Deep Learning and Its Impact on Artificial Intelligence

Vassilis Athitsos
Vision-Learning-Mining (VLM) Lab
Computer Science and Engineering Department
University of Texas at Arlington
April 4, 2018

What Does the Title Mean?

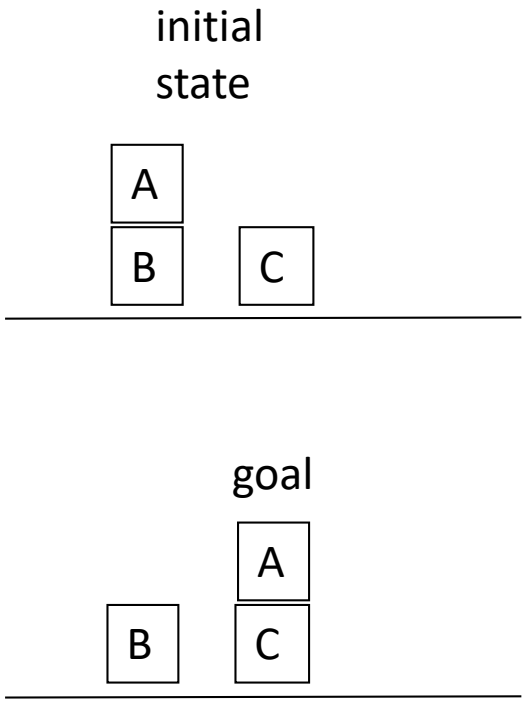
- “Deep Learning and Its Impact on Artificial Intelligence”.
- What do we mean by:
  - Artificial Intelligence.
  - Machine Learning.
  - Deep Learning.

Artificial Intelligence

- In Sci-Fi:
  - Robots that think and act intelligently, often surpassing human performance.
    - Data in Star Trek.
    - C-3P0 in Star Wars…
  - All-knowing computers.
- In real life: artificial intelligence is the science of designing systems (hardware or software) that can:
  - perceive their environment, and
  - have specific goals, and
  - use reasoning to improve their chances of achieving those goals.

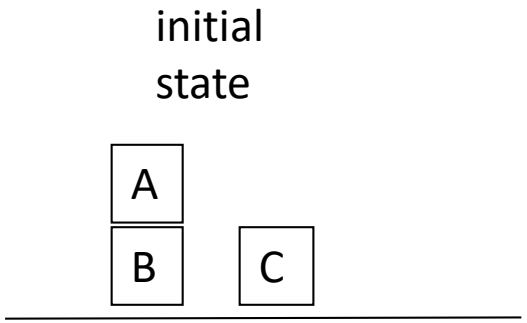
A Simple Example: Blocks World

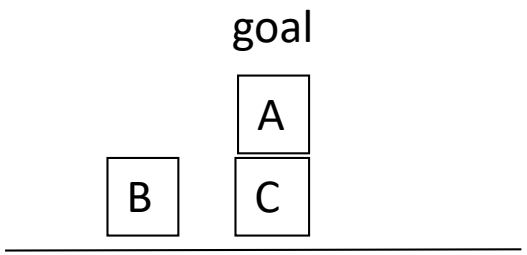
- Studied in the 1960s-1970s.
- There are cubic blocks on the table, in a specific arrangement.
- We want a different arrangement.
- The system figures out a sequence of actions to accomplish that goal.
- Such a system must:
  - Perceive its environment.
  - Be able to come up with a plan to achieve its goal.
  - Be able to perform the required actions.



A Simple Example: Blocks World

- Studied in the 1960s-1970s.
- There are cubic blocks on the table, in a specific arrangement.
- We want a different arrangement.
- The system figures out a sequence of actions to accomplish that goal.
- Such a system must:
  - Perceive its environment.
  - Be able to come up with a plan to achieve its goal.
  - Be able to perform the required actions.





Not quite as impressive as Data or C-3PO…

AI Progress

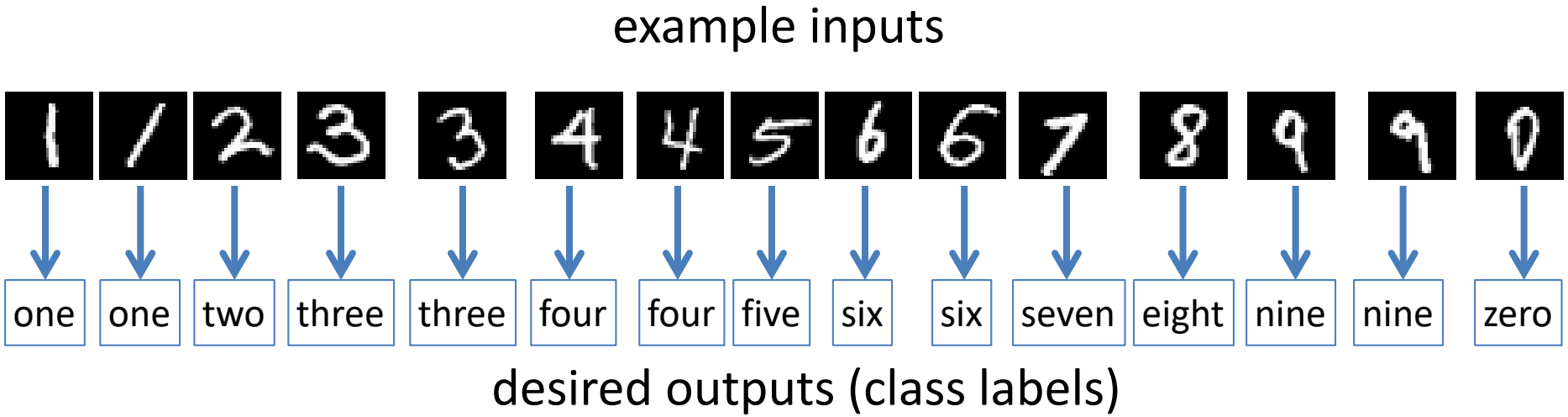
- AI systems have been getting more complex, more “intelligent”.
  - 1960s-1970s: Blocks world.
  - 1997: Beat the world champion in chess.
  - 1990s: commercial speech recognition systems.
  - 2000s: face detectors in consumer cameras.
  - 2015: highly accurate machine translations from English to French, and for other pairs of languages with large amounts of training data.

What Is Machine Learning?

- We focus on **supervised learning**.
  - The computer is presented with training examples, consisting of example inputs and their desired outputs.
    - The job of the "supervisor" is to prepare those training examples.
  - Goal: learn a general function that maps inputs to outputs.

Supervised Learning

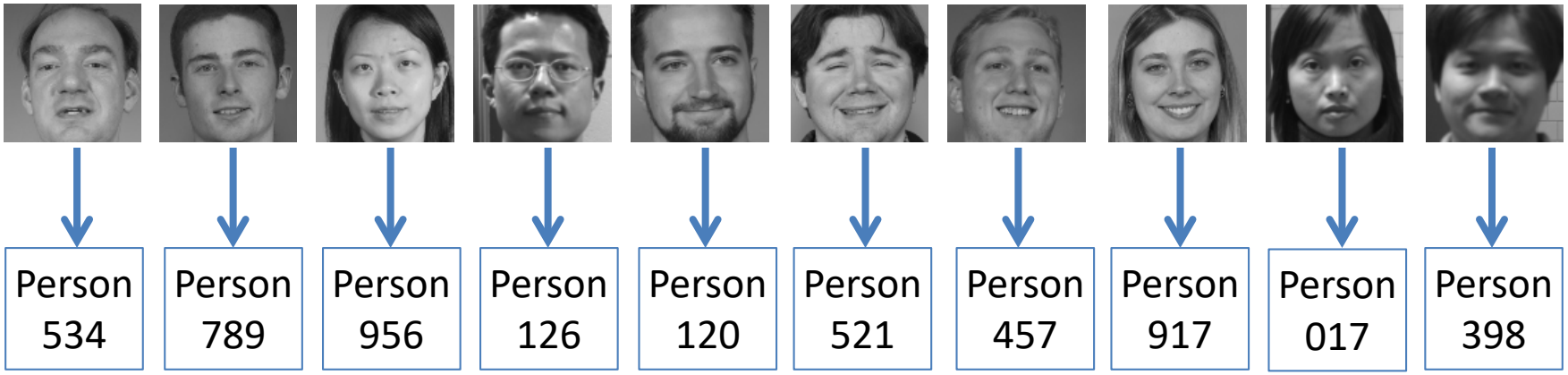
- Example: recognizing the digits of zip codes.
  - The training set consists of images of digits and the names of those digits.



Supervised Learning

- Example: face recognition
  - The training set consists of images of faces and the IDs of those faces.

example inputs



desired outputs (class labels)

AI and Machine Learning

- Machine Learning is a branch of Artificial Intelligence.
- People have been producing AI systems without using machine learning.
- How can you do AI without learning?
  - Use rules:
    - Chess: make sure your pieces are not threatened, capture the highest value piece if possible...
  - Use search:
    - Blocks world: explore various sequences of actions, find the one that achieves the goal.
    - Chess: try billions of sequences of moves, find the one that leads to the most advantageous state.

AI and Machine Learning

- Examples of AI systems over the year:

  NO LEARNING, SEARCH BASED - 1960s: blocks world

  NO LEARNING, SEARCH BASED - 1997: beat the world champion in chess.

  LEARNING – 1990s: commercial speech recognition systems.

  LEARNING – 2000s: face detectors in consumer cameras.

  DEEP LEARNING – 2010s: highly accurate machine translations.

  DEEP LEARNING - 2015: world championship level play in Go.

Why Machine Learning?

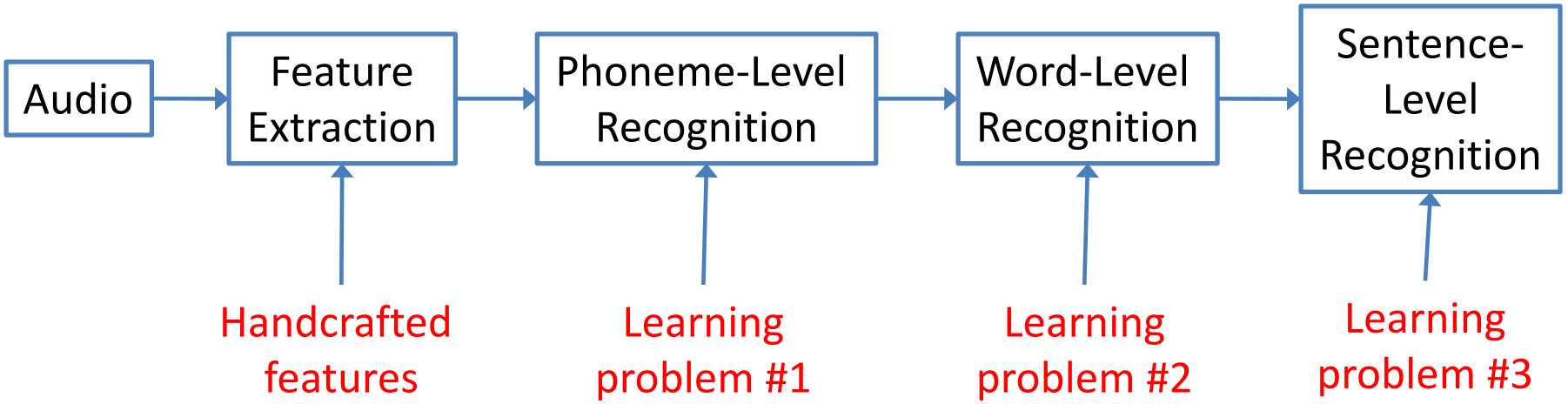
- Machine Learning has dominated AI in recent years.
- Reasons:
  - More efficient for humans: coming up with manually hardcoded rules takes time, and must be done every time, for each new problem.
  - More efficient for computers: directly mapping inputs to outputs can be far more efficient than searching through a huge number of possible outputs, and waiting to stumble upon the right output.
  - More accurate: learning from lots of examples consistently beats systems that are rule-based or search-based.

Shallow Learning

- To understand what is “deep” about deep learning, we should see how things were done back in the age of “shallow” learning.
- Example: speech recognition in the old days.

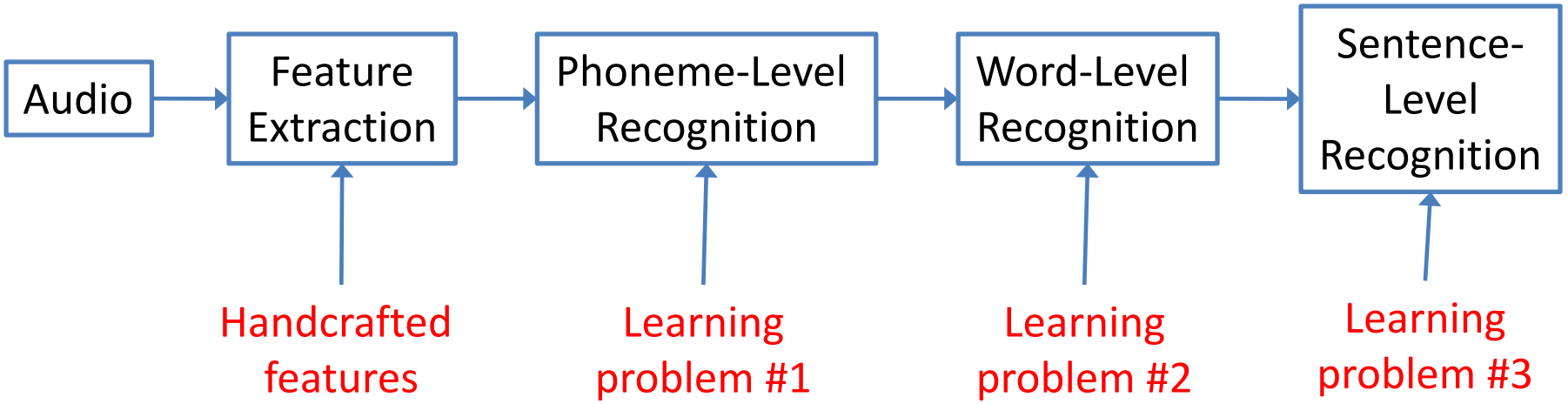
Shallow Learning

- To understand what is “deep” about deep learning, we should see how things were done back in the age of “shallow” learning.
- Example: speech recognition in the old days.



