Introduction to Deep Learning

and its applications to the LHC

David Shih

“Machine Learning at LHC”
KMI Nagoya

February 2020

The AI Revolution is Here

The past decade has seen remarkable advances in machine learning and artificial intelligence.

These technological breakthroughs are reshaping the world around us.

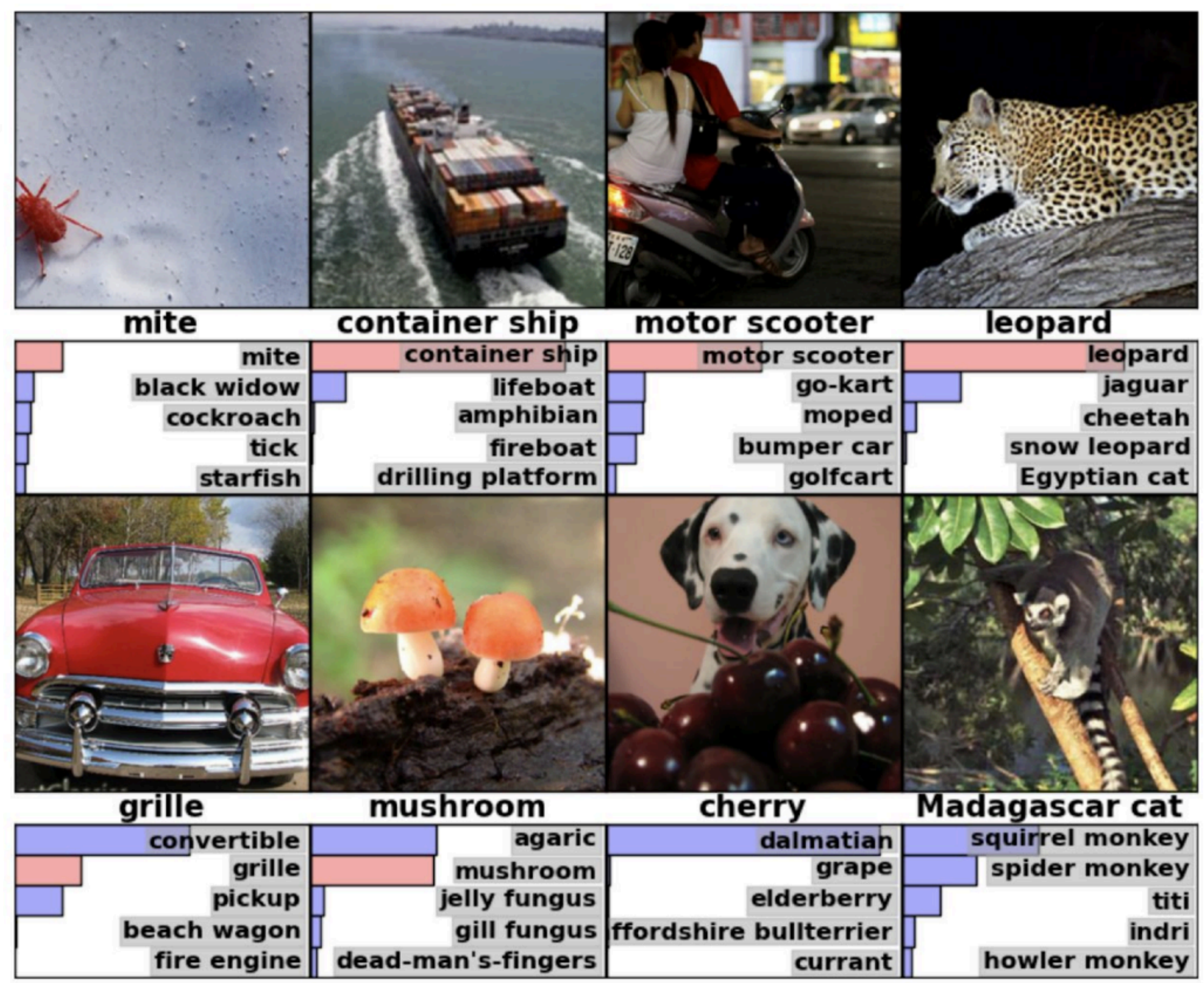
Deep learning also has the potential to revolutionize physics at the LHC.

Deep Learning Breakthrough

ImageNet Challenge

IMAGENET

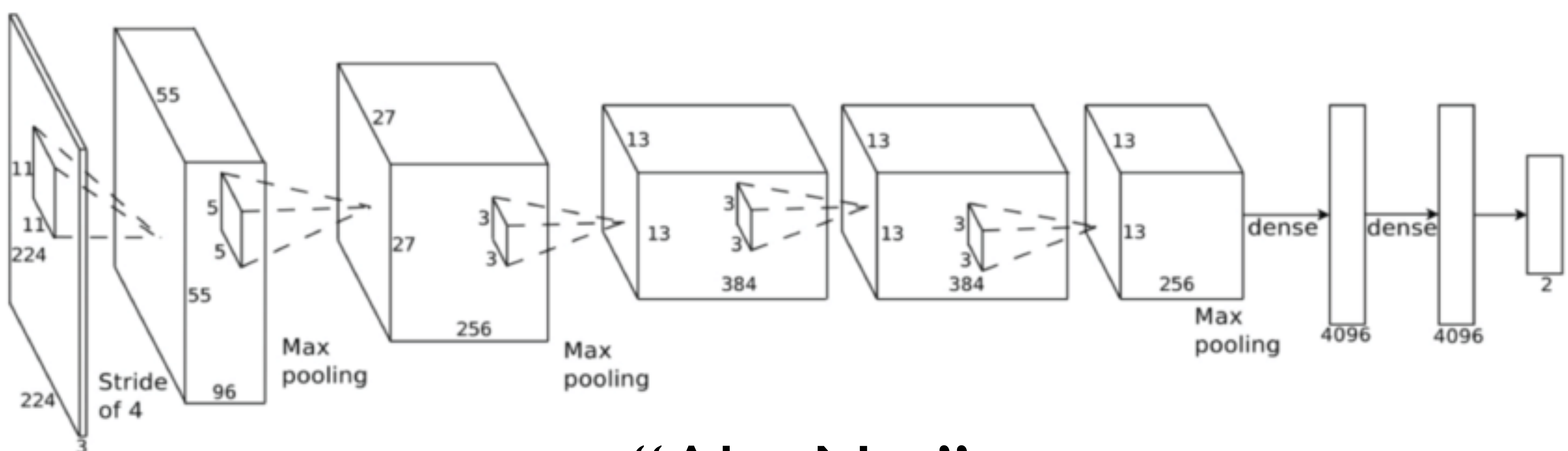
- 1,000 object classes (categories).
- Images:
  - 1.2 M train
  - 100k test.



Deep Learning Breakthrough

In 2012, a deep convolutional neural network won the "ImageNet" image classification competition by a huge margin (Krizhevsky, Sutskever, Hinton)

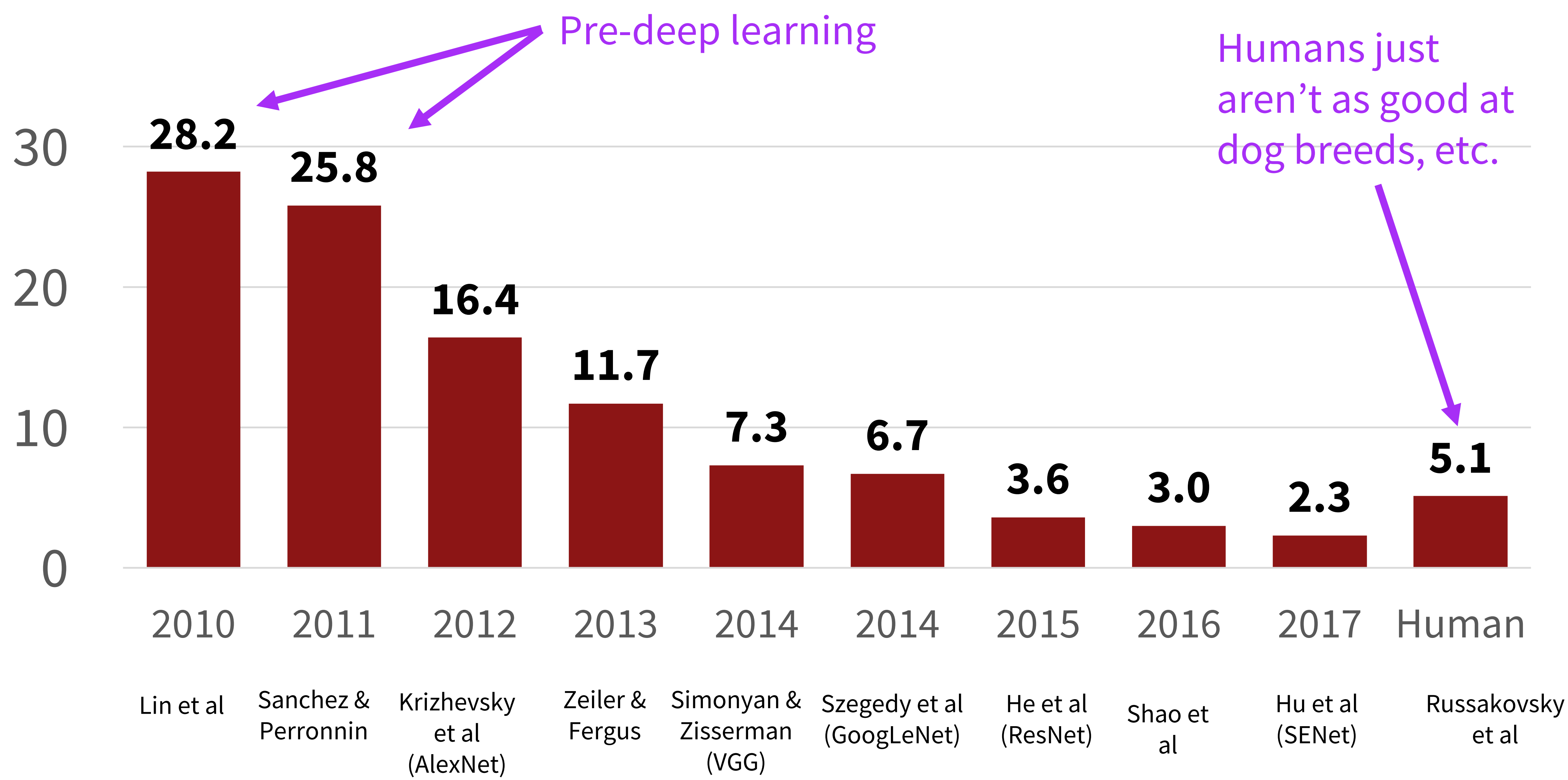
This dramatic breakthrough inaugurated the modern revolution in deep learning.



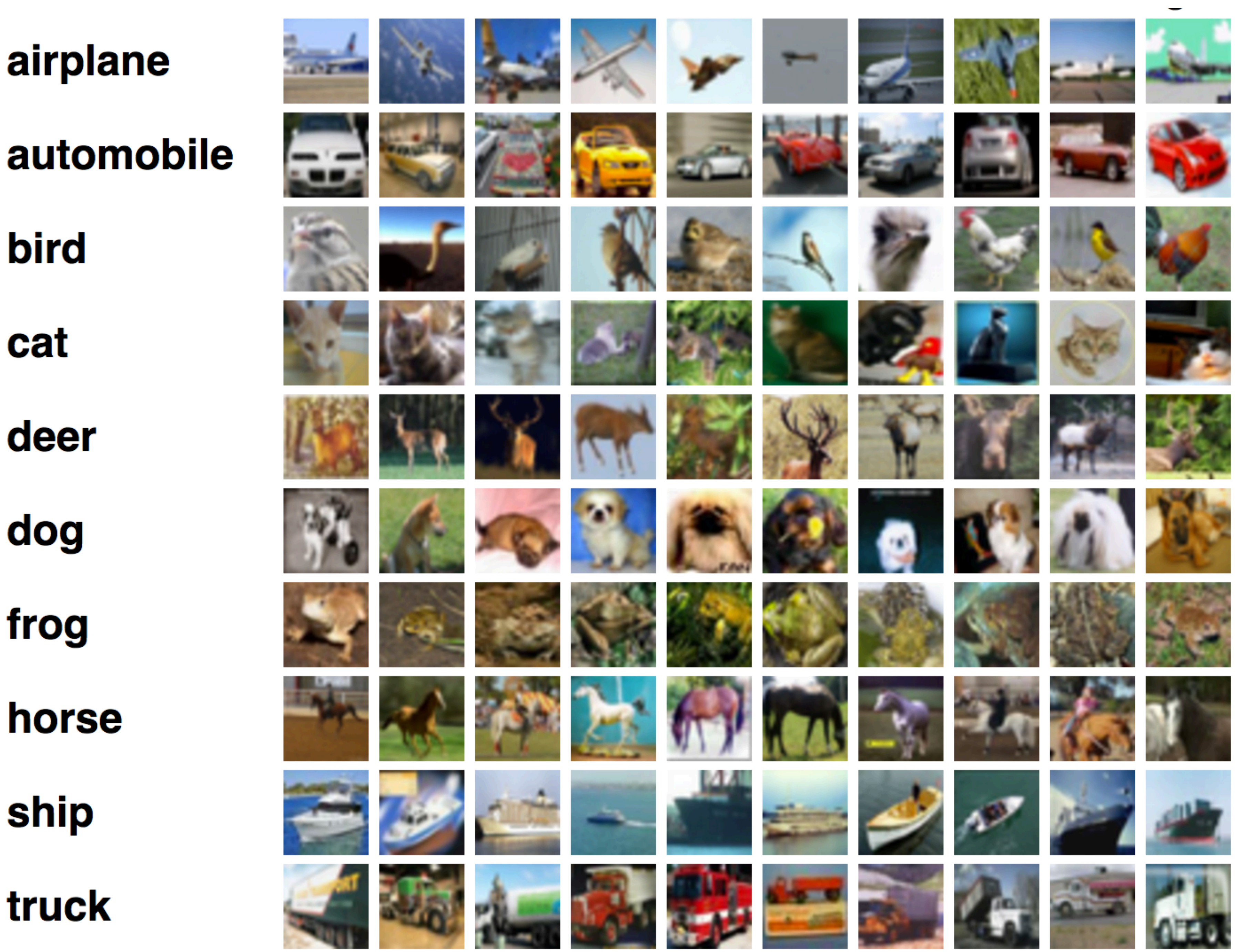
"AlexNet"

# of parameters 1000 x LeNet (60M). Required training on a GPU.

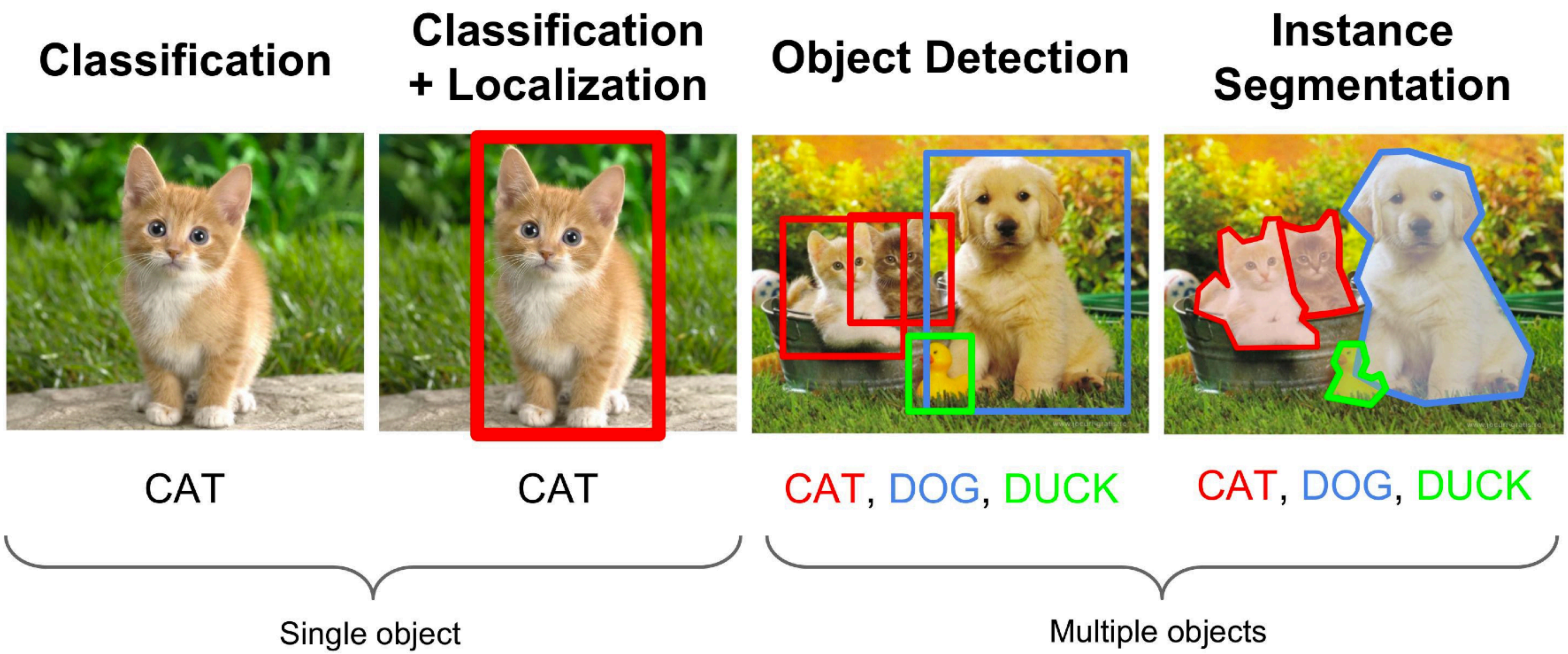
Deep Learning Breakthrough



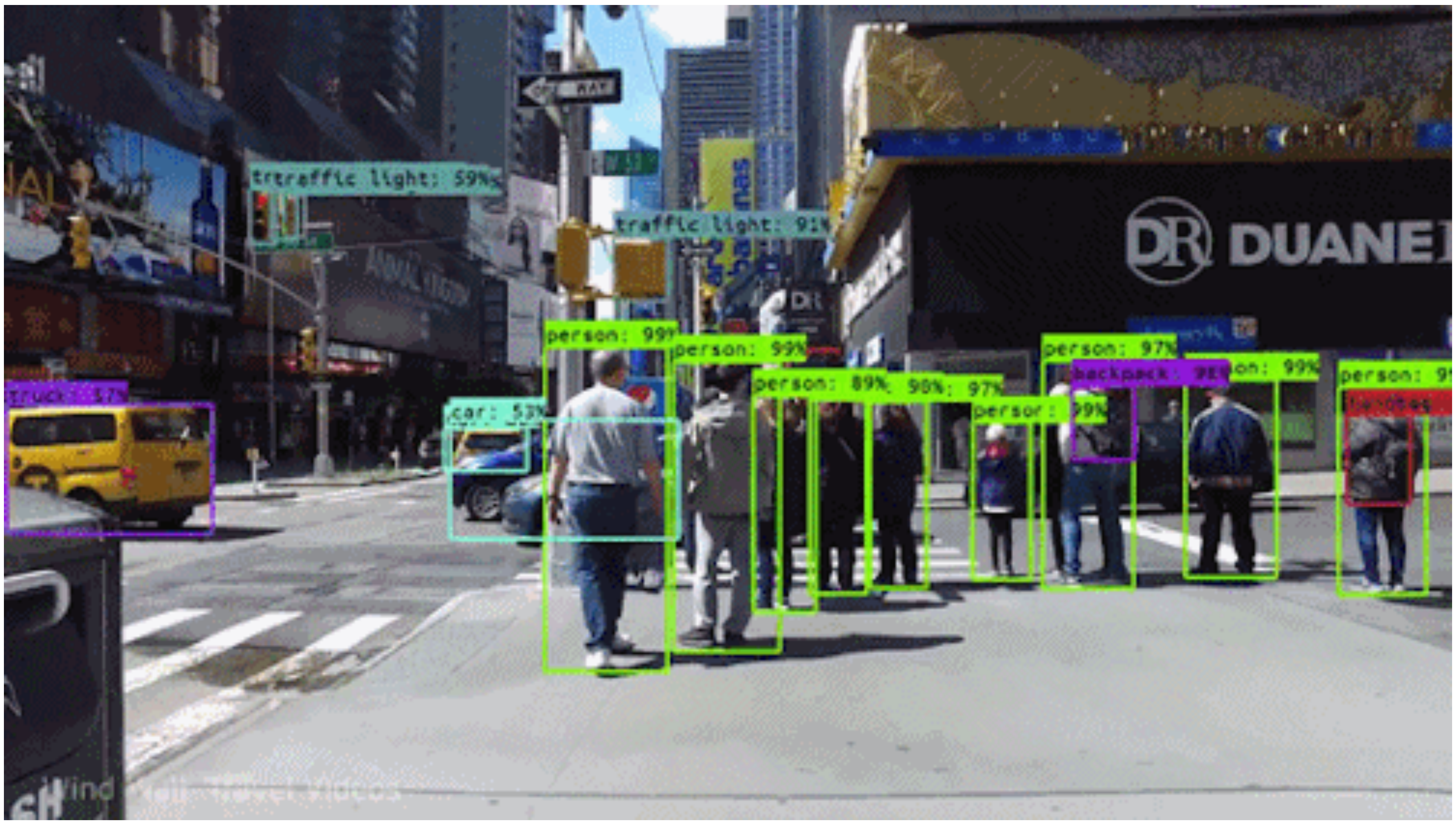
Computer vision



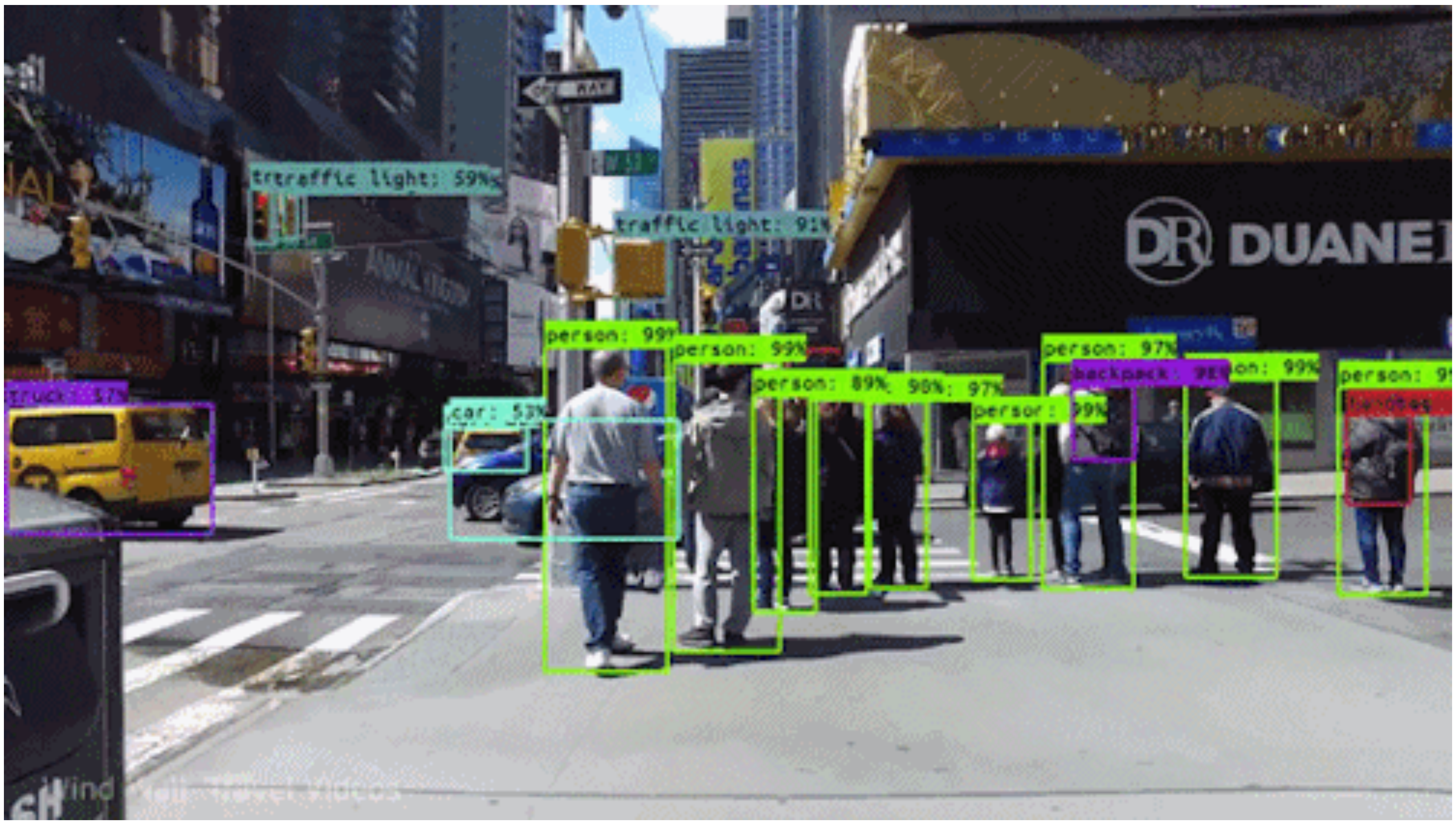
Computer vision



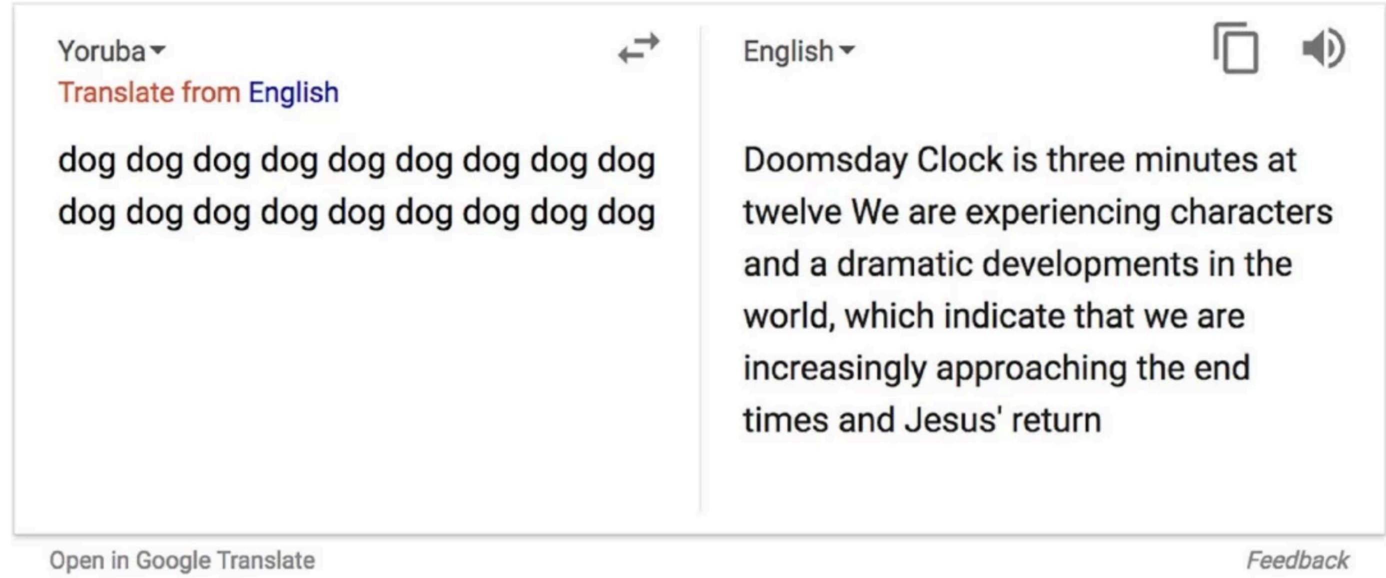
Computer vision



Computer vision



Natural language




Amazon Echo †


Google Home *

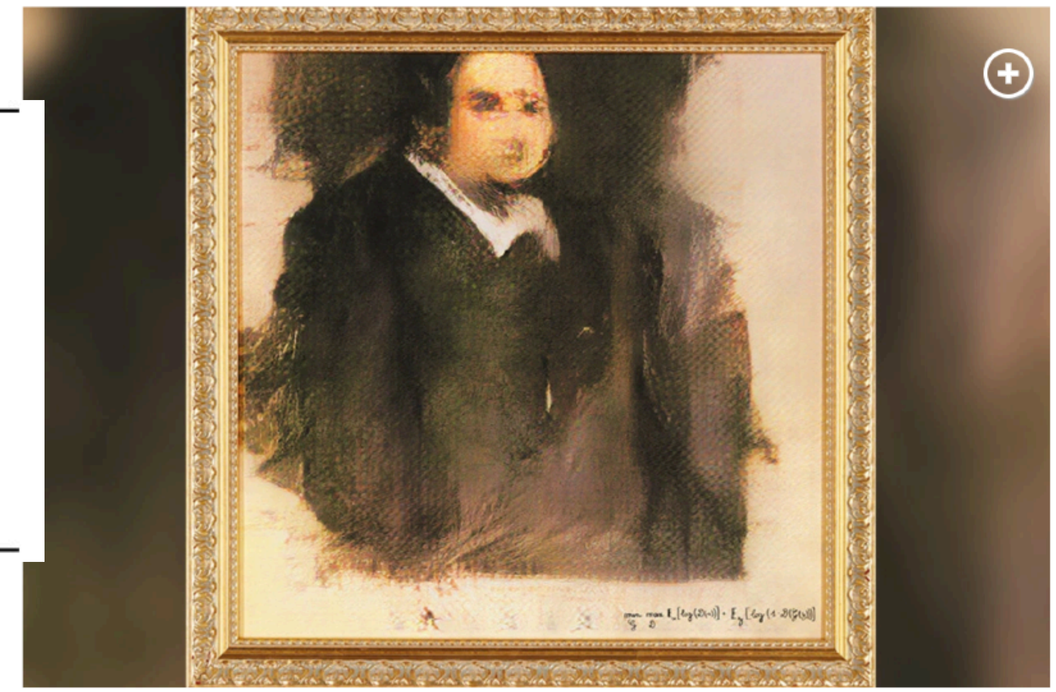

Apple HomePod


Microsoft Invoke

Generative modeling

Painting created by AI sells for stunning $432K

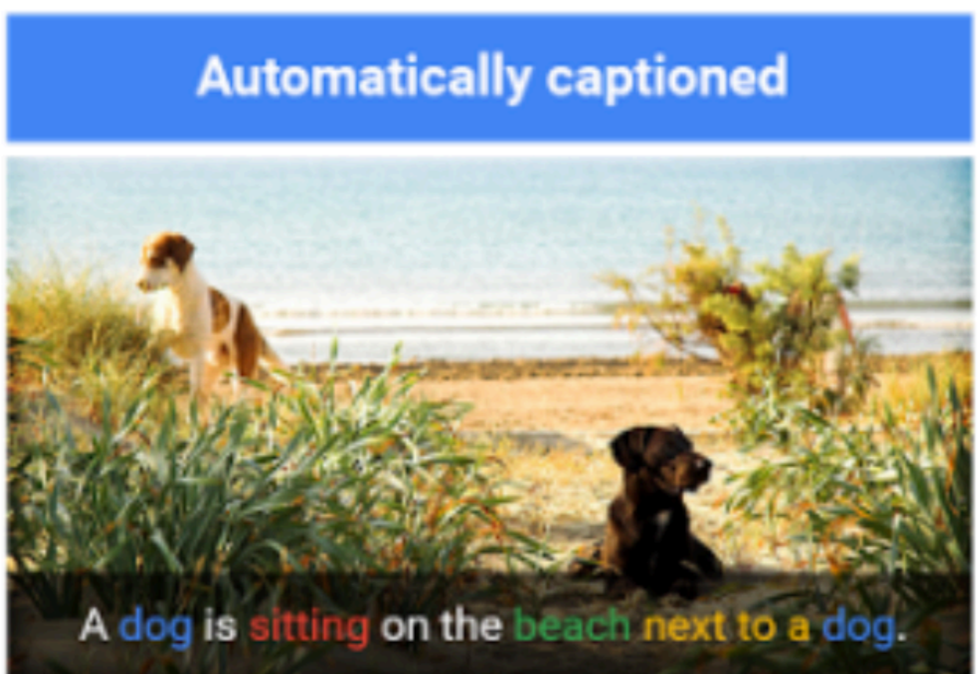
By Tamar Lapin October 26, 2018 | 2:38am | Updated



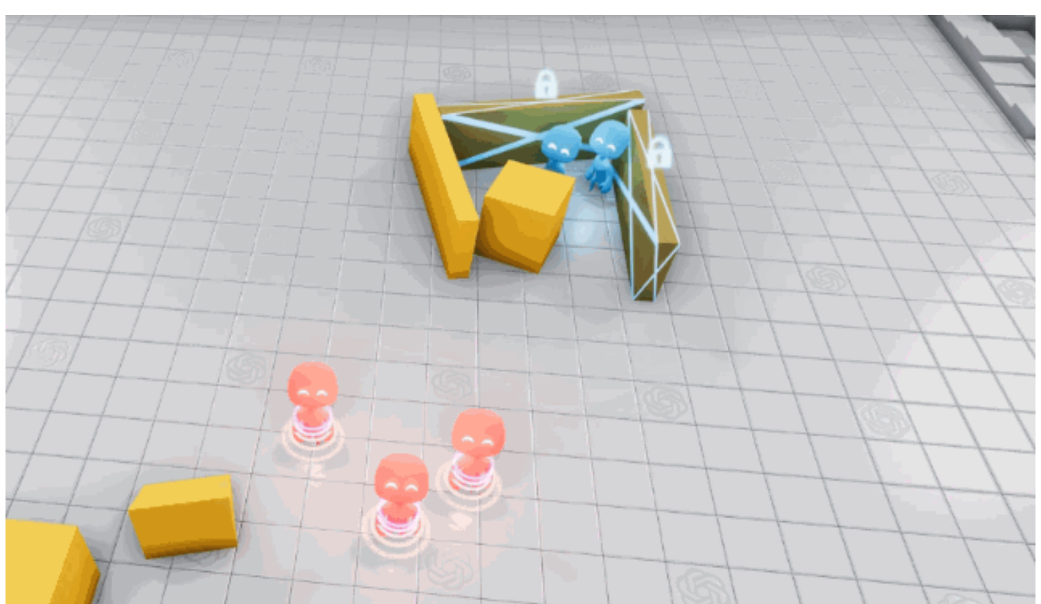
"Portrait of Edmond de Belamy"

AP

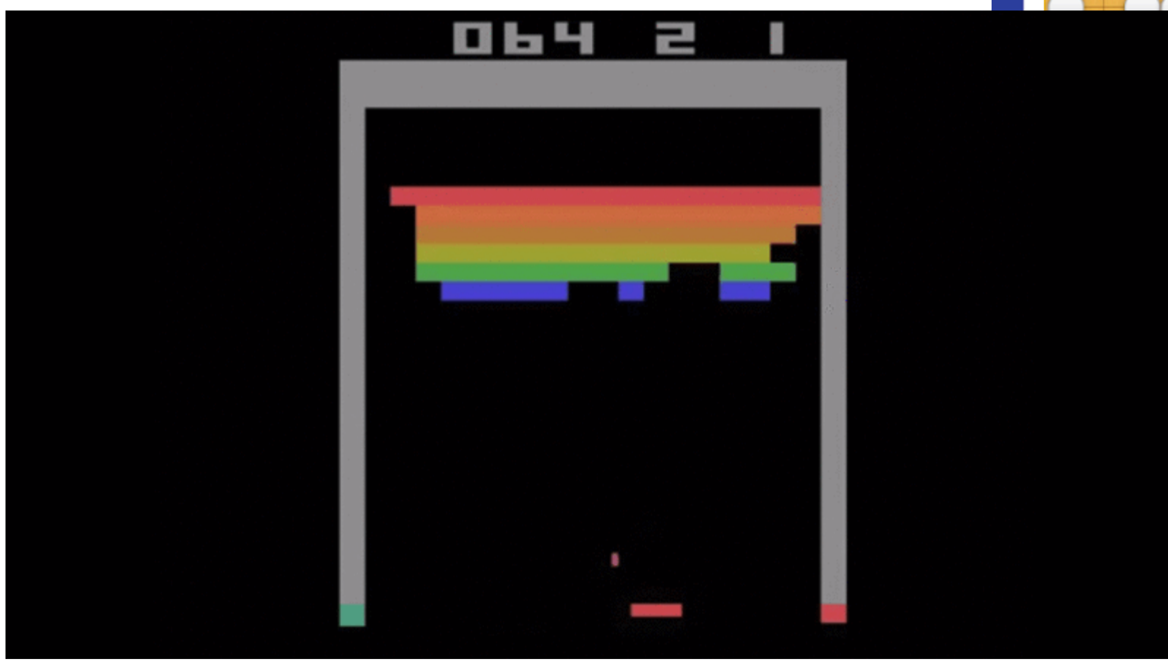
there is no one else in the world .	**this was the only way .**
there is no one else in sight .	*it was the only way .*
they were the only ones who mattered .	*it was her turn to blink .*
they were the only ones left .	*it was hard to tell .*
he had to be with me .	*it was time to move on .*
she had to be with him .	*he had to do it again .*
i had to do this .	*they all looked at each other .*
i wanted to kill him .	*they all turned to look back .*
i started to cry .	*they both turned to face him .*
i turned to him .	**they both turned and walked away .**

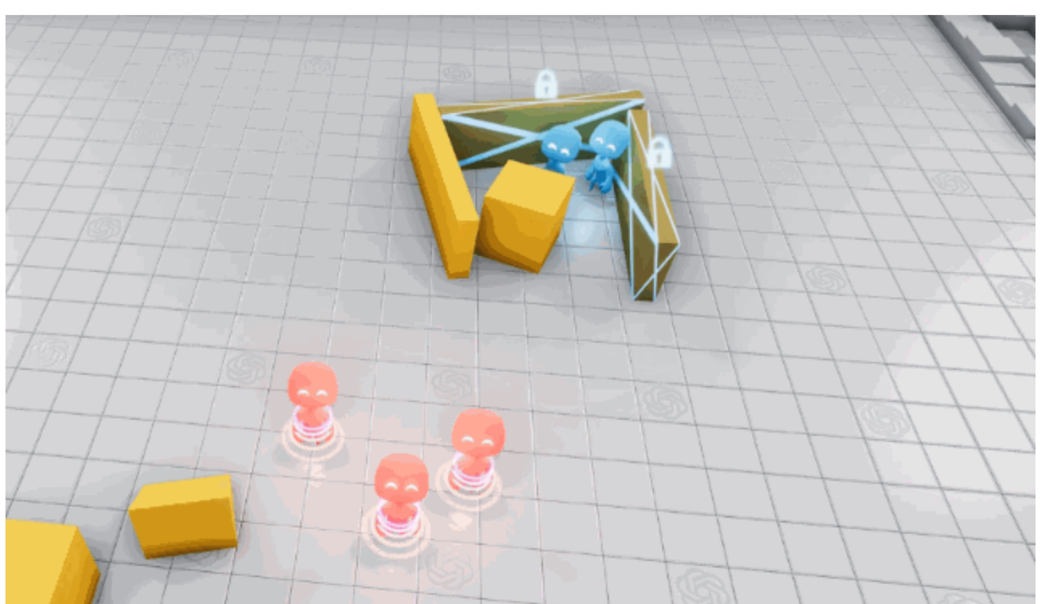


Game playing



Game playing





Plan of the lecture

1. Machine Learning Basics
2. Intro to Deep Learning
3. Deep Learning at the LHC

I will assume most of you know some collider physics.

I will not assume any familiarity with machine learning or neural networks.

1. Machine learning basics

Brief (re)fresher on machine learning

"ML is glorified function fitting"

Want to fit a function $f(x; \theta)$ with some parameters θ ("weights") to a collection of examples $\{x_i\}$ in order to achieve some objective.

Brief (re)fresher on machine learning

"ML is glorified function fitting"

Want to fit a function $f(x; \theta)$ with some parameters θ ("weights") to a collection of examples $\{x_i\}$ in order to achieve some objective.

$x \in \mathbb{R}^d$ is called the feature space

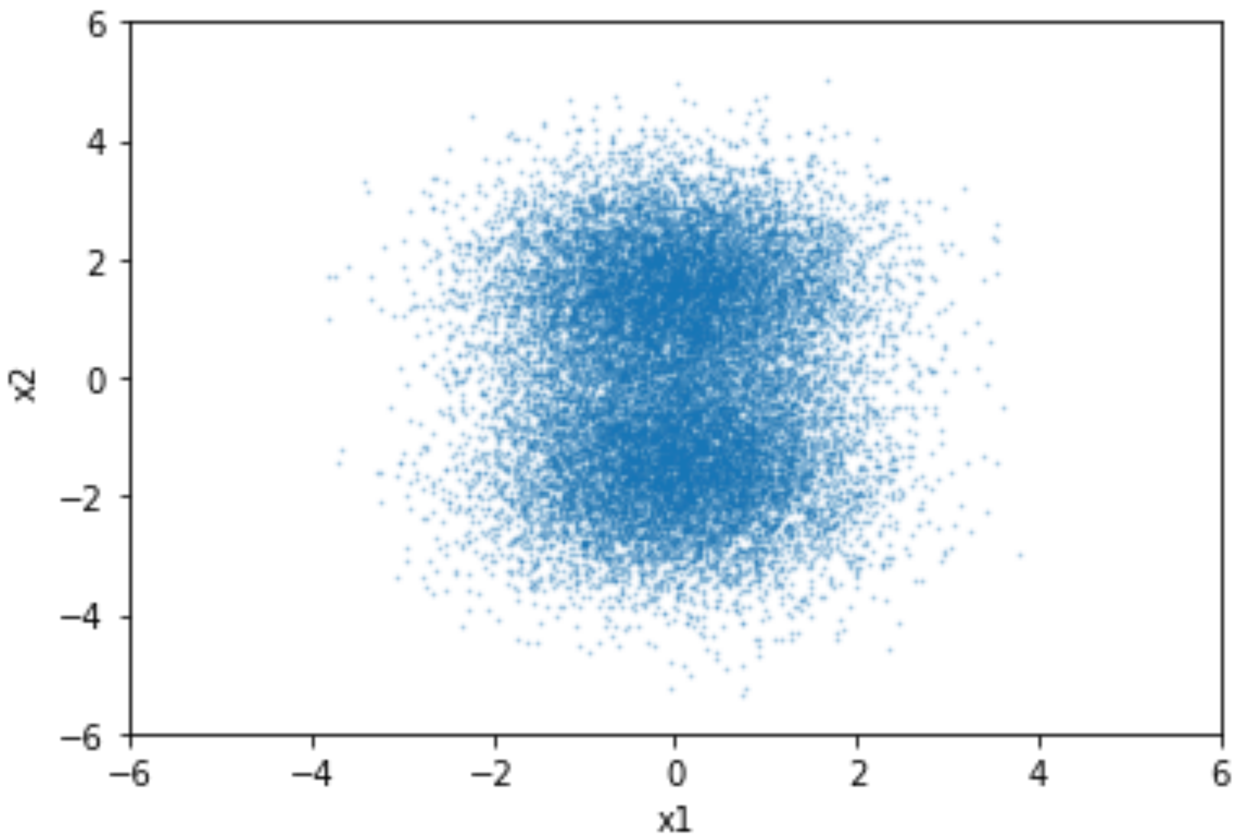
$i=1,\ldots N$ indexes the training dataset.

Brief (re)fresher on machine learning

“ML is glorified function fitting”

Want to fit a function $f(x; \theta)$ with some parameters θ (“weights”) to a collection of examples $\{x_i\}$ in order to achieve some objective.

Some typical objectives:

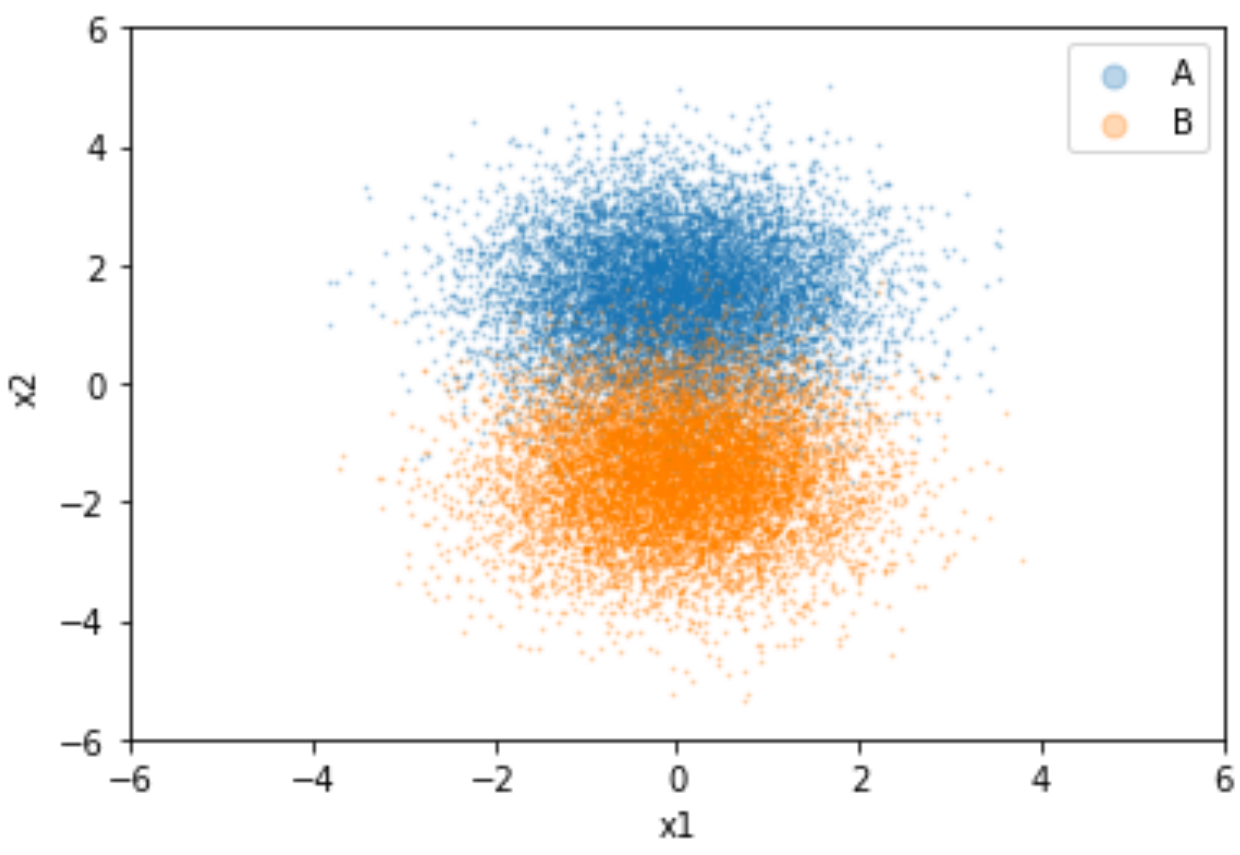


Brief (re)fresher on machine learning

"ML is glorified function fitting"

Want to fit a function $f(x; \theta)$ with some parameters θ ("weights") to a collection of examples $\{x_i\}$ in order to achieve some objective.

Some typical objectives:



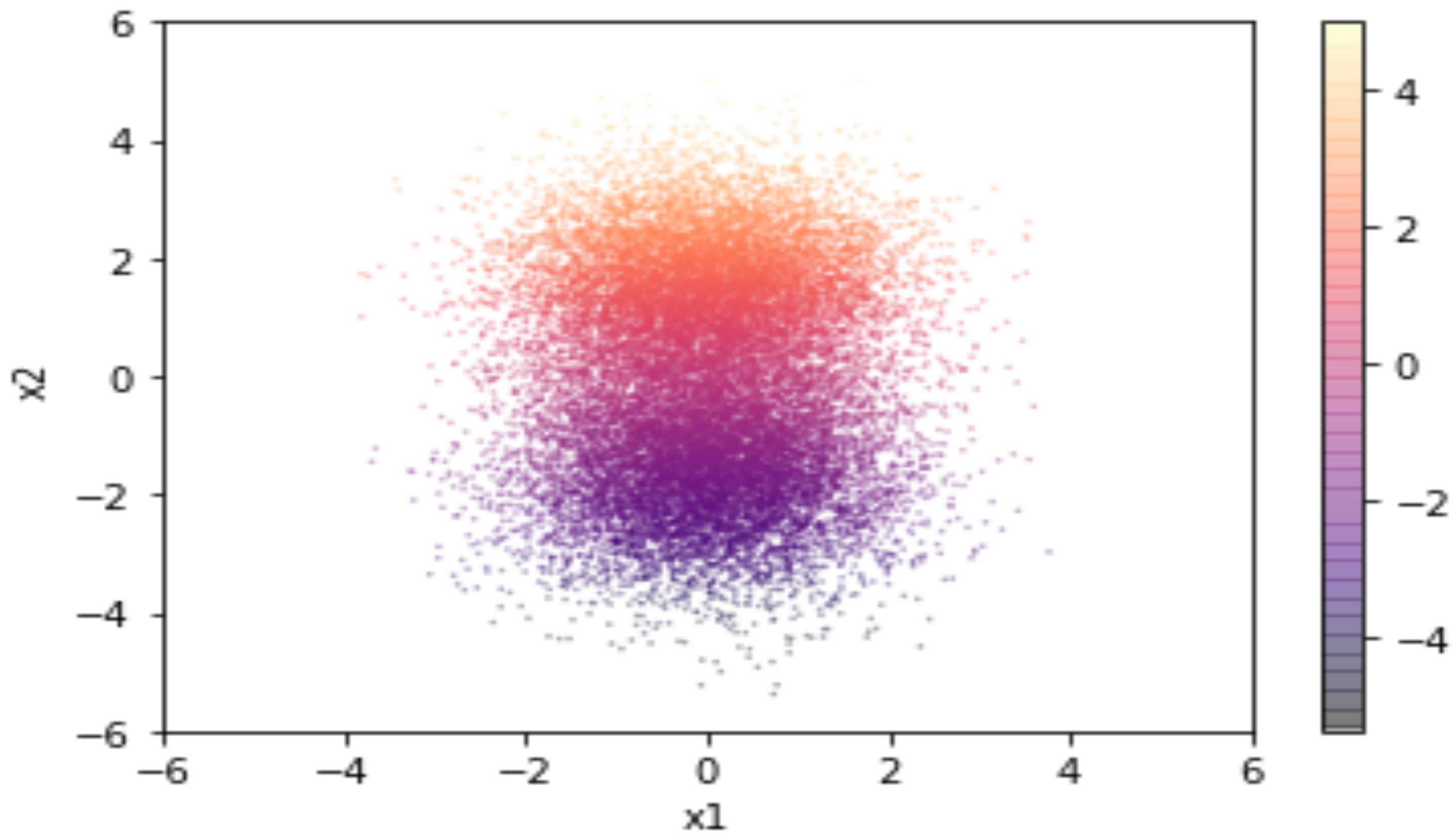
- Classification

Brief (re)fresher on machine learning

"ML is glorified function fitting"

Want to fit a function $f(x; \theta)$ with some parameters θ ("weights") to a collection of examples $\{x_i\}$ in order to achieve some objective.

