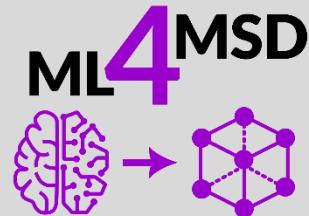


ME 5374-ST



Machine Learning for Materials Science and Discovery

Fall 2025

Asst. Prof. Peter Schindler

Lecture 15 – Deep Learning

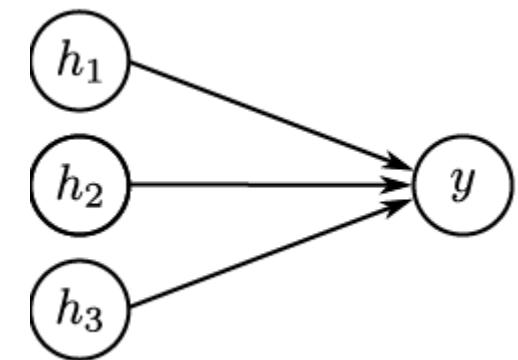
- Shallow and Deep Neural Networks
- Universal Approximation Theorem
- Activation Functions, Backpropagation, and Stochastic Gradient Descent
- Hyperparameters and Regularization of Neural Networks
- Convolutional and Pooling Layers
- Deep Learning in Materials Science and Chemistry
- Graph Neural Networks, Equivariant Models, Compositional Deep Neural Networks

Artificial vs. Real Neuron

Neural networks are *loosely based* on how neurons in the brain pass information.

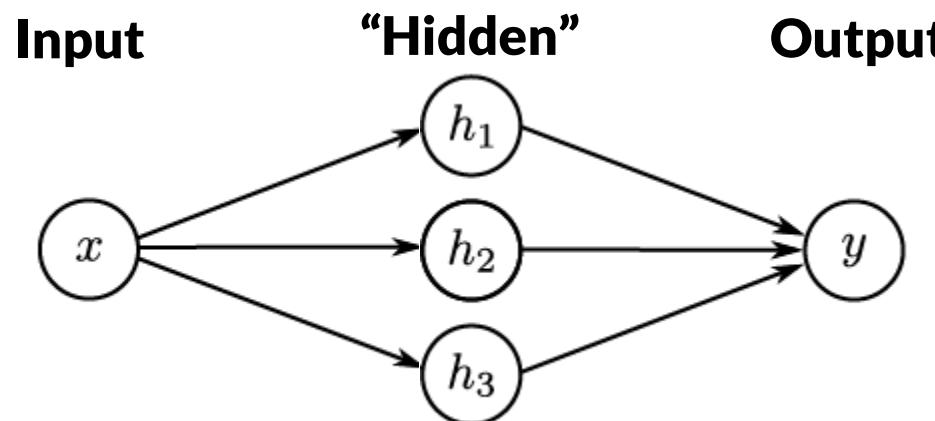
However, there are a few major differences:

- **Size:** 86 billions vs. 10-1000
- **Topology:**
Brain neurons not really connected (synapses = gaps)
- **Speed:**
Much faster in computer, and no rest periods.
- **Fault tolerance:**
No redundant info wired into artificial NN system
- **Power consumption:**
Brain runs at 20W, NVidia GPUs is >200W.
- **Signal:** 0 or 1 for brain, no intermediate)



$$y = \phi_0 + \phi_1 h_1 + \phi_2 h_2 + \phi_3 h_3.$$

Shallow Neural Network



$$y = \phi_0 + \phi_1 h_1 + \phi_2 h_2 + \phi_3 h_3.$$

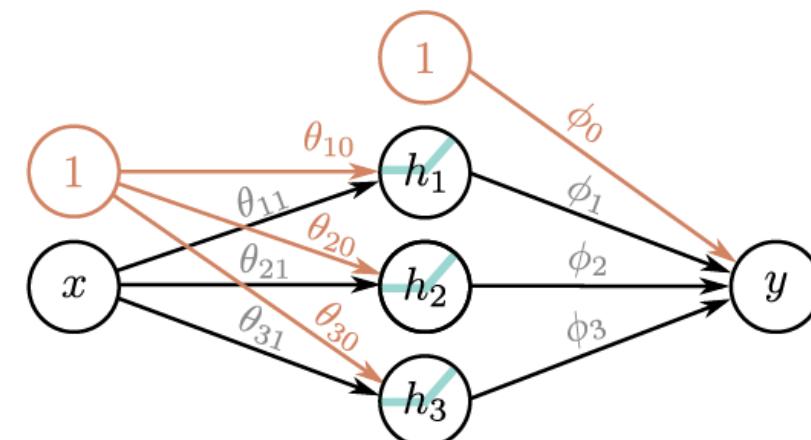
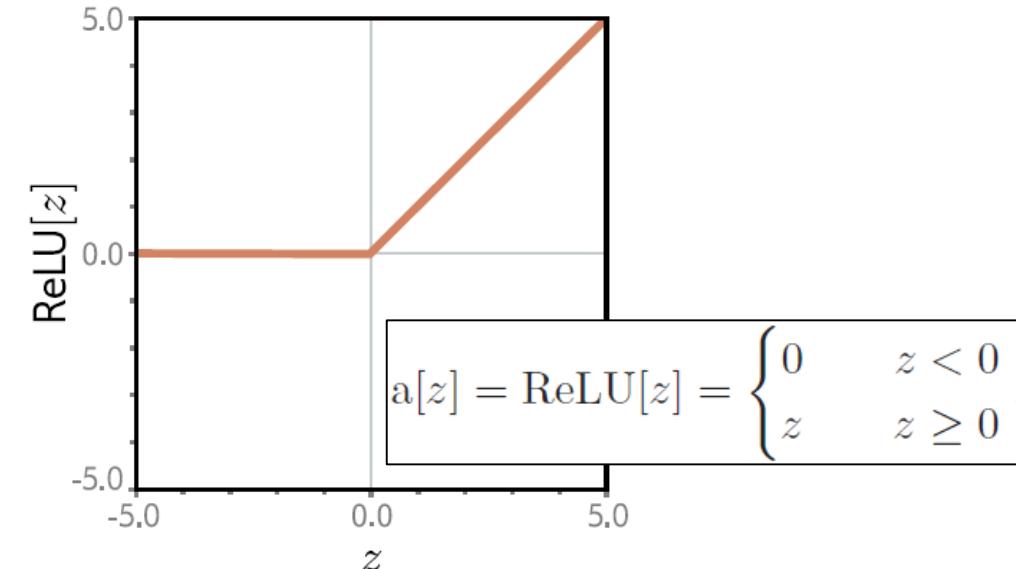
$$h_1 = a[\theta_{10} + \theta_{11}x]$$

$$h_2 = a[\theta_{20} + \theta_{21}x]$$

$$h_3 = a[\theta_{30} + \theta_{31}x],$$

$$y = f[x, \phi]$$

$$= \phi_0 + \phi_1 a[\theta_{10} + \theta_{11}x] + \phi_2 a[\theta_{20} + \theta_{21}x] + \phi_3 a[\theta_{30} + \theta_{31}x].$$

