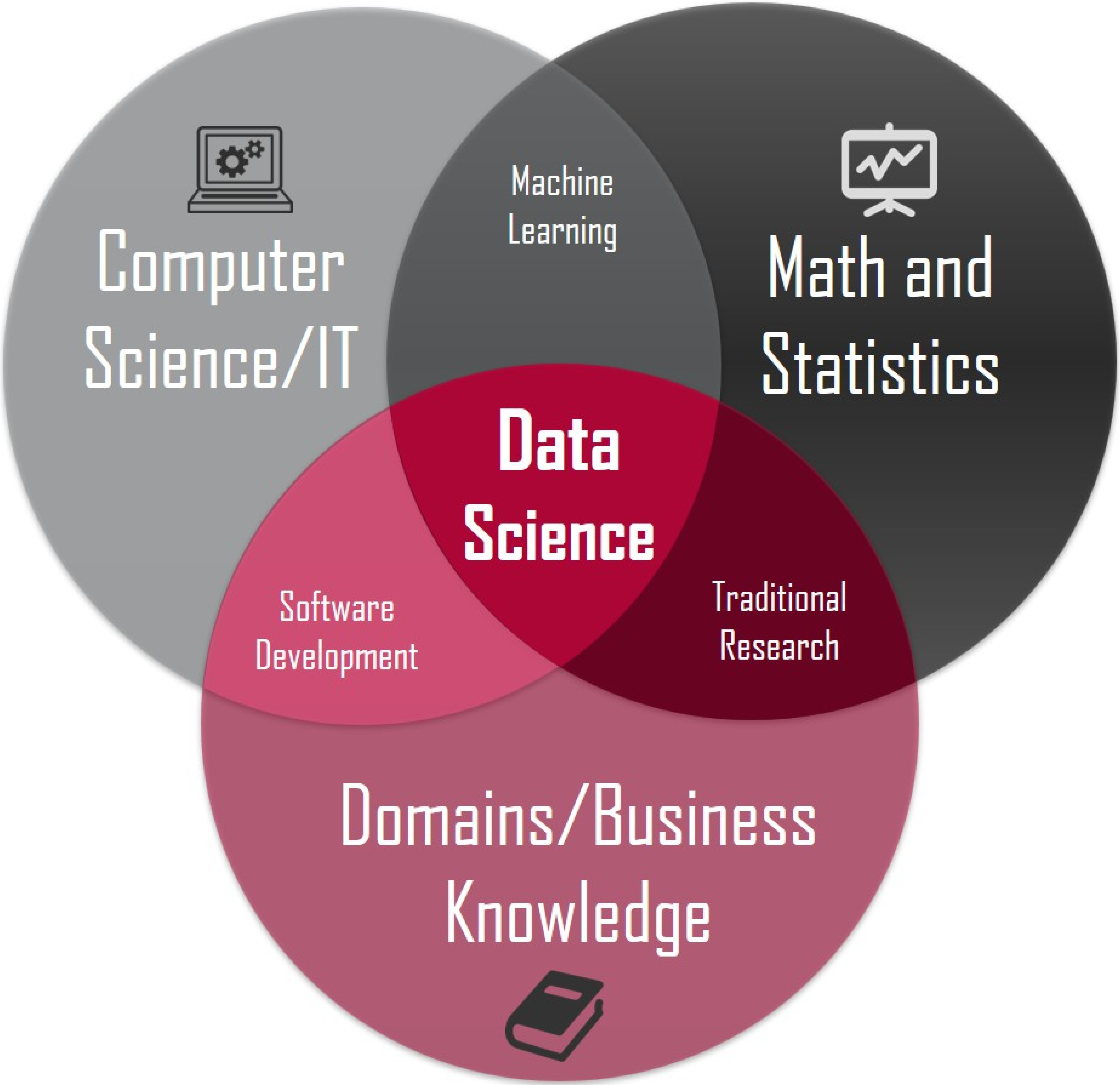
Summary/Recap Lecture 2

- Unsupervised learning for clustering, dimensionality reduction, density estimation and generative models
- Supervised learning for regression and classification
- Artificial neural network are in rapid evolution. Methods providing lots of flexibility and at the forefront of performance on many complex tasks


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

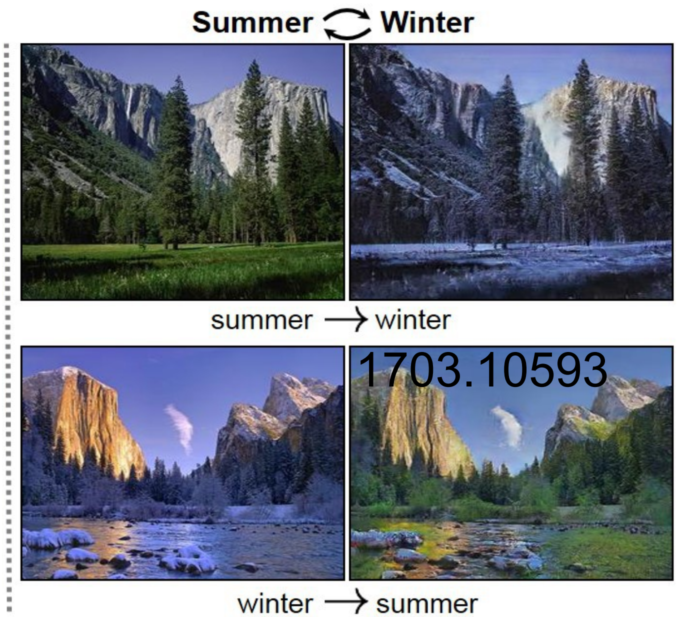
Lecture - Part 3/3

Cutting Edge Technique : Outline

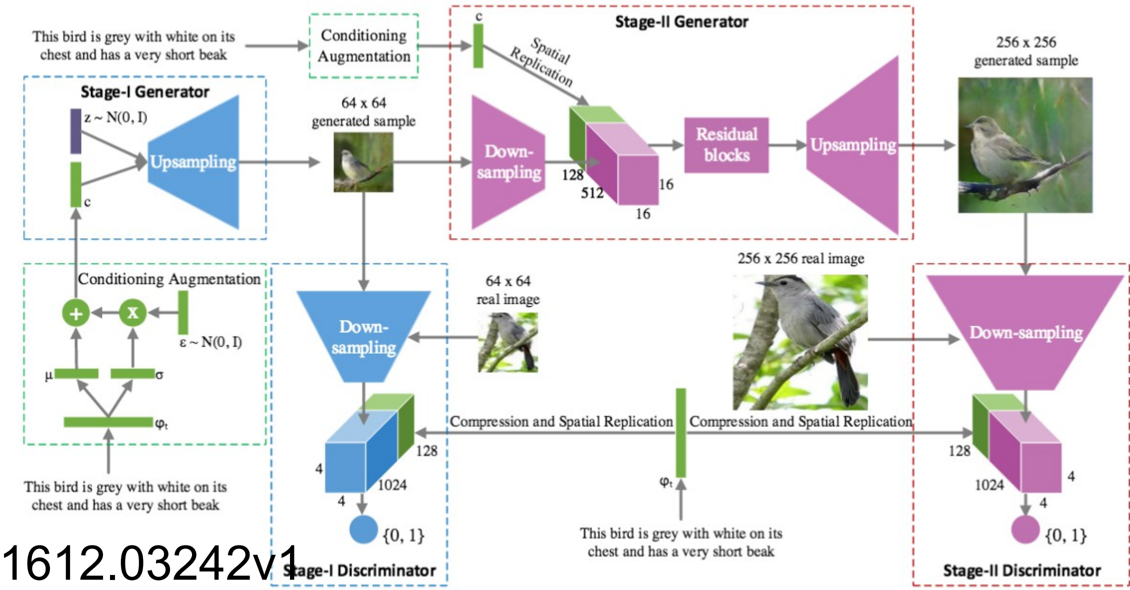
- Generative Models
- Nuisance parameters
- Graph Networks
- Information Representation
- Control Learning
- Neuromorphic Computing
- Quantum Machine Learning

Generative Models

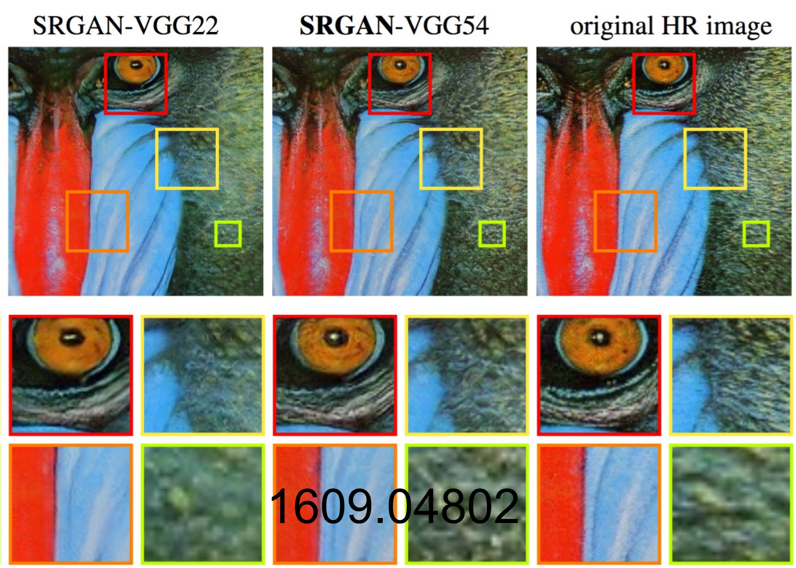
(Generative) Adversarial Models



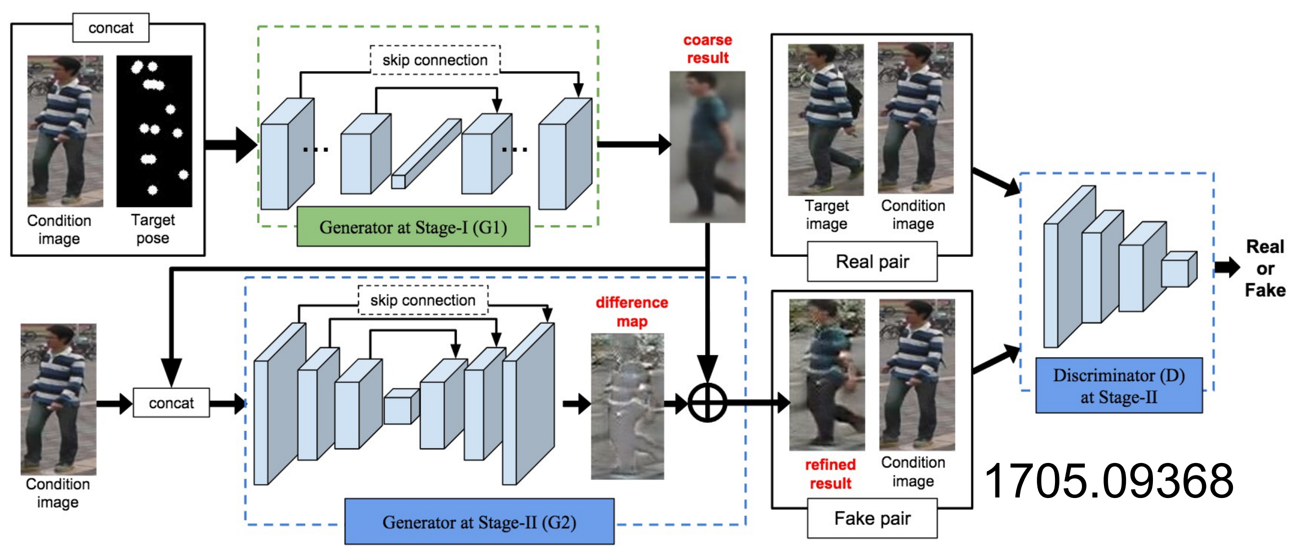
1703.10593



1612.03242v1

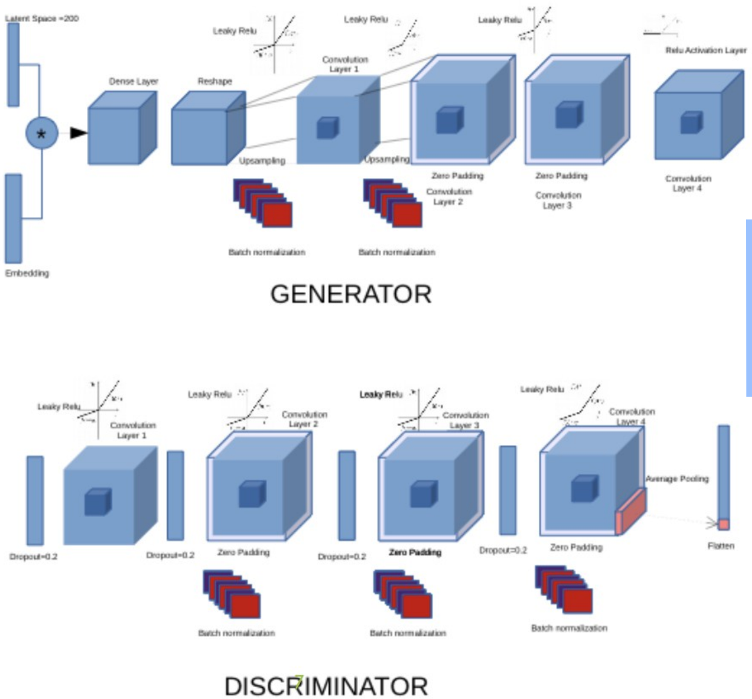


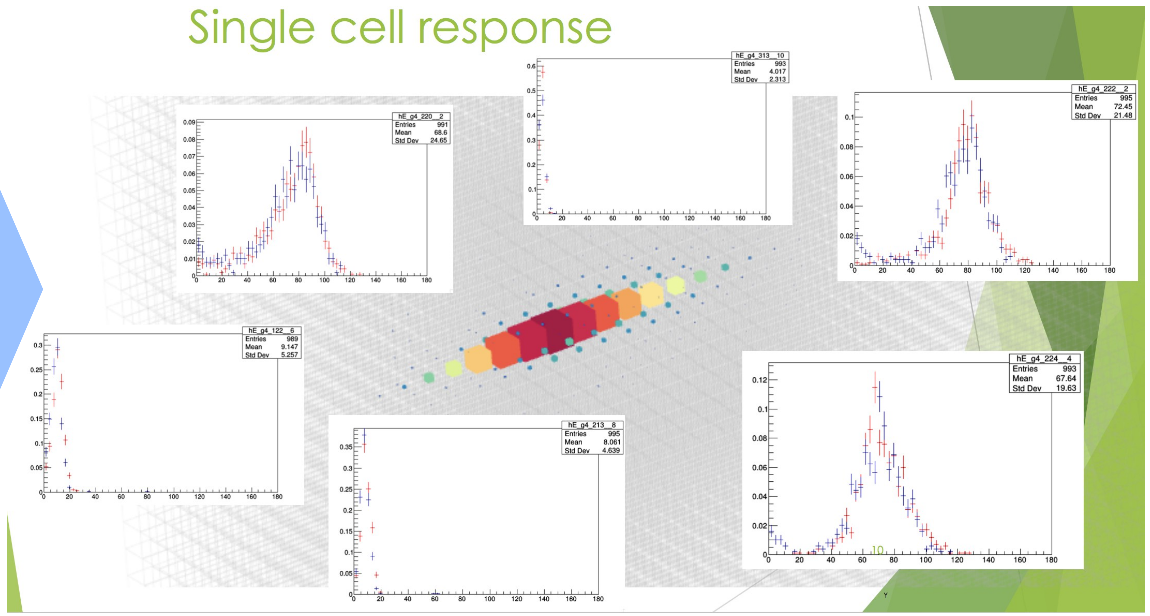
1609.04802



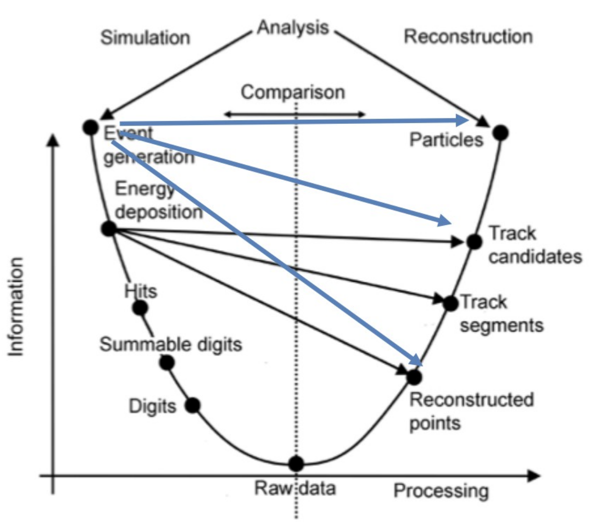
1705.09368

3D GAN

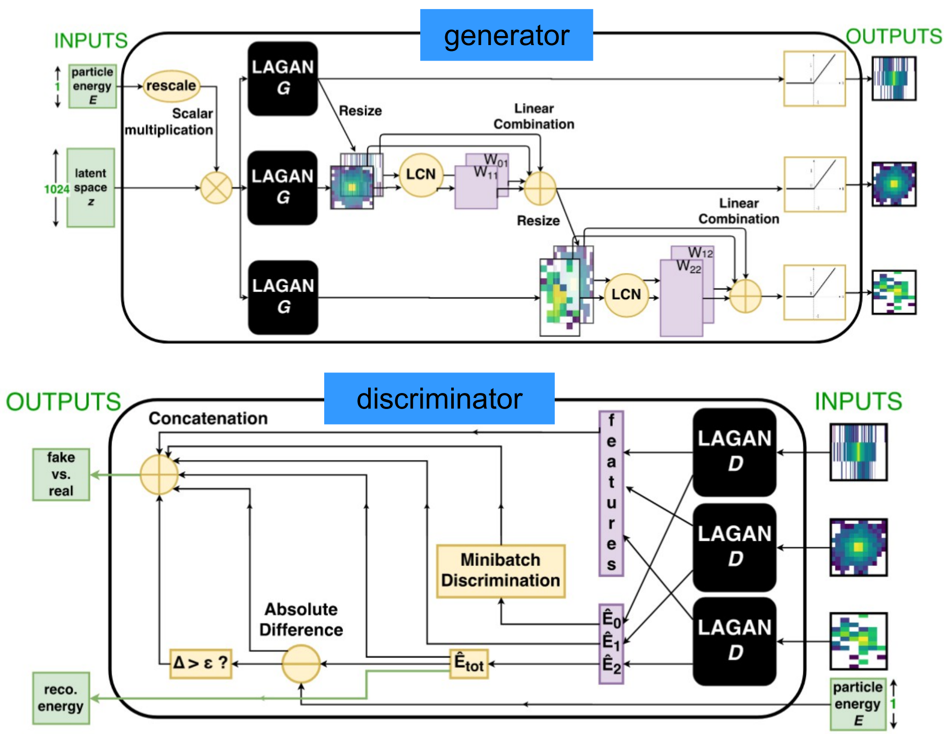




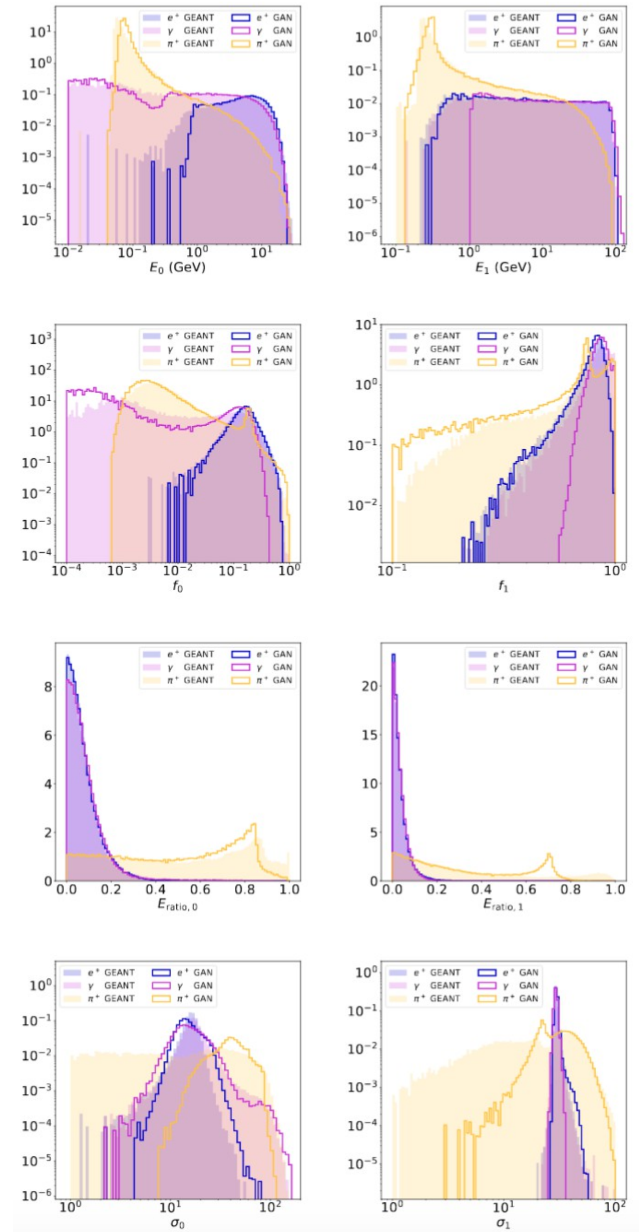
See S. Vallecorsa's seminar for details



Calo GAN



Calogan 1712.10321



- Model conditioned on energy
- Successive layers conditioned on previous ones
- Fair agreement over shower shape variables
- Tremendous speed up over generation

NADE

- Neural Auto-Regressive Density Estimator (NADE) is a family of models for learning the pdf of the input dataset
- Relies on the the probability chain rule
- Modeling conditional probabilities as a mixture model (e.g Gaussian)

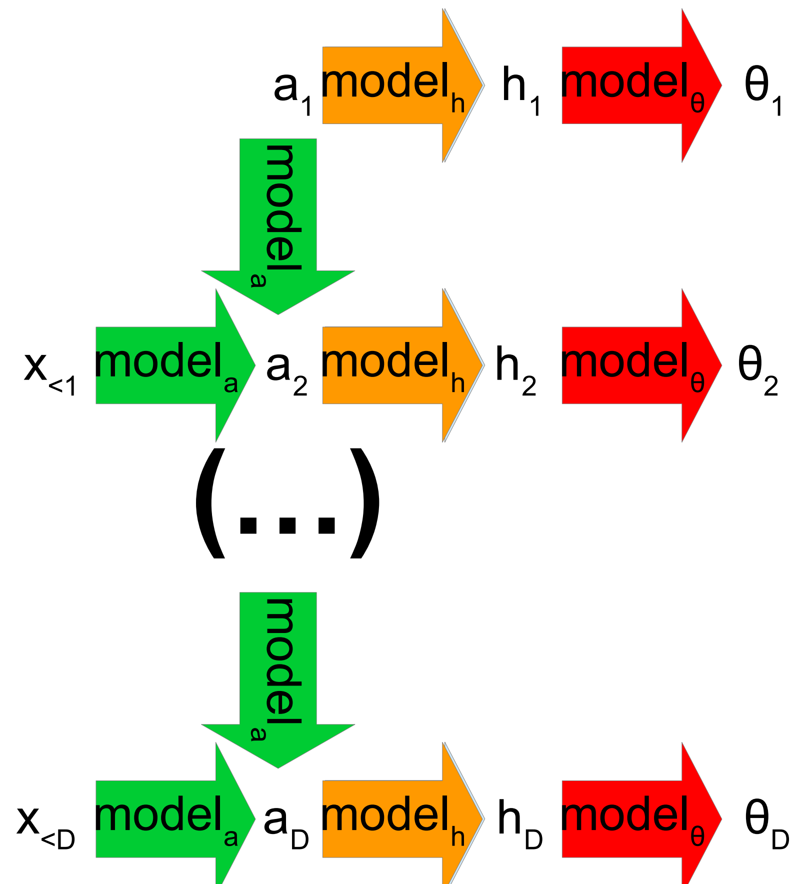
$$x \equiv (x_{1,} x_{2,} \ldots , x_D) ; P(x) = \prod_{d=1}^{D} p(x_d | x_{<d})$$

$$p(x_d | x_{<d}) \equiv M(x_d , \theta_d) \quad where \quad \theta_d \equiv f(x_{<d})$$

