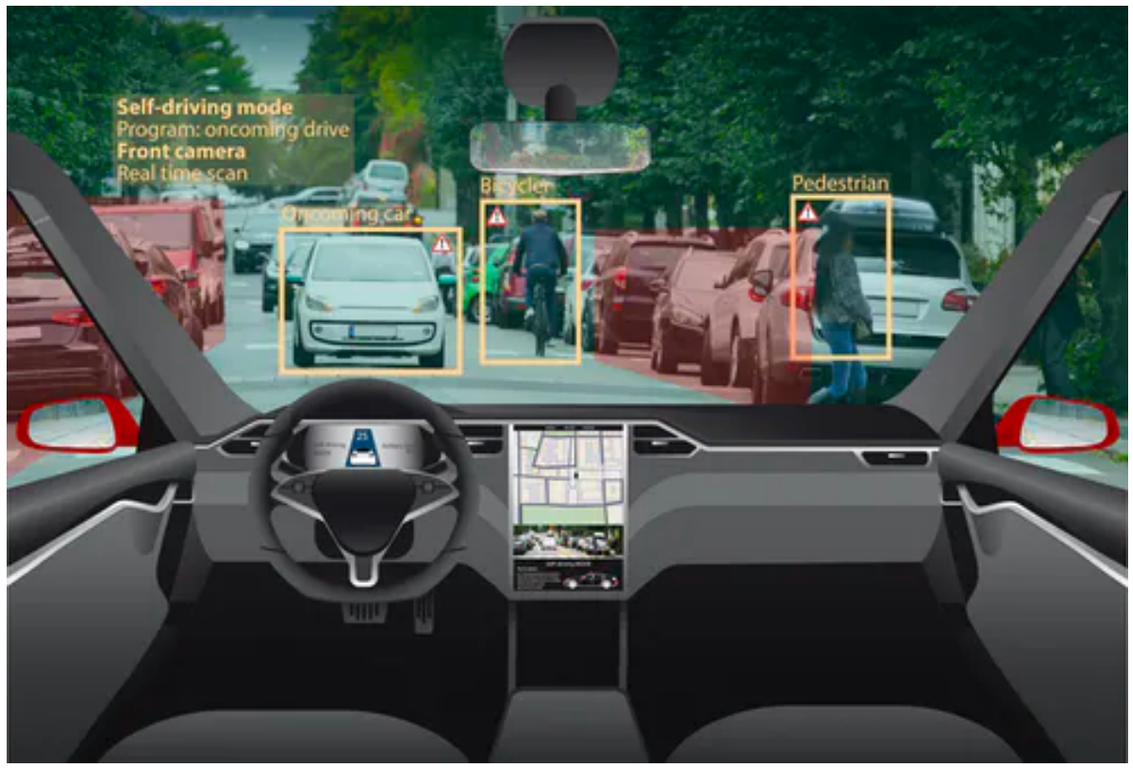
Deep Learning

MIT_8.S50



Dylan Rankin

January 14th, 2022

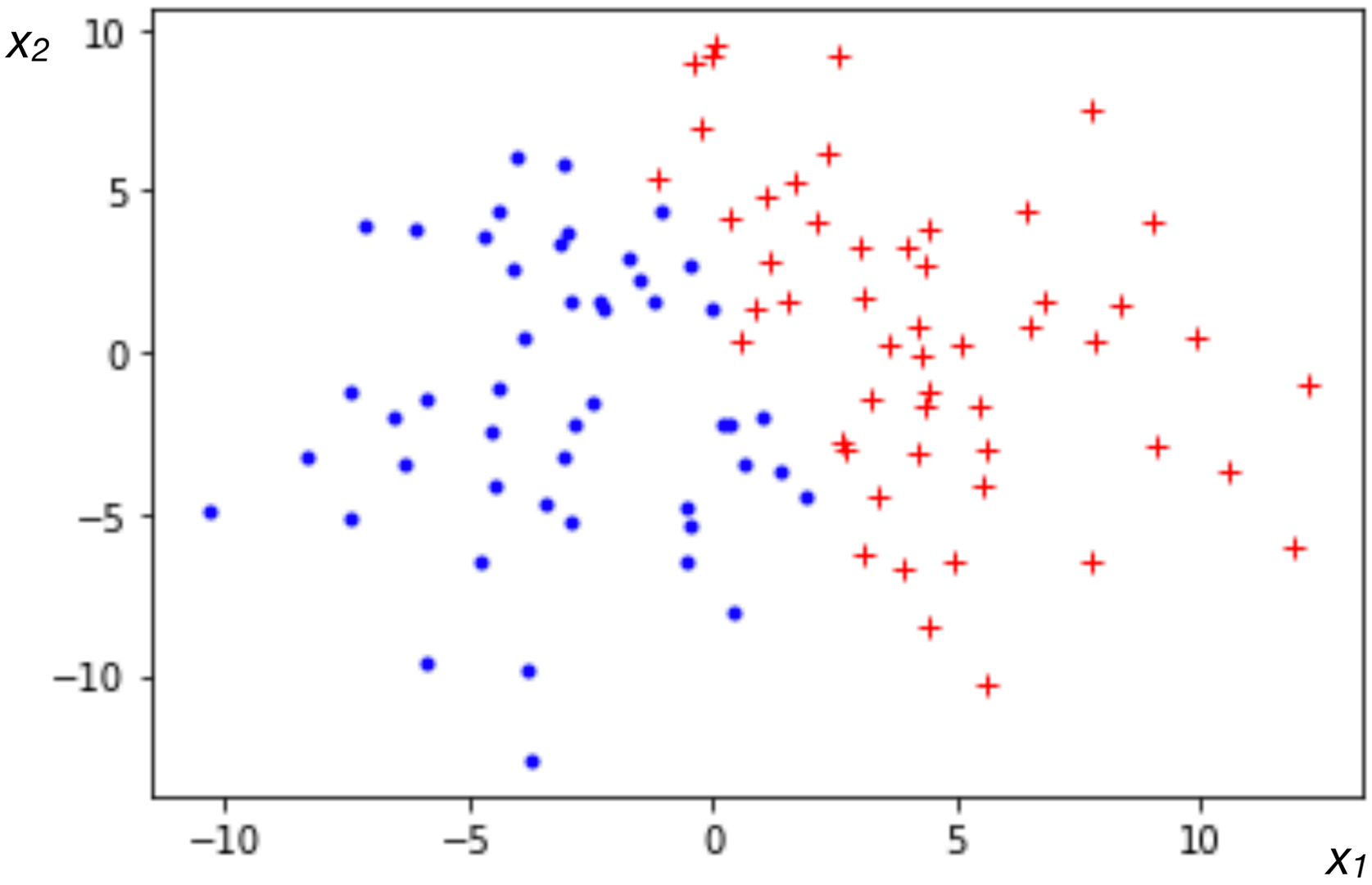
What Will We Cover Today

- What is a neural network?
  - Historical context
  - Why do they work?
- How does a neural network “learn”?
- How can neural networks be designed?

Key Terms

- Supervised Learning
    - Classification
    - Regression
- Unsupervised Learning
    - Clustering
    - Dimensional Reduction
- Architectures
    - Linear Models
        - Perceptron, support vector machine, logistic regression
    - Neural network
- Training
    - Backpropogation
    - (Stochastic) gradient descent

Simple Machine Learning

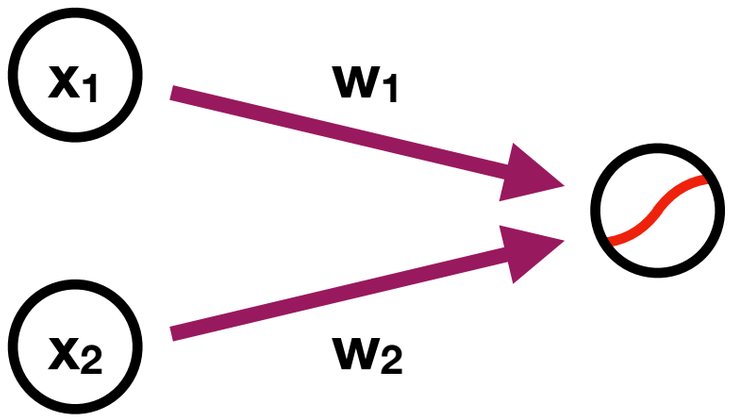


How can I predict if a point is red or blue given x_1 and x_2?

(Lets use the notebook)

Logistic Regression

- Simplest neural network
  - No hidden layers
  - Two inputs
  - One output neuron with a sigmoid activation.

