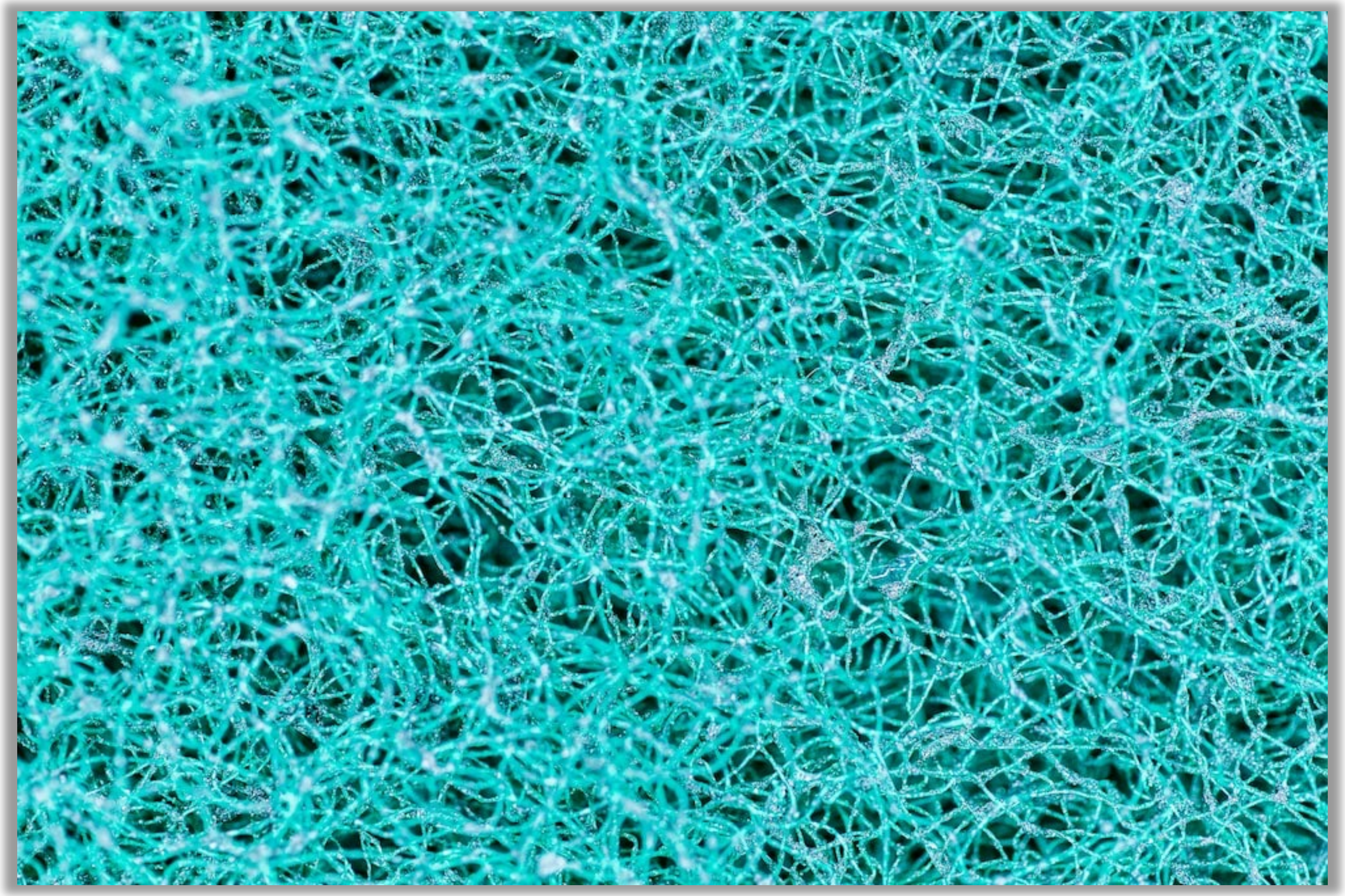
Intro to Deep Learning

neural networks, CNNs, backpropagation



CSC420
David Lindell
University of Toronto
cs.toronto.edu/~lindell/teaching/420
Slide credit: Babak Taati ←Ahmed Ashraf ←Sanja Fidler

Toronto Computational Imaging Group

Logistics

- HW2 is out, due in 3 weeks

Overview

- Motivation
- Fully-connected Networks
- Convolutional Neural Networks
- Training networks

The Recognition Tasks

- Let's take some typical tourist picture. What all do we want to recognize?

[Adopted from S. Lazebnik]

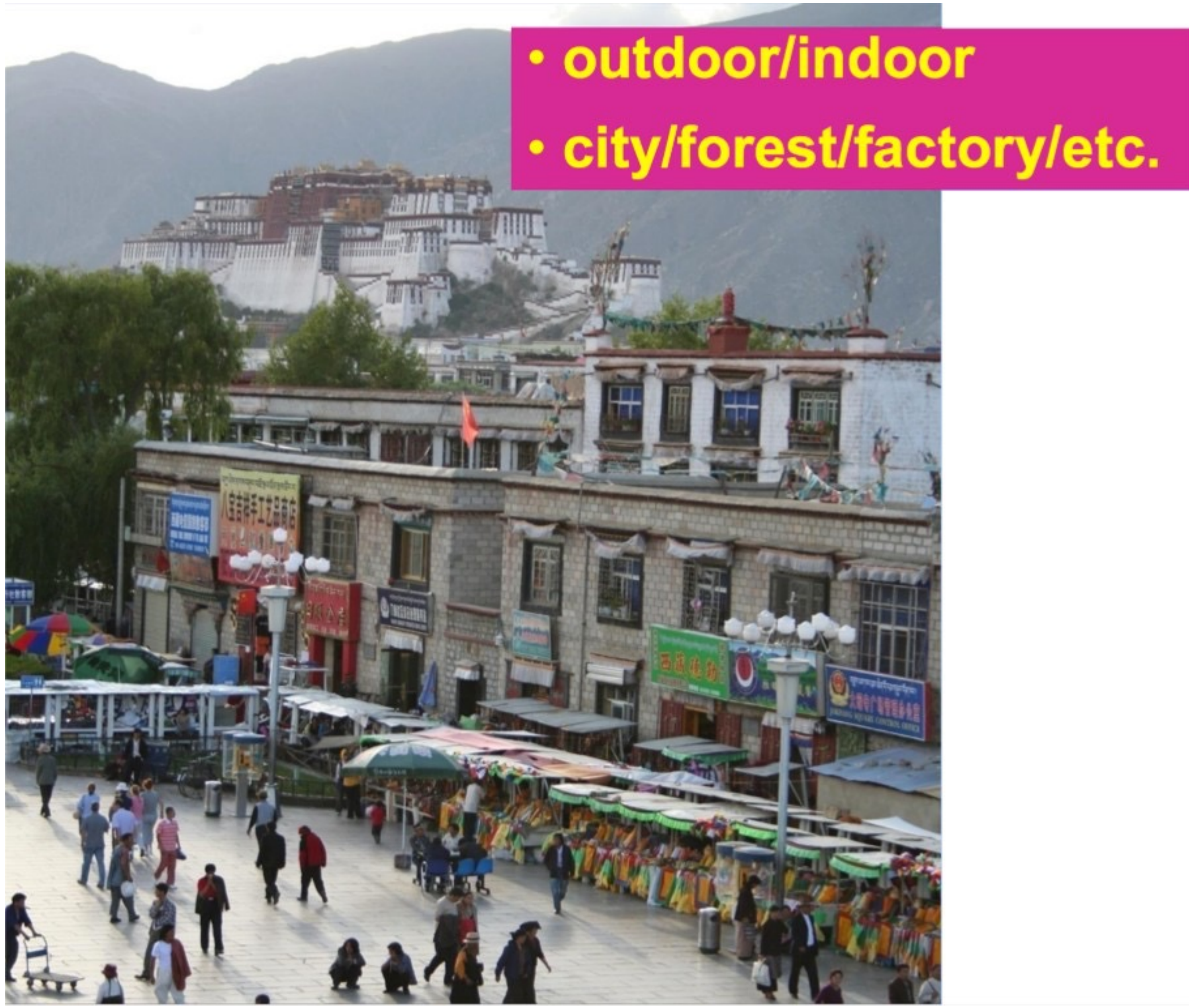
The Recognition Tasks

- Identification: we know this one (like our DVD recognition pipeline)

[Adopted from S. Lazebnik]

The Recognition Tasks

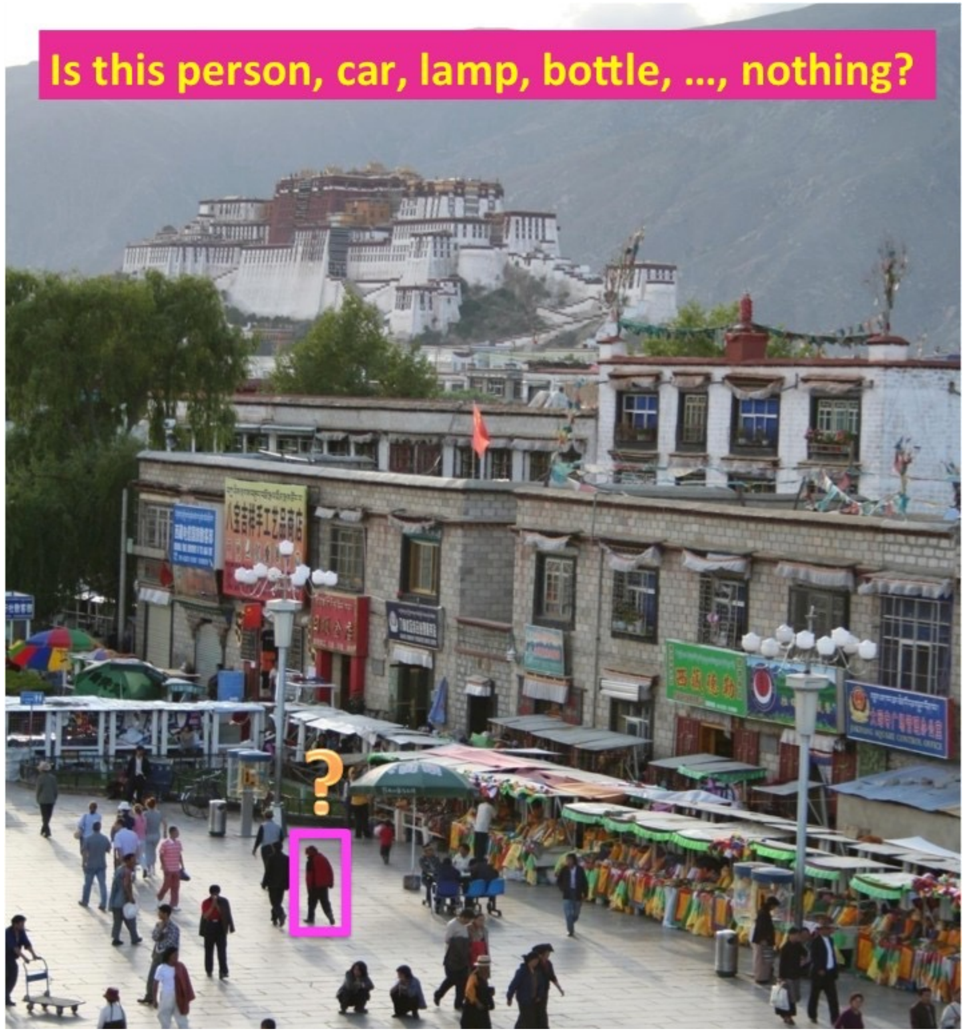
- Scene classification: what type of scene is the picture showing?



