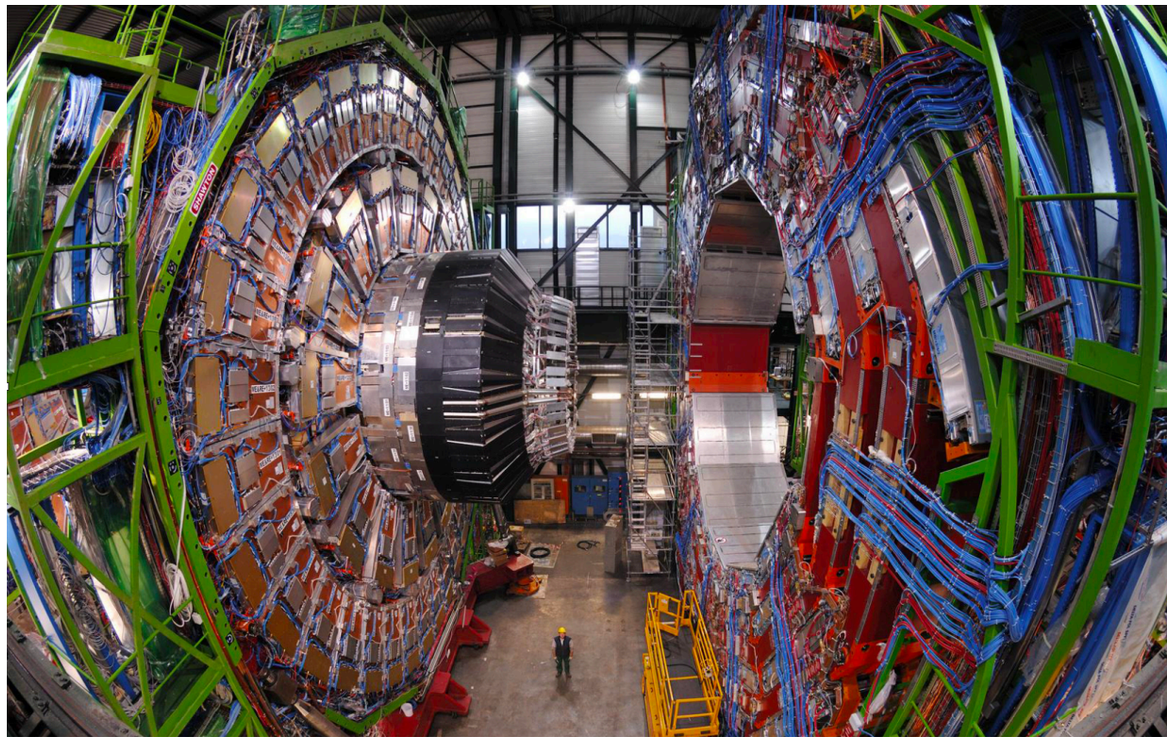
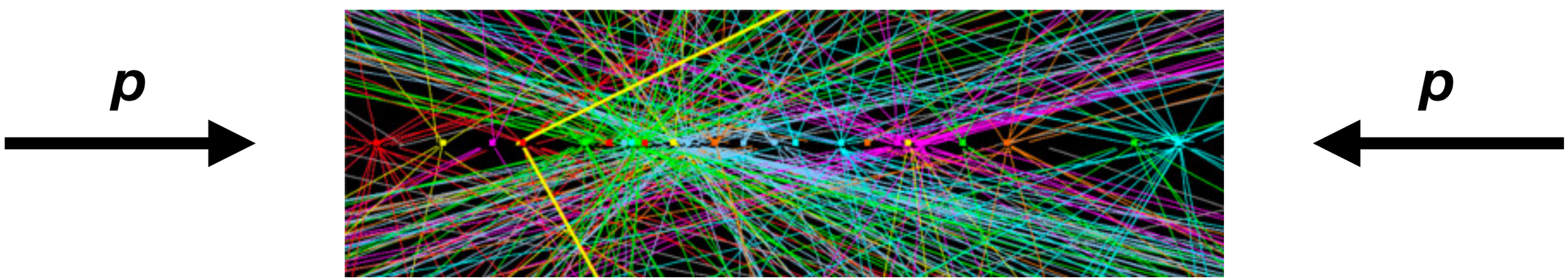
Large Hadron Collider