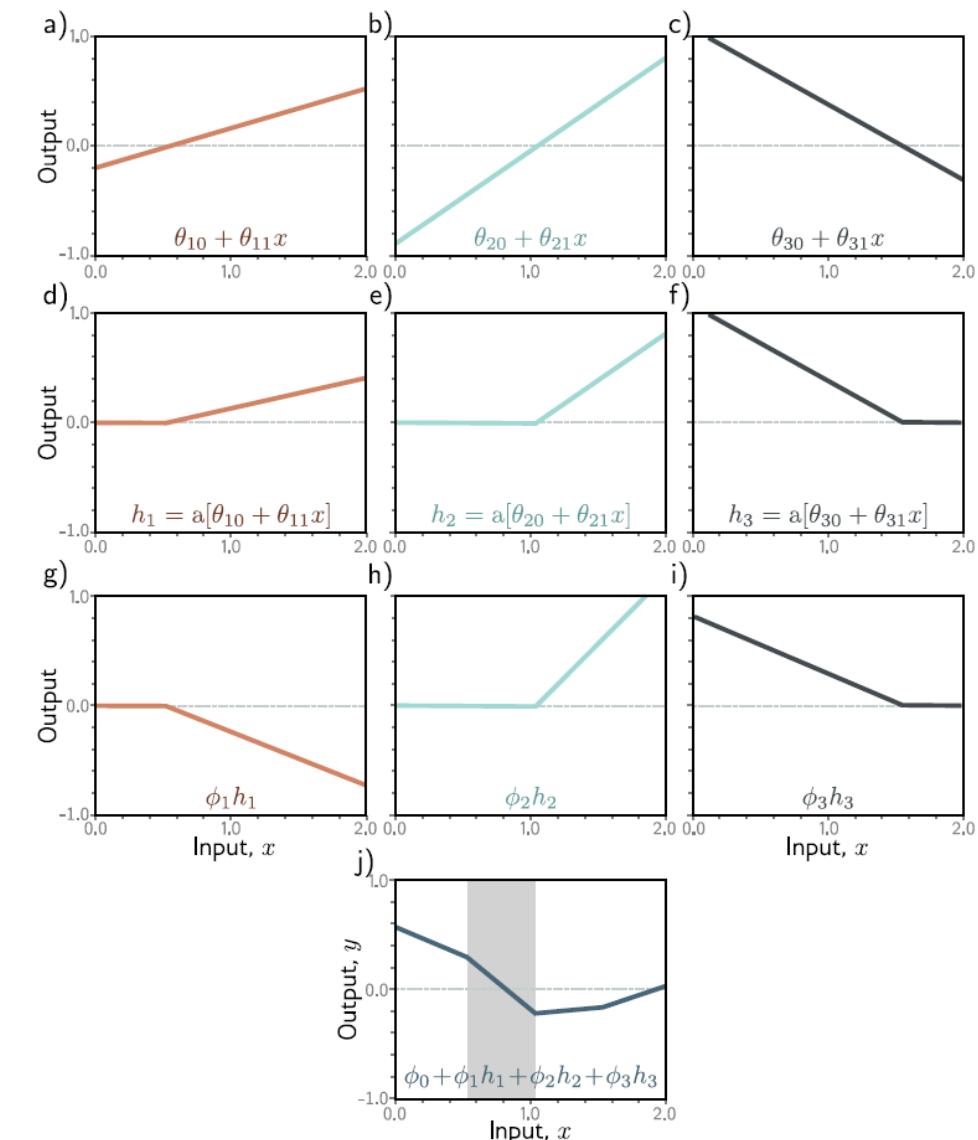


Shallow Neural Network

Piecewise linear function approximation

Interactive illustration:

<https://udlbook.github.io/udlfigures/>

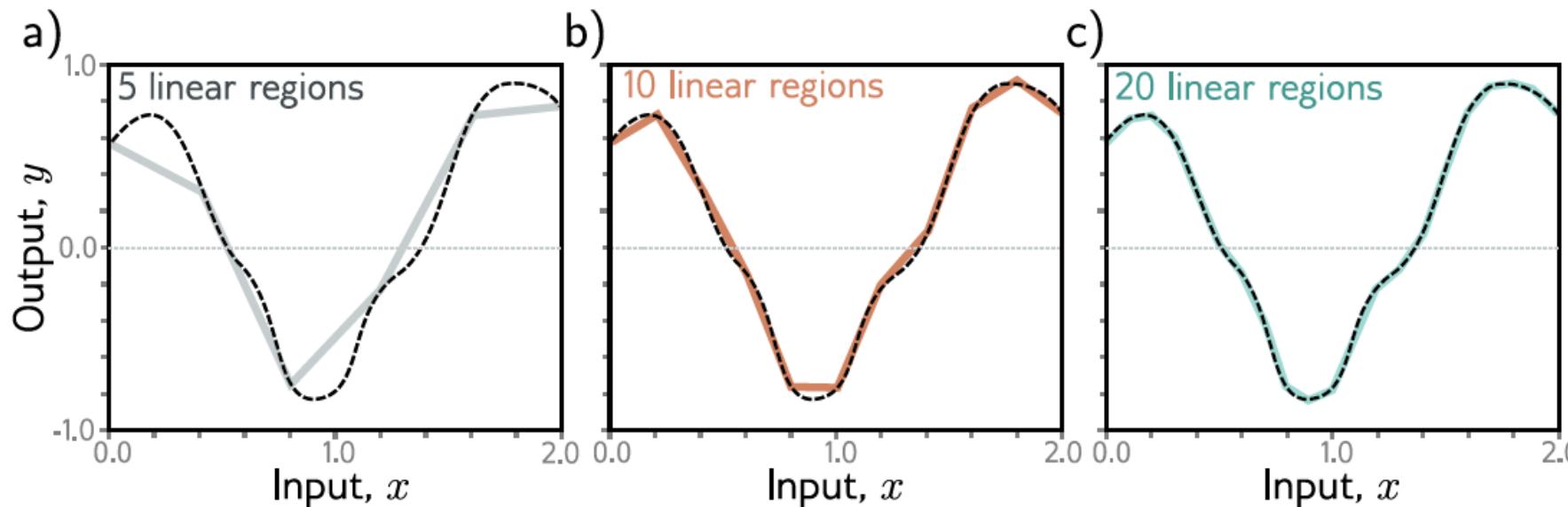


Neural Networks: Universal Approximation Theorem

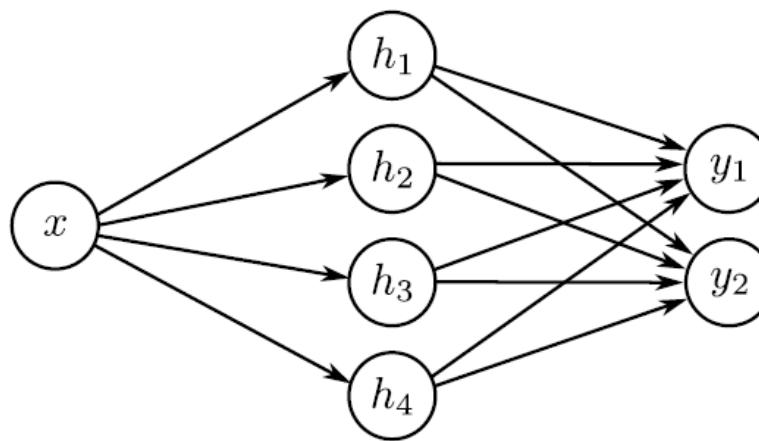
Assume D hidden units
(=measure of *network capacity*)

$$y = \phi_0 + \sum_{d=1}^D \phi_d h_d \quad h_d = a[\theta_{d0} + \theta_{d1}x]$$

With enough capacity (hidden units), a shallow network can describe any continuous 1D function defined on a compact subset of the real line to arbitrary precision



Multiple Inputs or Outputs



$$h_1 = a[\theta_{10} + \theta_{11}x]$$

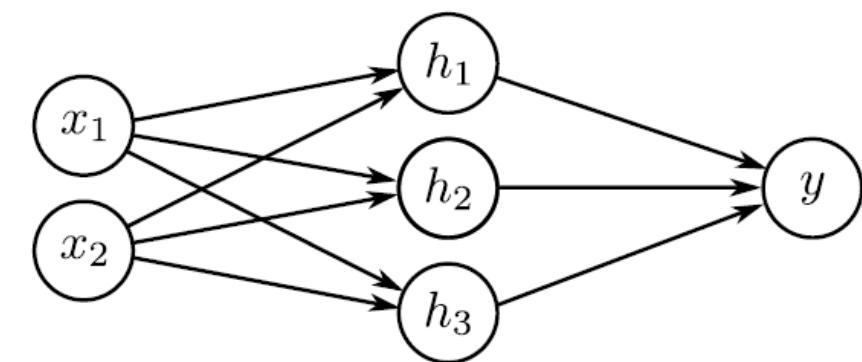
$$h_2 = a[\theta_{20} + \theta_{21}x]$$

$$h_3 = a[\theta_{30} + \theta_{31}x]$$

$$h_4 = a[\theta_{40} + \theta_{41}x]$$

$$y_1 = \phi_{10} + \phi_{11}h_1 + \phi_{12}h_2 + \phi_{13}h_3 + \phi_{14}h_4$$

$$y_2 = \phi_{20} + \phi_{21}h_1 + \phi_{22}h_2 + \phi_{23}h_3 + \phi_{24}h_4$$



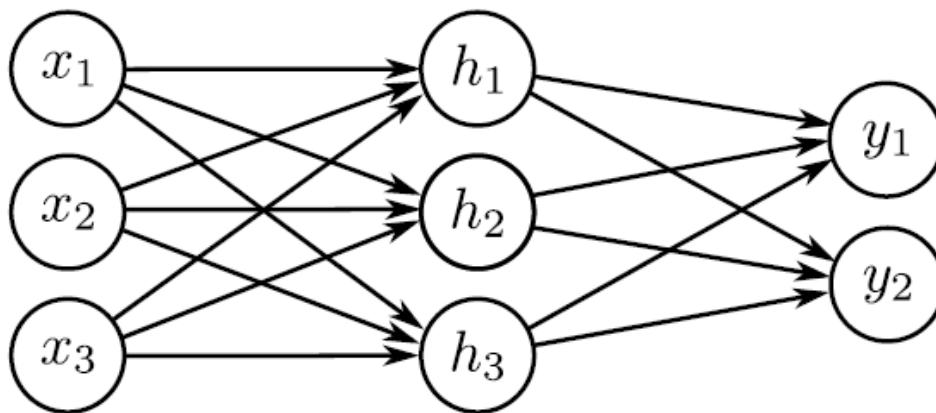
$$h_1 = a[\theta_{10} + \theta_{11}x_1 + \theta_{12}x_2]$$

