Deep Learning for Data Science
DS 542

https://dl4ds.github.io/fa2025/

Deep Neural Networks

Original slides by Tom Gardos,
other content from Understanding Deep Learning unless otherwise cited

Deep neural networks

- Composing two networks
- Combining the two networks into one
- Hyperparameters
- Notation change and general case
- Shallow vs. deep networks

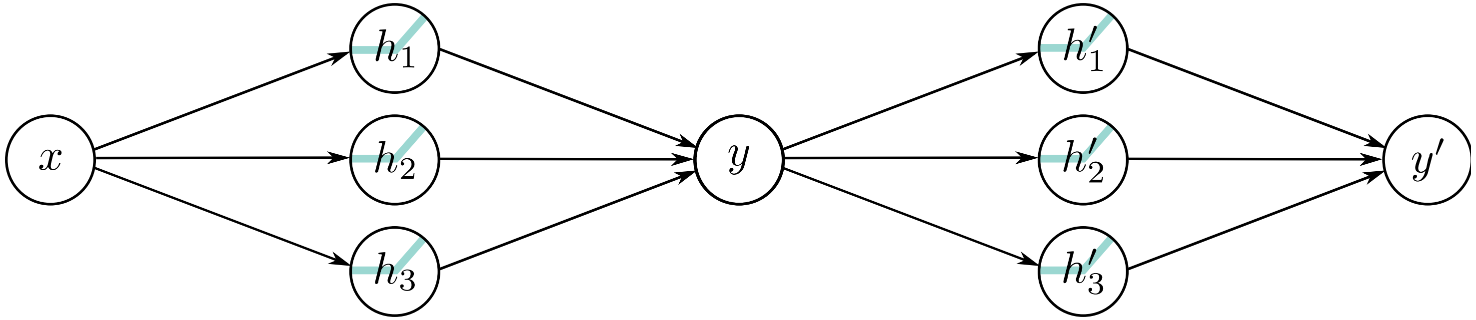
Composing two networks.

Network 1:

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

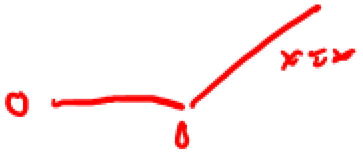
Network 2:

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

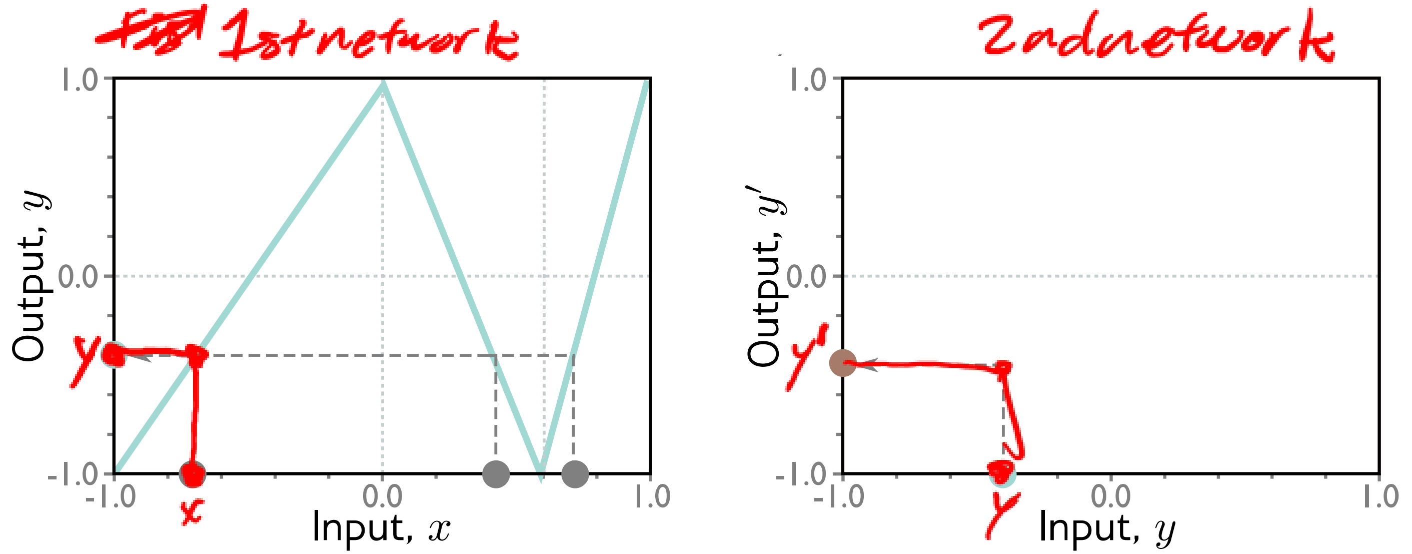


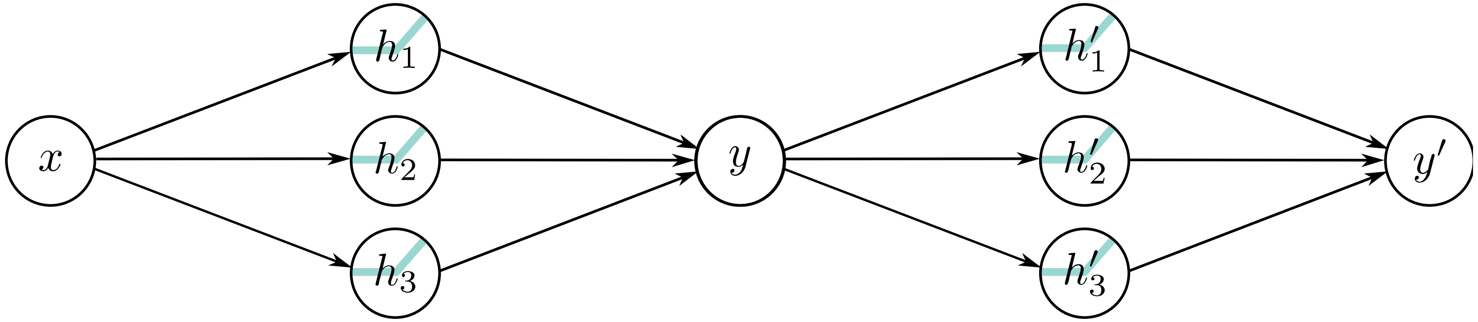
Composing two networks: Example

Assume:

- ReLU Activation 
- Slopes and Intercepts as shown
- 3 hidden units in each

Example: Pick parameters so that
$x \in [-1,1]$ maps to
$y \in [-1,1]$ with alternating slope



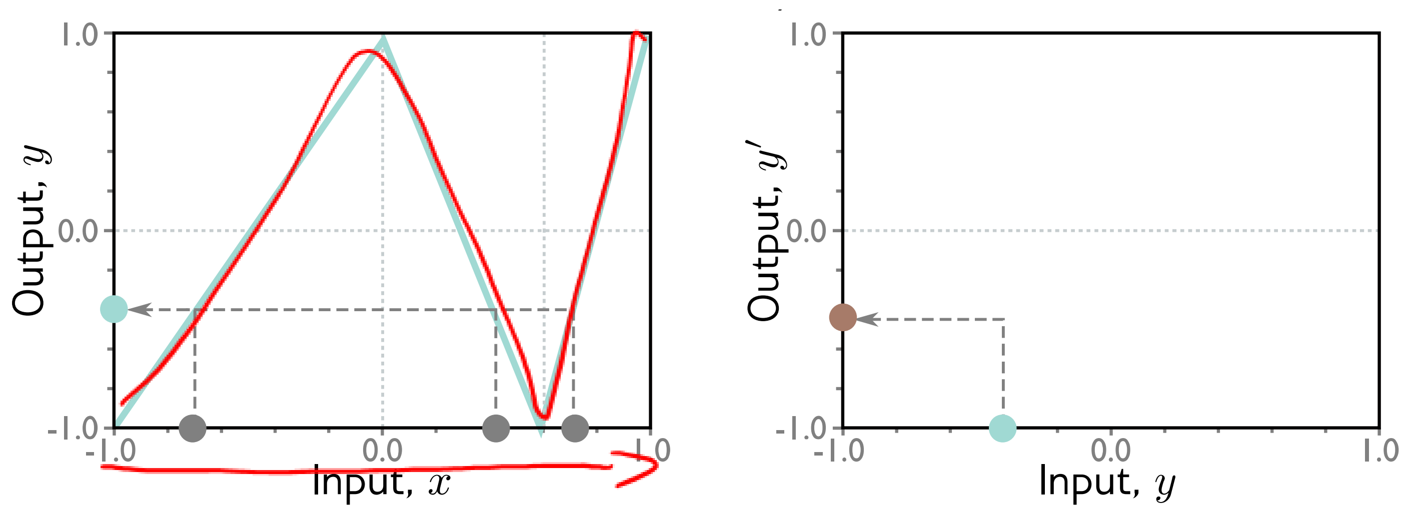


Composing two networks: Example

Assume:

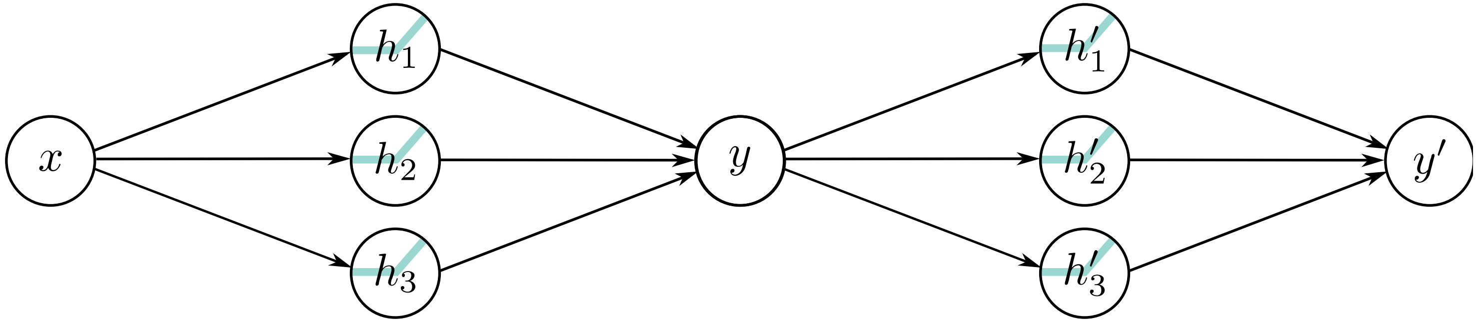
- ReLU Activation
- Slopes and Intercepts as shown
- 3 hidden units in each

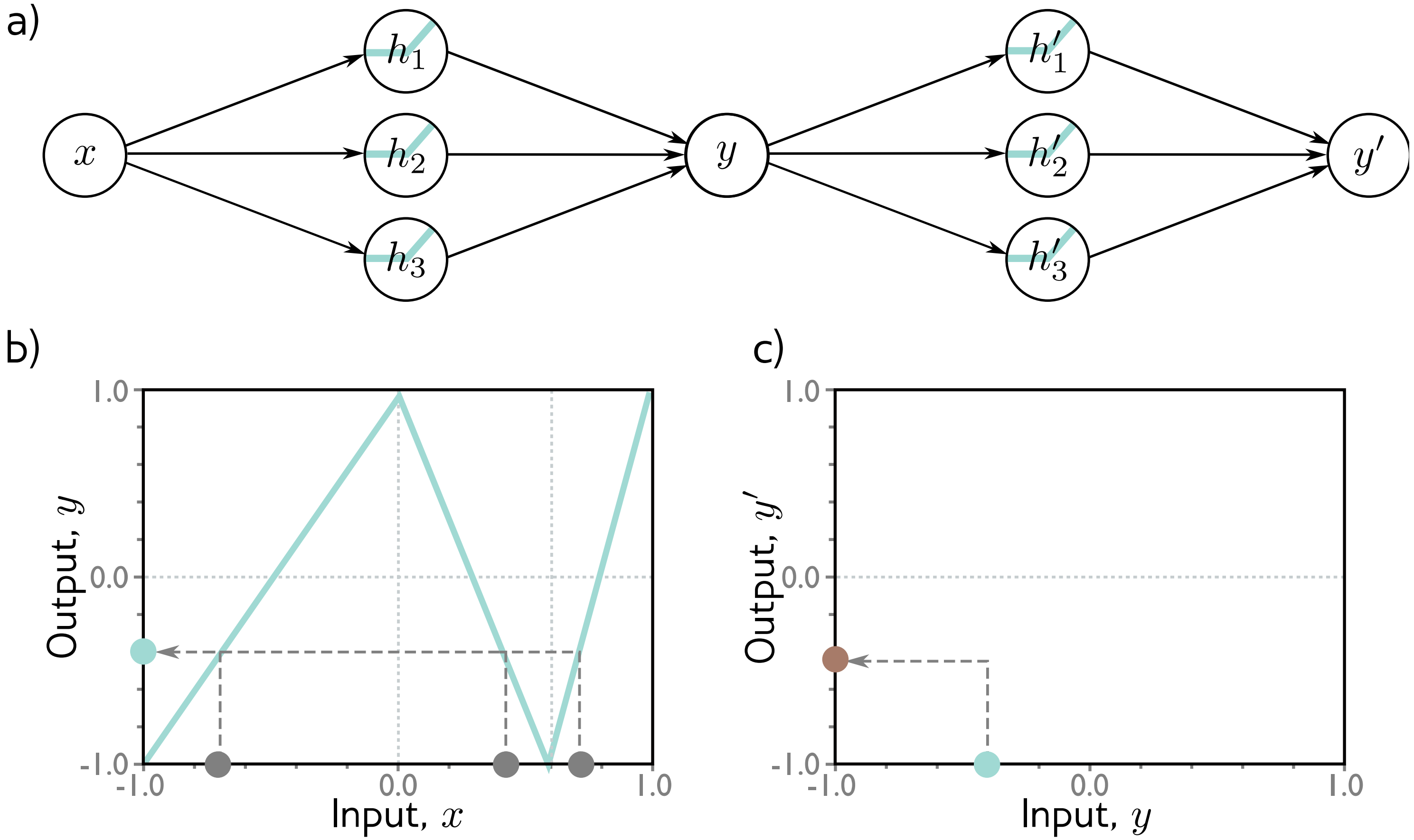
Example: Pick parameters so that
$x \in [-1,1]$ maps to
$y \in [-1,1]$ with alternating slope

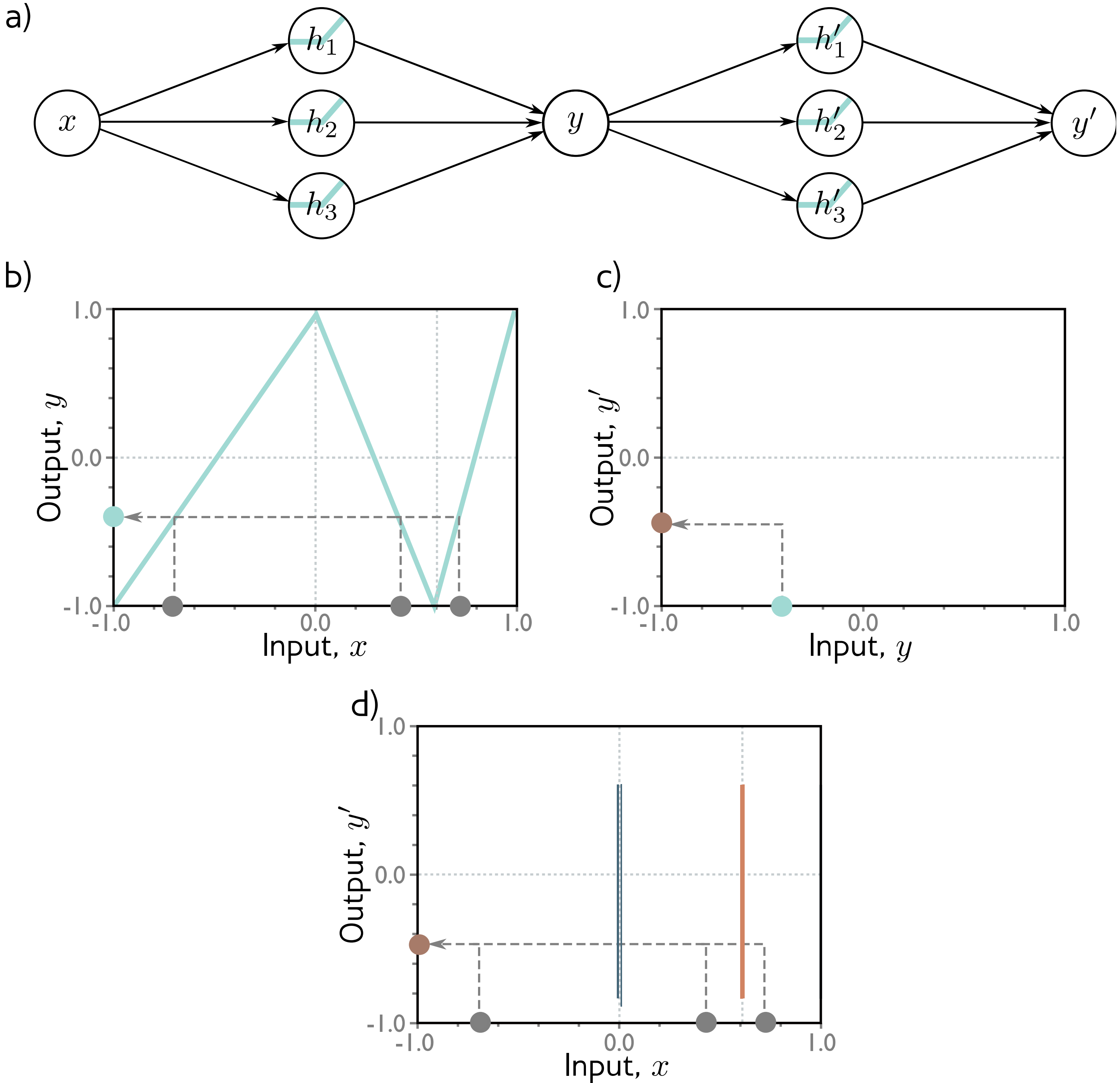


Let's see what happens when we map
$x \to y \to y'$

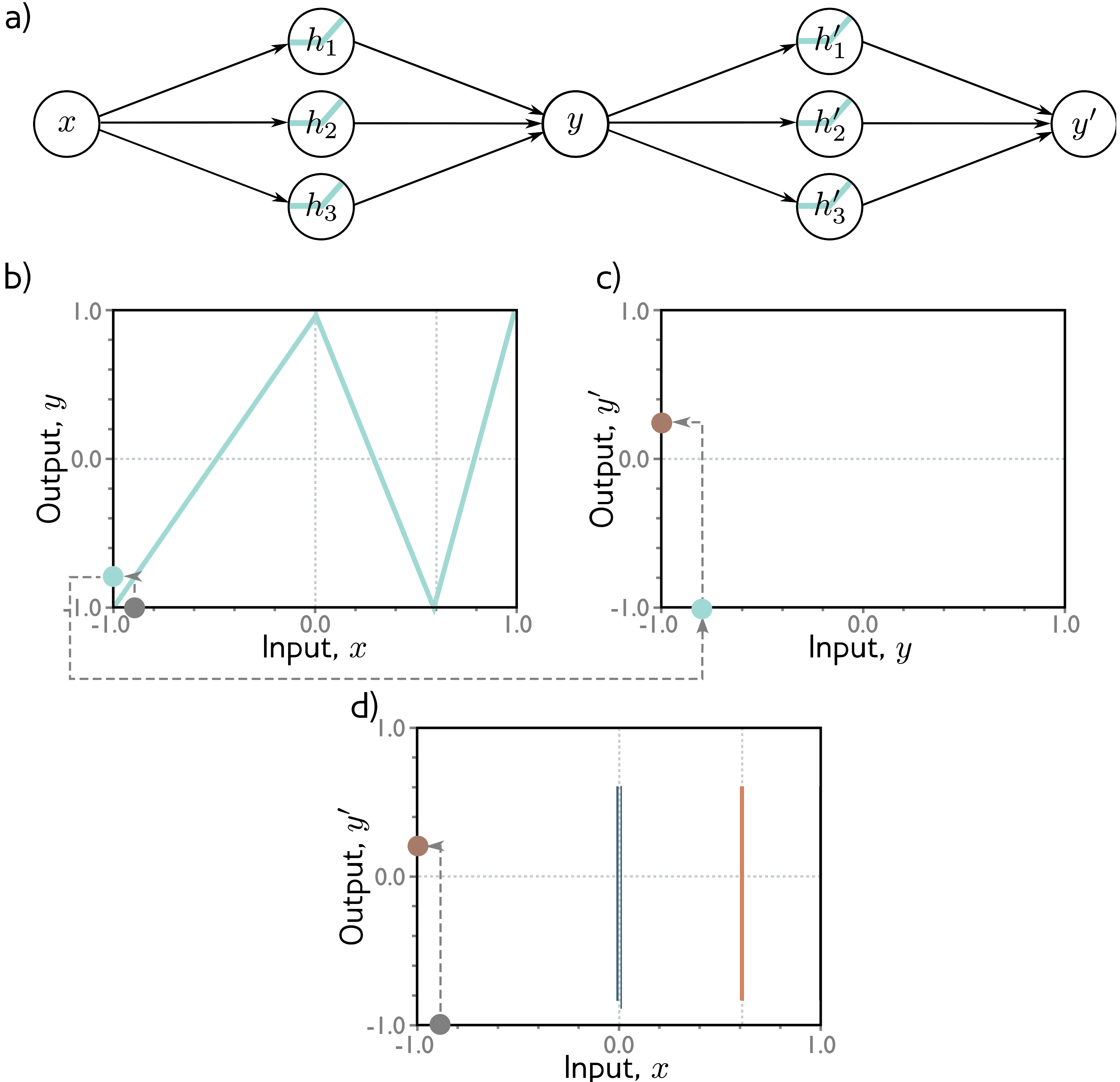
$-1 \to 1$



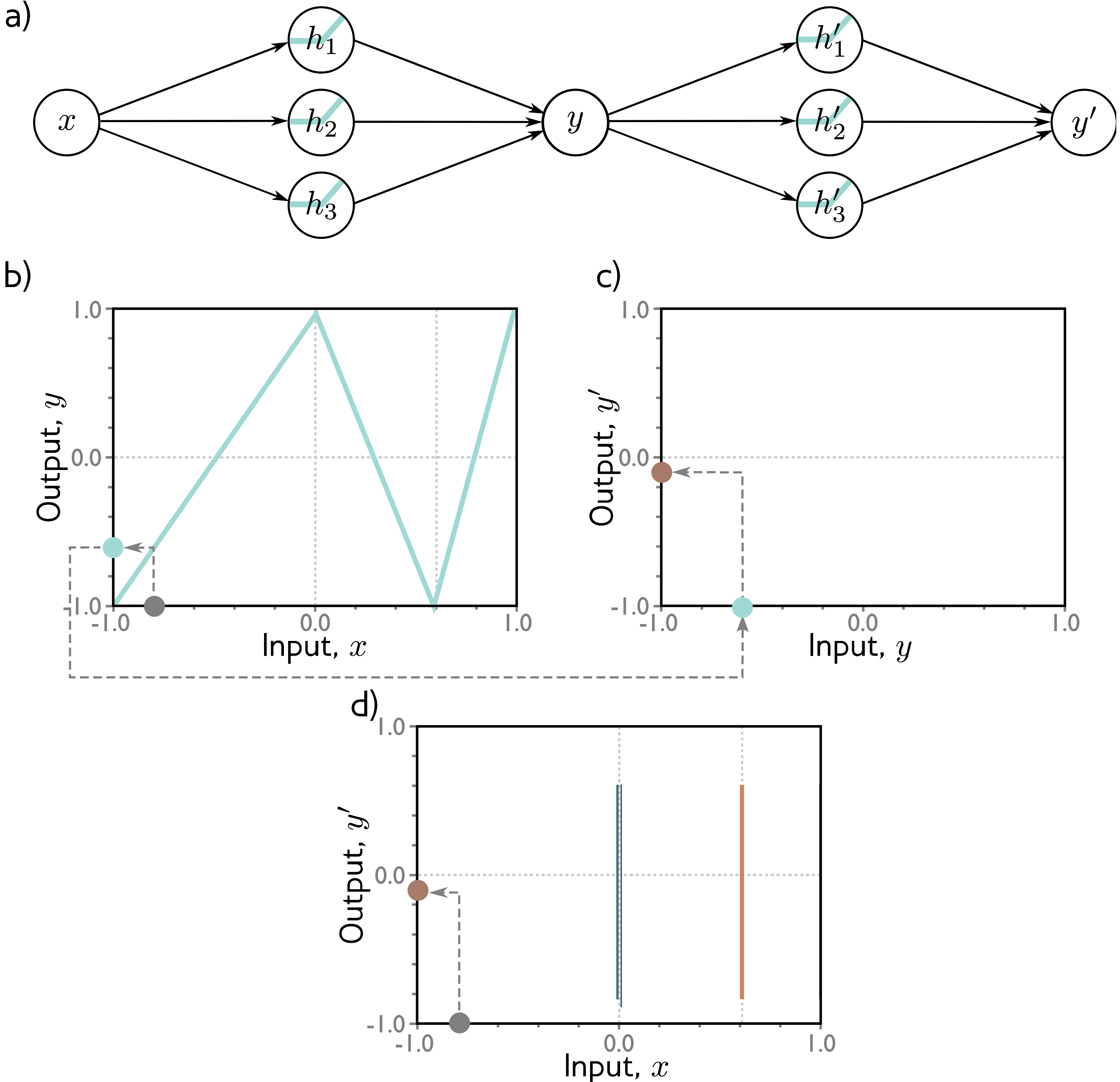

a)
x
h_1
h_2
h_3
y
h'_1
h'_2
h'_3
y'
b)
1.0
0.0
-1.0
-1.0
0.0
1.0
Output, y
Input, x
c)
1.0
0.0
-1.0
-1.0
0.0
1.0
Output, y'
Input, y