Some typical objectives:



- Classification
- Regression

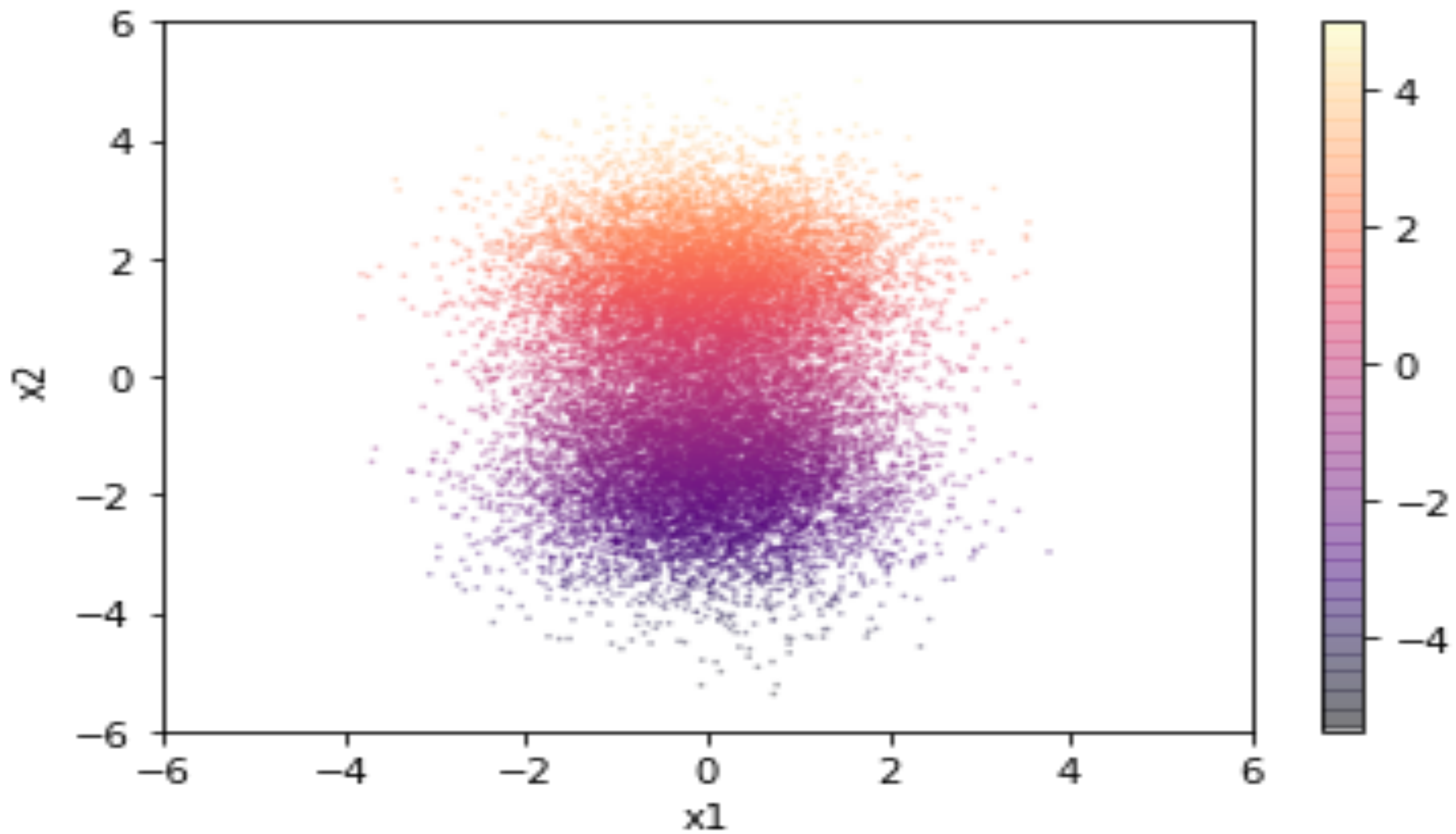
Brief (re)fresher on machine learning

"ML is glorified function fitting"

Want to fit a function $f(x; \theta)$ with some parameters θ ("weights") to a collection of examples $\{x_i\}$ in order to achieve some objective.

Some typical objectives:

"supervised ML"



- Classification
- Regression

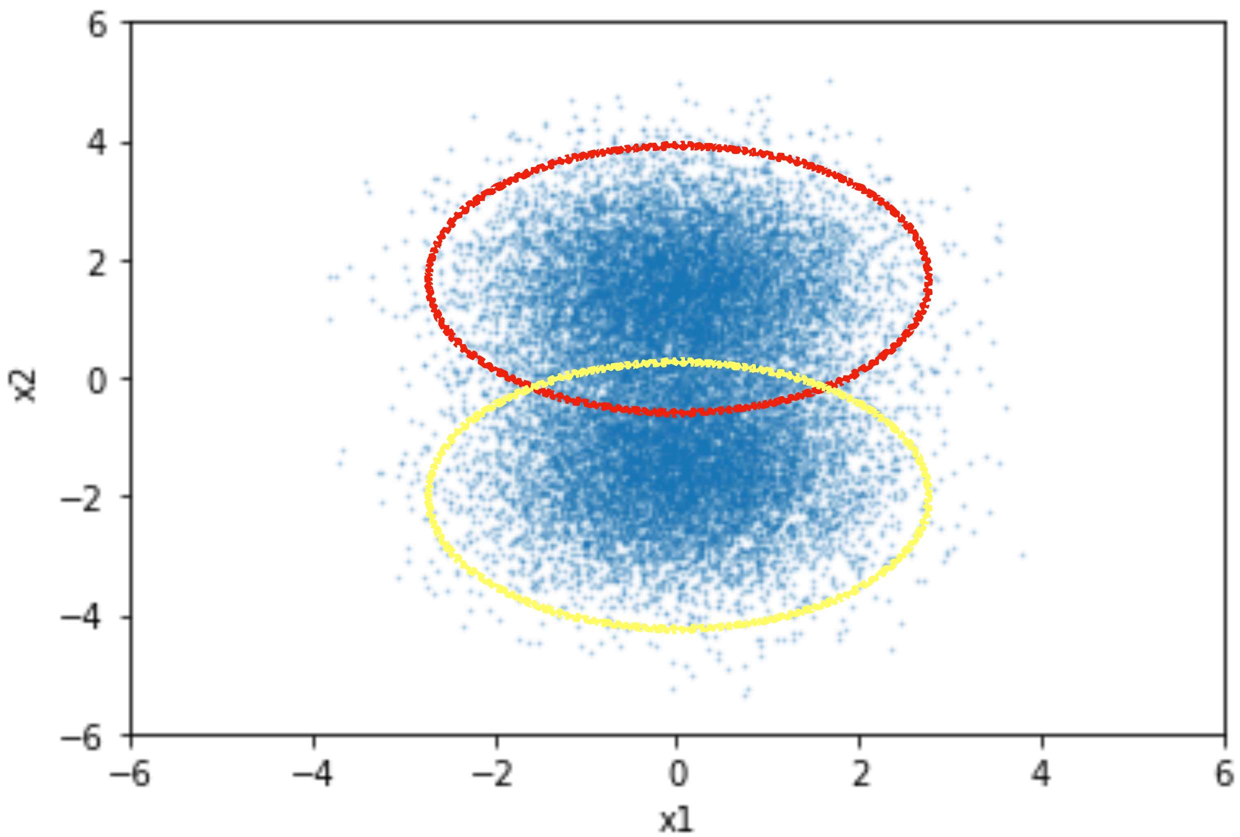
Brief (re)fresher on machine learning

"ML is glorified function fitting"

Want to fit a function $f(x; \theta)$ with some parameters θ ("weights") to a collection of examples $\{x_i\}$ in order to achieve some objective.

Some typical objectives:

"supervised ML"



- Classification
- Regression
- Clustering

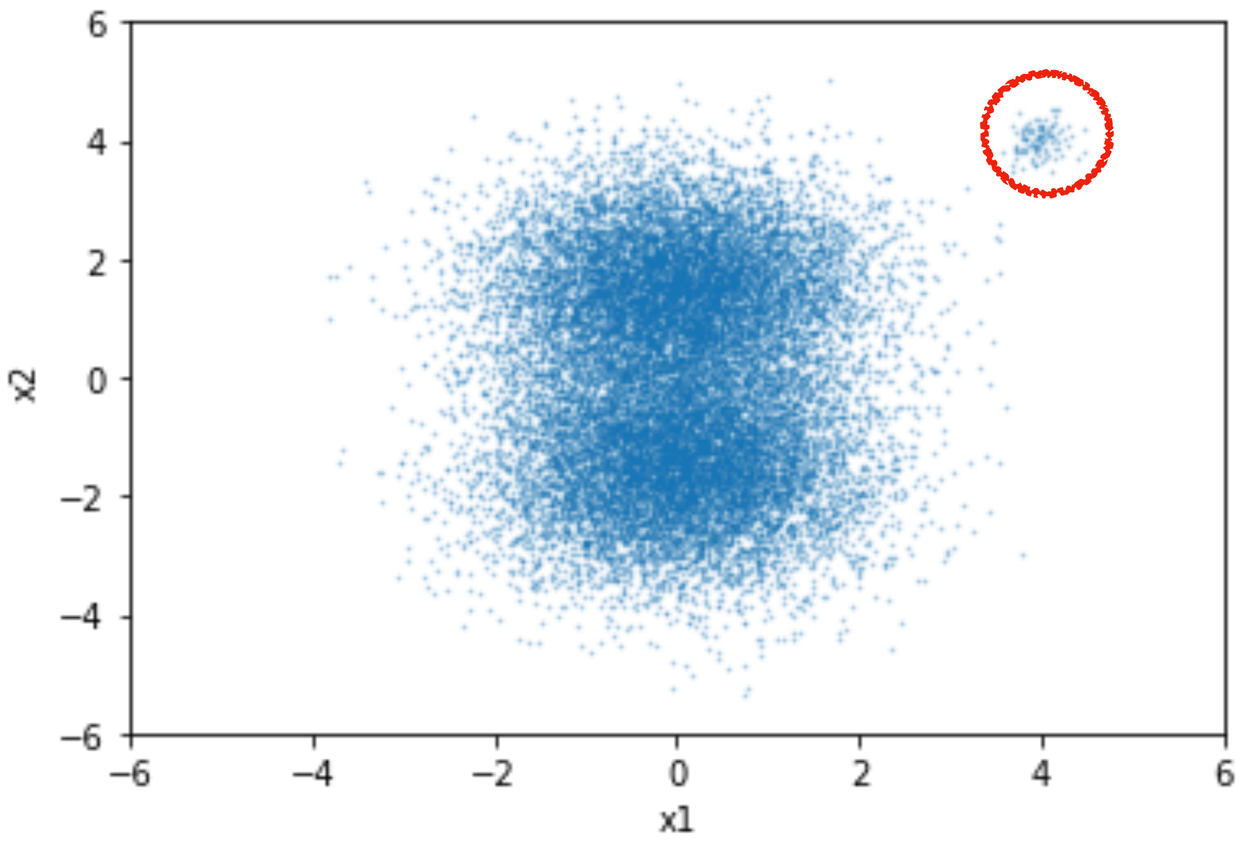
Brief (re)fresher on machine learning

"ML is glorified function fitting"

Want to fit a function $f(x; \theta)$ with some parameters θ ("weights") to a collection of examples $\{x_i\}$ in order to achieve some objective.

Some typical objectives:

"supervised ML"



- Classification
- Regression
- Clustering
- Anomaly detection

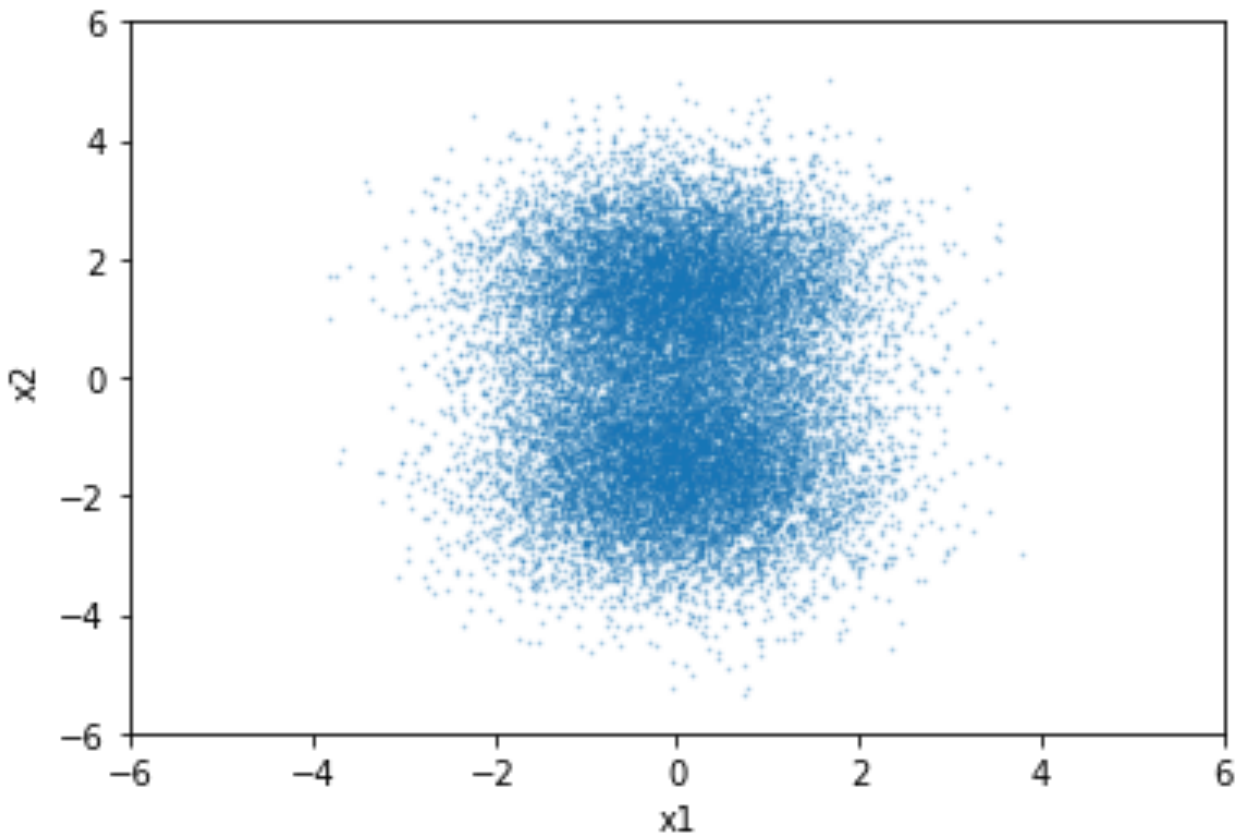
Brief (re)fresher on machine learning

"ML is glorified function fitting"

Want to fit a function $f(x; \theta)$ with some parameters θ ("weights") to a collection of examples $\{x_i\}$ in order to achieve some objective.

Some typical objectives:

"supervised ML"



- Classification
- Regression
- Clustering
- Anomaly detection
- Density estimation

$$P(x) = 0.5 * \mathcal{N}(x|\mu = (0, -1.5), \sigma = (1, 1)) \\ + 0.5 * \mathcal{N}(x|\mu = (0, +1.5), \sigma = (1, 1))$$

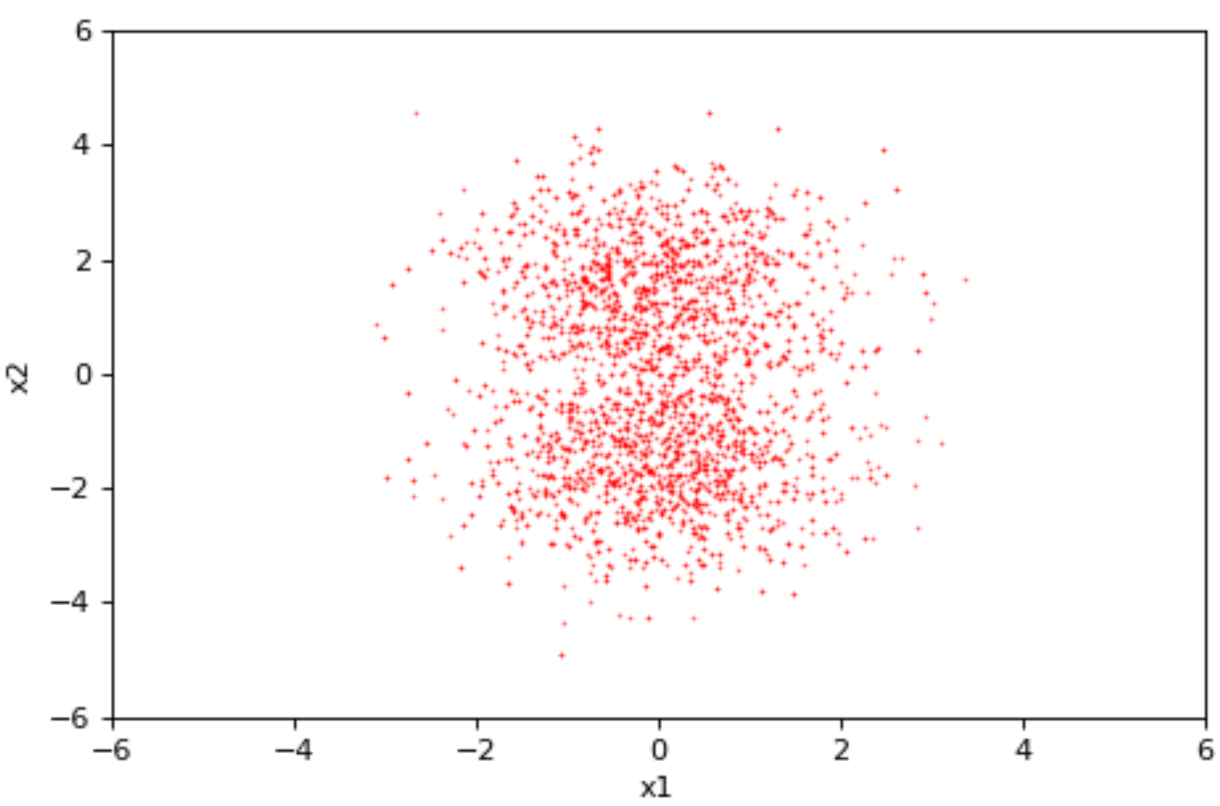
Brief (re)fresher on machine learning

"ML is glorified function fitting"

Want to fit a function $f(x; \theta)$ with some parameters θ ("weights") to a collection of examples $\{x_i\}$ in order to achieve some objective.

Some typical objectives:

"supervised ML"



- Classification
- Regression
- Clustering
- Anomaly detection
- Density estimation
- Generation

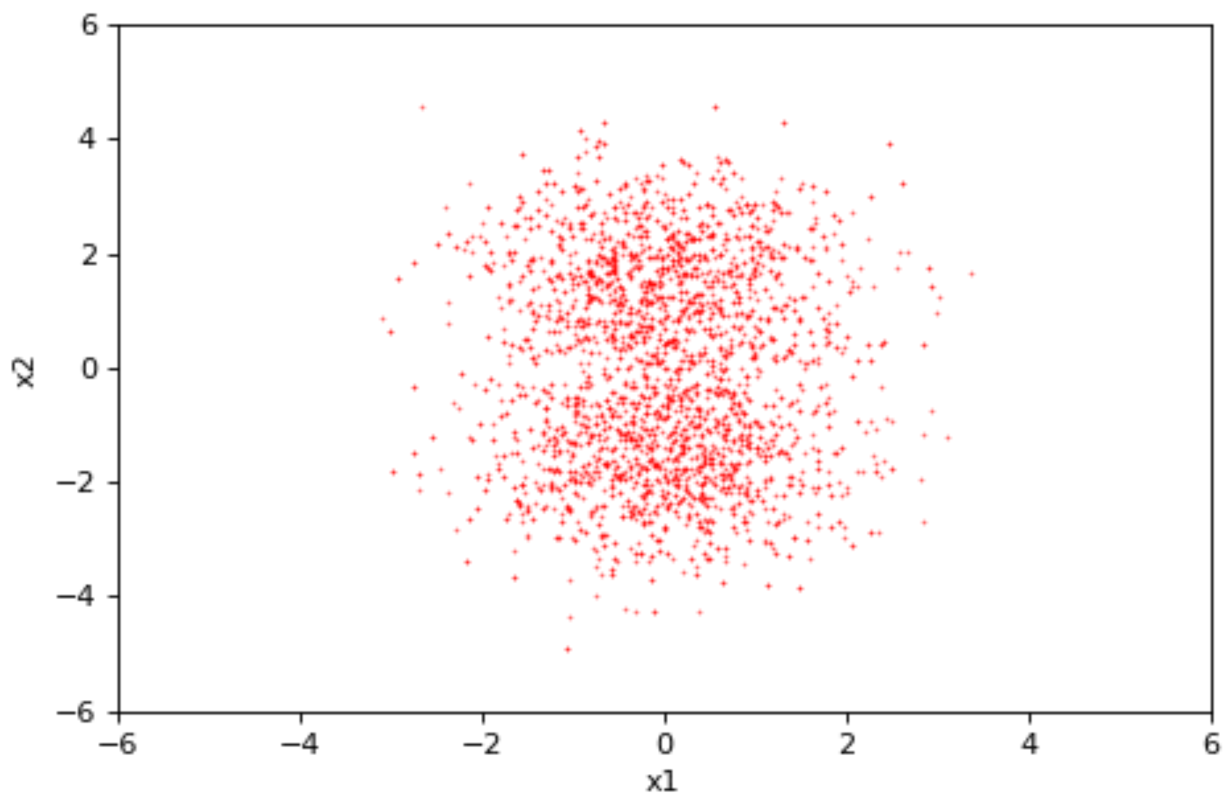
Brief (re)fresher on machine learning

"ML is glorified function fitting"

Want to fit a function f(x; θ) with some parameters θ ("weights") to a collection of examples $\{x_i\}$ in order to achieve some objective.

Some typical objectives:

"supervised ML"



- Classification
- Regression
- Clustering
- Anomaly detection
- Density estimation
- Generation

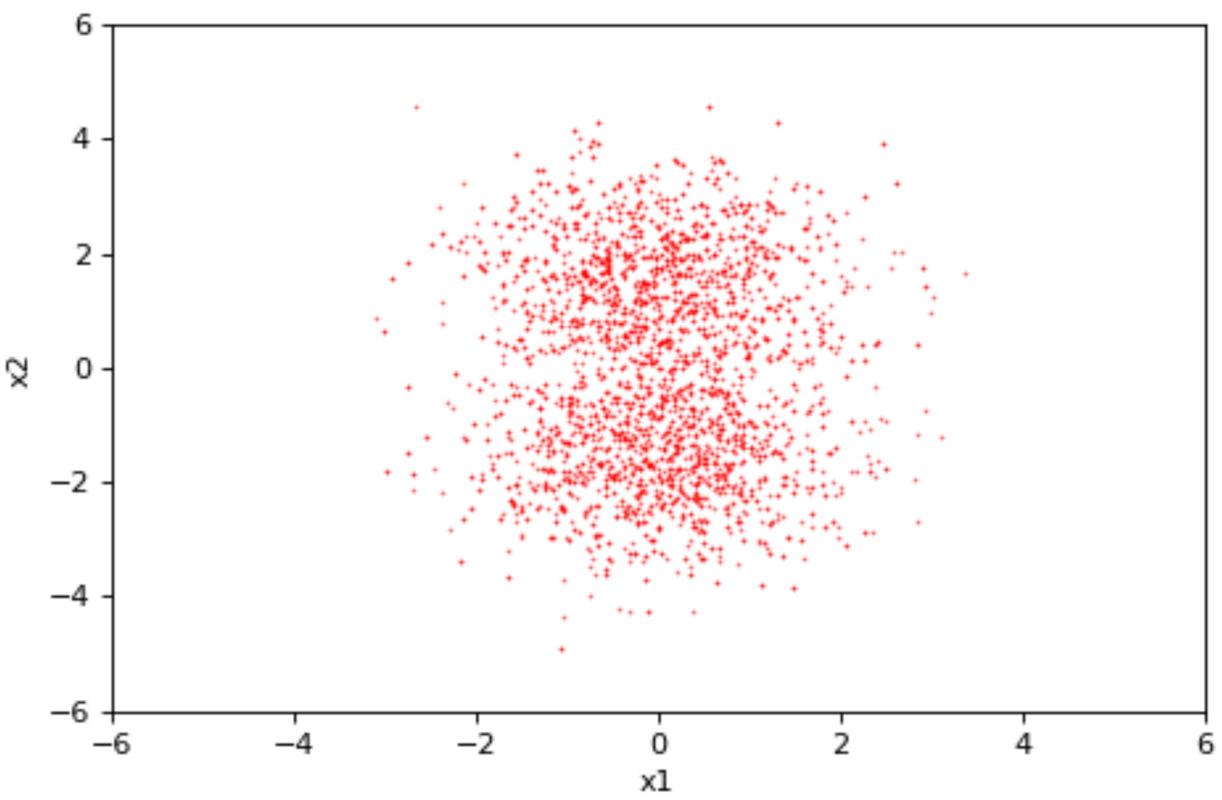
Brief (re)fresher on machine learning

"ML is glorified function fitting"

Want to fit a function $f(x; \theta)$ with some parameters θ ("weights") to a collection of examples $\{x_i\}$ in order to achieve some objective.

Some typical objectives:

"supervised ML"

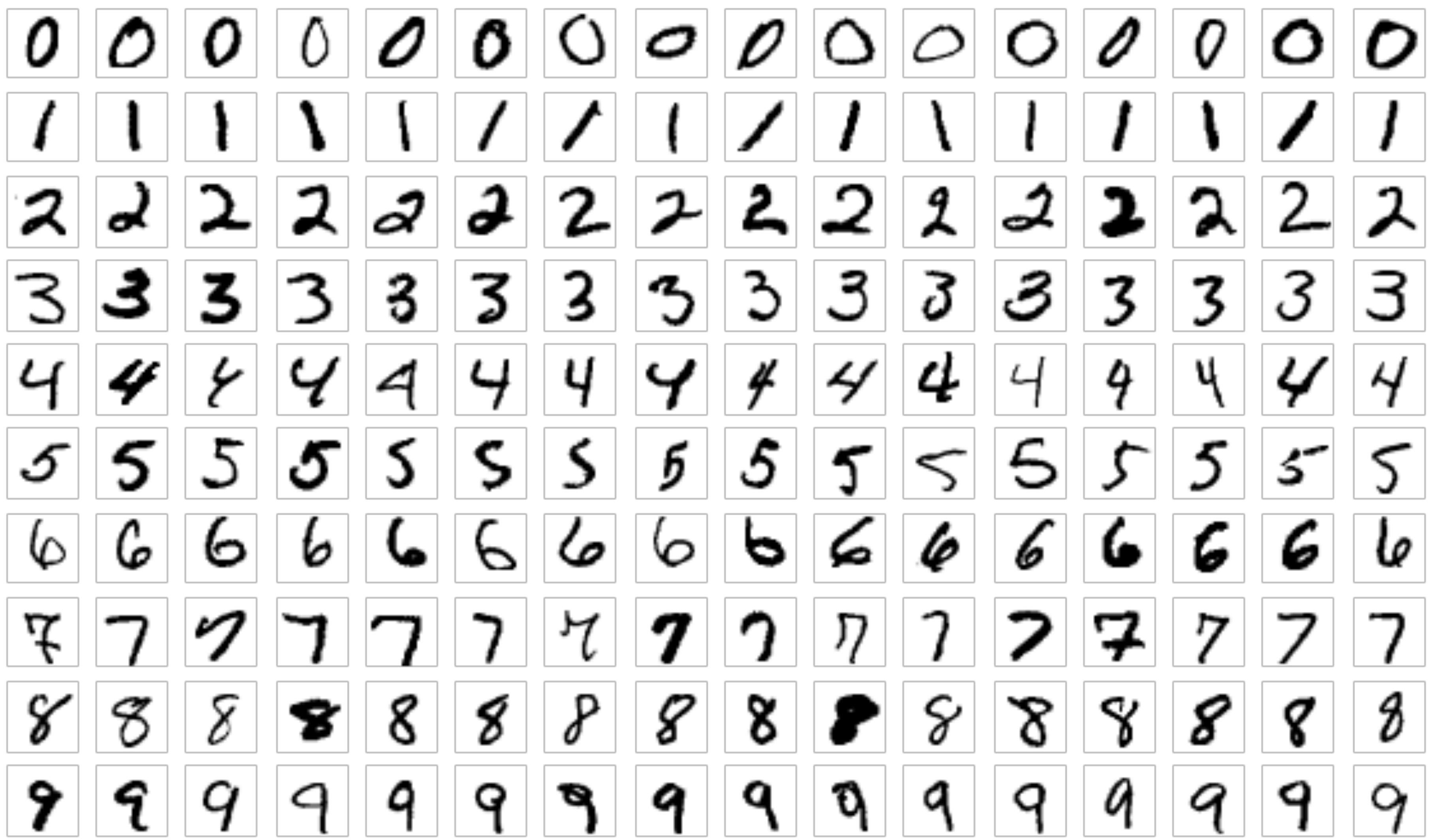


- Classification
- Regression
- Clustering
- Anomaly detection
- Density estimation
- Generation

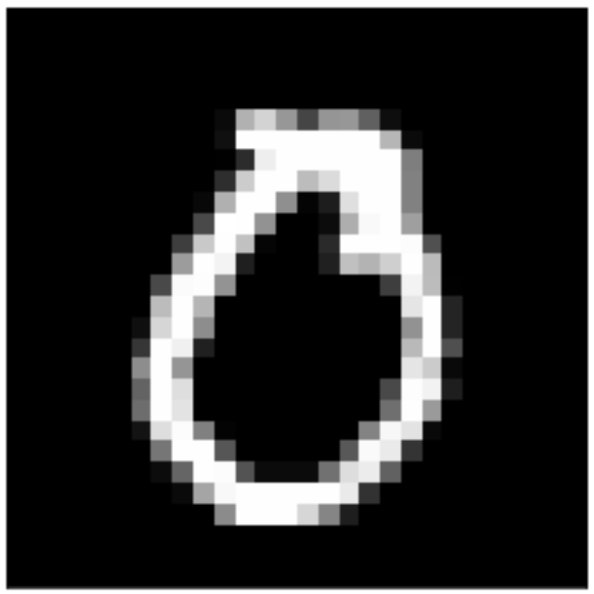
"unsupervised ML"

MNIST example

image database of 70,000 handwritten digits



MNIST example



x

Input: handwritten digit image

28x28 pixel intensities

from MNIST database

θ

parameters of fitting function

$$\begin{pmatrix} P_0(x;\theta) \\ P_1(x;\theta) \\ \vdots \\ P_9(x;\theta) \end{pmatrix}$$

$$\sum_i P_i(x;\theta) = 1$$

Output: probability it's a 0, 1, …,9

Brief (re)fresher on machine learning

"ML is glorified function fitting"

Brief (re)fresher on machine learning

"ML is glorified function fitting"

Want to fit a function $f(x; \theta)$ with some parameters θ ("weights") to a collection of examples $\{x_i\}$ in order to achieve some objective.

Brief (re)fresher on machine learning

"ML is glorified function fitting"

Want to fit a function $f(x; \theta)$ with some parameters θ ("weights") to a collection of examples $\{x_i\}$ in order to achieve some objective.

How to perform the fit?

Brief (re)fresher on machine learning

"ML is glorified function fitting"

Want to fit a function $f(x; \theta)$ with some parameters θ ("weights") to a collection of examples $\{x_i\}$ in order to achieve some objective.

How to perform the fit?

Minimize a loss function! Loss function quantifies how well the objective has been achieved.

Brief (re)fresher on machine learning

"ML is glorified function fitting"

Want to fit a function $f(x; \theta)$ with some parameters θ ("weights") to a collection of examples $\{x_i\}$ in order to achieve some objective.

How to perform the fit?

Minimize a loss function! Loss function quantifies how well the objective has been achieved.

$\sum_{(x,y)} L(P(x;\theta), y)$	$\sum_{x} L(P(x;\theta))$
Data is labeled	Data is not labeled
supervised ML	unsupervised ML

Examples of loss functions

$$L = (P(x;\theta) - y)^2$$

"mean-squared error"
used for regression

$$L = -\Big(y \log P(x;\theta) + (1-y)\log(1 - P(x;\theta))\Big)$$

"binary cross entropy"
used for binary classification

$$L = -y_j \log P_j(x;\theta)$$

"categorical cross entropy"
used for multi-class classification

Examples of loss functions

$$L = (P(x;\theta) - y)^2$$

"mean-squared error"
used for regression

$$L = -\Big(y \log P(x;\theta) + (1-y)\log(1 - P(x;\theta))\Big)$$

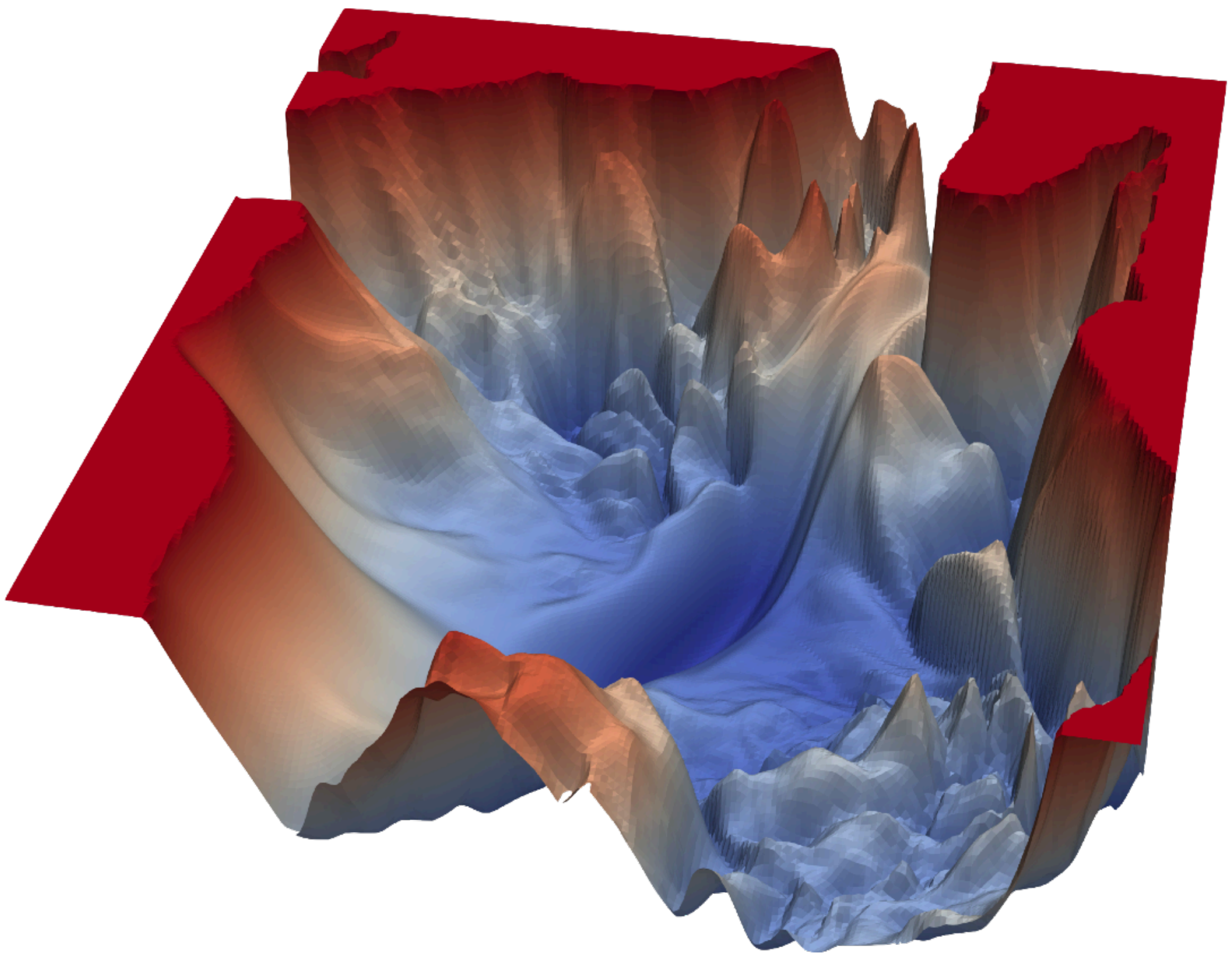
"binary cross entropy"
used for binary classification

$$L = -y_j \log P_j(x;\theta)$$

MNIST example:
labels "one-hot encoding"
0 → [1,0,…,0]
1 → [0,1,…,0]
…
9 → [0,0,…,1]

"categorical cross entropy"
used for multi-class classification

Minimizing the loss function



Highly nonconvex function over a many-dimensional space.
Many local minima.

Gradient descent

$$\langle L \rangle(\theta) \equiv \sum_{(x,y)} L(P(x;\theta), y)$$

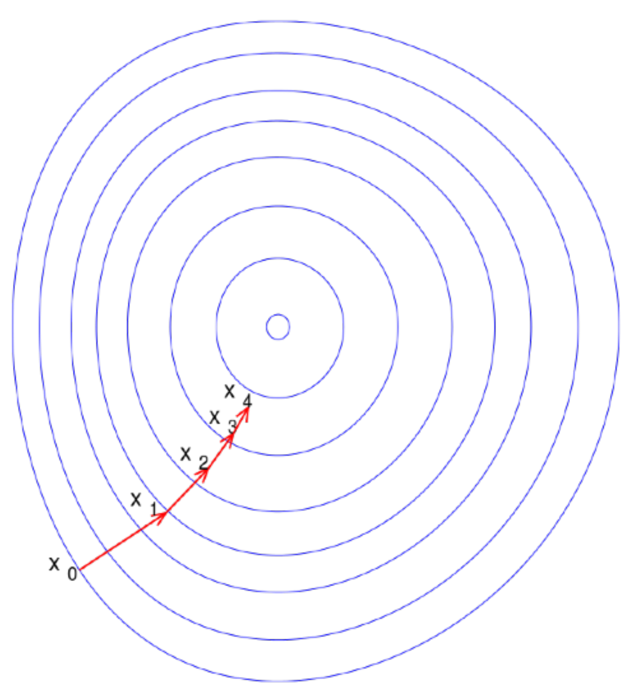
Want to minimize wrt θ.

Obvious idea: $\theta \to \theta - \alpha\, \partial_\theta \langle L \rangle(\theta)$

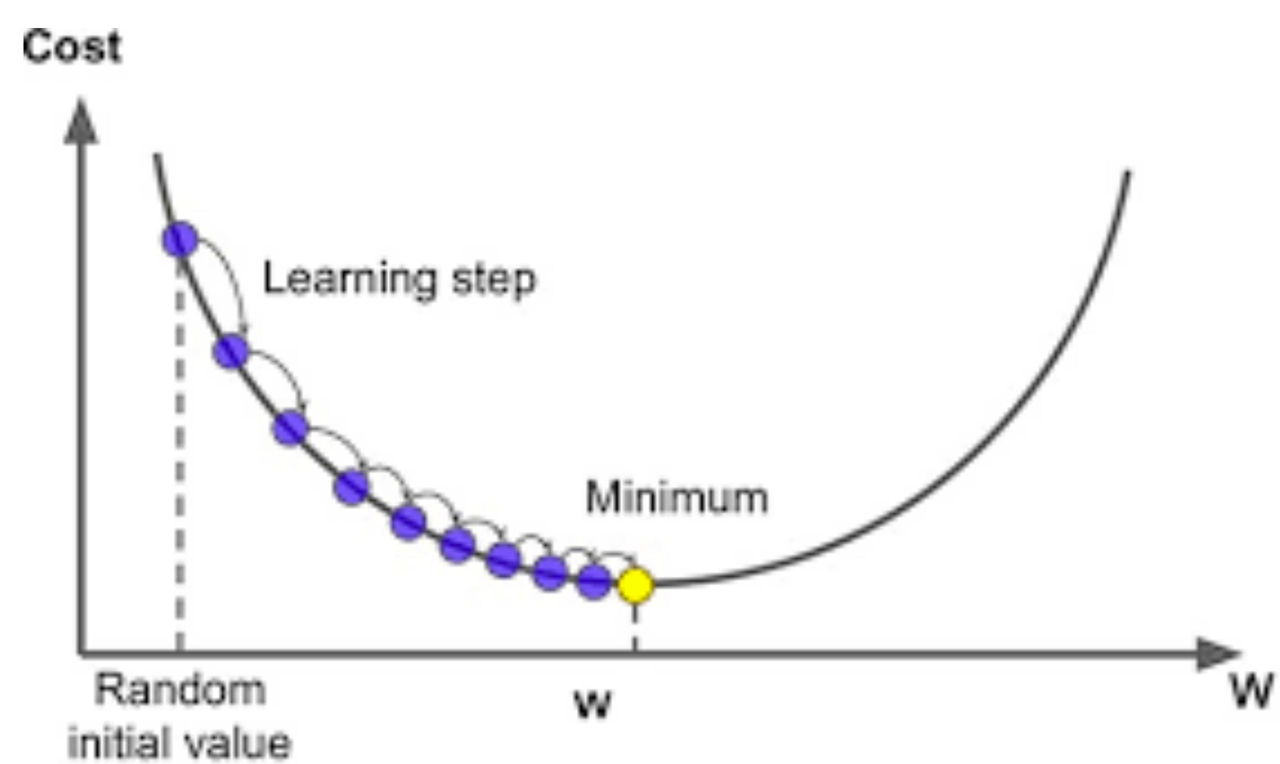
α : "learning rate"

"gradient descent method"

(generalization of Newton-Raphson method)



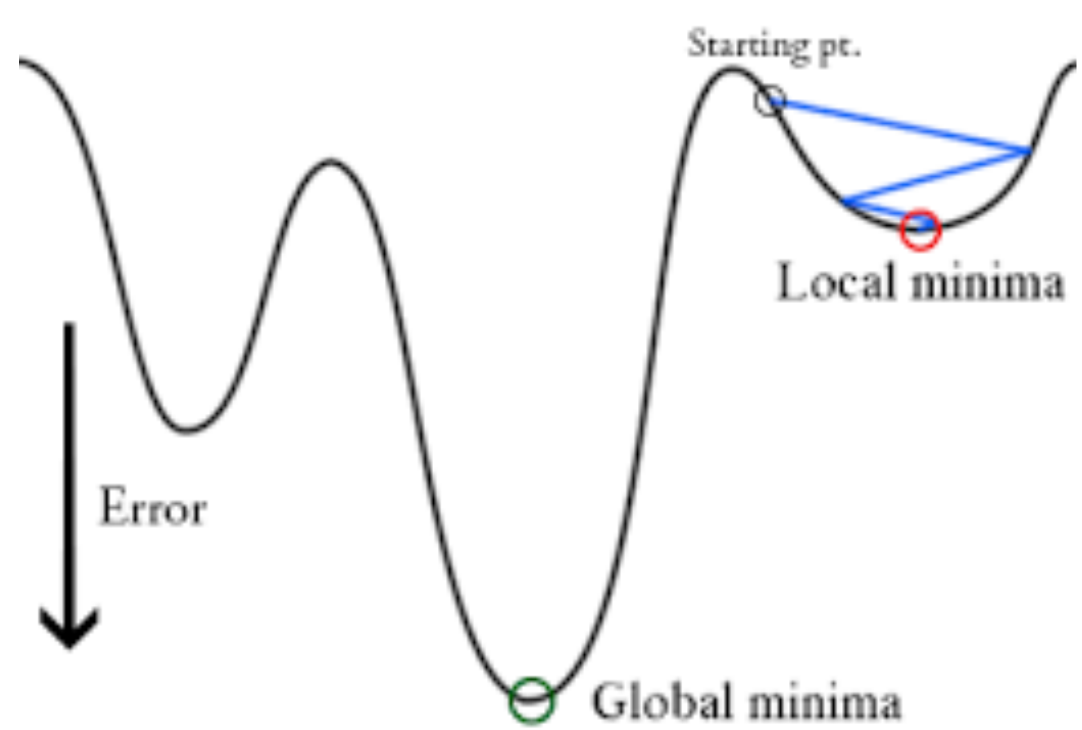
source: Wikipedia



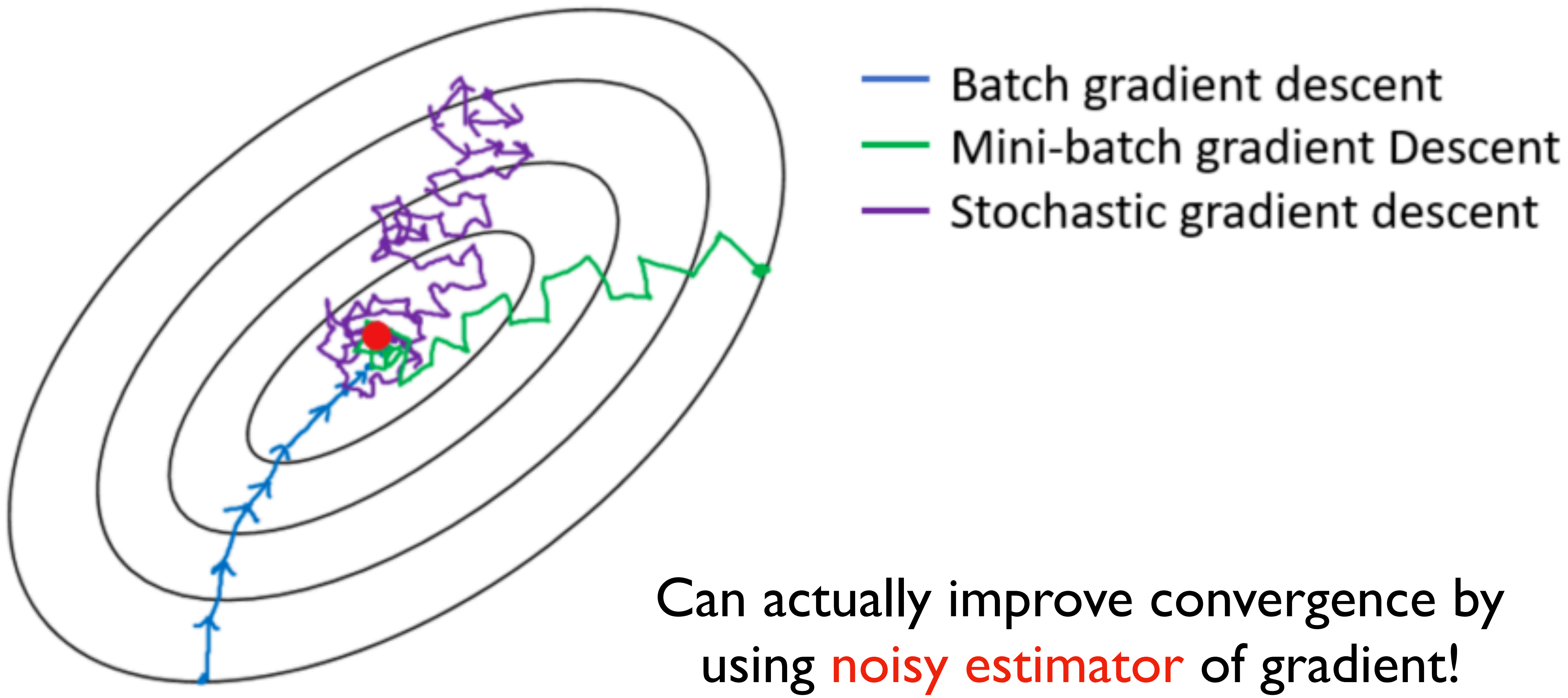
Gradient descent: problems

Downsides to gradient descent:

- average over full dataset <L(θ)> can be expensive to compute
- poor initial guess or learning rate can lead to becoming stuck in poor local minimum



Stochastic Gradient Descent



Can actually improve convergence by using noisy estimator of gradient!

$$\sum_{(x,y)\in\text{full dataset}} L(P(x;\theta), y) \longrightarrow \sum_{(x,y)\in\text{minibatch}} L(P(x;\theta), y)$$

Stochastic Gradient Descent

Method:

1. Divide up training data into minibatches.

2. Update weights minibatch by minibatch

$$\theta \to \theta - \alpha\, \partial_\theta \langle L \rangle(\theta)$$

(average computed on each minibatch)

3. Repeat until convergence.

minibatch 1
minibatch 2
minibatch 3
…

Stochastic Gradient Descent

Method:

1. Divide up training data into minibatches.

2. Update weights minibatch by minibatch

$$\theta \to \theta - \alpha\, \partial_\theta \langle L \rangle (\theta)$$

(average computed on each minibatch)

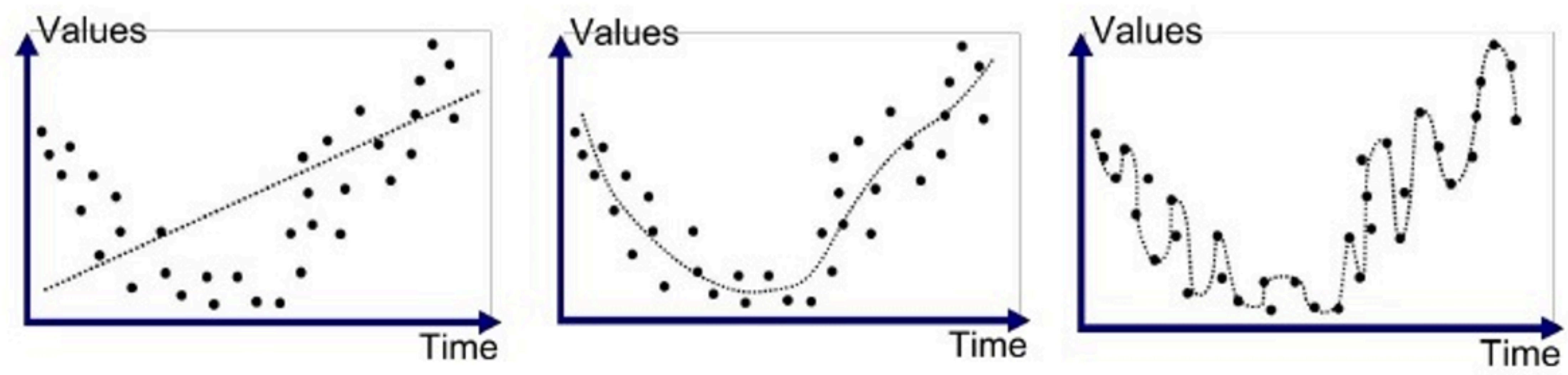
3. Repeat until convergence.