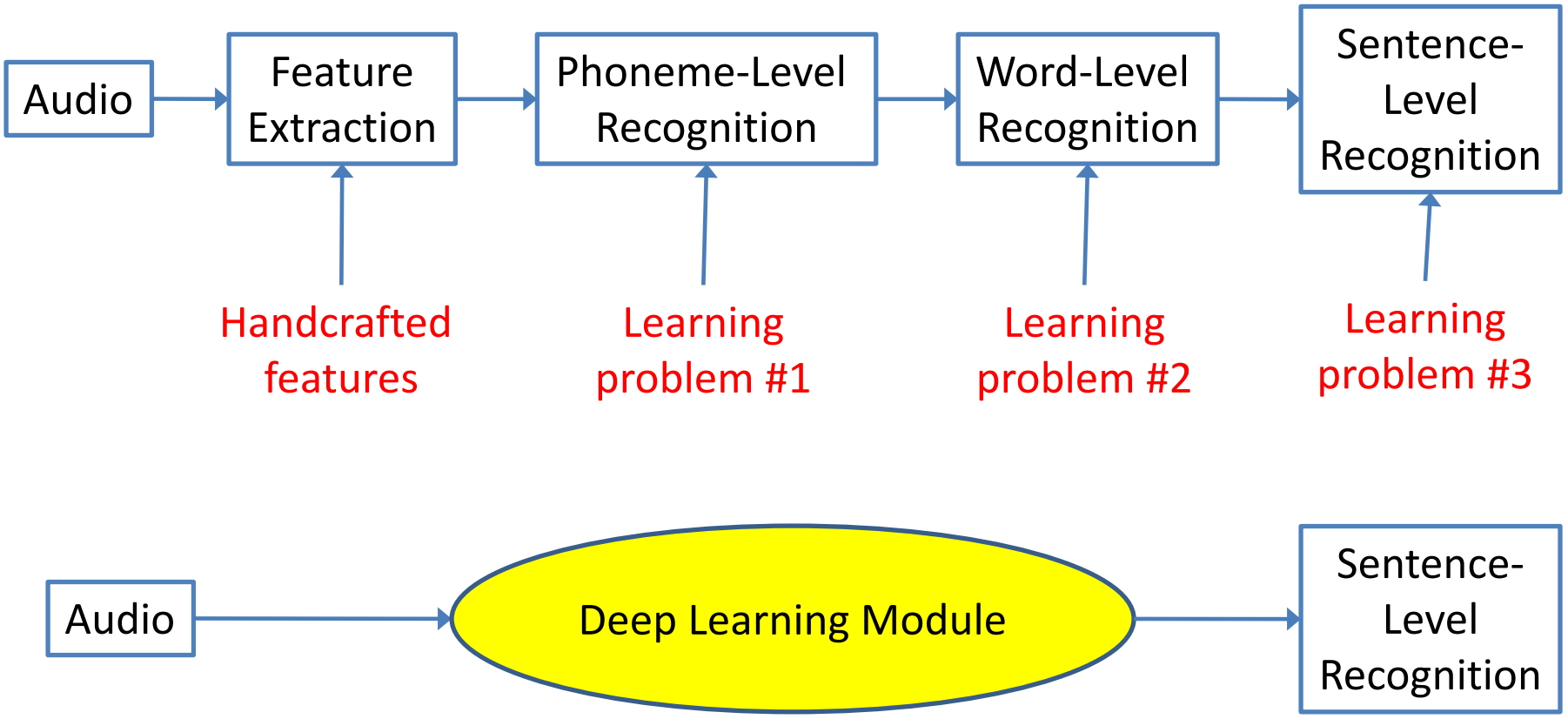
Shallow Learning

- To understand what is “deep” about deep learning, we should see how things were done back in the age of “shallow” learning.
- Example: speech recognition in the old days.



- The big problem was decomposed into smaller problems.
- A lot of manual design was needed for this decomposition.

Shallow vs. Deep Learning



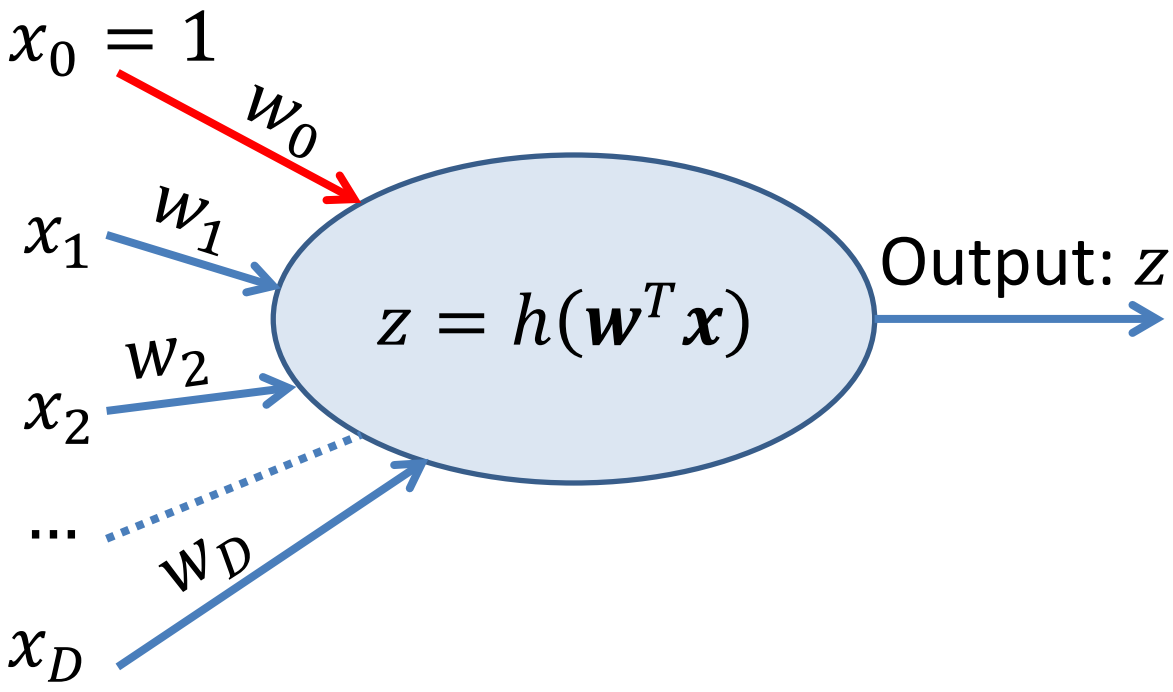
- In deep learning, we try to build a single "deep" model for the entire end-to-end processing that maps input to output.
- Using a deep model eliminates the need to manually define and train separate subsystems for different parts of the process.

Deep Learning and Neural Networks

- Deep learning models are neural networks.
- However, neural networks have been around since the 1960s.
  - The deep learning craze started around 2010.
- So, deep learning and neural networks are related, but not synonyms.
- First, we will look at the basic definition of neural networks.
- Then, we will see how neural networks are used at deep learning.

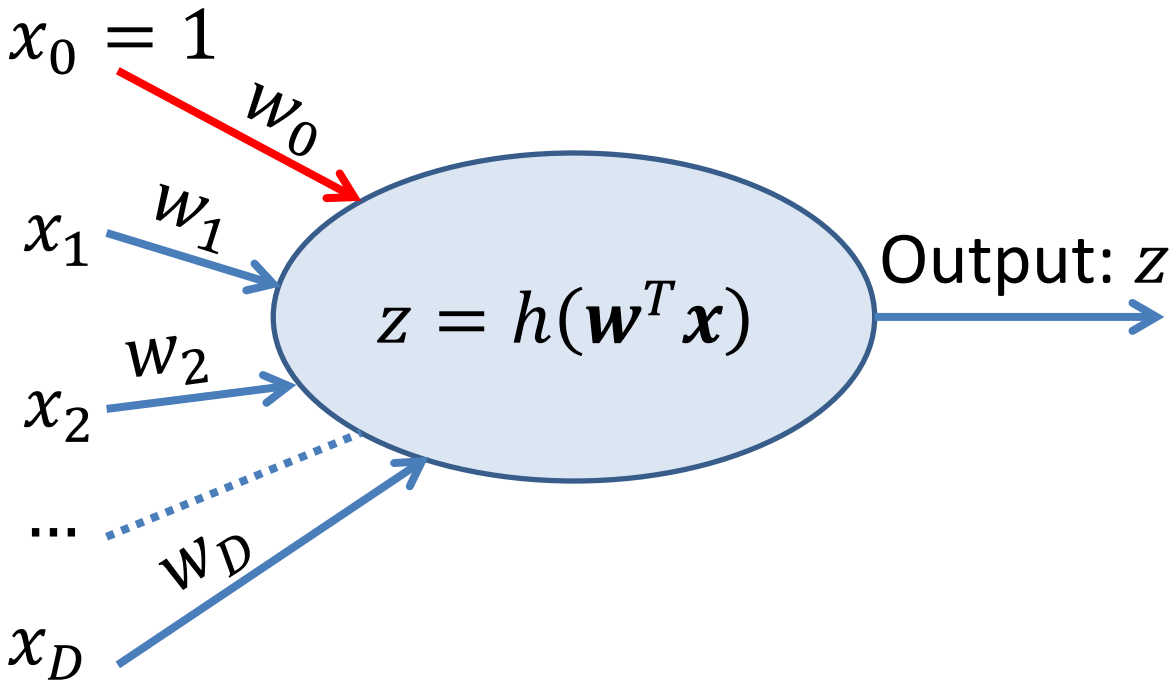
Perceptrons

$$\mathbf{x} = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ \dots \\ x_D \end{bmatrix}$$



- Perceptrons are the building blocks of neural networks.
- A single perceptron takes a multidimensional input, and produces a single number as output.
- What is "learnable" in a perceptron is a set of weights.
  - Each weight corresponds to a dimension of the input vector.

Perceptrons



- A perceptron computes its output z in two steps:

First step: dot product. $a = \boldsymbol{w}^T \boldsymbol{x} = \sum_{i=0}^{D} (w_i x_i)$

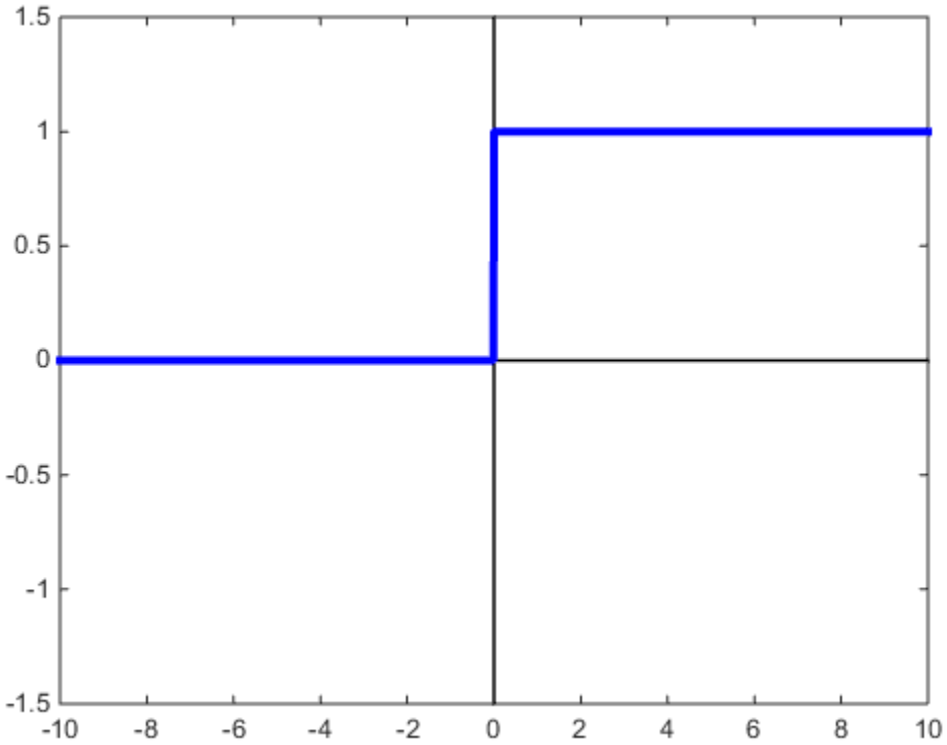
$$\mathbf{x} = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ \cdots \\ x_D \end{bmatrix}$$

