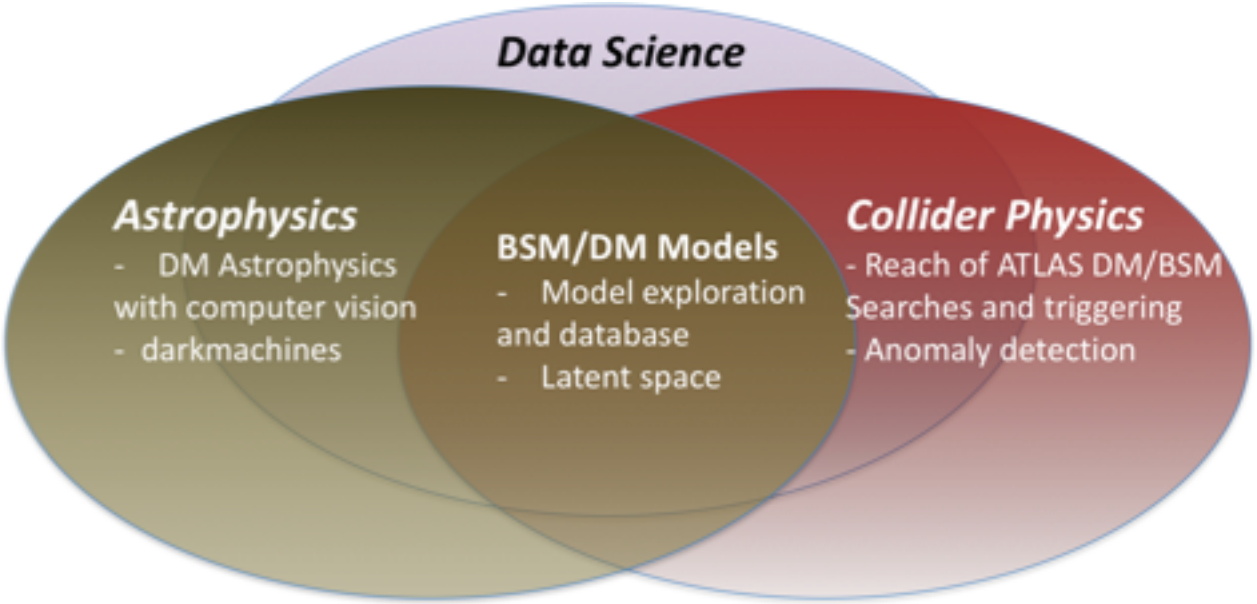
Accelerating searches for new physics (and Dark Matter) with Machine Learning

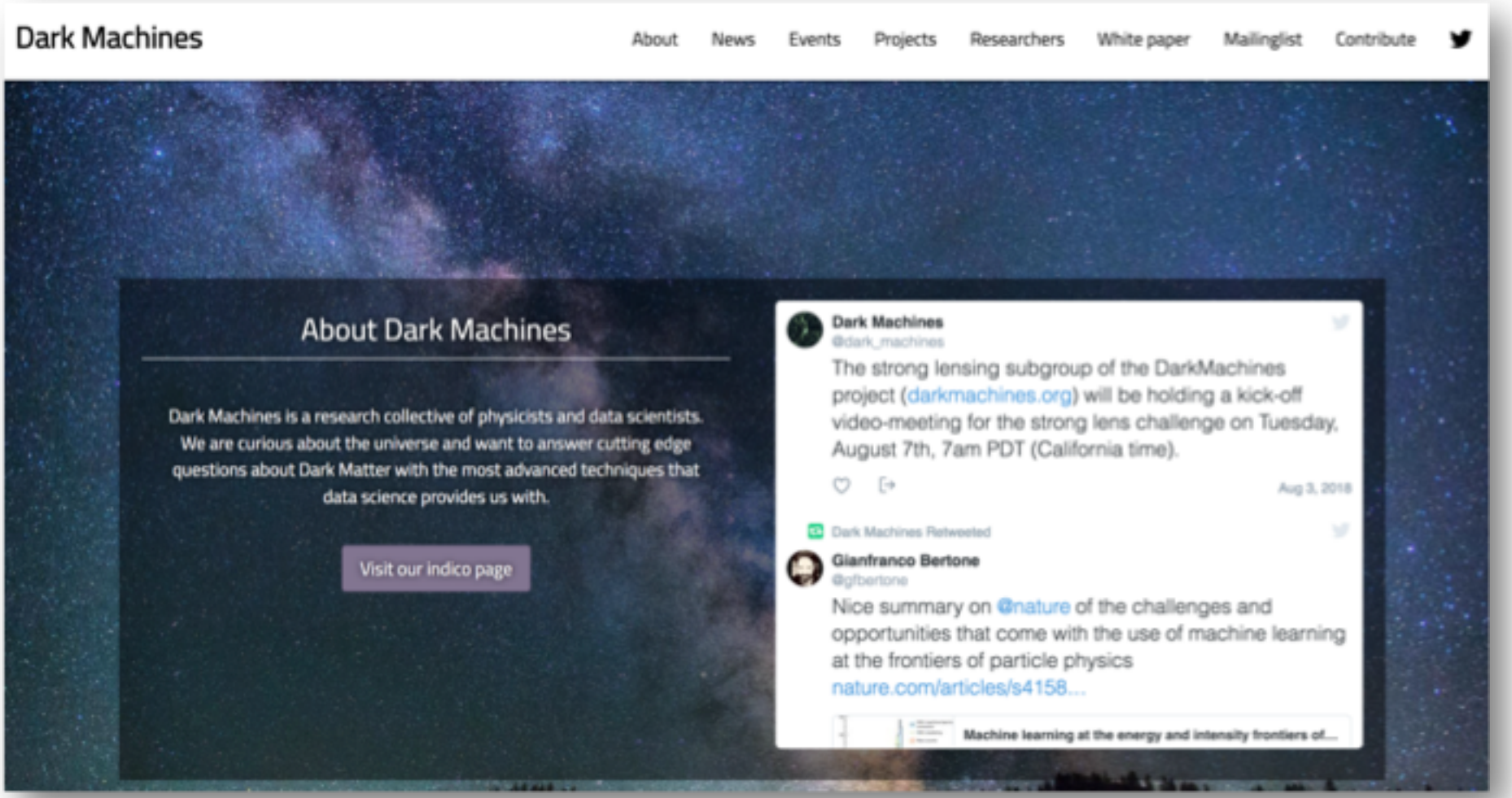
Sascha Caron
(Radboud University and Nikhef)

Idea of the talk

- Improve your big-picture, i.e. discuss overview, new ideas, discussion
- Generate ideas to use your technical skills
- No technical details

Who am I ?
Physics beyond the SM



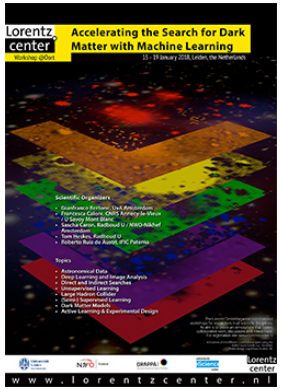

Dark Machines
About
News
Events
Projects
Researchers
White paper
Mailinglist
Contribute
About Dark Machines
Dark Machines is a research collective of physicists and data scientists. We are curious about the universe and want to answer cutting edge questions about Dark Matter with the most advanced techniques that data science provides us with.
Visit our indico page
Dark Machines
@dark_machines
The strong lensing subgroup of the DarkMachines project (darkmachines.org) will be holding a kick-off video-meeting for the strong lens challenge on Tuesday, August 7th, 7am PDT (California time).
Aug 3, 2018
Dark Machines Retweeted
Gianfranco Bertone
@gfbertone
Nice summary on @nature of the challenges and opportunities that come with the use of machine learning at the frontiers of particle physics nature.com/articles/s4158...
Machine learning at the energy and intensity frontiers of...

Why darkmachines ?

The darkmachines initative is a result of the workshop:

Accelerating the Search for Dark Matter with Machine Learning

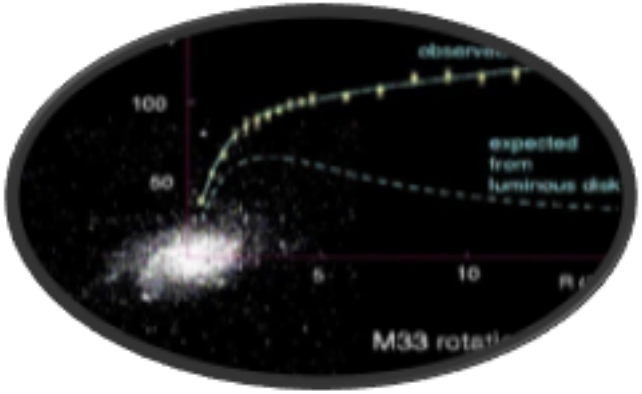
from 15 Jan 2018 through 19 Jan 2018



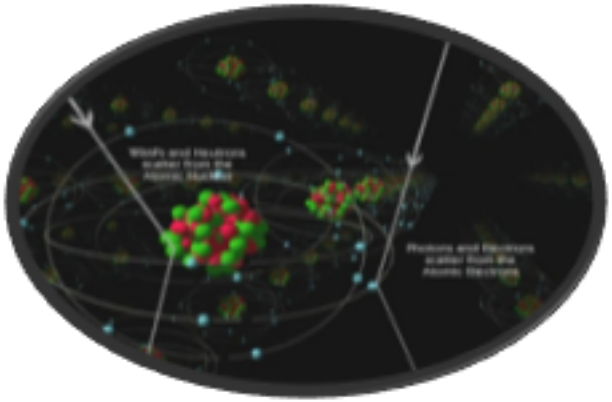
Our objective :

We aim to ***explore, and to encourage, the utilization of state-of-the-art machine learning algorithms for research in dark matter physics and astronomy***. Our objective is to accelerate the identification of Dark Matter with a ***multidisciplinary*** approach:
bringing together expertise in experimental and theoretical particle physics, astrophysics, astronomy, statistics and Machine Learning.
We like to generate ***a new open research community*.**
We plan a ***whitepaper, follow-up workshops, a webpage and a mailing list*** for the DM-ML community.
We plan to work on/organize **problems/challenges**

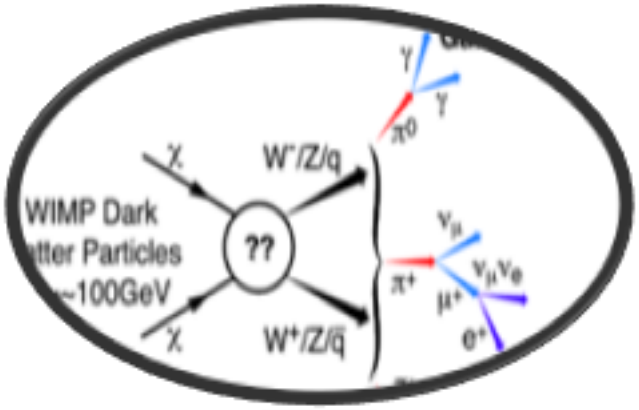
Dark Matter experiments



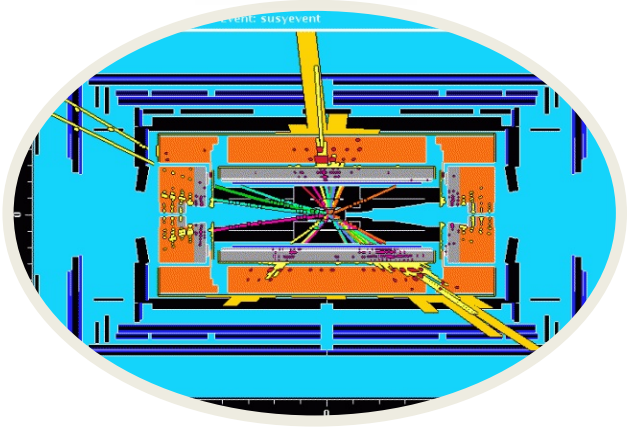
Gravitational interactions



Direct Detection



Indirect Detection



Production

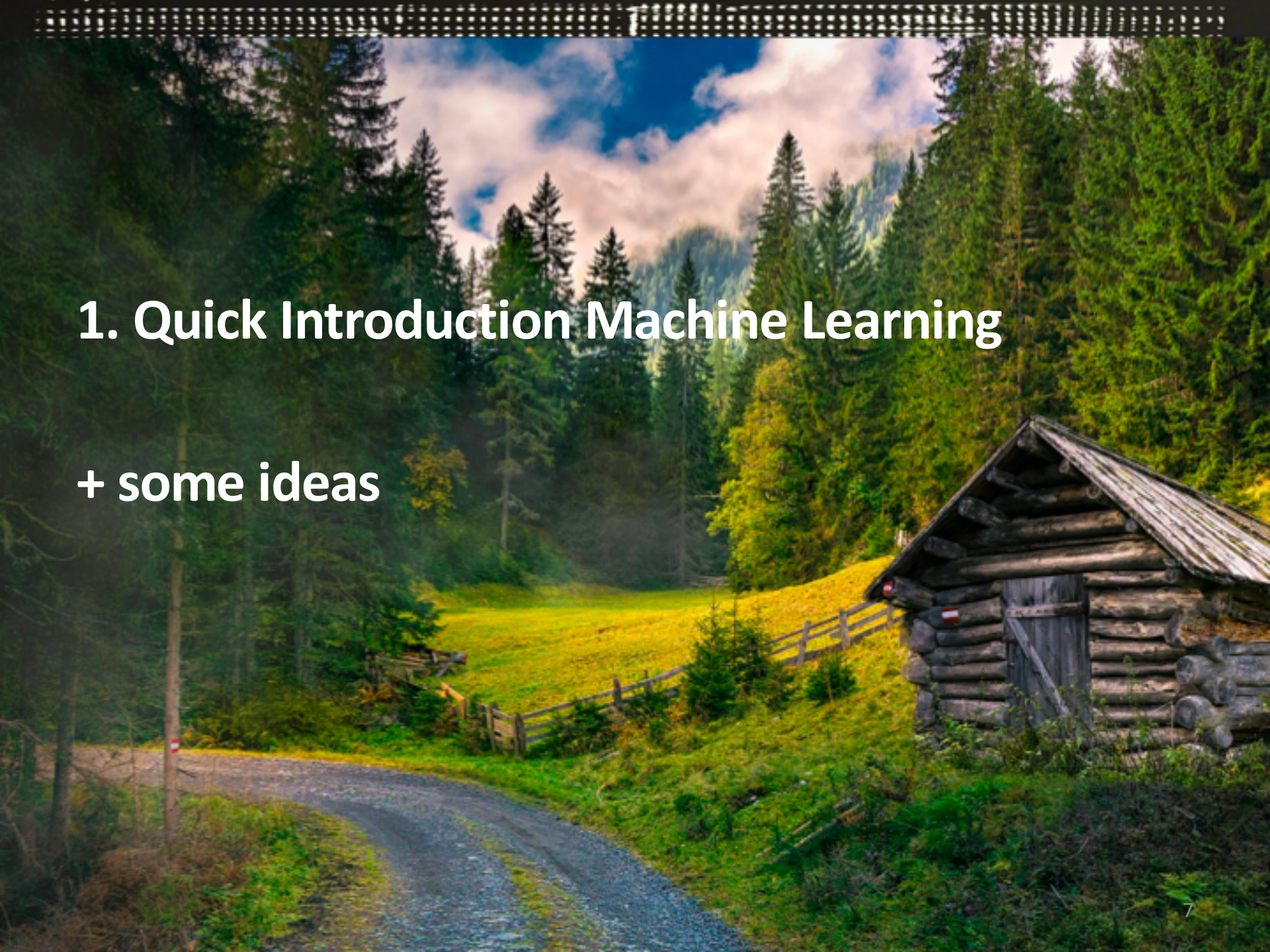
Data Science topics for today

Real world (physics + astrophysics/darkmatter) examples from

1. **Introduction ML**
2. **Computer vision (DM searches)**
3. **Physics models, simulators and cross sections (classification, regression, sampling, active learning)**
4. **Data gathering**
5. **(Unknown) data anomalies (BSM searches)**
6. **Generative models and latent space**

+ I will mention some ideas/new things I find interesting

+ Real world (physics + astrophysics/darkmatter) examples from darkmachines + recent papers


1. Quick Introduction Machine Learning
+ some ideas

What is Machine Learning ?

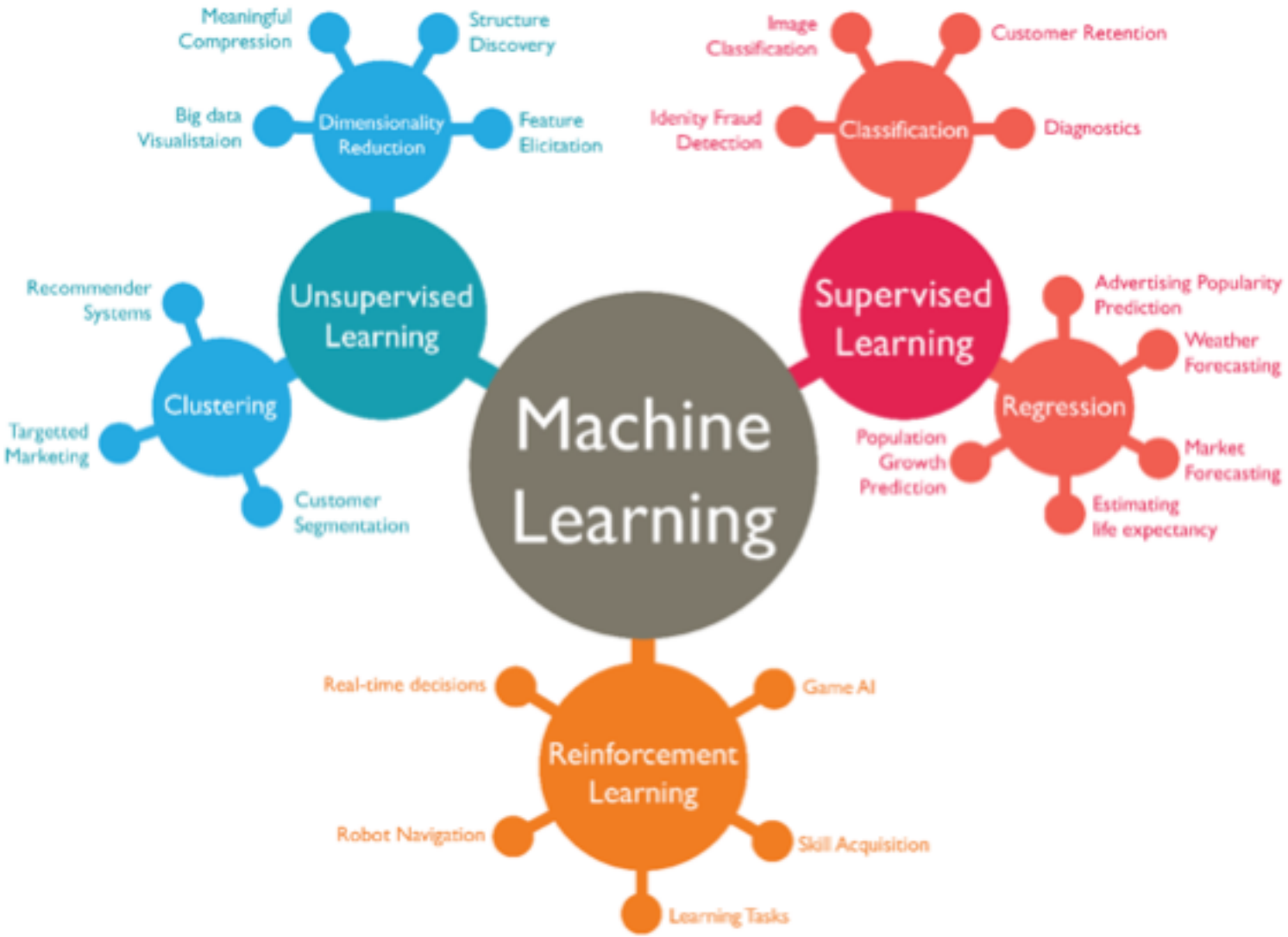


Image via Abdul Rahid

Most important example: Supervised Learning

Computer systems "learn" with data

Actually the computer "learn/derives and fits"
a continuous estimator $\hat{\mathbf{f}}(\vec{x})$ for an unknown function $\mathbf{f}(\vec{x})$
from i discrete data points $\vec{x_i}$ with known function values $\mathbf{f}(\vec{x_i})$.

i discrete data points $\vec{x_i}$ with known function values $\mathbf{f}(\vec{x_i})$.

is called the "**training set**".
Determining $\hat{\mathbf{f}}(\vec{x})$ is called "**training**".
The axis values of the $\vec{x_i}$ values are called "**features**".

