

Supervised Learning: Neural networks and Deep learning — Part I

Jean Feng & Ali Shojaie

July 28-30, 2025
Summer Institute in Statistics for Big Data
University of Washington

Outline

1. What is deep learning?
2. What can DL do and not do
3. Introduction to:
 1. Dense neural networks
 2. Convolutional neural networks
4. How to get started using neural networks

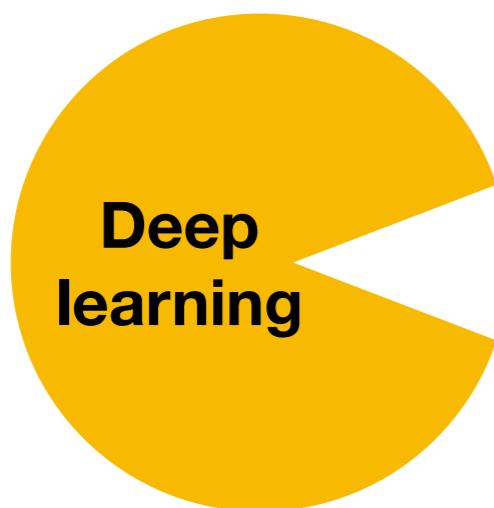
Outline

1. What is deep learning?
2. What can DL do and not do
3. Introduction to:
 1. Dense neural networks
 2. Convolutional neural networks
4. How to get started using neural networks

What is deep learning?

“The grandly named statistical technique, put simply, gives computers a way to learn by processing vast amounts of data... Deep learning algorithms train on a batch of related data — like pictures of human faces — and are then fed more and more data, which steadily improve the software’s pattern-matching accuracy.”

- The New York Times, “Is there a smarter path to artificial intelligence? Some experts hope so.” June 20, 2018.

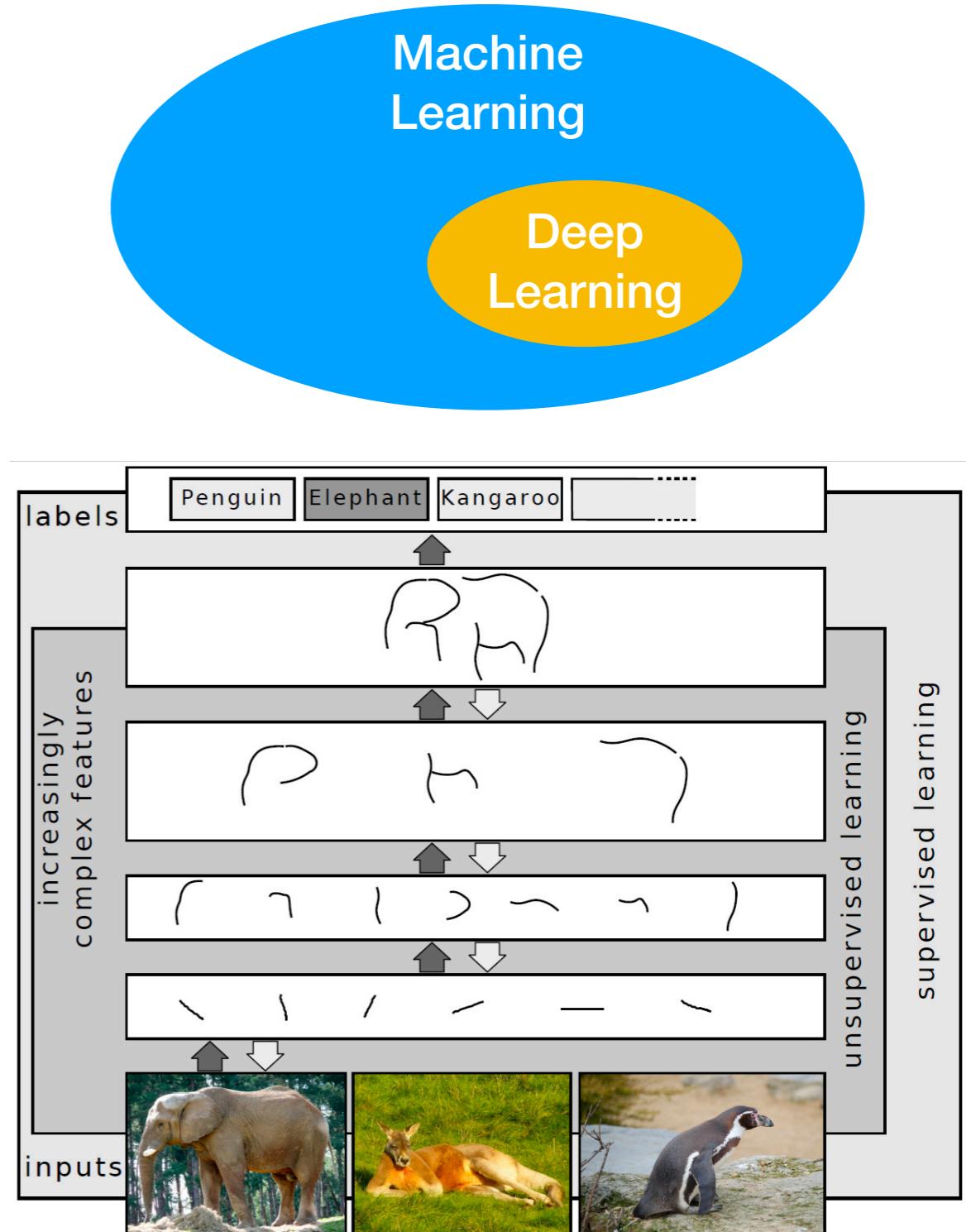


- **Data • • • Data • • • Data • • • Data • •**

What is deep learning?

“Deep learning is part of a broader family of machine learning methods based on artificial neural networks with representation learning... Deep learning is a class of machine learning algorithms that uses multiple layers to progressively extract higher level features from the raw input.”

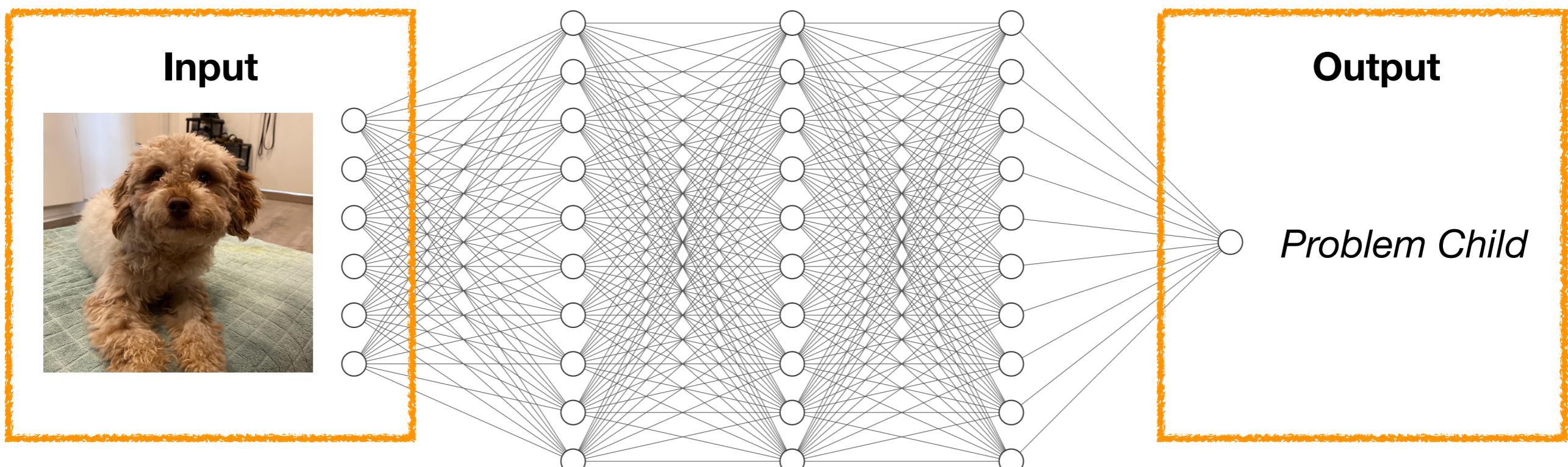
— Wikipedia



What is deep learning?

Deep learning (DL) is a non-parametric machine learning method that builds large computational graphs called neural networks to model complex nonlinearities, interactions, and hierarchies in the data.

— A statistical perspective of DL

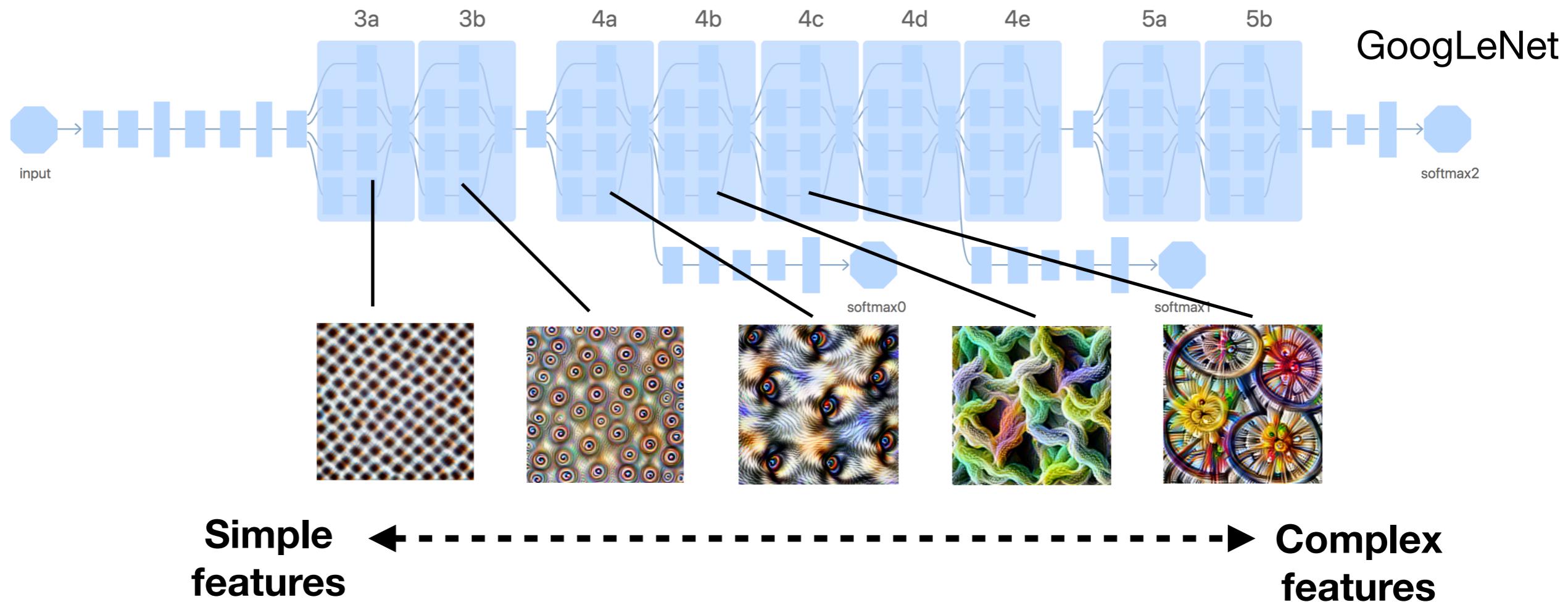


Strengths of deep learning

- **Universal Approximation Theorem:** Given any continuous function, there exists a sufficiently large neural network that can approximate it within any positive margin of error (Cybenko 1989, Hornik 1991, Leshno 1993).
- **Memory efficient:** Neural networks are able to approximate complex functions using a relatively small number of parameters.
- **Computationally efficient:** Information can be propagated efficiently through neural networks, so training them is (reasonably) fast.