Second step: activation function. $z = h(a)$

For example, h could be the sigmoid: $\sigma(a) = \frac{1}{1+e^{-a}}$

- In a single formula: $z = h\left(\sum_{i=0}^{D} (w_i x_i)\right)$

Activation Functions

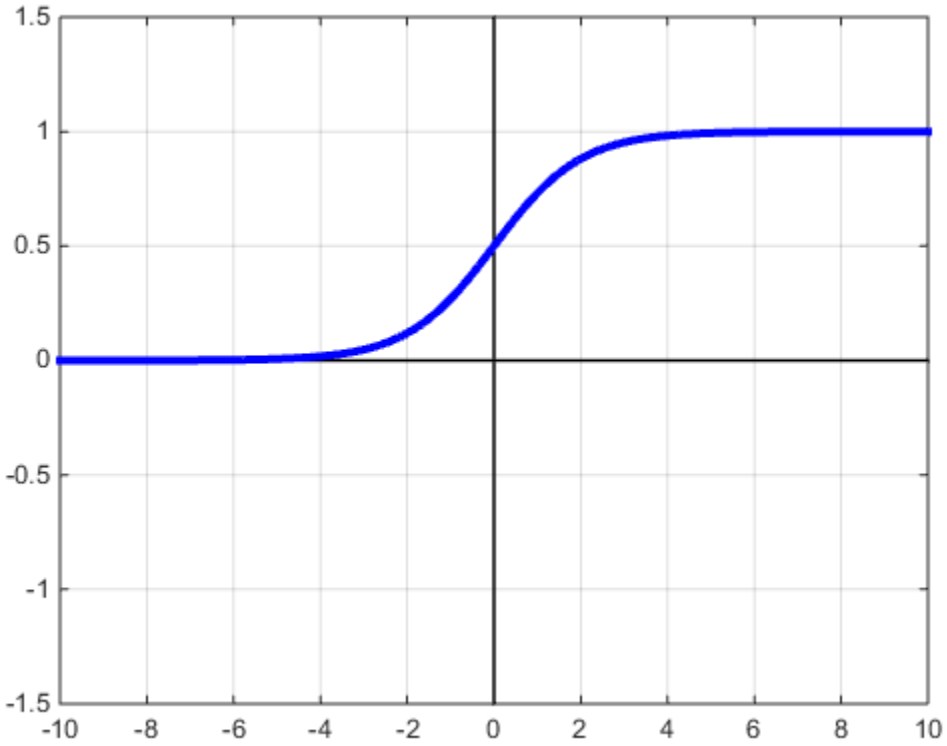


- A perceptron produces output $z = h(\boldsymbol{w}^T \boldsymbol{x})$.
- One choice for the activation function h: the **step function**.

$$h(a) = \begin{cases} 0, \text{if } a < 0 \\ 1, \text{if } a \geq 0 \end{cases}$$

- The step function is useful for providing some intuitive examples.
- It is not useful for actual real-world systems.
  - Reason: it is not differentiable, it does not allow optimization via gradient descent.

Activation Functions

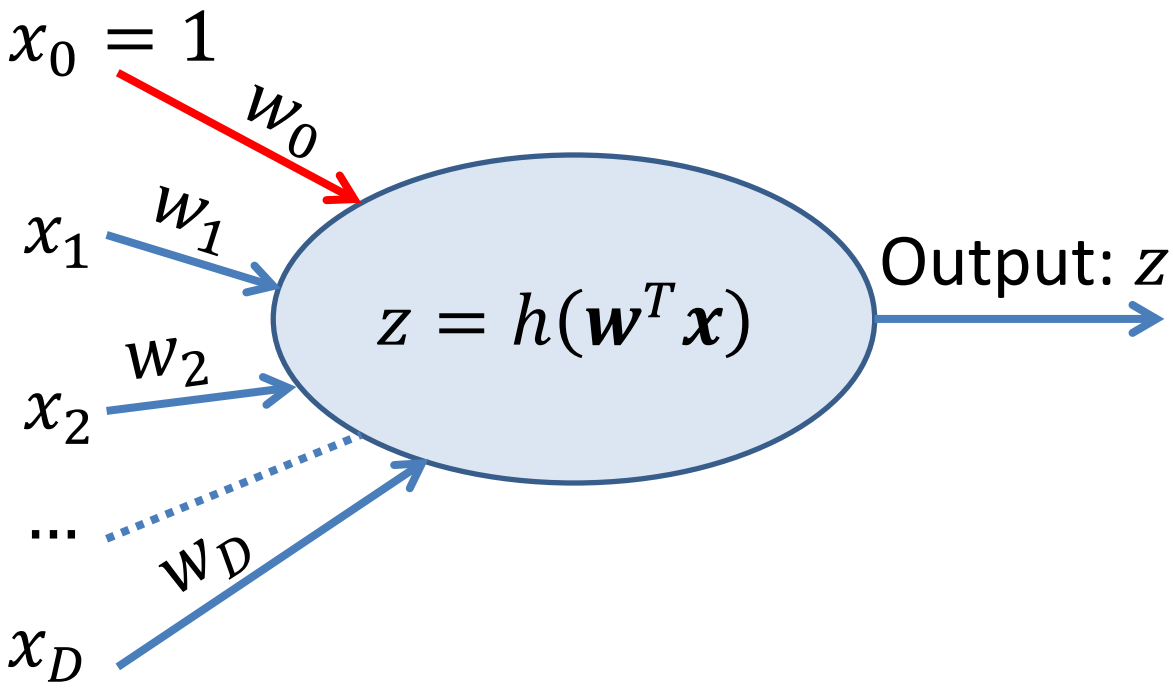


- A perceptron produces output $z = h(\boldsymbol{w}^T \boldsymbol{x})$.
- Another choice for the activation function $h(a)$: the **sigmoidal function**.

$$\sigma(a) = \frac{1}{1+e^{-a}}$$

- The sigmoidal is often used in real-world systems.
- It is a differentiable function, it allows use of gradient descent.
- Other commonly used activation functions: tanh, ReLU (rectified linear unit).

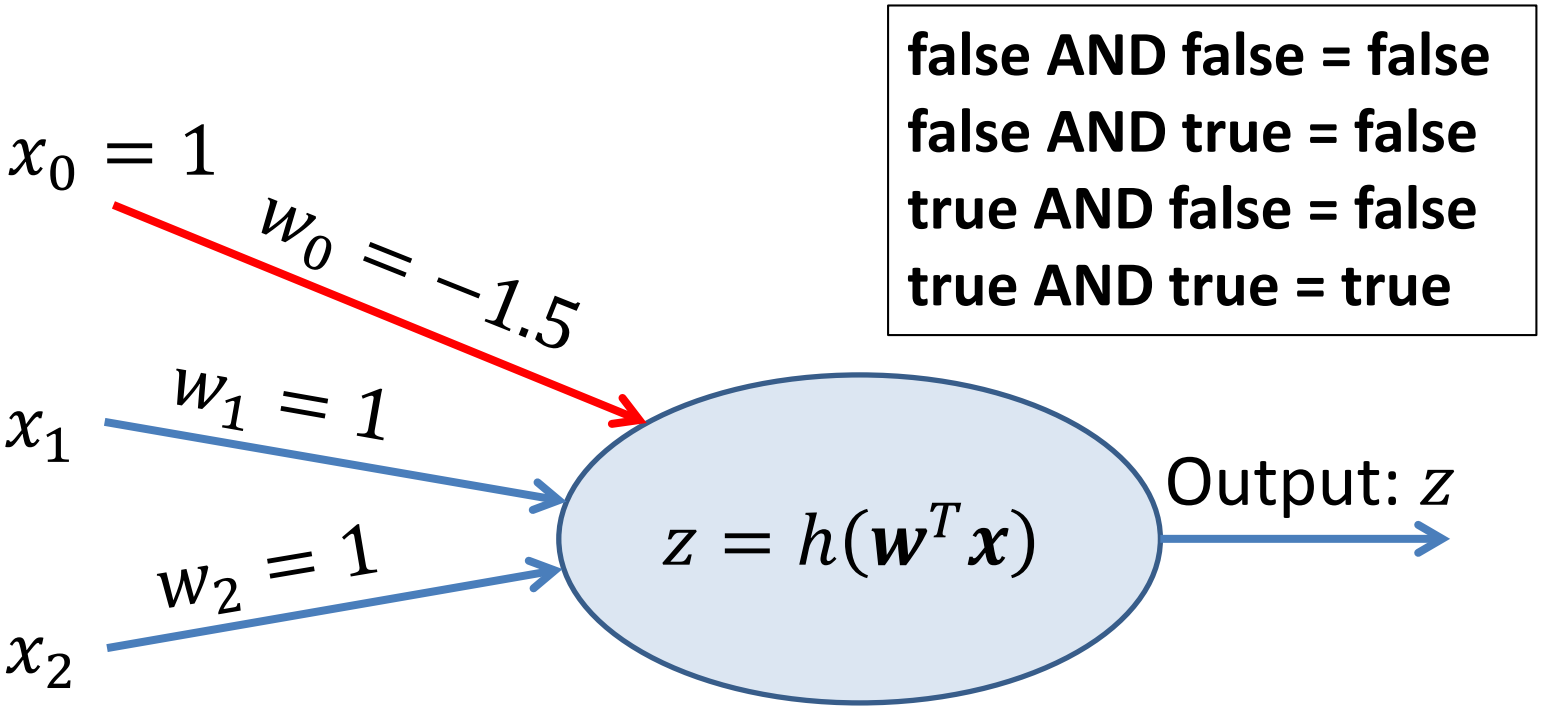
Perceptrons and Neurons



- Perceptrons are inspired by neurons.
  - Neurons are the cells forming the nervous system, and the brain.
  - Neurons somehow compute a weighted sum of their inputs, and if the sum exceeds a threshold, they "fire".
- Since brains are "intelligent", AI researchers have been hoping for a long time that perceptron-based systems can be used to model intelligence.

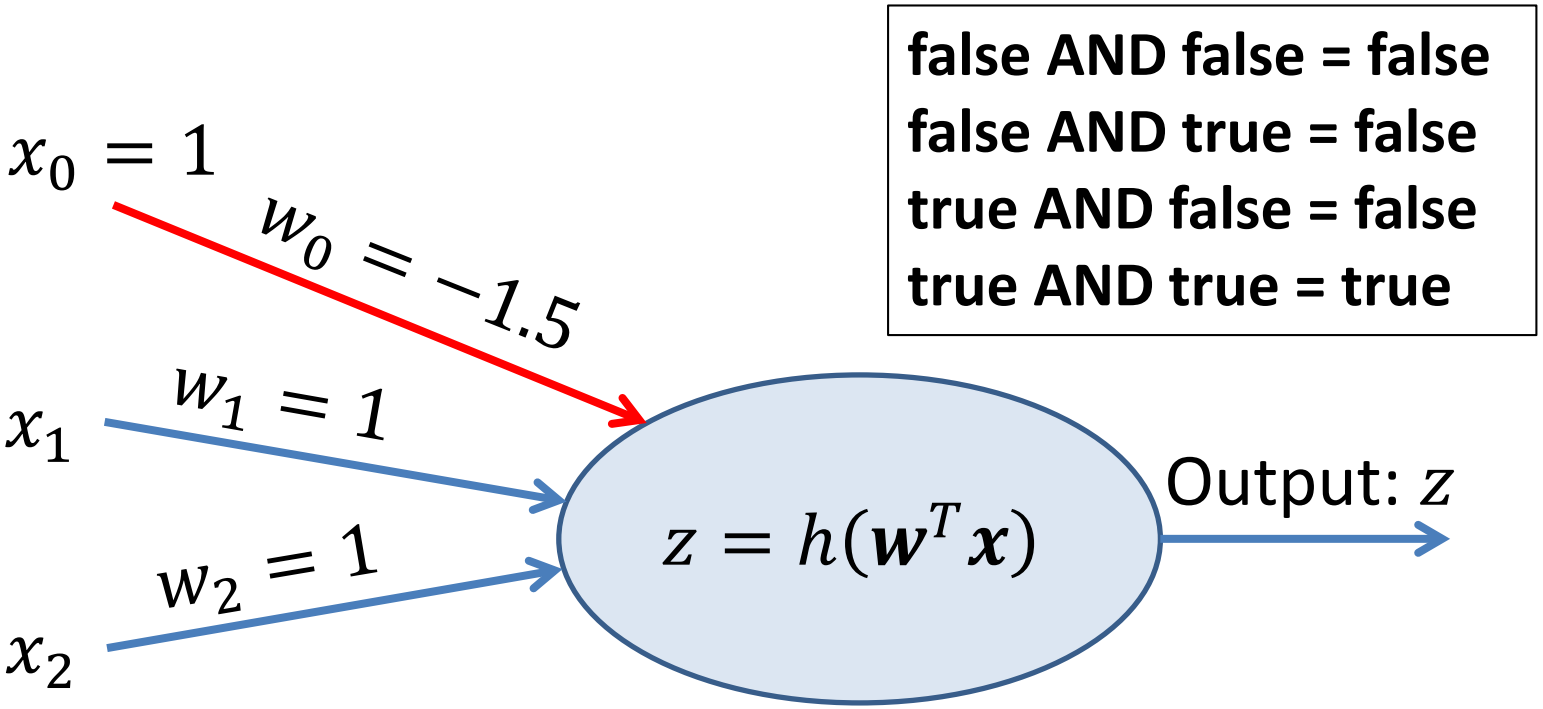
Example: The AND Perceptron

- Suppose we use the **step function** for activation.
- Suppose boolean value **false** is represented as number 0.
- Suppose boolean value tru**e** is represented as number 1.
- Then, the perceptron below computes the boolean AND function:



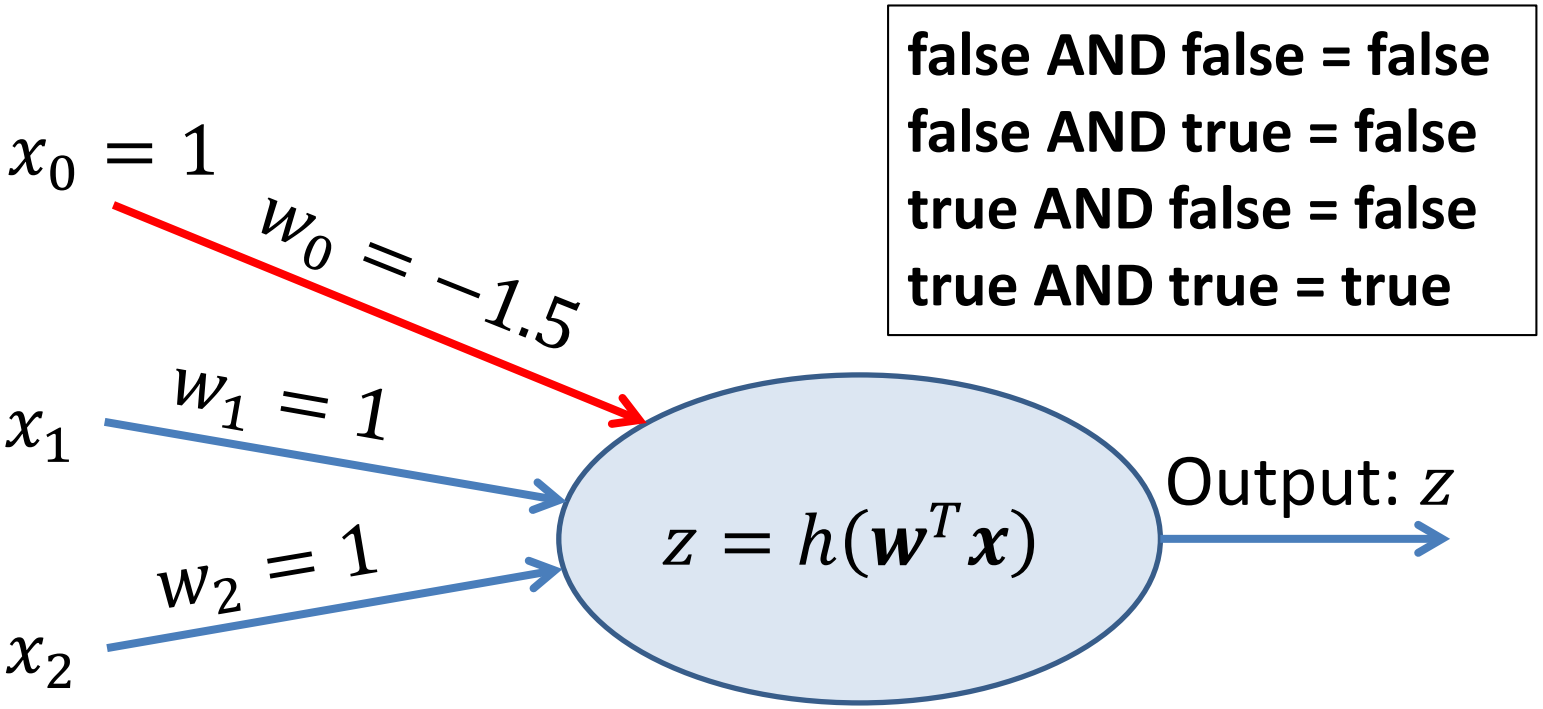
Example: The AND Perceptron

- Verification: If $x_1 = 0$ and $x_2 = 0$:
  - $\boldsymbol{w}^T\boldsymbol{x} = -1.5 + 1 * 0 + 1 * 0 = -1.5.$
  - $h(\boldsymbol{w}^T\boldsymbol{x}) = \mathrm{h}(-1.5) = 0.$
- Corresponds to case **false AND false = false**.



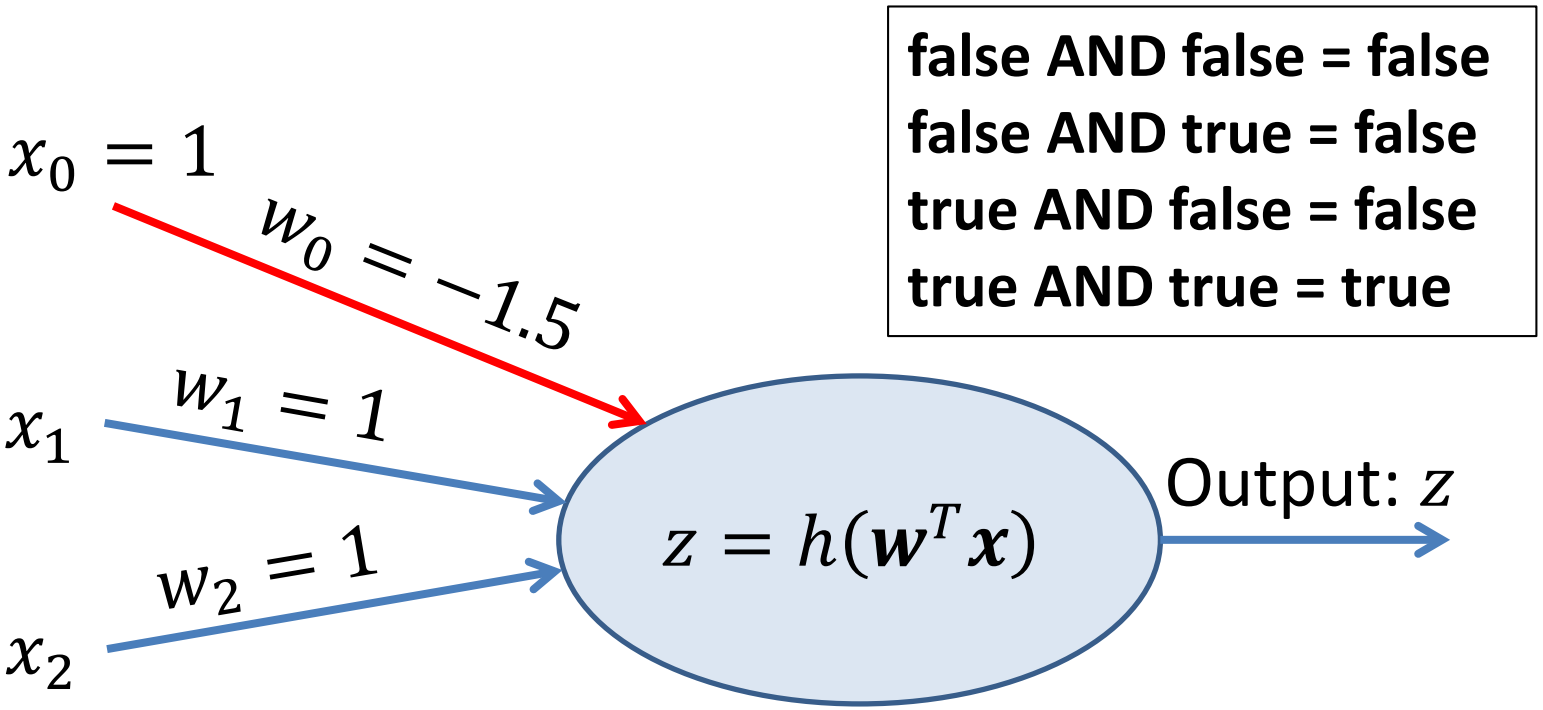
Example: The AND Perceptron

- Verification: If $x_1 = 0$ and $x_2 = 1$:
  - $\boldsymbol{w}^T\boldsymbol{x} = -1.5 + 1 * 0 + 1 * 1 = -0.5.$
  - $h(\boldsymbol{w}^T\boldsymbol{x}) = \mathrm{h}(-0.5) = 0.$
- Corresponds to case **false AND true = false**.



Example: The AND Perceptron

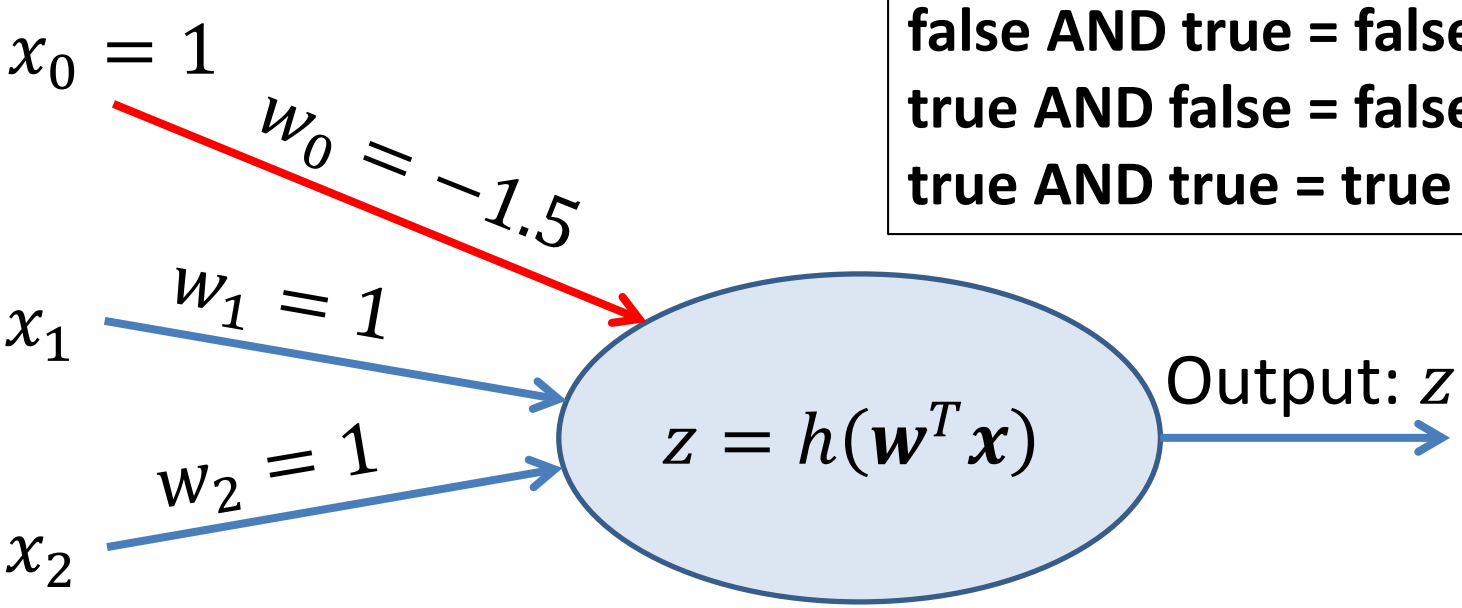
- Verification: If $x_1 = 1$ and $x_2 = 0$:
  - $\boldsymbol{w}^T\boldsymbol{x} = -1.5 + 1 * 1 + 1 * 0 = -0.5.$
  - $h(\boldsymbol{w}^T\boldsymbol{x}) = \mathrm{h}(-0.5) = 0.$
- Corresponds to case **true AND false = false**.



Example: The AND Perceptron

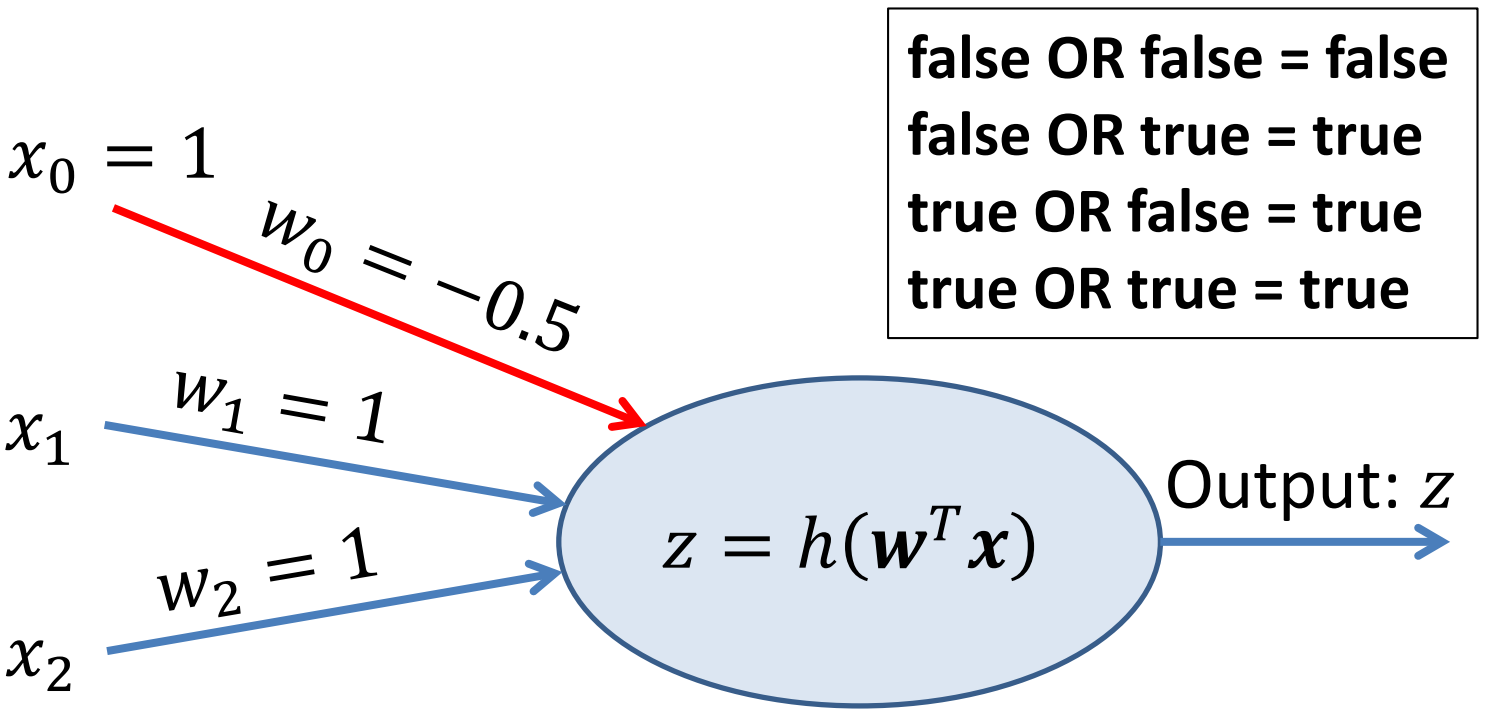
- Verification: If $x_1 = 1$ and $x_2 = 1$:
  - $\boldsymbol{w}^T\boldsymbol{x} = -1.5 + 1 * 1 + 1 * 1 = 0.5.$
  - $h(\boldsymbol{w}^T\boldsymbol{x}) = \mathrm{h}(0.5) = 1.$
- Corresponds to case **true AND true = true**.