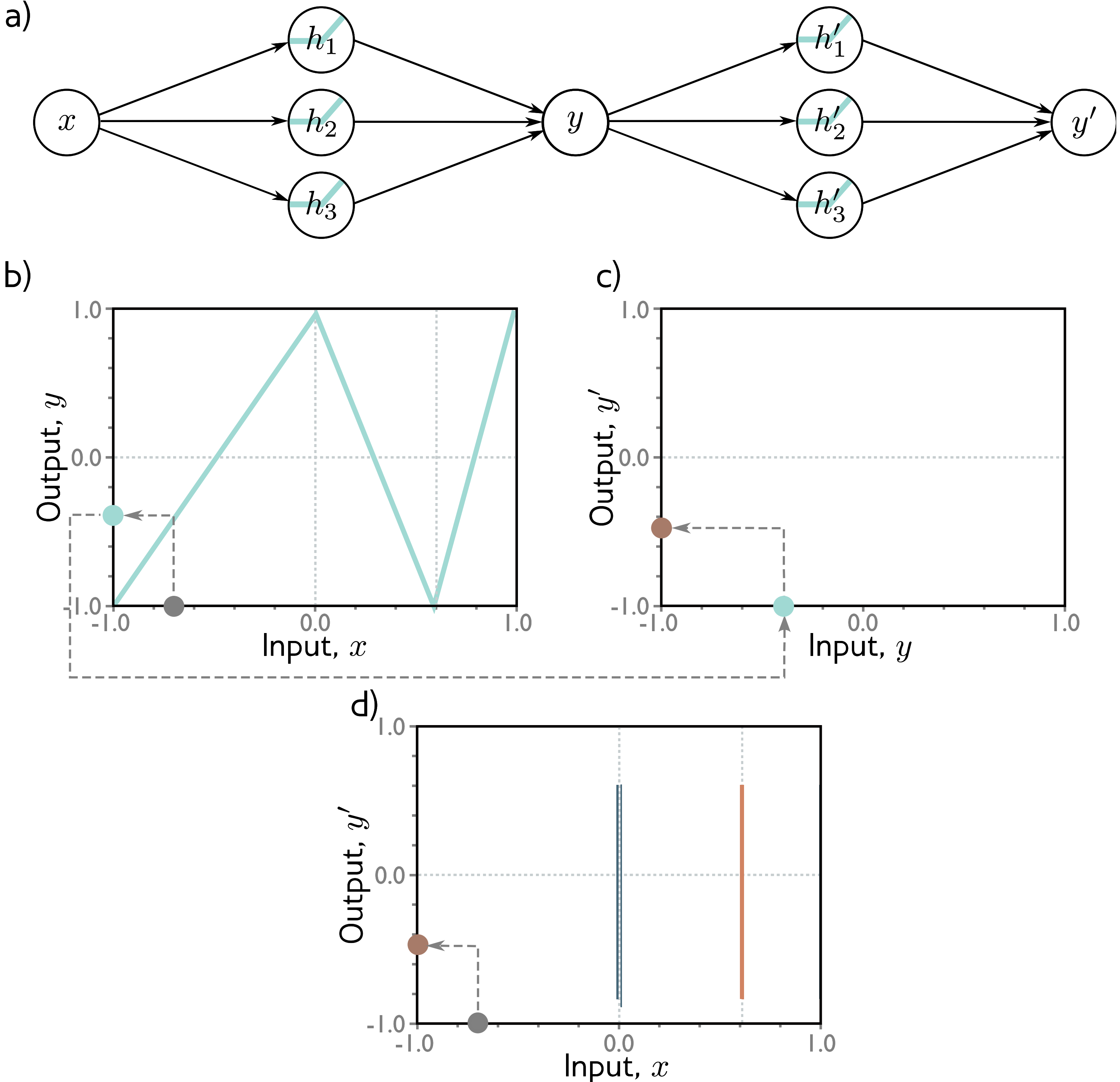
a)
x
h_1
h_2
h_3
y
h'_1
h'_2
h'_3
y'
b)
Output, y
Input, x
1.0
0.0
-1.0
-1.0
0.0
1.0
c)
Output, y'
Input, y
1.0
0.0
-1.0
-1.0
0.0
1.0
d)
Output, y'
Input, x
1.0
0.0
-1.0
-1.0
0.0
1.0



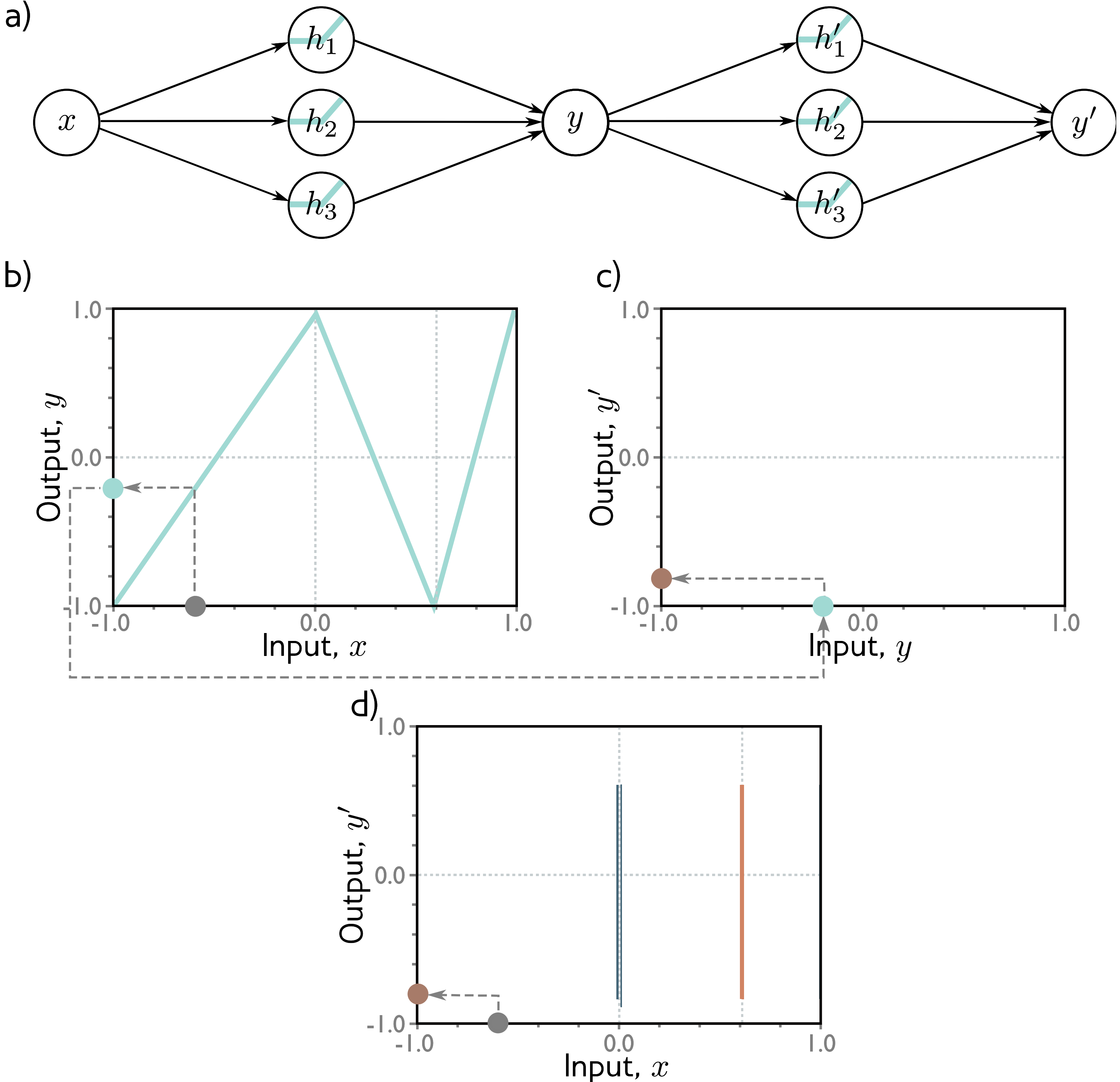
a)
x
h_1
h_2
h_3
y
h'_1
h'_2
h'_3
y'
b)
Output, y
Input, x
1.0
0.0
-1.0
c)
Output, y'
Input, y
d)
Output, y'
Input, x

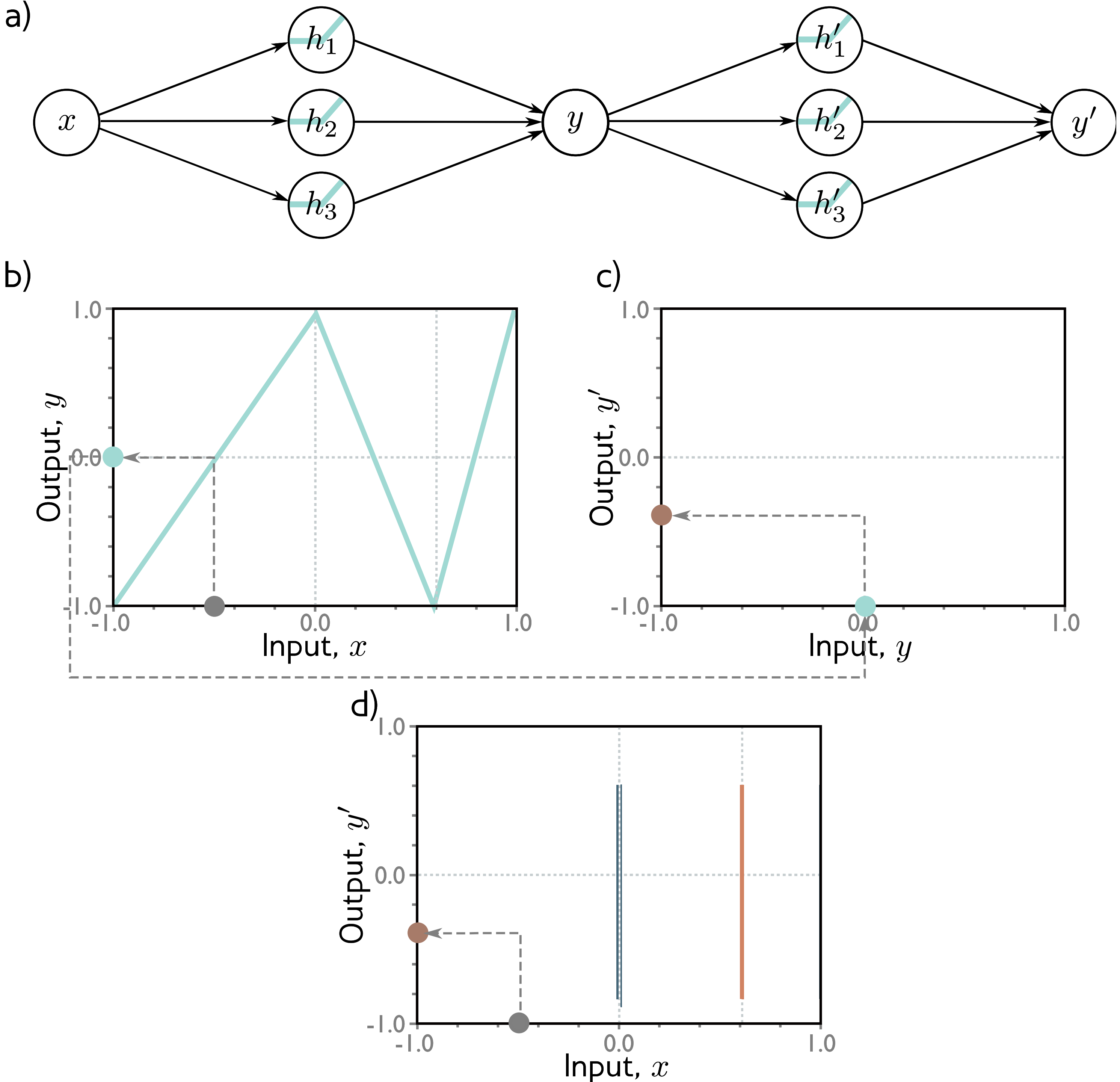


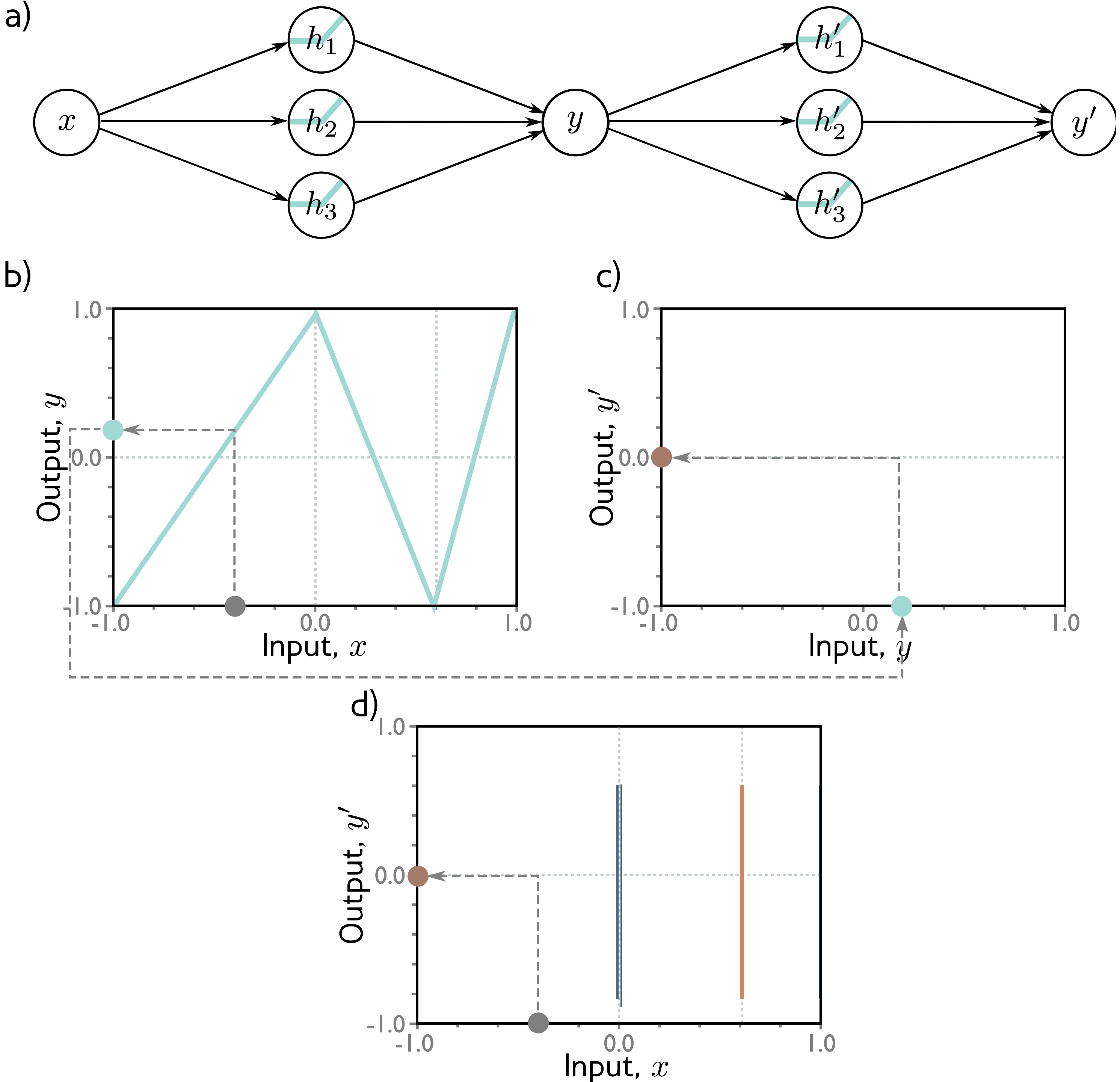
a)
x
h_1
h_2
h_3
y
h'_1
h'_2
h'_3
y'
b)
Output, y
Input, x
c)
Output, y'
Input, y
d)
Output, y'
Input, x
1.0
0.0
-1.0

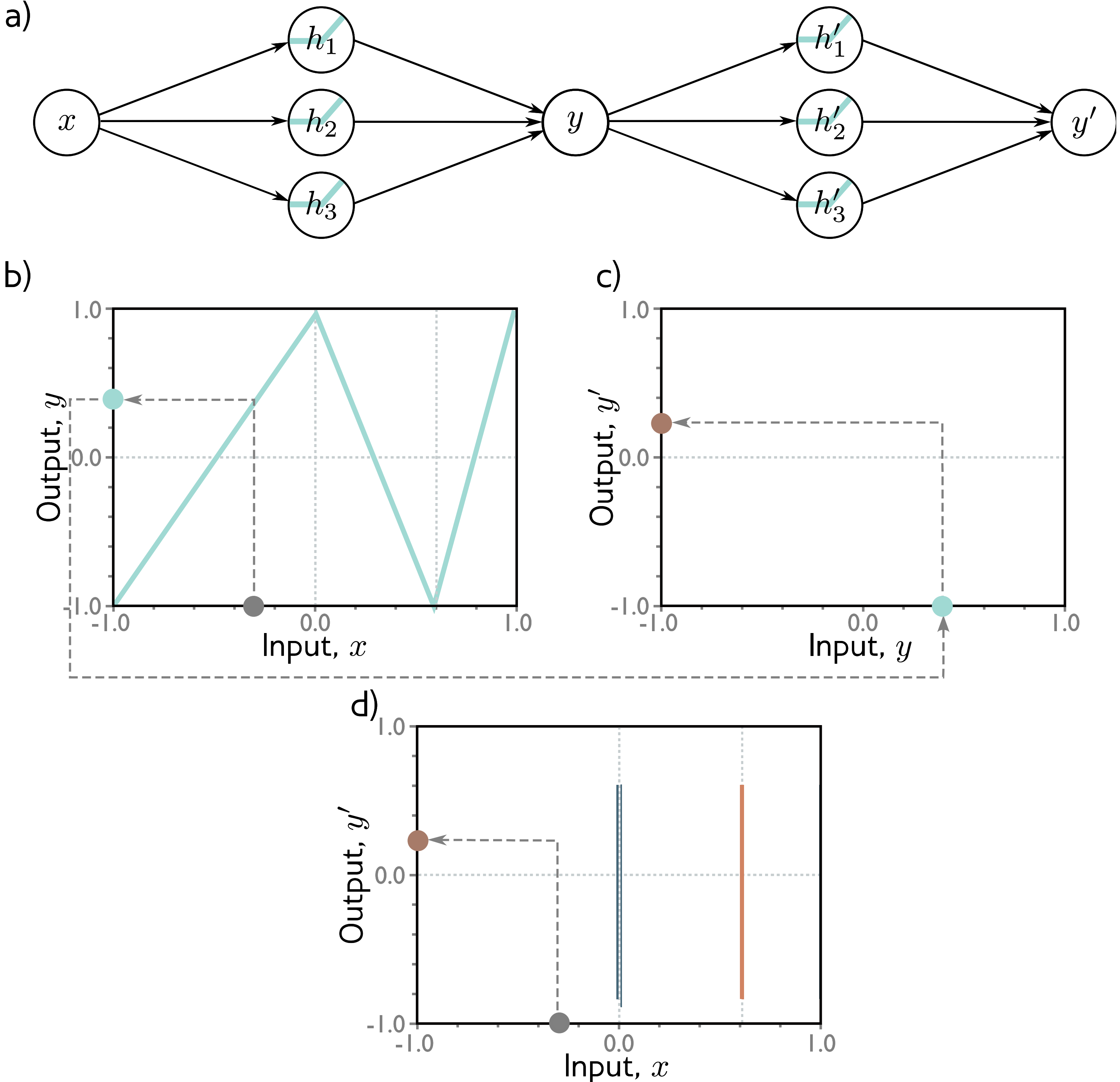


a)
x
h_1
h_2
h_3
y
h'_1
h'_2
h'_3
y'
b)
Output, y
Input, x
1.0
0.0
-1.0
-1.0
0.0
1.0
c)
Output, y'
Input, y
1.0
0.0
-1.0
-1.0
0.0
1.0
d)
Output, y'
Input, x
1.0
0.0
-1.0
-1.0
0.0
1.0

