Machine Learning in Particle Physics

Mike Williams
MIT

August 30, 2018

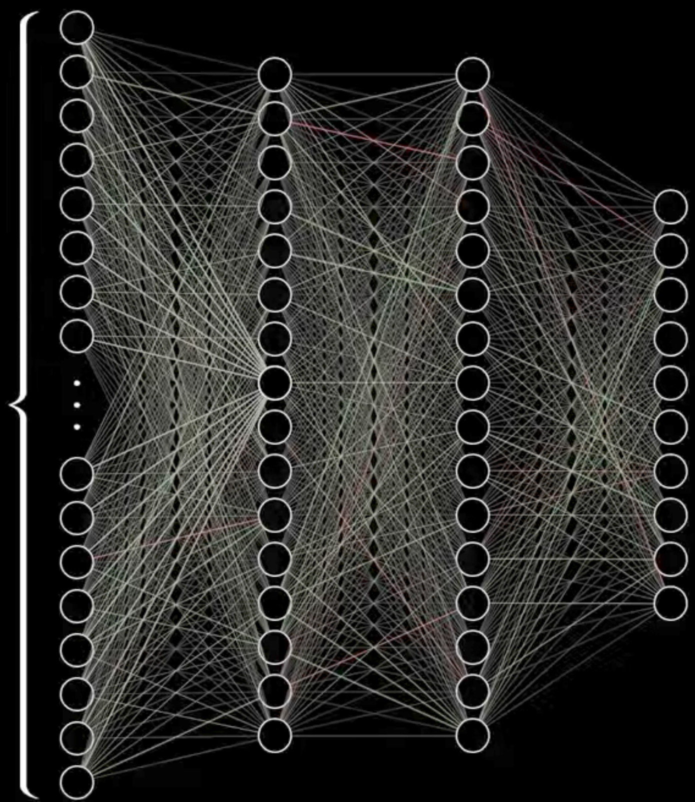
Largely inspired by a Nature review article written in collaboration with A Radovic, D Rousseau, M Kagan, D Bonacorsi, A Himmel, A Aurisano, K Terao, & T Wongjirad.

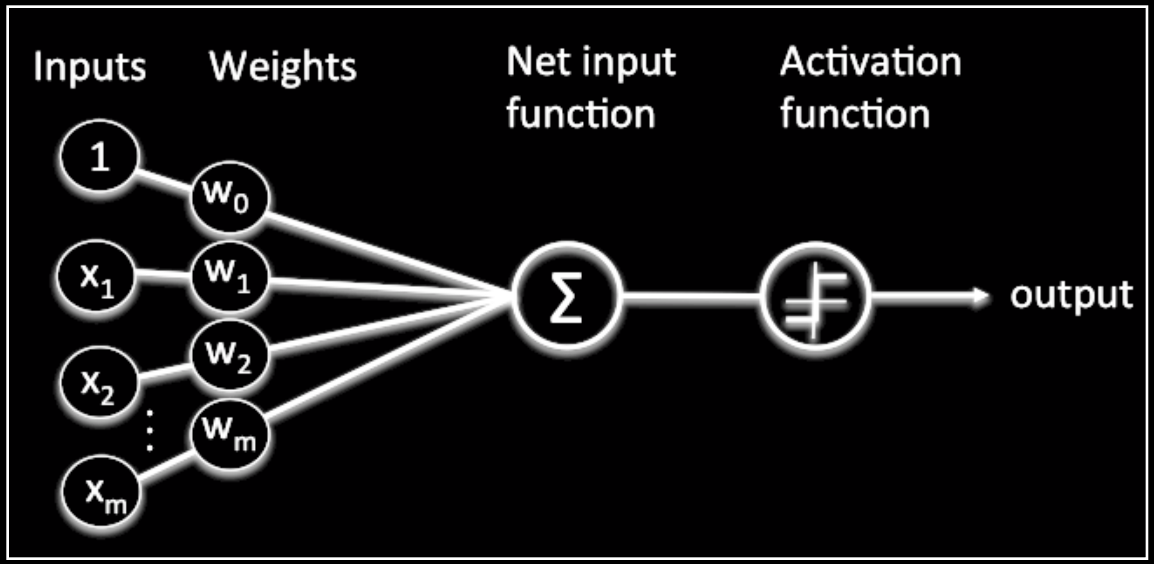
Review of Lecture 1

- For the case where PDFs are known, there is no need for ML — but it's rare to truly be in that situation.
- ML can be viewed as simply a large and complicated optimization problem. All of the same pitfalls—and solutions—to overfitting a less opaque model also apply to training ML algorithms. Hopefully these points were reinforced by the tutorial example.
- The use of ML has become ubiquitous in HEP. Many common classification and regression tasks already performed by ML-based algorithms, including in the real-time (trigger) event-classification systems.
- A key to ML usage is data-driven characterization of the performance, typically done using standard candle calibration samples. Of course, like anything else, you should always be skeptical—and use data to prove that things are working (to the desired CL).
- Physics-aware loss functions are a powerful way to use ML in situations where out-of-the-box algorithms fail.

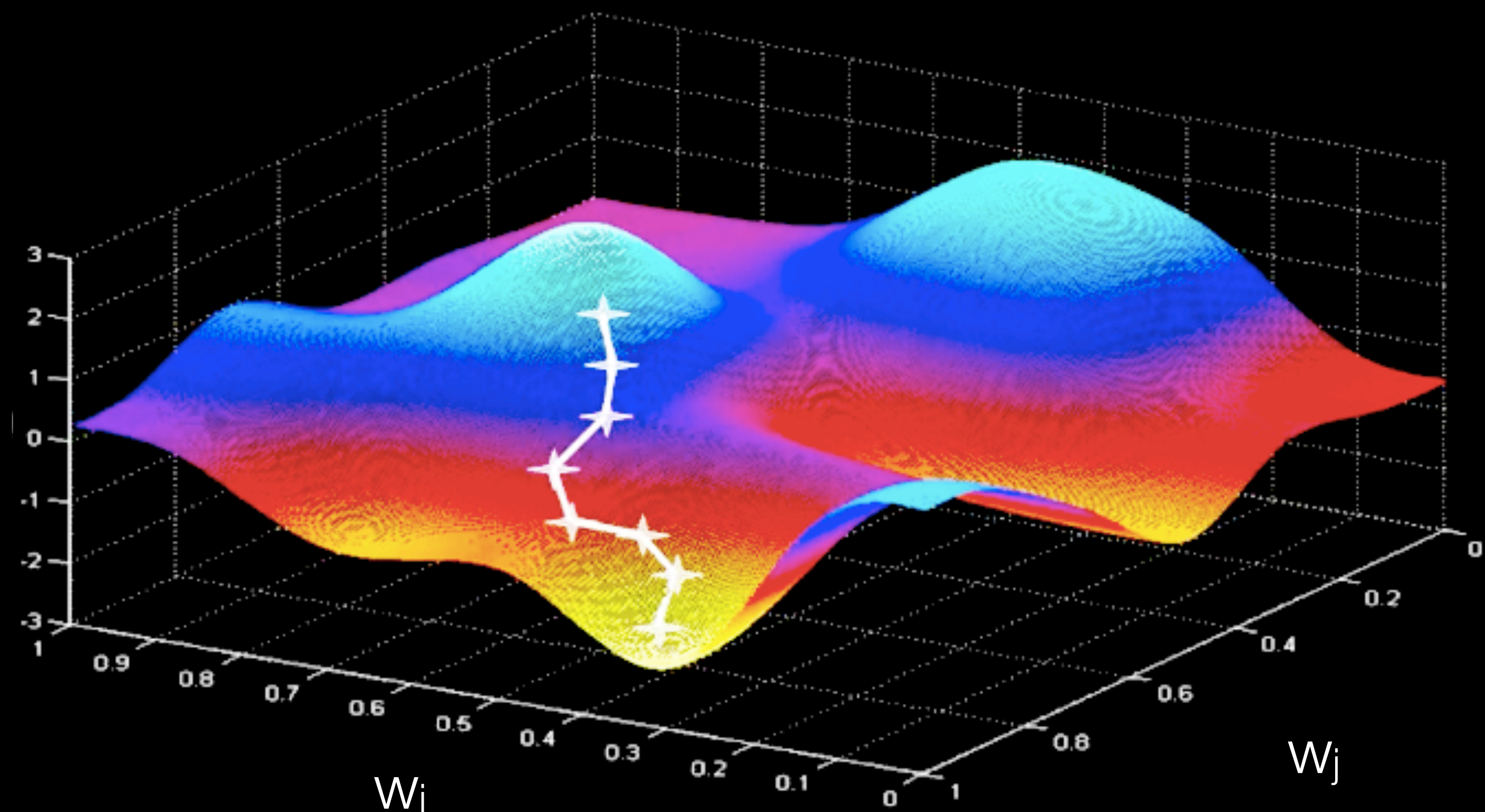
Review of Lecture 1

The NN is trained using gradient descent to minimize loss. The end result is just a function; it's simple to explore its behavior.



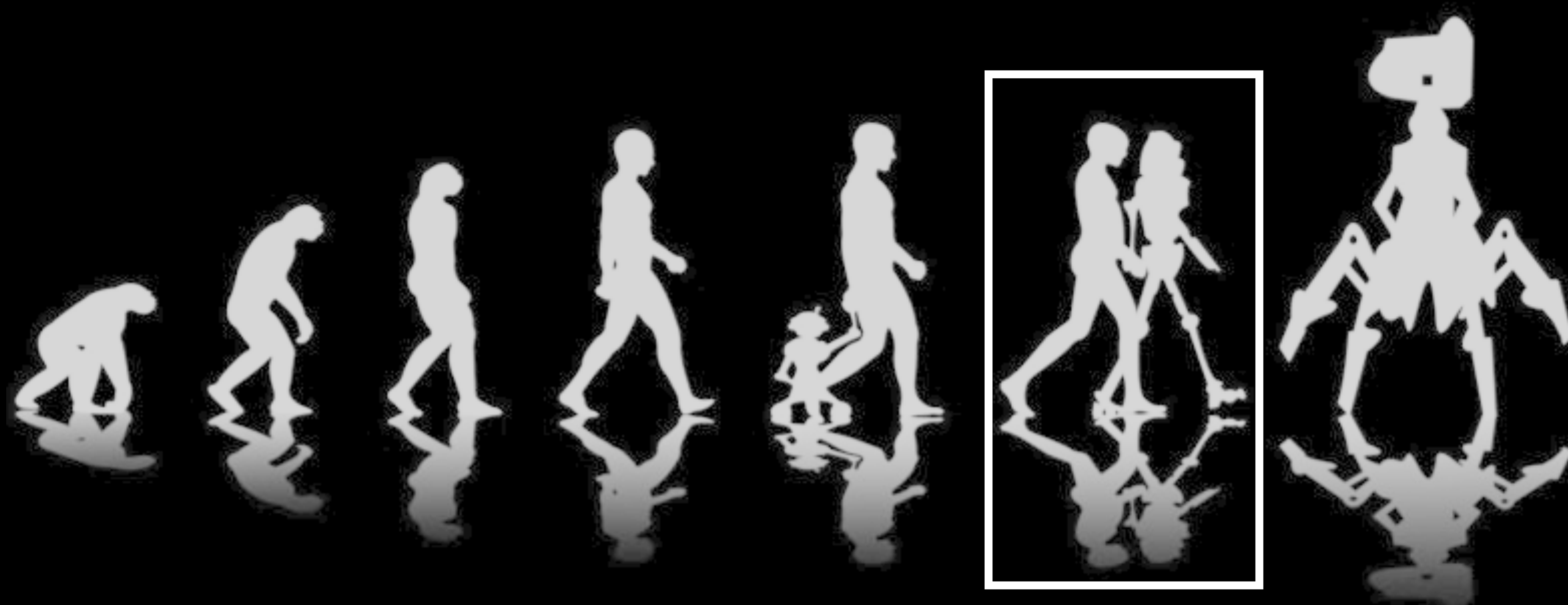


You can enforce desired behavior in the loss function, NN design, etc. E.g., LHCb's trigger ML is forced to asymptote its treatment of high-mass and long-lifetime candidates at the GeV and ps scales.