Classification and Regression

- If $\mathbf{f}(\vec{x})$ outputs real values people call this **regression**
- If $\mathbf{f}(\vec{x})$ outputs integers (classes) people call this **classification**

1a) **regression**

$\mathbf{f}(\vec{x})$ is now a 1d function of a 1d variable x

Estimate function using polynominals

Trained function:

$$y(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + ...$$

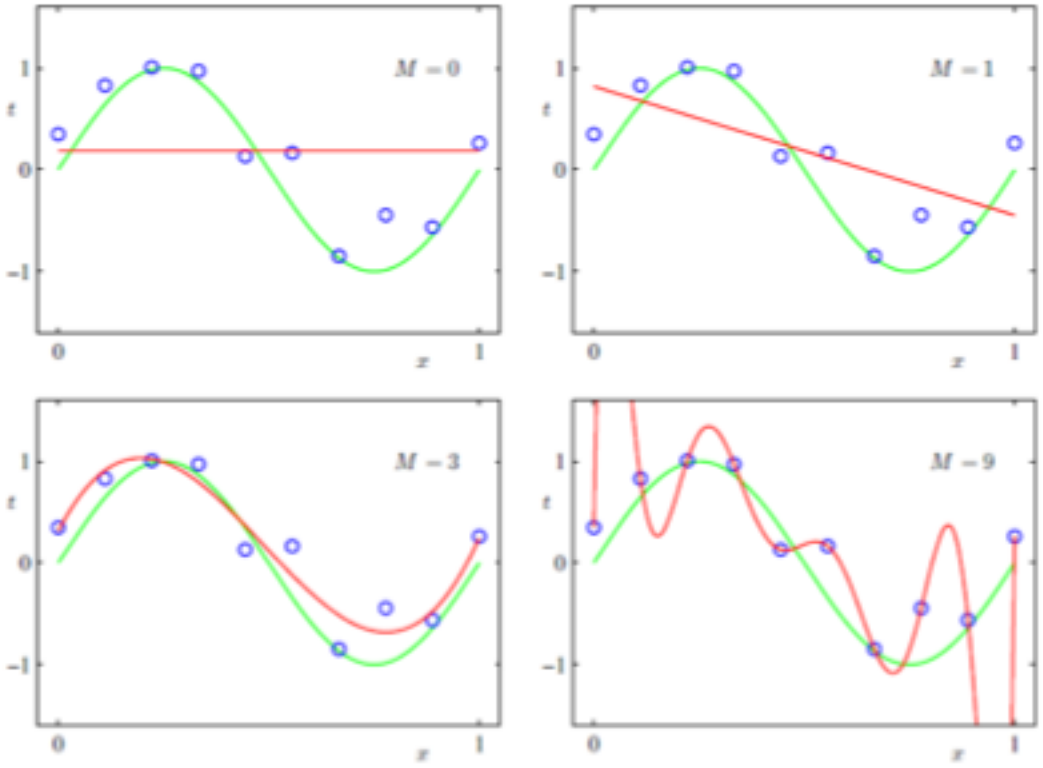
(linear model in the unknown parameter $\vec{w}$).

Problem is to determine "best" model parameters
This is done by defining an "error function" or loss function which is then "minimized".

$$E = \sum_{n=1}^{N} (y(x_n, \mathbf{w}) - f(x_n))^2$$

➔ Easy to solve

However: Which order of the polynomial ?



Example 1dim **regression**

$$E = \sum_{n=1}^{N} (y(x_n, \mathbf{w}) - f(x_n))^2$$

Trained function:

$$y(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + ...$$

(linear model in the unknown parameter $\vec{w}$).

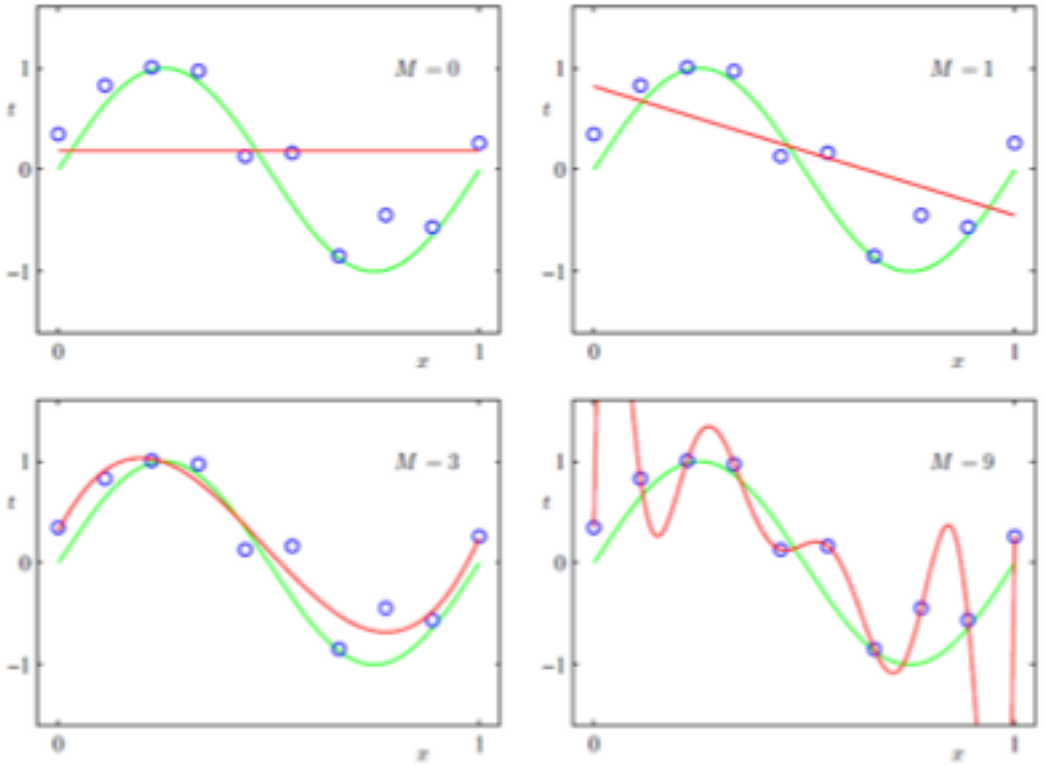
➔ Higher order polynominals Naturally fit better, but they do not agree with the true curve.

➔ Overfitting

Can be seen in data by calculating the error function of an independent data set

➔ **Test data error function would be large !**

➔ Dataset typically split in a "**training set**" and a "**testing set**"



Large datasets ←→ Complex models

Large datasets allow complex model

Training set to determine weights etc. of the model

→Various ways already to prevent “overfitting”

(simple: divide number degree of freedom)

Testing set to “check/validate” model

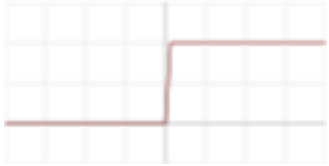
1b. Classification

Assume you would need to return labels in form of integers

Could be done via an “activation function” f:

$$y(x, \mathbf{w}) = f(\sum_i w_i x^i)$$

Simplest activation function could be a binary



In our example this would yield as output **classes**
(e.g. below or above 0, or is it a cat or a dog ?)

→ Easy to extend to multiple classes

Classification

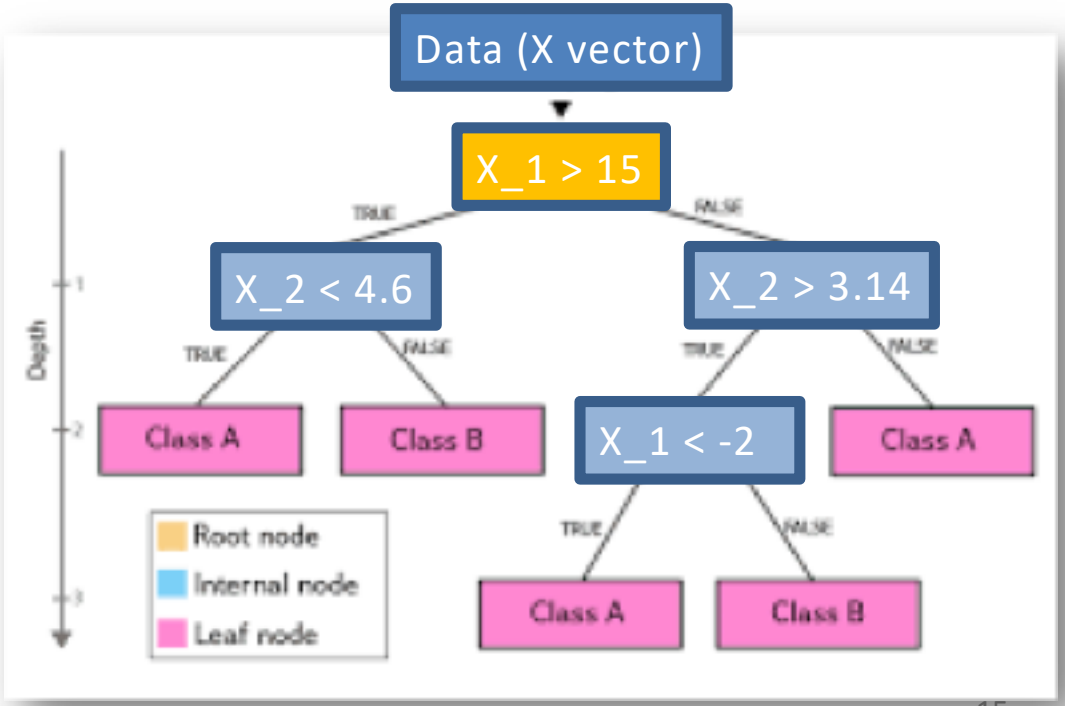
You might think that a “binning” might also work well for classification

-> and it does

-> Generalization of “binning” to multiple variables is a decision tree

Using various different “binnings”

is a forest of decision trees.



1c. Regression: Non-linear?

Various tool for regressions (fisher etc.) with linear functions in the parameters w

→But not clear what basis function to chose

→Need to adapt basis function to the data (SVM)

→Or best would be to “learn the basis function” as well…

Neural Network - regression

- Suppose we have a trivial model:

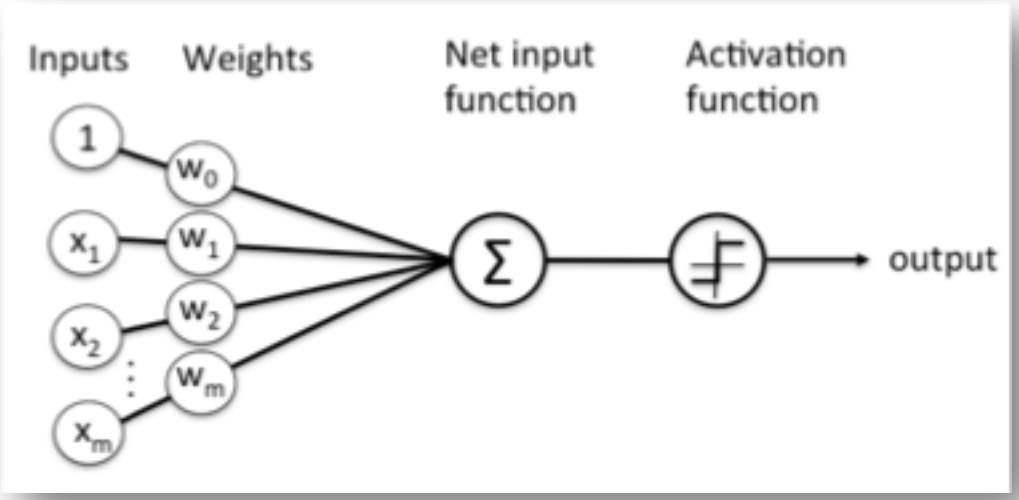
$$y(x, \mathbf{w}) = f(\sum_i w_i x)$$

and f is a non-linear activation function.

Let's make the input a vector (more input variables):

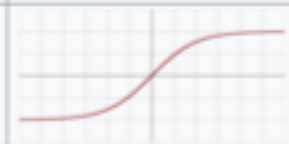
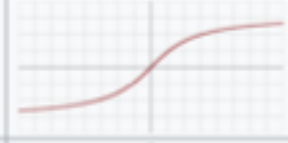
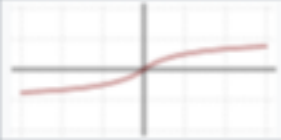
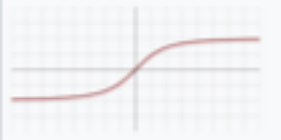
$$y(\vec{x}, \mathbf{w}) = f(\sum_i w_i x_i)$$

+ bias vector (here neglected, see NN lectures)



Activation functions examples

Does not work for
NN -> No non-linearities

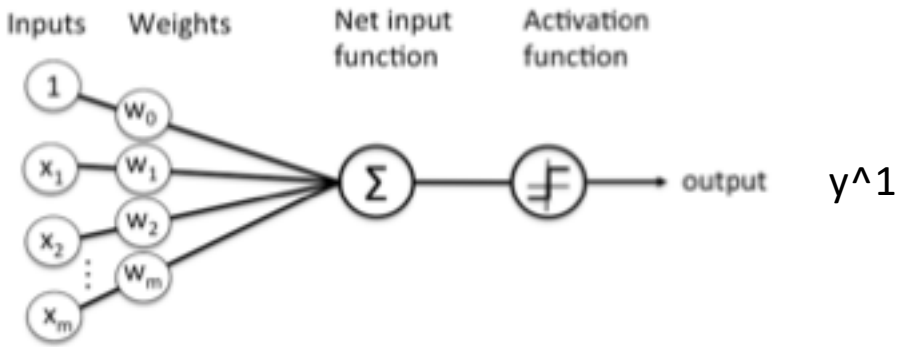
Identity		$f(x) = x$	$f'(x) = 1$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$
Logistic (a.k.a. Sigmoid or Soft step)		$f(x) = \sigma(x) = \frac{1}{1+e^{-x}}$ [1]	$f'(x) = f(x)(1-f(x))$
TanH		$f(x) = \tanh(x) = \frac{(e^x - e^{-x})}{(e^x + e^{-x})}$	$f'(x) = 1 - f(x)^2$
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2+1}$
Softsign [7][8]		$f(x) = \frac{x}{1+\lvert x\rvert}$	$f'(x) = \frac{1}{(1+\lvert x\rvert)^2}$
Inverse square root unit (ISRU)[9]		$f(x) = \frac{x}{\sqrt{1+\alpha x^2}}$	$f'(x) = \left(\frac{1}{\sqrt{1+\alpha x^2}}\right)^3$
Rectified linear unit (ReLU)[10]		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$

1d. Neural Networks

- Let us now make the basis function itself nonlinear combinations of its inputs

$$y(\vec{x}, \mathbf{w}) = f(\sum_i w_i x_i)$$

and f is a non-linear activation function



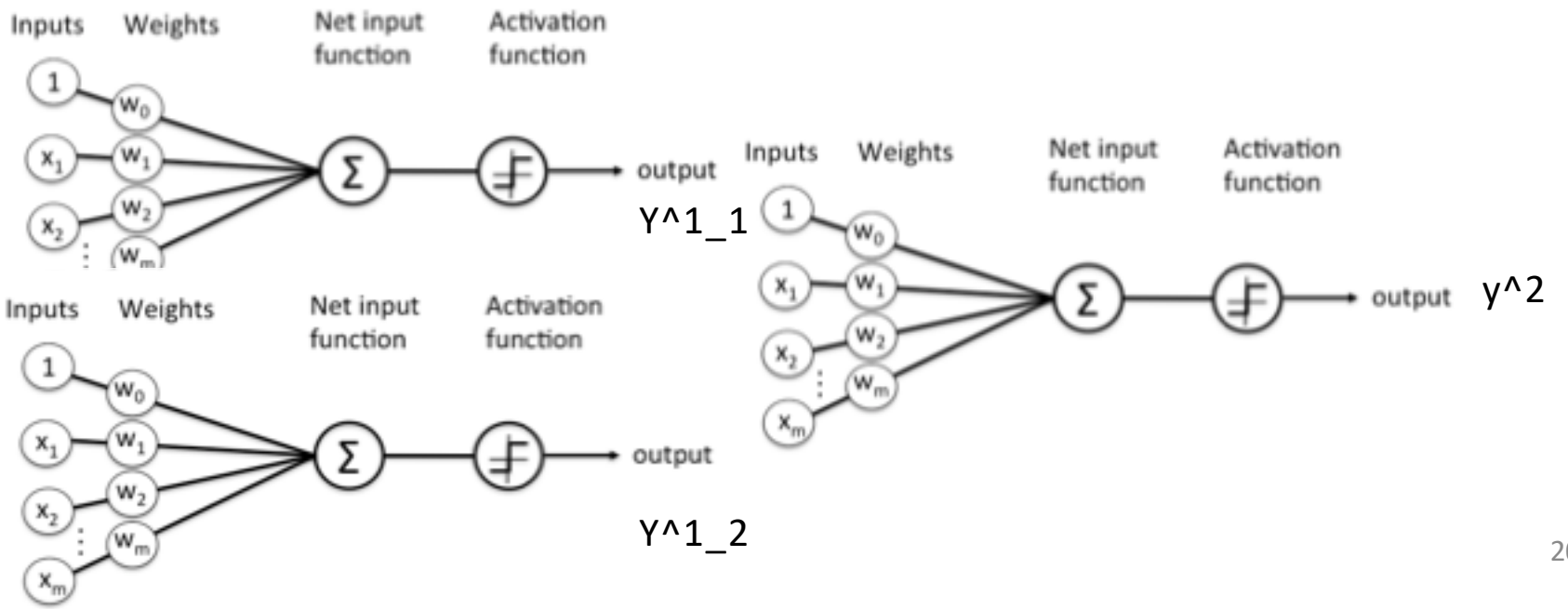
y^1

Neural Networks

- Let us now make the basis function itself nonlinear combinations of its inputs

$$y(\vec{x}, \mathbf{w}) = f(\sum_i f(\sum_j w_j x_j)$$

and f is a non-linear activation function



Neural Networks

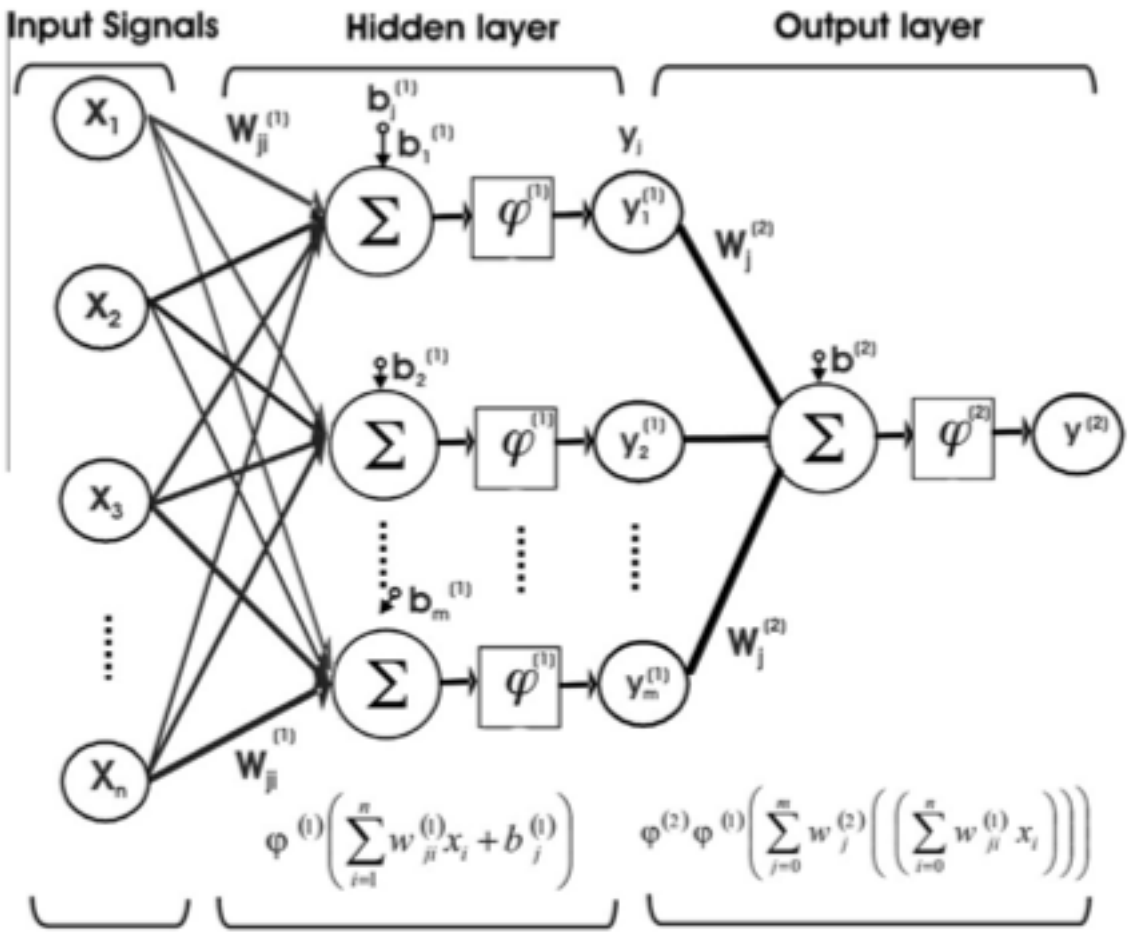
- Let us now make the basis function itself nonlinear combinations of its inputs

$$
\begin{aligned}
y^{(2)} &= \varphi^{(2)} \left(\sum_{j=1}^{m} \left(w_j^{(2)} y_j^{(1)} + b^{(2)} \right) \right) \\
&= \varphi^{(2)} \left(\sum_{j=1}^{m} w_j^{(2)} \varphi^{(1)} \left(\sum_{i=1}^{n} w_{ji}^{(1)} x_i + b_j^{(1)} \right) + b^{(2)} \right)
\end{aligned}
$$

