LEARNING NEW PHYSICS FROM A MACHINE

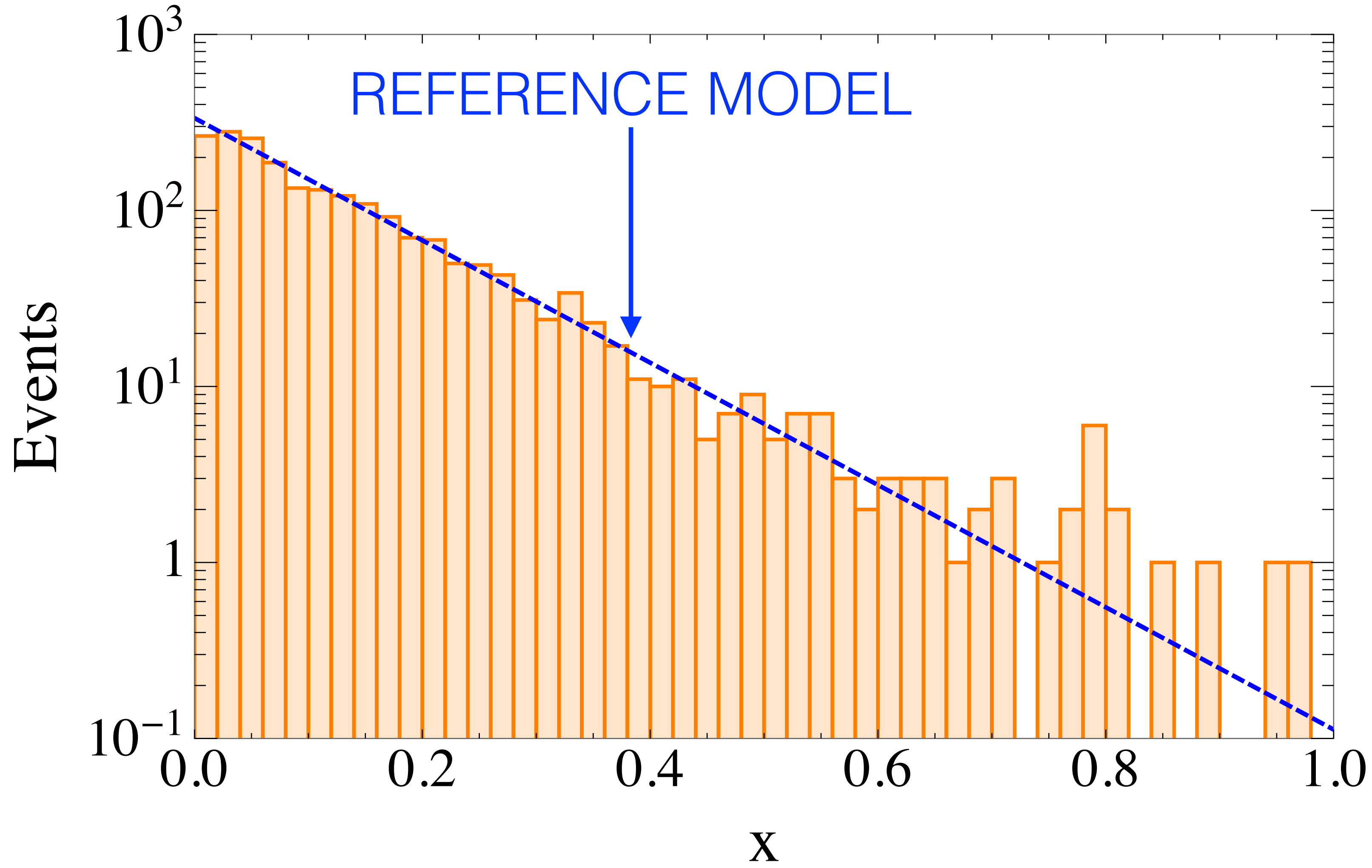
Raffaele Tito D'Agnolo - SLAC

GGI 2018

RTD and Andrea Wulzer

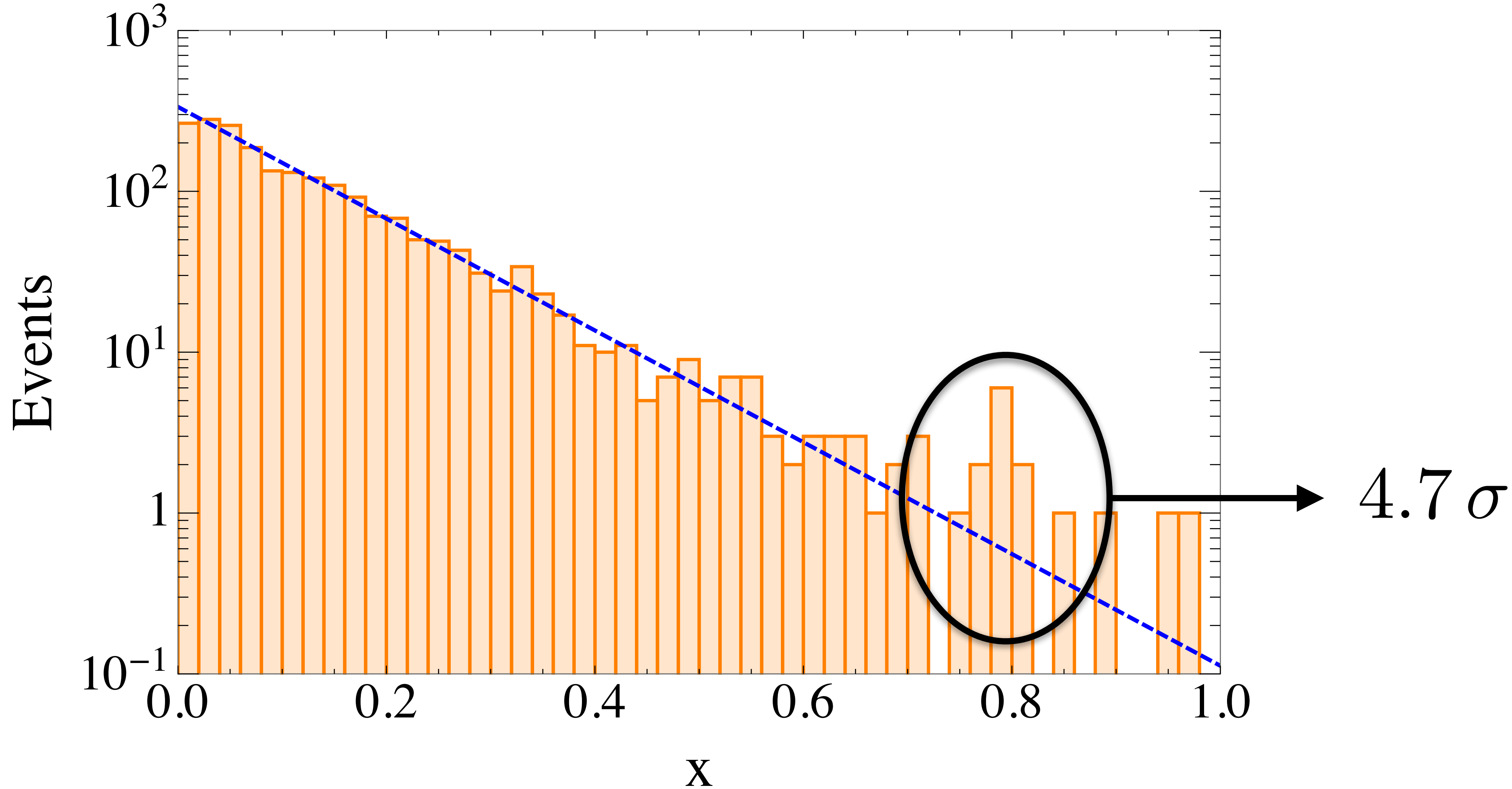
arXiv:1806.02350

THE PROBLEM



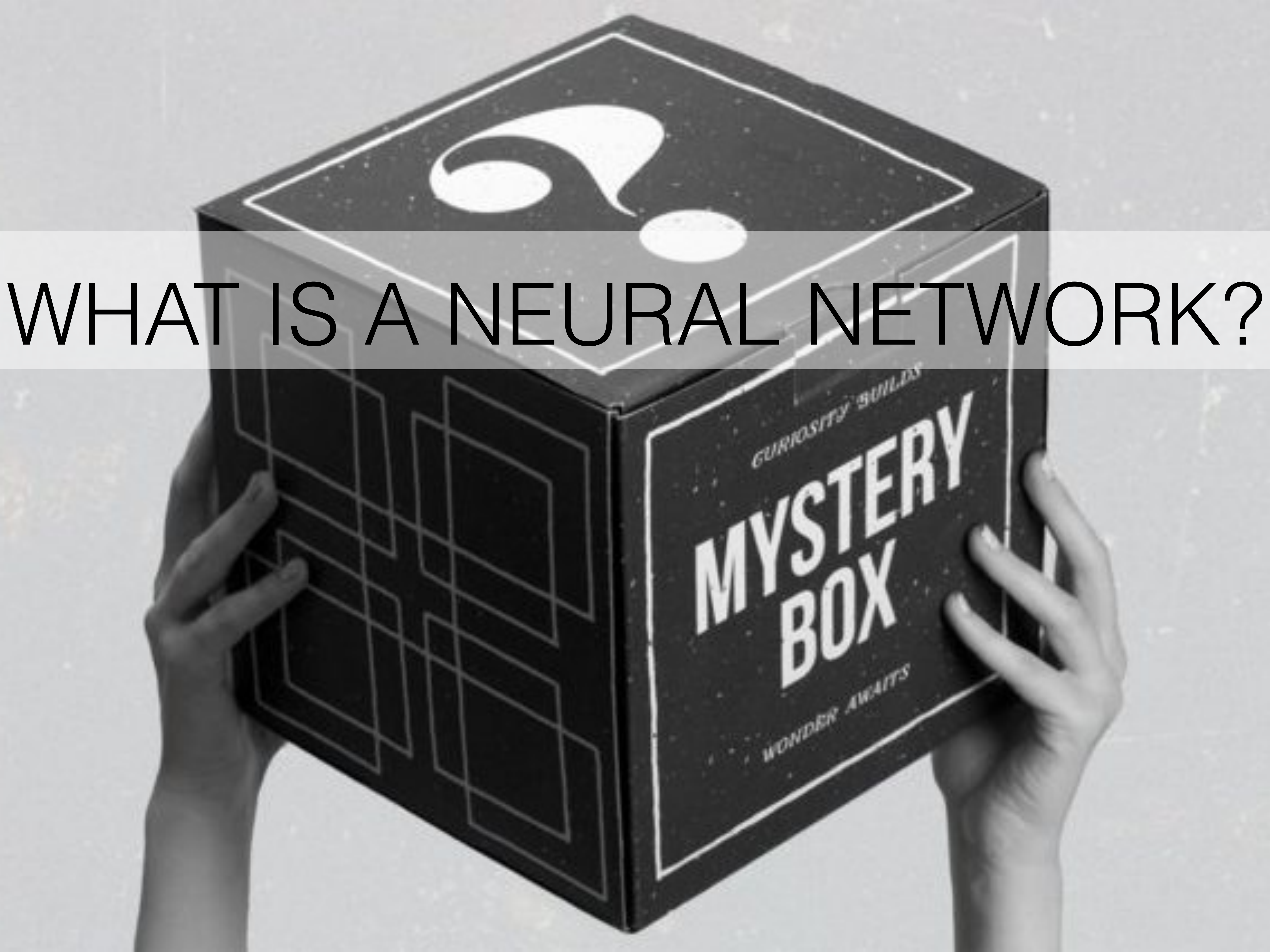
$$\chi^2 = 47 \quad N_{\rm bins} = 50 \quad p-\text{value} < 1\sigma$$

THE PROBLEM



$$t_{\mathrm{id}}(\mathcal{D}) = 2\,\log\left[\frac{e^{-N(\mathrm{NP})}}{e^{-N(\mathrm{R})}}\prod_{x\in\mathcal{D}}\frac{n(x|\mathrm{NP})}{n(x|\mathrm{R})}\right]$$

WHAT IS A NEURAL NETWORK?



WHAT IS A NEURAL NETWORK?

SET OF FUNCTIONS

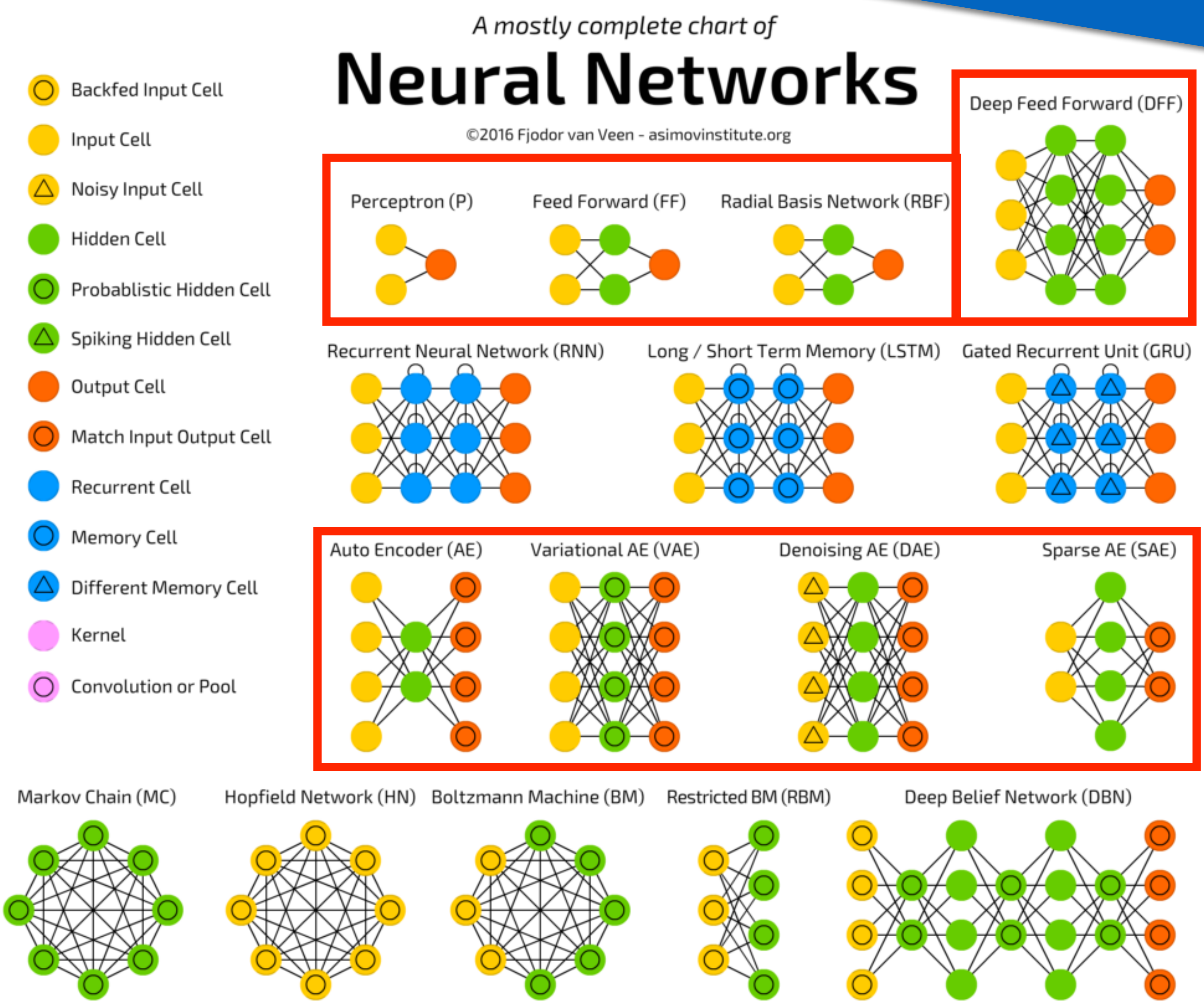
+

FITTING ALGORITHM

WHAT IS A NEURAL NETWORK?

SET OF FUNCTIONS

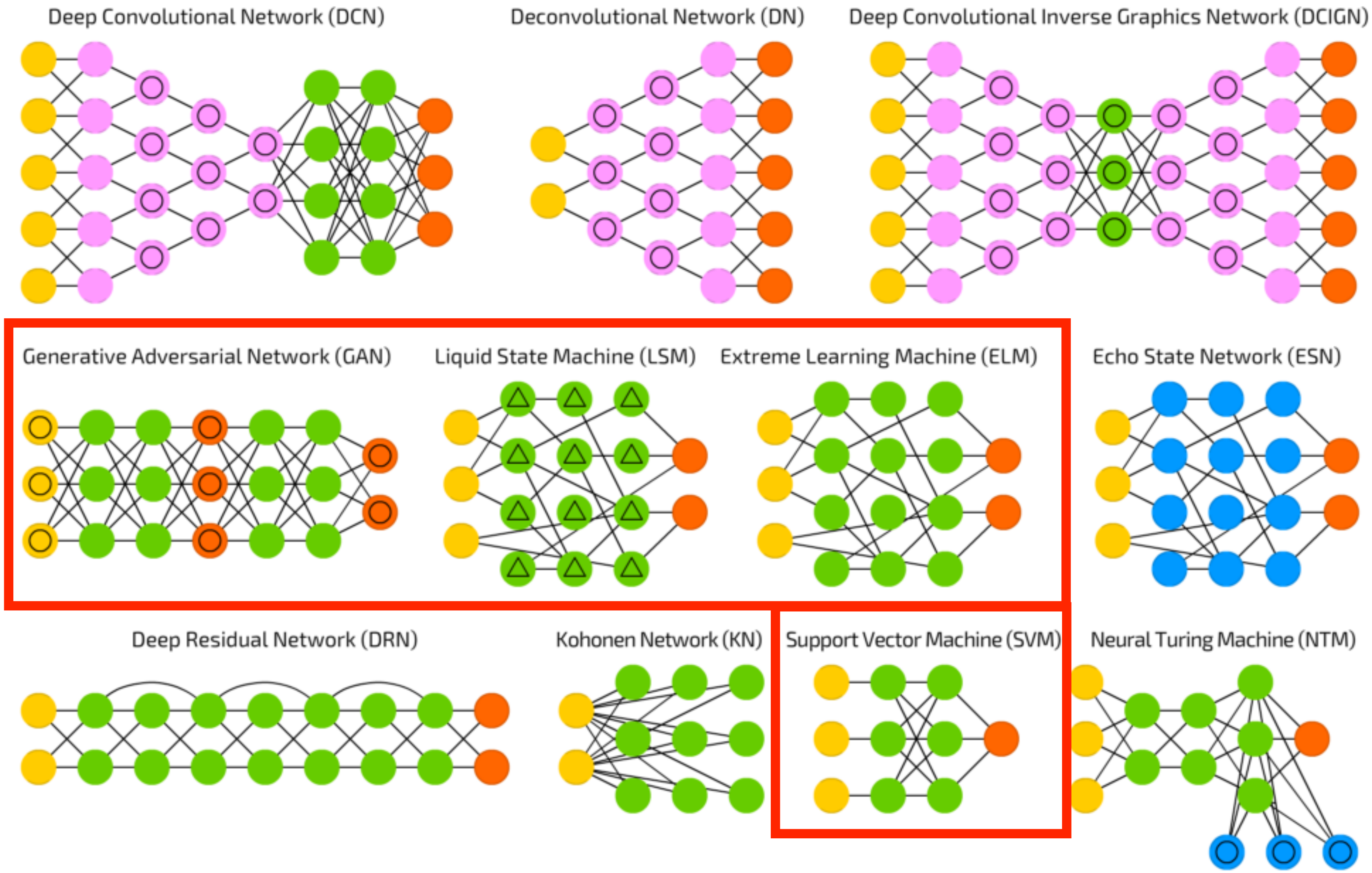
$$f_{w_1}^{(1)}\left(f_{w_2}^{(2)}\left(f_{w_3}^{(3)}(\ldots)\right)\right)$$



https://towardsdatascience.com/

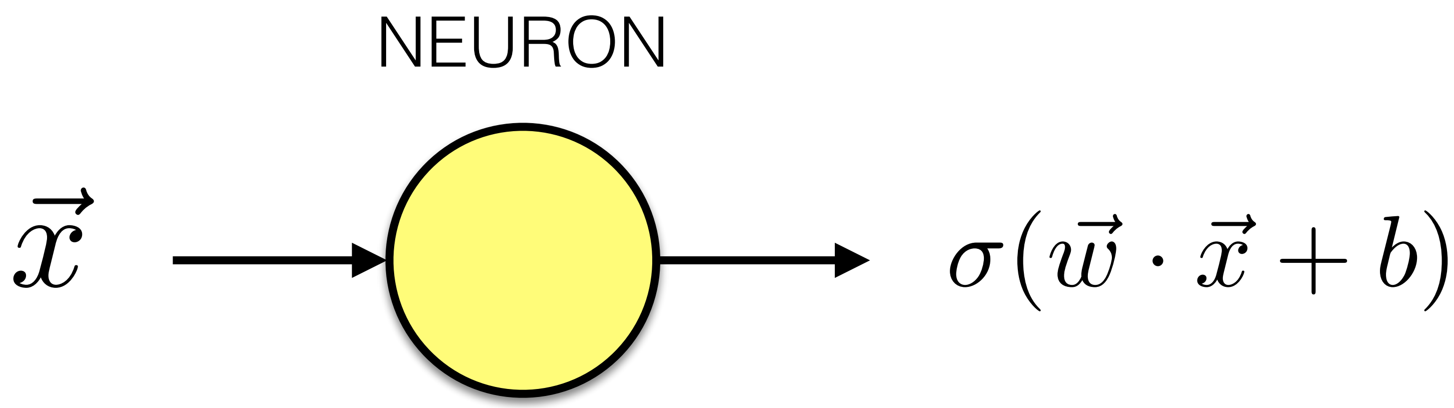
A mostly complete chart of

Neural Networks



https://towardsdatascience.com/

BUILDING BLOCKS



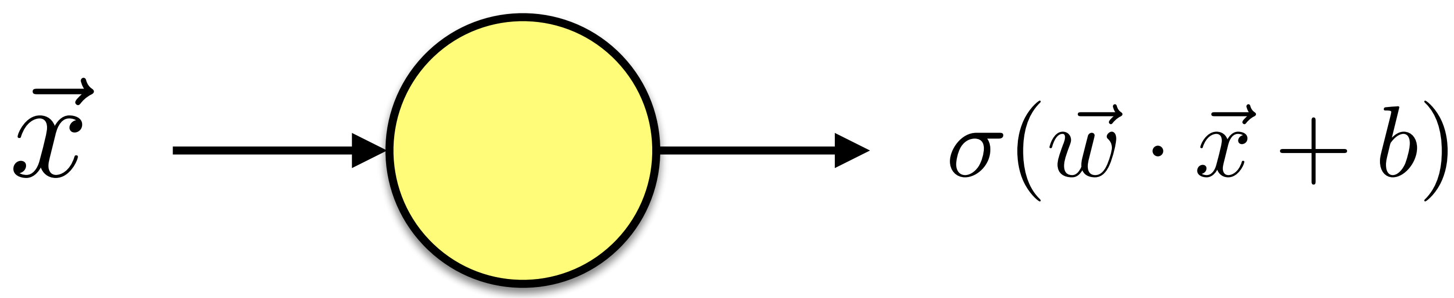
1. LINEAR TRANSFORMATION $z = \vec{w} \cdot \vec{x} + b$

FREE PARAMETERS

2. NON-LINEAR TRANSFORMATION $\sigma(z)$ FIXED

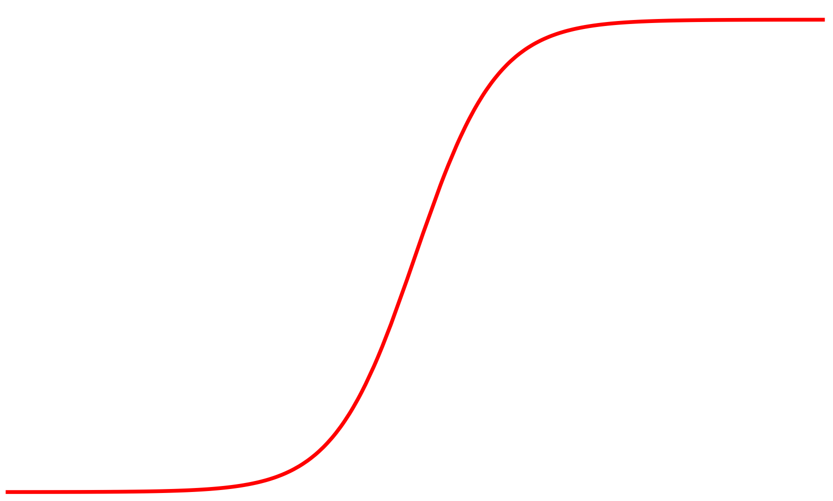
BUILDING BLOCKS

NEURON



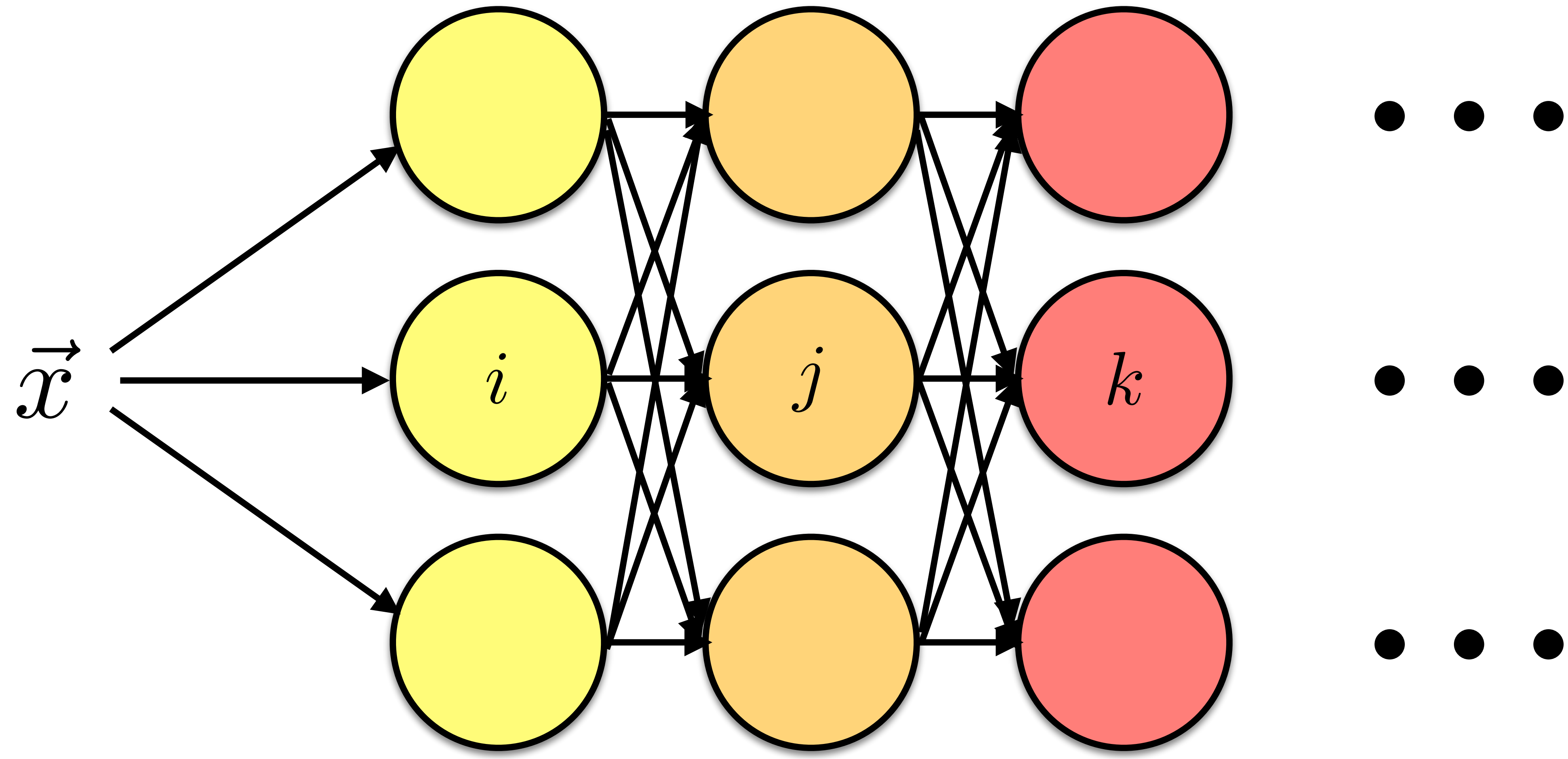
2. NON-LINEAR TRANSFORMATION

$$\sigma(z) = \begin{cases} \tanh(z) \\ \text{ReLU} \\ \frac{1}{1+e^{-z}} \\ \ldots \end{cases}$$