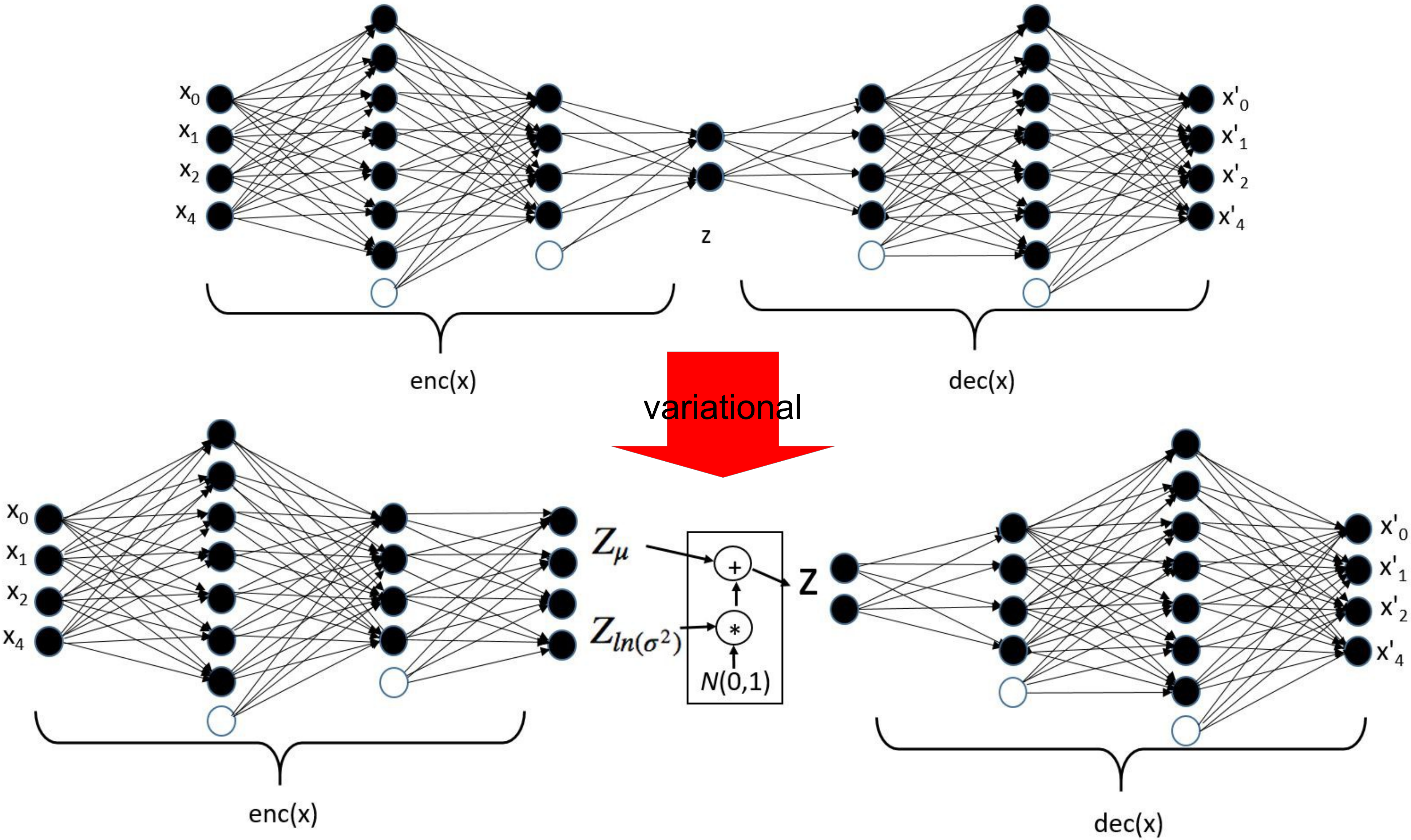
$$Loss(x) \equiv -\log P(x)$$

$$Loss = -\frac{1}{N} \sum_i \sum_d M(x_{i,d} , \theta_d)$$

NADE 1605.02226



Variational Auto-Encoder

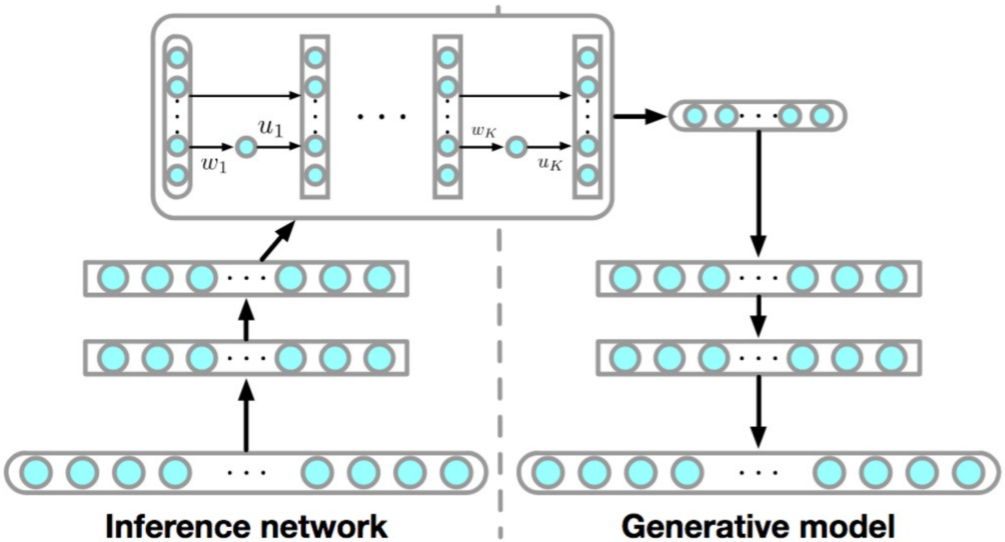


Normalizing Flow

- Variational model with normalizing flow are very similar to variational auto-encoder, in which the latent variable distribution is approximated by normalizing flow
- Normalizing flow is a technique to evolve a probability distribution through a sequence of invertible transformations

$$loss(x_i) = -D_{KL}\left[q(z|x_i) \| Gauss(0,1)\right] + E_{q(z|x_i)}\left[\log p(x_i|z)\right]$$

$$q(z|x_i) \sim \underset{t}{\circ} f_k(z) | x_i$$



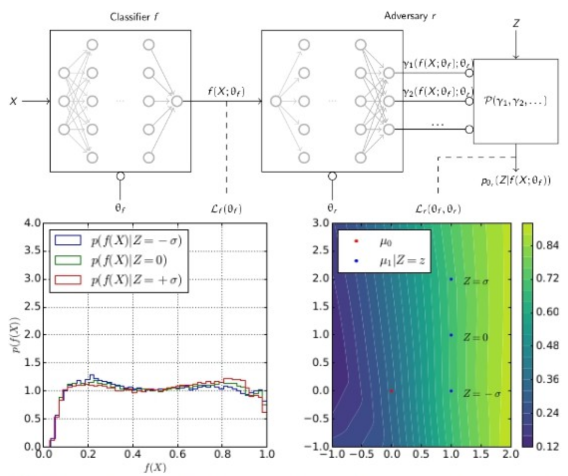
Norm-flow VAE 1505.05770

Systematic Uncertainties

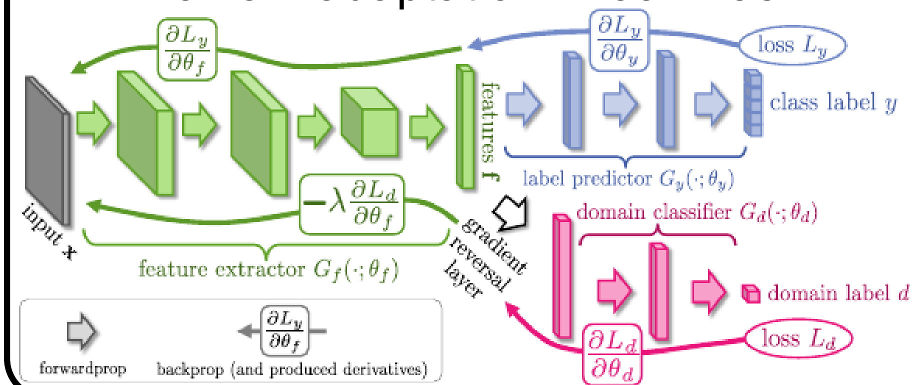
Learn Against Nuisance

- Several proposed method to combat the impact of certain source of systematic uncertainties on the model performance

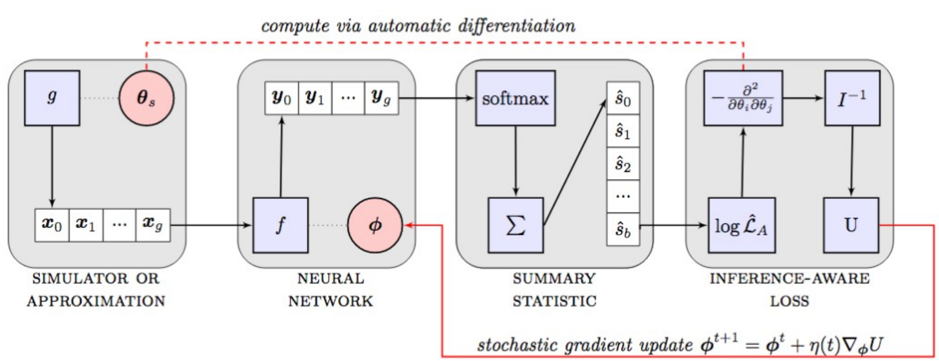
Learn to pivot 1611.01046



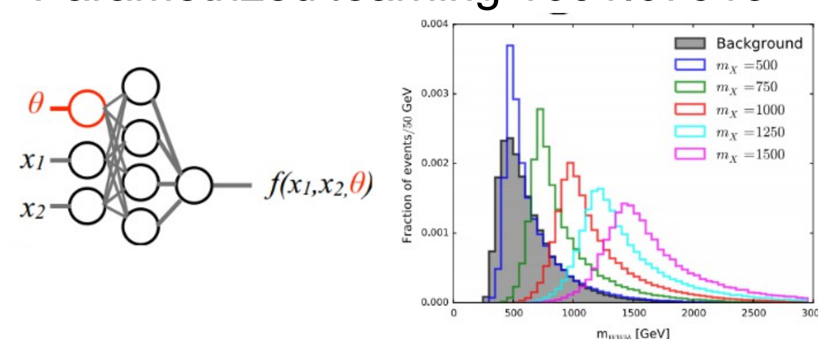
Domain adaptation 1409.7495



inferno 1806.04743

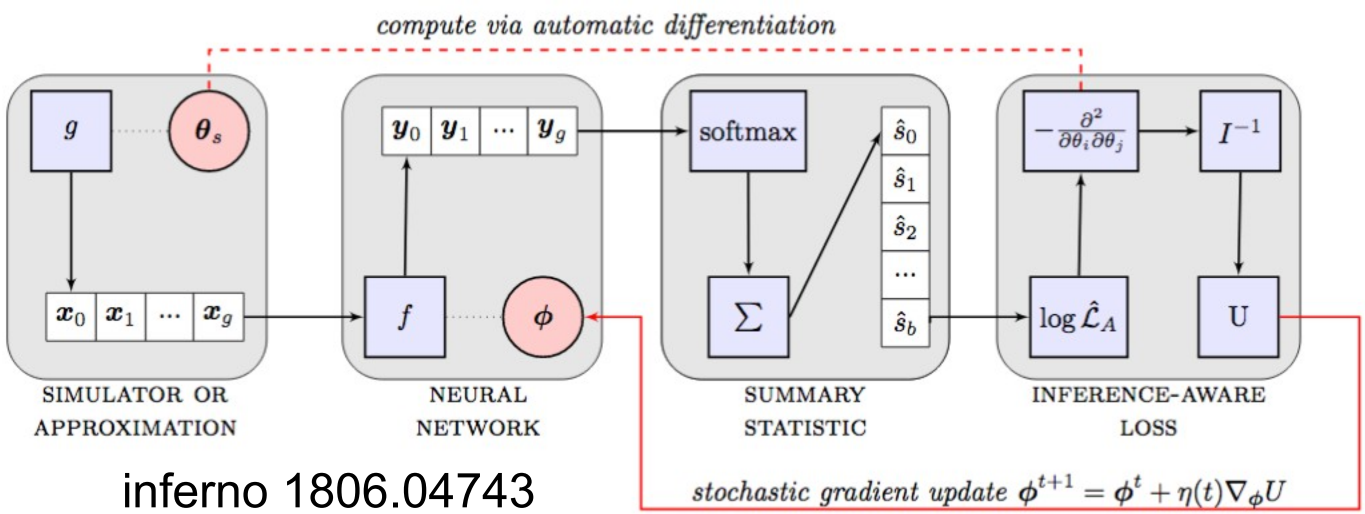


Parametrized learning 1601.07913



Inference Aware Training

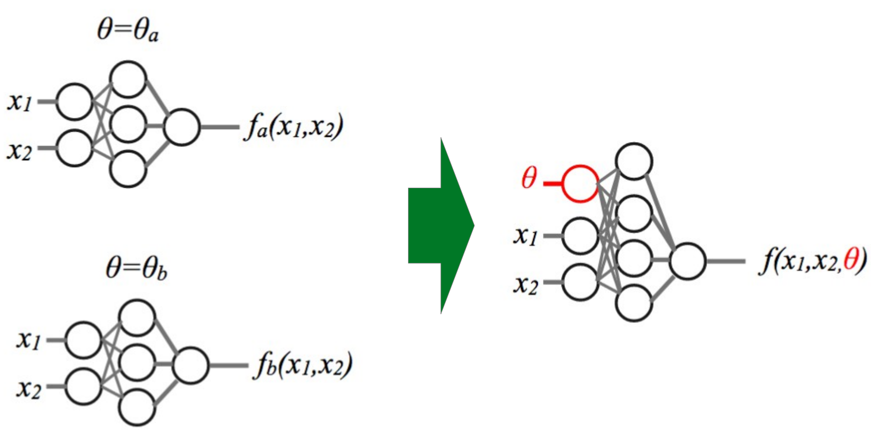
- The classification model derived for an analysis is often subject to systematic uncertainty due to physics parameters model
- In cases where the simulation is differentiable with respect to that parameter : model can be made more robust against such uncertainties

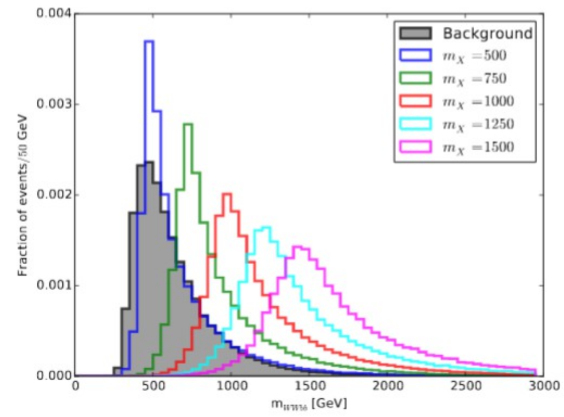


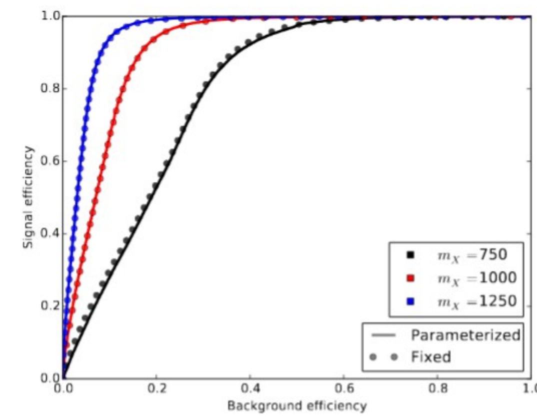
inferno 1806.04743

Parametrized Learning

- Often confronted with signal samples over a parameter scan (mass of a particle, coupling, …)
- Training a model for each sample or a mixture of all samples is not optimal
- Parametrized learning uses the parameter as an additional input
- Model exhibit good interpolation properties
- Can be marginalized later-on





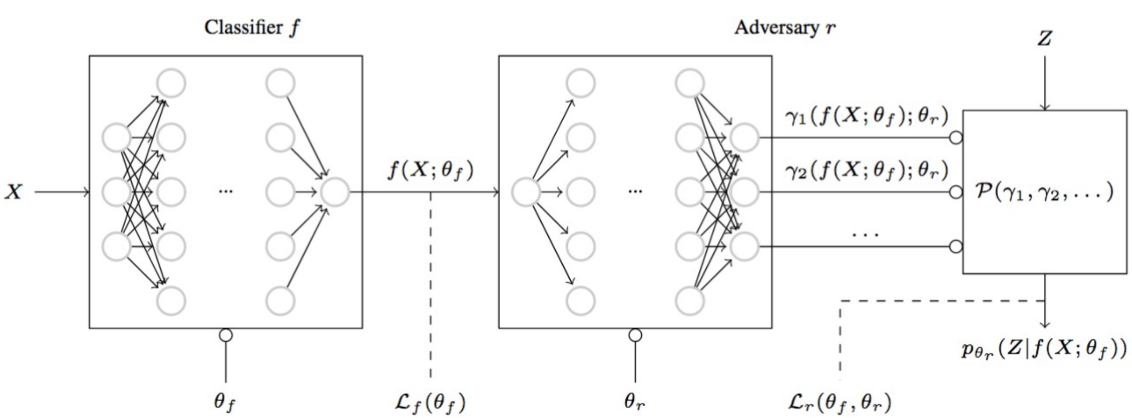


Parametrized learning 1601.07913

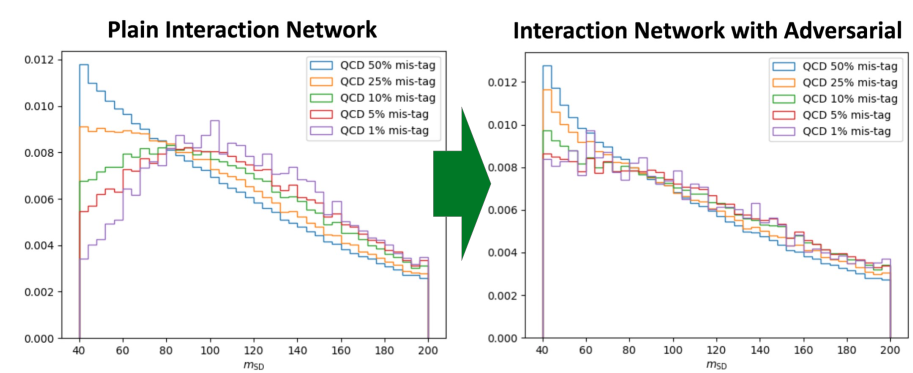
Adversarial Training

- Model can develop internal representation related to physical quantities and use this to perform the classification
- This bias toward the physical quantity might be damaging in sub-sequent data analysis
- Addition of an adversarial network helps in reducing the bias

$$L(x_i, y_i) \rightarrow L(x_i, y_i) - \lambda L_{adversary}(x_i, y_i)$$

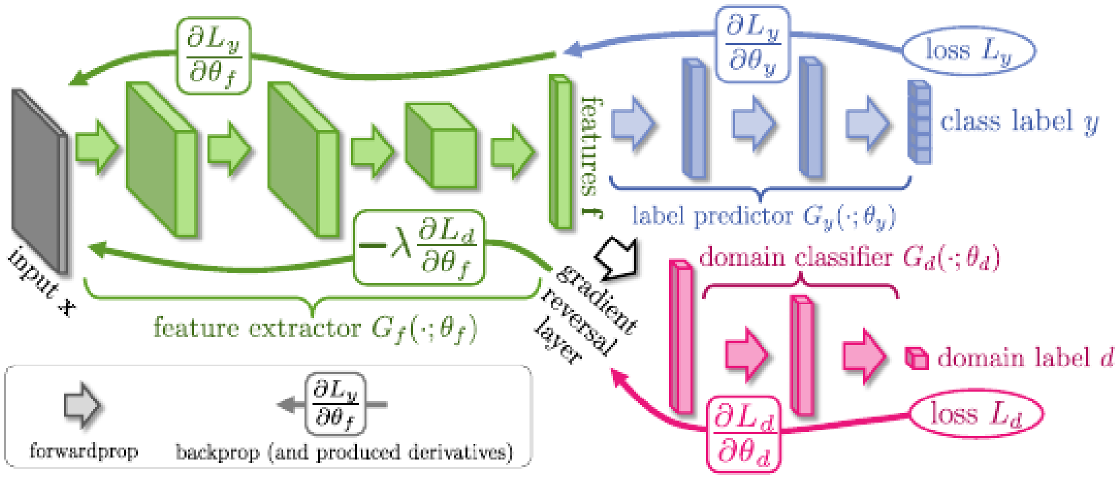


Learn to pivot 1611.01046



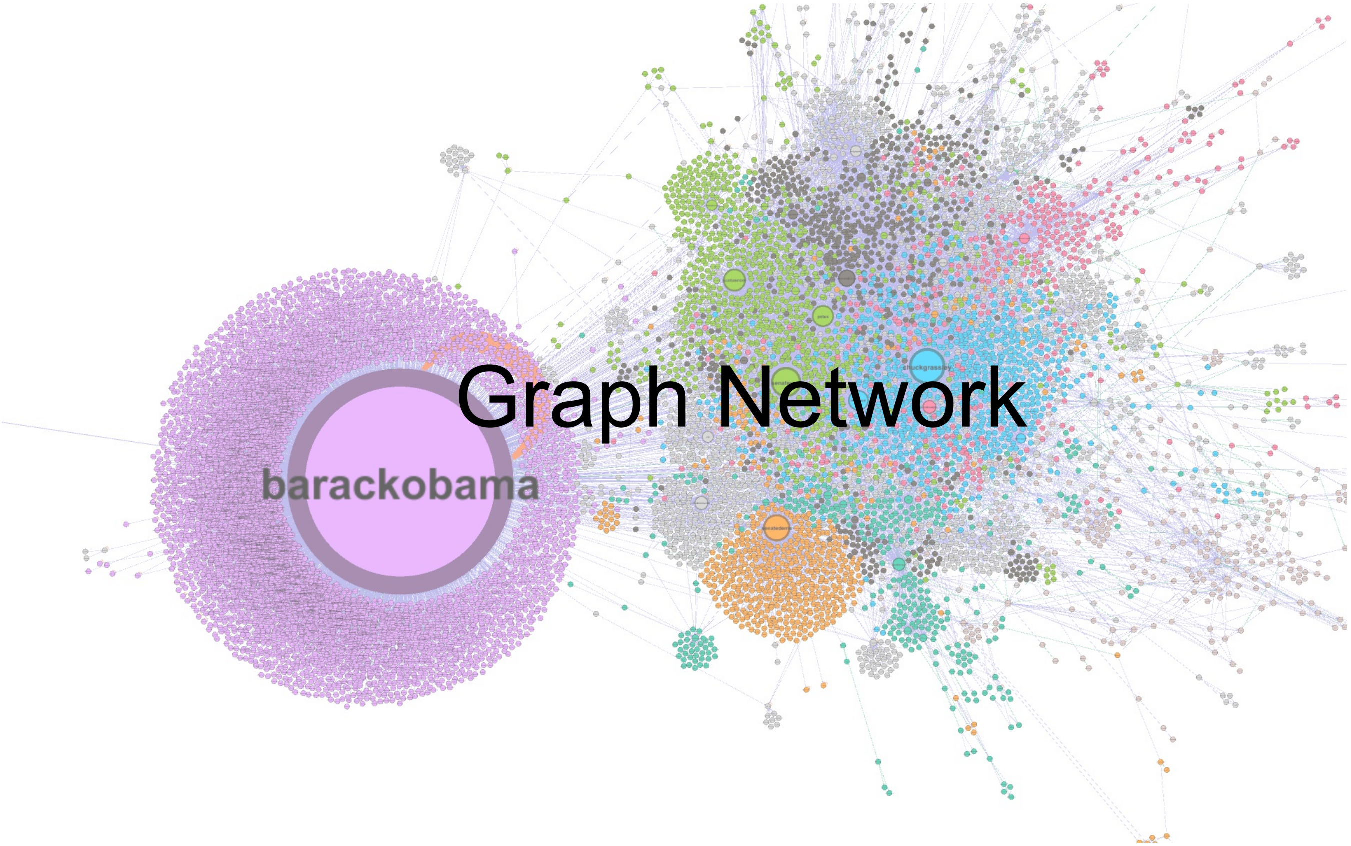
Gradient Reversal

- Demonstrated in the context of domain adaptation
  - Labelled training set is available
  - Unlabelled dataset, from different environment to be classified
- Final model performs classification over the unlabelled dataset
- Labelled : simulation, unlabelled : data



Domain adaptation 1409.7495

Graph Network



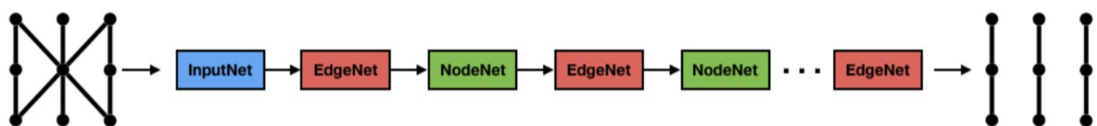
Graph Neural Network

- Relational data can be represented on a directed/undirected graph
- Operations on graph can mostly be represented with matrix operations
- Advantage over sequential representation when relation is not linear
- Adapted to social network analysis
- Dimensionality can be an obstacle
- Field of deep learning in development
  https://github.com/deepmind/graph_nets

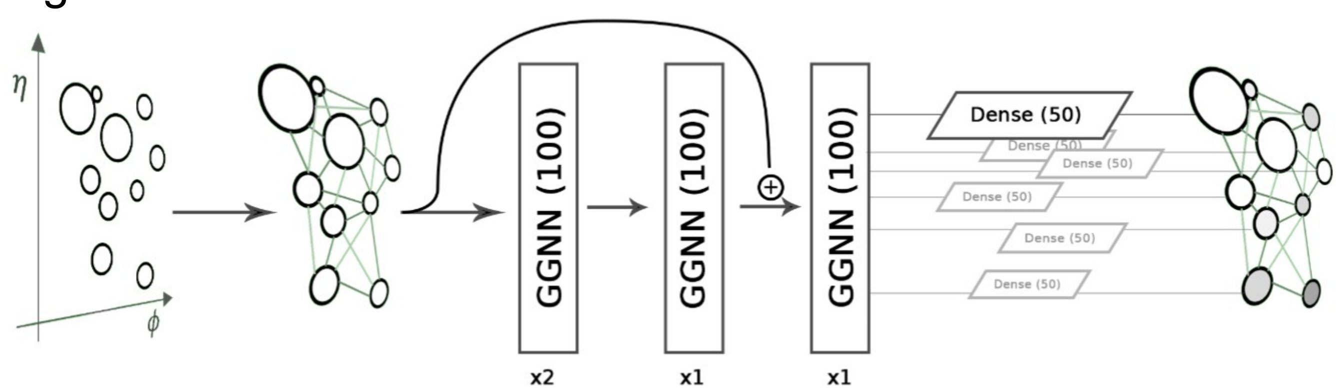
$$x \rightarrow (x, A)$$
$$s.t. \quad A_{ij} = 1 \text{ if i connects to j}$$

Graph Network

- Model node and edge with internal representation
- Learn edge representation
- Propagate information through the graph (message passing) back and forth between edges and nodes
- Iterate the procedure to distill information
- Extract information relevant to the problem, per edge or per node

