Applications of Deep Learning to High Energy Physics

Amir Farbin

UTADL Account

- Request an account: https://www.utadl.org
- Log into jupyter.
- Mac/Linux:
  - Open terminal.
  - Type: (replace afarbin with your username)

```
ssh -NfL 8000:127.0.0.1:8000 afarbin@orodruin.uta.edu
```

  - Point your browser to 127.0.0.1:8000
- Windows:
  - Download plink.exe
    - http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html
  - Open command prompt
  - Type: (replace afarbin with your username)

```
plink.exe -N -L 8000:127.0.0.1:8000 afarbin@orodruin.uta.edu
```

  - Point your browser to 127.0.0.1:8000

Goal

- Understand ***Context***:
    - ***Hammer***: Deep Learning
        - What is possible?
    - ***Nail***: High Energy Physics
        - What are the problems?
- ***Practical Knowledge:***
    - Initiative Understanding:
        - Feynman Diagram Like understanding of Deep Learning
        - High Level. No Details… trust that it works.
    - Data sets: so you can play…
    - Technical: Software and Hardware
- I just want you to know what is possible… so you know what to google to learn more…

Menu

1. Introduction
    - What is Deep Learning?
    - DL in HEP?
    - What physicists bring to DL?
    - DL in HEP Big Picture
    - A bit about detectors…
    - From Data to Physics
    - HEP Problems

2. “Lecture”
    - DL Basics
    - DL Techniques & Examples in HEP
        - Feature Learning
            - HEP Searches (SUSY Example)
        - Recurrent NNs
        - Unsupervised Learning
        - Generative Models
    - Special Topics: Calorimetry & Jet Physics

3. Hands on…
    - Example Datasets
    - DL Software & Technical Challenges
    - DL Model Components

Menu

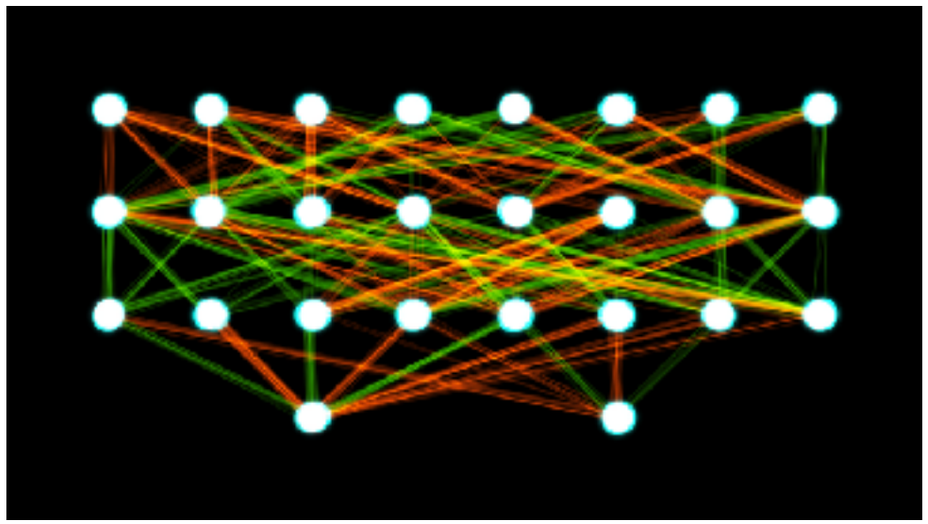
- What is Deep Learning?
- DL in HEP?
- What physicists bring to DL?
- DL in HEP Big Picture
- A bit about detectors…
- From Data to Physics
- HEP Problems

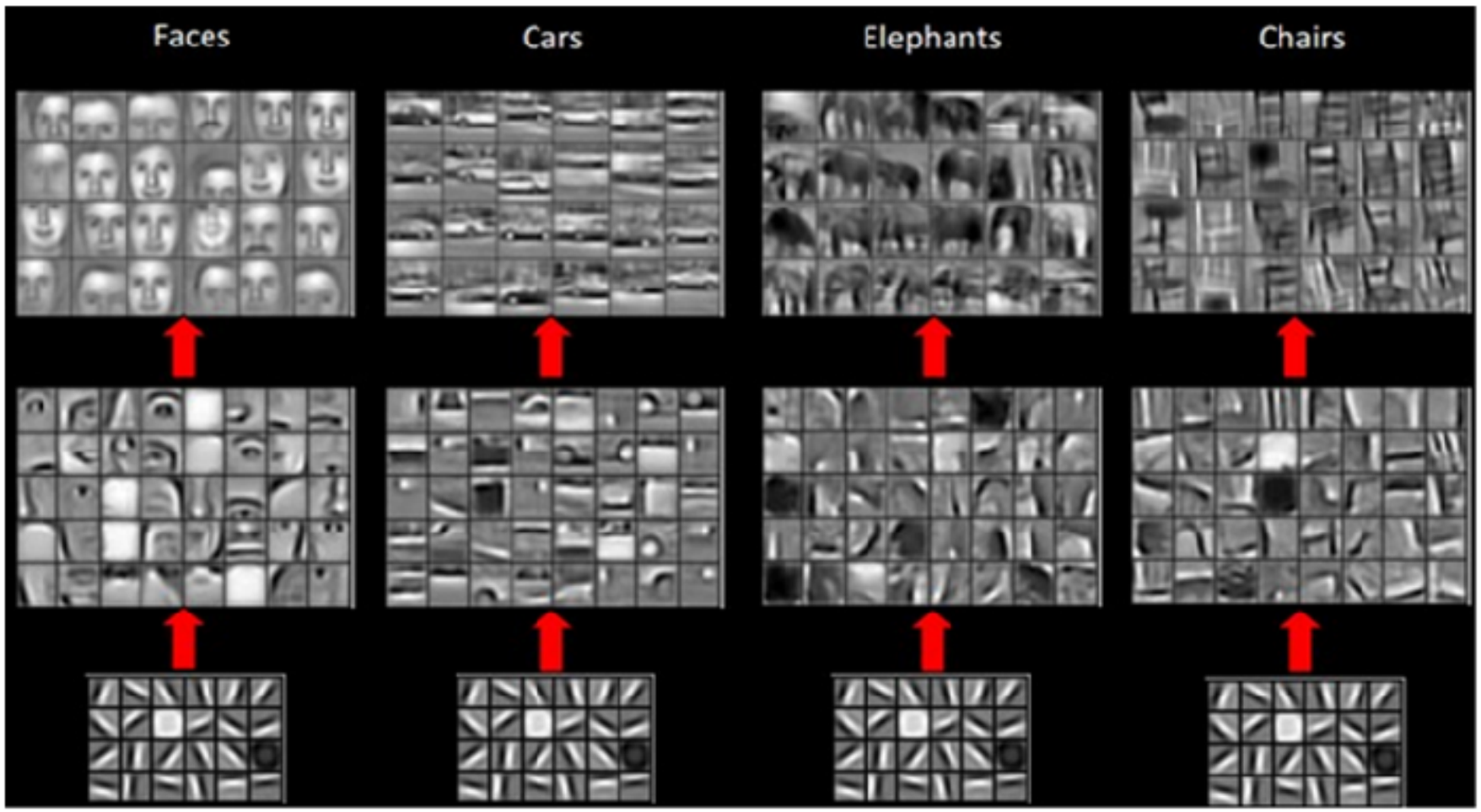
What is Deep Learning?

Animal Brains

- The ***brain takes in sensory data… builds hierarchical models*** of the world. For example:
  - Cells in visual cortex respond specific ***low level features***, like contrast in color, or vertical or horizontal lines
  - Other cells combine low level features to identify ***higher level features***, e.g. specific shapes
  - Pattern repeats until get to recognition of actual objects, e.g. chairs.
- So effectively, a ***representation*** of the input is assembled in the brain.
  - Eyes see a limited window… but they scan around and establish a ***model*** of surrounding.
  - These include maps of the environment, including cells that light up when we are at specific locations.
    - Recently reproduced with DL by DeepMind
  - Same cells light up when we imagine (~ simulate?) a specific object or location.
- When making decisions, we use these models to predict outcomes of actions.

Deep Neutral Networks



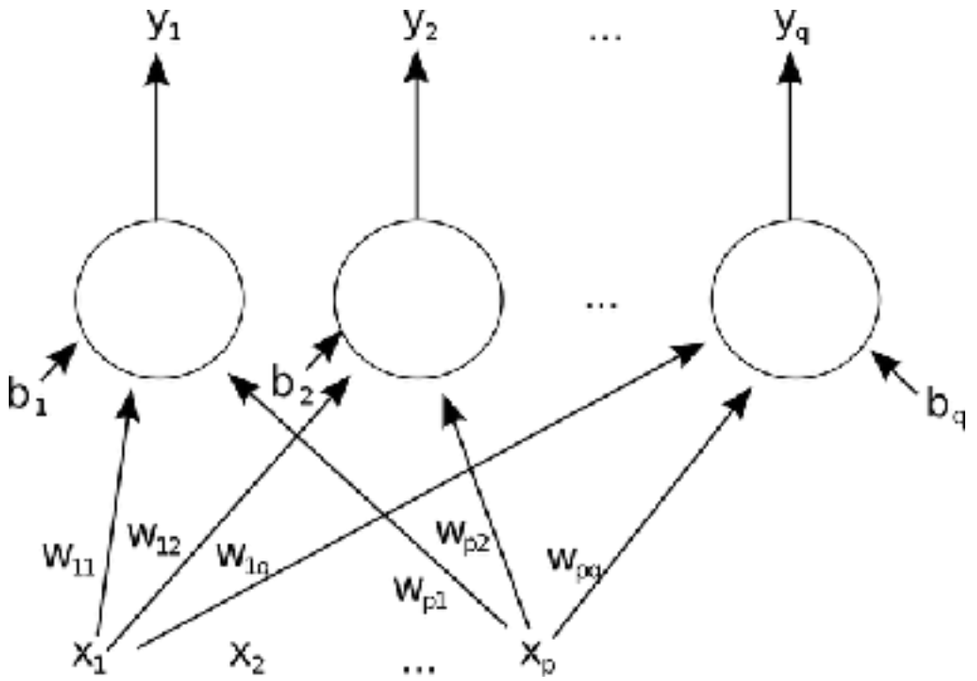


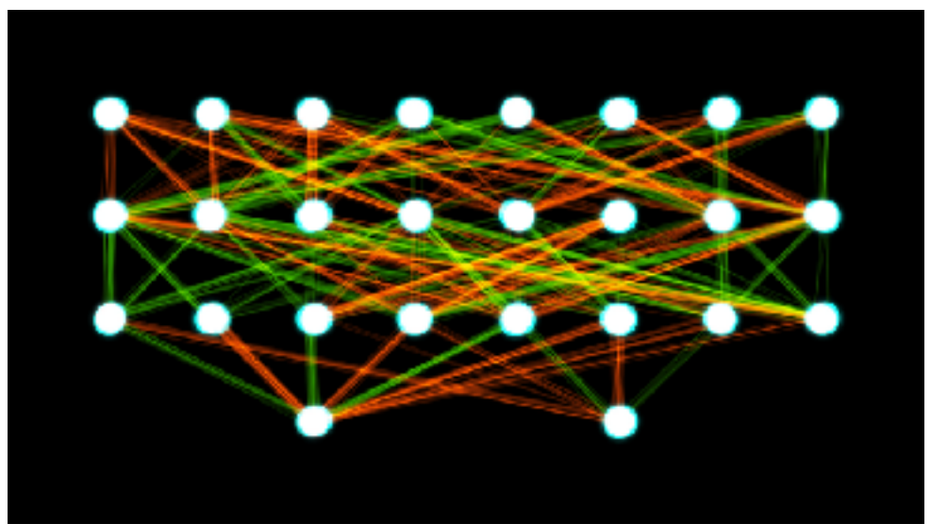
Recent History

- Deep Learning feats that sparked broad interest:
  - 2012, Google 1B DNN learns to identify cats (and 20000 other types of objects) (Wired Article, paper)
    - ***Raw input:*** trained with 200x200 pixel images from YouTube
    - ***Unsupervised***: the pictures were unlabeled.
    - Google cluster 16000 cores ~ $1M. Redone with $20k system with GPUs.
- 2013: Deep Mind builds AI that plays ATARI (Blogpost, Nature,YouTube,YouTube)

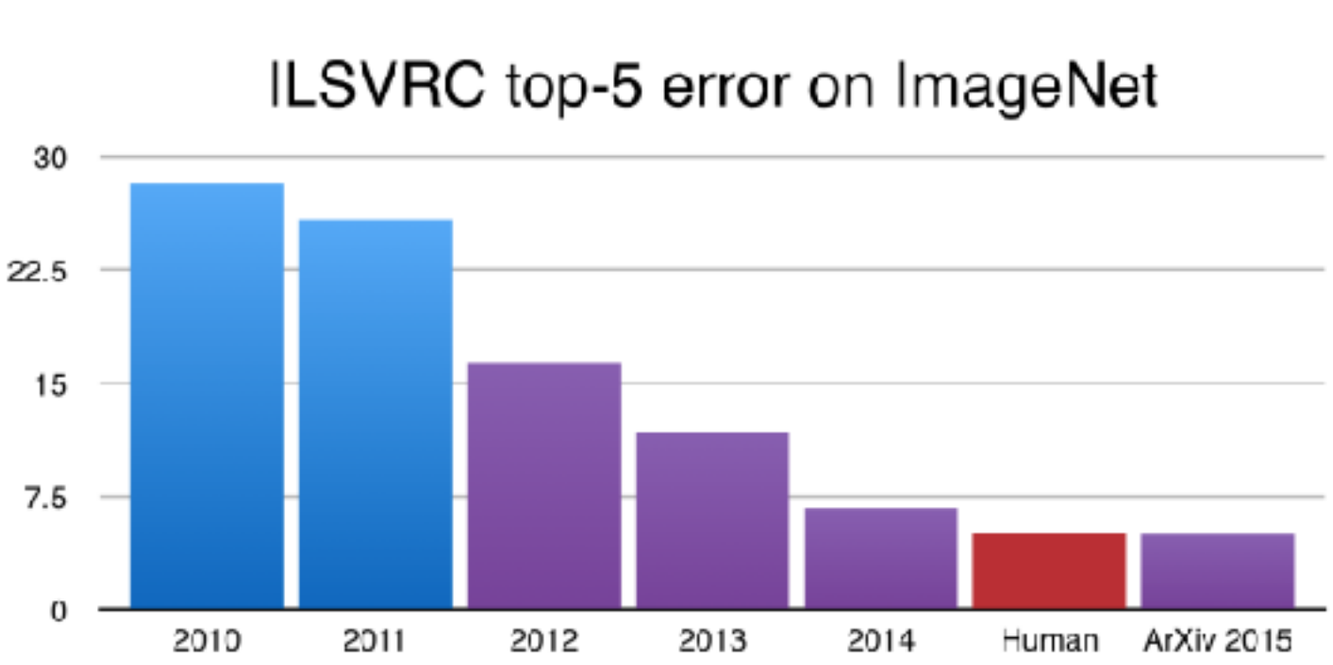
Artificial Neural Networks

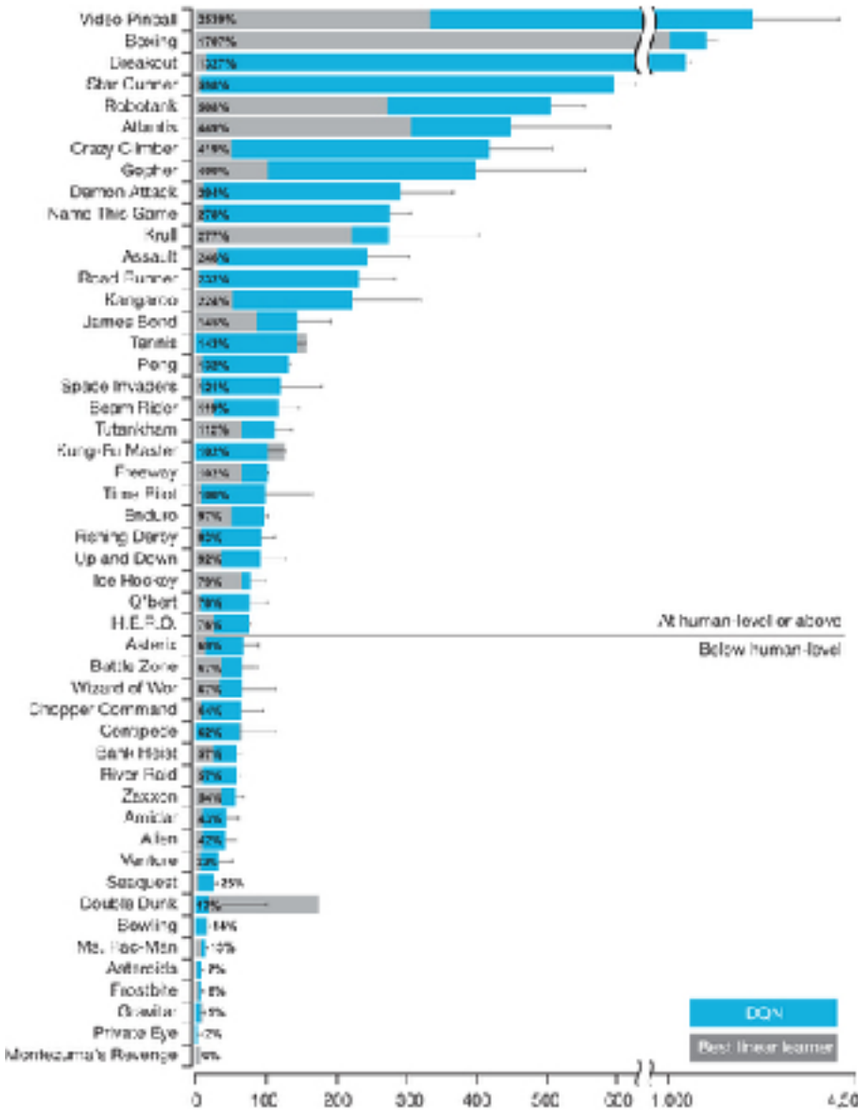
- ***Biologically inspired computation***, (first attempts in 1943)
  - ***Probabilistic Inference***: e.g. signal vs background
  - ***Universal Computation Theorem*** (1989)
- Multi-layer (***Deep***) Neutral Networks:
  - Not a new idea (1965), just impractical to train. ***Vanishing Gradient problem*** (1991)
  - Solutions:
    - New techniques: e.g. better activation or layer-wise training
    - ***More training***: big training datasets and lots of computation … ***big data and GPUs***
- ***Deep Learning Renaissance***. First DNN in HEP (2014).
- ***Amazing Feats***: Audio/Image/Video recognition, captioning, and generation. Text (sentiment) analysis. Language Translation. Video game playing agents.
- ***Rich field***: Variety of architectures, techniques, and applications.



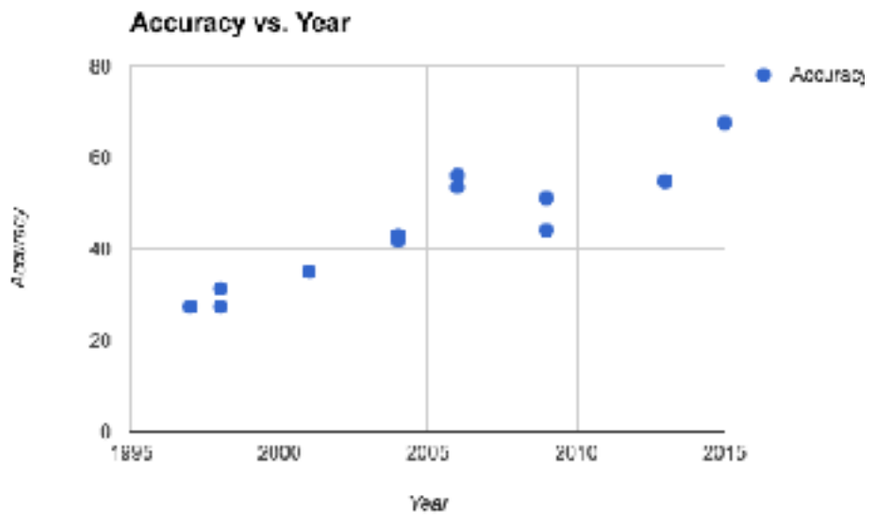


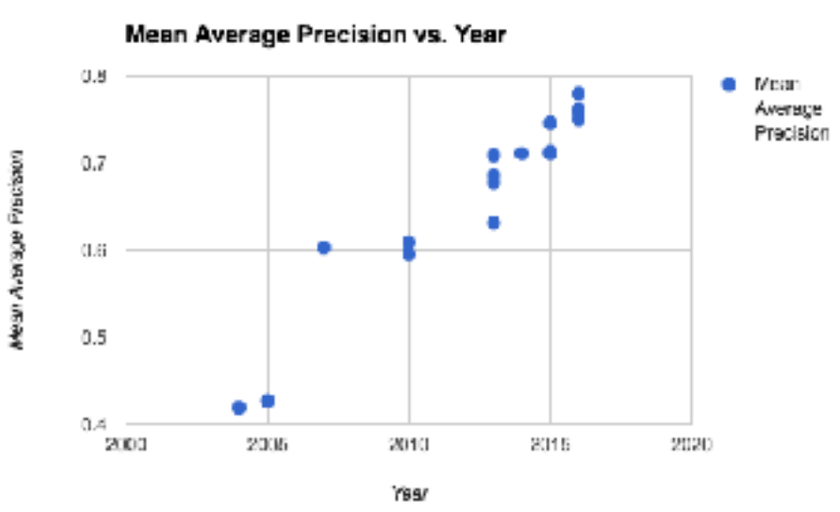
Images from Wikipedia

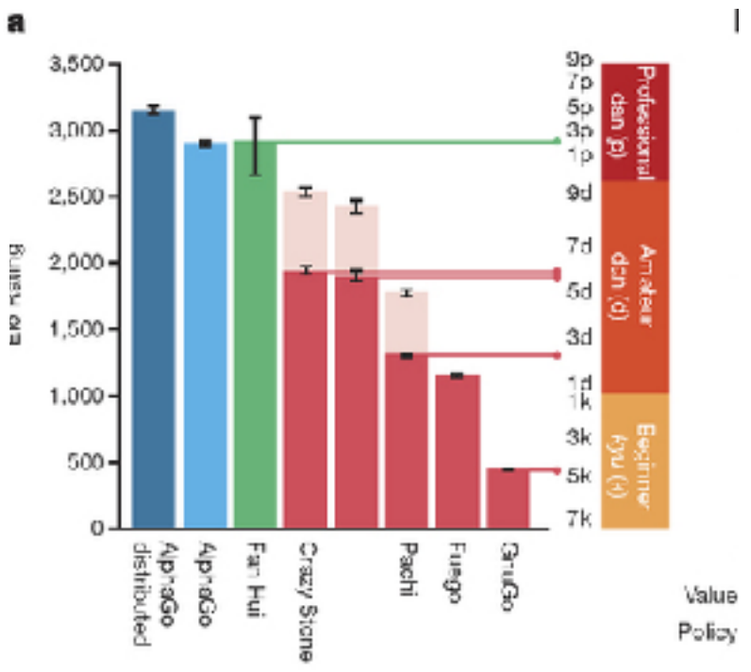




- Continuous server ASR word error rate (WER) reduction ~18% / year: combination of algorithms, data, and computing
- Deep learning (DNNs) is driving recent performance improvements in ASR and meaning extraction



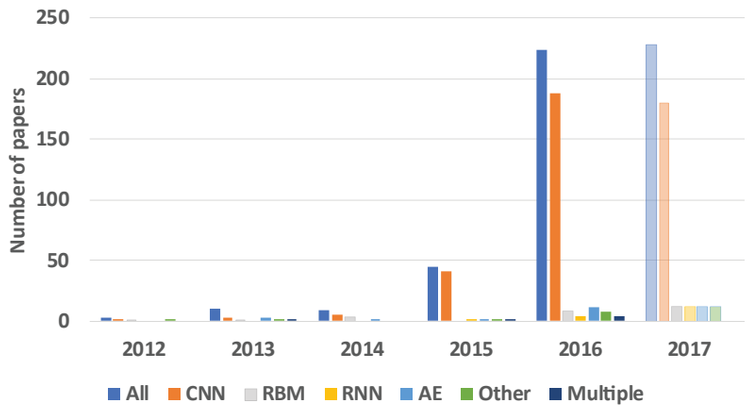


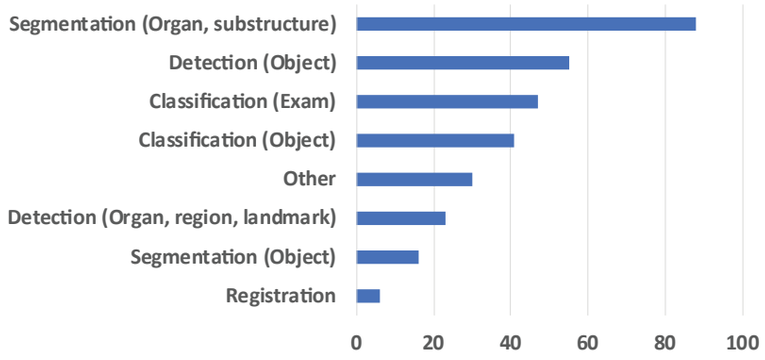


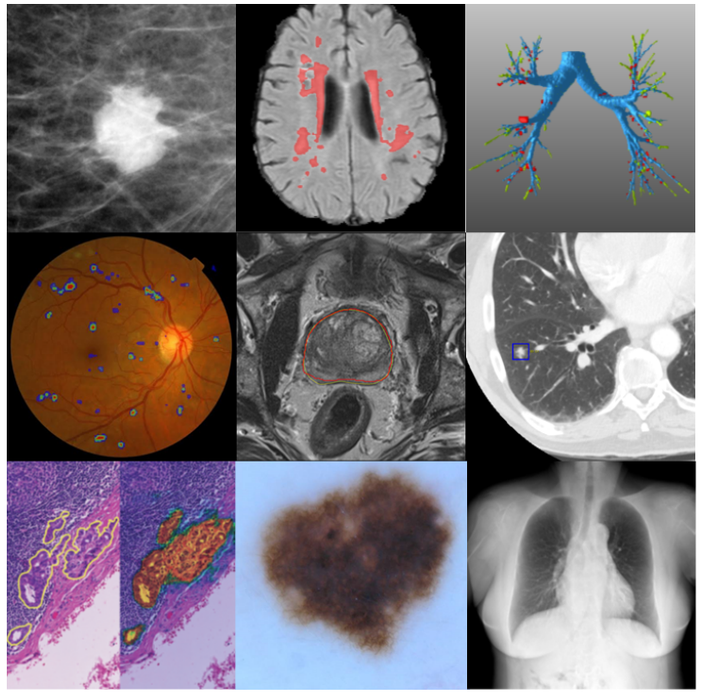
A Survey on Deep Learning in Medical Image Analysis

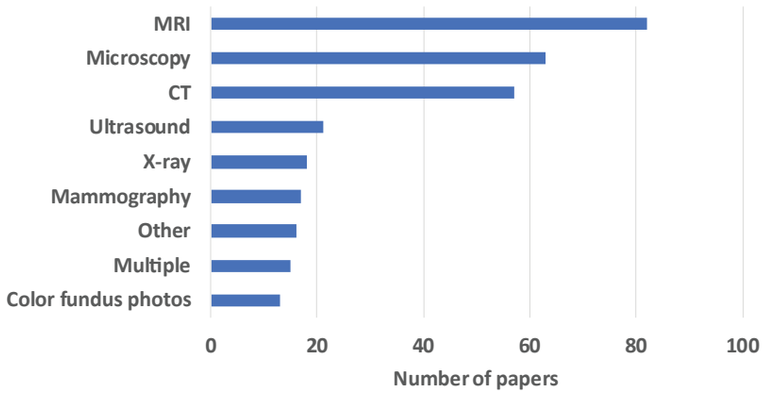
Geert Litjens, Thijs Kooi, Babak Ehteshami Bejnordi, Arnaud Arindra Adiyoso Setio, Francesco Ciompi,
Mohsen Ghafoorian, Jeroen A.W.M. van der Laak, Bram van Ginneken, Clara I. Sánchez

Diagnostic Image Analysis Group
Radboud University Medical Center
Nijmegen, The Netherlands









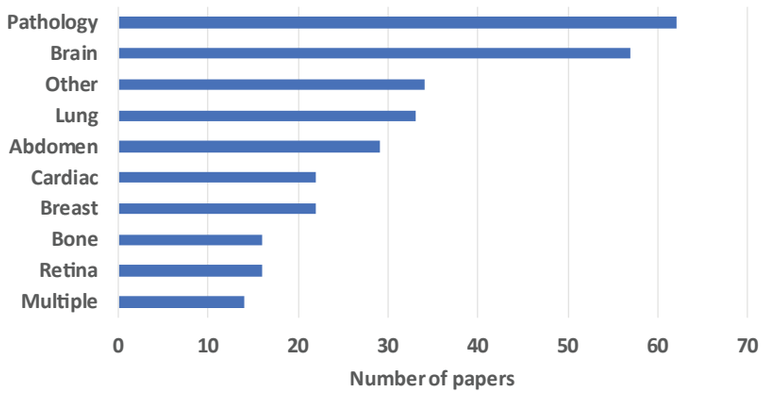


Figure 3: Collage of some medical imaging applications in which deep learning has achieved state-of-the-art results. From top-left to bottom-right: mammographic mass classification (Kooi et al., 2016), segmentation of lesions in the brain (top ranking in BRATS, ISLES and MRBrains challenges, image from Ghafoorian et al. (2016b), leak detection in airway tree segmentation (Charbonnier et al., 2017), diabetic retinopathy classification (Kaggle Diabetic Retinopathy challenge 2015, image from van Grinsven et al. (2016), prostate segmentation (top rank in PROMISE12 challenge), nodule classification (top ranking in LUNA16 challenge), breast cancer metastases detection in lymph nodes (top ranking and human expert performance in CAMELYON16), human expert performance in skin lesion classification (Esteva et al., 2017), and state-of-the-art bone suppression in x-rays, image from Yang et al. (2016c).

https://arxiv.org/pdf/1702.05747.pdf

Captioning

(a) MSVD dataset

Sentences:
- A dog walks around on its front legs.
- The dog is doing a handstand.
- A pug is trying for balance walk on two legs.

Sentences:
- A man lights a match book on fire.
- A man playing with fire sticks.
- A man lights matches and yells.

(b) M-VAD dataset

Sentence:
- Later he drags someone through a jog.

Sentence:
- A waiter brings a pastry with a candle.

(c) MPII-MD dataset

Sentence:
- He places his hands around her waist as she opens her eyes.

Sentence:
- Someone's car is stopped by a couple of uniformed police.

(d) MSR-VTT-10K dataset

Sentences:
- People practising volleyball in the play ground.
- A man is hitting a ball and he falls.
- A man is playing a football game on green land.

Sentences:
- A cat is hanging out in a bassinet with a baby.
- The cat is in the baby bed with the baby.
- A cat plays with a child in a crib.

https://arxiv.org/pdf/1609.06782.pdf

Deep Feature Interpolation for Image Content Changes

Paul Upchurch[1,2], Jacob Gardner[1,2], Kavita Bala[2], Robert Pless[3], Noah Snavely[2], and Kilian Weinberger[2]

[1]Authors contributed equally
[2]Cornell University
[3]Washington University in St. Louis

Abstract

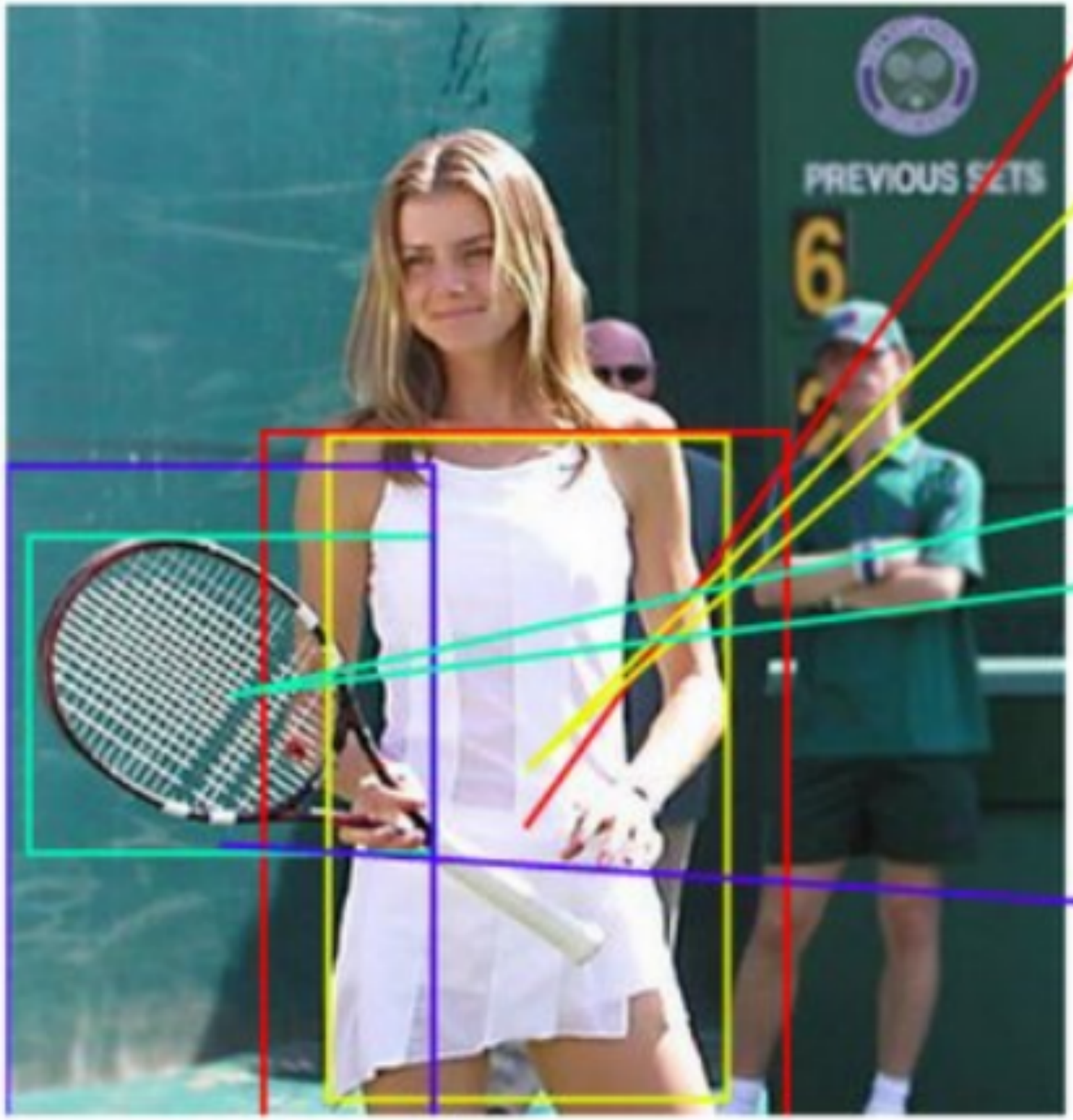
We propose Deep Feature Interpolation (DFI)*, a new data-driven baseline for automatic high-resolution image transformation. As the name suggests, it relies only on simple linear interpolation of deep convolutional features from pre-trained convnets. We show that despite its simplicity, DFI can perform high-level semantic transformations like "make older/younger", "make bespectacled", "add smile", among others, surprisingly well—sometimes even matching or outperforming the state-of-the-art. This is particularly unexpected as DFI requires no specialized network architecture or even any deep network to be trained for these tasks. DFI therefore can be used as a new baseline to evaluate more complex algorithms and provides a practical answer to the question of which image transformation tasks are still challenging in the rise of deep learning.*

Style Transfer

DeepFakes

https://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/Gatys_Image_Style_Transfer_CVPR_2016_paper.pdf

DL in HEP?



1.12 woman

-0.28 in

1.23 white

1.45 dress

0.06 standing

-0.13 with

3.58 tennis

1.81 racket

0.06 two

0.05 people

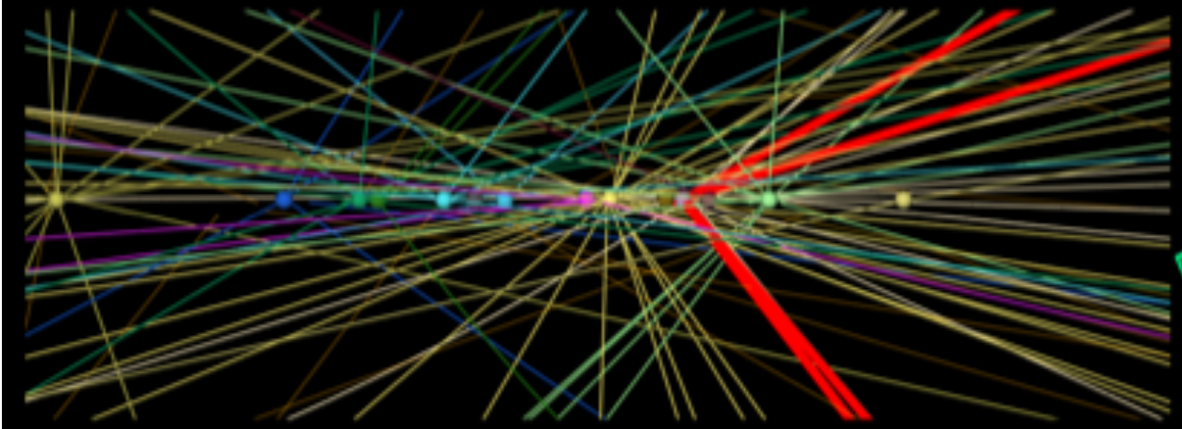
-0.14 in

0.30 green

-0.09 behind

-0.14 her

$H \to ZZ \to 4l$



ATLAS
EXPERIMENT
http://atlas.ch
Run: 204769
Event: 71902630
Date: 2012-06-10
Time: 13:24:31 CEST

What can physicists bring to DL?

What can physicists bring to DL?

- Data
- Interesting problems…

https://www.wired.com/2017/01/move-coders-physicists-will-soon-rule-silicon-valley/

MOVE OVER, CODERS— PHYSICISTS WILL SOON RULE SILICON VALLEY

Most of the graduate students and postdocs in this room are probably headed in this direction…

it's happening across Silicon Valley. Because structurally and technologically, ***the things that just about every internet company needs to do are more and more suited to the skill set of a physicist***.

But this is a particularly ripe moment for physicists in computer tech, thanks to the rise of machine learning, where machines learn tasks by analyzing vast amounts of data. This ***new wave of data science and AI is something that suits physicists right down to their socks***.

these neural networks are really just math on an enormous scale, mostly linear algebra and probability theory.

Chris Bishop, who heads Microsoft's Cambridge research lab, … ***"There is something very natural about a physicist going into machine learning," he says, "more natural than a computer scientist."***

Physicists know how to handle data—at MIT, Cloudant's founders handled massive datasets from the the Large Hadron Collider—and ***building these enormously complex systems requires its own breed of abstract thought.***

They come because they're suited to the work. And they come because of the money. As Boykin says: "The salaries in tech are arguably absurd." But they also come because there are so many hard problems to solve.

Machine learning will change not only how the world analyzes data but how it *builds software*.

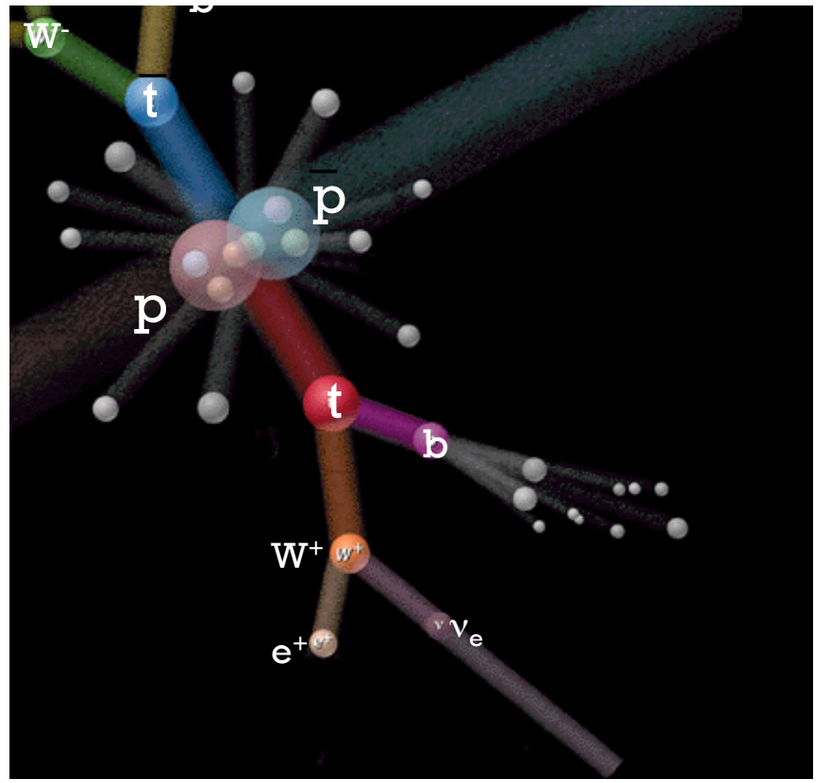
In other words, ***all the physicists pushing into the realm of the Silicon Valley engineer is a sign of a much bigger change to come. Soon, all the Silicon Valley engineers will push into the realm of the physicist.***

DL in HEP

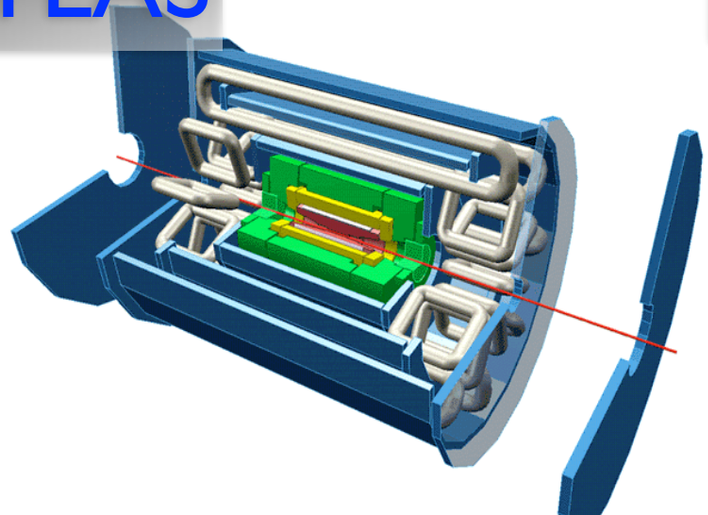
The Big Picture…

HEP Experiments

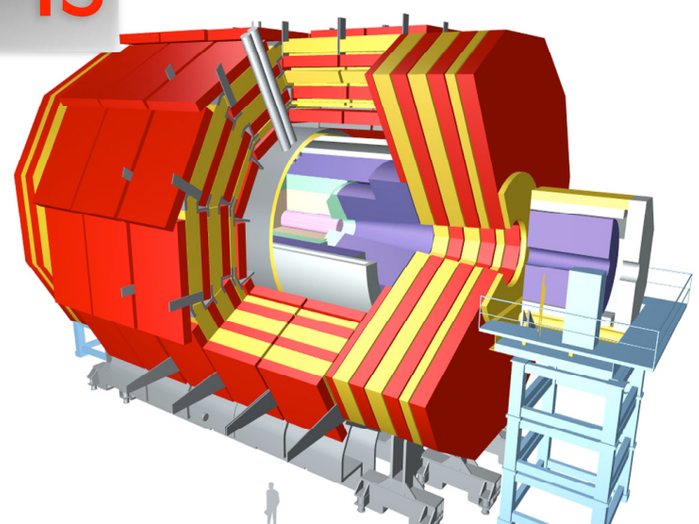
- 5 technical components to HEP experiment:
  - ***Accelerator***: e.g. LHC collisions creating quickly decaying heavy particles. Extremely high rate: 40 * O(50) Million collisions/sec.
  - ***Detector***: a big camera. ~ e.g. LHC 1.5 MB/event (60 TB/s)
    - Pictures of long-lived decay products of short lived heavy/ interesting particles.
    - Sub-detectors parts: Tracking, Calorimeters, Muon system, Particle ID (e.g. Cherenkov, Time of Flight)
  - ***DAQ/Trigger:*** Hardware/software
  - ***Software:*** Reconstruction (Raw data -> particle “features”) / Analysis
  - ***Computing:*** GRID Monarch Model “Cloud” Computing/Data Management (software/hardware)

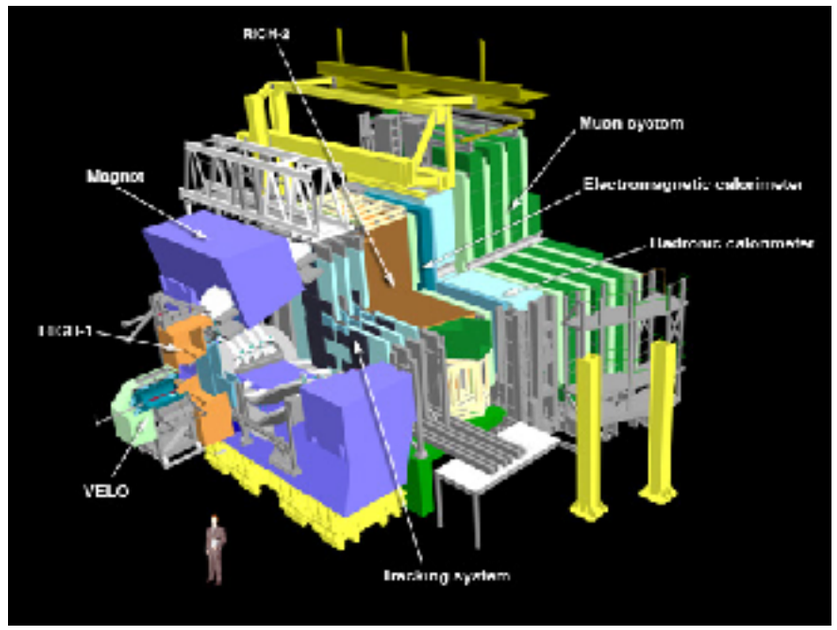


ATLAS



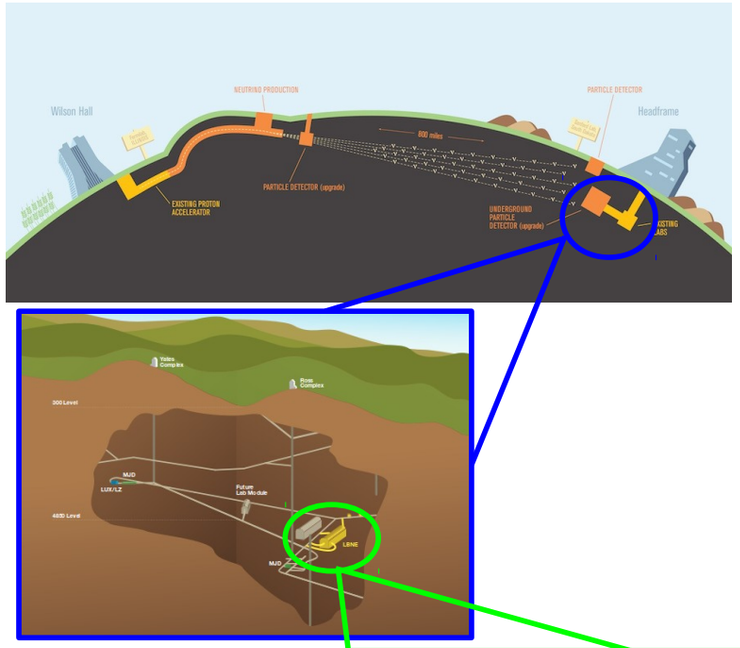
CMS

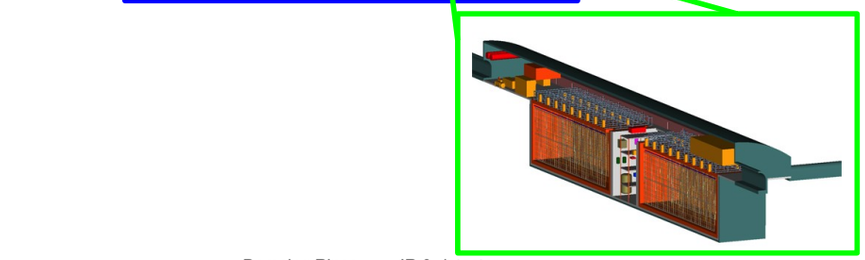


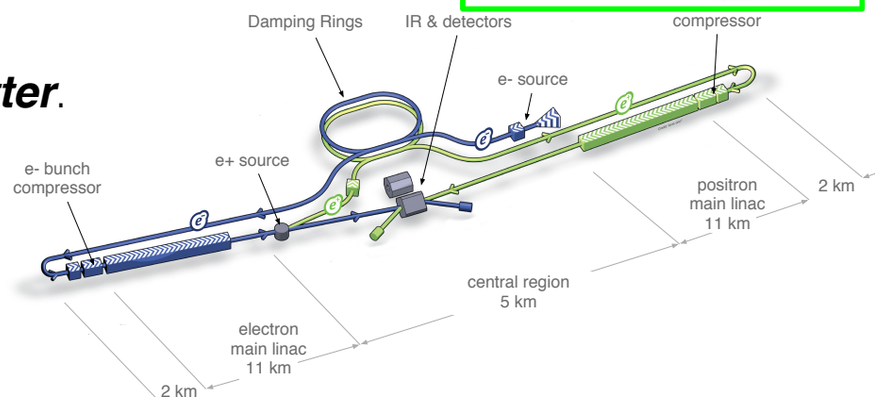


Frontiers

- ***Energy Frontier***: ***Large Hadron Collider (LHC)*** at 13 TeV now, ***High Luminosity (HL)- LHC*** by 2025, perhaps 33 TeV LHC or 100 TeV Chinese machine in a couple of decades.
    - Having found Higgs, moving to studying the SM ***Higgs*** find new Higgses
    - Test ***naturalness***
    - Find ***Dark Matter*** (reasons to think related to naturalness)
- ***Intensity Frontier***:
    - ***B Factories***: upcoming SuperKEKB/SuperBelle
    - ***Neutrino Beam Experiments***:
        - Series of current and upcoming experiments: Nova, MicroBooNE, SBND, ICURUS
        - ***US's flagship experiment*** in next decade: ***Long Baseline Neutrino Facility (LBNF)/Deep Underground Neutrino Experiment (DUNE) at Intensity Frontier***
    - Measure properties of ***b-quarks*** and ***neutrinos*** (newly discovered mass)… search for ***matter/anti-matter asymmetry***.
    - Auxiliary Physics: Study ***Supernova***. Search for ***Proton Decay*** and ***Dark Matter***.
- ***Precision Frontier***: ***International Linear Collider (ILC)***, hopefully in next decade. Most energetic e^+e^- machine.
    - ***Precision studies*** of ***Higgs*** and hopefully ***new particles*** found at LHC.







Why go Deep?

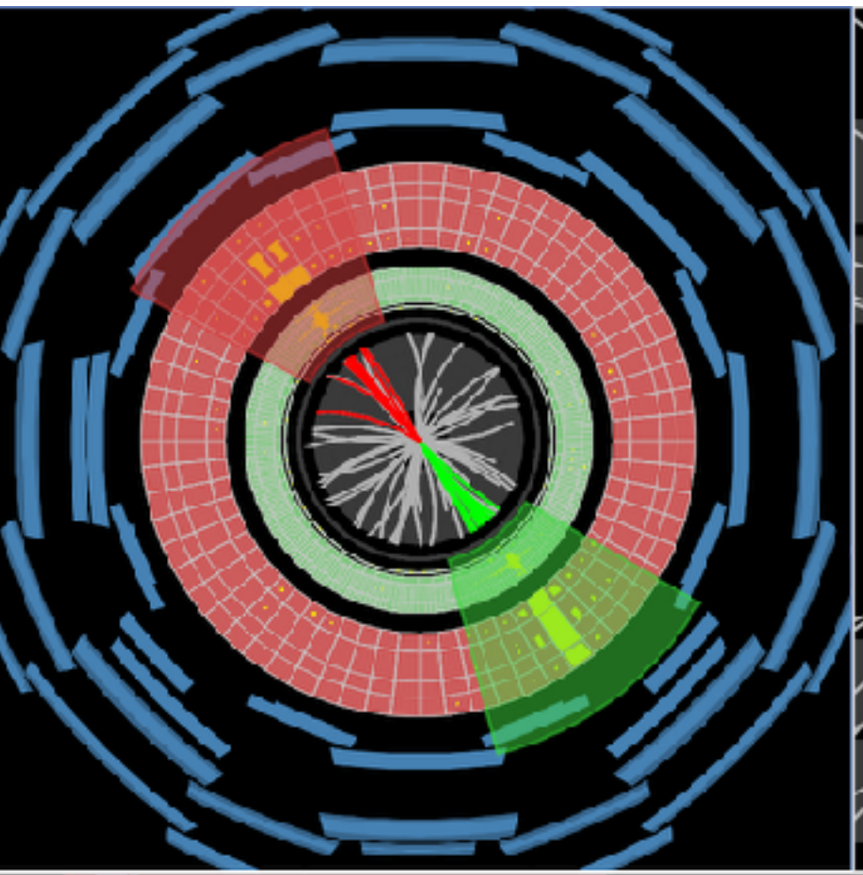
- ***Better Algorithms***
  - DNN-based classification/regression generally **out perform** hand crafted algorithms.
  - In some cases, it may provide a ***solution*** where ***algorithm approach doesn't exist or fails***.
  - ***Unsupervised learning***: make sense of complicated data that we don't understand or expect.
- ***Easier Algorithm Development***: ***Feature Learning*** instead of *Feature Engineering*
  - Reduce time physicists spend writing developing algorithms, ***saving time and cost***. (e.g. ATLAS > $250M spent software)
  - Quickly perform performance ***optimization*** or ***systematic studies***.
- ***Faster Algorithms***
  - After training, DNN inference is often *faster* than sophisticated algorithmic approach.
  - DNN can ***encapsulate expensive computations***, e.g. Matrix Element Method.
  - ***Generative Models*** enable fast simulations.
  - ***Already parallelized*** and optimized for GPUs/HPCs.
  - ***Neuromorphic*** processors.

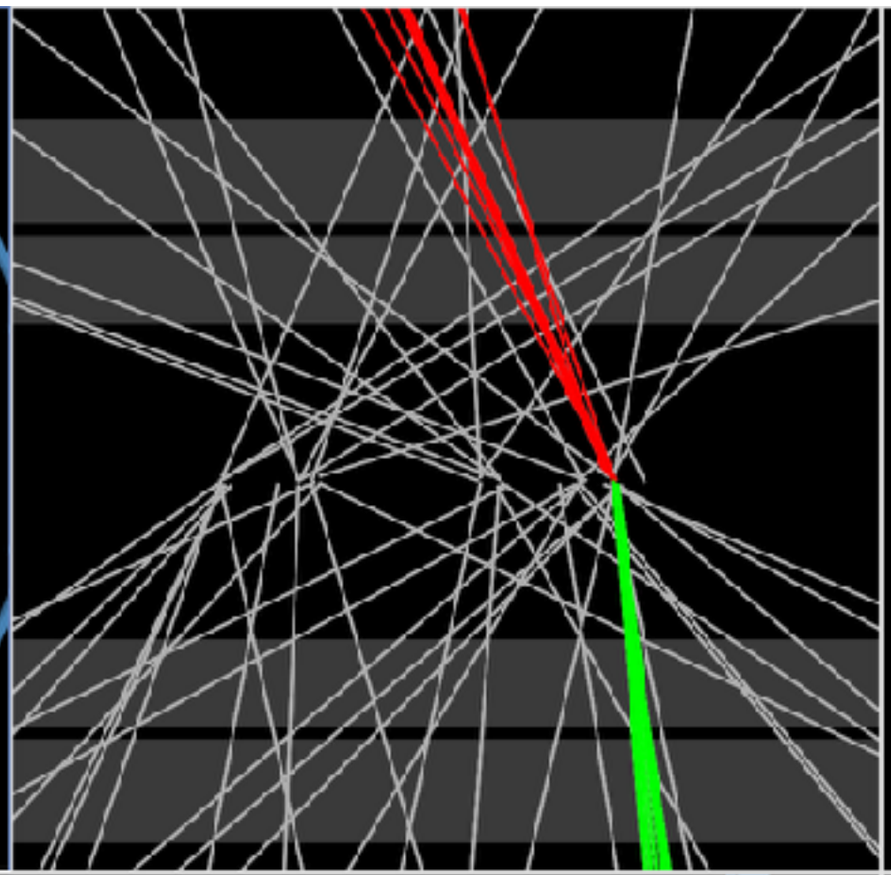
Where is ML needed?

- Traditionally ML Techniques in HEP
    - Applied to Particle/Object Identification
    - Signal/Background separation
    - Here, ML maximizes reach of existing data/detector… equivalent to additional integral luminosity.
    - There is lots of interesting work here… and potential for big impact.

- Now we hope ML can help address looming computing problems of the next decade:
    - ***Reconstruction***
        1. Intensity Frontier- ***LArTPC*** Automatic Algorithmic ***Reconstruction*** still struggling
        2. Energy Frontier- ***HL-LHC Tracking***- Pattern Recognition blows up due to combinatorics
    - ***Simulation***
        3. LHC Calorimetry- Large Fraction of ATLAS CPU goes into ***shower simulation***.

