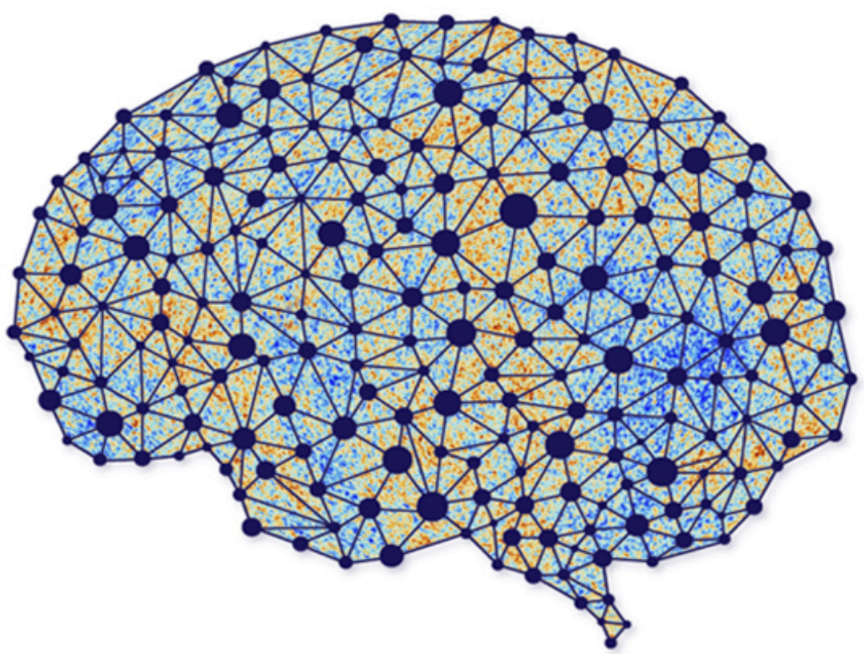
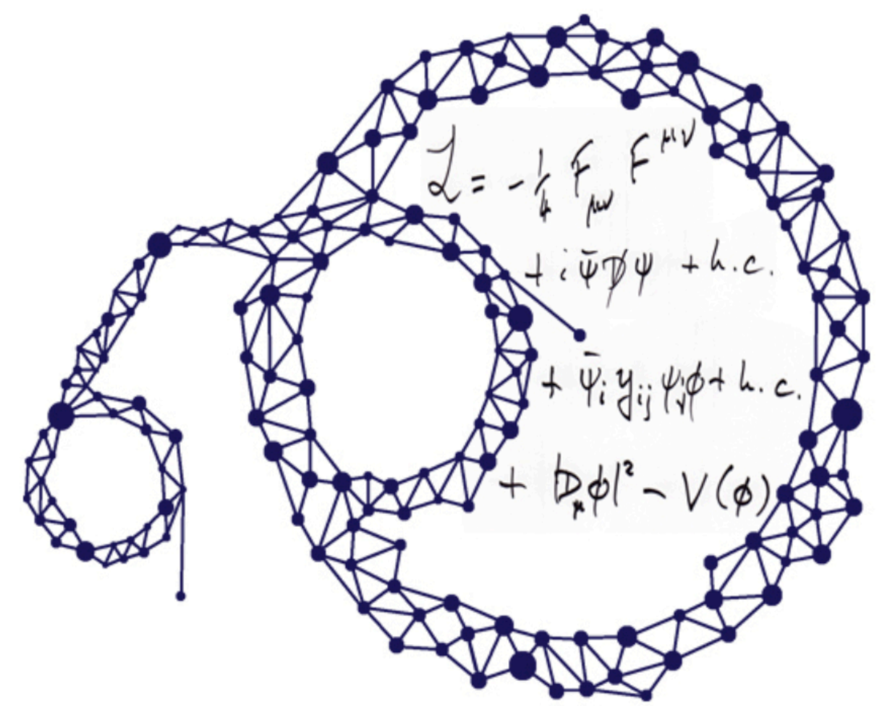
PHY 835: Machine Learning in Physics

Lecture 9: Deep Neural Network

February 20, 2024



AI
∩
Universe



Gary Shiu

Why going deep?

- A shallow NN with only a single hidden layer can already approximate any continuous function to a specific precision, using piecewise linear functions.
- However, the network capacity (# hidden units) may be impractically large. A deep NN can produce more linear regions for a given # parameters.