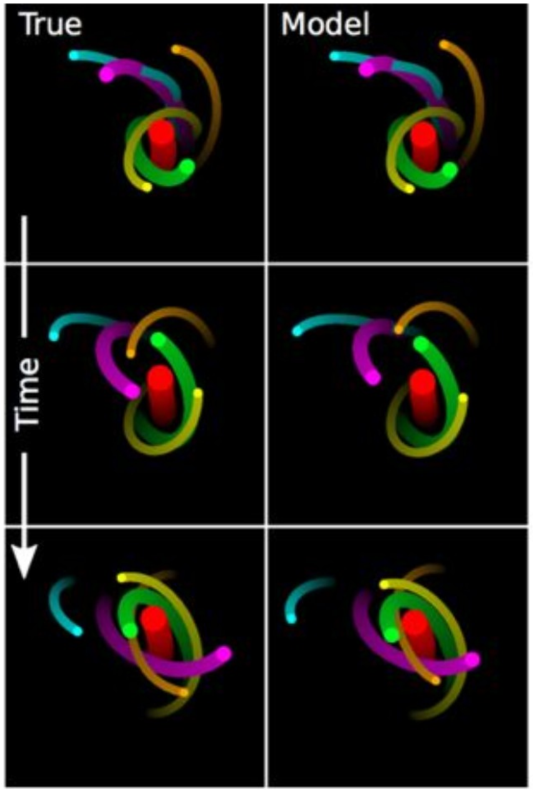


Tracking with gnn 1810.06111



Pile-up with gnn 1810.07988

Interaction Network



- Graph $G = \langle O, R \rangle$, objects connected by relations
- Interaction Network

$$\phi_O(a(G,\ X,\ \phi_R(\ m(G)\ )\ ))$$

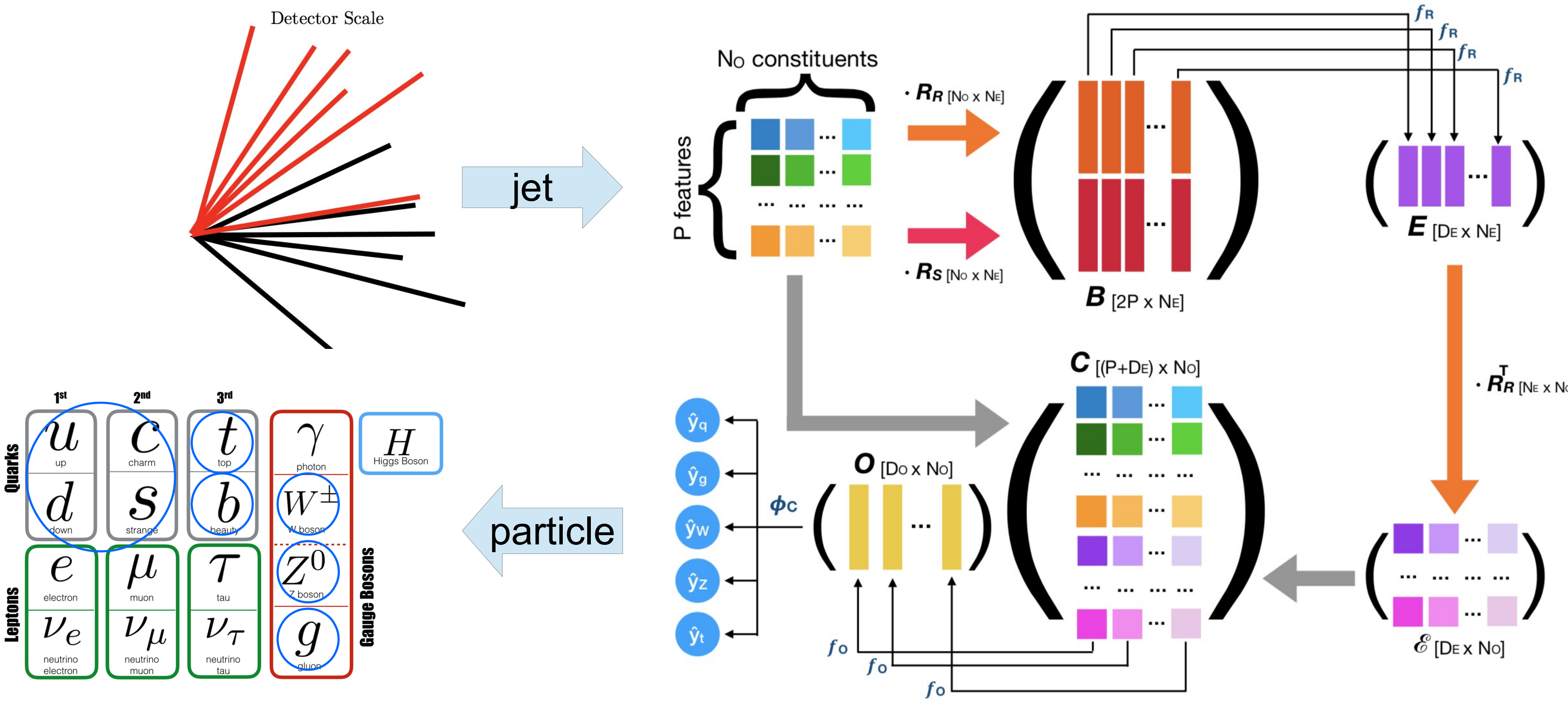
$$\begin{aligned} m(G) &= B = \{b_k\}_{k=1\ldots N_R} \\ f_R(b_k) &= e_k \\ \phi_R(B) &= E = \{e_k\}_{k=1\ldots N_R} \end{aligned} \qquad \begin{aligned} a(G,X,E) &= C = \{c_j\}_{j=1\ldots N_O} \\ f_O(c_j) &= p_j \\ \phi_O(C) &= P = \{p_j\}_{j=1\ldots N_O} \end{aligned}$$

- ϕ_R predicts relational effects
- ϕ_O predicts effect on objects
- Allows for longer-range interactions than a standard CNN

Interaction Networks for Learning about Objects, Relations and Physics
P. W. Battaglia, R. Pascanu, M. Lai, D. Rezende, K. Kavukcuoglu
https://arxiv.org/abs/1612.00222

- Learning the relation between particles (gravity, spring, wall, …)
- Learn the dynamics of the system and predict future evolution

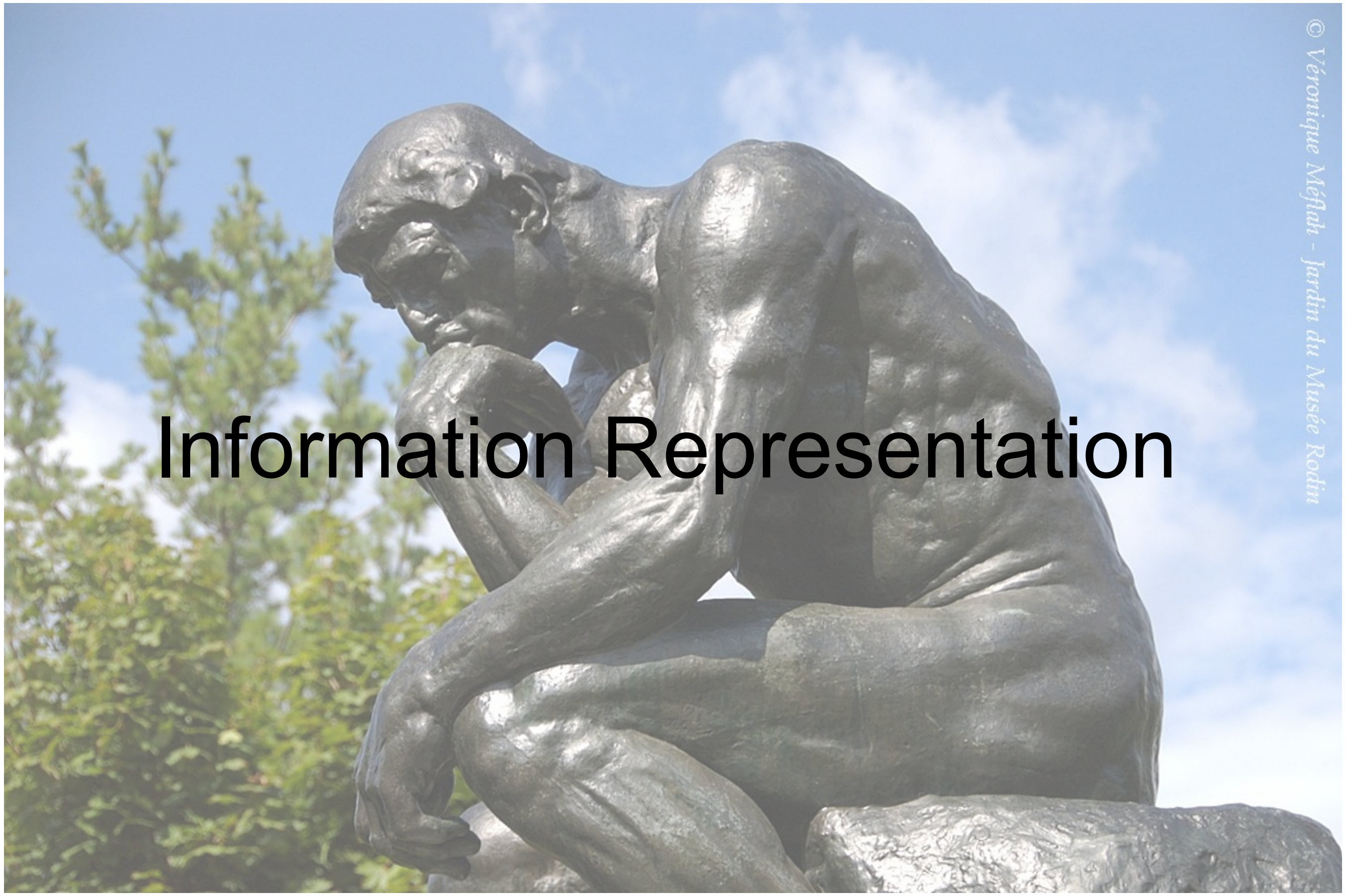
Interaction Network For Jet Id



N_O: # of constituents
P: # of features
$N_E = N_O(N_O-1)$: # of edges
D_E: size of internal representations
D_O: size of post-interaction internal representation

ϕ_C, f_O, f_R parameterized as neural networks

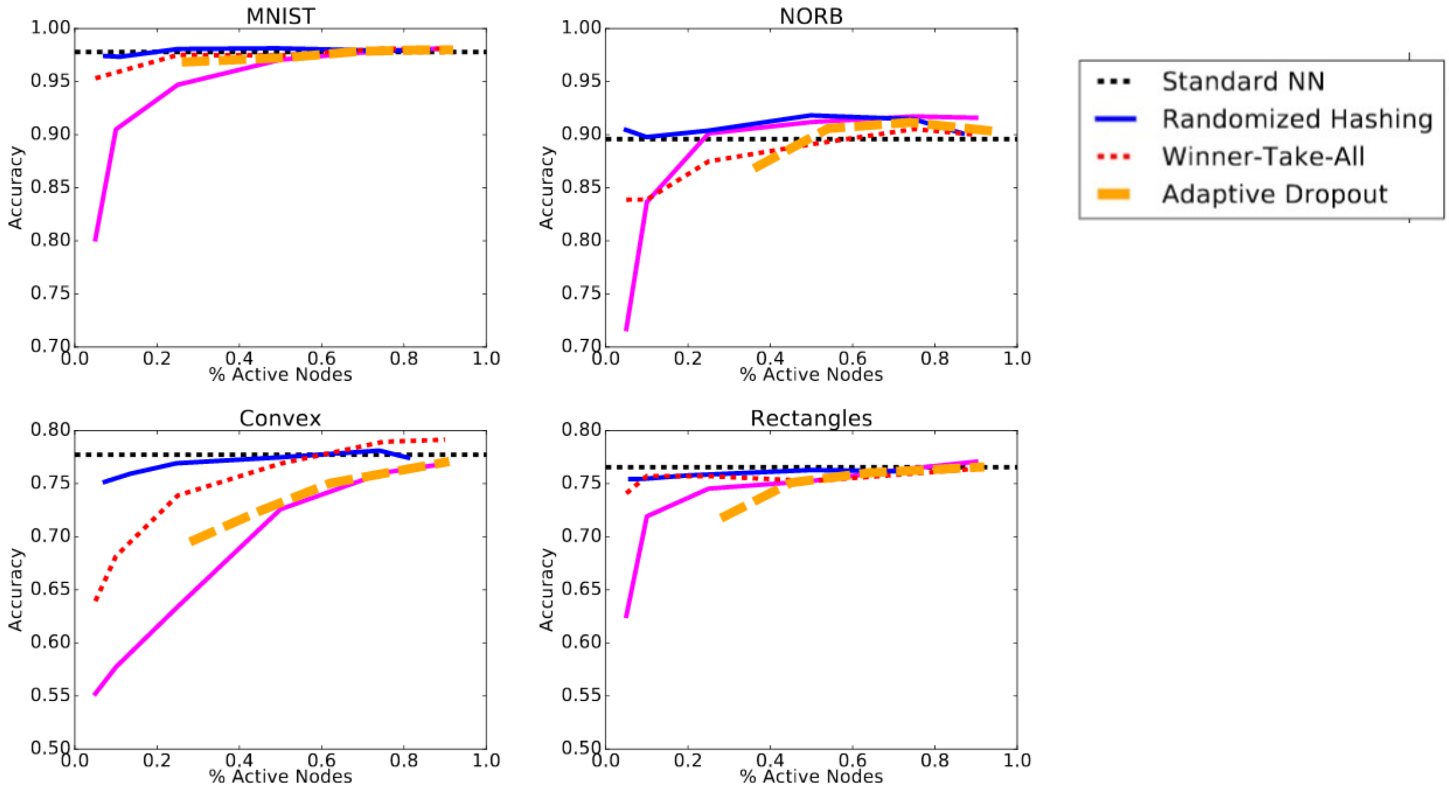
Inspired from https://arxiv.org/abs/1612.00222



Information Representation

Probabilistic Hashing

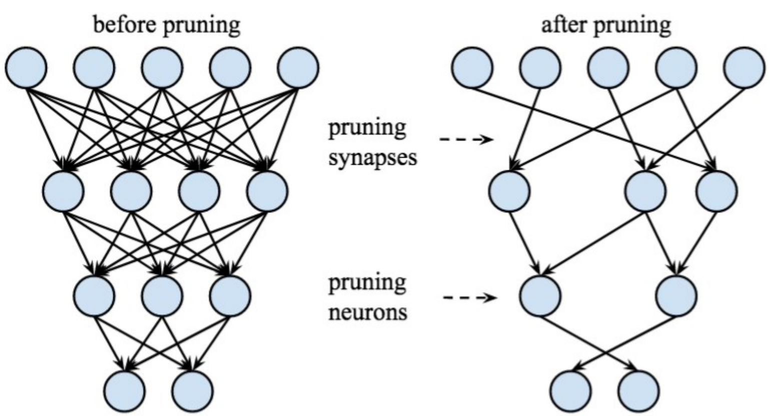
Anshumali Shrivastava, Rice University

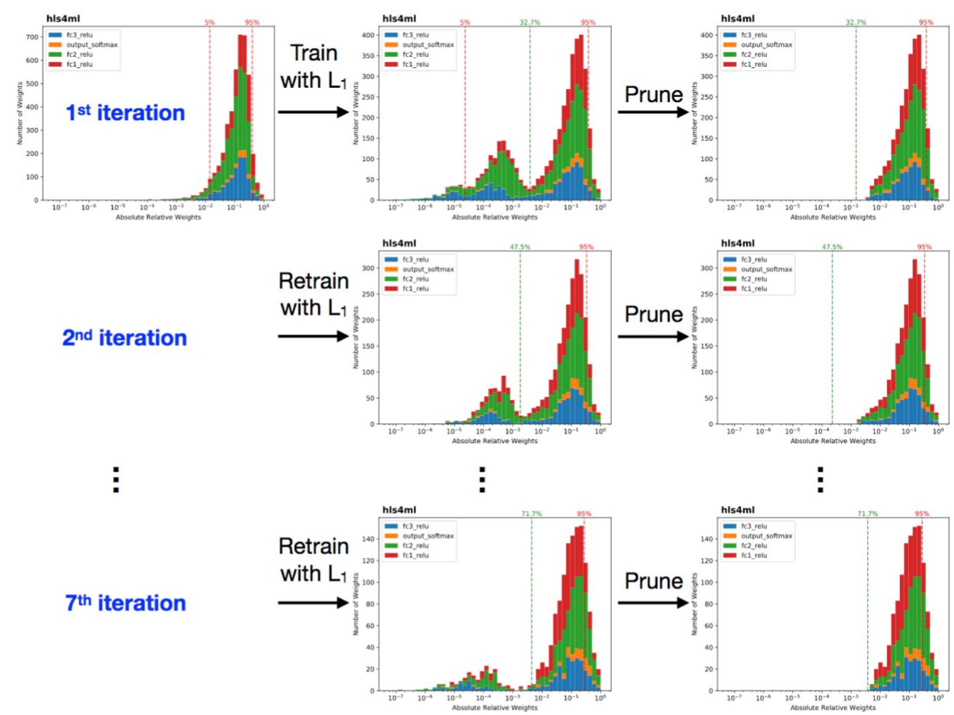


- Introduced properties of hashing
- Relates to their work on anomaly detection : https://arxiv.org/abs/1706.06664
- Hashing in neural net training same perf much less computation

Network Compression

- Redundancy in network weights once trained is a known phenomenon
- Very important when application is time/computing critical inference
  - Cell phone app, self-driving, trigger, …
- Quasi loss-less compression

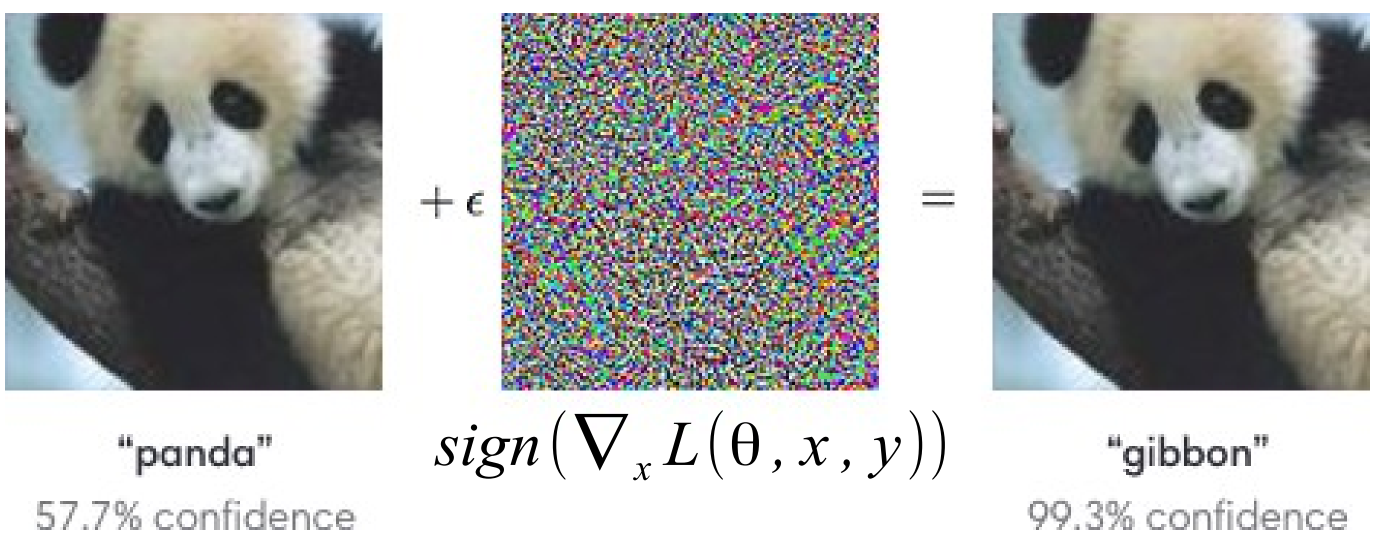




hls4ml 1804.06913

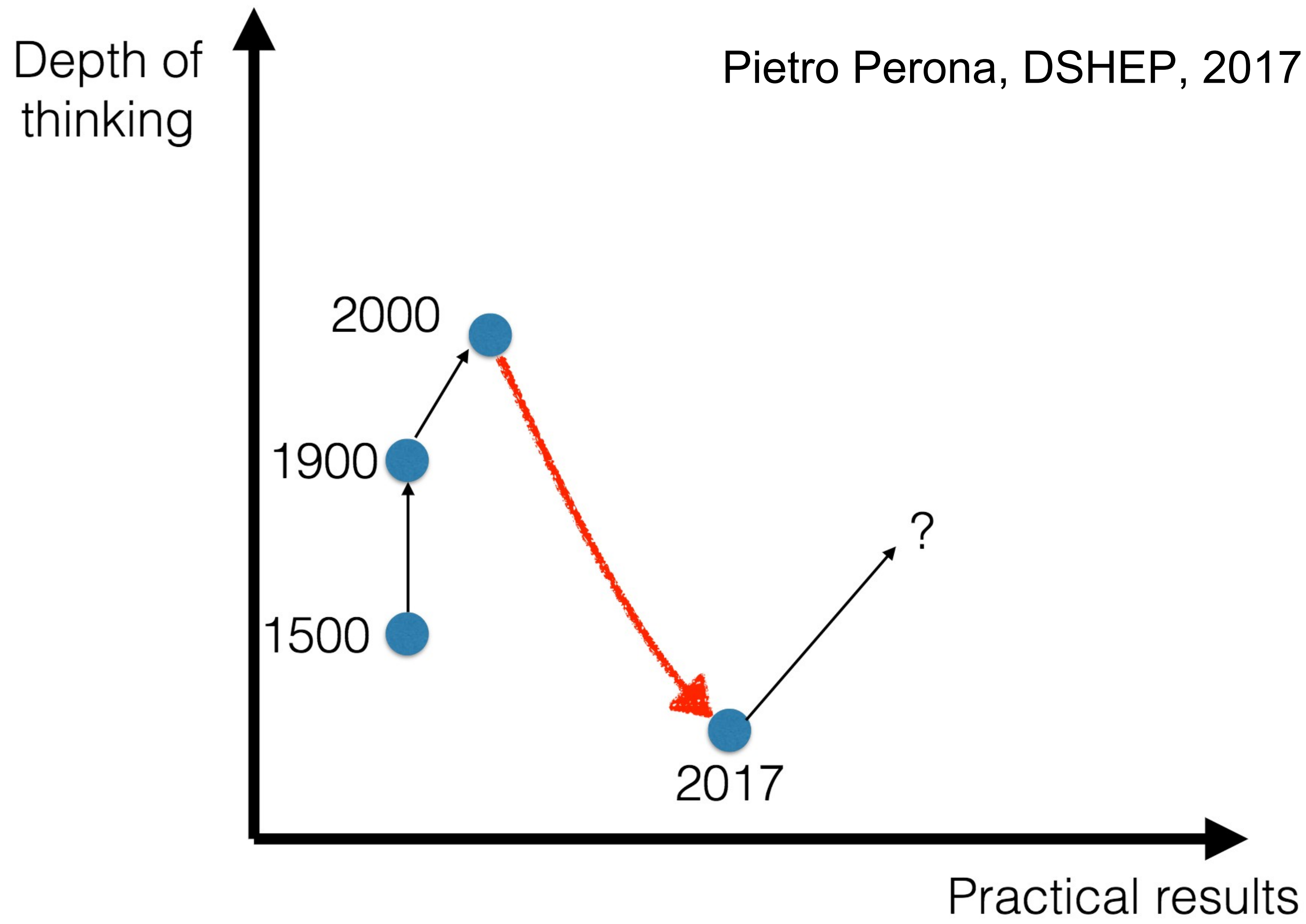
Adversarial Examples

- The loss function L of a model drives the optimization of the model to assign input x to the correct label y
- Possible to alter a given input by gradient descent to classify with a different class
- "Universal" adversarial example build from the gradient of the loss function with respect to the input
- Model can be trained with a corresponding regularizing term
- Puzzling observations. Little risk in physics analysis.