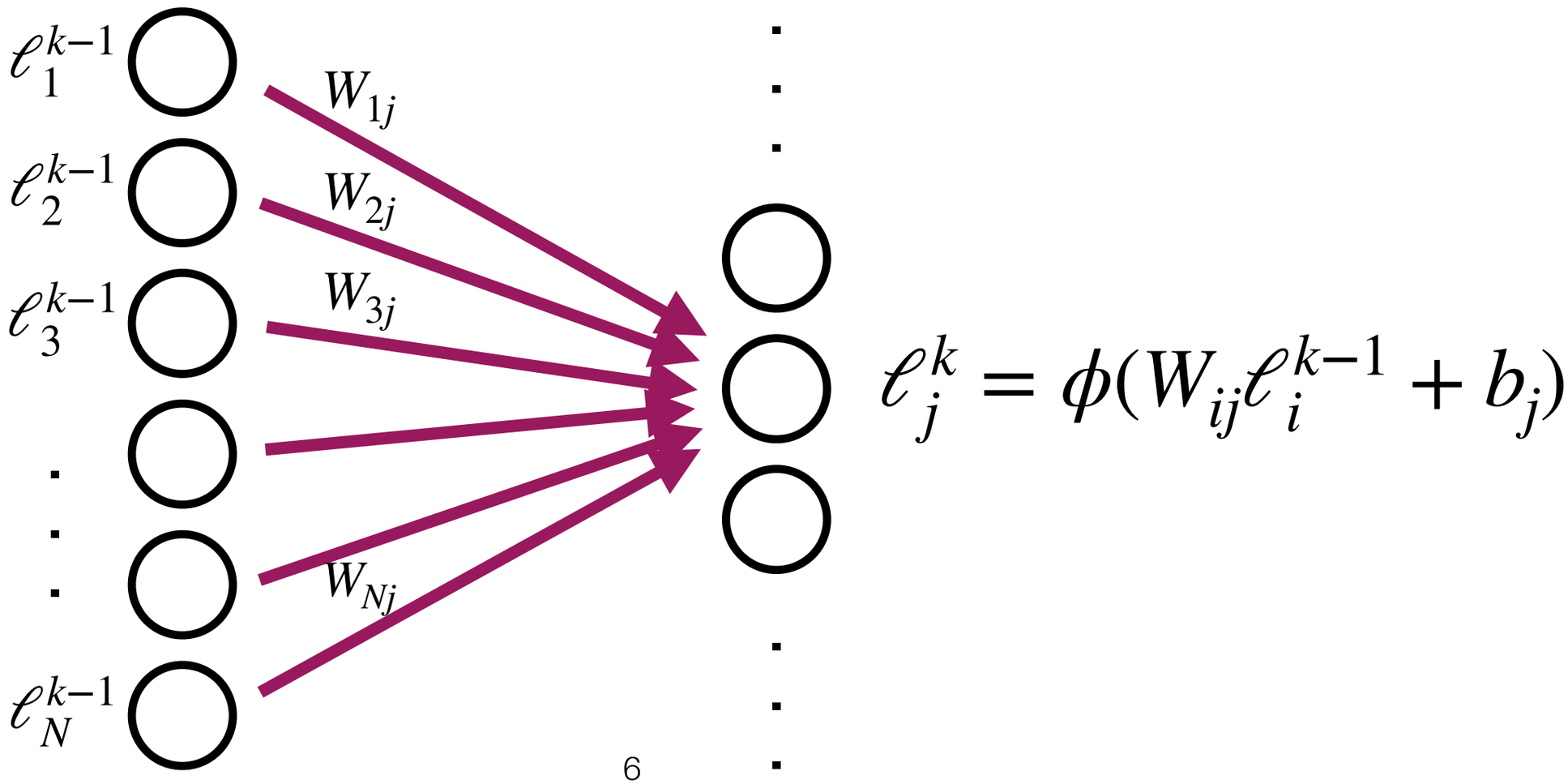


$$\ell = \frac{1}{1 + e^{\mathbf{w}^T\mathbf{x}+b}}$$

$$\mathbf{w}^T = (w_1 \; w_2)$$

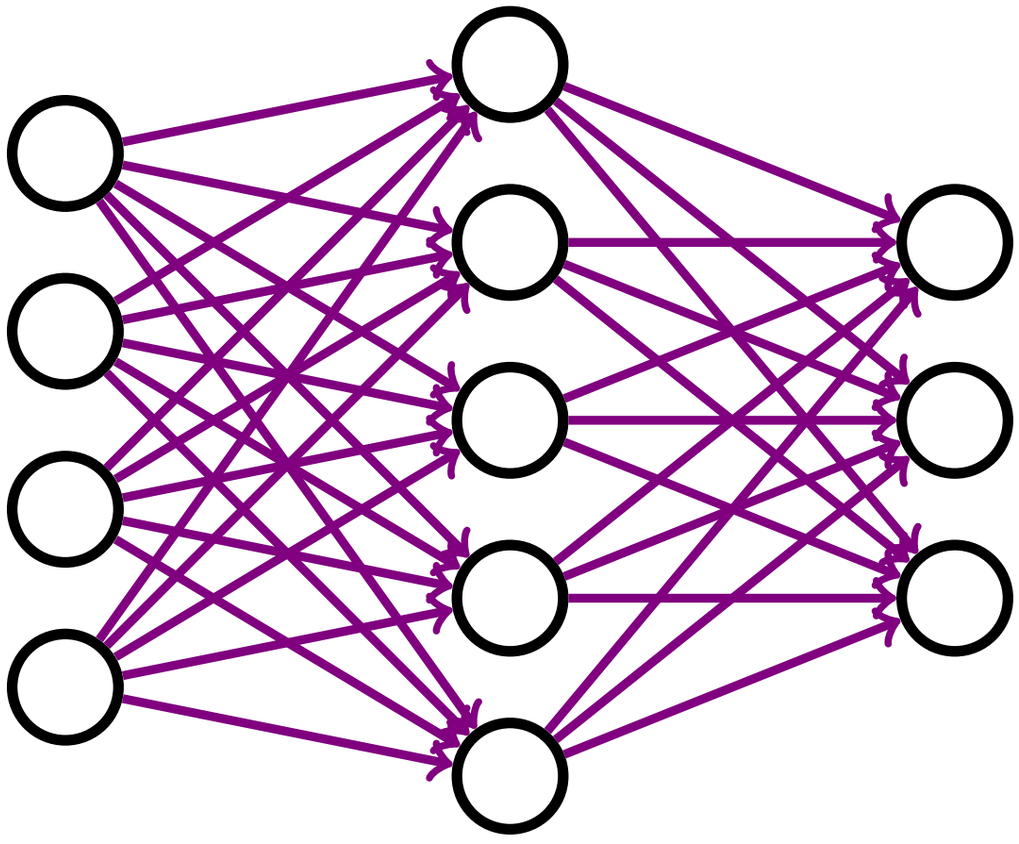
Neural Network

- Generically composed of neurons which receive inputs with *weights (W)* and a *bias (b)*, and pass outputs based on an *activation function (ϕ)*



Dense Neural Network

- **Multiple layers:** output of previous layer is **fed forward** to next layer after applying **non-linear** activation function
- **Fully connected:** many independent weights
- **Learning:** Use analytic derivatives and stochastic gradient descent to find optimal weights

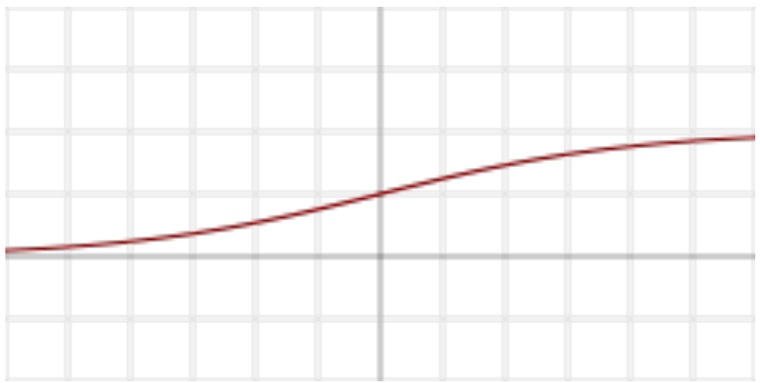


Architecture:

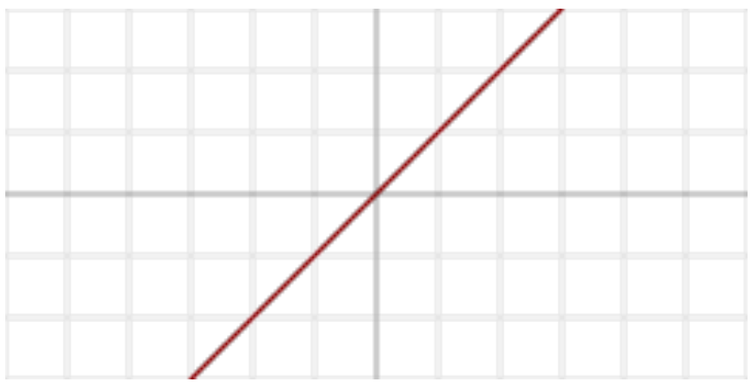
Dense Network
Fully-connected Network (FC)
Multi-Layer Perceptron (MLP)

Activation Functions

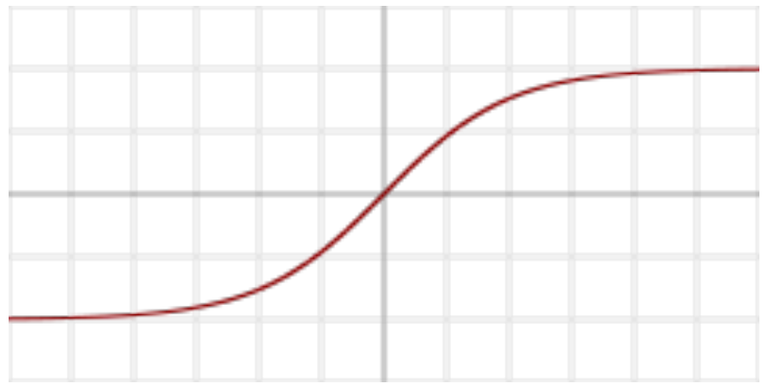
Sigmoid



Linear



Tanh

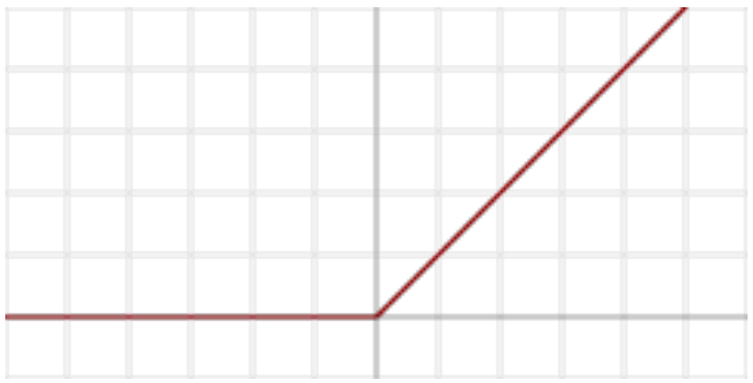


Softmax

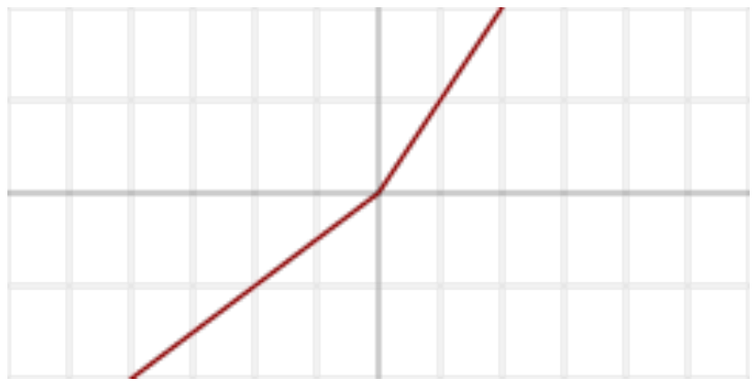
(multiclass)

