



Lecture 04

Deep Networks

DL4DS – Spring 2025

DS542 Gardos – [Understanding Deep Learning](#), Other Content Cited

Project Proposal Template

<https://dl4ds.github.io/sp2025/project/#proposal>

Proposal

Deadline: February 16, Friday 11:59 PM

Here's a proposal template [pdf](#) and the source [LaTeX](#).

The proposal format is:

- **Project Title**
- **Abstract**
- **Team Members** (From one to three people.)
- **Introduction:** Give the motivation for the problem you are solving or application you are developing and why it is worthwhile.
- **Related Work:** Results from initial literature search.
- **Proposed Work:** What are you going to do and how are you going to do it?
- **Datasets:** What dataset will you be using? Does it exist already? What dataset preparation will be needed?
- **Evaluation:** How are you going to evaluate your results?
- **Timeline:** Approximate time line for the project over the course of the semester.
- **Conclusion:** You can recap your proposal.
- **References:** References for your citations.

More explanation of each section is in the template.

Submission will be on GradeScope, but feel free to share early draft with the instructor to get early feedback.

If you don't have a [LaTeX](#) authoring environment set up, we recommend the [LaTeX Workshop](#) extension for [Visual Studio Code](#).

DL4DS Project Title

Team Member 1, Optional Team Member 2, Optional Team Member 3

January 26, 2024

Abstract

A short abstract of what your proposal is about. This is your elevator pitch for what this project is about and why it is interesting. This should be no more than 3 sentences. Typically you write this last.

Introduction

This is a template for your project proposal which, by the way, is very similar to the template for your final report. It's also pretty standard template used for conference and journal publications. For the project proposal, however, your context in each section will be shorter and preliminary. In other words it is likely to change for the final report. Remove all these italicized instructions before submitting.

The introduction is where you motivate the problem you are trying to solve. You should give a brief overview of the problem and why it is important. You should also give a brief overview of your approach and why it is interesting.

Introduction text here.

Related Work

This is where you give a brief overview of any prior work by others (or yourself) that is relevant to the problem and solution you are proposing. For the proposal you should do a preliminary literature search and cite any papers using the citation and bibliography syntax illustrated below.

Let me know if you want help finding teammates.

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.

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

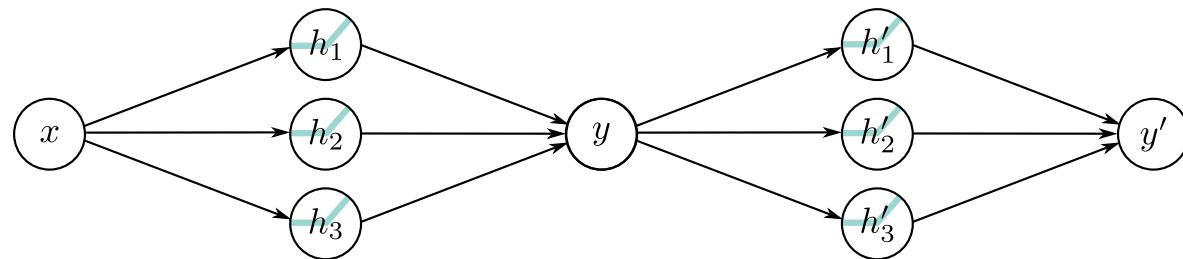
Network 1:

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

$$h'_1 = a[\theta'_{10} + \theta'_{11}y]$$

Network 2:

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

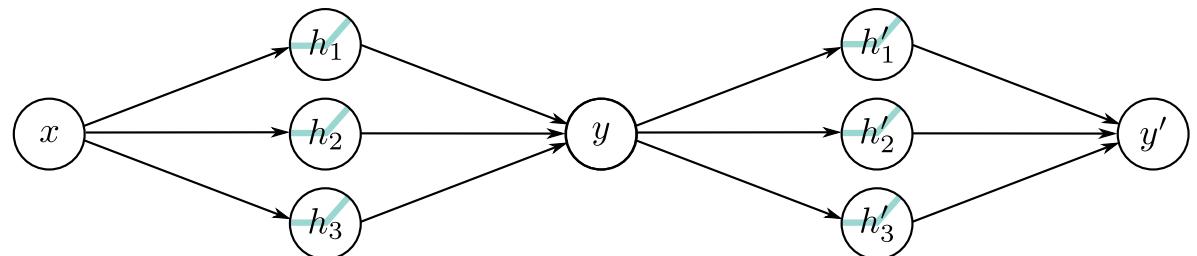
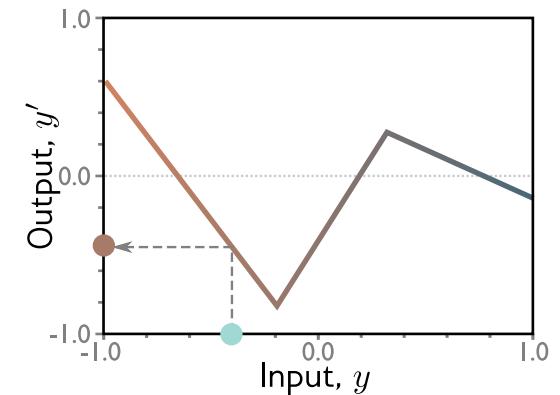
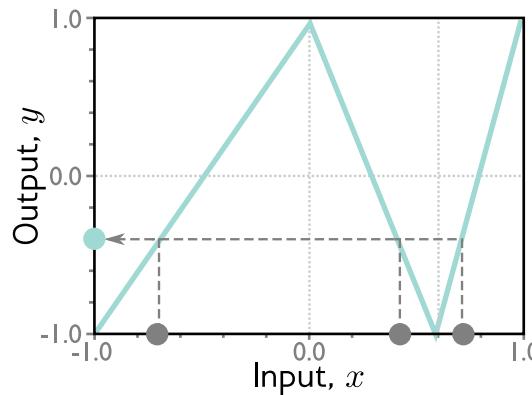


Composing two networks: Example

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



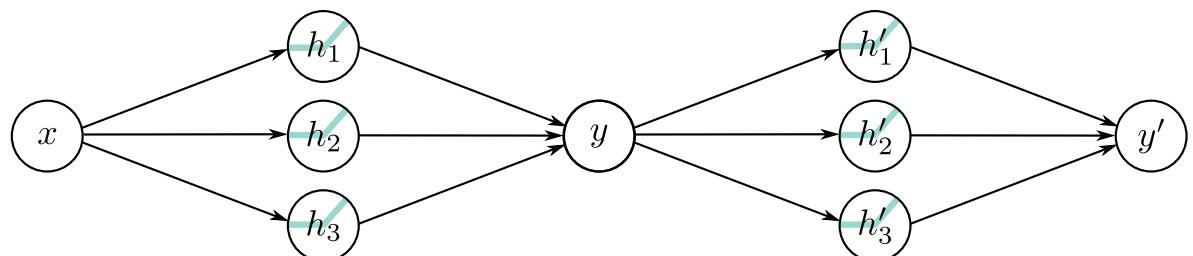
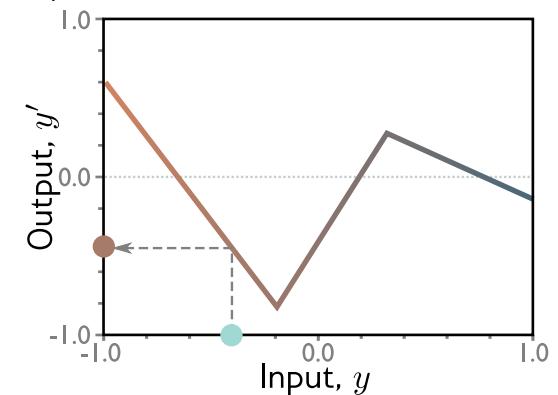
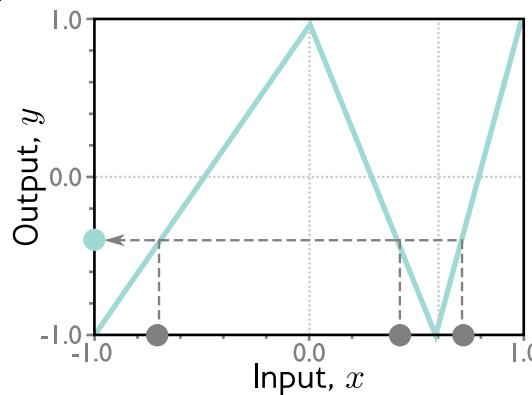
Composing two networks: Example

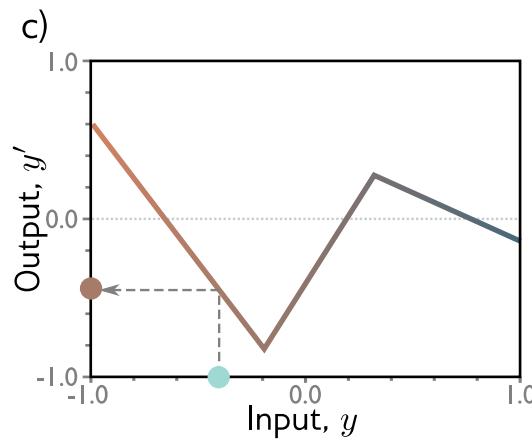
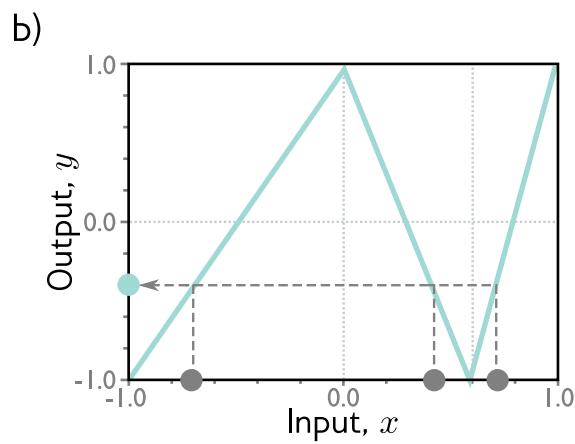
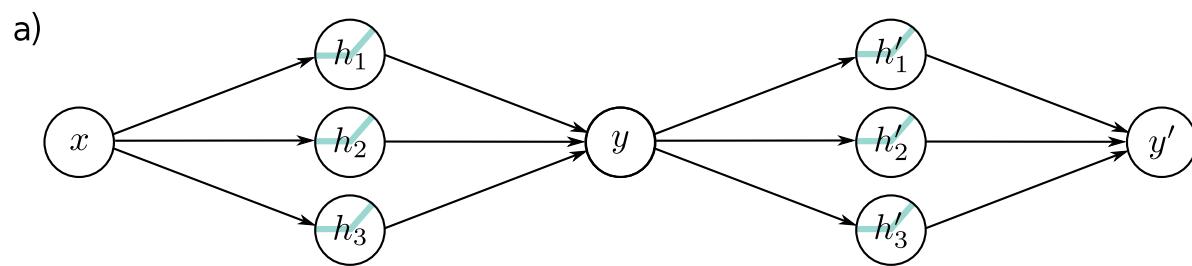
Assume:

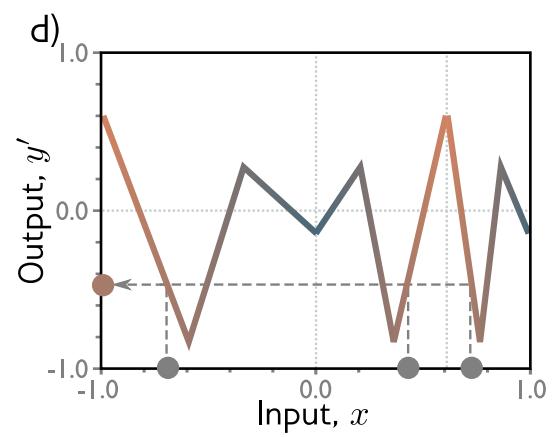
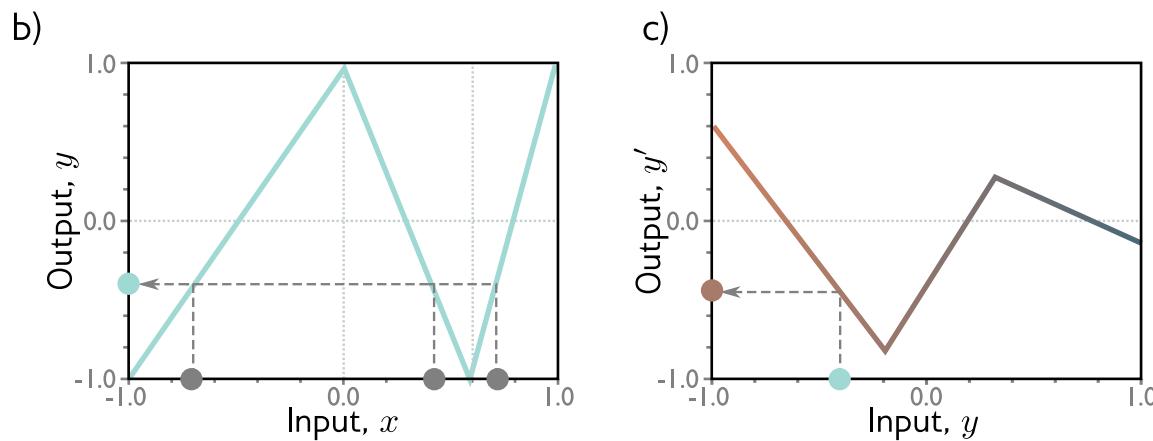
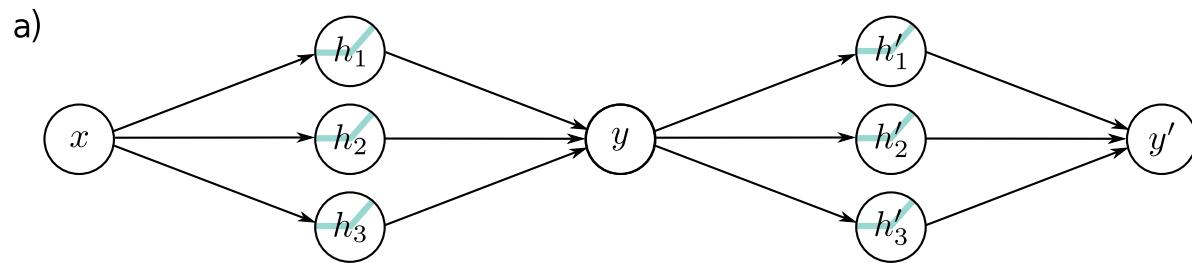
- ReLU Activation
- Slopes and Intercepts as shown
- 3 hidden units in each

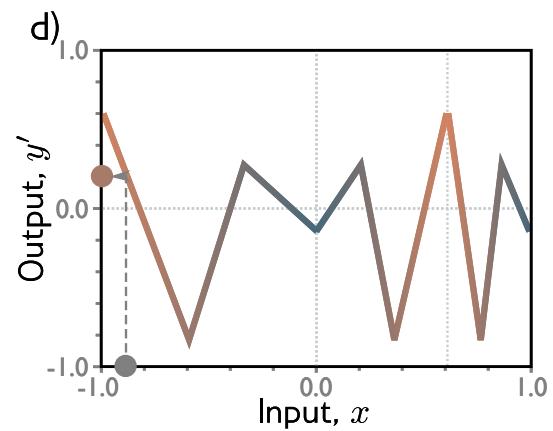
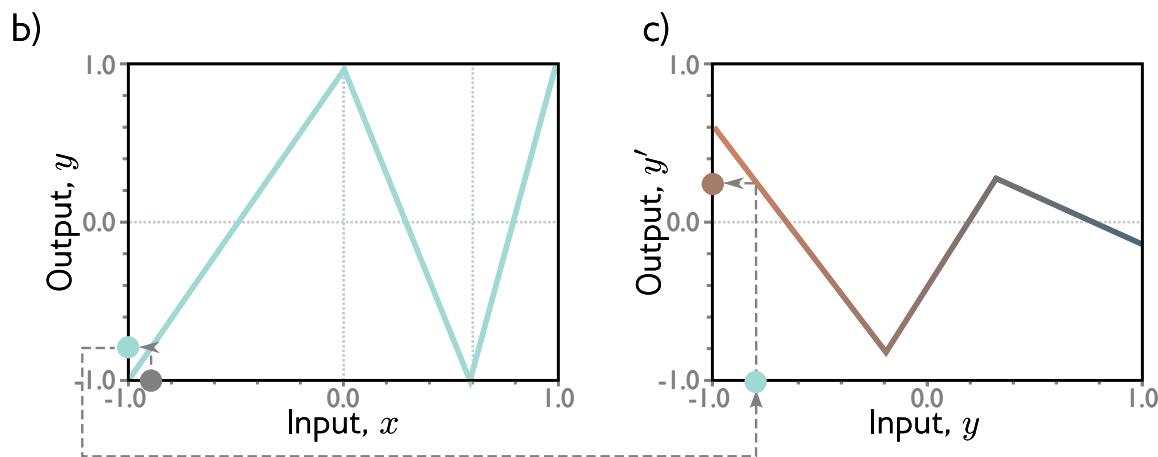
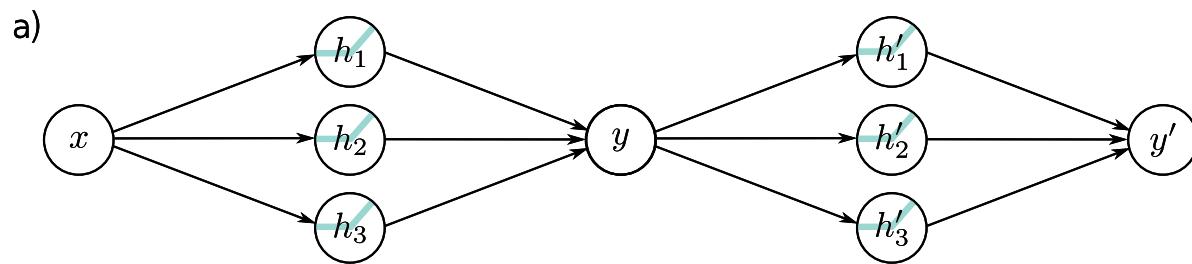
Let's see what happens when we map
 $x \rightarrow y \rightarrow y'$