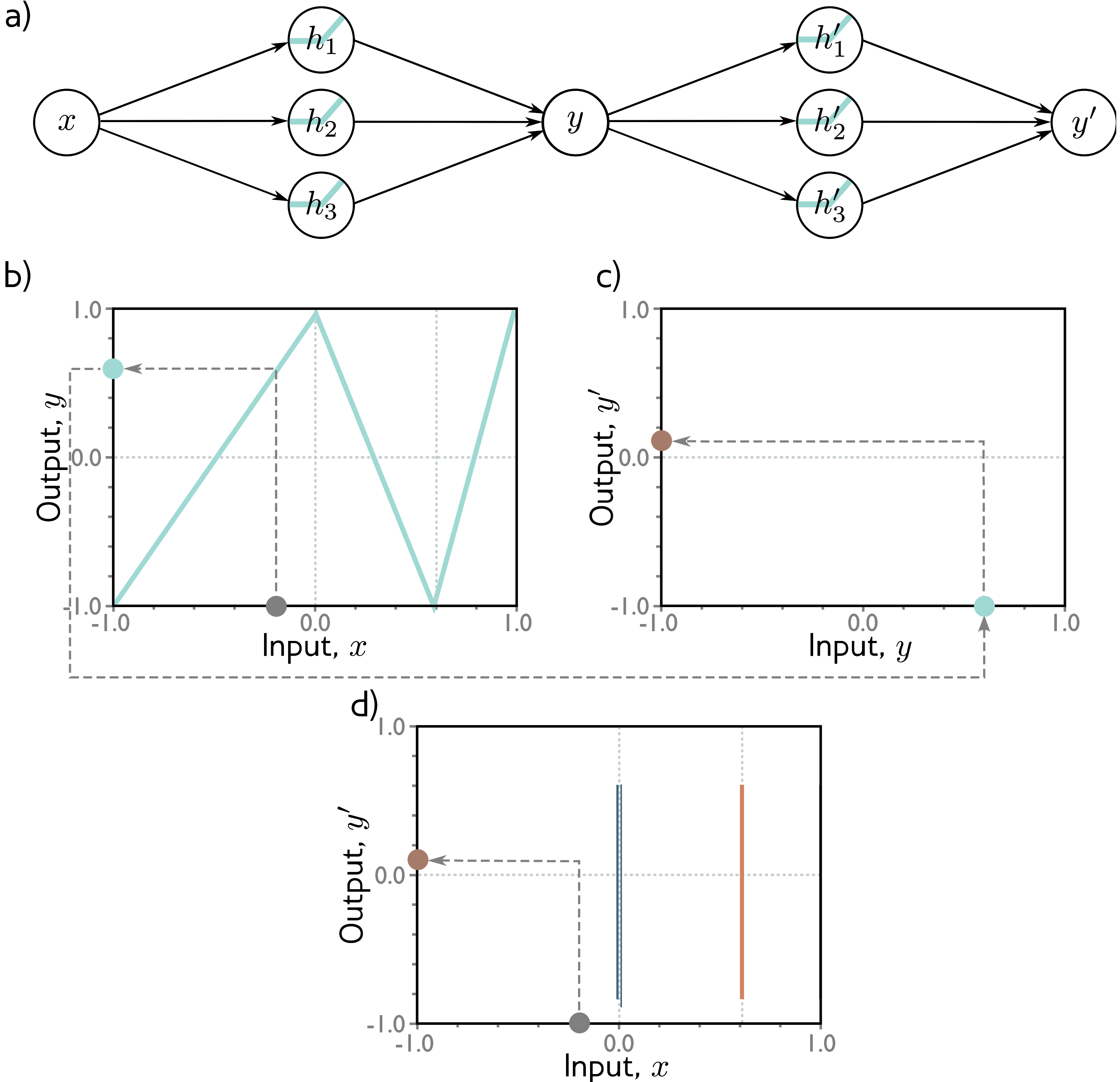

a)
x
h_1
h_2
h_3
y
h'_1
h'_2
h'_3
y'
b)
Output, y
Input, x
1.0
0.0
-1.0
c)
Output, y'
Input, y
d)
Output, y'
Input, x


a)
x
h_1
h_2
h_3
y
h'_1
h'_2
h'_3
y'
b)
c)
d)
1.0
0.0
-1.0
Output, y
Input, x
Output, y'
Input, y

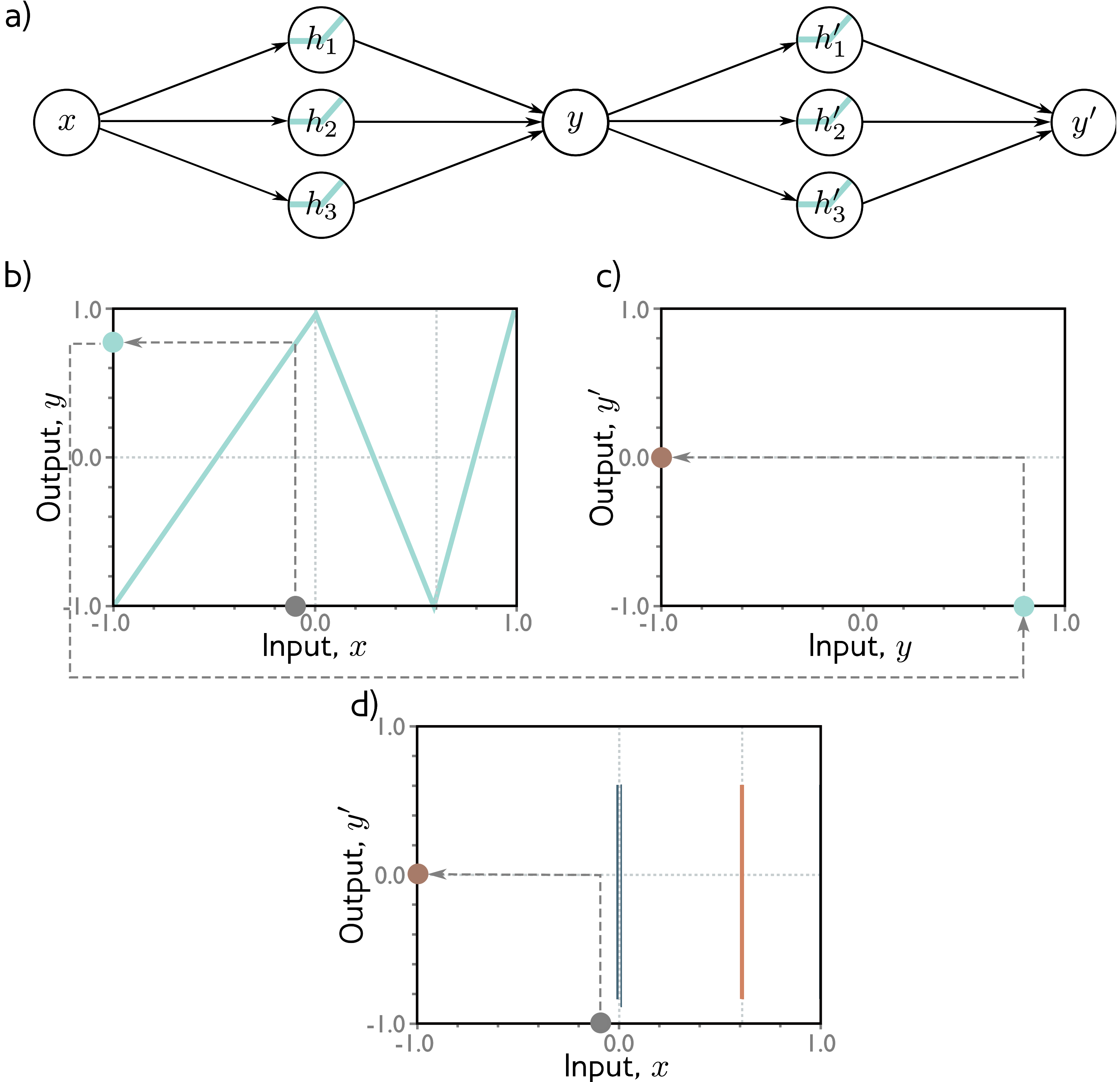

a)
x
h_1
h_2
h_3
y
h'_1
h'_2
h'_3
y'
b)
Output, y
Input, x
c)
Output, y'
Input, y
d)
Output, y'
Input, x
1.0
0.0
-1.0



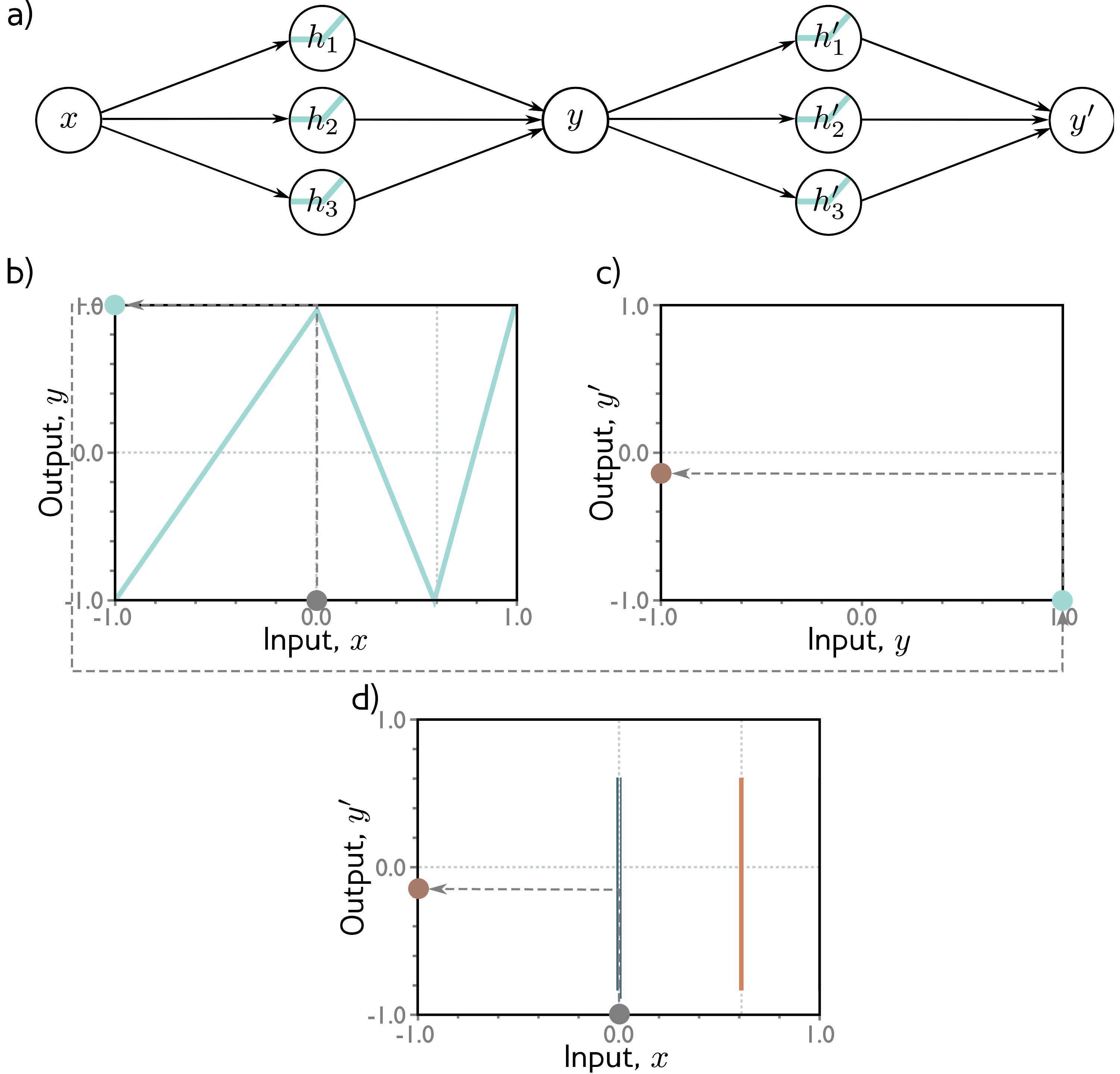
a)
x
h_1
h_2
h_3
y
h'_1
h'_2
h'_3
y'
b)
Output, y
Input, x
1.0
0.0
-1.0
-1.0
0.0
1.0
c)
Output, y'
Input, y
1.0
0.0
-1.0
-1.0
0.0
1.0
d)
Output, y'
Input, x
1.0
0.0
-1.0
-1.0
0.0
1.0



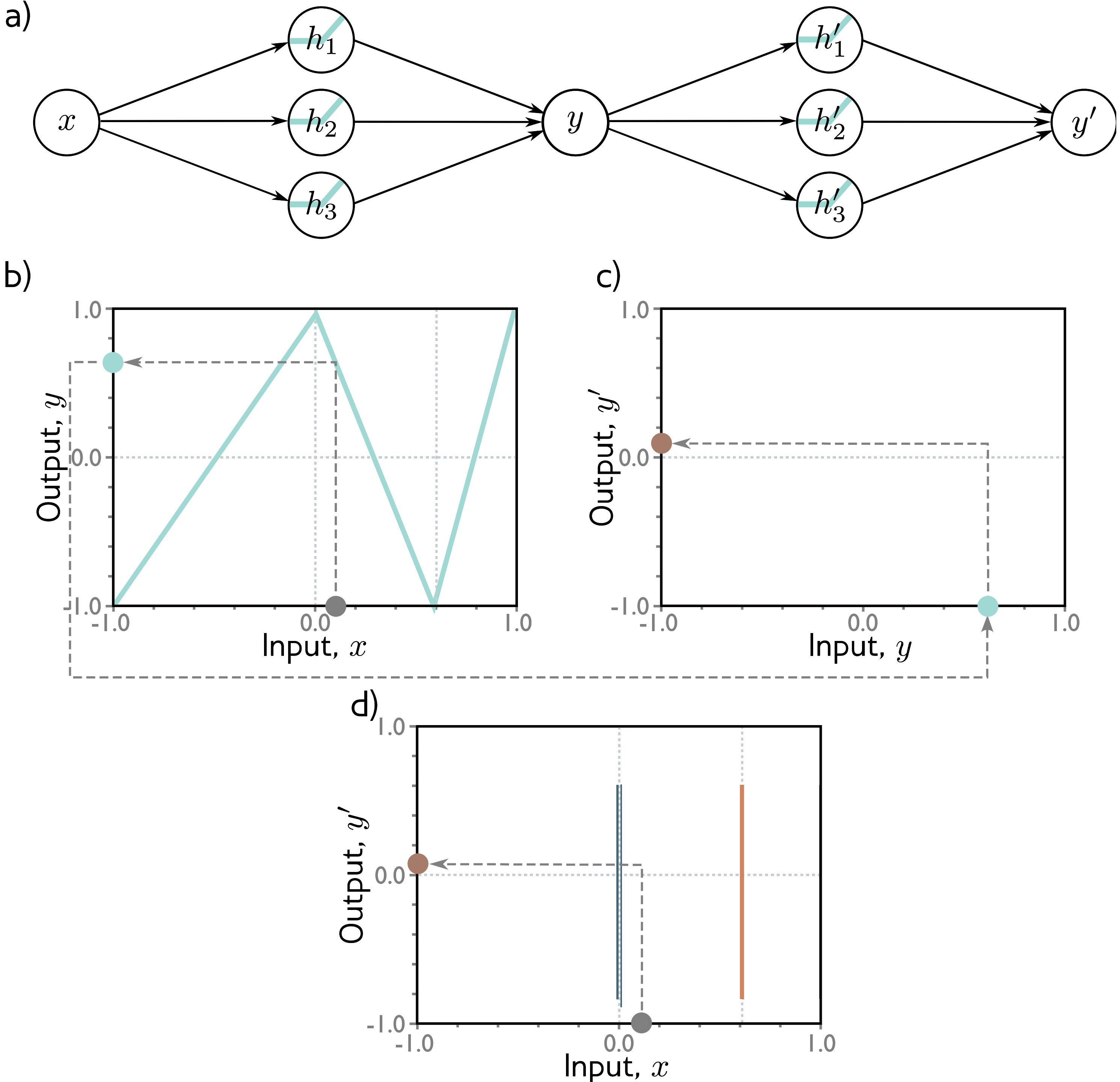
a)
x
h_1
h_2
h_3
y
h'_1
h'_2
h'_3
y'
b)
1.0
0.0
-1.0
-1.0
0.0
1.0
Output, y
Input, x
c)
1.0
0.0
-1.0
-1.0
0.0
1.0
Output, y'
Input, y
d)
1.0
0.0
-1.0
-1.0
0.0
1.0
Output, y'
Input, x



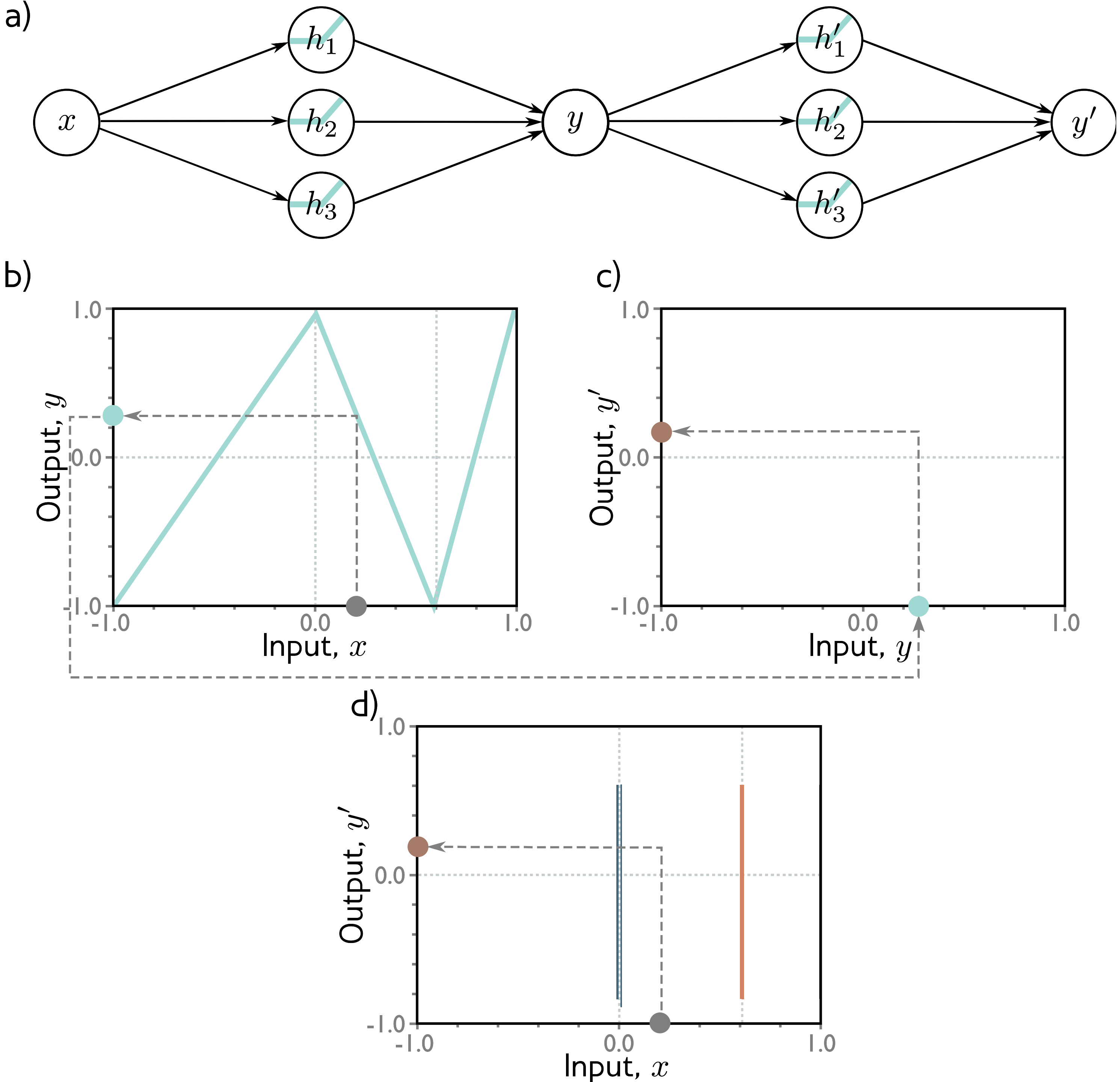
a)
x
h_1
h_2
h_3
y
h'_1
h'_2
h'_3
y'
b)
1.0
0.0
-1.0
-1.0
0.0
1.0
Output, y
Input, x
c)
1.0
0.0
-1.0
-1.0
0.0
1.0
Output, y'
Input, y
d)
1.0
0.0
-1.0
-1.0
0.0
1.0
Output, y'
Input, x

