Neural Networks Architectures

2nd Terascale School of Machine Learning
1-11 March, 2021

Jean-Roch Vlimant (California Institute of Technology)
jvlimant@caltech.edu @vlimant

Outline

I. Introduction

II. Dense Networks

III. Convolutional Layers

IV. Recurrent Cells

V. Graph Models

VI. Outlook


Introduction
https://www.rugbyworld.com/news/scrum-put-in-rules-rugby-explained-88838

A Definition

“Giving computers the ability to learn without explicitly programming them” A. Samuel (1959).

Is fitting a straight line machine learning ?
Models that have enough capacity to define its own internal representation of the data to accomplish a task : **learning from data.**

In practice : a statistical method that can extract information from the data, not obviously apparent to an observer.

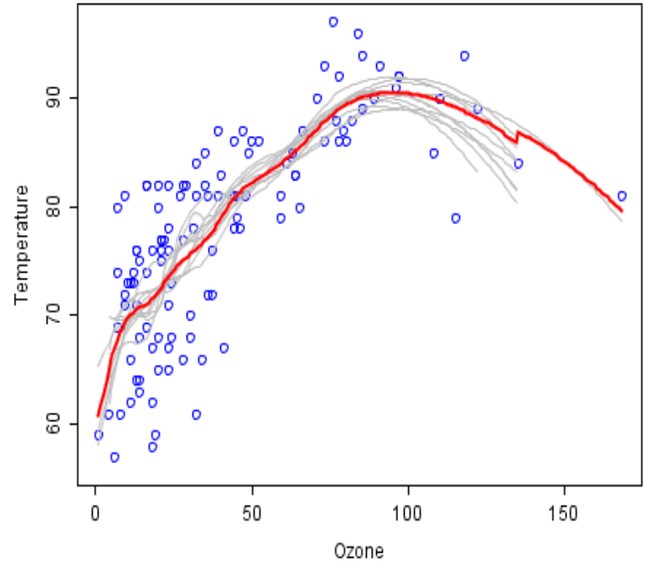
- Most approach will involve a **mathematical model** and a cost/ reward function that needs to be **optimized.**
- The more **domain knowledge** is incorporated, the better.

Supervised Learning

- Given a dataset of samples, a subset of features is qualified as **target**, and the rest as **input**
- Find a **mapping from input to target**
- The mapping should **generalize to any extension** of the given dataset, provided it is generated from the same mechanism

$$dataset \equiv \{(x_i, y_i)\}_i$$
$$find \;\; function \;\; f \;\; s.t. \;\; f(x_i) = y_i$$

- Finite set of target values :
  - → **Classification**
- Target is a continuous variable :
  - → **Regression**

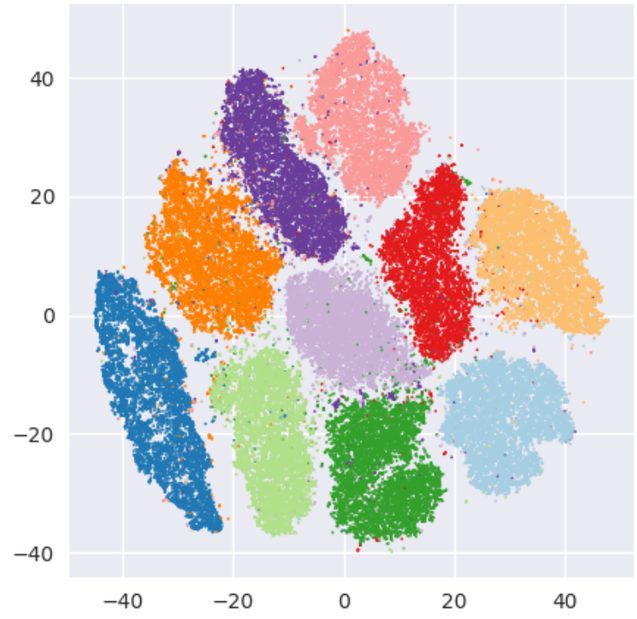


Unsupervised Learning

- Given a dataset of samples, but there is no subset of feature that one would like to predict
- Find mapping of the samples to a lower dimension manifold
- The mapping should generalize to any extension of the given dataset, provided it is generated from the same mechanism

$$dataset \equiv \{(x_i)\}_i$$
$$find \;\; f \;\; s.t. \;\; f(x_i) = p_i$$

- Manifold is a finite set
  - → **Clusterization**
- Manifold is a lower dimension manifold :
  - → **Dimensionality reduction, density estimator**

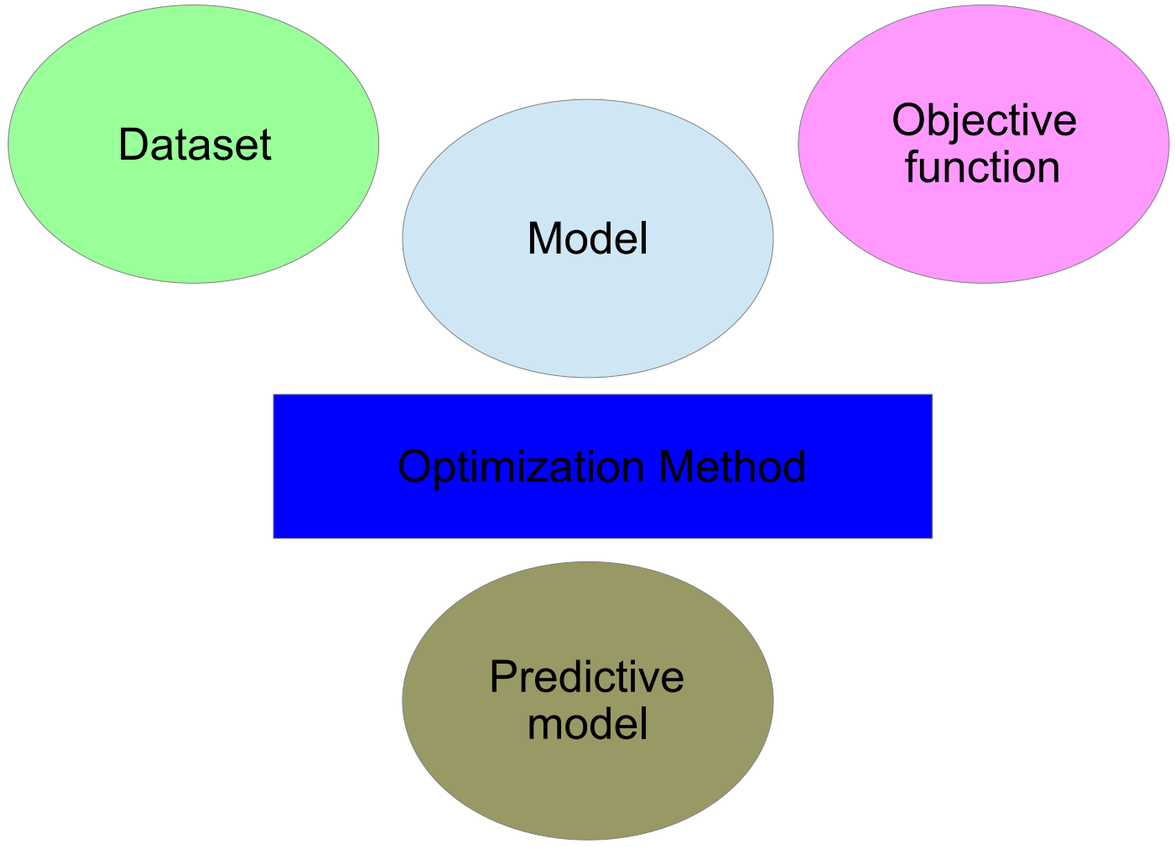


Reinforcement Learning

- Given an **environment** with multiple states, given a reward upon action being taken over a state
- Find an **action policy to drive** the environment toward maximum cumulative reward

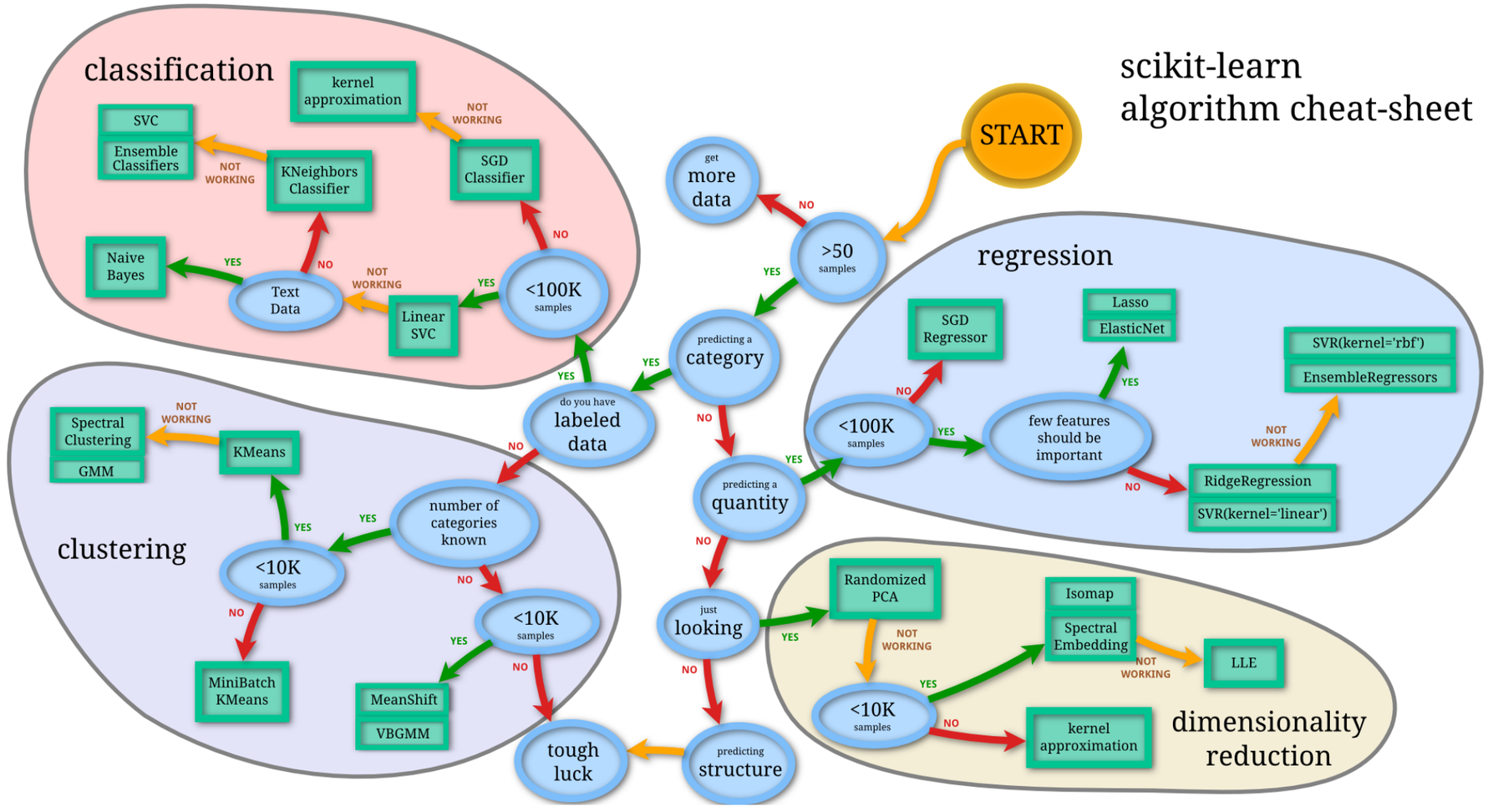
$$s_{t+1} = Env(s_t, a_t)$$
$$r_t = Rew(s_t, a_t)$$
$$\pi(a|s) = P(A_t = a | S_t = s)$$
$$find\ \pi\ s.t.\ \sum_t r_t\ is\ maximum$$

Overview



Many optimization methods adapted to the various type of the dataset, model, objective.
Gradient descent, evolutionary algorithms, ...

(Some) Machine Learning Methods



http://scikit-learn.org/stable/tutorial/index.html

There are a lot of methods out there.
Focusing on artificial neural networks in this lecture.

Artificial Neural Network

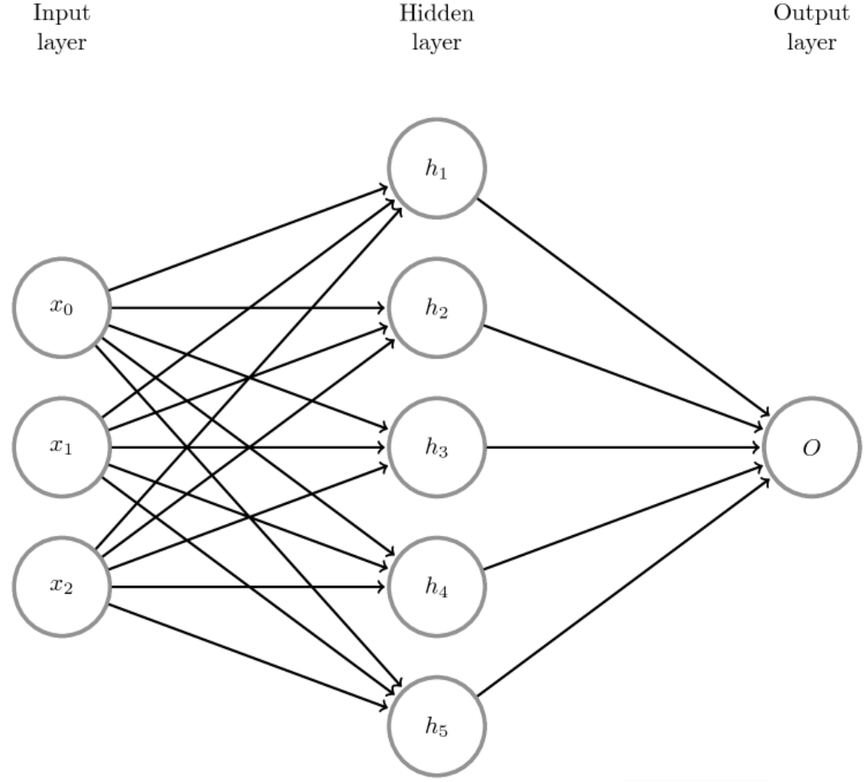
- **Biology inspired** analytical model, but **not bio-mimetic**
- Booming in recent decade thanks to large dataset, increased computational power and theoretical novelties
- Origin tied to logistic regression with change of data representation
- Part of any “deep learning” model nowadays
- Usually large number of parameters trained with stochastic gradient descent

$$h = \phi(Ux + v)$$

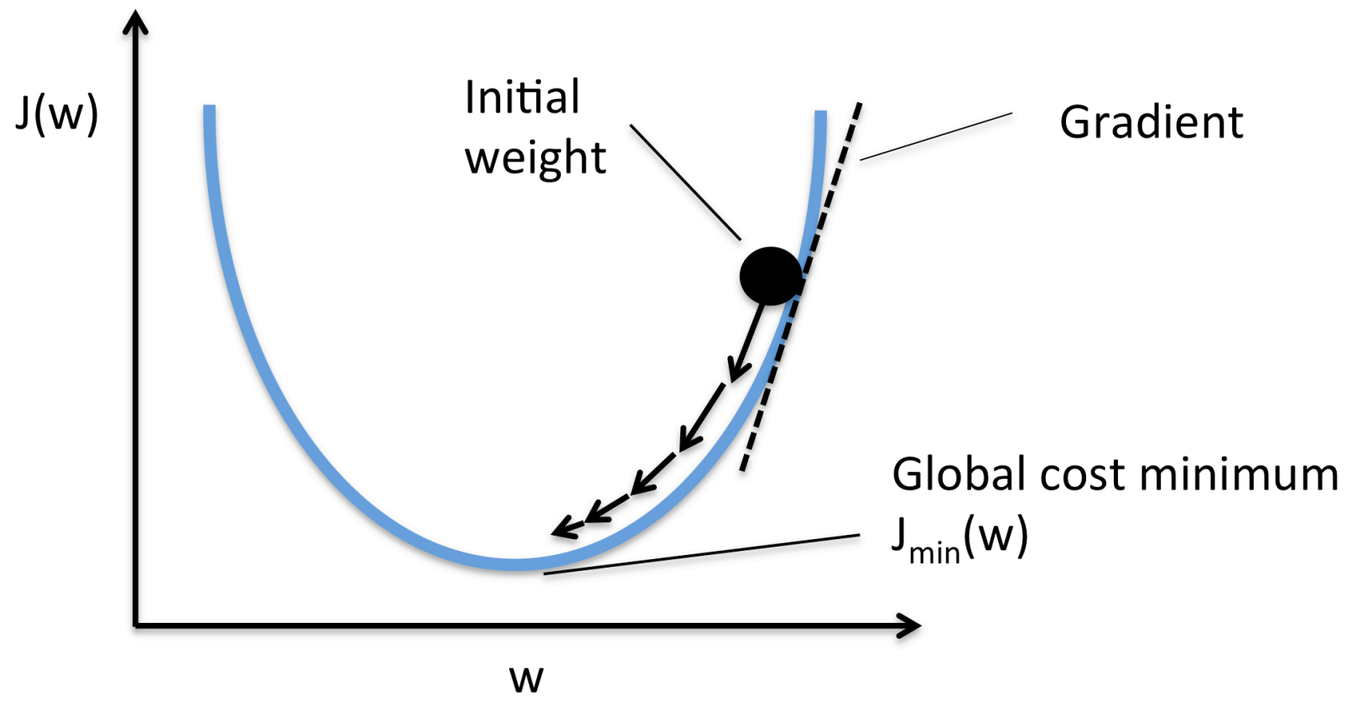
$$o(x) = \omega^T h + b$$

$$p_i \equiv p(y = 1 | x) \equiv \sigma(o(x)) = \frac{1}{1 + e^{-o(x)}}$$

$$loss_{XE} = -\sum_i y_i \ln(p_i) + (1 - y_i) \ln(1 - p_i)$$



Gradient Descent Optimization



- For a differentiable loss function f, the first Taylor expansion gives

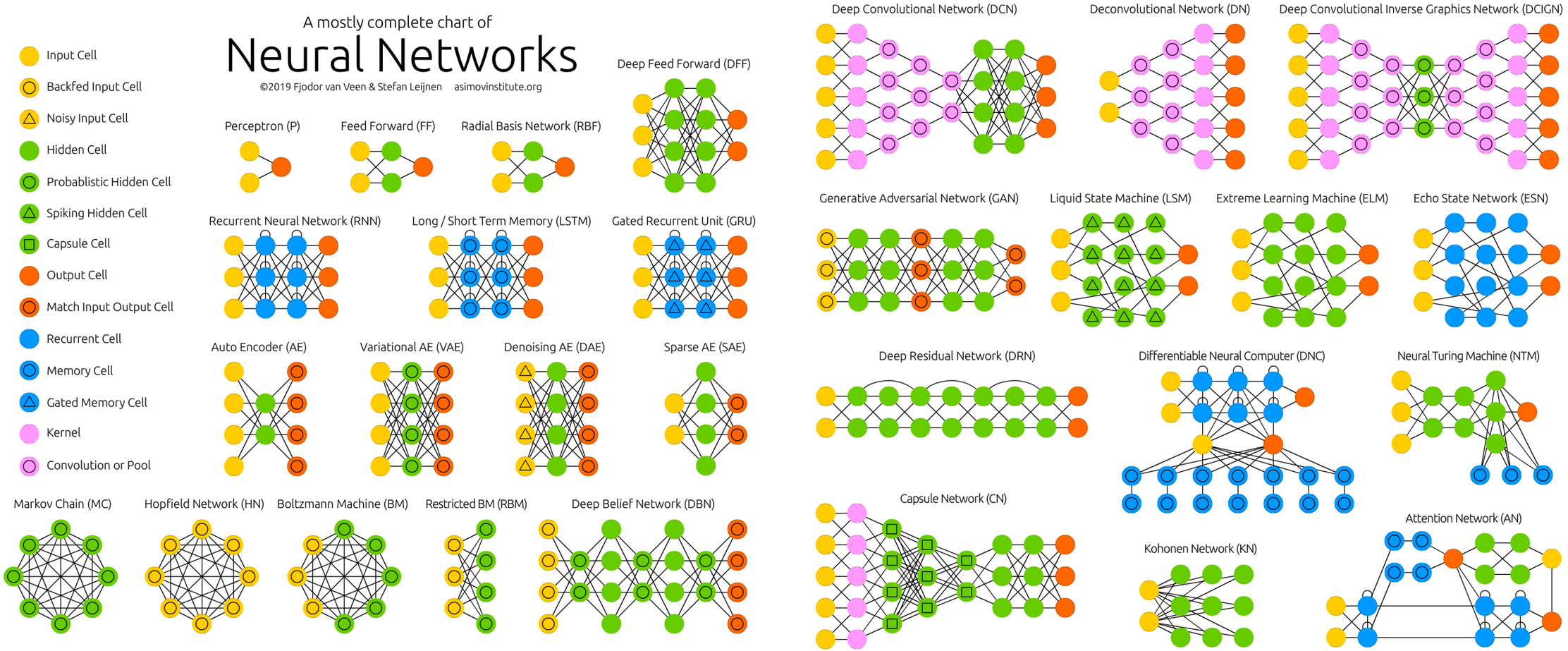
$$f(x+\varepsilon) = f(x) + \varepsilon \nabla f(x)$$

- The direction to locally maximally decrease the function value is anti-collinear to the gradient

$$\varepsilon = -\gamma \nabla f(x)$$

- Amplitude of the step γ to be taken with care to prevent overshooting

Neural Net Architectures



http://www.asimovinstitute.org/neural-network-zoo

Does not even cover it all : densenet, graph network, ...

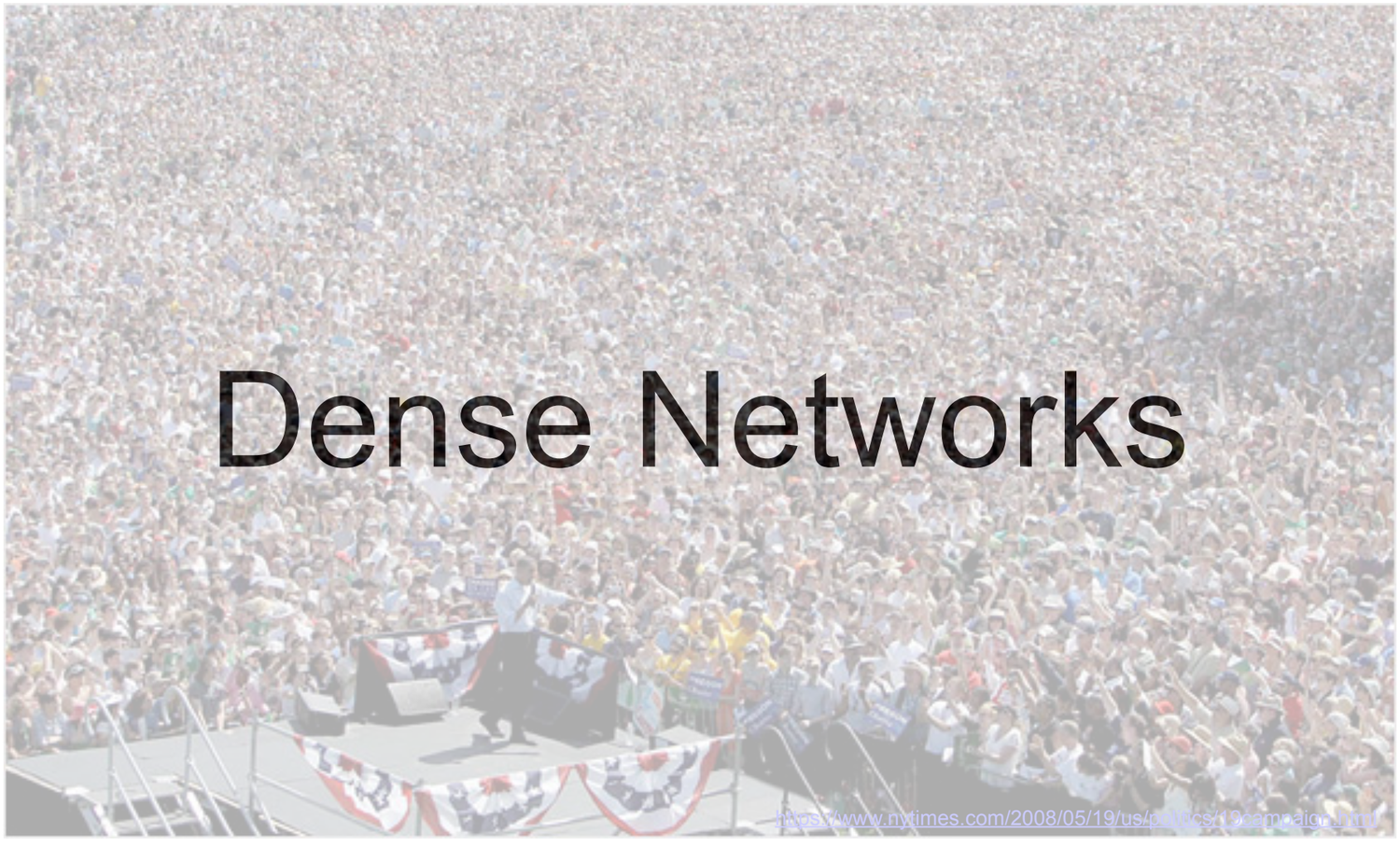
Plasticity of Neural Networks

- Models can become a complex assembly of
  - ➡Various layers of neurons
  - ➡Branches from various heterogenous inputs
  - ➡Branches to various complementary objectives
  - ➡Analytical components
  - ➡Non-analytical re-indexing
  - ➡…

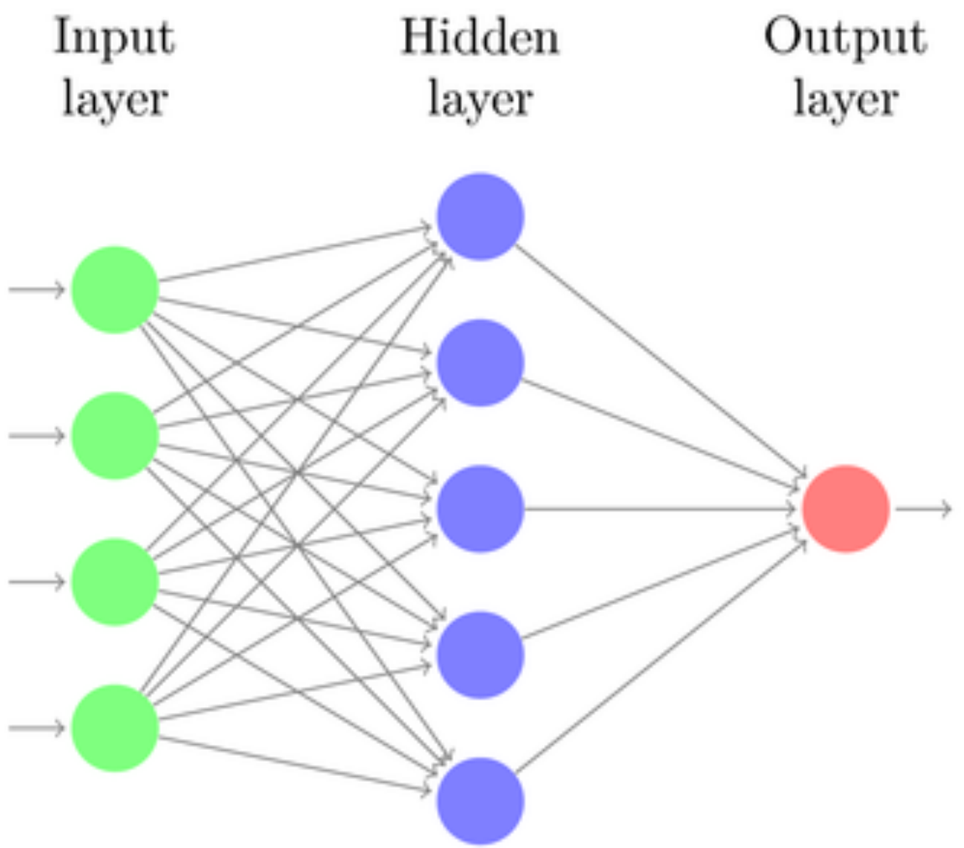
- **Not covering** the uber-structures tailored for various end-goal tasks
  - ➡Multi-input, Multi-objectives models
  - ➡Auto-encoders (AE), variational auto-encoders (VAE)
  - ➡Generative adversarial networks (GAN)
  - ➡Density estimators (DE)
  - ➡Normalizing flows (NF)
  - ➡…

- **Focusing in this lecture** on the meta-structures that can be used to compose a more complete and complex model


Dense Networks

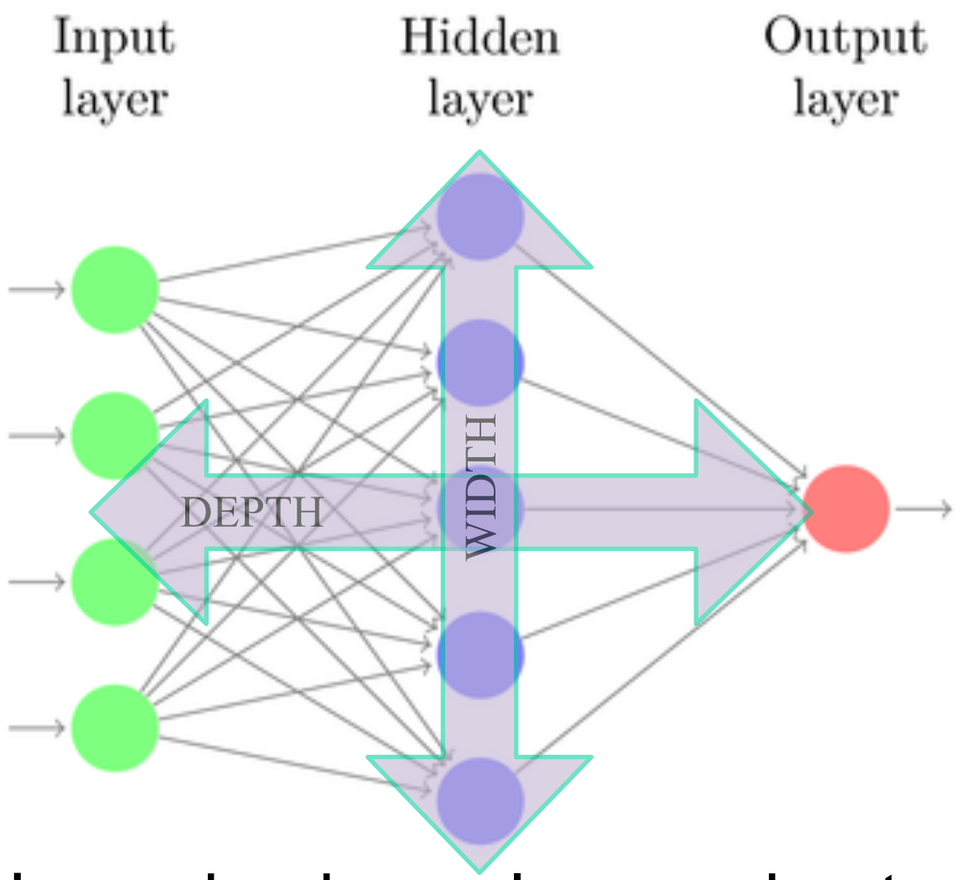


Feed Forward Networks



Simplest structure.
All-to-all connection between neurons of neighboring layers.