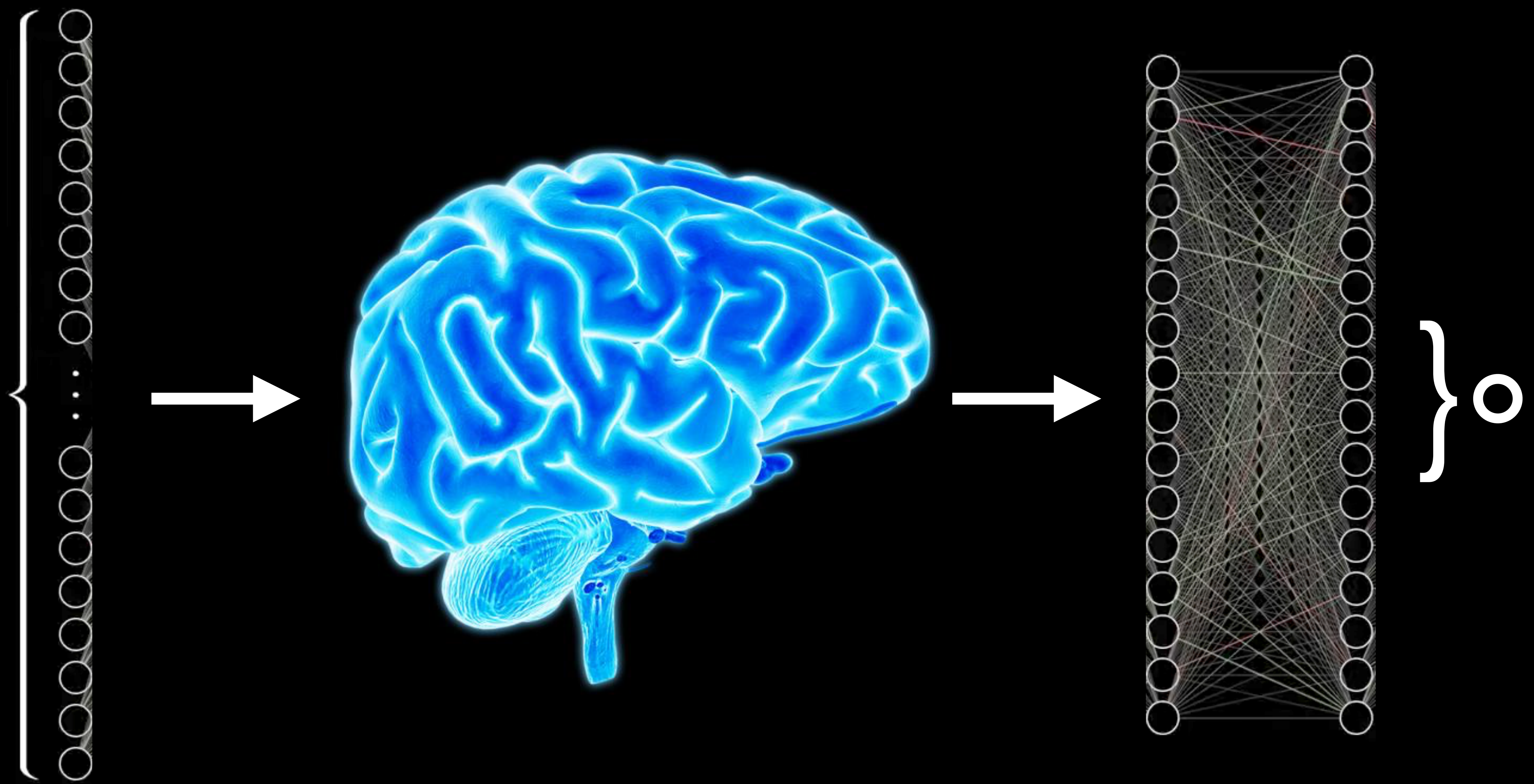
Deep Learning

Now let's look at (at least partially) skipping the feature-engineering step. How well can we do using deeper networks and/or special architectures?



Shallow Learning

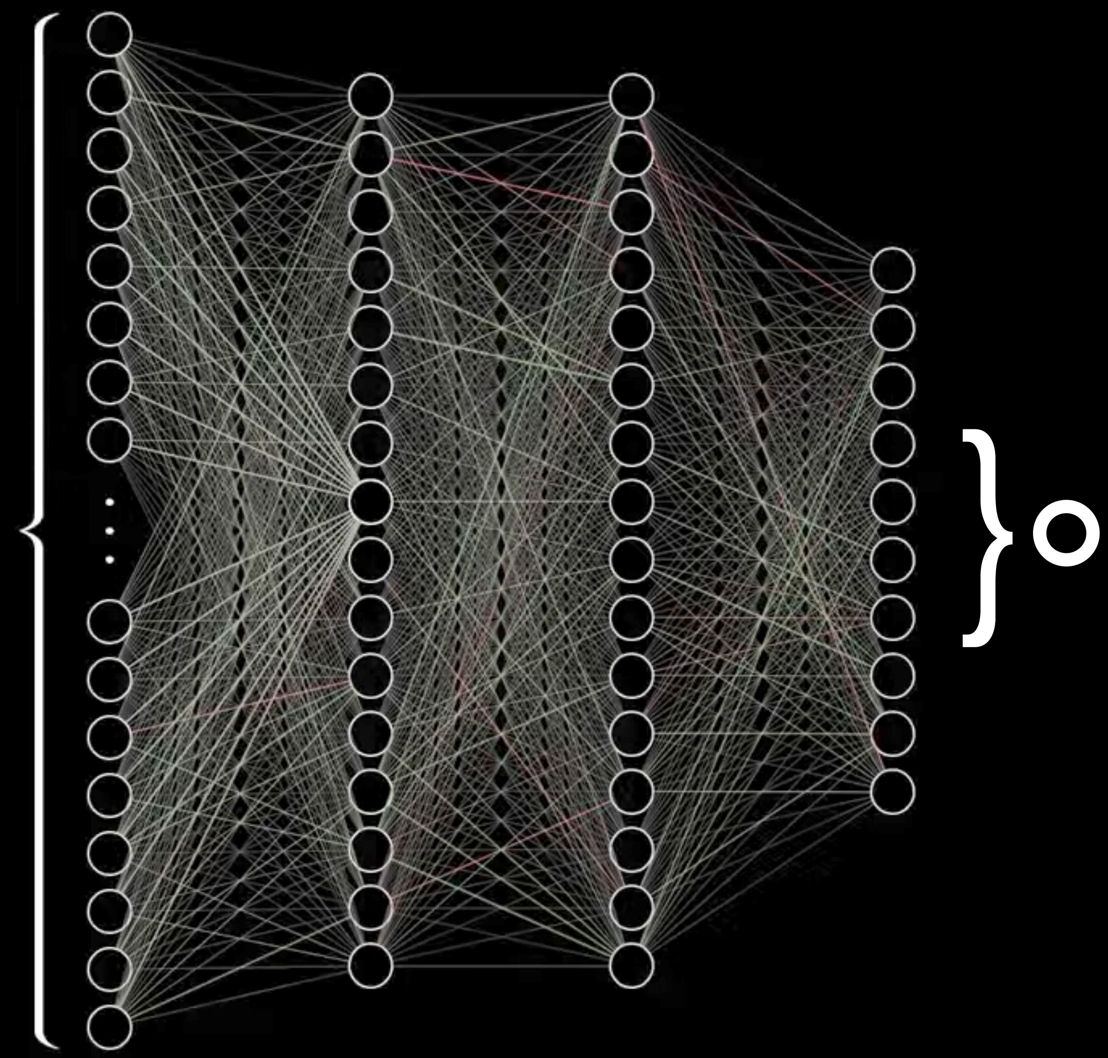
Yesterday we focused on using domain knowledge to engineer input features.



We physicists have the SM, detailed detector simulations, decades of successful experimentation, etc., so we do a good job of dimensional reduction in our heads.

Deep Learning

In principle, we could skip the human-led step; however, in practice, even the mass is a rather odd manifold in a high-dimensional feature space (i.e. difficult to learn).

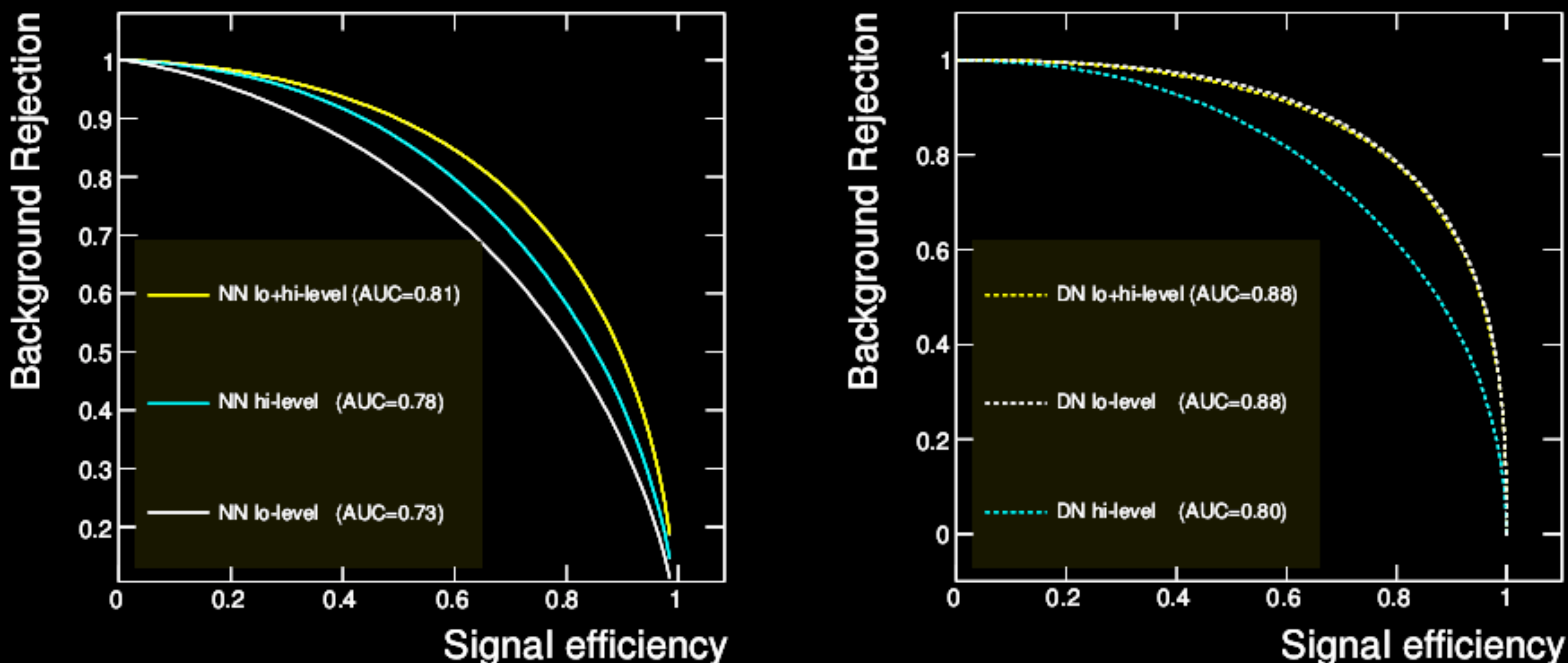


Adding more layers (making the network deeper) makes it possible, in principle, for the machine to learn to form complicated features from low-level unorganized inputs — but to what extent does this work? What architectures are best suited to what types of learning?

Deep Learning vs Deep Thinking

Does a deep NN really need our help? Does it need high-level features like invariant masses, or can it just learn the physics itself from the 4-vectors (given examples)?

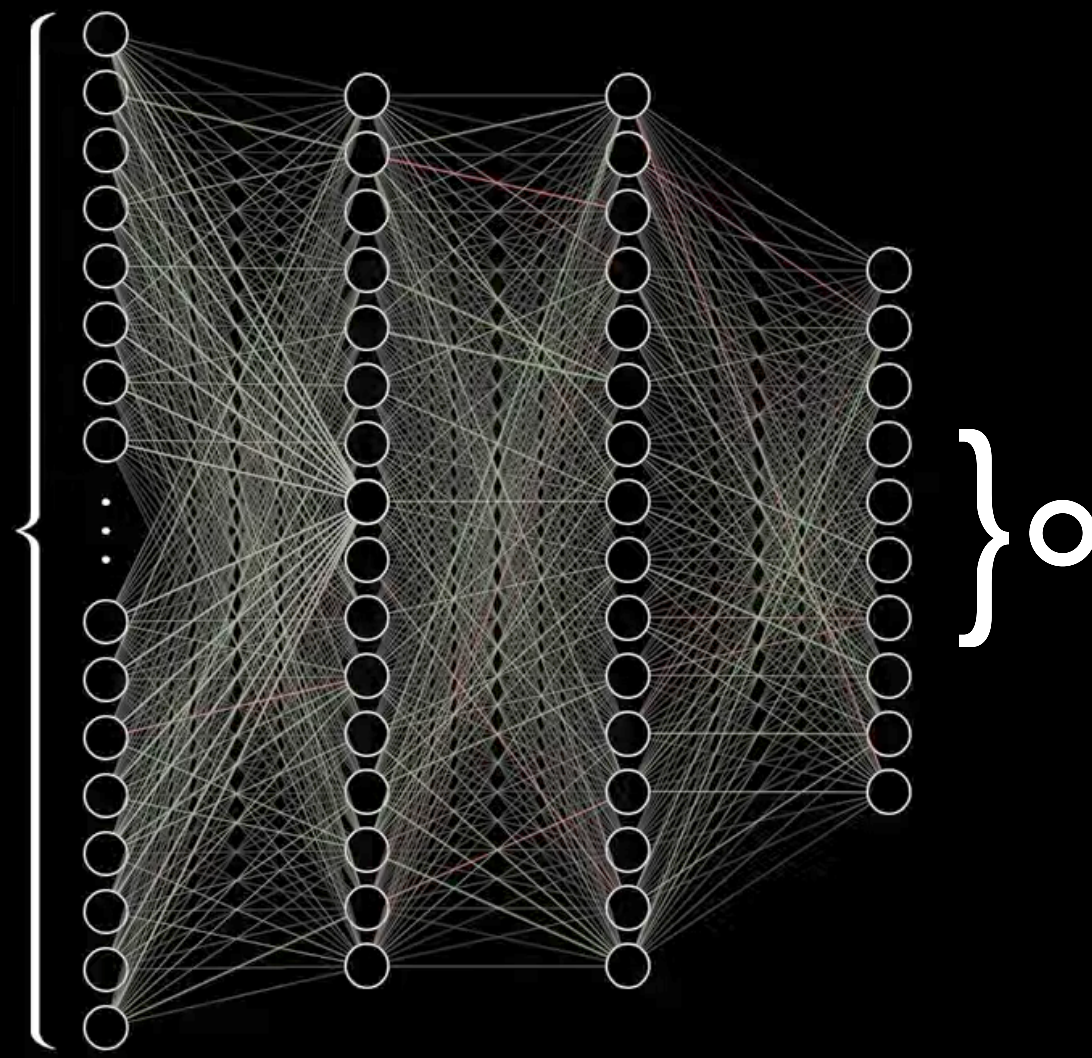
Baldi, Sadowski, Whiteson [1402.4735]



The DNN is able to learn all that it needs in this case, as providing high-level features results in no gains — in fact, the DNN using low-level features outperforms any selection based only on high-level features.