[Adopted from S. Lazebnik]

The Recognition Tasks

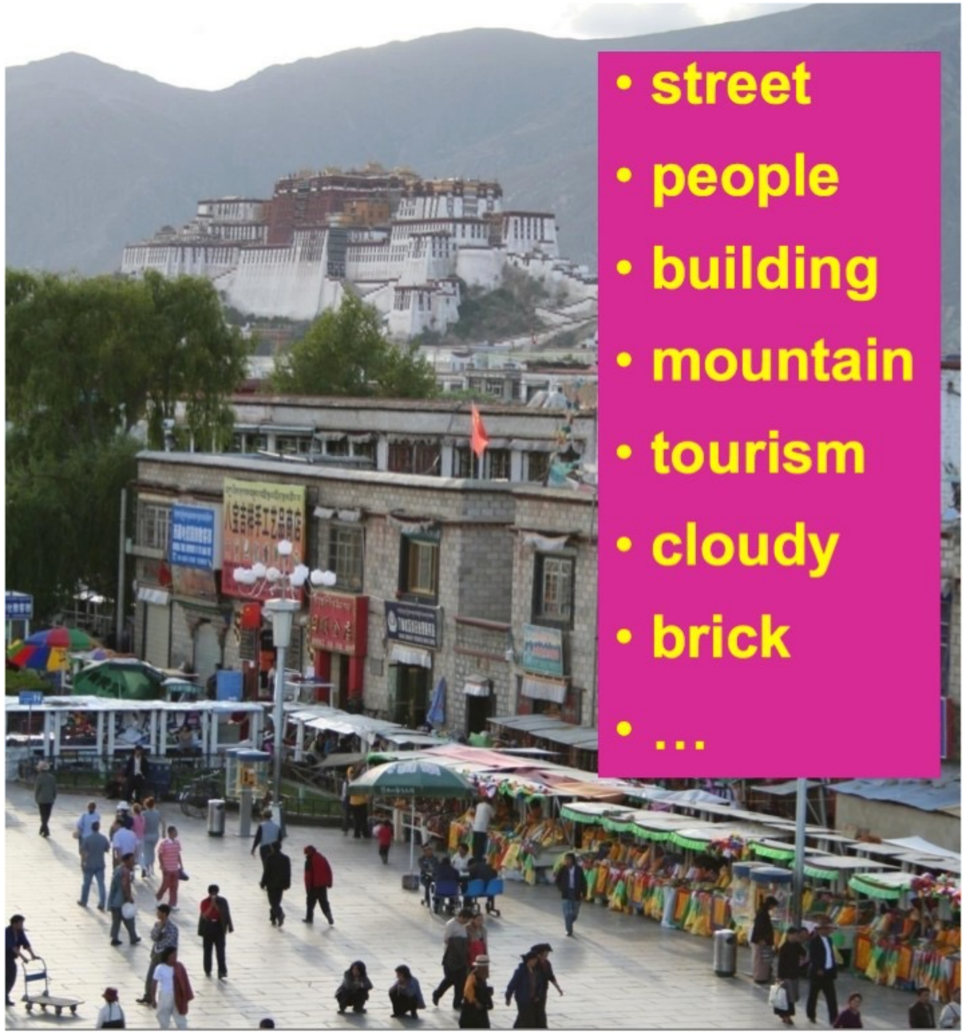
- Classification: Is the object in the window a person, a car, etc



[Adopted from S. Lazebnik]

The Recognition Tasks

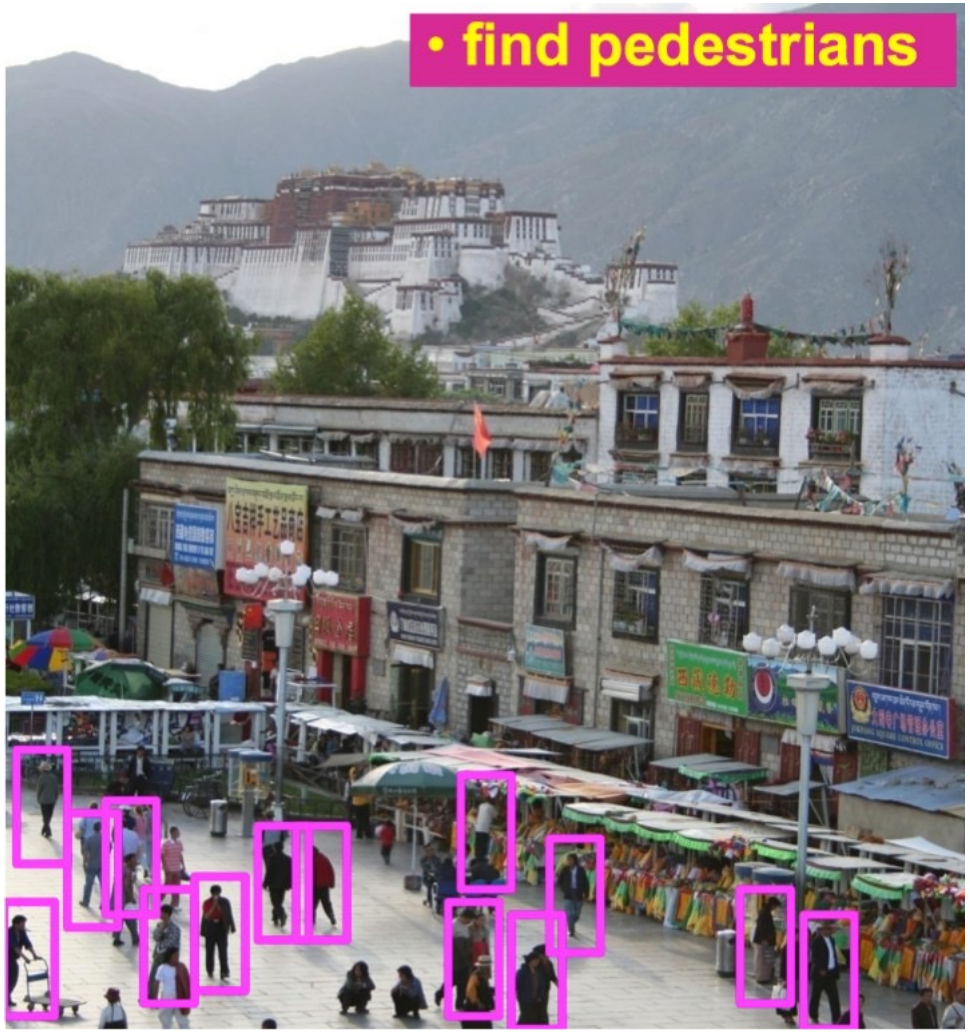
- Image Annotation: Which types of objects are present in the scene?



[Adopted from S. Lazebnik]

The Recognition Tasks

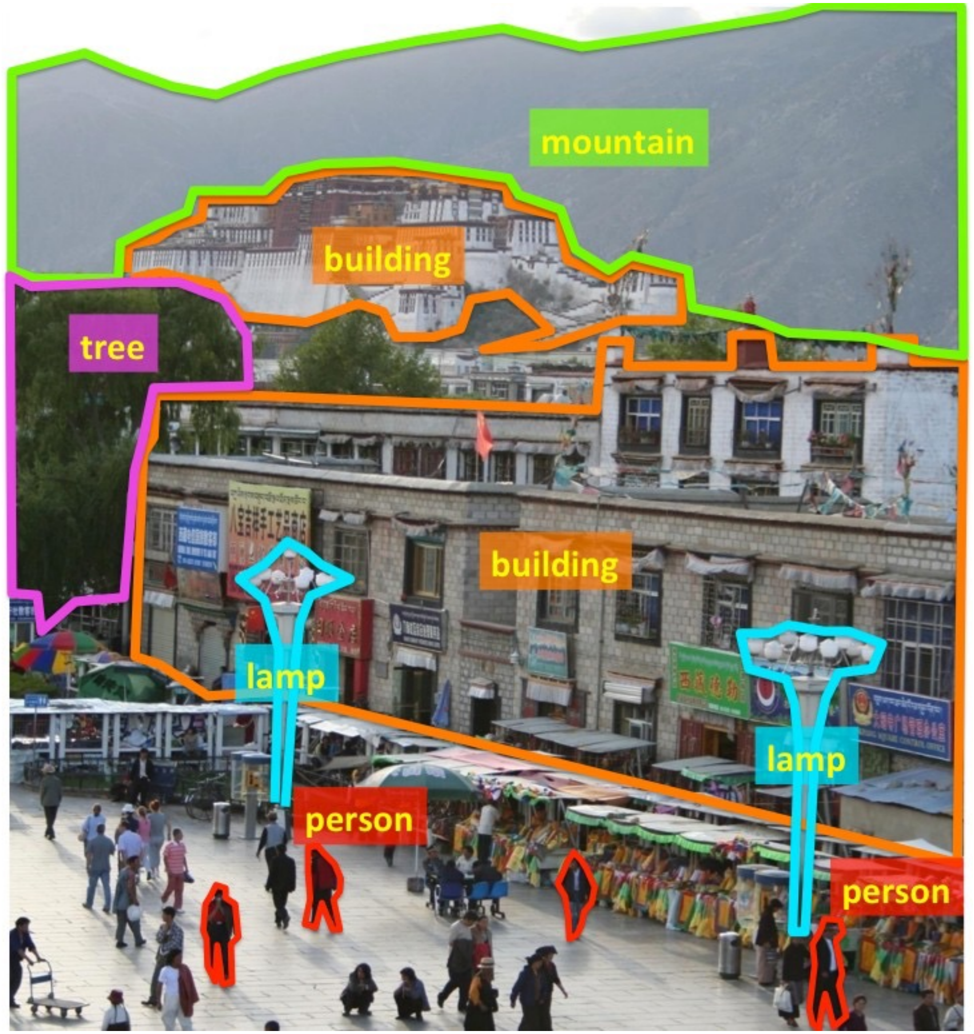
- Detection: Where are all objects of a particular class?