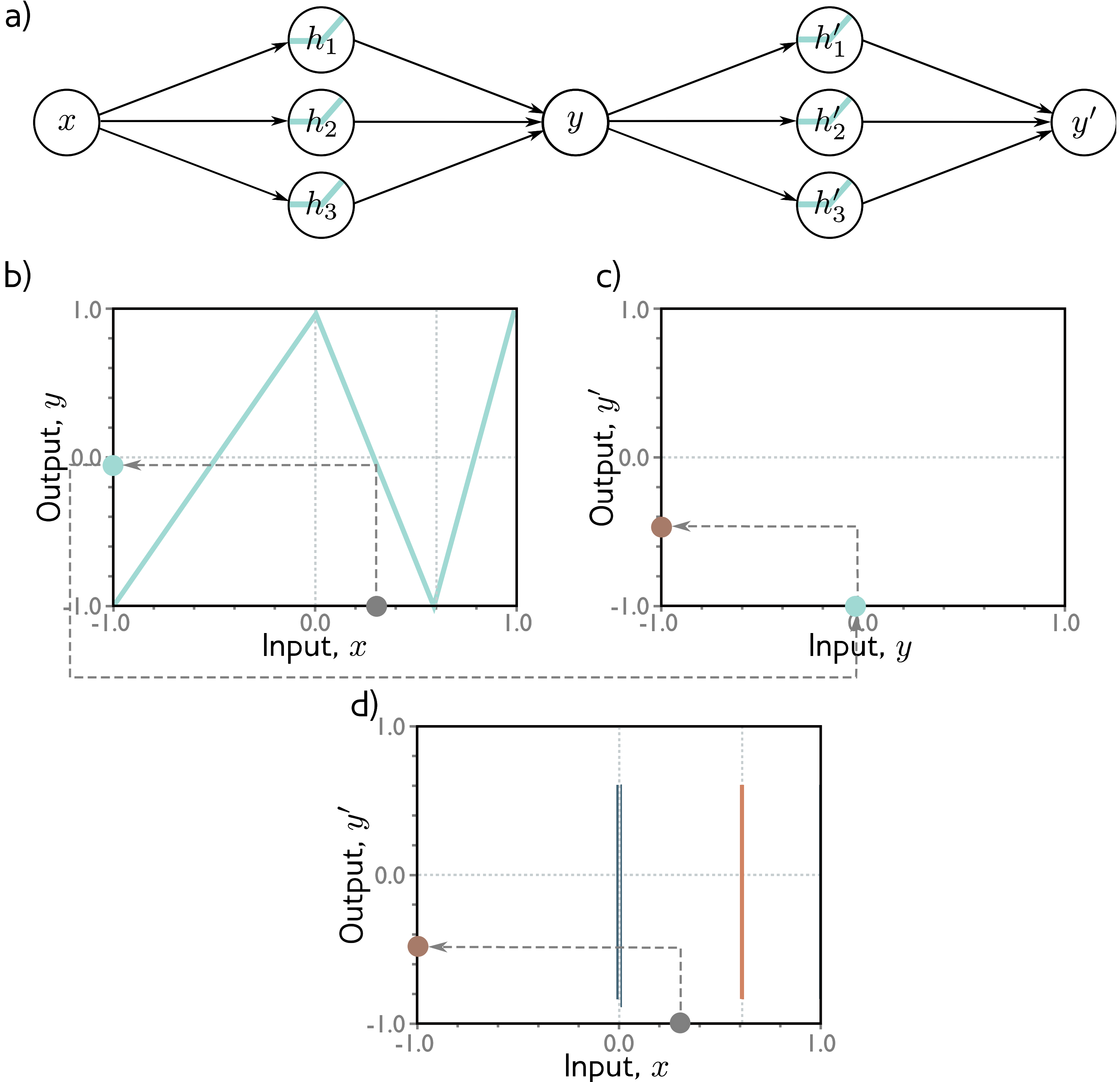


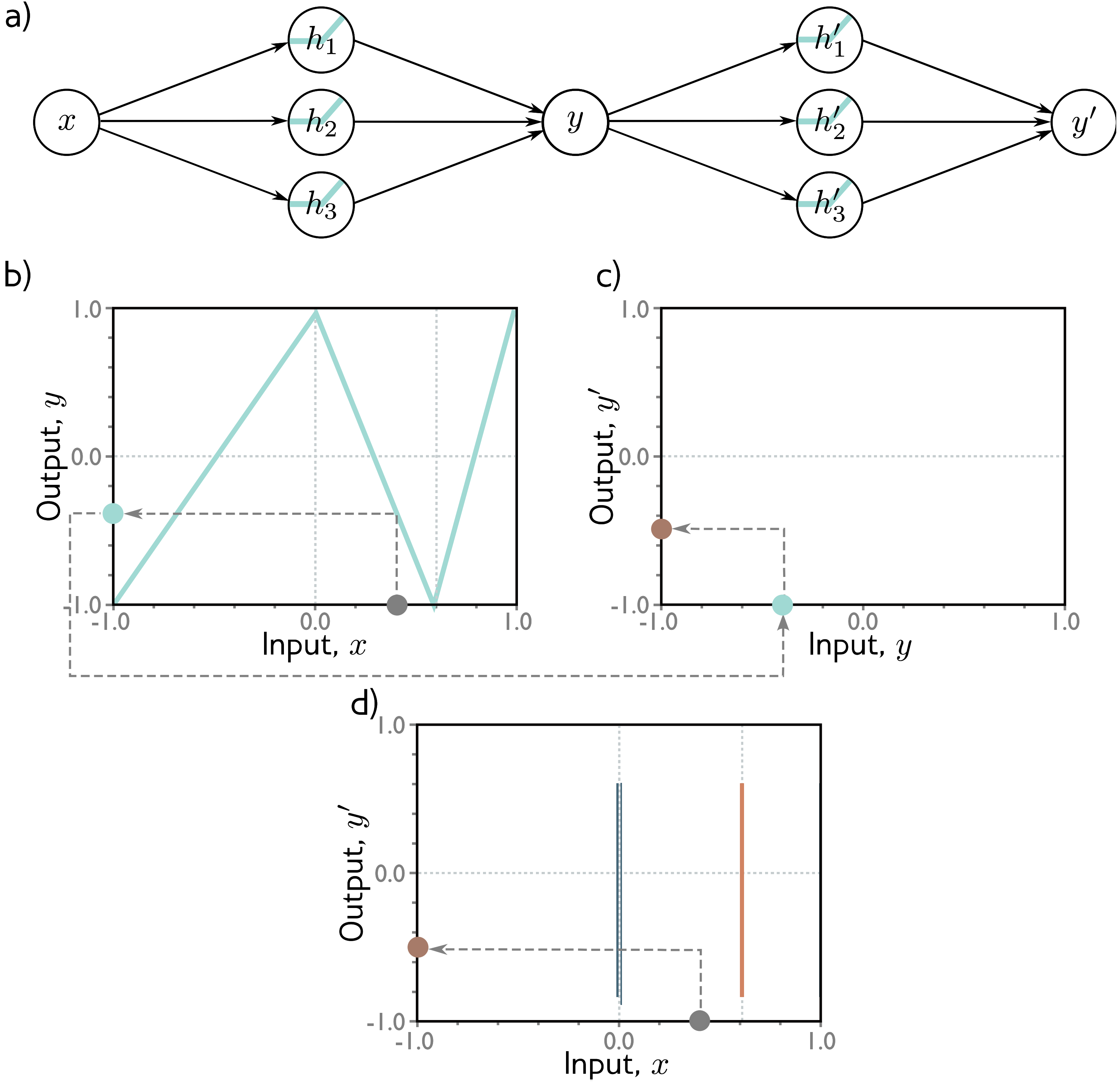
a)
x
h_1
h_2
h_3
y
h'_1
h'_2
h'_3
y'
b)
1.0
0.0
-1.0
-1.0
0.0
1.0
Output, y
Input, x
c)
1.0
0.0
-1.0
-1.0
0.0
1.0
Output, y'
Input, y
d)
1.0
0.0
-1.0
-1.0
0.0
1.0
Output, y'
Input, x



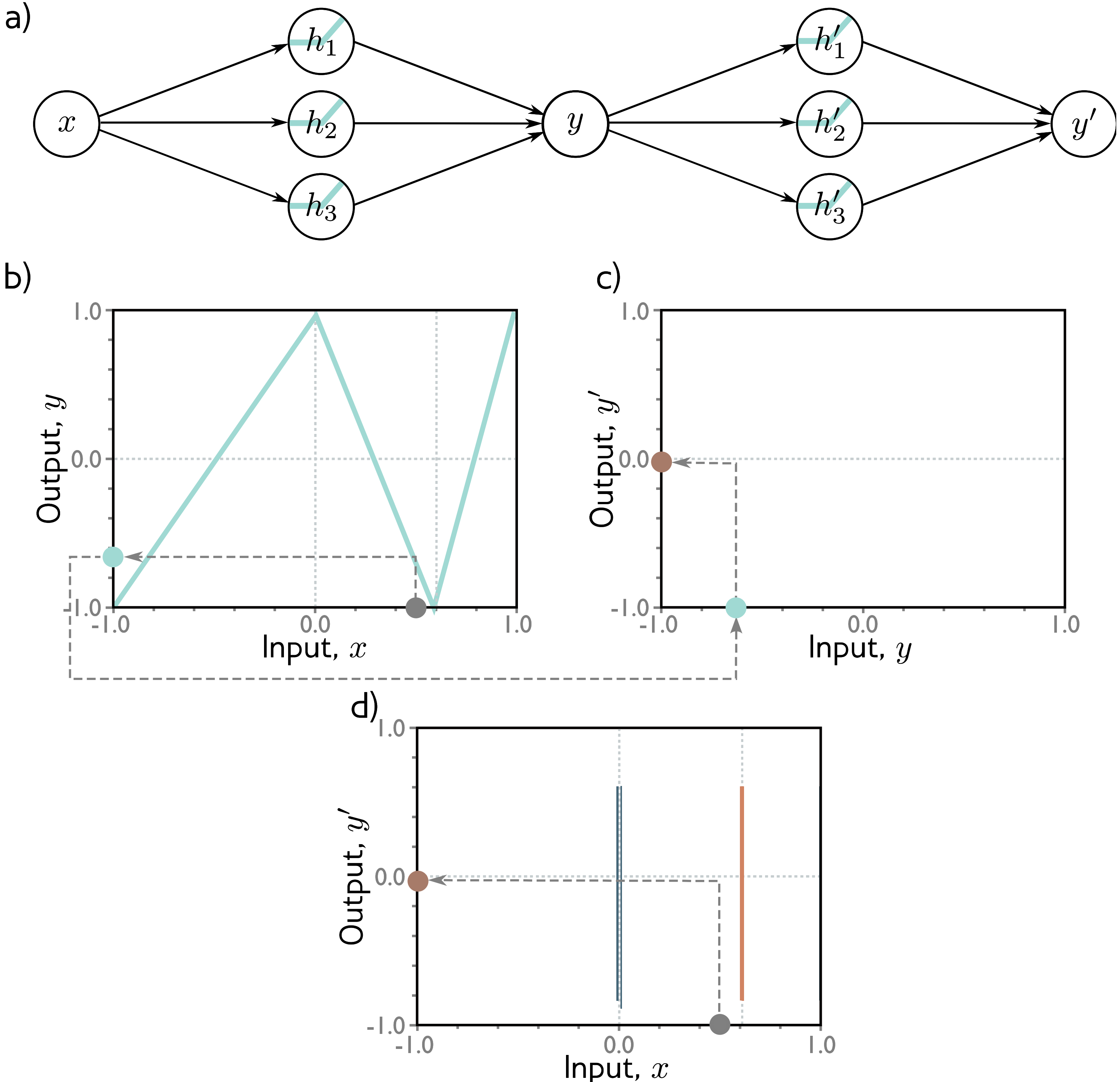
a)
x
h_1
h_2
h_3
y
h'_1
h'_2
h'_3
y'
b)
1.0
0.0
-1.0
-1.0
0.0
1.0
Output, y
Input, x
c)
1.0
0.0
-1.0
-1.0
0.0
1.0
Output, y'
Input, y
d)
1.0
0.0
-1.0
-1.0
0.0
1.0
Output, y'
Input, x


a)
x
h_1
h_2
h_3
y
h'_1
h'_2
h'_3
y'
b)
Output, y
Input, x
c)
Output, y'
Input, y
d)
Output, y'
Input, x
1.0
0.0
-1.0

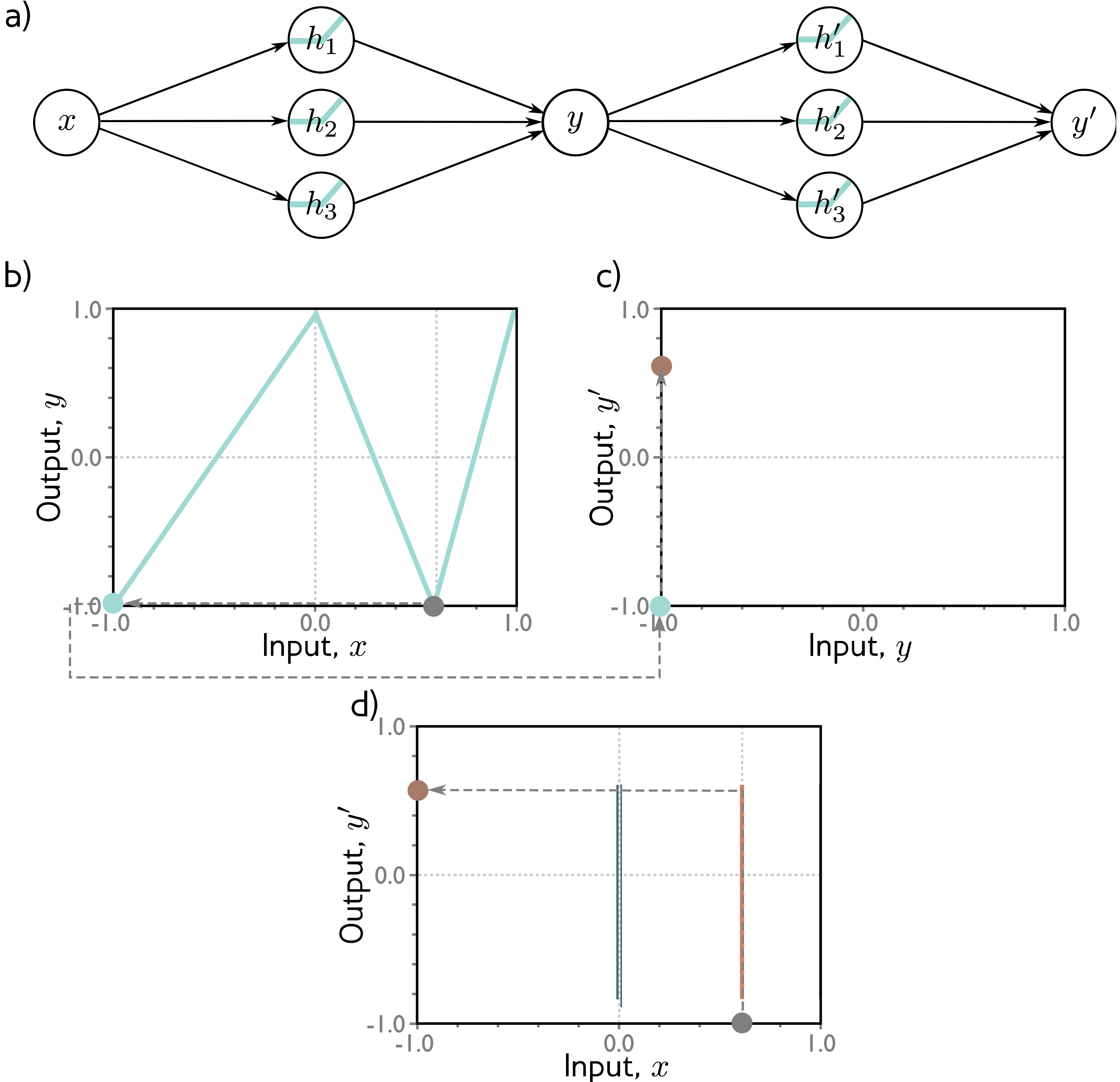

a)
x
h_1
h_2
h_3
y
h'_1
h'_2
h'_3
y'
b)
1.0
0.0
-1.0
-1.0
0.0
1.0
Output, y
Input, x
c)
1.0
0.0
-1.0
-1.0
0.0
1.0
Output, y'
Input, y
d)
1.0
0.0
-1.0
-1.0
0.0
1.0
Output, y'
Input, x

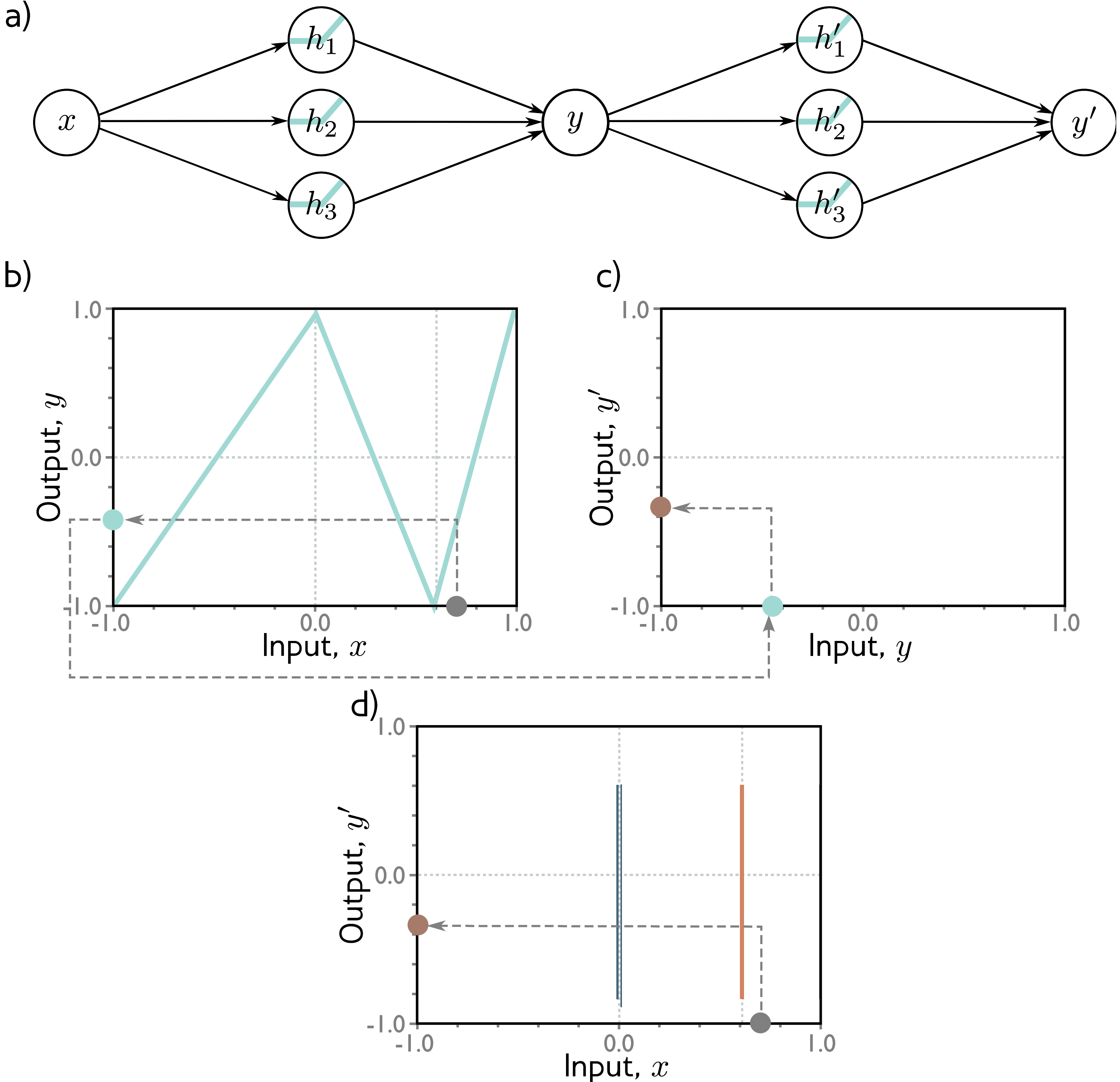


a)
x
h_1
h_2
h_3
y
h'_1
h'_2
h'_3
y'
b)
Output, y
Input, x
1.0
0.0
-1.0
c)
Output, y'
Input, y
1.0
0.0
-1.0
d)
Output, y'
Input, x
1.0
0.0
-1.0

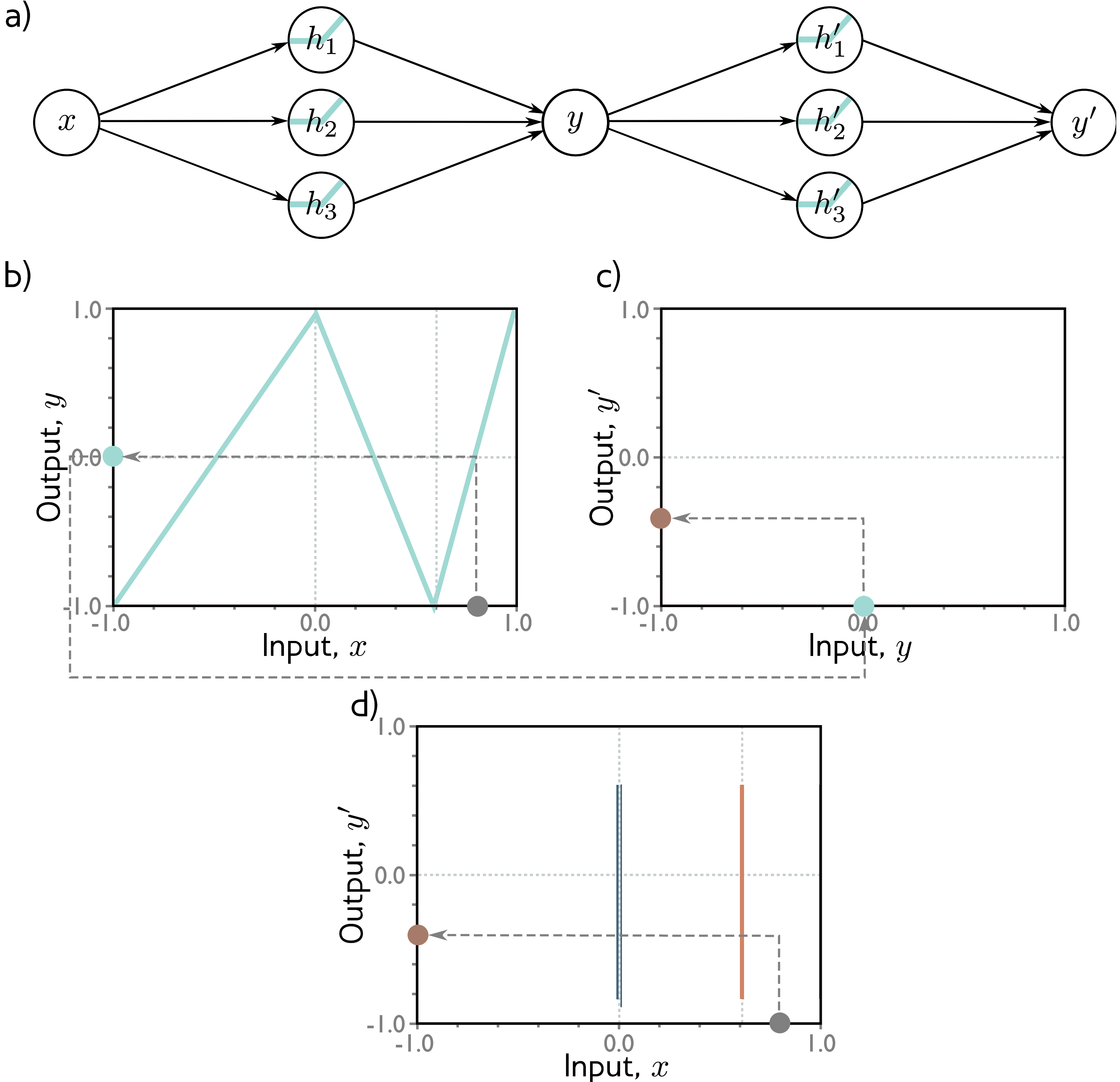


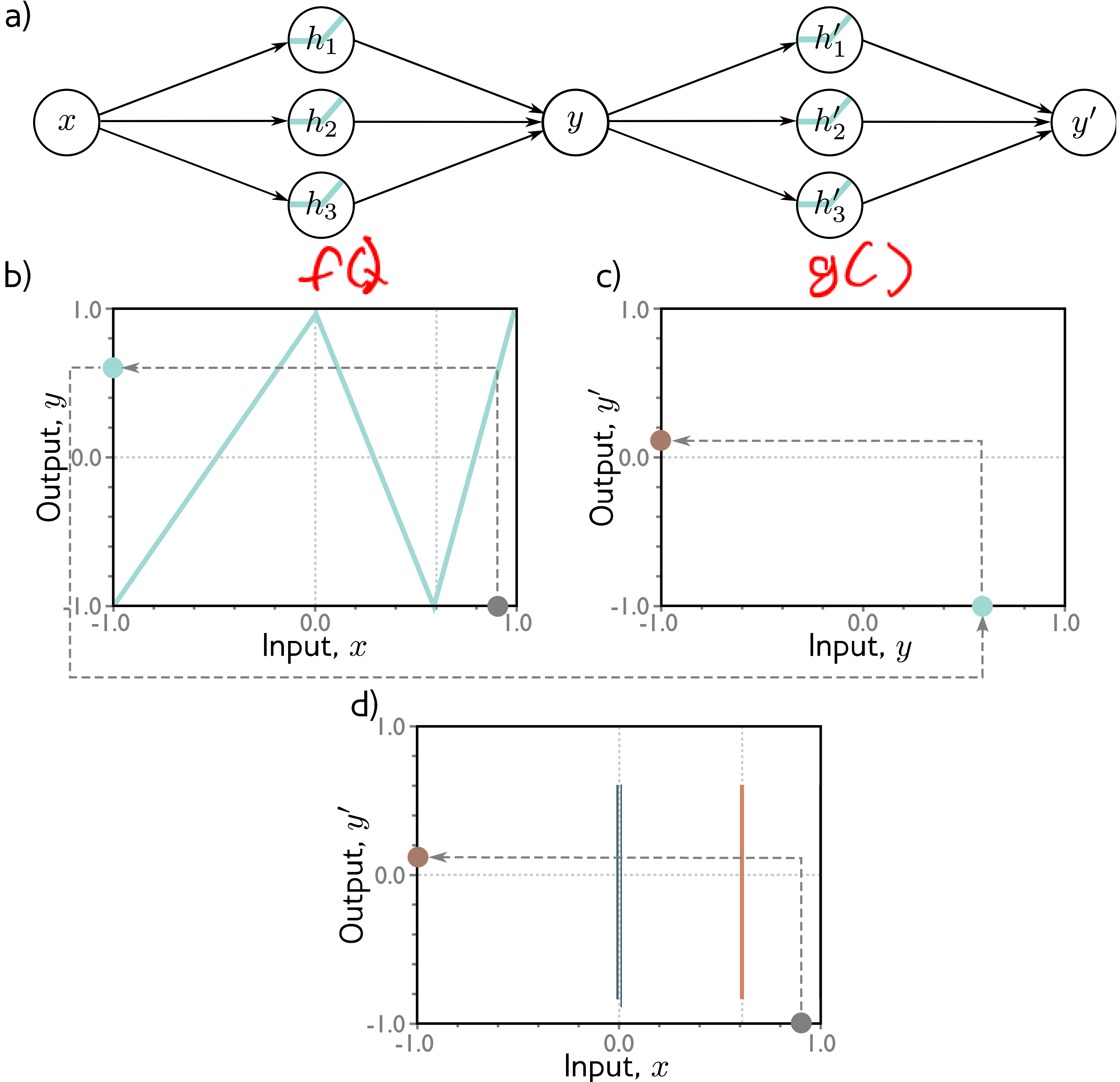
a)
x
h_1
h_2
h_3
y
h'_1
h'_2
h'_3
y'
b)
1.0
0.0
-1.0
-1.0
0.0
1.0
Output, y
Input, x
c)
1.0
0.0
-1.0
-1.0
0.0
1.0
Output, y'
Input, y
d)
1.0
0.0
-1.0
-1.0
0.0
1.0
Output, y'
Input, x


a)
x
h_1
h_2
h_3
y
h'_1
h'_2
h'_3
y'
b)
Output, y
Input, x
c)
Output, y'
Input, y
d)
Output, y'
Input, x
1.0
0.0
-1.0