Going Deep



Kolmogorov's, and universal approximate theorems push towards **wide** and **deep** densely connected networks.
Depth helps with **decomposition.**
Width helps with **approximation.**

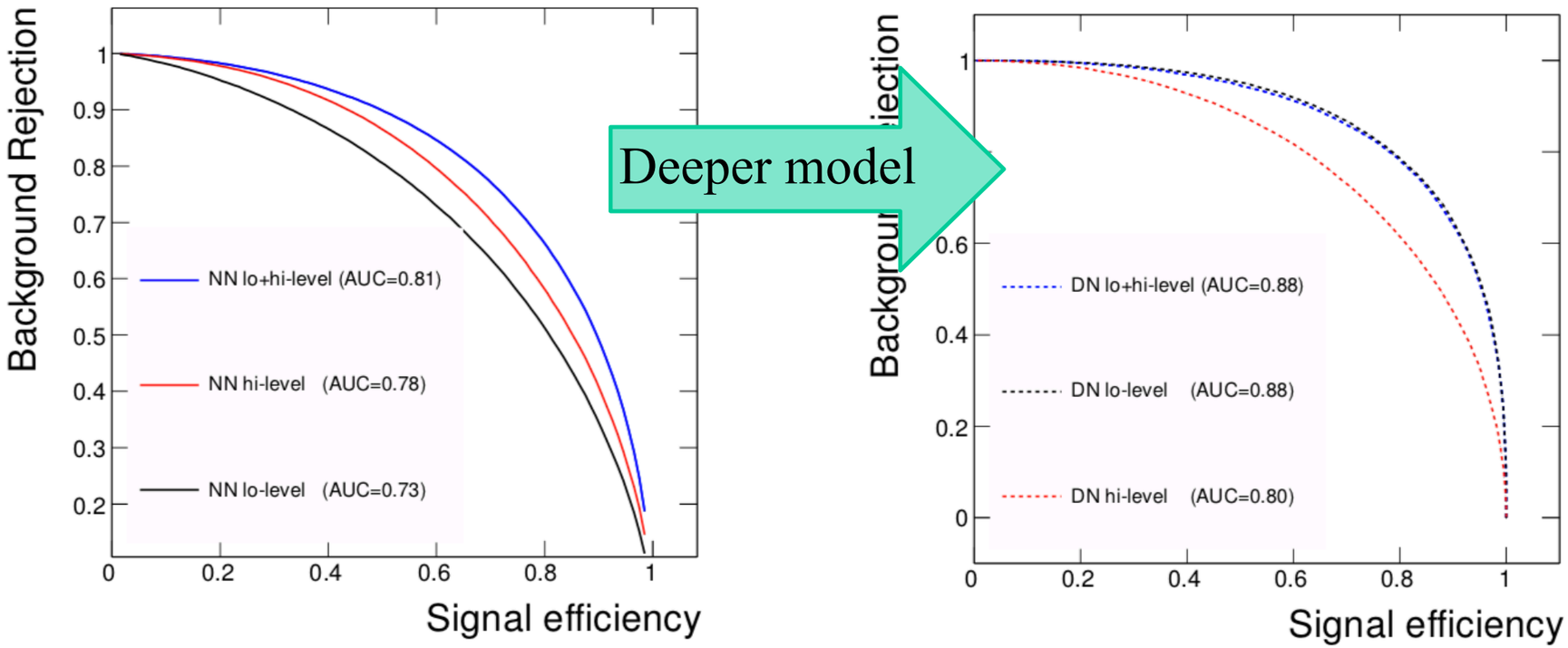
Curse of Dimensionality

- Fully connected layers require a large number of parameters

$$N^{l}_{par} = N^{l}_{input} \times N^{l}_{node} + N^{l}_{node}$$

- Lots of a capacity in this kind of models
- Convergence of models with millions/billions of parameters can be hard numerically
- Computing intensive in training and inference
- Hashing and pruning studies showed lots of redundancies : not all weights are necessary
- Weight sharing helps reducing dimensionality

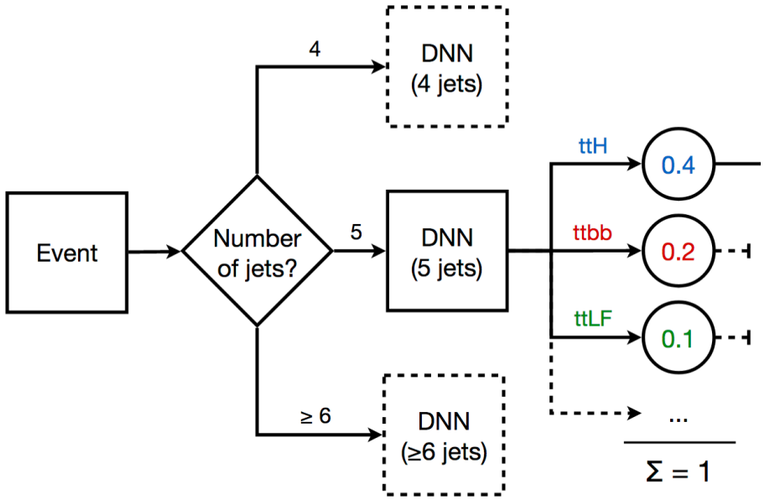

Low Level Feature Exploitation



Searching for exotic Particles in High-Energy Physics with Deep Learning [1402.4735]

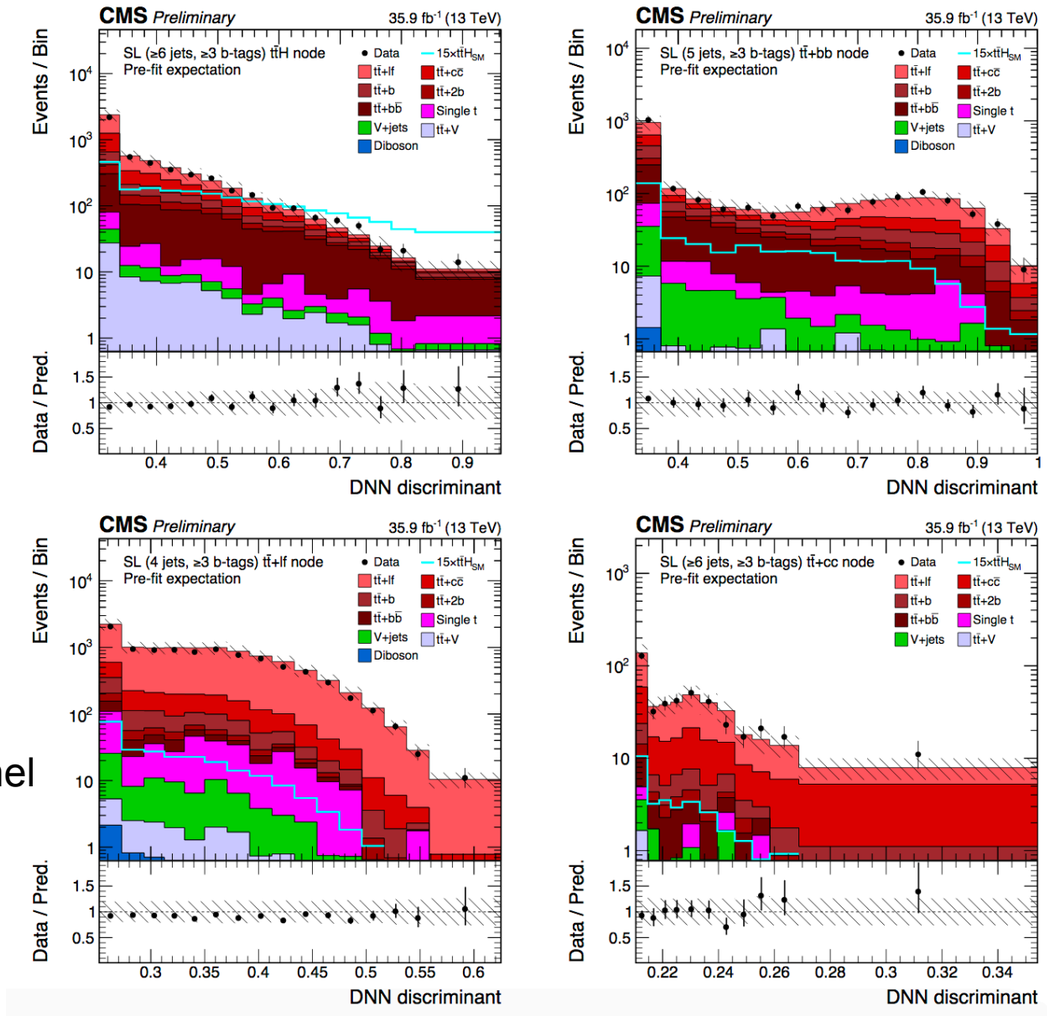
Provided enough training data is available,
deeper models act as **feature extraction** from low-level information

Multi-category Classification



Slide M. Rieger

Search for ttH production in the H-to-bb decay channel with leptonic tt decays [cds:2308267]



Regular analysis fit categories sub-divided using DNN output nodes for added sensitivity.

Take Home Message

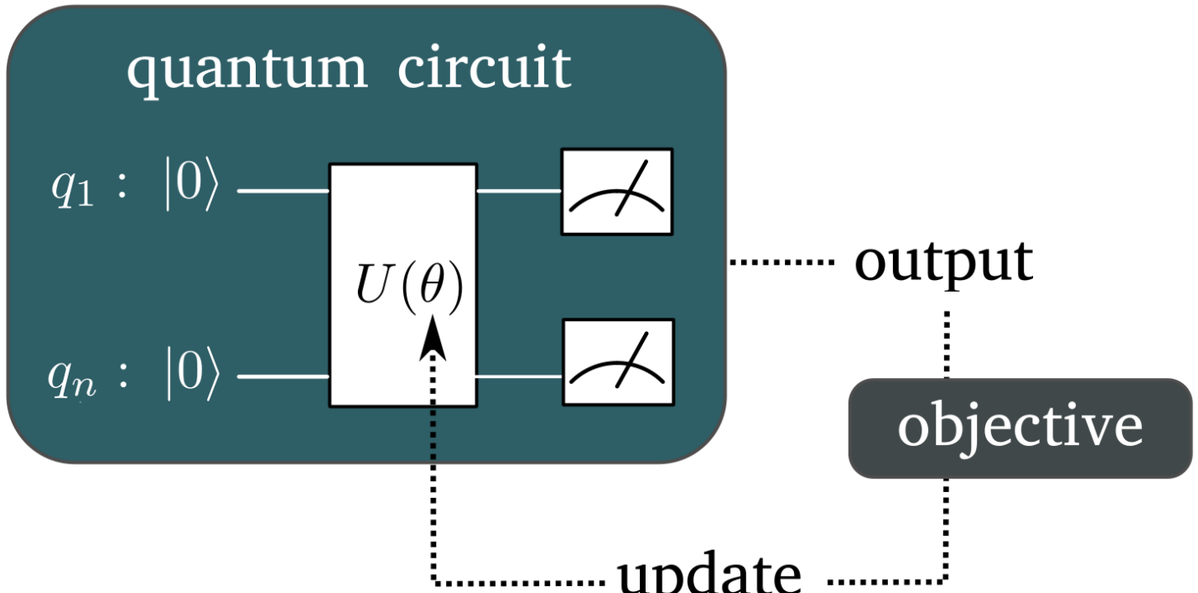
Dense models are the first ones to try out.

Can help in simple classification, from low level info.

Plagued with too many parameters.

Quantum Derivatives

https://www.ibm.com/quantum-computing/



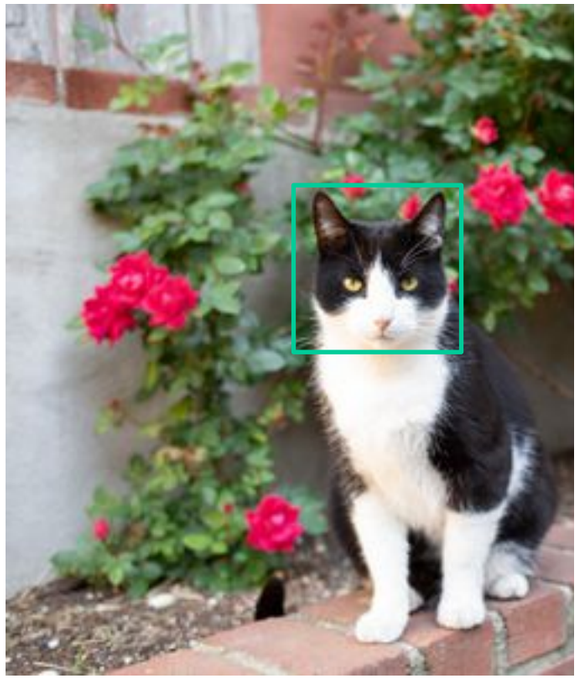
Objective based on quantum measurement.
Parameters of a quantum circuits as weights.
Trainable circuits for quantum machine learning.

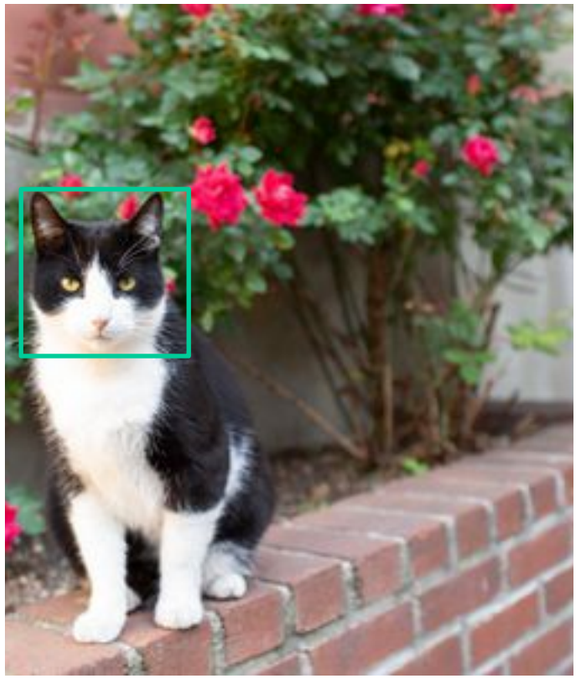
Quantum Machine Learning [1611.09347]
Quantum Machine Learning in High Energy Physics [2005.08582]
Quantum Machine Learning Models are Kernel Methods [2101.11020]

Convolutional Layers

https://news.cnrs.fr/articles/joseph-fourier-is-still-transforming-science

Translational Invariance





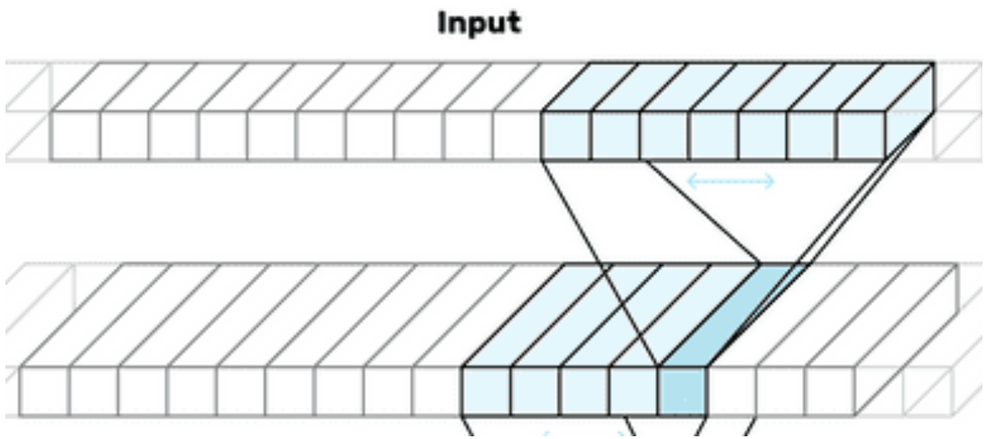
Same object can in at different place in images.
Learning of dense model would have to happen at all locations.
Tremendous overhead for model training.
Inductive bias: translation invariance.

Convolutional Layer

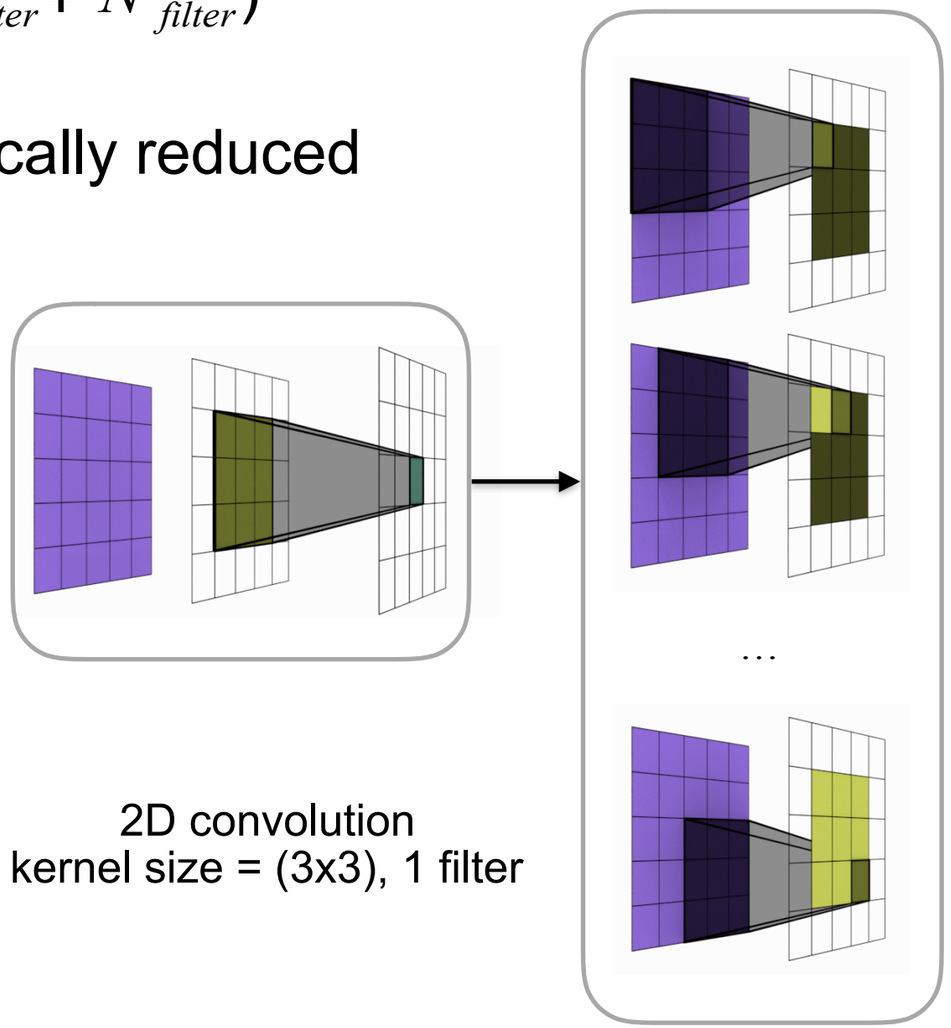
- Smaller dense network defined as filter
- Filter applied as stencil code / patch

$$N^l_{par} = (N^l_{input} / S^l_{kernel}) \times (S^l_{kernel} \times N^l_{filter} + N^l_{filter})$$

- Filter parameters are shared
- Total Number of parameters is dramatically reduced
- Available in 1D, 2D and 3D



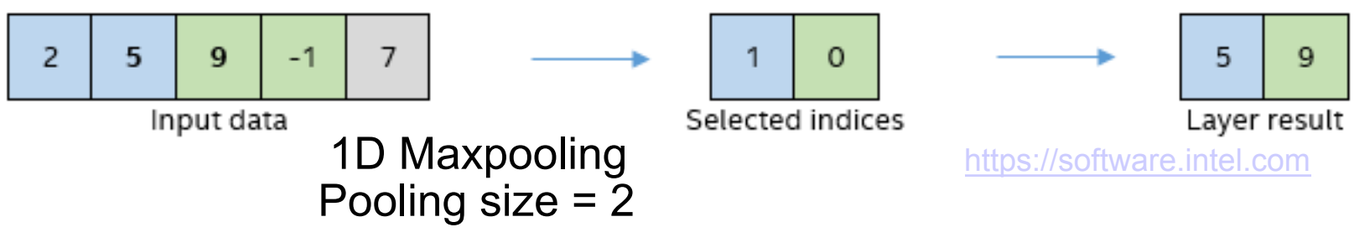
1D convolution, kernel size = 7, 1 filter



2D convolution
kernel size = (3x3), 1 filter

Pooling

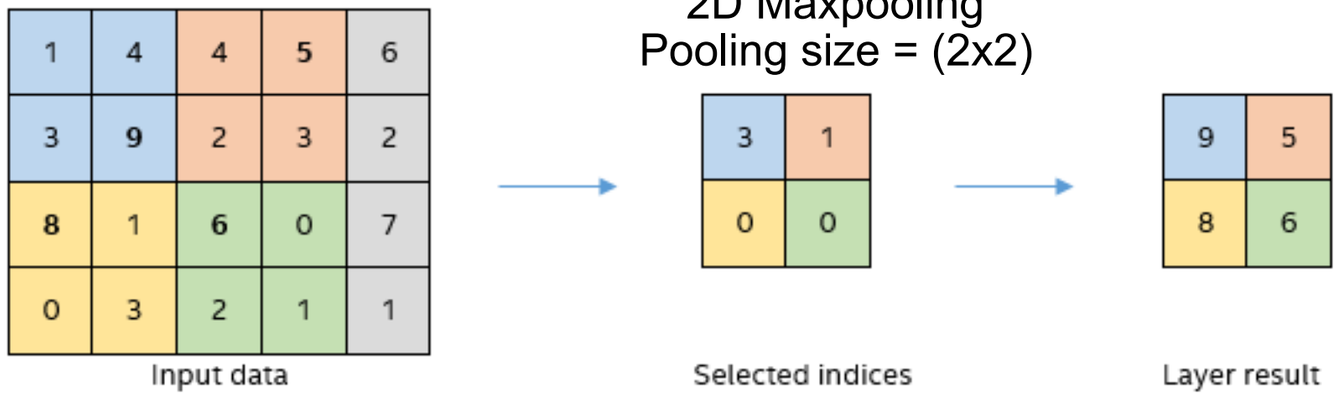
- Convolutional layer barely reduce the layer size, usually followed by a dense layer (back to large numbers of parameters)
- Not all neighboring filters will “fire” consequently
- Maxpooling is collecting the maximum activity within a region
- Non-Analytical re-indexing, but gradients can flow to train the filter



1D Maxpooling
Pooling size = 2

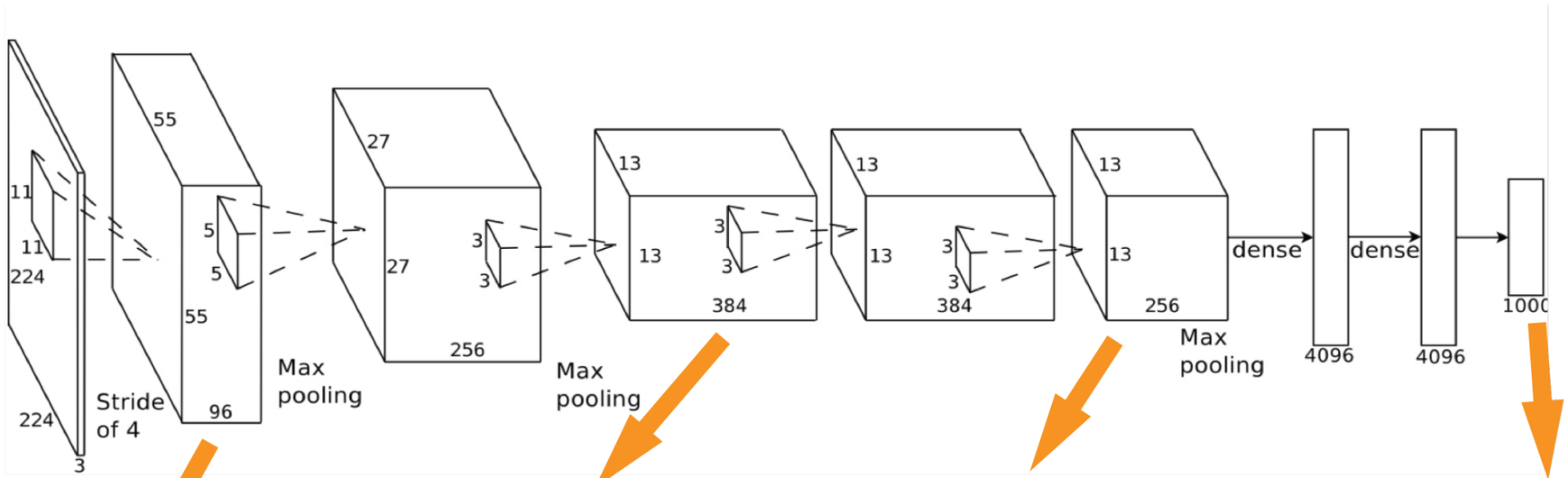
https://software.intel.com

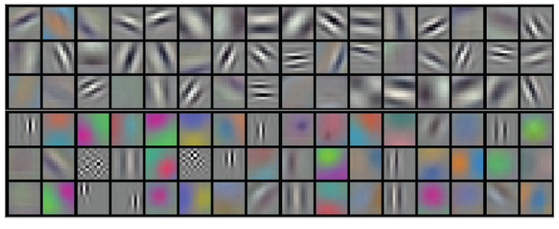
2D Maxpooling
Pooling size = (2x2)