[Adopted from S. Lazebnik]

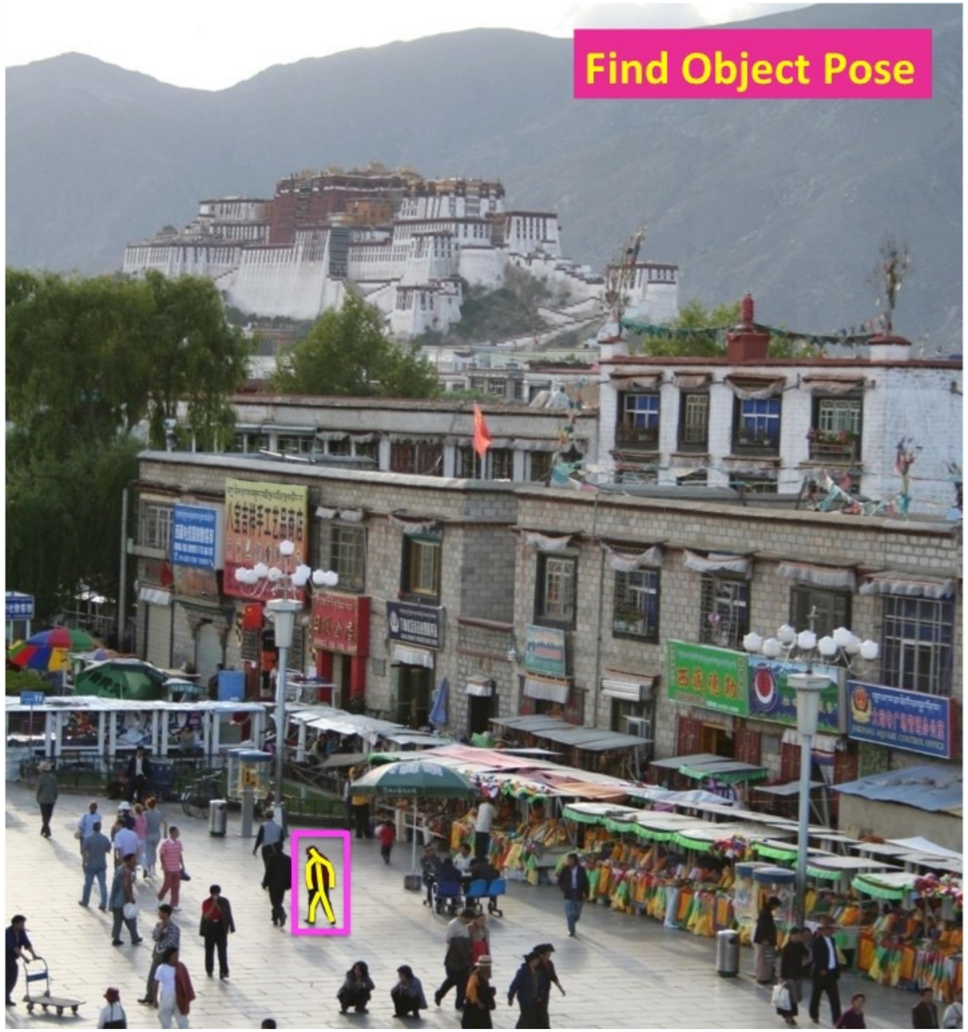
The Recognition Tasks

- Segmentation: Which pixels belong to each class of objects?



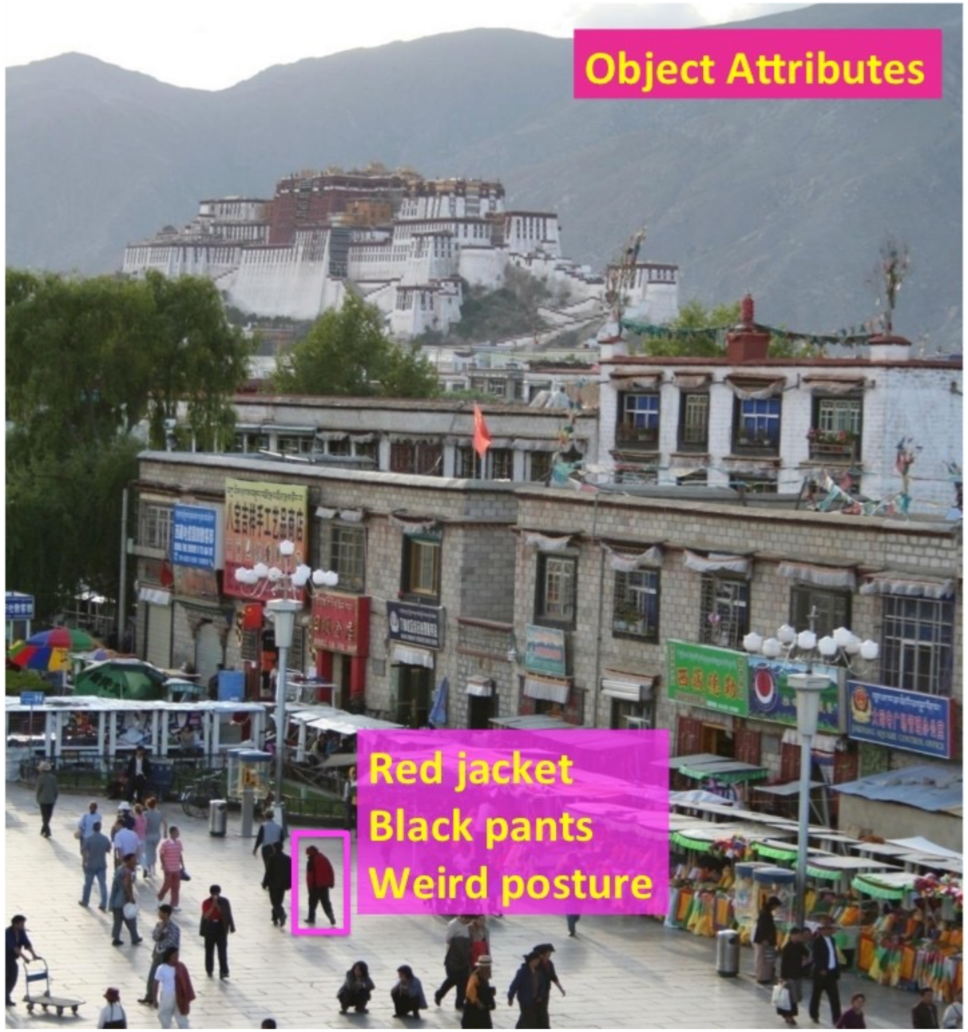
The Recognition Tasks

- Pose estimation: What is the pose of each object?



The Recognition Tasks

- Attribute recognition: Estimate attributes of the objects (color, size, etc)



The Recognition Tasks

- Action recognition: What is happening in the image?

The Recognition Tasks

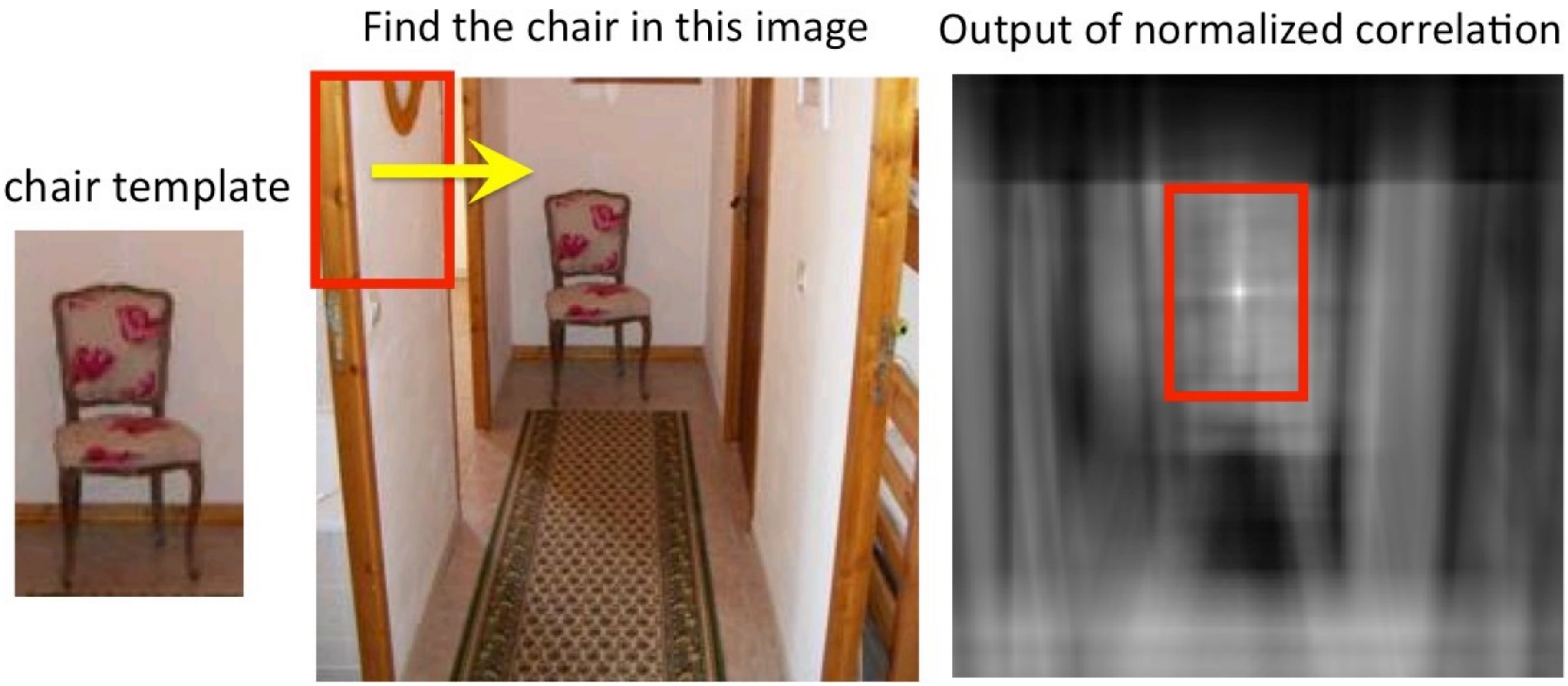
- Surveillance: Why is something happening?

Have we encountered these things before?

- Before we proceed, let's first give a shot to the techniques we already know
- Let's try detection
- These techniques are:
    - Template matching (remember Waldo in Lecture 3-5?)
    - Large-scale retrieval: store millions of pictures, recognize new one by finding the most similar one in database. This is a Google approach.