“one epoch”

minibatch 1
minibatch 2
minibatch 3
…

Overfitting

The fitting function (especially if it is a neural network) may be overparametrized. So overfitting is a major concern.

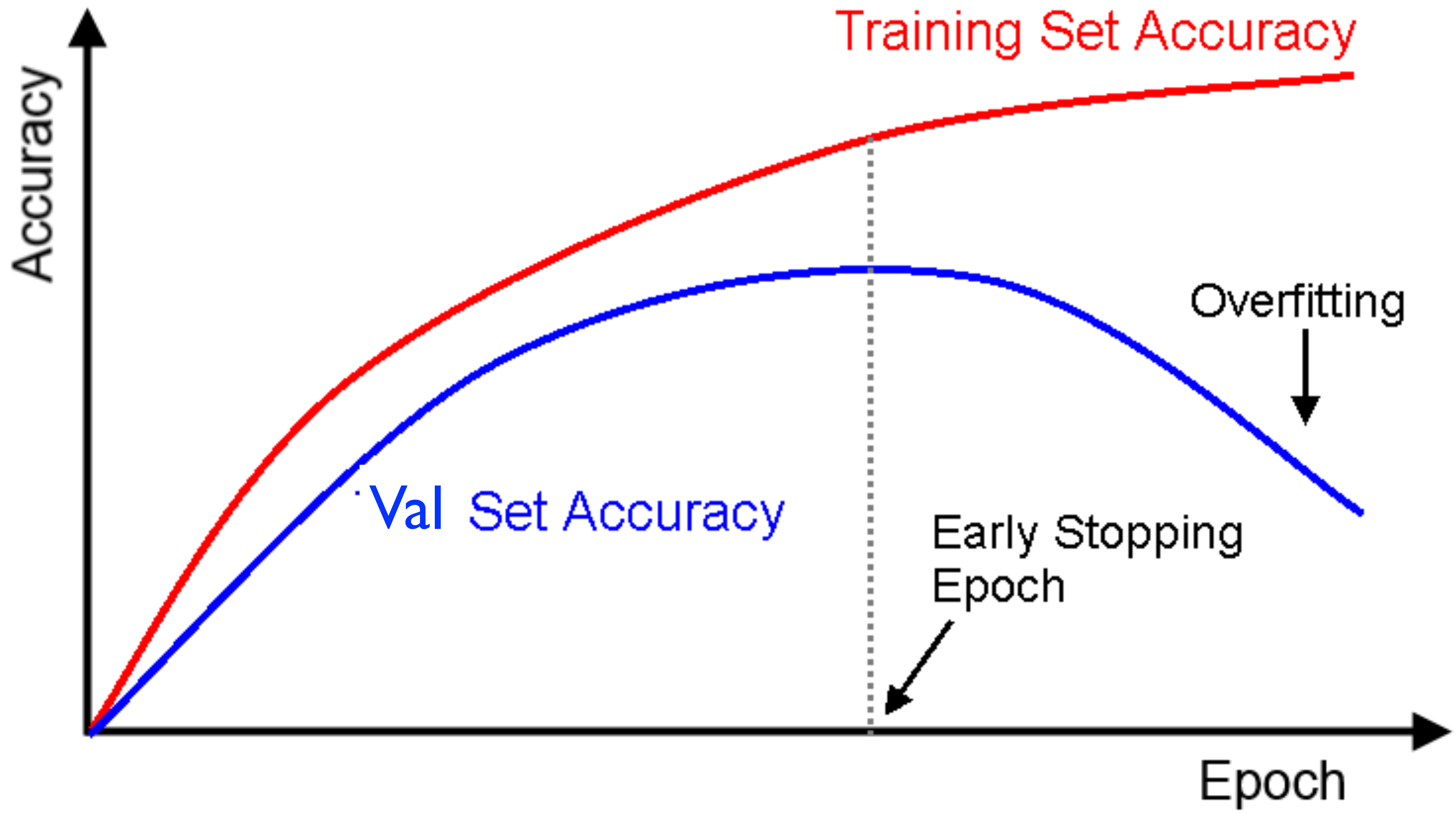


Underfitted   Good Fit/Robust   Overfitted

Overfitting

Many ways to mitigate overfitting problem. Eg early stopping.

Key concept: train/val/test split



Brief (re)fresher on machine learning

"ML is glorified function fitting"

Want to fit a function $f(x; \theta)$ with some parameters θ ("weights") to a collection of examples $\{x_i\}$ in order to achieve some objective.

Brief (re)fresher on machine learning

"ML is glorified function fitting"

Want to fit a function $f(x; \theta)$ with some parameters θ ("weights") to a collection of examples $\{x_i\}$ in order to achieve some objective.

Finally, what functions to use?

Brief (re)fresher on machine learning

"ML is glorified function fitting"

Want to fit a function $f(x; \theta)$ with some parameters θ ("weights") to a collection of examples $\{x_i\}$ in order to achieve some objective.

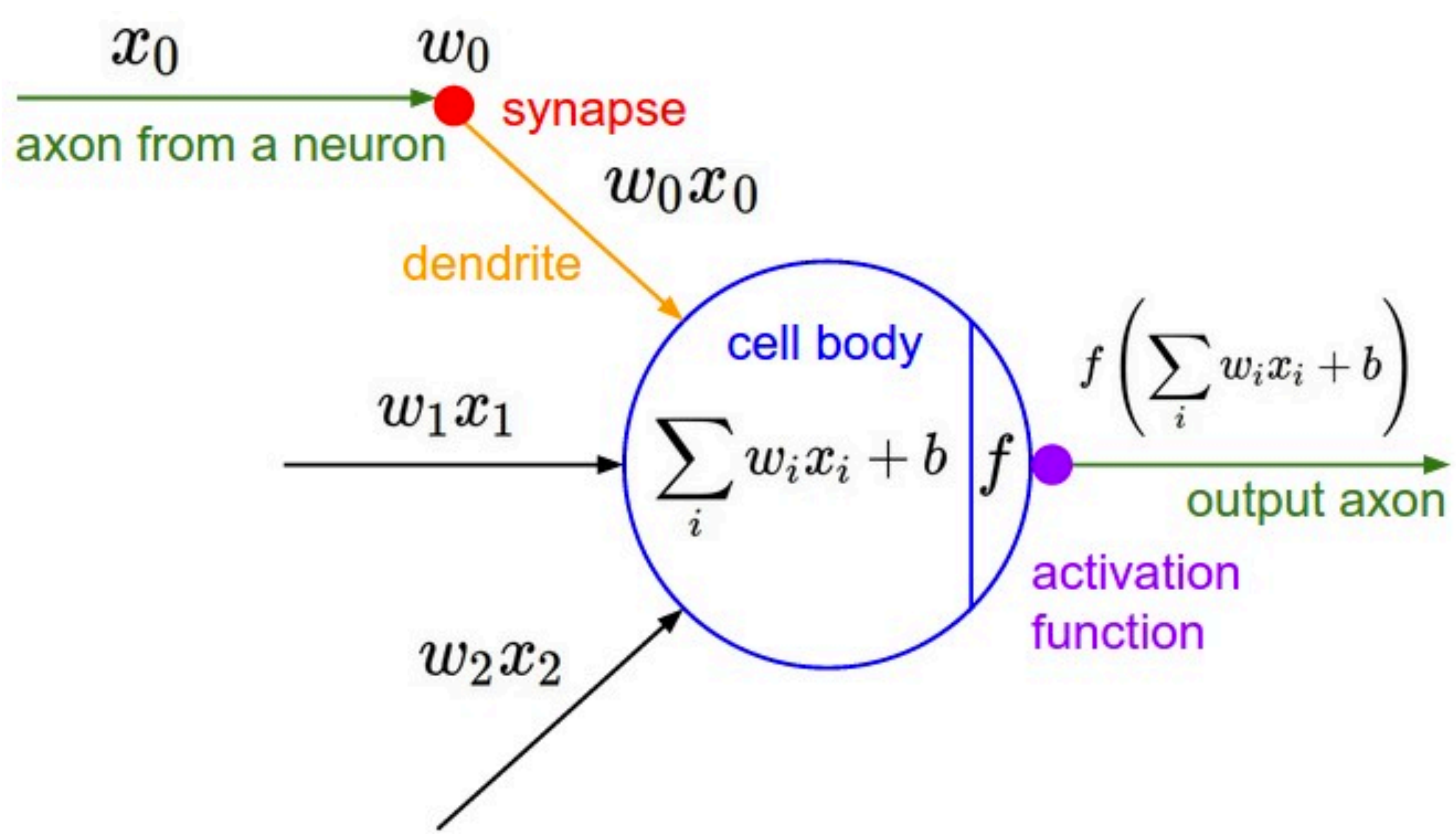
Finally, what functions to use?

Current trend: deep neural networks!

2. Intro to Deep Learning

What is a (deep) neural network?

Basic building block of a neural network:

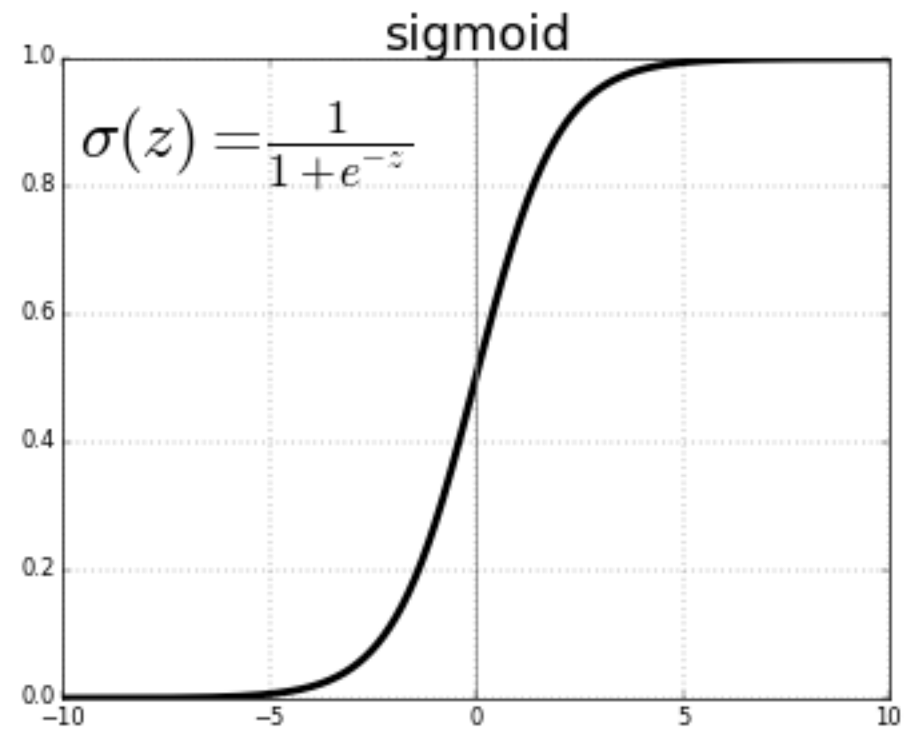


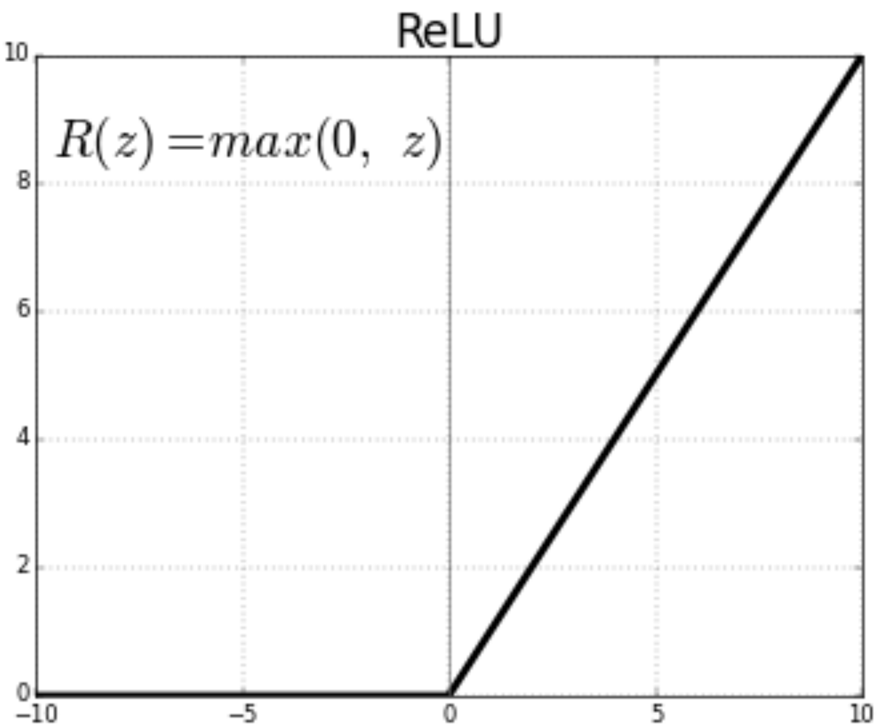
Modeled on biological systems

Activation functions

NNs need a source of non-linearity so they can learn general functions.

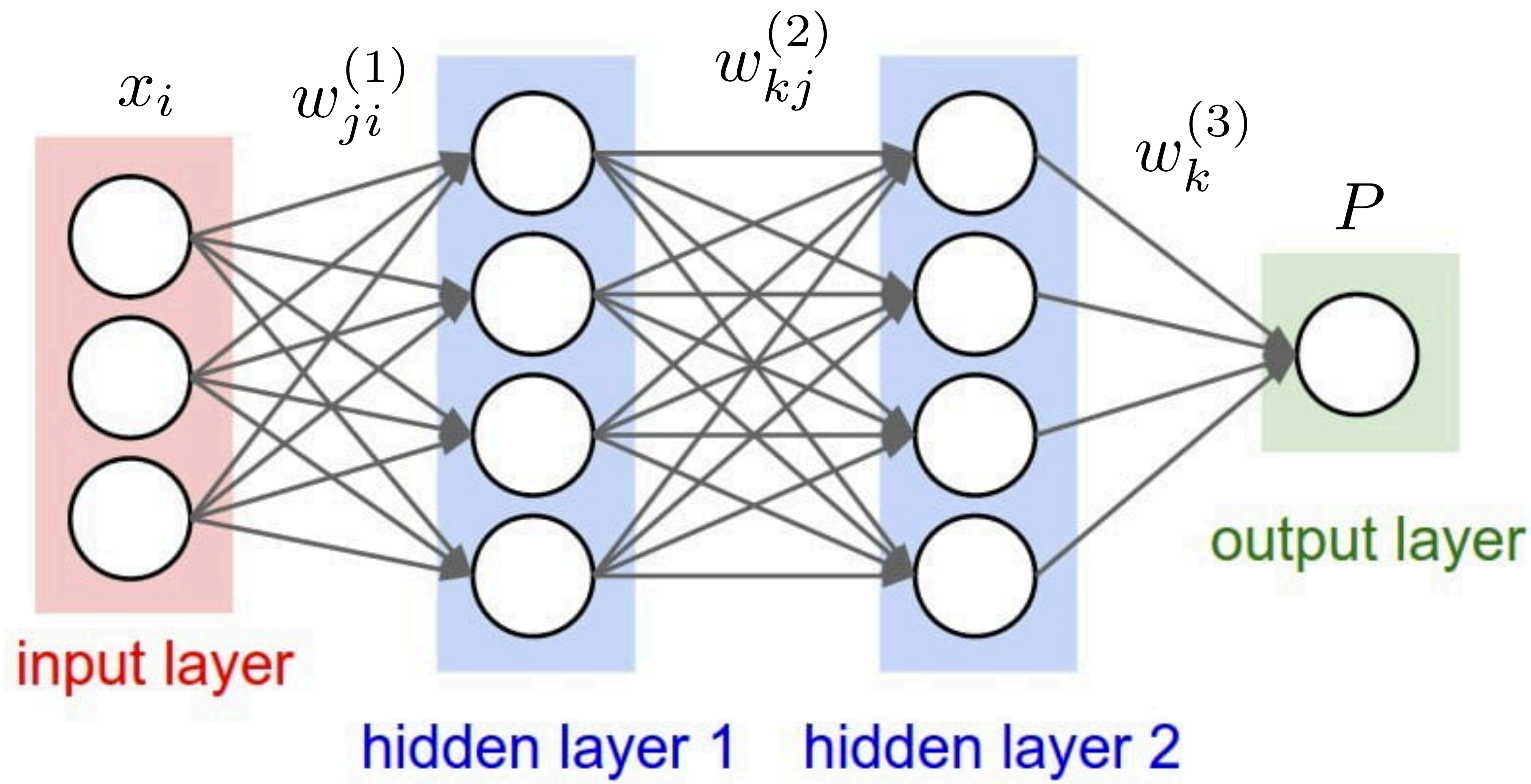
This is usually implemented with the activation function.





Sigmoid used to be standard. But this led to the vanishing gradient problem. The ReLU activation was invented to solve this problem. Now it is the standard.

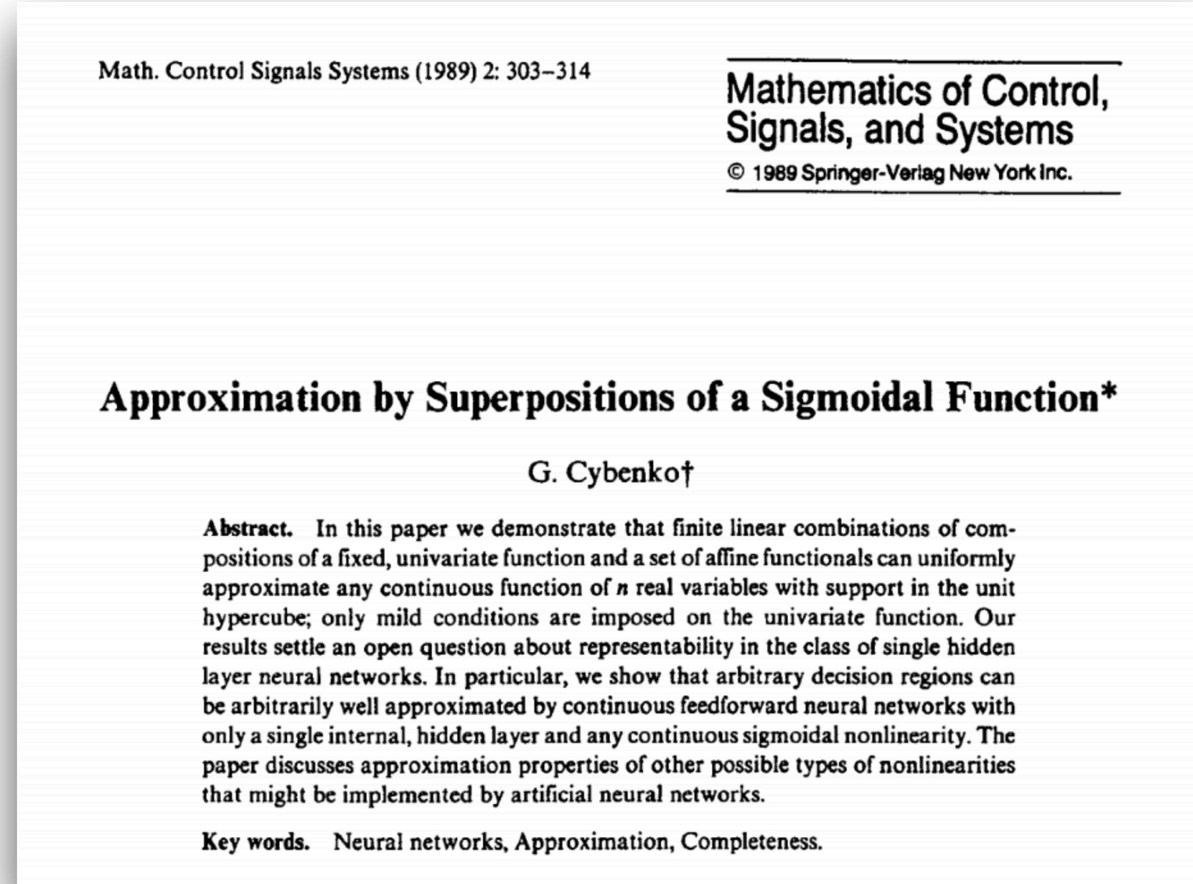
“Fully connected” or “Dense” Neural Network



$$P = A^{(3)}(w_k^{(3)} A^{(2)}(w_{kj}^{(2)} A^{(1)}(w_{ji}^{(1)} x_i + b_j^{(1)}) + b_k^{(2)}) + b^{(3)})$$

Expressiveness

One reason why NNs "work" is that they are *universal function approximators* (asymptotically)



Math. Control Signals Systems (1989) 2: 303–314

Mathematics of Control, Signals, and Systems

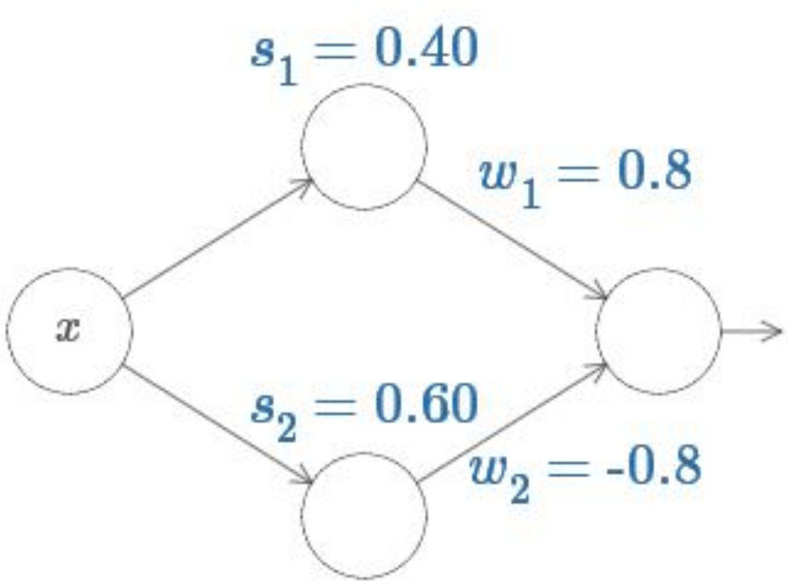
© 1989 Springer-Verlag New York Inc.

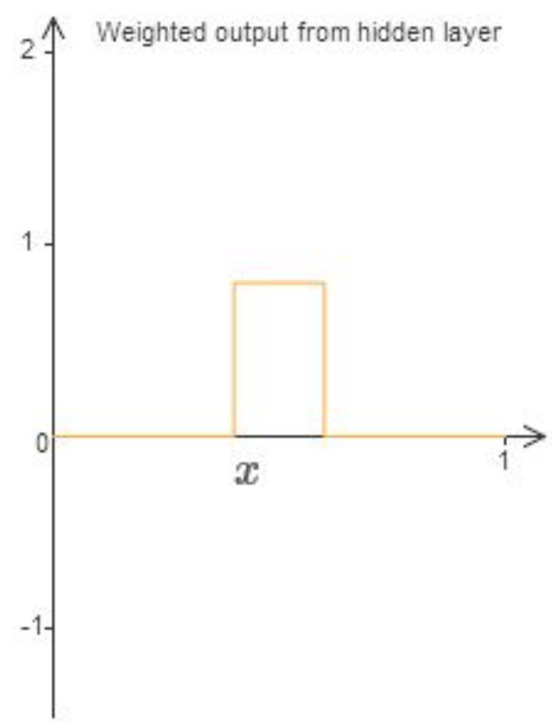
Approximation by Superpositions of a Sigmoidal Function*

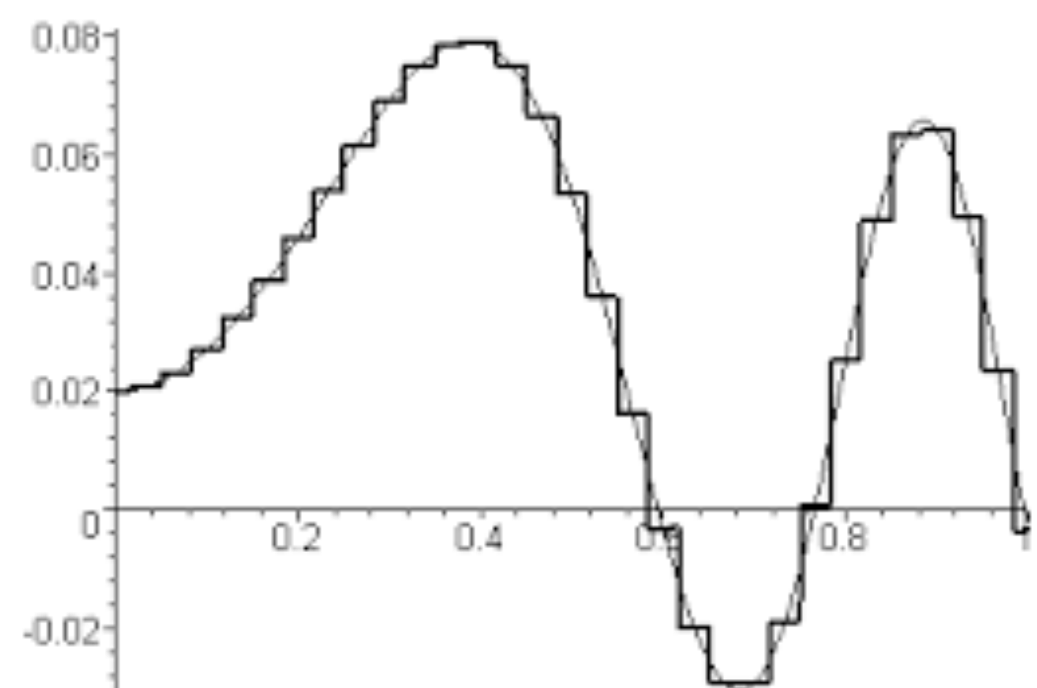
G. Cybenko†

Abstract. In this paper we demonstrate that finite linear combinations of compositions of a fixed, univariate function and a set of affine functionals can uniformly approximate any continuous function of *n* real variables with support in the unit hypercube; only mild conditions are imposed on the univariate function. Our results settle an open question about representability in the class of single hidden layer neural networks. In particular, we show that arbitrary decision regions can be arbitrarily well approximated by continuous feedforward neural networks with only a single internal, hidden layer and any continuous sigmoidal nonlinearity. The paper discusses approximation properties of other possible types of nonlinearities that might be implemented by artificial neural networks.

Key words. Neural networks, Approximation, Completeness.

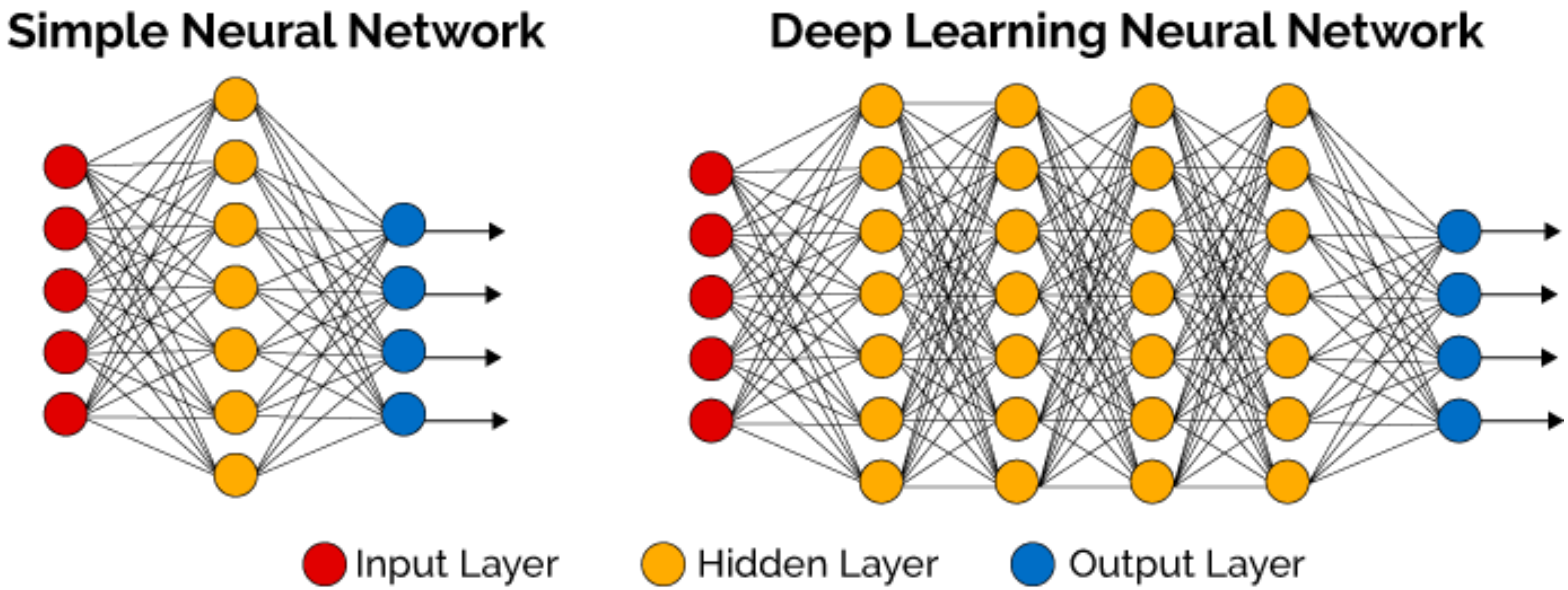






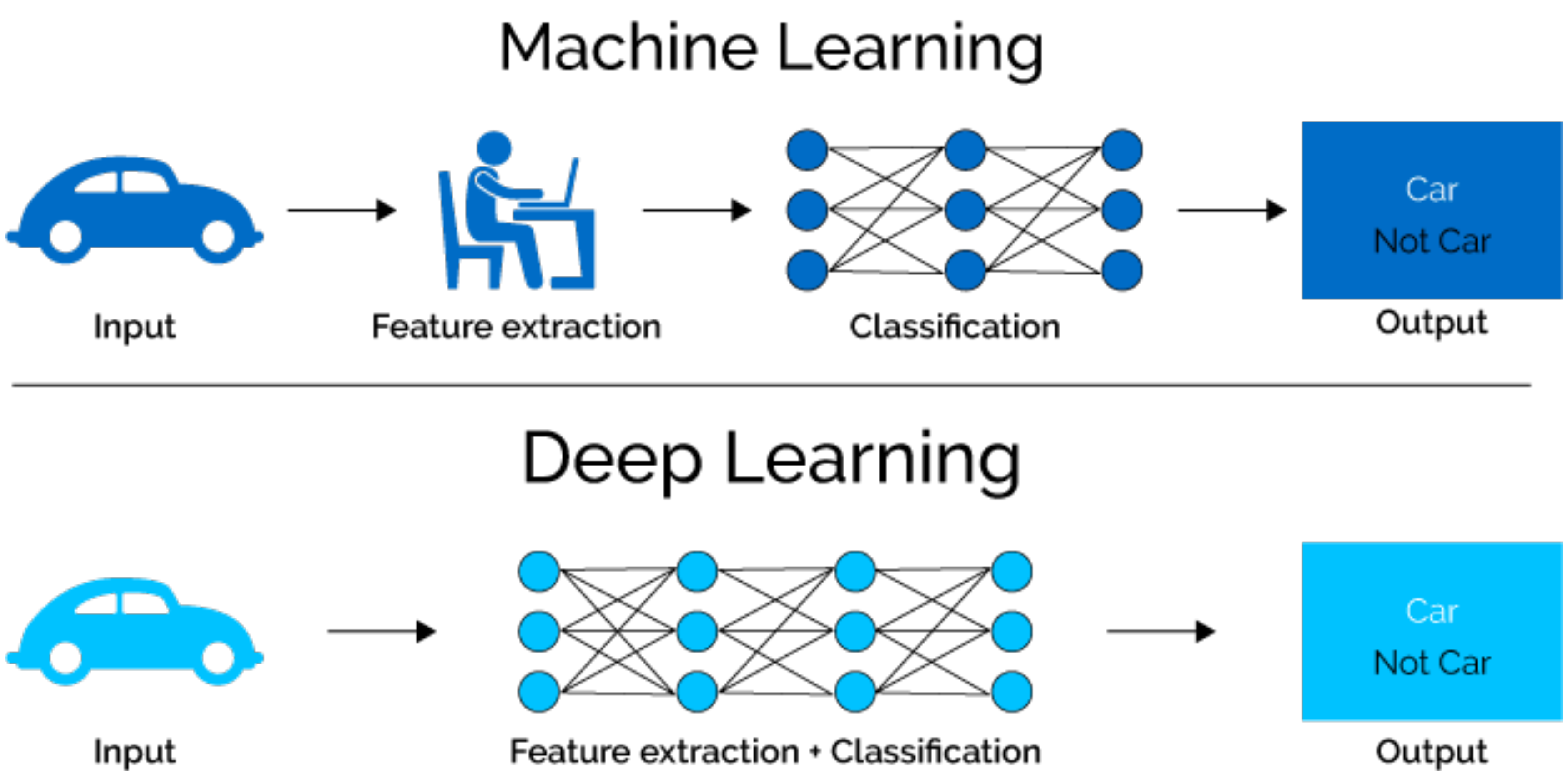
Can fit any function with a single, infinitely-wide hidden layer

Deep neural networks



Expressivity of a neural network increases exponentially with the number of layers (Delalleau and Bengio, 2011, Montufar et al 1402.1869, Ganguli et al 1606.05336, 1606.05340)

End-to-end learning



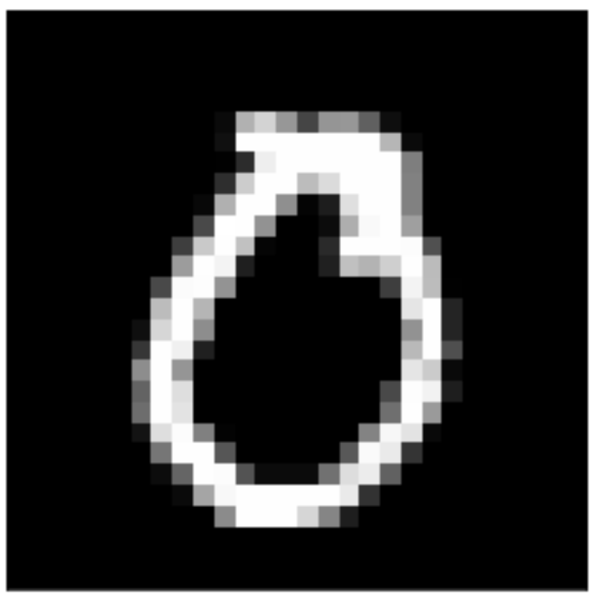
From towardsdatascience.com

Universal function approximation and high expressivity means that deep NNs can learn abstract concepts from low-level, high-dimensional inputs

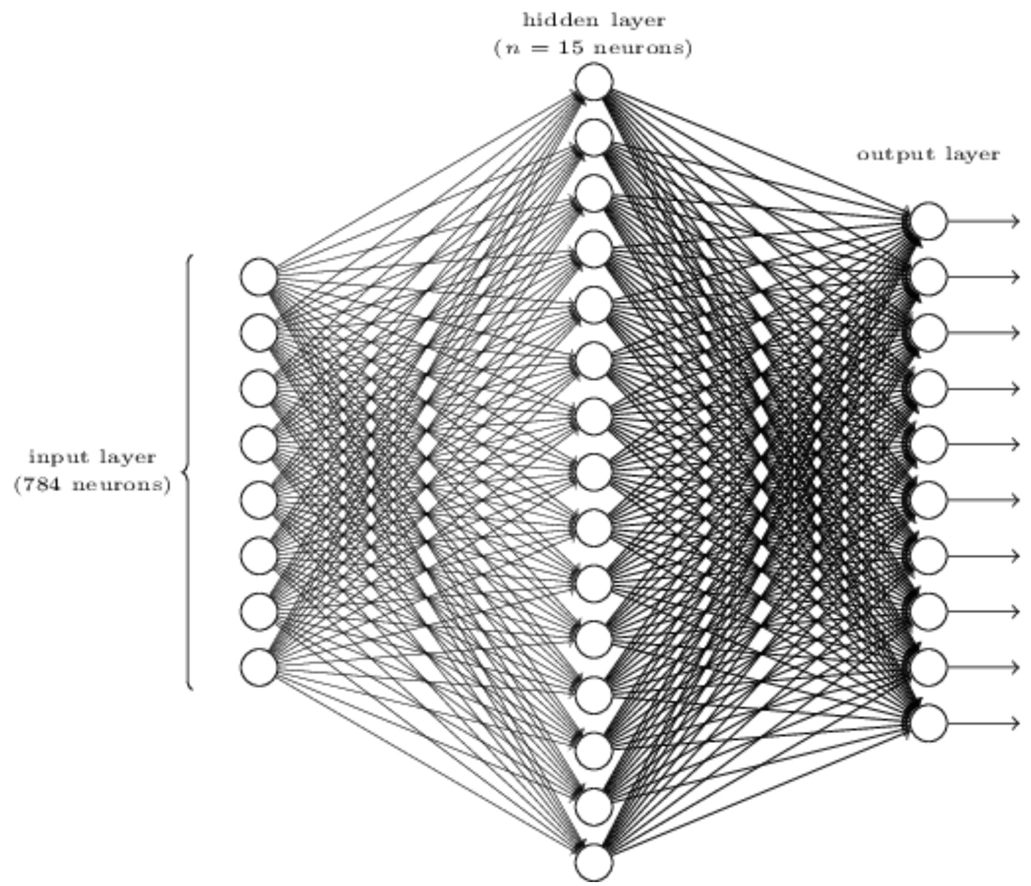
"Automated feature engineering"

"End-to-end learning"

Example: *MNIST in more detail*



x



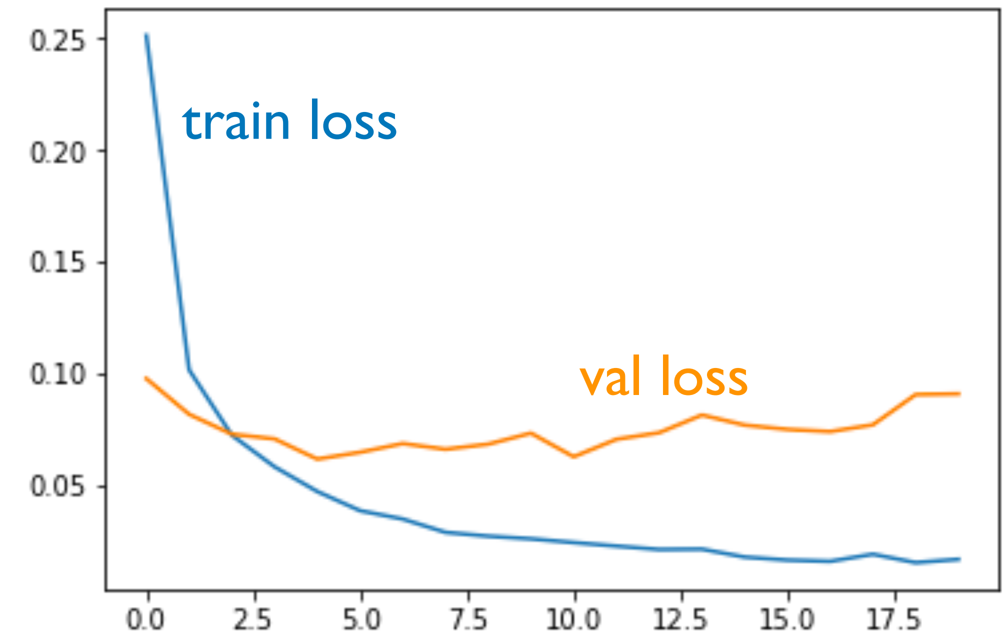
$$\begin{pmatrix} P_0(x;\theta) \\ P_1(x;\theta) \\ \vdots \\ P_9(x;\theta) \end{pmatrix}$$

$$\sum_i P_i(x;\theta) = 1$$

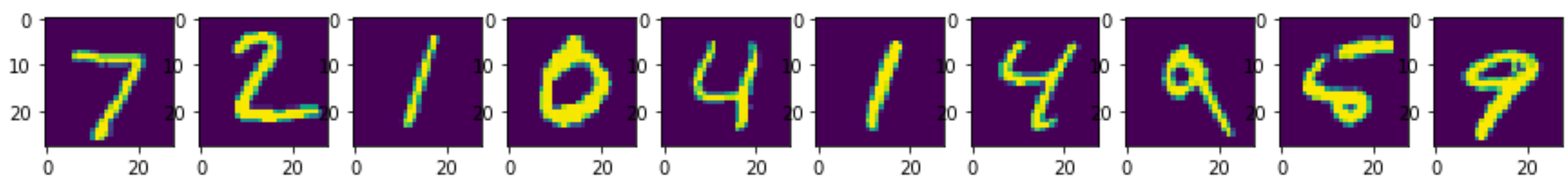
Output: probability it's a 0, 1, …,9

Example: *MNIST in more detail*

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 512)               401920
_________________________________________________________________
dropout_1 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_2 (Dense)              (None, 512)               262656
_________________________________________________________________
dropout_2 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_3 (Dense)              (None, 10)                5130
=================================================================
Total params: 669,706
Trainable params: 669,706
Non-trainable params: 0
```

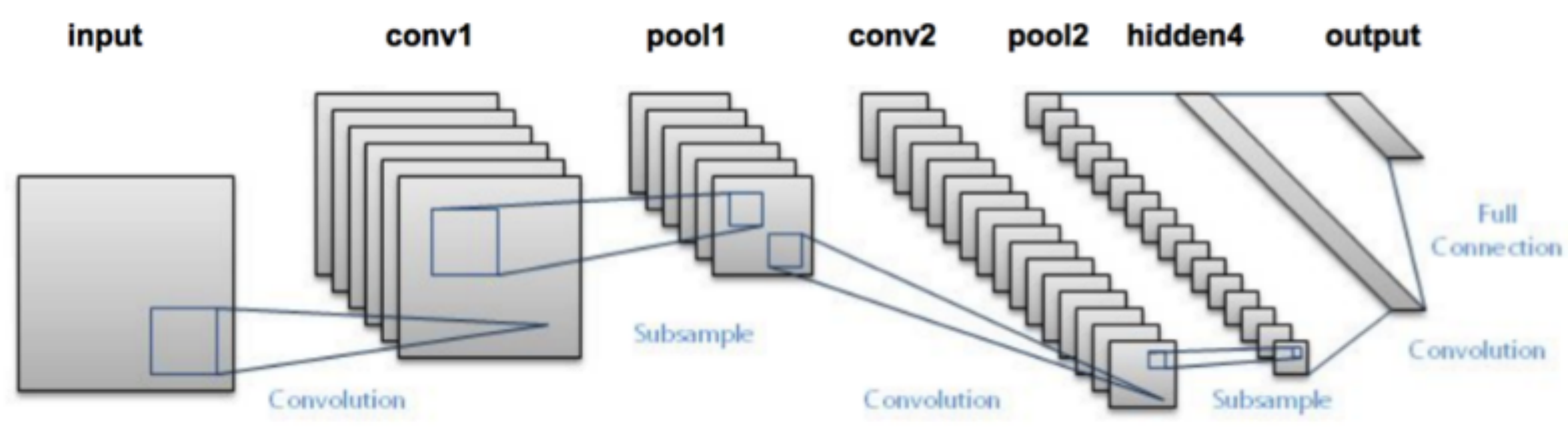