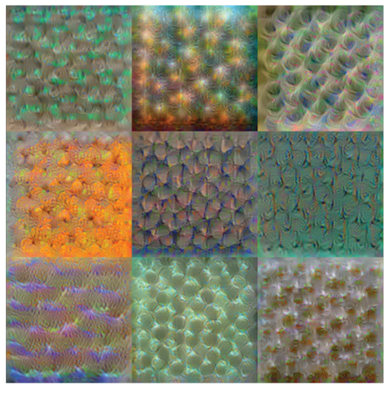
https://software.intel.com

Stacked Convolution





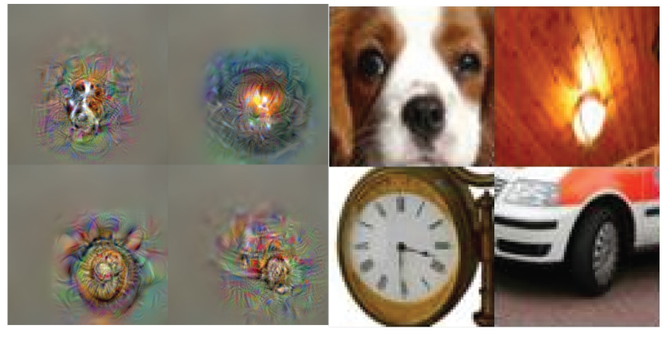
Conv 1: Edge+Blob



Conv 3: Texture

Numerical

Data-driven



Conv 5: Object Parts

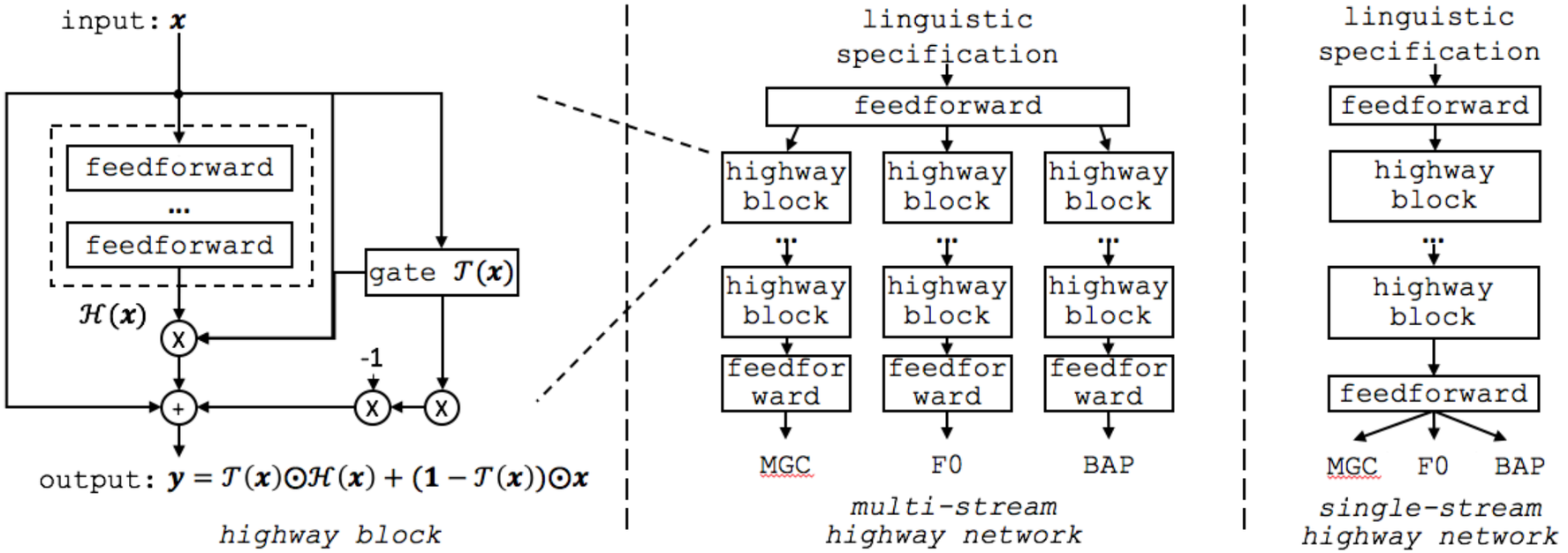
Fc8: Object Classes

Early convolution layer capture **local information.**
Late convolution layer capture **global information.**

Highway Connection

- Stacked layers distill information at consecutive scales
- Highway network controls how much information from previous layer needs to move forward as input to the next

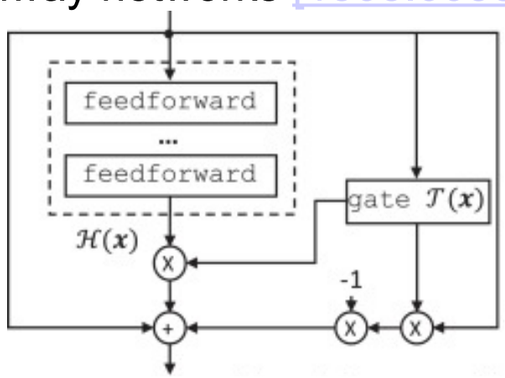
Highway networks [1505.00387]



Skip Connections

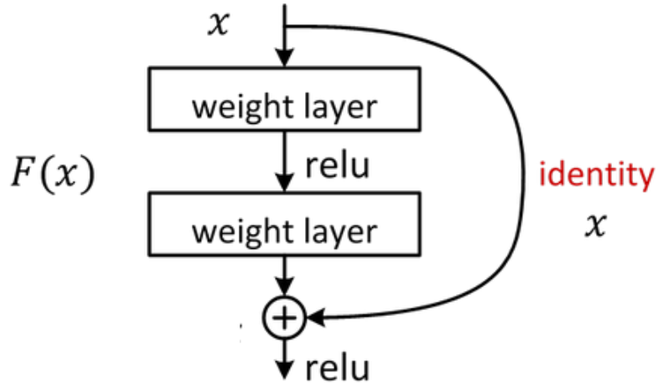
- Stacked convolution layers distill information at consecutive scales
- Several **ways of conserving the initial information** from input

Highway networks [1505.00387]



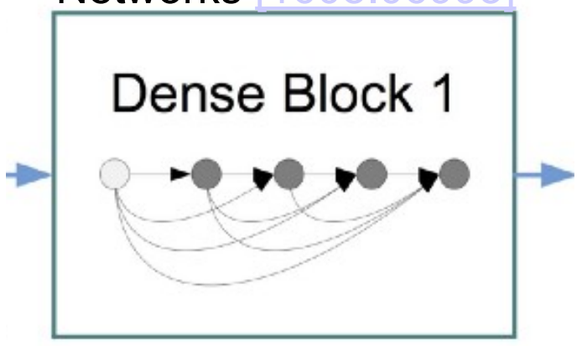
$$f_l(i_l) \rightarrow f_l(i_l) \cdot T(i_l) + i_l \cdot (1 - T(i_l))$$

Deep Residual Learning for Image Recognition [1512.03385]



$$f_l(i_l) \rightarrow f_l(i_l) + i_l$$

Densely Connected Convolution Networks [1608.06993]

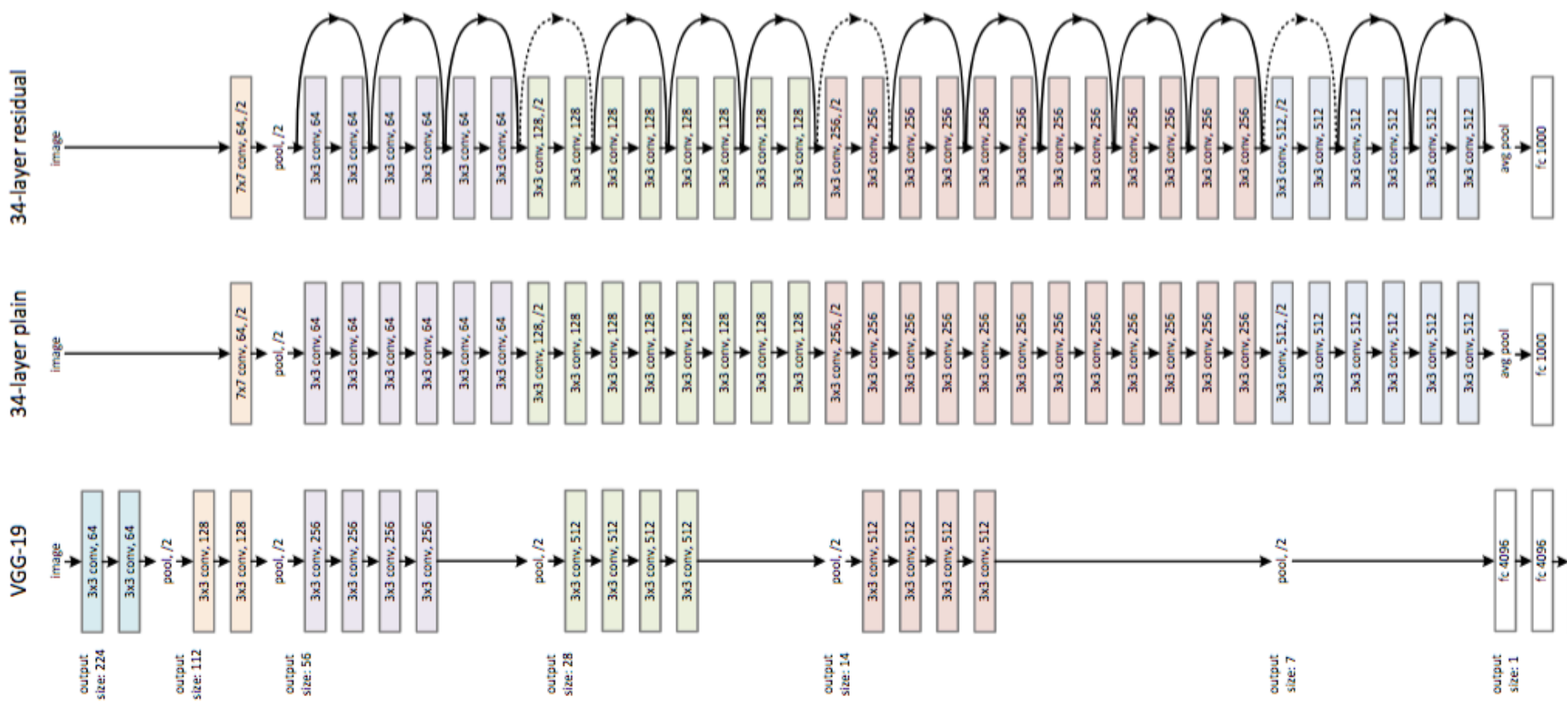


$$f_l(i_l) \rightarrow f_l(\bigcup_{k \leq l}(i_k))$$

Residual Connection

- Stacked layers distill information at consecutive scales
- Residual connection carries the input other to the output, dimensionality allowing

$$f_l(i_l) \rightarrow f_l(i_l) + i_l$$

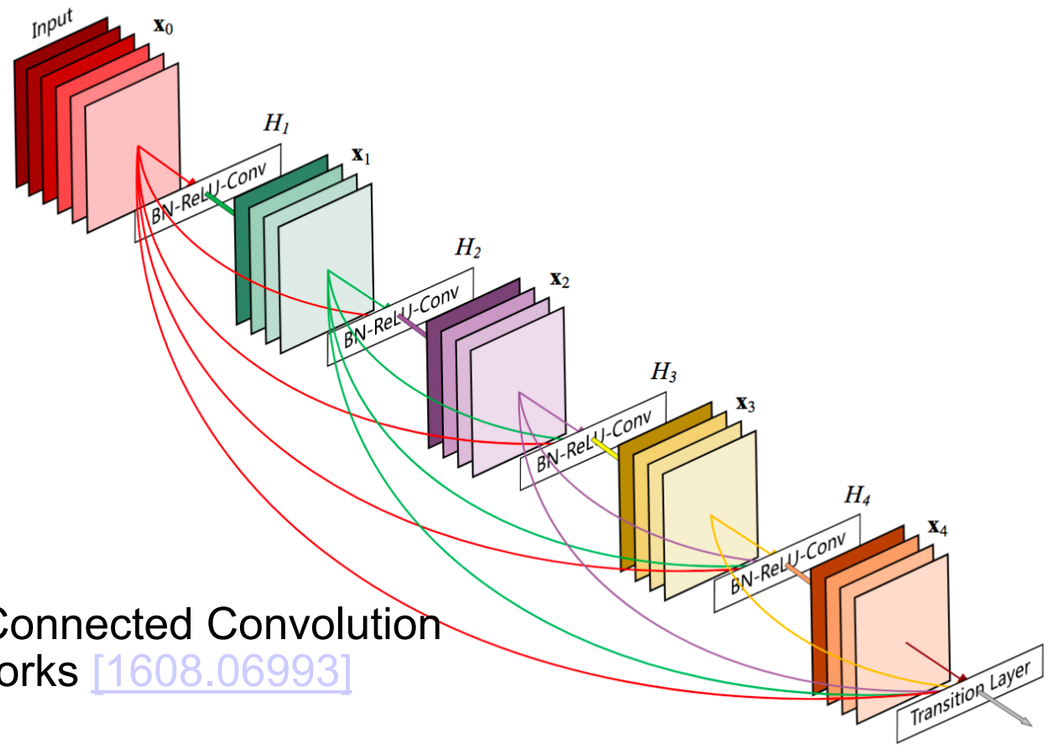


Deep Residual Learning for Image Recognition [1512.03385]

Dense Connection

- Stacked layers distill information at consecutive scales
- dense-net provides the concatenation of all previous layer input to the next layer

$$f_l(i_l) \rightarrow f_l(\bigcup_{k \leq l}(i_k))$$

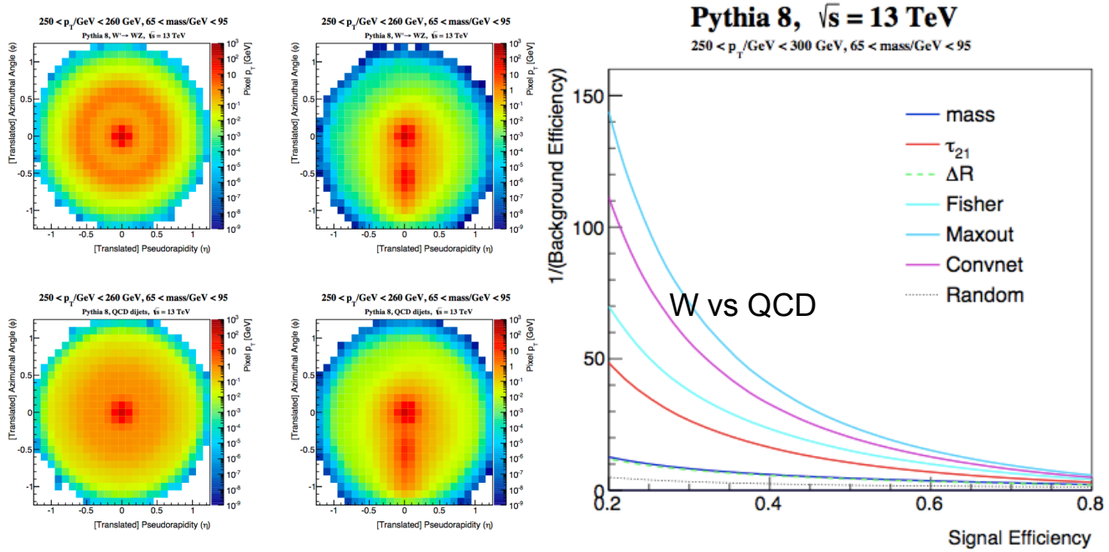


Densely Connected Convolution Networks [1608.06993]

Image Representation

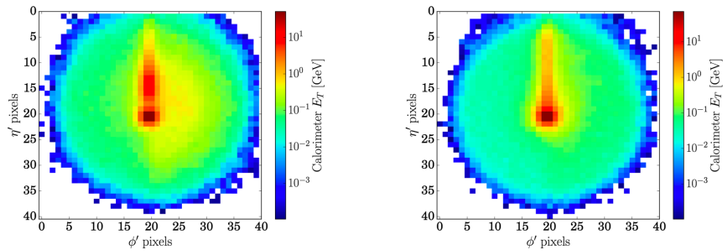
Jet-Images – Deep learning edition

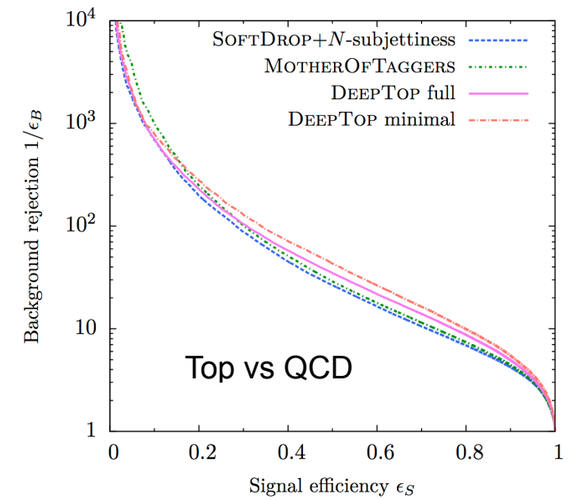
[1511.05190]



Deep-learning top taggers or the end of QCD?

[1701.08784]





Calorimeter signal are image-like.
Projection of reconstructed particle properties onto images possible.
Potential loss of information during projection.

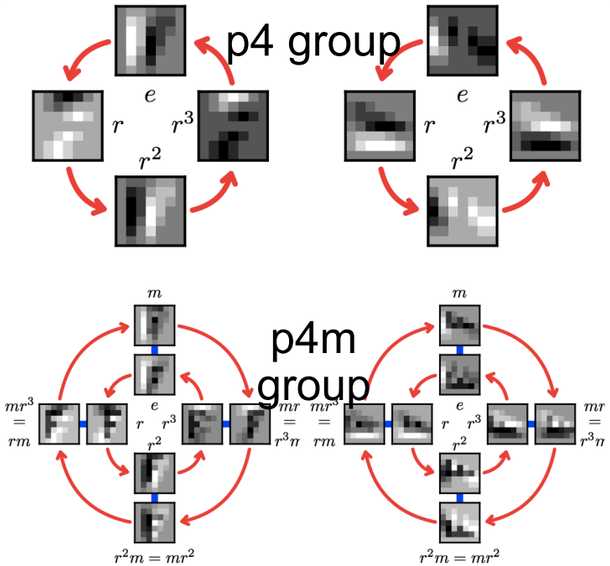
Take Home Message

The Swiss-knife of image processing.

Translation invariance only.

Other invariance with model variants.

Equivariant Derivatives

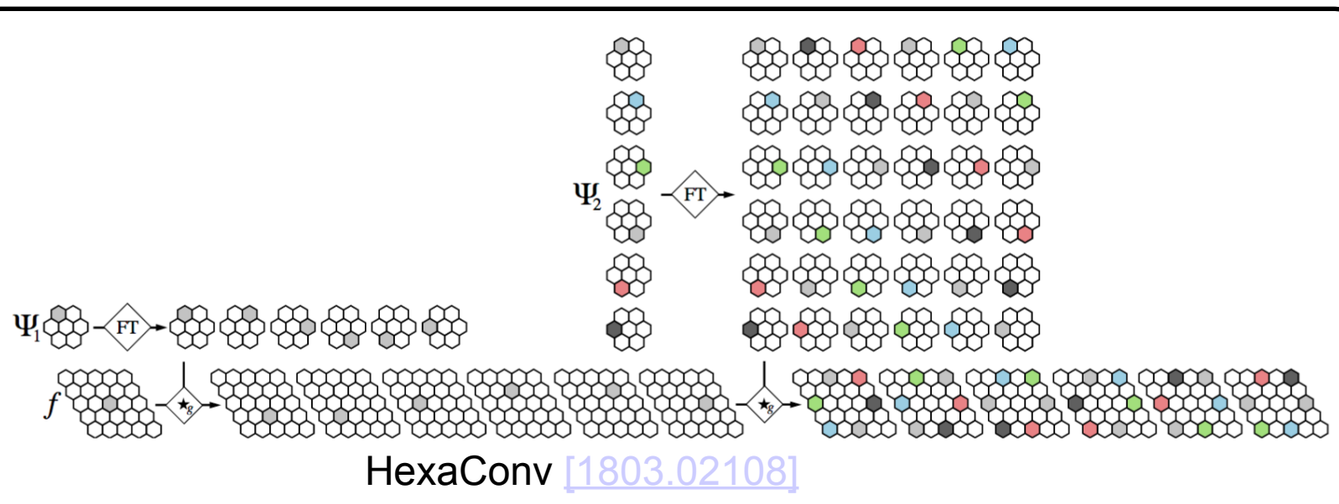


Group Equivariant Convolution Networks [1602.07576]

$$k_{\mu,i} \xrightarrow{\text{CoLa}} \tilde{k}_{\mu,j} = k_{\mu,i} \begin{pmatrix} 1 & 0 & \cdots & 0 & C_{1,N+2} & \cdots & C_{1,M} \\ 0 & 1 & & \vdots & C_{2,N+2} & \cdots & C_{2,M} \\ \vdots & \vdots & \ddots & 0 & \vdots & & \vdots \\ 0 & 0 & \cdots & 1 & C_{N,N+2} & \cdots & C_{N,M} \end{pmatrix}$$

$$\tilde{k}_j \xrightarrow{\text{LoLa}} \hat{k}_j = \begin{pmatrix} m^2(\tilde{k}_j) \\ p_T(\tilde{k}_j) \\ w_{jm}^{(E)}\, E(\tilde{k}_m) \\ w_{jm}^{(d)}\, d_{jm}^2 \end{pmatrix}$$

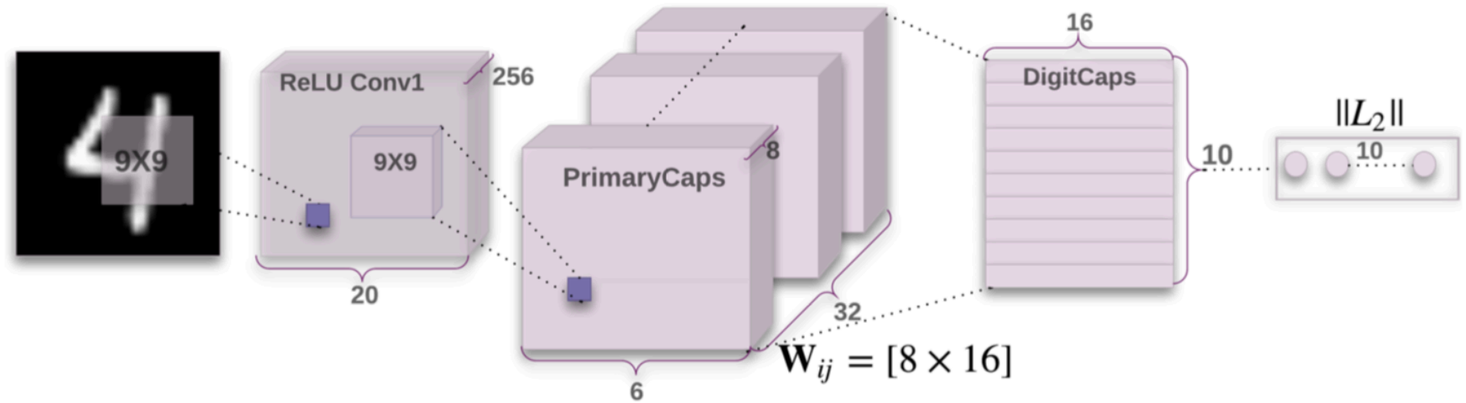
Deep-learned Top Tagging with Lorentz Layer [1707.08966]



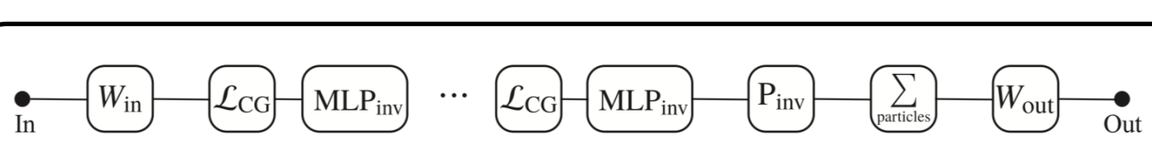
HexaConv [1803.02108]

- Training with further knowledge of invariance brings improvements
- Including domain knowledge on how object transform brings improvements

Equivariant Derivatives

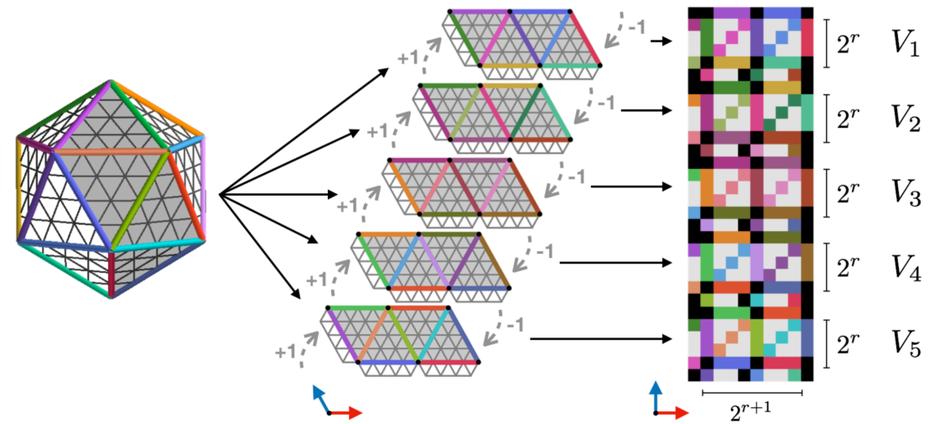


Dynamic Routing Between Capsules [1710.09829]



$$\mathscr{F}_i \mapsto W \cdot \left(\mathscr{F}_i \oplus \mathscr{F}_i^{\otimes 2} \oplus \sum_j f\left(p_{ij}^2\right) \cdot p_{ij} \otimes \mathscr{F}_j \right)$$

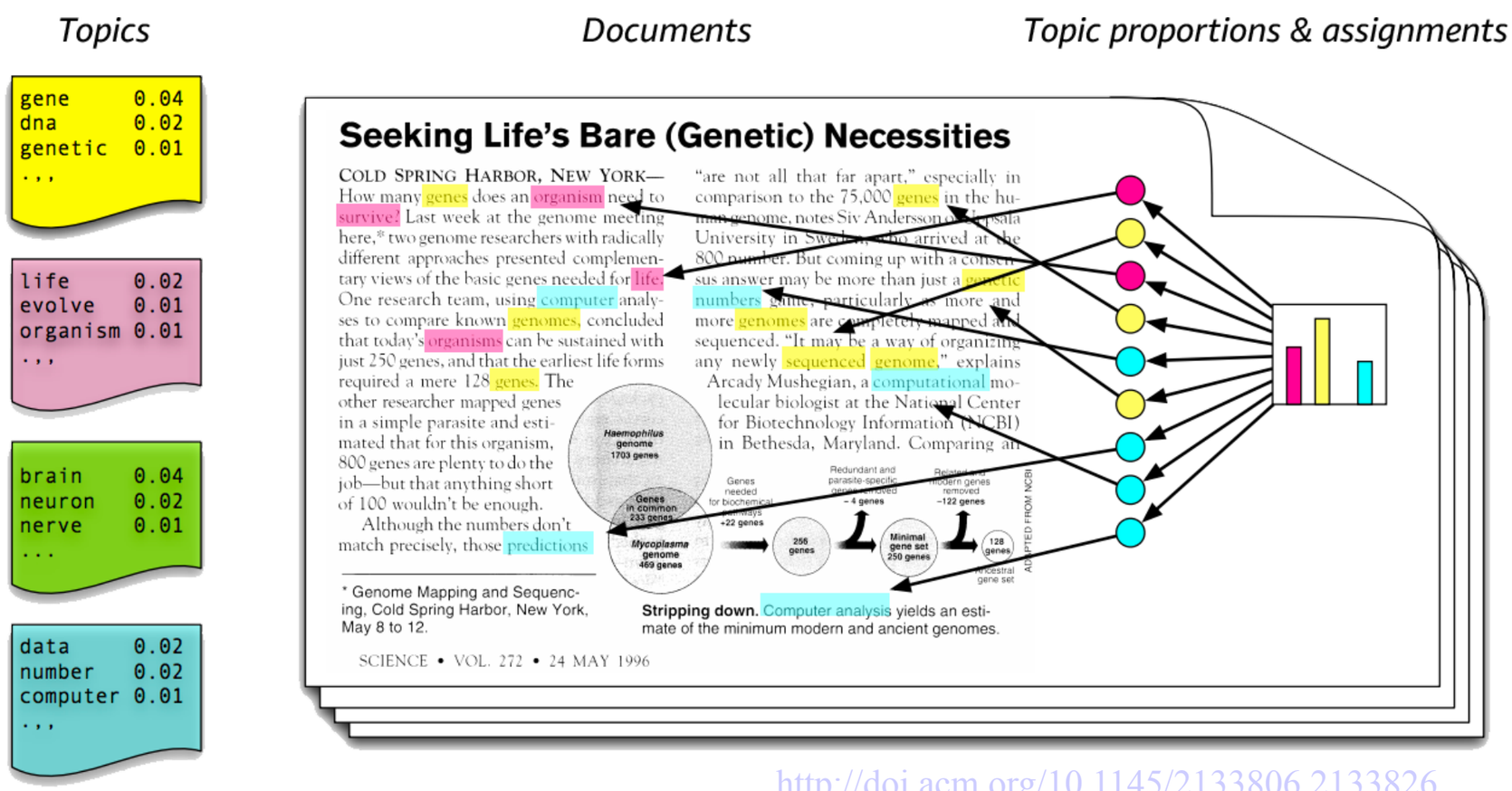
Lorentz group Equivariant networks [2006.04780]



Gauge Equivariant Convolutional Networks and the Icosahedral CNN [1902.04615]