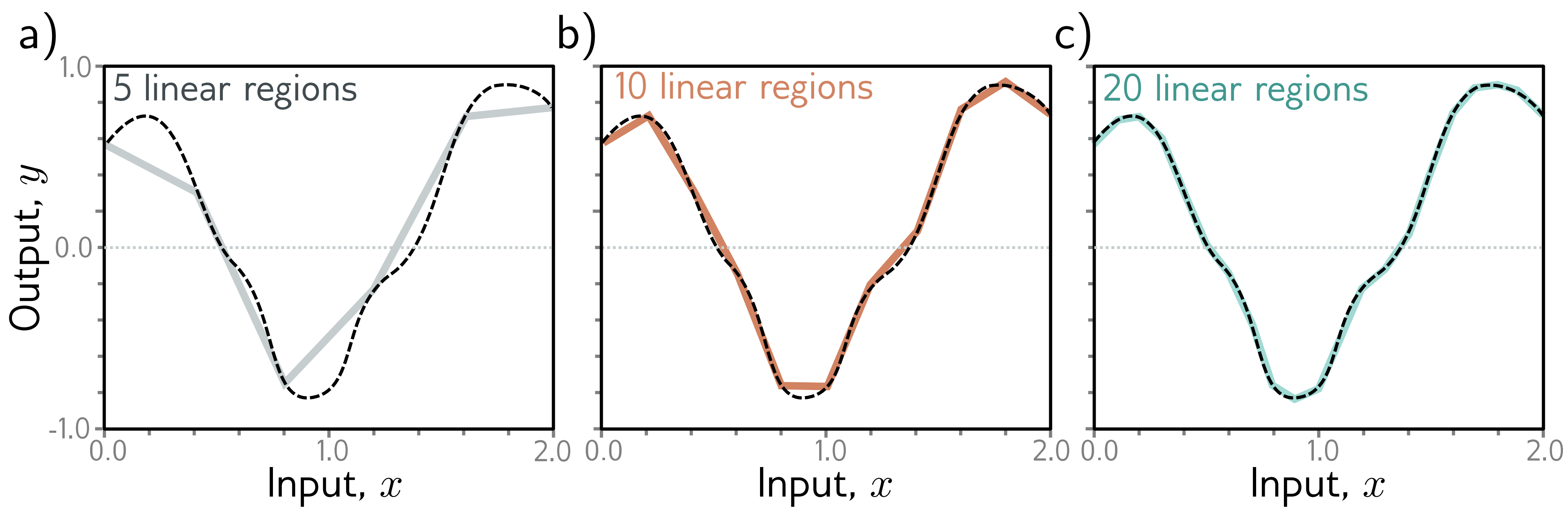
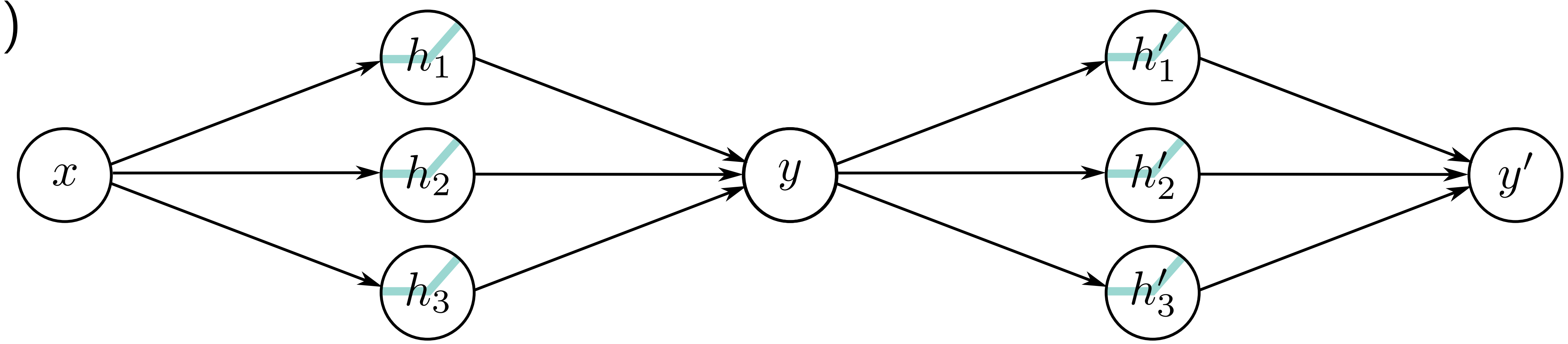


Figure from Simon Prince “Understanding Deep Learning"

Composing Neural Networks

a)