And phi is a non-linear activation function, b is called bias

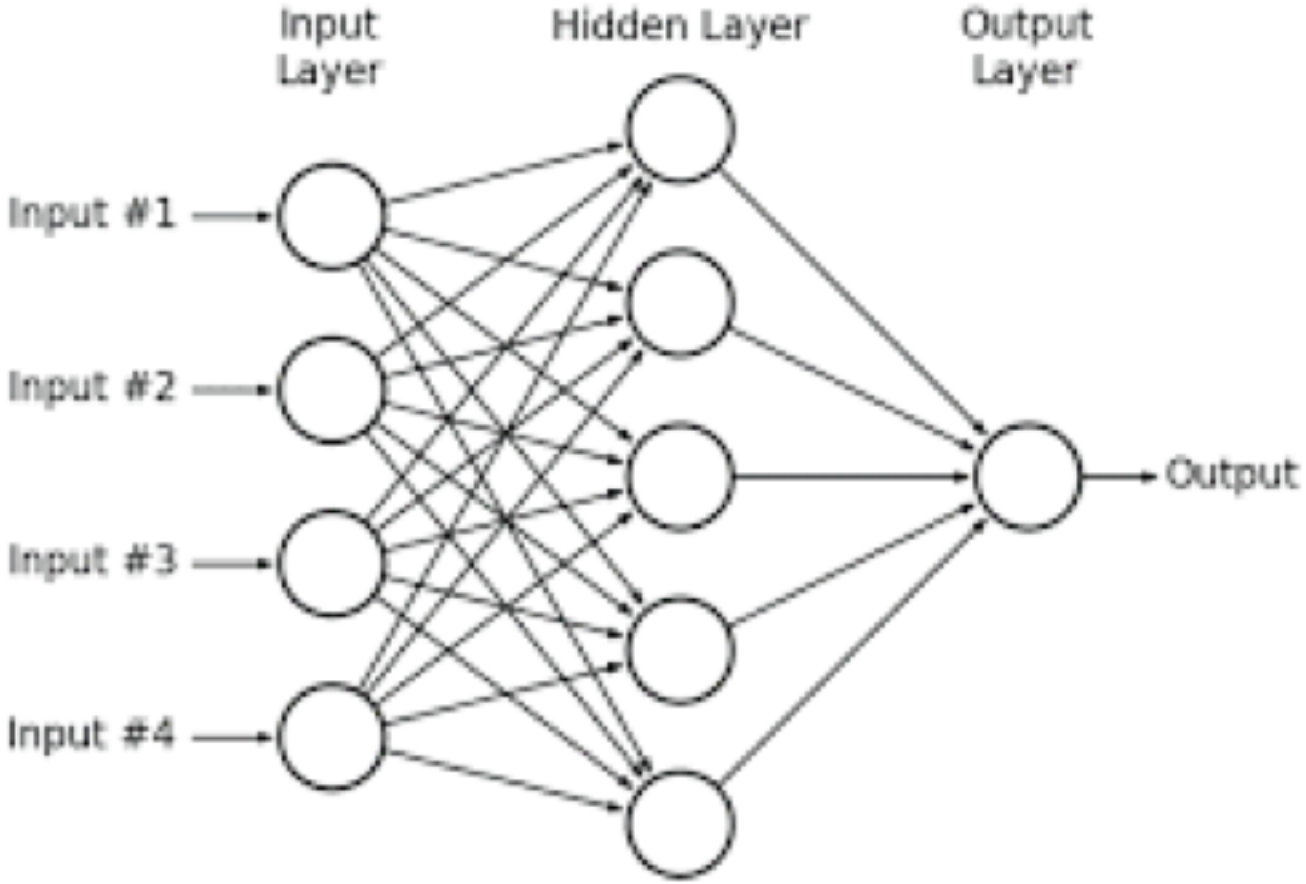
(bias allows to "shift" the activation function

$$
\begin{aligned}
y^{(2)} &= \varphi^{(2)} \left(\sum_{j=1}^{m} \left(w_j^{(2)} y_j^{(1)} + b^{(2)} \right) \right) \\
&= \varphi^{(2)} \left(\sum_{j=1}^{m} w_j^{(2)} \varphi^{(1)} \left(\sum_{i=1}^{n} w_{ji}^{(1)} x_i + b_j^{(1)} \right) + b^{(2)} \right)
\end{aligned}
$$



y

$$
\begin{aligned}
y^{(2)} &= \varphi^{(2)} \left(\sum_{j=1}^{m} \left(w_j^{(2)} y_j^{(1)} + b^{(2)} \right) \right) \\
&= \varphi^{(2)} \left(\sum_{j=1}^{m} w_j^{(2)} \varphi^{(1)} \left(\sum_{i=1}^{n} w_{ji}^{(1)} x_i + b_j^{(1)} \right) + b^{(2)} \right)
\end{aligned}
$$



This is a 3 layer (1 hidden layer) feedforward (multilayer) perceptron

→This is the “simplest network”

“Training”:

Finding the set of weights which minimize the error function

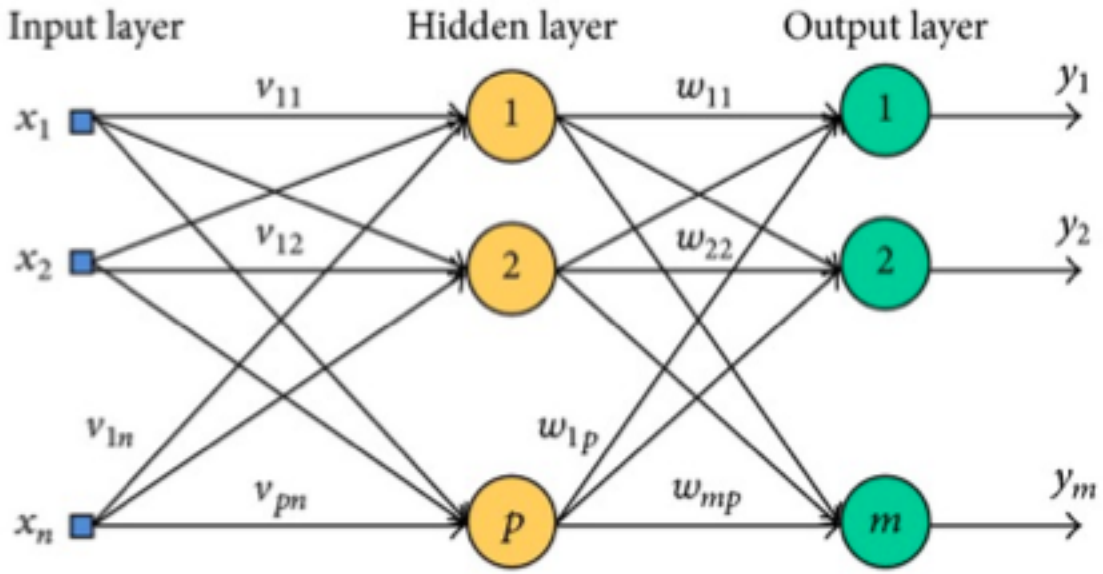
Example: NNpdfs

Fitting pdfs without assuming the underlying function

10. **Parton distributions for the LHC Run II**
NNPDF Collaboration (Richard D. Ball (U. Edinburgh, Higgs Ctr. Theor. Phys. & CERN) *et al.*). Oct 31, 2014. 138 pp.
Published in **JHEP 1504 (2015) 040**
EDINBURGH-2014-15, IFUM-1034-FT, CERN-PH-TH-2013-253, OUTP-14-11P, CAVENDISH-HEP-14-11
DOI: 10.1007/JHEP04(2015)040
e-Print: **arXiv:1410.8849 [hep-ph]** | PDF
References | BibTeX | LaTeX(US) | LaTeX(EU) | Harvmac | EndNote
CERN Document Server; ADS Abstract Service; Link to Article from SCOAP3
Detailed record - Cited by 1054 records 1000+

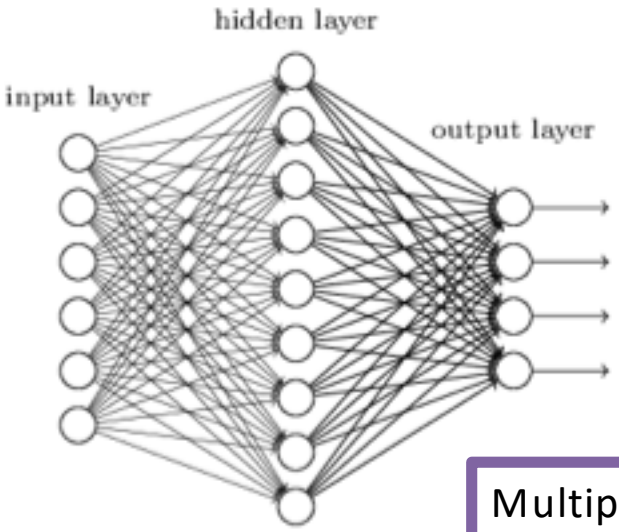
Neural Networks

- Of course we can have multiple output nodes



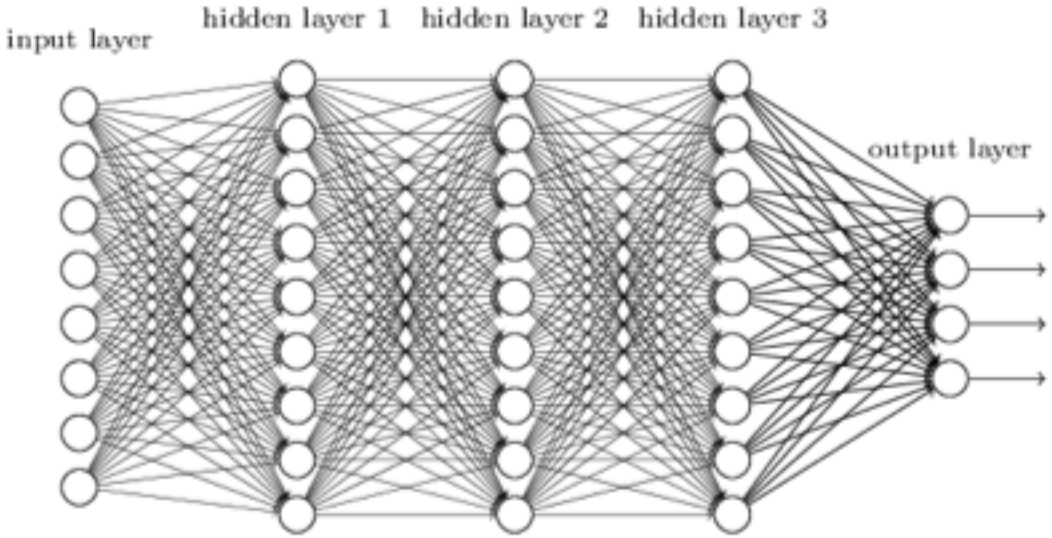
… or multiple hidden layers…

"Non-deep" feedforward neural network



Multiple Output Nodes !

Deep neural network



We often use 5-10 layers

2014 First deep network in HEP (begin 2018 we had 50 on arxiv)

Searching for Exotic Particles in High-Energy Physics with Deep Learning

P. Baldi,[1] P. Sadowski,[1] and D. Whiteson[2]

[1]*Dept. of Computer Science, UC Irvine, Irvine, CA 92617*
[2]*Dept. of Physics and Astronomy, UC Irvine, Irvine, CA 92617*

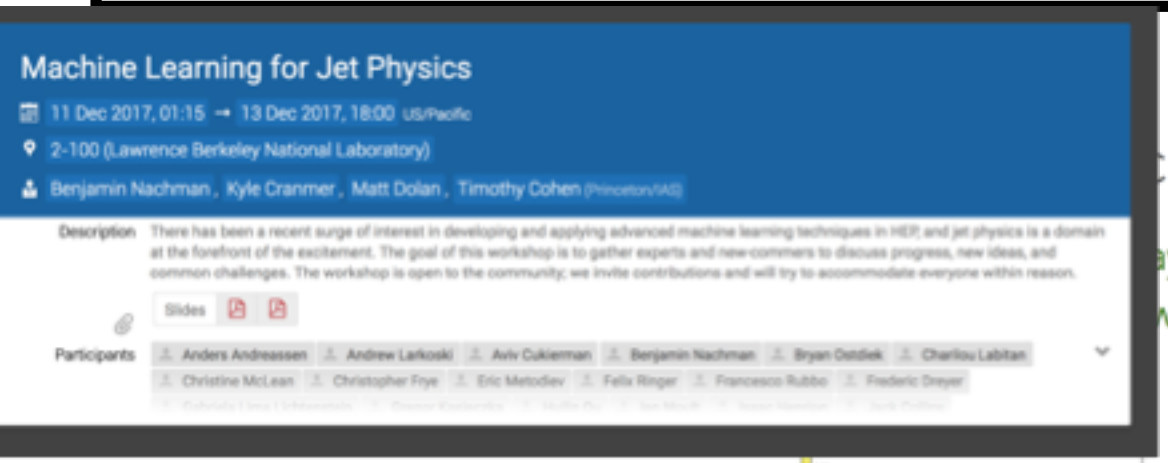
Collisions at high-energy particle colliders are a traditionally fruitful source of exotic particle discoveries. Finding these rare particles requires solving difficult signal-versus-background classification problems, hence machine learning approaches are often used. Standard approaches have relied on 'shallow' machine learning models that have a limited capacity to learn complex non-linear functions of the inputs, and rely on a pain-staking search through manually constructed non-linear features. Progress on this problem has slowed, as a variety of techniques have shown equivalent performance. Recent advances in the field of deep learning make it possible to learn more complex functions and better discriminate between signal and background classes. Using benchmark datasets, we show that deep learning methods need no manually constructed inputs and yet improve the classification metric by as much as 8% over the best current approaches. This demonstrates that deep learning approaches can improve the power of collider searches for exotic particles.

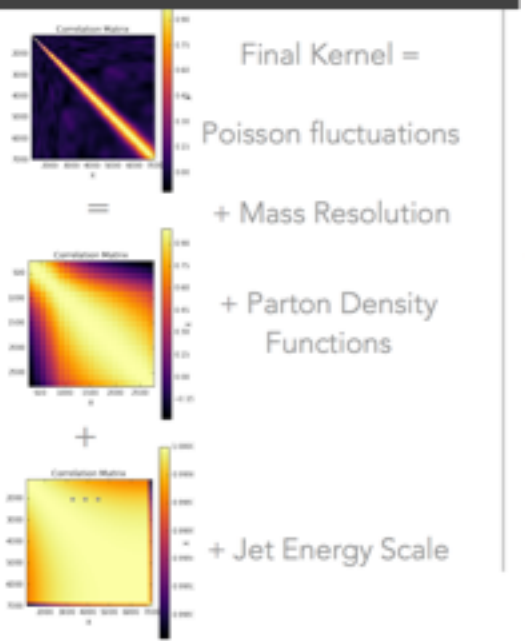
	AUC		
Technique	Low-level	High-level	Complete
BDT	0.73 (0.01)	0.78 (0.01)	0.81 (0.01)
NN	0.733 (0.007)	0.777 (0.001)	0.816 (0.004)
DN	0.880 (0.001)	0.800 (< 0.001)	0.885 (0.002)

	Discovery significance		
Technique	Low-level	High-level	Complete
NN	2.5σ	3.1σ	3.7σ
DN	4.9σ	3.6σ	5.0σ

Important:
Input only 4 vectors !!!!
No knowledge about physics !!!!

Then ..hyped in QCD … jet algorithms.. … top taggers.. Showering and calorimeters





CHINE LEARNING

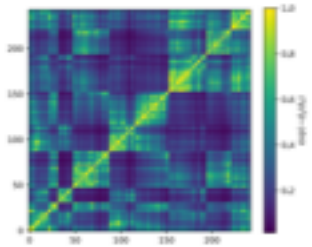
ays of injecting physics knowledge into well…

QCD-Aware recursive neural networks

arXiv:1702.00748

QCD-Aware graph convolutional neural networks

NIPS2017 workshop [http://bit.ly/2AkwYRG]



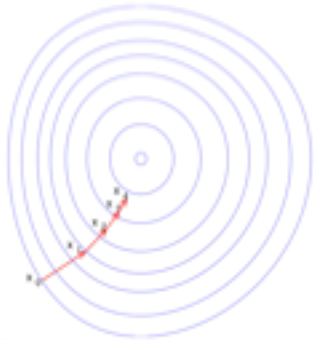
$$d_{ii'}^{\alpha} = \min(p_{ti}^{2\alpha}, p_{ti'}^{2\alpha}) \frac{\Delta R_{ii'}^2}{R^2}$$

Discussed in
talk by Michael Kagan

Why can we now go deep ?
→ Need to minimize the error/loss functions for many weights

How ? "Gradient descent"

(going into the direction of the negative gradient to quickly find the minimum)



- **Computing power increase**

→ Matrix multiplications" ➔ GPUs

- **Smart algorithm: "Back-propagation"**

(derive error per dataset, then change w_i according to their contribution to error going backwards from output)

- **Large amount of data**

→ helps to constrain more w

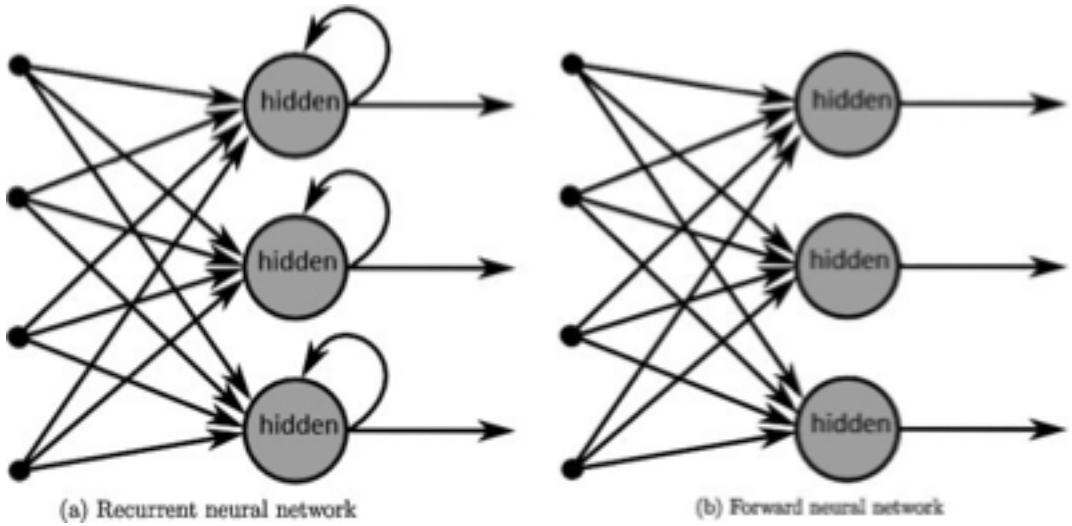
1e. Feed forward or recurrent ?

RNN take as their input not just the current input, but also what they have perceived previously

→ "Notion of time / ordering"

Useful to learn "time dependent functions/relations "

(*particle decays ?)*



(a) Recurrent neural network (b) Forward neural network

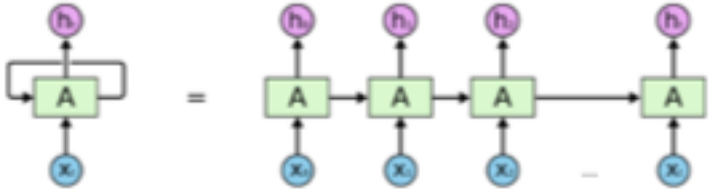
RNN node will be a function with inputs x(t) (input vector) and a memory (hidden) of previous state m(t−1) . The new output state will be y(t)

→ Can hold a bit of memory of previous state.

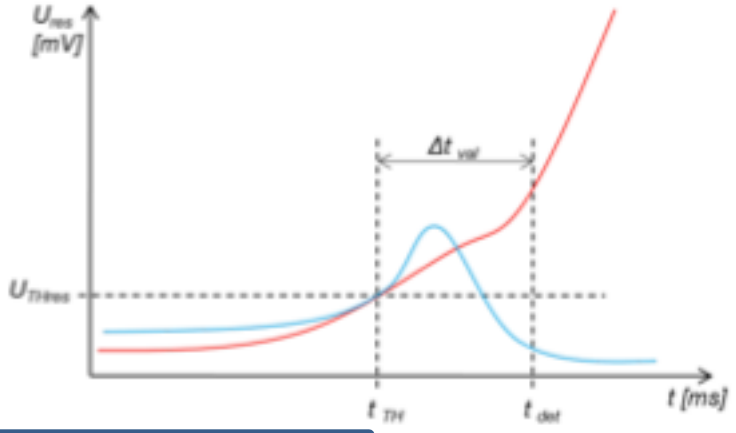
RNN Example: Monitoring and fault protection of the Large Hadron Collider (LHC) superconducting magnets

https://arxiv.org/abs/1611.06241 Maciej Wielgosz, Andrzej Skoczeń, Matej Mertik

Idea: *RNN can model the voltage time series of LHC superconducting magnets → including severe phenomena such as quenches.*



An unrolled recurrent neural network.