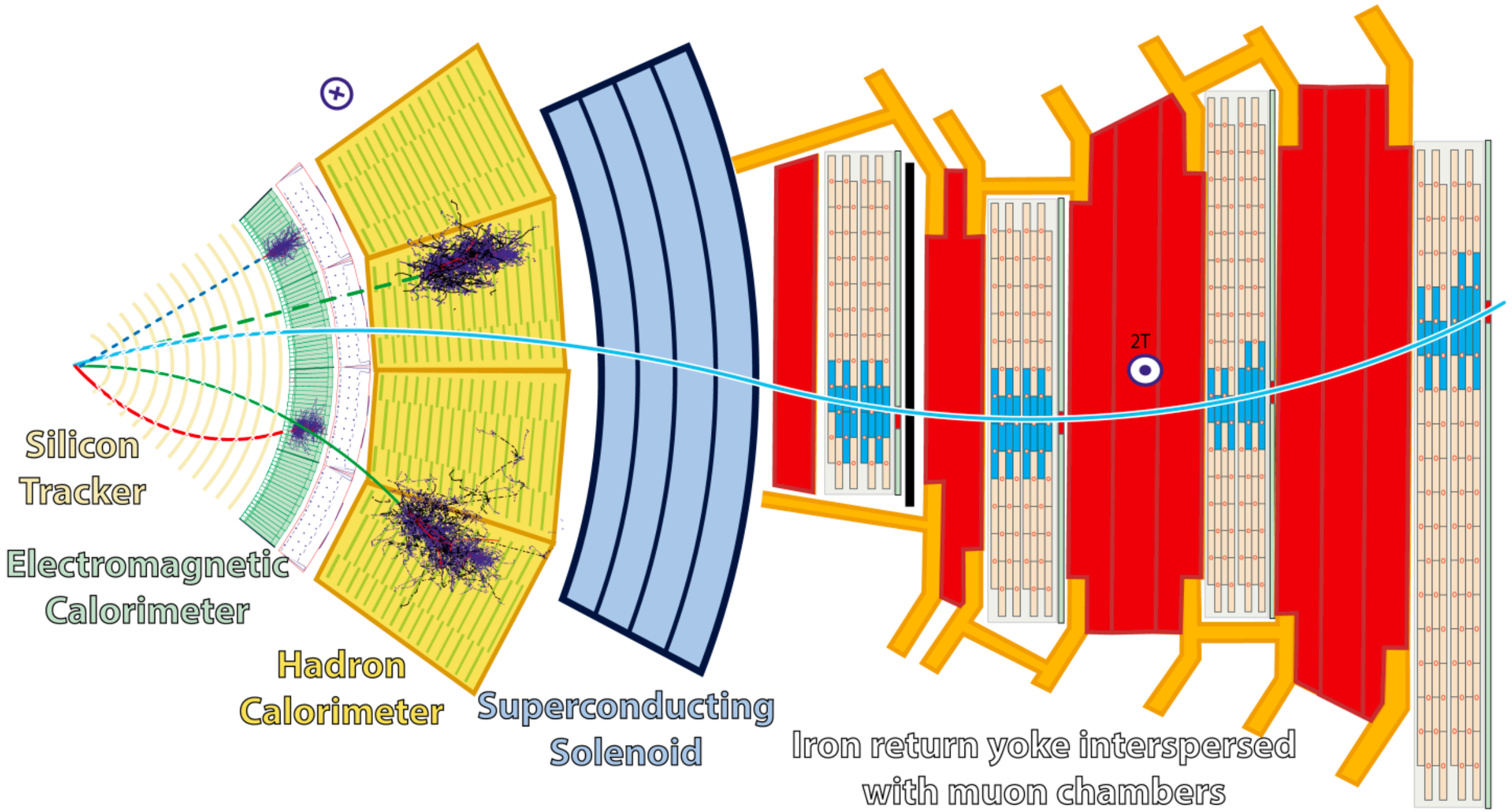
- Large Hadron Collider (proton - proton)
- Dedicated detectors to record data from collisions
    - Electromagnetic calorimeter (ECAL) is designed to measure energy of EM particles, **e + γ (EG)**



- Many overlapping collisions in addition to the primary one, called **pileup (PU)**
- Want to design algorithm to distinguish primary **EG** energy deposits from **PU**