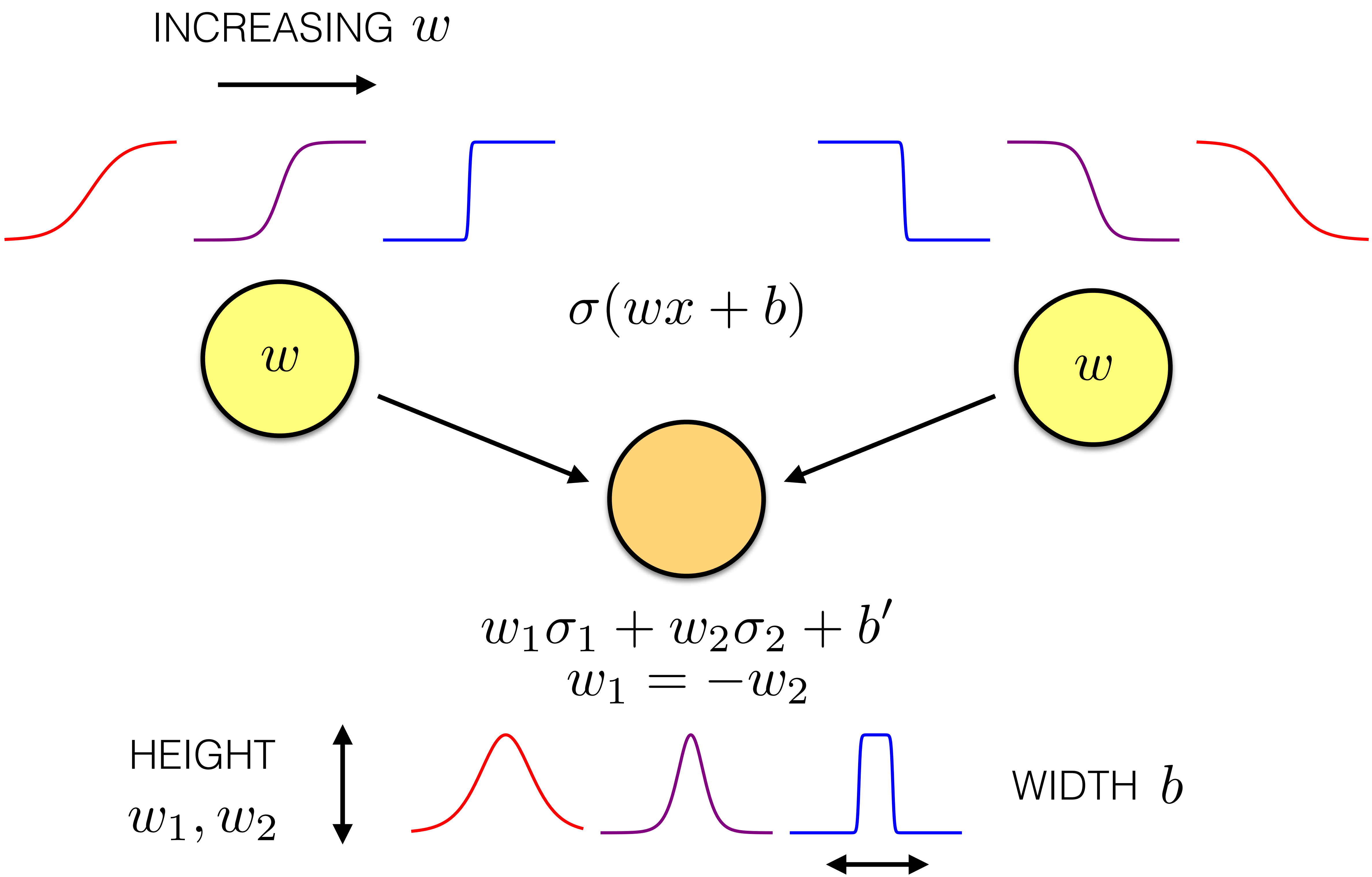
THE NETWORK

FEEDFORWARD, FULLY CONNECTED



$$\sigma\left(\sum_{k=1}^{3} w_{jk}\sigma_k\left(\sum_{i=1}^{3} w_{ki}\sigma_i\left(\sum_{l=1}^{d} w_{il}x_l + b_i\right) + b_k\right) + b_j\right)$$

UNIVERSAL APPROXIMANTS



MAXIMUM LIKELIHOOD

TO ESTIMATE THE UNKNOWN PARAMETERS θ MAXIMIZE THE PROBABILITY $\mathcal{L}(\theta; x)$ THAT THEY DESCRIBE THE OBSERVED DATA x

$$\hat{\theta} = \arg\max \mathcal{L}(\theta; x)$$

- CONSISTENT (CONVERGES IN PROBABILITY TO THE TRUE VALUE)
- EFFICIENT (SATURATES THE CRAMÉR-RAO BOUND)
- ASYMPTOTICALLY GAUSSIAN

MAXIMUM LIKELIHOOD DICTIONARY

ELEMENTARY STATISTICS	NEURAL NETWORK
PARAMETERS θ	WEIGHTS AND BIASES w, b
LIKELIHOOD $\mathcal{L}(\theta; x)$	LOSS FUNCTION $-L(w, b; x)$
DATA	TRAINING SAMPLE

FITTING ALGORITHM (SUPERVISED)

LOSS FUNCTION(AL)

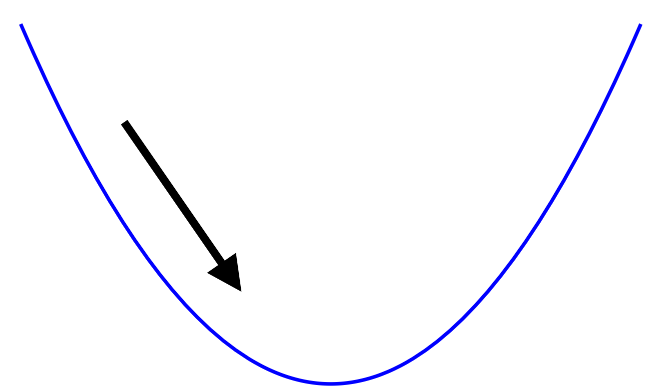
$$L = \frac{1}{N_c}\sum_{i=1}^{N_c}[1 - f_{NN}(\vec{x}_i, \mathbf{w}, \mathbf{b})]^2 + \frac{1}{N_d}\sum_{j=1}^{N_d}[f_{NN}(\vec{x}_j, \mathbf{w}, \mathbf{b})]^2$$

TRAINING

$$w_{t+1} \rightarrow w_t - \epsilon\partial_w \hat{L}$$

$\hat{L}$ SUBSET OF THE SAMPLE

ϵ LEARNING RATE

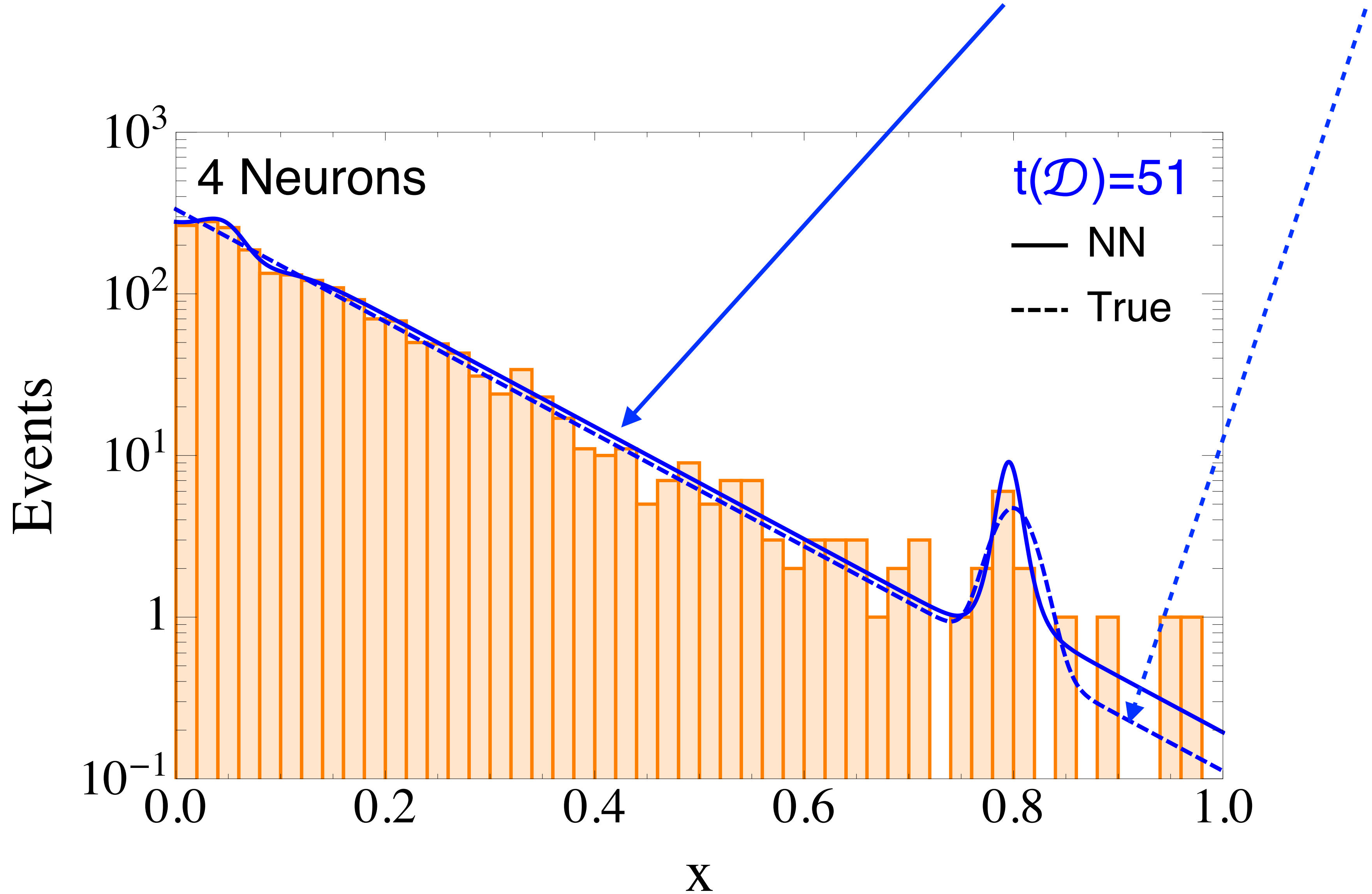



arXiv:1806.02350
LEARNING NEW PHYSICS

A SIMPLE STRATEGY

BINNED HISTOGRAM ⟶ SMOOTH APPROXIMANT

1. LEARN THE DATA DISTRIBUTION $n(x|\widehat{\mathbf{w}}) \approx n(x|\mathrm{T})$



A SIMPLE STRATEGY

1. LEARN THE DATA DISTRIBUTION $n(x|\widehat{\mathbf{w}}) \approx n(x|\mathrm{T})$

2. CHECK IF IT IS DIFFERENT FROM THE REFERENCE ONE

$$t(\mathcal{D}) = 2\,\log\left[\frac{e^{-N(\widehat{\mathbf{w}})}}{e^{-N(\mathrm{R})}}\prod_{x\in\mathcal{D}}\frac{n(x|\widehat{\mathbf{w}})}{n(x|\mathrm{R})}\right] \qquad p_{\mathrm{obs}} = \int_{t_{\mathrm{obs}}}^{\infty} dt\, P(t|\mathrm{R})$$

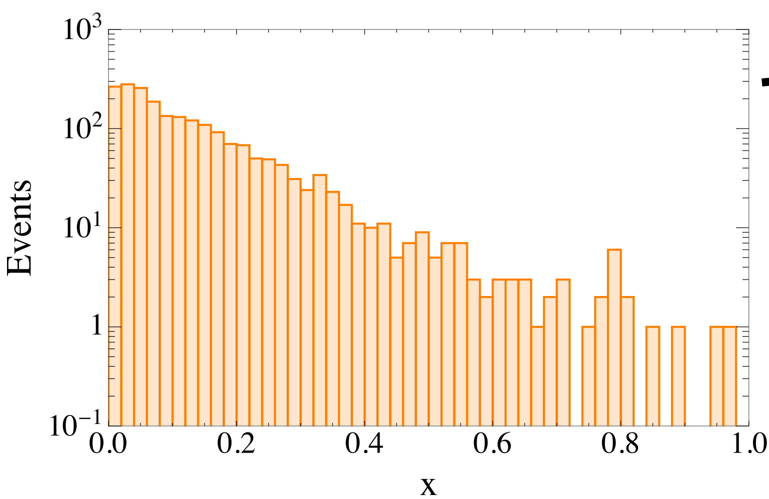
STANDARD LIKELIHOOD RATIO
NEYMAN-PERSON TEST STATISTIC

REFERENCE
DISTRIBUTED
TOYS

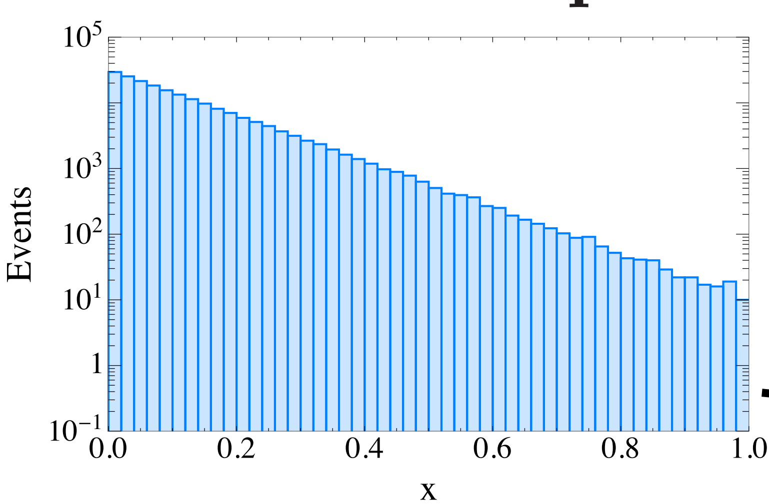
SUMMARY I

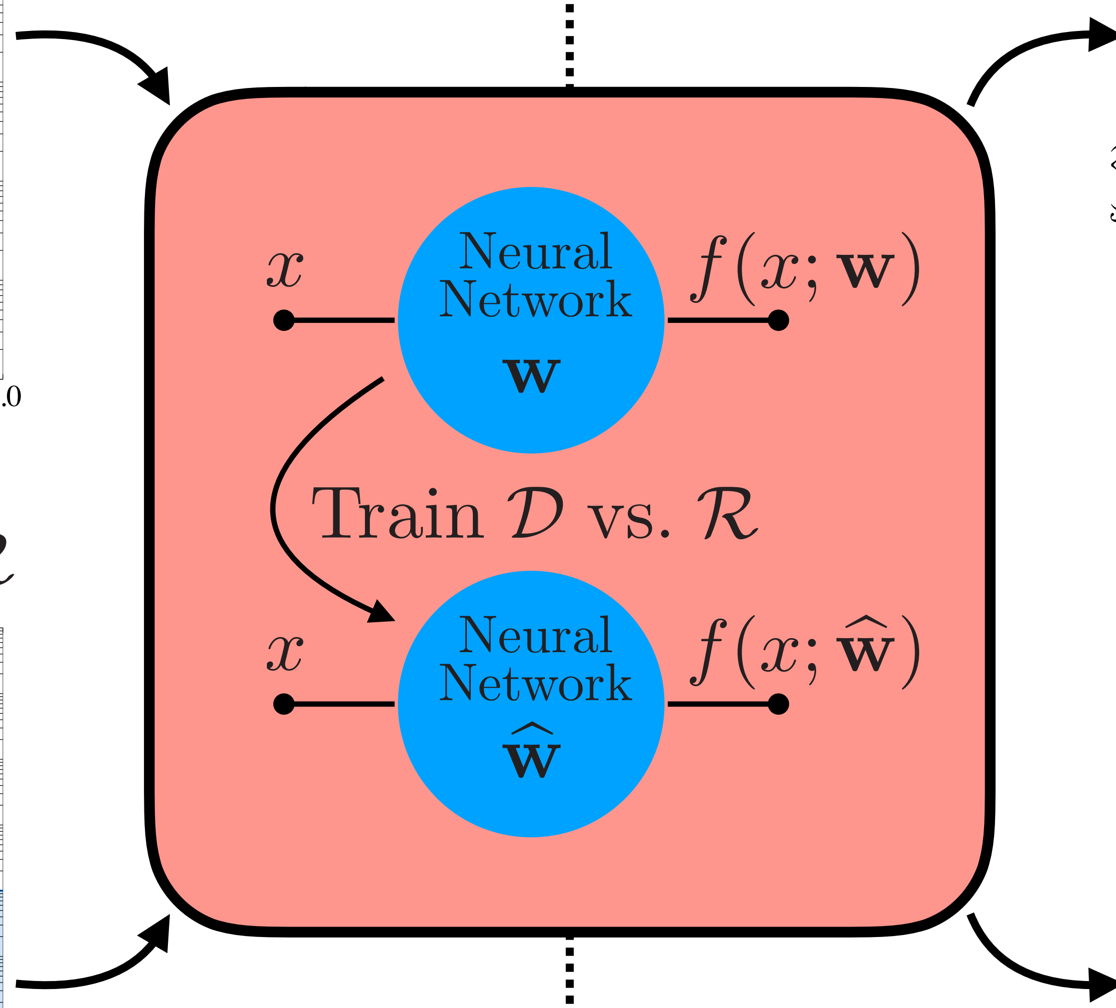
INPUT

Data sample $\mathcal{D}$



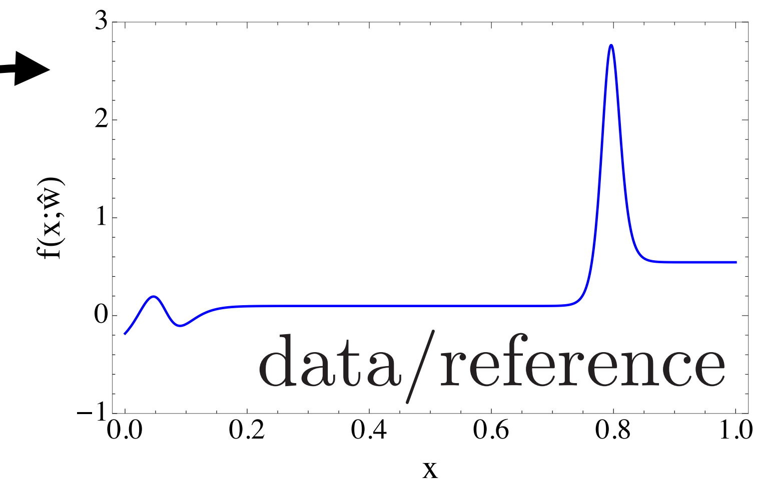
Reference sample $\mathcal{R}$





OUTPUT

Dist. log ratio



$$f(x;\widehat{\mathbf{w}}) \simeq \log\left[\frac{n(x|\mathrm{T})}{n(x|\mathrm{R})}\right]$$

Test statistic t computed on the data sample $\mathcal{D}$

$$t(\mathcal{D}) = -2\,\underset{\{\mathbf{w}\}}{\mathrm{Min}}\, L[f]$$

THE LOSS FUNCTION

$$n(x|\mathbf{w}) = n(x|\mathrm{R})\, e^{f(x;\mathbf{w})}$$ → NEURAL NETWORK

$$t(\mathcal{D}) = 2\log\left[\frac{e^{-N(\widehat{\mathbf{w}})}}{e^{-N(\mathrm{R})}}\prod_{x\in\mathcal{D}}\frac{n(x|\widehat{\mathbf{w}})}{n(x|\mathrm{R})}\right]$$

$$= -2\,\underset{\{\mathbf{w}\}}{\mathrm{Min}}\left[\frac{N(\mathrm{R})}{\mathcal{N}_{\mathcal{R}}}\sum_{x\in\mathcal{R}}\left(e^{f(x;\mathbf{w})}-1\right)-\sum_{x\in\mathcal{D}}f(x;\mathbf{w})\right]$$

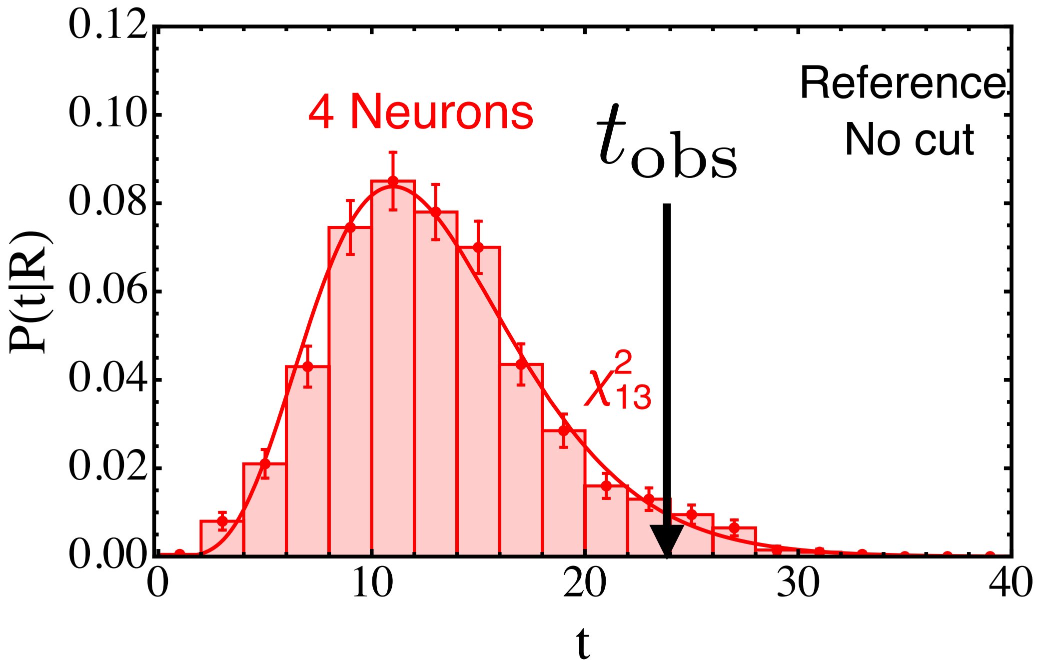
THE NETWORK IS DOING A MAXIMUM LIKELIHOOD FIT TO THE DATA AND COMPUTING THE “OPTIMAL” TEST STATISTIC AT THE SAME TIME

SUMMARY II

1. TRAIN THE NETWORK ON THE DATA
   - **INPUT**: ONE DATA SAMPLE AND ONE REFERENCE SAMPLE
   - **OUTPUT**: TEST STATISTIC ON THE DATA SAMPLE AND DISTRIBUTION LOG-RATIO
2. GENERATE TOY DATA SAMPLES THAT FOLLOW THE REFERENCE DISTRIBUTION AND TRAIN THE NETWORK AGAIN USING THEM AS DATA
   - **INPUT**: TOY DATA AND SAME REFERENCE SAMPLE AS ABOVE
   - **OUTPUT**: DISTRIBUTION OF THE TEST STATISTIC IN THE REFERENCE HYPOTHESIS

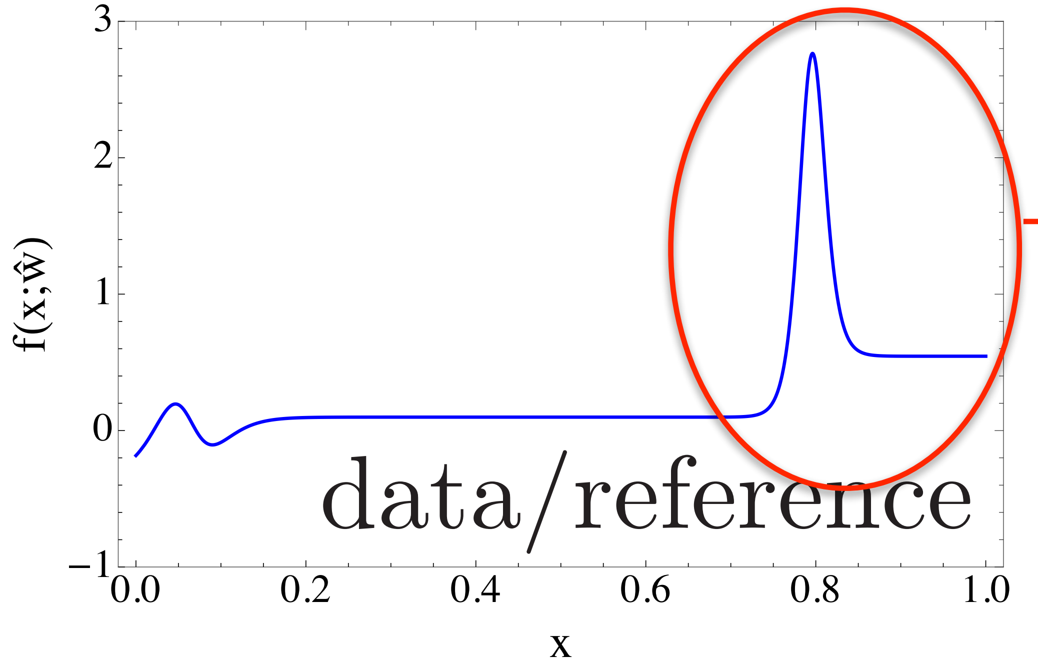
SUMMARY II

3.