$$h_2 = a[\theta_{20} + \theta_{21}x_1 + \theta_{22}x_2]$$

$$h_3 = a[\theta_{30} + \theta_{31}x_1 + \theta_{32}x_2]$$

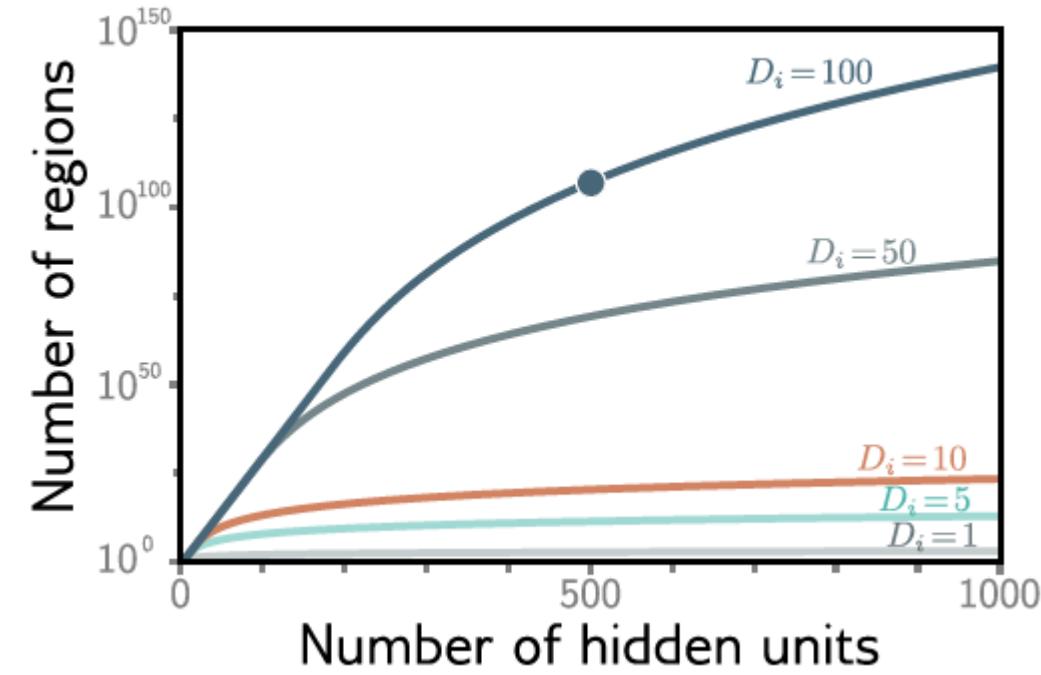
$$y = \phi_0 + \phi_1h_1 + \phi_2h_2 + \phi_3h_3$$

Shallow Neural Network – General Case



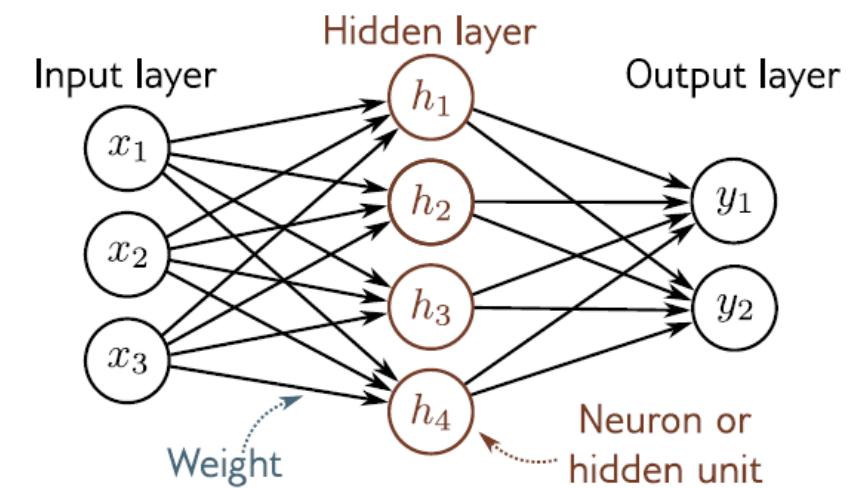
$$y_j = \phi_{j0} + \sum_{d=1}^D \phi_{jd} h_d$$

$$h_d = a \left[\theta_{d0} + \sum_{i=1}^{D_i} \theta_{di} x_i \right]$$

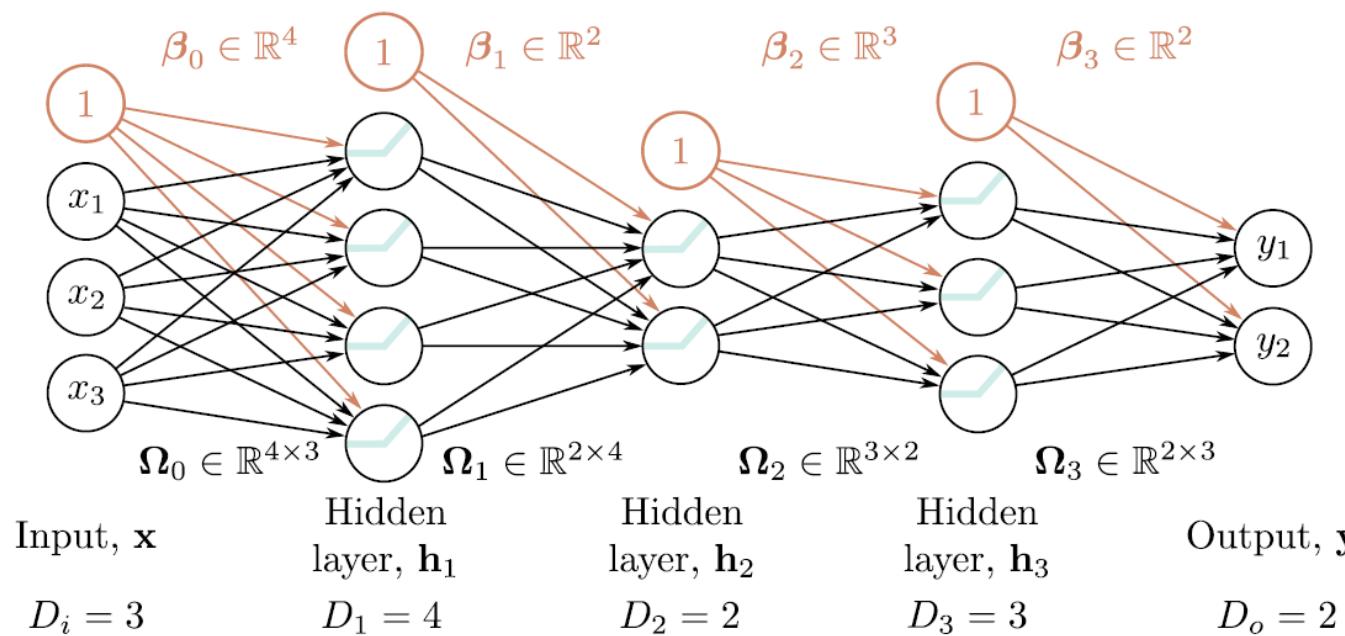


Neural Network Terminology

- “Layers”: Input, Hidden, and Output Layers
- NNs with at least 1 hidden layer is called a *multi-layer perceptron* (MLP)
- Neurons get “activated” (non-zero value)
- 1 hidden layer → *Shallow Neural Network*
- 2+ hidden layers → *Deep Neural Network*
- NNs in which the connections form a graph without loops are termed “Feed-forward NNs”
- If every element in one layer connects to every element in the next, the network is “fully connected”.