$$sign(\nabla_x L(\theta, x, y))$$

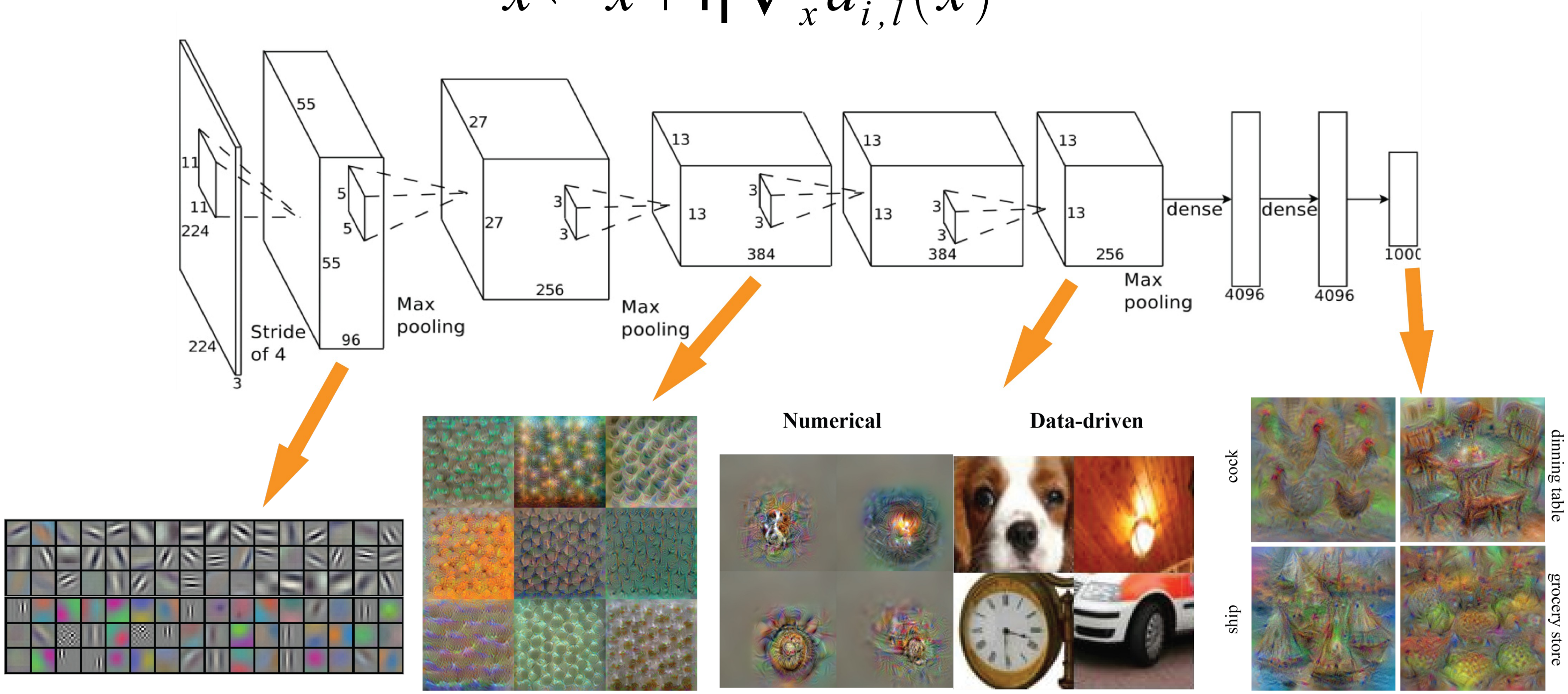
(Lack of) Interpretability



Interpretability

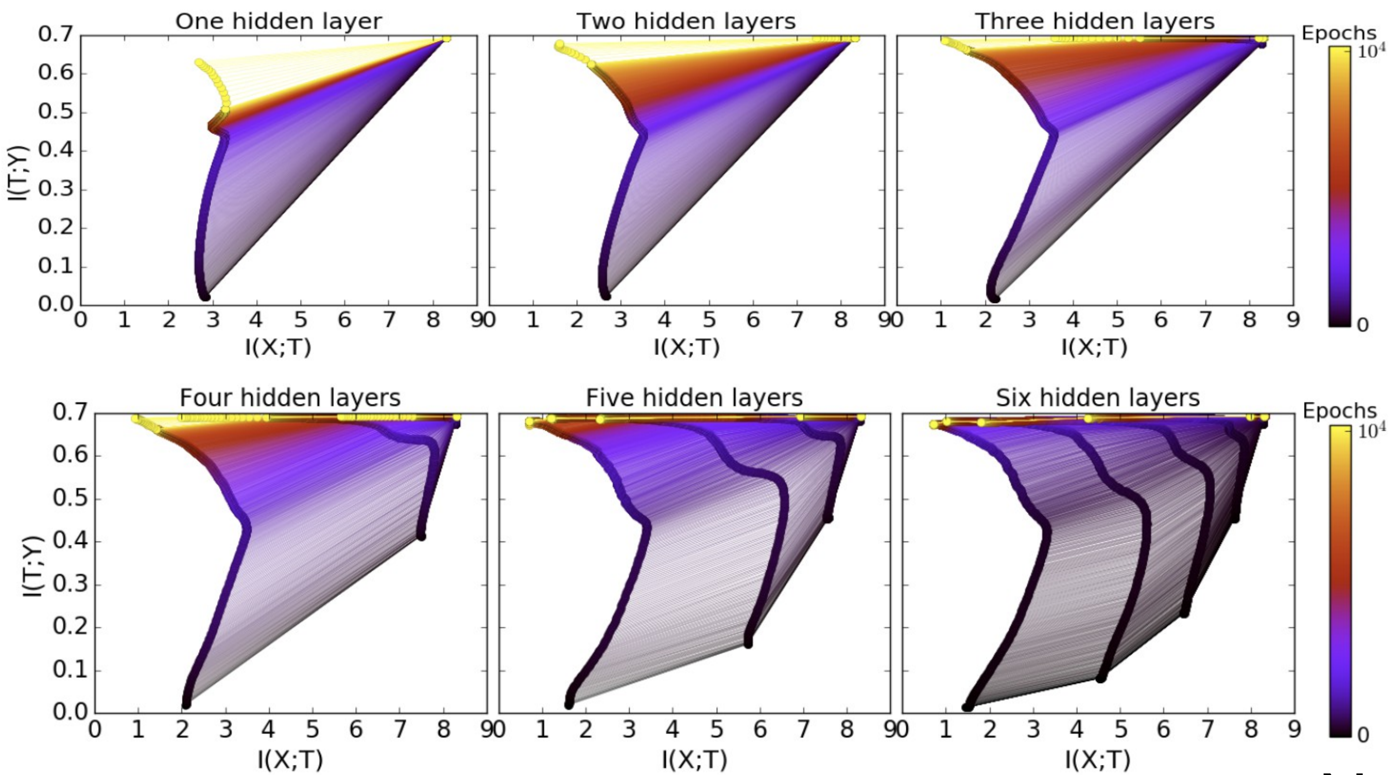
- Trained convolution layers correspond to templated filters applied to input images
- Insightful to create artificial data that maximize a filter activation : $a_{i,l}(x)$
- Can be done with gradient ascent from random input

$$x \leftarrow x + \eta \nabla_x a_{i,l}(x)$$



Conv 1: Edge+Blob
Conv 3: Texture
Conv 5: Object Parts
Fc8: Object Classes

Information Flow



More layers take much FEWER training epochs for good generalization.

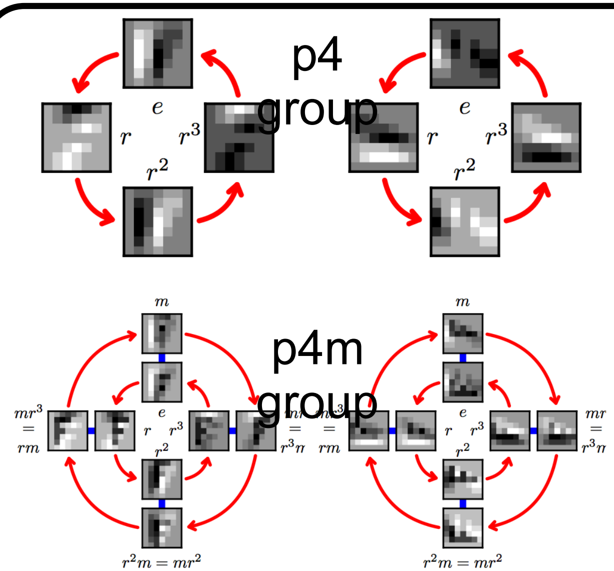
The optimization time depend super-linearly (exponentially?) on the compressed information, delta Ix, for each layer.

N. Tishby, HUJI

$$I(X,Y) = H(X) - H(X|Y)$$

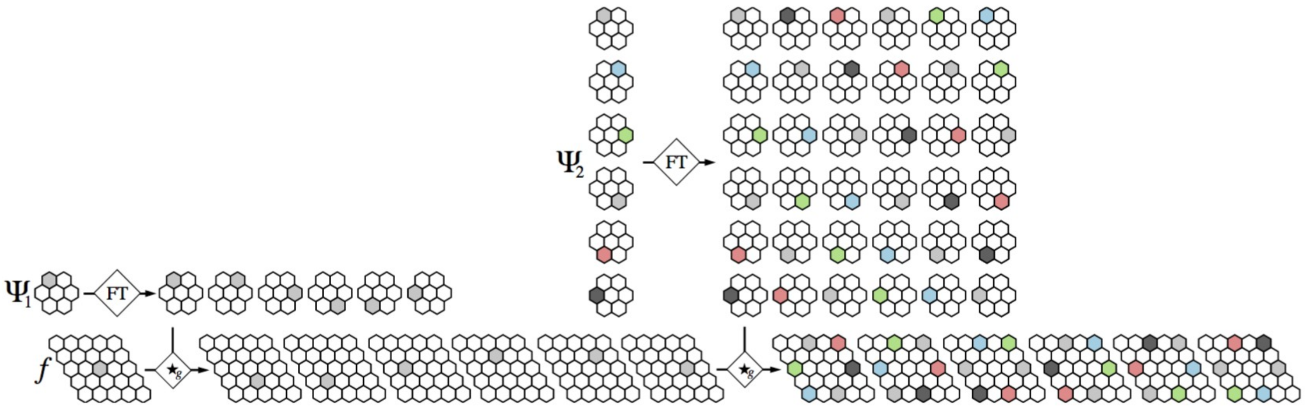
$$I(X,Y) = \sum_{y} \sum_{y} p(x,y) \log\left(\frac{p(x,y)}{p(x)\,p(y)}\right)$$

Embedding Symmetries



T.S. Cohen, M. Welling ICML2016

- Translation invariance brought convolutional layers
- Training with further knowledge of invariance brings improvements
- Including domain knowledge on how object transform brings improvements

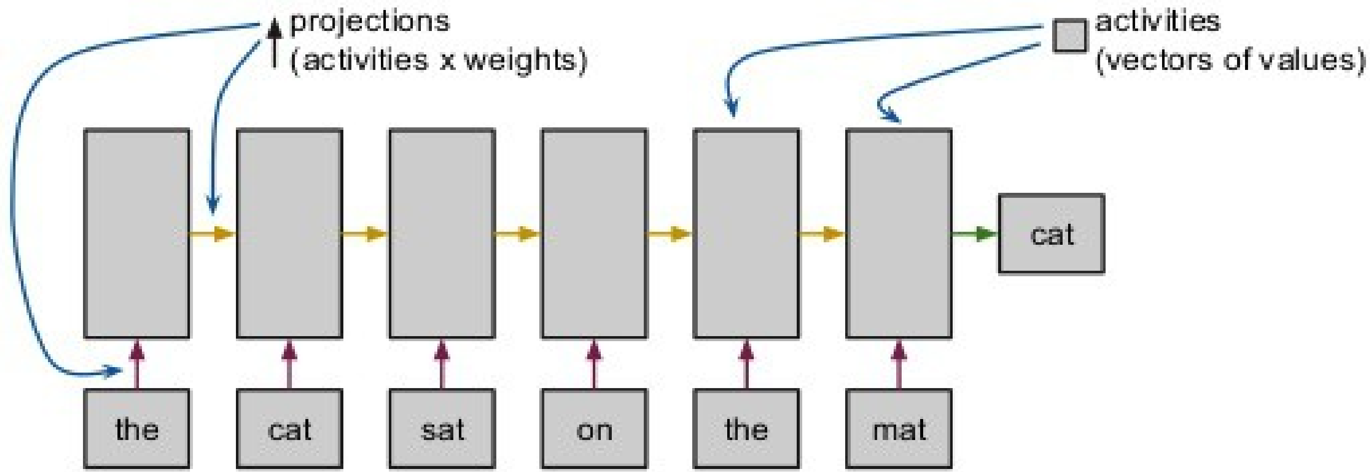


HexaConv 1803.02108

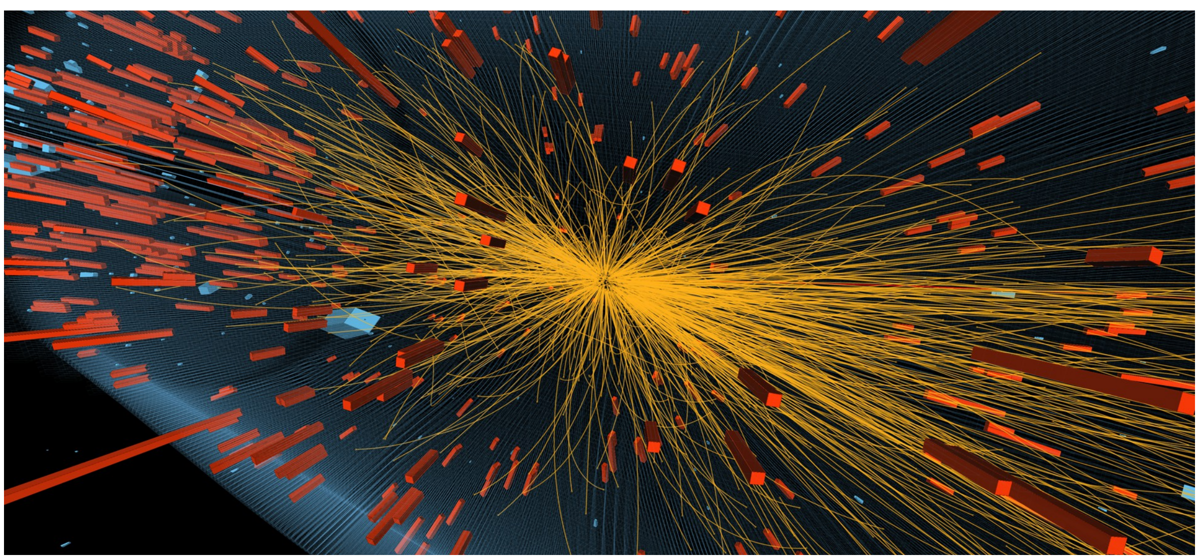
$$\tilde{k}_j \xrightarrow{\text{LoLa}} \hat{k}_j = \begin{pmatrix} m^2(\tilde{k}_j) \\ p_T(\tilde{k}_j) \\ w_{jm}^{(E)}\, E(\tilde{k}_m) \\ w_{jm}^{(d)}\, d_{jm}^2 \end{pmatrix}$$

LOLA 1707.08966

Challenge in Natural Ordering



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

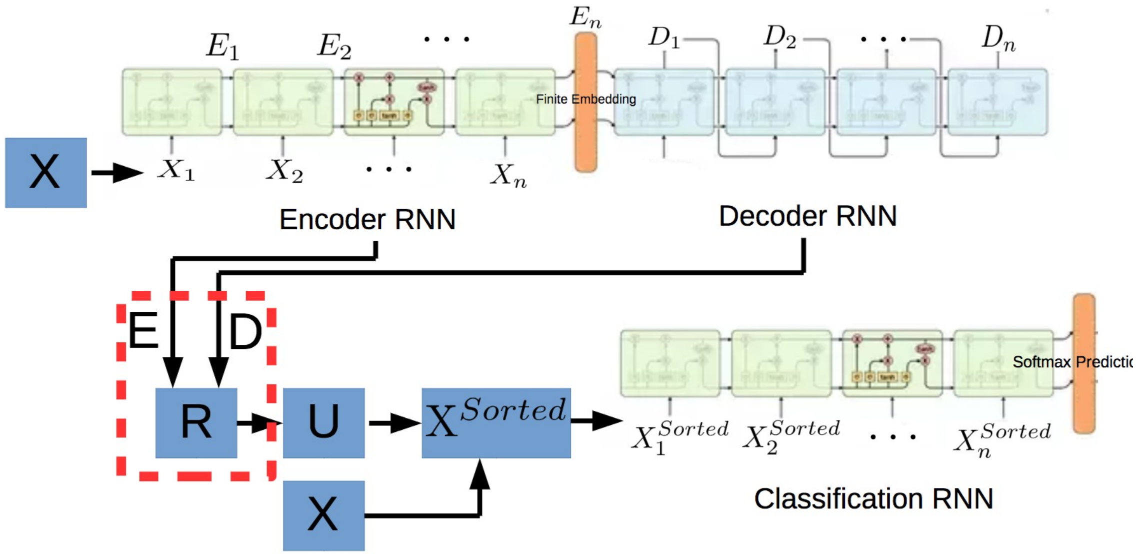


There is underlying order in collision events. Smeared through timing resolution. No natural order in observable

- **Learn how to sort**

Learn How To Sort

Recurrent Neural Networks (RNNs):
-Long Short-Term Memory (LSTM)
-Gated Recurrent Unit (GRU)



- Custom:

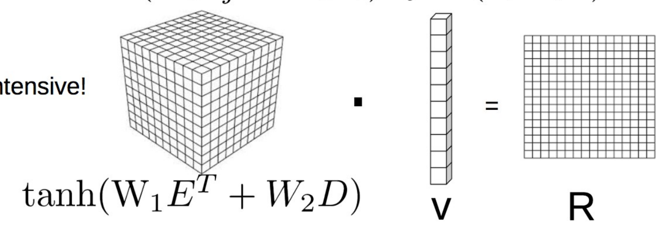
$$R = W1 \begin{bmatrix} — & E_1 & — \\ \vdots & \vdots & \vdots \\ — & E_n & — \end{bmatrix} + W2 \begin{bmatrix} | & \cdots & | \\ D_1 & \cdots & D_n \\ | & \cdots & | \end{bmatrix}$$

Where W_1, W_2 are trainable (nxn) matricies

- Ptr_Net(https://arxiv.org/pdf/1506.03134.pdf)

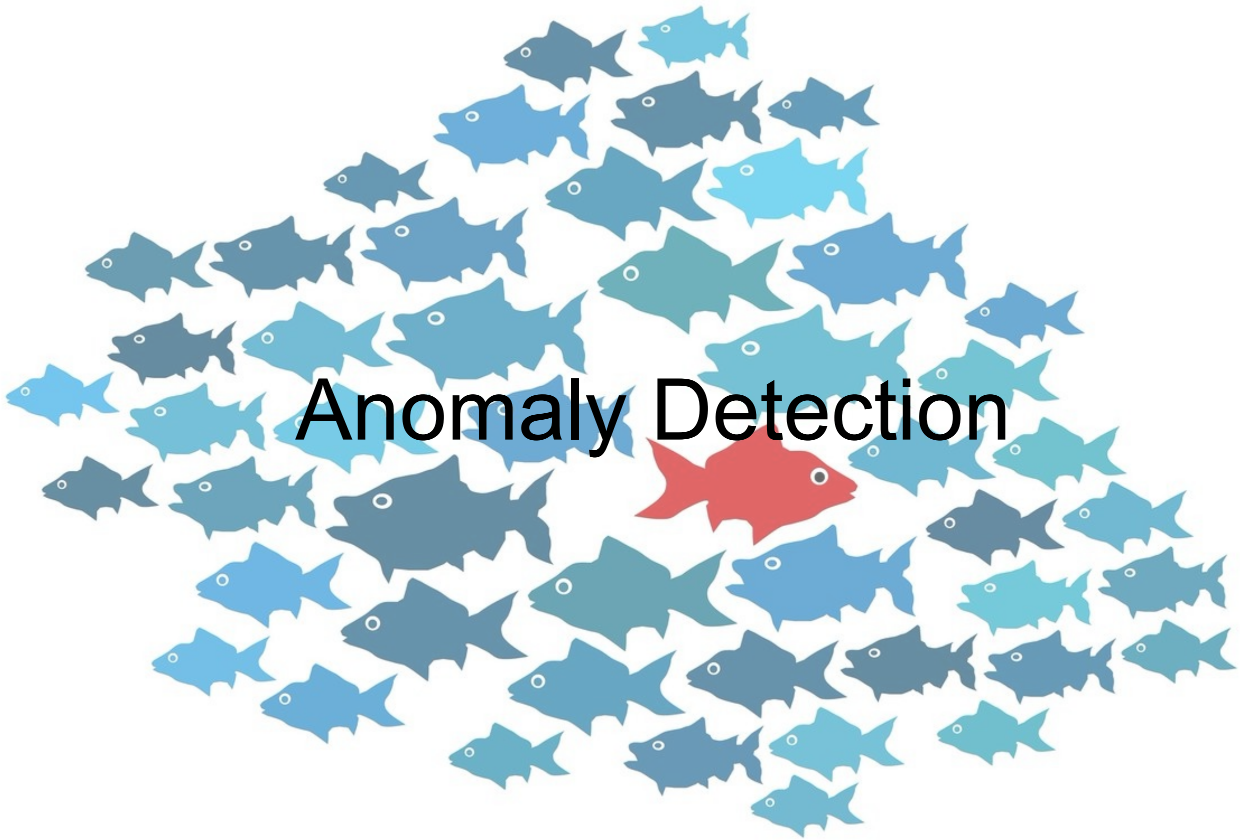
$$\mathrm{R}_j^i = v^T tanh(W_1 e_j + W_2 d_i) \quad j \in (1, ..., n)$$

Memory Intensive!



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

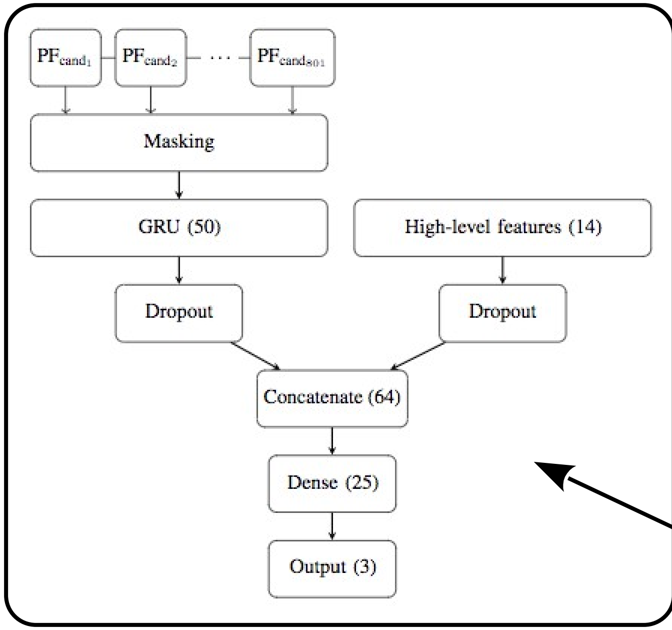
Anomaly Detection

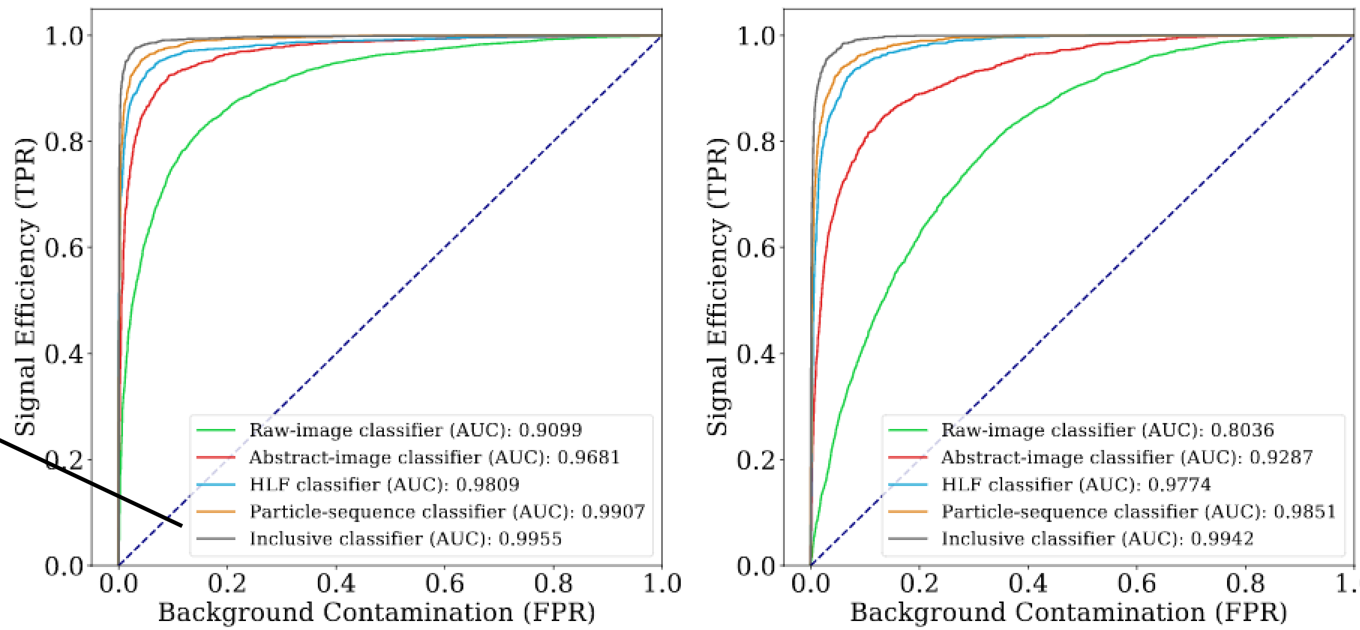


Anomaly Detection

- *An observation which deviates so much from other observation as to arouse suspicion that is was generated by a different mechanism* [Hawkins D.]
- Examples in banking fraud detection, computing system security, network intrusion, …
- Requires a probabilistic model of what the usual data is
  - v-SVM (one-sided SVM), auto-encoder, density estimator, …
- In practice, one can derive a model to guide further data analysis : i.e. trigger human intervention

Topology Classification





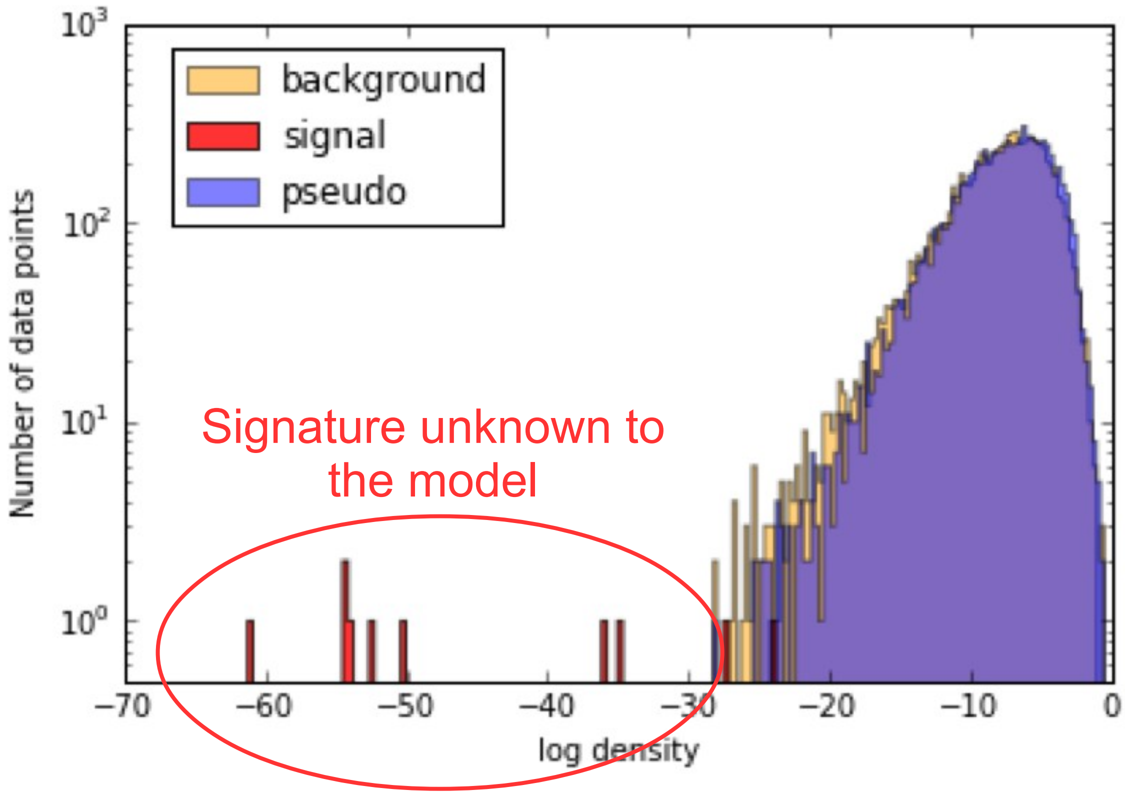
(a) $t\bar{t}$ selector (b) W selector

- Various approaches on the benchmark
- AI still needs the physicist's derived features
- Quasi unbiased x10 rejection factor on background triggers

