I}(u_i = S)\right)^2$, where *N* is the number of events in the training sample, $\mathbb{I}$ is the indicator function ($\mathbb{I}$ = 1 if the argument is true and 0 otherwise), and $u \in$ {S, B} is the label
- Another example of a loss function is the cross-entropy:
  - ★ $L_{\text{CE}} = -\frac{1}{N}\left(\mathbb{I}(u_i = S)\log h(\vec{x}_i) + (1 - \mathbb{I}(u_i = S))\log\left(1 - h(\vec{x}_i)\right)\right)$
- With large enough *N*, flexible model parameterization, and a good minimizer, ML'ed *h* will approximate the optimal likelihood ratio classifier
- These are the basics of the *supervised* ML approach

Weak Supervision

- But what if we are not confident about labeling the training sample?
  - ★ Could happen when backgrounds come from control samples in data with potential signal contamination
- In this case, we deal with a mixture of S and B events, where we are not exactly sure about the labels
- Let's consider two processes M_1 and M_2 that mix S and B events in the following way: $p_{M_1}(\vec{x}) = f_1 p(\vec{x}\,|\,S) + (1-f_1)p(\vec{x}\,|\,B)$,
  $p_{M_2}(\vec{x}) = f_2 p(\vec{x}\,|\,S) + (1-f_2)p(\vec{x}\,|\,B)$, with $0 \le f_2 < f_1 < 1$
- Now, instead of S and B labels, we only know M_1 and M_2 labels and *known* signal fractions $f_{1,2}$
- It turns out that one still could learn optimal h in this case; they key is to use several samples with sufficiently different fractions $f_{1,2}$ to avoid degenerate cases
- A possible loss function to use is:
  - ★ $L_{\text{LLP}} = \frac{1}{N_1}\left|\sum_{i=1}^{N_1} h(\vec{x}_i) - f_1\right| + \frac{1}{N_2}\left|\sum_{i=1}^{N_2} h(\vec{x}_i) - f_2\right|$, where N_i is the number of training events in the M_i sample
- One could easily generalize this for the case of more than 2 samples

Weak Supervision: CWoLa

- The idea behind CWoLa (Classification Without Labels, pronounced "koala") is instead of modifying the loss function, simply use fully supervised ML to train a classifier $h(\vec{x})$ to distinguish M_1 from M_2 and then use it directly to separate S from B



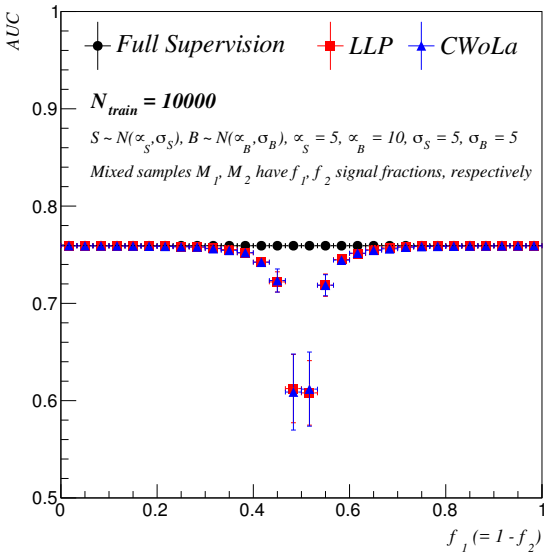
  - ★ Remarkably, the optimal classifier to distinguish
  - ★ M_1 from M_2 is also the optimal classifier to distinguish S from B!
  - ★ Indeed, $L_{M_1/M_2} = \dfrac{p(\vec{x}|M_1)}{p(\vec{x}|M_2)} = \dfrac{f_1 p(\vec{x}|S) + (1-f_1)p(\vec{x}|B)}{f_2 p(\vec{x}|S) + (1-f_2)p(\vec{x}|B)} = \dfrac{f_1 L_{S/B} + (1-f_1)}{f_2 L_{S/B} + (1-f_2)}$, which is a monotonically increasing rescaling with $L_{S/B}$ as long as $f_1 > f_2$
    - ✤ Easy to check by noting that $\partial L_{M_1/M_2}/\partial L_{S/B} = \dfrac{f_1 - f_2}{(f_2 L_{S/B} - f_2 + 1)^2} > 0$ for $f_1 > f_2$
- The beauty of CWoLa is that unlike the previous example, one does not need to know f_1 and f_2, as long as $f_1 > f_2$!

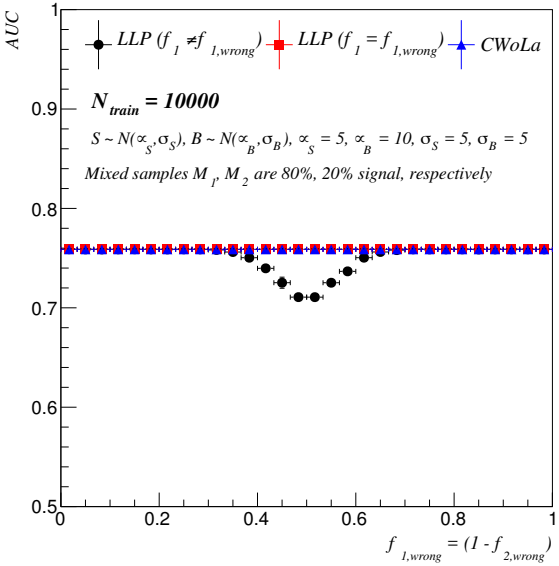
Metodiev, Nachman, Thaler, JHEP 10 (2017) 174

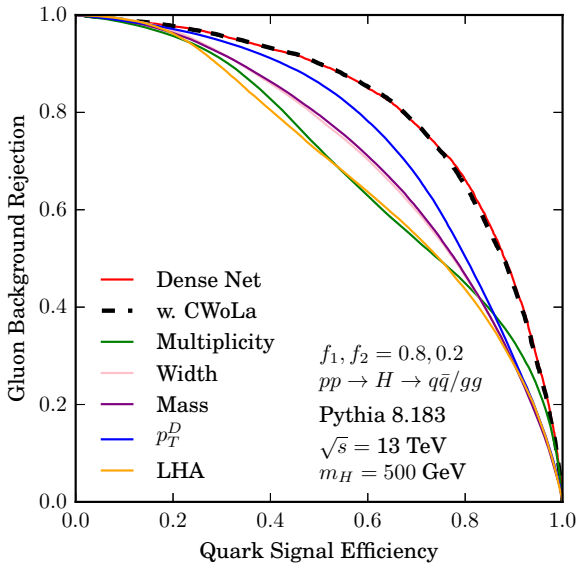
CWoLa Performance

- Compare full supervision, LLP method, and CWoLa
- Compare LLP and CWoLa when a wrong f_i is passed to LLP
- Performance of CWoLa on a realistic q/g jet separation example