```
Test accuracy: 0.9831
```



prediction: 7 2 1 0 4 1 4 9 6 9

Convolutional neural network (CNN)



Principal neural network architecture for image recognition.
Invented in 1998 (LeCun, Bottou, Bengio, Haffner)

Achieved 99% accuracy on MNIST!

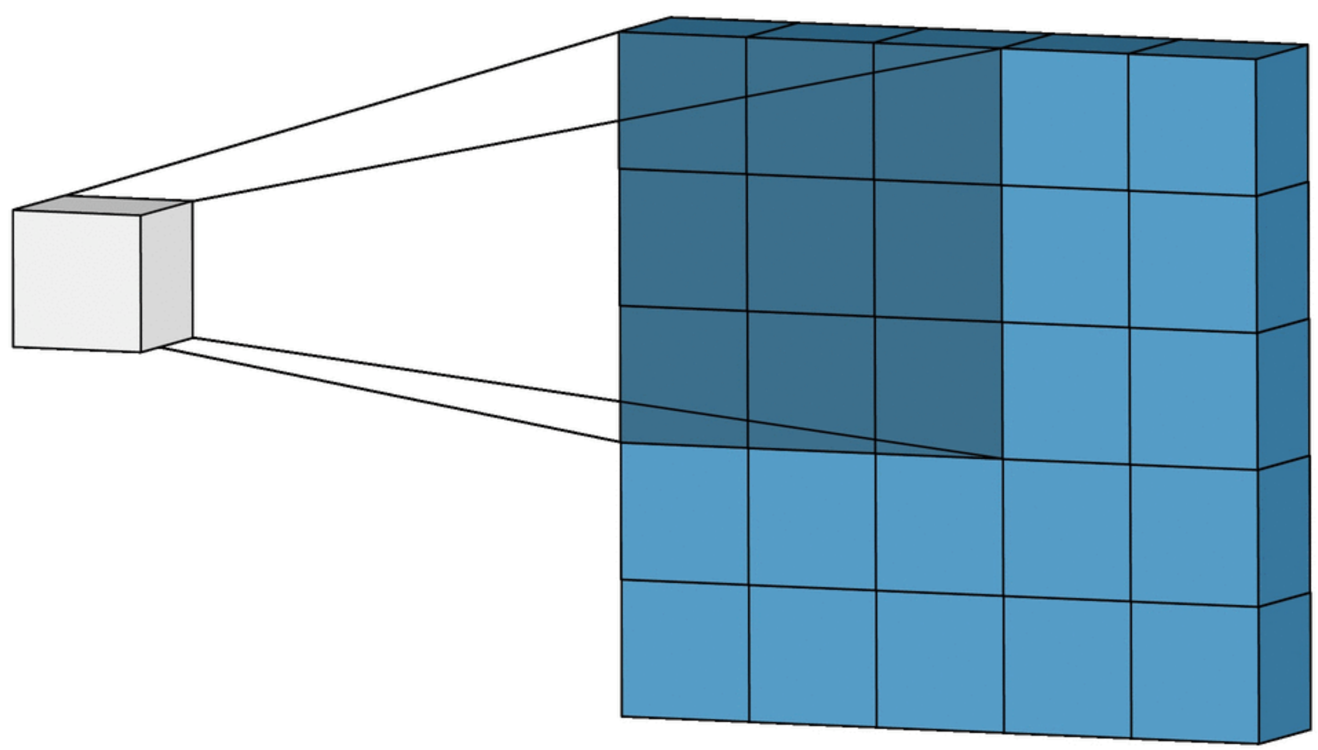
However, CNNs fell out of favor (until AlexNet in 2012) when they did not immediately generalize well to more complex image recognition tasks such as ImageNet.

Convolutional neural network (CNN)

Main idea: features in an image (edges, curves, corners,…eyes, noses,…) are the same no matter where they occur.

Goal: Want to find these features in a translationally invariant way.

Solution: Drag or convolve a "filter" across the image that selects out interesting features.

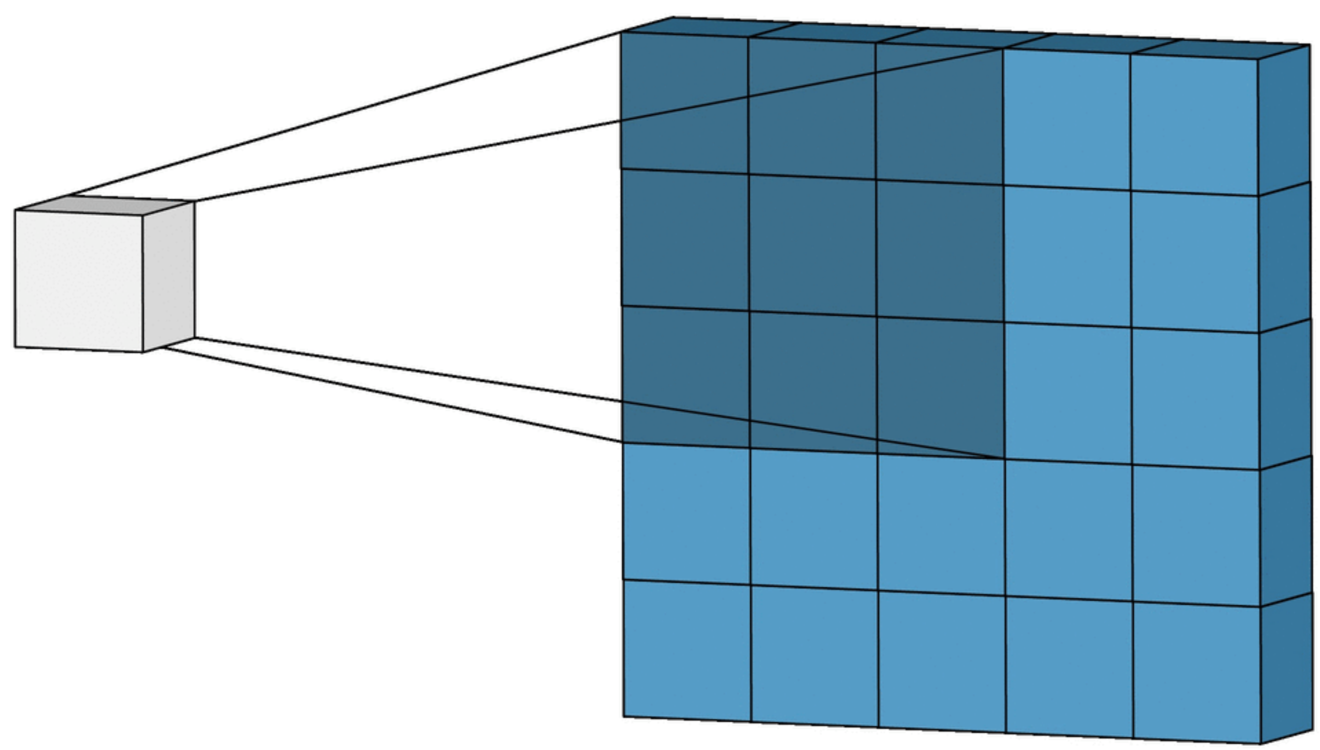


Convolutional neural network (CNN)

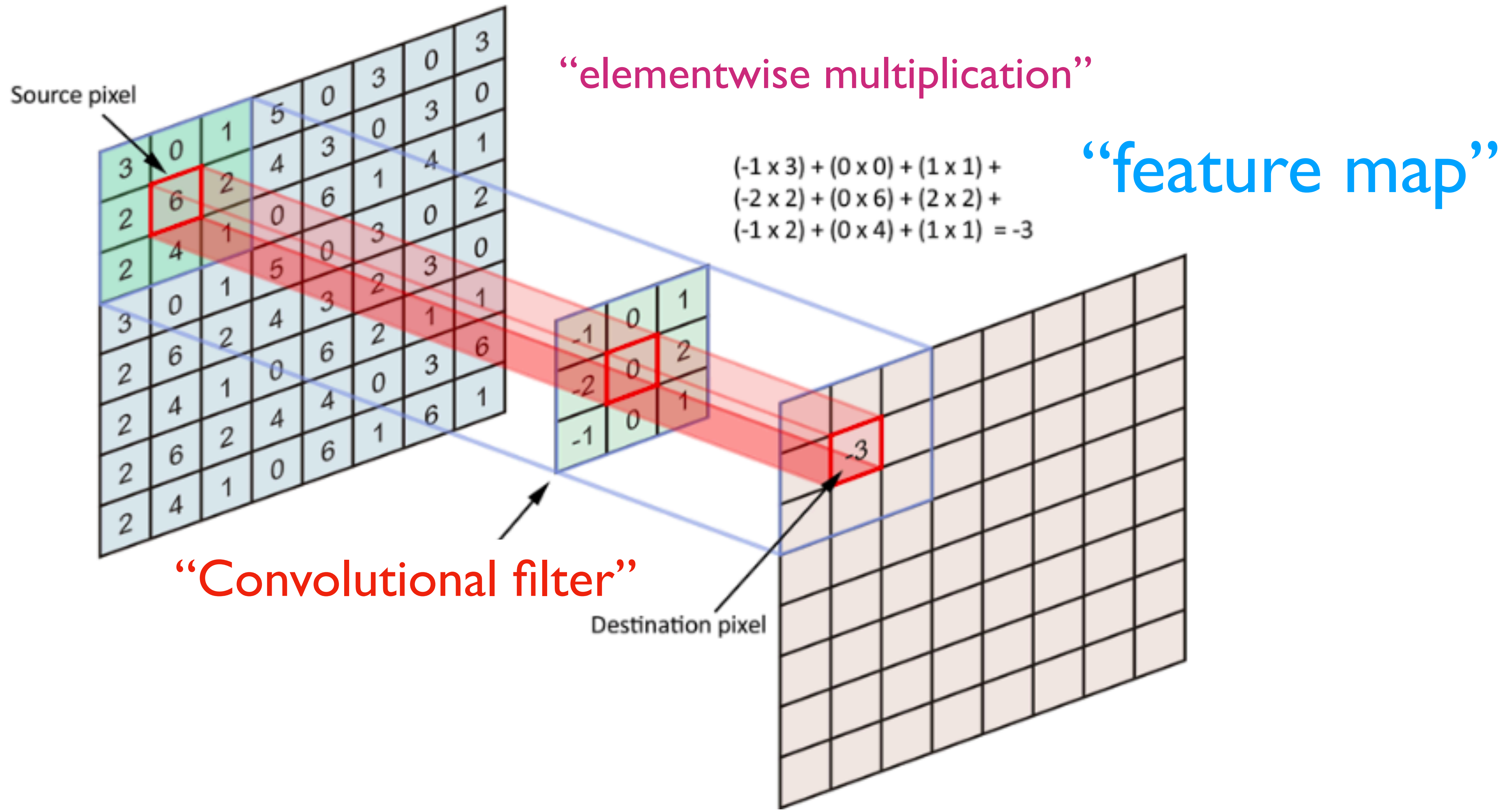
Main idea: features in an image (edges, curves, corners,…eyes, noses,…) are the same no matter where they occur.

Goal: Want to find these features in a translationally invariant way.

Solution: Drag or convolve a “filter” across the image that selects out interesting features.



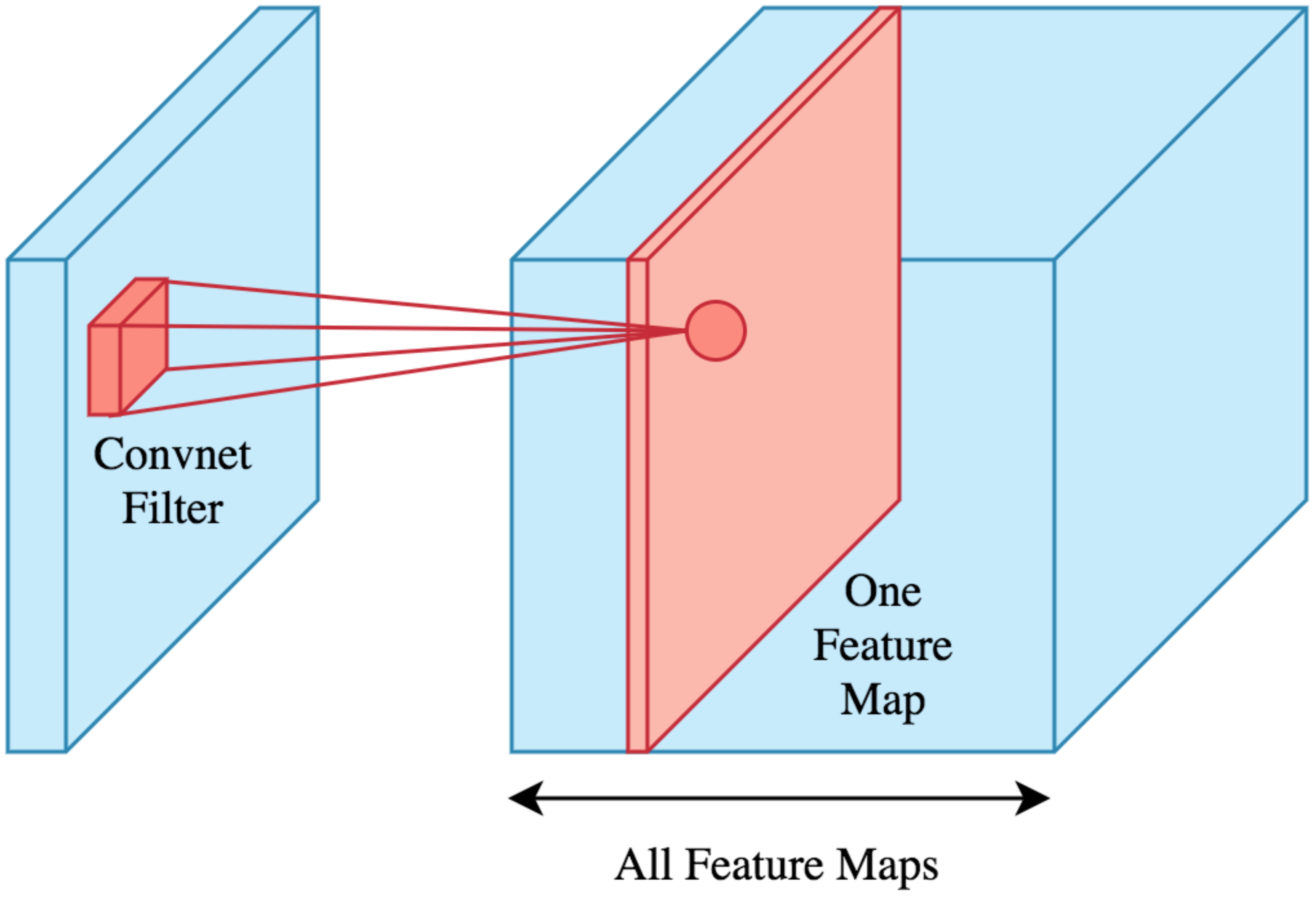
Convolutional neural network (CNN)



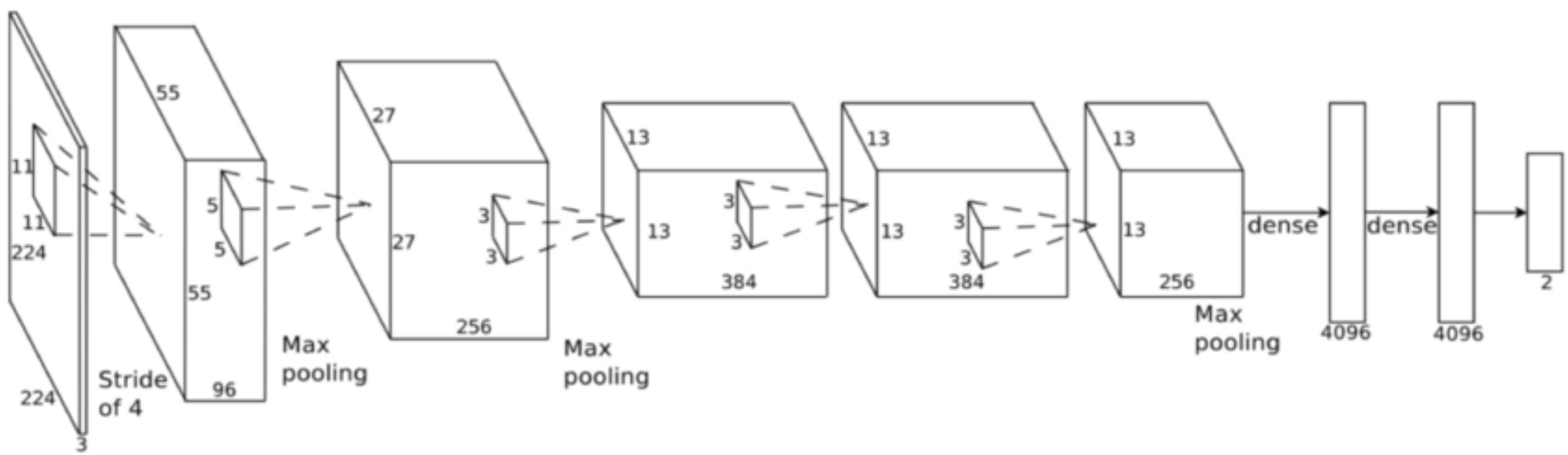
Finds features in the image in a translation invariant way

Convolutional neural network (CNN)

Can apply multiple filters to image to produce a stack of feature maps



Convolutional neural network (CNN)

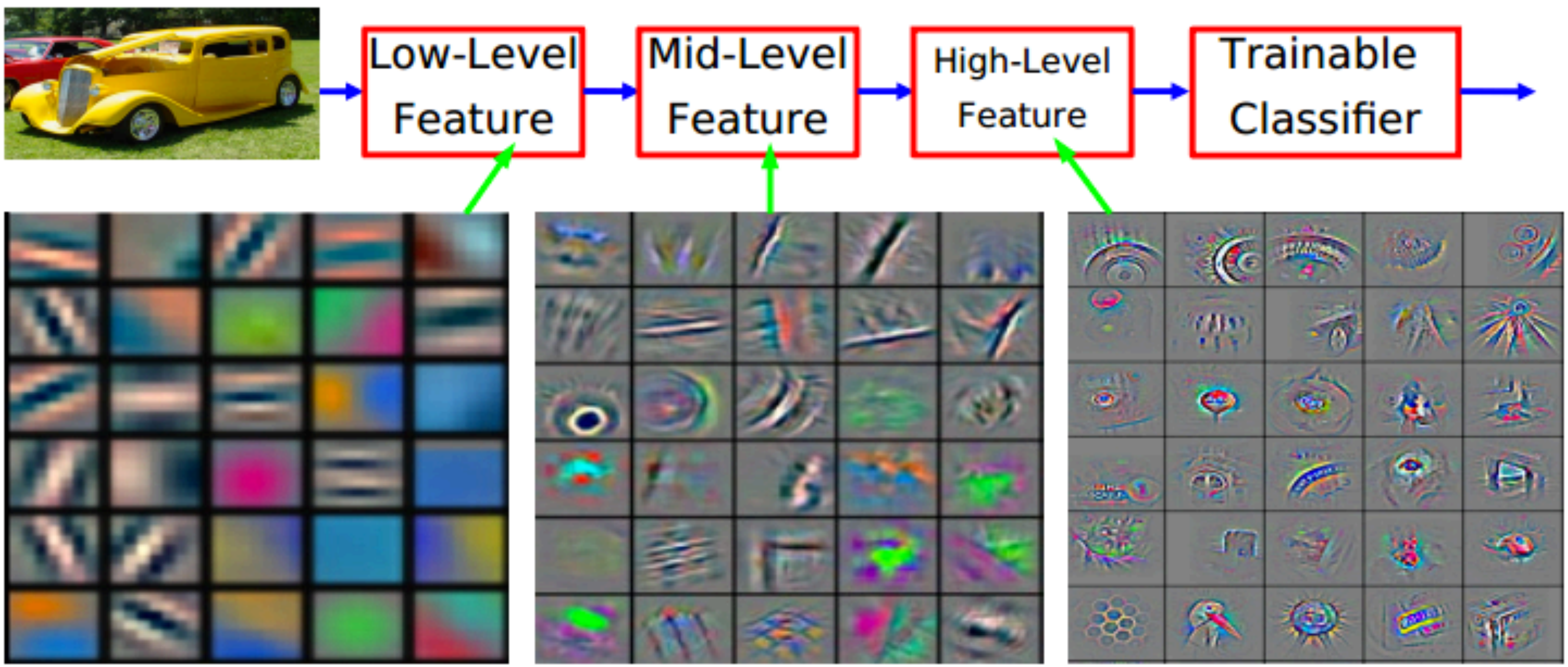


CNNs typically end with fully connected layers.

These are thought to extract the highest level information and to perform the actual classification task on the feature maps found by the convolutional layers.

Convolutional neural network (CNN)

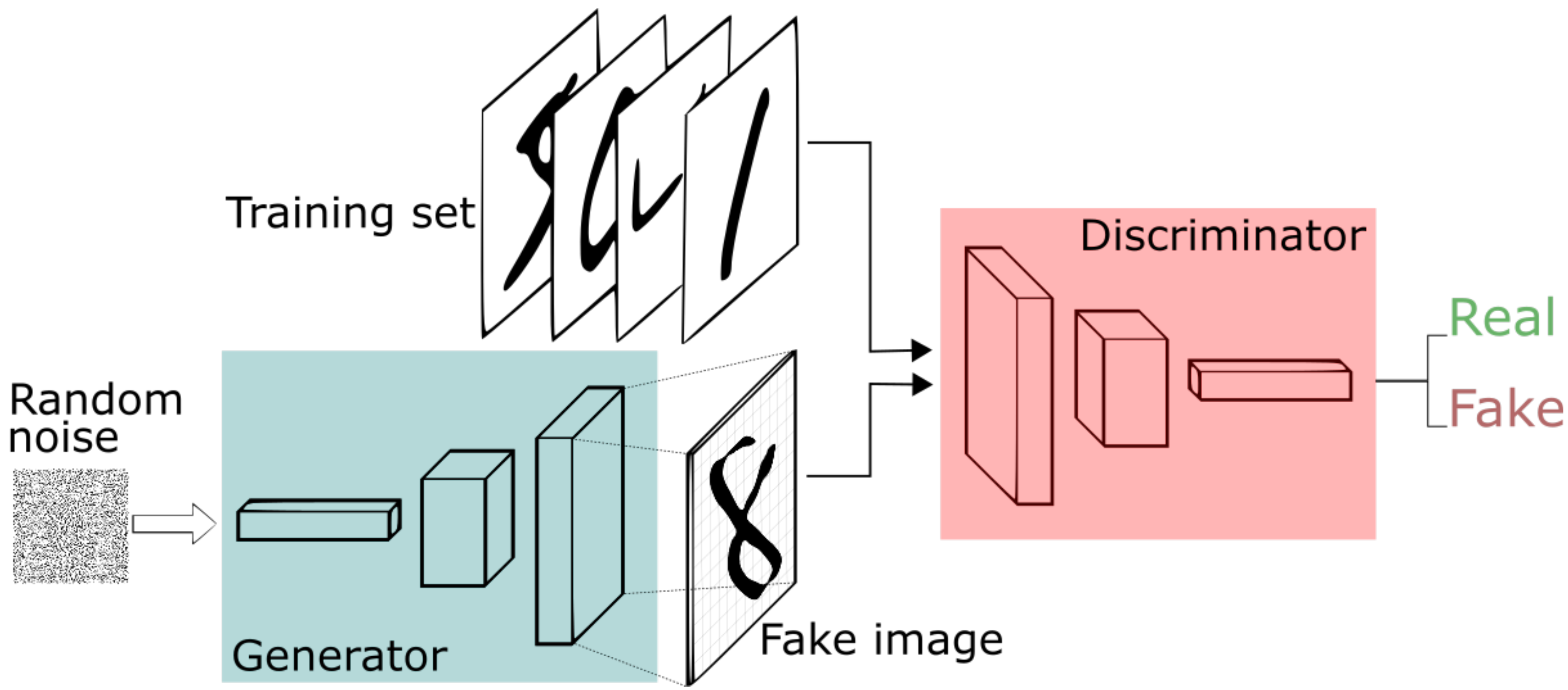
What does the machine learn?



Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

Generative Adversarial Networks (GANs)

Breakthrough method in generative modeling and unsupervised learning (Goodfellow et al. 2014)



Idea: train two neural networks: a "generator" that attempts to generate fake, random images, and a "discriminator" that tries to tell them apart from a database of real images.

Generative Adversarial Networks (GANs)

$$L_{GAN} = \sum_{x \in \text{real}} \log D(x) + \sum_{z \in \text{random}} \log(1 - D(G(z)))$$

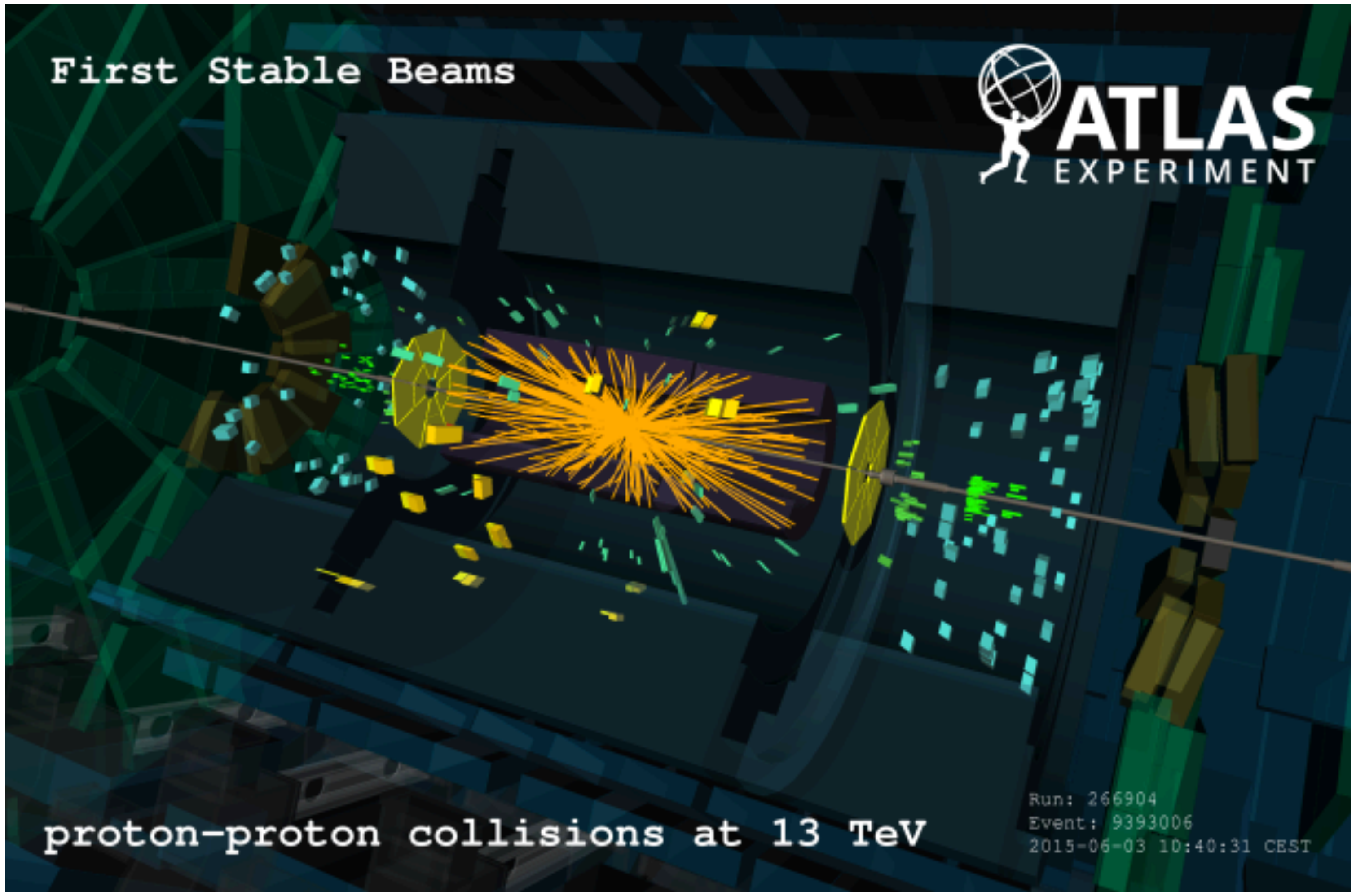
Training is performed "adversarially"

- Discriminator tries to minimize loss
- Generator tries to ***maximize*** loss
- Take turns training discriminator and generator to optimize fake image generator

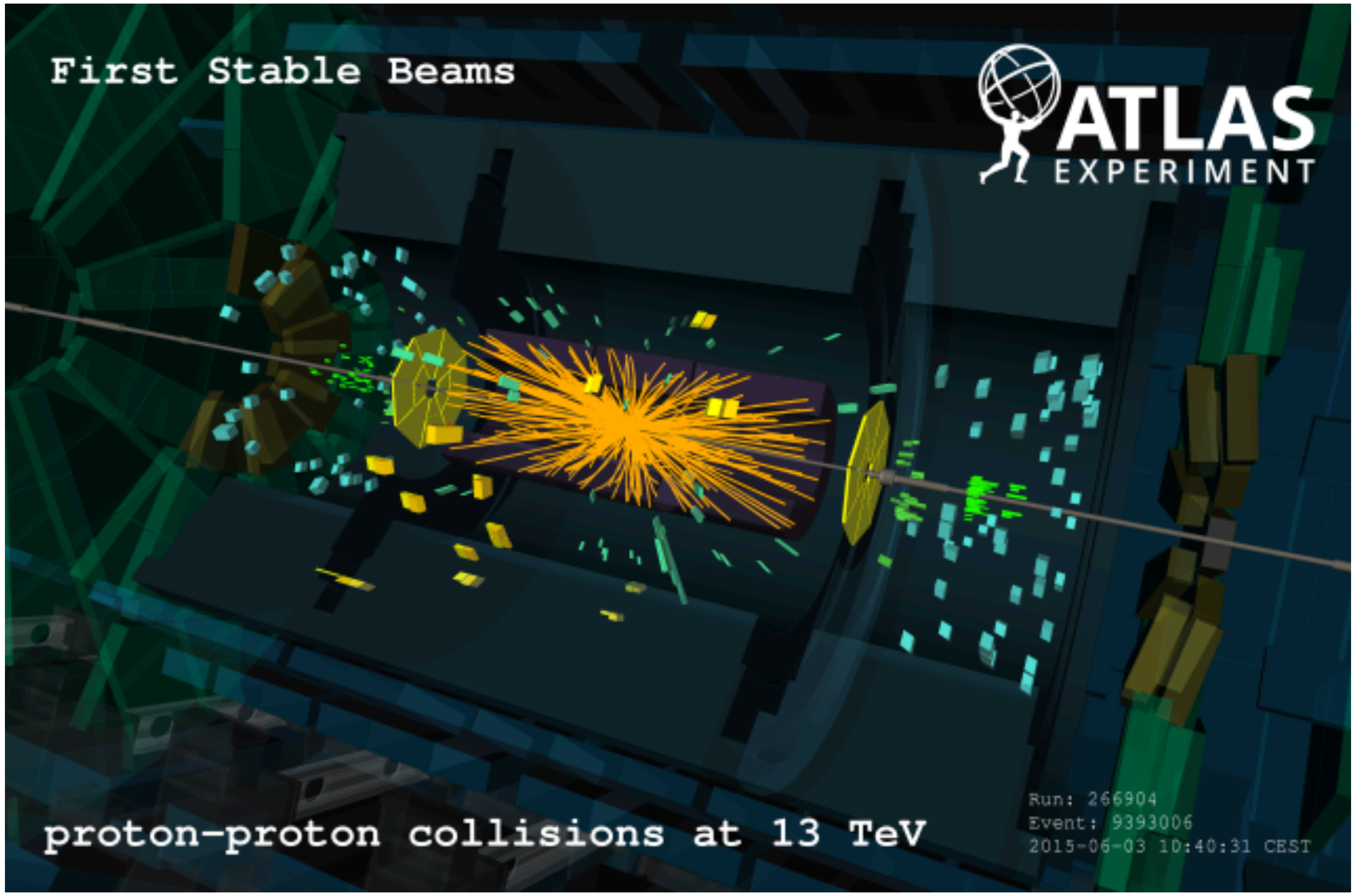
Generative Adversarial Networks (GANs)

Real or fake?

3. Deep Learning at the LHC

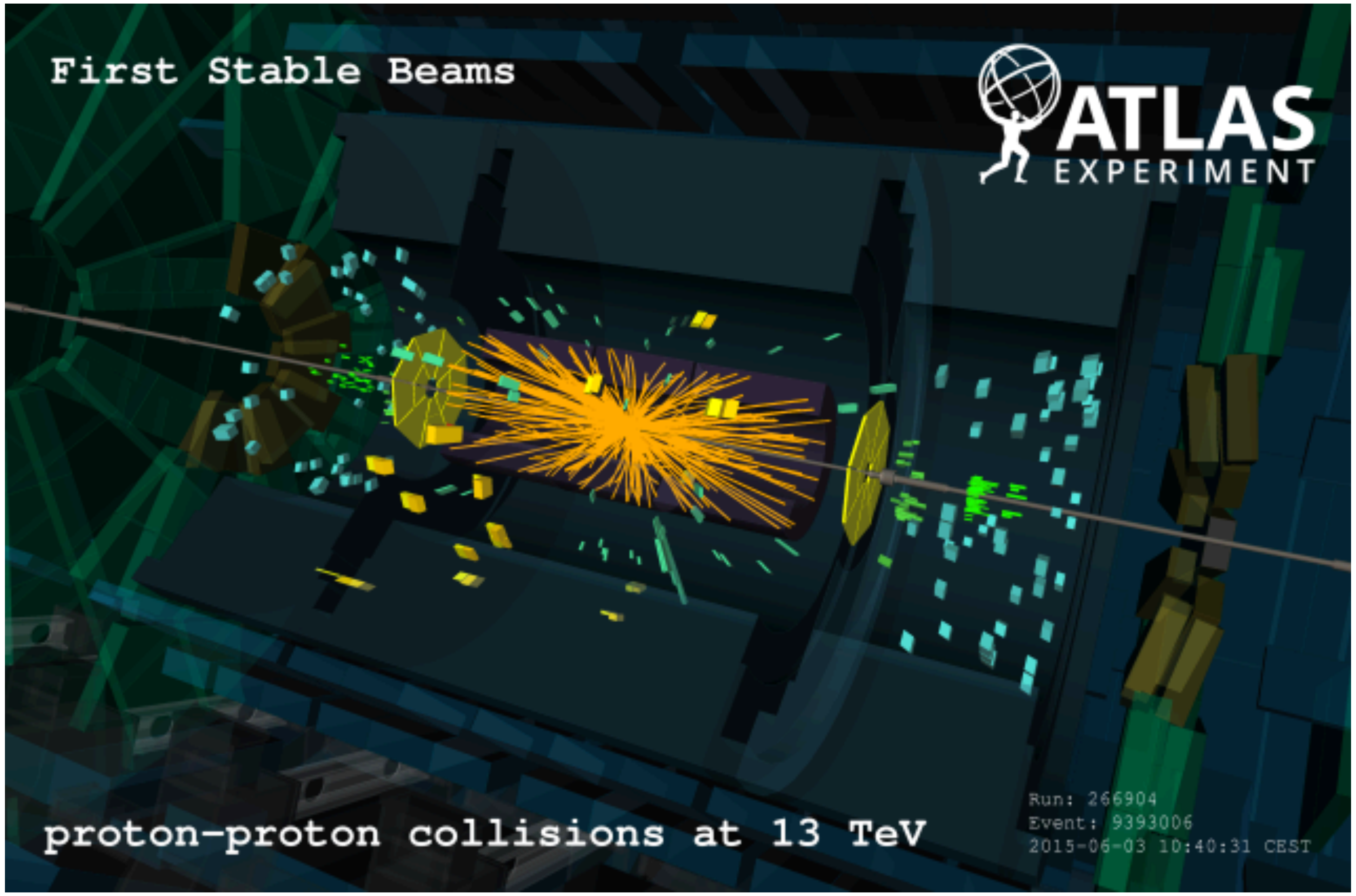


LHC and Big Data



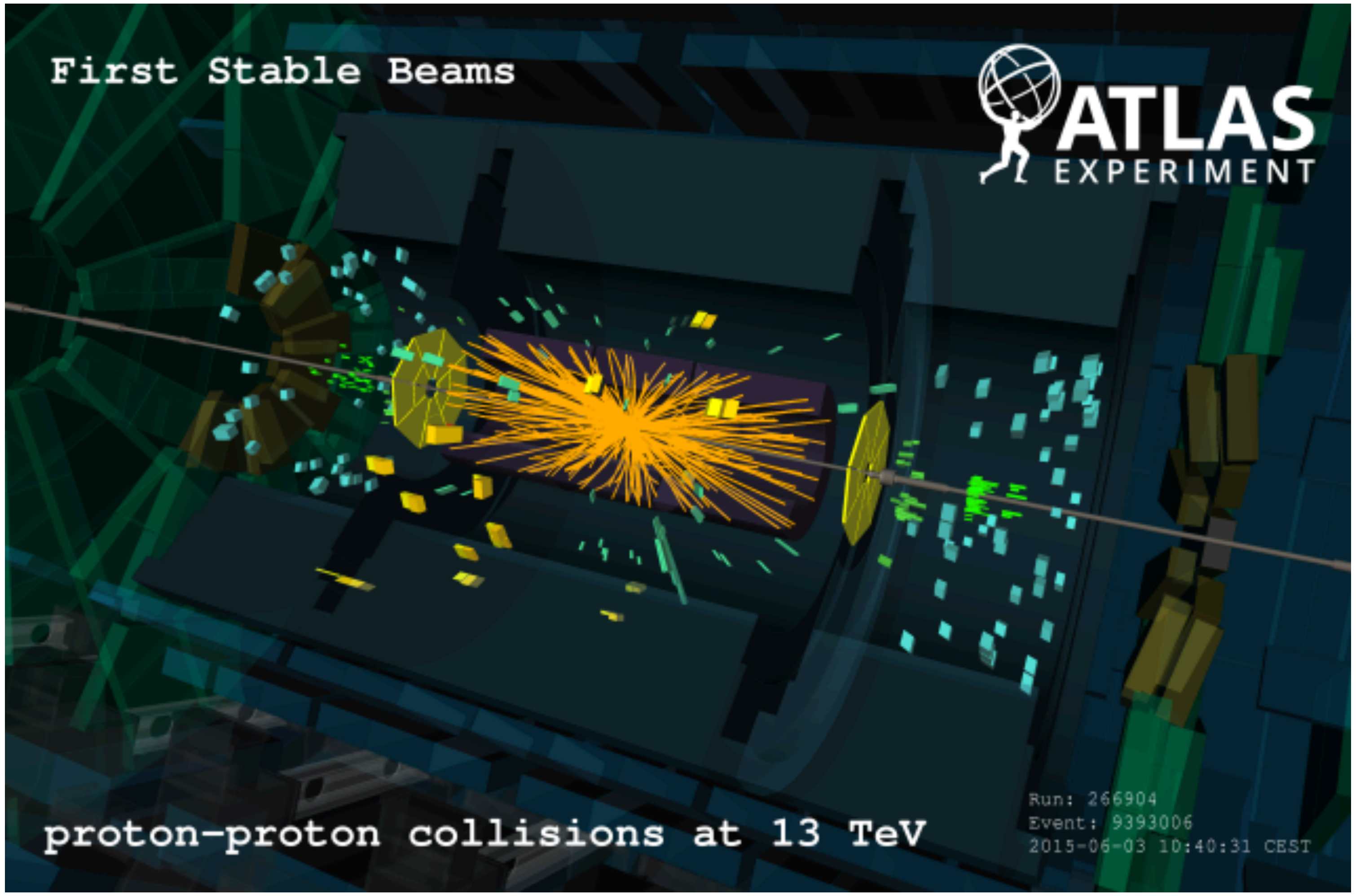
LHC and Big Data

- 600 million collisions per second



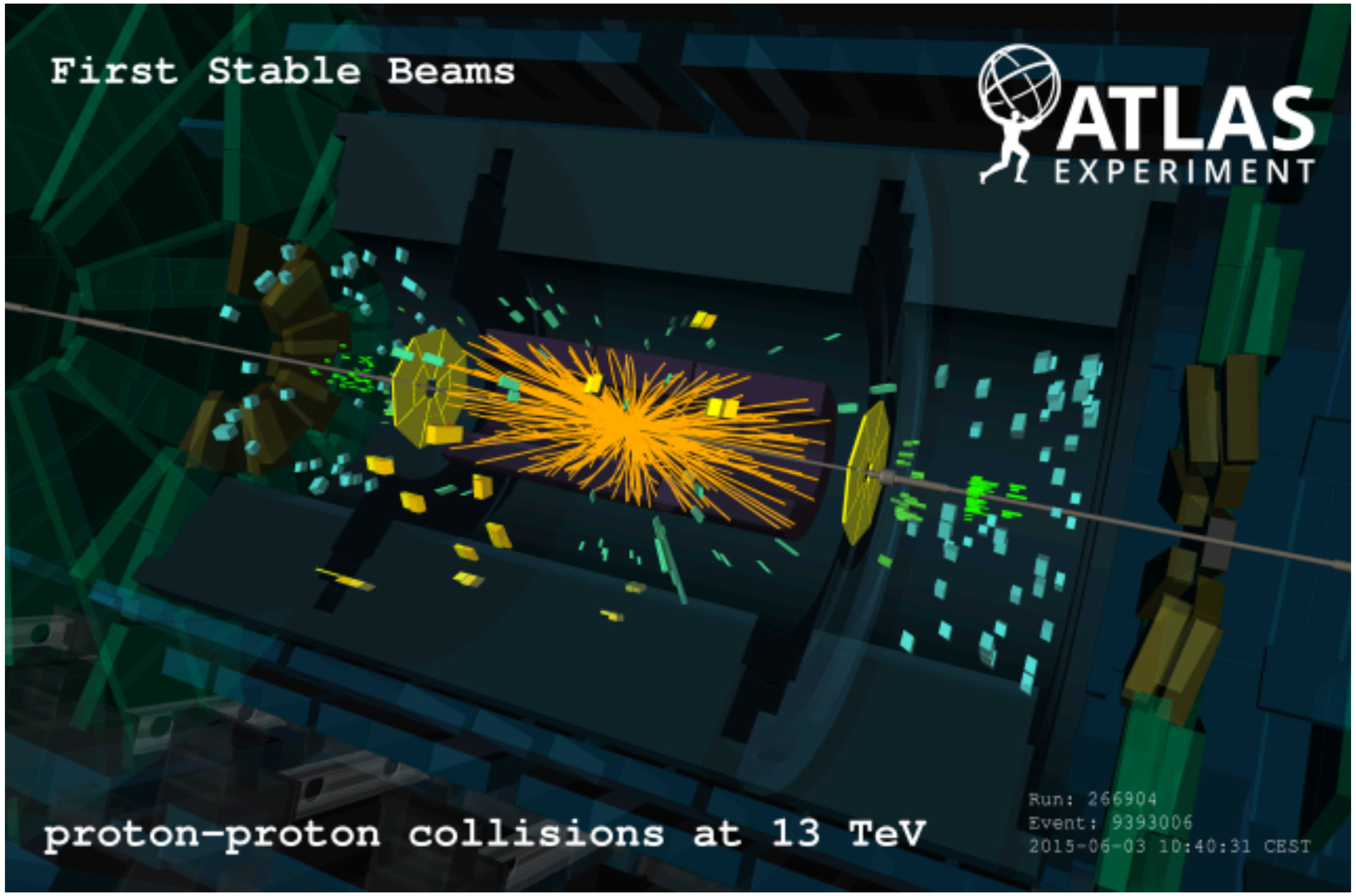
LHC and Big Data

- 600 million collisions per second
- Raw data rate ~ 1 PB/s (1 PB=10^6 GB)



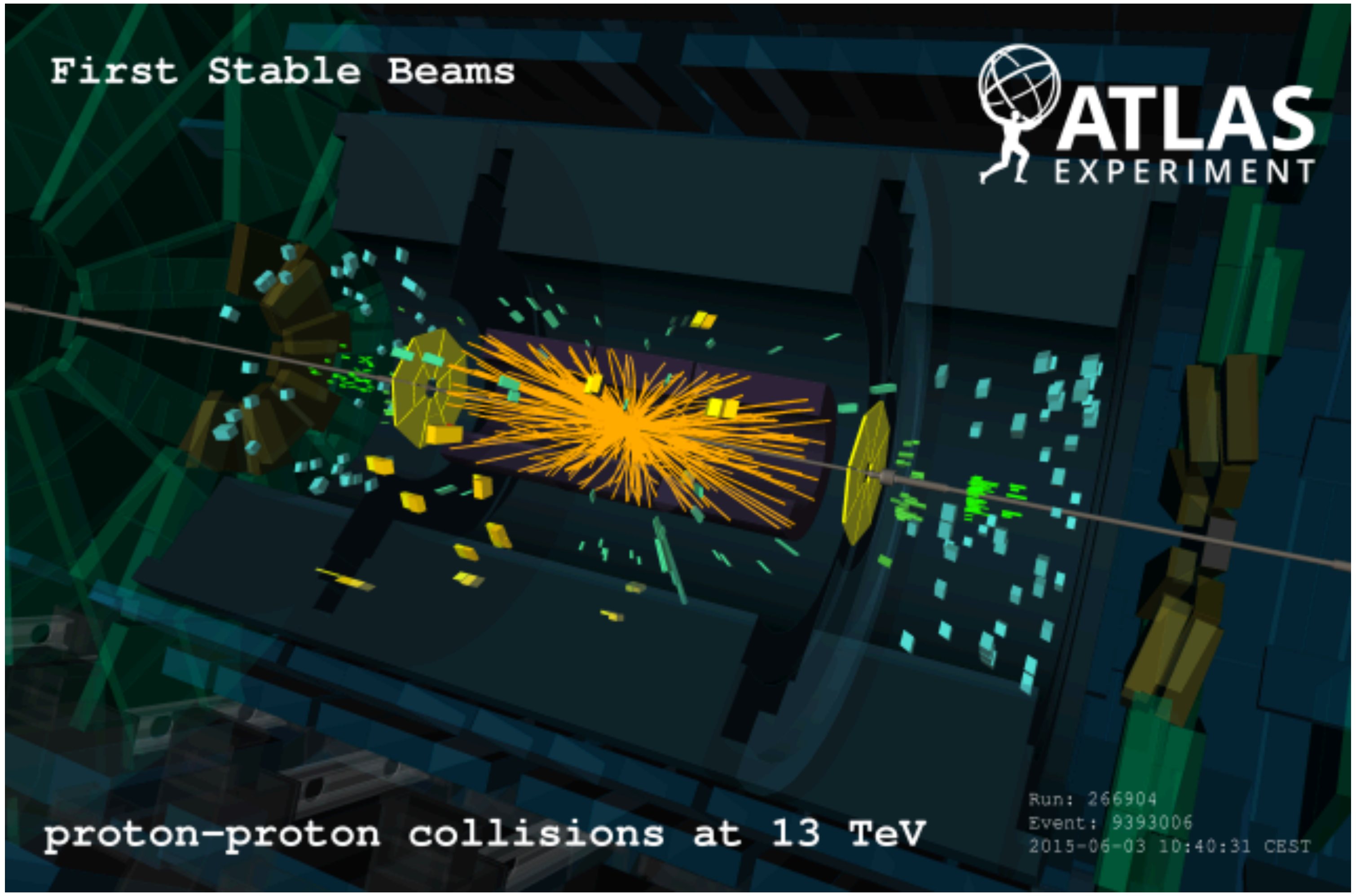
LHC and Big Data

- 600 million collisions per second
- Raw data rate ~ 1 PB/s (1 PB=10^6 GB)
- Actual data rate ~ 25 GB/s



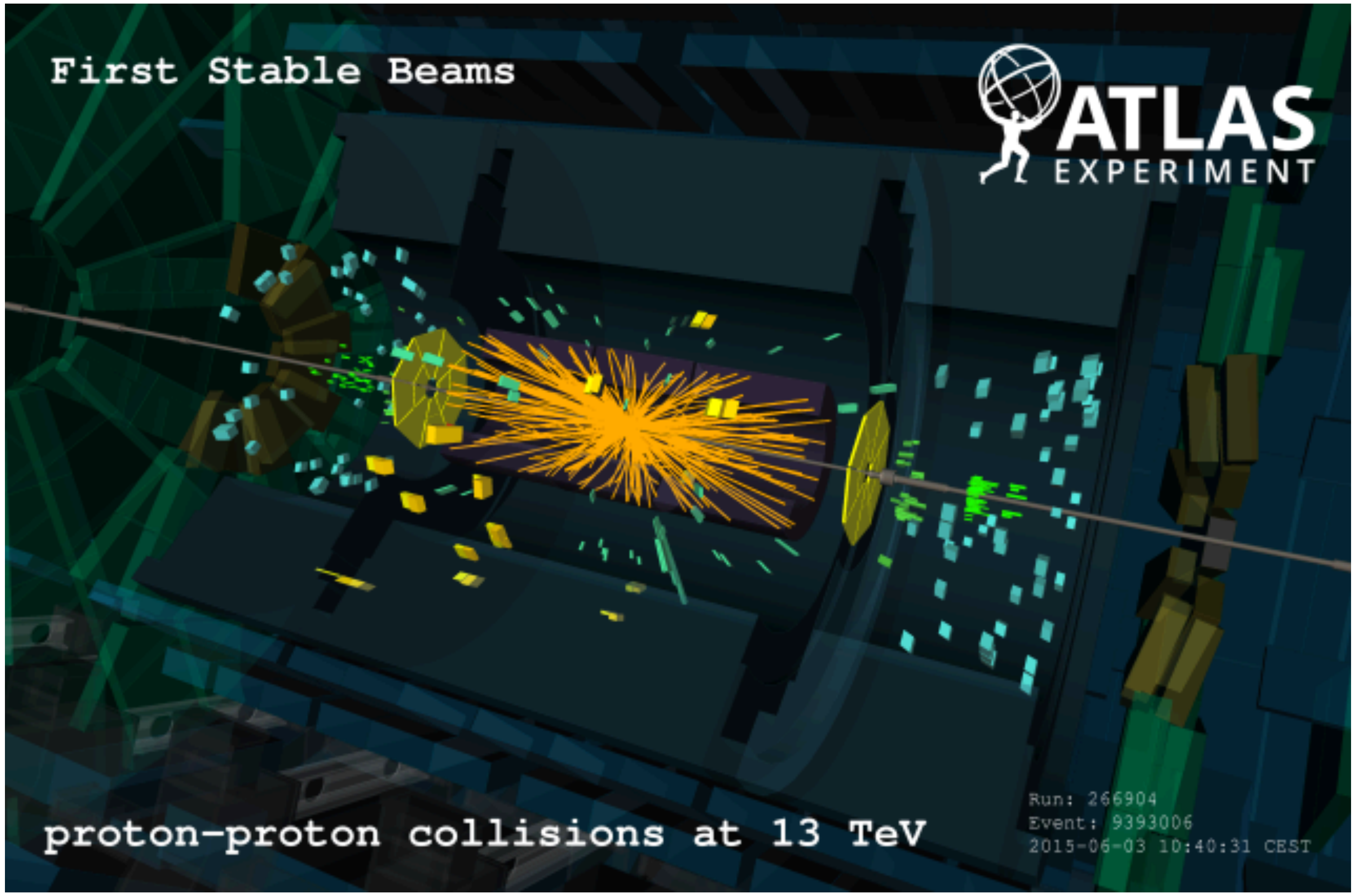
LHC and Big Data

- 600 million collisions per second
- Raw data rate ~ 1 PB/s (1 PB=10^6 GB)
- Actual data rate ~ 25 GB/s
  - Need to trigger on 1 out of 40,000 events



LHC and Big Data

- 600 million collisions per second
- Raw data rate ~ 1 PB/s (1 PB=10^6 GB)
- Actual data rate ~ 25 GB/s