$$\frac{e^{x_i}}{\sum_{j=1}^{J} e^{x_j}}$$

ReLU

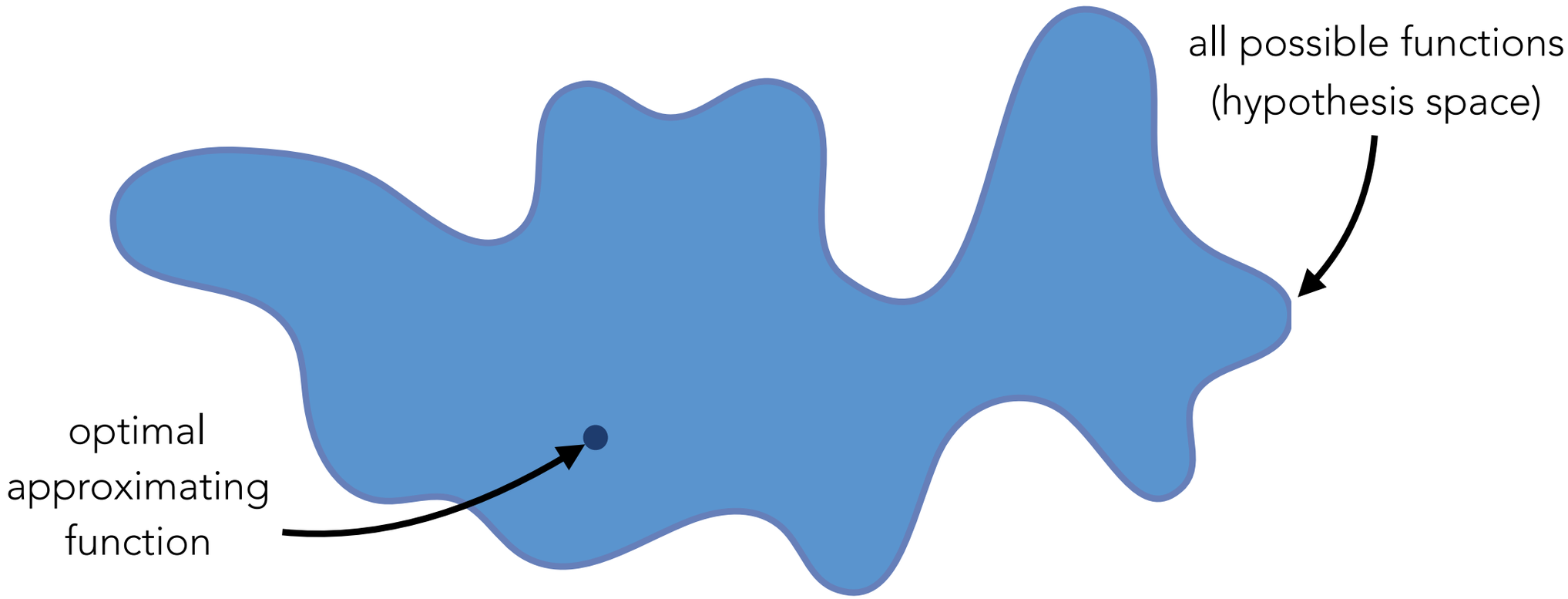


LeakyReLU



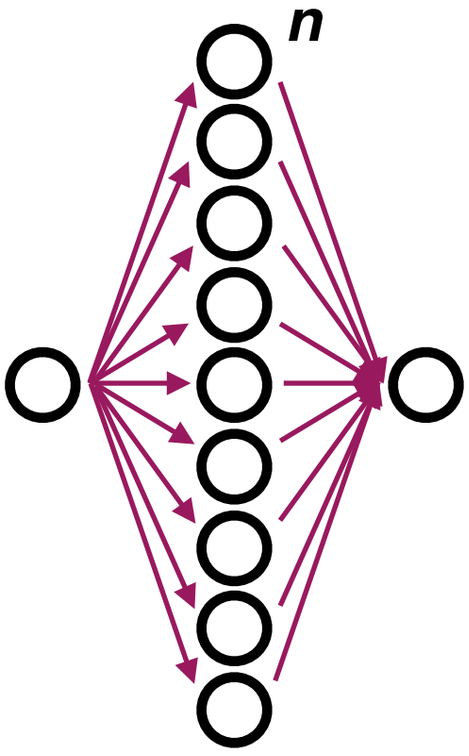
Neural Network

- Neural networks are universal function approximators, but we still must find an optimal approximating function



- We do so by adjusting the weights, architecture

Approximating



- Layer: $\ell(\mathbf{x}) = \phi(\mathbf{W}^T\mathbf{x} + \mathbf{b})$

- Linear+Linear: $\ell_{\text{linear}}(\ell_{\text{linear}}(\mathbf{x})) = \ell_{\text{linear}}(\mathbf{W}_1^T\mathbf{x} + \mathbf{b}_1)$

$$= \mathbf{W}_2^T(\mathbf{W}_1^T\mathbf{x} + \mathbf{b}_1) + \mathbf{b}_2$$

$$= \mathbf{W}_2^T\mathbf{W}_1^T\mathbf{x} + \mathbf{W}_2^T\mathbf{b}_1 + \mathbf{b}_2$$

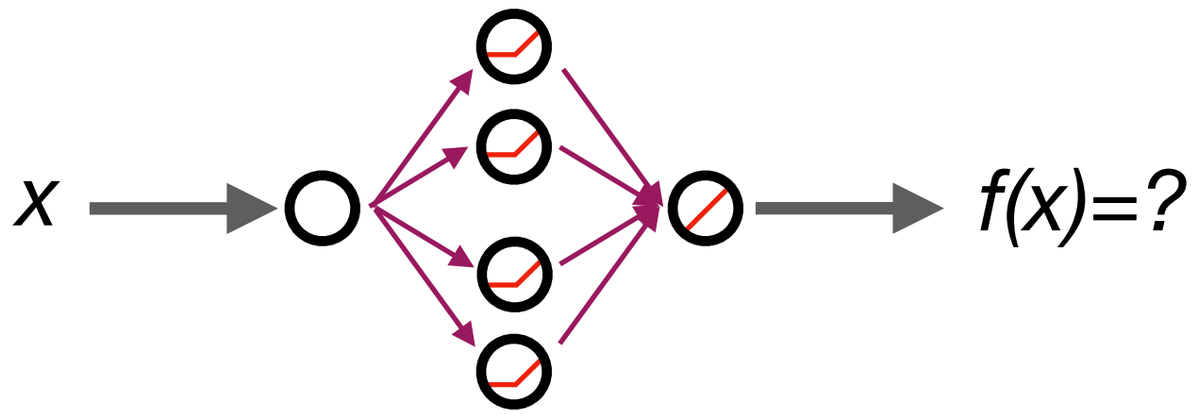
$$\tilde{\ell}_{\text{linear}}(\mathbf{x}) = \tilde{\mathbf{W}}^T\mathbf{x} + \tilde{\mathbf{b}}$$

- ReLU+Linear: $\ell_{\text{linear}}(\ell_{\text{ReLU}}(\mathbf{x})) = \tilde{\mathbf{W}}^T\mathbf{x} + \tilde{\mathbf{b}}$, $\mathbf{W}_1^T\mathbf{x} + \mathbf{b}_1 > 0$

$$\mathbf{b}_2 \text{ , } \mathbf{W}_1^T\mathbf{x} + \mathbf{b}_1 < 0$$

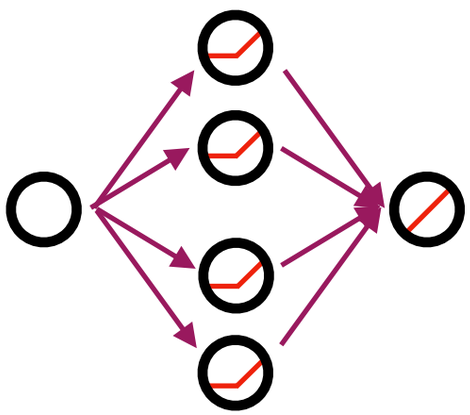
n boundaries

Approximating (Example)



What kind of function can this network architecture approximate?