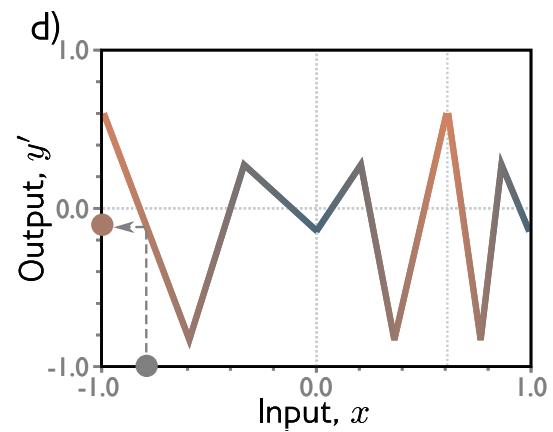
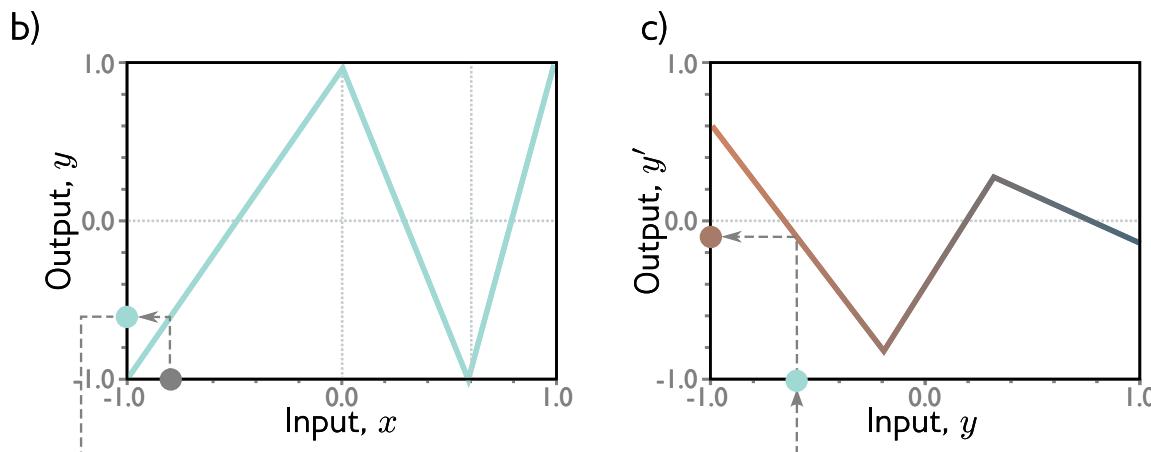
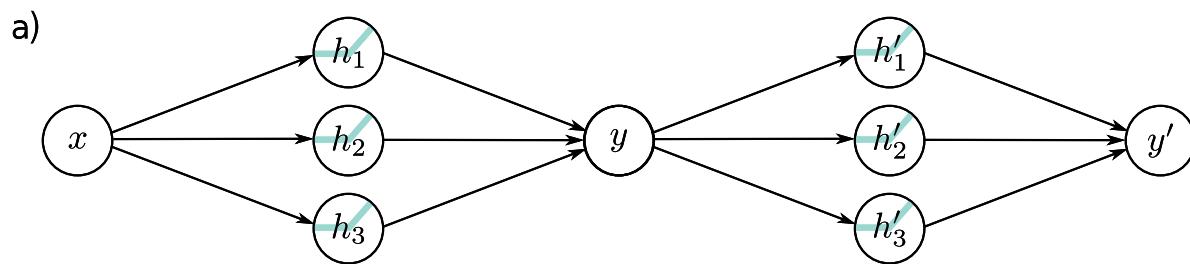
Example: Pick parameters so that
 $x \in [-1,1]$ maps to
 $y \in [-1,1]$ with alternating slope

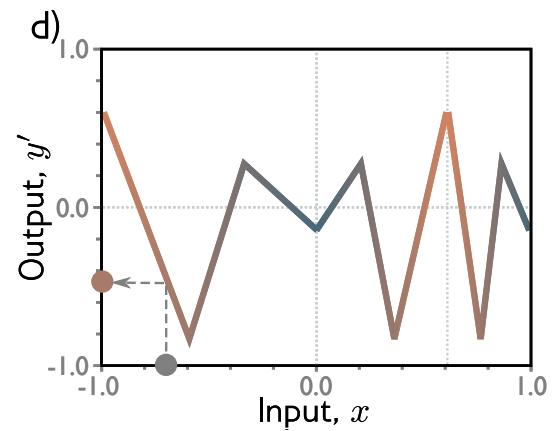
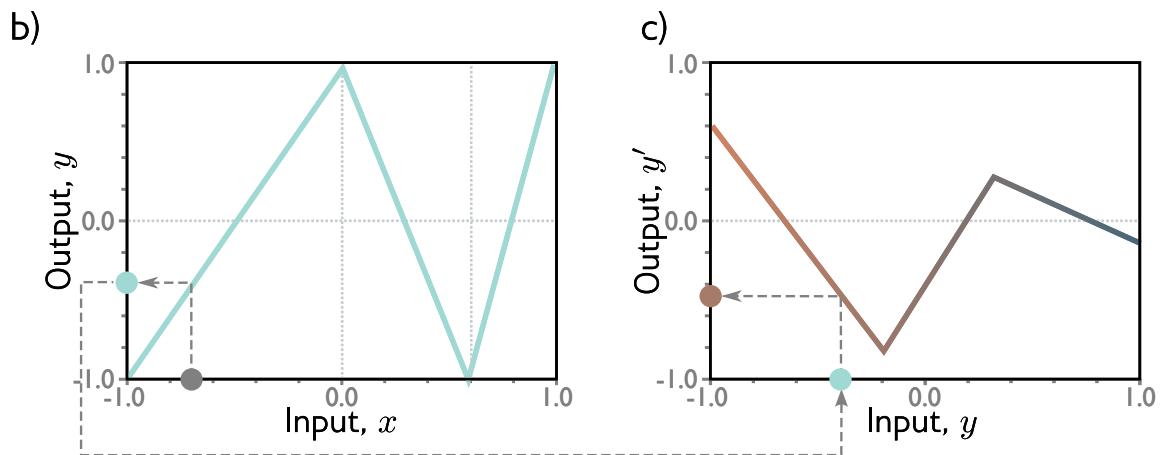
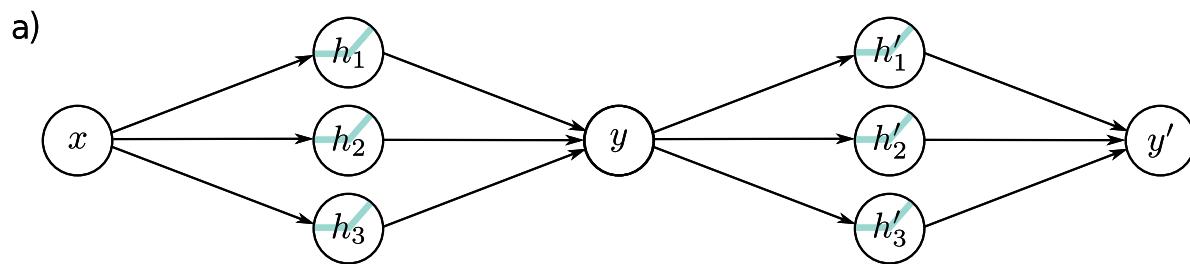


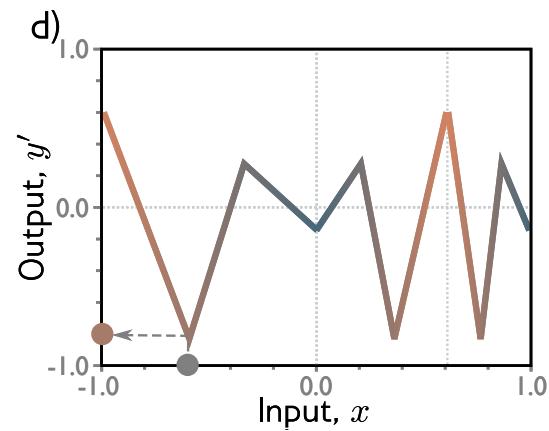
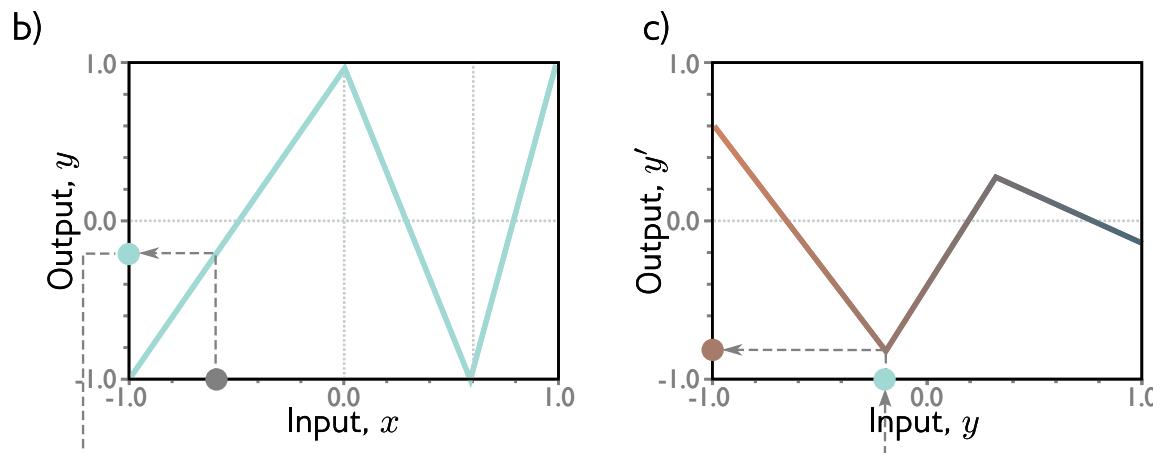
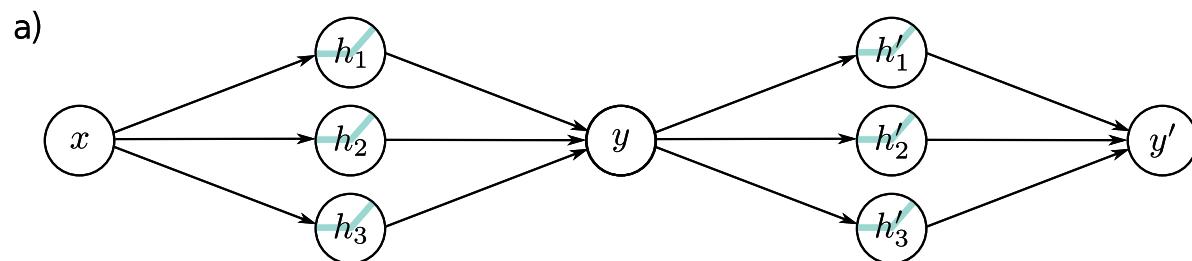


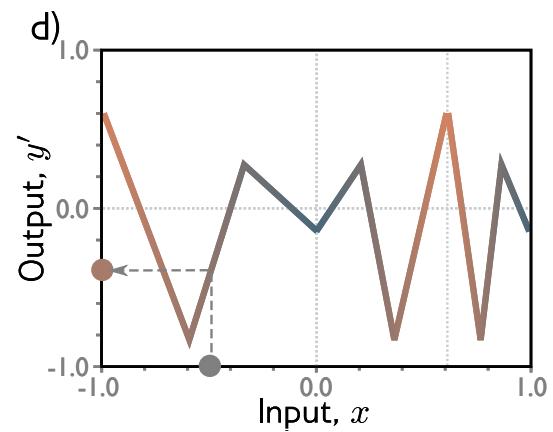
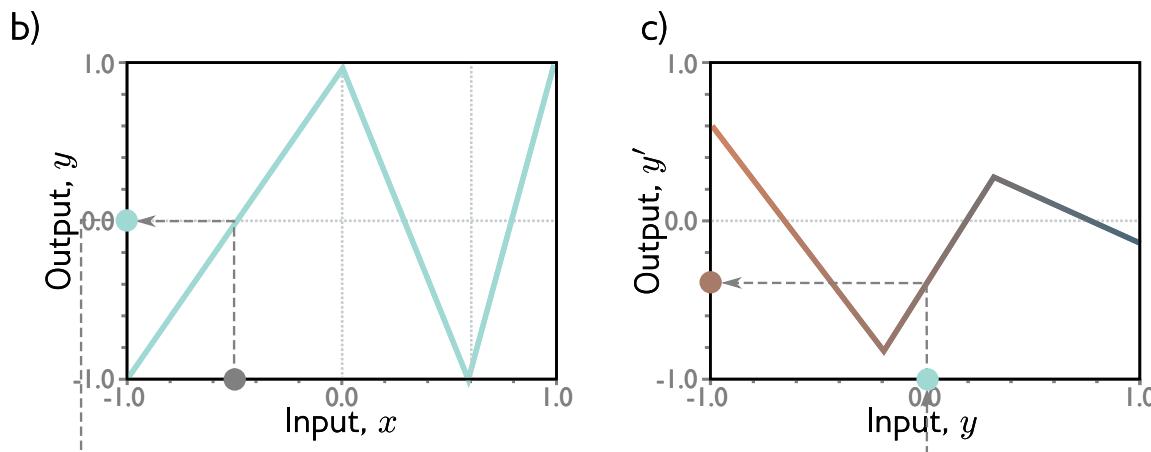
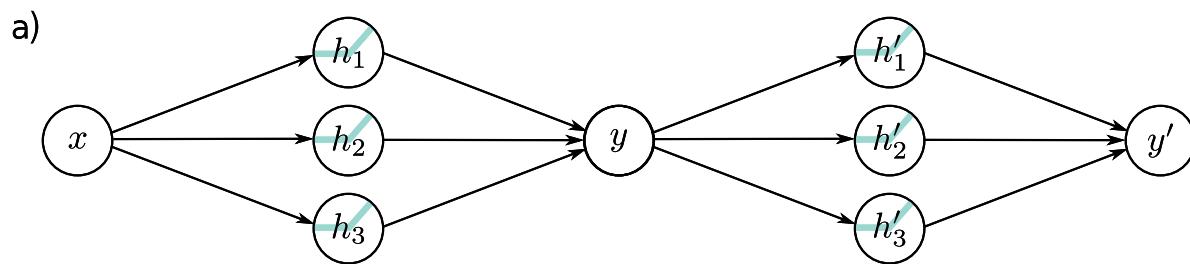


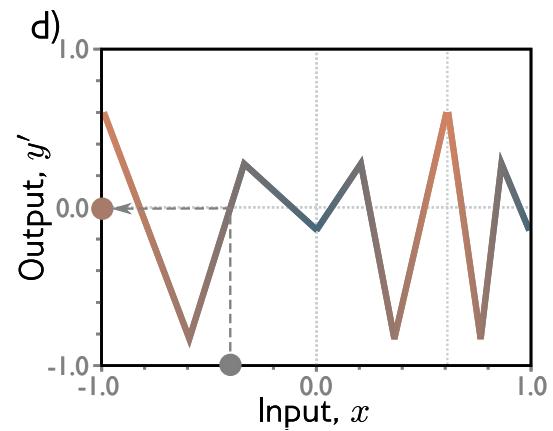
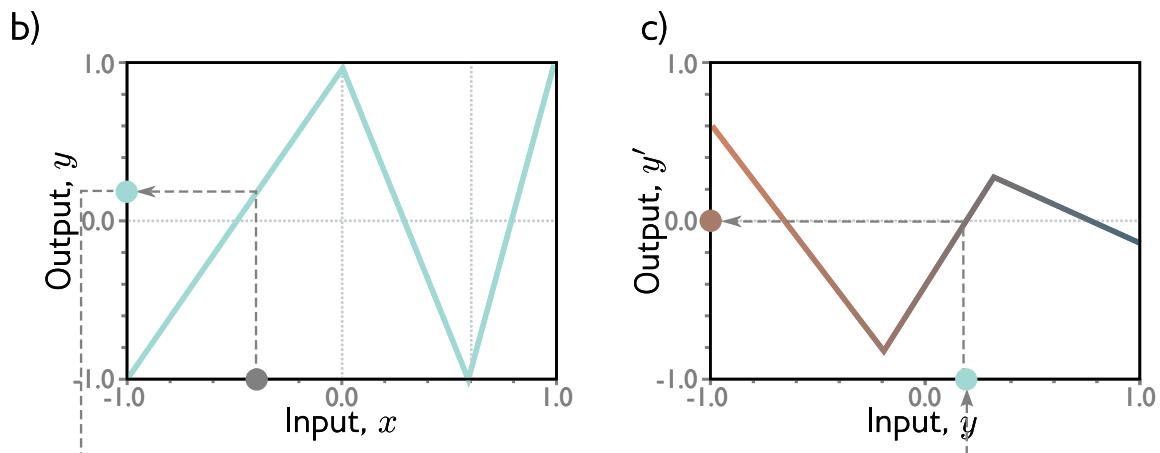
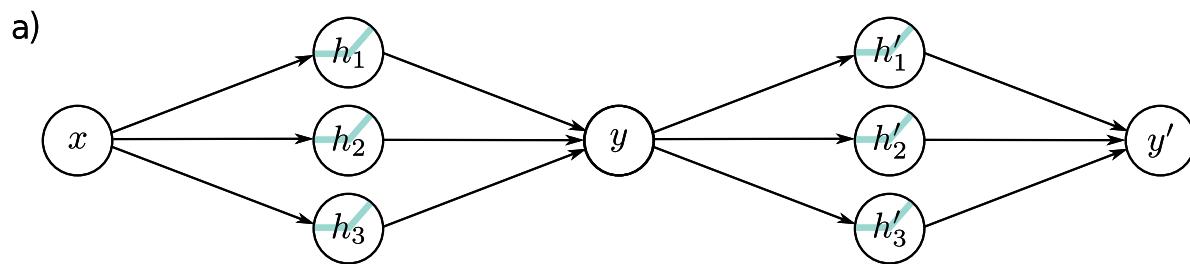


