ORIE 4741: Learning with Big Messy Data

Neural Networks

Professor Udell

Operations Research and Information Engineering
Cornell

December 2, 2021

Announcements 12/2/21

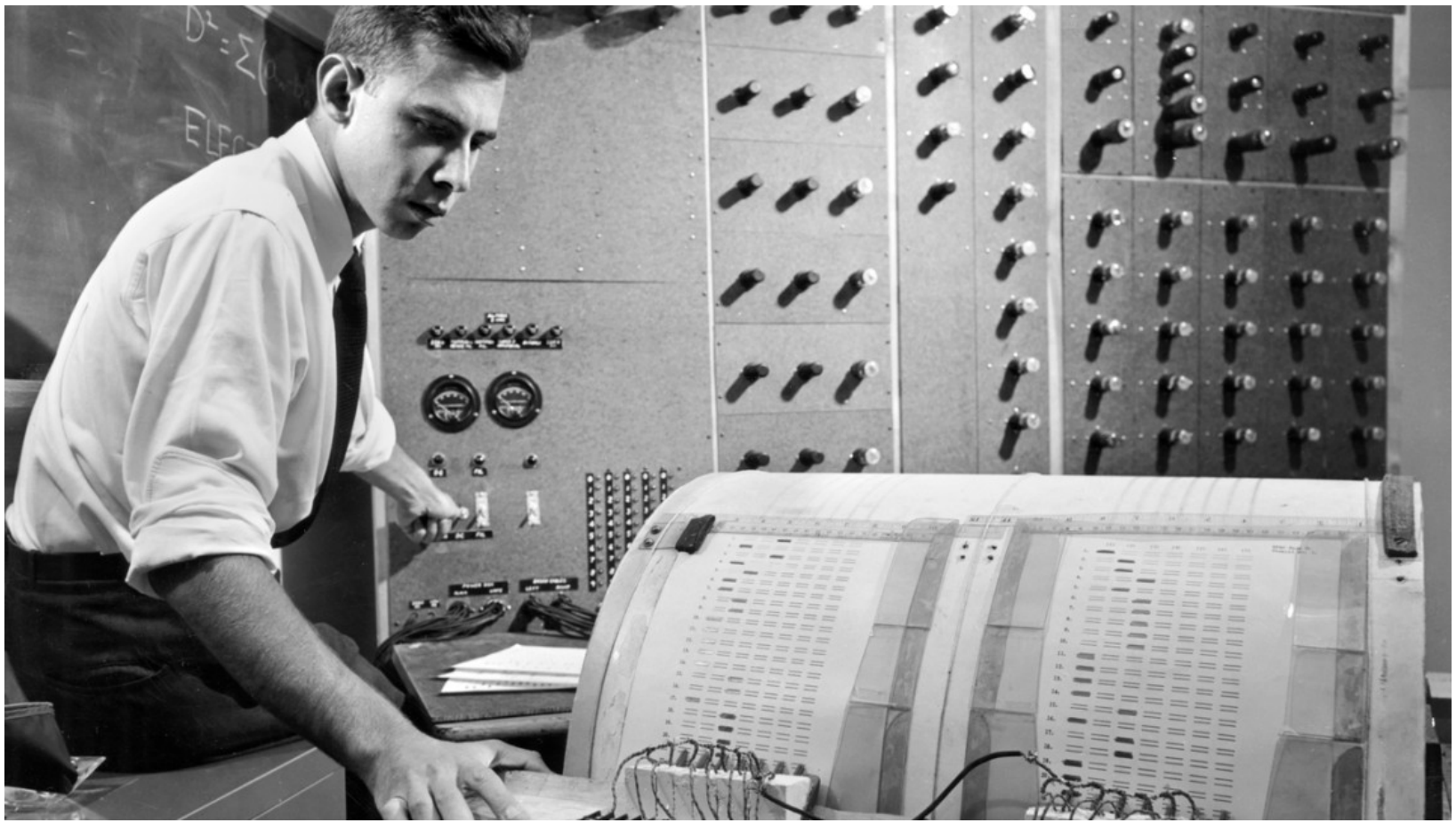
- for ORIE 5741: project presentations due (as video) Friday 12/3/21 11:59pm
  submit by upload to YouTube + add link to github README
- project final report due Sunday 12/5/21 11:59pm
- homework 6 due 9:15am Tuesday 12/7/21
- project peer review due Sunday 12/12/21 11:59pm

Poll

I have trained a neural network on data before

- yes
- no

Neural networks: history



Neural networks: history

- 1958: Frank Rosenblatt's perceptron
- 1959: Widrow and Hoff ADALINE / MADALINE
  - eliminates echoes on phone lines
  - still used today!
- 1969: Minsky and the AI winter: computing with DNNs is too hard!
- 1982: Hopfield net and the thaw: backprop!
- 2000's: more winter
  - AI by other names: informatics, machine learning, analytics, knowledge-based systems, business rules management, cognitive systems, intelligent systems, intelligent agents or computational intelligence
- 2010s+: an AI spring
  - Geoffrey Hinton, Yann LeCun and Yoshua Bengio
  - why now? more computing power + more data

Recall: Perceptron

- $\mathcal{X} = \mathbf{R}^d$, $\mathcal{Y} = \{-1, +1\}$
- data $\mathcal{D} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, $x_i \in \mathcal{X}$, $y_i \in \mathcal{Y}$ for each $i = 1, \ldots, n$

make decision using a linear function

- approve credit if

$$\sum_{j=1}^{d} w_j x_j = w^\top x \geq b;$$

  deny otherwise.
- parametrized by weights $w \in \mathbf{R}^d$
- decision boundary is the hyperplane $\{x : w^\top x = b\}$

The Limits of Linearity

- Linearity $\implies$ monotonicity: increase in feature i causes an increase (for $w_i > 0$) or decrease (for $w_i < 0$) in the prediction $\hat{y}$
- Monotonicity requires feature engineering to represent nonlinearities
- Many real-world problems are nonlinear!

The Limits of Linearity

- **Loan origination.** Should going from an income of \$0 to \$50,000 a year have the same effect as going from \$550,000 to\$600,000?

