Statistics

or “How to find answers to your questions”

Pietro Vischia[1]

[1] CP3 — IRMP, Université catholique de Louvain

UCLouvain
Institut de recherche
en mathématique et physique

CP3—IRMP, Intensive Course on Statistics for HEP, 07–11 December 2020

Program for today

UCLouvain
Institut de recherche
en mathématique et physique

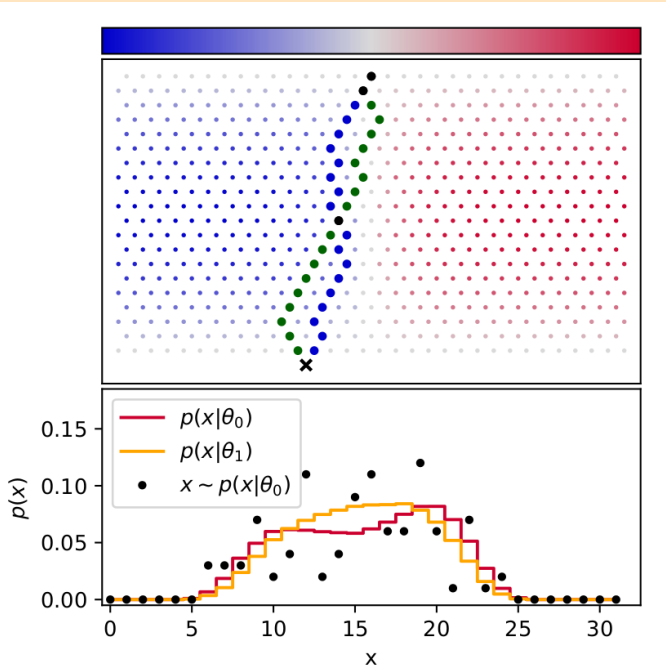
Machine Learning

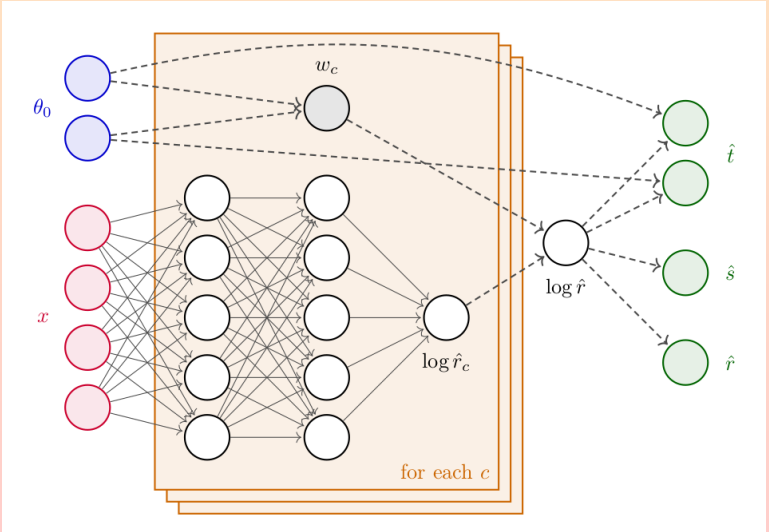
- Lesson 5
- Object ID
- Signal extraction
- What if you don’t know your signal?
- What about the uncertainties?
- Which data should we take?

- **Lesson 1 - Fundaments**
  - Bayesian and frequentist probability, theory of measure, correlation and causality, distributions
- **Lesson 2 - Point and Interval estimation**
  - Maximum likelihood methods, confidence intervals, most probable values, credible intervals
- **Lesson 3 - Advanced interval estimation, test of hypotheses**
  - Interval estimation near the physical boundary of a parameter
  - Frequentist and Bayesian tests, CLs, significance, look-elsewhere effect, reproducibility crysis
- **Lesson 4 - Commonly-used methods in particle physics**
  - Unfolding, ABCD, ABC, MCMC, estimating efficiencies
- **Lesson 5 - Machine Learning**
  - Overview and mathematical foundations, generalities most used algorithms, automatic Differentiation and Deep Learning

What if we don't have a likelihood?

- Likelihood $p(x|\theta) = \int dz p(x,z|\theta) = \int dz p_x(x|\theta,z) \prod_i p_i(z_i|\theta, z_{<i})$
  - Latent states sampled from $z_i \sim p_i(z_i|\theta, z_{<i})$
  - Final output sampled from $x \sim p_x(x|\theta, z)$
  - Observables x from particle generator; dependency on latent zs (matrix element, parton shower, detector...)
- Want to do inference in θ given a $p(x|\theta)$ which is intractable; likelihood trick;
  - Train a classifier (NN) to separate samples from $p(x|\theta_0)$ and $p(x|\theta_1)$
  - Likelihood ratio between θ_0 and θ_1 by inverting the minimization of the binary cross-entropy loss
- Joint score $t(x,z|\theta_0)$ and likelihood ratio $r(x,z|\theta_0,\theta_1)$ computable from simulated samples
  - Train parameterized estimators, then likelihood ratio is the minimum of loss function
  - Or local approximation, then the score is a sufficient statistic for inference
- Rewrite the EFT likelihood in a basis in which it is a mixture model
- Calculate the full true parton-level likelihood starting from N simulated events
  - Obtain a sufficient statistic for inference; exploit all available information!
  - Inference not limited anymore by the size of the generated samples





Images from arXiv:1805.12244 and arXiv:1805.00020

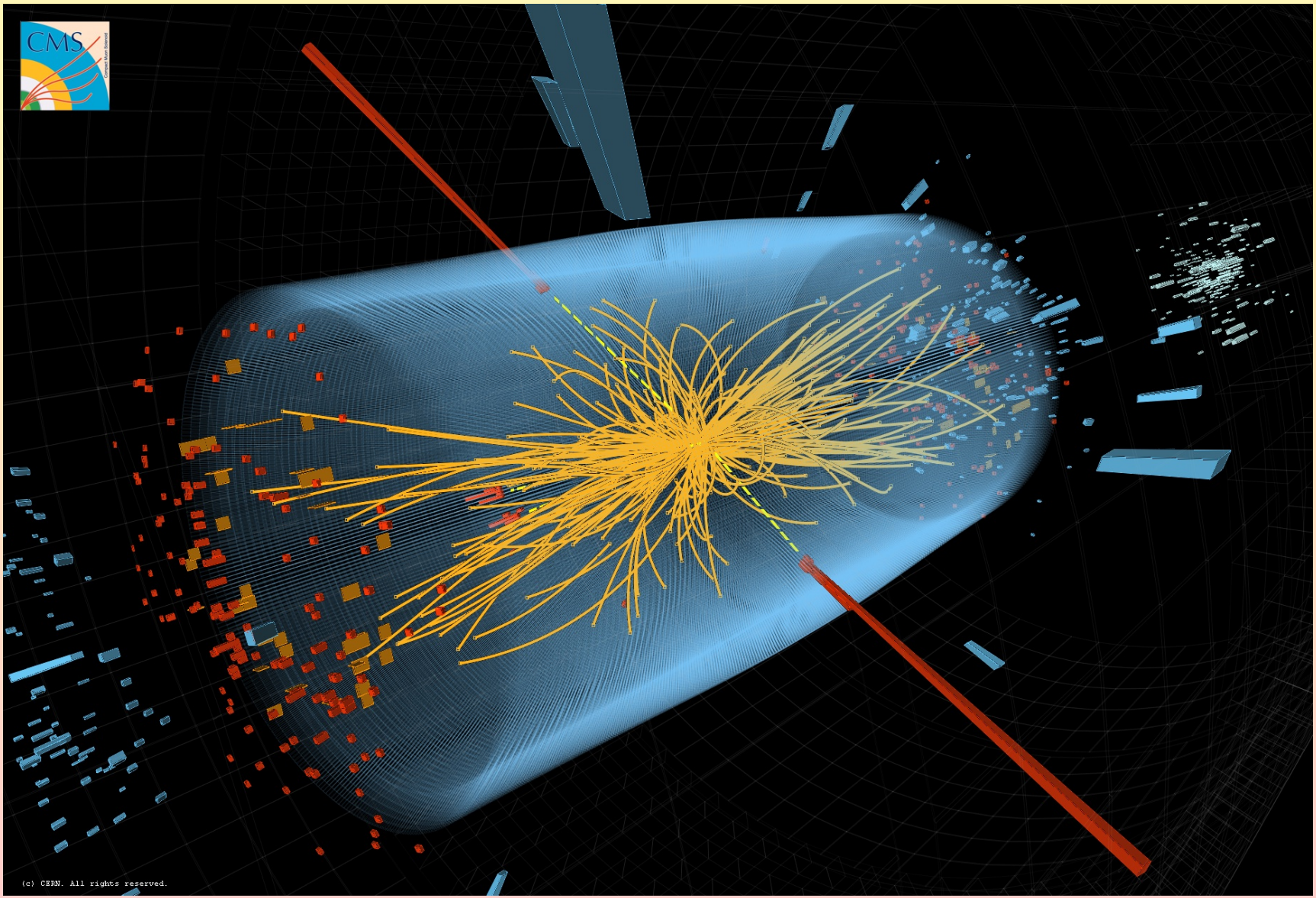
Machine learning

Vast amounts of data are being generated in many fields, and the statistician's job is to make sense of it all: to extract important patterns and trends, and understand "what the data says." We call this learning from data.

Hastie, Tibshirani, Friedman (Springer 2017)

We must efficiently collect and well reconstruct data

- ~ 40 MHz (millions per second) collision photos
  - Can store and reconstruct only a few of them



We must efficiently sample from our models

UCLouvain
Institut de recherche en mathématique et physique

$$P(\mathbf{x}|\alpha) = \frac{1}{A_\alpha \sigma_\alpha} \int \mathrm{d}\Phi(y) \frac{\mathrm{d}x_1 \mathrm{d}x_2}{x_1 x_2 s} f(x_1) f(x_2) |\mathcal{M}_\alpha(y, x_1, x_2)|^2 W(\mathbf{x}|y)\, \epsilon_\alpha(y)$$

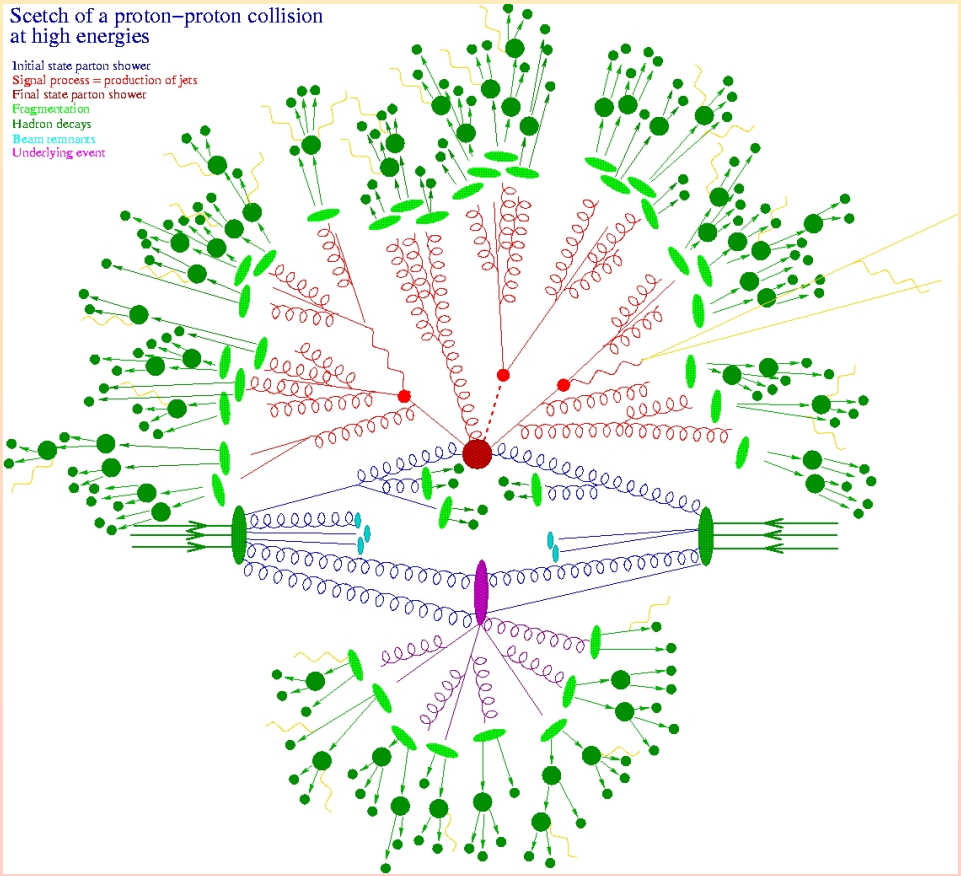
Normalization factor

We collide protons and that is a mess

Linear operator in Hamiltonian formalism, describes the physics process

Detector response, experimental efficiencies

- Costly MonteCarlo simulations, sampling from these high-dimensional probability density functions



We must improve the tools for detailed studies (e.g. EFT, differential)

- The Standard Model leaves some questions open
  - What is the origin of the Higgs mechanism? The Higgs field vacuum expectation (246 GeV) very far from Planck scale (quantum gravity): hierarchy problem
  - Origin of the observed neutrino masses? Most explanations of neutrino non-zero masses and mixing are beyond the SM
  - Dark Matter: a new, hidden sector of particles and forces?
  - Is the Higgs boson discovered in 2012 the Standard Model one?
- The study of Higgs boson physics is crucial for many of these topics
  - New scalar bosons (e.g. charged Higgs bosons) by simple extensions of the Higgs sector of the SM
  - **Slight deviations** from the expected properties of the observed Higgs boson could reveal signs for new physics

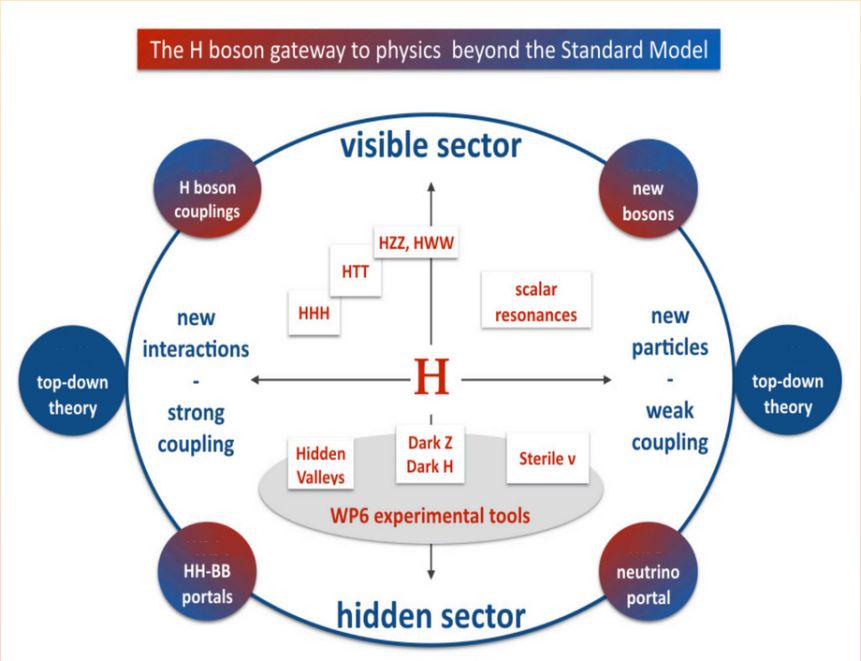
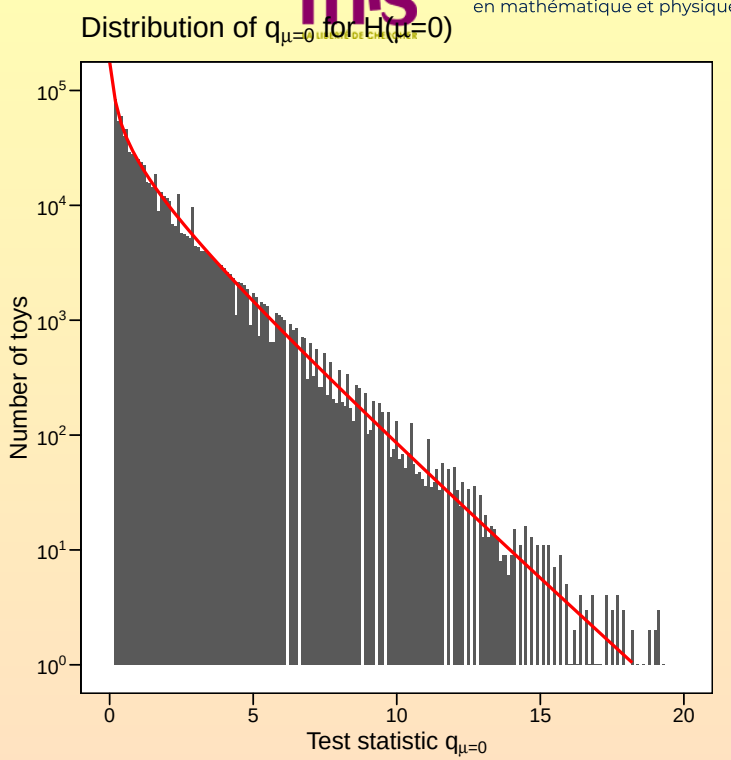


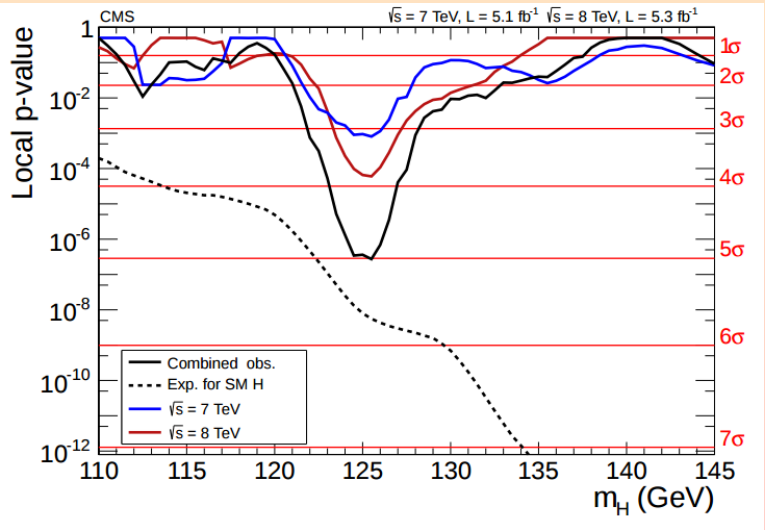
Image by the EOS be.h network

Ultimately, we must improve our inference: the end goal!

UCLouvain
Institut de recherche en mathématique et physique

- Statistical inference to make statements about parameters of our models
- New physics?
  - Probability of extreme fluctuation under the null measures significance of excess
  - Function of other parameters under investigation (e.g. Higgs boson mass in 2012)
- Systematic uncertainties induce variations in the number of events in the search region
  - We account for them in our statistical procedures at the hypothesis testing stage
- Often machine learning techniques are employed to optimize the analysis at early stages: **systematic uncertainties not accounted for in the optimization**





Images from Phys. Lett. B 716 (2012) 30 and P. Vischia, ***** (textbook to be published by Springer in 2021)

UCLouvain
Institut de recherche
en mathématique et physique

- I was told *"this is a black box, we cannot trust it for physics"*
  Comment by one of the researchers assisting to my final summer-student internship seminar
- There was still some diffidence towards machine learning algorithms

Trainee: Pietro Vischia **Year:** 2006

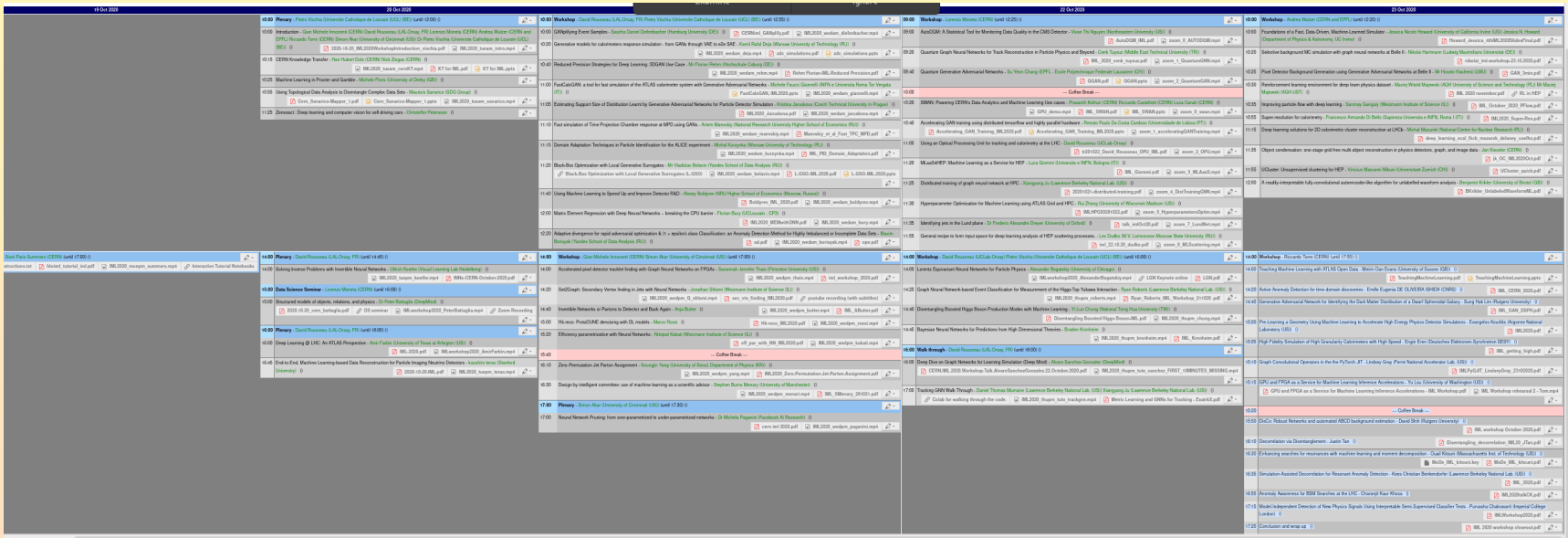
Mentor: Stephan Lammel

A) What did you learn while at Fermilab?

In my period as a Trainee I worked on b-jet energy corrections, thus deepening my knowledge on b-physics and data analysis methods.
In particular, I gained knowledge about neural networks and their possible use in physics.
I indeed used a neural network in the attempt to improve the b-jet energy resolution: this type of improvement is important for obtaining a better measurement of top quark mass and for having more chances of eventually discovering the Higgs boson at CDF.
The work started with gaining familiarity with basics concepts about neural networks, in particular multilayer perceptrons. This was accomplished by studying the software documentation and the work of some physicists who already used neural networks for data analysis. A preliminary work has been done by using some material sent by Brandon Parks (University of Ohio).
I then developed a network and applied it to our Z->b+anti-b and QCD b+anti-b signal. I obtained a substantial improvement of the resolution on b-jet energy by obtaining a scale factor which modifies the measured energy of the quark.

UCLouvain
Institut de recherche en mathématique et physique

- I co-organized the CERN IML workshop (October 19th–23rd, 2020)
  - 951 registered participants
  - 71 contributions



1. ML for data reduction : Application of Machine Learning to data reduction, reconstruction, building/tagging of intermediate object
2. ML for analysis : Application of Machine Learning to analysis, event classification and fundamental parameters inference
3. ML for simulation and surrogate model : Application of Machine Learning to simulation or other cases where it is deemed to replace an existing complex model
4. Fast ML : Application of Machine Learning to DAQ/Trigger/Real Time Analysis
5. ML algorithms : Machine Learning development across applications
6. ML infrastructure : Hardware and software for Machine Learning
7. ML training, courses and tutorials
8. ML open datasets and challenges
9. ML for astroparticle
10. ML for experimental particle physics
11. ML for phenomenology and theory
12. ML for particle accelerators
13. Other

- Let's formalize the concept of learning from data
- We'll look into the formalism mostly for supervised learning
- Fore more mathematical details, see arXiv:1712.04741 and Joan Bruna's lectures online

- $\mathcal{X}$: a high-dimensional input space
  - The challenges come from the high dimensionality!
- If all dimensions are real-valued, $\mathbb{R}^d$
  - For square images of side $\sqrt{d}$, $\mathcal{X} = \mathbb{R}^d$, $d \sim \mathcal{O}(10^6)$

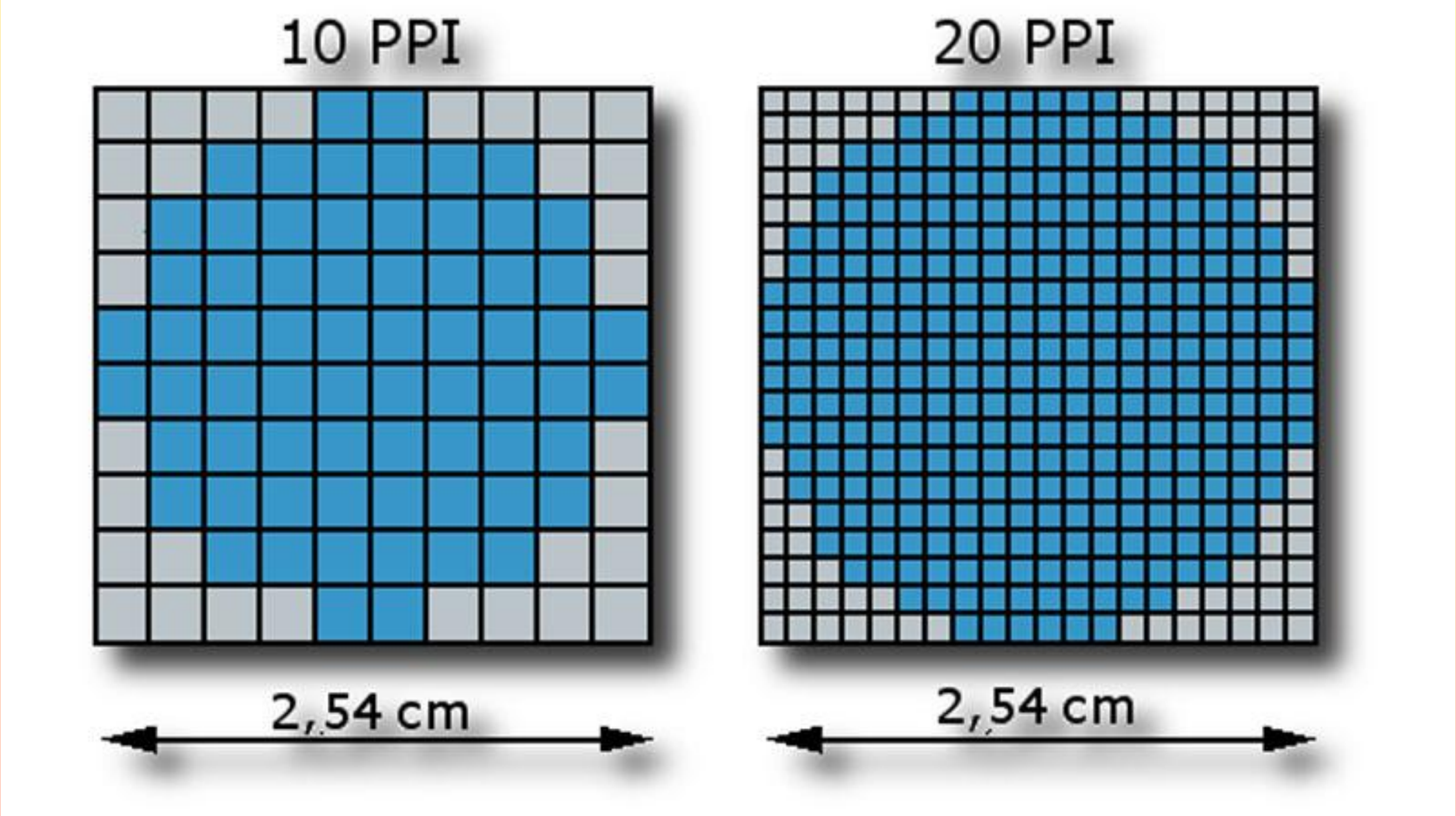


Figure frm scientiamobile.com

- ν: unknown data probability distribution
  - We can sample from it to obtain an arbitrary amount of data points
  - We are not allowed to use any analytic information about it in our computations

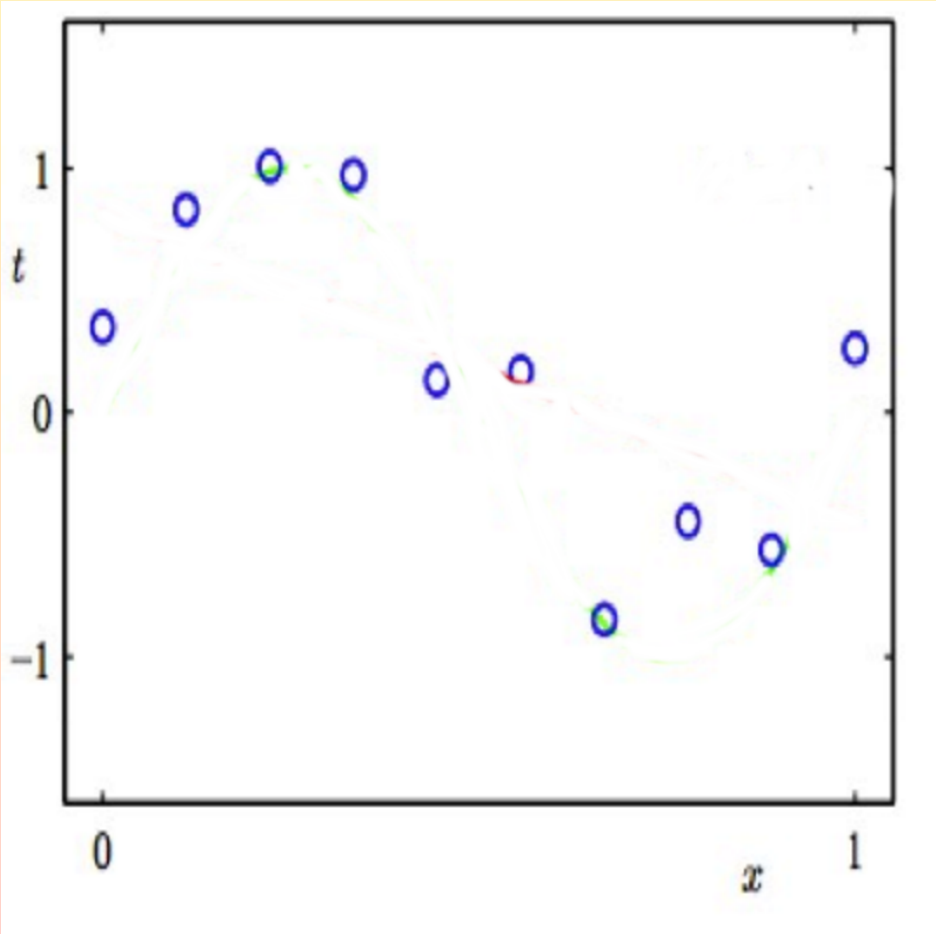
- $f^*: \mathcal{X} \to \mathbb{R}$, unknown target function
  - In case of multidimensional output to a vector of dimension k, $f^*: \mathcal{X} \to \mathbb{R}^k$
  - Some loose assumptions (e.g. square-integrable with respect to the ν measure, i.e. finite moments, bounded...)

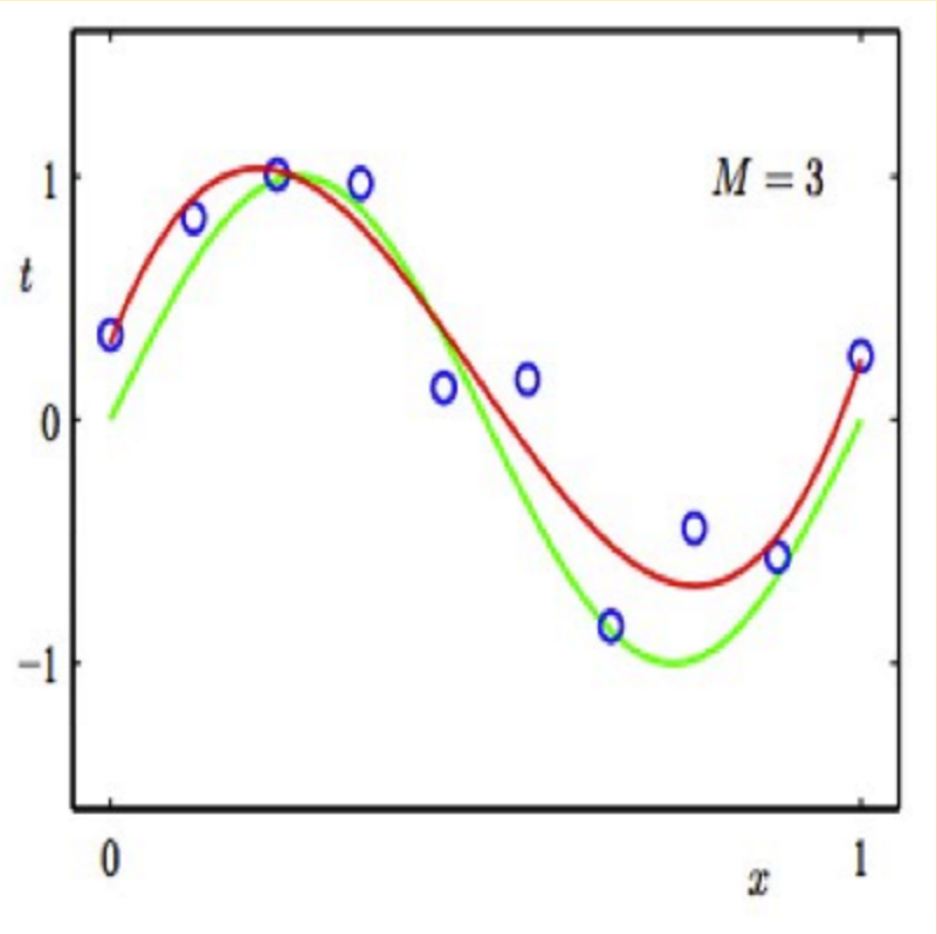
How do we learn? The loss functional

- $L[f] = \mathbb{E}_\nu \left[l\Big(f(x), f^*(x)\Big) \right]$
  - The metric that tells us how good our predictions are
- The function $l(\cdot, \cdot)$ is a given expression, e.g. regression loss, logistic loss, etc
  - In this lecture, typically it is the L^2 norm: $\|f - f^*\|_{L^2(\mathcal{X}, \nu)}$

Learning goal

- Goal: predict f^* from a finite i.i.d. sample of points sampled from ν
- Sample: $\left\{x_i, f^*(x_i)\right\}_{i=1,\ldots,n}$, $x_i \sim \nu$
  - For each of the points x_i, we know the value of the unknown function (our true labels)
  - We want to *interpolate* for any arbitrary x inbetween the labelled x_i...
  - ...in million of dimensions!

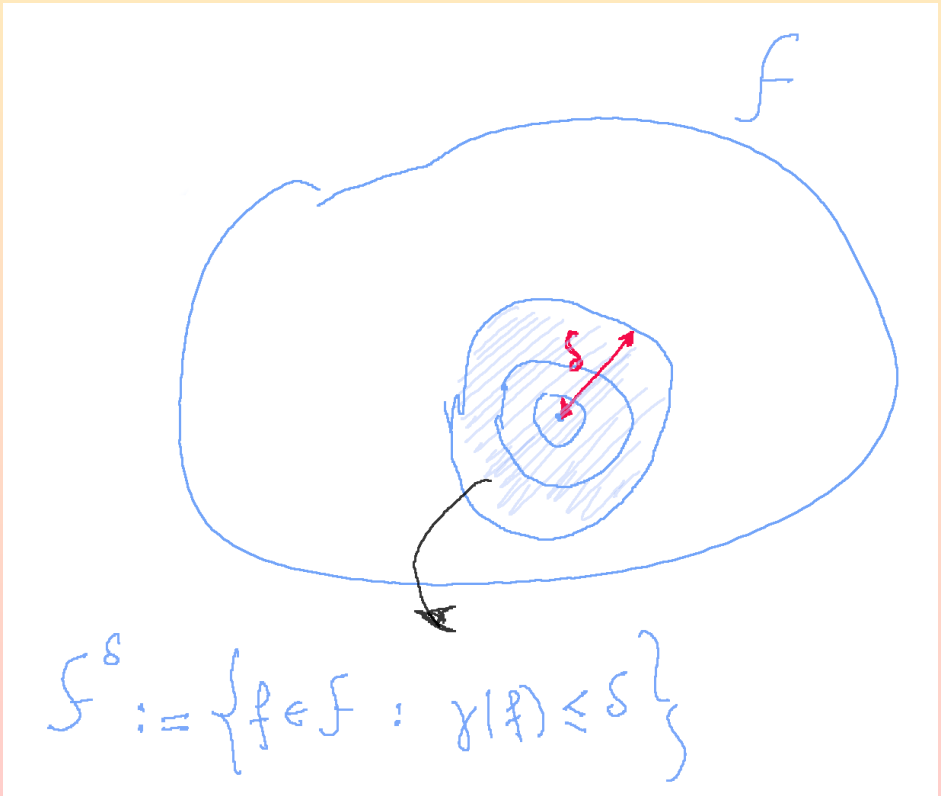




Images by Victor Lavrenko

The space of possible solutions

- The space of functionals that can potentially solve the problem is vast: $\mathcal{F} \subseteq \left\{ f : \mathcal{X} \to \mathbb{R} \right\}$ (hypothesis class)
- We need a notion of complexity to "organize" the space
- $\gamma(f), f \in \mathcal{F}$: *complexity* of f
  - It can for example be the norm, i.e. we can augment the space $\mathcal{F}$ with the norm
- When the complexity is defined via the norm, $\mathcal{F}$ is highly organized: Banach space!
  - The simplest function according to the norm criterion is the 0 function
  - If we increase the complexity by increasing the norm, we obtain **convex balls** $\left\{ f \in \mathcal{F}; \gamma(f) \leq \delta \right\} =: \mathcal{F}^{\delta}$
  - Convex minimization is considerably easier than non-convex minimization



- For each element of $\mathcal{F}$, a measure of how well it's interpolating the data
- Empirical risk: $\hat{L}(f) = \frac{1}{n} \sum_{i=1}^{n} |f(x_i) - f^*(x_i)|^2$
  - $|\cdot|$ is the empirical loss. If it's the norm, then $\hat{L}(f)$ is the empirical Mean Square Error
  - If you find an analogy with least squares method, it's because for one variable it's exactly that!

- **Constraint form:** $\min_{f \in \mathcal{F}^\delta} \hat{L}(f)$.
  - Not trivial
- **Penalized form:** $\min_{f \in \mathcal{F}} \hat{L}(f) + \lambda\gamma(f)$.
  - More typical
  - λ is the price to pay for more complex solutions. Depends on the complexity measure
- **Interpolant form:** $\min_{f \in \mathcal{F}} \gamma(f)$ s.t. $\hat{L}(f) = 0 \iff f(x_i) = f^*(x_i) \qquad \forall i$
  - In ML, most of the times there is no noise, so $f(x_i)$ is exactly the value we expect there (i.e. we really know that x_i is of a given class, without any uncertainty)
  - The interpolant form exploits this (*"give me the least complex elements in $\mathcal{F}$ that interpolates"*)

- These forms are not completely equivalent. The penalized form to be solved requires averaging a full set of penalized forms, so it's not completely equivalent
- There is certainly an implicit correspondence between δ and λ (the larger λ, the smaller δ and viceversa)

- We want to relate the result of the empirical risk minimization (ERM) with the prediction
  - Let's use the constraint form
- Let's assume we have solved the ERM at a precision ϵ (we are ϵ-away from...)
  we then have $\hat{f} \in \mathcal{F}^{\delta}$ such that $\hat{L}(\hat{f}) \leq \epsilon + \min_{f \in \mathcal{F}^{\delta}} \hat{L}(f)$
- How good is $\hat{f}$ at predicting f^*? In other words, what's the true loss?
  - Can use the triangular inequality

$$L(\hat{f}) - \inf_{f \in \mathcal{F}} L(f) \leq \inf_{f \in \mathcal{F}^{\delta}} L(f) - \inf_{f \in \mathcal{F}} L(f)$$

Approximation error

(how appropriate is my measure of complexity)

$$+2 \sup_{f \in \mathcal{F}^{\delta}} \left| L(f) - \hat{L}(f) \right|$$

Statistical error

(impact of having the empirical loss instead of the true loss)

$$+\epsilon$$

Optimization error

- The minimization is regulated by the parameter δ (the size of the ball in the space of functions)
- Changing δ results in a **tradeoff** between the different errors
  - Very small δ makes the statistical error blow up
- We are better at doing convex optimization (easier to find minimum), but even then the optimization error ϵ will not be negligible
  - ϵ: how much are ou willing to spend in resources to minimize $\hat{L}(f)$
  - We kind of control it!
  - If the other errors are smaller than ϵ, then it makes sense to spend resources to decrease it
  - Otherwise, don't bother

Bottou and Bousquet, 2008, Shalev-Shwartz, Ben-David

- **Approximation:** we want to design "good" spaces $\mathcal{F}$ to approximate f^* in high-dimension
  - Rather profound problem, on which we still struggle
- **Optimization:** how to design algorithms to solve the ERM in general
  - We essentally have ONE answer: Question Time: The Optimization Problem

- **Approximation:** we want to design "good" spaces $\mathcal{F}$ to approximate f^* in high-dimension
  - Rather profound problem, on which we still struggle
- **Optimization:** how to design algorithms to solve the ERM in general
  - We essentally have ONE answer: Question Time: The Optimization Problem
  - Gradient Descent!

- How many samples do we need to estimate f^* depending on assumptions on its regularity?
- Question time: Curse of Dimensionality

- How many samples do we need to estimate f^* depending on assumptions on its regularity?
- Question time: Curse of Dimensionality
- f^* constant $\rightarrow$ 1 sample
- f^* linear $\rightarrow d$ samples
  - Space of functionals is $\mathcal{F} = \left\{ f : \mathbb{R}^d \rightarrow \mathbb{R}; f(x) = < x, \theta > \right\} \simeq \mathbb{R}^d$ (isomorphic)
  - It's essentially like solving a system of linear equations for the linear form $< x_i, \theta^* >$
  - d equations, d degrees of freedom
- The reason why it's so easy is that linear functions are regular at a global level
  - Knowing the function locally tells us automatically the properties everywhere

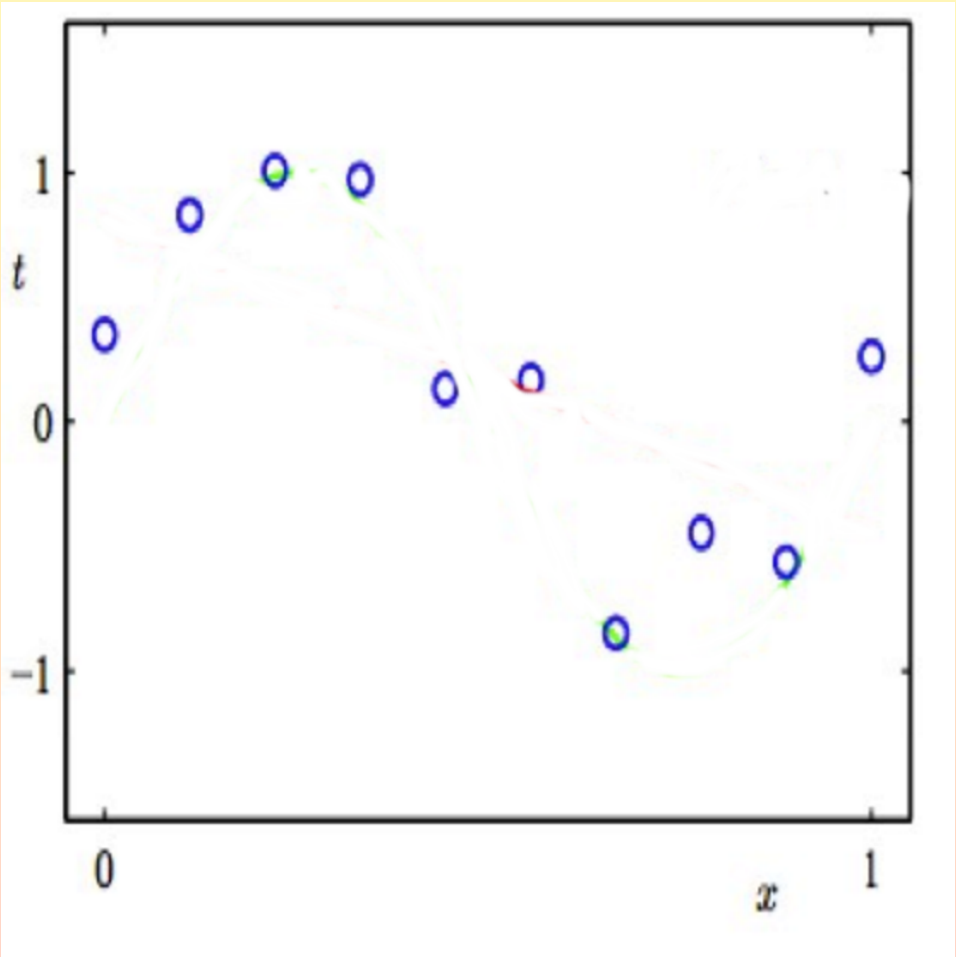
- f^* locally linear, i.e. f^* is Lipschitz
  - $|f^*(x) - f^*(y)| \leq \beta \|x - y\|$
  - $Lip(f^*) = inf\Big\{\beta; |f^*(x) - f^*(y)| \leq \beta \|x - y\| \text{ is true}\Big\}$
  - Lip(f^*) is a measure of smoothness
- Space of functionals that are Lipschitz: $\mathcal{F} = \Big\{f : \mathbb{R}^d \to \mathbb{R}; f \text{ is Lipschitz}\Big\}$
- We want a normed space to parameterize complexity, so we convert to a Banach space
  - $\gamma(f) := max(Lip(f), |f|_\infty)$
  - The parameterization of complexity **is** the Lipschitz constant

- $\forall \epsilon > 0$, find $f \in \mathcal{F}$ such that $||f - f^*|| \leq \epsilon$ from n i.i.d. samples
  - n: sample complexity, "how many more samples to I need to make the error a given amount of times smaller"
- If f^* is Lipschitz, it can be demonstrated that $n \sim \epsilon^{-d}$
  - Upper bound: approximate f with its value in the closest of the sampled data points, find out expected error $\sim \epsilon^2$, upper bound is exponential
  - Lower bound: maximum discrepancy (the worst case scenario): unless you sample exponential number of data points, knowing $f(x_i)$ for all of them doesn't let youwell approximate outside

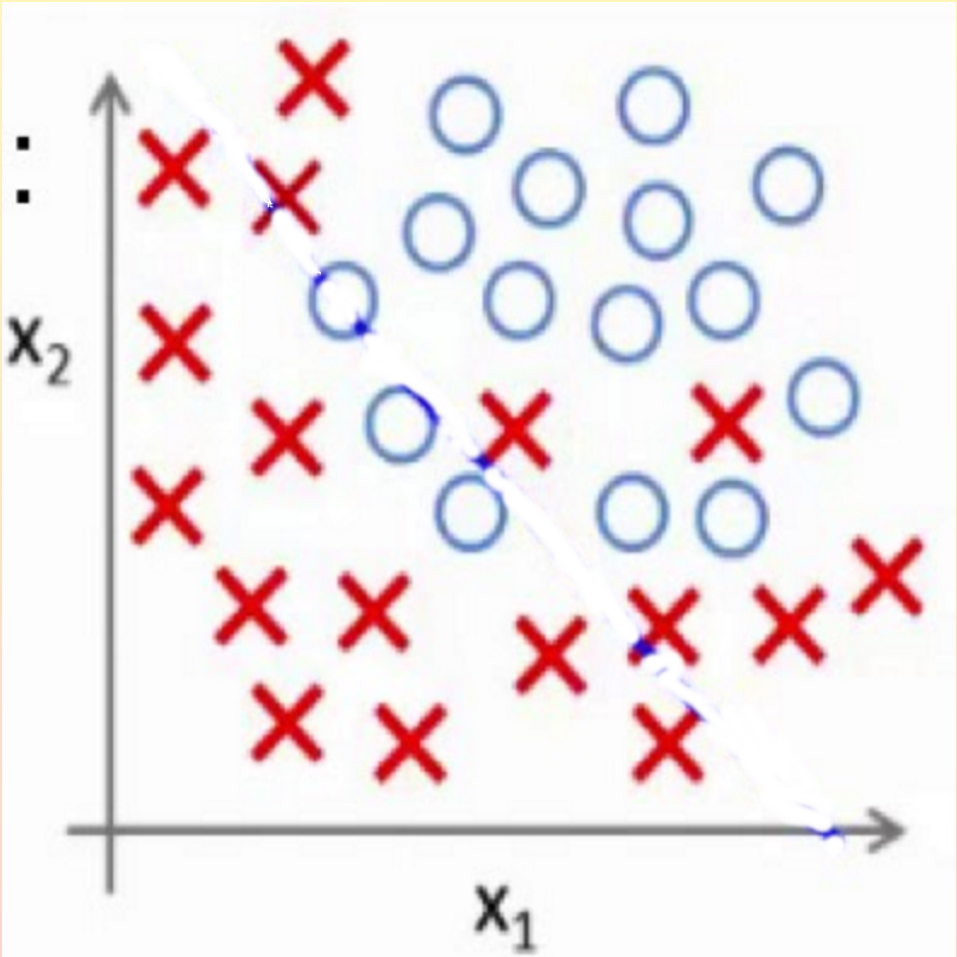
Enough of the math?