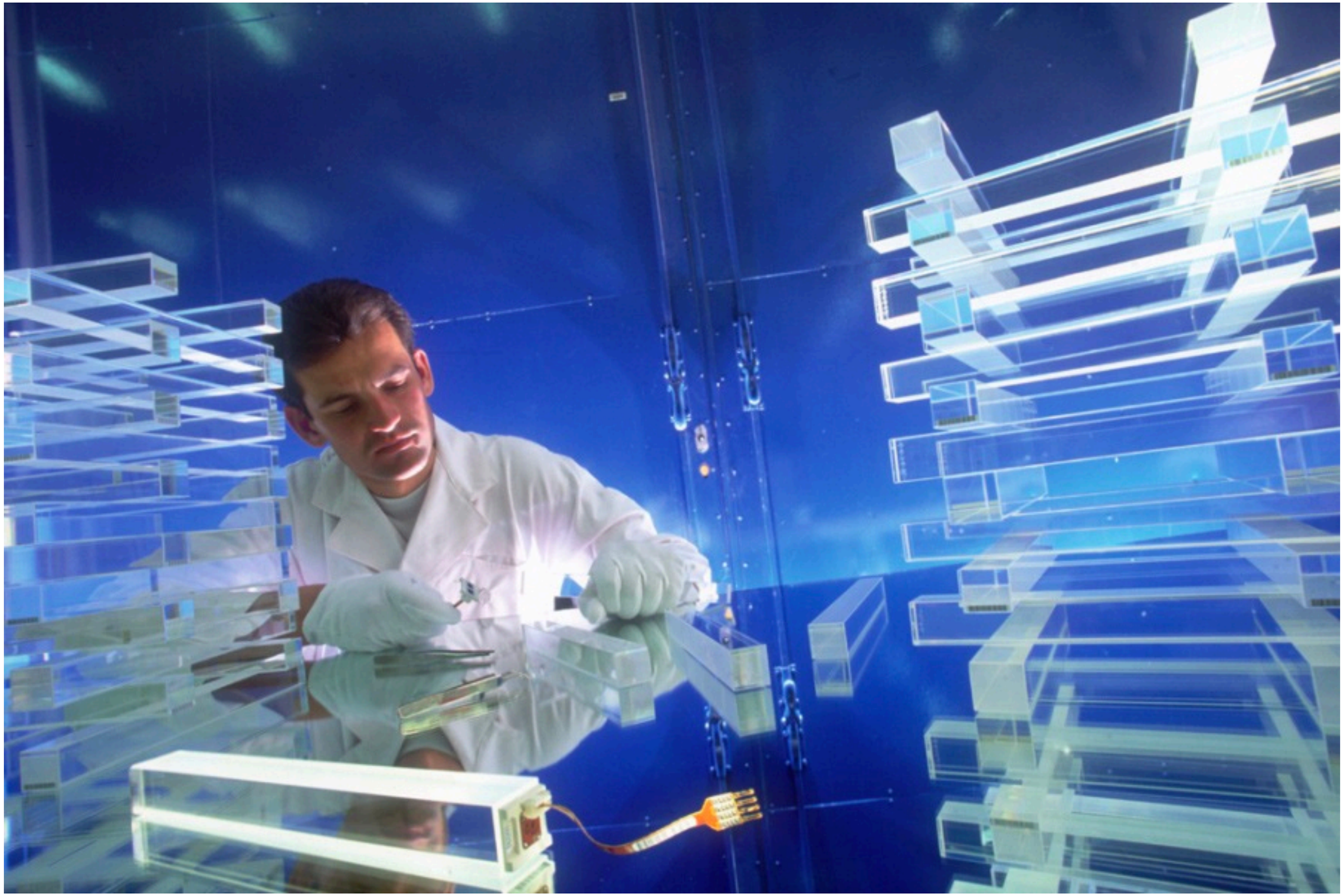


CMS Detector



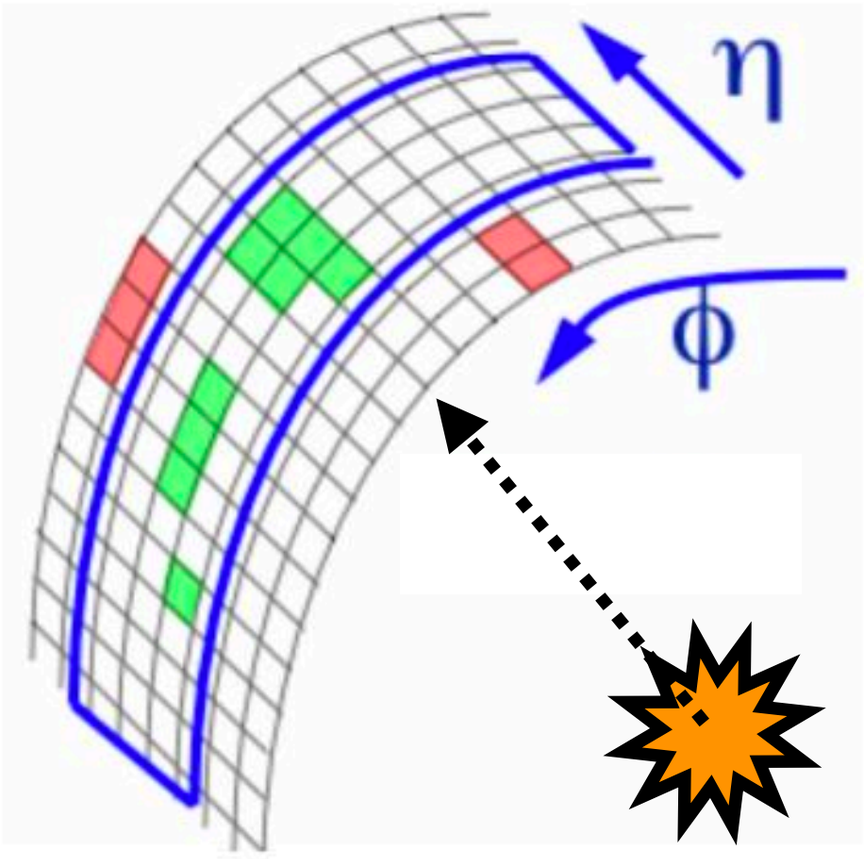
Muon
Electron
Charged hadron (e.g. pion)
Neutral hadron (e.g. neutron)
Photon