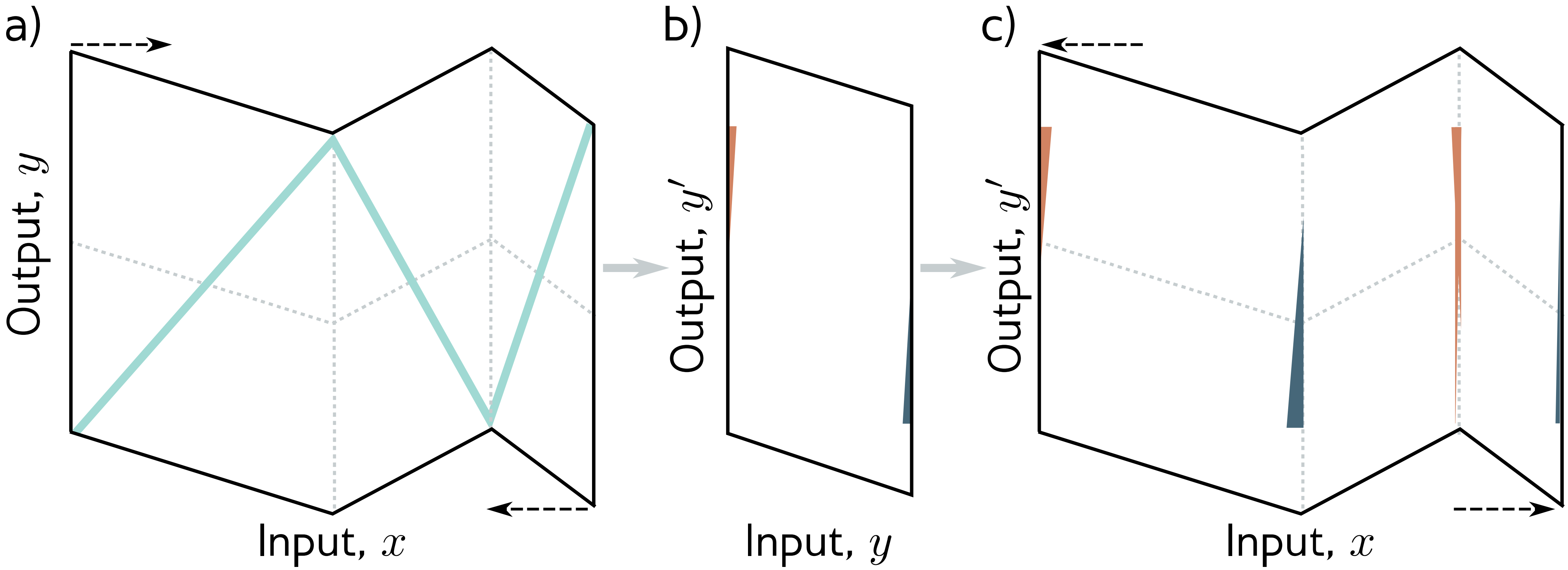
a)
x
h_1
h_2
h_3
y
h'_1
h'_2
h'_3
y'
b)
Output, y
Input, x
1.0
0.0
-1.0
-1.0
0.0
1.0
c)
Output, y'
Input, y
1.0
0.0
-1.0
-1.0
0.0
1.0
d)
Output, y'
Input, x
1.0
0.0
-1.0
-1.0
0.0
1.0


a)
x
h_1
h_2
h_3
y
h'_1
h'_2
h'_3
y'
b)
Output, y
Input, x
c)
Output, y'
Input, y
d)
Output, y'
Input, x
1.0
0.0
-1.0


a)
x
h_1
h_2
h_3
y
h'_1
h'_2
h'_3
y'
b)
f()
c)
g( )
1.0
0.0
-1.0
-1.0
0.0
1.0
Output, y
Input, x
Output, y'
Input, y
d)
Output, y'
Input, x

$g(f(x))$

"Folding analogy"



Interactive Figure 4.1 – Concatenating Nets

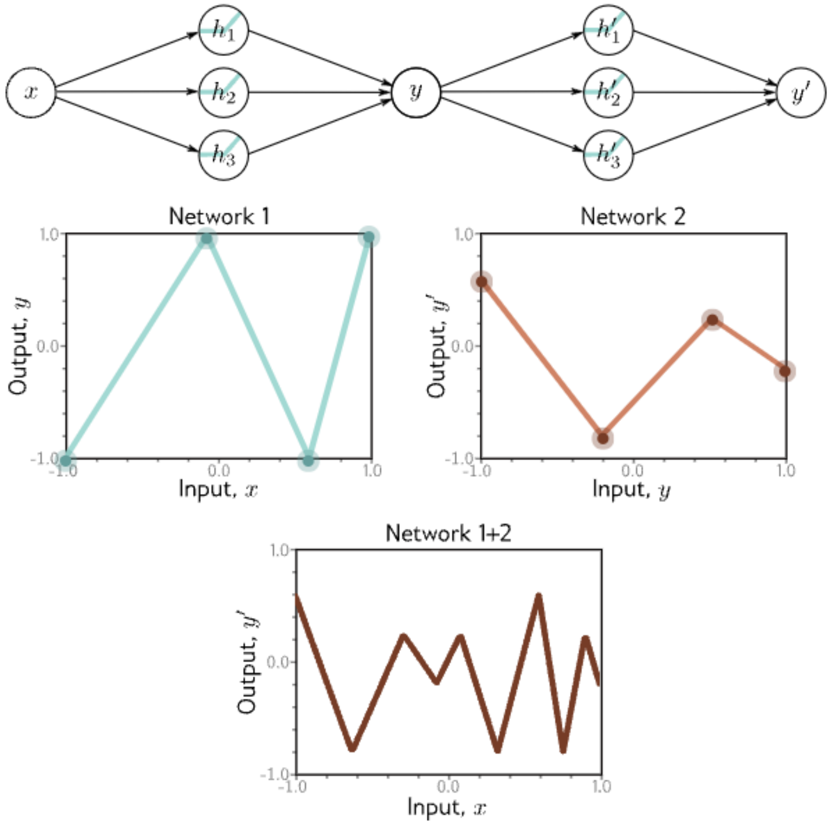
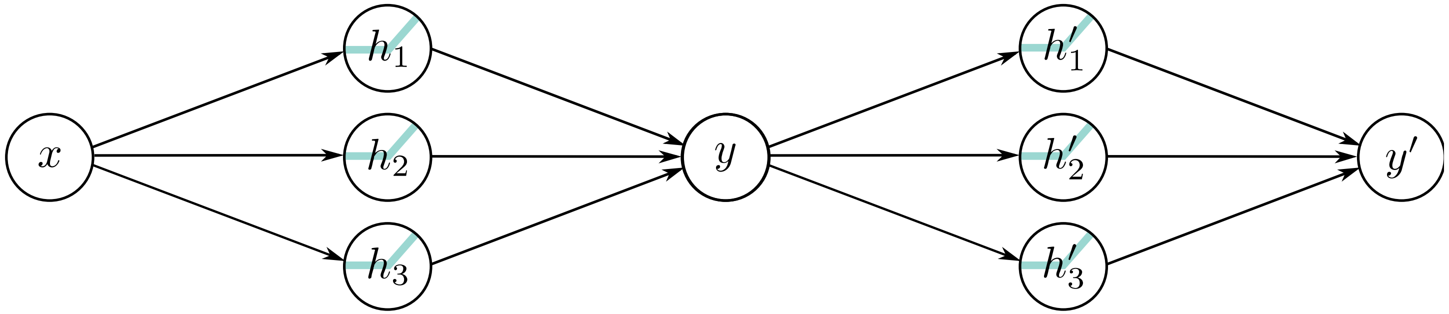


Figure 3.8b Composing two single-layer networks with three hidden units each. The output y of the first network constitutes the input to the second network. (Top left) The first network maps inputs $x \in [-1, 1]$ to outputs $y \in [-1, 1]$ to outputs using a function comprising three linear regions (fourth linear region is outside range of graph). (Top right) The second network defines a function comprising three linear regions that takes y and returns y'. (Bottom) The combined effect of these two functions when composed.

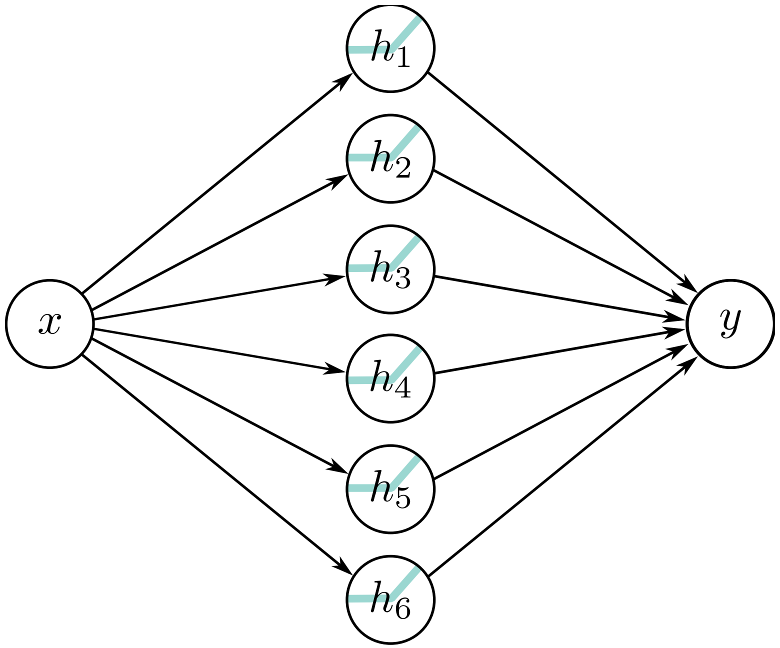
Manipulate the functions defined by the two shallow networks (using the circular handles) to see the effect of composing the functions.

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

Comparing to shallow with six hidden units

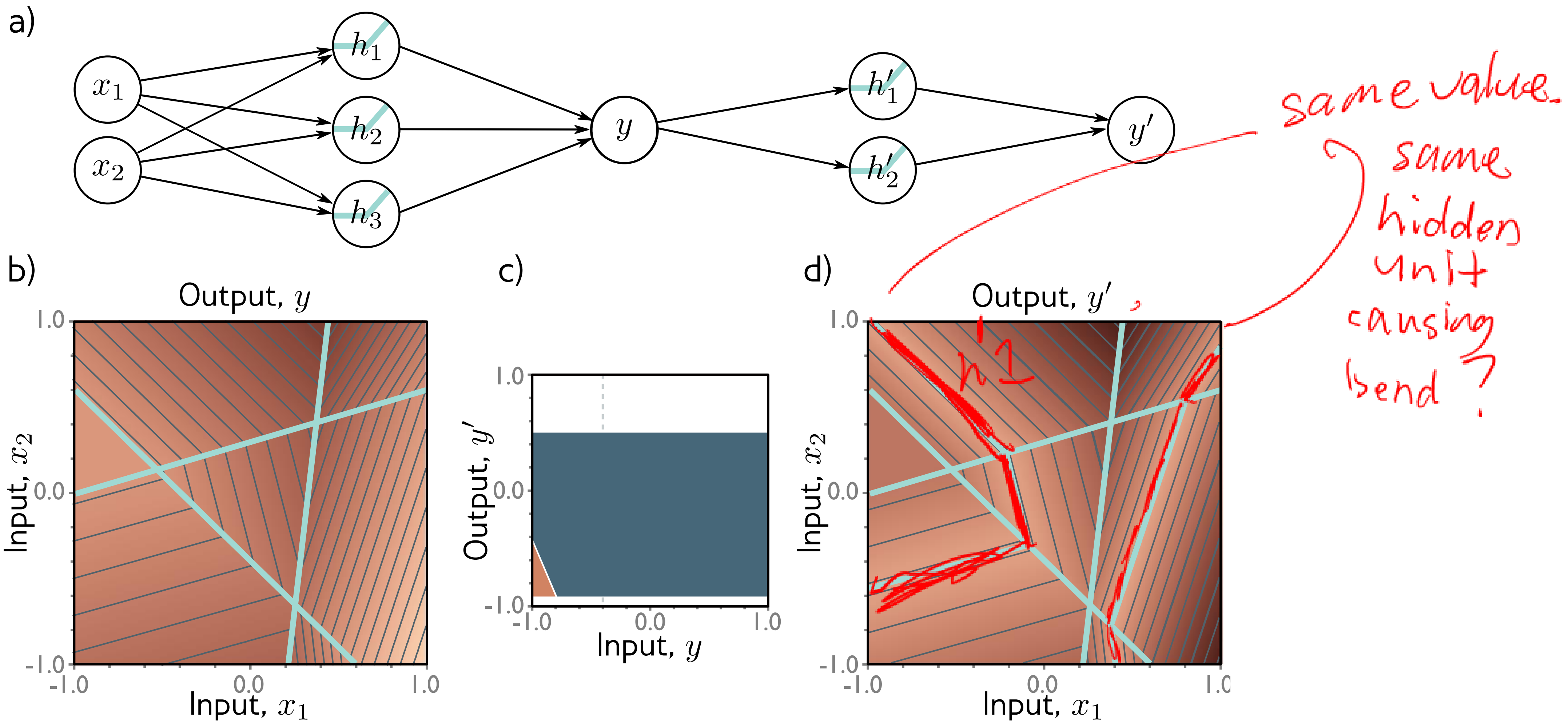


- 20 parameters
- (at least) 9 regions



- 19 parameters
- Max 7 regions

Composing networks in 2D



Any questions?

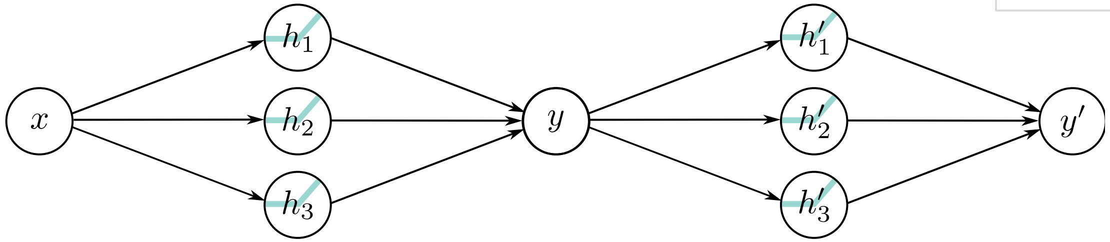
Deep neural networks

- Composing two networks
- Combining the two networks into one
- Hyperparameters
- Notation change and general case
- Shallow vs. deep networks

Combine two networks into one

θ : theta
ϕ : phi

Let's start with 2 networks:



Network 1:
(input is x)

$$h_1 = \text{a}[\theta_{10} + \theta_{11}x]$$
$$h_2 = \text{a}[\theta_{20} + \theta_{21}x]$$
$$h_3 = \text{a}[\theta_{30} + \theta_{31}x]$$

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

linear combination of h_1, h_2, h_3,
no activation function.

Network 2:
(input is y)

$$h'_1 = \text{a}[\theta'_{10} + \theta'_{11}y]$$
$$h'_2 = \text{a}[\theta'_{20} + \theta'_{21}y]$$
$$h'_3 = \text{a}[\theta'_{30} + \theta'_{31}y]$$

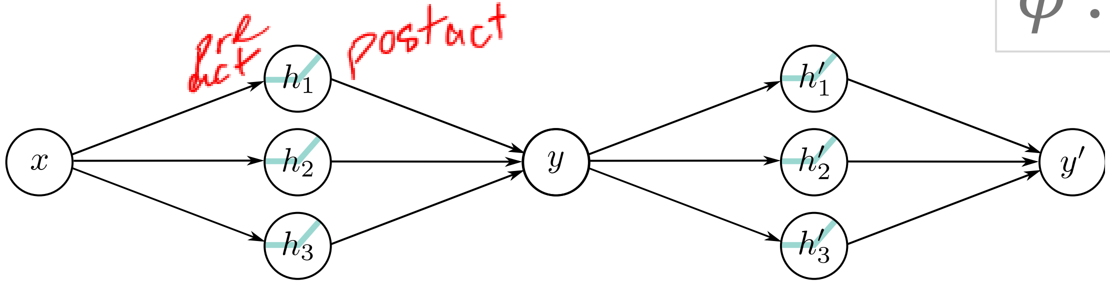
$$y' = \phi'_0 + \phi'_1 h'_1 + \phi'_2 h'_2 + \phi'_3 h'_3$$

2nd hidden layer
preactivations.
linear function of y.
so linear function of previous post activations

Combine two networks into one

θ : theta
ϕ : phi

Let's start with 2 networks:



Network 1:
(input is x)

$$h_1 = \mathrm{a}[\theta_{10} + \theta_{11}x]$$
$$h_2 = \mathrm{a}[\theta_{20} + \theta_{21}x]$$
$$h_3 = \mathrm{a}[\theta_{30} + \theta_{31}x]$$

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

linear

Linear

Network 2:
(input is y)

$$h'_1 = \mathrm{a}[\theta'_{10} + \theta'_{11}y]$$
$$h'_2 = \mathrm{a}[\theta'_{20} + \theta'_{21}y]$$
$$h'_3 = \mathrm{a}[\theta'_{30} + \theta'_{31}y]$$

$$y' = \phi'_0 + \phi'_1 h'_1 + \phi'_2 h'_2 + \phi'_3 h'_3$$

constants when parameters are fixed

Substitute for y to get hidden units of second network in terms of first:

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

ad bd cd
ae be ce
af bf ??

$cf = bf \cdot \frac{ce}{be}$

θ : theta
ϕ : phi
ψ : psi

Create new variables: ψ (psi)

Hidden units of 2nd network in terms of hidden units of first network.

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

Collect and rename the variables for conciseness.

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

Now these new coefficients are not explicitly connected

We get a two-layer network

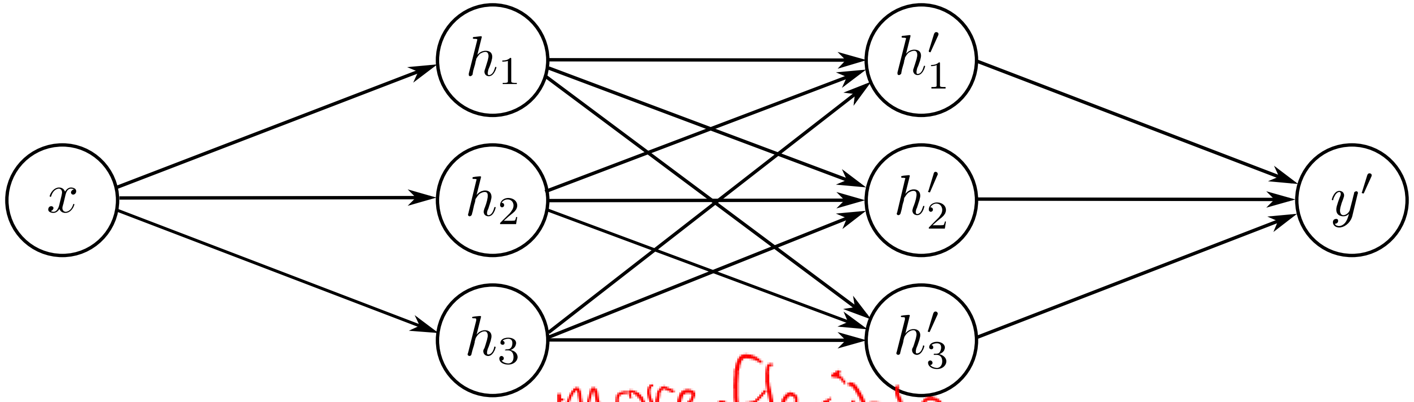
θ : theta
ϕ : phi
ψ : psi

$$h_1 = \mathrm{a}[\theta_{10} + \theta_{11}x]$$
$$h_2 = \mathrm{a}[\theta_{20} + \theta_{21}x]$$
$$h_3 = \mathrm{a}[\theta_{30} + \theta_{31}x]$$

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

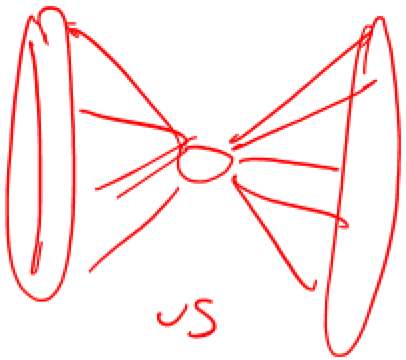
$$y' = \phi'_0 + \phi'_1 h'_1 + \phi'_2 h'_2 + \phi'_3 h'_3$$

y is gonenow.