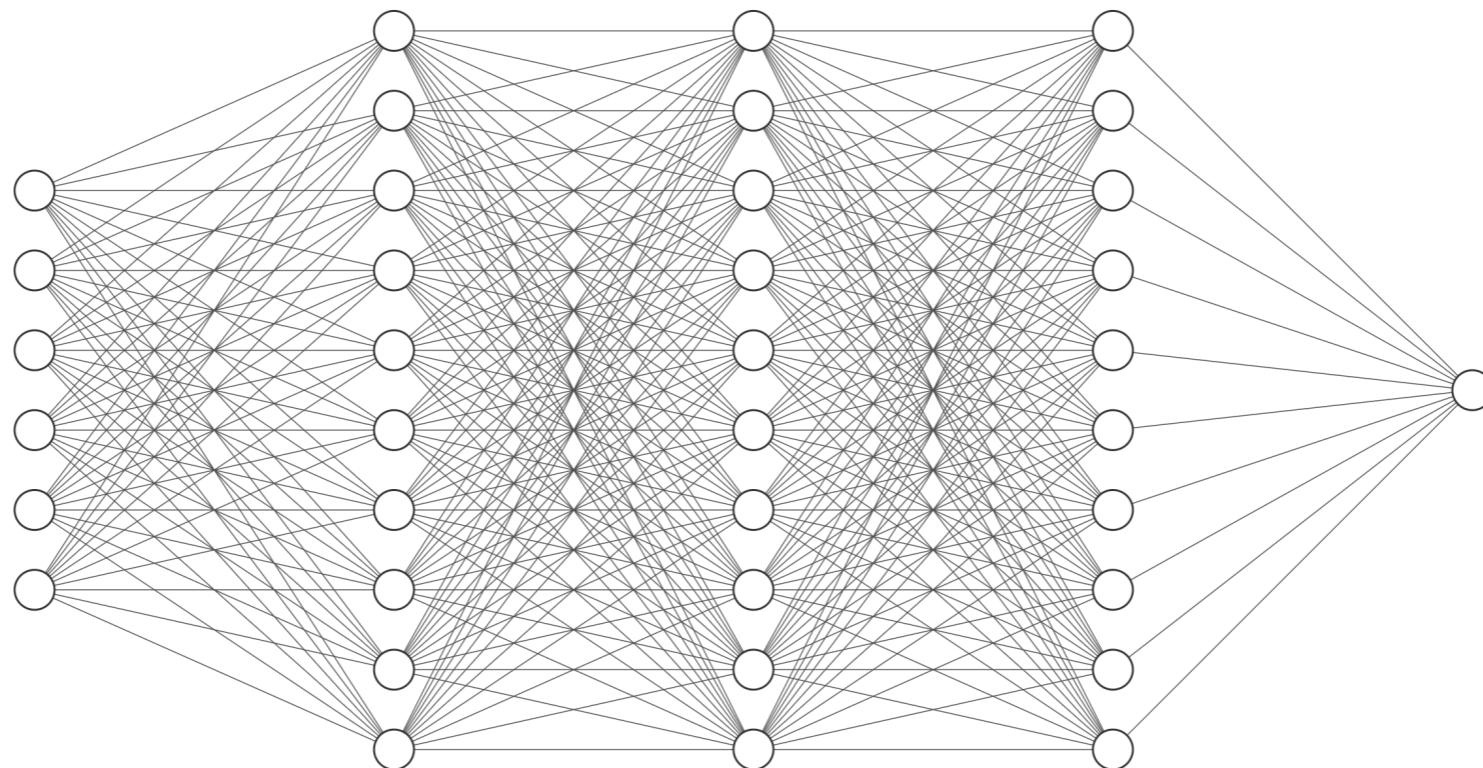
Strengths of deep learning

- **Feature/Representational learning:** Neural networks iteratively apply non-linear transformations to the data to gradually build up increasingly abstract representations.



Strengths of deep learning

- **Highly modular:** You can add nodes, layers, modules with different properties to the neural network. Software tools freely available online make this very easy.



Outline

1. What is deep learning?
2. **What can DL do and not do**
3. Introduction to:
 1. Dense neural networks
 2. Convolutional neural networks
4. How to get started using neural networks

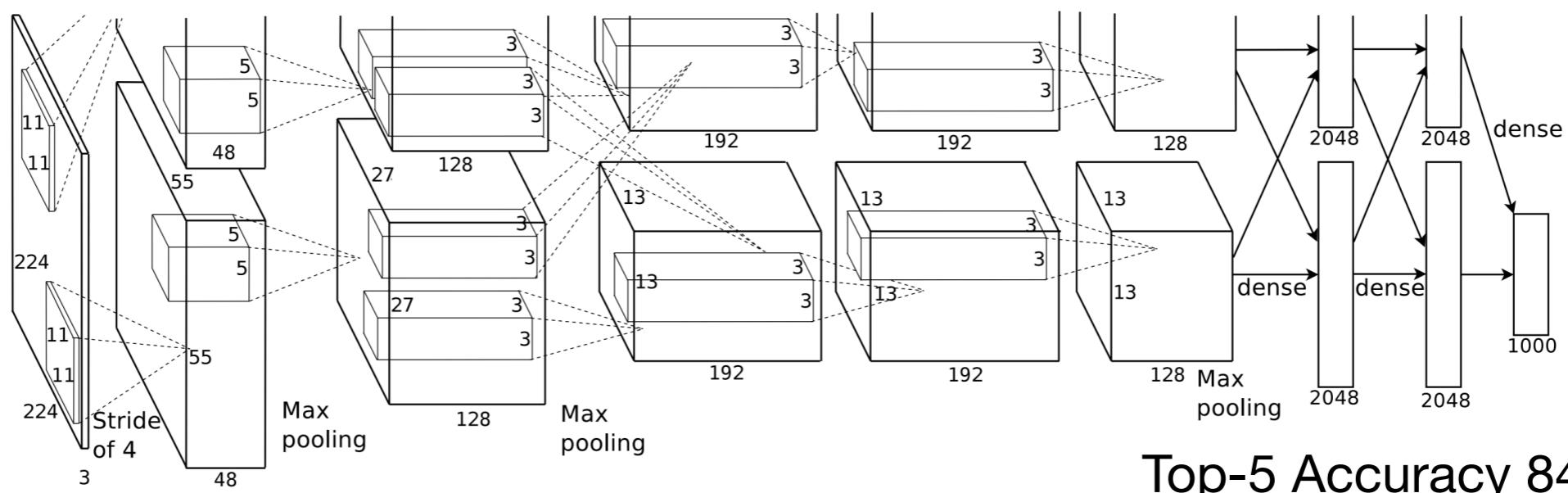
Supervised learning

- **Task:** ImageNet Large Scale Visual Recognition Challenge (ILSVRC)
 - 1000 classes with roughly 1000 images per class
- **A more traditional approach:** Apply well-studied image feature extraction methods and train a classifier on top of this



Supervised learning

- **Task:** ImageNet Large Scale Visual Recognition Challenge (ILSVRC)
 - 1000 classes with roughly 1000 images per class
- **DL Method:** AlexNet (Krizhevsky 2012): 8-layer convolutional neural network with 60 million parameters

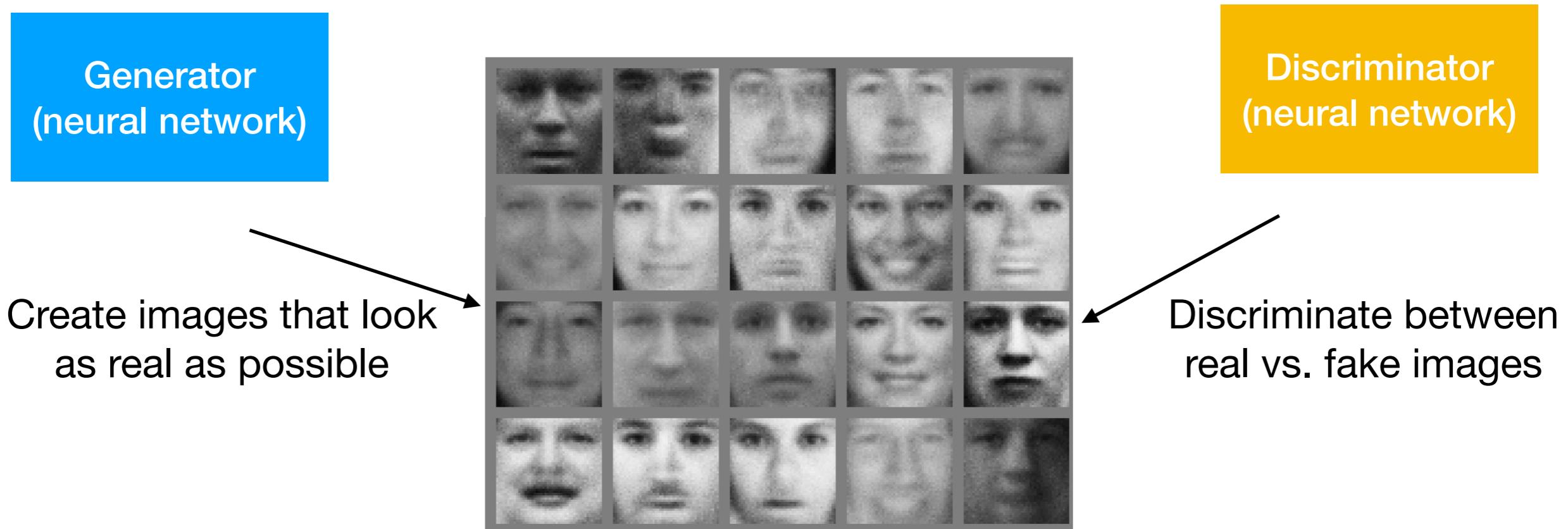


Unsupervised learning

- **Task:** Generate images of people
- **A more traditional approach:** Compose parametric models that learns the probability distribution of images

Unsupervised learning

- **Task:** Generate images of people
- **DL Method:** Generative Adversarial Networks
(Goodfellow 2014): Simultaneously trains two models from the perspective of a two-player game