What's the best function

To describe the data points?
(regression)



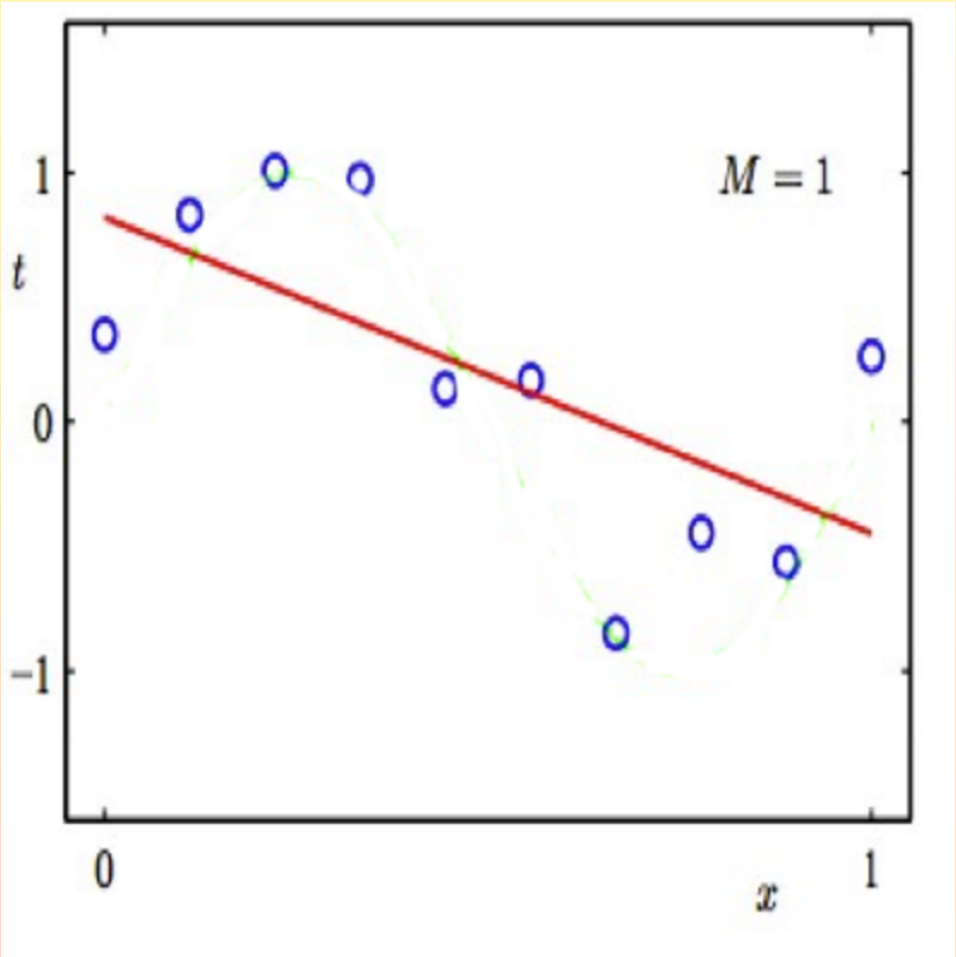
To separate into two classes?
(classification)



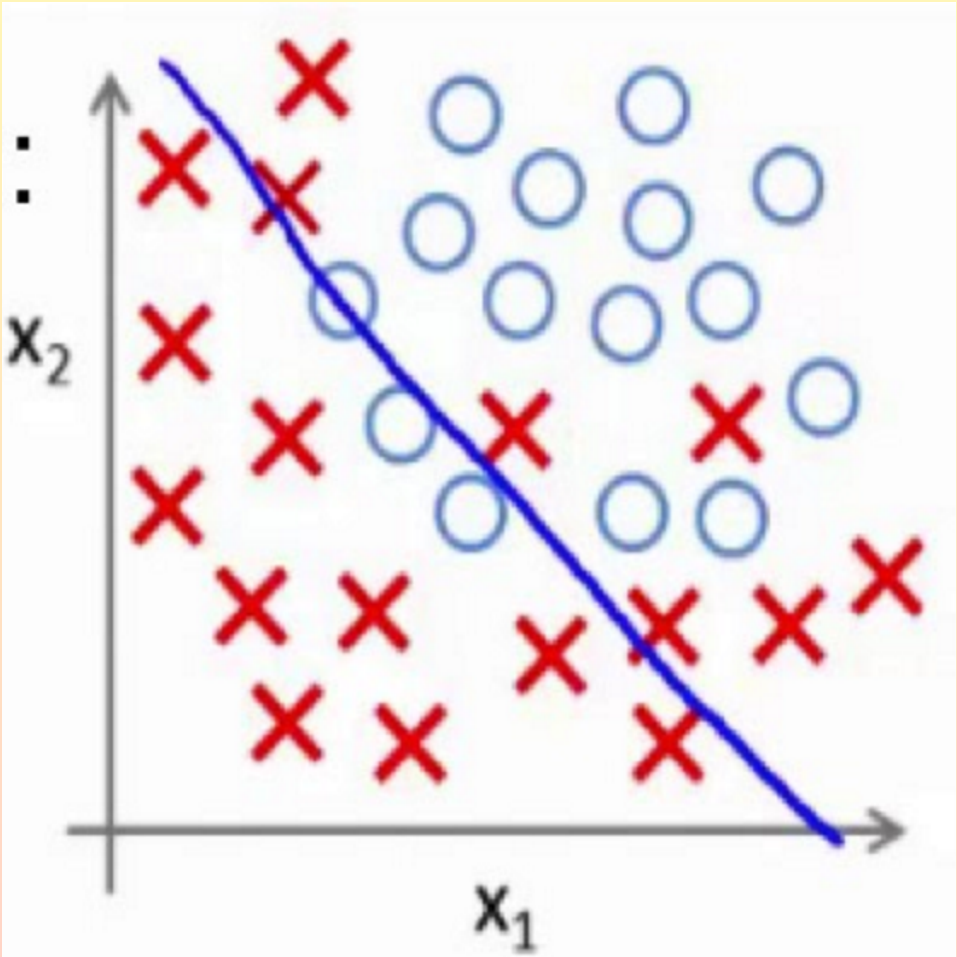
Images by Victor Lavrenko

What's the best function

To describe the data points?
(regression)



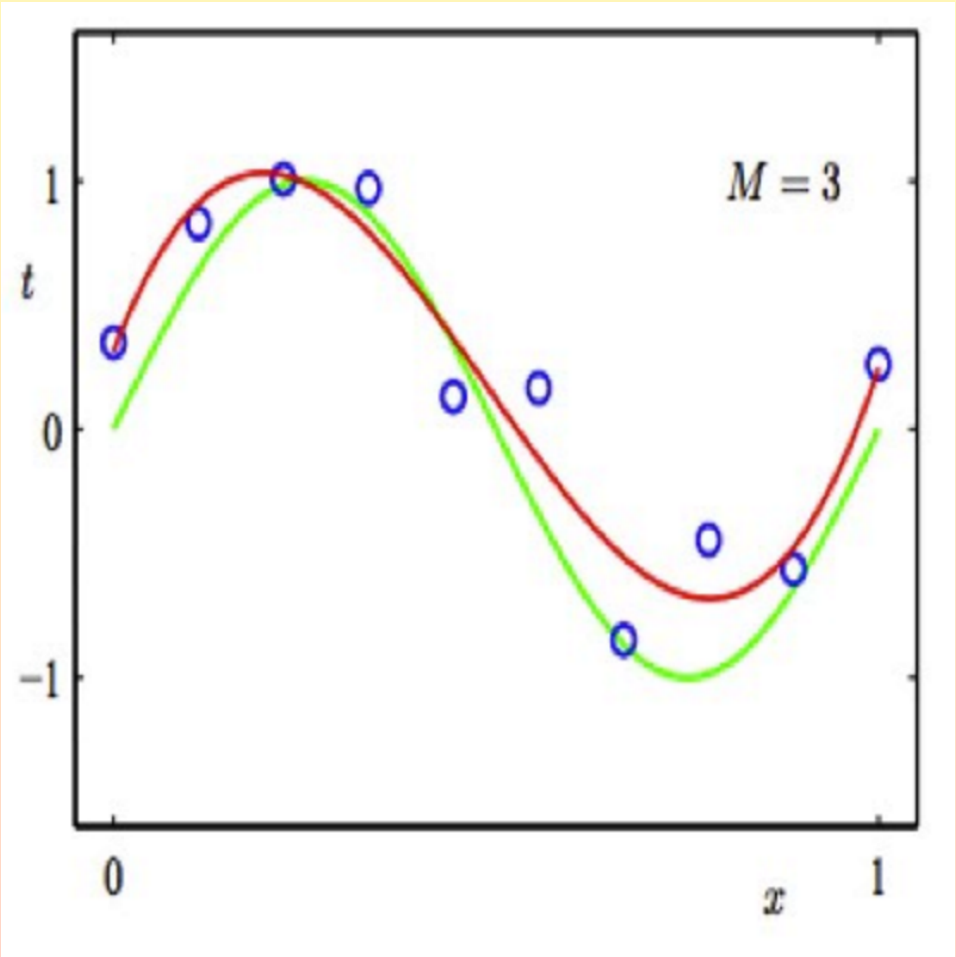
To separate into two classes?
(classification)



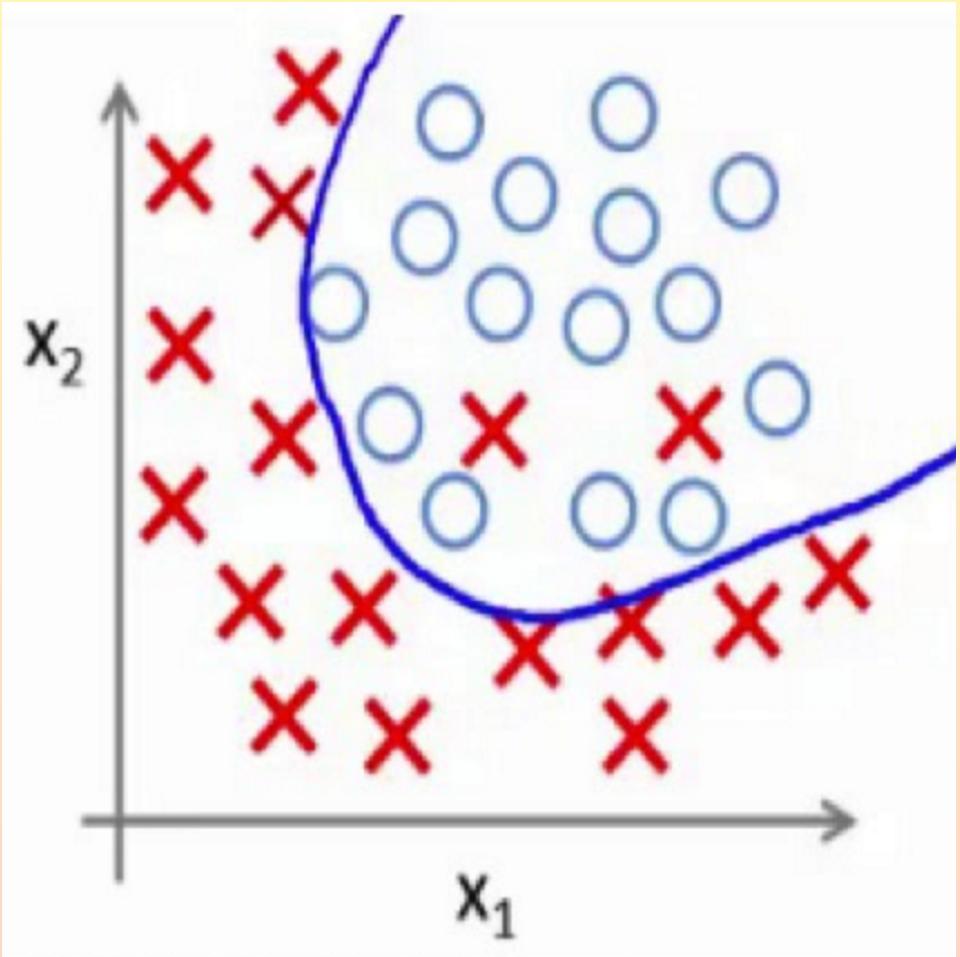
Images by Victor Lavrenko

What's the best function

To describe the data points?
(regression)



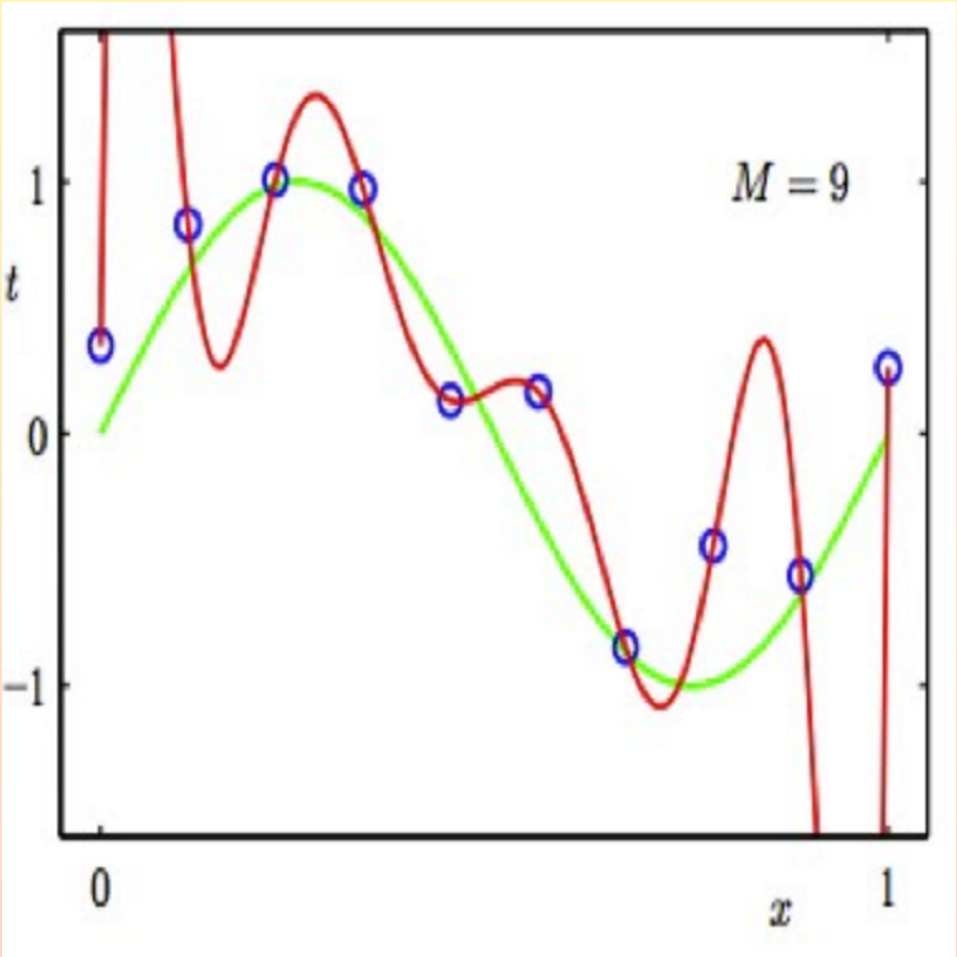
To separate into two classes?
(classification)



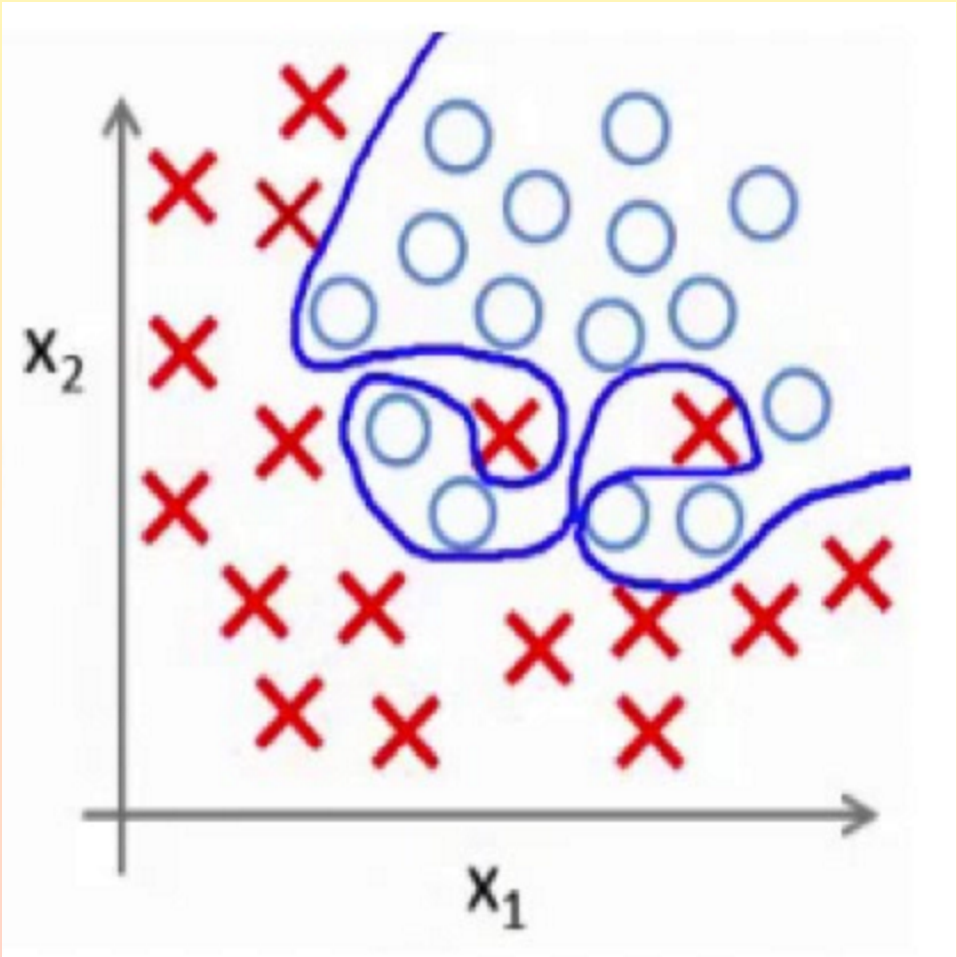
Images by Victor Lavrenko

What's the best function

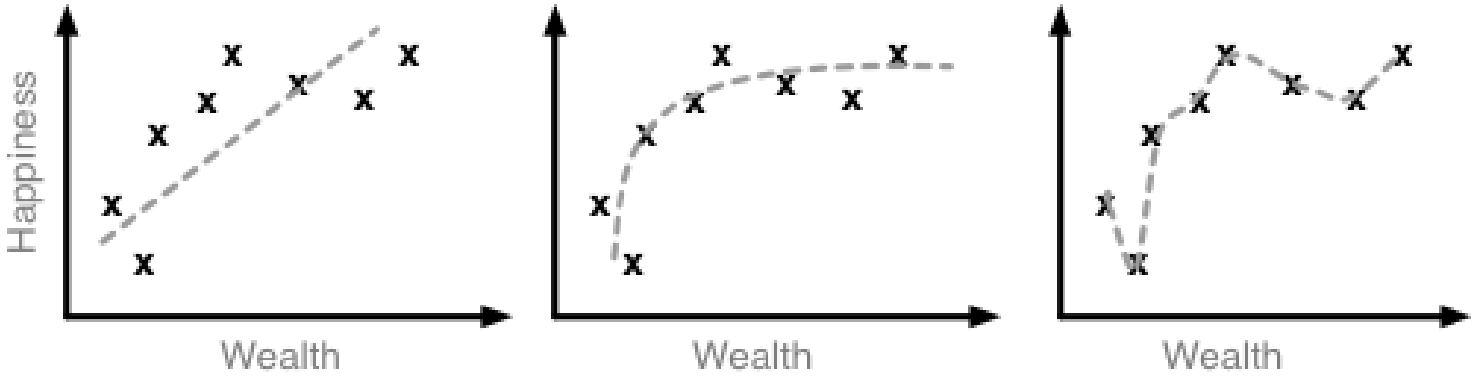
To describe the data points?
(regression)

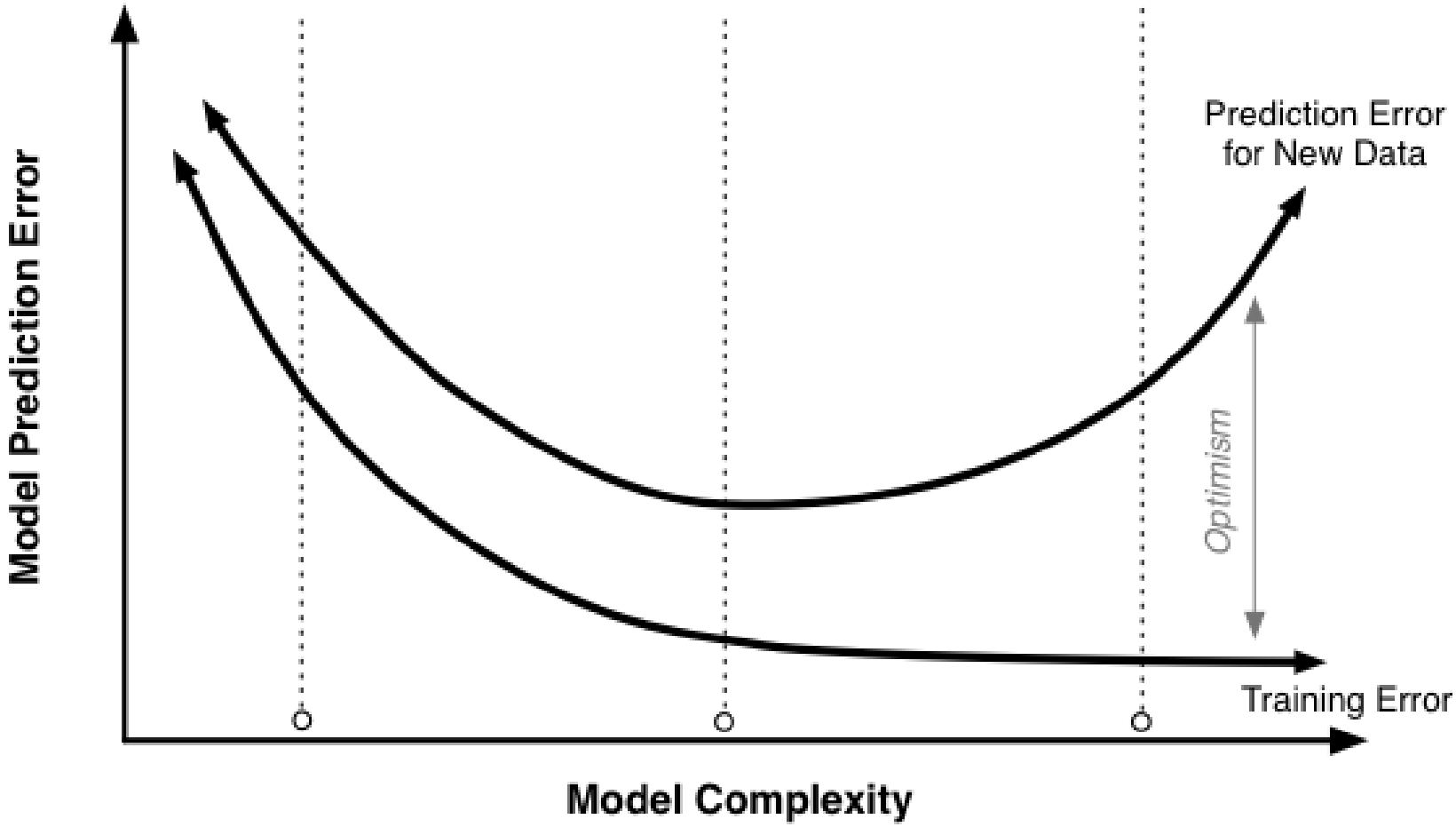


To separate into two classes?
(classification)

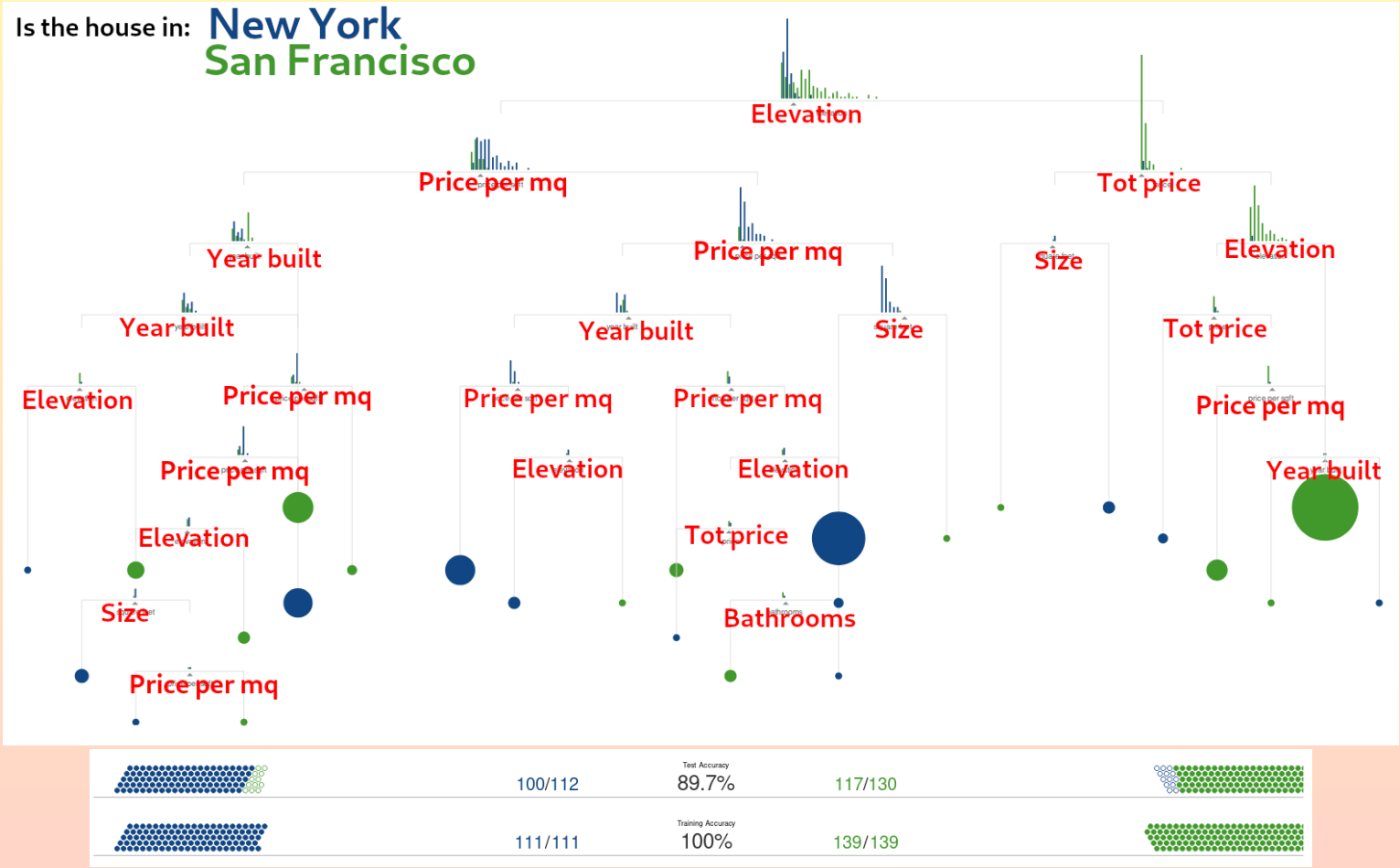


Images by Victor Lavrenko


Happiness
Wealth
Wealth
Wealth


Model Prediction Error
Prediction Error
for New Data
Optimism
Training Error
Model Complexity

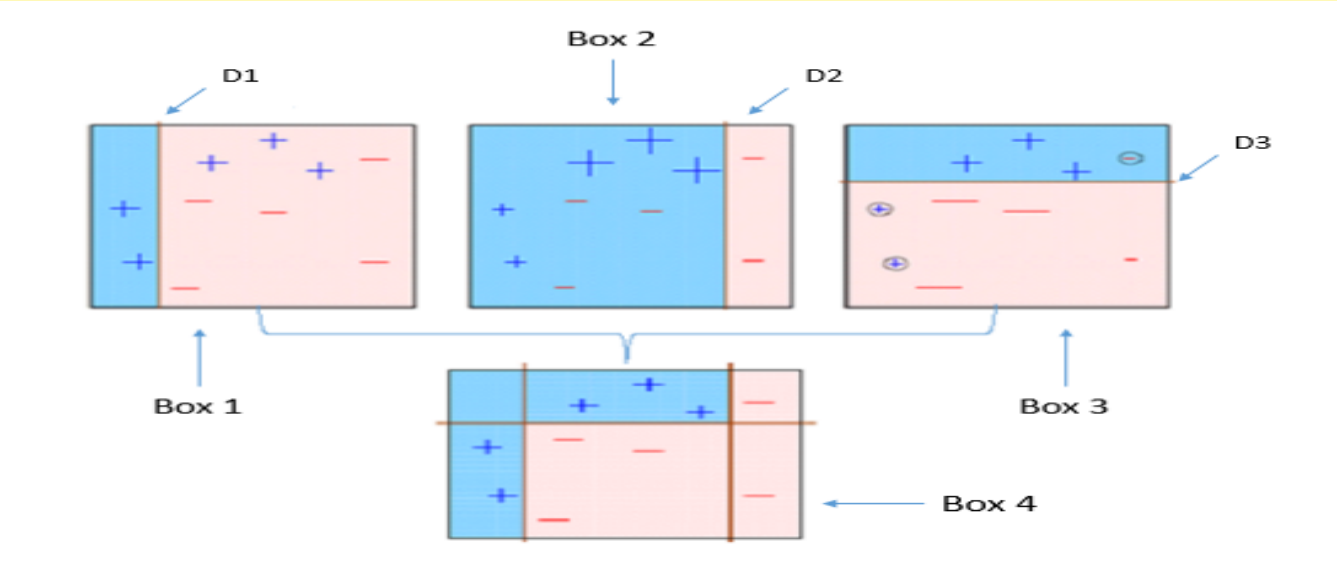
- Rather simple technique inspired by the standard approach of classifying events by selecting thresholds on several variables



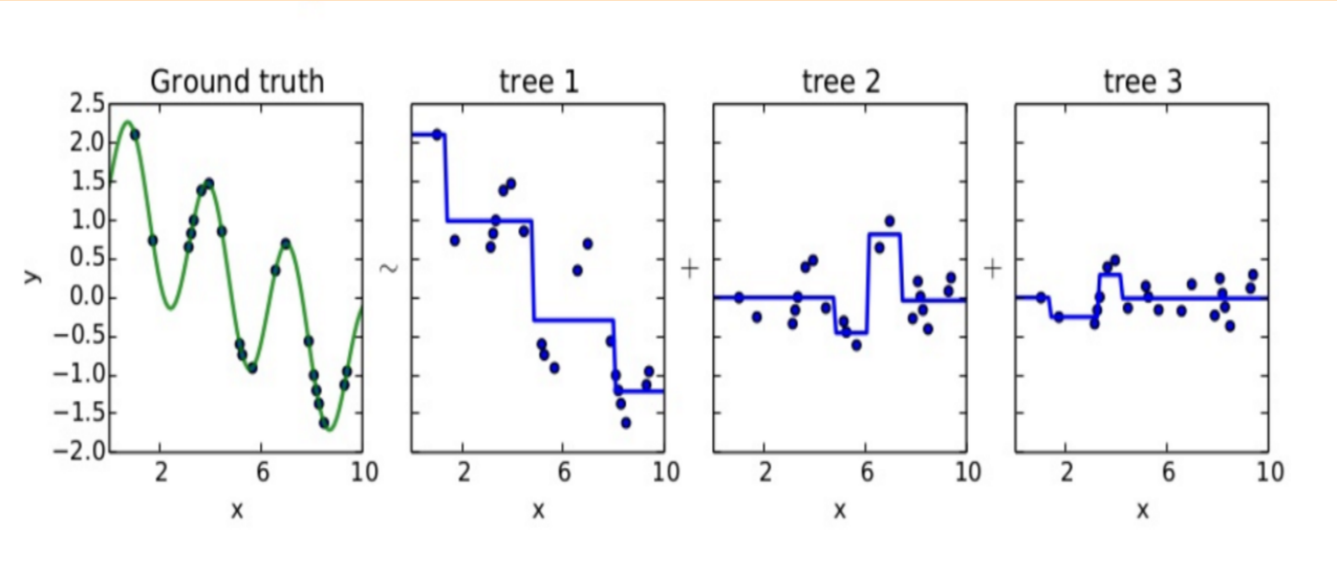
From http://www.r2d3.us/visual-intro-to-machine-learning-part-1/

Boosted decision trees)

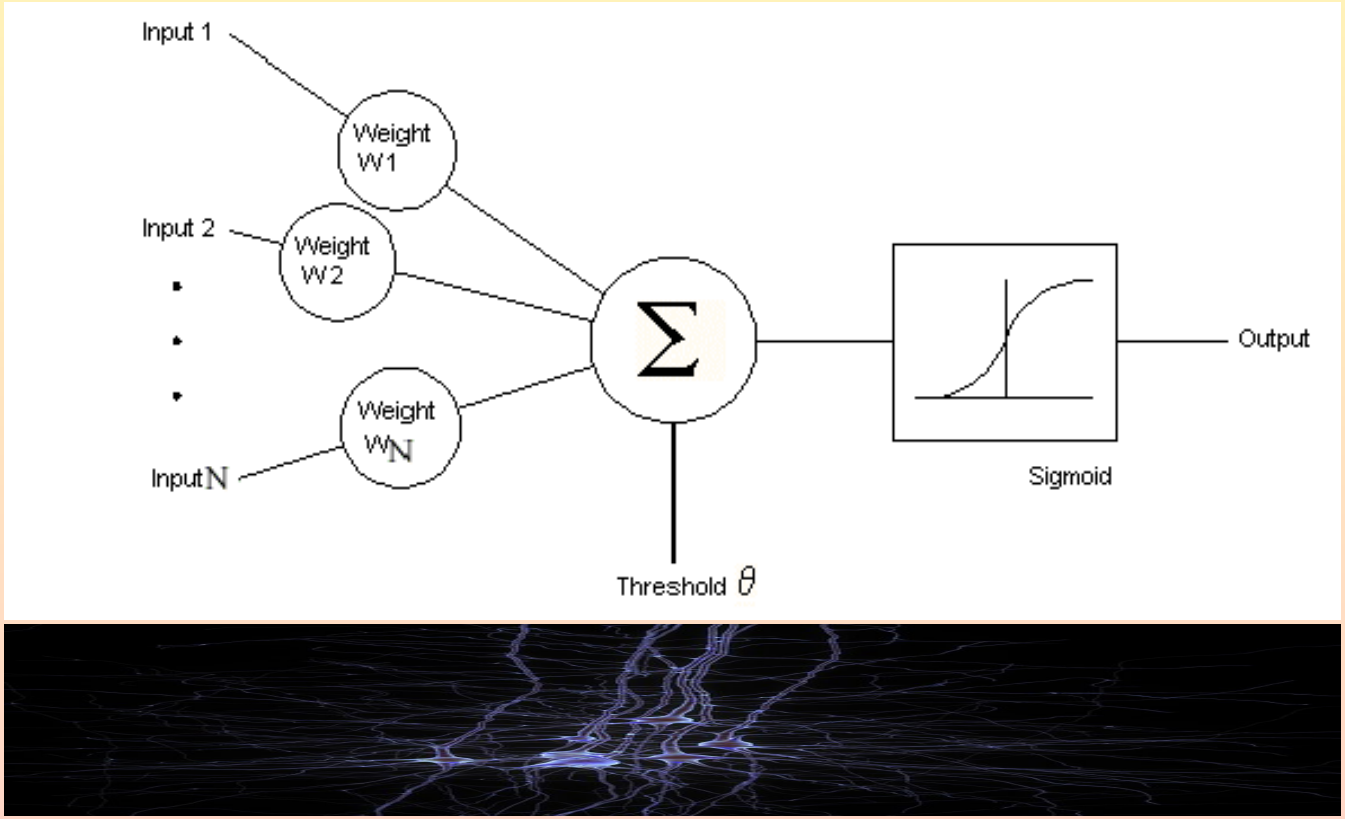
- Ada(ptive)Boost: increase at each iteration the importance of events incorrectly classified in the previous iteration



- GradientBoost: fit the new predictor to the residual errors of the previous one



- Perceptron: simplest mathematical model of a neuron
  - Activation function provides nonlinearity in the response
  - A network of these can demonstrably approximate any (insert loose conditions here) function



From http://homepages.gold.ac.uk/nikolaev/perceptr.gif and https://i.pinimg.com/originals/e3/fa/f5/e3faf5e2a977f98db1aa0b191fc1030f.jpg

Neural networks...

- Connecting neurons into a network
- Fully-connected networks: the most common a few decades ago
- Each weight is a free parameter that must be determined during the “training”

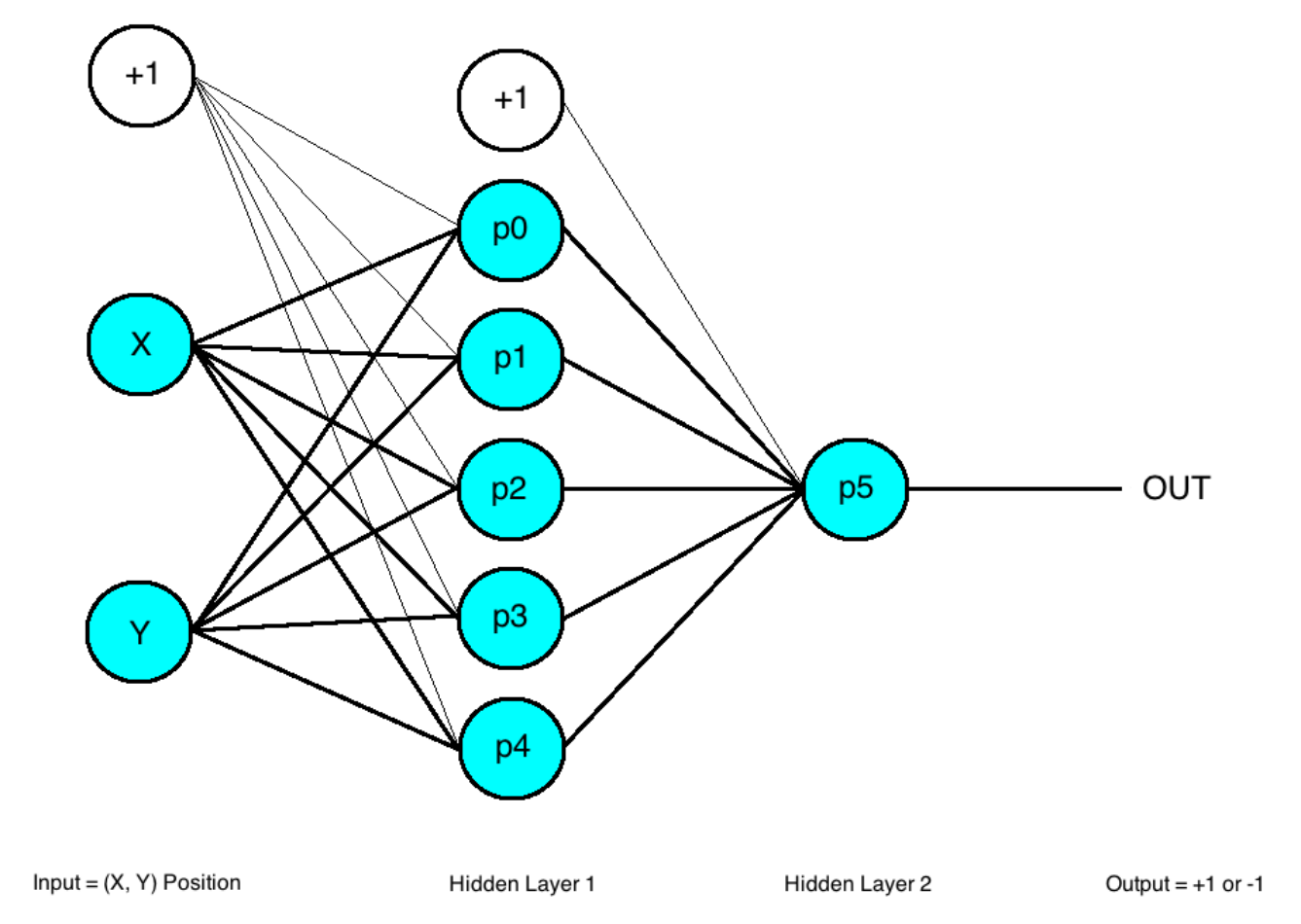


Image https://www.cs.utexas.edu/ teammco/misc/mlp/mlp.png

... and how to train them

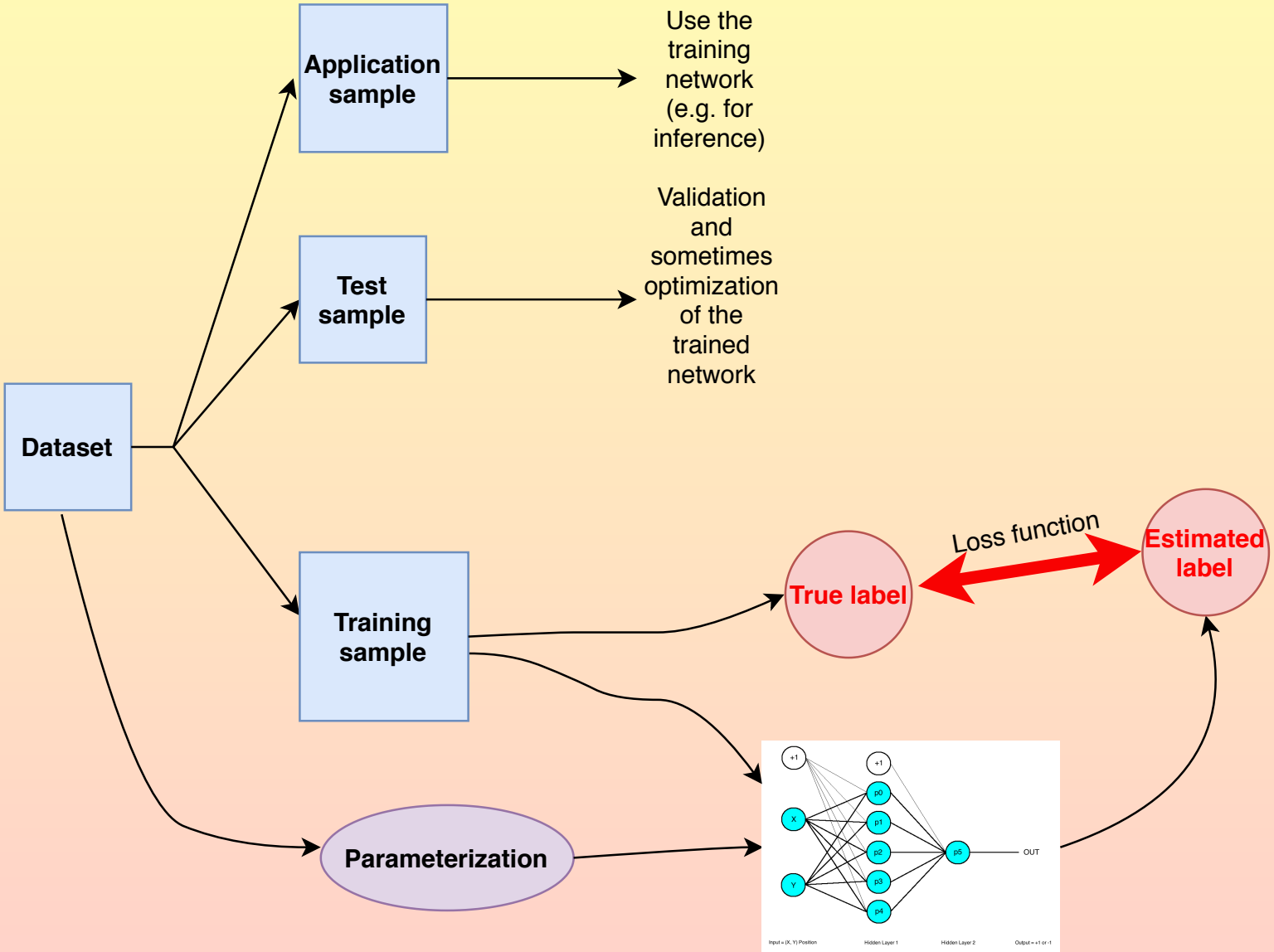
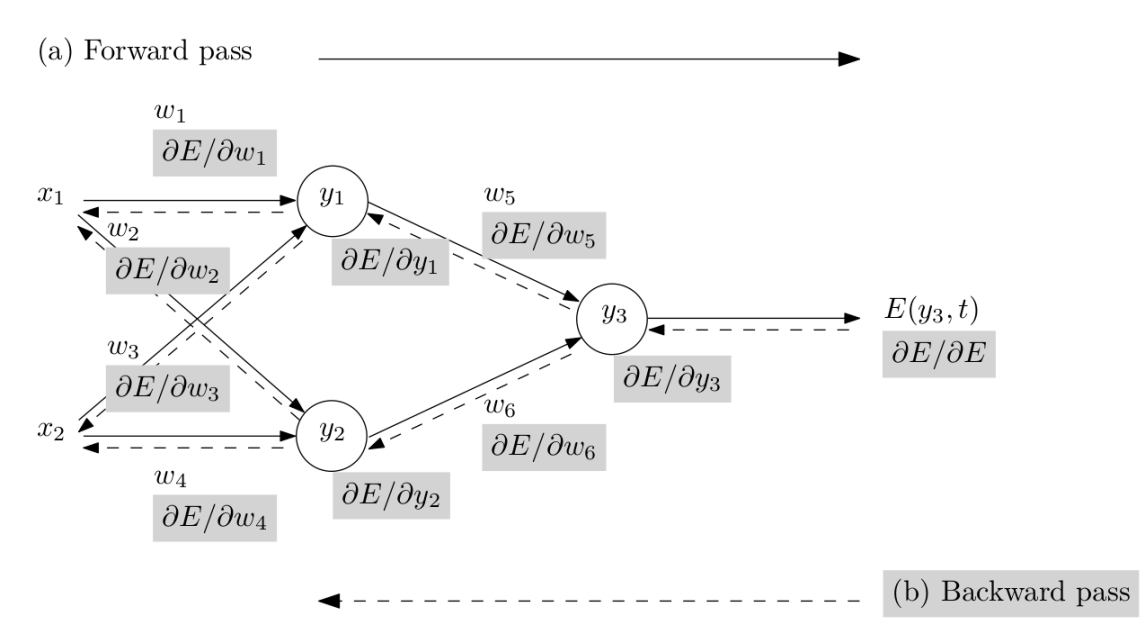
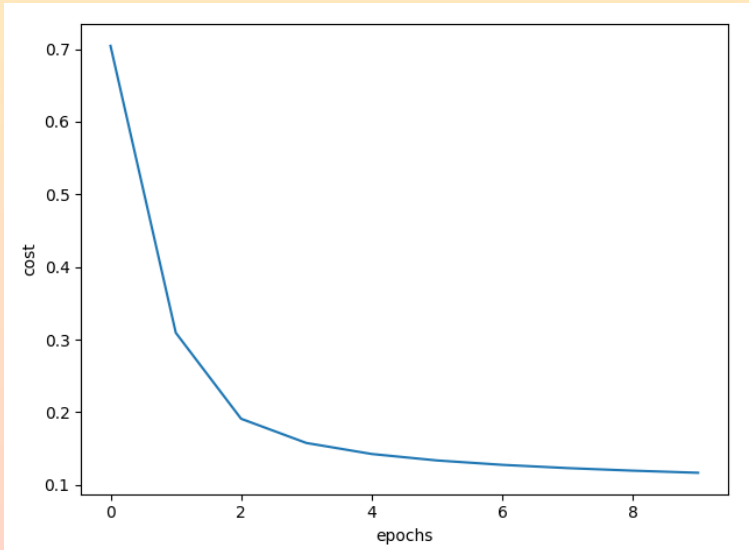


Image copyright Vischia, 2019

Loss function and backpropagation

- Adjust the parameters of each neuron and connection by backpropagating the difference between the estimated and the true output
- Differentiation and matrix (tensor) operations; dedicated software, automatic differentiation frameworks (e.g. tensorflow)
- Minimization of a cost (loss) function; the loss function can be tweaked to optimize w.r.t. several different objectives





Images from Güneş Baydin et al, JMLR 18 (2018) 1–43 and http://www.adeveloperdiary.com

Make the training (mostly) possible

- In real problems, it's not guaranteed that a simple gradient descent can find $argmin(Loss)$
- Several techniques to help the process to happen

Choose sampling scheme for loss and backpropagation

- Batch: compute on the whole training set (for large sets becomes too costly)
- Stochastic: compute on one sample (large noise, difficult to converge)
- Mini-batch: use a relatively small sample of data (tradeoff)

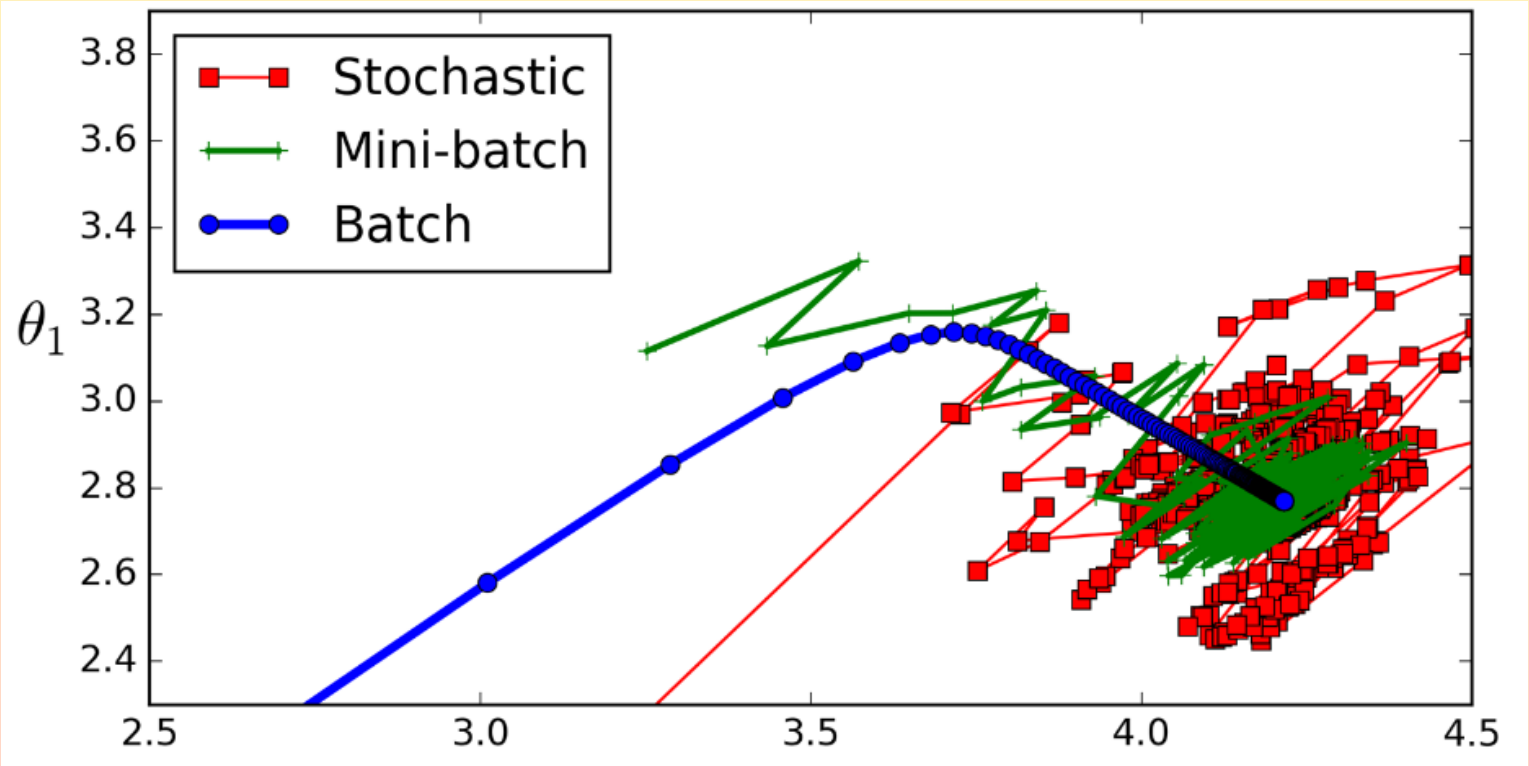
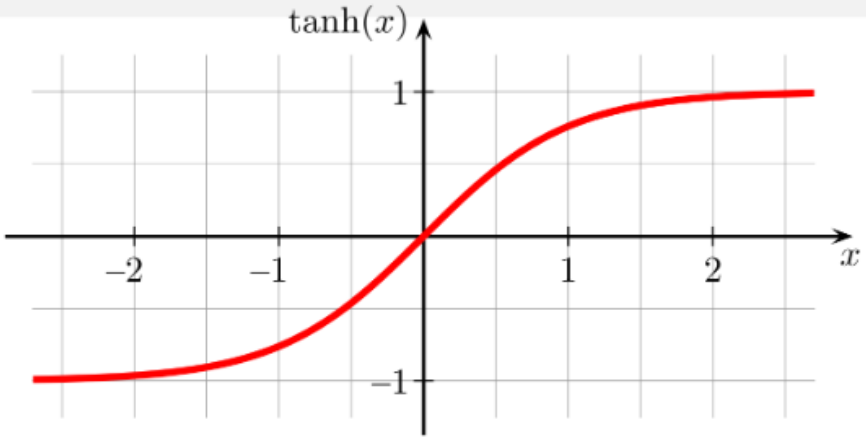
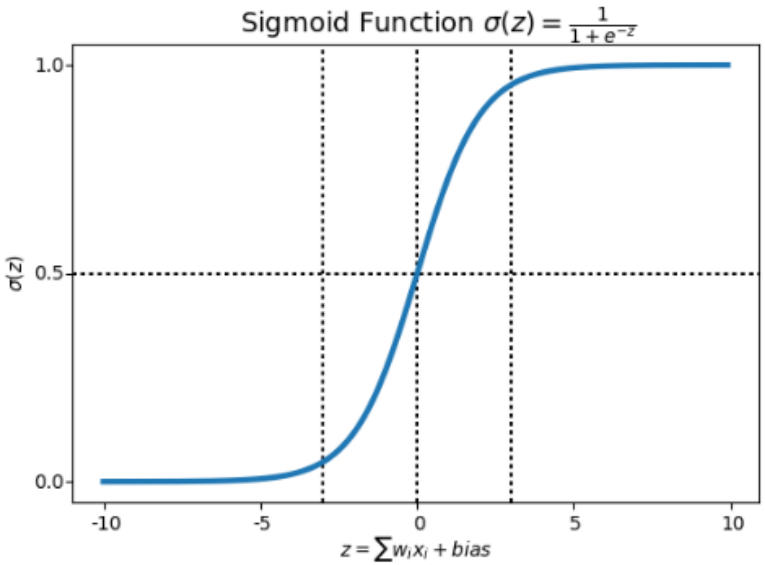


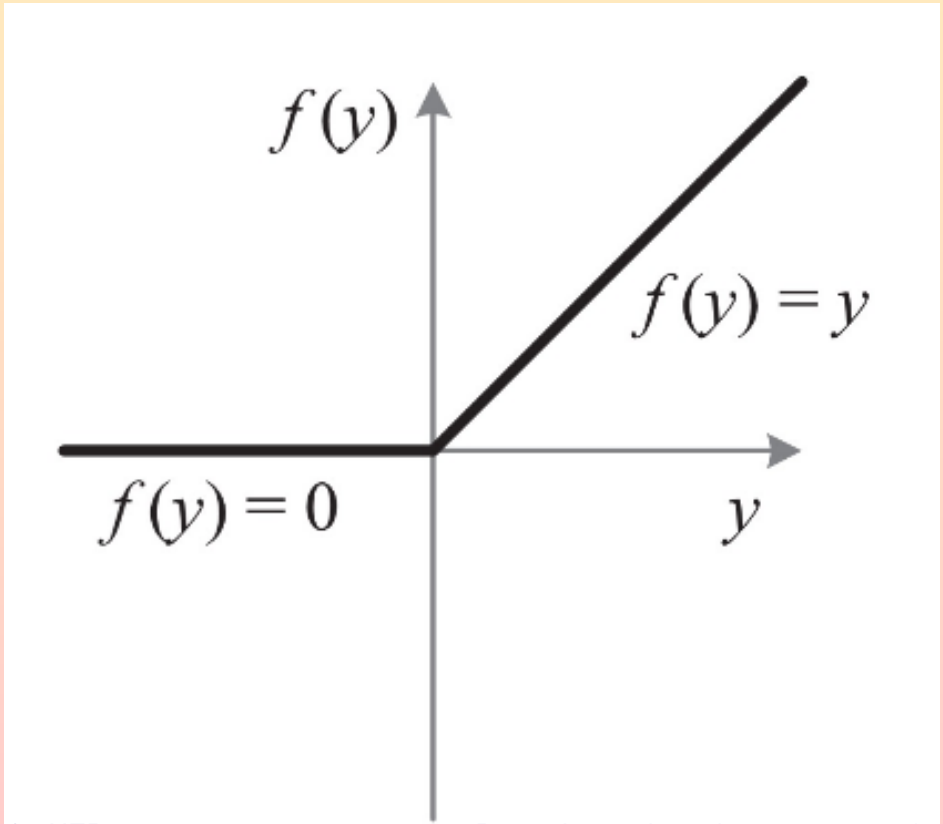
Image from a talk by W. Verbeke

Choose your activation function wisely

- *tanh* and *sigmoid* used a lot in the past
  - Seemed desirable to constrain neuron output to $[0, 1]$
  - For deep networks, vanishing gradients
  - *sigmoid* still used for output of the networks (outputs interpretable as probability)
- ReLU: a generally good choice for modern problems
  - Tricky cases may require variants

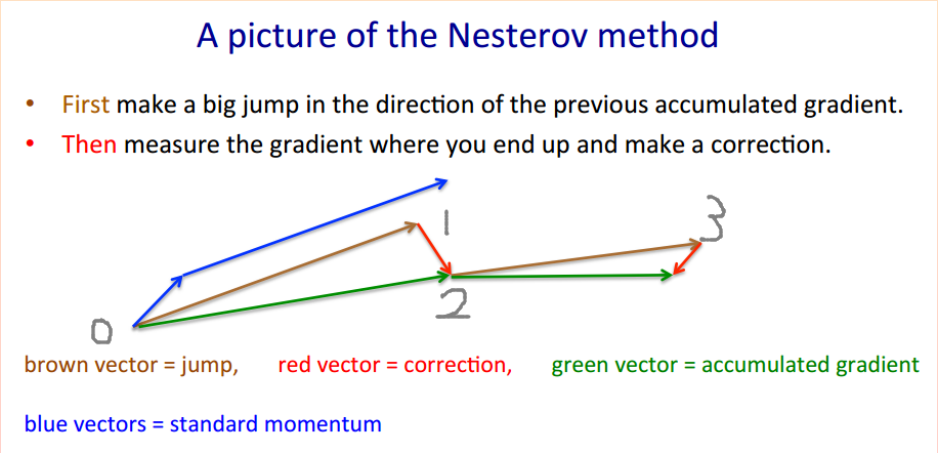






Improve algorithm to follow the gradient

- Mostly nonconvex optimization: very complicated problem, convergence in general not guaranteed
- Nesterov momentum: big jumps followed by correction seem to help!
- Adaptive moments: gradient steps decrease when getting closer to the minimum (avoids overshooting)