Deep Neural Networks



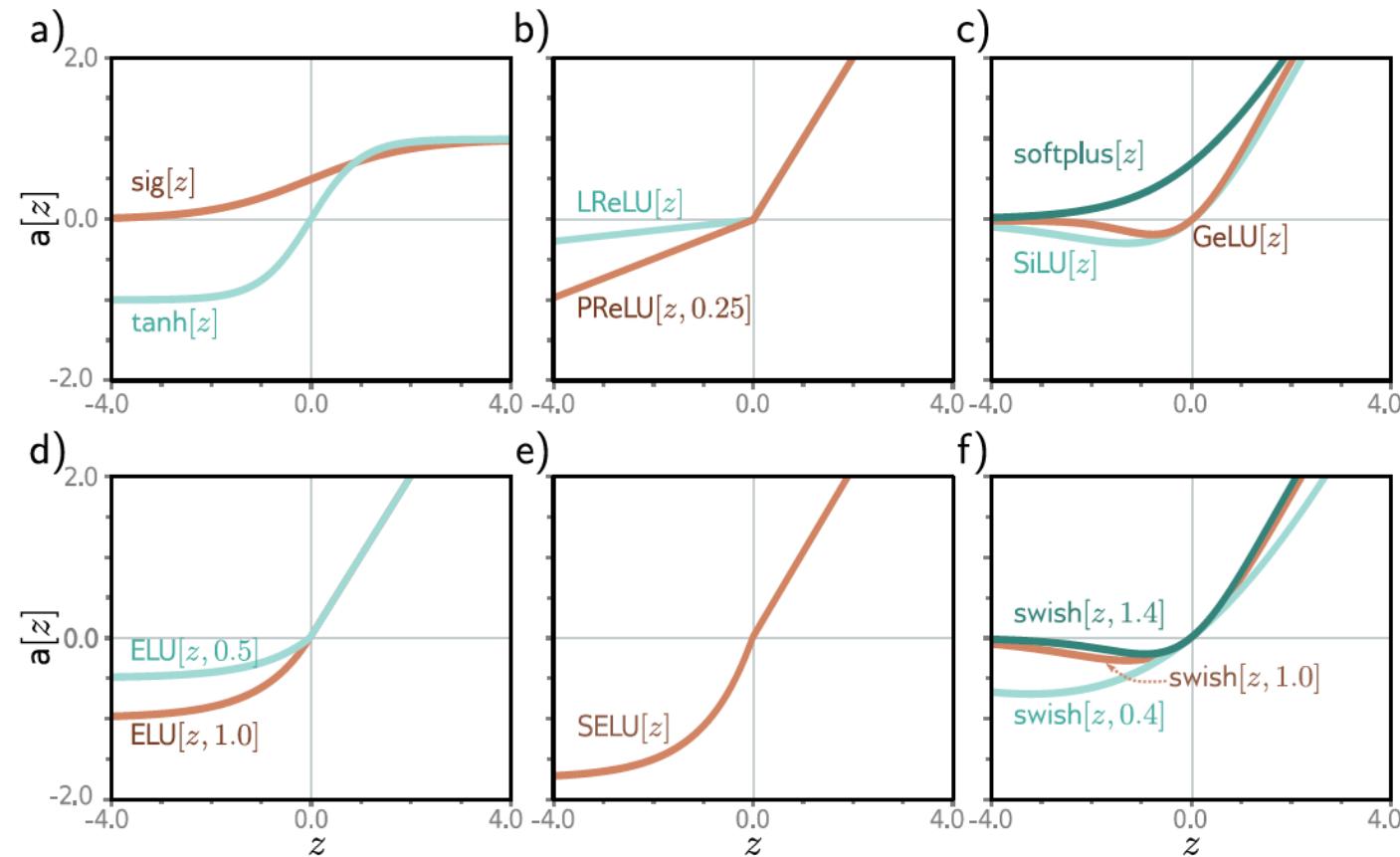
$$\begin{aligned} \mathbf{h}_1 &= \mathbf{a}[\beta_0 + \Omega_0 \mathbf{x}] \\ \mathbf{h}_2 &= \mathbf{a}[\beta_1 + \Omega_1 \mathbf{h}_1] \\ \mathbf{h}_3 &= \mathbf{a}[\beta_2 + \Omega_2 \mathbf{h}_2] \\ &\vdots \\ \mathbf{h}_K &= \mathbf{a}[\beta_{K-1} + \Omega_{K-1} \mathbf{h}_{K-1}] \\ \mathbf{y} &= \beta_K + \Omega_K \mathbf{h}_K. \end{aligned}$$

A deep network with 1 input, 1 output, and K layers of $D > 2$ hidden units
→ can create a function with up to $(D + 1)^K$ linear regions

Depth Efficiency: Functions have been identified that require a shallow network with exp. more hidden units to achieve an equivalent approximation to that of a deep network

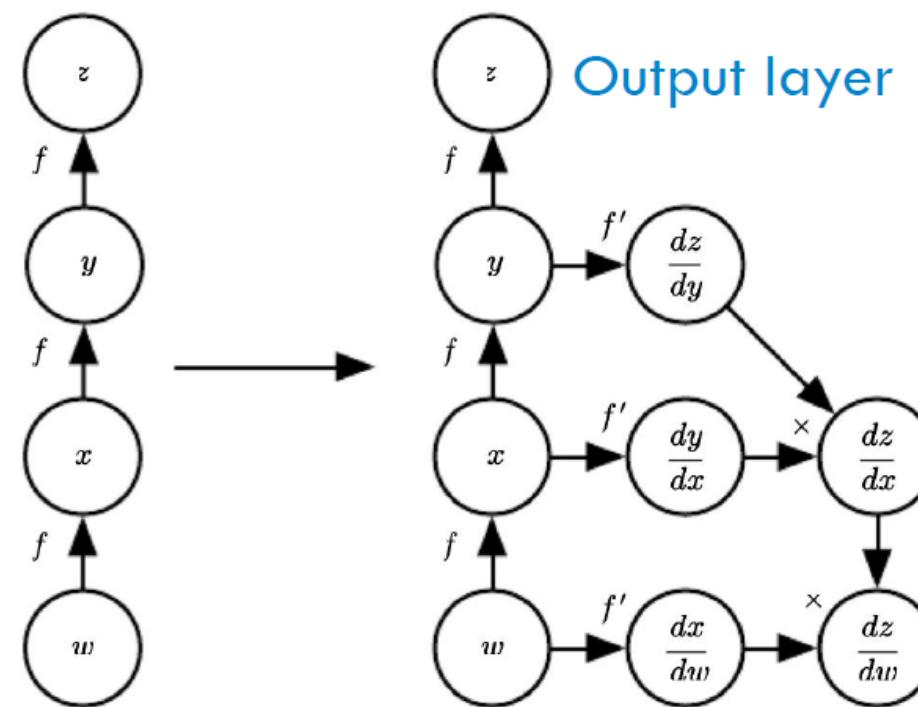
Activation Functions

Leaky ReLU, parameterized ReLU, and continuous functions can be shown to provide minor performance gains over the ReLU in particular situations



Training NN Weights: Backpropagation

Application of **chain rule** for gradient descent optimization of the loss function.
Intuitive explanation: <https://youtu.be/Ilg3gGewQ5U?si=o76IUk2QGBIj6aVU>



Stochastic Gradient Descent (SGD)

Gradient descent to **find the global optimum** of a high-dimensional loss function is *challenging*!

SGD: Dataset is partitioned into *minibatches* for which gradient descent is computed. Batches are drawn without replacement until all data is used. One full pass through the dataset defines an “epoch”.

Advantages:

- Adds randomness/noise but still improves the fit with each update.
- Therefore, may be able to **escape local minima**.
- Reduces the chance of getting stuck at saddle points.
- **Less computationally expensive** per update.
- Often leads to **better generalization** to new data.

