RADONC AI CURRICULUM

Introduction to Deep Learning

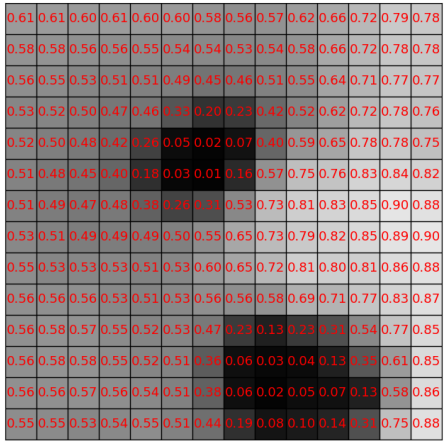
LECTURE 4: Image Data

ANDREW Y. K. FOONG, PH.D.

April 24th 2026

MAYO CLINIC

Radiation Oncology
AI & Data Analytics
AIDA



Today's lecture

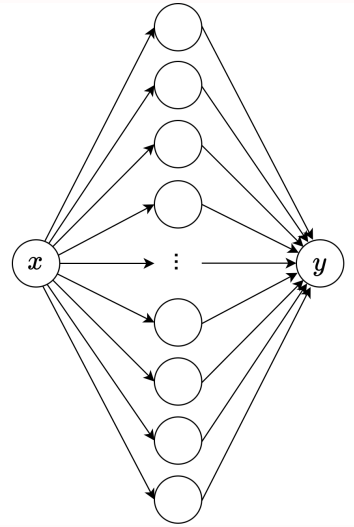
1. Images as data
   - *Representing an image as numbers*
2. Tensors
   - *Higher-dimensional grids of numbers*
3. Images as inputs
   - *Inputting many numbers into a network*
4. Converting outputs to labels
   - *From regression to classification*
5. The softmax function
   - *Defining valid probability distributions*
6. Two-way classification
   - *A simple example*
7. Q&A

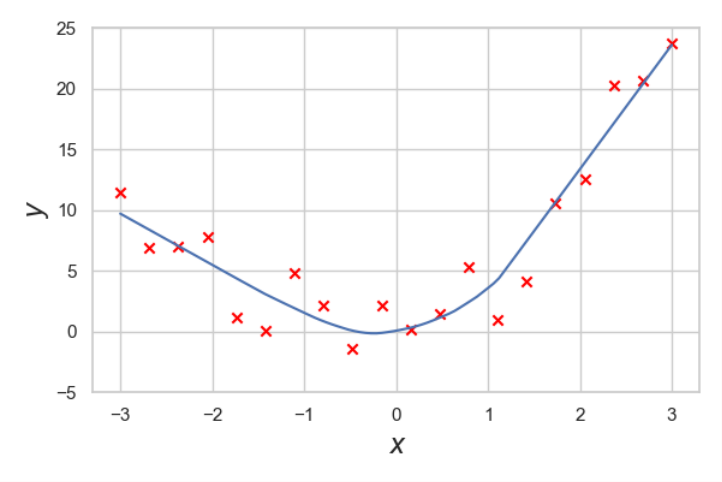
Images as data

Representing an image as numbers

Story so far

- So far, take *one* number and predict *one* number: $y = f(x)$
- How to use neural networks to take *image* as input?



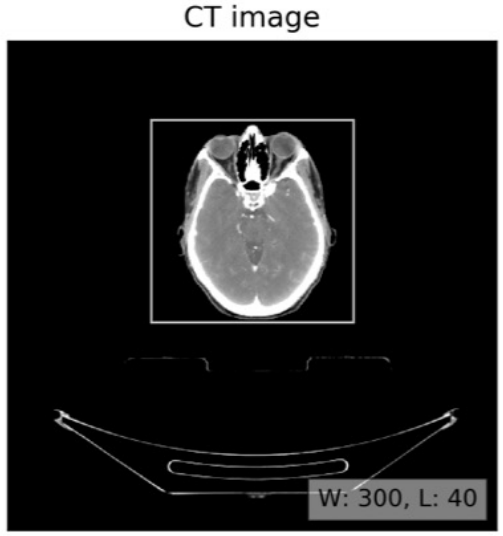


One-dimensional input, one-dimensional output regression

Image prediction tasks

- How to use neural networks for image auto-segmentation?

EXAMPLE:

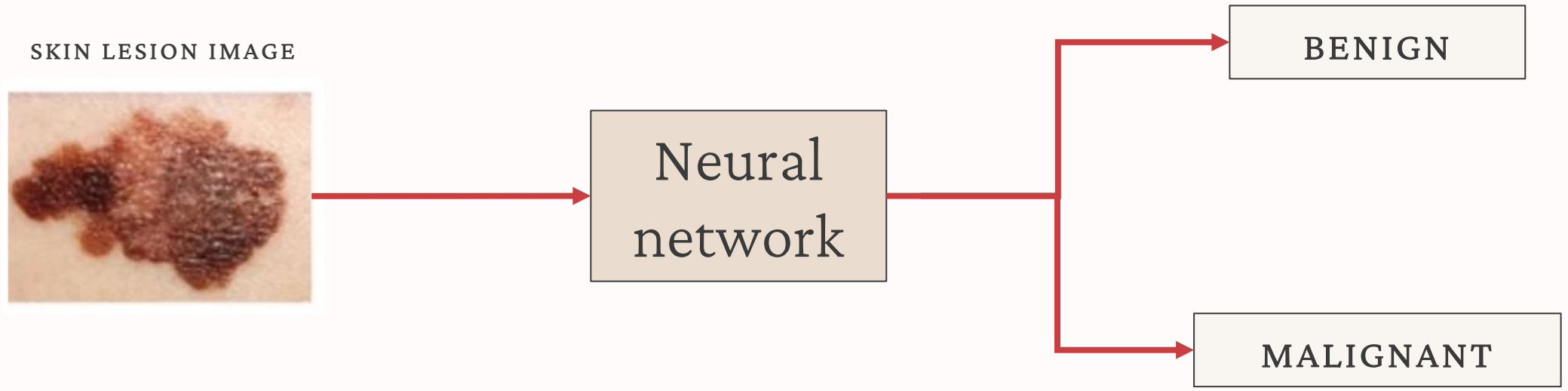


Nikolov et al. 2021, Deep learning to achieve clinically applicable segmentation of head and neck anatomy for radiotherapy, arXiv preprint.

Image prediction tasks

- How to use neural networks for image classification?
- *Tens of thousands* of medical applications.

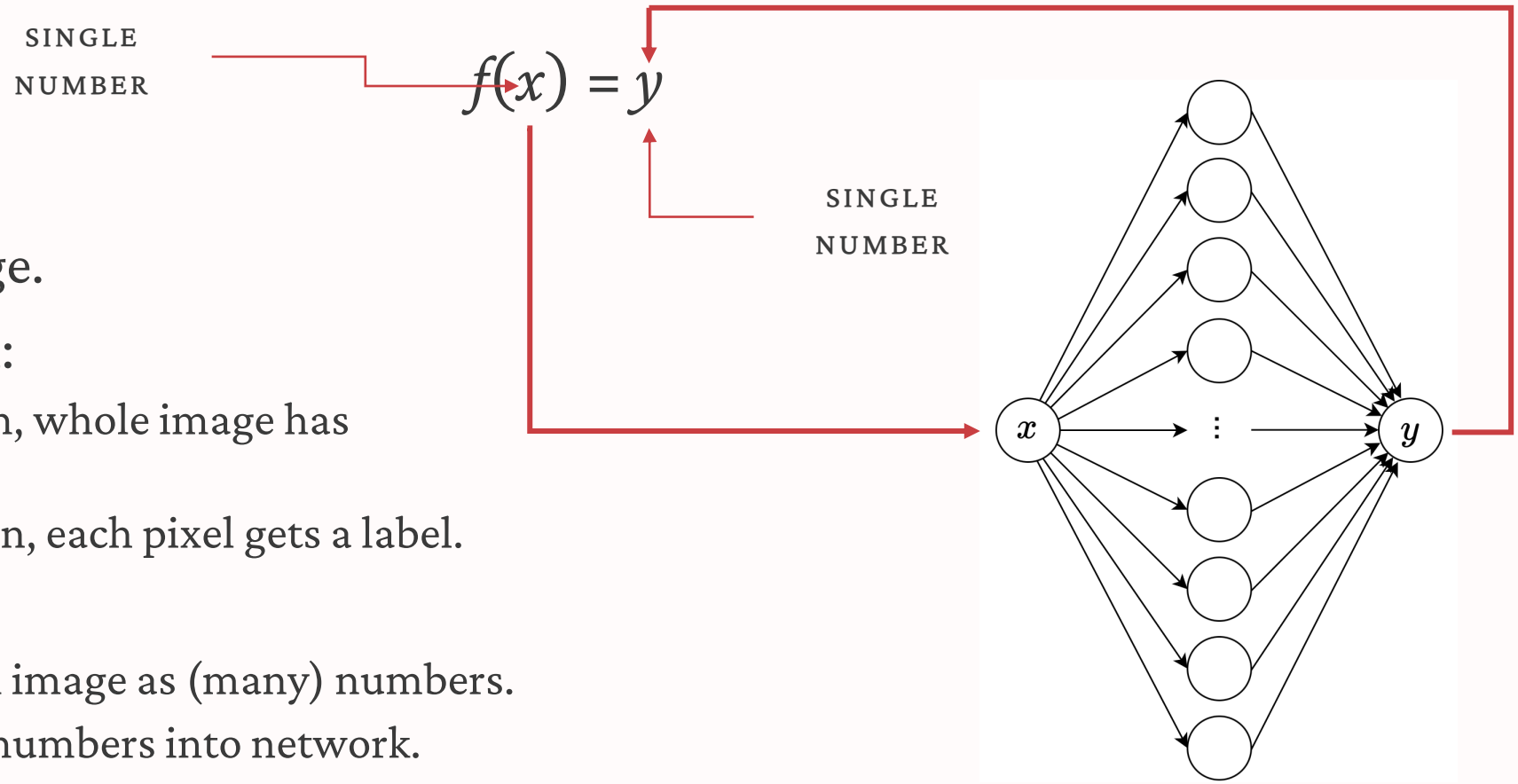
EXAMPLE:



Esteva, A., Kuprel, B., Novoa, R. et al. Dermatologist-level classification of skin cancer with deep neural networks. Nature 542, 115–118 (2017). https://doi.org/10.1038/nature21056

Deep learning for images: 40,000 foot view

- Use *same ideas* as when predicting a single number from another!



- Need x to be image.
- Need y to be label:
  - For classification, whole image has one label.
  - For segmentation, each pixel gets a label.
- PLAN:
  1. Represent an image as (many) numbers.
  2. Input many numbers into network.
  3. Convert y into labels.

Images to numbers

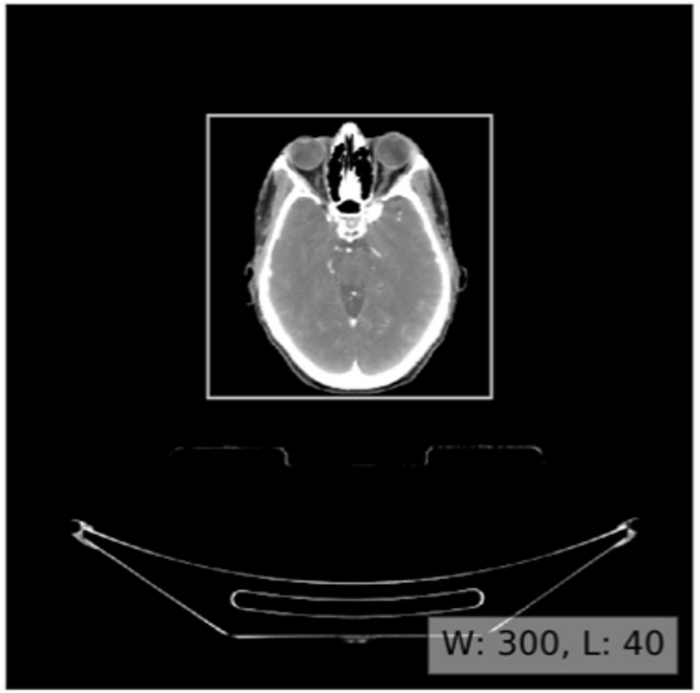
- Computers store images as a GRID OF NUMBERS.
- Illustrate with *everyday images* for simplicity.

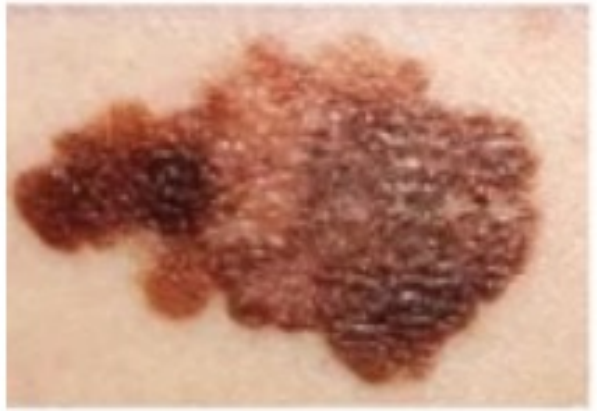
> "Everything should be as simple as it can be, but not simpler"
>
> — ALBERT EINSTEIN

- *Same principles apply for medical images!*
- Power of deep learning:
  - *Single* approach that works for *many* tasks.
  - No need to hand-craft one method for "everyday" images and another for medical images.