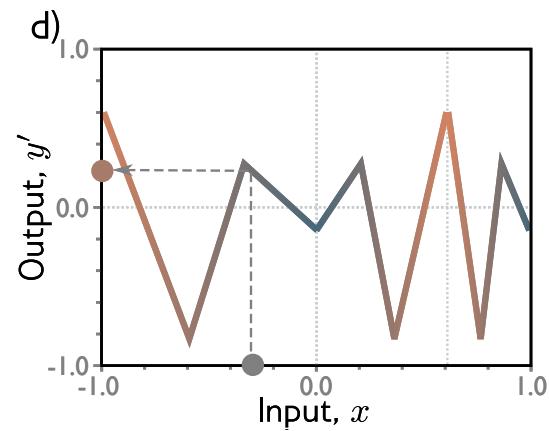
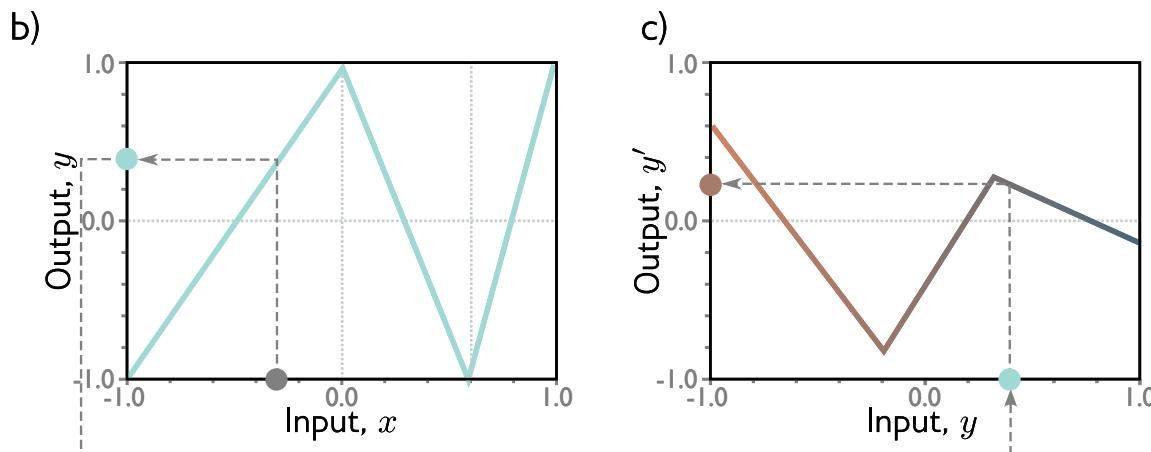
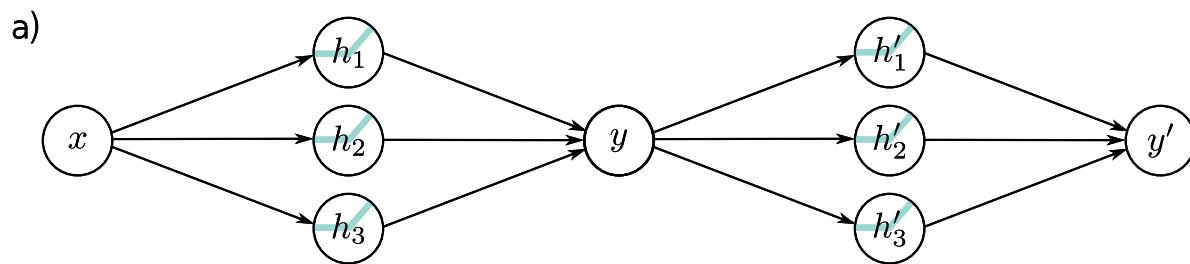


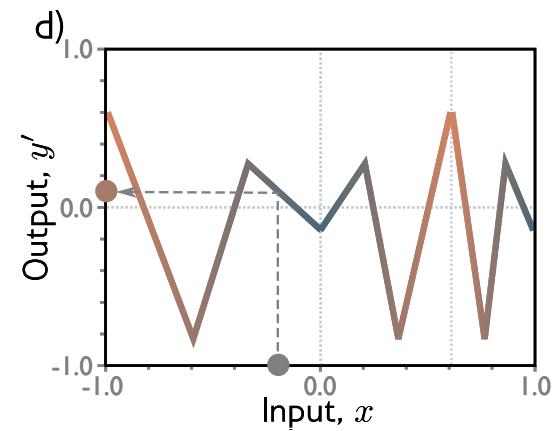
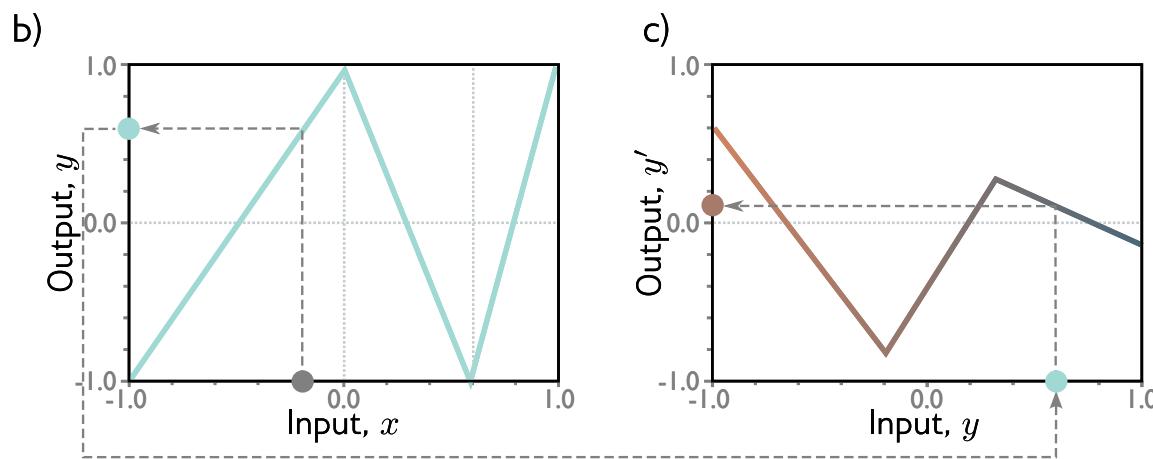
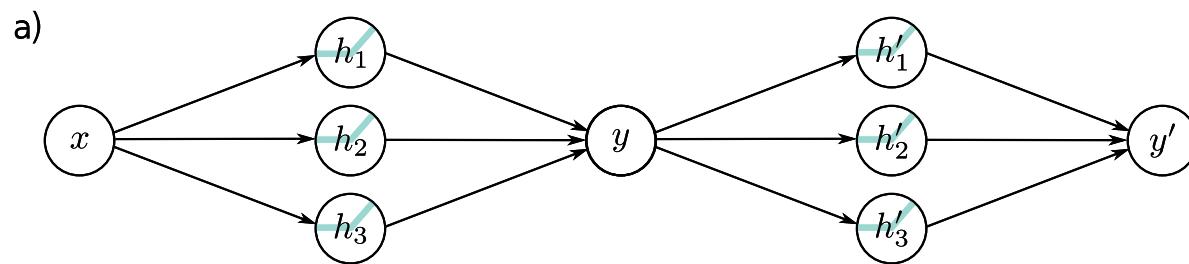


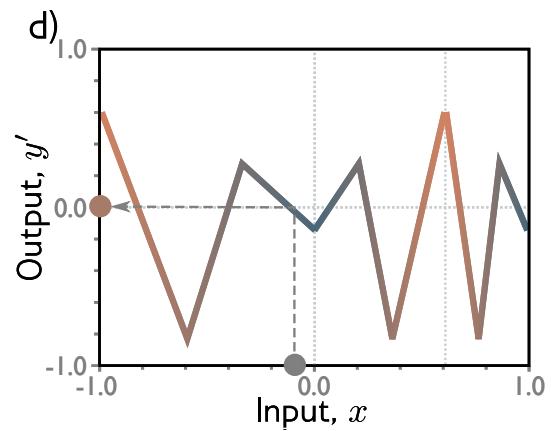
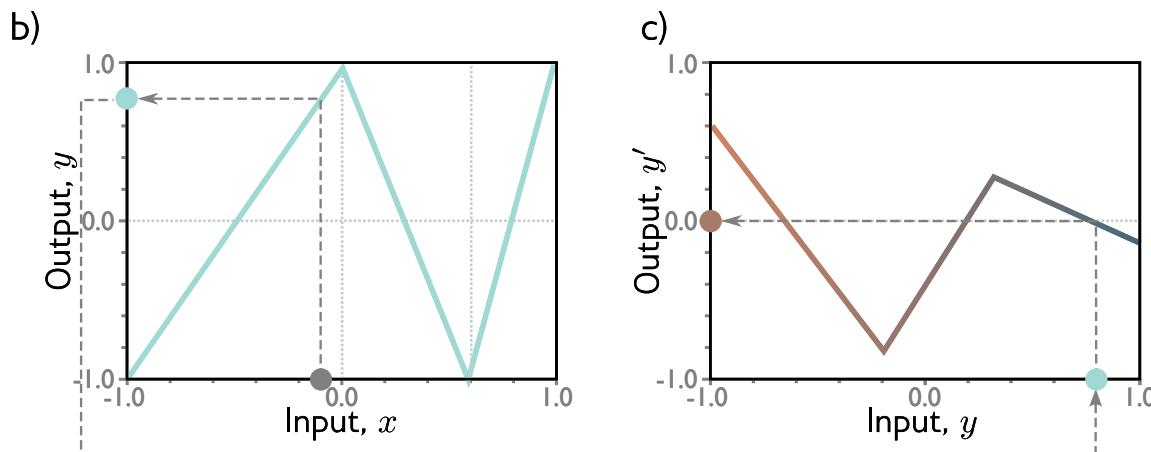
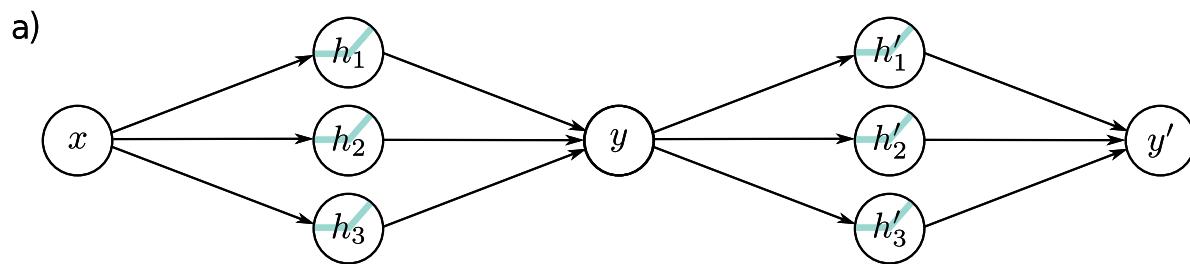


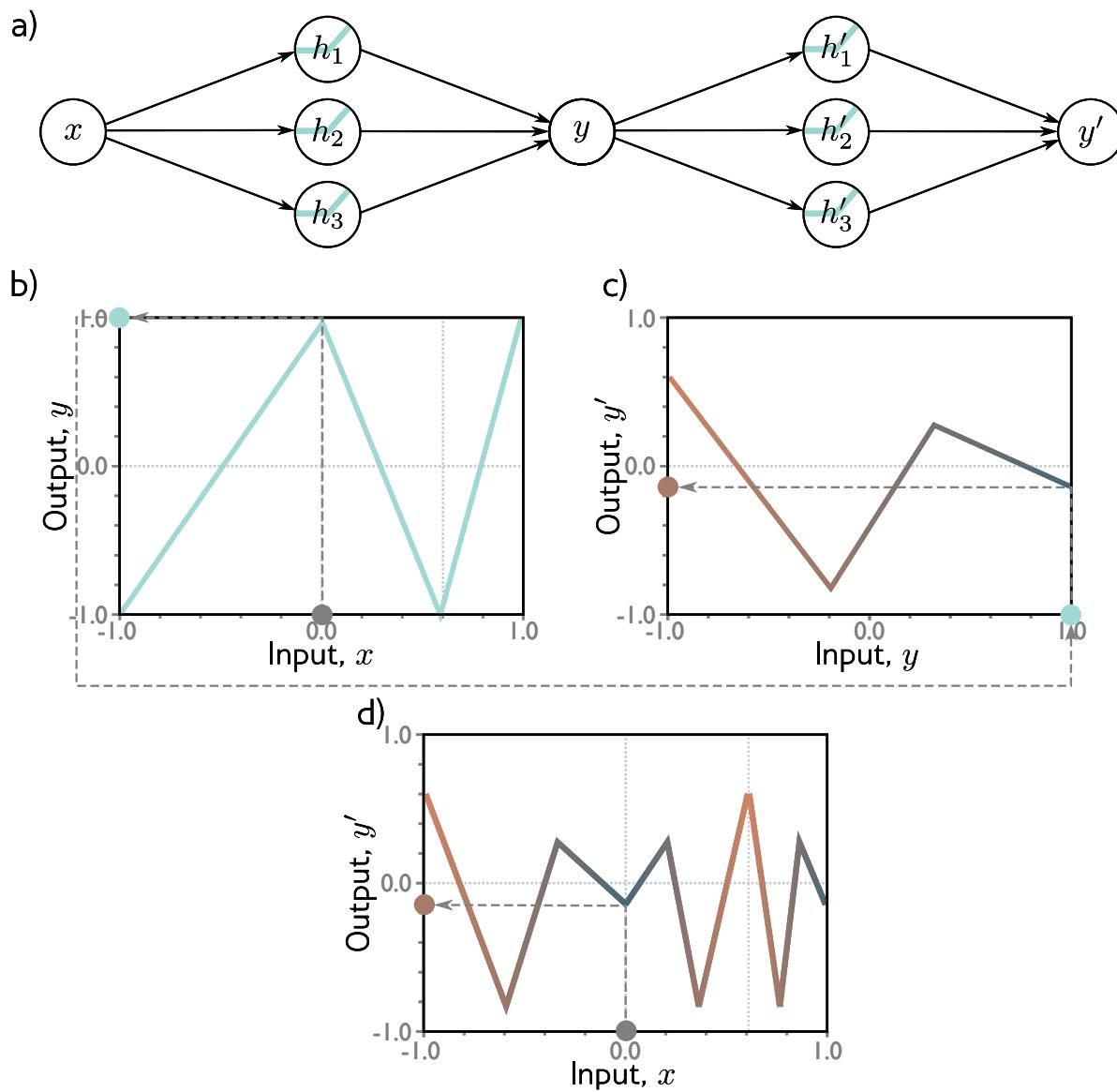


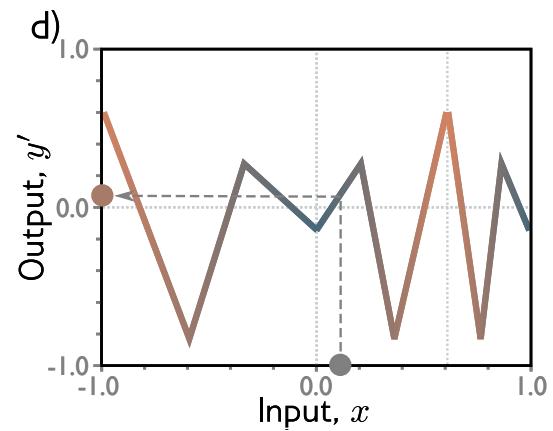
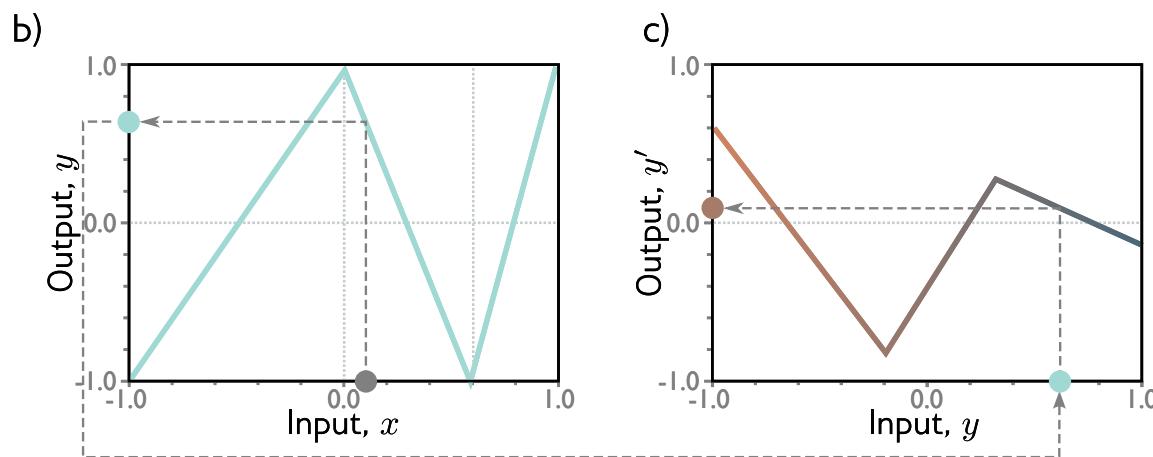
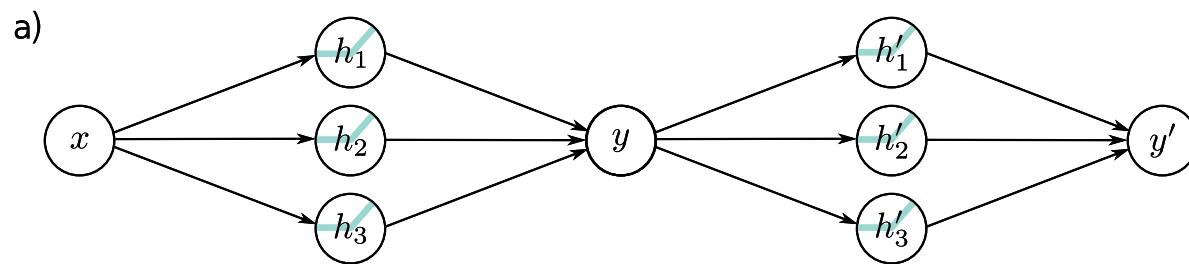