Latent Spaces

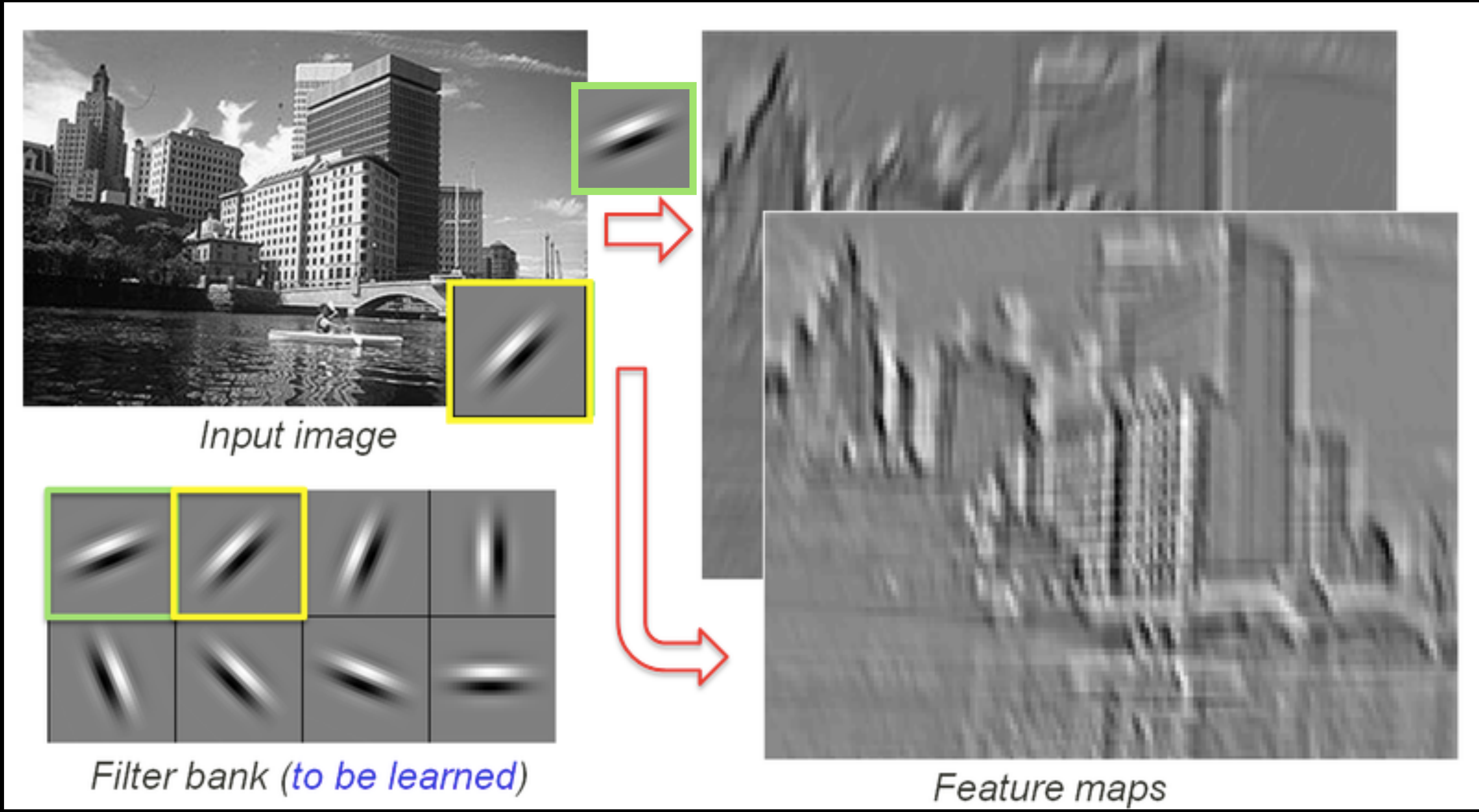
Each hidden layer in a DNN can be viewed as a latent-space representation of the data, rather than just some intermediate step in a complicated matrix multiplication.



From this perspective, we are now trying to design the algorithm to learn useful/meaningful latent spaces—which is what human-led feature engineering does, and what science does in general! (We don't just use data to learn to improve task performance, we try and gain knowledge and understanding about nature from it.)

Convolutional Neural Networks

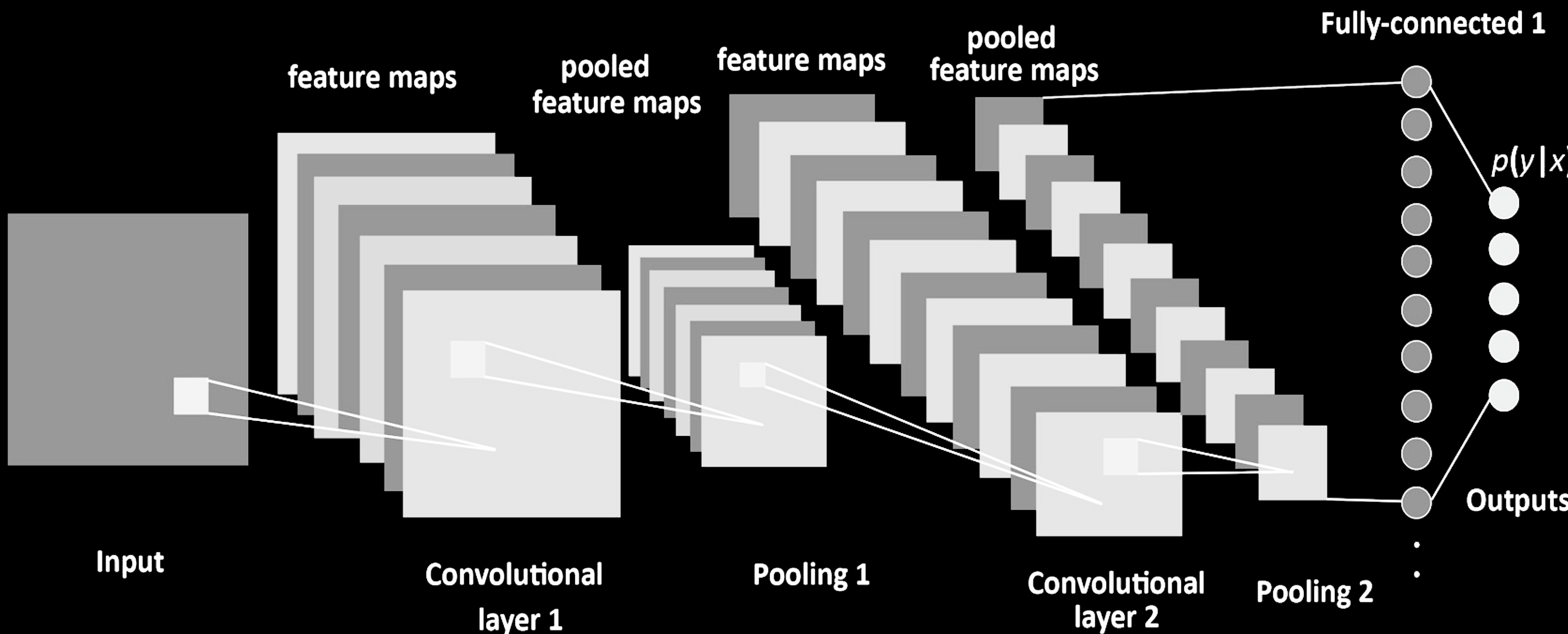
CNNs are deep feed-forward NNs whose architecture was inspired by the visual cortex. They have been used to solve a variety of problems, including many in image-recognition.



The neurons in a CNN look for local examples of translationally invariant features. This is done using convolutional filters to locate patterns producing maps of simple features, then build complex features using many layers of simple feature maps.

Convolutional Neural Networks

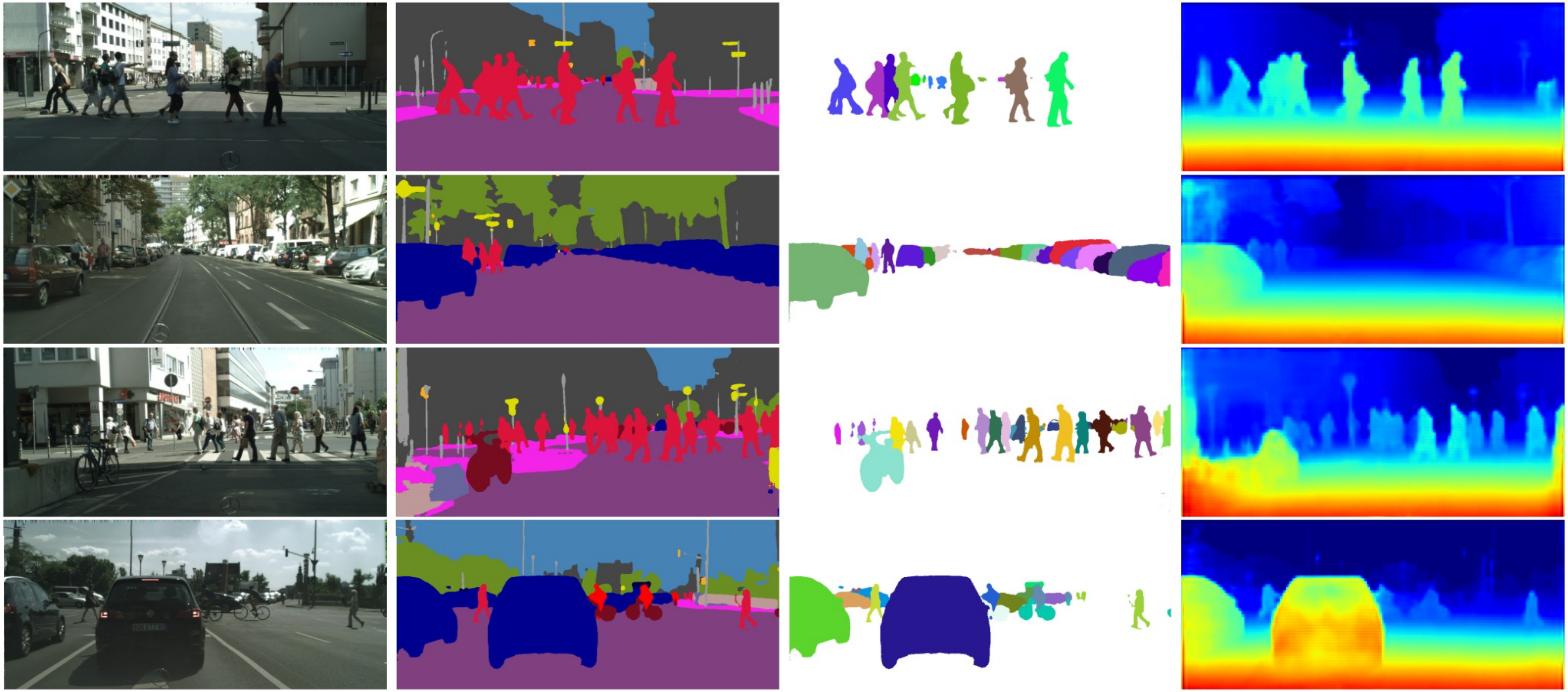
The use of reduced receptive fields for the neurons—who also share weights—in the early layers provides both translational invariance and makes training a tractable problem.



A major advantage of CNNs is that they learn the convolution filters themselves, i.e. these do not need to be provided (known a priori). Notice how this explicitly looks like dimensional reduction via feature extraction followed by a simple fully connected optimization function.

Convolutional Neural Networks

CNNs have demonstrated super-human performance in many computer-vision tasks.



(a) Input image (b) Segmentation output (c) Instance output (d) Depth output

The ability to not just classify an entire image but to segment the pixels into different sources should immediately sound useful for HEP reconstruction tasks.

CNNs for Neutrinos

The first published measurement in HEP to use a CNN is from NOvA [1703.03328].