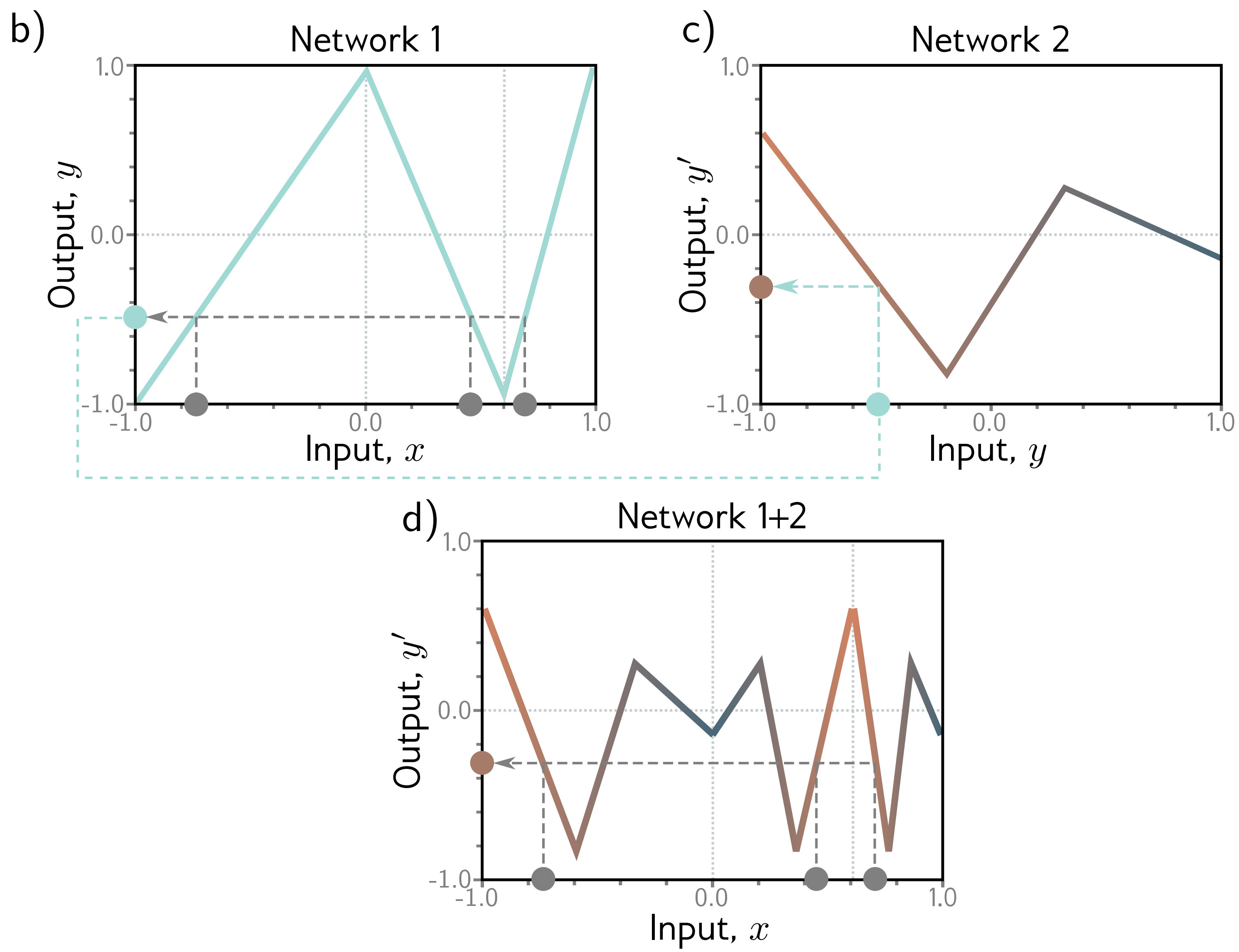


$$
\begin{aligned}
h_1 &= \mathrm{a}[\theta_{10} + \theta_{11} x] \\
h_2 &= \mathrm{a}[\theta_{20} + \theta_{21} x] \\
h_3 &= \mathrm{a}[\theta_{30} + \theta_{31} x],
\end{aligned}
\qquad y = \phi_0 + \phi_1 h_1 + \phi_2 h_2 + \phi_3 h_3.
$$

$$
\begin{aligned}
h'_1 &= \mathrm{a}[\theta'_{10} + \theta'_{11} y] \\
h'_2 &= \mathrm{a}[\theta'_{20} + \theta'_{21} y] \\
h'_3 &= \mathrm{a}[\theta'_{30} + \theta'_{31} y],
\end{aligned}
\qquad y' = \phi'_0 + \phi'_1 h'_1 + \phi'_2 h'_2 + \phi'_3 h'_3.
$$

- As we will see, compositions of NN are **special cases** of Deep NNs which are even more expressive.
- Generate piecewise linear functions. However, # linear regions is more than a single layer with 6 hidden units. To understand why, see next page.

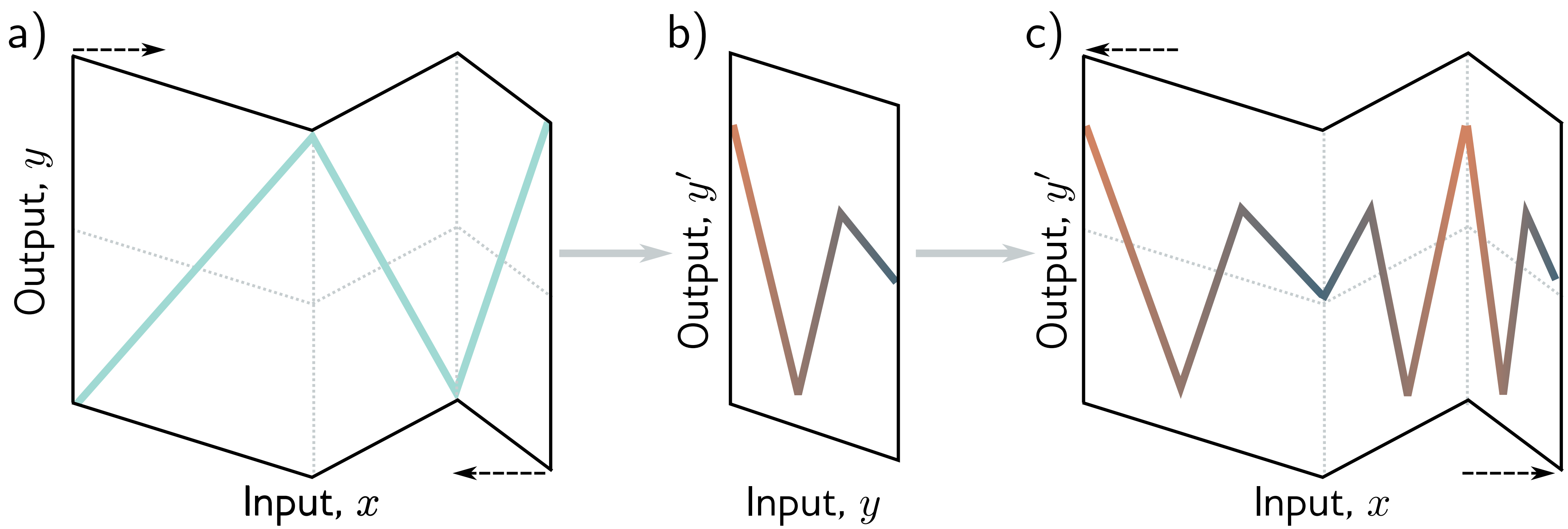
Mapping multiple inputs to the same output



Three different ranges of x are mapped to the same output range $y \in [-1,1]$ and the subsequent mapping from this range of y to y' is applied three times.