The Limits of Linearity

- **Loan origination.** Should going from an income of \$0 to \$50,000 a year have the same effect as going from \$550,000 to\$600,000?
- **Severity of illness.** Is a heart rate of 200 BPM or 10 BPM healthier than 70 BPM?

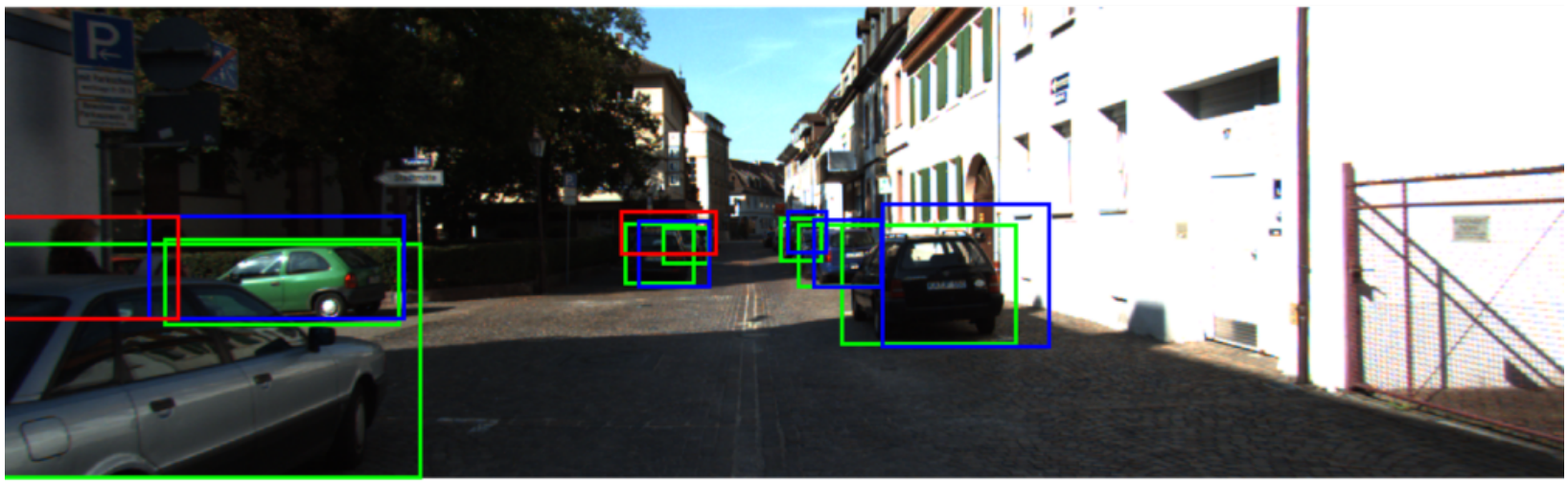
The Limits of Linearity

- **Loan origination.** Should going from an income of \$0 to \$50,000 a year have the same effect as going from \$550,000 to\$600,000?
- **Severity of illness.** Is a heart rate of 200 BPM or 10 BPM healthier than 70 BPM?
- **Valuing used cars.** More miles is worse; but should the value ever be negative?

Neural networks

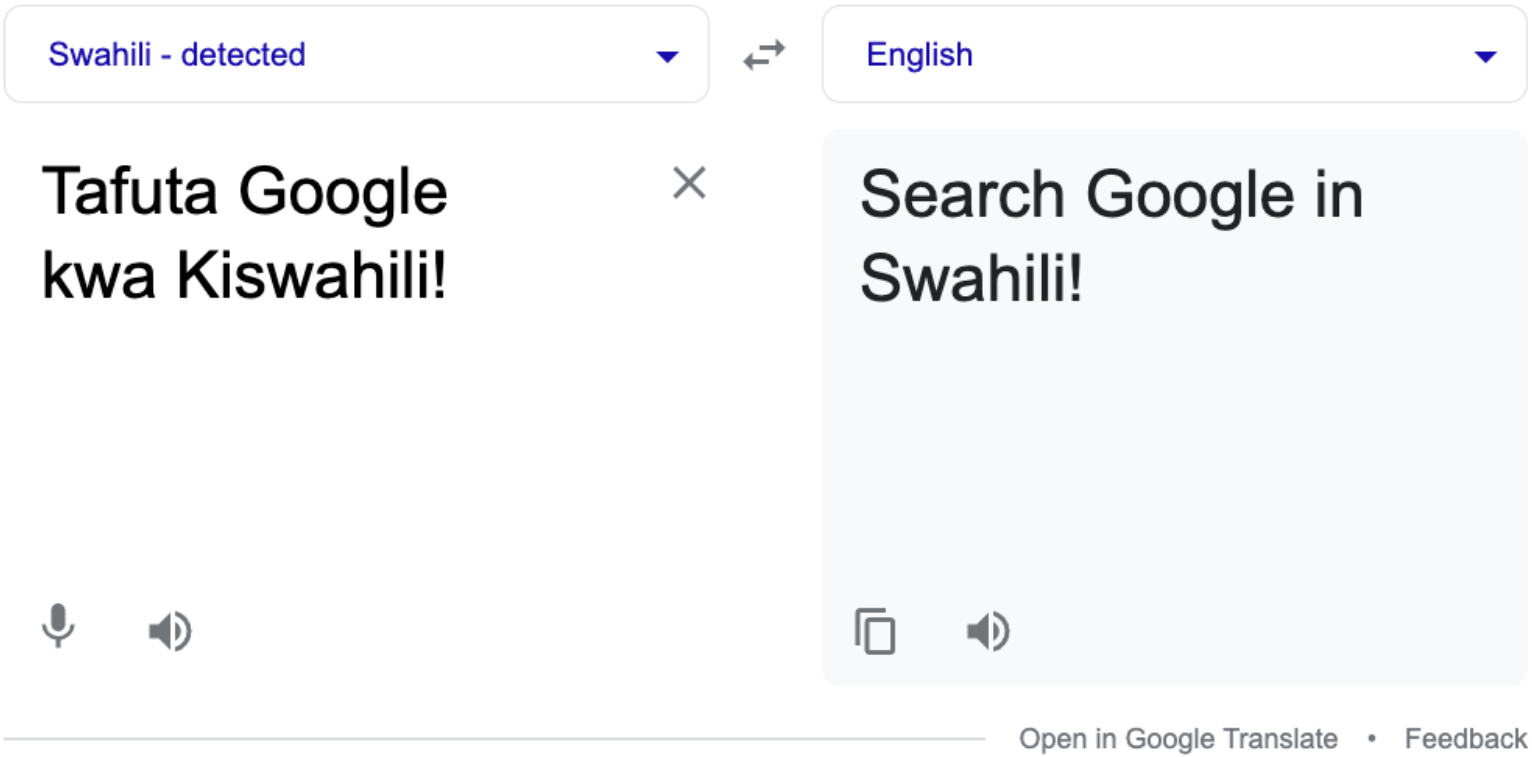
Q: What kind of data needs a neural network?