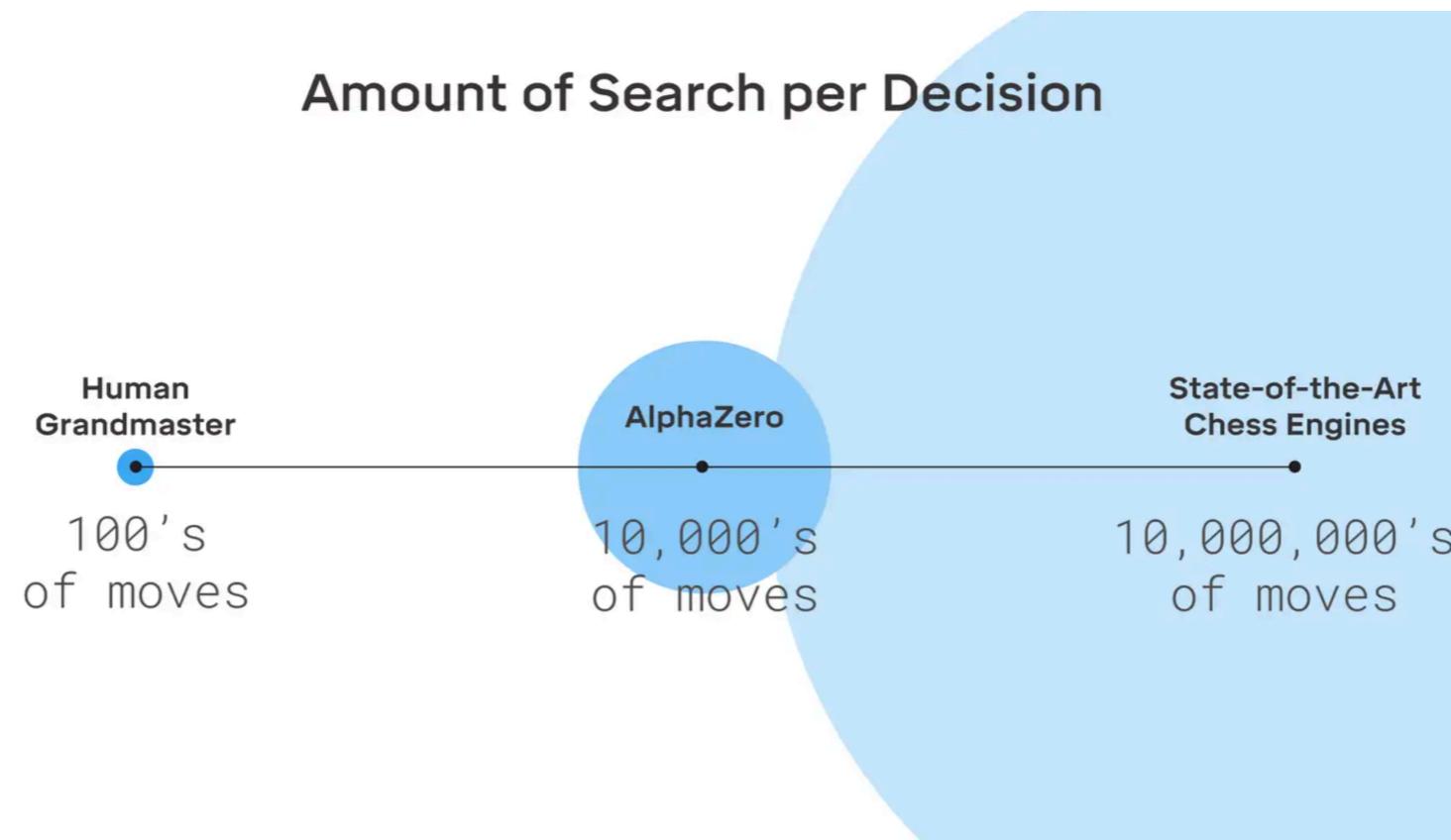


Reinforcement learning

- **Task:** Teach a machine to play chess
- **A more traditional approach:** List out a subset of the possible move sequences and pick the best one. Rely on rules and heuristics handcrafted by experts.

Reinforcement learning

- **Task:** Teach a machine to play chess
- **DL method:** AlphaZero (Silver 2018): A Monte Carlo tree search guided by a neural network. Learns the most promising moves by playing millions of games

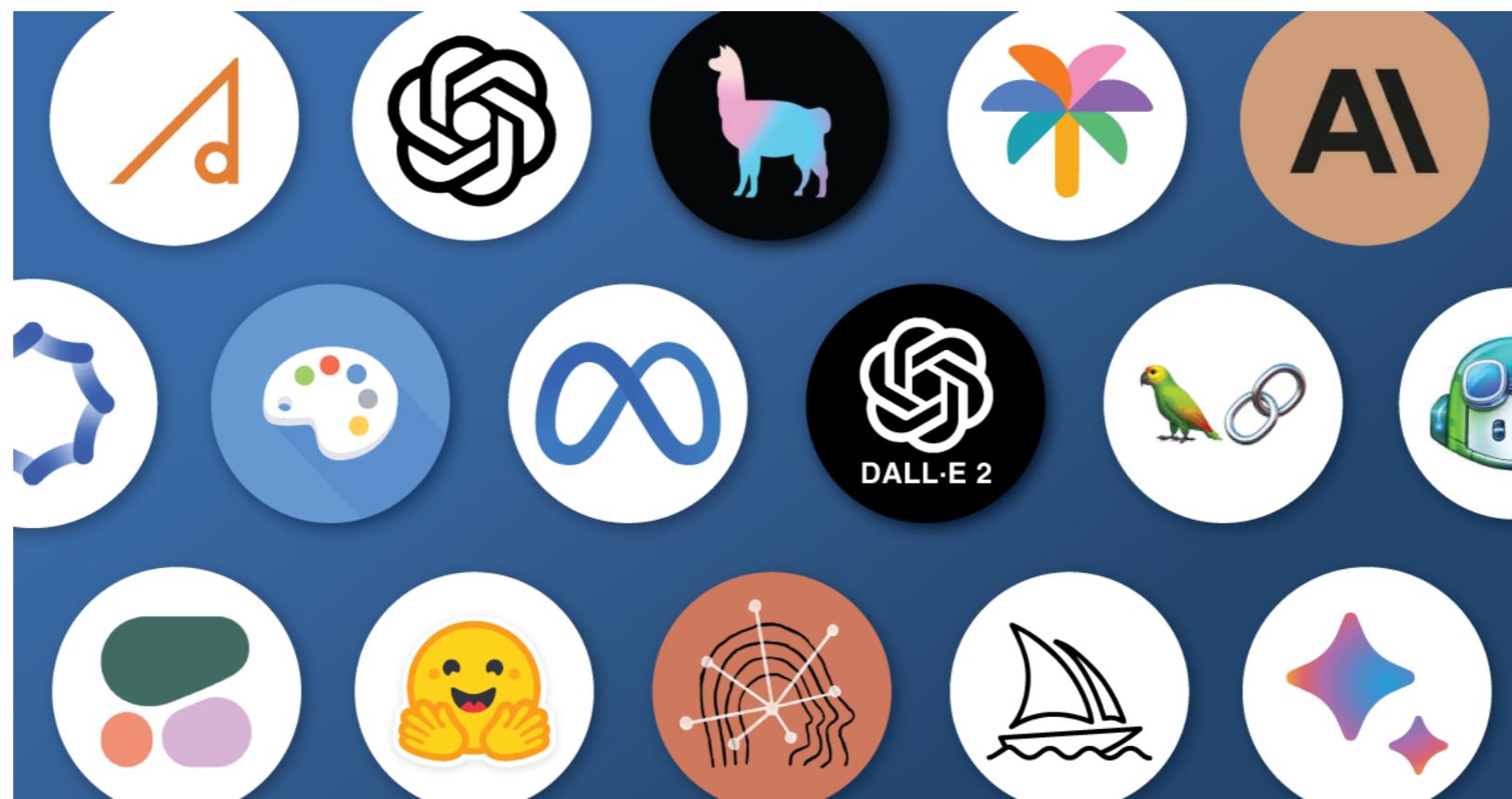


Chatbots

- **Task:** Build a chatbot
- **A more traditional approach:** Hardcode a chatbot.

Chatbots

- **Task:** Build a chatbot
- **DL Method:**



ChatGPT

this code is not working like i expect — how do i fix it?

```
resultWorkerErr := make(chan error)
defer close(resultWorkerErr)
go func() {
    defer cancel()
    resultWorkerErr <- b.resultWorker(ctx)
}()

err := b.worker(ctx)
cancel()
if err == nil {
    return <-resultWorkerErr
}
return multierror.Append(err, <-resultWorkerErr)
```

User

ChatGPT

It's difficult to say what's wrong with the code without more context. Can you provide more information about what the code is supposed to do and what isn't working as expected? Also, is this the entire code or just a part of it?

User

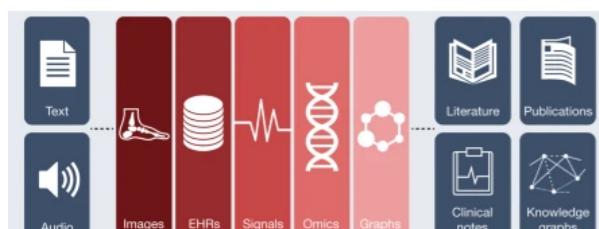
ChatGPT

just part of it — the error never surfaces. I think there is something wrong with the channel
It's difficult to say without more information about what the code is supposed to do and what's happening when it's executed. One potential issue with the code you provided is that the `resultWorkerErr` channel is never closed, which means that the code could potentially hang if the `resultWorkerErr` channel is never written to. This could happen if `b.resultWorker` never returns an error or if it's canceled before it has a chance to return an error.

Which of the following techniques are used to train ChatGPT?

- A: Supervised learning
- B: Unsupervised learning
- C: Reinforcement learning

Step 0: Pre-training (Unsupervised)

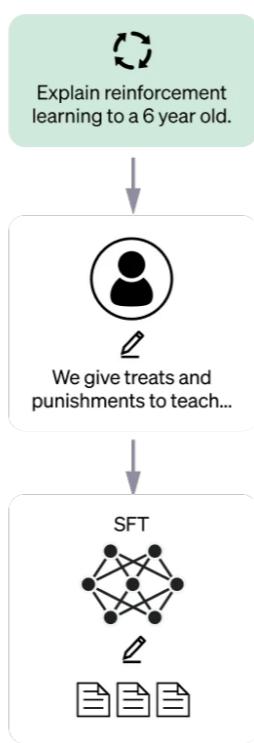


**Self-supervised learning tasks,
e.g. predict the next word**

Step 1

Collect demonstration data and train a supervised policy.

A prompt is sample from our prompt dataset.



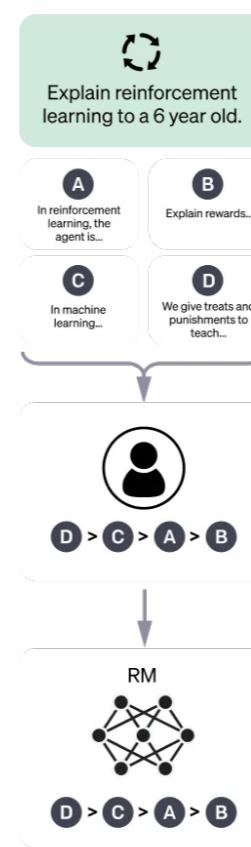
A labeler demonstrates the desired output behavior.

This data is used to fine-tune GPT-3.5 with supervised learning.

Step 2

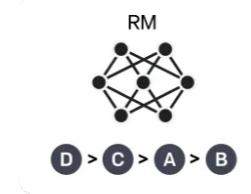
Collect comparison data and train a reward model.

A prompt and several model outputs are sampled.



A labeler ranks the outputs from best to worst.

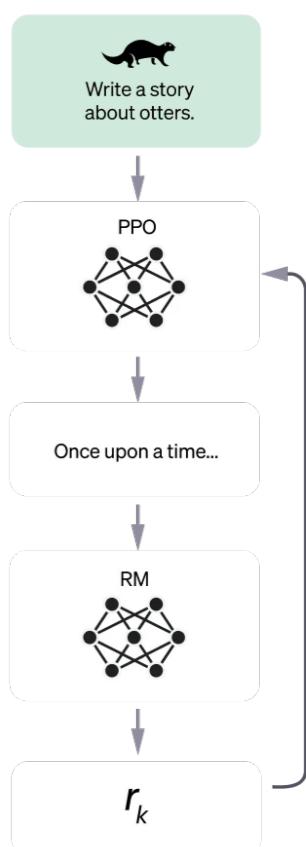
This data is used to train our reward model.



