

Deep Learning Theory and Applications

# Feedforward Networks

Yale

Kevin Moon (kevin.moon@yale.edu)

Guy Wolf (guy.wolf@yale.edu)

CPSC/AMTH 663





1. Feedforward network overview
  - Sigmoid neuron
  - Architecture of a neural network

# Handwritten digit recognition



504 / 92

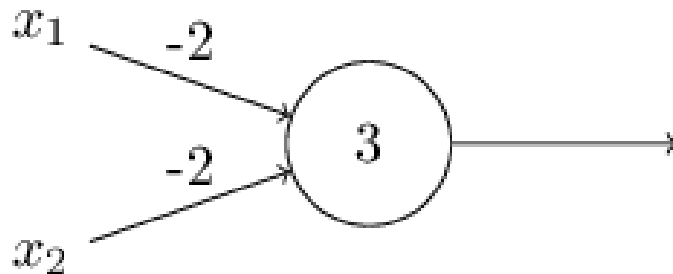
# Handwritten digit recognition



# Logic circuits with perceptrons



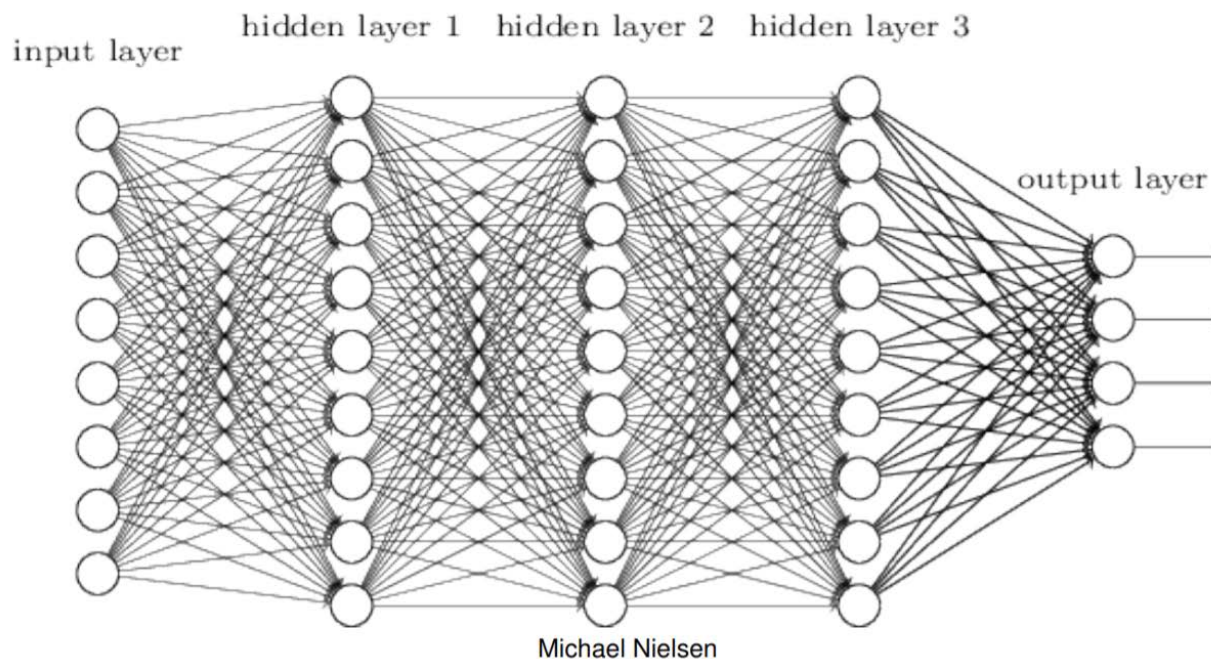
- NAND gates can be constructed from perceptrons
- NAND gates are universal for computation
  - Any computation can be built from NAND gates
  - Therefore, perceptrons are universal for computation



# Feedforward Network



- We can create *learning algorithms* that automatically tune the weights and biases
  - Tuning occurs in response to external stimuli and w/o direct intervention
  - Creates a circuit designed for the problem at hand