Template Matching

- Template matching: normalized cross-correlation with a template (filter)



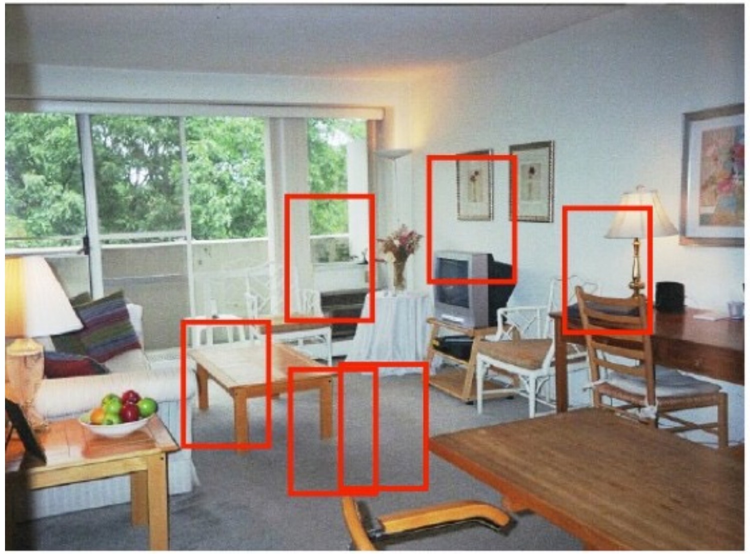
[Slide from: A. Torralba]

Template Matching

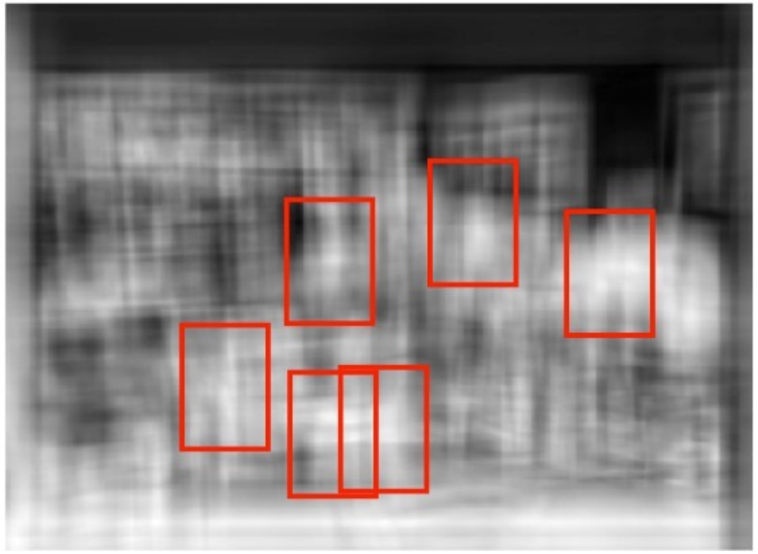
- Template matching: normalized cross-correlation with a template (filter)

template

Find the chair in this image



Pretty much garbage
Simple template matching is
not going to make it



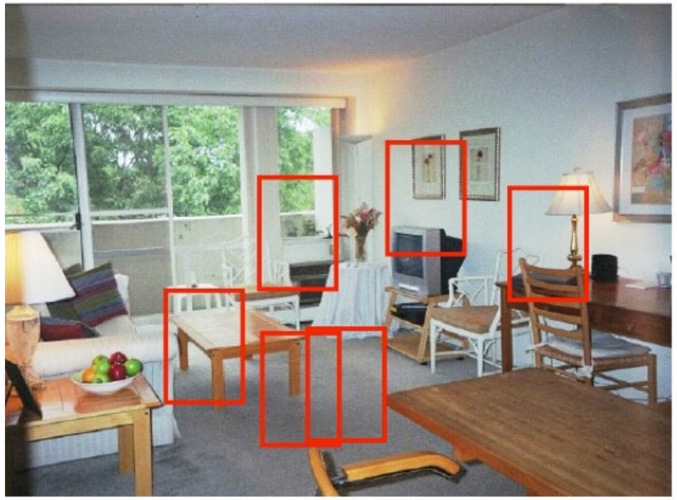
[Slide from: A. Torralba]

Template Matching

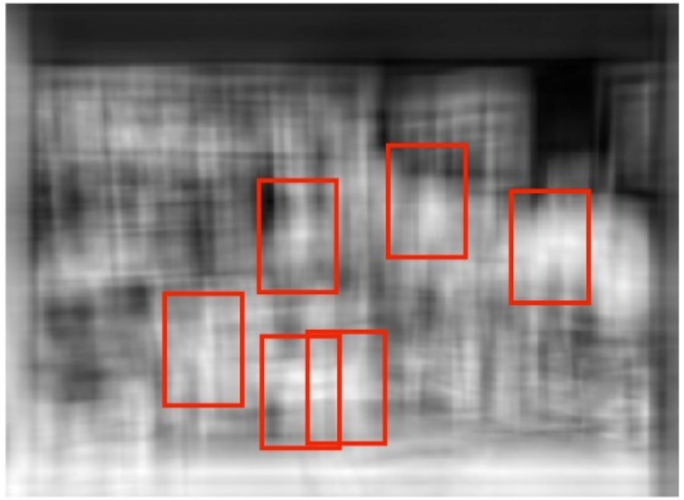
- Template matching: normalized cross-correlation with a template (filter)

template

Find the chair in this image



Pretty much garbage
Simple template matching is
not going to make it

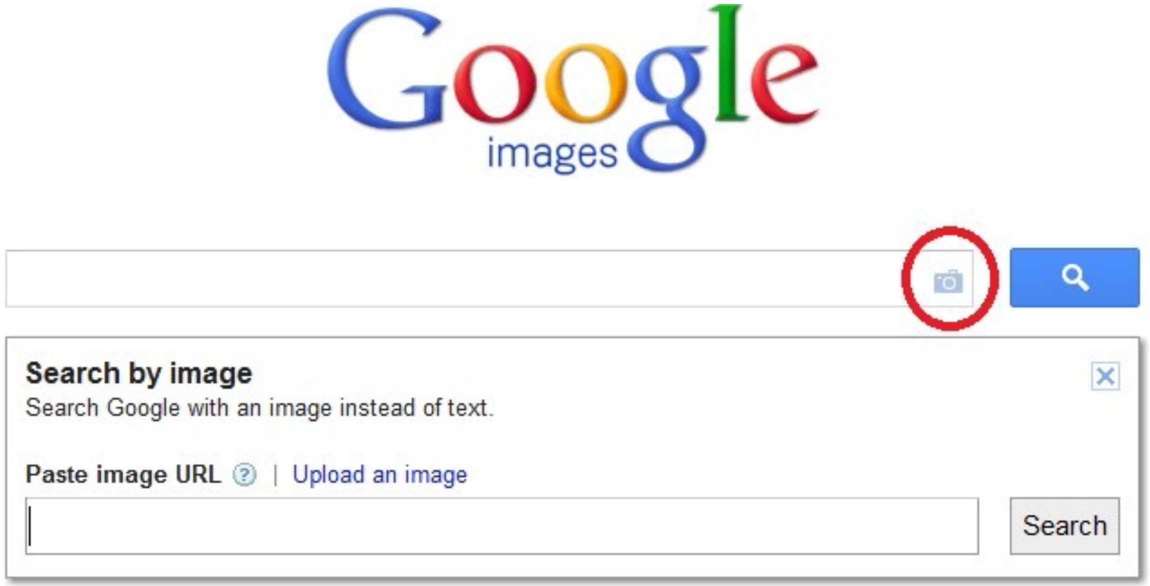


A "popular method is that of template matching, by point to point correlation of a model pattern with the image pattern. These techniques **are inadequate for three-dimensional scene analysis for many reasons, such as occlusion, changes in viewing angle, and articulation of parts**." Nevatia & Binford, 1977.

[Slide from: A. Torralba]

Recognition via Retrieval by Similarity

- Upload a photo to Google image search and check if something reasonable comes out



query

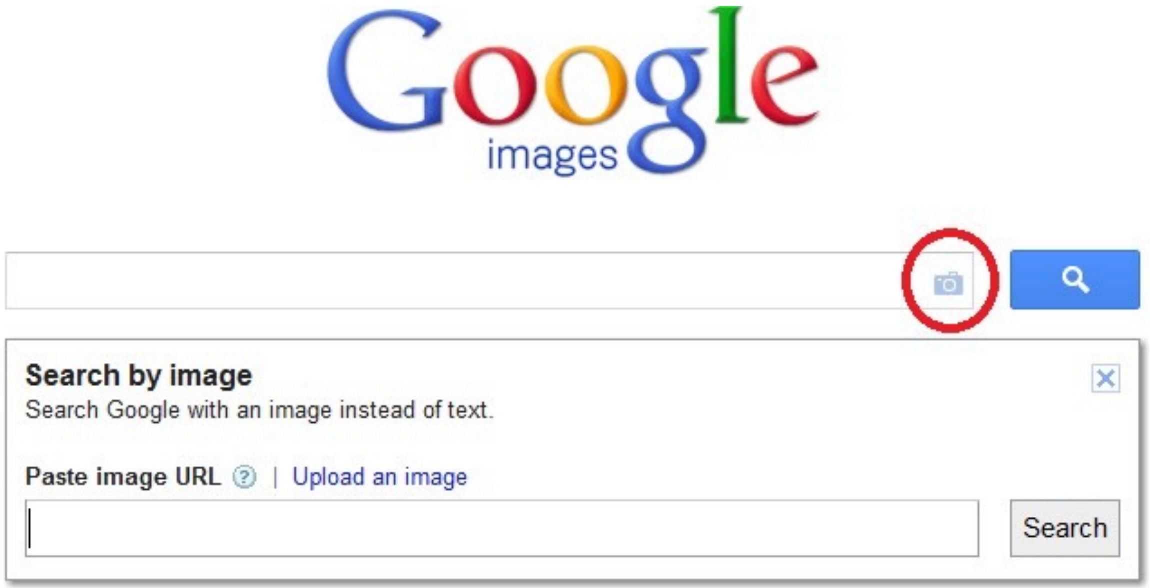
Recognition via Retrieval by Similarity

- Upload a photo to Google image search
- Pretty reasonable, both are Golden Gate Bridge

query

Recognition via Retrieval by Similarity

- Upload a photo to Google image search Let’s try a typical bathtub object



query

Recognition via Retrieval by Similarity

- Upload a photo to Google image search
- A bit less reasonable, but still some striking similarity

query

Recognition via Retrieval by Similarity

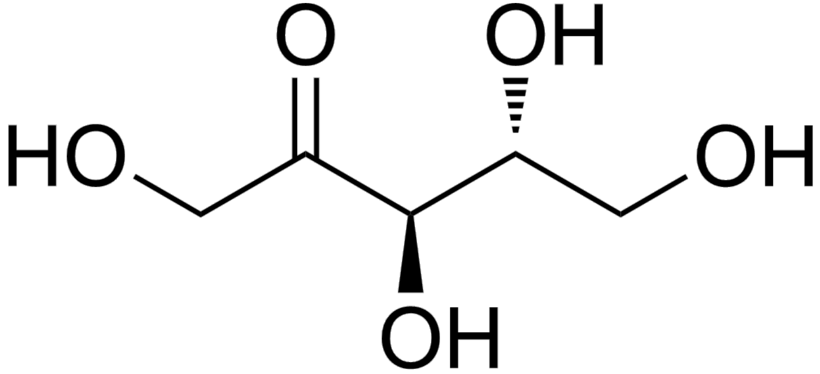
- Make a beautiful drawing and upload to Google image search
- Can you recognize this object?