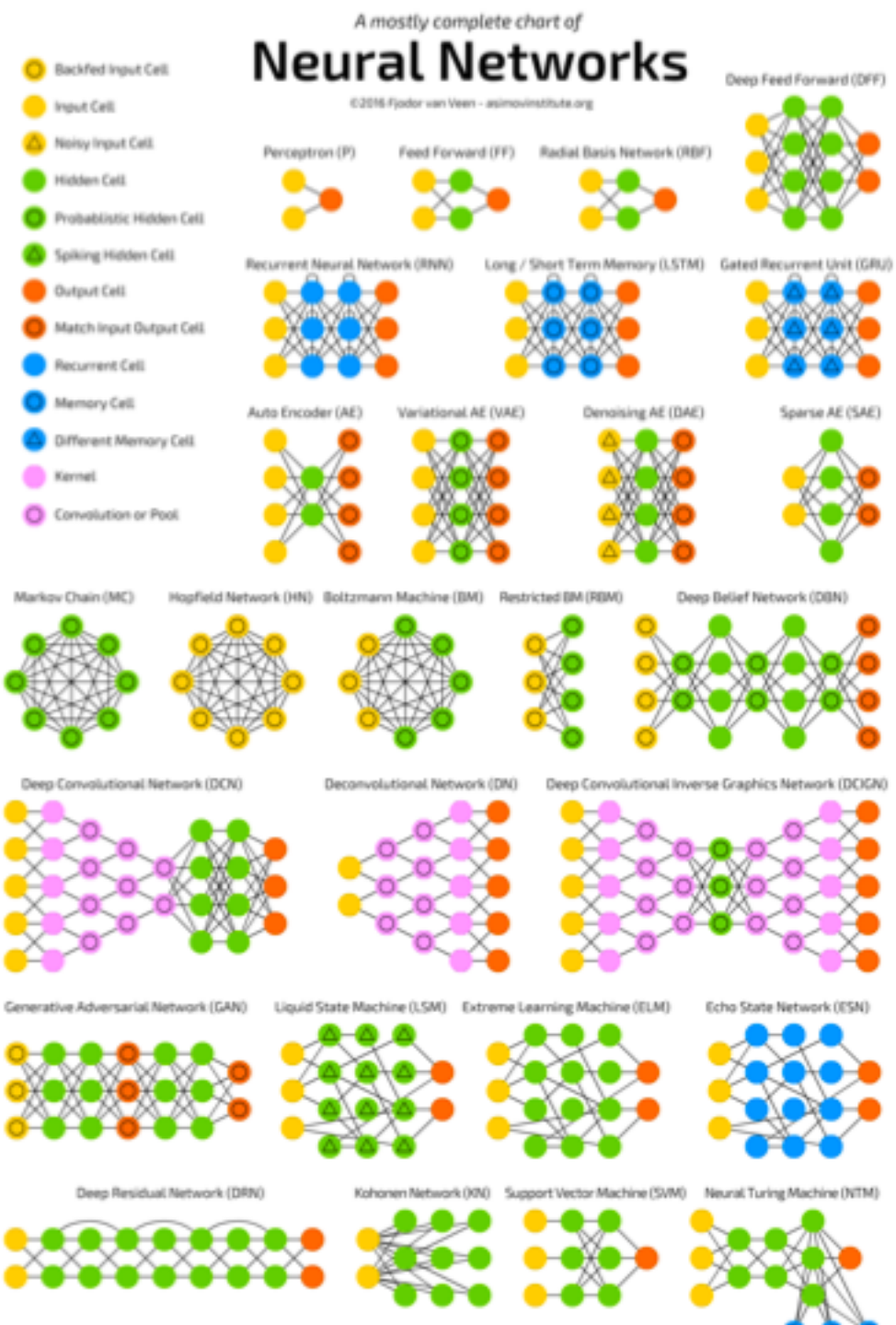
If quench causes it may **gradually unfold in time**

→ There might be the ability to predict likely quench…

Voltage time series → Probability for quench

Could be solved with RNN

(actually a Long Short Term Memory , LSTM, has cell to hold long term memory))


A mostly complete chart of
Neural Networks
©2016 Fjodor van Veen - asimovinstitute.org
Backfed Input Cell
Input Cell
Noisy Input Cell
Hidden Cell
Probabilistic Hidden Cell
Spiking Hidden Cell
Output Cell
Match Input Output Cell
Recurrent Cell
Memory Cell
Different Memory Cell
Kernel
Convolution or Pool
Perceptron (P)
Feed Forward (FF)
Radial Basis Network (RBF)
Deep Feed Forward (DFF)
Recurrent Neural Network (RNN)
Long / Short Term Memory (LSTM)
Gated Recurrent Unit (GRU)
Auto Encoder (AE)
Variational AE (VAE)
Denoising AE (DAE)
Sparse AE (SAE)
Markov Chain (MC)
Hopfield Network (HN)
Boltzmann Machine (BM)
Restricted BM (RBM)
Deep Belief Network (DBN)
Deep Convolutional Network (DCN)
Deconvolutional Network (DN)
Deep Convolutional Inverse Graphics Network (DCIGN)
Generative Adversarial Network (GAN)
Liquid State Machine (LSM)
Extreme Learning Machine (ELM)
Echo State Network (ESN)
Deep Residual Network (DRN)
Kohonen Network (KN)
Support Vector Machine (SVM)
Neural Turing Machine (NTM)

A mostly complete chart of

Neural Networks

©2016 Fjodor van Veen - asimovinstitute.org

- Backfed Input Cell
- Input Cell
- Noisy Input Cell
- Hidden Cell
- Probablistic Hidden Cell
- Spiking Hidden Cell
- Output Cell
- Match Input Output Cell
- Recurrent Cell
- Memory Cell
- Different Memory Cell
- Kernel
- Convolution or Pool

Perceptron (P)

Feed Forward (FF)

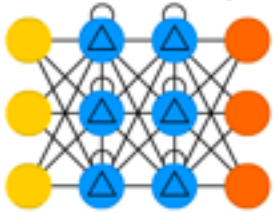
Radial Basis Network (RBF)

Deep Feed Forward (DFF)

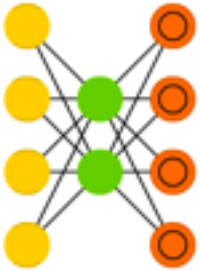
Recurrent Neural Network (RNN)

Long / Short Term Memory (LSTM)

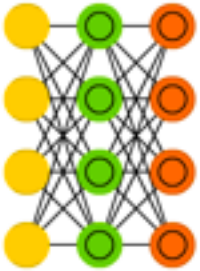
Gated Recurrent Unit (GRU)



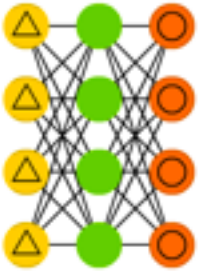
Auto Encoder (AE)



Variational AE (VAE)



Denoising AE (DAE)



Sparse AE (SAE)

Markov Chain (MC)

Hopfield Network (HN)

Boltzmann Machine (BM)

Restricted BM (RBM)

Deep Belief Network (DBN)

1f. Training methods

- **Supervised learning**

→In the learning process we give the sample labels (i.e. we know/ give true f_i(x) for some x_i)

→Most common approach

- **Unsupervised learning**

→We do not know the labels

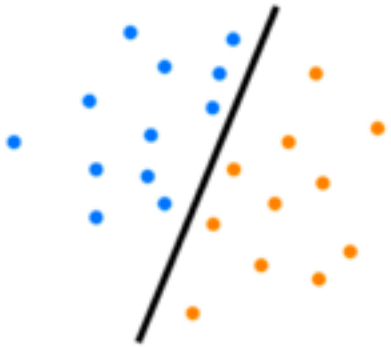
- **Reinforcement learning**

→We train on a “long term reward function”

Training methods

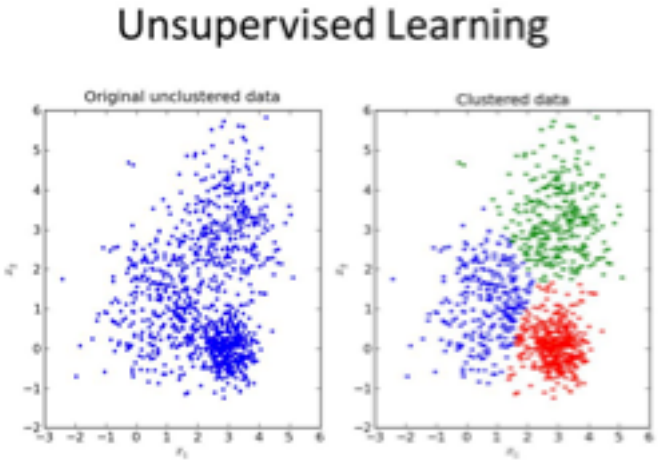
Supervised

Learning known patterns



Unsupervised

Learning unknown patterns



Reinforcement

Generating data

Learning patterns

… there is also e.g. semi-supervised using labeled + unlabeled data

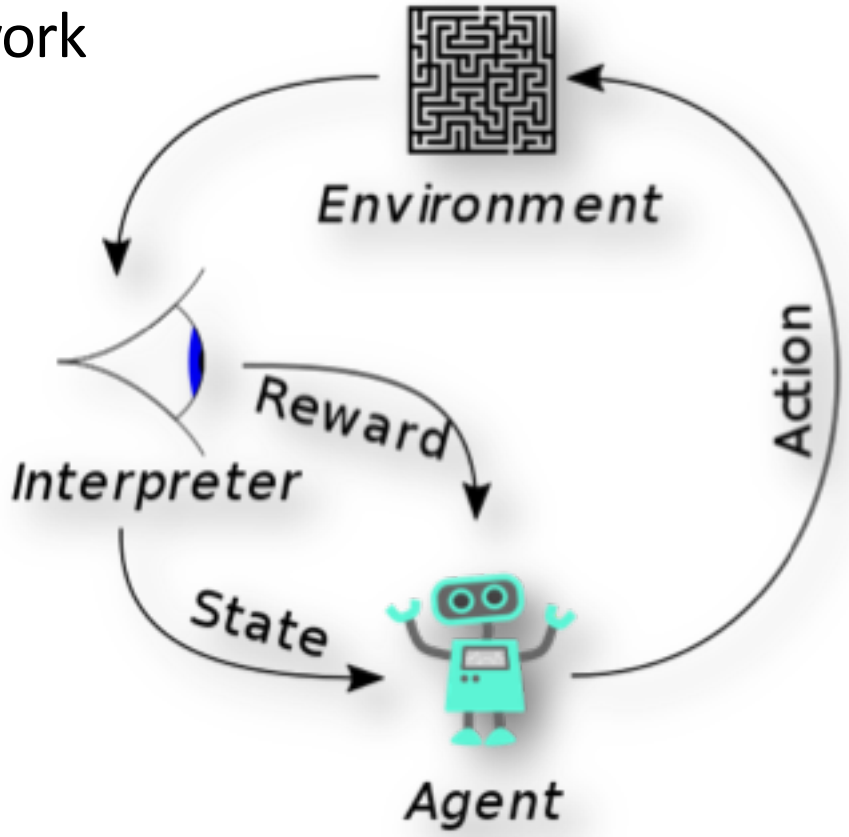
Reinforcement learning

Agent could be a recurrent neural network

It trains (and restructures) itself by **playing** with an environment (maze)

We need to define a reward function (e.g. time it takes to solve the maze)

Training is then maximizing/minizine the reward function



Solving games with reinforcement learning

AlphaGo or here AlphaZero:
Starting from random play, and given no domain knowledge except the game rules, AlphaZero achieved **within 24 hours a superhuman level** of play in the games of chess and shogi (Japanese chess) as well as Go, and convincingly defeated a world-champion program in each case.

https://arxiv.org/pdf/1712.01815.pdf

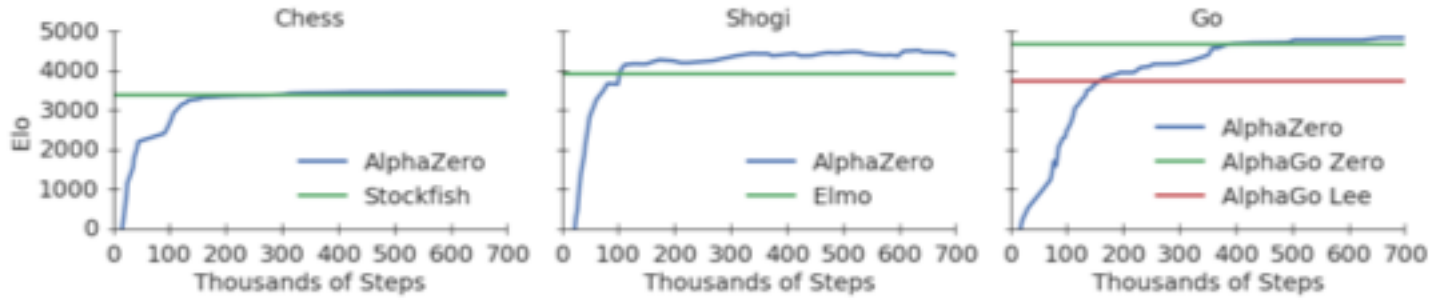


Figure 1: Training *AlphaZero* for 700,000 steps. Elo ratings were computed from evaluation games between different players when given one second per move. **a** Performance of *AlphaZero* in chess, compared to 2016 TCEC world-champion program *Stockfish*. **b** Performance of *AlphaZero* in shogi, compared to 2017 CSA world-champion program *Elmo*. **c** Performance of *AlphaZero* in Go, compared to *AlphaGo Lee* and *AlphaGo Zero* (20 block / 3 day) (*29*).

reinforcement learning

This works very well in games.

⇒Agent can act in the system very fast (game)

⇒Can use this in physics if the problem can be formulated / simulated as a game

Need Physics simulator/ World simulator model

➔Goal is then to build physics simulator…

(see simulator network)

Your Ideas for physics ?

Deep Convolutional Networks

Actually Alpha-go used a deep convolutional network… What is this ?

We recently used deep convolutional networks to analyse gamma ray images for

Dark Matter

https://arxiv.org/abs/1708.06706

2. Computer vision for astroparticle physics

Convolutional Networks

- Convolutional networks have convolution layers based on “filters”, a filter (a **matrix**) maps “a group of numbers” to “a number” reducing the data → CONV layers
- There are also layers which only do a downsampling (lower the dimensionality) POOL or “fully connected layers” to process the final numbers…

important paper:LeCun, Yann. "LeNet-5, convolutional neural networks". Retrieved 16 November 2013.

Filters (Matrix)

- Unity

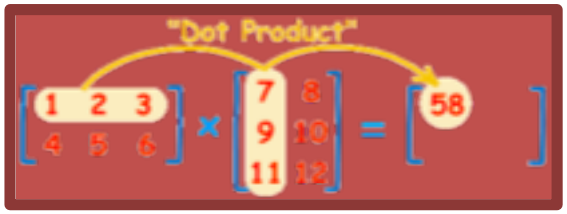
$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

"Edge detector":

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

"Convolutional" Network use "invariances" (rotation, translation) in data (e.g. images)

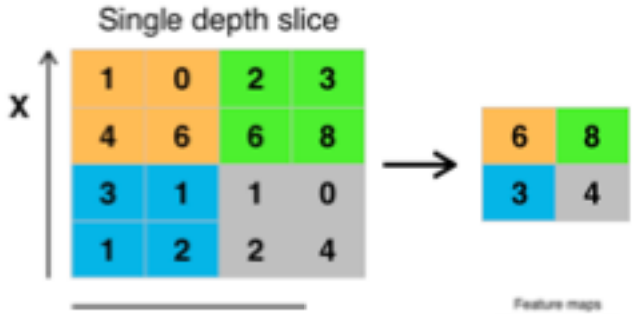
CONV layer

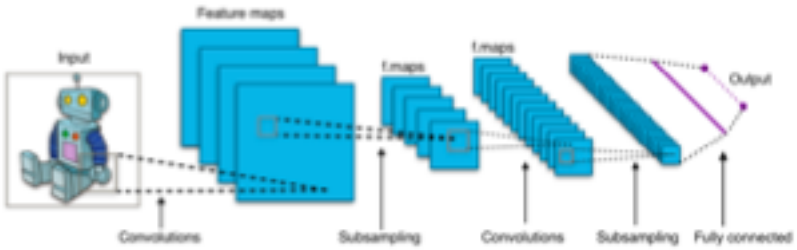


+ adding bias vector + applying e.g. RELU as non-linearity

Filter coefficients learned by backpropagation

Followed by
a POOLING layer
(partition the input matrix into Submatrices, here max POOL, i.e. Storing the maximum)





(you can have Red, Green and Blue matrices)

Use case: gamma rays from galactic center

A method to investigate the origin of an excess emission of GeV γ rays in the direction of the Galactic Center
(reported by several groups by analyzing Fermi-LAT data)

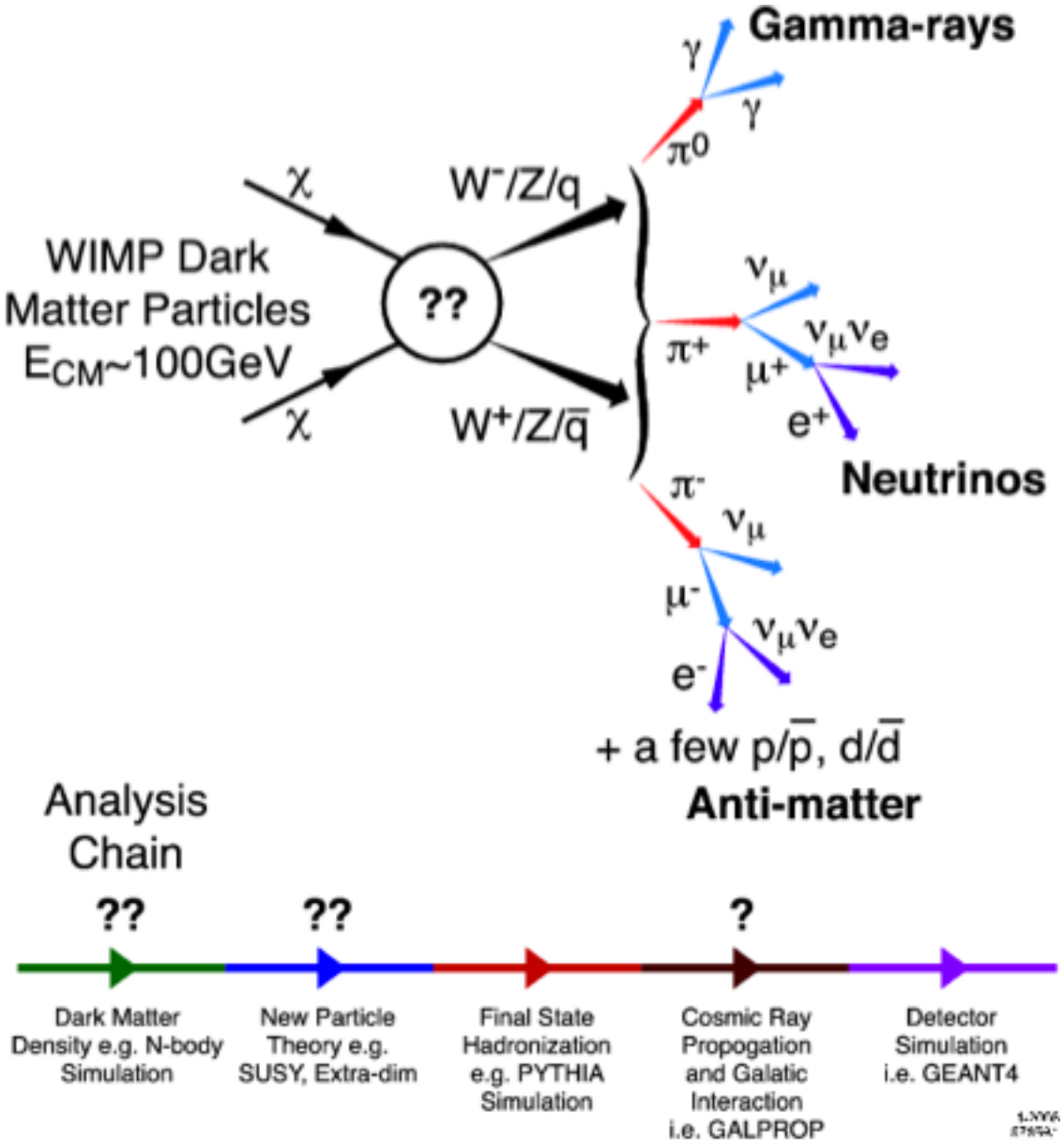
Interpretations of this excess include γ rays created **by the annihilation of dark matter particles** and γ rays originating from a **collection of unresolved point sources, such as millisecond pulsars.**

What have we done ?