Neutrino experiments consist of large homogeneous volumes; therefore, CNNs can be applied naturally here (e.g. spatial translational invariance holds).

Each event consists of two 2-D projections, and each pixel is really the energy response in a cell.

Objective is to categorize events, e.g., 4 categories are shown at right.

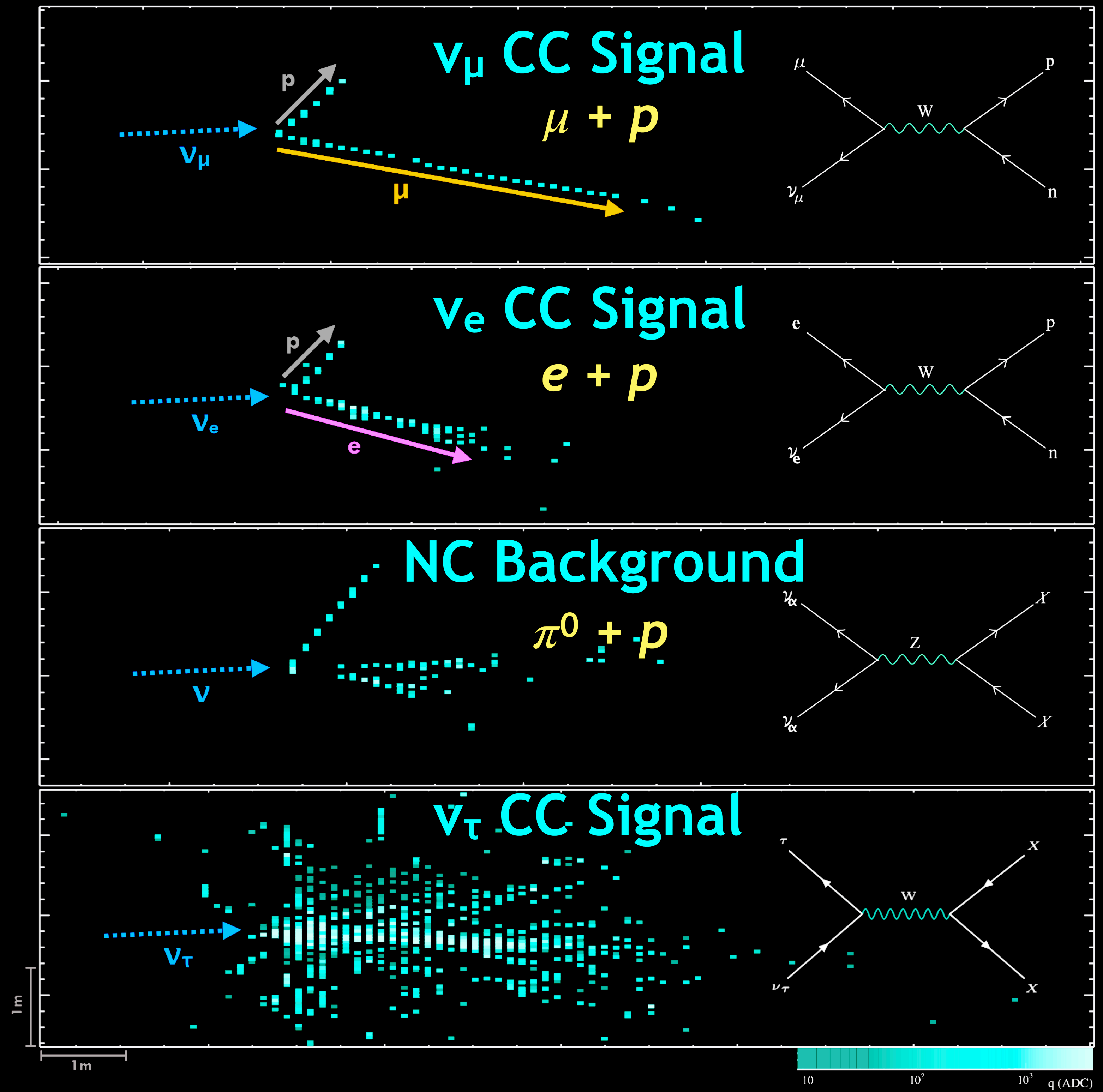


Image from A Radovic.

CNNs for Neutrinos

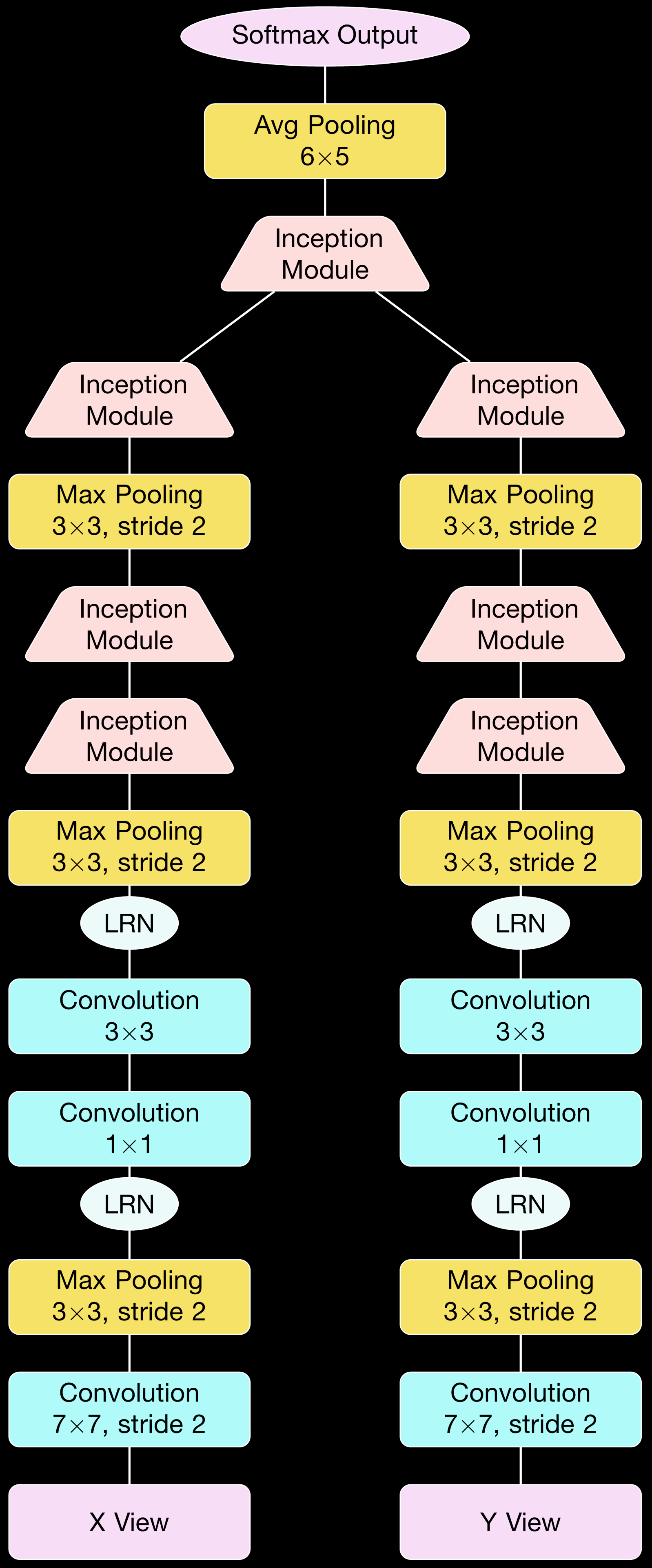
The NOvA CNN, which is similar to the first GoogLeNet CNN, splits each 2-D projection of a NOvA event into a separate sequence in the early layers, then concatenates their outputs near the end.

The objective is to categorize events as (e,μ,τ)x(QE,RES,DIS), or NC, or cosmic.

Built in CAFFE framework and utilizes softmax output (since this is a multi-class problem).

The NOvA CNN improves in performance over previous PID algorithms by an amount equivalent to collecting data for an additional 30% exposure time.

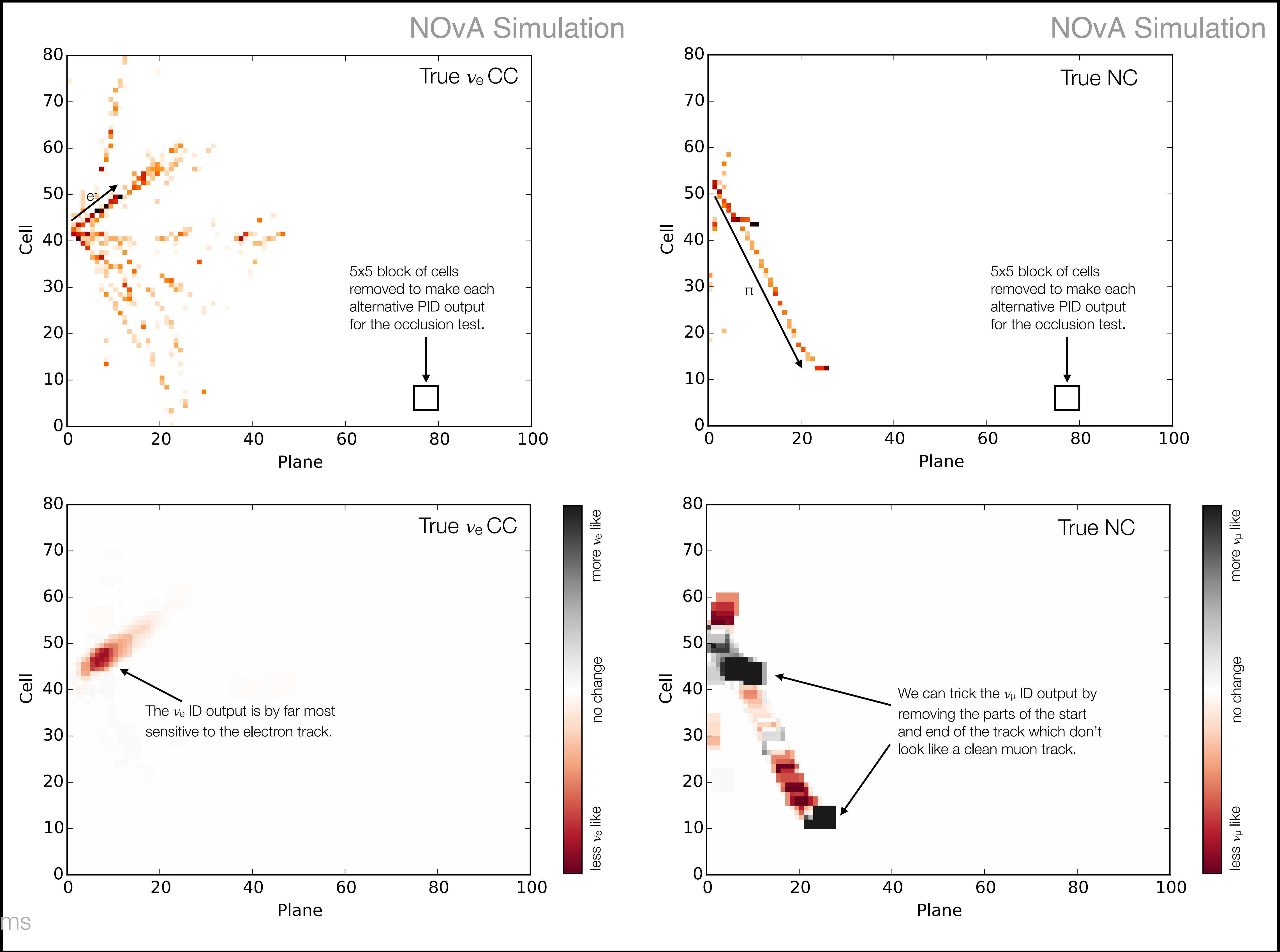
Aurisano, Radovic, Rocco, et al [1604.01444]