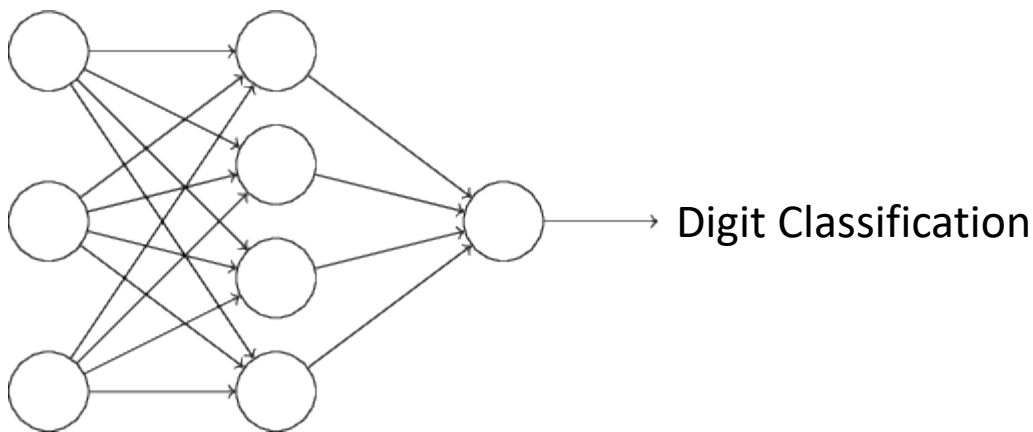


# Learning the weights and biases



- Goal: use a network of perceptrons to learn to solve a problem
- Example: learn the weights and biases in a network to perform handwritten digit recognition