false AND false = false
false AND true = false
true AND false = false
true AND true = true



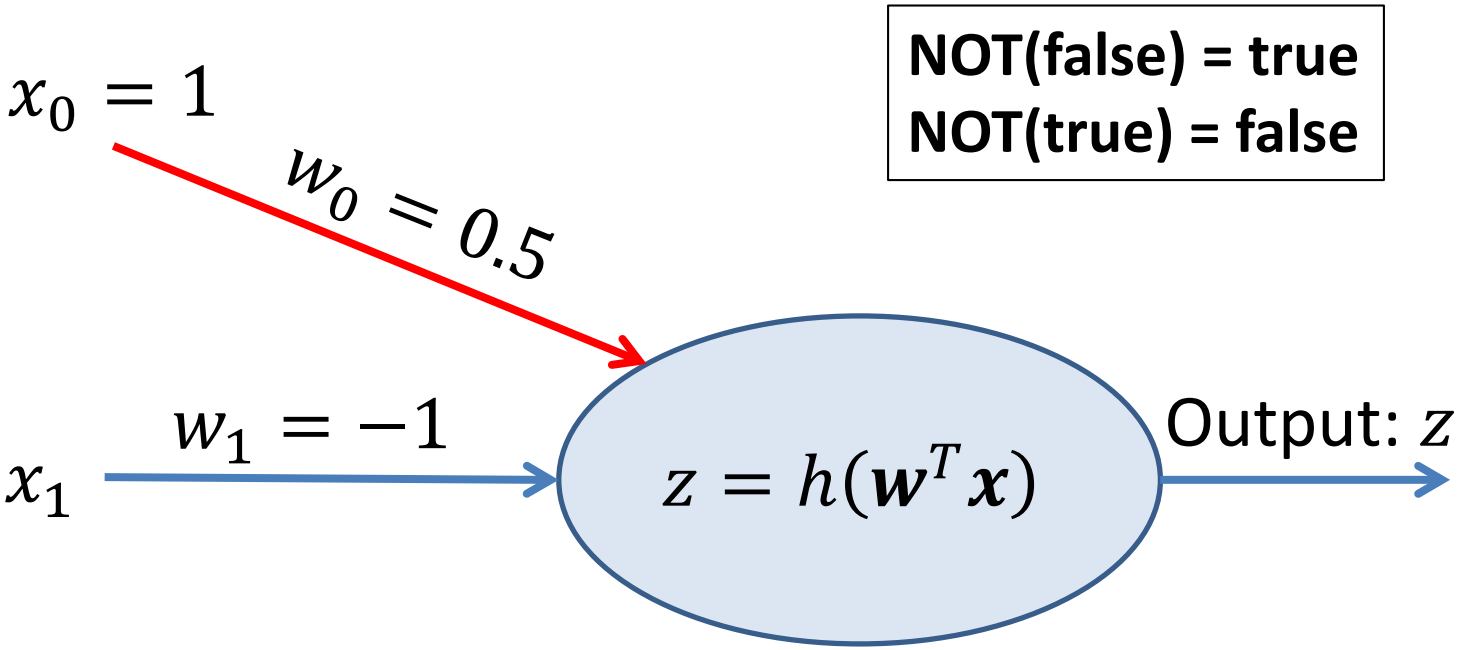
Example: The OR Perceptron

- Suppose we use the **step function** for activation.
- Suppose boolean value **false** is represented as number 0.
- Suppose boolean value tru**e** is represented as number 1.
- Then, the perceptron below computes the boolean OR function:



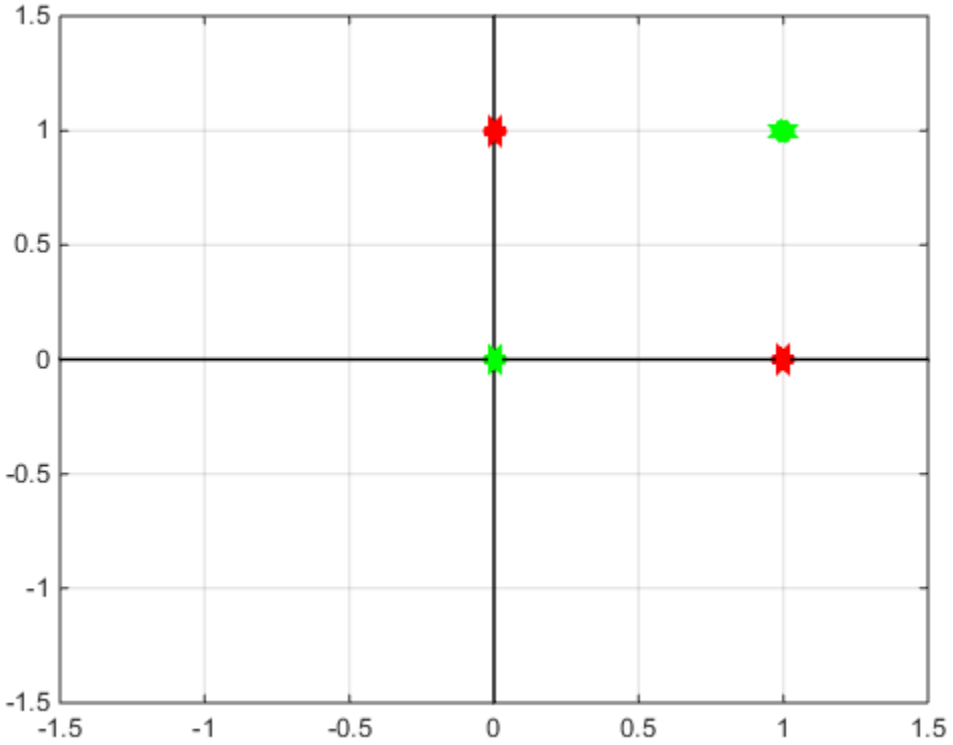
Example: The NOT Perceptron

- Suppose we use the **step function** for activation.
- Suppose boolean value **false** is represented as number 0.
- Suppose boolean value tru**e** is represented as number 1.
- Then, the perceptron below computes the boolean NOT function:



The XOR Function

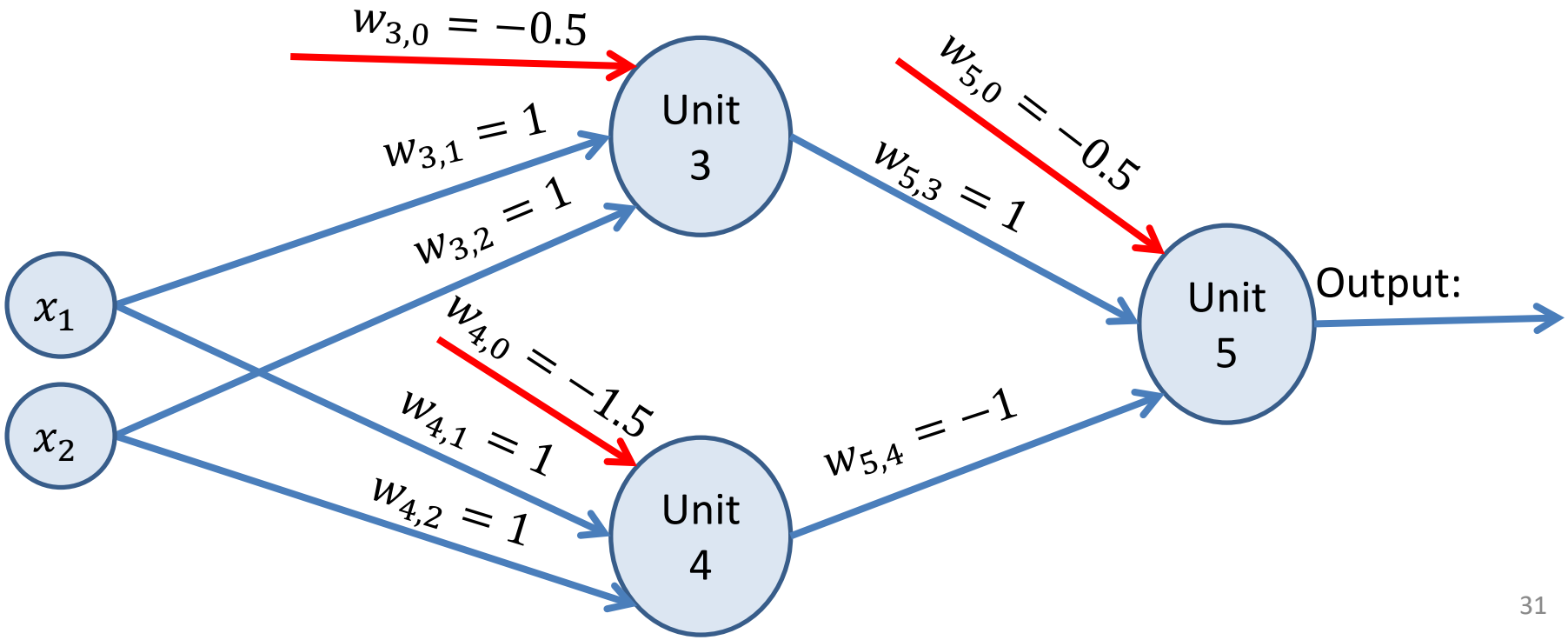
false XOR false = false
false XOR true = true
true XOR false = true
true XOR true = false



- As before, we represent **false** with 0 and **true** with 1.
- The figure shows the four input points of the XOR function.
  - green corresponds to output value **true**.
  - red corresponds to output value **false**.
- The two classes (true and false) are not linearly separable.
- Therefore, no perceptron can compute the XOR function.

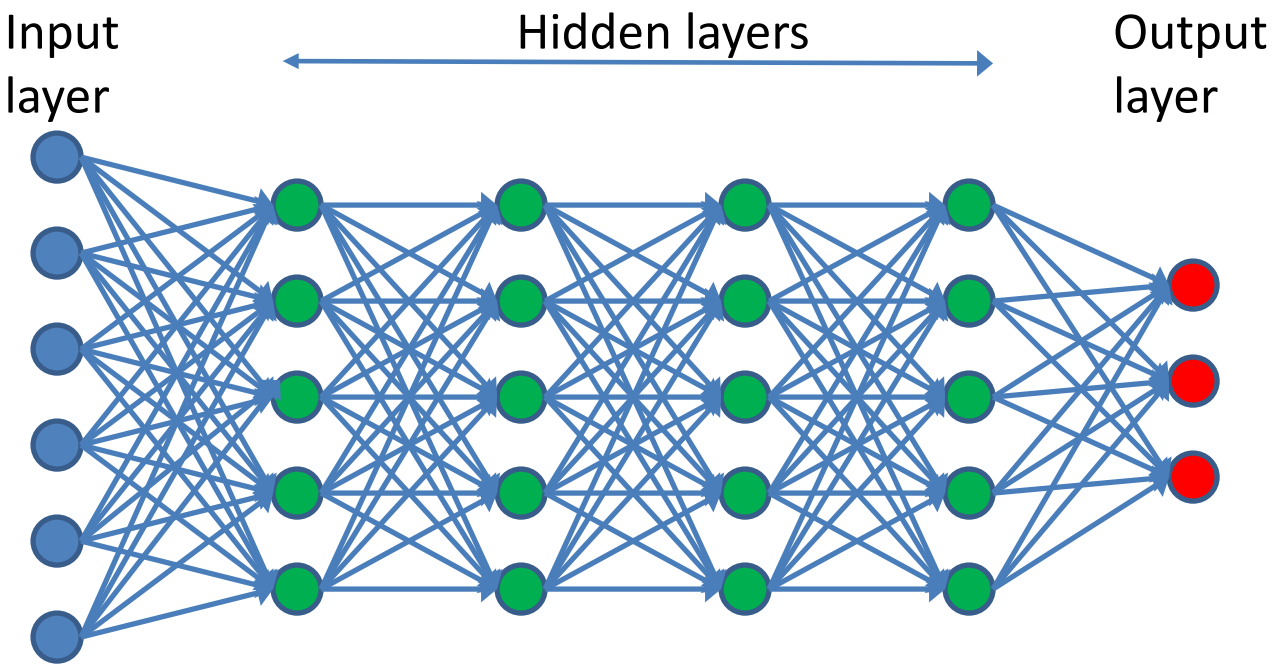
Neural Networks

- A neural network is built using perceptrons as building blocks.
- The inputs to some perceptrons are outputs of other perceptrons.
- Here is an example neural network computing the XOR function.