CNNs for Neutrinos

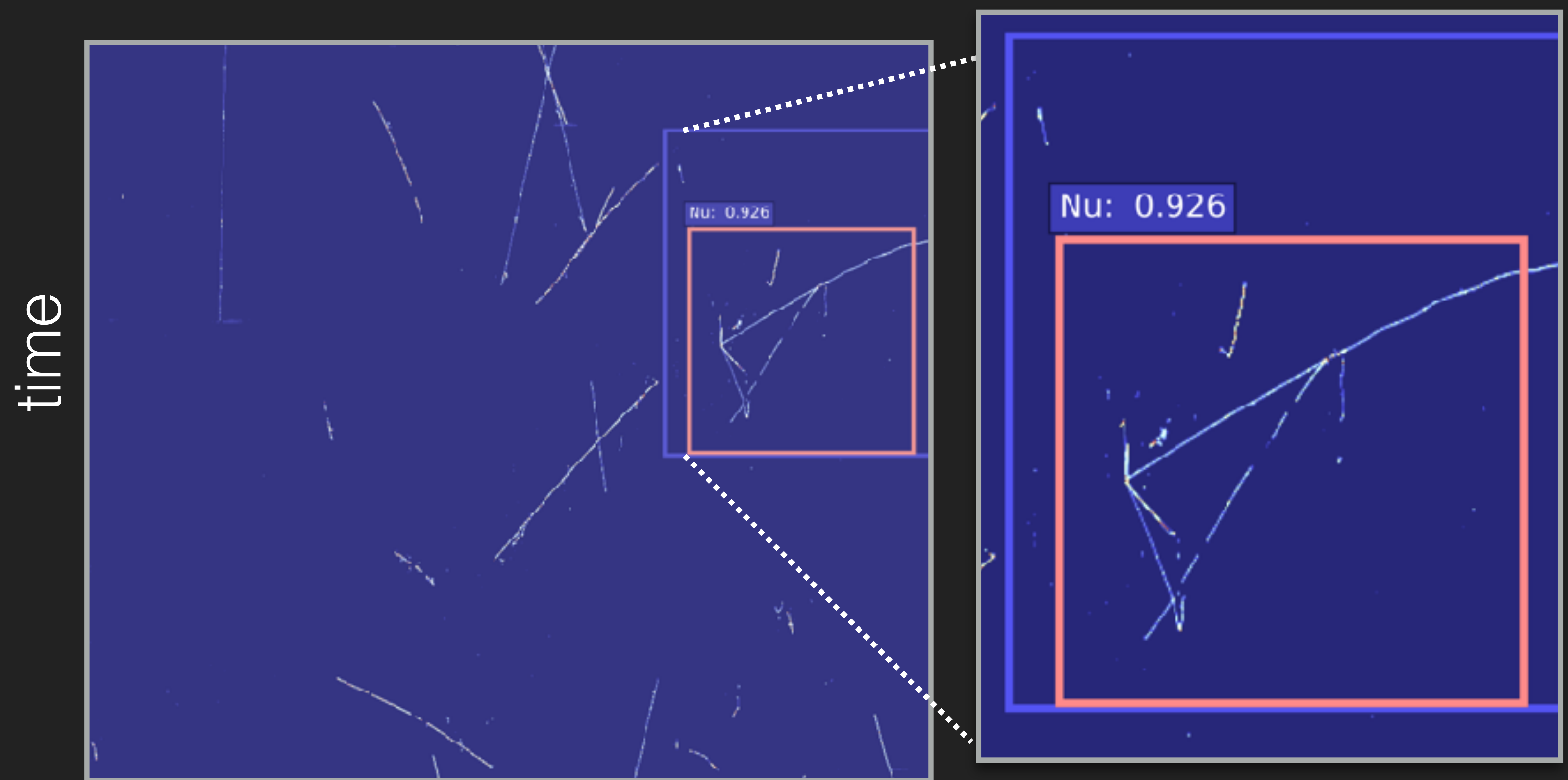
There are many ways to study what the CNN has learned. One example shown here.

Slide below from A Radovic



CNNs for Neutrinos

MicroBoone has managed to train CNNs that can locate neutrino interactions within an event (draw bounding boxes), identify objects and assign pixels to them [1611.05531].

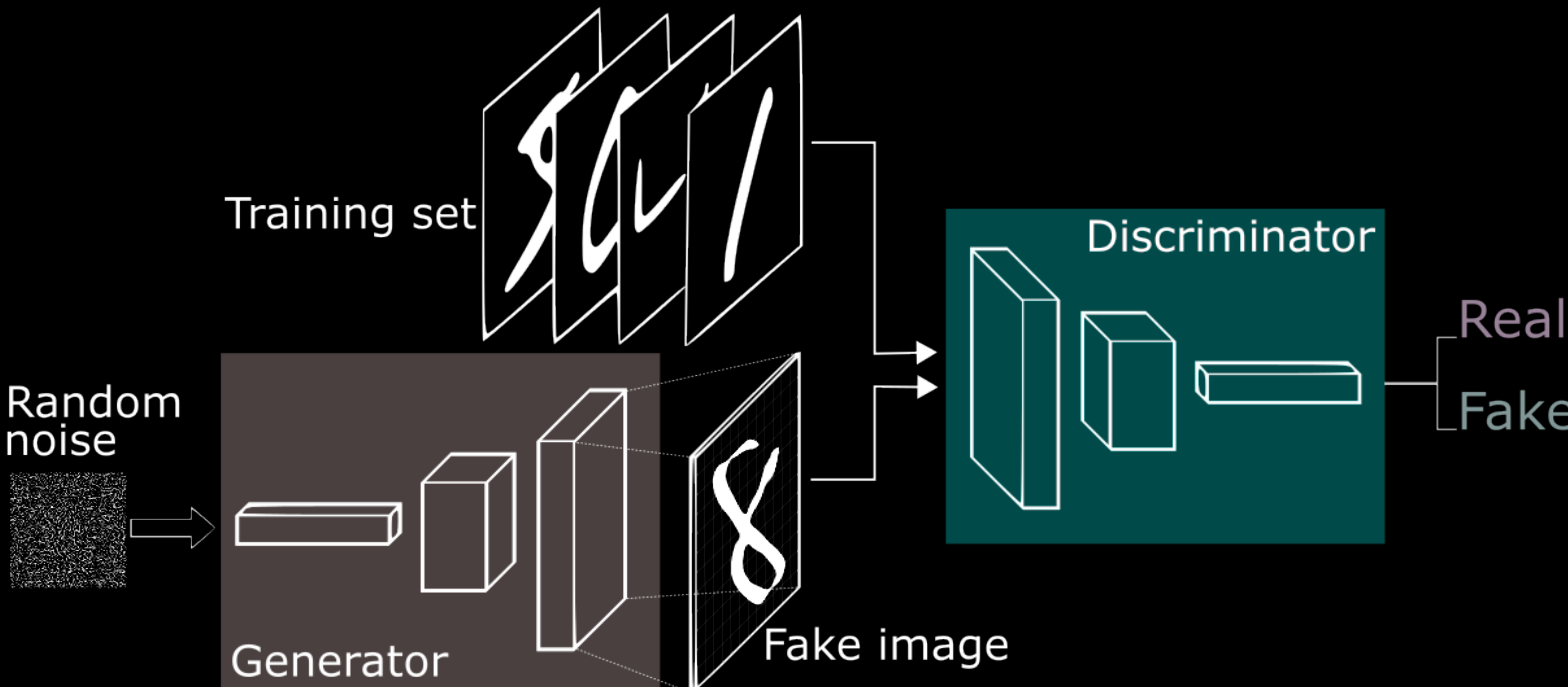


[image from T. Wongjirad]

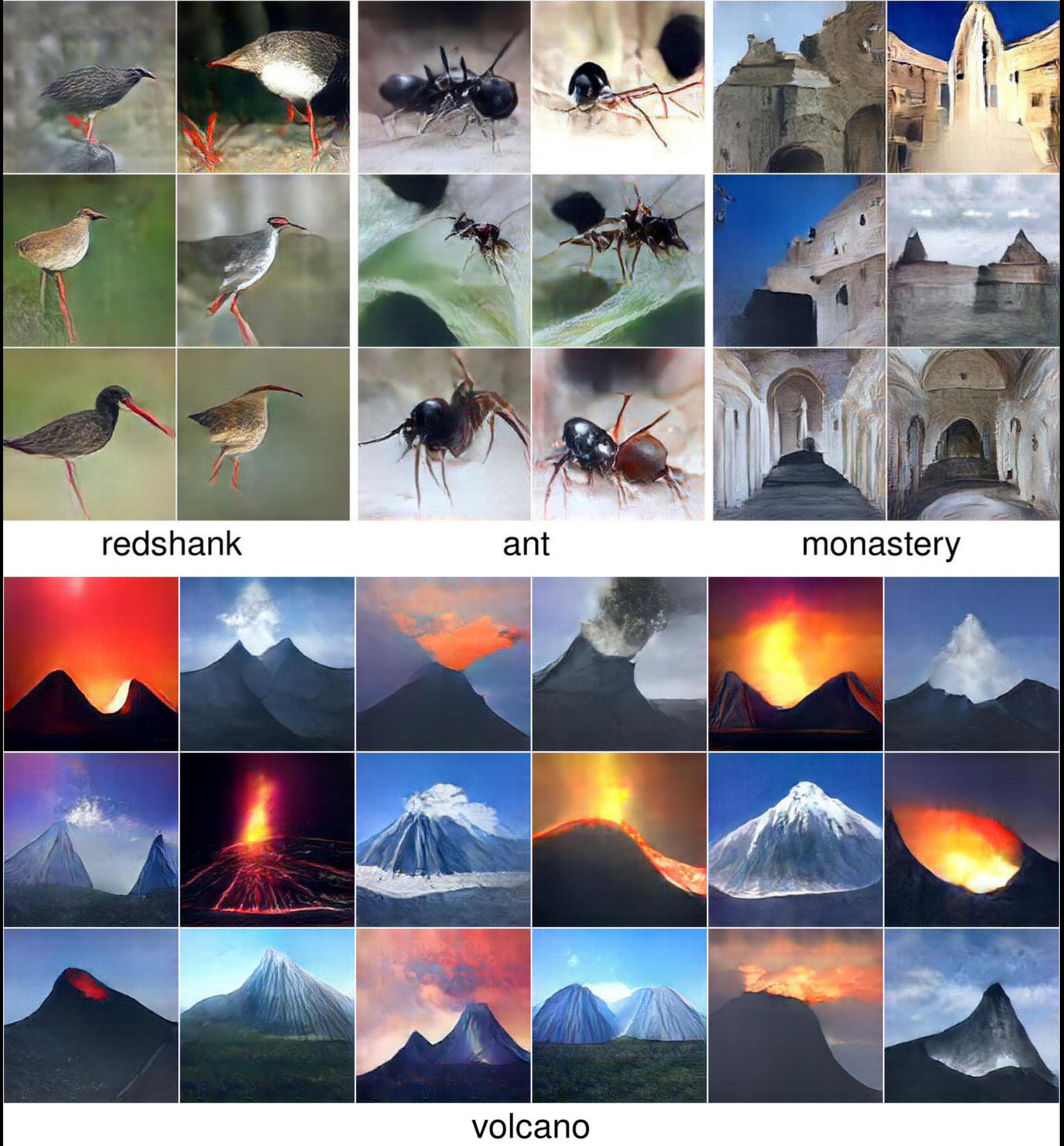
Similar work using CNNs ongoing at collider experiments in the area of jet physics (see [1511.05190], [1603.09349],…).

GANs

Generative adversarial networks (GANs) are a way of training a generative model to produce realistic data taking noise as input by pitting it against an adversary.



During training, the discriminator learns to better classify real and fake data, while the generator learns to better fool its adversary.


redshank
ant
monastery
volcano

More (non-HEP) GANs

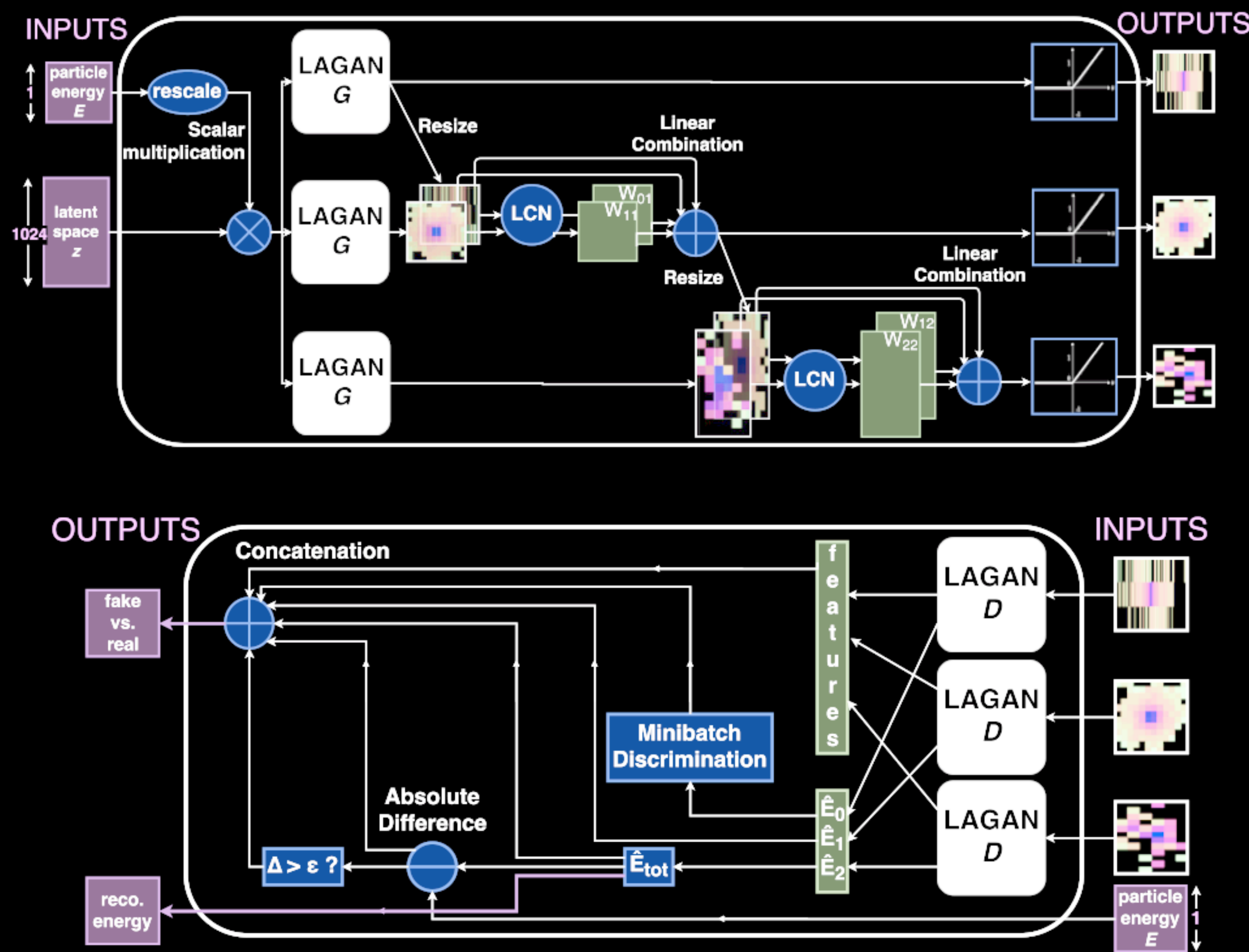
This is not a real person, but rather a machine's hallucination!