Metodiev, Nachman, Thaler, JHEP 10 (2017) 174







Unsupervised Learning

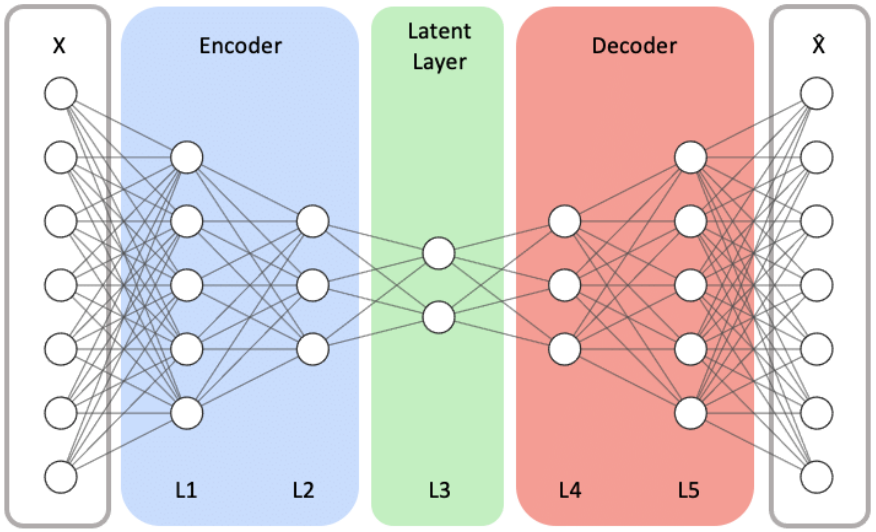
- Given the success of weakly supervised learning, it is very tempting to try optimizing a search for an arbitrary signal
  - ★ Unfortunately, this is not mathematically possible, as has been shown by Pearson in 1930-ies
  - ★ Only the likelihood ratio (e.g., L(S+B)/L(B)) possess well-defined properties; while individual likelihoods change property under transformation of variables
    - ✤ That means that "optimizing" in, e.g., p_T or p_T^2 yields completely different results
  - ★ Unfortunately, this is largely underappreciated in the HEP community, and every few years there is another attempt to reinvent the wheel and do "model-independent optimization", which is simply mathematically impossible (latest attempts use machine learning!)
- What can be done is what's typically done in SUSY searches: create a large number of independent or overlapping search categories ("signal regions", SRs) and look for the maximum excess of events over the respective background prediction among them
- Alternatively, one can simply search for "anomalies" in data using unsupervised ML based on background-dominated samples

ML Anomaly Detection

- Unsupervised ML approach does not use labels at all
- It is based on learning $p(\vec{x}\,|\,B)$ using large sample of background-dominated events (either from simulation or data) and then focus on the cases of $p(\vec{x}\,|\,B) \ll 1$
- The most common approach to learning $p(\vec{x}\,|\,B)$ currently used involves data compression, which essentially eliminates rare features from the background-dominated sample, by focusing on the region of moderately large $p(\vec{x}\,|\,B)$
  - ★ This makes it a powerful tool to identify anomalies in a sample that contains potential signal, i.e., focus on the events that look differently from the bulk of the background
- A powerful tool to achieve this is an *autoencoder*

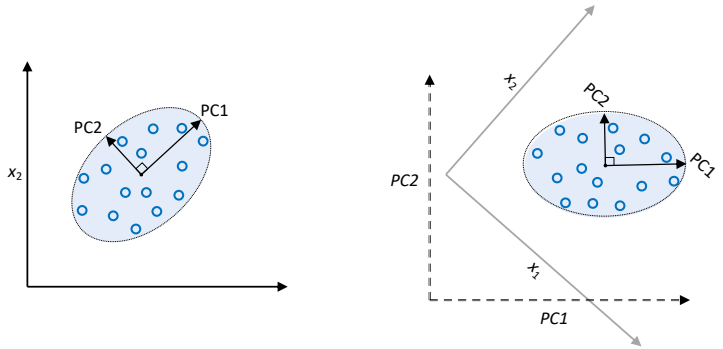
Autoencoders

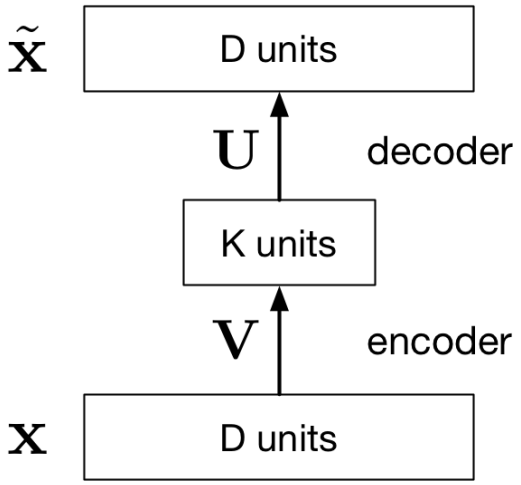
- An autoencoder is essentially a lossy ML compression algorithm
- An autoencoder consists of an encoder that could be a multi-layered fully connected or convolutional network (you will have an opportunity to play with them during the hands-on session), a latent layer with a significantly reduced dimensionality, and a decoder that typically mirrors the encoder and restores the dimensionality of the input
- The latent layer could be as simple as 2D, thus reducing the dimensionality of the input considerably and yet being able to capture the characteristics of the input



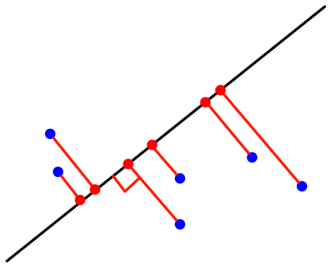
Autoencoders (cont'd)

- A simple autoencoder is similar to principle component analysis (PCA), which transforms features onto directions of maximum variance





- The simplest kind of autoencoder has one hidden layer, linear activations, and squared error loss function: $L(\vec{x}, \vec{\tilde{x}}) = \| \vec{x} - \vec{\tilde{x}} \|^2 \equiv \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{n} (x_i^j - \tilde{x}_i^j)^2$, where i is the index over the training data set (of size *N*) and j is the vector index (dim *n*)
- This network computes $\vec{\tilde{x}} = UV\vec{x}$, which is a linear function of the input
- If K < D, V maps x to a K-dimensional sub-space S, so it is performing the dimensionality reduction
- A linear autoencoder essentially does the PCA by the projection of x onto S, i.e., onto the point in S, which minimizes the distance to x

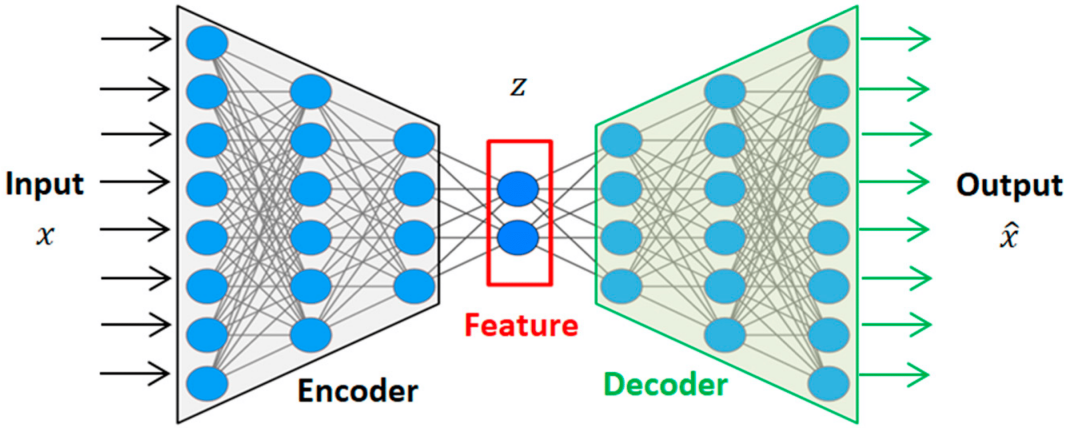


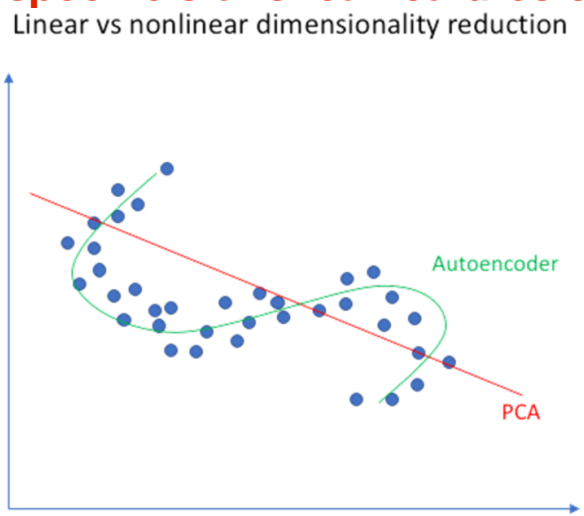
Autoencoders (cont'd)

- A more powerful autoencoder uses a deep architecture and non-linear encoder/decoder function
  - ★ Convolutional layers are often used for image processing
- Typical loss function used is either squared error or cross-entropy, defined as

$$L(\vec{x}, \vec{\tilde{x}}) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{n}\left(x_i^j \log \tilde{x}_i^j + (1 - x_i^j)\log(1 - \tilde{x}_i^j)\right)$$