CMS ECAL

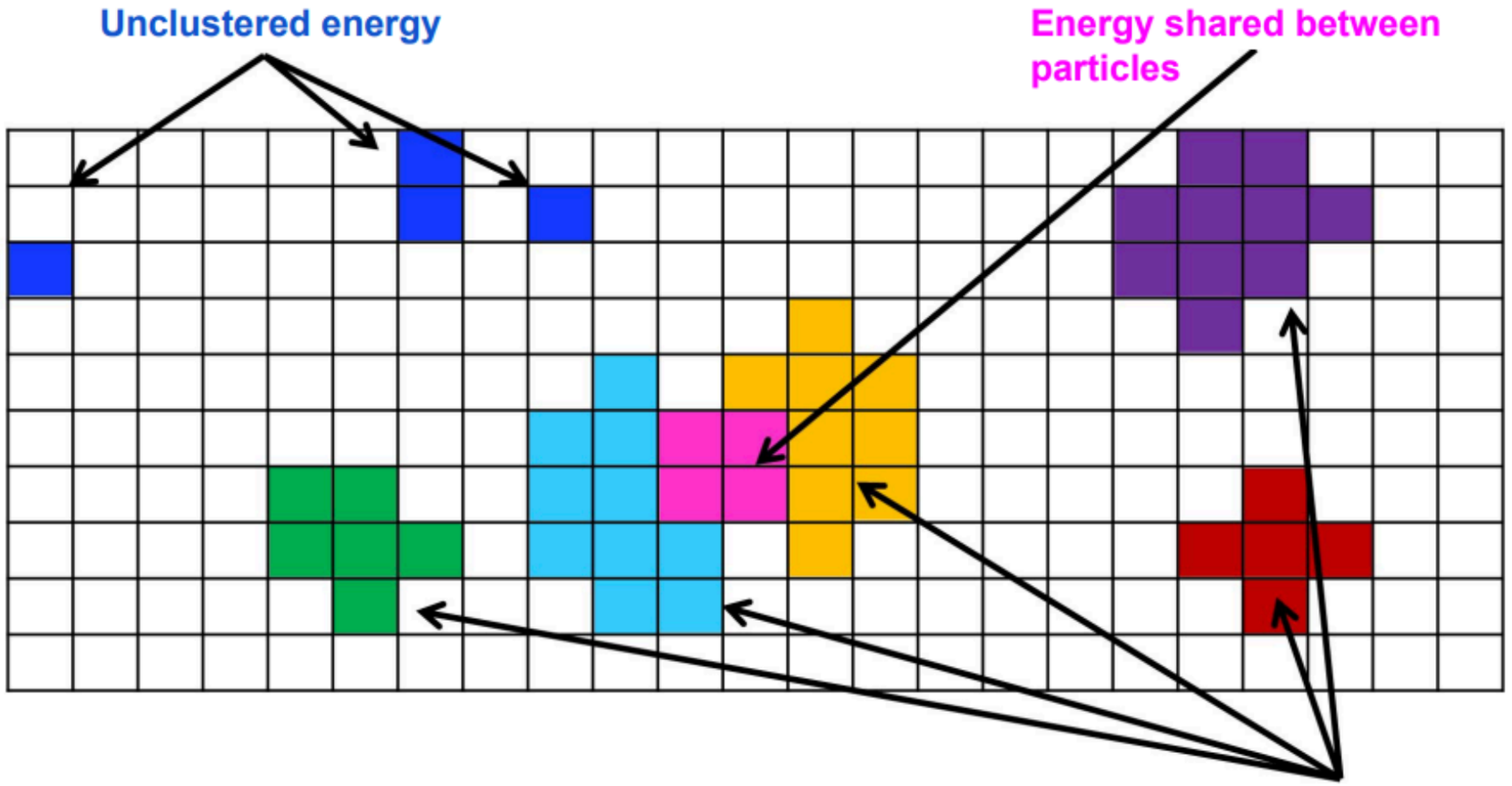


Dataset Explanation

- Real photon and electrons will deposit energy in certain patterns
    - PU is either random or from other sources, should not match
- We describe energy deposits using “shower shape variables”
    - i.e. How much energy is in certain regions around the center? How correlated are deposits in η and ϕ? How many crystals in particular directions?