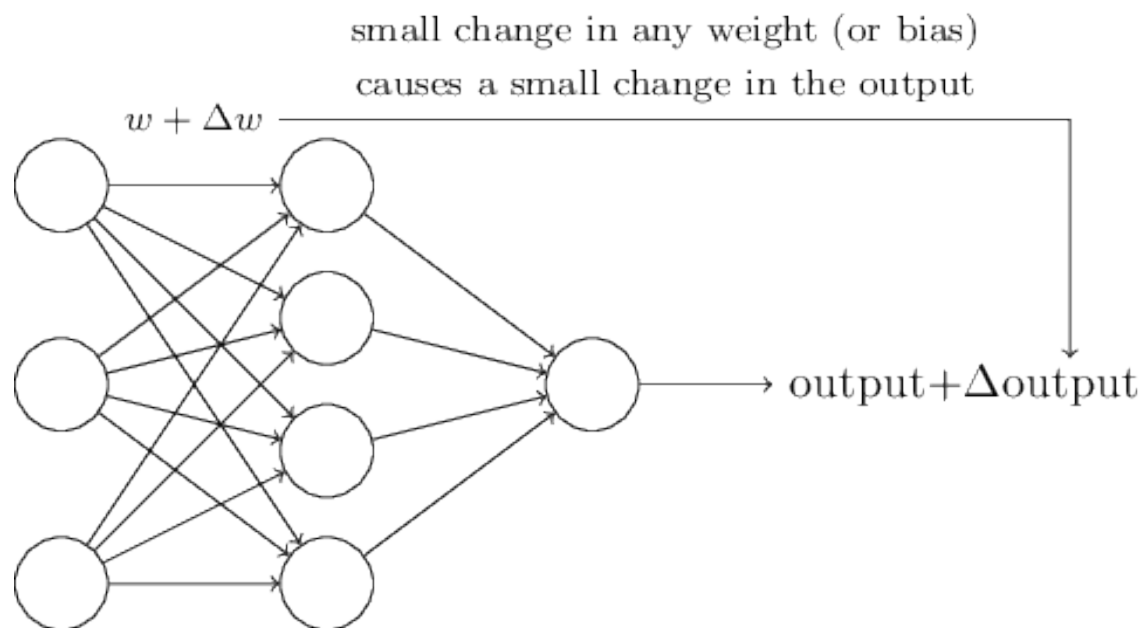
Input



# Learning the weights and biases



- Want small changes in weights or biases to result in small changes in output





# Learning the weights and biases

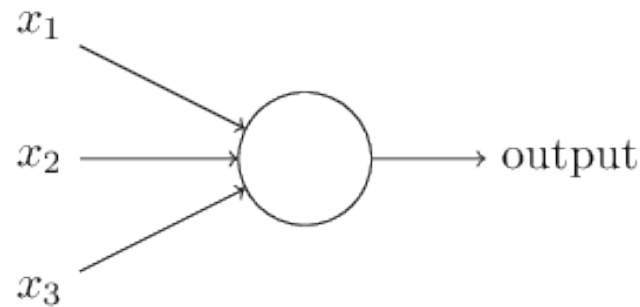


- Want small changes in weights or biases to result in small changes in output
- Example: suppose network misclassifies an “8” as a “9”
  - Make small changes in weights and biases to get network a little closer to correct classification
  - Repeat to produce better and better output (i.e. learn)
- **Problem:** this doesn’t happen with perceptron networks
  - Small changes in weights and bias of a single perceptron may cause output to flip (big change)
  - This flip may create complicated changes in the network
  - So even if we now correctly classify the “9”, the network behavior on all other images may be drastically different
  - Difficult to learn appropriate weights and biases

# Sigmoid Neuron



- A ***sigmoid neuron*** is a modified perceptron so that small changes to weights and biases  $\Rightarrow$  small changes in output



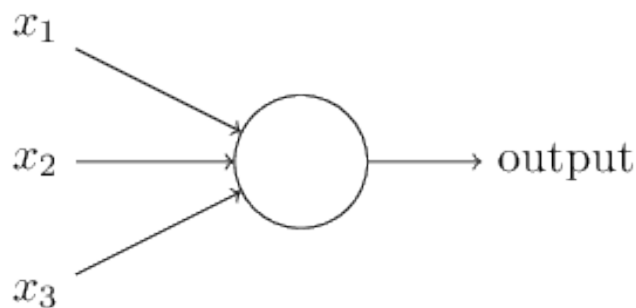
- Inputs may be *between* 0 and 1
  - E.g. 0.638 is a valid input
- Weights  $w_i$  for each input and bias  $b$
- Output is  $\sigma(w \cdot x + b)$

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

# Sigmoid Neuron



- A ***sigmoid neuron*** is a modified perceptron so that small changes to weights and biases  $\Rightarrow$  small changes in output



- Output is

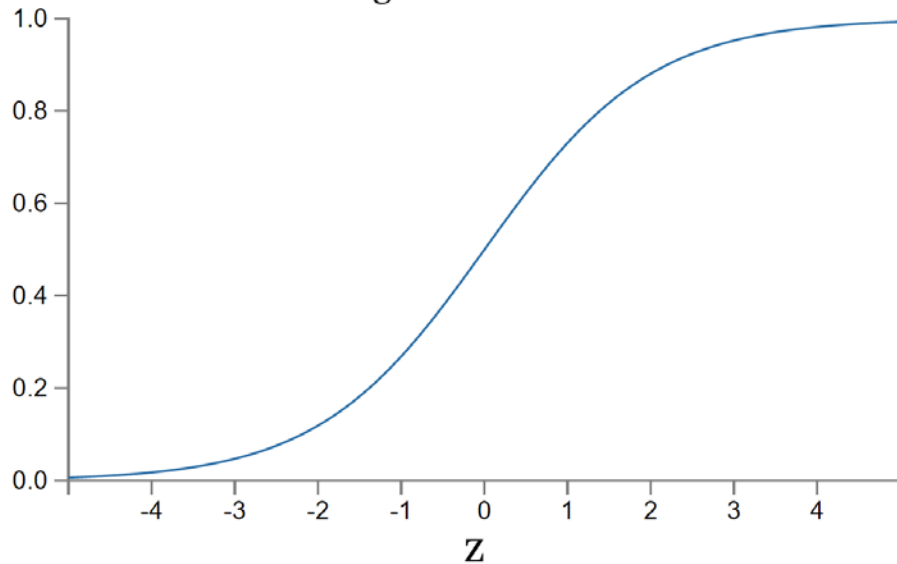
$$\sigma(w \cdot x + b) = \frac{1}{1 + \exp(-\sum_j w_j x_j - b)}$$