Step 3

Optimize a policy against the reward model using the PPO reinforcement learning algorithm.

A new prompt is sampled from the dataset.

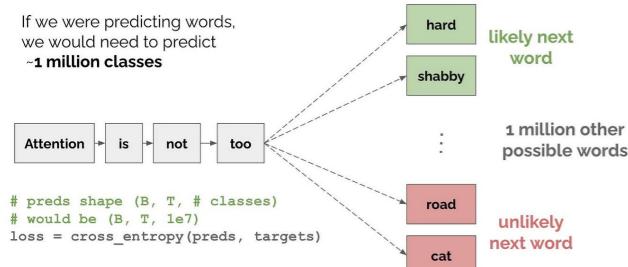


The PPO model is initialized from the supervised policy.

The policy generates an output.

The reward model calculates a reward for the output.

The reward is used to update the policy using PPO.



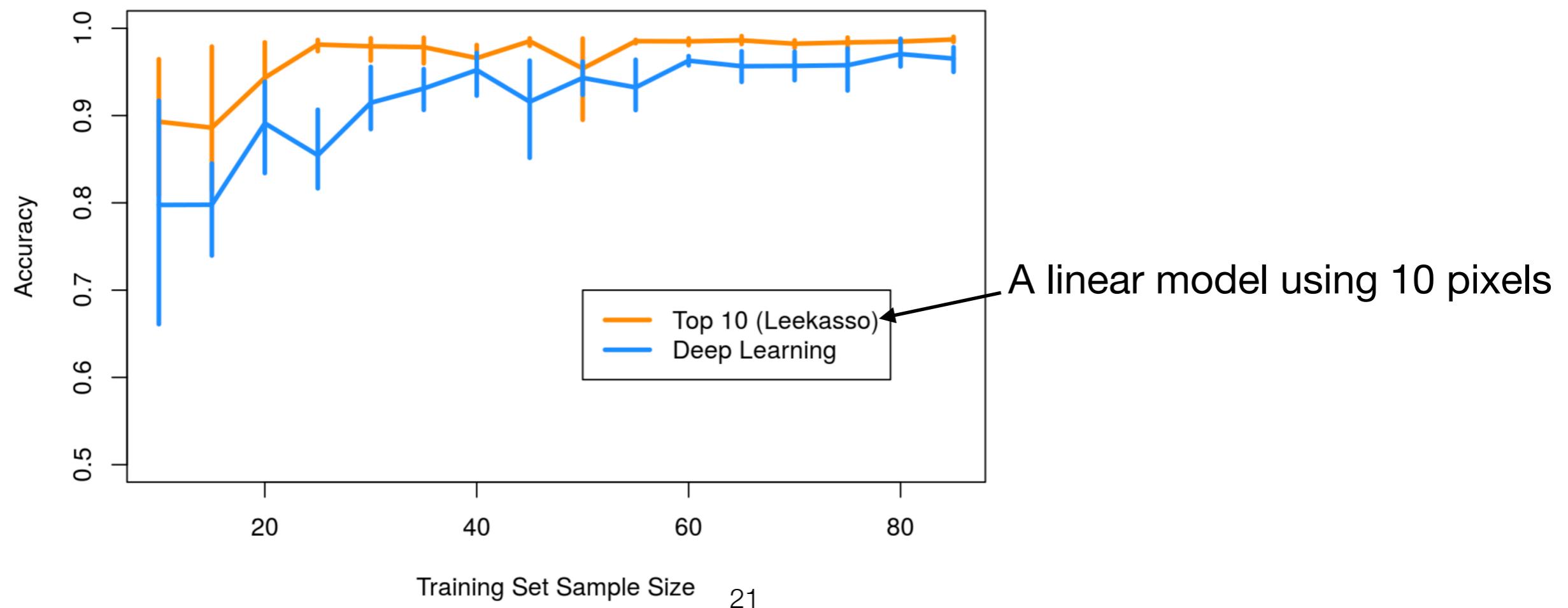
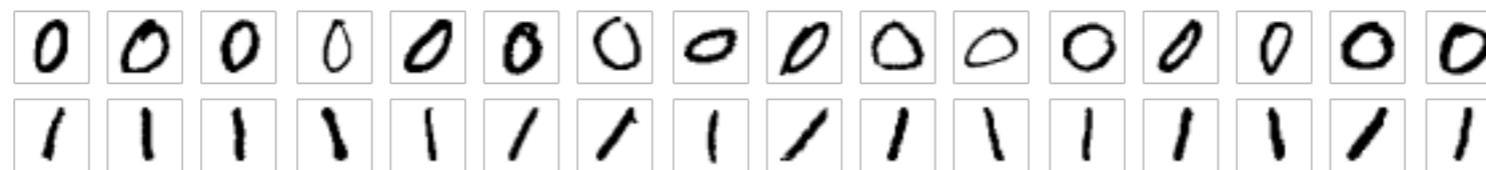
<https://openai.com/index/chatgpt/>

Where deep learning struggles

“Don’t use deep learning - your data isn’t that big”

— Jeff Leek, [simplystats](#)

Example: Predict 0 vs 1 in the image



Deep learning tends to struggle when...

- High-dimensional settings where the number of observations n is "small" compared to the number of covariates p
- Signal-to-noise ratio is low
- You need an interpretable model



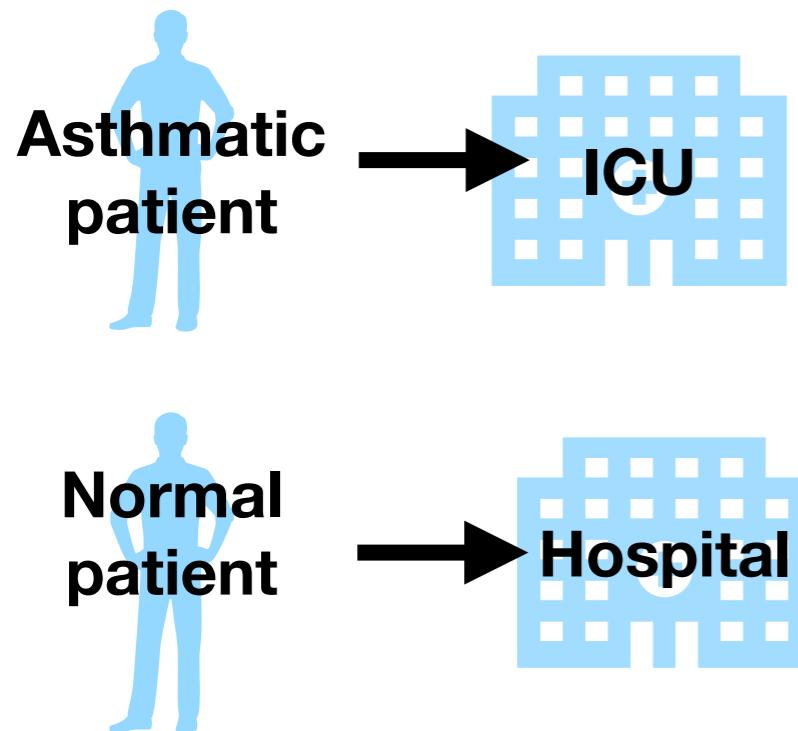
Bias/Variance tradeoff!

In these cases, try using a simpler methods like linear/logistic/cox regression or off-the-shelf machine learning algorithms like random forests.

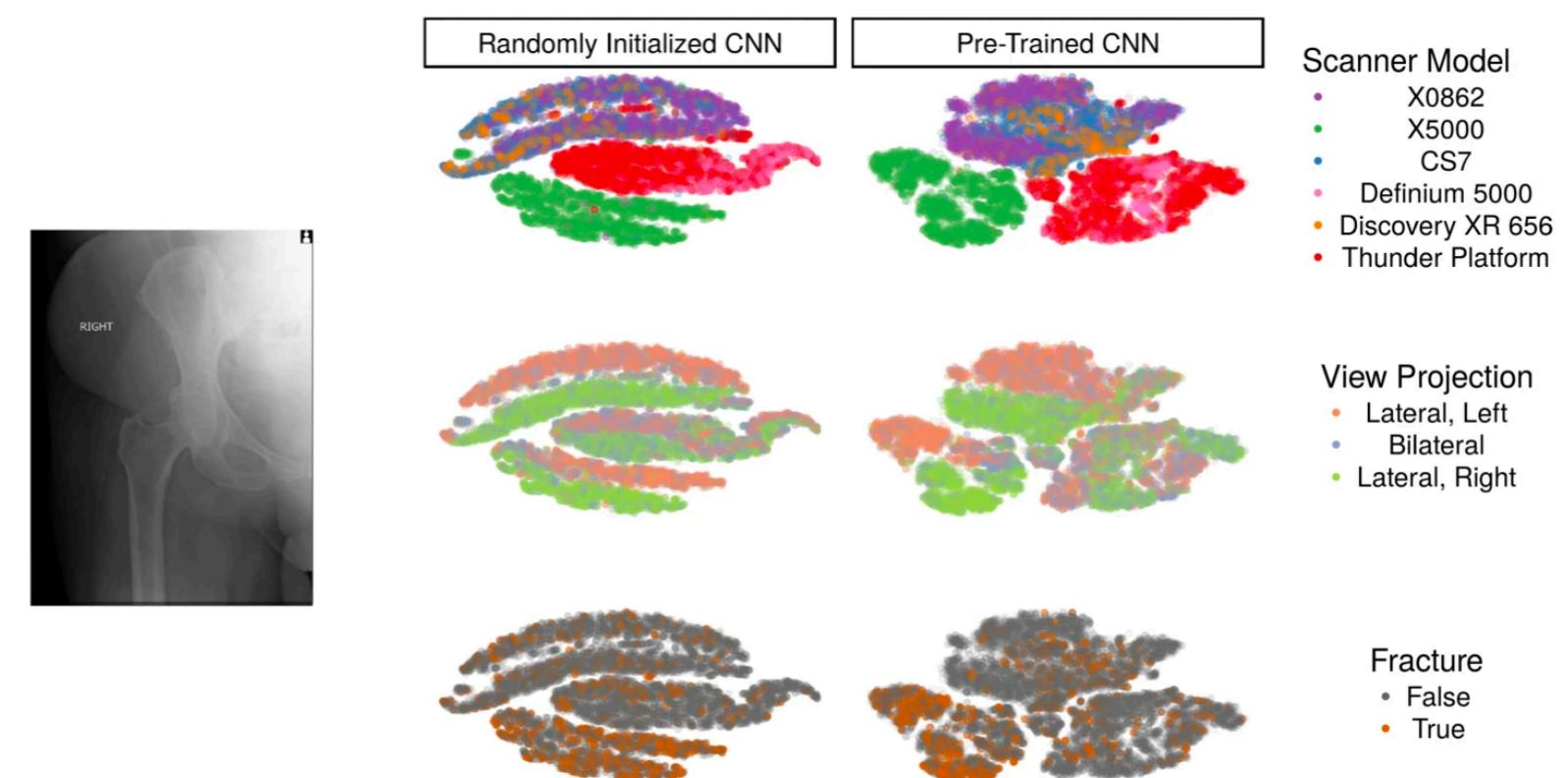
Deep learning tends to struggle when...