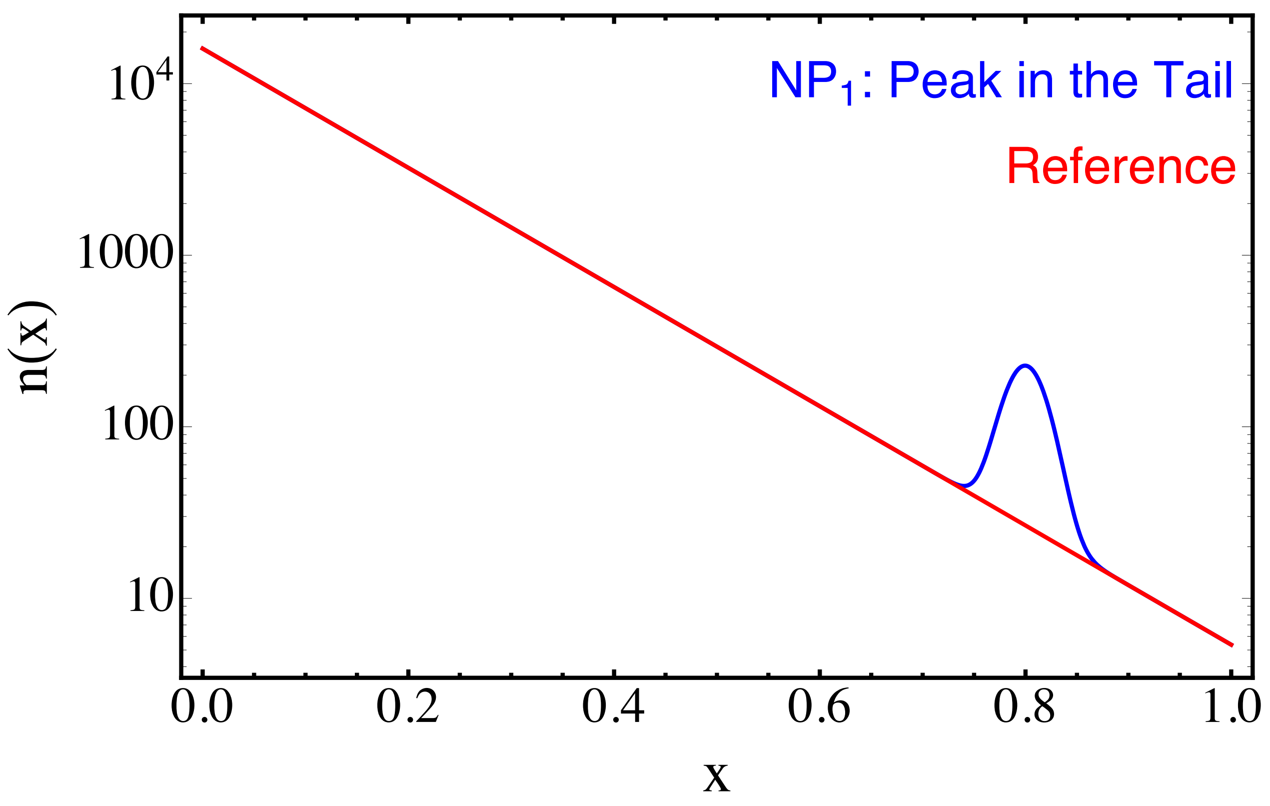
$$p_{\rm obs} = \int_{t_{\rm obs}}^{\infty} dt\, P(t|\mathrm{R})$$

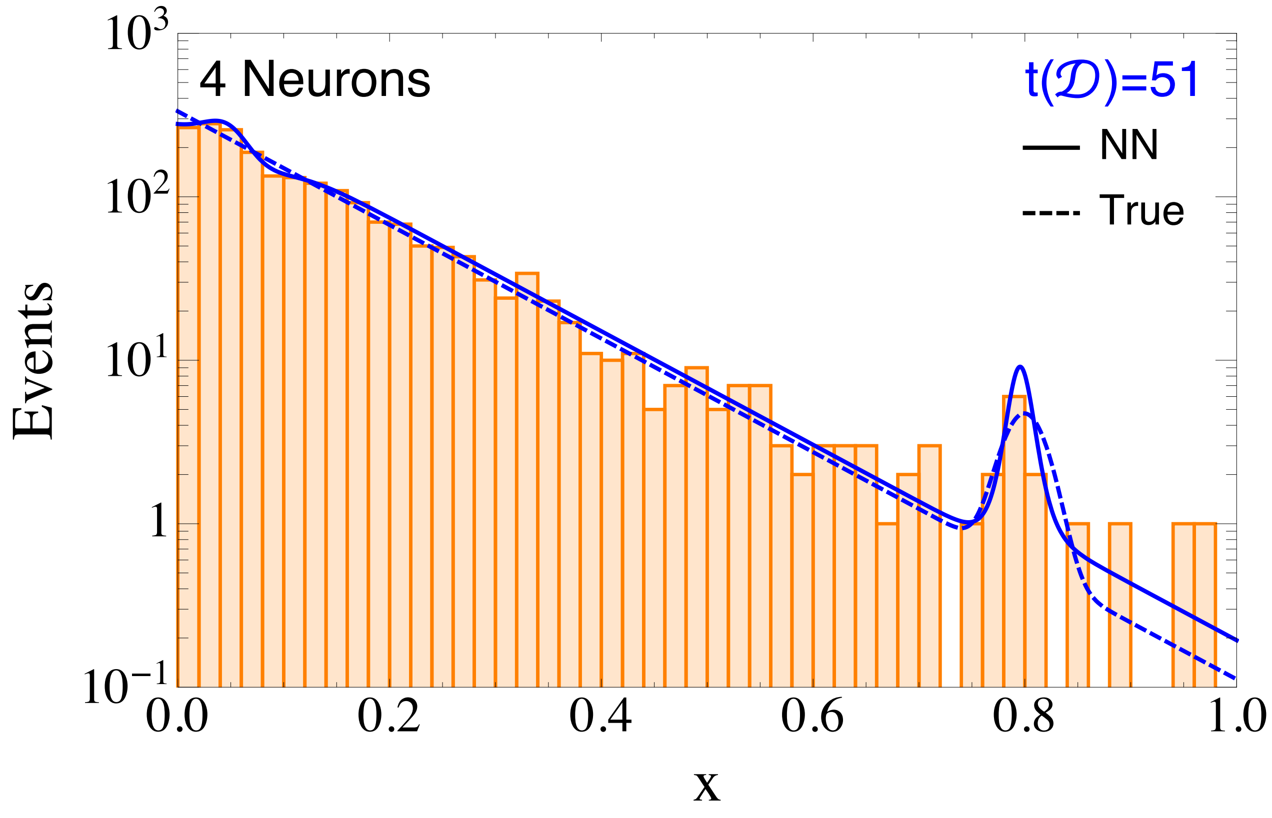
4.