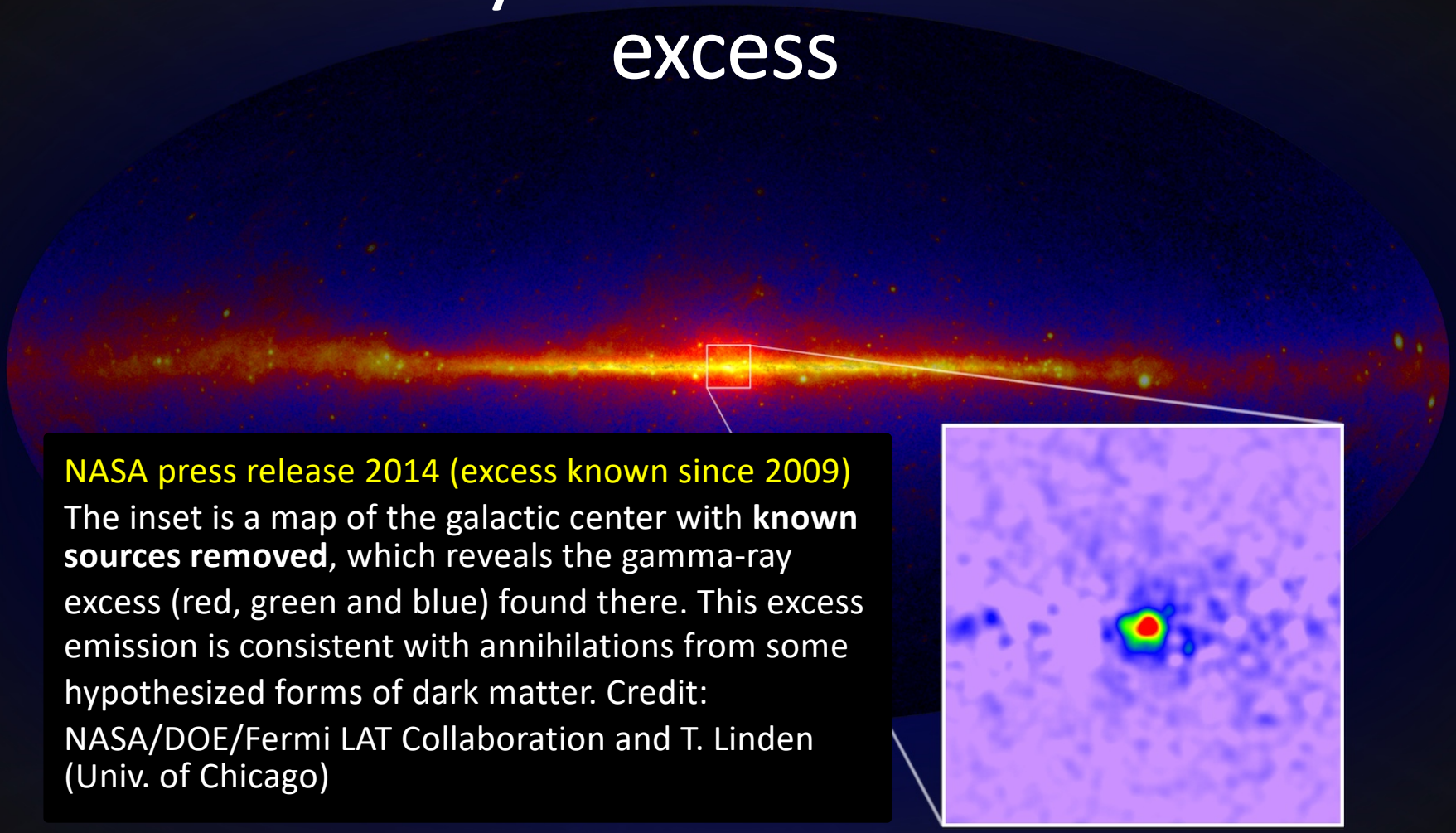
Simulated Fermi-LAT images based on point and diffuse emission models of the Galactic Center tuned to measured γ ray data

➔ Train and test convolutional Network on this

WIMP Astrophysics



Gamma rays & the Galactic Center excess



NASA press release 2014 (excess known since 2009)

The inset is a map of the galactic center with **known sources removed**, which reveals the gamma-ray excess (red, green and blue) found there. This excess emission is consistent with annihilations from some hypothesized forms of dark matter. Credit: NASA/DOE/Fermi LAT Collaboration and T. Linden (Univ. of Chicago)

Official paper in 2015

Fermi-LAT Observations of High-Energy Gamma-Ray Emission Toward the Galactic Center

Fermi-LAT Collaboration (M. Ajello (Clemson U.) *et al.*). Nov 9, 2015. 29 pp.

e-Print: arXiv:1511.02938 [astro-ph.HE] | PDF

Our convolutional network (convnet)

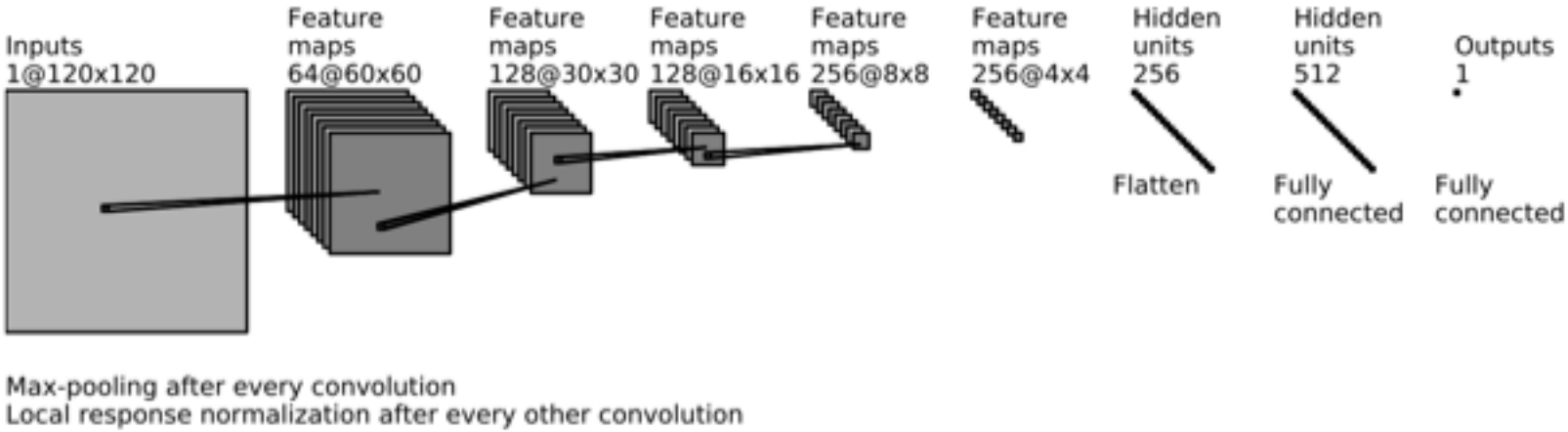


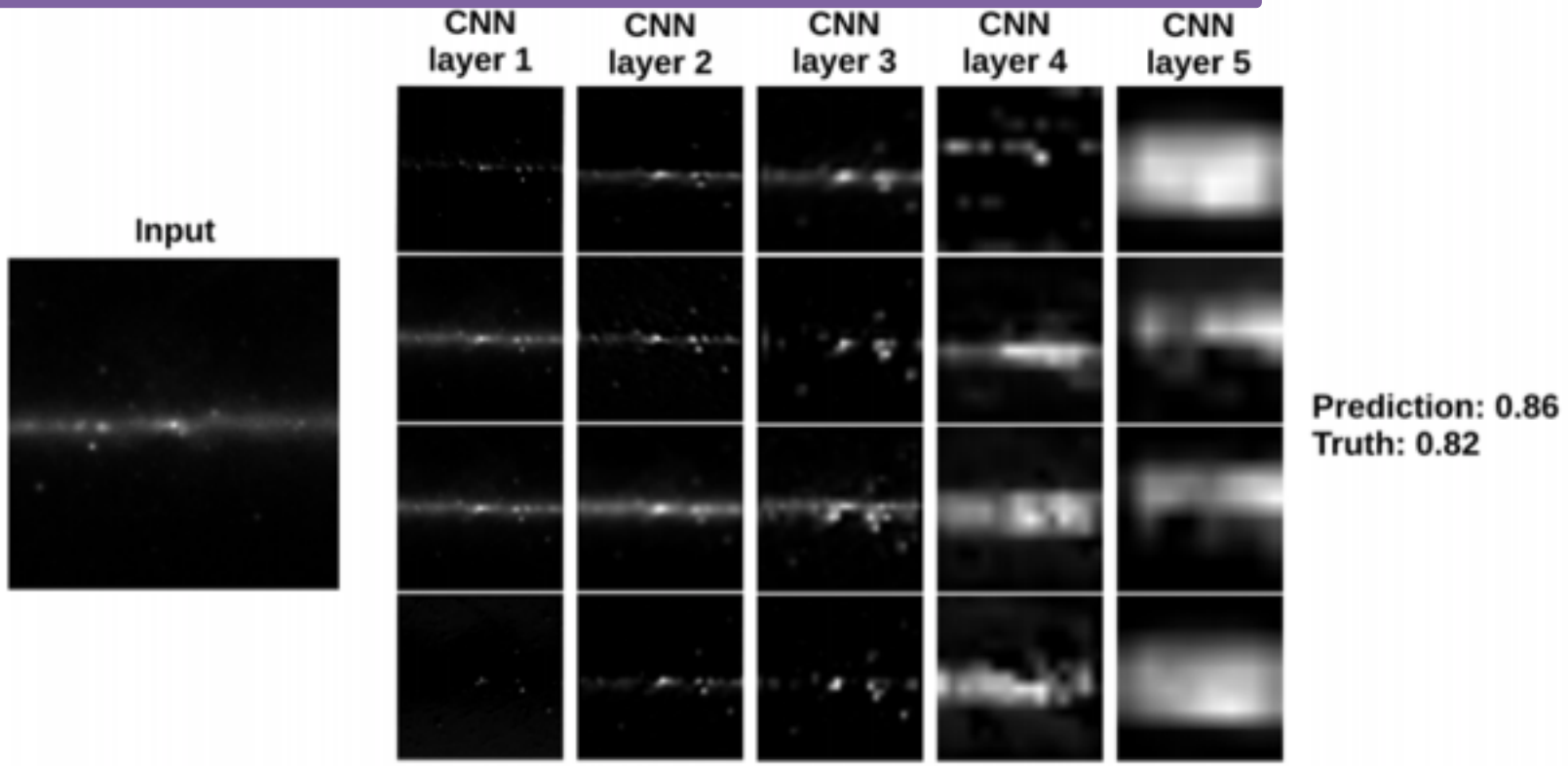
Figure 6: Visualization of the convolutional neural network. The network consists of an input layer, 5 convolutional + pooling layers, 2 fully connected layers and finally an output layer.

Hierarchical/compositional structure → smaller to larger structures
(reason: visible system is hierarchical as well…)

In comparison: GoogleLeNet has like 30 layers…

Isotropic or point sources: A Deep Convolutional Network approach

Output of the 5 convolutional layers can be “visualized” per event.

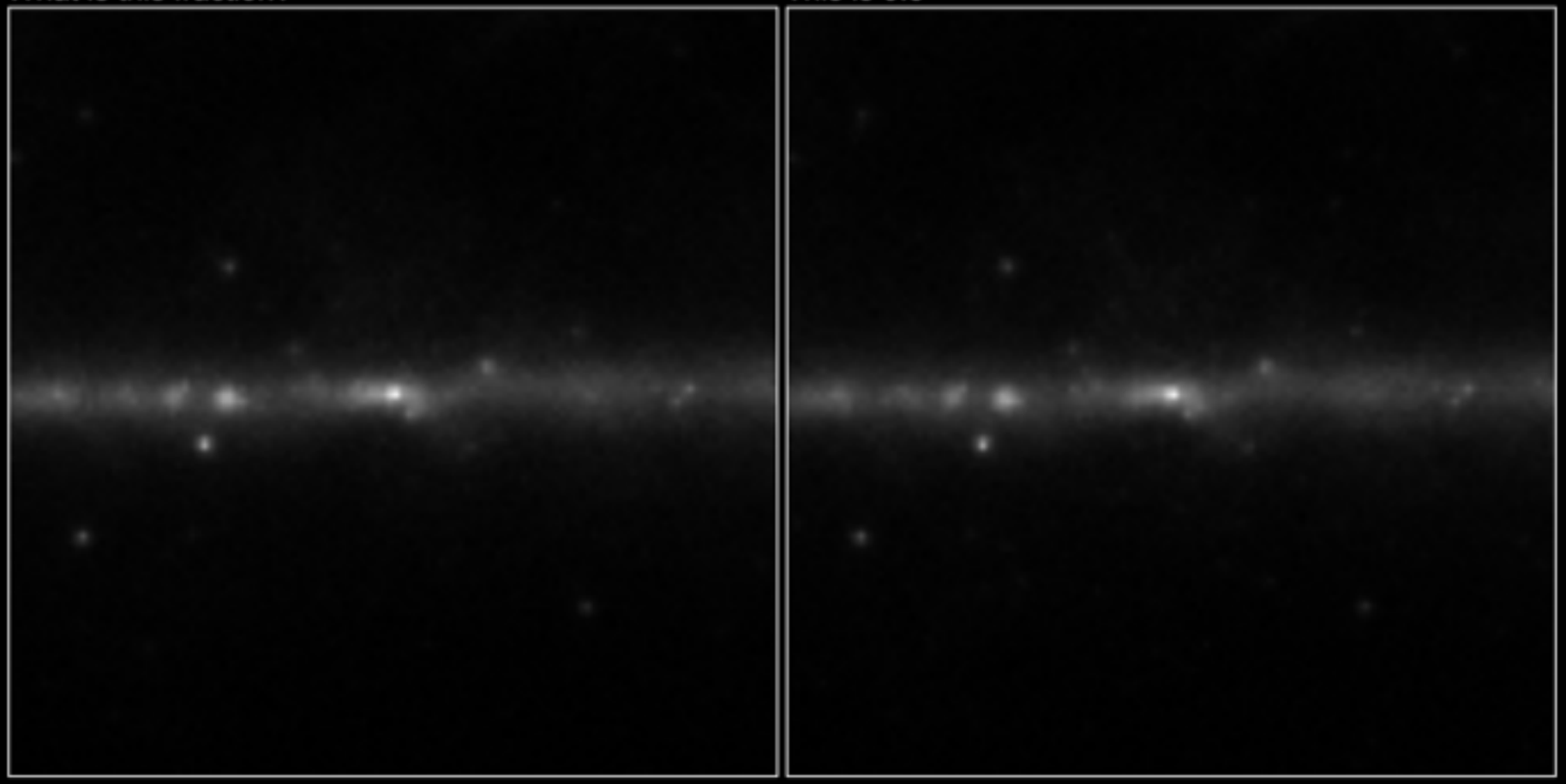


Activations of the network. Only four filters per layers are shown for clarity, between 256 and 65 filters are used for the different layers

Luc Hendriks, S. Caron et al. Fermi-LAT cat. 2 paper arXiv:1708.06706

Guess the fraction of point sources

www.mydarkmachine.org



Your prediction:

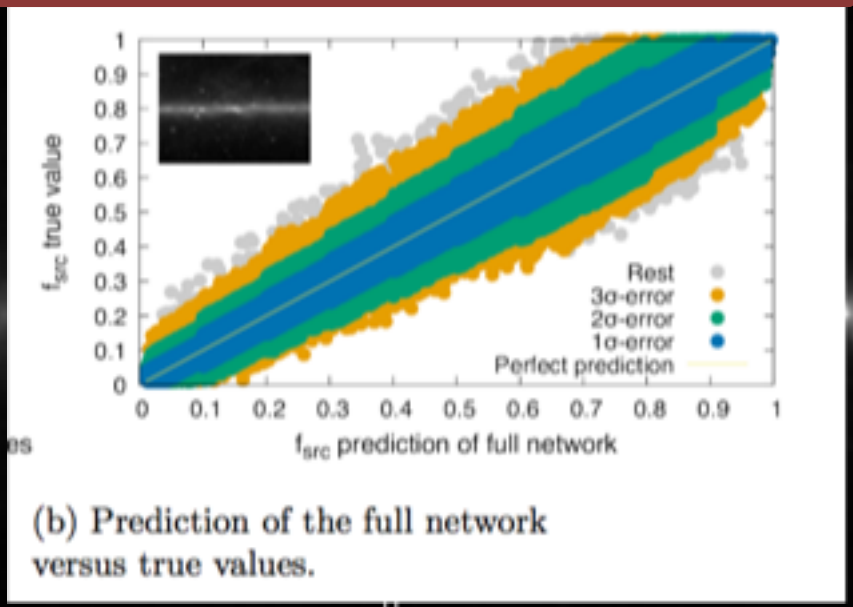
Invert image:

Guess

What is this fraction?

This is 0.5

Network can generalize over randomness



(b) Prediction of the full network versus true values.

Your prediction:

Invert image:

Truth: 0.052

Network: 0.1230

Your guess: 0.5

Who is better? The network

Interpretation here is frequentists and relies on the model to be correct (uncertainties from toy experiments, no p-value yet)

Main message: Parameter determination of the physical model with a Neural Network

- Finally our goal is to **determine the model parameters** from 1 image (“real data”)
- We do this by **training the network on “simulation”** (“simulated data”)
- We need to **ensure that simulation agrees with data:** *Is the true image in the simulation parameter space of images ?* ***If not DM parameters maybe wrong !***

In simple words we do a “fit” to the image including all kinds of “unknown correlations” using a deep convolutional network trained on simulations

➔ Inference

Next steps

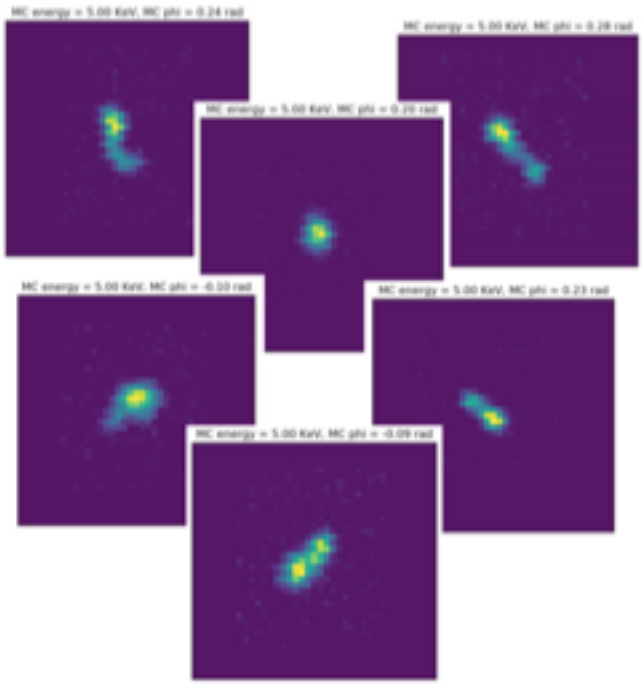
- Categorize objects on the gamma-ray sky

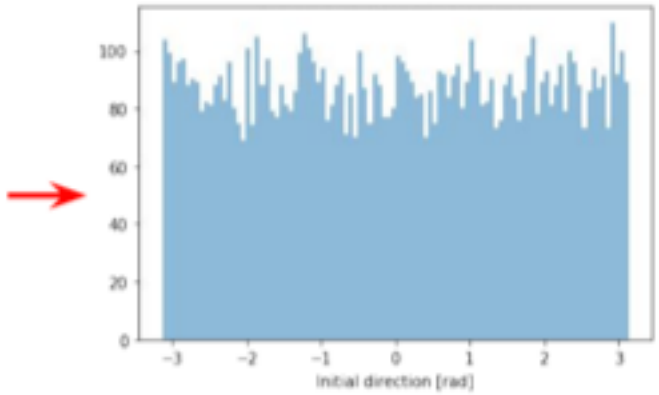
Also point source detection now ➔ New paper called "deepsource" https://arxiv.org/abs/1807.02701

Other example from darkmachines: Event reconstruction with ConvNets

What is the Idea and The Goal of The Challenge?

Original problem we started working on in Leiden: Given a set of simulated track, train a CNN to find the initial direction for each event!