Approximating (Example)



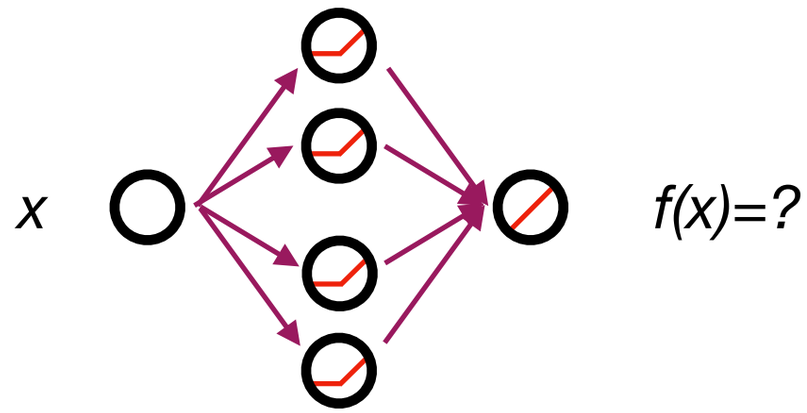
$$= \tilde{\mathbf{W}}^T\mathbf{x} + \tilde{\mathbf{b}} \;, \quad \mathbf{W}_1^T\mathbf{x} + \mathbf{b}_1 > 0$$

$$\mathbf{b}_2 \;, \quad \mathbf{W}_1^T\mathbf{x} + \mathbf{b}_1 < 0$$

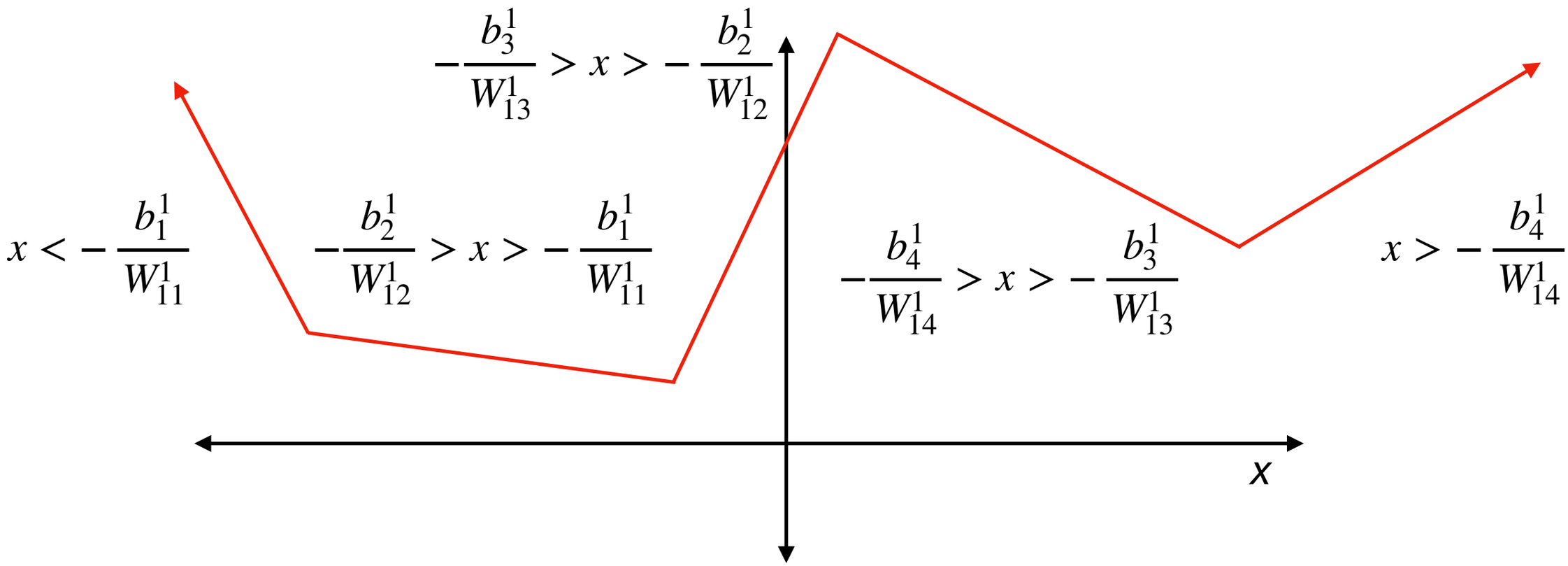
$$W_{11}^2 W_{11}^1 x + W_{11}^2 b_1^1 + W_{21}^2 W_{12}^1 x + W_{21}^2 b_2^1 + W_{31}^2 W_{13}^1 x + W_{31}^2 b_3^1 + W_{41}^2 W_{14}^1 x + W_{41}^2 b_4^1 + b_1^2$$

$x > -\frac{b_4^1}{W_{14}^1}$	$(W_{11}^2 W_{11}^1 + W_{21}^2 W_{12}^1 + W_{31}^2 W_{13}^1 + W_{41}^2 W_{14}^1)x + W_{11}^2 b_1^1 + W_{21}^2 b_2^1 + W_{31}^2 b_3^1 + W_{41}^2 b_4^1 + b_1^2$
$-\frac{b_4^1}{W_{14}^1} > x > -\frac{b_3^1}{W_{13}^1}$	$(W_{11}^2 W_{11}^1 + W_{21}^2 W_{12}^1 + W_{31}^2 W_{13}^1)x + W_{11}^2 b_1^1 + W_{21}^2 b_2^1 + W_{31}^2 b_3^1 + b_1^2$
$-\frac{b_3^1}{W_{13}^1} > x > -\frac{b_2^1}{W_{12}^1}$	$(W_{11}^2 W_{11}^1 + W_{21}^2 W_{12}^1)x + W_{11}^2 b_1^1 + W_{21}^2 b_2^1 + b_1^2$
$-\frac{b_2^1}{W_{12}^1} > x > -\frac{b_1^1}{W_{11}^1}$	$(W_{11}^2 W_{11}^1)x + W_{11}^2 b_1^1 + b_1^2$
$x < -\frac{b_1^1}{W_{11}^1}$	b_1^2

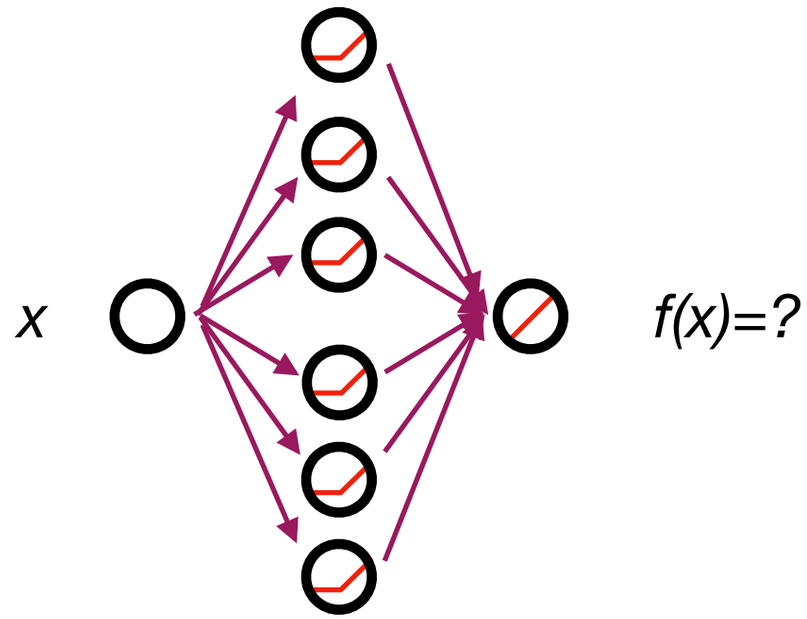
Approximating (Example)



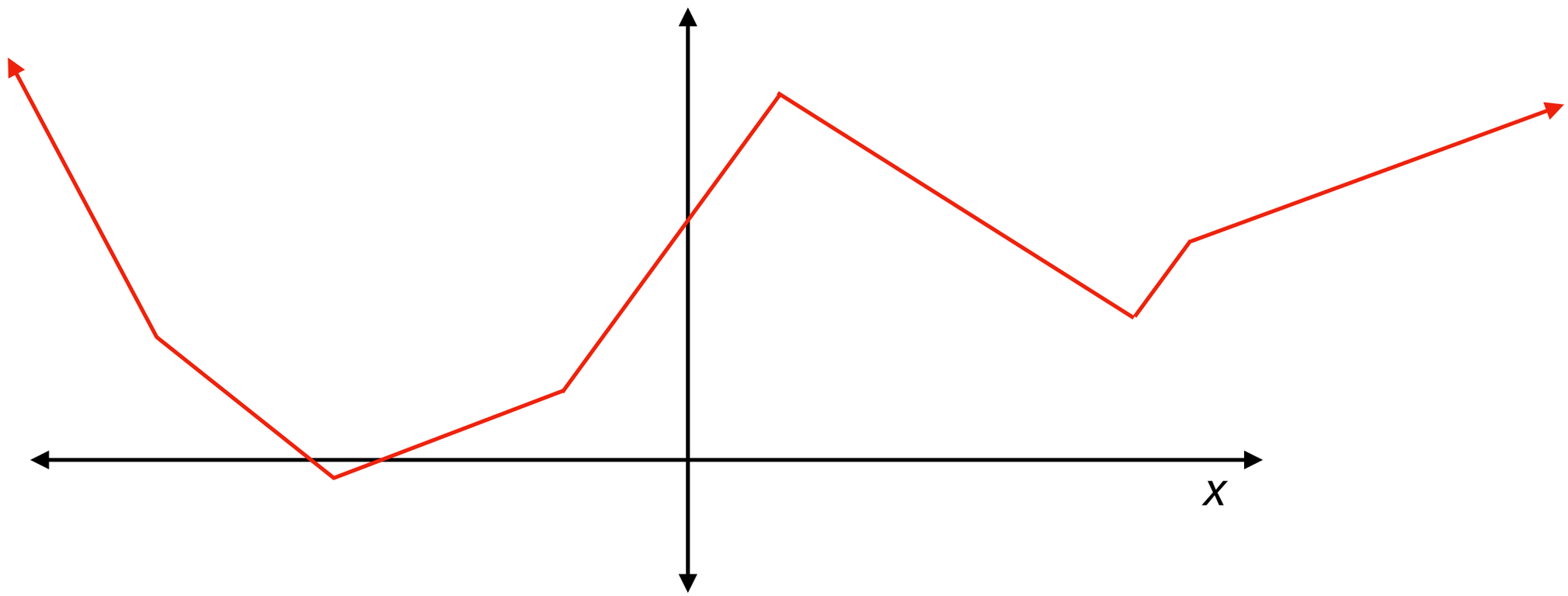
What kind of function can this network architecture approximate?