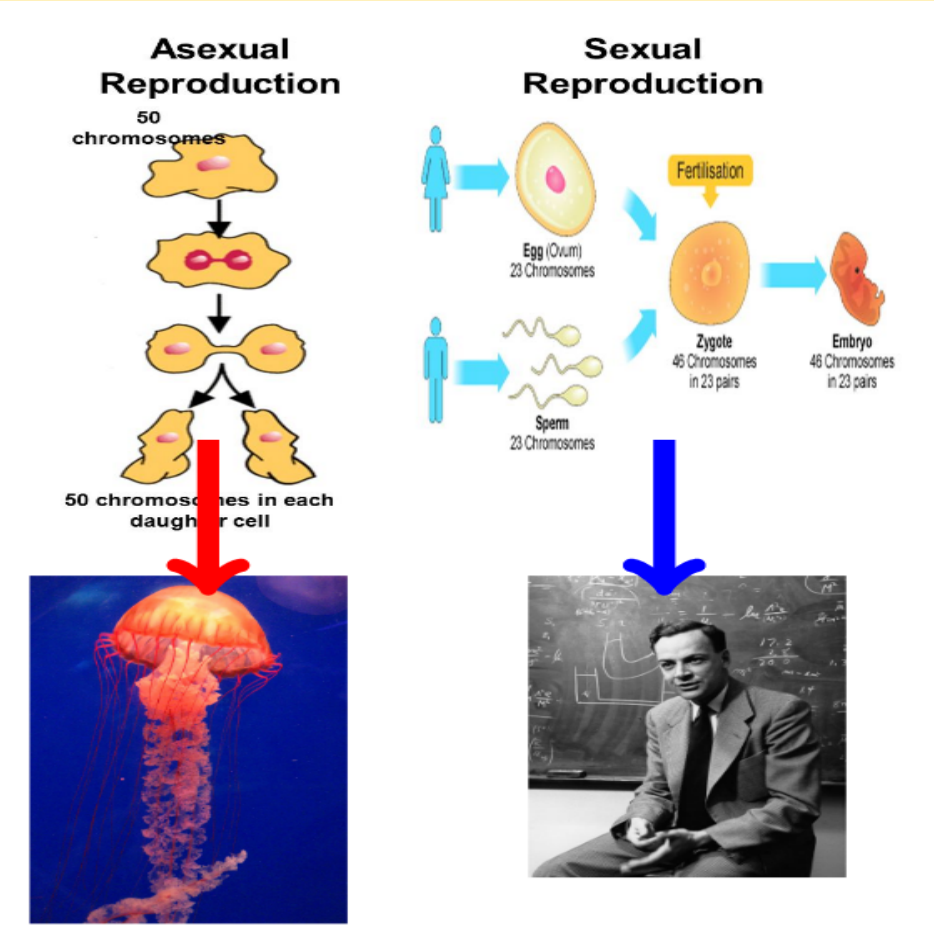


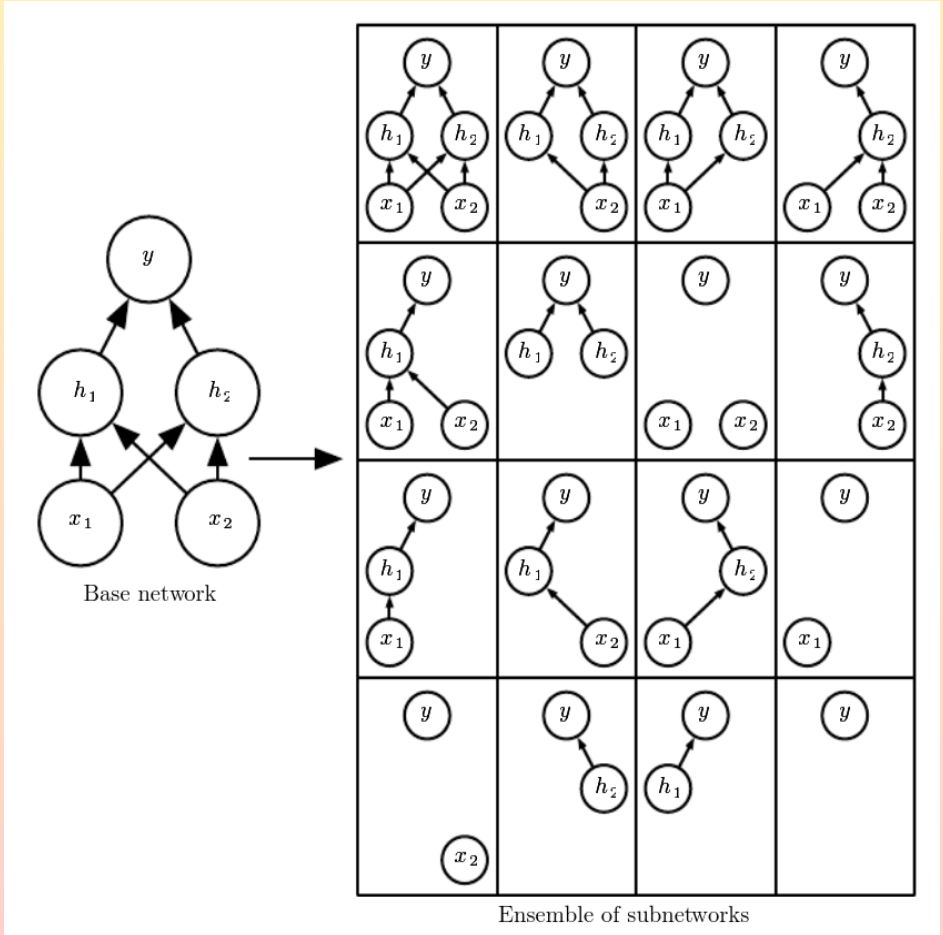
Nesterov Video

Diagram by Geoffrey Hinton, animation by Alec Radford

Regularization

- Batch normalization
  - Normalize (transform by $(x - \bar{x})/var(x)$) each input coming from previous layer over the (mini-)batch
  - Stabilizes response and reduces dependence among layers
- Dropoup: randomly shut down nodes in training
  - Avoids a weight to acquire too much importance
  - Inspired in genetics





Images from a talk by W. Verbeke (likely originally #theInternet) and from Goodfellow-Bengio-Courville book

1. Manual calculation, followed by explicit coding
2. Symbolic differentiation with expression manipulation (e.g. `Mathematica`)
3. Numerical differentiation with finite-difference approximations
4. Automatic (algorithmic) differentiation (AD): *autodiff*

- Question Time: Best Differentiation

Manual differentiation

- Manual calculation, followed by explicit coding
  - Time consuming and prone to error, require a closed-form model

$$l_1 = x$$
$$l_{n+1} = 4l_n(1 - l_n)$$
$$f(x) = l_4 = 64x(1-x)(1-2x)^2(1-8x+8x^2)^2$$

Coding

```
f(x):
   v = x
   for i = 1 to 3
     v = 4*v*(1 - v)
   return v
```

or, in closed-form,

```
f(x):
  return 64*x*(1-x)*((1-2*x)^2)
    *(1-8*x+8*x*x)^2
```

$$f'(x) = 128x(1-x)(-8+16x)(1-2x)^2(1-8x+8x^2)+64(1-x)(1-2x)^2(1-8x+8x^2)^2-64x(1-2x)^2(1-8x+8x^2)^2-256x(1-x)(1-2x)(1-8x+8x^2)^2$$

Coding

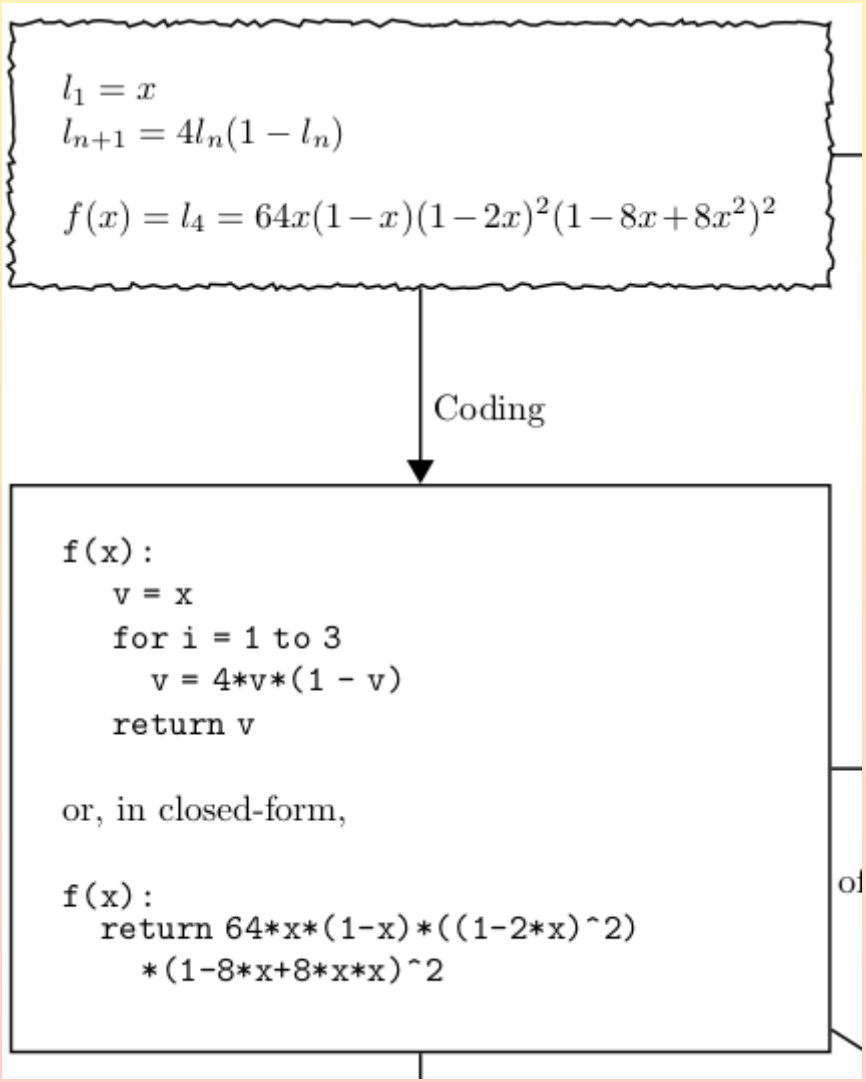
```
f'(x):
 return 128*x*(1 - x)*(-8 + 16*x)
   *((1 - 2*x)^2)*(1 - 8*x + 8*x*x)
   + 64*(1 - x)*((1 - 2*x)^2)*((1
   - 8*x + 8*x*x)^2) - (64*x*(1 -
   2*x)^2)*(1 - 8*x + 8*x*x)^2 -
   256*x*(1 - x)*(1 - 2*x)*(1 - 8*x
   + 8*x*x)^2
```

$f'(x_0) = f'(x_0)$
Exact

Image from Güneş Baydin et al, JMLR 18 (2018) 1–43

Symbolic differentiation

- Symbolic differentiation with expression manipulation (e.g. `Mathematica`, `Theano`)
  - Complex expressions, require a closed-form model
  - Sometimes can just minimize the problem without requiring derivative calculation
  - Nested duplications produce exponentially large symbolic expressions (*expression swell*, slow to evaluate)



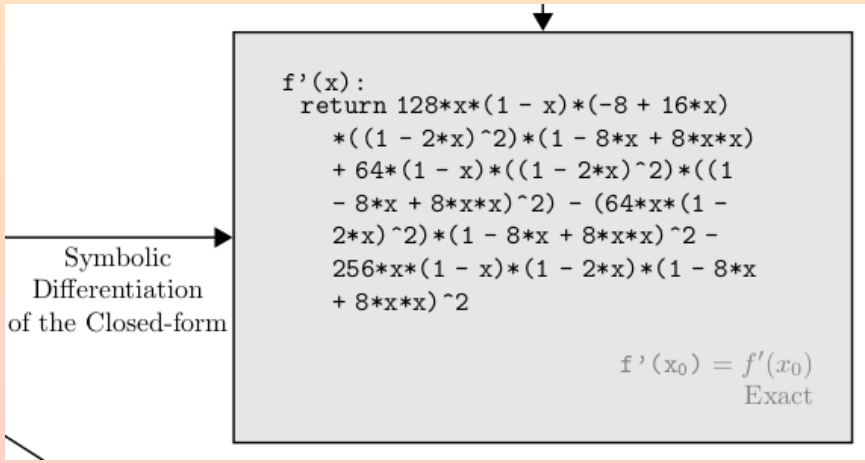


Image from Güneş Baydin et al, JMLR 18 (2018) 1–43

Standard numerical differentiation

- Numerical differentiation with finite-difference approximations
  - Rounding errors and truncation errors can make it very inaccurate
  - Mitigation techniques that cancel first-order errors are computationally costly
  - Accuracy must be traded off for performance for high dimensionalities

$$l_1 = x$$
$$l_{n+1} = 4l_n(1 - l_n)$$

$$f(x) = l_4 = 64x(1-x)(1-2x)^2(1-8x+8x^2)^2$$

Coding

```
f(x):
   v = x
   for i = 1 to 3
     v = 4*v*(1 - v)
   return v
```

or, in closed-form,

```
f(x):
  return 64*x*(1-x)*((1-2*x)^2)
    *(1-8*x+8*x*x)^2
```

```
f'(x):
  h = 0.000001
  return (f(x + h) - f(x)) / h
```

$f'(x_0) \approx f'(x_0)$
Approximate

Image from Güneş Baydin et al, JMLR 18 (2018) 1–43

Automatic differentiation

- Automatic (algorithmic) differentiation (AD): *autodiff*
  - Class of techniques to generate numerical derivative evaluations during code execution rather than derivative expressions
  - Accurate at machine precision with small constant overhead and asymptotic efficiency
  - No need to rearrange the code in a closed-form expression
  - Reverse AD generalizes the common chain-rule-based neural network backpropagation

$$l_1 = x$$
$$l_{n+1} = 4l_n(1 - l_n)$$

$$f(x) = l_4 = 64x(1-x)(1-2x)^2(1-8x+8x^2)^2$$

Coding

```
f(x):
   v = x
   for i = 1 to 3
     v = 4*v*(1 - v)
   return v
```

or, in closed-form,

```
f(x):
  return 64*x*(1-x)*((1-2*x)^2)
    *(1-8*x+8*x*x)^2
```

```
f'(x):
  (v,dv) = (x,1)
  for i = 1 to 3
    (v,dv) = (4*v*(1-v), 4*dv-8*v*dv)
  return (v,dv)
```

$$f'(x_0) = f'(x_0)$$
Exact

Image from Günes Baydin et al, JMLR 18 (2018) 1–43

The power of autodifferentiation

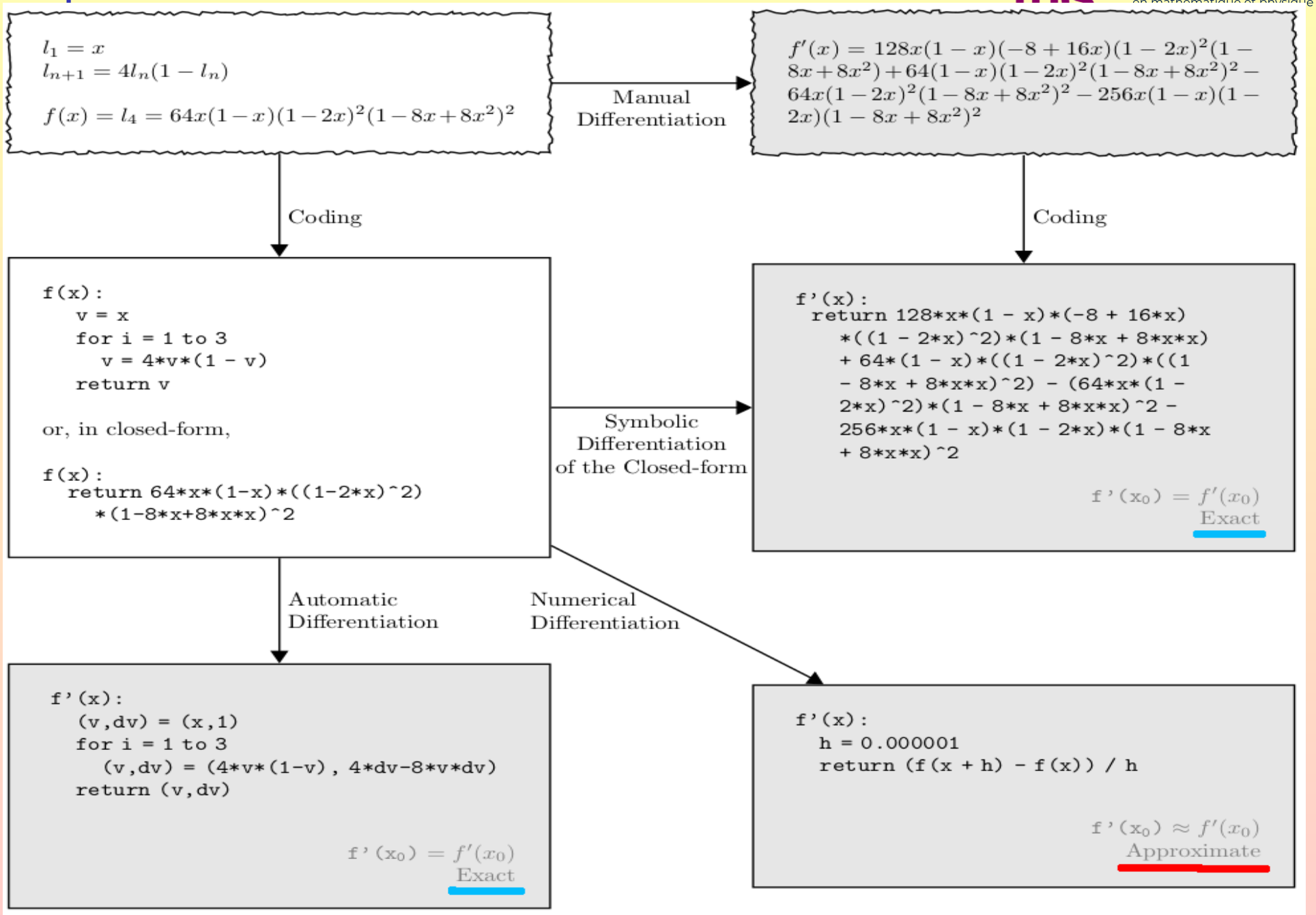


Image from Güneş Baydin et al, JMLR 18 (2018) 1–43

The two modes of autodiff

UCLouvain
Institut de recherche
en mathématique et physique

Forward mode

- Associate with each intermediate v_i a derivative
  $\dot{v} = \dfrac{\partial v_i}{\partial x_1}$
- Apply the chain rule
- Single pass for $f : \mathbb{R} \to \mathbb{R}^n$
- n passes for $f : \mathbb{R}^n \to \mathbb{R}$

Forward Primal Trace

v_{-1}	$= x_1$	$= 2$
v_0	$= x_2$	$= 5$
v_1	$= \ln v_{-1}$	$= \ln 2$
v_2	$= v_{-1} \times v_0$	$= 2 \times 5$
v_3	$= \sin v_0$	$= \sin 5$
v_4	$= v_1 + v_2$	$= 0.693 + 10$
v_5	$= v_4 - v_3$	$= 10.693 + 0.959$
y	$= v_5$	$= 11.652$

Forward Tangent (Derivative) Trace

$\dot{v}_{-1}$	$= \dot{x}_1$	$= 1$
$\dot{v}_0$	$= \dot{x}_2$	$= 0$
$\dot{v}_1$	$= \dot{v}_{-1}/v_{-1}$	$= 1/2$
$\dot{v}_2$	$= \dot{v}_{-1} \times v_0 + \dot{v}_0 \times v_{-1}$	$= 1 \times 5 + 0 \times 2$
$\dot{v}_3$	$= \dot{v}_0 \times \cos v_0$	$= 0 \times \cos 5$
$\dot{v}_4$	$= \dot{v}_1 + \dot{v}_2$	$= 0.5 + 5$
$\dot{v}_5$	$= \dot{v}_4 - \dot{v}_3$	$= 5.5 - 0$
$\dot{y}$	$= \dot{v}_5$	$= \mathbf{5.5}$

Reverse mode

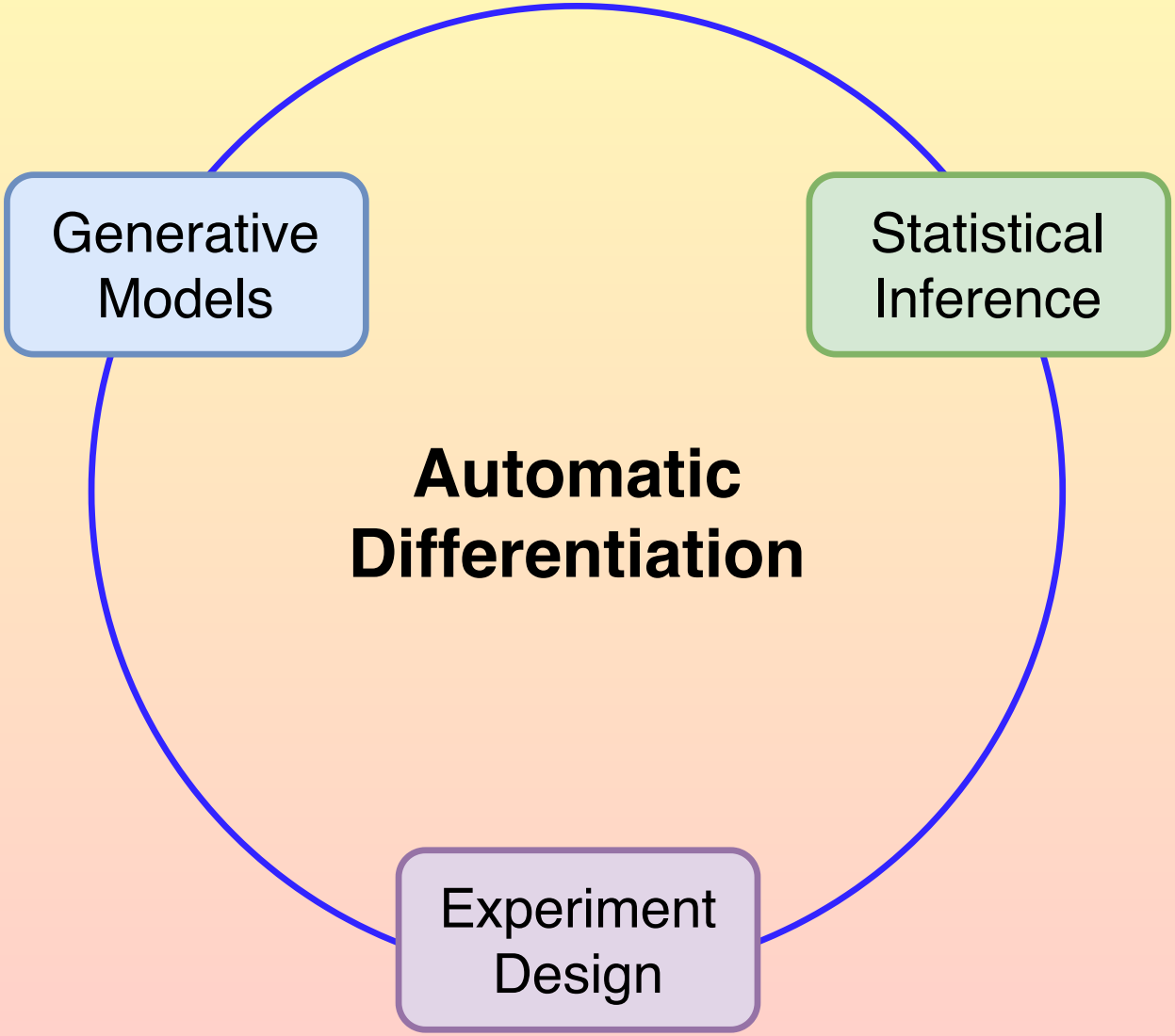
- Associate with each intermediate v_i an adjoint
  $\bar{v} = \dfrac{\partial y_j}{\partial v_i}$
- Run forwards and backwards as in backpropagation
- Single pass for $f : \mathbb{R}^n \to \mathbb{R}$ (functions with many inputs)
- Must store several values

Forward Primal Trace

v_{-1}	$= x_1$	$= 2$
v_0	$= x_2$	$= 5$
v_1	$= \ln v_{-1}$	$= \ln 2$
v_2	$= v_{-1} \times v_0$	$= 2 \times 5$
v_3	$= \sin v_0$	$= \sin 5$
v_4	$= v_1 + v_2$	$= 0.693 + 10$
v_5	$= v_4 - v_3$	$= 10.693 + 0.959$
y	$= v_5$	$= 11.652$

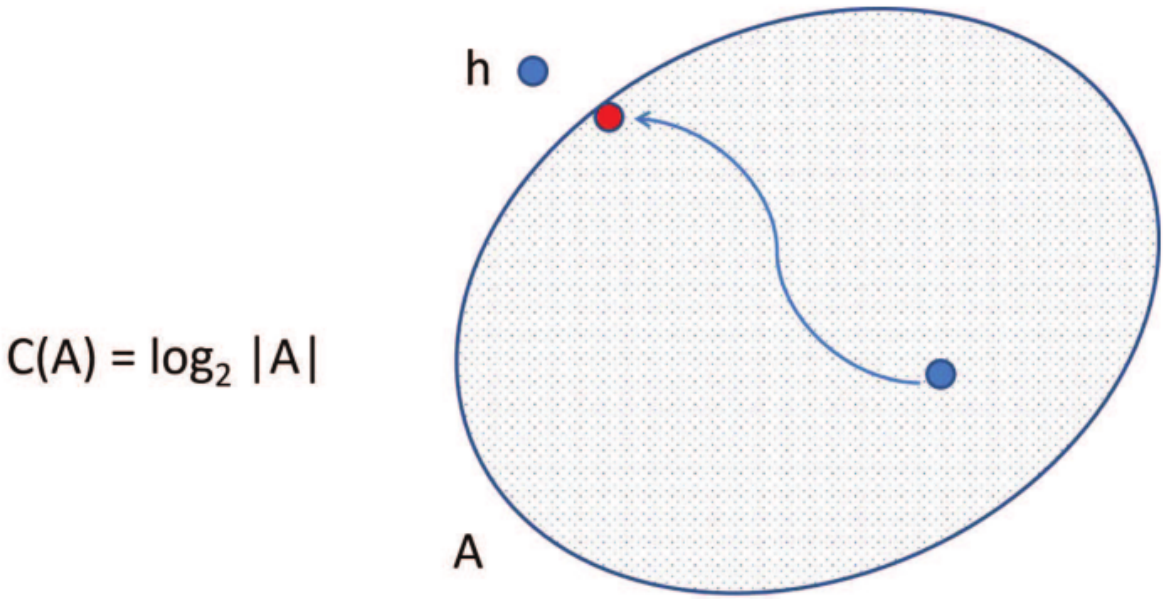
Reverse Adjoint (Derivative) Trace

$\bar{x}_1$	$= \bar{v}_{-1}$		$= \mathbf{5.5}$
$\bar{x}_2$	$= \bar{v}_0$		$= \mathbf{1.716}$
$\bar{v}_{-1}$	$= \bar{v}_{-1} + \bar{v}_1 \frac{\partial v_1}{\partial v_{-1}}$	$= \bar{v}_{-1} + \bar{v}_1 / v_{-1}$	$= 5.5$
$\bar{v}_0$	$= \bar{v}_0 + \bar{v}_2 \frac{\partial v_2}{\partial v_0}$	$= \bar{v}_0 + \bar{v}_2 \times v_{-1}$	$= 1.716$
$\bar{v}_{-1}$	$= \bar{v}_2 \frac{\partial v_2}{\partial v_{-1}}$	$= \bar{v}_2 \times v_0$	$= 5$
$\bar{v}_0$	$= \bar{v}_3 \frac{\partial v_3}{\partial v_0}$	$= \bar{v}_3 \times \cos v_0$	$= -0.284$
$\bar{v}_2$	$= \bar{v}_4 \frac{\partial v_4}{\partial v_2}$	$= \bar{v}_4 \times 1$	$= 1$
$\bar{v}_1$	$= \bar{v}_4 \frac{\partial v_4}{\partial v_1}$	$= \bar{v}_4 \times 1$	$= 1$
$\bar{v}_3$	$= \bar{v}_5 \frac{\partial v_5}{\partial v_3}$	$= \bar{v}_5 \times (-1)$	$= -1$
$\bar{v}_4$	$= \bar{v}_5 \frac{\partial v_5}{\partial v_4}$	$= \bar{v}_5 \times 1$	$= 1$
$\bar{v}_5$	$= \bar{y}$		$= 1$


Generative Models
Statistical Inference
Automatic Differentiation
Experiment Design

Using deep neural networks is not always necessary

- Each realization of a machine learning algorithm has a certain complexity
- *Capacity* can be defined as the upper bound to the number of bits that can be stored in the network during learning
  - Transfer of (Fisher or Shannon) information from the training data to the weights of the synapses
- Sometimes the problem does not need the capacity of a neural network, and simpler algorithms are enough
  - Identifying true leptons from leptons produced in b hadron decays is an example



$C(A) = \log_2 |A|$

Figure 1. Learning framework where h is the function to be learnt and A is the available class of hypothesis or approximating functions. The cardinal capacity is the logarithm base two of the number, or volume, of the functions contained in A.

Plot from Baldi and Vershynin, arXiv:1901.00434

Separate prompt from fake leptons...

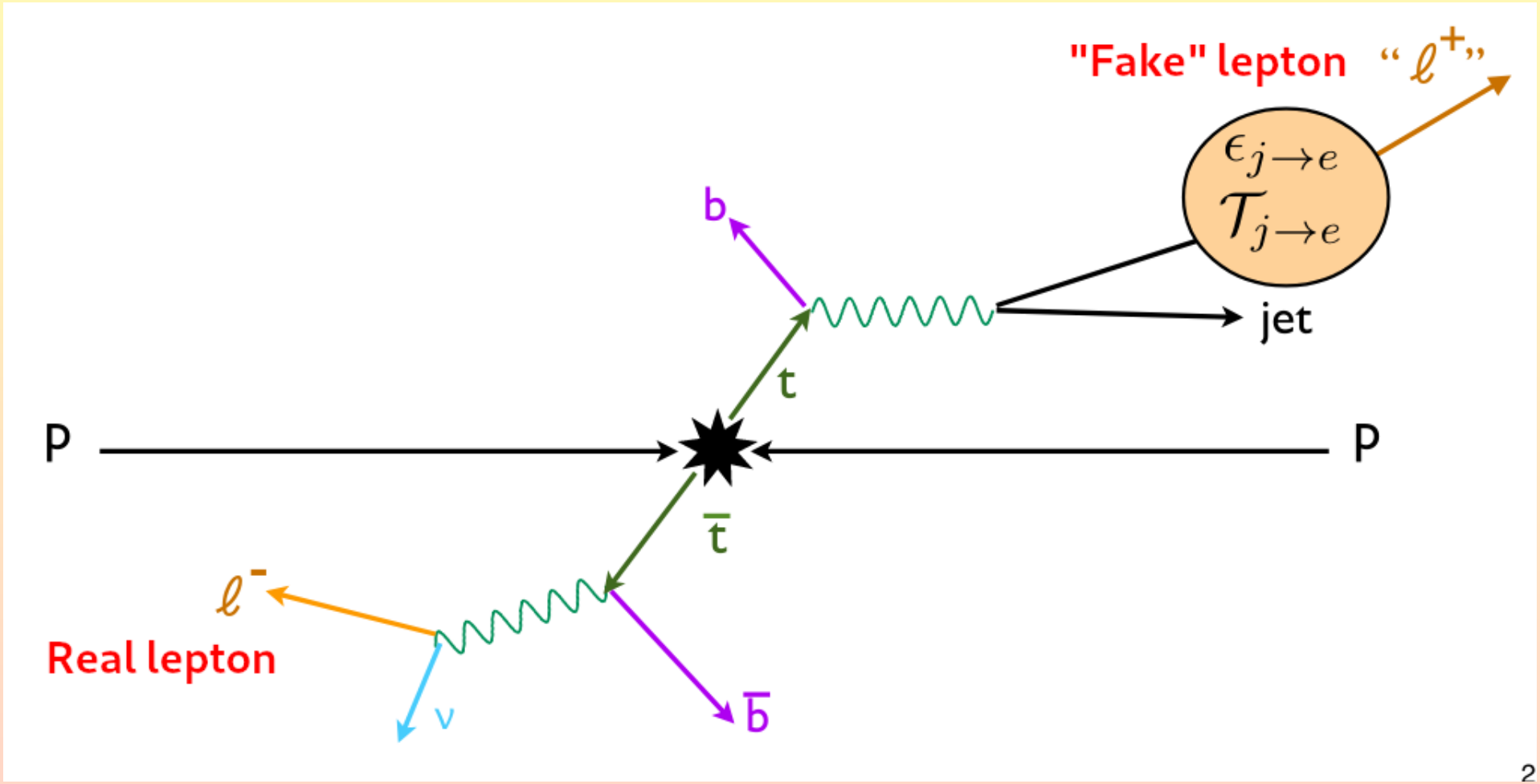
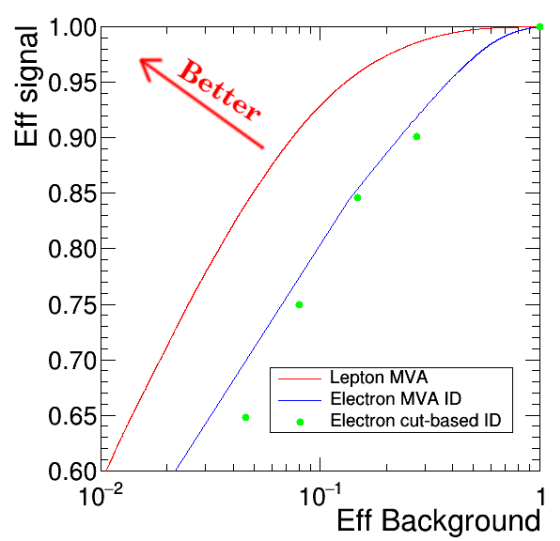
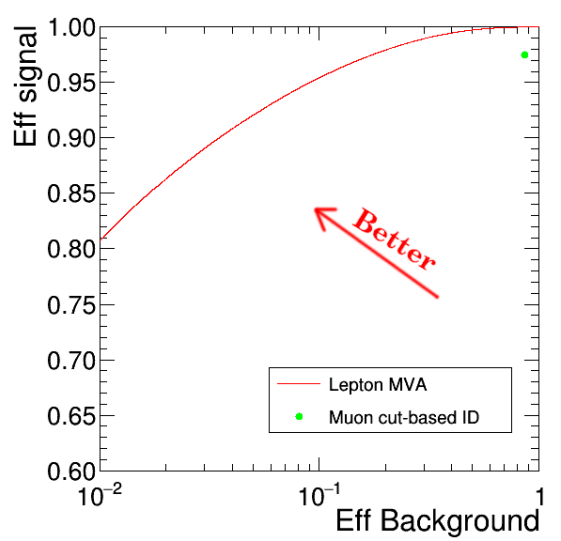


Image edited from David Curtin's talk at MC4BSM-2014

...is not very difficult

- Baseline algorithms: select particular ranges of discriminant observables
- BDT-based MVA ID improves substantially w.r.t baseline algorithms





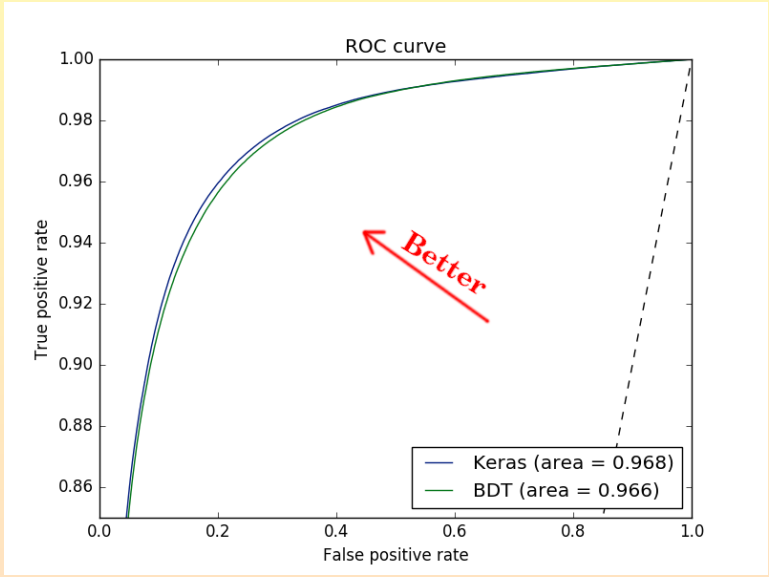
Plots by S.S. Cruz

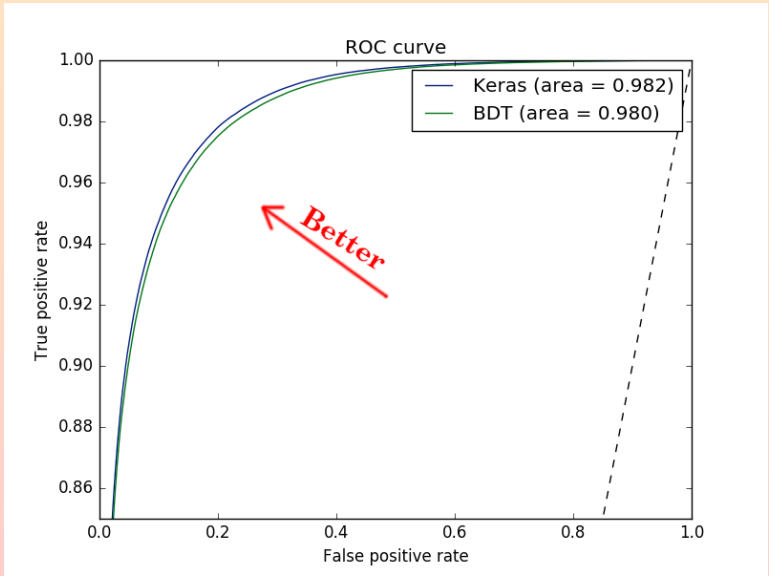
Corte	Background Fondo					Signal Señal ($t\bar{t}H$)		Datos		$\frac{\text{Predicción}}{\text{Datos}}$	$\frac{S}{\sqrt{F+(\Delta F)^2}}$
	Fakes			Total							
	Valor	Δ	$\frac{Fakes}{Fakes(t\bar{t}H)}$ (%)	*Valor*	Δ	*Valor*	Δ	*Valor*	Δ		
> 0.97	14	7	5	246	22	46	5	363	19	0.804	1.694
> 0.95	177	71	60	471	75	62	7	524	23	1.017	0.788
$t\bar{t}H$	295	103	100	658	109	79	9	752	27	0.981	0.731
Extra tight	517	168	175	938	173	96	10	1056	32	0.979	0.545
Very tight	751	238	255	1200	242	102	11	1338	37	0.973	0.417
Tight	1032	323	350	1500	326	107	12	1624	40	0.990	0.325
Medium	1498	466	508	1988	468	111	12	2074	46	1.012	0.235

Cuadro 5.7: Resultados en número de eventos del análisis del proceso $t\bar{t}H$ para todas las categorías según el corte realizado en la variable `Lepton MVA`.

Table by Víctor Rodríguez Bouza

- Deep neural network (DNN) does not help much w.r.t. BDT





Plots by Antonio Márquez García

Neural networks can approximate any continuous real-valued function

- A feed-forward network with sigmoid activation functions can approximate any continuos real-valued function. Cybenko, G. (1989)
- Any failure in mapping a function comes from inadequate choice of weights or insufficient number of neurons. Hornik et al (1989), Funahashi (1989)
- Derivatives can be approximated as well as the functions, even in case of non-differentiability (e.g. piecewise differentiable functions). Hornik et al (1990)
- These results are valid even with other classes of activation functions. Light (1992), Stinchcombe and White (1989), Baldi (1991), Ito (1991), etc

Neural networks can be used to build fully invertible models

- The backpropagation algorithm is a special case of *automatic differentiation*
- A fully invertible model is a powerful tool that can be used for many frontier applications in particle physics

Difficult tasks for humans may be easy for artificial networks

UCLouvain
Institut de recherche
en mathématique et physique

```
[vischia@lxplus797 ~]$ python dump_cpTree.py
Info in <TCanvas::MakeDefCanvas>:  created default TCanvas with name c1
*****************************************************************************************************************************************************************
*    Row   * Instance * Lep1_pt.L * Lep2_pt.L * Lep3_pt.L * Lep1_eta. * Lep2_eta. * Lep3_eta. * Lep1_phi. * Lep2_phi. * Lep3_phi. *   met.met * met_phi.m * weight_CP * HTT_score *
*****************************************************************************************************************************************************************
*    56350 *        0 * 280.82870 * 93.600341 * 59.040779 * -1.360351 * 0.7332763 * -0.819335 * -2.826171 * 0.7969970 * -3.100097 * 118.03358 * 2.5146484 * 40.820312 * 0.8255668 *
*    56350 *        1 * 280.82870 * 93.600341 * 59.040779 * -1.360351 * 0.7332763 * -0.819335 * -2.826171 * 0.7969970 * -3.100097 * 118.03358 * 2.5146484 * 40.820312 * 0.8255668 *
*    56350 *        2 * 280.82870 * 93.600341 * 59.040779 * -1.360351 * 0.7332763 * -0.819335 * -2.826171 * 0.7969970 * -3.100097 * 118.03358 * 2.5146484 * 40.820312 * 0.8255668 *
*    56350 *        3 * 280.82870 * 93.600341 * 59.040779 * -1.360351 * 0.7332763 * -0.819335 * -2.826171 * 0.7969970 * -3.100097 * 118.03358 * 2.5146484 * 40.820312 * 0.8255668 *
*    79791 *        0 * 67.791183 * 42.294036 * 17.310911 * -1.846923 * 1.4458007 * -0.762207 * -0.628540 * -2.910644 * 2.9912109 * 8.8895807 * -1.773925 * 94.398437 *       -99 *
*****************************************************************************************************************************************************************
==> 5 selected entries
-------------------------------------------------------------
*****************************************************************************************************************************************************************
*    Row   * Instance * Lep1_pt.L * Lep2_pt.L * Lep3_pt.L * Lep1_eta. * Lep2_eta. * Lep3_eta. * Lep1_phi. * Lep2_phi. * Lep3_phi. *   met.met * met_phi.m * weight_CP * HTT_score *
*****************************************************************************************************************************************************************
*    58751 *        0 * 86.053443 * 32.593345 * 13.077508 * -1.346435 * 2.0512695 * -0.503906 * -0.392944 * -2.925781 * 0.9309082 * 17.544691 * 0.1735229 * 3.478e-08 * 0.9726203 *
*    58751 *        1 * 86.053443 * 32.593345 * 13.077508 * -1.346435 * 2.0512695 * -0.503906 * -0.392944 * -2.925781 * 0.9309082 * 17.544691 * 0.1735229 * 3.478e-08 * 0.9726203 *
*    58751 *        2 * 86.053443 * 32.593345 * 13.077508 * -1.346435 * 2.0512695 * -0.503906 * -0.392944 * -2.925781 * 0.9309082 * 17.544691 * 0.1735229 * 3.478e-08 * 0.9726203 *
*    58751 *        3 * 86.053443 * 32.593345 * 13.077508 * -1.346435 * 2.0512695 * -0.503906 * -0.392944 * -2.925781 * 0.9309082 * 17.544691 * 0.1735229 * 3.478e-08 * 0.9726203 *
*    58751 *        4 * 86.053443 * 32.593345 * 13.077508 * -1.346435 * 2.0512695 * -0.503906 * -0.392944 * -2.925781 * 0.9309082 * 17.544691 * 0.1735229 * 3.478e-08 * 0.9726203 *
*    58751 *        5 * 86.053443 * 32.593345 * 13.077508 * -1.346435 * 2.0512695 * -0.503906 * -0.392944 * -2.925781 * 0.9309082 * 17.544691 * 0.1735229 * 3.478e-08 * 0.9726203 *
*****************************************************************************************************************************************************************
==> 6 selected entries
```

Signal

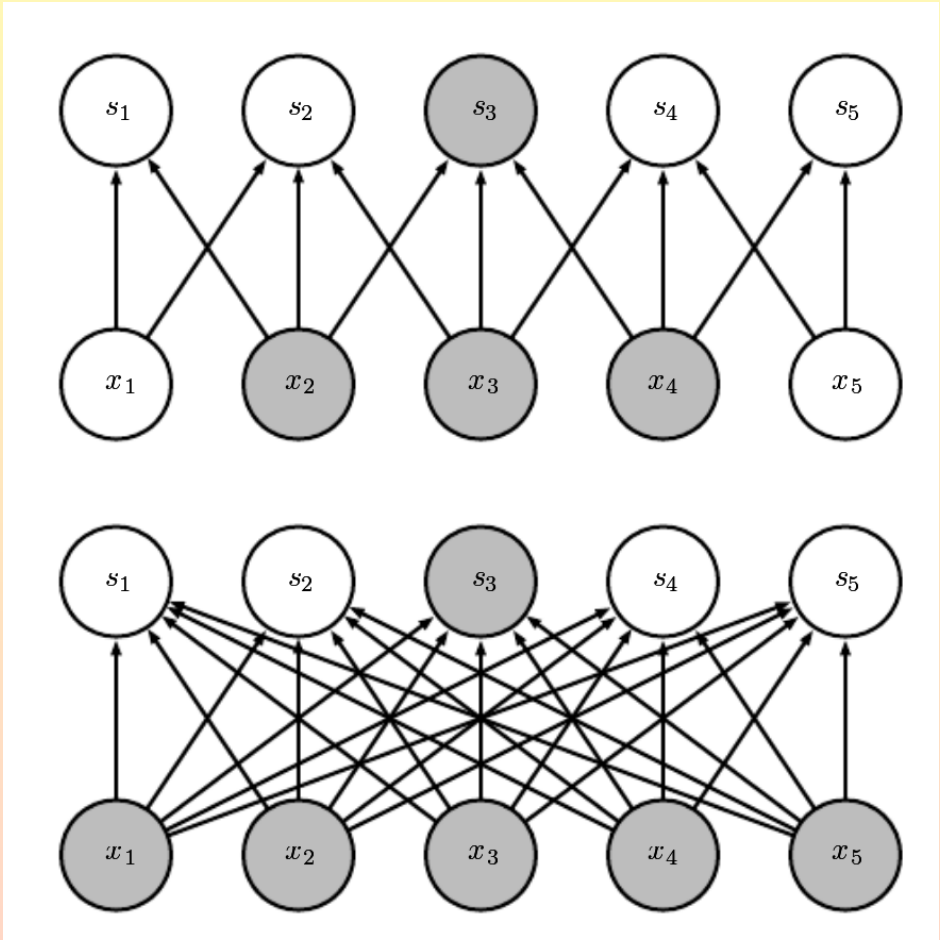
Background

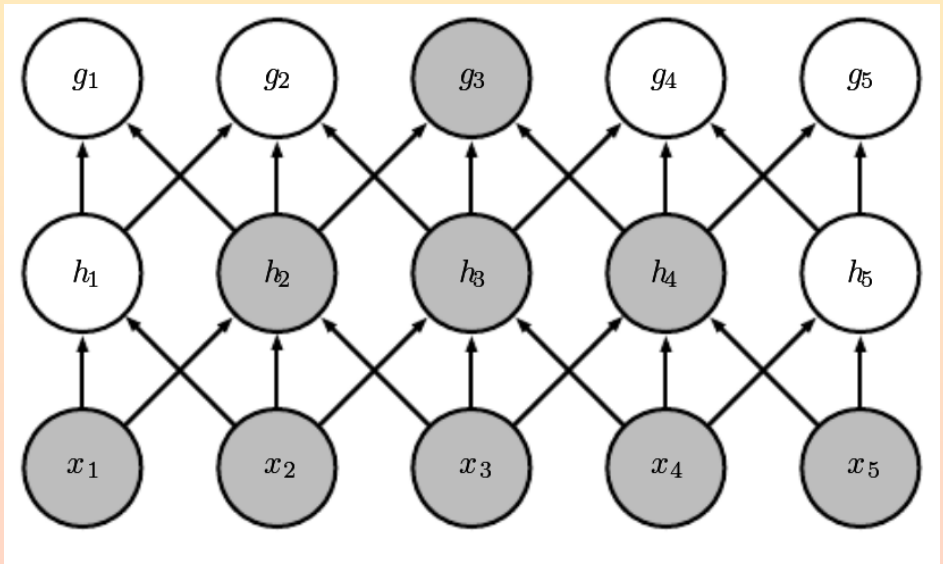
Image by Pietro Vischia

Easy tasks for humans may be very difficult for artificial networks

Image from indiatimes.com

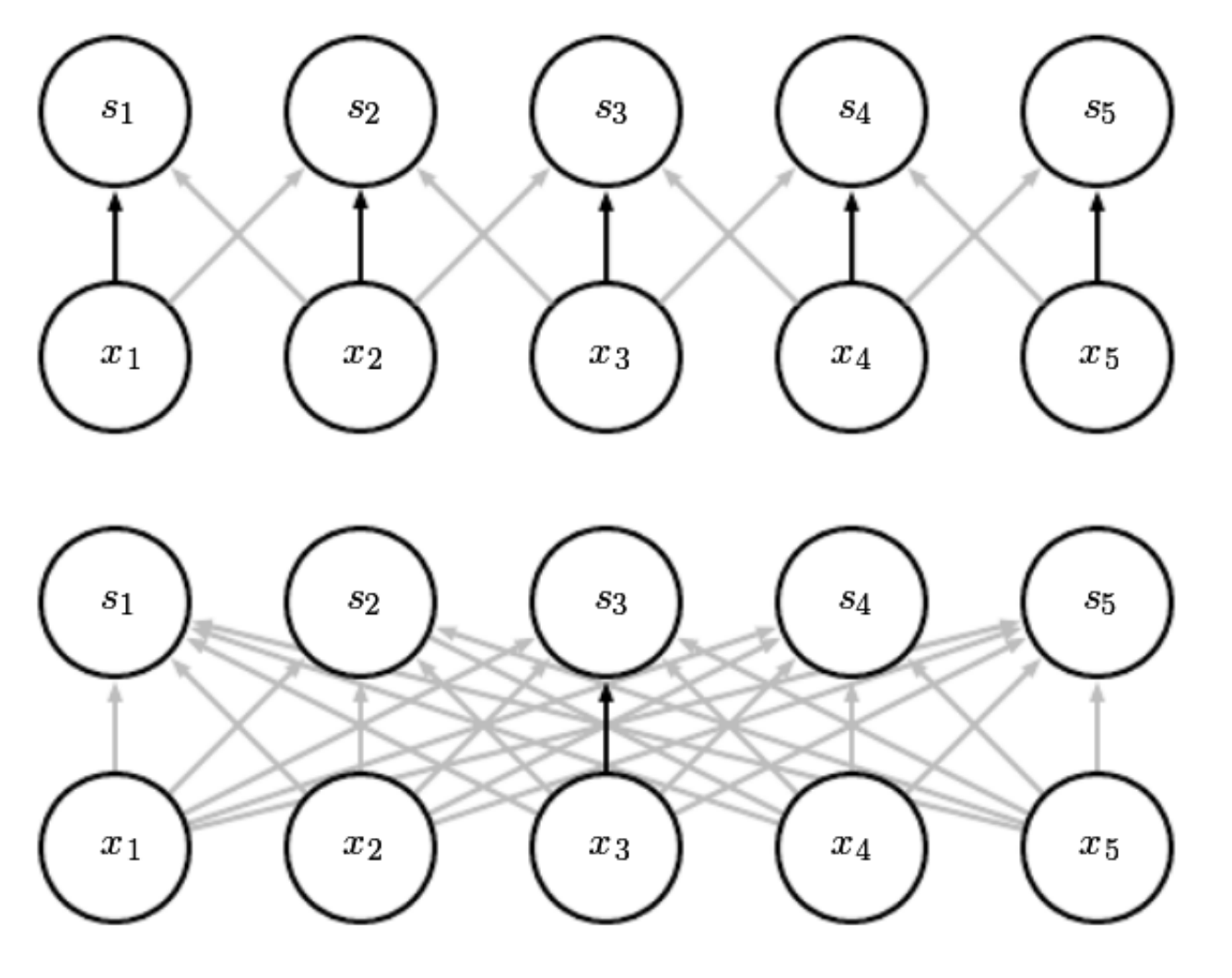
Sparse connectivity and receptive fields





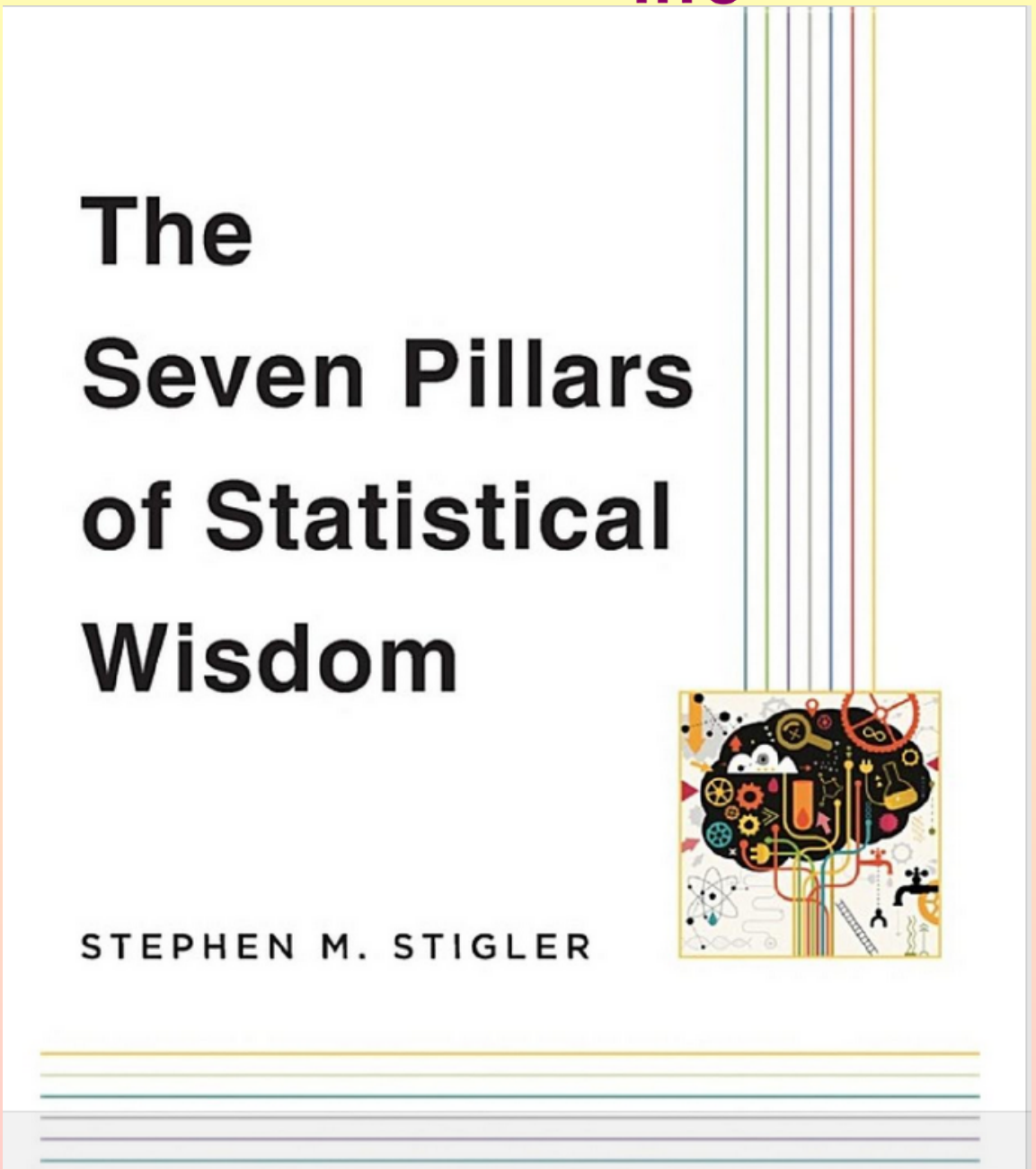
Images from https://www.deeplearningbook.org/