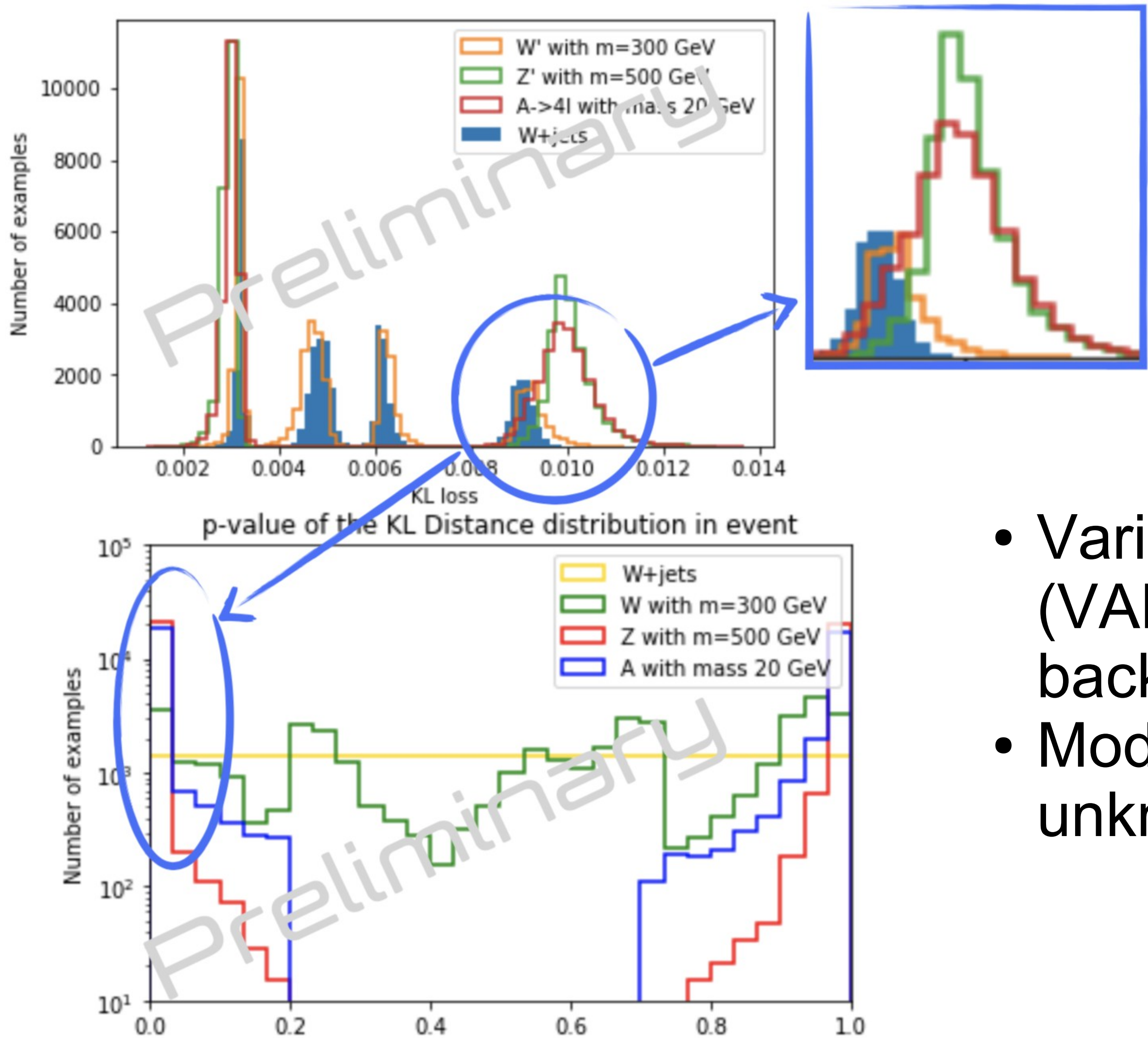
https://arxiv.org/abs/1807.00083

Outlier Identification

- Train a NADE (https://arxiv.org/abs/1306.0186) model on mixture of the known backgrounds
- Use a synthetic dataset with small injected unknown signature
- Log density singles out the injected signal



New Physics Triggering

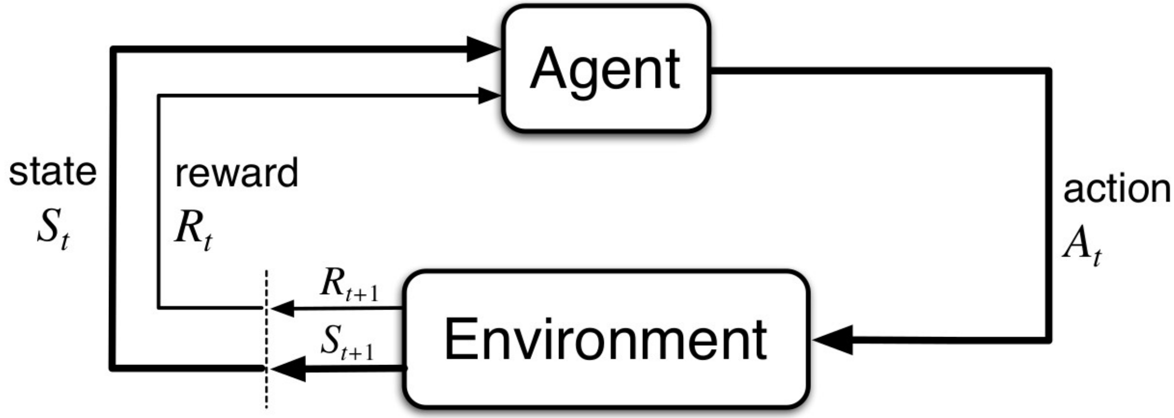


- Variational Autoencoder (VAE) trained on the major background of a trigger line
- Model is used to identify unknown signatures

Control Learning

Reinforcement Learning

- Supervised learning with objective provided by an environment
- Computational intensive optimization problem
  - p-Learning : modeling/optimize the action/policy
  - Q-learning : modeling/optimize the action-value function
- Requires either an environment or a simulator to compute reward



$$\pi(a|s)=P(A_t=a|S_t=s)$$

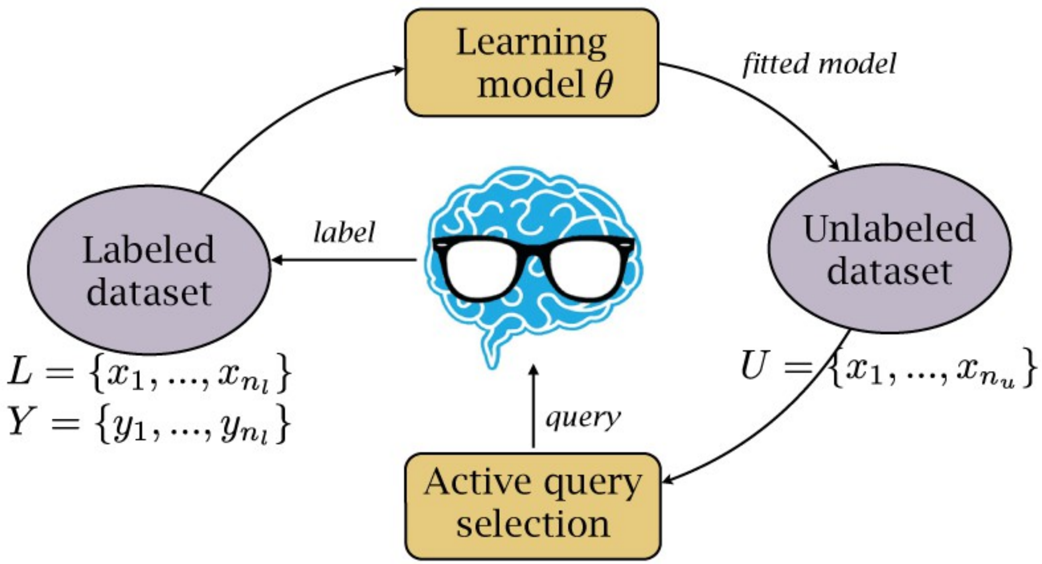
$$\underset{\pi}{argmax}\, E\left[\sum_t \gamma^t R_t|S_0\right]$$

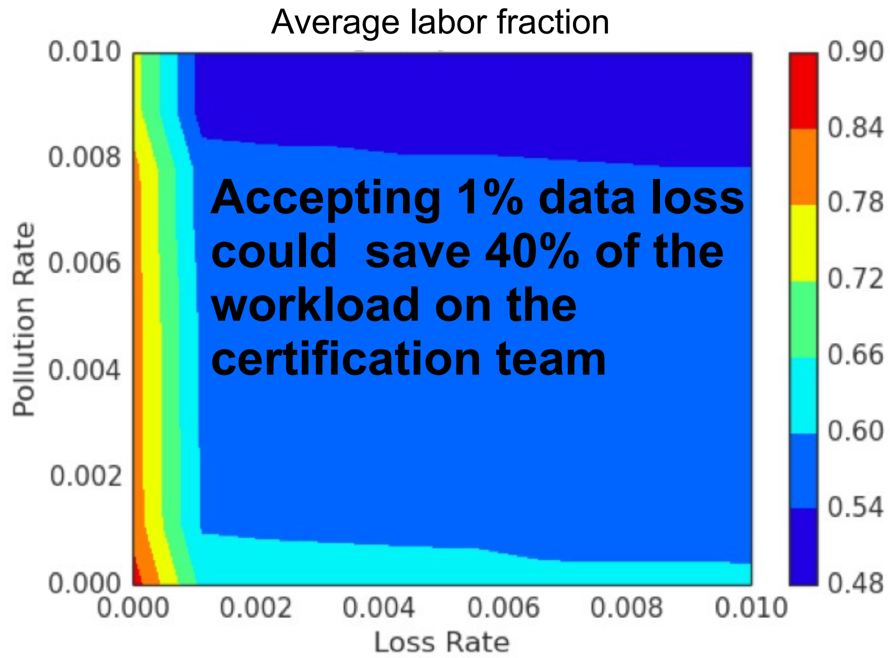
$$V_\pi(s)=E_\pi\left[\sum_k \gamma^k R_{t+k}|S_t=s\right]$$

$$Q(s,a)=E_\pi\left[\sum_k \gamma^k R_{t+k}|S_t=s, A_t=a\right]$$

Active Learning

- Semi-supervised technique to tackle the problem of unlabelled dataset
- The model provides the unlabelled samples most relevant to the classification convergence

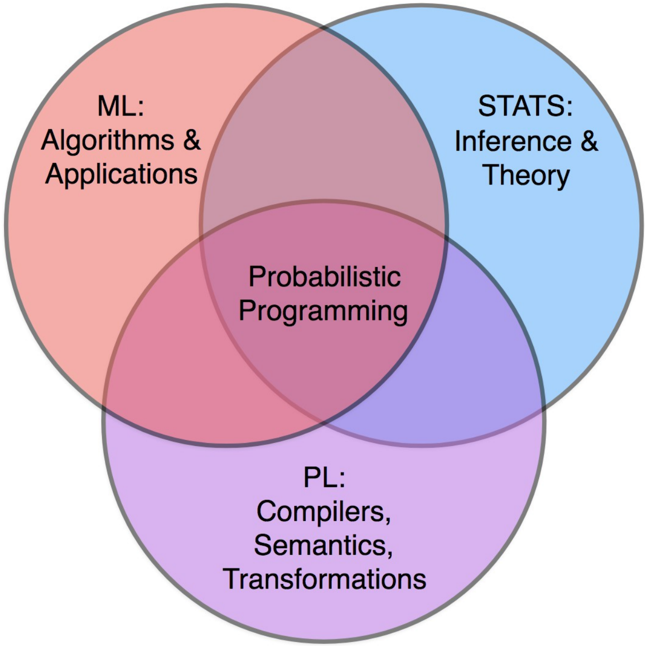




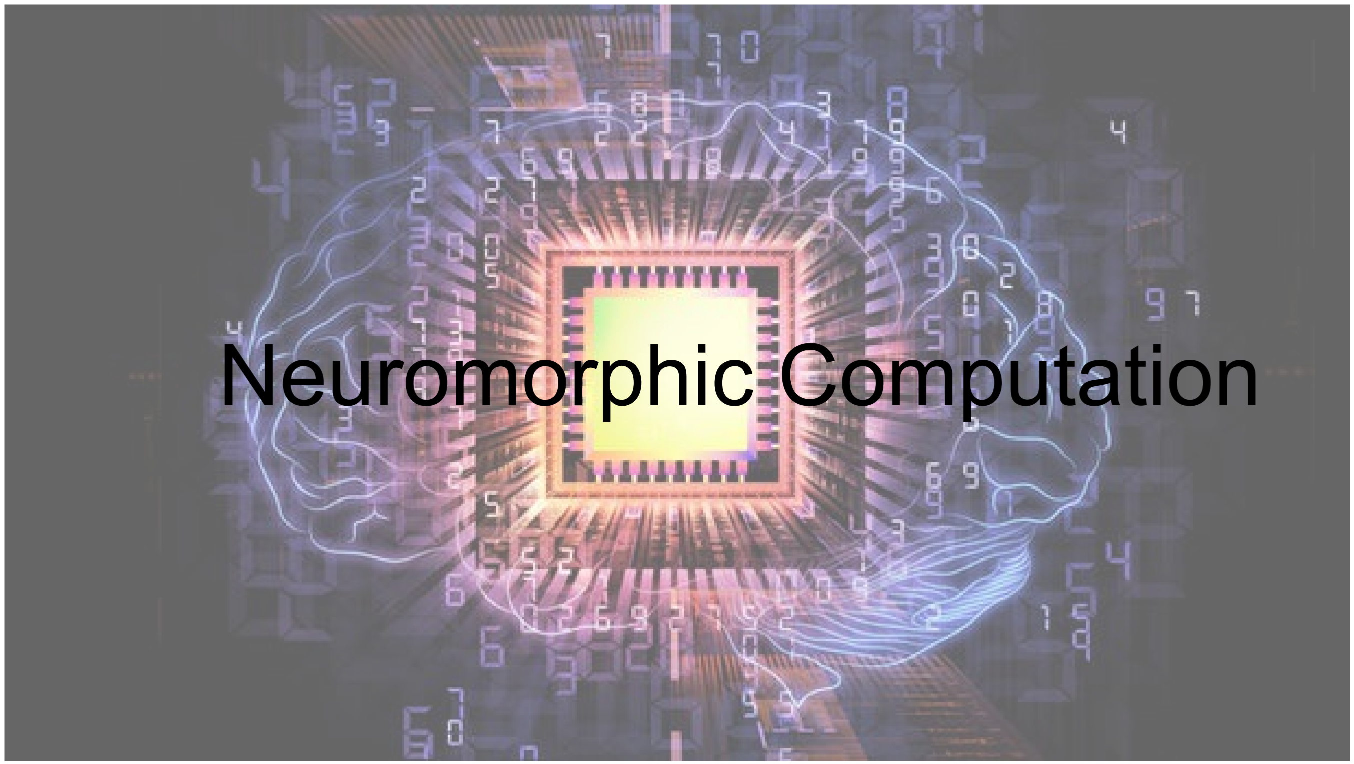
Probabilistic Programming

- Instrumenting computer program with control over probabilistic variables : $X \rightarrow P(X)$
- Provides efficient tools for inferring the conditional probability of model parameters, given a set of observation

$$P(model|X) = \frac{P(X|model)\,P(model)}{P(X)}$$

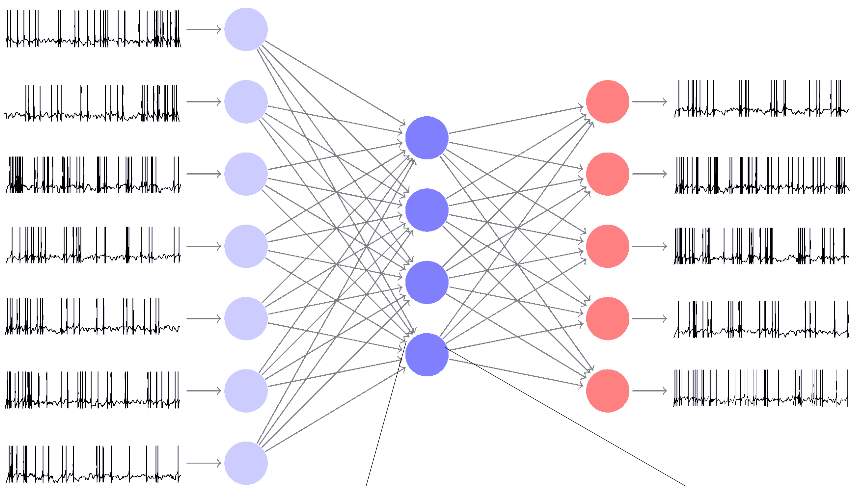


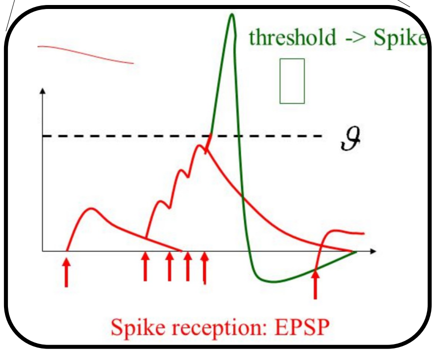
Neuromorphic Computation



Spiking Neural Network

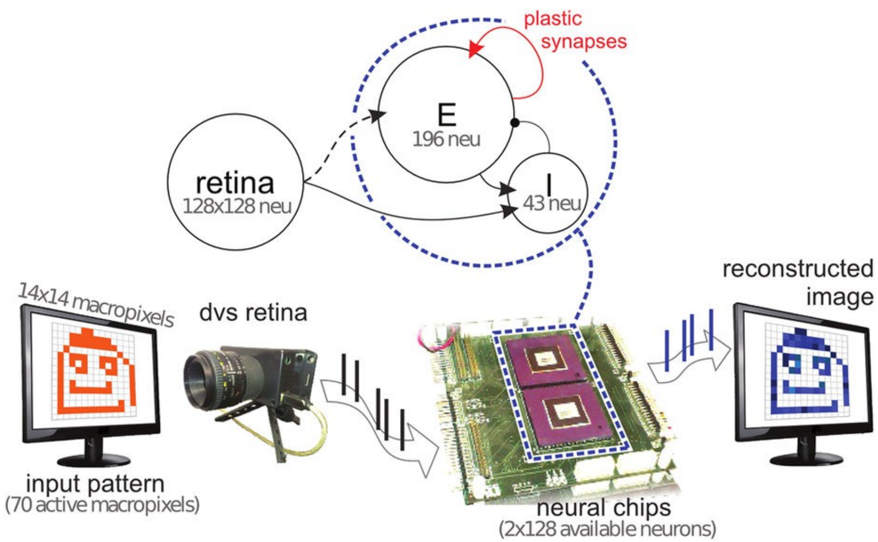
- Closer to the actual biological brain
- Adapted to temporal data
- Hardware implementation with low power consumption
- Trained using evolutionary algorithms
- Economical models

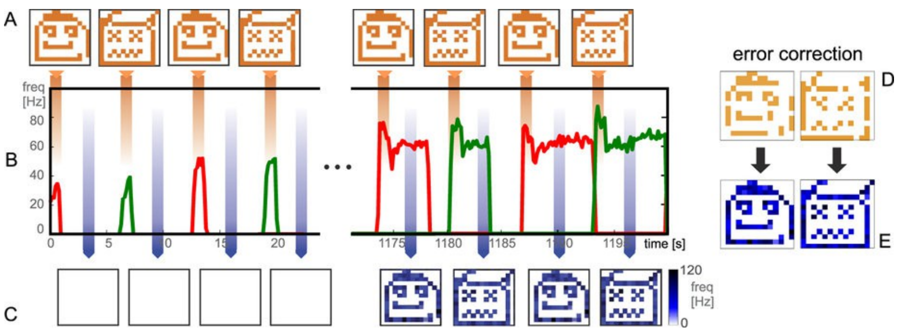




	Deep Learning	Spiking
Training Method	Back-propagation	Not well established (here, genetic algorithms)
Native Input Types	Images/Arrays of values	Spikes
Network Size	Large (many layers, many neurons and synapses per layer)	Relatively small (fewer neurons and sparser synaptic connections)
Processing Abilities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

Neuromorphic Hardware



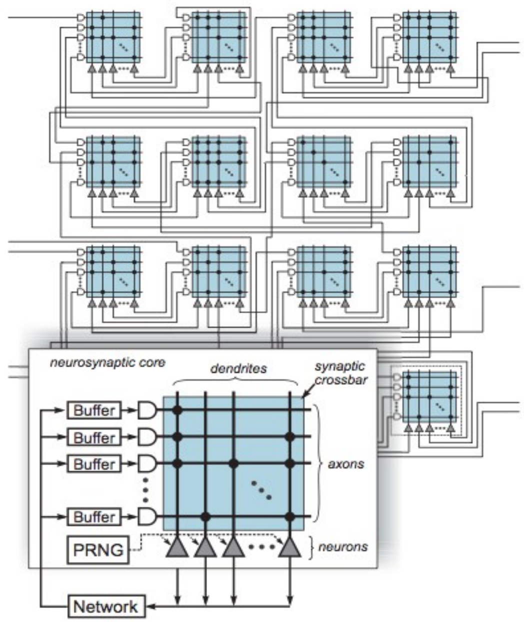


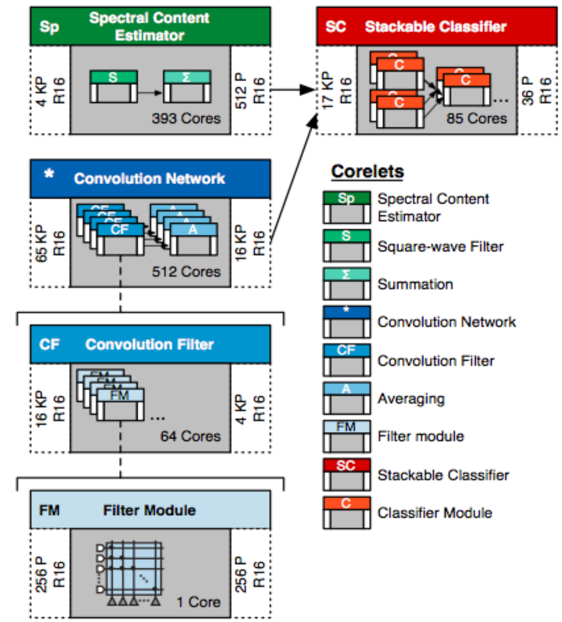
http://www.nature.com/articles/srep14730

- Implementing plasticity in hardware
- Process signal from detector and adapt to categories of pattern (unsupervised)
- Post-classified from data analysis or rate throttling

➢ NCCR consortium assembling to develop this technology further, with our use case in mind

Cognitive Computing





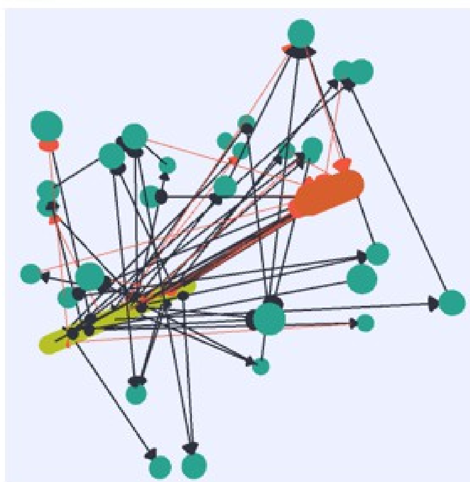
- Spiking neural net as processing units :
  ➔ Cognitive Computing Processing Unit : CCPU
- Adopt a **new programming scheme**, translate existing software

Neutrino Identification with SNN

Best Results: Single View

x-view (127x50)	conv1 (8x3)	pool1 (2x1)	conv2 (7x3)	pool2 (2x1)	conv3 (6x3)	pool3 (2x1)	conv4 (6x3)	pool4 (2x1)	fc1 (196)	drop out	fc2 (98)	drop out	fc3 (11)	classification

Convolutional Neural Network Result: ~80.42%



- 90 neurons, 86 synapses
- Estimated energy for a single classification for mrDANNA implementation: 1.66 μJ

Spiking Neural Network Result: ~80.63%

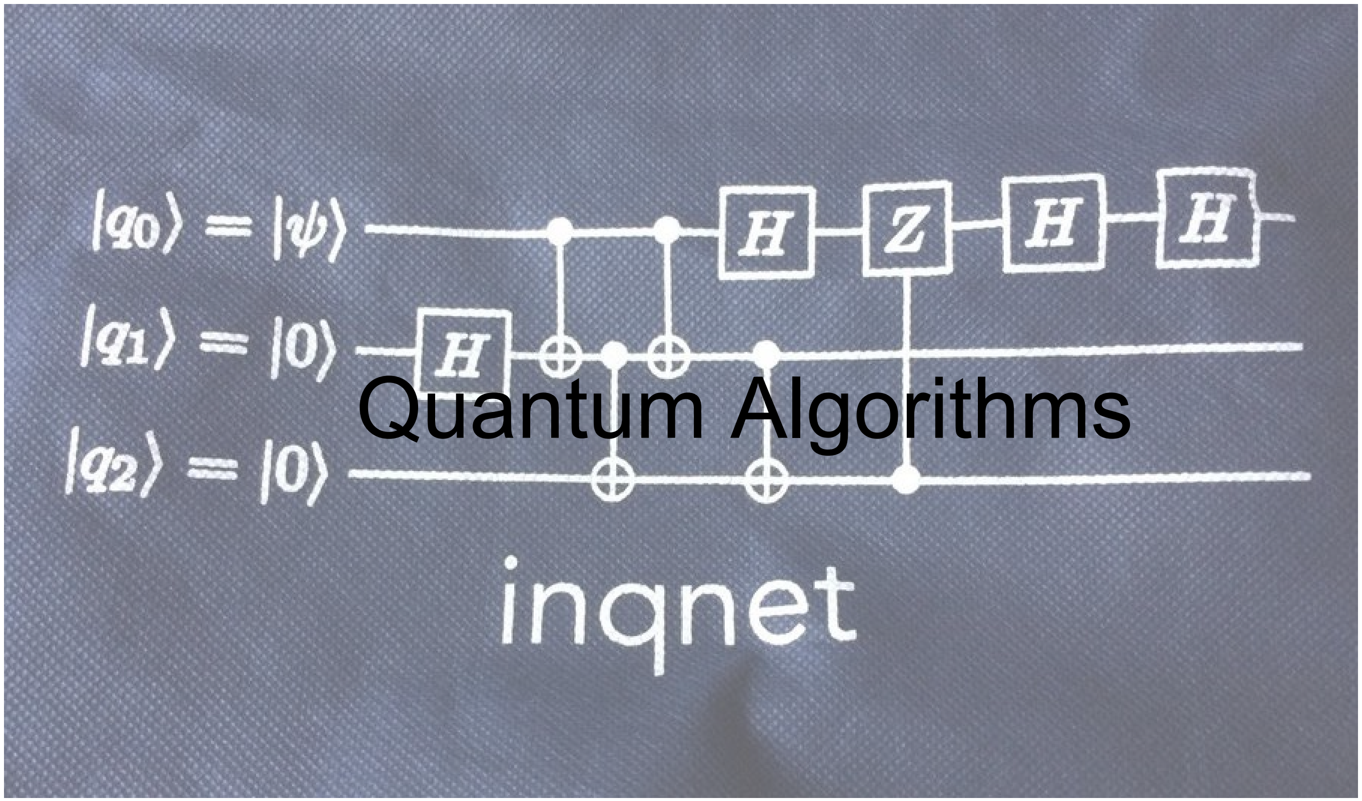
Source for CNN results: A. Terwilliger, et al. Vertex Reconstruction of Neutrino Interactions using Deep Learning. IJCNN 2017.