Images to numbers

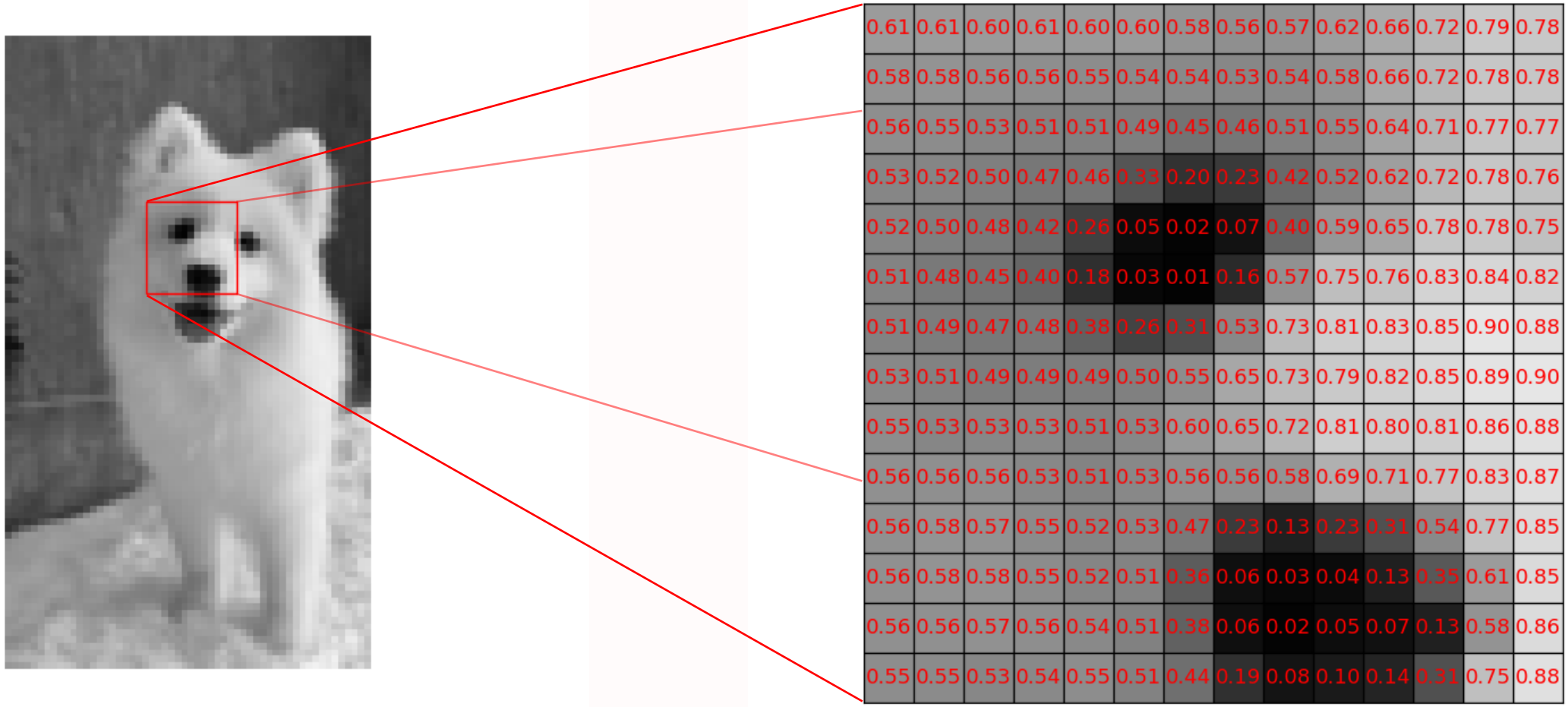
- Computers store images as a GRID OF NUMBERS.





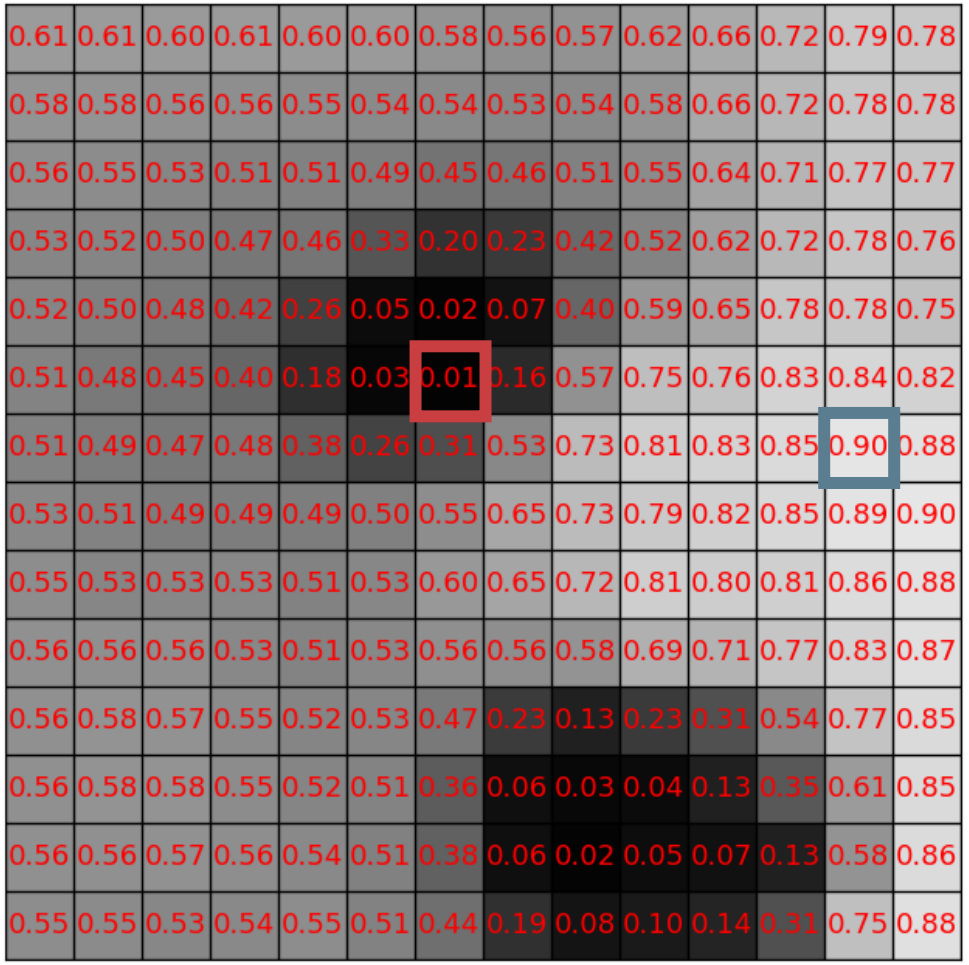
Images to numbers

- Computers store images as a **GRID OF NUMBERS.**



Images to numbers

- Computers store images as GRID OF NUMBERS.
- Each pixel is a number.
  - Larger → brighter.
  - Smaller → dimmer.



Tensors

Higher-dimensional grids of numbers

Some mathematics: tensors

- Grids of numbers are known in deep learning as **TENSORS.**
- A tensor is a *generalization* of the vector:
  - Can think of a vector as just a list of numbers.
  - Usually written in **boldface**.

EXAMPLE:

$$\mathbf{x} = [x_1, x_2, x_3]$$

Some mathematics: tensors

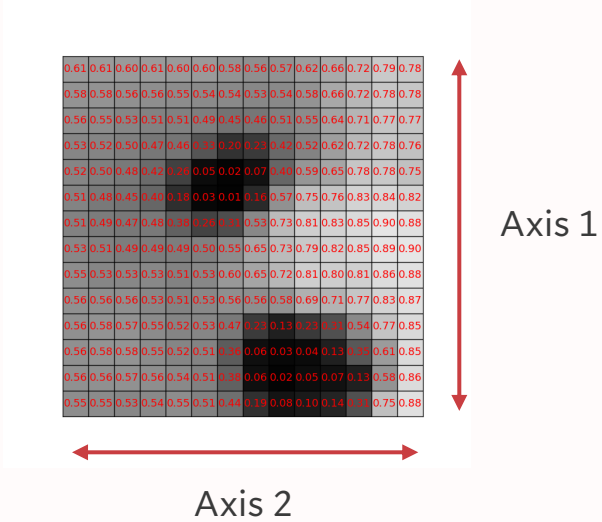
- Unlike vectors, in general, a *tensor* can have more than one *axis*.
- A *two-axis tensor* is also known as a *matrix*.

EXAMPLE:

$$\mathbf{x} = \begin{bmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \\ x_{31} & x_{32} \end{bmatrix}$$

Axis 1

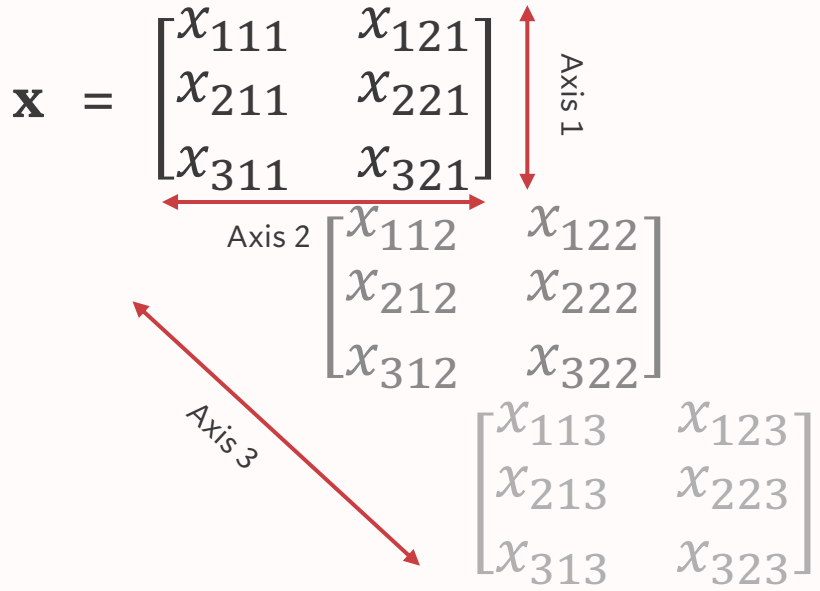
Axis 2



Some mathematics: tensors

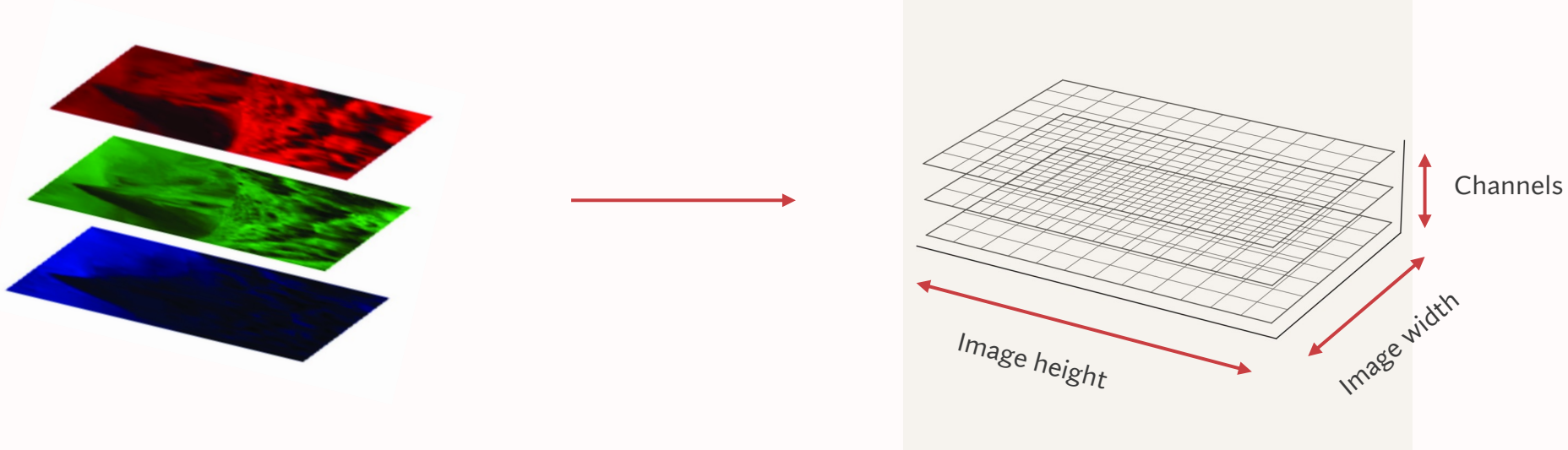
- Unlike matrices, in general, a *tensor* can also have more than *two axes*.
- A *tensor with N axes* is also known as a:
  - *N*th-order tensor
  - Rank *N* tensor
  - *N*-dimensional tensor

EXAMPLE: third-order tensor (difficult to draw)



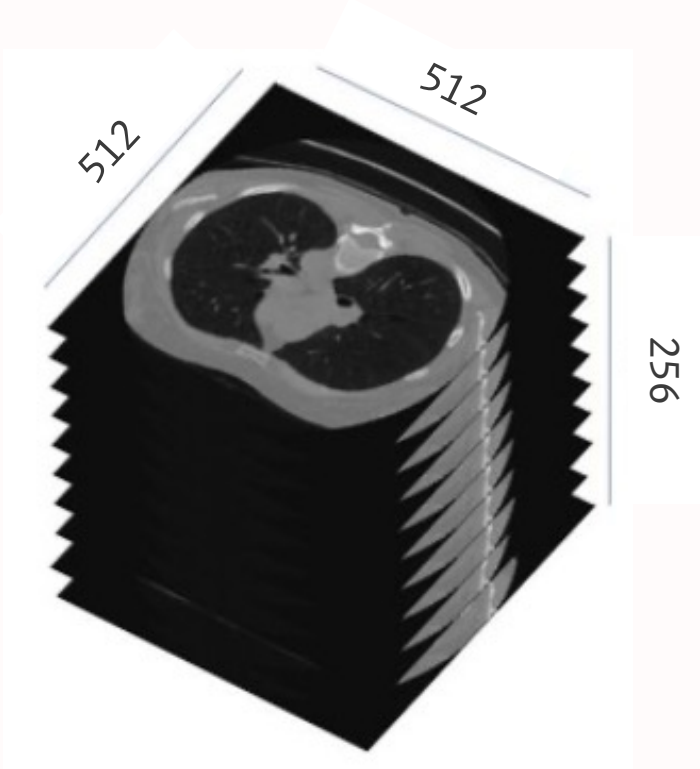
RGB images

- A *color image* is a common example of a third-order tensor.
  - Colors can be broken down into three *channels*: RED, GREEN, BLUE.
- The list of sizes of each axis is known as the *shape*.
  - EXAMPLE: an image tensor may have shape [3, 512, 512].