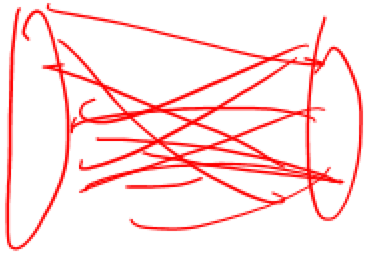
more flexible.

linear



vs

quadratic



Two-layer network as one equation

θ : theta
ϕ : phi
ψ : psi

$$h_1 = \mathrm{a}[\theta_{10} + \theta_{11}x]$$
$$h_2 = \mathrm{a}[\theta_{20} + \theta_{21}x]$$
$$h_3 = \mathrm{a}[\theta_{30} + \theta_{31}x]$$

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

$$y' = \phi'_0 + \phi'_1 h'_1 + \phi'_2 h'_2 + \phi'_3 h'_3$$

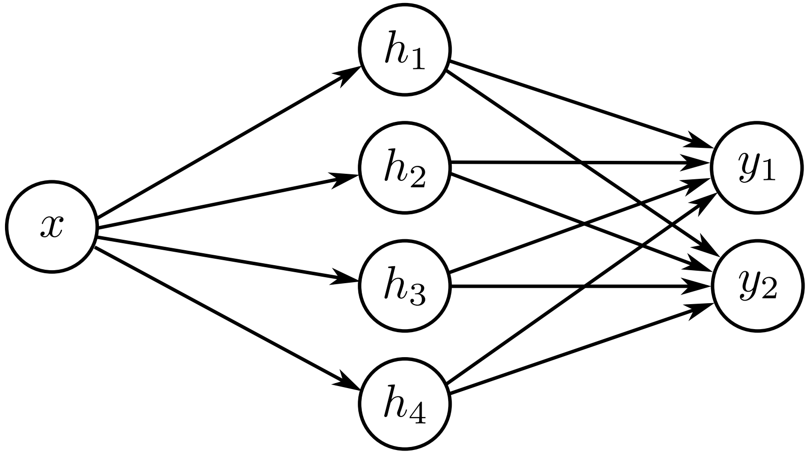
$$\begin{aligned} y' = \phi'_0 &+ \phi'_1 \mathrm{a}\left[\psi_{10} + \psi_{11}\mathrm{a}[\theta_{10} + \theta_{11}x] + \psi_{12}\mathrm{a}[\theta_{20} + \theta_{21}x] + \psi_{13}\mathrm{a}[\theta_{30} + \theta_{31}x]\right] \\ &+ \phi'_2 \mathrm{a}[\psi_{20} + \psi_{21}\mathrm{a}[\theta_{10} + \theta_{11}x] + \psi_{22}\mathrm{a}[\theta_{20} + \theta_{21}x] + \psi_{23}\mathrm{a}[\theta_{30} + \theta_{31}x]] \\ &+ \phi'_3 \mathrm{a}[\psi_{30} + \psi_{31}\mathrm{a}[\theta_{10} + \theta_{11}x] + \psi_{32}\mathrm{a}[\theta_{20} + \theta_{21}x] + \psi_{33}\mathrm{a}[\theta_{30} + \theta_{31}x]] \end{aligned}$$

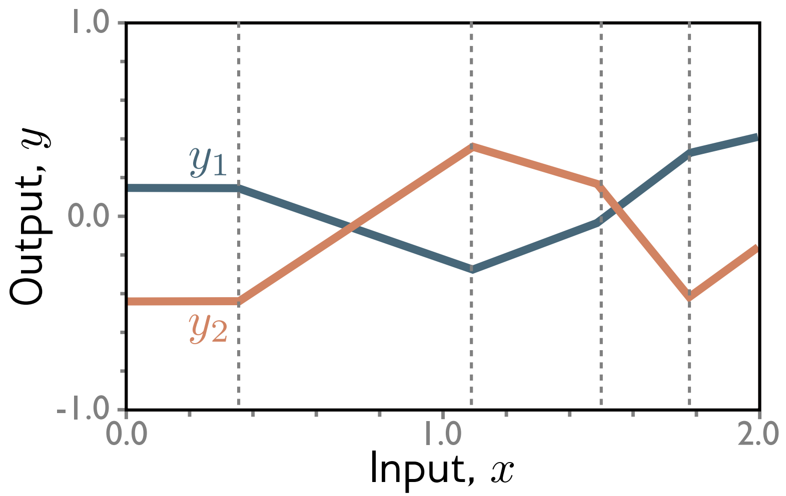
Remember shallow network with two outputs?

- 1 input, 4 hidden units, 2 outputs

$$
\begin{aligned}
h_1 &= \mathrm{a}[\theta_{10} + \theta_{11}x] \\
h_2 &= \mathrm{a}[\theta_{20} + \theta_{21}x] \\
h_3 &= \mathrm{a}[\theta_{30} + \theta_{31}x] \\
h_4 &= \mathrm{a}[\theta_{40} + \theta_{41}x]
\end{aligned}
$$

$$
\begin{aligned}
y_1 &= \phi_{10} + \phi_{11}h_1 + \phi_{12}h_2 + \phi_{13}h_3 + \phi_{14}h_4 \\
y_2 &= \phi_{20} + \phi_{21}h_1 + \phi_{22}h_2 + \phi_{23}h_3 + \phi_{24}h_4
\end{aligned}
$$



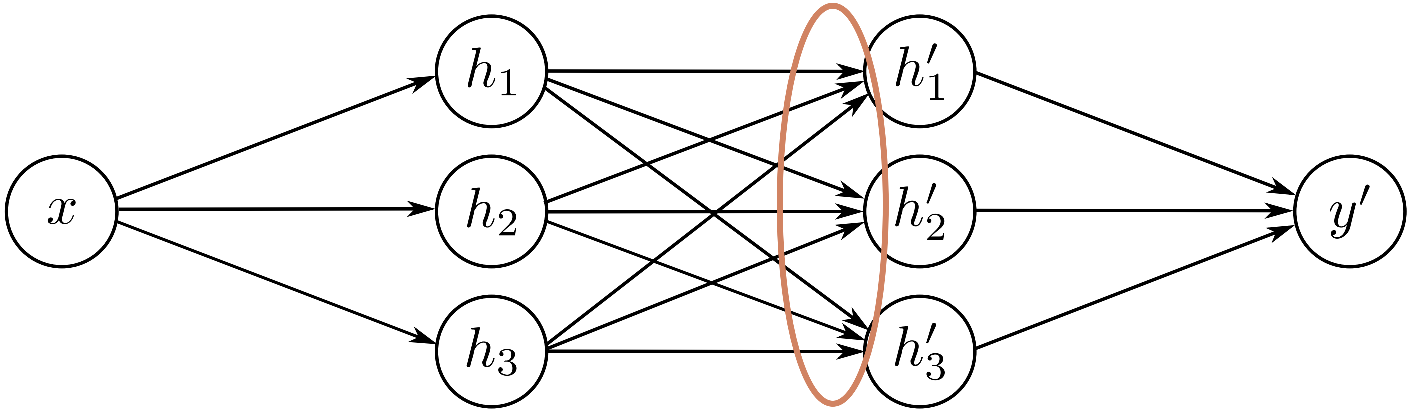


Networks as composing functions

$$h_1 = \mathrm{a}[\theta_{10} + \theta_{11}x]$$
$$h_2 = \mathrm{a}[\theta_{20} + \theta_{21}x]$$
$$h_3 = \mathrm{a}[\theta_{30} + \theta_{31}x]$$

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

Consider the pre-activations at the second hidden units
At this point, it's a one--layer network with three outputs

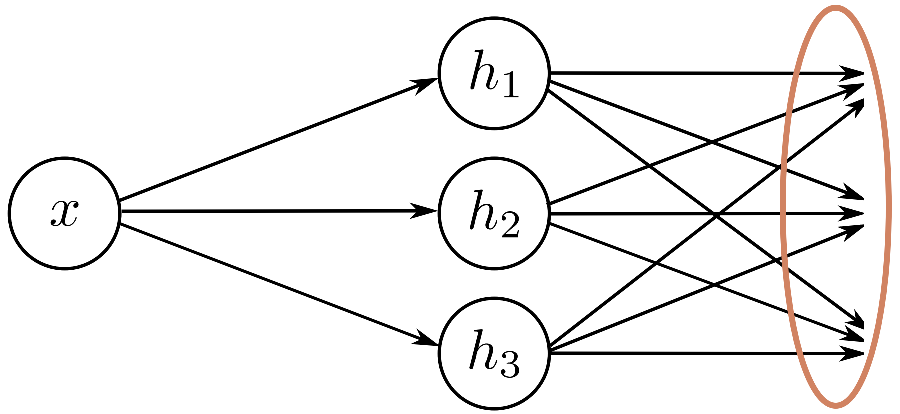


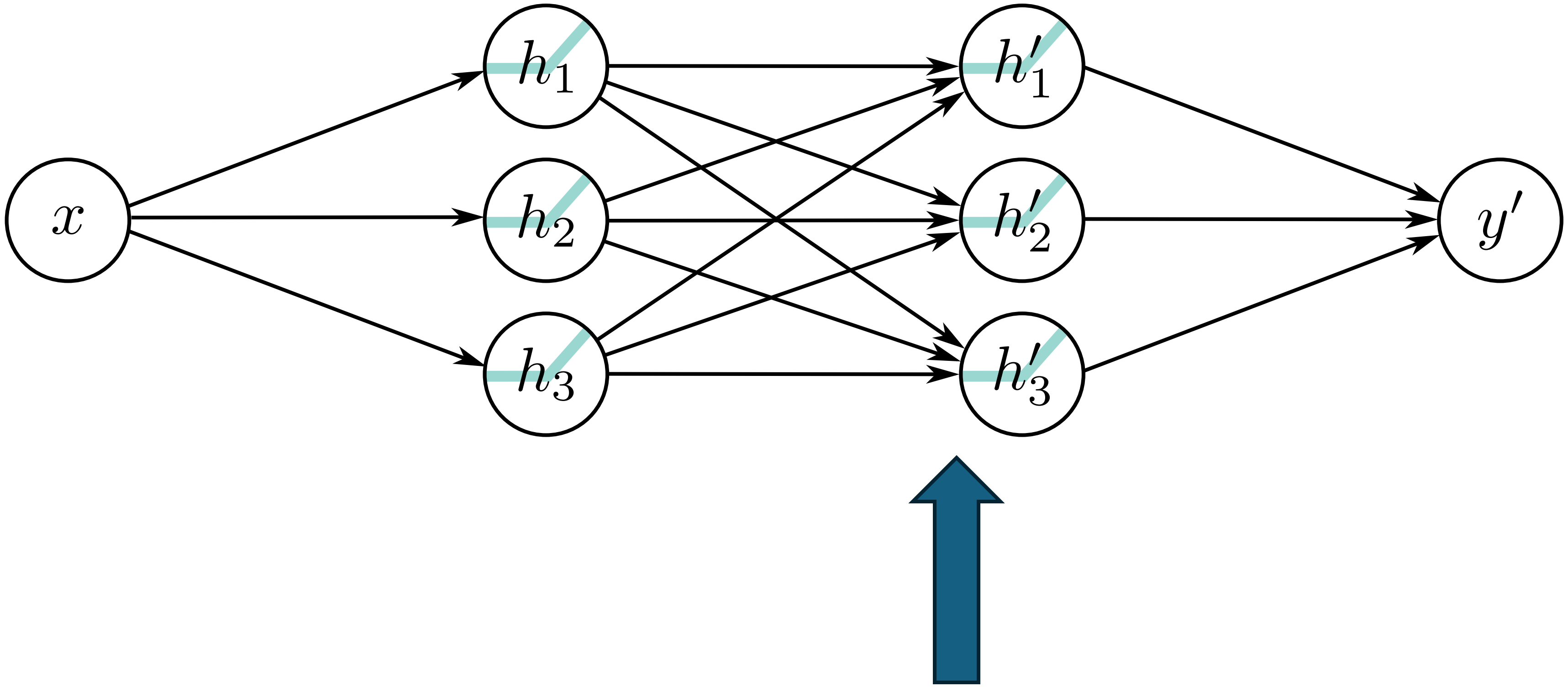
Networks as composing functions

$$h_1 = \mathrm{a}[\theta_{10} + \theta_{11}x]$$
$$h_2 = \mathrm{a}[\theta_{20} + \theta_{21}x]$$
$$h_3 = \mathrm{a}[\theta_{30} + \theta_{31}x]$$

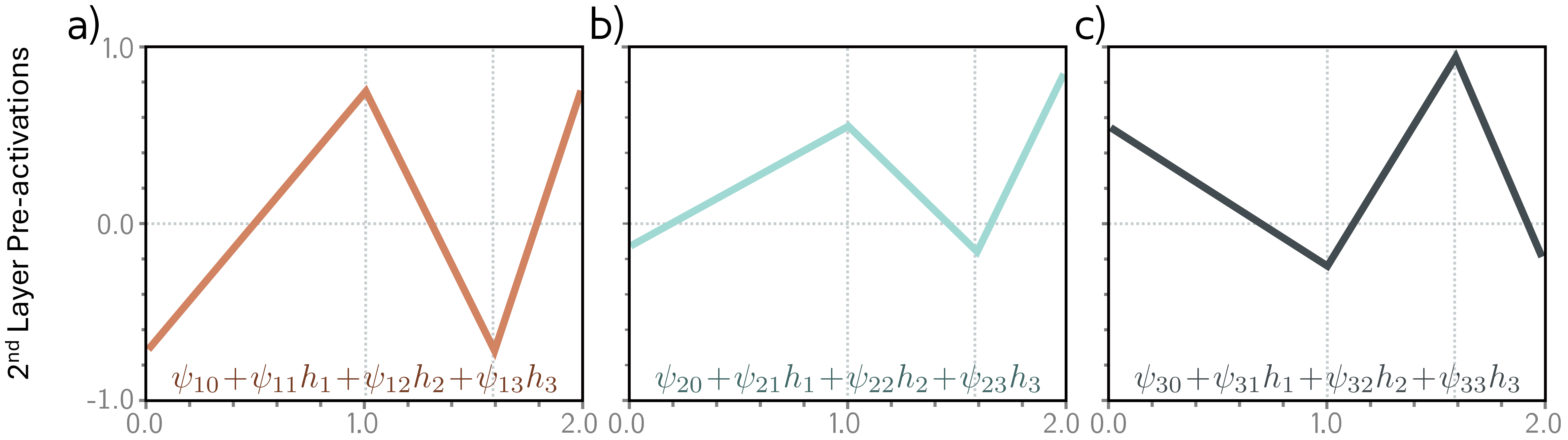
$$h'_1 = \mathrm{a}[\psi_{10} + \psi_{11}h_1 + \psi_{12}h_2 + \psi_{13}h_3]$$
$$h'_2 = \mathrm{a}[\psi_{20} + \psi_{21}h_1 + \psi_{22}h_2 + \psi_{23}h_3]$$
$$h'_3 = \mathrm{a}[\psi_{30} + \psi_{31}h_1 + \psi_{32}h_2 + \psi_{33}h_3]$$

Consider the pre-activations at the second hidden units
At this point, it's a one--layer network with three outputs

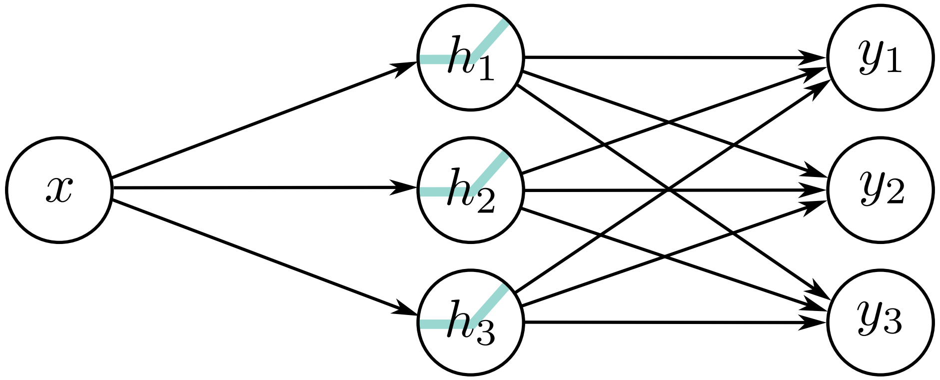


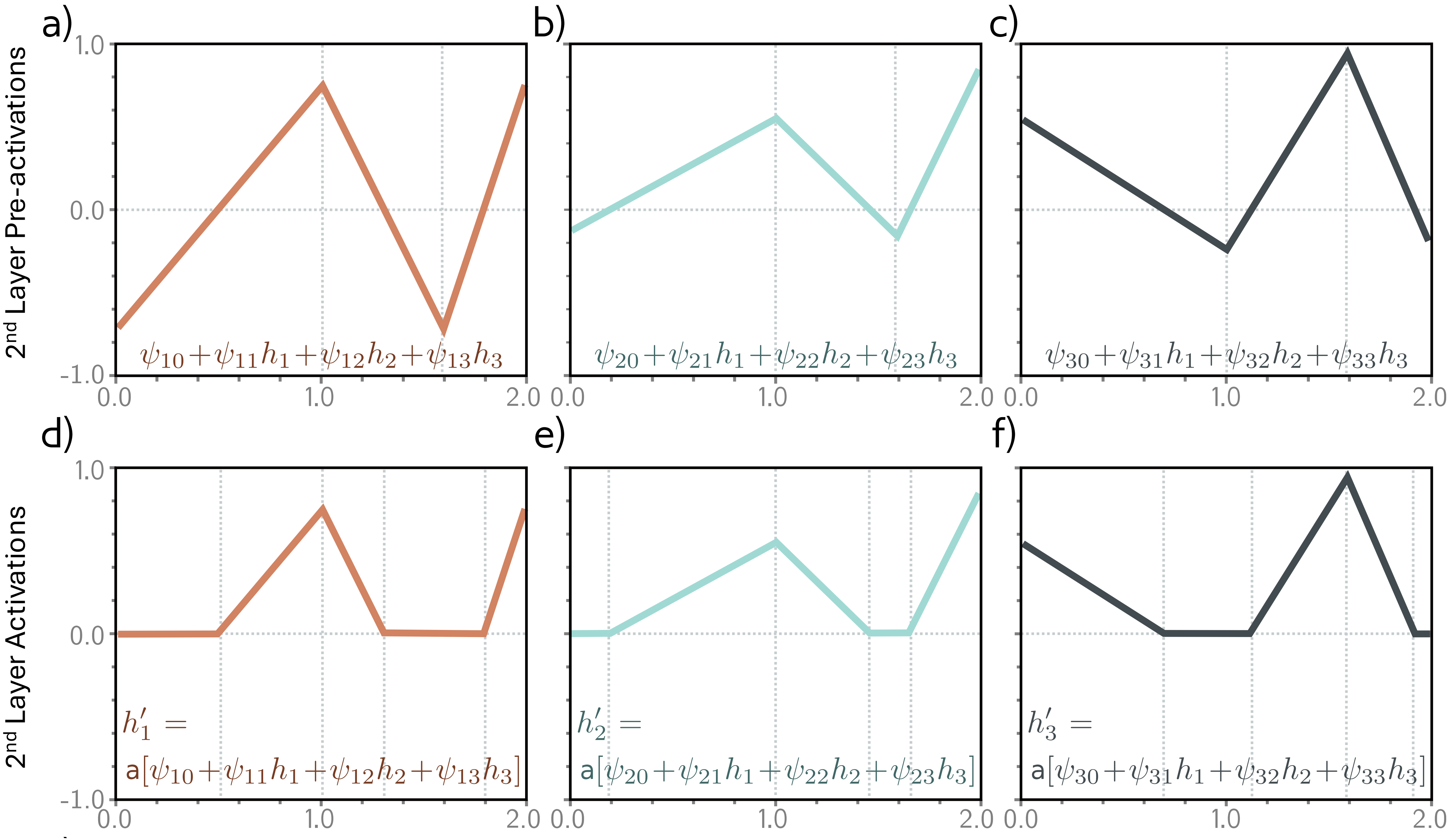


Let's walk through example activations starting with pre-activations to the 2nd layer.

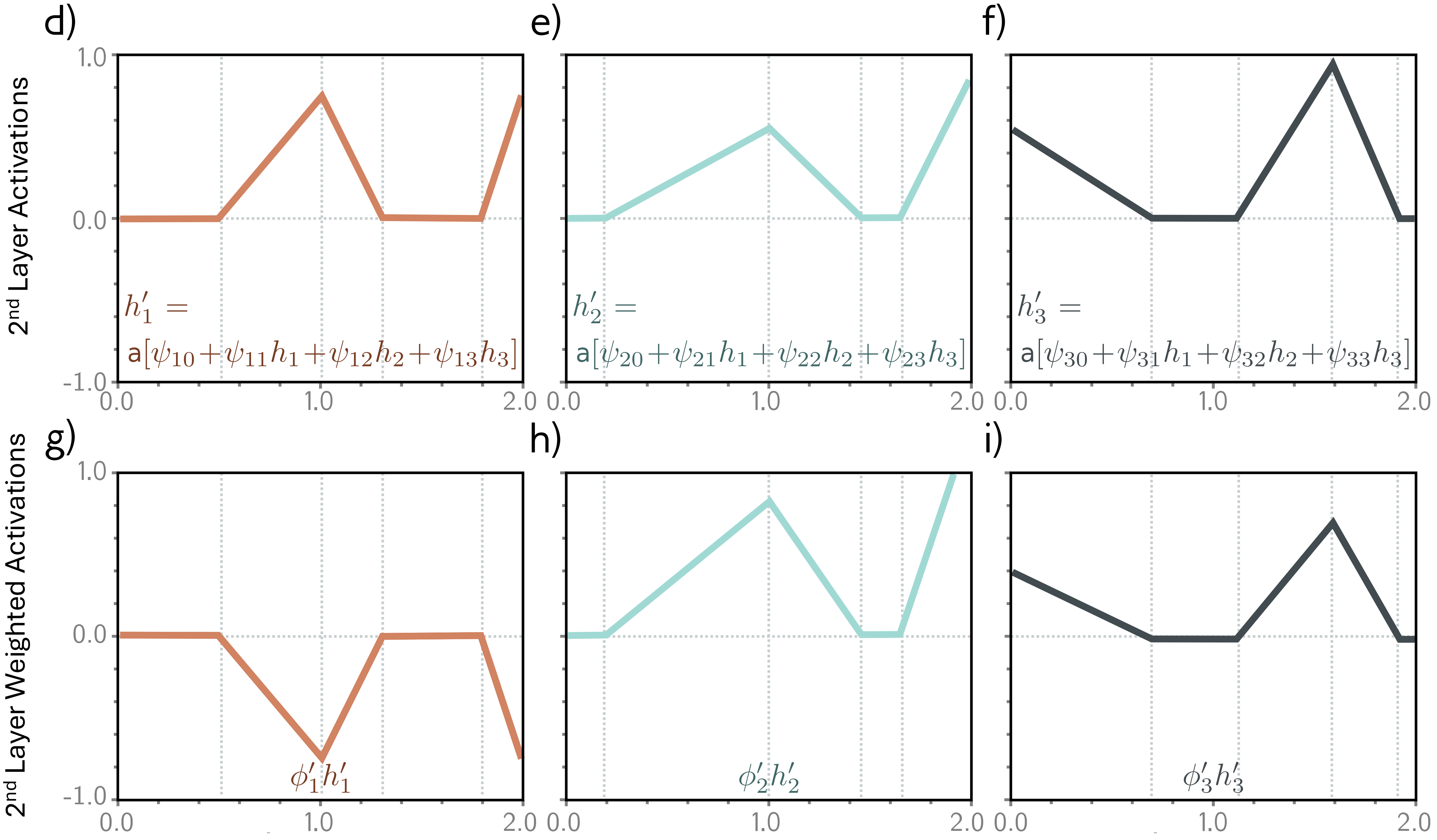


Like a shallow network with three hidden units and three outputs.