The composition creates **nine linear regions**.

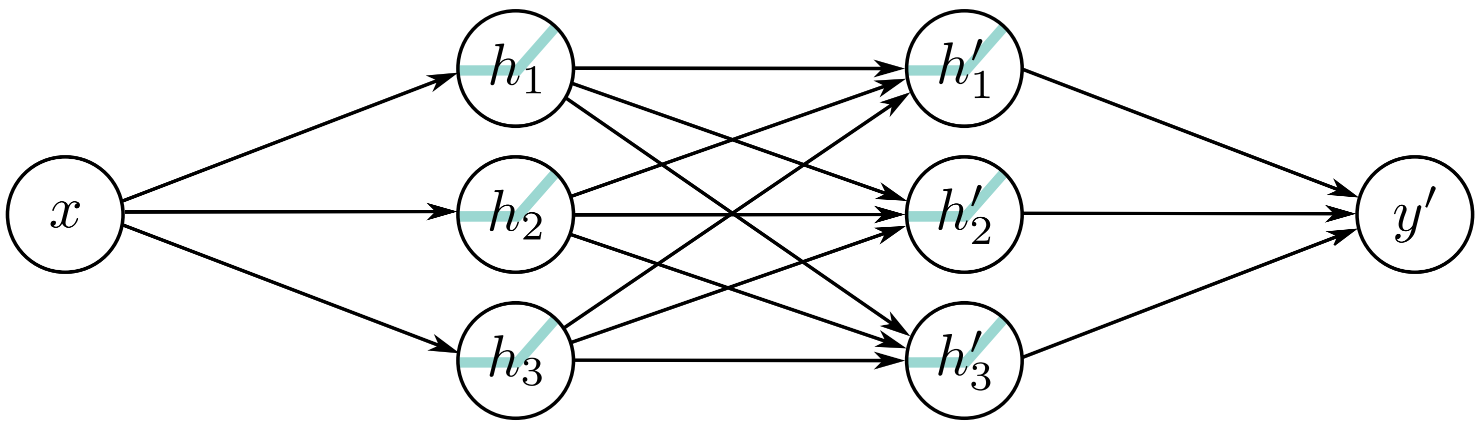
Folding Input Space



The first network "folds" the input space x back onto itself so that multiple inputs generate the same output. Then the second network applies a function, which is replicated at all points that were folded on top of one another.

Deep Neural Networks

- The composition of 2 shallow networks results in a 2-layer network:



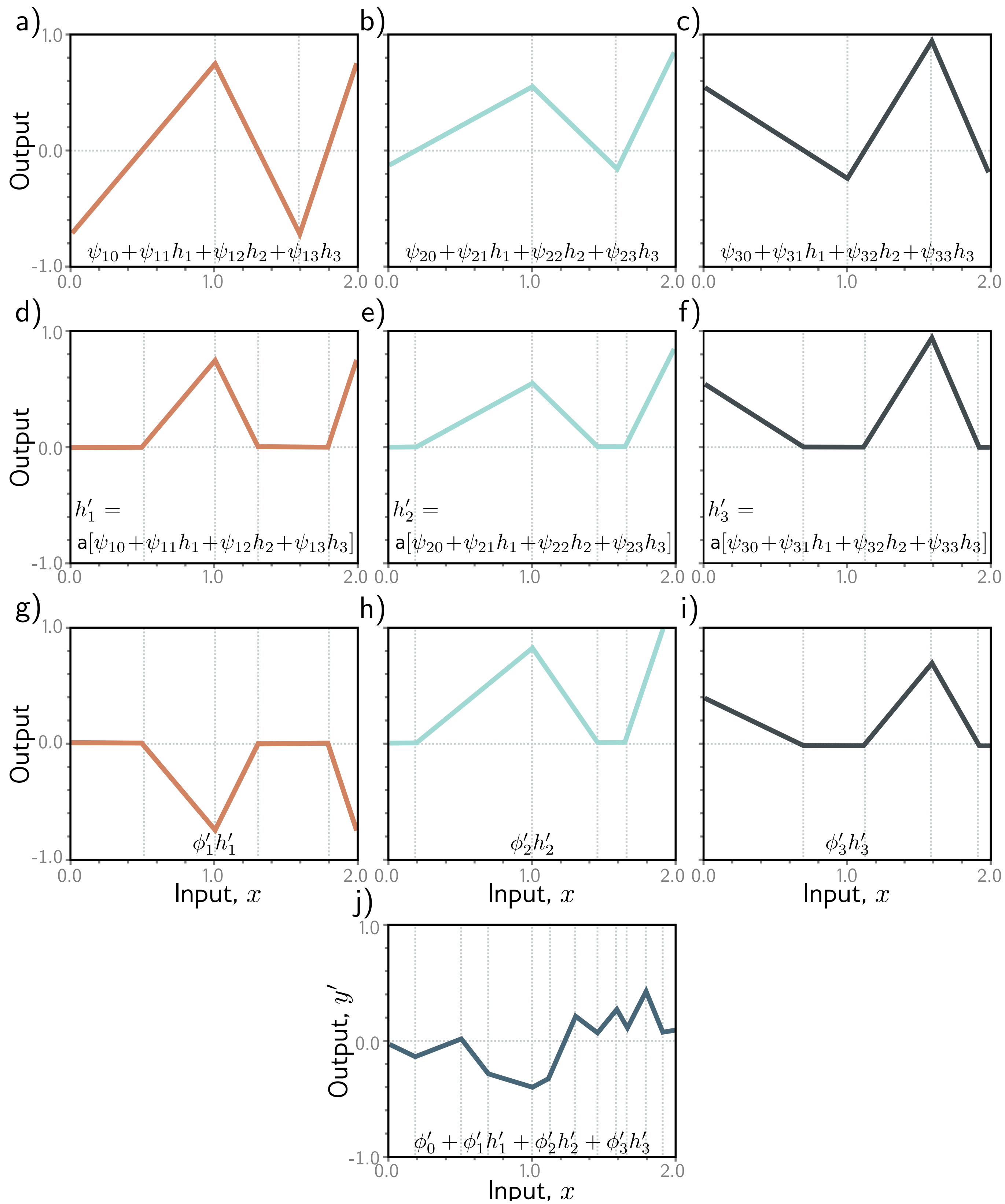
- This is because we can eliminate the **"mediator"** y:

$$
\begin{aligned}
h'_1 &= \mathrm{a}[\theta'_{10} + \theta'_{11} y] &= \mathrm{a}[\theta'_{10} + \theta'_{11}\phi_0 + \theta'_{11}\phi_1 h_1 + \theta'_{11}\phi_2 h_2 + \theta'_{11}\phi_3 h_3] \\
h'_2 &= \mathrm{a}[\theta'_{20} + \theta'_{21} y] &= \mathrm{a}[\theta'_{20} + \theta'_{21}\phi_0 + \theta'_{21}\phi_1 h_1 + \theta'_{21}\phi_2 h_2 + \theta'_{21}\phi_3 h_3] \\
h'_3 &= \mathrm{a}[\theta'_{30} + \theta'_{31} y] &= \mathrm{a}[\theta'_{30} + \theta'_{31}\phi_0 + \theta'_{31}\phi_1 h_1 + \theta'_{31}\phi_2 h_2 + \theta'_{31}\phi_3 h_3],
\end{aligned}
$$

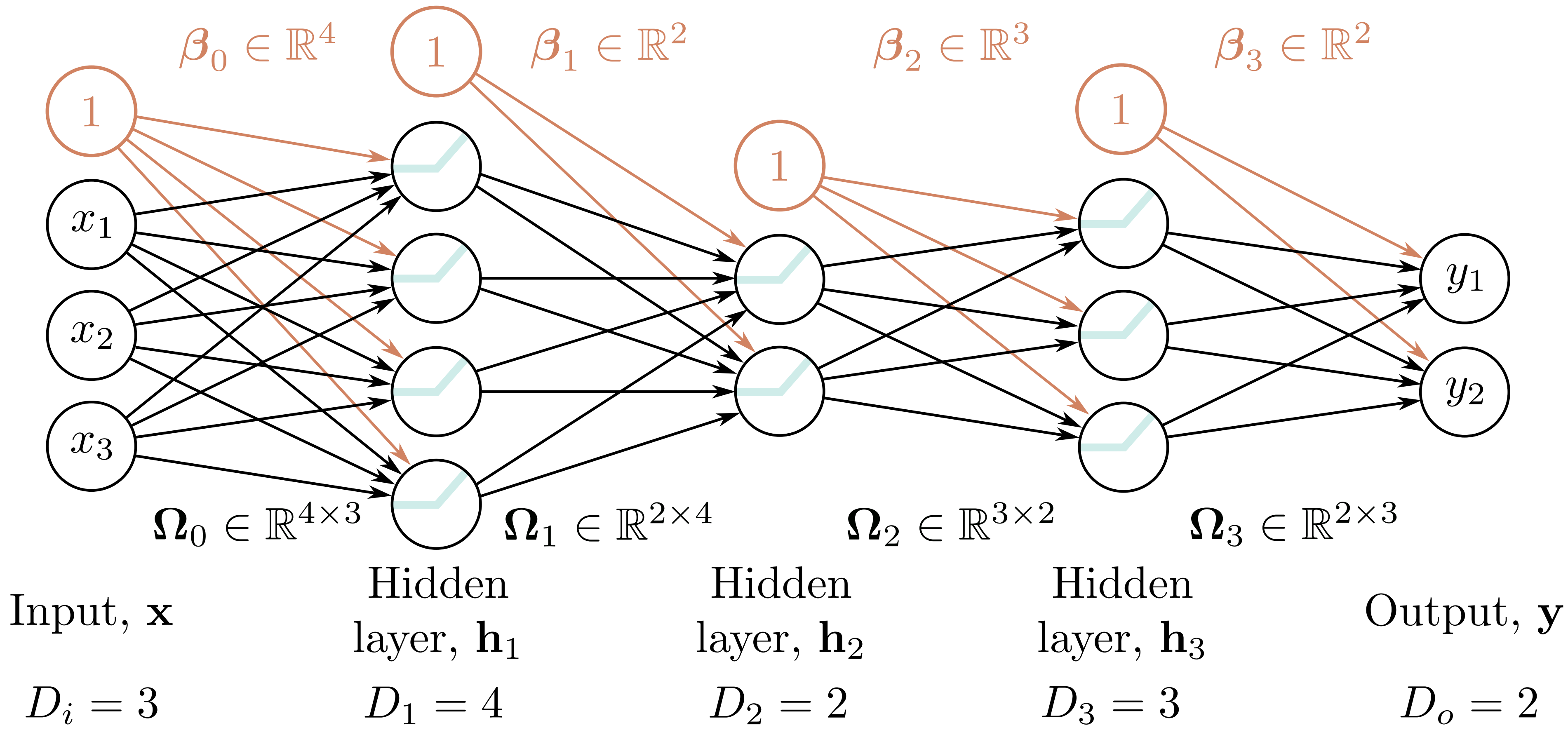
- However, a 2-layer network is more general since there are 9 unconstrained slope parameters instead of 6:

$$
\begin{aligned}
h'_1 &= \mathrm{a}[\psi_{10} + \psi_{11} h_1 + \psi_{12} h_2 + \psi_{13} h_3] \\
h'_2 &= \mathrm{a}[\psi_{20} + \psi_{21} h_1 + \psi_{22} h_2 + \psi_{23} h_3] \\
h'_3 &= \mathrm{a}[\psi_{30} + \psi_{31} h_1 + \psi_{32} h_2 + \psi_{33} h_3],
\end{aligned}
$$

Deep Neural Networks



Hyperparameters



- Modern deep NNs might have $\gtrsim \mathcal{O}(10^2)$ layers with $\mathcal{O}(10^3)$ of hidden units in each layer.
- The number of layers K = **depth**, & the number of hidden units in each layer (=**width**) $D_1, D_2, \ldots, D_K$ are hyperparameters. The network **capacity** = # number of hidden units.
- For fixed hyperparameters, the model describes a family of functions, and the parameters θ (known as weights) determine a specific function.

General Formulation

- We can express the above 2-layer network in **matrix notation**:

$$\begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix} = \mathbf{a}\left[\begin{bmatrix} \theta_{10} \\ \theta_{20} \\ \theta_{30} \end{bmatrix} + \begin{bmatrix} \theta_{11} \\ \theta_{21} \\ \theta_{31} \end{bmatrix} x\right],$$

$$\begin{bmatrix} h'_1 \\ h'_2 \\ h'_3 \end{bmatrix} = \mathbf{a}\left[\begin{bmatrix} \psi_{10} \\ \psi_{20} \\ \psi_{30} \end{bmatrix} + \begin{bmatrix} \psi_{11} & \psi_{12} & \psi_{13} \\ \psi_{21} & \psi_{22} & \psi_{23} \\ \psi_{31} & \psi_{32} & \psi_{33} \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix}\right],$$

and

$$y' = \phi'_0 + \begin{bmatrix} \phi'_1 & \phi'_2 & \phi'_3 \end{bmatrix} \begin{bmatrix} h'_1 \\ h'_2 \\ h'_3 \end{bmatrix},$$

$$\begin{aligned} \mathbf{h} &= \mathbf{a}\left[\boldsymbol{\theta}_0 + \boldsymbol{\theta} x\right] \\ \mathbf{h}' &= \mathbf{a}\left[\boldsymbol{\psi}_0 + \boldsymbol{\Psi}\mathbf{h}\right] \\ y' &= \phi'_0 + \boldsymbol{\phi}'\mathbf{h}', \end{aligned}$$

- More generally, a K-layer network:

$$\begin{aligned} \mathbf{h}_1 &= \mathbf{a}[\boldsymbol{\beta}_0 + \boldsymbol{\Omega}_0 \mathbf{x}] \\ \mathbf{h}_2 &= \mathbf{a}[\boldsymbol{\beta}_1 + \boldsymbol{\Omega}_1 \mathbf{h}_1] \\ \mathbf{h}_3 &= \mathbf{a}[\boldsymbol{\beta}_2 + \boldsymbol{\Omega}_2 \mathbf{h}_2] \\ &\vdots \\ \mathbf{h}_K &= \mathbf{a}[\boldsymbol{\beta}_{K-1} + \boldsymbol{\Omega}_{K-1} \mathbf{h}_{K-1}] \\ \mathbf{y} &= \boldsymbol{\beta}_K + \boldsymbol{\Omega}_K \mathbf{h}_K. \end{aligned}$$

Hyperparameters:

$$K, D_1, D_2, \ldots, D_K$$

Parameters: biases and weights

$$\boldsymbol{\phi} = \{\boldsymbol{\beta}_k, \boldsymbol{\Omega}_k\}_{k=0}^{K}.$$

What are the sizes of the biases & weights in terms of the hyperparameters?