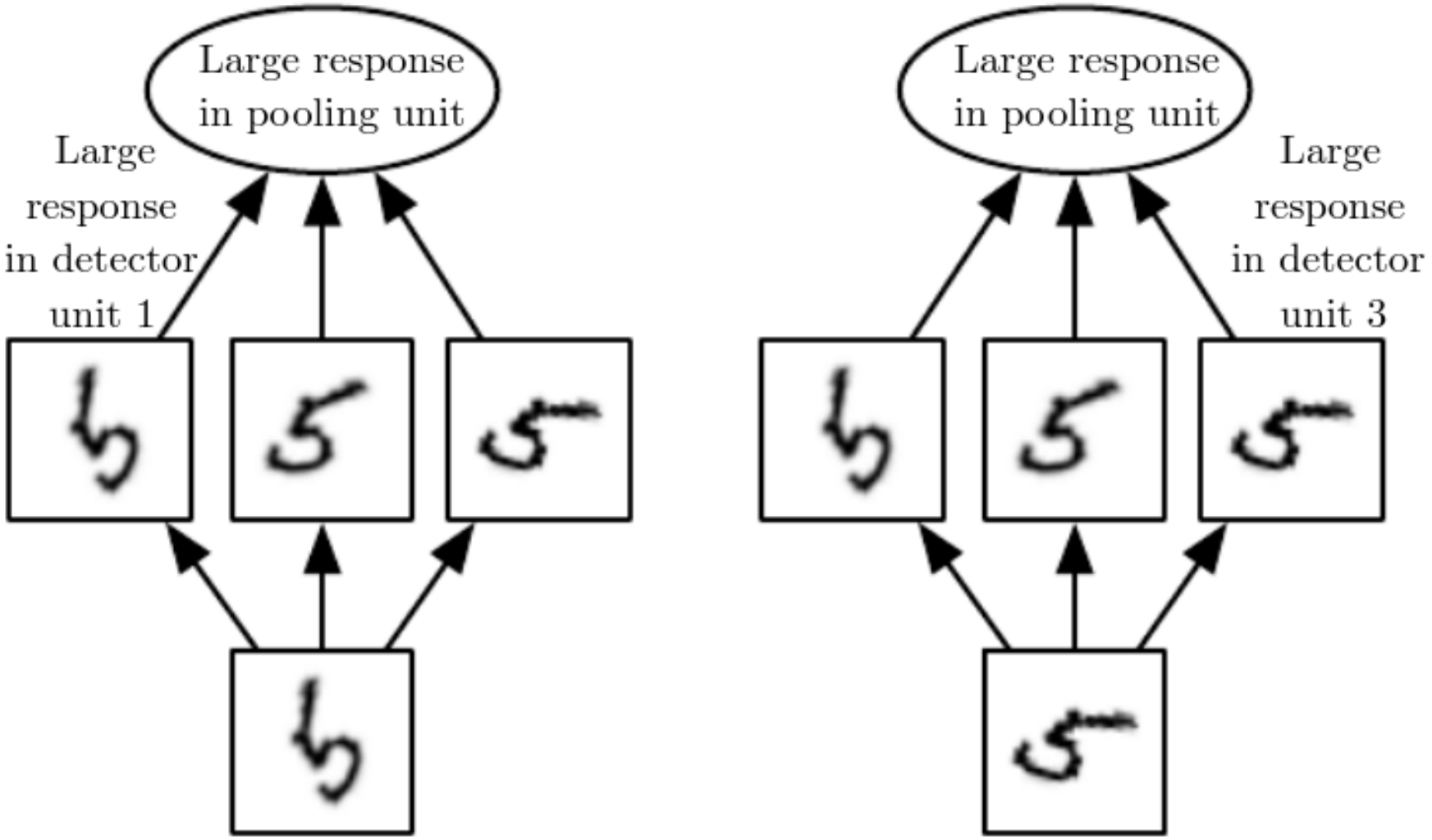
Parameter sharing and equivariance to translations



Images from https://www.deeplearningbook.org/

- **Aggregation**
- Information
- Likelihood
- Intercomparison
- Regression
- Design
- Residual



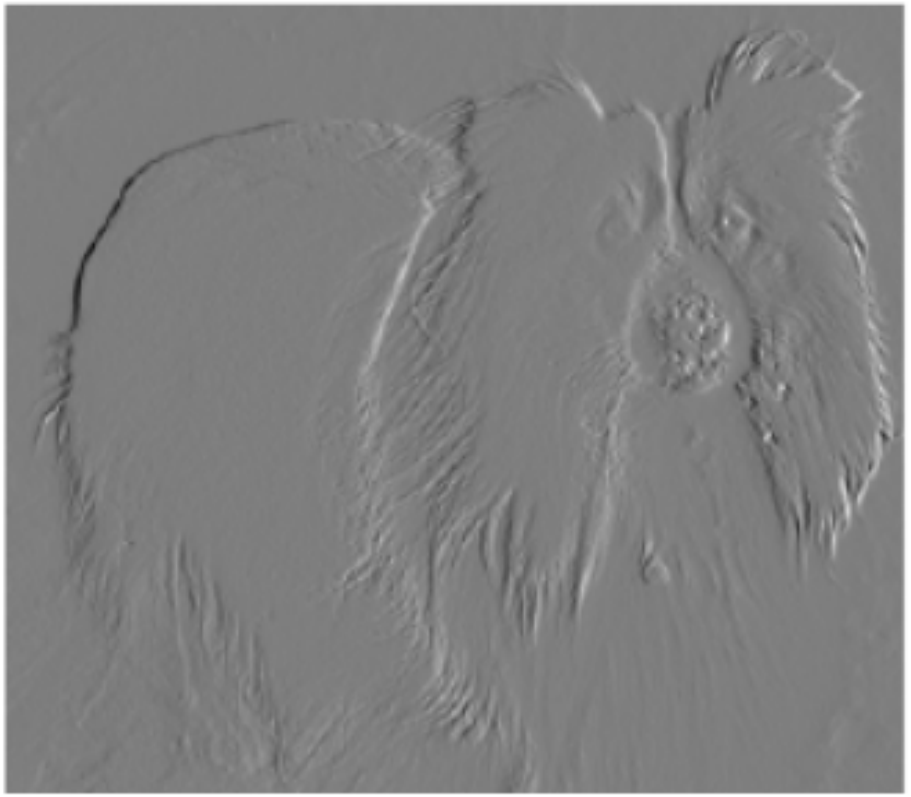


Images from https://www.deeplearningbook.org/

Convolution makes it easier to learn transformations

- Standard fully connected network: 8 billion matrix entries, 16 billion floating-point operations
- Convolutional network: 2 matrix entries, 267960 floating-point operations
  - 4 billion times more efficient in representing the transformation
  - 60000 times more efficient computationally

Edge detection



Images from https://www.deeplearningbook.org/

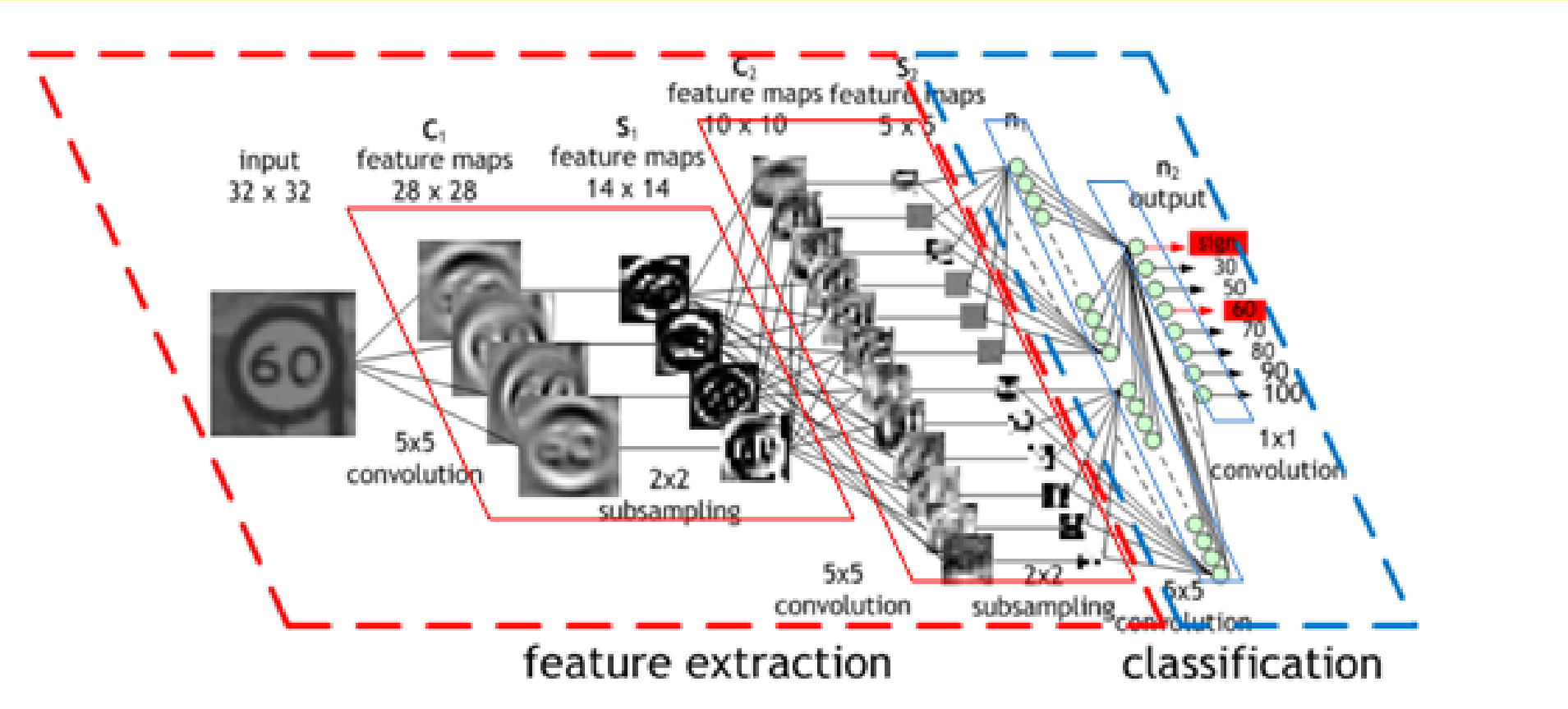
- LeNet (Yann LeCun 1998, http://yann.lecun.com/exdb/lenet/)

LeNet

- LeNet (Yann LeCun 1998, http://yann.lecun.com/exdb/lenet/)

LeNet

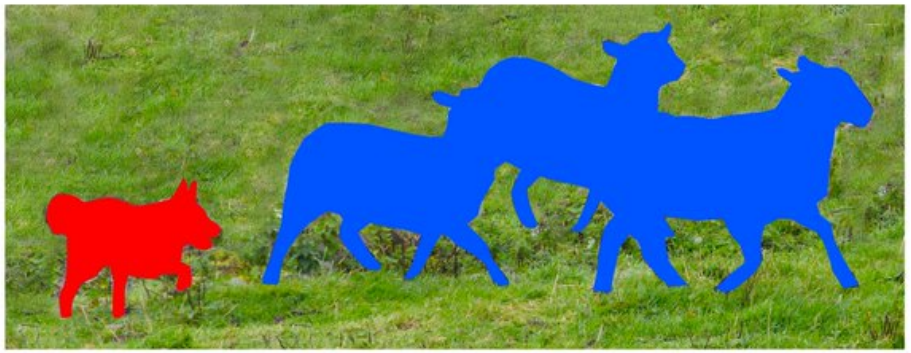
Learning in stages



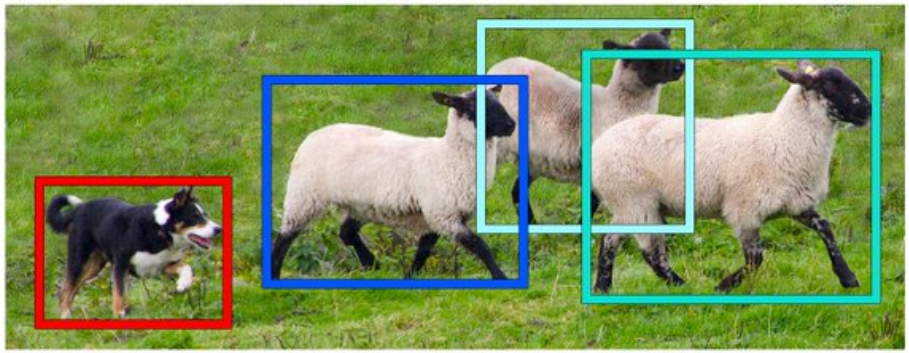
From http://parse.ele.tue.nl/education/cluster0

Segmentating objects

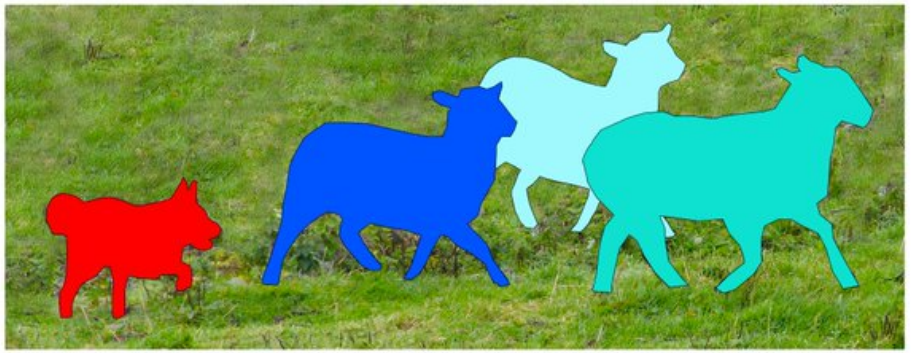
Image Recognition



Semantic Segmentation

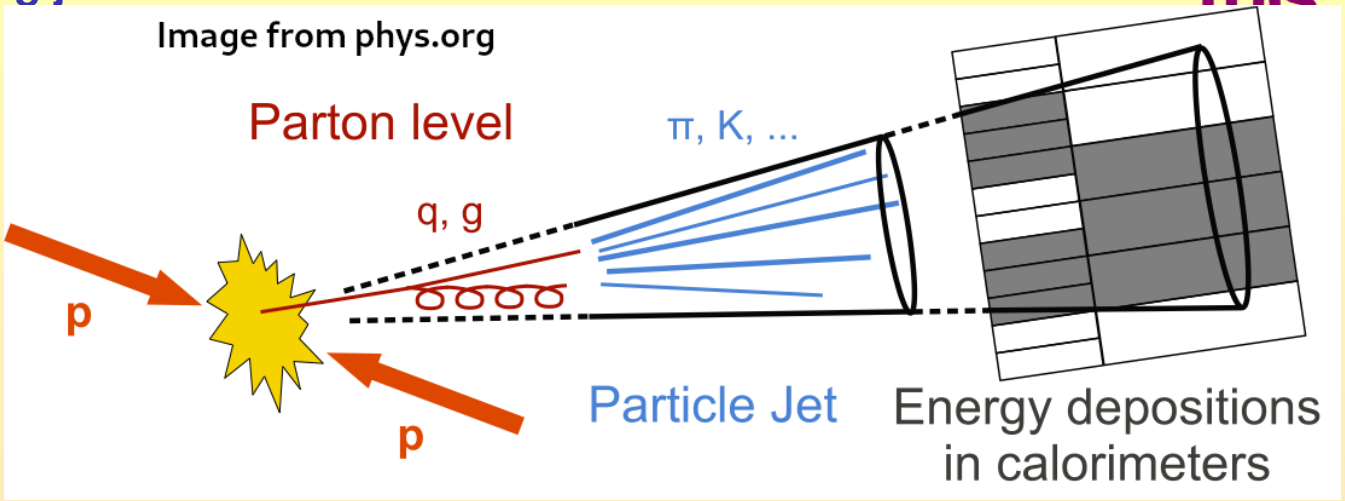


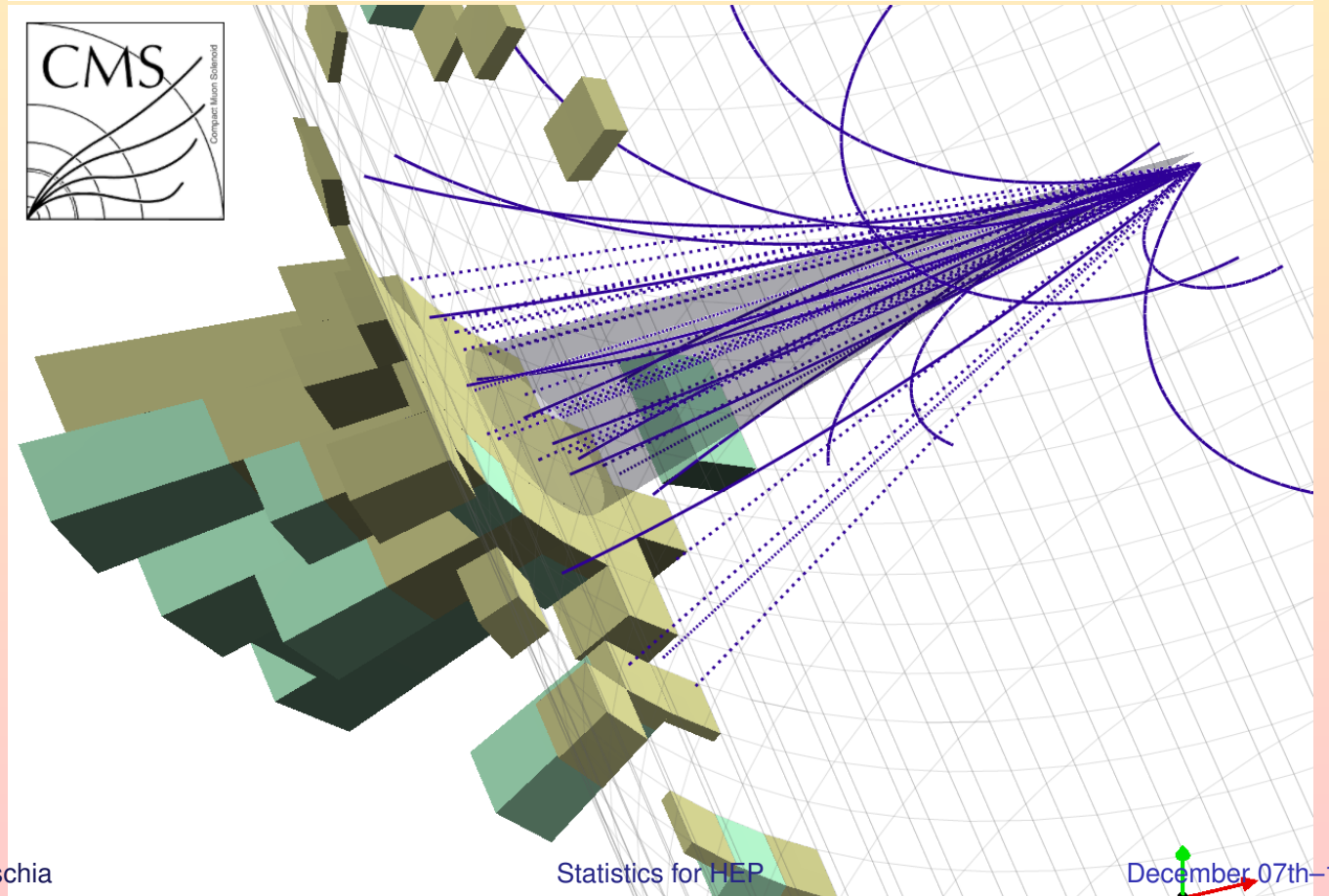
Object Detection



Instance Segmentation

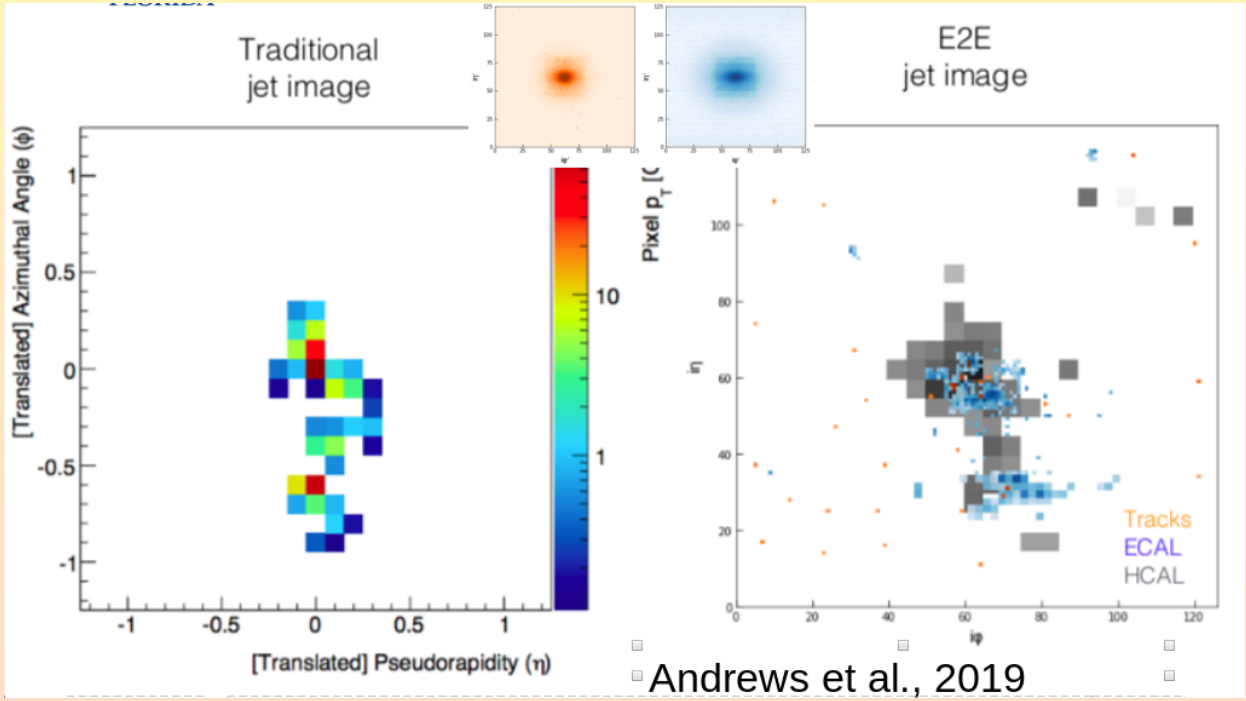
Reconstructing jets



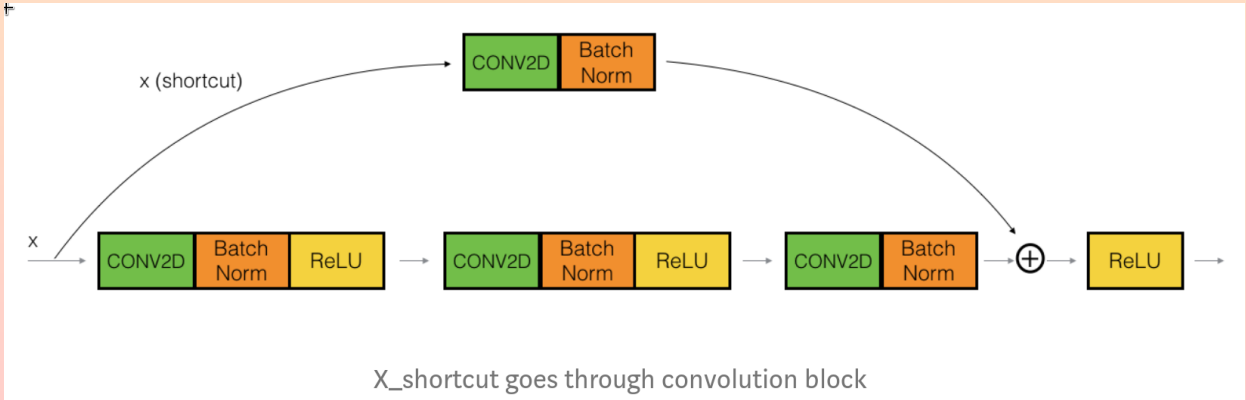


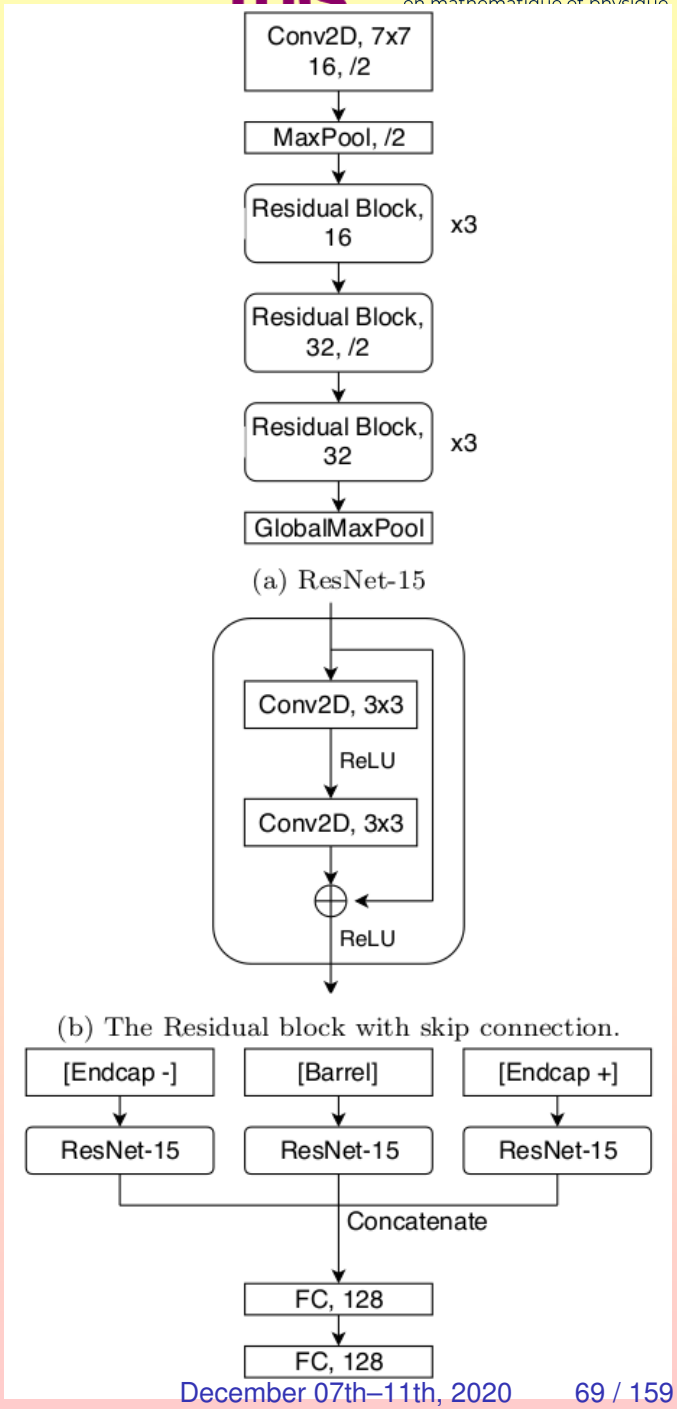
End-to-end jet reconstruction

- Build images by projecting different layers into a single one
- Treat the result as an image with Res(idual)Net(works)
- Role of tracks in jet reco from network matches physics we know



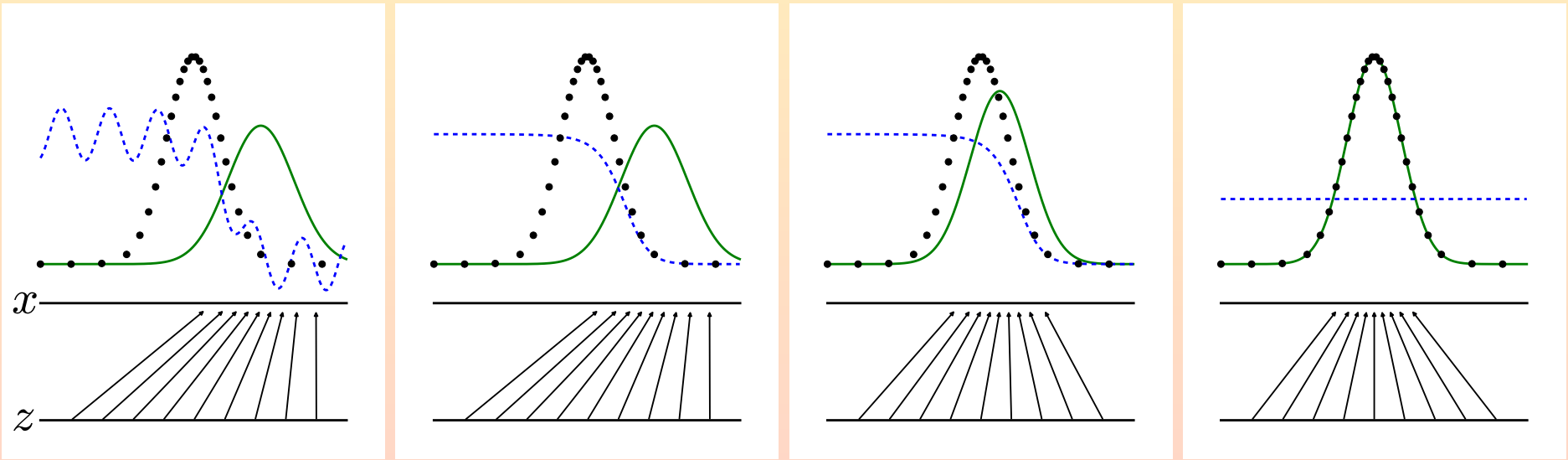
Andrews et al., 2019





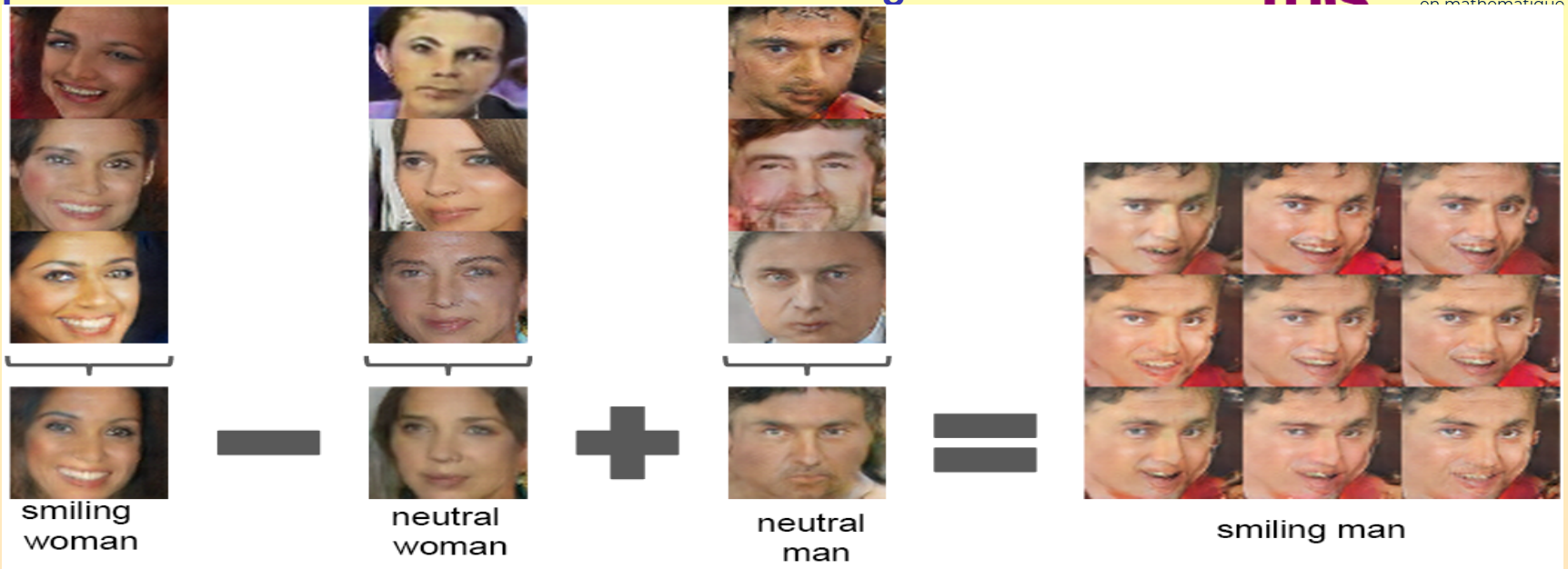
arXiv:1902.08276, S. Gleyzer's talk at 3rd IML workshop, Priya Dwivedi

- Train two networks
- Green network: tries to capture the shape of the data
- Blue network: estimates the probability that an event comes from data rather than the green network
- Strategy: Green tries to fool Blue
  (Javier C. says: Green is Barcelona FC, Blue is Real Madrid)



From https://arxiv.org/abs/1406.2661

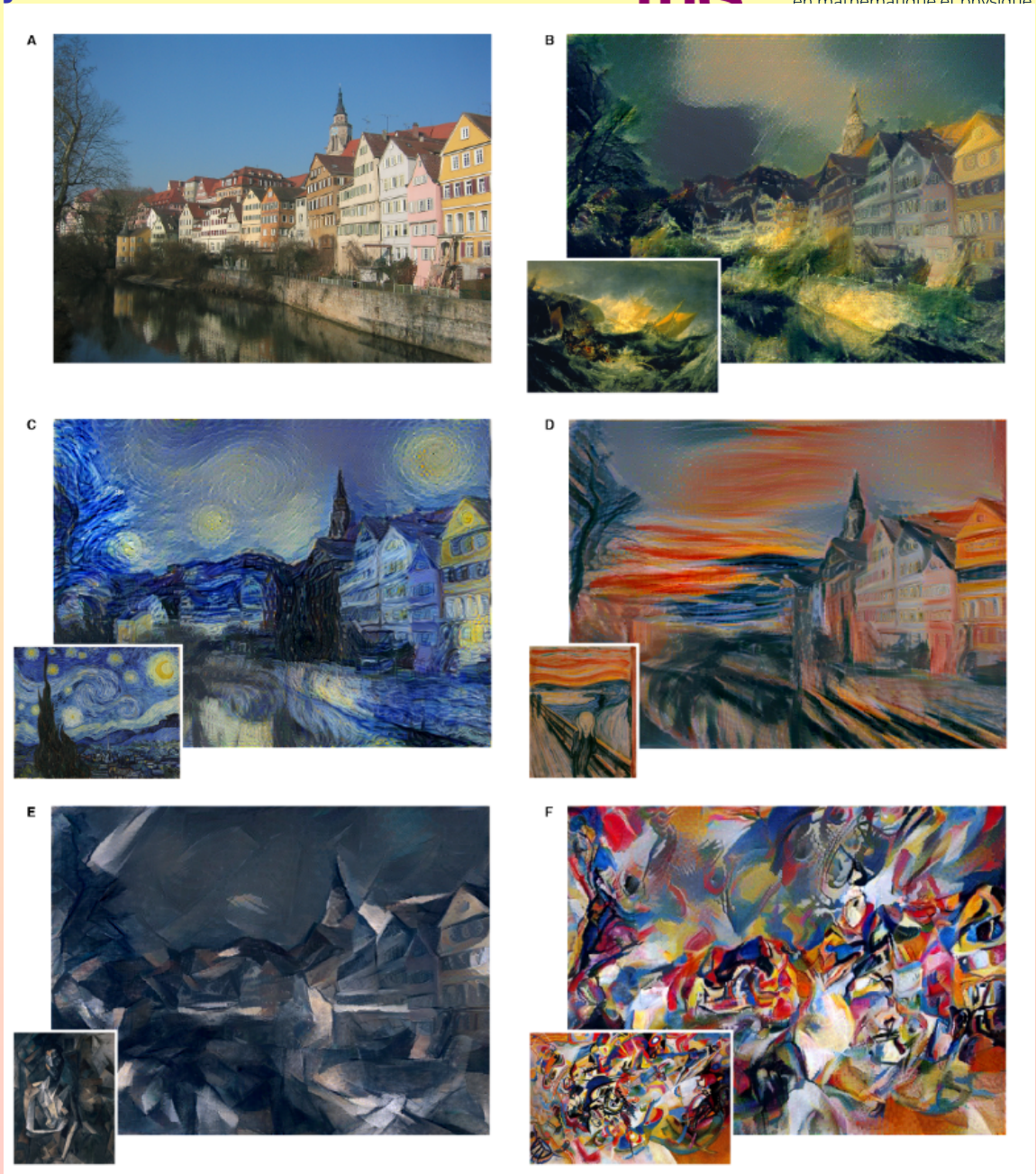
Can pick elements and combine them into new images



From https://github.com/Newmu/dcgan_code/blob/master/images/faces_arithmetic_collage.png

If you can write a loss function for it, you can learn it

- C = mathematical representation of **content**
- S = mathematical representation of **style**
- Loss = distance[S(reference) - S(generated image) + distance[C(original image) - C(generated image)



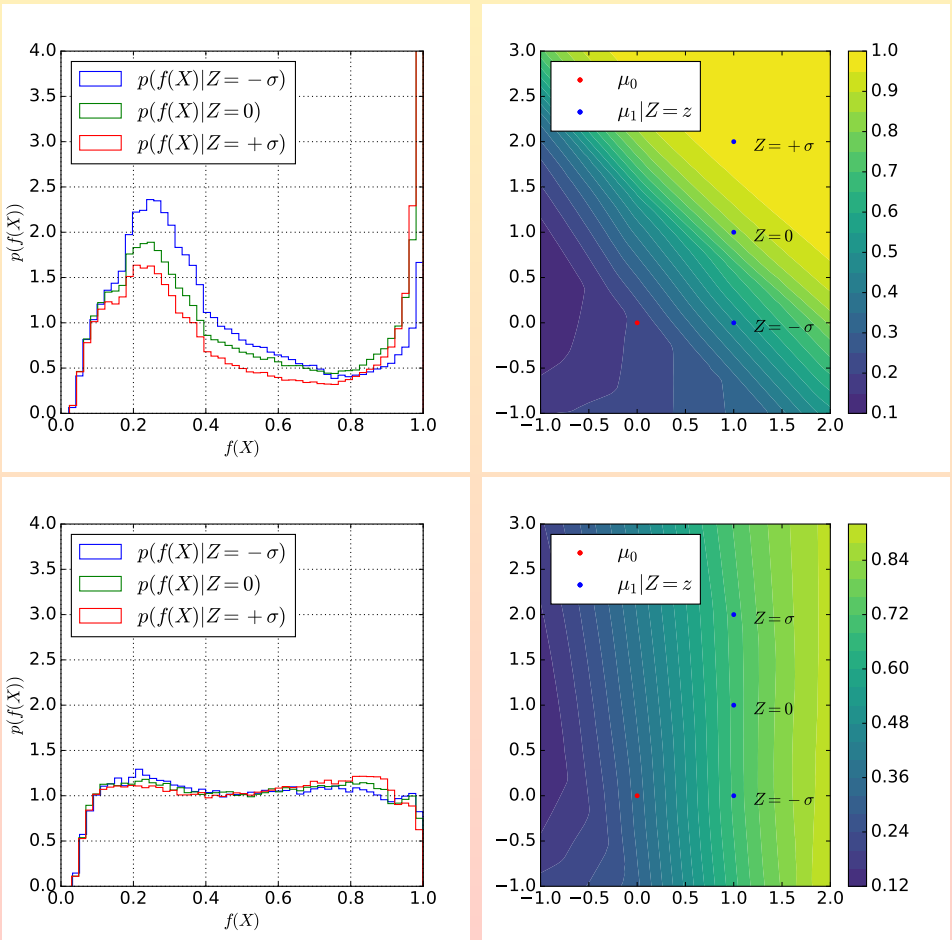
From https://arxiv.org/abs/1508.06576

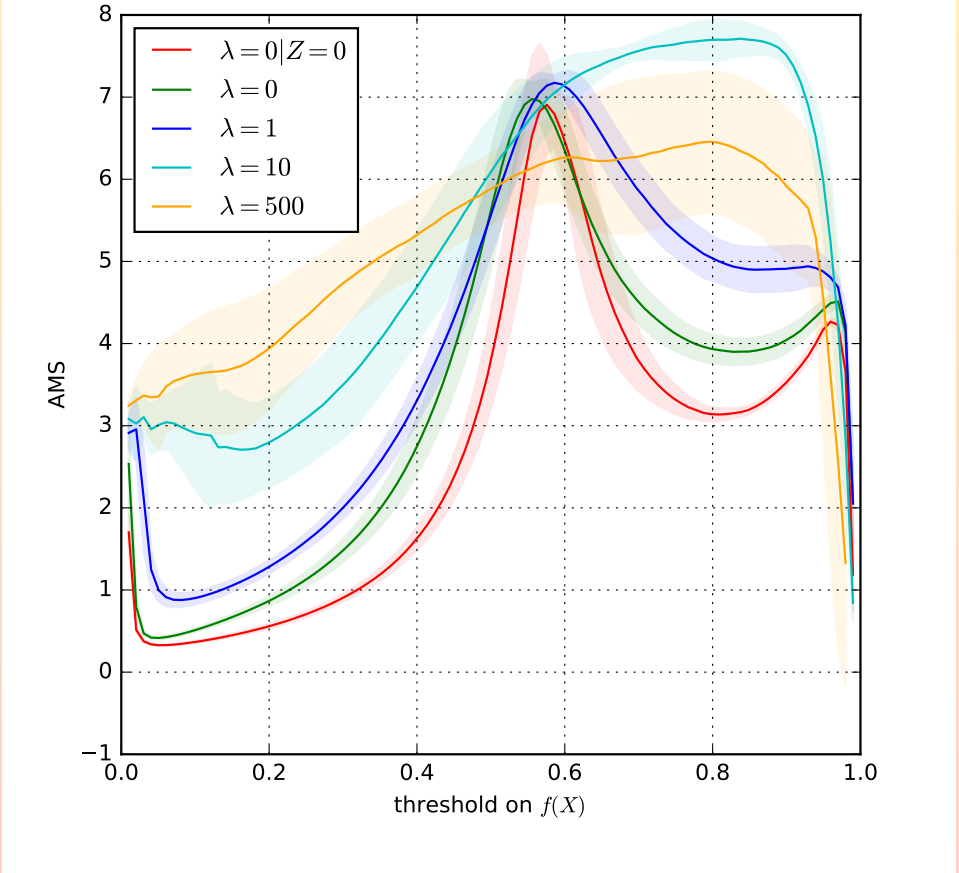
This person does not exist!

From https://thispersondoesnotexist.com/: try it out!

Reduce the impact of systematic uncertainties on our results

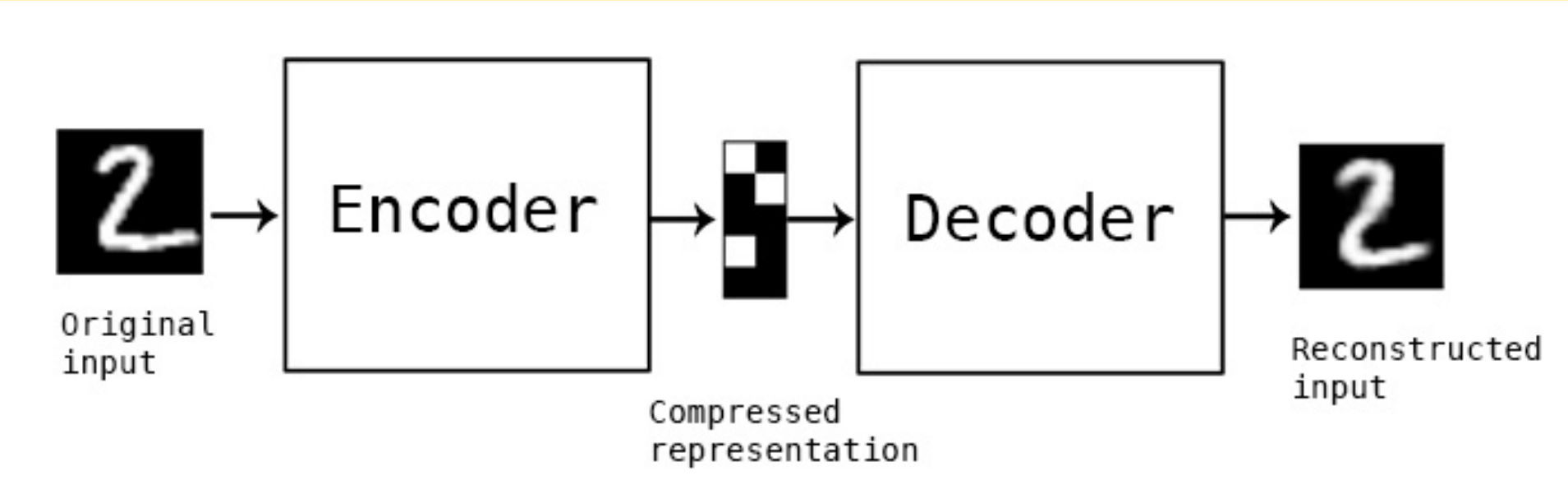
- Adversarial networks used to build pivot quantities
  - Quantities that are invariant in some parameter (typically a nuisance parameter representing a source of uncertainty)
- Best Approximate Mean Significance as tradeoff optimal/pivotal
  $E_\lambda(\theta_f, \theta_r) = \mathcal{L}_f(\theta_f) - \lambda \mathcal{L}_r(\theta_f, \theta_r)$





From Louppe-Kagan-Cranmer, arXiv:1611.01046

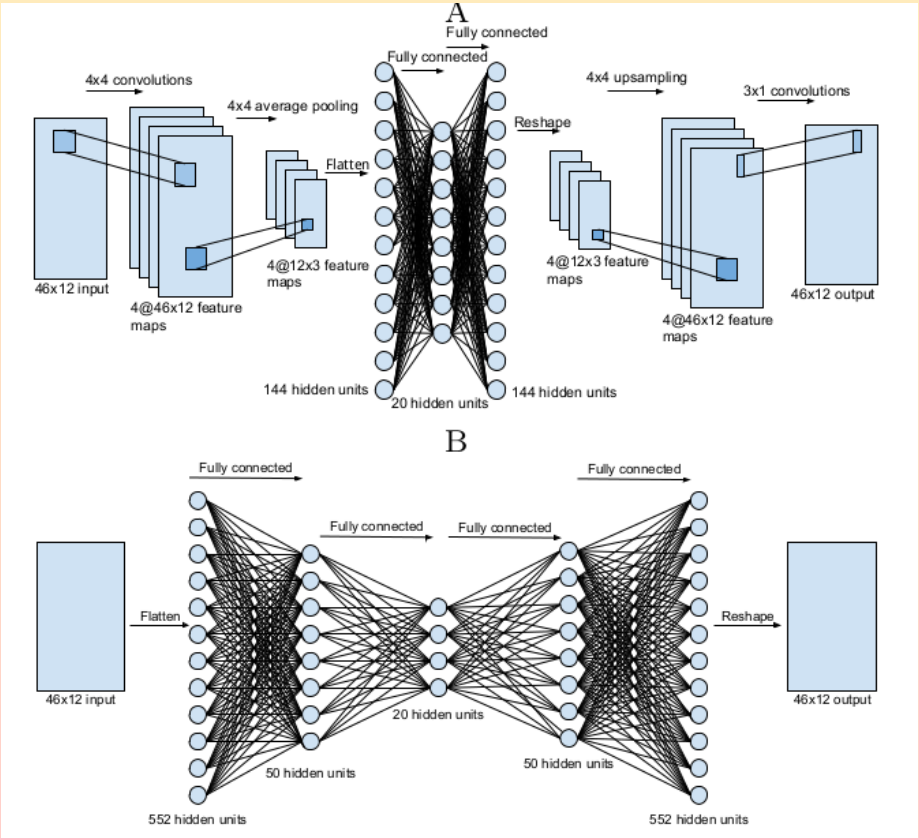
- Learn how to transform an object into almost itself

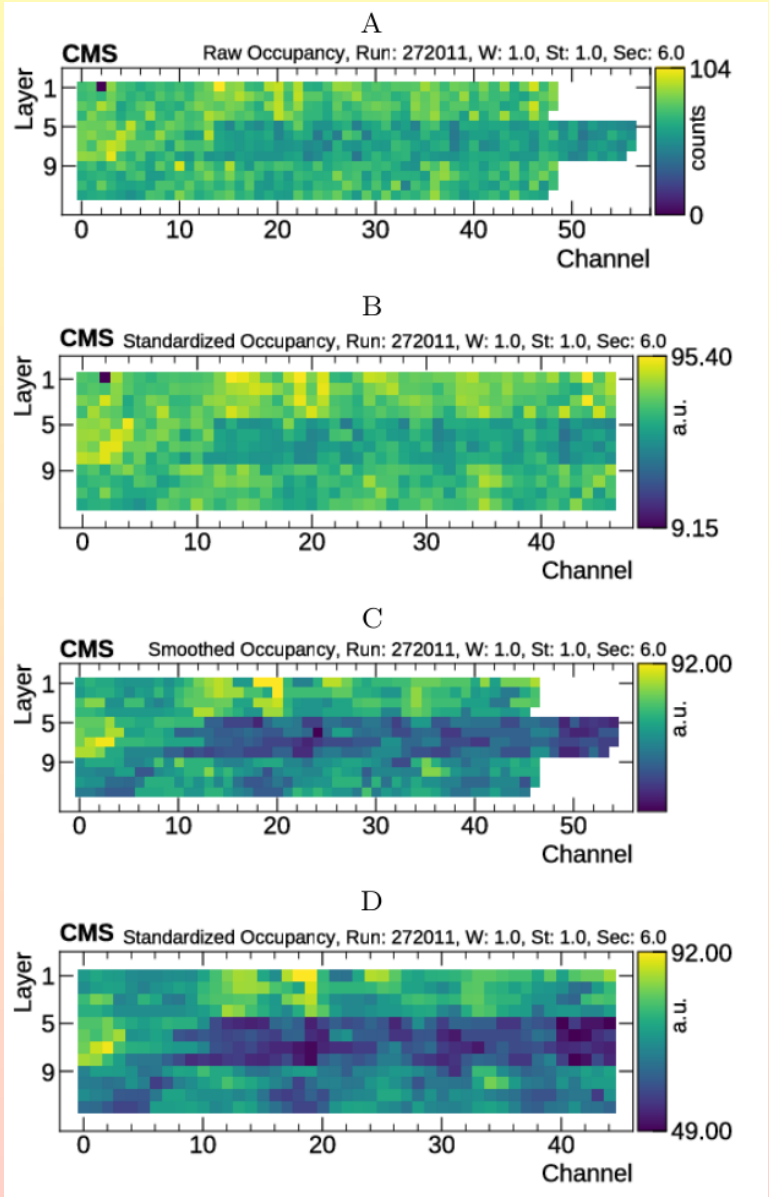


From Chollet and Allaire, Deep Learning With R

...in physics

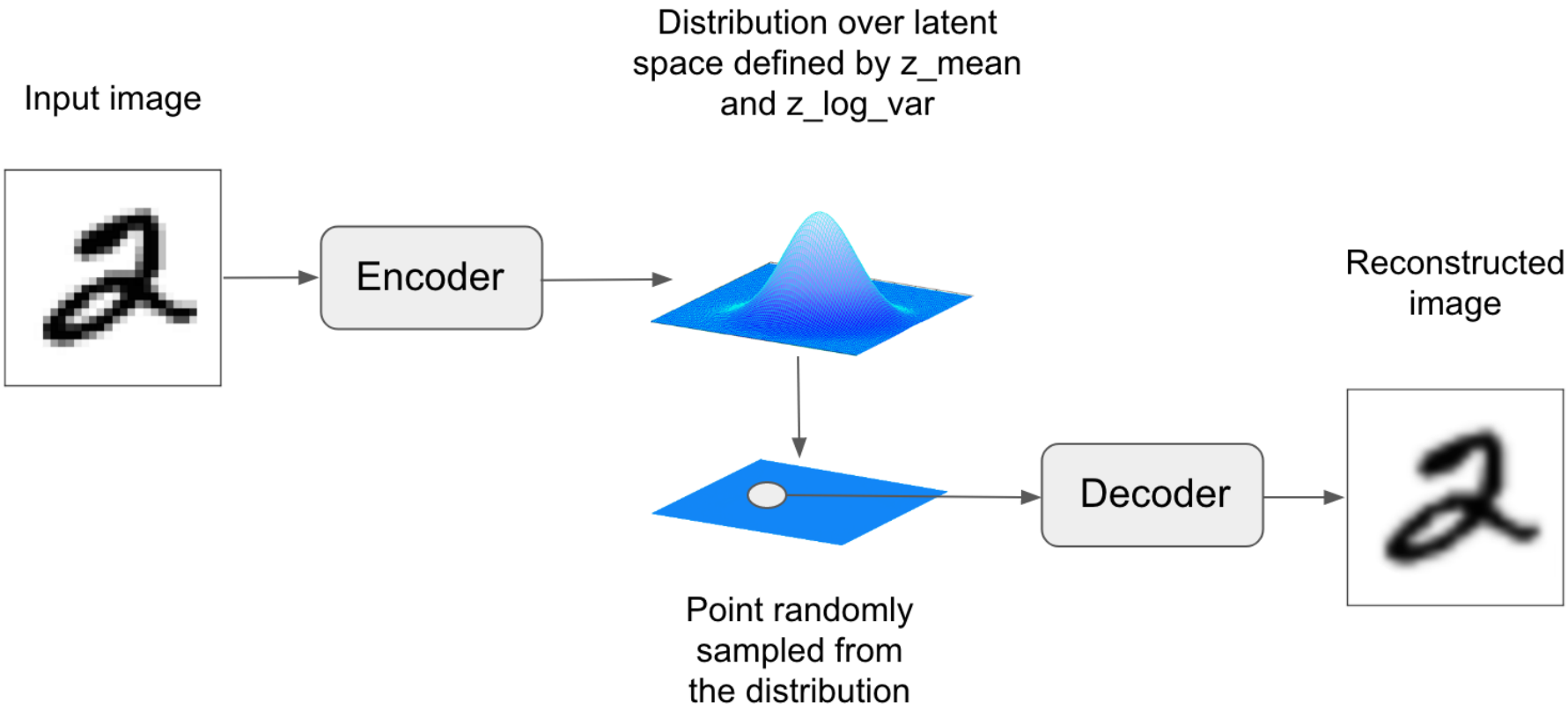
- Use it to spot objects that are different from those you have trained on
- CMS Muon Chamber detectors modelled as geographic layered maps
  - Map is an image: use **convolutional** autoencoders
  - Local approach (independent layers): spot anomalies in a layer
  - Regional approach (simultaneusly across the layers): spot intra-chamber issues





From arXiv:1808.00911

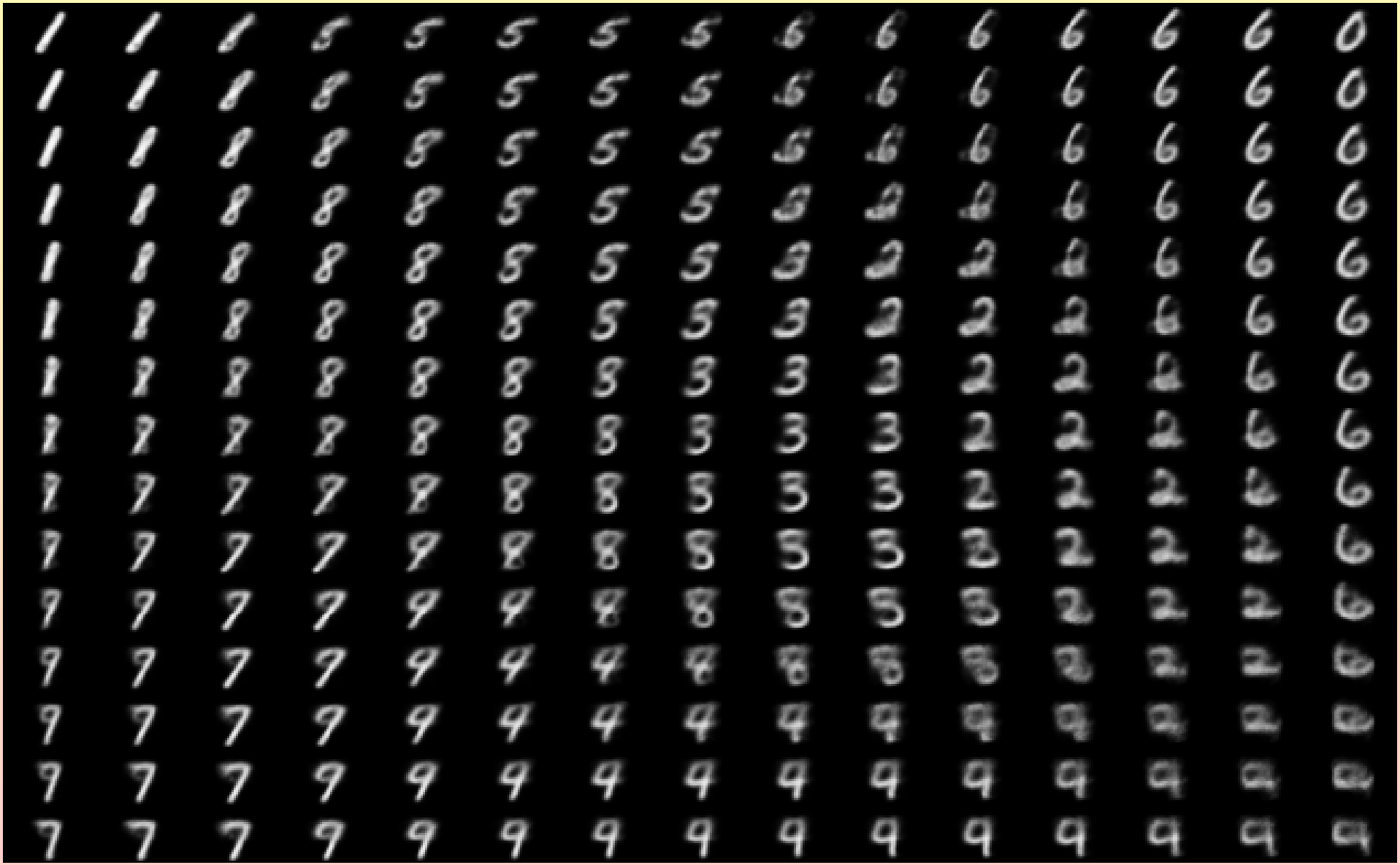
- Learn a space of continous representations of the inputs



From Chollet and Allaire, Deep Learning With R

...and Variational autoencoders

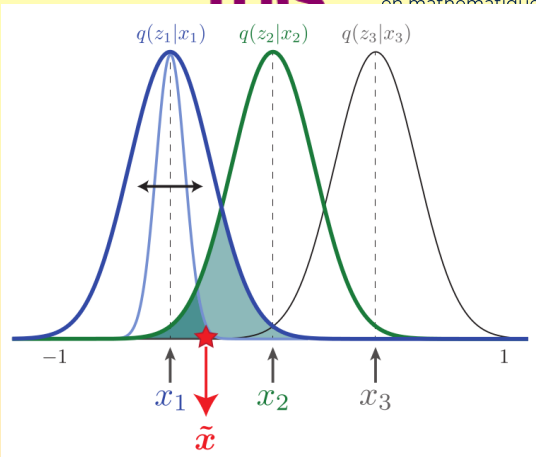
- "How do I transform a 1 into a 0?"
- Space directions have a meaning! "four-ness", "one-ness"

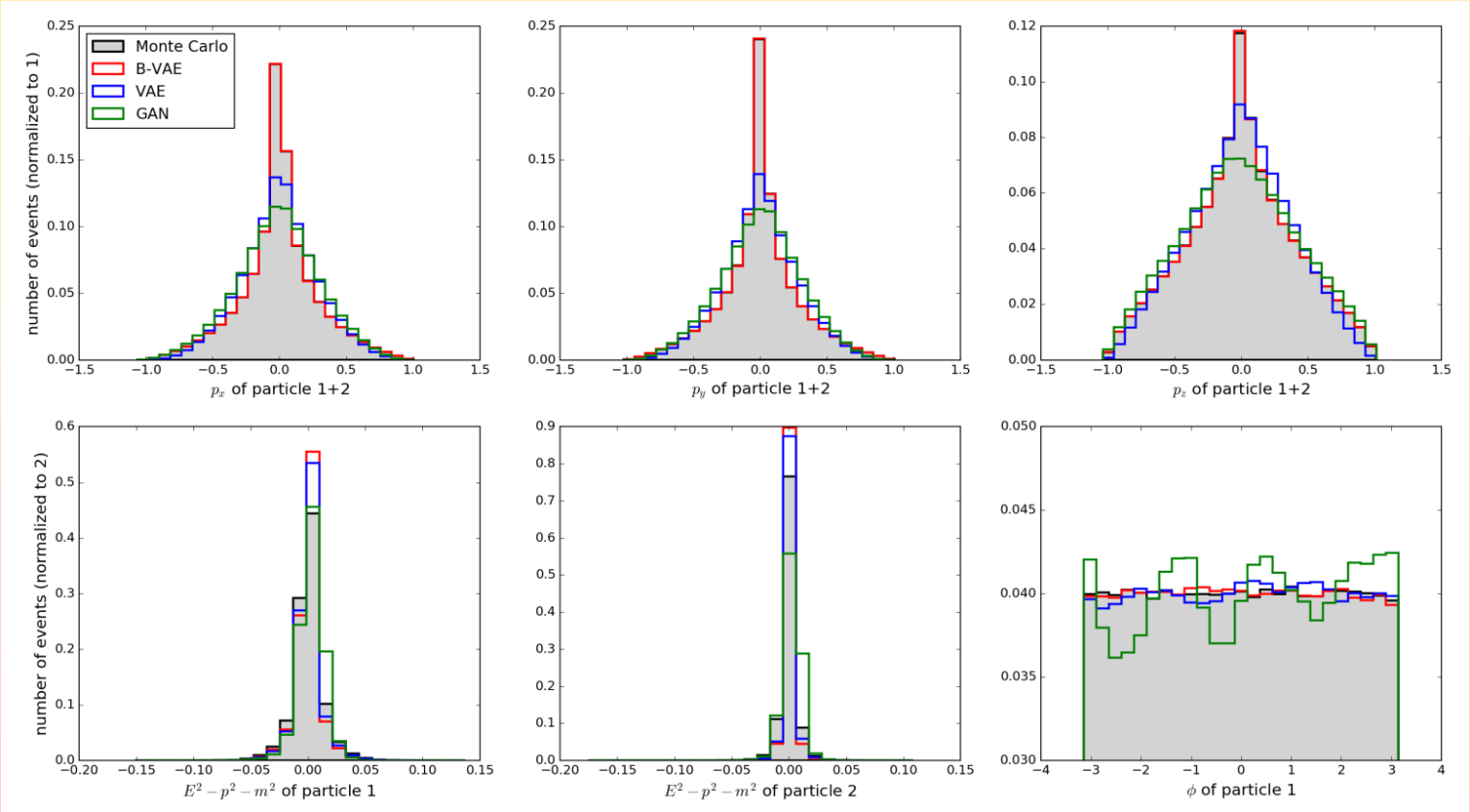


From Chollet and Allaire, Deep Learning With R

...also in physics

- Fast generation of collision events in a multidimensional phase space
- Balancing goodness-of-reconstruction and overlap in latent space
  - B-VAE $Loss = \frac{1}{M}\sum_{i=1}^{M}(1-B)\cdot MSE + B\cdot D_{KL}$.
- Works better than a GAN!