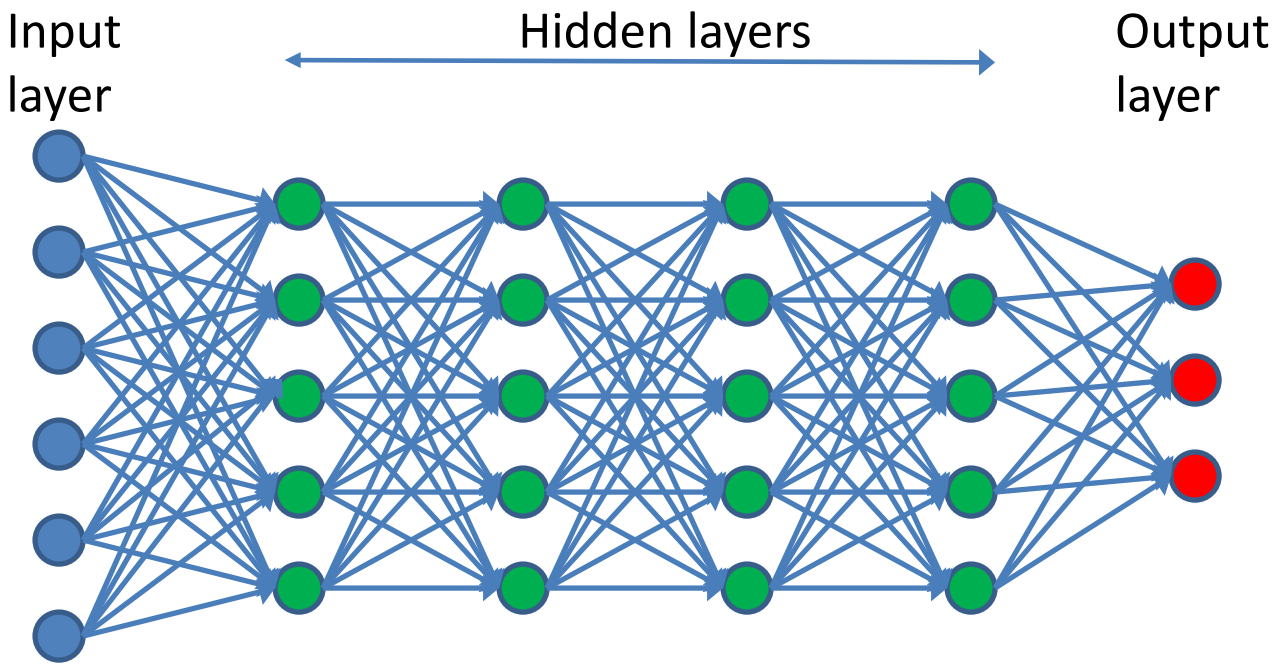
Neural Network Layers

- Usually neural networks are organized into layers.
- The **input layer** is the initial layer of input units.
- The output layer is at the end.
- Zero, one or more hidden layers can be between the input and output layers.



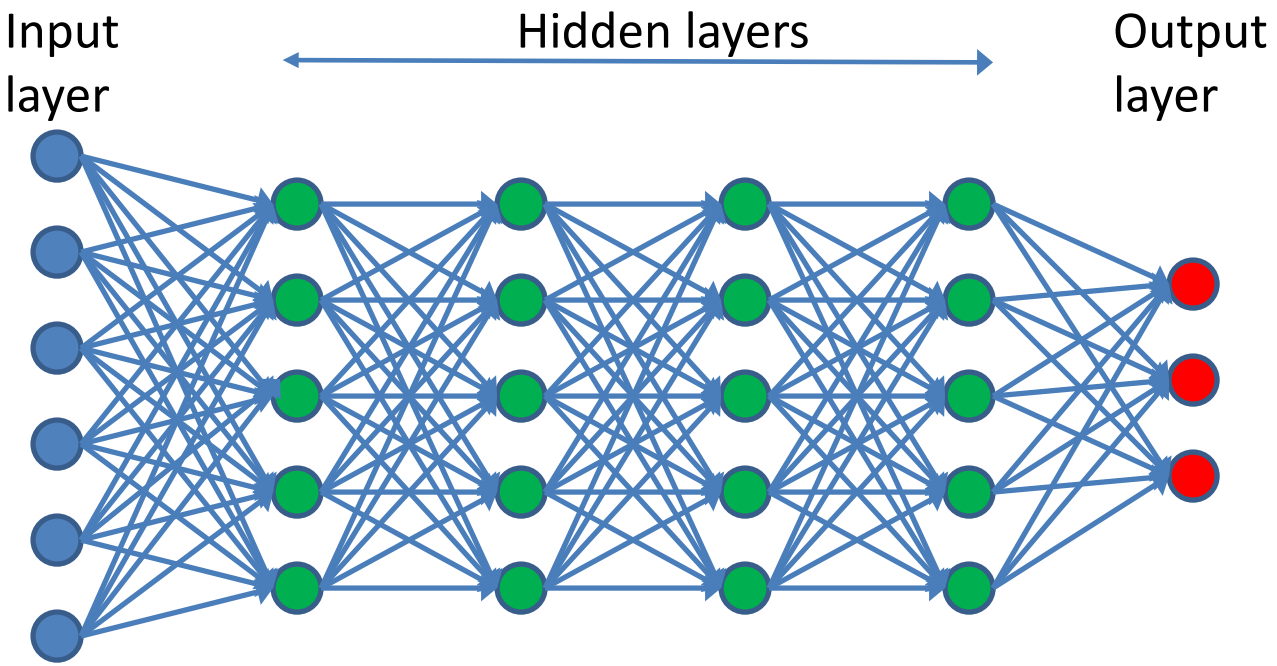
Neural Network Layers

- Each hidden layer's inputs are outputs from the previous layer.
- Each hidden layer's outputs are inputs to the next layer.
- The first hidden layer's inputs come from the input layer.
- The last hidden layer's outputs are inputs to the output layer.



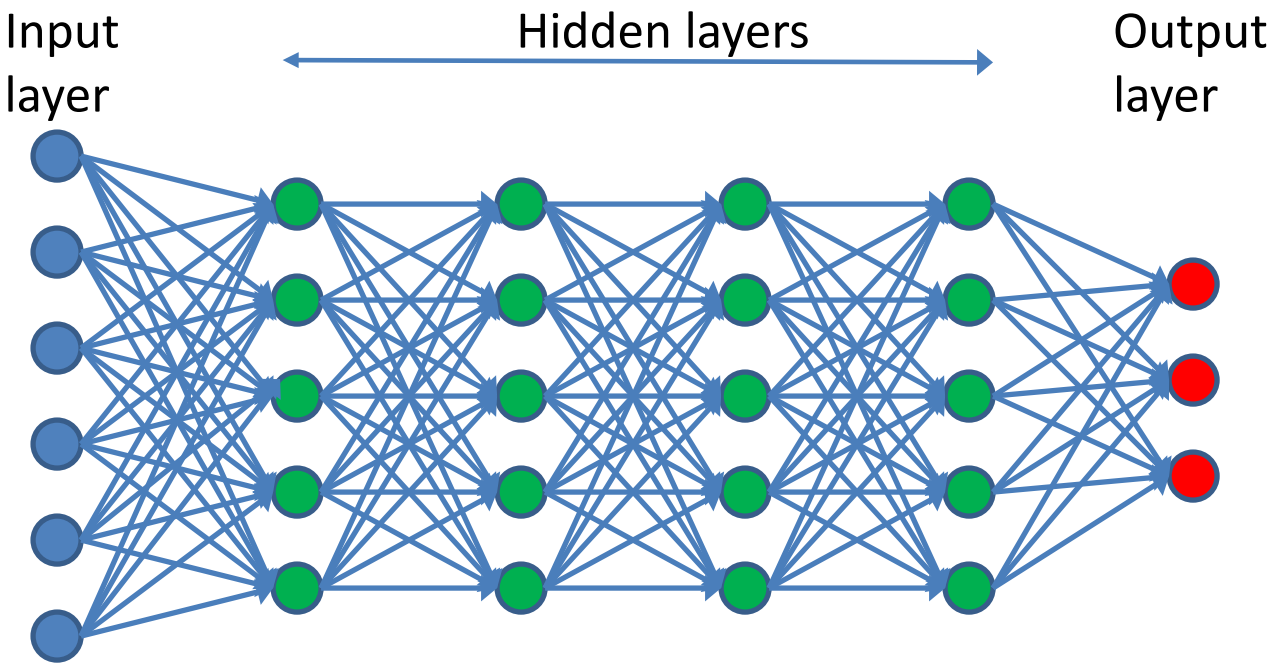
Computing the Output

- To compute the outputs of layer l (where $l > 1$), we simply need to compute the output of each perceptron belonging to layer l .
  - For each such perceptron, its inputs are coming from outputs of perceptrons at layer $l - 1$.
  - So, we compute layer outputs in increasing order of l.



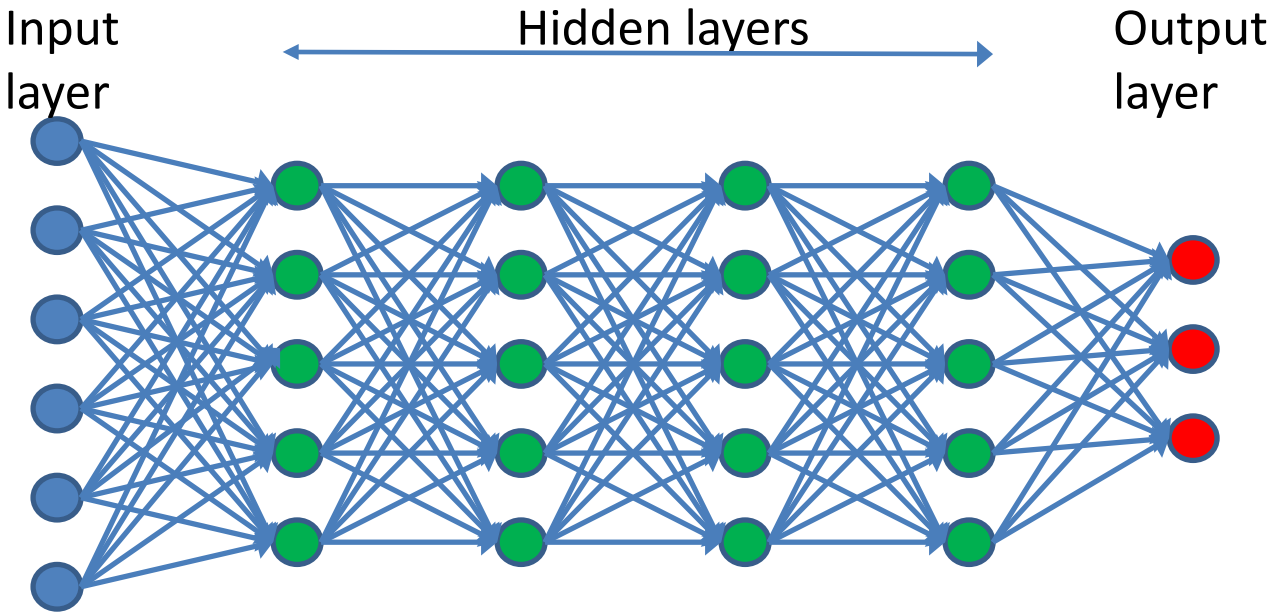
Computing the Output

- Given values for the input units, the output of the network is computed as follows:
- For $(l = 2; l \leq L; l = l + 1)$: // L is the number of layers, layer 1 is input layer.
  - Compute the outputs of layer l, given the outputs of layer $l - 1$.



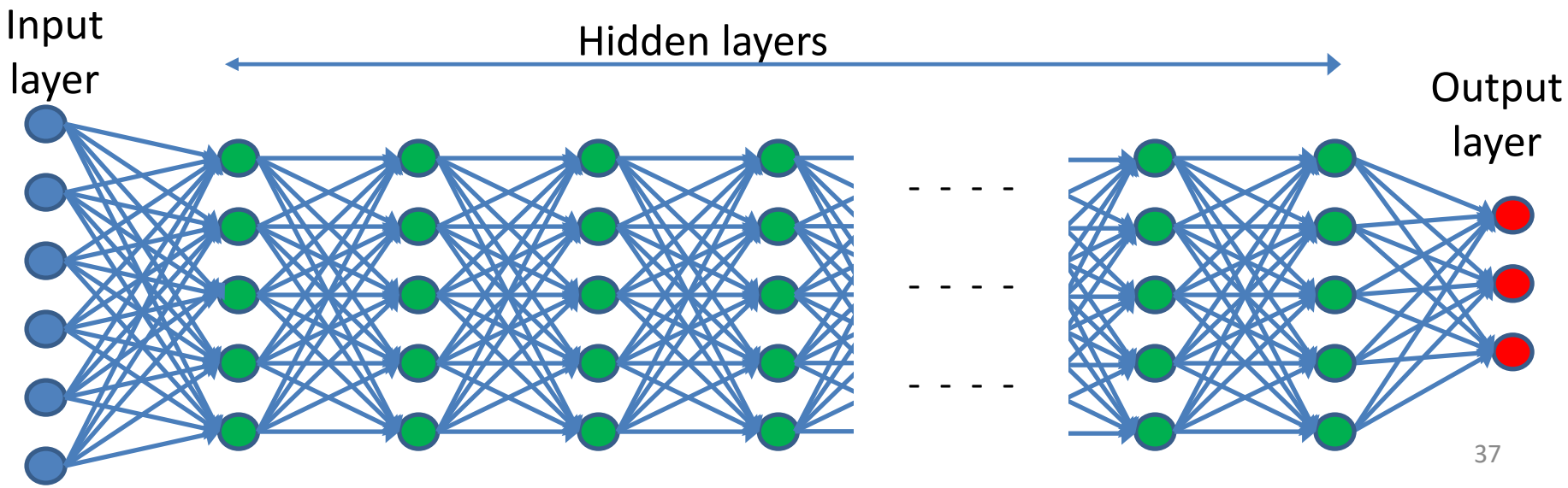
What Neural Networks Can Compute

- Neural networks with two hidden layers can compute any mathematical function.
- However, there are some catches:
  - How do we find the number of units per layer?
  - How do we find the right weights?