Image Recognition

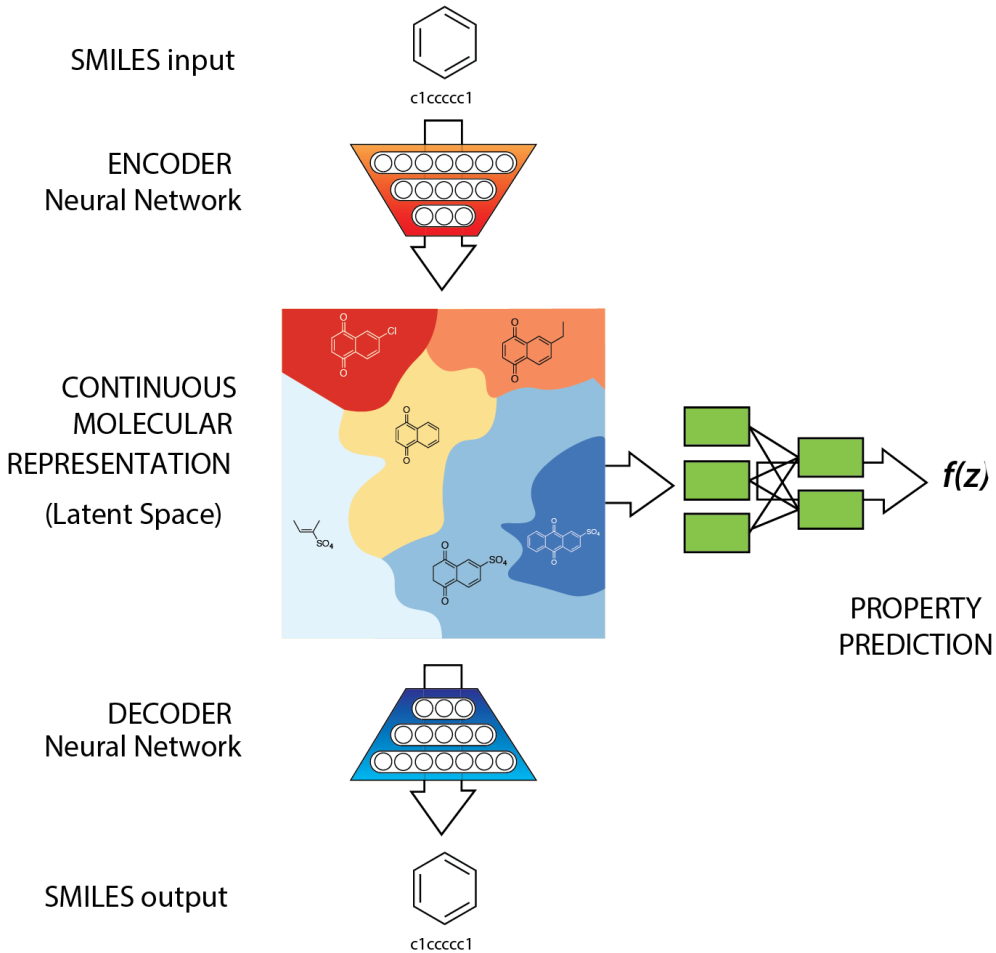


source: "Orientation-boosted Voxel Nets for 3D Object Recognition"
https://arxiv.org/abs/1604.03351

Neural Machine translation



Drug Design



source: "Automatic chemical design using a data-driven continuous representation of molecules" `https://arxiv.org/abs/1610.02415`

Linear Neural Networks

The perceptron is the simplest feedforward neural network.

Q: Can we represent nonlinearities with a perceptron?

Linear Neural Networks

The perceptron is the simplest feedforward neural network.

Q: Can we represent nonlinearities with a perceptron?

Idea: iterate the perceptron map:

$$x_1 = w_1 x + b_1$$
$$x_2 = w_2 x_1 + b_2$$
$$\hat{y} = w_3 x_2 + b_3$$

Definitions and notation

Definition

A **hidden layer** is the output of a perceptron-style map between the input and output layer.