- Deep learning will *not* correct biases in the data! It is *very* sensitive to any confounders in your data

Caruana 2015: DL predicted that asthmatics are less likely to die from pneumonia



Badgeley 2019: DL made predictions by detecting the scanner model from the radiograph



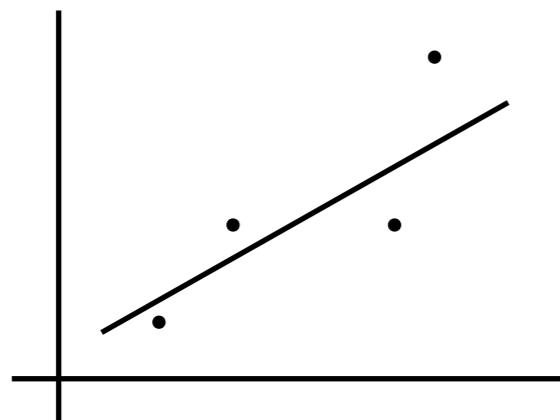
Outline

1. What is deep learning?
2. What can DL do and not do
3. Introduction to:
 1. **Dense neural networks**
 2. Convolutional neural networks
4. How to get started using neural networks

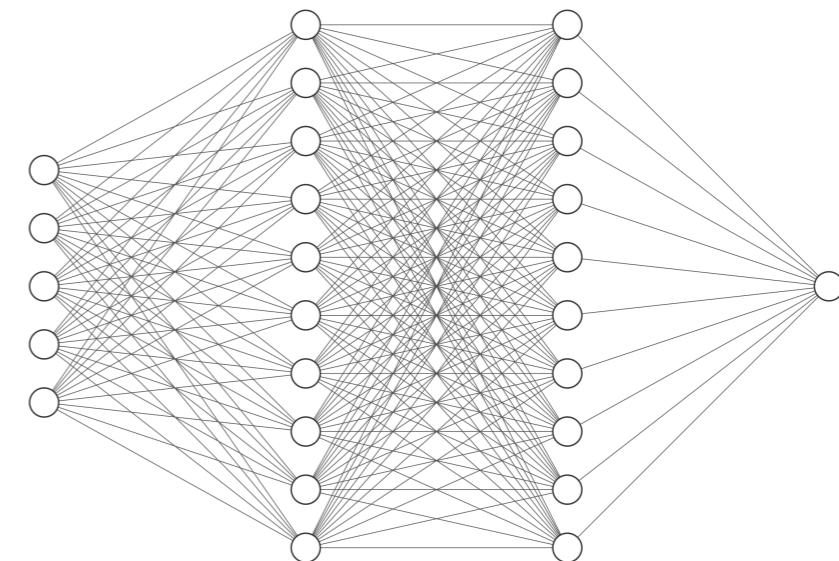
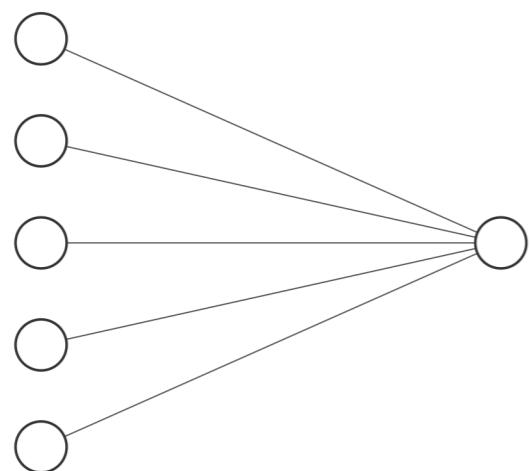
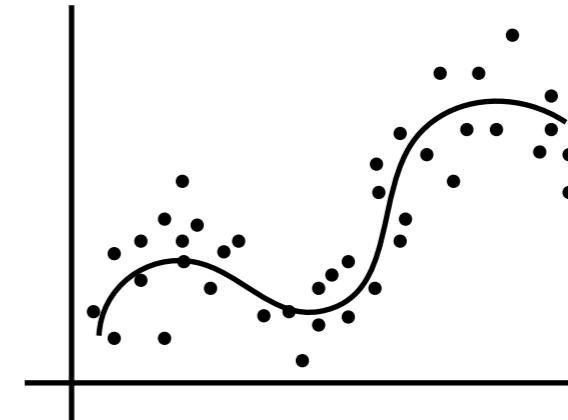
Neural networks

- Neural networks span a wide range of models:

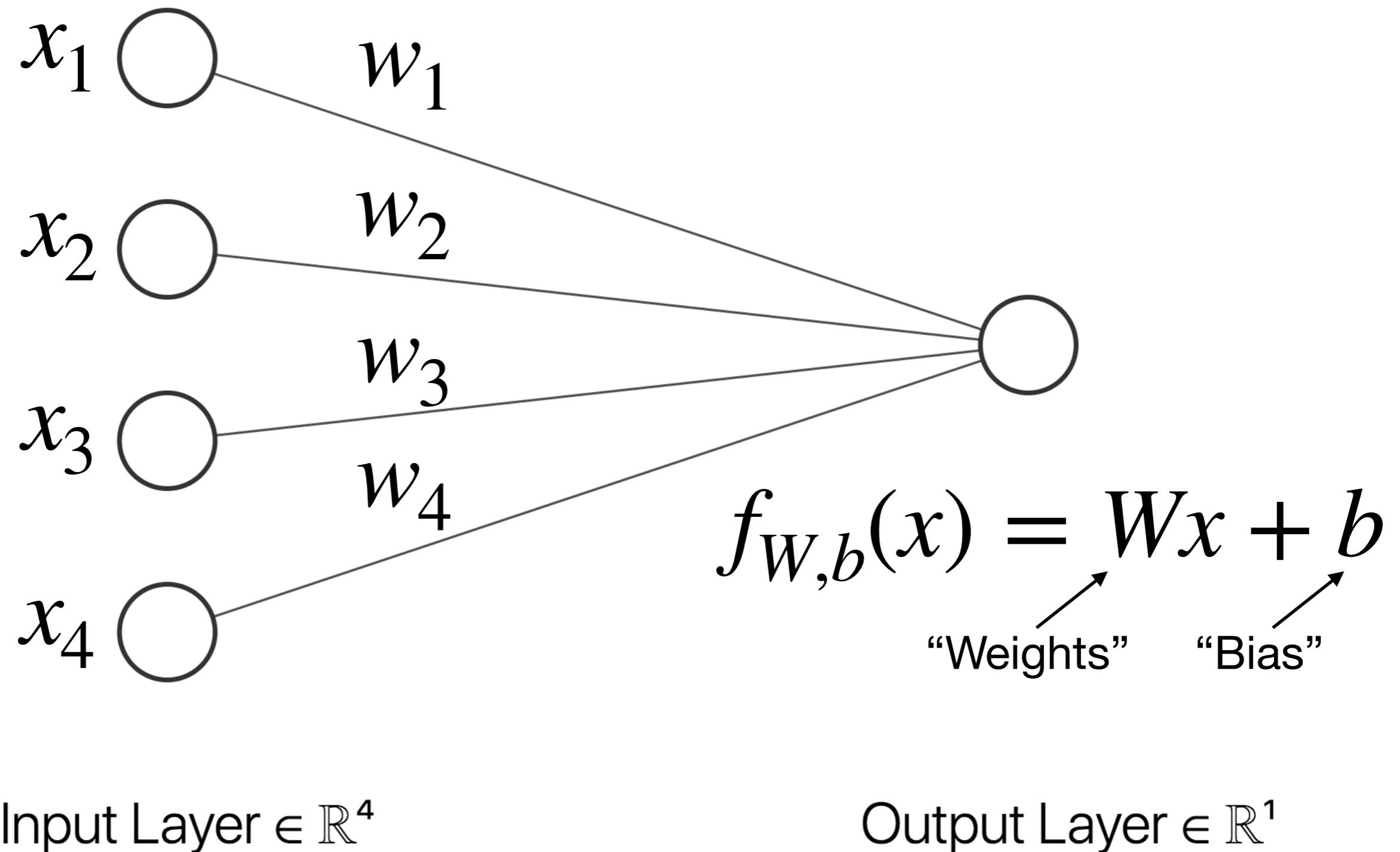
Linear model



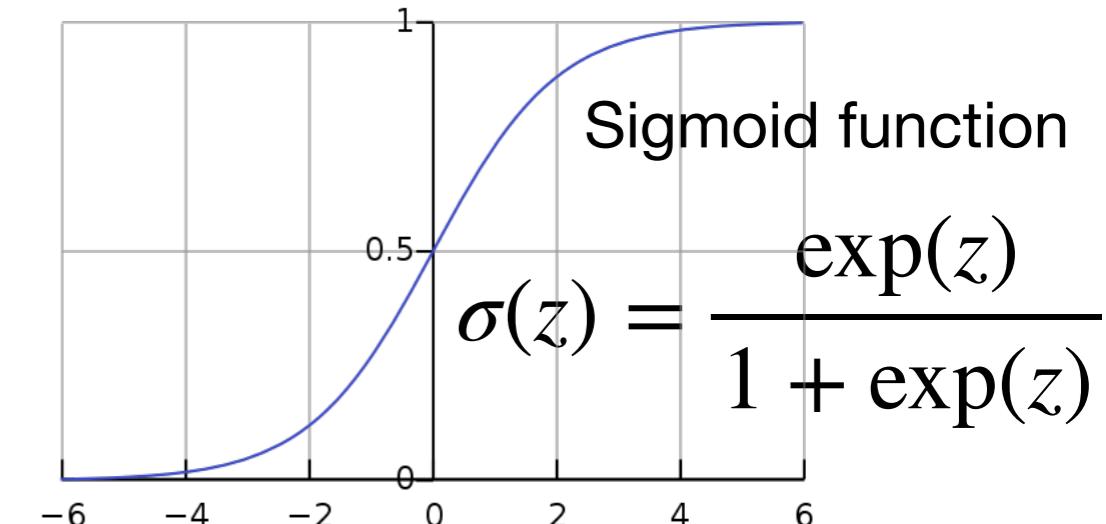
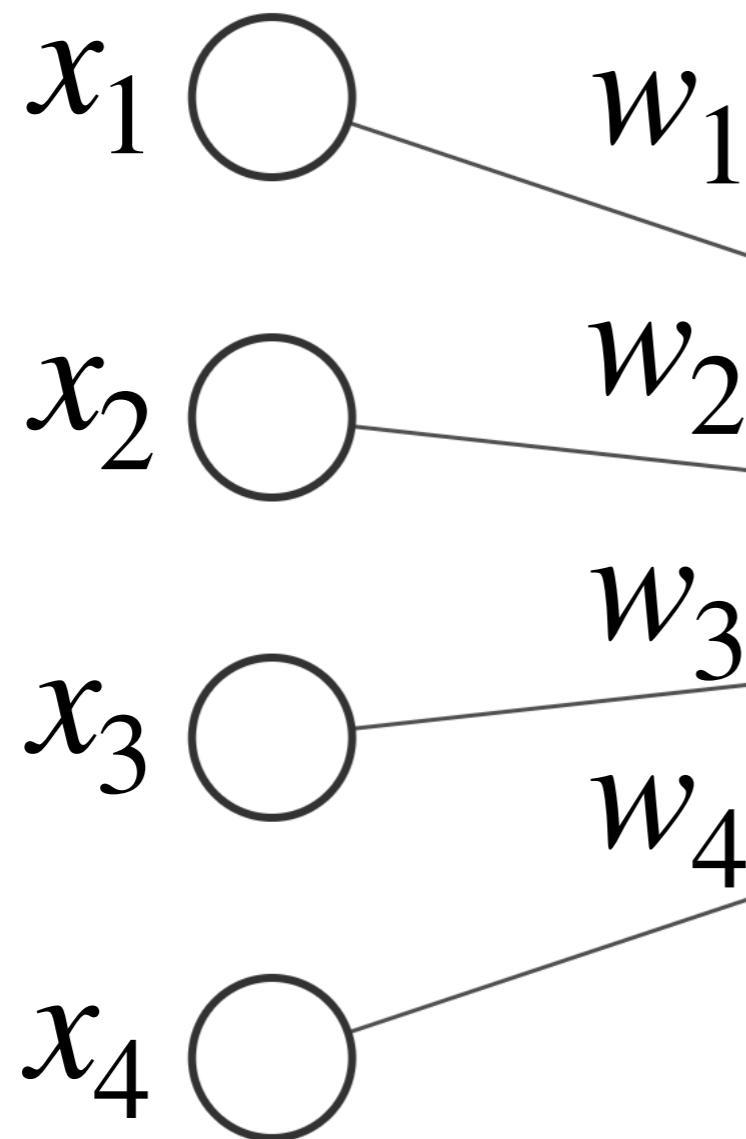
Nonlinear model with complex interactions