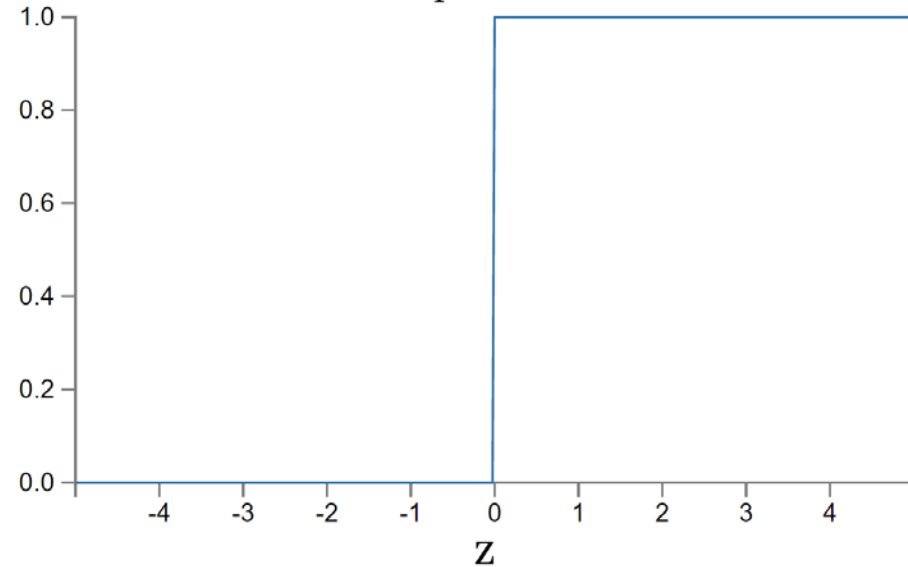
- $\sigma(z) = \frac{1}{1+e^{-z}}$
- Suppose  $z = w \cdot x + b$  is large and positive
  - $e^{-z} \approx 0 \Rightarrow \sigma(z) \approx 1$
  - I.e. if  $z = w \cdot x + b$  is large and positive, the sigmoid neuron output is approximately 1 (like a perceptron)
- Suppose  $z = w \cdot x + b$  is large and negative
  - $e^{-z} \rightarrow \infty \Rightarrow \sigma(z) \approx 0$
  - I.e. if  $z = w \cdot x + b$  is large and negative, the sigmoid neuron output is approximately 0 (like a perceptron)
- Some deviation for modest values of  $z = w \cdot x + b$



sigmoid function



step function



- For the perceptron,  $\sigma =$  the step function
- Smoothness of the sigmoid function  $\Rightarrow$  small changes in weights and biases produce small change in output

# Weights, bias, and output



- Change in weight  $\Delta w_j$ , change in bias  $\Delta b$

$$\Delta \text{output} \approx \sum_j \frac{\partial \text{output}}{\partial w_j} \Delta w_j + \frac{\partial \text{output}}{\partial b} \Delta b$$

- $\Delta \text{output}$  is a *linear function* of the changes  $\Delta w_j$  and  $\Delta b$  in the weights and bias
  - Linearity makes it easier to choose small changes in weights and biases to achieve the desired small change in output
- Thus sigmoid neurons have similar behavior as perceptrons but are easier to use