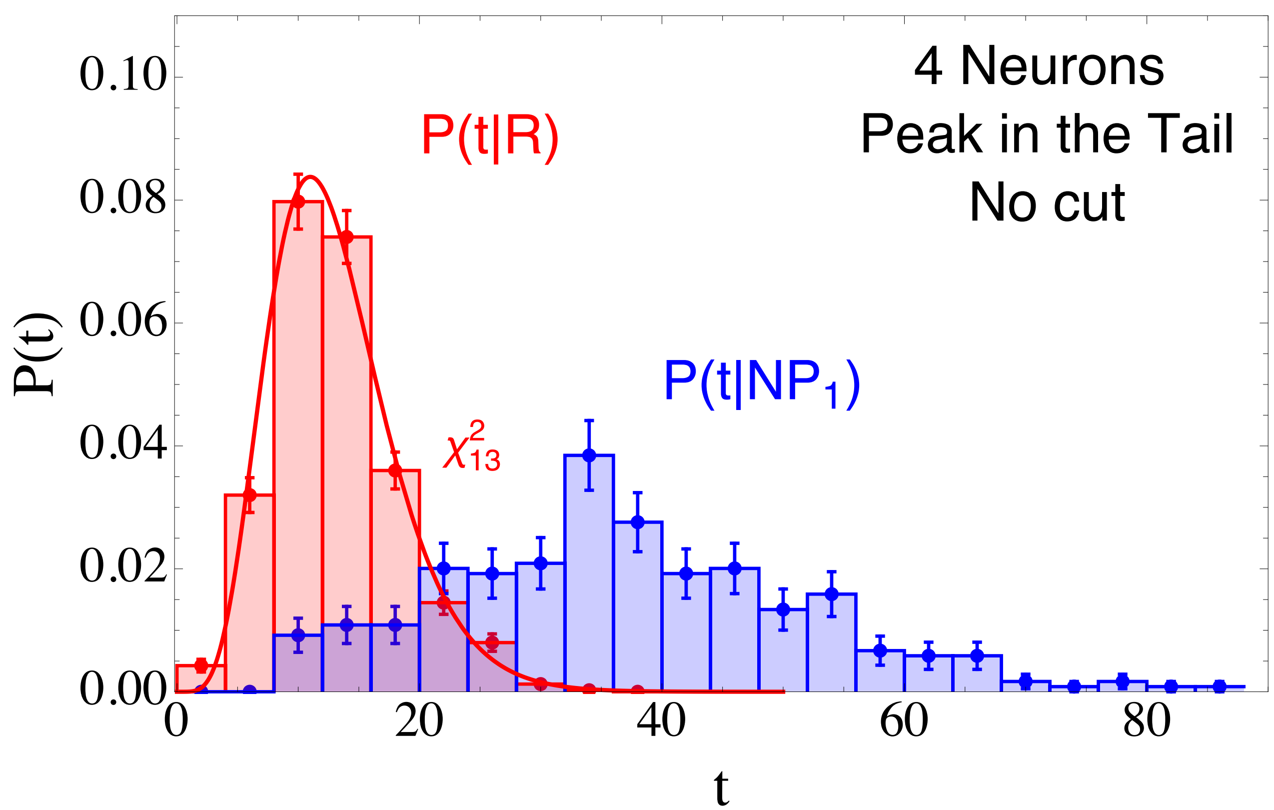
IDENTIFY AND CHARACTERIZE NEW PHYSICS

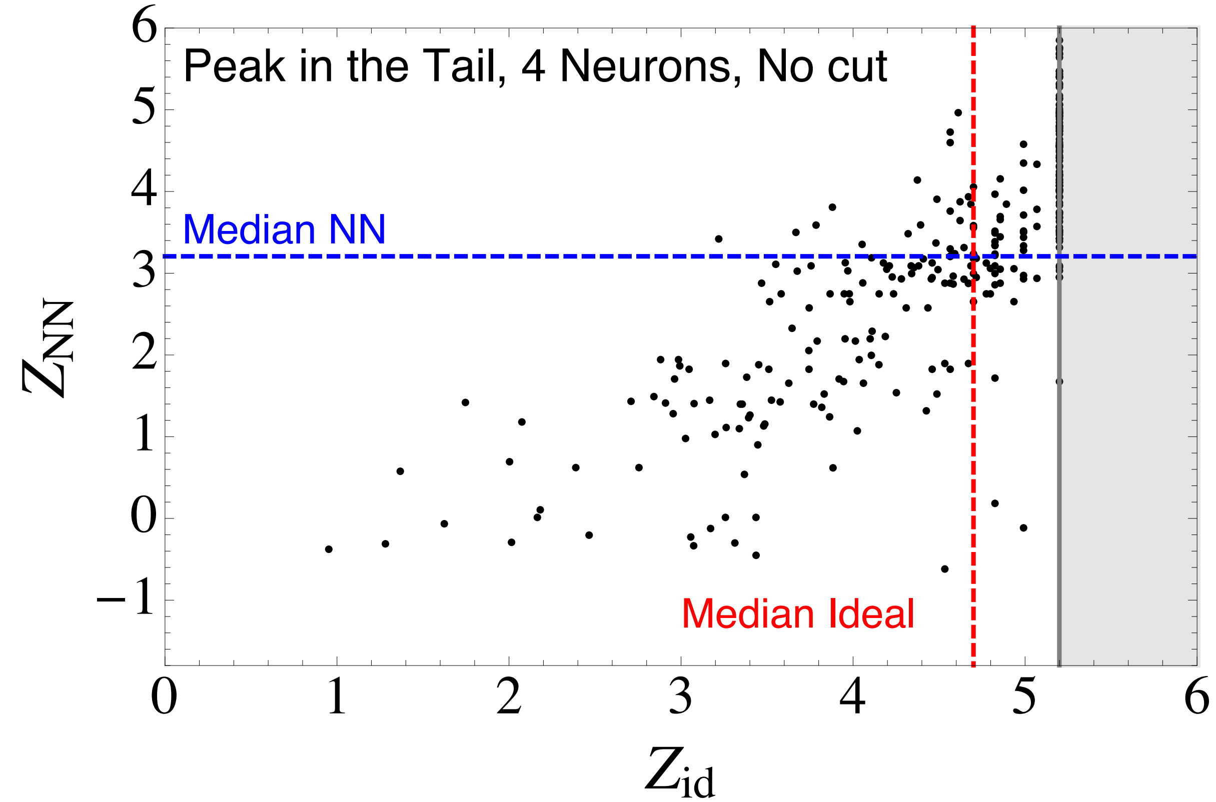
SENSITIVE TO NEW PHYSICS



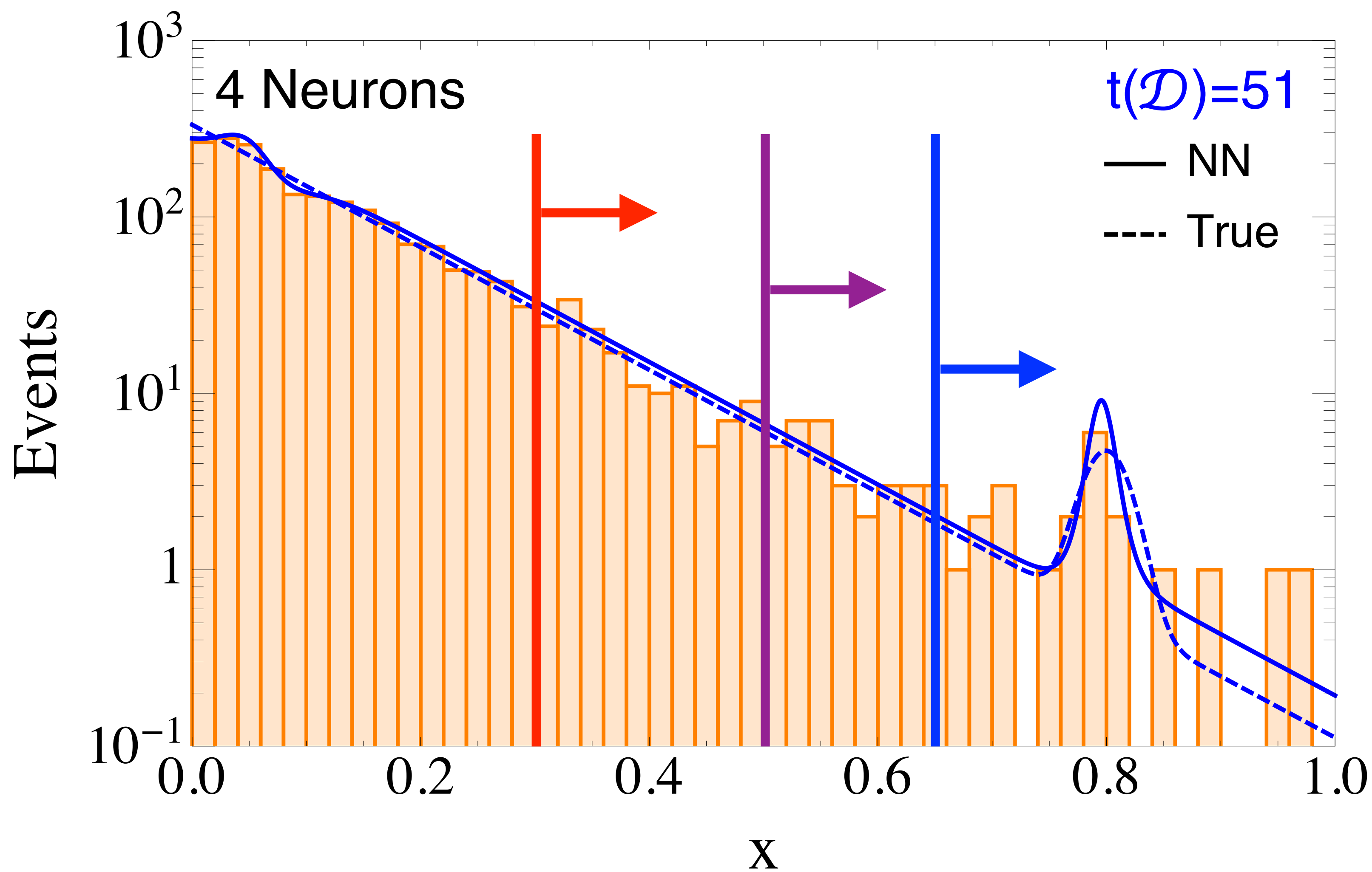


SENSITIVE TO NEW PHYSICS

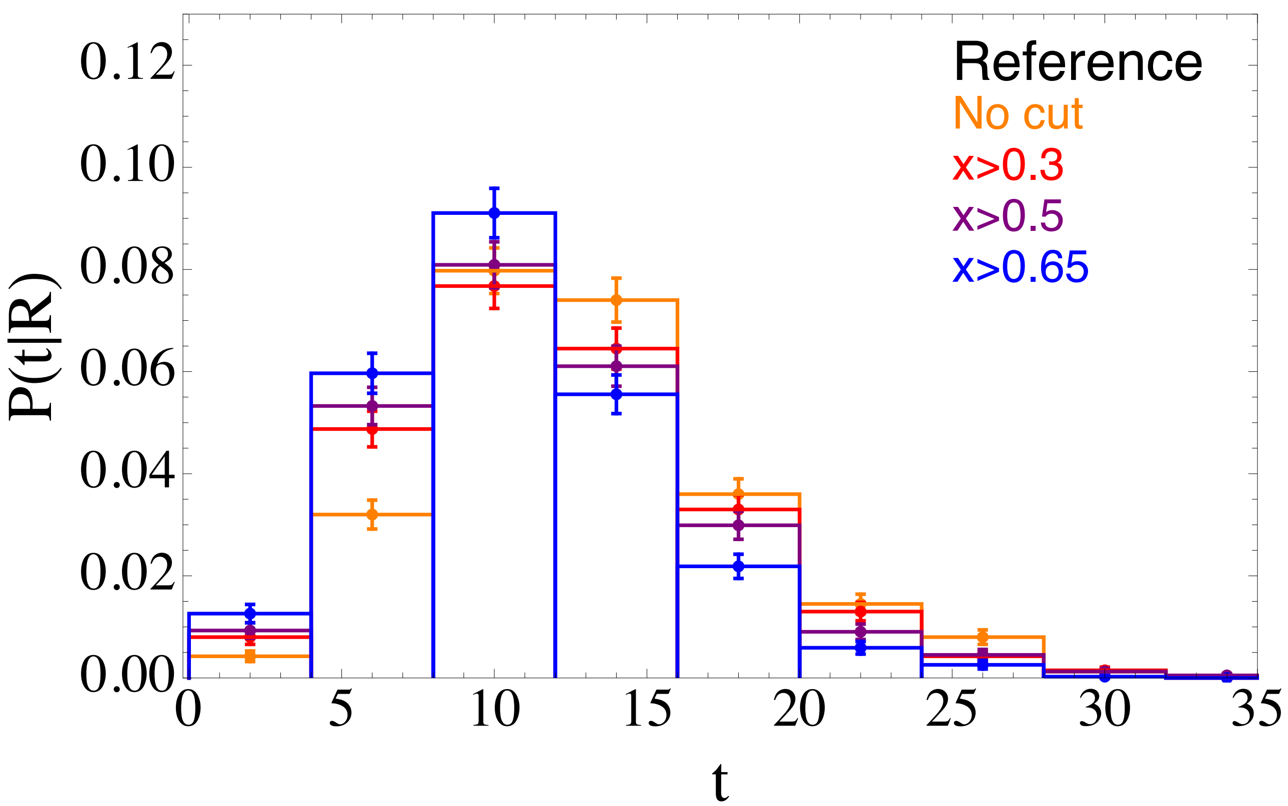


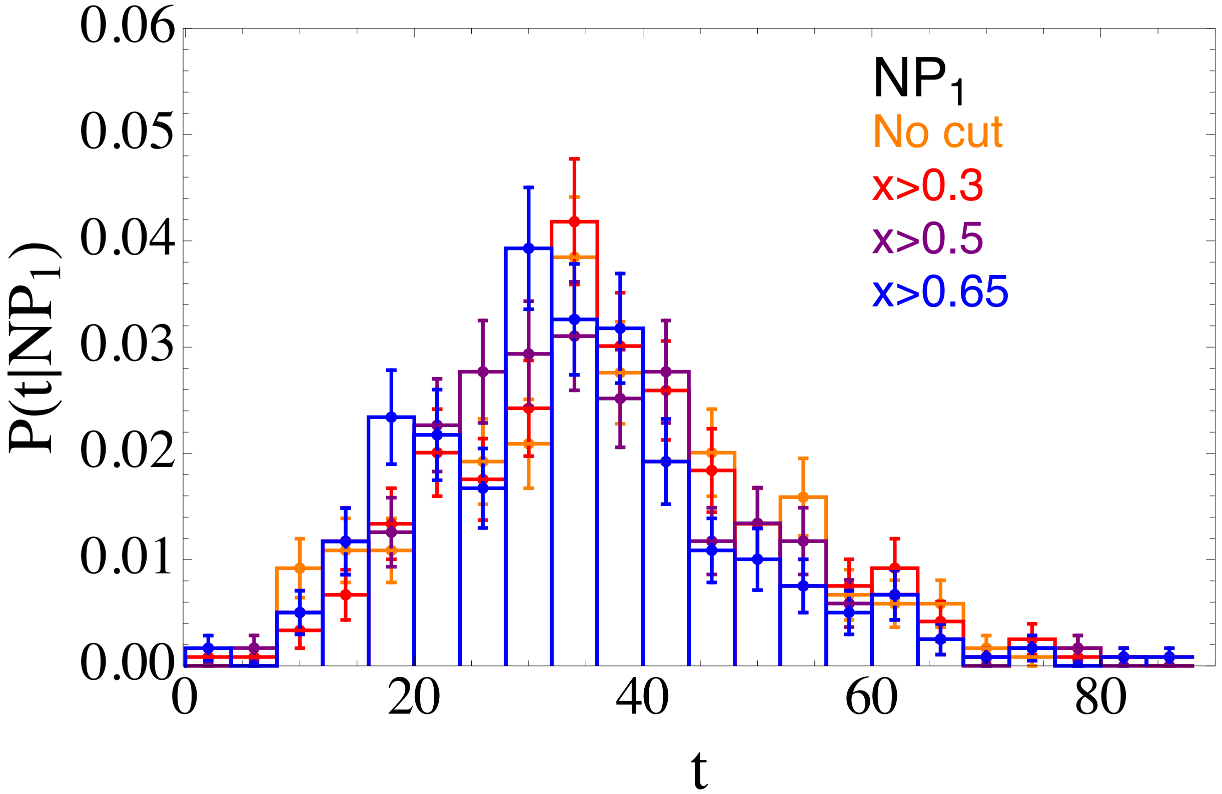


INSENSITIVE TO CUTS

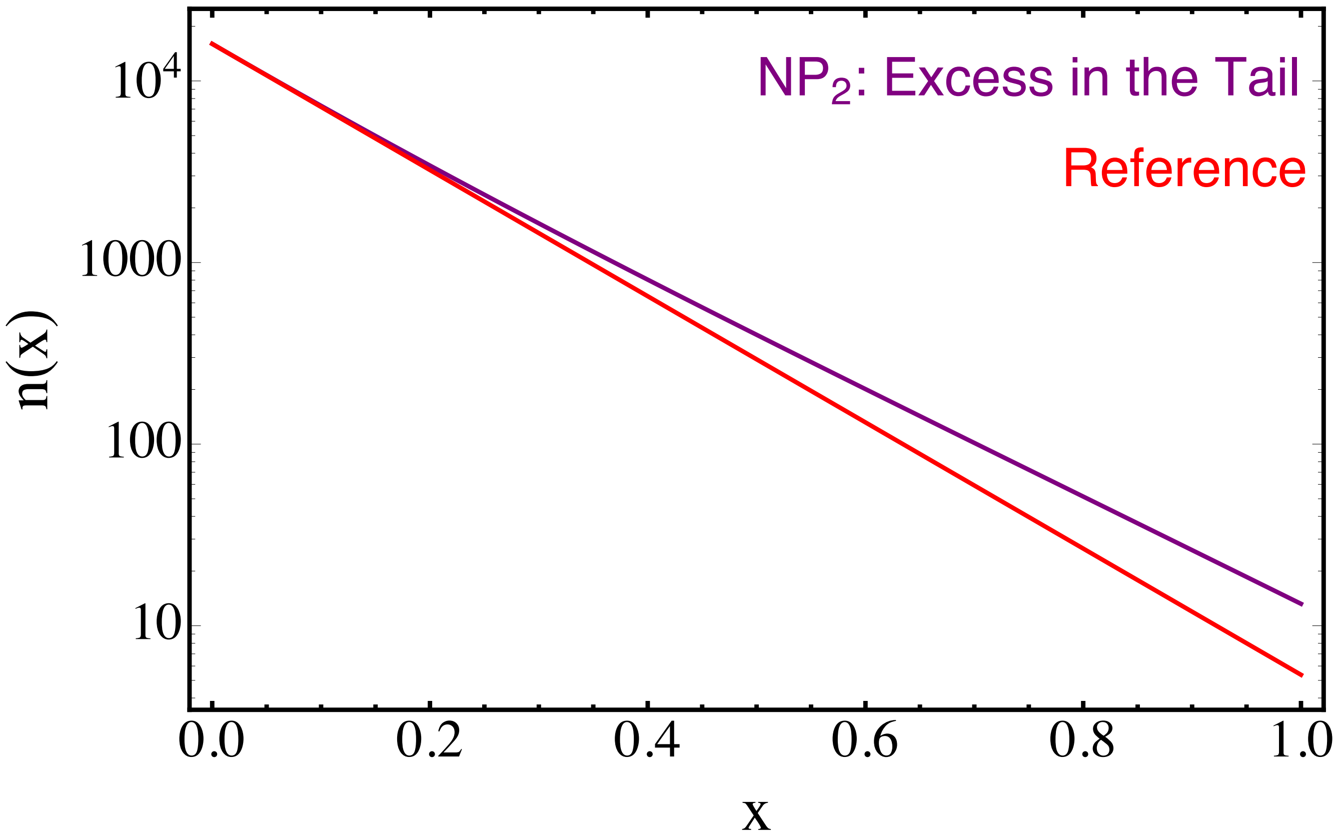


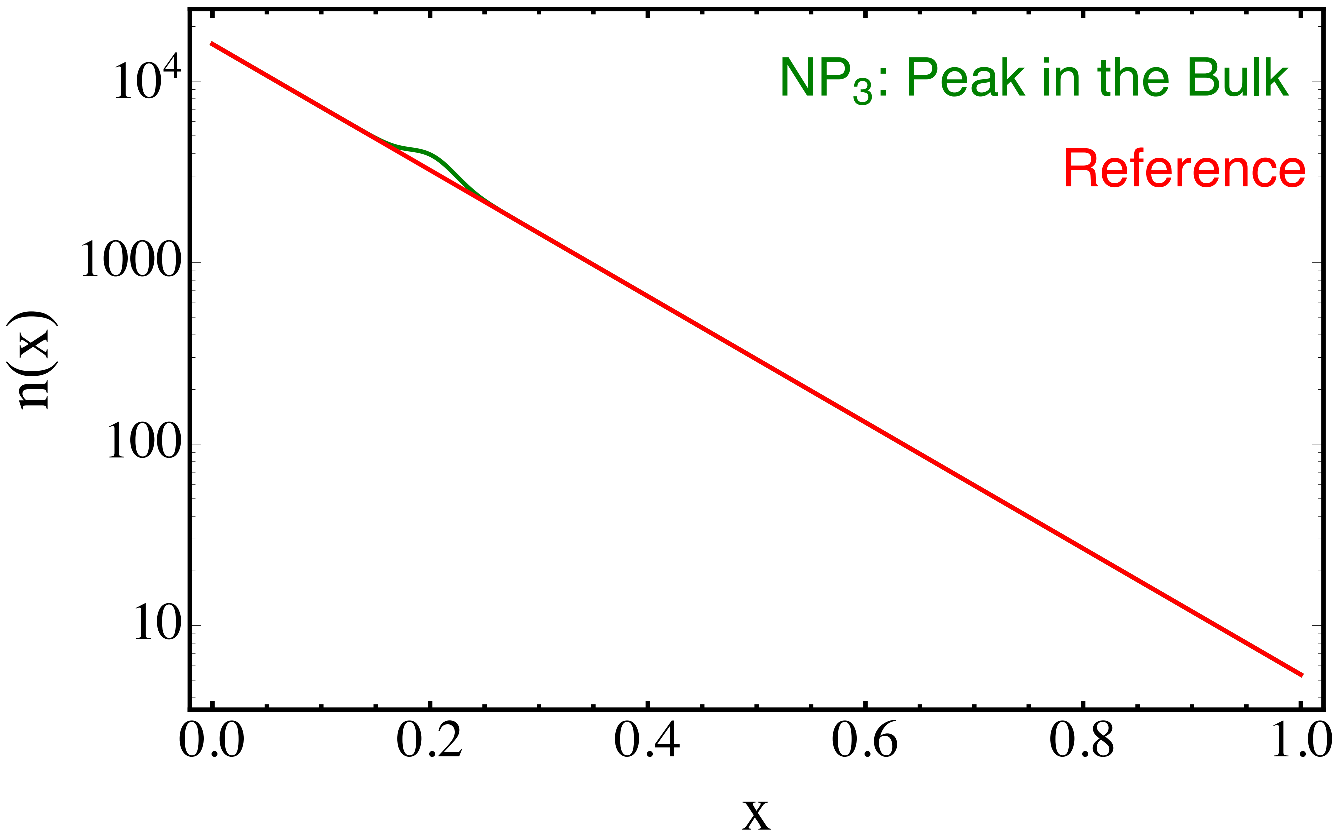
INSENSITIVE TO CUTS



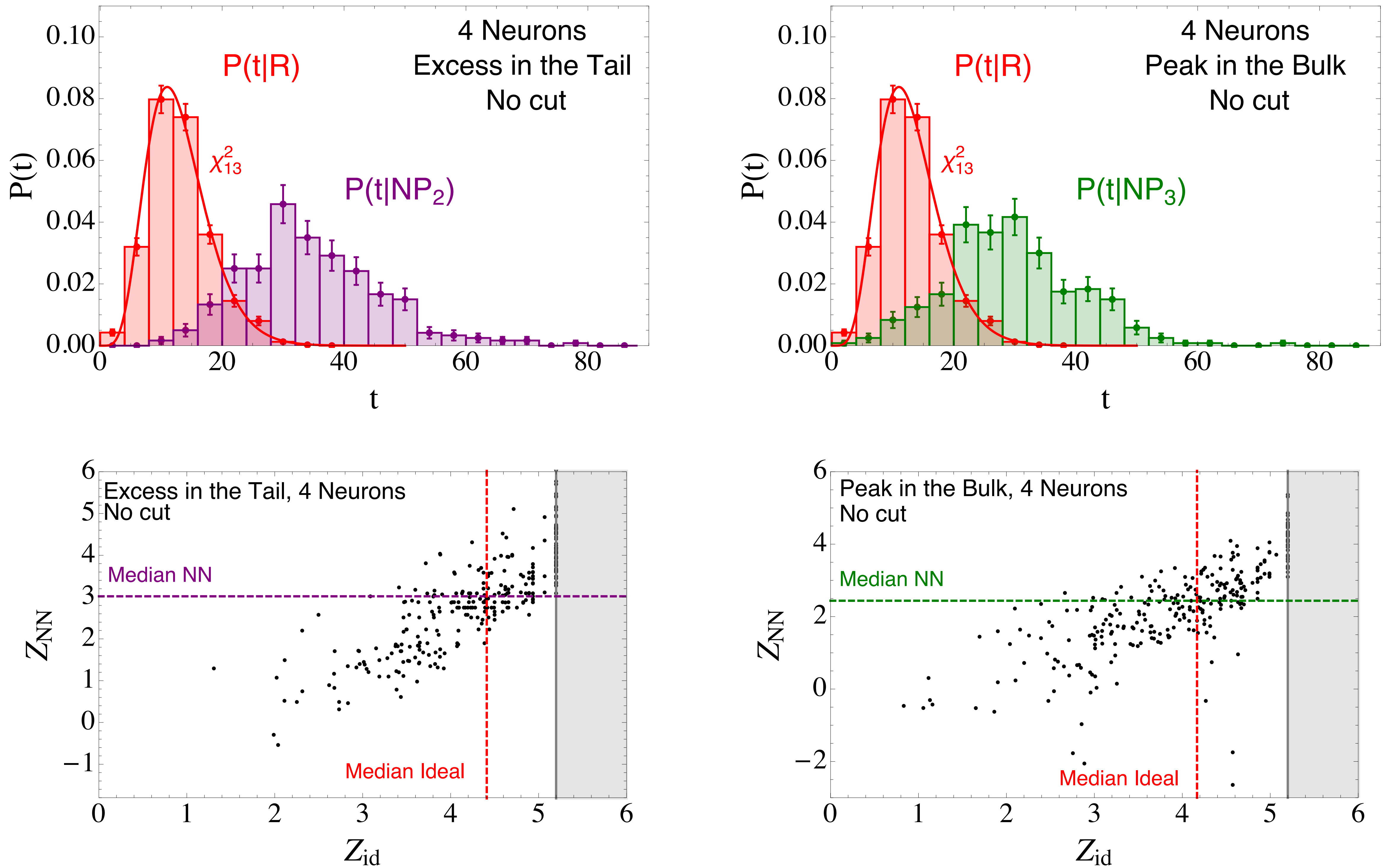


MODEL-INDEPENDENT

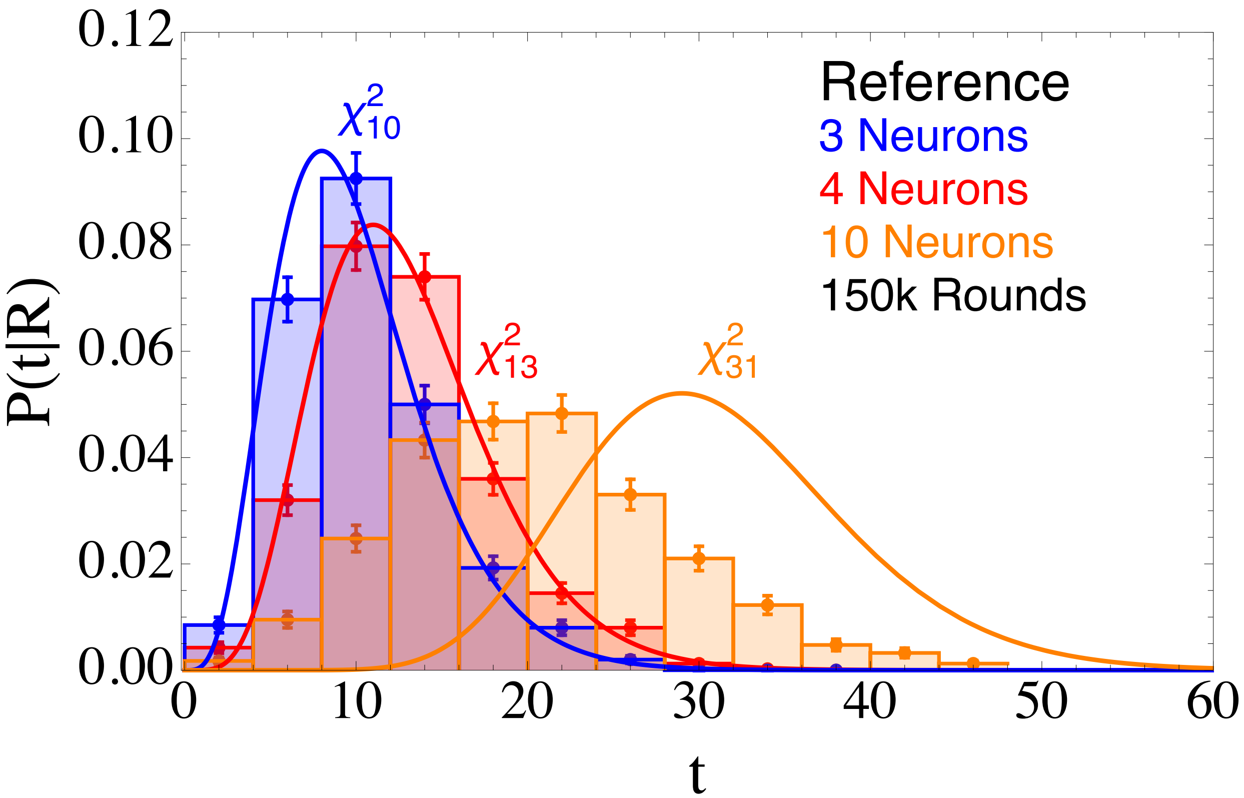


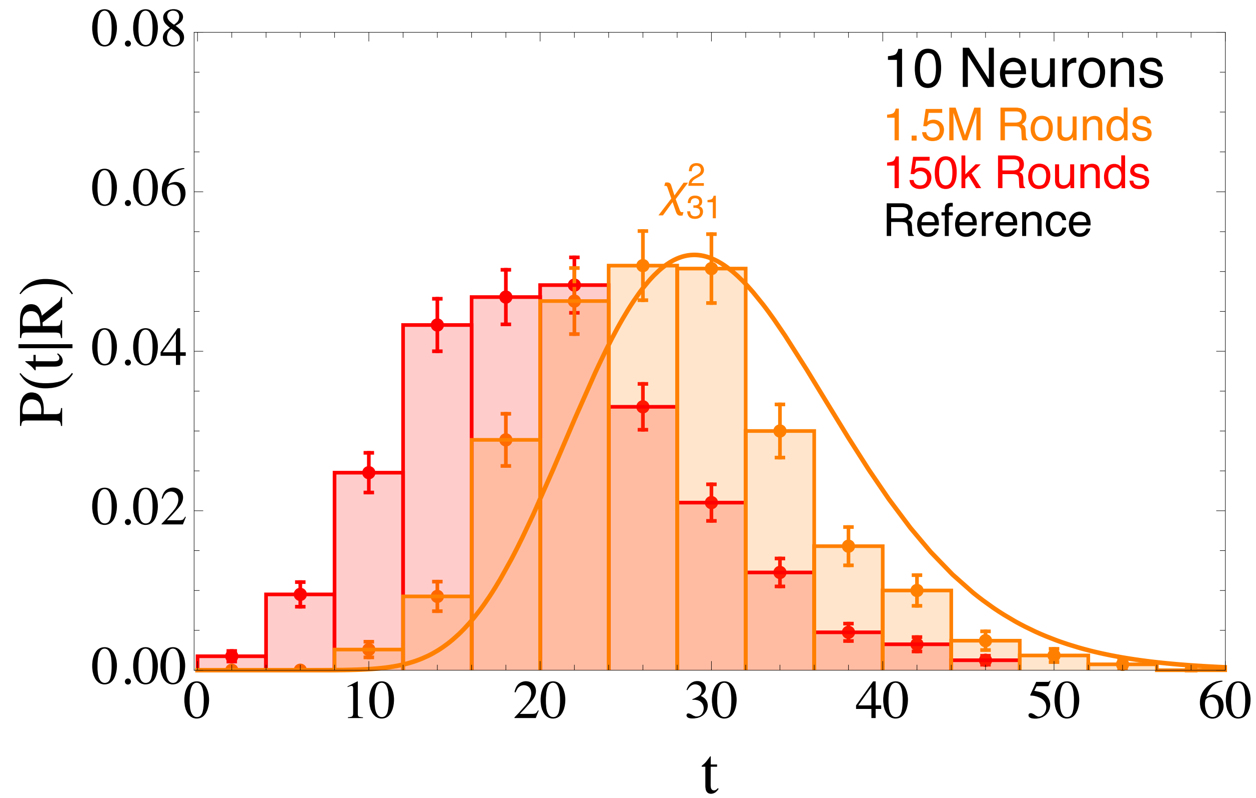


MODEL-INDEPENDENT



NETWORK ARCHITECTURE





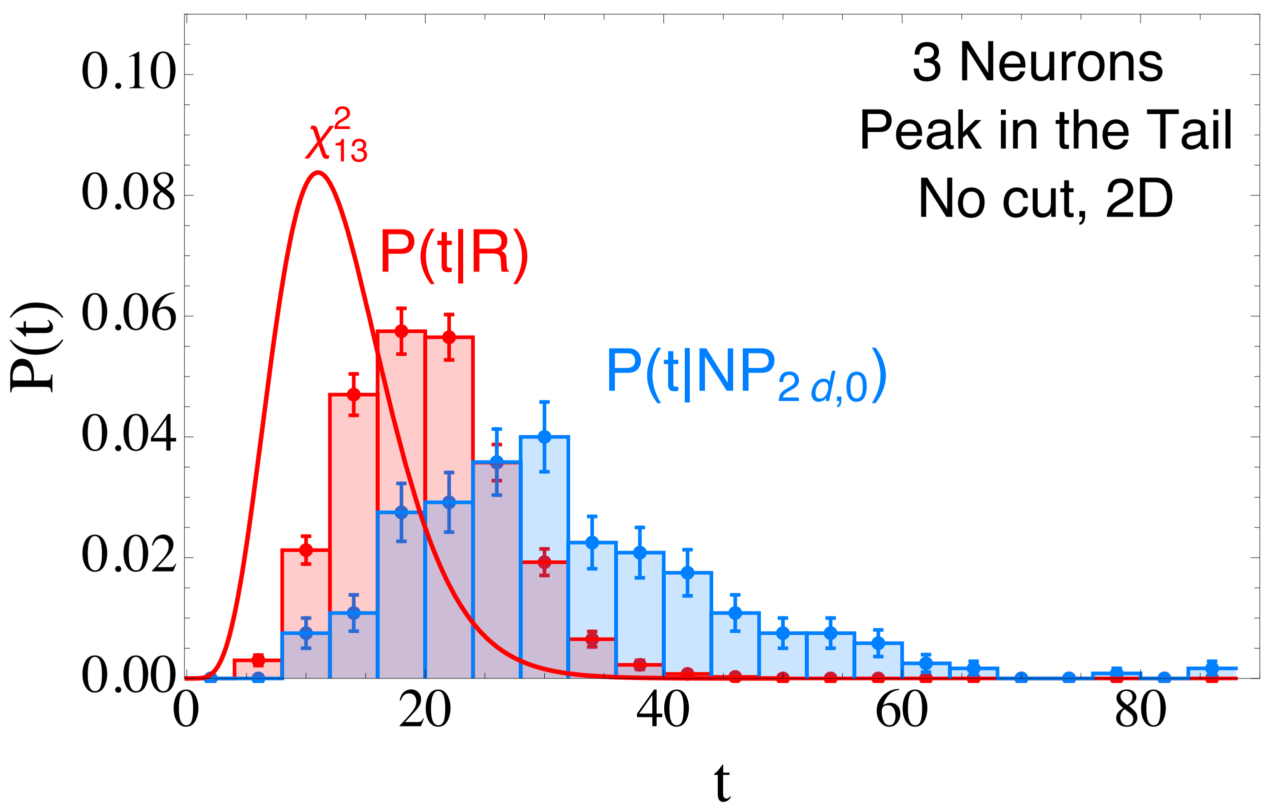
CONCLUSION AND OUTLOOK

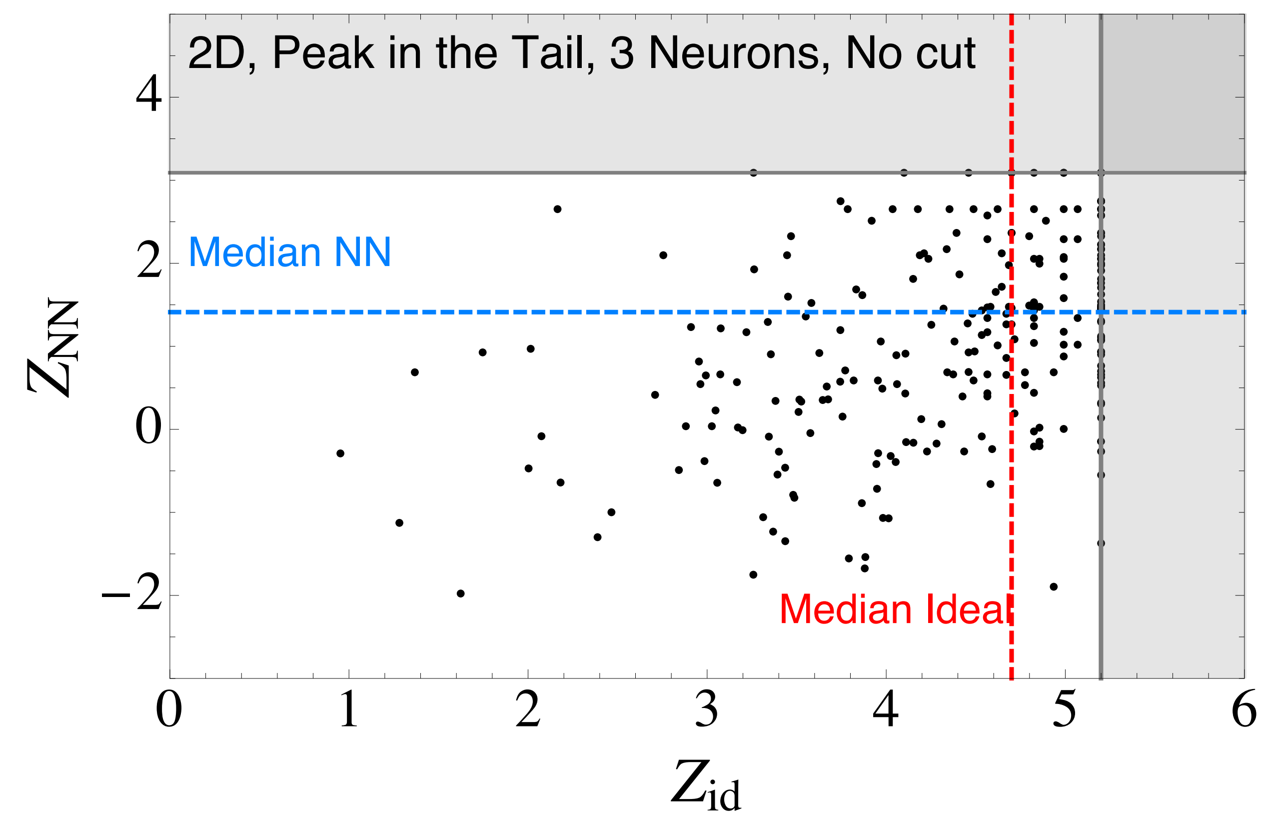
- TODAY IN FUNDAMENTAL PHYSICS WE HAVE LARGE, MULTIVARIATE, SM-LIKE DATASETS AND STRONG REASONS TO BELIEVE THAT THEY SHOULD NOT BE SM-LIKE
- OUR BEST GUESSES FOR NEW PHYSICS ARE NOT BEING DETECTED AND ANYTHING THAT HELPS US TO SEARCH WITHOUT ANY BIAS CAN BE USEFUL
- NEURAL NETWORKS ARE WIDELY USED TO APPROXIMATE PROBABILITY DISTRIBUTIONS AND ARE IDEAL CANDIDATES FOR THIS TYPE OF PROBLEM
- TODAY I HAVE DESCRIBED AN APPLICATION OF NEURAL NETWORKS, FOUNDED ON SOLID STATISTICAL PRINCIPLES, WHICH GOES IN THIS DIRECTION
  - ITS VIRTUES (SENSITIVITY TO NP, MODEL-INDEPENDENCE, INSENSITIVITY TO CUTS) HAVE BEEN TESTED ON SIMPLE 1D AND 2D EXAMPLES
  - MORE WORK IS NEEDED IN THE 2D AND HIGHER-DIMENSIONAL CASE

BACKUP

TWO DIMENSIONS

NP: x~EXPONENTIAL+PEAK y~UNIFORM
R: x~EXPONENTIAL y~UNIFORM





RECOVERS COMPARABLE SENSITIVITY TO 1D FOR x>0.3 OR DOUBLING THE EVENTS

AN INCOMPLETE NN CHART

Perceptron (P)

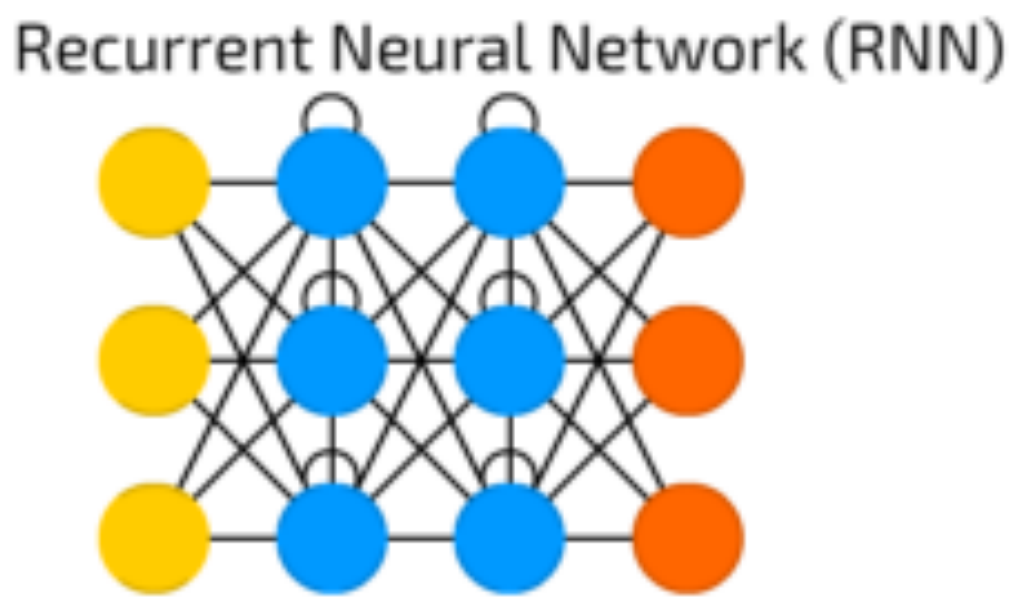
$$f(\vec{w} \cdot \vec{x} + b) \qquad f(x) = \begin{cases} 1 & x \geq 0 \\ 0 & x < 0 \end{cases}$$

Radial Basis Network (RBF)

$$\phi(\vec{x}) = \phi(|\vec{x}|)$$

$$\phi(\vec{w}'' \cdot \phi(\vec{w}' \cdot \vec{\phi}(\vec{w} \cdot \vec{x} + b) + b') + b'')$$

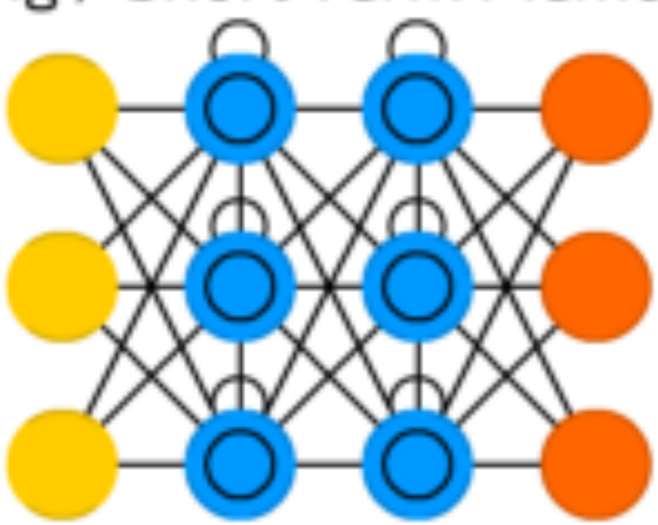
AN INCOMPLETE NN CHART



ITS OUTPUT AT TIME t DEPENDS
ON ITS PAST OUTPUT (t-1, t-2, …)

DESIGNED FOR APPLICATIONS
THAT NEED CONTEXT
(TEXT, SPEECH, SOUND RECOGNITION)

AN INCOMPLETE NN CHART



$$f_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f)$$
$$i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i)$$
$$o_t = \sigma_g(W_o x_t + U_o h_{t-1} + b_o)$$
$$c_t = f_t \circ c_{t-1} + i_t \circ \sigma_c(W_c x_t + U_c h_{t-1} + b_c)$$
$$h_t = o_t \circ \sigma_h(c_t)$$

$f_t =$ forget gate

$i_t =$ input gate

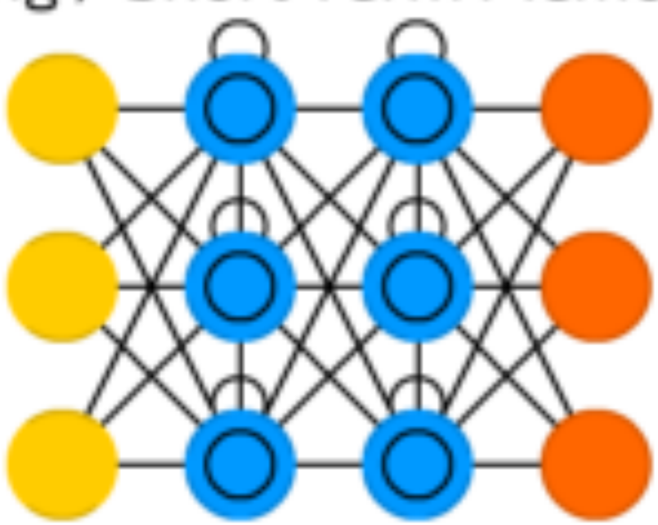
$o_t =$ output gate

$c_t =$ memory gate

$h_t =$ output

AN INCOMPLETE NN CHART

Long / Short Term Memory (LSTM)



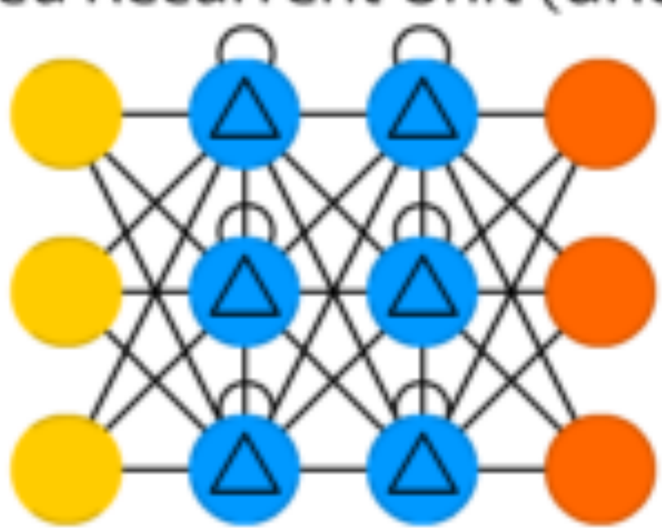
$$f_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f)$$
$$i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i)$$
$$o_t = \sigma_g(W_o x_t + U_o h_{t-1} + b_o)$$
$$c_t = f_t \circ c_{t-1} + i_t \circ \sigma_c(W_c x_t + U_c h_{t-1} + b_c)$$
$$h_t = o_t \circ \sigma_h(c_t)$$

SIMPLE RECURRENT

$$o_t = \vec{1}, i_t = \vec{1}, f_t = \vec{0}$$

(IT DOESN'T FORGET)

AN INCOMPLETE NN CHART



$$z_t = \sigma_g(W_z x_t + U_z h_{t-1} + b_z)$$

$$r_t = \sigma_g(W_r x_t + U_r h_{t-1} + b_r)$$

$$h_t = (1 - z_t) \circ h_{t-1} + z_t \circ \sigma_h(W_h x_t + U_h(r_t \circ h_{t-1}) + b_h)$$

$x_t =$ input vector

$z_t =$ update gate

$r_t =$ reset gate

$h_t =$ output

Weak(er) Supervision

Knowns and Unknowns in Learning from Data

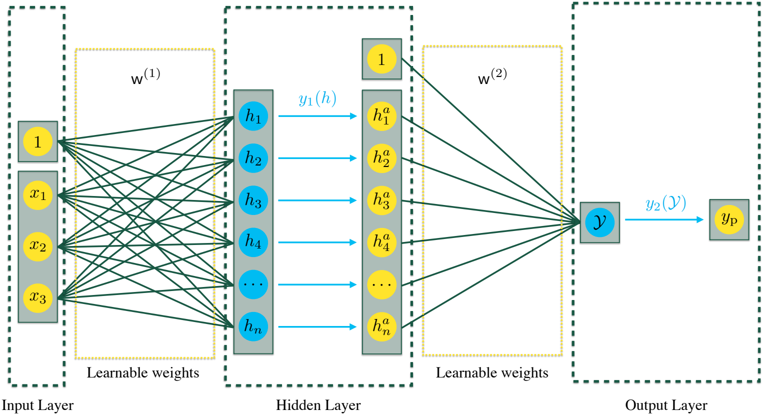
Marat Freytsis

U. of Oregon ⟶ Tel Aviv/IAS

Beyond SM: Where do we go from here? — GGI, September 19, 2018

Tim Cohen, MF, Bryan Ostdiek
JHEP 1802, 034 [arXiv:1706.09451] + ongoing work

Traditional feed-forward NN classification



$$\ell_{\mathrm{BCE}}(\{y_t\},\{y_p\}) = -\sum_i \left(y_{t,i}\log y_{p,i} + (1-y_{t,i})\log(1-y_{p,i})\right)$$

requires event-by-event labels for (simulated) training sample — can we relax this?