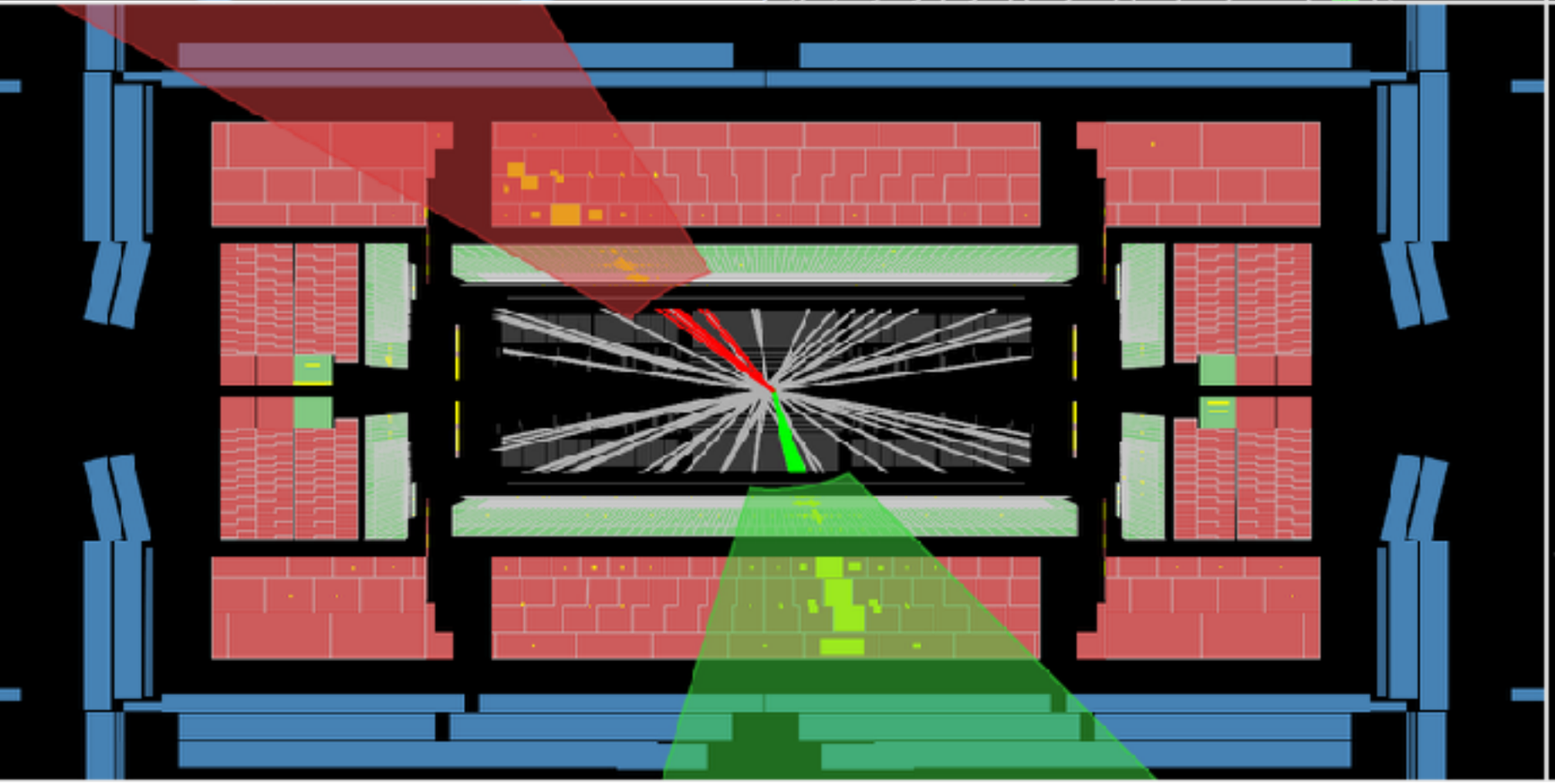
A bit about detectors…

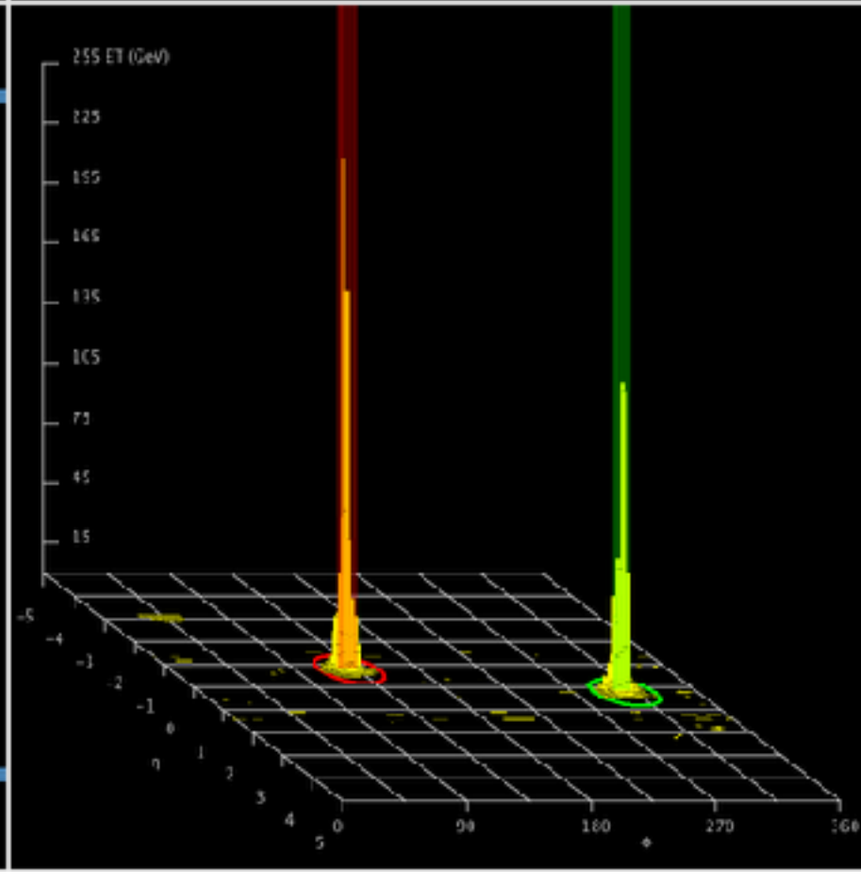




Run Number: 271298, Event Number: 403602858

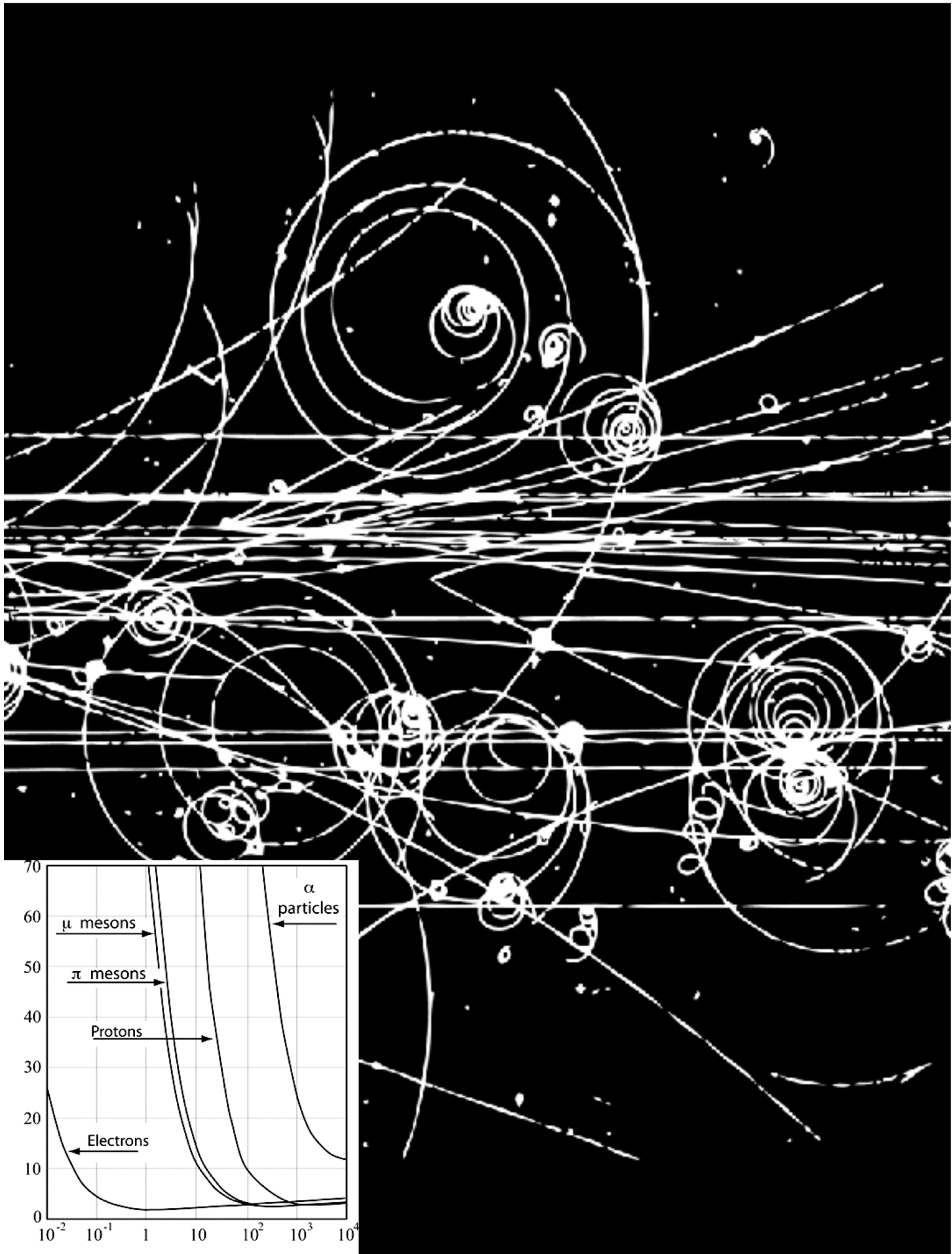
Date: 2015-07-11 02:09:14 CEST





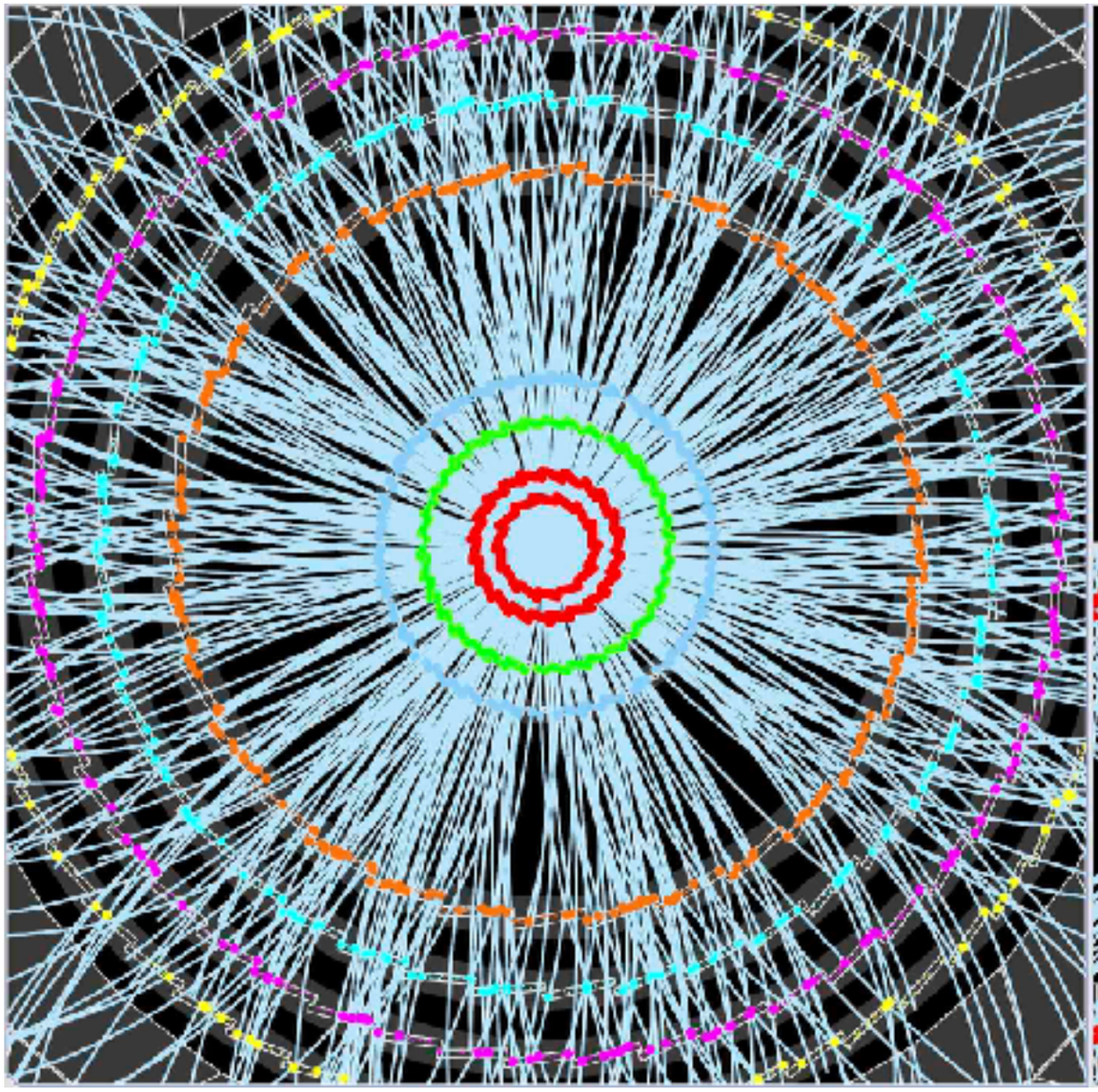
How do we “see” particles?

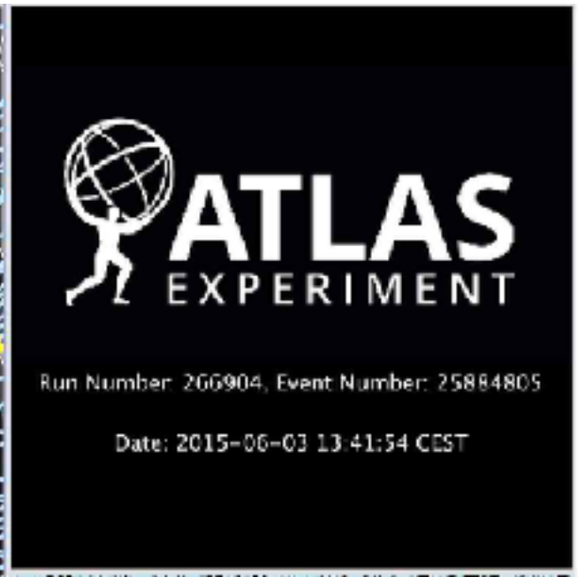
- ***Charged particles ionize media***
  - Image the ions.
    - In ***Magnetic Field*** the ***curvature*** of trajectory ***measures momentum***.
    - Momentum resolution degrades as less curvature: $\sigma(p) \sim c\ p \oplus d$.
      - d due to multiple scattering.
- Measure ***Energy Loss*** (~ # ions)
  - dE/dx = Energy Loss / Unit Length = f(m, v) = Bethe-Block Function
    - Identify the particle type
  - ***Stochastic process*** (Laudau)
- Loose all energy → range out.
  - Range characteristic of particle type.

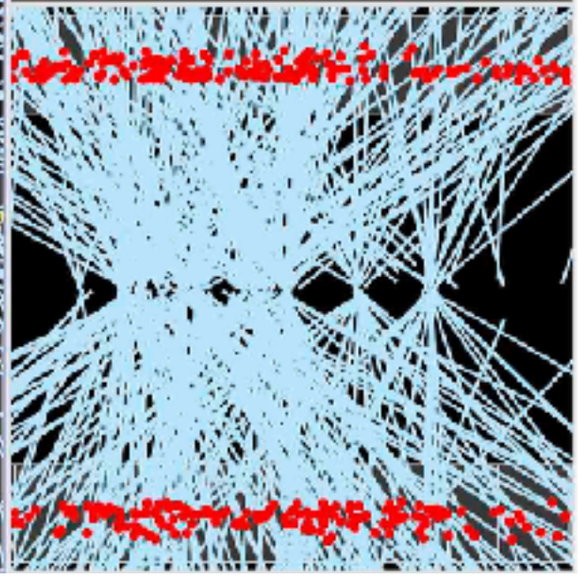


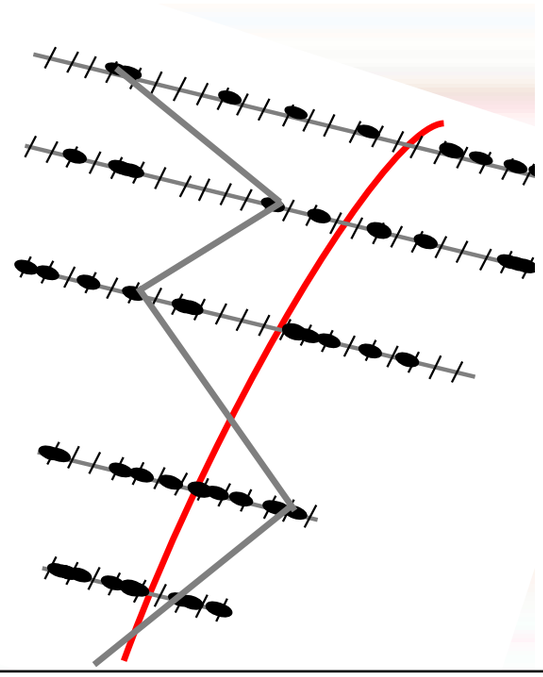
Tracking

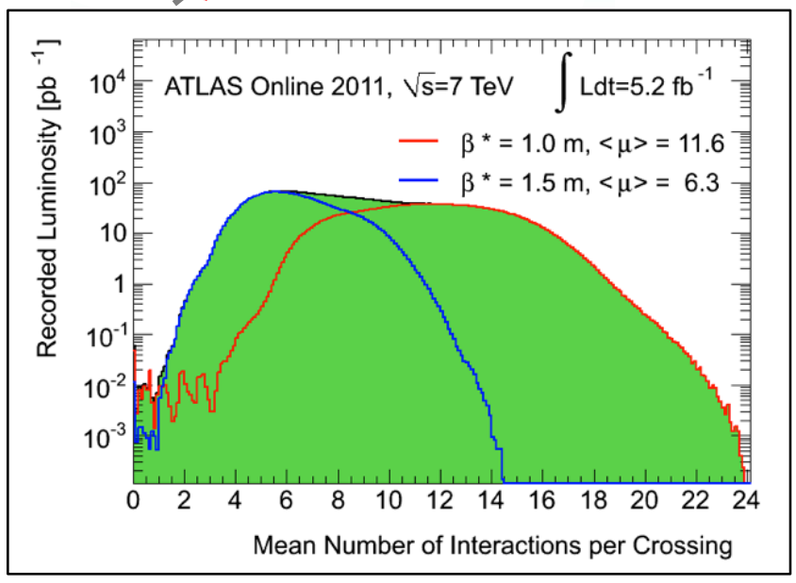
- Measure Charged particle trajectories. If B-field, then measure momentum.





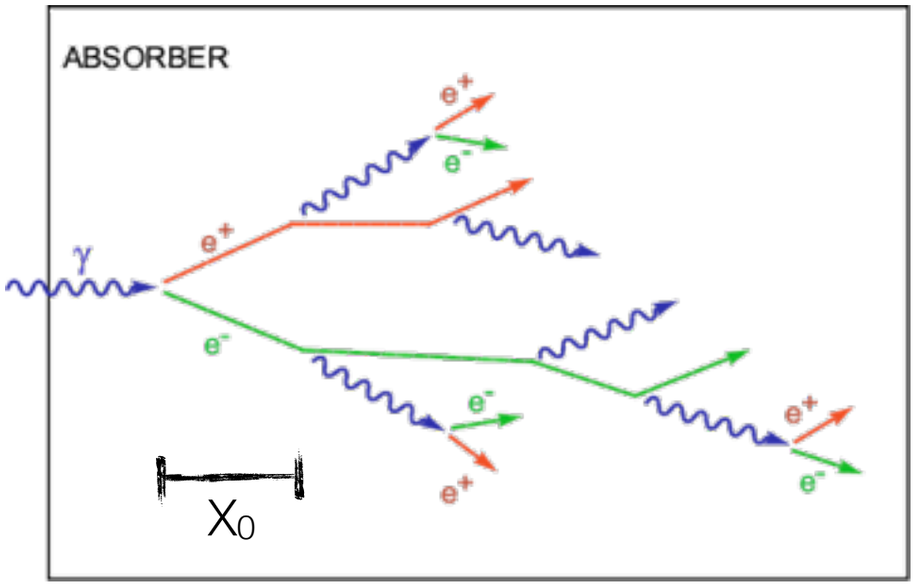




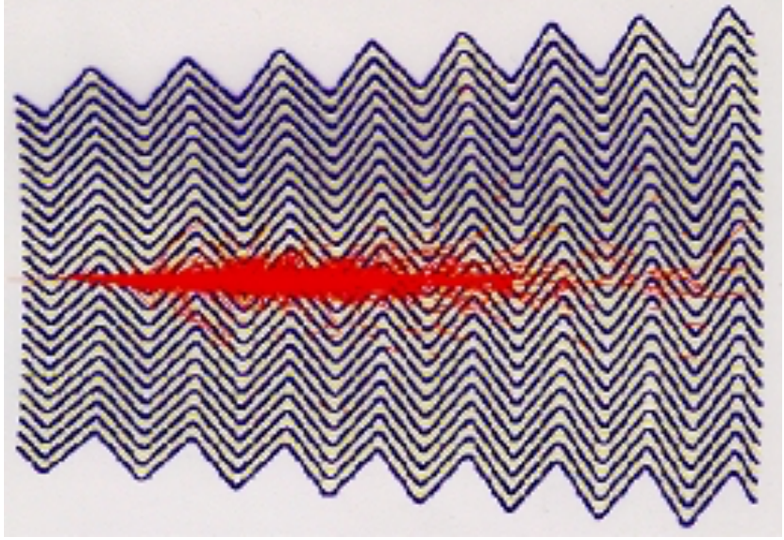


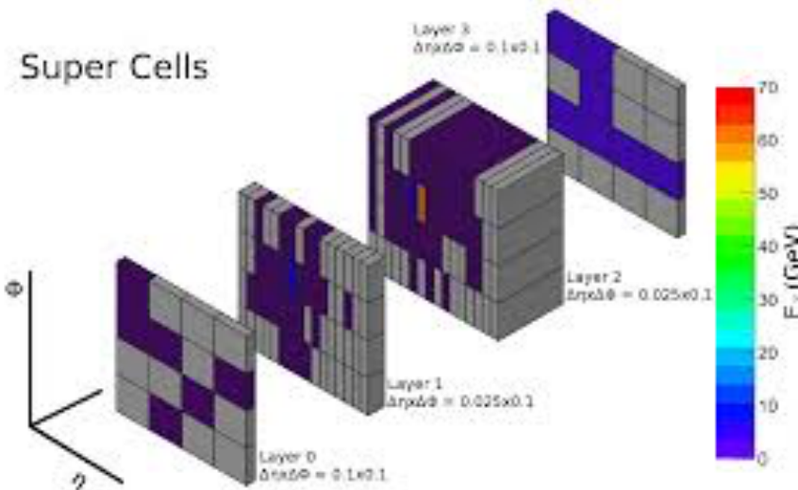
How do we “see” particles?

- Particles deposit their energy in a ***stochastic process*** know as ***“showering”***, secondary particles, that in turn also shower.
  - Number of secondary particles ~ Energy of initial particle.
  - Energy resolution improves with energy: $\sigma(E) / E = a/\sqrt{E} \oplus b/E \oplus c$.
    - a = sampling, b = noise, c = leakage.
  - Density and Shape of shower characteristic of type of particle.



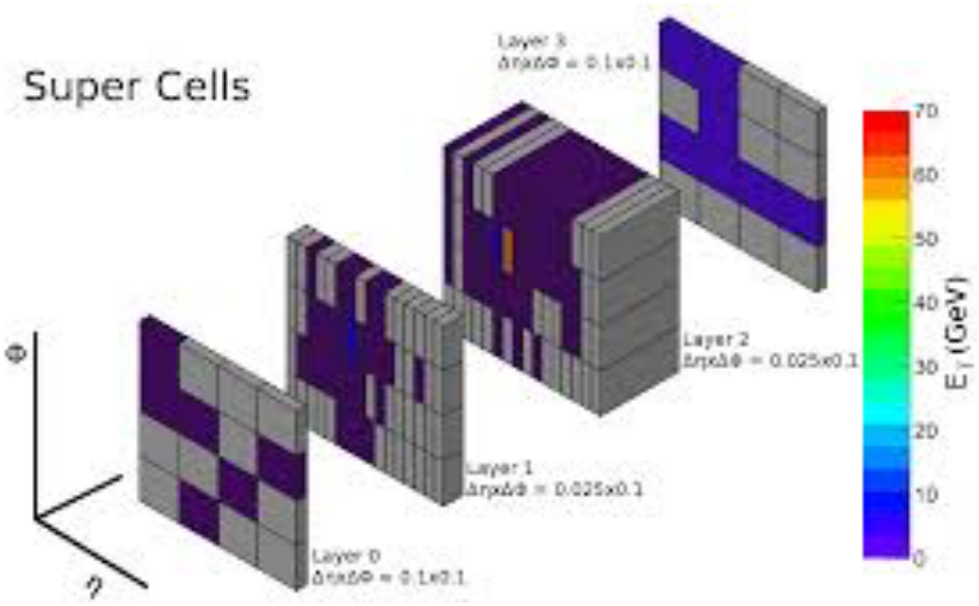
- ***Electromagnetic calorimeter***: Low Z medium
  - ***Light particles***: electrons, photons, $\pi^0 \rightarrow \gamma\gamma$ interact with electrons in medium
- ***Hadronic calorimeters***: High Z medium
  - ***Heavy particles***: Hadrons (particles with quarks, e.g. charged pions/protons, neutrons, or jets of such particles)
    - Punch through low Z.
    - Produce secondaries through strong interactions with the nucleus in medium.
    - Unlike EM interactions, not all energy is observed.

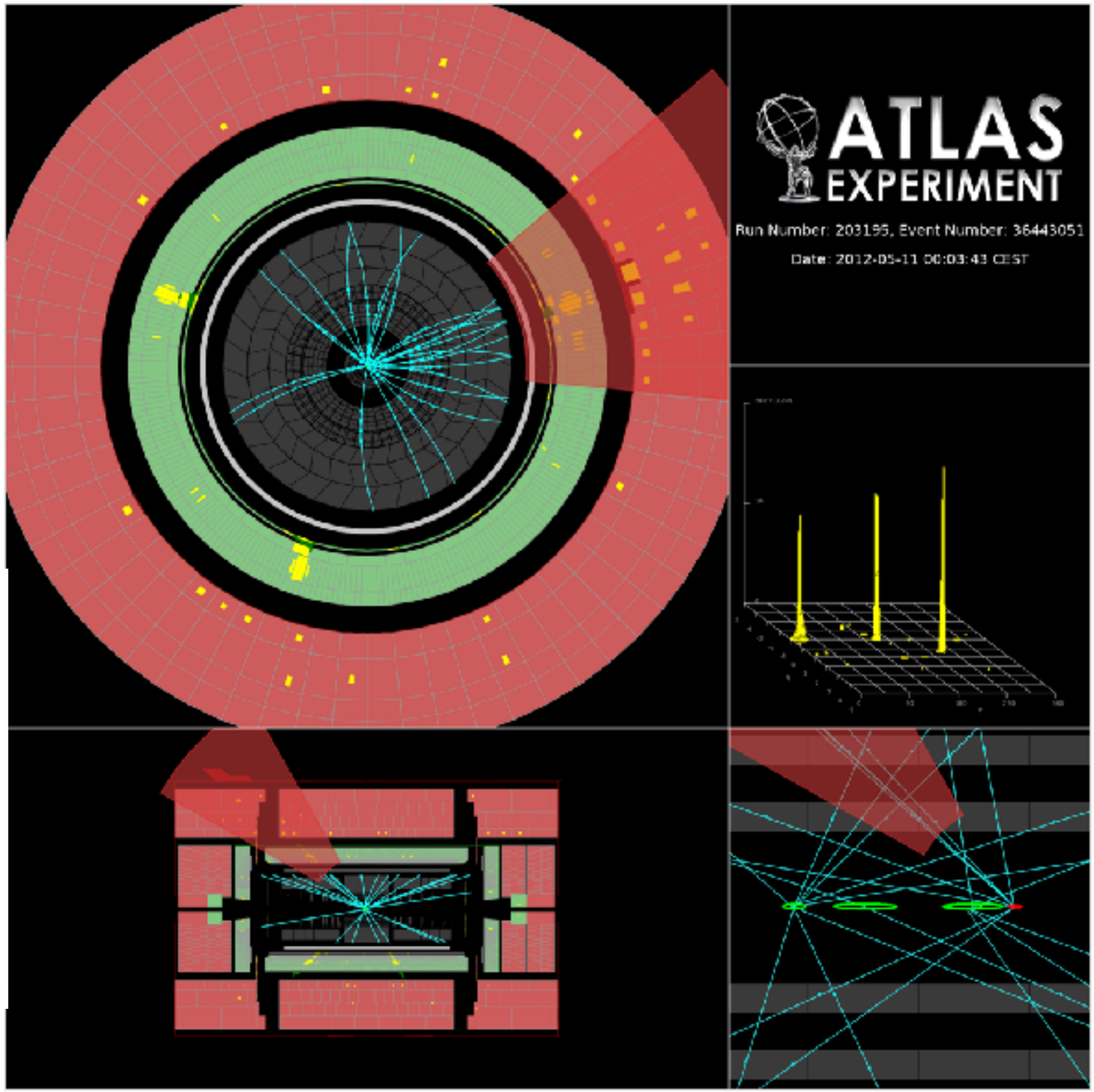




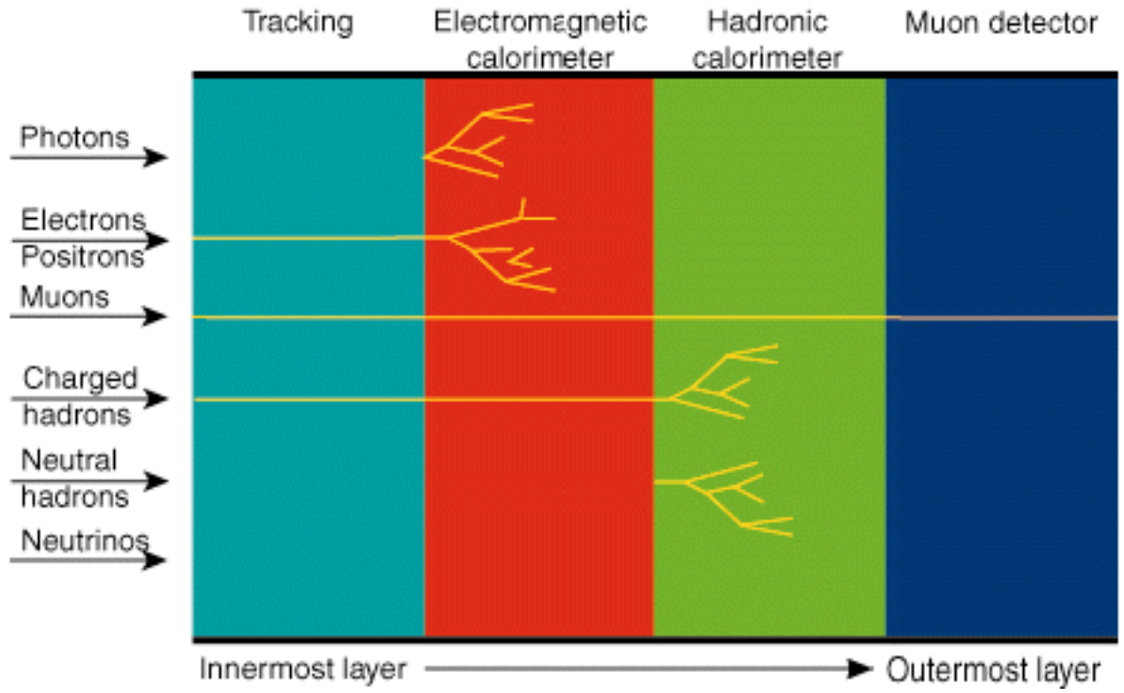
Calorimetry

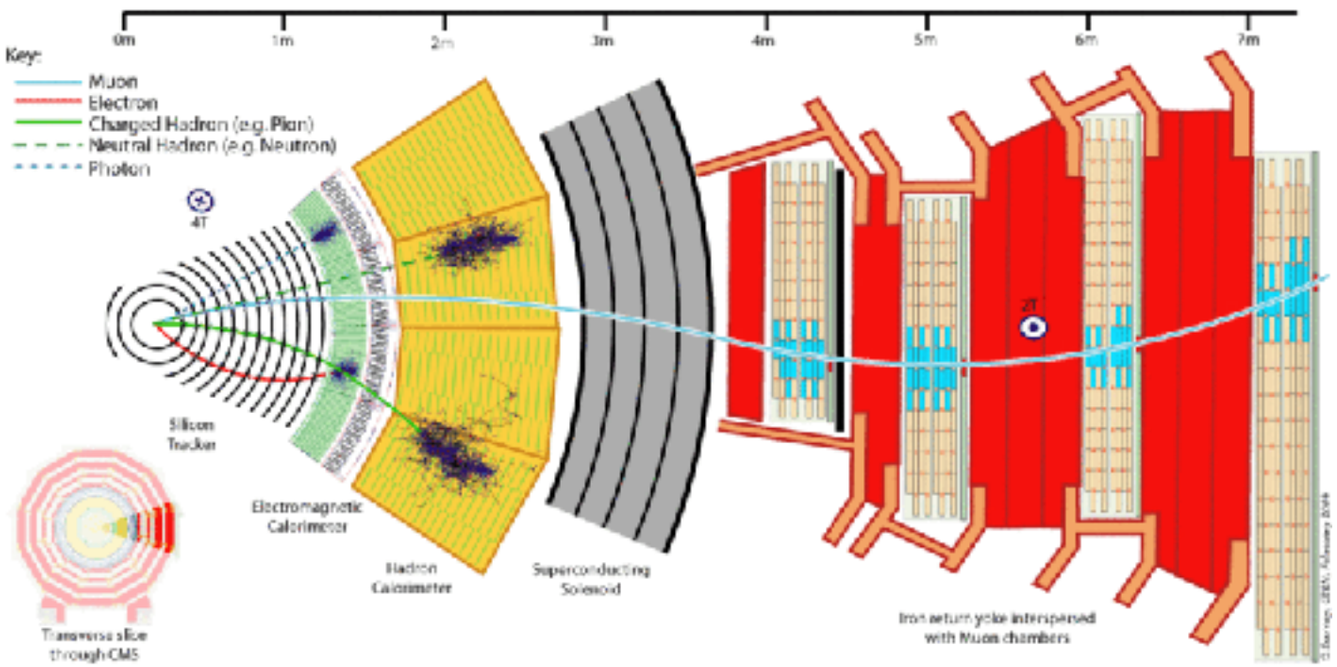
- Make particle interact and loose all energy, which we measure. 2 types:
  - Electromagnetic: e.g. crystals in CMS, Liquid Argon in ATLAS.
  - Hadronic: e.g. steel + scintillators
  - e.g ATLAS:
    - 200K Calorimeter cells measure energy deposits.
    - 64 x 36 x 7 3D Image

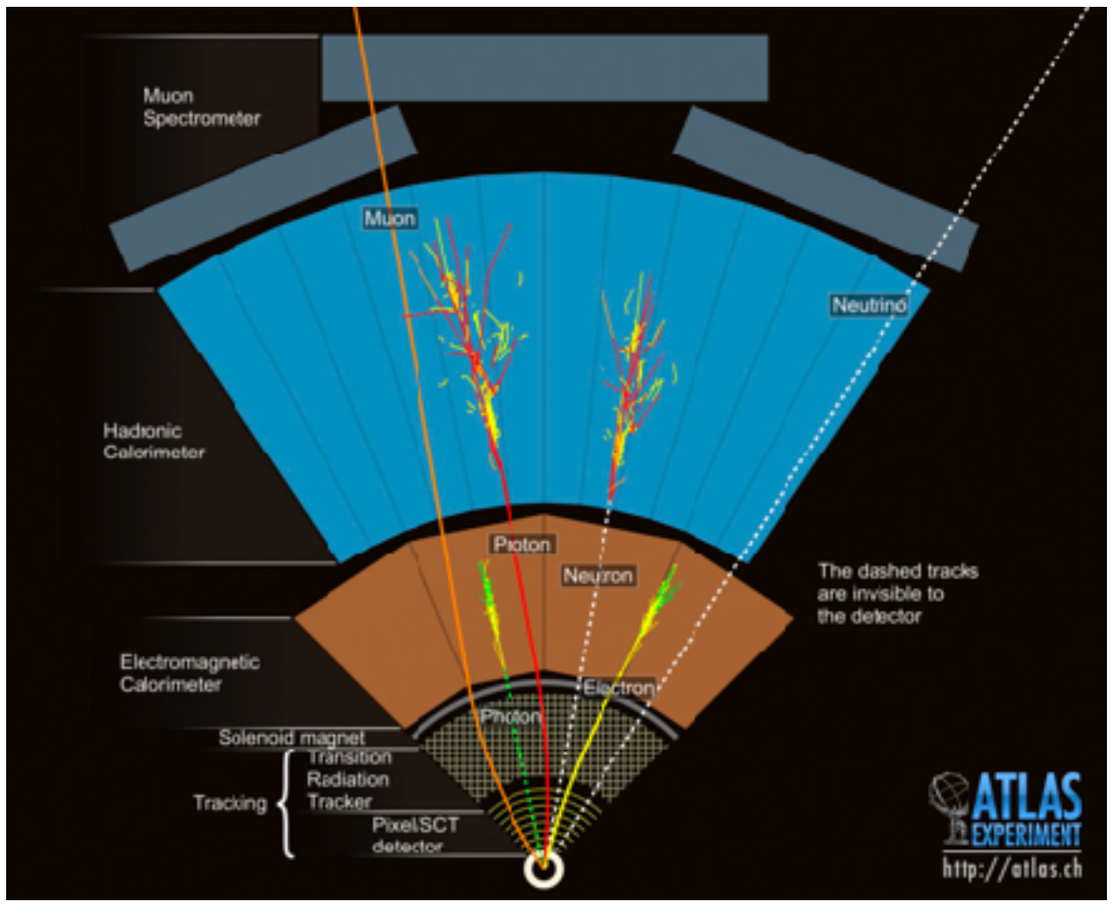




LHC/ILC detectors

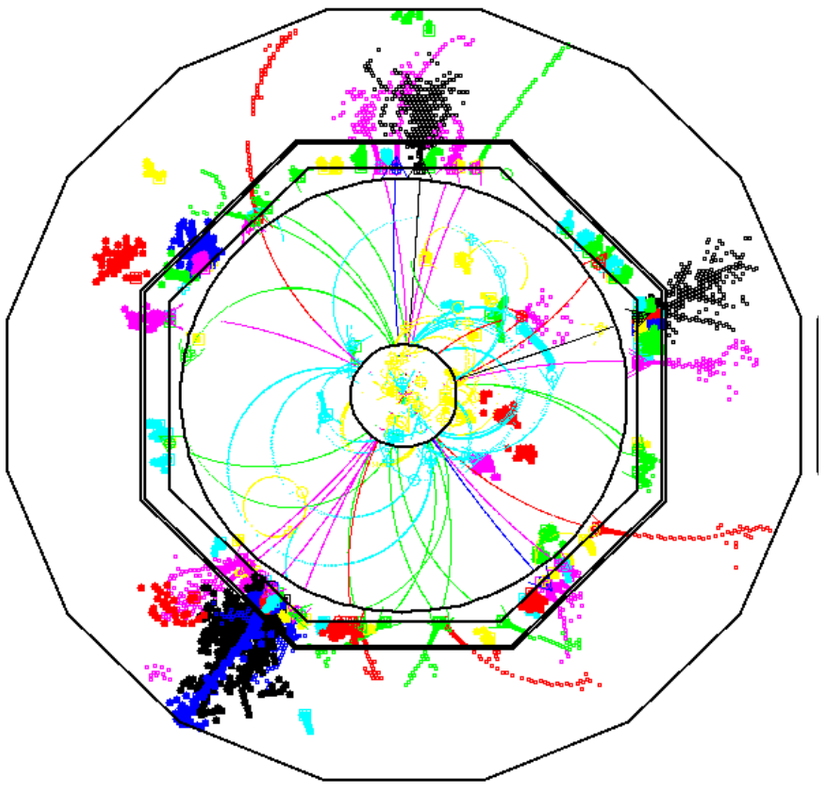


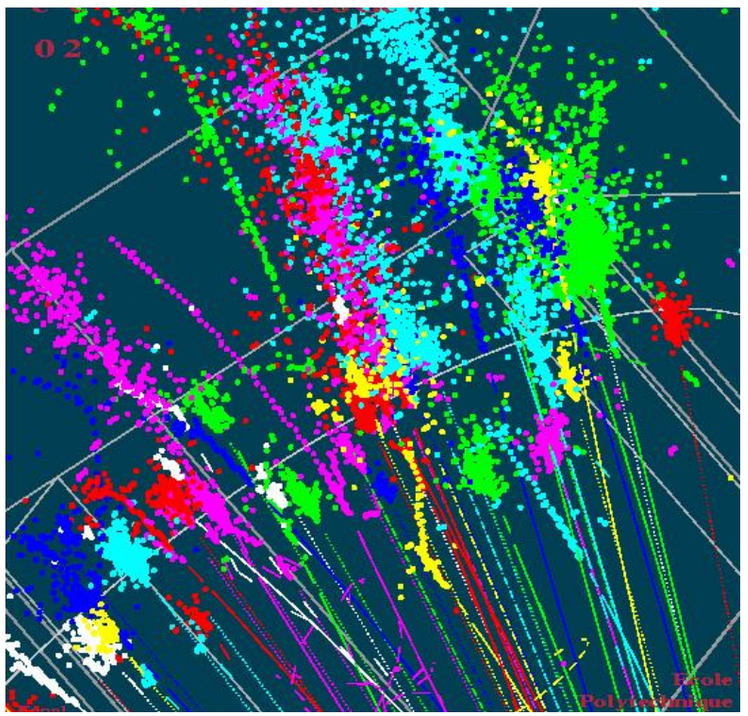


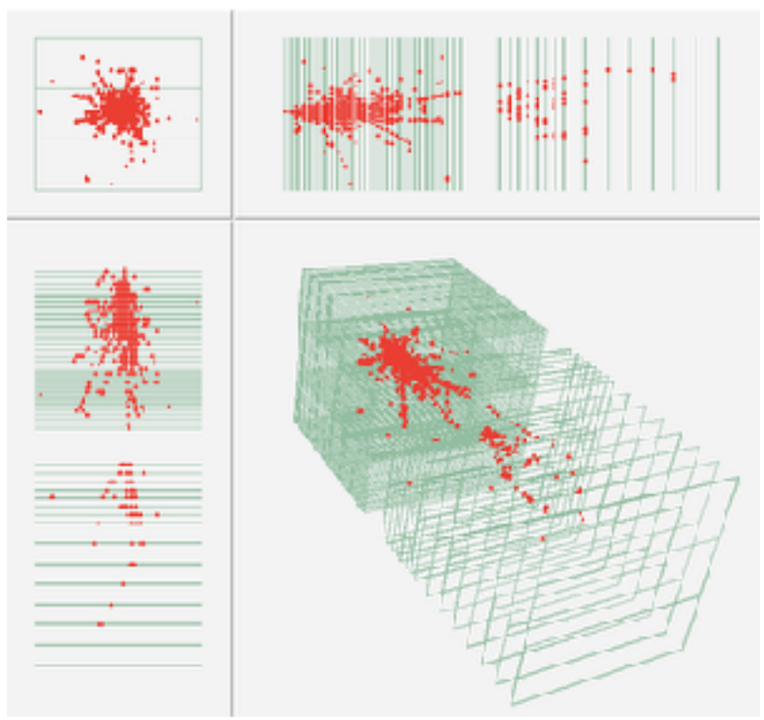


ILC Detectors

- Precision measurements require excellent calorimetry
  - Aim for jet energy resolution giving di-jet mass resolution similar to Gauge boson widths
  - Various concepts ~ digital/high granularity calorimetry + particle flow.
- Similarities to upgrade LHC forward detectors







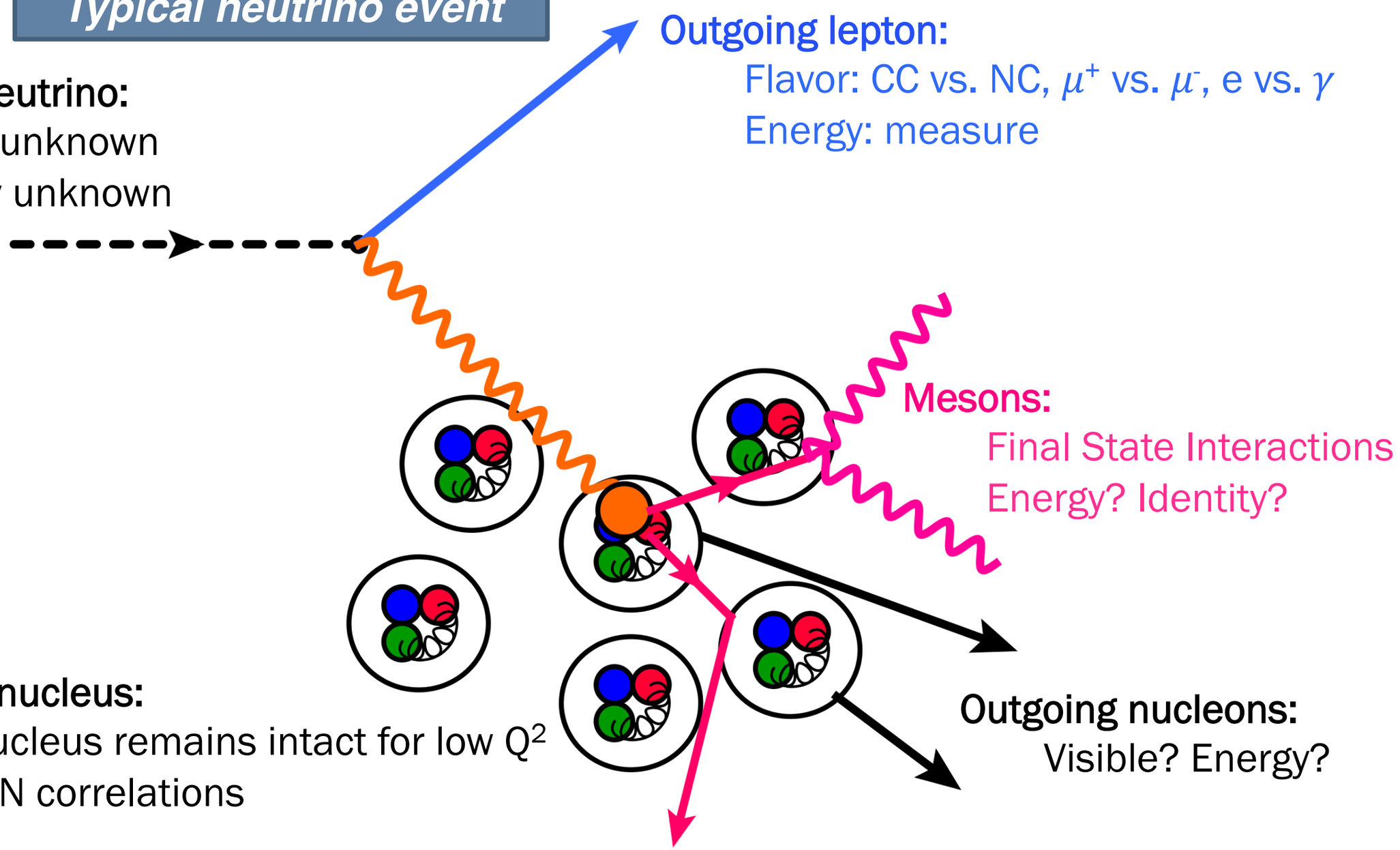
Neutrino Detection

In neutrino experiments, try to determine flavor and estimate energy of incoming neutrino by looking at outgoing products of the interaction.

Typical neutrino event

Incoming neutrino:
Flavor unknown
Energy unknown

Outgoing lepton:
Flavor: CC vs. NC, μ^+ vs. μ^-, e vs. γ
Energy: measure



Mesons:
Final State Interactions
Energy? Identity?

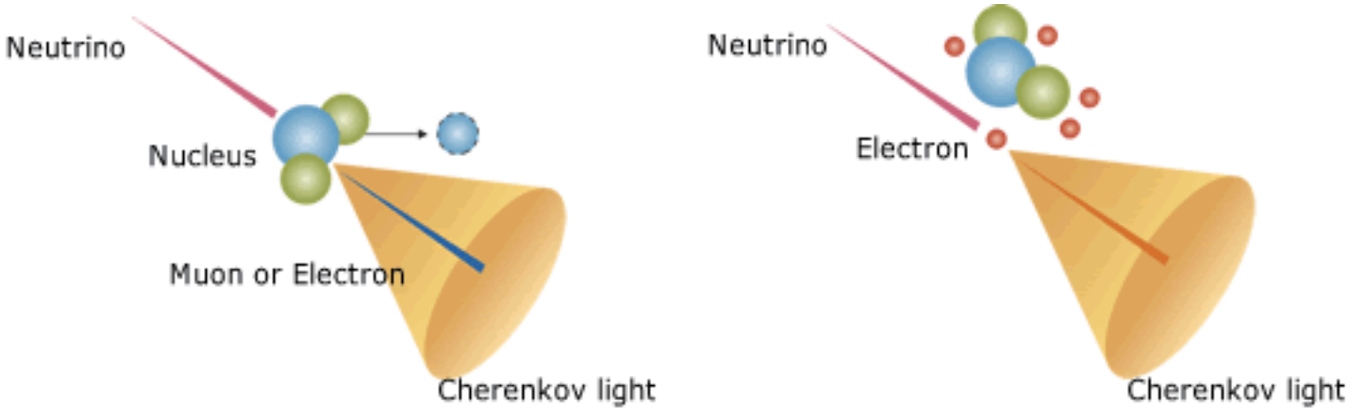
Target nucleus:
Nucleus remains intact for low Q^2
N-N correlations

Outgoing nucleons:
Visible? Energy?

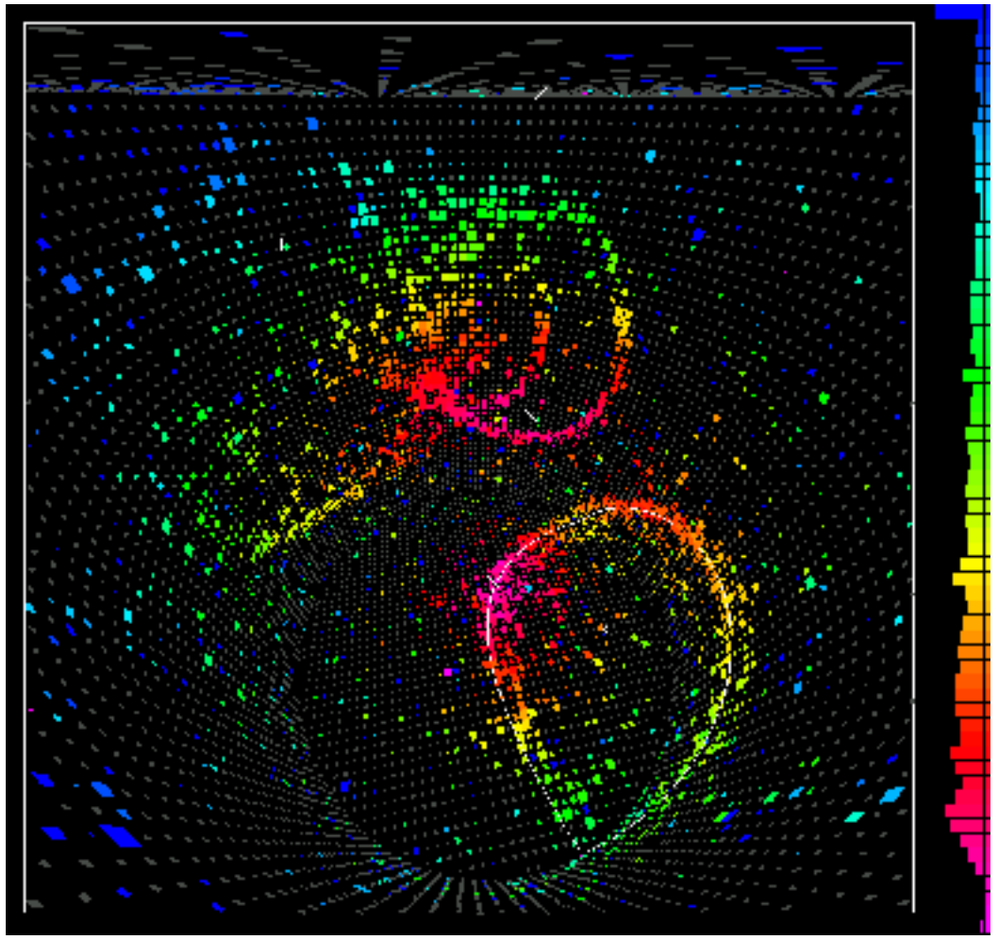
Jen Raaf

How do we “see” particles?

- Charged Particles traveling faster than speed of light in medium emit ***Cherenkov light*** (analogous to sonic boom).
  - Light emitted in cone, with angle function of speed and mass.
  - Depending on context, allow for particle identification and/or speed measurement.

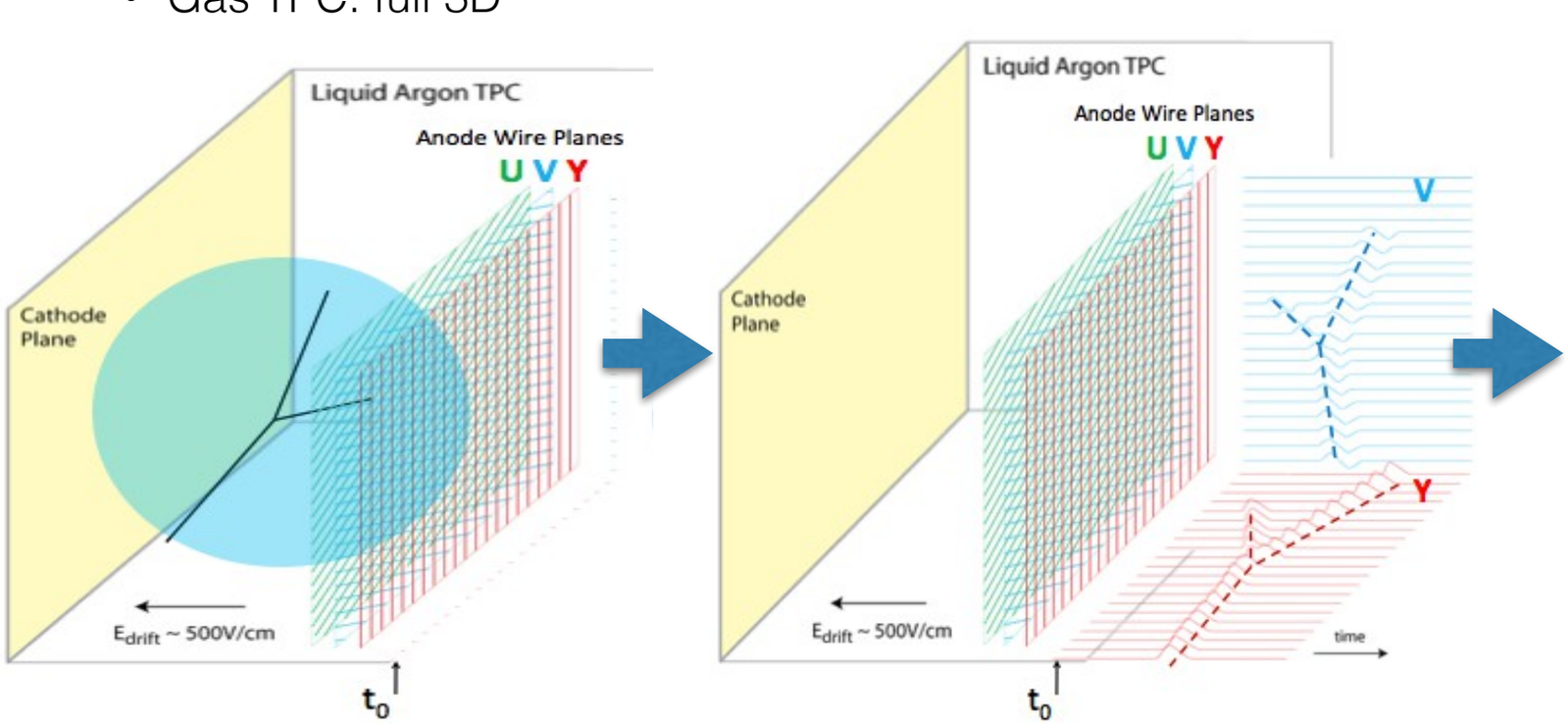


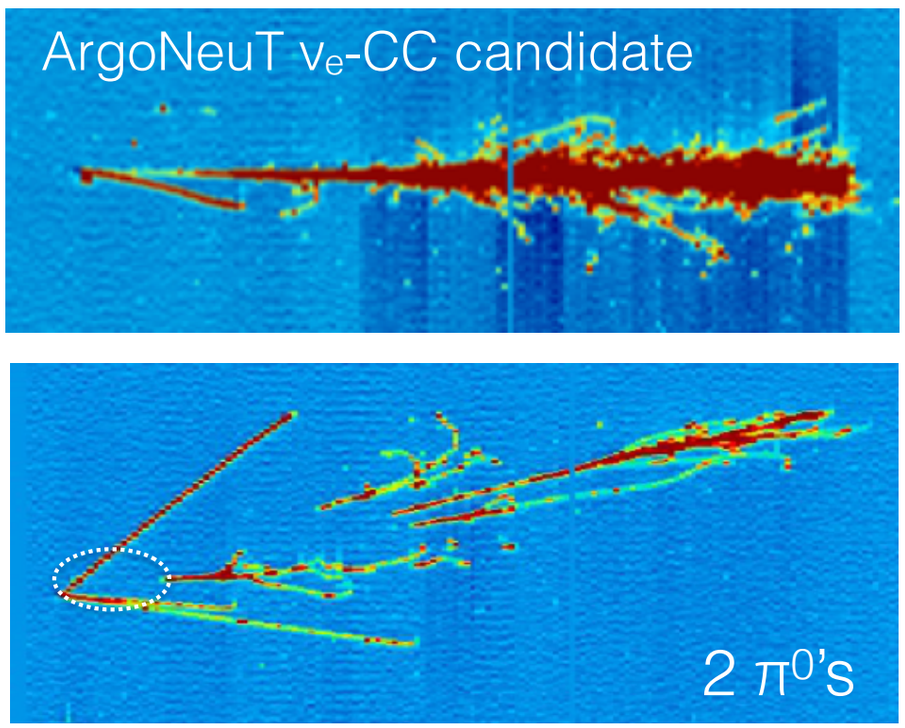
The generated charged particle emits the Cherenkov light.