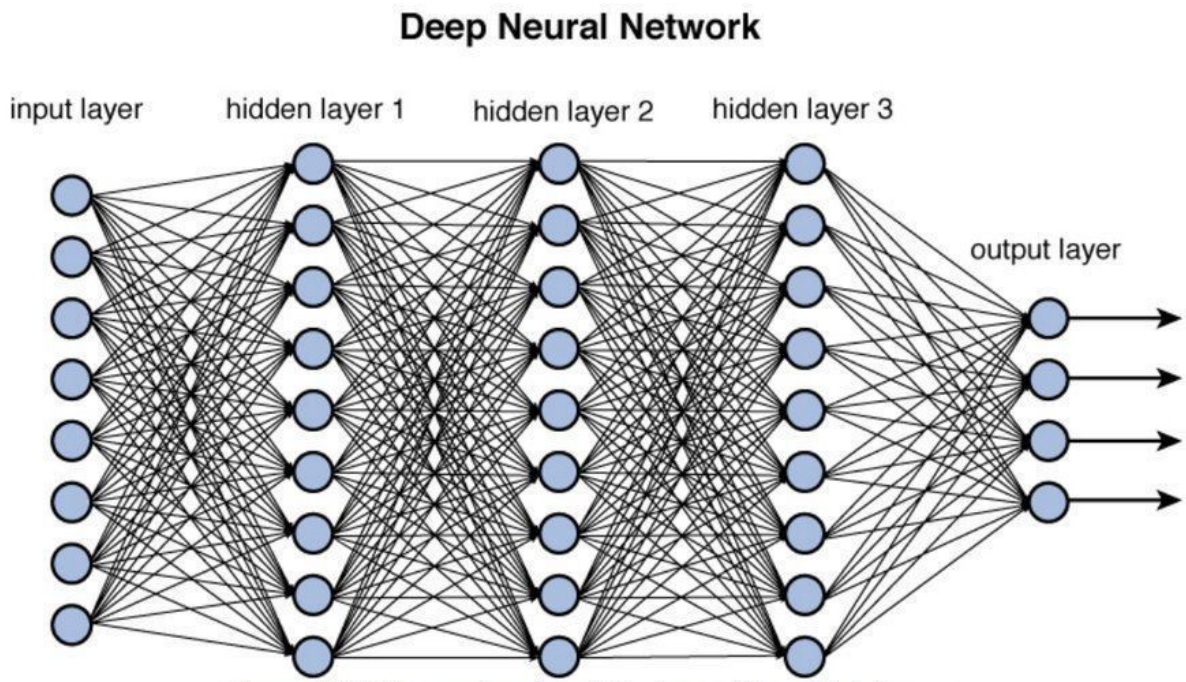


Figure 12.2 Deep network architecture with multiple layers.

Linear Neural Networks

The perceptron is the simplest feedforward neural network.

Q: Do simple multilayer linear neural networks do anything a single layer network can't?

Linear Neural Networks

The perceptron is the simplest feedforward neural network.

Q: Do simple multilayer linear neural networks do anything a single layer network can't?

For a network of linear activations and biases as described before, we can formulate an equivalent linear classifier: if $W = W_1 W_2$ and $b = b_1 W_2 + b_2$, then

$$\begin{aligned}\hat{y} &= (XW_1 + b_1) W_2 + b_2 \\ &= XW_1 W_2 + b_1 W_2 + b_2 = XW + b\end{aligned}$$

Activation Functions

Definition

In a neural network, an **activation function** σ is a function that takes as input the dot product of the linear portion and outputs an **activation**

$$x^{(i)} = \sigma\left(W_i x + b_i\right).$$

Activation Functions

Definition

In a neural network, an **activation function** σ is a function that takes as input the dot product of the linear portion and outputs an **activation**

$$x^{(i)} = \sigma\left(W_i x + b_i\right).$$

Thresholding in a perceptron acts as a nonlinearity. For hidden layers, common nonlinearities include

- ReLU: $\sigma(x) = \max(x, 0)$
- sigmoid: $\sigma(x) = \log(1 + \exp(x))$

Definitions and notation

Definition

A **multilayer, feedforward neural network** is a model in which linear weight vectors are sequentially applied with intermediate activation functions in between each **layer**:

$$\text{NN}(x) = \sigma(W_1\sigma(W_2 \ldots \sigma(W_\ell x))))$$

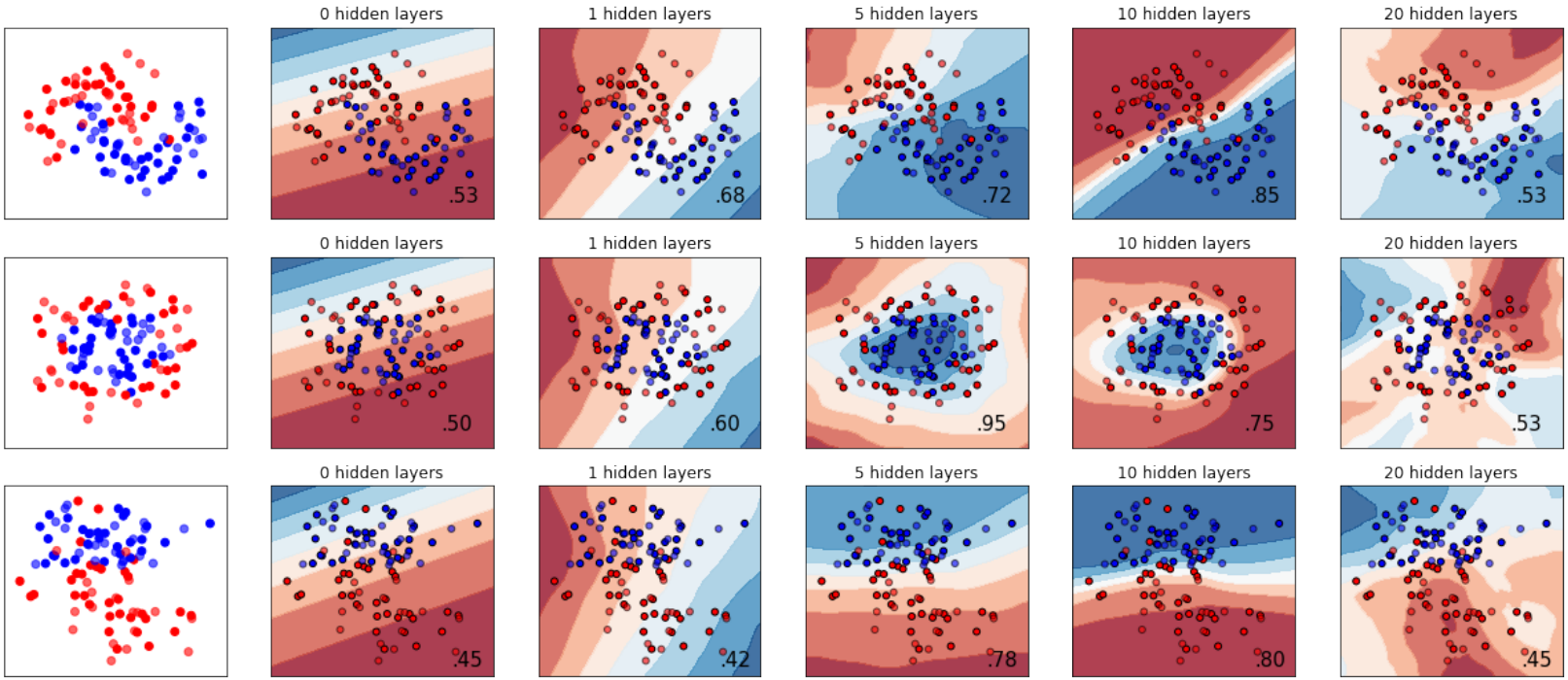
For classification problems, this can be thought of as a sort of **stacked** logistic regression.