# Activation functions

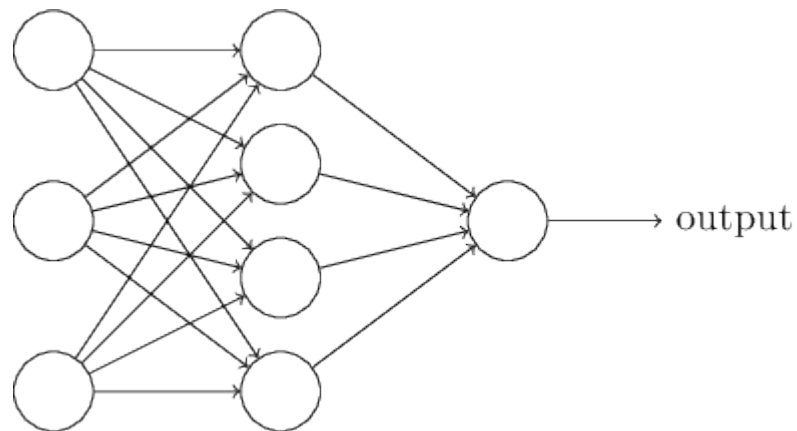


- If smoothness is what we need, why the sigmoid function  $\sigma$ ?
- Using  $\sigma$  simplifies the algebra
- Other ***activation functions***  $f(w \cdot x + b)$  are used in practice
  - Depends on the application
  - More on these later in the semester

# Classification with sigmoid neurons

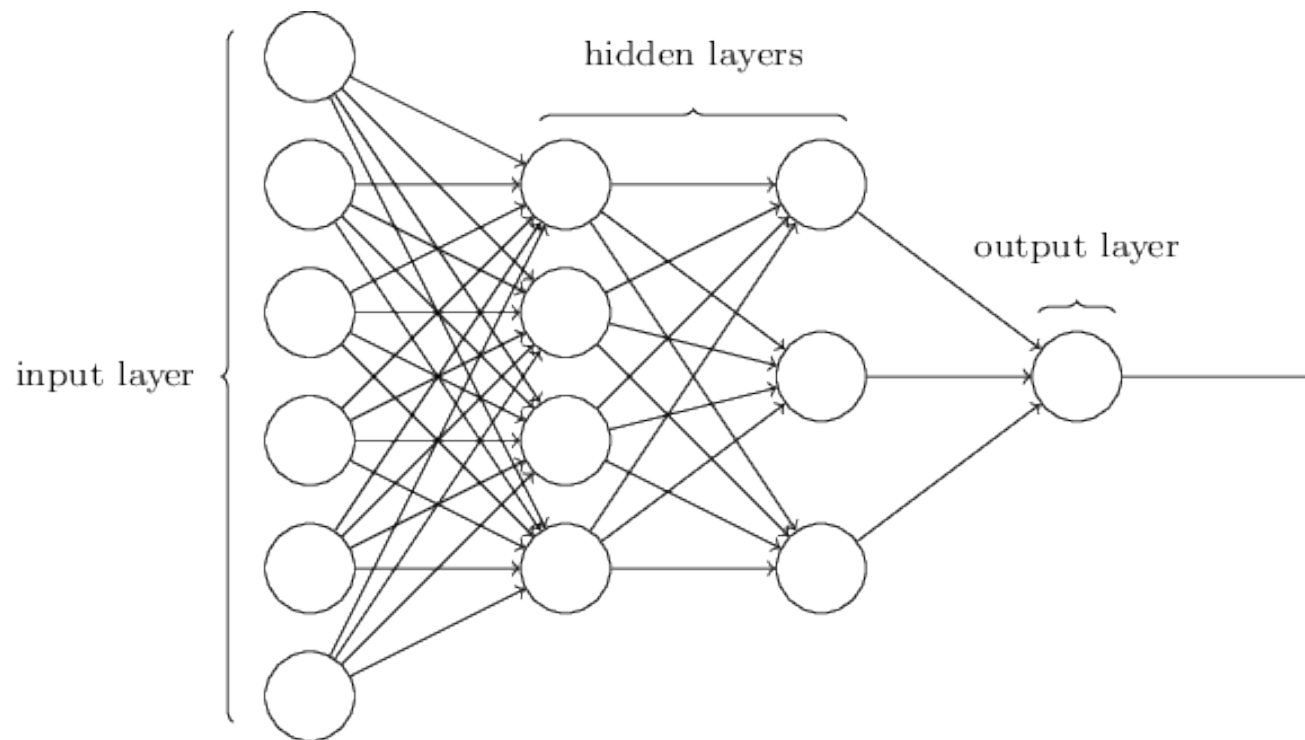


- Sigmoid neuron output = any real number between 0 and 1
- How do we do classification?
- Threshold the final output
  - E.g., a value above 0.5 indicates a “9” while a value below 0.5 does not