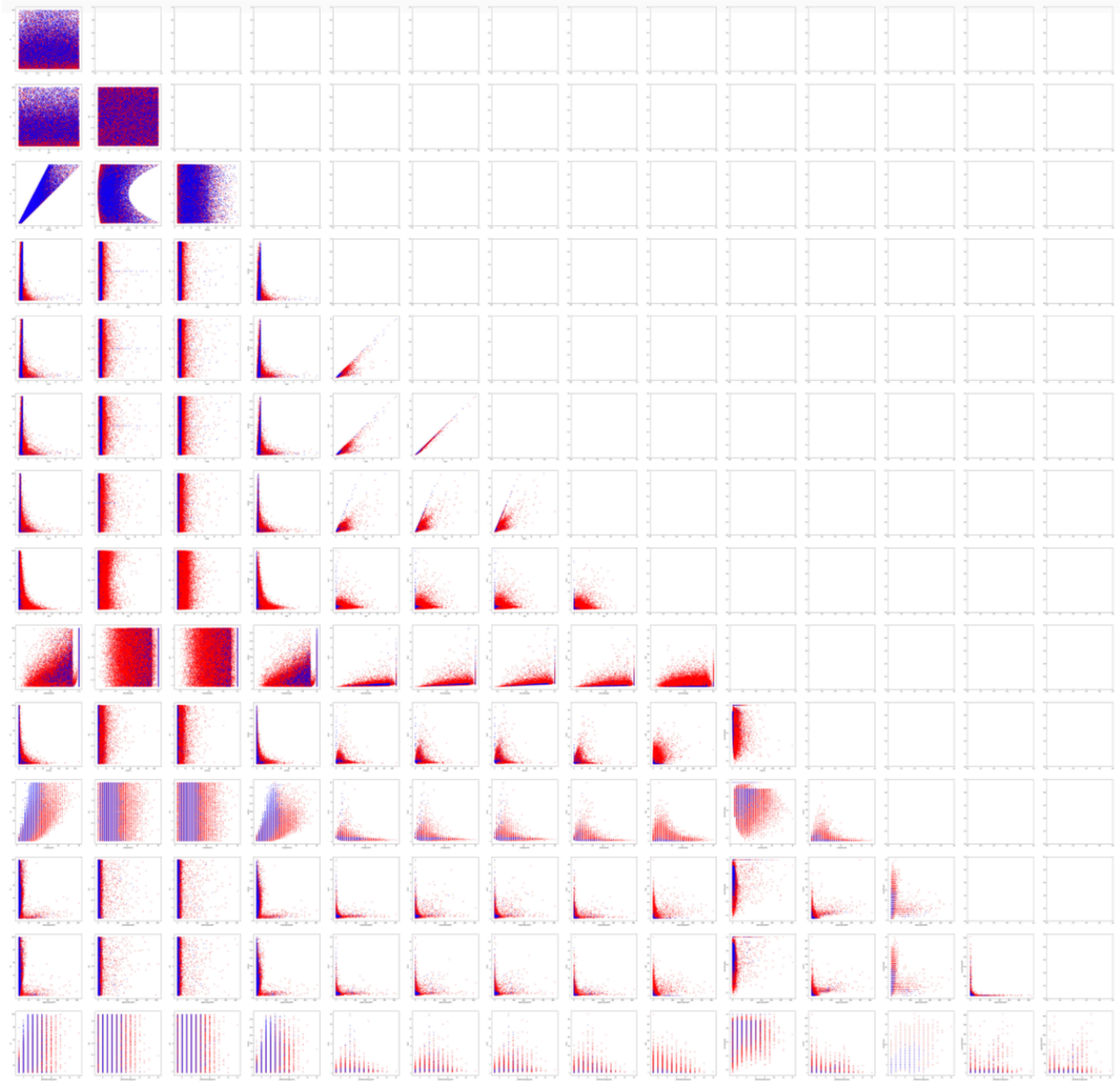


Dataset Visualization



- Start with pattern of detector hits
- Two main tasks:
  - Associate each incident particle with a collection of hits

Designing ML



How can I predict if a point is red or blue given x_1 and x_2 and x_3 and …?

(Lets use the notebook)

Batch Norm & Dropout

- In addition we often employ these two to improve perf

Batch Norm makes output Gaussian-like

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

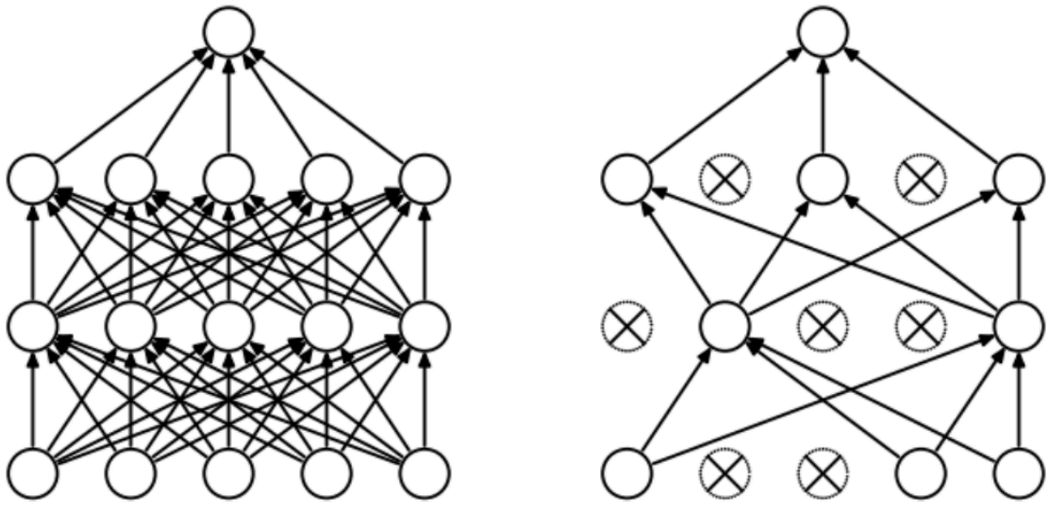
$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Dropout randomly removes weights