More (non-HEP) GANs

CALO GAN

Paganini, de Oliveria, Nachman [1712.10321].

Monte Carlo simulations in particle physics are based on well-known/well-modeled microphysics and the concept of factorization. GEANT is an amazing tool — but it is way too slow and prohibitive for generating large enough MC samples for future experimental runs.

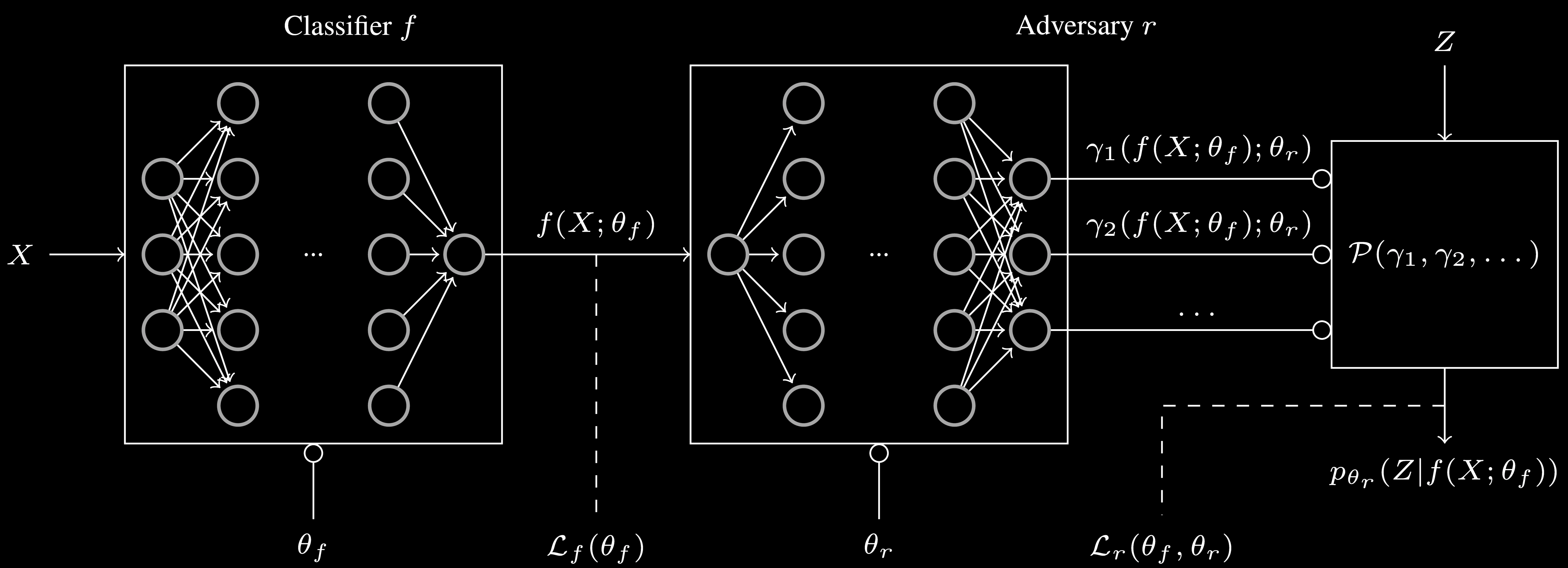


CALO GAN can generate the reconstructed CALO image using random noise, skipping the GEANT and RECO steps — making it 10,000 x faster than GEANT! (Every LHC experiment is studying GANs for HL-LHC MC generation.)

Adversaries & Loss

Yesterday we talked about physics-aware loss functions and de-correlating from some unknown inputs. What if we don't know how to write a loss function for what we want?

Louppe, Kagan, Cranmer [1611.01046]
Shimmin et al [1703.03507]

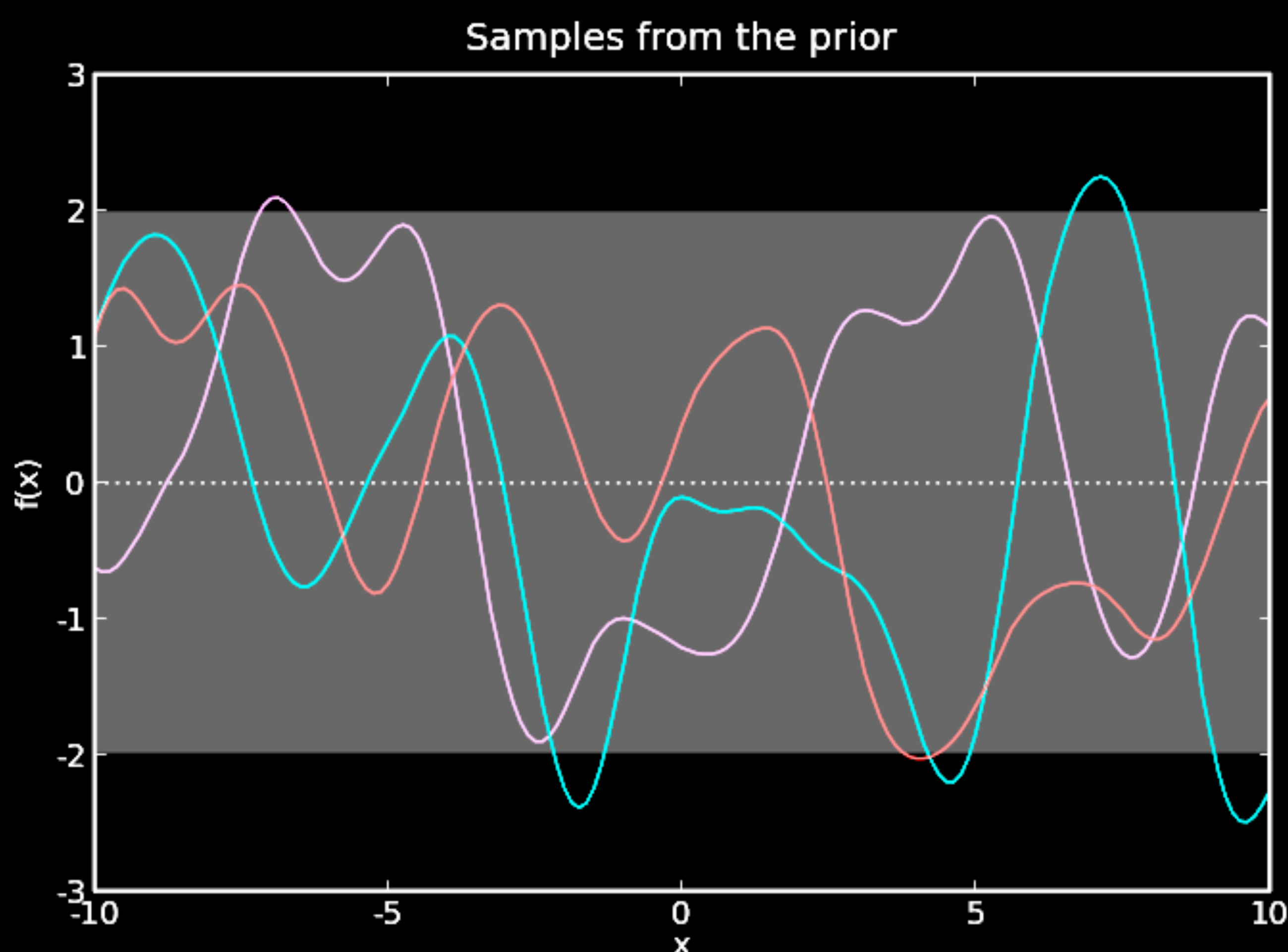


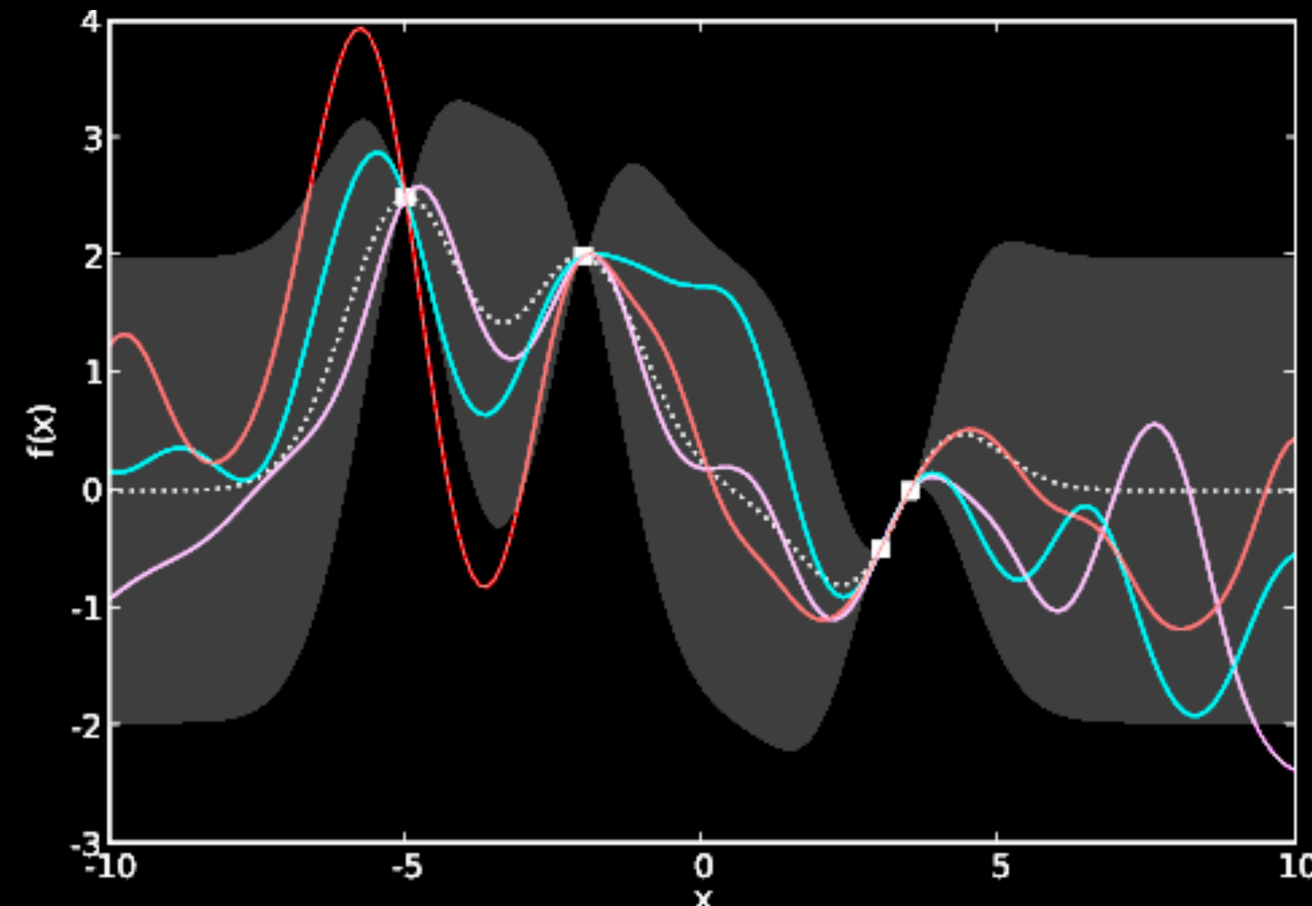
We can learn one! Now the adversary tries to guess nuisance parameters given the output of the classifier. Training drives the classifier response to be independent of these features—making it robust. (Shown to increase the sensitivity when systematics are considered.)

Gaussian Processes

What if it's not possible to calculate the gradient of the loss function, and therefore, a ML algorithm cannot be trained using gradient descent (e.g. tuning hyperparameters in a NN)?

The first few steps of any rigorous explanation of Gaussian processes seem like it's the least useful idea ever — but it's actually very useful. I'll try and explain it like a physicist.



