Analyses Report

Meisam Ghasemi Bostanabad

Performance report meeting:
2025-2-24

Overview

Birth of AI

1952-1956

In the 1940s and 50s, a handful of scientists from a variety of fields (mathematics, psychology, engineering, economics and political science) began to discuss the possibility of creating an artificial brain.

Symbolic AI

1956-1974

- The programs developed in the years after the Dartmouth Workshop were,
- Computers were solving algebra word problems, proving theorems in geometry and learning to speak English.

Boom

1980-1987

- Rise of Expert Systems
- The Knowledge Revolution
- The Money Return

AI Winter

1974-1980

- Limited Computer Power
- There are many problems that can probably only be solved in exponential time
- Can still only handle trivial versions of the problems.
- The end of funding

Bust : 2nd AI Winter

1987-1993

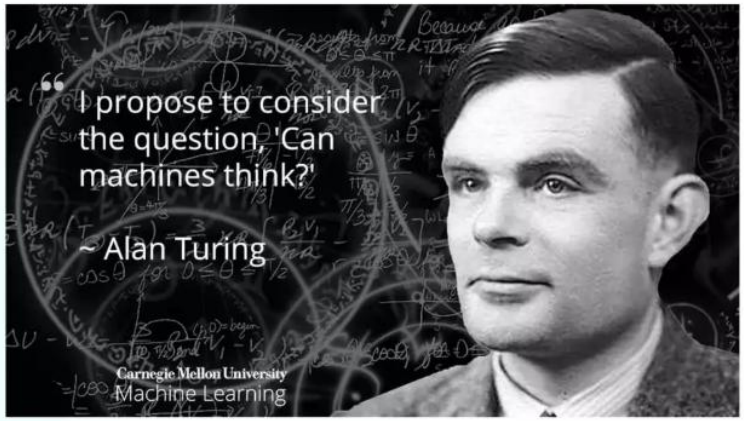
The collapse was due to the failure of commercial vendors to develop a wide variety of workable solutions.As dozens of companies failed, the perception was that the technology was not viable

Overview

1950 - Alan Turing

Can Machine Think?

1950: Alan Turing publishes "Computing Machinery and Intelligence" which proposes the Turing Test, a method for determining whether a machine can exhibit intelligent behaviour equivalent to, or indistinguishable from, that of a human.



Computing Machinery and Intelligence

A. M. Turing

1950

1 The Imitation Game

I propose to consider the question, “Can machines think?” This should begin with definitions of the meaning of the terms “machine” and “think.” The definitions might be framed so as to reflect so far as possible the normal use of the words, but this attitude is dangerous, If the meaning of the words “machine” and “think” are to be found by examining how they are commonly used it is difficult to escape the conclusion that the meaning and the answer to the question, “Can machines think?” is to be sought in a statistical survey such as a Gallup poll. But this is absurd. Instead of attempting such a definition I shall replace the question by another, which is closely related to it and is expressed in relatively unambiguous words.

https://web.iitd.ac.in/~sumeet/Turing50.pdf

Overview

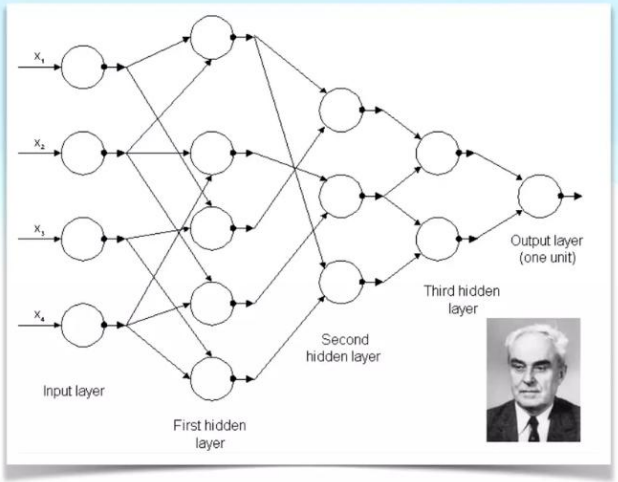
1965 - Anatolii Gershman, Alexey Ivakhnenko, Valentin Lapa

Deep Learning - Multi Layered Perceptron

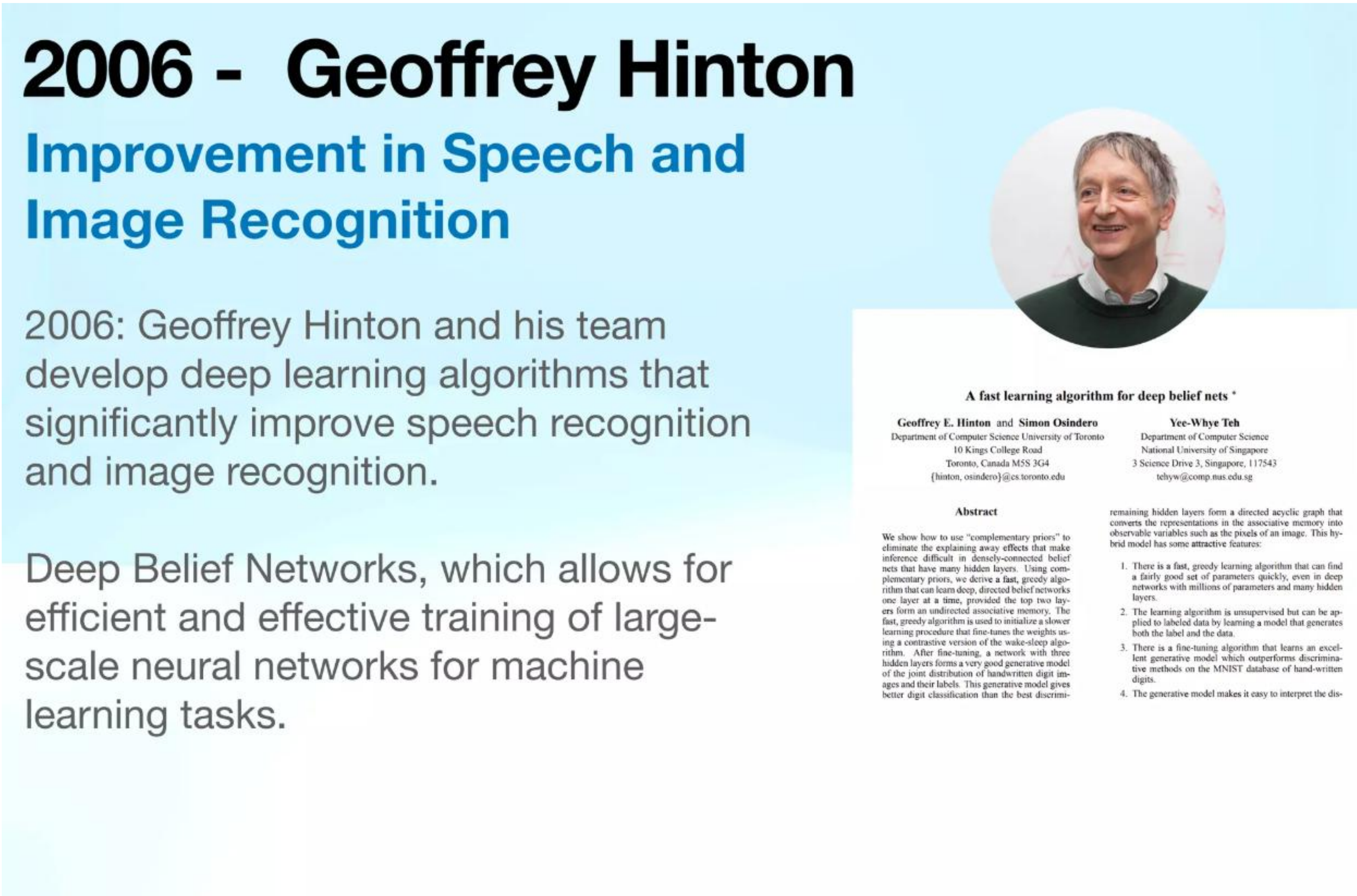
1965 : **The Group Method of Data Handling** (GMDH) is a data-driven approach to modeling that is based on a multi-layered architecture of interconnected polynomial models.

A multi-layer perceptron (MLP) is a type of artificial neural network (ANN) that is composed of multiple layers of interconnected nodes, or "neurons". MLPs are typically used for supervised learning tasks, such as classification and regression.

Ivakhnenko is often considered as the father of **deep learning.**



Overview



A fast learning algorithm for deep belief nets *

Geoffrey E. Hinton and **Simon Osindero**
Department of Computer Science University of Toronto
10 Kings College Road
Toronto, Canada M5S 3G4
{hinton, osindero}@cs.toronto.edu

Yee-Whye Teh
Department of Computer Science
National University of Singapore
3 Science Drive 3, Singapore, 117543
tehyw@comp.nus.edu.sg

Abstract

We show how to use "complementary priors" to eliminate the explaining away effects that make inference difficult in densely-connected belief nets that have many hidden layers. Using complementary priors, we derive a fast, greedy algorithm that can learn deep, directed belief networks one layer at a time, provided the top two layers form an undirected associative memory. The fast, greedy algorithm is used to initialize a slower learning procedure that fine-tunes the weights using a contrastive version of the wake-sleep algorithm. After fine-tuning, a network with three hidden layers forms a very good generative model of the joint distribution of handwritten digit images and their labels. This generative model gives better digit classification than the best discrimi- remaining hidden layers form a directed acyclic graph that converts the representations in the associative memory into observable variables such as the pixels of an image. This hybrid model has some attractive features:

1. There is a fast, greedy learning algorithm that can find a fairly good set of parameters quickly, even in deep networks with millions of parameters and many hidden layers.
2. The learning algorithm is unsupervised but can be applied to labeled data by learning a model that generates both the label and the data.
3. There is a fine-tuning algorithm that learns an excellent generative model which outperforms discriminative methods on the MNIST database of hand-written digits.
4. The generative model makes it easy to interpret the dis-

Overview

Large Language Model

2015: OpenAI is founded by a group of entrepreneurs - Elon Musk, Sam Altman, Reid Hoffman etc - they pledged $1Billion

2017: OpenAI releases GPT-1

2018: OpenAI releases GPT-2

2019: Microsoft backed OpenAI with $1Billion

2020: OpenAI releases a new version of GPT-3

2020: OpenAI releases a tool known as DALL-E

2021: OpenAI announces plans to develop and release GPT-3 under an open-source license.