Poll

A neural network with 2 hidden layers, each of which has d nodes, is a(n) ___-determined system

- Under
- Over
- Well

Simple Comparison of Expressivity



Visualization

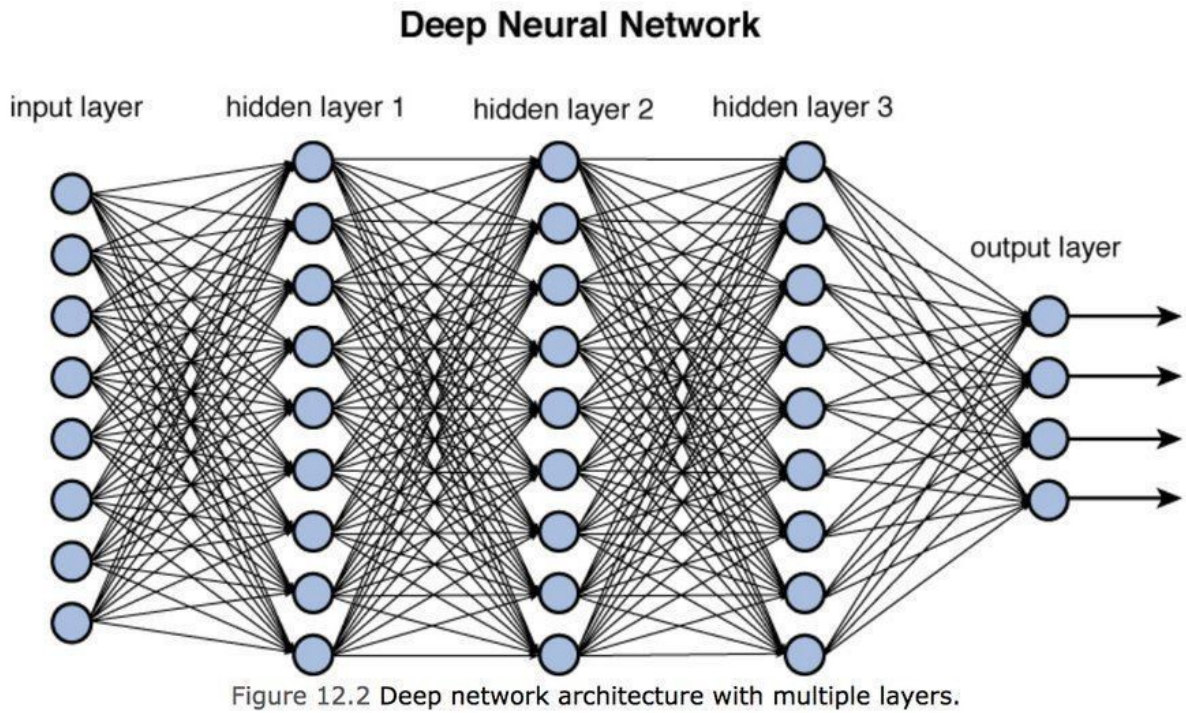


Figure 12.2 Deep network architecture with multiple layers.

Q: What are the hidden layers predicting? How can we train such a model, ie, choose weights for the hidden layers?