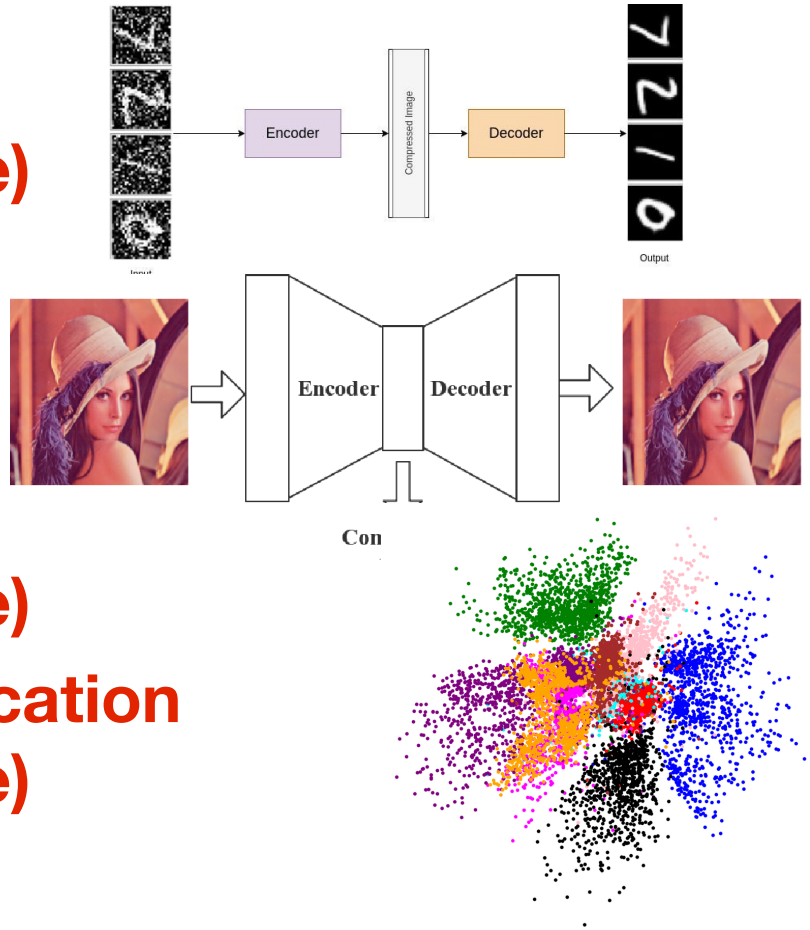
- Often a regularization term is added to the loss function, e.g., a sparsity condition, which essentially requires that only small fraction of a hidden layer neurons is activated
  - ★ This is useful for, e.g., classification task or to learn specific statistical features of the training data set





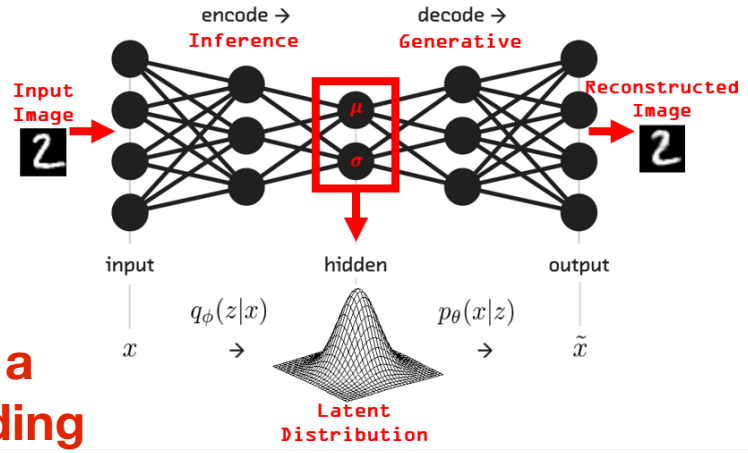
Autoencoder Applications

- There are many applications of autoencoders, apart from the anomaly detection
- Some examples include:
  - ★ Denoising (see the hands-on exercise)
  - ★ Lossy compression (a la MPEG or GIF) and the dimensionality reduction
  - ★ Data generation (see the hands-on exercise)
  - ★ Feature extraction/classification (see the hands-on exercise)



Variational Autoencoders

- This is a special class of autoencoders that are widely used for feature extraction and data generation
- One could think about the variational autoencoder as an autoencoder whose training is regularized to avoid overfitting, with the latent space that has good properties that enable generative process
  - ★ Instead of encoding a single input as a single point in the latent space, we encode it as a *distribution* in the latent space!
  - ★ In practice, a Gaussian distribution with a mean μ and RMS σ is used for the encoding
  - ★ The regularization is based on the Kullback–Leibler (KL) divergence, which is a measure of how much one statistical distribution ***P*** is different from another **Q**: $D_{\mathrm{KL}}(P \parallel Q) = \int_{-\infty}^{\infty} p(x) \log \frac{p(x)}{q(x)} dx$, where $p(x)$ and $q(x)$ are the pdf's of ***P*** and ***Q***

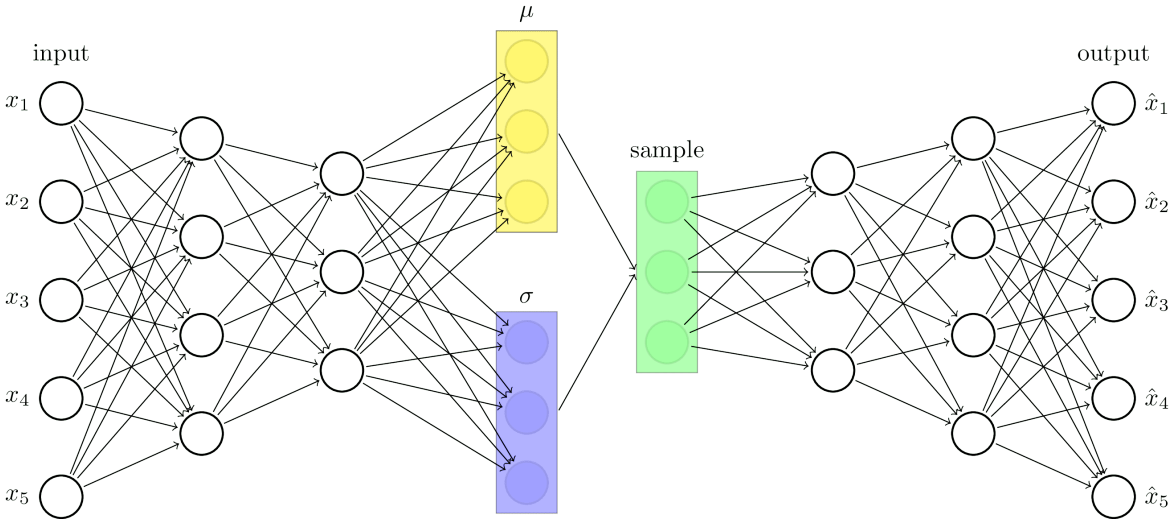


Variational Autoencoders (cont'd)

- In practice, one usually uses dense networks for encoding/decoding
  - ★ A reparameterization trick is often used by representing z ~ G(μ,σ) with ε = G(0,1) as z = εσ + μ
- The loss function is defined as a sum of the reconstruction loss (e.g., cross-entropy) and KL divergence (which is equivalent to maximizing the *evidence low bound* (ELBO))
  - ★ For two Gaussian distributions, G(μ,σ) and G(0,1), the KL divergence is equal to

$$D_{\mathrm{KL}} = -\frac{1}{2}\sum_{i=1}^{N}(1+\log(\sigma_i^2)-\mu_i^2-\sigma_i^2)$$

- Sampling the latent space can used to generate new objects, similar to the one the autodecoder was trained on, with a continuous "morphing"
- Widely used for generative approaches, such as image generation