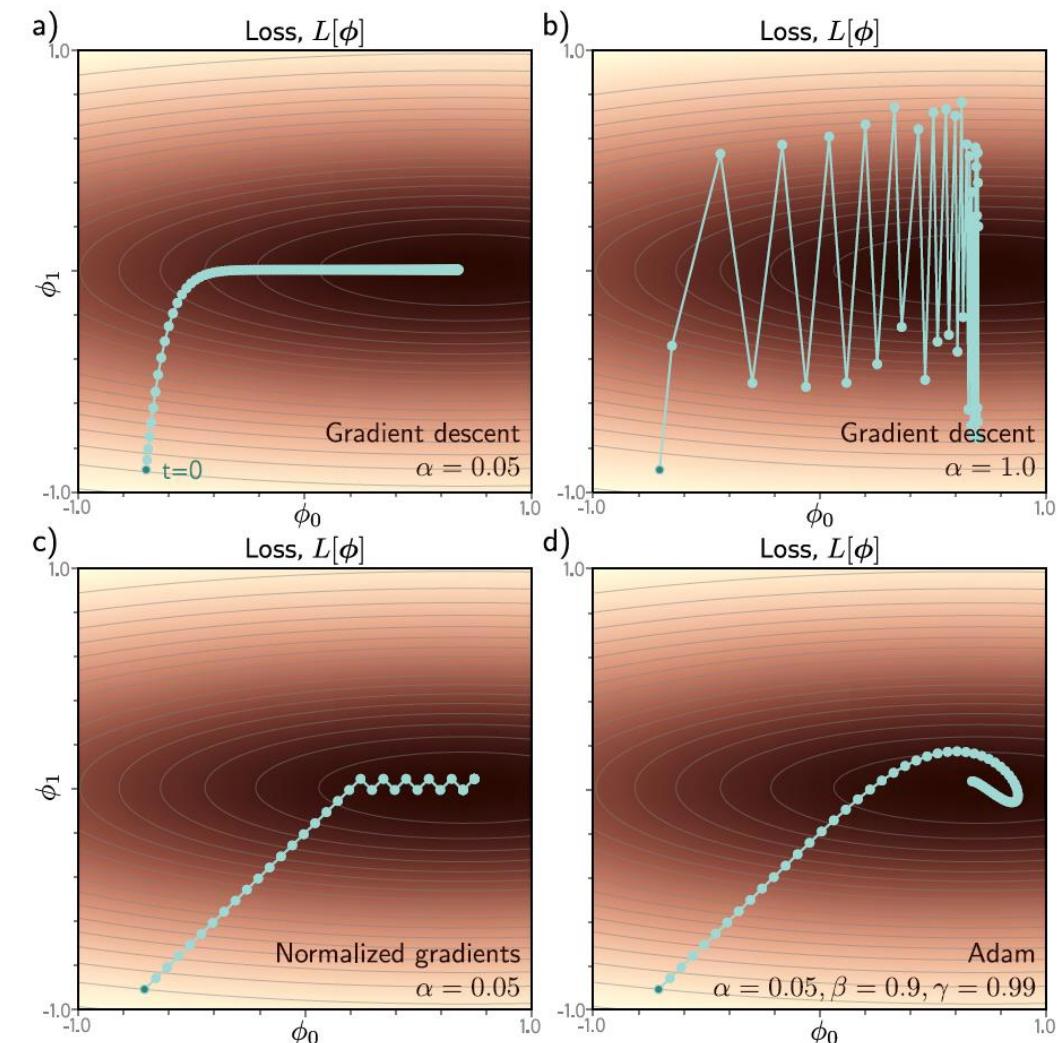
Improvements for SGD

Adding a momentum term:

Parameters updated with a weighted combination of the gradient computed from the current batch and the direction moved in the previous step

Adaptive moment estimation (Adam):

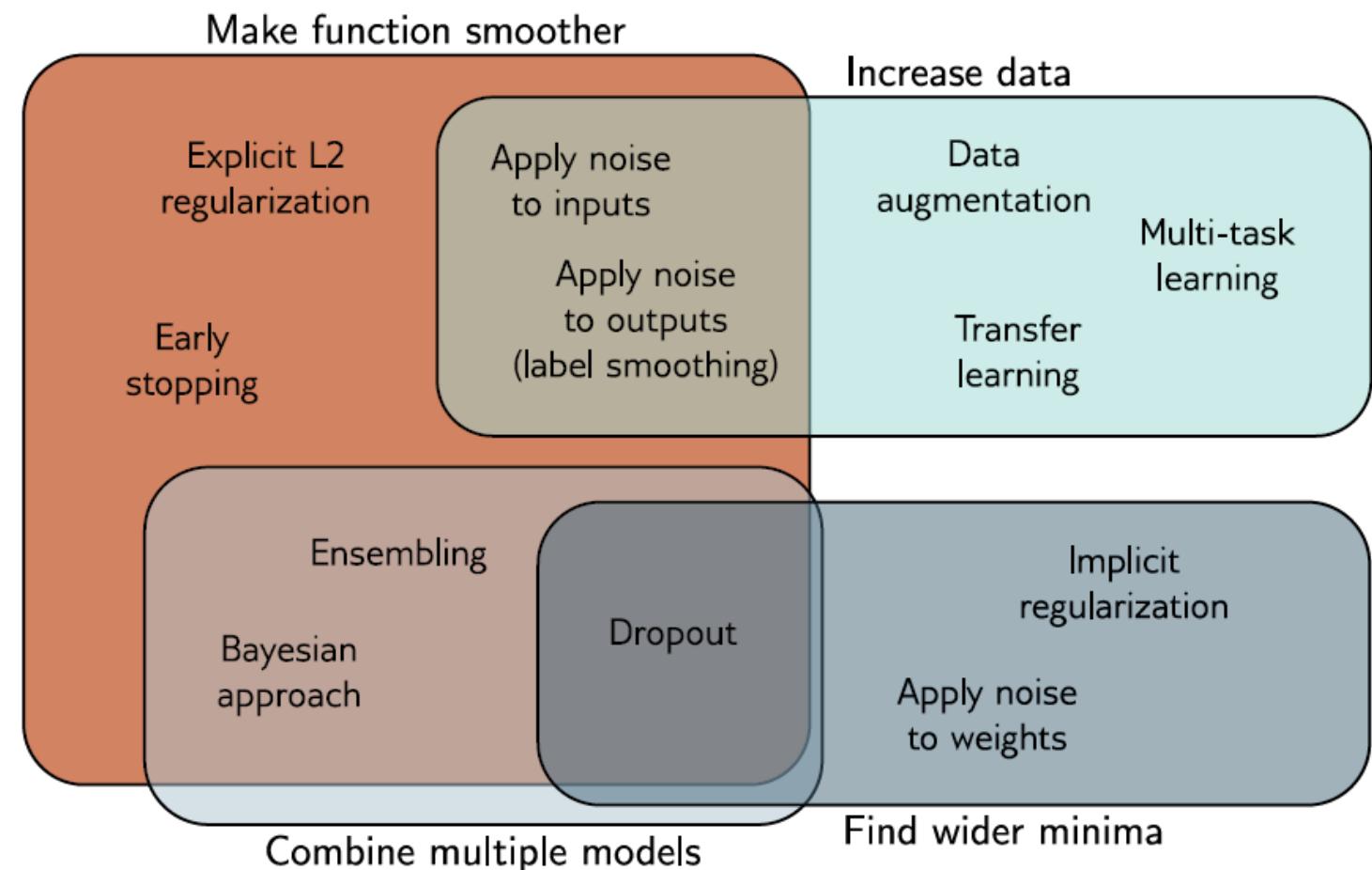
Extends momentum-based optimization by also adapting/normalizing the learning rate using estimates of first and second moments of the gradients.



Regularization of NNs

Same bias/variance trade-off as discussed in earlier lecture

Performance is assessed on test set.



Deep Learning Choices/Hyperparameters

- Number of hidden layers
- Number of hidden neurons for each hidden layer
- Activation function
- Loss function
- Type of SGD (usually Adam)
- Batch size
- Number of epochs
- Learning rate and/or learning rate schedule
- Regularization methods:
Dropout rate, “*patience*” for early stopping, $l2$ strength,...

Interactive Demo of neural network (can tweak hyperparameters):
<https://playground.tensorflow.org>

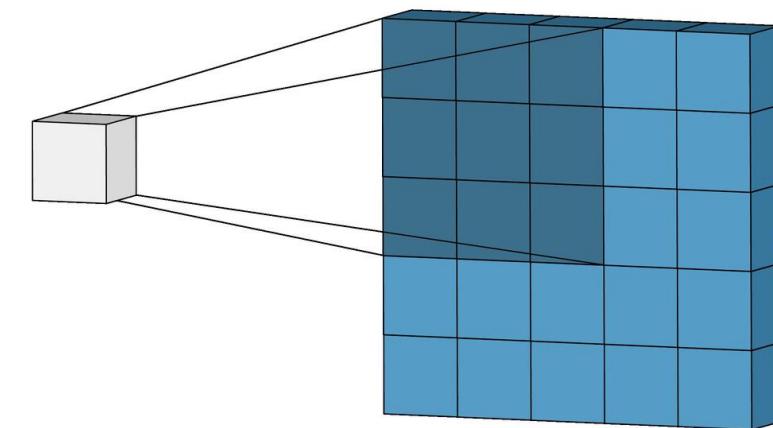
Other Types of Neural Network Layers

Convolution:

Commonly used for array-type inputs (like pixels of image) to extract **hierarchical features**

- Kernel (or “filter”) is a small matrix that slides/convolves across the input
- Filter is element-wise multiplied with the corresponding portion
- Element-wise products summed to produce a single value