Neutrino Detectors

- ***Need large mass/volume*** to maximize chance of neutrino interaction.
- Technologies:
  - Water/Oil Cherenkov
  - Segmented Scintillators
  - ***Liquid Argon Time Projection Chamber: promises ~ 2x detection efficiency.***
    - ***Provides tracking, calorimetry, and ID all in same detector***.
    - Chosen technology for US's flagship LBNF/DUNE program.
      - Usually 2D read-out… 3D inferred.
    - Gas TPC: full 3D

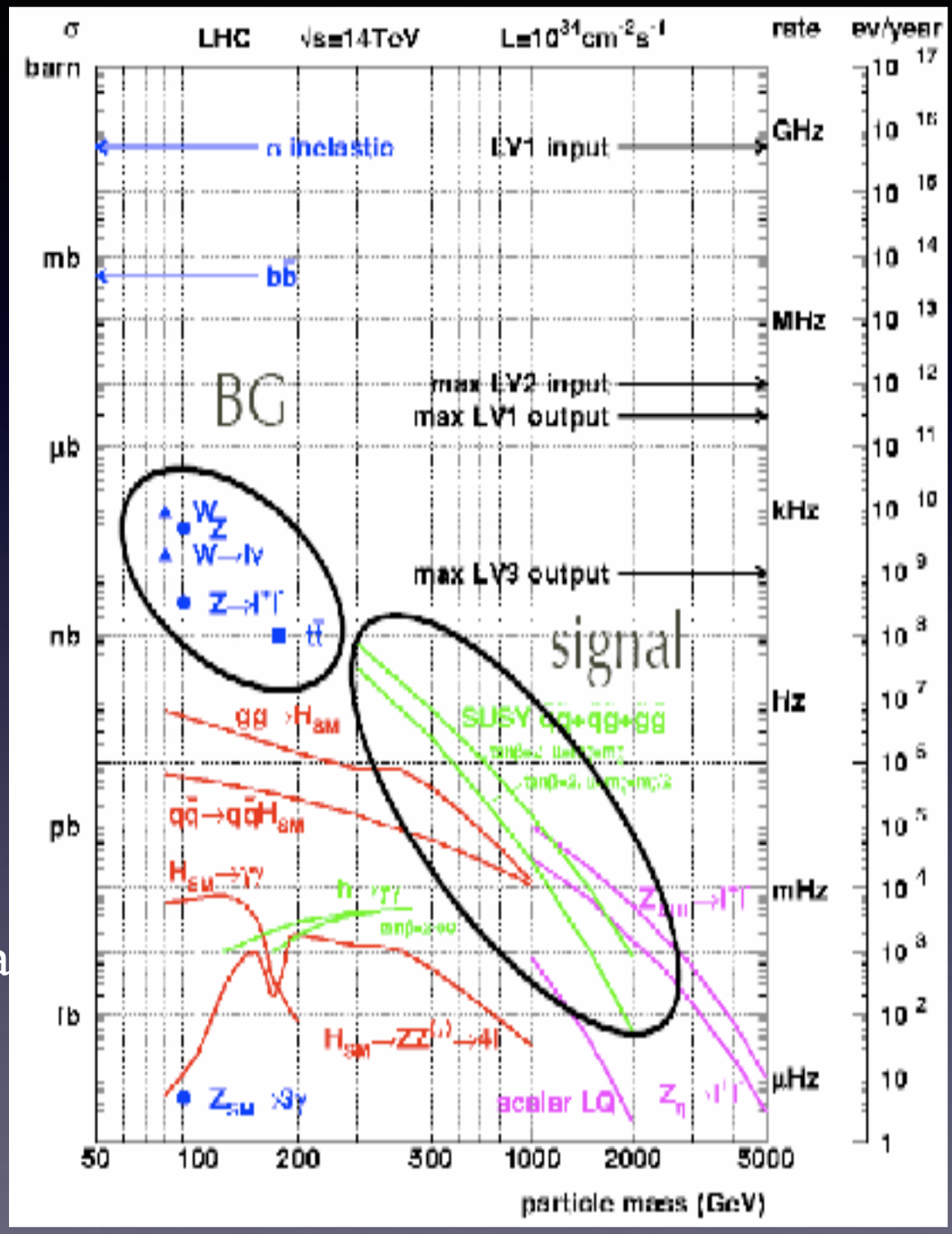




From Data to Physics

Trigger

- Back of the envelope:
  - 100M Electronic Channels
  - 40 million collisions / sec at 1.5 MB/Event = 60 TB/ sec.
  - Requires 2.5 m diameter bundle of Fibers to read off detector. (90's Tech, so 1 Gb/s)
- Fortunately interesting physics happens ~ 1 in 10^{11}
- Trigger system (input 40 MHz):
  - look for unique features of "interesting" events
  - analogue hardware determines if we should read data off of detector (@ 100 KHz)
  - Computing farm further reduces to 1 KHz (Run 2)
- ATLAS/CMS collect 10 PB/month, each. (?)
- High Luminosity LHC will have much busier events



HEP Computing

Full Simulation

KHz

Generation

mHz

Simulation

Hz

Digitization

Fast Simulation

KHz

Generation

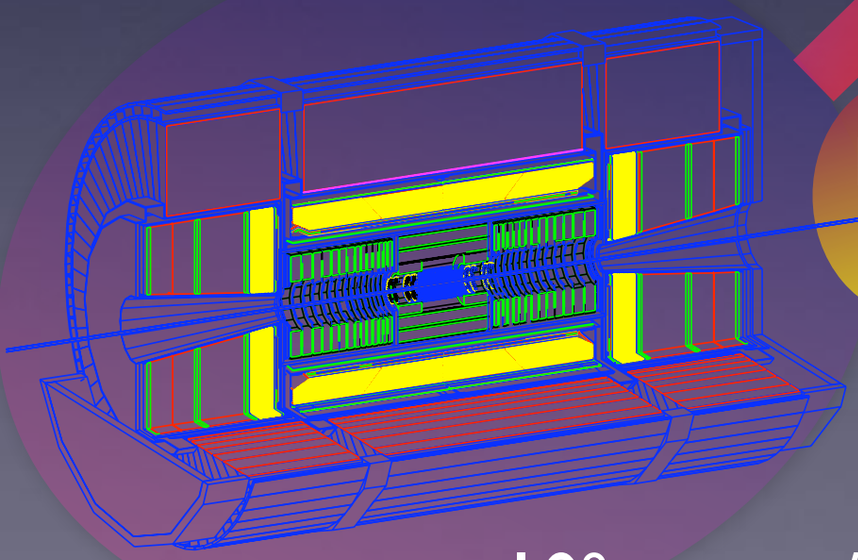
Hz

Fast Simulation

Hz

Reconstruction

High-level Trigger



1000 Hz

10^9 events/year

KHz

Hz

Derivation

Statistical Analysis

Data Analysis & Calibration

The Computing Model

- Resources Spread Around the GRID

•Reprocessing of full data with improved calibrations 2 months after data taking.
•Managed Tape Access: RAW, ESD
•Disk Access: AOD, fraction of ESD

•Derive 1st pass calibrations within 24 hours.
•Reconstruct rest of the data keeping up with data taking.

•Interactive Analysis
•Plots, Fits, Toy MC, Studies, ...

Tier 3

DPD

Tier 2

AOD

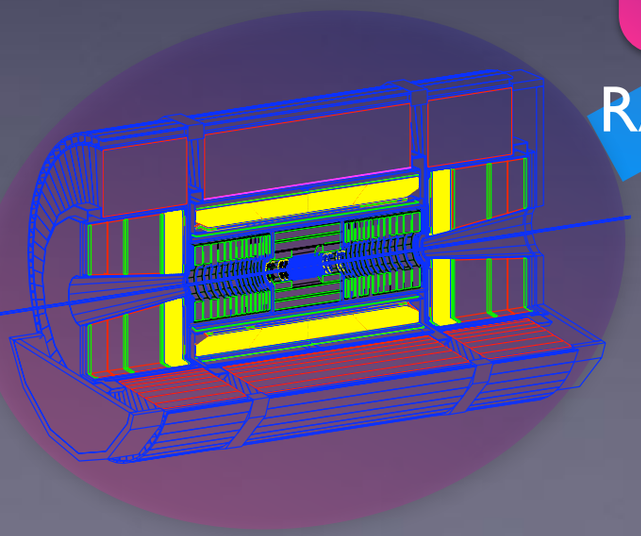
Tier 1

RAW/
AOD/
ESD

Tier 0

RAW

O(300) Sites Worldwide



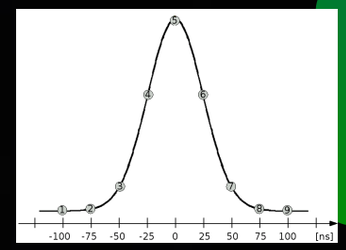
CERN Analysis Facility

•Production of simulated events.
•User Analysis: 12 CPU/ Analyzer
•Disk Store: AOD

•Primary purpose: calibrations
•Small subset of collaboration will have access to full ESD.
•Limited Access to RAW Data.

Reconstruction

- Starts with ***raw inputs*** (e.g. Voltages)
- Low level ***Feature Extraction***: e,g, Energy/Time in each Calo Cell
- ***Pattern Recognition***: Cluster adjacent cells. Find hit pattern.
- ***Fitting***: Fit tracks to hits.
- ***Combined reco***: e.g.:
  - Matching Track+EM Cluster = Electron.
  - Matching Track in inter detector + muon system = Muon
- ***Output particle candidates*** and measurements of their properties (e.g. energy)



Data Analysis

- Objectives:
  - ***Searches*** (hypothesis testing): Likelihood Ratio Test (Neyman-Pearson lemma)
  - ***Measurements***: Maximum Likelihood Estimate

$$\frac{P(x|H_1)}{P(x|H_0)} > k_\alpha$$

  - ***Limits*** (confidence intervals): Also based on Likelihood

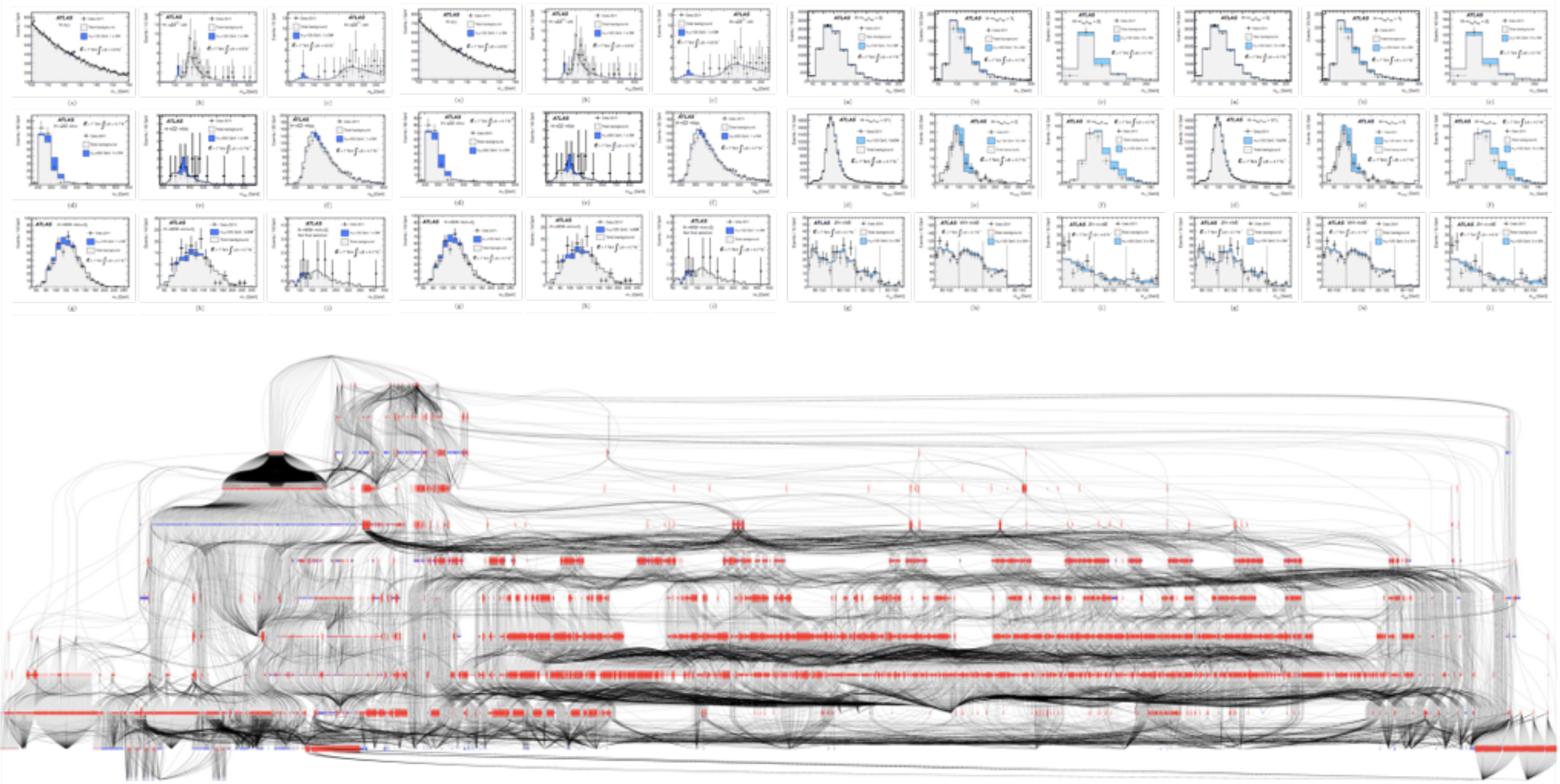
- ***Likelihood***

$$p(\{x\}|\theta) = \mathrm{Pois}(n|\nu(\theta)) \prod_{e=1}^{n} p(x_e|\theta)$$

  - *n* Independent Events (*e*) with Identically Distributed Observables ({*x*})
  - Significant part of Data Analysis is ***approximating the likelihood*** as best as we can.

Slide from Kyle Cranmer:

LIKELIHOOD-BASED COMBINATIONS



$$\mathbf{f}_{\text{tot}}(\mathcal{D}_{\text{sim}}, \mathcal{G}|\boldsymbol{\alpha}) = \prod_{c \in \text{channels}} \left[\text{Pois}(n_c|\nu_c(\boldsymbol{\alpha})) \prod_{e=1}^{n_c} f_c(x_{ce}|\boldsymbol{\alpha}) \right] \cdot \prod_{p \in \mathbb{S}} f_p(a_p|\alpha_p)$$

Approximating the Likelihood

- Physics is all about establishing a very precise "model" of the underlying phenomena… so ***we can model our data very well.***

- Enables ***multi-step ab-initio simulations***:

  1. ***Generation***: Standard Model and New Physics are expressed in language of Quantum Field Theory.

     ➡ Feynman Diagrams simplify perturbative prediction of HEP interactions among the most fundamental particles (leptons, quarks)

  2. ***Hadronization***: Quarks turn to jets of particles via Quantum Chromodynamics (QCD) at energies where theory is too strong to compute perturbatively.

     ➡ Use semi-empirical models tuned to Data.

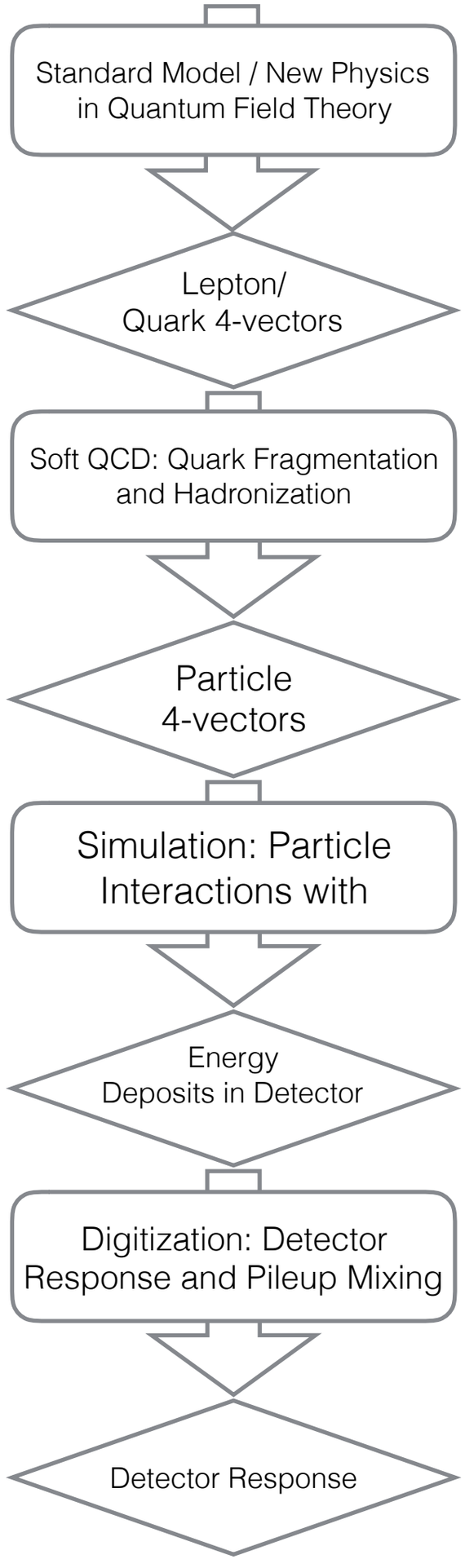
  3. ***Simulation***: Particles interact with the Detector via stochastic processes

     ➡ Use detailed Monte Carlo integration over the "micro-physics"

  4. ***Digitization***: Ultimately the energy deposits lead to electronic signals in the O(100 Million) channels of the detector.

     ➡ Model using test beam data and calibrations.

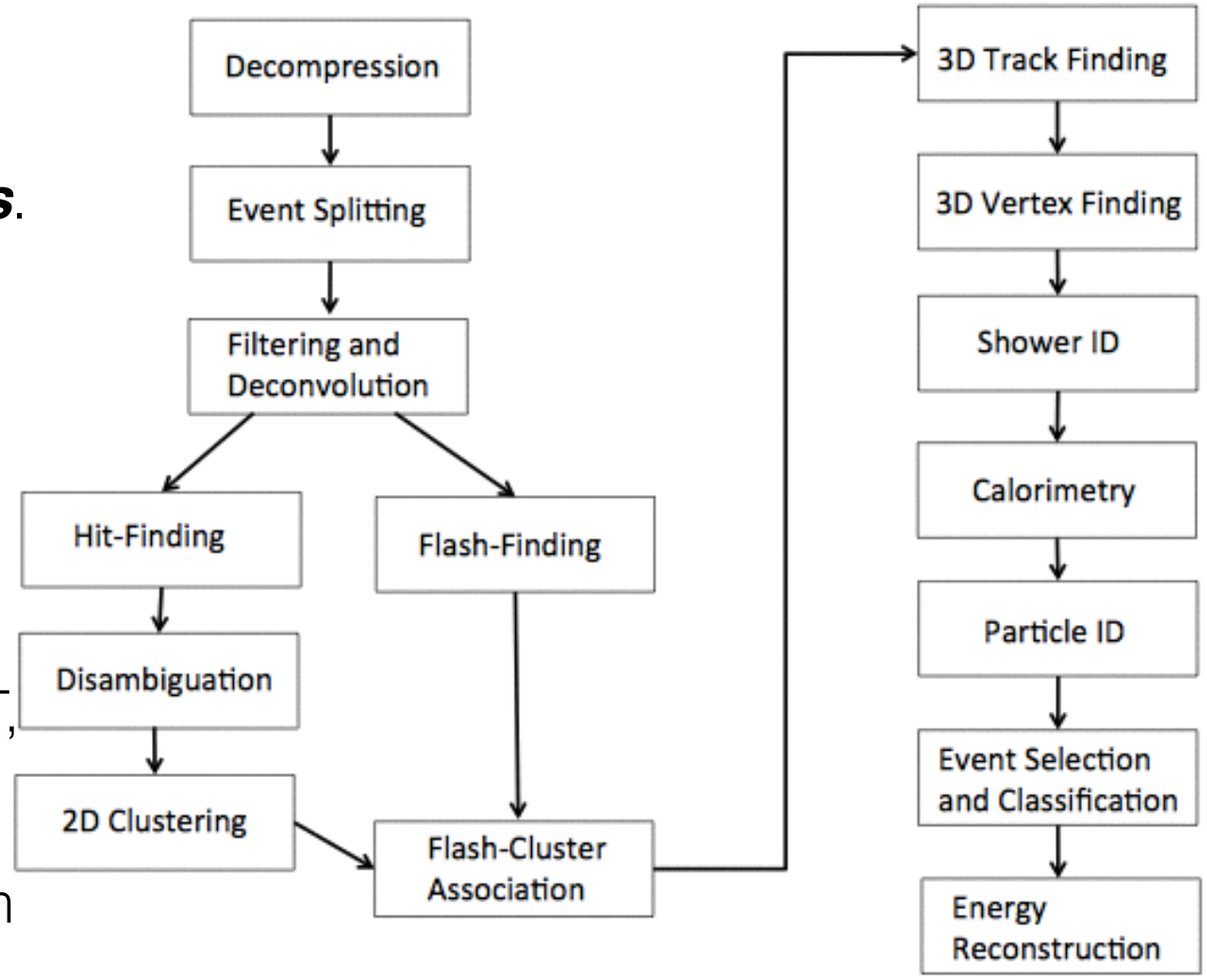
- Output is fed through ***same reconstruction as real data***.

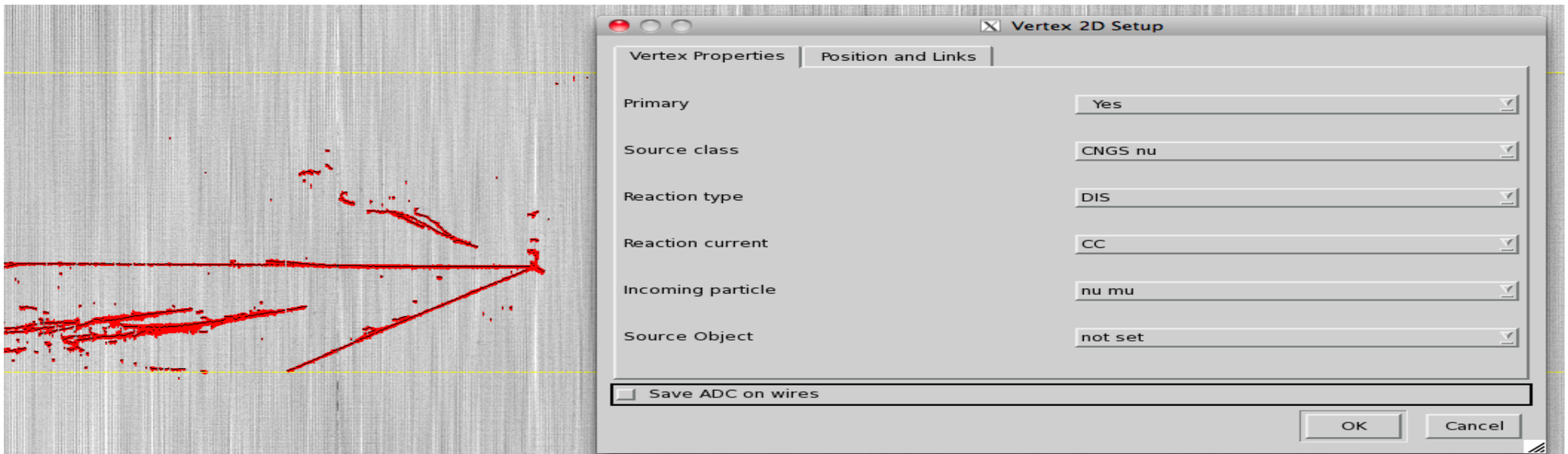


Problems in HEP

Reconstruction

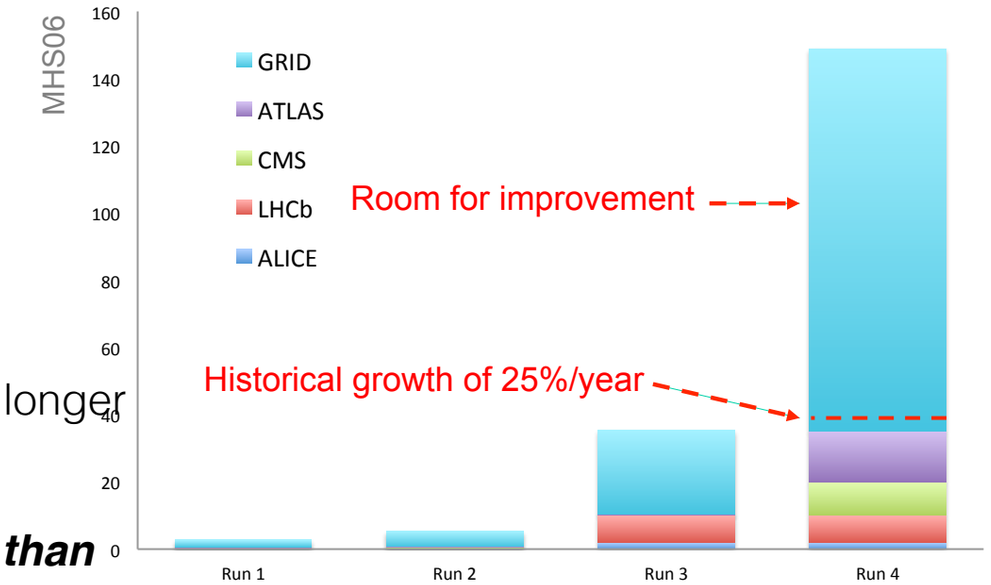
- Better and faster is always worthwhile… but there are unsolved problems too
- Neutrino Physics has a long history of ***hand scans***.
  - QScan: ICARUS user assisted reconstruction.
  - ***Full automatic reconstruction*** has yet to be demonstrated.
- LArSoft project: art framework + LArTPC reconstruction algorithms developed by LArIAT, MicroBooNE, DUNE, …
- Still… full neutrino reconstruction is still far from expected performance.

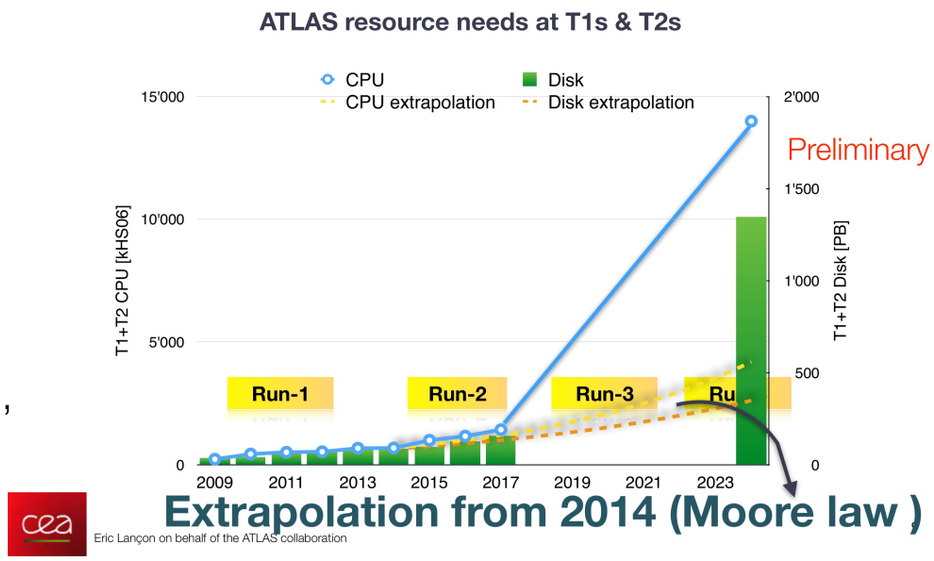




Computing

- ***Computing*** is perhaps the biggest challenge for the HL-LHC
  - ***Higher Granularity*** = larger events.
  - ***O(200) proton collision / crossing: tracking pattern recognition combinatorics becomes untenable***.
  - O(100) times data = multi ***exabyte datasets***.
  - ***Moore's law has stalled***: Cost of adding more transistors/silicon area no longer decreasing.
  - Preliminary estimates of ***HL-LHC computing budget many times larger than LHC***.
    - Problems: Tracking Pattern Recognition, Simulation (Geant and NNLO), Calorimetry (better trigger)
- ***Solutions***:
  - ***Leverage opportunistic resources and HPC*** (most computation power in highly parallel processors).
  - ***Highly parallel processors*** (e.g. GPUs) are already > 10x CPUs for certain computations.
    - Trend is away from x86 towards ***specialized hardware*** (e.g. GPUs, Mics, FPGAs, Custom DL Chips)
    - Unfortunately parallelization (i.e. Multi-core/GPU) has been extremely difficult for HEP.





Plots from here.

Likelihood Approximations

- Need $P(\{x_e\}|\theta)$ of an observed event (e). The better we do, the more sensitive our measurements.
- Steps 2 (Hadronization) and 3 (Simulation) can only be done in the ***forward mode***…
  - ➡ ***cannot evaluate the likelihood.***
- So we simulate a lot of events and use a ***Probability Density Estimator (PDE)***, e.g. a histogram.
  - $\{x_e\}$ = {100M Detector Channels} or even { particle 4-vectors } are too high dimension.
  - Instead we derive $\{x_e\}$ = { small set of physics motivated observables } → ***Lose information.***
    - ***Isolate signal*** dominating regions of $\{x_e\}$ → ***Lose Efficiency.***
    - Sometimes use ***classifiers*** to further reduce dimensionality and improve significance
    - ***Profile the likelihood*** in 1 or 2 (ideally uncorrelated) observables.
- Alternative, try to brute force calculate via ***Matrix Element Method***:

$$\mathcal{P}(\boldsymbol{p}^{vis}|\alpha) = \frac{1}{\sigma_\alpha}\int d\Phi dx_1 dx_2 |M_\alpha(\boldsymbol{p})|^2 W(\boldsymbol{p}, \boldsymbol{p}^{vis})$$

- But it's technically difficult, computationally expensive, mistreats hadronization, and avoids simulation by highly simplifying the detector response.

Summary: Why DL+HEP?

- HEP handles world's largest datasets.
    - HEP scientists regularly apply and develop "Data Science" techniques
    - HEP scientists are well versed in data engineering.
- Build Models
    - Physics models are mathematical based on minimal set of rules with as little empirical modeling as possible.
    - ML/DL models are completely empirical… but written in mathematical language familiar to HEP.
- Hard problems in HEP require new approaches… such as DL. More at later lecture.
    - Insufficient computing resources for upgrade to LHC.
    - Some cases, difficulty writing algorithms that perform as well as human eye.
    - …

“Lecture”

DL Basics

Ingredients of ML

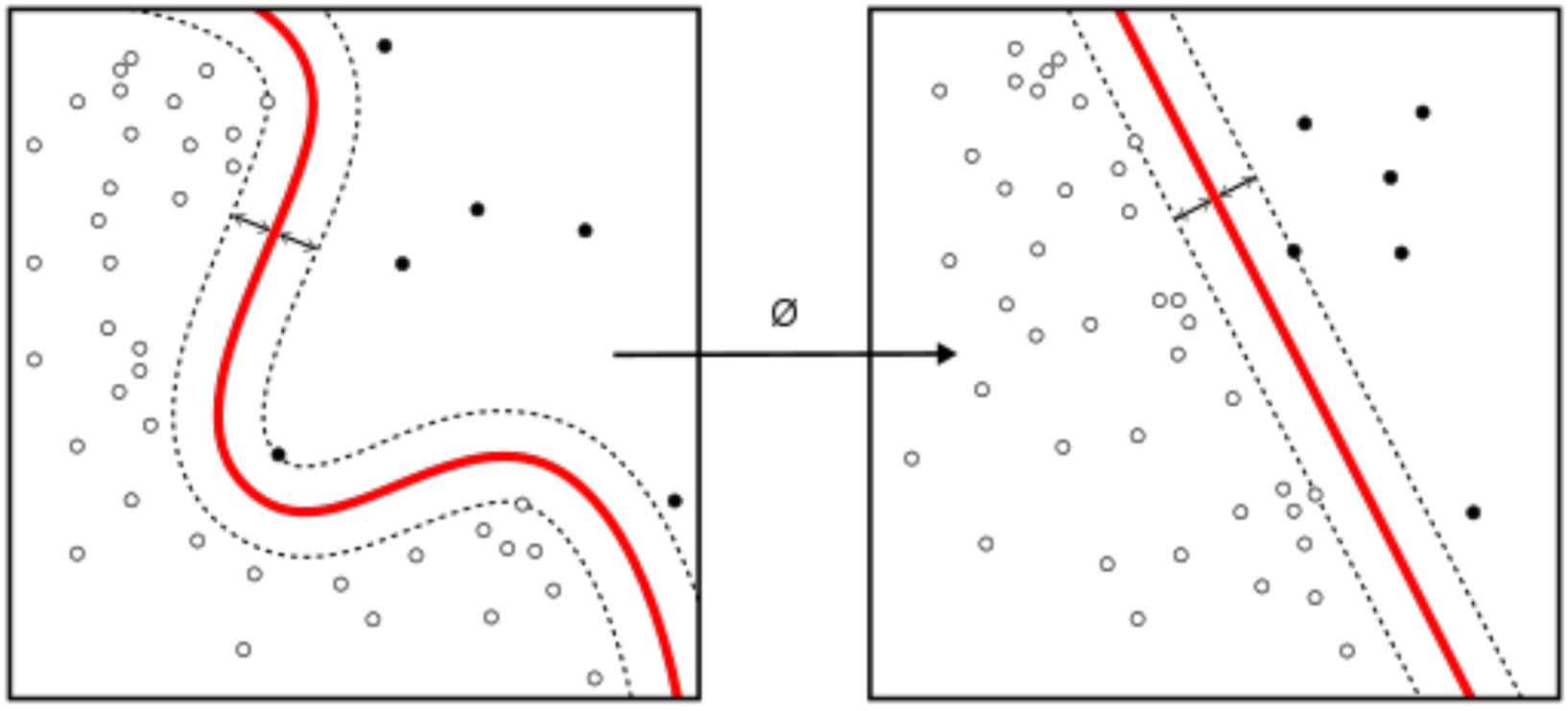
- ***Problem Formulation***. Specify:
  - ***Data Set***: Inputs/Outputs
    - F (Input | Parameters) = Output
  - ***Cost (aka loss) function***
    - Compare outputs: ML vs Ground Truth
  - ***Optimization***
    - Choose how to find best parameters

Data Basics

- How do we train neural networks… or apply them (inference)?
- Data are stored in “tensors”.
    - Basically an N- Dimensional Array with a “shape”
        - shape = (): Scalar
        - shape = (N,): Vector
        - shape = (N,M): Matrix
        - shape = (N_1, N_2, N_3, …, N_R): Rank R Tensor
- Inputs: **X**
    - Can be arbitrary shape. Typically first dimension is the example index (usually an “event” or collision in HEP)
    - Example: Let’s say your examples are students, and your data is their age, sex, years at UTA, undergrad/grad, and department
        - X = [[20, 0, 2, 0, 4] , # 20 year old, 0=male, 2=junior, 0=undergrad, 4=computer science
                [25, 1, 2, 1, 3] , # 25 year old, 1=female, 2=3nd year, 0=grad, 4=physics
                [23, 0, 0, 1, 3]] # 25 year old, 1=make, 2=1st year, 0=grad, 4=physics
        - X[0] = [20, 0, 2, 0, 4]: the first students data.
        - X[0][3] = 1. This is a graduate student
- Outputs : **Y**
    - Can be arbitrary shape. Typically first dimension is the example index (usually an “event” or collision in HEP)
    - Example: Y = 0/1, student does not / does know python

Learning

- Supervised
  - Classification
  - Regression
- Transfer
- Unsupervised
  - Clustering
- Semi-supervised
  - Auto-encoders
- Reinforcement



Supervised

- ***Tasks***: Classification, Classification with missing inputs, Regression, Transcription, Machine Translation, Structured Output
- “Traditional” Techniques:
  - Linear/Logistic Regression
  - Support Vector Machines
  - Decision Trees

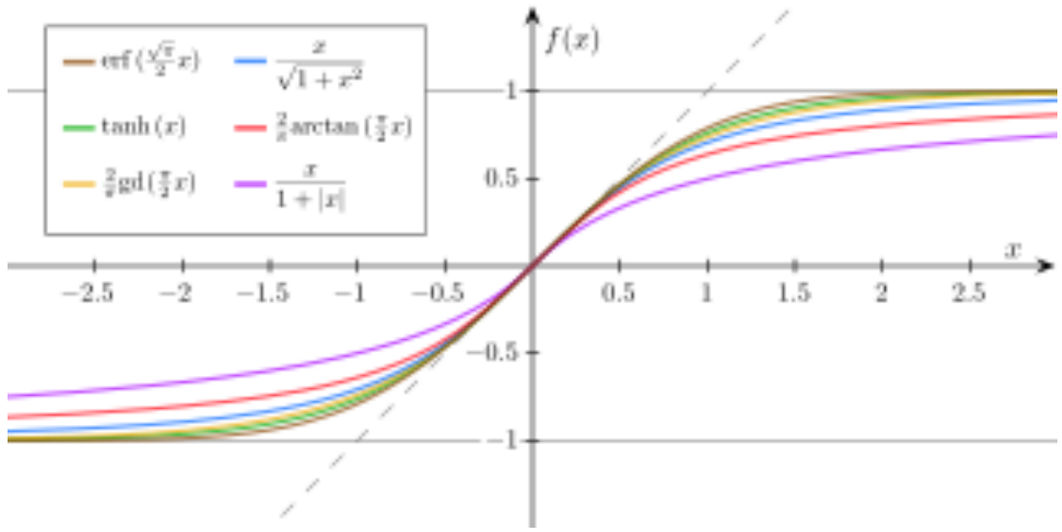
Un-supervised

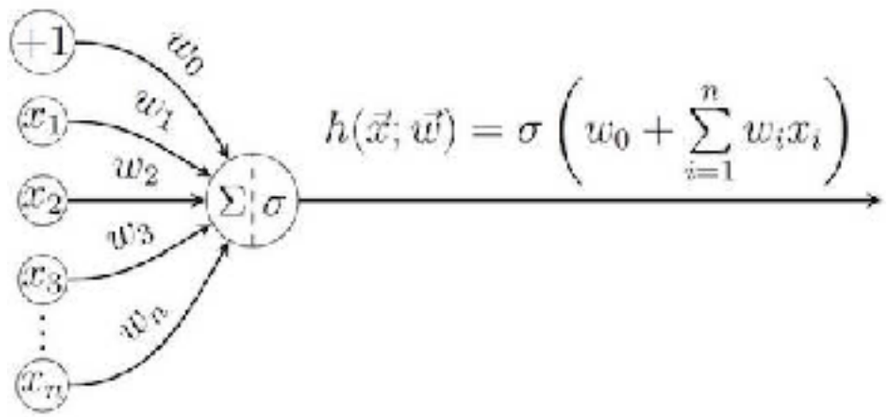
- ***Tasks***: Clustering, Anomaly Detection, Imputation of Missing Values, Synthesis & Sampling, Denoising, Density Estimation

- “Traditional” Techniques:
  - Principle Component Analysis
  - k-means Clustering

Artificial Neural Network

- A simple one layer NN
  - F(**X** | **a**= **W**,**b**) = f (**W X** + **b**)
  - **W, b** = “weights”, “biases”
  - f(x)= “activation function”
    - Must be non-linear.



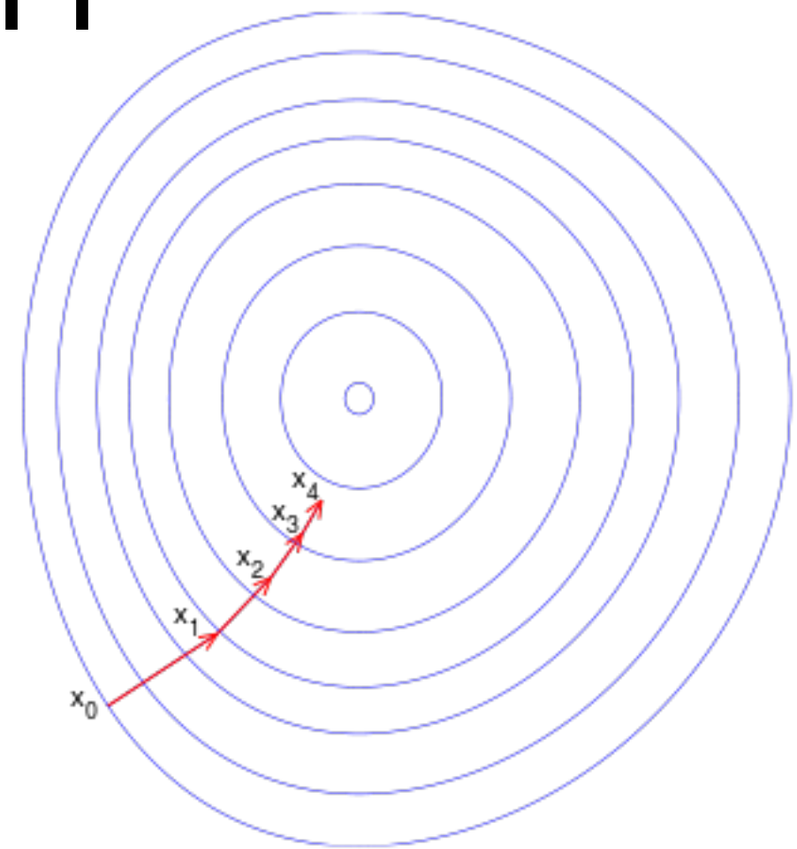


Optimization

- Training = Minimizing cost function w.r.t. parameters **α**

$$C[F(\vec{X}_{train}|\vec{\alpha}), \vec{Y}_{train}] \equiv C(\vec{\alpha})$$

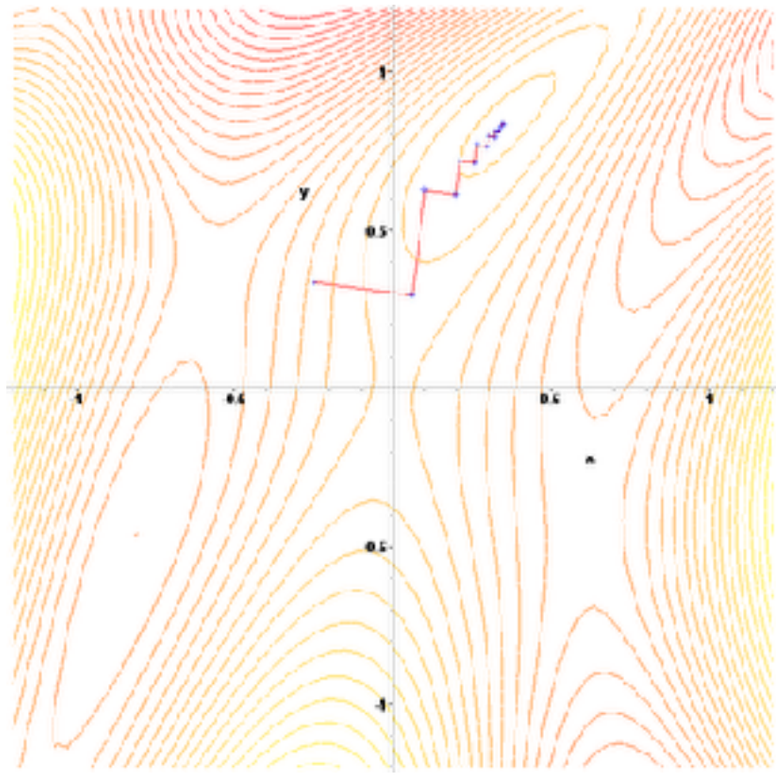
- Gradient Decent (Newton's Method):



- Gradient points to direction of maximal change.

- Iterate (ε sets the step size)

$$\vec{\alpha}_{i+1} = \vec{\alpha}_i - \epsilon \nabla C(\vec{\alpha})$$



Bayesian vs Frequentist

- Supervised Learning:
  - Data: (X , Y)
  - True: f*(X) = Y
  - Learn f(X I w) ~ f*
- Frequentist:
  - (X,Y) random.
  - Ideal w exists.
  - Estimate w.
- Bayesian:
  - (X,Y) fixed.
  - w is random.
  - Learn p(w)

Machine Learning Problem Formulation

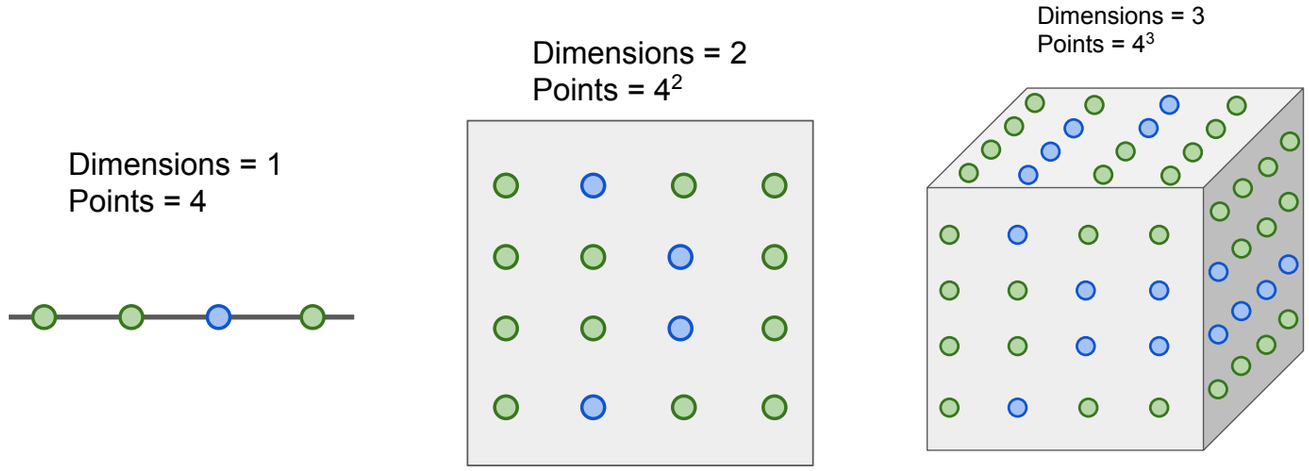
- Split *Datasets*:
  - ($\mathbf{X}_{train}$, $\mathbf{Y}_{train}$) = training dataset
  - ($\mathbf{X}_{test}$, $\mathbf{Y}_{test}$) = test dataset
  - ($\mathbf{X}$) = unlabeled data
- Set *Goal*:
  - *Inference* algorithm/function F($\mathbf{X}$ | $\boldsymbol{\alpha}$) = $\mathbf{Y}_{predict}$.
    - F can be a heuristic. e.g. if (computer science student) then (student knows python).
    - F can be anything
  - $\boldsymbol{\alpha}$ are parameters of the function, for Neural Networks, these are weights.
    - Note that in a simple classification problem, $\mathbf{Y}_{train}$ can be 0 or 1 for any example. But $\mathbf{Y}_{predict}$ will usually be between 0 and 1.
- *Training*: (for Neural Networks)
  - Optimize (usually a minimization) a *cost function* F($\mathbf{X}$ | $\boldsymbol{\alpha}$) = C(F($\mathbf{X}_{train}$| $\boldsymbol{\alpha}$), $\mathbf{Y}_{train}$) w.r.t. $\boldsymbol{\alpha}$
  - For example, C = [F($\mathbf{X}$ | $\boldsymbol{\alpha}$) - $\mathbf{Y}_{train}$]^2
  - $\boldsymbol{\alpha}_{trained}$= result of training
- Validation:
  - Compute cost function on test data C(F($\mathbf{X}_{test}$| $\boldsymbol{\alpha}_{trained}$), $\mathbf{Y}_{test}$)
  - Other metrics. For example:
    - Select $\mathbf{Y}_{test}$=1 and see how often F($\mathbf{X}_{test}$| $\boldsymbol{\alpha}_{trained}$) > 0.5
- Inference:
  - $\mathbf{Y}_{predict}$ = F($\mathbf{X}$| $\boldsymbol{\alpha}_{trained}$)

Deep Learning Techniques & Examples in HEP

Feature Learning

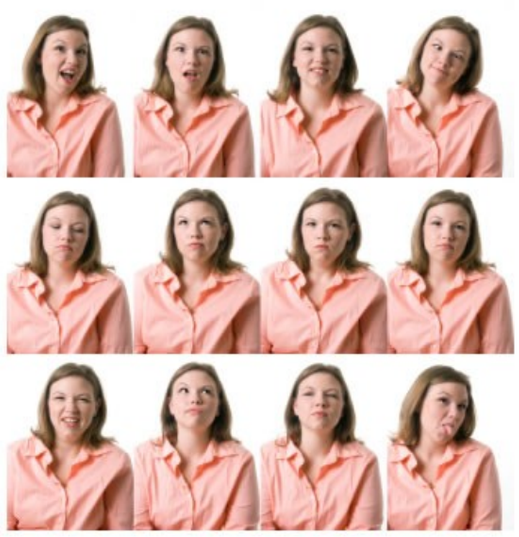
Motivations

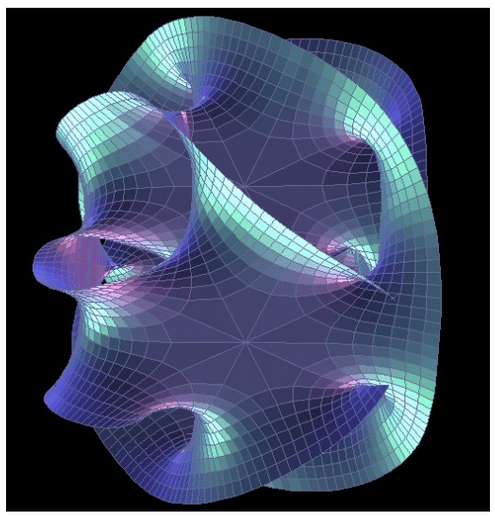
- Curse of Dimensionality
- Smoothness
- Manifold Learning

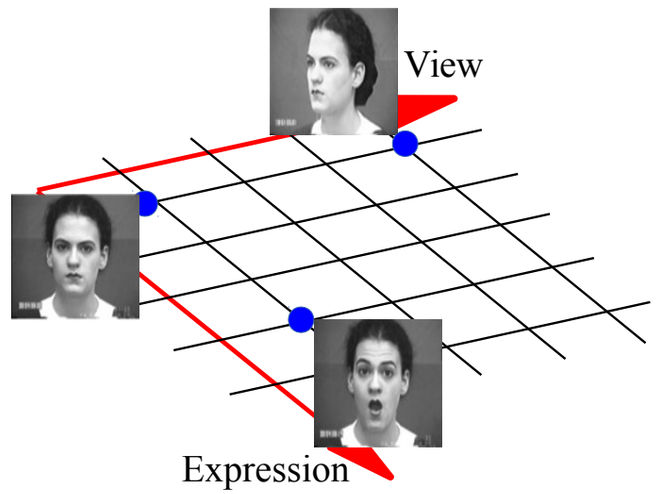


- Natural Data Lives in low dimensional (Non-Linear) Manifold.

Pics from Yann Lacun and Fei Fei Li Lectures



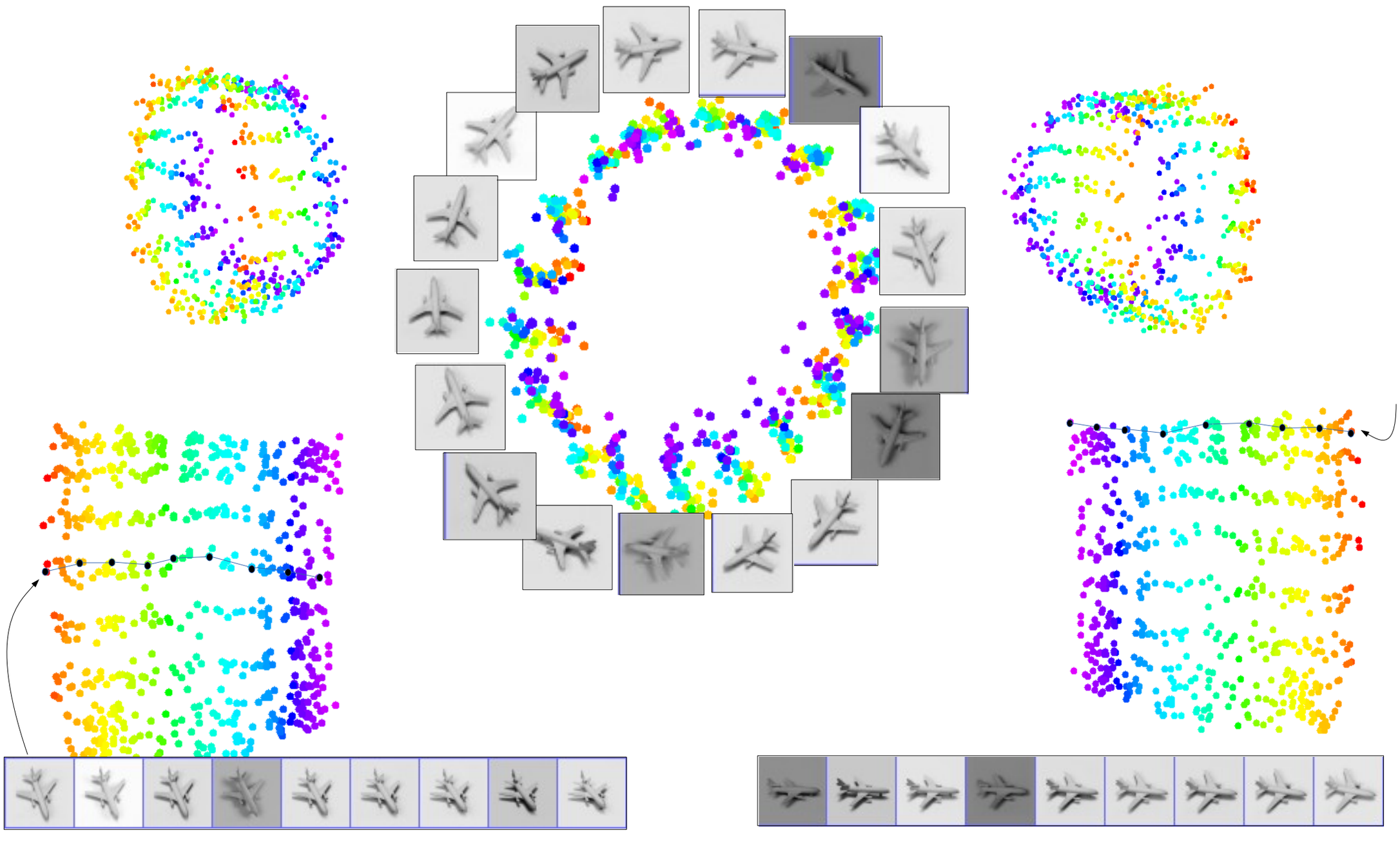




Data Manifold & Invariance: Some variations must be eliminated

Y LeCun
MA Ranzato

- Azimuth-Elevation manifold. Ignores lighting. [Hadsell et al. CVPR 2006]



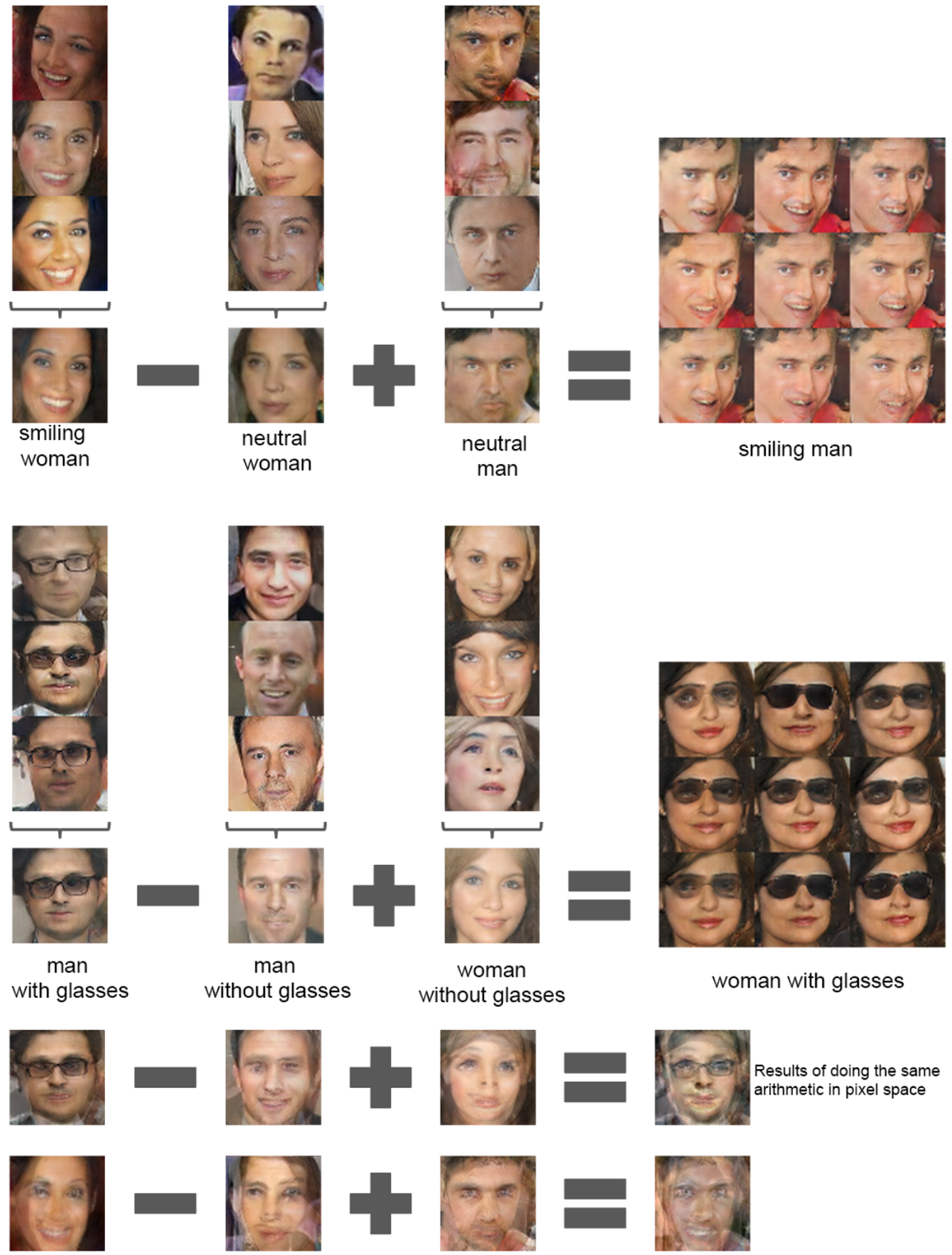
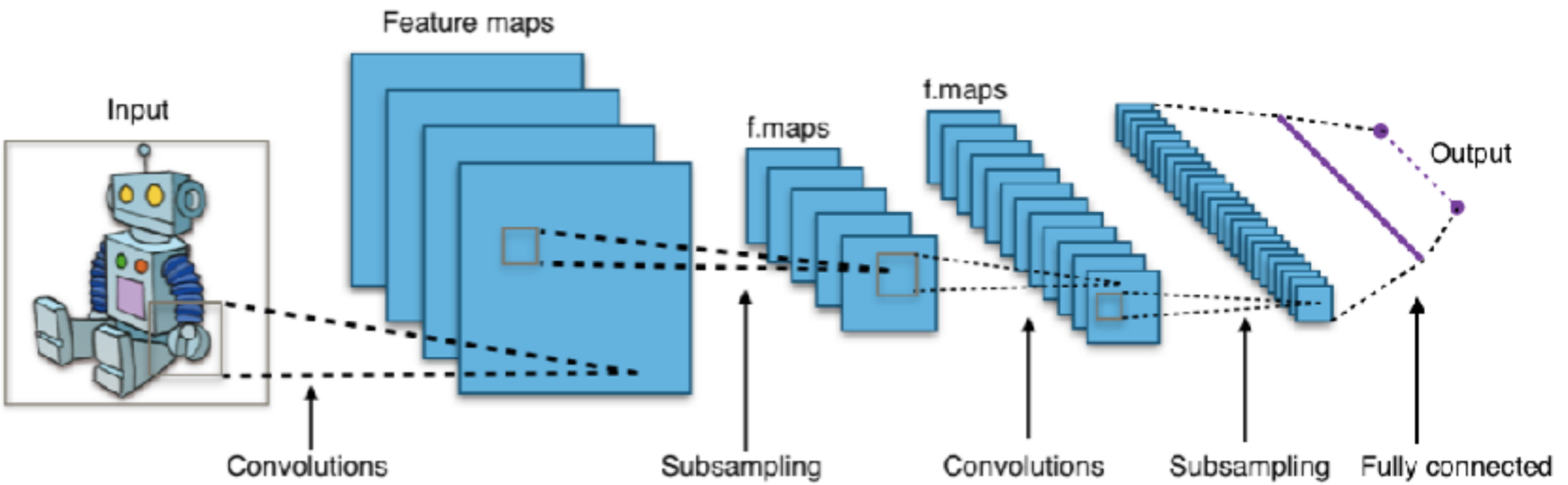


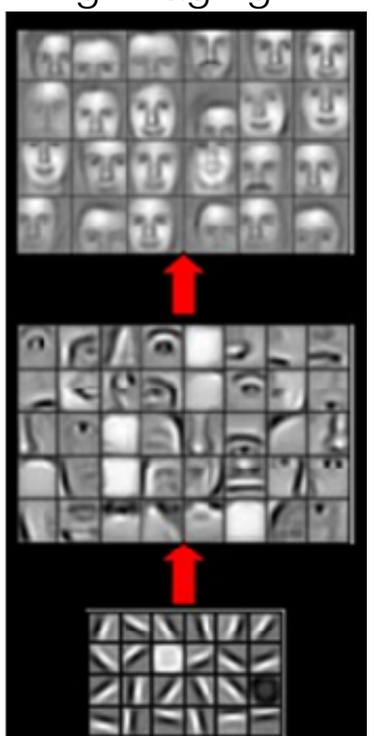
Figure 7: Vector arithmetic for visual concepts. For each column, the Z vectors of samples are averaged. Arithmetic was then performed on the mean vectors creating a new vector Y. The center sample on the right hand side is produce by feeding Y as input to the generator. To demonstrate the interpolation capabilities of the generator, uniform noise sampled with scale +-0.25 was added to Y to produce the 8 other samples. Applying arithmetic in the input space (bottom two examples) results in noisy overlap due to misalignment.