Shallow vs Deep

- **Universal approximation theorem:** deep NNs can approximate any continuous function arbitrarily closely given sufficient capacity.
  - We can reproduce a shallow network if all but one layer is the identity function. Since we showed that a shallow NN can approximate any continuous function, deep NNs also work.
- **More expressive** (more linear regions per parameter):
  - A shallow NN with 1 input, 1 output, $D > 2$ hidden units can create up to $D + 1$ linear regions using $3D + 1$ parameters.
  - A deep NN with 1 input, 1 output, $D > 2$ hidden units can create up to $(D + 1)^K$ linear regions using $3D + 1 + (K - 1)D(D + 1)$ parameters (more next page).

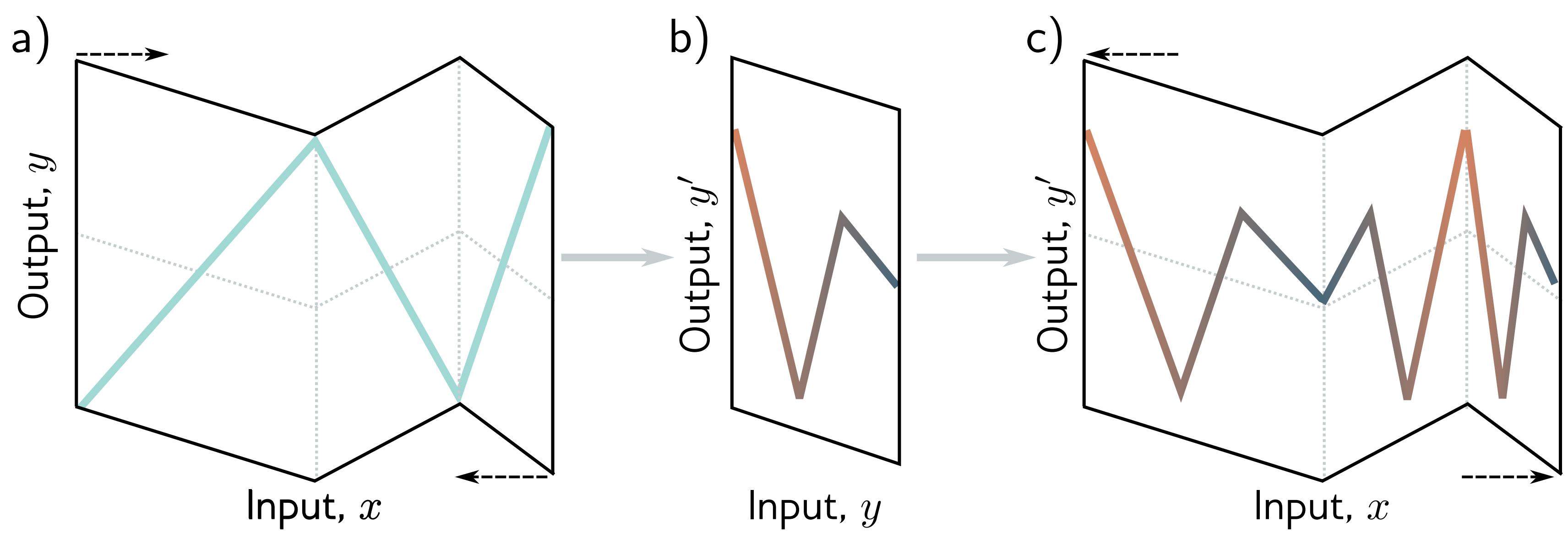
This exponential growth in linear regions is what makes deep NN more expressive.

Shallow vs Deep

- The counting of parameters for shallow NNs goes as follows:
  - There are D hidden units, each has two parameters (bias, weight). The output layer has D weights and one bias. # parameter=2D+D+1.
- The counting of parameters for deep NNs goes as follows:
  - There are D weights between the input and the first hidden layer, $K-1$ lots of $D \times D$ inputs between adjacent hidden layers, and D weights between the last hidden layer and the output. There are D biases at each of the K hidden layers and 1 bias for the output. This gives $D+(K-1)D^2+D+KD+1 = 3D+(K-1)D^2+(K-1)D+1$ parameters.