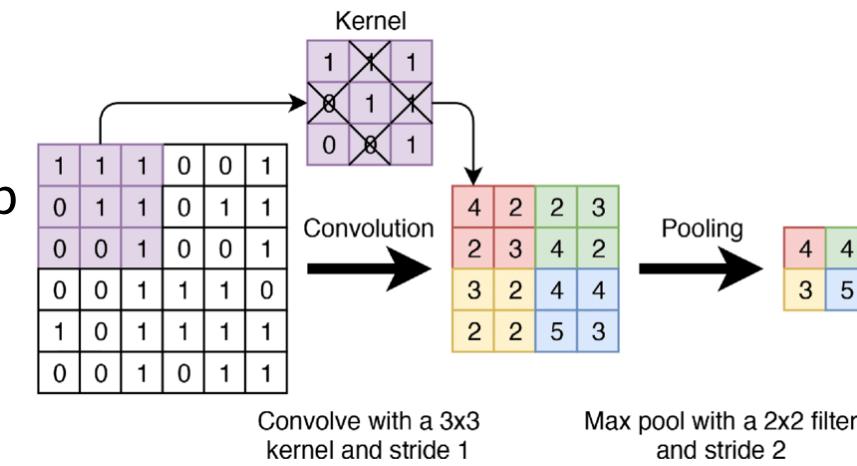
<https://medium.com/@nikitamalviya/convolution-pooling-f8e797898cf9>
<https://doi.org/10.5334/jcaa.32>



Max Pooling:

Downsampling operation to reduce the spatial dimensions of the feature maps

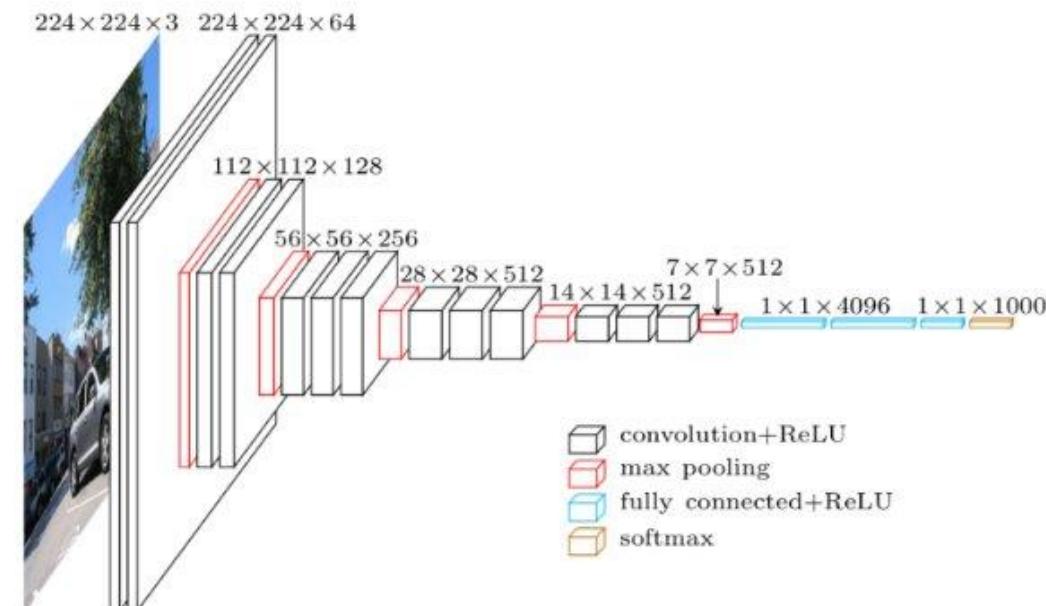
- Small window (e.g., 2x2 or 3x3) slides over the feature map
- Within the window, the max pooling layer selects the **maximum value** (i.e., *most important*).



State-of-the-Art Convolutional Neural Network

VGG16 Computer Vision Model

Combines many different layer types with $10^3\text{-}10^{12}$ parameters

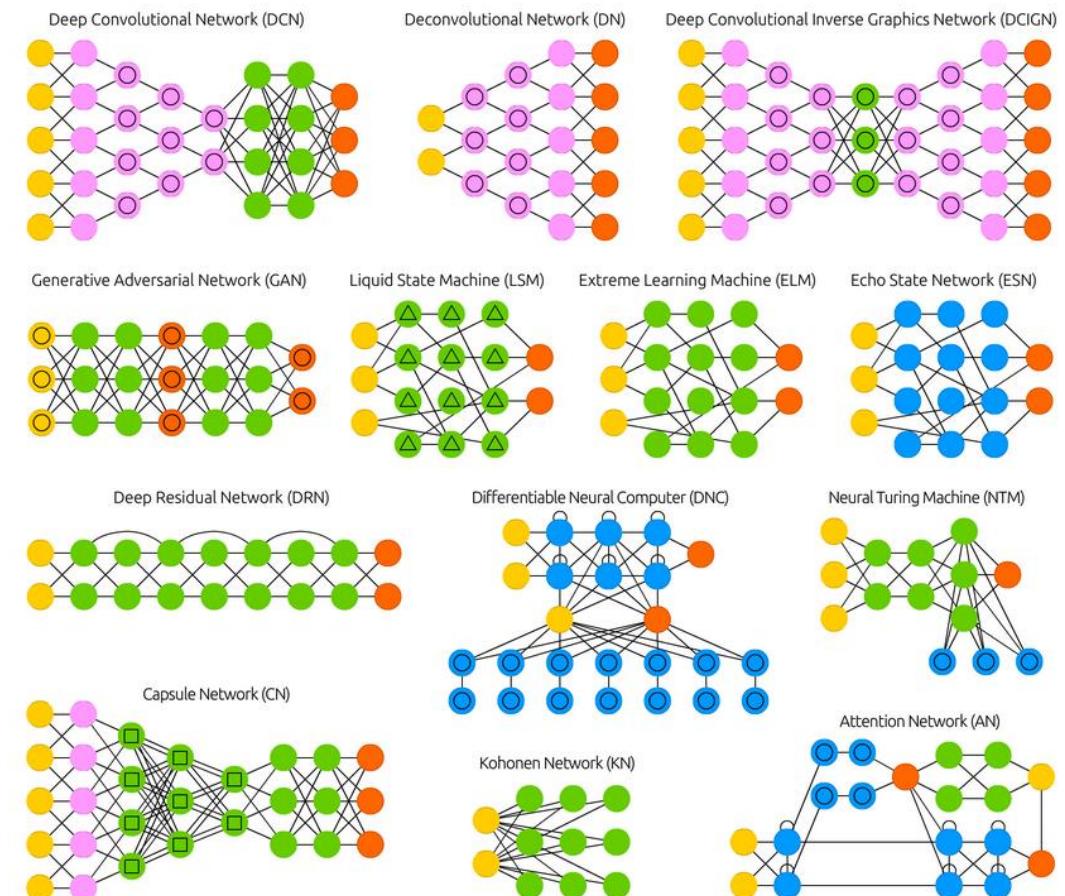
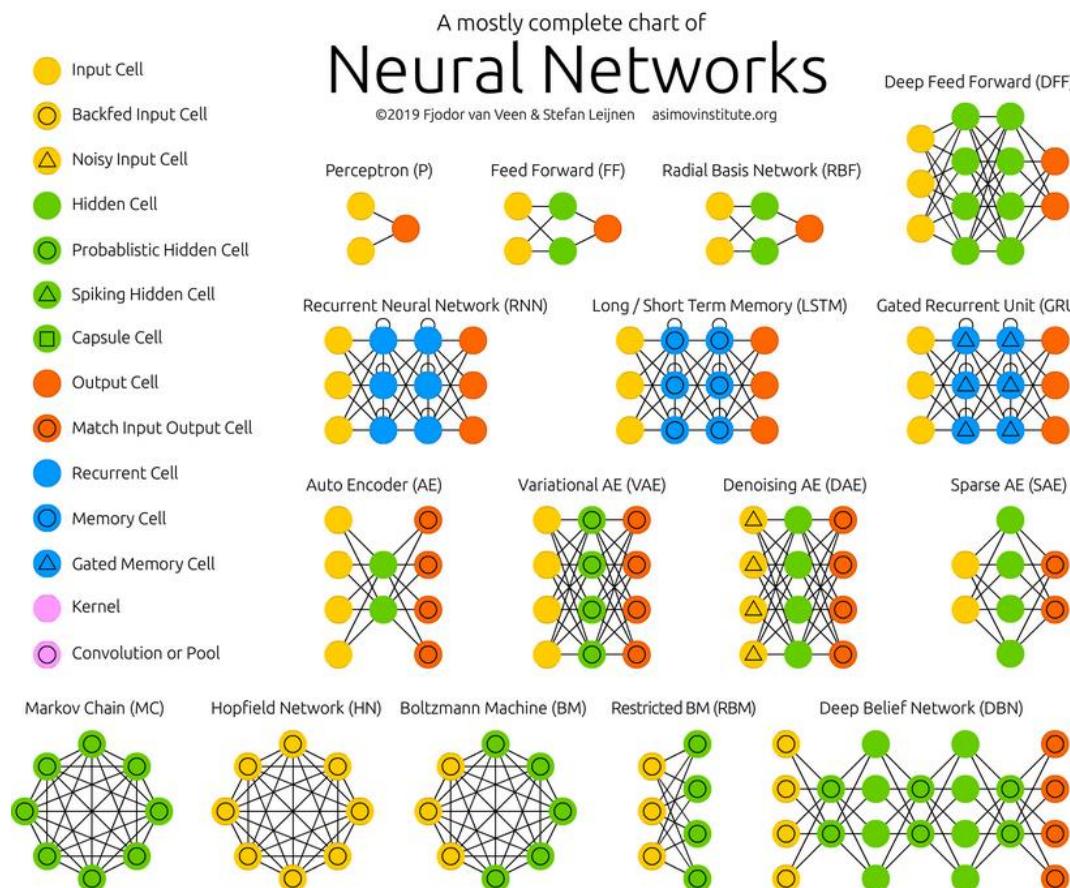