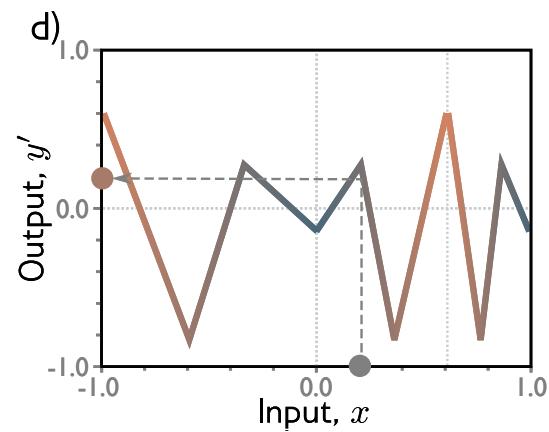
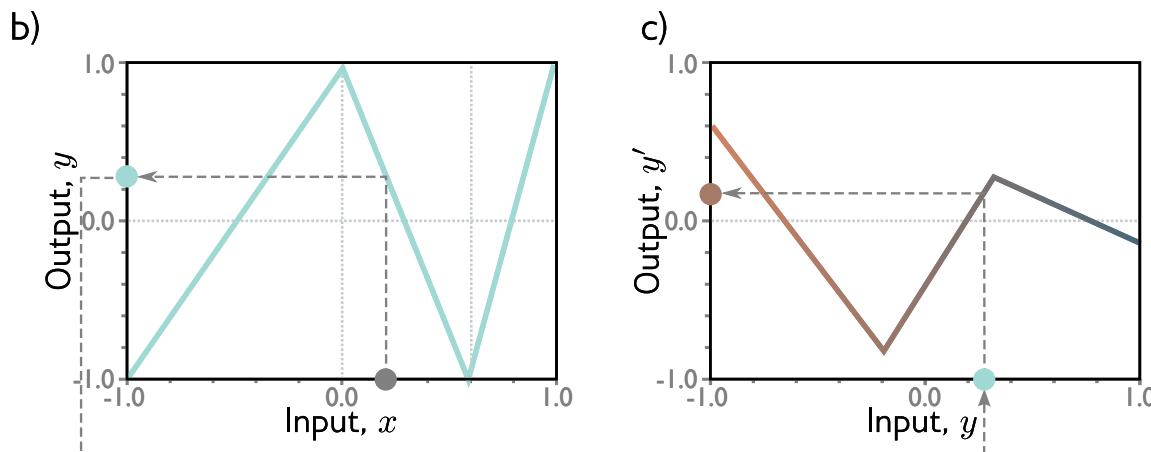
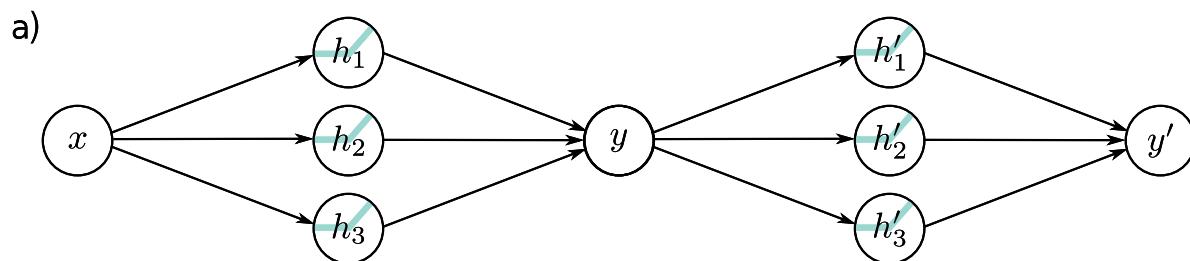


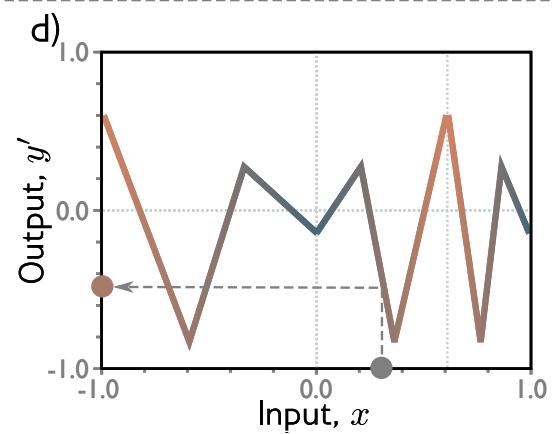
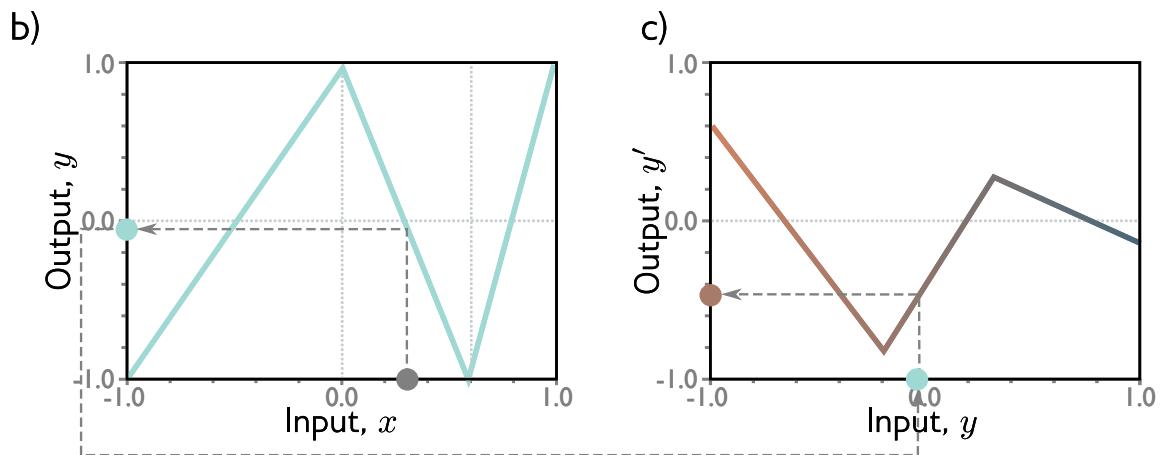
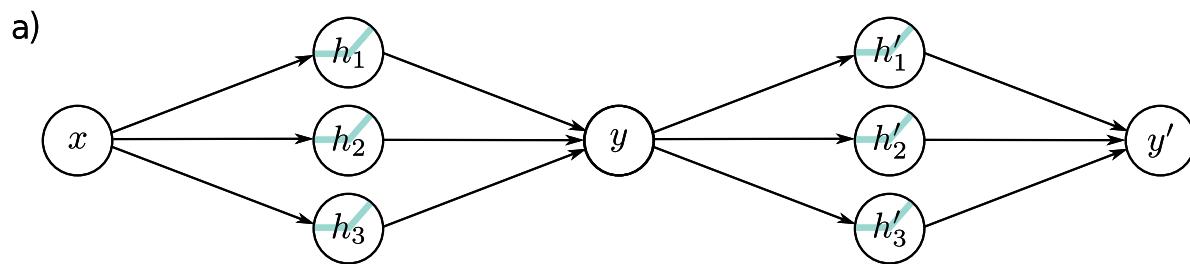


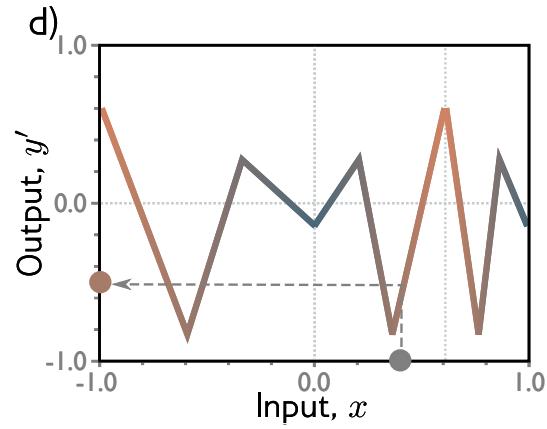
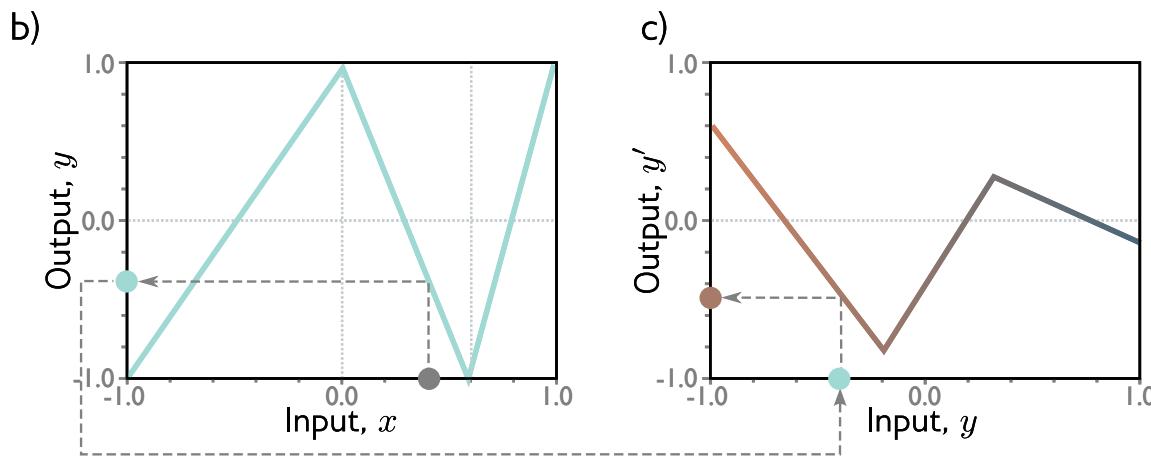
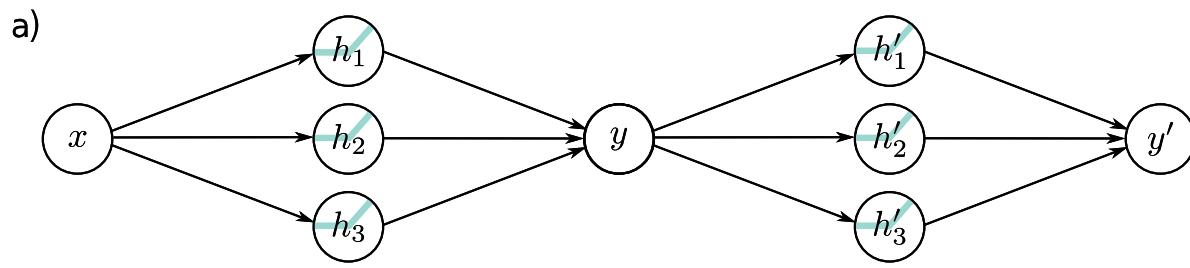


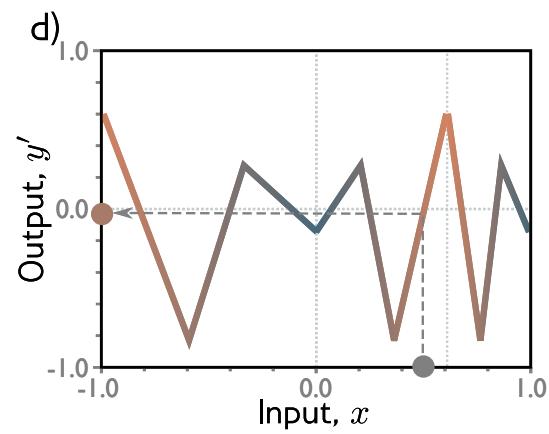
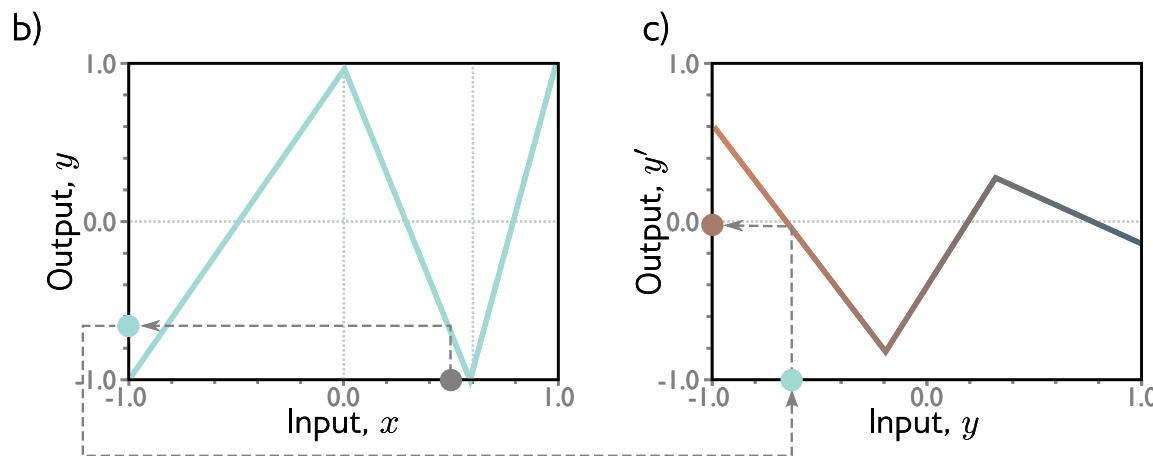
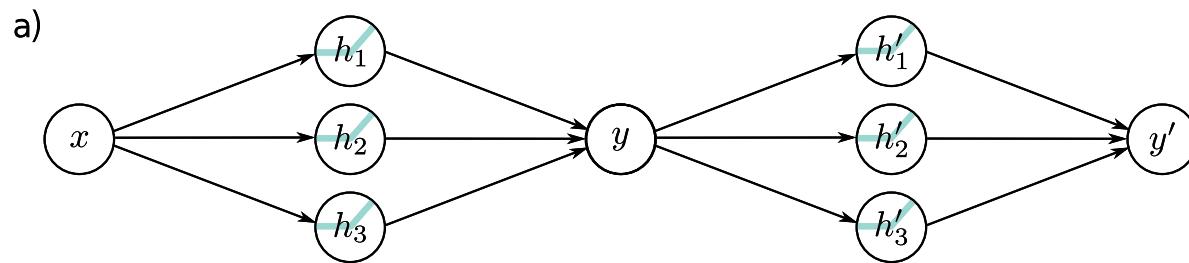


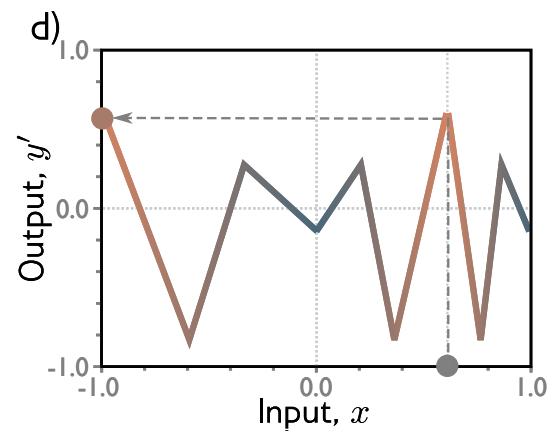
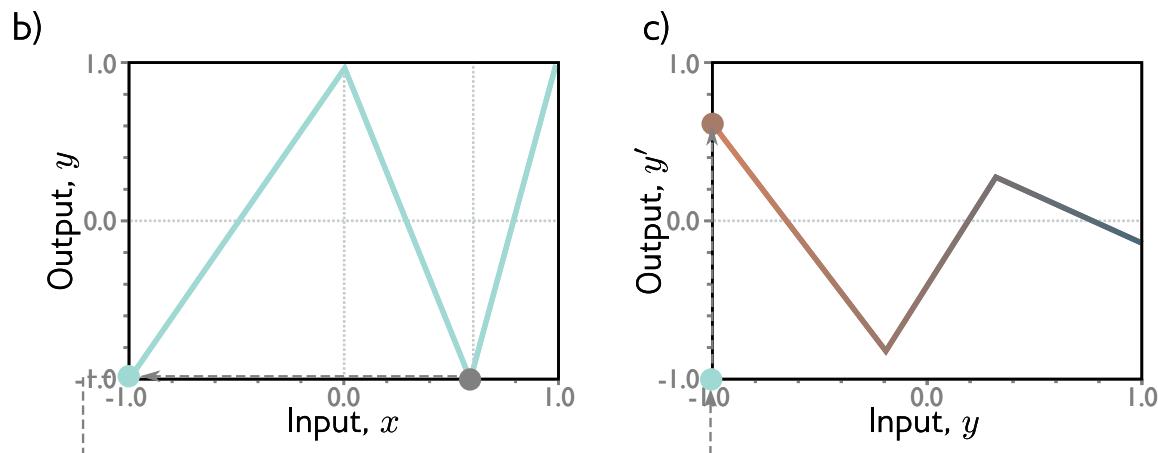
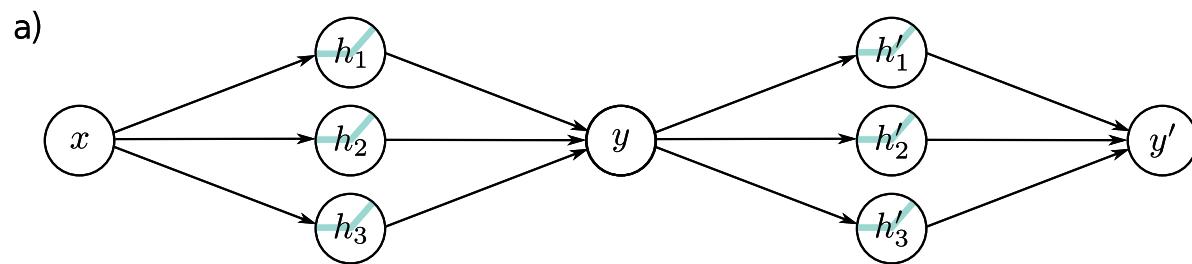