Why bother?

In theory there's no difference between theory and practice. In practice there is. – Yogi Berra

the data is reality
we can only produce approximations

not always good ones —

ubiquitous situation in jet physics

ideally

- avoid spurious features
- exploit correlations where present
- learn features we haven't thought of

Why bother?

> *In theory there's no difference between theory and practice. In practice there is. – ~~Yogi Berra~~*

the data is reality
we can only produce approximations

not always good ones —

ubiquitous situation in jet physics

ideally

- avoid spurious features
- exploit correlations where present
- learn features we haven't thought of

Why bother?

In theory there's no difference between theory and practice. In practice there is. – ~~Yogi Berra~~

the data is reality

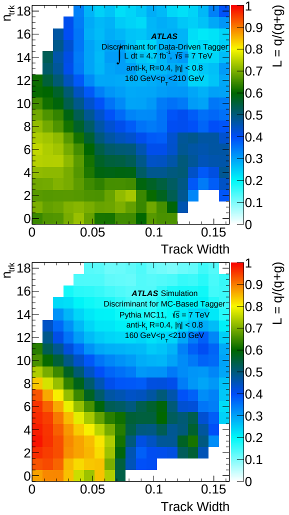
we can only produce approximations

not always good ones —

ubiquitous situation in jet physics

ideally

- avoid spurious features
- exploit correlations where present
- learn features we haven't thought of



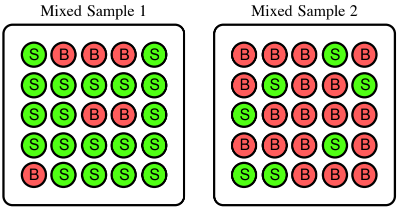
CERN-PH-EP-2014-058

Plan

- Simulation and its discontents
- Letting data drive with weak supervision
- Other features and approaches

Supervising with data

real data: can't assign truth labels, can't create pure samples
what to do? use mixed training events directly!



[arXiv:1708.02949]

only thing known is fractional composition
requires more care than fully curated training data:

- all training sets sample identical distributions
- multiple training sets with different mixtures f_S required

fractional labels in physics *are* observables: integrated cross sections

Loss functions

how to identify signal events?

1. direct attack (learning with label proportions):

$$\ell_{\text{LLP}}(\{f_t\},\{y_p\}) = |\langle f_{t,i}\rangle - \langle y_{p,i}\rangle|$$

Dery, Nachman, Rubbo, Schwartzman [arXiv:1702.00414]

requires new loss function and training algorithm

2. clever trick (classification without labels):

$$\ell_{\text{CWoLa}}(\{f_t\},\{y_p\}) = \sum_i |f_{t,i} - y_{p,i}|$$

Metodiev, Nachman, Thaler [arXiv:1708.02949]

or your fully-supervised loss function of choice

Both of these have antecedents in the ML literature

Classification without labels

why does the second version work at all? [arXiv:1708.02949]

Theorem

Given mixed samples M_1 and M_2 defined in terms of pure samples S and B with signal fractions $f_1 > f_2$, an optimal classifier trained to distinguish M_1 from M_2 is also optimal for distinguishing S from B.

Proof.

The optimal classifier to distinguish examples drawn from p_{M_1} and p_{M_2} is the likelihood ratio $L_{M_1/M_2}(\mathbf{x}) = p_{M_1}(\mathbf{x})/p_{M_2}(\mathbf{x})$. Similarly, the optimal classifier to distinguish examples drawn from p_S and p_B is the likelihood ratio $L_{S/B}(\mathbf{x}) = p_S(\mathbf{x})/p_B(\mathbf{x})$. Where p_B has support, we can relate these two likelihood ratios algebraically:

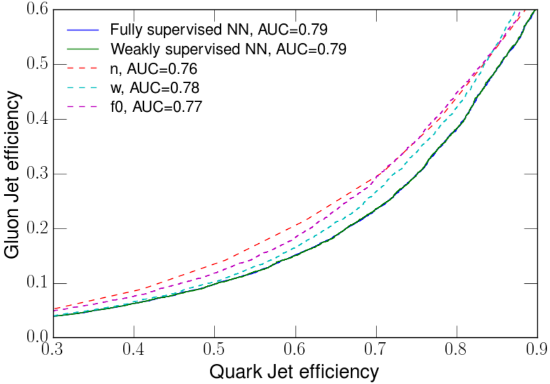
$$L_{M_1/M_2} = \frac{p_{M_1}}{p_{M_2}} = \frac{f_1 p_S + (1-f_1)p_B}{f_2 p_S + (1-f_2)p_B} = \frac{f_1 L_{S/B} + (1-f_1)}{f_2 L_{S/B} + (1-f_2)},$$

which is a monotonically increasing rescaling of the likelihood $L_{S/B}$ as long as $f_1 > f_2$, since $\partial_{L_{S/B}} L_{M_1/M_2} = (f_1 - f_2)/(f_2 L_{S/B} - f_2 + 1)^2 > 0$. If $f_1 < f_2$, then one obtains the reversed classifier. Therefore, $L_{S/B}$ and L_{M_1/M_2} define the same classifier. □

still need to know $f_{1,2}$ if you need to know efficiency/rate

Performance in simulation

LLP

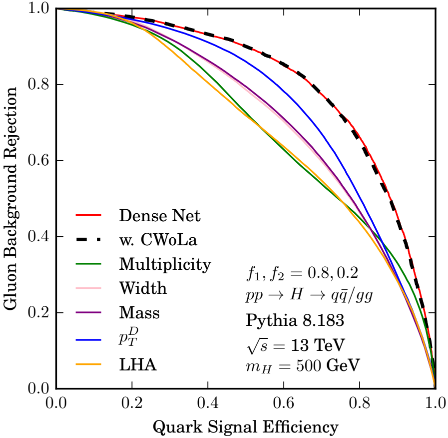


[arXiv:1702.00414]

⚠ full and weak NNs have different architectures here
interpret with caution!

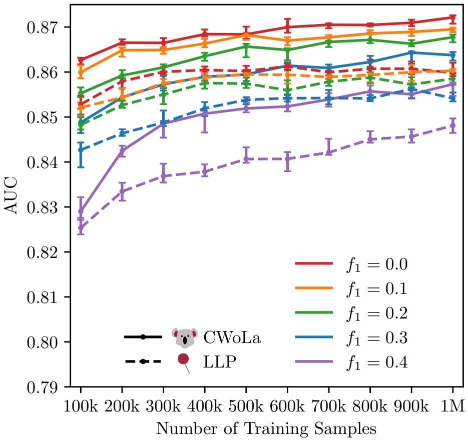
Performance in simulation

CWoLa



[arXiv:1708.02949]

Performance in simulation
Jet images



[arXiv:1801.10158]

also works directly with sparsely populated event-by-event features

Plan

- Simulation and its discontents
- Letting data drive with weak supervision
- Other features and approaches

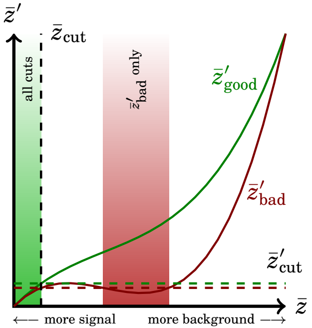
Label insensitivity

easier to understand effect of wrong fractions with LLP

$$h_A = f_A h_1 + (1 - f_A) h_0 \qquad h_B = f_B h_1 + (1 - f_B) h_0 \quad \Longrightarrow \quad h_0 = \frac{f_A h_B - f_B h_A}{f_A - f_B} \qquad h_1 = \frac{(1-f_B) h_A - (1-f_A) h_B}{f_A - f_B}$$

optimal classifier $\bar{z} = \frac{h_1}{h_0 + h_1}$ mis-reconstructed as $\bar{z}'$ if $f_A \to f_A + \delta$

know analytic form of $\bar{z}'$



Cohen, MF, Ostdiek [arXiv:1706.09451]

A BSM example

Technical details

$$pp \to \tilde{g}\tilde{g} \text{ vs. } (Z \to \nu\bar{\nu}) + nj, \quad m_{\tilde{g}} = 2 \text{ TeV}$$

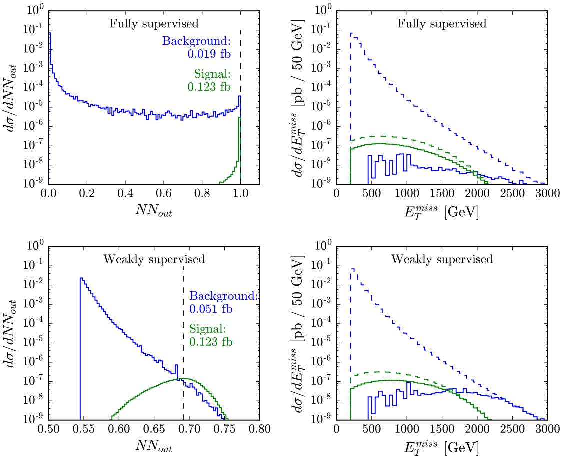
simulate in MadGraph5 + Pythia6 + Delphes3
train on p_T of jets
Keras with TensorFlow backend

Loss function	BCE
n_{input}	11
Hidden Nodes	30
Activation	Sigmoid
Initialization	Normal
Learning algorithm	SGD
Learning rate	0.01
Batch size	64
Epochs	20

A BSM example

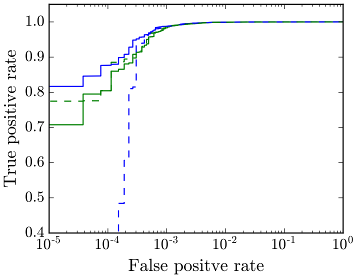
Network performance

Network	AUC	Signal efficiency
Full	0.99992393(31)	0.999373(17)
Weak	0.9998978(35)	0.999286(30)

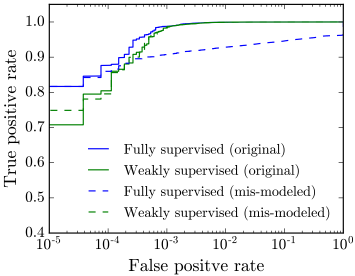


A BSM example

Impact of mismodelling



randomly swap 15% of each class



swap the 10% (15%) most signal-like (background-like)

Open questions, concrete & speculative

- performance for multi-component classification?
  - does CWoLa even have a multi-component generalization?
- how do the optimality arguments change at finite statistics?
- can we propagate input uncertainties through the network?
  - would this be useful?
- can we invert any of this to see what our models get wrong
- can we go even weaker?
  - *e.g.*, Hopfield networks, Boltzmann machines, etc.
    - can solve certain classification tasks unsupervised
    - some use in astrophysics, nearly no collider proposals to date
  - unsupervised anomaly detection already demonstrated
    - CWoLa [arXiv:1805.02664]
    - auto-encoders (coming up next…)
- …

RUTGERS
THE STATE UNIVERSITY
OF NEW JERSEY

Searching for New Physics with Deep Autoencoders

Yuichiro Nakai (Rutgers)

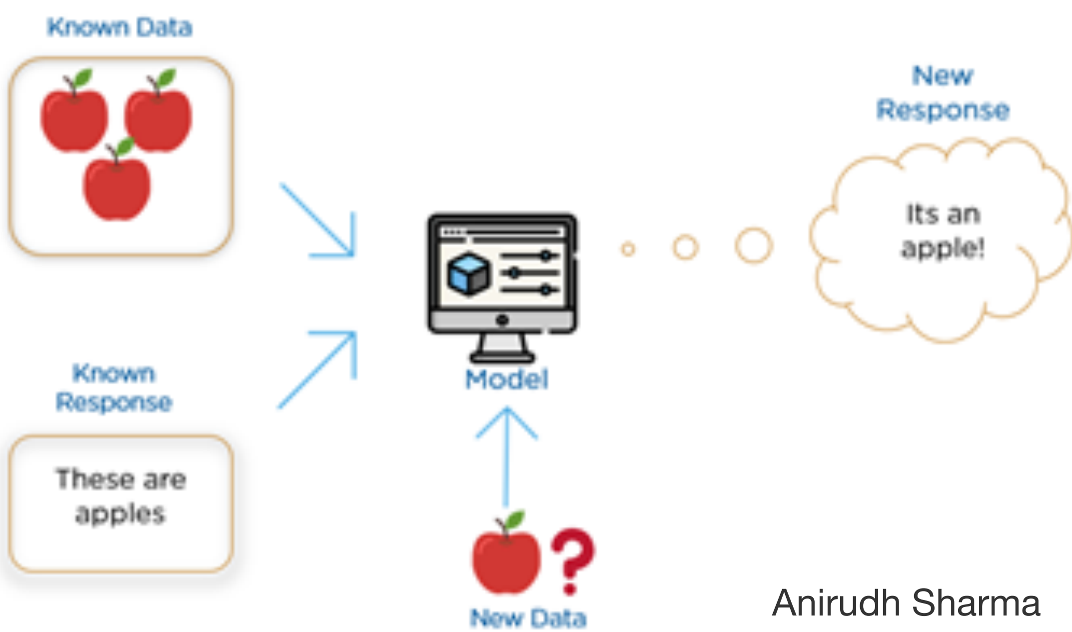
Based on M. Farina, YN and D. Shih, arXiv:1808.08992 [hep-ph].

Supervised or Unsupervised

Machine learning algorithms can be classified into:

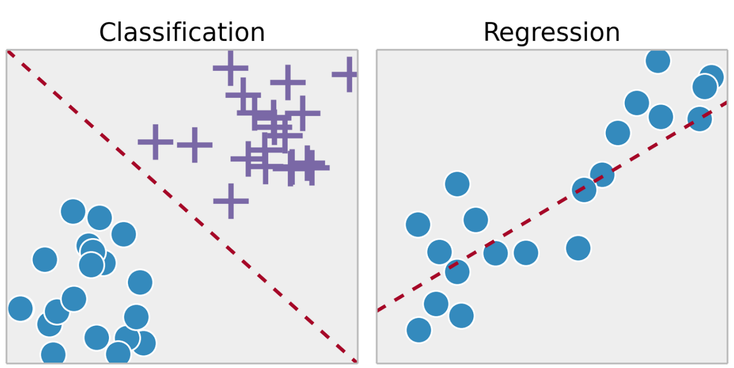
Supervised learning

Learn from labeled data



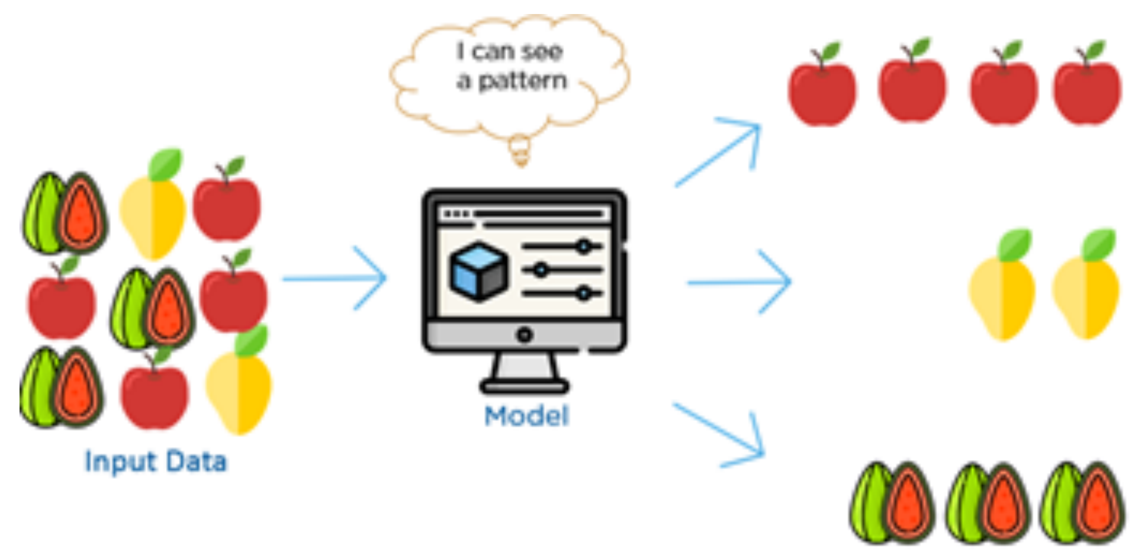
Anirudh Sharma

Applications)



Unsupervised learning

Learn from unlabeled data



The system looks for patterns and extracts features in data.

Applications) Clustering

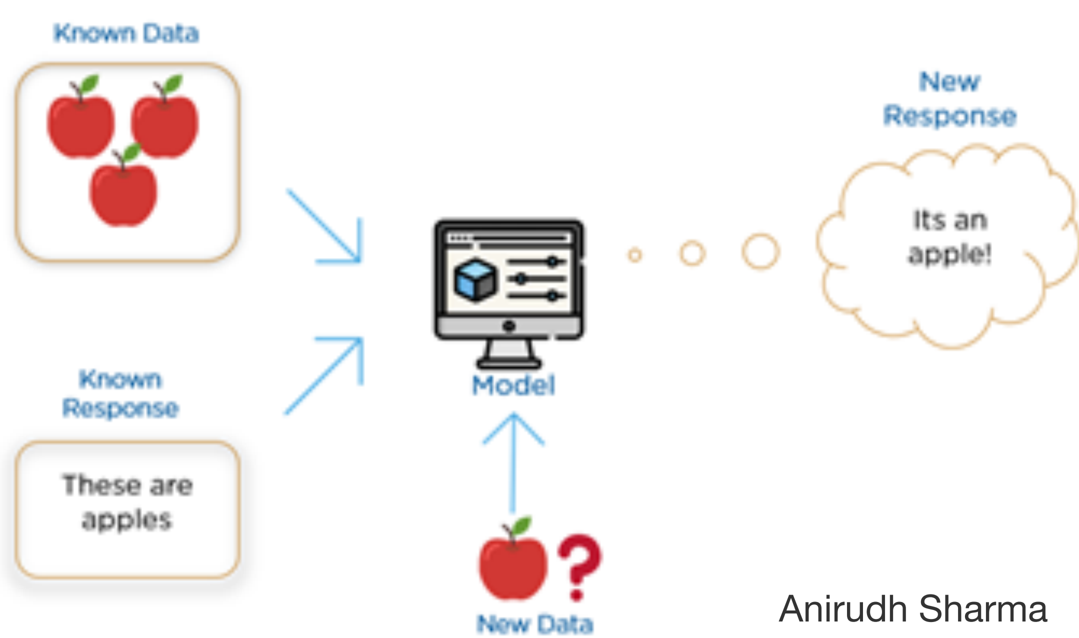
Anomaly detection

Supervised or Unsupervised

Machine learning algorithms can be classified into:

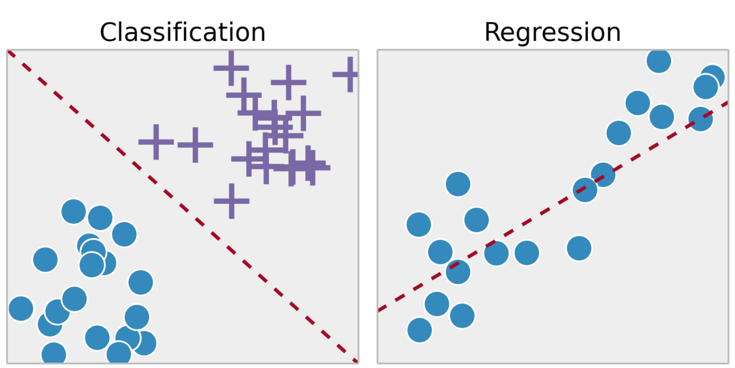
Supervised learning

Learn from labeled data



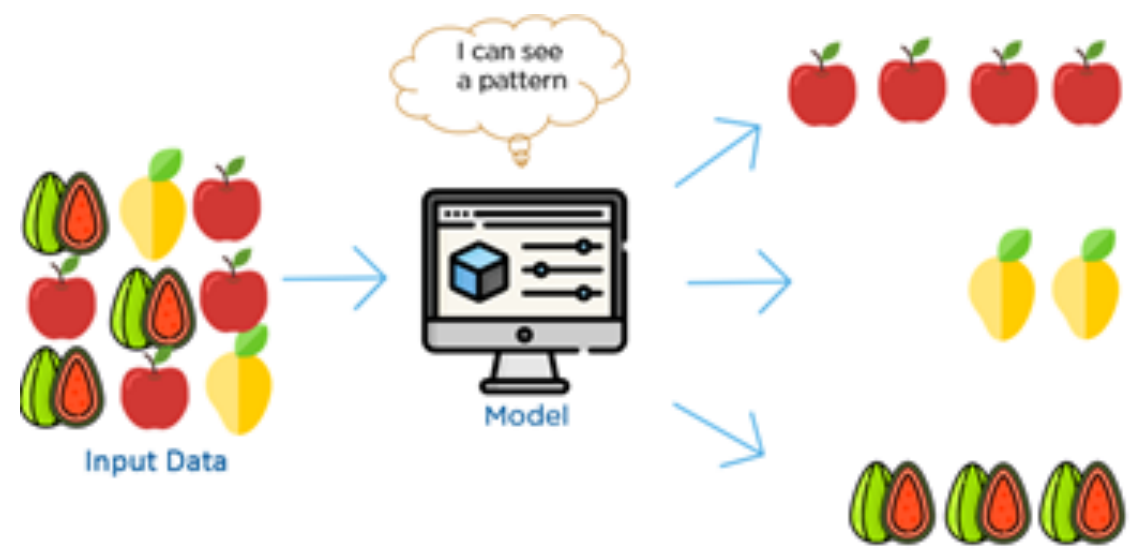
Anirudh Sharma

Applications)



Unsupervised learning

Learn from unlabeled data



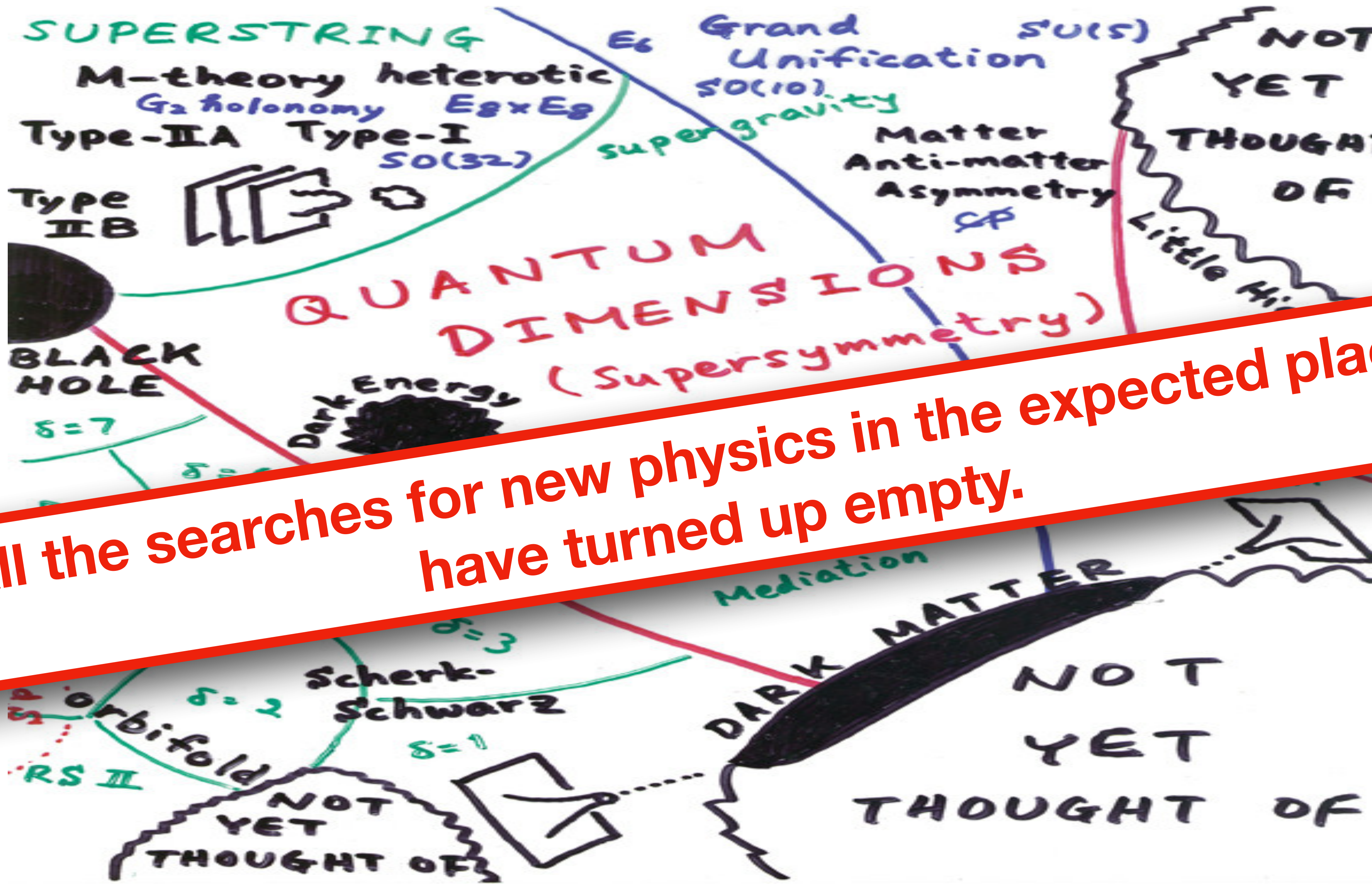
The system looks for patterns and extracts features in data.