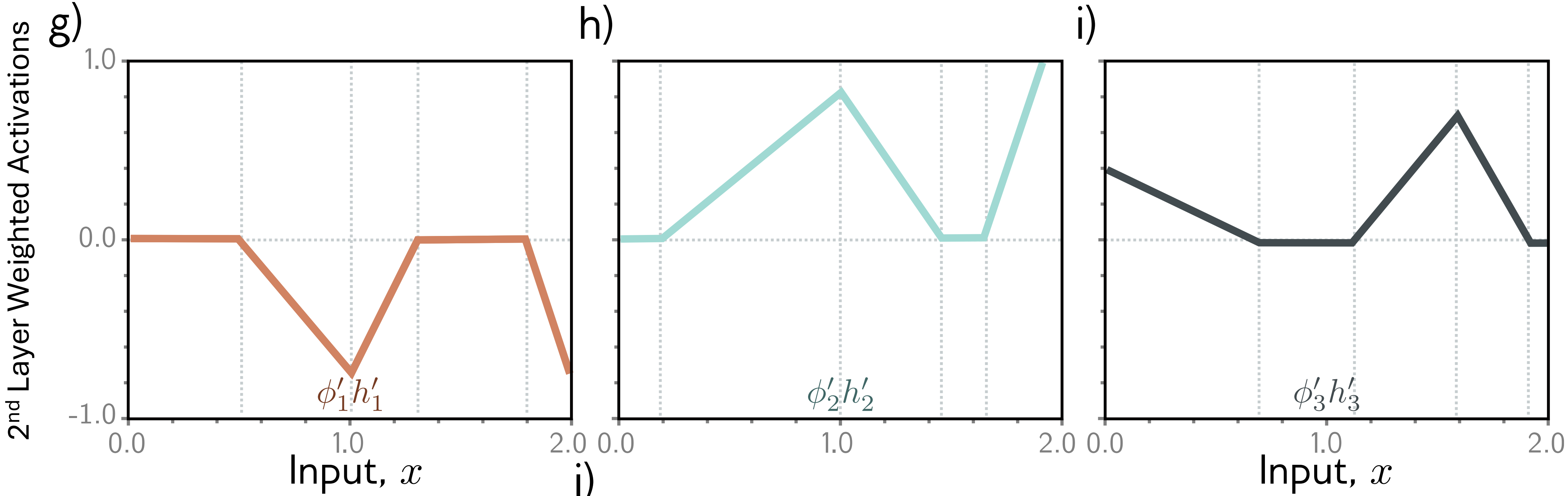


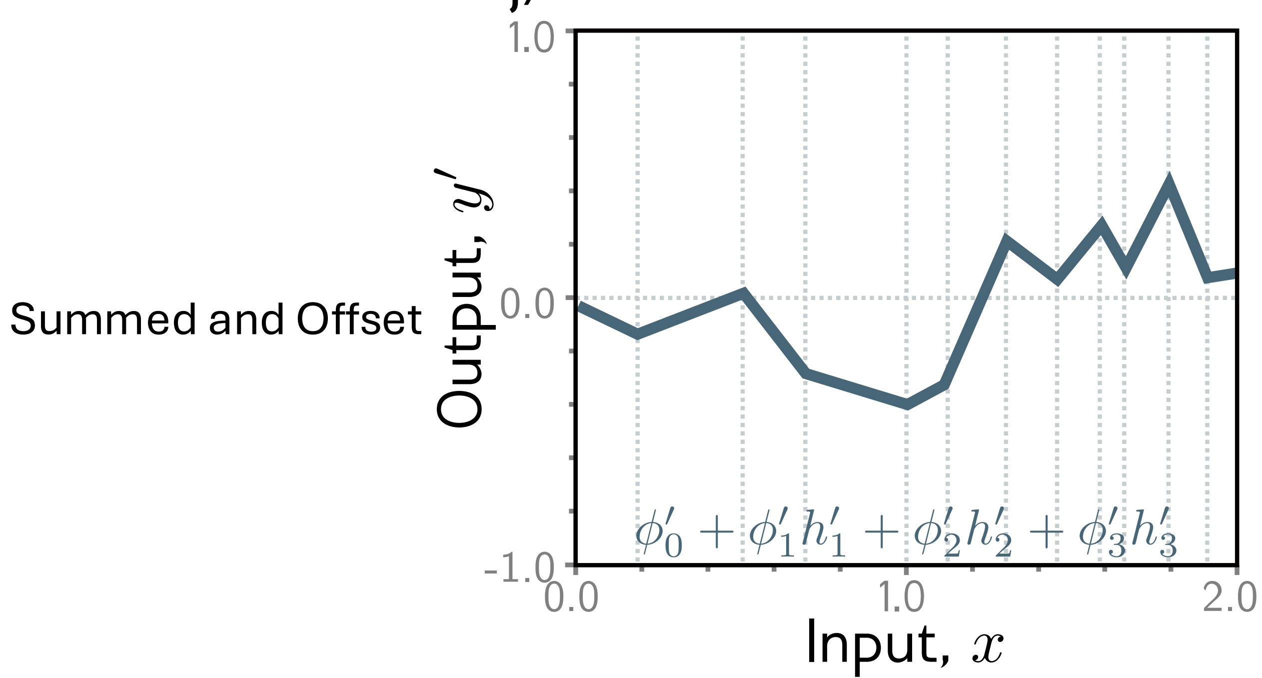


a)
b)
c)
d)
e)
f)
2nd Layer Pre-activations
2nd Layer Activations
1.0
0.0
-1.0
0.0
1.0
2.0
$\psi_{10}+\psi_{11}h_1+\psi_{12}h_2+\psi_{13}h_3$
$\psi_{20}+\psi_{21}h_1+\psi_{22}h_2+\psi_{23}h_3$
$\psi_{30}+\psi_{31}h_1+\psi_{32}h_2+\psi_{33}h_3$
$h'_1 =$
$\mathrm{a}[\psi_{10}+\psi_{11}h_1+\psi_{12}h_2+\psi_{13}h_3]$
$h'_2 =$
$\mathrm{a}[\psi_{20}+\psi_{21}h_1+\psi_{22}h_2+\psi_{23}h_3]$
$h'_3 =$
$\mathrm{a}[\psi_{30}+\psi_{31}h_1+\psi_{32}h_2+\psi_{33}h_3]$



d)
e)
f)
2nd Layer Activations
$h'_1 =$
$\mathrm{a}[\psi_{10}+\psi_{11}h_1+\psi_{12}h_2+\psi_{13}h_3]$
$h'_2 =$
$\mathrm{a}[\psi_{20}+\psi_{21}h_1+\psi_{22}h_2+\psi_{23}h_3]$
$h'_3 =$
$\mathrm{a}[\psi_{30}+\psi_{31}h_1+\psi_{32}h_2+\psi_{33}h_3]$
1.0
0.0
-1.0
0.0
1.0
2.0
g)
h)
i)
2nd Layer Weighted Activations
$\phi'_1 h'_1$
$\phi'_2 h'_2$
$\phi'_3 h'_3$





2nd Analogy:
Create new functions which are clipped and recombined.

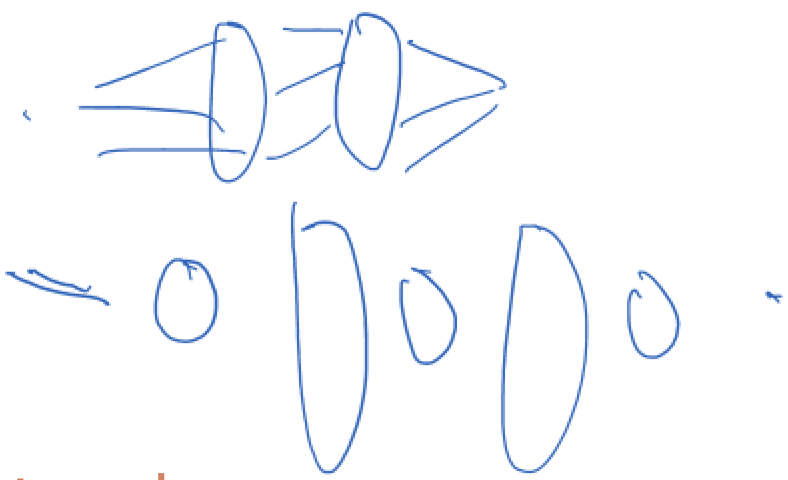
Any questions?

Deep neural networks

- Composing two networks
- Combining the two networks into one
- Hyperparameters
- Notation change and general case
- Shallow vs. deep networks

Hyperparameters

not gradient descent friendly



- K layers = depth of network
- D_k hidden units per layer = width of network

usually powers of 2

- These are called hyperparameters – chosen before training the network
- Can try retraining with different hyperparameters – hyperparameter optimization or hyperparameter search
  - This can be either manual or automated (e.g. Hyperparameter Tuning with Ray Tune)

Any questions?

Deep neural networks

- Composing two networks
- Combining the two networks into one
- Hyperparameters
- Notation change and general case
- Shallow vs. deep networks

Propose 3 notation changes to be able to generalize to arbitrary deep neural networks.

Notation change #1

$$h_1 = \mathrm{a}[\theta_{10} + \theta_{11}x]$$
$$h_2 = \mathrm{a}[\theta_{20} + \theta_{21}x]$$
$$h_3 = \mathrm{a}[\theta_{30} + \theta_{31}x]$$

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

$$y' = \phi'_0 + \phi'_1 h'_1 + \phi'_2 h'_2 + \phi'_3 h'_3$$

Notation change #1

Vector Notation

$$h_1 = \mathrm{a}[\theta_{10} + \theta_{11}x]$$
$$h_2 = \mathrm{a}[\theta_{20} + \theta_{21}x]$$
$$h_3 = \mathrm{a}[\theta_{30} + \theta_{31}x]$$

$$\longrightarrow$$

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

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

$$y' = \phi'_0 + \phi'_1 h'_1 + \phi'_2 h'_2 + \phi'_3 h'_3$$

Notation change #1

Vector Notation

$$h_1 = \mathrm{a}[\theta_{10} + \theta_{11}x]$$
$$h_2 = \mathrm{a}[\theta_{20} + \theta_{21}x]$$
$$h_3 = \mathrm{a}[\theta_{30} + \theta_{31}x]$$

$$\longrightarrow$$

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

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

$$\longrightarrow$$

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

Vector & Matrix Notation

Notation change #1

GPUs are good at repeating the same operations many times over different pieces of data.

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

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

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

Notation change #2

<u>Notation Reminder</u>
x, ψ : normal lower case -- scalar
$\boldsymbol{x}, \boldsymbol{\psi}$: bold face lower case -- vector
$\boldsymbol{X}, \boldsymbol{\Psi}$: bold face upper case -- matrix

$$\begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix} = \mathbf{a}\left[\begin{bmatrix} \theta_{10} \\ \theta_{20} \\ \theta_{30} \end{bmatrix} + \begin{bmatrix} \theta_{11} \\ \theta_{21} \\ \theta_{31} \end{bmatrix} x\right] \longrightarrow \mathbf{h} = \mathbf{a}[\boldsymbol{\theta}_0 + \boldsymbol{\theta}_1 x]$$

vector

$$\begin{bmatrix} h'_1 \\ h'_2 \\ h'_3 \end{bmatrix} = \mathbf{a}\left[\begin{bmatrix} \psi_{10} \\ \psi_{20} \\ \psi_{30} \end{bmatrix} + \begin{bmatrix} \psi_{11} & \psi_{12} & \psi_{13} \\ \psi_{21} & \psi_{22} & \psi_{23} \\ \psi_{32} & \psi_{32} & \psi_{33} \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix}\right] \longrightarrow \mathbf{h}' = \mathbf{a}[\boldsymbol{\psi}_0 + \boldsymbol{\Psi}\mathbf{h}]$$

$$y' = \phi'_0 + \begin{bmatrix} \phi'_1 & \phi'_2 & \phi'_3 \end{bmatrix} \begin{bmatrix} h'_1 \\ h'_2 \\ h'_3 \end{bmatrix} \longrightarrow y' = \phi'_0 + \boldsymbol{\phi}'^T \mathbf{h}'$$

ω : omega
Ω : Omega

Notation change #3

1

k

$$\mathbf{h} = \mathbf{a}\left[\boldsymbol{\theta}_0 + \boldsymbol{\theta} x\right] \longrightarrow \mathbf{h}_1 = \mathbf{a}[\boldsymbol{\beta}_0 + \boldsymbol{\Omega}_0 \mathbf{x}]$$

matrix

$$\mathbf{h}' = \mathbf{a}\left[\boldsymbol{\psi}_0 + \boldsymbol{\Psi} \mathbf{h}\right] \longrightarrow \mathbf{h}_2 = \mathbf{a}[\boldsymbol{\beta}_1 + \boldsymbol{\Omega}_1 \mathbf{h}_1]$$

$$y = \phi_0' + \boldsymbol{\phi}' \mathbf{h}' \longrightarrow \mathbf{y} = \boldsymbol{\beta}_2 + \boldsymbol{\Omega}_2 \mathbf{h}_2$$

ω : omega
Ω : Omega

Notation change #3

Bias vector

Weight matrix

$$\mathbf{h} = \mathbf{a}\left[\boldsymbol{\theta}_0 + \boldsymbol{\theta}x\right] \longrightarrow \mathbf{h}_1 = \mathbf{a}[\boldsymbol{\beta}_0 + \boldsymbol{\Omega}_0\mathbf{x}]$$

$$\mathbf{h}' = \mathbf{a}\left[\boldsymbol{\psi}_0 + \boldsymbol{\Psi}\mathbf{h}\right] \longrightarrow \mathbf{h}_2 = \mathbf{a}[\boldsymbol{\beta}_1 + \boldsymbol{\Omega}_1\mathbf{h}_1]$$

$$y = \phi'_0 + \boldsymbol{\phi}'\mathbf{h}' \longrightarrow \mathbf{y} = \boldsymbol{\beta}_2 + \boldsymbol{\Omega}_2\mathbf{h}_2$$

General equations for deep network

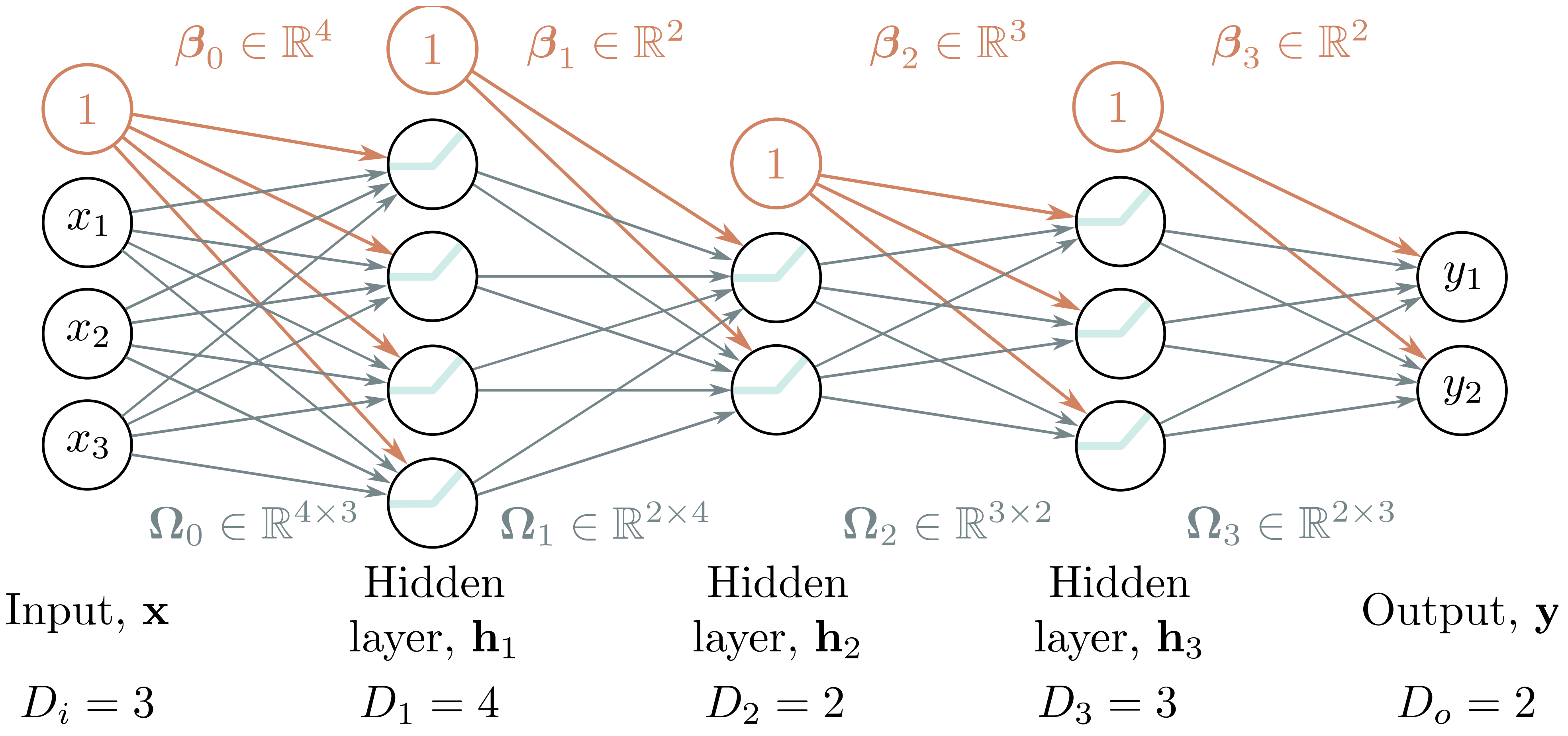
$$\begin{aligned}
\mathbf{h}_1 &= \mathbf{a}[\boldsymbol{\beta}_0 + \boldsymbol{\Omega}_0\mathbf{x}]\\
\mathbf{h}_2 &= \mathbf{a}[\boldsymbol{\beta}_1 + \boldsymbol{\Omega}_1\mathbf{h}_1]\\
\mathbf{h}_3 &= \mathbf{a}[\boldsymbol{\beta}_2 + \boldsymbol{\Omega}_2\mathbf{h}_2]\\
&\vdots\\
\mathbf{h}_K &=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

previous length $\propto$ #parameters

$$\mathbf{y} = \boldsymbol{\beta}_K + \boldsymbol{\Omega}_K\mathbf{a}\left[\boldsymbol{\beta}_{K-1} + \boldsymbol{\Omega}_{K-1}\mathbf{a}\left[\ldots\boldsymbol{\beta}_2 + \boldsymbol{\Omega}_2\mathbf{a}\left[\boldsymbol{\beta}_1 + \boldsymbol{\Omega}_1\mathbf{a}\left[\boldsymbol{\beta}_0 + \boldsymbol{\Omega}_0\mathbf{x}\right]\right]\ldots\right]\right]$$

length of equation $\propto$ #of layers

Example



$\boldsymbol{\beta}_0 \in \mathbb{R}^4$
$\boldsymbol{\beta}_1 \in \mathbb{R}^2$
$\boldsymbol{\beta}_2 \in \mathbb{R}^3$
$\boldsymbol{\beta}_3 \in \mathbb{R}^2$
1
1
1
1
x_1
x_2
x_3
y_1
y_2
$\boldsymbol{\Omega}_0 \in \mathbb{R}^{4\times 3}$
$\boldsymbol{\Omega}_1 \in \mathbb{R}^{2\times 4}$
$\boldsymbol{\Omega}_2 \in \mathbb{R}^{3\times 2}$
$\boldsymbol{\Omega}_3 \in \mathbb{R}^{2\times 3}$
Input, $\mathbf{x}$
Hidden layer, $\mathbf{h}_1$
Hidden layer, $\mathbf{h}_2$
Hidden layer, $\mathbf{h}_3$
Output, $\mathbf{y}$
$D_i = 3$
$D_1 = 4$
$D_2 = 2$
$D_3 = 3$
$D_o = 2$

Any questions?

Deep neural networks

- Composing two networks
- Combining the two networks into one
- Hyperparameters
- Notation change and general case
- Shallow vs. deep networks

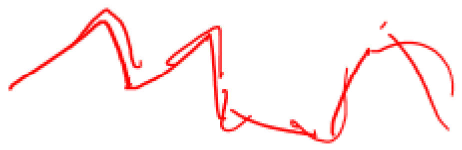
Shallow vs. deep networks

The best results are created by deep networks with many layers.

- 50-1000 layers for most applications
- Best results in
  - Computer vision
  - Natural language processing
  - Graph neural networks
  - Generative models
  - Reinforcement learning

All use deep networks.
But why?