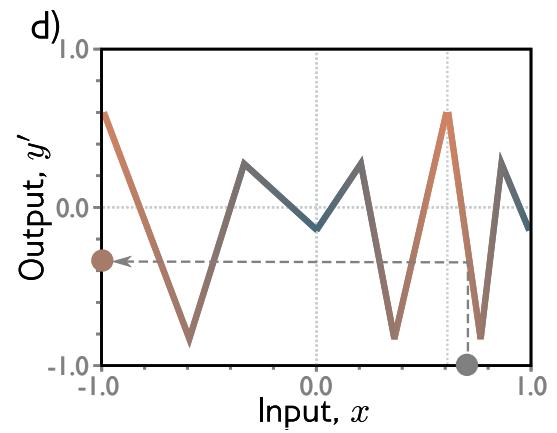
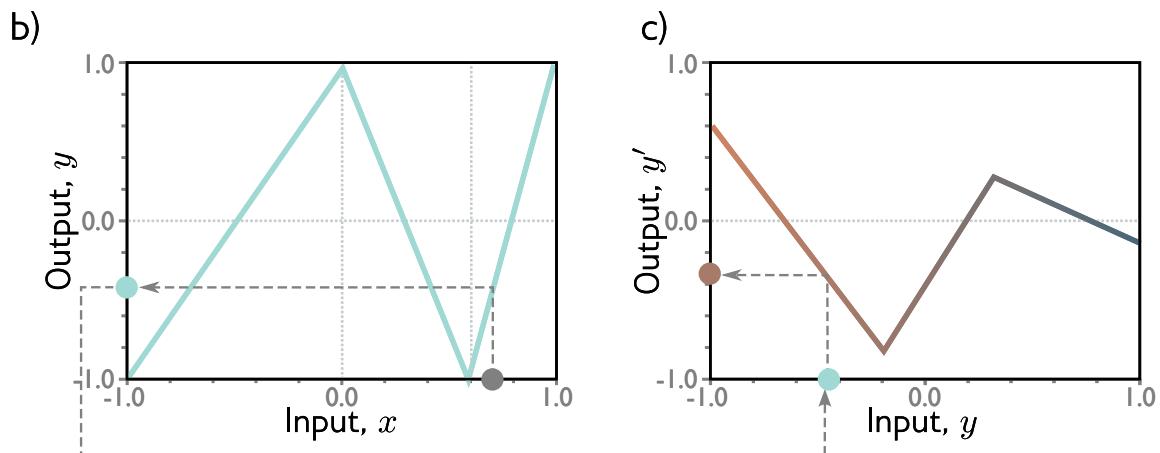
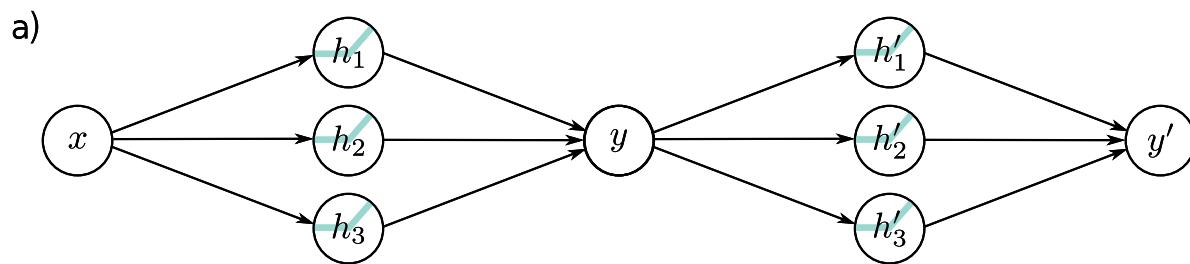


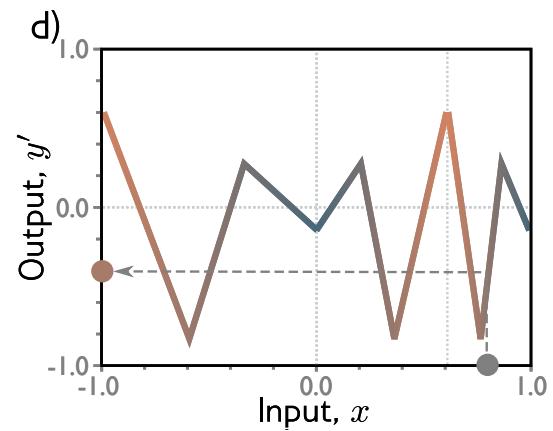
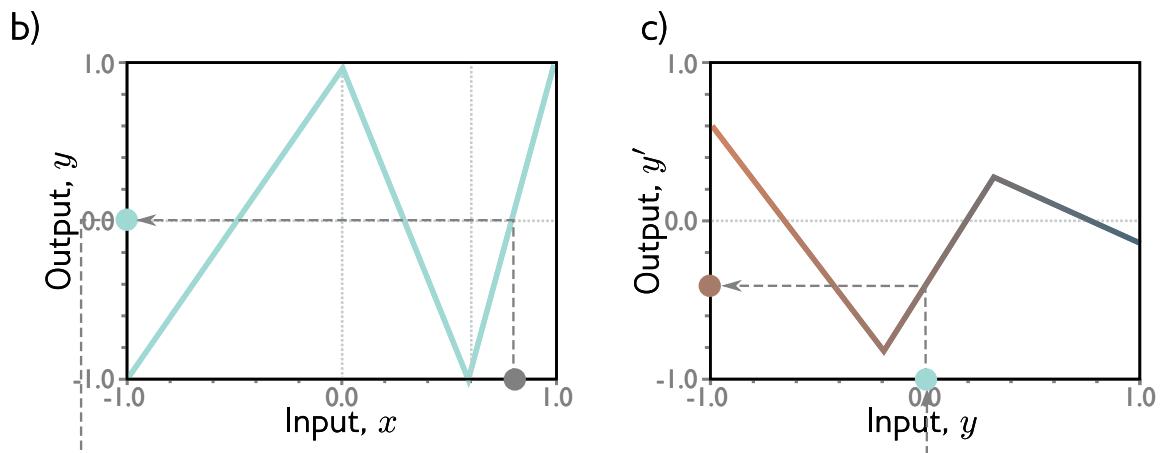
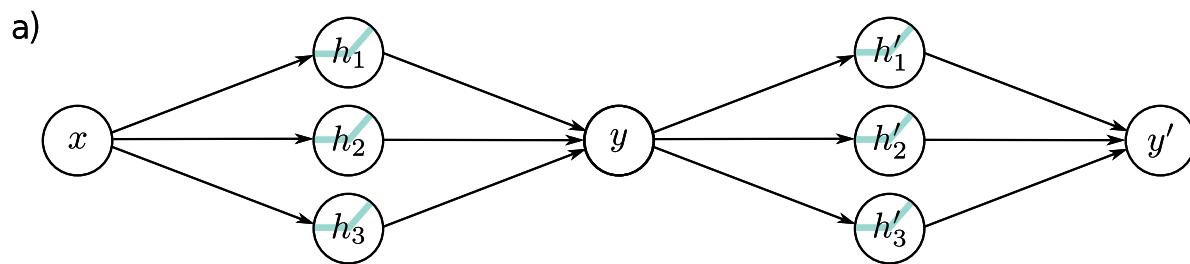


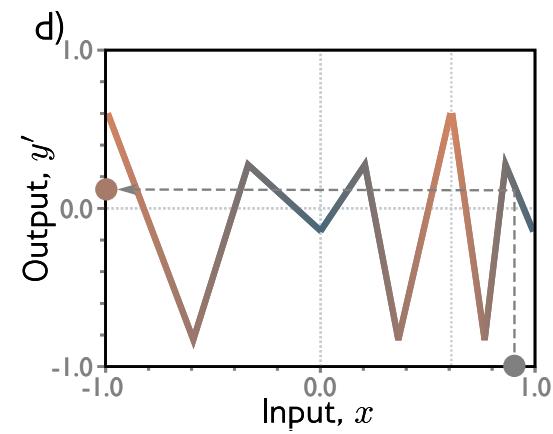
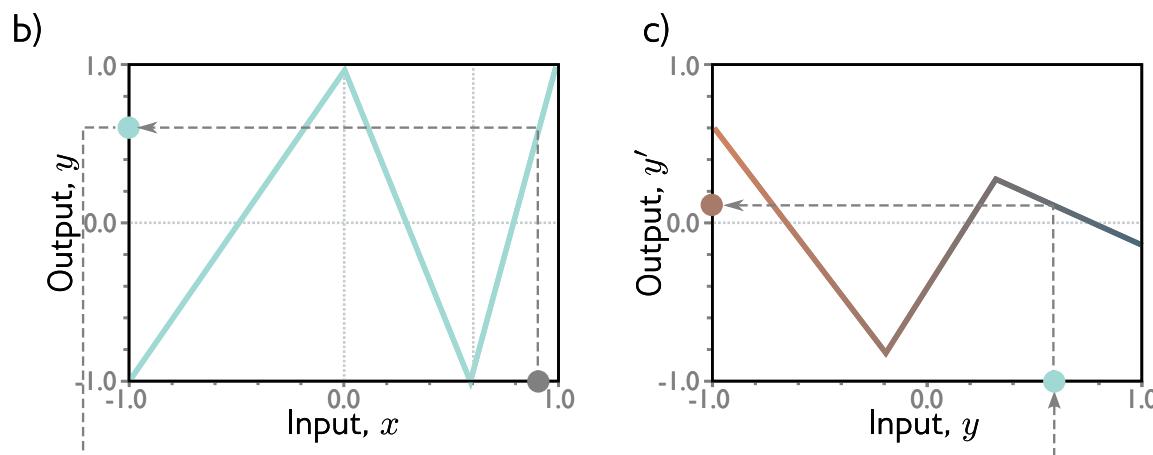
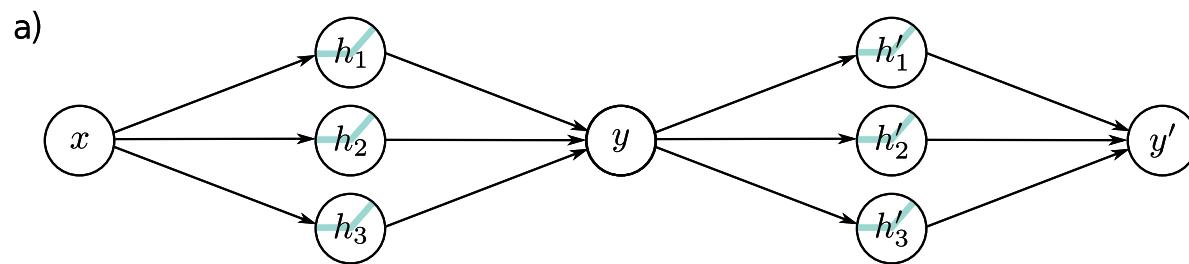




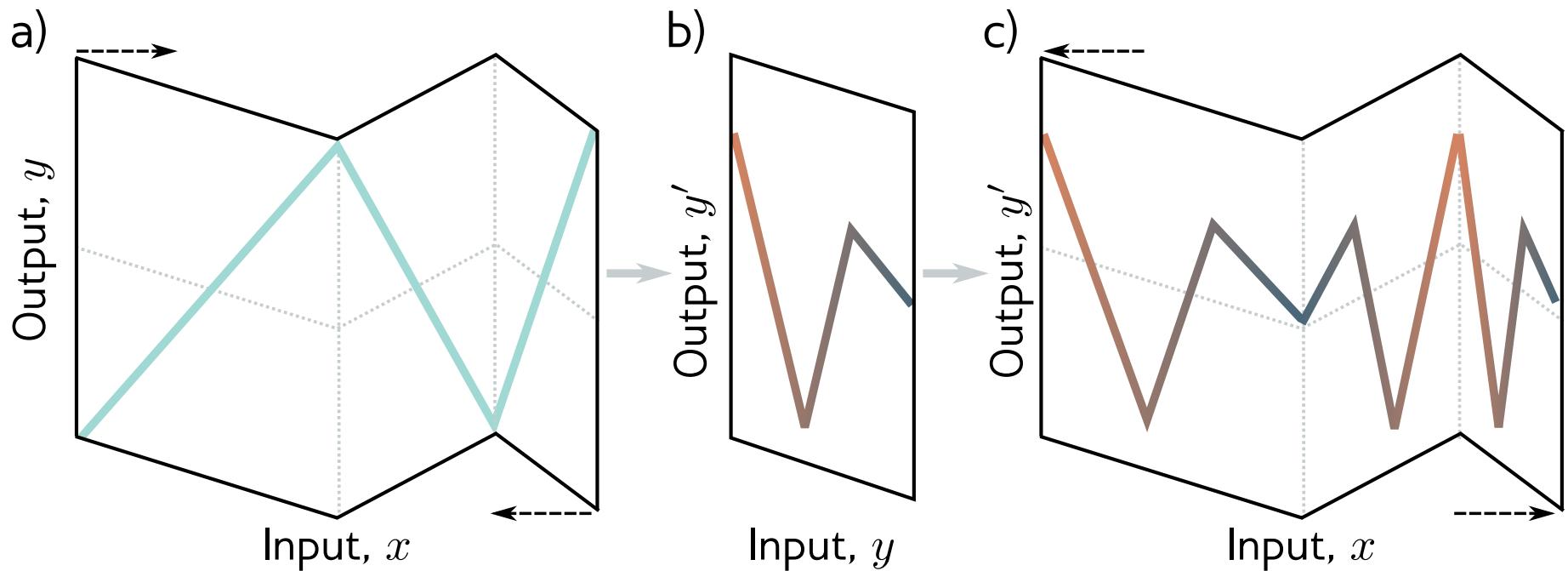




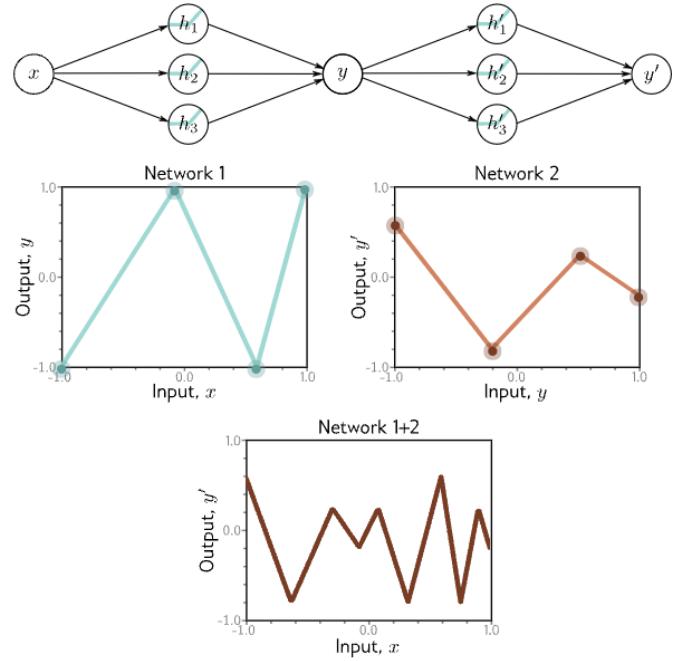




“Folding analogy”



Interactive Figure 4.1 – Concatenating Nets

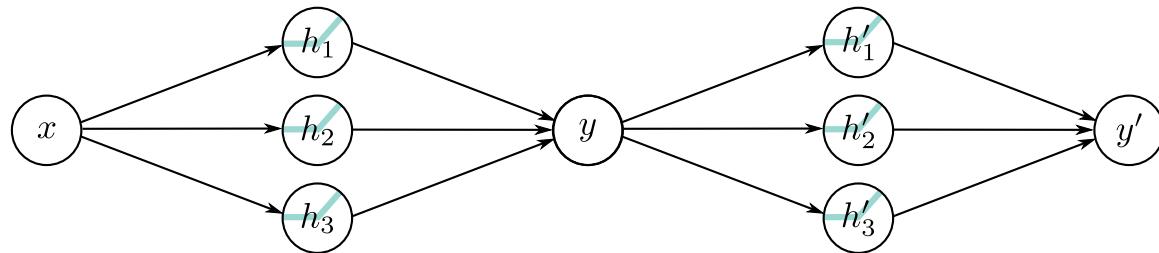


<https://udlbook.github.io/udlfigures/>

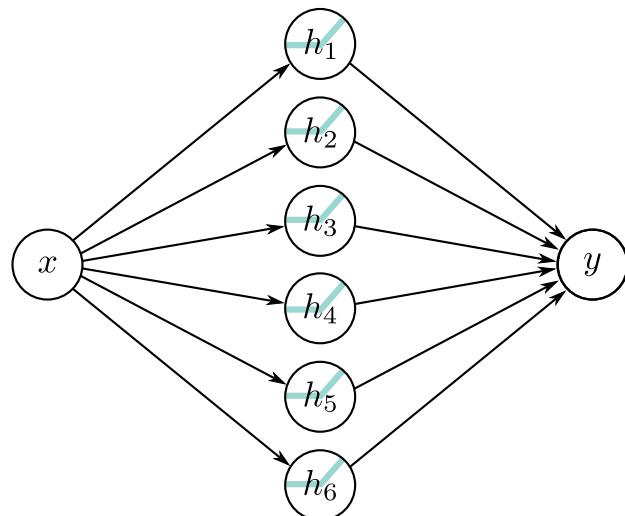
Figure 3.8b Composing two single-layer networks with three hidden units each. The output y of the first network constitutes the input to the second network. (Top left) The first network maps inputs $x \in [-1, 1]$ to outputs $y \in [-1, 1]$ to outputs using a function comprising three linear regions (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.