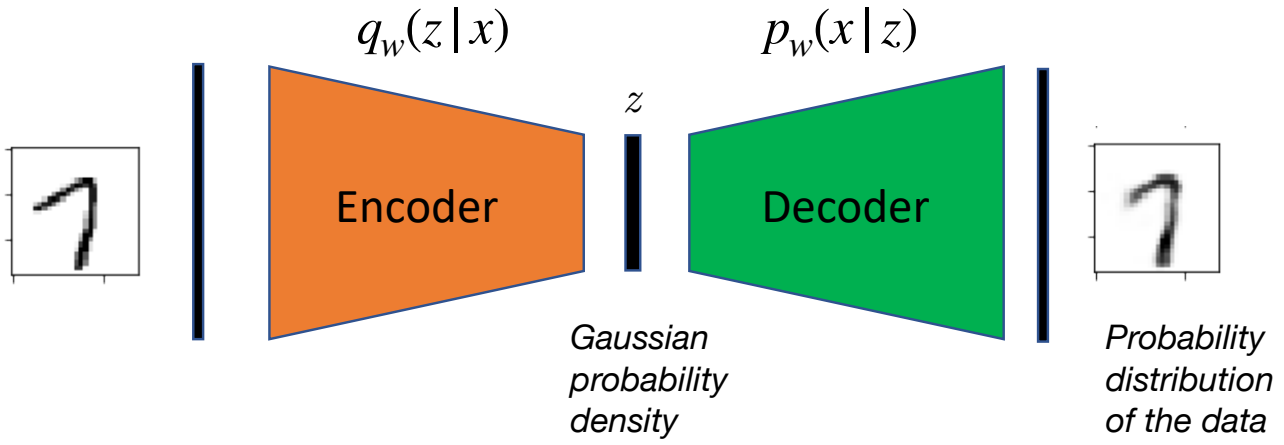
Variational Autoencoders (cont'd)

- In practice, one usually uses dense networks for encoding/decoding
  - ★ A reparameterization trick is often used by representing z ~ G(μ,σ) with ε = G(0,1) as z = εσ + μ
- The loss function is defined as a sum of the reconstruction loss and KL divergence (which is equivalent to maximizing the *evidence low bound* (ELBO))

$$\mathcal{L} = -\mathbb{E}_{z\sim q_w(z|x^{[i]})}\left[\log p_w\left(x^{[i]}|z\right)\right] + \mathbf{KL}\left(q_w\left(z|x^{[i]}\right)\|p(z)\right)$$

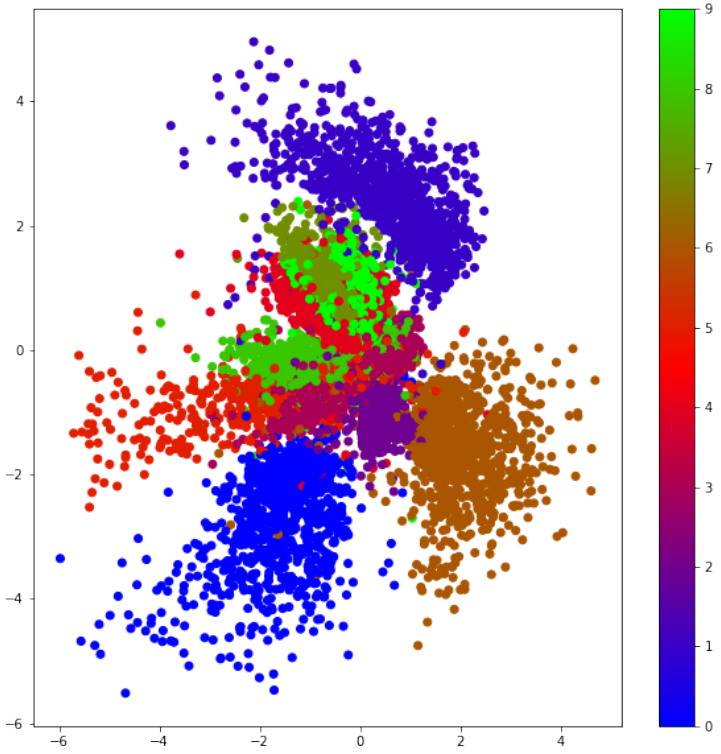
Expected neg. log likelihood term; wrt to encoder distribution

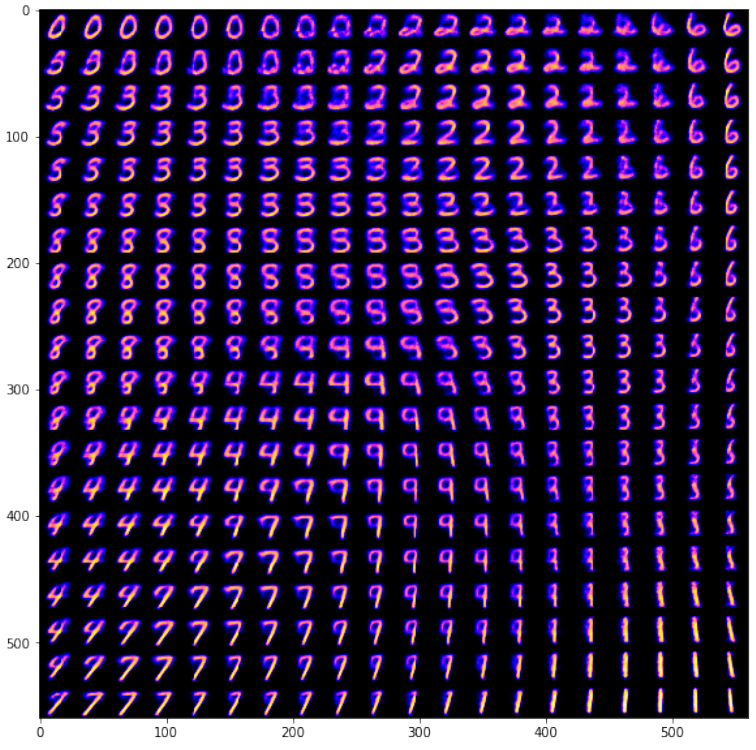
Kullback-Leibler divergence term where $p(z) = \mathcal{N}\left(\mu = 0, \sigma^2 = 1\right)$



Variational Autoencoders (cont'd)

- Unlike for a regular autoencoder, the latent space of the variational autoencoder is continuous and has good properties
- Sometimes one does not need the decoder part of the autoencoder and could simply infer features of the training data set from the latent space itself
- Example (in your hands-on exercise): train an autoencoder on the MNIST data set of handwritten digits (0-9) and look at the separation in the latent space, as well as continuous generation of digits