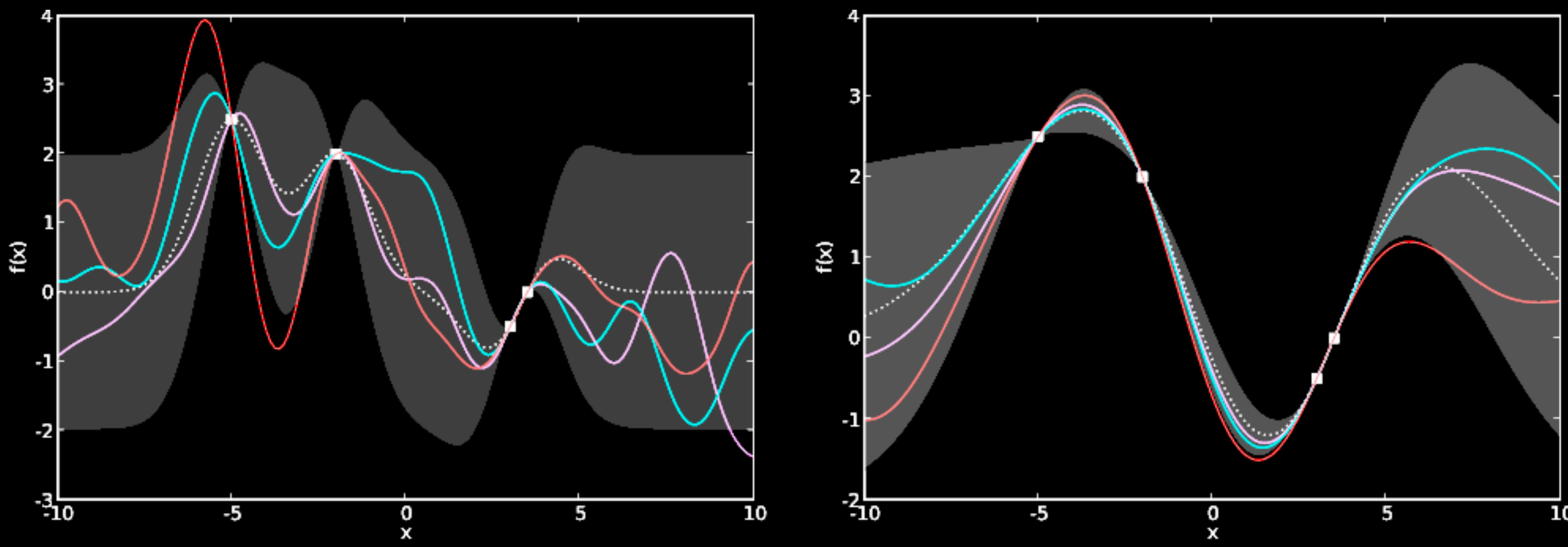


We want to learn a function, but we don't know it's functional form. Using a Bayesian non-parametric approach gives us the flexibility we need to describe this unknown function. What do observations tell us about the underlying model? (Should be a familiar question!)

Gaussian Processes

Any physicist should immediately think about the relevant scales in the problem. The plots below show the simplistic 1-D case. What is the smallest length scale in x? Answering that tells us how "wiggly" the function can be. (I.e., if we know the length scale, we know how much information nearby observations provide about the function at an unobserved location.)

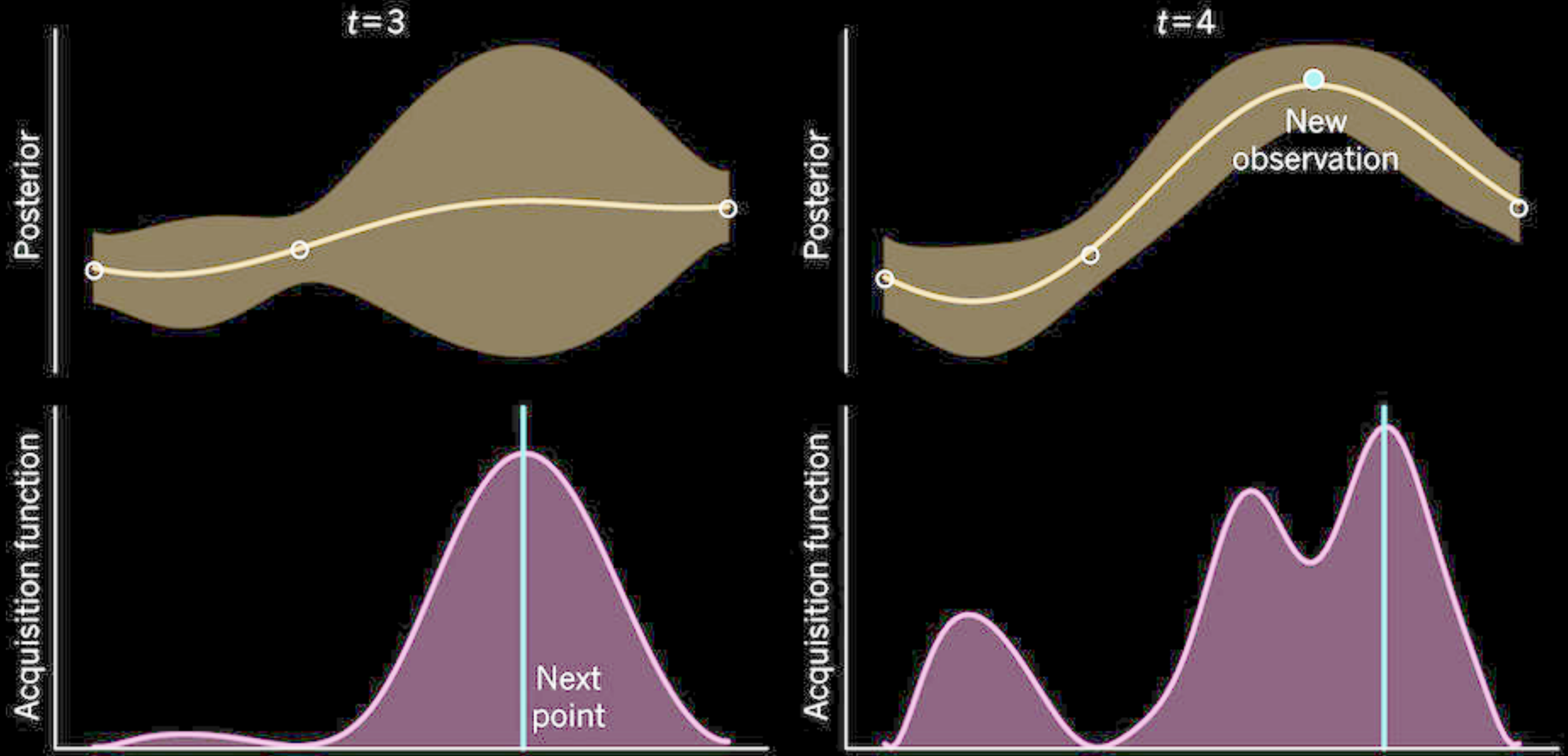
same data, different length scales



Assuming that we don't know the relevant scales (especially true in high dimensionality for black box type problems), we'll have to learn them from the data — not just the current observations, we can collect more data.

Bayesian Optimization

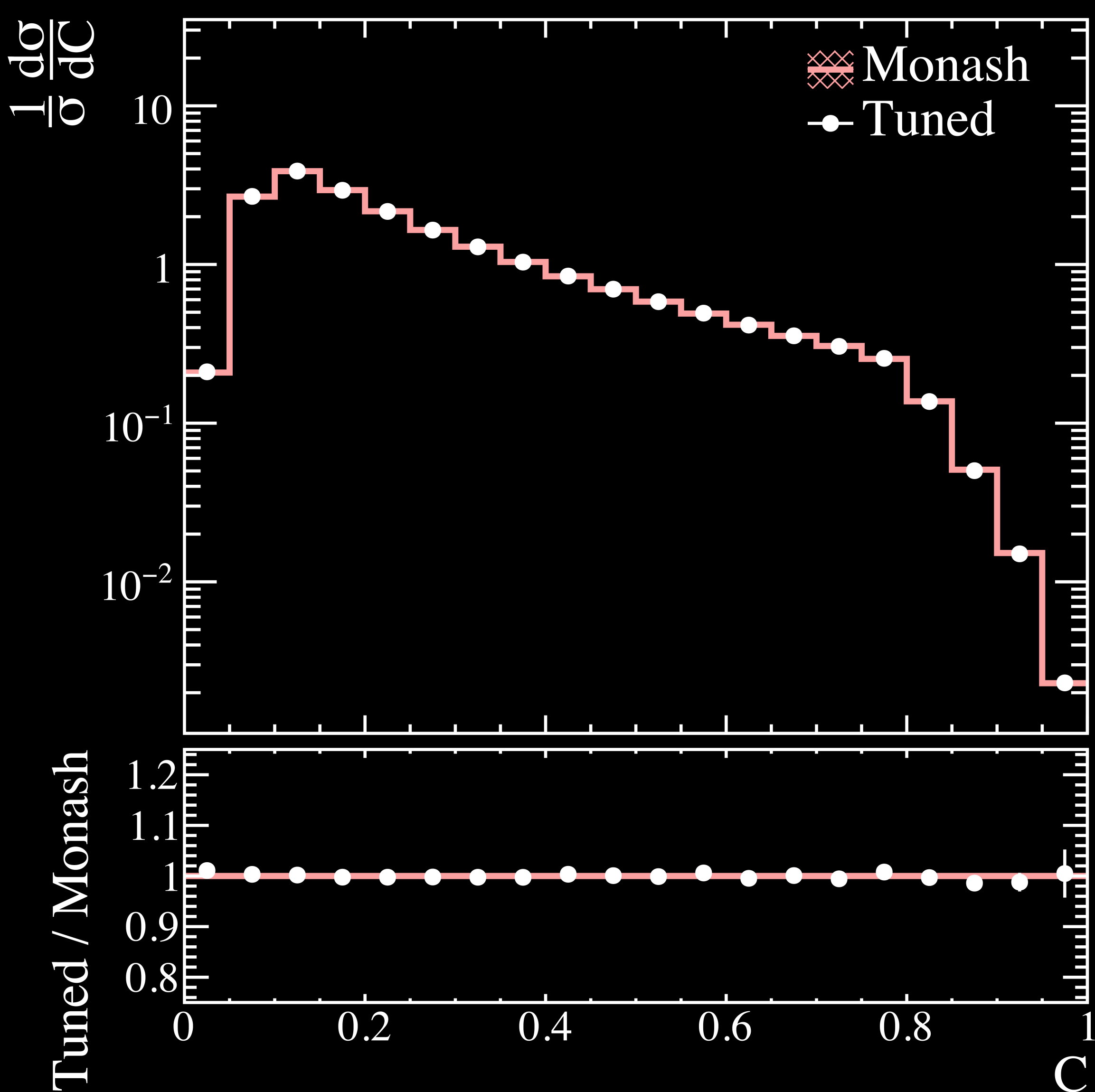
Start from prior for objective function, treat evaluations as data and produce a posterior used to determine the next point to sample (balances exploration vs exploitation). One of the main assumptions is that observations are expensive, so we want to find the optimal point using the fewest possible observations.

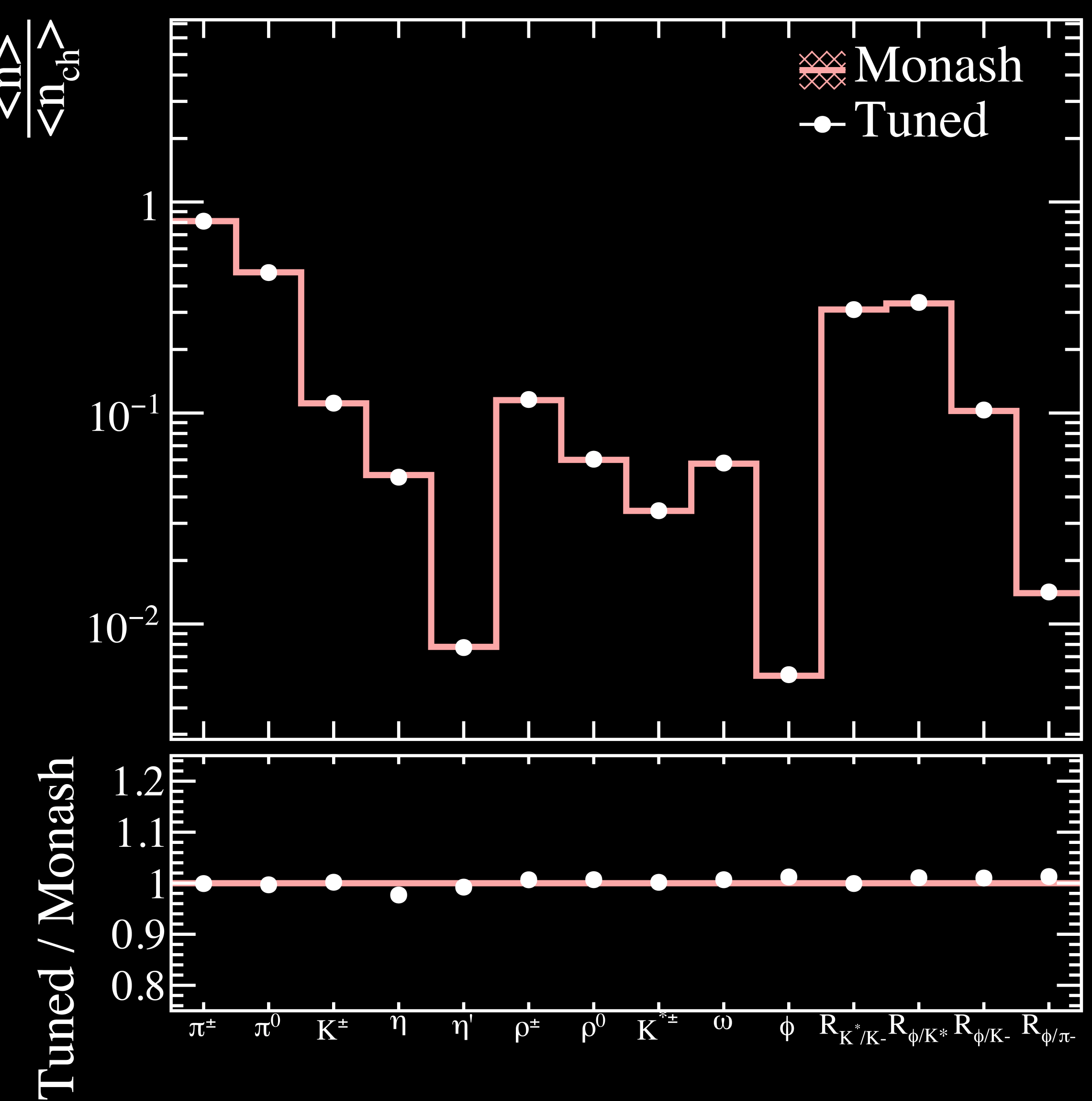


Pythia Tune

We considered tuning 20 parameters of Pythia using Monash MC as experimental data using BO to perform a closure test. BO quickly tuned the Pythia and was able to find the true input parameter values within the quoted uncertainties.