Applications) Clustering

Anomaly detection

Anomaly Detection

We have considered many possibilities of BSM physics with top-down theory prejudice (supersymmetry, extra dimension, …)



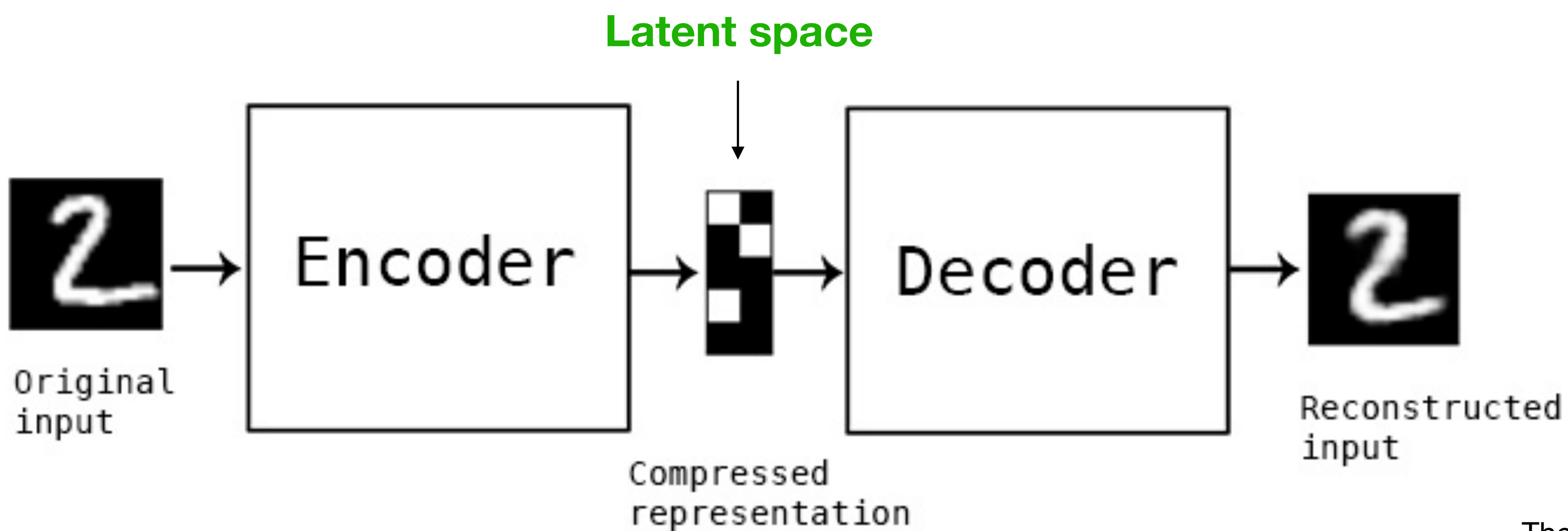
All the searches for new physics in the expected places have turned up empty.

Hitoshi Murayama

We need more ways to discover <u>the unexpected</u> at the LHC, and here is where unsupervised machine learning comes into play.

Autoencoder

Autoencoder is an unsupervised learning algorithm that maps an input to a latent compressed representation and then back to itself.



The Keras Blog

Anomaly detection with autoencoder

- Autoencoder learns to map background events back to themselves.
- It fails to reconstruct anomalous events that it has never encountered.

→ Signal the existence of anomaly !

Sample Generation

The idea is general, but concentrate on detection of anomalous jets.

Generate jet samples by using PYTHIA for hadronization and Delphes for detector simulation.

Background : QCD jets $p_T \in [800,\ 900]$ GeV $|\eta| < 1$

Signal jets: top jets, RPV gluino jets $m_{\tilde{g}} = 400$ GeV
(decay to 3 light quark jets)

Match requirement : heavy resonance is within the fat jet, $\Delta R < 0.6$

Merge requirement : the partonic daughters of heavy resonance is within the fat jet, $\Delta R < 0.6$

We use sample sizes of 100k events for training and testing.
(The performance seems to saturate.)

Jet Images

Concentrate on jet images (2D of eta and phi) whose pixel intensities correspond to total pT.

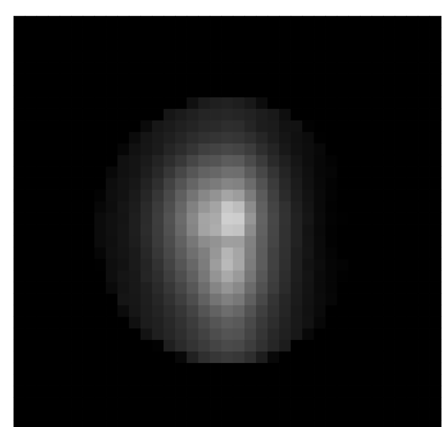
Image pre-processing

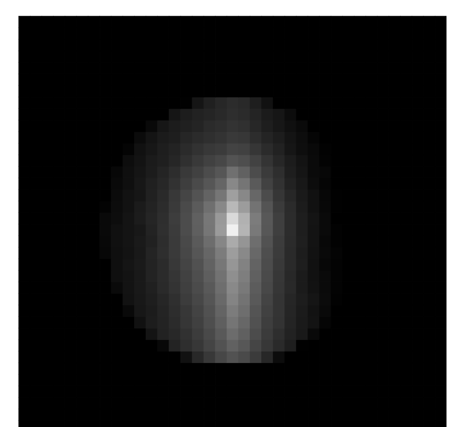
1. Shift an image so that the centroid is at the origin
2. Rotate the image so that the major principal axis is vertical
3. Flip the image so that the maximum intensity is in the upper right region
4. Normalize the image to unit total intensity
5. Pixelate the image (37 x 37 pixels)

Average images

Left : top jets

Right : QCD jets





Macaluso, Shih (2018)

Autoencoder Architectures

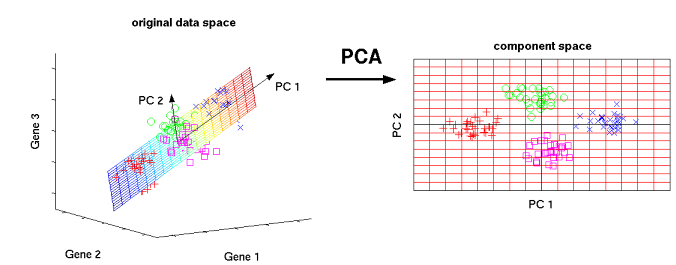
Reconstruction error : a measure for how well autoencoder performs.

$$L(x,\hat{x}) = \frac{1}{n}\sum_{i=1}^{n}\left|x_i - \hat{x}_i\right|^2$$

x : inputs

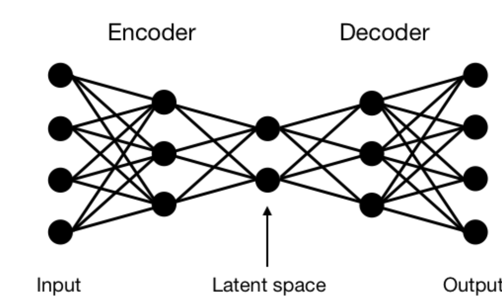
$\hat{x}$: outputs

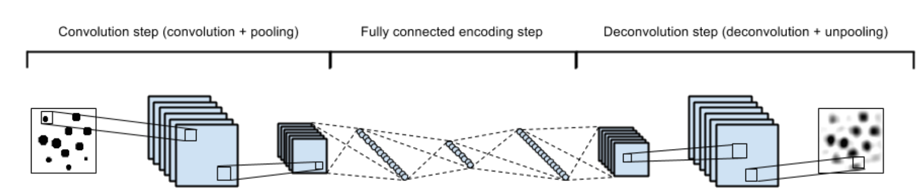
Train autoencoder to <u>minimize the reconstruction error</u> on background events.



Architectures we consider :

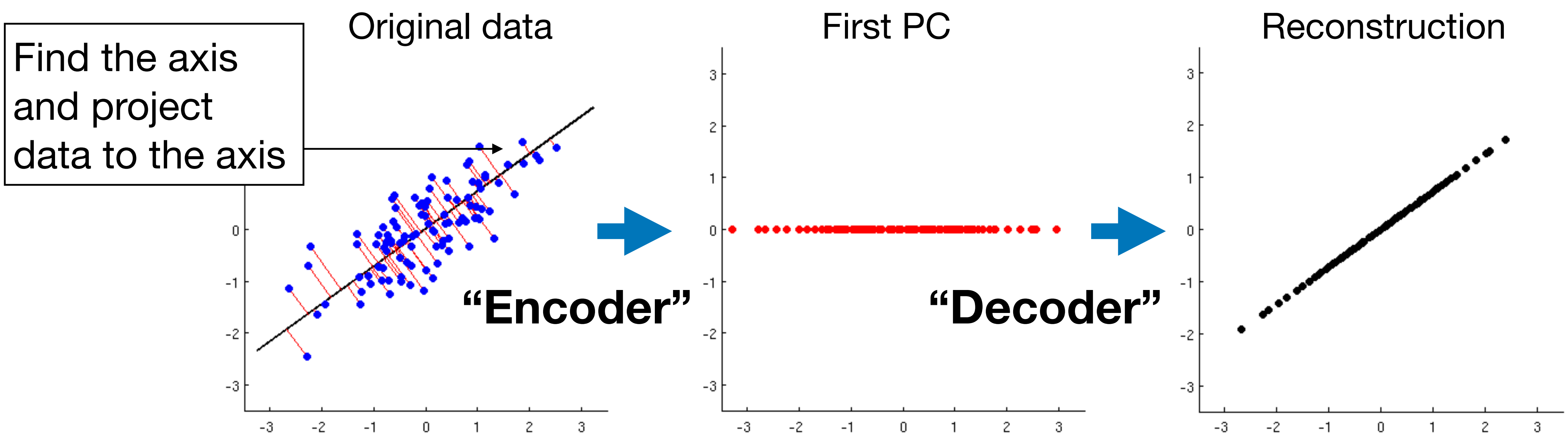
- ✓ Principal Component Analysis (PCA)
- ✓ Simple (dense) autoencoder
- ✓ Convolutional (CNN) autoencoder





Principal Component Analysis

PCA is a technique to drop the least important variables by focusing on variance of data.



Eigenvectors of covariance matrix of $\boldsymbol{x}_n - \boldsymbol{c}_0 \ \ (\boldsymbol{c}_0 = \sum_n \boldsymbol{x}_n / N)$ give desired axes.

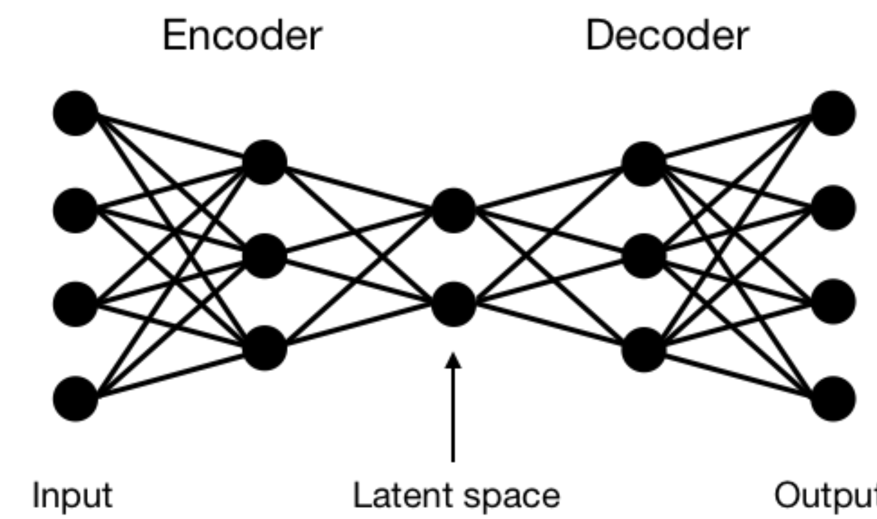
$\Gamma = (\boldsymbol{e}_1 \ \boldsymbol{e}_2 \ \ldots \ \boldsymbol{e}_d)$ d : the number of principal components ($d < D$)

"PCA autoencoder"

"Encoder" : $\tilde{\boldsymbol{x}}_n = (\boldsymbol{x}_n - \boldsymbol{c}_0)\Gamma$ "Decoder" : $\boldsymbol{x}'_n = \tilde{\boldsymbol{x}}_n \Gamma^T + \boldsymbol{c}_0$

Simple Autoencoder

Autoencoder with a single dense (fully-connected) layer as encoder and as decoder.



✓ Encoder and decoder are symmetric.

✓ The number of neurons in a hidden layer = 32.

✓ Flatten a jet image into a single column vector.

✓ We use Keras with Tensorflow backend for implementation.

Training details

✦ The default Adam algorithm for optimizer.

✦ Minibatch size of 1024 ⟵ The number of images fed into the network at one time; ~100 iterations of optimization in one epoch

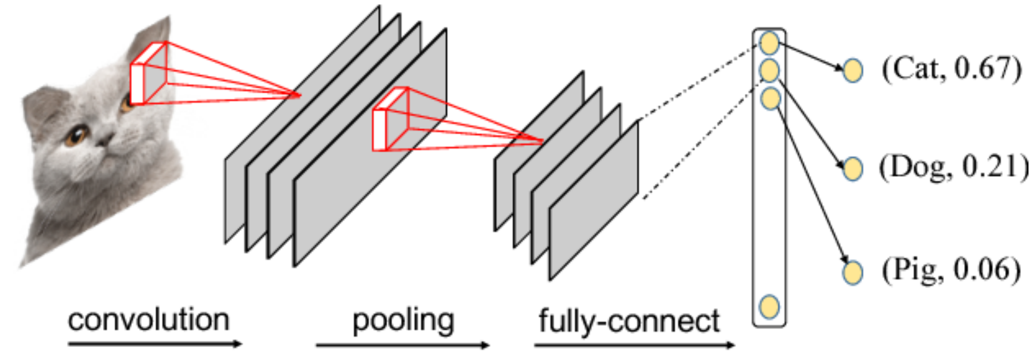
✦ Early stopping : threshold = 0 and patience = 5 ⟵ To avoid overtraining

Convolutional Autoencoder

Convolutional Neural Network (CNN)

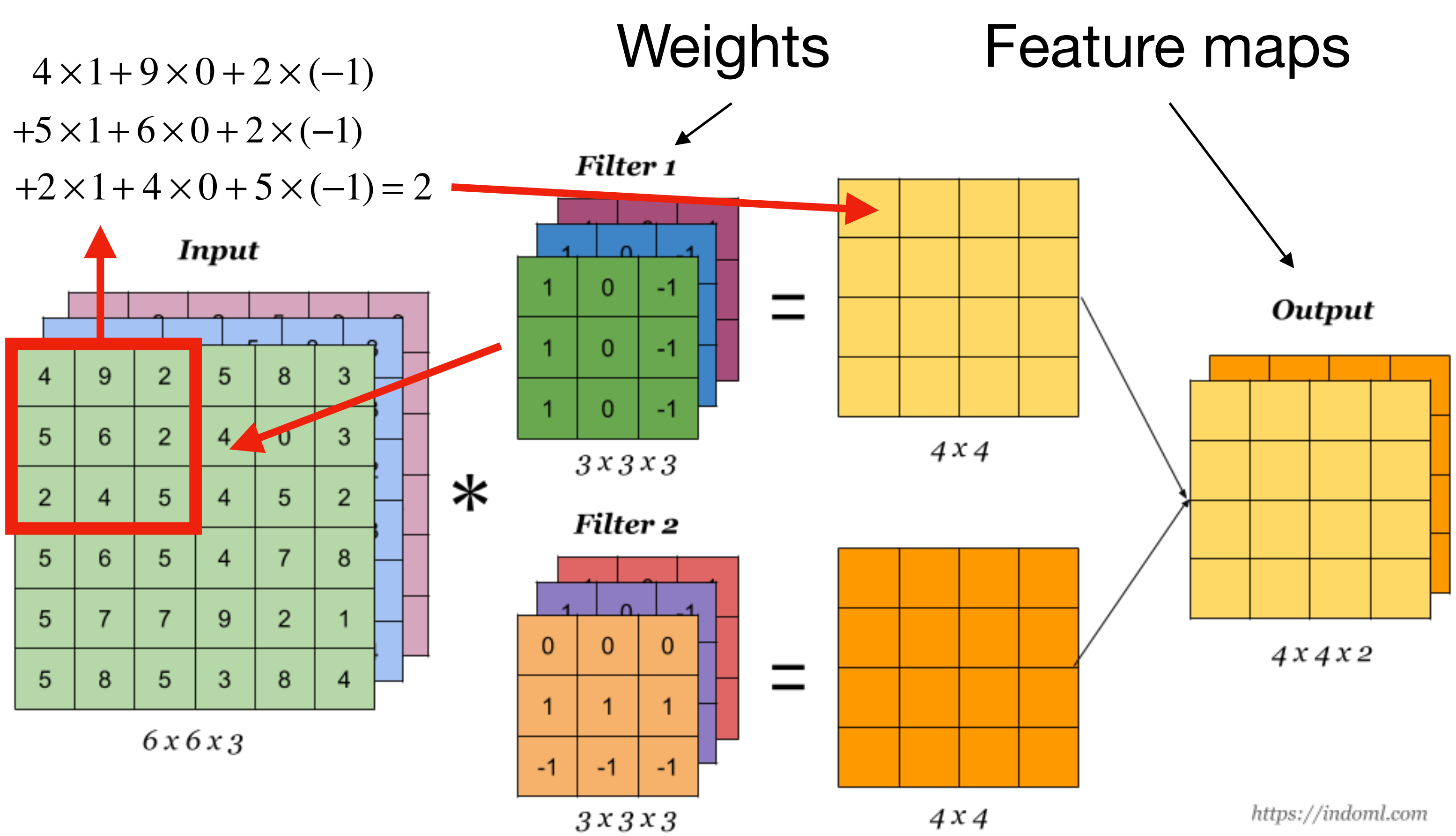
✓ Show high performance for image recognitions

✓ Maintain the spacial information of images



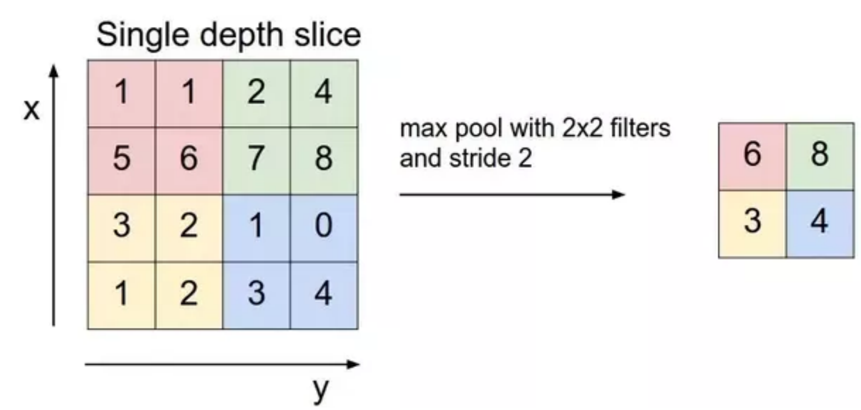
arXiv:1712.01670

Convolutional layer



Max pooling

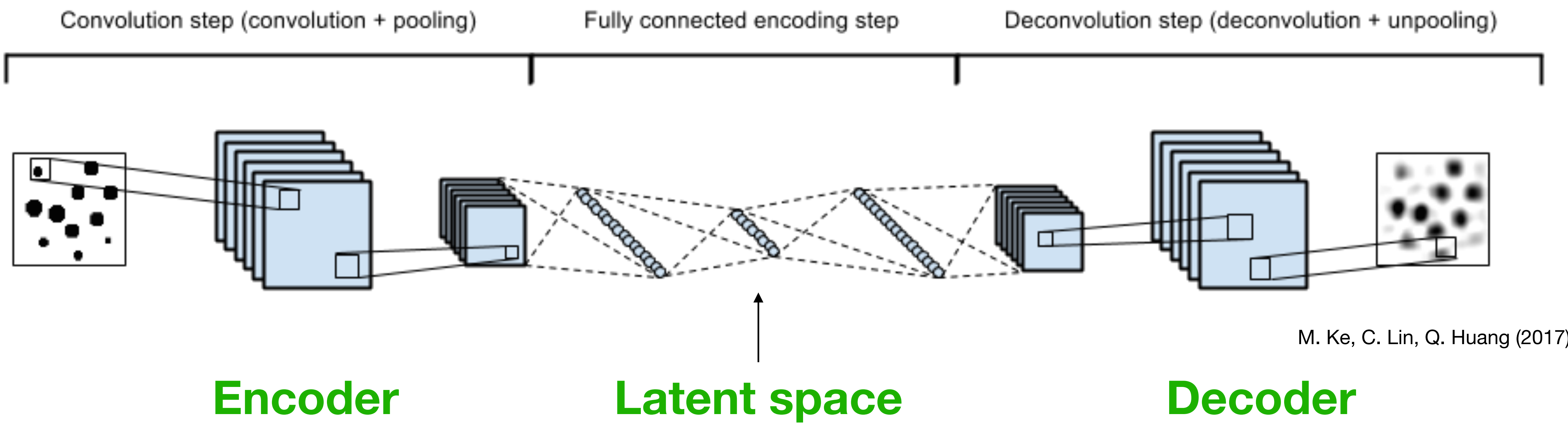
Reduce the image size



Up sampling (pooling)
also exists in autoencoder.

Convolutional Autoencoder

Autoencoder architecture :



128C3-MP2-128C3-MP2-128C3-32N-6N-32N-12800N-128C3-US2-128C3-US2-1C3

128C3 : 128 filters with a 3x3 kernel

MP2 : max pooling with a 2x2 reduction factor

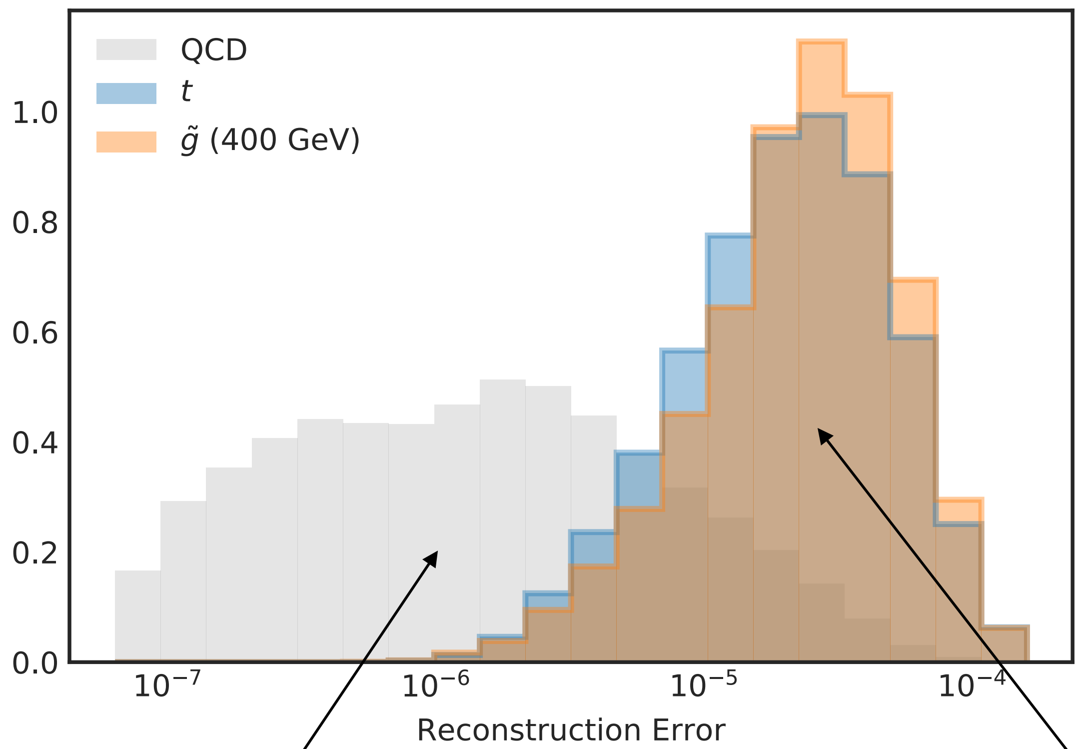
32N : a fully-connected layer with 32 neurons

US2 : up sampling with a 2x2 expansion factor

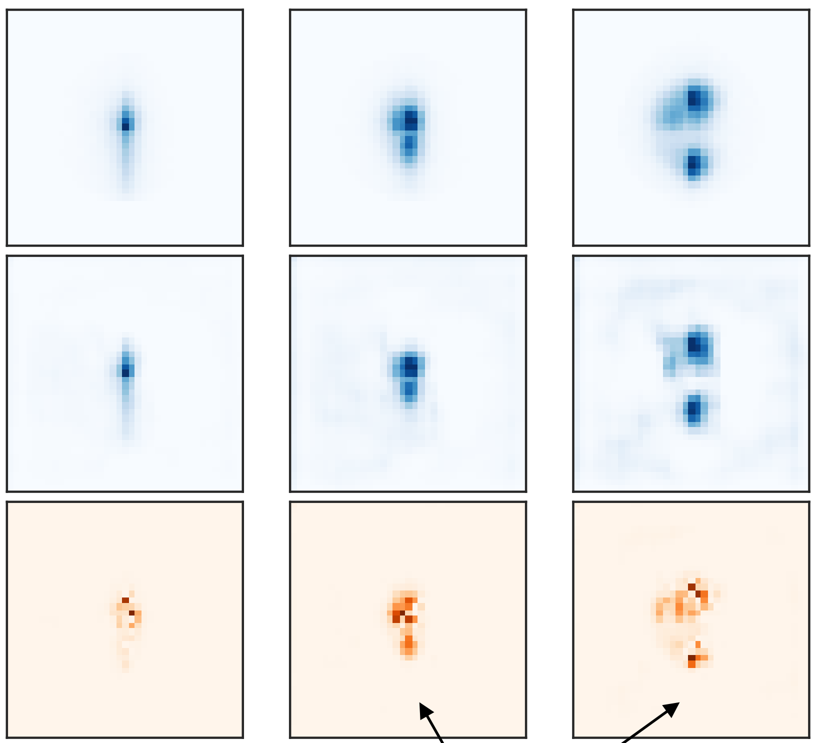
Weakly-supervised mode

Weakly-supervised case with pure background events for training.

Convolutional autoencoder



Average images



More error

Autoencoder learns to reconstruct the QCD background.

Autoencoder fails to reconstruct the signals.

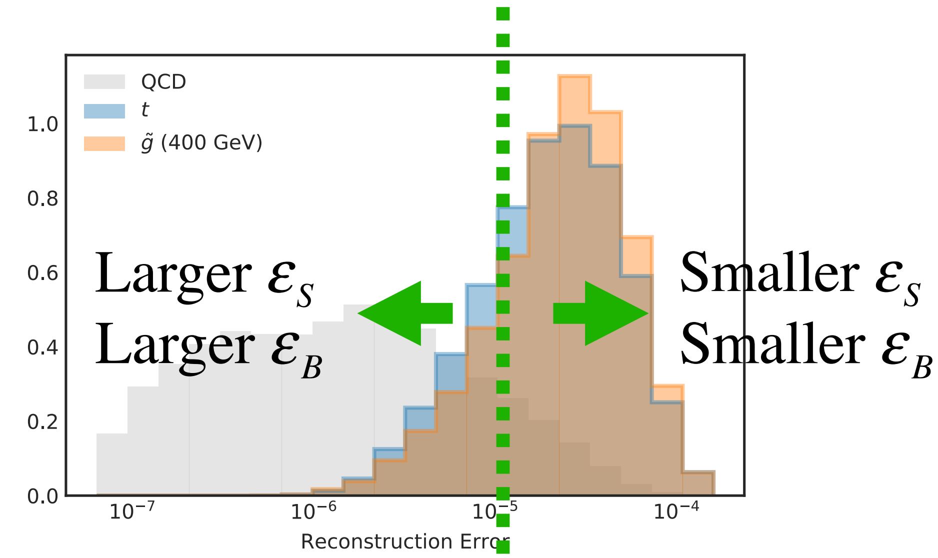
Reconstruction error is used as an anomaly threshold.