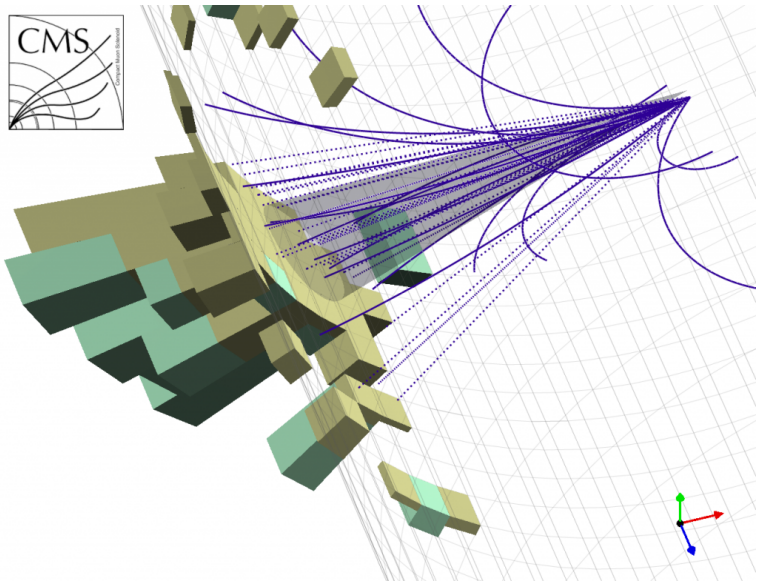


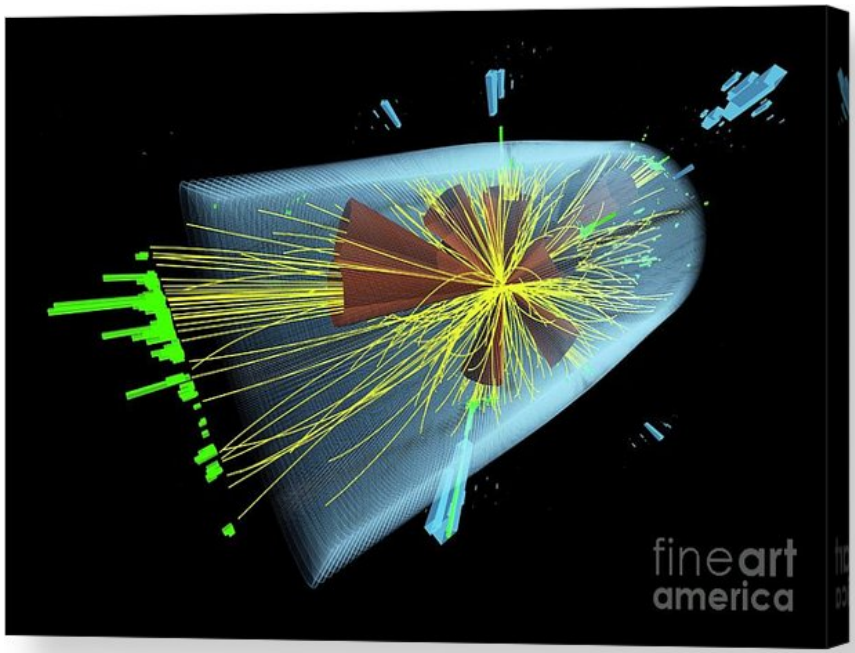
Back to Anomaly Detection

- Now, when we learned a great deal about autoencoders, how can we use them for unsupervised anomaly detection?
- One have to be careful of Pearson's theorem about impossibility to optimize vs. the unknown signal, so one can't just use autoencoders out of the box to do a proper search
- Indeed, the meaning of $p(\vec{x} \mid B)$ if applied to a signal event generally does depend on the signal
- If one simply focuses on events with low $p(\vec{x} \mid B)$, the look-elsewhere effect is very hard to compute as it's hidden both in the training of the autoencoder and the sample used for finding anomalies
- Thus, the autoencoder needs to be augmented with proper statistical analysis in order to produce meaningful results, such as a hypothetic signal significance or a limit on such signal

Example of Anomaly Detection: Jets

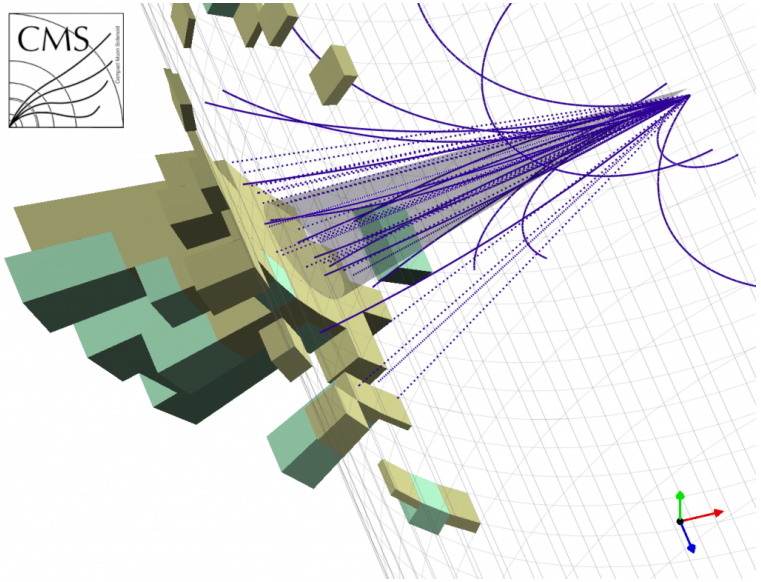
- As an example of anomaly detection via autoencoders, we could use jets - collimated stream of particles produced as a result of fragmentation and hadronization of quarks and gluons abundantly produced at the LHC
- One of the projects our group is pursuing is to identify anomalies in the very jet structure (i.e., looking for jets from anomalous sources)





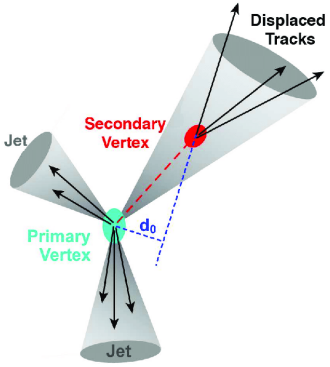
Example of Anomaly Detection: Jets

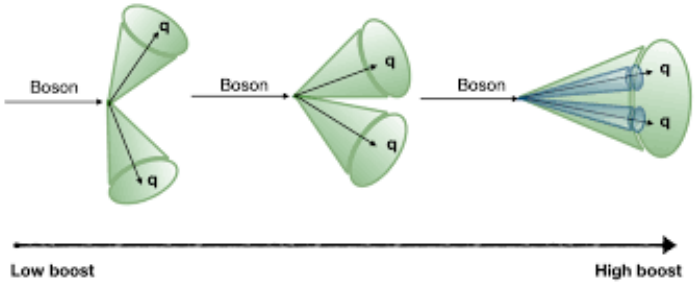
- As an example of anomaly detection via autoencoders, we could use jets - collimated stream of particles produced as a result of fragmentation and hadronization of quarks and gluons abundantly produced at the LHC
- One of the projects our group is pursuing is to identify anomalies in the very jet structure (i.e., looking for jets from anomalous sources)

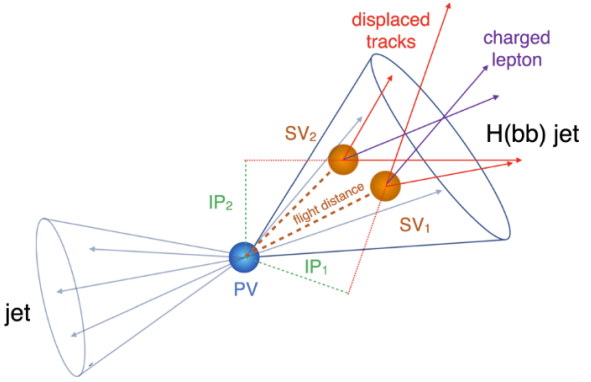


Jet Anomalies

- The idea is to use both low- (e.g., distribution of individual particles in a jet) and high-level (e.g., secondary vertices inside the jet) features of a jet as inputs to an autoencoder and train it on a large sample of ordinary QCD jets (obtained either incisively in data or from simulation)
  - ★ This way, the autoencoder will learn basic features of ordinary jets
- We can now apply it to search for "known anomalies", e.g., b-quark jets, which have displaced tracks and secondary vertices or jets that are products of decays of highly Lorentz-boosted objects into quarks and gluons, which are reconstructed as single jets with a characteristic substructure
  - ★ We could use b quark jets produced in top quark decays to see if the autoencoder recognizes them as anomalous (given that their fraction in the training sample is pretty small)
  - ★ For example, Lorentz boosted W boson decays result in two-prong jets with different spatial energy distribution from QCD jets in the training sample; will the autoencoder tag them as "anomalous"?
  - ★ We could also use boosted H → bb decays and see if the autoencoder tags these jets as "double-anomalous"





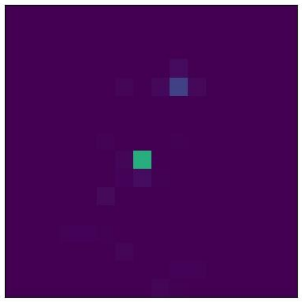


Jet Anomalies: Ultimate Goal

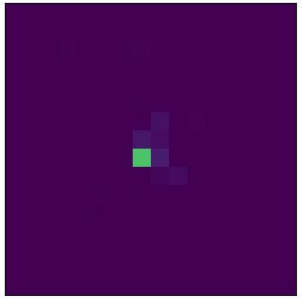
- Progressively add the "known anomalies" to the training samples and then let the autoencoder to run on collision data
- Focus on jets with very high anomaly score (very low $p(\vec{x} \mid B)$) and examine them
- Will likely find various instrumental effects (e.g., dead calorimeter channels), but could also find jets that do not look as known jets at all, which may signal new physics
- In the latter case, could construct a dedicated search for jets with the observed odd features in orthogonal data sets and/or in future data