2022: OpenAI releases GPT-3 Prime

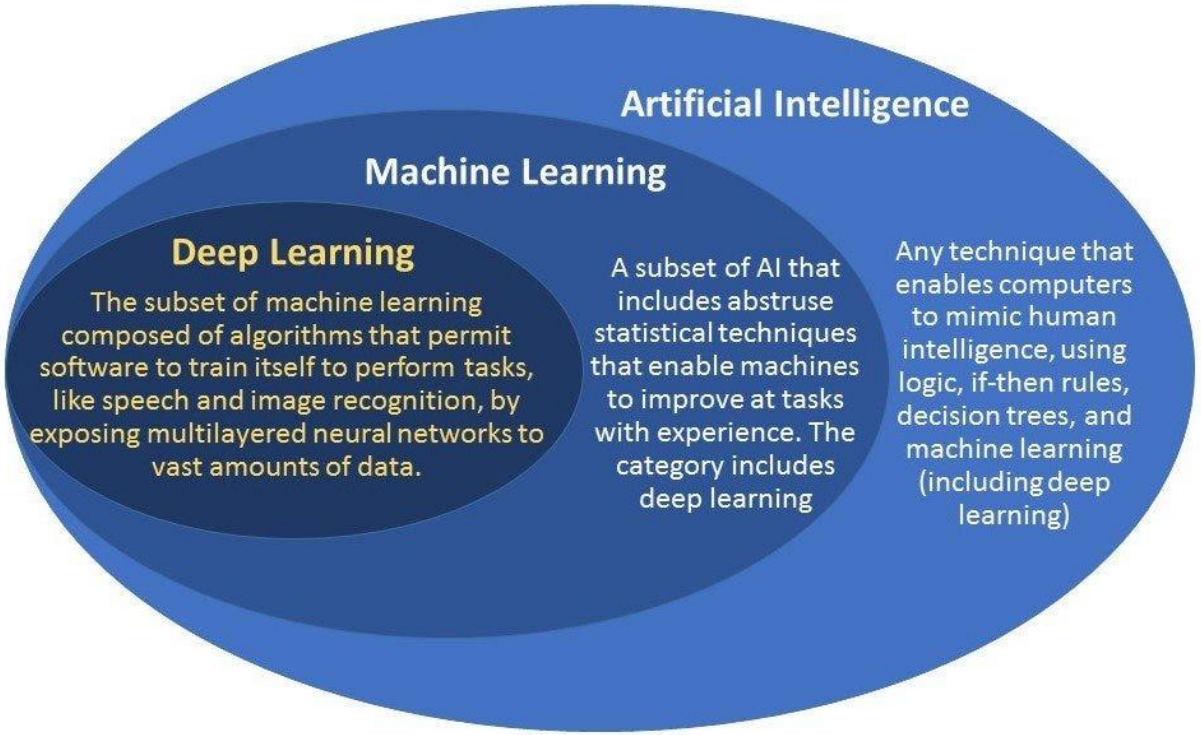
~ In Trillion
GPT-4

Significance of Number of Params

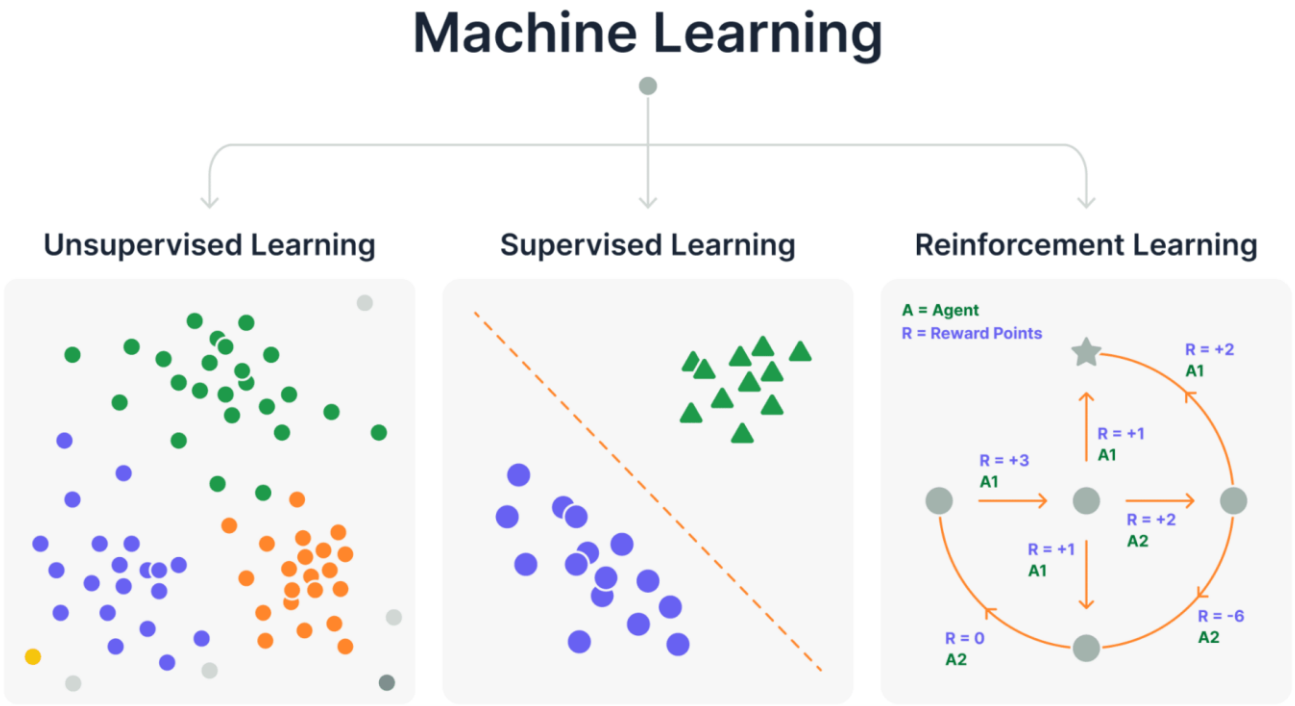
These are tuneable variables that the model has learned during the training process.

More params means more flexibility in the model's ability to generate diverse and coherent text output

Overview


Artificial Intelligence
Any technique that enables computers to mimic human intelligence, using logic, if-then rules, decision trees, and machine learning (including deep learning)
Machine Learning
A subset of AI that includes abstruse statistical techniques that enable machines to improve at tasks with experience. The category includes deep learning
Deep Learning
The subset of machine learning composed of algorithms that permit software to train itself to perform tasks, like speech and image recognition, by exposing multilayered neural networks to vast amounts of data.

Overview


Machine Learning
Unsupervised Learning
Supervised Learning
Reinforcement Learning
A = Agent
R = Reward Points
R = +2
A1
R = +1
A1
R = +3
A1
R = +2
A2
R = +1
A1
R = -6
A2
R = 0
A2

What about ML included

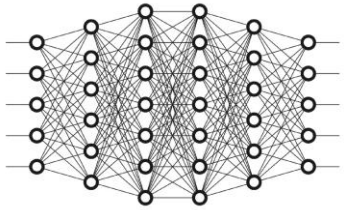
- Using machine learning (ML) models (mainly deep learning and Gen AI) in:

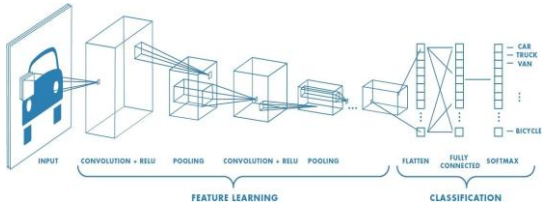
What about ML not included

Deep Neural Network
(Multi layer perceptron)

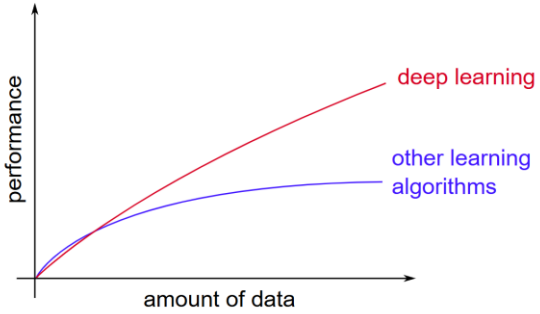
Why Deep Learning Now?

1. Better algorithm and understanding





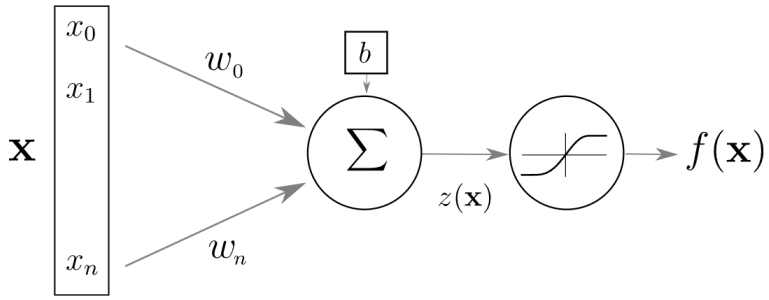
3. Data with labels



2. Computing power (GPU and TPU)

4. Open source tools and models

Simplest DNN

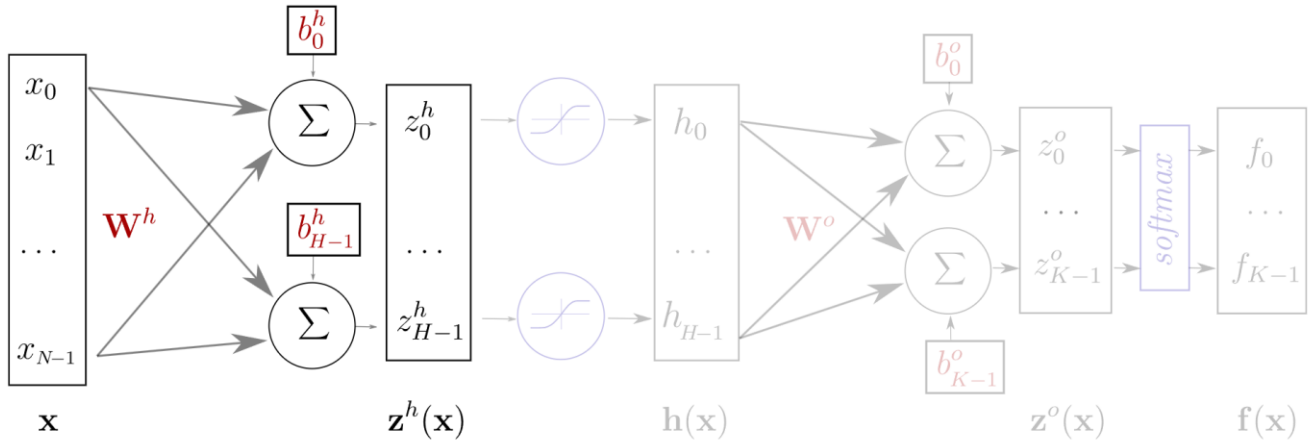


$$z(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$$

$$f(\mathbf{x}) = g(\mathbf{w}^T\mathbf{x} + b)$$

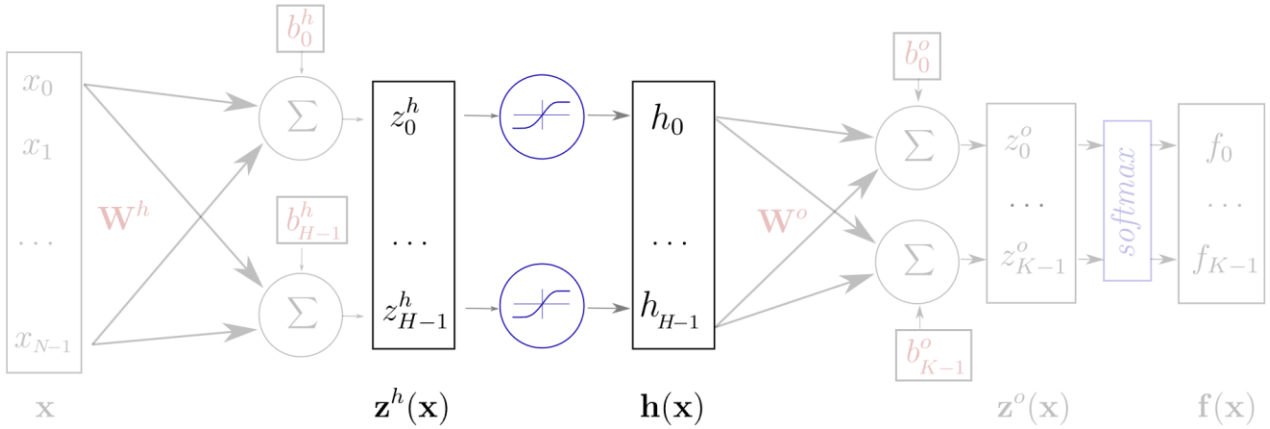
- $\mathbf{x}, f(\mathbf{x})$ input and output
- $z(\mathbf{x})$ pre-activation
- $\mathbf{w}, b$ weights and bias
- g activation function