Interactive demos:

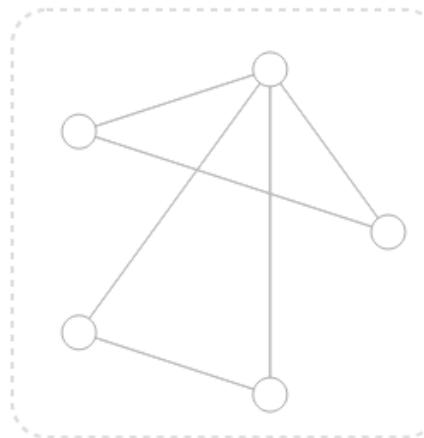
https://adamharley.com/nv_vis/cnn/2d.html

<https://poloclub.github.io/cnn-explainer/>

Other Deep Learning Architectures



MatSci & Chemistry: Graph Neural Networks (GNNs)



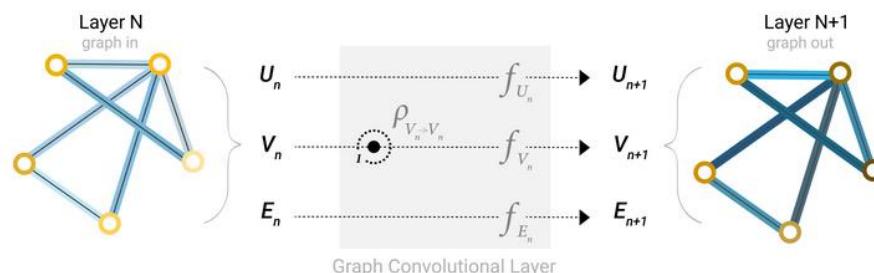
- V** Vertex (or node) attributes
 - e.g., node identity, number of neighbors
- E** Edge (or link) attributes and directions
 - e.g., edge identity, edge weight
- U** Global (or master node) attributes
 - e.g., number of nodes, longest path

Typically elemental embedding

Typically interatomic distance

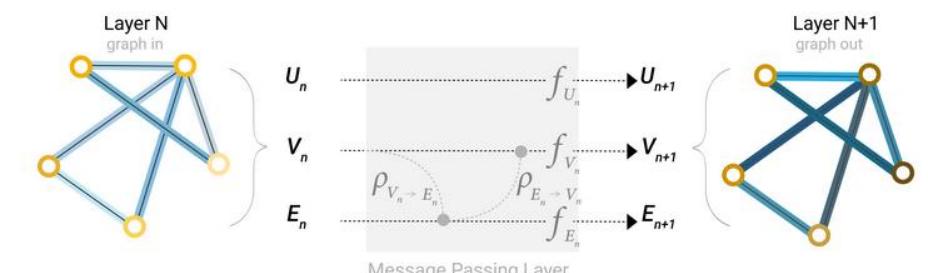
e.g., unit cell parameters

Graph convolution (GCNN)



update function $f = \dots$, ...
pooling function ρ

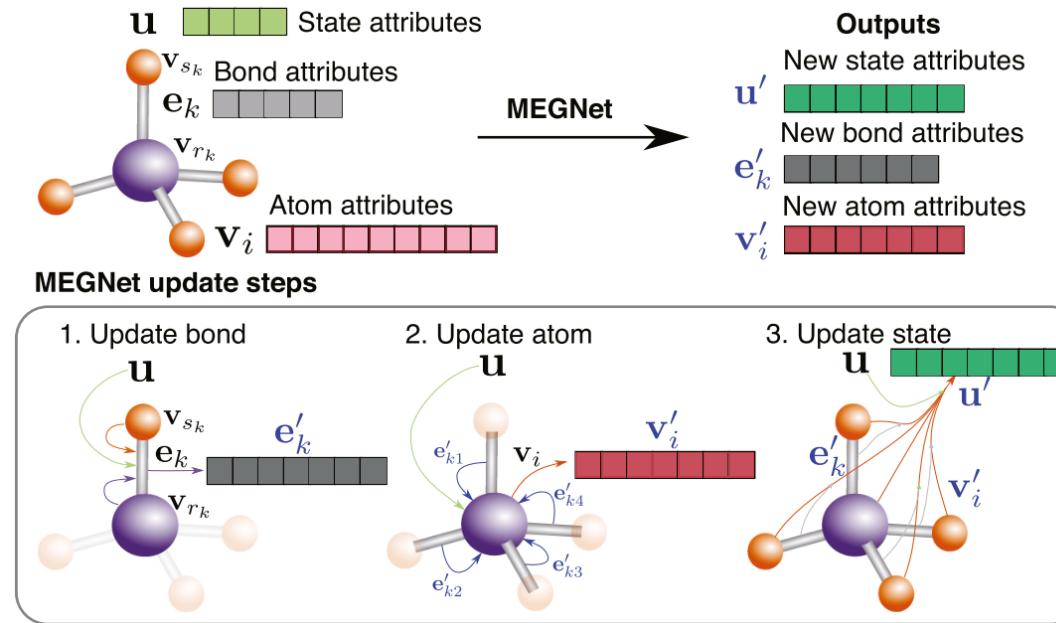
Message Passing (MPGNN)



update function $f = \dots$, ...
pooling function ρ

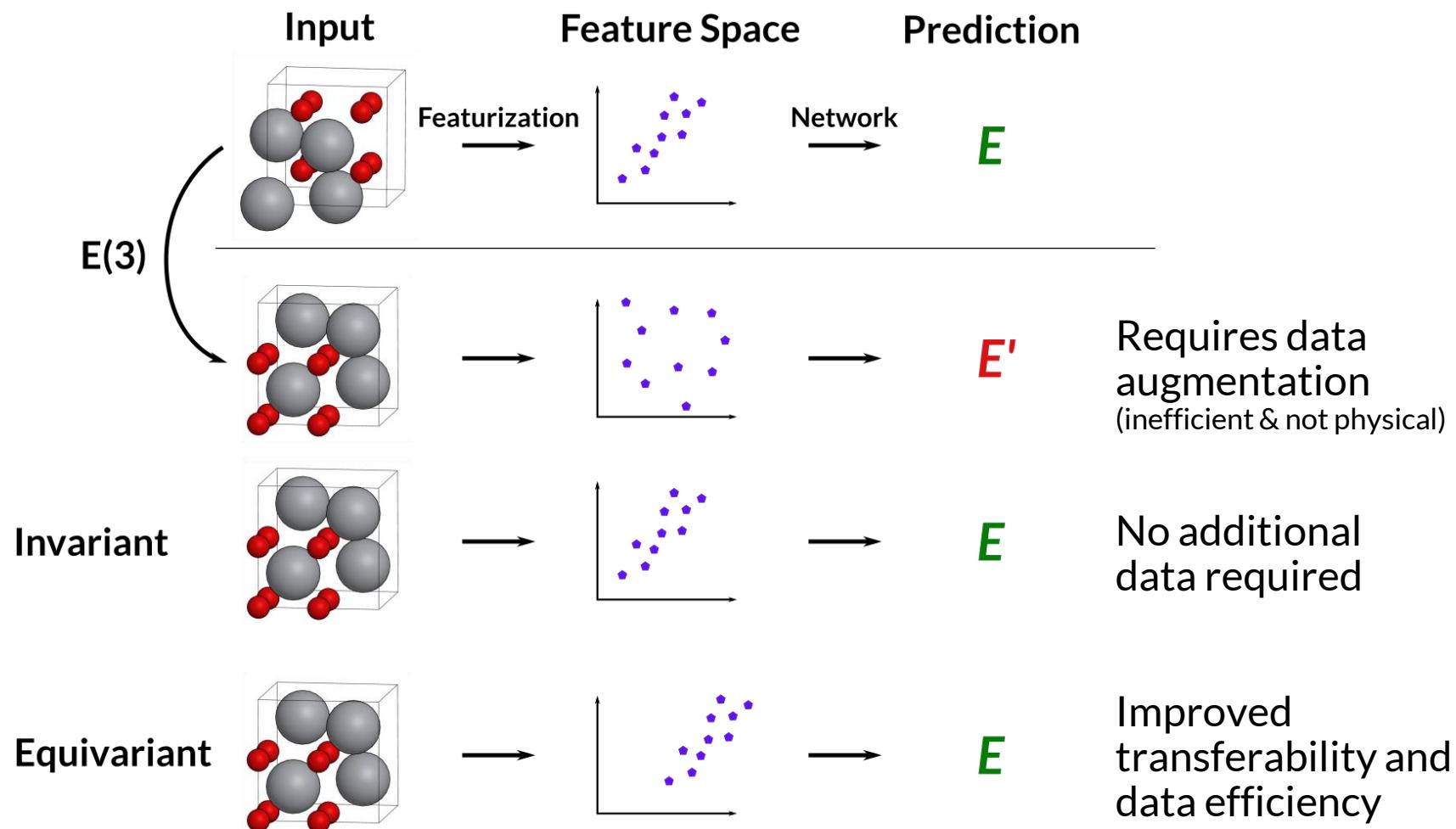
Interactive demos: <https://distill.pub/2021/gnn-intro/>