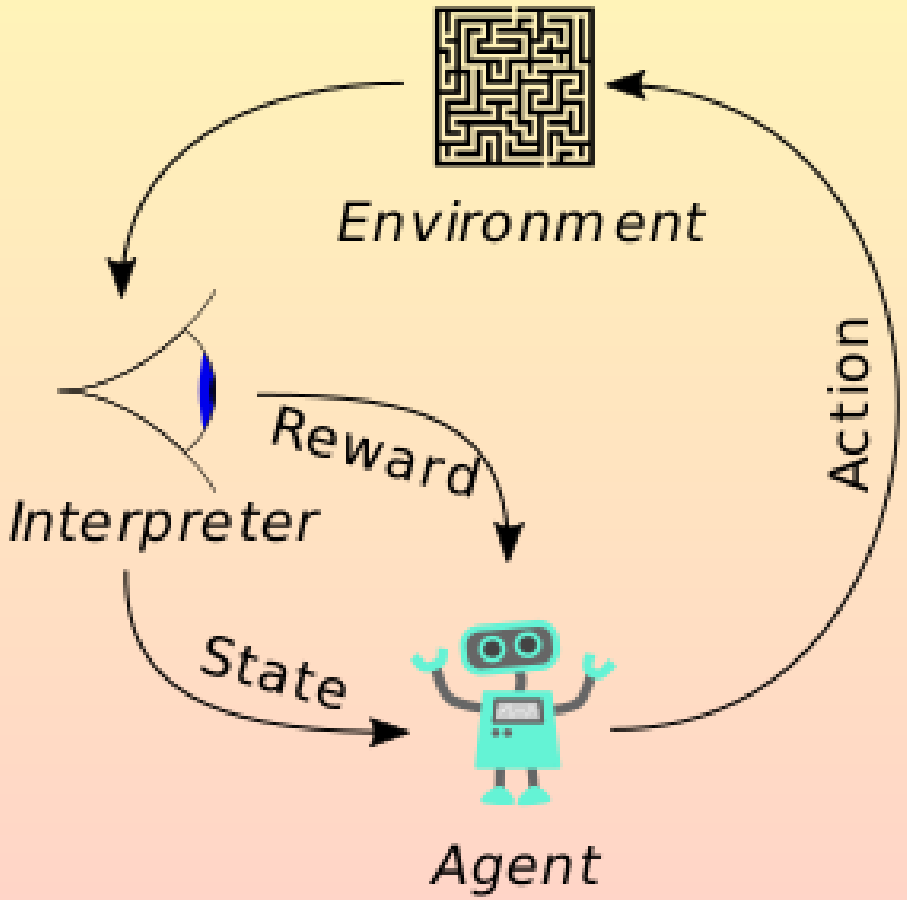


Plots from arXiv:1804.03599 and arXiv:1901.00875

We figure out images

- What about adding a time component?
- A single network is not complex enough for driving a car
- What if we permit a network to modify itself?

- Reinforcement Learning
- “Q” is the letter denoting the reward function for an action



By Megajuice - Own work, CC0, https://commons.wikimedia.org/w/index.php?curid=57895741

...is what you do to train your pets

From The Auckland Dog Coach

From videogames...

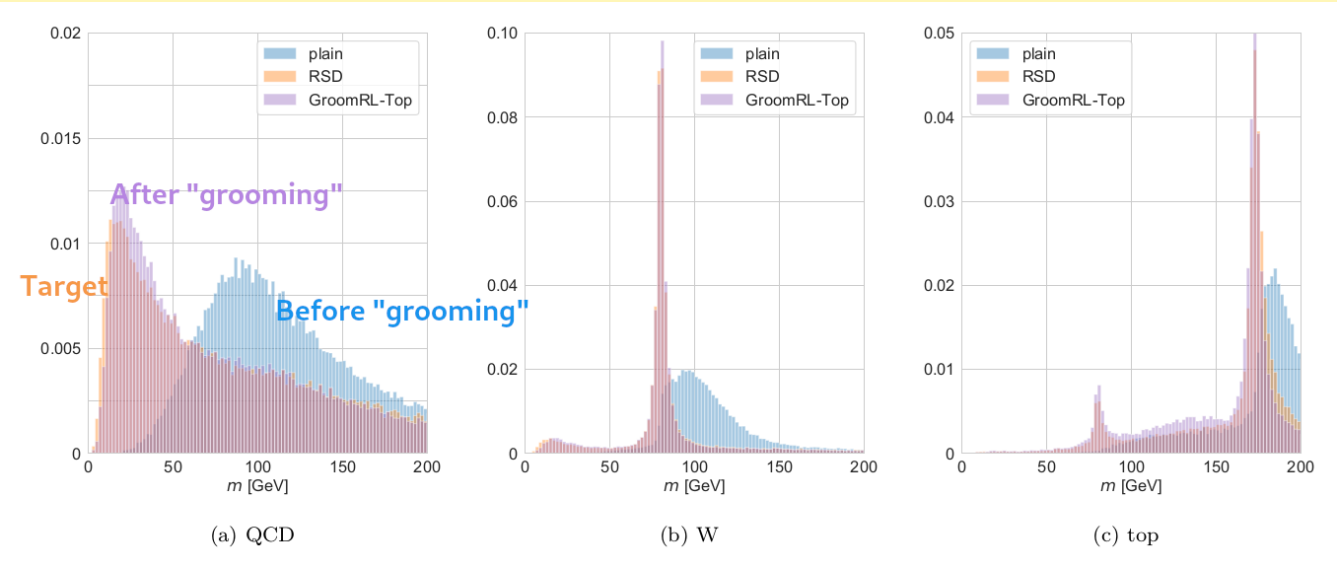
- ATARI Blackout (Google Deep Mind)
- *https://deepmind.com/research/dqn/*

...to self driving cars...

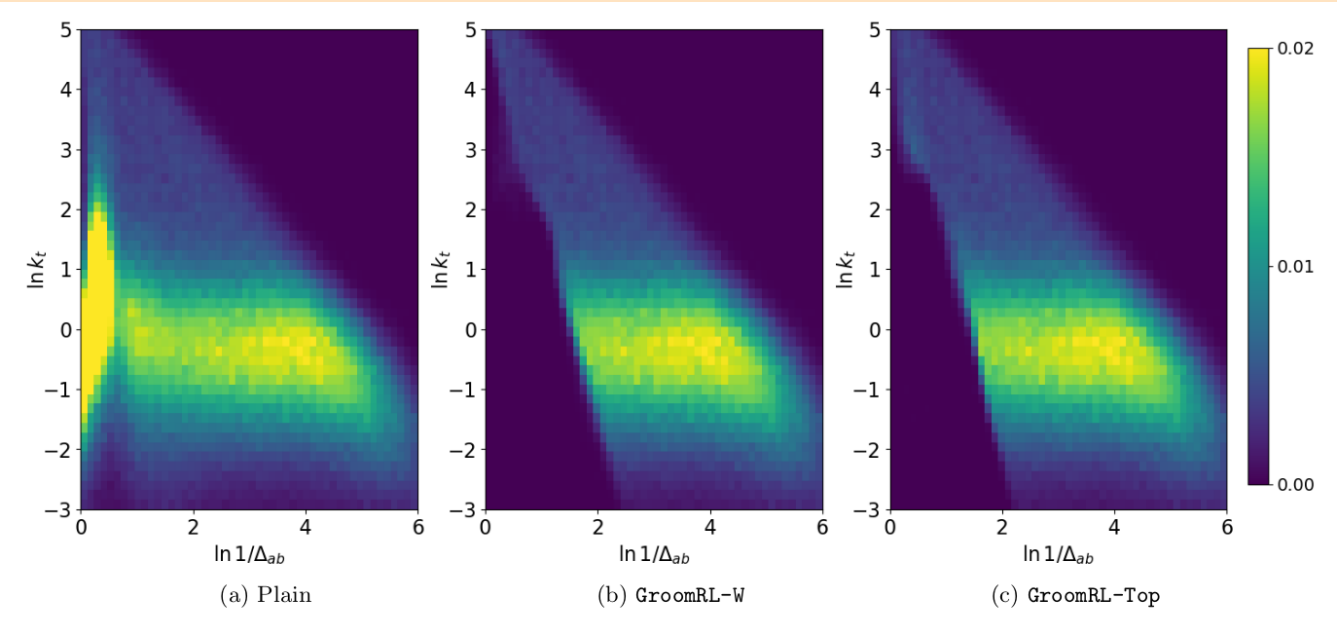
- *https://www.youtube.com/watch?v=MqUbdd7ae54*
- Build your own simulated driver: http://selfdrivingcars.mit.edu/deeptraffic/

...to physics

- Boosted objects decay to collimated jets reconstructed as single fat jet
- Fat jet grooming: remove soft wide-angle radiation not associated with the underlying hard substructure



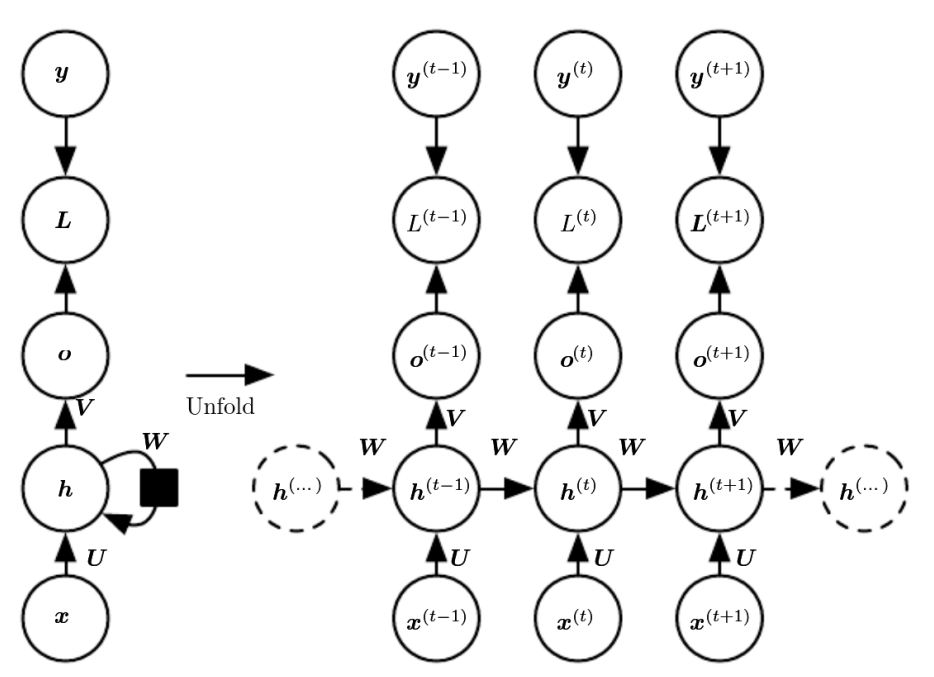
(a) QCD (b) W (c) top



(a) Plain (b) GroomRL-W (c) GroomRL-Top

Learn sequences

- Recurrent architectures insert a “time” component: learn sequences!
  - In general a dimension that is supposed to be ordered (time, position of words in a sentence, etc)
- Can even learn how to generate Shakespearian text
  - With Markov Chains, the results are rather worse: https://amva4newphysics.wordpress.com/2016/09/20/hermione-had-become-a-bit-pink/

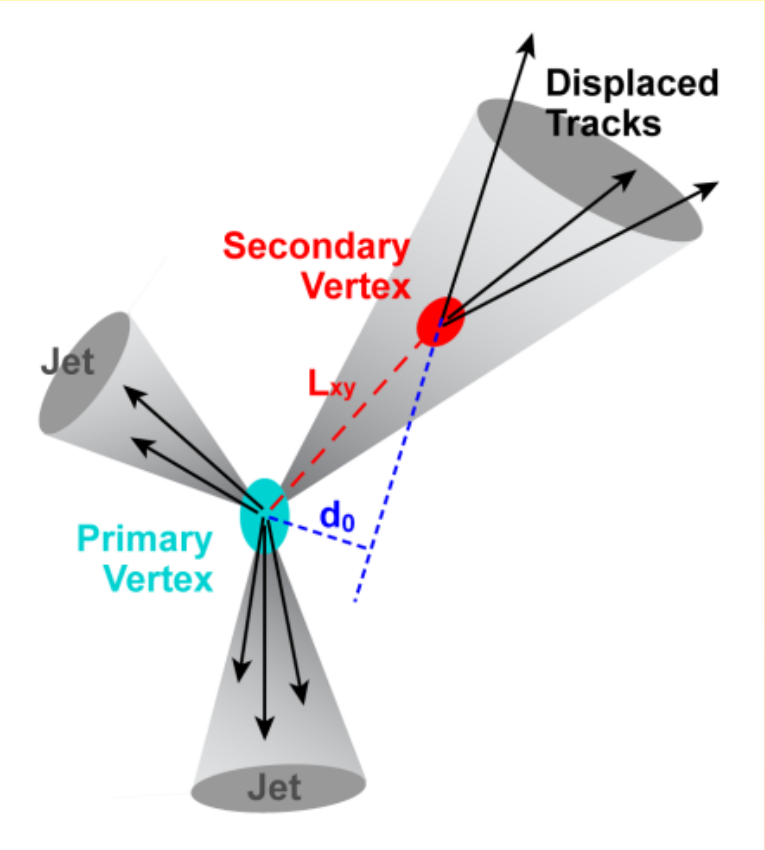



```
QUEENE:
I had thought thou hadst a Roman; for the oracle,
Thus by All bids the man against the word,
Which are so weak of care, by old care done;
Your children were in your holy love,
And the precipitation through the bleeding throne.

BISHOP OF ELY:
Marry, and will, my lord, to weep in such a one were prettiest;
Yet now I was adopted heir
Of the world's lamentable day,
To watch the next way with his father with his face?

ESCALUS:
The cause why then we are all resolved more sons.
```

From https://www.deeplearningbook.org/ and https://www.tensorflow.org/tutorials/text/text_generation

Identifying jets from b quarks...

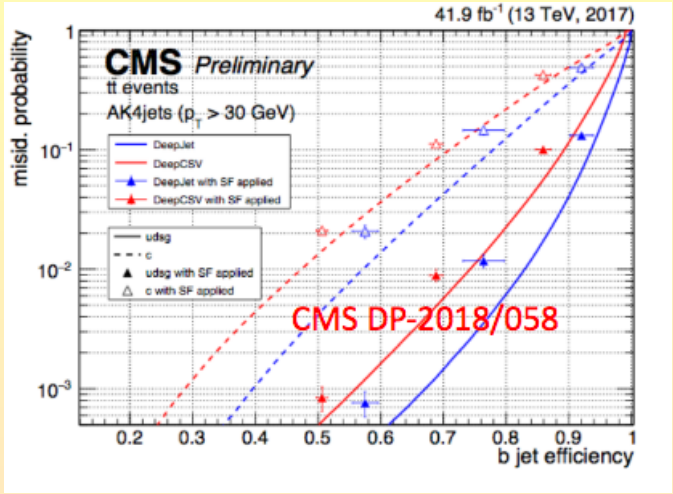
- Quarks produced in proton-proton collisions give rise to collimated “jets” of particles
- Bottom quarks travel for a while before fragmenting into jets

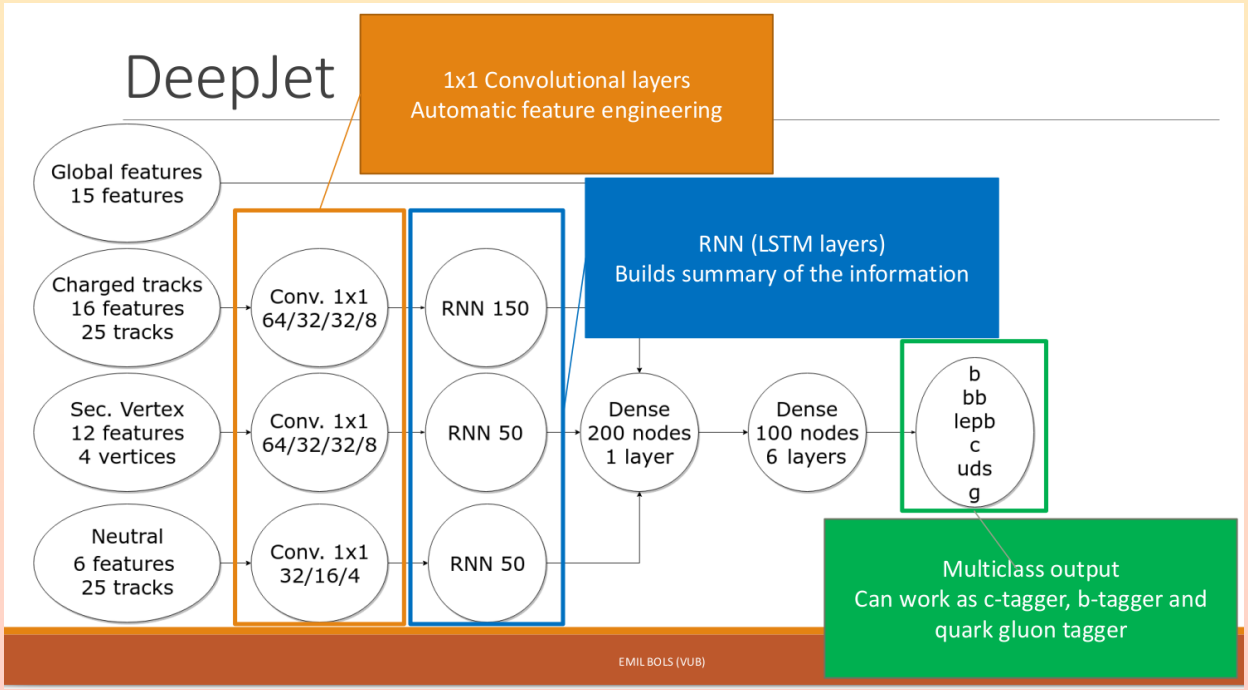


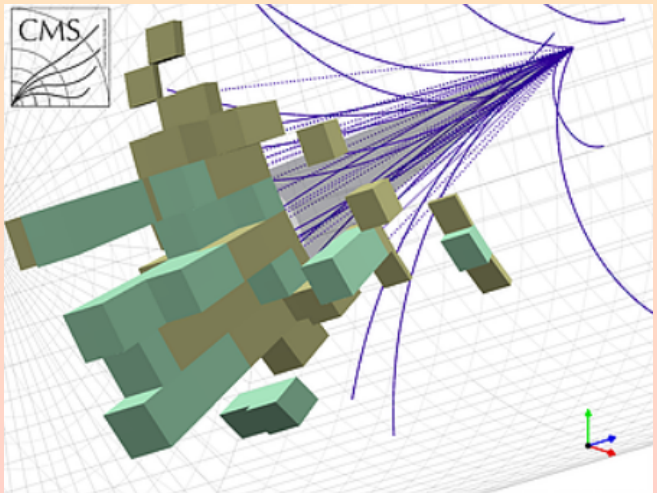
Plot from D0

...requires combining image and sequential processing!

- b tagging at CMS
  - CSV (Run I and early Run II): BDT sensitive to secondary vertexes
- DeepCSV: similar inputs, generic DNN
- Domain knowledge can inform the representation used!
  - Leading criterion for choice of technique for the classifier
- What is the best representation for jets?
  - Convolutional networks for images
  - Particle-based structure





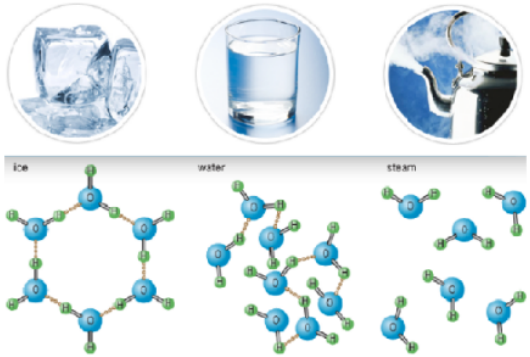


CMS DeepJet, plot from Emil Bols' talk at IML workshop

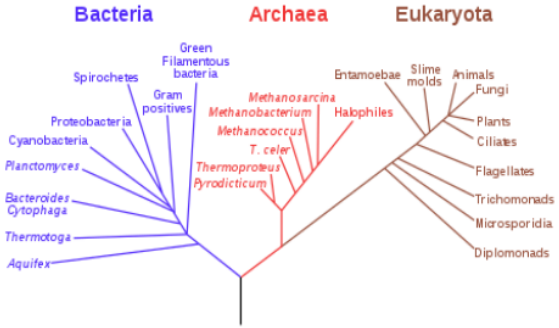
Learn graph of the underlying structure...

UCLouvain
Institut de recherche
en mathématique et physique

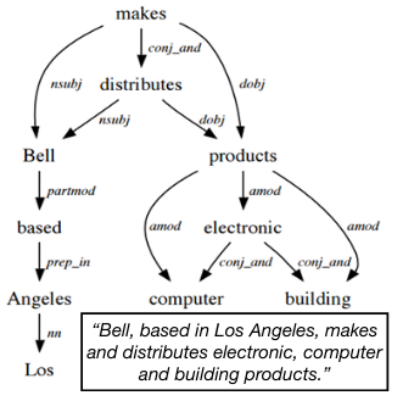
Molecules



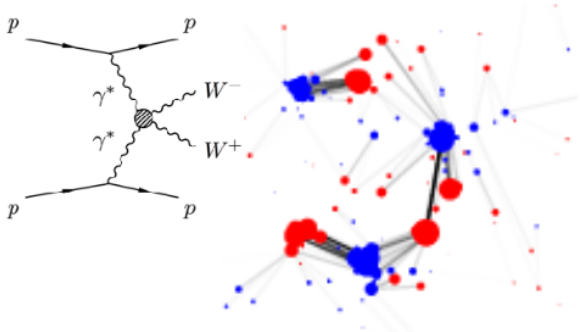
Biological species



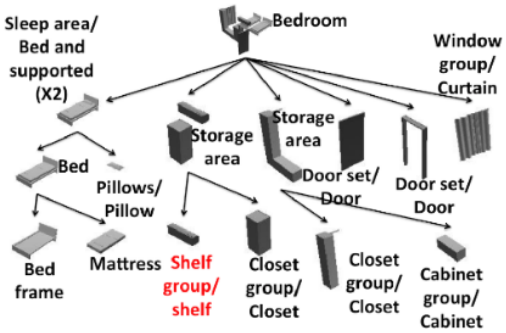
Natural language



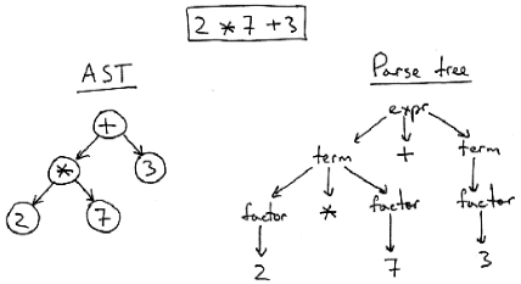
Sub-atomic particles



Everyday scenes



Code



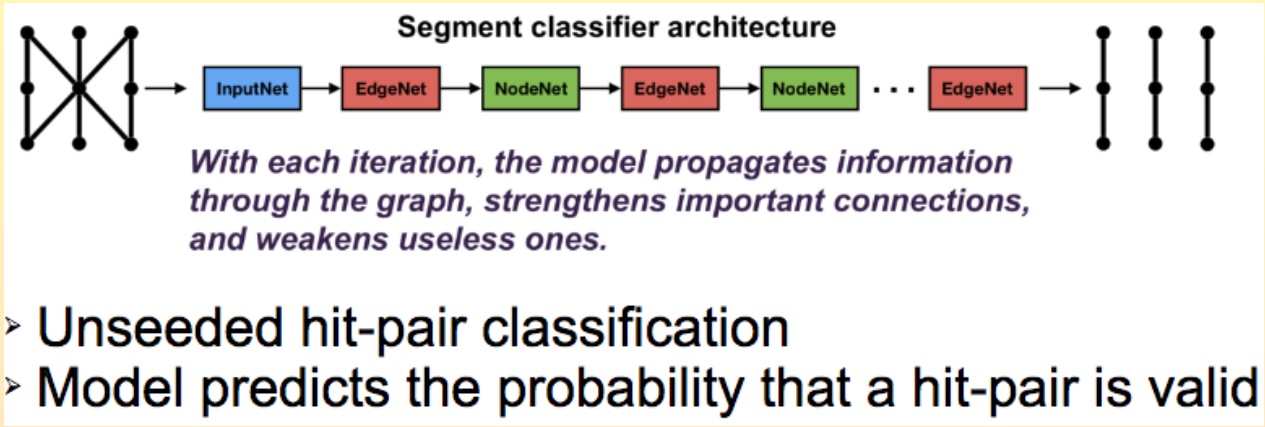
From Peter Battaglia's talk at the IML2020 Workshop

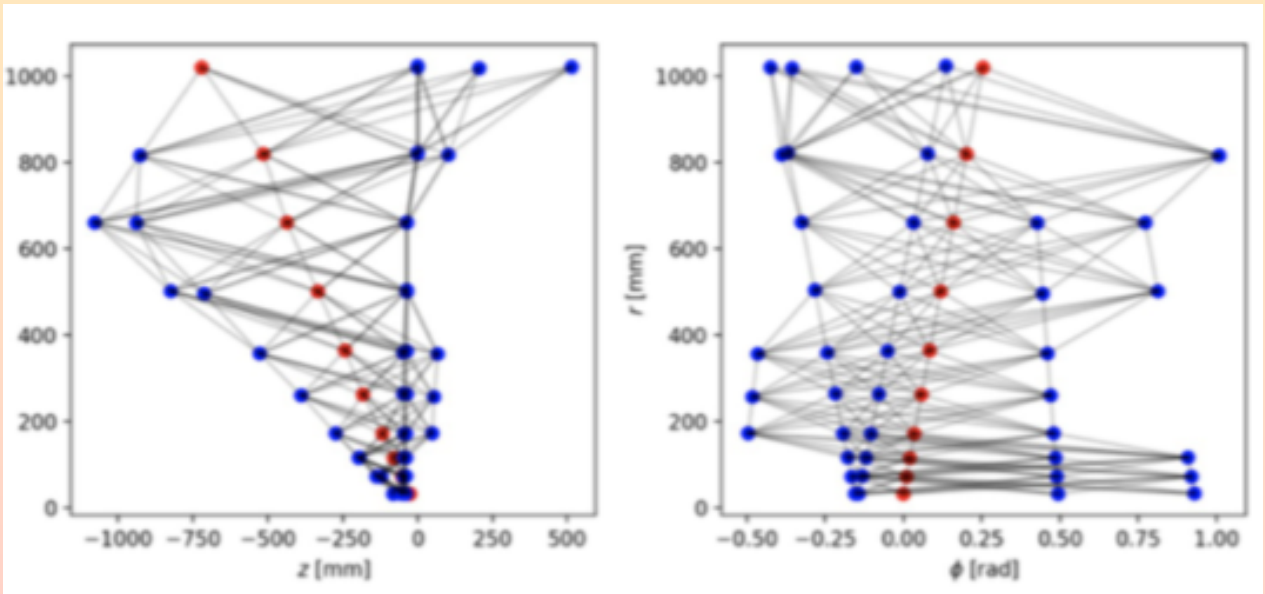
...until the structure is learned

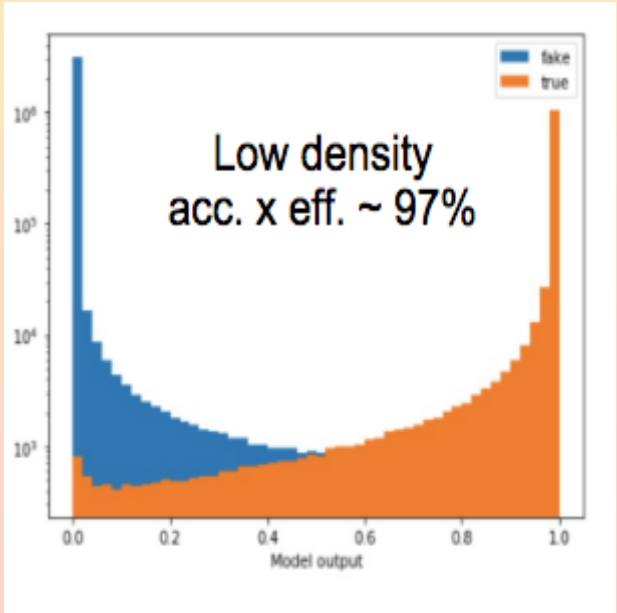
Water

Video from https://sites.google.com/view/learning-to-simulate

- Graph networks to literally connect the dots





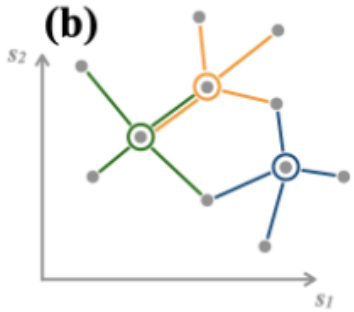


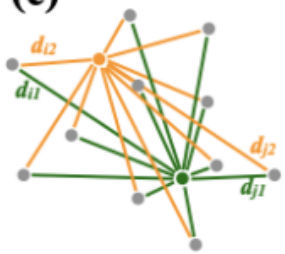
The HEP.TrkX project, S. Gleyzer's talk at 3rd IML workshop

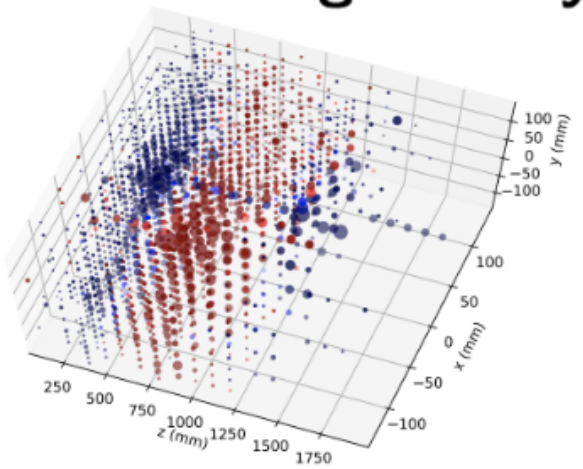
High-granularity calorimeter

- 600m^2 of sensors, 50 layers: 6 million cells with $\sim$3mm spatial resolution
  - Some square cells, some exagonal cells
  - Non-projective geometry

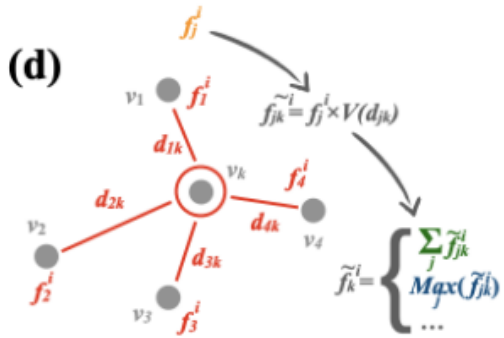
Learning representations of irregular particle-detector geometry with distance-weighted graph networks







(a) Truth



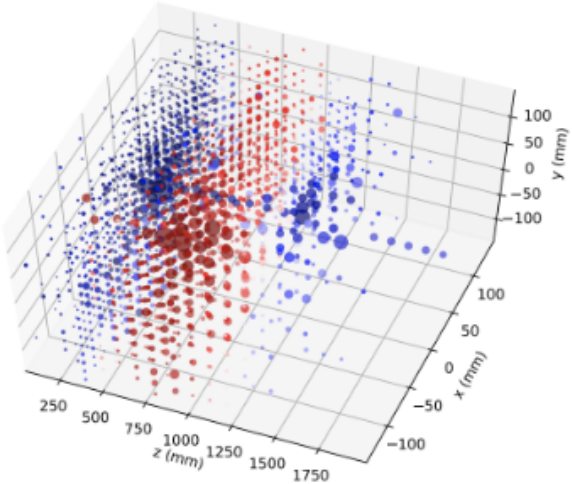


Image from a talk by André David and the HGCAL team

- GPT3: autoregressive model with 175 billion parameters
  - Non-recurrent, attention-based (non-fixed-length sequences)
  - Standard RNN-based autoencoders have problems due to fixed-length (different languages have different information density)

Title: United Methodists Agree to Historic Split
Subtitle: Those who oppose gay marriage will form their own denomination
Article: **After two days of intense debate, the United Methodist Church has agreed to a historic split - one that is expected to end in the creation of a new denomination, one that will be "theologically and socially conservative," according to The Washington Post. The majority of delegates attending the church's annual General Conference in May voted to strengthen a ban on the ordination of LGBTQ clergy and to write new rules that will "discipline" clergy who officiate at same-sex weddings. But those who opposed these measures have a new plan: They say they will form a separate denomination by 2020, calling their church the Christian Methodist denomination.**
The Post notes that the denomination, which claims 12.5 million members, was in the early 20th century the "largest Protestant denomination in the U.S.," but that it has been shrinking in recent decades. The new split will be the second in the church's history. The first occurred in 1968, when roughly 10 percent of the denomination left to form the Evangelical United Brethren Church. The Post notes that the proposed split "comes at a critical time for the church, which has been losing members for years," which has been "pushed toward the brink of a schism over the role of LGBTQ people in the church." Gay marriage is not the only issue that has divided the church. In 2016, the denomination was split over ordination of transgender clergy, with the North Pacific regional conference voting to ban them from serving as clergy, and the South Pacific regional conference voting to allow them.

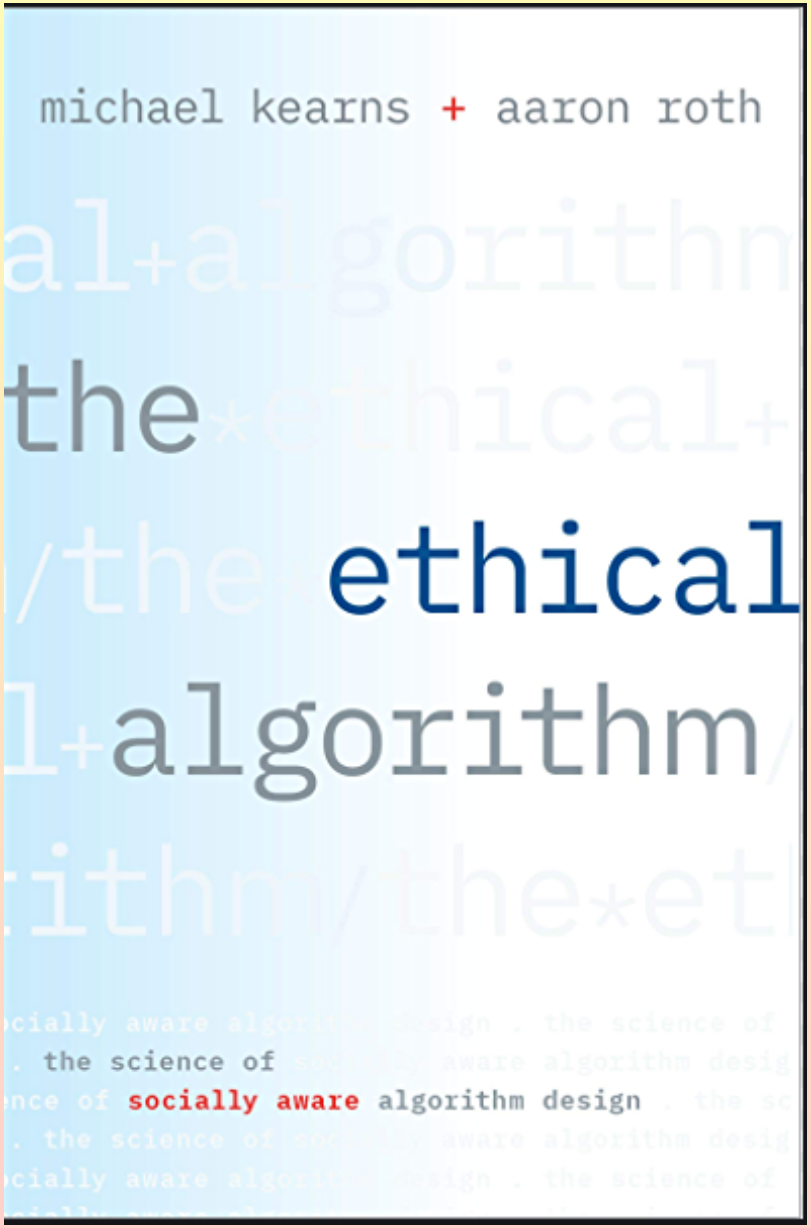
Figure 3.14: The GPT-3 generated news article that humans had the greatest difficulty distinguishing from a human written article (accuracy: 12%).

From https://arxiv.org/abs/2005.14165

HUGE potential for misuse

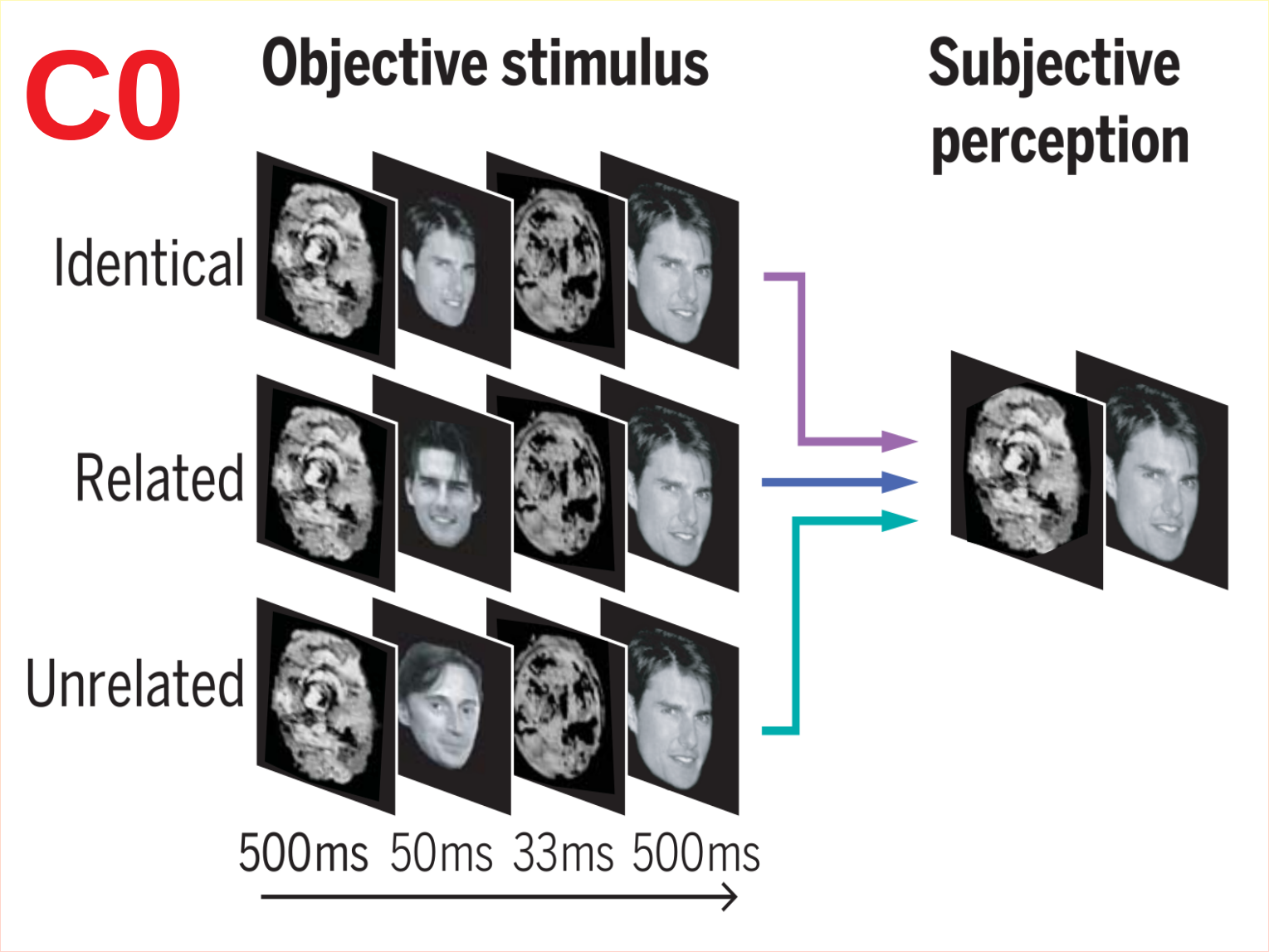
- Is an image real or fake?
- Is a video real or fake?
- Is a text real or fake?
- If a self-driving car kills someone, who's fault is that?
- You can be tracked anywhere
- Your behaviour can be modelled and exploited

HUGE potential for misuse



The Problem of Intelligence

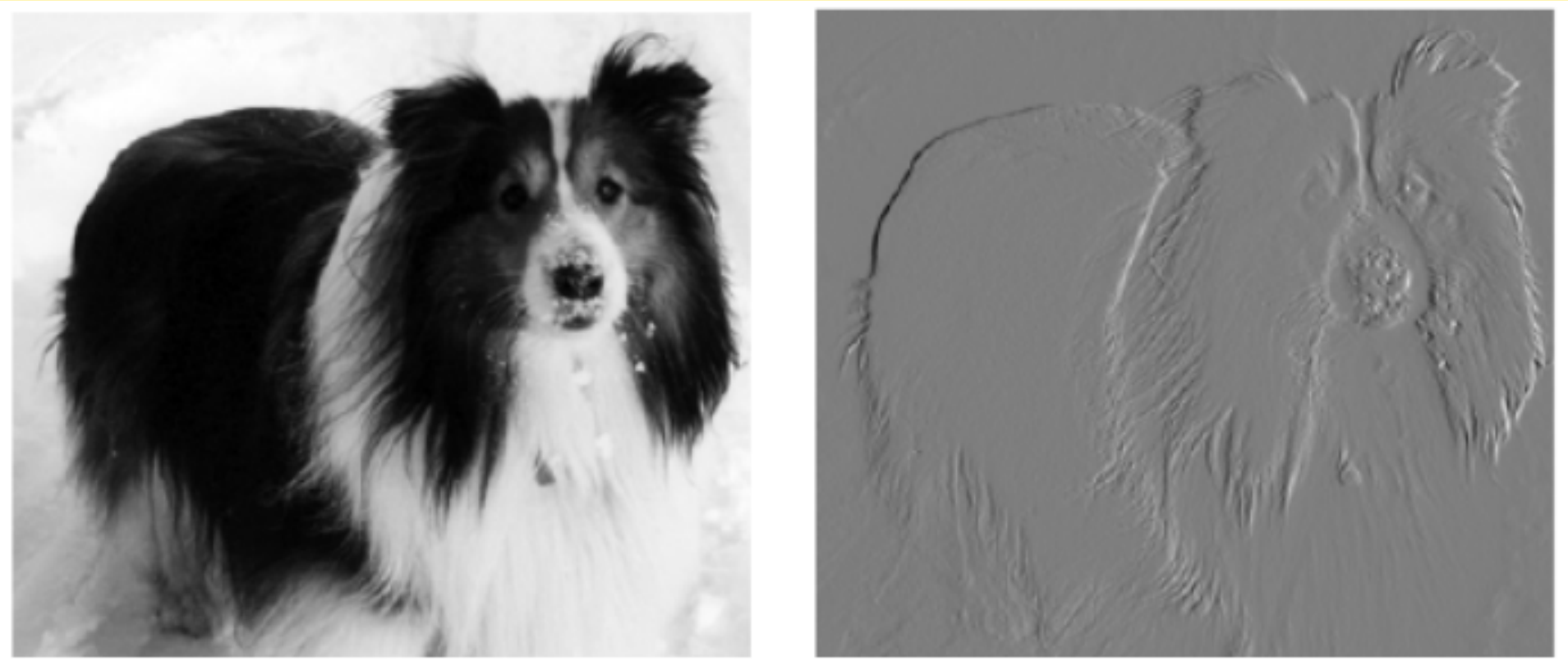
UCLouvain
Institut de recherche
en mathématique et physique



UCLouvain
Institut de recherche
en mathématique et physique

Information about self

We thought we knew how convolutional networks learn images...



Images from https://www.deeplearningbook.org/

$+ .007 \times$

$=$

$\boldsymbol{x}$	$\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$	$\boldsymbol{x} + \epsilon\,\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$
$y =$"panda"	"nematode"	"gibbon"
w/ 57.7%	w/ 8.2%	w/ 99.3 %
confidence	confidence	confidence

Images from https://www.deeplearningbook.org/

...not even in text generation

Q: If m x q changes to q x m , what does p a b m change to?
A: m b a p

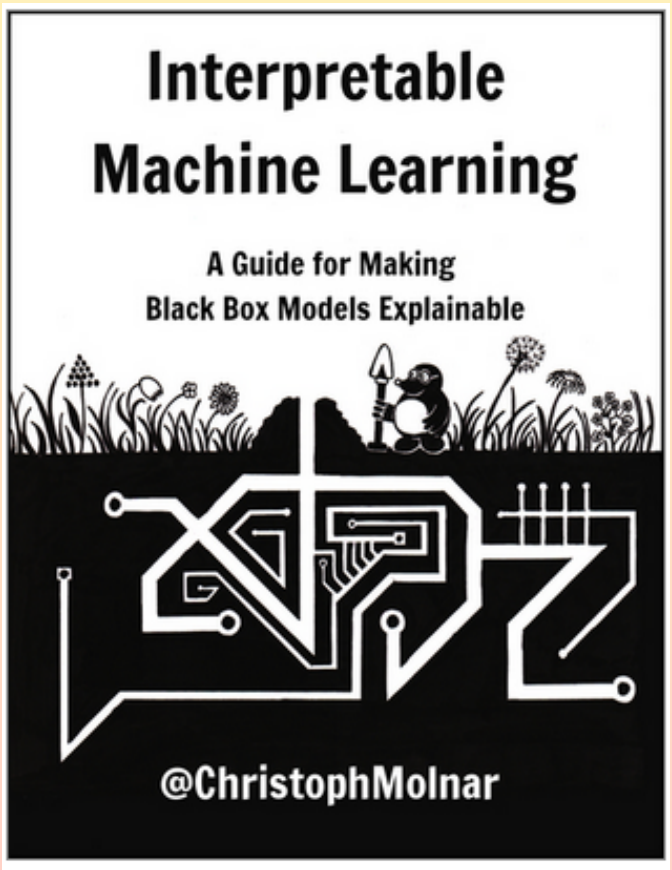
Q: If m x q changes to q x m , what does y r q l v change to?