Overview



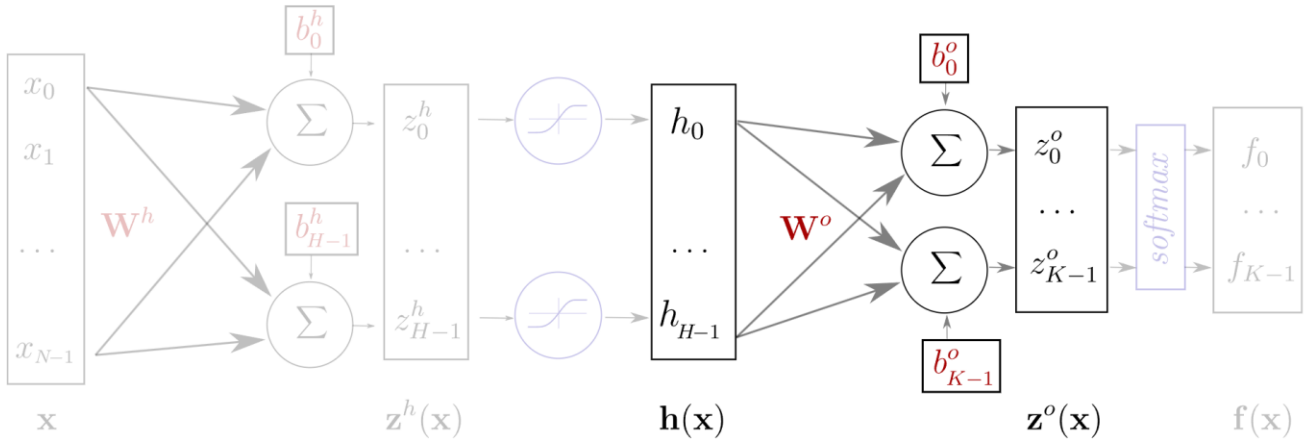
- $\mathbf{z}^h(\mathbf{x}) = \mathbf{W}^h\mathbf{x} + \mathbf{b}^h$
- $\mathbf{h}(\mathbf{x}) = g(\mathbf{z}^h(\mathbf{x})) = g(\mathbf{W}^h\mathbf{x} + \mathbf{b}^h)$
- $\mathbf{z}^o(\mathbf{x}) = \mathbf{W}^o\mathbf{h}(\mathbf{x}) + \mathbf{b}^o$
- $\mathbf{f}(\mathbf{x}) = softmax(\mathbf{z}^o) = softmax(\mathbf{W}^o\mathbf{h}(\mathbf{x}) + \mathbf{b}^o)$

Overview



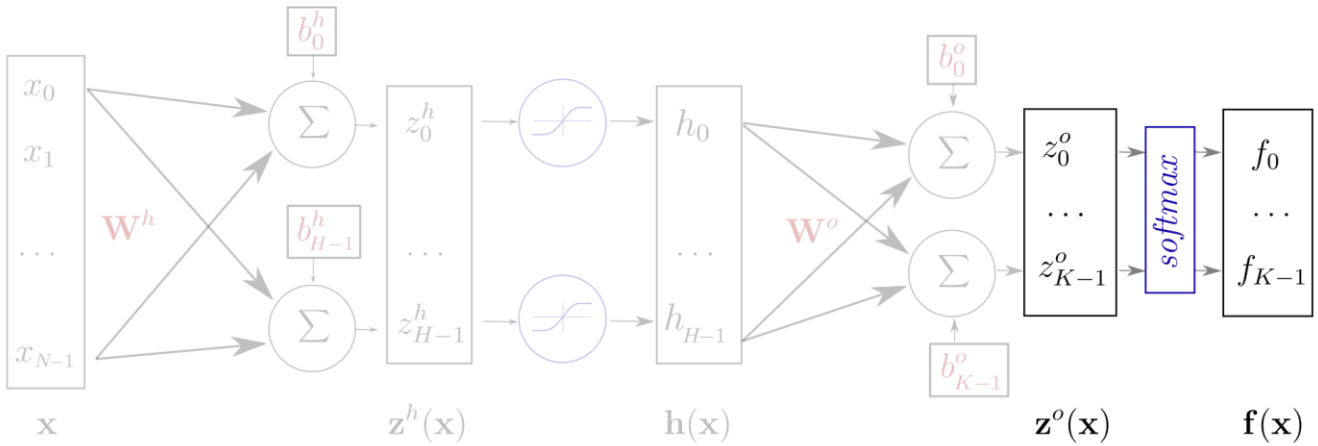
- $\mathbf{z}^h(\mathbf{x}) = \mathbf{W}^h\mathbf{x} + \mathbf{b}^h$
- $\mathbf{h}(\mathbf{x}) = g(\mathbf{z}^h(\mathbf{x})) = g(\mathbf{W}^h\mathbf{x} + \mathbf{b}^h)$
- $\mathbf{z}^o(\mathbf{x}) = \mathbf{W}^o\mathbf{h}(\mathbf{x}) + \mathbf{b}^o$
- $\mathbf{f}(\mathbf{x}) = softmax(\mathbf{z}^o) = softmax(\mathbf{W}^o\mathbf{h}(\mathbf{x}) + \mathbf{b}^o)$

Overview



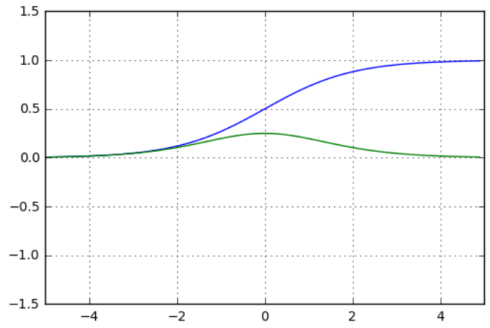
- $\mathbf{z}^h(\mathbf{x}) = \mathbf{W}^h\mathbf{x} + \mathbf{b}^h$
- $\mathbf{h}(\mathbf{x}) = g(\mathbf{z}^h(\mathbf{x})) = g(\mathbf{W}^h\mathbf{x} + \mathbf{b}^h)$
- $\mathbf{z}^o(\mathbf{x}) = \mathbf{W}^o\mathbf{h}(\mathbf{x}) + \mathbf{b}^o$
- $\mathbf{f}(\mathbf{x}) = softmax(\mathbf{z}^o) = softmax(\mathbf{W}^o\mathbf{h}(\mathbf{x}) + \mathbf{b}^o)$

Overview



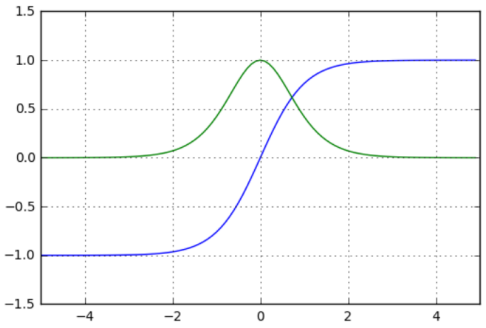
- $\mathbf{z}^h(\mathbf{x}) = \mathbf{W}^h\mathbf{x} + \mathbf{b}^h$
- $\mathbf{h}(\mathbf{x}) = g(\mathbf{z}^h(\mathbf{x})) = g(\mathbf{W}^h\mathbf{x} + \mathbf{b}^h)$
- $\mathbf{z}^o(\mathbf{x}) = \mathbf{W}^o\mathbf{h}(\mathbf{x}) + \mathbf{b}^o$
- $\mathbf{f}(\mathbf{x}) = softmax(\mathbf{z}^o) = softmax(\mathbf{W}^o\mathbf{h}(\mathbf{x}) + \mathbf{b}^o)$

What about ML not included



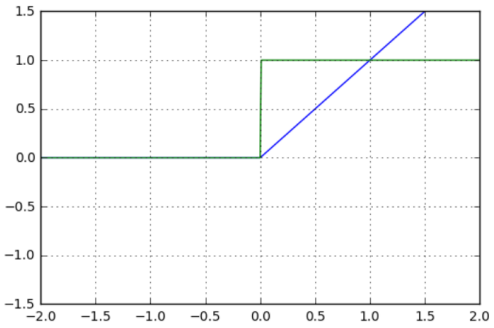
$$\text{sigm}(x) = \frac{1}{1+e^{-x}}$$

$$\text{sigm}'(x) = \text{sigm}(x)(1-\text{sigm}(x))$$



$$\tanh(x) = \frac{e^{2x}-1}{e^{2x}+1}$$

$$\tanh'(x) = 1-\tanh(x)^2$$



$$\text{relu}(x) = \max(0,x)$$

$$\text{relu}'(x) = 1_{x>0}$$

$$softmax(\mathbf{x}) = \frac{1}{\sum_{i=1}^{n} e^{x_i}} \cdot \begin{bmatrix} e^{x_1} \\ e^{x_2} \\ \vdots \\ e^{x_n} \end{bmatrix}$$

What about ML not included

Find parameters $\theta = (\mathbf{W}^h; \mathbf{b}^h; \mathbf{W}^o; \mathbf{b}^o)$ that minimize the **negative log likelihood** (or cross entropy)

$$l(\mathbf{f}(\mathbf{x}^s;\theta), y^s) = nll(\mathbf{x}^s, y^s; \theta) = -\log \mathbf{f}(\mathbf{x}^s;\theta)_{y^s}$$

example $y^s = 3$

$$l(\mathbf{f}(\mathbf{x}^s;\theta), y^s) = l\left(\begin{bmatrix} f_0 \\ \dots \\ f_3 \\ \dots \\ f_{K-1} \end{bmatrix}, \begin{bmatrix} 0 \\ \dots \\ 1 \\ \dots \\ 0 \end{bmatrix}\right) = -\log\ f_3$$

What about ML not included

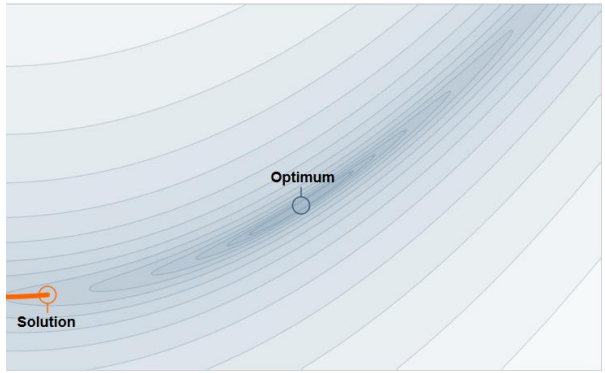
Initialize θ randomly

For E epochs perform:

- Randomly select a small batch of samples $(B \subset S)$
  - Compute gradients: $\Delta = \nabla_\theta L_B(\theta)$
  - Update parameters: $\theta \leftarrow \theta - \eta\Delta$
  - $\eta > 0$ is called the learning rate
- Repeat until the epoch is completed (all of S is covered)

What about ML not included

- Very sensitive:
  - Too high $\rightarrow$ early plateau or even divergence
  - Too low $\rightarrow$ slow convergence
  - Try a large value first: $\eta = 0.1$ or even $\eta = 1$
  - Divide by 10 and retry in case of divergence
- Large constant LR prevents final convergence
  - multiply η_t by $\beta < 1$ after each update
  - or monitor validation loss and divide η_t by 2 or 10 when no progress
  - See [ReduceLROnPlateau] in Keras



We often think of Momentum as a means of dampening oscillations and speeding up the iterations, leading to faster convergence. But it has other interesting behavior. It allows a larger range of step-sizes to be used, and creates its own oscillations. What is going on?