Recurrent Cells
https://en.wikipedia.org/wiki/Halley's_Comet

Natural Language Processing

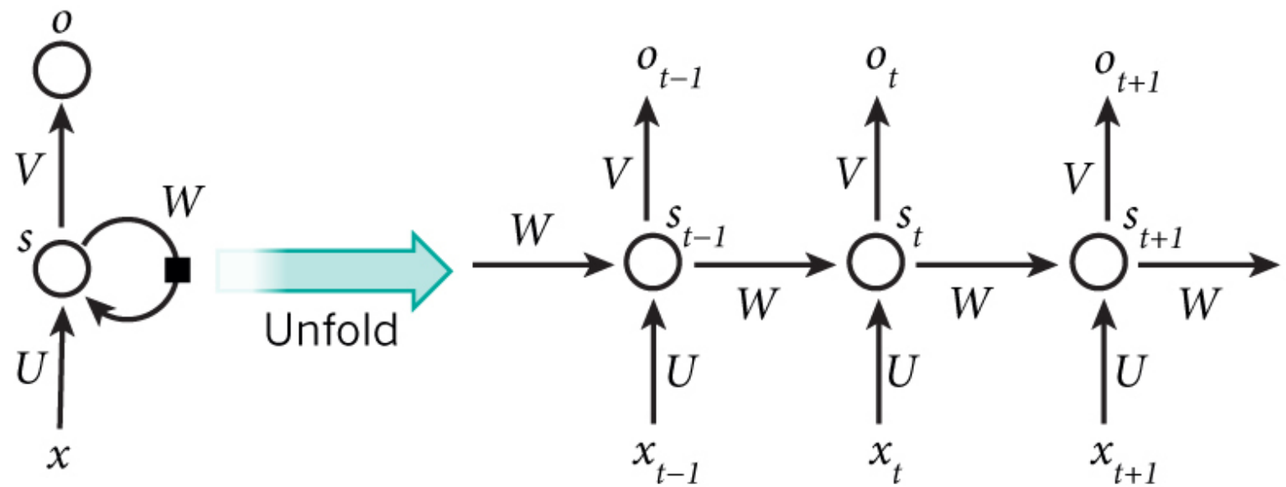


http://doi.acm.org/10.1145/2133806.2133826

- Words have a meaning of their own
- Order of the words contains additional information
- Natural language represented as an ordered sequence of variable size

Recurrent Neural Network

- Sequential (text) of temporal (voice) data contains information in their structure
- Model that can naturally accommodate for **variable sized input**
- Characterized by an hidden state carried over steps

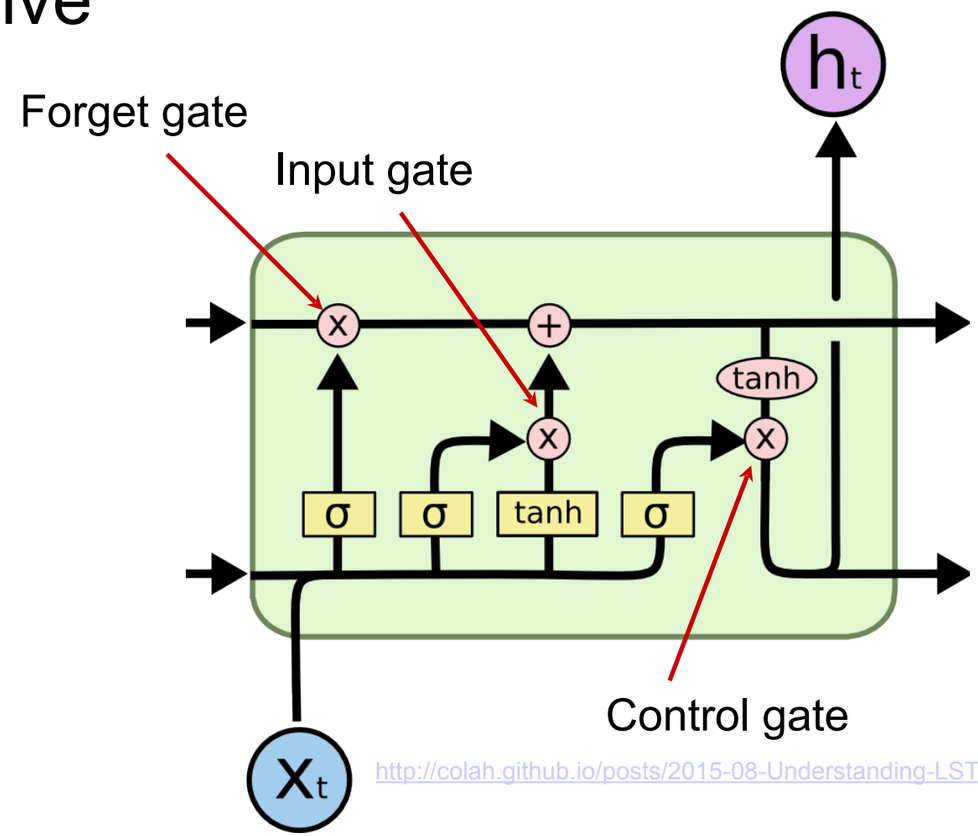


$$s_t = \tanh(U\, x_t + W\, s_{t-1} + b_r)$$
$$o_t = \sigma(V\, s_t + b_o)$$

Long Short Term Memory Cell

- LSTM revolutionized text processing in the late 90s
- Carries around a cell state (C_t) and hidden state (h_t)
- Computationally expensive

$$c_t = [h_{t-1}, x_t]$$
$$f_t = \sigma(W_f c_t + b_f)$$
$$i_t = \sigma(W_i c_t + b_i)$$
$$\tilde{C} = \tanh(W_C c_t + b_C)$$
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$
$$o_t = \sigma(W_o c_t + b_o)$$
$$h_t = o_t * \tanh(C_t)$$



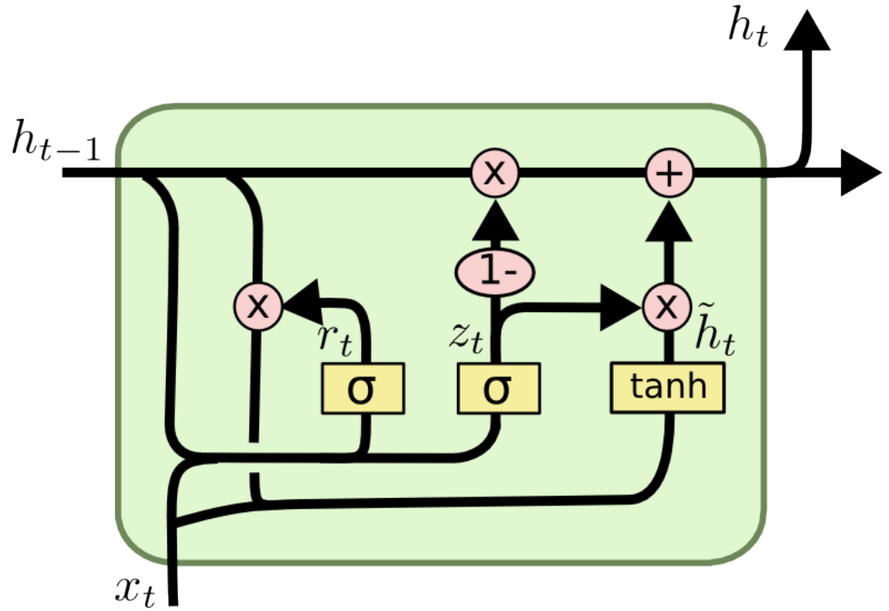
http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Long Short-Term Memory
[doi:10.1162/neco.1997.9.8.1735]

Gated Recurrent Unit

- GRU simplifies the computation from LSTM
- Only hidden state

$$c_t = [h_{t-1}, x_t]$$
$$z_t = \sigma(W_z c_t + b_z)$$
$$r_t = \sigma(W_r c_t + b_r)$$
$$\tilde{h}_t = \tanh(W_h [r_t * h_{t-1}, x_t] + b_h)$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

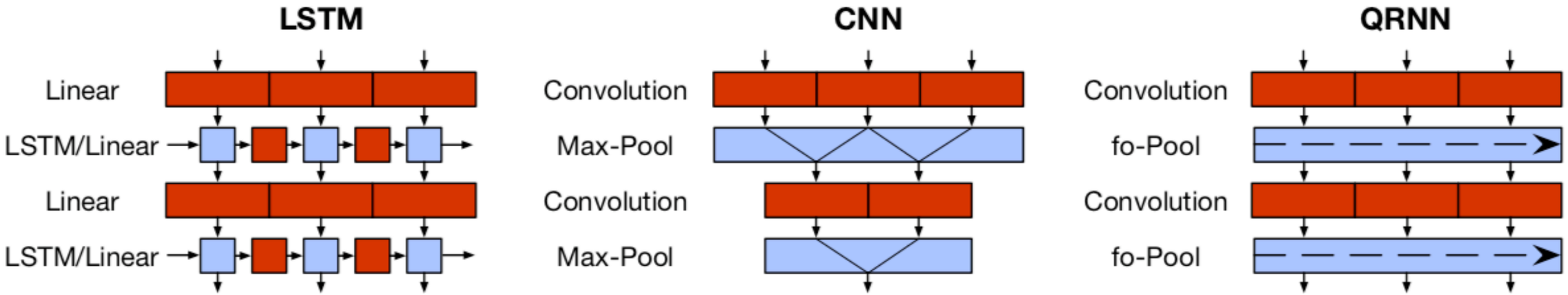


http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Learning Phrase Representations using RNN Encode-Decoder for Statistical Machine Translartion [1406.1078]

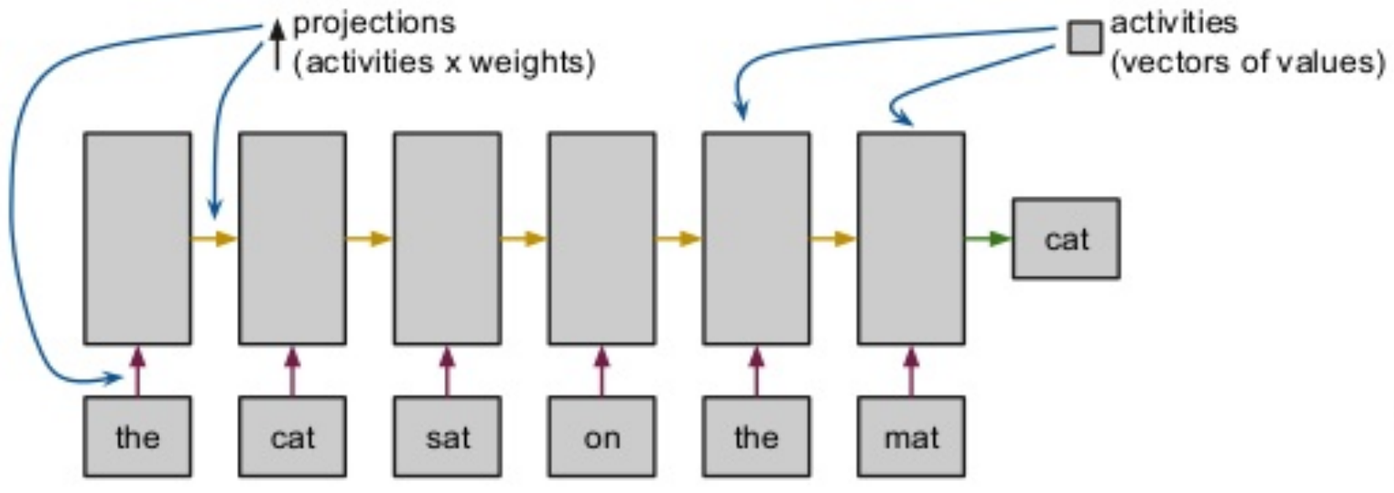
Quasi-Recurrent Models

- QRNN takes advantages from CNN and RNN
- More efficient, accurate on long sequences.

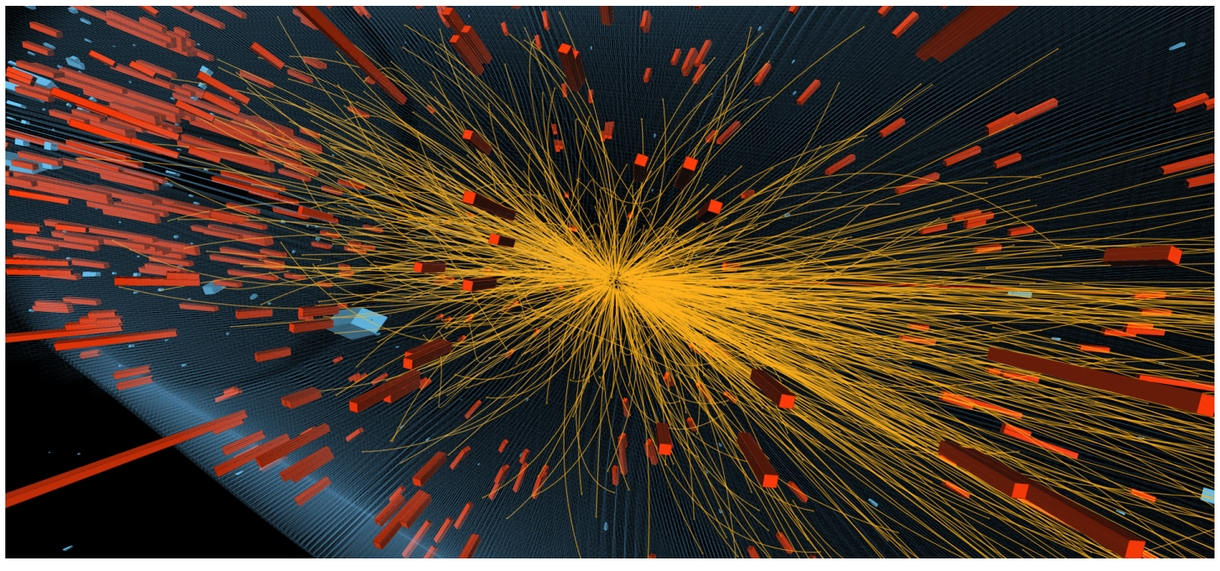


Quasi-Recurrent Neural Networks [1611.01576]

Challenge in Natural Ordering



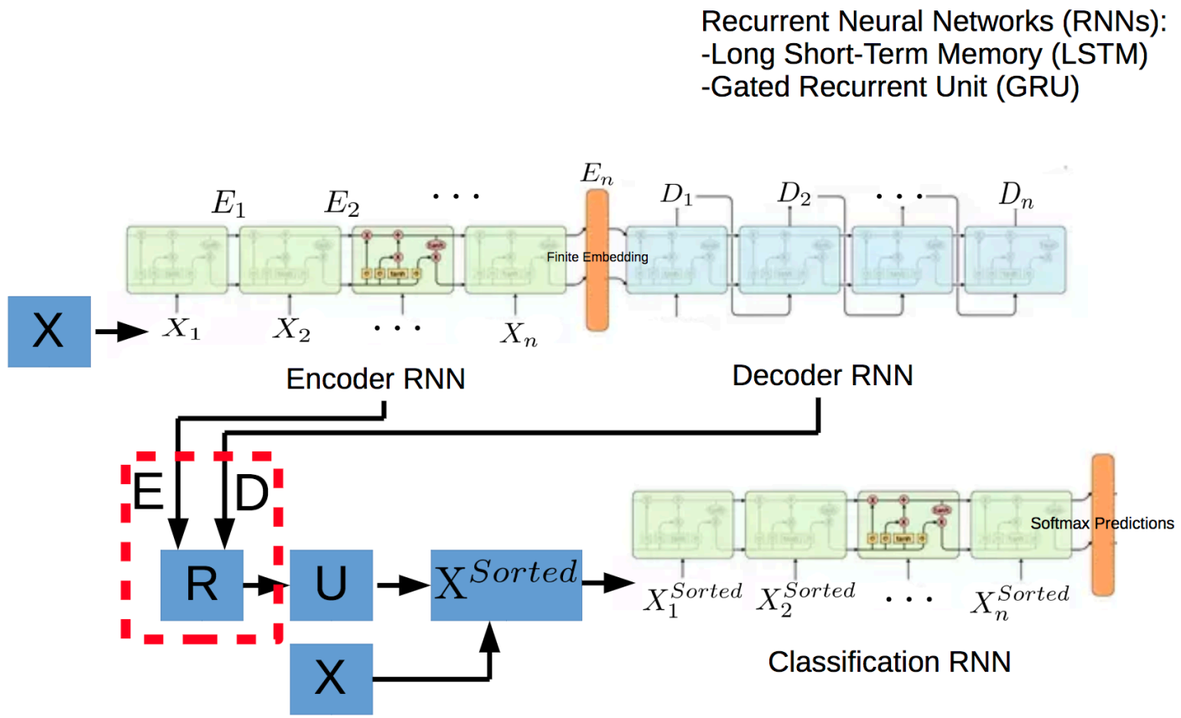
Text have natural order. RNN/LSTM can correlate the information to internal representation



There is underlying order in collision events. Smeared through timing resolution. No natural order in observable

- **Learn how to sort**

Learn How To Sort



- Custom:

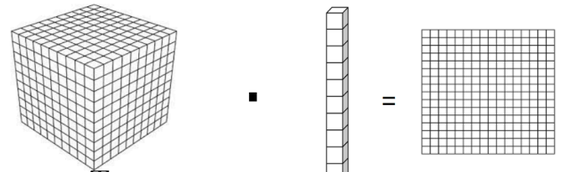
$$R = W1 \begin{bmatrix} — & E_1 & — \\ \vdots & \vdots & \vdots \\ — & E_n & — \end{bmatrix} + W2 \begin{bmatrix} | & \cdots & | \\ D_1 & \cdots & D_n \\ | & \cdots & | \end{bmatrix}$$

Where W_1, W_2 are trainable (nxn) matricies

- Pointer Networks https://arxiv.org/abs/1506.03134

$$\mathrm{R}_j^i = v^T tanh(W_1 e_j + W_2 d_i) \quad j \in (1, ..., n)$$

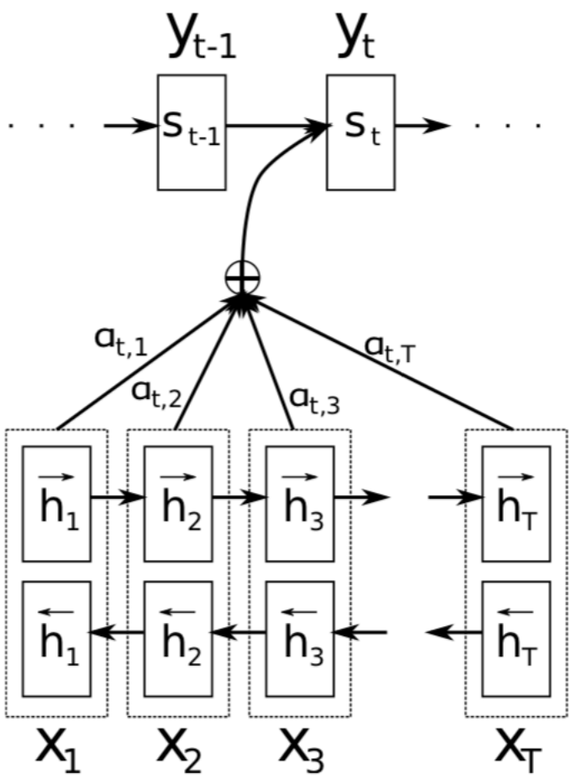
Memory Intensive!



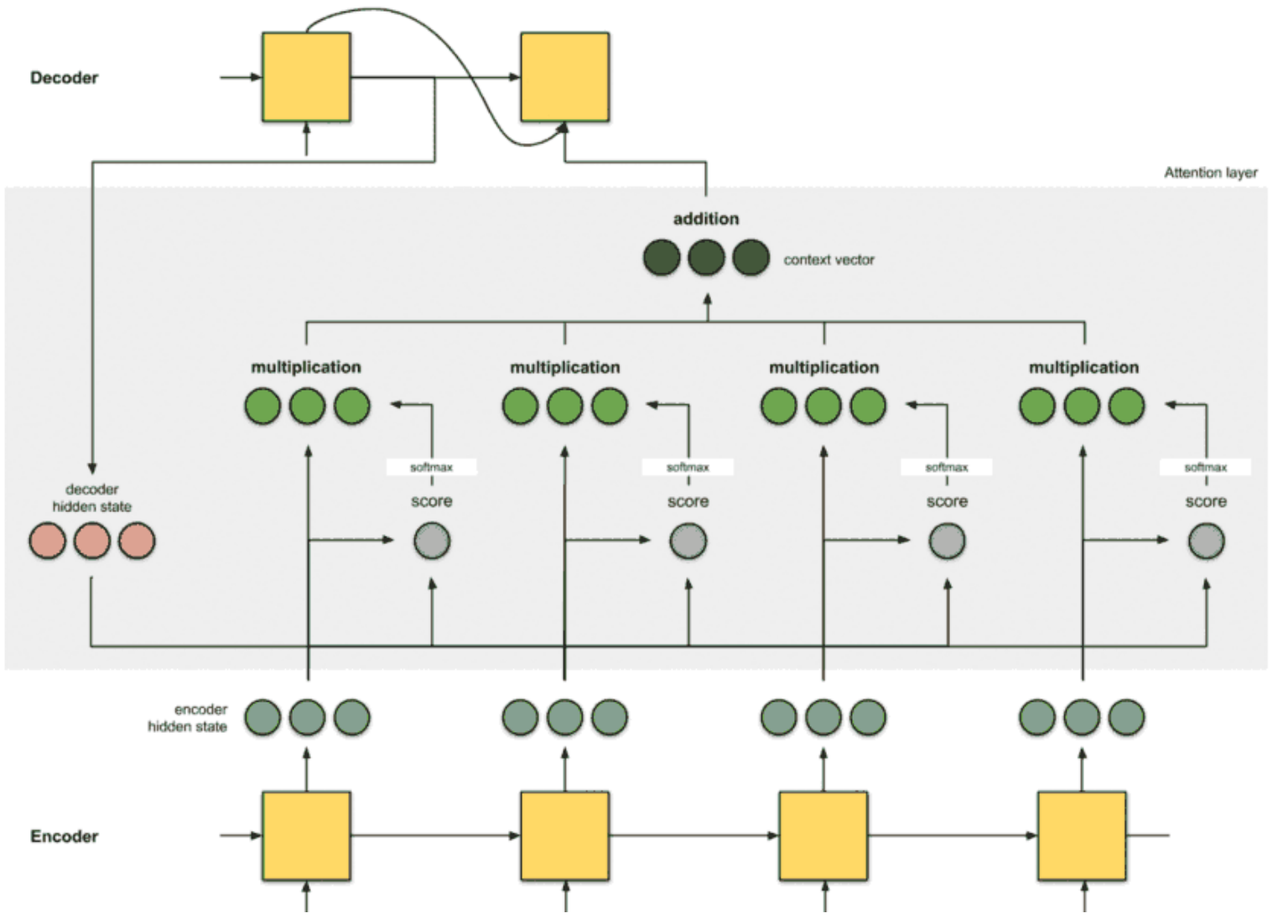
Sorting and “soft” sorting models can be concurrently trained with recurrent networks. Expensive and tricky to train.

Attention Mechanism

First introduced in natural language translation.
Provides contextual information where only local information is available.
Concept has been derived to other architectures.

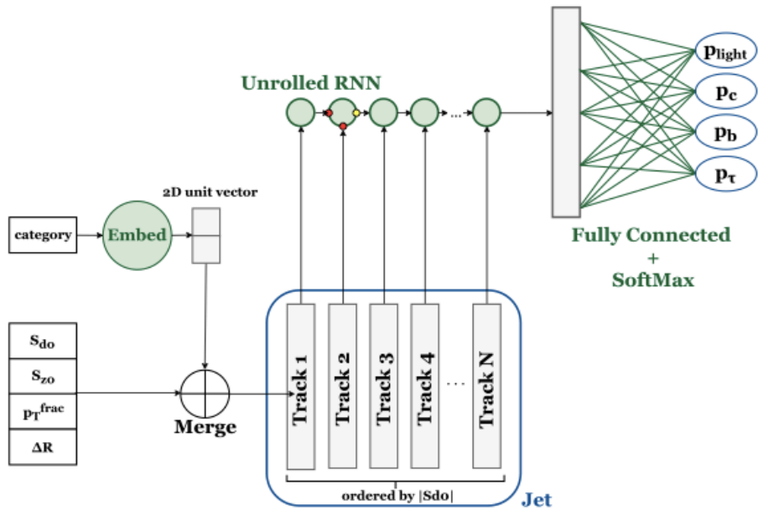


Neural Machine Translation by Jointly Learning to Align and Translate [1409.0473]

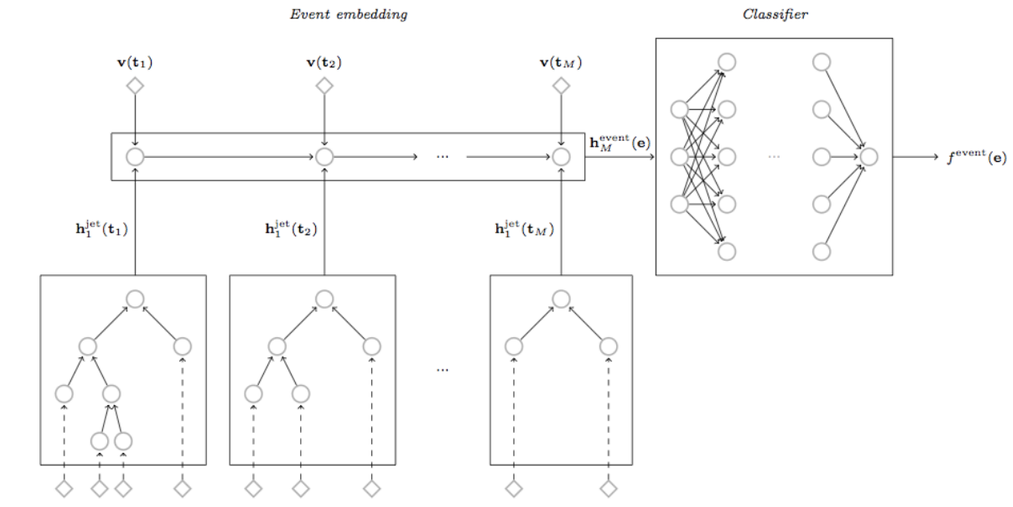


https://towardsdatascience.com/attn-illustrated-attention-5ec4ad276ee3

Sequence Representation



B-Jet with Recurrent Neural Networks
[cds:2255226]



QCD-Aware Recursive Neural Networks for Jet Physics.
[1702.00748]

Somehow arbitrary choice on ordering with sequence representation.
Physics-inspired ordering as inductive bias.
Ordering can be learned too somehow.

Take Home Message

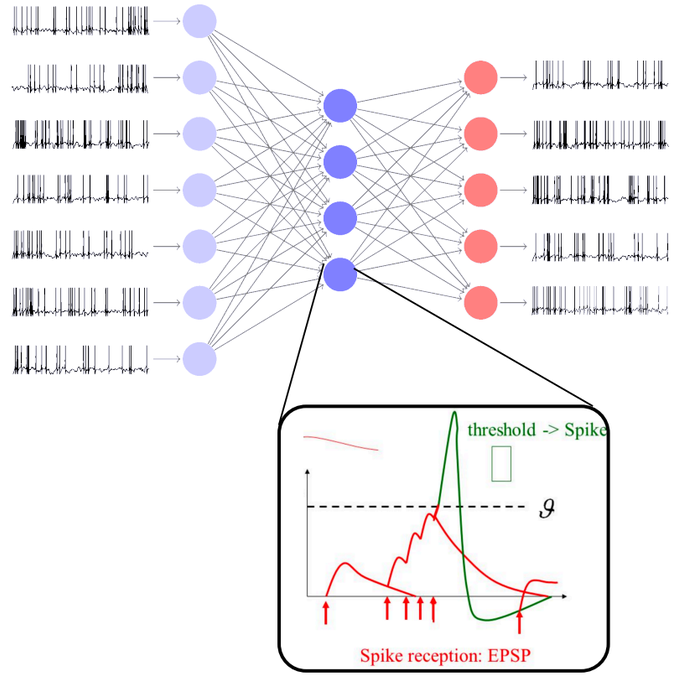
The Swiss-knife of natural language processing.

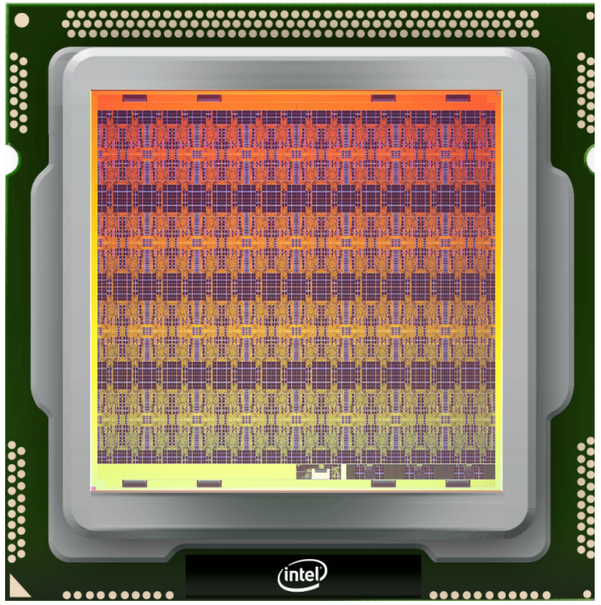
Good with variable size input.

Ordering might be an issue.