33 Programming Neuromorphic Computing Systems

OAK RIDGE National Laboratory

https://indico.fnal.gov/event/13497/contribution/0

Quantum Algorithms



Quantum Machine Learning

MENU nature International journal of science

Letter

Solving a Higgs optimization problem with quantum annealing for machine learning

Alex Mott, Joshua Job, Jean-Roch Vlimant, Daniel Lidar & Maria Spiropulu

Nature **550**, 375–379 (19 October 2017)

doi:10.1038/nature24047

Download Citation

Computational science

Experimental particle physics | Qubits

Theoretical particle physics

Received: 04 April 2017

Accepted: 28 July 2017

Published online: 18 October 2017

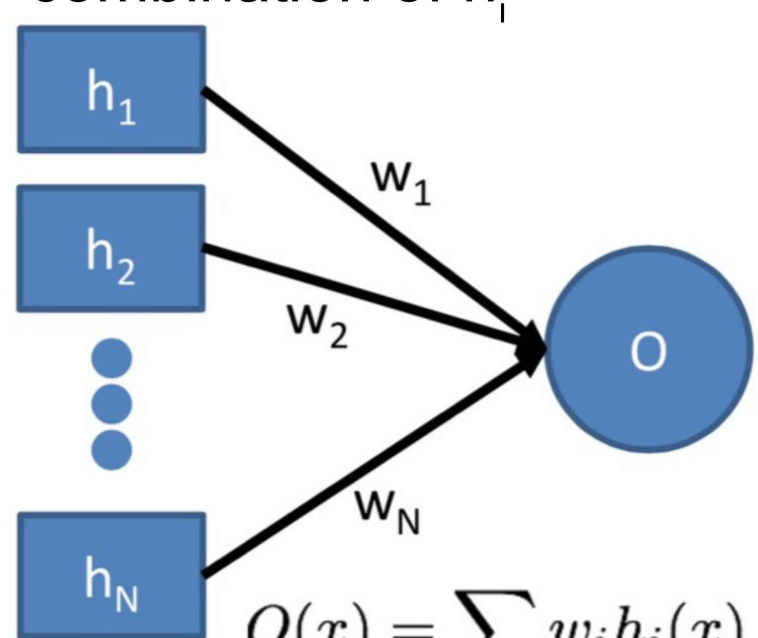
QAML Weak/Strong Classifier

Define functions $\mathbf{h_i}$ of the input variables into [-1,1] such that

- P(signal|h>0) > P(bkg|h>0)
- P(bkg|h<0) > P(signal|h<0)

i.e. Most signal on h>0, most bkg on h<0

Define $\mathbf{w_i}$ as binary linear combination of h_i



$$O(x) = \sum_i w_i h_i(x)$$

https://arxiv.org/abs/1109.0325

QAML Target/Objective

Define as a "target" function

$$y(x) = \begin{cases} +1, & \text{if } \in S, \\ -1, & \text{if } \in B \end{cases}$$

Per event error

$$E(x) = y(x) - \sum_{i=1}^{N} w_i h_i(x)$$

Full error

$$\delta(\vec{w}) \propto \sum_{i,j} C_{ij} w_i w_j + \sum_i (\lambda - 2C_{iy}) w_i$$

- ➔ C_{ij} and C_{iy} are summations over the values of h_i over the training set
- ➔ λ is a parameter penalizing the number of non-zero w_i

QUBO

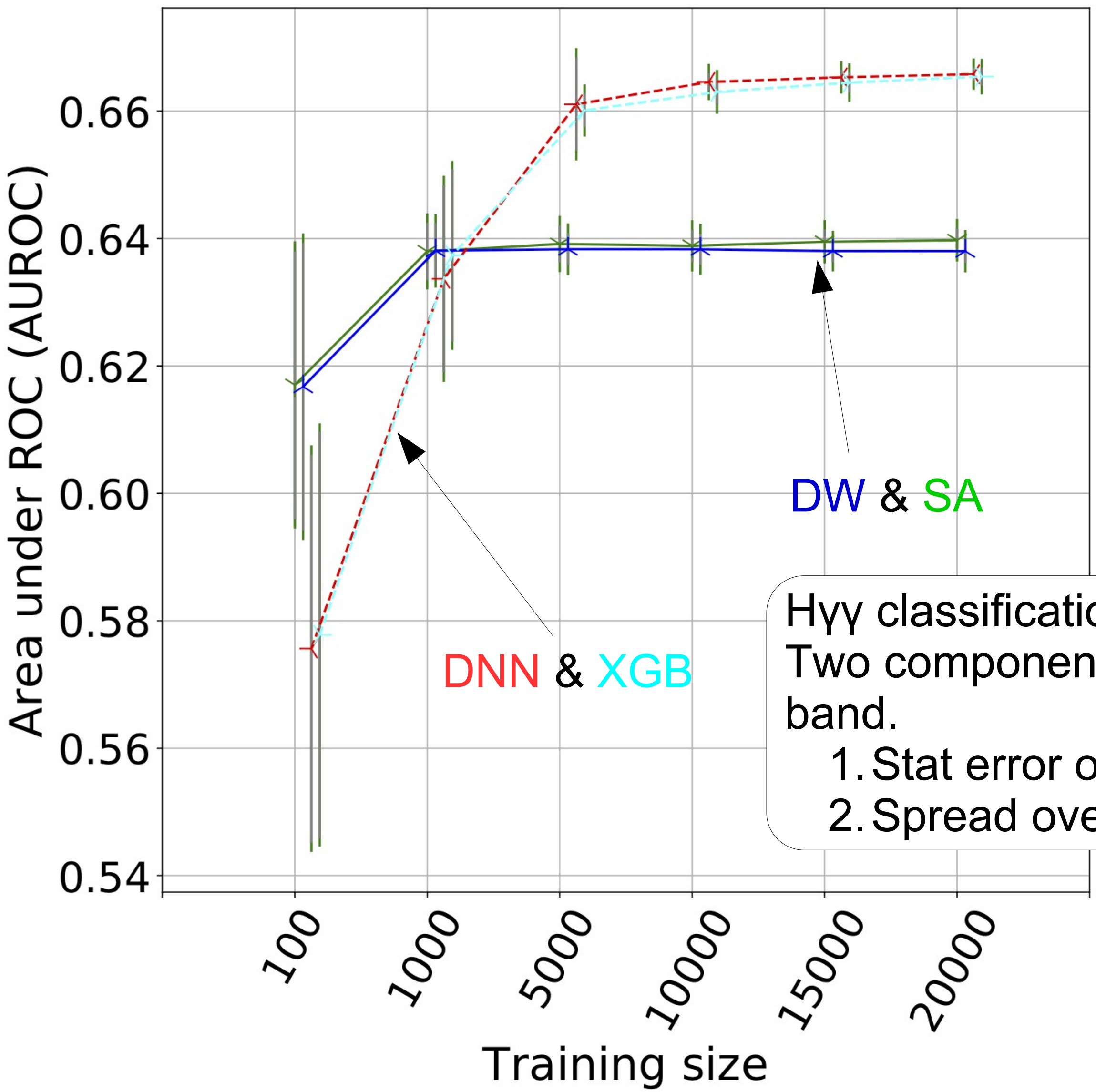
Quadratic Unconstrained Binary Optimization

$$\delta(\vec{w}) \propto \sum_{i,j} C_{ij} w_i w_j + \sum_i (\lambda - 2C_{iy}) w_i$$

Simple conversion of binary weights to ±1

$$H_{\text{Ising}} = \sum_i h_i \sigma_i^z + \sum_{ij} J_{ij} \sigma_i^z \sigma_j^z$$

QAML for HEP

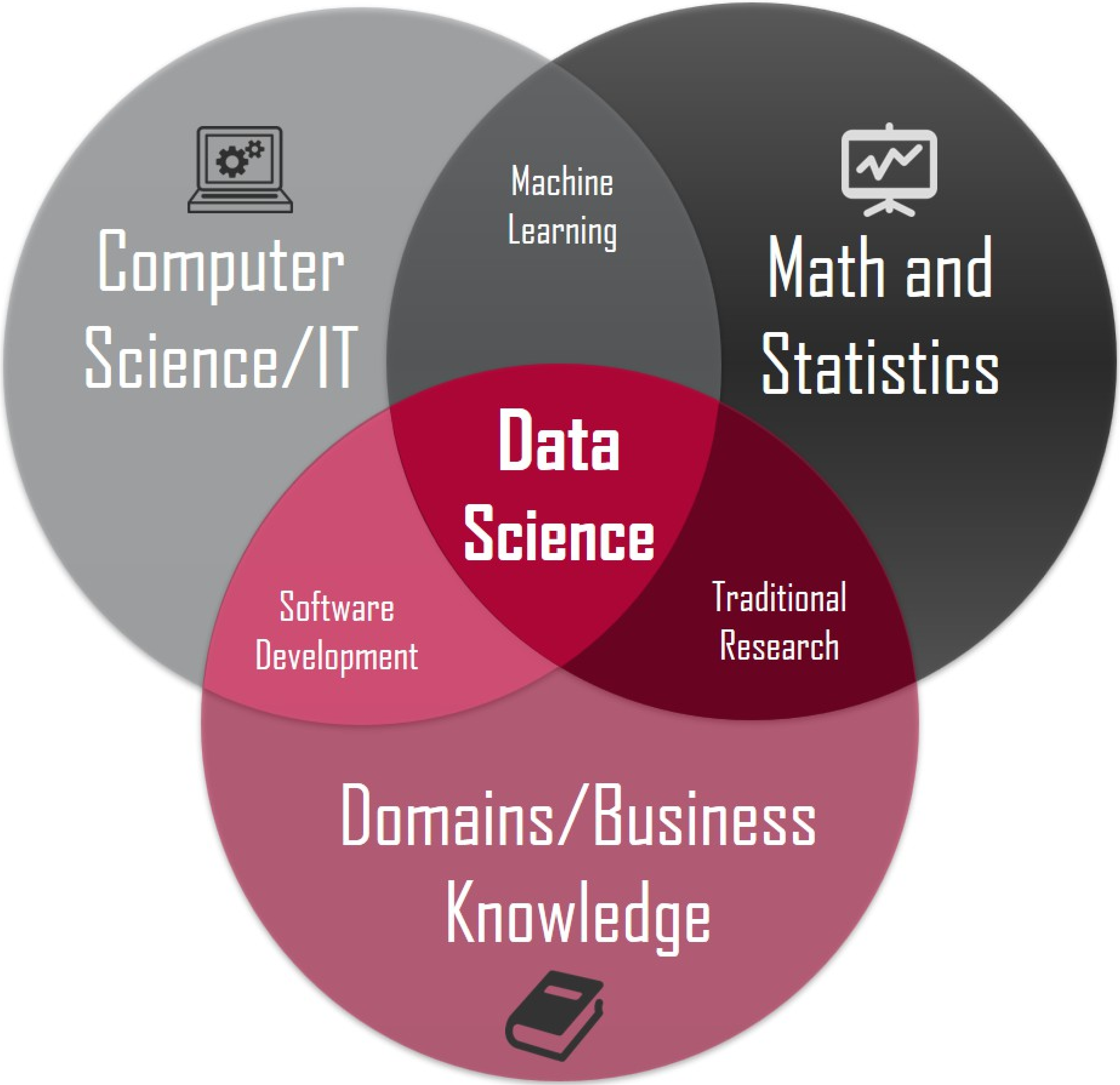


Hγγ classification problem
Two components in the error band.

1. Stat error of the test set.
2. Spread over the folds.

Summary/Recap Lecture 3

- The field of machine learning is still in evolution
- The field of deep learning is in exponential evolution
- There is much more than regression and classification
- There is more than artificial neural networks
- Experimental field where the scientific method of a physicist can make a difference


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

Final Remarks

- Many thanks to the organizers of the school for the invitation to give this lecture series.
- Relevant credits go to Y. Le Cun, G. Louppe, M. Kagan, A. Rogozhnikov, A. Artemov for the past lectures I have inspired myself with.
- Thanks to M. Pierini for reviewing the content of the lecture and providing feedback.

Extra Slides

References

Books

- Statistical analysis techniques in particle physics, I. Narsky, F. Porter
- Deep Learning, I. Goodfellow, Y. Bengio, A. Courville

Lectures

- M. Kagan https://indico.cern.ch/event/619370/
- http://comet.lehman.cuny.edu/owen/teaching/datasci/sp2017.html

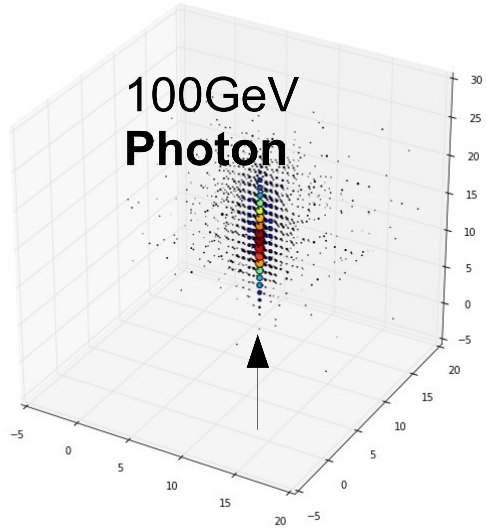
Conference Series

- Data-science-HEP Series http://dshep.fnal.gov/
- MLHEP series https://indico.cern.ch/event/687473/ https://github.com/yandexdataschool/mlhep2018
- https://dl4physicalsciences.github.io/
- https://indico.fnal.gov/event/ANLHEP1017/

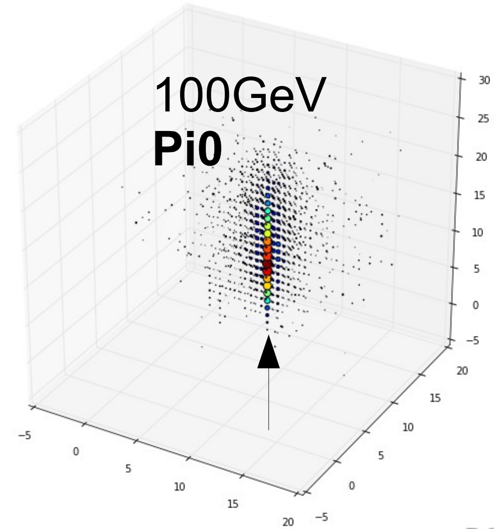
Article and blogs

- Machine learning at the energy and intensity frontiers of particle physics https://www.nature.com/articles/s41586-018-0361-2
- http://www.shivonzilis.com/machineintelligence
- https://www.nvidia.com/en-us/deep-learning-ai/
- http://bigdata-madesimple.com/machine-learning-explained-understanding-supervised-uns
- http://ruder.io/optimizing-gradient-descent/
- http://colah.github.io/posts/2015-08-Understanding-LSTMs/
- https://indico.cern.ch/event/737584/contributions/3105461/
- https://medium.com/@jonathan_hui/

3D Calorimetry Imaging

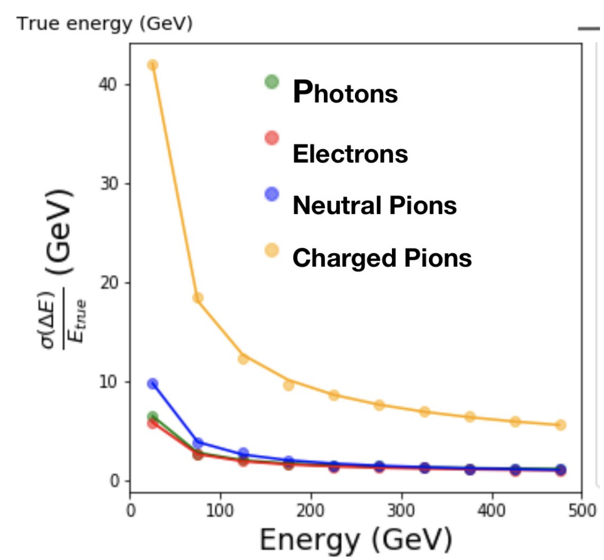


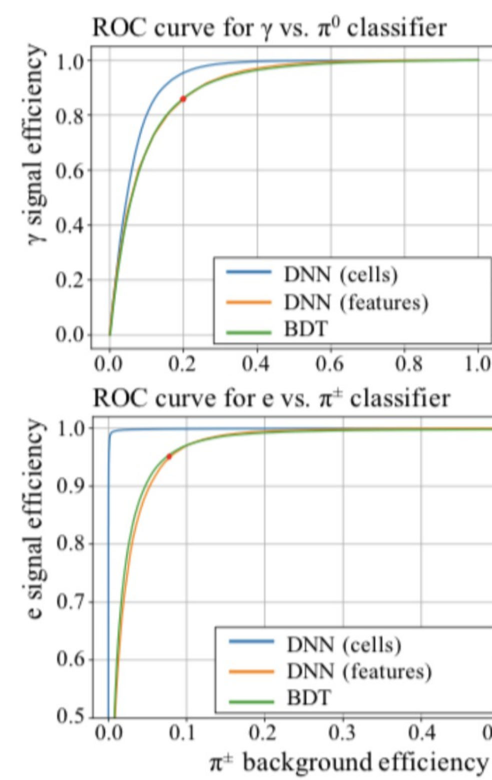
≠



LCD Calorimeter configuration
http://lcd.web.cern.ch
5x5 mm Pixel calorimeter
28 layer deep for Ecal
70 layer deep for Hcal

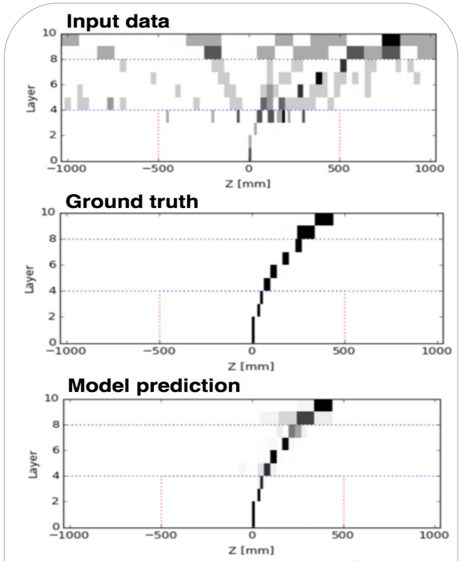
Using the raw cell information
➔ Particle classification
➔ Energy regression



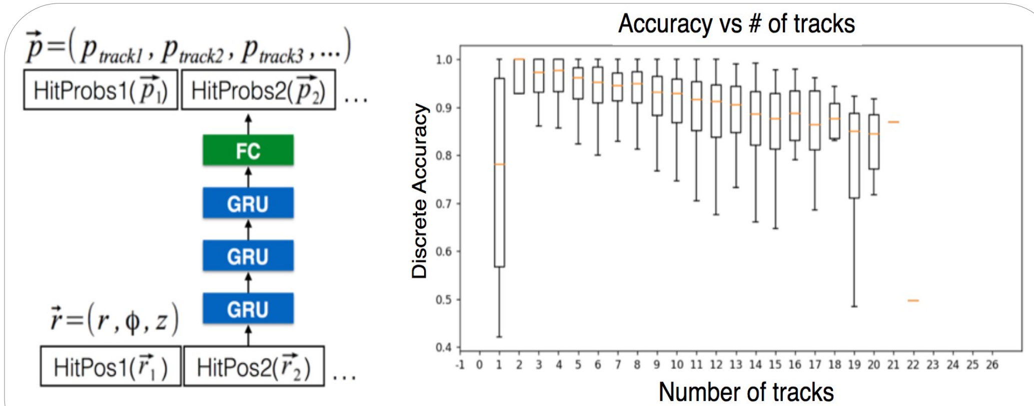


https://dl4physicalsciences.github.io/files/nips_dlps_2017_15.pdf

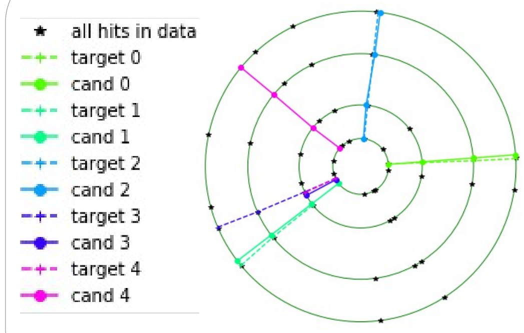
HEP.TrkX Approaches



Track following with RNN

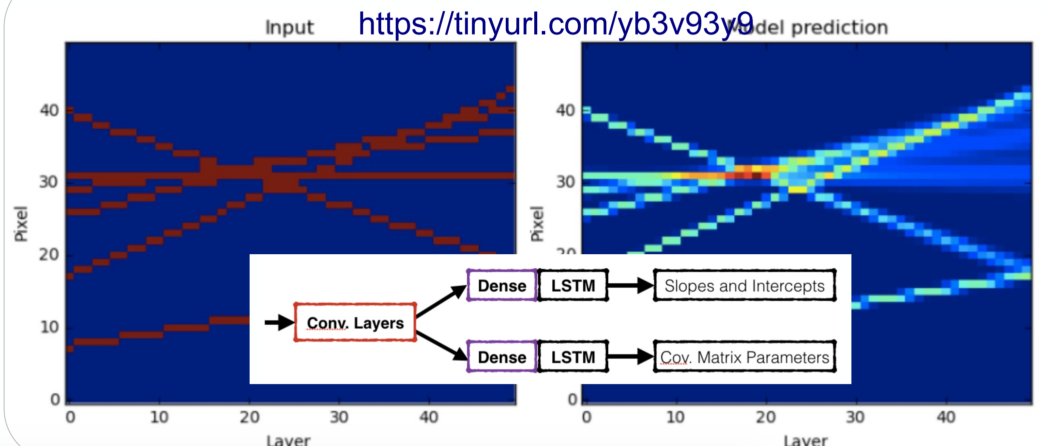


End-to-end hit assignment

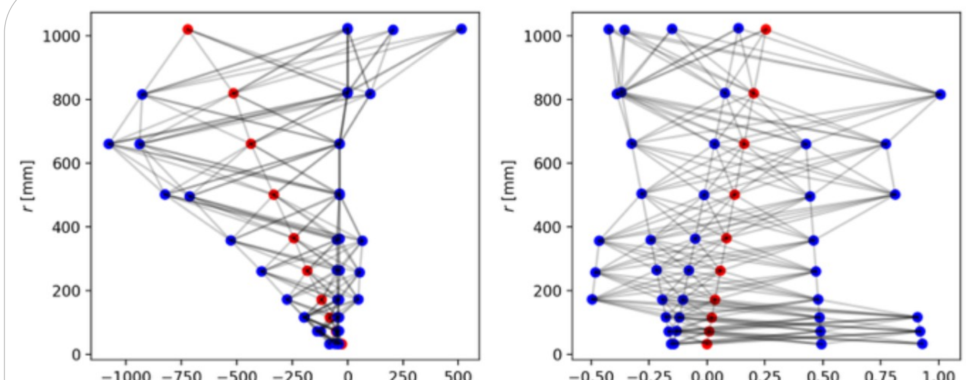


Seq-to-seq track finding

https://heptrkx.github.io/



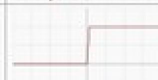
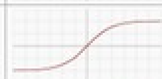
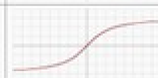
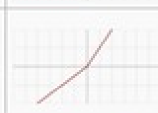
https://tinyurl.com/yb3v93y9



https://tinyurl.com/y87saehf

Pilot project funded by **DOE ASCR** and **COMP HEP**. Part of **HEP CCE**. *LBNL, Fermilab, Caltech consortium*