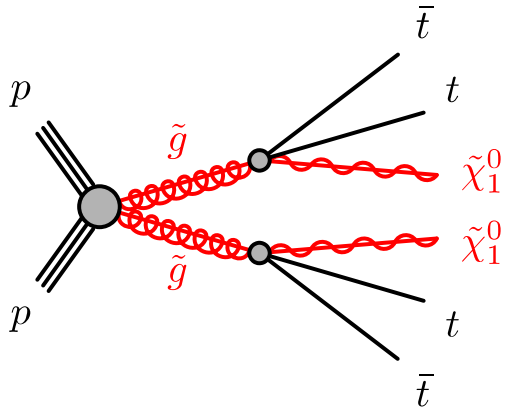
What about ML not included

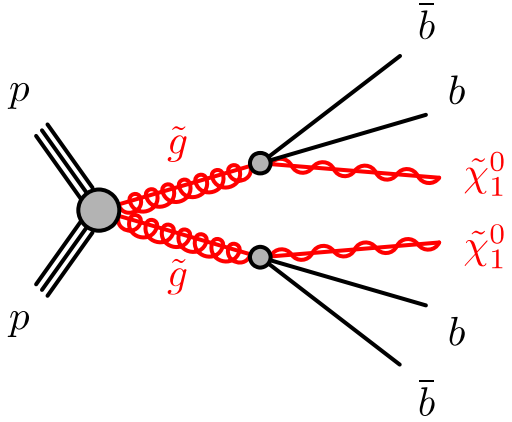
metrics

SUSY Simplified Models

We target gluino pair production (from a strong SUSY interaction) with off-shell top and bottom squarks in the decay products

Gtt





Gbb

0 lepton (0L)

1 lepton (1L)

channels

0 lepton (0L)

~~1 lepton (1L)~~

- Potentially high cross-section for gluino pair production
- Target final states with large amount of E_T^{miss} and several *b*-jets
- Main background is **semi-leptonic $t\bar{t}$**

Definition of Key Variables

$\Delta\phi_{\min}^{4j}$ to suppress multijets in which E_T^{miss} is aligned with one jet:

$$\Delta\phi_{\min}^{4j} = \min(|\phi_1 - \phi_{E_T^{miss}}|, \ldots, |\phi_4 - \phi_{E_T^{miss}}|)$$

Inclusive effective mass, to select highly energetic events:

$$m_{eff}^{incl} = \Sigma_{i \le n}\, p_T^{j_i} + \Sigma_{j \le m}\, p_T^{lep_j} + E_T^{miss}$$

m_T to remove semileptonic $t\bar{t}$ and W+jets events (region ≥ 1 lepton) :

$$m_T = \sqrt{2 p_T^l E_T^{miss} (1 - \cos\Delta\phi(\vec{p}_T^{miss}, \vec{p}_T^{\,l}))}$$

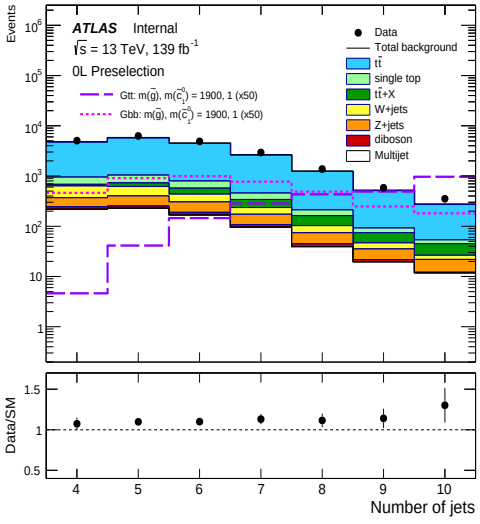
$m_{T,min}^{b-jets}$ min transverse mass between E_T^{miss} and three leading b-jets:

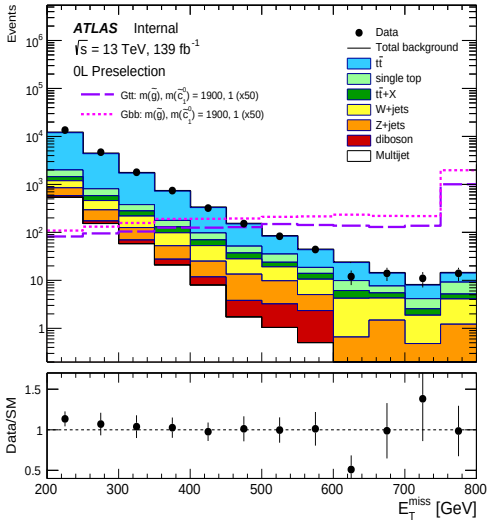
$$m_{T,min}^{b-jets} = \min_{i \le 3}\left(\sqrt{2 p_T^{b-jet_i} E_T^{miss} \left(1 - \cos\Delta\phi\left(\vec{p}_T^{miss}, \vec{p}_T^{b-jet_i}\right)\right)}\right)$$

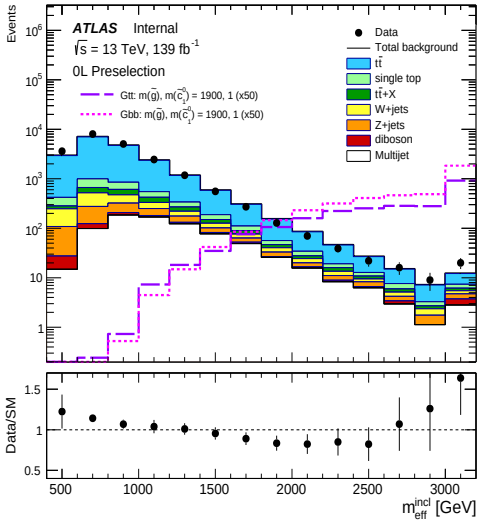
$M_J^{\Sigma,4}$ sum of the mass of re-clustered jets (higher for Gtt signal):

$$M_J^{\Sigma,4} = \Sigma_{i \le 4} m_{J,i}$$

Preselection Distributions





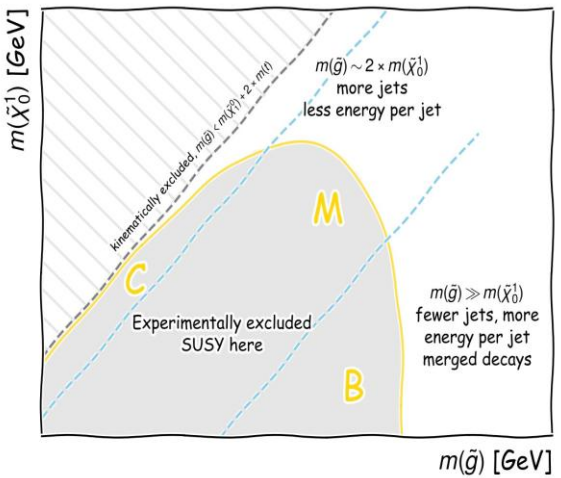


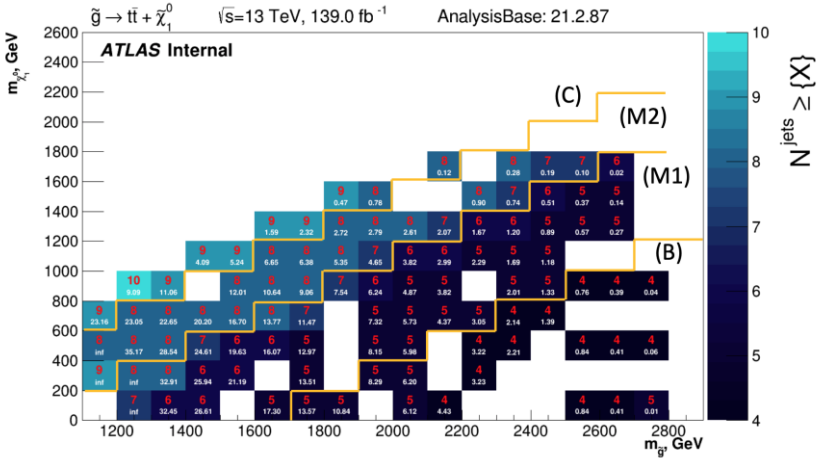
$m_{T,min}^{b-jets}$ min transverse mass between E_T^{miss} and three leading b-jets:

$$m_{T,min}^{b-jets} = \min_{i \leq 3}\left(\sqrt{2p_T^{b-jet_i} E_T^{miss}\left(1 - \cos\Delta\phi\left(\vec{p}_T^{\,miss}, \vec{p}_T^{\,b-jet_i}\right)\right)}\right)$$

Cut-and-count

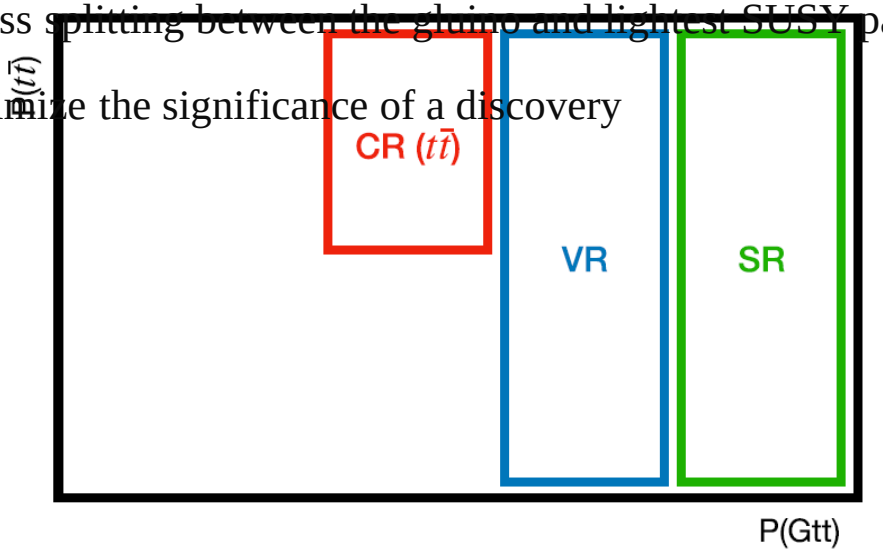
- Define **signal regions** (SRs) targeting different regio parametrized by the mass splitting between the gluino anc
  - optimized to maximize the significance of a discove
- Non-overlapping **control regions** (CRs):
  - low signal contamination
  - high ttbar purity
  - used to derive scale factors by fitting to data
- Non-overlapping **validation regions** (VRs):
  - validate CRs normalization factors
  - check the extrapolations between si control region
- If there are no large excesses or deficits in then open the box (unblind)!





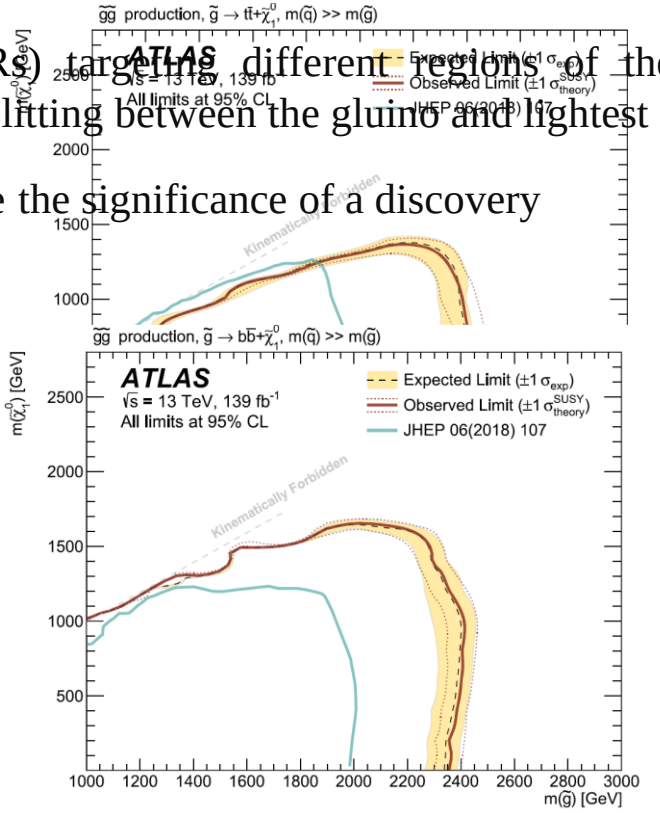
What about ML not included

- Define **signal regions** (SRs) targeting different regions of the signal grid that is parametrized by the mass splitting between the gluino and lightest SUSY particle (LSP):
  - optimized to maximize the significance of a discovery



What about ML not included

- Define **signal regions** (SRs) targeting different regions of the signal grid that is parametrized by the mass splitting between the gluino and lightest SUSY particle (LSP):
  - optimized to maximize the significance of a discovery



Convolutional Neural Network

What about ML not included

[Krizhevsky 2012]

[Ciresan et al. 2013]

[Faster R-CNN - Ren 2015]

[NVIDIA dev blog]

What about ML not included

Motivations

Standard Dense Layer for an image input:

```
x = Input((640, 480, 3), dtype='float32')
# shape of x is: (None, 640, 480, 3)
x = Flatten()(x)
# shape of x is: (None, 640 x 480 x 3)
z = Dense(1000)(x)
```

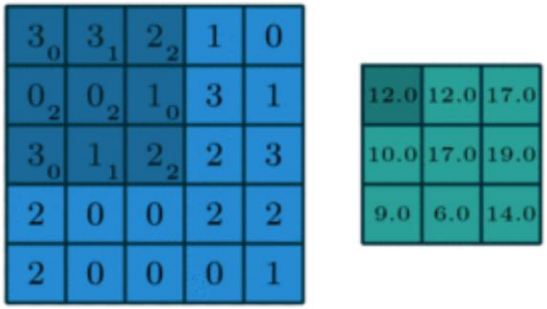
How many parameters in the Dense layer?