query

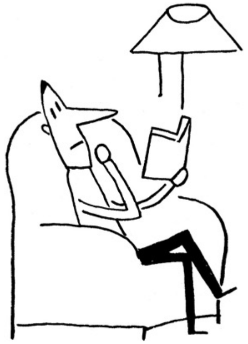
Recognition via Retrieval by Similarity

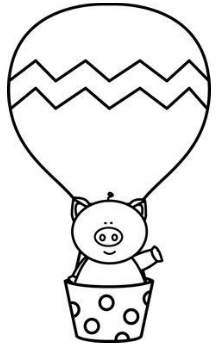
- Make a beautiful drawing and upload to Google image search
- Not a very reasonable result

query



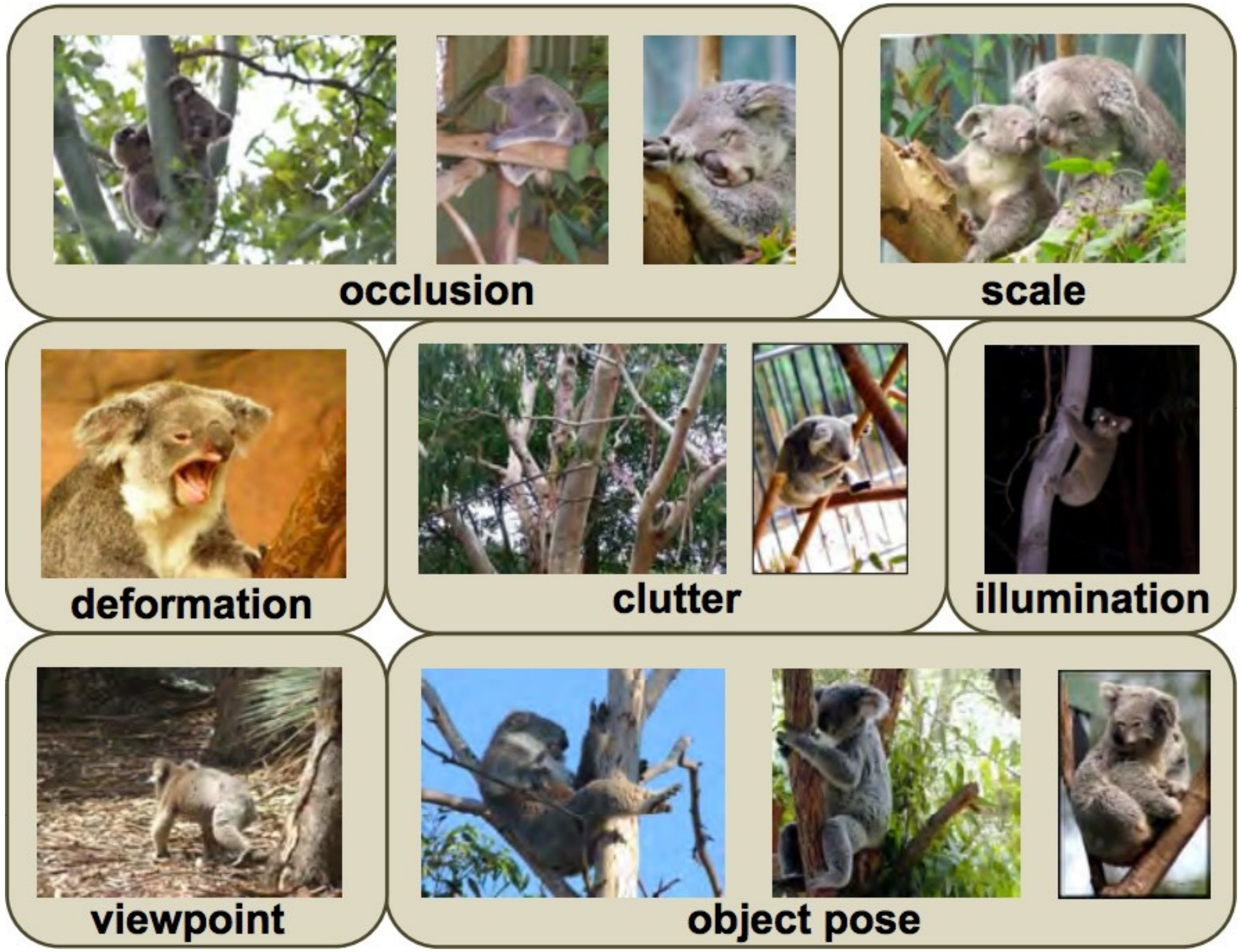
other retrieved results:





Why is it a Problem?

- Difficult scene conditions



[From: Grauman & Leibe]

Why is it a Problem?

- Huge within-class variations. Recognition is mainly about modeling variation.






[Pic from: S. Lazebnik]

Why is it a Problem?

- Tons of classes

Biederman

Overview

- We cannot explicitly model these variations!

Overview

- We cannot explicitly model these variations!
- Instead our models should be relatively simple and we should learn let complexity live in the data

Overview

- We cannot explicitly model these variations!
- Instead our models should be relatively simple and we should learn let complexity live in the data
- Neural networks follow this paradigm

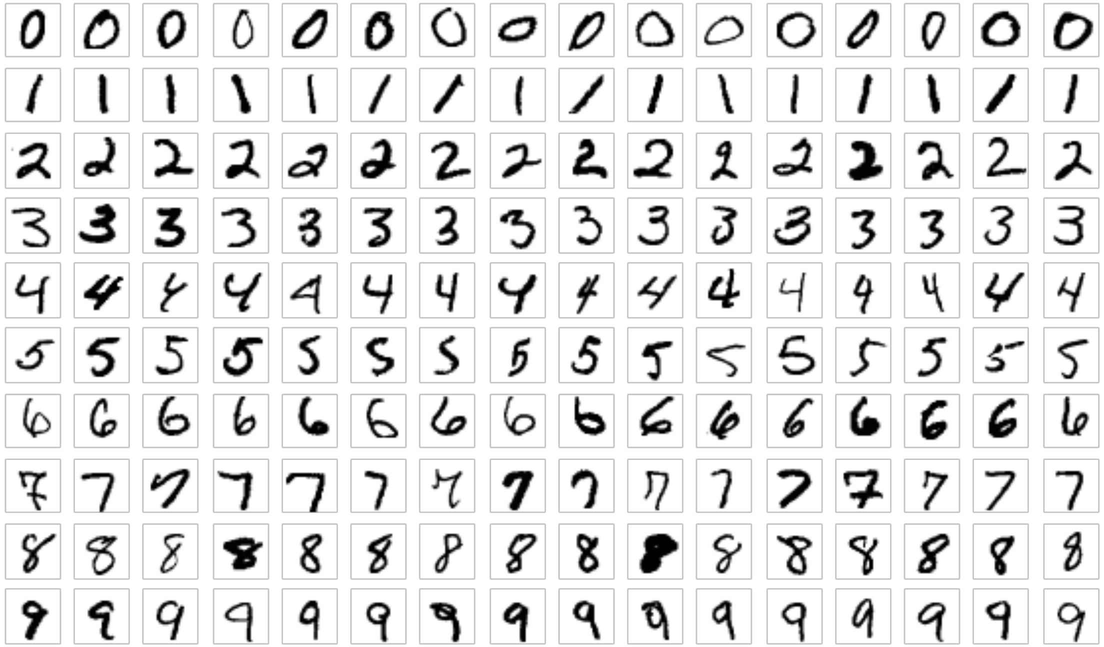
Overview

- Motivation
- Fully-connected Networks
- Convolutional Neural Networks
- Training networks

Image Classification

- Image classification example

Images

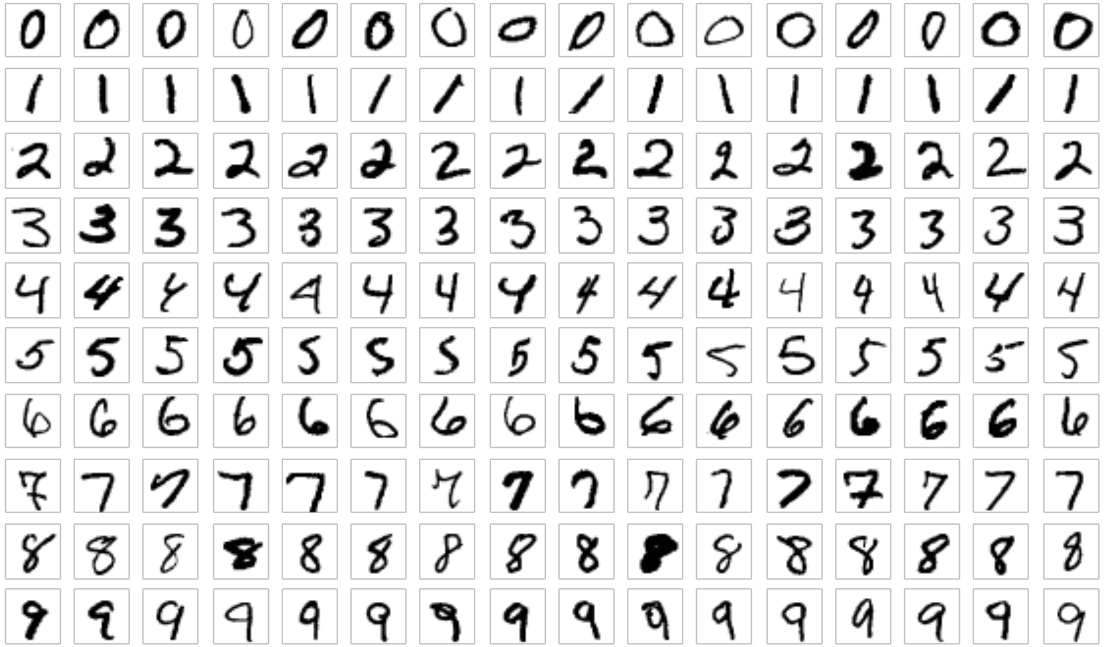


MNIST Dataset

Image Classification

- Image classification example

Images



Class

“zero”

“one”

…

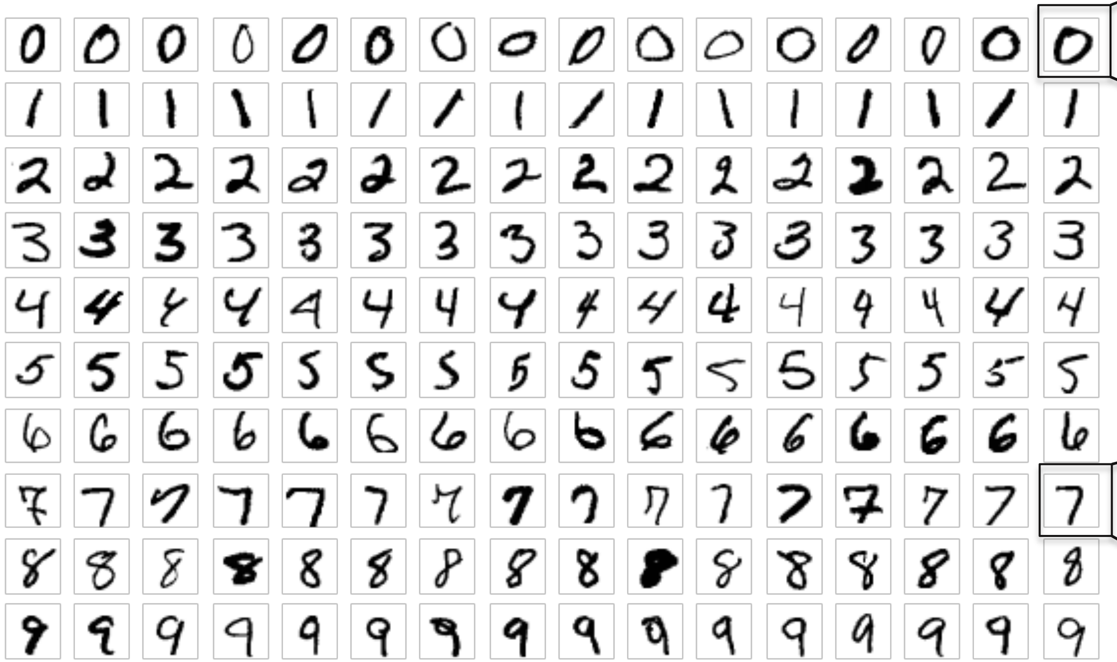
“nine”