CT images

- A *three-dimensional CT image* is a common example of a third-order tensor.
  - There is no color, but the scan provides values in all three dimensions.
  - This tensor has shape [512, 512, 256].
- Higher-order tensors are possible too:
  - A *batch* of 16 CT images would be a *fourth-order tensor* with shape [16, 512, 512, 256].

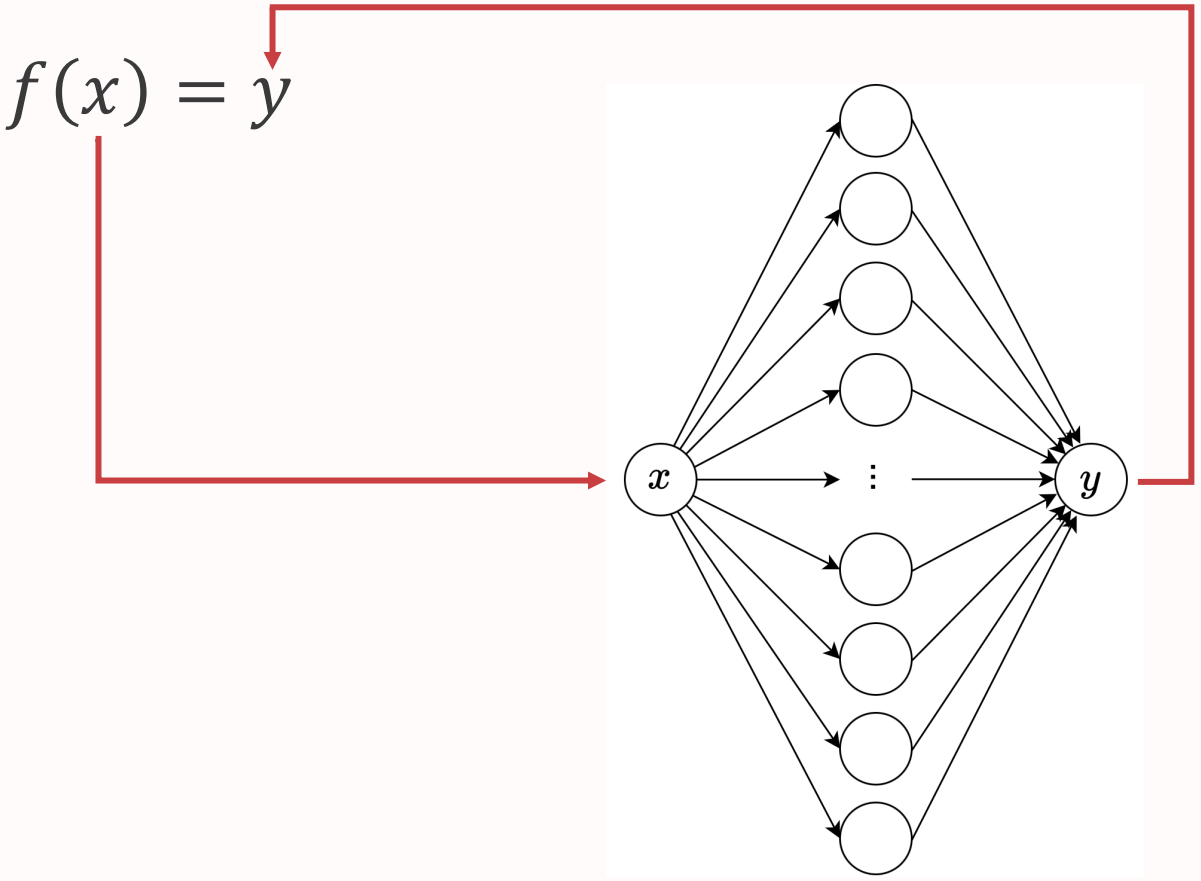


Images as inputs

Inputting many numbers into a network

Handling multiple inputs

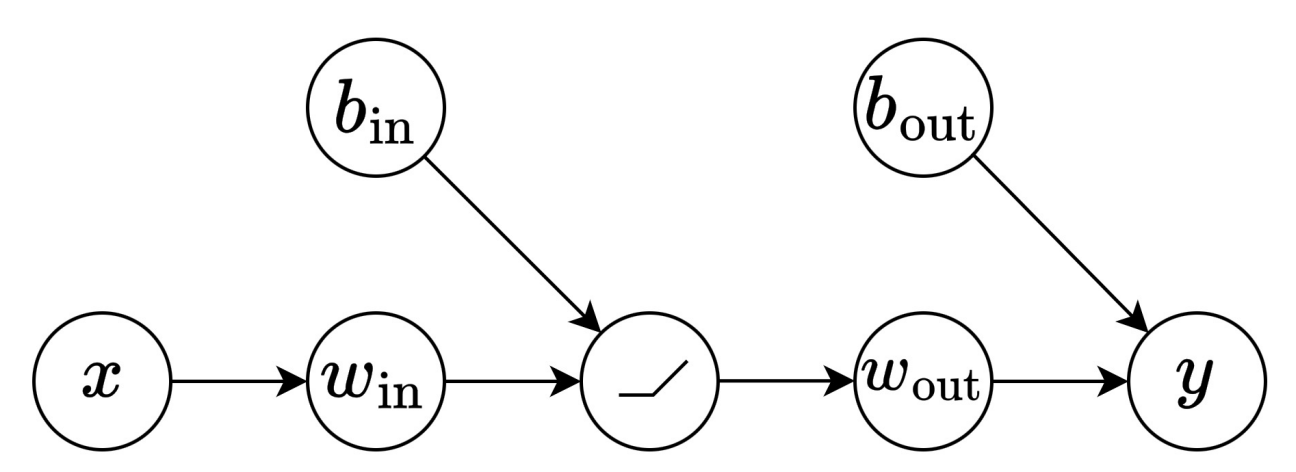
- PLAN:
  1. Represent an image as (many) numbers. ✓
  2. Input many numbers into network.
  3. Convert y into labels.
- Design with architecture diagram.



Recap: architecture diagrams

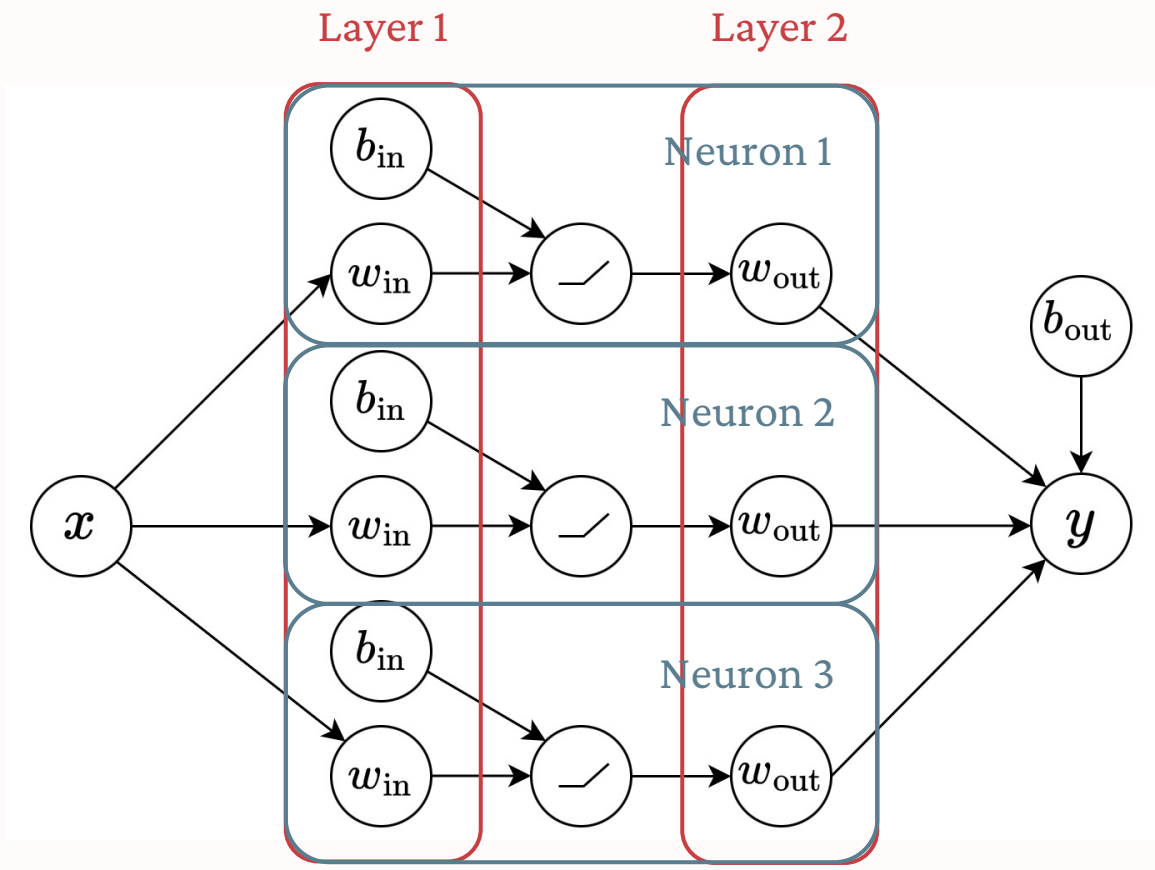
- Represent *equations* with *diagrams*:

$$y = w_{\text{out}}\text{ReLU}(w_{\text{in}}x + b_{\text{in}}) + b_{\text{out}}$$



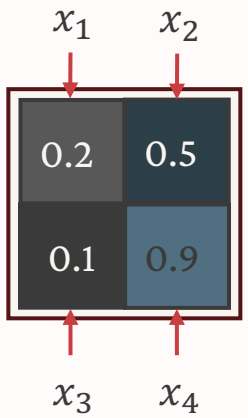
Recap: architecture diagrams

- Represent *equations* with *diagrams*:



Handling multiple inputs

- **PLAN:**
  1. Represent image as (many) numbers. ✓
  2. Input many numbers into network.
  3. Convert y into labels.
- Imagine image has 4 pixels:



Handling multiple inputs

- **PLAN:**
  1. Represent image as (many) numbers. ✓
  2. Input many numbers into network
  3. Convert y into labels.
- Imagine image has 4 pixels:

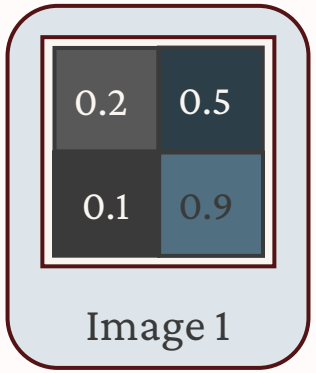
$$x = (x_1, x_2, x_3, x_4)$$

Squint and imagine this is: CT/skin lesion/dog image

Handling multiple inputs

- PLAN:
  1. Represent image as (many) numbers. ✓
  2. Input many numbers into network.
  3. Convert y into labels.
- Imagine image has 4 pixels:

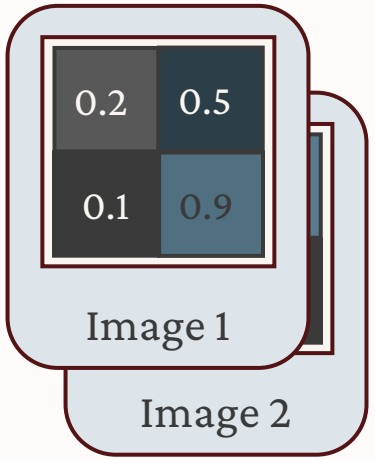
$$x = (x_1, x_2, x_3, x_4)$$



Handling multiple inputs

- **PLAN:**
  1. Represent image as (many) numbers. ✓
  2. Input many numbers into network.
  3. Convert y into labels.
- Imagine image has 4 pixels:

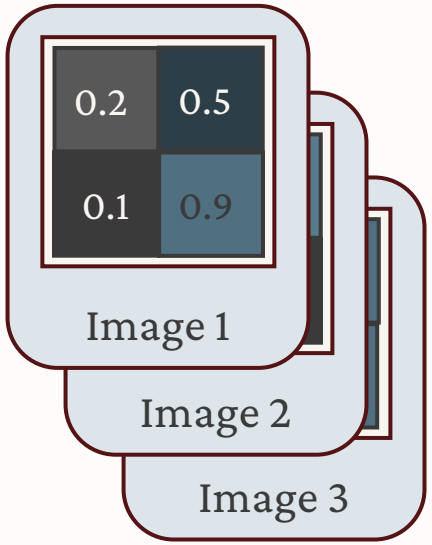
$$x = (x_1, x_2, x_3, x_4)$$



Handling multiple inputs

- **PLAN:**
  1. Represent image as (many) numbers. ✓
  2. Input many numbers into network.
  3. Convert y into labels.
- Imagine image has 4 pixels:

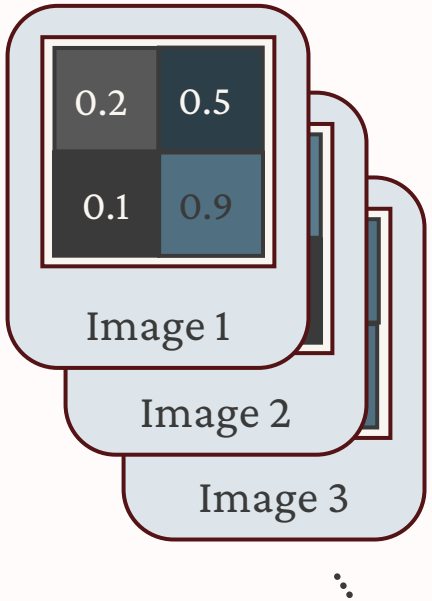
$$x = (x_1, x_2, x_3, x_4)$$



Handling multiple inputs

- **PLAN:**
  1. Represent image as (many) numbers. ✓
  2. Input many numbers into network.
  3. Convert y into labels.
- Imagine image has 4 pixels:

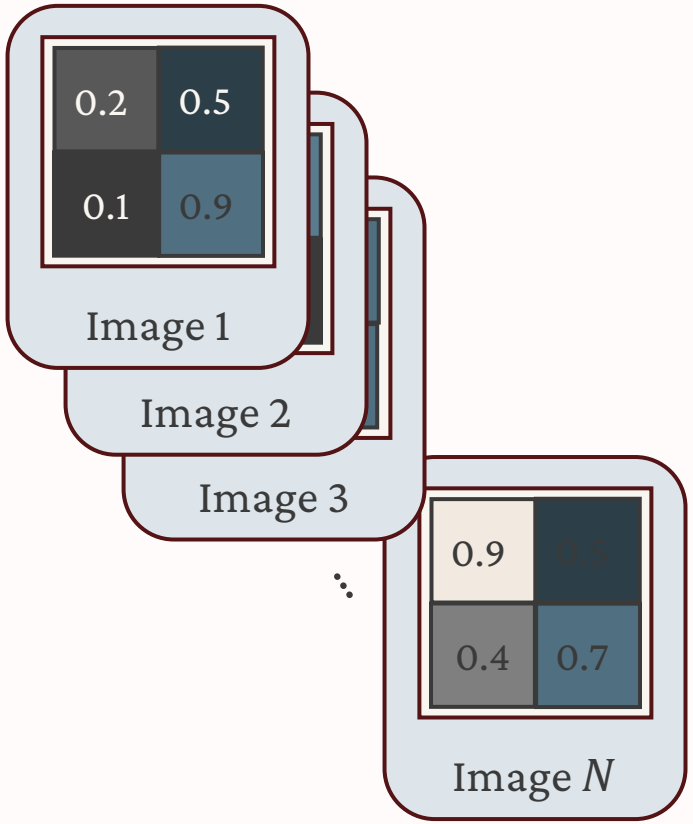
$$x = (x_1, x_2, x_3, x_4)$$



Handling multiple inputs

- **PLAN:**
  1. Represent image as (many) numbers. ✓
  2. Input many numbers into network.
  3. Convert y into labels.
- Imagine image has 4 pixels:

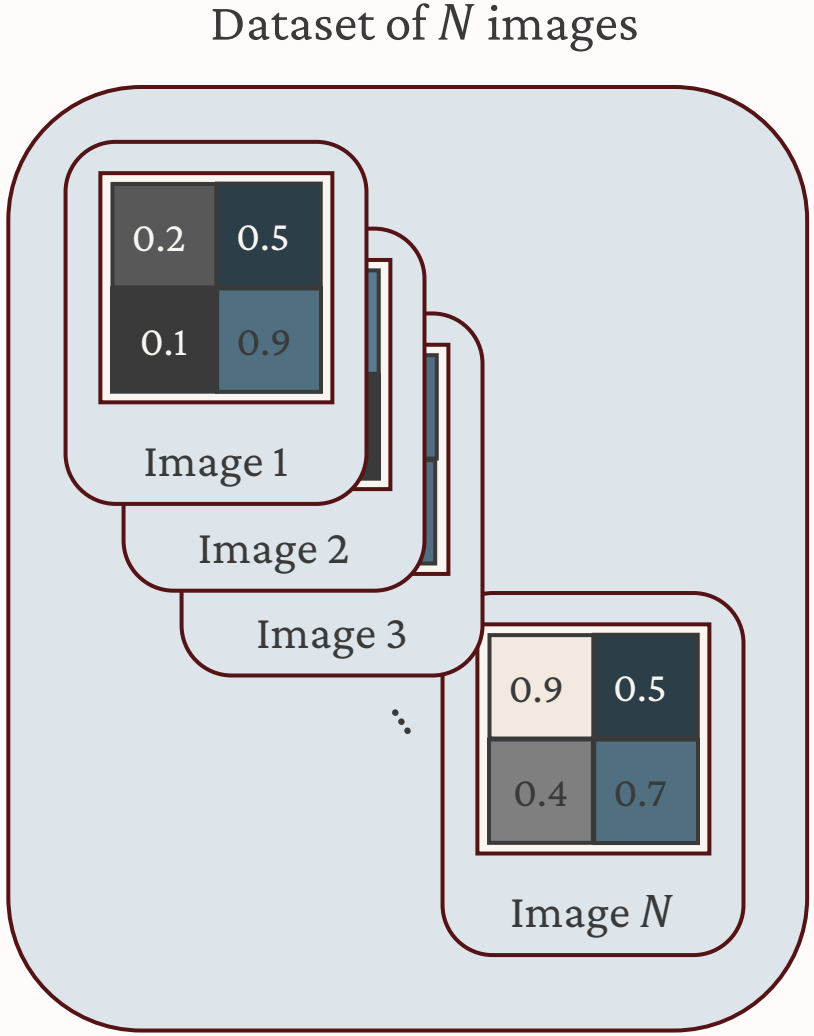
$$x = (x_1, x_2, x_3, x_4)$$



Handling multiple inputs

- **PLAN:**
  1. Represent image as (many) numbers. ✓
  2. Input many numbers into network.
  3. Convert y into labels.
- Imagine image has 4 pixels:

$$x = (x_1, x_2, x_3, x_4)$$

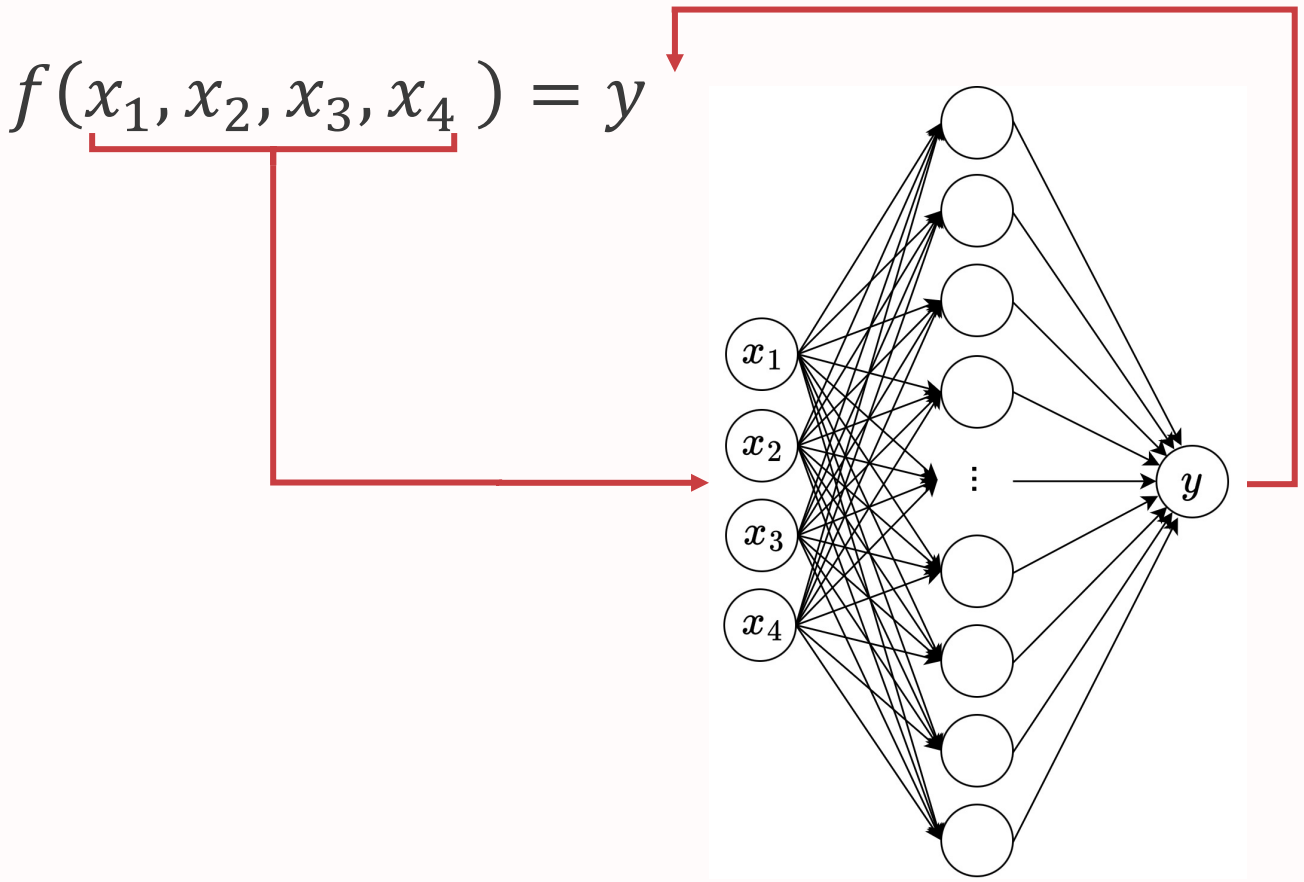


Handling multiple inputs

- PLAN:
  1. Represent image as (many) numbers. ✓
  2. Input many numbers into network.
  3. Convert y into labels.
- Imagine image has 4 pixels:

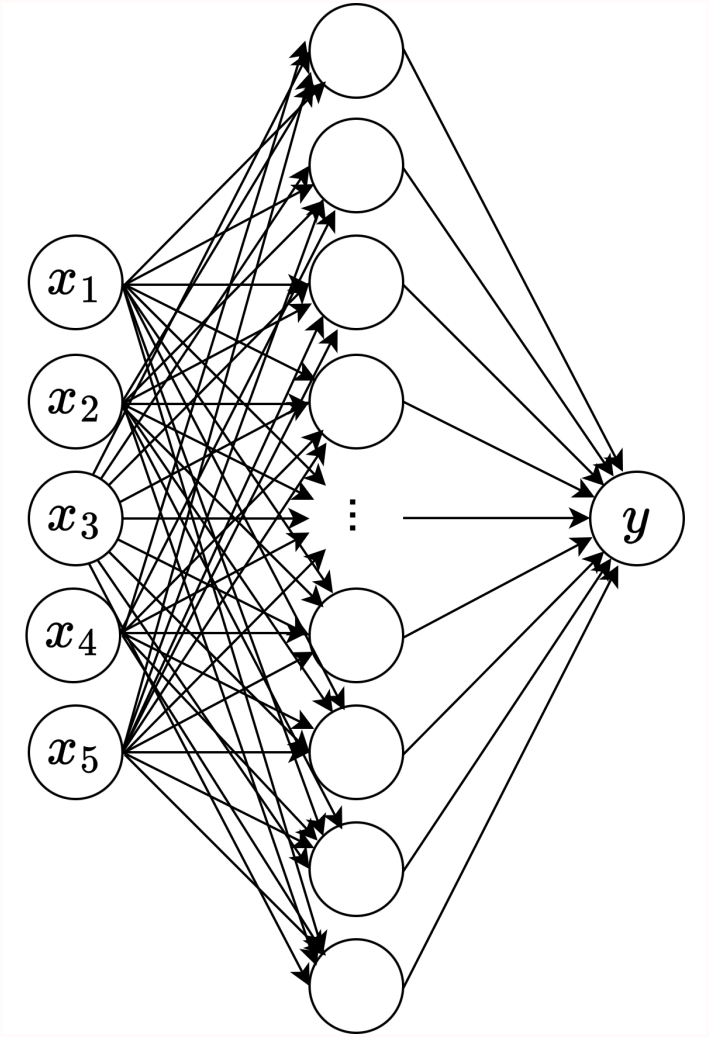
$$x = (x_1, x_2, x_3, x_4)$$

- Connect each pixel to each neuron.



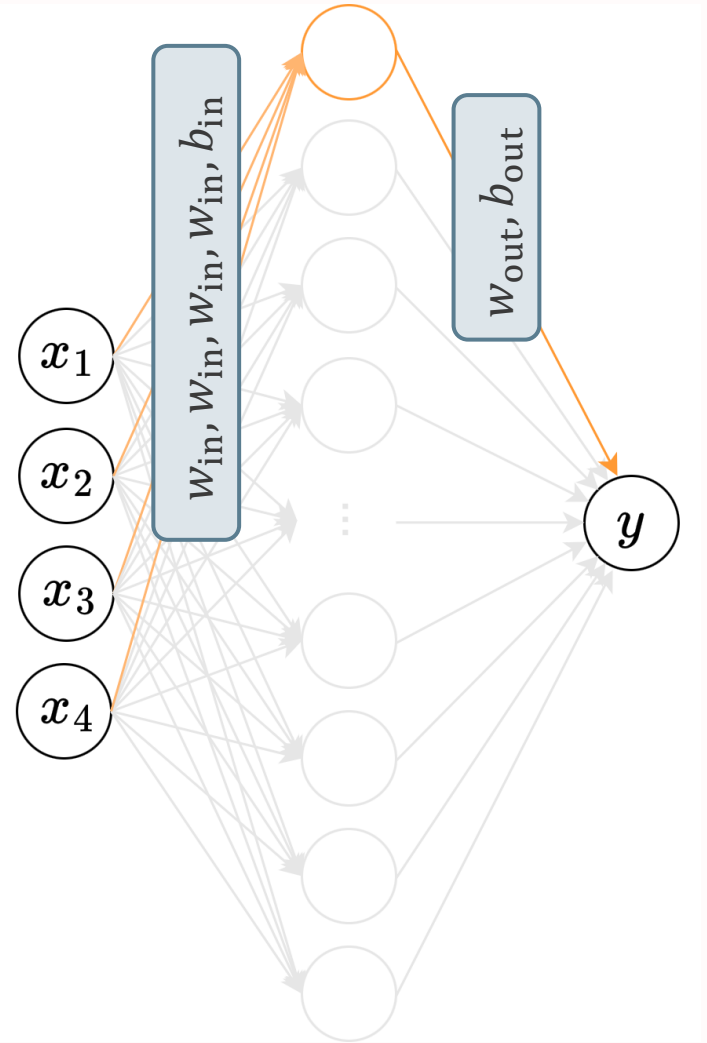
Interpreting the diagram

- Focus on one neuron.