    - Need to trigger on 1 out of 40,000 events
- ~ 10's of PB annually



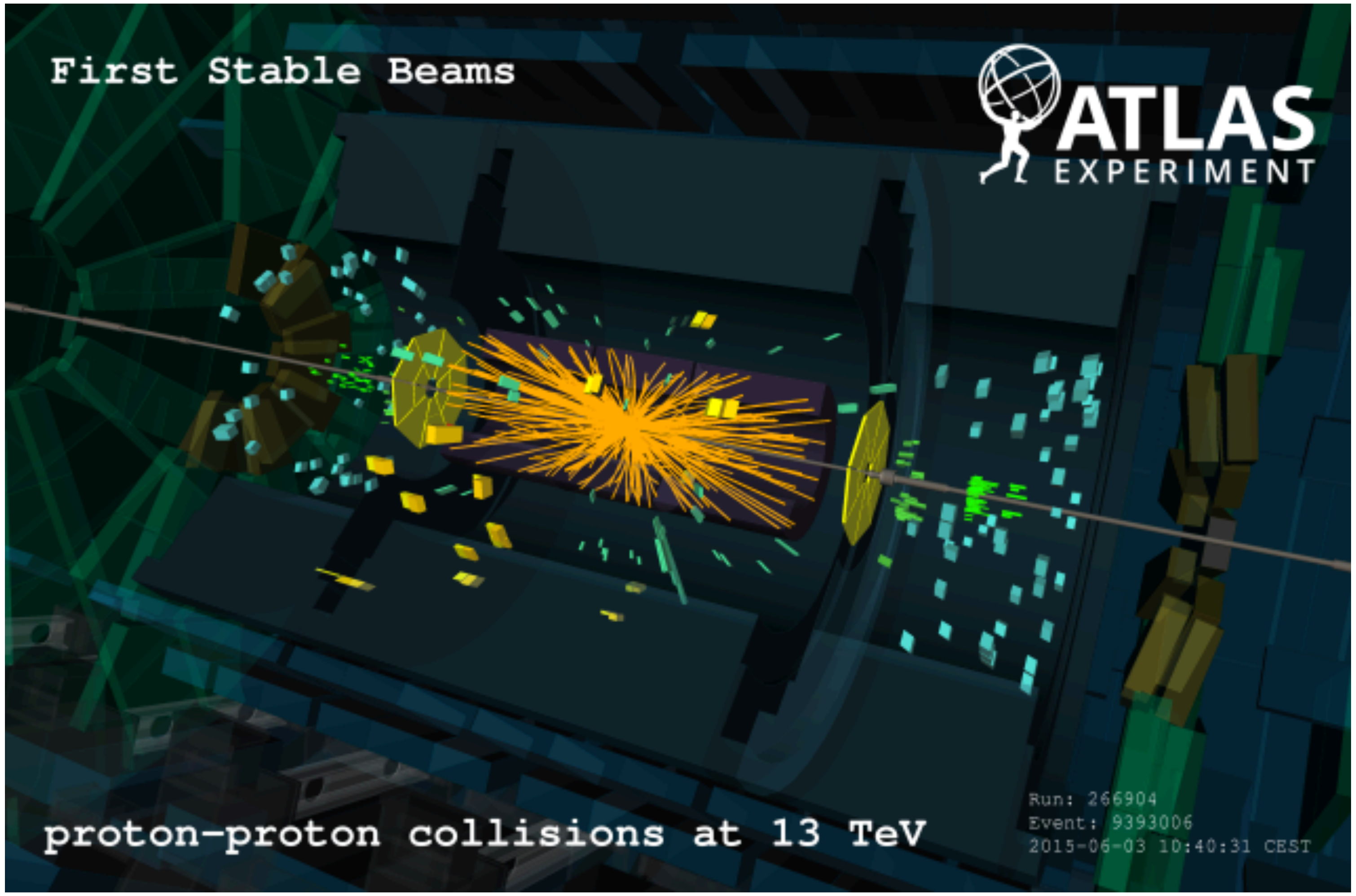
LHC and Big Data

- 600 million collisions per second
- Raw data rate ~ 1 PB/s (1 PB=10^6 GB)
- Actual data rate ~ 25 GB/s
    - Need to trigger on 1 out of 40,000 events
- ~ 10's of PB annually

The data is

- large (billions of events on tape)
- complex (hundreds of particles per event)
- well-understood (Standard Model of particle physics).

Also, it is relatively easy to generate realistic simulated data.



LHC and Big Data

The data is

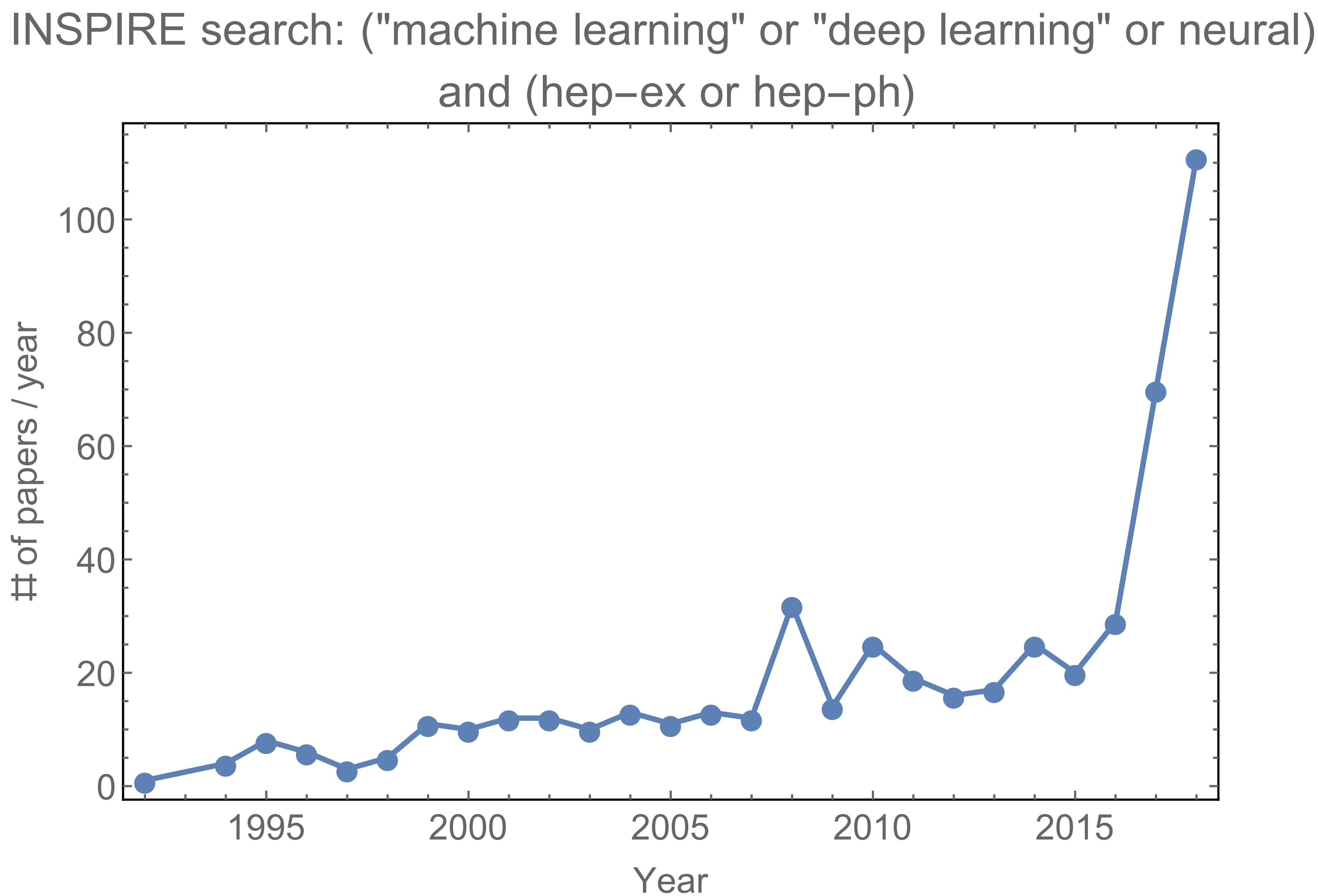
- large (billions of events on tape)
- complex (hundreds of particles per event)
- well-understood (Standard Model of particle physics).

Also, it is relatively easy to generate realistic simulated data.

- 600 million collisions per second
- Raw data rate ~ 1 PB/s (1 PB=10^6 GB)
- Actual data rate ~ 25 GB/s
  - Need to trigger on 1 out of 40,000 events
- ~ 10's of PB annually

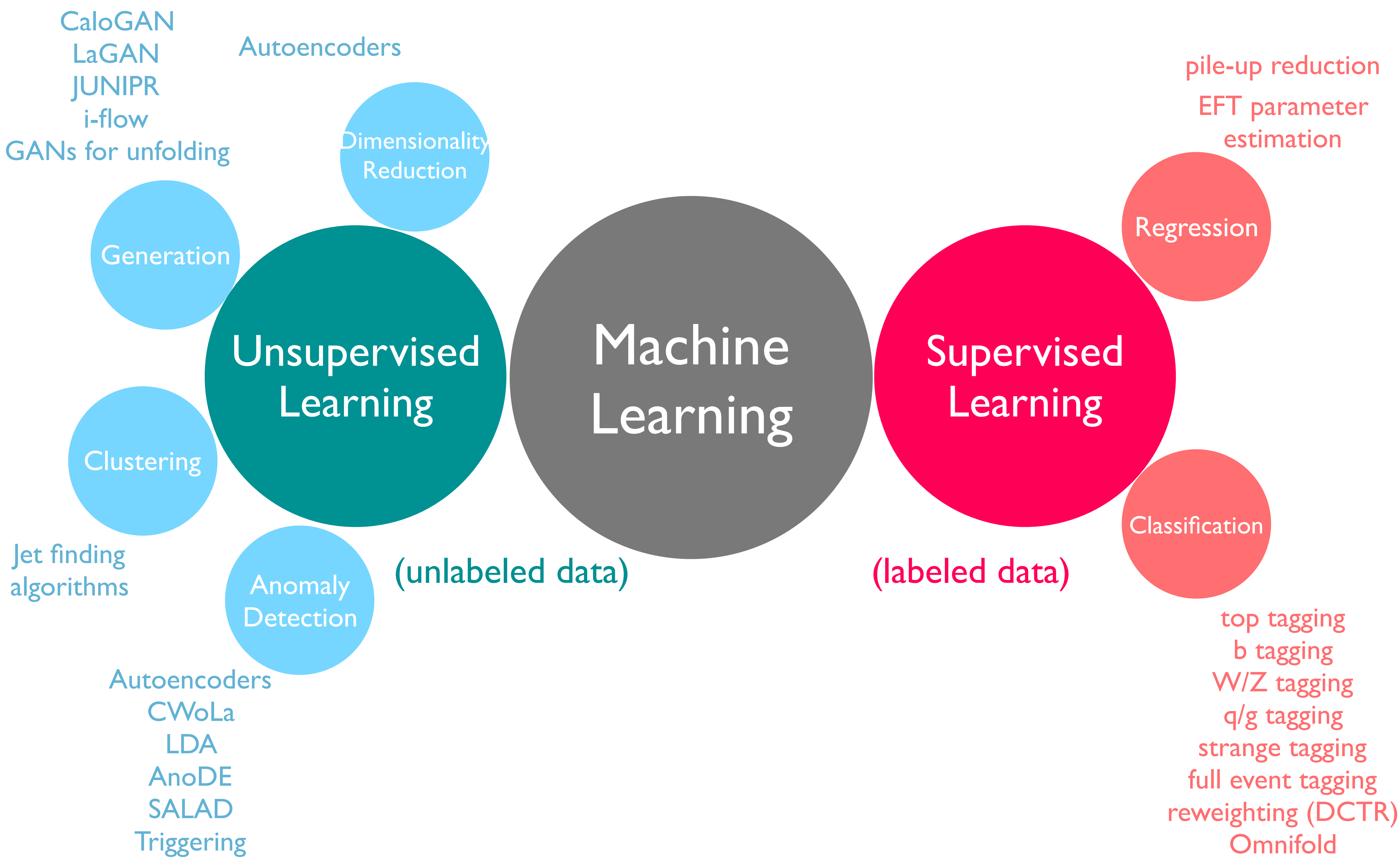
The LHC is a great setting for deep learning!

Deep Learning Papers

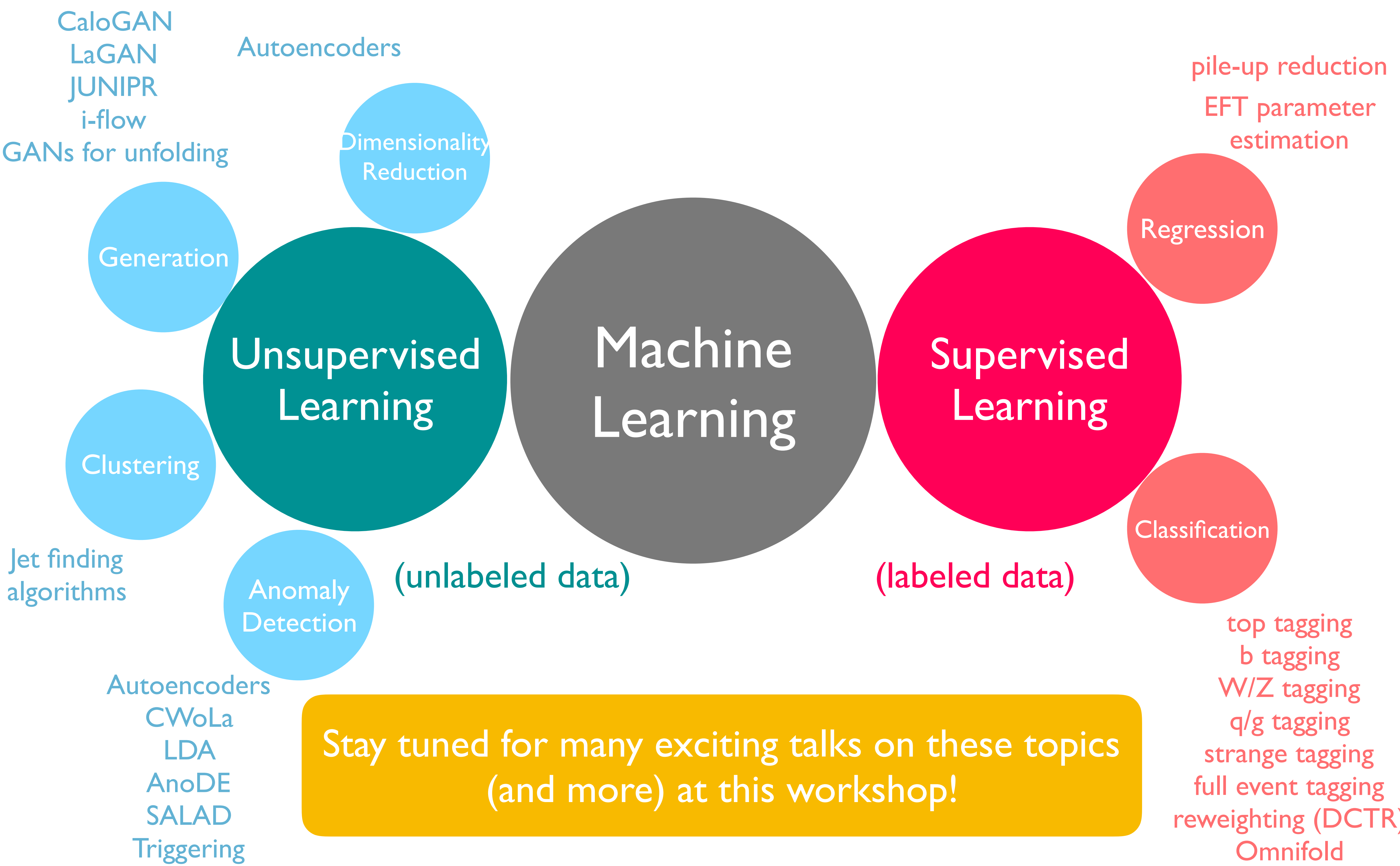


An explosion of interest in machine learning!

The Landscape of DL @ LHC

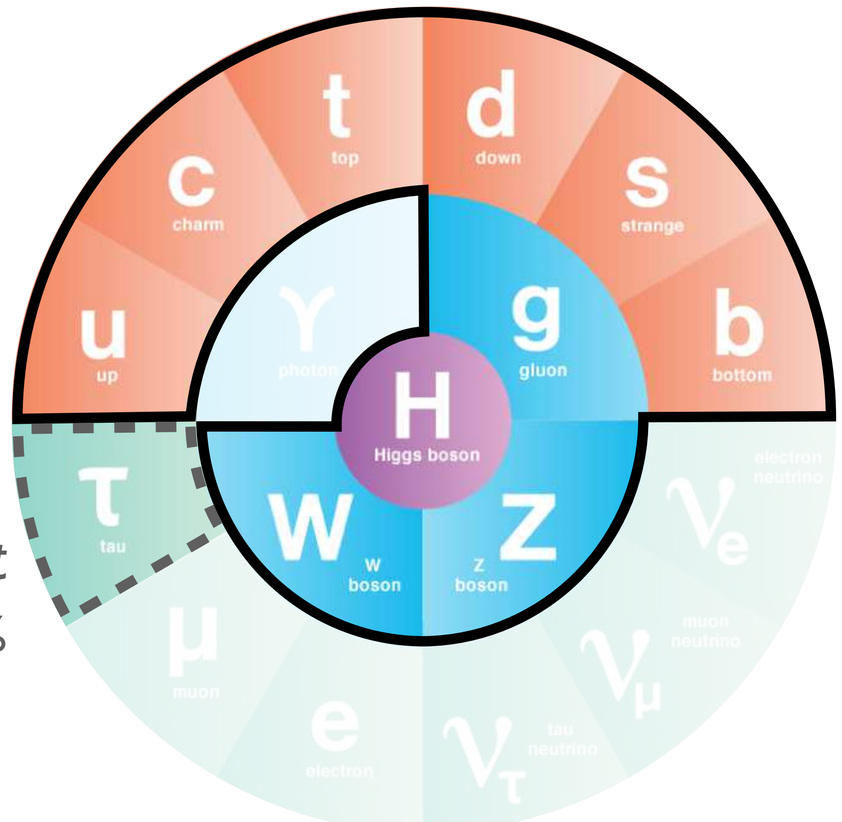


The Landscape of DL @ LHC



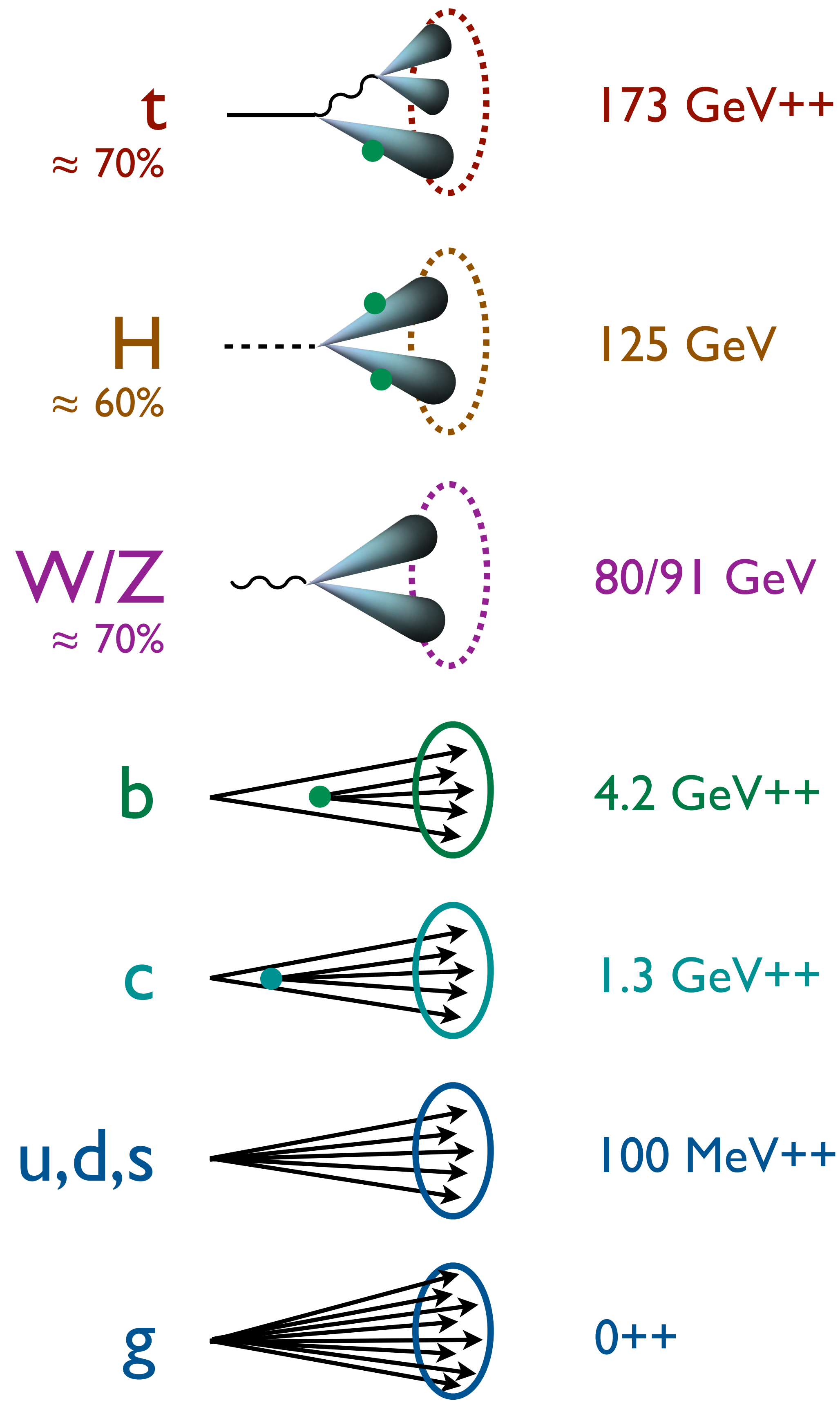
See talks by Michael Kagan and Tilman Plehn

Jet classification (“tagging”)



micro-jet
≈ 65%

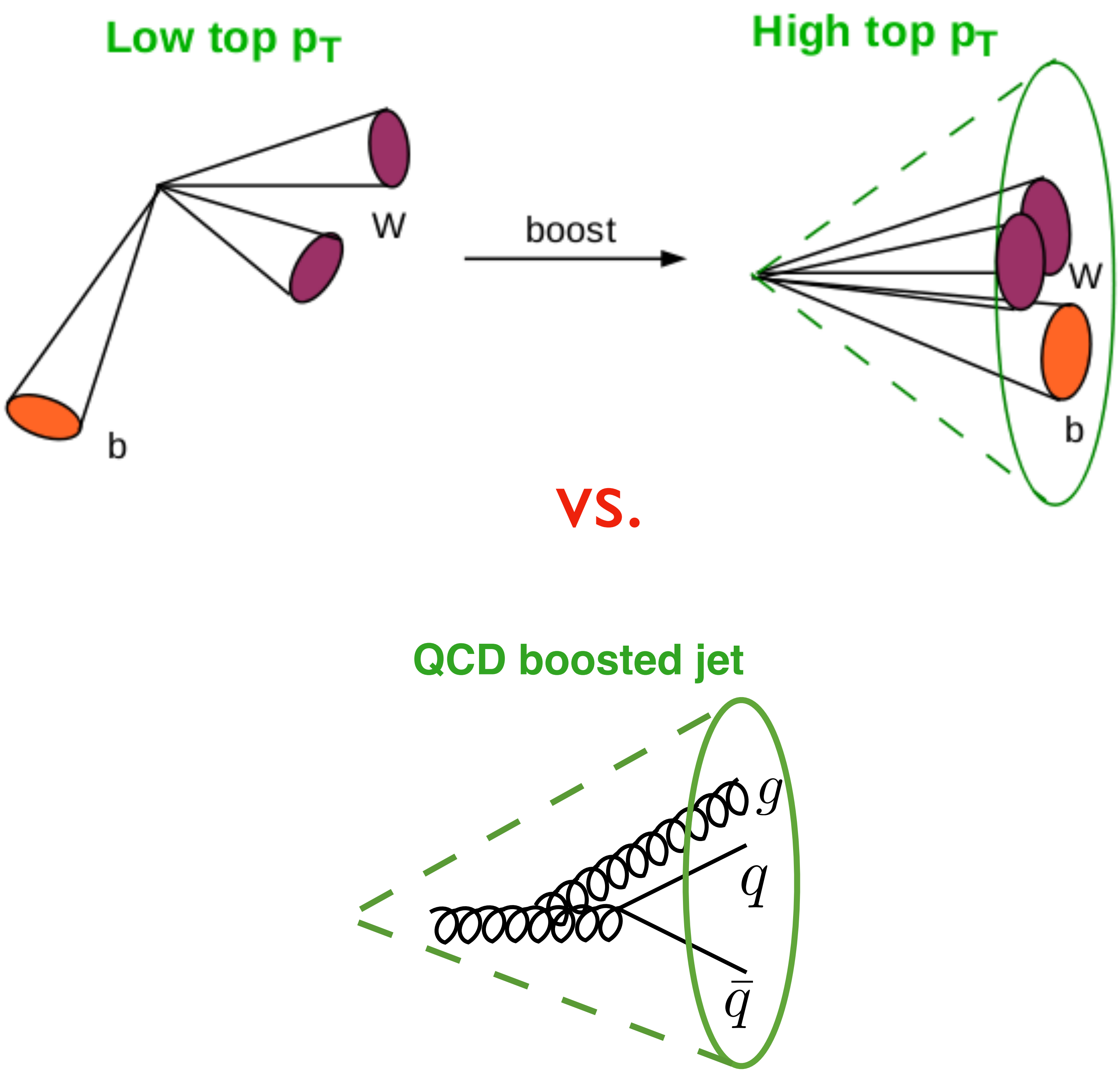
Jets from QCD and the Standard Model



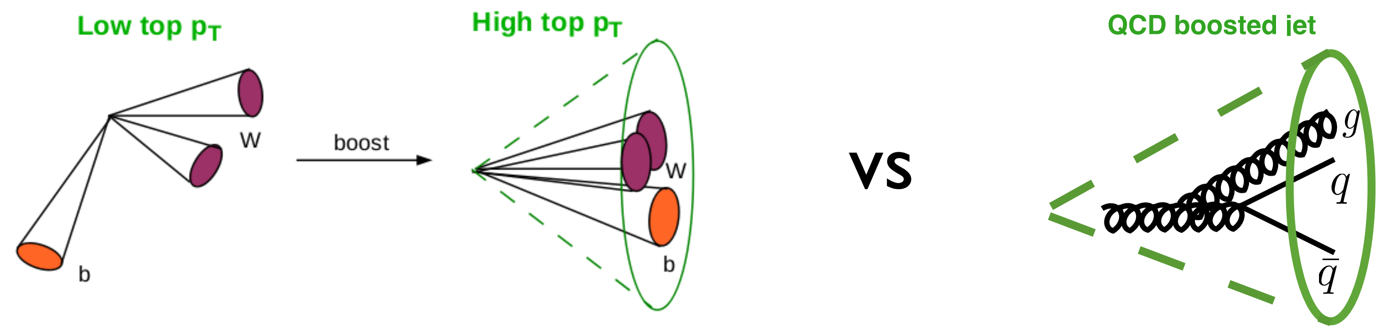
++ = Mass from QCD Radiation

slide credit: J. Thaler

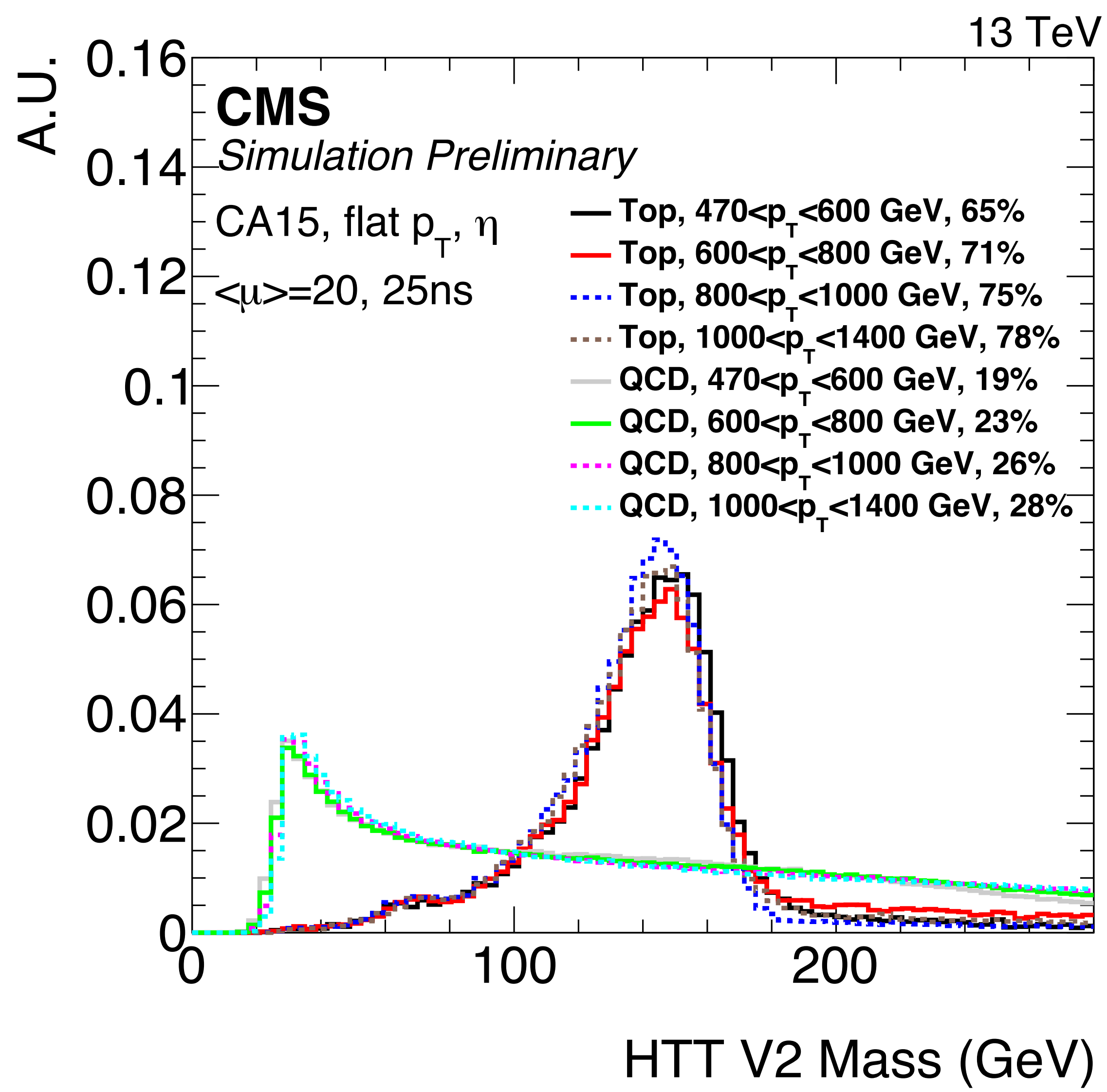
A popular example: boosted top tagging



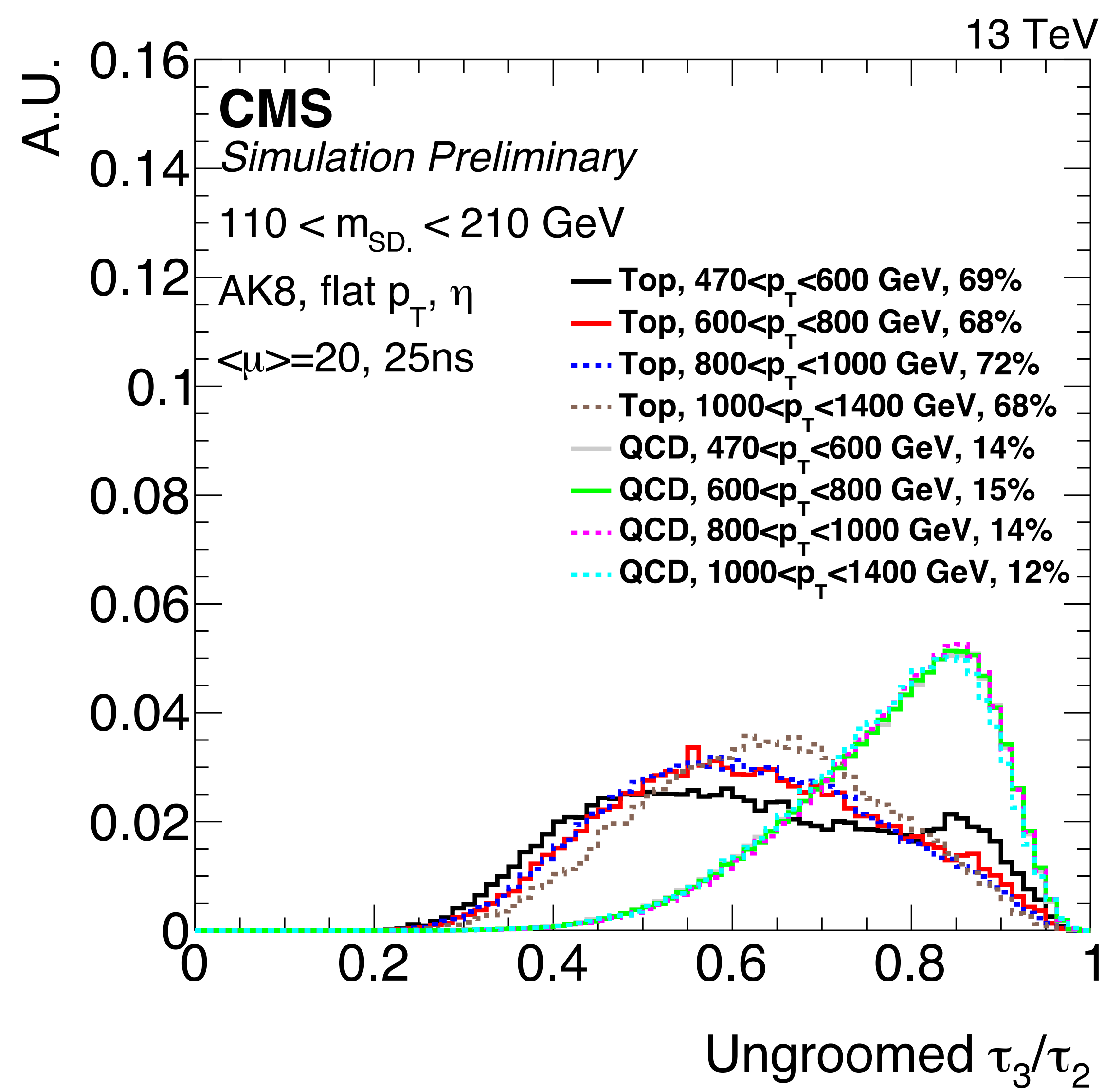
How to differentiate between these two types of jets?



Some obvious ideas:



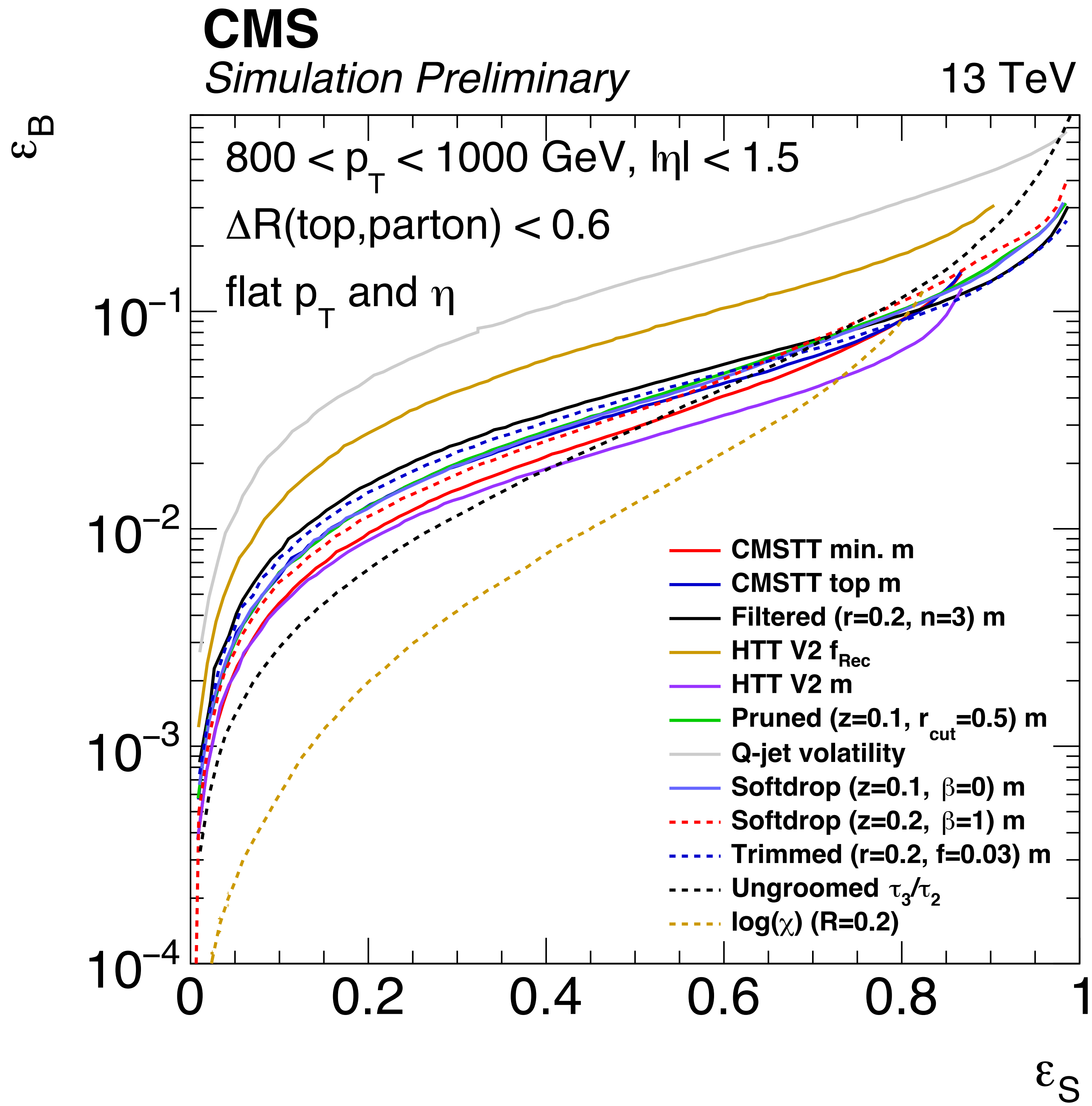
jet mass (m_{top} vs 0)



jet substructure (3 vs 1)

State of the art with cuts on kinematic quantities:

QCD jet mistag rate



"ROC curve"

top tagging efficiency

Can deep learning do better??

Jet images

Cogan et al 1407.5675, Almeida et al 1501.05968, de Olivera et al 1511.05190

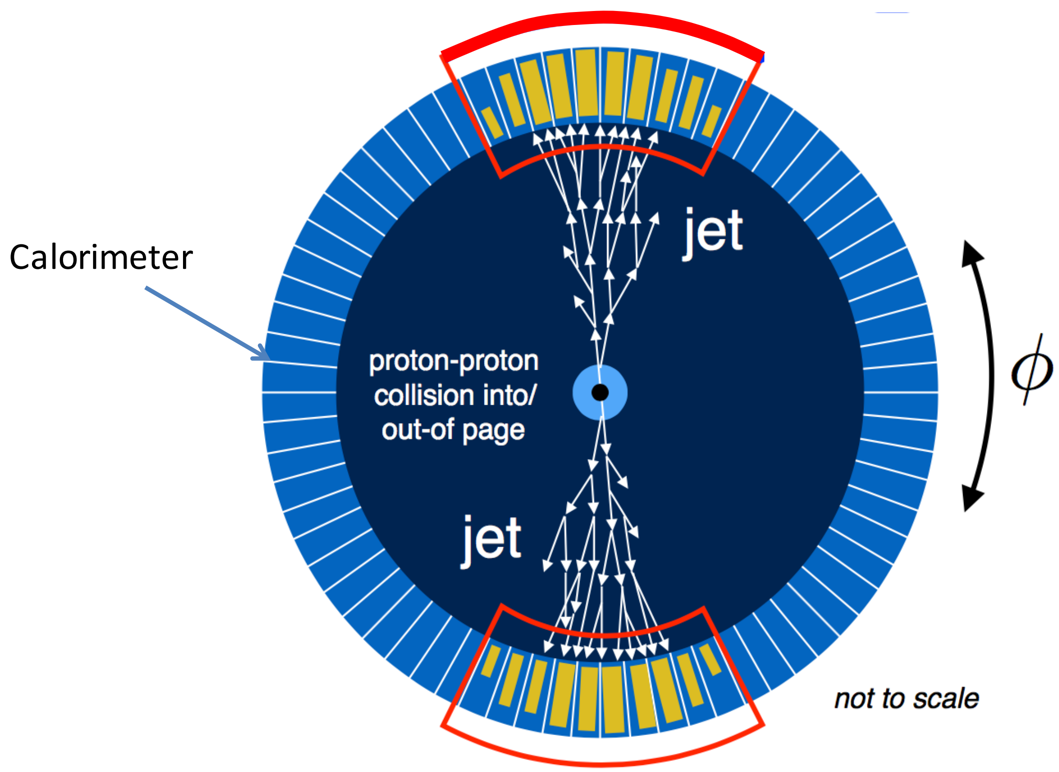
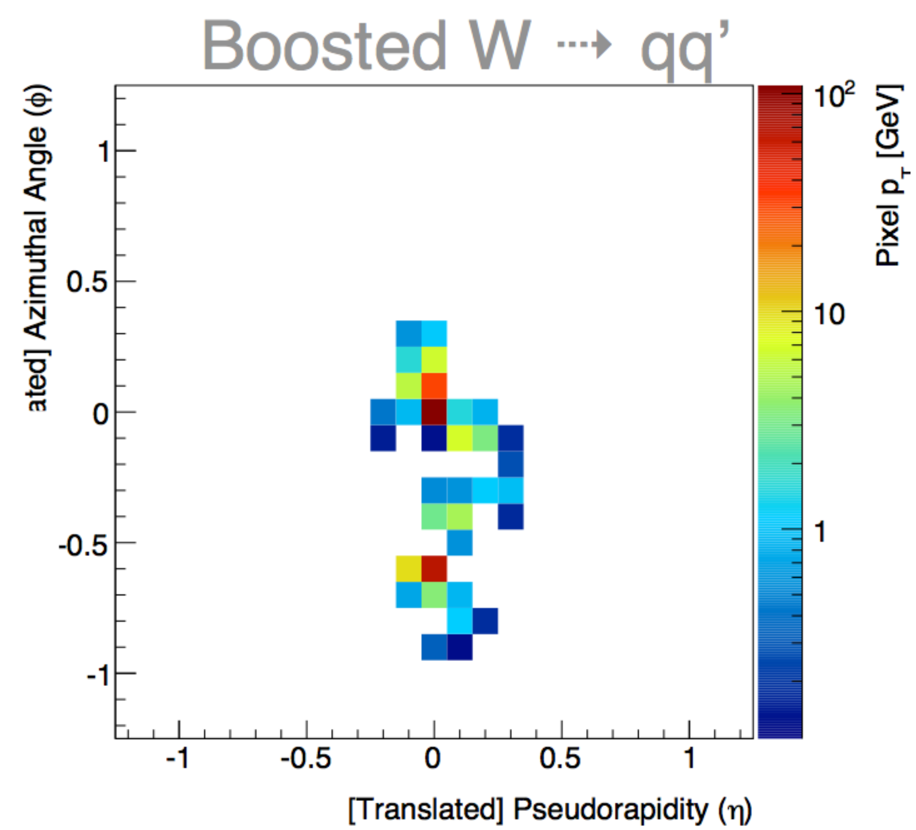


Figure credit:
B. Nachman



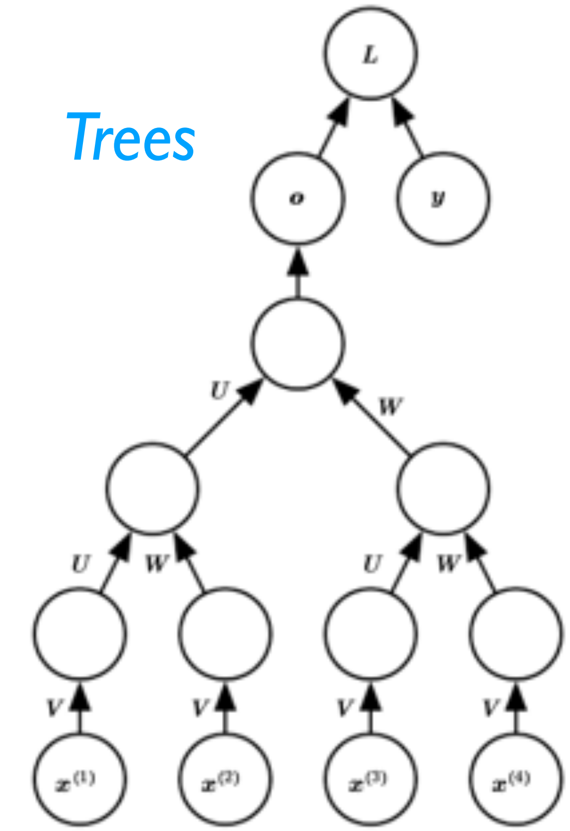
Jets are naturally images in eta and phi.

Should be able to apply "off-the-shelf" NNs developed for image recognition to classify jets at the LHC! de Oliveira et al 1511.05190

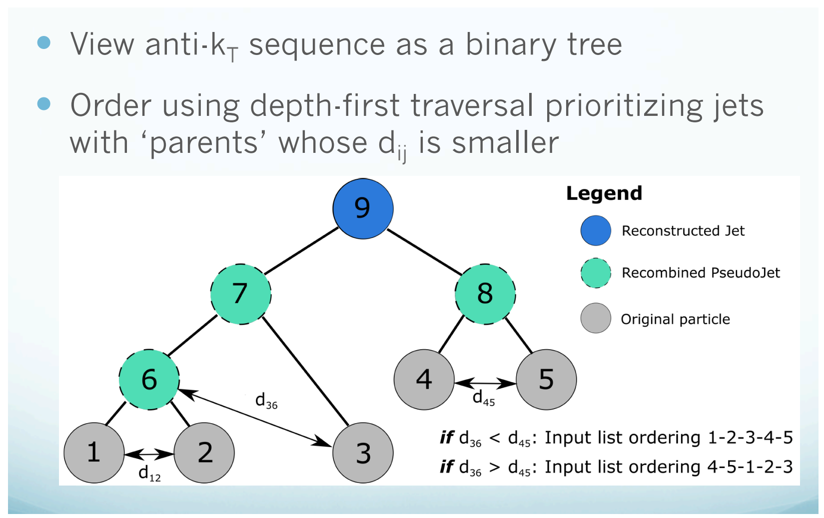
Other jet representations

Many other ways to represent a jet besides jet images!

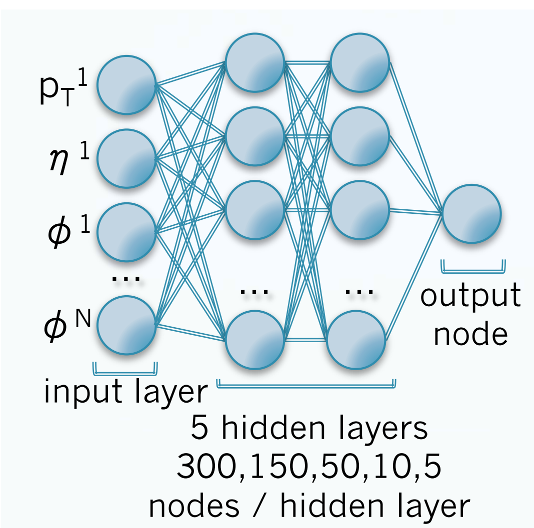
Trees



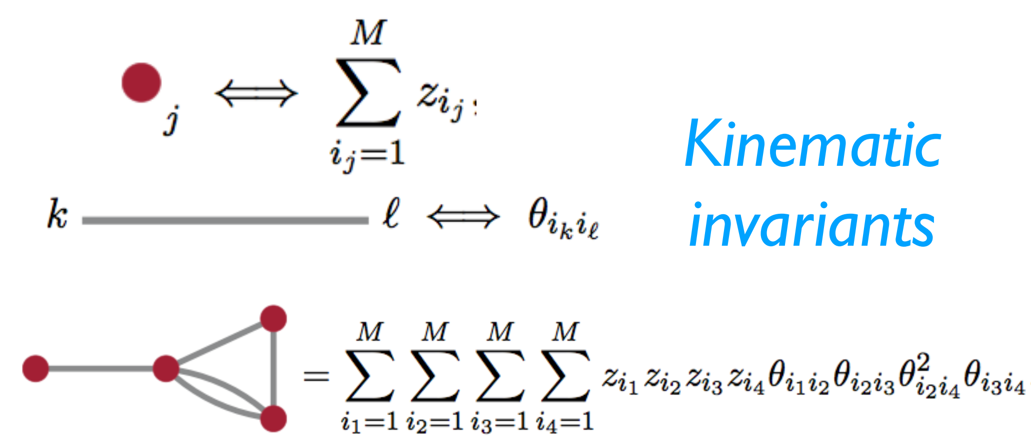
Sequences

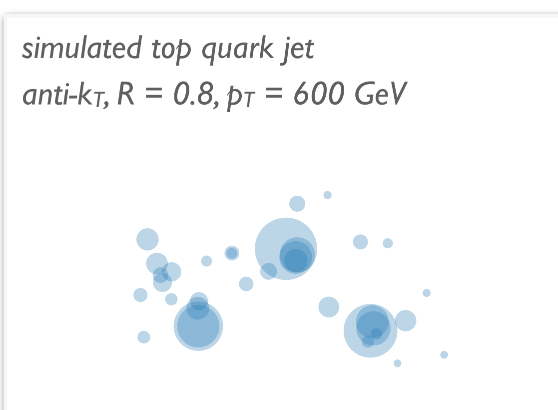


Lists of 4-vectors

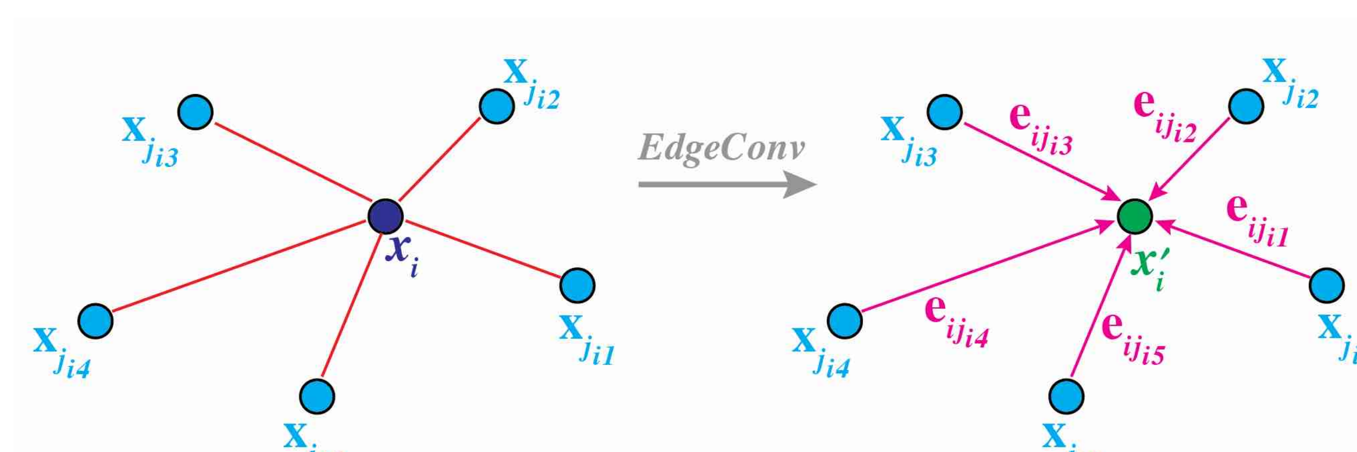


Kinematic invariants





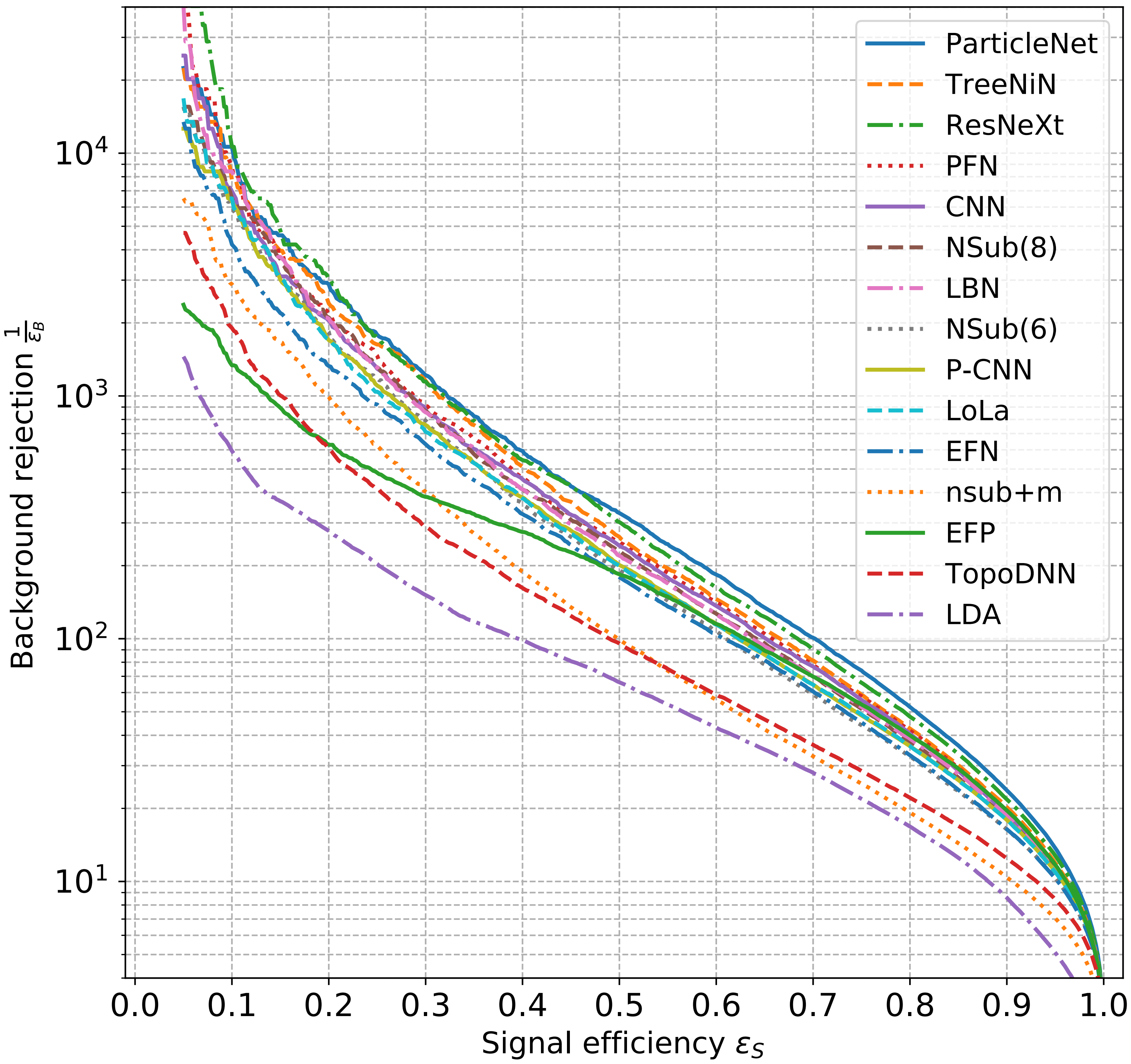
Point clouds



Graphs

Community top tagging comparison

Kasieczka, Plehn et al 1902.09914



Q: There are many papers developing jet taggers with different jet representations and architectures. How can we evaluate their relative strengths and weaknesses?

A: Let's perform an apples-to-apples comparison of various top taggers on a common dataset!

Community top tagging comparison

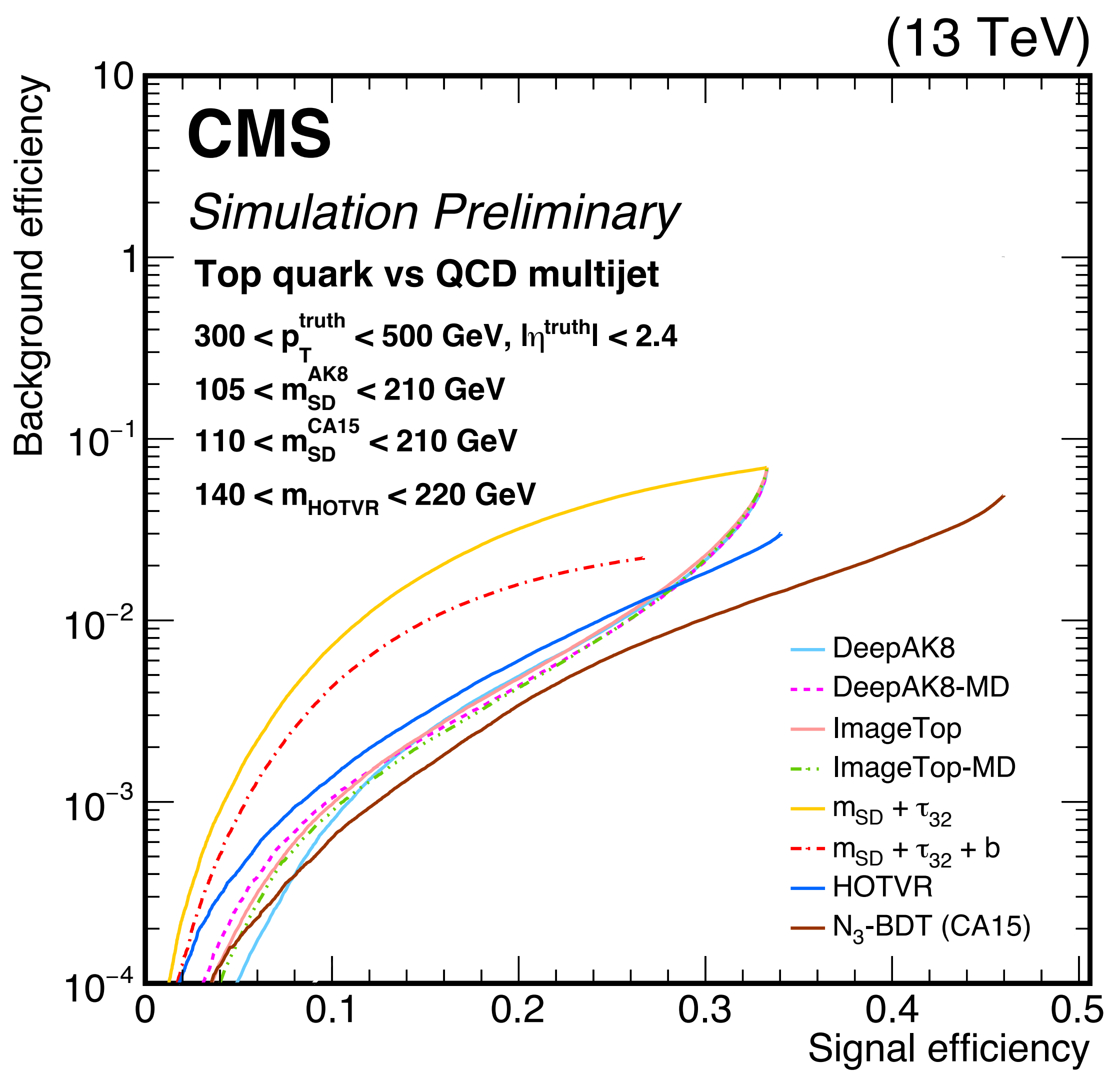
Kasieczka, Plehn et al 1902.09914

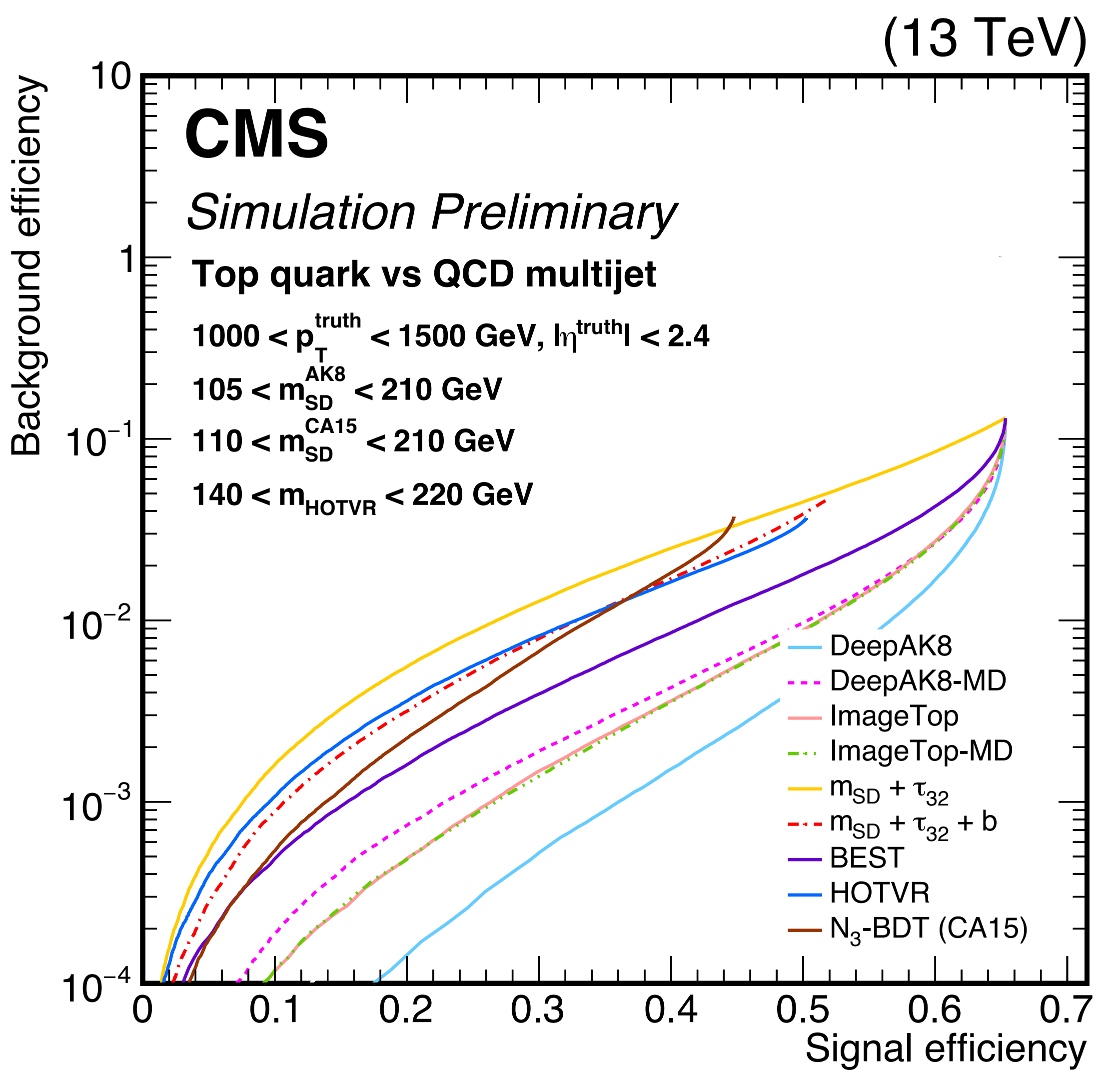
	AUC	Acc	$1/\epsilon_B$ ($\epsilon_S = 0.3$) single	mean	median	#Param
CNN [16]	0.981	0.930	914±14	995±15	975±18	610k
ResNeXt [31]	0.984	0.936	1122±47	1270±28	1286±31	1.46M
TopoDNN [18]	0.972	0.916	295±5	382± 5	378 ± 8	59k