Linear regression as a neural network



Logistic regression as a neural network



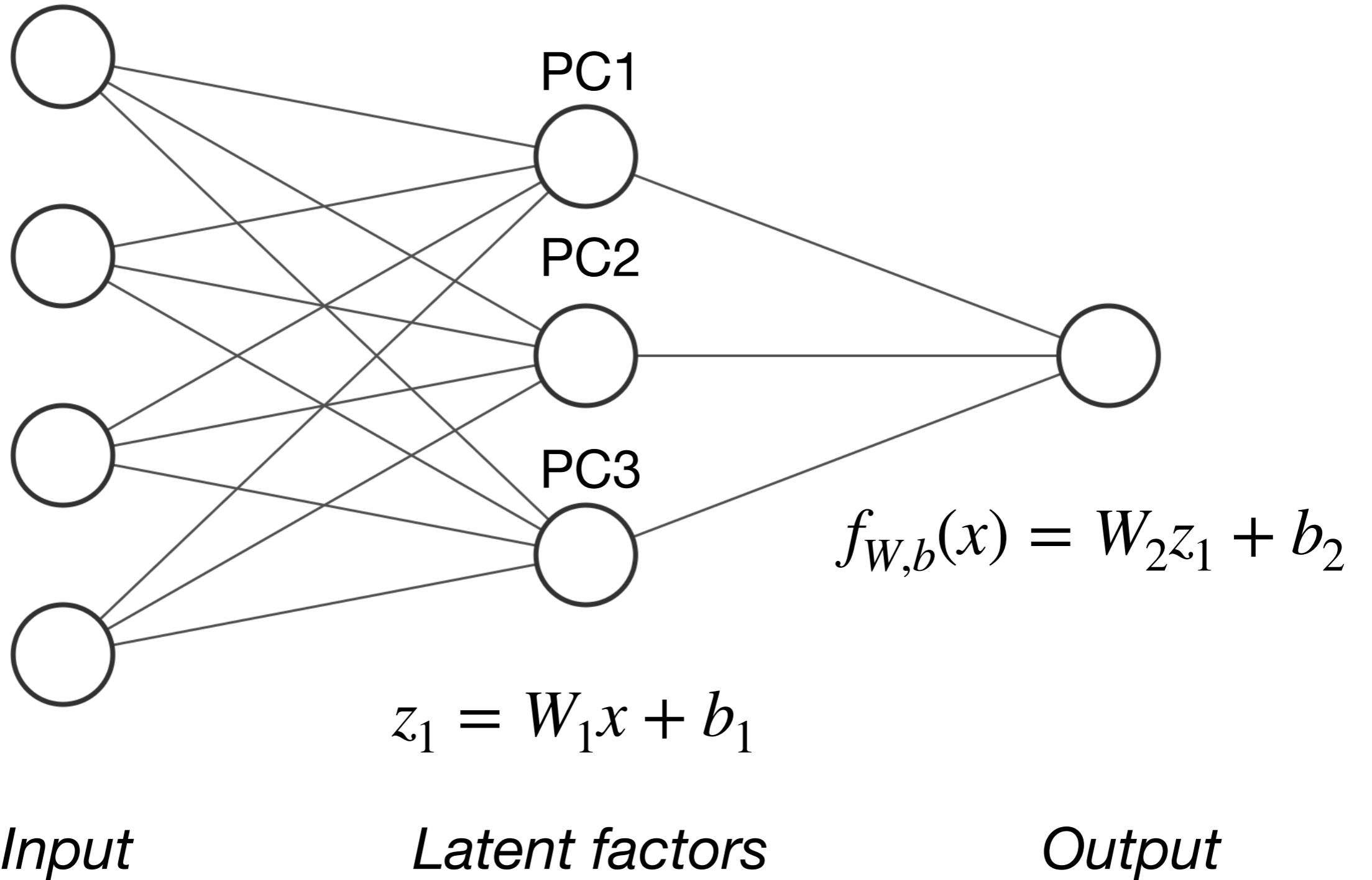
$$f_{W,b}(x) = \sigma(Wx + b)$$

“Activation function”

Input Layer $\in \mathbb{R}^4$

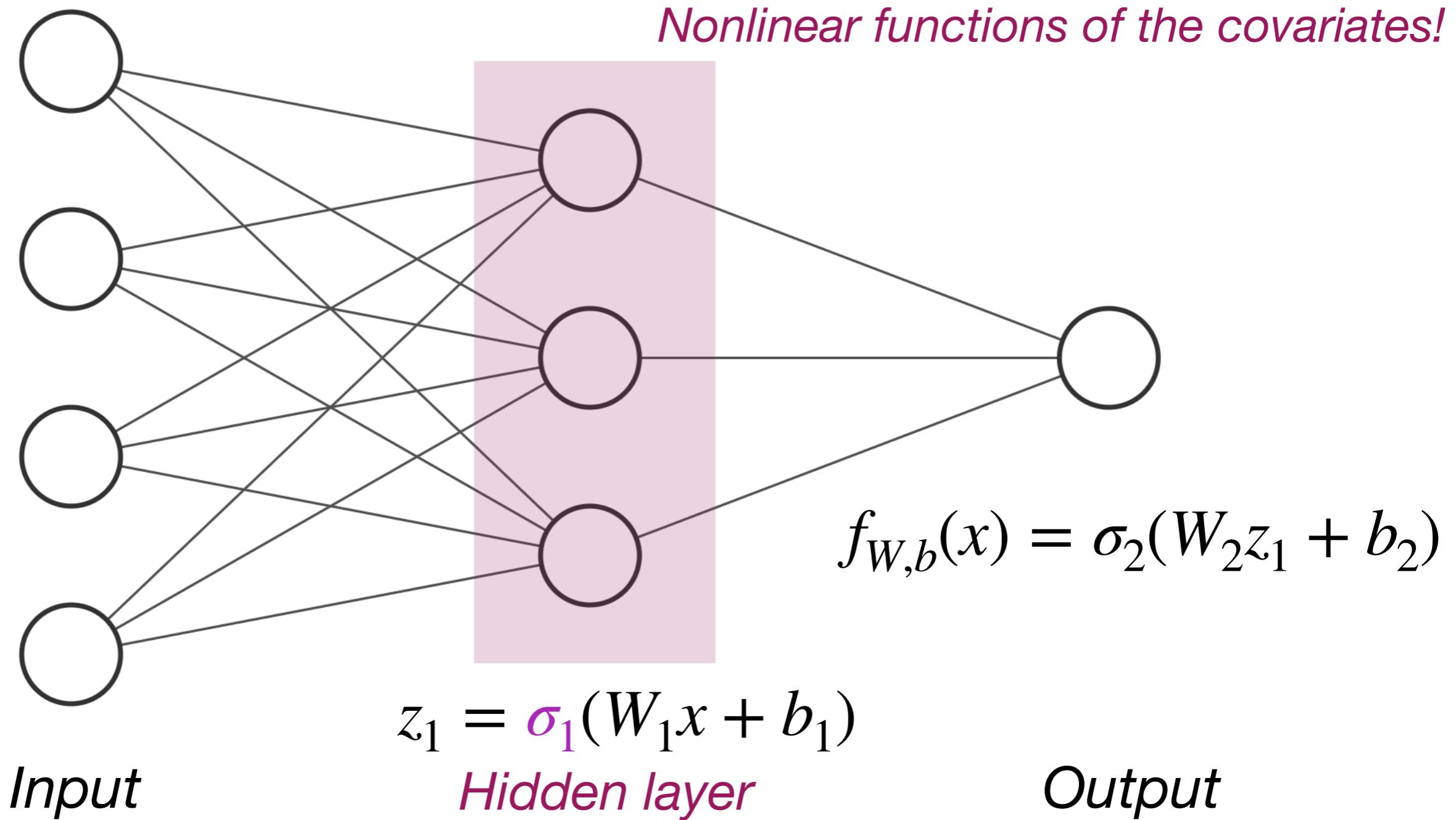
Output Layer $\in \mathbb{R}^1$

Principal component regression as a neural network



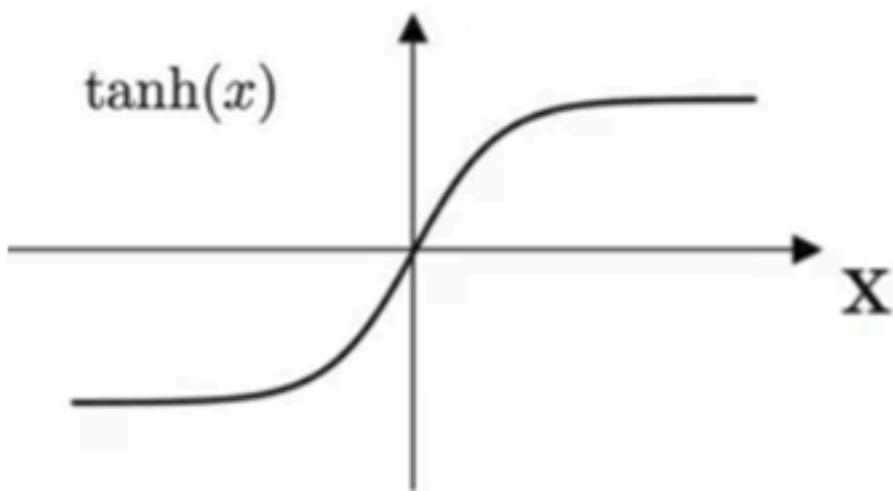
Dense neural networks

- Also known as "Multilayer Perceptrons"