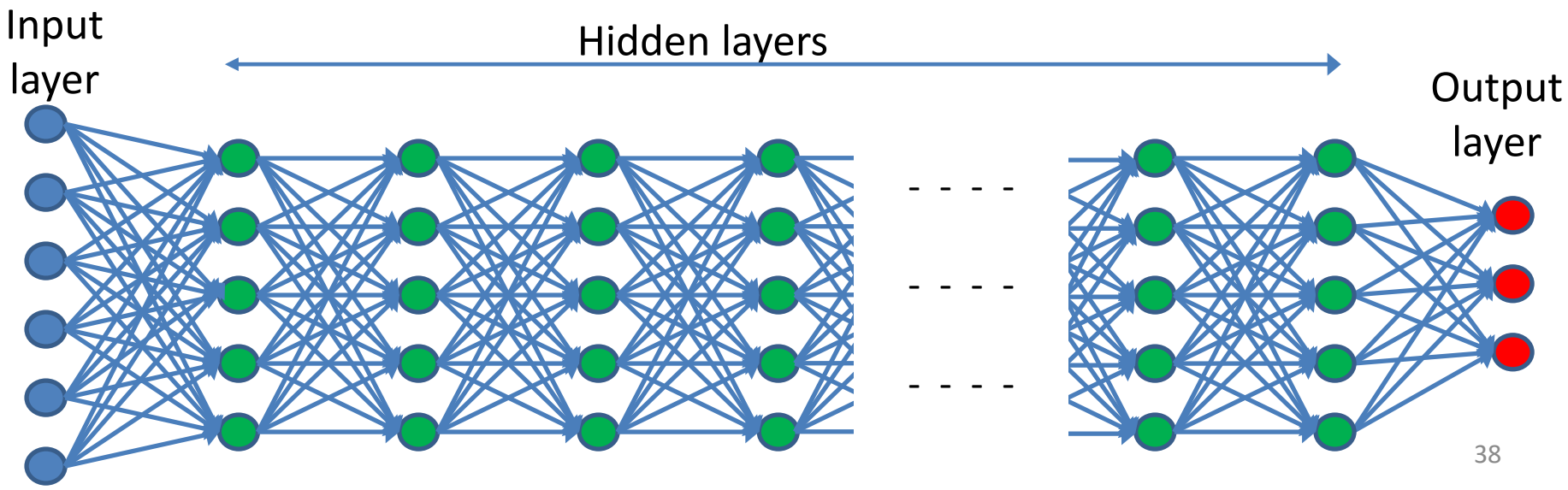
Neural Networks and Deep Learning

- Deep Learning models are essentially deep neural networks.
- In other words, they are neural networks with a large number of hidden layers.
- In the older days, neural networks might include one or two hidden layers.
- These days they may include over 100 hidden layers.



Deep vs. Shallow Networks

- 2 hidden layers are sufficient for modeling any function.
- However, the number of perceptrons in each layer may need to be extremely large.
- Deeper networks may need a smaller total amount of perceptrons for the same task.
  - Layers allow for modularization/factorization of a learning problem.



Training a Neural Network

- Training a neural network is done using an algorithm called **backpropagation**.
- Backpropagation is essentially gradient descent.
- Pseudocode (simplified, and ignoring sophisticated variations):

Repeat until convergence:

Pick a training example x_j, with desired output t_j.

Compute the output o_j of the neural network on x_j.

Measure the **error** E_j between o_j and t_j.

Compute the partial derivatives $\frac{\partial \mathrm{E}_j}{\partial \mathrm{w}_i}$ for every weight w_i of the network.

Update each weight: $\mathrm{w}_i \leftarrow \mathrm{w}_i - \alpha \frac{\partial \mathrm{E}_j}{\partial \mathrm{w}_i}$, where α is called the **learning rate**.

On Training Neural Networks

- Backpropagation converges to a local optimum.
  - It is unlikely to find the universally best solution.
- The billion dollar question: how likely is training to converge to a useful solution?

On Training Neural Networks

- Before the 2010s, most people were skeptical about the practicality of training large neural networks.
  - Even shallow neural networks had a large number of parameters.
  - It took lots of data and lots of time to estimate good values for such networks.
  - Deeper networks made that problem even harder, because they had even more parameters.
- Now we know that deep neural networks are indeed trainable, actually better than other models.
  - Prerequisite: a sufficient (i.e., very large) amount of training data.

Neural Networks, pre-2010

- Before 2010, neural networks were usually not the first choice for people doing research in AI (computer vision, speech recognition, …).
- Other methods were more popular, gave usually better results:
  - Boosting methods.
  - Decision trees and random forests.
  - Support vector machines.
  - Probabilistic graphical models (e.g., Hidden Markov Models for speech recognition).

Post 2010 – What Changed

- Some deep learning methods that are now popular were known in the 1990s.
  - Convolutional Neural Networks proposed by Yann LeCun in the 1990s [LeCun95].
- Limitations in hardware did not allow using and evaluating such methods with sufficient training data.
- Improved hardware (CPUs, GPUs, RAM) eventually allowed experiments with sufficiently large datasets.
- In computer vision, deep neural networks started outperforming other methods by large margins around 2012.

Post 2010 – What Changed

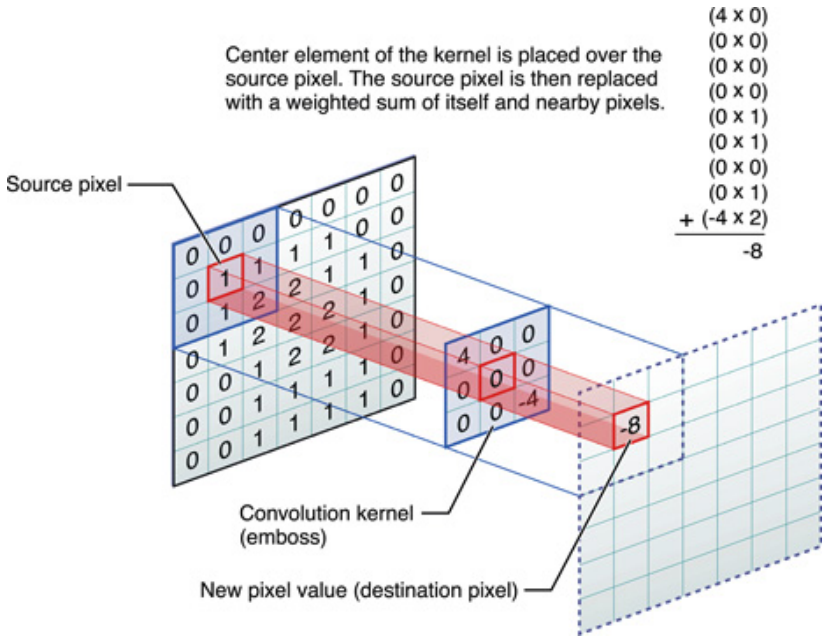
- Initial successes started a positive feedback loop:
  - More researchers started working on improving deep neural networks.
  - New improvements led to even better results.
  - Even better results encouraged even more people to start using deep neural networks…
- Continuing improvements in hardware, and continuing algorithmic improvements allow the use of ever deeper models, with even better results:
  - 7 hidden layers in 2012.
  - 152 hidden layers in 2016.

Performance vs. Size of Data

- Pre-2010, relatively smaller training sets:
  - Other machine learning methods tended to outperform neural network methods.
- Post-2010, larger training sets:
  - Deep neural networks get a big bump in accuracy, and outperform other methods.
  - Compared to other methods, deep neural networks are more capable of taking advantage of large amounts of data.

Convolutional Neural Networks

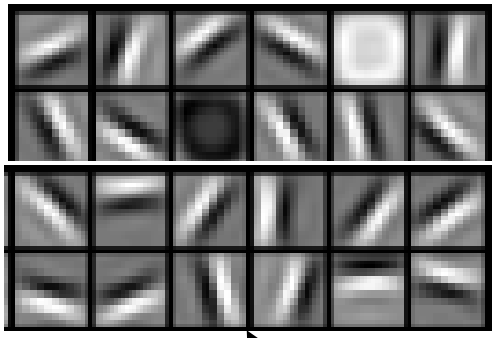
- Convolutional Neural Networks (CNNs) are the dominant paradigm in deep learning models for computer vision.
  - Originally proposed by Yann LeCun in the 1990s [LeCun95].
- Hidden layers perform convolutions on their input.
  - Convolutional layers have a far smaller number of parameters, compared to fully-connected layers.
  - Fewer parameters can be trained with smaller amounts of data.

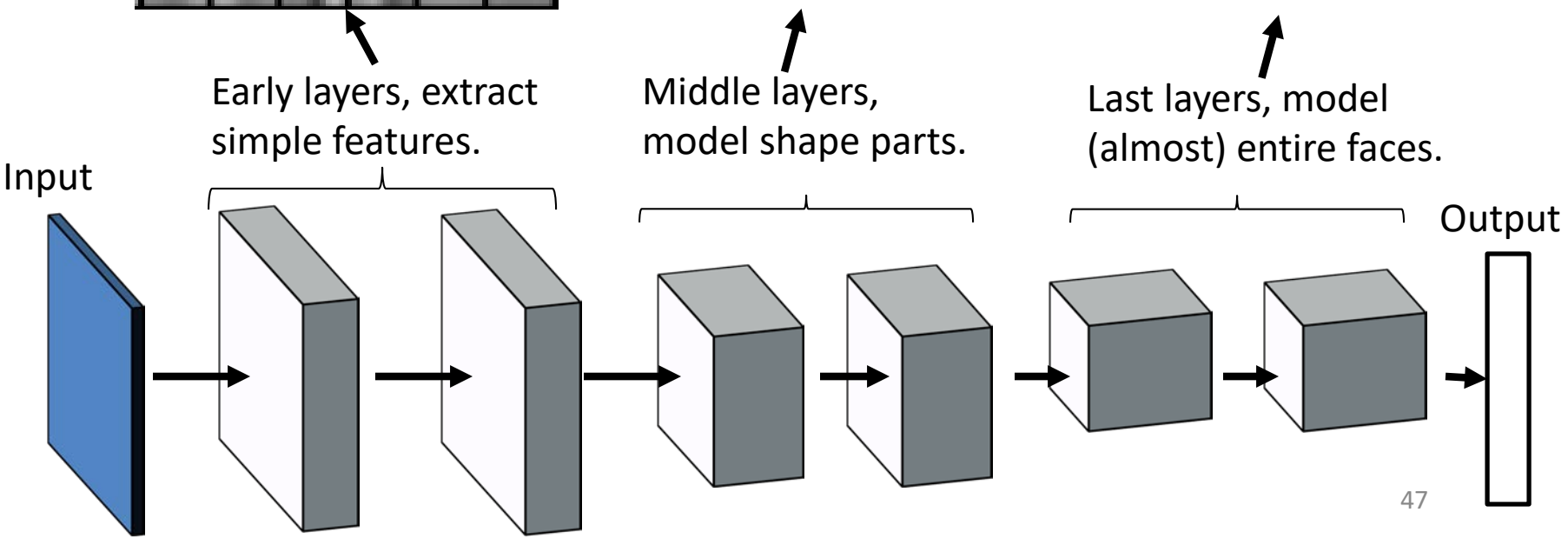


Credit: figure from apple.com developer website.

CNN Visualization

- Visualization of a deep neural network trained with face images.
  - Credit for feature visualizations: [Lee09] Honglak Lee, Roger Grosse, Rajesh Ranganath and Andrew Y. Ng., ICML 2009.





Transfer Learning with CNNs

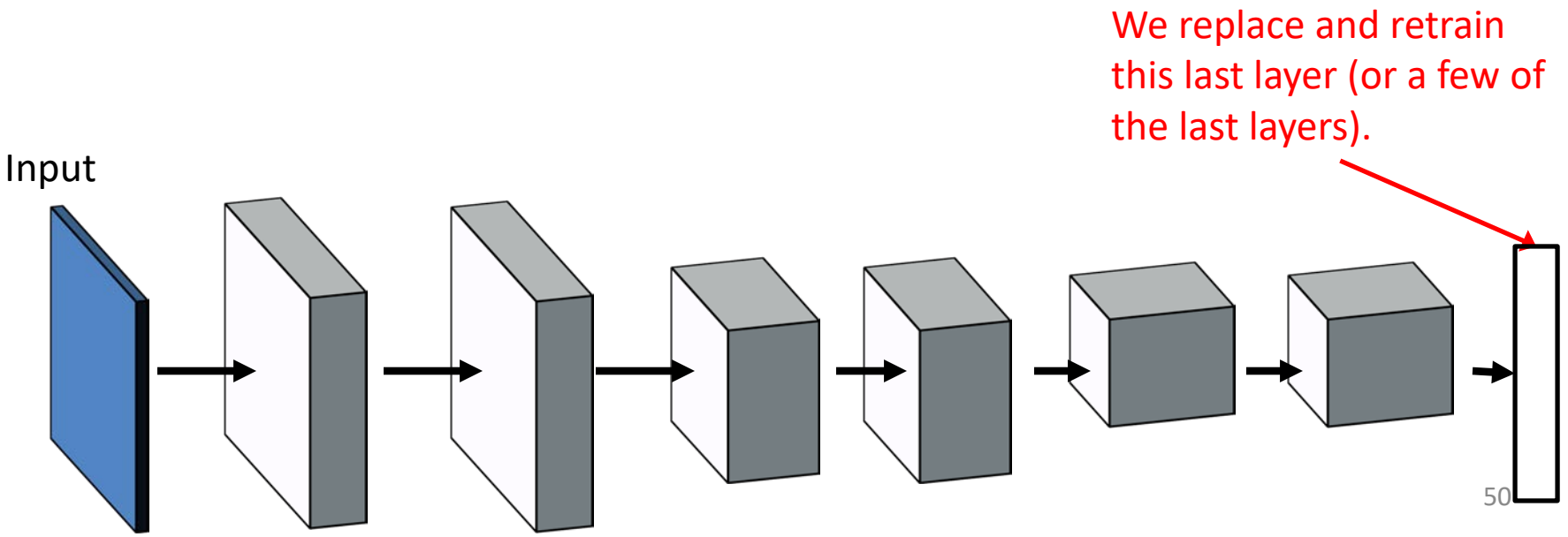
- Suppose you have a very large training set:
  - with millions images
  - thousands of categories
  - thousands of examples per category.
- Suppose you only want to recognize two classes:
  - Images of cats.
  - Images of dogs.
- However, you only have few example images of cats and dogs.
- What do you do?