Spiking Derivatives

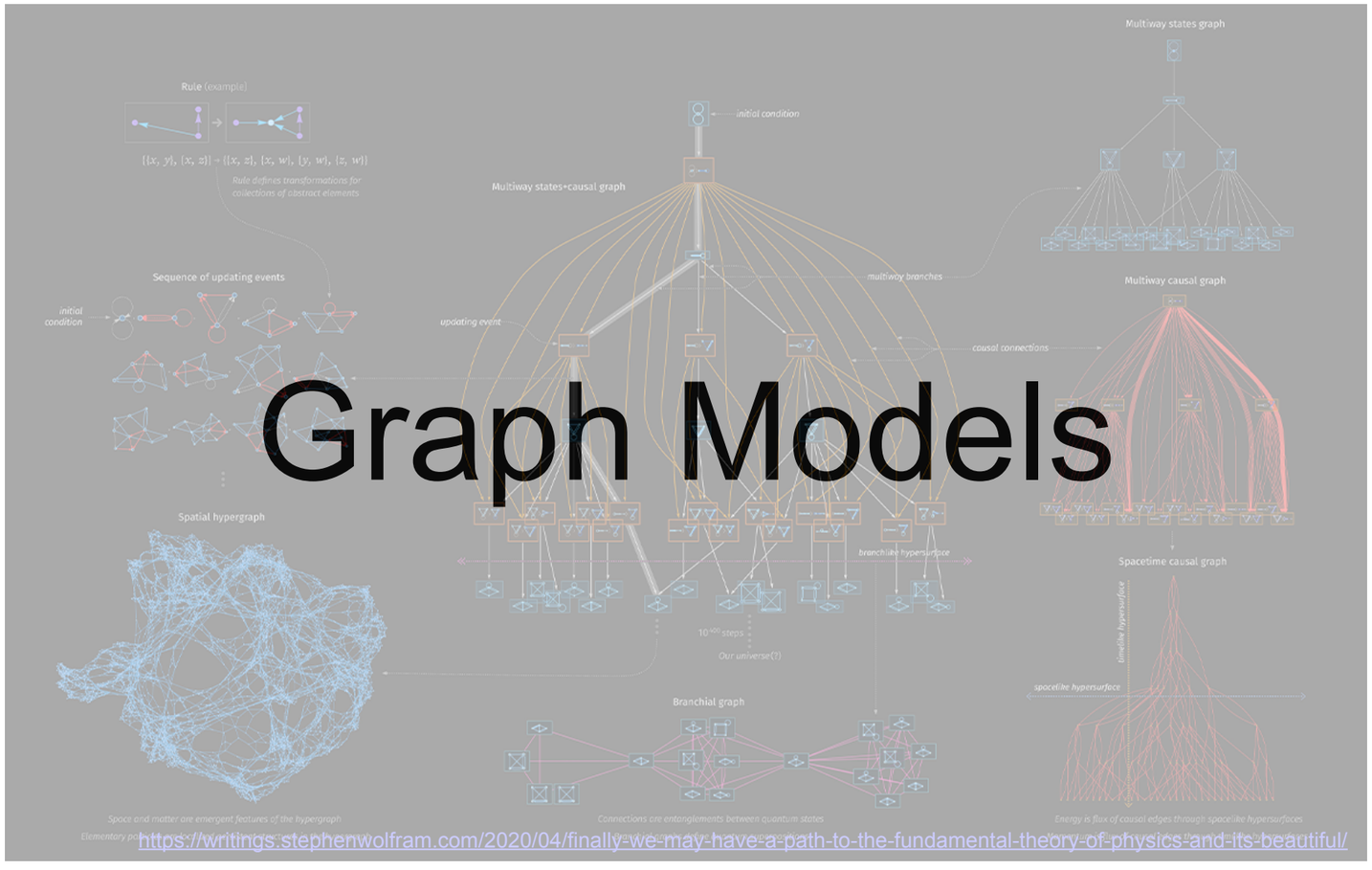
- Spiking Neural Networks are closer to the actual biological brain
- Adapted to temporal data
- Hardware implementation with low power consumption
- Usually trained using evolutionary algorithms
- Demonstrated to be economical models





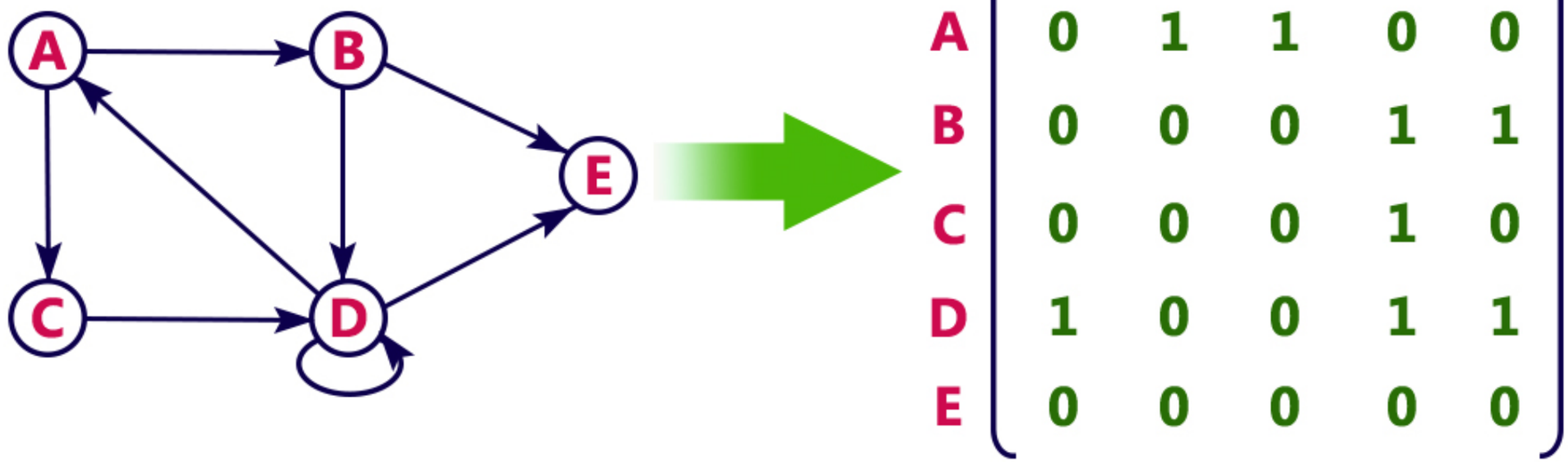
https://ieeexplore.ieee.org/document/8259423

	Deep Learning	Spiking
Training Method	Back-propagation	Not well established (here, genetic algorithms)
Native Input Types	Images/Arrays of values	Spikes
Network Size	Large (many layers, many neurons and synapses per layer)	Relatively small (fewer neurons and sparser synaptic connections)
Processing Abilities	Good for spatial	Good for temporal
Performance	Well understood and state-of-the-art	Not well understood



Graph Models

Forewords on Graph



http://btechsmartclass.com/data_structures/graph-representations.html

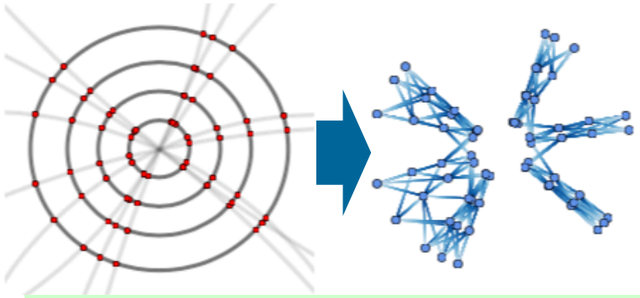
A graph is composed of

- **Nodes** that can be represented as a vector.
- **Edges** that can be represented with the adjacency matrix.

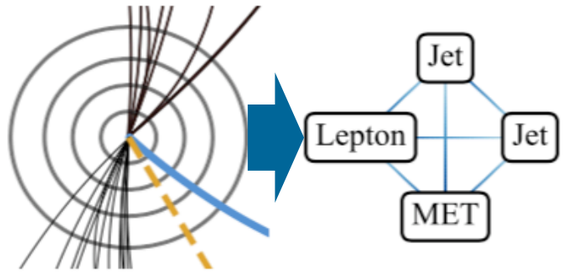
→ Flowing of information using matrix operations.

→ With machine learning on graphs, edges and nodes might acquire latent representations.

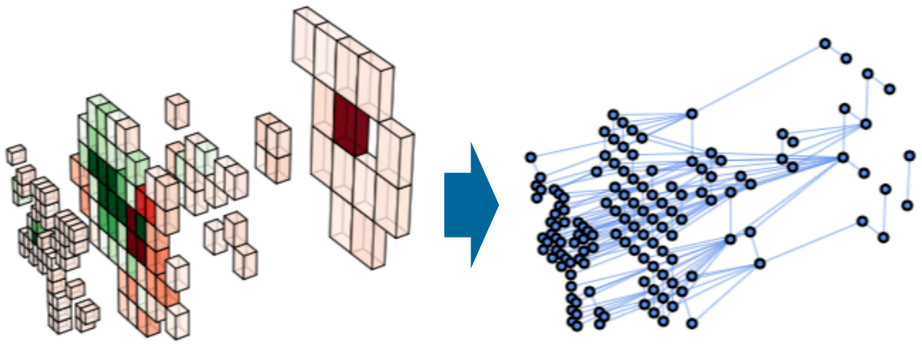
Graph Representation



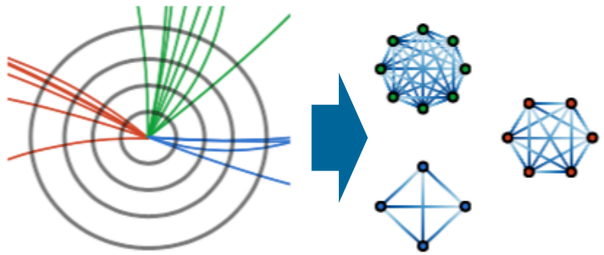
Hits in tracking detector



Objects in an event



Hits in calorimeter detector



Object sub-structure in an event

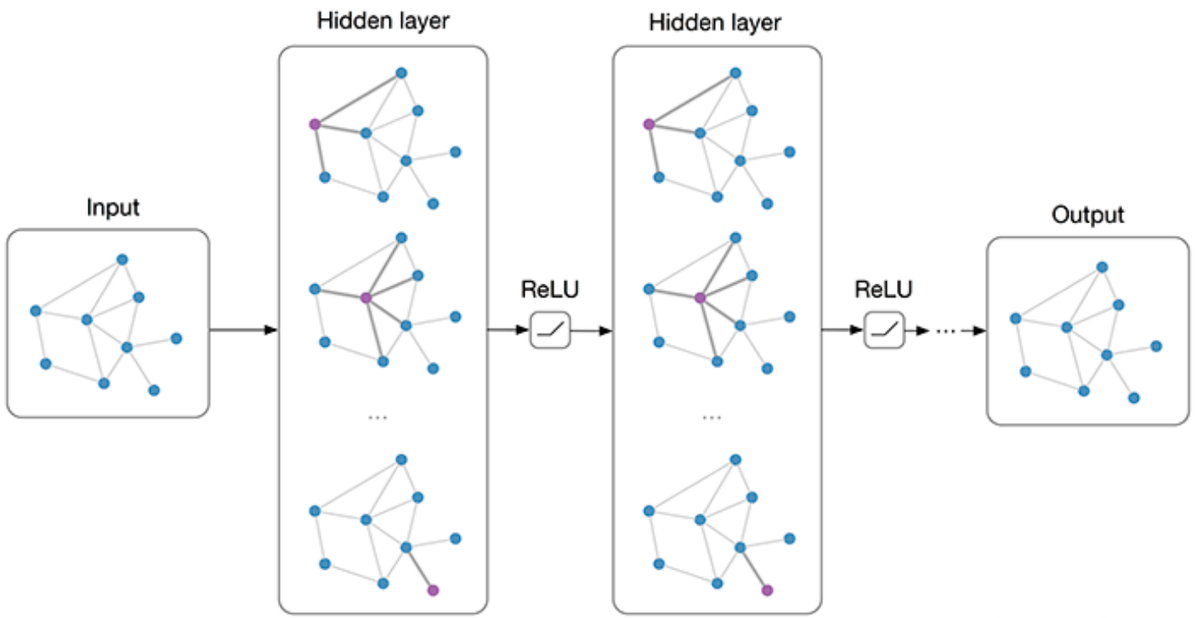
Graph Neural Networks in Particle Physics
[2007.13681]

Heterogenous data fits well in graph/set representation.

The two Infinite

- Reminder: Neural networks can take many forms. **Any analytical functions** with trainable parameter can serve as neural networks.
- Graph Neural Networks, operating on graph-like data is sitting **in between meta-structure and uber-structure**.
- Quite fertile ground for innovation. Challenging to cover all possible architectures.
- Covering below some of the essential concepts and features.

Graph Convolutional Network



https://tkipf.github.io/graph-convolutional-networks/

$$H^{(0)} = X$$

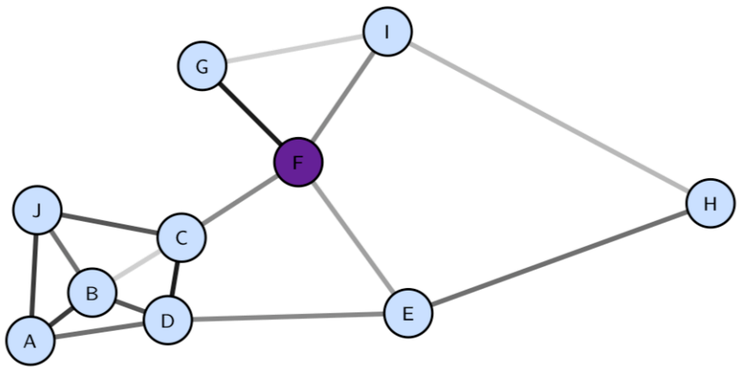
$$H^{(l+1)} = \sigma\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}\right)$$

$$\tilde{A} = A + I_N \quad \tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$$

Semi-Supervised Classification with Graph Convolutional Networks [1609.02907]

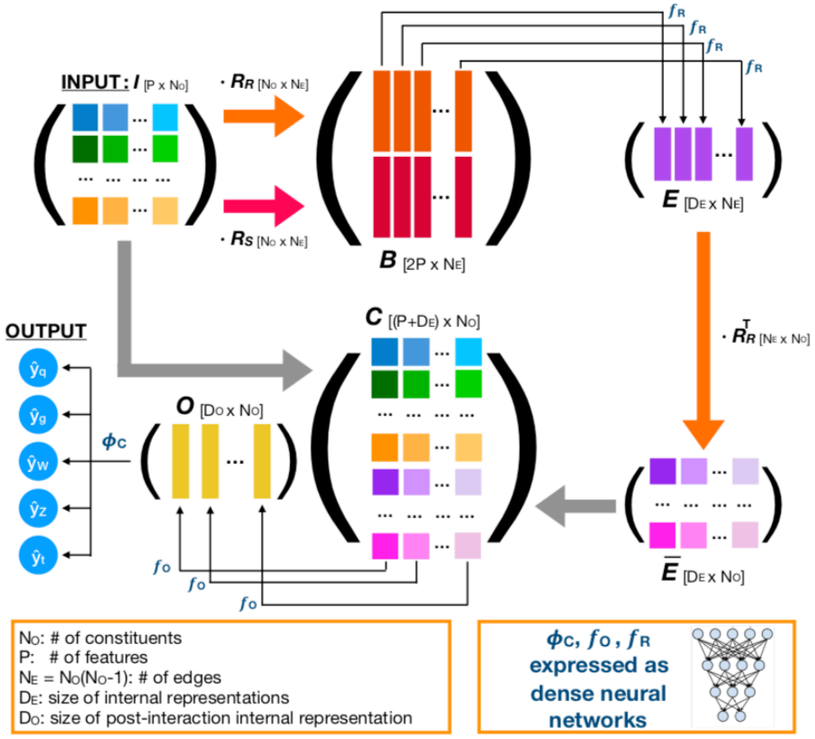
Fixed graph connectivity and adjacency matrix (A).
Make us of spectral graph theory.
Update rule **based on adjacency matrix** and learnable parameters (W).

Message Passing

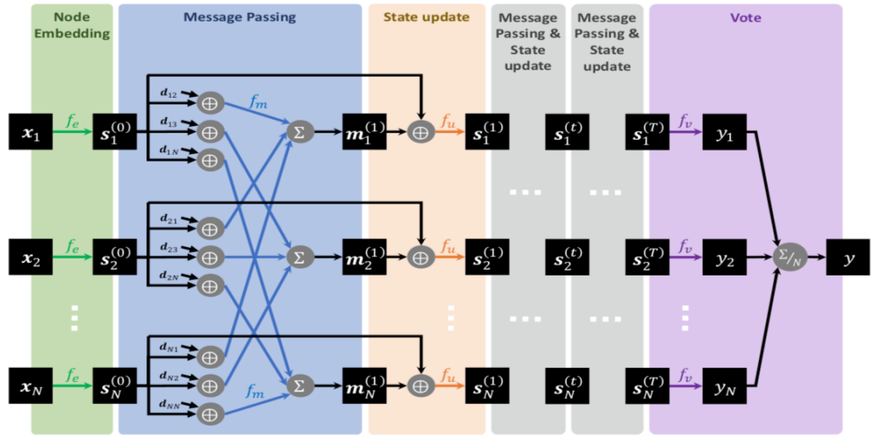


$$\tilde{m}_j^t = f(h_j^{t-1})$$
$$m_{j\to i}^t = \sigma(A_{ij}\tilde{m}_j^t)$$
$$h_i^t = \mathrm{GRU}(h_i^{t-1}, \Sigma_j m_{j\to i}^t)$$

Neural Message Passing for Jet Physics
ML4PS2017.29



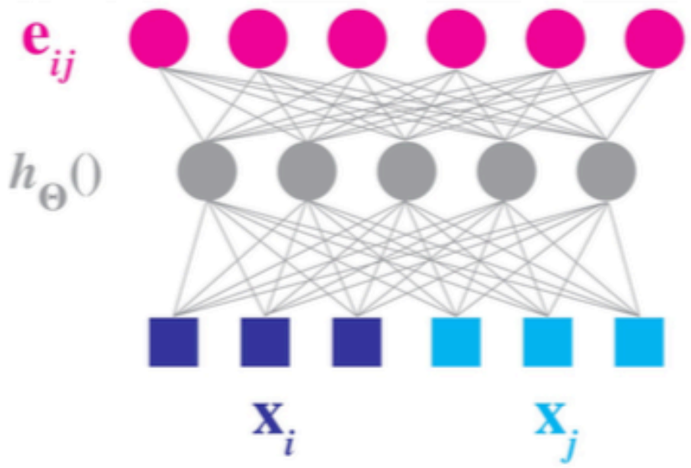
JEDI-Net: a jet identification algorithm based on interaction networks
[1908.05318]

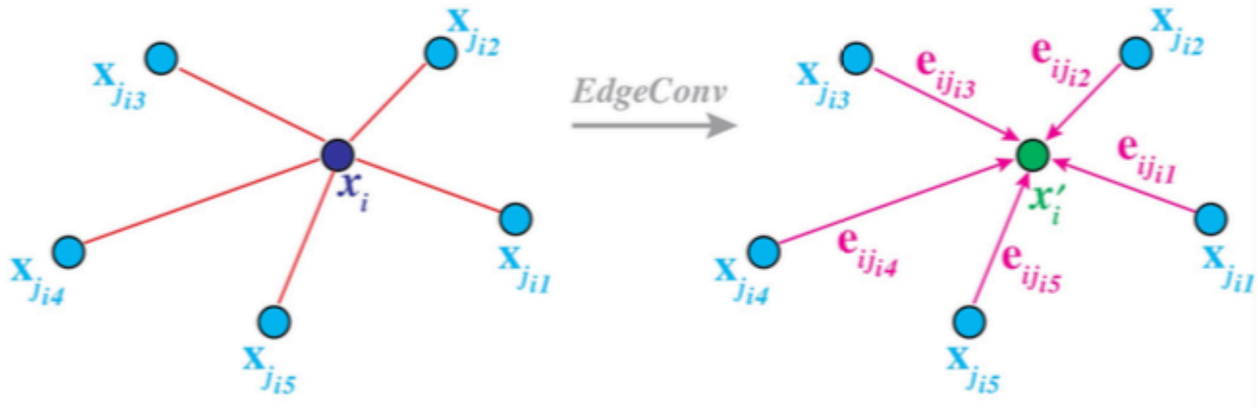


Probing stop pair production at the LHC with graph networks [1807.09088]

Information constructed on nodes is propagated to connected nodes.
Multiple ways of achieving **message passing concept**.

Dynamic Graph CNN (DGCNN)





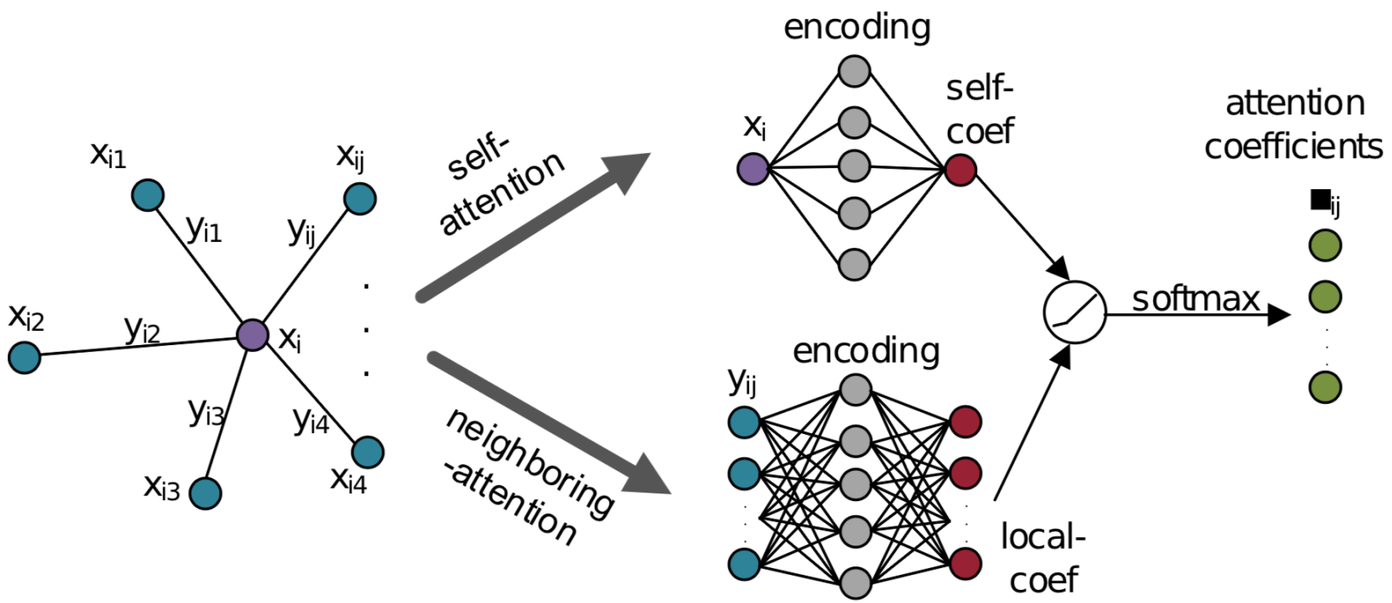
$$h_\Theta : \mathbb{R}^F \times \mathbb{R}^F \longrightarrow \mathbb{R}^{F'}$$

$$\mathbf{x}'_i = \square_{j:(i,j)\in\mathcal{E}} h_\Theta(\mathbf{x}_i, \mathbf{x}_j)$$

Dynamic Graph CNN for Learning on Point Clouds [1801.07829]

Convolutional-type edge representation model.
Representation aggregation mechanism for node update.
Dynamical connectivity as k-NN in latent representation.

Graph Attention



$$y_{ij} = (x_i - x_{ij})$$

$$c_{ij} = LeakyReLU(h(x'_i, \theta) + h(y'_{ij}, \theta))$$

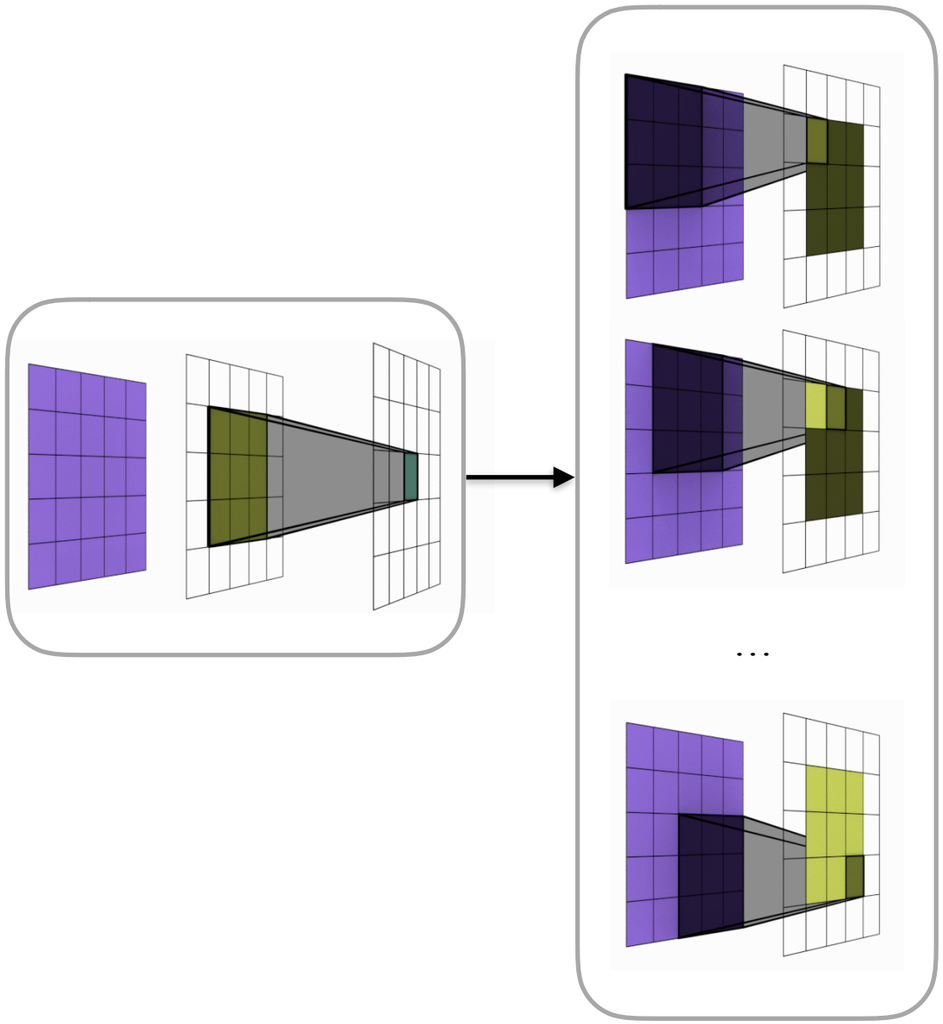
$$\alpha_{ij} = \frac{exp(c_{ij})}{\sum_{k \in N_i} exp(c_{ik})}$$

$$\hat{x}_i = f(\sum_{j \in N_i} \alpha_{ij} y'_{ij})$$

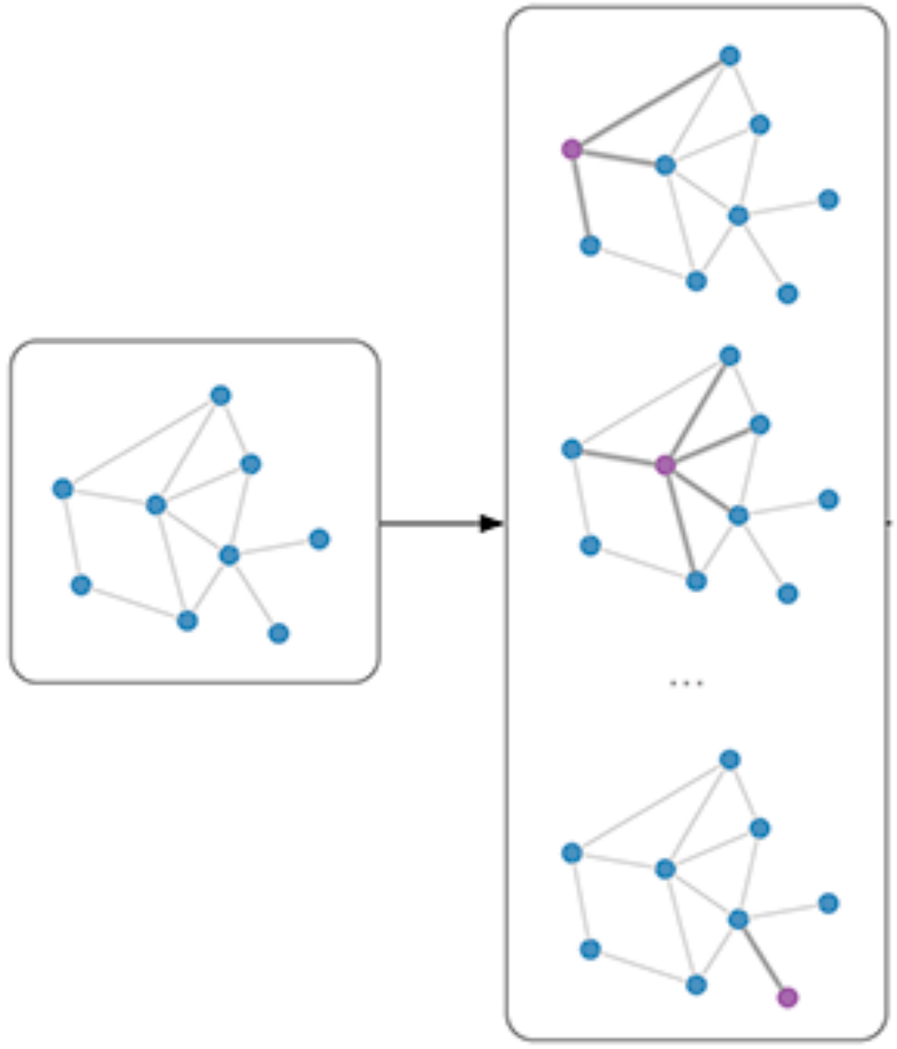
GAPNet: Graph Attention Based Point Neural Network for exploiting Local Feature of Point Cloud [1905.08705]

Graph connectivity as k-NN in node representation.
Edge features as node-feature difference.
Attention coefficients normalized over neighbors (c↦α).
Multi-head mechanism: stabilization by ensembling.