Approximating (Example)

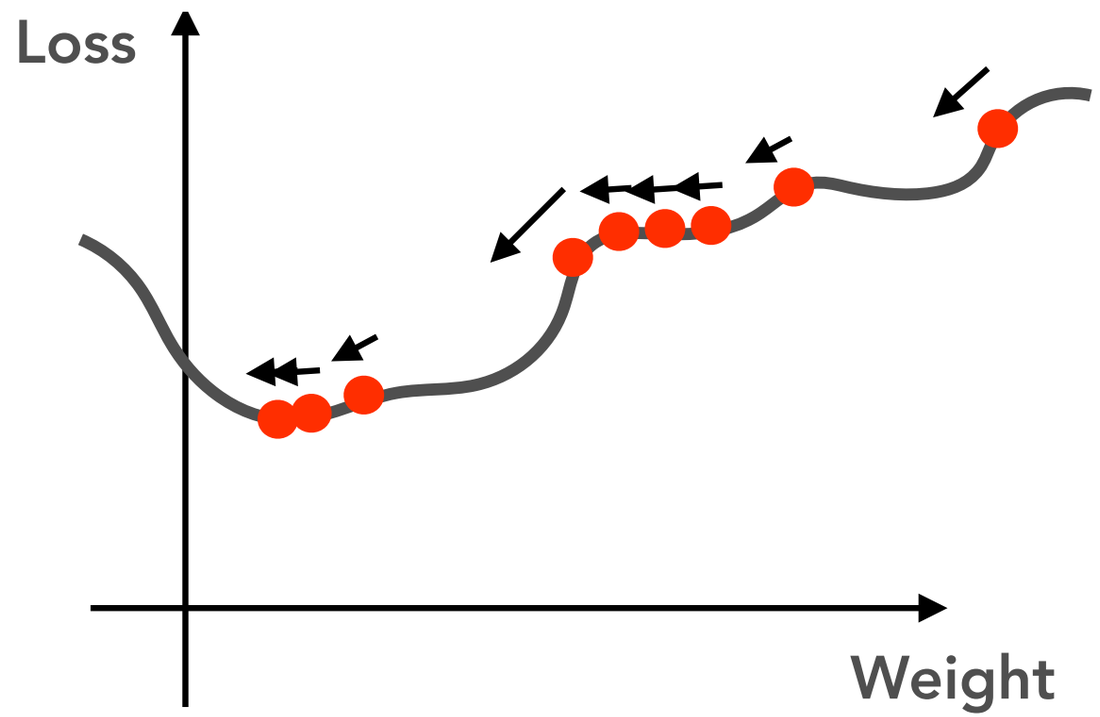


What kind of function can this network architecture approximate?

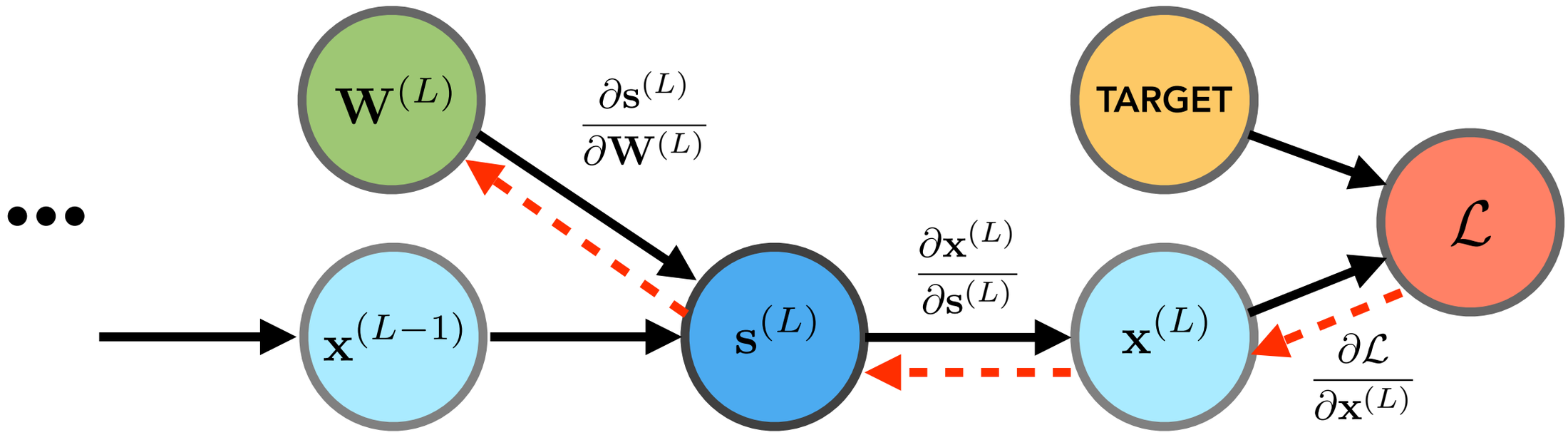


Learning - Optimization

- To learning the optimal weights we need the **derivative** of the loss w.r.t. weight
  - $w \rightarrow w - \alpha \dfrac{\partial \mathcal{L}}{\partial w}$
- With multiple weights we need the gradient of the loss w.r.t. weights
  - $\mathbf{w} \rightarrow \mathbf{w} - \alpha \nabla_{\mathbf{w}} \mathcal{L}$

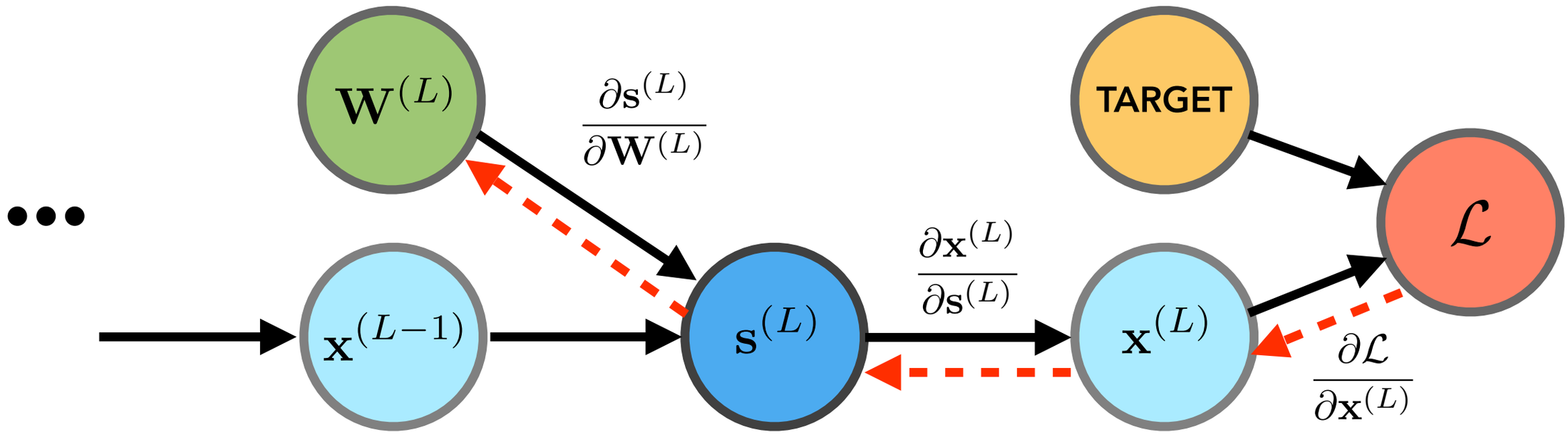


Backpropagation



- Can write a neural network as a function of composed operations
  - $\phi_L(\mathbf{w}_L, \phi_{L-1}(\mathbf{w}_{L-1}, \ldots \phi_1(\mathbf{w}_1, \mathbf{x}) \ldots))$
- The loss is a function of the network output → use chain rule to calculate gradients

Backpropagation



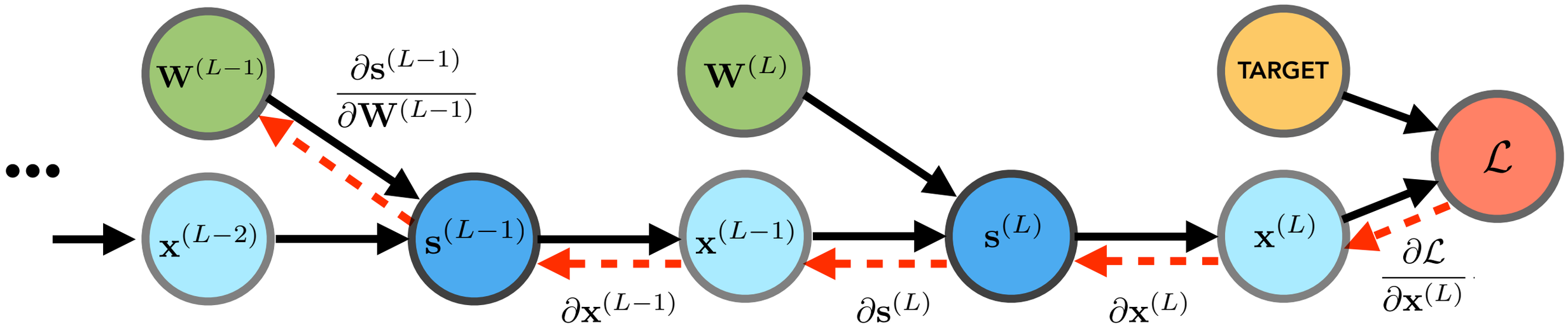
$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}^{(L)}} = \frac{\partial \mathcal{L}}{\partial \mathbf{x}^{(L)}} \frac{\partial \mathbf{x}^{(L)}}{\partial \mathbf{s}^{(L)}} \frac{\partial \mathbf{s}^{(L)}}{\partial \mathbf{W}^{(L)}}$$

depends on the form of the loss

derivative of the non-linearity

$$\frac{\partial}{\partial \mathbf{W}^{(L)}}(\mathbf{W}^{(L)\intercal}\mathbf{x}^{(L-1)}) = \mathbf{x}^{(L-1)\intercal}$$