Visualization

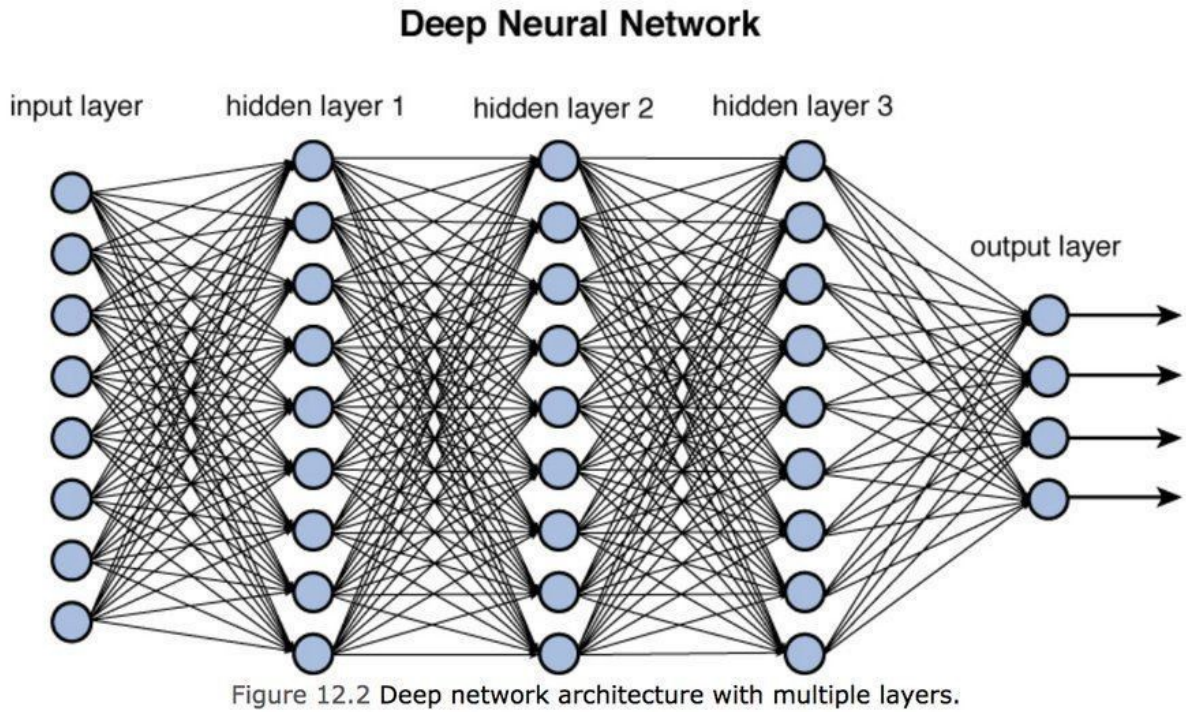


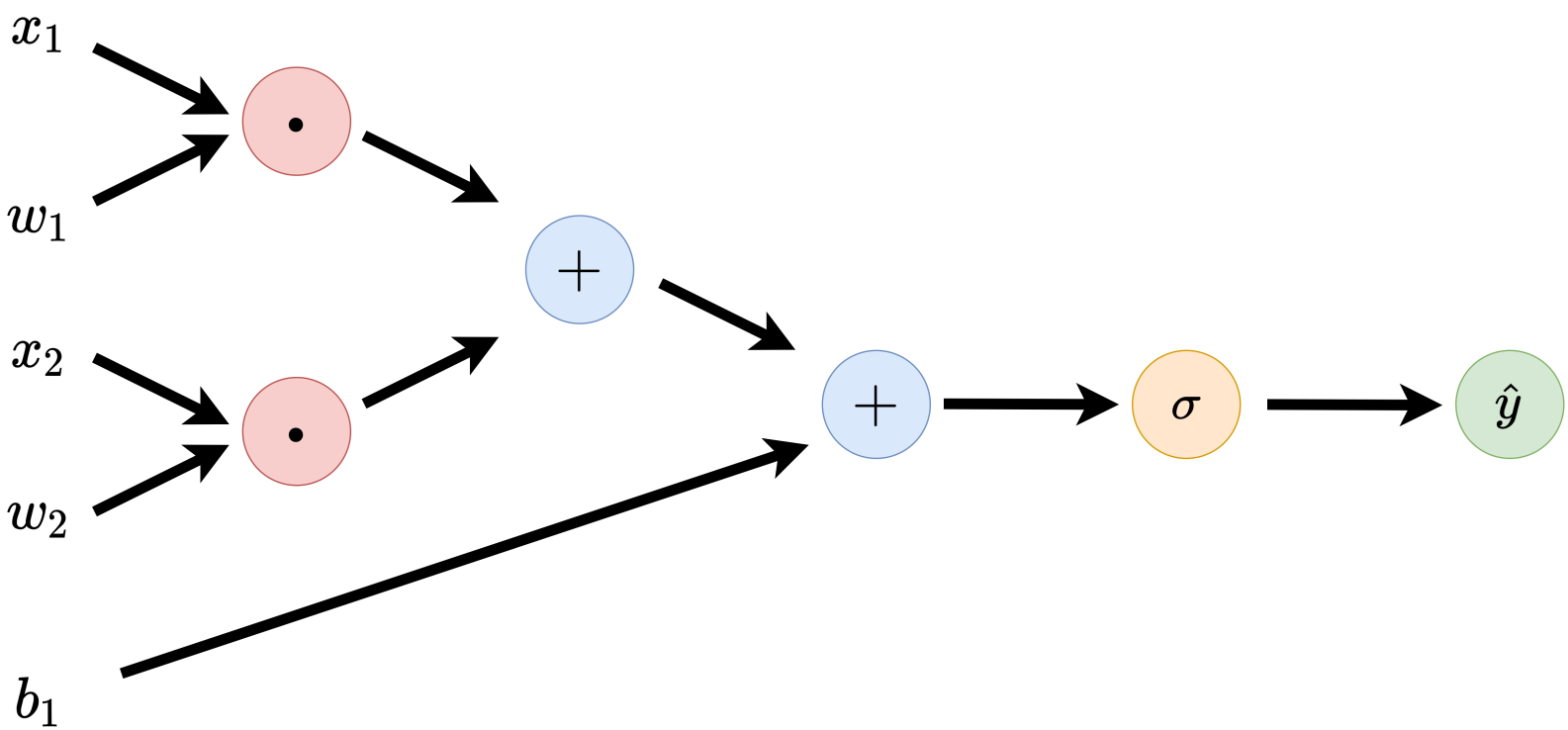
Figure 12.2 Deep network architecture with multiple layers.

Q: What are the hidden layers predicting? How can we train such a model, ie, choose weights for the hidden layers?

A: Gradient descent!