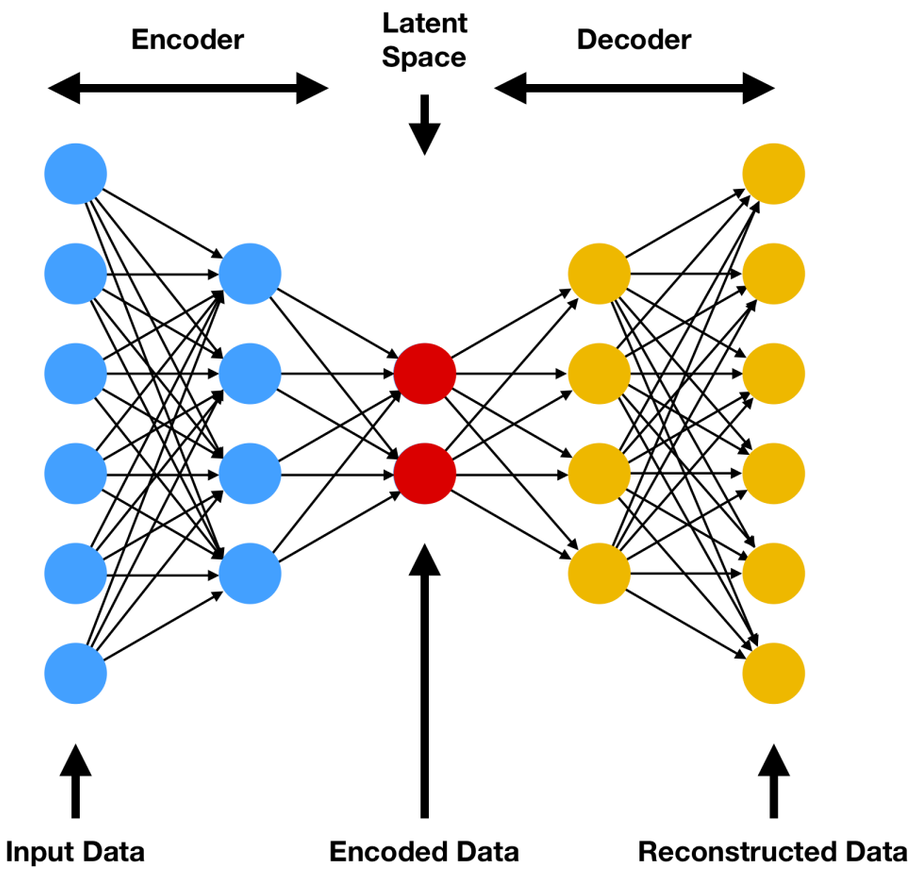
(a) Standard Neural Net (b) After applying dropout.