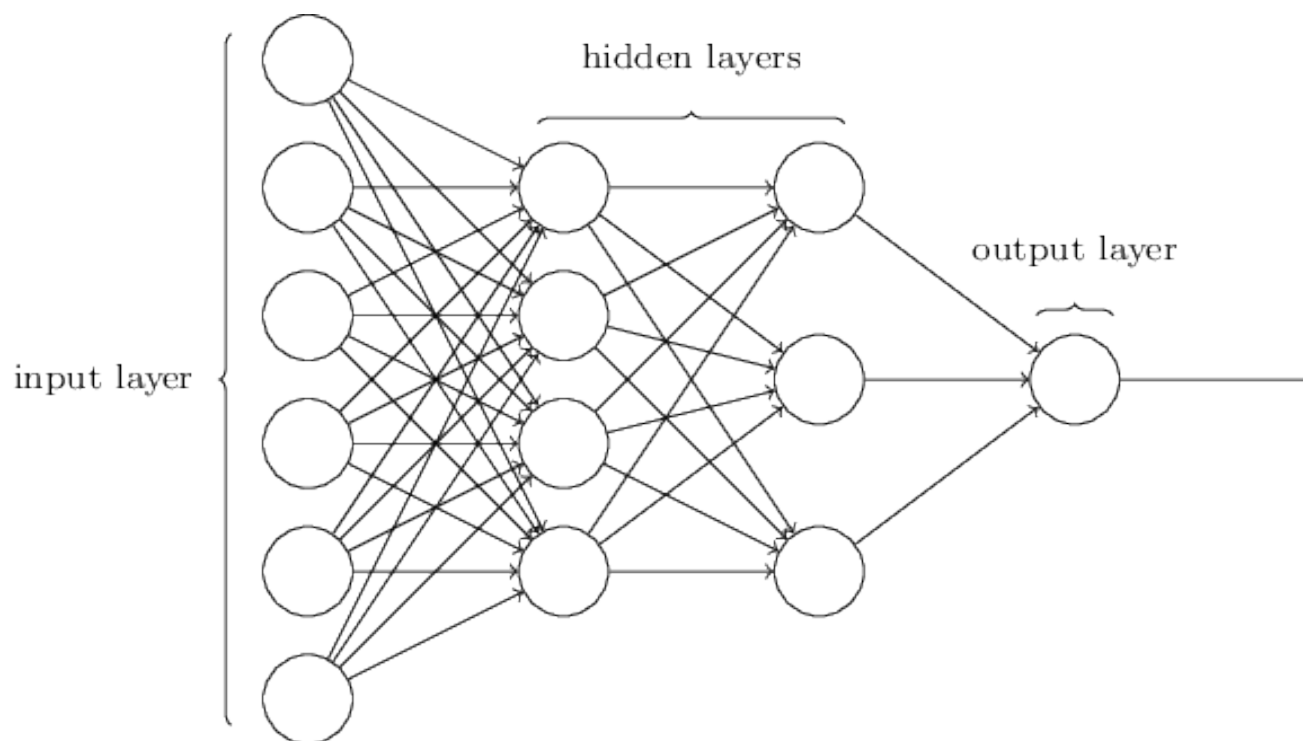


# Architecture of Neural Networks



- Sometimes called multi-layer perceptron (although sigmoid neurons are used)
- Output from one layer is used as input for the next (feedforward network)

# Architecture of Neural Networks



- Input and output layers are easily designed
- Hidden layers are harder
  - Many heuristics exist (which we'll cover later)