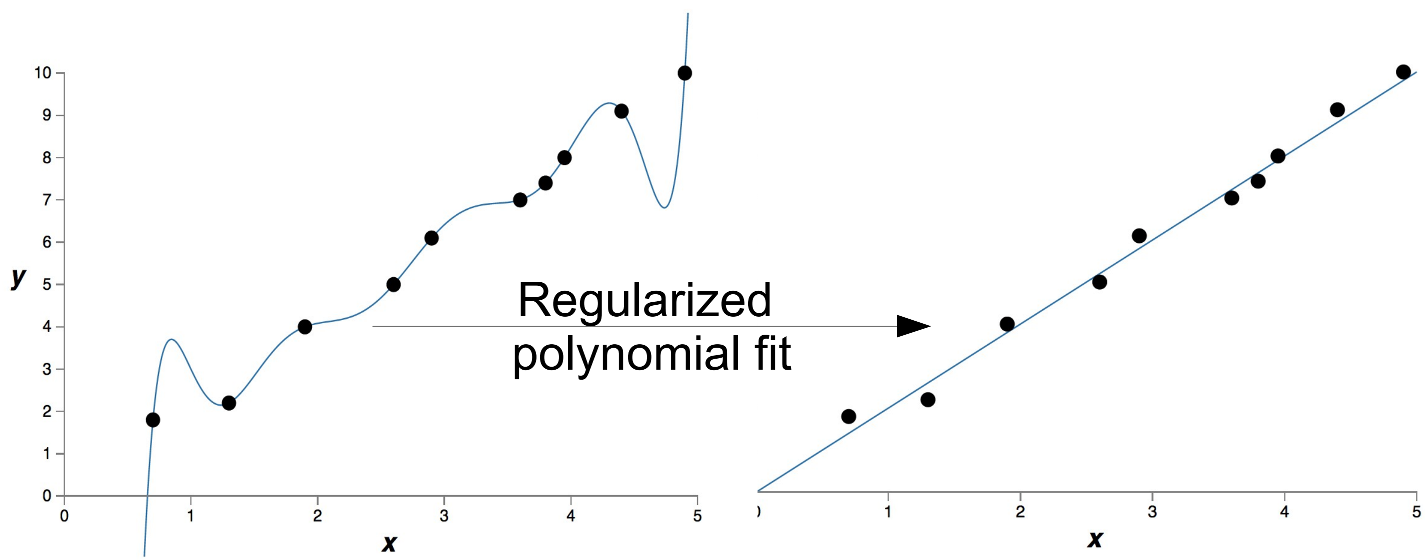
Internal Node Activation

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		$f(x) = \log_e(1 + e^x)$	$f'(x) = \frac{1}{1+e^{-x}}$

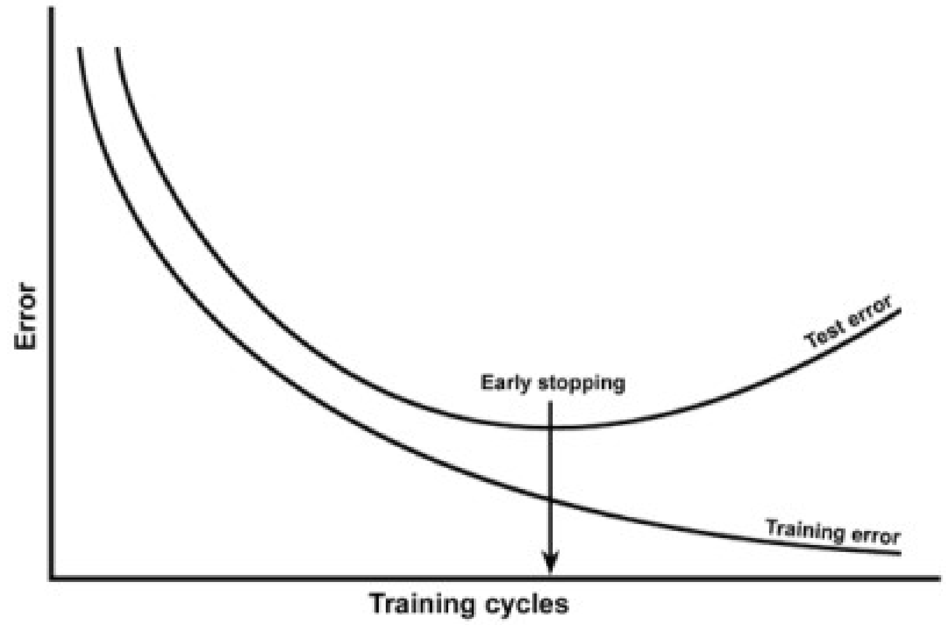
- Any function with a derivative may work
- Many activation to pick from (and there are more, like cos, ...)
- Sigmoid, tanh suffer from vanishing gradients : slow convergence
- Relu and PRelu solve some of the vanishing gradient issue, and accelerate computation

Regularization

- “With four parameters I can fit an elephant, and with five I can make him wiggle his trunk.” *John Von Neumann*
- Add terms in the loss function to reduce the amount parameter actively used
- Prevents overfitting the data and improves generalization
- Caveat : regularization strength needs to be tuned



Early Stopping

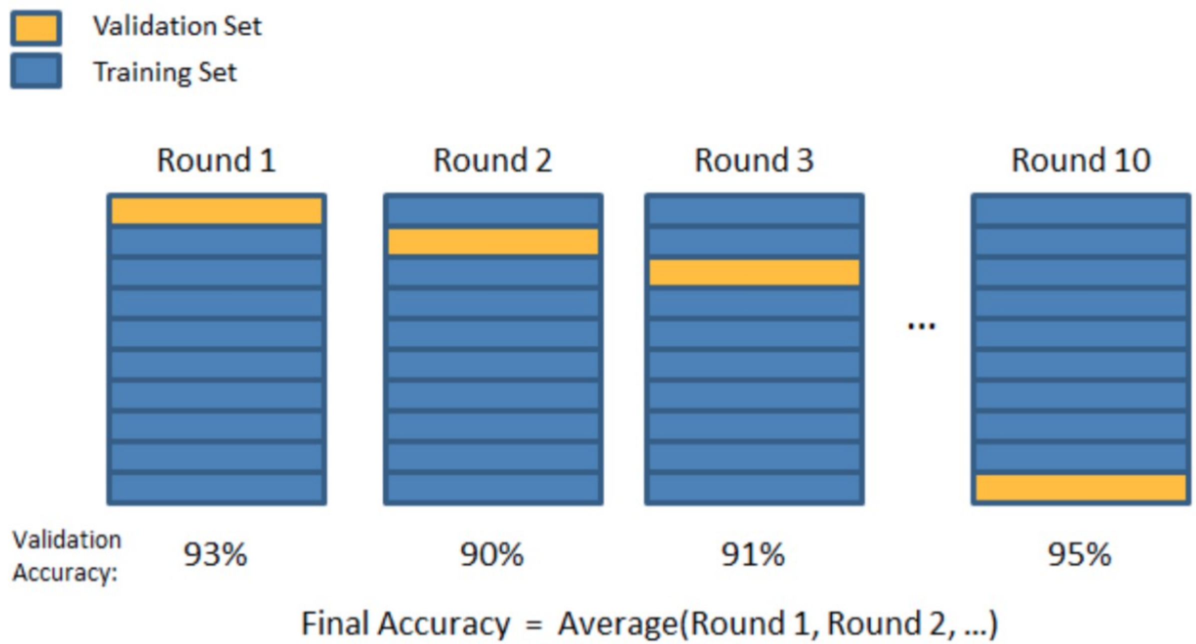


- Even regularized infinite training might lead to overfitting
- Loose generalizing power
- Stop training when generalization performance stabilizes
- Be careful of “choppy” test error, needs averaging
- Bias towards the test sample is minimal. However better to do it on a fraction of the train sample

Batch Size

- Batch≡"stochastic" in stochastic gradient descent
- Batch size = 1
  - ➢ Weights move too much towards each sample
  - ➢ Noisy gradients
  - ➢ Computationally expensive
- Increasing batch size
  - ✔ Speed up by using parallelism
  - ✗ Slow down due to lack of update cycle
- In theory, would need to be tuned
  - ➢ Not practical as one of the aspect is speed-up
  - ➢ Can be optimize with a couple of epochs based on Δloss/s metric
- Often does not have a effect on converged model
- Adaptive batch size https://arxiv.org/abs/1712.02029 can bring faster convergence

K-Folding



- 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

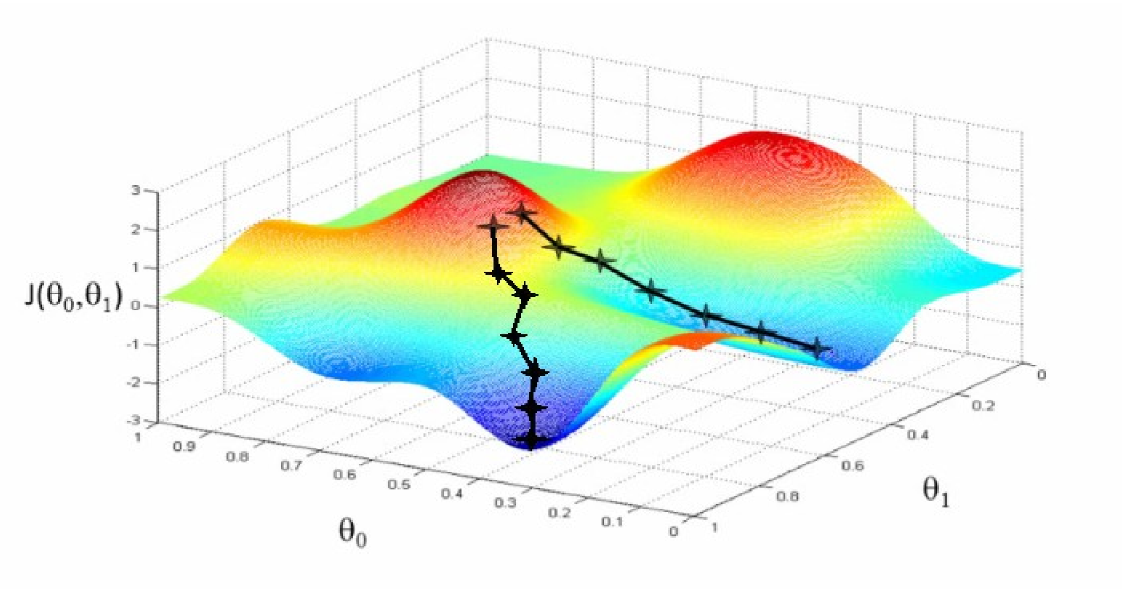
http://scikit-learn.org/stable/modules/cross_validation.html

Loss Function

- Any differentiable function can define the loss function
- Canonical functions
  - Categorization : https://arxiv.org/abs/1702.05659
    binary cross entropy for binary classification
    categorical cross entropy for >2 category
  - Regression :
    mean squared errors (mse) is common

- Choice of the loss implies an assumption on the distribution of the data and how loosely similar a pair of sample are
- Any loss definition over a training batch is allowed
- Can consider combining neural net outputs into a loss without having a target value for each output

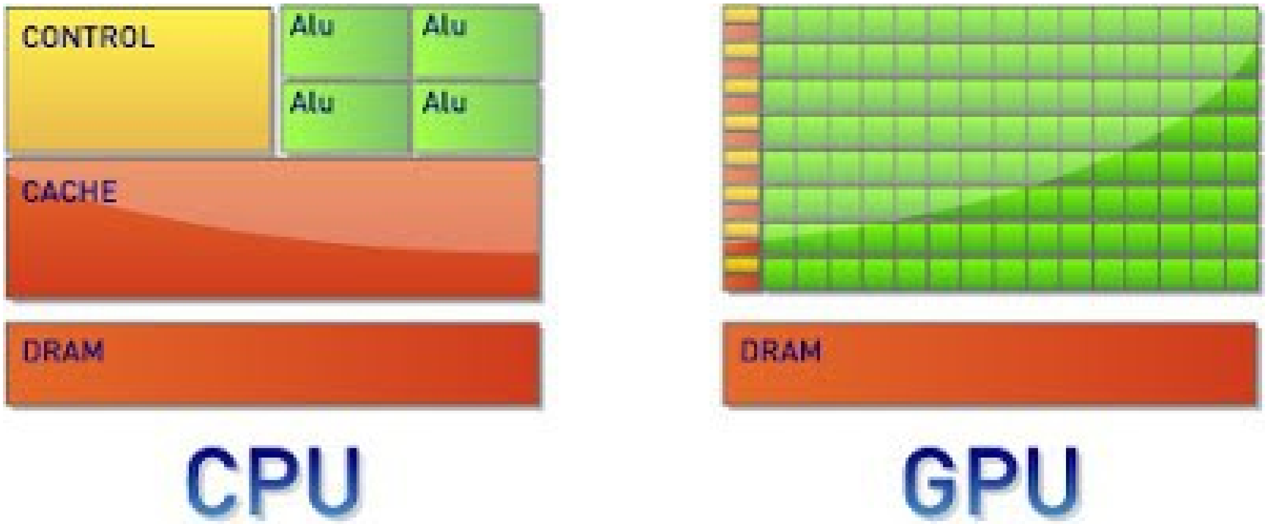
Optimizer



- Loss function has a highly non convex high dimension landscape
- Presence of multiple saddle points and local minimum
- Simple SGD (gradient average over batch) converges poorly on complex models
- Varieties of optimizer, with various characteristics http://ruder.io/optimizing-gradient-descent/
- Best practice is use Adam, with tuning of learning hyper-parameters

GP-GPU

https://sites.google.com/site/computationvisualization/programming/cuda/article1



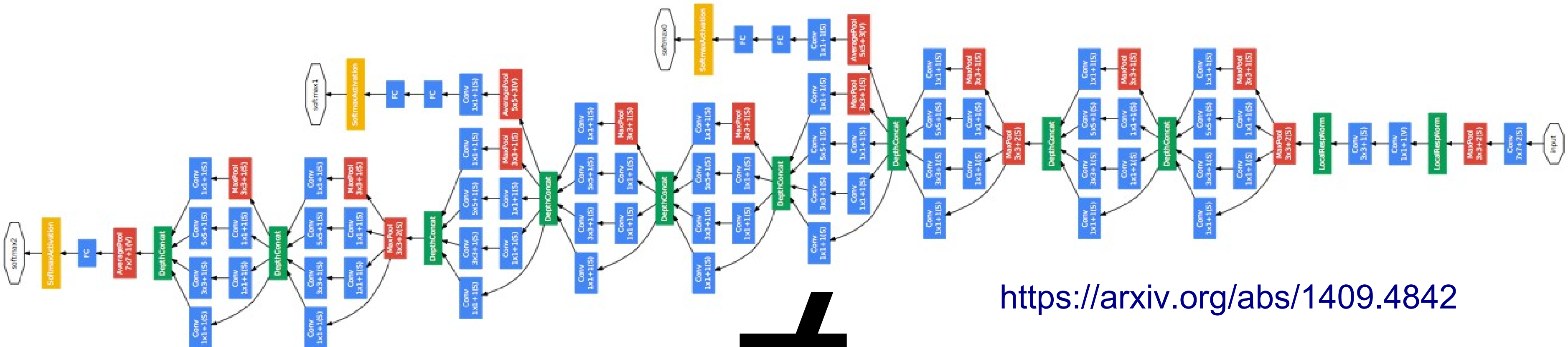
- CPU : optimized for multiple different sequential operations

- GPU : optimized for multiple identical parallel operations

CPU hardware are bridging the gap somehow (e.g KNL)
GPU are exponentially growing in FOPS
→ P100 : 21 ½, 10 single, 5 ddouble TeraFLOPS

➢ Most operations in training neural net are naturally parallel and therefore particularly suited for computation on GPU

Brain Inspiration

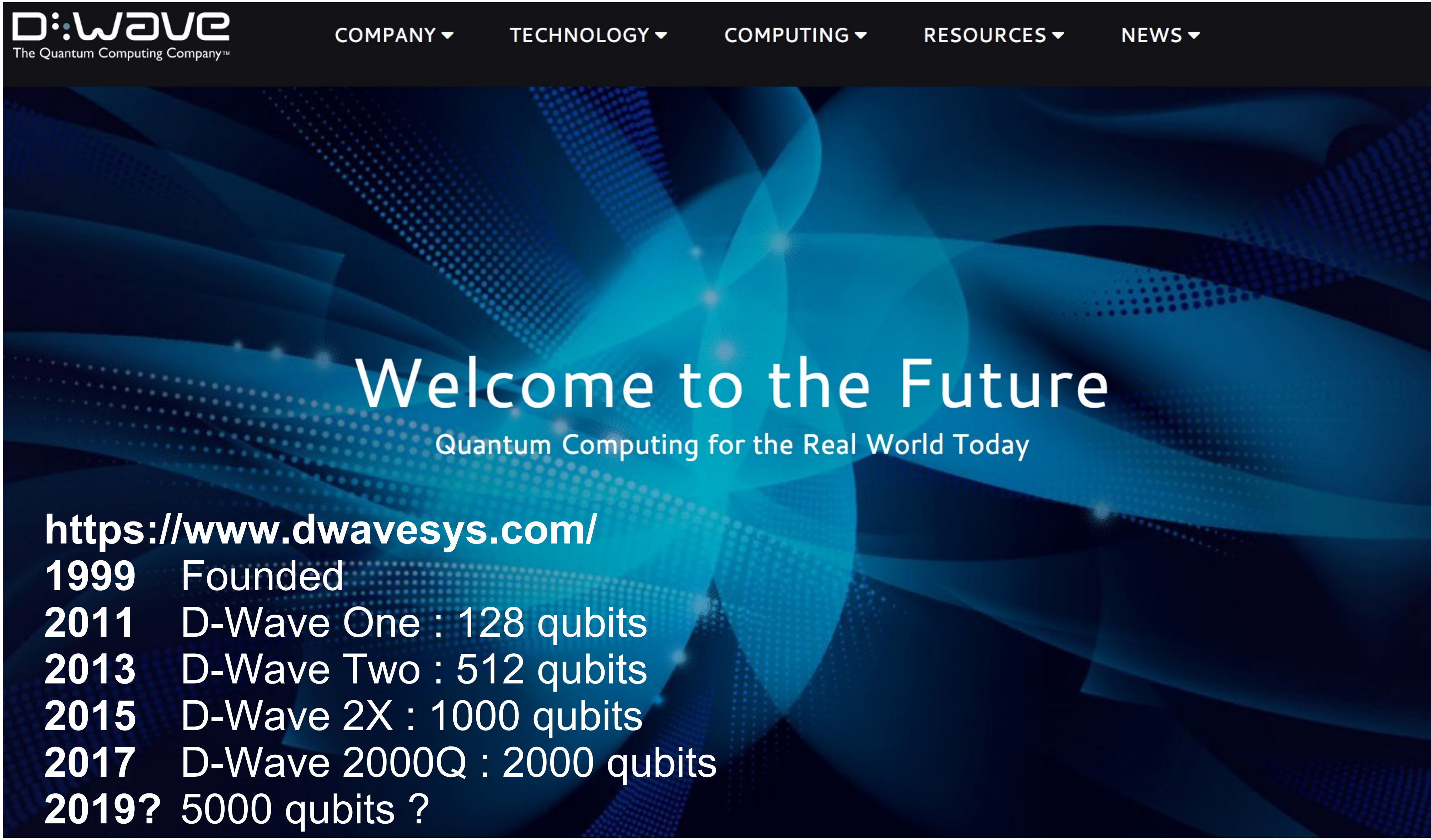


https://arxiv.org/abs/1409.4842

≠

- ANN are brain-inspired, but have no biological analogy
- Spiking neural nets are closer to reality

The D-Wave Company

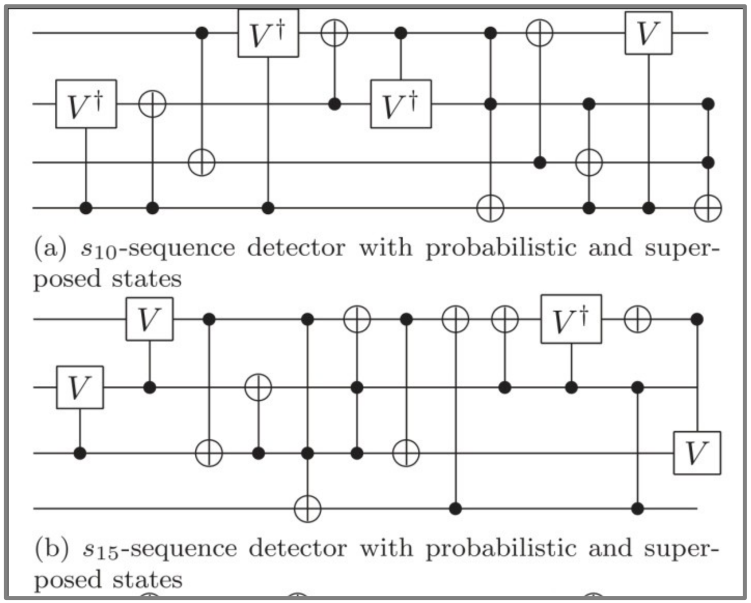


https://www.dwavesys.com/

1999 Founded
2011 D-Wave One : 128 qubits
2013 D-Wave Two : 512 qubits
2015 D-Wave 2X : 1000 qubits
2017 D-Wave 2000Q : 2000 qubits
2019? 5000 qubits ?

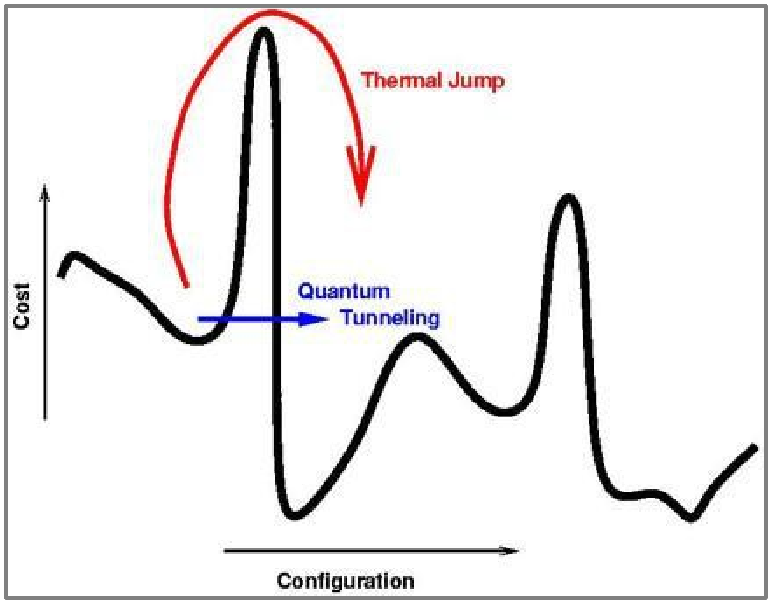
D-Wave 2X^{TM}

qubit and qubit



<u>Quantum Circuits</u>

Series of quantum gates operating on a set of quantum states.

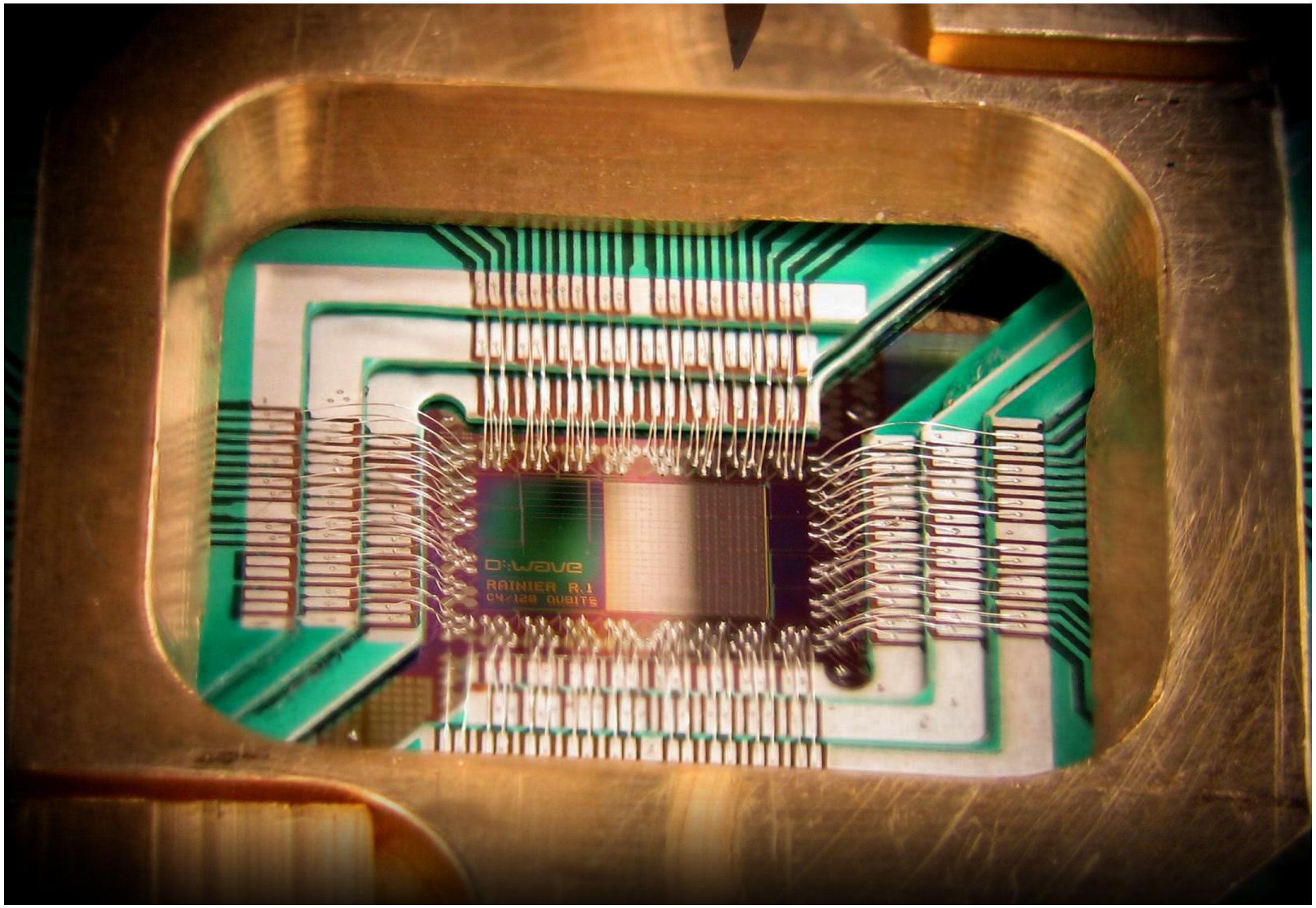


<u>Quantum Annealing</u>

Evolution of a quantum system to a low T Gibbs state

That's D-Wave !