Shared Operations



https://imgur.com/gallery/AIFHqe9



https://tkipf.github.io/graph-convolutional-networks/

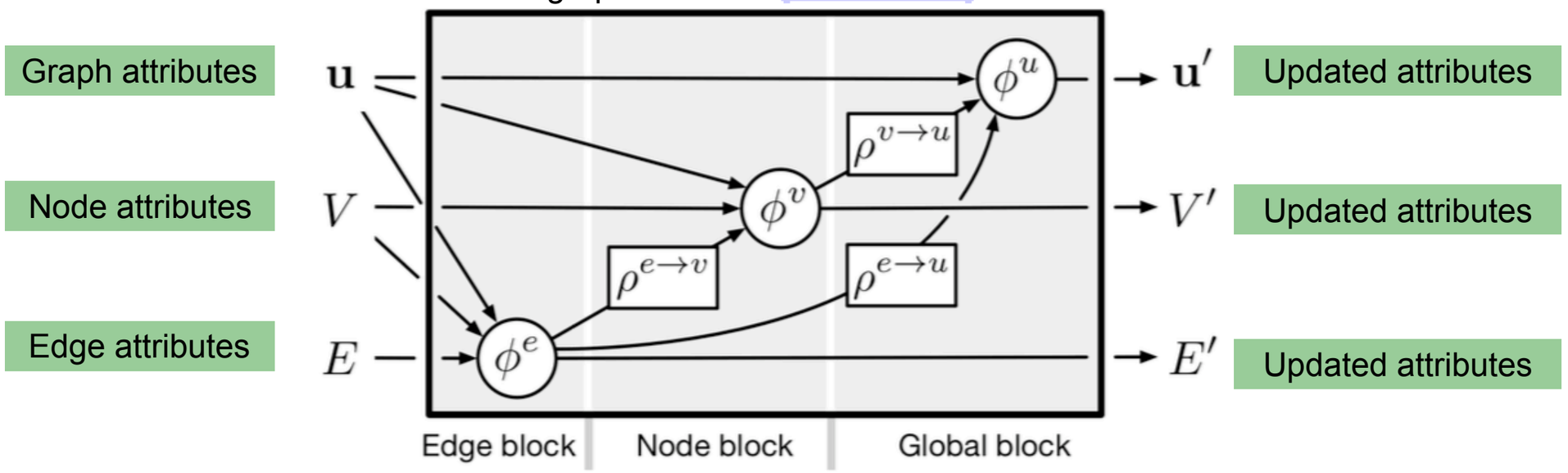
Graph Connectivity

- “Sets” come with no connectivity at all.
- Fully connected graphs has the most information flow, though can become computationally prohibitive.
- GNN can be made tractable through sparsification.
- Fixed/dynamic connectivity from input/latent space.
- Dynamic connectivity is only re-indexing, and let the gradients flow (similarly to maxpooling).

- Risk of dynamic connectivity from latent space to not have the adequate gradient flow to train the latent space. Random initialization or additional elements required.
- Risk of having heavy algorithms in dynamic edge definition.


Graph Neural Networks Formalism

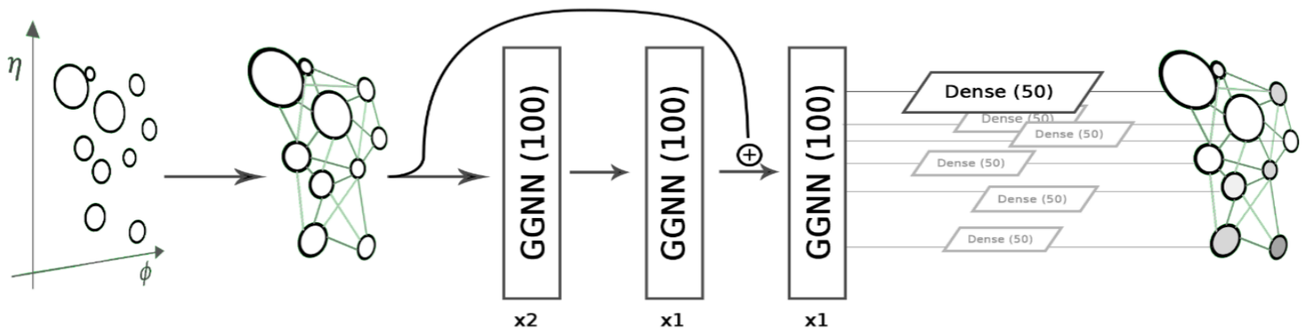
Relational inductive biases, deep learning, and graph networks [1806.01261]



Lots of possibilities to operate on a graph.
Most available architectures can be expressed with Φ and ρ.

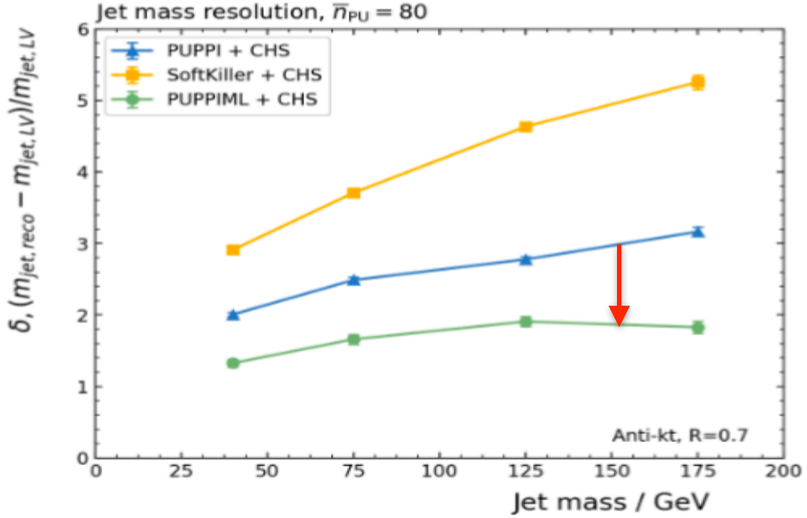
Readily software:
https://github.com/deepmind/graph_nets (TF)
https://pytorch-geometric.readthedocs.io (Torch)
https://jraph.readthedocs.io (JAX)
https://docs.dgl.ai (py)

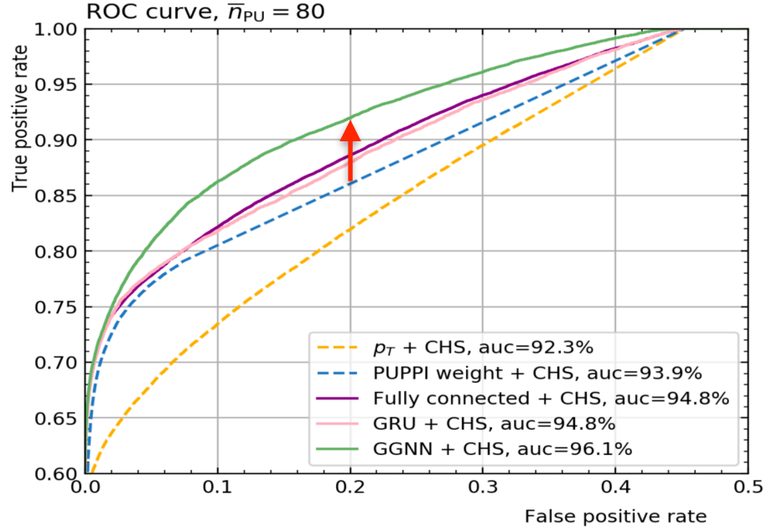
Pile-Up Mitigation



Pileup mitigation at the Large Hadron Collider with Graph Neural Networks [1810.07988]

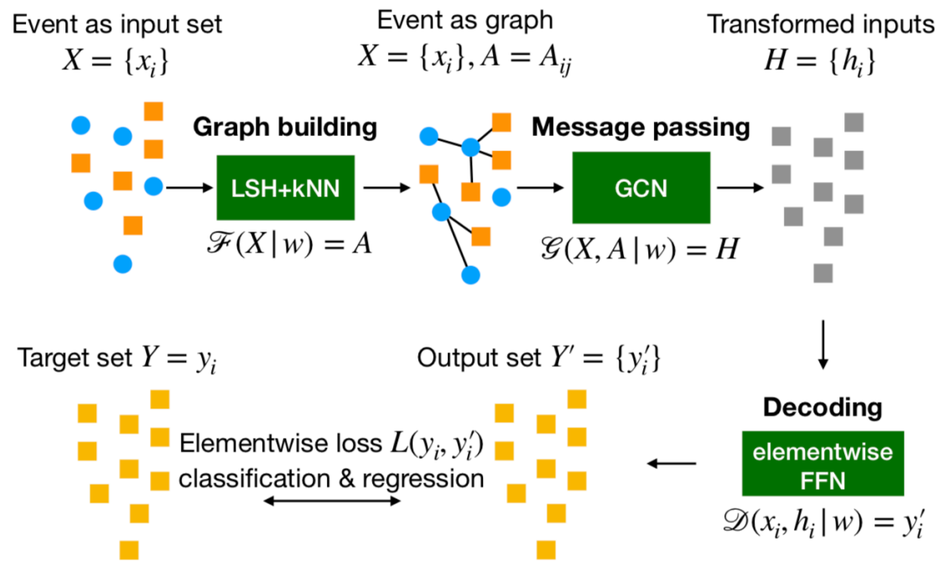
- Locally connected graph of reconstructed particle flow candidates
- Gated graph neural network (GGNN) to evolve node representations
- per-particle pile-up classification extract for neutrals





Particle Flow Reconstruction

MLPF: Efficient Machine-Learned Particle-Flow Reconstruction Using Graph Neural Networks [2101.08578]

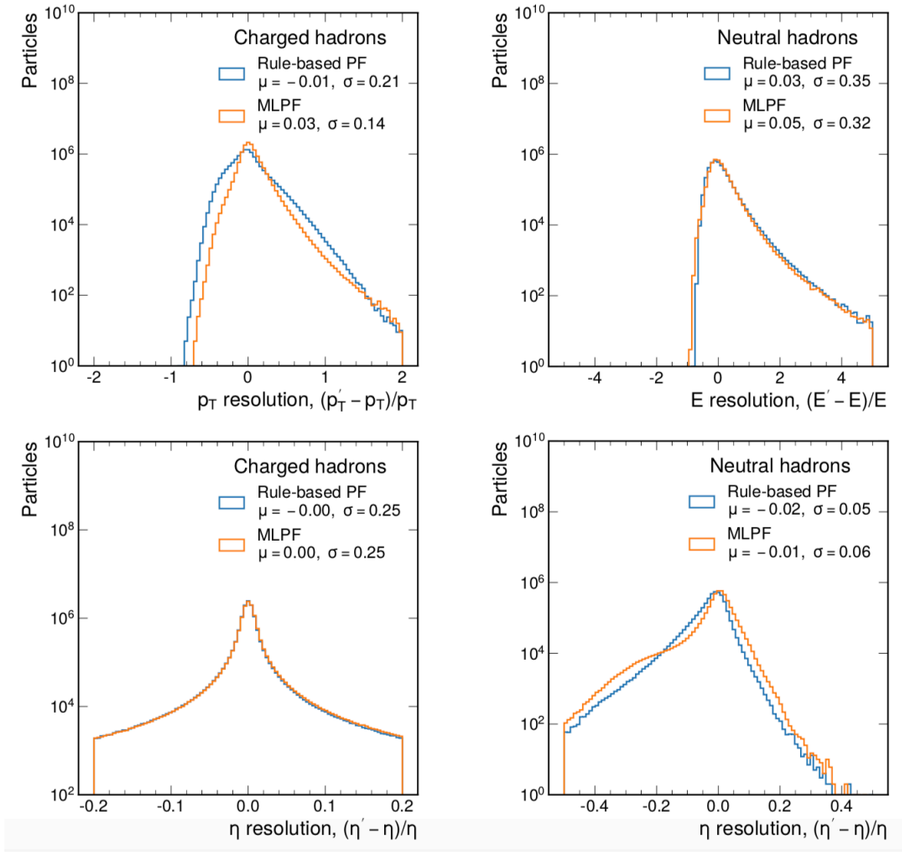


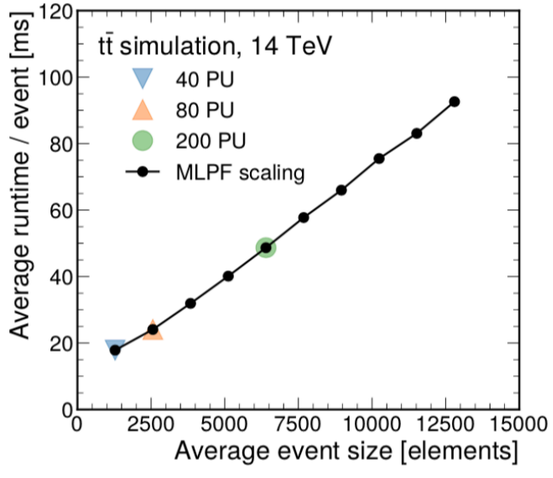
$x_i = [\text{type}, p_T, E_{ECAL}, E_{HCAL}, \eta, \phi, \eta_{outer}, \phi_{outer}, q, \ldots]$, type $\in$ {track, cluster}

$y_i = [\text{PID}, p_T, E, \eta, \phi, q, \ldots]$, PID $\in$ {none, charged hadron, neutral hadron, $\gamma, e^{\pm}, \mu^{\pm}$}

$h_i \in \mathbb{R}^N, N = 256$

Trainable neural networks: $\mathcal{F}, \mathcal{G}, \mathcal{D}$





- Set of tracks & clusters in input
- Classify sub-set of graph nodes
- Regress parton kinematics
- Execution time linear with PU

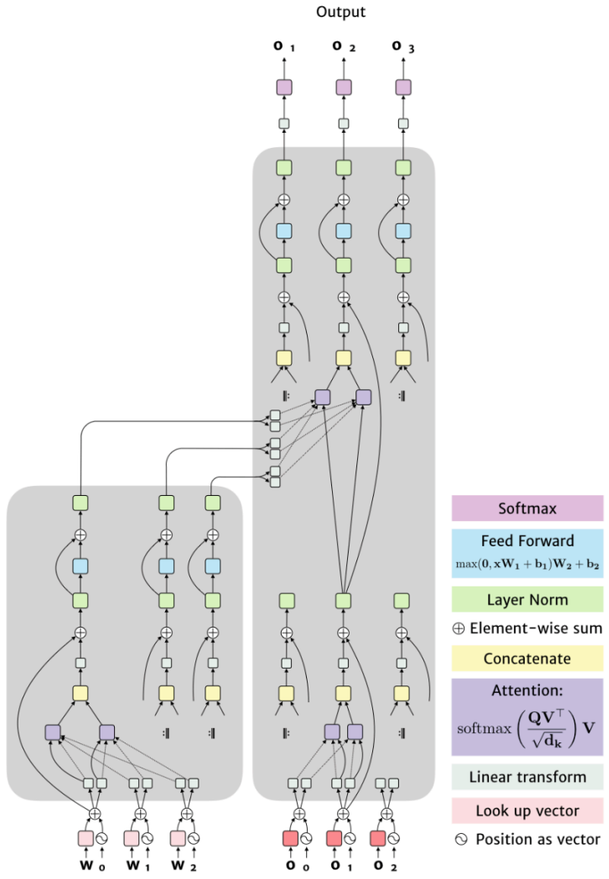
Take Home Message

Most HEP data is amenable to graph network.

Numerous ways of building a model.

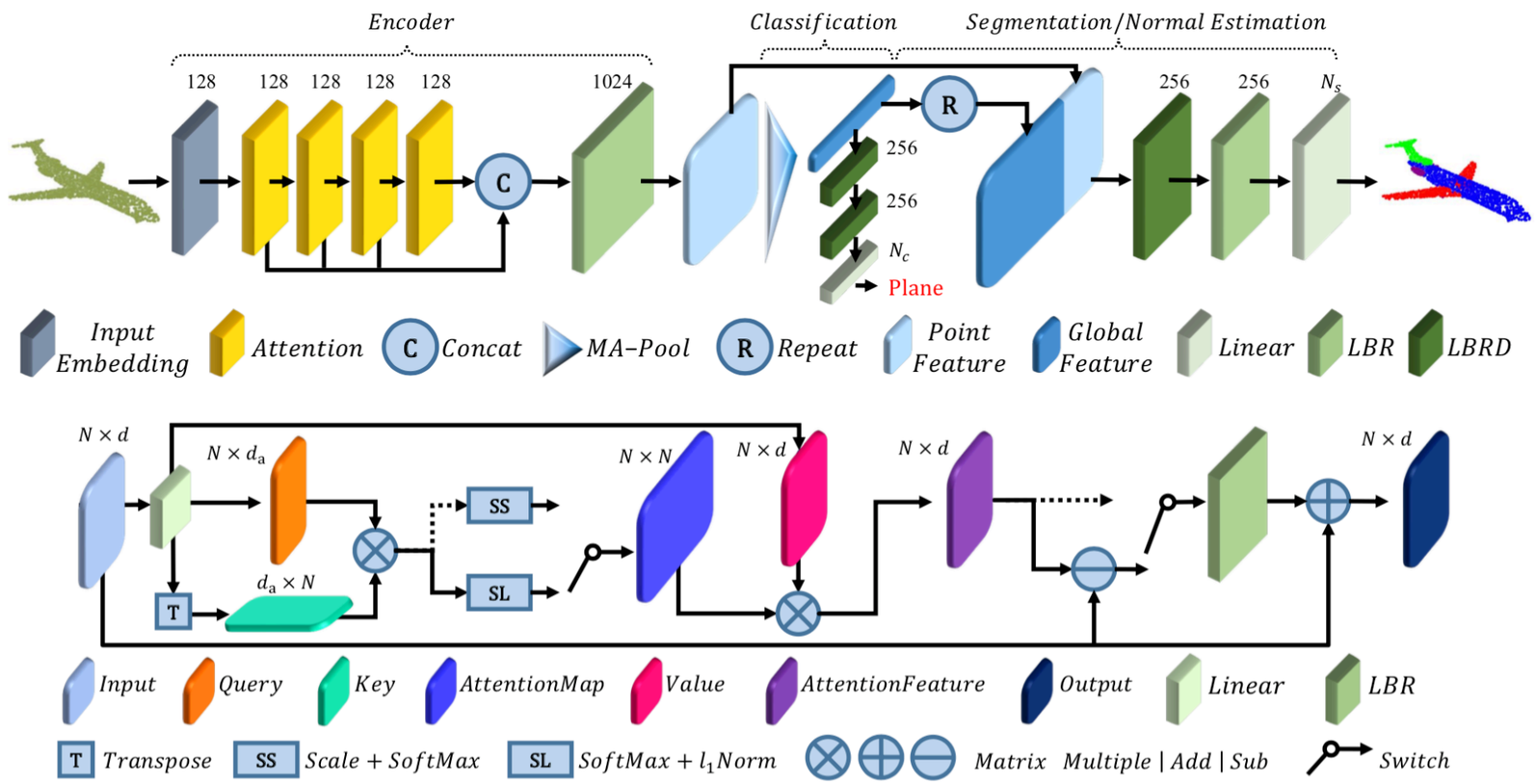
Computationally challenging at times.

Transforming Derivative



https://jkk.name/post/2017-10-20_onlyattention/

Attention is all you need [1706.03762]



PCT: Point Clout Transformer [2012.09688]

Transformer, based on attention, initial developed for NLP.
With minor modification, can be **applied to set-like data**.
Computationally efficient.

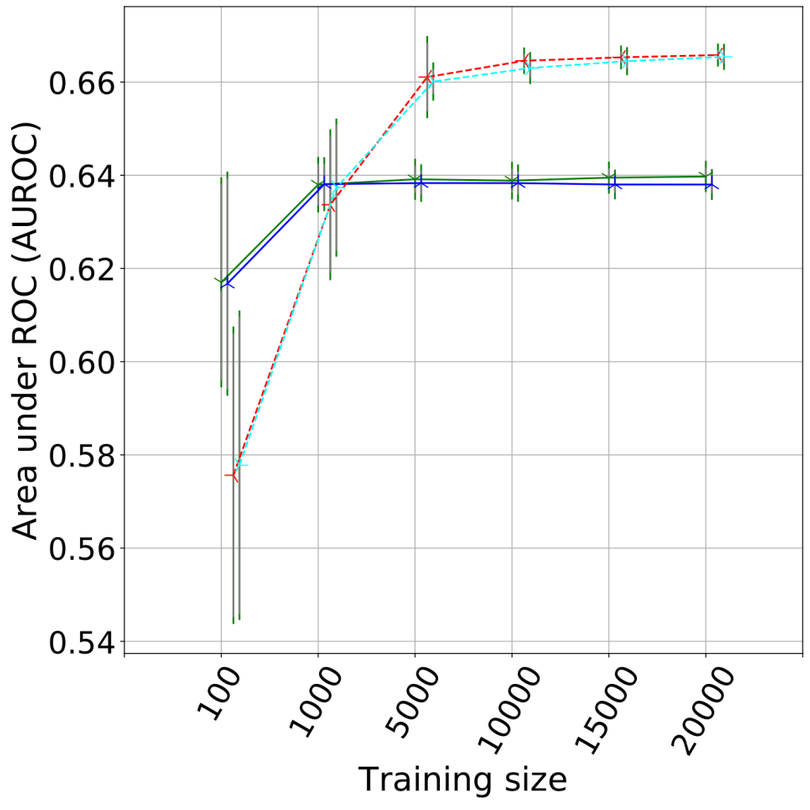

Outlook
https://www.jpl.nasa.gov/news/testing-proves-its-worth-with-successful-mars-parachute-deployment

Architecture Search

- By now you realize the extent of the space of meta-structure, and models : **“the sky is the limit”** situation.
- Two main directions of search:
  - ➡structural: e.g use layer of **type X**
  - ➡hyper-parameters: e.g use **N layers** of type X
- Compare models in controlled manner to be conclusive:
  - ➡Enough data provided: over-parametrization, efficiency.
  - ➡Convergence reached: generalization.
  - ➡Accurate performance estimation: cross-validation
  - ➡Efficient navigation: bayesian optimization, genetic. algorithms, inductive bias.

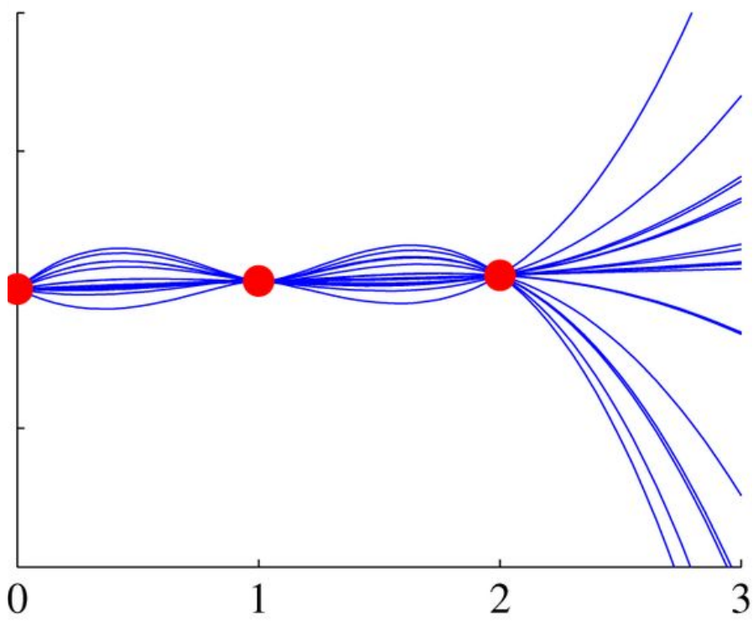

Need for Data

- "What is the **best performance one can get** ?" rarely has an answer
- When comparing multiple models, one can answer "what is the **best of these models, for this given dataset** ?"
- It does not answer "what is the **best model at this task** ?"



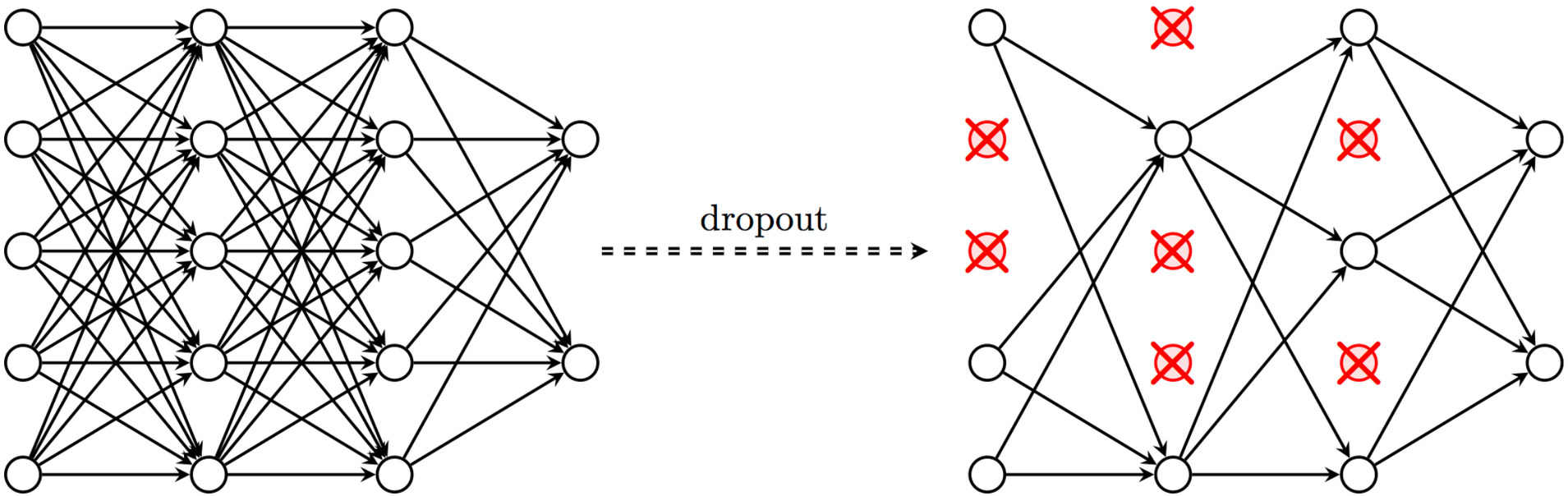
Data Efficiency

- Degeneracy in objective function can arise from under-constrained model
  - Not enough data point to constraint all parameters
- Family of solutions make the optimization algorithm stationary on the training set, while fluctuating on the testing set
- Limit of under-constraint is model type dependent.
- Can usually happen with deep learning models



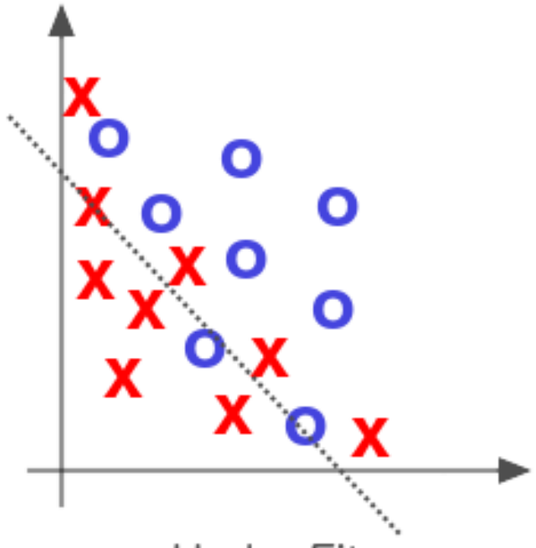
Drop-Out

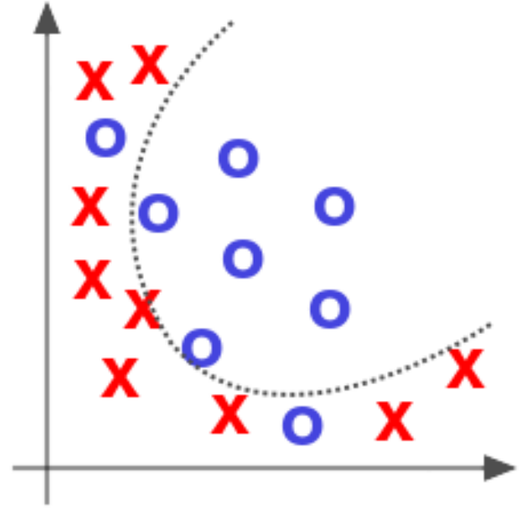
- Special case of regularization in artificial neural network
- During training of the model, set some nodes inactive at a random rate