Image Classification

- Image classification example

What the computer "sees"

Images



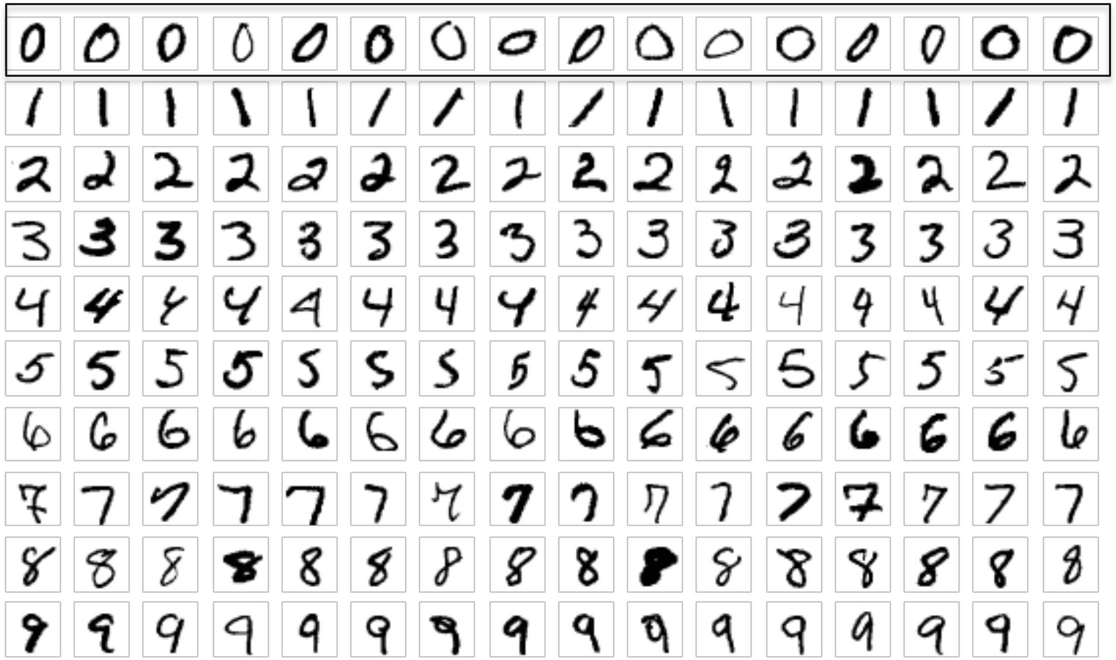
000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000
000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000
000	000	000	000	000	000	000	000	019	146	146	018	000	000	000	000
000	000	000	000	000	000	000	023	188	248	236	115	003	000	000	000
000	000	000	000	000	000	059	192	249	246	112	236	086	000	000	000
000	000	000	000	000	017	214	243	145	223	064	180	113	000	000	000
000	000	000	000	014	161	232	137	006	014	000	156	209	003	000	000
000	000	000	000	108	230	042	009	000	000	000	155	227	004	000	000
000	000	000	020	223	138	000	000	000	000	000	156	220	003	000	000
000	000	000	054	237	023	000	000	000	000	024	206	111	000	000	000
000	000	000	055	210	003	000	000	001	057	216	088	006	000	000	000
000	000	000	055	242	085	022	089	160	219	108	001	000	000	000	000
000	000	000	040	233	246	226	242	175	070	001	000	000	000	000	000
000	000	000	000	058	211	212	069	001	000	000	000	000	000	000	000
000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000
000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000

000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000
000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000
000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000
000	000	000	000	000	000	000	000	000	001	003	002	000	000	000	000
000	000	000	000	000	001	002	035	101	171	239	229	052	000	000	000
000	000	000	000	026	195	223	242	202	219	243	250	134	000	000	000
000	000	000	002	195	249	239	096	005	024	231	229	028	000	000	000
000	000	000	086	245	200	037	000	004	079	243	138	001	000	000	000
000	000	000	033	120	016	000	000	082	245	176	007	000	000	000	000
000	000	000	000	000	000	000	015	245	227	015	000	000	000	000	000
000	000	000	000	000	000	000	112	248	110	001	000	000	000	000	000
000	000	000	000	000	001	095	225	124	006	000	000	000	000	000	000
000	000	000	000	000	026	224	177	002	000	000	000	000	000	000	000
000	000	000	000	003	172	234	038	000	000	000	000	000	000	000	000
000	000	000	000	046	234	076	002	000	000	000	000	000	000	000	000
000	000	000	000	010	087	005	000	000	000	000	000	000	000	000	000

Image Classification

- Image classification example

Images



Challenges

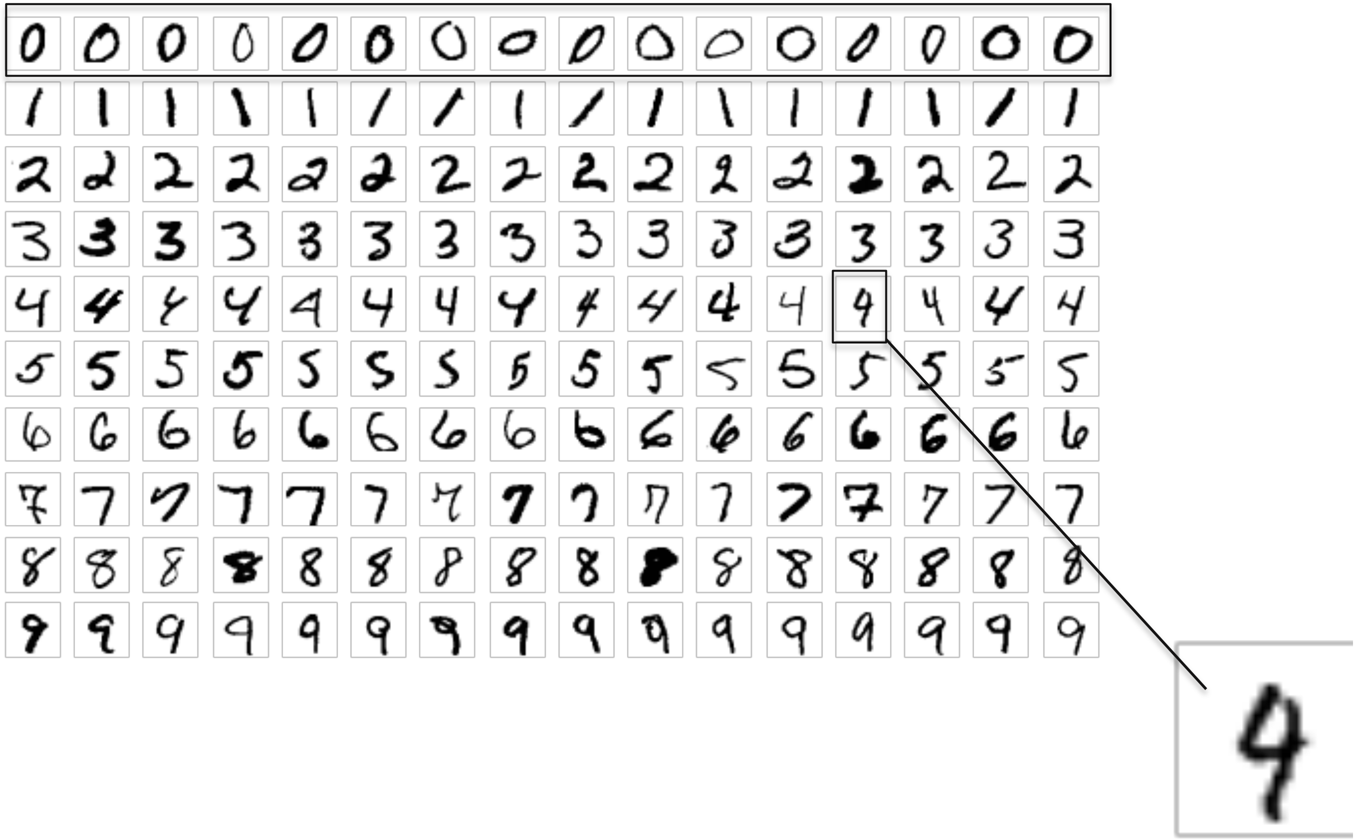
Intra-class variation

- stroke widths
- alignment
- writing styles

Image Classification

- Image classification example

Images



Challenges

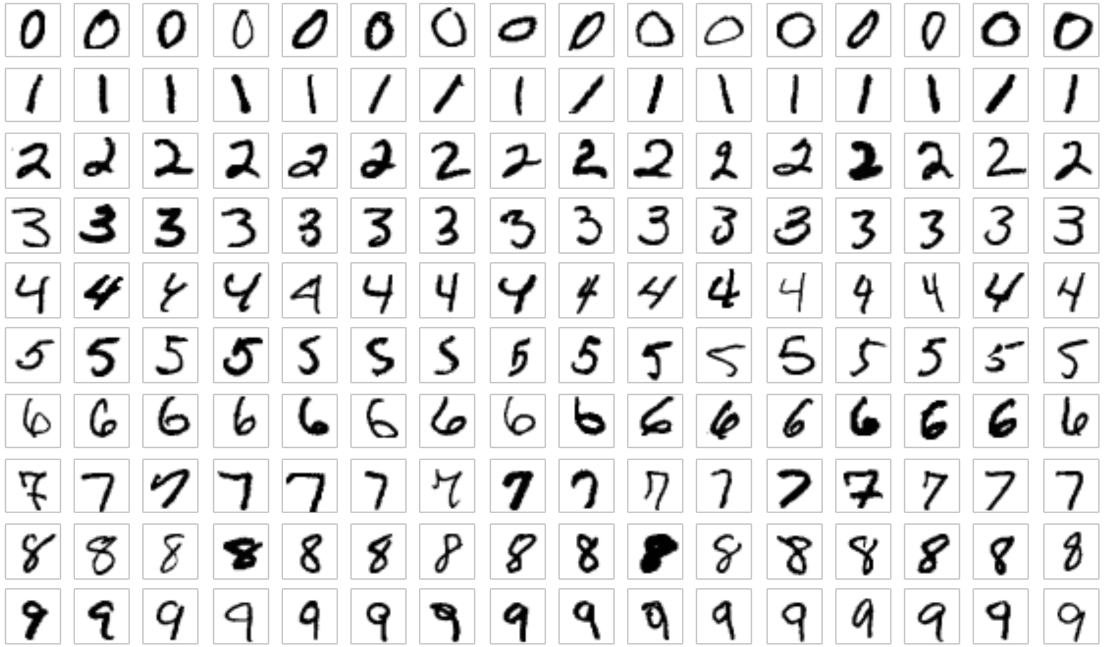
Intra-class variation
- stroke widths
- alignment
- writing styles

Inter-class similarities
- “four” or “nine”?