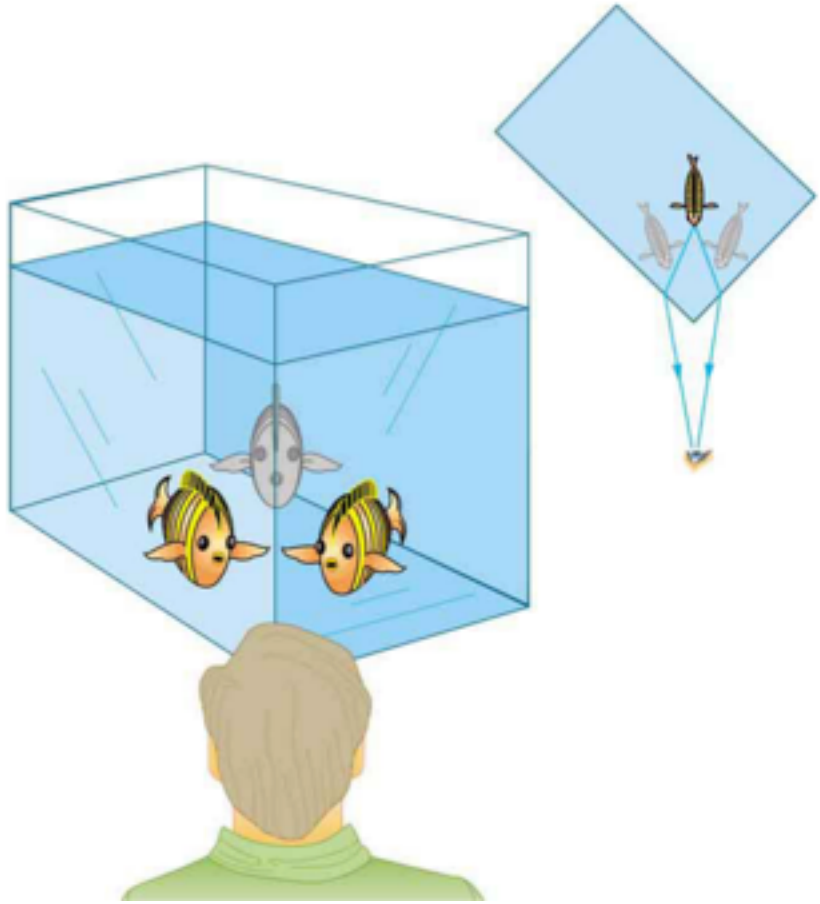




2

Strong Gravitational Lensing

Formation of **multiple images** of a single distant object due to the **deflection of its light** by the **gravity** of intervening structures.



Challenge:

Get mass distribution from seen pictures…

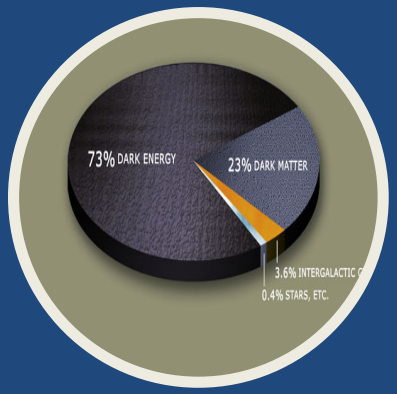
Kick off meeting this week

3.

Physics models, simulators and cross sections (coming back to regression and classification)

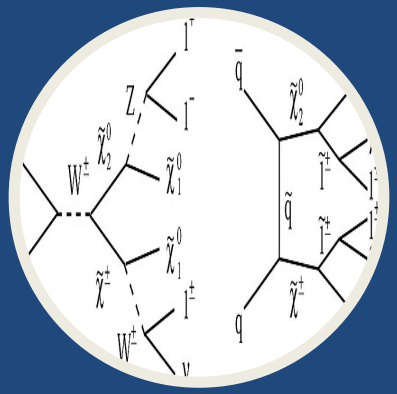
Phenomenology ? Theory?

Dark Matter and the LHC



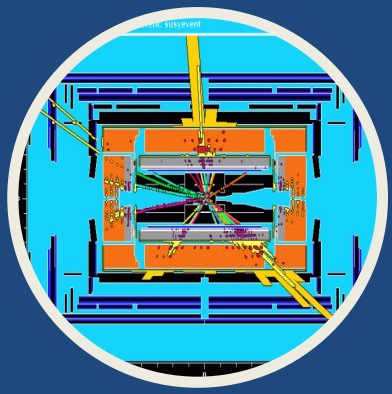
Evidence from Astroparticle physics

- Dark Matter relic density
- Cross section measurements



Theoretical connections/Models

- Supersymmetry
- Extra Dimensions
- … , ??



Consequences for LHC

- LHC phenomenology
- Model testing

Many theoretical models developed to solve other mysteries of the SM like the fine-tuning problem of EWSB turn out to deliver perfect candidates for cold dark matter

Generic Idea for many tasks

Inputs ➔ ***Long simulations + many programs*** ➔ Output

Train classification / regression tool to replace ***this*** by ML

Advantages:

- **Speed ! (see next examples)**
- **Speed allows you to do many many more things…**

Coupling Theory and Machine Learning part 1

"Learning a function" from datasets with known labels sounds boring and old-fashioned.

However we can couple it to simulators+ experiments + phenomenology ….

Coupling Theory and Machine Learning part 1

"Learning a function" from datasets with known labels sounds boring and old-fashioned.

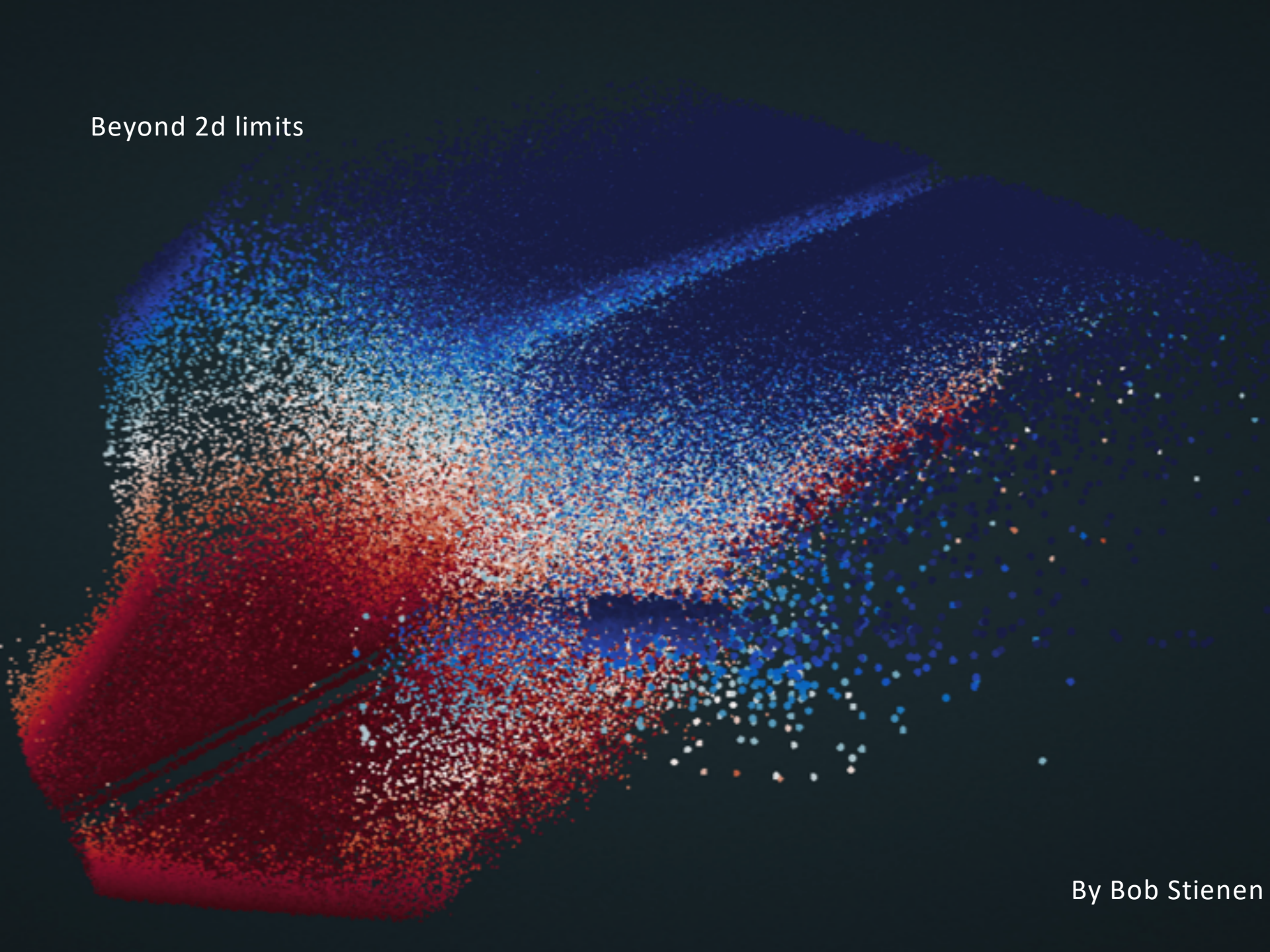
However we can couple it to simulators+ experiments + phenomenology ….

Coupling Theory and Machine Learning part 1

"Learning a function" from datasets with known labels sounds boring and old-fashioned.

However we can couple it to simulators+ experiments + phenomenology ….

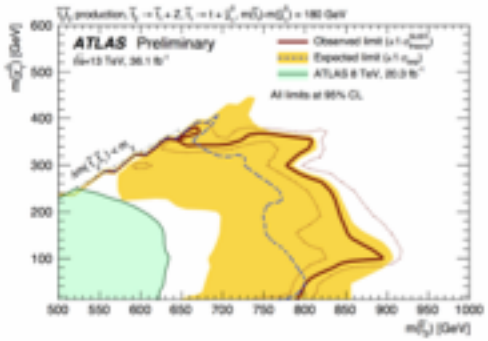
Beyond 2d limits



By Bob Stienen

SUSY-AI

- Exclusion determination in 19d pMSSM
- 310,324 model points with known exclusion as data input
- Algorithm: a collection of decision trees (Random Forest)
- **Idea: going from 2d slices to N-dim representations**



***Prevent overfitting**: Boosting: many trees + not subset of all features for each tree*
*Bagging: random picking training data -> each tree of the forest sees only 0.68*data (see extra slides)*

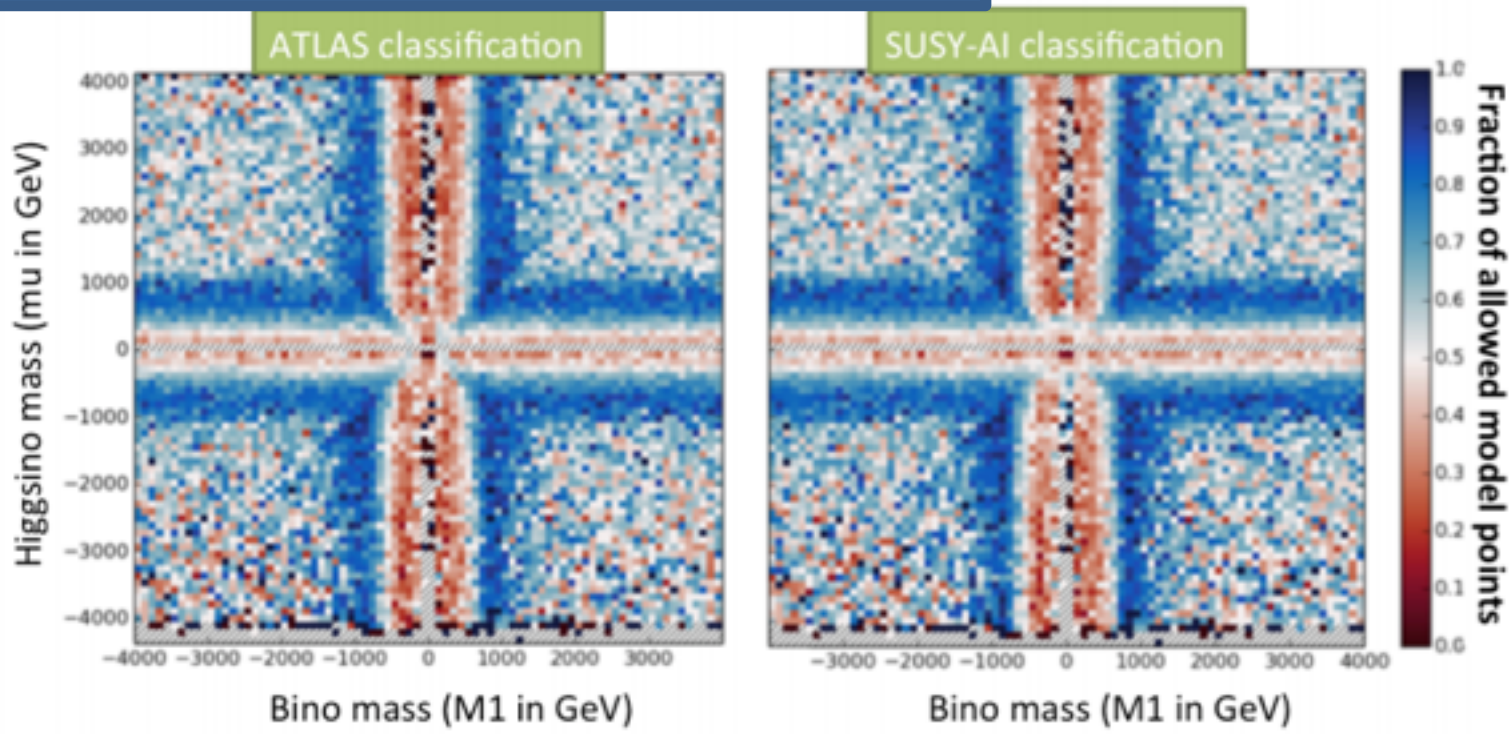
SUSY-AI

Bob Stienen, >10 collaborators

Encoding of model constraints with Machine Learning

Aim: Generic framework (**all** models)

Testing with out-of-bag estimation (remember 0.68!)

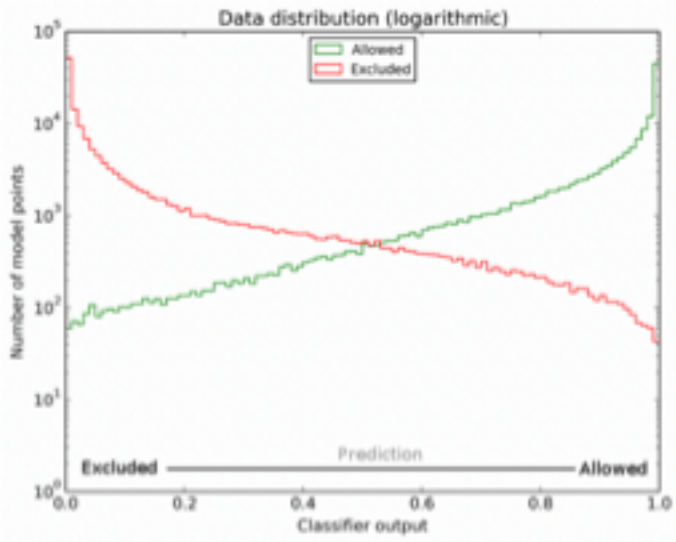


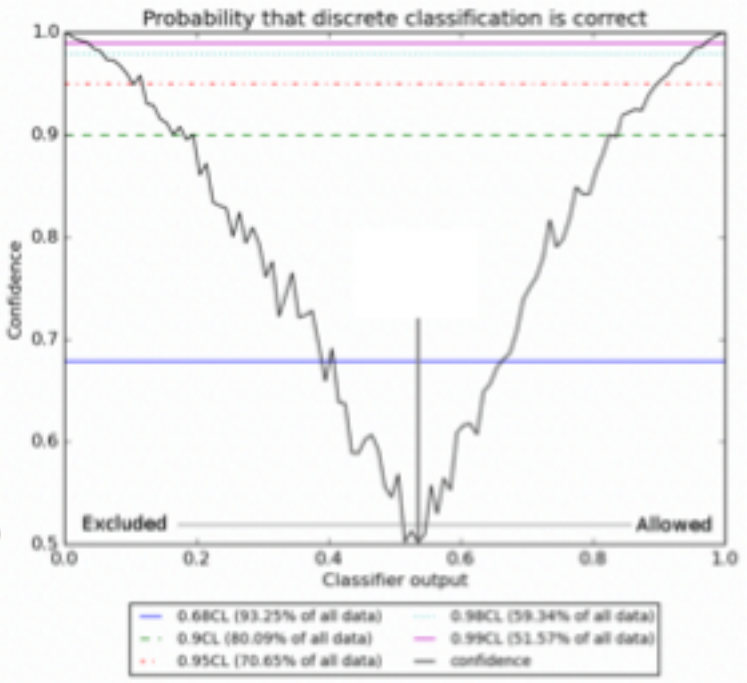
Used training data to learn classification

It determines a **confidence** level of its **classification** using the training data.

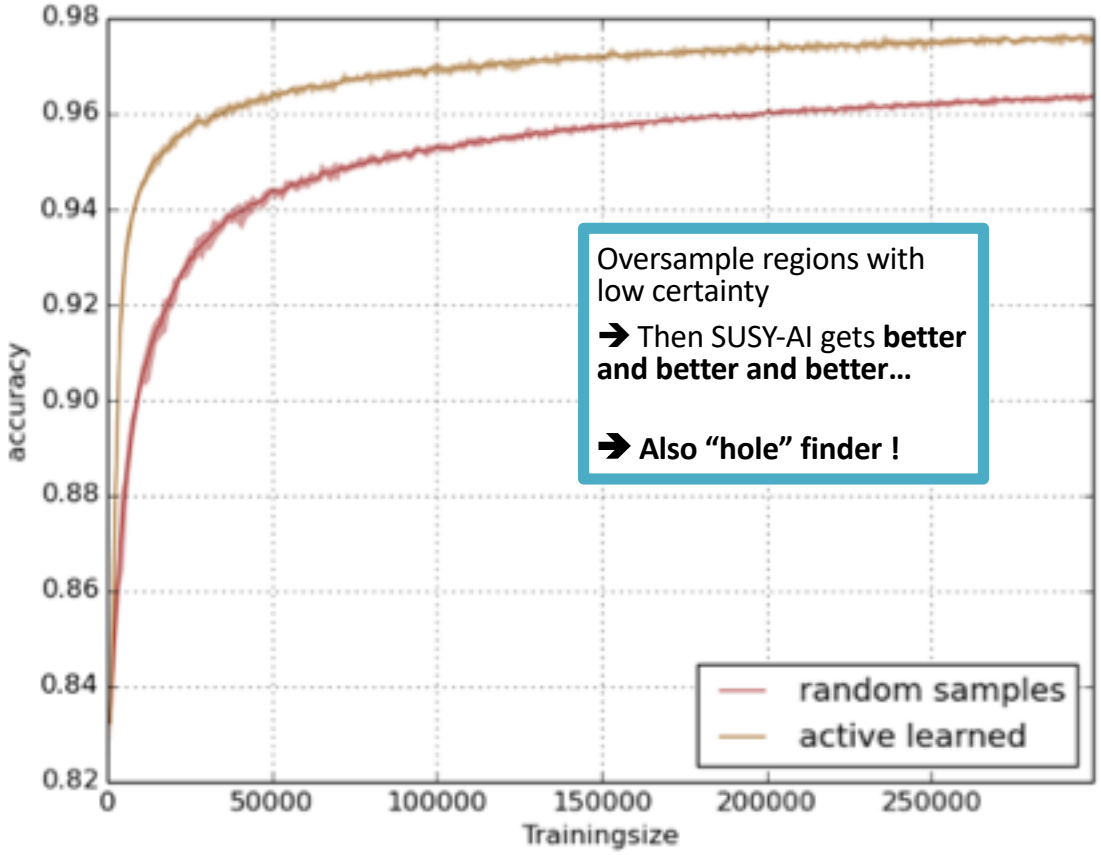
→ Need **more points** in regions of low certainty

Ratio of majority class per bin





How to improve ? Active Learning



Oversample regions with low certainty

➔ Then SUSY-AI gets **better and better and better…**

➔ **Also "hole" finder !**

SUSY-AI --> PhenoAI

- Encoding model constraints for everybody in the full model parameter space (e.g. LHC constraints on various high-dimensional models)

→ NO NEED ANYMORE TO DO SIMPLIFIED MODELS

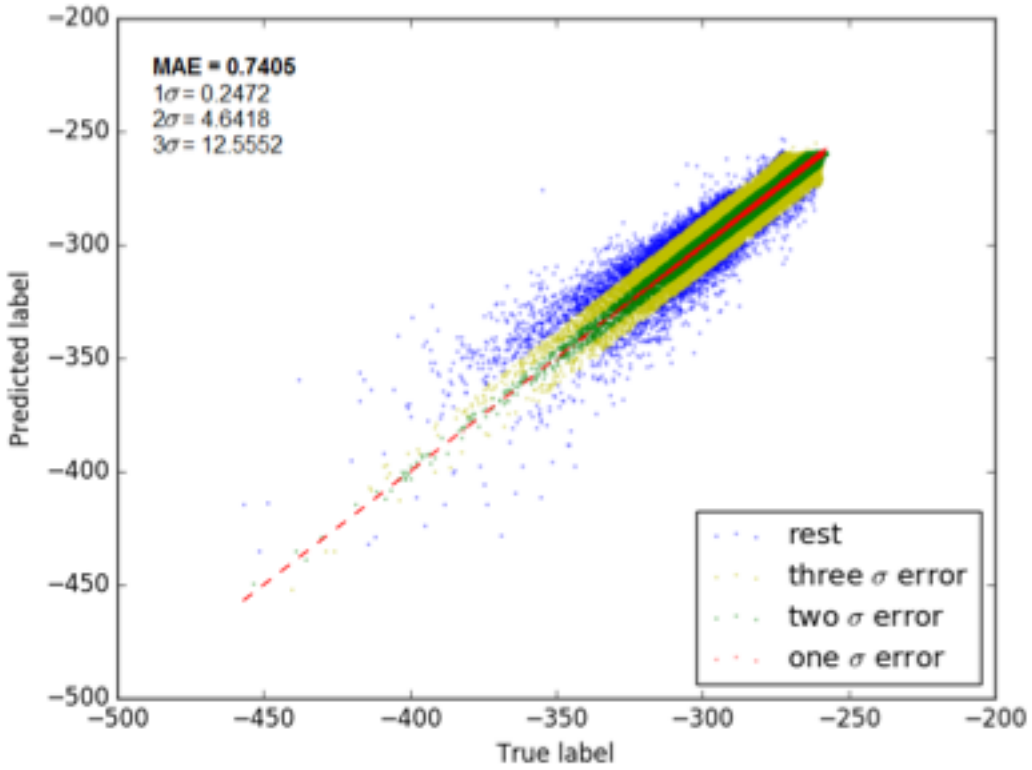
Regression: Cross sections

- Running NLO code to derive SUSY cross sections can take up to 10 minutes
- Can we “learn the cross sections” and derive in a microsecond for *any* model parameter set? ➔ Ongoing project

Regression: Likelihoods

- Fitting groups derive likelihood plots for given models
- Can we “learn the likelihoods” in return in a microsecond for *any* model parameter set ?