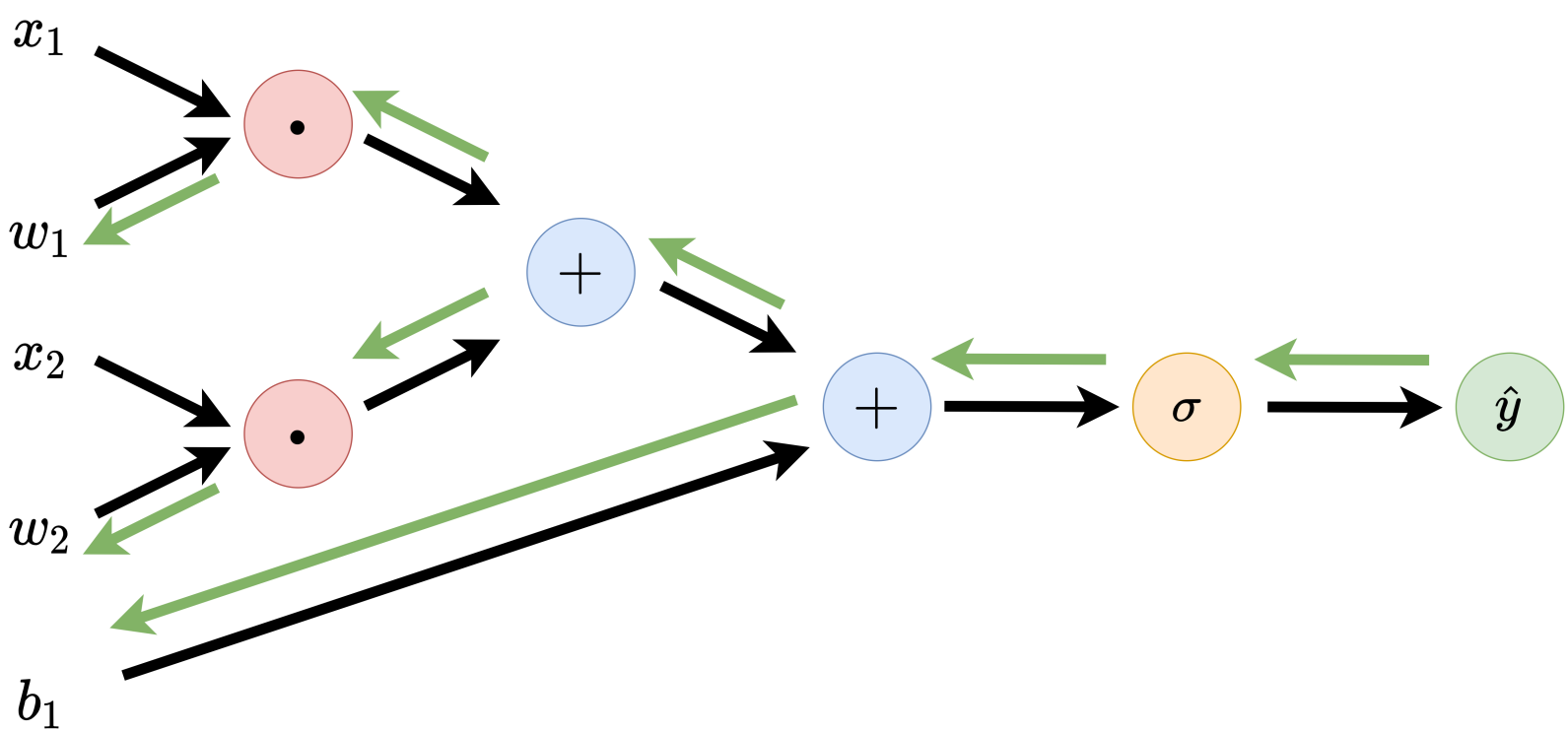
Forward Propagation

Forward propagation is the iterated application of weights and activation functions through a network.

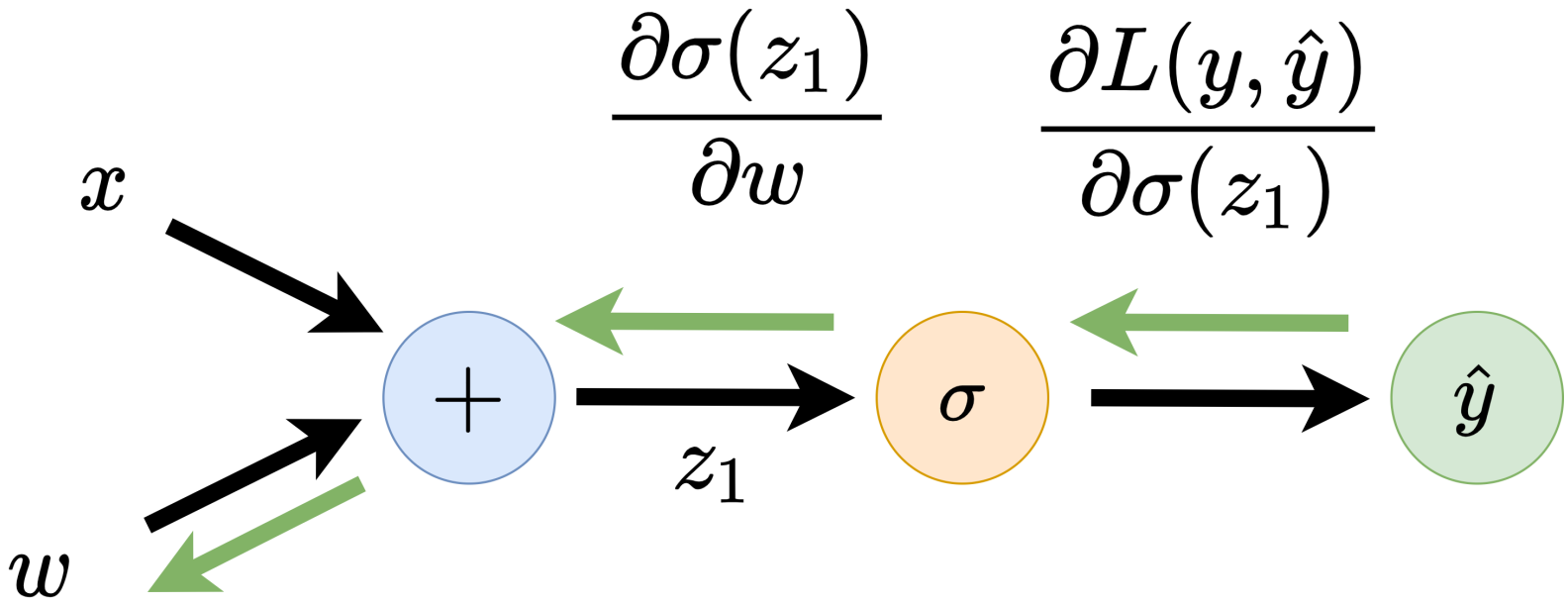


Backpropagation

Backprop propogates information backwards along the connections of the neural network. We can use backprop to efficiently compute the gradient in a neural net.



Applying the Chain Rule of Calculus



Automatic Differentiation

Symbolic differentiation can be unwieldy and slow. Numerical differentiation can introduce rounding errors (especially with higher derivatives). Automatic differentiation (AD) is used because

- AD works for function with differentiable components.
- AD uses computation proportional to forward pass (objective evaluation).
- AD can calculate derivatives to arbitrary precision.

Why GPUs?

- GPUs have 1000s+ cores. For simple, parallel tasks (like matrix multiplication!), this can provide a huge speedup.
- GPUs enable the success of neural networks in the last decade.
- GPUs are optimized for float32 options: add / multiply / MAC operations 64 times faster than for float64.