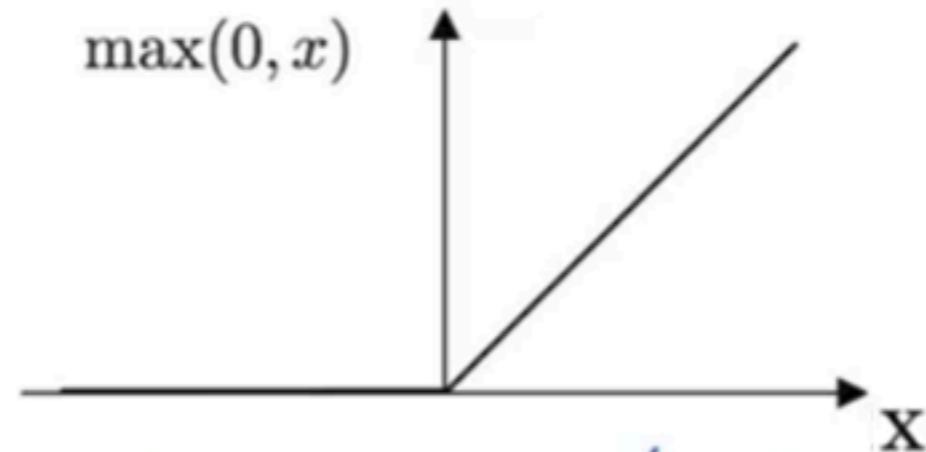


Activation functions σ

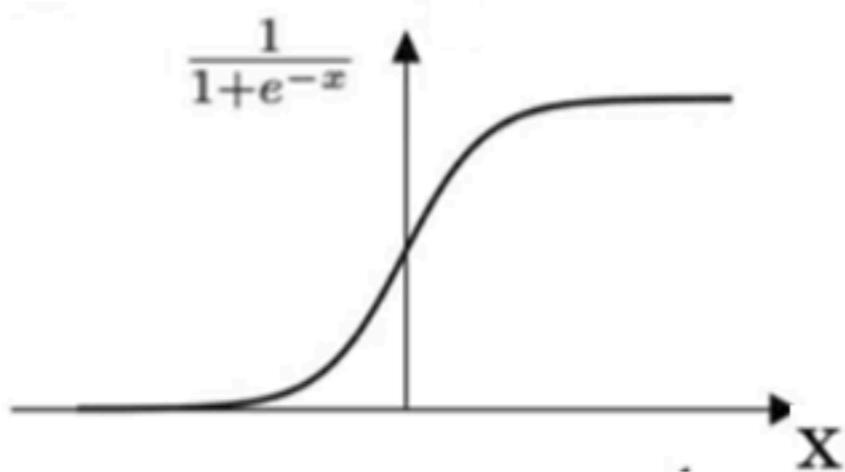
Hyper Tangent Function



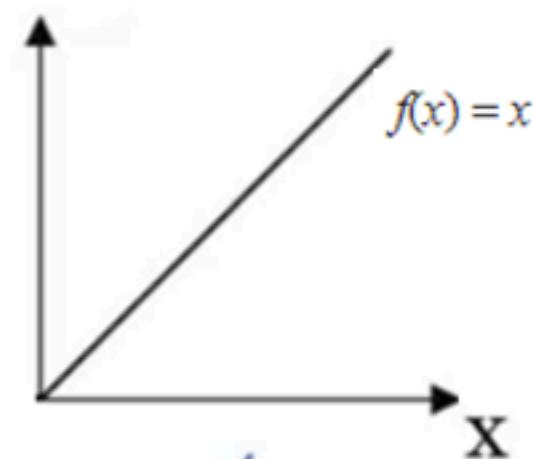
ReLU Function



Sigmoid Function

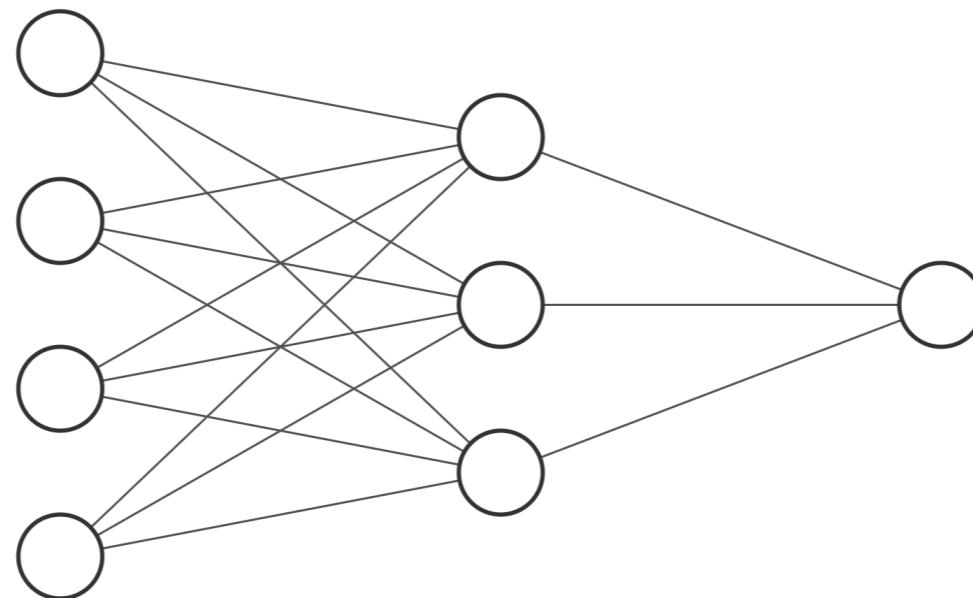


Identity Function



Quick quiz

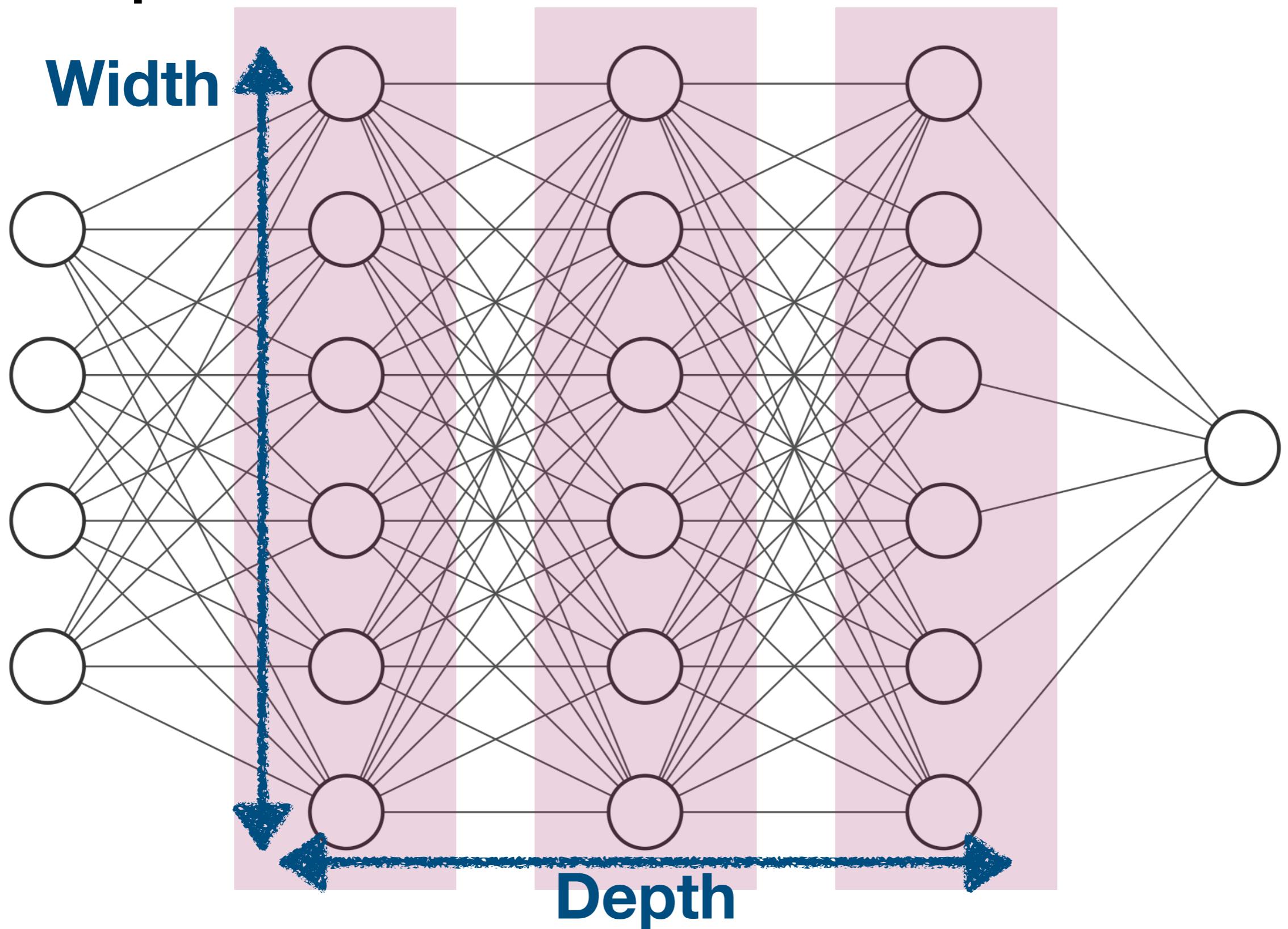
- How many parameters do we need to estimate in the following neural network?



- A: 8 parameters
- B: 15 parameters
- C: 19 parameters

*12 weights and 3 bias parameters in the first layer
3 weights and 1 bias parameter in the second layer*