Transfer Learning with CNNs

- Old-fashioned way:
  - Learn some classifier using your few training examples of cats and dogs.
  - Typically limited accuracy, due to small number of training examples.

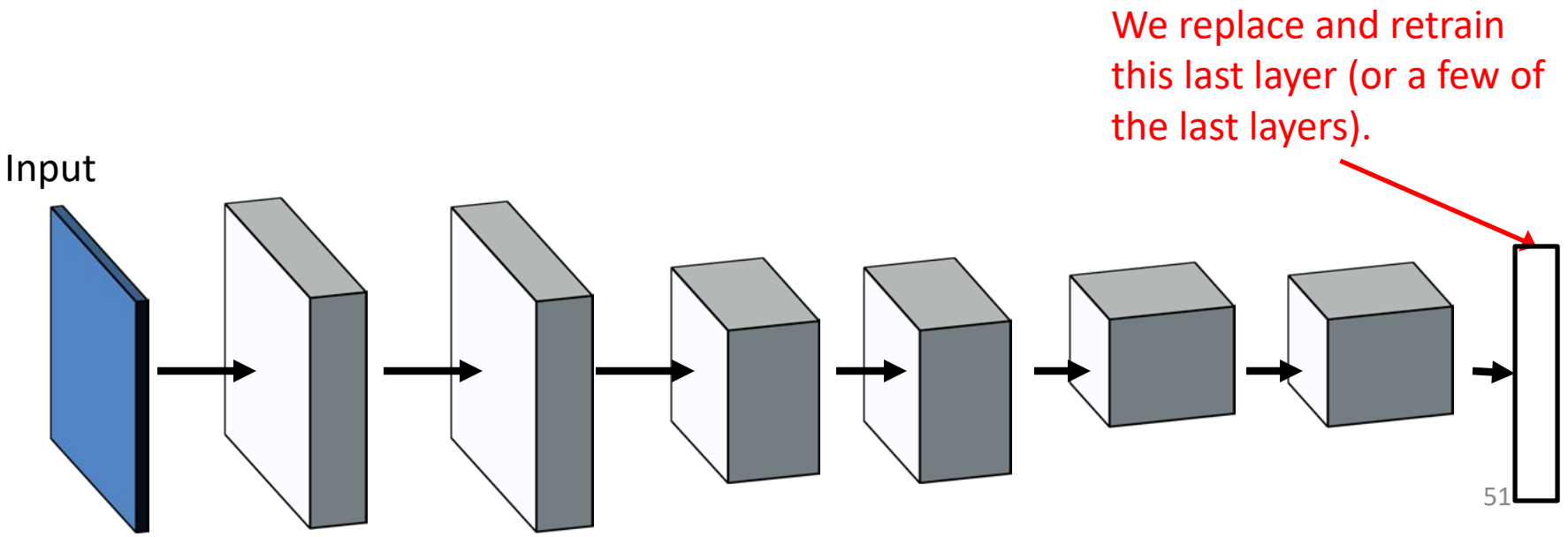
Transfer Learning with CNNs

- The deep learning way: transfer learning.
  - Learn a deep model on your large datasets of millions of examples and thousands of categories.
  - Replace the last layer of the deep model, that recognizes those thousands of categories, with a new layer that only recognizes cats and dogs.
  - Train the new layer using the few examples of cats and dogs.



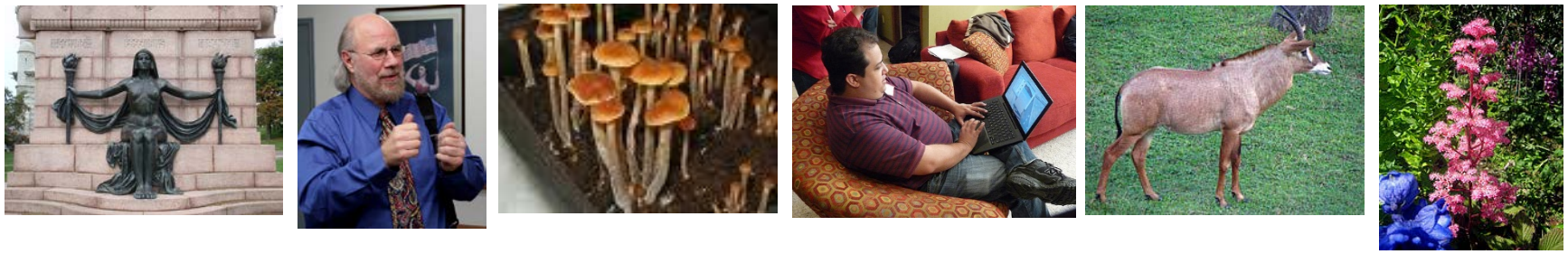
Transfer Learning with CNNs

- This is called transfer learning.
  - We transfer knowledge extracted from one large dataset to another small dataset.
  - The layers we keep have learned useful features (shape parts that are building blocks of objects) from millions of examples.
  - We just create a new last layer, that learns how to distinguish cats and dogs based on those useful features.



Success Stories – ImageNet

- Task: classify each image into over 1000 categories.
- Before deep learning: 25% error rate.
- 2012: deep learning model, 16% error rate.
  - Had a huge impact in the computer vision community.
  - Reference: [Kriz12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton, NIPS 2012.
- Now: under 5% error rate.



Images from the ImageNet dataset.

Success Stories – Face Recognition

- 99.6% accuracy in recognizing a dataset of 5,000 individuals.
  - Reference: [Kem16] Ira Kemelmacher-Shlizerman, Steve Seitz, Daniel Miller, Evan Brossard, CVPR 2016.
- Face verification used for unlocking smartphones and laptops.
- Social media websites identify people in photos.
  - Google Photos, Facebook…

Images from the MegaFace dataset.

Success Stories - Translation

- Google Translate switched in 2015 to a deep learning model.
- Accuracies improved significantly, according to ratings provided by humans (a score of 6 means “perfect”).

From	To	Previous model	Deep learning	Human translation
English	Spanish	4.885	5.428	5.504
English	French	4.932	5.295	5.496
English	Chinese	4.035	4.594	4.987
Spanish	English	4.872	5.187	5.372
French	English	5.046	5.343	5.404
Chinese	English	3.694	4.263	4.636

Results from: [Wu16] Yonghui Wu et al., “Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation”, at arxiv.org, 2016.

Limitations – Amounts of Data

- Deep learning methods do great when lots of training data is available.
- However, it is often hard to find sufficient amounts of data.
  - Example: for face recognition, Google uses a proprietary dataset with tens of millions of individuals. As a university, we cannot compete with that.
  - Example: for American Sign Language recognition, NO current dataset is sufficiently comprehensive to train an acceptably accurate general-purpose translation system.
- Humans need less data to learn, so there is room for improvement.

Limitations - Accuracy

- In most tasks, accuracy is still far from being on par with human accuracy. Just a few examples:
- Mistakes in face detection:

Examples from MegaFace website

- Mistakes in machine translation:
  - Google Translate, input in Greek:
    - Δε φτάνει να θέλεις να κάνεις κάτι, πρέπει και να το μπορείς.
  - Google Translate output:
    - You do not want to do something, you have to.

Limitations - Accuracy

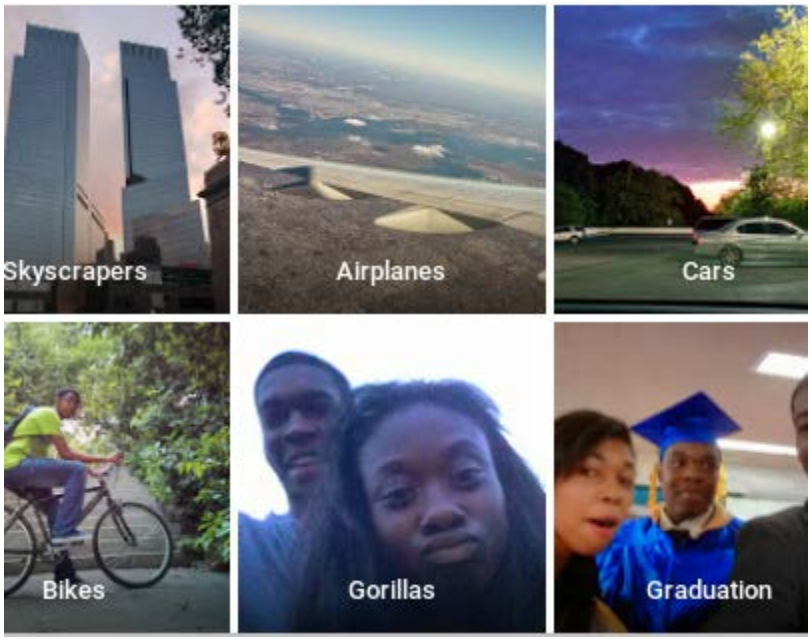
- In most tasks, accuracy is still far from being on par with human accuracy. Just a few examples:
- Mistakes in face detection:

Examples from MegaFace website

- Mistakes in machine translation:
  - Google Translate, input in Greek:
    - Δε φτάνει να θέλεις να κάνεις κάτι, πρέπει και να το μπορείς.
  - Google Translate output:
    - You do not want to do something, you have to.
  - My translation:
    - It is not enough to want to do something, you also have to be able to do it.

Limitations - Bias

- Example: black people labeled as gorillas by Google photos.



Picture retrieved from https://twitter.com/jackyalcine/status/615329515909156865
See articles at Mashable, New York Magazine, The Guardian.

Limitations - Bias

- Example: a language model was trained to complete analogies, such as:
  - Input: Paris is to France as Rome is to ???
  - Output: Italy.
- That model provided some heavily biased answers:
  - Input: Man is to computer programmer as woman is to ???
  - Output:

Limitations - Bias

- Example: a language model was trained to complete analogies, such as:
  - Input: Paris is to France as Rome is to ???
  - Output: Italy.
- That model provided some heavily biased answers:
  - Input: Man is to computer programmer as woman is to ???
  - Output: homemaker.

Limitations - Bias

- Example: a language model was trained to complete analogies, such as:
  - Input: Paris is to France as Rome is to ???
  - Output: Italy.
- That model provided some heavily biased answers:
  - Input: Man is to computer programmer as woman is to ???
  - Output: homemaker.
  - Input: Man is to doctor as woman is to ???
  - Output:

Limitations - Bias

- Example: a language model was trained to complete analogies, such as:
  - Input: Paris is to France as Rome is to ???
  - Output: Italy.
- That model provided some heavily biased answers:
  - Input: Man is to computer programmer as woman is to ???
  - Output: homemaker.
  - Input: Man is to doctor as woman is to ???
  - Output: Nurse.

Reference: [Bol16] Tolga Bolukbasi, Kai-Wei Chang, James Y. Zou, Venkatesh Saligrama, Adam Tauman Kalai: “Man is to Computer Programmer as Woman is to Homemaker? Debiasing Word Embeddings.” NIPS 2016: 4349-4357

Limitations - Bias

- Bias is a big problem, that needs to be overcome as deep learning models are used to make decisions that affect humans.
  - Systems that are trained with black dogs and white cats think that a white dog is a cat.
  - Similarly, systems trained based on human decisions, learn human biases.
    - Can lead to unfair decisions for loans, release from prison, college admissions, employment…
- An exacerbating factor is the lack of explainability of the output of deep learning models.
  - Decisions are influenced by millions of perceptrons, it is hard to tell if an output was influenced by bias.

Conclusions

- Deep learning models are “deep” in two aspects:
  - In terms of functionality: they are end-to-end models, directly mapping input to output.
  - In terms of design: deep neural networks with many hidden layers.
- For many years, people were skeptical about the practicality of training large neural networks.
- Better hardware, more memory, and more data have demonstrated that deep neural networks significantly outperform other machine learning methods in many tasks.
- Many success stories, many commercial applications:
  - Object recognition, face recognition, machine translation, speech recognition…
- Still, lots of room for improvement:
  - Improving accuracy, training with fewer examples, avoiding bias…

Conclusions

- 20 years ago, computer vision and machine learning were not that relevant to society.
- Now, these areas have huge financial impact, enabling exciting technologies.
- No one can predict the future, but there is potential for much larger impact than what we got so far:
  - Self-driving cars (banning human drivers as too careless and error-prone).
  - Robots doing house chores.
  - More accurate diagnosis from medical data, tests, images.
  - Computer tutors offering personalized help to students…
- We live in exciting times!!!

References

[Bol16] Tolga Bolukbasi, Kai-Wei Chang, James Y. Zou, Venkatesh Saligrama, Adam Tauman Kalai: “Man is to Computer Programmer as Woman is to Homemaker? Debiasing Word Embeddings.” NIPS 2016.

[Kem16] Ira Kemelmacher-Shlizerman, Steve Seitz, Daniel Miller, Evan Brossard: “The MegaFace Benchmark: 1 Million Faces for Recognition at Scale.” CVPR 2016.

[Kriz12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." NIPS 2012.

[LeCun95] Y. LeCun and Y. Bengio: “Convolutional Networks for Images, Speech, and Time-Series.” in Arbib, M. A. (Eds), The Handbook of Brain Theory and Neural Networks, MIT Press, 1995.

[Lee09] Honglak Lee, Roger Grosse, Rajesh Ranganath and Andrew Y. Ng: “Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations.” ICML 2009.

[Wu16] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, Jeffrey Dean: “Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation.” arxiv.org, 2016.