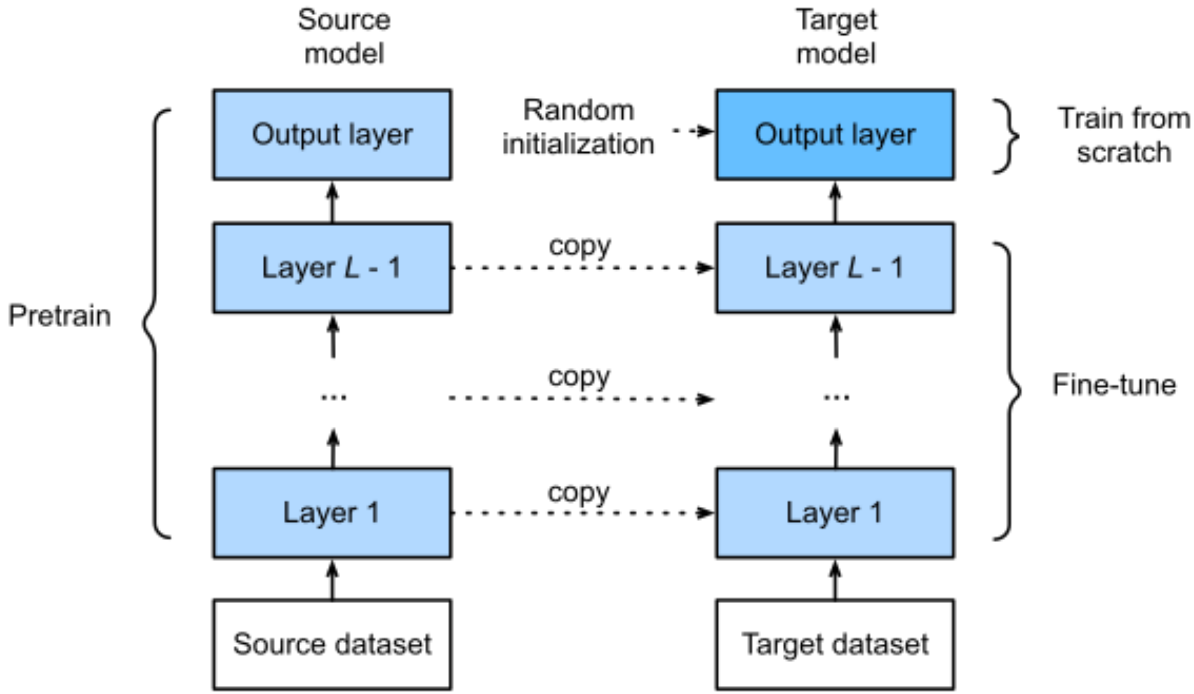
Transfer learning from foundation models

DNNs work better with more training data! so

- first, train on the whole internet
- then, **fine-tune** weights to work for your problem

Vision Models

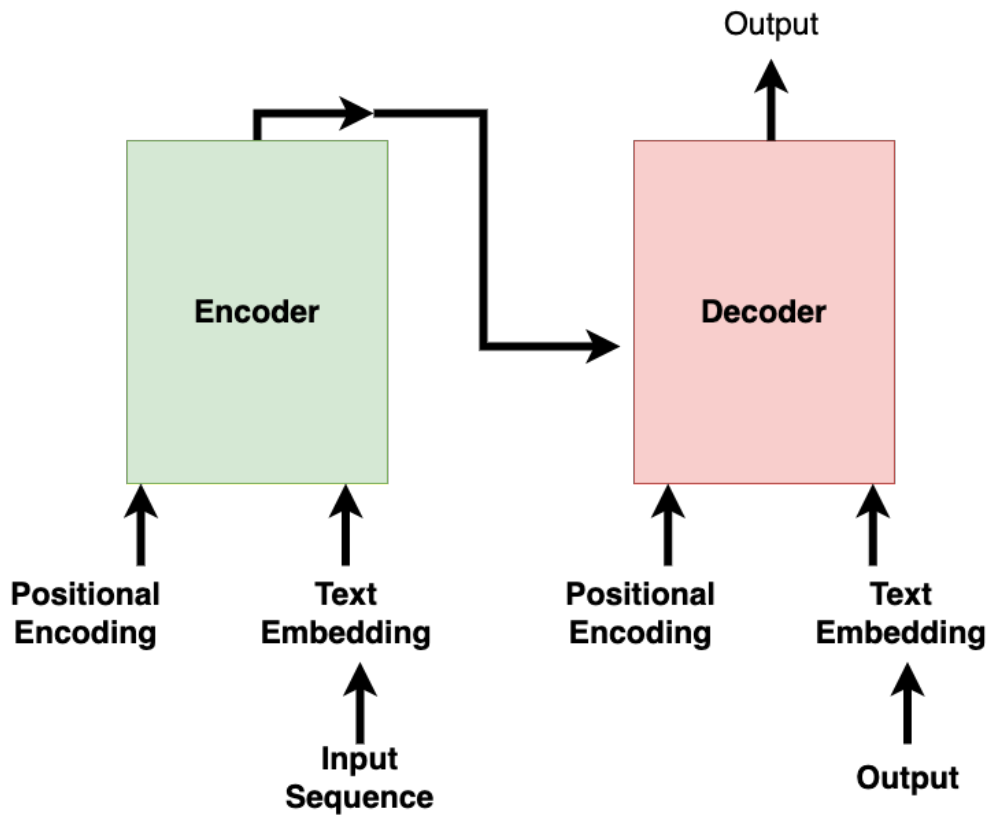
For vision models, standard practice uses object detection models trained on huge datasets to provide feature embeddings of a set of images. We can then apply a simpler model (e.g. an SVM) to these embeddings for classification, learn the top layer, and fine-tune all the weights.



source: "Dive Into Deep Learning"
https://arxiv.org/pdf/1512.03385.pdf

Text Data

State-of-the-art results for text classification can be achieved by fine-tuning pre-trained **transformer models**. These models use fixed sequence length data trained on text sources along with positional encoding information.



Text Data

- Existing baseline models often trained with millions of dollars in compute.
- Minimal tuning needed to specialize to new problem, but there can be a big advantage to doing some.