Interpreting the diagram

- Focus on one neuron.
- Recall: arrows have parameters w, b.



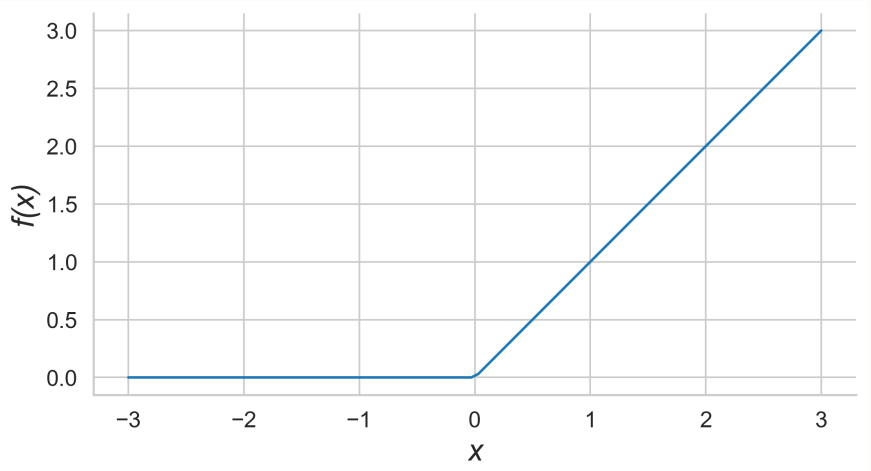
Interpreting the diagram

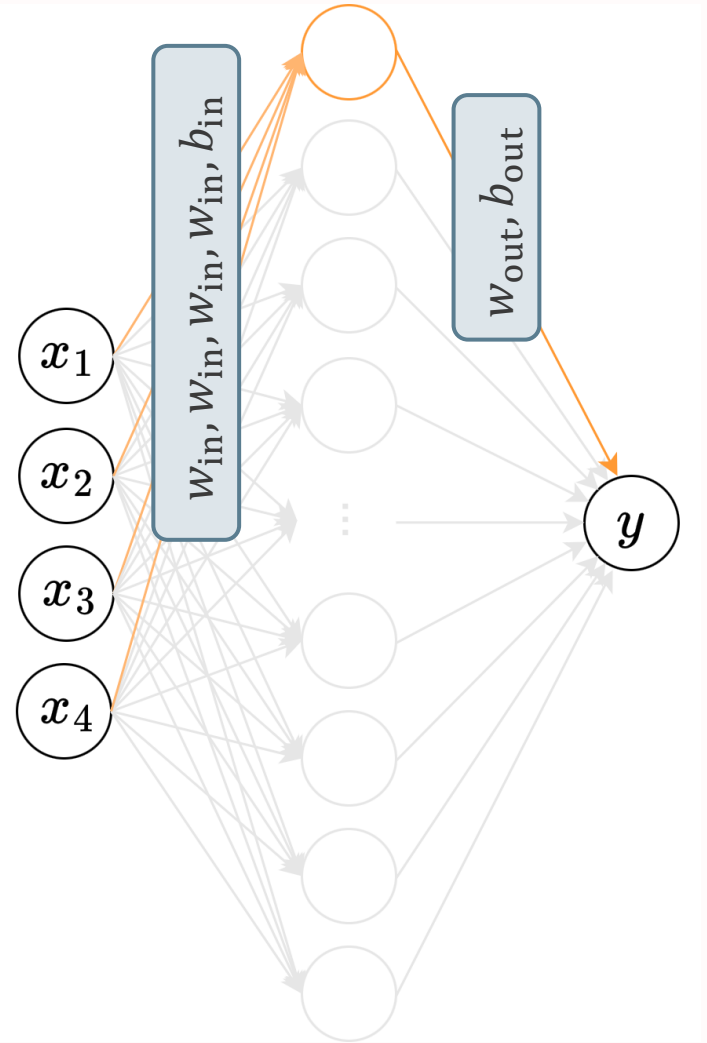
- Focus on one neuron.
- Recall: arrows have parameters w, b.
- Orange neuron:

$$y = w_{\text{out}}\text{ReLU}\left(w_{\text{in},1}x_1 + w_{\text{in},2}x_2 + w_{\text{in},3}x_3 + w_{\text{in},4}x_4 + b_{\text{in}}\right) + b_{\text{out}}$$

- Neuron "fires" if

$$w_{\text{in},1}x_1 + w_{\text{in},2}x_2 + w_{\text{in},3}x_3 + w_{\text{in},4}x_4 + b_{\text{in}} > 0$$





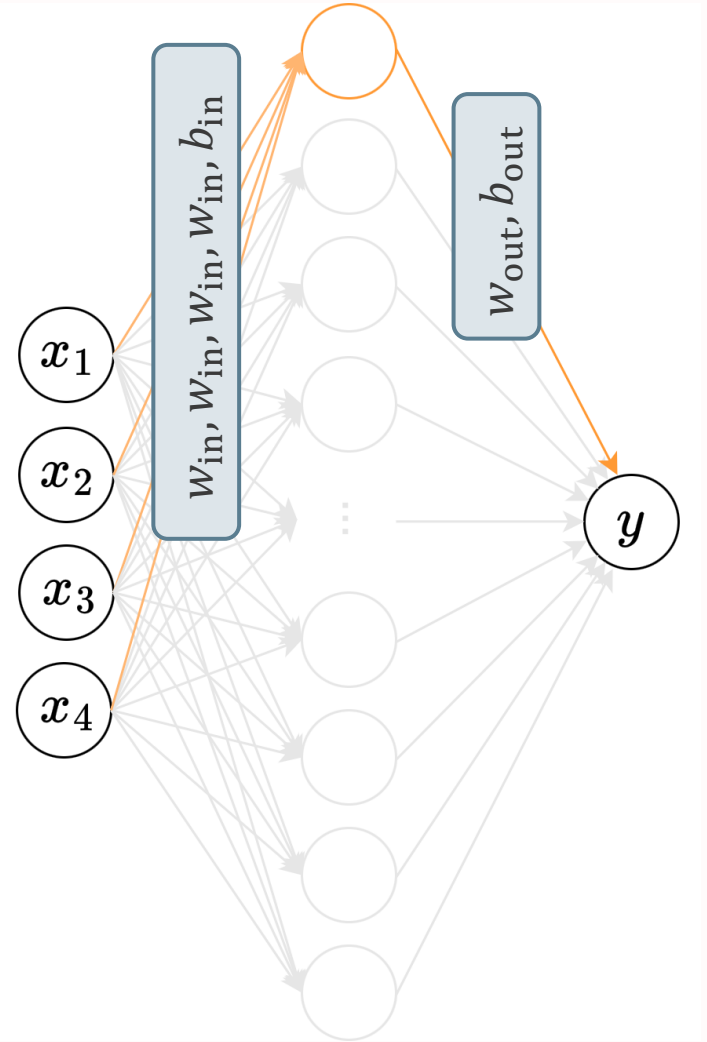
Neurons as feature detectors

INTERPRETATION OF ONE NEURON:

- Detects if

  $w_{\text{in},1}x_1 + w_{\text{in},2}x_2 + w_{\text{in},3}x_3 + w_{\text{in},4}x_4 + b_{\text{in}} > 0$

  - If so, "*activated*"
  - If not, "*deactivated*"
- $w_{\text{in},1}x_1 + w_{\text{in},2}x_2 + w_{\text{in},3}x_3 + w_{\text{in},4}x_4$ interpreted as "feature" of image.



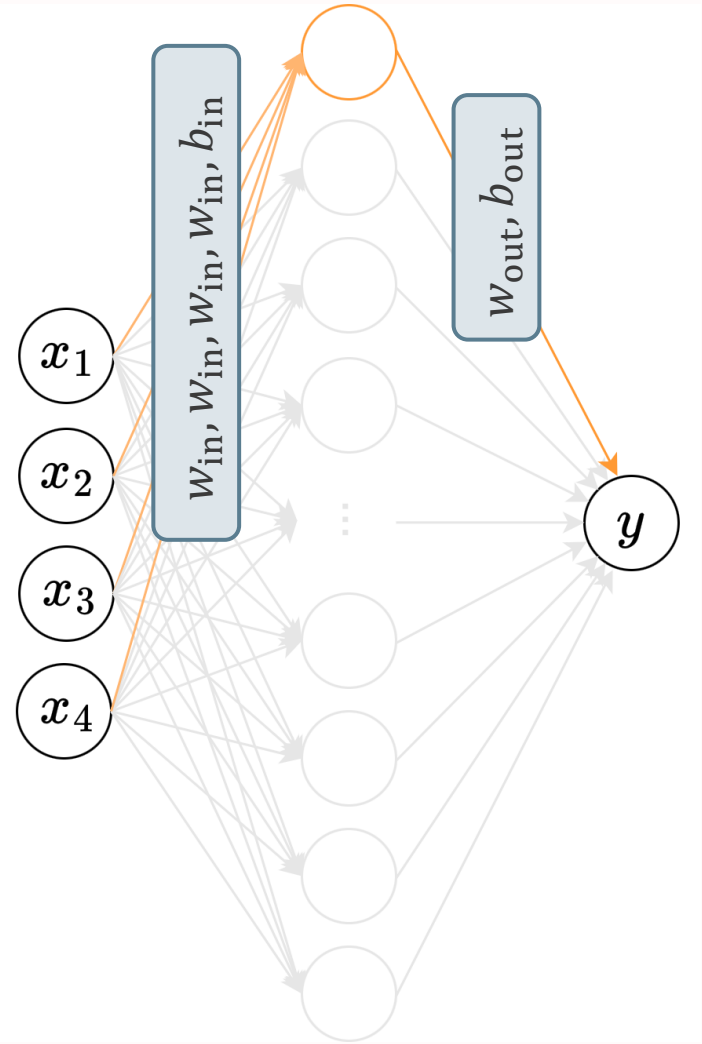
Neurons as feature detectors

- FEATURE: some property of the input used for prediction.
  - EXAMPLE: suppose we set all $w_{\text{in}} = 1/4$:

    $$w_{\text{in},1}x_1 + w_{\text{in},2}x_2 + w_{\text{in},3}x_3 + w_{\text{in},4}x_4 = (x_1 + x_2 + x_3 + x_4)/4.$$

  - This feature is *average brightness*.
  - Neuron is activated if brightness above some threshold.
  - This particular feature may or may not be useful.
- What features are useful for prediction?
  - Difficult to know in advance—depends on the task.
  - No need to to know in advance!

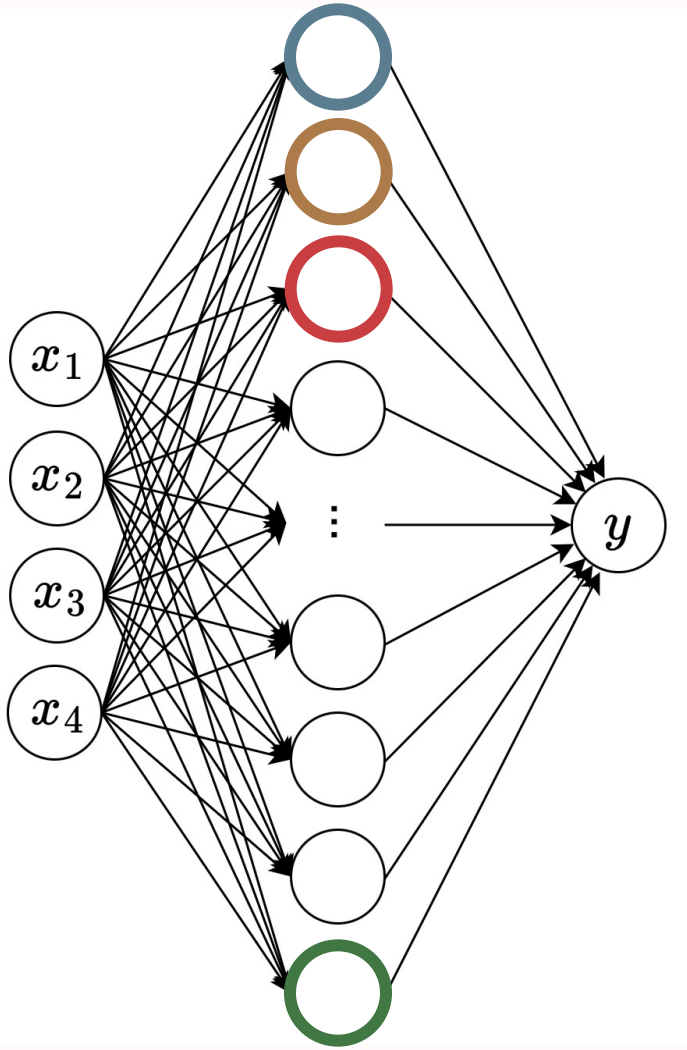
Features are learned automatically during gradient descent.