Under-fitting

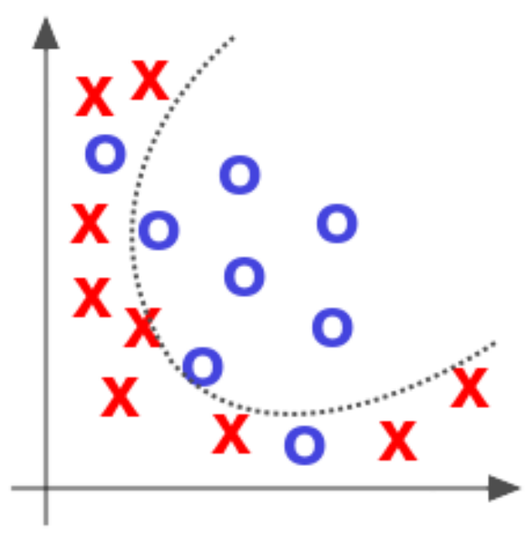
- Poor model performance can be explained
  - Lack of modeling capacity (not enough parameters, inappropriate parametrization, …)
  - Model parameters have not reached optimal values



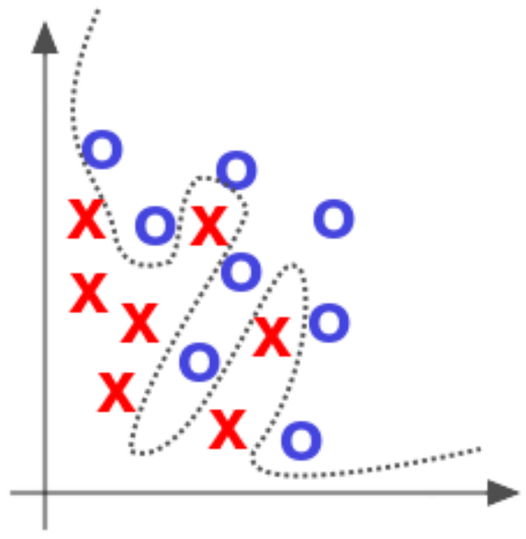


Over-fitting

- "Too good to be true" model performance can be explained
  - Excessive modeling capacity (too many parameters, parametrization is too flexible, ...)
  - Model parameters have learn the trained data by heart
- Characterized by very good performance on the training set and (much) lower performance on unseen dataset



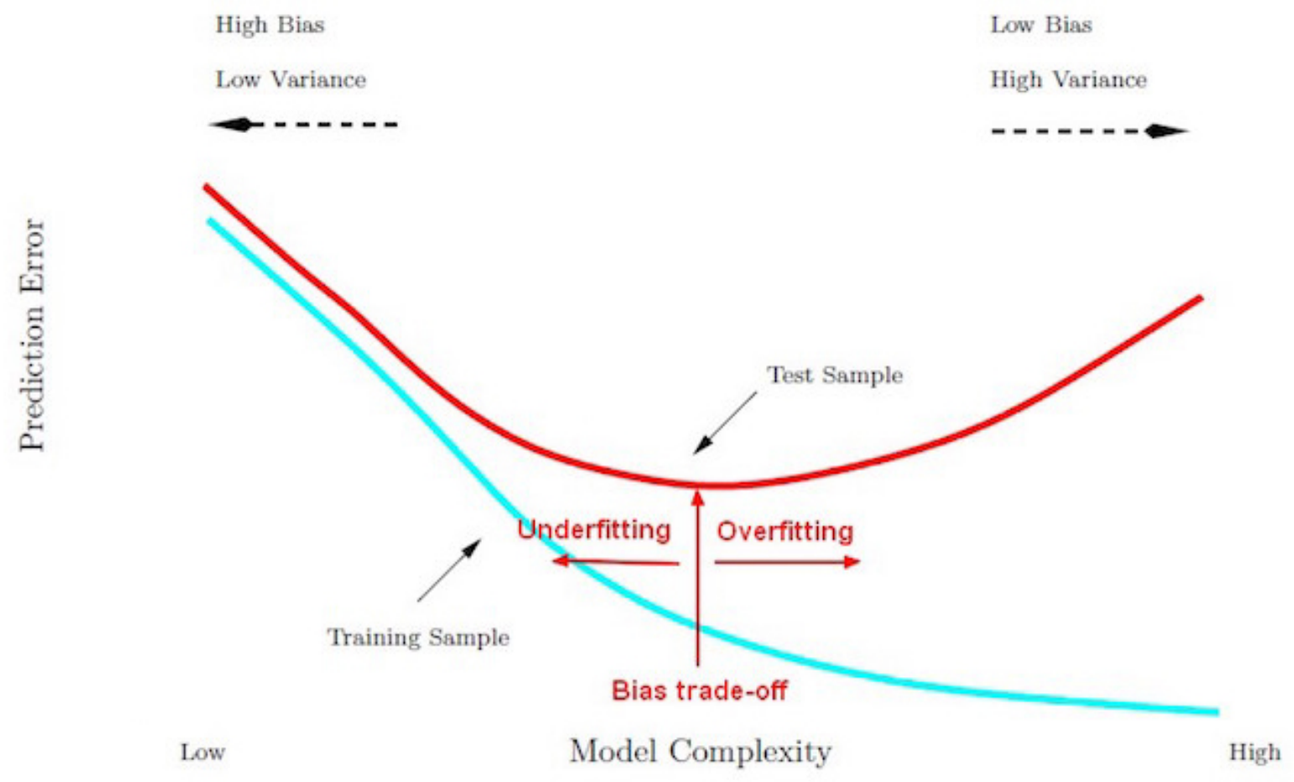
Appropriate



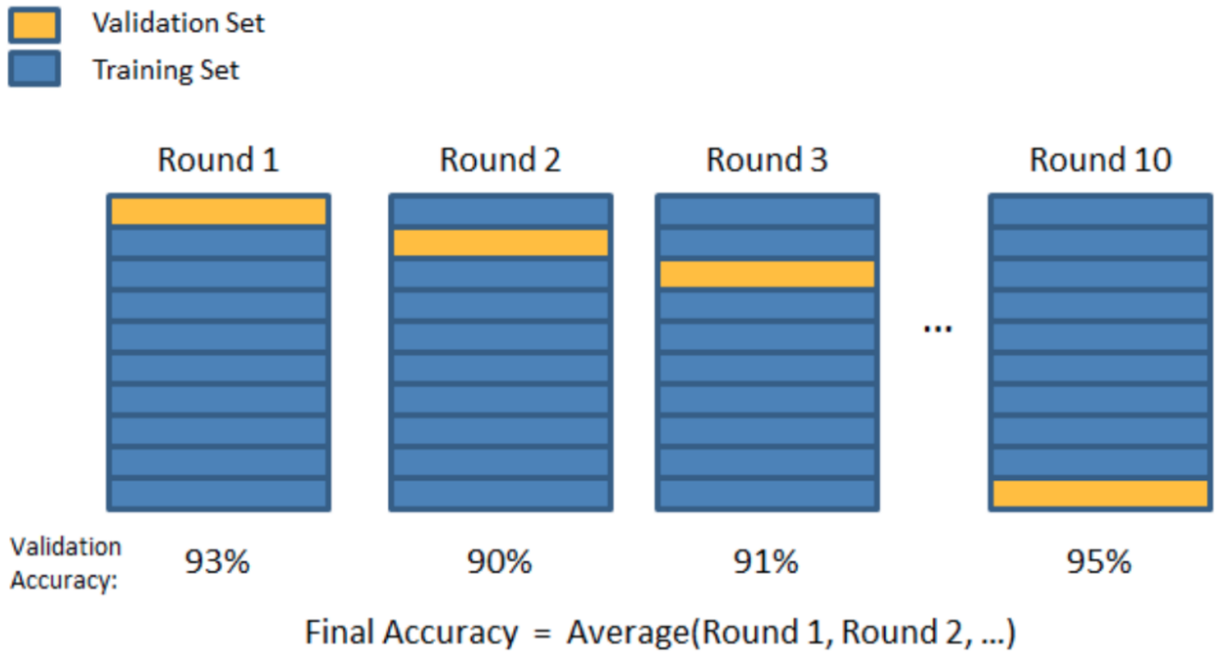
Over Fit

Generalization

- Systematic error ≡ bias
- Sensitivity of prediction ≡ variance
- A good model is a tradeoff both



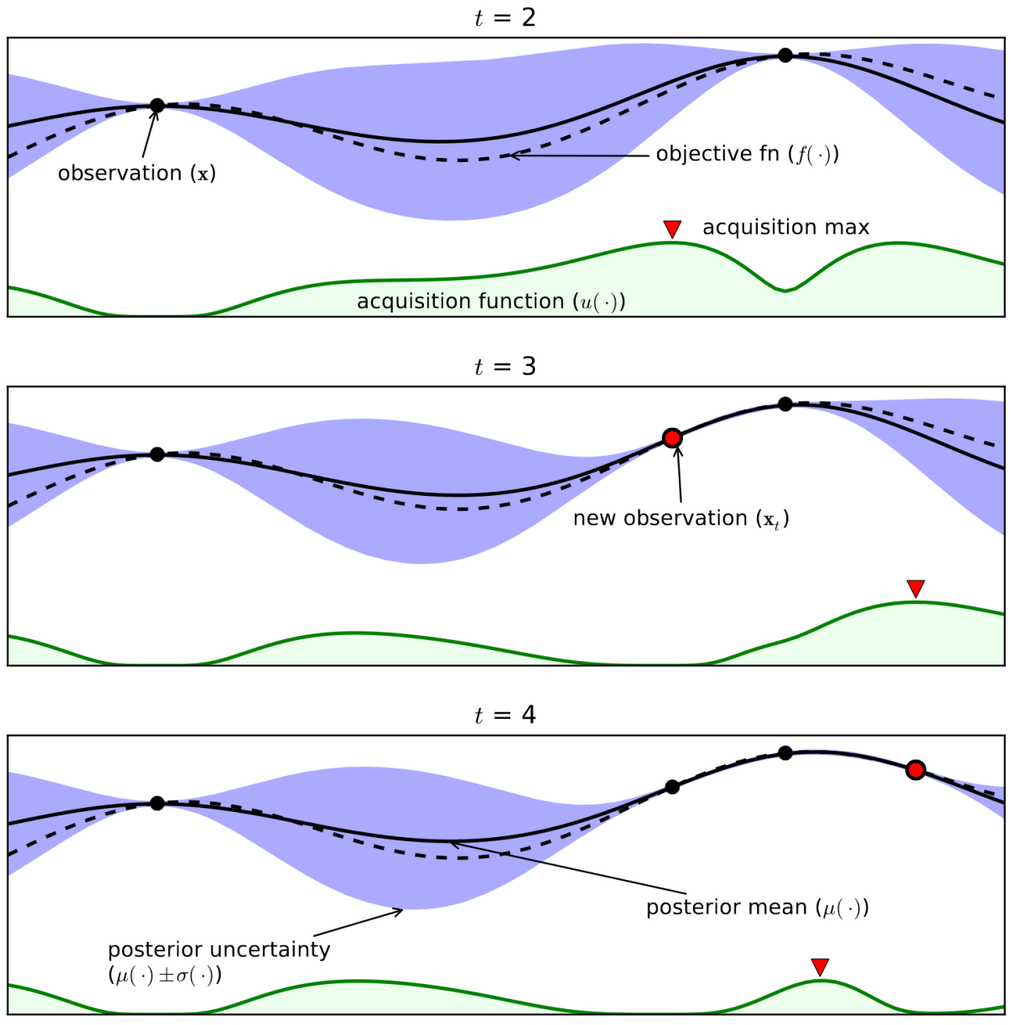
Cross Validation



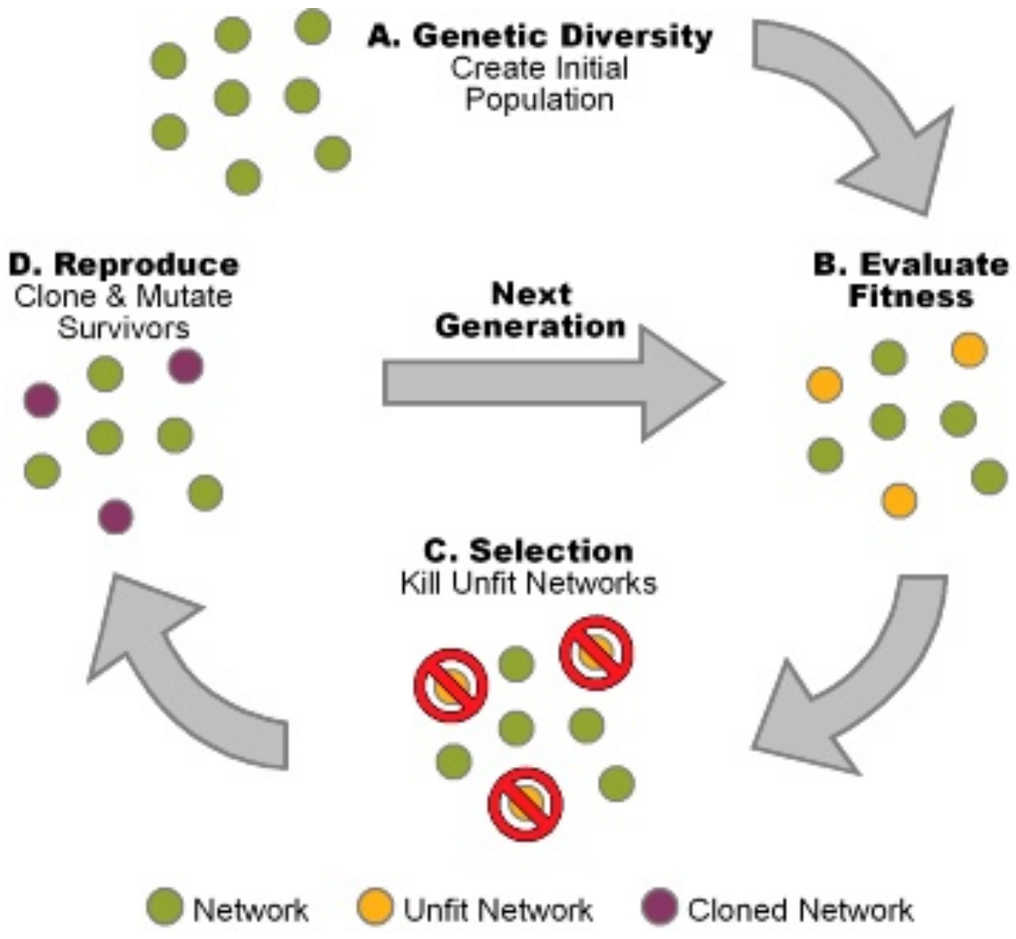
- Model selection requires to have an estimate of the uncertainty on the metric used for comparison
- K-folding provides an un-biased way of comparing models
- Stratified splitting (conserving category fractions) protects from large variance coming from biased training
- Leave-one-out cross validation : number folds ≡ sample size

Bayesian Optimization

- Applicable to optimize function **without close form** and that are **expensive to call** (numerical gradient impractical)
- Approximate the objective function with **Gaussian processes** (GP)
- Start at random points, then sample according to optimized acquisition function
  - Expected improvement
    - $EI(x) = -E(f_{GP}(x) - f(x_{best}))$
  - Lower confidence bound
    $LCB(x) = \mu_{GP}(x) + \kappa\sigma_{GP}(x)$
  - Probability of improvement
    - $PI(x) = -P(f_{GP}(x) \geq f(x_{best}) + \kappa)$



Evolutionary Algorithms



- Applicable to function in high dimensions, with a non regular landscape
- Start from random population
- Estimate fittest fraction of individuals
- Bread and mutate individuals

- Direction of optimization is given by the cross-over and mutation definition
- Multiple over algorithms : particle swarn, ...

Inductive Bias

- In data-science, one derives new architectures by introducing new concept to constrain the information flow, the latent representation, …, to what one think should happen.

  ➡CNN, attention, message passing, …

- Further constraints imposed by the structure present naturally the data

  ➡RNN, GNN, …

- In Science, there is another handle to this, by further formatting the model architecture to match Physics principles

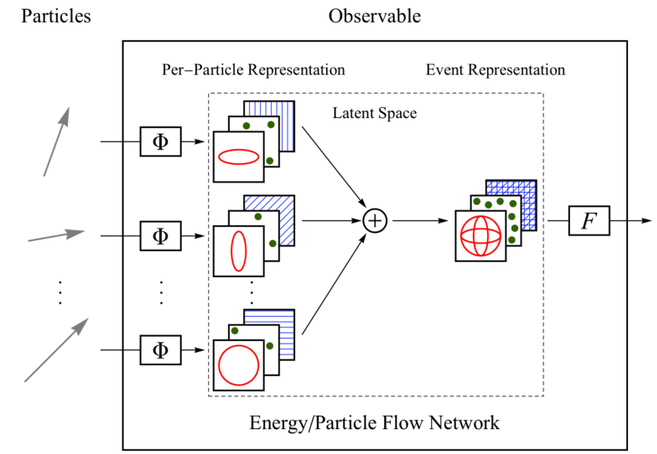
  ➡Equivariance, invariance, conservation, equation of motion, …


Equivariance and Invariance

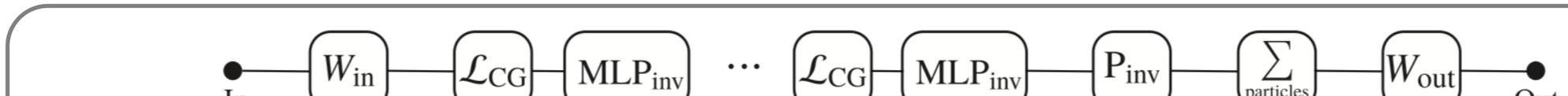
$$k_{\mu,i} \xrightarrow{\text{CoLa}} \tilde{k}_{\mu,j} = k_{\mu,i} \begin{pmatrix} 1 & 0 & \cdots & 0 & C_{1,N+2} & \cdots & C_{1,M} \\ 0 & 1 & & \vdots & C_{2,N+2} & \cdots & C_{2,M} \\ \vdots & \vdots & \ddots & 0 & \vdots & & \vdots \\ 0 & 0 & \cdots & 1 & C_{N,N+2} & \cdots & C_{N,M} \end{pmatrix}$$

$$\tilde{k}_j \xrightarrow{\text{LoLa}} \hat{k}_j = \begin{pmatrix} m^2(\tilde{k}_j) \\ p_T(\tilde{k}_j) \\ w_{jm}^{(E)}\ E(\tilde{k}_m) \\ w_{jm}^{(d)}\ d_{jm}^2 \end{pmatrix}$$

Lorentz Learning Layer
[1707.08966]



Deep set
[1810.05165]

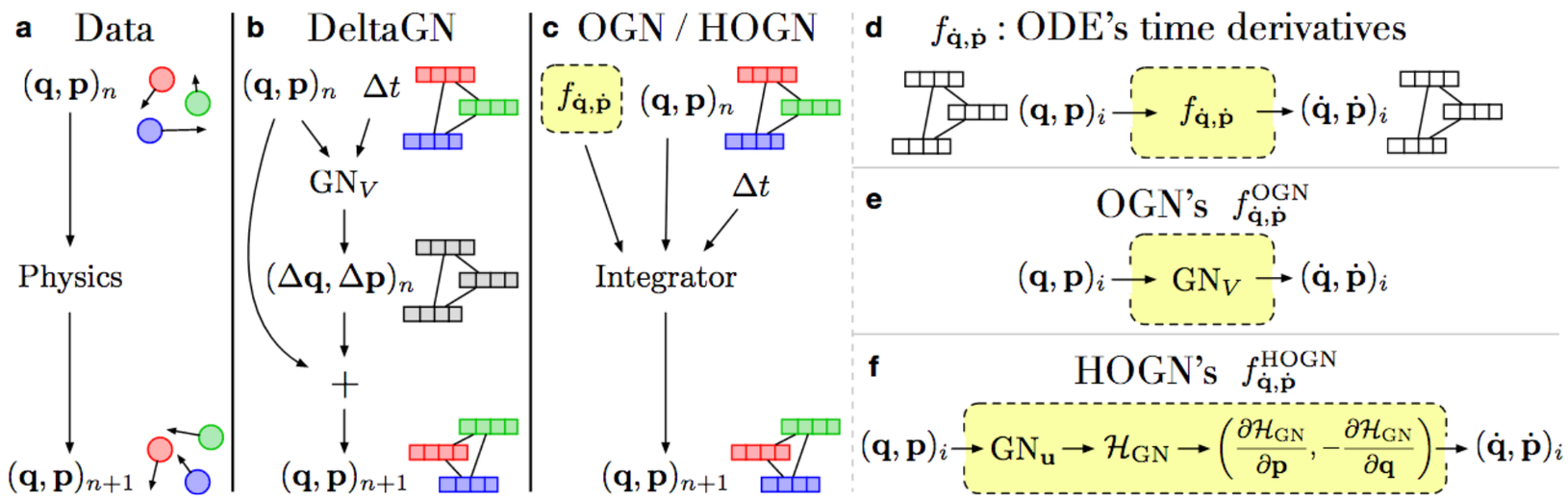


$$\mathcal{F}_i \mapsto W \cdot \left(\mathcal{F}_i \oplus \mathcal{F}_i^{\otimes 2} \oplus \sum_j f\left(p_{ij}^2\right) \cdot p_{ij} \otimes \mathcal{F}_j \right)$$

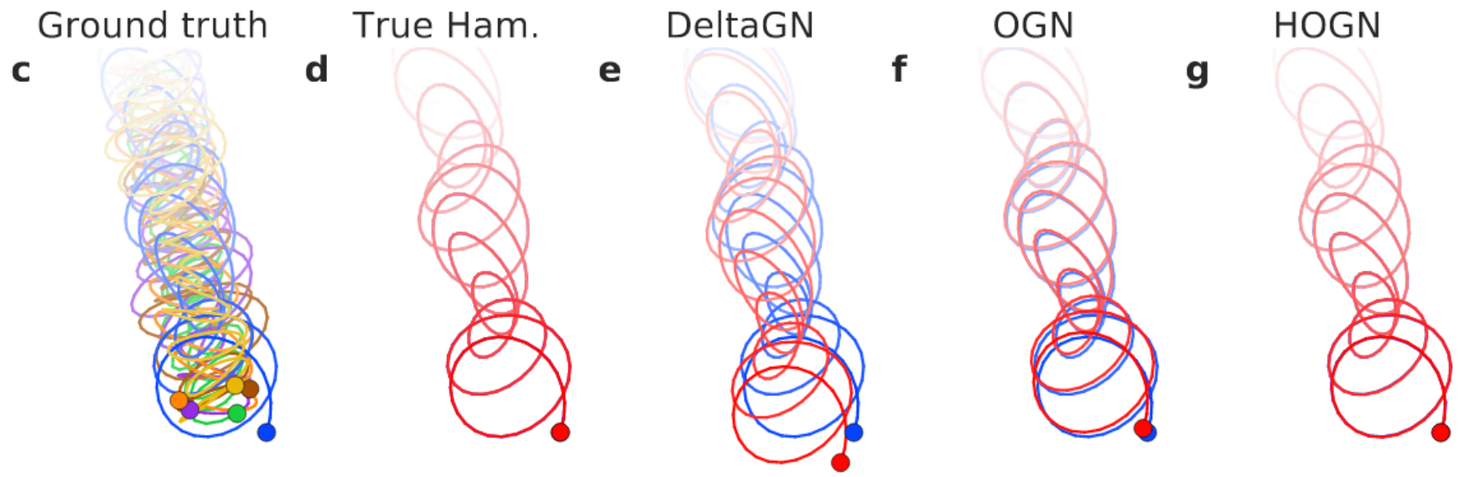
Lorentz group equivariant networks
[2006.04780]

Embed the symmetry and invariance in the model.
Economy of model parameters.

Use Physics



Rollout trajectories per model



A. Sanchez-Gonzalez, V. Bapst, K. Cranmer, P. Battaglia [1909.12790]

Let the model **include Physics principles** to master convergence

Summary

➡Artificial Neural Networks have a lot of plasticity.

➡Complex models are made of meta-structures.

➡Intuition carved the design of meta-structures.

➡Roam the architecture landscape with care.


DEPARTMENT OF ENERGY
UNITED STATES OF AMERICA


mPP


UHH

Stochastic Gradient Descent

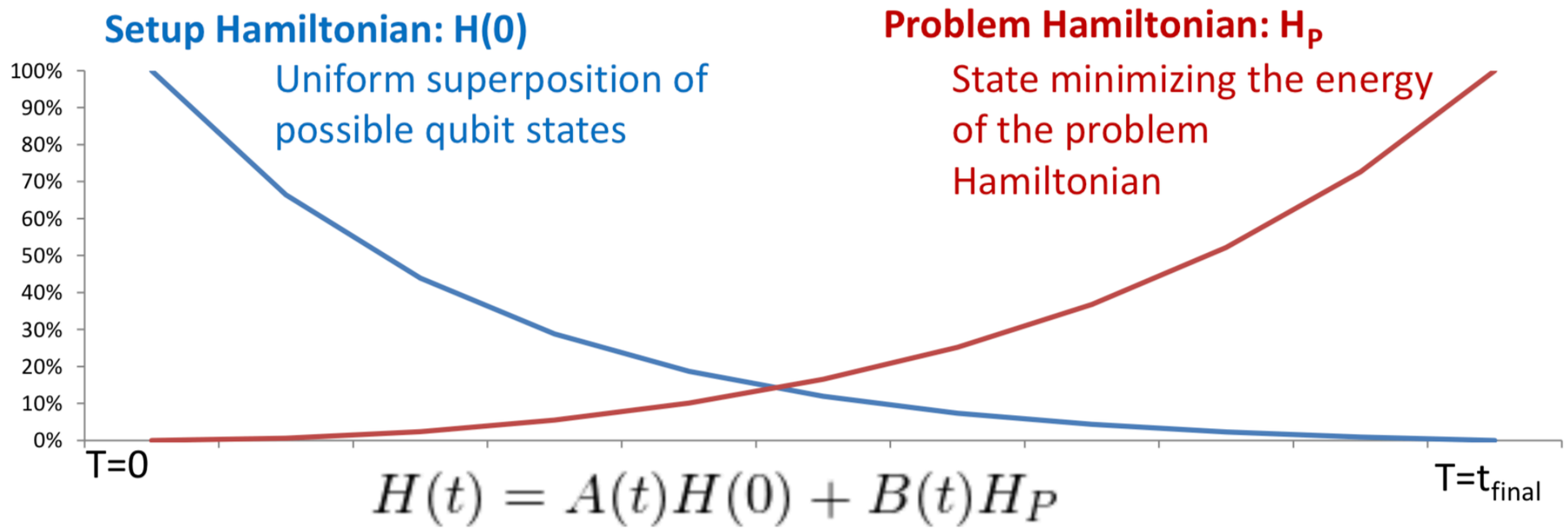
- Application of one gradient descent is expensive. Can be prohibitive with large datasets
- Following the gradient update from each and every sample of a dataset leads to tensions
    - In binary classification, samples from opposite categories would have “opposite gradients”
- Gradients over multiple samples are independent, and can be computationally parallelyzed

→ Estimate the effective gradient over a batch of samples

$$\nabla_{eff} f(x) = \frac{1}{N} \sum_{i \in batch} \nabla_i f(x)$$


Adiabatic Quantum Annealing

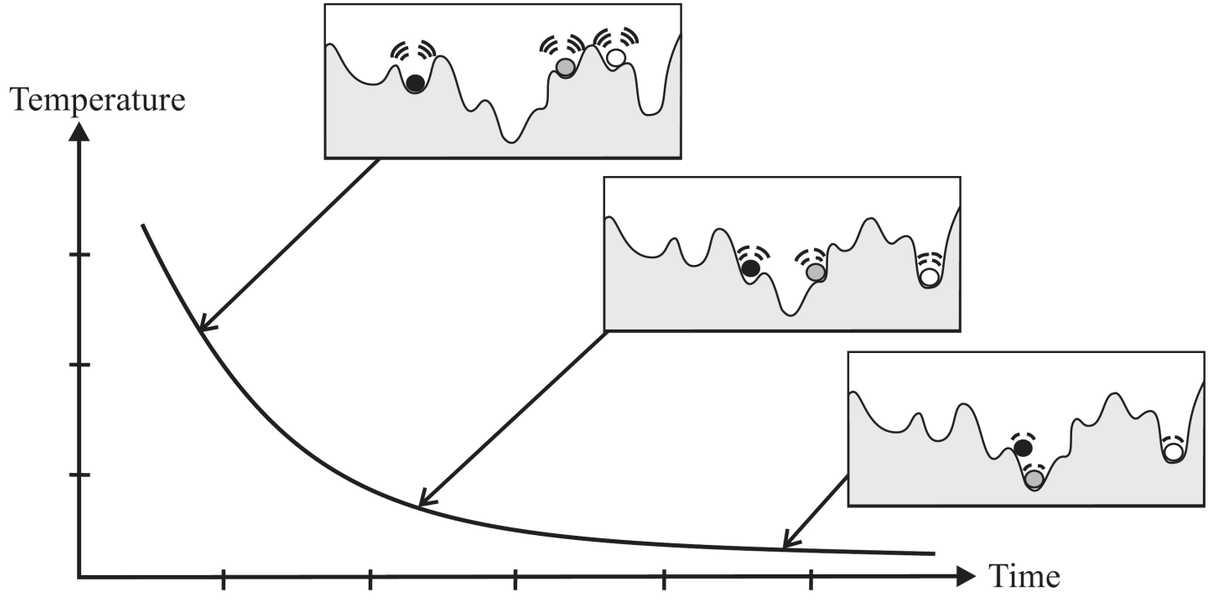
- System setup with trivial Hamiltonian H(0) and ground state
- Evolve adiabatically the Hamiltonian towards the desired Hamiltonian H_p
- **Adiabatic theorem** : with a slow evolution of the system, the state stays in the ground state.