Feature Learning

- ***Feature Engineering***: e.g. Event Reconstruction ~ Feature Extraction, Pattern Recognition, Fitting, …
- Deep Neutral Networks can ***Learn Features*** from ***raw data***.
- Example: ***Convolutional Neural Networks*** - Inspired by visual cortex
  - ***Input***: Raw data… for example 1D = Audio, 2D = Images, 3D = Video
  - ***Convolutions*** ~ learned feature detectors
  - ***Feature Maps***
  - ***Pooling*** - dimension reduction / invariance
  - ***Stack***: Deeper layers recognize higher level concepts.

- Over the past few years, CNNs have lead to ***exponential improvement / superhuman performance on Image classification*** challenges. Current best > 150 layers.
- Obvious HEP application: ***"Imaging" Detectors*** such as TPCs, High Granularity Calorimeters, or Cherenkov Ring Imaging.





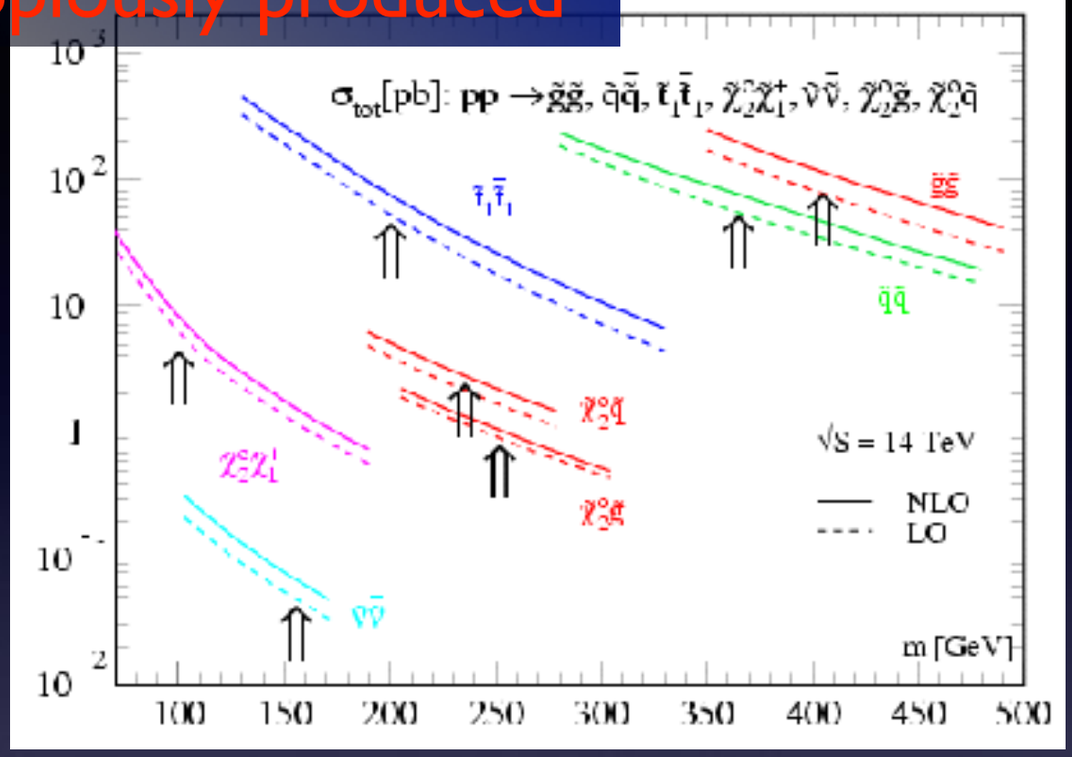
Why Deep?

- "We can approximate any function as close as we want with shallow architecture. Why would we need deep ones?"
    - ***Deep machines are more efficient*** for representing certain classes of functions
    - They can represent more complex functions with less "hardware"
- ***Hierarchy of representations*** with increasing level of abstraction.
    - ***Images***: Pixel→Edge→Texton→Motif→Part→Object
    - ***Text***: Character→Word→Word Group→Clause→Sentence→Story
    - ***Speech***: Samples→Spectral Band→Sound→…→Phone→Phoneme→Word
- In DL, these are learned features…
    - Each stage transforms input representation into high-level representation
    - High-level are more global/invariant
    - Low-level are shared among categories.

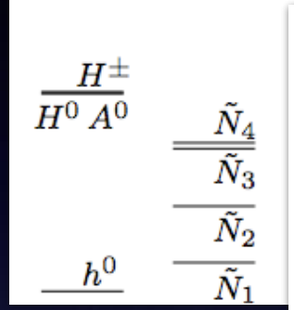
HEP Searches (SUSY Example)

SUSY at LHC

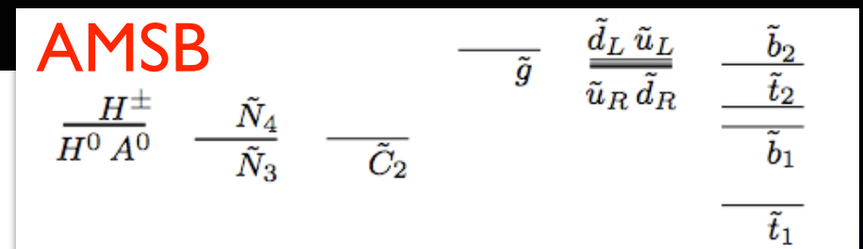
Gluinos and squarks copiously produced*



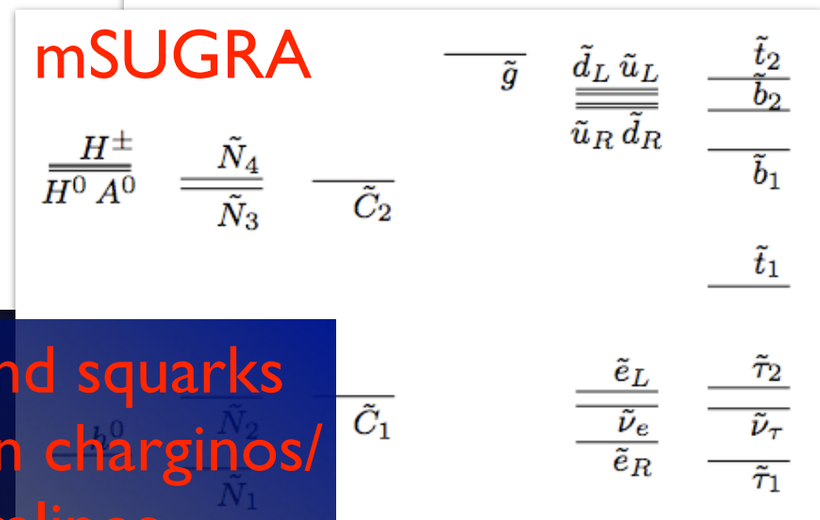
GMSB



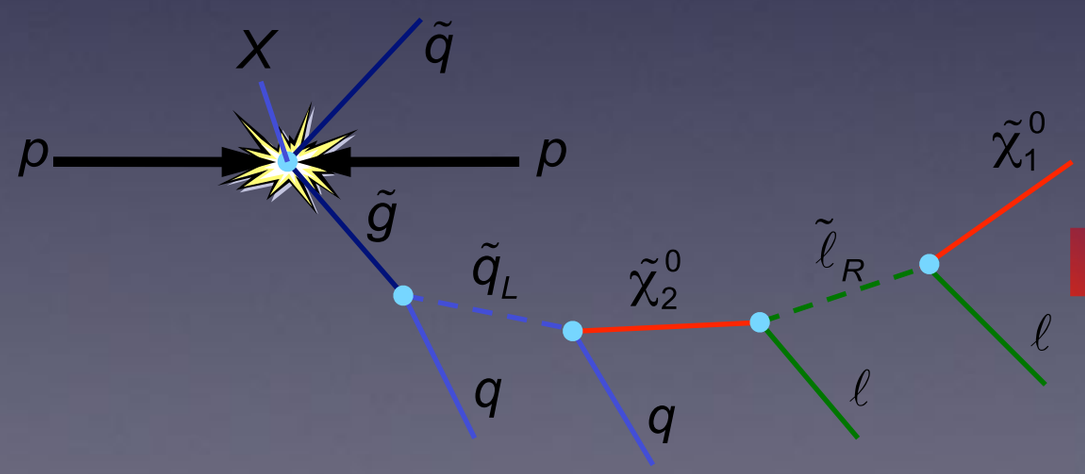
AMSB



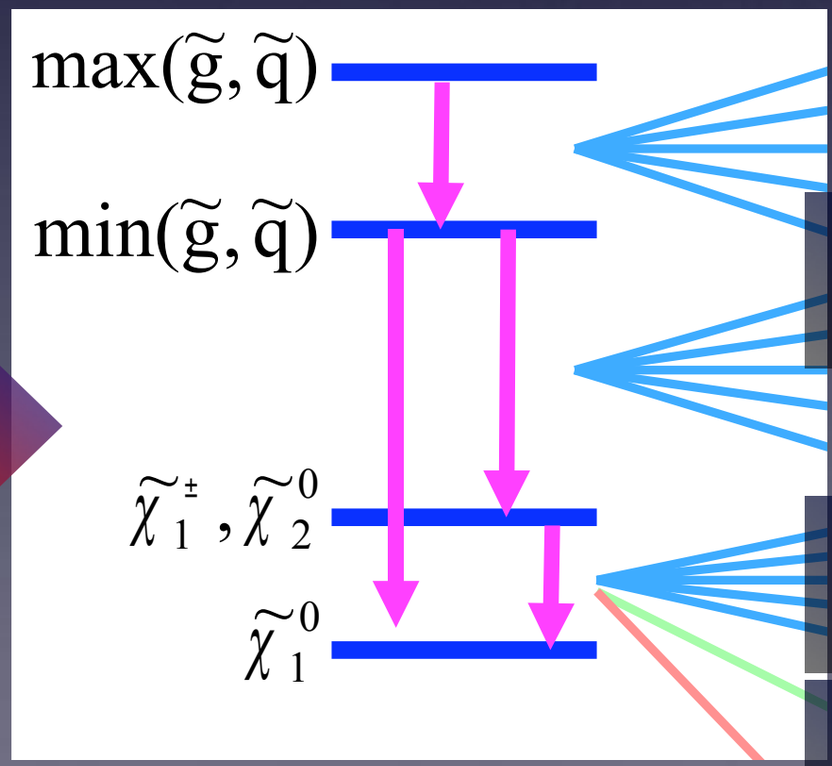
mSUGRA



Gluinos and squarks heavier than charginos/ neutralinos



Long Cascades



Many Jets (100's of GeV)

Leptons (10's of GeV)

Missing Energy (100's of GeV)

Inclusive Signatures

Signature	Motivating Model(s)	Comments

1 Jet + 0 Lepton + MET	• Large Extra Dim (ExoGraviton)
	  • strong qG production, G propagate in extra Dim
	  • Planck Scale is MD in 4+δ dim
	  • Normal Gravity >> R
	• SUSY
	  • qg→ISR + 2 Neutralino or squark + Neutralino
		• Not primary discovery channel for SUGRA, GMSB, AMSB... but helps in characterization
		• Possible leading discovery for neutralino NLSP with nearly degenerate gluino

2,3,4 [b]-Jet + 0 Lepton + MET	• Squark/gluino production • squark→q+LSP, gluino→q+squark+LSP	• Possible leading squark/ gluino discovery channel • Must manage QCD bkg
2,3,4 [b]-Jet + 1 Lepton + MET	• squark/gluino production with cascades which include electroweak (or partner) decays • high tan β leads to more b/t/τ's	• Lepton requirement suppresses QCD • τ's partially covered by e/μ
2 lepton + MET	• Same sign: gluino cascade can have either sign lepton... squark/gluino prod can produce same sign. • Opposite sign: squark/gluino decay mediated by Z (or partner) • Same flavor: 2 leptons from same sparticle cascade must be same flavor	• Reduced SM backgrounds for same sign • Opposite Sign-Flavor Subtraction
3 lepton + MET	• SUSY events ending in Chargino/neutralino pair decays • Weak Chargino/Neutralino production • Exotic sources	• Low SM bkgs
2 photon + MET	• GMSB models with gravitino LSP and neutralino or stau NLSP • UED- each KK partons cascade to LKP which decays to graviton + γ	• No SUSY limit (not sensitive at the time)

0 Lepton Event Selections

- No leptons (medium electrons and muons) >10 GeV
- 4 signal regions defined to maximize m_{squark}-m_{gluino} coverage :
  - At least 2 Jets
    - Low mass squark anti-squark (A)
    - High mass squark anti-squark (B)
  - At least 3 Jets
    - Direct gluino pairs (C)
    - Associated gluino-squark (D)
      - Higher x-section → Tighter cuts!

		A	B	C	D
Pre-selection	Number of required jets	≥ 2	≥ 2	≥ 3	≥ 3
	Leading jet p_{T} [GeV]	> 120	> 120	> 120	> 120
	Other jet(s) p_{T} [GeV]	> 40	> 40	> 40	> 40
	$E_{\mathrm{T}}^{\mathrm{miss}}$ [GeV]	> 100	> 100	> 100	> 100
	$\Delta\phi(\mathrm{jet}, \vec{p}_{\mathrm{T}}^{\mathrm{miss}})_{\mathrm{min}}$	> 0.4	> 0.4	> 0.4	> 0.4
Final selection	$E_{\mathrm{T}}^{\mathrm{miss}}/m_{\mathrm{eff}}$	> 0.3	–	> 0.25	> 0.25
	m_{eff} [GeV]	> 500	–	> 500	> 1000
	m_{T2} [GeV]	–	> 300	–	–

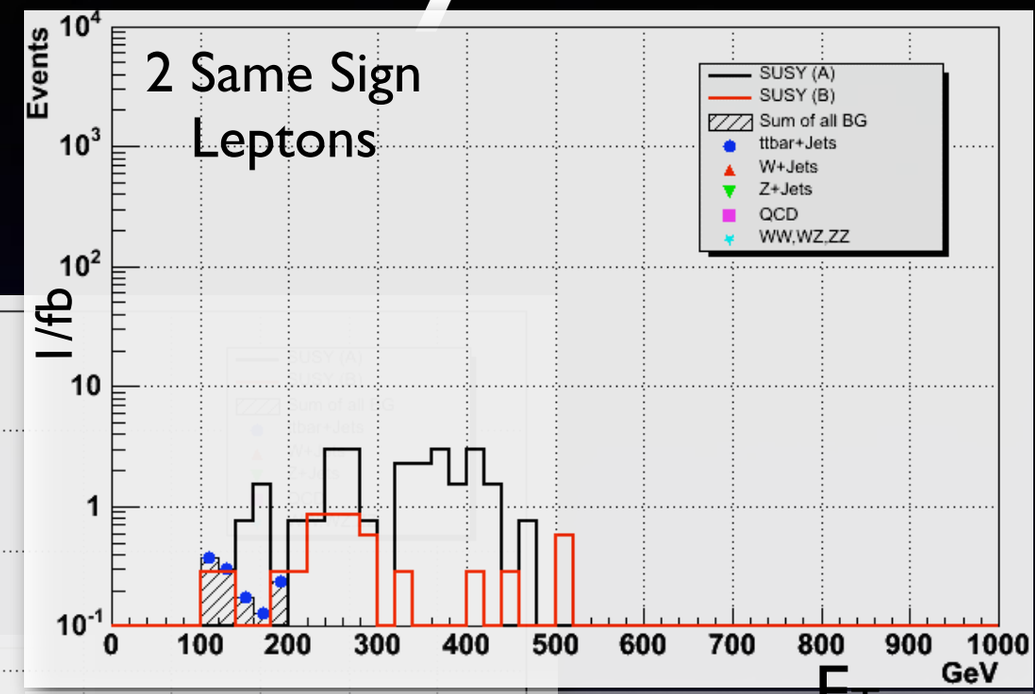
$$m_{\mathrm{eff}} \equiv \sum_{i=1}^{n} |\mathbf{p}_{\mathrm{T}}^{(i)}| + E_{\mathrm{T}}^{\mathrm{miss}}$$

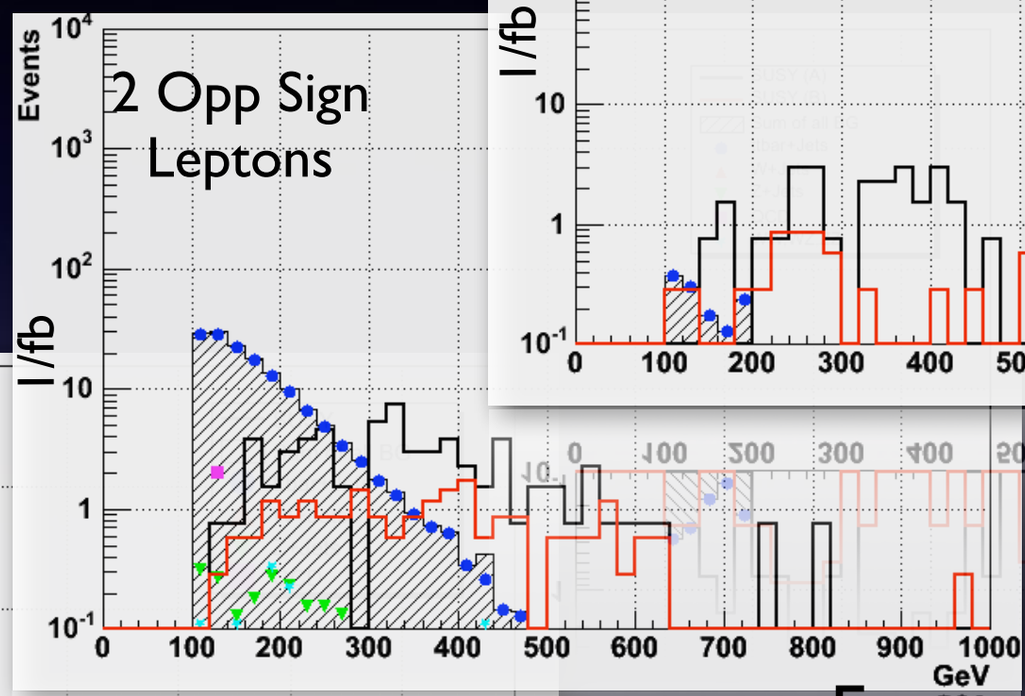
$$m_{\mathrm{T2}}\left(\mathbf{p}_{\mathrm{T}}^{(1)}, \mathbf{p}_{\mathrm{T}}^{(2)}, \not{\mathbf{p}}_{\mathrm{T}}\right) \equiv \min_{\not{\mathbf{q}}_{\mathrm{T}}^{(1)}+\not{\mathbf{q}}_{\mathrm{T}}^{(2)}=\tilde{\mathbf{E}}_{\mathrm{T}}^{\mathrm{miss}}} \left\{\max\left(m_{\mathrm{T}}\left(\mathbf{p}_{\mathrm{T}}^{(1)}, \not{\mathbf{q}}_{\mathrm{T}}^{(1)}\right), m_{\mathrm{T}}\left(\mathbf{p}_{\mathrm{T}}^{(2)}, \not{\mathbf{q}}_{\mathrm{T}}^{(2)}\right)\right)\right\}$$

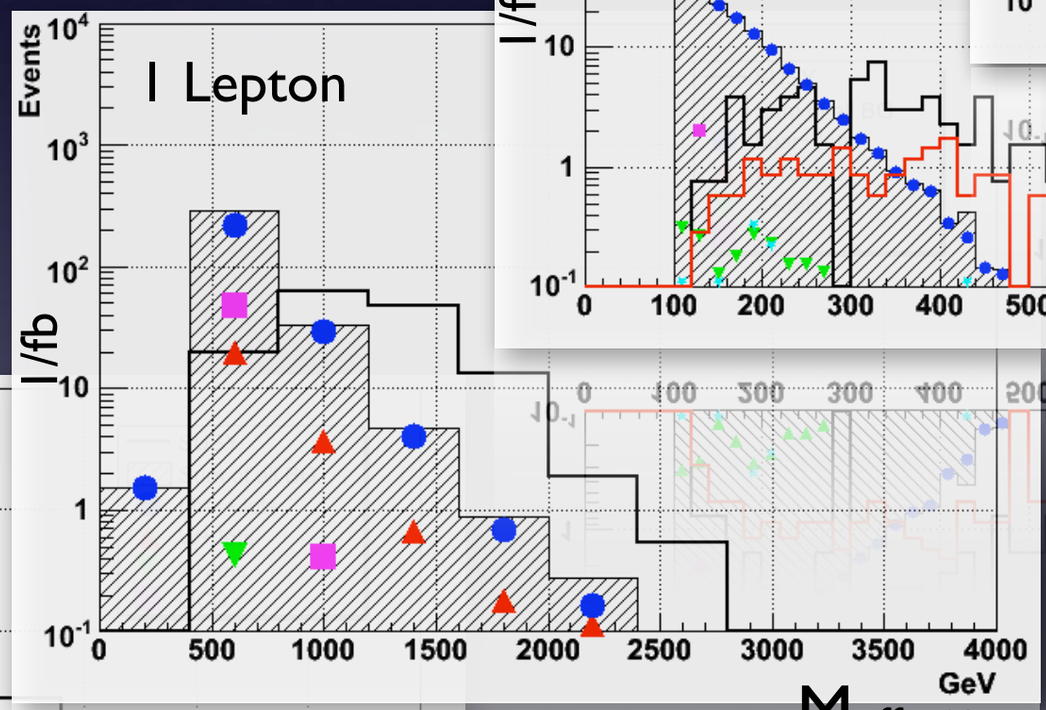
$$m_{\mathrm{T}}^{2}\left(\mathbf{p}_{\mathrm{T}}^{(i)}, \not{\mathbf{q}}_{\mathrm{T}}^{(\mathbf{i})}\right) \equiv 2|\mathbf{p}_{\mathrm{T}}^{(i)}||\not{\mathbf{q}}_{\mathrm{T}}^{(\mathbf{i})}| - 2\mathbf{p}_{\mathrm{T}}^{(\mathbf{i})} \cdot \not{\mathbf{q}}_{\mathrm{T}}^{(\mathbf{i})}$$

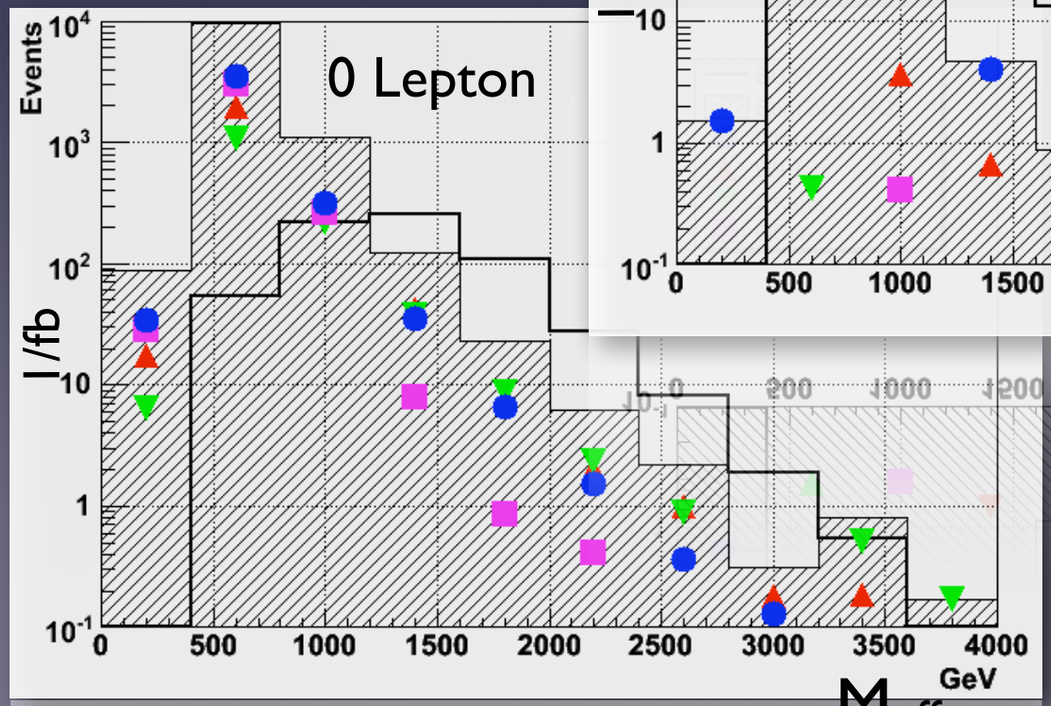
Standard SUSY Analyses

- Require:
  - Large $\not{E}_T$ (> 100 GeV)
  - 4 Hard Jets
  - Sphericity?
- Look at: $M_{eff} = \sum^4_{jets} p_T + \not{E}_T$ for N=0,1,2 (SS/OS) leptons





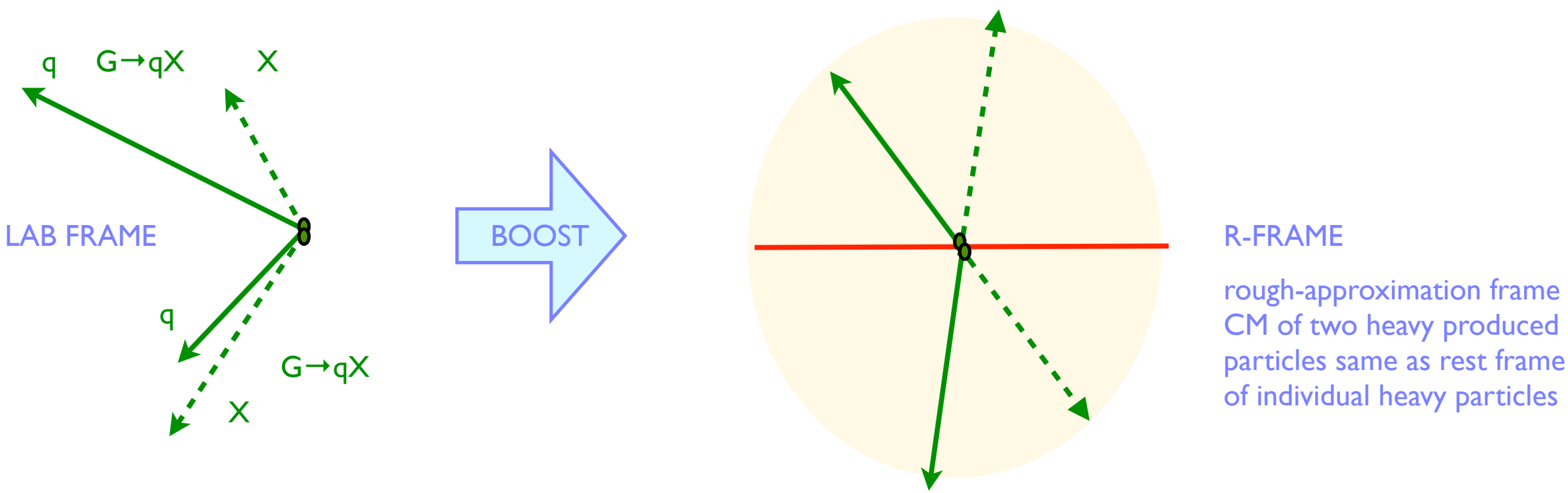




- Adding Leptons reduces QCD Background
- Better S/B in same sign than opposite sign

Razor variables

- Razor variables (C. Rogan arXiv:1006.2727) are kinematical variables to identify SUSY-like events
- Variables take advantage of symmetric decay of SUSY events by forming two hemispheres (aka mega-jets) using all final state visible objects



- define variables that take advantage of the symmetry of the SUSY event:

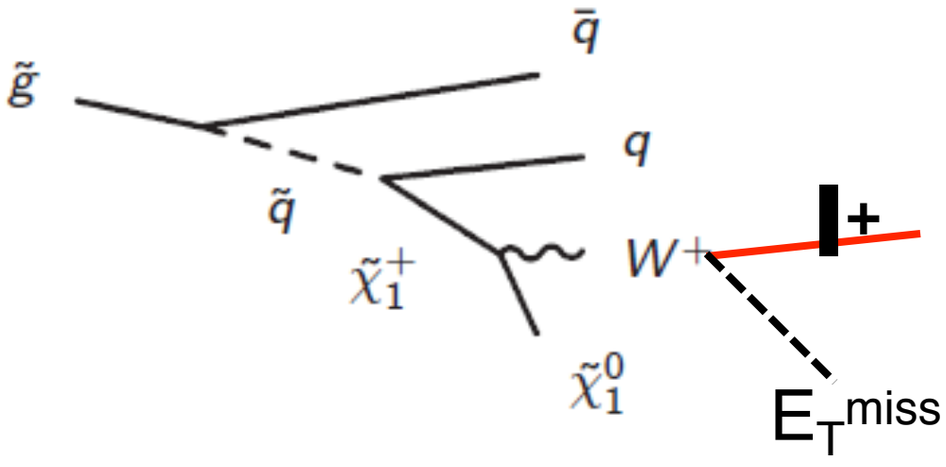
  $\gamma_R M_R$ contains longitudinal event information, related to the SUSY mass scale

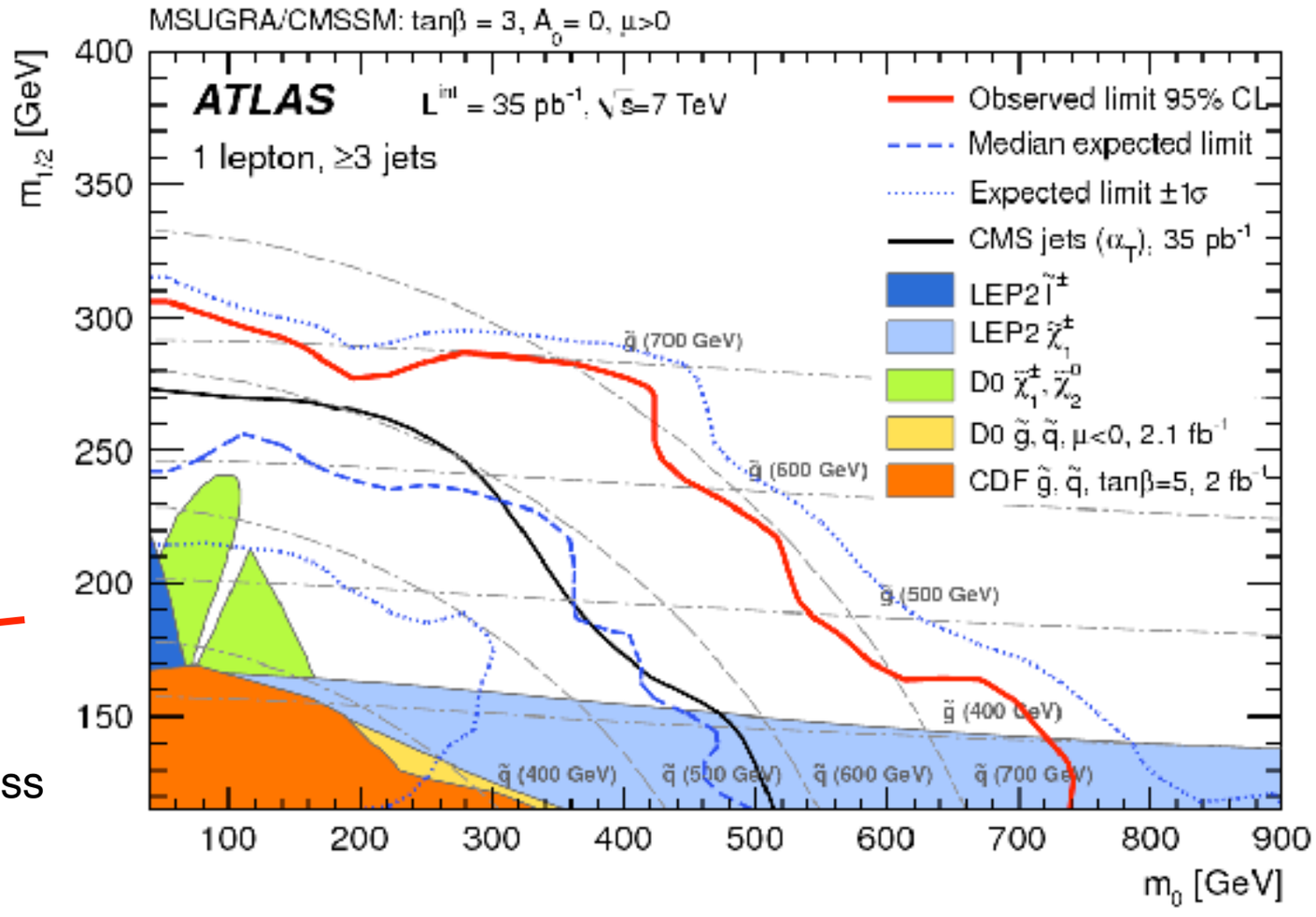
  M_T^R contains transverse event information

  $R = M_T^R / \gamma_R M_R$ a signal-to-background discriminant

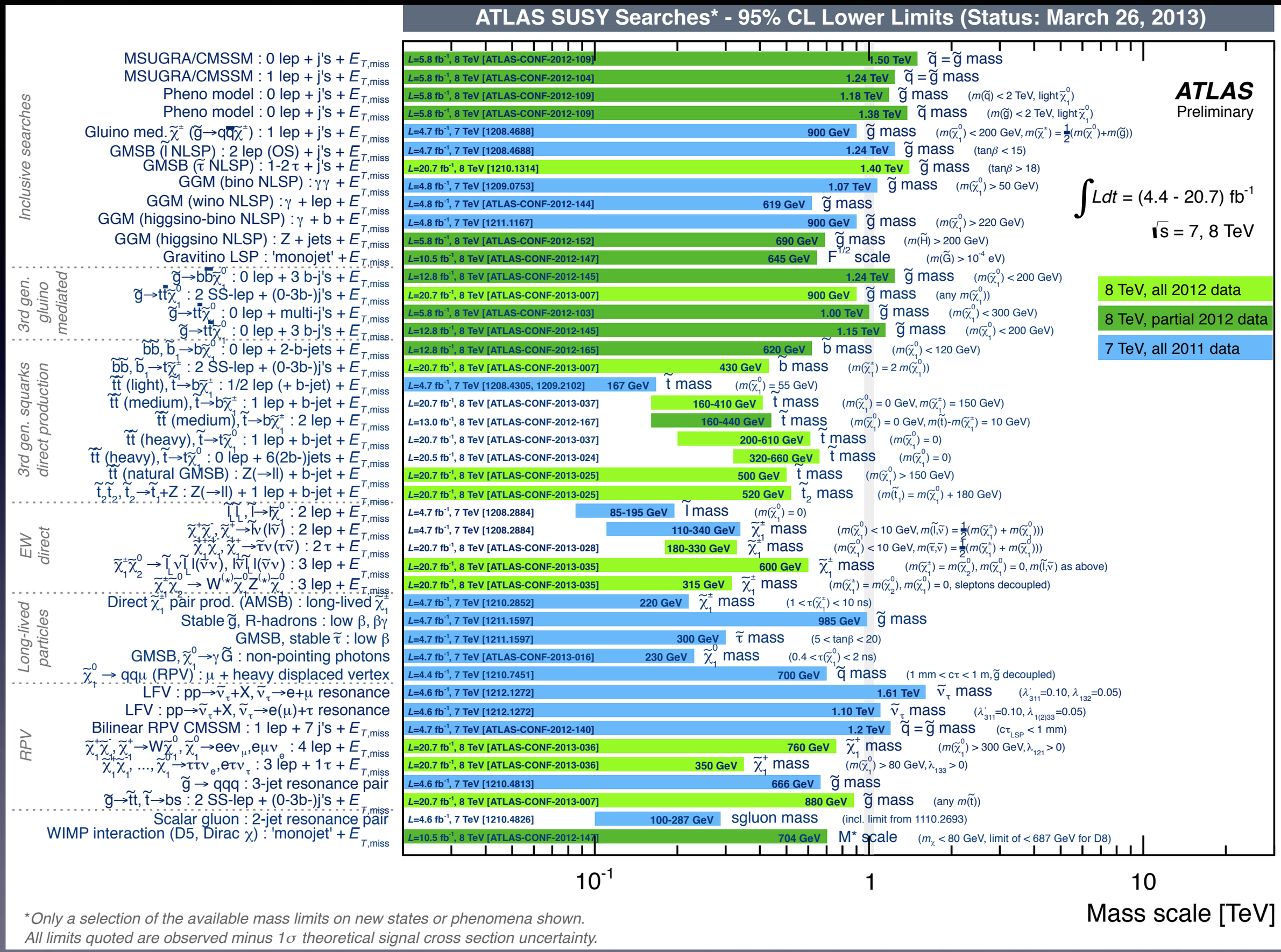
The 1st ATLAS SUSY paper

Phys. Rev. Lett. **106**, 131802 (2011)





- heavy colored particles production fully benefits from the LHC energy
  - if SUSY: gluinos & squarks
- surpassed Tevatron with only 0.035 fb^{-1}
  - one of the top-cited LHC papers

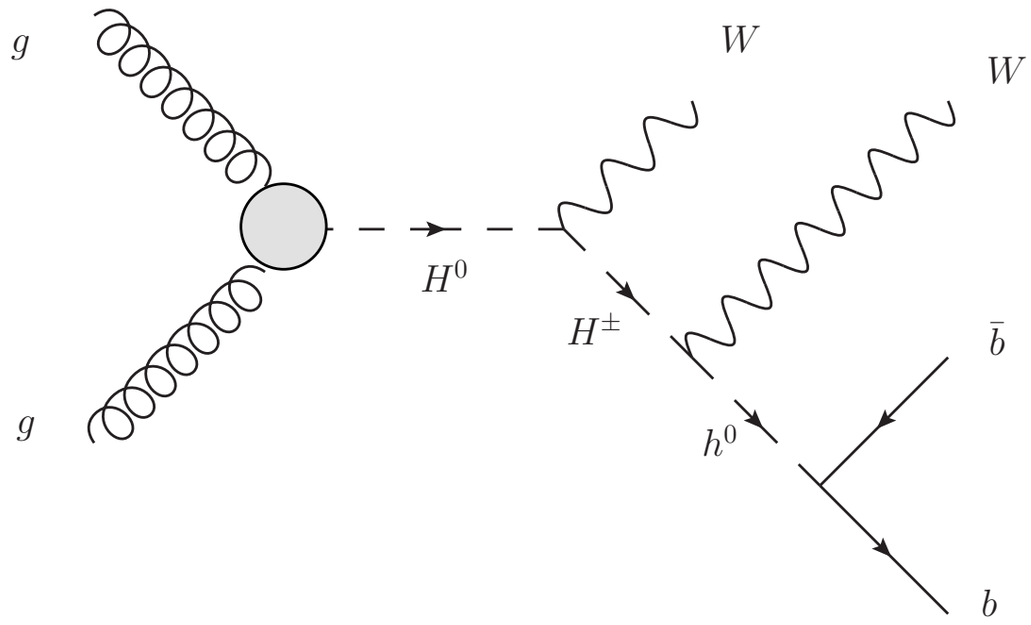


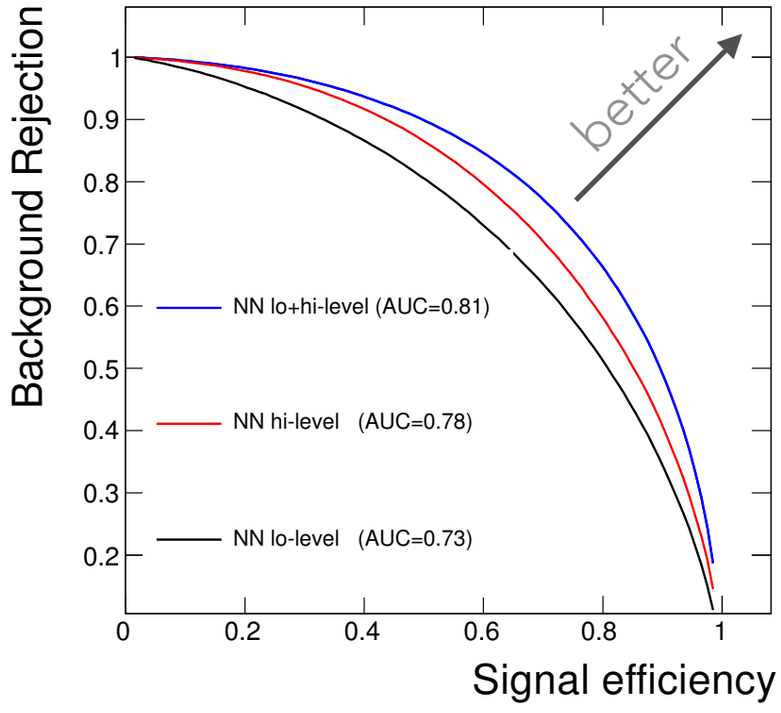
*Only a selection of the available mass limits on new states or phenomena shown.
All limits quoted are observed minus 1σ theoretical signal cross section uncertainty.

Higgs is at 125 GeV and no sign of new physics at LHC → Nature is not "natural"?

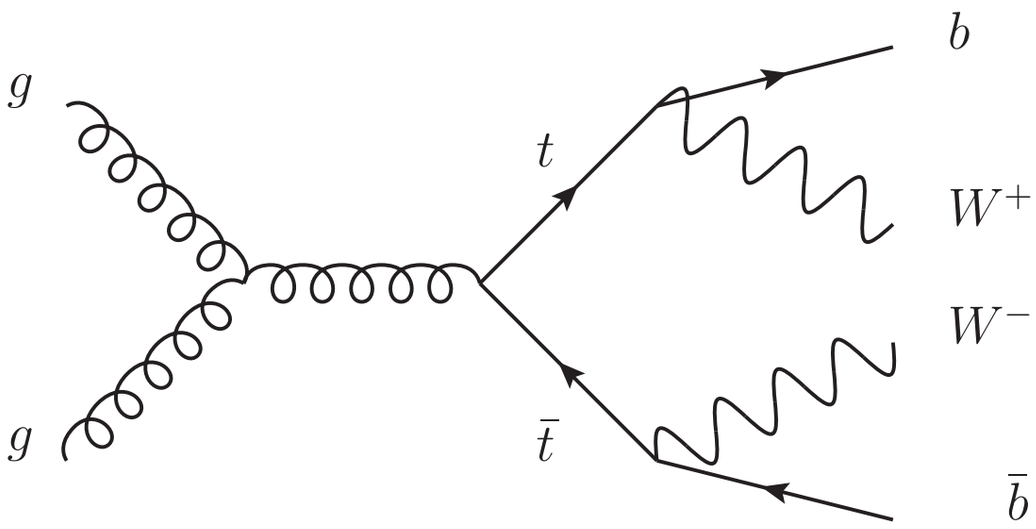
DEEP LEARNING IN HEP

Baldi, Sadowski,Whiteson
arxiv:1402.4735

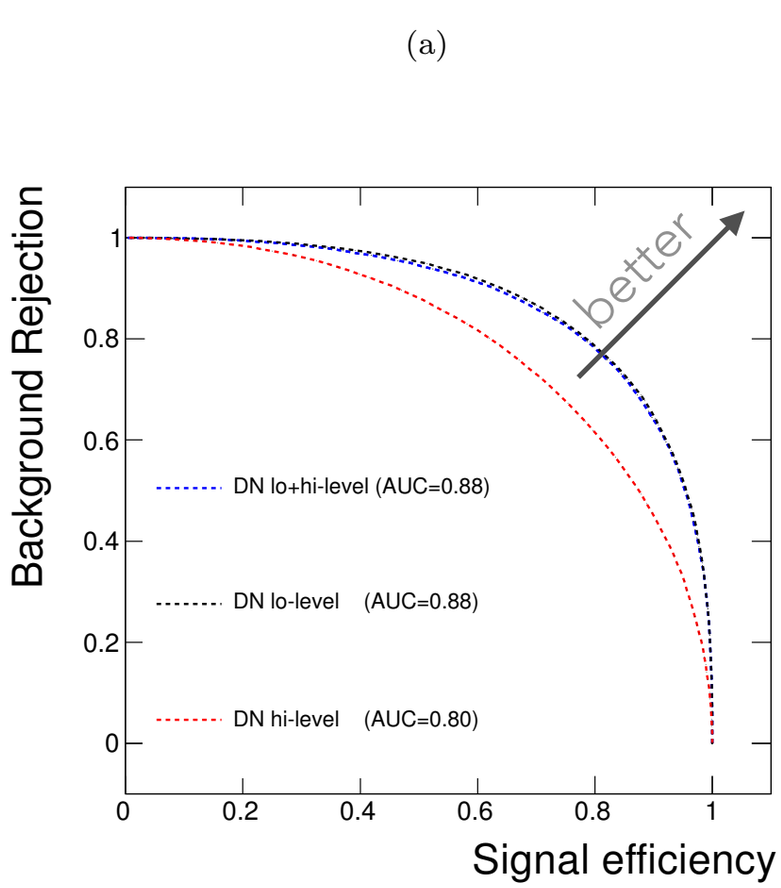




(a)



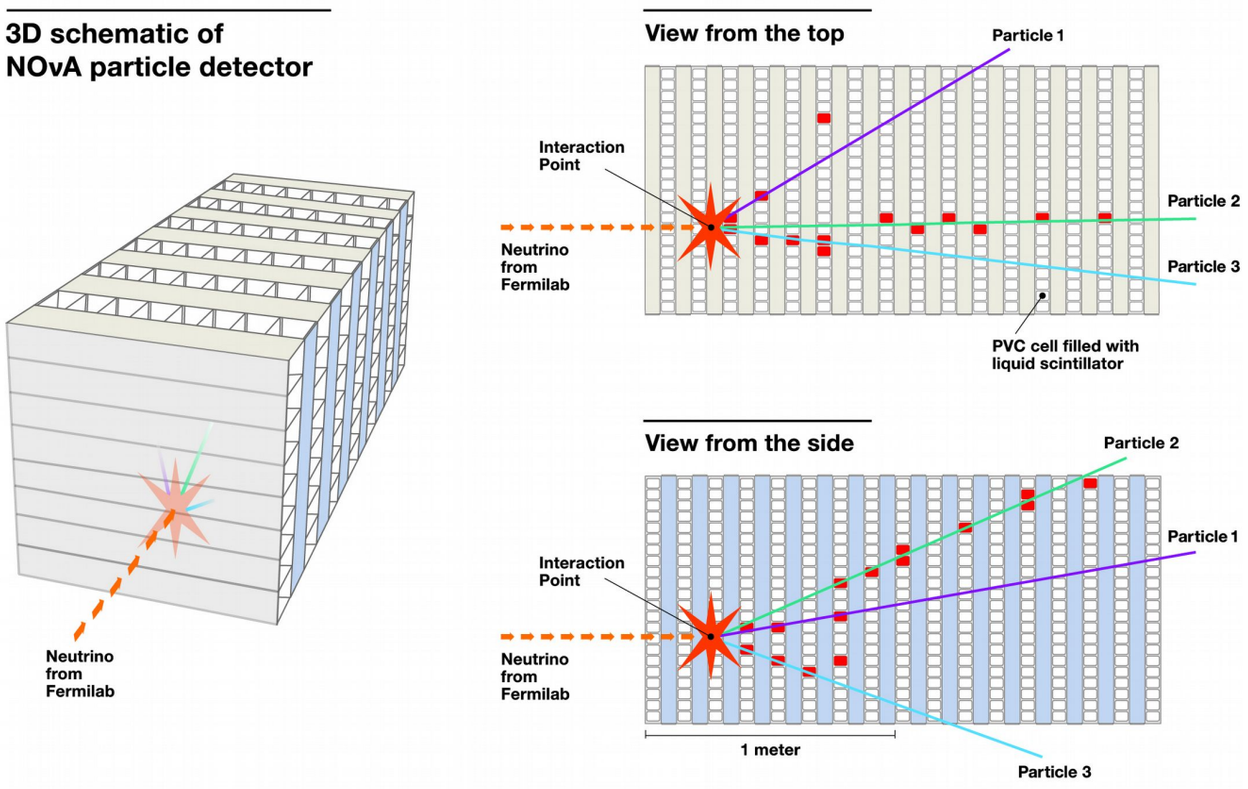
(b)



Slide from Kyle Cranmer

Neutrino Physics

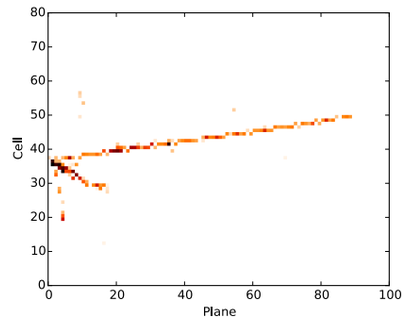
- Core Physics requires just measuring ***neutrino flavor and energy***.
- Generally clean (low multiplicity) and high granularity.
- ***First HEP CNN application: <u>Nova</u>*** using Siamese Inception CNN.



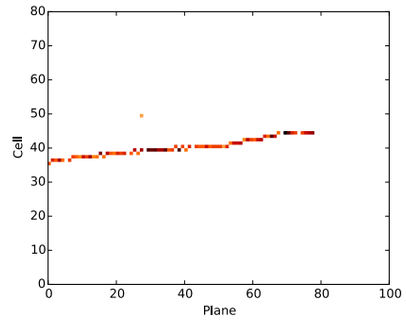
	CVN Selection Value	ν_e sig	Tot bkg	NC	ν_μ CC	Beam ν_e	Signal Efficiency	Purity
Contained Events	–	88.4	509.0	344.8	132.1	32.1	–	14.8%
$s/\sqrt{b}$ opt	0.94	43.4	6.7	2.1	0.4	4.3	49.1%	86.6%
$s/\sqrt{s+b}$ opt	0.72	58.8	18.6	10.3	2.1	6.1	66.4%	76.0%

	CVN Selection Value	ν_μ sig	Tot bkg	NC	Appeared ν_e	Beam ν_e	Signal Efficiency	Purity
Contained Events	–	355.5	1269.8	1099.7	135.7	34.4	–	21.9%
$s/\sqrt{b}$ opt	0.99	61.8	0.1	0.1	0.0	0.0	17.4%	99.9%
$s/\sqrt{s+b}$ opt	0.45	206.8	7.6	6.8	0.7	0.1	58.2%	96.4%

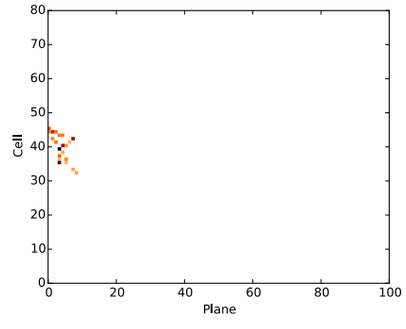
- ***40% Better Electron Efficiency for same background.***



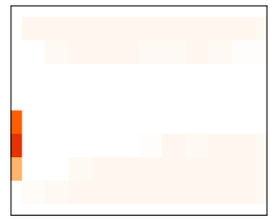
Muon Neutrino
DIS CC



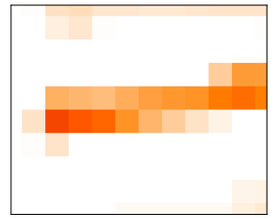
Muon Neutrino
QE CC



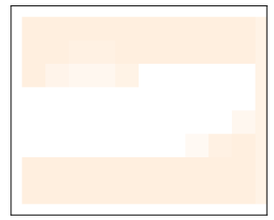
Muon Neutrino
NC

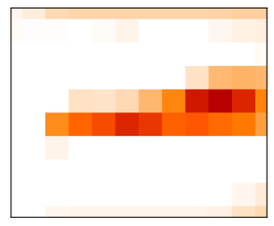


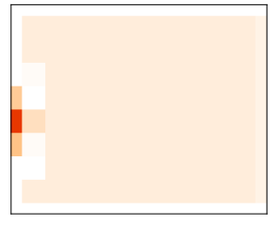
Hadronic
Feature
Map

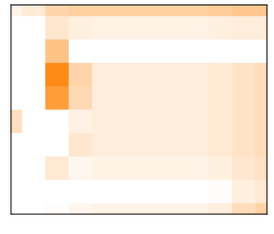


Muon
Feature
Map









http://arxiv.org/pdf/1604.01444.pdf

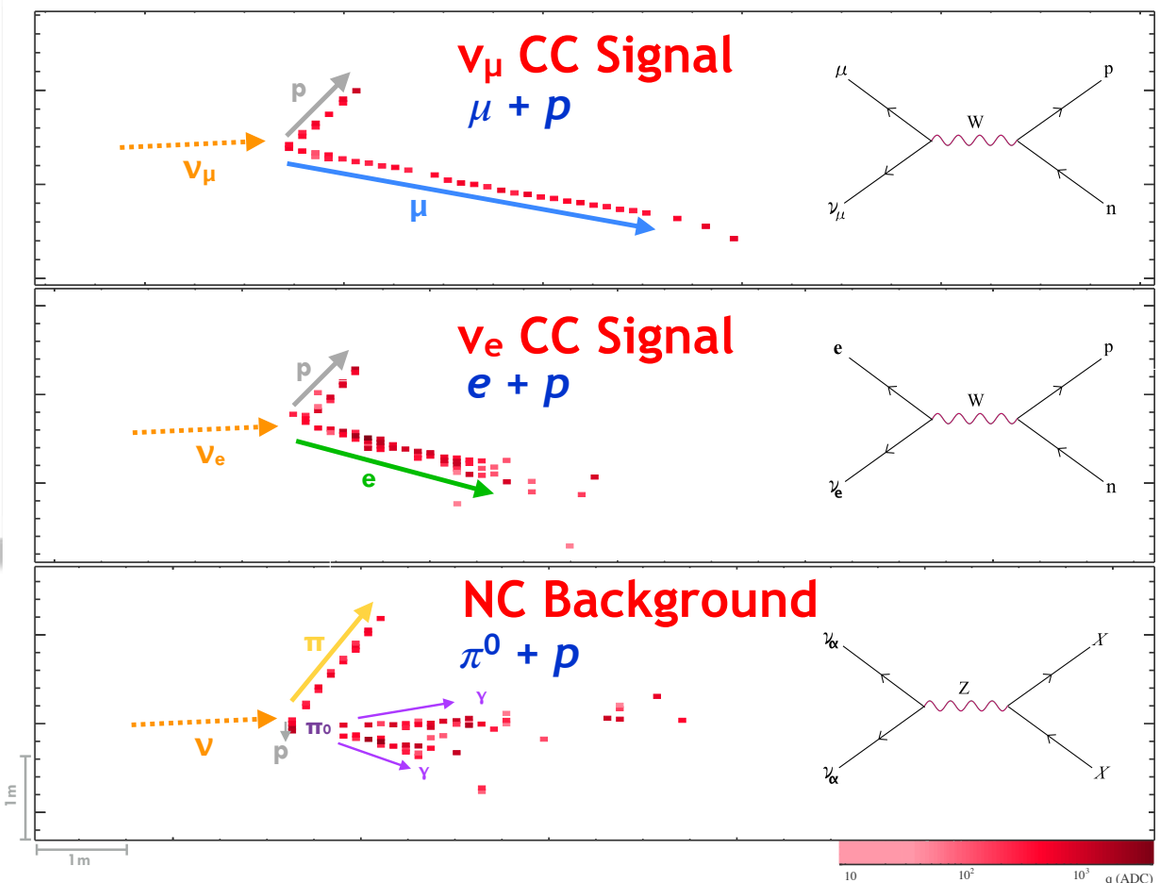
Convolutional Neural Networks for Neutrino Experiments

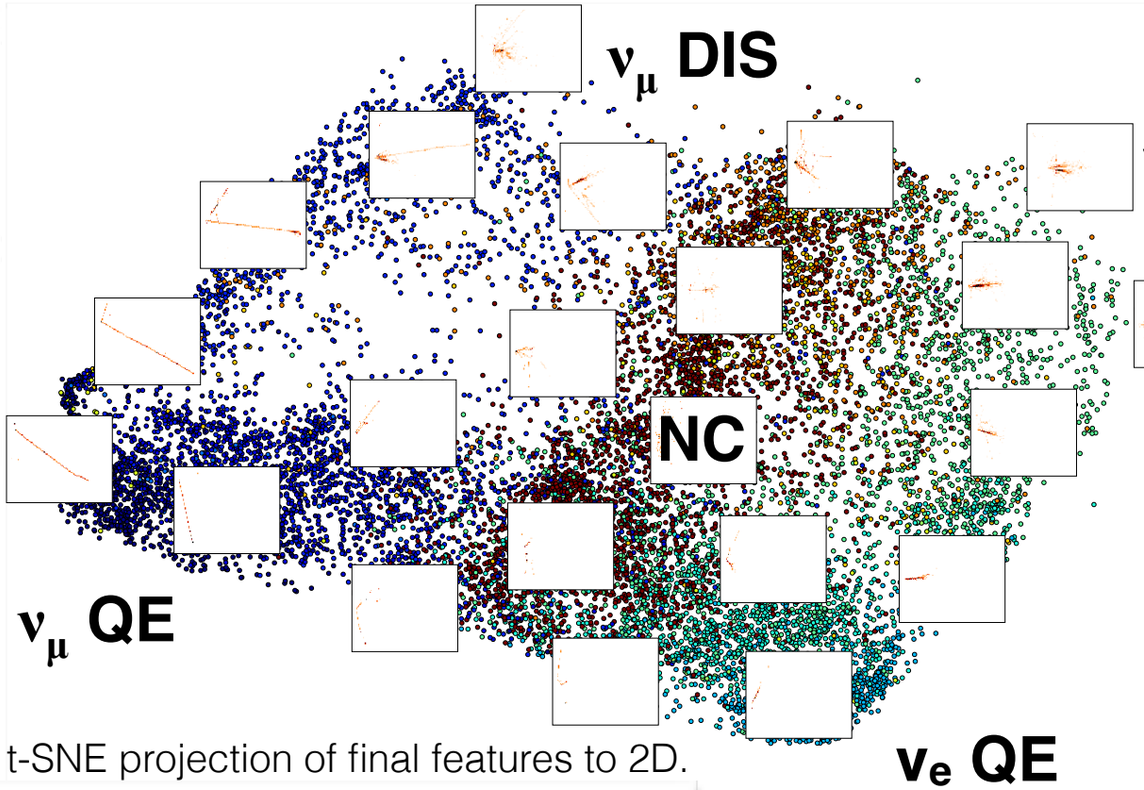
Alexander Radovic
College of William and Mary

Why Convolutional Neural Networks?