Combining features

- Final output *adds up and combines* all neurons:

$$y = w_{\text{out}}\text{ReLU}\left(w_{\text{in},1}x_1 + w_{\text{in},2}x_2 + w_{\text{in},3}x_3 + w_{\text{in},4}x_4 + b_{\text{in}}\right)$$



Converting outputs to labels

From regression to classification

Converting outputs to labels

- PLAN:
  1. Represent image as (many) numbers. ✓
  2. Input many numbers into network. ✓
  3. Convert y into labels.
- SO FAR: output $f(x) = y$ just a number—could be positive or negative.
- WANT: output as a *discrete label*:
  - *Benign* vs *malignant*
  - *Heart* vs *lung*
  - *Dog* vs *cat*

Converting outputs to labels

- IDEA: interpret output as *probability of an outcome.*
- RECAP: probability distributions, $\Pr(y)$:
  - It is a *function* that, given an outcome, assigns a probability value to that outcome.
  - If we know $\Pr(y)$ for all possible outcomes y, then we know how likely *any* event is going to be.
- In our case, the neural network outputs y would represent:
  - *Benign* vs *malignant*
  - *Heart* vs *lung*
  - *Dog* vs *cat*
- But what would the input x represent?

Some mathematics: conditional probability

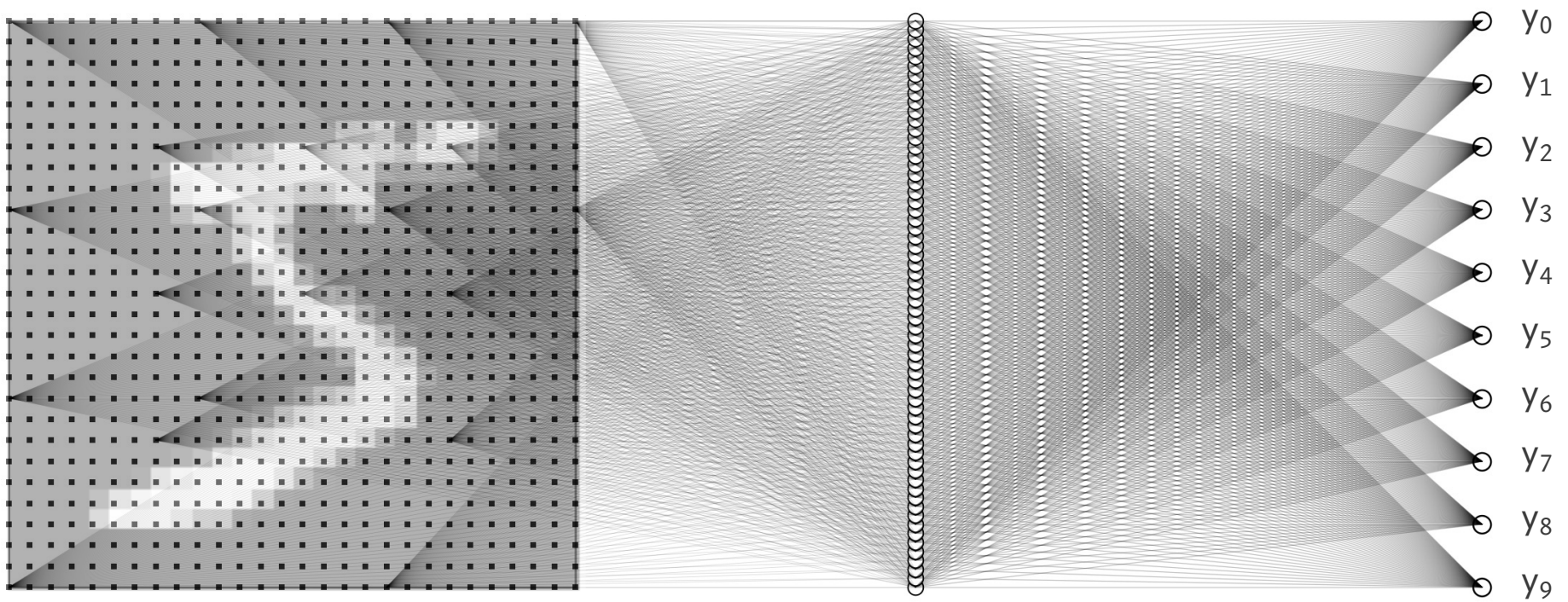
- So far, we have discussed *unconditional probability*:
  - Pr(HEADS), Pr(TAILS)
- But in prediction tasks, we care about *conditional probabilities*:
  - "What is the probability that the patient has malignant skin cancer, *given that their skin lesion looks like this?*"
  - "What is the probability that this pixel belong to the heart, *given that the CT image looks like this?*"
- NOTATION: *given-ness* is represented in conditional probability as a *bar*: "|"
  - Pr(MALIGNANT | IMAGE)—one per image.
  - Pr(HEART PIXEL | CT IMAGE)—one per *pixel* in the image.

Some mathematics: conditional probability

- Neural network: $f(x) = y$.
  - x is the image.
- Use network to express a conditional probability:
  - For each possible outcome, network needs one dedicated output.
  - *Example*: MNIST
    - Has ten possible outputs, one for each digit.
    - Network needs ten outputs: $y_1, y_2, \ldots, y_{10}$.
  - *Example*: Auto-segmentation of CT images.
    - Need one output for each possible organ.
    - E.g., if there are 80 organ types, need 80 outputs: $y_1, y_2, \ldots, y_{80}$.
- These probabilities will be completely wrong to begin with, but improve with training—just like the MSE of the best-fit curve.

Converting outputs to labels

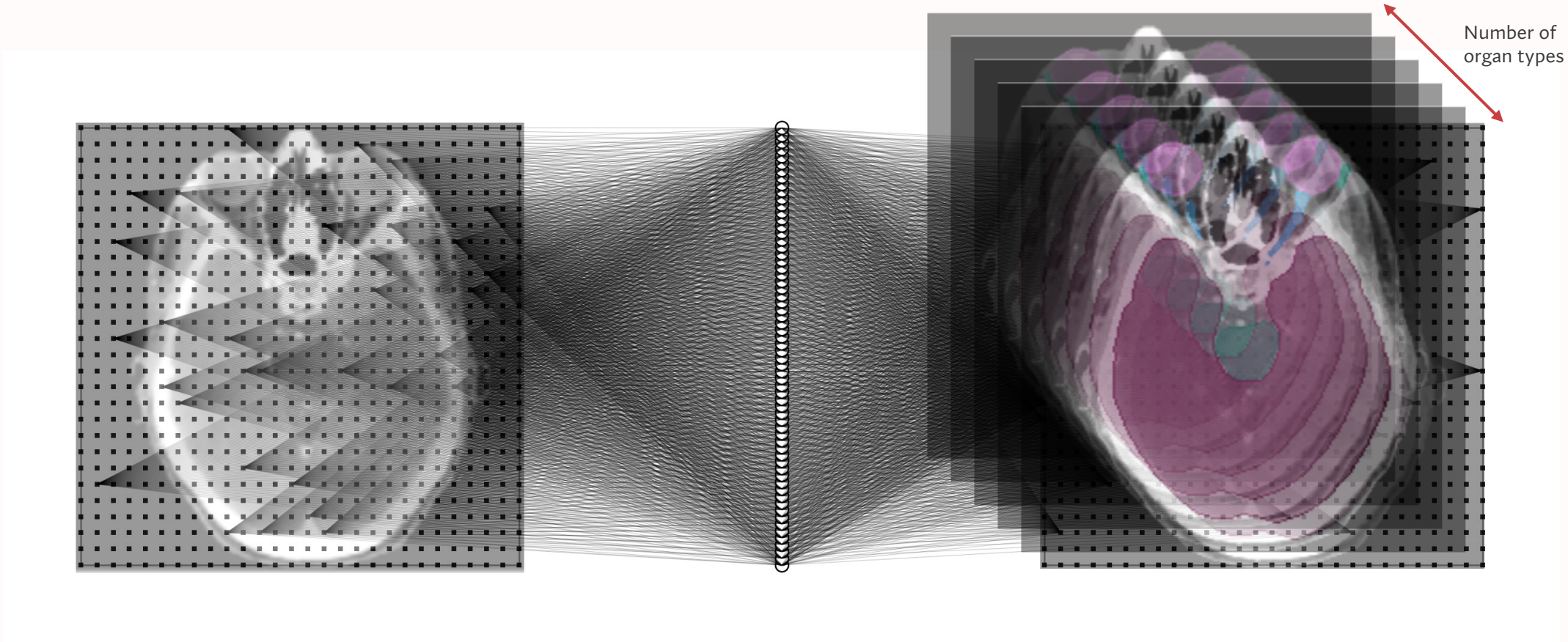
EXAMPLE: MNIST digit classifier



Note: connections to only a few pixels are shown to avoid clutter

Converting outputs to labels

EXAMPLE: 2D auto-segmentation (unrealistically simple architecture)



Input image, x: [height, width]

Note: connections to only a few pixels are shown to avoid clutter

Hidden layer, width 64

Outputs, y: [height, width, number of organ types]

Note: connections to only a few pixels are shown to avoid clutter

The softmax function

Defining valid probability distributions

Output range

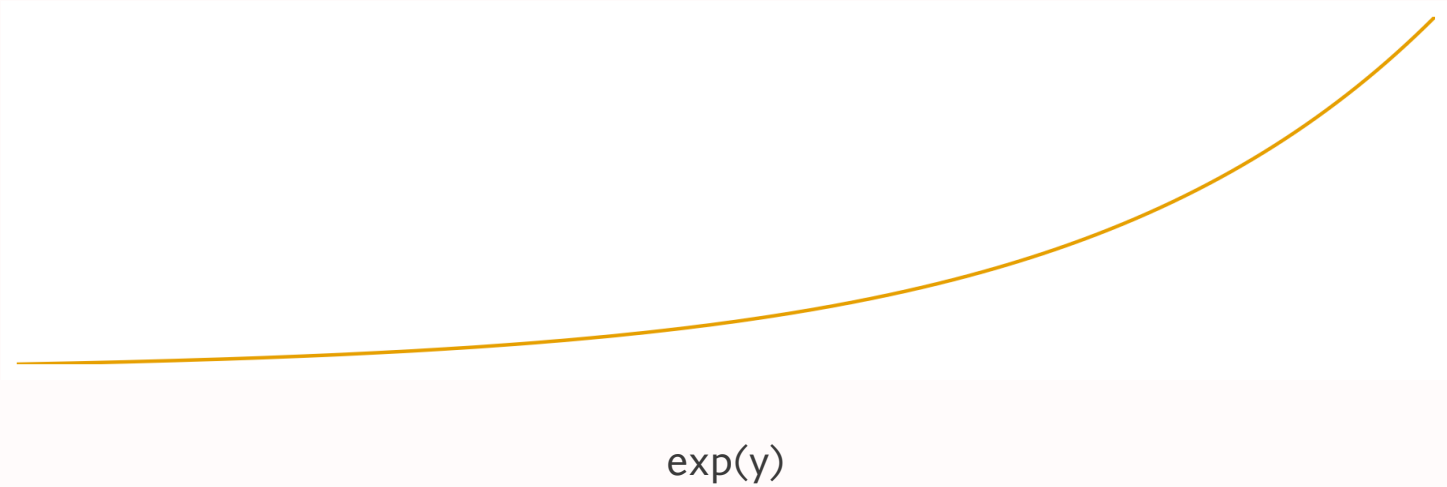
- PROBLEM:
  1. Probabilities are *always* between zero and one.
     - Neural network output can be greater than one or negative!
  2. Total probability of *all* events must add up to one.
     - Sum of $y_1, y_2, \ldots$ could potentially be anything!
- SOLUTION: Transform the output so it *always* satisfies both constraints, using the *Softmax function*.

Some mathematics: softmax

- Softmax is a function that:
  - Takes N inputs $y_1, y_2, \ldots, y_N$
  - Transforms it into N outputs $p_1, p_2, \ldots, p_N$
  - Such that:
    1. $0 < p_1 < 1, 0 < p_2 < 1, \ldots, 0 < p_N < 1.$
    2. $p_1 + p_2 + \ldots + p_N = 1.$
  - Regardless of what the inputs are, the outputs of softmax are *always* a valid probability distribution.

Some mathematics: softmax

- **DEFINITION:**
  - Let $\mathbf{y} = [y_1, y_2, \ldots, y_N]$.
  - Then to compute the function $\text{Softmax}(\mathbf{y}) = \mathbf{p}$, we do the following:
    1. Compute $\exp(y_1), \exp(y_2), \ldots, \exp(y_N)$.