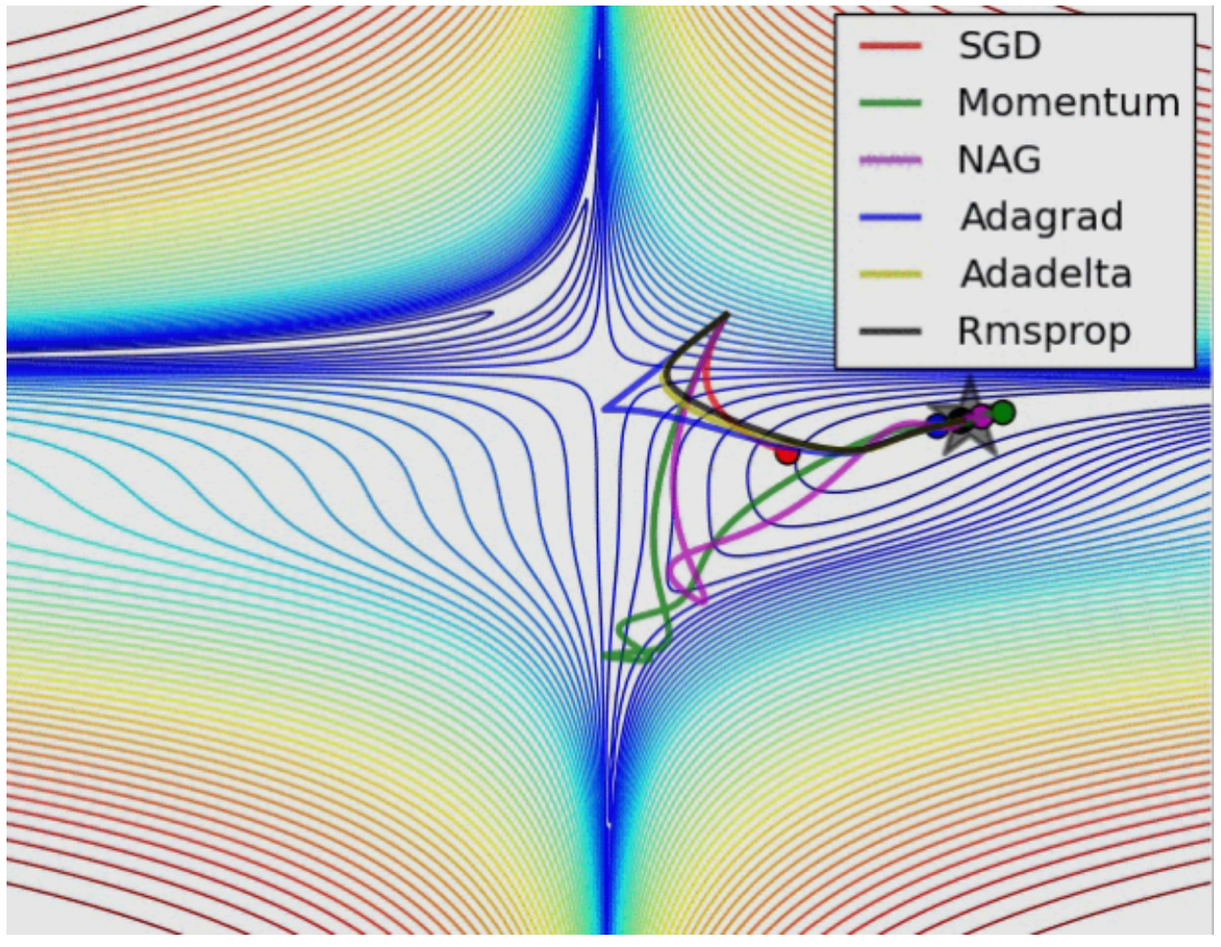
Backpropagation



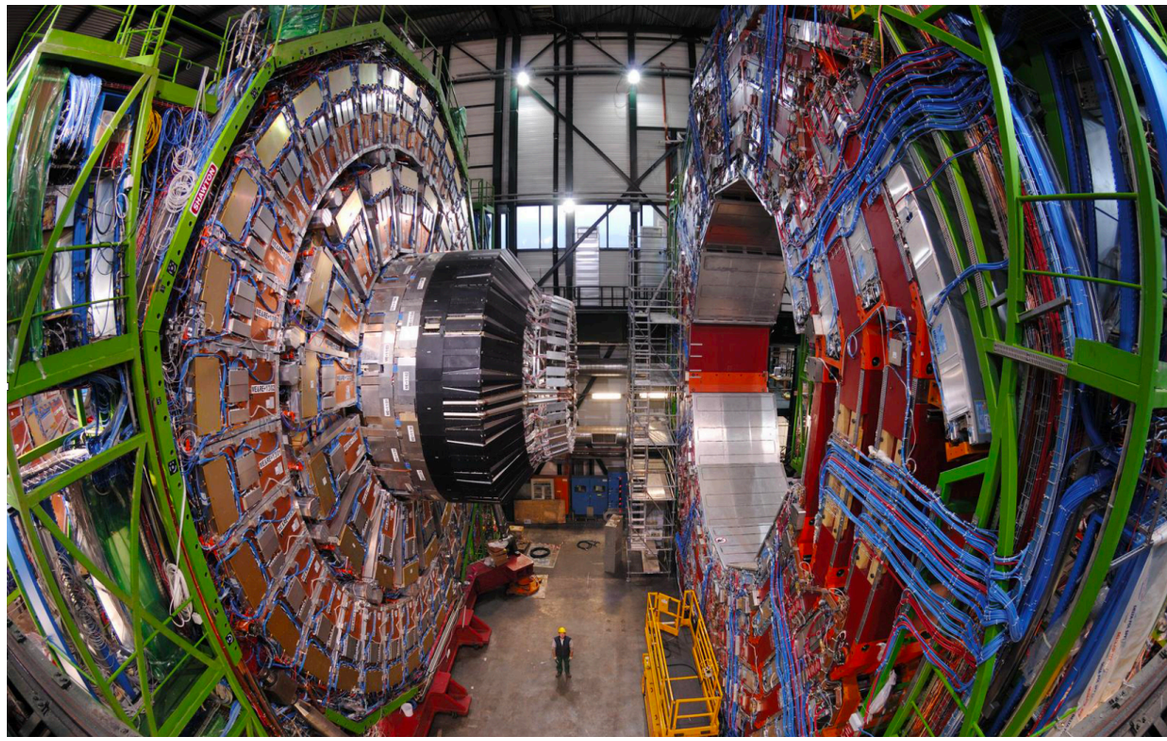
$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}^{(L)}} = \frac{\partial \mathcal{L}}{\partial \mathbf{x}^{(L)}} \frac{\partial \mathbf{x}^{(L)}}{\partial \mathbf{s}^{(L)}} \frac{\partial \mathbf{s}^{(L)}}{\partial \mathbf{x}^{(L-1)}} \frac{\partial \mathbf{x}^{(L-1)}}{\partial \mathbf{s}^{(L-1)}} \frac{\partial \mathbf{s}^{(L-1)}}{\partial \mathbf{W}^{(L-1)}}$$

Stochastic Gradient Descent

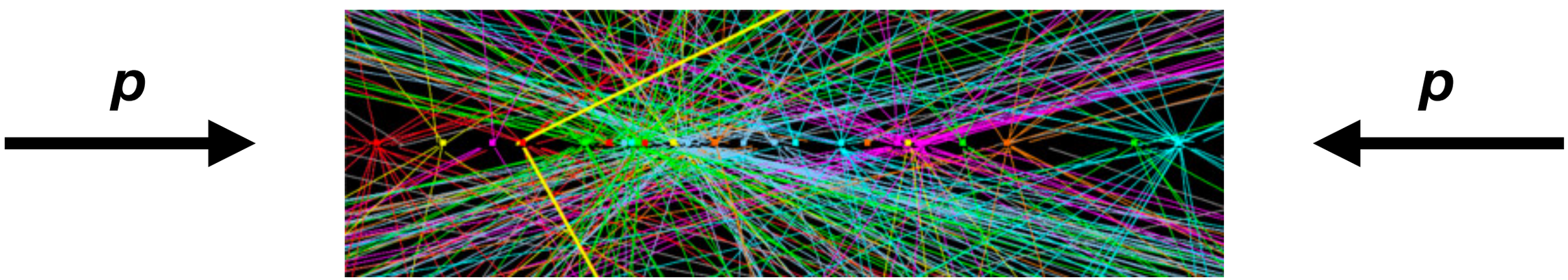
- SGD: $w = w - \alpha \tilde{\nabla}_w \mathscr{L}$
  - Use estimate to traverse the loss function
- Advanced estimates use “memory”, other optimizations
- Able to handle large dimensionality, complex surfaces (saddle points, local minima)