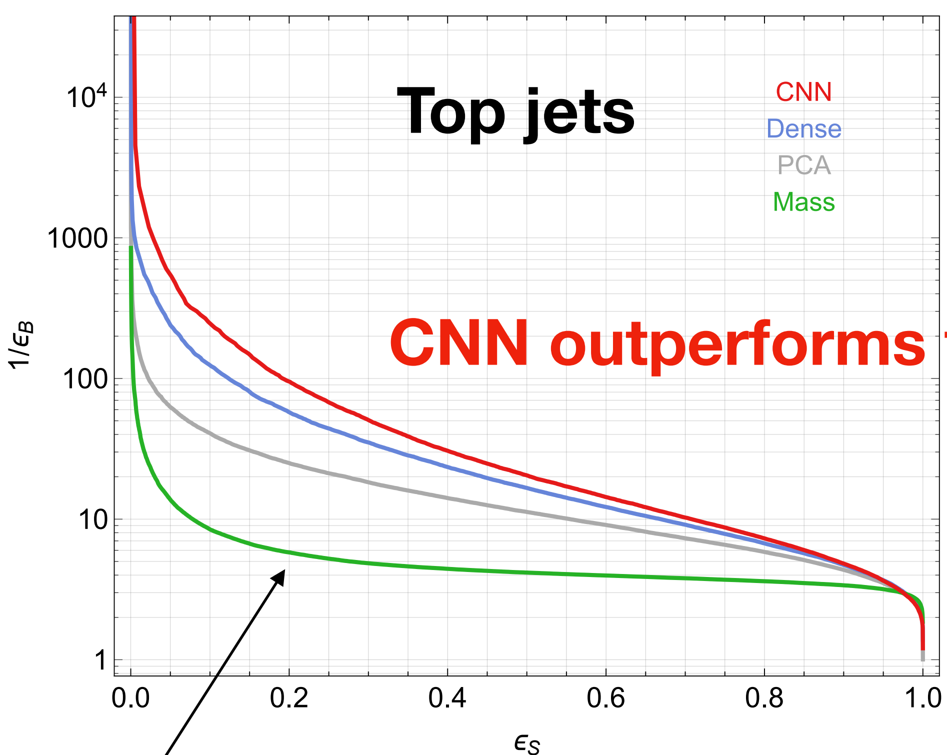
Autoencoder Performance

Performance measure :

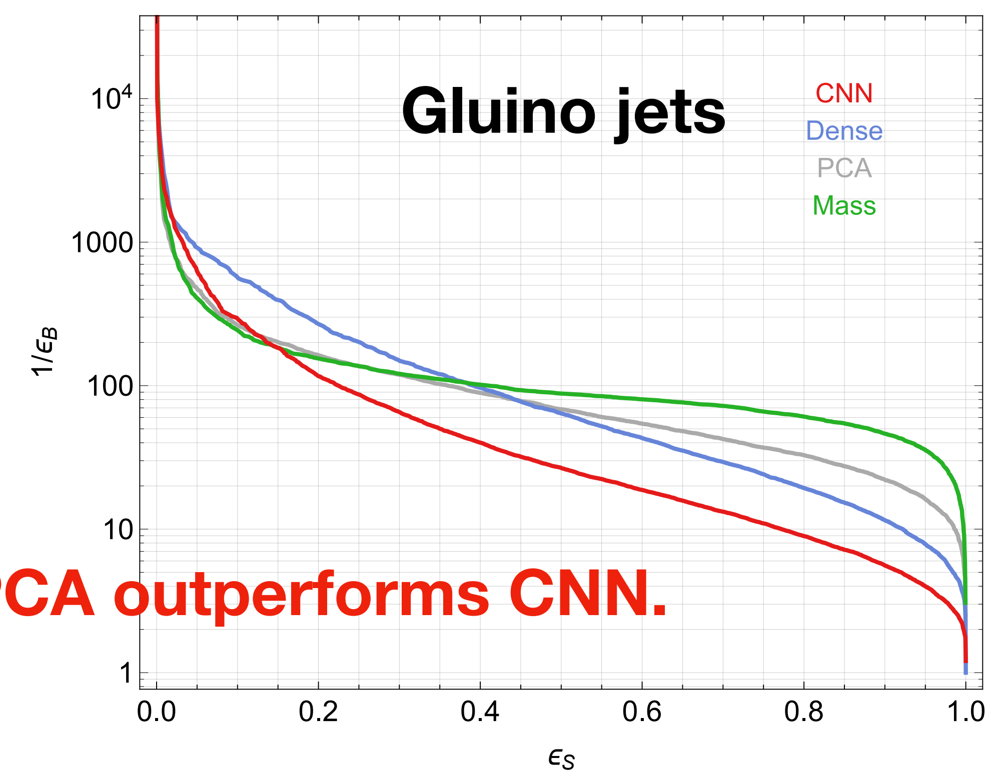
$$\varepsilon_S = \frac{\text{(Correctly classified into signals)}}{\text{(Total number of signal jets)}}$$

$$\varepsilon_B = \frac{\text{(Misclassified into signals)}}{\text{(Total number of backgrounds)}}$$





CNN outperforms the others.



PCA outperforms CNN.

Jet mass as anomaly threshold

For gluino jets, PCA ROC curve approaches jet mass ROC curve, suggesting PCA reconstruction error is highly correlated with jet mass.

Choosing the Latent Dimension k

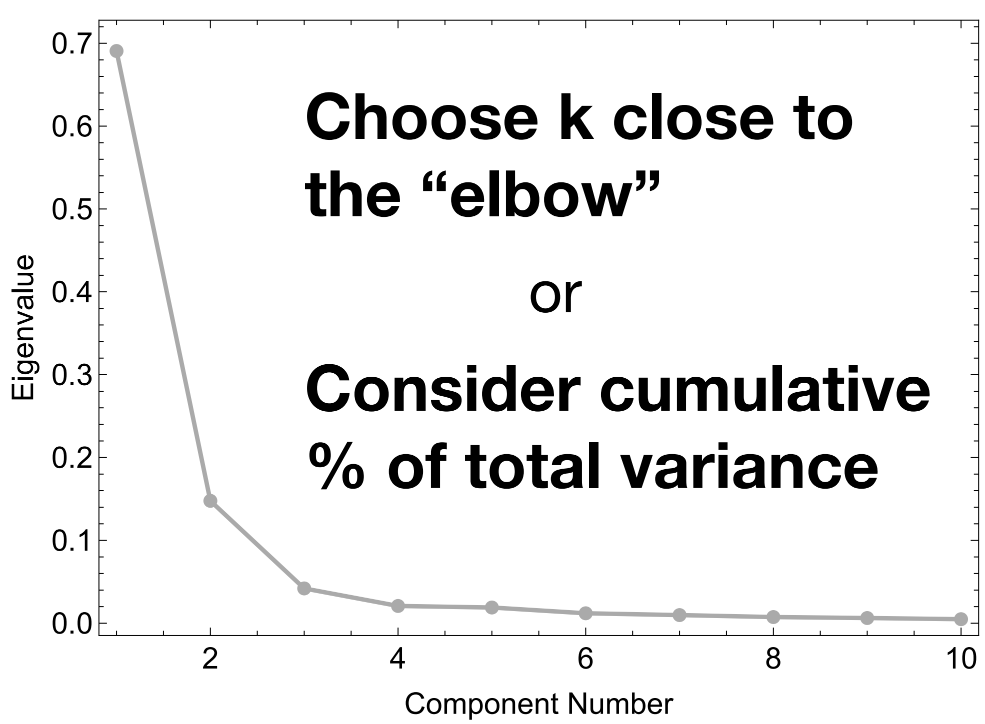
Too small k → Autoencoder cannot capture all the features.

Too large k → Autoencoder approaches trivial representation.

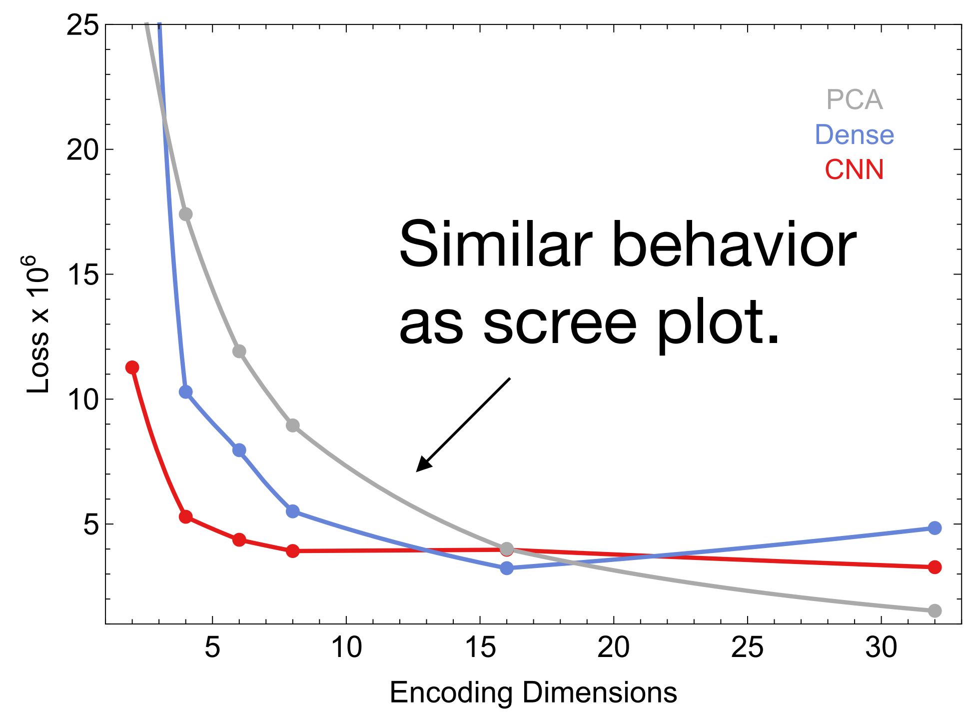
Optimizing the latent dimension using various signals is **NOT** a good idea.

Instead, we use the number of principal components in PCA and reconstruction error.

Amount of variance ("scree plot") :



Reconstruction error :

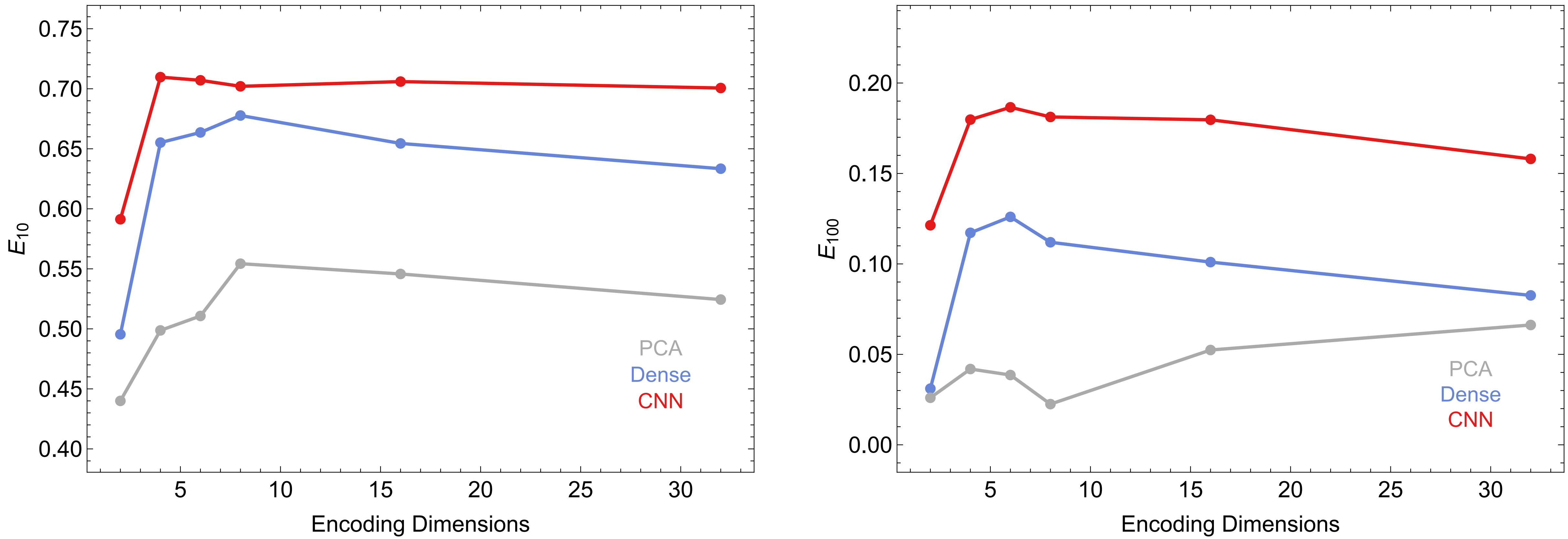


We choose k = 6.

Choosing the Latent Dimension k

Let's examine our choice by looking at the top signal.

$E_{10,\,100}$: the signal efficiency at 90% and 99% background rejection



Each dot corresponds to the average of 5 independent training runs.

Autoencoder performance plateaus around k = 6.

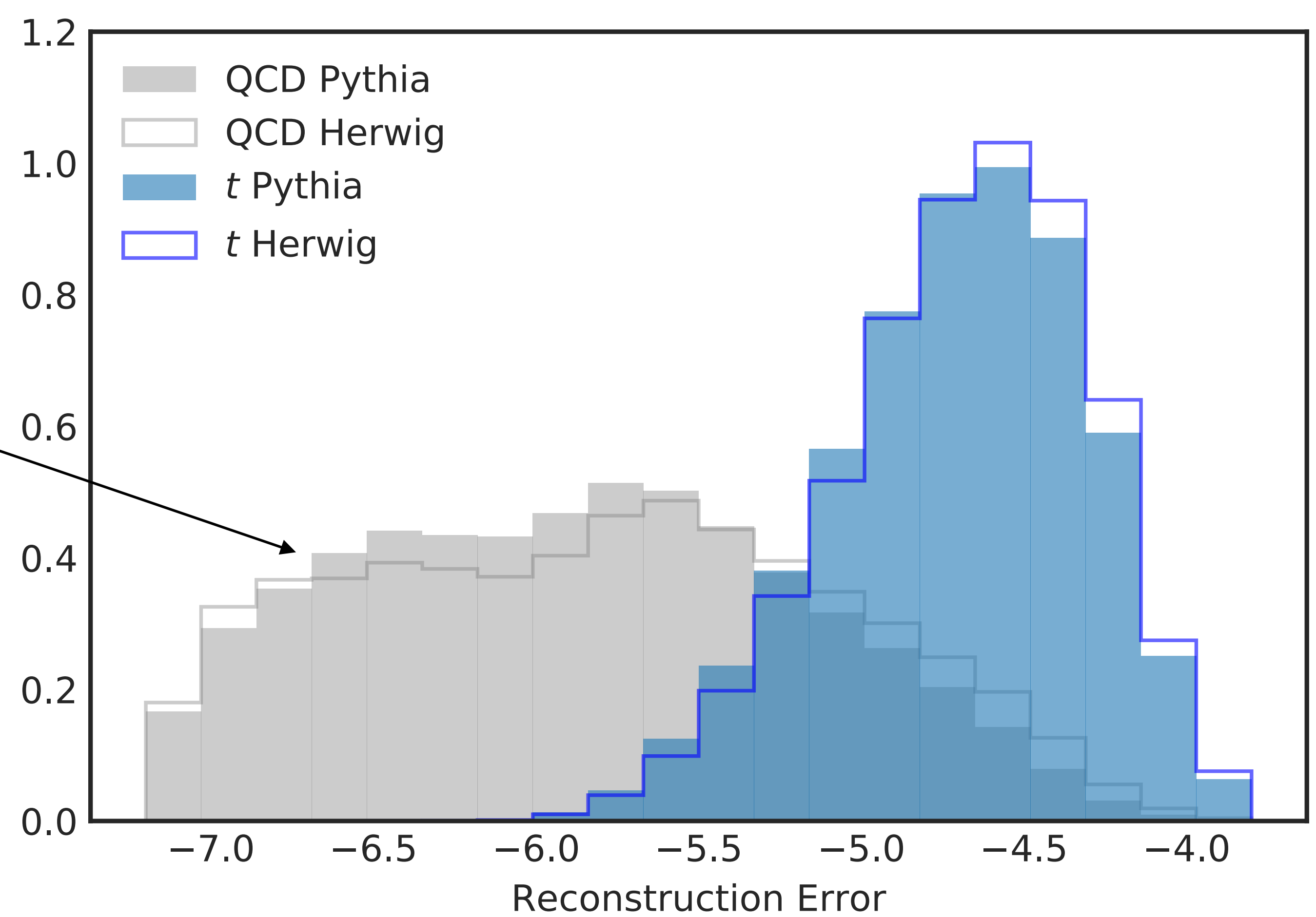
Robustness with Other Monte Carlo

Autoencoder really does not learn artifacts special to a Monte Carlo?

One possible check :

Evaluate autoencoder (trained on PYTHIA samples) on jet samples produced with HERWIG.

Comparison of reconstruction error (top jets, CNN)



The differences are small.

Separation between background and anomaly is preserved.

Autoencoder probably learns fundamental jet features.

Unsupervised mode

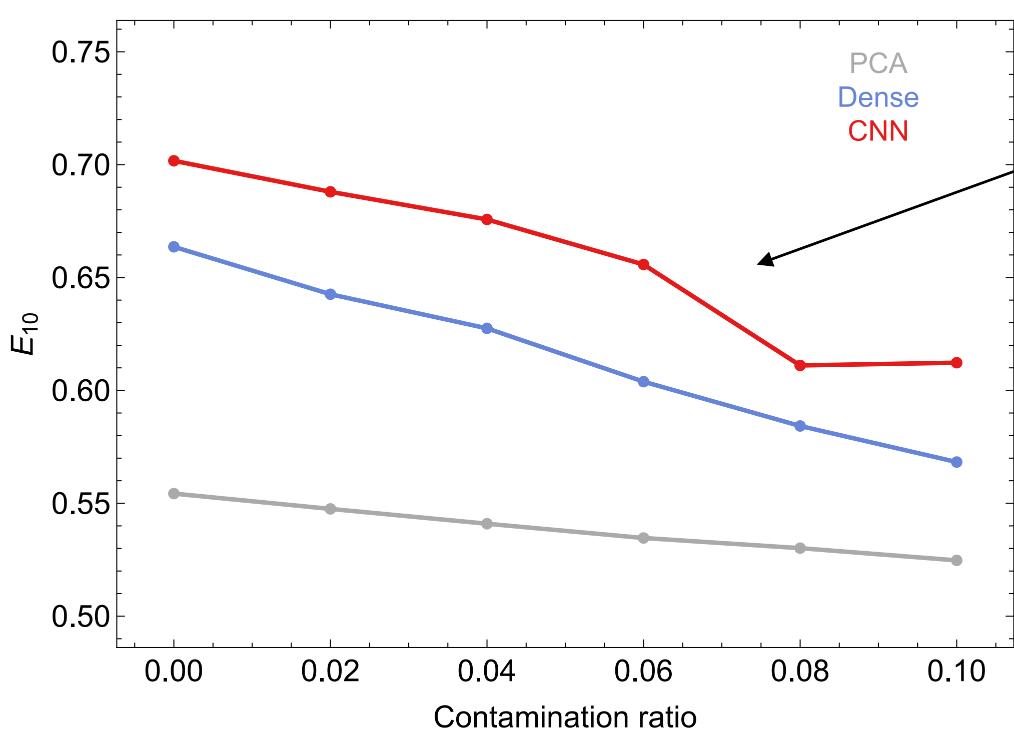
A much more exciting possibility is to train autoencoder on actual data (which may contain some amount of signals).

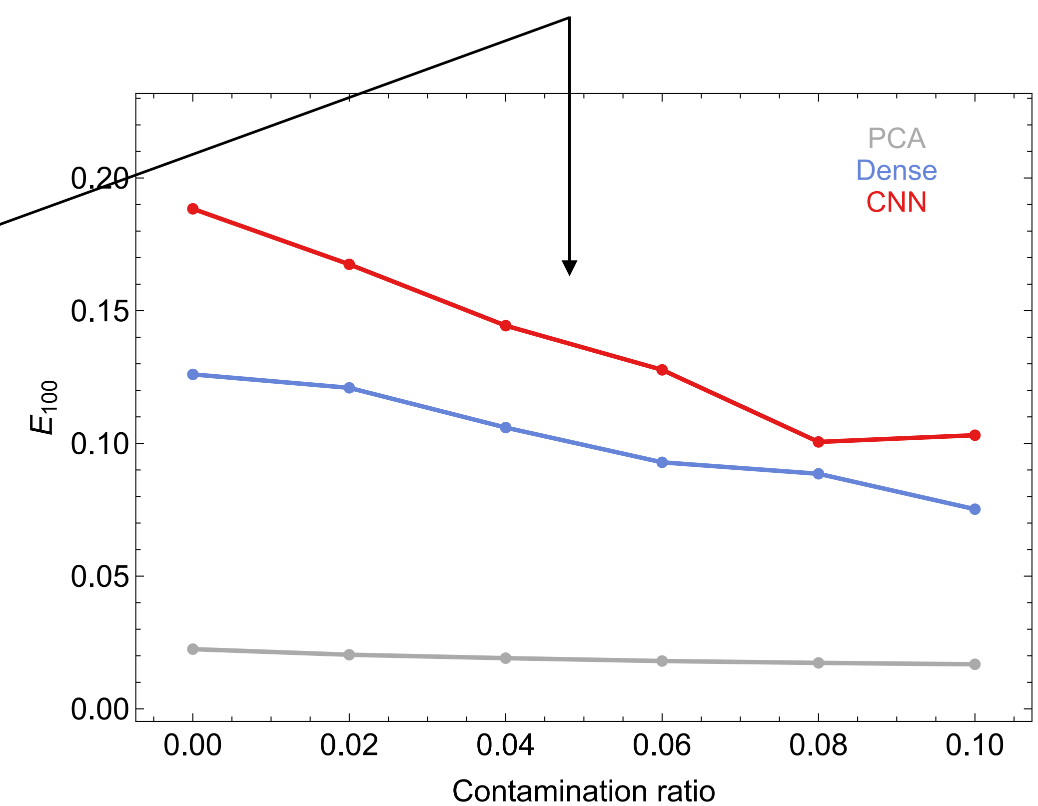
Train autoencoder on a sample of backgrounds contaminated by a small fraction of signal events.

➡ **Autoencoder performance is remarkably stable against signal contamination.**

Reduction is not dramatic !

Top jets for anomalous events





Correlation with Jet Mass

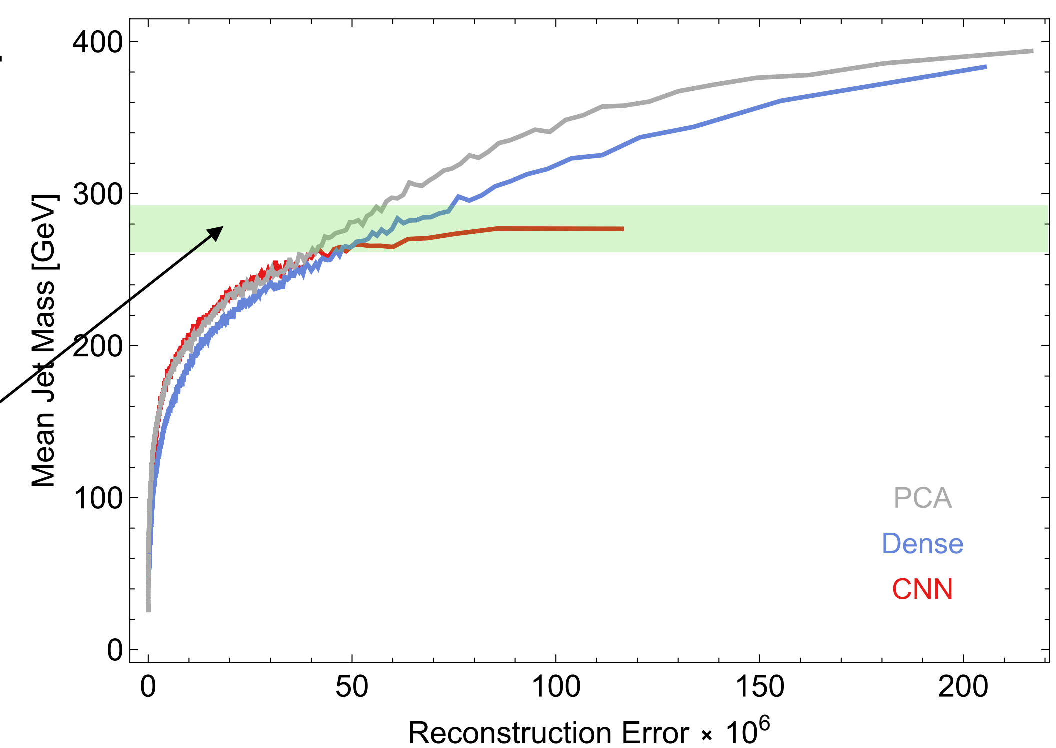
In actual new physics searches, we look for subtle signals …

It's more powerful to combine autoencoder with another variable such as jet mass.

Cut hard on reconstruction error to clean out the QCD background and look for a bump in jet mass distribution.

Reconstruction error should not be correlated with jet mass.

Mean jet mass in bins of reco error for the QCD background

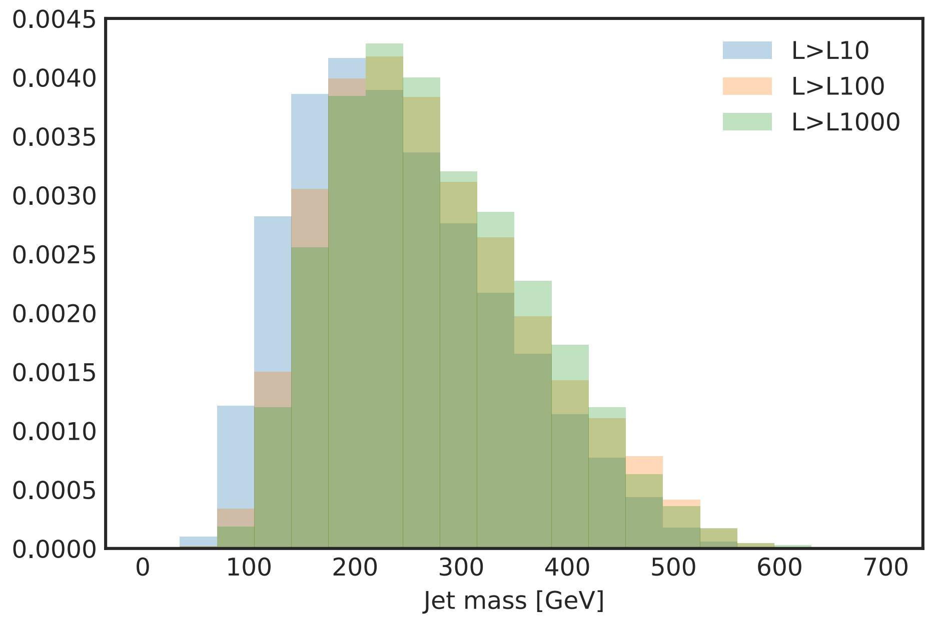


For **PCA and dense**, reco error is correlated with jet mass.