Some mathematics: softmax

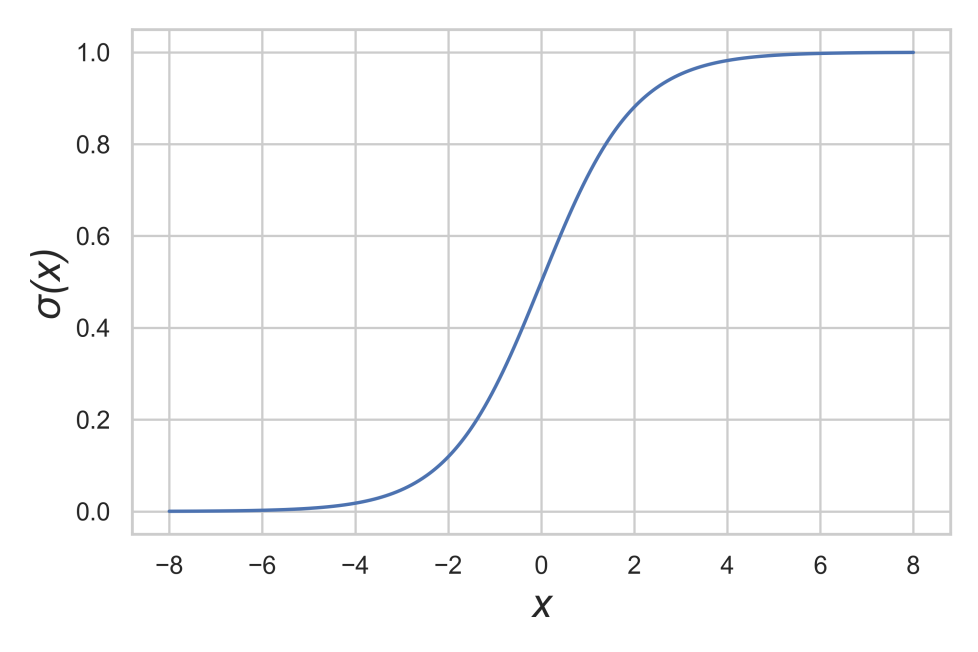
- DEFINITION:
  - Let $\mathbf{y} = [y_1, y_2, \ldots, y_N]$.
  - Then to compute the function $\text{Softmax}(\mathbf{y}) = \mathbf{p}$, we do the following:
    1. Compute $\exp(y_1), \exp(y_2), \ldots, \exp(y_N)$.
    2. Normalize: $p_1 = \exp(y_1) / \left(\exp(y_1) + \exp(y_2) + \ldots + \exp(y_N)\right)$
    3. And similarly for $p_1, p_2, \ldots, p_N$
  - Normalization *guarantees* that all the elements of $\mathbf{p}$ sum up to one.
- Why softmax?
  - Exponential is a smooth function—easy for gradient descent to "slide down" it.
  - It's simple and it works.

Two-way classification

A simple example

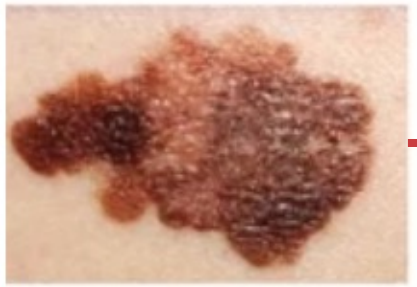
Simple case: two-way classification

- In two-way or binary classification, we don't need the softmax yet.
- Squash output using a *sigmoid function*—this is probability of the first class.
- Probability of the second class is just one minus probability of the first.



Simple case: two-way classification

Skin lesion image



Neural network

Sigmoid function

BENIGN

$p = 0.85$

MALIGNANT

$1 - p = 0.15$

$x = (x_1, x_2, x_3 \ldots)$

$y = f(x)$

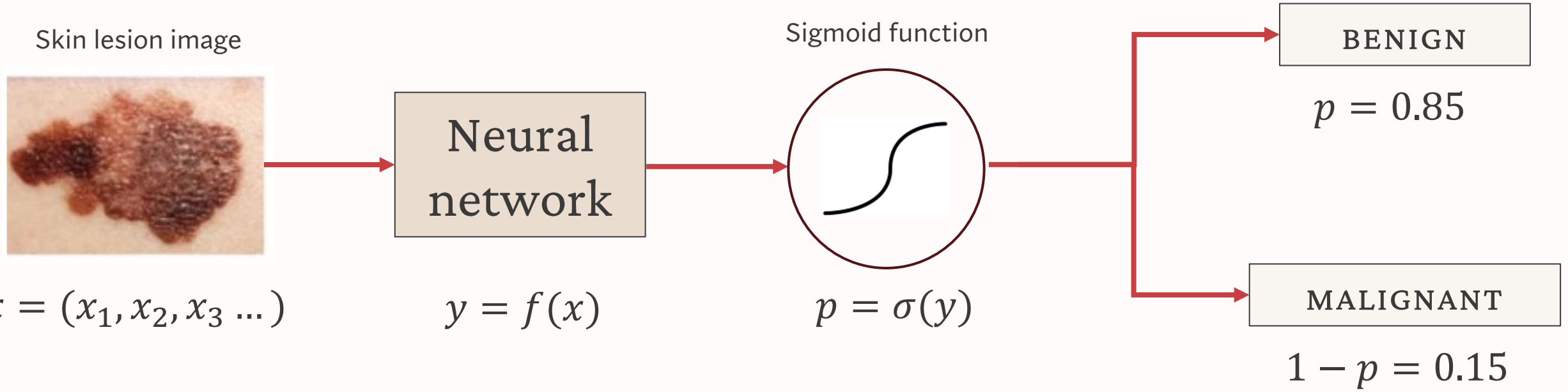
$p = \sigma(y)$

Loss function for classification

- Optimization strategy: GRADIENT DESCENT:
  1. Choose parameters $w_1, b_1, w_2, b_2, \ldots$ randomly (terrible fit!).
  2. Calculate derivative/gradient of loss $L(w_1, b_1, w_2, b_2, \ldots)$.
  3. Adjust parameters by small amount in direction of gradient.
  4. Repeat 2-3 until $L(w_1, b_1, w_2, b_2, \ldots)$ is low (good fit!).
- *How to choose loss function?*
- Want:
  - Inaccurate predictions $\longrightarrow L$ high.
  - Accurate predictions $\longrightarrow L$ low.

Simple case: two-way classification

- Common choice that works: LOG-LIKELIHOOD

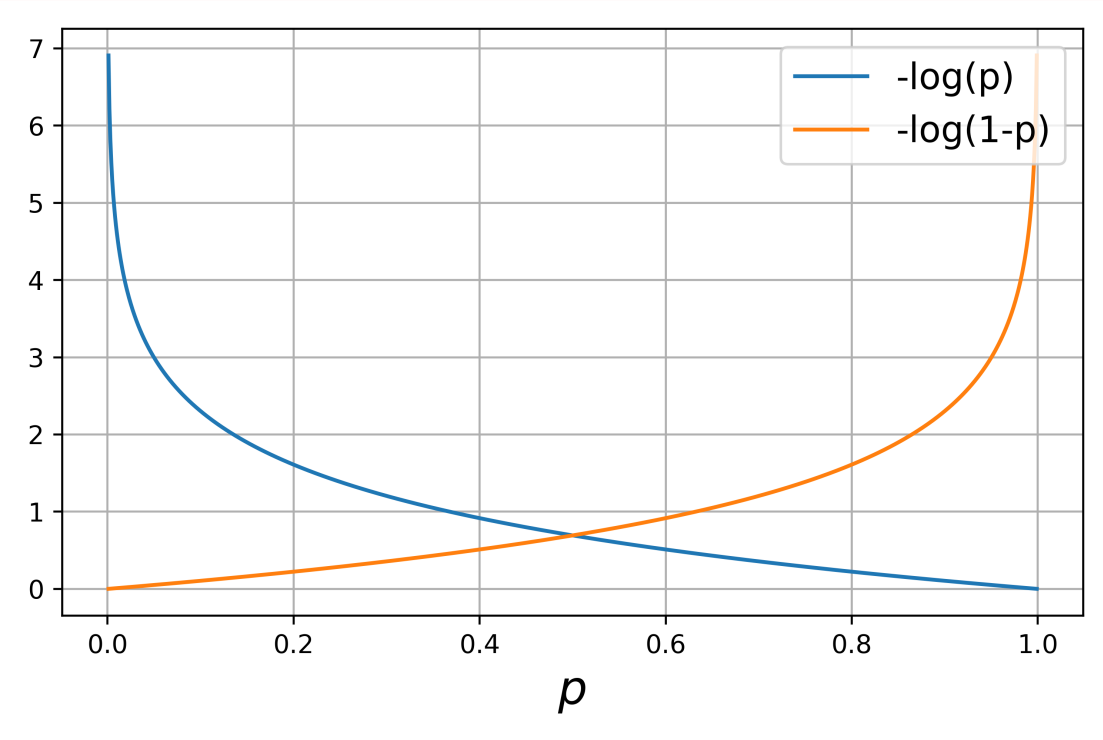


$$\text{Loss} = L = \begin{cases} -\log(p), & \text{if true label is benign} \\ -\log(1-p), & \text{if true label is malignant} \end{cases}$$

Loss function for classification

$$L(p) = \begin{cases} -\log(p), & \text{if true label is benign} \\ -\log(1-p), & \text{if true label is malignant} \end{cases}$$

- If true label benign:
  - Orange curve.
  - Best loss when $p \to 1$.
- If true label malignant:
  - Blue curve.
  - Best loss when $p \to 0$.
- Loss low if correct and confident.
- Loss high if wrong and confident.



MNIST example: binary version

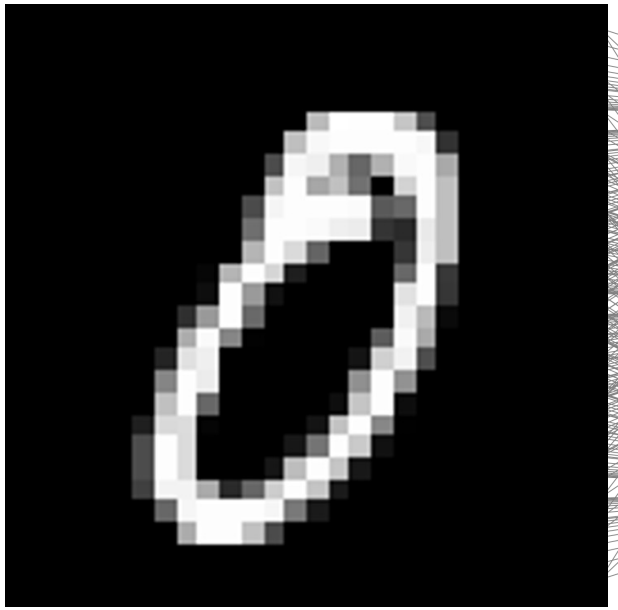


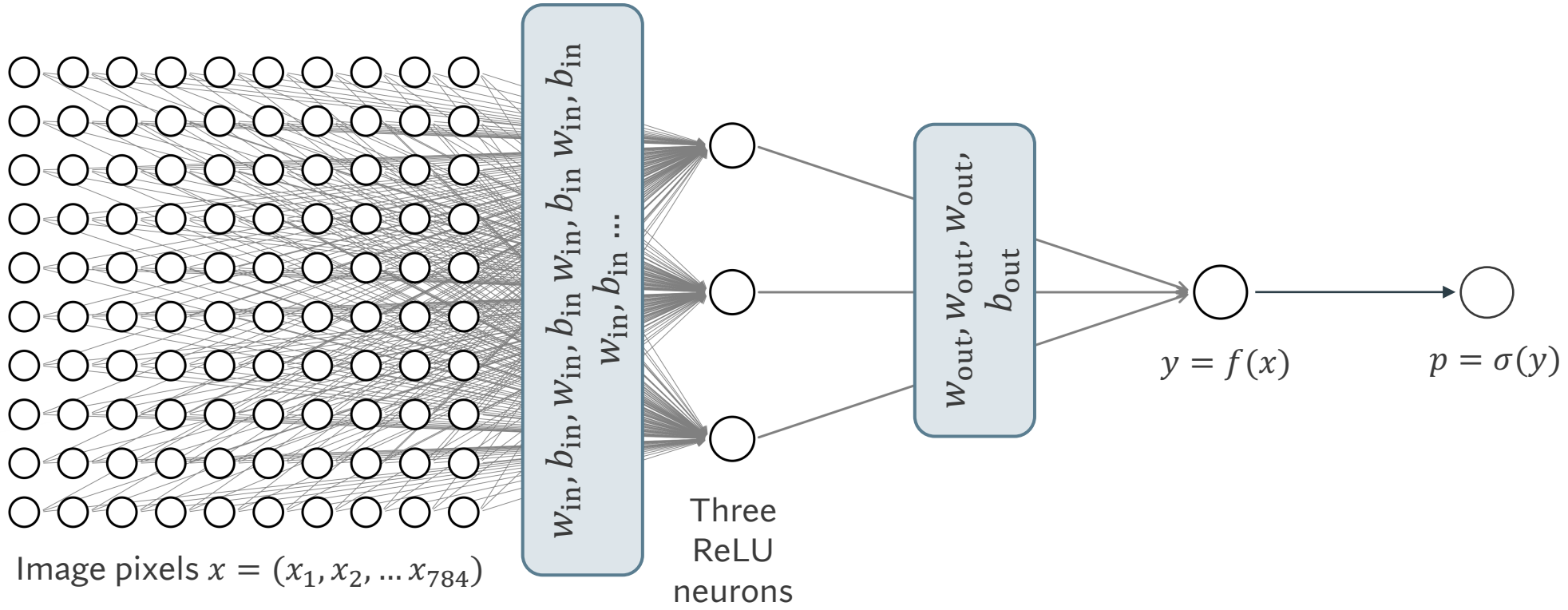
Image pixels $x = (x_1, x_2, \ldots x_{784})$