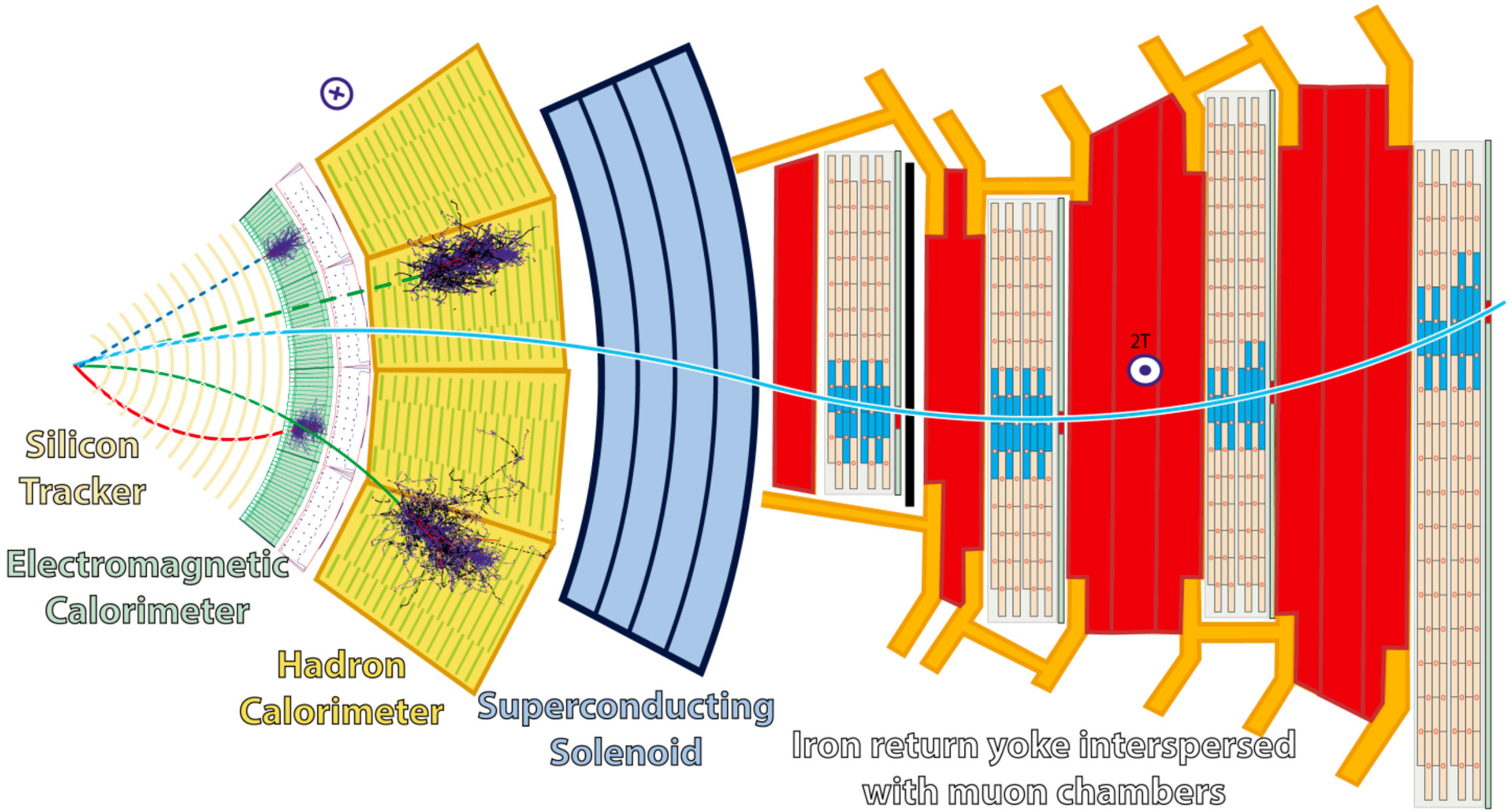
Large Hadron Collider



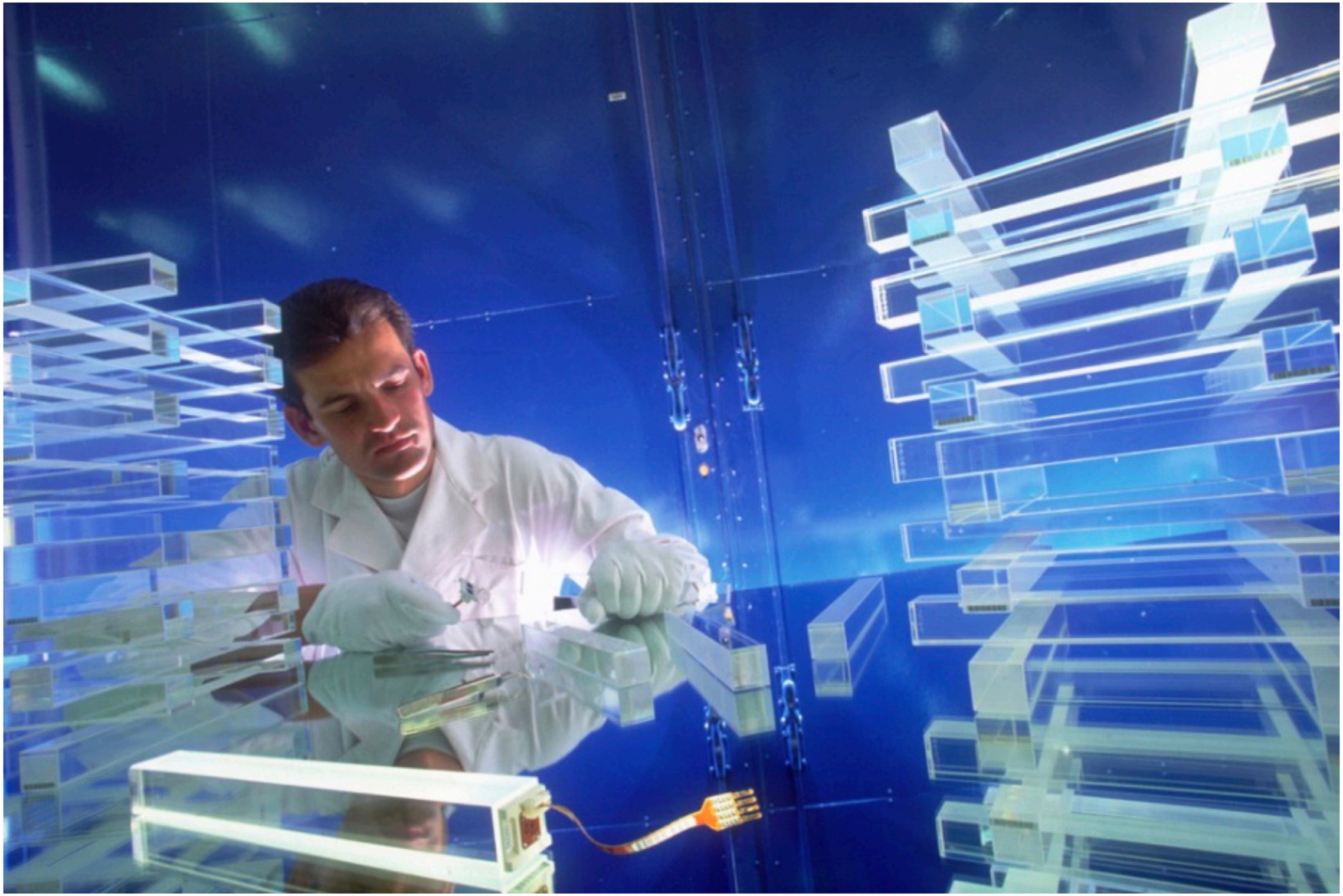
- Large Hadron Collider (proton - proton)
- Dedicated detectors to record data from collisions
  - Electromagnetic calorimeter (ECAL) is designed to measure energy of EM particles, **e + γ (EG)**
- Many overlapping collisions in addition to the primary one, called **pileup (PU)**
- Want to design algorithm to distinguish primary **EG** energy deposits from **PU**