Deep neural networks



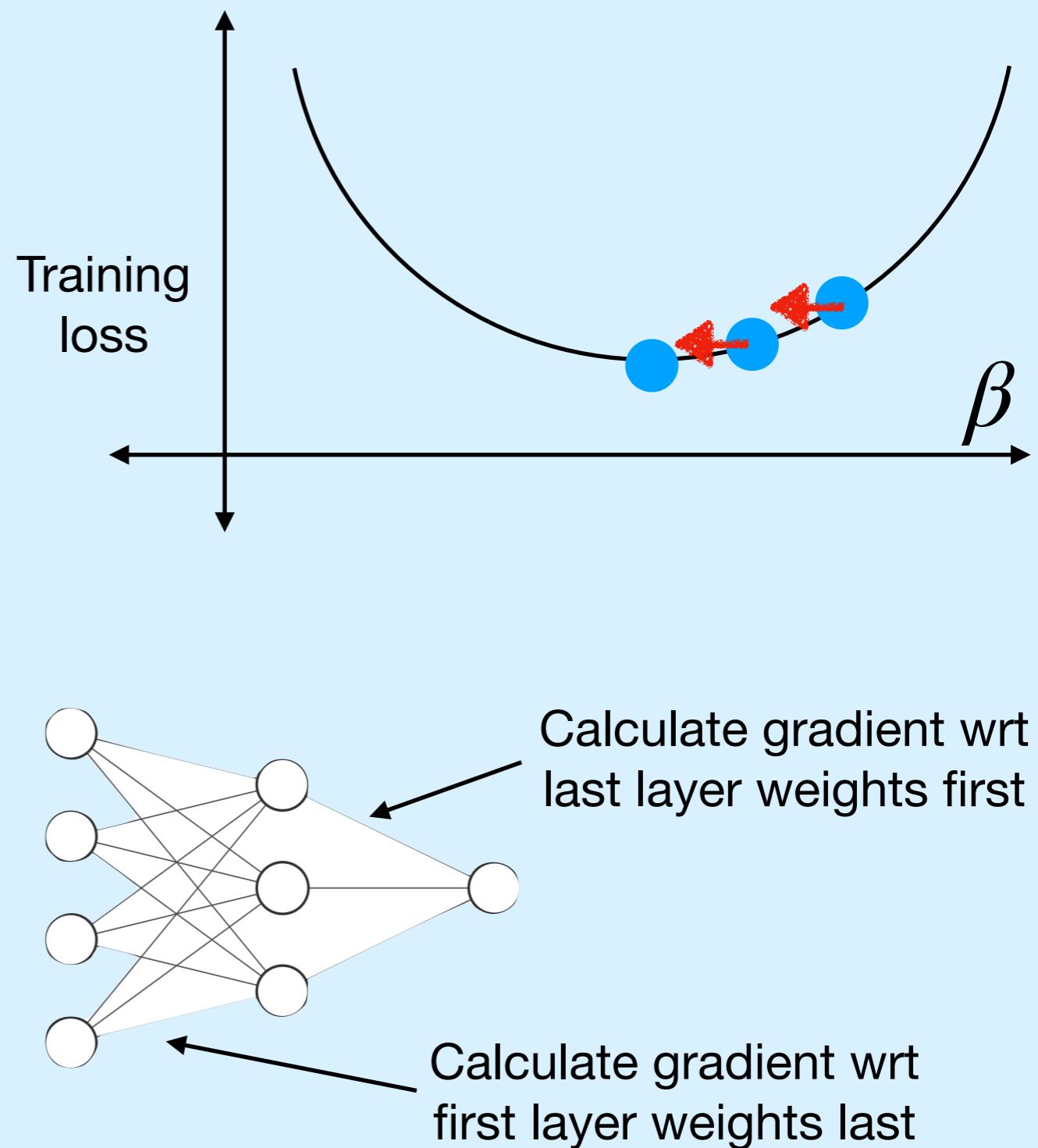
Training a neural network

1. Specify a fixed network architecture:
 - Network depth (number of hidden layers)
 - Width of each hidden layer
 - Activation function at each node
2. Find the network parameters (weights and biases) that minimize the objective function, e.g. the empirical mean squared error.
3. Evaluate the model on a held-out data using sample-splitting or cross-validation. Go back to step 1 to evaluate other network architectures. Iterate until you have a good set of hyperparameters.

Training a neural network: Step 2

2. Find the network parameters (weights and biases) that minimize the objective function, e.g. the empirical mean squared error.

- Use gradient-based minimization, e.g. batch gradient descent, stochastic gradient descent, Adam, etc.
- To calculate the gradients efficiently, apply **back-propagation**.



Training a neural network

1. Specify a fixed network architecture:
 - Network depth (number of hidden layers)
 - Width of each hidden layer
 - Activation function at each node

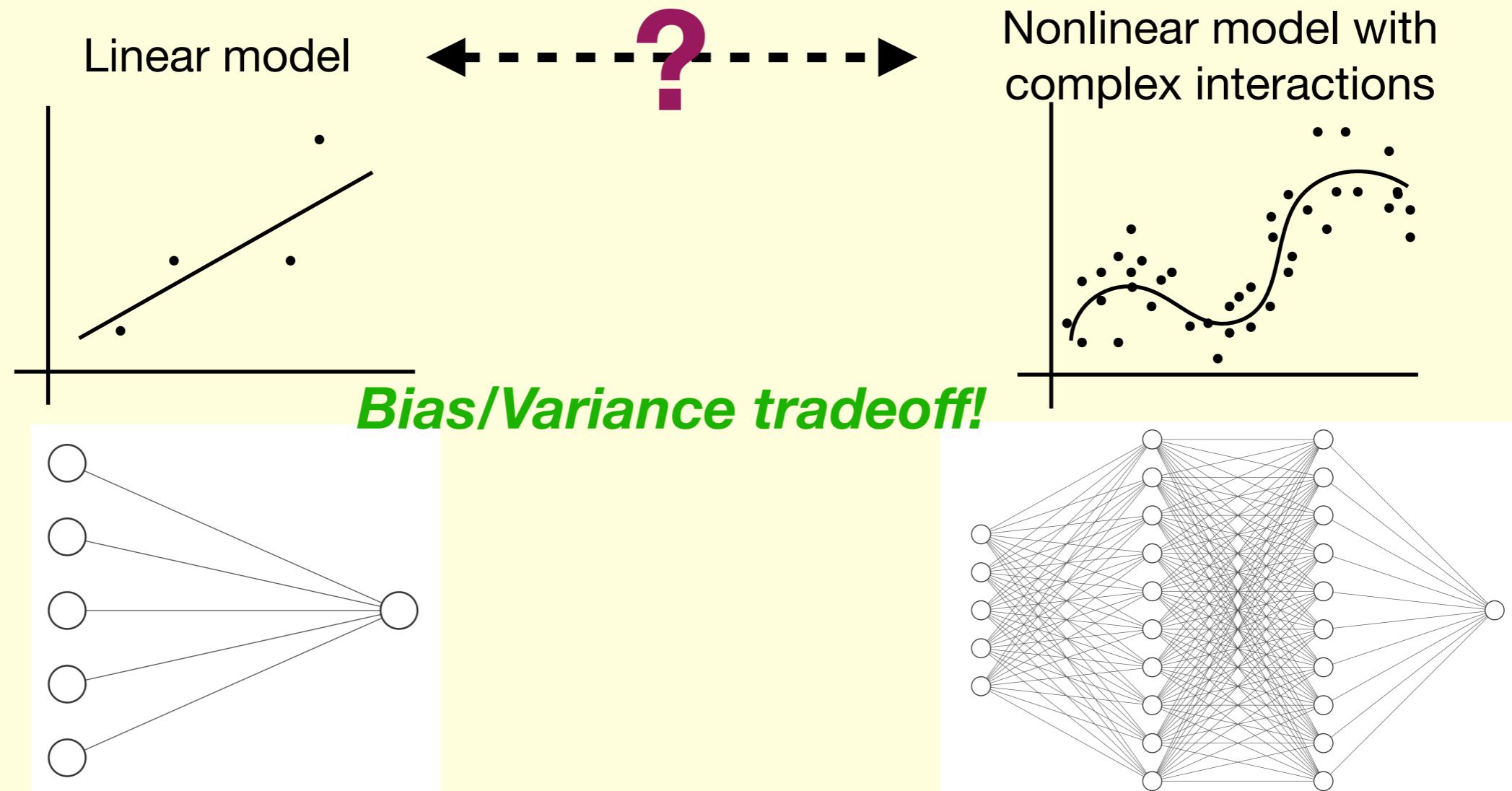
2. Find the network parameters (weights and biases) that minimize the objective function, e.g. the empirical mean squared error.

3. Evaluate the model on a held-out data using sample-splitting or cross-validation. Go back to step 1 to evaluate other network architectures. Iterate until you have a good set of hyperparameters.

This is the most difficult and time-consuming step!

Training a neural network: Step 3

3. Evaluate the model on a held-out data using sample-splitting or cross-validation. Go back to step 1 to evaluate other network architectures. Iterate until you have a good set of hyperparameters.

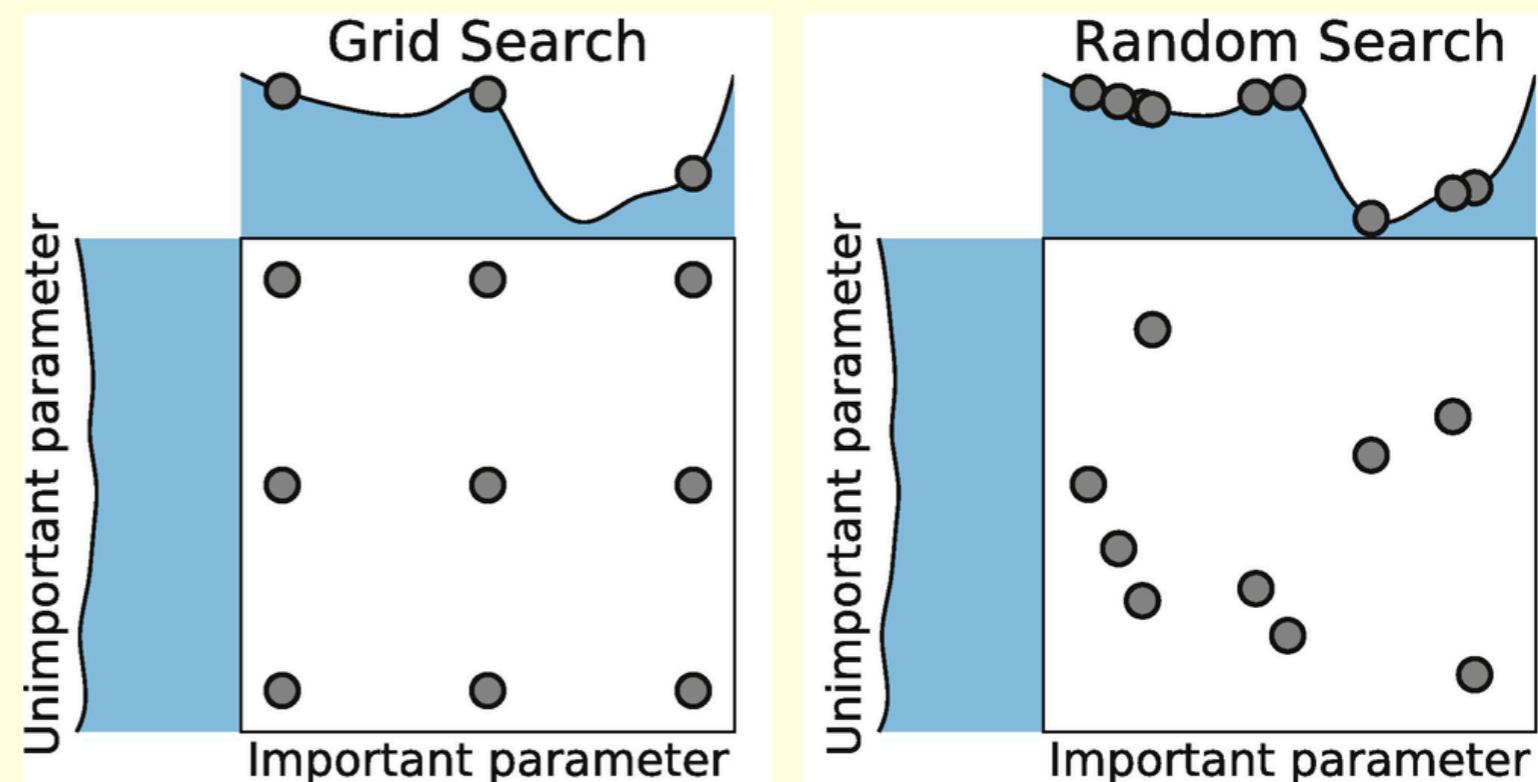


Training a neural network: Step 3

3. Evaluate the model on a held-out data using sample-splitting or cross-validation. Go back to step 1 to evaluate other network architectures. Iterate until you have a good set of hyperparameters.

Hyperparameters to think about:

- Network depth (number of hidden layers)
- Width of each hidden layer
- Activation function at each node



Regularization

- Since neural networks have a tendency to overfit to the training data, researchers have developed many new ways to regularize networks:
 - **Penalization:** Add penalties on the network weights to the training objective, e.g. $\|W\|_2^2$ and $\|W\|_1$
 - **Dropout** (Srivastava 2014): Randomly drop out nodes from the network at each training iteration
 - **Stochastic gradient descent:** Each training iteration only uses a random subset of observations. This both speeds up training and improves generalization.
 - ...

Summary: Dense neural network

- Neural networks are expressive estimators that generalize many well-known statistical methods
- The size of the neural network determines the bias/variance tradeoff and should be carefully tuned
- Neural networks are highly modular and can be combined with other regression tools, e.g. regularization

Lab: Let's go to
<https://playground.tensorflow.org/>