- Both Strategies make NNs more robust to deviations

More Complex Architectures

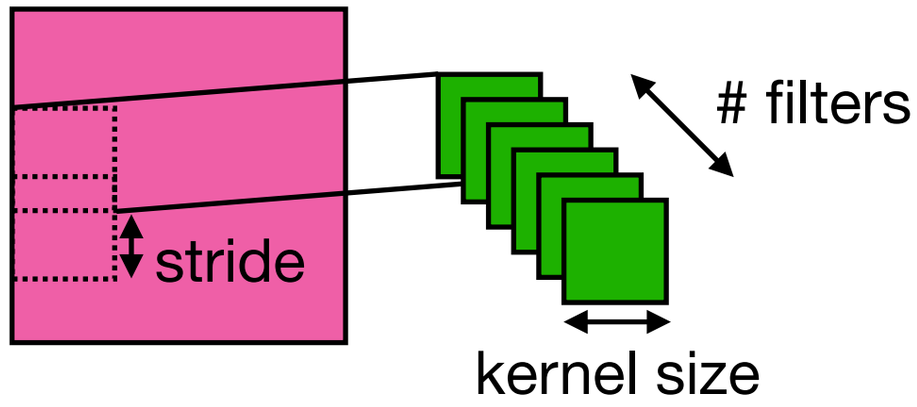
Unsupervised Learning

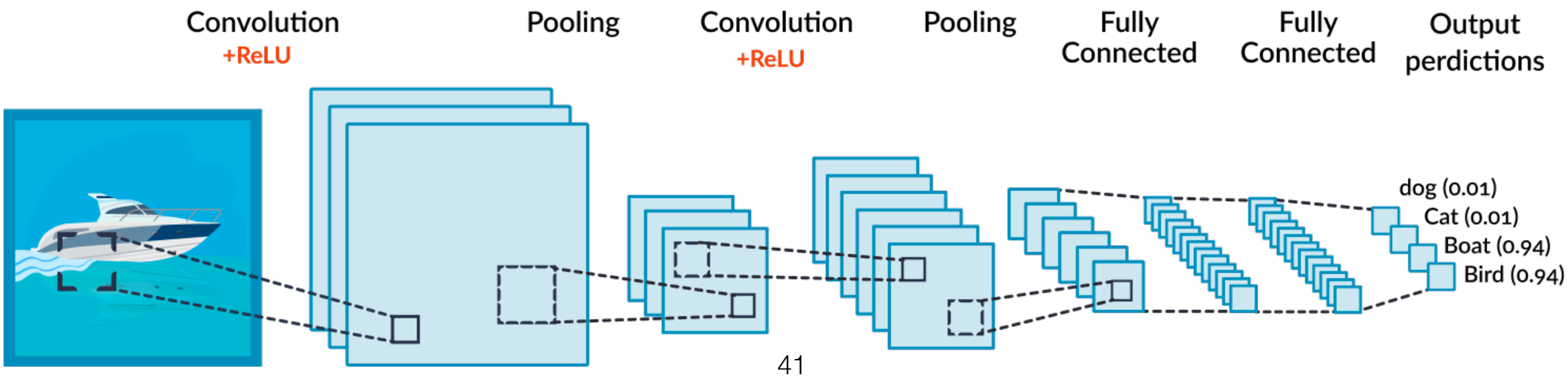
- What if we don’t have/use labels?
- Autoencoder (AE):
  - *Loss = output - input*
  - Latent space can be used for clustering
  - If one class of data is used to train, different classes may not reconstruct well (anomaly detection)



Convolutional Neural Network (CNN)

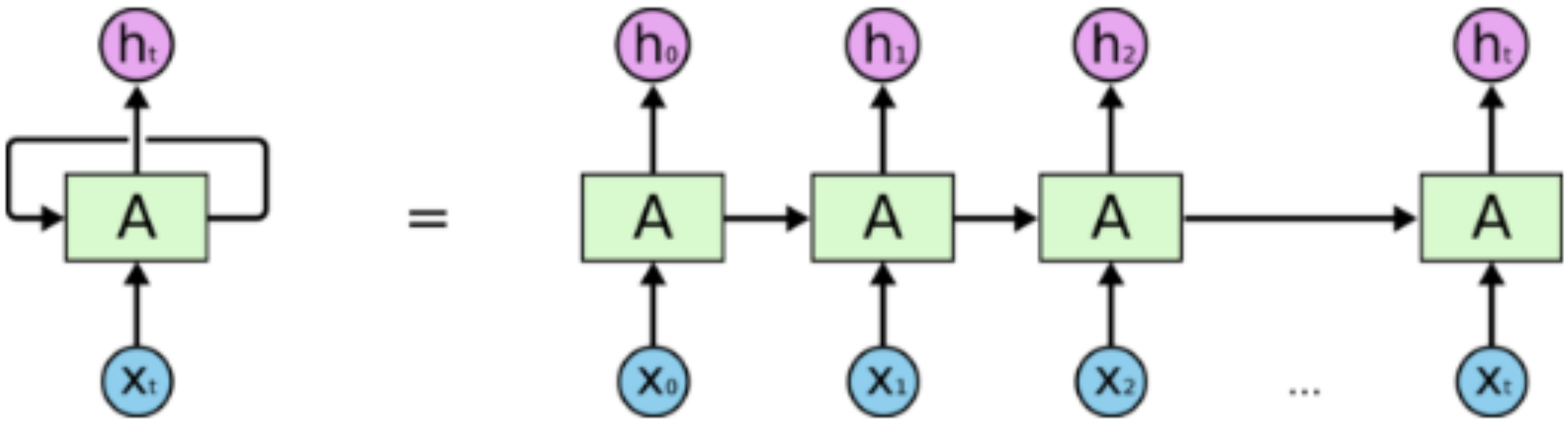
- Used extensively for images (Conv2D), also useful for 1D and 3D cases
- Small dense network takes a local region as input, scans over whole image
  - *# filters, kernel size, stride*
  - Typically followed by Pooling layer to reduce dimensionality





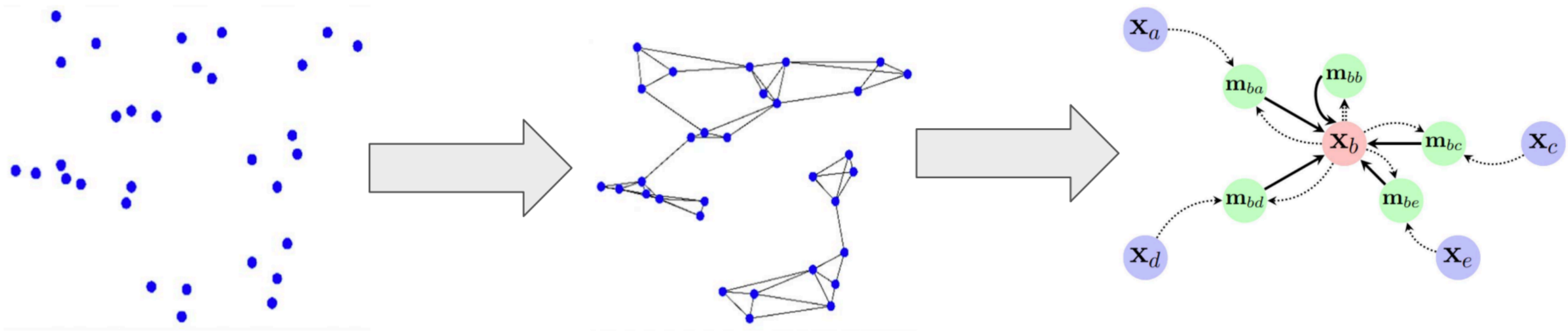
Recurrent Neural Network (RNN)