Multi-body N-subjettiness 6 [24]	0.979	0.922	792±18	798±12	808±13	57k
Multi-body N-subjettiness 8 [24]	0.981	0.929	867±15	918±20	926±18	58k
TreeNiN [43]	0.982	0.933	1025±11	1202±23	1188±24	34k
P-CNN	0.980	0.930	732±24	845±13	834±14	348k
ParticleNet [47]	0.985	0.938	1298±46	1412±45	1393±41	498k
LBN [19]	0.981	0.931	836±17	859±67	966±20	705k
LoLa [22]	0.980	0.929	722±17	768±11	765±11	127k
LDA [54]	0.955	0.892	151±0.4	151.5±0.5	151.7±0.4	184k
Energy Flow Polynomials [21]	0.980	0.932	384			1k
Energy Flow Network [23]	0.979	0.927	633±31	729±13	726±11	82k
Particle Flow Network [23]	0.982	0.932	891±18	1063±21	1052±29	82k
GoaT	0.985	0.939	1368±140		1549±208	35k

Have we found the optimal tagger??

CMS performance study

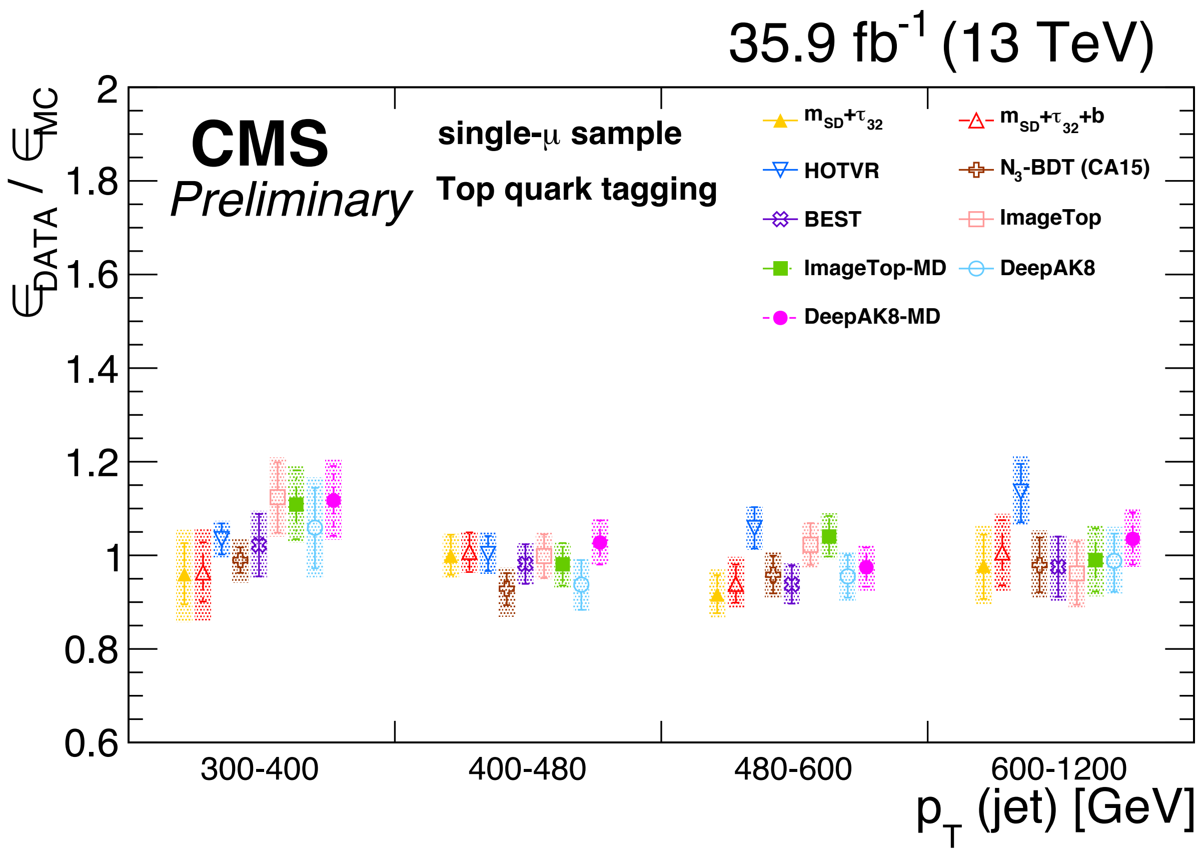
JME-18-002-PAS

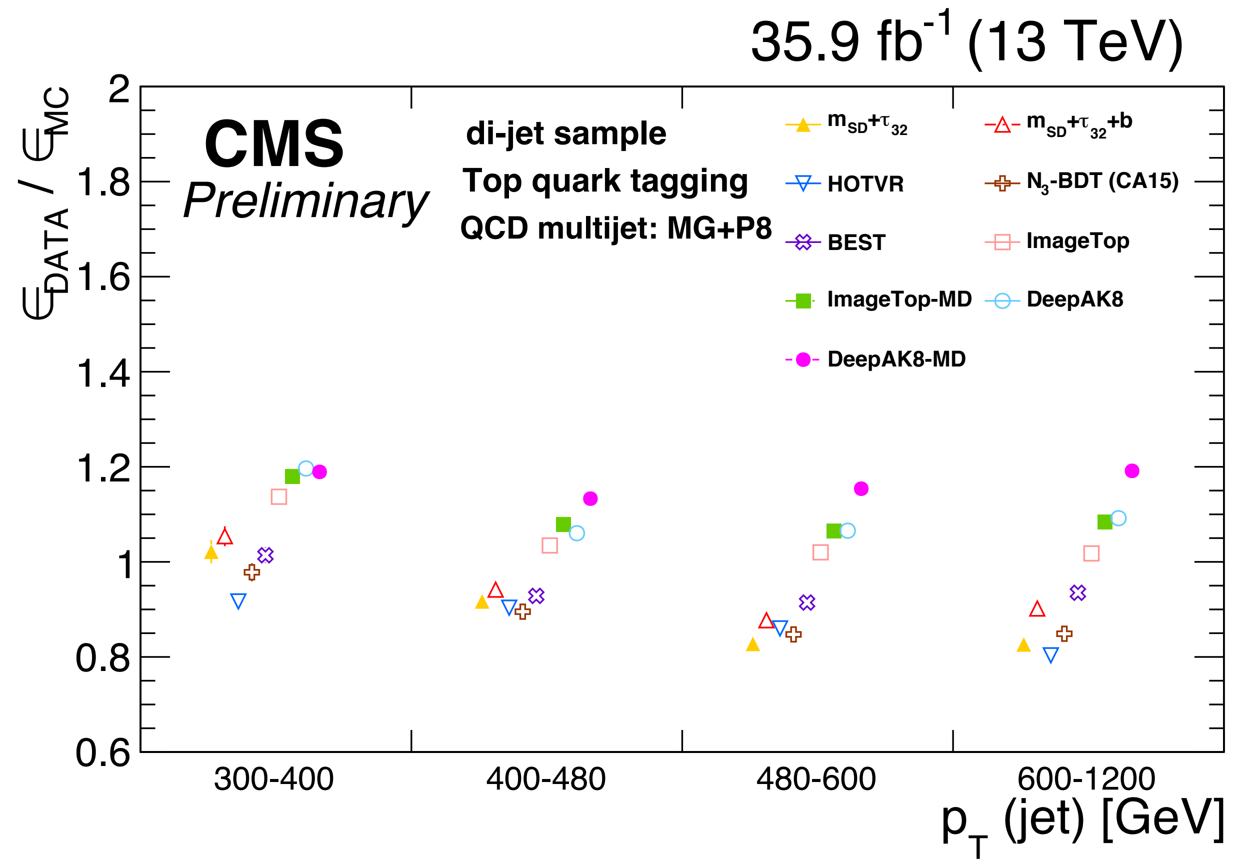




CMS performance study

JME-18-002-PAS





Performance measured in data!

Beyond top tagging

Many other classification tasks at LHC are enhanced with deep learning:

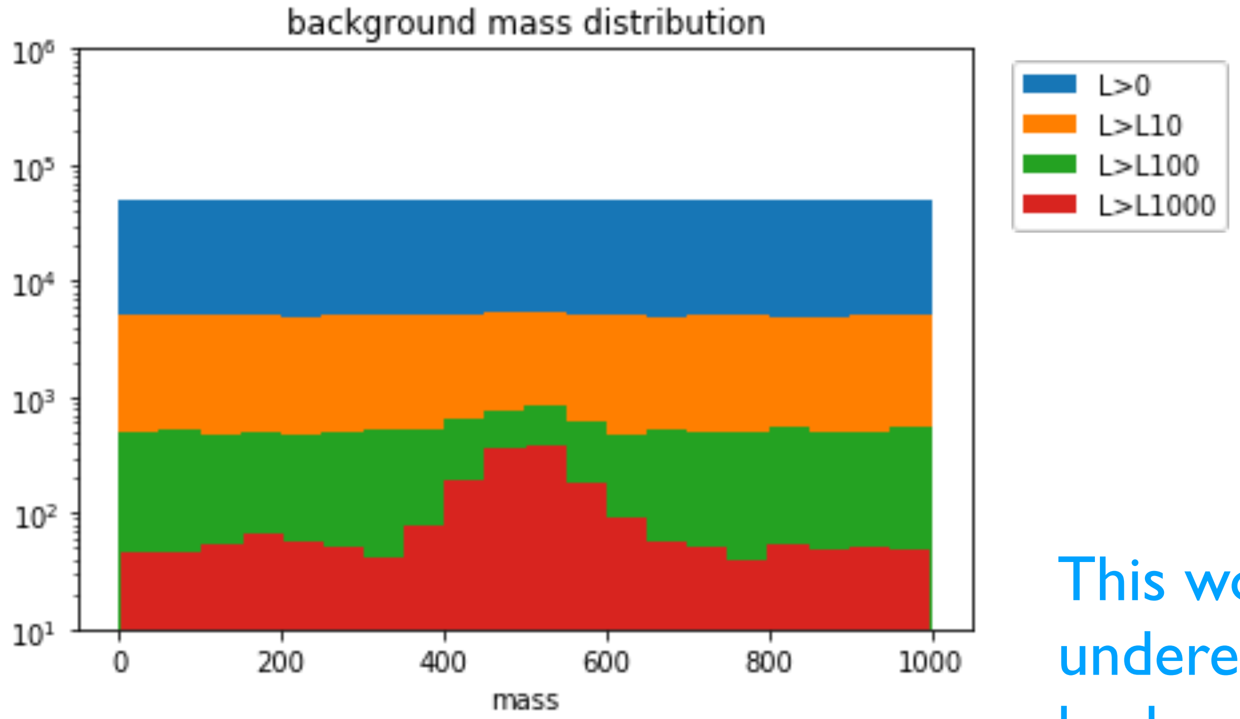
- quark/gluon tagging (Komiske, Metodiev & Schwartz '16, et seq)
- b and c tagging (DeepCSV)
- boosted W/Z tagging (Oliveira et al '15, et seq)
- strange tagging (Nakai, Thomas & DS to appear)
- u vs d tagging using jet charge (Fraser & Schwartz '18)
- boosted W+/Z/W- tagging using jet charge (Chen, Chiang, Cottin & DS 1908.08256)
- ….

See talks by Michael Kagan and DS

Beyond classification — decorrelation

Raw tagger performance not the only consideration.

For robust background estimation, generally need to ensure tagger does not sculpt the background mass distribution.



This would greatly underestimate the background in the SR

Two approaches with deep learning:

- adversarial decorrelation (Louppe et al 1611.01046, Shimmin et al 1703.03507)
- DisCo decorrelation (Kasieczka & DS 2001.05310)

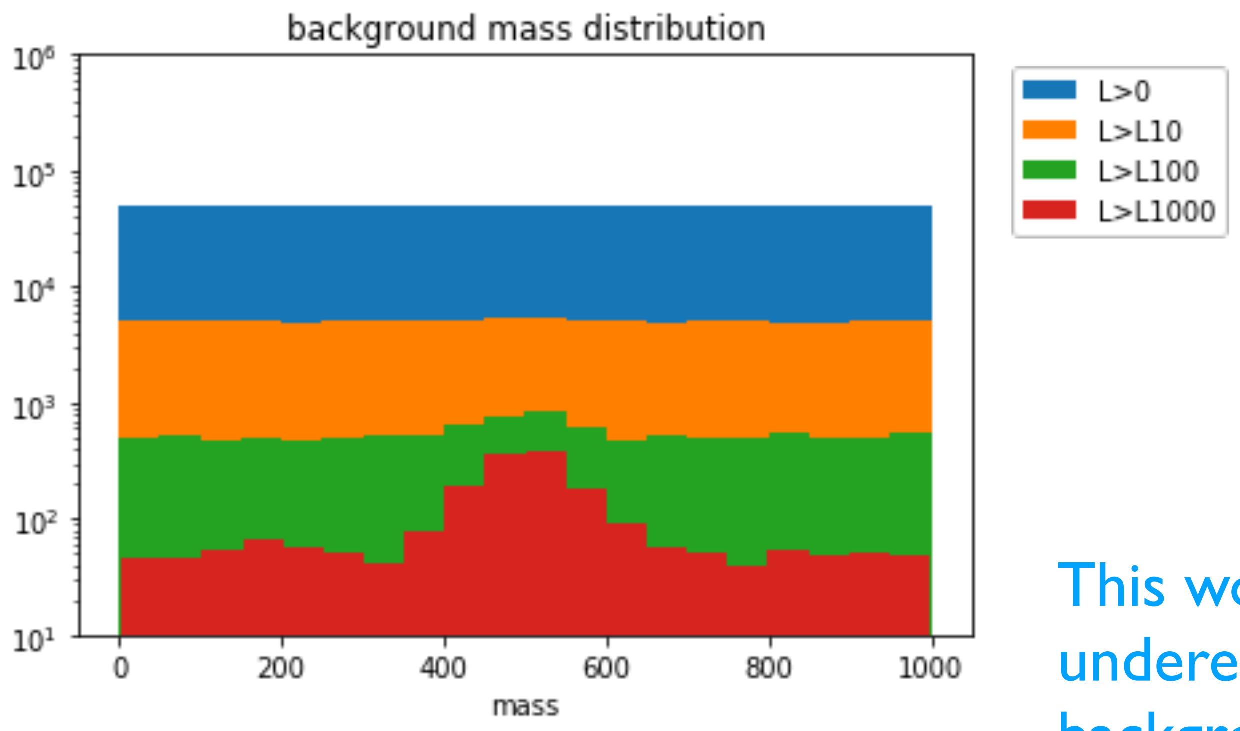
See talks by Michael Kagan and DS

Beyond classification — decorrelation

Could have real-world applications, e.g. in designing fairer AIs?

Raw tagger performance not the only consideration.

For robust background estimation, generally need to ensure tagger does not sculpt the background mass distribution.



This would greatly underestimate the background in the SR

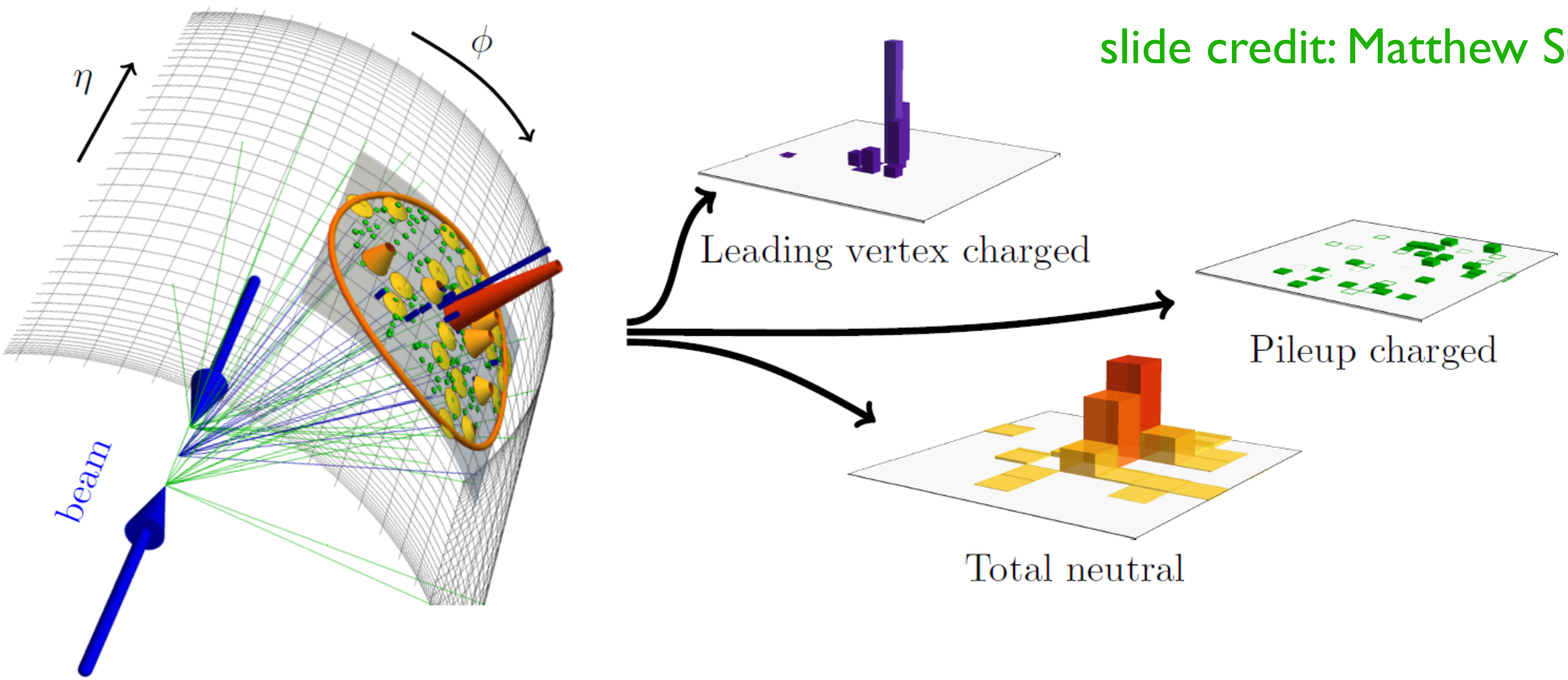
Two approaches with deep learning:

- adversarial decorrelation (Louppe et al 1611.01046, Shimmin et al 1703.03507)
- DisCo decorrelation (Kasieczka & DS 2001.05310)

Beyond classification — pileup regression

PileUp Mitigation with Machine Learning (PUMML)
(Komiske, Metodiev, Nachman & Schwartz 1707.08600)

slide credit: Matthew Schwartz



Can measure

1. Leading vertex charged particles
2. Pileup charged particles
3. Total neutral particles

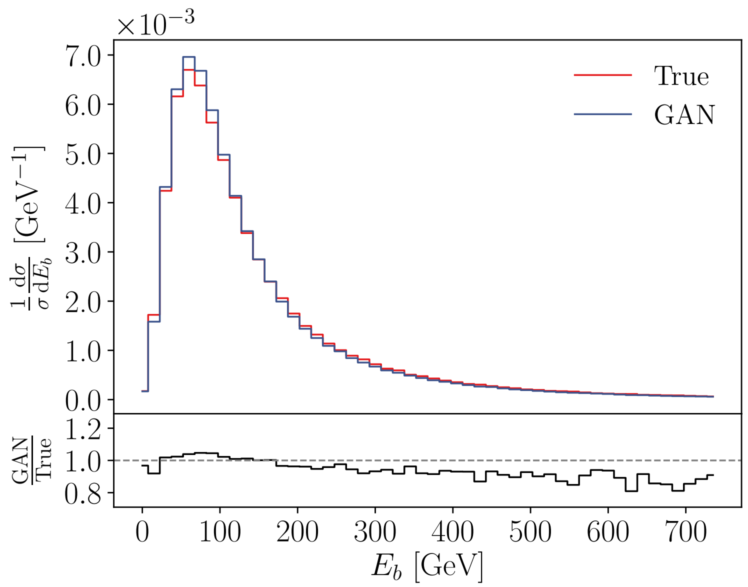
?

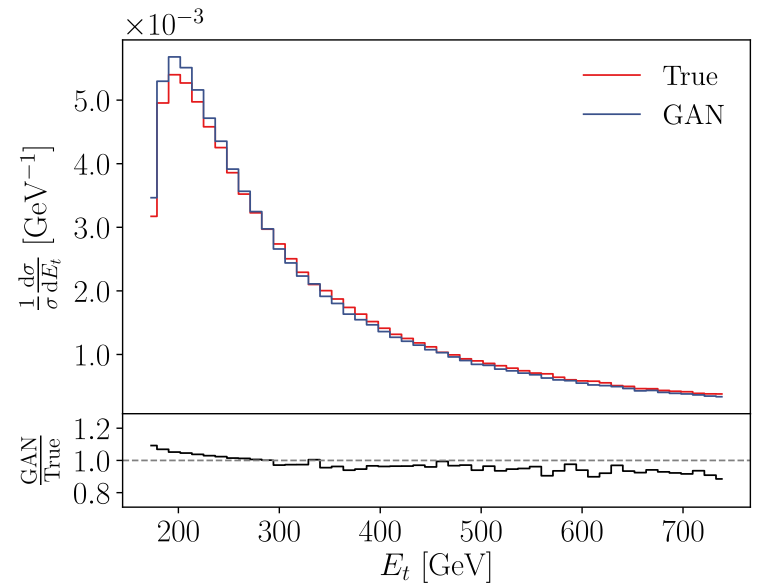
Leading vertex neutral particles

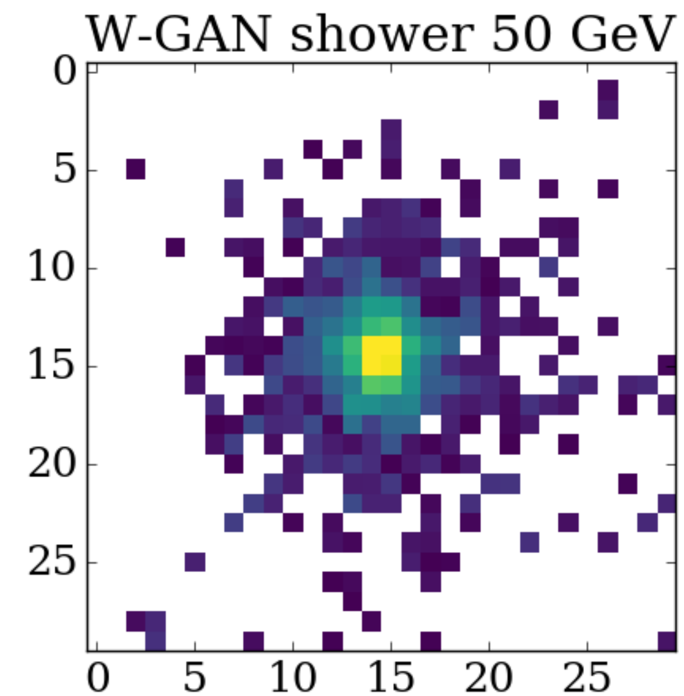
See Anja Butter's talk

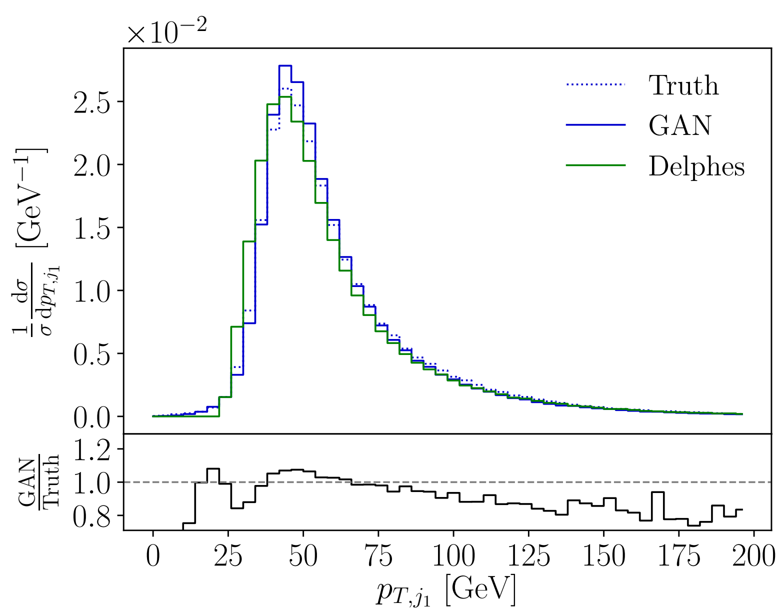
Beyond classification — G$_{\text{erm}}$ANs for everything

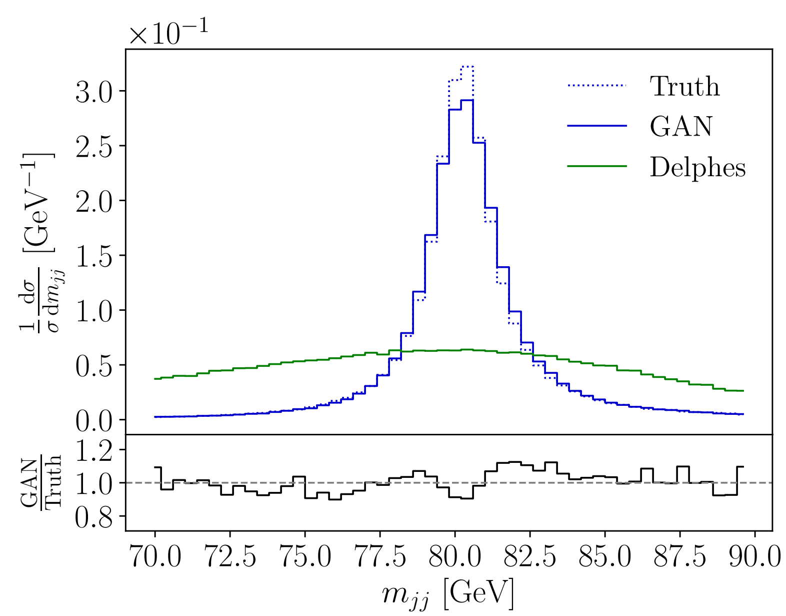
Can use GANs for event generation, fast detector simulation, unfolding, …







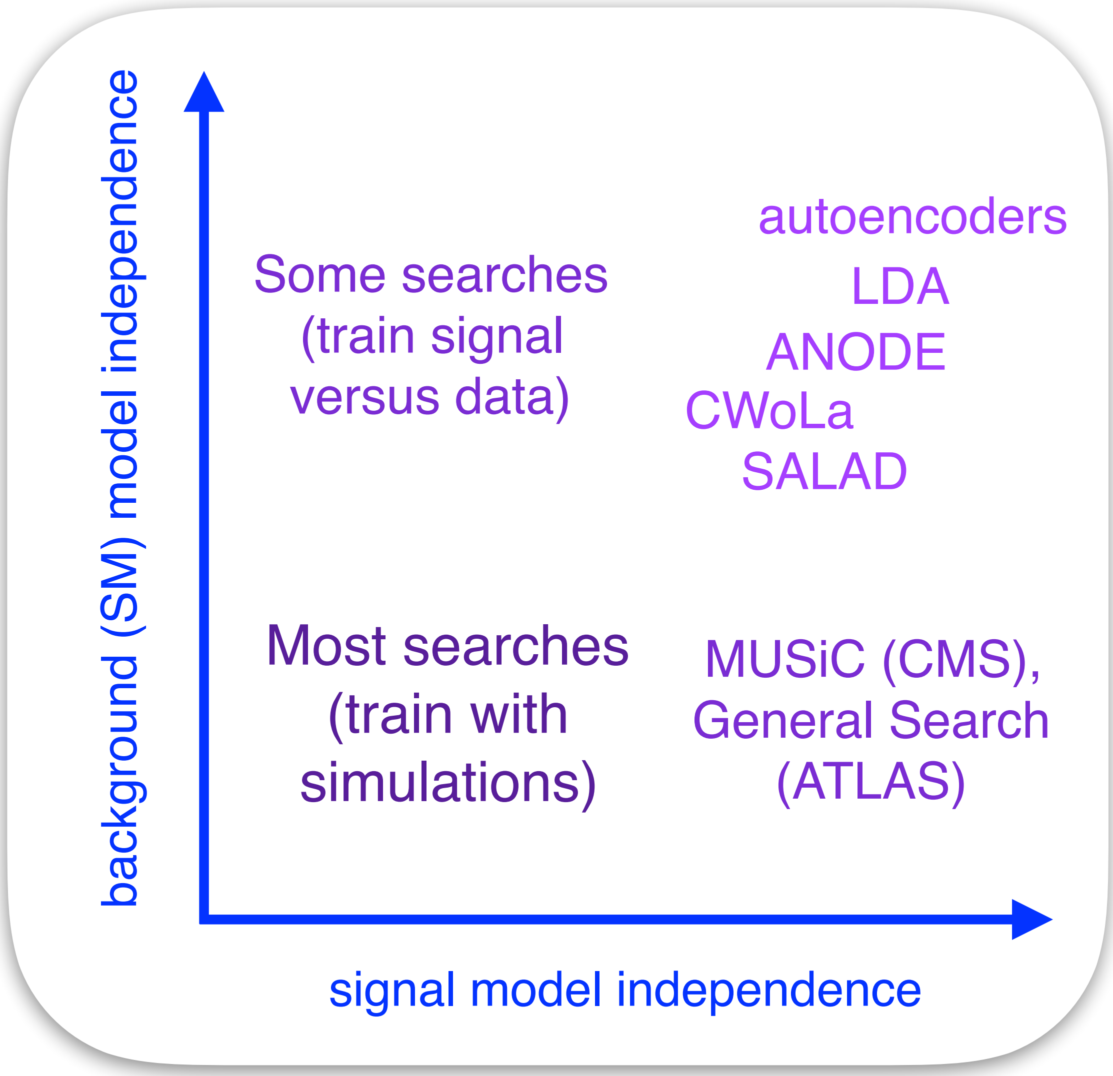




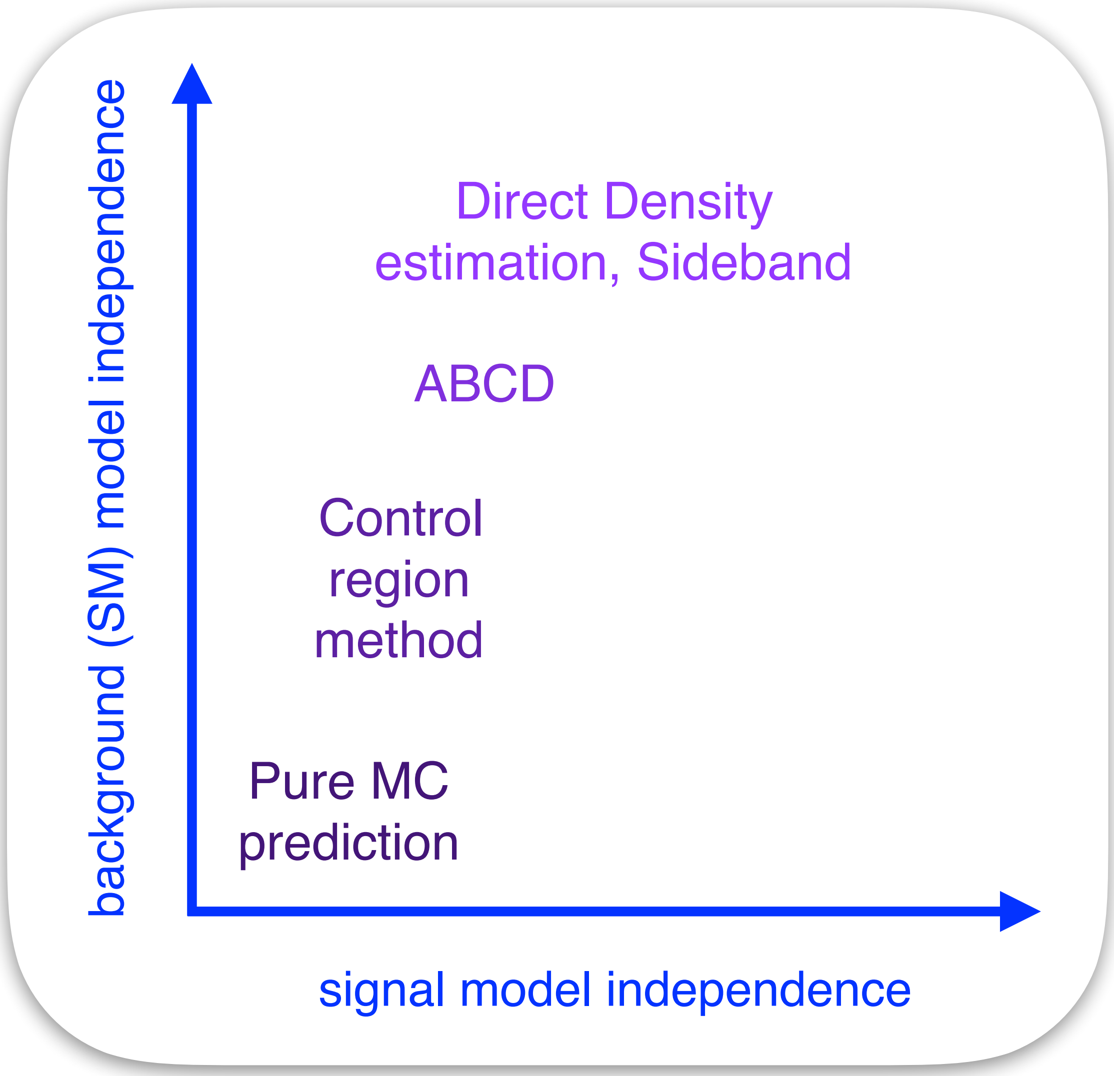
See talks by Tao Liu, Ben Nachman and DS

Way beyond classification: anomaly detection

from Nachman & DS 2001.04990



(a) Signal sensitivity

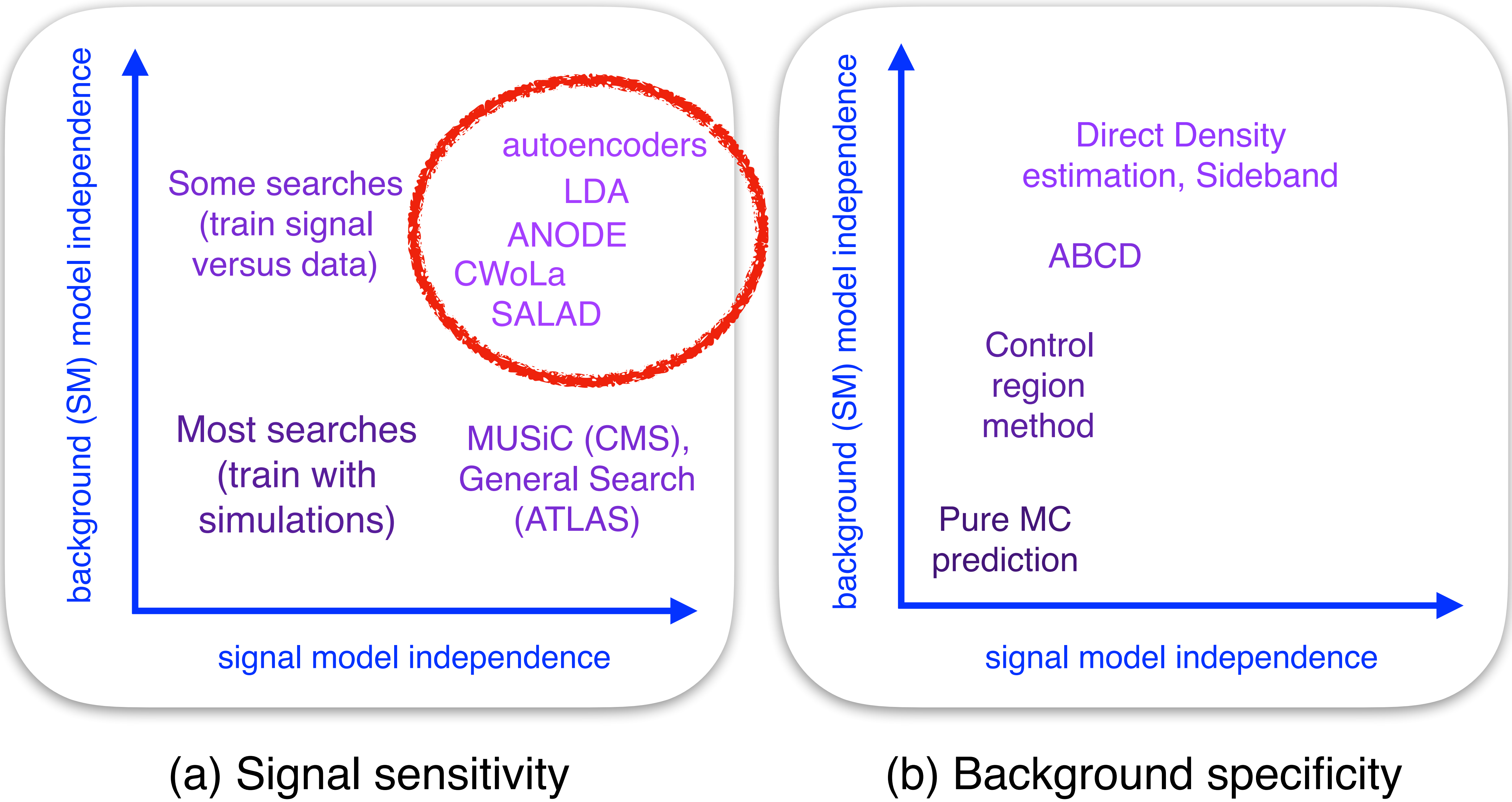


(b) Background specificity

See talks by Tao Liu, Ben Nachman and DS

Way beyond classification: anomaly detection