$640 \times 480 \times 3 \times 1000 + 1000 = 922M!$

Spatial organization of the input is destroyed by `Flatten`

We never use Dense layers directly on large images. Most standard solution is **convolution** layers

What about ML not included

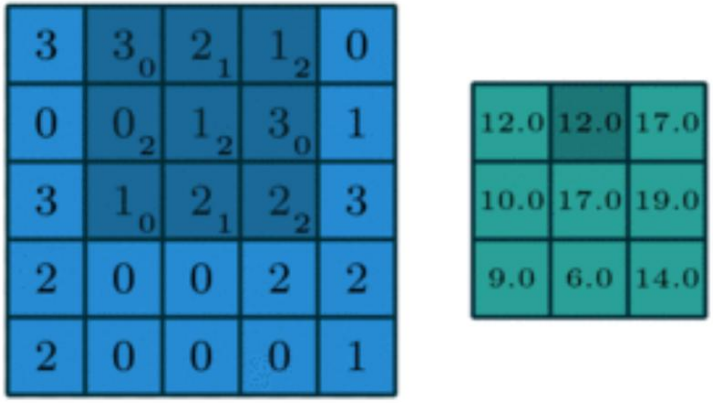


- x is a 3×3 chunk (dark area) of the image *(blue array)*
- Each output neuron is parametrized with the 3×3 weight matrix $\mathbf{w}$ *(small numbers)*

The activation obtained by sliding the 3×3 window and computing:

$$z(x) = relu(\mathbf{w}^T x + b)$$

What about ML not included

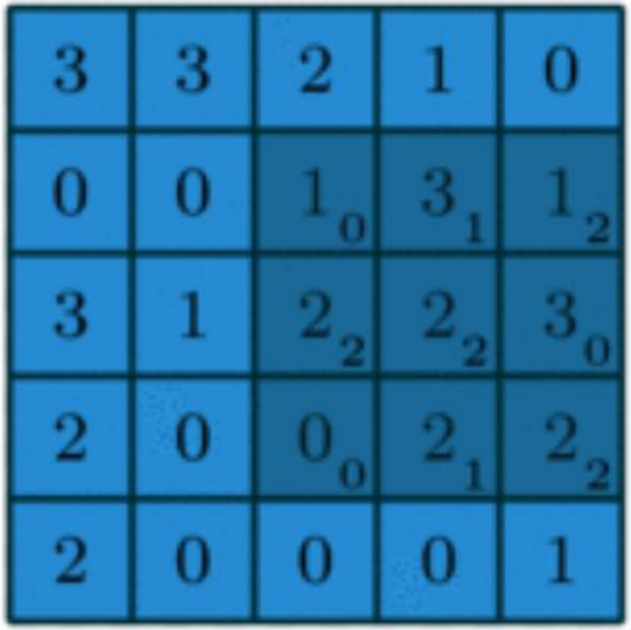


2D-convolutions (actually 2D cross-correlation):

$$(f \star g)(x, y) = \sum_n \sum_m f(n, m).\, g(x + n, y + m)$$

f is a convolution **kernel** or **filter** applied to the 2-d map g (our image)

What about ML not included

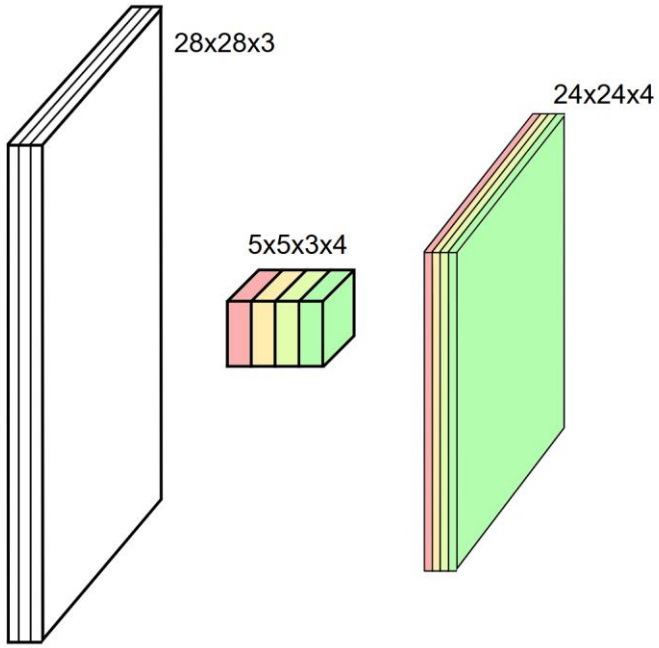


What about ML not included

Comparison to Fully connected

- Parameter sharing: reduce overfitting
- Make use of spatial structure: **strong prior** for vision!

What about ML not included

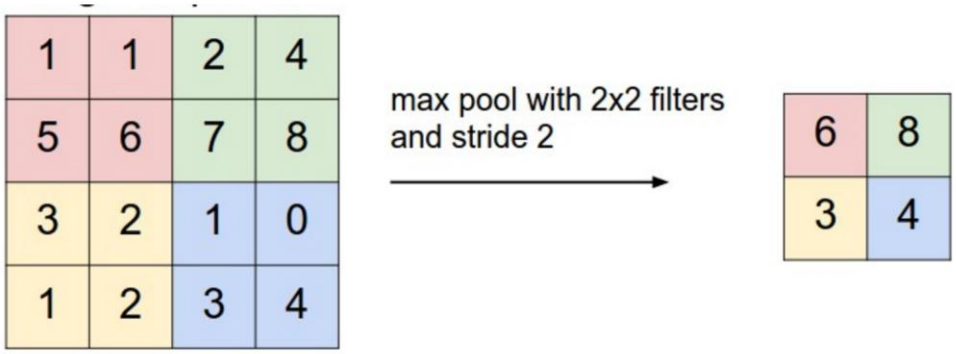


- Kernel size aka receptive field (usually 1, 3, 5, 7, 11)
- Output dimension: `length - kernel_size + 1`

What about ML not included

Pooling

- Spatial dimension reduction
- Local invariance
- No parameters: max or average of 2x2 units



What about ML not included

Classic ConvNet Architecture

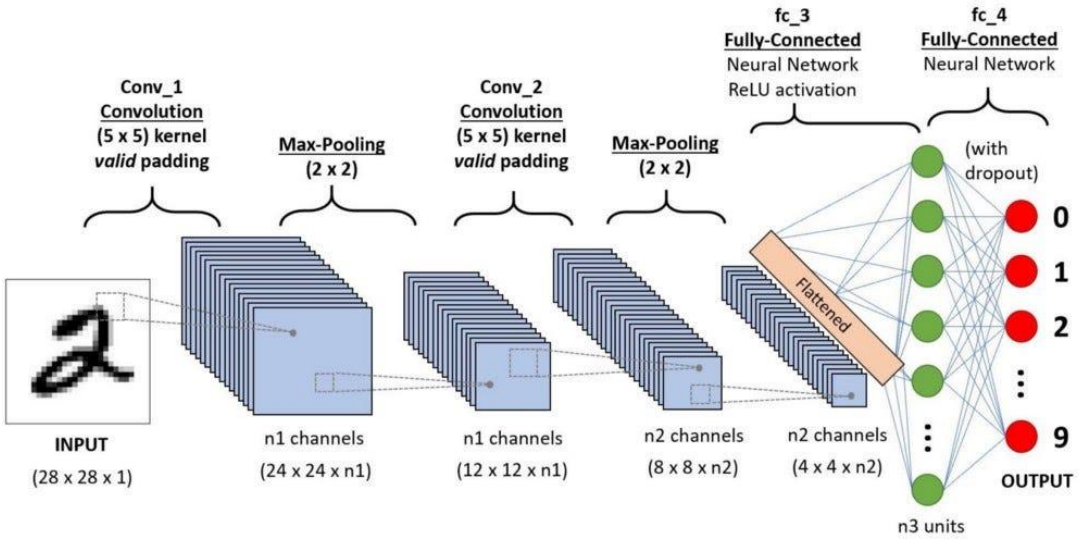
Input

Conv blocks

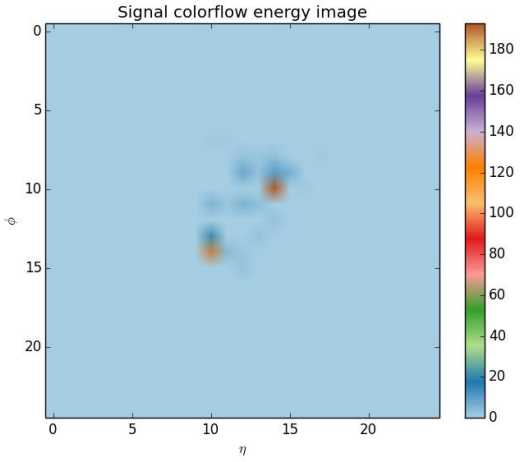
- Convolution + activation (relu)
- Convolution + activation (relu)
- ...
- Maxpooling 2x2