Comparing to shallow with six hidden units

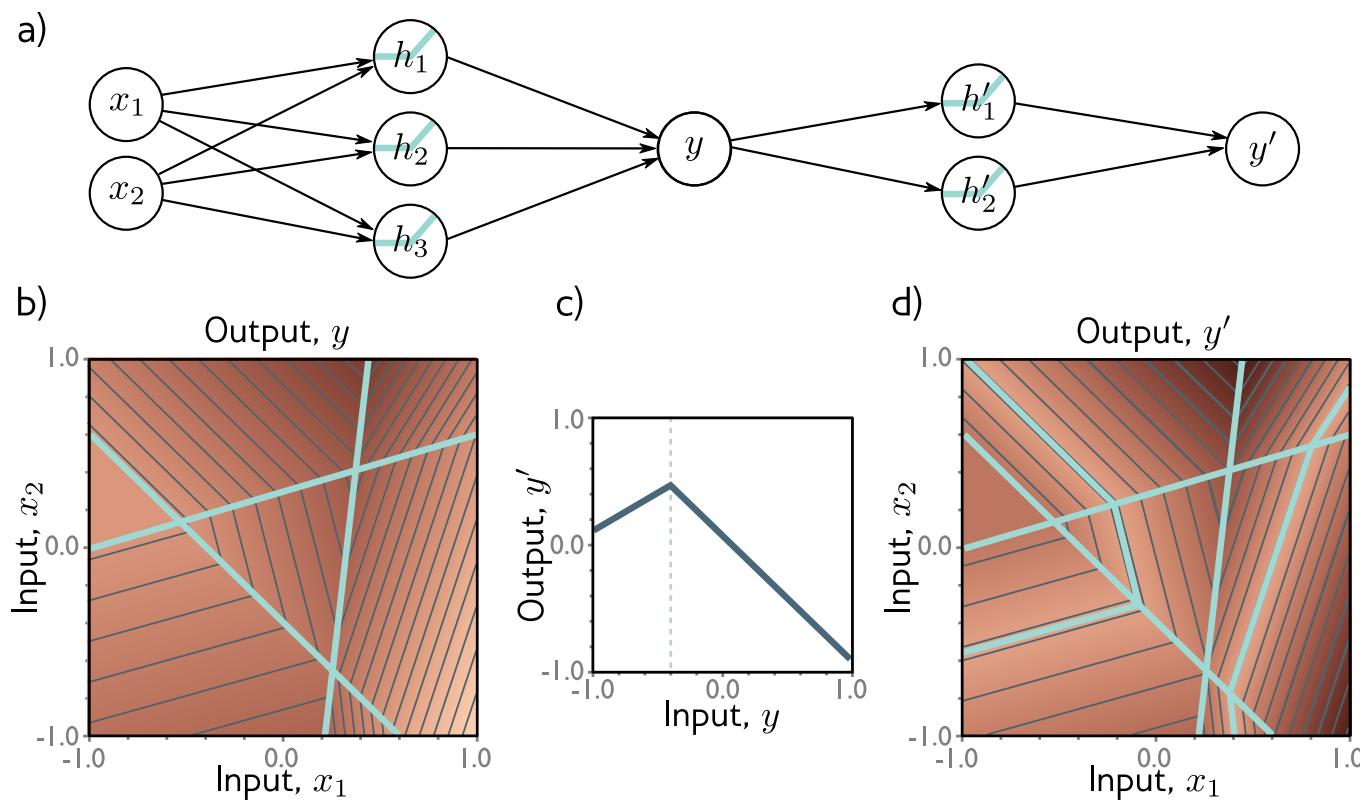


- 20 parameters
- (at least) 9 regions



- 19 parameters
- Max 7 regions

Composing networks in 2D



Deep neural networks

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

θ : theta
 ϕ : phi

Combine two networks into one

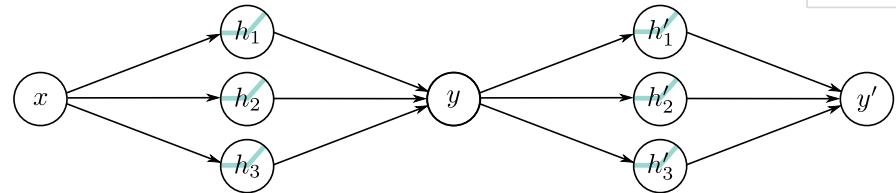
Let's start with 2 networks:

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

Network 1:
(input is x)

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

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



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

$$h'_1 = a[\theta'_{10} + \theta'_{11}y]$$

Network 2:
(input is y)

$$h'_2 = a[\theta'_{20} + \theta'_{21}y]$$

$$h'_3 = a[\theta'_{30} + \theta'_{31}y]$$

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

θ : theta
 ϕ : phi

Combine two networks into one

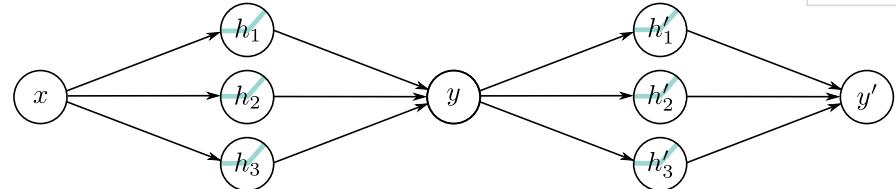
Let's start with 2 networks:

Network 1:
(input is x)

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

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

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



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

Network 2:
(input is y)

$$h'_1 = a[\theta'_{10} + \theta'_{11}y]$$

$$h'_2 = a[\theta'_{20} + \theta'_{21}y]$$

$$h'_3 = a[\theta'_{30} + \theta'_{31}y]$$

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

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

$$h'_1 = a[\theta'_{10} + \theta'_{11}y] = a[\theta'_{10} + \theta'_{11}\phi_0 + \theta'_{11}\phi_1 h_1 + \theta'_{11}\phi_2 h_2 + \theta'_{11}\phi_3 h_3]$$

$$h'_2 = a[\theta'_{20} + \theta'_{21}y] = a[\theta'_{20} + \theta'_{21}\phi_0 + \theta'_{21}\phi_1 h_1 + \theta'_{21}\phi_2 h_2 + \theta'_{21}\phi_3 h_3]$$

$$h'_3 = a[\theta'_{30} + \theta'_{31}y] = a[\theta'_{30} + \theta'_{31}\phi_0 + \theta'_{31}\phi_1 h_1 + \theta'_{31}\phi_2 h_2 + \theta'_{31}\phi_3 h_3]$$

θ	: theta
ϕ	: phi
ψ	: psi

Create new variables: ψ (psi)

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

$$\begin{aligned} h'_1 &= a[\theta'_{10} + \theta'_{11}y] = a[\theta'_{10} + \theta'_{11}\phi_0 + \theta'_{11}\phi_1h_1 + \theta'_{11}\phi_2h_2 + \theta'_{11}\phi_3h_3] \\ h'_2 &= a[\theta'_{20} + \theta'_{21}y] = a[\theta'_{20} + \theta'_{21}\phi_0 + \theta'_{21}\phi_1h_1 + \theta'_{21}\phi_2h_2 + \theta'_{21}\phi_3h_3] \\ h'_3 &= a[\theta'_{30} + \theta'_{31}y] = a[\theta'_{30} + \theta'_{31}\phi_0 + \theta'_{31}\phi_1h_1 + \theta'_{31}\phi_2h_2 + \theta'_{31}\phi_3h_3] \end{aligned}$$

Collect and rename the variables for conciseness.

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

θ : theta

ϕ : phi

ψ : psi

We get a two-layer network

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

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

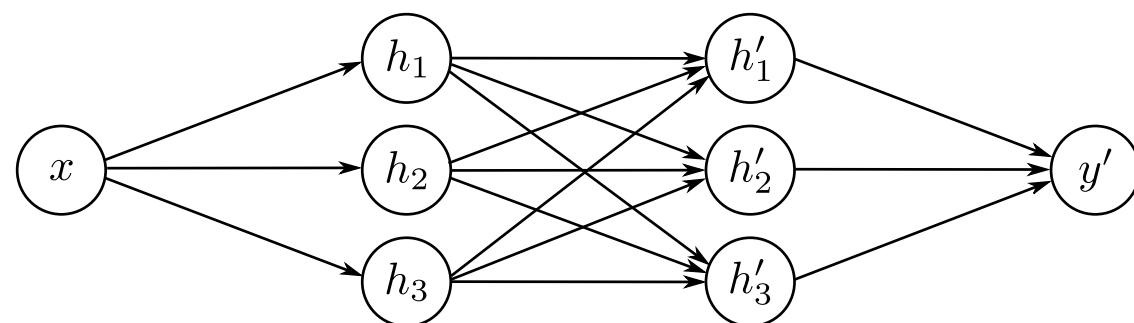
$$h_3 = a[\theta_{30} + \theta_{31}x]$$

$$h'_1 = a[\psi_{10} + \psi_{11}h_1 + \psi_{12}h_2 + \psi_{13}h_3]$$

$$h'_2 = a[\psi_{20} + \psi_{21}h_2 + \psi_{22}h_2 + \psi_{23}h_3]$$

$$h'_3 = a[\psi_{30} + \psi_{31}h_2 + \psi_{32}h_2 + \psi_{33}h_3]$$

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



θ : theta

ϕ : phi

ψ : psi

Two-layer network as one equation

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

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

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

$$h'_1 = a[\psi_{10} + \psi_{11}h_1 + \psi_{12}h_2 + \psi_{13}h_3]$$

$$h'_2 = a[\psi_{20} + \psi_{21}h_1 + \psi_{22}h_2 + \psi_{23}h_3]$$

$$h'_3 = a[\psi_{30} + \psi_{31}h_1 + \psi_{32}h_2 + \psi_{33}h_3]$$

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

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

Remember shallow network with two outputs?

- 1 input, 4 hidden units, 2 outputs

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

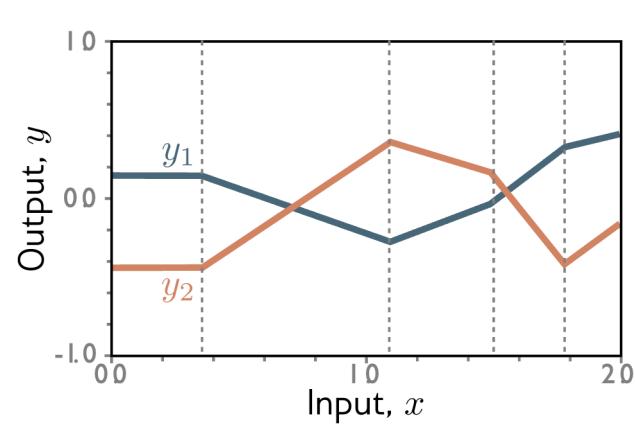
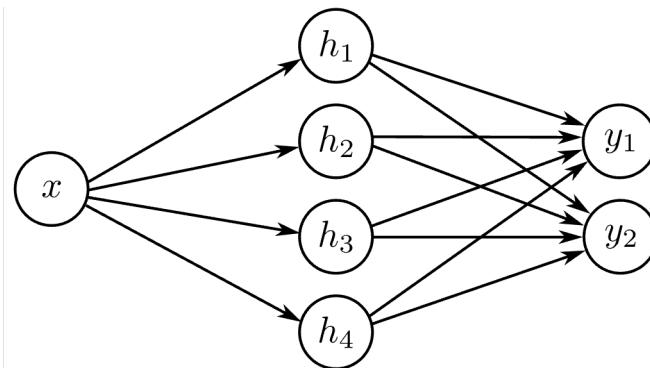
$$h_2 = a[\theta_{20} + \theta_{21}x]$$

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

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

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

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



Networks as composing functions

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

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

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

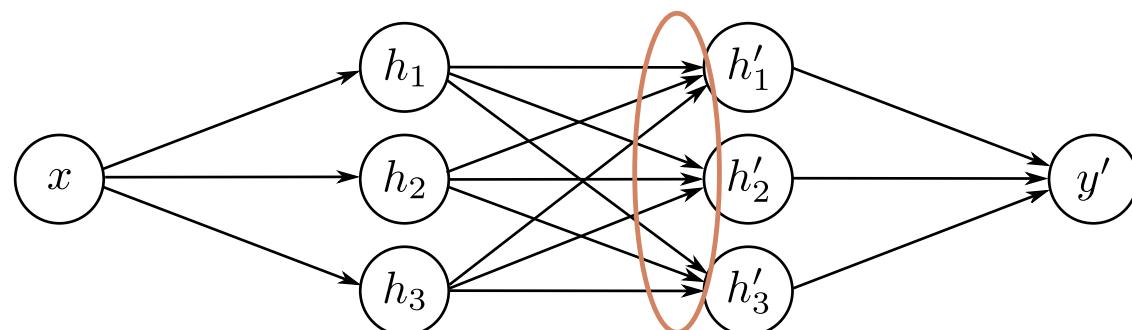
$$h'_1 = a[\psi_{10} + \psi_{11}h_1 + \psi_{12}h_2 + \psi_{13}h_3]$$

$$h'_2 = a[\psi_{20} + \psi_{21}h_1 + \psi_{22}h_2 + \psi_{23}h_3]$$

$$h'_3 = a[\psi_{30} + \psi_{31}h_1 + \psi_{32}h_2 + \psi_{33}h_3]$$

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

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

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

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

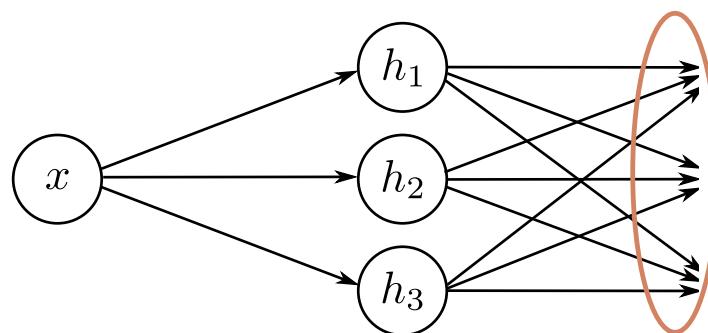
$$h'_1 = a[\psi_{10} + \psi_{11}h_1 + \psi_{12}h_2 + \psi_{13}h_3]$$