# Handwritten digit recognition



Two problems

1. Segmentation
2. Digit recognition

504 / 92

# Handwritten digit recognition



Two problems

1. Segmentation
2. Digit recognition



- We'll focus on digit recognition

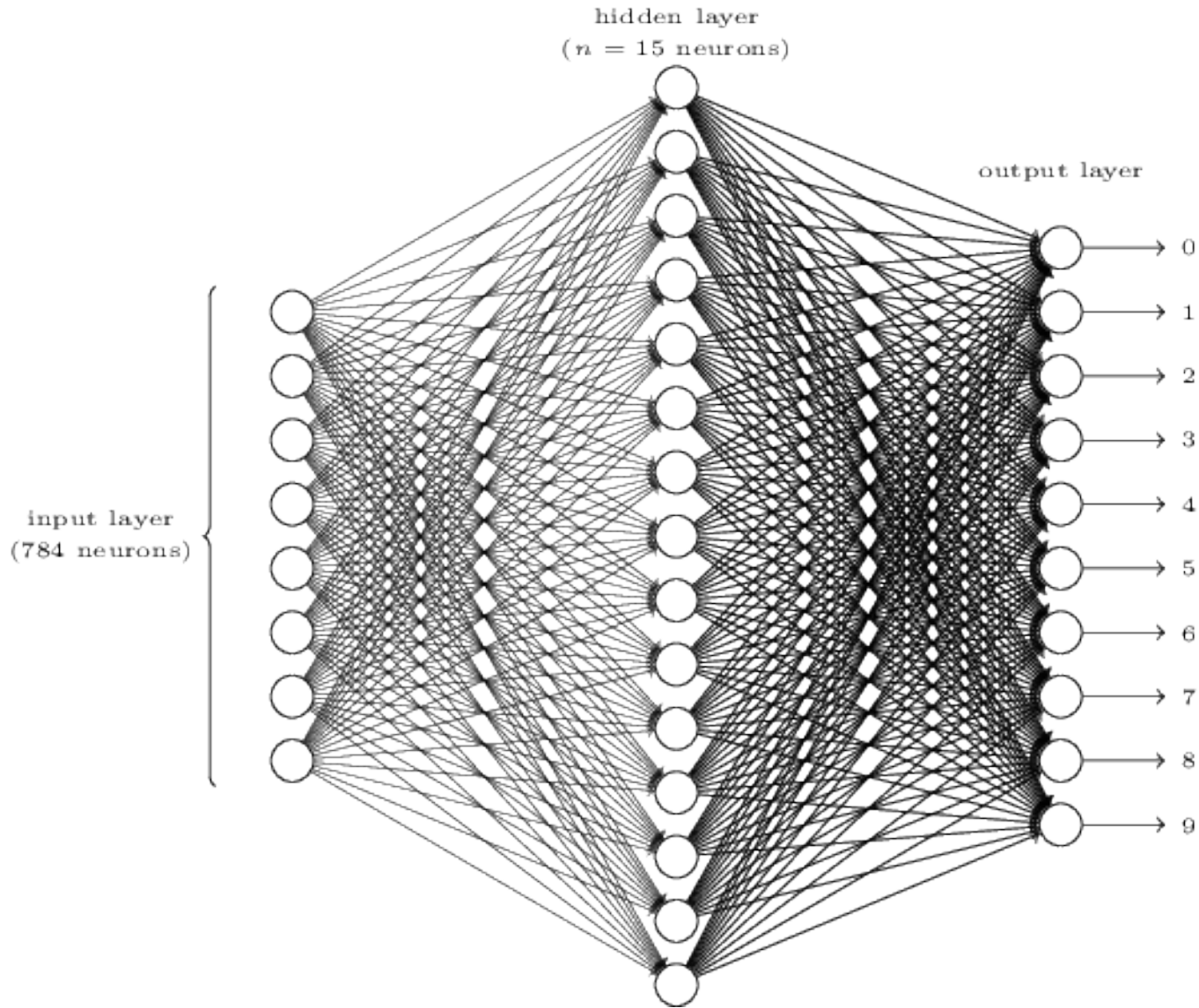
# Image segmentation approach



- Run digit classifier on multiple segmentations
  - High score if classifier is confident in all segments
  - Low score if the classifier fails in one or more segments
- I.e., if the classifier is struggling, it's due to a poor segmentation