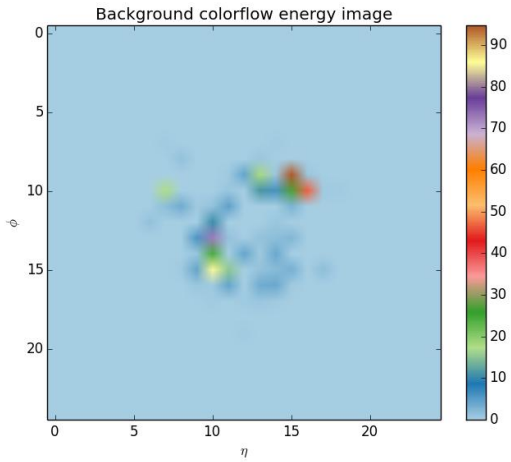
Output

- Fully connected layers
- Softmax

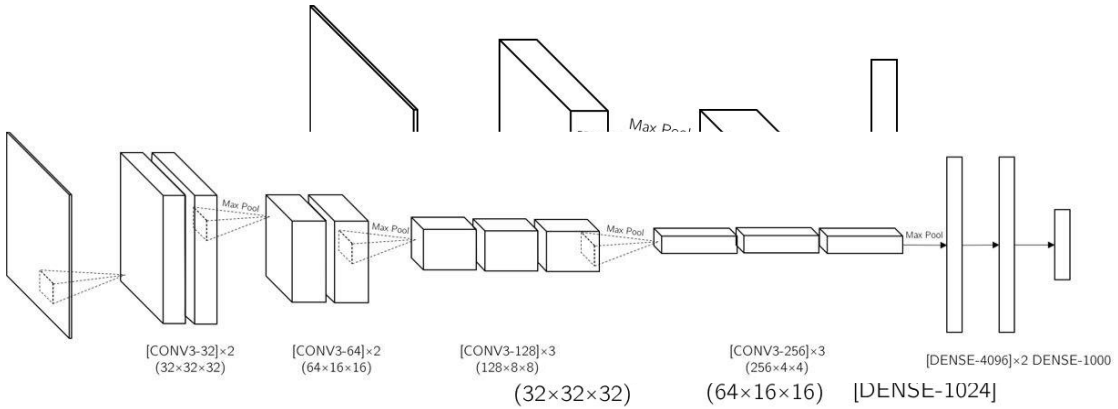


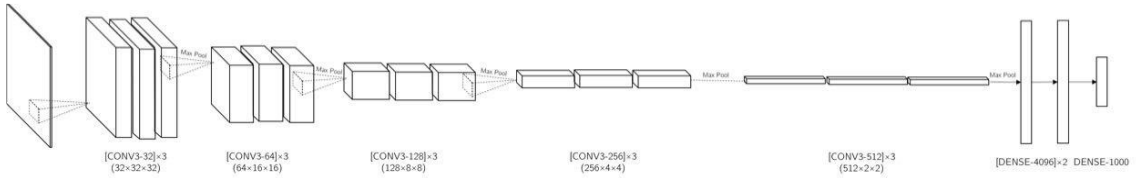
What about ML not included



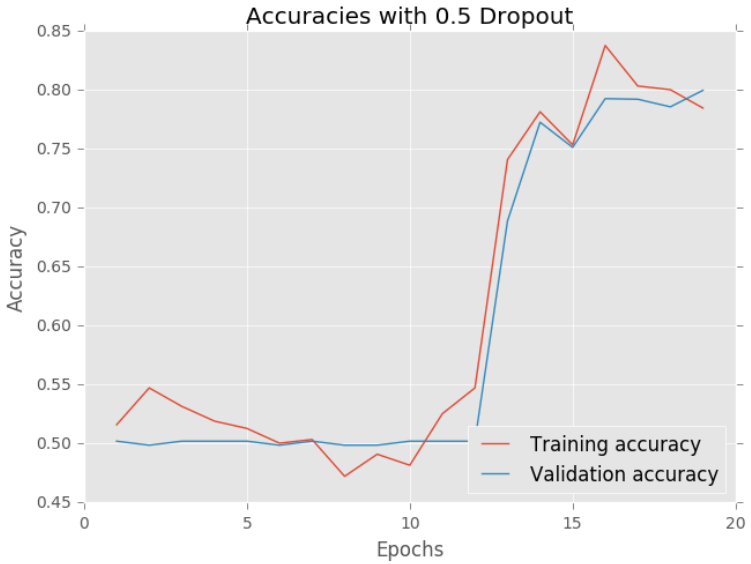


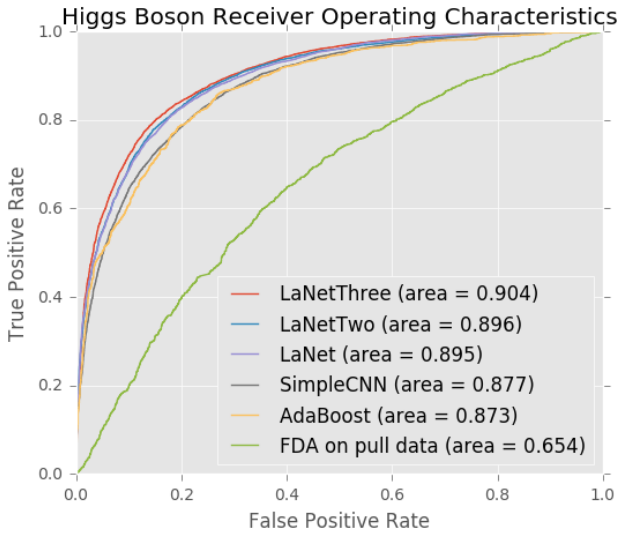
What about ML not included





What about ML not included

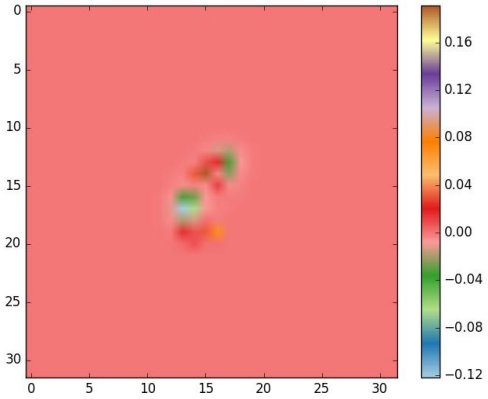




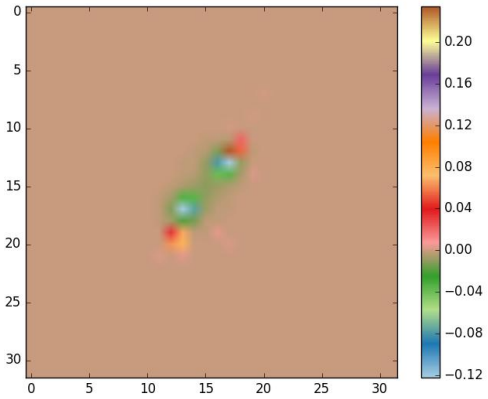
Model	AUC	Accuracy
FDA	0.654	-
AdaBoost	0.873	0.798
SimpleModel	0.877	0.774
LaNet	0.895	0.812
LaNetTwo	0.896	**0.825**
LaNetThree	**0.904**	0.820

What about ML not included

metrics



background



Higgs

Quantum Machine Learning

What about ML not included

Introduction

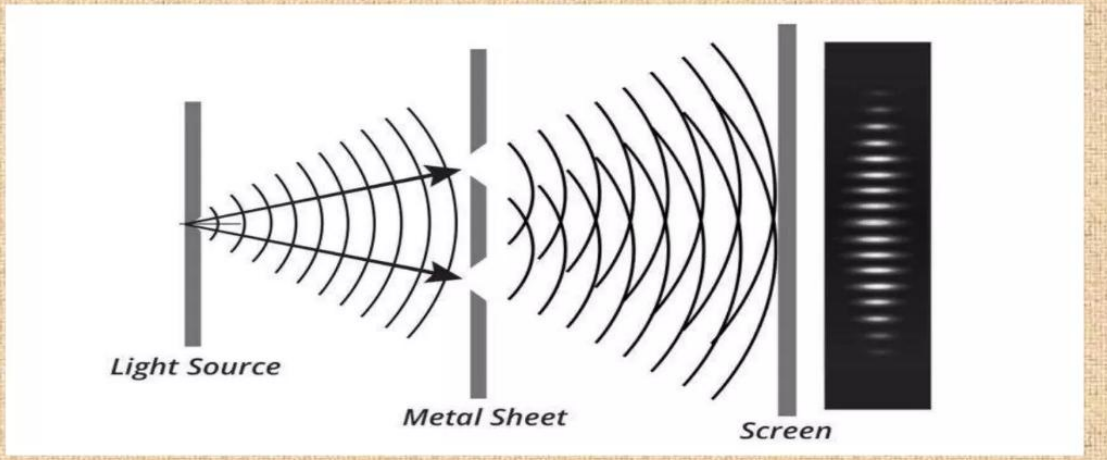
- Quantum computing is an area of computer science that uses the principles of quantum theory. Quantum theory explains the behavior of energy and material on the atomic and subatomic levels.
- Quantum computing uses subatomic particles, such as electrons or photons. Quantum bits, or qubits, allow these particles to exist in more than one state (i.e., 1 and 0) at the same time.
- A qubit can be in state $\frac{1}{\sqrt{2}}|0\rangle$ and $\frac{1}{\sqrt{2}}|1\rangle$ or (unlike a classical bit) in a linear combination of both states. The name of this phenomenon is superposition.

What about ML not included

Graphical Representation of QUBIT

Quantum Superposition

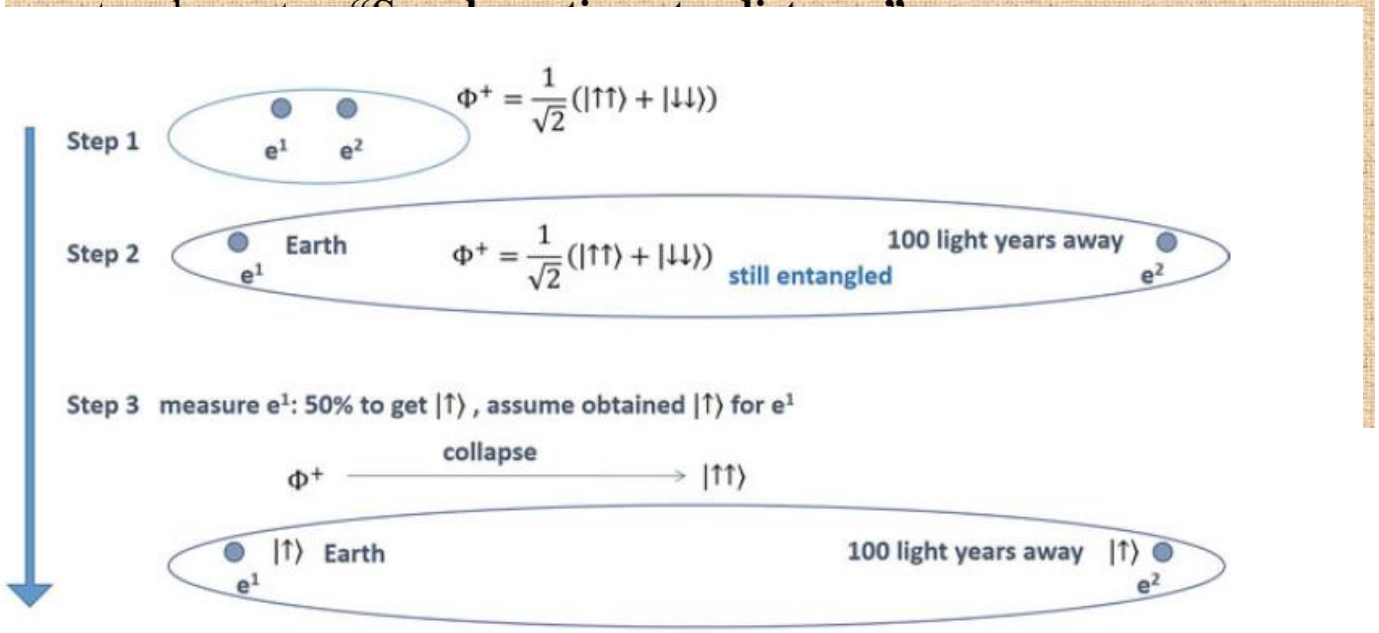
- The quantum system is capable of being in several different states at the same time.
- Example – Young's Double Slit Experiment