Three
ReLU
neurons

$y = f(x)$

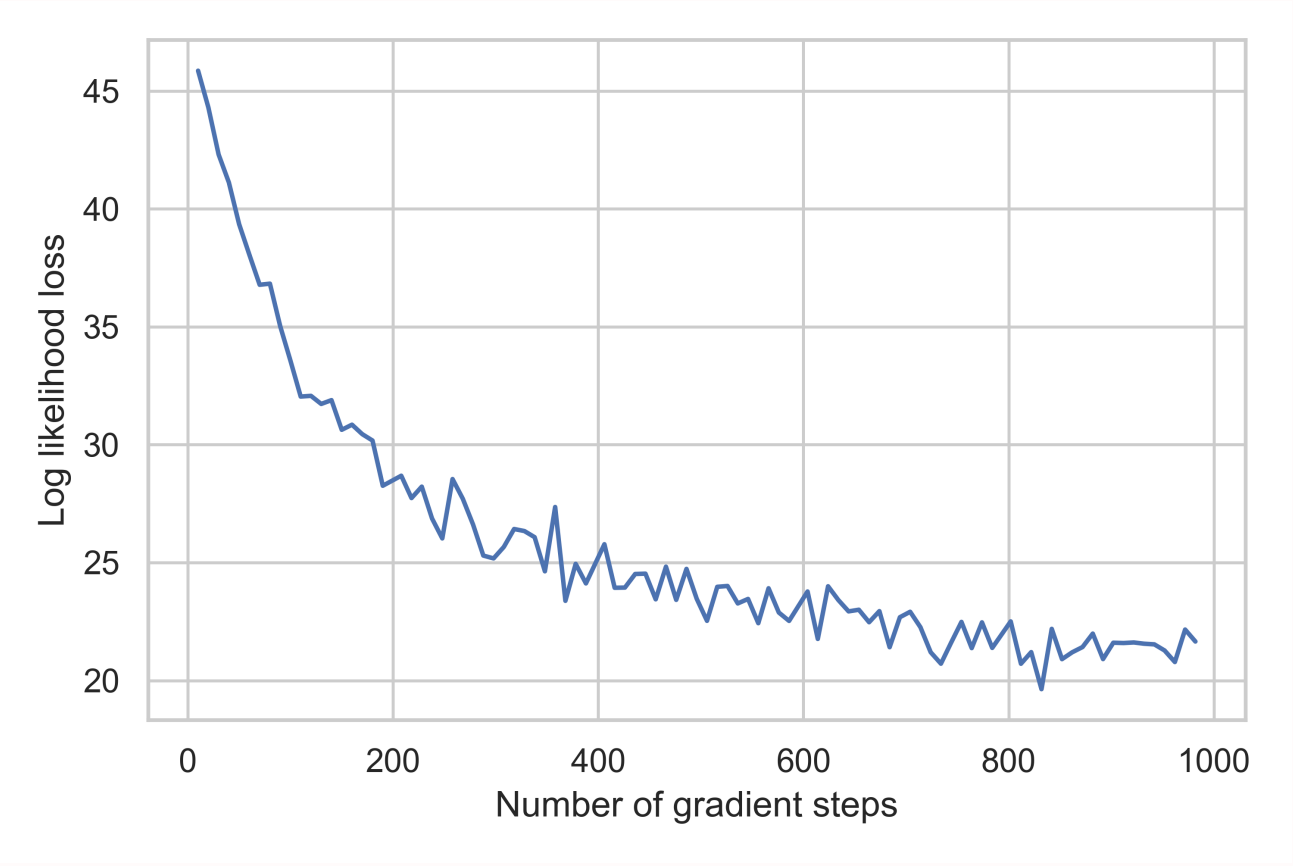
$p = \sigma(y)$

MNIST example: binary version



MNIST example: binary version

- Run gradient descent.
- Evaluate on validation set.
  - Accuracy before training:
    - 53.7% (random guess)
  - Accuracy after training:
    - 98.9% (almost perfect)

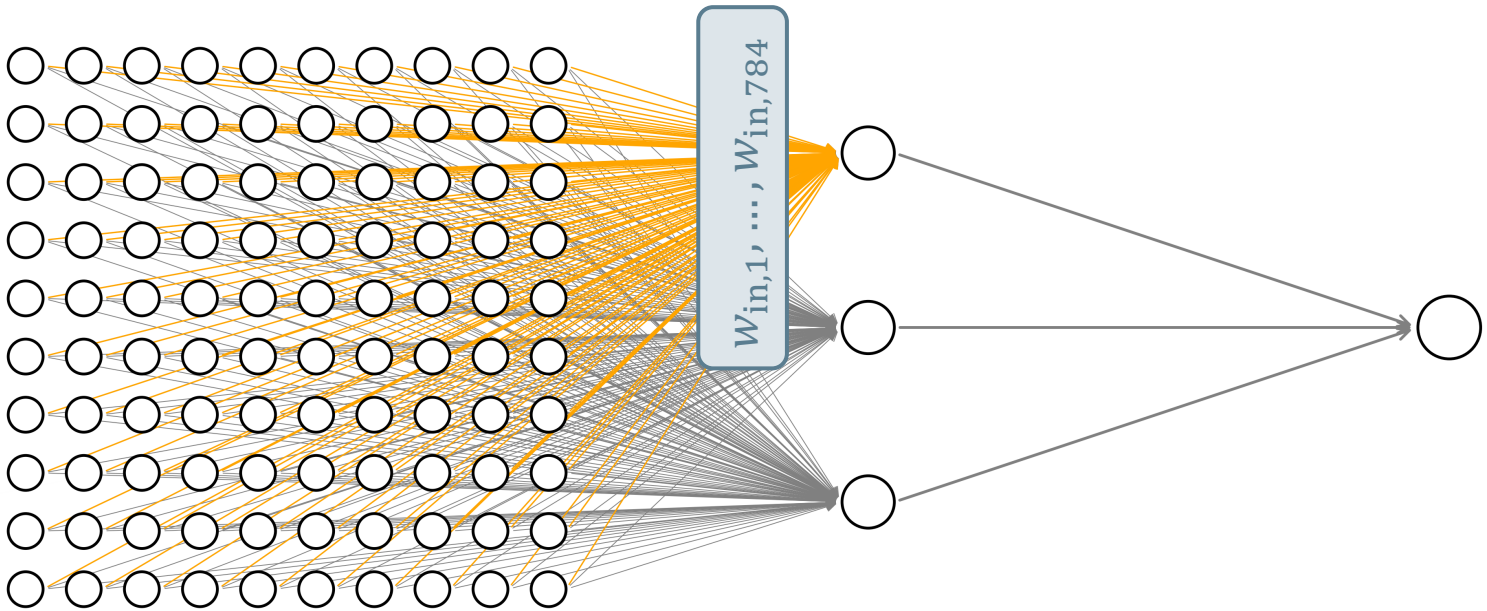


Interpreting the features

- How does network make decision?
  - Imagine cancer diagnosis!
- Look at *features* learned during gradient descent.

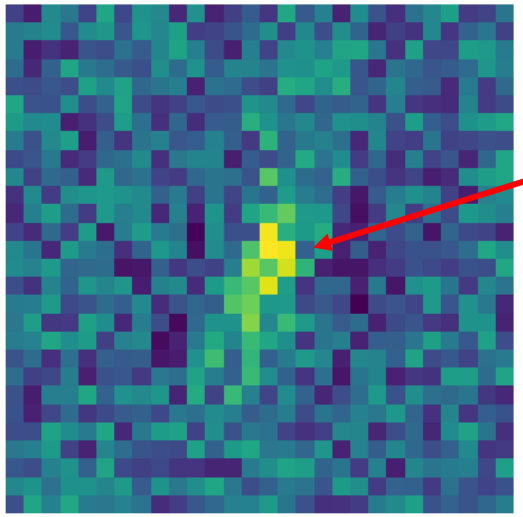
Interpreting the features

- Neuron detects if $w_{\text{in},1}x_1 + w_{\text{in},2}x_2 + \cdots + w_{\text{in},784}x_{784} + b_{\text{in}} > 0$
- Take $w_{\text{in},1}, \ldots, w_{\text{in},784}$ for first neuron and plot in grid:



Interpreting the features

- Neuron detects if $w_{\text{in},1}x_1 + w_{\text{in},2}x_2 + \cdots + w_{\text{in},784}x_{784} + b_{\text{in}} > 0$
- Take $w_{\text{in},1}, \ldots, w_{\text{in},784}$ for first neuron and plot in grid:

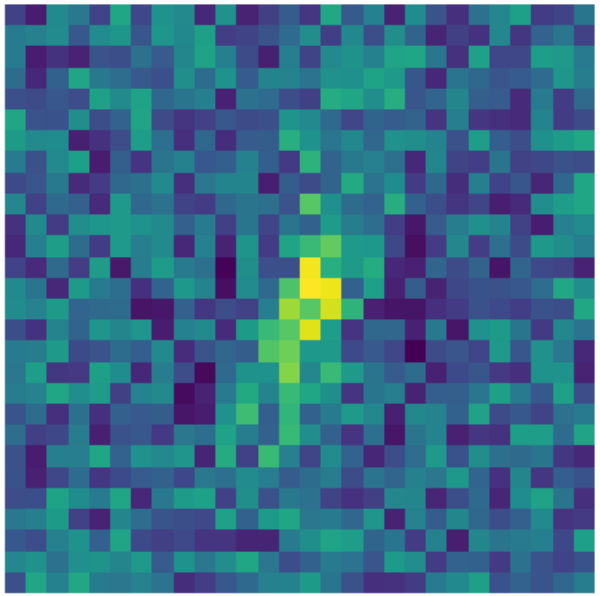
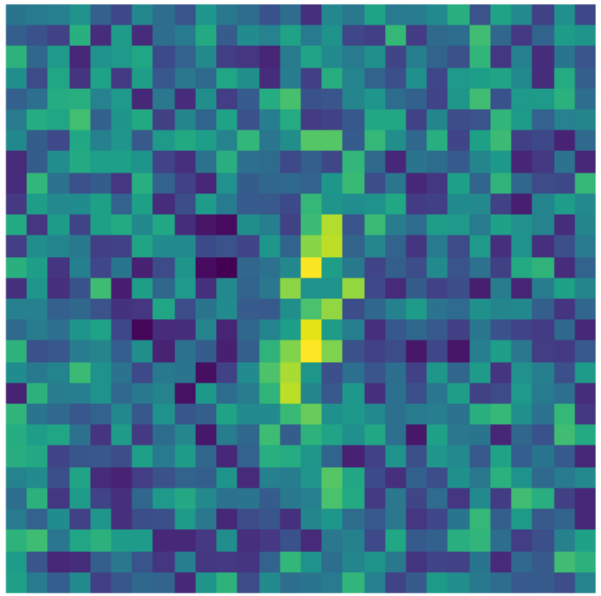
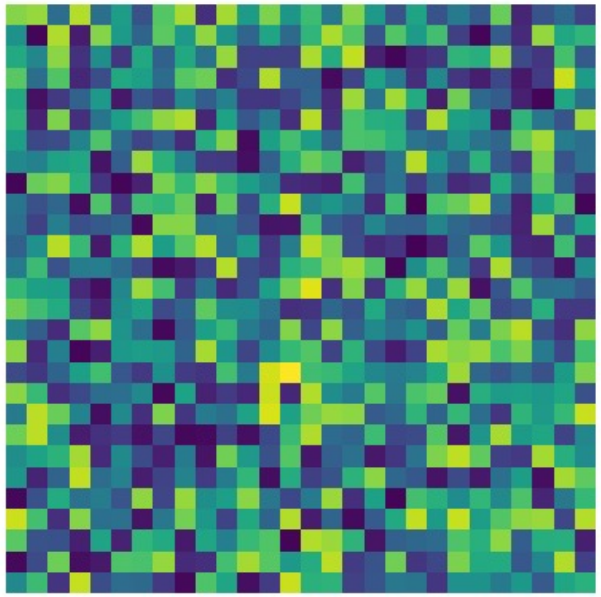


Detects if center of image is bright!

- Great way to distinguish “1” vs “0”.
  - Learned automatically.
  - Equivalent for cancer diagnosis…?

Interpreting the features

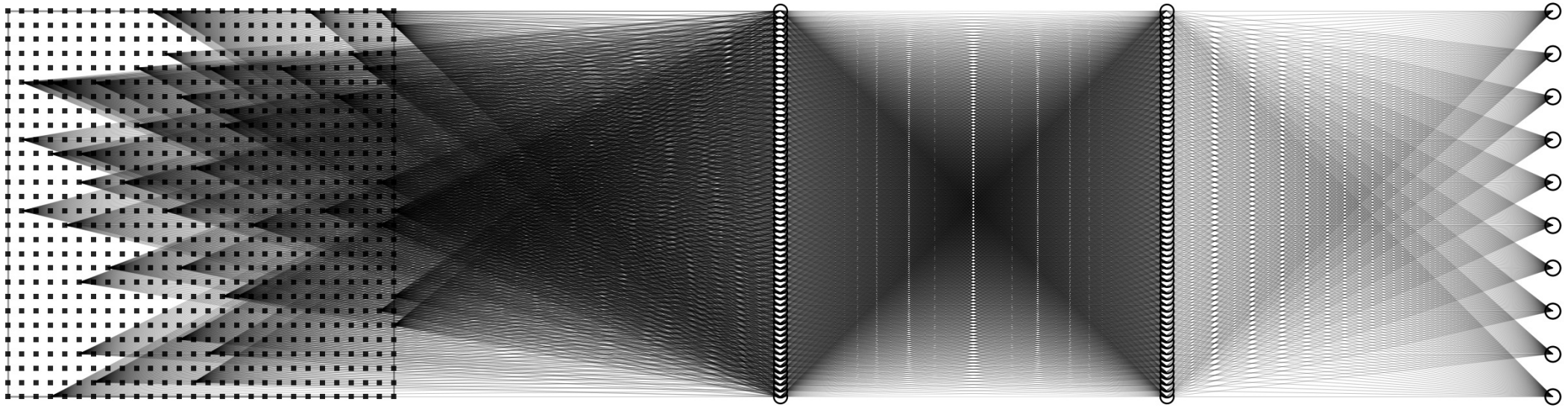
- All three neuron features:

- Not all interpretable!
- Challenge of deep learning:
  - Can construct accurate model *without understanding how it makes its predictions.*

Deeper networks

- In practice, most neural networks have *more than one hidden layer*.
  - The *outputs* of one layer and the *inputs* of the next.
  - These are called *deep neural networks*.
  - Typically, the deeper we go, the harder it is to interpret the weights.



Question & Answer