First Results

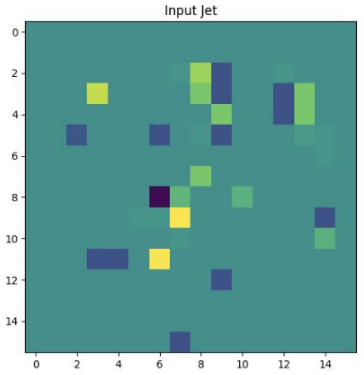
- **Use a regular multilayer autoencoder with a mix of convolutional and fully-connected layers**

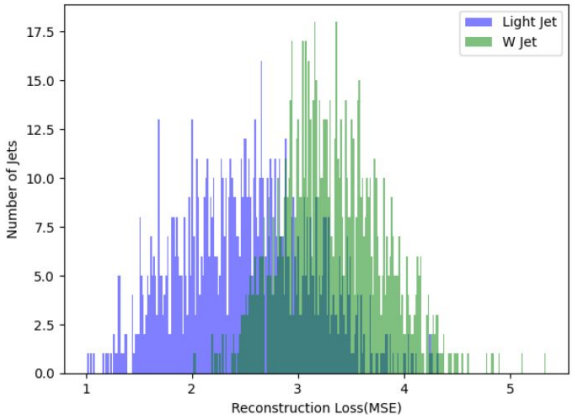


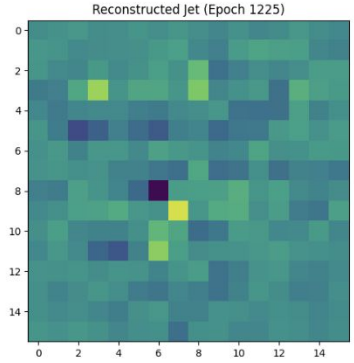
W-jet image, satellite structure

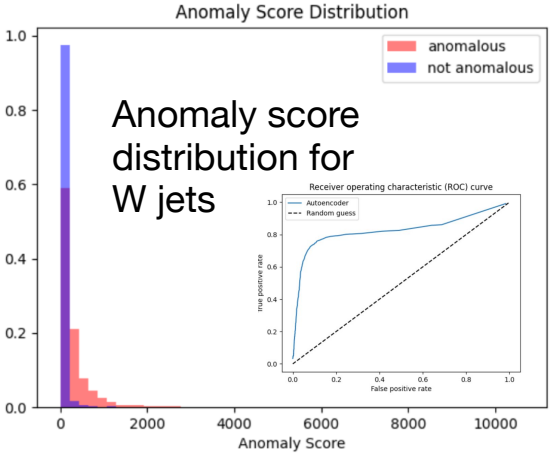


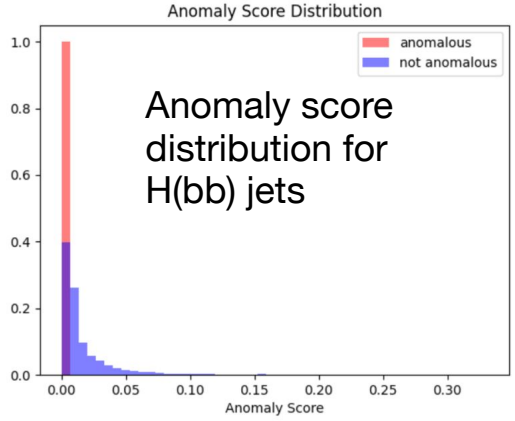
Light jet image





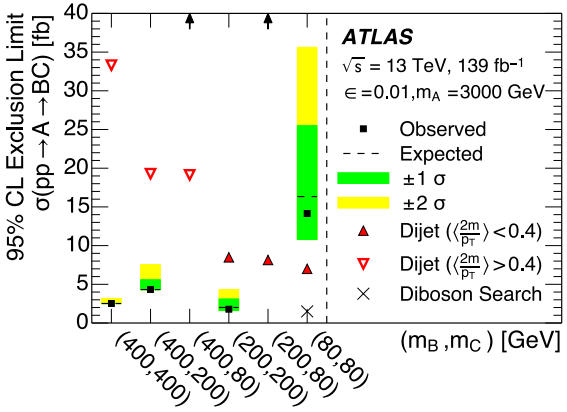


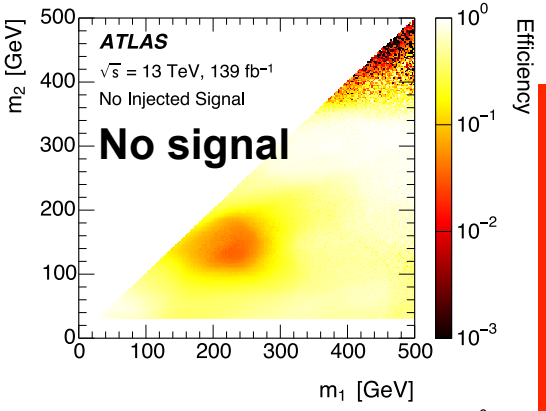


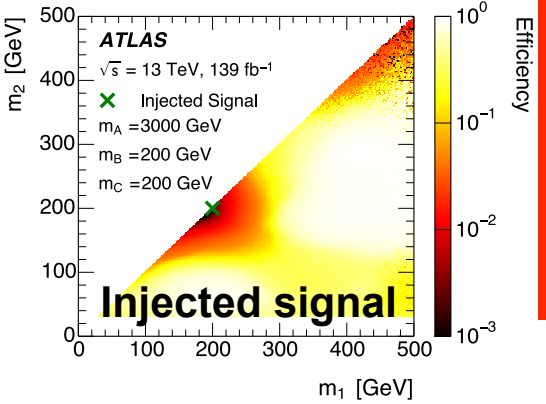


Resonance Search w/ Weak Supervision

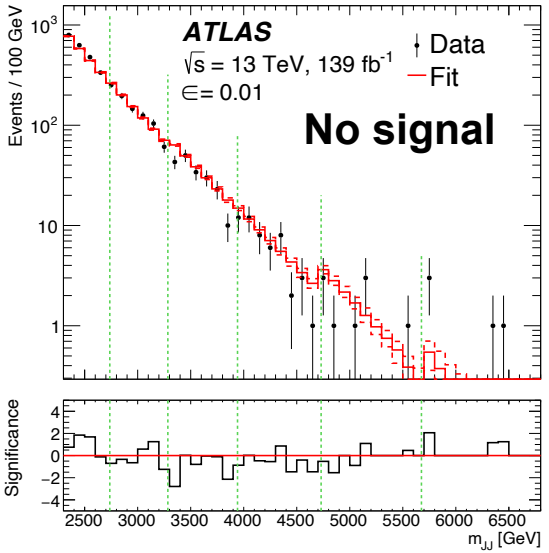
- ATLAS search for A → BC decays with B and C particles reconstructed as massive jets
- Uses weakly supervised machine learning to search for a bump-like signal across a large phase space without a specific physics model
  - ★ Uses data to infer the background model, even in the presence of signal; validated by using signal-suppressed validation region with large jet |Δy| separation
  - ★ NN are trained in the plane of two jet masses mass and learn about localized excesses
  - ★ The dijet mass distribution is then used to look for resonance A
- Sets limits stronger than dedicated searches for dijet resonances or vector boson pairs (far from W/Z)
- ✦ Con: if an excess is seen, it would be very hard to estimate the look-elsewhere effect in such an analysis