Image Classification

- Image classification example

Images



Implementation?

```python
def classify_digit(image):
    # ???
    return image_class
```

Can’t hardcode solution!

Image Classification

- Data-driven approach
  - Collect training images and labels
  - Train a classifier using machine learning
  - Evaluate the classifier on unseen images

Implementation?

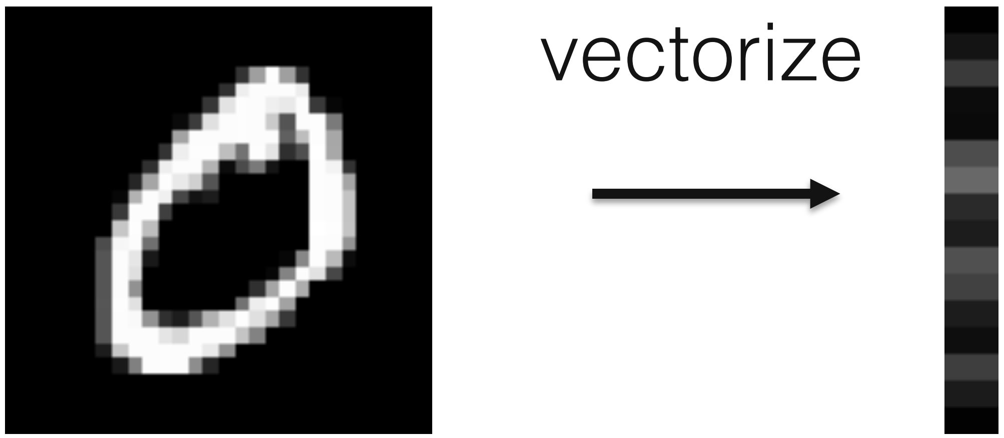
```python
def train(images, labels):
    # machine learning model
    return image_class

def evaluate(model, test_images):
    # machine learning model
    return test_labels

```

Image Classification

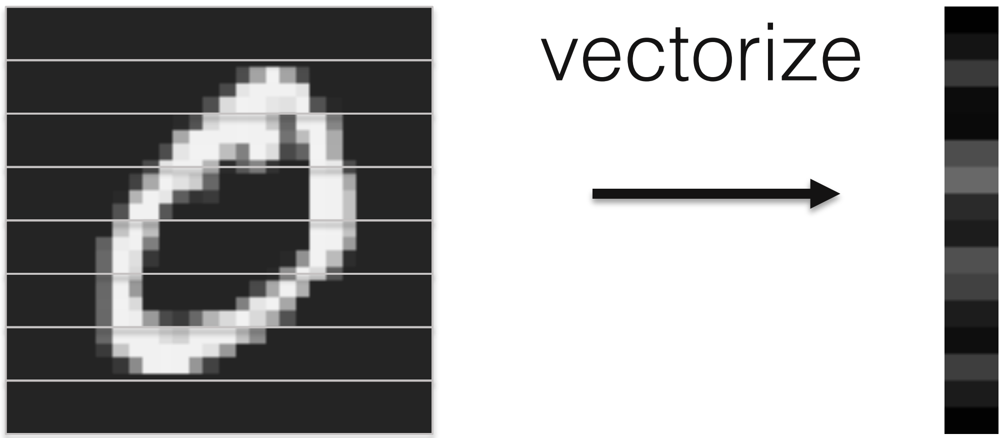
- Linear Model $f(x, W) = Wx$



x

Image Classification

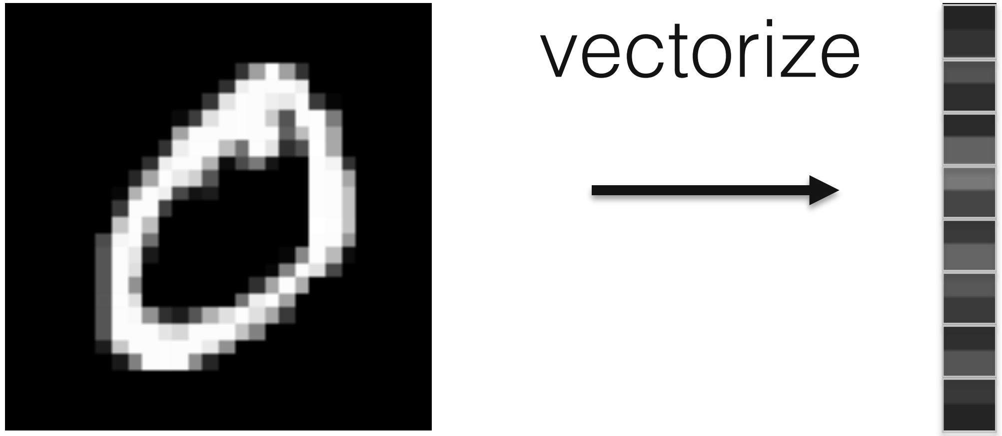
- Linear Model $$f(x, W) = Wx$$



$$x$$

Image Classification

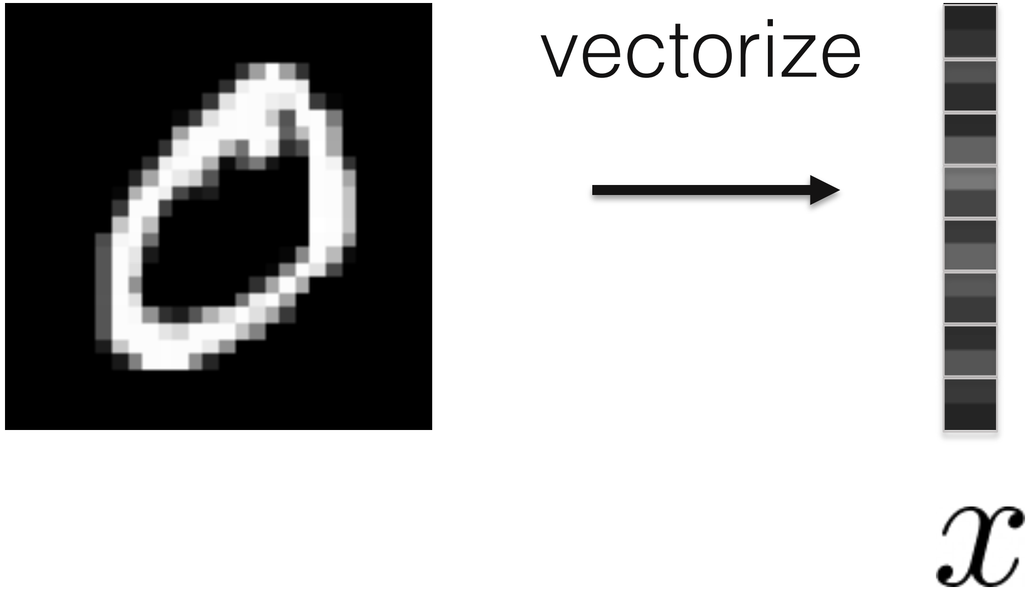
- Linear Model $f(x, W) = Wx$



x

Image Classification

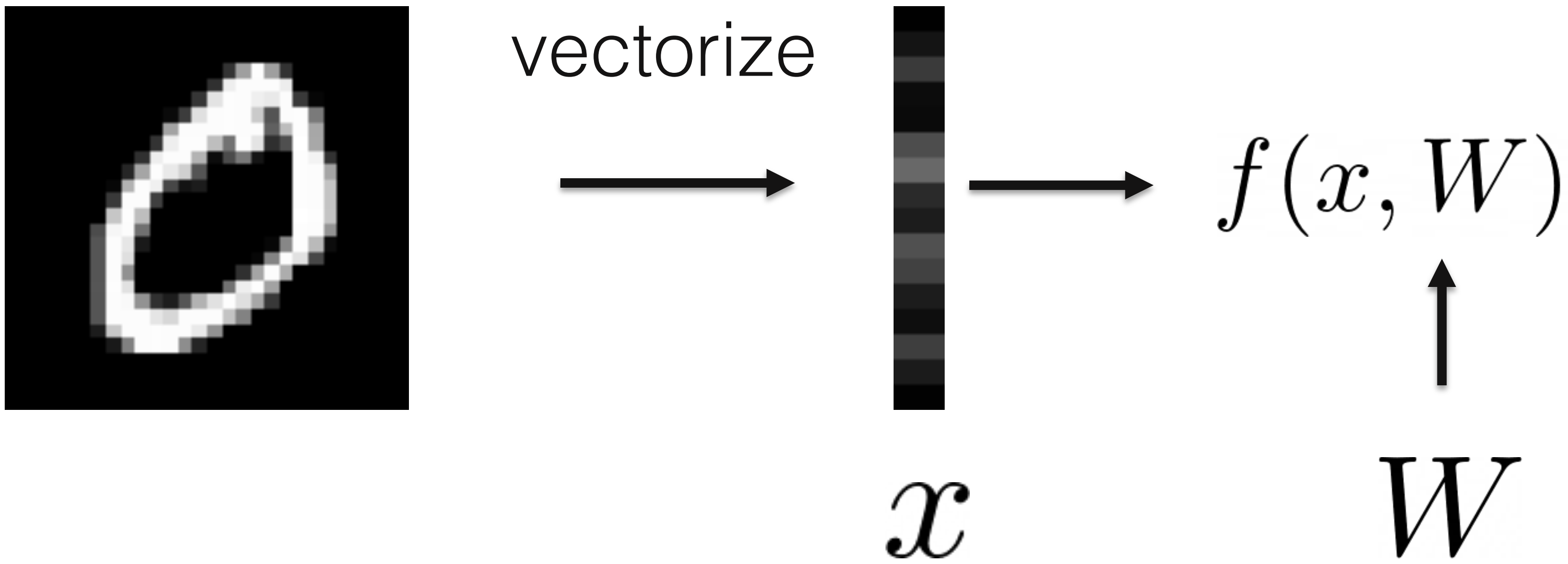
- Linear Model $$f(x, W) = Wx$$



Length of this vector is the “dimensionality” of our problem!