https://arxiv.org/abs/quant-ph/0001106
https://arxiv.org/abs/quant-ph/0104129

Simulated Annealing

- Monte-Carlo based method to find ground state of energy functions
- Random walk across phase space
  - → accepting descent
  - → accepting ascent with probability $e^{-\Delta E/kT}$
- Decrease T with time



Non Analytical SGD

- Some valuable loss function might not be analytical and their gradients cannot be derived
- Used finite element method to estimate the gradient numerically

$$\nabla f(x) = \frac{f(x+\varepsilon) - f(x)}{\varepsilon}$$

- Method can be extended to using more sampling and better precision
- Quite expensive computationally in number of function calls and impractical in large dimension
- Robust methods available in most program library

Second Order Methods

- Newton-Raphson method defines a recursive procedure to find the root of a function, using its gradient.
- Finding optimum is equivalent to finding roots of the gradient, hence applying NR method to the gradient using the Hessian

$$f(x+\varepsilon) = f(x) + \varepsilon \nabla f(x) + \frac{1}{2}\varepsilon^T H(x)\varepsilon$$

$$\varepsilon \sim -H(x)^{-1} \nabla f(x)$$

- Convergence guaranteed in certain conditions
- Alternative numerical methods tackle the escape of saddle points and computation issue with inverting the Hessian
- In deep learning “hessian-free” methods are prohibitive computationally wise

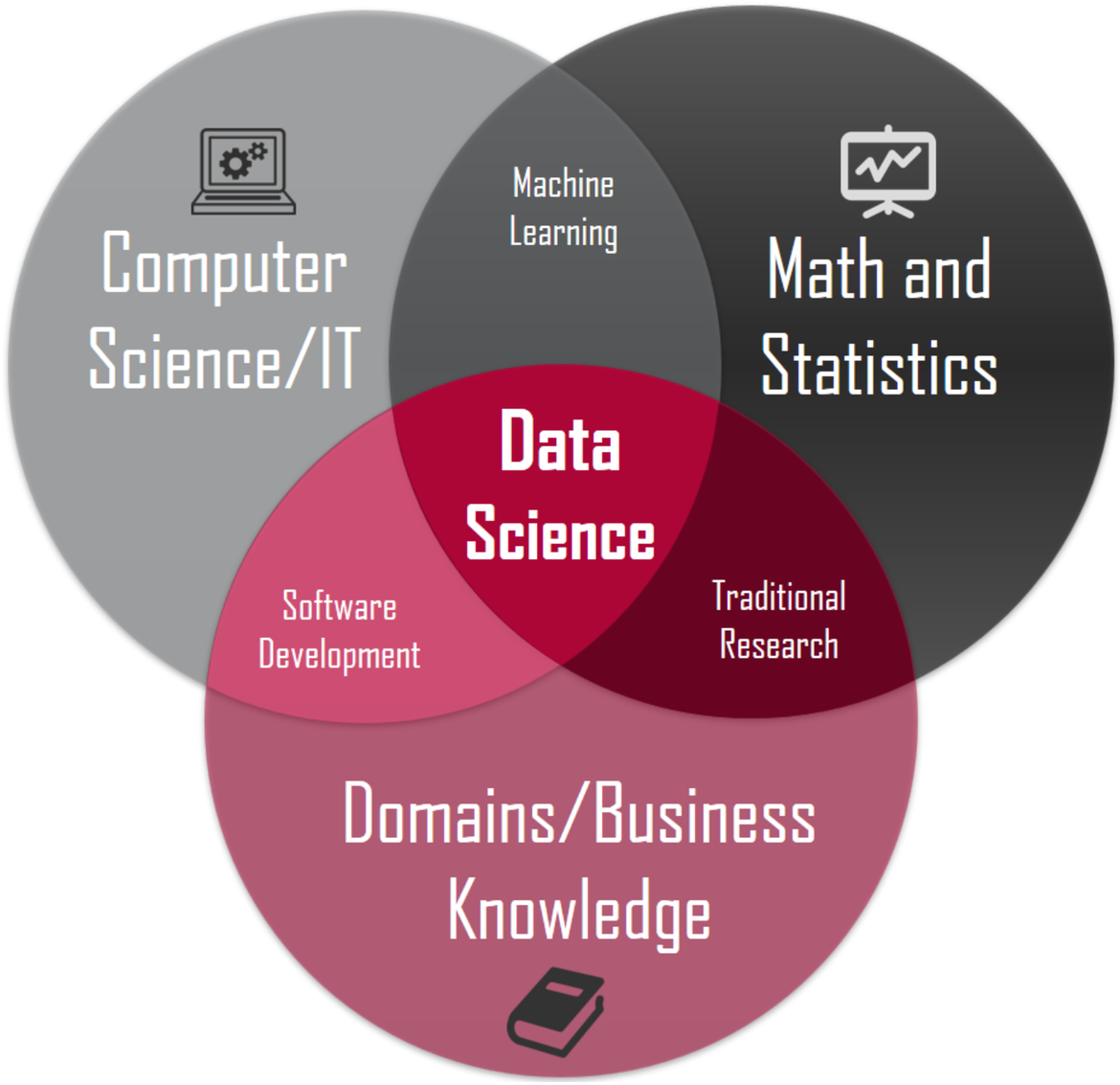

Approximate Bayesian Computation

$$\pi(model|data) = \frac{\pi(data|model)\,\pi(model)}{\pi(data)}$$

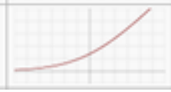
- ABC is applicable when the likelihood $\pi(data|model)$ is intractable/unknown
- The method requires a simulator or surrogate model
- Generate simulated data for models drawn from the prior, accept/reject whether matching data

- Overly expensive in calls to simulator
  - Introduce summary statistics to enhance border cases
  - Efficient sampling to boost acceptable models
  - Generalized methods for comparing simulated samples with data

→ Most relevant work on likelihood-free inference in HEP https://arxiv.org/abs/1805.12244

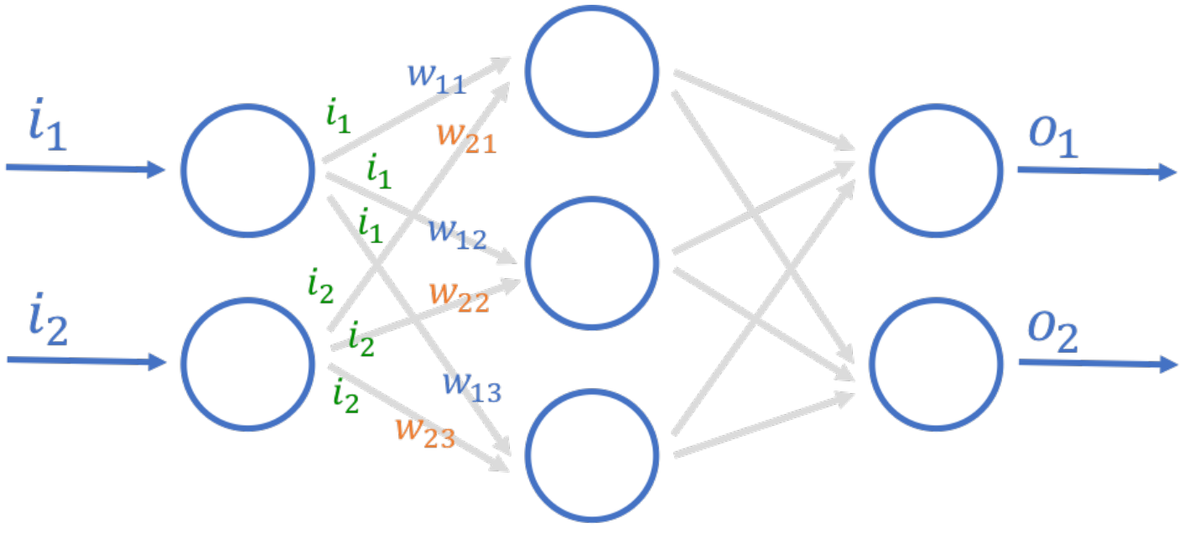


Computer Science/IT
Machine Learning
Math and Statistics
Data Science
Software Development
Traditional Research
Domains/Business Knowledge

Internal Node Activation

Name	Plot	Equation	Derivative
Identity		$f(x) = x$	$f'(x) = 1$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$
Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1+e^{-x}}$	$f'(x) = f(x)(1-f(x))$
TanH		$f(x) = \tanh(x) = \frac{2}{1+e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2+1}$
Rectified Linear Unit (ReLU)		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Parameteric Rectified Linear Unit (PReLU) [2]		$f(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Exponential Linear Unit (ELU) [3]		$f(x) = \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} f(x) + \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
SoftPlus		$f(x) = \log_e(1+e^x)$	$f'(x) = \frac{1}{1+e^{-x}}$

- Any function with a derivative may work
- Many activation to pick from (and there are more, like cos, ...)
- Sigmoid, tanh suffer from vanishing gradients : slow convergence
- Relu and PRelu solve some of the vanishing gradient issue, and accelerate computation


Operation Vectorization

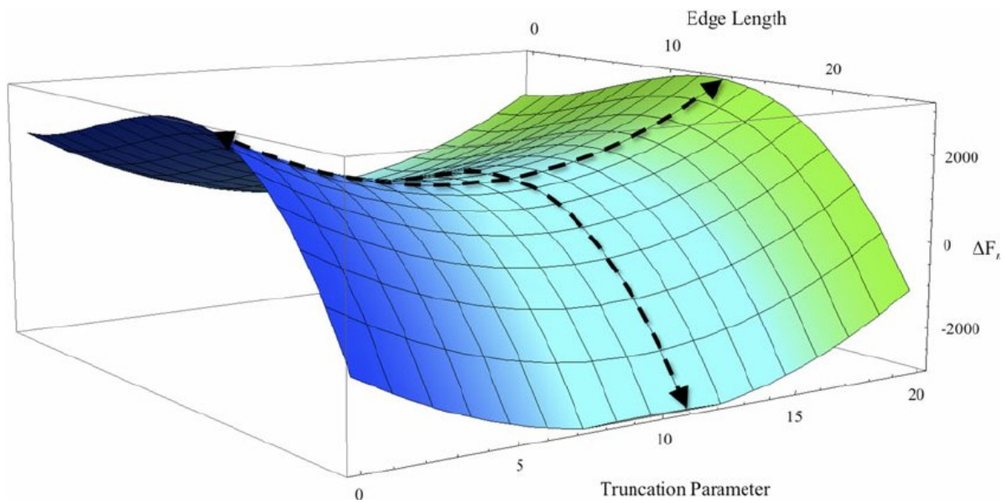


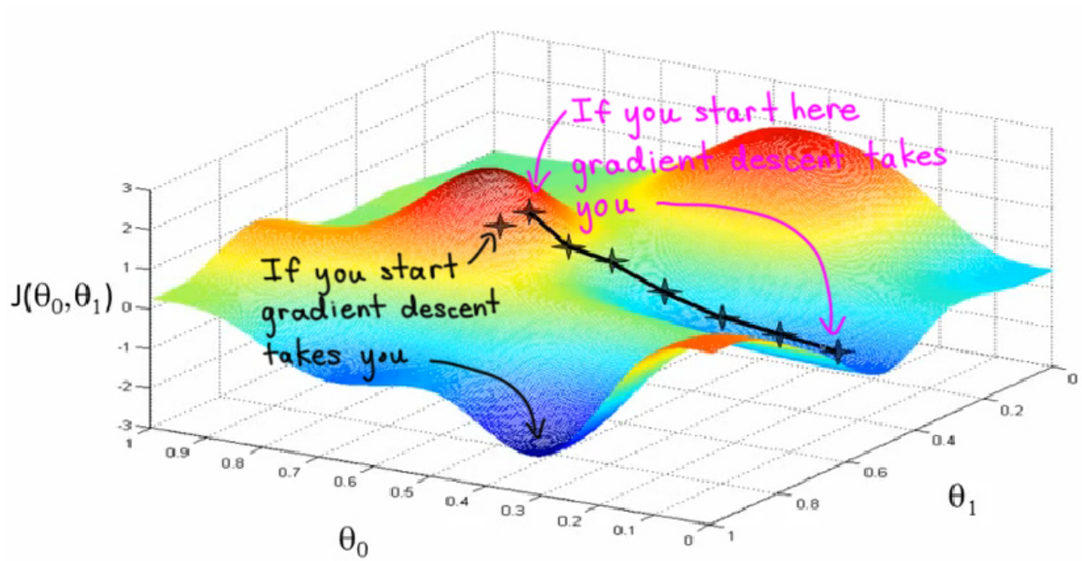
ANN ≡ matrix operations ≡ parallelizable

$$\begin{bmatrix} w_{11} & w_{21} \\ w_{12} & w_{22} \\ w_{13} & w_{23} \end{bmatrix} \cdot \begin{bmatrix} i_1 \\ i_2 \end{bmatrix} = \begin{bmatrix} (w_{11} \times i_1) + (w_{21} \times i_2) \\ (w_{12} \times i_1) + (w_{22} \times i_2) \\ (w_{13} \times i_1) + (w_{23} \times i_2) \end{bmatrix}$$

Computation of prediction from artificial neural network model can be **vectorized to a large extend.**

Non-Convex Optimization



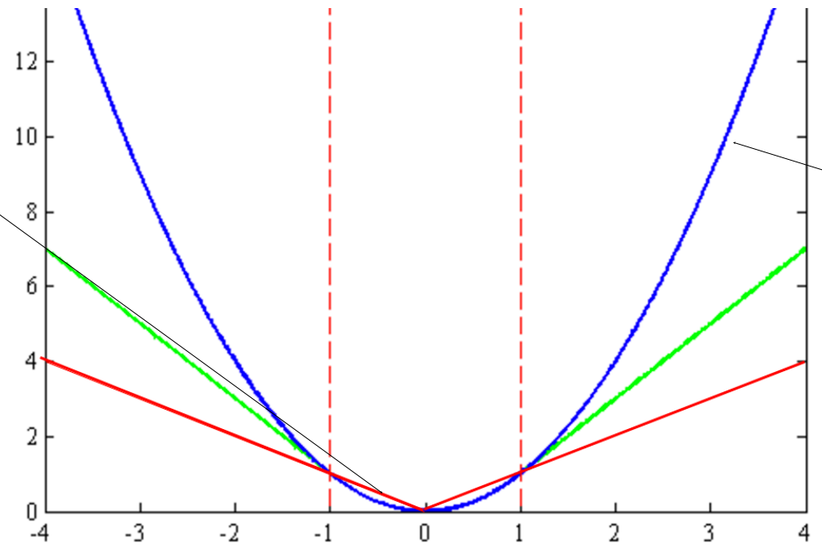


- The objective functions optimized in machine learning are usually non-convex
- Non guaranteed convergence of gradient descent
- Gradients may vanish near local optimum and saddle point

Regularization

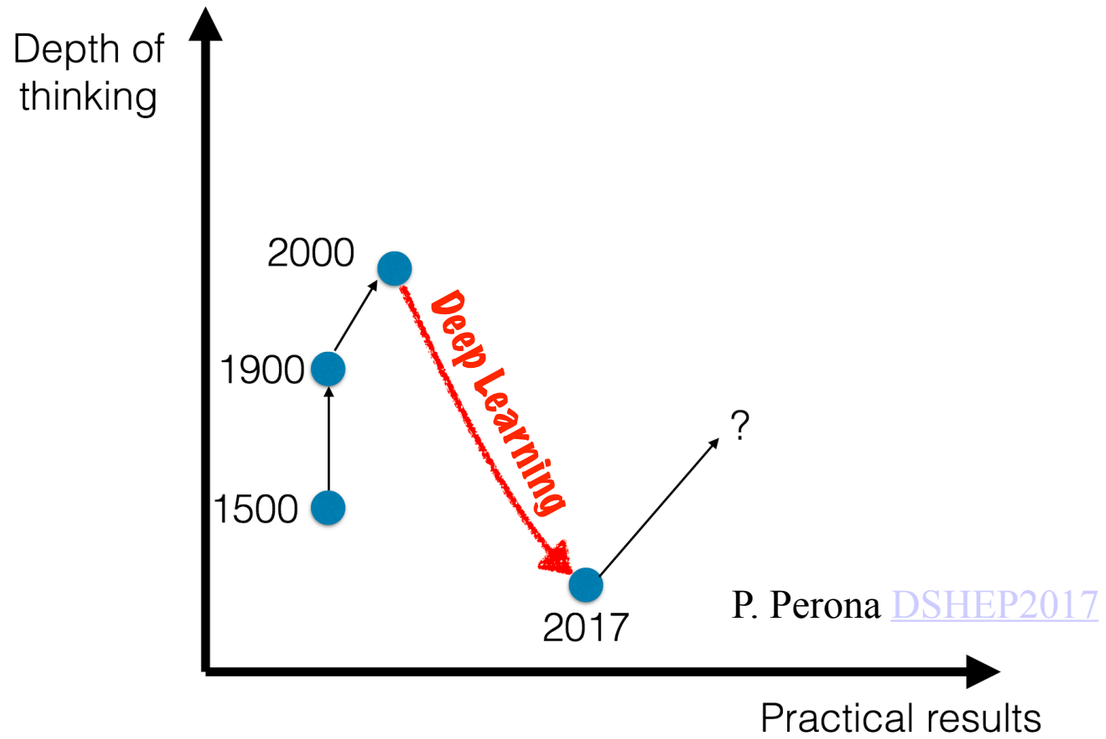
- Over-fitting can be limited by additional terms to the objective function
- Any term that aim at reducing the capacity of the model
  - L1 : $Loss_{reg} = Loss + \lambda_1 \sum |w_i|$
  - L2 : $Loss_{reg} = Loss + \lambda_2 \sum w_i^2$
  - Huber : $Loss_{reg} = Loss + \lambda_h \begin{cases} \frac{1}{2} L_2 & if \ L1 \leq \delta \\ else \ \delta(L_1 - \frac{1}{2}\delta) \end{cases}$
- Regularization parameters need to be optimized too

L1 over-constrains to few parameters



L2 explodes with outliers

The Black-box Dilemma



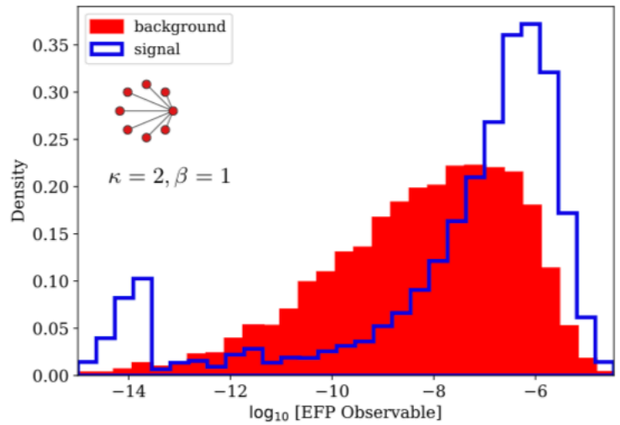
Deep learning may yield great improvements.
Having the “best classification performance” is not always sufficient.
Forming an understand of the processes at play is often crucial.

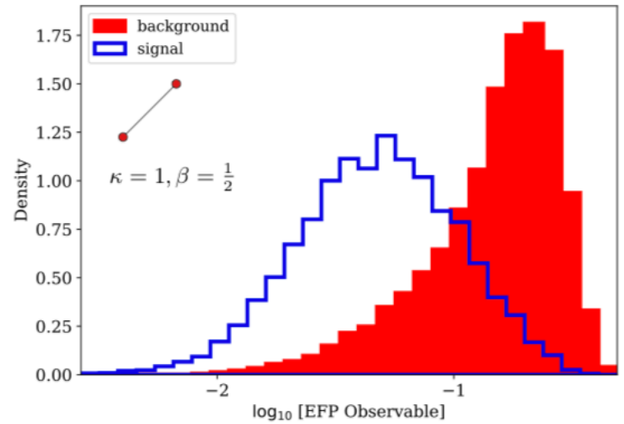
Learning Observables

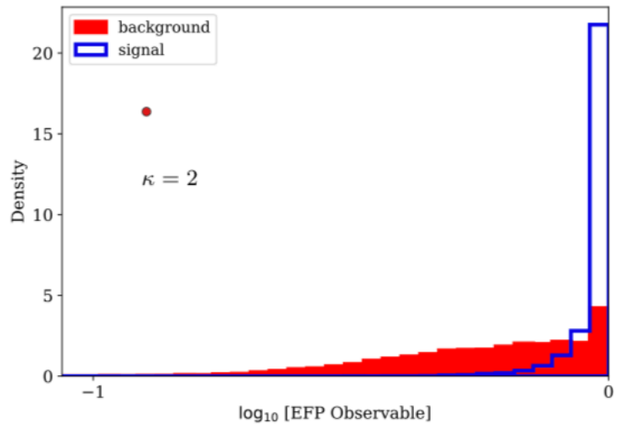
Electron classification performance

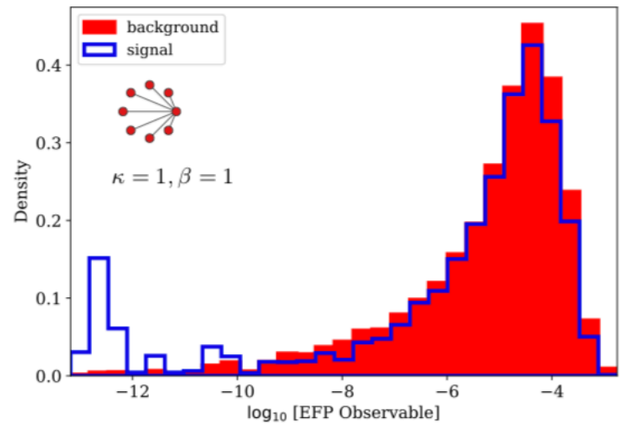
Base		Additions (κ, β)		(AUC)
7HL				0.945
7HL	$+M_{\text{jet}}$			0.956
7HL		$(1, \frac{1}{2})$		0.970
7HL	$+M_{\text{jet}}$	$(1, 1)$	$(1, \frac{1}{2})$	0.971
7HL		$(2, -)$		0.970
7HL	$+M_{\text{jet}}$	$(2, 1)$	$(2, -)$	0.971
CNN				0.972

https://arxiv.org/abs/2010.11998
https://arxiv.org/abs/2011.01984

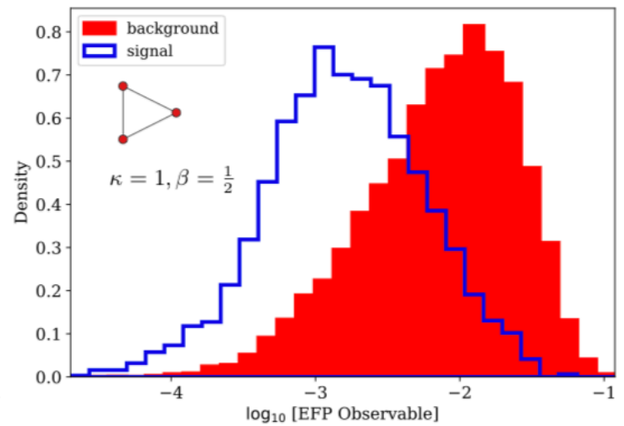




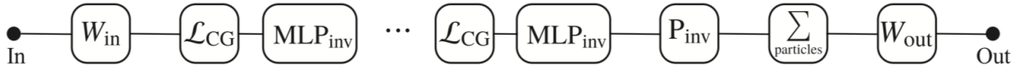




Search in the space of functions using decision ordering.
Simplified to the energy flow polynomial subspace.
Extract set of EFP that matches DNN performance.

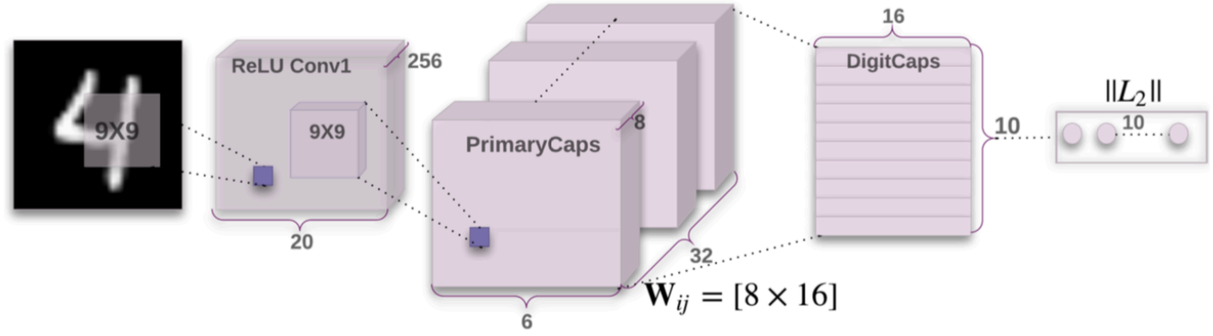


Equivariant Derivatives



$$\mathscr{F}_i \mapsto W \cdot \left(\mathscr{F}_i \oplus \mathscr{F}_i^{\otimes 2} \oplus \sum_j f\left(p_{ij}^2\right) \cdot p_{ij} \otimes \mathscr{F}_j \right)$$

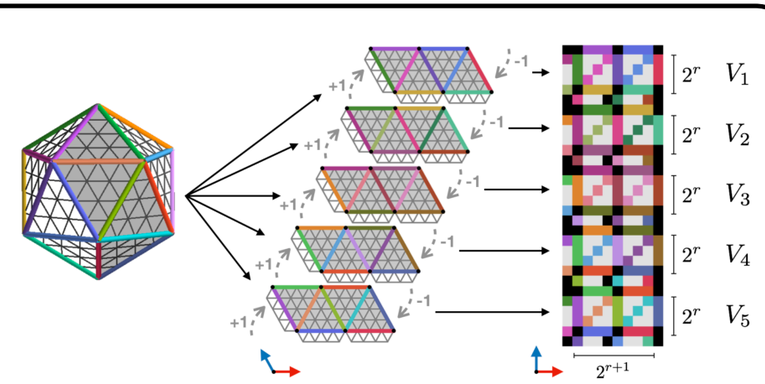
Lorentz group Equivariant networks [2006.04780]



Dynamic Routing Between Capsules [1710.09829]

	GNN	Radial Field	TFN	Schnet	EGNN
Edge	$\mathbf{m}_{ij} = \phi_e(\mathbf{h}_i^l, \mathbf{h}_j^l, a_{ij})$	$\mathbf{m}_{ij} = \phi_{\text{rf}}(\|\mathbf{r}_{ij}^l\|)\mathbf{r}_{ij}^l$	$\mathbf{m}_{ij} = \sum_k \mathbf{W}^{lk}\mathbf{r}_{ji}^l\mathbf{h}_i^{lk}$	$\mathbf{m}_{ij} = \phi_{\text{cf}}(\|\mathbf{r}_{ij}^l\|)\phi_{\text{s}}(\mathbf{h}_j^l)$	$\mathbf{m}_{ij} = \phi_e(\mathbf{h}_i^l, \mathbf{h}_j^l, \|\mathbf{r}_{ij}^l\|^2, a_{ij})$ $\hat{\mathbf{m}}_{ij} = \mathbf{r}_{ij}^l\phi_x(\mathbf{m}_{ij})$
Agg.	$\mathbf{m}_i = \sum_{j\in\mathcal{N}(i)} \mathbf{m}_{ij}$	$\mathbf{m}_i = \sum_{j\neq i} \mathbf{m}_{ij}$	$\mathbf{m}_i = \sum_{j\neq i} \mathbf{m}_{ij}$	$\mathbf{m}_i = \sum_{j\neq i} \mathbf{m}_{ij}$	$\mathbf{m}_i = \sum_{j\in\mathcal{N}(i)} \mathbf{m}_{ij}$ $\hat{\mathbf{m}}_i = \sum_{j\neq i} \hat{\mathbf{m}}_{ij}$
Node	$\mathbf{h}_i^{l+1} = \phi_h(\mathbf{h}_i^l, \mathbf{m}_i)$	$\mathbf{x}_i^{l+1} = \mathbf{x}_i^l + \mathbf{m}_i$	$\mathbf{h}_i^{l+1} = w^{ll}\mathbf{h}_i^l + \mathbf{m}_i$	$\mathbf{h}_i^{l+1} = \phi_h(\mathbf{h}_i^l, \mathbf{m}_i)$	$\mathbf{h}_i^{l+1} = \phi_h\left(\mathbf{h}_i^l, \mathbf{m}_i\right)$ $\mathbf{x}_i^{l+1} = \mathbf{x}_i^l + \hat{\mathbf{m}}_i$
	Non-equivariant	E(n)-Equivariant	SE(3)-Equivariant	E(n)-Invariant	E(n)-Equivariant

E(n) Equivariant Graph Neural Networks [2102.09844]



Gauge Equivariant Convolutional Networks and the Icosahedral CNN [1902.04615]