# A simple network



# Number of output neurons

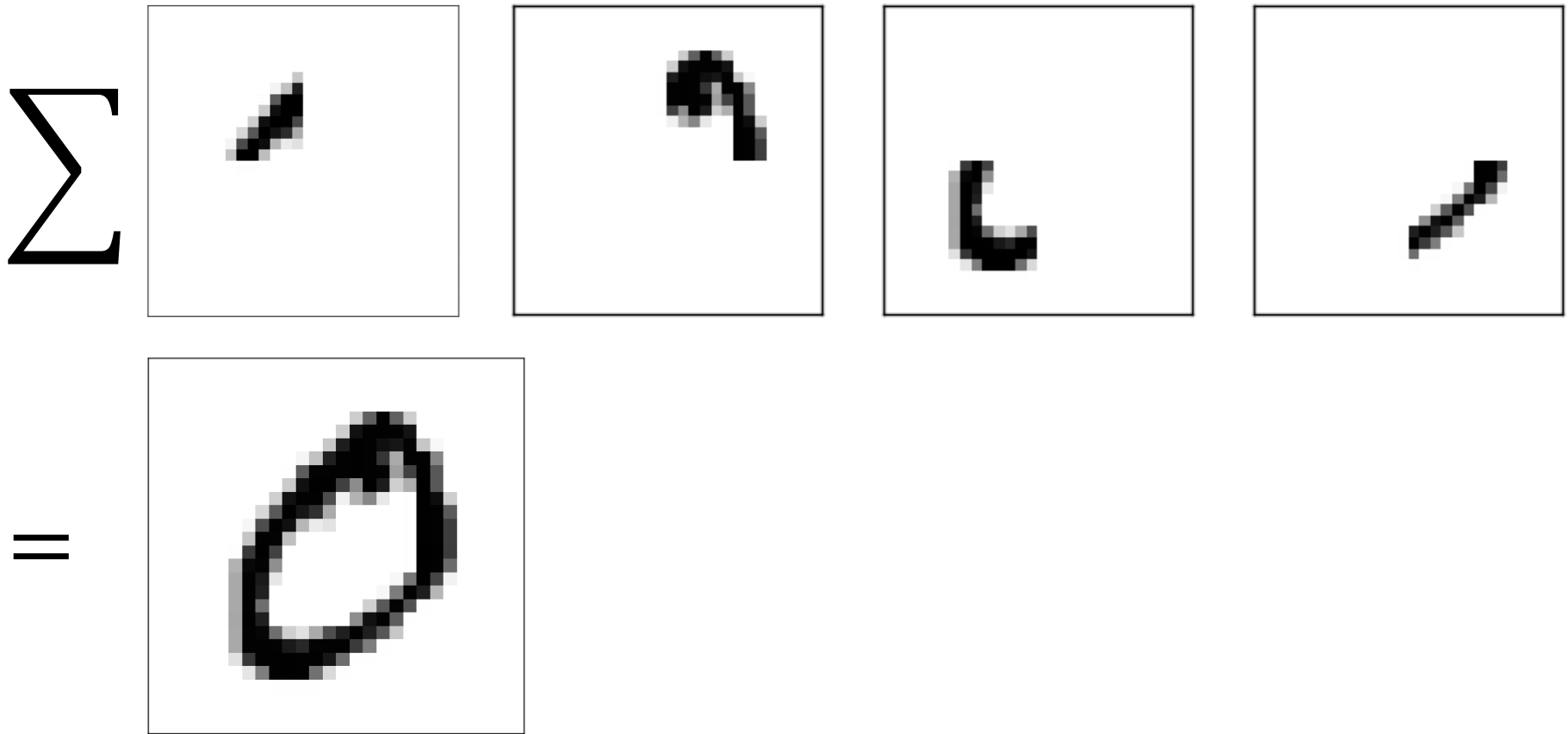


- Why use 10?
- Four bits are enough to encode the answer
  - Using more seems inefficient
- Empirically, 10 works better
  - Why?

# Number of output neurons



- Possible heuristic:





# Further Reading



- Nielsen, chapter 1
- Goodfellow et al., chapter 6