- Designed for sequential inputs (ex. language)
    - Retain “memory”
- Long short-term memory (LSTM), gated recurrent unit (GRU)



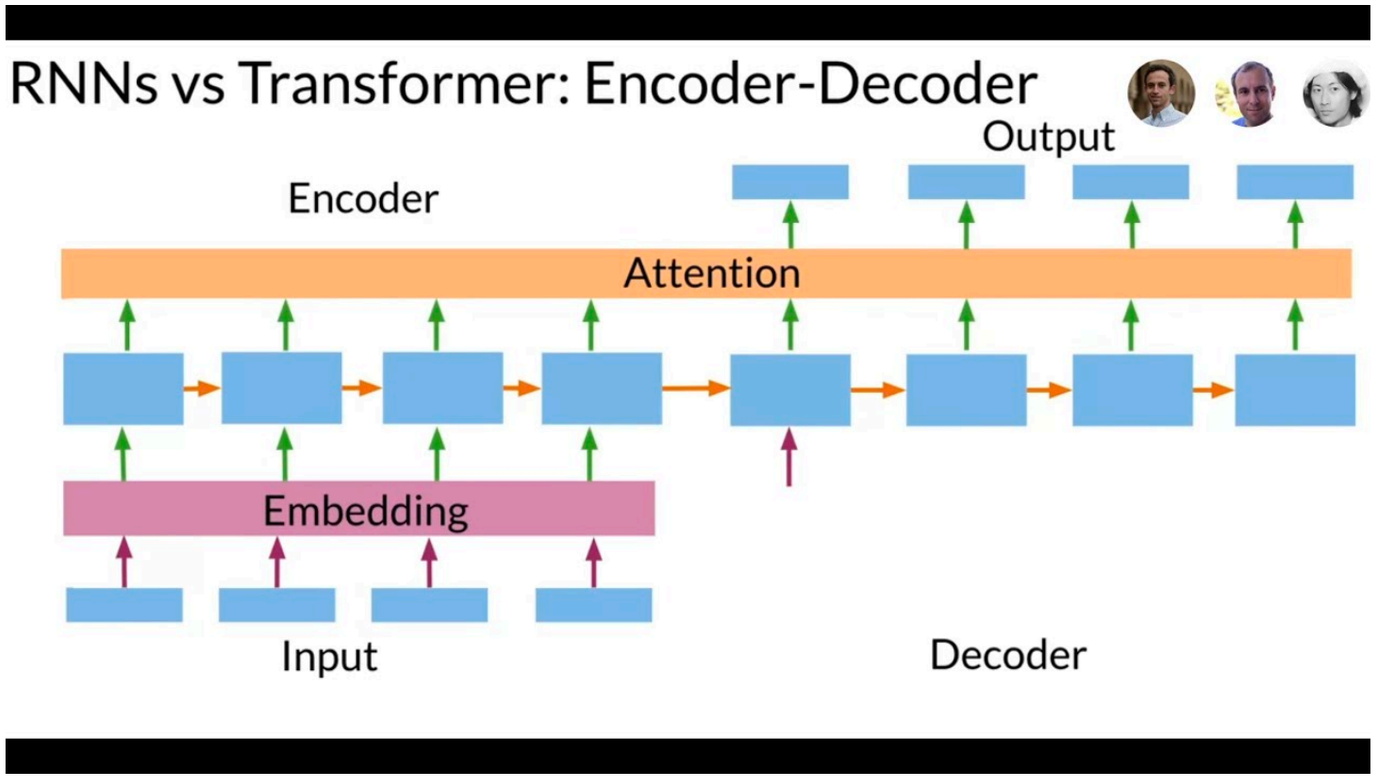
Graph Neural Network (GNN)

- Take in a set of points
  - Construct a graph out of these points
    - Make links between the neighbors
    - Iterate back and forth with the neighbors



Transformers

- Originated from recursive neural networks
  - Reading a whole sentence is better than iteratively
    - Done by applying Attention (effectively a big linear layer)
  - Basically a Graph with a notion of ordering



Next Lecture

- We are going to look at another application of ML
  - Using ML for regressions
  - How can we solve for complex physical scenarios