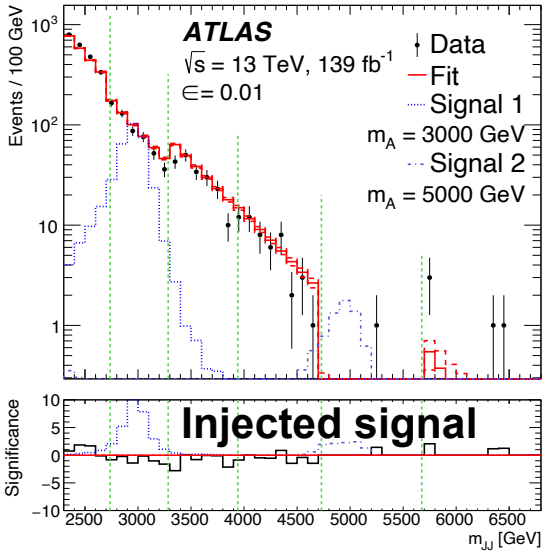






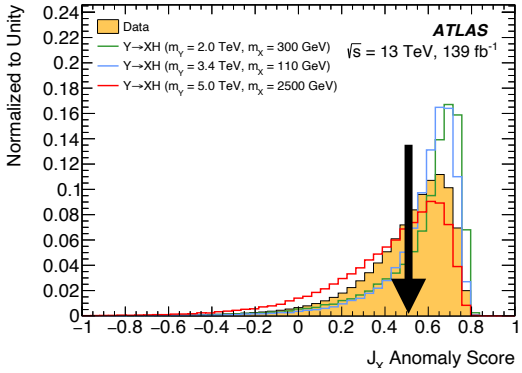
ATLAS, PRL 125 (2020) 131801

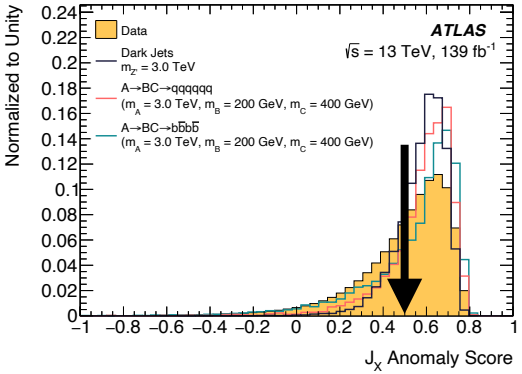


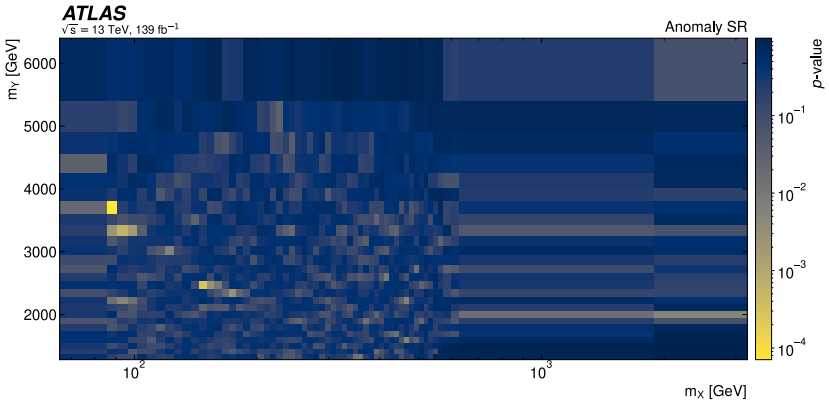


Unsupervised Anomaly Detection

- New ATLAS result focusing on Y → X(J)H(J_{bb}) in the Lorentz-boosted regime (two merged jets)
  - ★ H(J_{bb}) is identified via dedicated double-b tagger
  - ★ X(J) is sought using jet anomaly score determined by unsupervised ML via a variational recursive neural net trained on jets in data
    - ✣ Sensitive to various hadronic decays of X, e.g., into two b quarks, dark jets, or three prongs
  - ★ Background is determined from Higgs boson mass sideband

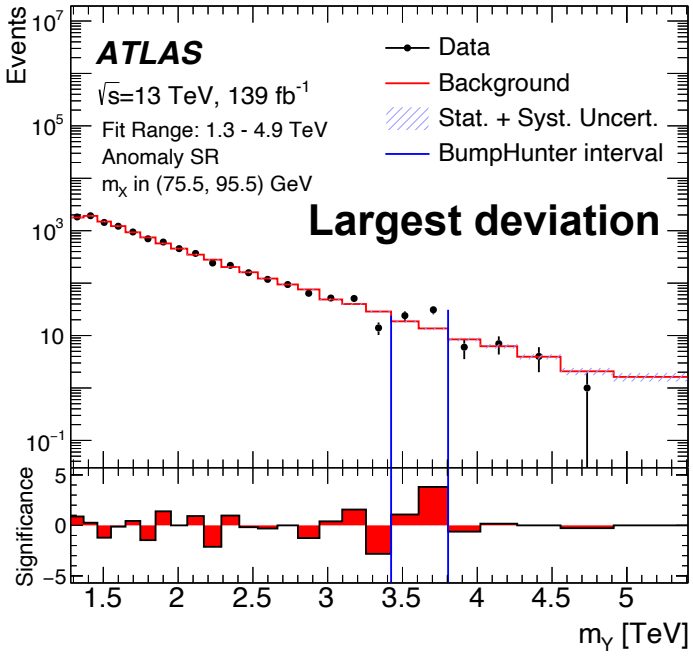






ATLAS PRD 108

LEE is large and not trivial to estimate; global p-value is 1.43σ



2-body Unsupervised Search

- Look for 2-body resonances in final states with at least one lepton (ℓ = e or μ) or photon, and at least one jet using various combinations, ℓj, γj, and jj, with some jets being b-tagged
- Fully unsupervised search using the rapidity-mass matrix (RMM) as the ML input
  - ★ The matrix is of a fixed size with kinematic information for 36 final state combinations; missing combinations are padded with zeroes; the energy/mass variables are normalized by 1/√s to make them dimensionless; all energies are ordered in strictly decreasing order
  - ★ The RMM elements are between 0 and 1 by construction; most are invariant under longitudinal boosts
  - ★ This matrix is shown to provide more robust training results than, e.g., 4-momenta of the individual objects
  - ★ An example of RMM for the case of jets and muons is shown below
- The chosen matrix size is 36 x 36; 9 mass variables are excluded to reduce biases, resulting in 1287 inputs fed into autoencoder, which contains two hidden layers of 800 and 400 neurons and the latent space of 200 neurons
- Autoencoder is trained on 1% of randomly selected collision events

$$\begin{pmatrix}
\mathbf{e_T^{miss}} & m_T(j_1) & m_T(j_2) & \dots m_T(j_N) & m_T(\mu_1) & m_T(\mu_2) & \dots m_T(\mu_N) \\
h_L(j_1) & \mathbf{e_T(j_1)} & m(j_1,j_2) & \dots m(j_1,j_N) & m(j_1,\mu_1) & m(j_1,\mu_2) & \dots m(j_1,\mu_N) \\
h_L(j_2) & h(j_1,j_2) & \delta\mathbf{e_T(j_2)} & \dots m(j_2,j_N) & m(j_2,\mu_1) & m(j_2,\mu_2) & \dots m(j_2,\mu_N) \\
\dots & \dots & \dots & \dots, & \dots & \dots & \dots \\
h_L(j_N) & h(j_1,j_N) & \dots & \dots \delta\mathbf{e_T(j_N)} & m(j_N,\mu_1) & m(j_N,\mu_2) & \dots m(j_N,\mu_N) \\
h_L(\mu_1) & h(\mu_1,j_1) & h(\mu_1,j_2) & \dots h(\mu_1,j_N) & \mathbf{e_T(\mu_1)} & m(\mu_1,\mu_2) & m(\mu_1,\mu_N) \\
h_L(\mu_2) & h(\mu_2,j_1) & h(\mu_1,j_2) & \dots h(\mu_2,j_N) & h(\mu_1,\mu_2) & \delta\mathbf{e_T(\mu_2)} & m(\mu_2,\mu_N) \\
\dots & \dots & \dots & \dots & \dots & \dots & \\
h_L(\mu_N) & h(\mu_N,j_1) & h(\mu_N,j_2) & \dots h(\mu_N,j_N) & h(\mu_N,\mu_1) & h(\mu_N,\mu_2) & \delta\mathbf{e_T(\mu_N)}
\end{pmatrix}$$