Shallow vs Deep

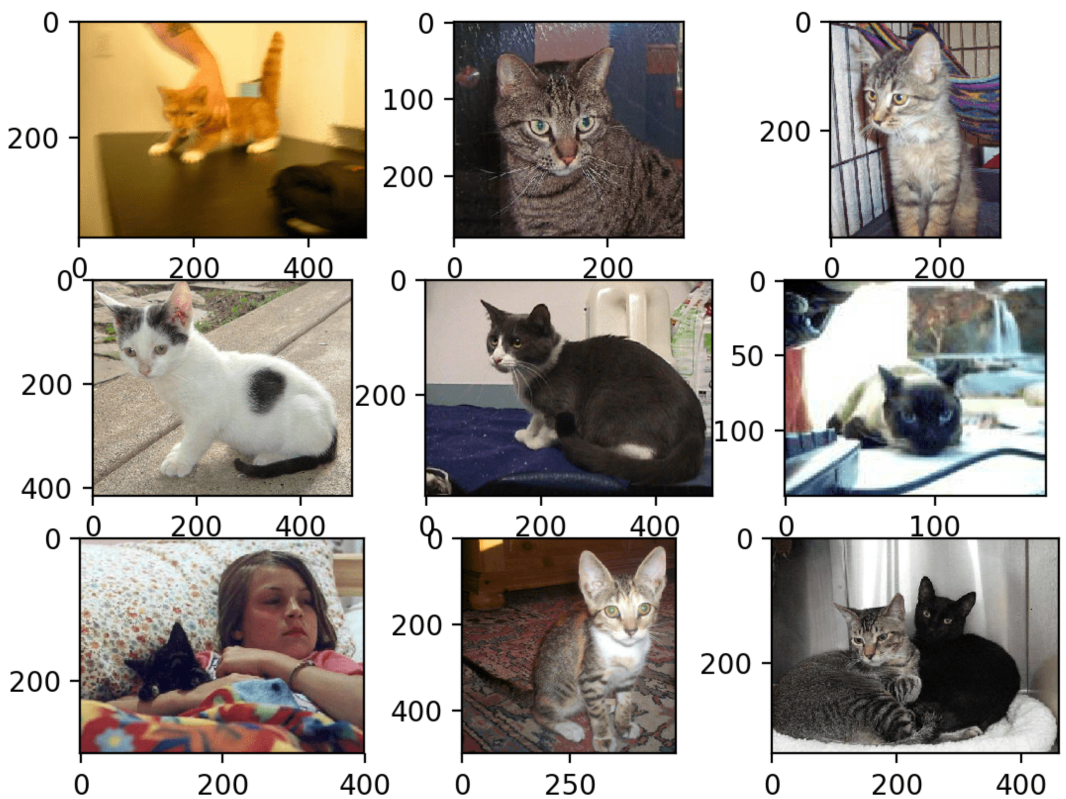
- Deep NNs create much more linear regions for a fixed parameter budget, but they contain **complex dependence and symmetries**.



- The greater number of regions is an advantage if:
  1. there are similar symmetries in the function to approximate;
  2. the input→output map is a composition of simpler functions.
- **Depth efficiency** refers to the phenomenon that a shallow NN needs exponentially more hidden units to achieve an equivalent approximation to that of a deep NN.

Shallow vs Deep

- **Large, structured inputs:** We have discussed fully connected networks where every element of each layer contributes to every element of the subsequent one.

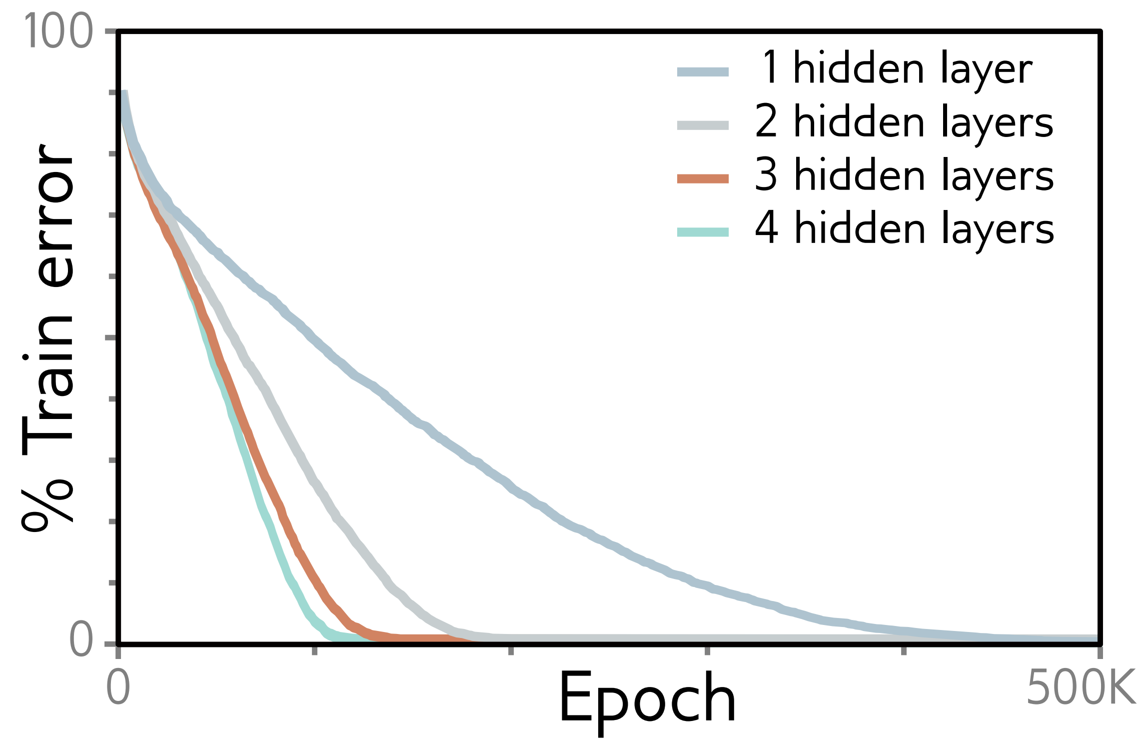


- not practical for large structured inputs like images $\sim 10^6$ pixels
- no point in independently learning to recognize the same object at every position in the image.

- CNN (which we will discuss later): process local image regions in parallel and integrate information from increasing large regions. Difficult to do this local-to-global processing with a single layer.

Shallow vs Deep

- **Training and generalization:** It is easier to train moderately deep networks than to train shallow ones.



- Deep NNs also seem to generalize to new data better than shallow ones.

- Empirically, one finds best results for most tasks using networks with tens to hundreds of layers.