BSM-AI regression example… Learning GAMBIT likelihoods



MSSM - 7

https://arxiv.org/abs/1705.07917

Plot by Sydney Otten

4. Data storage

Store THEORY “data” online with your arxiv paper in the full-theory parameter space…

DM model database ?

Within idark project Dutch escience Center

Faruk Diblen

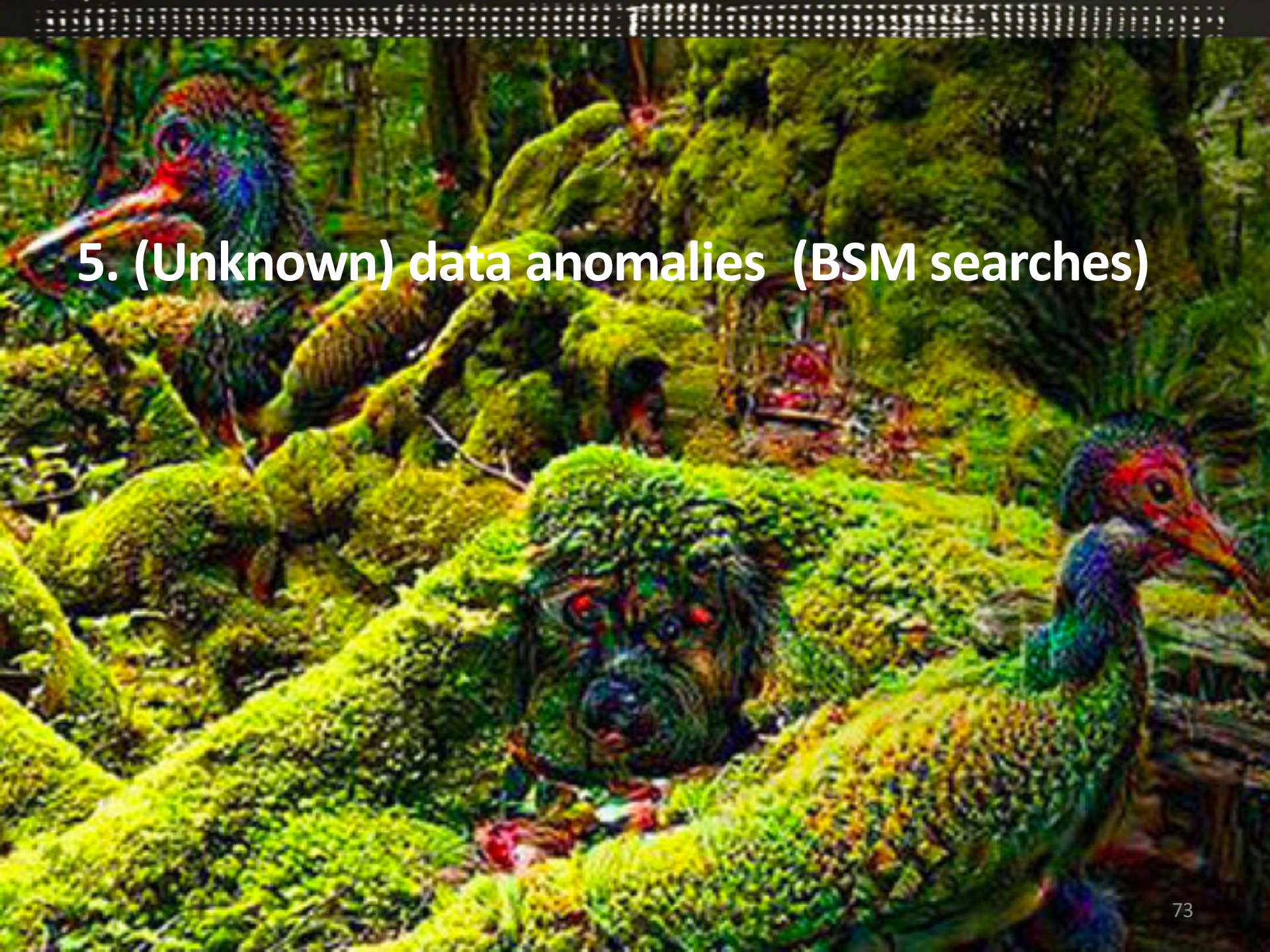
Jisk Attema

Collect model solutions in a database

Use them as target !

Use Machine Learning to interpolate between them → Generalization of DM searches

www.idarksurvey.com


5. (Unknown) data anomalies (BSM searches)

Model parameter space (pre- LHC)

Unknown signals/ unknown labels

Typical task at the LHC is **supervised discrimination of signal and background** (particle ID, Higgs search)

→Discriminator

(typically BDT/TMVA, now Deep Network)

Interesting:

What can we do if the signal is unknown ?

Related to a simpler question:

What can we do if the signal is vaguely known (i.e. a simulation is possible) ?

Attempts

- **In Machine Learning:**

→Unsupervised learning, clustering, anomaly or novelty detection,

→But also “adversarial training” (according to some definition)

Idea: In High Energy Physics use cases for searching for the unknown

→*Sleuth (**hep-ex/0006011**), Generic Searches, bumphunting, some new ideas on arxiv in last 1-2 years using ML*

Unsupervised learning

- Labels unknown (signal or background)
- Learn everything from data

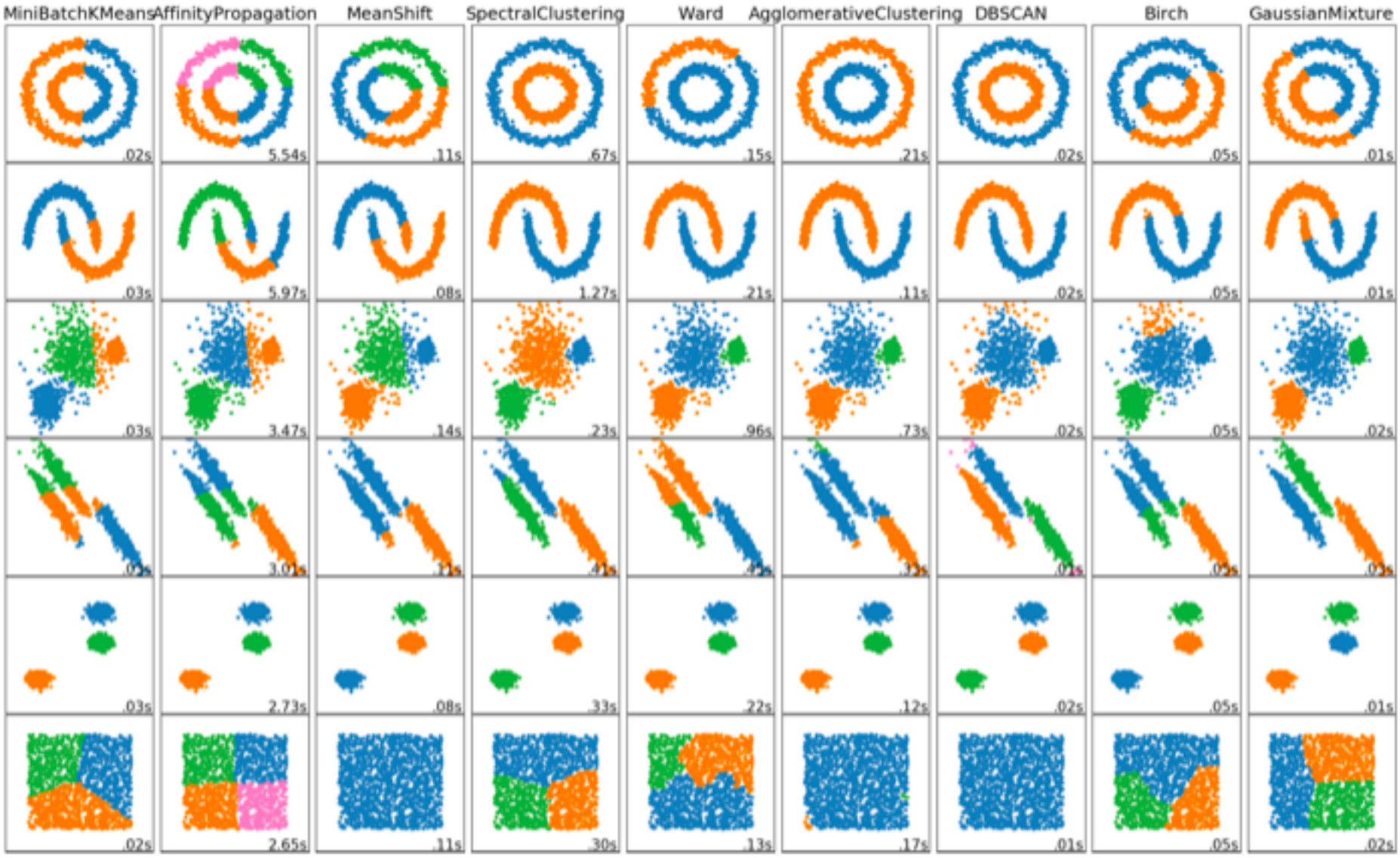
Typical use cases: **Which data belongs together** ?

➔ *finding groups / subgroups from data by investigating, the similarities and correlations of data*

Kids: Learning only from observation + playing

- Finding good variables (e.g. principal components)
- Finding clusters

Clustering (scikit-learn)



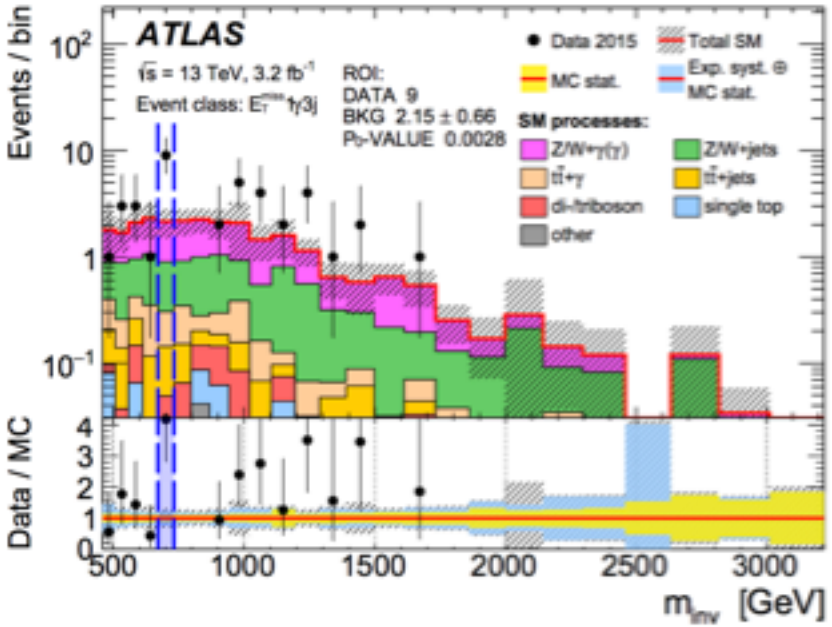
But maybe this is not our problem in HEP ?

Problem is to find new over-densities in the data compared with the SM expectation
not necessary a new cluster or outliers…

Classical approach

- Look everywhere for new overdensities
- Compare data to the SM using a test statistics and a scan algorithm

➔ e.g. General Search (on arxiv now: https://arxiv.org/abs/1807.07447)



Automatize:
>1600 distributions
>800 channels
>10^5 regions

Which quantity is optimal ?
How to determine background ?
How many hypothesis tests are optimal?

Density estimation

What is the problem ?
A search for statistically relevant over-densities in data compared to the Standard Model
taking into account all systematic and theoretical uncertainties.

How to estimate the point density p(x):
without assuming a function
→ Typically done with 3 non-parametric methods

$$p(x) \cong \frac{k}{NV} \text{ where } \begin{cases} V & \text{volume surrounding } x \\ N & \text{total \#examples} \\ k & \text{\#examples inside } V \end{cases}$$

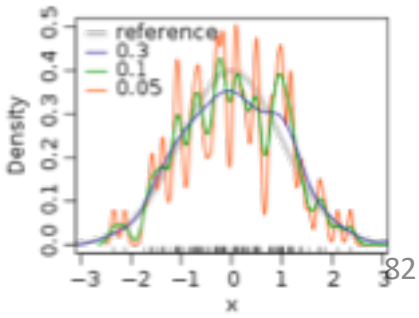
- A histogram → density !

- **K nearest neighbours –type**

→ fix k -> measure the size of volume V in k → density

- **Kernel estimation**

→fix volume → density



Finding a signal

Next step is to find the region

with the largest statistically relevant difference
when comparing the point densities
of data and prediction

Big problem: Take care of the look-elsewhere-effect
→ **Data-derived signal regions in** 1807.07447

New ideas for searches with unknown signal -> recent developments ?

- Fit a ML based background model to be less sensitive on MC prediction (gaussian processes in arXiv:1709.05681)
- Unsupervised techniques (clustering as hypothesis test…)

K- Nearest Neighbour to estimate the point density of two samples, KL-test statistics to compare the samples

- Classification without Labels (CWOLA) arXiv:1805.02664:

Here the idea is to train a NN to seperate signal region + sideband region (as two samples) --> this can be possible due to a signal in the signal region …

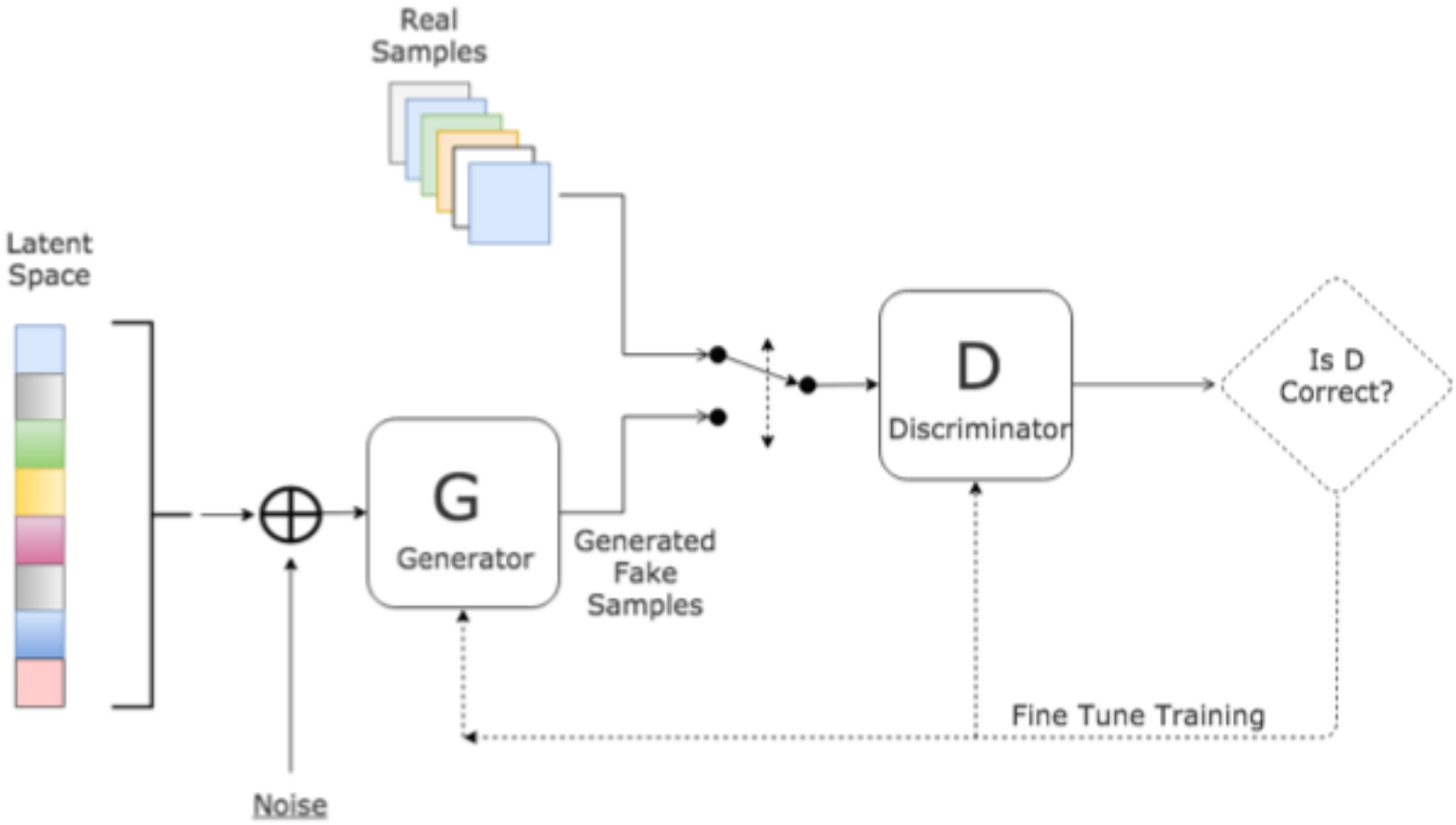
- Autoencoders..
- Self Organising Maps…

6. Adversarial Generative models

Expert in darkmachines
: Gilles Louppe
(Liege)

- Adversarial training = indirectly specifying complicated loss functions.
  - For generation
  - For enforcing constraints
- Directly useful in domain sciences, such as particle physics.