Jet mass distribution is stable against cutting on **CNN loss.**

Correlation with Jet Mass

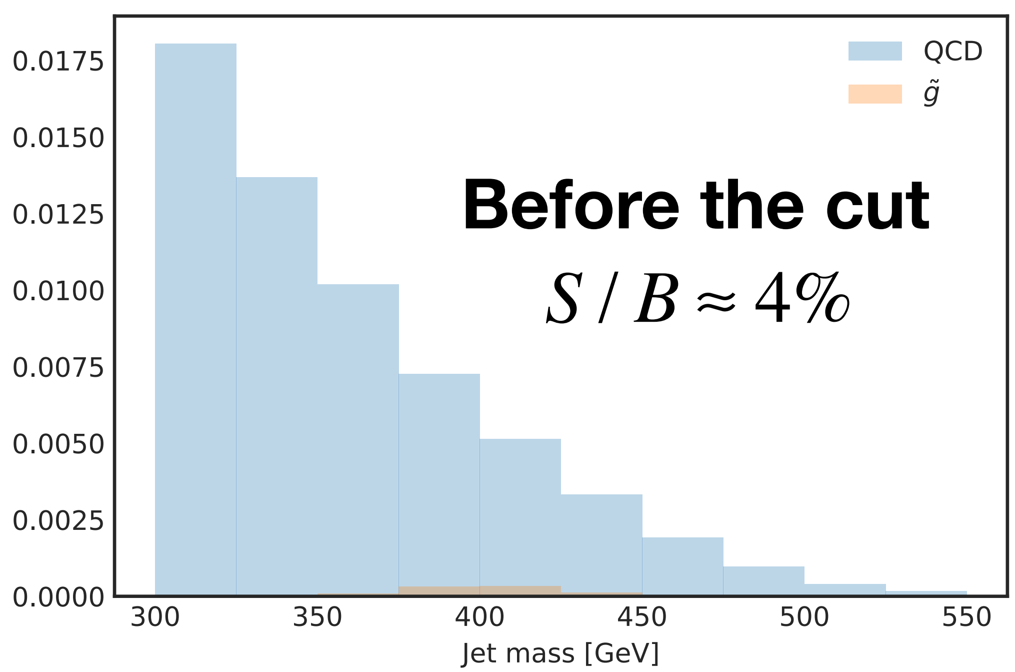
Jet mass distributions after cuts on CNN loss

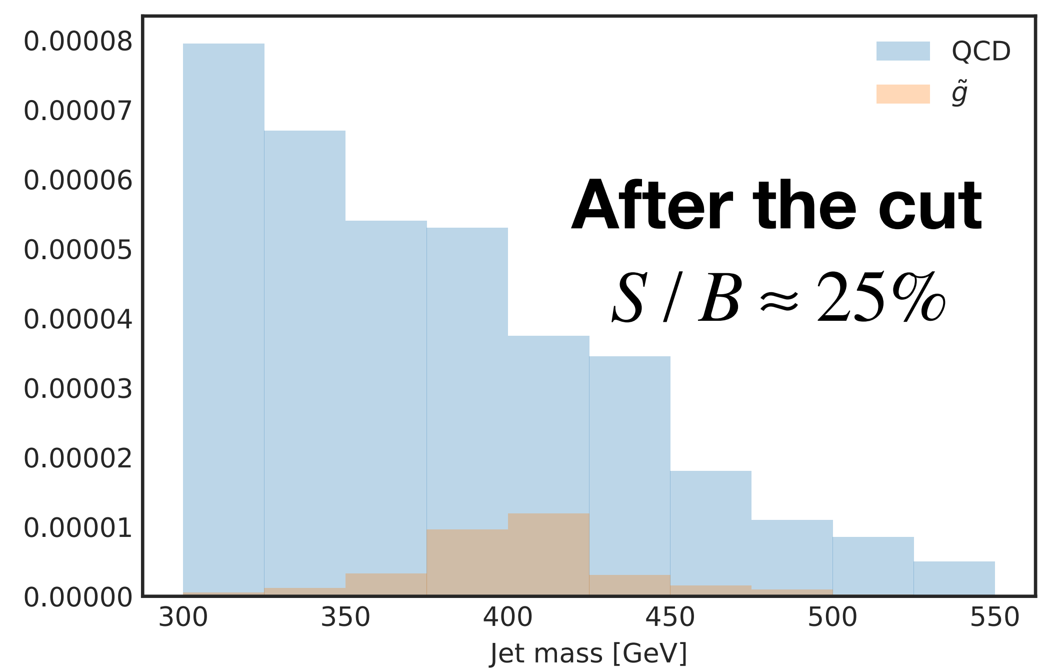


Reduce the QCD background by a factor of 10, 100 and 1000.

Convolutional autoencoder is useful for a bump hunt in jet mass above 300 GeV.

Jet mass histograms normalized to LO gluino and QCD cross sections





Comments on “QCD or What?”

T. Heimel, G. Kasieczka, T. Plehn, J. Thompson, arXiv:1808.08979 [hep-ph].

They also consider anomaly detection through autoencoder.

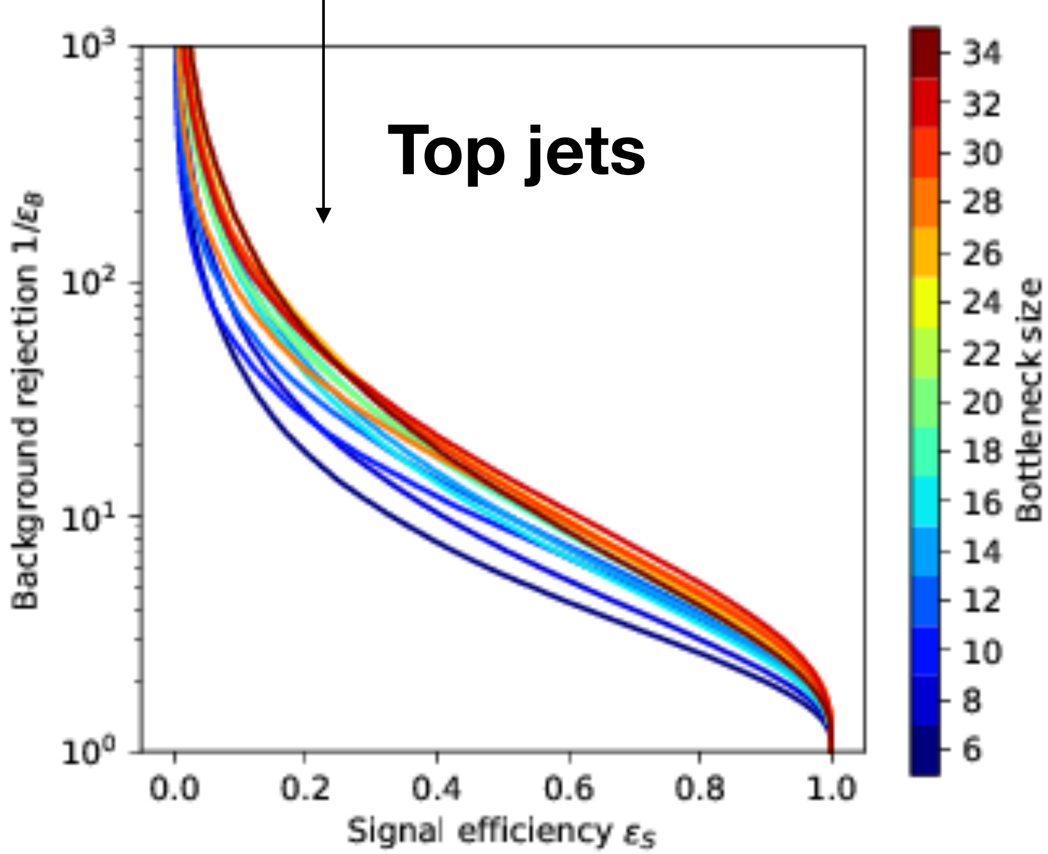
Signal jets : top jets, scalar decay to jets, dark showers

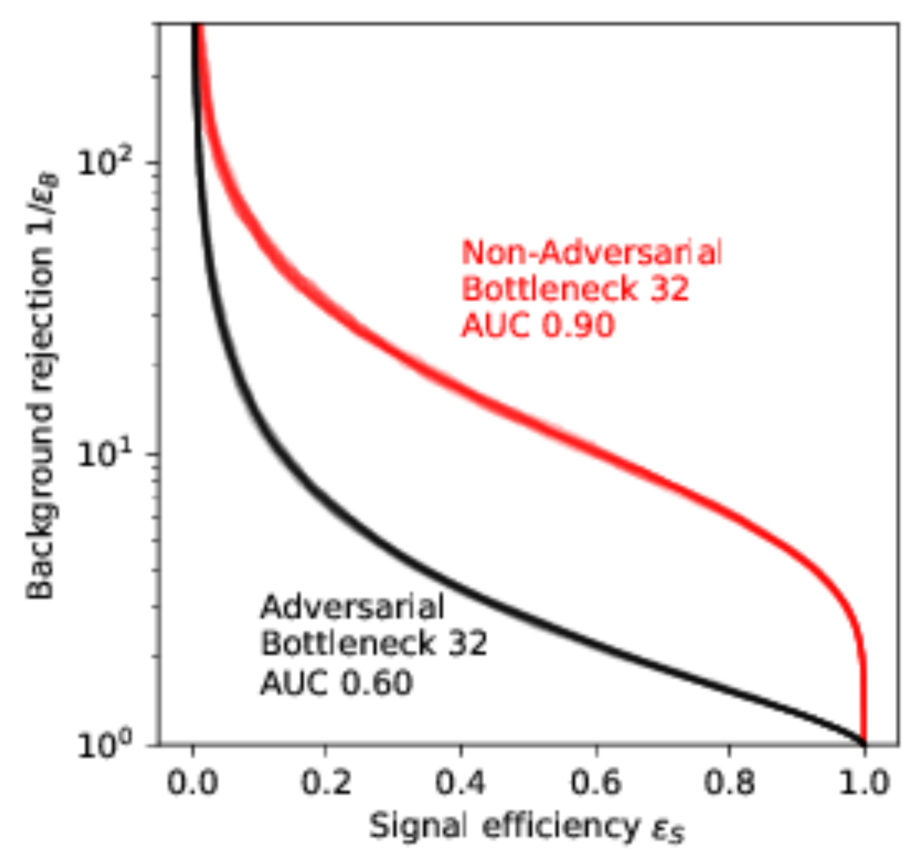
Performance is comparable.

$$pp \to (\phi \to aa \to c\bar{c}\ c\bar{c}) + \text{jets}$$

$$m_a = 4\ \text{GeV} \qquad m_\phi = m_t = 175\ \text{GeV}$$

Top jets





Comments on “QCD or What?”

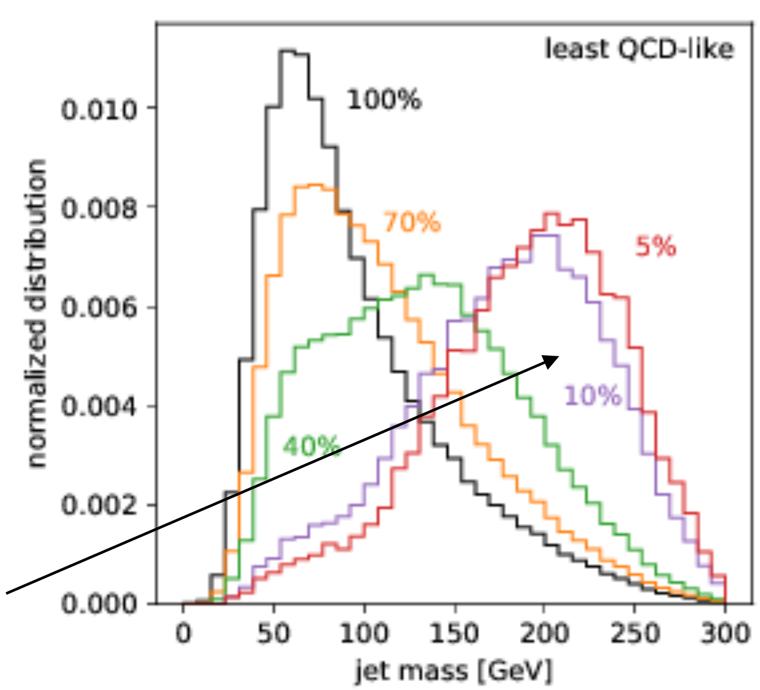
Correlation with jet mass

They take an alternative approach using adversarial networks.

Additional adversary tries to extract jet mass from autoencoder output. ↔ Autoencoder wants the adversary to be as unsuccessful as possible.

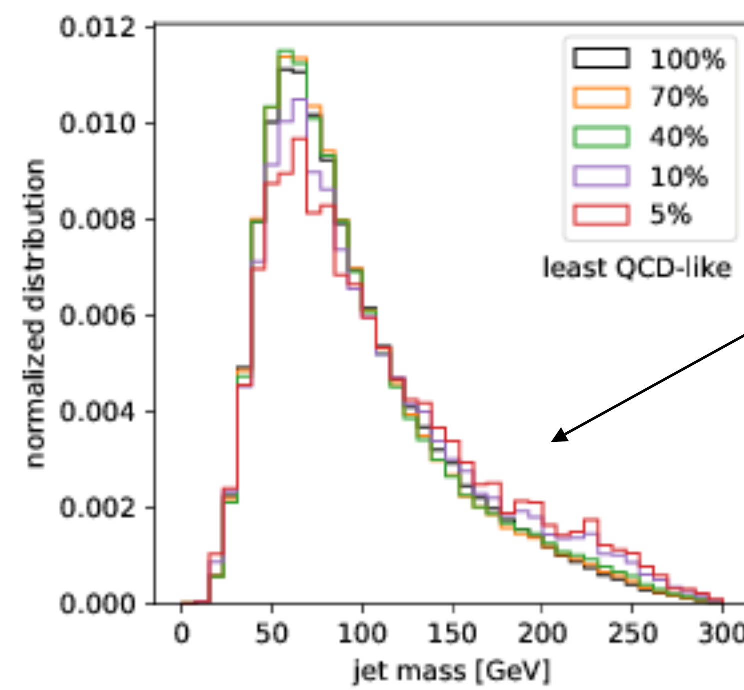
Autoencoder will avoid all information on jet mass.

Non-adversarial



Fake peak

Adversarial



Flatten

Summary

- ✓ Autoencoder learns to map background events back to themselves but fails to reconstruct signals that it has never encountered before.
- ✓ Reconstruction error is used as an anomaly threshold.
- ✓ Autoencoder performance is stable against signal contamination which enables us to train autoencoder on actual data.
- ✓ Jet mass distribution is stable against cutting on CNN loss and convolutional autoencoder is useful for a bump hunt in jet mass.
- ✓ Thresholding on reco error gives a significant improvement of S/B.

Future directions

✓ Testing out autoencoder on other signals.
(Other numbers of subjets, non-resonant particles, …)

✓ Training autoencoder to flag entire events as anomalous, instead of just individual fat jets.

✓ Trying other autoencoder architectures on the market to improve the performance.

✓ Understanding what the latent space actually learns.
(Jet mass? N-subjettiness?)

…

Autoencoder is a powerful new method to search for any signal of new physics without prejudice !

Thank you.

Backup Material

What is Machine Learning?

Machine learning : technique to give computer systems the ability to learn with data without being explicitly programmed.

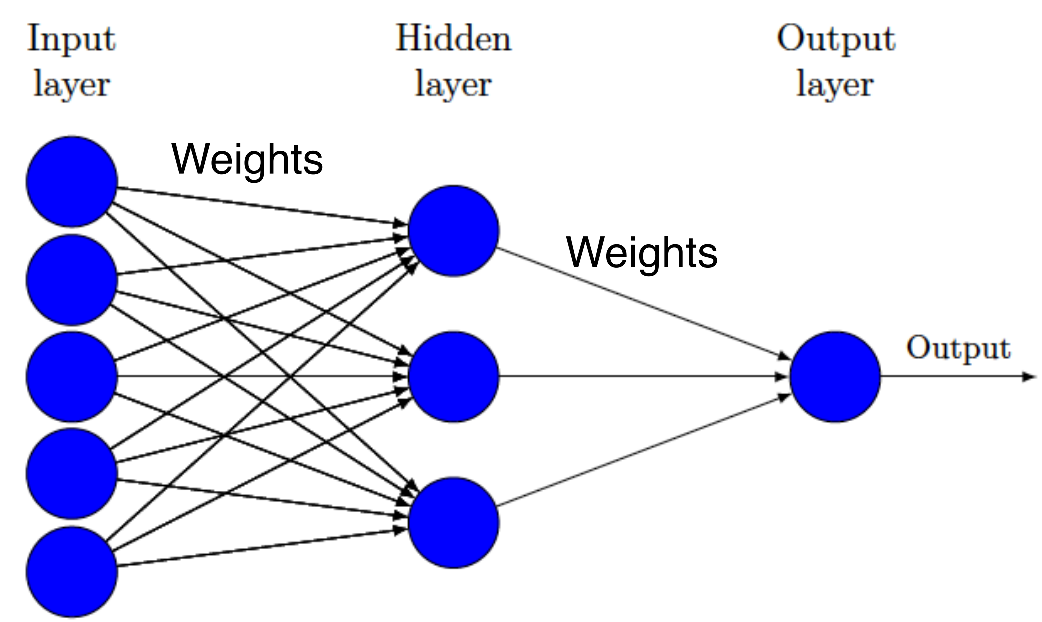
Machine can learn the feature of data which human has not realized !

Neural Networks

✓ Powerful machine learning-based techniques used to solve many real-world problems

✓ Modeled loosely after the human brain

✓ Containing weights between neurons that are tuned by learning from data



Networks contain multiple hidden layers → **Deep learning**

What is Machine Learning?

The goal of training is to minimize loss function :

$$L = \sum_i f(p(\theta, x_i), y_i)$$

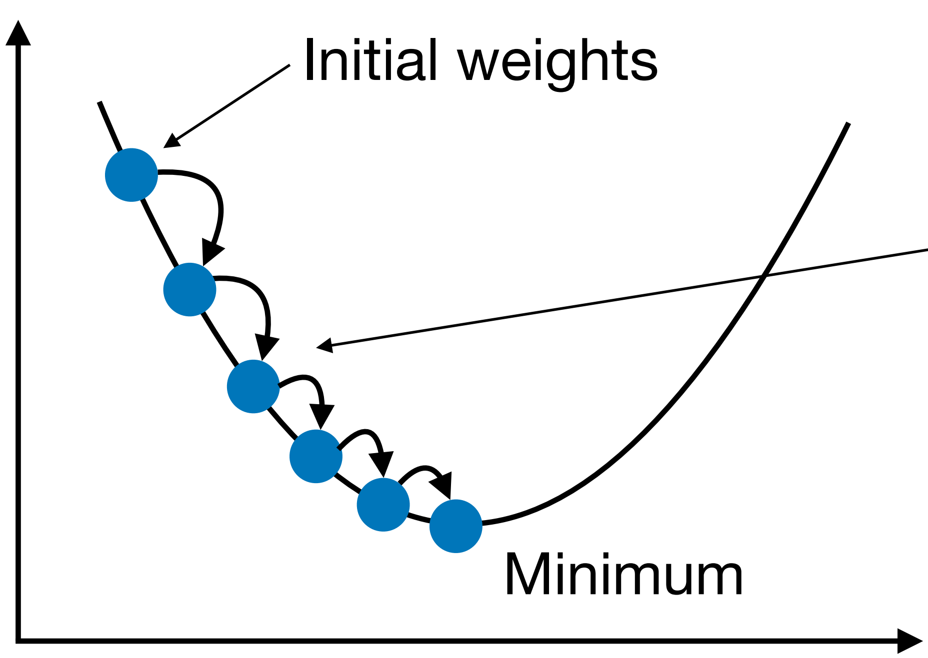
$p(\theta, x_i)$: Prediction θ : Weights

x_i : Input y_i : Target value of example i

Mean squared error (MSE) : $f(p, y) = (p - y)^2$

Cross entropy : $f(p, y) = -(y \log p + (1 - y)\log(1 - p))$

Loss function L



Weights θ

Weights are updated according to derivative of loss function :

$$\Delta\theta = -\eta \nabla L$$

Learning rate

Keras Codes

- **Simple autoencoder**

```
input_img = Input(shape=(37*37,))
layer = Dense(32, activation='relu')(input_img)
encoded = Dense(6, activation='relu')(layer)

layer = Dense(32, activation='relu')(encoded)
layer = Dense(37*37, activation='relu')(layer)
decoded=Activation('softmax')(layer)

autoencoder=Model(input_img,decoded)
autoencoder.compile(loss=keras.losses.mean_squared_error,
              optimizer=keras.optimizers.Adam())
```

Keras Codes

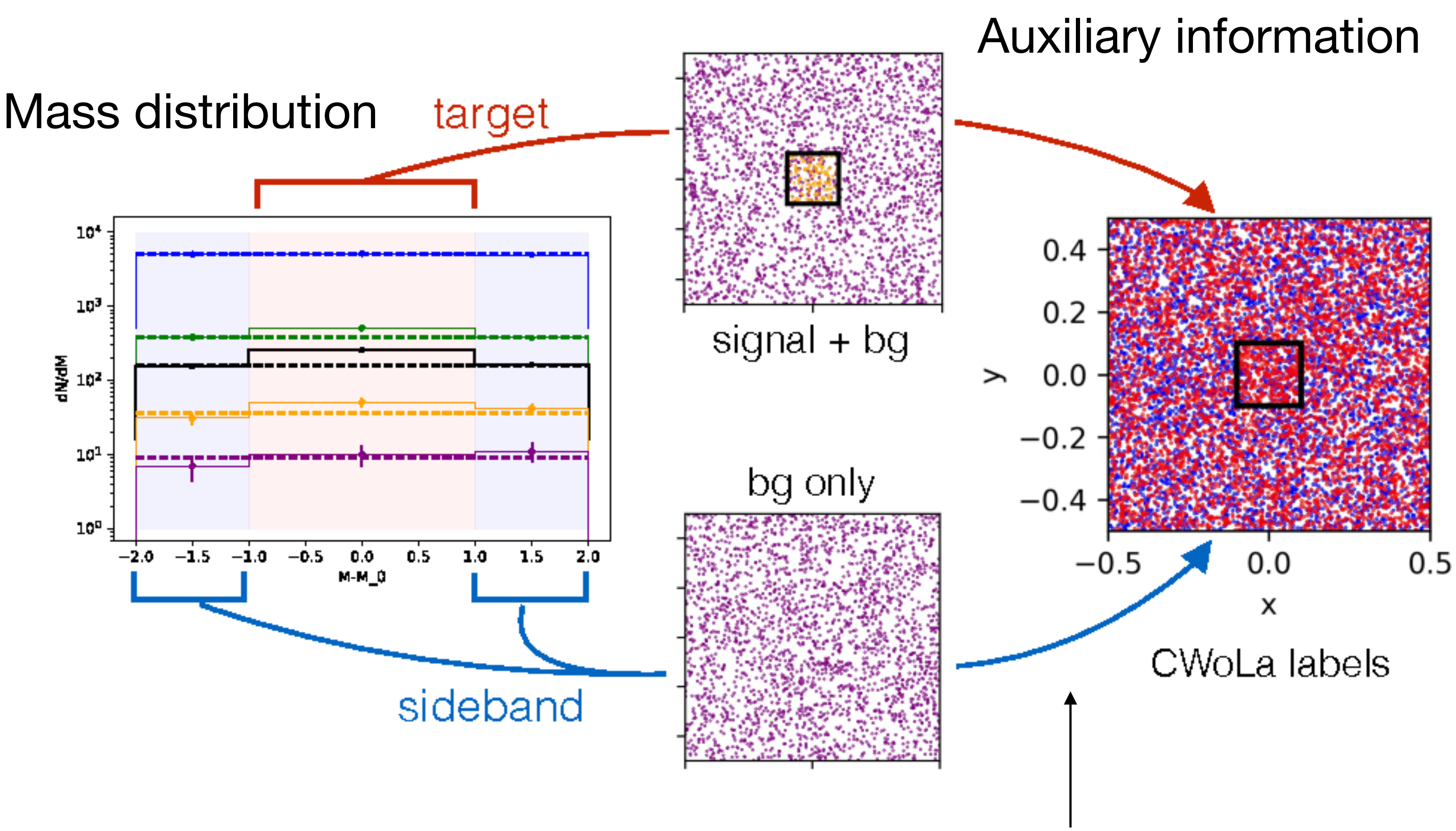
- **Convolutional autoencoder**

```
input_img=Input(shape= (40, 40, 1))

layer=input_img
layer=Conv2D(128, kernel_size=(3, 3),
             activation=’relu’,padding=’same’)(layer)
layer=MaxPooling2D(pool_size=(2, 2),padding=’same’)(layer)
layer=Conv2D(128, kernel_size=(3, 3),
             activation=’relu’,padding=’same’)(layer)
layer=MaxPooling2D(pool_size=(2, 2),padding=’same’)(layer)
layer=Conv2D(128, kernel_size=(3, 3),
             activation=’relu’,padding=’same’)(layer)
layer=Flatten()(layer)
layer=Dense(32, activation=’relu’)(layer)
layer=Dense(6)(layer)
encoded=layer

layer=Dense(32, activation=’relu’)(encoded)
layer=Dense(12800, activation=’relu’)(layer)
layer=Reshape((10,10,128))(layer)
layer=Conv2D(128, kernel_size=(3, 3),
             activation=’relu’,padding=’same’)(layer)
layer=UpSampling2D((2,2))(layer)
layer=Conv2D(128, kernel_size=(3, 3),
             activation=’relu’,padding=’same’)(layer)
layer=UpSampling2D((2,2))(layer)
layer=Conv2D(1, kernel_size=(3, 3),padding=’same’)(layer)
layer=Reshape((1,1600))(layer)
```

CWoLa Hunting

J. Collins, K. Howe, B. Nachman, arXiv:1805.02664 [hep-ph].

Another approach to anomaly detection to extend bump hunt with machine learning.



Classification without labels (CWoLa)

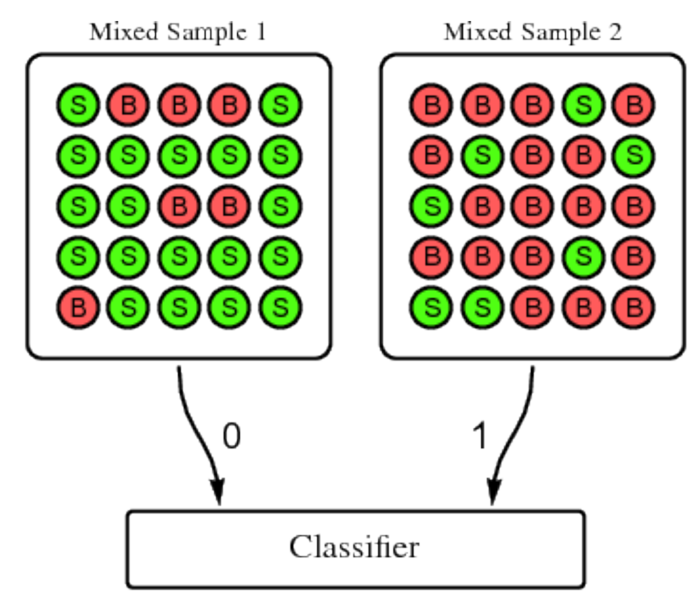
A classifier is trained to distinguish statistical mixtures of classes.

Toy model

$$Y = (x, y)$$

Background : $-0.5 < x < 0.5$, $-0.5 < y < 0.5$

Signal : $-w/2 < x < w/2$, $-w/2 < y < w/2$



Metodiev, Nachman, Thaler