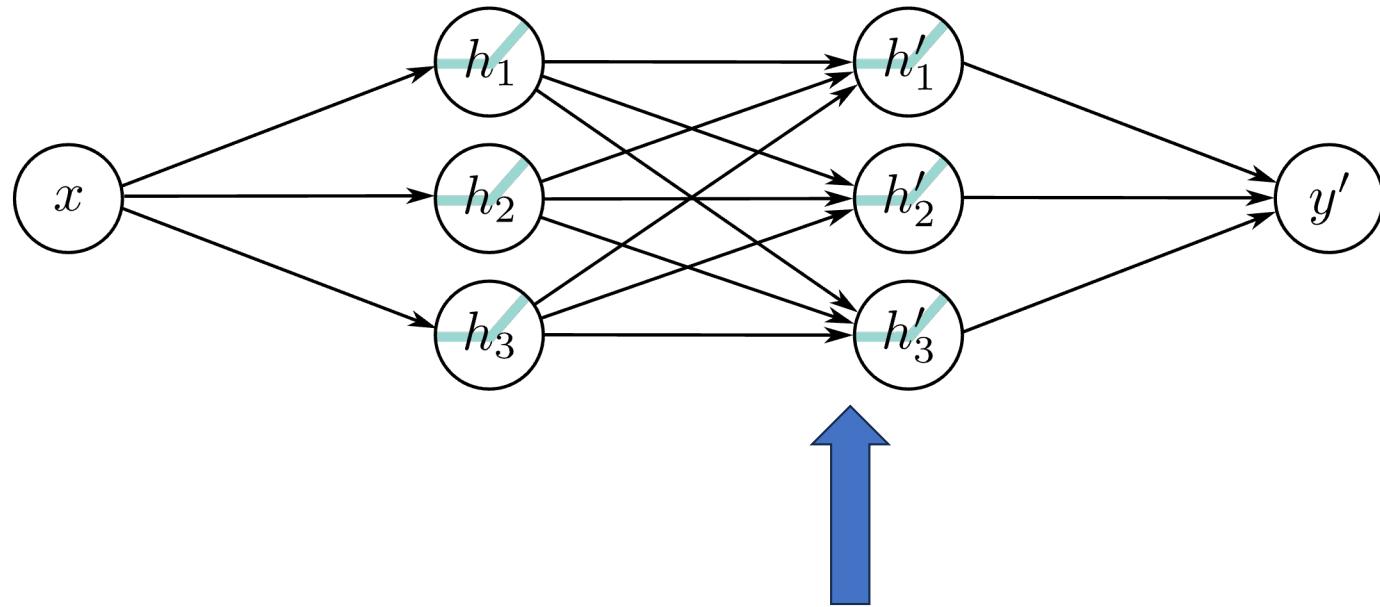
$$h'_2 = a[\psi_{20} + \psi_{21}h_1 + \psi_{22}h_2 + \psi_{23}h_3]$$

$$h'_3 = a[\psi_{30} + \psi_{31}h_1 + \psi_{32}h_2 + \psi_{33}h_3]$$

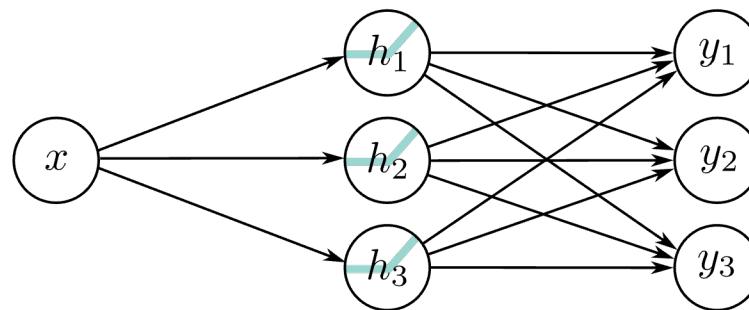
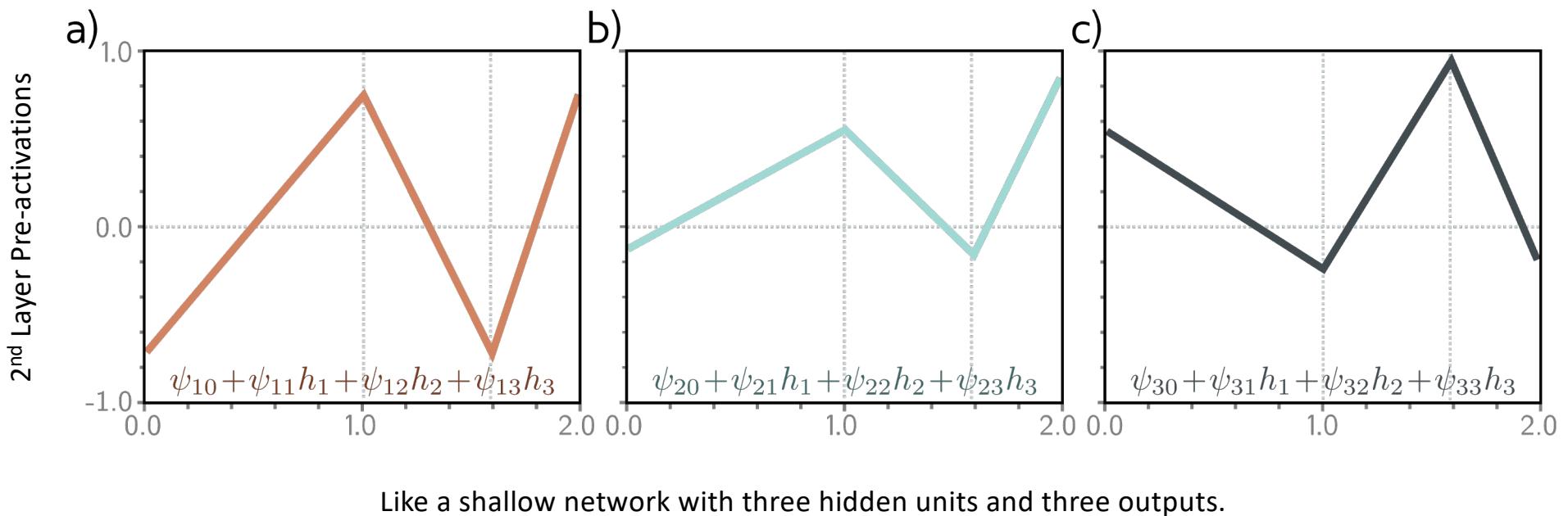
Consider the pre-activations at the second hidden units

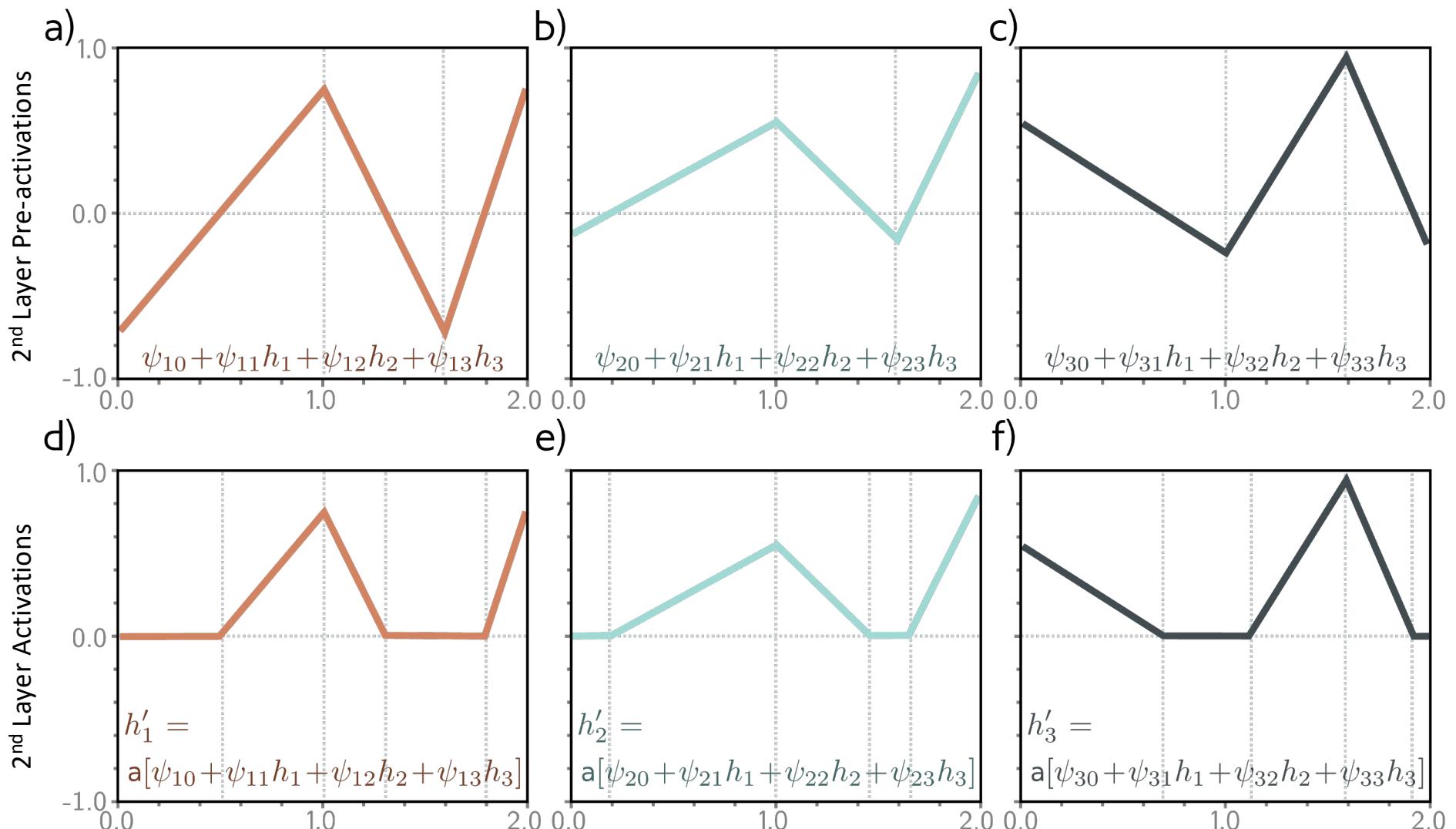
At this point, it's a one--layer network with three outputs

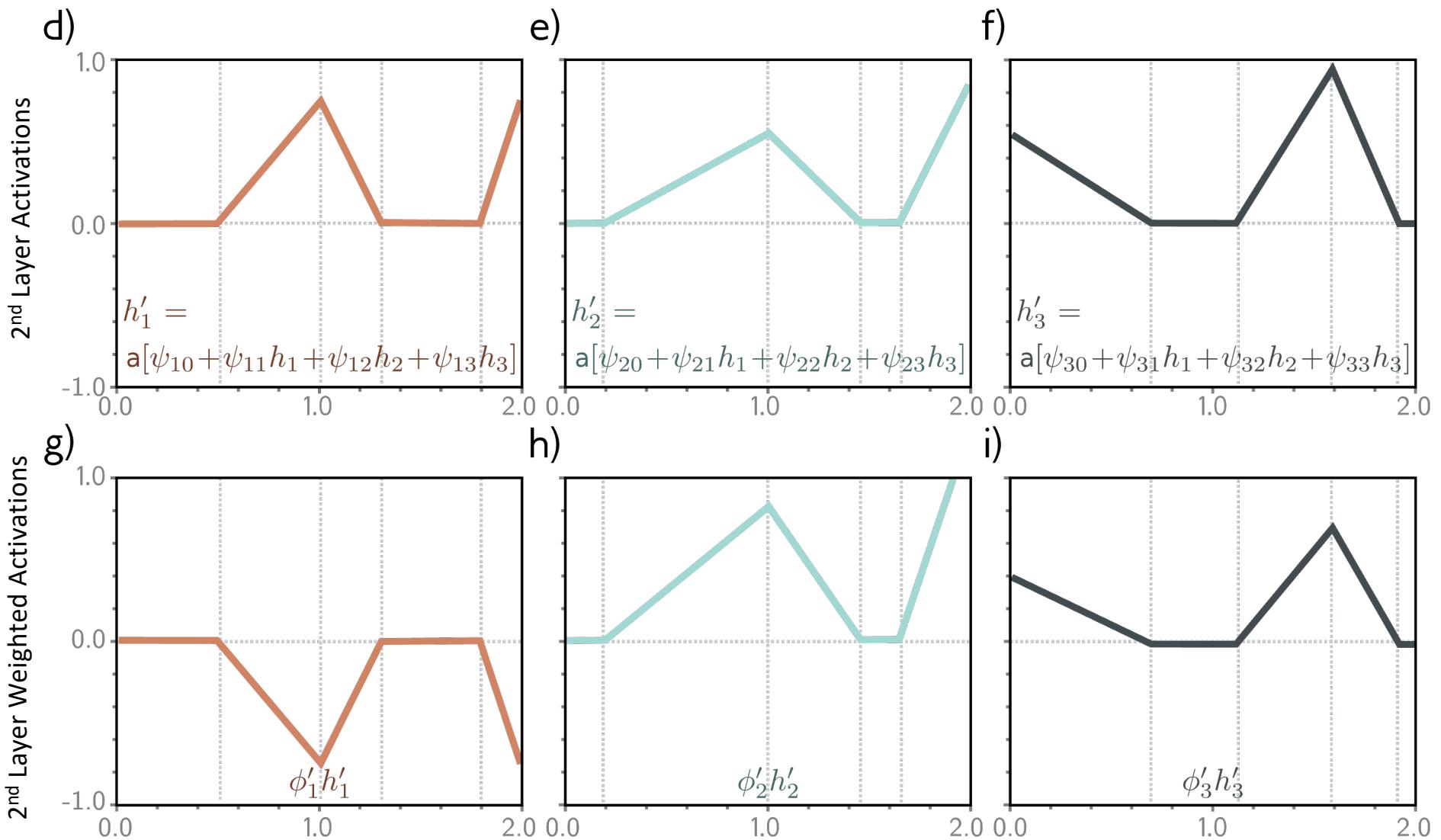


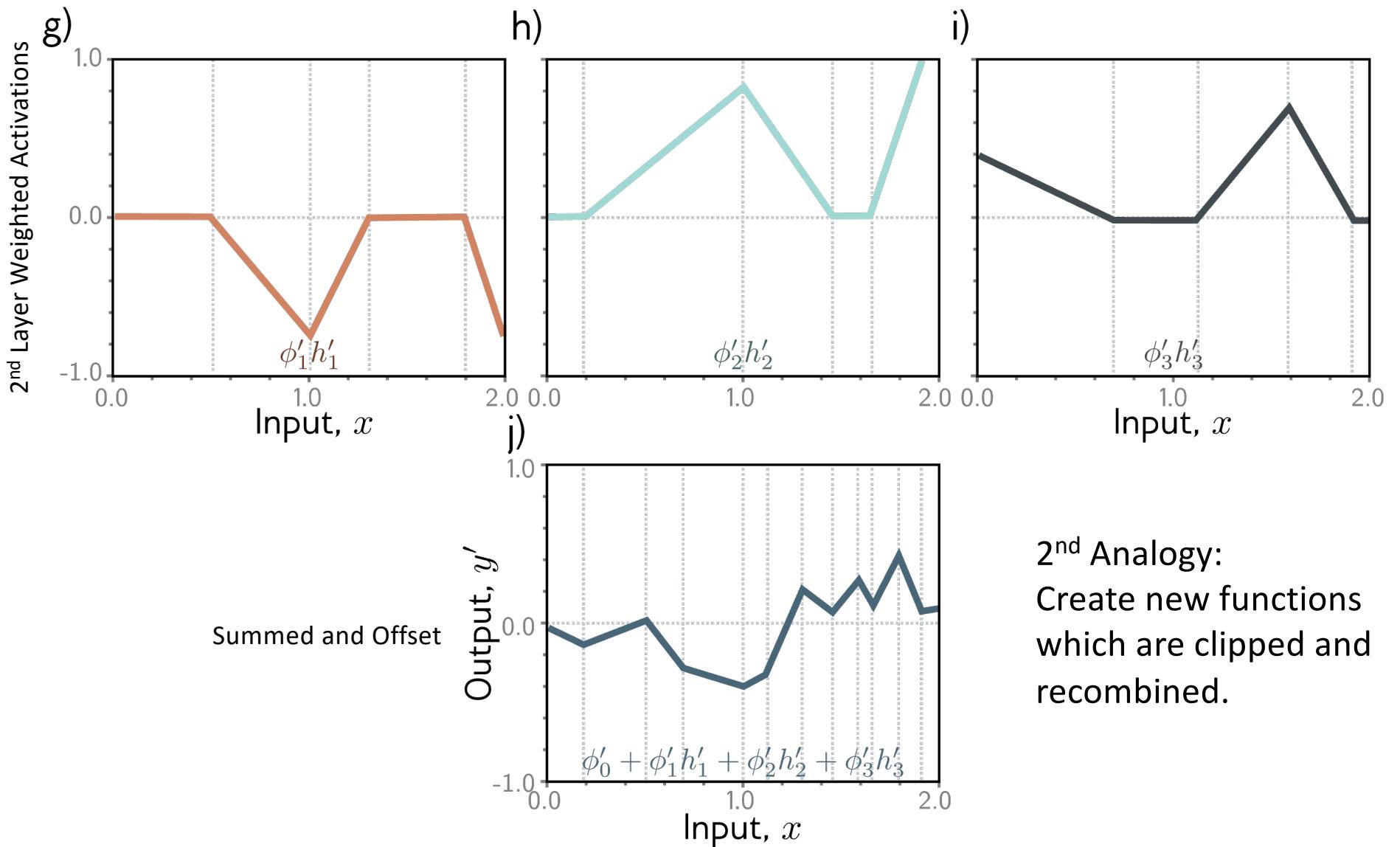


Let's walk through example activations
starting with pre-activations to the 2nd layer.