- That means that any oscillation analysis can benefit fror precise identification of the interaction in two ways:
  - Estimating the lepton flavor of the incoming neutrino.
  - Correctly identifying the type of neutrino interaction, t better estimate the neutrino energy, aka is it a quasi elastic event or a resonance event?

- Our detectors are also often the perfect domain:
  - Large ~uniform volumes where spatially invariant response is a benefit.
  - Usually only one or two detector systems.





t-SNE projection of final features to 2D.

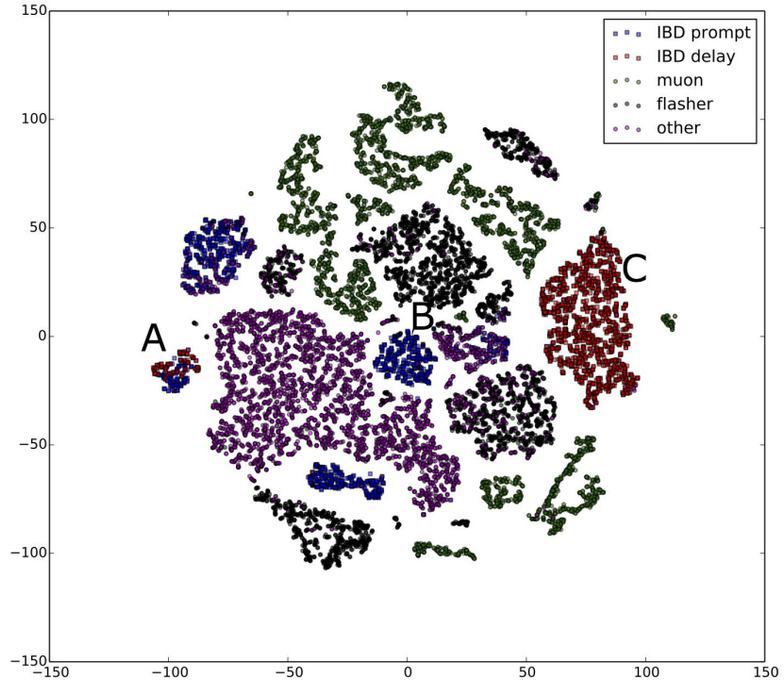
However our CNN achieves **73%** efficiency and **76%** purity on ν_e selection at the $s/\sqrt{s+b}$ optimized cut.
Equivalent to **30%** more exposure with the old PIDs.

Other interesting problems…

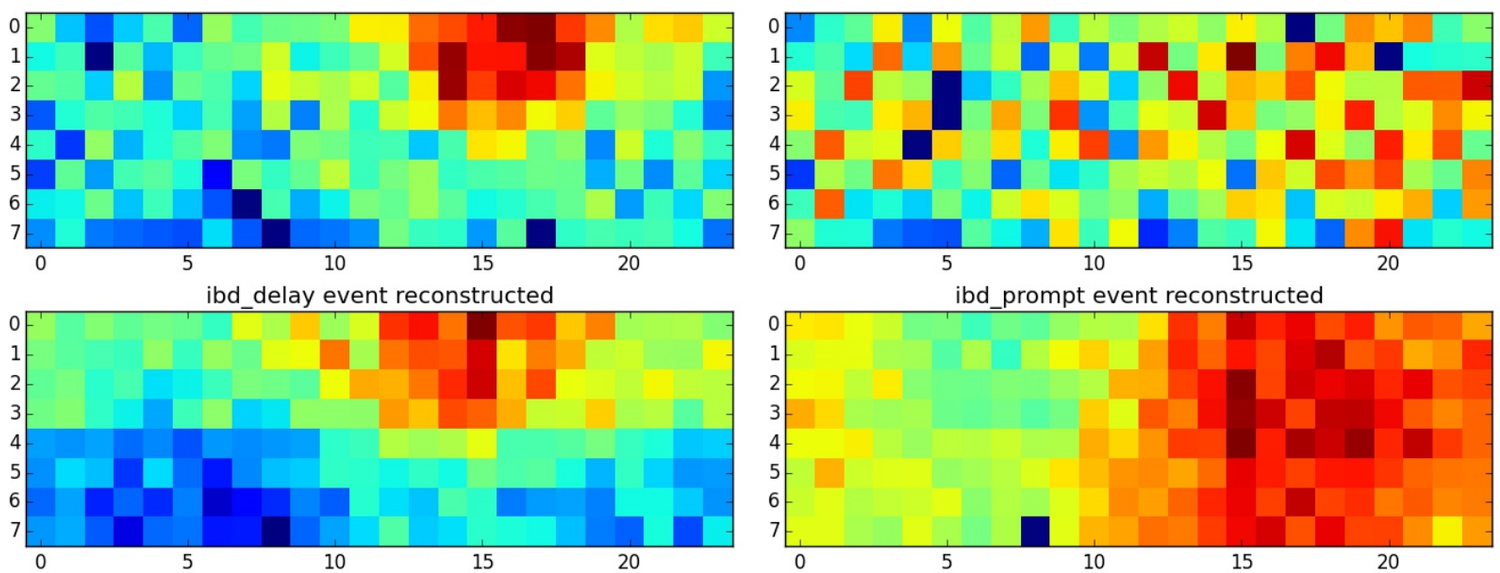
- Matrix Element Method vs DNNs.
  - MEM is supposed to lead to most sensitive measurements.
  - But DNNs can reproduce MEM, and avoid simplifications in MEM by learning on full simulation.
- Fast Next to Leading Order (NLO) and NNLO…
  - LHC/HL-HLC require ever improving theoretical calculations,
    - Essentially the probability of seeing a set of particles with specific kinematics.
    - Calculations become exponentially more computationally demanding as we go to higher orders in perturbation theory.
  - FastNLO technique performs the computation once (function from high dimensional space to a number) and projects down to 3D histogram.
  - DNNs could instead learn the high dimensional function.
- We can imagine perhaps learning soft QCD (~ pythia) from data…
- Simplify PDF systematic error evaluation…

Learning Representations

- Example: ***Daya Bay Experiment*** (*Evan Racah, et al*)
- Input: 8 x 24 PMT unrolled cylinder. ***Real Data (no simulation)***
- 2 Studies:
  - ***Supervised CNN Classifier***
    - Labels from standard analysis: Prompt/Delayed Inverse Beta Decay, Muon, Flasher, Other.
  - ***Convolutional Auto-encoder*** (semi-supervised)
    - Clearly separates muon and IBD delay ***without any physics knowledge***.
    - Potentially could have ID'ed problematic data (e.g. flashers) much earlier.

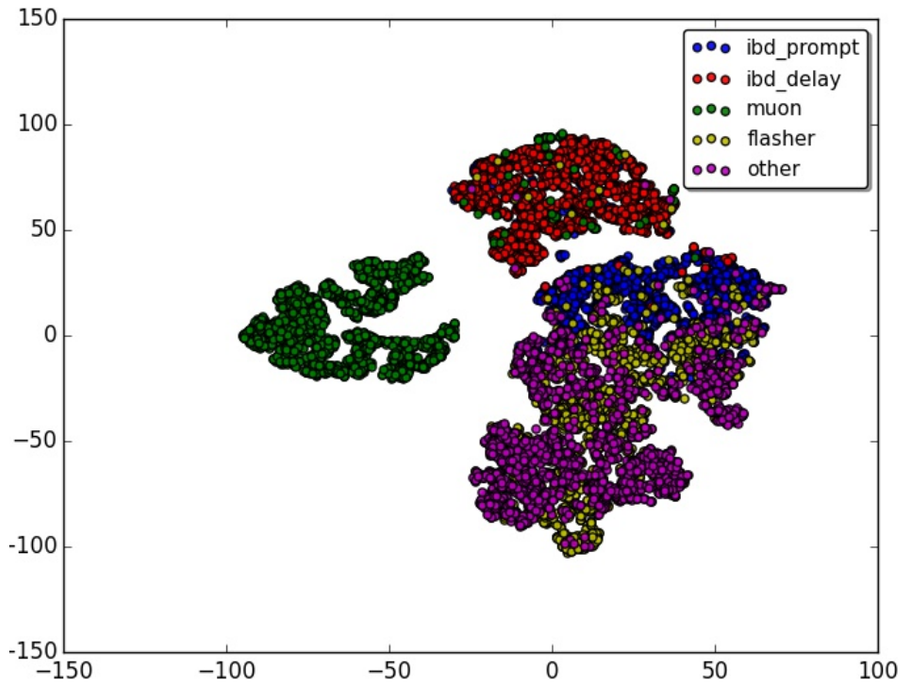


t-SNE reduction of 26-dim representation of the last fully connected layer.



(a) Example of an "IBD delay" event

(b) Example of an "IBD prompt" event

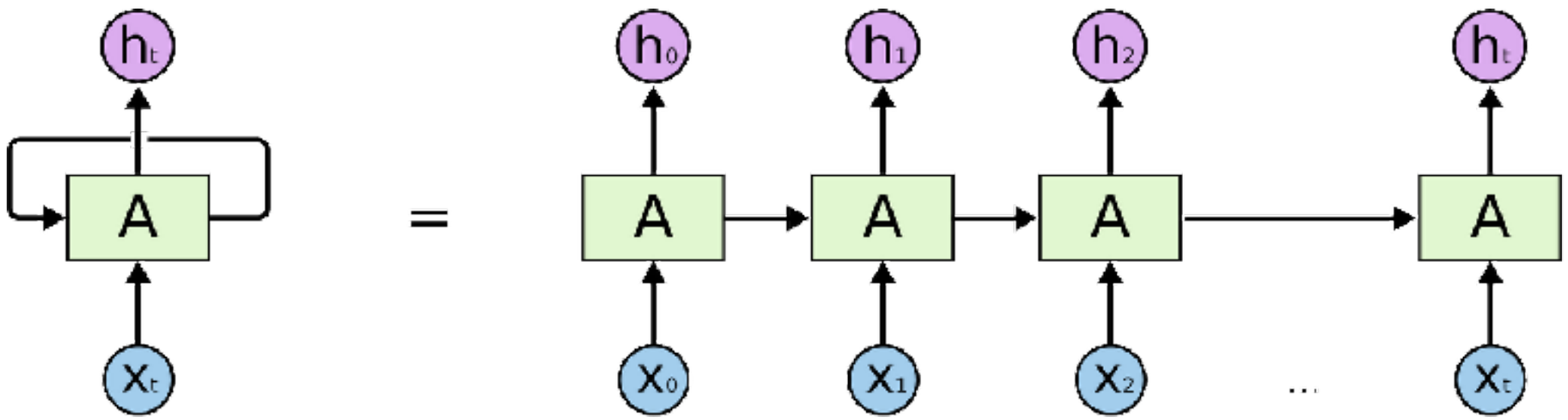


t-SNE reduction of 10 parameter latent representation.

Recurrent DNNs

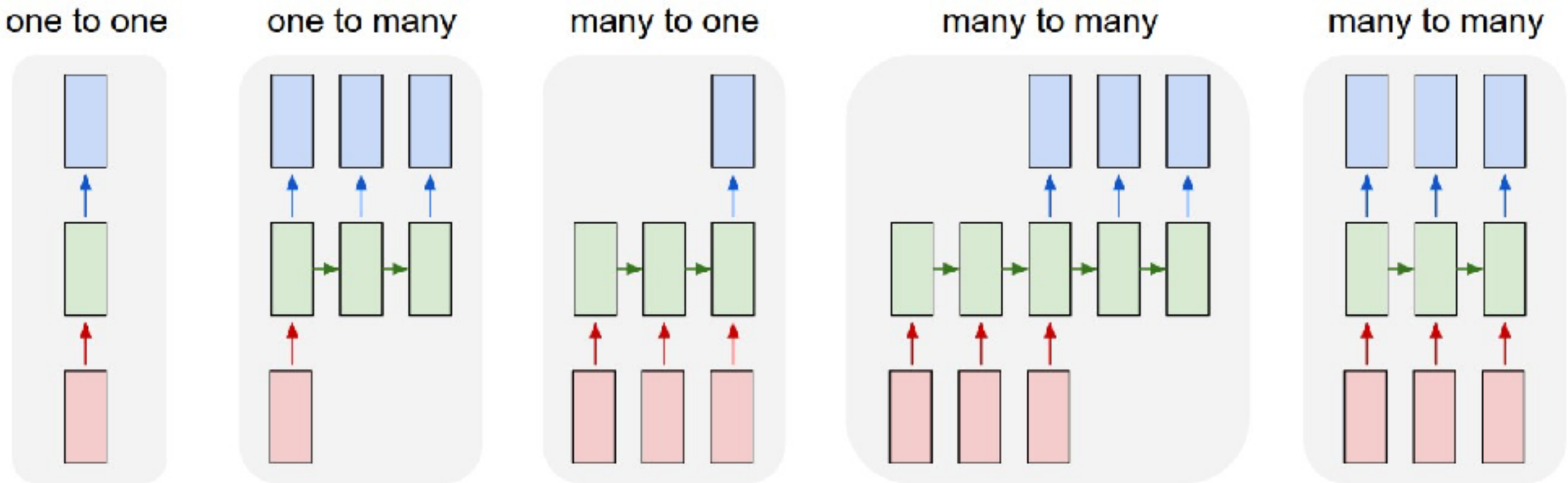
Motivation

- DNN inputs:
  - Fixed size: images, video, raw data from detector…
  - Variable size: audio, text, particles in event… sequences.
- Usual NNs map input to output.
  - No memory of previous information.
- Recurrent NN: feed some output back into self.
- DNNs can represent arbitrary functions… RNNs can represent arbitrary programs.



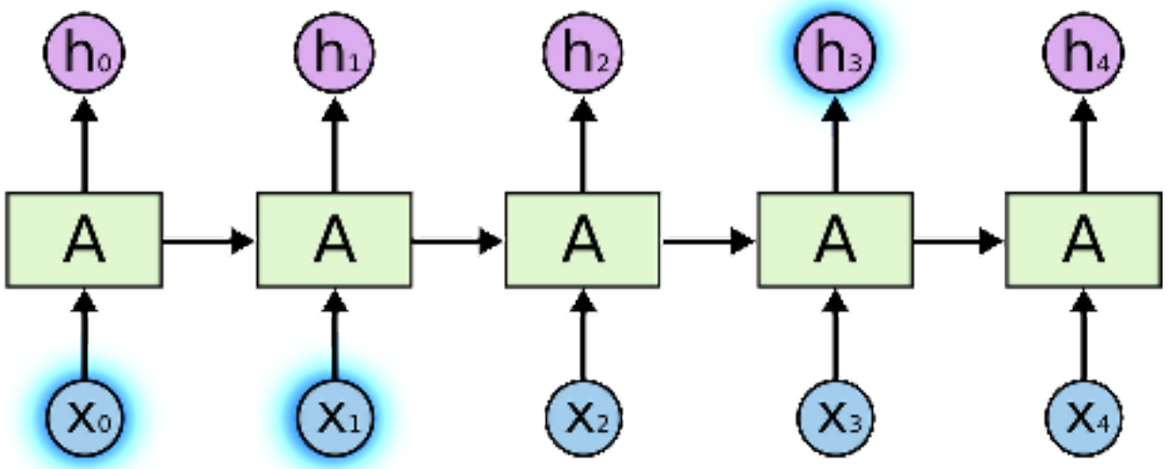
http://colah.github.io/posts/2015-08-Understanding-LSTMs/

RNN input/output



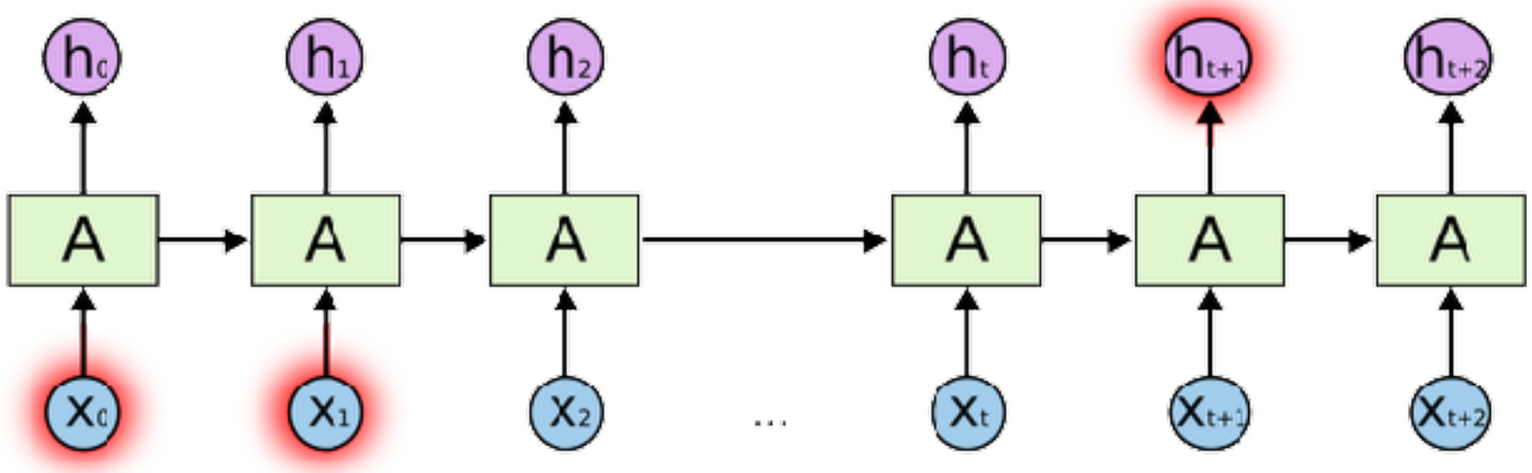
http://karpathy.github.io/2015/05/21/rnn-effectiveness/

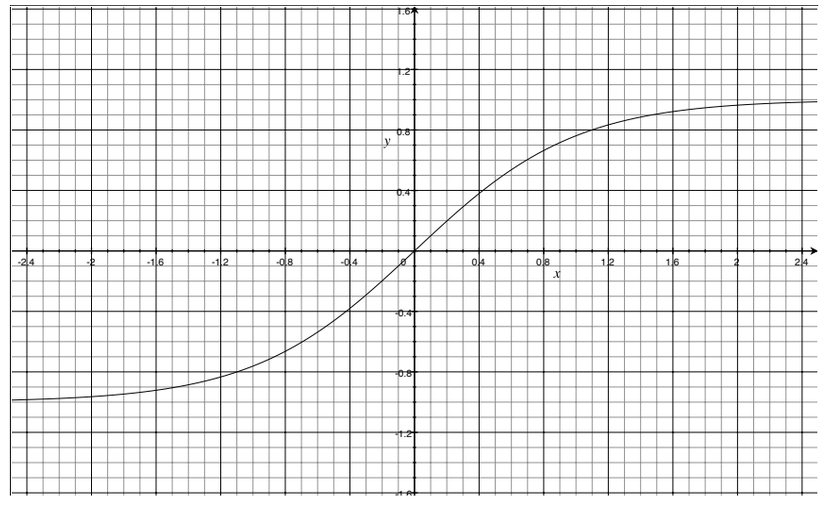
Basic RNN

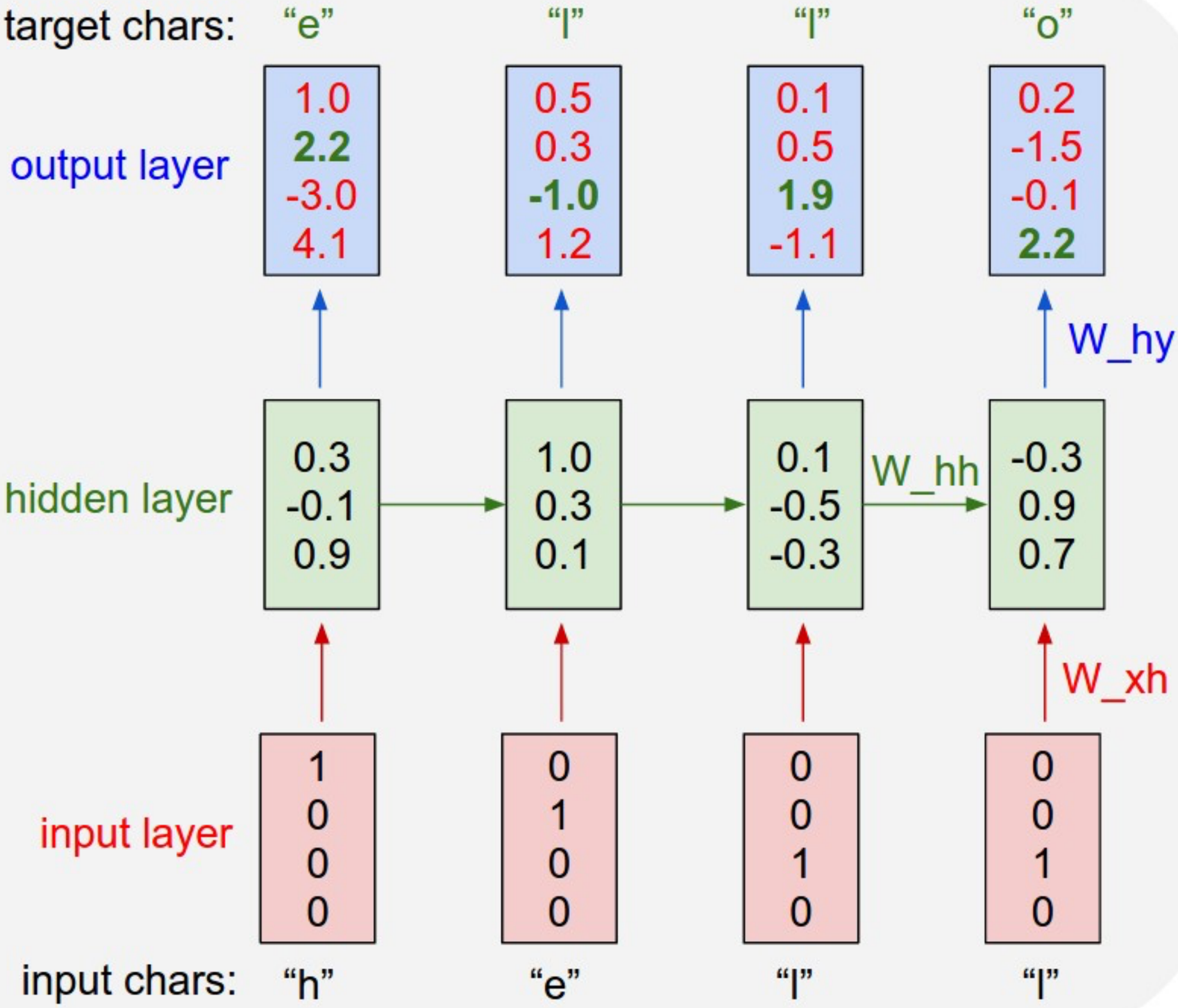


$$h_t = \tanh(W_{hh} h_{t-1} + W_{xh} x_t)$$

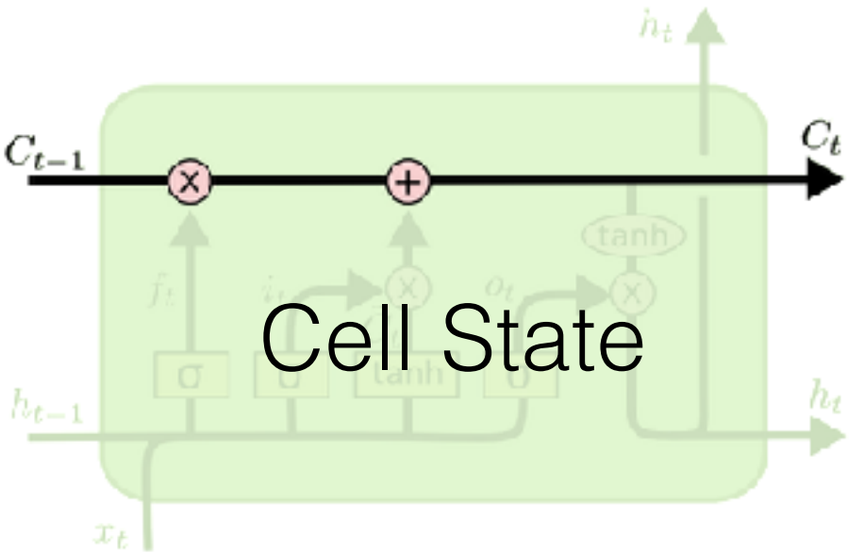
$$y = W_{hy} \cdot h_t$$



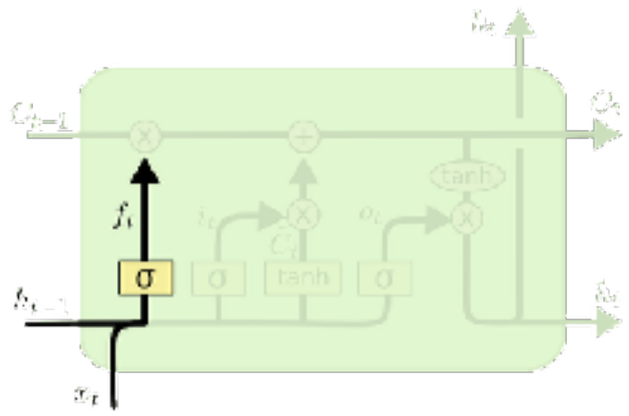



target chars:
"e"
"l"
"l"
"o"
output layer
1.0
2.2
-3.0
4.1
0.5
0.3
-1.0
1.2
0.1
0.5
1.9
-1.1
0.2
-1.5
-0.1
2.2
W_hy
hidden layer
0.3
-0.1
0.9
1.0
0.3
0.1
0.1
-0.5
-0.3
W_hh
-0.3
0.9
0.7
W_xh
input layer
1
0
0
0
0
1
0
0
0
0
1
0
0
0
1
0
input chars:
"h"
"e"
"l"
"l"

LSTM

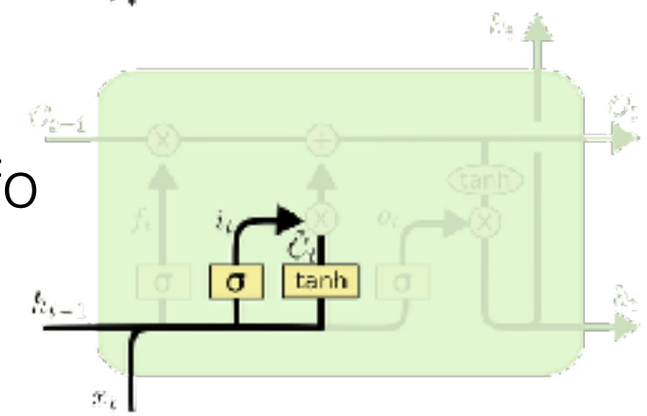


Forget



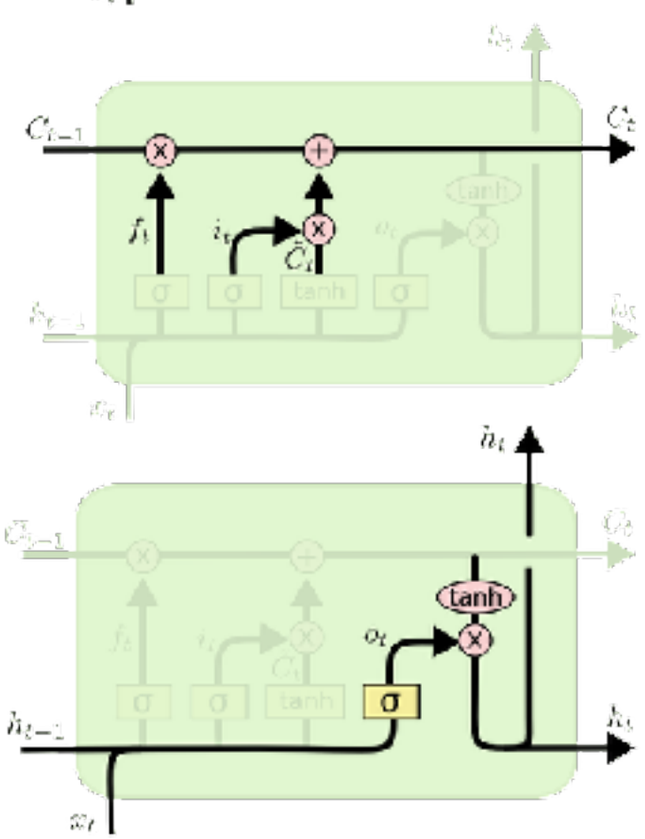
$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right)$$

New info



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] + b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

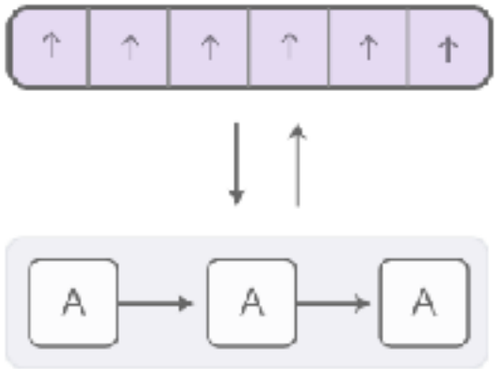
Construct output



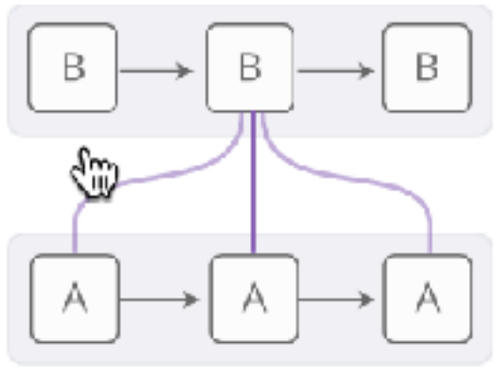
$$C_t = f_t * C_{t-1} - i_t * \tilde{C}_t$$

$$o_t = \sigma\left(W_o \left[h_{t-1}, x_t\right] + b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

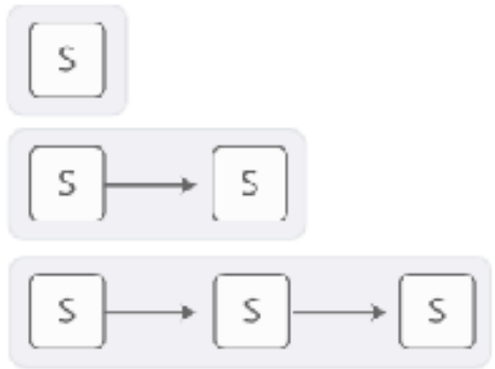
Other RNNs



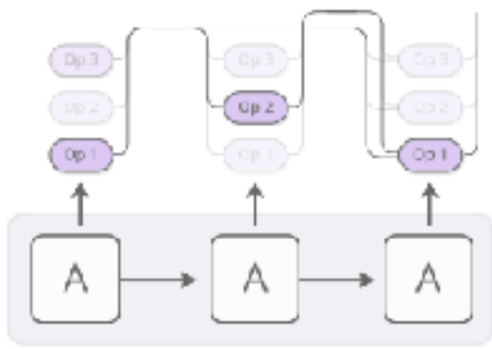
Neural Turing Machines
have external memory that they can read and write to.



Attentional Interfaces
allow RNNs to focus on parts of their input.



Adaptive Computation Time
allows for varying amounts of computation per step.

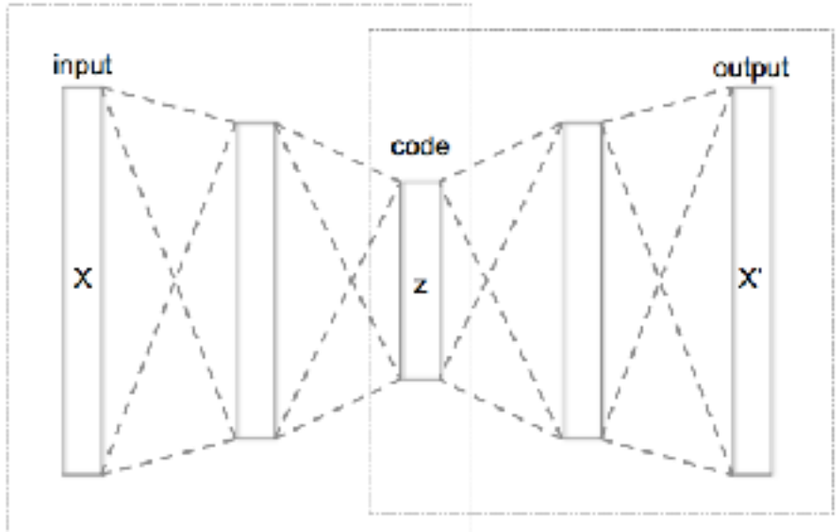


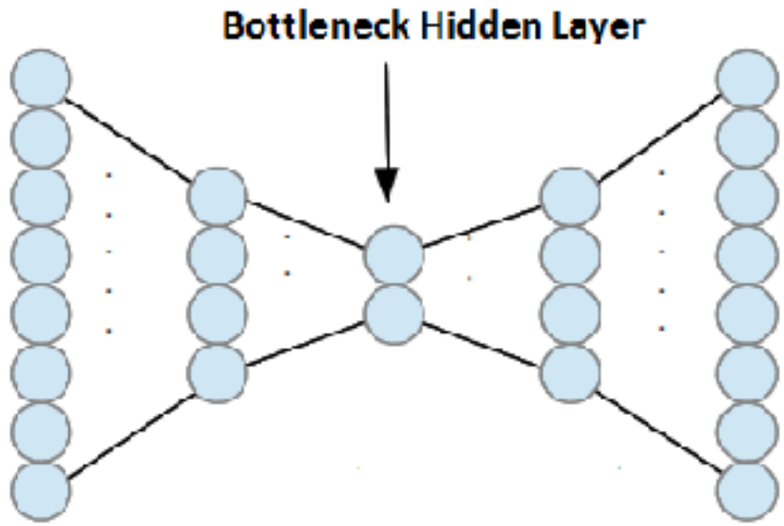
Neural Programmers
can call functions, building programs as they run.

Unsupervised Learning

Semi-supervised Learning

- Basic idea: Train network to ***reproduce the input***.
- Example: ***Auto-encoders***
  - ***De-noising auto-encoders***: add noise to input only.
  - ***Sparse auto-encoders***:
    - ***Sparse latent (code) representation*** can be exploited for ***Compression, Clustering, Similarity testing,*** …
    - ***Anomaly Detection***
      - Reconstruction Error
      - Outliers in latent space
    - ***Transfer Learning***
      - Small labeled training sample?
        - Train auto-encoder on large unlabeled dataset (e.g. data).
        - Train in latent space on small labeled data. (e.g. rare signal MC).
- Easily think of a dozen applications.





(a) Azimuth (pose)

(b) Elevation

(c) Lighting

(d) Wide or Narrow

Figure 3: **Manipulating latent codes on 3D Faces:** We show the effect of the learned continuous latent factors on the outputs as their values vary from -1 to 1. In (a), we show that one of the continuous latent codes consistently captures the azimuth of the face across different shapes; in (b), the continuous code captures elevation; in (c), the continuous code captures the orientation of lighting; and finally in (d), the continuous code learns to interpolate between wide and narrow faces while preserving other visual features. For each factor, we present the representation that most resembles prior supervised results [7] out of 5 random runs to provide direct comparison.

https://arxiv.org/pdf/1606.03657.pdf

Generative Models

Auto-Encoding Variational Bayes

Diederik P. Kingma
Machine Learning Group
Universiteit van Amsterdam
dpkingma@gmail.com

Max Welling
Machine Learning Group
Universiteit van Amsterdam
welling.max@gmail.com

Abstract

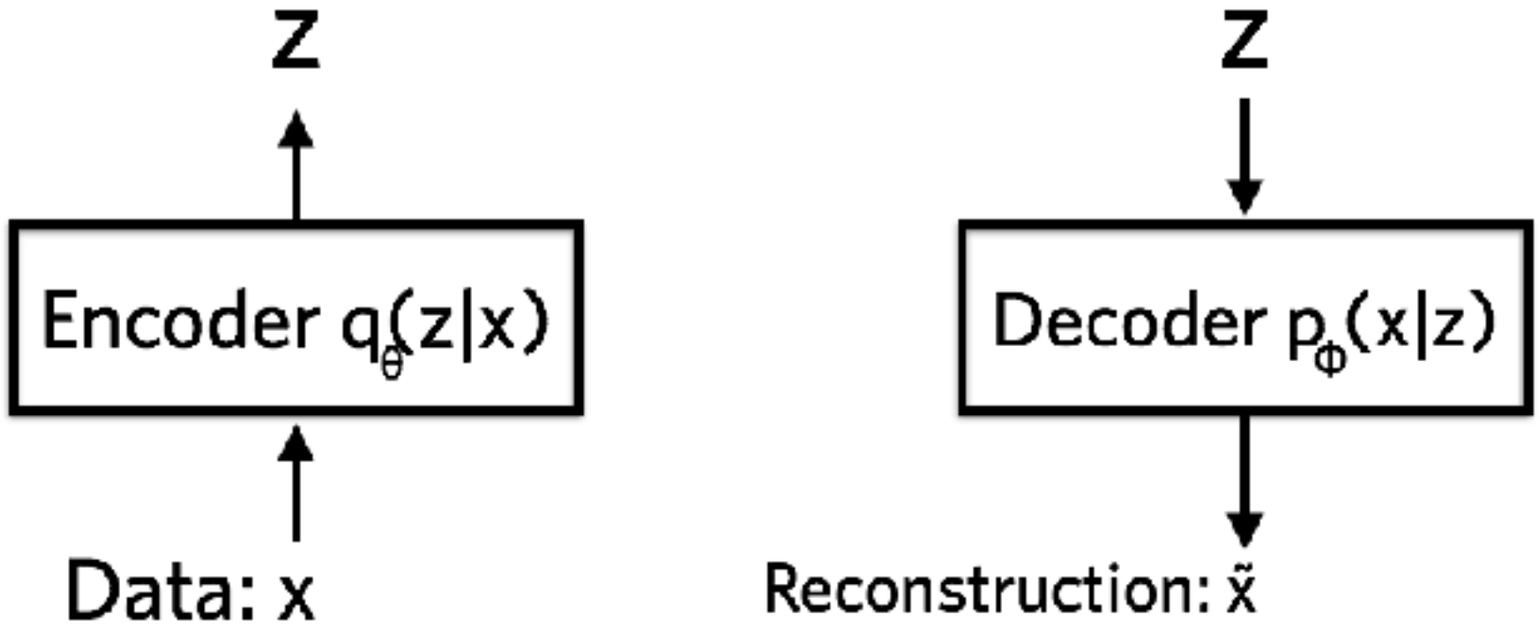
How can we perform efficient inference and learning in directed probabilistic models, in the presence of continuous latent variables with intractable posterior distributions, and large datasets? We introduce a stochastic variational inference and learning algorithm that scales to large datasets and, under some mild differentiability conditions, even works in the intractable case. Our contributions is two-fold. First, we show that a reparameterization of the variational lower bound yields a lower bound estimator that can be straightforwardly optimized using standard stochastic gradient methods. Second, we show that for i.i.d. datasets with continuous latent variables per datapoint, posterior inference can be made especially efficient by fitting an approximate inference model (also called a recognition model) to the intractable posterior using the proposed lower bound estimator. Theoretical advantages are reflected in experimental results.

https://arxiv.org/abs/1312.6114

Stochastic Backpropagation and Approximate Inference in Deep Generative Models

Danilo J. Rezende, Shakir Mohamed, Daan Wierstra
{danilor, shakir, daanw}@google.com
Google DeepMind, London

https://arxiv.org/abs/1401.4082



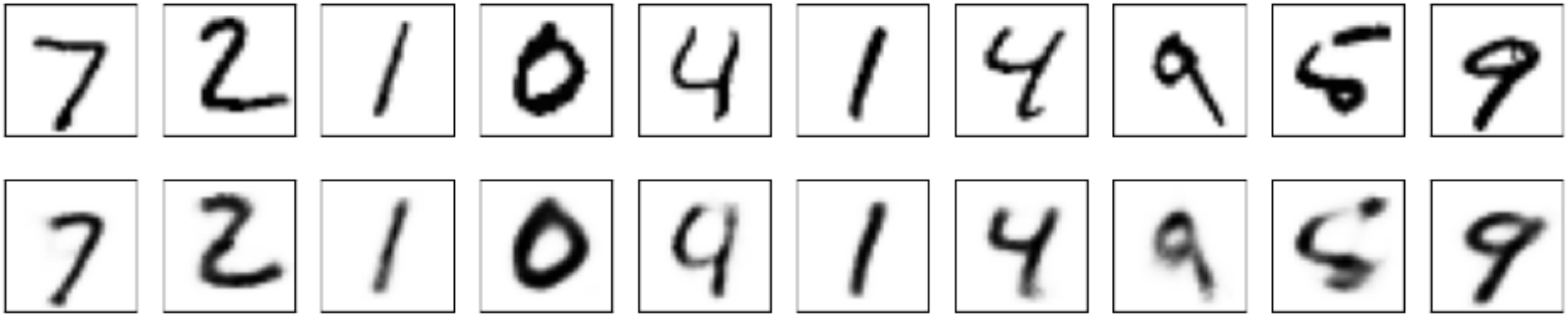
- a probabilistic encoder $q_\phi(z|x)$, approximating the true (but intractable) posterior distribution $p(z|x)$, and
- a generative decoder $p_\theta(x|z)$, which notably does not rely on any particular input x.

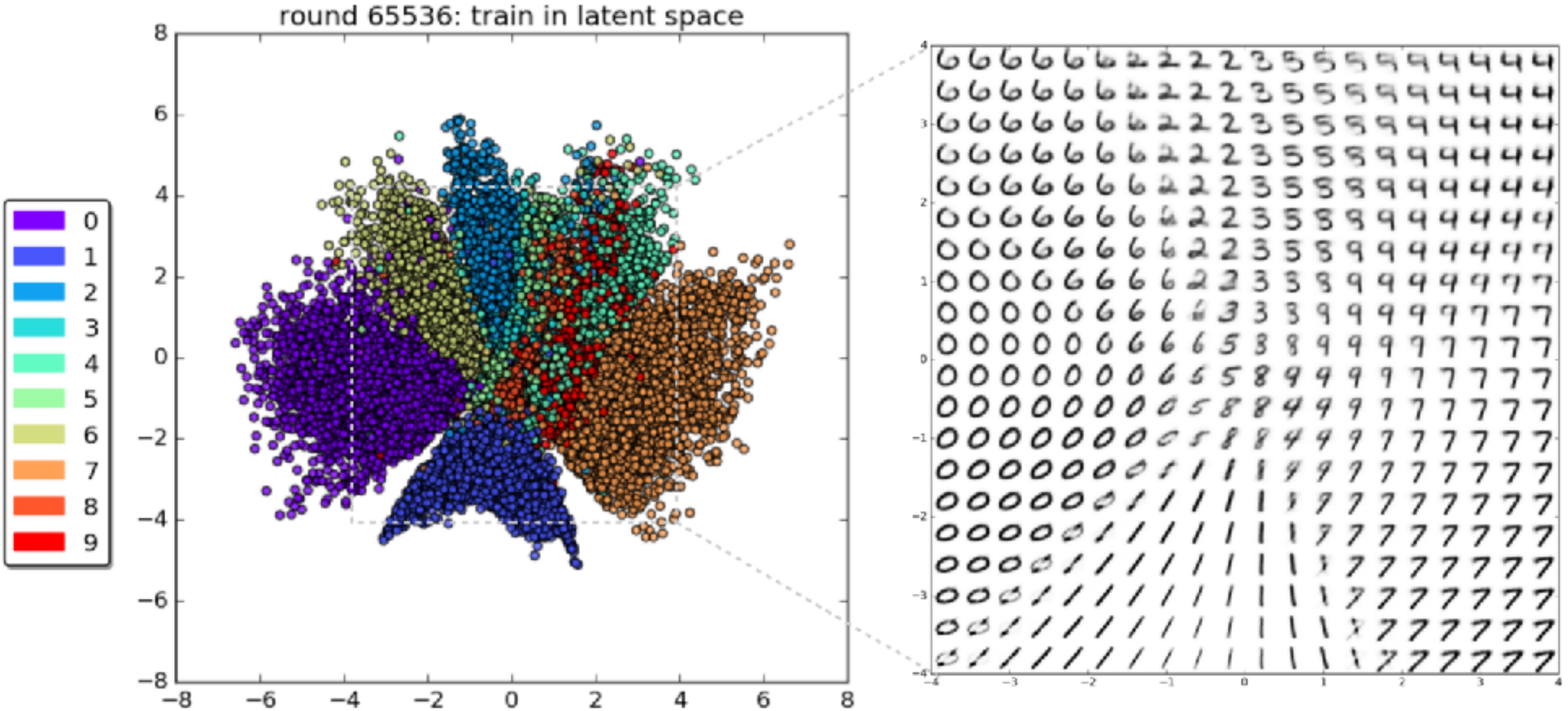
Both the encoder and decoder are artificial neural networks (i.e. hierarchical, highly nonlinear functions) with tunable parameters ϕ and θ, respectively.

Learning these conditional distributions is facilitated by enforcing a plausible mathematically-convenient prior over the latent variables, generally a standard spherical Gaussian: $z \sim \mathcal{N}(0, I)$.

$$l_i(\theta, \phi) = -E_{z \sim q_\theta(z|x_i)}[\log p_\phi(x_i|z)] + KL(q_\theta(z|x_i)||p(z))$$

http://blog.fastforwardlabs.com/2016/08/12/introducing-variational-autoencoders-in-prose-and.html




round 65536: train in latent space
0
1
2
3
4
5
6
7
8
9

(a) Azimuth (pose)

(b) Elevation

(c) Lighting

(d) Wide or Narrow

Figure 3: **Manipulating latent codes on 3D Faces:** We show the effect of the learned continuous latent factors on the outputs as their values vary from -1 to 1. In (a), we show that one of the continuous latent codes consistently captures the azimuth of the face across different shapes; in (b), the continuous code captures elevation; in (c), the continuous code captures the orientation of lighting; and finally in (d), the continuous code learns to interpolate between wide and narrow faces while preserving other visual features. For each factor, we present the representation that most resembles prior supervised results [7] out of 5 random runs to provide direct comparison.

https://arxiv.org/pdf/1606.03657.pdf

Generative Adversarial Nets

Ian J. Goodfellow, Jean Pouget-Abadie,* Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair,† Aaron Courville, Yoshua Bengio‡
Département d'informatique et de recherche opérationnelle
Université de Montréal
Montréal, QC H3C 3J7

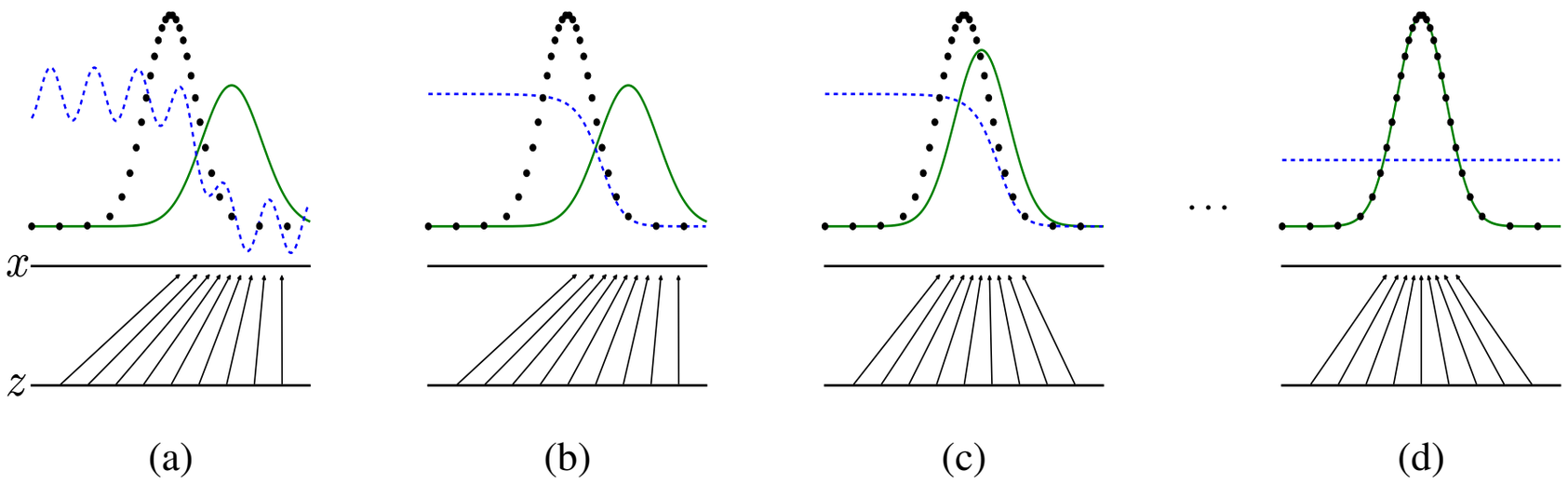
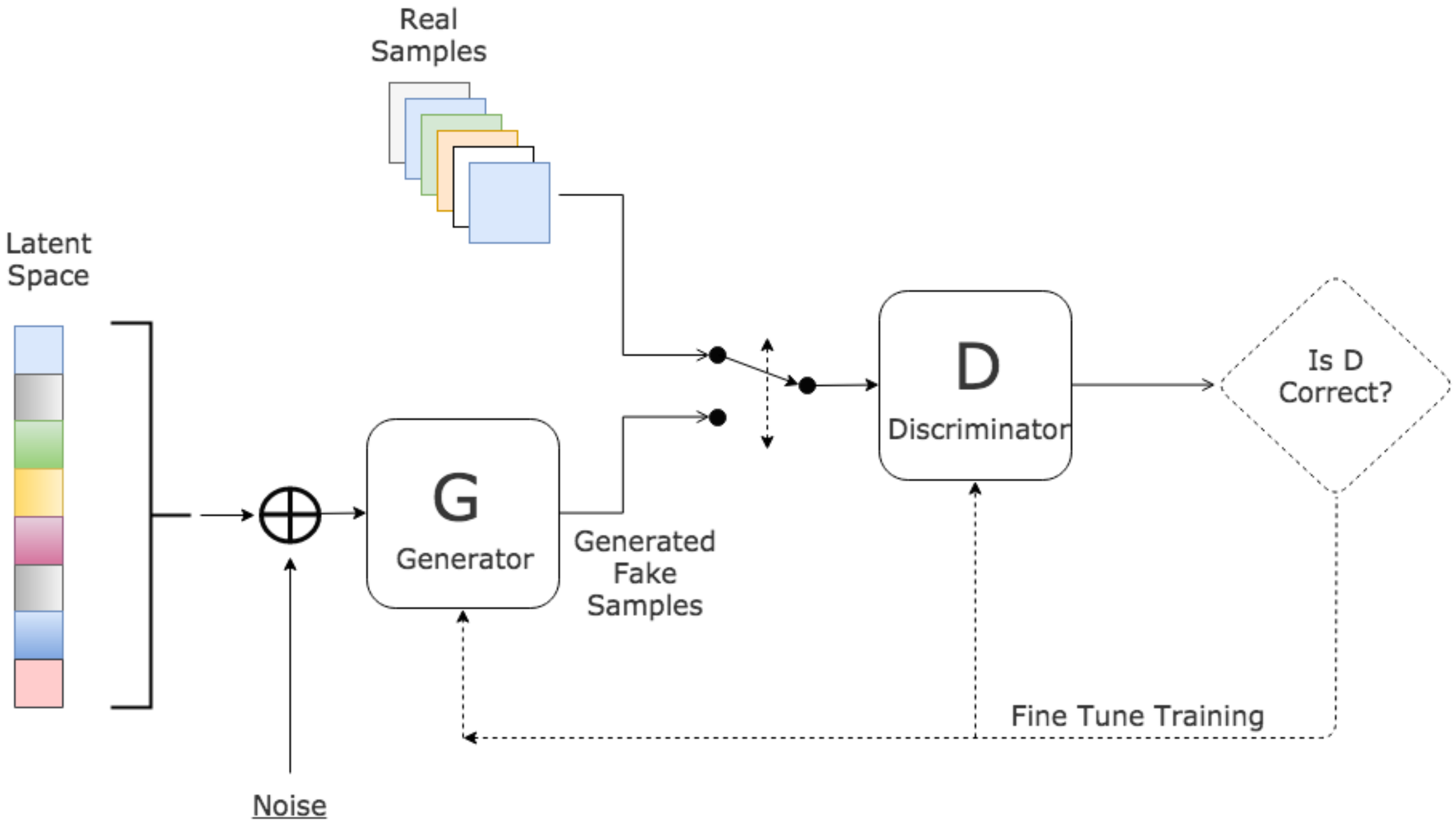


Figure 1: Generative adversarial nets are trained by simultaneously updating the **d**iscriminative distribution (D, blue, dashed line) so that it discriminates between samples from the data generating distribution (black, dotted line) $p_{\boldsymbol{x}}$ from those of the **g**enerative distribution p_g (G) (green, solid line). The lower horizontal line is the domain from which $\boldsymbol{z}$ is sampled, in this case uniformly. The horizontal line above is part of the domain of $\boldsymbol{x}$. The upward arrows show how the mapping $\boldsymbol{x} = G(\boldsymbol{z})$ imposes the non-uniform distribution p_g on transformed samples. G contracts in regions of high density and expands in regions of low density of p_g. (a) Consider an adversarial pair near convergence: p_g is similar to p_{data} and D is a partially accurate classifier. (b) In the inner loop of the algorithm D is trained to discriminate samples from data, converging to $D^*(\boldsymbol{x}) = \frac{p_{\text{data}}(\boldsymbol{x})}{p_{\text{data}}(\boldsymbol{x}) + p_g(\boldsymbol{x})}$. (c) After an update to G, gradient of D has guided $G(\boldsymbol{z})$ to flow to regions that are more likely to be classified as data. (d) After several steps of training, if G and D have enough capacity, they will reach a point at which both cannot improve because $p_g = p_{\text{data}}$. The discriminator is unable to differentiate between the two distributions, i.e. $D(\boldsymbol{x}) = \frac{1}{2}$.

https://arxiv.org/abs/1406.2661

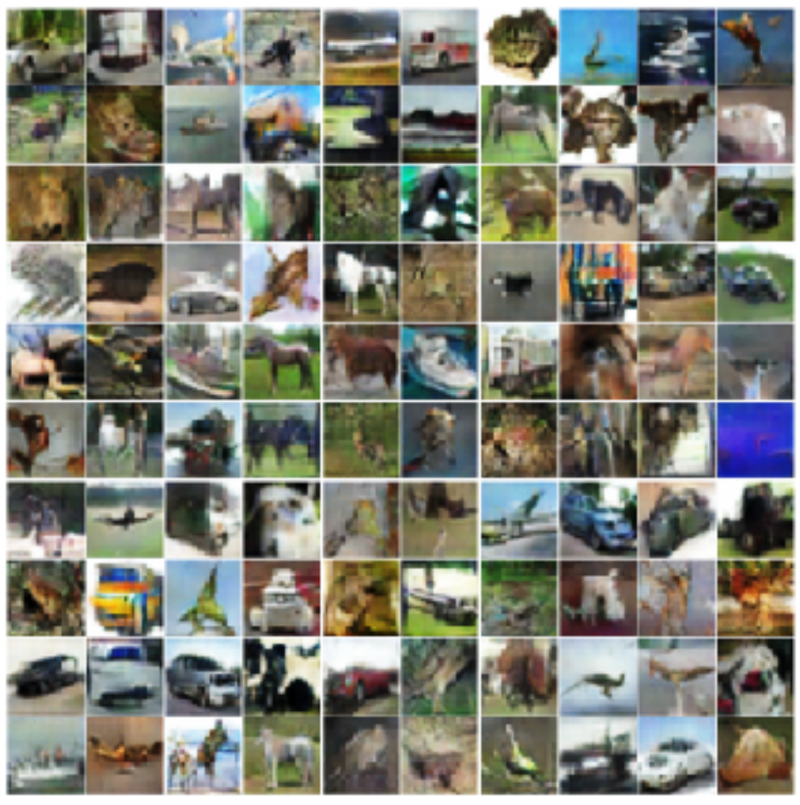
Example: http://cs.stanford.edu/people/karpathy/gan/

Generative Adversarial Network



http://www.kdnuggets.com/2017/01/generative-adversarial-networks-hot-topic-machine-learning.html

Generated bedrooms. Source: "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks" https://arxiv.org/abs/1511.06434v2



Generated CIFAR-10 samples. Source: "Improved Techniques for Training GANs" https://arxiv.org/abs/1606.03498

Generative Models

- ***Likelihood Approximation relies simulation***
  - Most ***computationally expensive*** step, so any speedup has huge impact.
  - More generally, ***simulation based on data*** would be a powerful tool.
    - For example, we can build a Hadronization model purely from data.
- DNNs Generative Models enable building simulations purely from examples.
  - ***Generative Adversarial Nets*** (Goodfellow, et. al. arxiv:1406.2661). Simultaneously train 2 Networks:
    - ***Discriminator*** (D) that tries to distinguish output and real examples
    - ***Generator*** (G) that generate the output that is difficult to distinguish
  - ***Variational Auto-encoders***:
    - Learn a ***latent variable probabilistic model*** of the input dataset.
    - ***Sample latent space*** and use ***decoder to generate data***.
- ***Particle showering*** is ***slowest*** part of the micro-physics simulation…
  - Various techniques for fast showering (e.g. shower template libraries) are common.
  - DNN Generative Models are being pursued inside the experiments (K. Cranmer, G. Louppe, …) for this task…

Learning Particle Physics by Example: Location-Aware Generative Adversarial Networks for Physics Synthesis

Luke de Oliveira[a], **Michela Pag**

[a] *Lawrence Berkeley National Lab*
[b] *Department of Physics, Yale Un*

E-mail: lukedeoliveira@lbl.