Generative Adversarial Network



arXiv.org > stat > arXiv:1406.2661

Statistics > Machine Learning

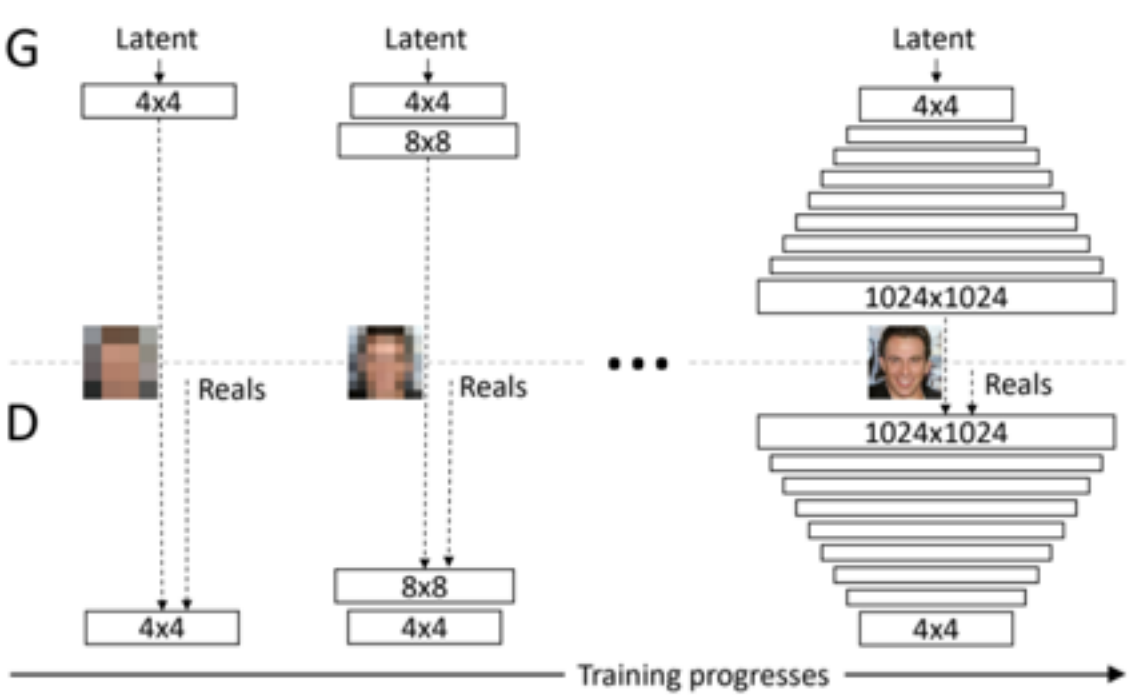
Generative Adversarial Networks

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio

(Submitted on 10 Jun 2014)

State-of-the-art

Karras et al, 2017, arXiv:1710.10196



Generative Networks and Particle Physics

Some ideas for darkmachines.org work group

Various use cases in phenomenology

Various use cases for astrophysics simulation

Let's discuss some ideas …

Some new things to look at are e.g. Coulomb GANs :
Used to generate samples according to the whole target distribution

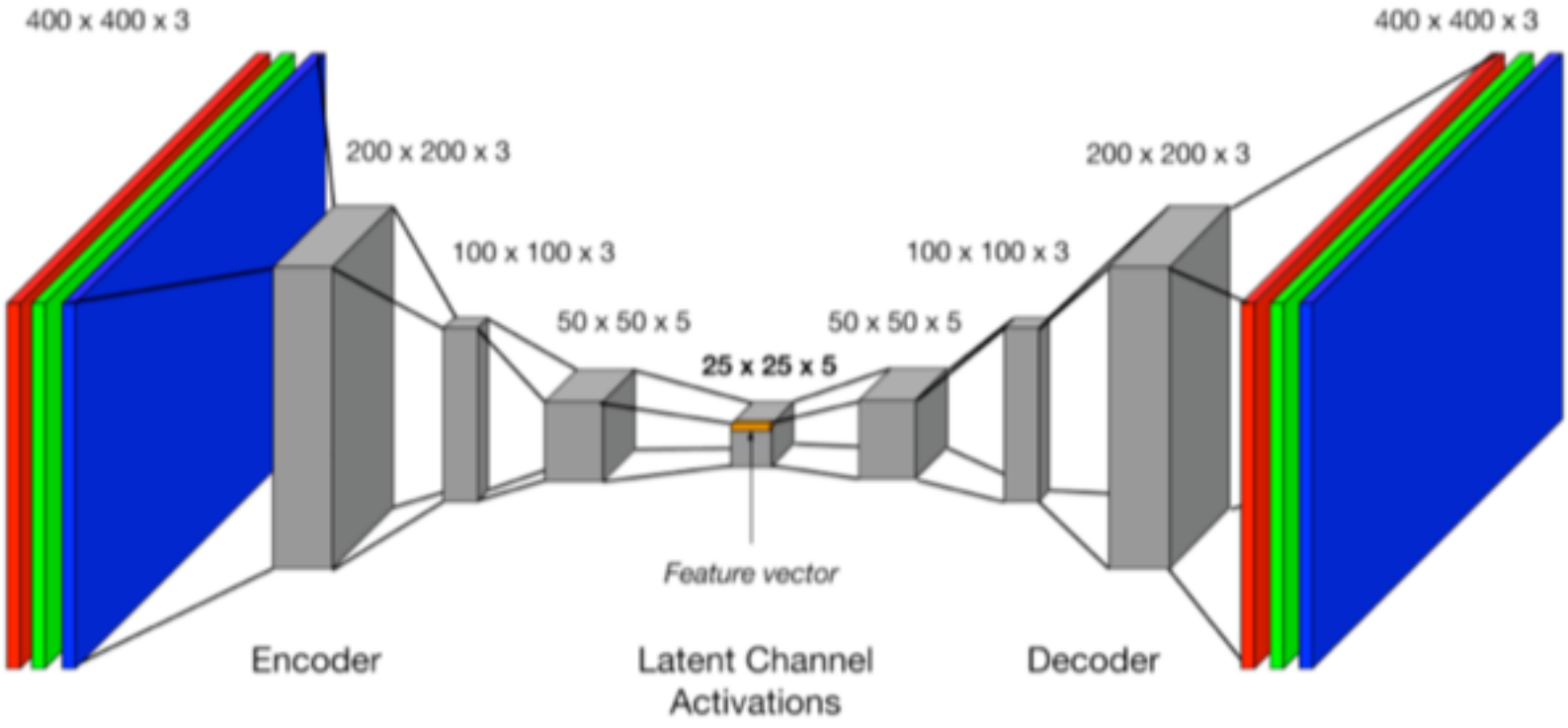
Summaries of Ideas ?

- **Convolutional Networks** → Putting symmetries/operation into the network structure → Image Analysis and much more !
- **Training methods** → Going beyond supervised learning !
- **Generative Networks** → A generator ? Wait we have this in physics since "von Neumann" ?
- **Adversarial Setups** → We can let networks get better by letting them fight against each other
- **Autoencoders and Latent Space** → Optimal decoding, finding the optimal parameter space
- **Active Learning and Sampling Algorithms** → I can do better sampling when I use Machine Learning
- **Unsupervised learning … /Clustering/Anomaly detection/data-derived** → Want do this for LHC , join the effort in darkmachines…

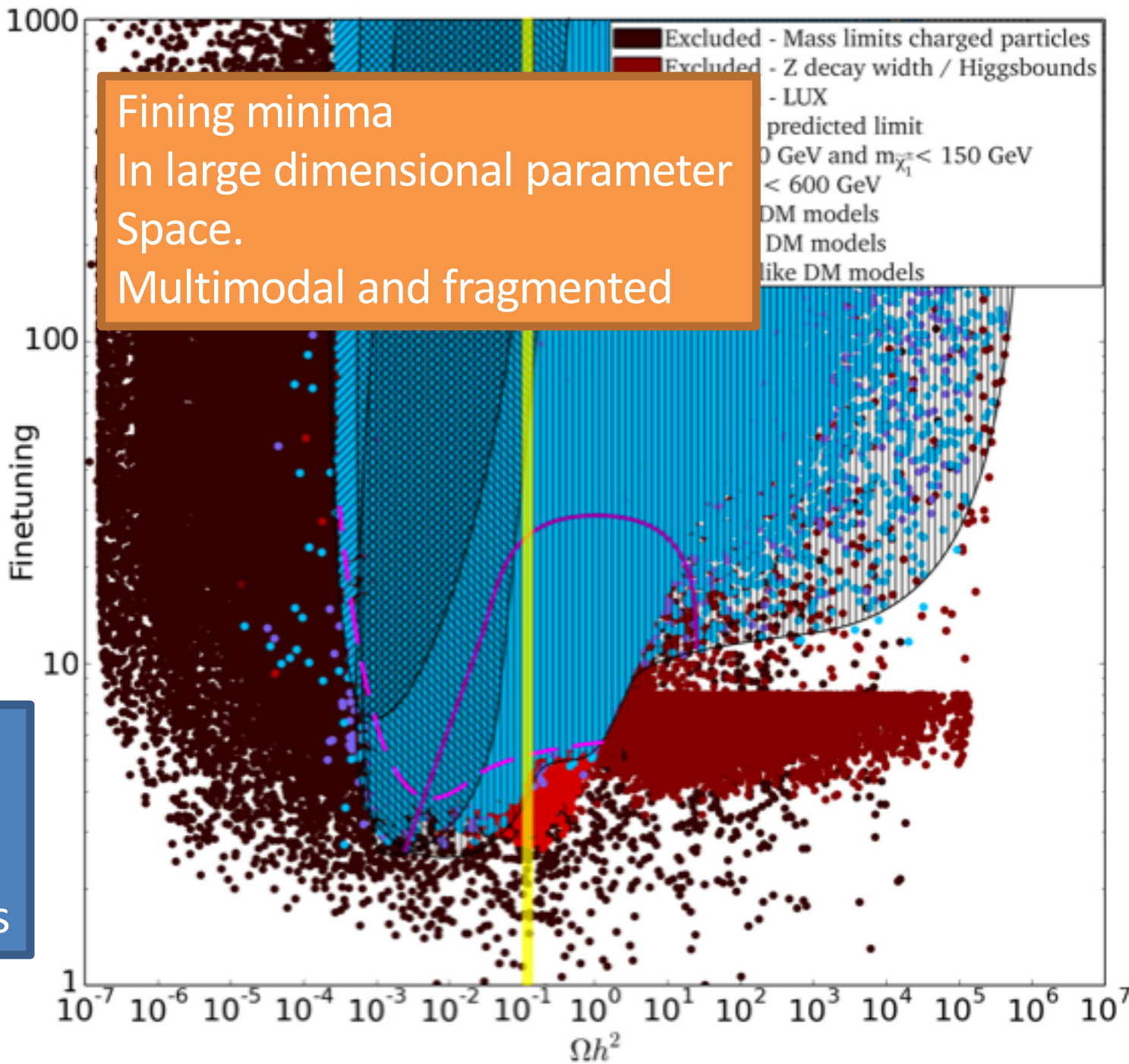
→ Join the effort ! There is something in for you !

Extra Slides

Autoencoder

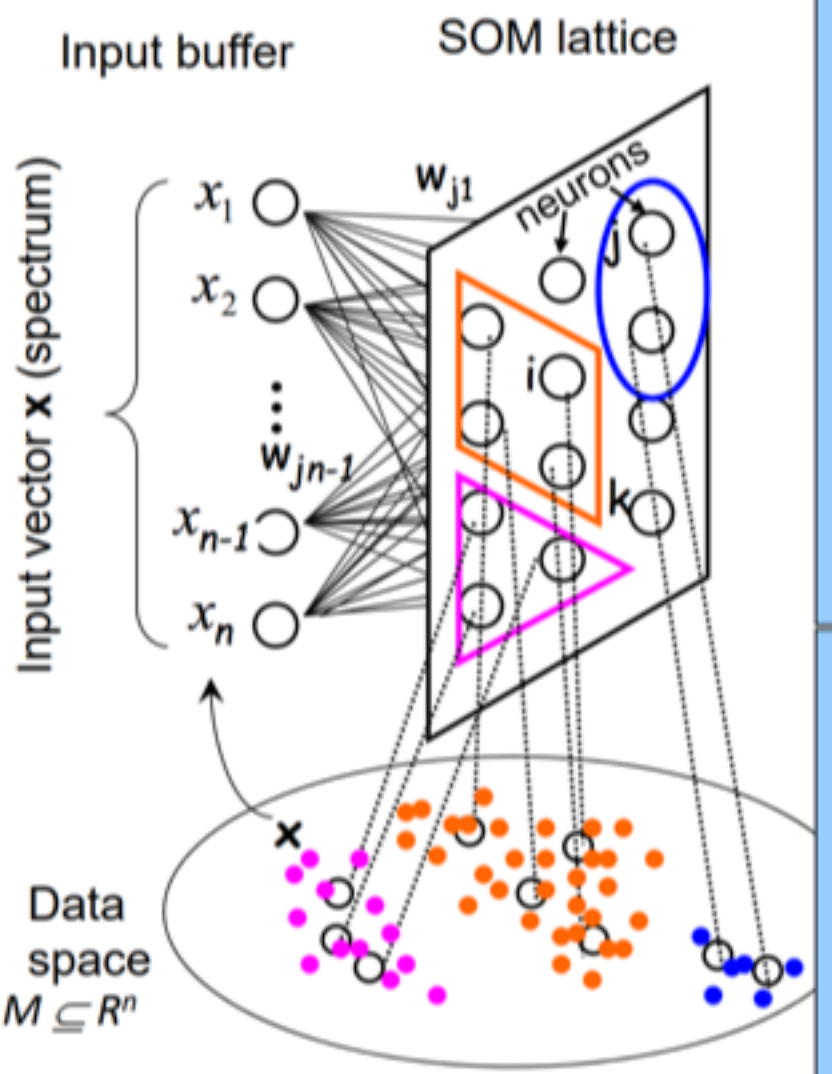


Dark Matter relic density



Plots by Melissa van Beekveld and Ruud Peters

Learn the data structure with Self-Organizing Maps
Machine learning analog of biological neural maps in the brain



Two simultaneous actions:

- ***Adaptive*** Vector Quantization (VQ): puts the prototypes in the "right" locations => allows summarization of N data vectors by ***O(sqrt(N)) prototypes***; while encoding salient properties

- Ordering of the prototypes on the SOM grid according to similarities; *only SOMs do this*.

I.e., SOM learns the structure (the distribution) AND expresses the topology (similarity relations) on a low-dimensional lattice.

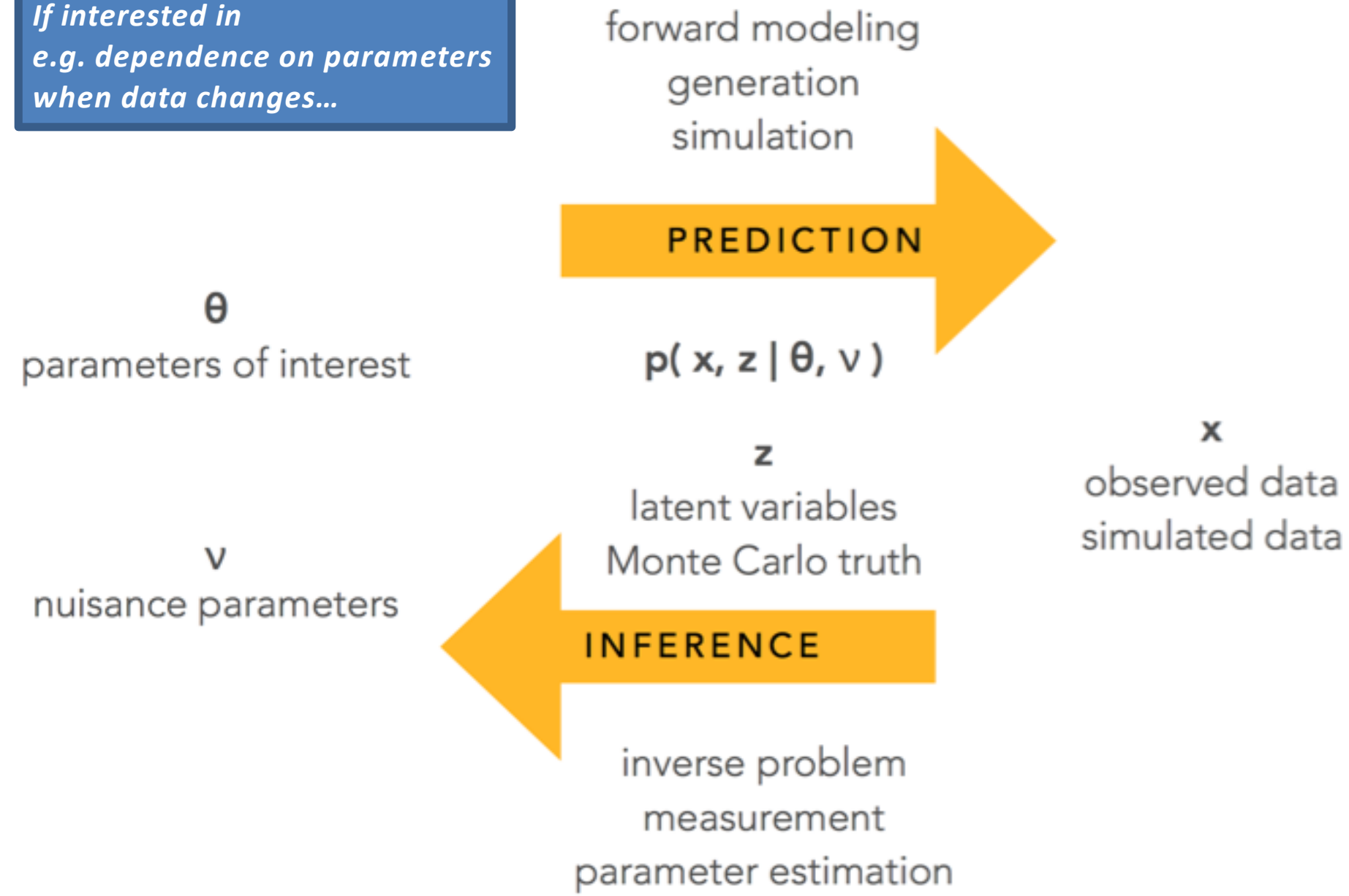
Finding the prototype groups: post-processing – segmentation of the SOM based on representations of the SOM's knowledge

E. Merényi, Rice U
erzsebet@rice.edu

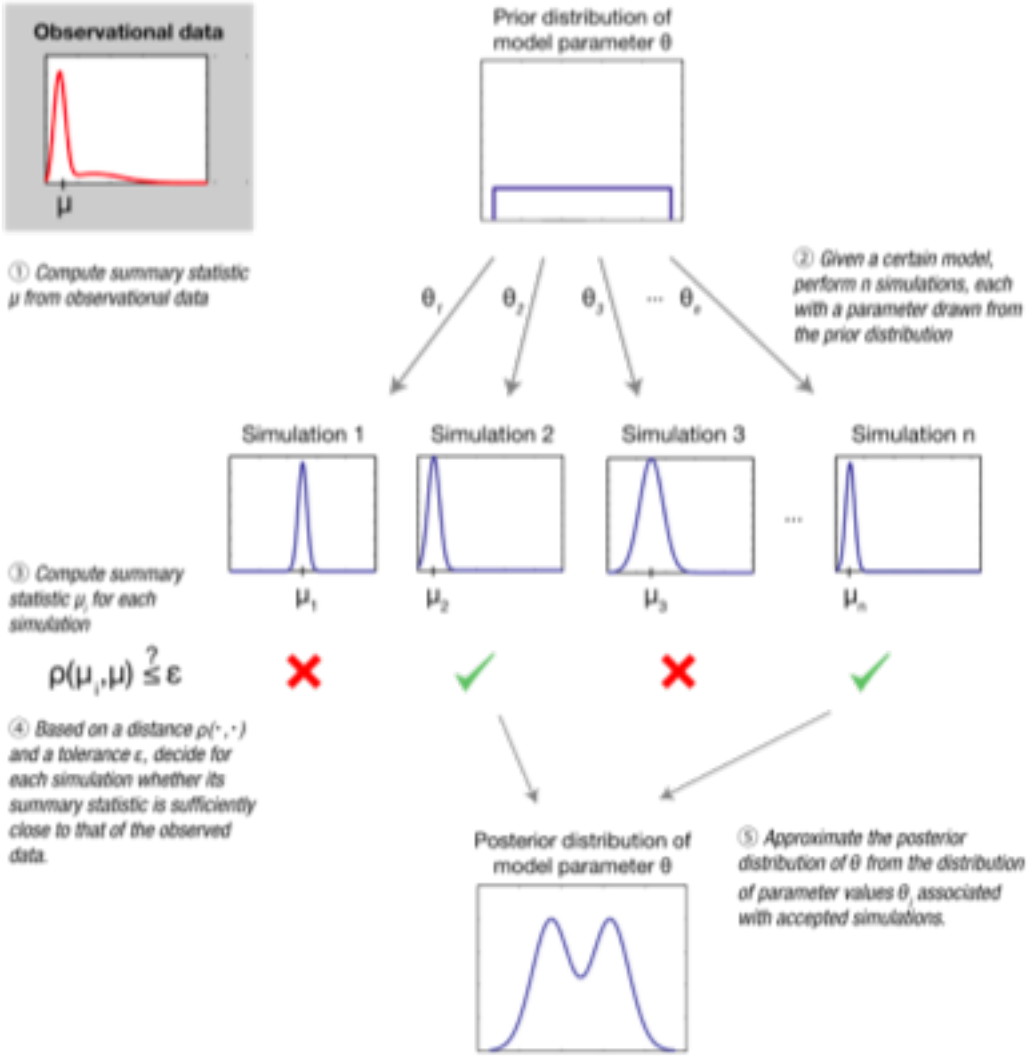
THE PLAYERS

Nice slide by Kyle Cranmer

If interested in
e.g. dependence on parameters
when data changes…



Inference example



(various ML ways to do inference without assuming 1d distributions, see e.g. Cranmer and Gilles Louppe papers or Approximate Bayesian Computation ABC)

On could also train all classifiers at the same time and introduce some randomness in the training. This randomness can be provided by bootstrap aggregating (*bagging*). The procedure behind this method is the following: given a training set D of size n, m new datasets D_i with size n' are created by selecting data out of D with replacement. Every data point in D has a chance of

$$\mathrm{P(picked)} = 1 - \left(1 - \frac{1}{n}\right)^n \tag{4.9}$$

of being picked in D_i, which – in the limit of n to infinity – becomes for $n' = n$:

$$\lim_{n\to\infty} \mathrm{P(picked)} = \lim_{n\to\infty} 1 - \left(1 - \frac{1}{n}\right)^n = 1 - \frac{1}{e} \approx 0.632 \tag{4.10}$$

The fraction of data points in D present in D_i it thus approximately 63.2% [55]. Although for single algorithm this method decreases the quality, in large boosted algorithms it increases it, since mistakes in the prediction made by a single algorithm are averaged out.

Taken from B. Stienen, Masterthesis