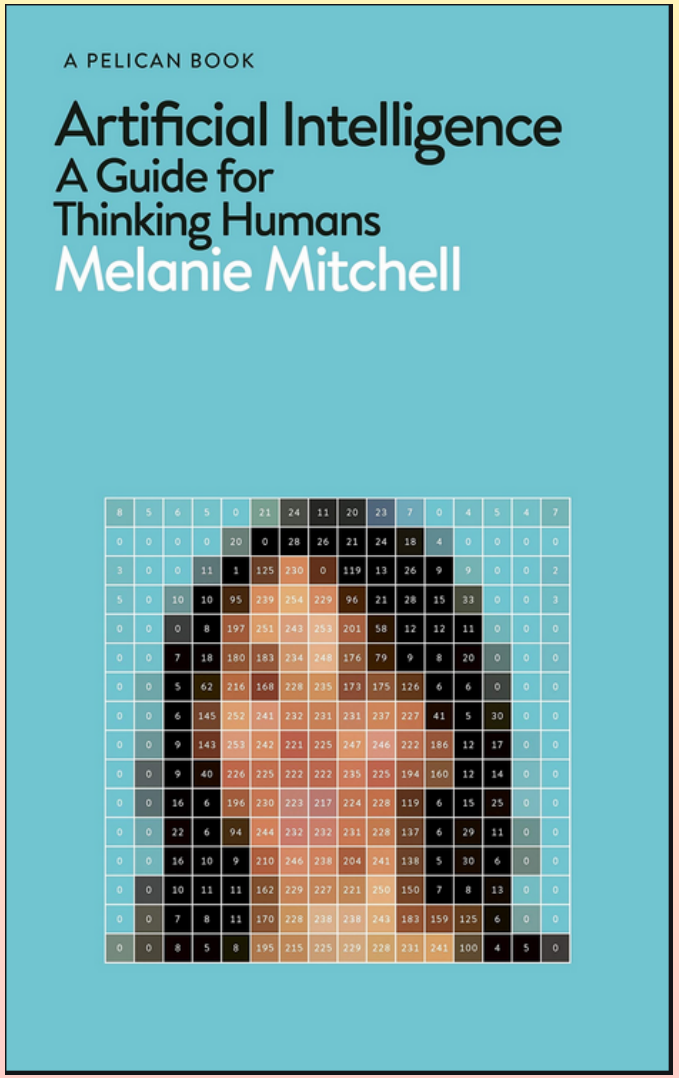
GPT-3 never gave the "reversal" answer on any of the five trials. Here are its answers:

l q r y v
r l y q v
l y r q v
r y l v q
l y r q v

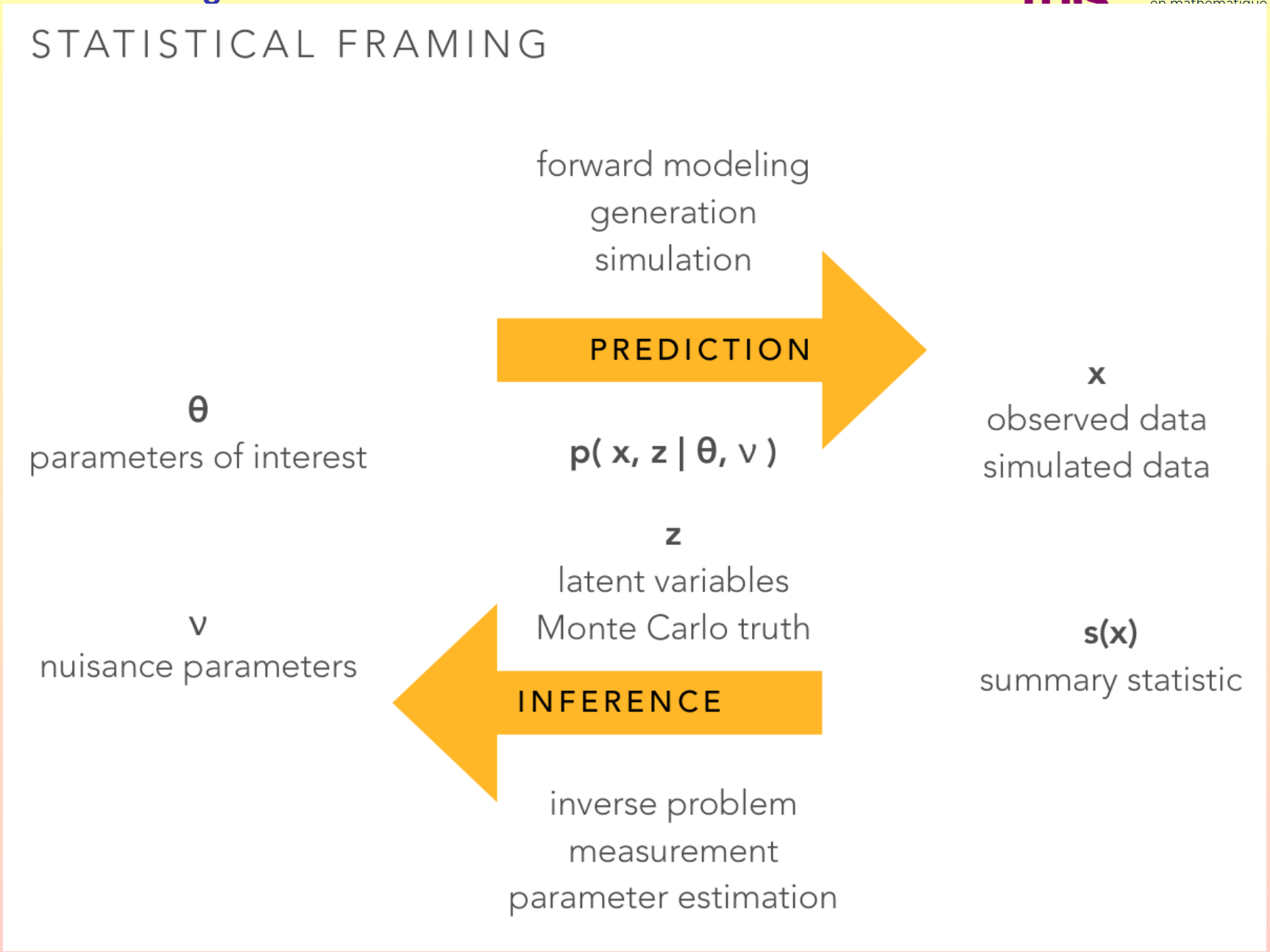
From https://medium.com/@melaniemitchell.me/follow-up-to-can-gpt-3-make-analogies-b202204bd292

- A few resources
  - Intepretability: Christoph Molnar, Interpretable Machine Learning
  - Artificial “intelligence”: Melanie Mitchell, Artificial Intelligence: A Guide for Thinking Humans





Likelihood-free inference



From Kyle Cranmer's PhyStat seminar

The likelihood is a landmark for most statistical inference methods

- (Profile) maximum likelihood fits
  - Point estimate by maximizing the likelihood function
  - Interval estimate by intersection with likelihood function
  - Feldman-Cousins (build intervals using the likelihood ratio for proper ordering of probability elements)
  - Test of hypothesis using ratios of the likelihood of the two hypotheses
- Bayesian methods rely on computing posterior distribution $p(\theta|\vec{x}) \propto p(\vec{x}|\theta)\pi(\theta)$
  - Need the likelihood to build the posterior: usually resample posterior (Markov Chain MonteCarlo)
  - Point and interval estimates from posterior shape
  - Test of hypotheses from ratio of marginal likelihoods (Bayes Factor)

- In HEP we often don't have access to the likelihood
- Monte Carlo generators are used to generate samples distributed according to a given likelihood function $x \sim p(x|\theta)$
- The likelihood sometimes is intractable

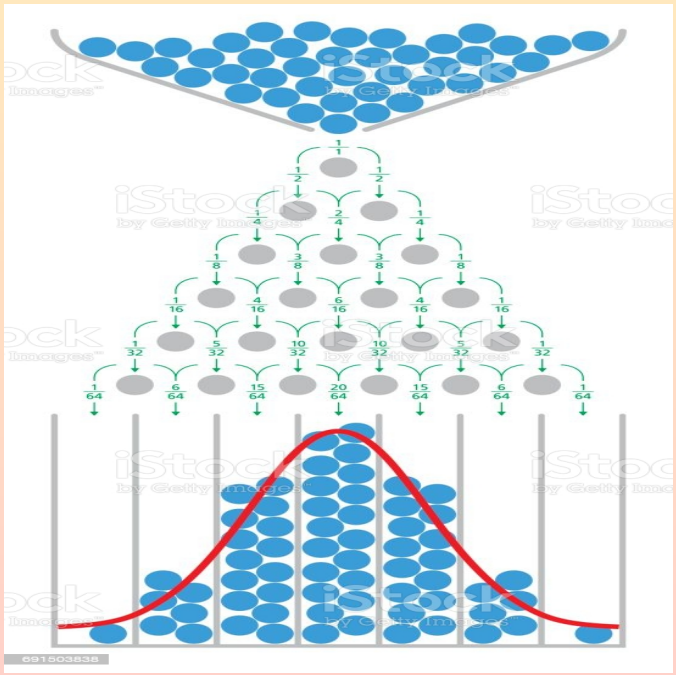
$$p(x|\theta) = \int dz p(x,z|\theta) = \int dz p_x(x|\theta,z) \prod_i p_i(z_i|\theta, z_{<i})$$

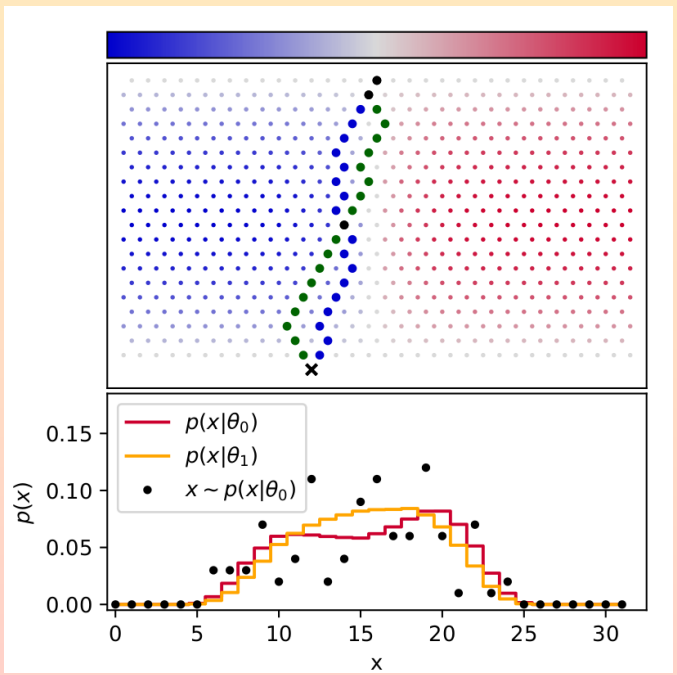
- Latent variables
  - Matrix element
  - Parton shower
  - Detector...

- Sample from the density, build an histogram representation, do inference from histogram
  - The product of Poisson likelihoods we have seen on Wednesday
- Sample from the density using accessory variables $\rightarrow$ Approximate Bayesian Computation
  - Yesterday
- Smart ways of integrating the generating function $\rightarrow$ Matrix Element Method
  - Yesterday
- Bypass the integration by learning the generating function $\rightarrow$ Surrogate models by learning the likelihood (ratio)
  - Today!
- Active learning: learn surrogates by alternating simulation and inference stages (not treated today)
  - Train autoregressive flows on simulated data in order to learn a model of the likelihood in the region of high posterior density
- Learn the structure of the data, together with the density, for intractable likelihoods $\rightarrow$ manifold learning
  - Today
- Do inference by finding a “smart”, optimal summary statistic $\rightarrow$ INFERNO
- Many more (not discussed today)

Likelihood-free inference with differentiable models

- Monte Carlo generators are used to generate samples distributed according to a given likelihood function $x \sim p(x|\theta)$
  - Sometimes likelihood intractable because it depends on latent variables $p(x|\theta) = \int dz p(x,z|\theta) = \int dz p_x(x|\theta,z) \prod_i p_i(z_i|\theta, z_{<i})$ (matrix element, parton shower, detector...)
- When the likelihood is intractable, inverting the problem to obtain $p(\theta|x)$ is impossible or requires huge amount of generated events or observations
  - Approximate Bayesian Computation (ABC): generate events by sampling from a prior $\pi(\theta)$, accept/reject algorithm to build the posterior
  - Probabilistic programming systems: use samples from generative model to train a tractable surrogate model





Galton Board example. Images from istockphoto and arXiv:1805.12244

- Our target is the likelihood ratio $\hat{r}(x|\theta_0, \theta_1)$
- Define a surrogate model by training a classifier to discriminate between equally-sized samples
  - $\{x_i\} \sim p(x|\theta_0)$
  - $\{x_i\} \sim p(x|\theta_1)$
- Use the standard binary cross-entropy loss

$$L_{\mathrm{XE}} = -\mathbb{E}[\mathbb{1}(\theta = \theta_1) \log \hat{s}(x|\theta_0, \theta_1) + \mathbb{1}(\theta = \theta_0) \log(1 - \hat{s}(x|\theta_0, \theta_1))]$$

- This is minimized by the optimal decision function

$$s(x|\theta_0, \theta_1) = p(x|\theta_1)/(p(x|\theta_0) + p(x|\theta_1))$$

- Invert the expression, to estimate the LR as

$$\hat{r}(x|\theta_0, \theta_1) = (1 - \hat{s}(x|\theta_0, \theta_1))/\hat{s}(x|\theta_0, \theta_1)$$

  - In real cases, will not in general learn exactly the optimal decision function
  - As long as the function is monotonic, can recalibrate

arXiv:1805.12244

Likelihood-free inference with differentiable models

- Likelihood is intractable $p(x|\theta) = \int \mathrm{d}z\, p(x,z|\theta) = \int \mathrm{d}z\, p_x(x|\theta,z)\ \prod_i p_i(z_i|\theta, z_{<i})$
- But the joint score can be computed by accumulating $\nabla_\theta \log p(z_i|\theta, z_{:i})$ while simulating conditioned on random trajectory z

$$t(x,z|\theta_0) \equiv \nabla_\theta \log p(x,z|\theta)\Big|_{\theta_0} = \sum_i \nabla_\theta \log p_i(z_i|\theta, z_{<i})\Big|_{\theta_0} + \nabla_\theta \log p_x(x|\theta,z)\Big|_{\theta_0} \tag{1}$$

  - The joint score can be seen as a reward function to optimize θ
- Can compute also the joint likelihood ratio

$$r(x,z|\theta_0,\theta_1) \equiv \frac{p(x,z|\theta_0)}{p(x,z|\theta_1)} = \frac{p_x(x|\theta_0,z)}{p_x(x|\theta_1,z)}\ \prod_i \frac{p_i(z_i|\theta_0,z_{<i})}{p_i(z_i|\theta_1,z_{<i})}\,. \tag{2}$$

- Joint score + likelihood ratio quantify how likely a particular simulated trajectory through the simulator is if you change θ
  - **Augmented** data (I prefer to use the term *armored*, to avoid confusion with resampling techniques)

Learning from these augmented data

- Ingredients (the "augmented" data)
  - Simulated observations x_i
  - Joint likelihood ratio $r(x_i, z_i|\theta_0, \theta_1)$
  - Joint score $t(x_i, z_i|\theta_0)$
- Goal: estimate likelihood $p(x|\theta)$ or likelihood ratio $r(x|\theta_0, \theta_1)$
  - Estimate $r(x|\theta_0, \theta_1)$ from $r(x, z|\theta_0, \theta_1)$: non-trivial ($\int \textit{ratio} \neq \textit{ratio} \int$)
  - Estimate $t(x|\theta_0) \equiv \nabla_\theta \log p(x|\theta)\Big|_{\theta_0}$ $t(x_i, z_i|\theta_0)$ from $t(x_i, z_i|\theta_0)$: non-trivial ($\int \log \neq \log \int$)
- For the joint likelihood ratio
  - Define mean-square-error loss :

$$L_r = \mathbb{E}_{p(x,z|\theta_1)} \left[\left(r(x, z|\theta_0, \theta_1) - \hat{r}(x) \right)^2 \right]$$

  - This is minimized by

$$r^*(x) = \arg\min_{\hat{r}} L_r = \mathbb{E}_{p(z|x,\theta_1)} \left[r(x, z|\theta_0, \theta_1) \right] = r(x|\theta_0, \theta_1)$$

- For the joint score (setting $\theta = \theta_0$)
  - Define mean-square-error loss:

$$L_t = \mathbb{E}_{p(x,z|\theta_0)} \left[\left(t(x, z|\theta_0) - \hat{t}(x|\theta_0) \right)^2 \right] ,$$

  - This is minimized by

$$t^*(x) = \mathbb{E}_{p(z|x,\theta_0)} \left[t(x, z|\theta_0) \right] = t(x|\theta_0)$$

- We have learned from these augmented data to regress on the two intractable quantities, i.e. to transform:
  - $t(x, z|\theta_0)$ into $t(x|\theta_0)$
  - $r(x, z|\theta_0, \theta_1)$ into $r(x|\theta_0, \theta_1)$

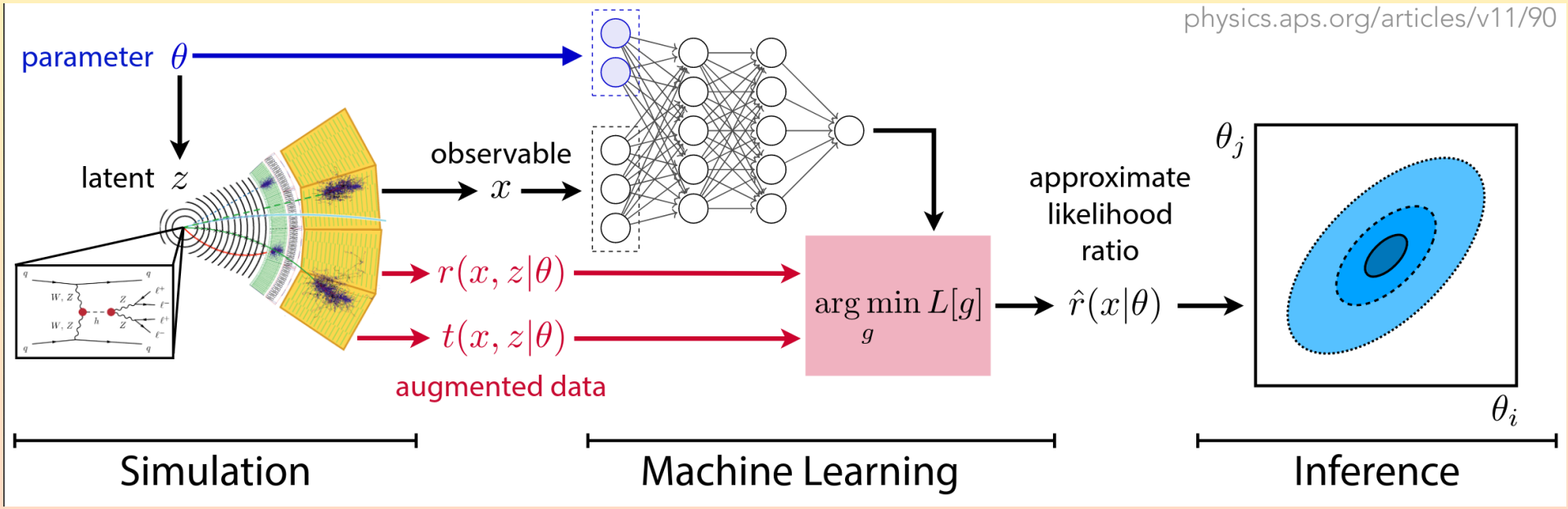
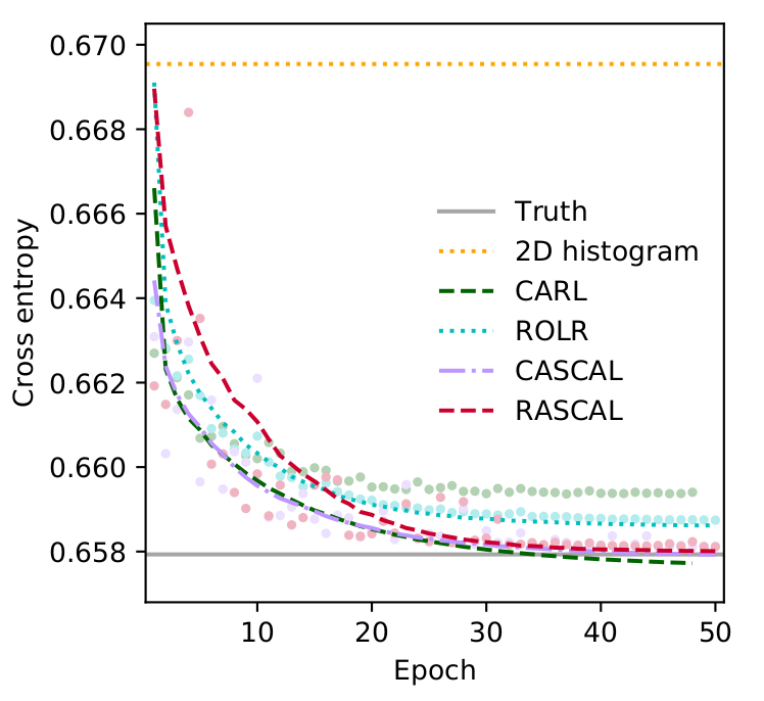
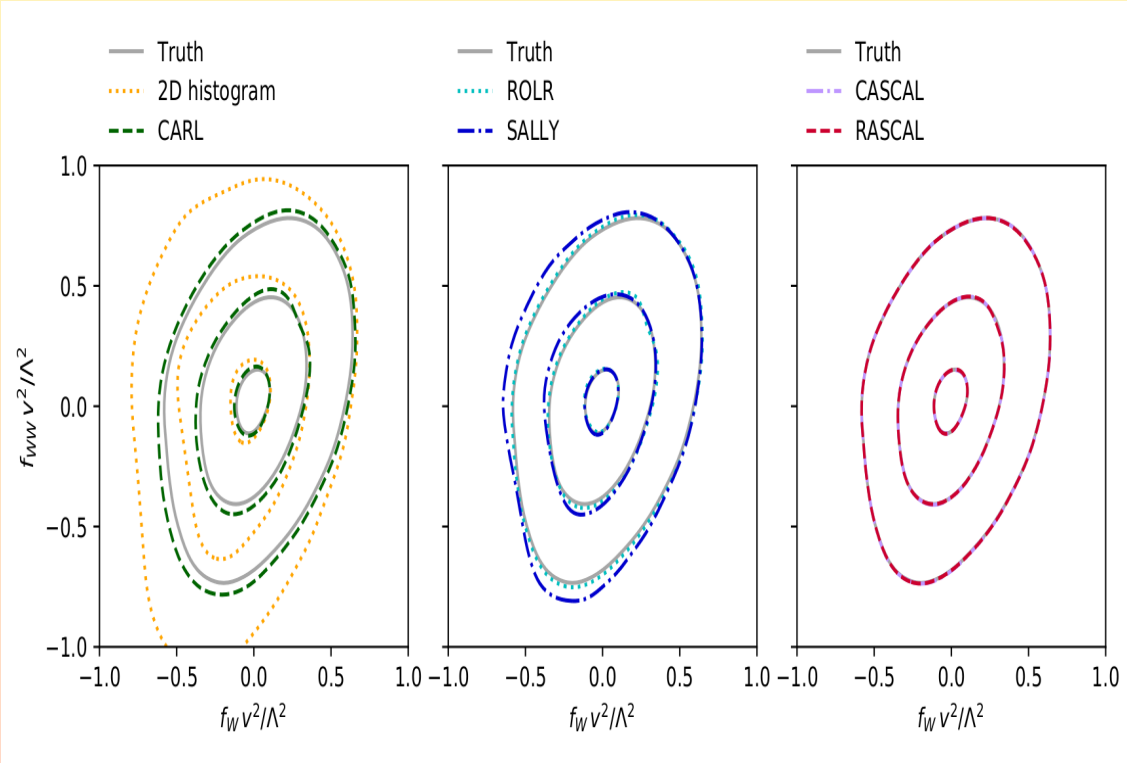


Image from K. Cranmer's PhyStat seminar

- Calculate the full true parton-level likelihood starting from N simulated events
  - Obtain a sufficient statistic for inference; exploit all available information!





Images from arXiv:1805.00020

- Calculate the full true parton-level likelihood starting from N simulated events
  - Inference not limited anymore by the size of the generated samples

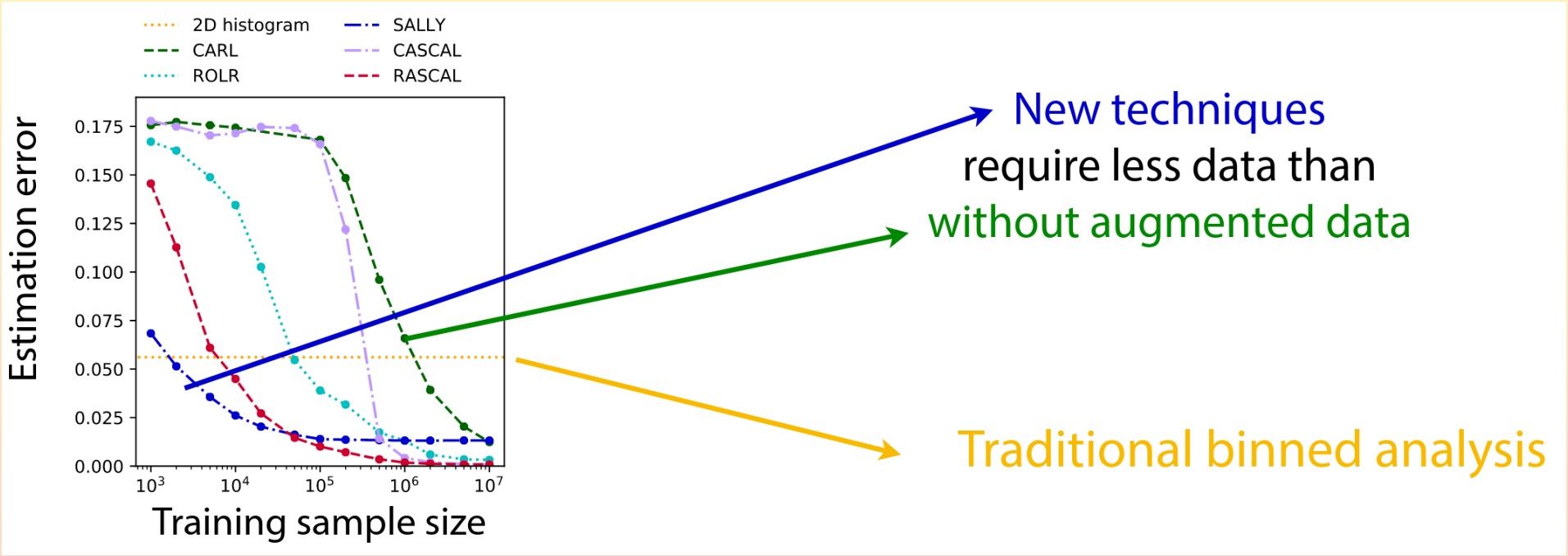


Image from K. Cranmer's PhyStat seminar and arXiv:1805.00020

EFT studies: from "basis" points to a full likelihood!

UCLouvain
Institut de recherche en mathématique et physique

- Effective field theories and similar parametrizations contribute a finite number of amplitudes to a given process
  - Each amplitude is multiplied by a function of the Wilson coefficients
- Can write the likelihood in terms of the different amplitude components c'
  - $f_{c'}(z)$ are not necessarily properly positive definite or normalized

$$p(z|\theta) = \sum_{c'} \tilde{w}_{c'}(\theta) f_{c'}(z)$$

- Example: SM amplitude $\mathcal{M}_0(z)$ interferes with one new physics amplitude $\mathcal{M}_{\text{BSM}}(z|\theta) = \theta \mathcal{M}_1(z)$, which scales linearly with a new physics parameter θ.
  - Differential cross section: $\mathrm{d}\sigma(z) \propto |\mathcal{M}_0(z)|^2 + 2\theta\ \mathrm{Re}\, \mathcal{M}_0(z)^\dagger \mathcal{M}_1(z) + \theta^2 |\mathcal{M}_1(z)|^2$
  - Each component (SM, interference, BSM) has its own parameter dependence $\tilde{w}_{c'}(z)$ and momentum dependence $f_{c'}(z)$
- Pick a number of basis parameter points, one θ_c for each c'
  - **Not related to the choice of EFT operator basis**
- If the choice makes the matrix $W_{cc'} = \tilde{w}_{c'}(\theta_c)$ invertible, then the likelihood is a mixture model

$$p(z|\theta) = \sum_{c} w_c(\theta)\, p_c(z)$$

  - Weights $w_c(\theta) = \sum_{c'} \tilde{w}_c(\theta)\, W^{-1}_{cc'}$
  - Properly normalized basis densities $p_c(z) = p(z|\theta_c)$
  - Weights $w_c(\theta)$ depend on the choice of basis points and are analytically known.

EFT studies: learn the parameterized likelihood when intractable

- From the morphing we obtain the full likelihood function $p(z|\theta)$ from a finite set of evaluations of basis densities $p_c(z)$.
- We still need $p(z|\theta)$ to be tractable

$$p(x|\theta) = \int \mathrm{d}z\, p(x,z|\theta) = \int \mathrm{d}z\, p(x|z)\, p(z|\theta)\,.$$

- Although it's true that even if it is intractable the following is true:

$$p(x|\theta) = \sum_c w_c(\theta)\, p_c(x)$$

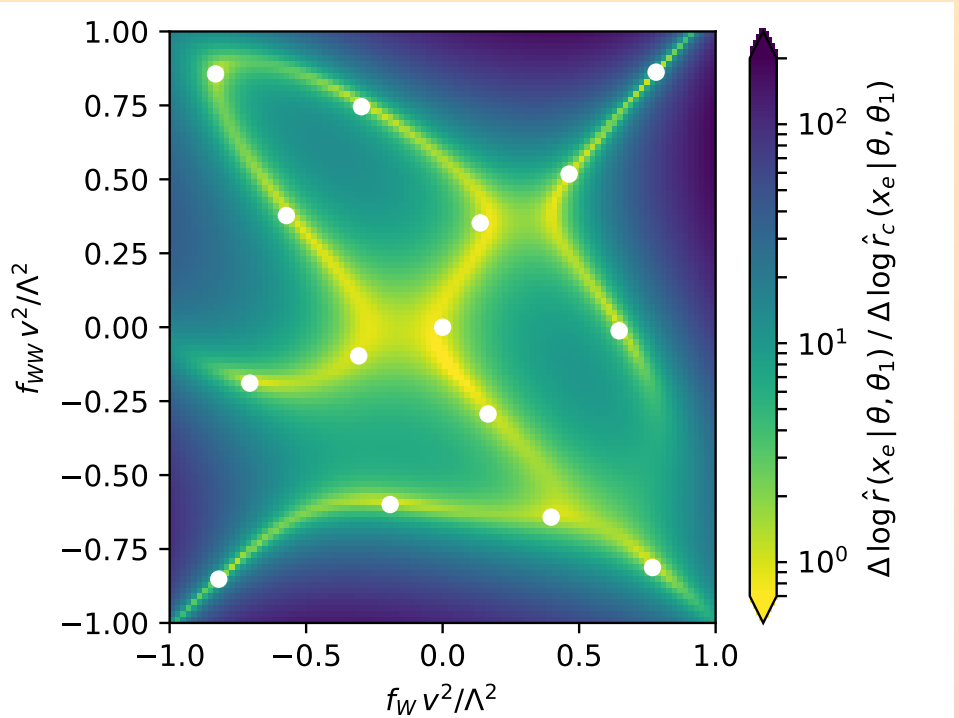
- With this, impose the morphing parametrization to the likelihood ratio estimators
- Finally, learn the LR estimators as we did before
  - Uncertainty small near the basis points

- Use a composite reference hypothesis (fixed denominator) (the basis estimators $\hat{r}_c(x) = \hat{r}(x|\theta_c, \theta_1)$ only depend on x)

$$\hat{r}(x|\theta_0, \theta_1) = \sum_c w_c(\theta_0)\, \hat{r}_c(x)$$

- Decompose both the numerator and denominator distributions with pairwise estimators $\hat{r}_{c',c}(x) = \hat{r}(x|\theta_{c'}, \theta_c)$.

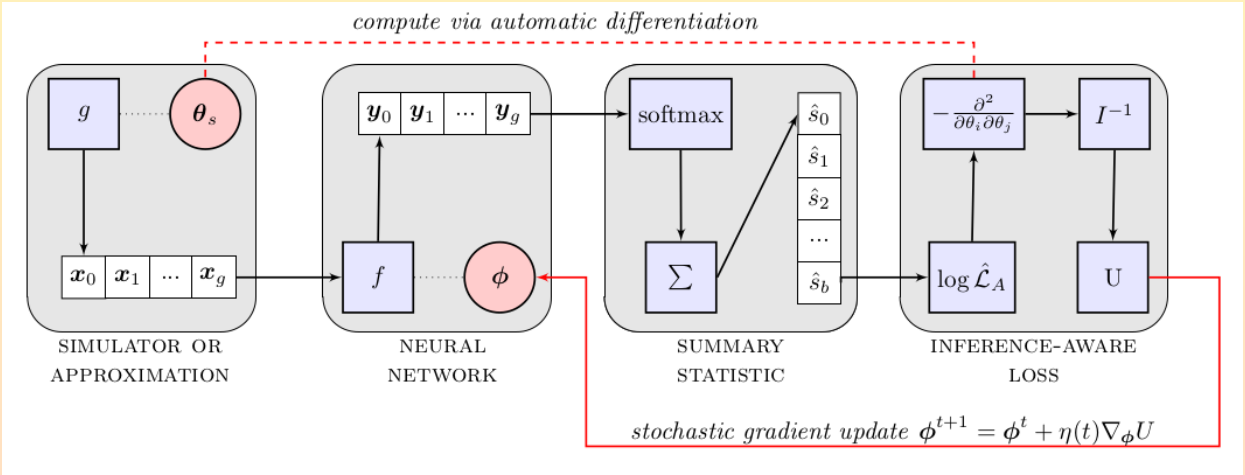
$$\hat{r}(x|\theta_0, \theta_1) = \sum_c \left[\sum_{c'} \frac{w_{c'}(\theta_1)}{w_{c'}(\theta_0)}\, \hat{r}_{c',c}(x) \right]^{-1}$$

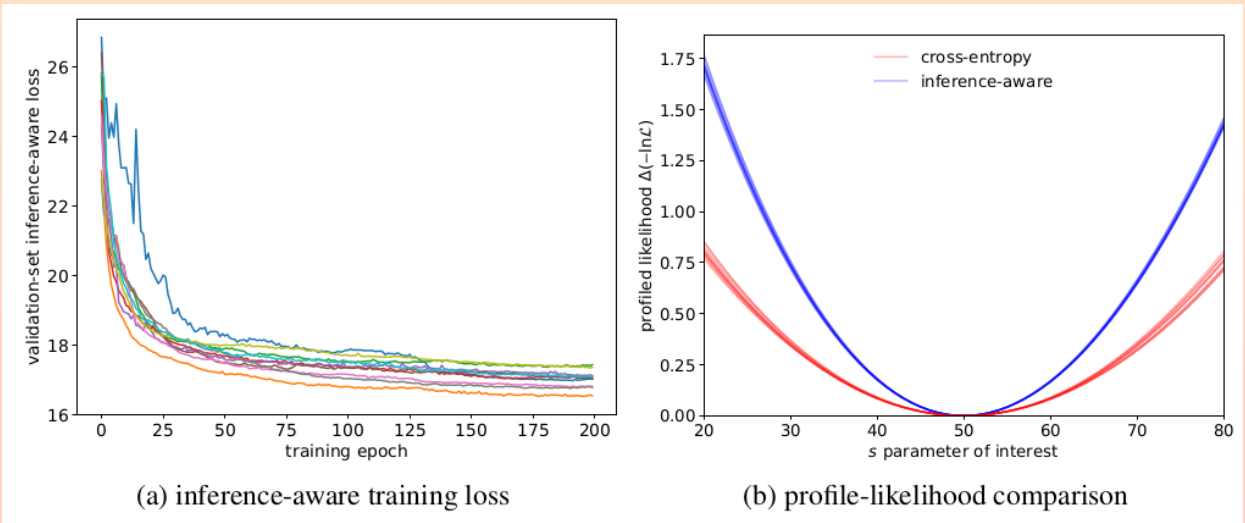


- Resources to start playing with this
  - https://github.com/diana-hep/madminer (in particular `examples/tutorial_particle_physics`)
  - Also look at https://github.com/johannbrehmer/goldmine and https://github.com/johannbrehmer/goldmine

INFERNO: inference-aware neural optimization in one slide

- Build a nonparametric likelihood function based on the simulation, and use it as summary statistic
- Minimize the expected variance of the parameter of interest
  - Obtain the Fisher information matrix via automatic differentiation, and use it as loss function!
  - For (asymptotically) unbiased estimators, Rao-Cramér-Frechet (RCF) bound $V[\hat{\theta}] \sim \frac{1}{\hat{\theta}}$
  - Constraints from auxiliary measurements (i.e. systematic uncertainties) included out of the box in the covariance matrix!





(a) inference-aware training loss (b) profile-likelihood comparison

From De Castro-Dorigo, arXiv:1806.04743, and AMVA4NewPhysics deliverable 1.4 public report

- Apply a NN classifier to each event in a dataset, obtaining a lower-dimensional summary statistic
  $\boldsymbol{f}(\boldsymbol{x};\boldsymbol{\phi}) : \mathcal{X} \subseteq \mathbb{R}^d \to \mathcal{Y} \subseteq \mathbb{R}^b$
  - ϕ are the neural network parameters learned via stochastic gradient descent
  - $dim(\mathcal{Y})$ is the number of output neurons (might be one, might be more)
- Now need to map this into a summary statistic, to be calibrated and optimized via a non-parameteric likelihood
  $\mathcal{L}(D;\boldsymbol{\theta},\boldsymbol{\phi})$
  - $G_s = \{\boldsymbol{x}_0, \ldots, \boldsymbol{x}_g\}$ simulated observations conditional on values of $\boldsymbol{\theta}_s$

- In HEP we tend to use histograms and express likelihoods as product of Poisson factors per each bin
- Consider the NN output $\boldsymbol{f}(\boldsymbol{x};\boldsymbol{\phi})$, and assign each observation $\boldsymbol{x}$ to a bin defined as:

$$s_i(D;\boldsymbol{\phi}) = \sum_{\boldsymbol{x}\in D} \begin{cases} 1 & i = argmax_{j=\{0,\dots,b\}}\left(f_j(\boldsymbol{x};\boldsymbol{\phi})\right) \\ 0 & i \neq argmax_{j=\{0,\dots,b\}}\left(f_j(\boldsymbol{x};\boldsymbol{\phi})\right) \end{cases}$$

- From this one can take the per-bin expectation

$$\mathcal{L}(D;\boldsymbol{\theta},\boldsymbol{\phi}) = \prod_{i=0}^{b} \text{Pois}\left(s_i(D;\boldsymbol{\phi}) \mid \left(\frac{n}{g}\right) s_i(G_s;\boldsymbol{\phi})\right)$$

  - n/g accounts for the different number of obs in the samples
  - Can extend to cases where N_{obs} is a random variable (typical HEP case) etc

- This likelihood is not differentiable (because of $argmax$), so use surrogate (softmax operator)

$$\hat{s}_i(D;\boldsymbol{\phi}) = \sum_{x \in D} \frac{e^{f_i(\boldsymbol{x};\boldsymbol{\phi})/\tau}}{\sum_{j=0}^{b} e^{f_j(\boldsymbol{x};\boldsymbol{\phi})/\tau}}$$

  - τ regulates how soft is softmax: $\hat{\boldsymbol{s}}(D;\boldsymbol{\phi}) \lim_{\tau \to 0+} \boldsymbol{s}(D;\boldsymbol{\phi})$
- End up with $\hat{\mathcal{L}}(D;\boldsymbol{\theta},\boldsymbol{\phi})$. In the Asimov dataset,

$$\hat{\mathcal{L}}_A(\boldsymbol{\theta};\boldsymbol{\phi}) = \prod_{i=0}^{b} \text{Pois}\left(\left(\frac{n}{g}\right)\hat{s}_i(G_s;\boldsymbol{\phi}) \mid \left(\frac{n}{g}\right)\hat{s}_i(G_s;\boldsymbol{\phi})\right)$$

  - MLE is the original generator parameters $argmax_{\boldsymbol{\theta} \in \theta}(\hat{\mathcal{L}}_A(\boldsymbol{\theta};\boldsymbol{\phi})) = \boldsymbol{\theta}_s$

Information matrix computable by autodiff (if time allows)

- Fully differentiable Fisher information:

$$\boldsymbol{I}(\boldsymbol{\theta})_{ij} = \frac{\partial^2}{\partial\theta_i \partial\theta_j}\left(-\log\hat{\mathcal{L}}_A(\boldsymbol{\theta};\boldsymbol{\phi})\right)$$

  - If the simulation is itself differentiable or if one can approximate variations of $\boldsymbol{\theta}$ over G_s
- For an unbiased MLE estimate $\hat{\boldsymbol{\theta}}$ of $\boldsymbol{\theta}$, Cramŕ-Rao bound (equality in high-stat limit) satisfied

$$\mathrm{cov}_{\boldsymbol{\theta}}(\hat{\boldsymbol{\theta}}) \geq I(\boldsymbol{\theta})^{-1}$$

- Can use inverse of the Fisher information as approximate estimator of the expected variance!
- Immediate extension to nuisance parameters constrained by auxiliary measurements $\{\mathcal{L}_C^0(\boldsymbol{\theta}), ..., \mathcal{L}_C^c(\boldsymbol{\theta})\}$

$$\hat{\mathcal{L}}'_A(\boldsymbol{\theta};\boldsymbol{\phi}) = \hat{\mathcal{L}}_A(\boldsymbol{\theta};\boldsymbol{\phi}) \prod_{i=0}^{c} \mathcal{L}_C^i(\boldsymbol{\theta}).$$

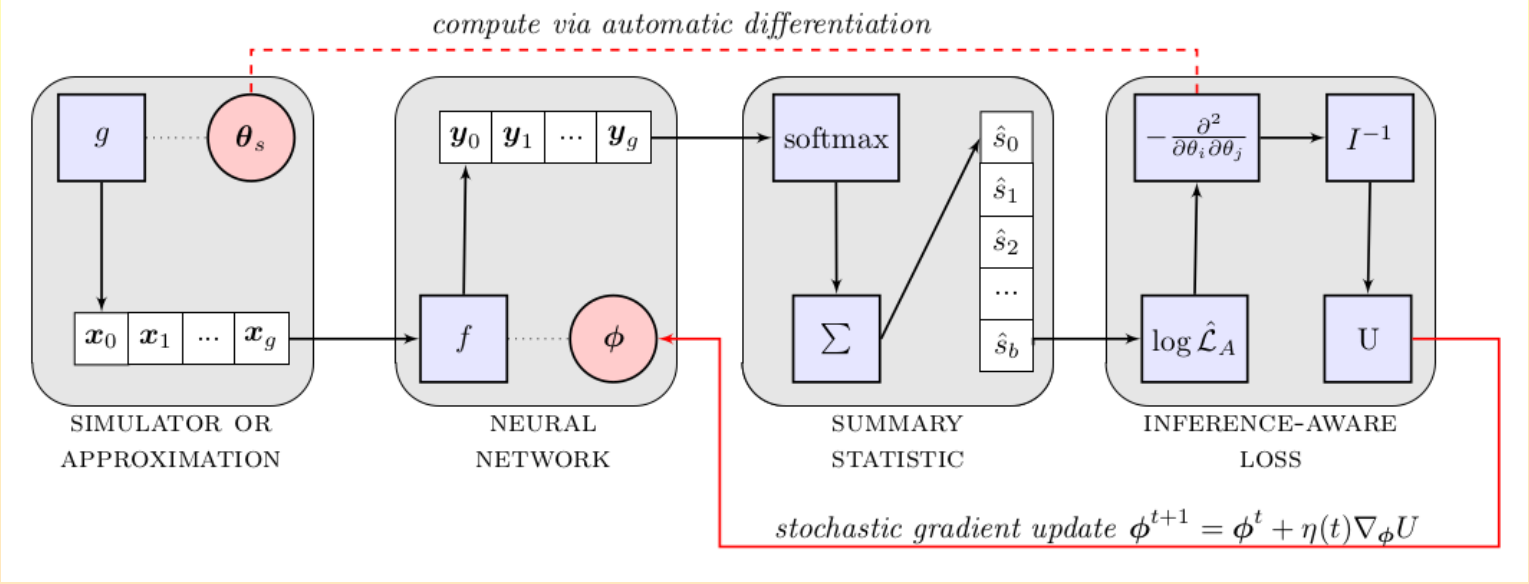
- Loss function: an function of the inverse of Fisher Information matrix at $\boldsymbol{\theta}_s$
  - Diagonal elements of $I_{ii}^{-1}(\boldsymbol{\theta}_s)$: expected variance of each of the ϕ_i under the normal approximation
  - Inference about $\omega_0 = \theta_k$: use loss function $U = I_{kk}^{-1}(\boldsymbol{\theta}_s)$ (expected width of the confidence interval for ω_0 accounting also for the nuisance parameters
- Also immediate extension to many parameters of interest

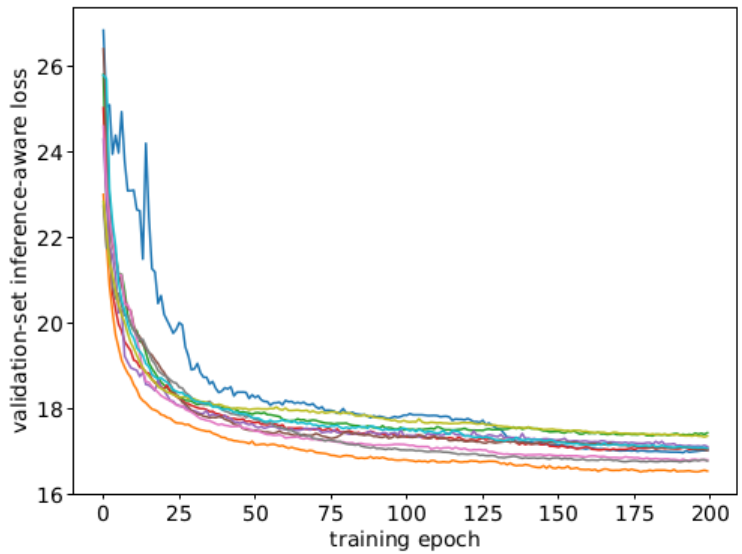
Algorithm 1 Inference-Aware Neural Optimisation.

Input 1: differentiable simulator or variational approximation $g(\boldsymbol{\theta})$.
Input 2: initial parameter values $\boldsymbol{\theta}_s$.
Input 3: parameter of interest $\omega_0 = \theta_k$.
Output: learned summary statistic $\boldsymbol{s}(D; \boldsymbol{\phi})$.

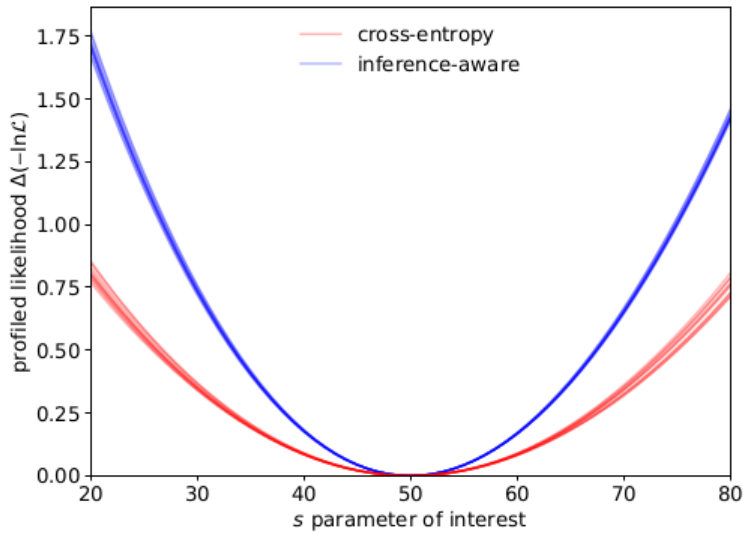
1: **for** $i = 1$ to N **do**
2: Sample a representative mini-batch G_s from $g(\boldsymbol{\theta}_s)$.
3: Compute differentiable summary statistic $\hat{\boldsymbol{s}}(G_s; \boldsymbol{\phi})$.
4: Construct Asimov likelihood $\mathcal{L}_A(\boldsymbol{\theta}, \boldsymbol{\phi})$.
5: Get information matrix inverse $I(\boldsymbol{\theta})^{-1} = \boldsymbol{H}_{\boldsymbol{\theta}}^{-1}(\log \mathcal{L}_A(\boldsymbol{\theta}, \boldsymbol{\phi}))$.
6: Obtain loss $U = I_{kk}^{-1}(\boldsymbol{\theta}_s)$.
7: Update network parameters $\boldsymbol{\phi} \rightarrow \text{SGD}(\nabla_{\boldsymbol{\phi}} U)$.
8: **end for**

Promising results (if time allows)





(a) inference-aware training loss



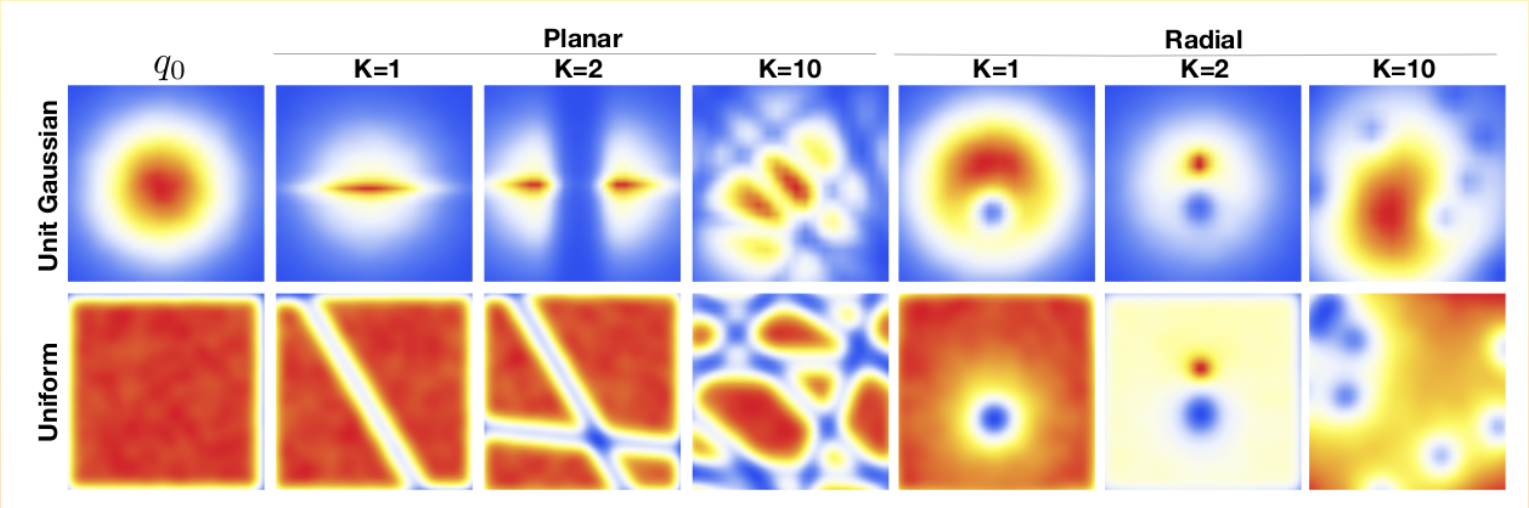
(b) profile-likelihood comparison

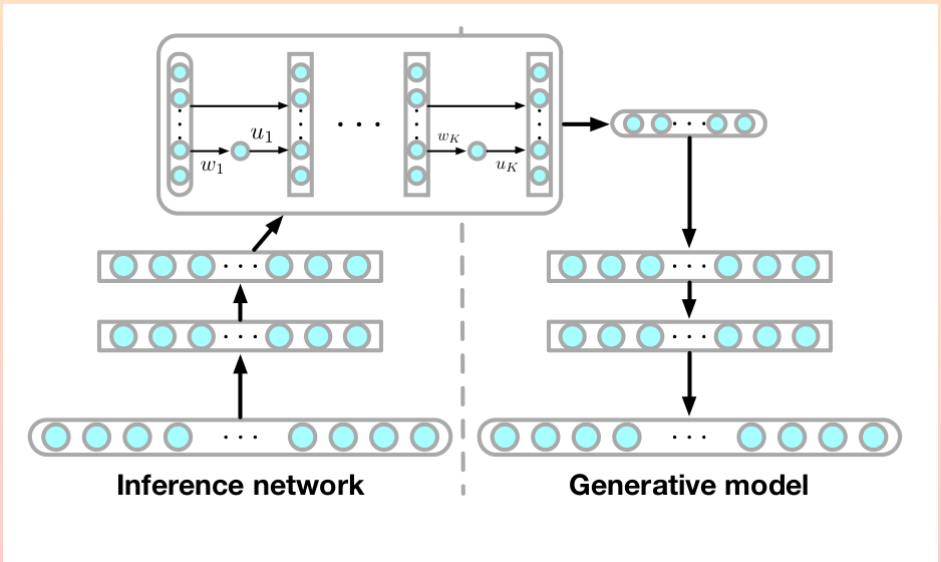
From De Castro-Dorigo, arXiv:1806.04743, and AMVA4NewPhysics deliverable 1.4 public report

- Resources
  - Tensorflow 1 implementation: https://github.com/pablodecm/paper-inferno (Pablo De Castro Manzano)
  - Tensorflow 2 implementation is under validation (Lukas Layer)
  - PyTorch implementation is under validation (Giles Strong)

Normalizing flows

- Variational inference (Rezende and Mohamed, 1505.05770)
  - Start from a simple density
  - Apply a sequence of invertible transformations until the desired complexity is reached
- Combination of inference and generative model
  - Inference network maps observations to the parameters of the model
  - Generative model receives the posterior samples from the inference network at training time





- Decompose a joint density into a product of conditional densities
  - Condition on *previous* variables (time series, or lower-index coordinates for some ordering
  - Predicted value of features depend on past values of the same feature, rather than on other predictors
- Used for density estimation
  - Take some variable with some implicit ordering (e.g. tensor)
  - Output a mean and standard deviation for each element of the input, conditioned on previous elements
- In a sense, a Bayesian network

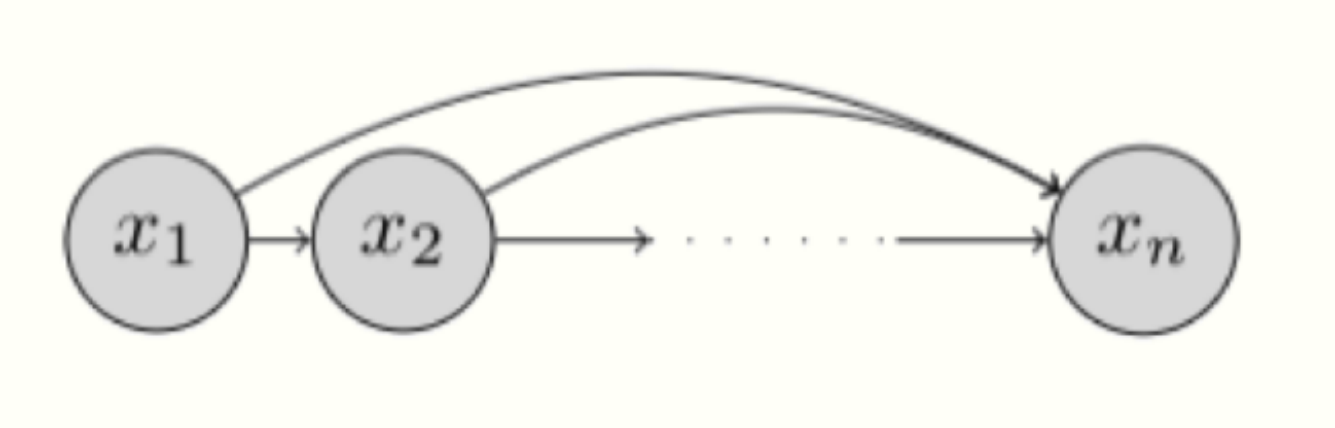
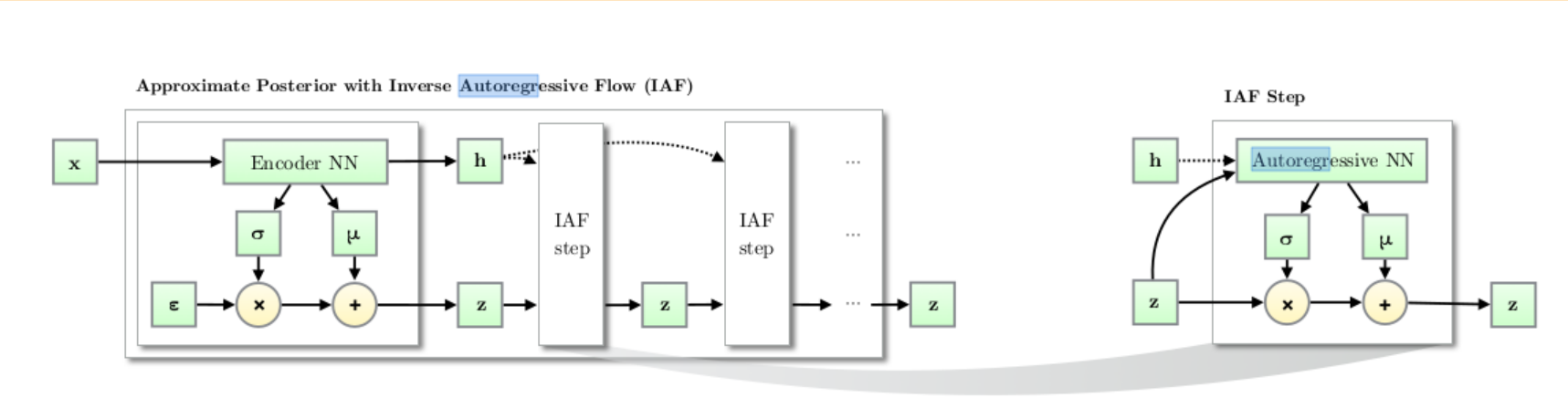


Image from https://deepgenerativemodels.github.io

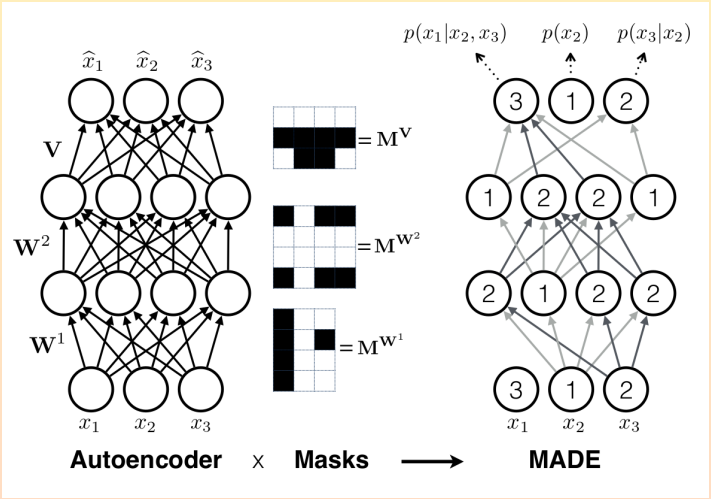
Autoregressive models are normalizing flows (if invertible)!