CMS Detector

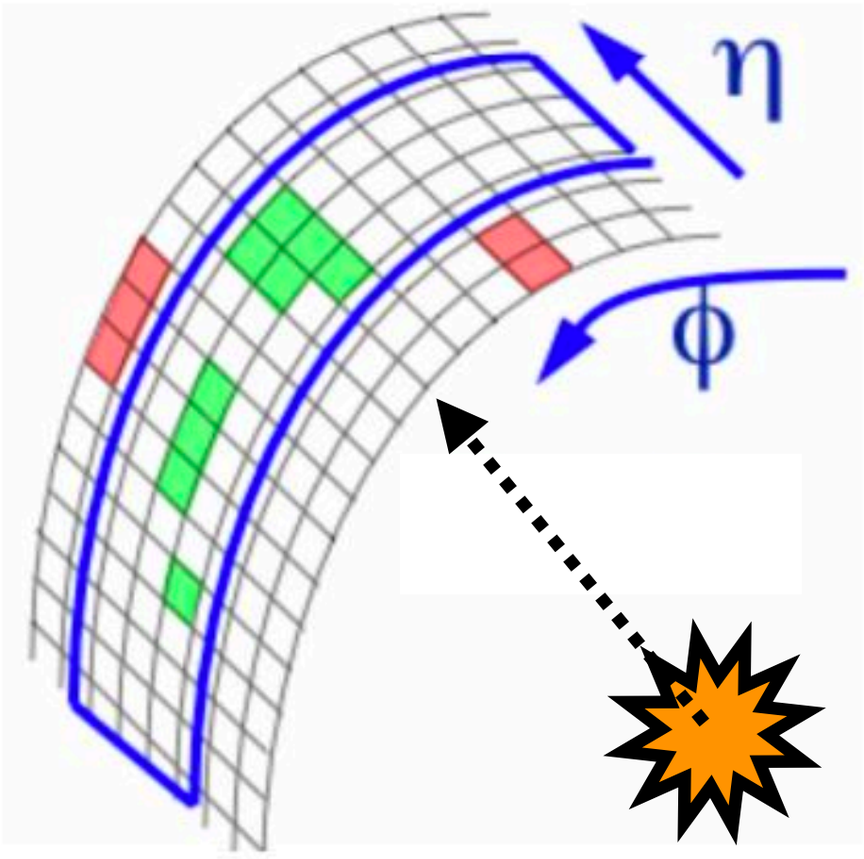


CMS ECAL

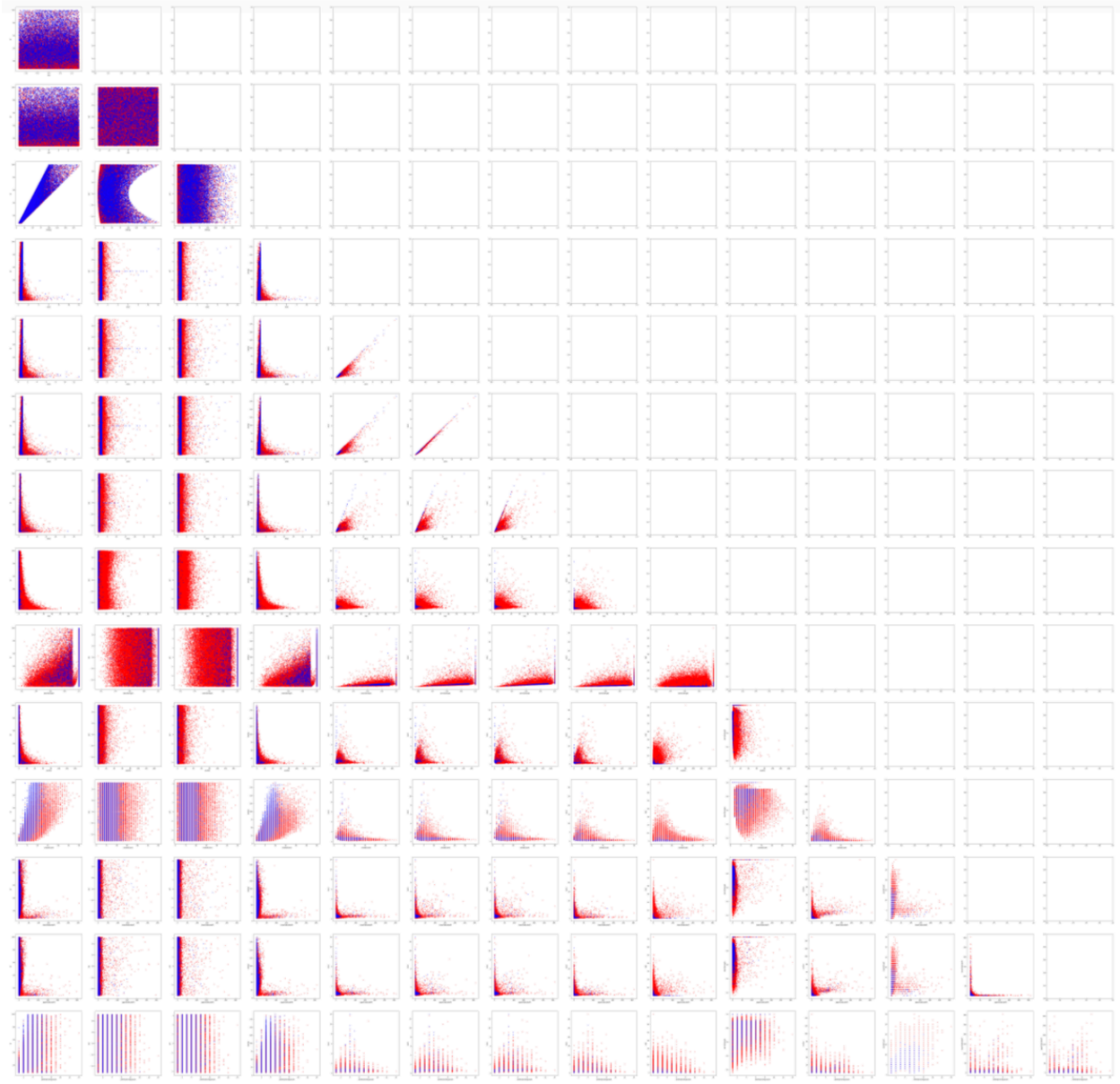


Dataset Explanation

- 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?



Designing ML



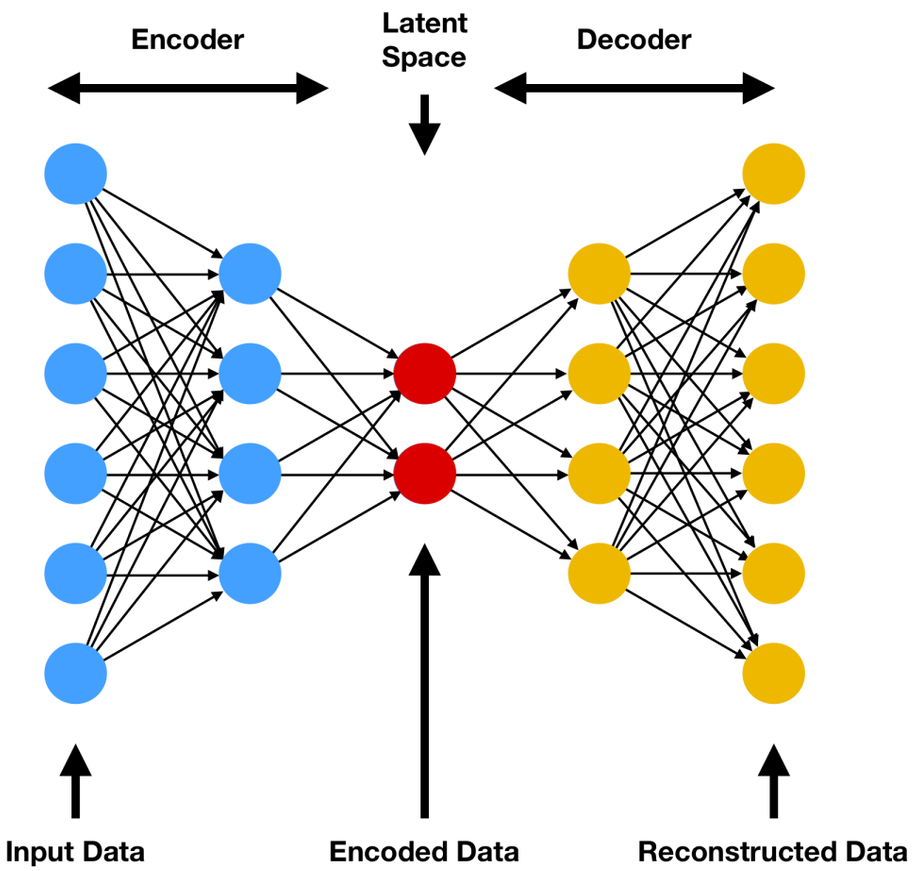
How can I predict if a point is red or blue given x_1 and x_2 and x_3 and …?

(Lets use the notebook)

More Complex Architectures

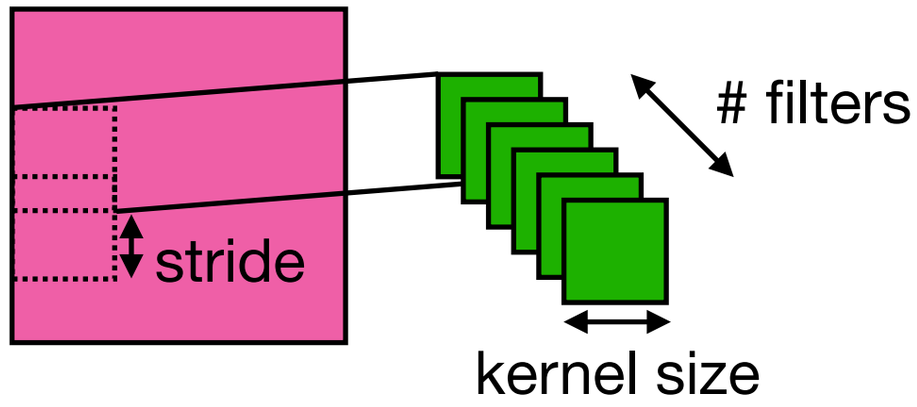
Unsupervised Learning

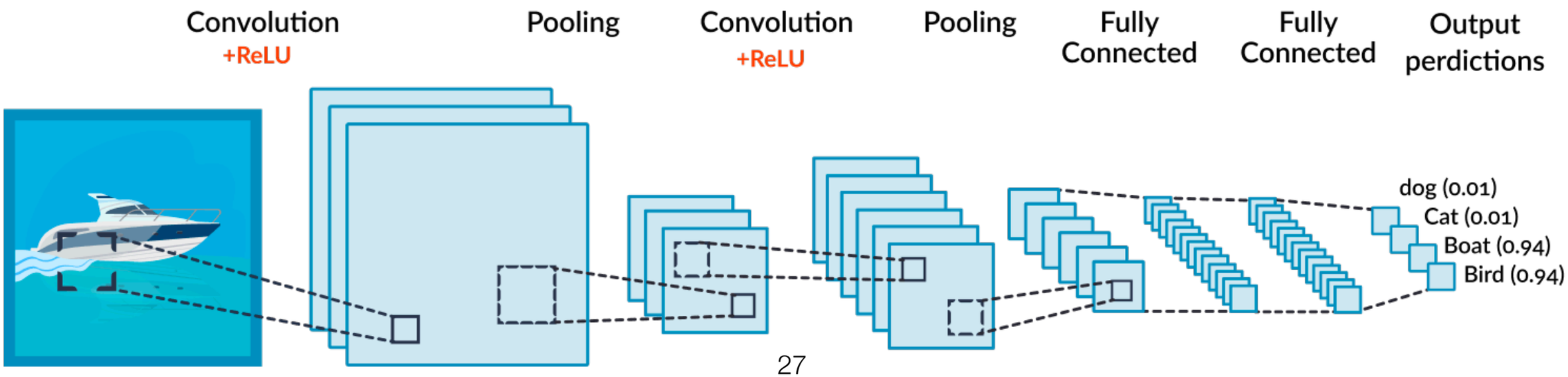
- What if we don’t have/use labels?
- Autoencoder (AE):
  - *Loss = output - input*
  - Latent space can be used for clustering
  - If one class of data is used to train, different classes may not reconstruct well (anomaly detection)



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)