Ilten, MW, Yang [1610.08328]

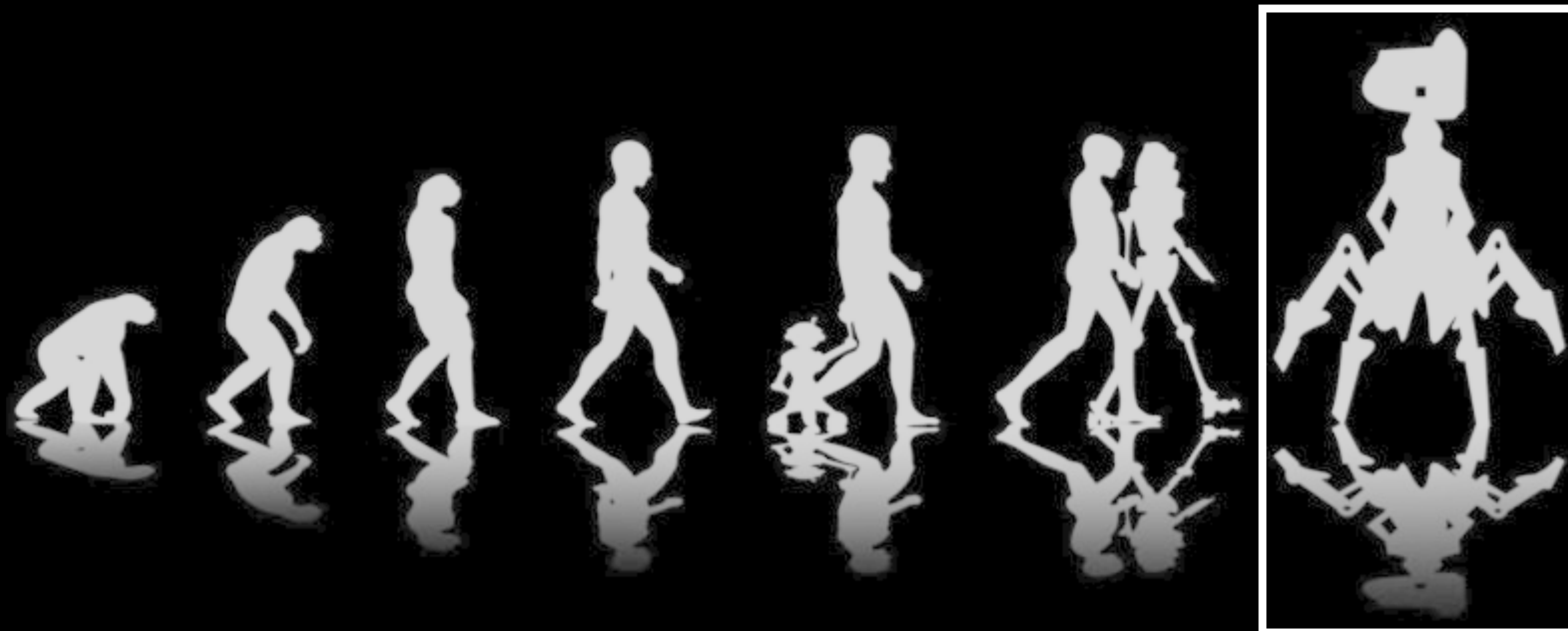




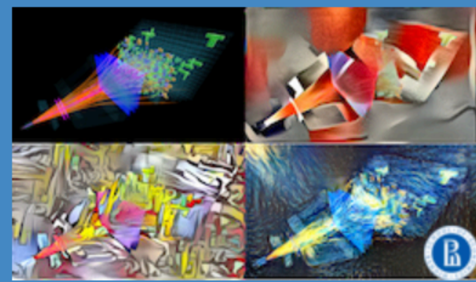
Converges took at most 50*n(parameters) iterations assuming no prior knowledge about the parameter values — and < 10*n(parameters) iterations using sensible ranges.

HEP AI?

Should we just send the raw data straight to some AI? First issue, we are constantly reformulating the problem; i.e., we don't *a priori* have a complete list of tasks to perform or questions to answer (maybe AI can ask the questions?).



Beyond this, our current approach has advantages: it's modular, making the outputs reusable (efficient), interpretable and easily validated, etc. If we are going to radically change strategy, we'd like to maintain these features—and whatever we do it must work in real time!



Fourth Machine Learning in High Energy Physics Summer School 2018

6-12 August 2018
University of Oxford
Europe/London timezone

Search...

- Overview
- Timetable
- School information
  - Speakers
  - Social programme
  - Application Process and Important dates
  - Committees
  - MLHEP participants feedback
- Local information
  - Visa
  - Venue
  - Accommodation
  - About Oxford
  - Food and drinks
  - Getting to Oxford
- Registration fee
- Application Form
- Frequently asked questions
- Competition

Support

mlhep2018@yandex.ru

The Fourth Machine Learning summer school organised by Yandex School of Data Analysis, Laboratory of Methods for Big Data Analysis of National Research University Higher School of Economics and University of Oxford will be held in Oxford, UK from 6 to 12 August 2018.

The school will cover the relatively young area of data analysis and computational research that has started to emerge in High Energy Physics (HEP). It is known by several names including "Multivariate Analysis", "Neural Networks", "Classification/Clusterization techniques". In more generic terms, these techniques belong to the field of "Machine Learning", which is an area that is based on research performed in Statistics and has received a lot of attention from the Data Science community.

There are plenty of essential problems in high energy physics that can be solved using Machine Learning methods. These vary from online data filtering and reconstruction to offline data analysis.

Students of the school will receive a theoretical and practical introduction to this new field and will be able to apply acquired knowledge to solve their own problems. Topics ranging from decision trees to deep learning and hyperparameter optimisation will be covered with concrete examples and hands-on tutorials. A special data-science competition will be organised within the school to allow participants to get better feeling of real-life ML applications scenarios.

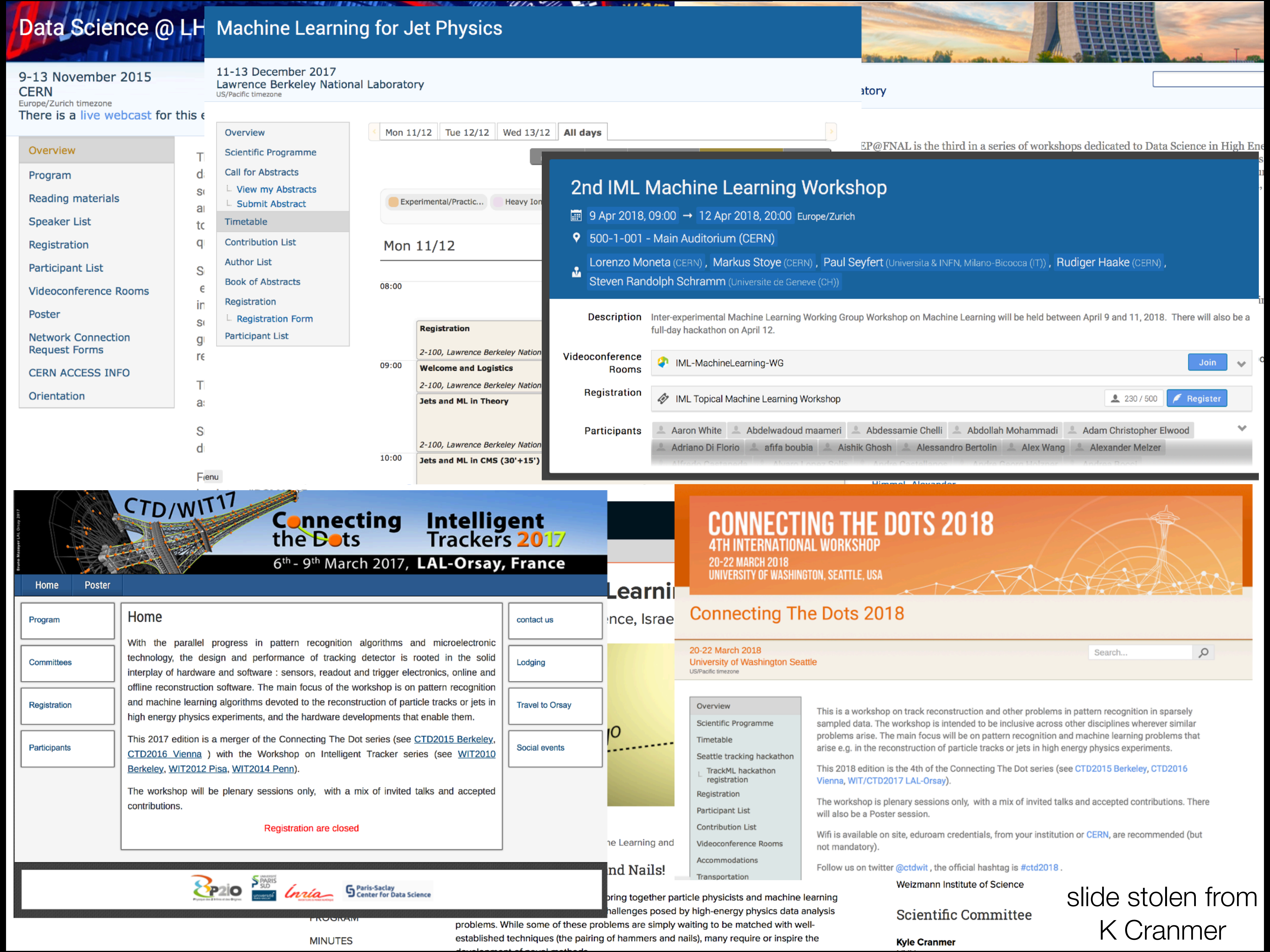
Expected number of students for the school is 50-60 people. The school is aimed at PhD students and postdoctoral researchers, but also open to masters students.

Pre-requisites for participation

- Python programming experience (e.g. http://nbviewer.jupyter.org/gist/rpmuller/5920182, https://www.codecademy.com/tracks/python)
- interest and/or background in HEP
- laptop with WiFi connectivity

Upon completion of the school participants would be able to

- formulate a HEP-related problem in ML-friendly terms;
- select quality criteria for a given problem;
- understand and apply principles of widely-used classification models (e.g. boosting, bagging, BDT, neural networks, etc) to practical cases;
- optimise features and parameters of a given model in efficient way under given restrictions;
- select the best classifier implementation amongst a variety of ML libraries (scikit-learn, xgboost, deep learning libraries, etc);
- understand and apply principles of generative model design;
- define & conduct reproducible data-driven experiments.



slide stolen from K Cranmer

Summary

- The use of ML has become ubiquitous in HEP. Many common classification and regression tasks already performed by ML-based algorithms.
- Deep learning is starting to make an impact, first with HEP problems that are closely related to those commonly solved using DL—but we're now moving towards a *producer* phase (rather than just consumer) in HEP.
- Systematics are vital in HEP. Our field is developing *systematics aware* ML algorithms—and has become adept at characterizing black boxes.
- Beyond the issue of systematics, our data/problems have other interesting features from a CS perspective: sparse data, irregular detector geometries, heterogeneous information, physical symmetries and conservation laws, etc.
- Finally, there is a lot of exciting cutting-edge work I did not have time to discuss, e.g., Deep Kalman Filters, jet tagging using RNNs, deep NNs are now running on FPGAs, automatic anomaly detection, etc.