Shallow vs. deep networks



1. Ability to approximate different functions?

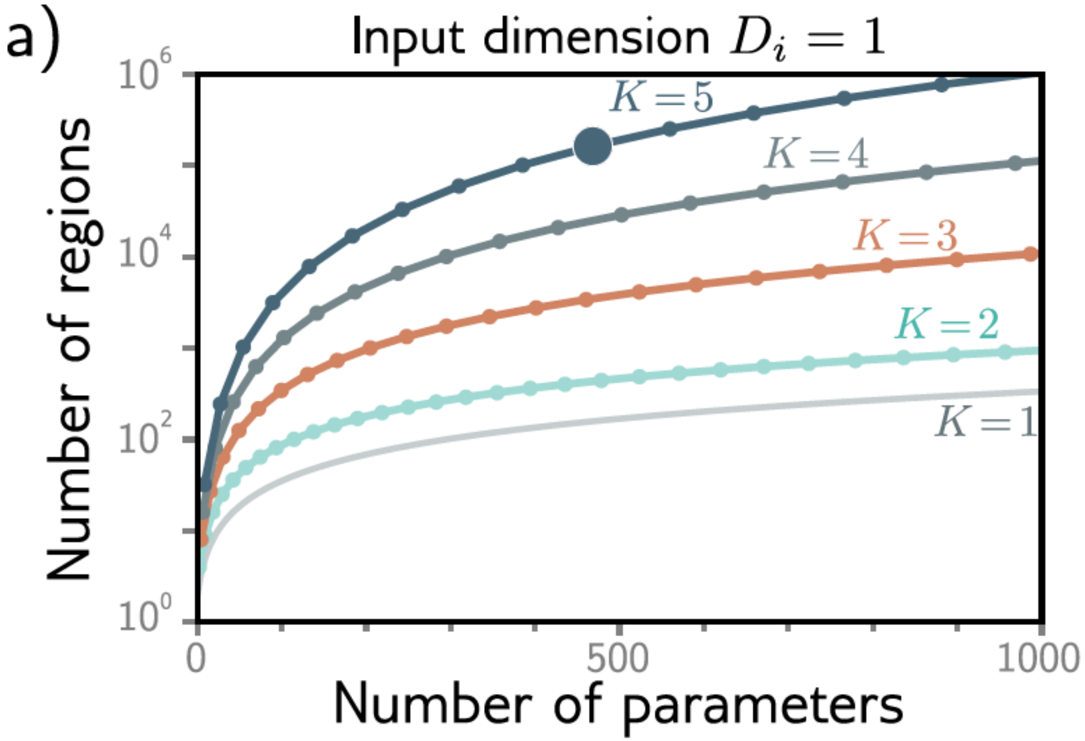
Both obey the universal approximation theorem.

Argument: One layer is enough, and for deep networks could arrange for the other layers to compute the identity function.

Shallow vs. deep networks

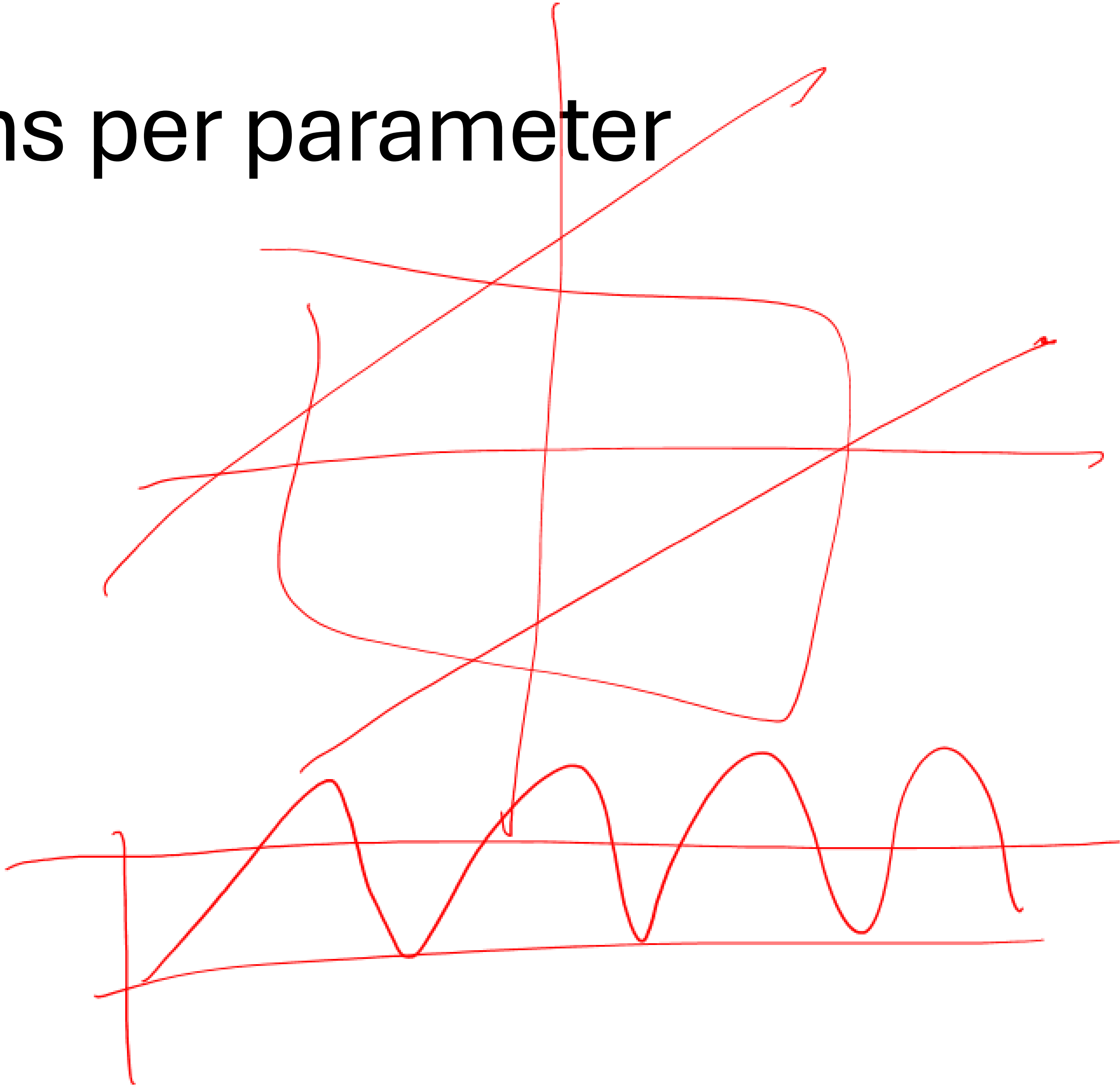
2. Number of linear regions per parameter

Number of linear regions per parameter

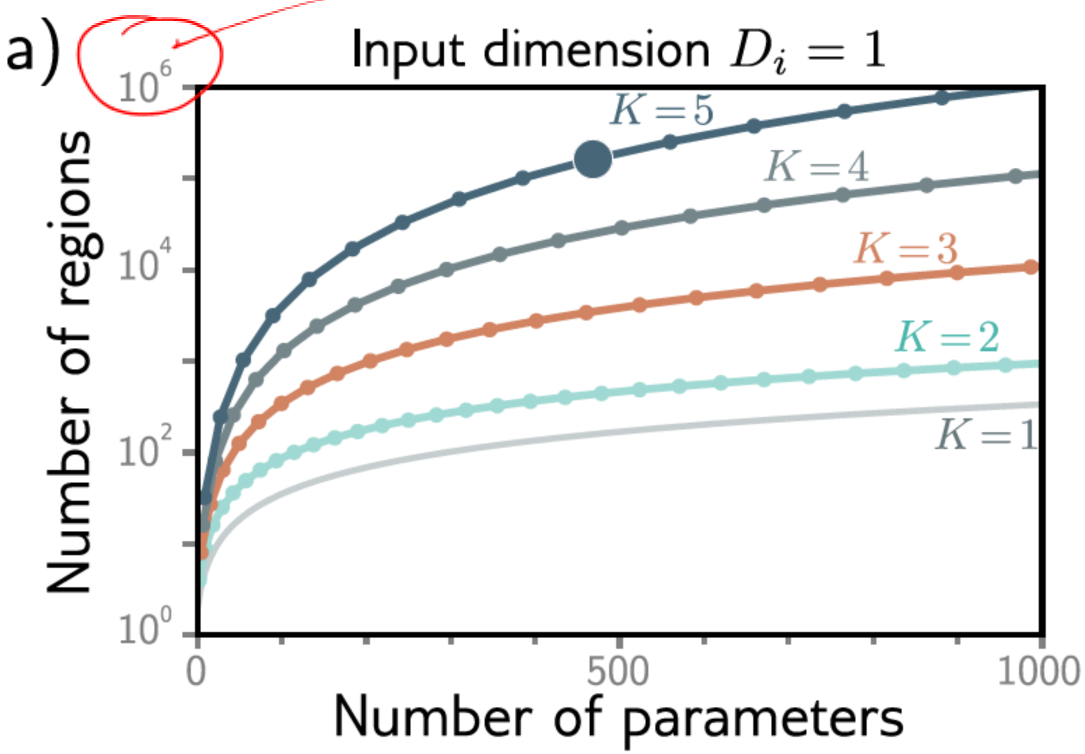


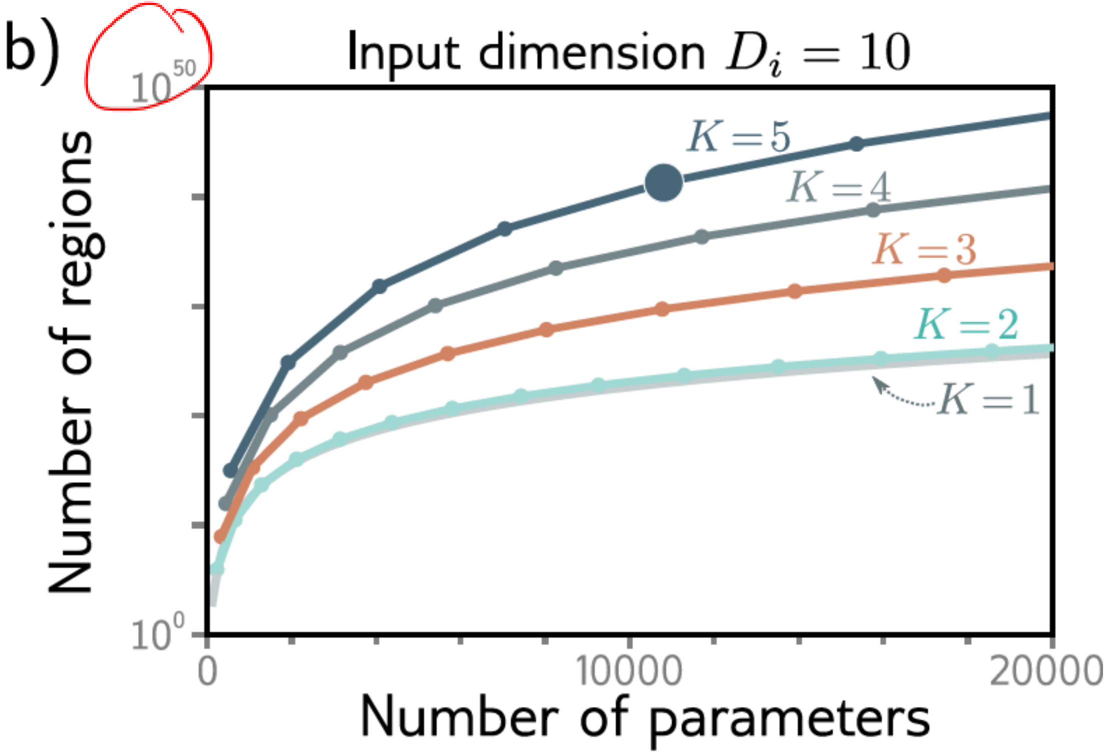
Each small dot is an additional hidden unit per layer.

K = 5 layers
10 hidden units per layer
471 parameters
161,501 linear regions



Number of linear regions per parameter





Each small dot is an additional hidden unit per layer.

● 5 layers
10 hidden units per layer
471 parameters
161,501 linear regions

Each small dot is an additional 10 hidden units per layer.

● 5 layers
50 hidden units per layer
10,801 parameters
$>10^{40}$ linear regions

Shallow vs. deep networks

If modeling 2^x

if form: $e^{\phi x}$ ($\phi = \ln 2$)

if form is

$$\sum_{i=0}^{1000} a_i x^i$$

2. Number of linear regions per parameter

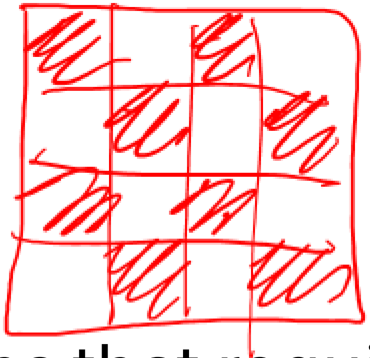
- Deep networks create many more regions per parameters
- But there are dependencies between them
  - Think of folding example
  - Perhaps similar symmetries in real-world functions? Unknown

Does available flexibility (from more regions) match desired flexibility?

Shallow vs. Deep Networks

Parity: is $x_1+x_2+x_3+\dots+x_n$ odd or even?

3. Depth efficiency



1 hidden layer $\rightarrow$ 2^n hidden nodes

$\log_2 n$ layers $\rightarrow$ $\sim n$ hidden nodes

- There are some functions that require a shallow network with exponentially more hidden units than a deep network to achieve an equivalent approximation

- This is known as the depth efficiency of deep networks

- But do the real-world functions we want to approximate have this property? Unknown. Rare if ever.

Shallow vs. Deep Networks

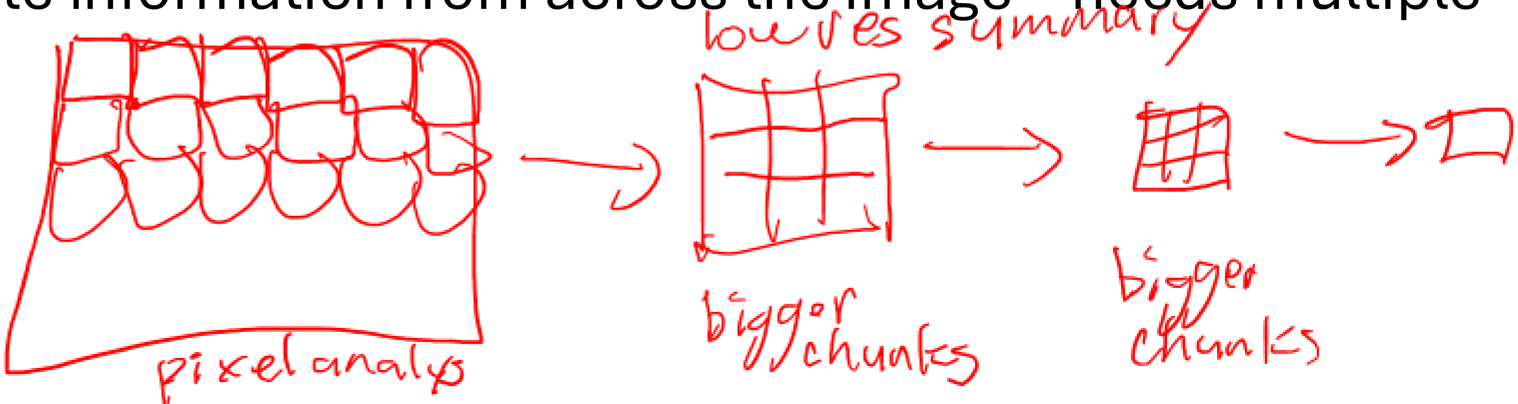
hierarchical processing

same low level processing "everywhere"

rolled up to higher abstraction w/ lower resolution

4. Large structured networks

- Think about images as input – might be 1M pixels
- Fully connected works not practical
- Answer is to have weights that only operate locally, and share across image
- This leads to convolutional networks
- Gradually integrate information from across the image – needs multiple layers

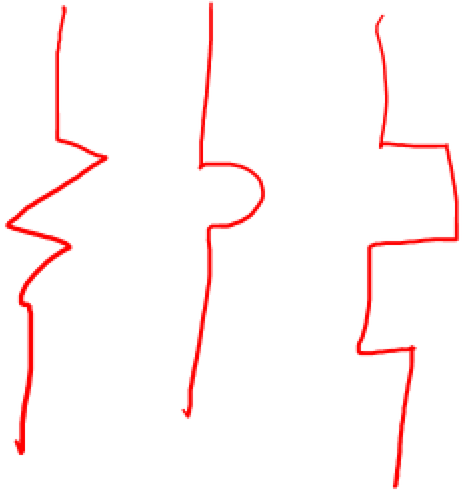


Shallow vs. Deep Networks

5. Fitting and generalization

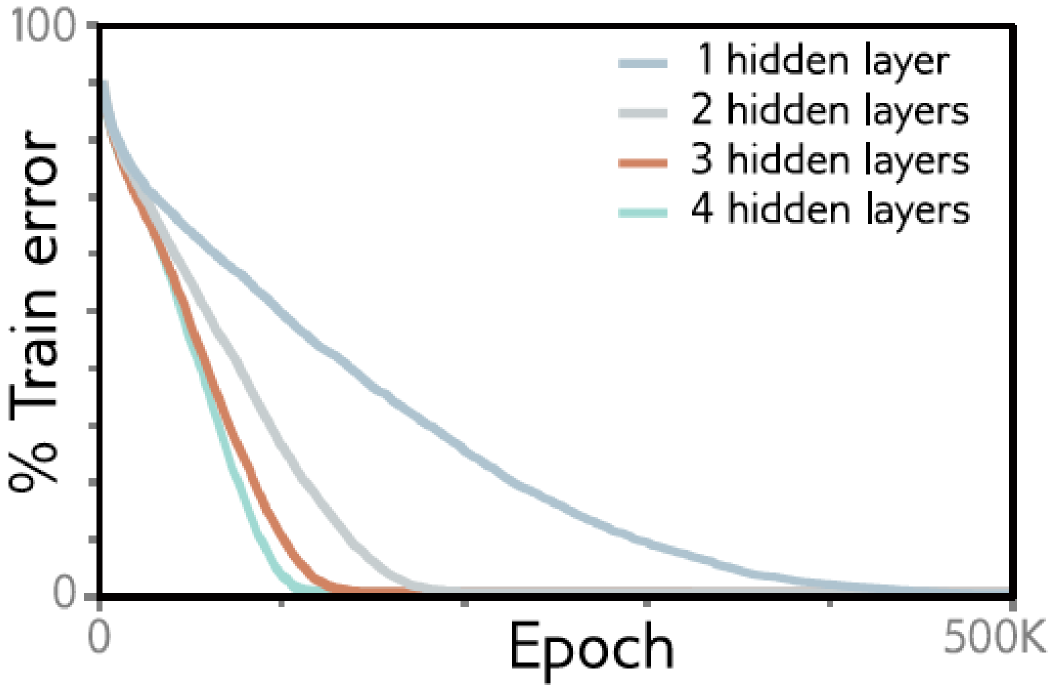
- Fitting of deep models seems to be easier up to about 20 layers
- Then needs various tricks to train deeper networks, so (in vanilla form), fitting becomes harder
- Generalization is good in deep networks. Why?

Shallow vs. Deep Networks



5. Fitting and generalization

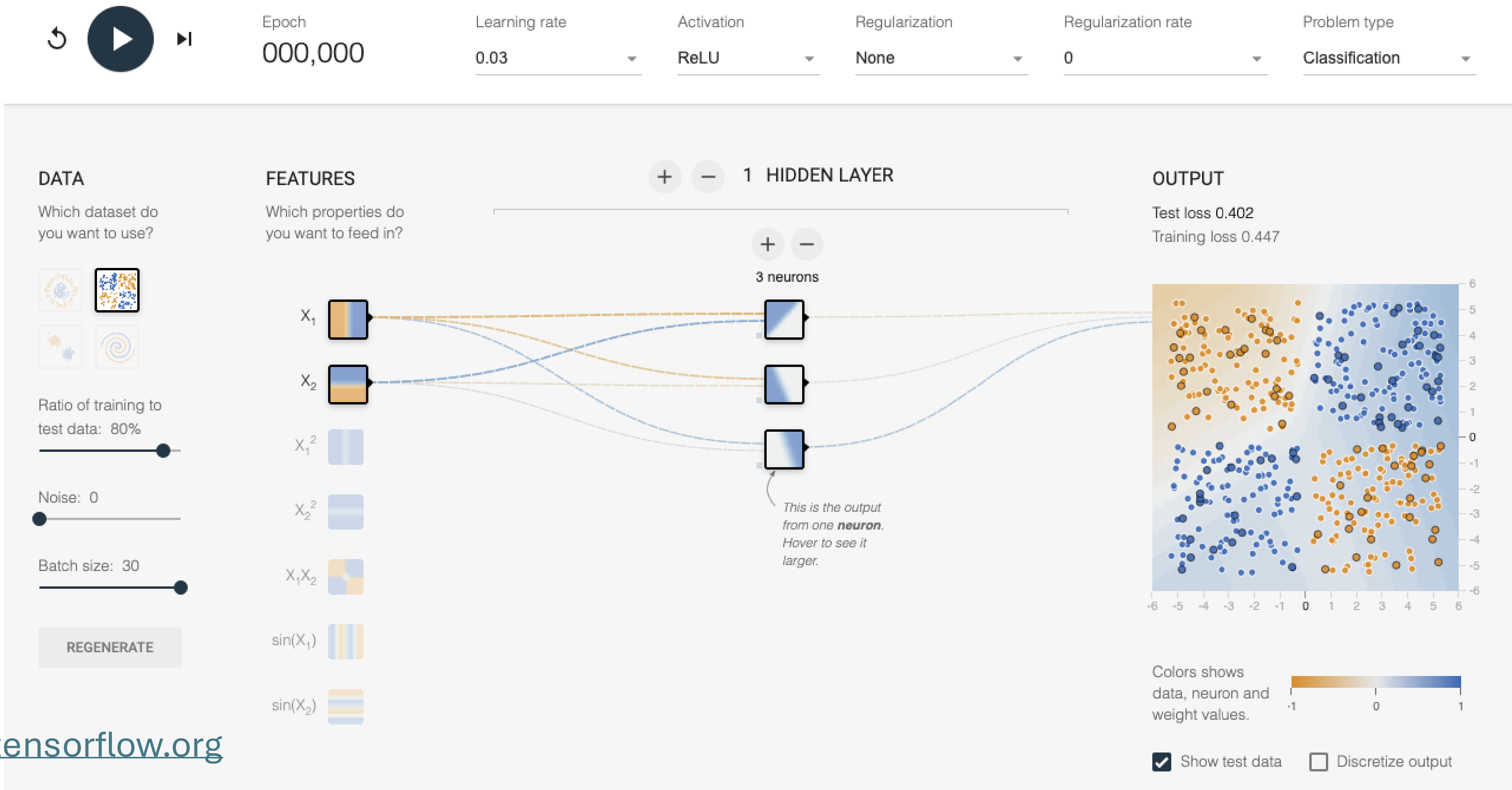
Figure 20.2 MNIST-1D training. Four fully connected networks were fit to 4000 MNIST-1D examples with random labels using full batch gradient descent, He initialization, no momentum or regularization, and learning rate 0.0025. Models with 1,2,3,4 layers had 298, 100, 75, and 63 hidden units per layer and 15208, 15210, 15235, and 15139 parameters, respectively. All models train successfully, but deeper models require fewer epochs.



Tensorflow Playground Example?

- Try 2 inputs, 3 hidden units, 1 output
- You can inspect and/or edit weights and biases

Do you ever get stuck in local minima? Are you getting the expected number of regions?



playground.tensorflow.org

Any questions?

Where are we going?

- We have defined families of very flexible networks that map multiple inputs to multiple outputs
- Now we need to train them
  - How to choose loss functions for different types of targets ✅
  - How to find minima of the loss function ✅
  - How to do this efficiently with deep networks ←
- Then how do we evaluate them?