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

Deep neural networks

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

Hyperparameters

- K layers = **depth of network**
- D_k hidden units per layer = **width of network**
- 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](#))

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

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

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

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

$$h'_1 = a[\psi_{10} + \psi_{11}h_1 + \psi_{12}h_2 + \psi_{13}h_3]$$

$$h'_2 = a[\psi_{20} + \psi_{21}h_1 + \psi_{22}h_2 + \psi_{23}h_3]$$

$$h'_3 = a[\psi_{30} + \psi_{31}h_1 + \psi_{32}h_2 + \psi_{33}h_3]$$

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

Notation change #1

Vector Notation

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

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

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



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

$$h'_1 = \mathbf{a}[\psi_{10} + \psi_{11}h_1 + \psi_{12}h_2 + \psi_{13}h_3]$$

$$h'_2 = \mathbf{a}[\psi_{20} + \psi_{21}h_1 + \psi_{22}h_2 + \psi_{23}h_3]$$

$$h'_3 = \mathbf{a}[\psi_{30} + \psi_{31}h_1 + \psi_{32}h_2 + \psi_{33}h_3]$$

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

Notation change #1

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

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

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



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

Vector Notation

$$h'_1 = a[\psi_{10} + \psi_{11}h_1 + \psi_{12}h_2 + \psi_{13}h_3]$$

$$h'_2 = a[\psi_{20} + \psi_{21}h_1 + \psi_{22}h_2 + \psi_{23}h_3]$$

$$h'_3 = a[\psi_{30} + \psi_{31}h_1 + \psi_{32}h_2 + \psi_{33}h_3]$$



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

Vector & Matrix Notation

Notation change #1

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

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

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



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

$$h'_1 = \mathbf{a}[\psi_{10} + \psi_{11}h_1 + \psi_{12}h_2 + \psi_{13}h_3]$$

$$h'_2 = \mathbf{a}[\psi_{20} + \psi_{21}h_1 + \psi_{22}h_2 + \psi_{23}h_3]$$

$$h'_3 = \mathbf{a}[\psi_{30} + \psi_{31}h_1 + \psi_{32}h_2 + \psi_{33}h_3]$$



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

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



$$y' = \phi'_0 + [\phi'_1 \quad \phi'_2 \quad \phi'_3] \begin{bmatrix} h'_1 \\ h'_2 \\ h'_3 \end{bmatrix}$$

Notation change #2

Notation Reminder

x, ψ : normal lower case -- scalar

$\mathbf{x}, \boldsymbol{\psi}$: bold face lower case -- vector

$X, \boldsymbol{\Psi}$: bold face upper case -- matrix

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

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

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

ω : omega
 Ω : Omega

Notation change #3

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

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

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

Notation change #3

$$\mathbf{h} = \mathbf{a} [\boldsymbol{\theta}_0 + \boldsymbol{\theta}x]$$



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

$$\mathbf{h}_2 = \mathbf{a}[\boldsymbol{\beta}_1 + \boldsymbol{\Omega}_1 \mathbf{h}_1]$$

$$y = \boldsymbol{\phi}'_0 + \boldsymbol{\phi}' \mathbf{h}'$$



$$\mathbf{y} = \boldsymbol{\beta}_2 + \boldsymbol{\Omega}_2 \mathbf{h}_2$$

ω : omega
 Ω : Omega

Bias
vector

Weight
matrix

General equations for deep network

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

$$\mathbf{h}_2 = \mathbf{a}[\boldsymbol{\beta}_1 + \boldsymbol{\Omega}_1 \mathbf{h}_1]$$

$$\mathbf{h}_3 = \mathbf{a}[\boldsymbol{\beta}_2 + \boldsymbol{\Omega}_2 \mathbf{h}_2]$$

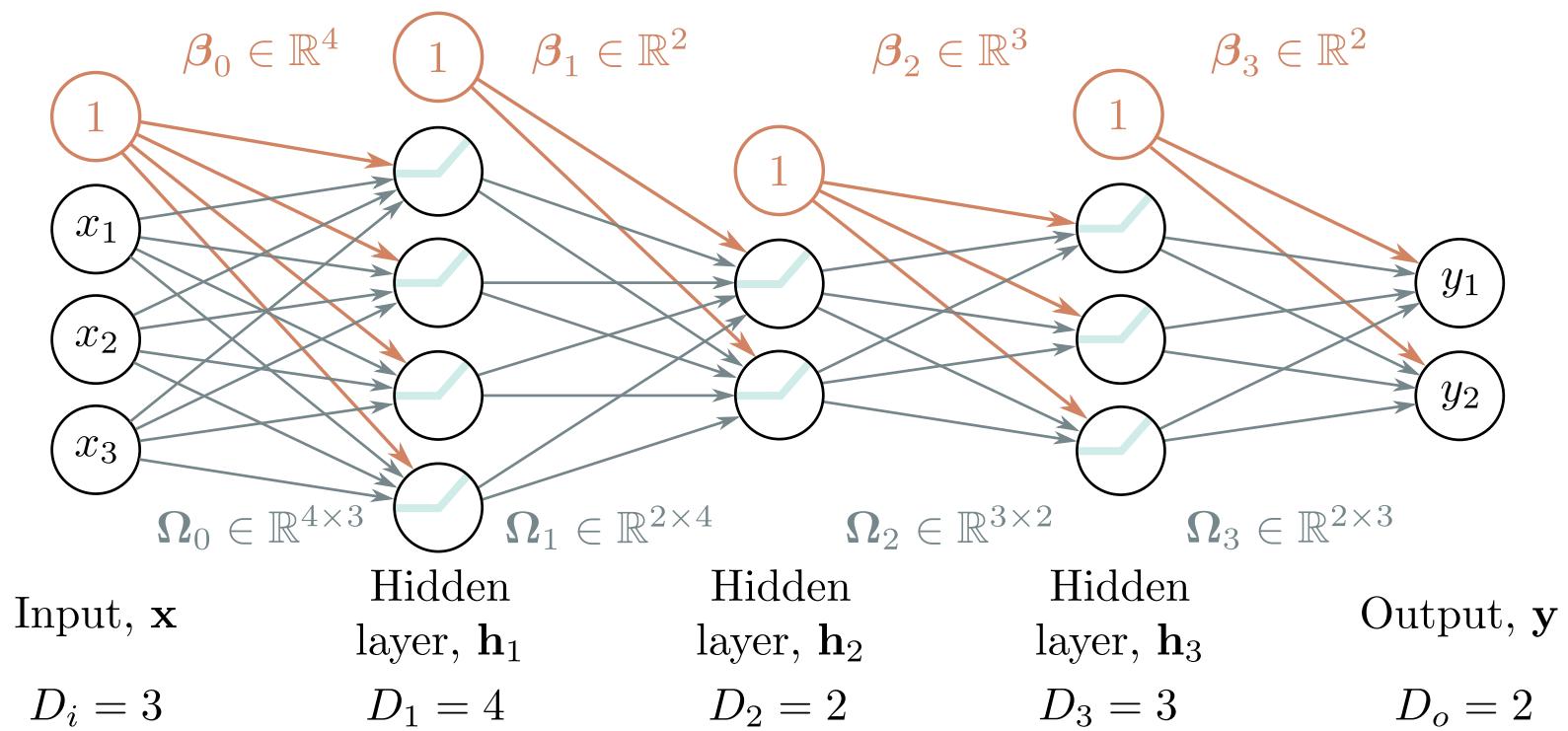
⋮

$$\mathbf{h}_K = \mathbf{a}[\boldsymbol{\beta}_{K-1} + \boldsymbol{\Omega}_{K-1} \mathbf{h}_{K-1}]$$

$$\mathbf{y} = \boldsymbol{\beta}_K + \boldsymbol{\Omega}_K \mathbf{h}_K,$$

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

Example



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?

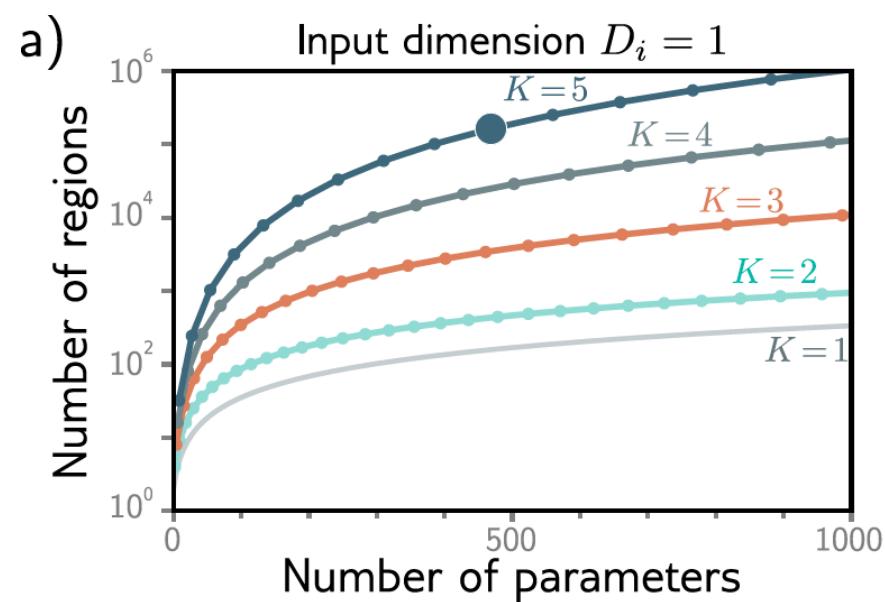
Both obey the universal approximation theorem.

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

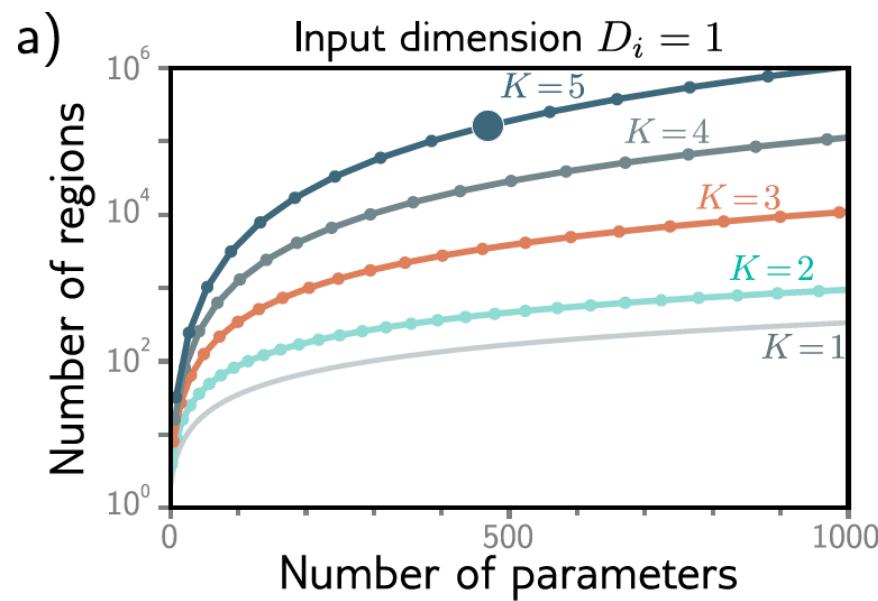
Number of linear regions per parameter



Each small dot is
an additional
hidden unit per
layer.

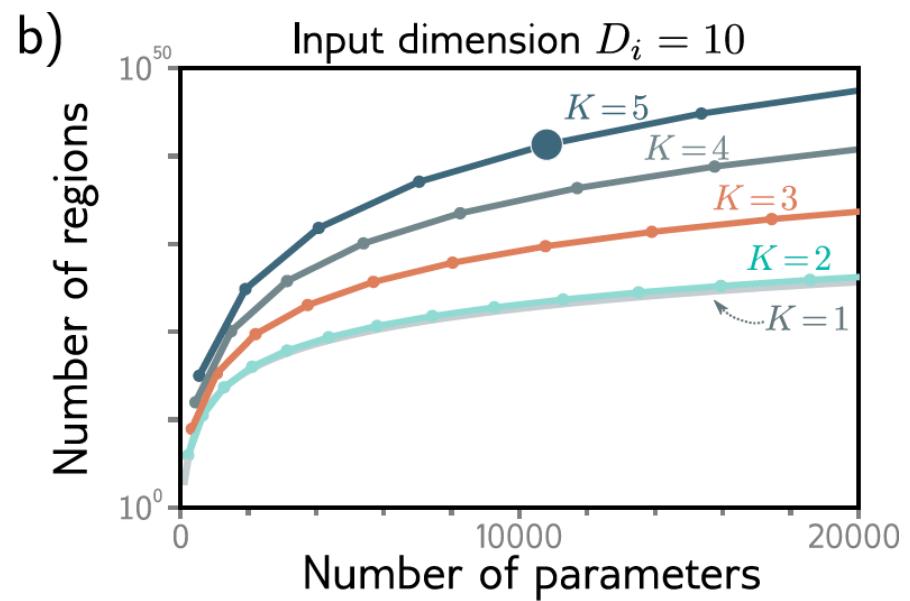
- $K = 5$ layers
- 10 hidden units per layer
- 471 parameters
- 161,501 linear regions

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

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

Shallow vs. Deep Networks

3. Depth efficiency

- 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.

Shallow vs. Deep Networks

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

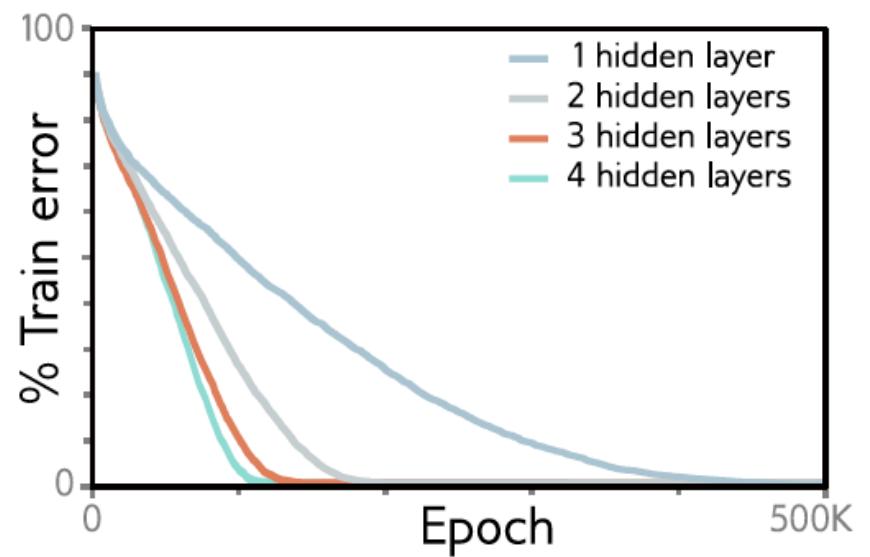
5. Fitting and generalization

- 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

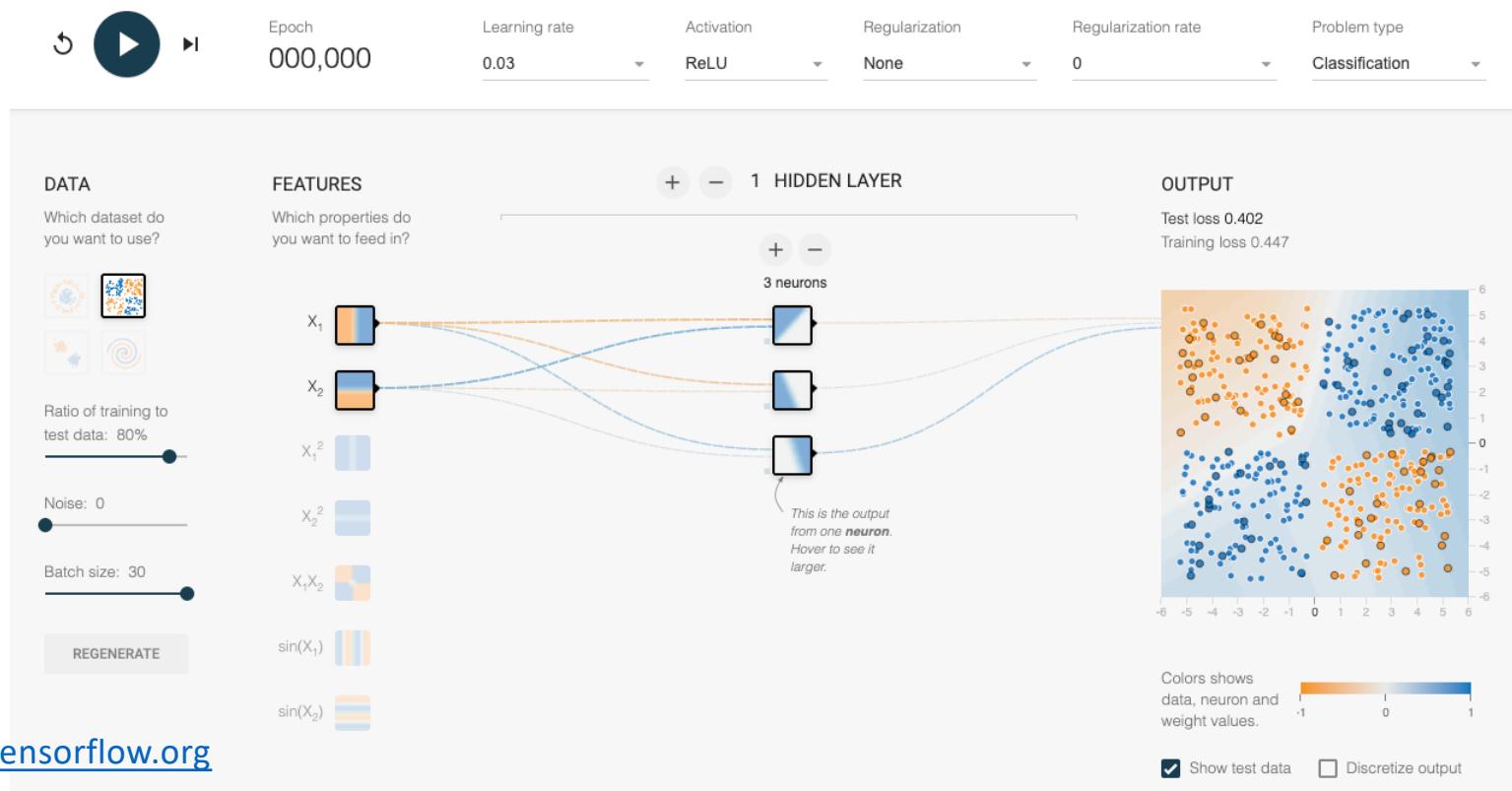
Figure 20.2 MNIST-1D training. Four fully connected networks were fit to 4000 MNIST-1D examples with random labels using full batch gradient descent, He initialization, no momentum or regularization, and learning rate 0.0025. Models with 1,2,3,4 layers had 298, 100, 75, and 63 hidden units per layer 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?

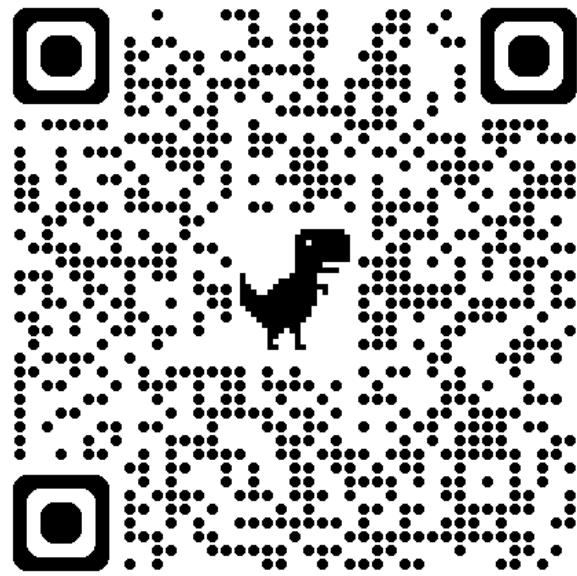


playground.tensorflow.org

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 (**Read Ch. 5**)
 - How to find minima of the loss function
 - How to do this efficiently with deep networks
- Then how do we evaluate them?

Feedback?



[Link](#)