MatSci & Chemistry: Graph Neural Networks (GNNs)

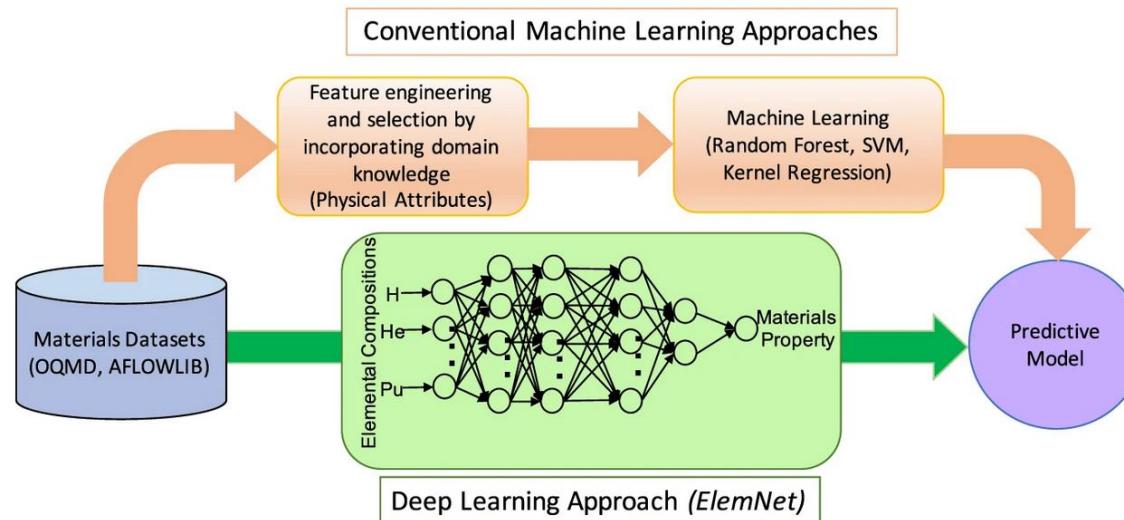


CGCNN, ALIGNN, MEGNet, SchNet, PointNet, ...

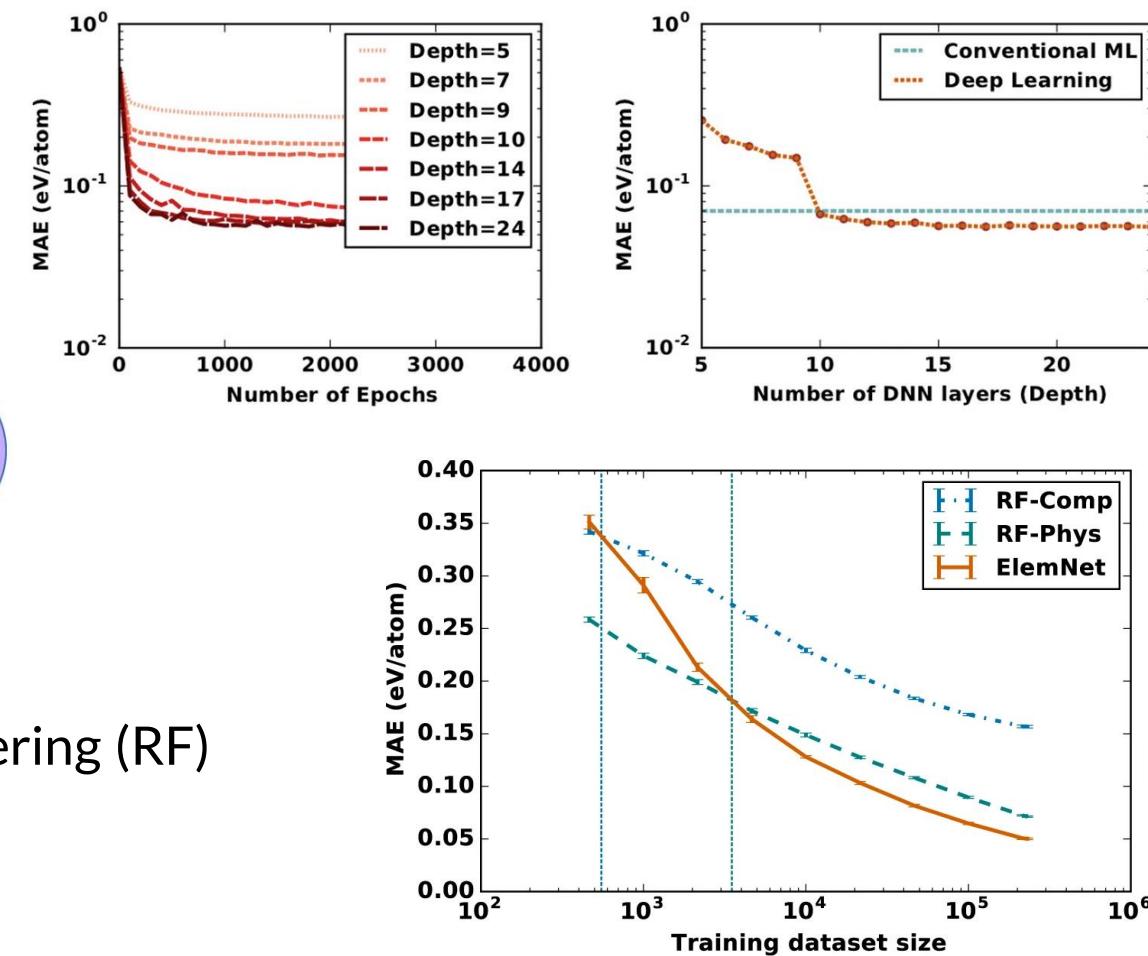
MatSci & Chemistry: E(3) Equivariant GNNs



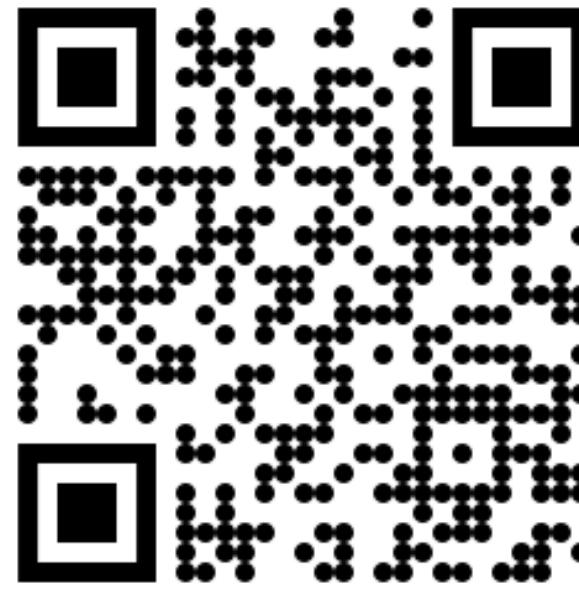
MatSci, Composition-only: ElemNet



- Slightly outperforms manual feature engineering (RF)
- Requires several 1000 datapoints
- More recent, state-of-the-art model:
CrabNet (with self-attention)



Lecture Feedback



Please, scan the QR code and take a minute to let me know how the lecture was and mention any **feedback/questions**

This form is **anonymous!**