from Nachman & DS 2001.04990



Many interesting new ideas for model-independent searches being proposed

Summary/Outlook

Deep learning is revolutionizing nearly every aspect of high-energy physics.

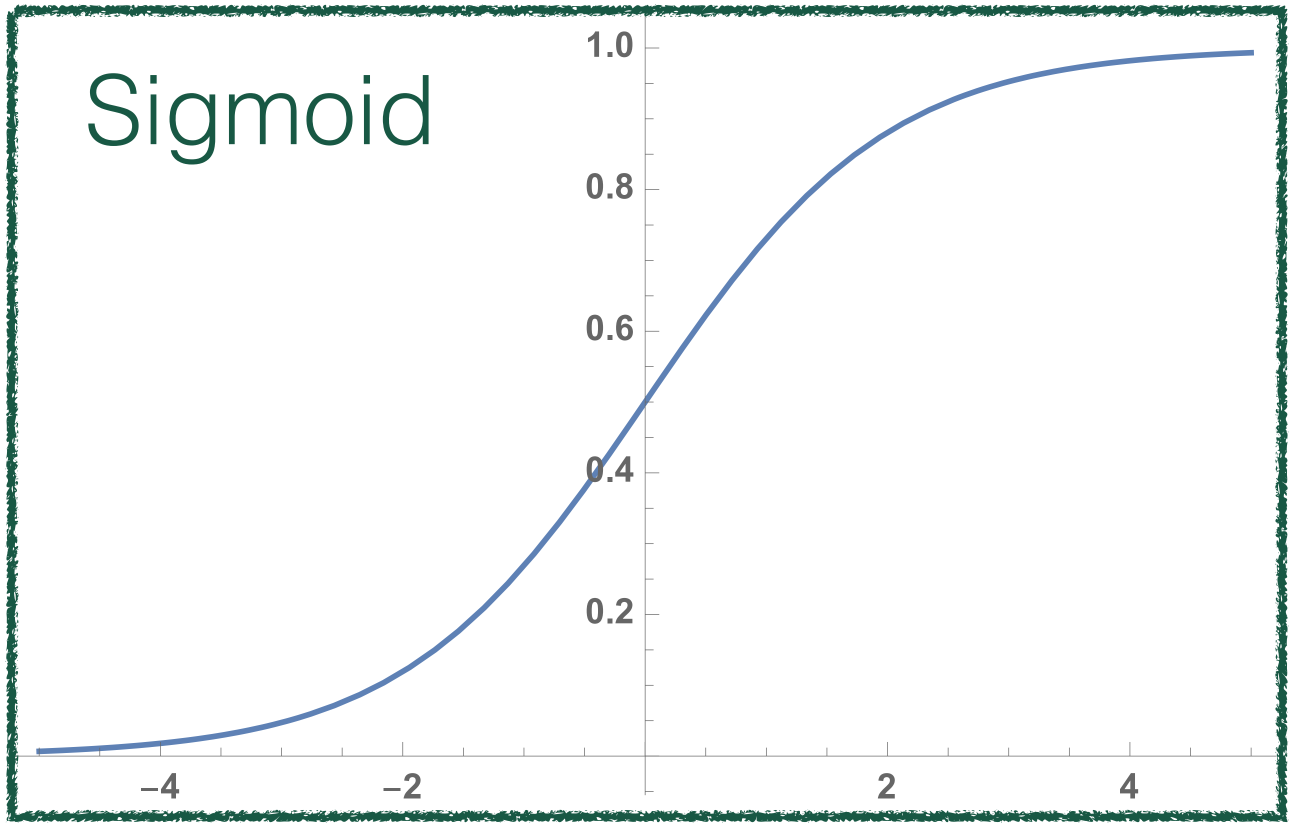
- Tagging
- Pileup
- Event generation
- Detector simulation
- Triggering
- Searches for NP
- ….

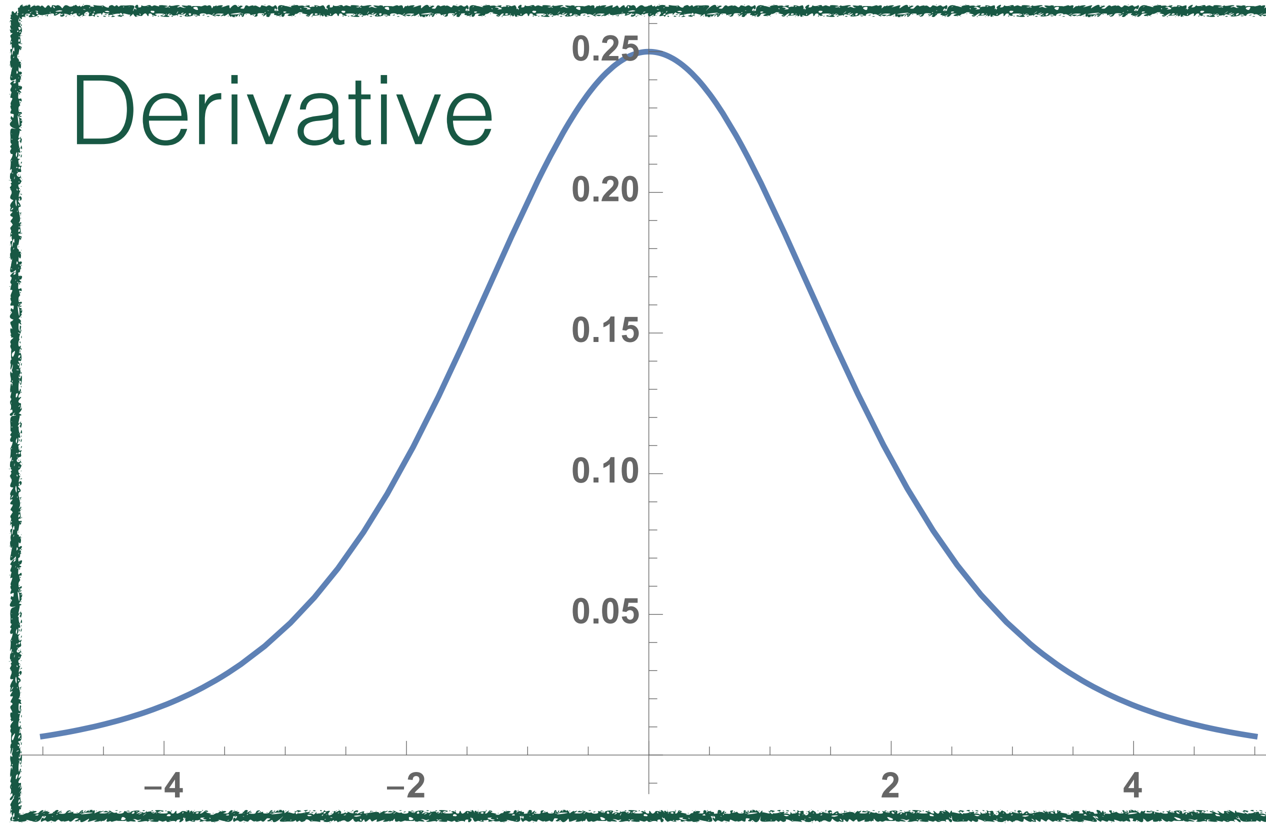
Exciting times are ahead!

Thanks for your attention!

Supplementary material

Disappearing Gradient





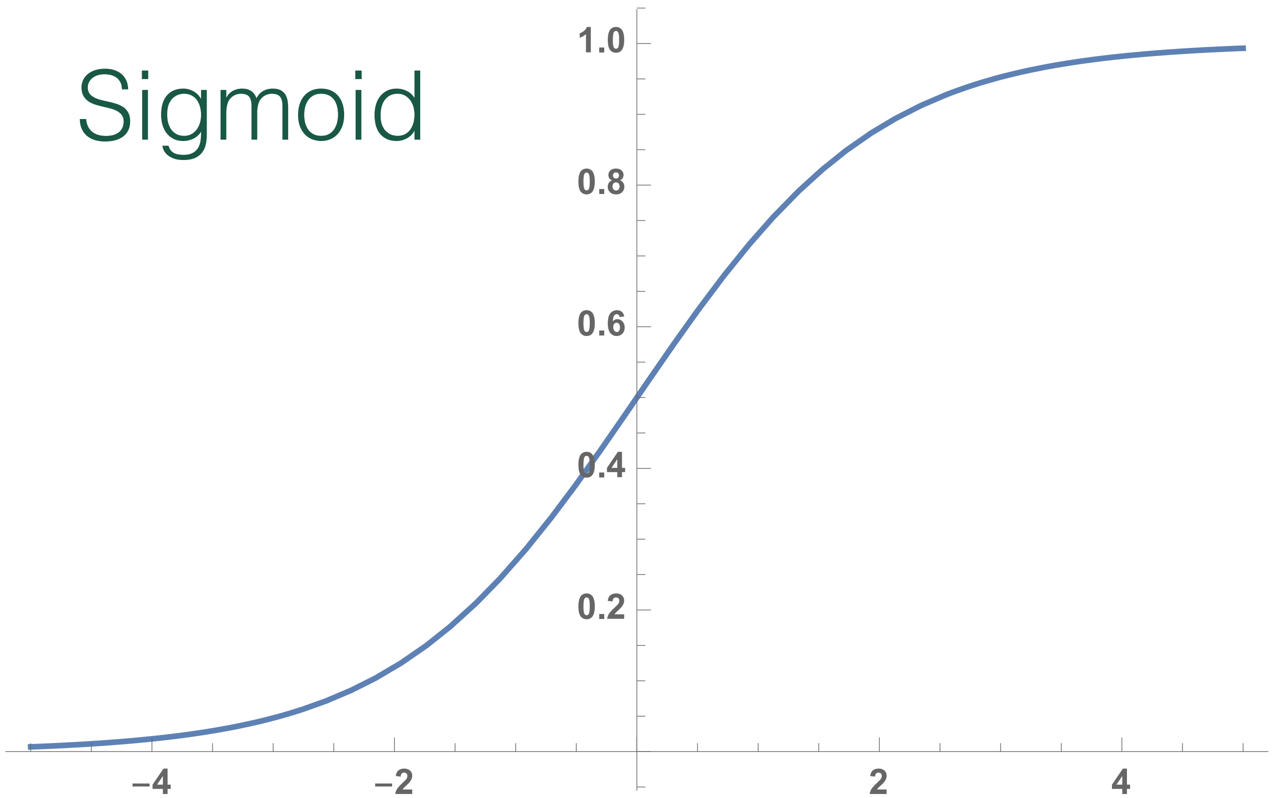
Chain rule for gradient of network involves multiple factors of the derivative multiplied together

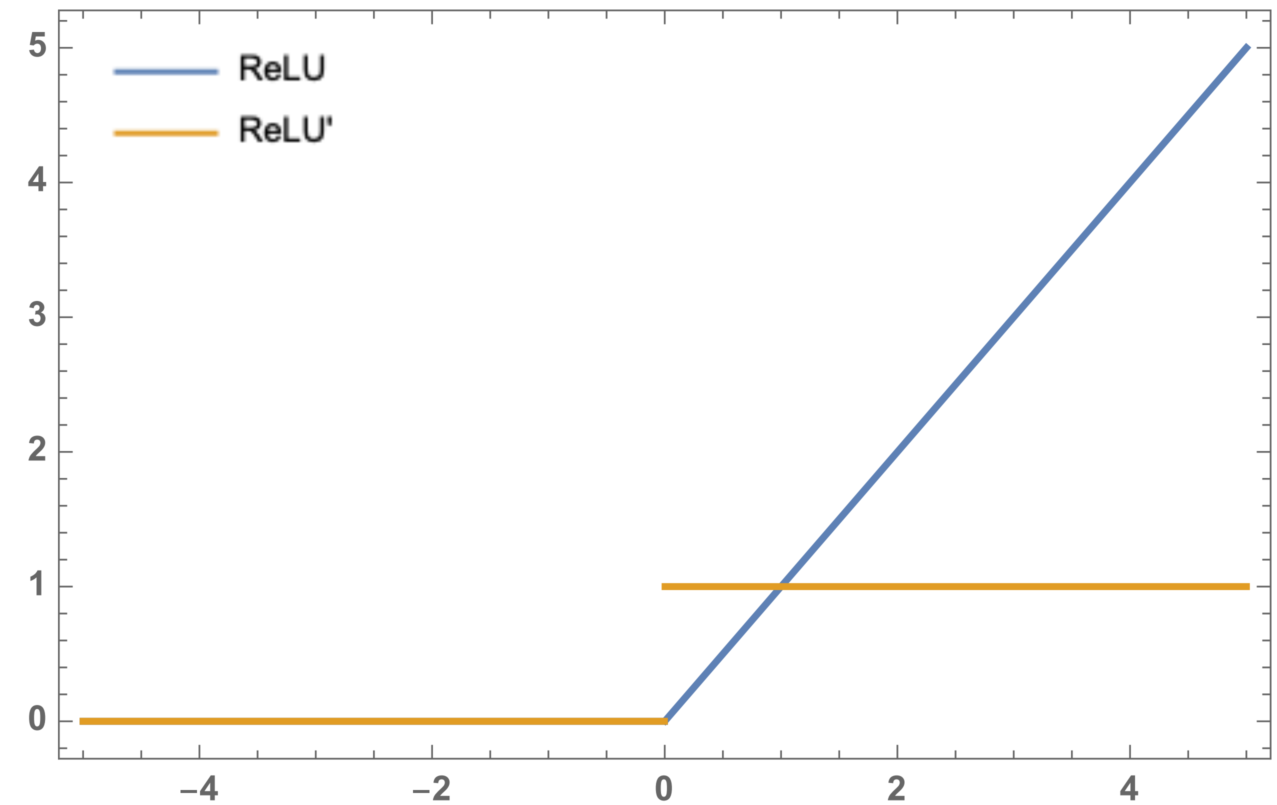
$$\boxed{(0.25)^4 = 0.0039}$$

Deep networks with Sigmoid activations have exponentially hard time training early layers

Disappearing Gradient

Sigmoid





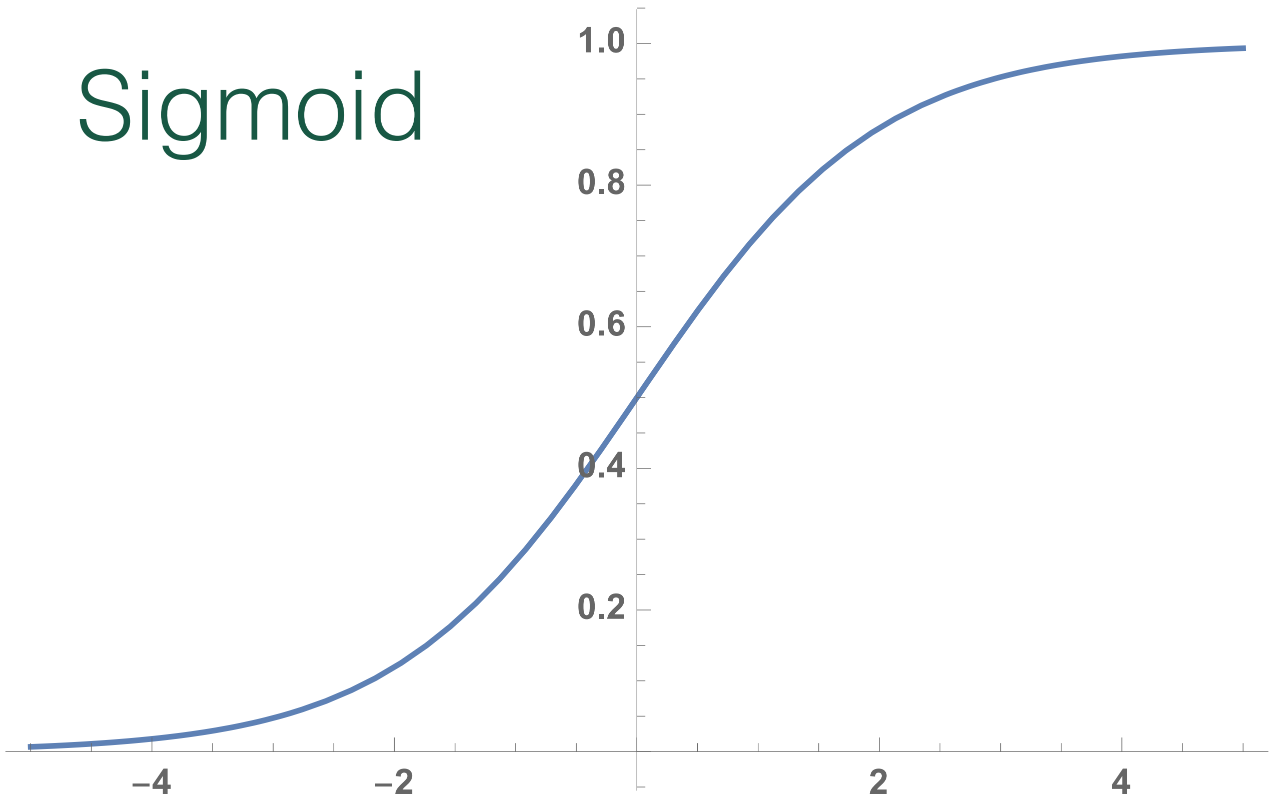
Using the Rectified Linear Unit (ReLU) solves this problem.
ReLU(x) = {0 if x <=0, x if x >0}

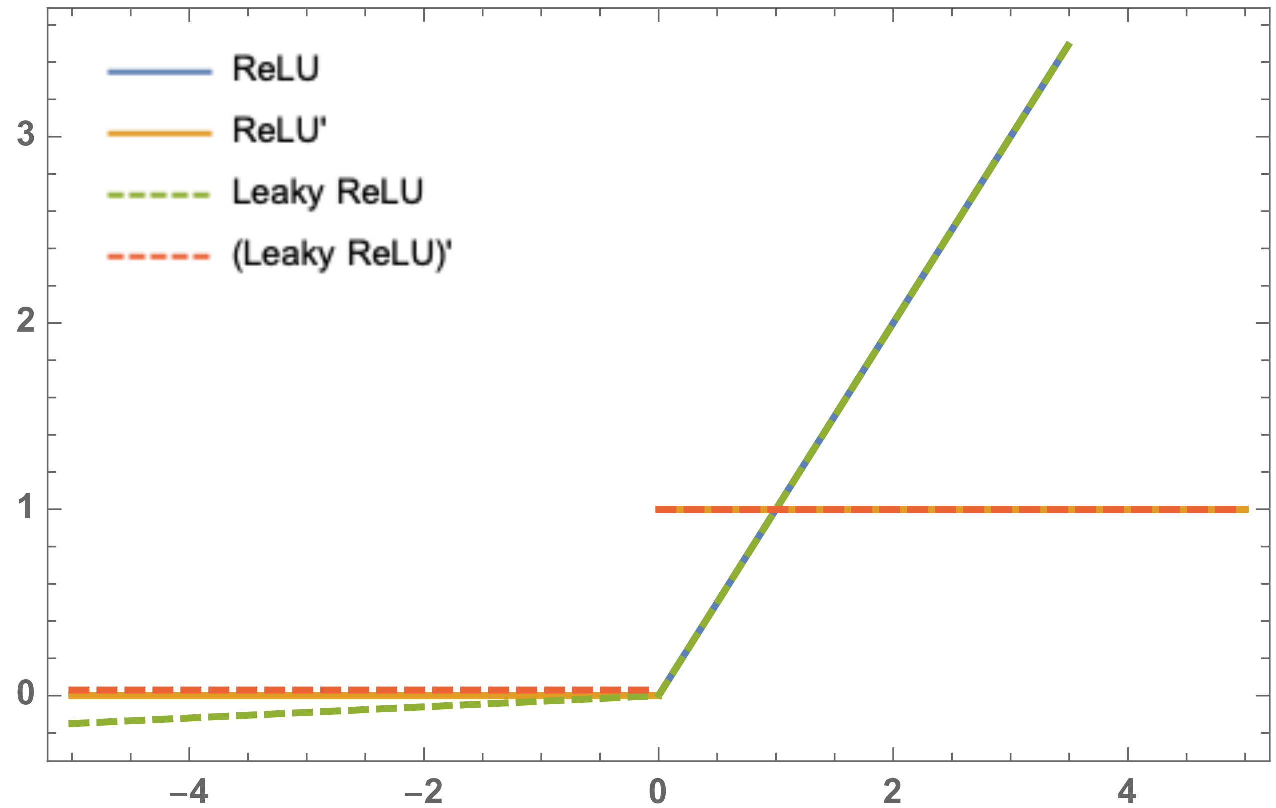
Still has nonlinearity which allows network to learn complicated patterns

Nodes can die (derivative always 0 so cannot update)

Disappearing Gradient

Sigmoid





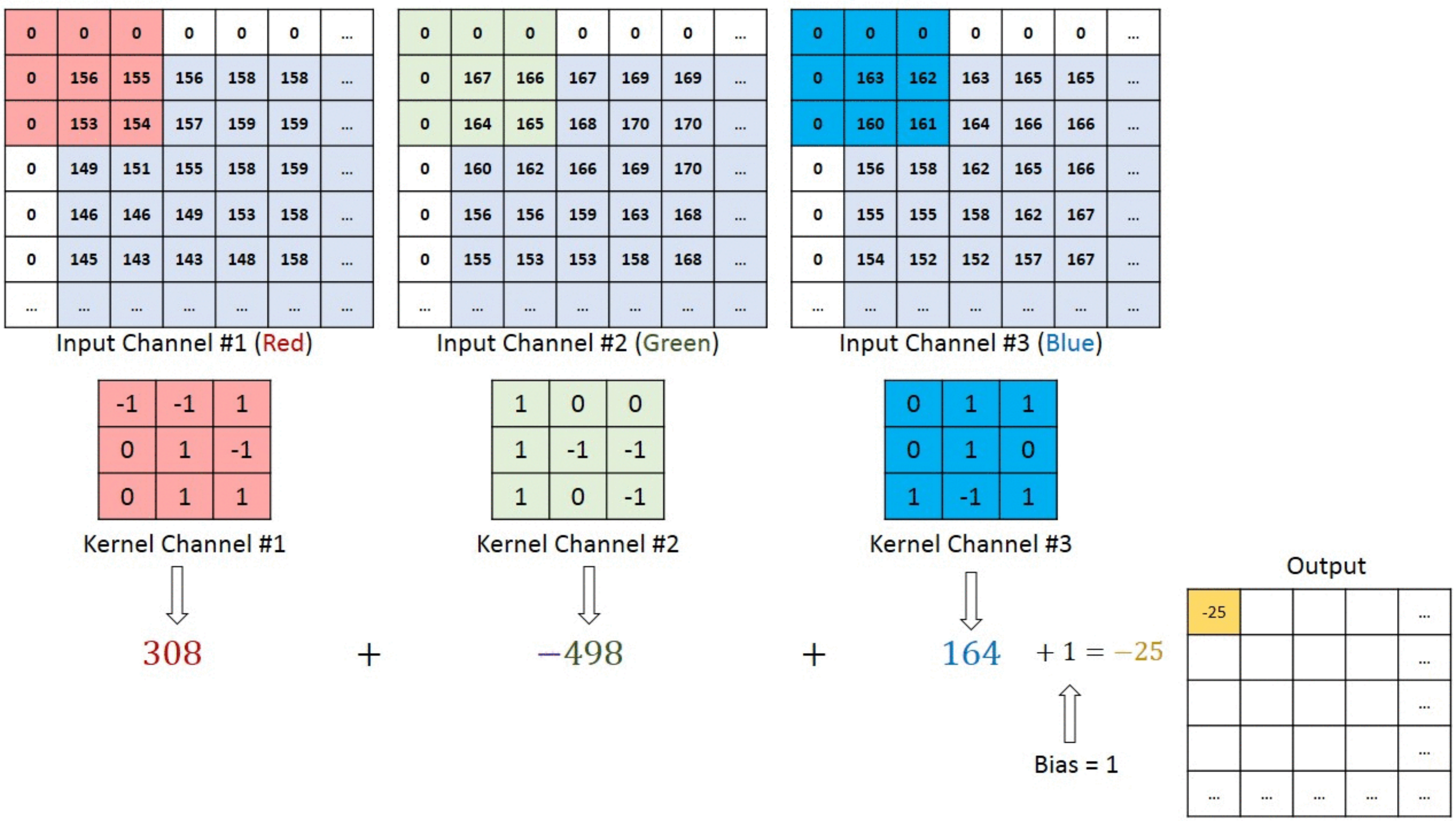
Leaky Rectified Linear Unit (LeakyReLU) solves this problem.

LeakyReLU(x) = {alpha*x if x <=0, x if x >0}

I have never had to use this in practice

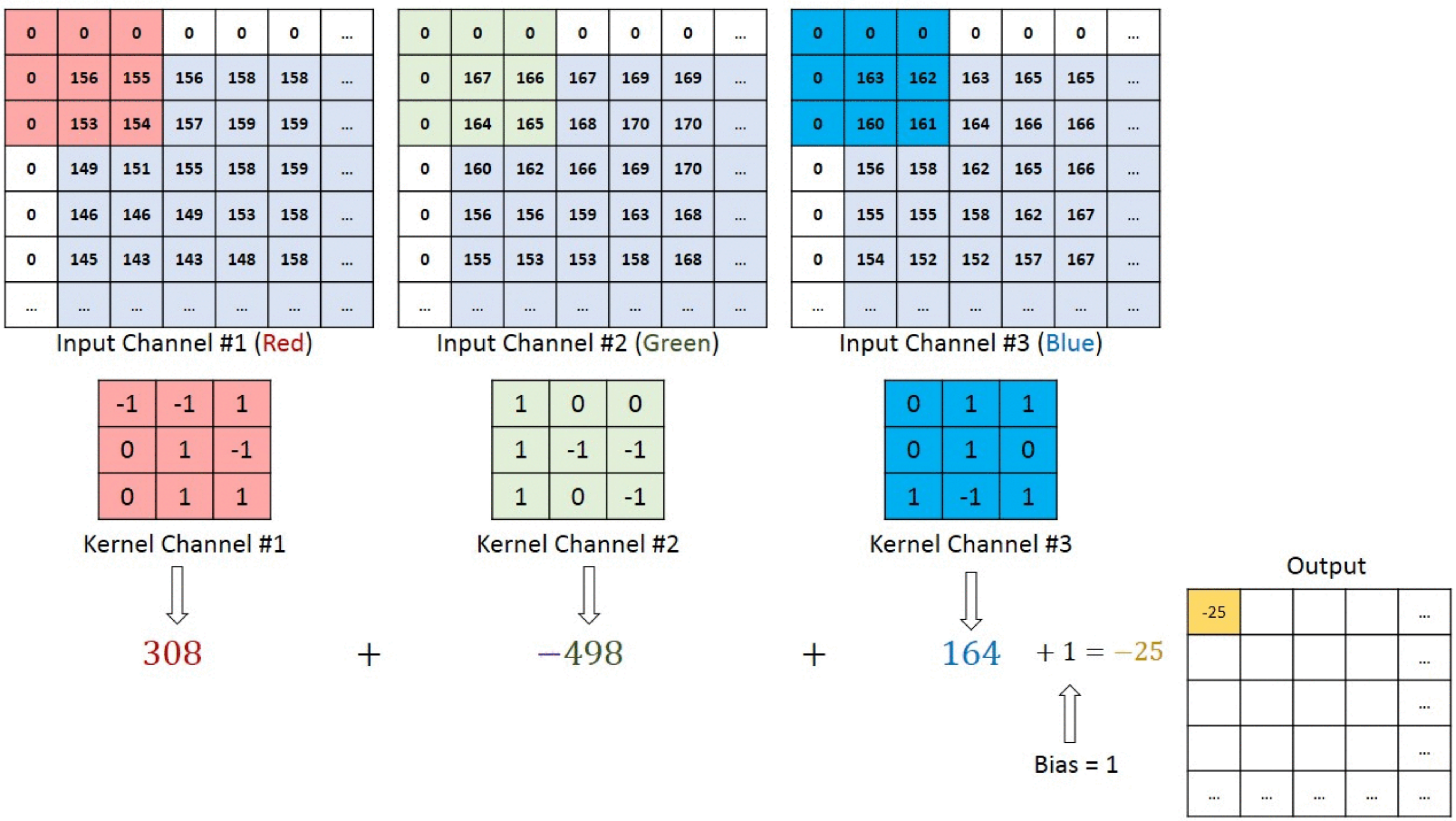
Convolutional neural network (CNN)

Dealing with multiple channels is straightforward — just enlarge filter to include channel dimension (3d filter) and perform element-wise multiplication along channel dimension as well.



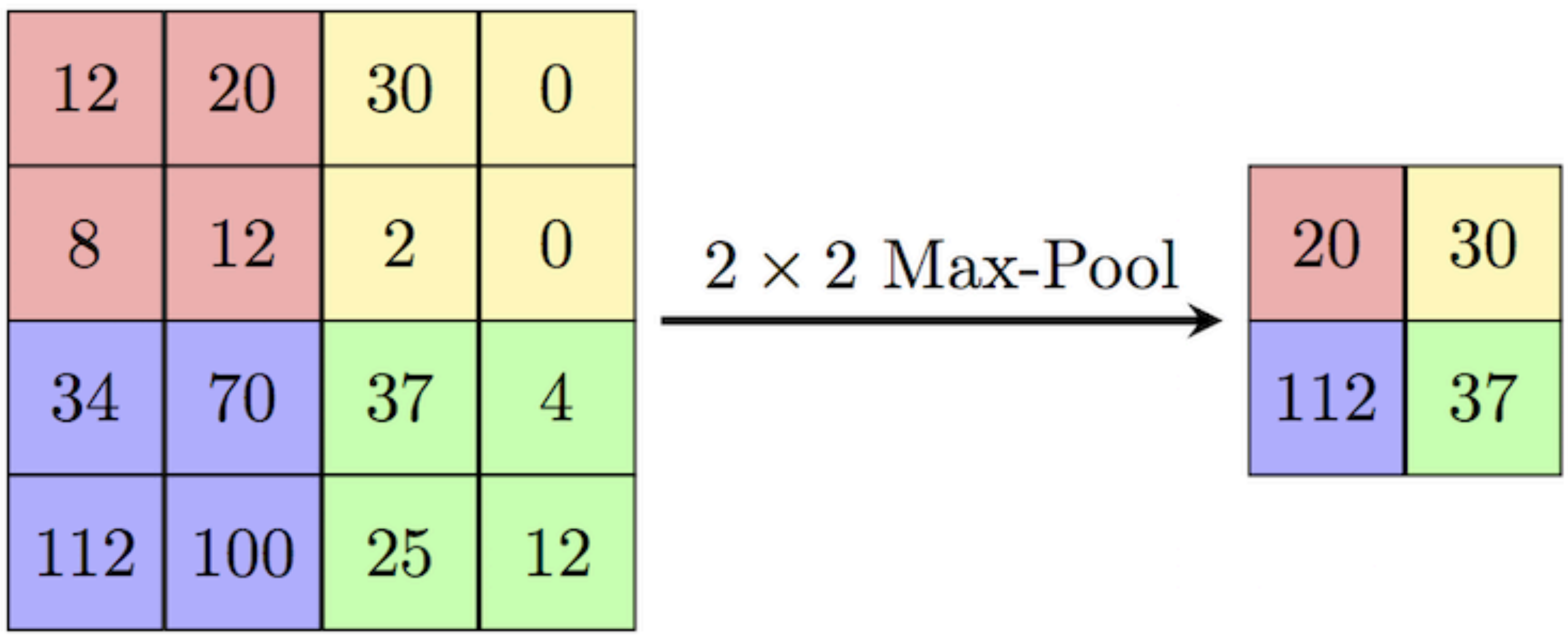
Convolutional neural network (CNN)

Dealing with multiple channels is straightforward — just enlarge filter to include channel dimension (3d filter) and perform element-wise multiplication along channel dimension as well.



Convolutional neural network (CNN)

"Max pooling"



Reduces image size, reducing parameters and mitigating overfitting

Allows NN to find spatially larger, often higher-level features

Recurrent Neural Networks (RNNs)

Popular architecture for natural language processing (sentence completion, autocorrect, translation, speech recognition…)

Starting point: sequence of numbers $x_1, x_2, x_3, \ldots$

Suppose we want to predict the next number in the sequence?

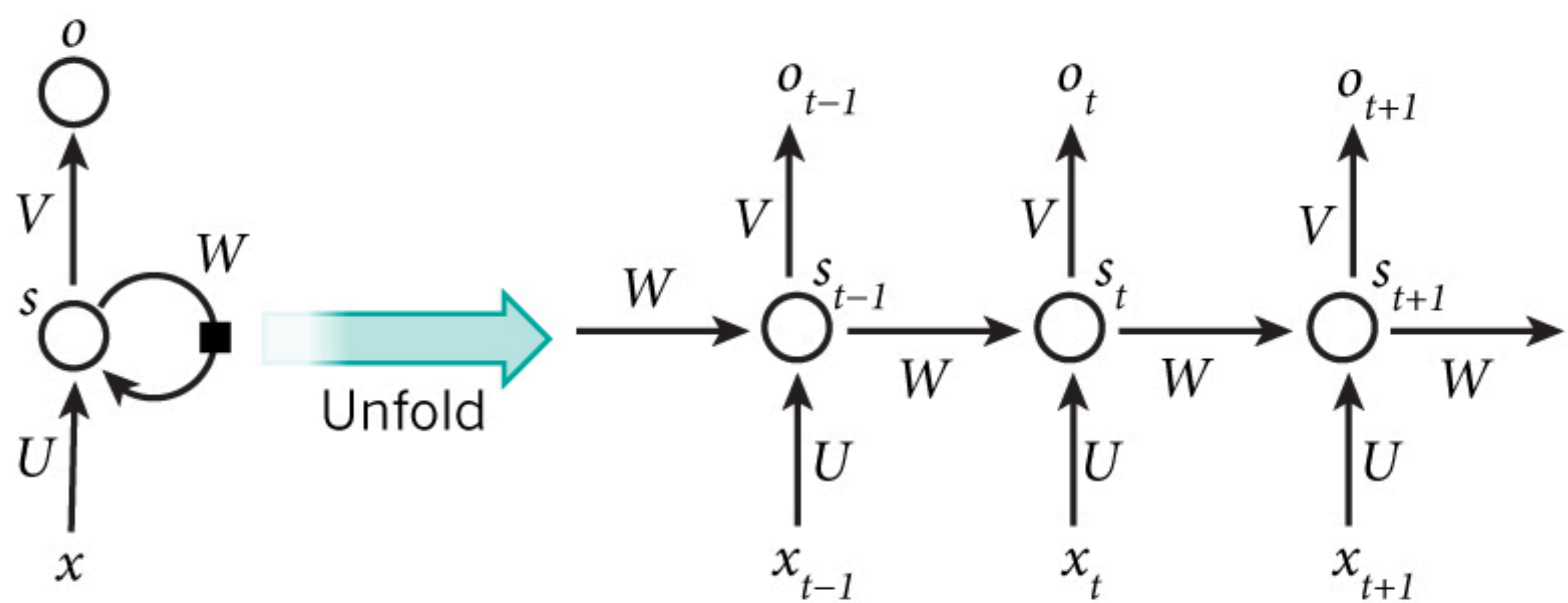
Idea of RNN:

- feed data sequentially to NN
- after each time step update *hidden state.* Hidden state encodes "memory" of sequence.
- Use hidden state to make predictions.

Recurrent Neural Networks (RNNs)

Basic RNN architecture

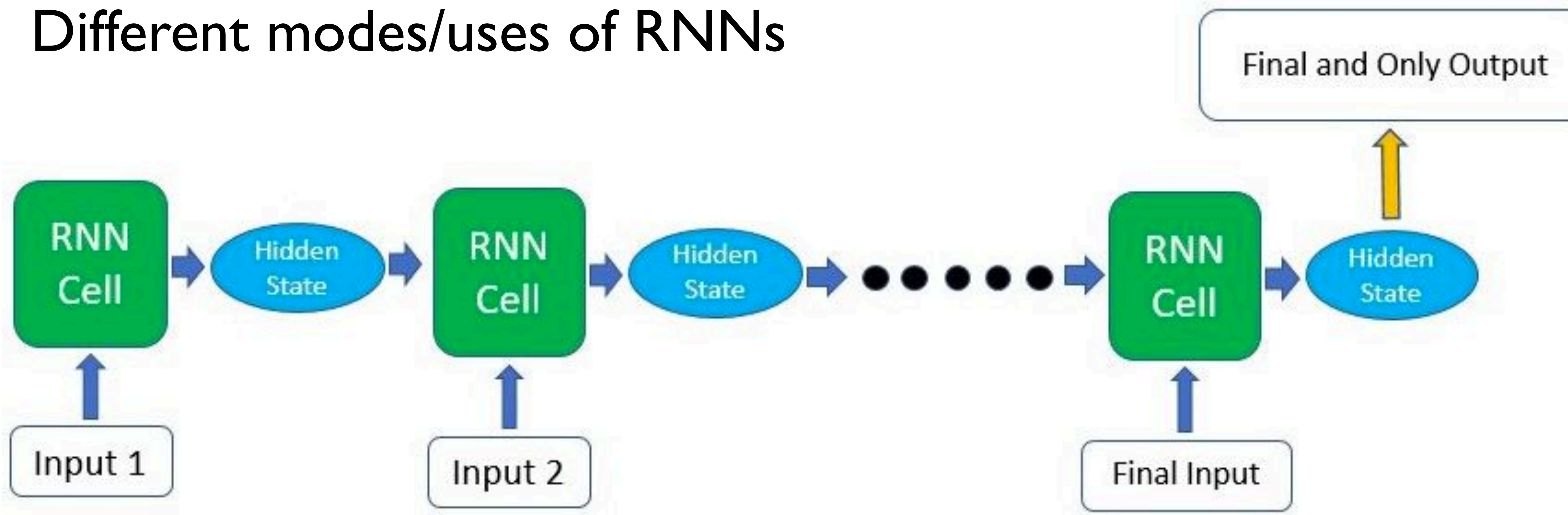
predictions, e.g. $o_t = \mathrm{softmax}(V s_t)$



$s_t = f(U x_t + W s_{t-1})$ hidden state after time step t

Recurrent Neural Networks (RNNs)

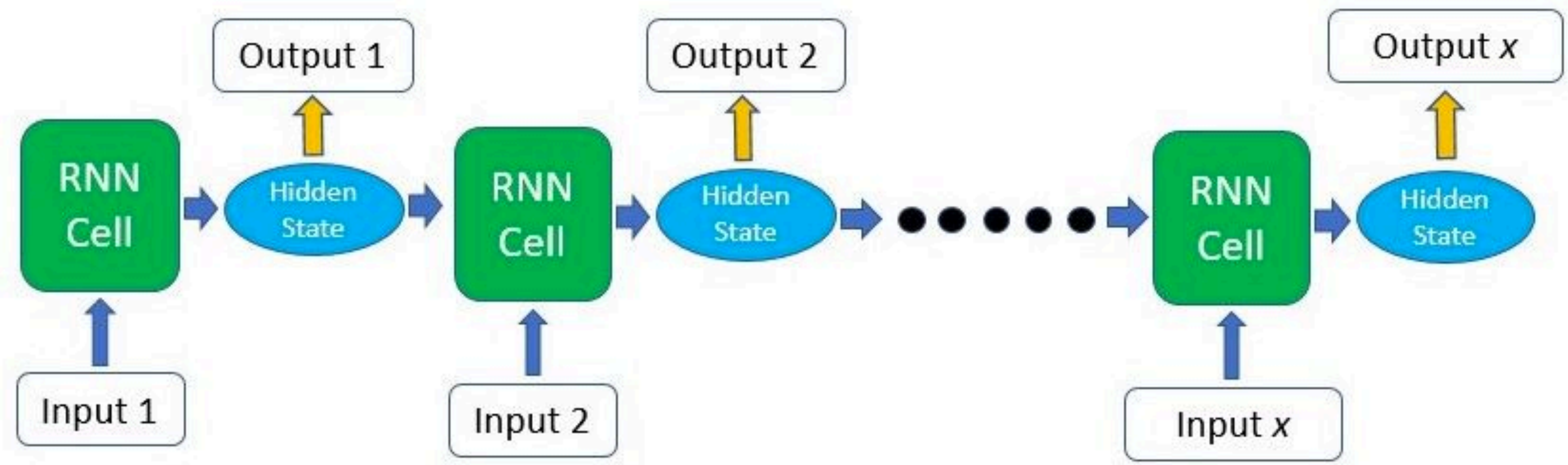
Different modes/uses of RNNs



sequence classification, regression

Recurrent Neural Networks (RNNs)

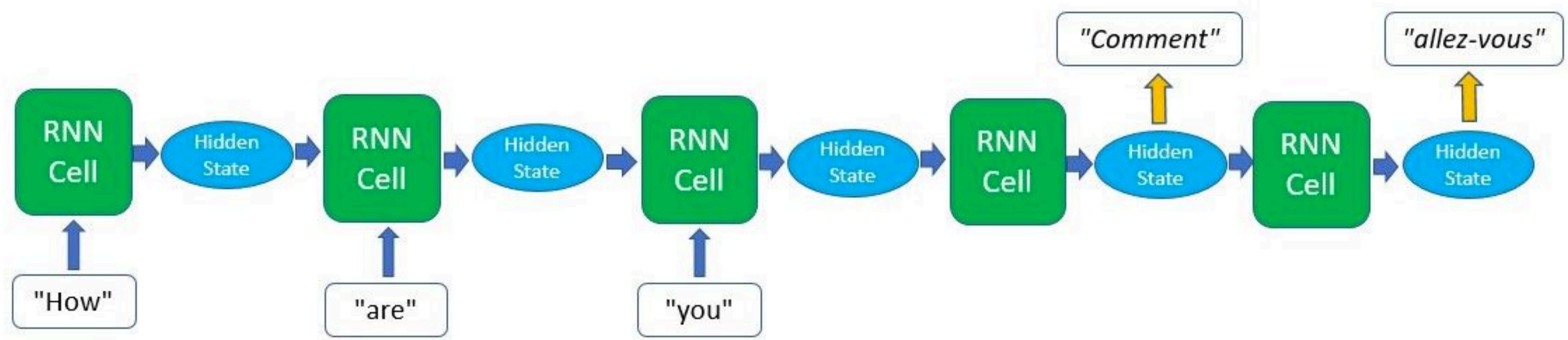
Different modes/uses of RNNs



real-time prediction

Recurrent Neural Networks (RNNs)

Different modes/uses of RNNs

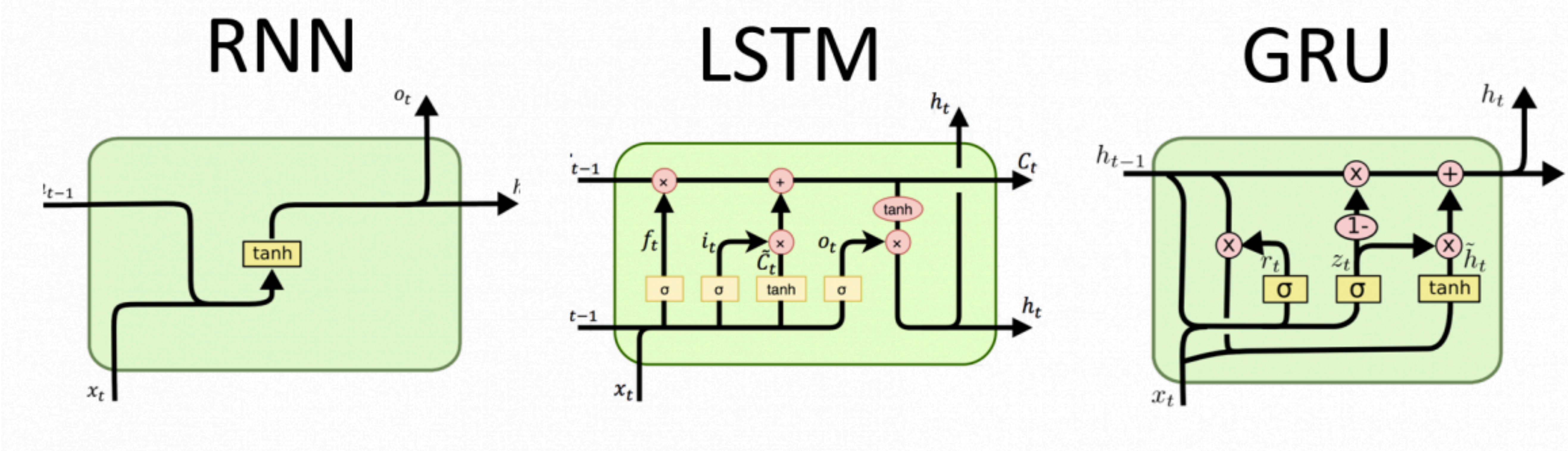


sequence-to-sequence

Recurrent Neural Networks (RNNs)

Simple RNNs applied to long sequences have a very serious exploding/vanishing gradient problem.

Prevents them from "remembering" relevant information from earlier in the sequence.



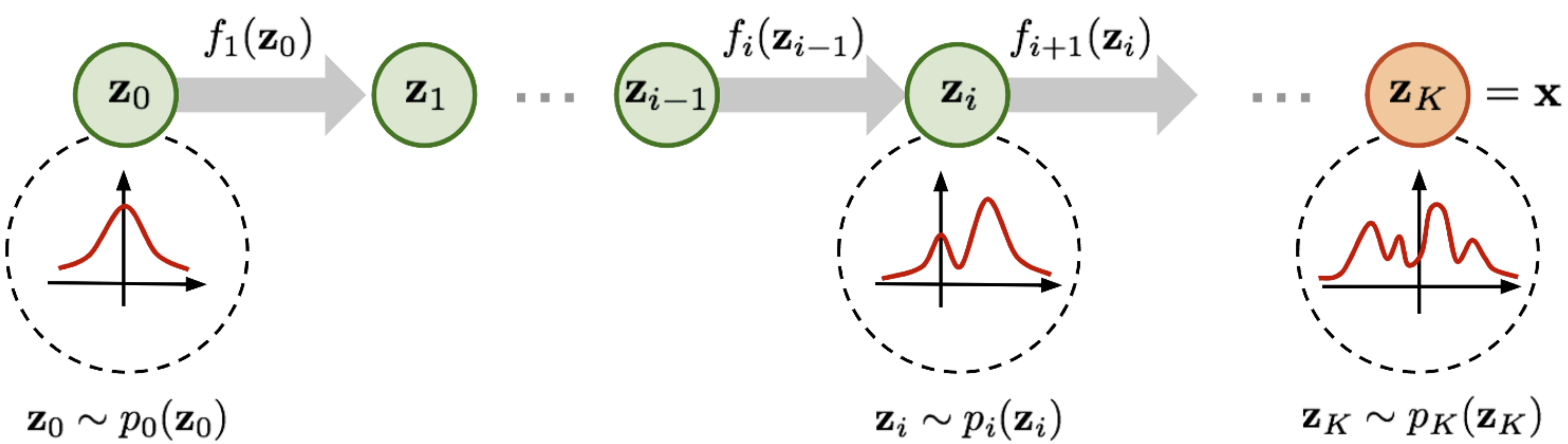
"Long-short term memory" and "Gated recurrent units" are two methods commonly used to solve the gradient problem and improve performance.