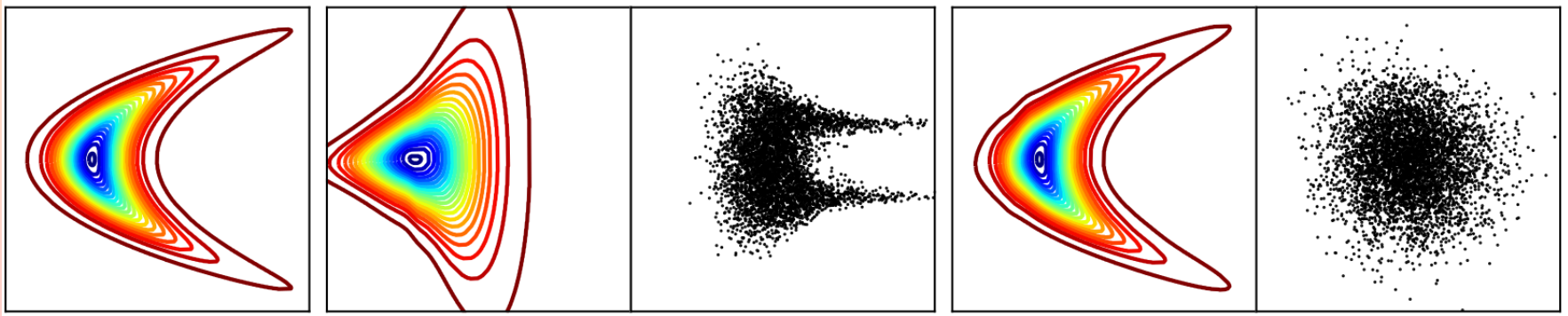
- Kingma and colleagues 1606.04934
  - Autoregressive models result in tranformations that by construction have a tractable Jacobian
  - If the transformation is also invertible, then the model is equivalent to a normalizing flow
- Can therefore be used for density estimation, e.g. Inverse Autoregressive Flow
- Decompose the operation of sampling from a posterior (typical Bayesian operation, e.g. in MCMC or ABC)
  - Sample from a very simple gaussian distribution
  - Apply chain of nonlinear invertible transformations
  - Obtain posterior sample



MADE and MAF

- MADE Germain and colleagues, 1502.03509
  - Masked Autoencoders for Distribution Estimation
  - Vectorized architecture for density estimation based on autoencoders
  - Fast and reliable
  - *Masked*: deactivate inputs to enforce autoregressive model!
- Masked Autoregressive Flows (MAF), Papamakarios and colleagues, 1705.07057
  - If you stack several autoregressive models that each learn a conditional density, you obtain a normalizing flow!

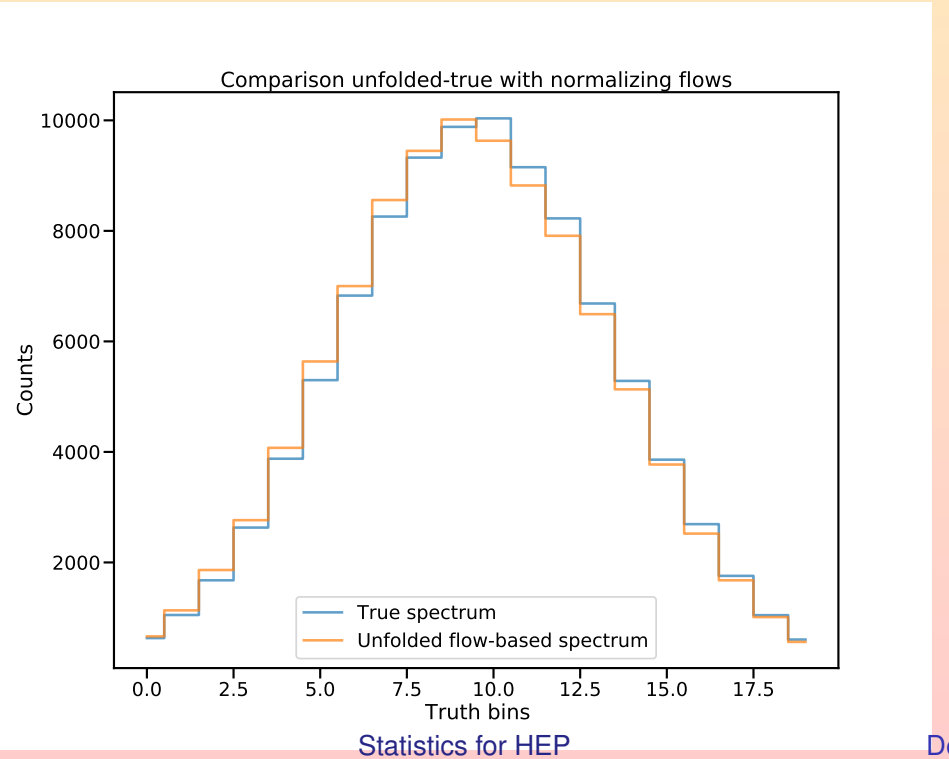




(a) Target density (b) MADE with Gaussian conditionals (c) MAF with 5 layers

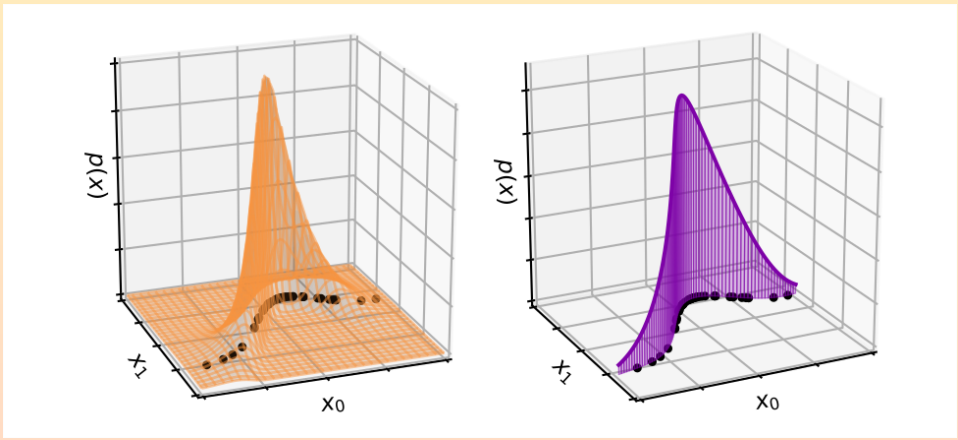
Can use inverted MAF transformation as inverse of response matrix

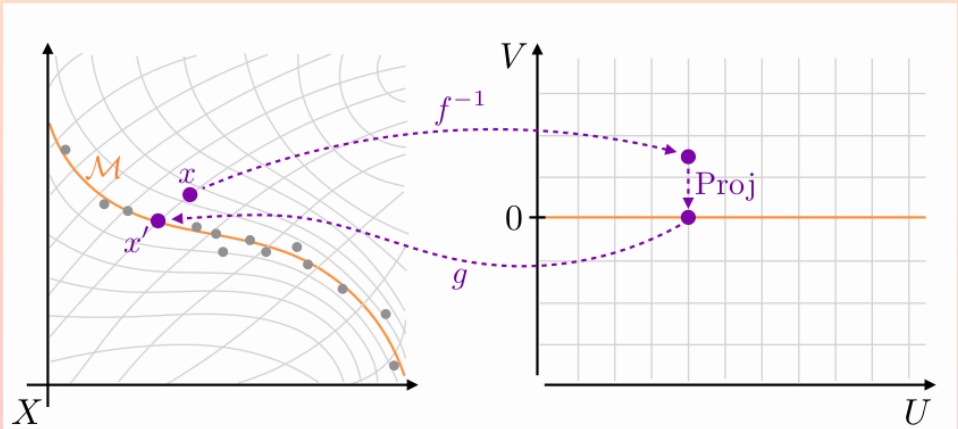
- Generation of events: how are generator-level events x transformed into reconstructed events y by passing through the detector?
  - Model the detector response as a matrix, $y = Rx$
  - Given observed data y, inverting the matrix R permits to "undo" the smearing induced by the detector
  - Nontrivial inverse problem! Many algorithms available, mostly based on regularizing an explicitly inverted matrix
- Vischia (2020b) arXiv:2009:02913 uses resampling to avoid matrix inversion
- Unfolding w/ normalizing flows described here scooped last week on the arxiv: https://arxiv.org/abs/2011.05836 (unfolding with normalizing flows using Empirical Bayes techniques). Check the paper out, it's very nice!
- Unfolding by learning the response matrix as an invertible trasformation with MAF
- More studies ongoing with more complex models (gaussian-to-gaussian might be too simple) and using manifold learning

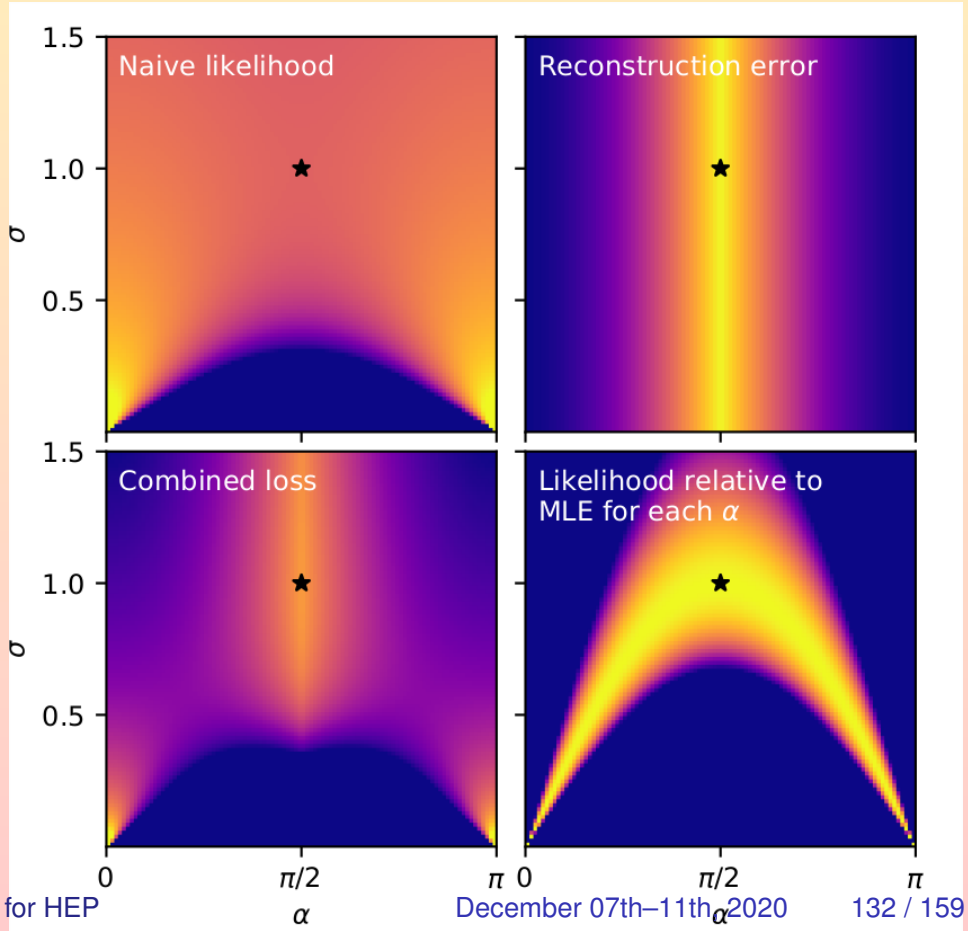


Manifold learning https://arxiv.org/abs/2003.13913

- Regular GAN/autoencoder applications: efficient sampling but describe lower-dim data manifold embedded in the data space
- Normalizing flows: obtain a density in the full data space (but if data points don't populate full space, will learn smeared-out version)
- Manifold-modelling flows: learn the shape of the manifold and a p.d.f. over the manifold
  - But cannot just maximize likelihood, as MFMF provides likelihood's projection on the manifold → additional training objective minimize $|| x - x' ||$
  - Separate parameters describing manifold from those describing the density
  - Useful for: use lower-dimensional latent spaces while retaining info on manifold; denoising; anomaly detection







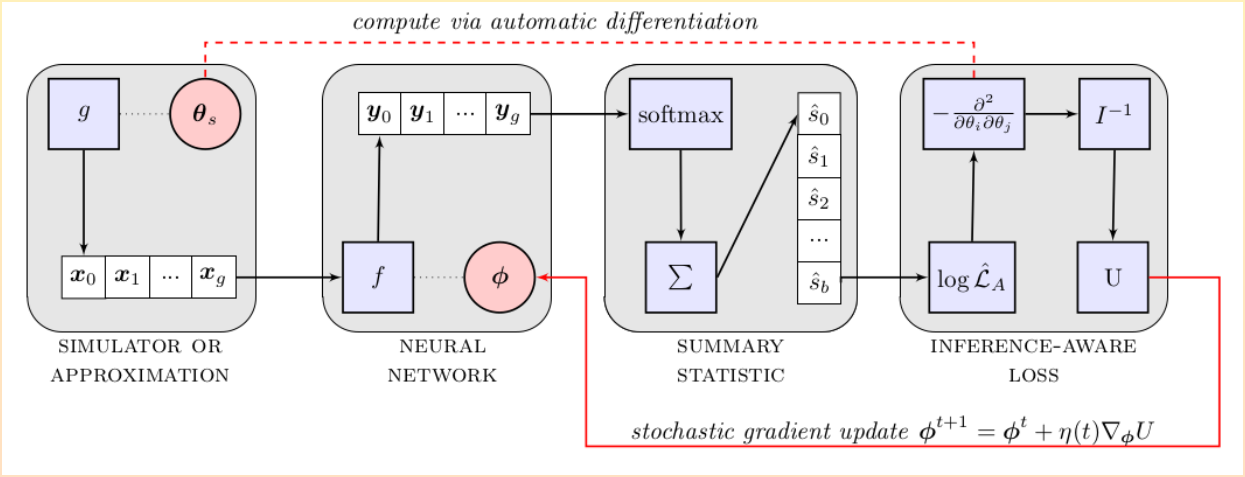
Manifold learning when likelihood is intractable

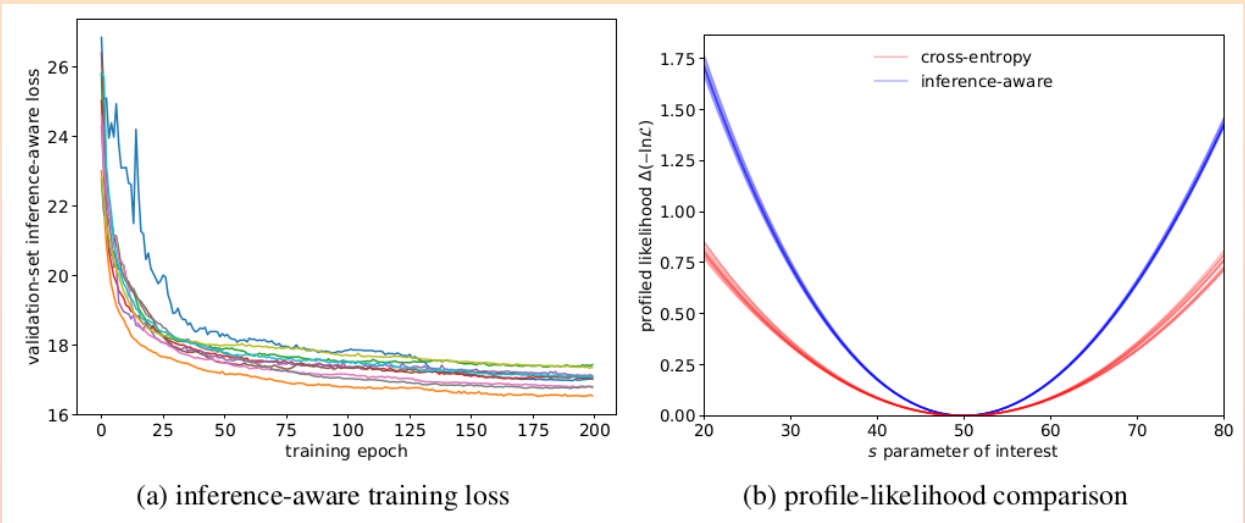
- Particle physics example: 40 features, manifold structure (quadrimomenta, etc)
- Use domain knowledge: data populate some 14D manifold within a 40D data space
  - Shape of the manifold independent of the model parameters θ
  - Probability density on the manifold is conditional on the model parameters θ
- Either learn regular likelihood ratio...
- ...or a score-augmented likelihood
  - Use differentiability of the parametrized neural density estimation w.r.t. θ
  - Add to the loss the MSE b/ween the model score and the joint score
- Compare with a "true posterior" (since likelihood is intractable, approximate it with kernel density estimation)
  - Learning the likelihood ratio provides better modelling but poorer inference
  - Learning the score-augmented likelihood provides poorer modelling but better inference

Model (algorithm)	Sample closure	Mean reconstruction error	Log posterior
AF	**0.0019** $\pm$ 0.0001	–	-3.94 ± 0.87
PIE (original)	0.0023 $\pm$ 0.0001	2.054 $\pm$ 0.076	-4.68 ± 1.56
PIE (unconditional manifold)	0.0022 $\pm$ 0.0001	1.681 $\pm$ 0.136	-1.82 ± 0.18
$\mathcal{M}$-flow	0.0045 $\pm$ 0.0004	**0.012** $\pm$ 0.001	-1.71 ± 0.30
$\mathcal{M}_e$-flow	0.0046 $\pm$ 0.0002	0.029 $\pm$ 0.001	-1.44 ± 0.34
AF (SCANDAL)	0.0565 $\pm$ 0.0059	0.000 $\pm$ 0.000	-0.40 ± 0.09
PIE (original, SCANDAL)	0.1293 $\pm$ 0.0218	3.090 $\pm$ 0.052	0.03 $\pm$ 0.17
PIE (uncond. manifold, SCANDAL)	0.1019 $\pm$ 0.0104	1.751 $\pm$ 0.064	**0.23** $\pm$ 0.05
$\mathcal{M}$-flow (SCANDAL)	0.0371 $\pm$ 0.0030	**0.011** $\pm$ 0.001	0.11 $\pm$ 0.04
$\mathcal{M}_e$-flow (SCANDAL)	0.0291 $\pm$ 0.0010	0.030 $\pm$ 0.002	**0.14** $\pm$ 0.09
Likelihood ratio estimator (ALICES)	–	–	0.05 $\pm$ 0.05

INFERNO: inference-aware neural optimization

- Build a nonparametric likelihood function based on the simulation, and use it as summary statistic
- Minimize the expected variance of the parameter of interest
  - Obtain the Fisher information matrix via automatic differentiation, and use it as loss function!
  - For (asymptotically) unbiased estimators, Rao-Cramér-Frechet (RCF) bound $V[\hat{\theta}] \sim \frac{1}{\hat{\theta}}$
  - Constraints from auxiliary measurements (i.e. systematic uncertainties) included out of the box in the covariance matrix!





(a) inference-aware training loss (b) profile-likelihood comparison

From De Castro-Dorigo, arXiv:1806.04743, and AMVA4NewPhysics deliverable 1.4 public report

ARE YOU READY TO INCLUDE MACHINE LEARNING IN YOUR RESEARCH?

Not before having coded a neural network from scratch!!!
This afternoon's session!
Now, if time allows: overview of the use of neural networks in physics

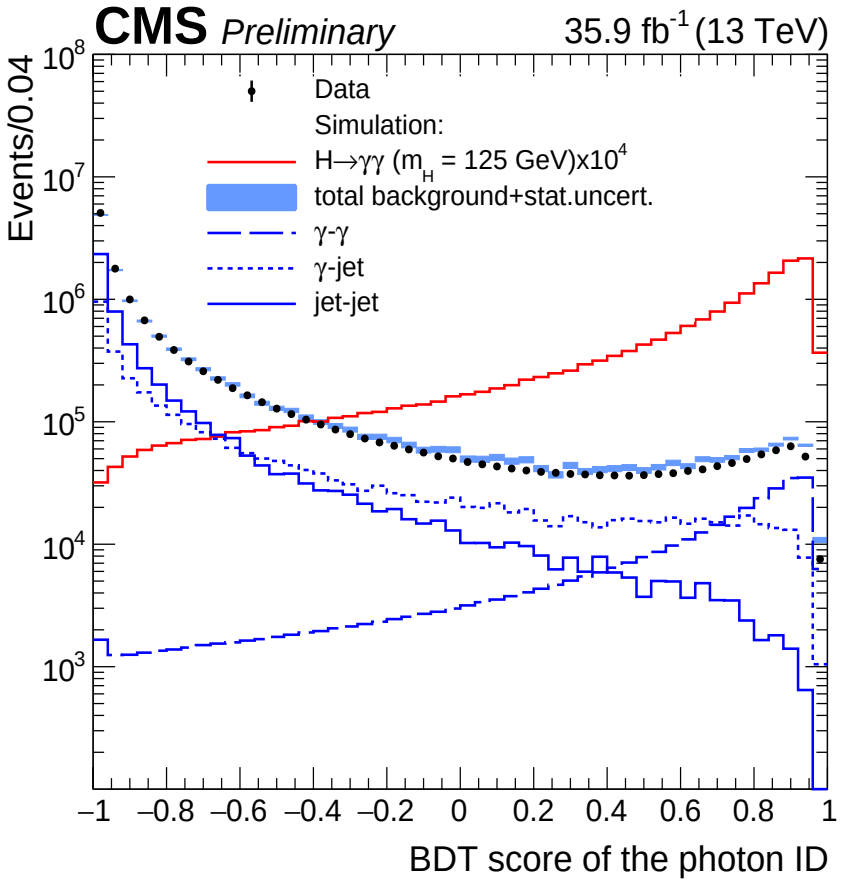
- I am a big fan of feedback: you'll receive in the next couple days a questionnaire
  - You'll receive it **at the email address you used for registering**
  - I'd be grateful if you could answer to the questions
  - There are also free fields for more articulated suggestions
- I will update the list of references of the last slide later today and reupload

Object ID

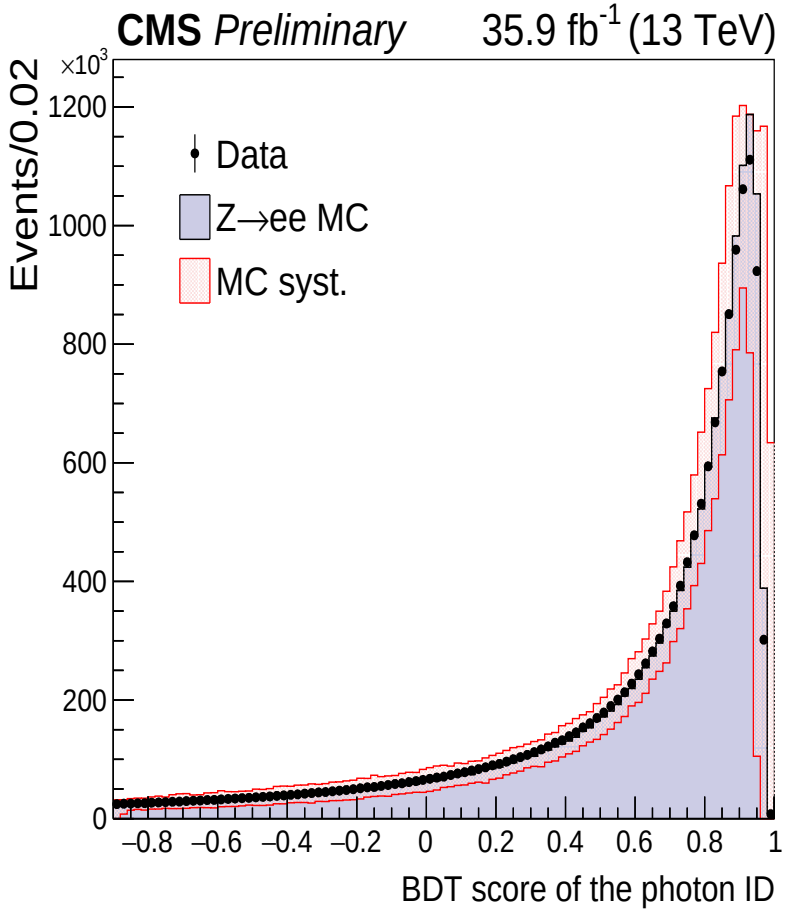
BDTs for object identification: the case of $\mathrm{H} \to \gamma\gamma$

- Object identification done with ML techniques since the Higgs discovery
- Classification problem (e.g. real photons vs objects misidentified as photons)

γ identification score for the lowest-score photons



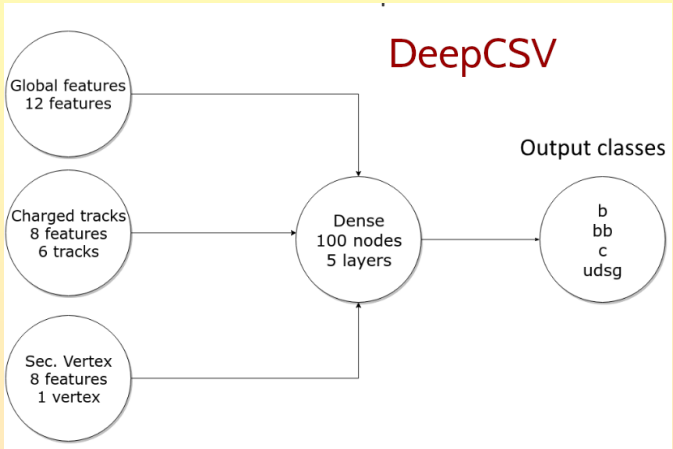
Validation in $\mathrm{Z} \to \mathrm{ee}$ events

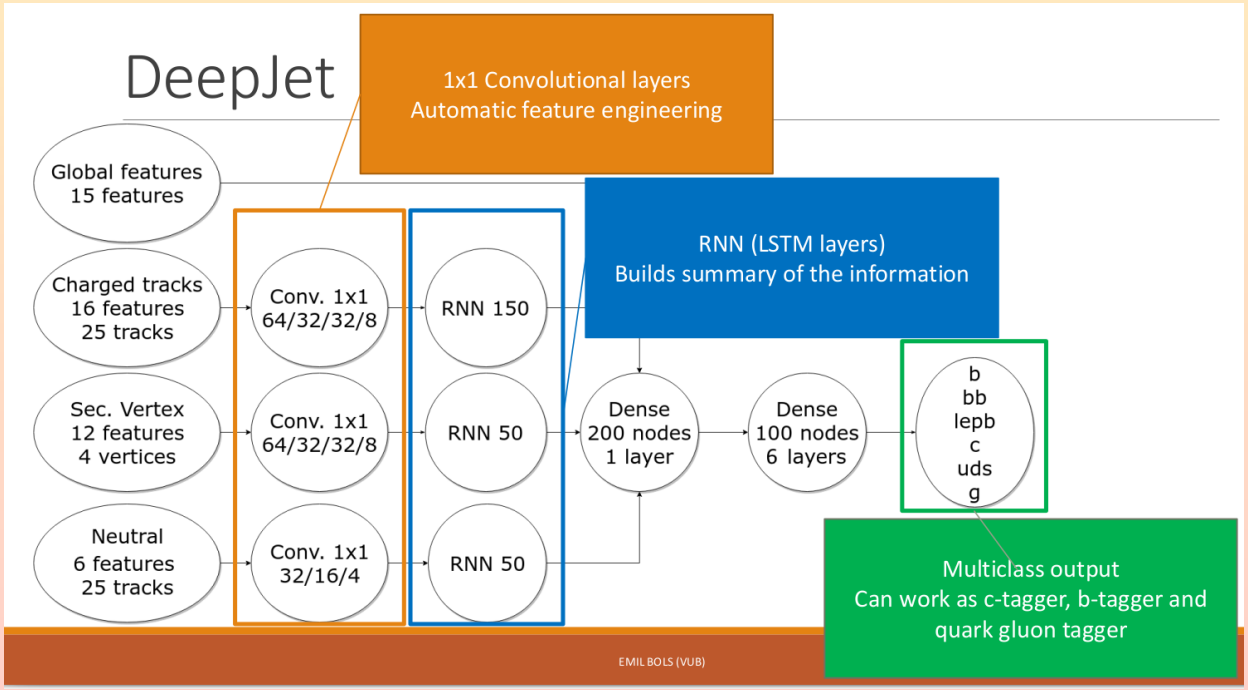


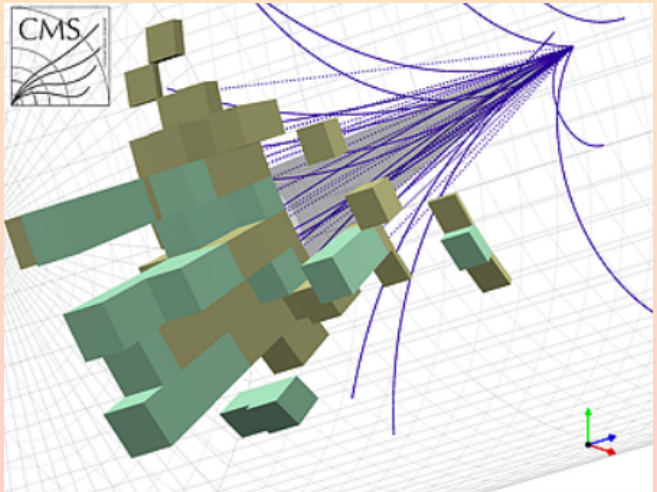
Plots from CMS-PAS-HIG-16-040

Object ID enters the era of mathematical representations — 1

- Identification of jets from bquarks (b tagging) at CMS
  - CSV (Run I and first part of Run II): BDT sensitive to the presence of secondary vertices
- DeepCSV: similar inputs, generic DNN
- Domain knowledge informs the choice of the better mathematical representation
  - Main criterion to choose the classification technique
- What's the best representation for jets?
  - Convolutional networks for images
  - Structure based on individual particles



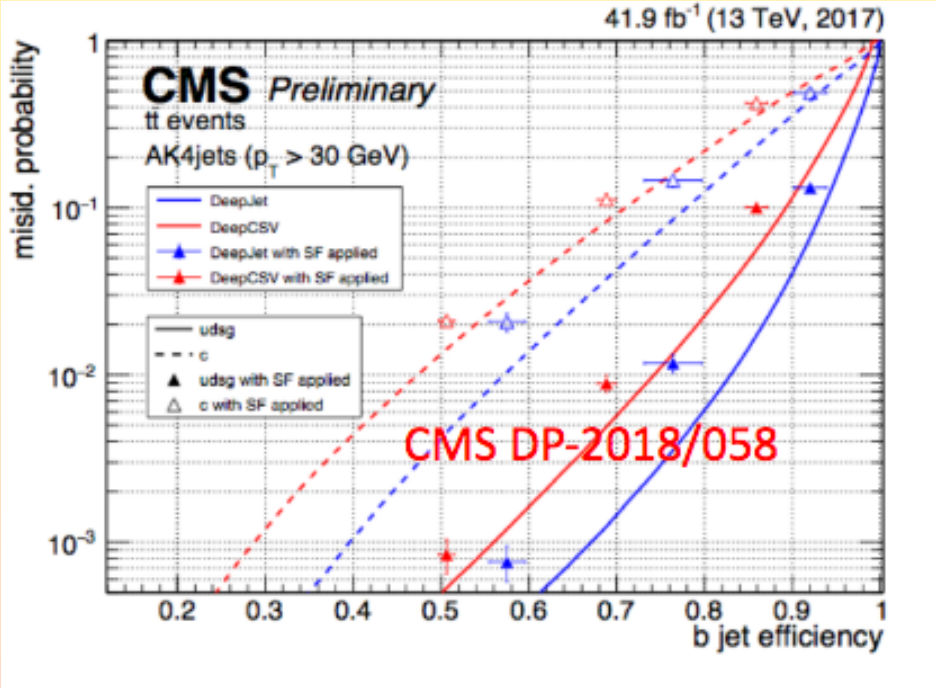


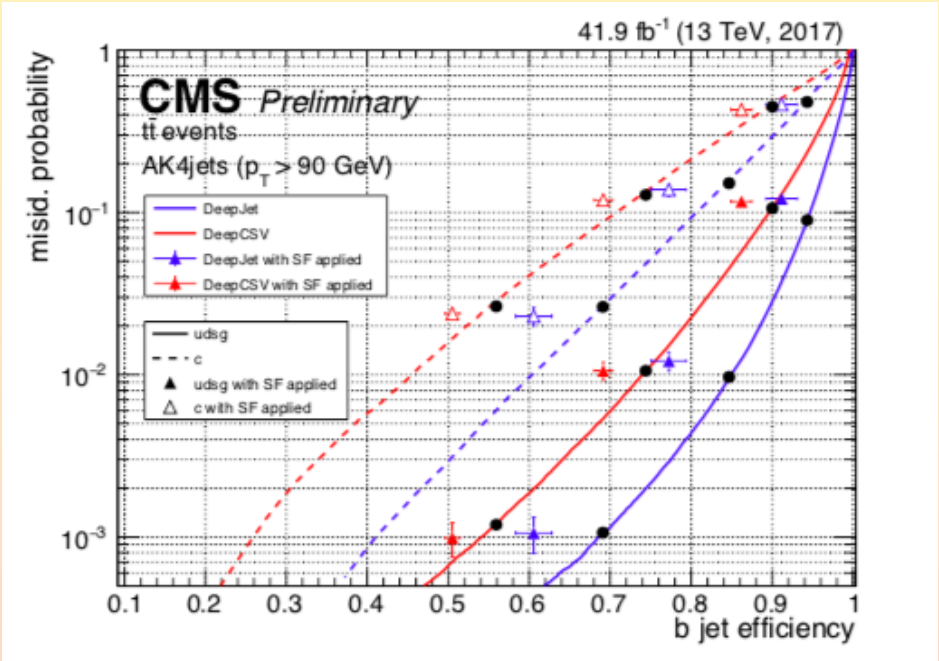


CMS DeepJet, plot from Emil Bols' talk at IML workshop

Object ID enters the era of mathematical representations — 2

- Clear gain even with respect to using a generic DNN (DeepCSV)

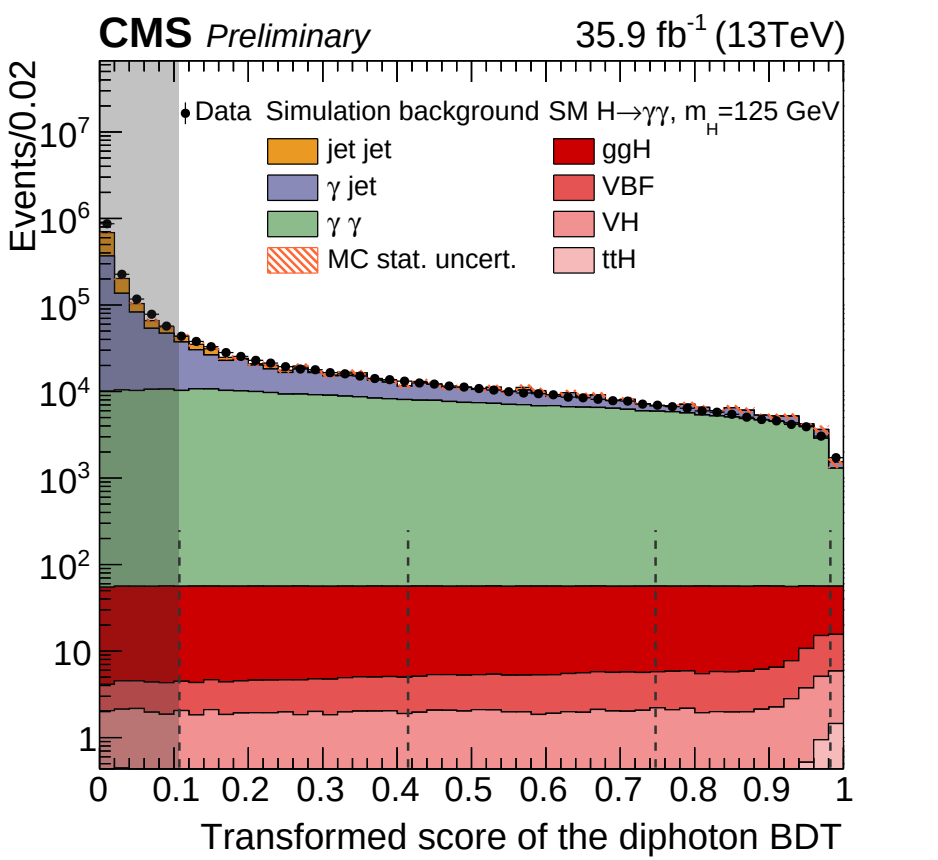


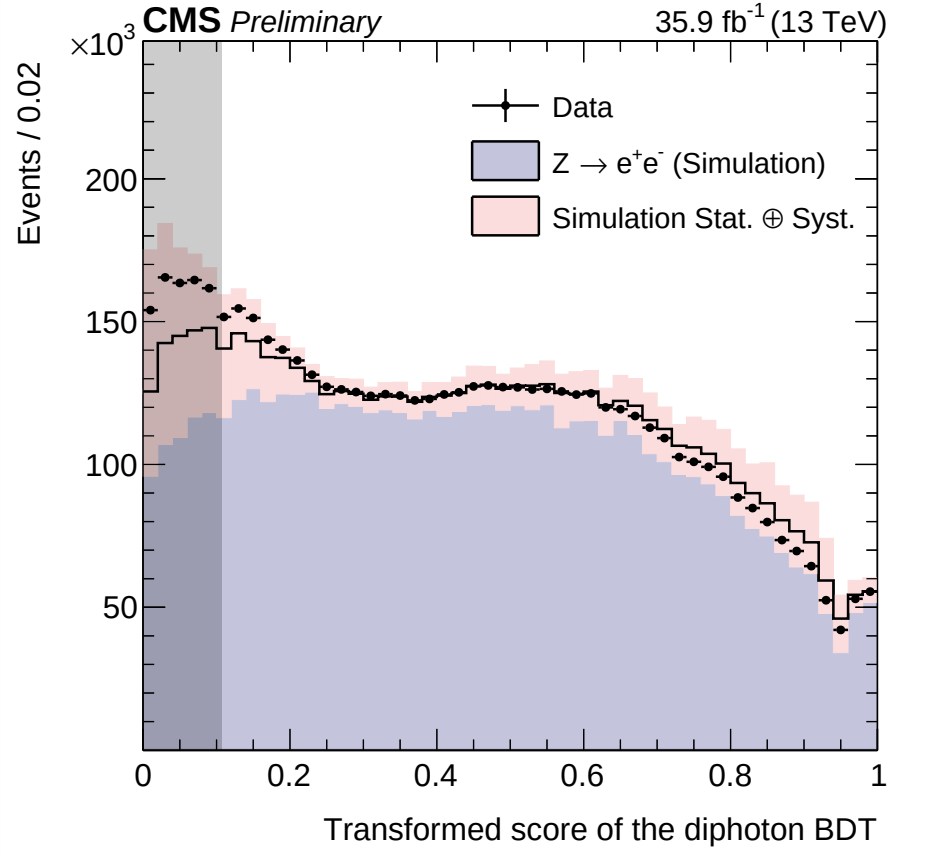


CMS DeepJet

Combining MVA ID for object identification

- Dedicated BDT, one score for each event, representing the mass resolution of the diphoton system
  - The photon ID BDT output is used as an input
  - High score for diphoton pairs with kinematic properties similar to signal, good mass resolution, and high individual γ ID score
- Validated in $\mathrm{Z} \to \mathrm{ee}$ events where electrons are reconstructed as photons

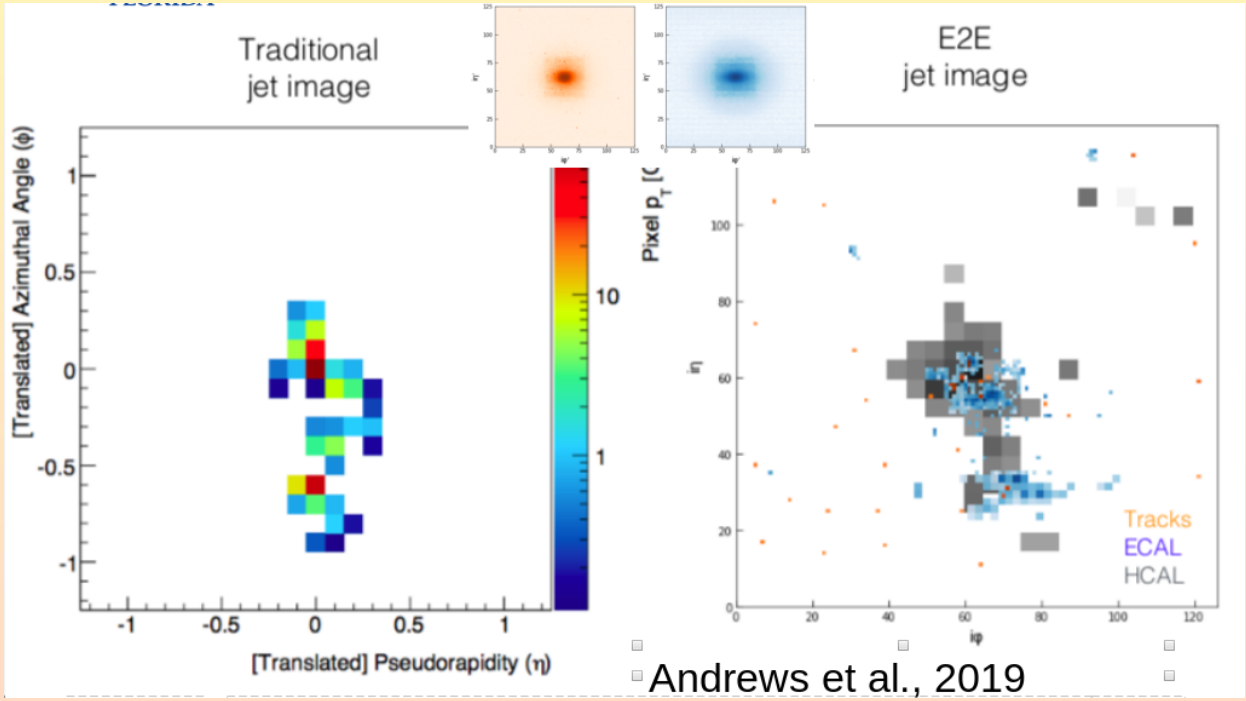




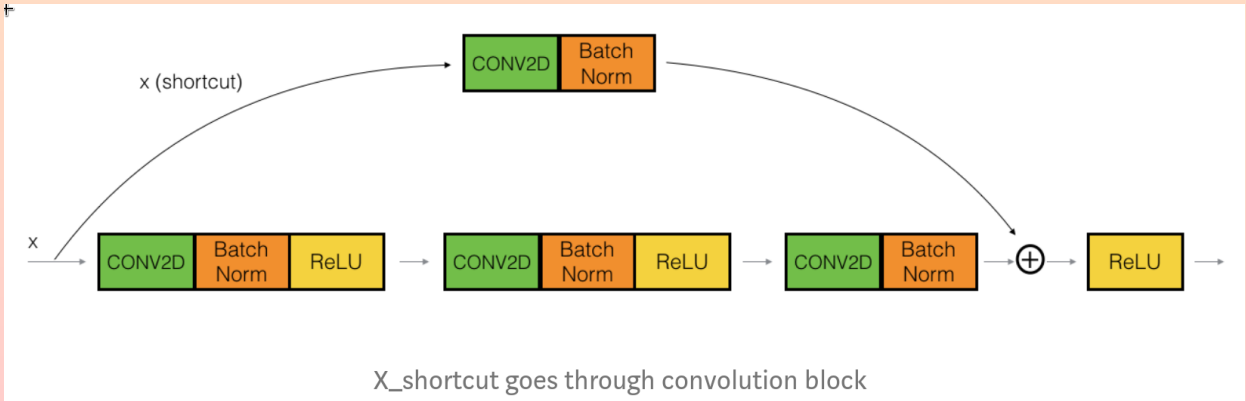
Plots from CMS-PAS-HIG-16-040

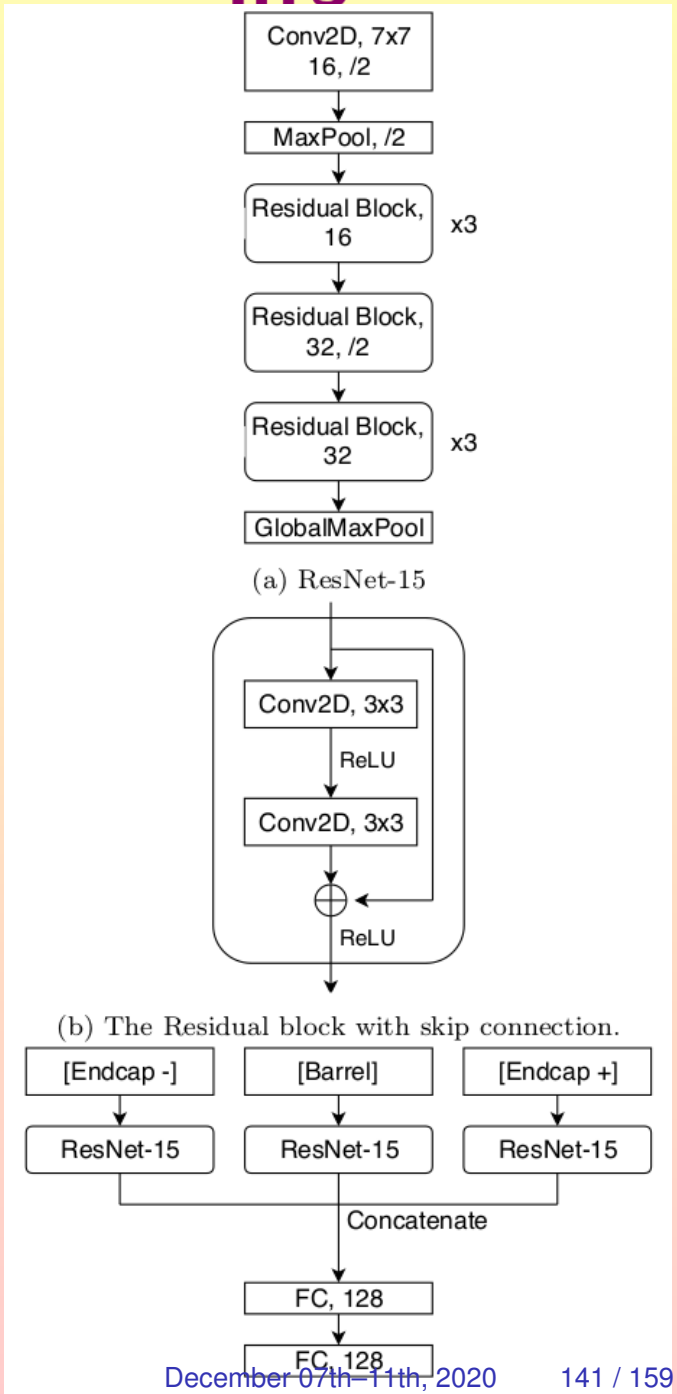
End-to-end reconstruction of jets

- Project detector layers in a single map
- Treat as an image: Res(idual)Net(works)
- Role of tracks in the reconstruction by the network is the same as we expect from the physics we know



Andrews et al., 2019





arXiv:1902.08276, S. Gleyzer's talk at 3rd IML workshop, Priya Dwivedi

Signal extraction

Separate signal from background using selection cuts

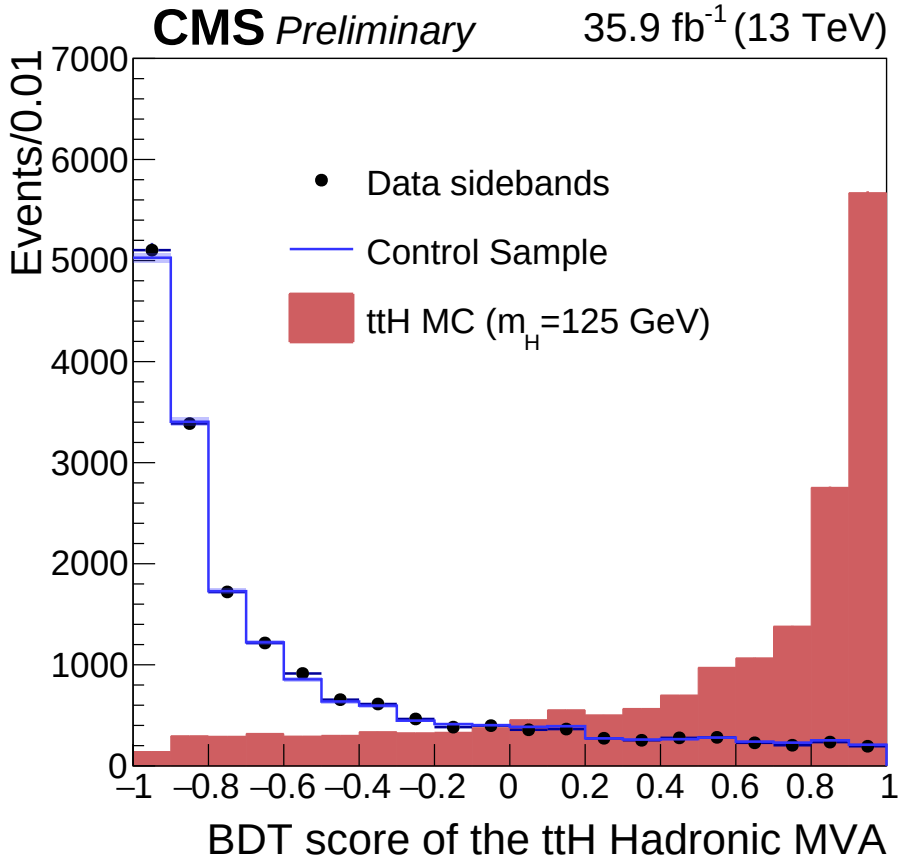
- High fraction of correct events in $t\bar{t}H$ categories by removing events from the dataset
- Delicate: removing events based on MVA output introduces tricky dependency on simulation
  - Dangerous, e.g. prevents from using unfolding results in comparisons with non-SM processes
- In both channels, remove events with low diphoton-BDT score
  - Threshold optimized simultaneously with $\gamma\gamma$-ID score, maximizing expected precision on signal strength

$t\bar{t}H$ leptonic

- ≥ 1 e/μ
- ≥ 2 jets
- ≥ 1 btagged jet

$t\bar{t}H$ hadronic

- ≥ 3 jets
- ≥ 1 btagged jet
- 0 e/μ
- BDT classifier (inputs: N_{jets}, $p_{\mathrm{T}}^{leadjet}$, lead and sublead btag scores)

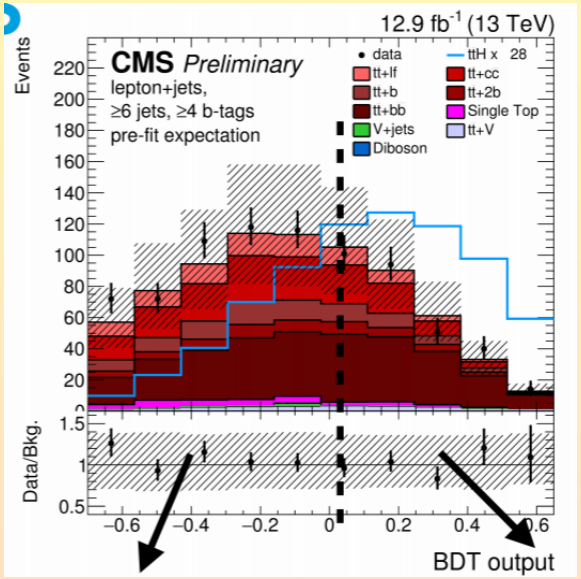


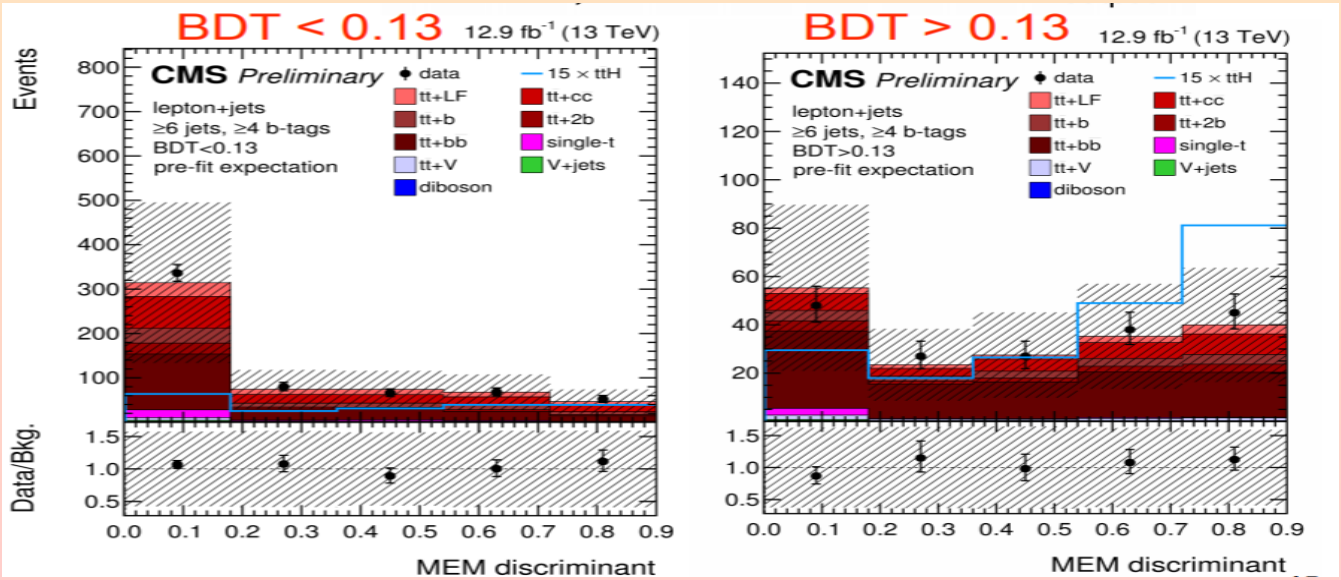
Evt Cat.	SM 125 GeV Higgs boson expected signal										
	Total	ggH	VBF	ttH	bbH	tHq	tHW	WH lep	ZH lep	WH had	ZH had
ttH Had.	5.85	10.99 %	0.70 %	77.54 %	2.02 %	4.13 %	2.02 %	0.09 %	0.05 %	0.63 %	1.82 %
ttH Lep.	3.81	1.90 %	0.05 %	87.48 %	0.08 %	4.73 %	3.04 %	1.53 %	1.15 %	0.02 %	0.02 %

Plots from CMS-PAS-HIG-16-040

Separate signal from background using all events

- Increase sensitivity by keeping the full MVA score distribution, possibly separating it into regions
  - Different fraction of signal/background
  - Constrain normalization or uncertainties in background-dominated regions

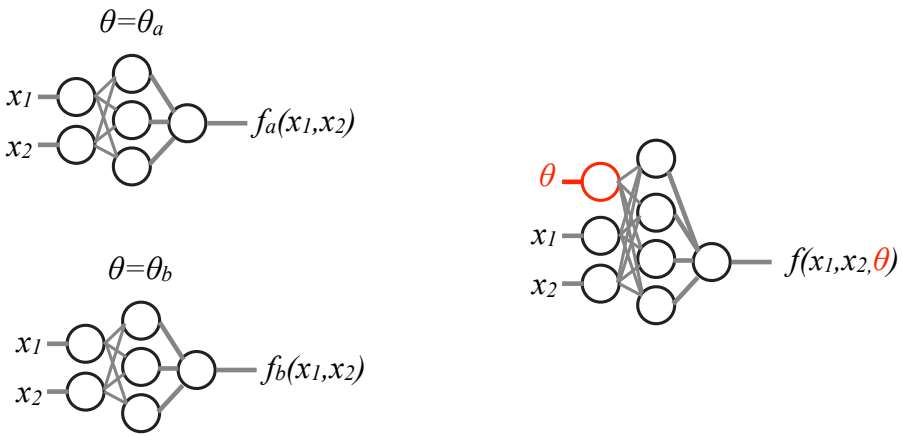


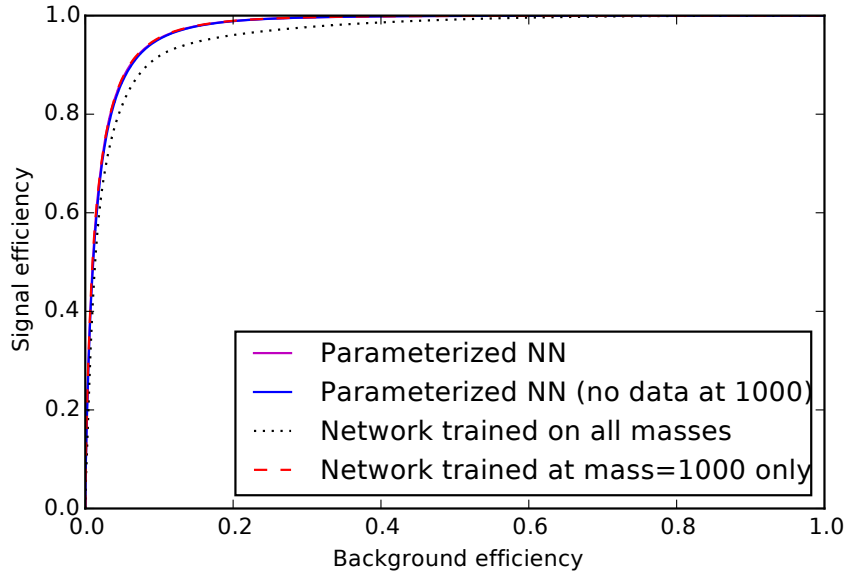


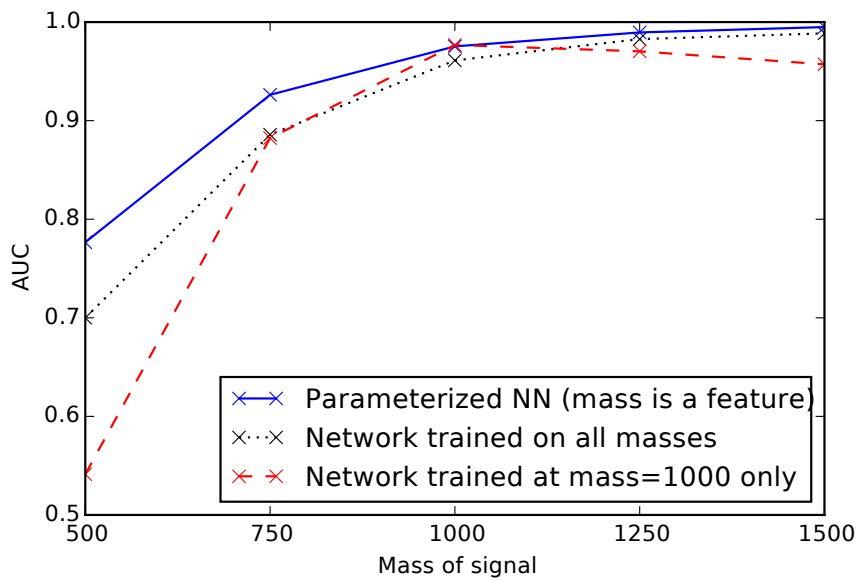
From $t\bar{t}H$ ($b\bar{b}$), CMS-PAS-HIG-16-004

Unknown parameters? Parameterized Machine Learning can help you!