What about ML not included

Quantum Entanglement

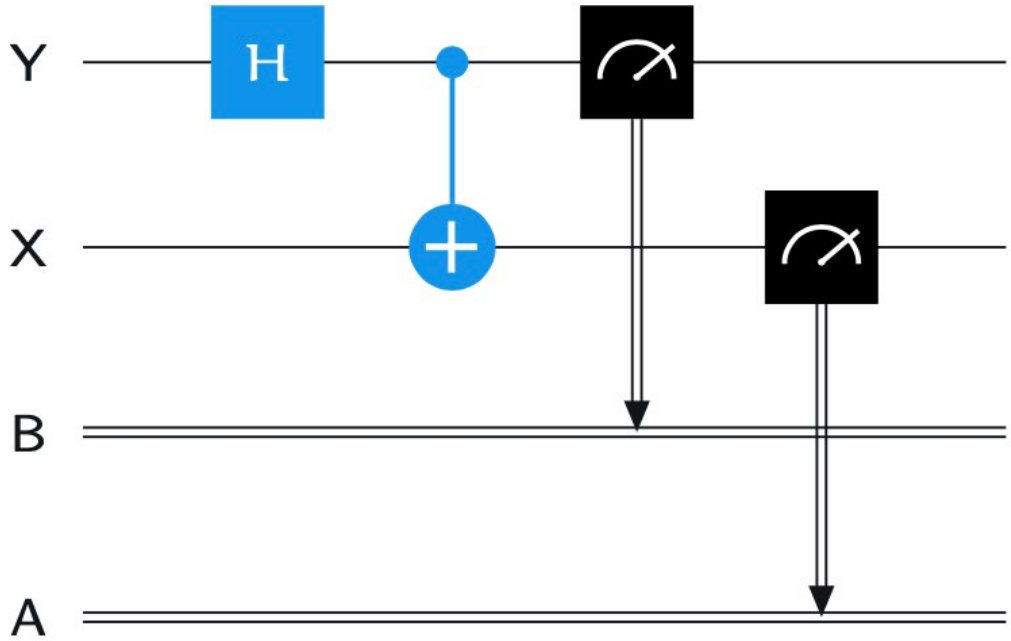
- It is an extremely strong correlation that exists between quantum particles
- Two or more quantum particles can be inextricably linked in perfect unison, even when placed at opposite ends of the universe.
- This seemingly impossible connection inspired Einstein to describe



What about ML not included

Example

Example



What about ML not included

What about ML not included

What about ML not included

What about ML not included

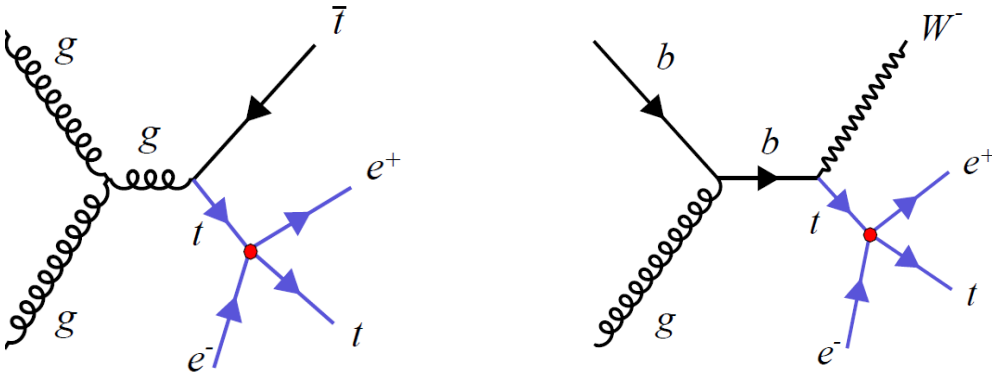
What about ML not included

What about ML not included

What about ML not included

Flavor Changing in Top sector

In this analysis we looking for FC ($t \to ull$ or $t \to cll$) in top sector as the heaviest quark which may be an indicator of new flavor physics.

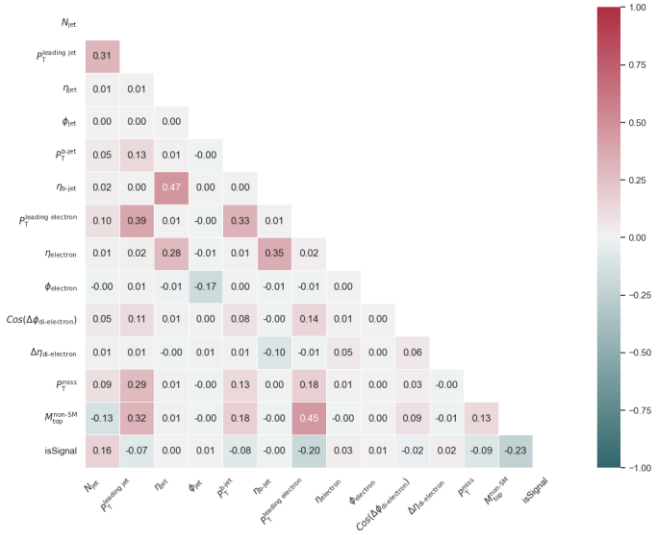


Effective Lagrangian $\mathcal{L}_{tu\ell\ell} = \frac{1}{\Lambda_\ell^2} \sum_{i,j=L,R} \left[V_{ij}^\ell \left(\bar{\ell}\gamma_\mu P_i \ell\right)\left(\bar{t}\gamma^\mu P_j u\right) + S_{ij}^\ell \left(\bar{\ell} P_i \ell\right)\left(\bar{t} P_j u\right) + T_{ij}^\ell \left(\bar{\ell}\sigma_{\mu\nu} P_i \ell\right)\left(\bar{t}\sigma_{\mu\nu} P_j u\right) \right],$

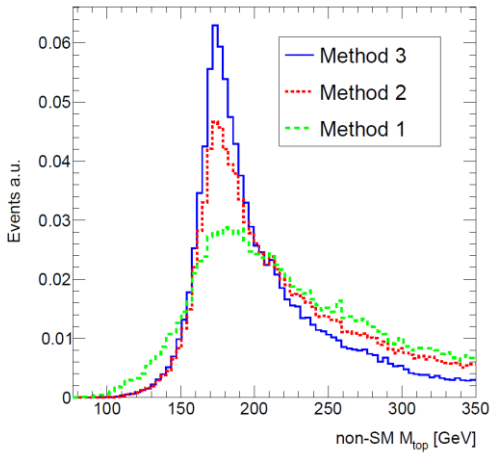
- Starting with **ttbar** and **tW**, targeting **final states** with three leptons (a pair of OS) and a b-tagged jet (one of the tops decays leptonically via w $\to l\, \nu_l$)
- There are **at least** two jets – other jets might come from showering
- The leading potential backgrounds are tZ, $t\bar{t}W$, $t\bar{t}Z$, $t\bar{t}t\bar{t}$, WZ, ZZ, $t\bar{t}$

Event generation and feature engineering

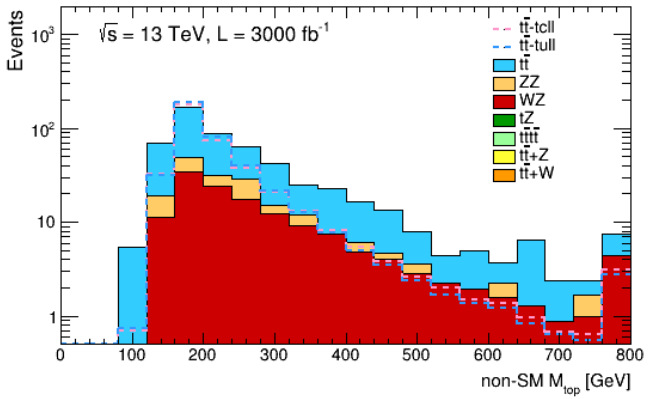
- Signal and background events are generated with MG5+PYTHIA (for PS and HAD) + Delphes (for HLLHC CMS card detection).
- Several variables are defined to be feed into the ML models for training.
- Three algorithms used to **reconstruct non-SM top mass**. The best one uses electrons and jets to get $\min(|m_{llq} - m_{top}|)$.



Feature heatmap



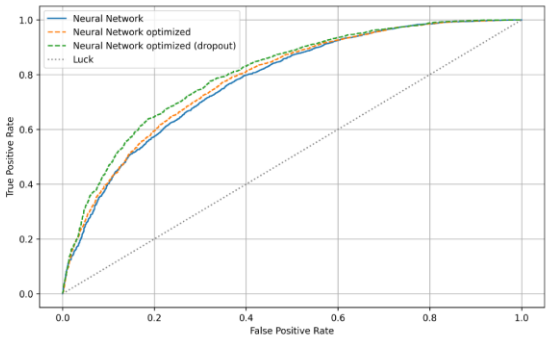
Non-SM top mass reconstruction



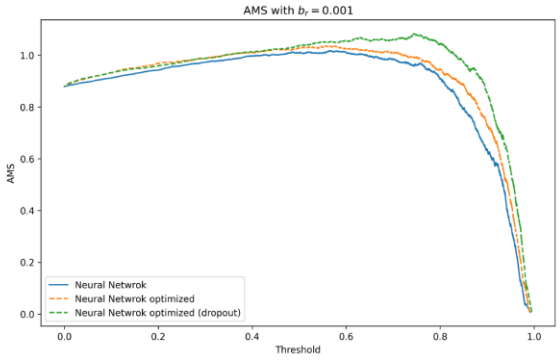
Non-SM top mass distribution

DNN and other ML classifiers

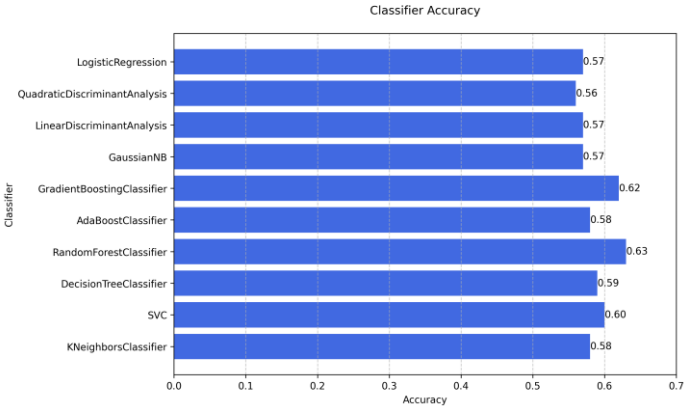
- A deep learning-based classification model (benchmark) is trained to distinguish signal from background detection).
- All hyperparameters are optimized and drop out layer + stopping layer are added to avoid overfitting. Achieves a **signal-to-background** improvement of up to 3.3x in ttbar channel.
- Other ML classifiers are trained for comparison.