Normalizing flows

Rezende & Mohamed 1505.05770

Recently a lot of excitement and progress in the problem of density estimation with neural networks.

Idea: map original distribution to normal distribution through series of invertible transformations.



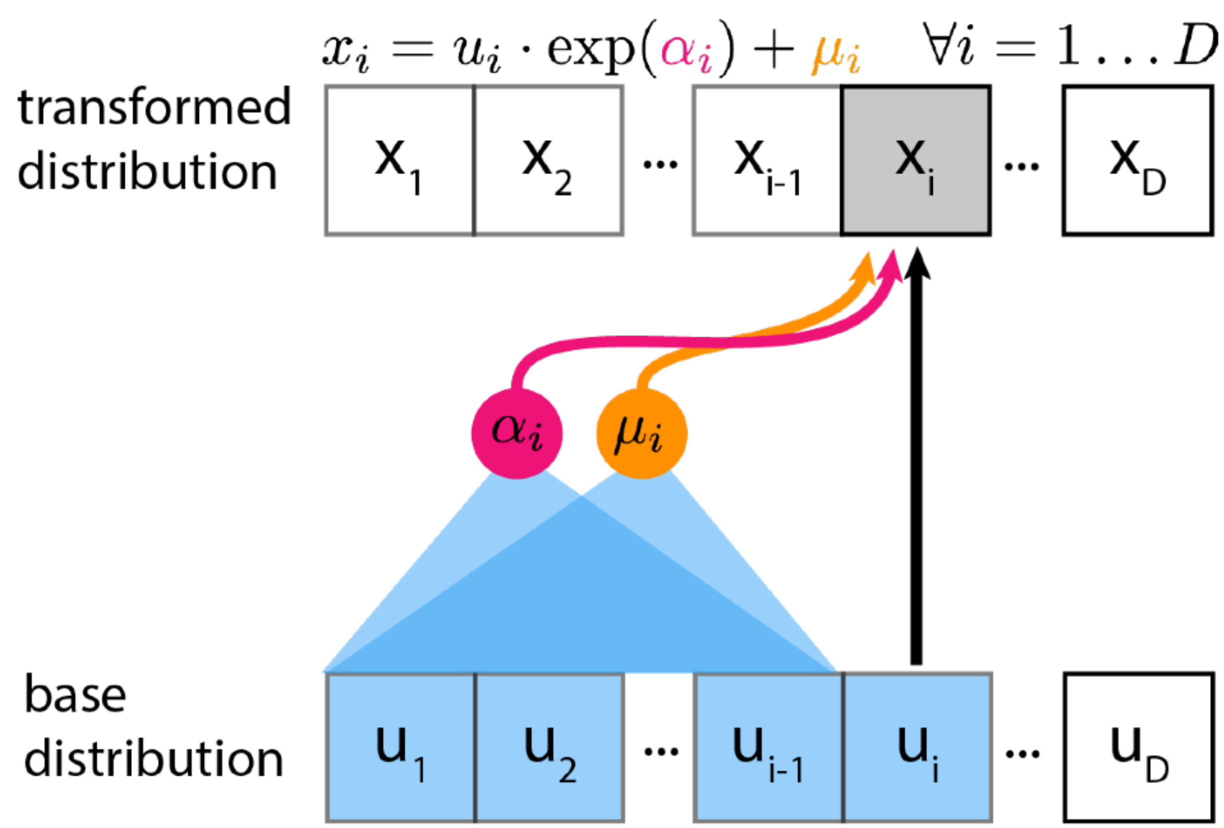
Examples: RealNVP, NICE, Glow, …

Autoregressive flows

$$p(x) = \prod_i p(x_i \,|\, x_{1:i-1})$$

Special type of normalizing flows. Learn probability density of each coordinate conditioned on previous coordinates.

Transformation upper triangular — automatically invertible. Allows for more expressive transformations.



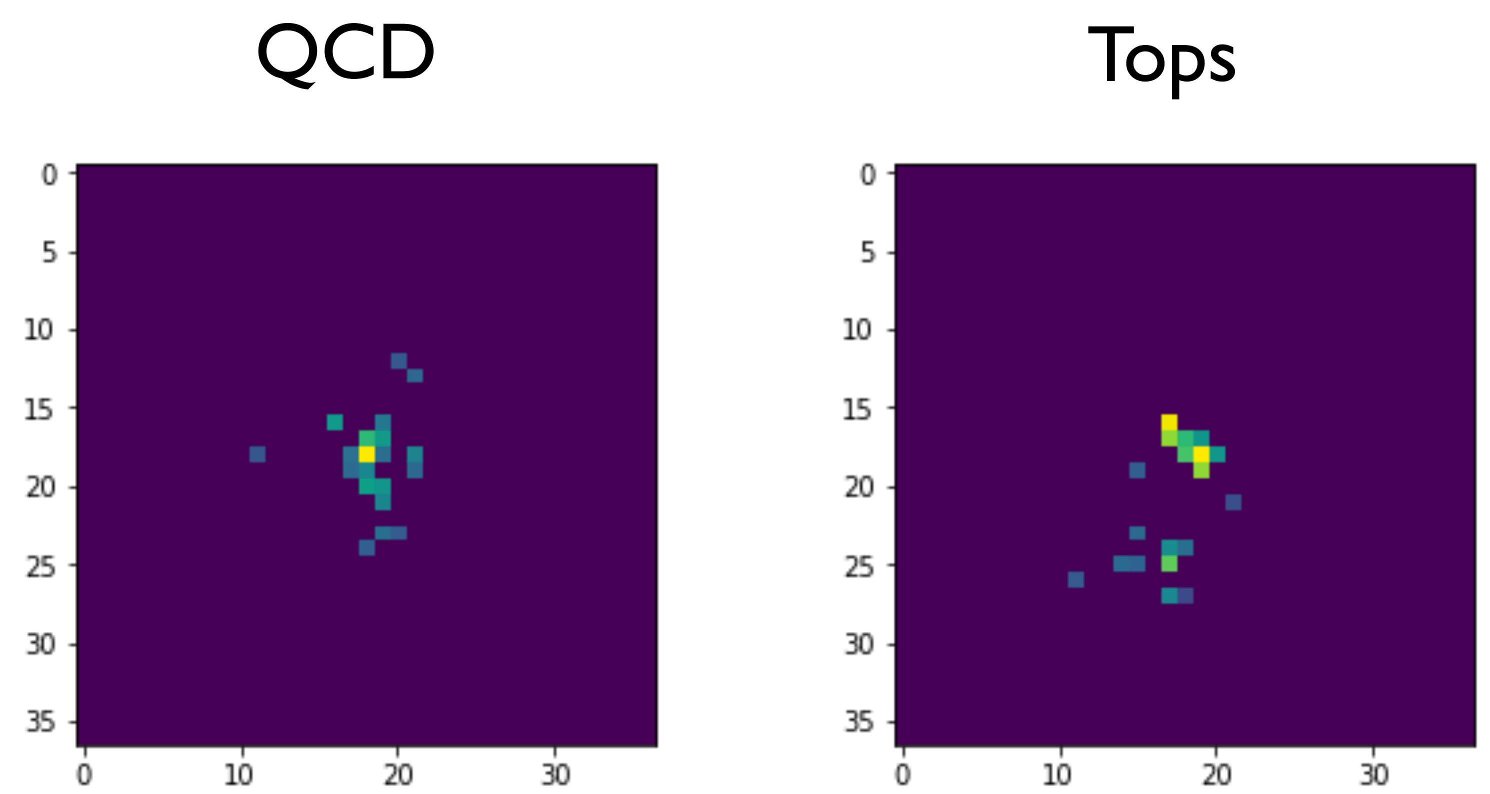
Examples: MADE, MAF, IAF, NAF, PixelRNN, Wavenet, …

Top Tagging with CNNs

Macaluso & DS 1803.00107

	CMS
Jet sample	13 TeV $p_T \in (800, 900)$ GeV, $\lvert\eta\rvert < 1$ PYTHIA 8 and DELPHES particle-flow match: $\Delta R(t, j) < 0.6$ merge: $\Delta R(t, q) < 0.6$ 1.2M + 1.2M
Image	37×37 $\Delta\eta = \Delta\phi = 3.2$
Colors	$(p_T^{neutral}, p_T^{track}, N_{track}, N_{muon})$

Individual images very sparse



Building on previous "DeepTop" tagger of Kasieczka et al 1701.08784

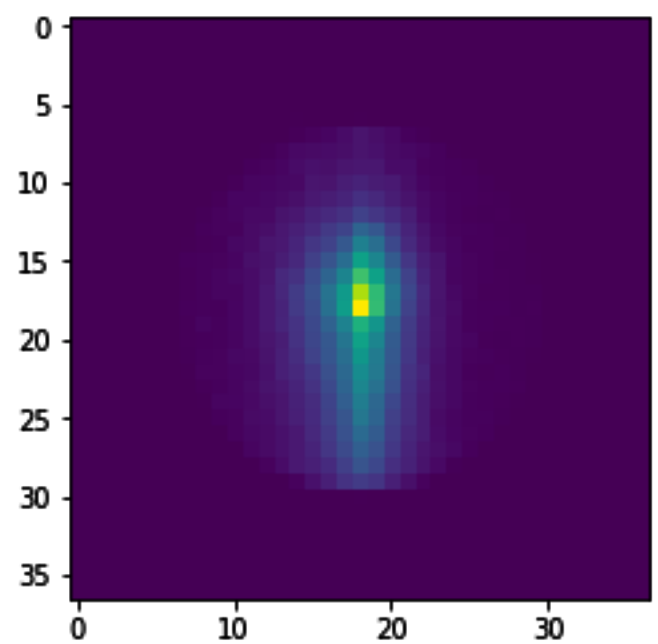
Top Tagging with CNNs

Macaluso & DS 1803.00107

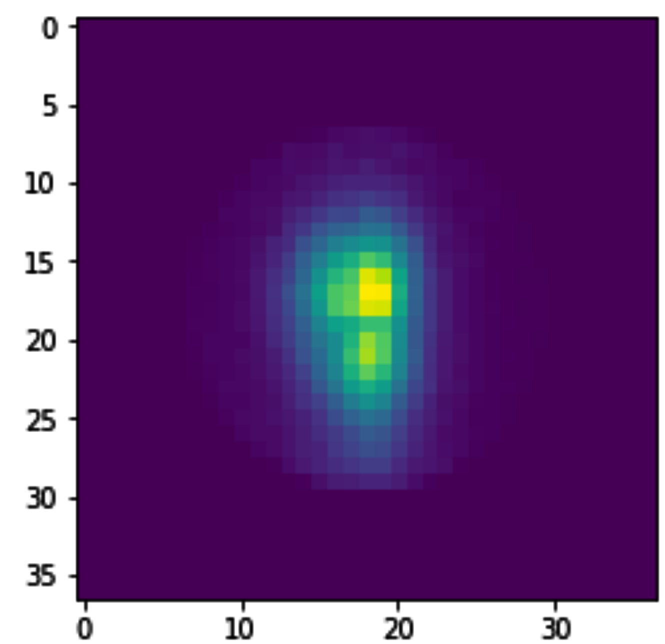
Average images clearly different

	CMS
Jet sample	13 TeV $p_T \in (800, 900)$ GeV, $\|\eta\| < 1$ PYTHIA 8 and DELPHES particle-flow match: $\Delta R(t, j) < 0.6$ merge: $\Delta R(t, q) < 0.6$ 1.2M + 1.2M
Image	37×37 $\Delta\eta = \Delta\phi = 3.2$
Colors	$(p_T^{neutral}, p_T^{track}, N_{track}, N_{muon})$

QCD



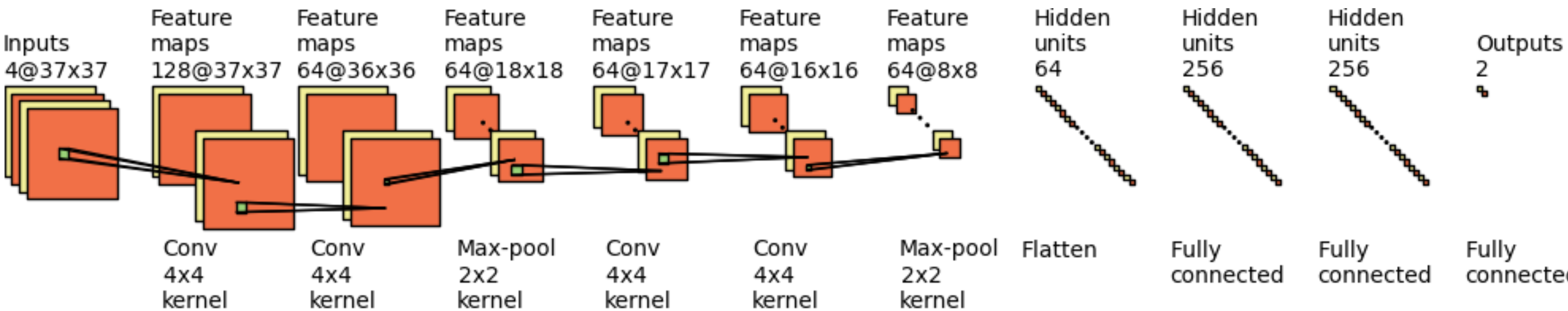
Tops



Building on previous "DeepTop" tagger of Kasieczka et al 1701.08784

Top Tagging with CNNs

Macaluso & DS 1803.00107



AdaDelta

$\eta = 0.3$ with annealing schedule

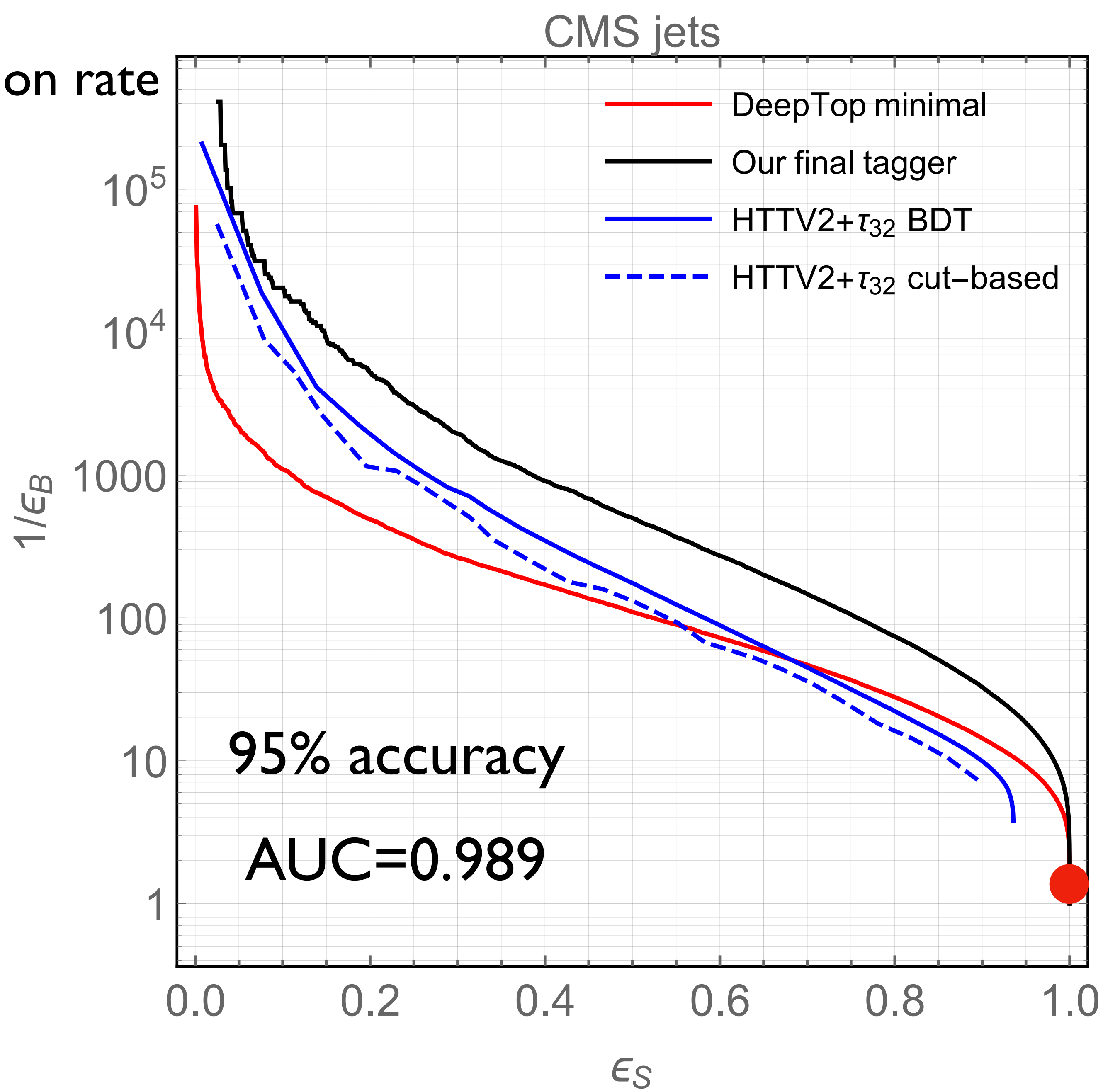
minibatch size=128

cross entropy loss

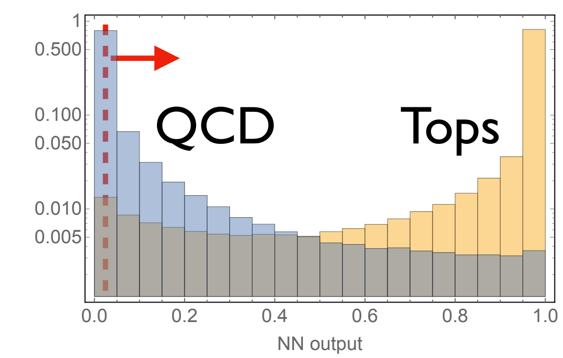
Top Tagging with CNNs

Macaluso & DS 1803.00107

QCD rejection rate



Top tagging efficiency

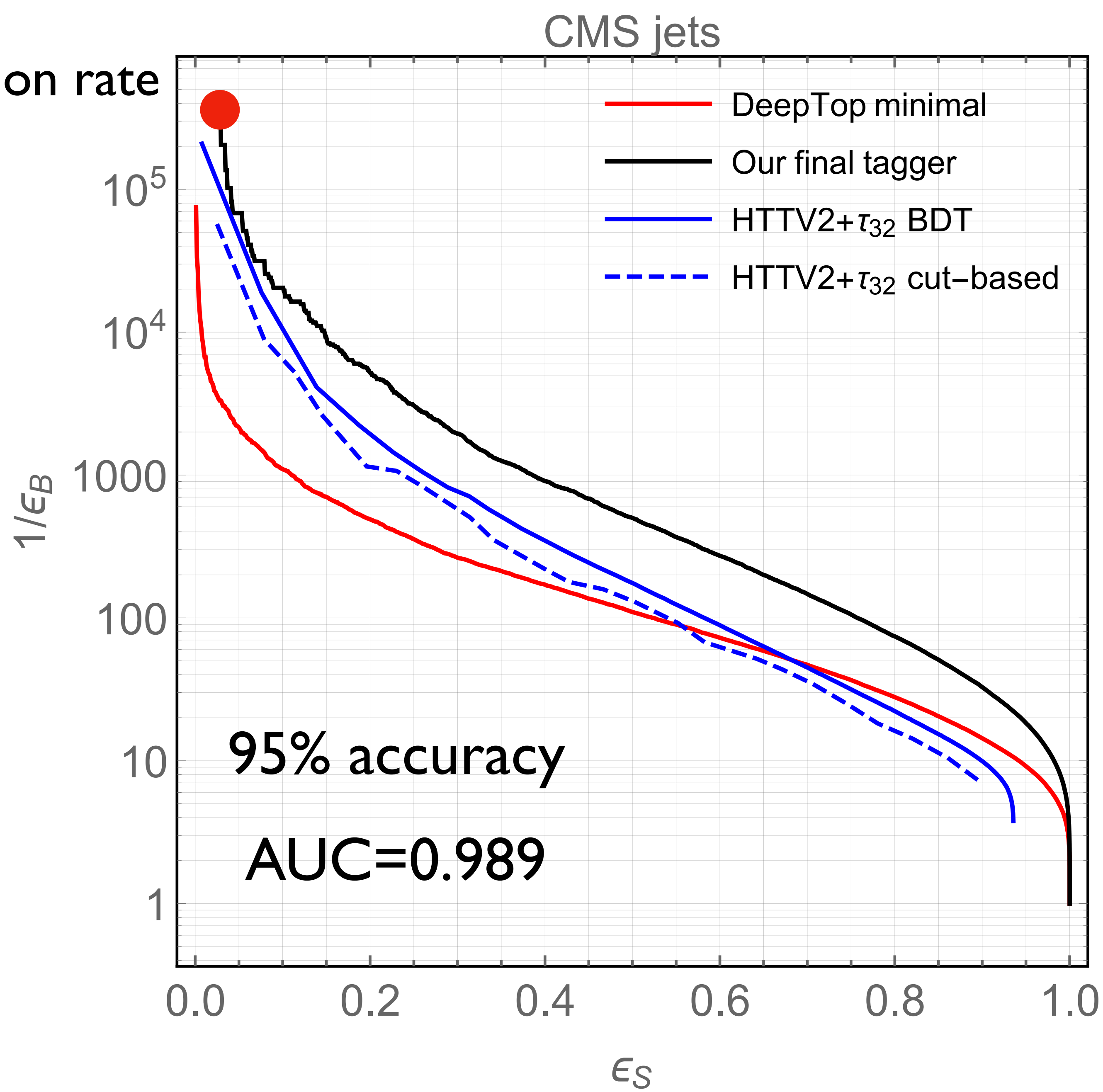


Can achieve factor of ~3 improvement over cut-based approaches and BDTs!

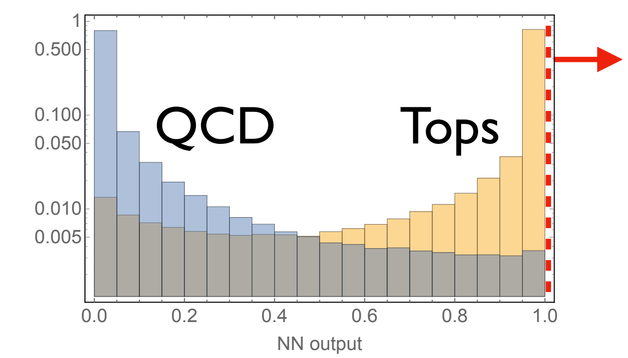
Top Tagging with CNNs

Macaluso & DS 1803.00107

QCD rejection rate



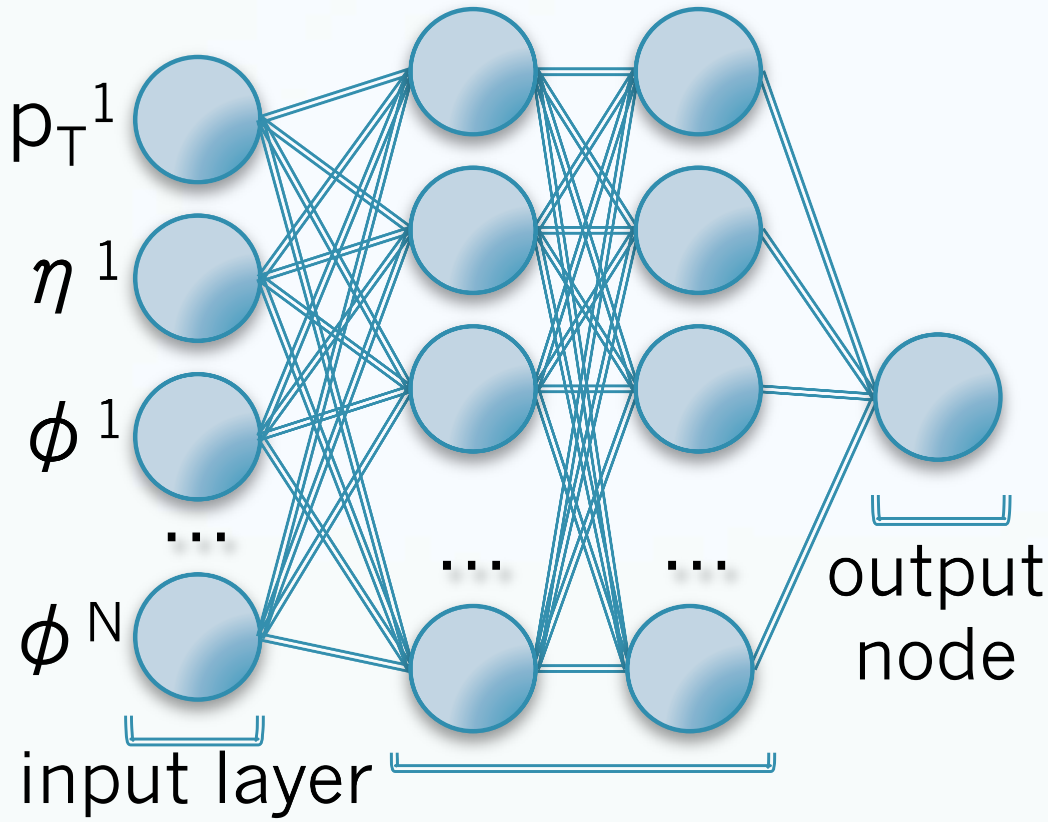
Top tagging efficiency



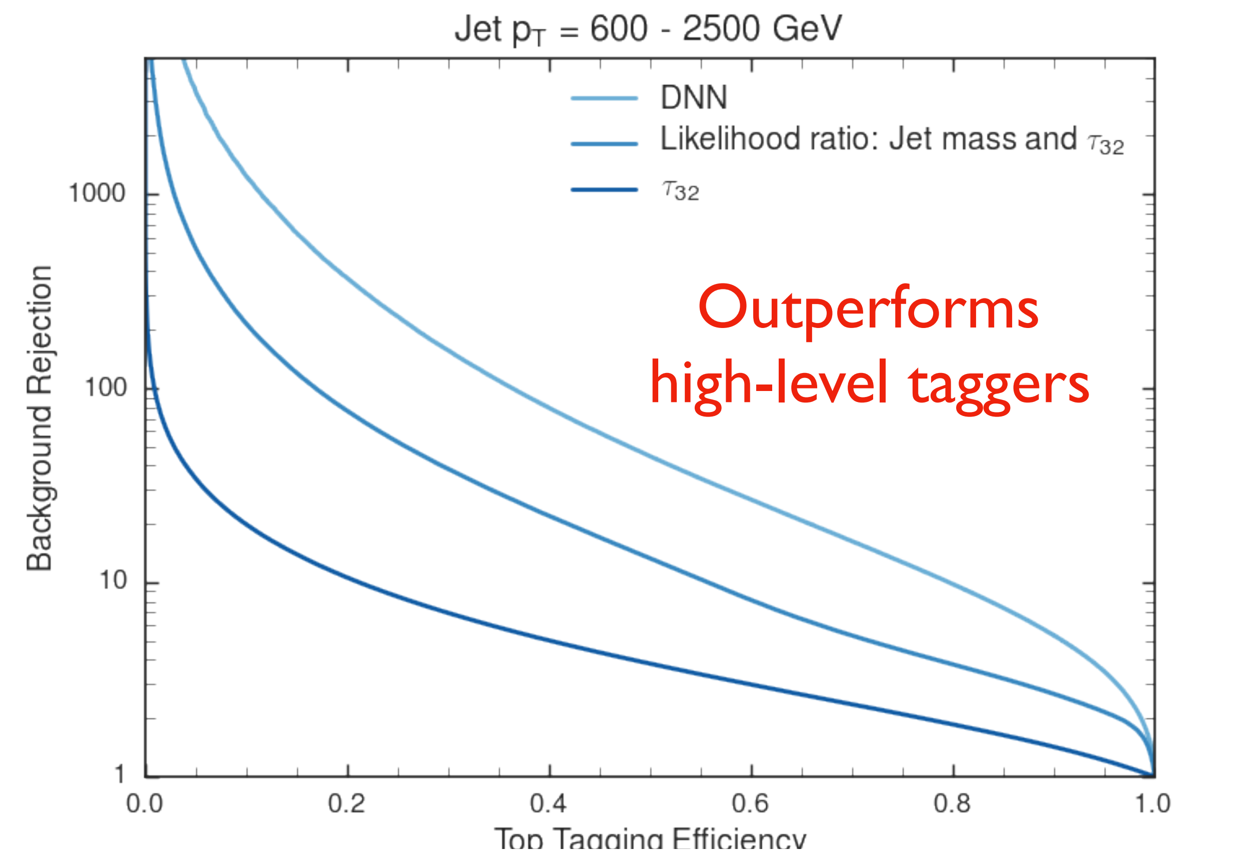
Can achieve factor of ~3 improvement over cut-based approaches and BDTs!

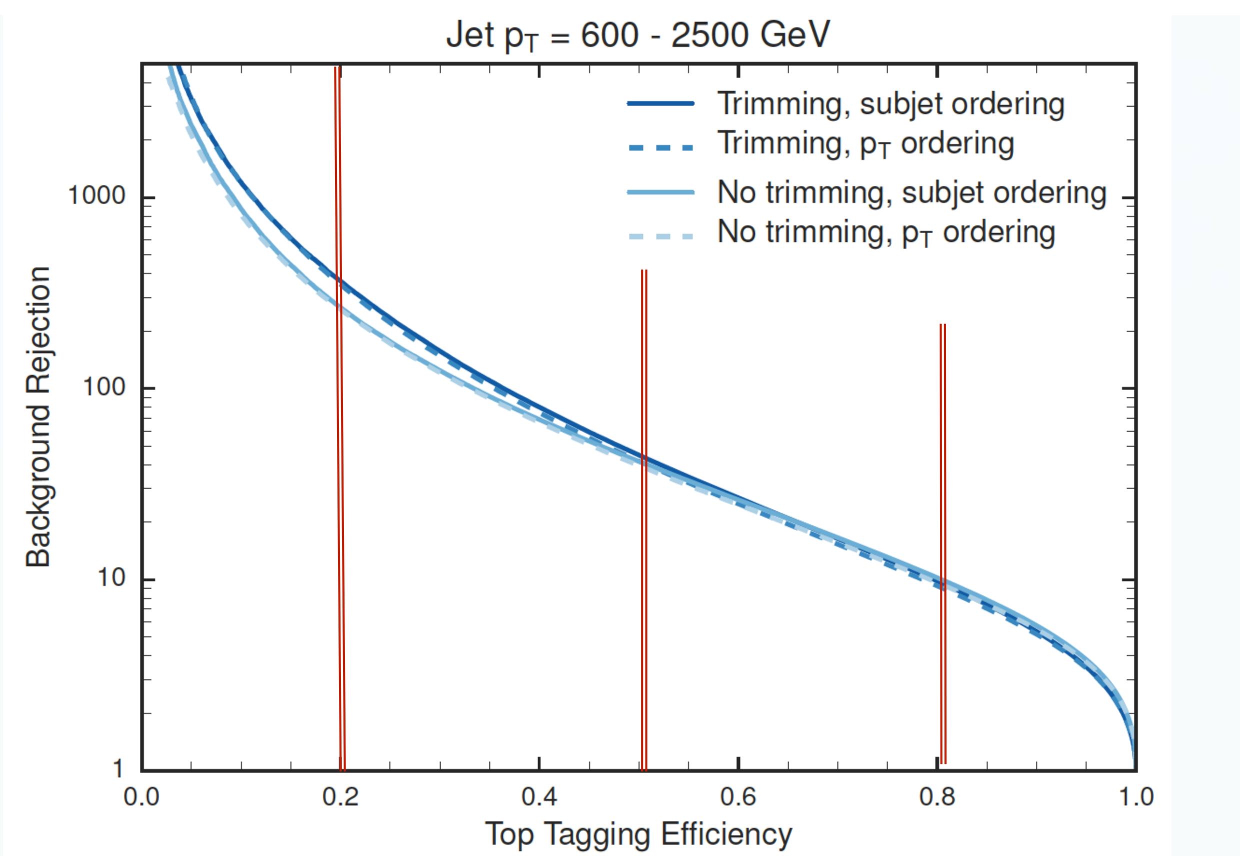
Lists of four vectors

Pearkes et al 1704.02124



5 hidden layers
300,150,50,10,5
nodes / hidden layer

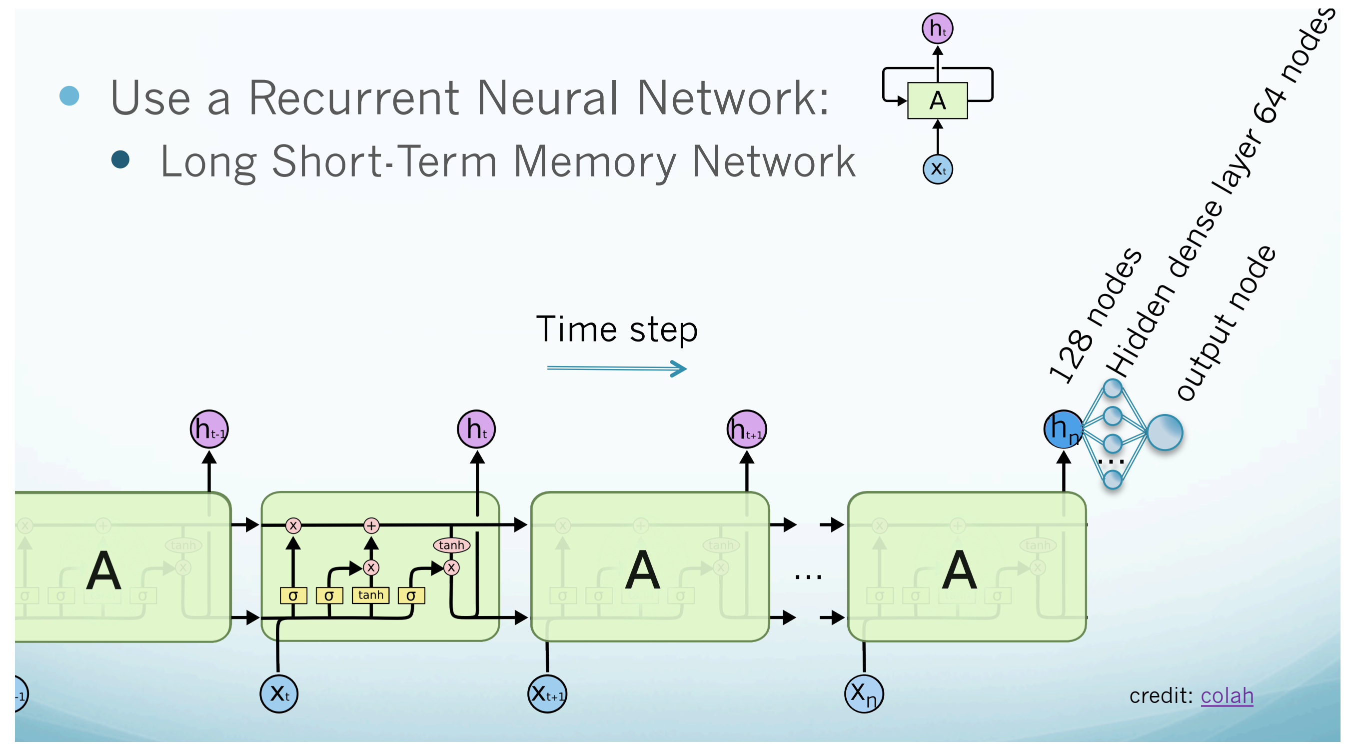


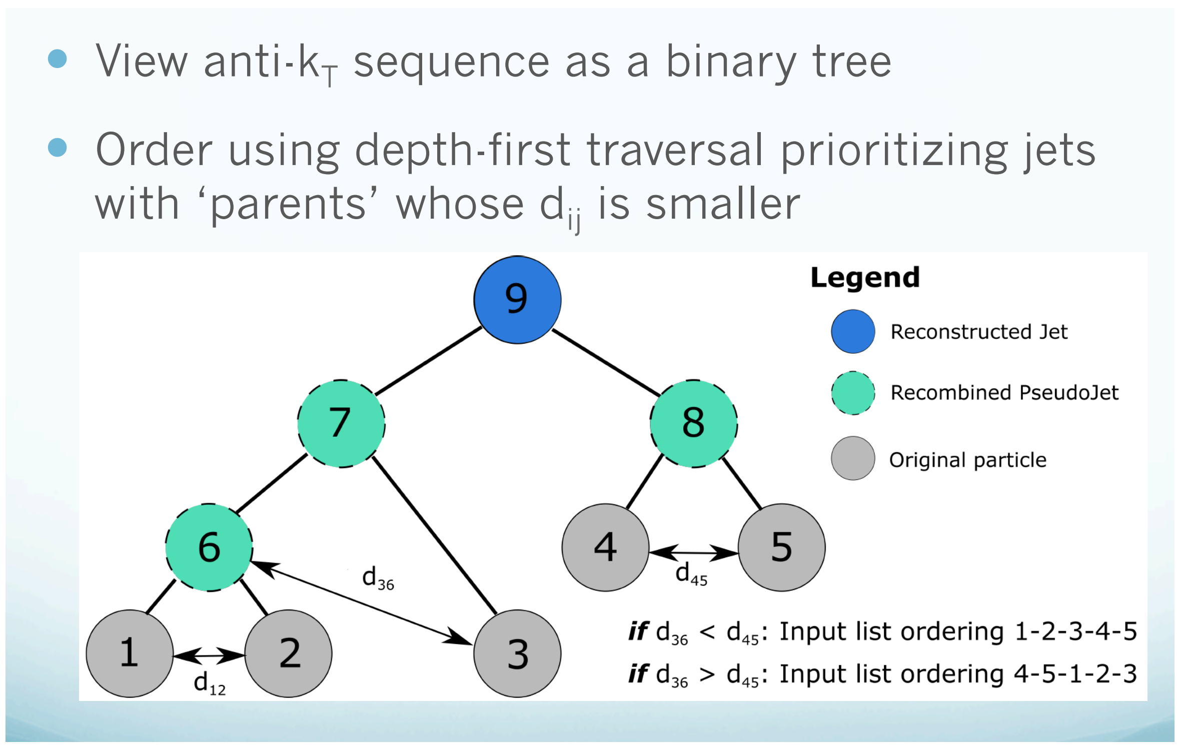


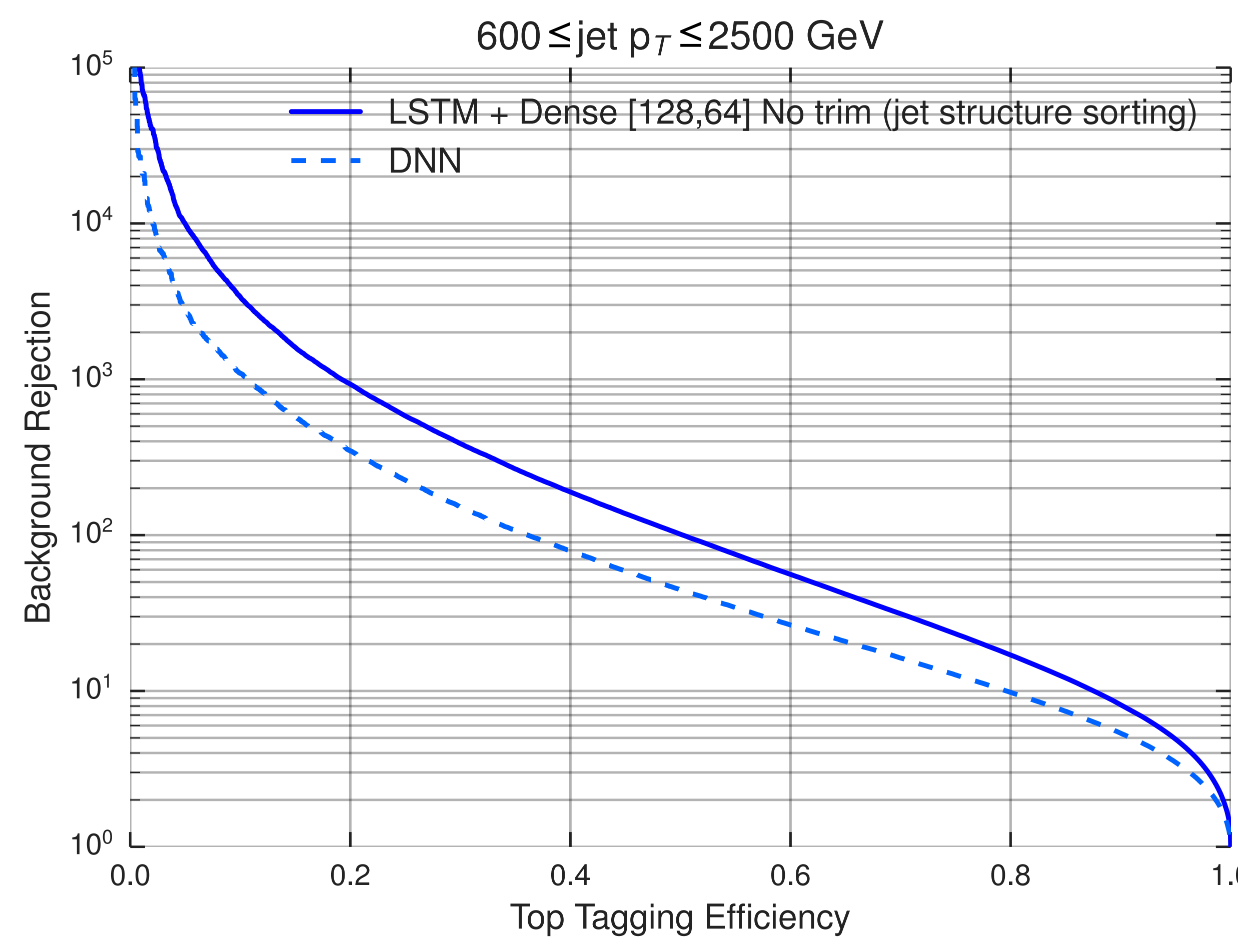
robust against different orderings

Sequences

Egan et al 1711.09059





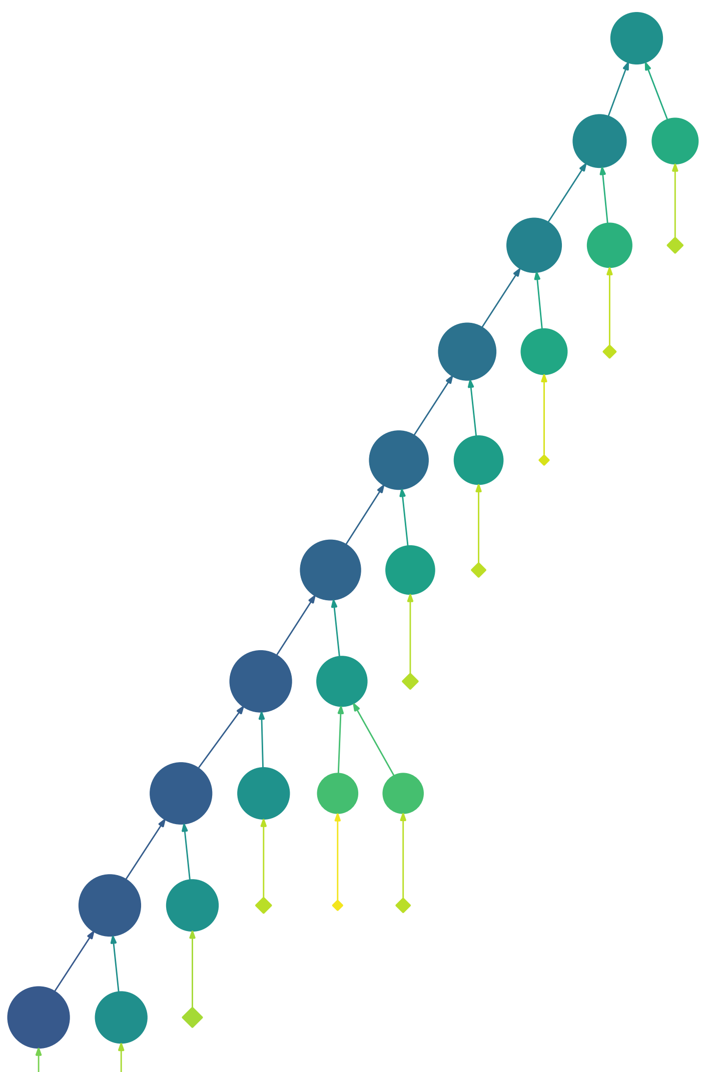


Turn jet into a sequence, e.g. using jet clustering history.

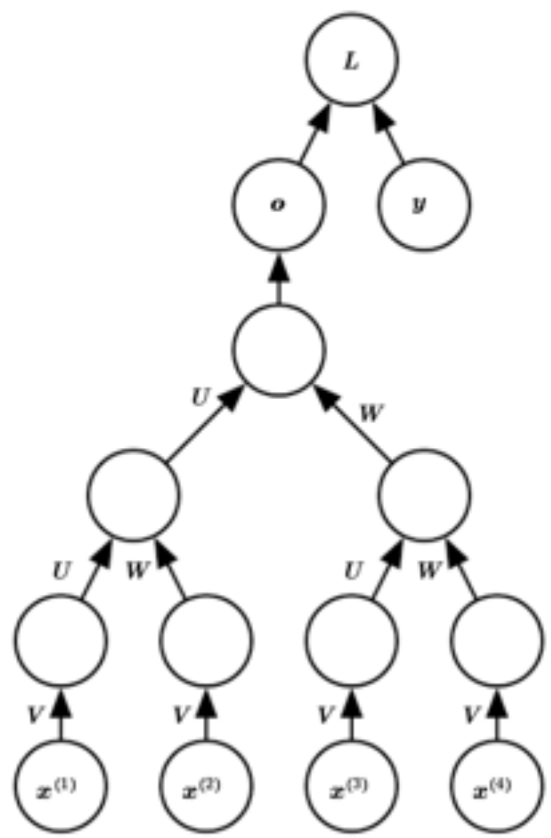
Feed sequence to RNN/LSTM, etc.

Trees

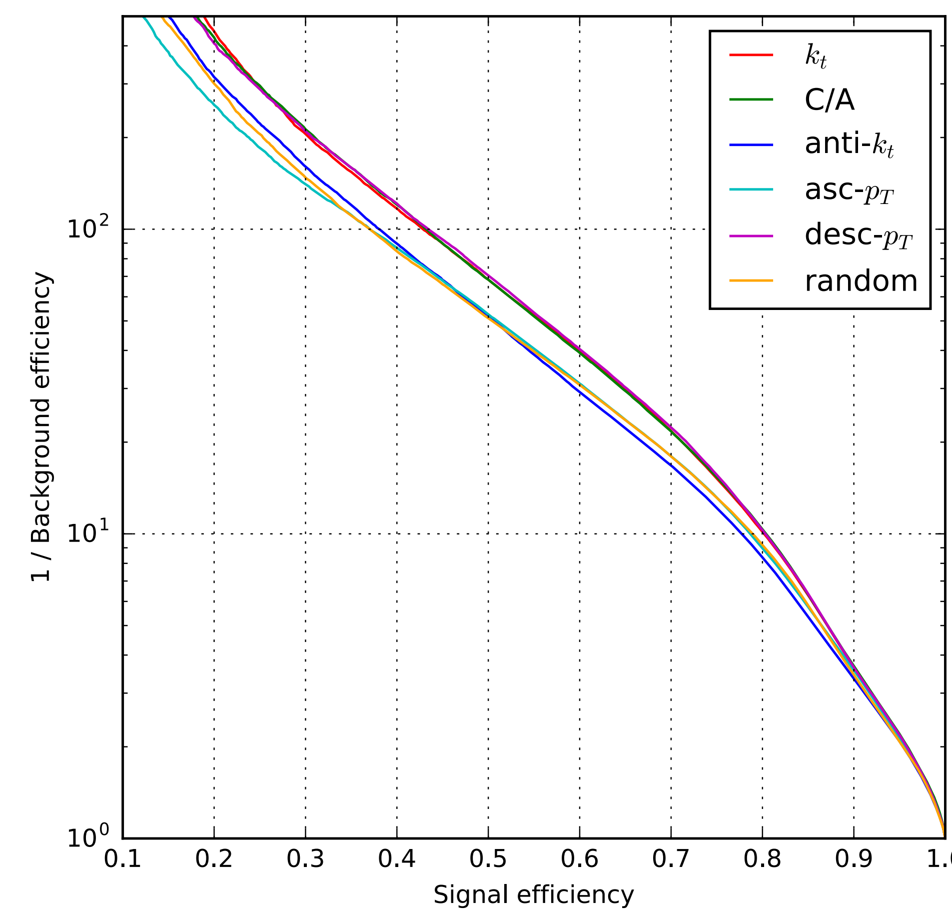
Louppe et al 1702.00748, Cheng 1711.02633



jet tree structure



RecNN



Use jet clustering history to build binary tree

Train recursive neural net on jet tree to learn “embedding” for classifier.

Graphs / Point clouds

Hu & Gouskos 1902.08570

- Particle cloud
  - particles are intrinsically unordered
  - primary information:
    - 2D coordinates in the η-φ space
  - but also additional "features":
    - energy/momenta
    - charge/particle type
    - track quality/impact parameters/etc.