- Classifier sensitive to the value of the parameter
  - Train using as an input the true value of the parameter (signal) or a random value (background)
  - Evaluate in slices at fixed values of the parameter
- Equal or better than training for individual values, and permits interpolation!
- We already use it!!
  - First application in: `CMS-HIG-17-006`
  - Recent application: CMS-HIG-18-004, arXiv:1908.09206 ☺



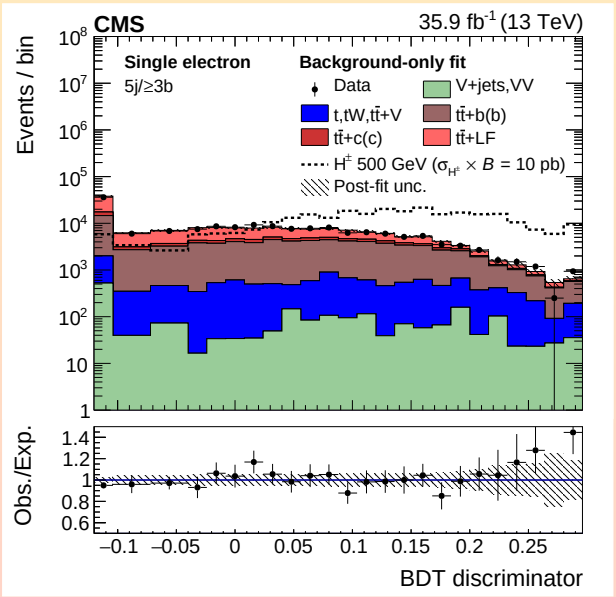


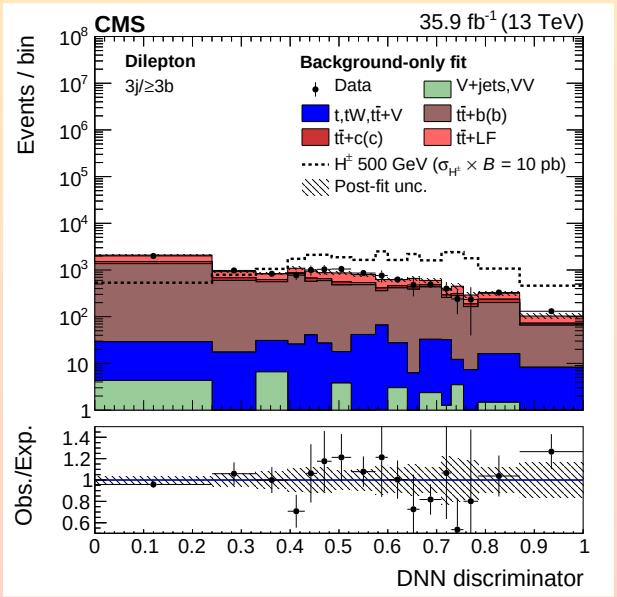


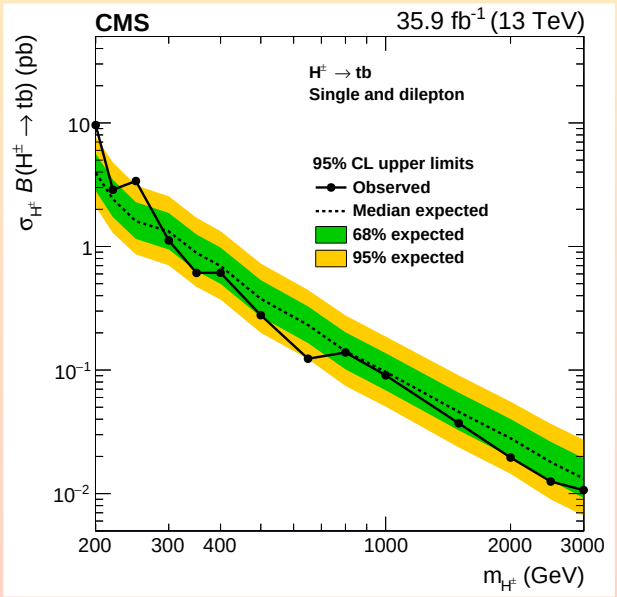
From Baldi *et al.* arXiv:1601.07913

Different techniques are “better” for different situations

- Each classification or regression problem is a distinct problem
  - Choice of the algorithm dictated e.g. by the structure of data and the complexity of the problem (network capacity)
- Sometimes not trivial: CMS-HIG-18-004, arXiv:1908.09206 ☺
  - 20–40% improvement w.r.t. single-variable result (H_T) usando BDT (single lepton) and parameterized DNN (dilepton)
  - DNN: more sensitive at low mass, where the BDT has not enough capacity to discriminate similar topologies ($t\bar{t}$ vs $H^{\pm}$)



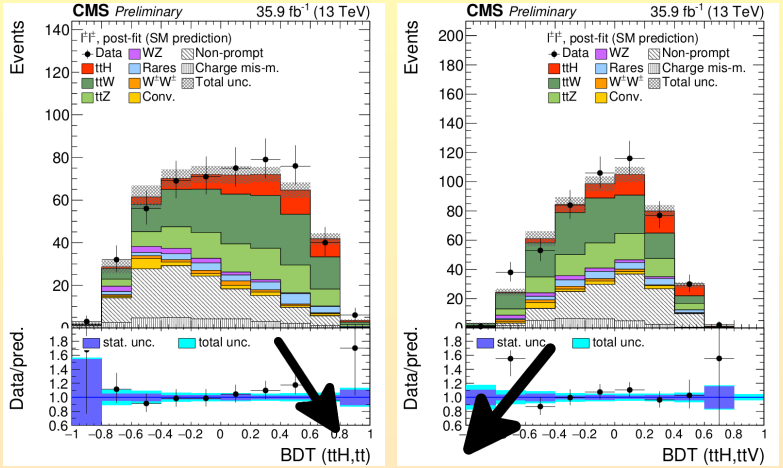




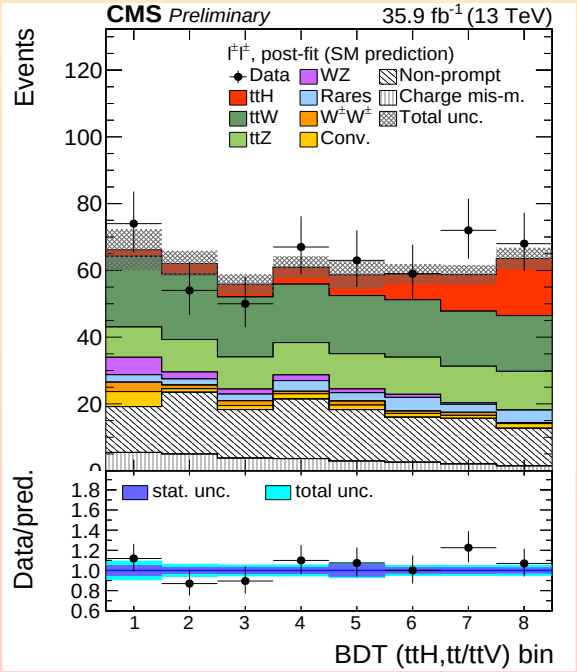
Reduce complexity: how many BDTs do you have?

- $t\bar{t}H$ multilepton: two different classifiers
  - BDT1: $t\bar{t}H$ vs $t\bar{t}$
  - BDT2: $t\bar{t}H$ vs $t\bar{t}V$
- Finely partition the 2D plane (BDT1, BDT2)
  - Use a training sample to calculate binning
  - Apply to the application sample used for inference
- Define the target N_{bins} with clustering techniques (k-means)
- Finally separate regions based on empirical likelihood
  - Likelihood ratio approximated by $\frac{S}{B}$
  - Ordering from the Neyman-Pearson lemma
  - Quantile-based binning

BDT classifier output (2LSS)



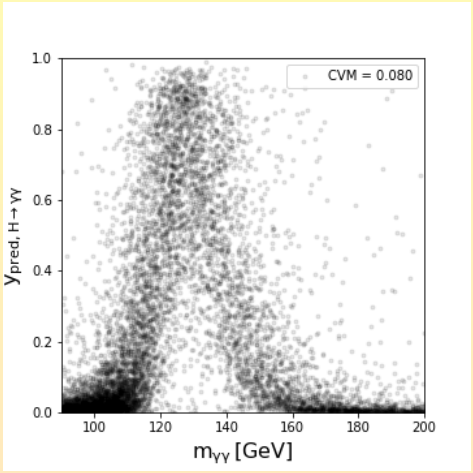
Final 1D discriminator (2LSS)

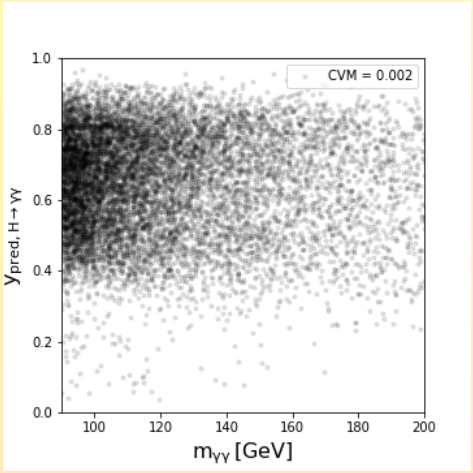


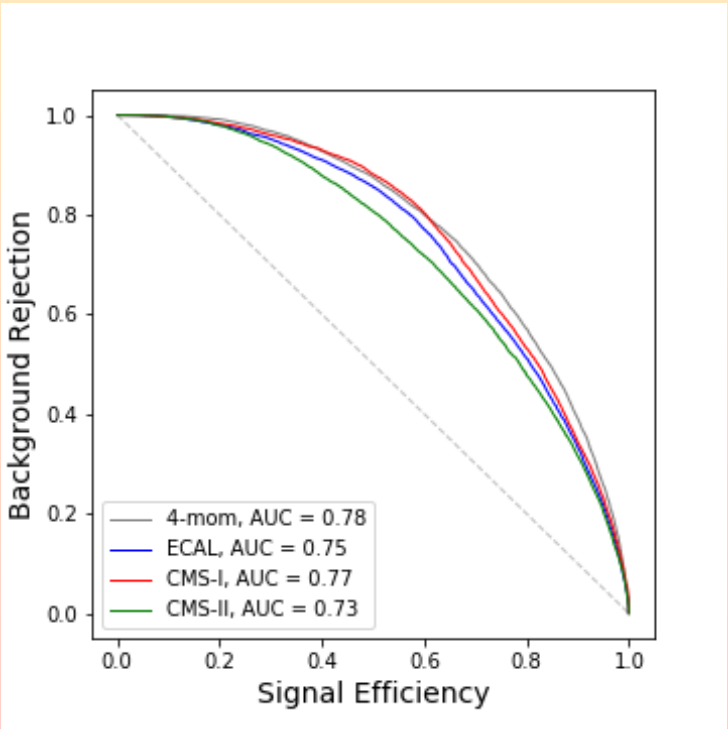
CMS-PAS-HIG-17-004, part of CMS-HIG-17-018: evidence for ttH production in multilepton final states

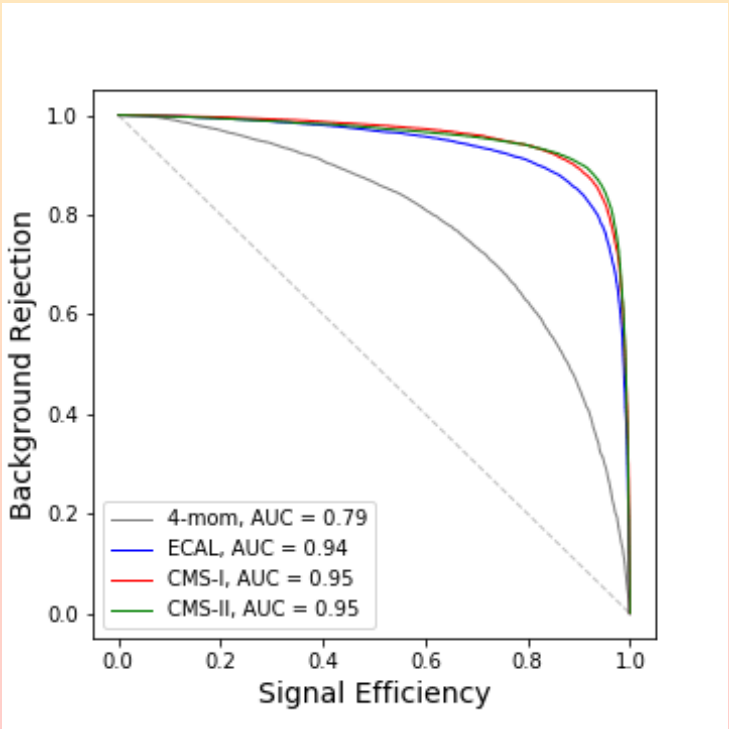
End-to-end event classification

- Low-level data representation
  - Tracker, electromagnetic calorimeter, hadronic calorimeter
  - Various possible geometries
- Mass decorrelation to avoid structure sculpting
  - Transform $E_{\gamma\gamma}$ in units of $M_{\gamma\gamma}$
  - Extension of pivoting technique
- Training with a 3-classes ResNet ($H \to \gamma\gamma$, $\gamma\gamma$, γ+jet)
- Statistically-limited technique









From arXiv:1807.11916

What if you don’t know your signal?

Gaussian processes)

- Multivariate gaussian associated to a set of random variables ($N_{dim} = N_{random\ variables}$)
  - *Kernel* as a similarity measure between bin centers (counts) and a *averaging function*

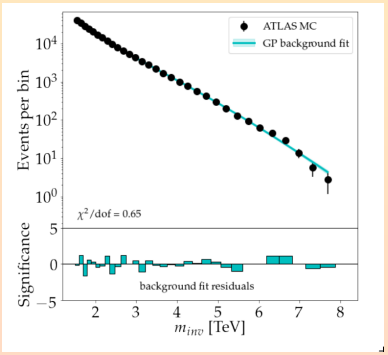
$$\mu(x) = 0\,, \tag{9}$$

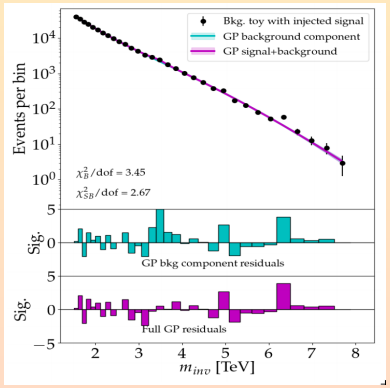
$$\Sigma_B(x,x') = A\ \exp\left(\frac{d-(x+x')}{2a}\right)\sqrt{\frac{2l(x)l(x')}{l(x)^2+l(x')^2}}\exp\left(\frac{-(x-x')^2}{l(x)^2+l(x')^2}\right), \tag{10}$$

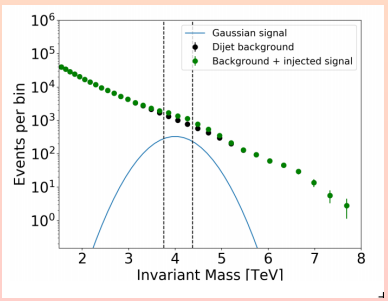
$$\Sigma_S(x,x') = C\ \exp\left(-\frac{1}{2}(x-x')^2/k^2\right)\exp\left(-\frac{1}{2}\left((x-m)^2+(x'-m)^2\right)/t^2\right), \tag{11}$$

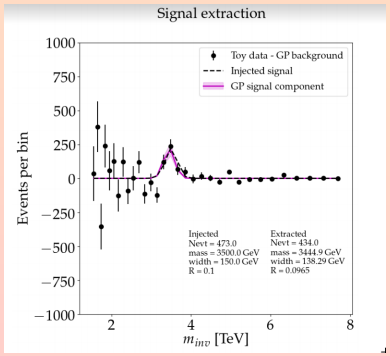
- Signal is not parameterized
- Hyperparameters fixed by the B-only fit

- S: residual of B-subtraction





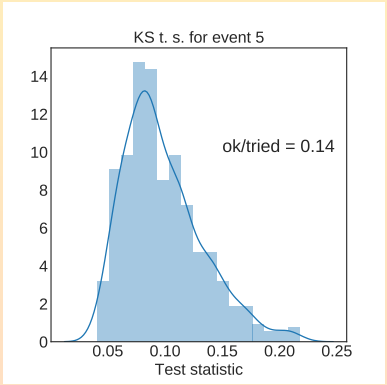


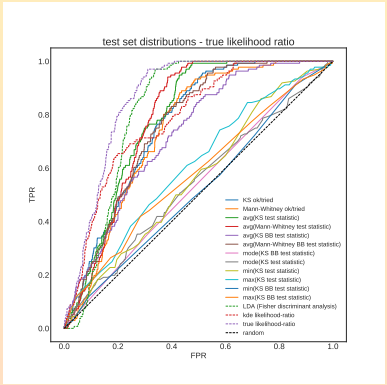


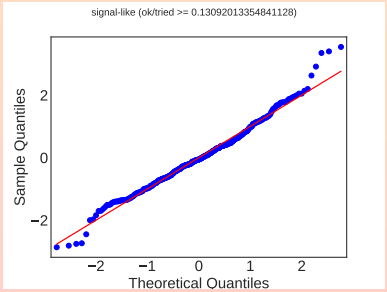
AMVA4NewPhysics deliverable 2.5 public report

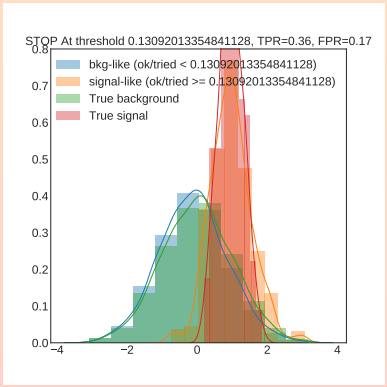
Inverse Bagging

- Data: mixture model with small S
- Classification based on sample properties
  - Compare bootstrapped samples with reference (pure B)
  - Use Metodiev theorem to translate inference into signal fraction
- Validate with LR y LDA
  - Promising results







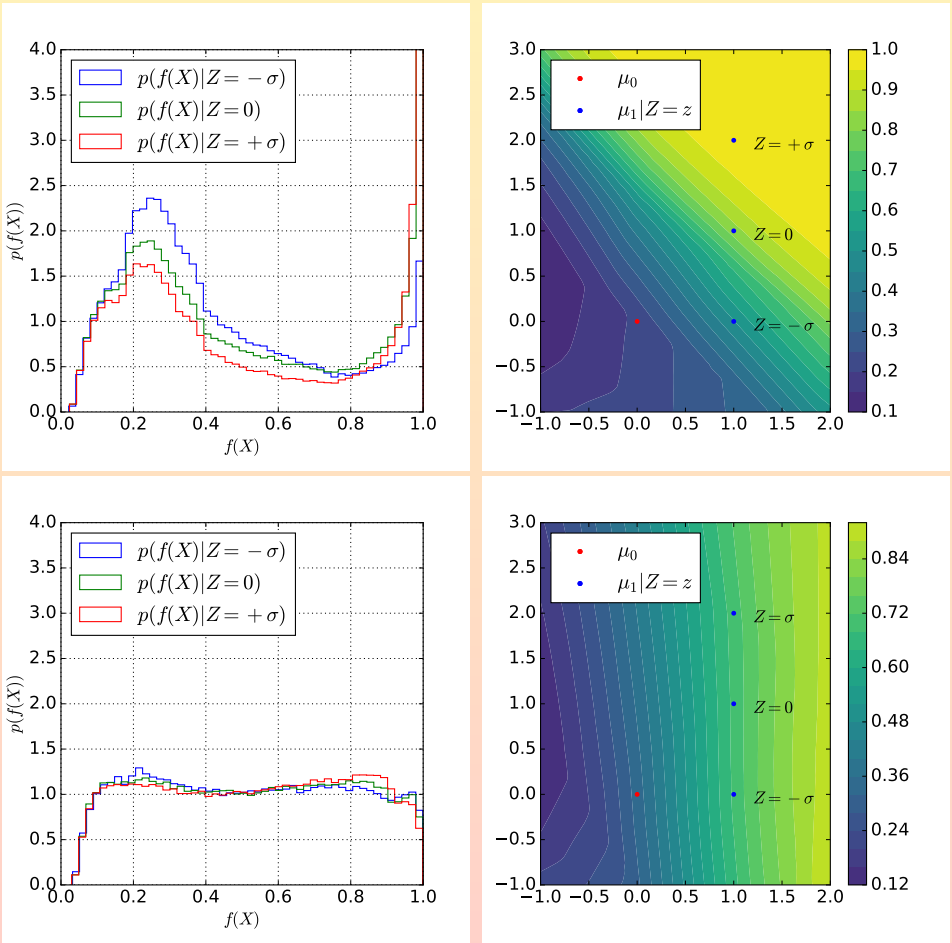


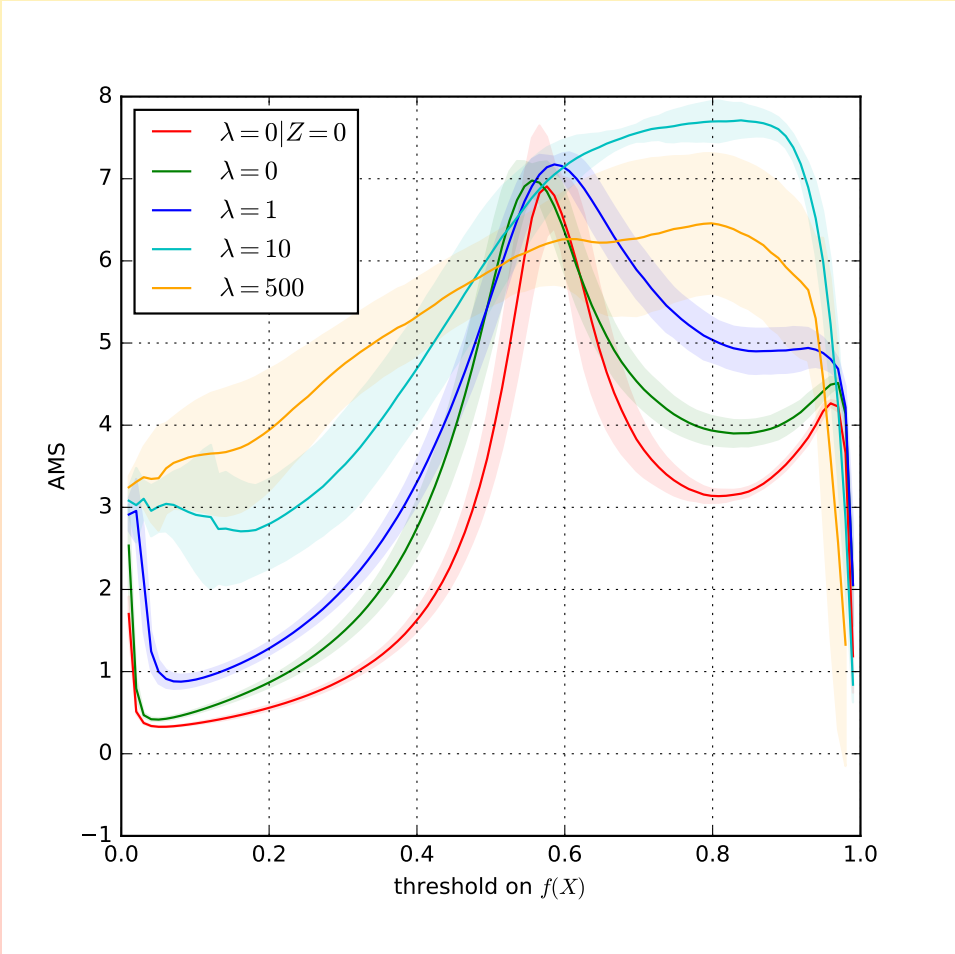
Vischia-Dorigo arXiv:1611.08256, doi:10.1051/epjconf/201713711009, and P. Vischia's talk at EMS2019

What about the uncertainties?

Can we reduce the impact of uncertainties on our results?

- Adversarial networks used to build pivot quantities
  - Quantities invariant in some parameter (typically nuisance parameter representing an uncertainty)
- Best Approximate Mean Significance as tradeoff optimal/pivotal
  $E_\lambda(\theta_f, \theta_r) = \mathcal{L}_f(\theta_f) - \lambda \mathcal{L}_r(\theta_f, \theta_r)$





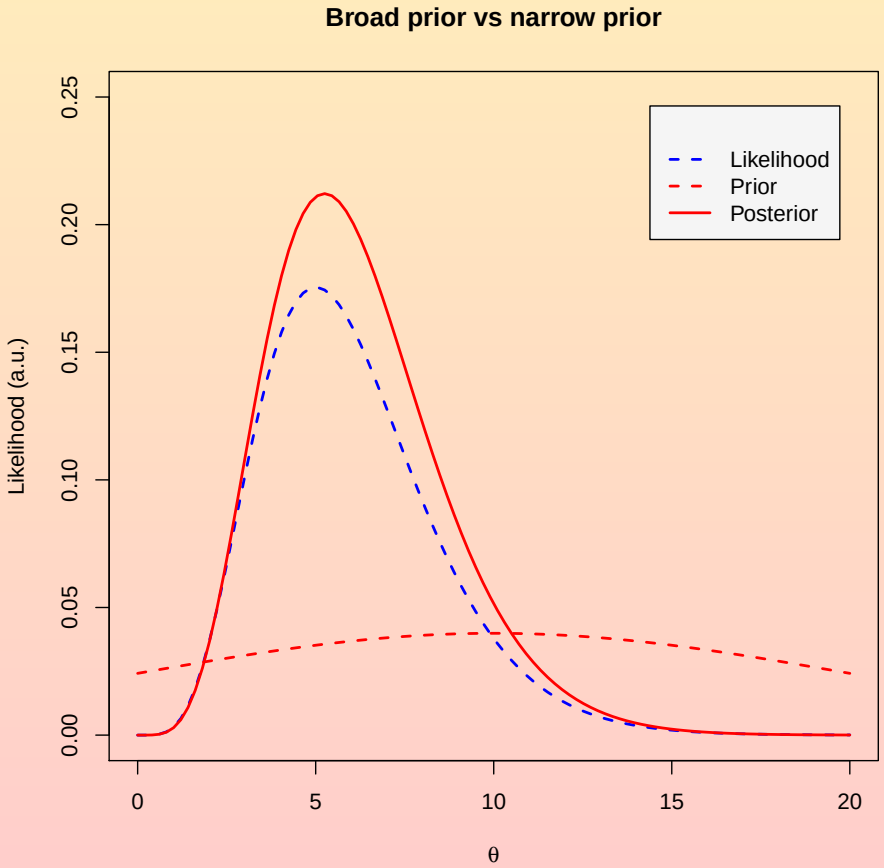
From Louppe-Kagan-Cranmer, arXiv:1611.01046

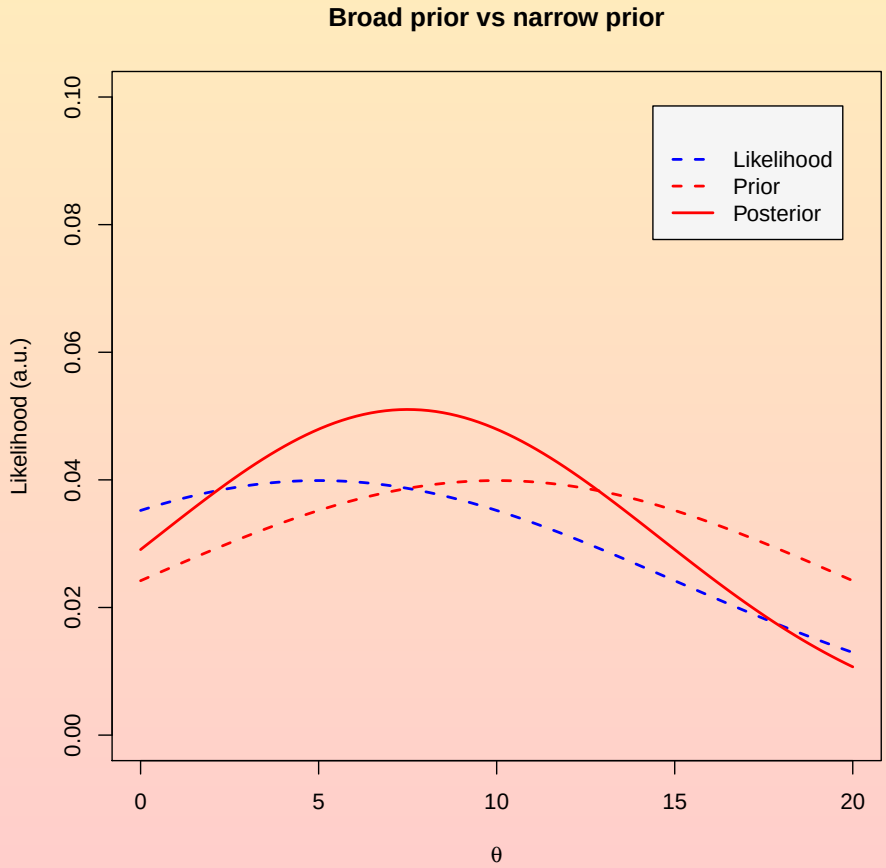
Reminder: likelihood function and Fisher information

- The (second) derivative of the likelihood function is connected to the quantity of information you can extract from data

$$I(\theta) = -E\left[\frac{\partial^2 lnL}{\partial\theta^2}\right] = E\left[\left(\frac{\partial lnL}{\partial\theta}\right)^2\right]$$

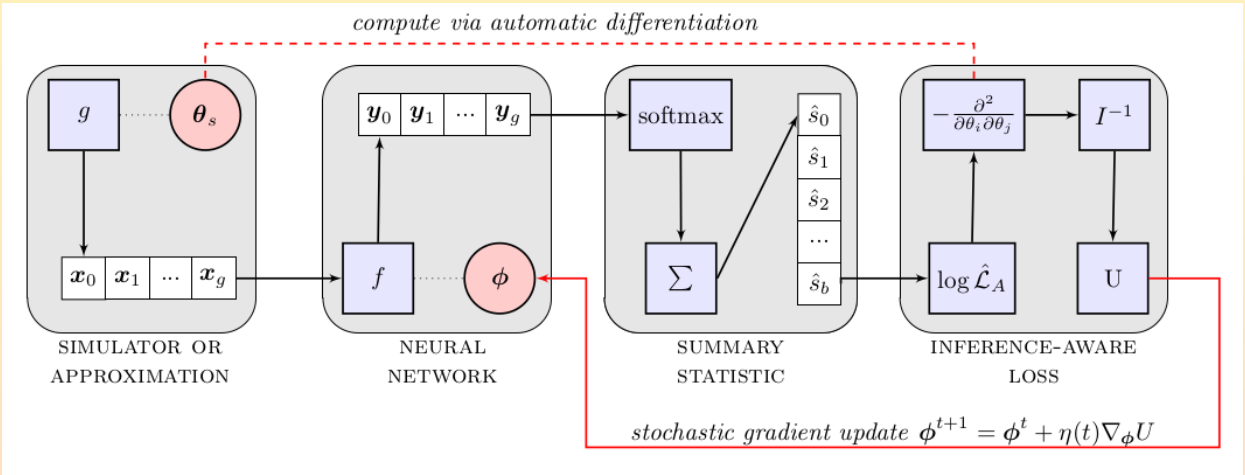
- The likelihood function contains all the information that you can extract from data on the parameter θ
- A narrow likelihood function carries more information than a broader one

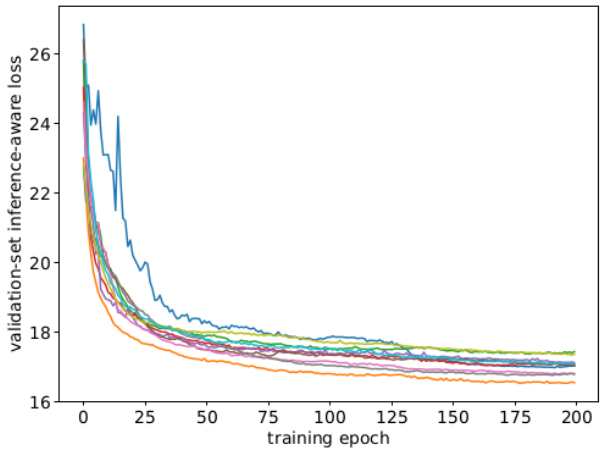




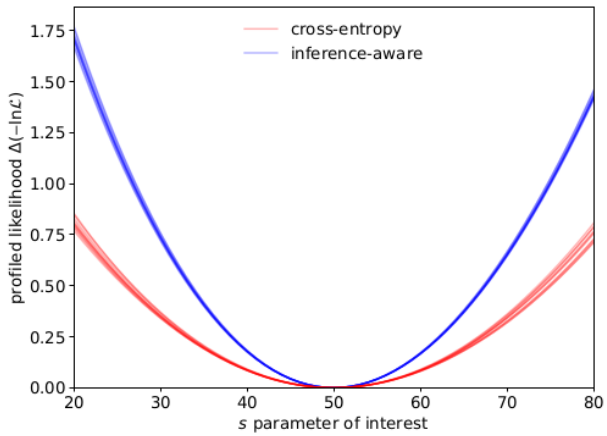
INFERNO: inference-aware neural optimization

- Build non-parametric likelihood function based on simulation, use it as summary statistic
- Minimize the expected variance of the parameter of interest
  - Obtain the Fisher information matrix with automatic differentiation, and use it as loss function
  - For (asymptotically) unbiased estimators, Rao-Cramér-Frechet (RCF) bound $V[\hat{\theta}] \sim \frac{1}{\hat{\theta}}$ (see my Monday lesson)
  - Constraints via auxiliary measurements (typically on nuisance parameters) included in covariance matrix out of the box





(a) inference-aware training loss



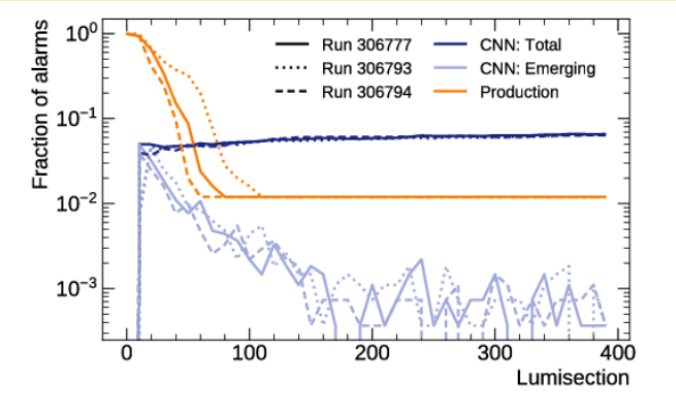
(b) profile-likelihood comparison

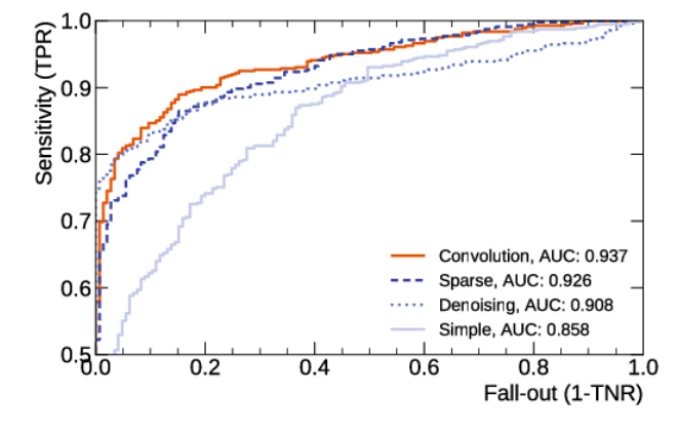
From De Castro-Dorigo, arXiv:1806.04743, and AMVA4NewPhysics deliverable 1.4 public report

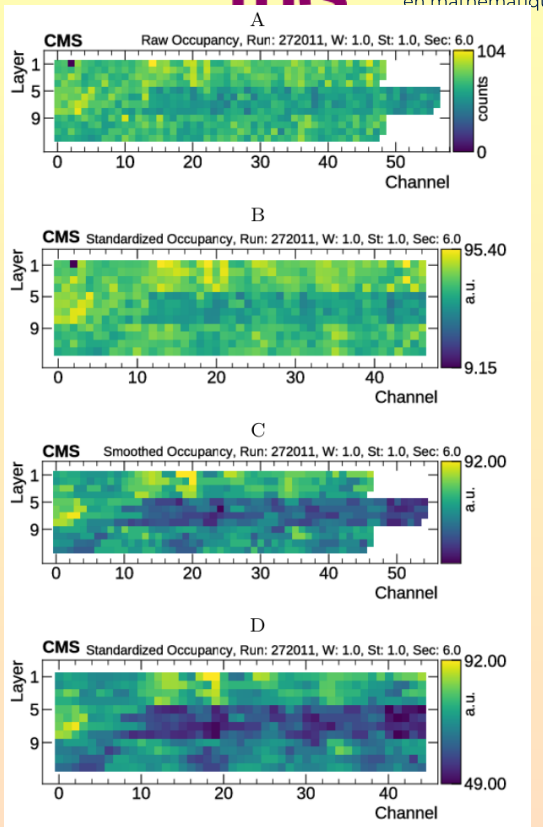
Which data should we take?

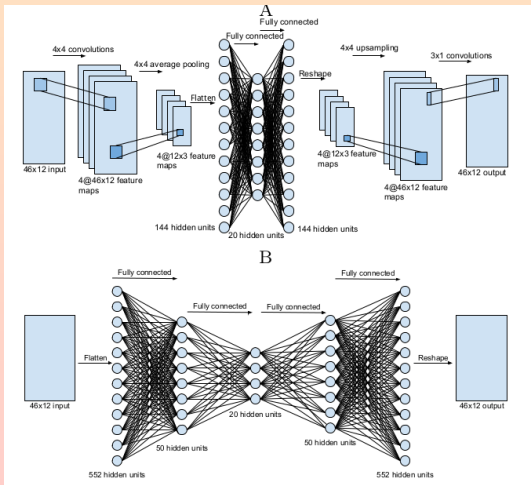
What if we don't know which data to take?

- Represent data as geographically-organized images
  - Local focus: detector layers treated independently
  - Regional focus: detector layers treated independently but simultaneously (spot problems between layers)
- Autoencoders (noise detection, dimensionality reduction)
  - Encode the inputs to the hidden layer
  - Decode the hidden layer to an approximate representation of the inputs



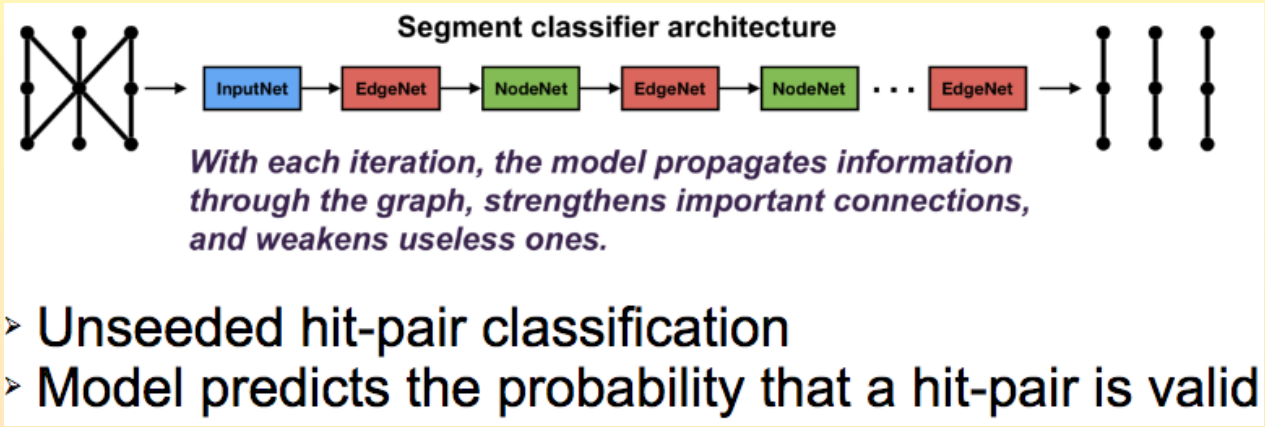


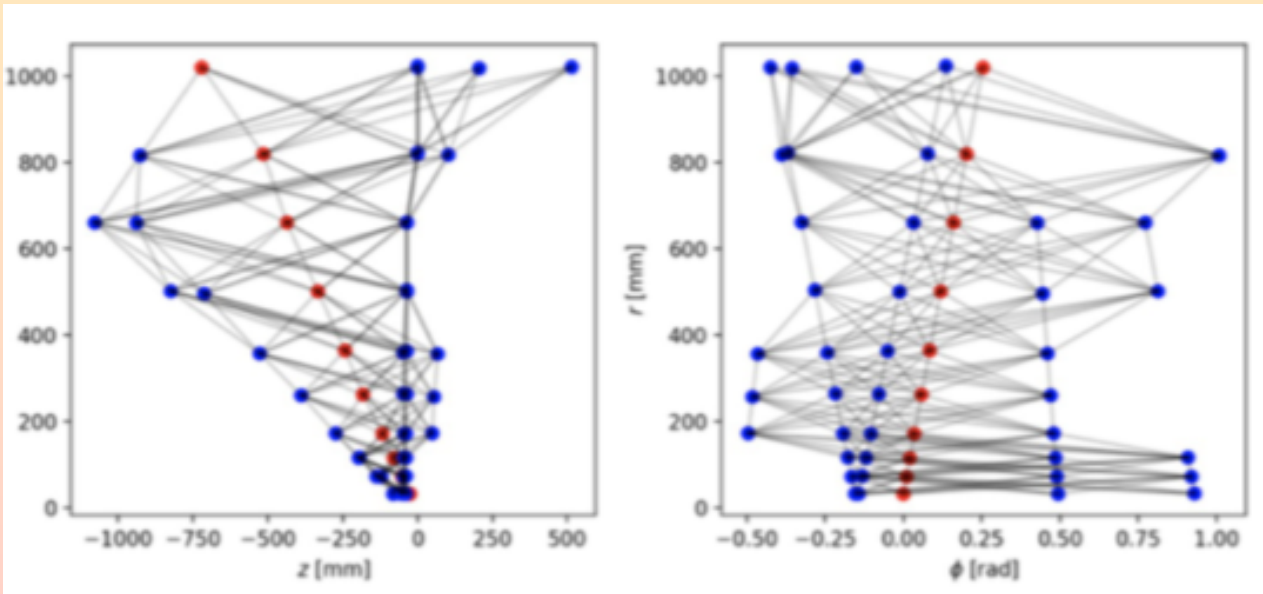


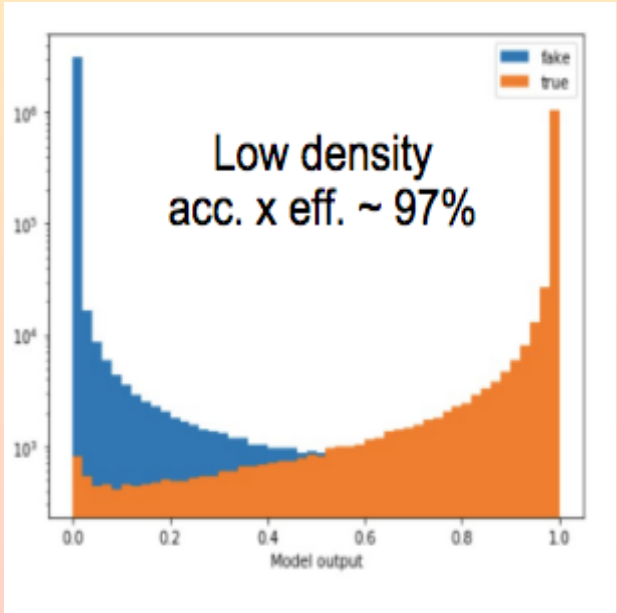


From arXiv:1808.00911

- Graph networks to literally connet the dots



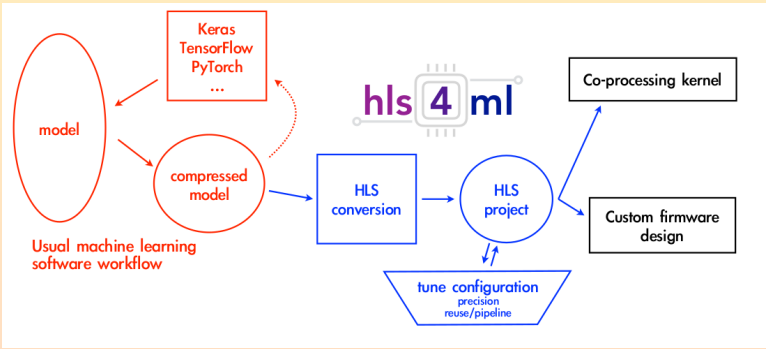


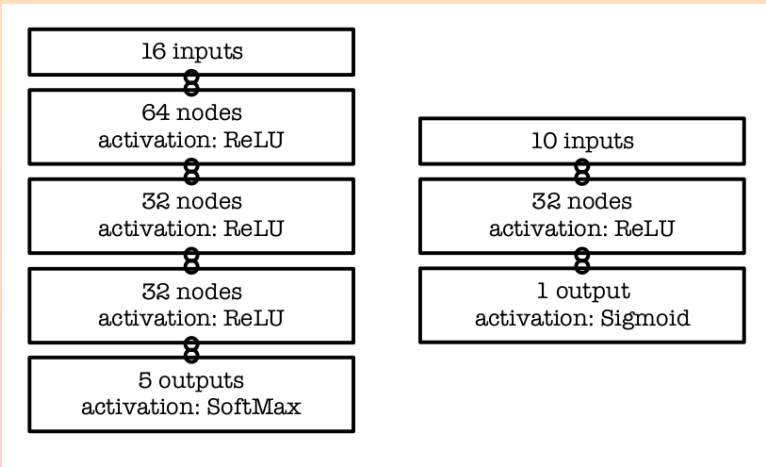


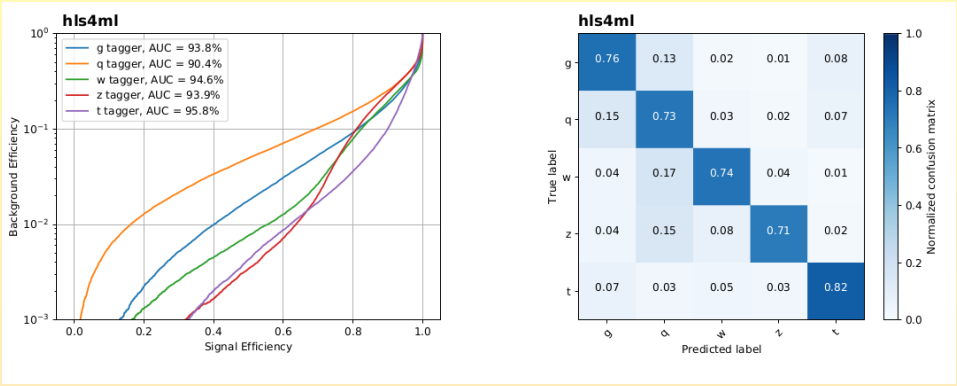
The HEP.TrkX project, S. Gleyzer's talk at 3rd IML workshop

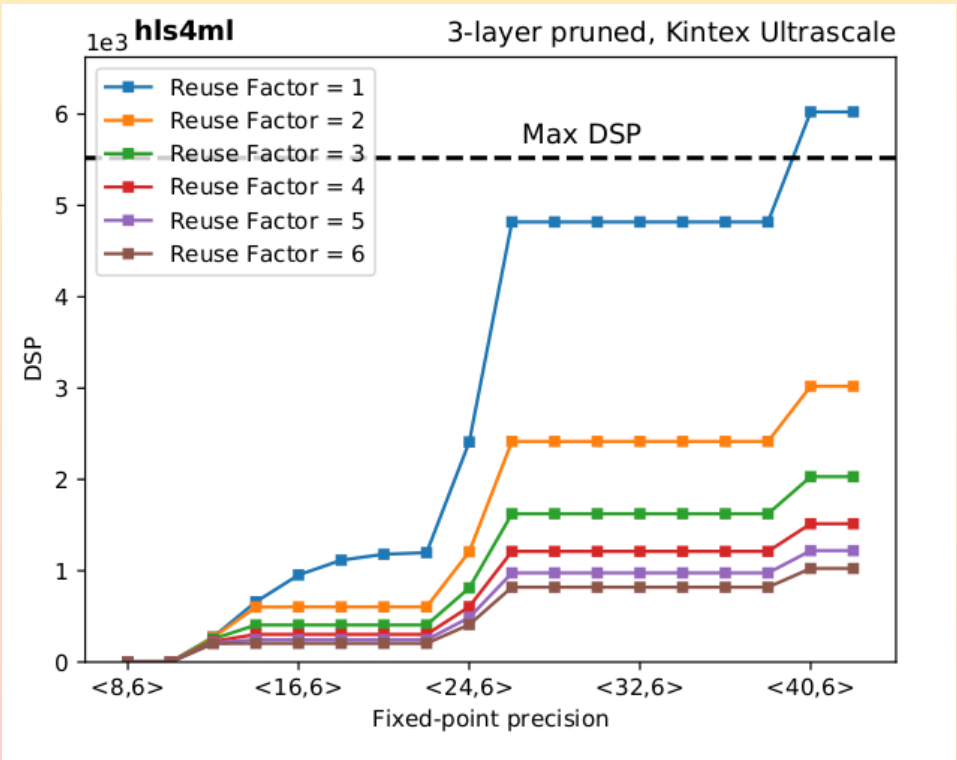
What if you need to do it quickly?

- Real-time event processing requires low-latency and low-power-consumption hardware: FPGAs
- Case study: classify structures inside jets (jet substructure)
- Compression, quantization, parallelization digital signal processing (arithmetic) blocks (DSPs),









From arXiv:1804.06913

- Frederick James: Statistical Methods in Experimental Physics - 2nd Edition, World Scientific
- Glen Cowan: Statistical Data Analysis - Oxford Science Publications
- Louis Lyons: Statistics for Nuclear And Particle Physicists - Cambridge University Press
- Louis Lyons: A Practical Guide to Data Analysis for Physical Science Students - Cambridge University Press
- E.T. Jaynes: Probability Theory - Cambridge University Press 2004
- Annis?, Stuard, Ord, Arnold: Kendall's Advanced Theory Of Statistics I and II
- Pearl, Judea: Causal inference in Statistics, a Primer - Wiley
- R.J.Barlow: A Guide to the Use of Statistical Methods in the Physical Sciences - Wiley
- Kyle Cranmer: Lessons at HCP Summer School 2015
- Kyle Cranmer: Practical Statistics for the LHC - http://arxiv.org/abs/1503.07622
- Roberto Trotta: Bayesian Methods in Cosmology - https://arxiv.org/abs/1701.01467
- Harrison Prosper: Practical Statistics for LHC Physicists - CERN Academic Training Lectures, 2015 https://indico.cern.ch/category/72/
- Christian P. Robert: The Bayesian Choice - Springer
- Sir Harold Jeffreys: Theory of Probability (3rd edition) - Clarendon Press
- Harald Crámer: Mathematical Methods of Statistics - Princeton University Press 1957 edition

Backup