ABSTRACT: We provide a bridg
and simulated physical processe
Adversarial Network (GAN) ar
energy depositions from particle
the Location-Aware Generative A
from simulated high energy partic
span over many orders of magnit
jet mass, n-subjettiness, etc.). W
of image quality and validity of
a base for further explorations of

CaloGAN: Simulating 3D High Energy Particle Showers in Multi-Layer Electromagnetic Calorimeters with Generative Adversarial Networks

Michela Paganini[a,b], **Luke de Oliveira**[a], **and Benjamin Nachman**[a]

[a] *Lawrence Berkeley National Laboratory, 1 Cyclotron Rd, Berkeley, CA, 94720, USA*
[b] *Department of Physics, Yale University, New Haven, CT 06520, USA*

E-mail: michela.paganini@yale.edu, lukedeoliveira@lbl.gov, bnachman@cern.ch

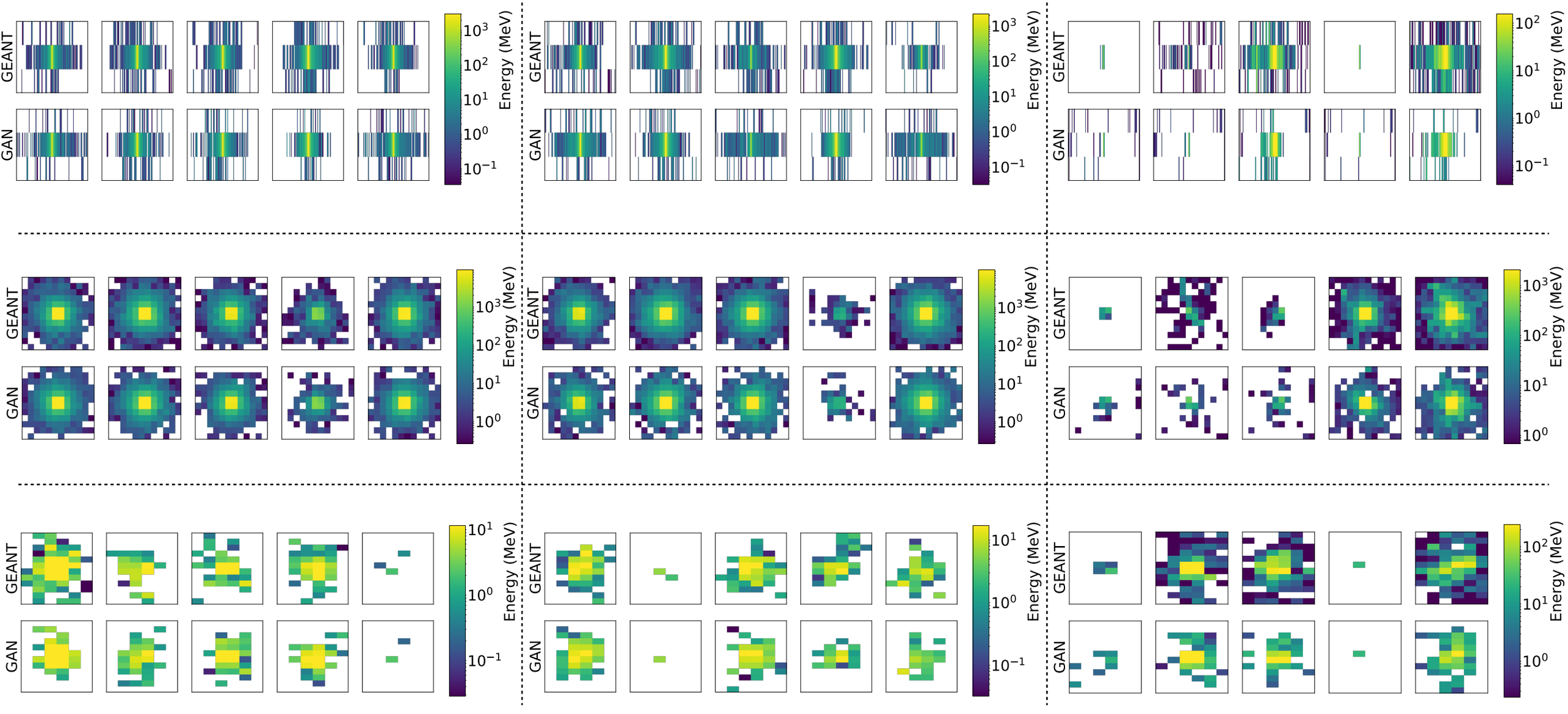
ABSTRACT: Simulation is a key component of physics analysis in particle physics and nuclear physics. The most computationally expensive simulation step is the detailed modeling of particle showers inside calorimeters. Full detector simulations are too slow to meet the growing demands resulting from large quantities of data; current fast simulations are not precise enough to serve the entire physics program. Therefore, we introduce CaloGAN, a new fast simulation based on generative adversarial neural networks (GANs). We apply the CaloGAN to model electromagnetic showers in a longitudinally segmented calorimeter. This represents a significant stepping stone toward a full neural network-based detector simulation that could save significant computing time and enable many analyses now and in the future. In particular, the CaloGAN achieves speedup factors comparable to or better than existing fast simulation techniques on CPU (100×-1000×) and even faster on GPU (up to $\sim 10^5\times$)) and has the capability of faithfully reproducing many aspects of key shower shape variables for a variety of particle types.

https://arxiv.org/pdf/1701.05927.pdf
https://arxiv.org/pdf/1705.02355.pdf

Qualitative Performance (2)

Yale

e^+ γ π^+



Timing

M. Paganini et al., 1705.02355

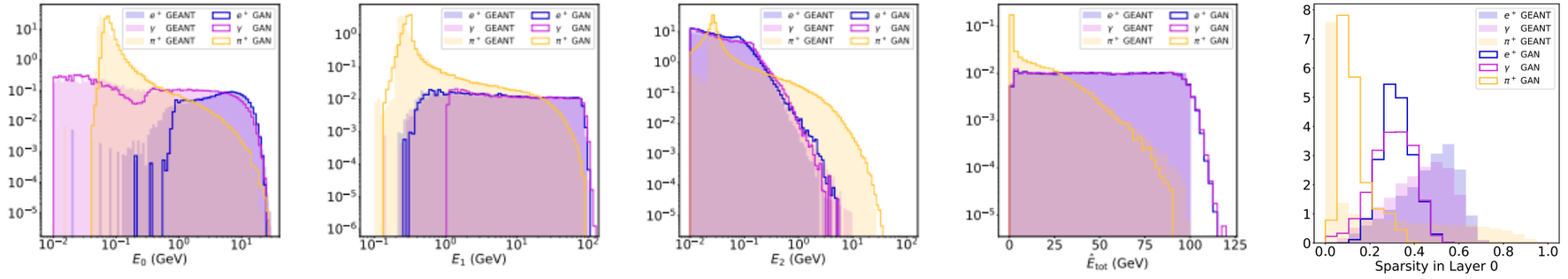
Generation Method	Hardware	Batch Size	milliseconds/shower
GEANT4	CPU	N/A	1772 ←
CALOGAN	CPU	1	13.1
		10	5.11
		128	2.19
		1024	2.03
	GPU	1	14.5
		4	3.68
		128	0.021
		512	0.014
		1024	0.012 ←

See also S. Vallecorsa et al. (GeantV), C. Guthrie et al. (NYU), W. Wei et al. (LCD dataset group), D. Salamani et al. (Geneva), D. Rousseau et al. (Orsay), L. de Oliveira et al. (Berkeley)

Shower Shapes

Yale

Check: does the LAGAN recover the true data distribution as projected onto a set of meaningful 1D manifolds?



Other Useful Techniques

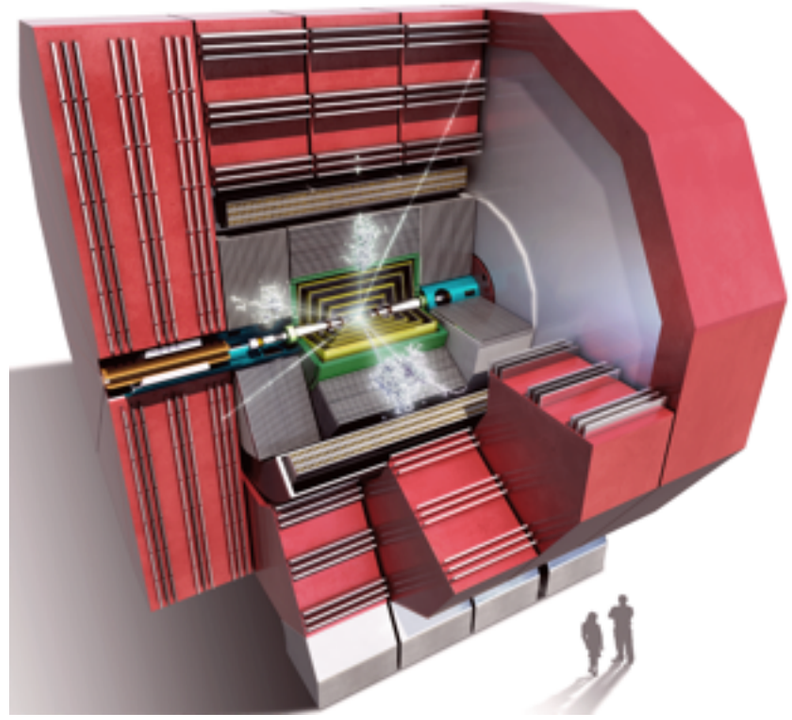
- Parameterized Classifiers
- Adversarial Techniques
  - Domain Adaptation
  - Pivot

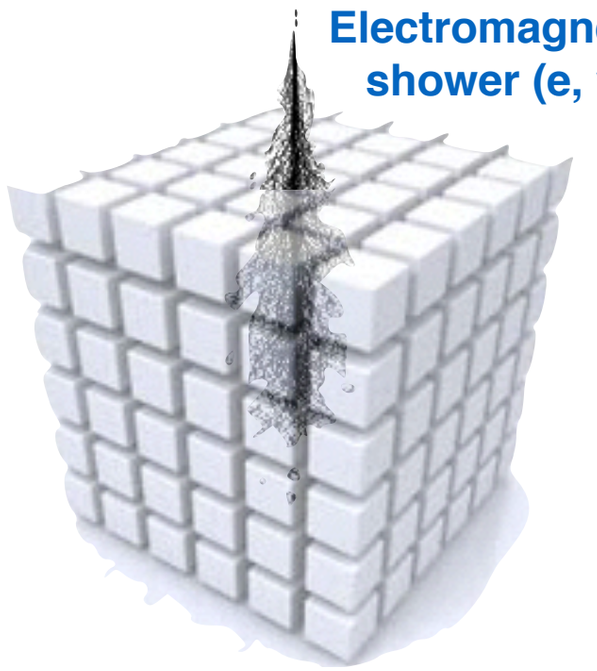
Special Topics

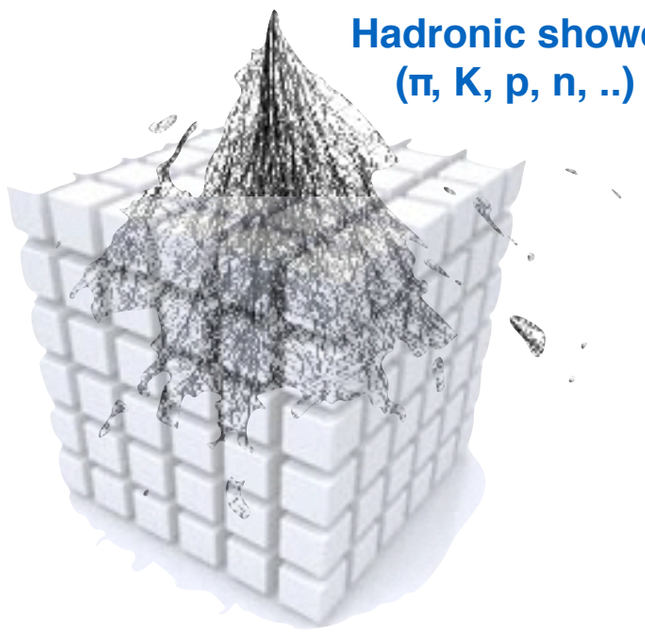
Calorimetry with Deep Learning

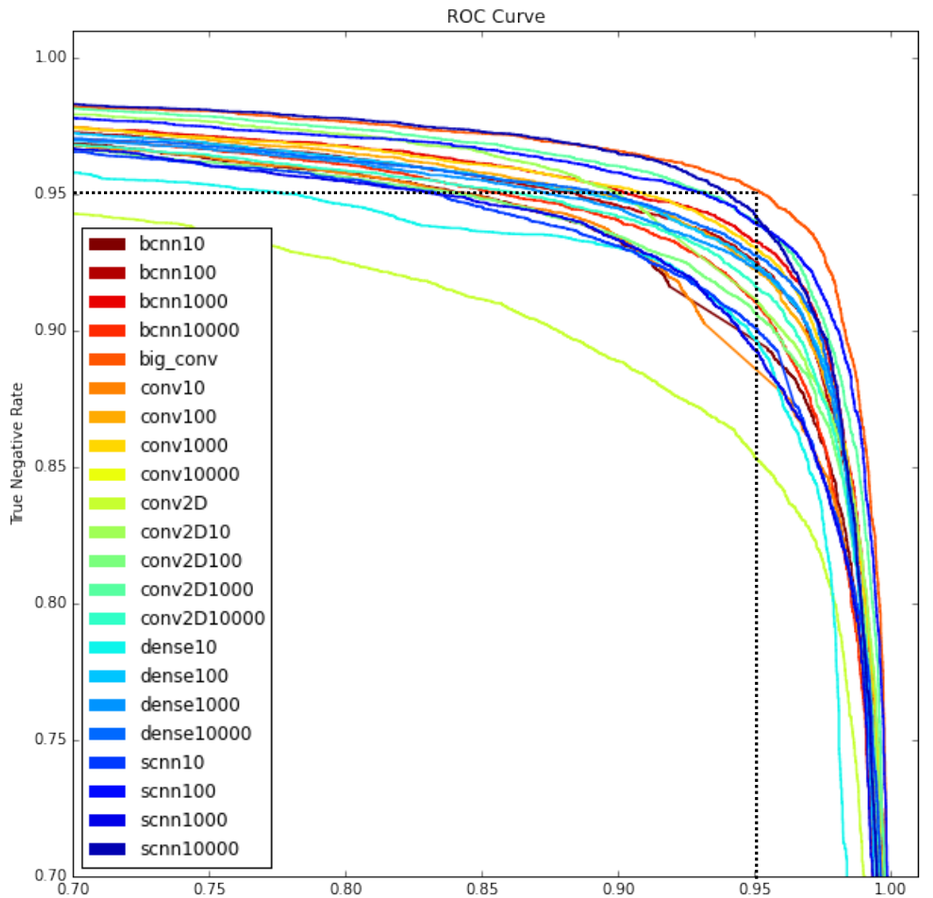
Calorimeter Dataset

- CLIC is a proposed CERN project for a linear accelerator of electrons and positrons to TeV energies (~ LHC for protons)
    - LCD is a detector concept.
    - Not a real experiment yet, so we could simulate data and make it public.
- The LCD calorimeter is an array of absorber material and silicon sensors comprising the most granular calorimeter design available
    - Data is essentially a 3D image
- With at effective eta/phi resolution of 0.003x0.003, we can down sample to get ~ ATLAS granularity: 0.025x0.1 (pre-sampler) to 0.2x0.1 Tile D.
- ***Data:*** 1 million single e, γ, $\pi^{+/-}$, π^0. 10-500 GeV of energy.









Calorimetry with Deep Learning: Particle Classification, Energy Regression, and Simulation for High-Energy Physics

Federico Carminati, Gulrukh Khattak, Maurizio Pierini
CERN

Amir Farbin
Univ. of Texas Arlington

Benjamin Hooberman, Wei Wei, and Matt Zhang
Univ. of Illinois at Urbana-Champaign

Vitória Barin Pacela
Univ. of Helsinki

Sofia Vallecorsafac
Gangneung-Wonju National Univ.

Maria Spiropulu and Jean-Roch Vlimant
California Institute of Technology

Abstract

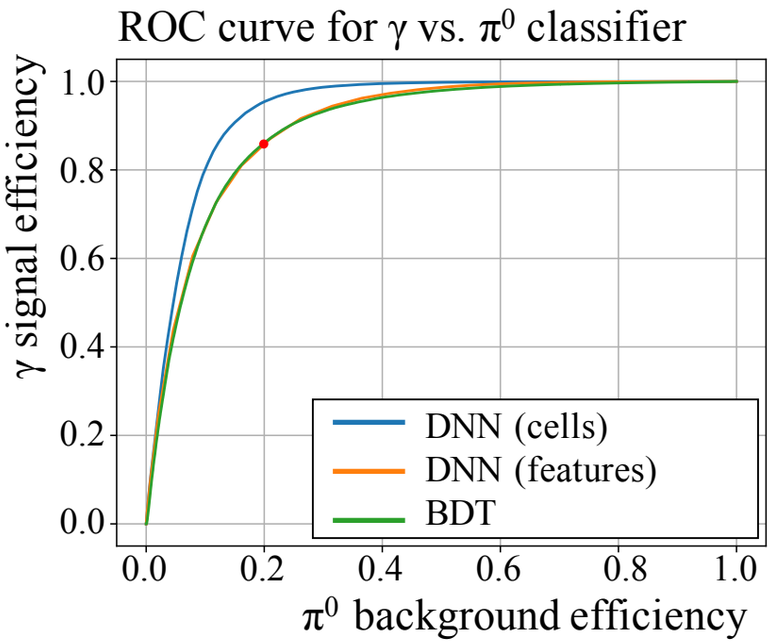
We present studies of the application of Deep Neural Networks and Convolutional Neural Networks for the classification, energy regression, and simulation of particles produced in high-energy particle collisions.We train cell-based Neural Nets that provide significant improvement in performance for particle classification and energy regression compared to feature-based Neural Nets and Boosted Decision Trees, and Generative Adversarial Networks that provide reasonable modeling of several but not all shower features.

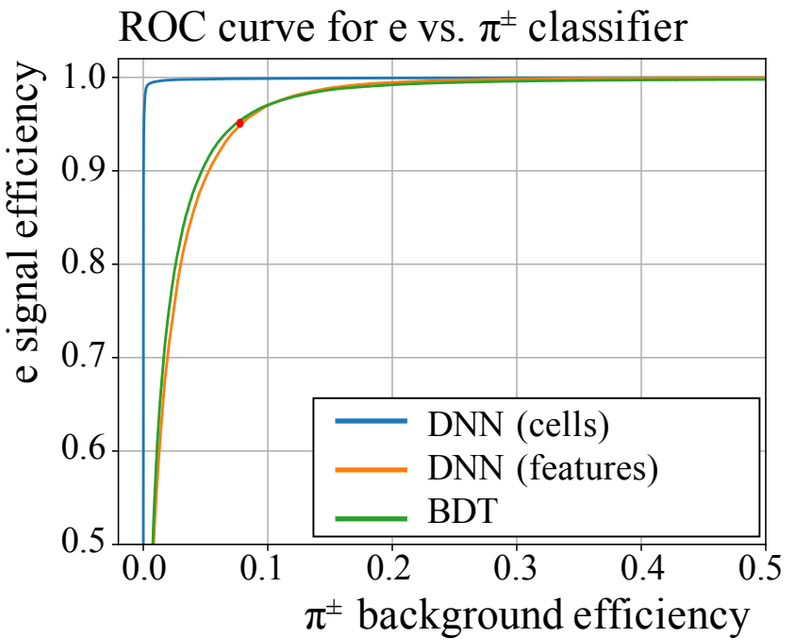
1. e/γ Particle Identification (Classification)

- Photon/lepton ID requires factor ~10000 jet rejection
- Jet like photon/lepton classification tasks:
  - *Task 1:* Electrons vs Electromagnetic $\pi^{+/-}$ (HCAL/ECAL Energy < 0.025)
  - *Task 2:* Photons vs Merging π^0 (2γ opening angel < 0.01 rad)
- *Comparison*:
  - ***Feature based*** BDT and DNN
  - ***Cell-based*** DNN (fully connected).
- Significant Improvement with cell-based DNNs.

	γ **vs.** π^0				e **vs.** π			
Model	**acc.**	**AUC**	$\Delta\epsilon_{\text{sig}}$	ΔR_{bkg}	**acc.**	**AUC**	$\Delta\epsilon_{\text{sig}}$	ΔR_{bkg}
BDT	83.1%	89.8%	-	-	93.8%	98.0%	-	-
DNN (features)	82.8%	90.2%	0.9%	0.95	93.6%	98.0%	-0.1%	0.95
DNN (cells)	87.2%	93.5%	9.4%	1.63	99.4%	99.9%	4.9%	151

Table 1: Performance parameters for BDT and DNN classifiers.





2. Energy Calibration (Regression)

- Energy ***resolution improves with energy***:
  - $\sigma(E) / E = a/\sqrt{E} \oplus b/E \oplus c$.
  - a = sampling, b = noise, c = leakage.
- *Comparison*:
  - ***Simple calibration****:* Sum energies (no noise) and scale.
  - ***CNN calibration****:* Cells → Particle energy

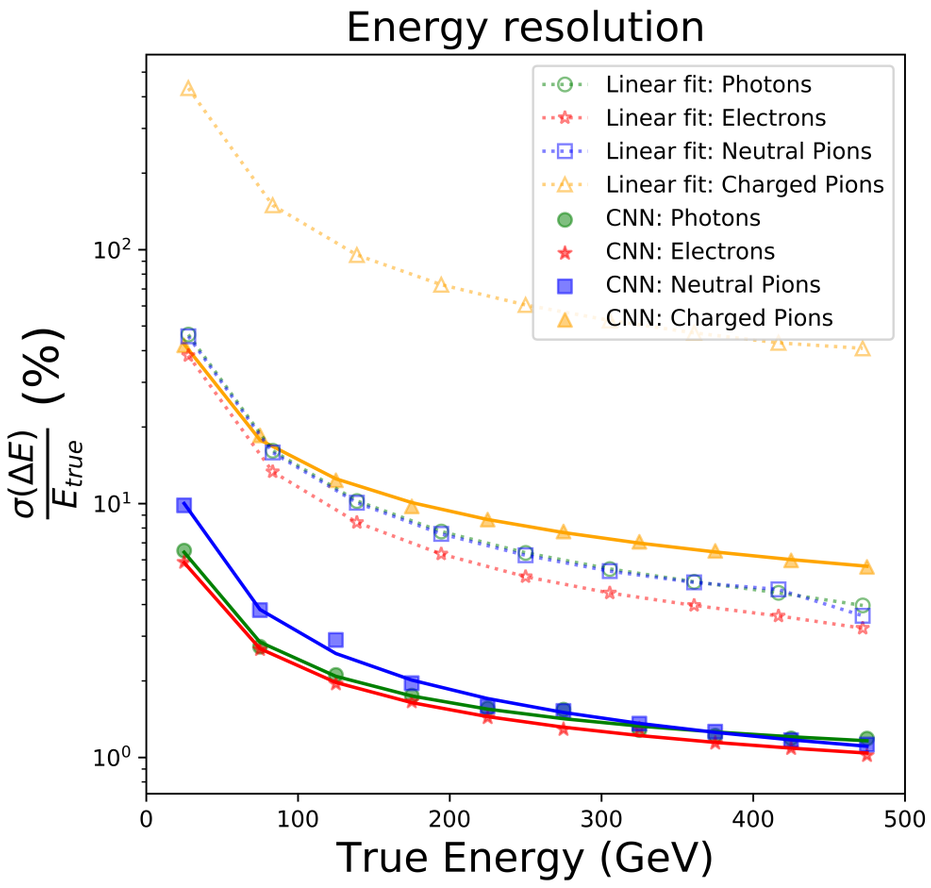
- Significant Improvement with CNN

Simple Linear Model

Particle Type	a	b	c
Photons	55.5	1.85	1245
Electrons	42.3	1.51	1037
Neutral pions	55.3	1.71	1222
Charged pions	442	25	11706

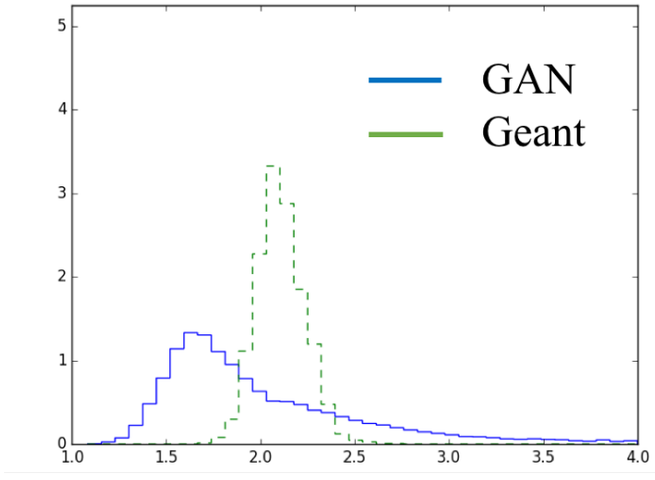
CNN Model

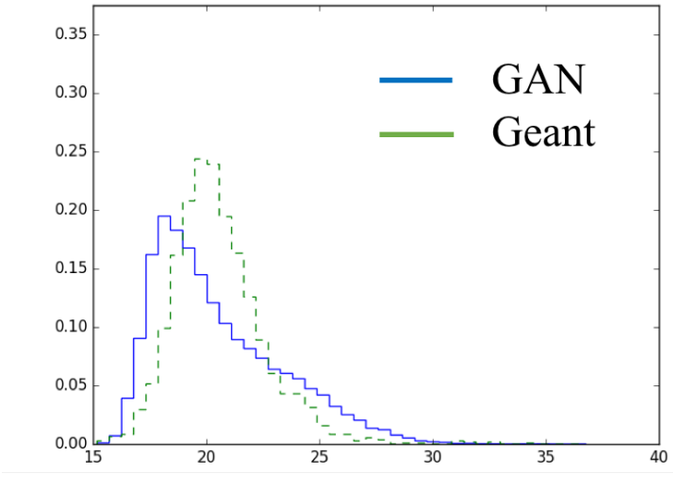
Particle Type	a	b	c
Photons	18.3	0.75	131
Electrons	18.7	0.574	111
Neutral pions	19.3	0.45	231
Charged pions	114	1.02	893



3. Simulation (Generative Model)

- Physics measurements typically require extremely detailed and precise simulation,
  - Software packages (e.g. Geant4) simulated the well understood ***micro-physics*** governing the interaction of particles with matter.
  - Generally very CPU intensive
    - *Example*: ATLAS experiment uses half of the experiment's computing resources for simulation.
- *Task:* ***CNN GAN conditioned on particle energy***
  - Accelerate simulation by many orders of magnitude.
- Promising start… but not yet faithfully reproducing all commonly used features extracted from generated images.





GANs for (fast) simulation

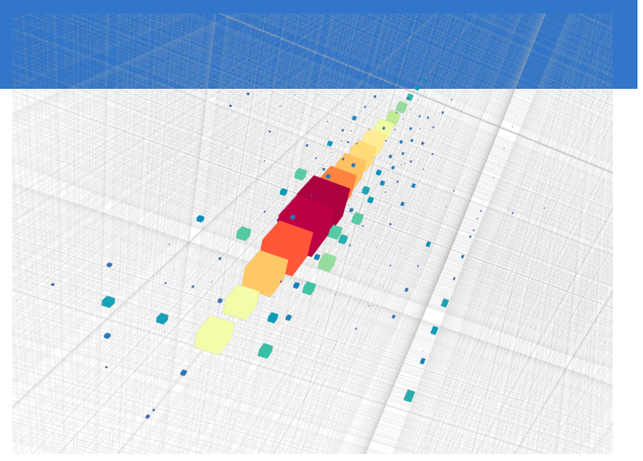
Sofia Vallecorsa for the GeantV team

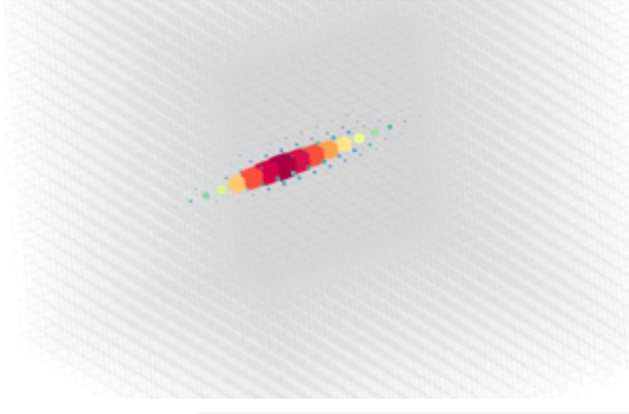
DS@HEP. FNAL. May 2017

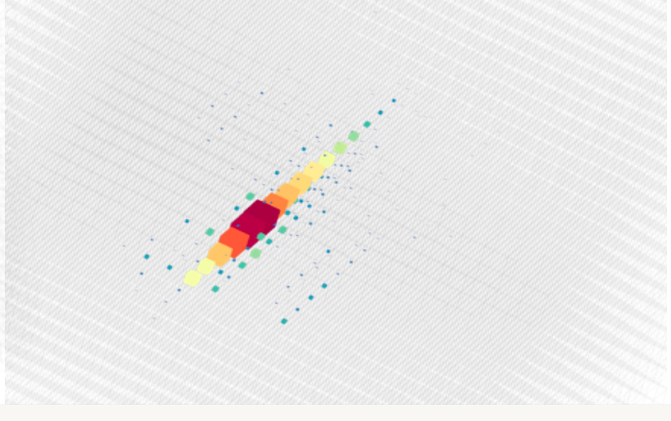
Preliminary

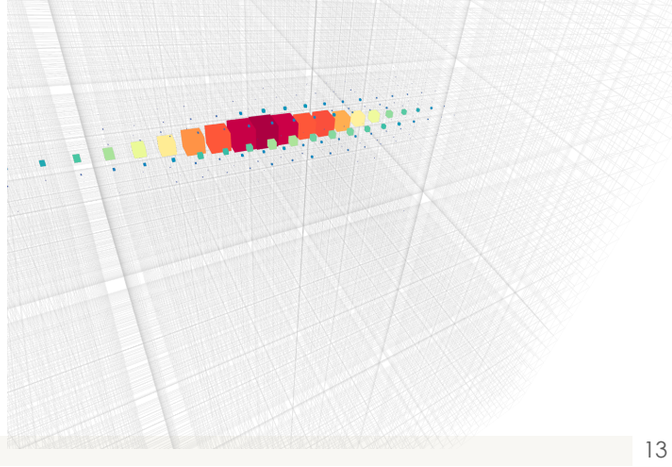
Some images

- Slice energy spectrum
- Start with photons & electrons



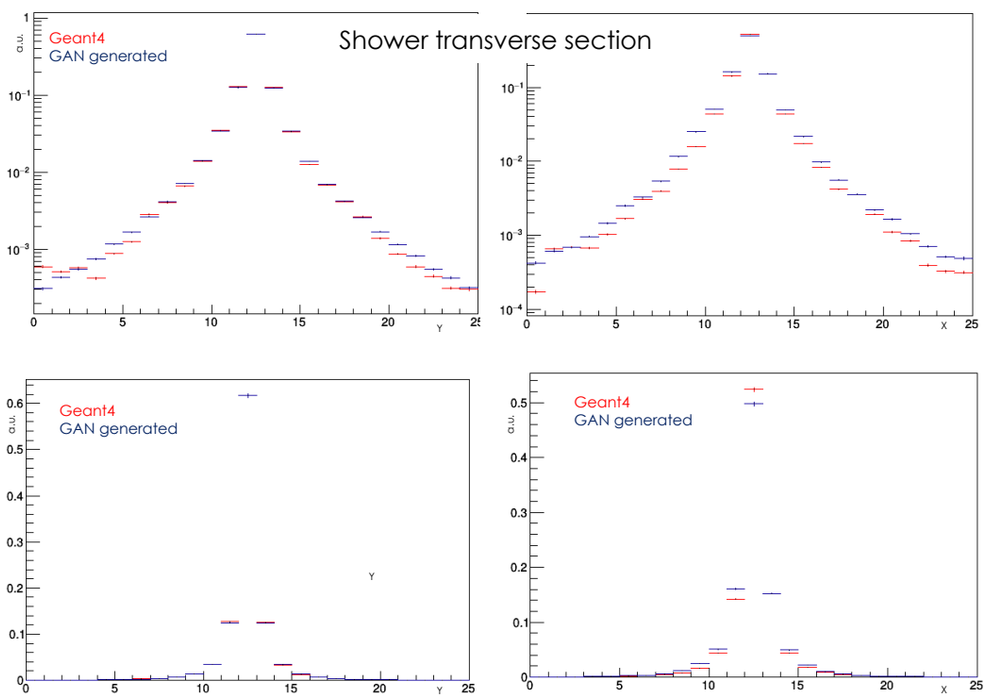


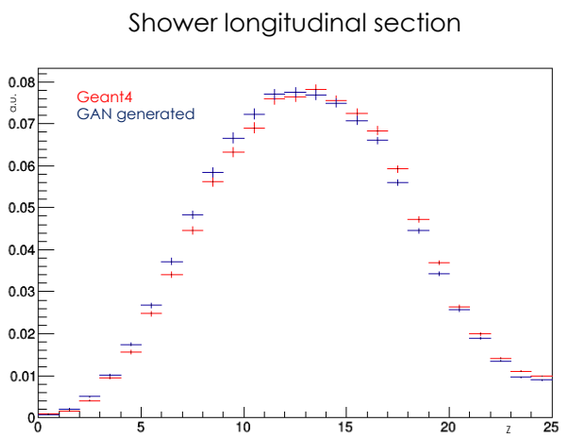




Preliminar

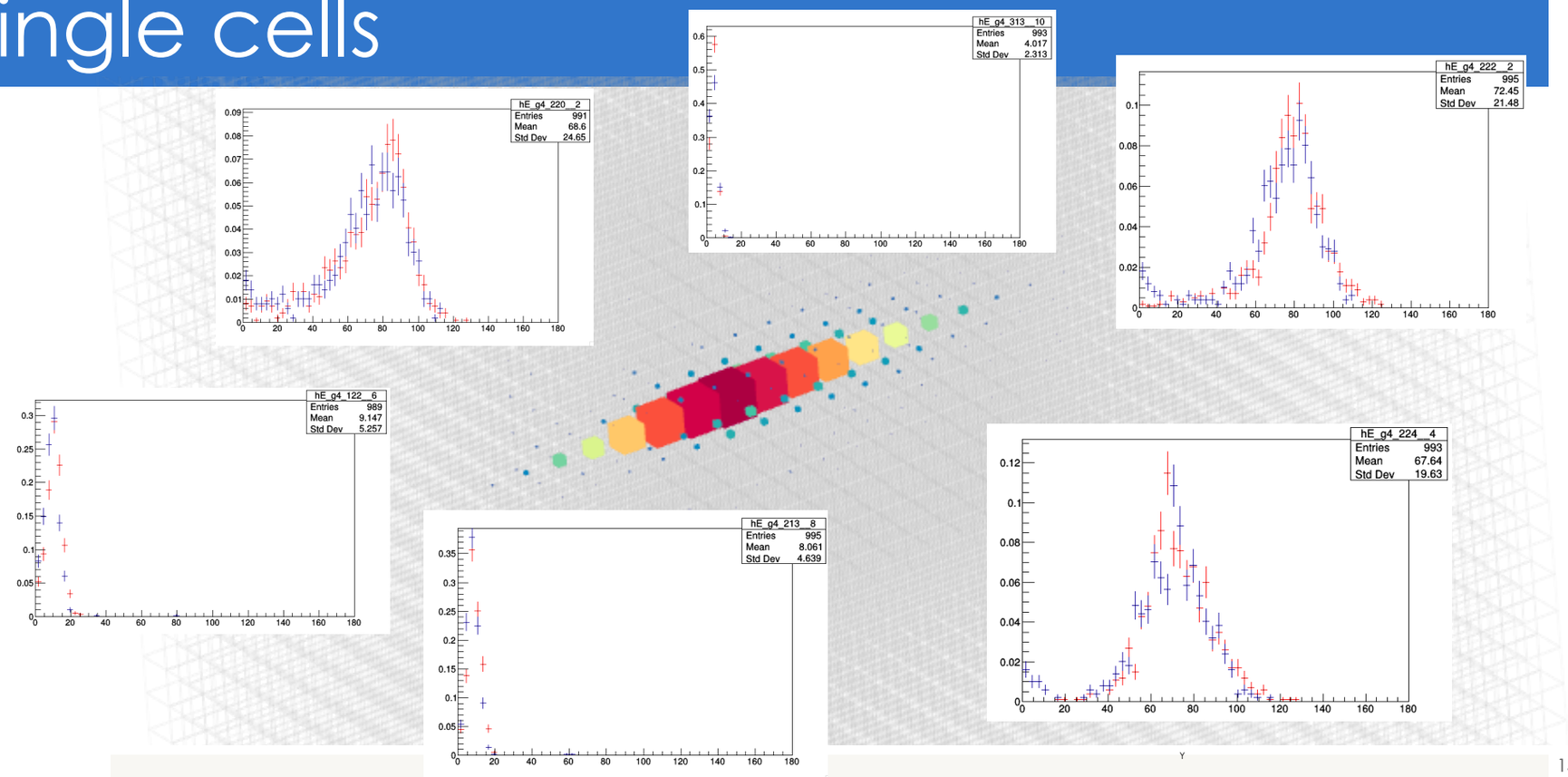
GAN generated electrons





Preliminary

Single cells



Jet Physics with Deep Learning

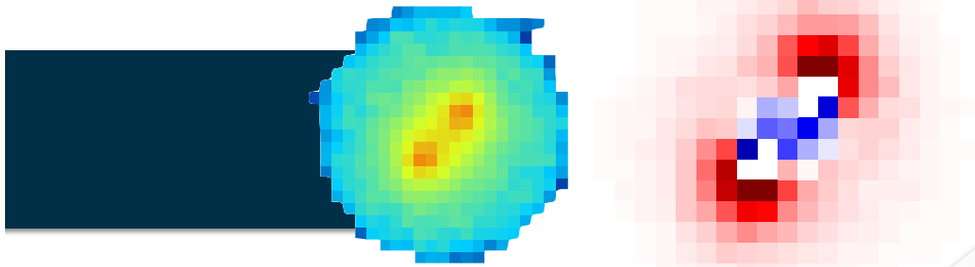
Modern Machine Learning

for Classification, Regression, and Generation in Jet Physics

BERKELEY LAB

Benjamin Nachman

Lawrence Berkeley National Laboratory



CERN Data Science Seminar, November

Next slides stolen from Ben Nachmans and Kyle Cranmer's Excellent CERN Seminars

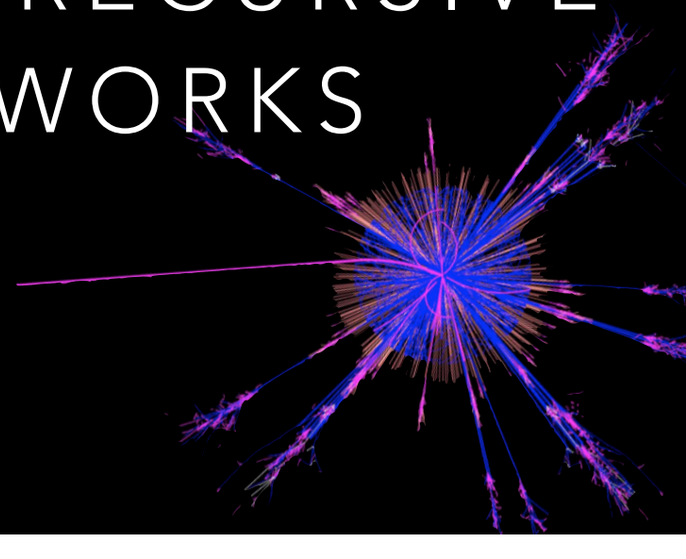
CENTER FOR COSMOLOGY AND PARTICLE PHYSICS

ARXIV:1702.00748

QCD-AWARE RECURSIVE NEURAL NETWORKS

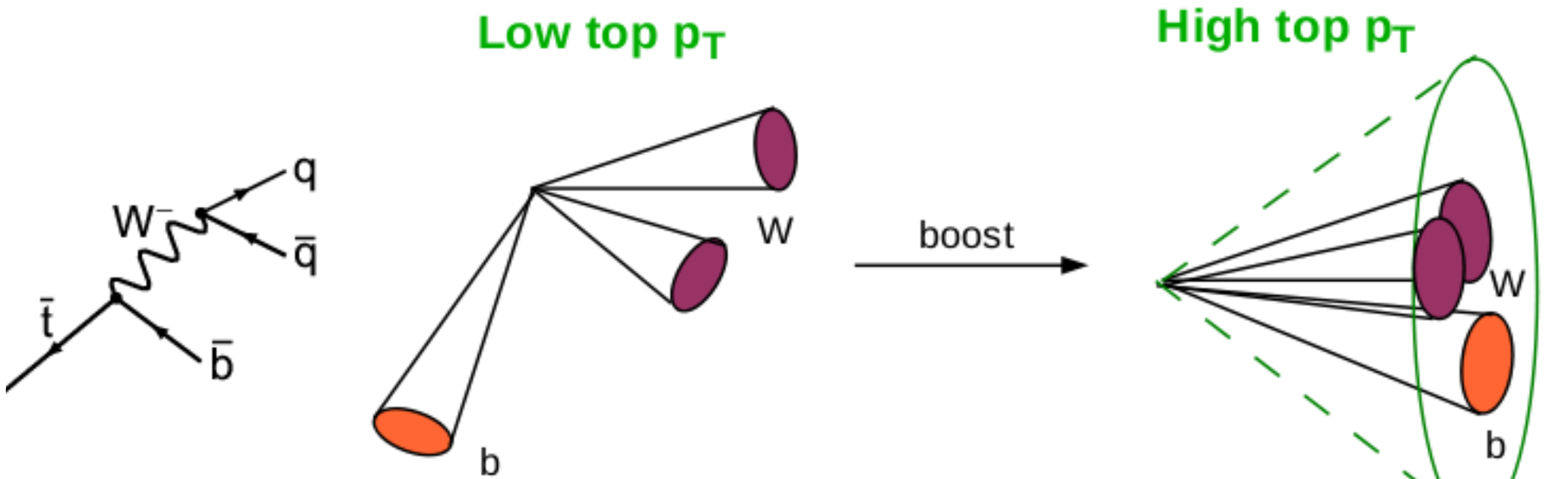
@KyleCranmer
New York University
Department of Physics
Center for Data Science

with:
Gilles Louppe
Kyunghyun Cho
Joan Bruna
Cyril Becot



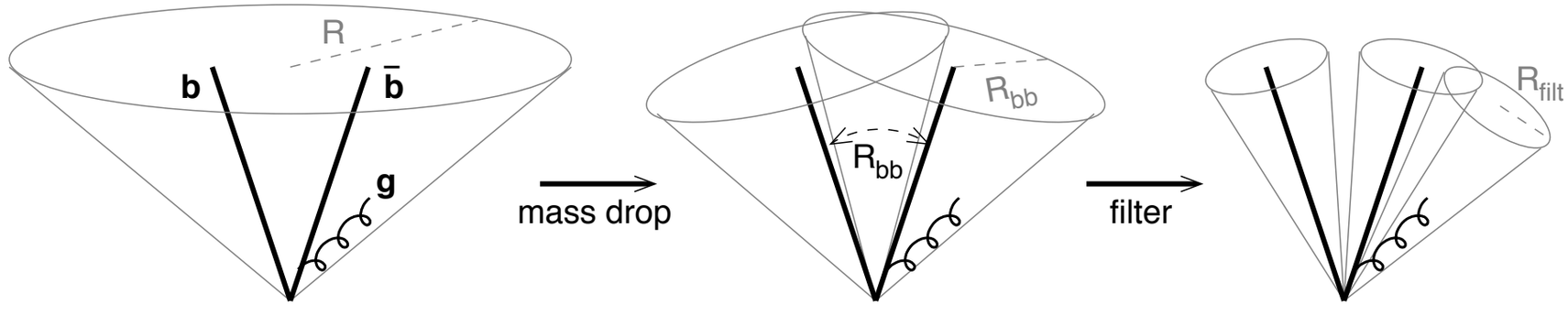
JET SUBSTRUCTURE

Many scenarios for physics Beyond the Standard Model include highly boosted W, Z, H bosons or top quarks



Identifying these rests on subtle substructure inside jets

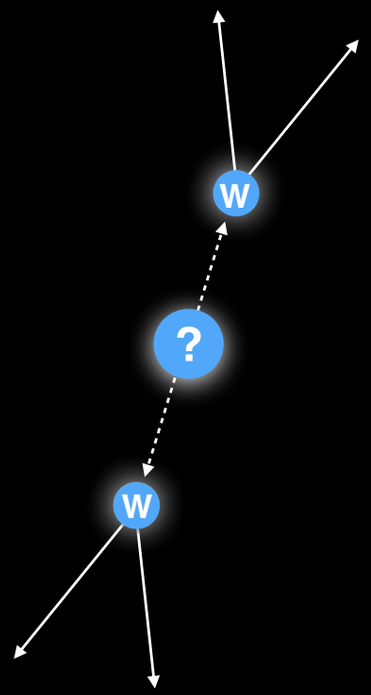
- an enormous number of theoretical effort in developing observables and techniques to tag jets like this



Goal: Find W jets in an enormous sea of generic q/g jets

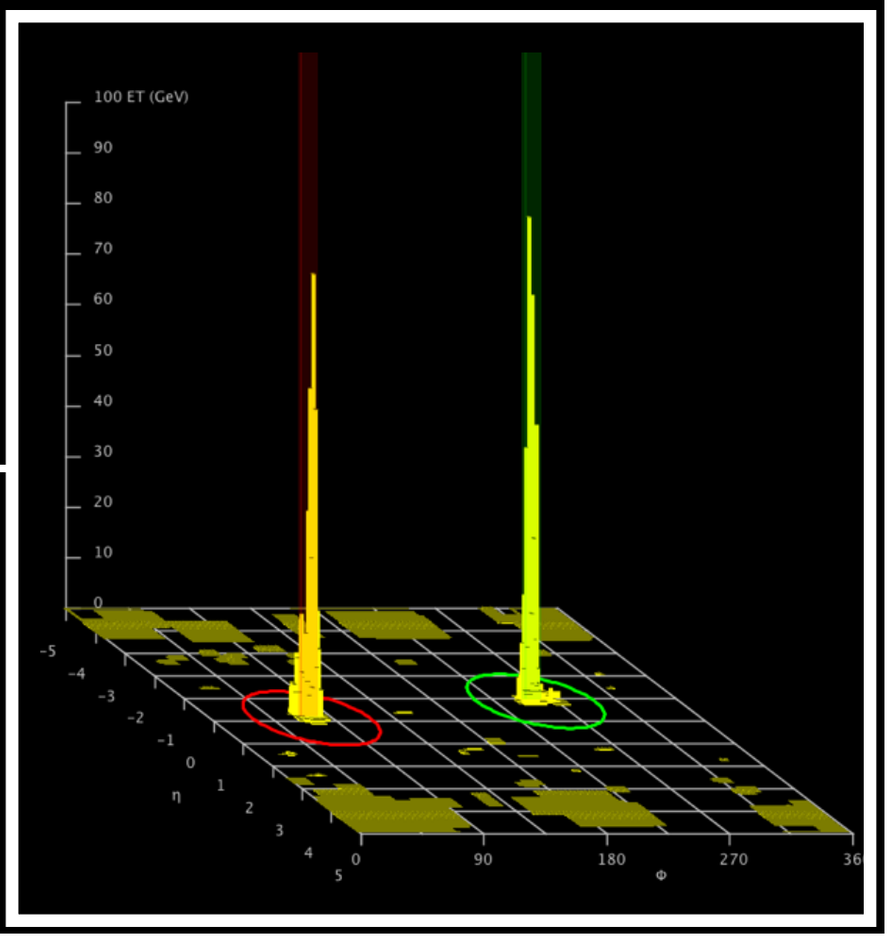
These jets have a non-trivial structure!

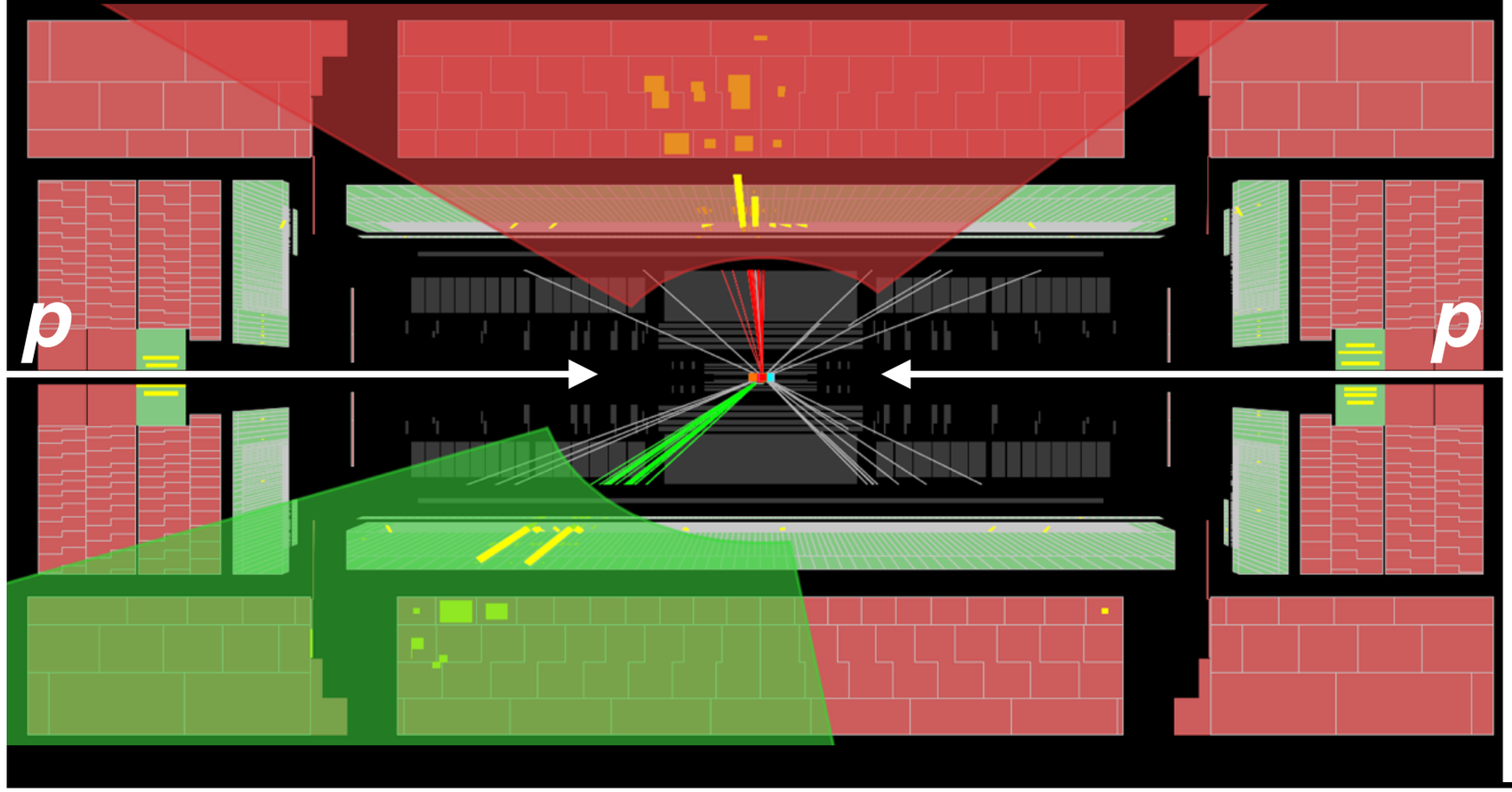
W bosons are naturally boosted if they result from the decay of something even heavier



Searching for new particles decaying into boosted W bosons requires **looking at the radiation pattern inside jets**

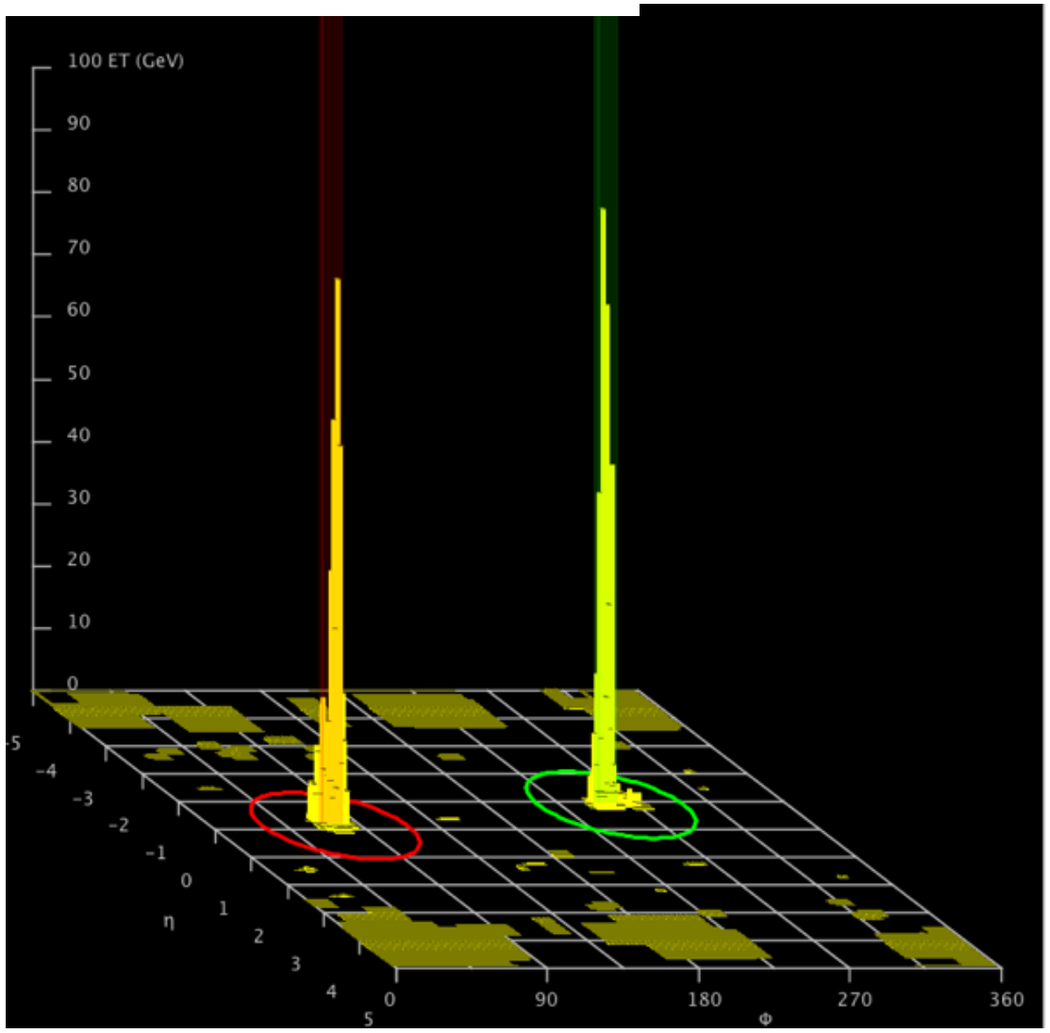
like a digital image!

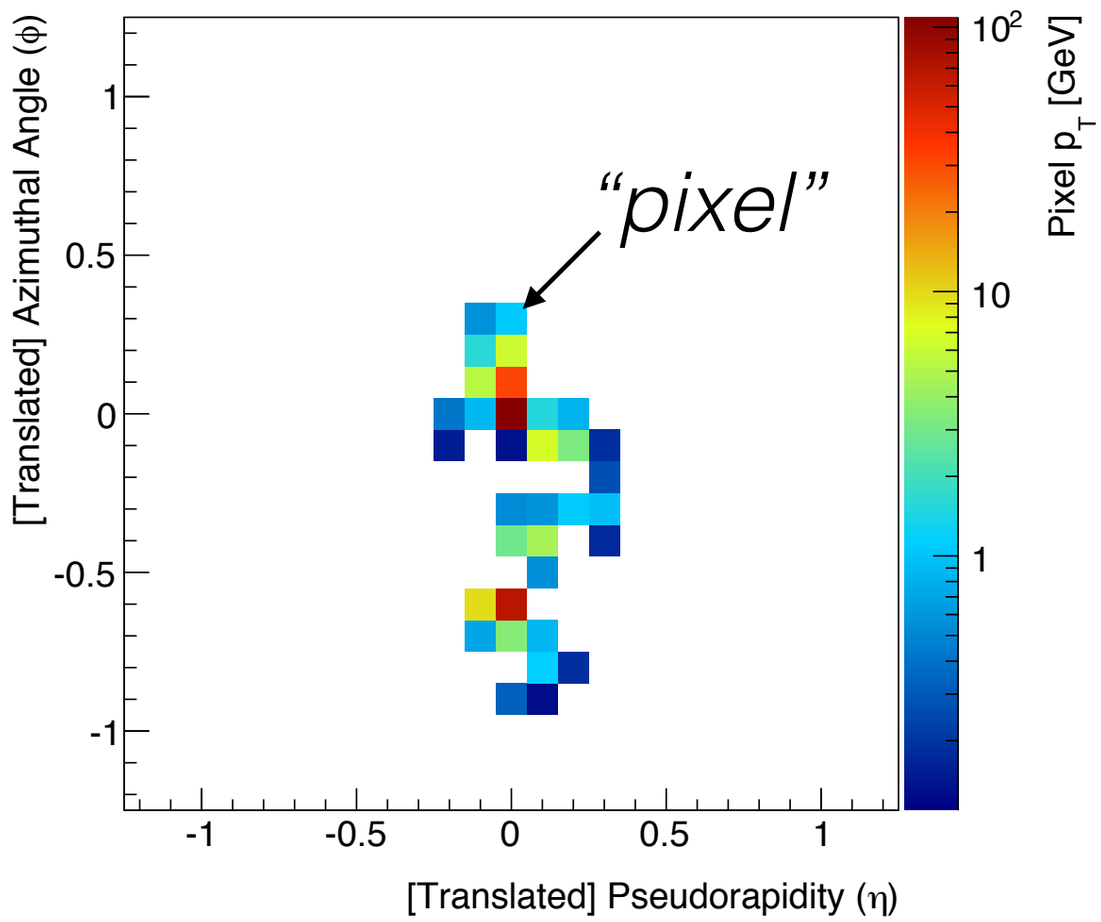




the Jet Image

J. Cogan et al. JHEP 02 (2015) 118



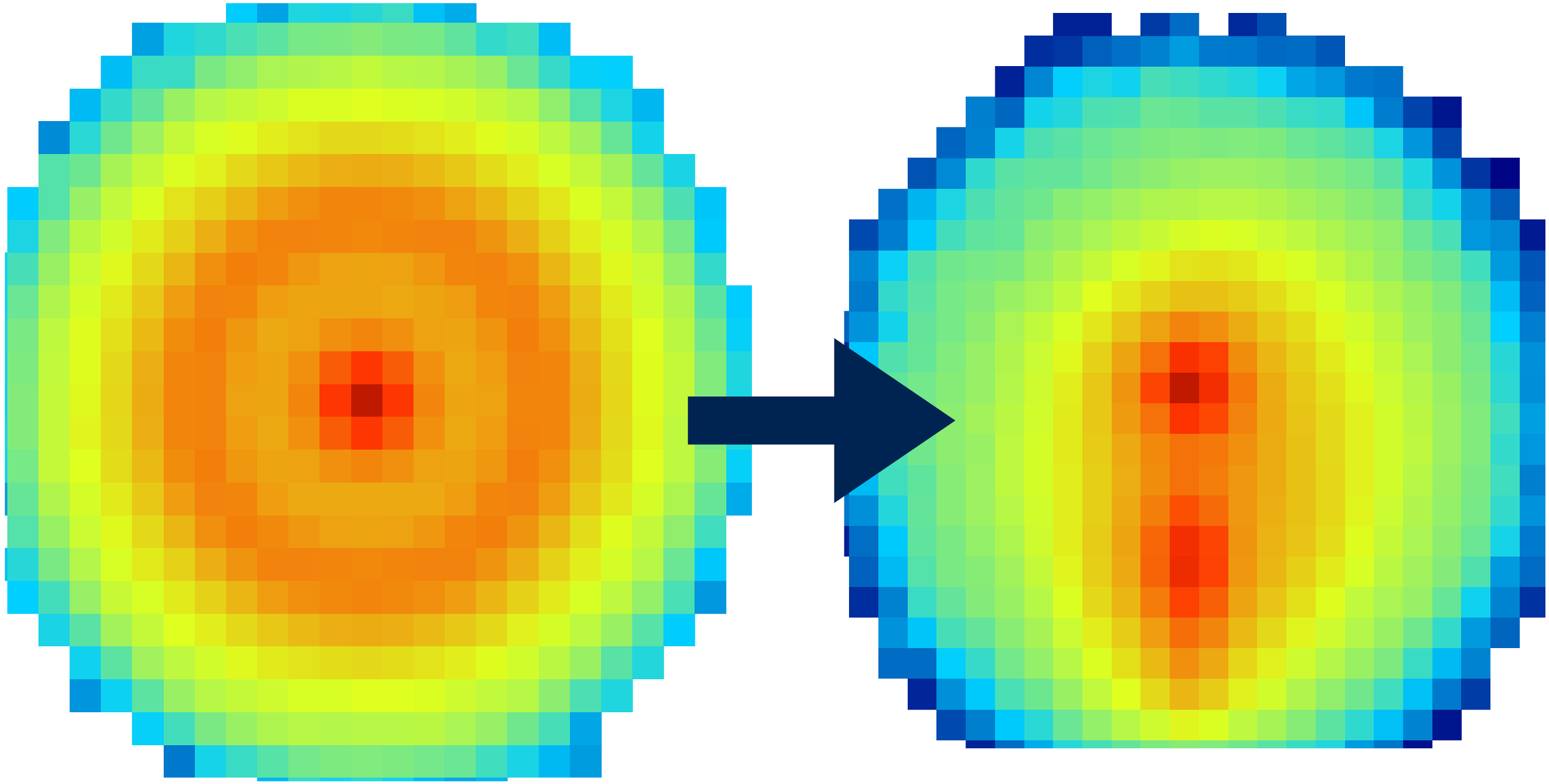


Credit: Peter G Trimming (Wikipedia)

no smooth edges, clear features, low occupancy (number of hit pixels)

Pre-processing & spacetime symmetries

One of the first typical steps is pre-processing



Can help to learn faster & smarter; but must be careful!