$$\delta e_T(i_n) = \frac{E_T(i_{n-1}) - E_T(i_n)}{E_T(i_{n-1}) + E_T(i_n)}, \quad n = 2, \dots, N,$$

$$h_L(i_n) = C(\cosh(y) - 1)$$

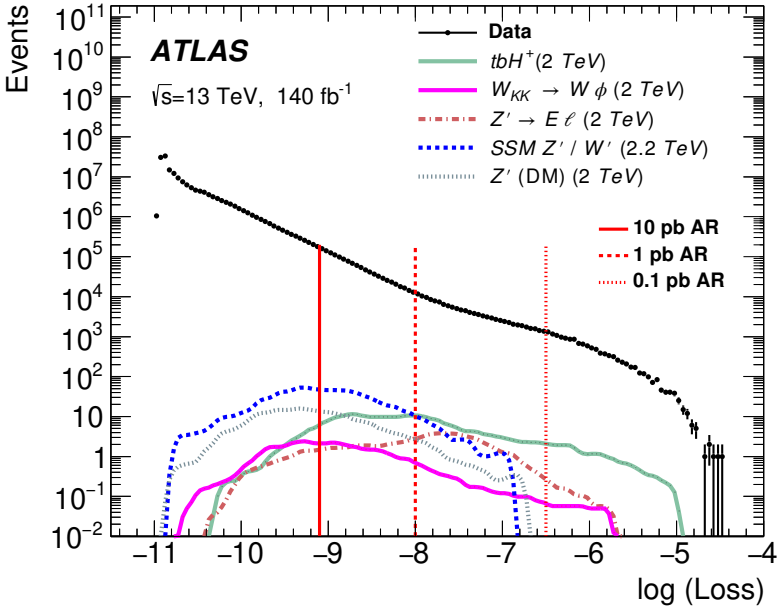
$$h(i_n, j_k) = C(\cosh(\Delta y/2) - 1)$$

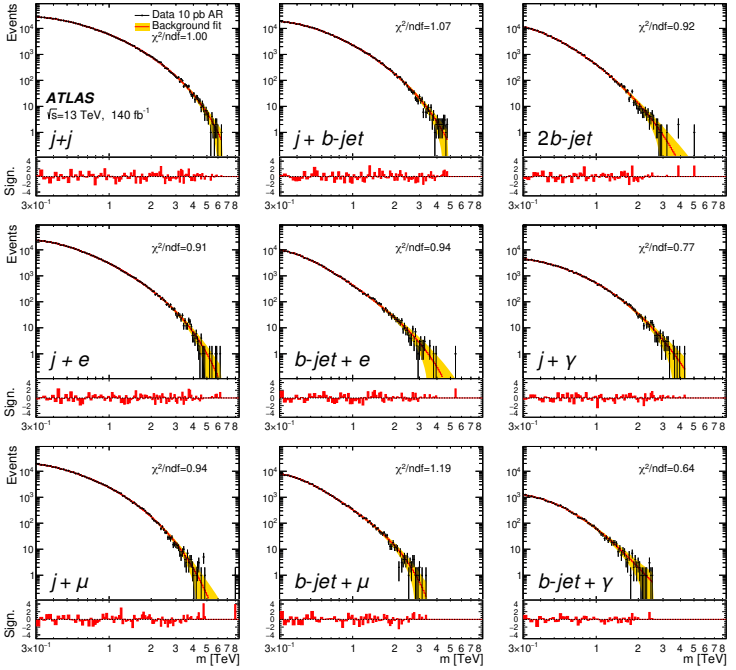
ATLAS arXiv:2307.01612

2-body Unsupervised Search

- The distribution of the anomaly score on the full data sample (~166M events), defined as a log of the reconstruction loss, which is calculated the mean squared error between the inputs and outputs
- Anomaly regions (ARs) are defined by the cross section of a hypothetical process, which would yield the same number of events as seen in data above a chosen threshold
- The ARs are used to reduce the background; the mass spectra of the object pairs in the selected events are then use for the bump search

ATLAS arXiv:2307.01612

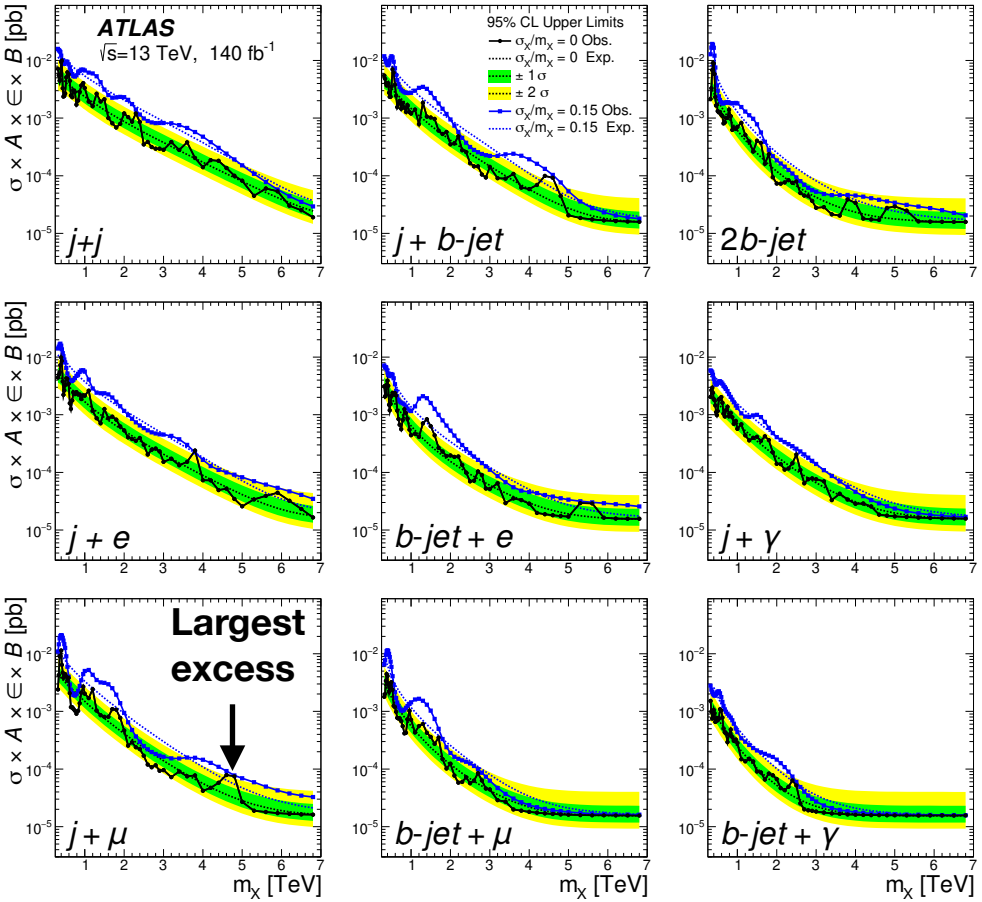




10 pb AR

2-body Unsupervised Search

- The limits on hypothetic signals are then set via standard means, which makes it straightforward to account for the LEE
- The largest excess for narrow resonance search has a local significance of 2.9σ and a global significance <2σ



ATLAS arXiv:2307.01612

Some References

- **Bibliography:**
  - ★ B. Nachman, "Anomaly Detection for Physics Analysis and Less than Supervised Learning", https://arxiv.org/pdf/2010.14554.pdf
  - ★ T. Plehn et al., "Modern Machine Learning for LHC Physics", https://arxiv.org/pdf/2211.01421.pdf
  - ★ V. Bellis et al., "Machine Learning for Anomaly Detection in Particle Physics", https://arxiv.org/pdf/2312.14190.pdf