ROC curve for DNN



DNN threshold using AMS

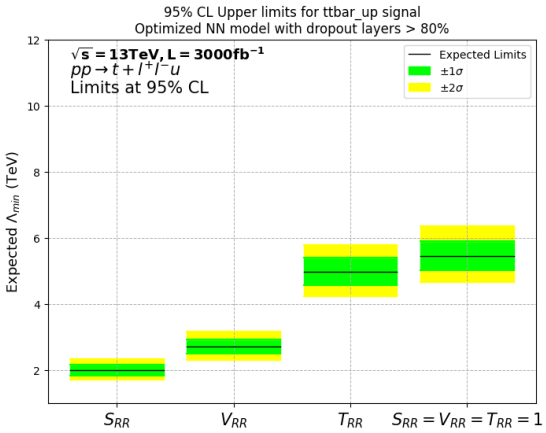


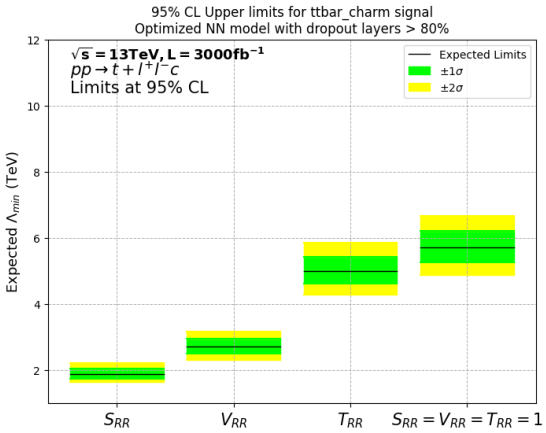
Other classifier accuracy

Exclusion limits on NP scale

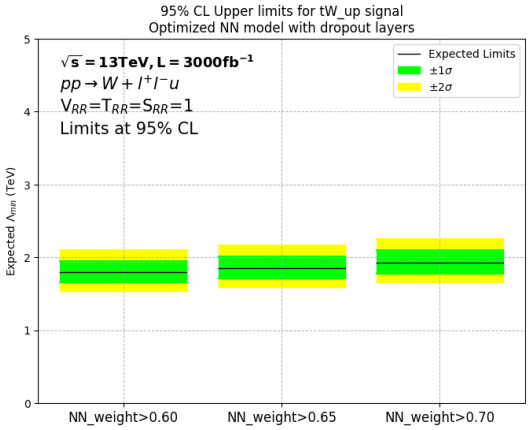
- Expected **95% CL exclusion limits** on the new physics scale (Λ) for all signal scenarios:

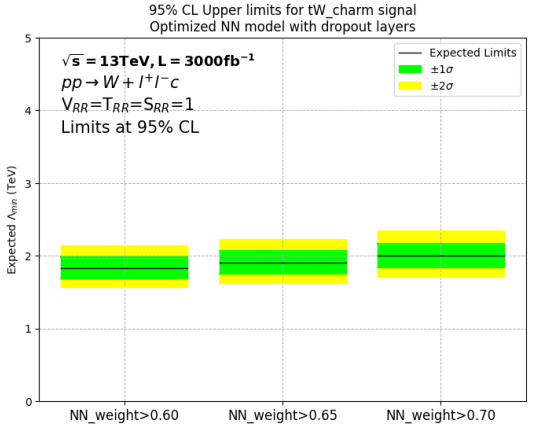
$t\bar{t}$ channel limit





tw channel limit



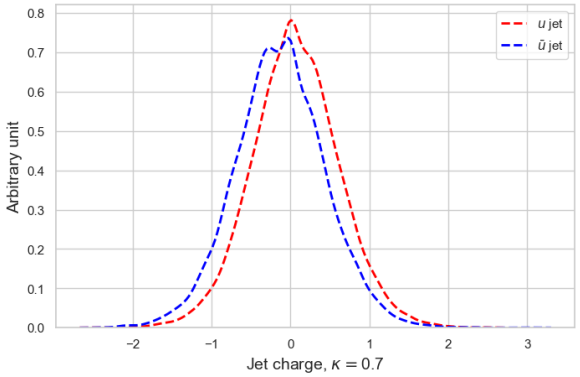


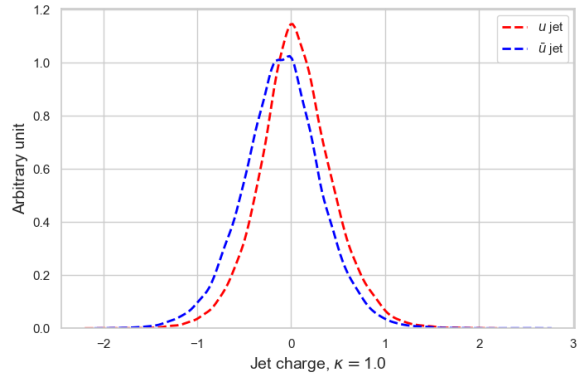
Application of Deep learning to Jet charge

- The jet charge method is crucial in verifying the top quark charge, heavy boson identification (W′/Z′), quark vs. gluon jet discrimination.
- This study explores **classical deep learning** models like DNN, CNN, GNN and **quantum ML models** for improved jet charge classification.
- The jet charge observables are defined as (where is a tunable parameter affecting charge sensitivity):

$$Q_j = \frac{1}{(p_T^{\text{jet}})^\kappa} \sum_{i \in \text{Tr}} Q_i (p_T^i)^\kappa$$

$$Q_{3,\kappa} = \frac{\sum_{i \in Tr} q_i \left|\Delta\eta_i\right|^\kappa}{\sum_{i \in Tr} \left|\Delta\eta_i\right|^\kappa}$$





Momentum weighted jet charge distribution

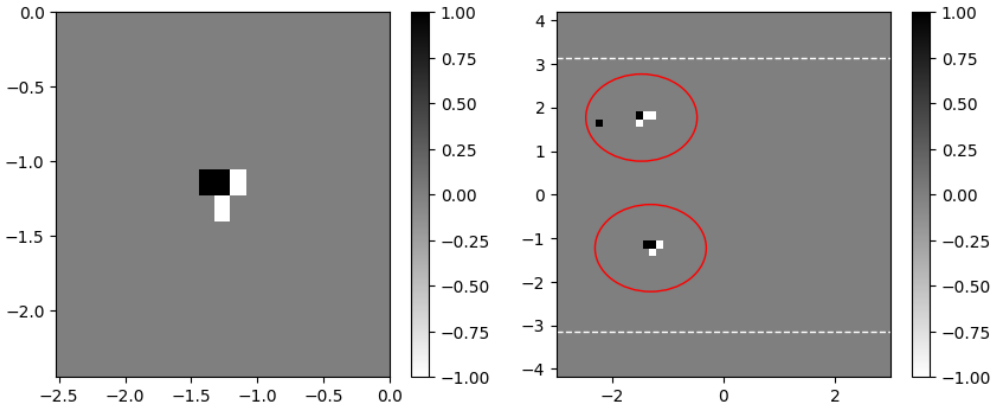
Convolutional NN and Graph NN

Convolutional Neural Networks (CNNs):

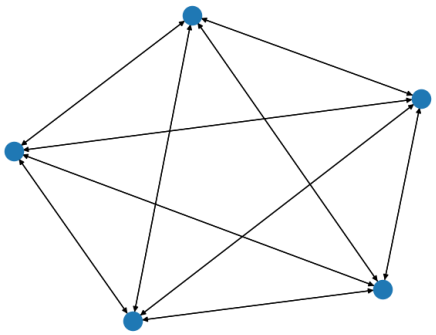
- CNNs process jet images by analyzing pixel charge distributions in **η-φ space**.
- They capture **spatial correlations** of energy deposits to classify jet charge.
- Can differentiate between quark-initiated and gluon-initiated jets using learned spatial features.

Graph Neural Networks (GNNs):

- GNNs model jets as graphs, where **tracks are nodes** and **edges encode relationships**.
- Capture relational and topological information beyond fixed-grid structures.
- Effective in handling **variable-sized track information** per jet.



Pixelated representation of jets in $\eta - \phi$ space

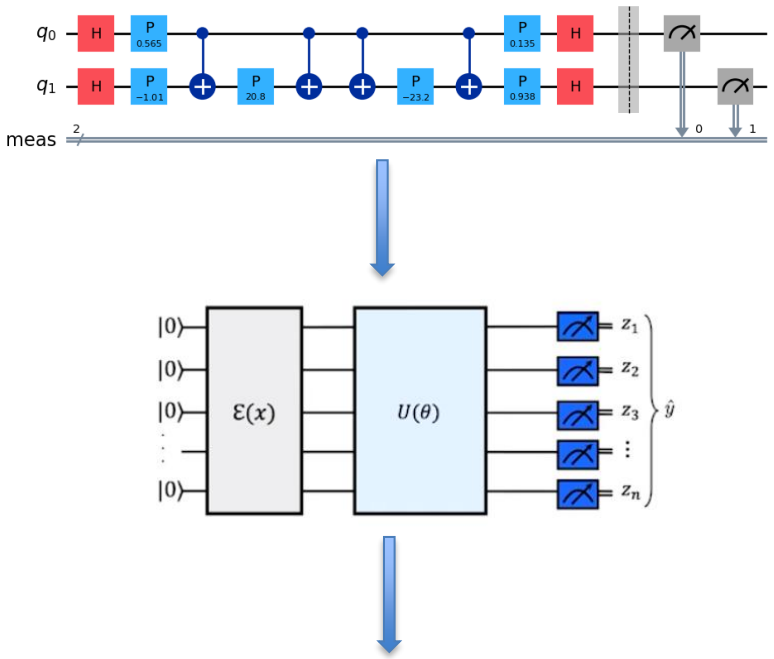


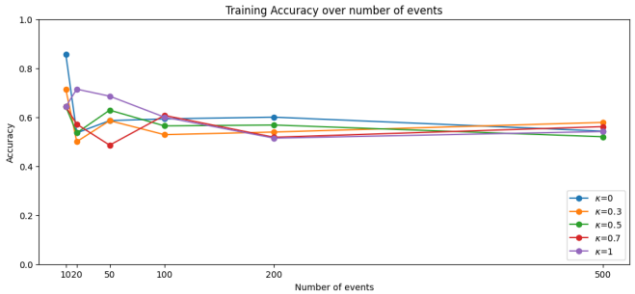
Graph representation of a leading jet

Quantum ML application

- Quantum Feature Map: Maps classical jet charge data into a higher-dimensional quantum Hilbert space using parametrized quantum circuits, enabling more expressive representations.
- Quantum Kernel: Compute similarity between quantum-embedded jets, allowing for efficient discrimination using quantum support vector machines.
- Challenges: Noisy quantum hardware, limited qubit connectivity, and circuit depth constraints impact practical implementation.

Quantum feature map and kernel



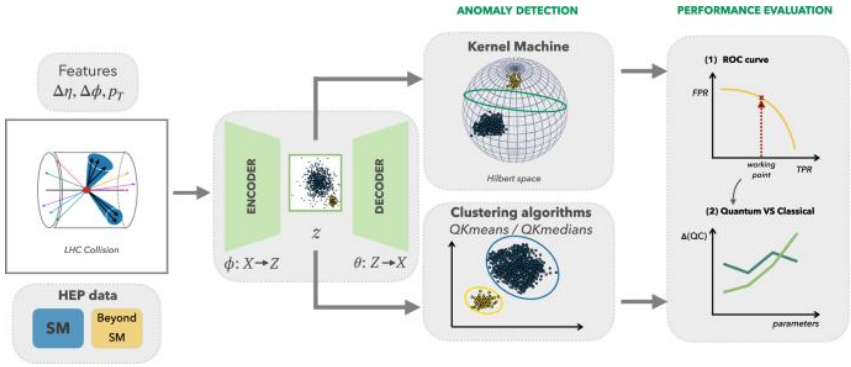


Accuracy for VQC in few events

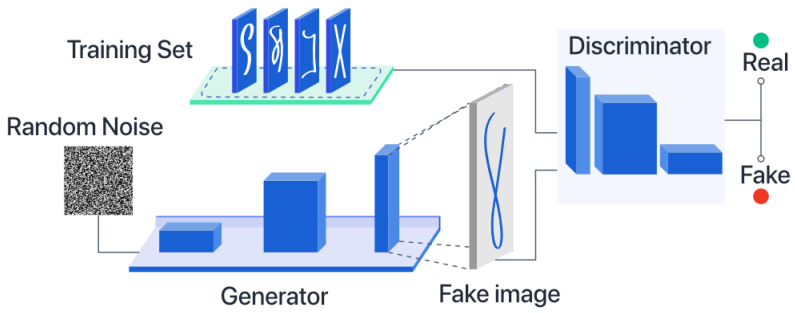
Accuracy for QSVM in few events

Anomaly detection using Gen AI

- Train Autoencoder or variational AE on SM background events and reconstruct them well.
- New Physics events, which deviate from learned patterns, lead to high reconstruction errors, signaling potential anomalies.



- In GAN-based anomaly detection, the **generator** learns to produce events that resemble Standard Model (SM) data, while the **discriminator** is trained to distinguish between real and generated events.
- if the discriminator assigns a high anomaly score (i.e., the event is unlike both real and generated SM events), it may indicate a **Beyond Standard.**



Summary & ongoing

- The paper on top FCNC is finished and ready for submission.
- The paper on classical and quantum ML applications in jet discrimination is in progress (70% complete). The remaining tasks include running QSVM on a real IBM quantum server and training GNN with GraphSAGE.
- Your feedback is welcome and greatly appreciated.