One of the most useful physics-inspired features is the ***jet mass***

$\phi = 0$

$\eta = 0$

Center on the highest energy *subjet*

From pixels

Translations in ϕ are rotations

Translations in η are boosts along the beam direction

Boosted W ⇢ qq'

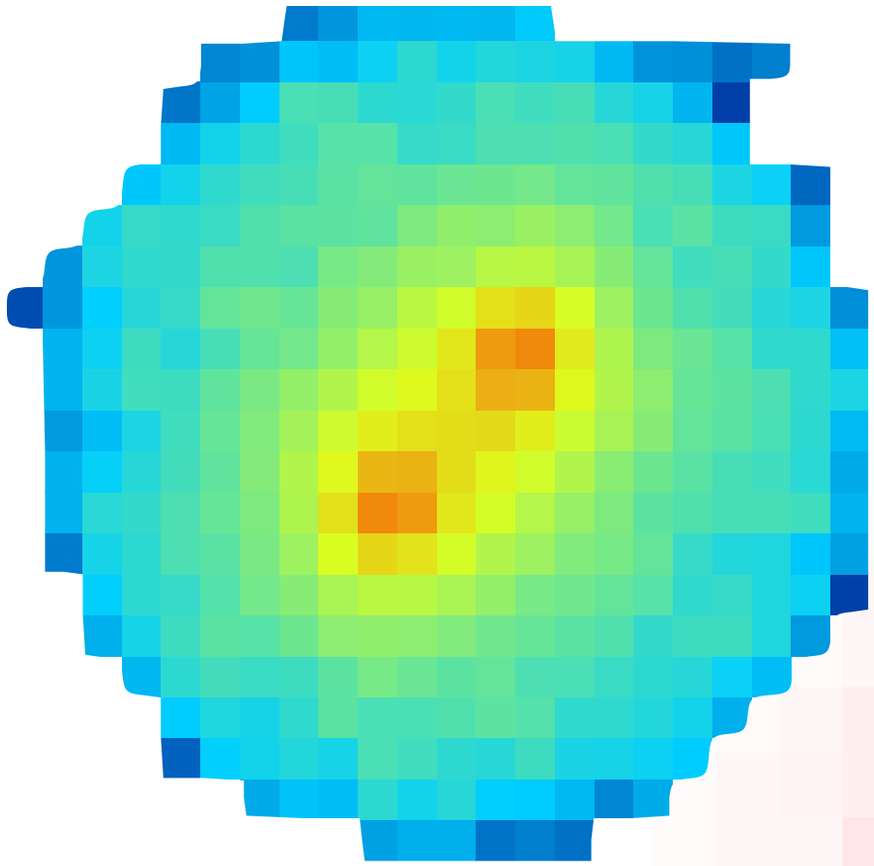
$m_W \sim 80$ GeV / c^2

Probability density

60 70 80 90 100 110

Jet Mass [GeV/c^2]

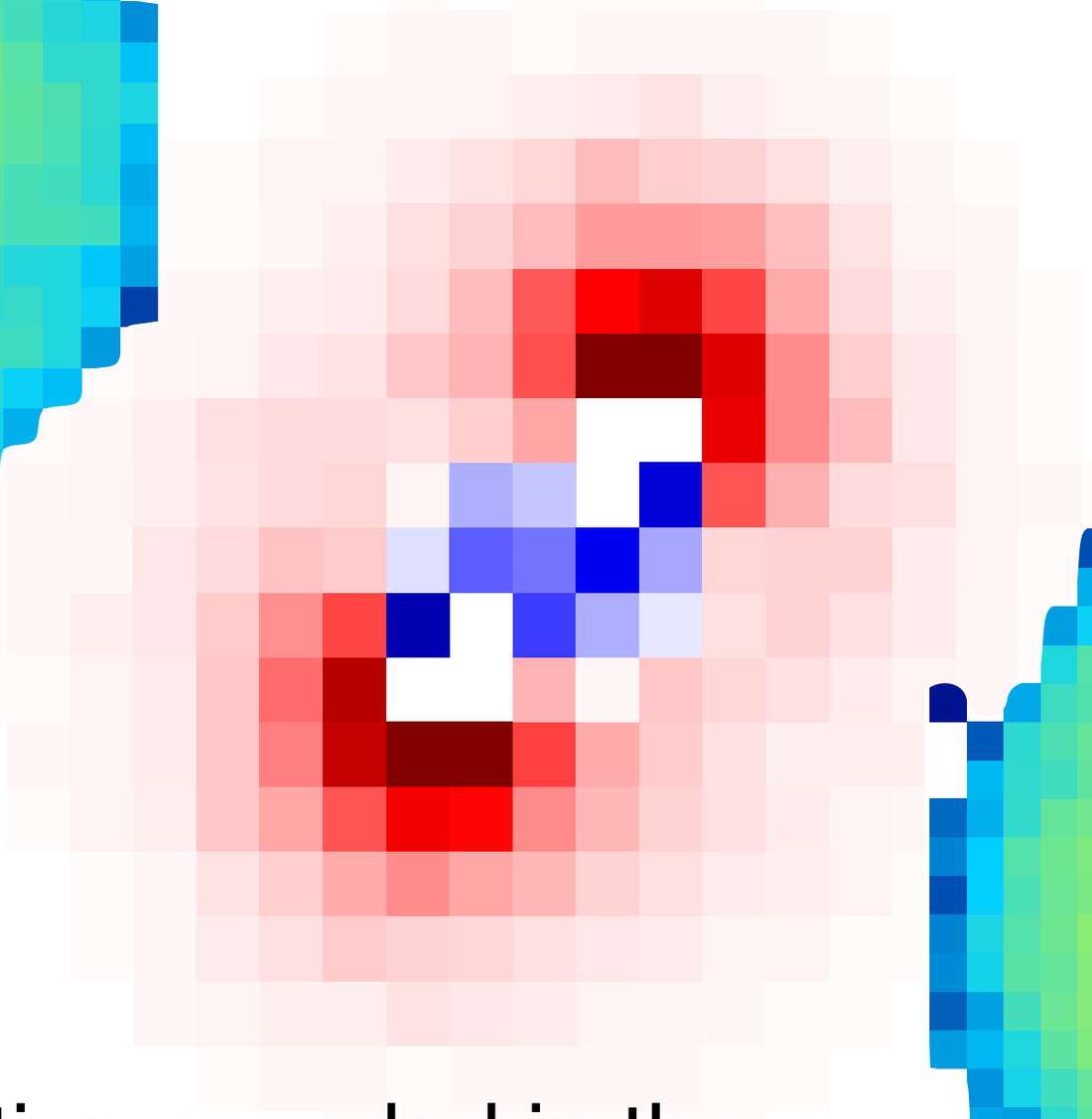
Why images?



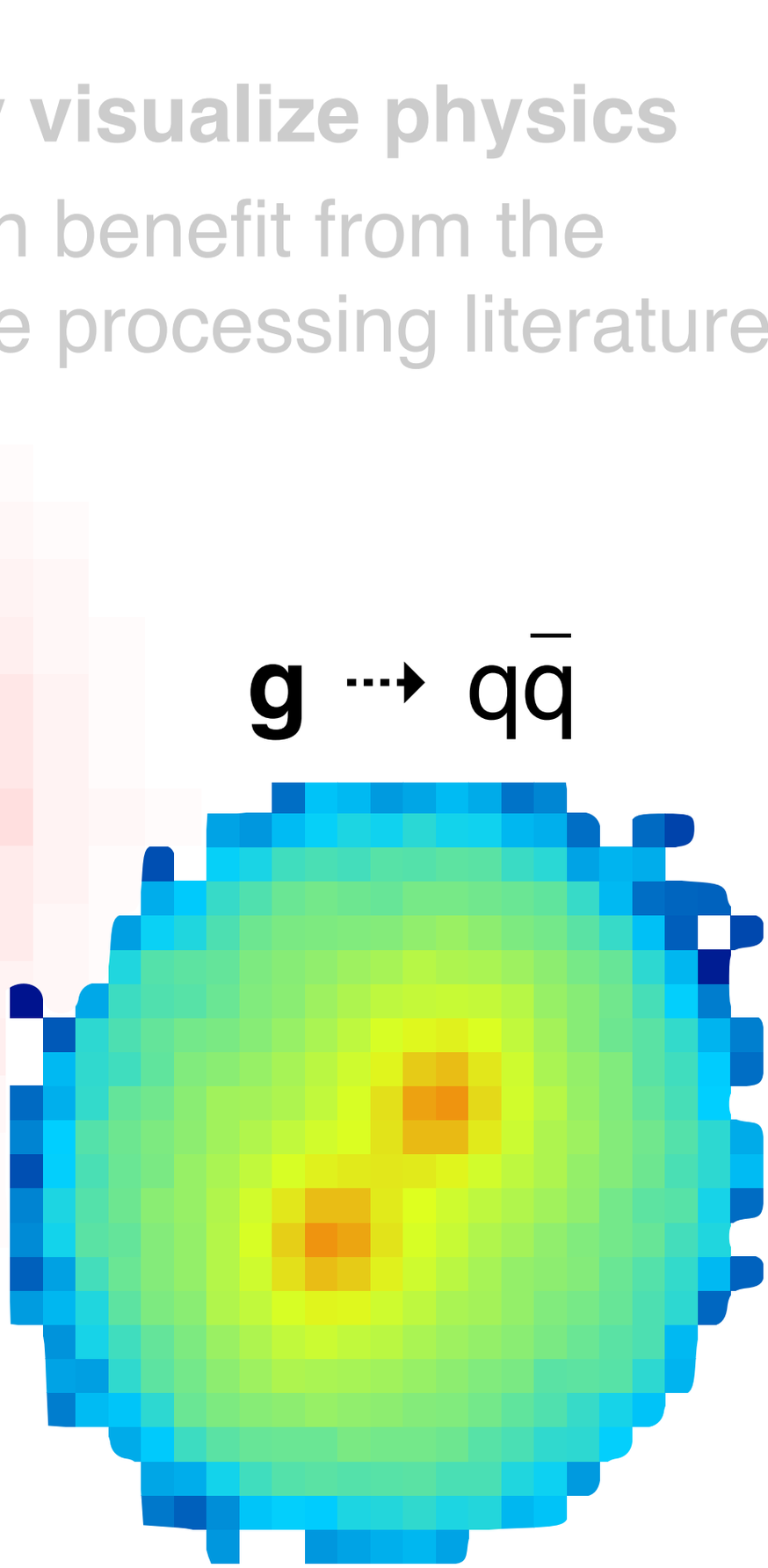
$W \dashrightarrow q\bar{q}$

Can directly visualize physics

and we can benefit from the extensive image processing literature

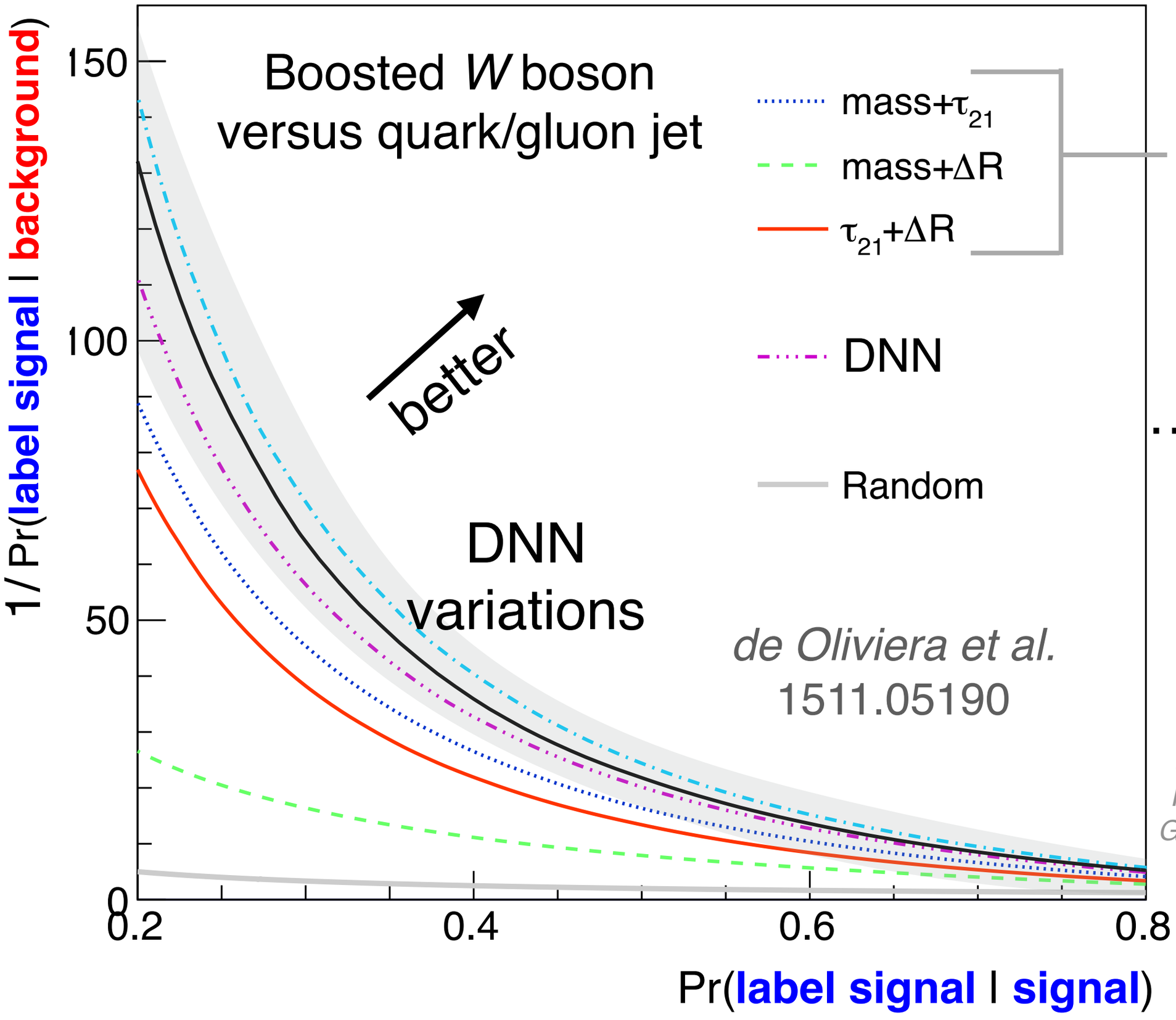


$g \dashrightarrow q\bar{q}$



there is information encoded in the physical distance between pixels

Modern Deep NN's for Classification



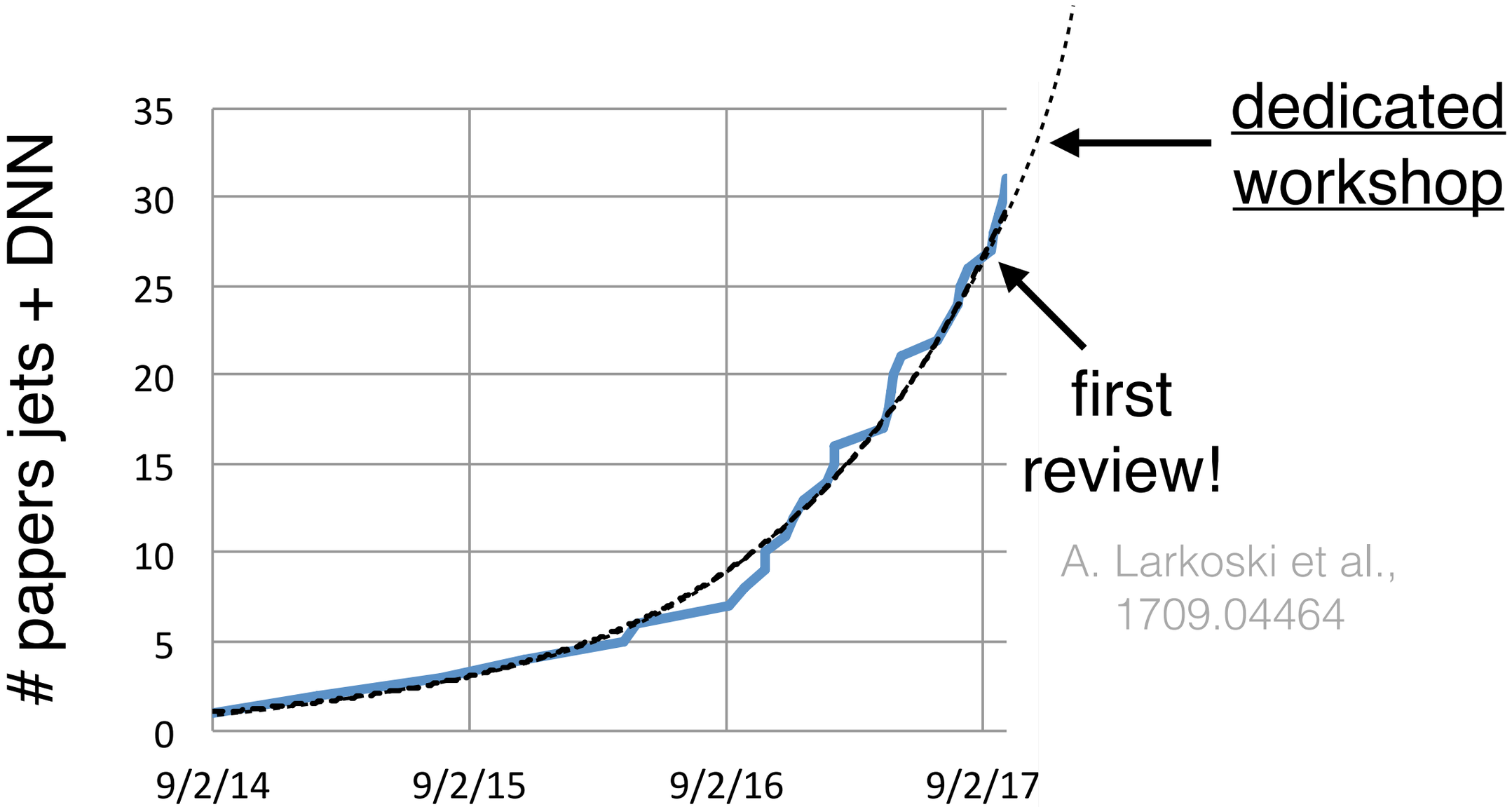
mass, τ_{21}, ΔR are all simple functions of the image

…what the DNN is learning is active R&D!

See also
L. Almeida et al. 1501.05968
Baldi et al. 1603.09349
J. Barnard et al. 1609.00607
P. Komiske et al. 1612.01551
G. Kasieczka et al. 1701.08784
W. Bhimji et al. 1711.03573

Exciting New Directions

So far only scratches the surface
…this is a very active field of research!



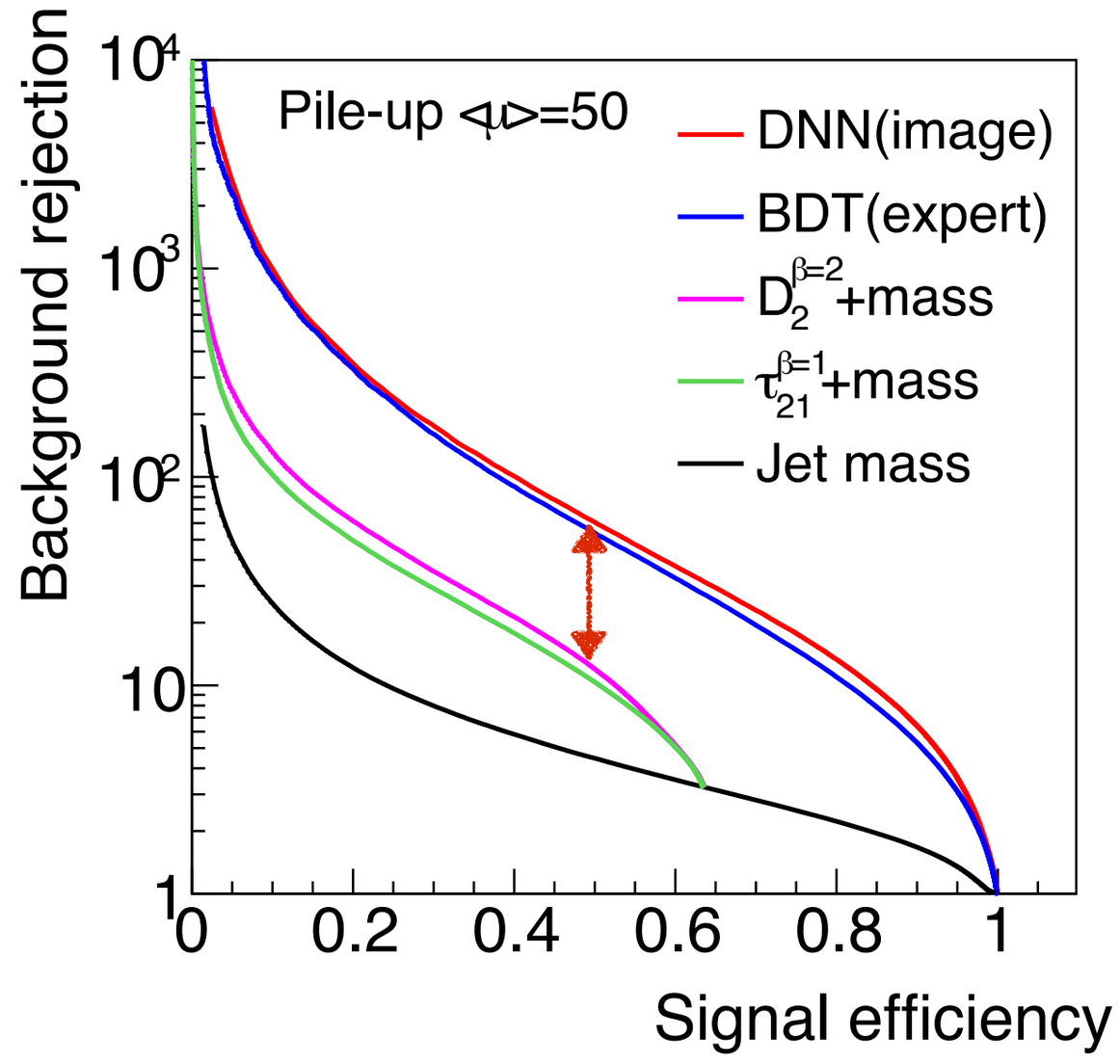
DEEP LEARNING VS. THEORY

Oliveira, et. al arXiv:1511.05190
Whiteson, et al arXiv:1603.09349
Barnard, et al arXiv:1609.00607

While the DNN shows a significant improvement with respect to the jet mass combined with single theory inspired variable (eg. τ_{21}, D_2), only a small improvement with respect to a BDT using several theory-inspired variables

Other Problems:

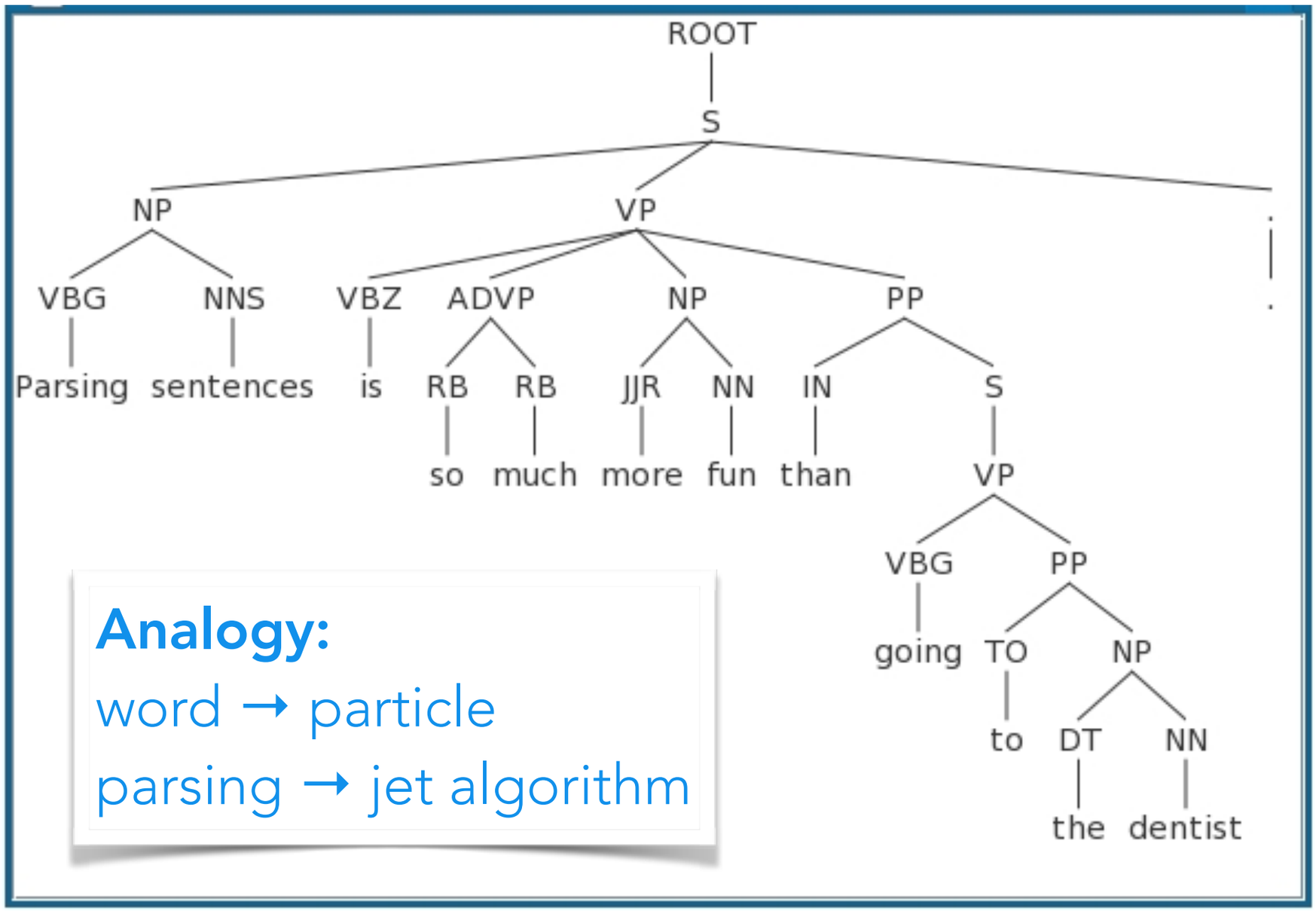
- image-based approach not easily generalized to non-uniform calorimeters
- not easy to extend to tracks, projecting into towers looses information
- theory inspired variables work on set of 4-vectors & have important theoretical properties



FROM IMAGES TO SENTENCES

Recursive Neural Networks showing great performance for Natural Language Processing tasks

- neural network's topology given by parsing of sentence!



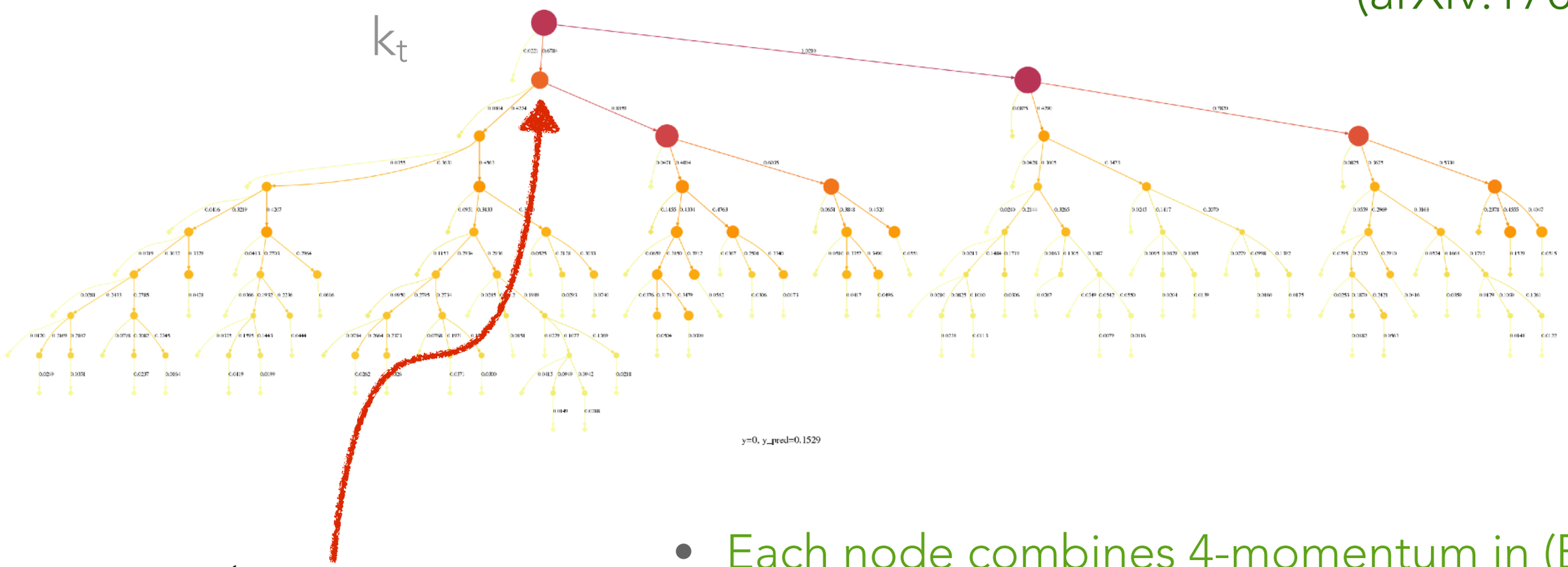
Analogy:

word → particle

parsing → jet algorithm

QCD-INSPIRED RECURSIVE NEURAL NETWORKS

(arXiv:1702.00748)



$$\mathbf{h}_k^{\text{jet}} = \begin{cases} \mathbf{u}_k & \text{if } k \text{ is a leaf} \\ \mathbf{z}_H \odot \tilde{\mathbf{h}}_k^{\text{jet}} + \mathbf{z}_L \odot \mathbf{h}_{k_L}^{\text{jet}} + \\ \hookrightarrow \mathbf{z}_R \odot \mathbf{h}_{k_R}^{\text{jet}} + \mathbf{z}_N \odot \mathbf{u}_k & \text{otherwise} \end{cases}$$

$$\mathbf{u}_k = \sigma\left(W_u g(\mathbf{o}_k) + b_u\right)$$

$$\mathbf{o}_k = \begin{cases} \mathbf{v}_{i(k)} & \text{if } k \text{ is a leaf} \\ \mathbf{o}_{k_L} + \mathbf{o}_{k_R} & \text{otherwise} \end{cases}$$

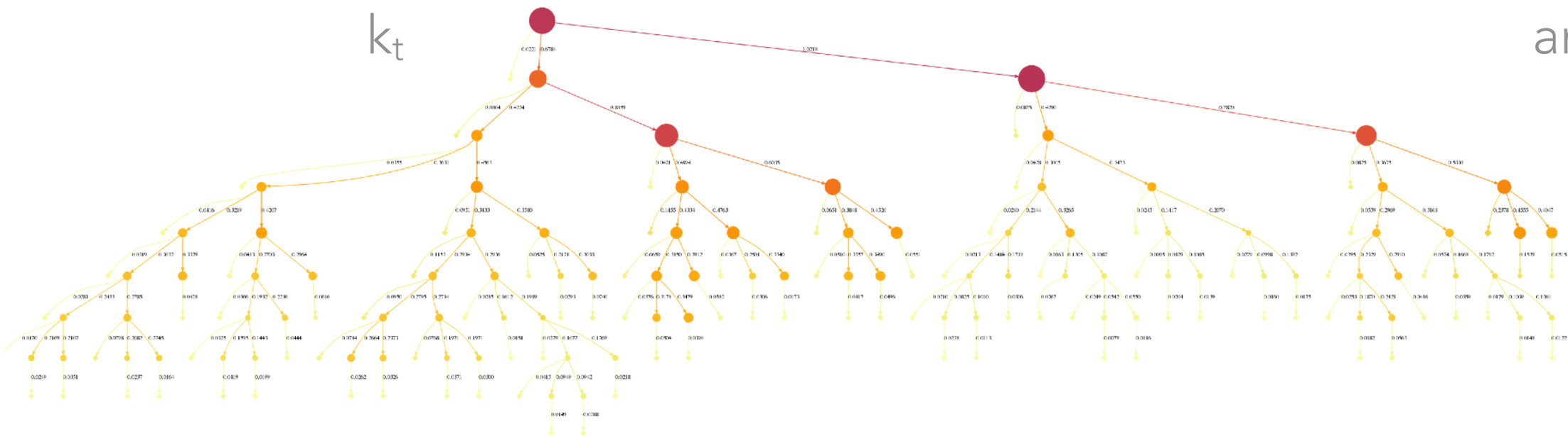
$$\tilde{\mathbf{h}}_k^{\text{jet}} = \sigma\left(W_{\tilde{h}} \begin{bmatrix} \mathbf{r}_L \odot \mathbf{h}_{k_L}^{\text{jet}} \\ \mathbf{r}_R \odot \mathbf{h}_{k_R}^{\text{jet}} \\ \mathbf{r}_N \odot \mathbf{u}_k \end{bmatrix} + b_{\tilde{h}}\right)$$

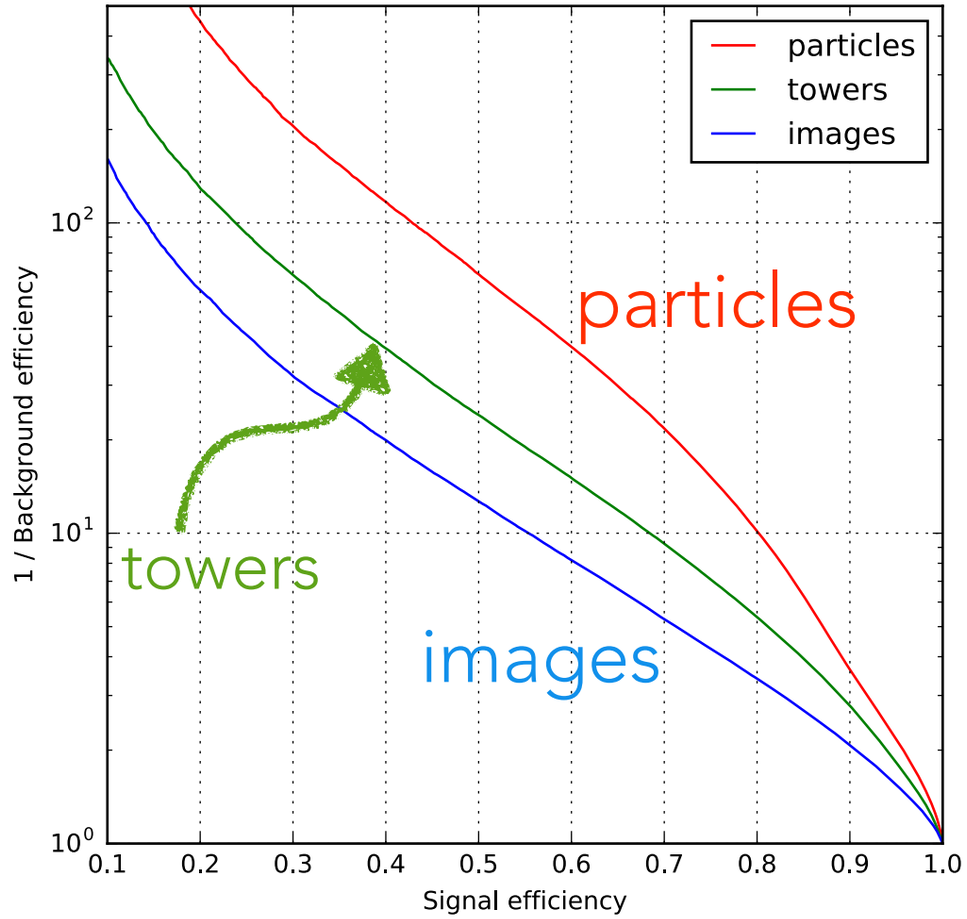
$$\begin{bmatrix} \mathbf{z}_H \\ \mathbf{z}_L \\ \mathbf{z}_R \\ \mathbf{z}_N \end{bmatrix} = \text{softmax}\left(W_z \begin{bmatrix} \tilde{\mathbf{h}}_k^{\text{jet}} \\ \mathbf{h}_{k_L}^{\text{jet}} \\ \mathbf{h}_{k_R}^{\text{jet}} \\ \mathbf{u}_k \end{bmatrix} + b_z\right)$$

$$\begin{bmatrix} \mathbf{r}_L \\ \mathbf{r}_R \\ \mathbf{r}_N \end{bmatrix} = \text{sigmoid}\left(W_r \begin{bmatrix} \mathbf{h}_{k_L}^{\text{jet}} \\ \mathbf{h}_{k_R}^{\text{jet}} \\ \mathbf{u}_k \end{bmatrix} + b_r\right)$$

- Each node combines 4-momentum in (E-scheme recombination of o_k) and a non-linear transformation of hidden state of children h_{kL}, $h_{kR} \in \mathbb{R}^{40}$
- Recursively applied (shared weights, Markov)
- "gating" allows for weighting of information of L/R children and for to flow directly along one branch

QCD-INSPIRED RECURSIVE NEURAL NETWORKS





- W-jet tagging example using data from Dawe, et al arXiv:1609.00607
- down-sampling by projecting into images looses information
- RNN needs much less data to train!

Neural Message Passing for Jet Physics

Isaac Henrion, Johann Brehmer, Joan Bruna, Kyunghun Cho, Kyle Cranmer
Center for Data Science
New York University
New York, NY 10012
{henrion*, johann.brehmer, bruna, kyunghyun, kyle.cranmer*}@nyu.edu

Gilles Louppe
Department of Computer Science
University of Liège
Belgium
g.louppe@ulg.ac.be

Gaspar Rochette
Department of Computer Science
École Normale Supérieure
Paris, France
gaspar.rochette@ens.fr

Abstract

Supervised learning has incredible potential for particle physics, and one application that has received a great deal of attention involves collimated sprays of particles called jets. Recent progress for jet physics has leveraged machine learning techniques based on computer vision and natural language processing. In this work, we consider message passing on a graph where the nodes are the particles in a jet. We design variants of a message-passing neural network (MPNN); (1) with a learnable adjacency matrix, (2) with a learnable symmetric adjacency matrix, and (3) with a set2set aggregated hidden state and MPNN with an identity adjacency matrix. We compare these against the previously proposed recursive neural network with a fixed tree structure and show that the MPNN with a learnable adjacency matrix and two message-passing iterations outperforms all the others.

Table 1: Summary of classification performance for several approaches.

Network	Iterations	ROC AUC	$R_{\epsilon=50\%}$
RecNN-k_t (without gating) [10]	1	0.9185 ± 0.0006	68.3 ± 1.8
RecNN-k_t (with gating) [10]	1	0.9195 ± 0.0009	74.3 ± 2.4
RecNN-desc-p_T (without gating) [10]	1	0.9189 ± 0.0009	70.4 ± 3.6
RecNN-desc-p_T (with gating) [10]	1	0.9212 ± 0.0005	$\mathbf{83.3 \pm 3.1}$
RelNet	1	0.9161 ± 0.0029	67.69 ± 6.80
MPNN (directed)	1	0.9196 ± 0.0015	89.35 ± 3.54
MPNN (directed)	2	0.9223 ± 0.0008	$\mathbf{98.26 \pm 4.28}$
MPNN (directed)	3	0.9188 ± 0.0031	85.93 ± 8.50
MPNN (undirected)	1	0.9193 ± 0.0015	86.41 ± 3.80
MPNN (undirected)	2	0.8949 ± 0.1004	$\mathbf{97.27 \pm 5.02}$
MPNN (undirected)	3	0.9185 ± 0.0036	84.53 ± 8.64
MPNN (set, directed)	1	0.9189 ± 0.0017	88.23 ± 4.53
MPNN (set, directed)	2	0.9191 ± 0.0046	87.46 ± 14.14
MPNN (set, directed)	3	0.9176 ± 0.0049	$\mathbf{88.33 \pm 9.84}$
MPNN (set, undirected)	1	0.9196 ± 0.0014	85.65 ± 4.48
MPNN (set, undirected)	2	0.9220 ± 0.0007	$\mathbf{94.70 \pm 2.95}$
MPNN (set, undirected)	3	0.9158 ± 0.0054	75.94 ± 12.54
MPNN (id)	1	0.9169 ± 0.0013	74.75 ± 2.65
MPNN (id)	2	0.9162 ± 0.0020	74.41 ± 3.50
MPNN (id)	3	0.9158 ± 0.0029	74.51 ± 5.20

Hands on…

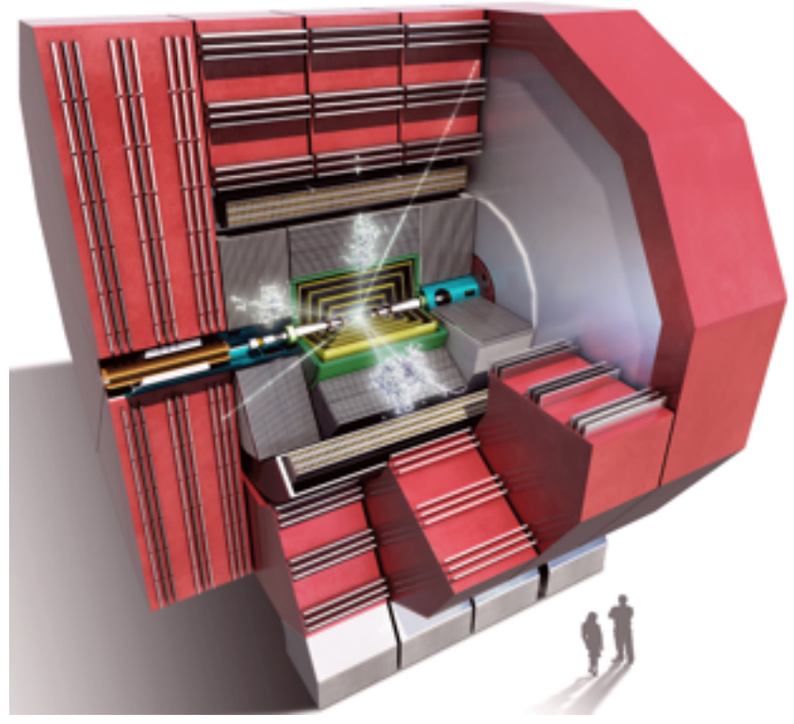
Example Datasets

Public Datasets

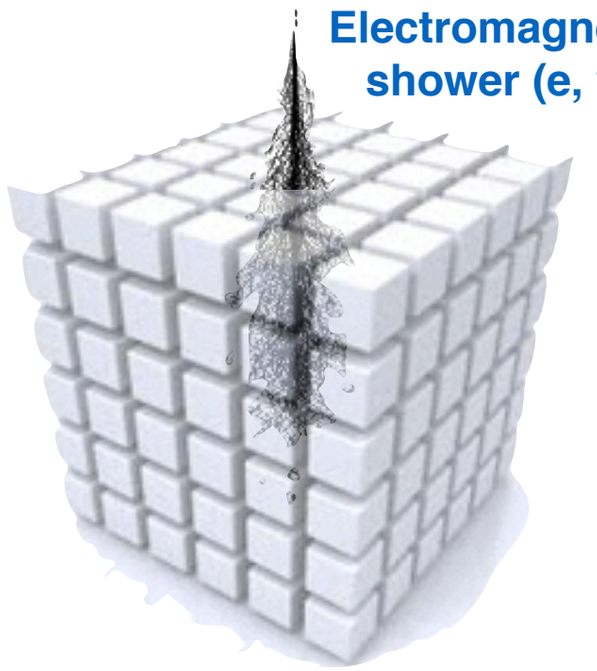
- ***Biggest obstacles*** to DNN research is ***Data accessibility***.
  - Detector level studies require ***CPU intensive simulations***.
  - DNNs require large training sets with ***full level of detail*** (i.e. not 4-vectors).
  - Experiments have such samples, but they are not easily accessible and ***not public***.
  - ***Difficult to collaborate*** with DL community or other experiments.
- ***Public datasets:***
  - Jet Images, SUSY/Higgs, HiggsML, …
  - ***LArTPC*** (Sepideh Shahsavarani, AF): LArIAT detector. 1 M of every particle species (including neutrinos).
    - Challenges: Particle/Neutrino Classification and Energy Reco, Noise Suppression, 2D->3D.
  - ***Calorimetry*** (Maurizio Pierini, Jean-Roch Vlimant, Nikita Smirnov, AF): LCD Calorimeter.
    - Challenges: PID/Energy Reco. Simulation.

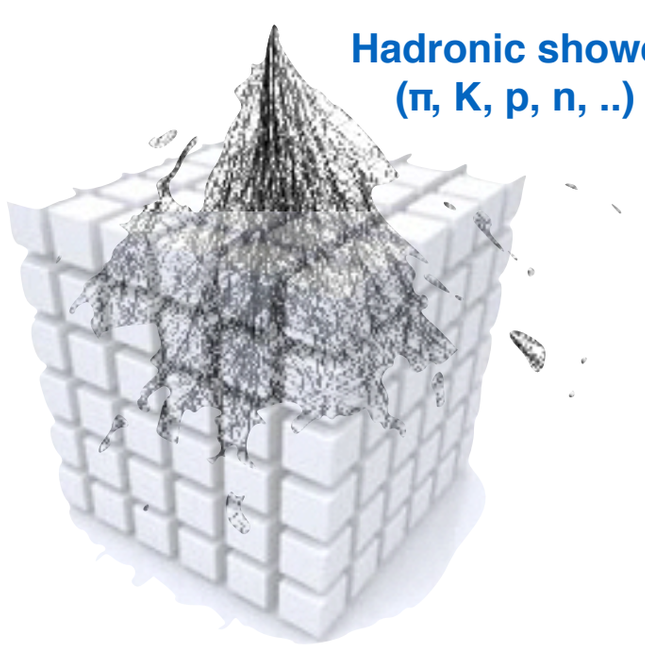
Calorimeter Dataset

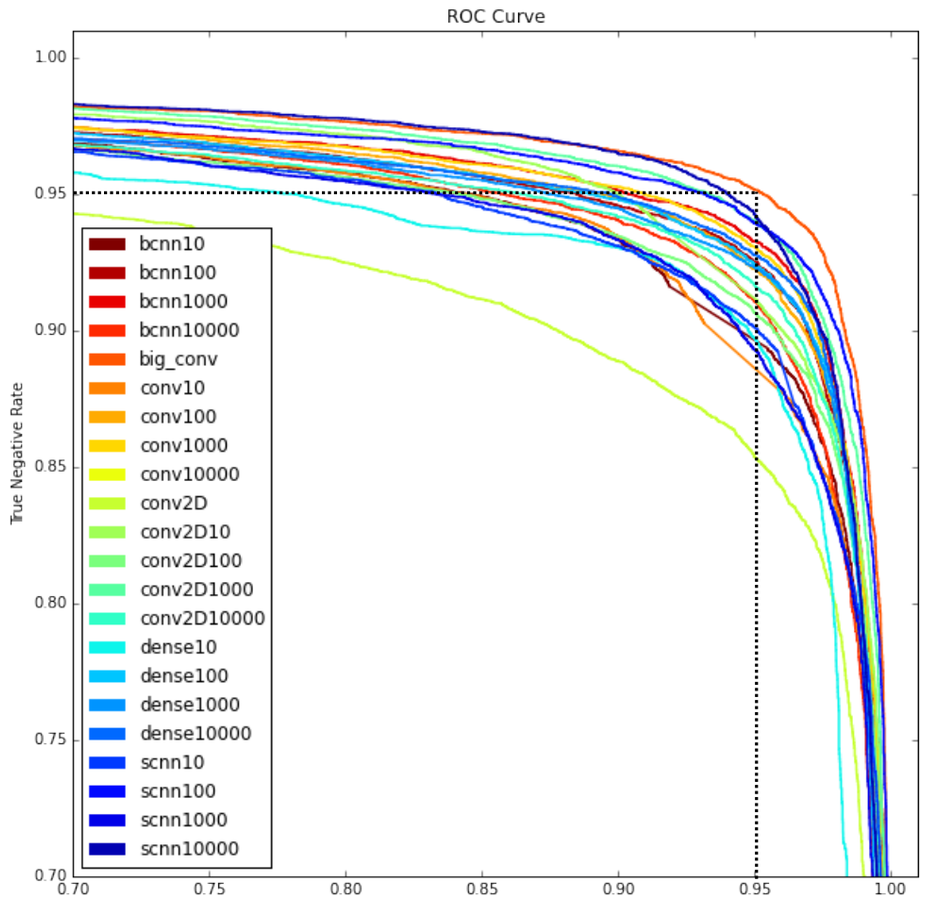
- CLIC is a proposed CERN project for a linear accelerator of electrons and positrons to TeV energies (~ LHC for protons)
    - LCD is a detector concept.
    - Not a real experiment yet, so we could simulate data and make it public.
- The LCD calorimeter is an array of absorber material and silicon sensors comprising the most granular calorimeter design available
    - Data is essentially a 3D image



- With at effective eta/phi resolution of 0.003x0.003, we can down sample to get ~ ATLAS granularity: 0.025x0.1 (pre-sampler) to 0.2x0.1 Tile D.