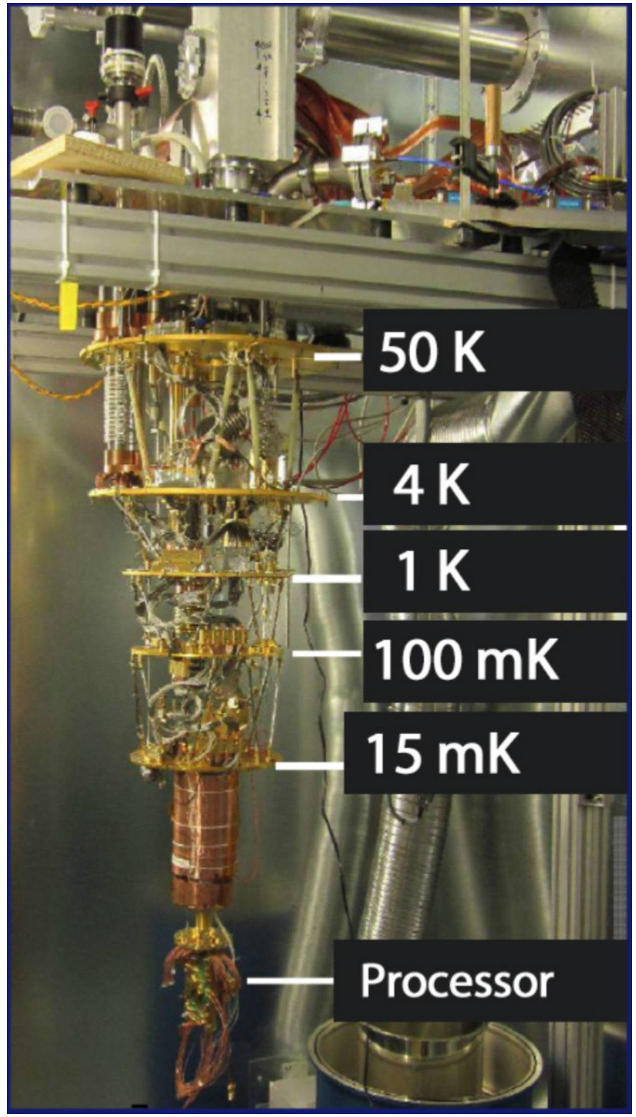
D-Wave's quBit



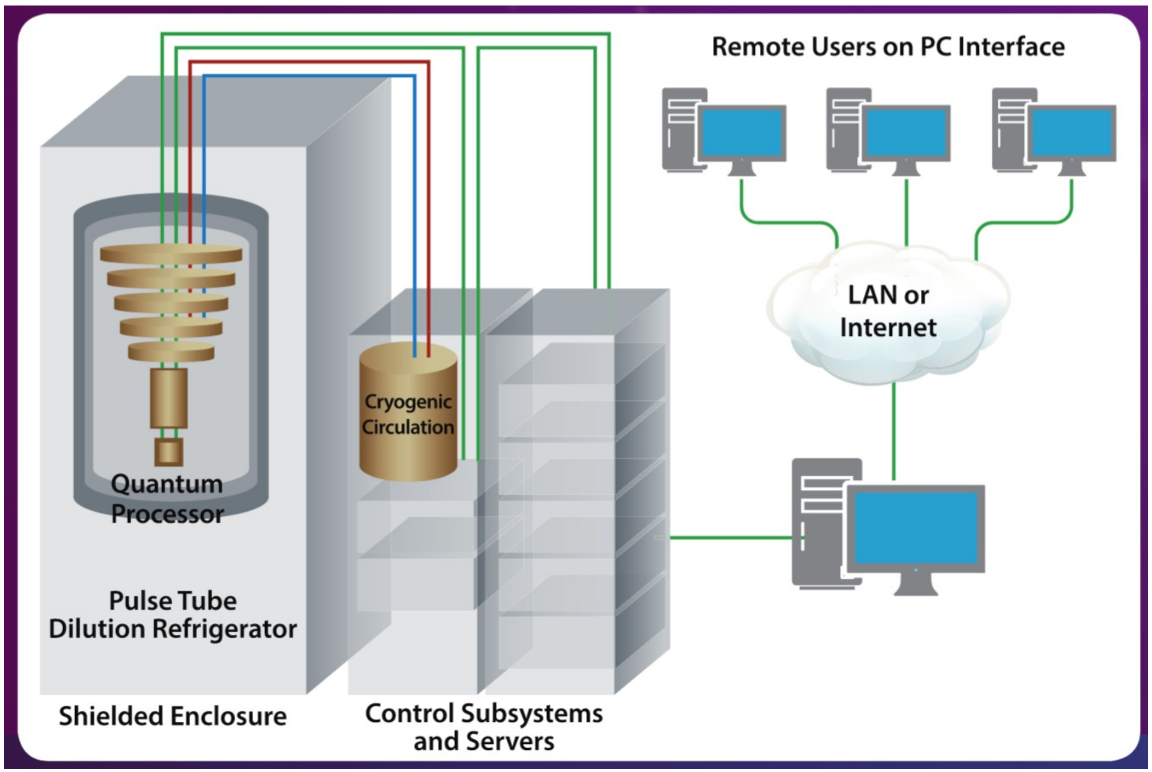
- Each qubit is a pair of Josephson junction (JJ)
- Able to apply local magnetic fields with programable digital-to-analog flux converters (DAC)
- Operates at 15 mK to remove noise

https://doi.org/10.1109/TASC.2014.2318294

Thermal Noise Isolation



Working on a D-Wave



- Web Interface to post the problem settings (Hp).
- Asynchronous processing.
- Solution is made available for download.

- Distributed library for performing embedding
  - Retain full intellectual property.

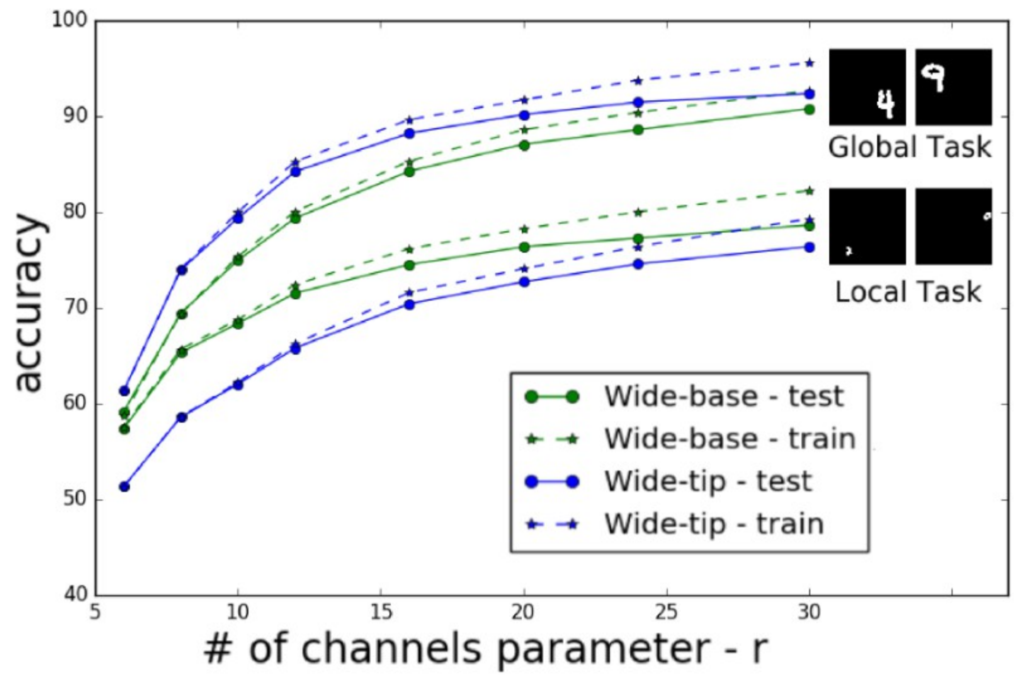
- Equivalent restapi to submit and retrieve solutions
  - D-Wave processor as a service

DL and Quantum Entanglement

Yoav Levine , Hebrew University

Correlations ⟷ Min-Cut over Layer Widths

Veryfied on common ConvNets (Relu activations & max - pooling):

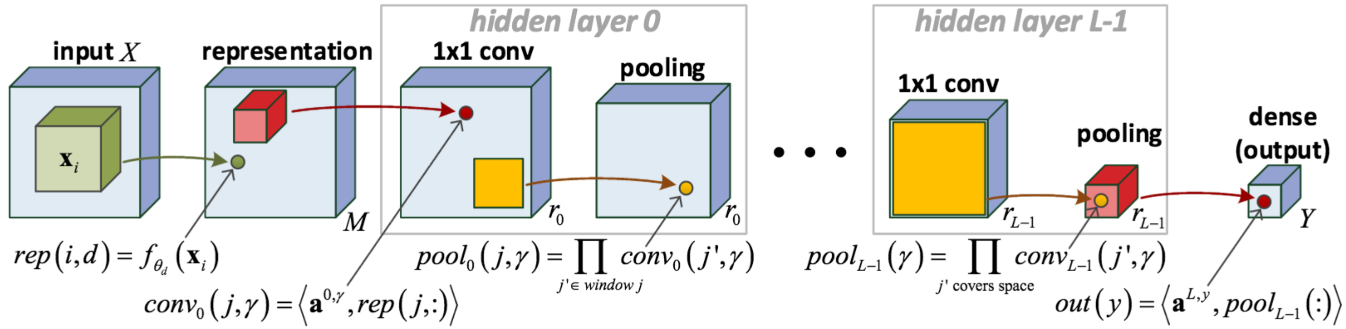


- ConvAC : Convolutional arithmetic circuit (a specific NN architecture)
- Equivalence to many-body quantum wave function : loosely used **IMO**

Expressiveness of Deep Networks

Amnon Shashua, HUJI

Convolutional Arithmetic Circuits: Baseline Architecture



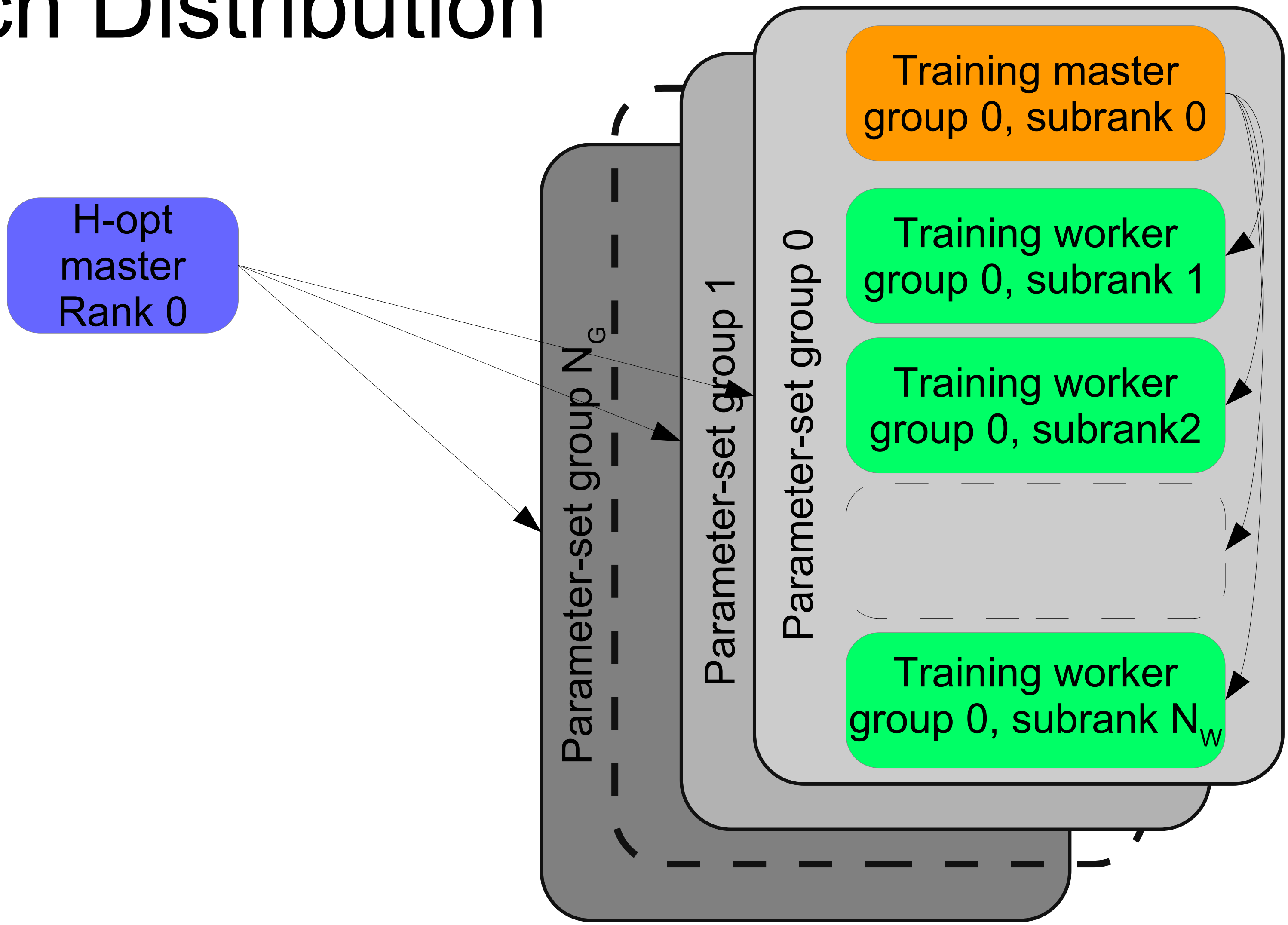
Baseline ConvAC architecture:

- *Linear activation* ($\sigma(z) = z$), *product pooling* ($P\{c_j\} = \prod_j c_j$)
- 1×1 convolution windows (non-overlapping convolution: stride = kernel size).

Intimate relationship to math machinery: tensor analysis, measure theory, functional analysis and graph theory.

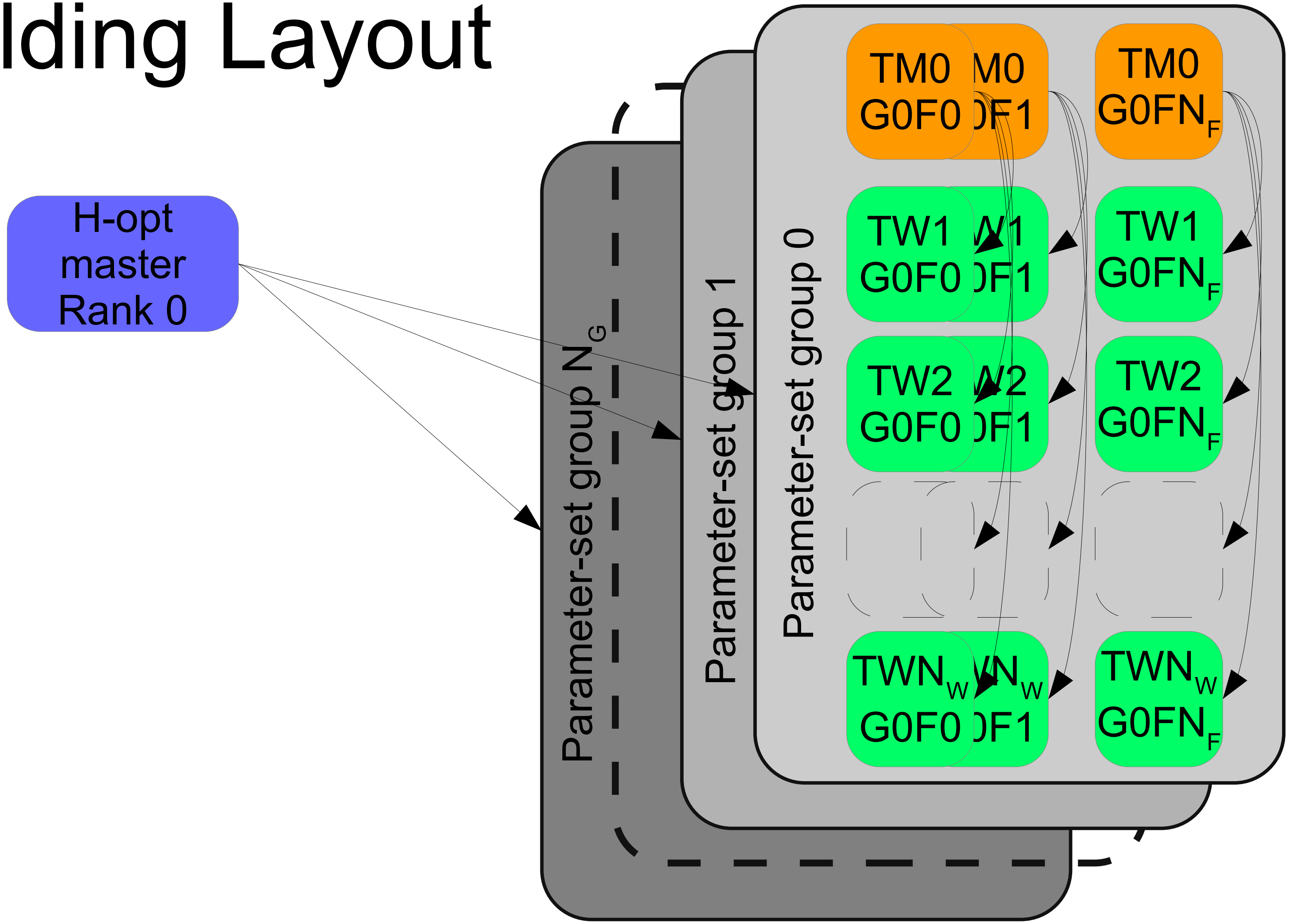
- ConvAC : Convolutional arithmetic circuit (a specific NN architecture)
- Theoretical proof of intuitive behaviors when changing NN architectures

Batch Distribution



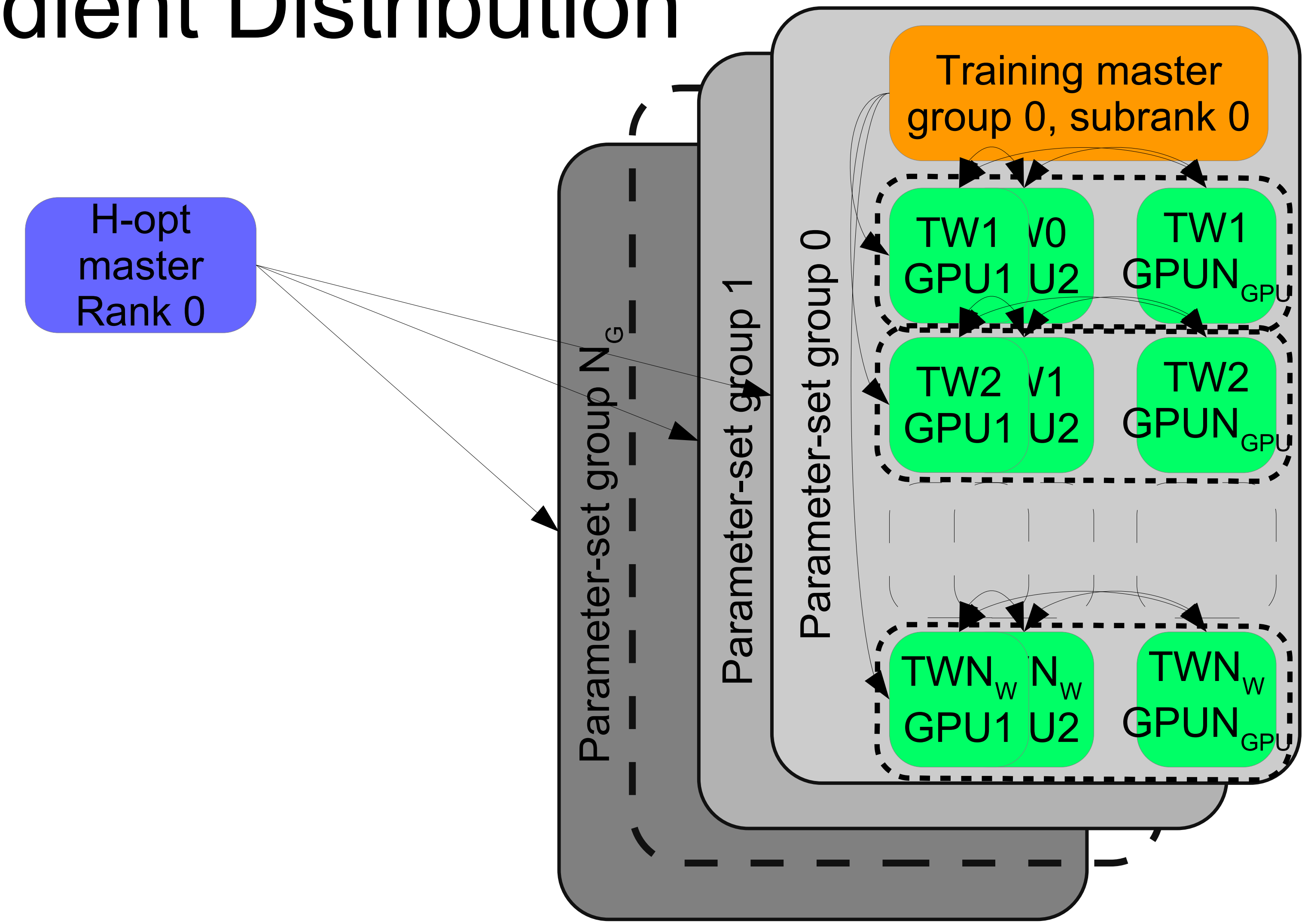
- One master process drives the hyper-parameter optimization
- N_G groups of nodes training on a parameter-set on simultaneously
  - One training master
  - N_W training workers

K-folding Layout



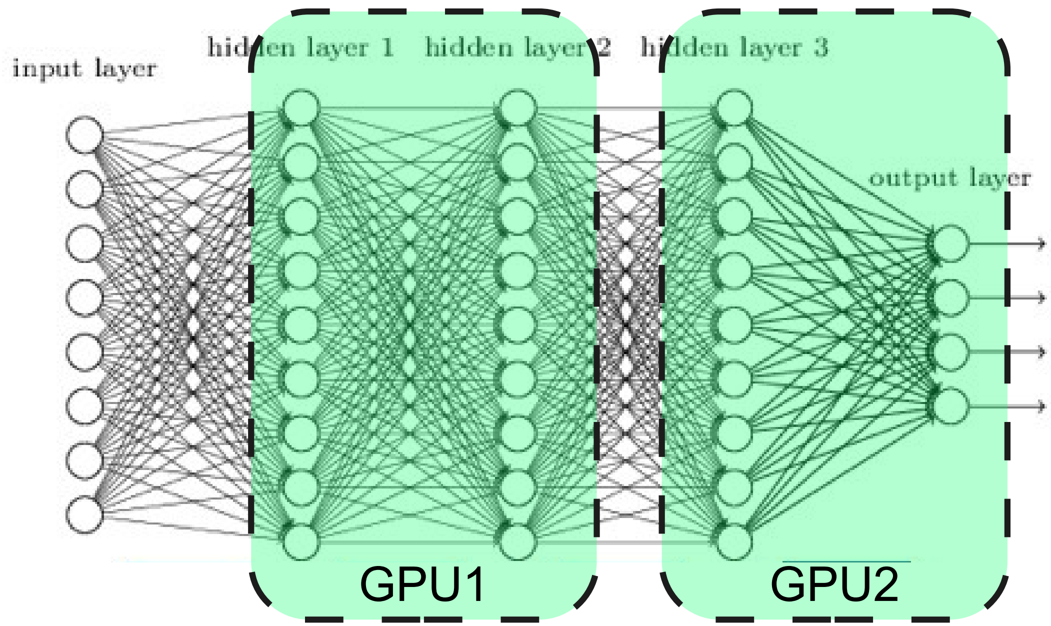
- One master running the optimization. Receiving the average figure of merit over N_F folds of the data
  - N_G groups of nodes training on a parameter-set on simultaneously
    - N_F groups of nodes running one fold each

Gradient Distribution



- One master running the bayesian optimization
- N_G groups of nodes training on a parameter-set on simultaneously
  - One training master
  - N_W training worker groups
    - N_{GPU} used for each worker group (either nodes or gpu)

Model Parallelism



- Perform the forward and backward pass of sets of layers on different devices
- Require good device to device communication
- Aiming for machines with multi-gpu per node topology (summit)