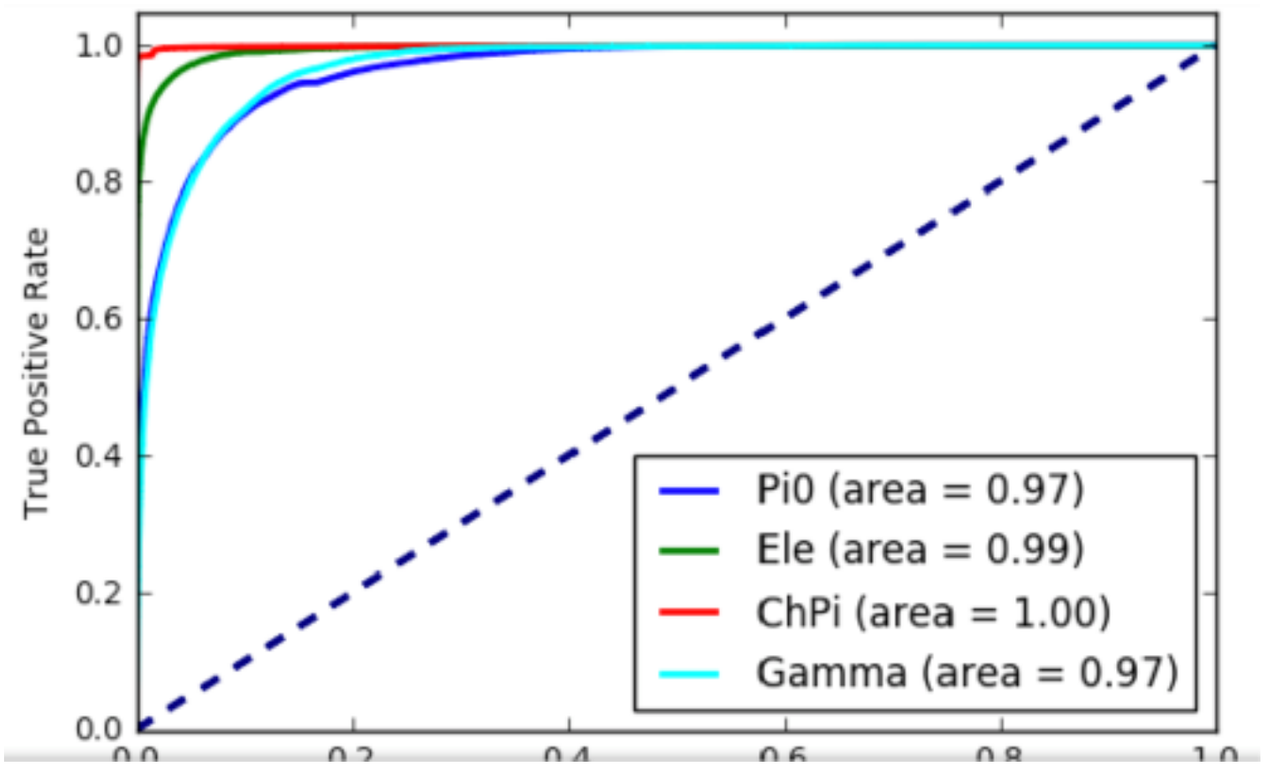


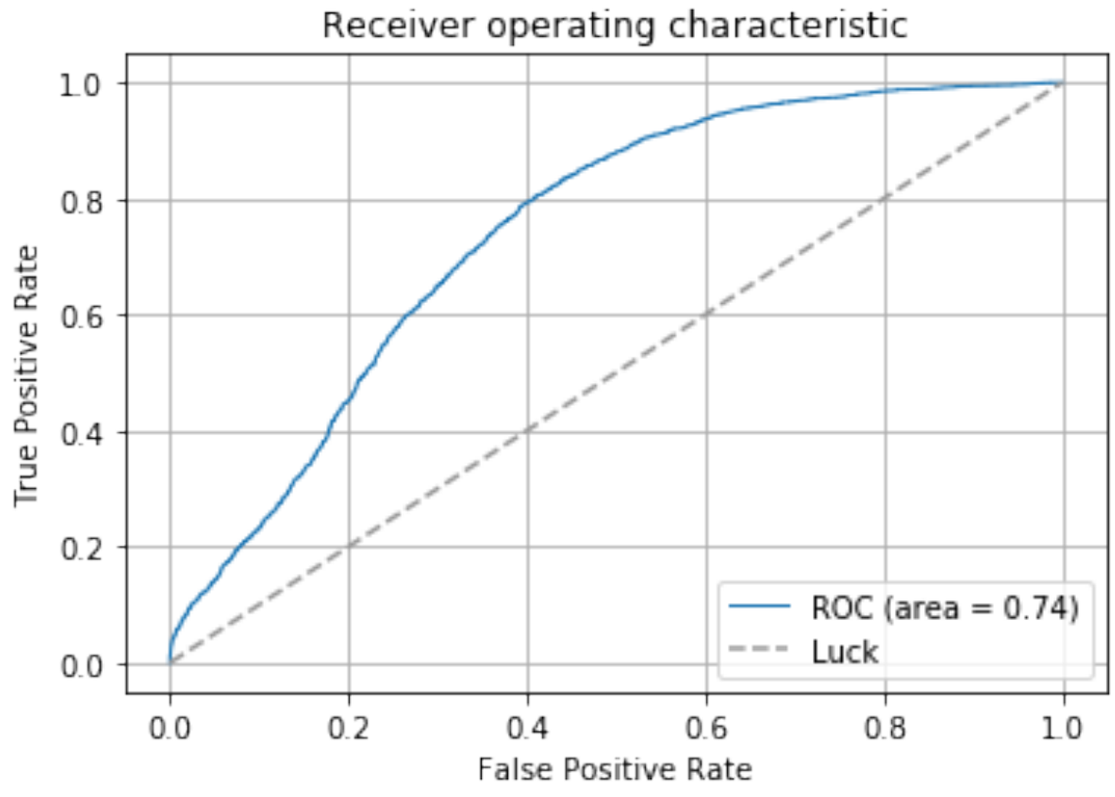
LCD Data Details

- 4 particle types, separate into directories. Needs to be mixed for training.
- Images:
  - ECAL: 25x25x25 cell section of calorimeter around particle.
  - HCAL: 5x5x60 cell section of calorimeter around particle.
- True Energy and PDG ID
- Features:
  - 'ECALMeasuredEnergy', 'ECALNumberOfHits', 'ECAL_ratioFirstLayerToTotalE', 'ECAL_ratioFirstLayerToSecondLayerE', 'ECALMoment1X', 'ECALMoment2X', 'ECALMoment3X', 'ECALMoment4X', 'ECALMoment5X', 'ECALMoment6X', 'ECALMoment1Y', 'ECALMoment2Y', 'ECALMoment3Y', 'ECALMoment4Y', 'ECALMoment5Y', 'ECALMoment6Y', 'ECALMoment1Z', 'ECALMoment2Z', 'ECALMoment3Z', 'ECALMoment4Z', 'ECALMoment5Z', 'ECALMoment6Z', 'ECAL_HCAL_ERatio', 'ECAL_HCAL_nHitsRatio'

DNN vs BDT

- The classification problem, as setup, ends up being very simple.
  - The real backgrounds are jets, not single particles.
  - V2 of dataset will address this shortcoming
- Comparison to BDT trained on features

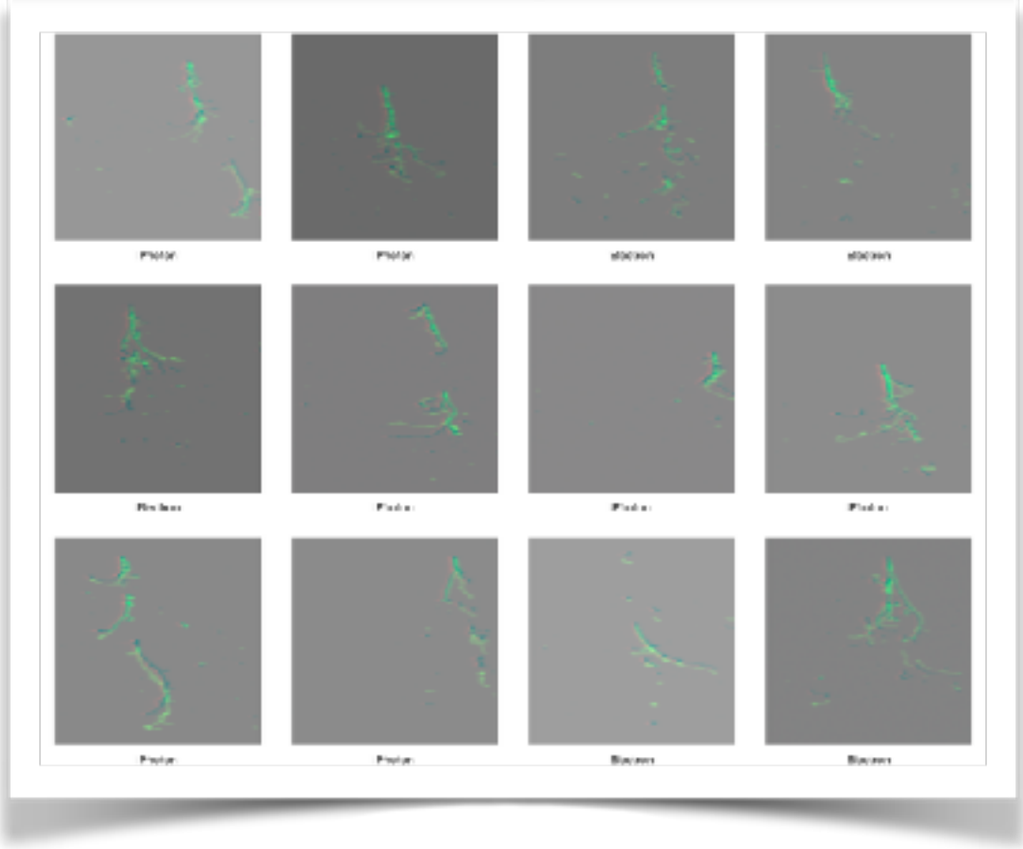


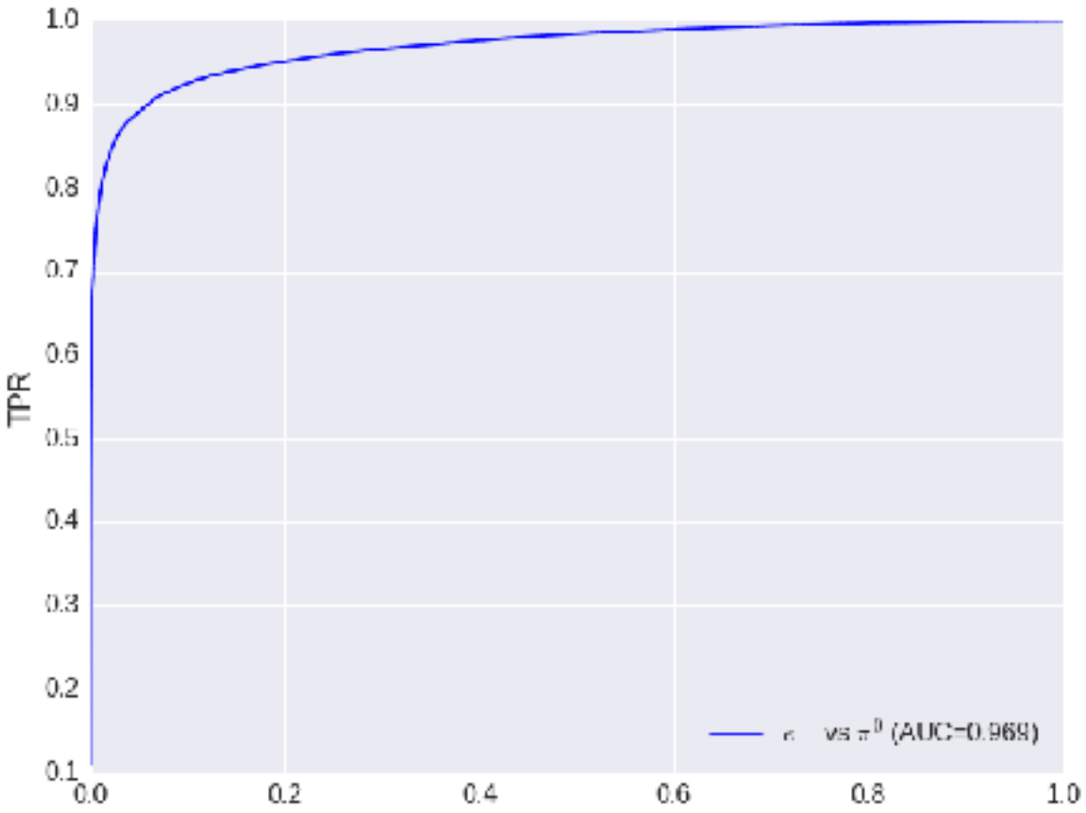


LCD Dataset Challenges/ Tasks

1. Classification
    - With existing setup, get excellent performance with simple DNN (not even a CNN).
    - It's not hard to get good performance here.
2. Energy Regression
    - Really just starting…
    - Interesting issues, e.g. accounting for known calorimetric resolution.
3. Generative Models
    - One of the primary challenges in HEP.
    - Early work very promising.
    - Sophia GAN work is based on LCD dataset…

LArTPC Dataset

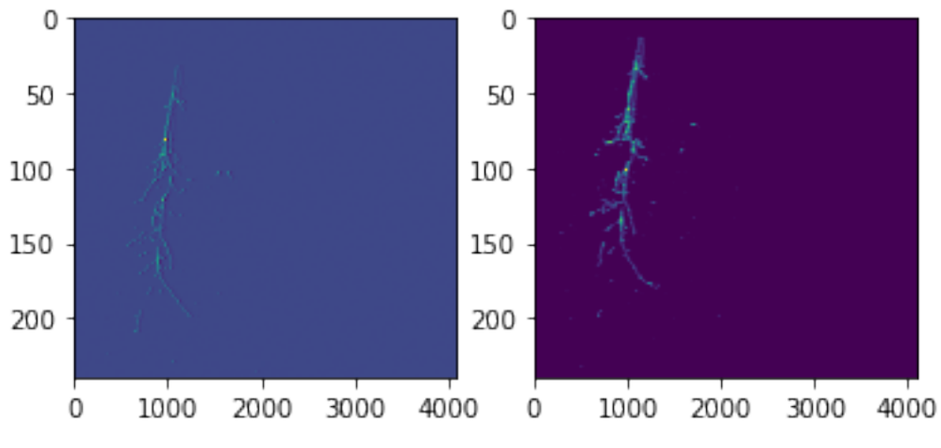


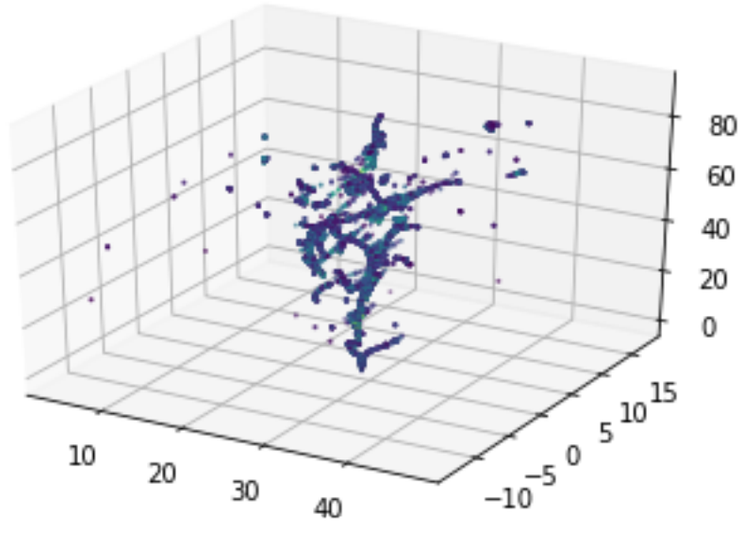


- Training samples have been at best ~100k examples…. usually much less.
- My students (S. Shahsavarani and G. Hilliard) simulated a huge sample of LArTPC events (LArIAT Detector).
  - Necessitated by Energy Regression studies.
  - 1 M of every particle species: $e^{\pm}$, $p^{\pm}$, $K^{\pm}$, $\pi^{\pm}$, π^0, $\mu^{\pm}$, γ, ν_e, ν_μ, ν_τ
  - Flat Energy distribution.
- Note that though this data is large, LArIAT is the smallest LArTPC detector with 2 x 240 wires.
  - DUNE will have 1 M wires.
- Have been working with P. Sadowski (UCI) to build inception-based CNN.

LArTPC Data Details

- 1 M of each particle type. Separate files for each files for each particle type.
  - For training they need to be mixed.
  - Images are large, so they are usually down-sampled.
  - Subset today… about 2.2 TB.
- Each “event” is two types of files:
  - 2D: LArTPC Reconstruction + True Info
    - images: (NEvents, 2, 240, 4096)
    - True: Energy, Px, Py, Pz,
    - Neutrino Truth: lep_mom_truth, nu_energy_truth, mode_truth
    - Track_length
  - 3D: Truth only
    - trajectory/C: x,y,z of charge deposits
    - trajectory/V: deposited charge

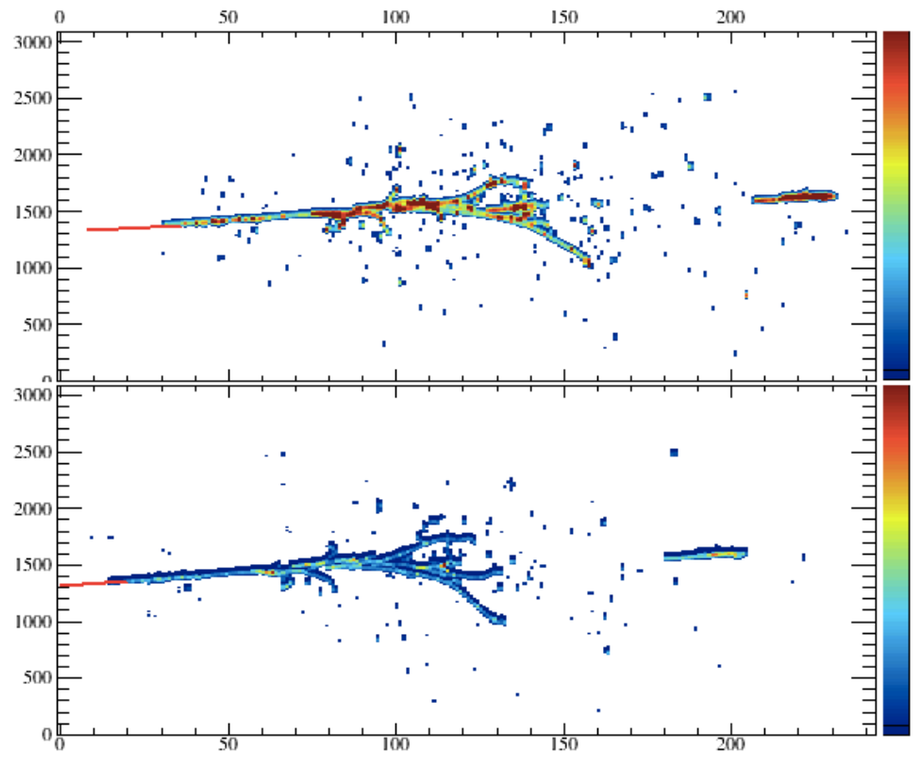




LArTPC Challenges/Tasks

1. Classification:

- Automatic reconstruction has proven to be very challenging
- CNNs have shown to perform better on classification… on down sampled data.
- Neither has achieved the performance assumed to be achievable for DUNE to achieve
  - Particles: ~90% efficiency, 1% fake
  - Neutrino: ~80% efficiency, 1% fake
- Scaling to full detector resolution (and 1M wire detector) is a challenge.

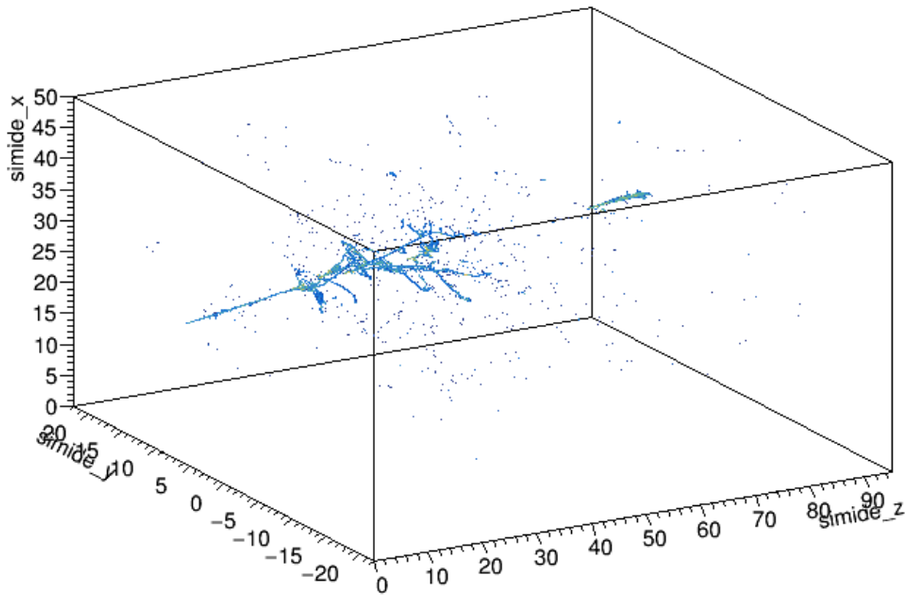


2. Energy Regression

- Our first attempts didn't give good result.
- Models should estimate error. Account for

3. 2D to 3D

- Full Pixelized readout would give ~ N^2 datapoint/time slice while wire readout give ~2N datapoint/time
- Information loss is "recovered" in reconstruction by assuming particle interaction topologies (track, shower, …)
- Tomographic approach (Wirecell) "resolves" ambiguities through costly Markov Chain MC
- Perhaps a DNN can learn the topologies and infer a 3D image… input to classification/regression?



4. Noise suppression / data compression / background filtering

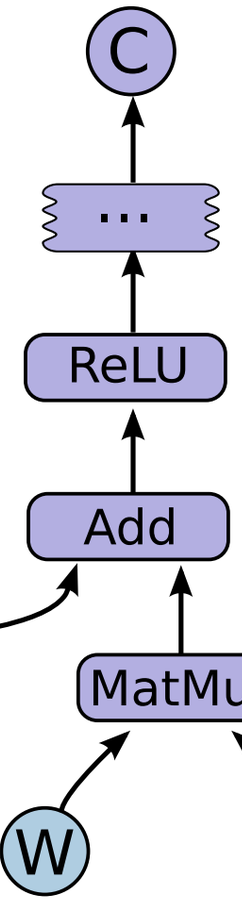
DL Software and Technical Challenges

numpy, Theano, Keras

- Numpy
    - Provides a tensor representation.
    - It's interface has been adopted by everyone.
        - e.g. HDF5, Then, TensorFlow, … all have their own tensors.
        - You can use other tensors, for the most part interchangeably with numpy.
    - Provides extensive library of tensor operations.
        - D = A * B + C, immediately computes the product of A and B matrices, and then computes the sum with C.
- Theano
    - Allows you write tensor expressions symbolically.
        - A * B + C is an expression.
    - Compiles the expression into fast executing code on CPU/GPU: F(A,B,C)
    - You apply the Compiled function to data get at a result.
        - D=F(A,B,C)
- Keras
    - Neutral Networks can be written as a Tensor mathematical expression.
    - Keras writes the expression for you.

DNN Software

- Basic steps
  - Prepare data
  - Build Model
  - Define Cost/Loss Function
  - Run training (most commonly Gradient Decent)
  - Assess performance.
  - Run lots of experiments…

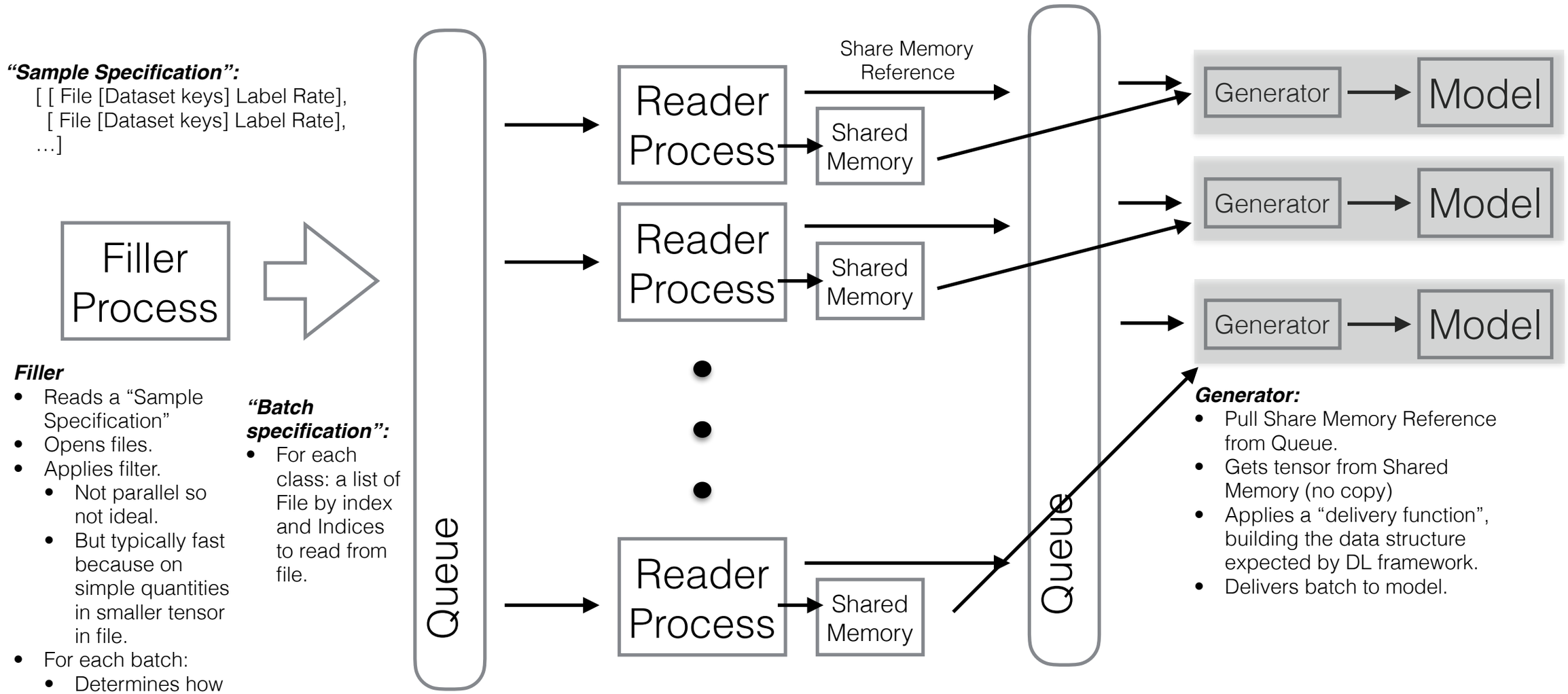


- 2 Classes of DNN Software: (Both build everything at runtime)
  - Hep-Framework-Like: e.g. Torch, Caffe, …
    - C++ Layers (i.e. Algorithms) steered/configured via interpreted script:
  - General Computation Frameworks: Theano and TensorFlow
    - Everything build by building mathematical expression for Model, Loss, Training from primitive ops on Tensors
    - Symbolic derivatives for the Gradient Decent
    - Builds Directed Acyclic Graph of the computation, performs optimizations
    - Theano-based High-level tools make this look like HEP Frameworks (e.g. pylearn2, Lasagna, Keras, …)

Technical Challenges

- Datasets are too large to fit in memory.
- Data comes as many h5 files, each containing O(1000) events, organized into directories by particle type.
- For training, data needs to be read, mixed, “labeled”, possibly augmented, and normalized…. can be time consuming.
- Very difficult to keep the GPU fed with data. GPU utilization often < 10%, rarely > 50%.
- Keras python multi-process generator mechanism has limitations…

Data Providers



Filler

- Reads a "Sample Specification"
- Opens files.
- Applies filter.
  - Not parallel so not ideal.
  - But typically fast because on simple quantities in smaller tensor in file.
- For each batch:
  - Determines how many events to read from each file.
- Generates a "Batch Specification"

"Batch specification":

- For each class: a list of File by index and Indices to read from file.

Readers:

- Fetch Data From files.
- Label and shuffle.
- Apply a process function
- May produce completely different tensors.
- Caches File handles

Shared Memory:

- Store Tensors
- Shape known until first batch comes through

Generator:

- Pull Share Memory Reference from Queue.
- Gets tensor from Shared Memory (no copy)
- Applies a "delivery function", building the data structure expected by DL framework.
- Delivers batch to model.

- Processing happens on First Epoch Only!
- Data Cached into file during first epoch.
  - All of the processing only needs to be done once.
- Automatically use another instance of data providers to read cache file for all other epochs.
- Note that for now, we have both train and test data providers… better use of resources if we merge.

```
acc', 'All_History.loss', 'All_History.val_loss', 'All_History.acc', 'All_Depth']
```

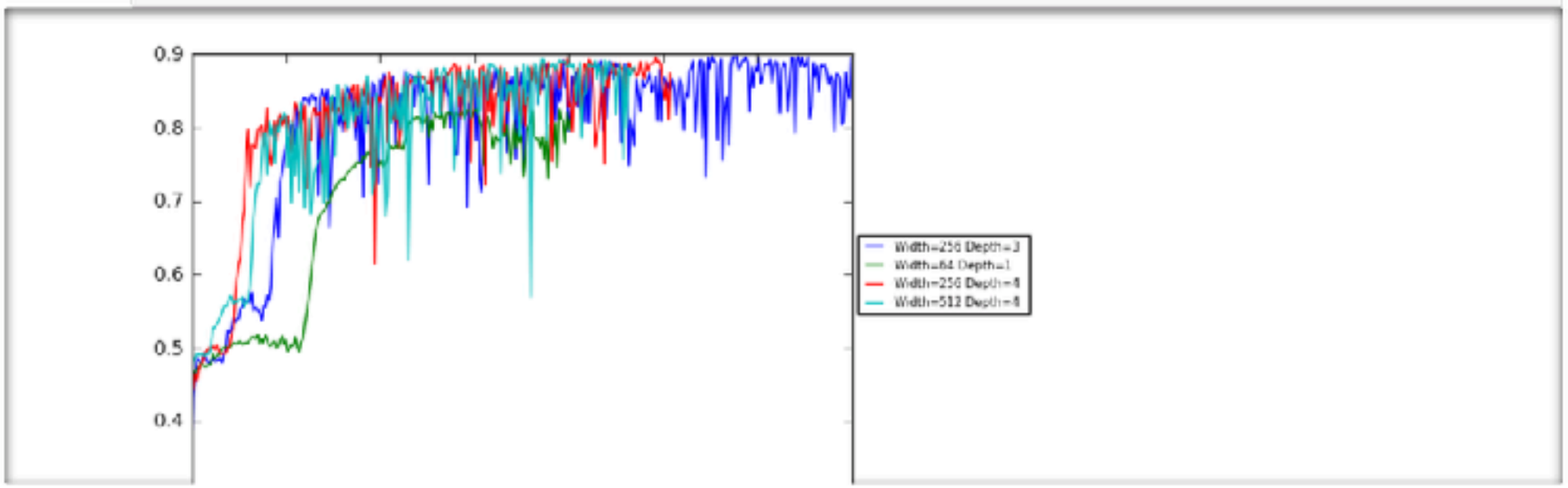
In [4]:
```
# Get a List of all numbers stored in MetaData
print "Available Parameters:", GetGoodParams(MyModels)
```

```
Available Parameters: ['Ele_AUC', 'Width', 'Depth', 'Pi0_AUC', 'Epochs', 'Gamma_AUC', 'ChPi_AUC']
```

In [5]:
```
# Make a Table of all relevant parameters, sort by 1,2,then 0 columns.
# Note: Parameters are optional... but the columns and rows will be not optimally sorted.
ScanTable(MyModels,['Model Name', 'Width', 'Depth', 'Epochs', 'Ele_AUC',  'Pi0_AUC', 'ChPi_AUC',  'Gamma_AUC'],[1,2,0])
```

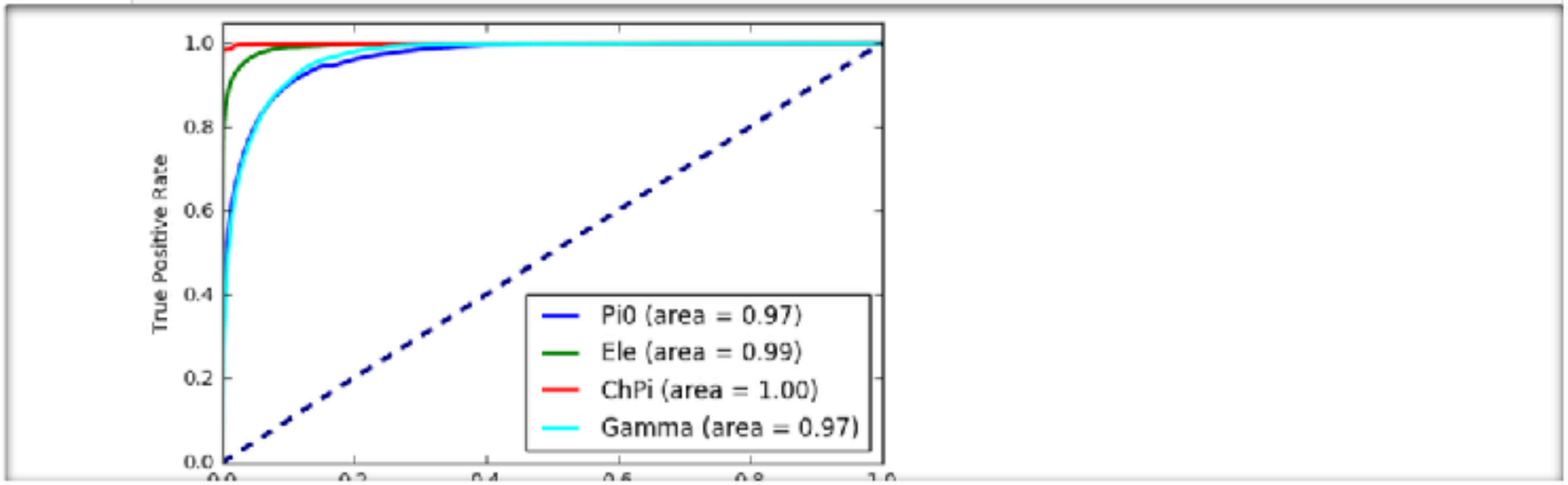
Model Name	Width	Depth	Epochs	Ele_AUC	Pi0_AUC	ChPi_AUC	Gamma_AUC
Width=32 Depth=1	32	1	228	0.9183	0.8657	0.9916	0.8833
Width=32 Depth=2	32	2	335	0.9364	0.8982	0.9885	0.8583
Width=32 Depth=3	32	3	298	0.9404	0.8979	0.9864	0.8572
Width=32 Depth=4	32	4	320	0.9139	0.8879	0.9518	0.8639
Width=64 Depth=1	64	1	251	0.9262	0.8712	0.9961	0.9022
Width=64 Depth=2	64	2	304	0.9320	0.9015	0.9887	0.9078
Width=64 Depth=3	64	3	432	0.9388	0.9164	0.9922	0.8186
Width=64 Depth=4	64	4	339	0.9808	0.9372	0.9983	0.9414
Width=128 Depth=1	128	1	342	0.9715	0.9154	0.9966	0.9357
Width=128 Depth=2	128	2	213	0.9500	0.8650	0.9956	0.9083
Width=128 Depth=3	128	3	318	0.9627	0.9322	0.9934	0.9261
Width=128 Depth=4	128	4	450	0.9879	0.9198	0.9984	0.9335
Width=256 Depth=1	256	1	305	0.9783	0.9191	0.9978	0.9436
Width=256 Depth=2	256	2	365	0.9473	0.9199	0.9913	0.9103
Width=256 Depth=3	256	3	437	0.9798	0.9544	0.9969	0.9570
Width=256 Depth=4	256	4	294	0.9276	0.9025	0.9859	0.9034
Width=512 Depth=1	512	1	292	0.9397	0.8777	0.9858	0.9067
Width=512 Depth=2	512	2	325	0.9588	0.9355	0.9869	0.9224
Width=512 Depth=3	512	3	289	0.9762	0.9339	0.9972	0.9195
Width=512 Depth=4	512	4	308	0.9349	0.8710	0.9921	0.8861

In [6]:
```
# Plot Historical MetaData... put 4 models per plot
#PlotMetaDataMany(MyModels,4,["History","val_loss"],loc="center left")
PlotMetaDataMany(MyModels,4,["All_History.val_acc"],loc="center left")
```



In [7]:
```
# Compare Number of Epochs each model ran (only last run)
PlotMetaData(MyModels,['Epochs'])
```

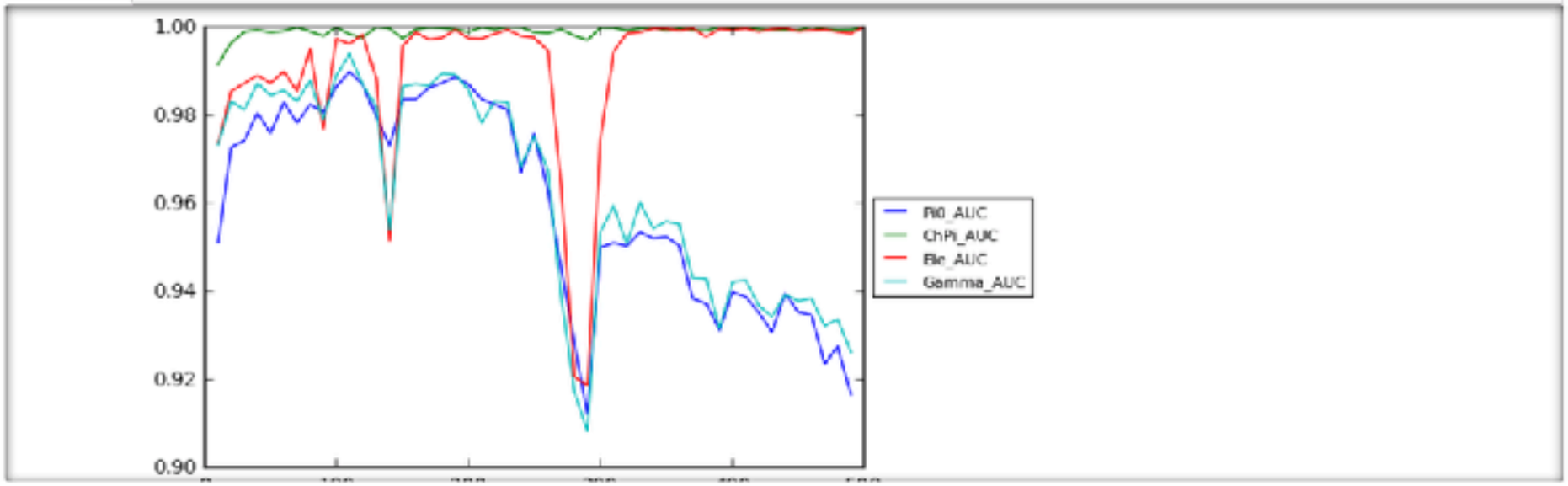
```
# Also performs inference on the test data, returning the results
from DLAnalysis.Classification import *
result,NewMetaData=MultiClassificationAnalysis(MyModel,[Test_X_ECAL,Test_X_HCAL],Test_Y,BatchSize,
                                               IndexMap={0:'Pi0', 2:'ChPi', 3:'Gamma', 1:'Ele'})
```



In [4]:
```
# Bin the data
Energy=target[:,:,2].flatten()

def AUCvsEnergy(E_min=10.,E_max=510.,E_bins=100.):
    BD,E_binning=BinDataIndex(Energy, E_min, E_max, E_bins)
    # Run the Classification Analysis in Bins
    return BinMultiClassificationAnalysis(MyModel,Test_Y=Test_Y,Y_binning=E_binning,
                                          bin_indecies=BD, result=result,
                                          IndexMap={0:'Pi0', 2:'ChPi', 3:'Gamma', 1:'Ele'})
```

In [5]:
```
# Full Energy Range
Res=AUCvsEnergy(10.,510.,50.)
```



In [6]:
```
# 10 to 100 GeV
```

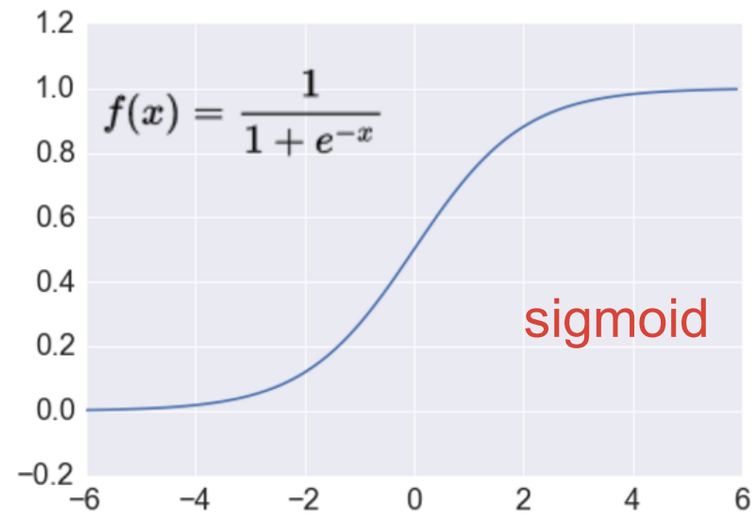
DL Model Components

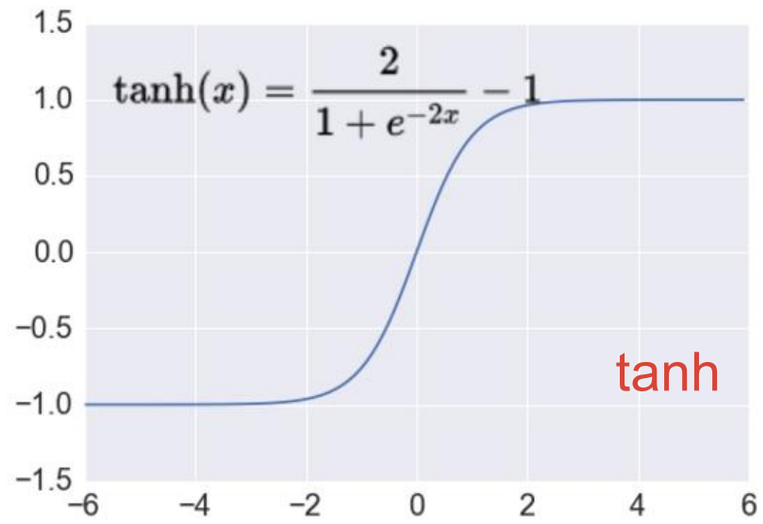
Keras

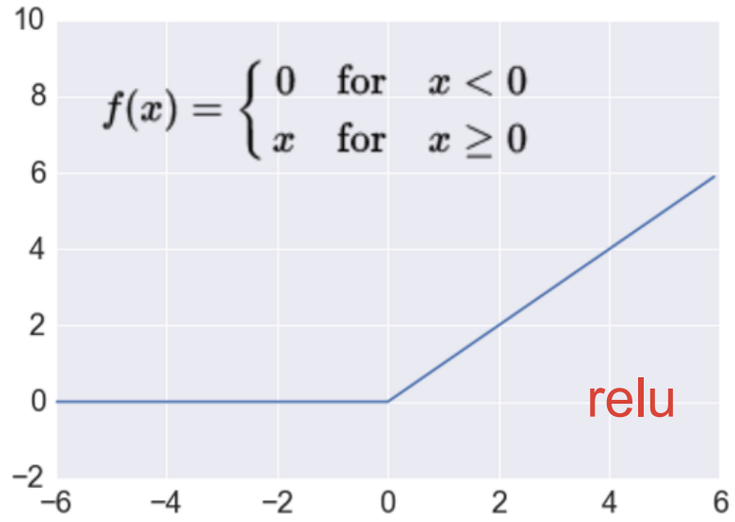
https://keras.io/

Activations

- Vanishing Gradients
- Sigmoid
  - Saturate
  - non-Zero centered
- Relu
  - Larger Gradient
  - Simple to compute
  - If learning rate too high, neurons can “Die” (never activate)
- Leaky Relu





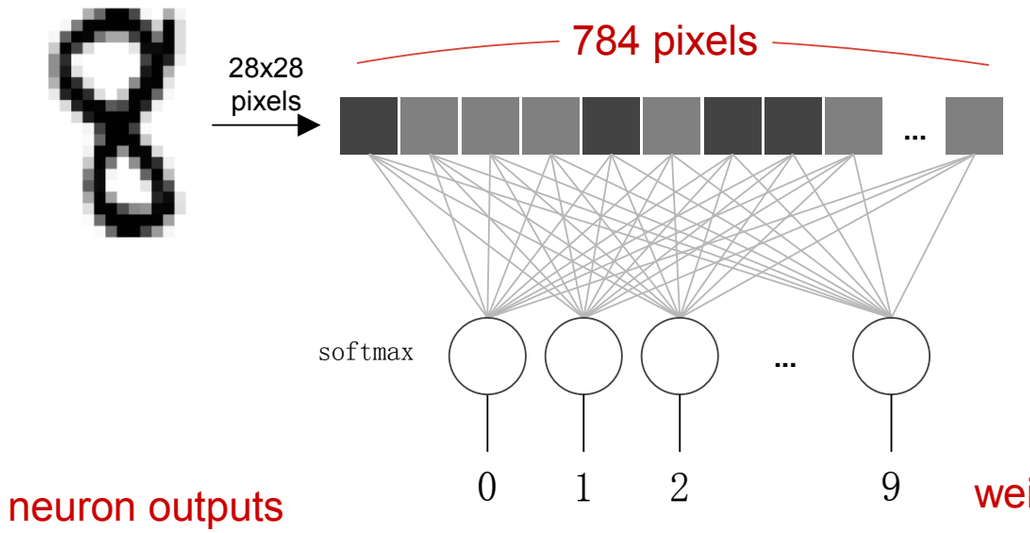


Output

- Classification: One-hot representation
- SoftMax
  - Boltzmann distribution: $e^{-E/kT}$
  - Takes vector of arbitrary values and maps to
    - vector with values in range 0 to 1
    - vector components sum to 1.
  - Uses:
    - Prob output from Multiclass classification
    - Normalization of data.

The softmax layer

➢ **The output from the softmax layer is a set of probability distribution, positive numbers which sum up to 1.**



neuron outputs

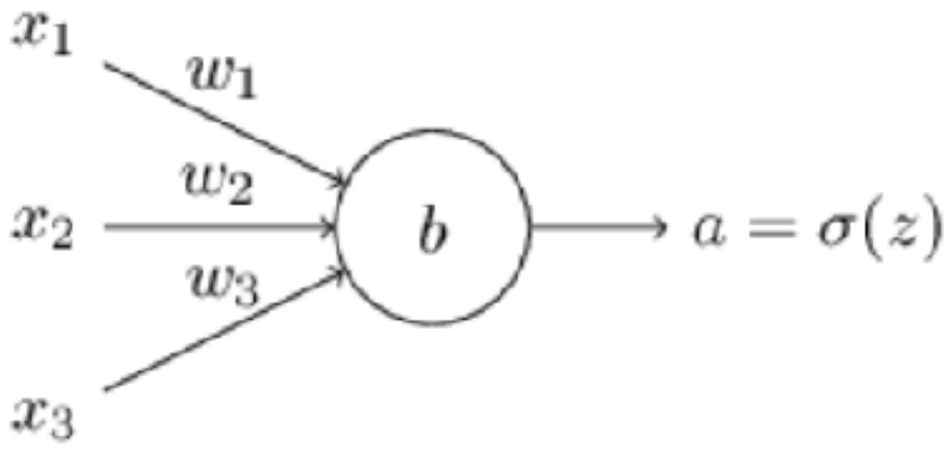
weighted sum of all pixels + biases

$$\text{softmax}(z_i) = \frac{\exp(z_i)}{\sum_j \exp(z_j)}$$

Cost/Loss

- MSE- Means square error.
    - Proven to give right y for given x.
- MSE-
    - Proven to give right median y for given x.
- Often train poorly: saturating outputs give small gradients.
    - Use cross-entropy for classification

Cost Functions



$$z = \sum_j w_j x_j + b$$

$$C = \frac{(y-a)^2}{2}$$

$$\frac{\partial C}{\partial w} = (a-y)\sigma'(z)x = a\sigma'(z)$$

$$\frac{\partial C}{\partial b} = (a-y)\sigma'(z) = a\sigma'(z),$$

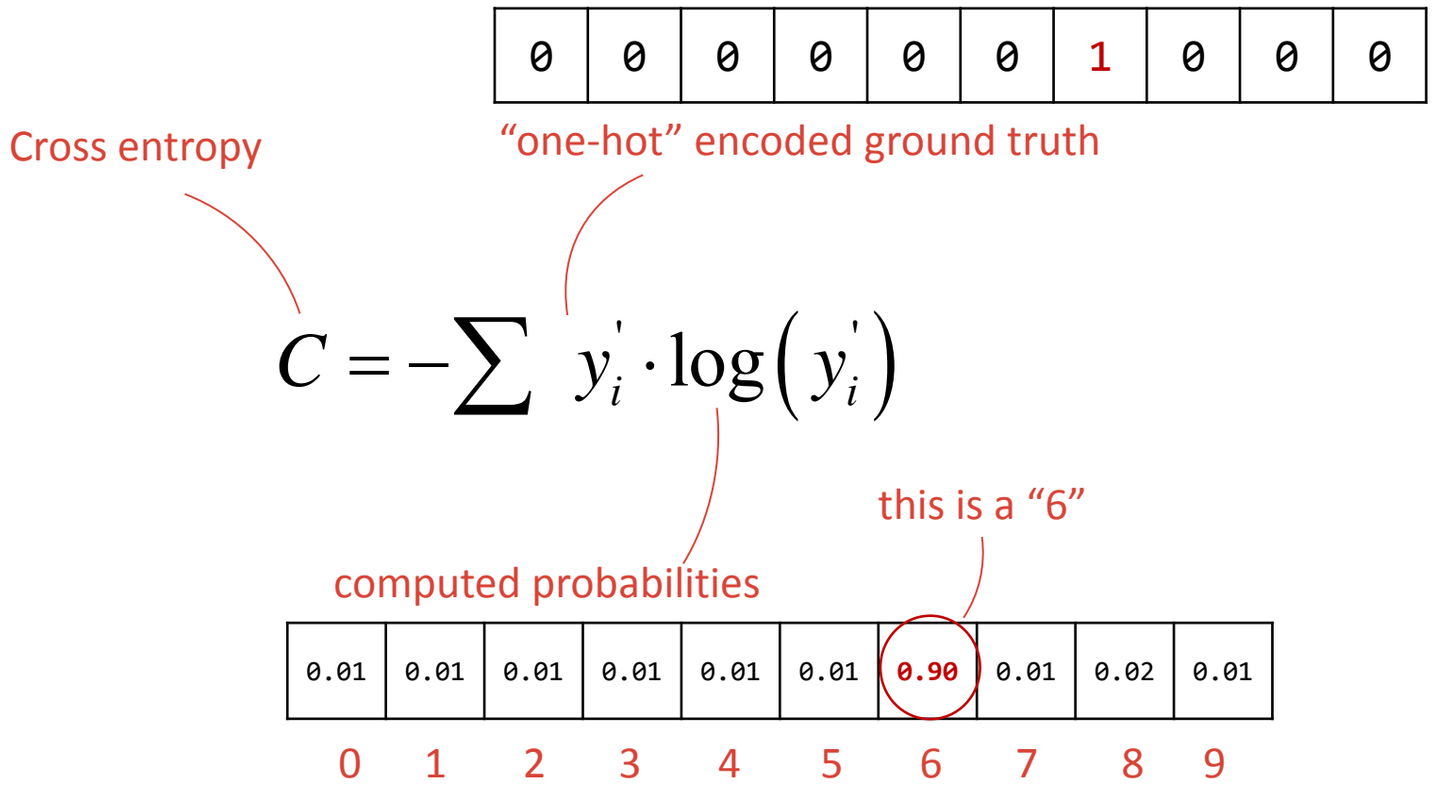
$$\frac{\partial C}{\partial w_j} = \frac{1}{n}\sum_x x_j(\sigma(z)-y).$$

$$C = -\frac{1}{n}\sum [y \ln a + (1-y)\ln(1-a)]$$

$$\frac{\partial C}{\partial b} = \frac{1}{n}\sum_x (\sigma(z)-y).$$

The Cross-Entropy Cost Function

- **For classification problems, the Cross-Entropy cost function works better than quadratic cost function.**
- **We define the cross-entropy cost function for the neural network by:**



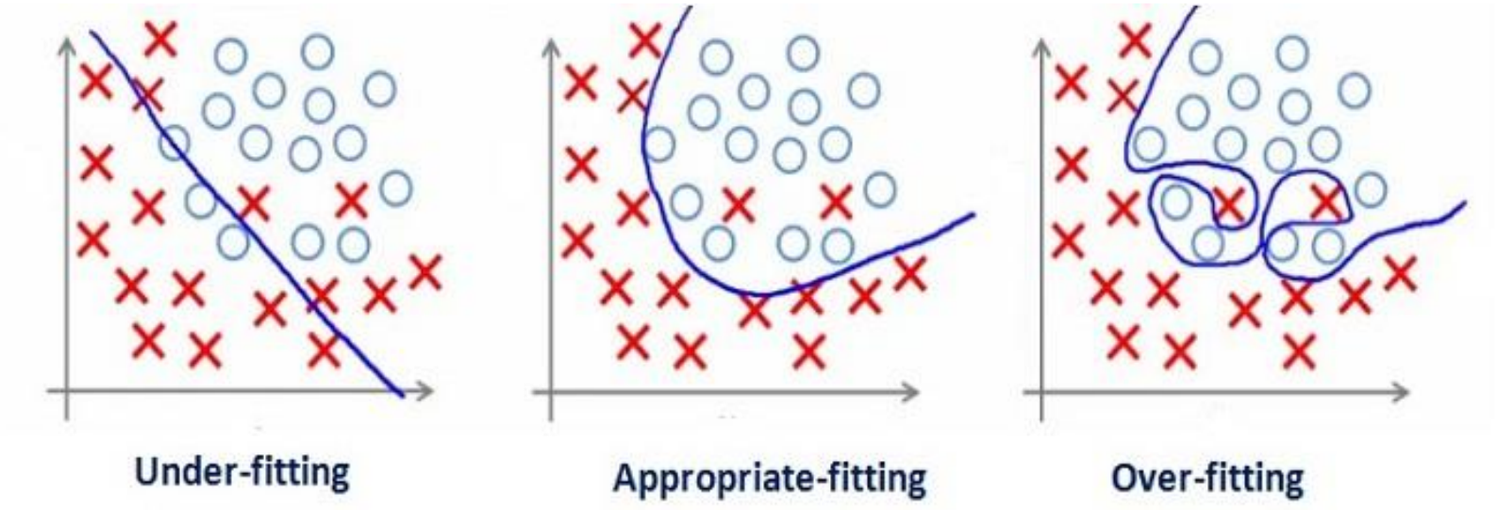
$$H(p, q) = -\sum_x p(x) \log q(x).$$

- In information theory, the cross entropy between two probability distributions p and q over the same underlying set of events measures the average number of bits needed to identify an event drawn from the set, if a coding scheme is used that is optimized for an "unnatural" probability distribution q rather than the "true" distribution p

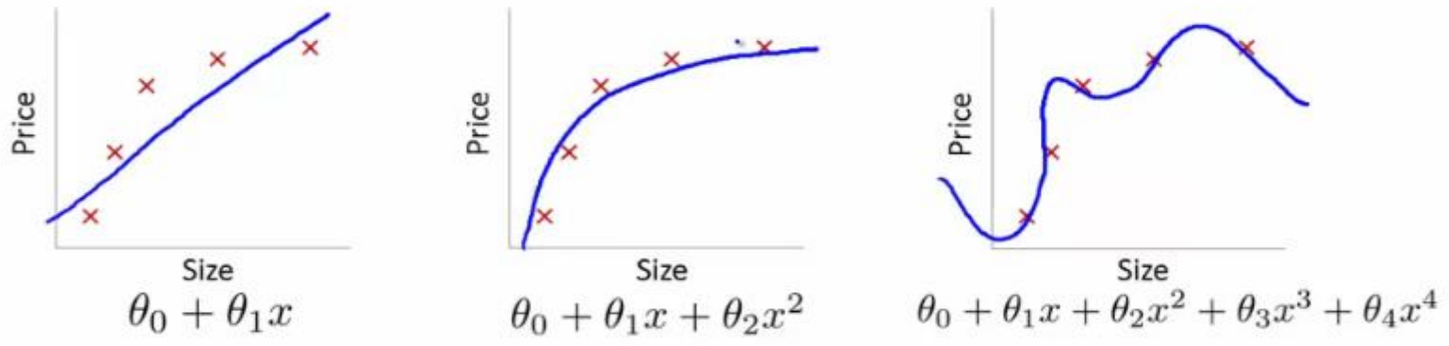
Over Fitting

- **Overfitting occurs when a model is excessively complex, such as having too many parameters relative to the number of observations. A model that has been overfit has poor predictive performance, as it overreacts to minor fluctuations in the training data.**

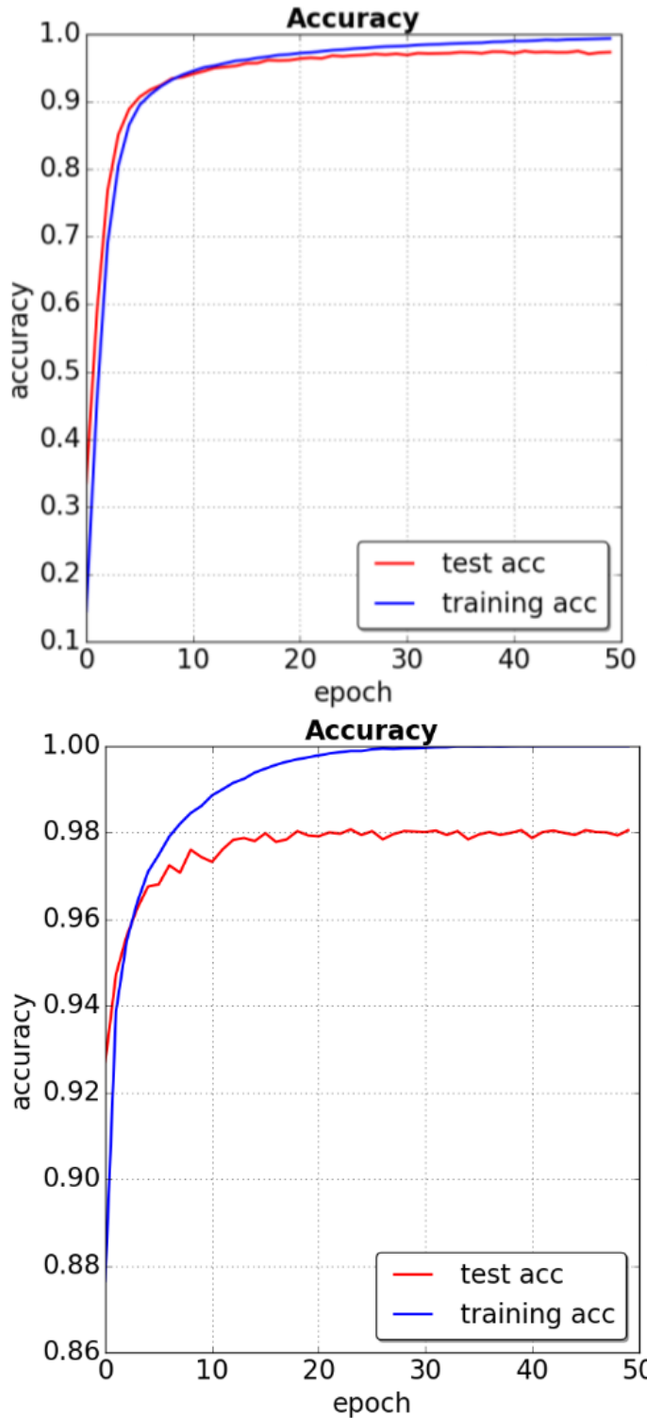
Classification:

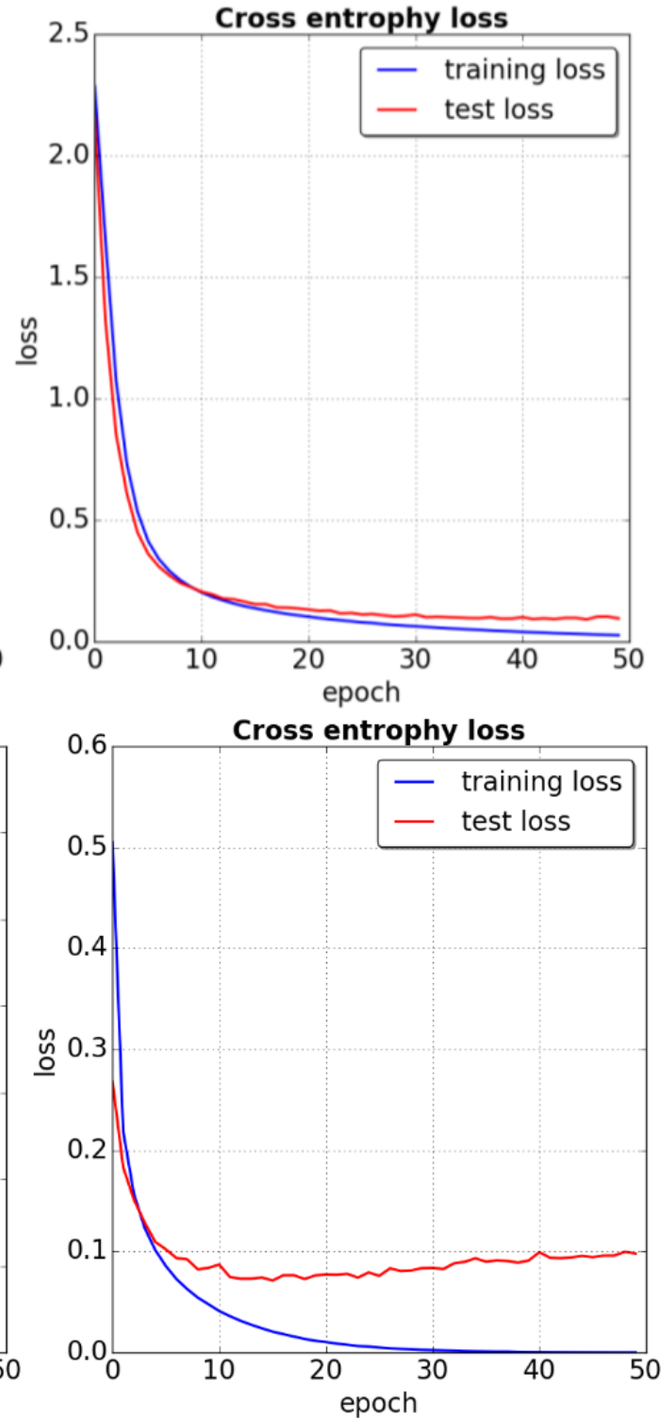


Regression:



High bias (underfit) | "Just right" | High variance (overfit)


Accuracy
accuracy
epoch
test acc
training acc
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
10
20
30
40
50

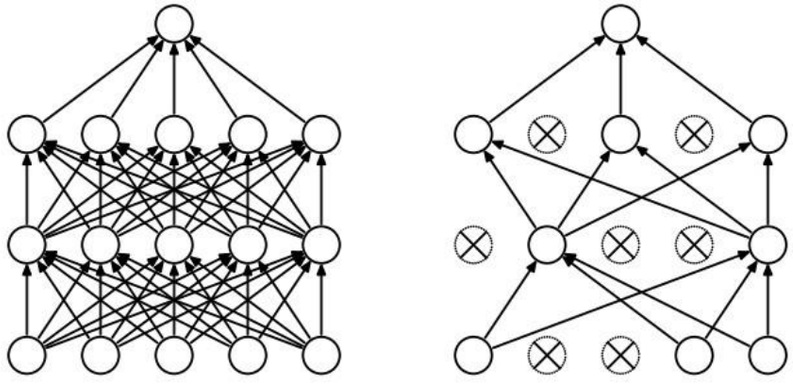

Cross entrophy loss
loss
epoch
training loss
test loss
2.5
2.0
1.5
1.0
0.5
0.0
0
10
20
30
40
50

Accuracy
accuracy
epoch
test acc
training acc
1.00
0.98
0.96
0.94
0.92
0.90
0.88
0.86
0
10
20
30
40
50

Cross entrophy loss
loss
epoch
training loss
test loss
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0
10
20
30
40
50

Regularization - Dropout

- **Dropout is an extremely effective, simple and recently introduced regularization technique by Srivastava et al (2014).**



(a) Standard Neural Net (b) After applying dropout.

- **While training, dropout is implemented by only keeping a neuron active with some probability p (a hyperparameter), or setting it to zero otherwise.**
- **It is quite simple to apply dropout in Keras.**

```
# apply a dropout rate 0.25 (drop 25% of the neurons)
model.add(Dropout(0.25))
```