Image Classification

- Linear Model $$f(x, W) = Wx$$



$$\boxed{\text{In general: } Wx + b}$$

Image Classification

- Linear Model $$f(x, W) = Wx$$

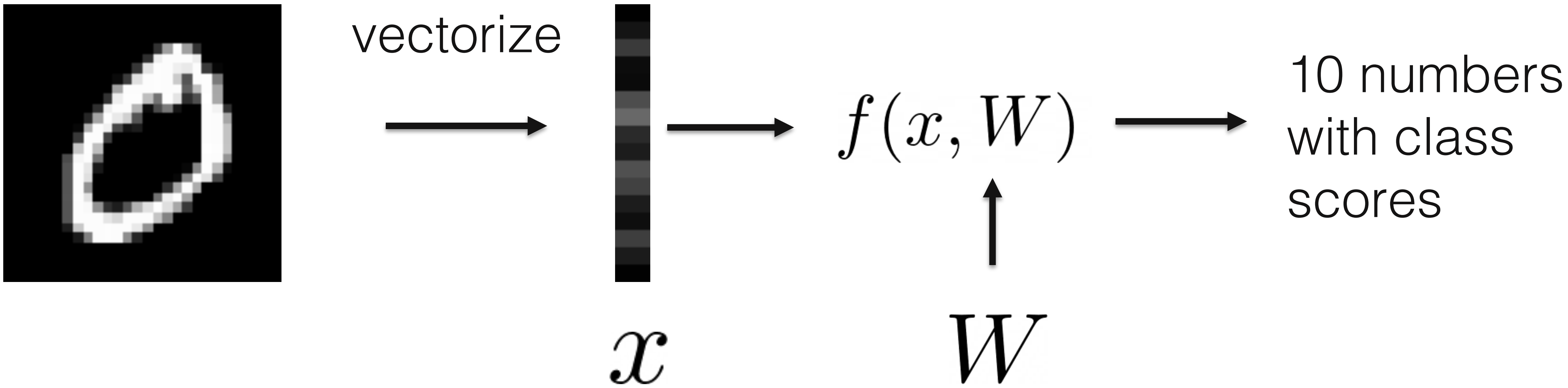
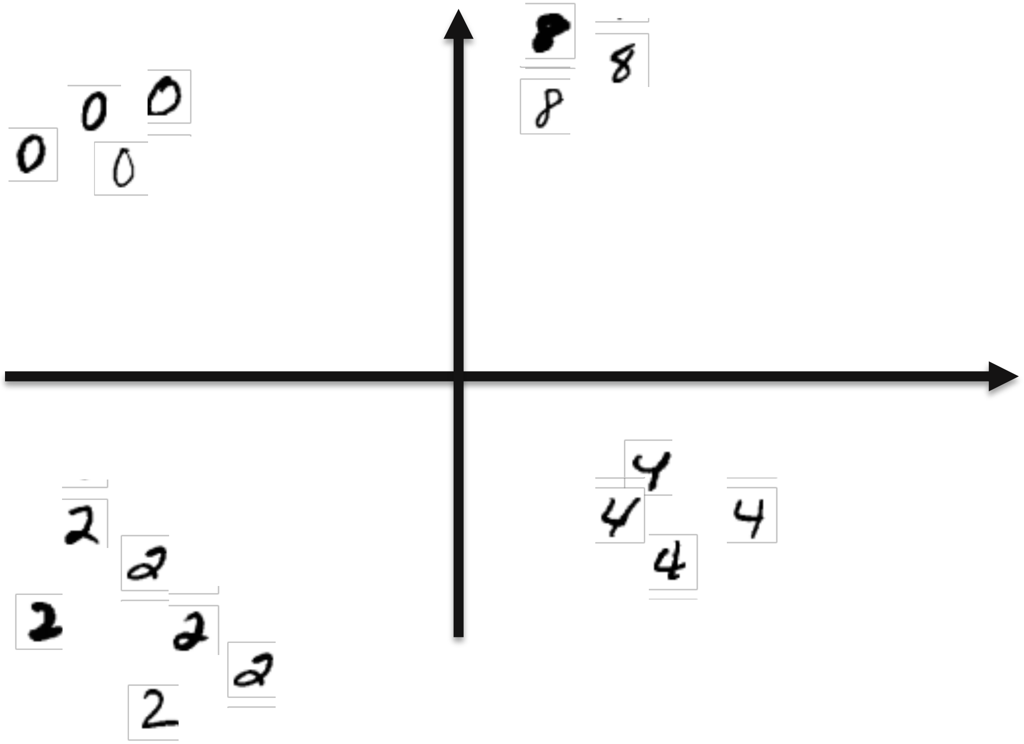


Image Classification

- Linear model: geometric intrepretation

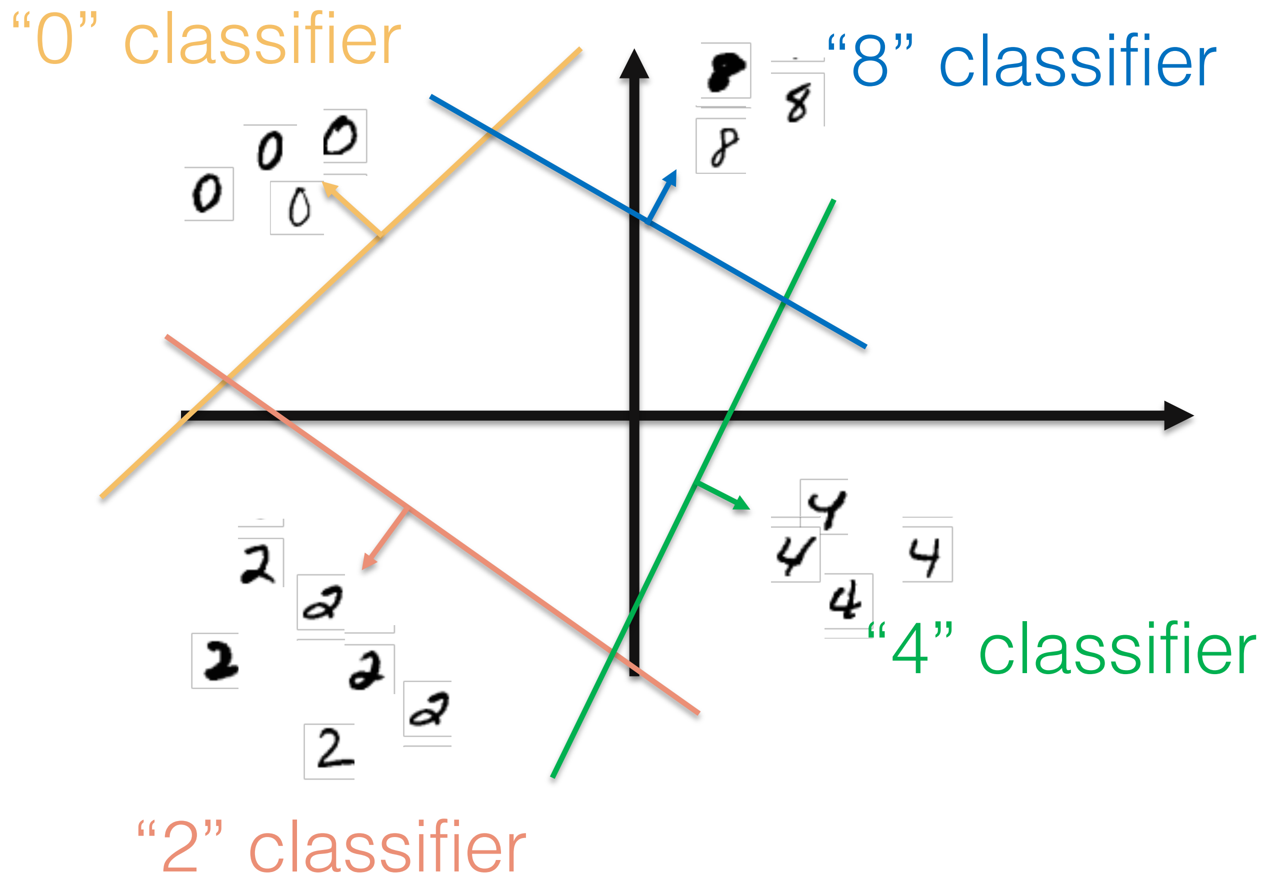


Each image is a point in an N-dimensional space

- N is the number of pixels

Image Classification

- Linear model: geometric interpretation



$$f(x, W) = Wx$$

Computes inner product between rows of W and x!

- Each row of W is a hyperplane
- Sign of inner product tells you which side of the hyperplane
- "separates" the digits

Image Classification

- Linear model (visual interpretation)

Learned filters (rows of W)

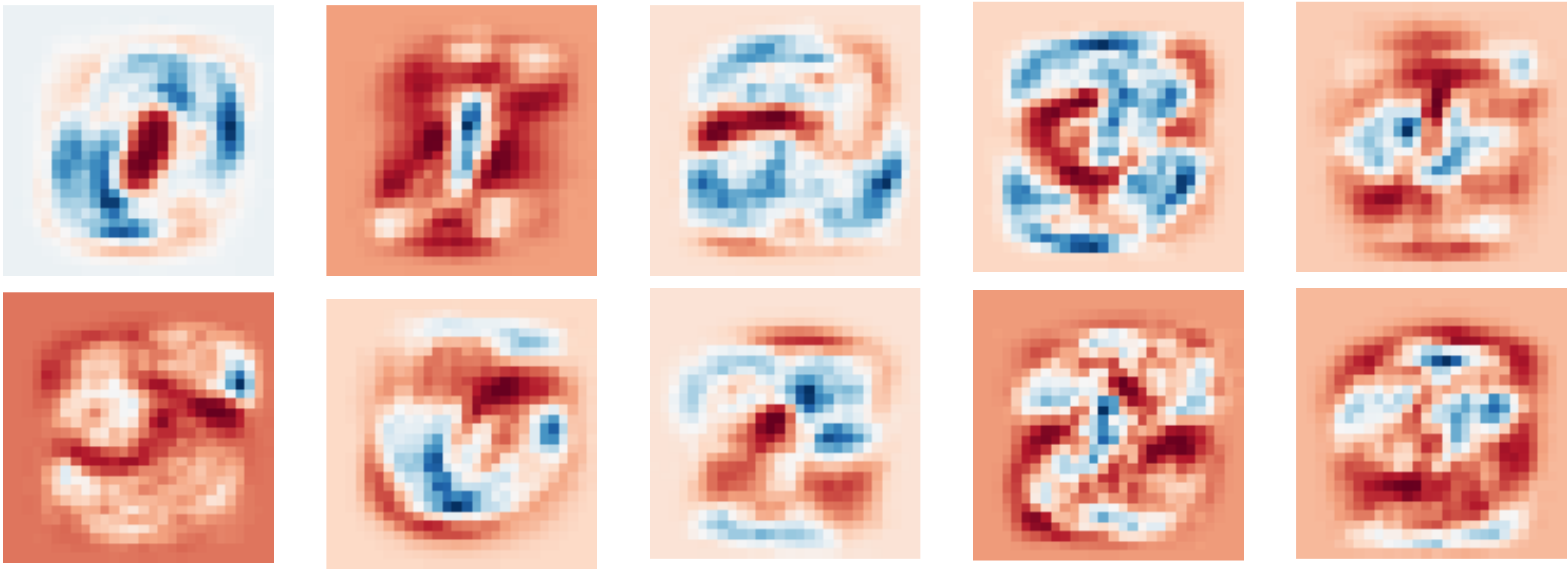
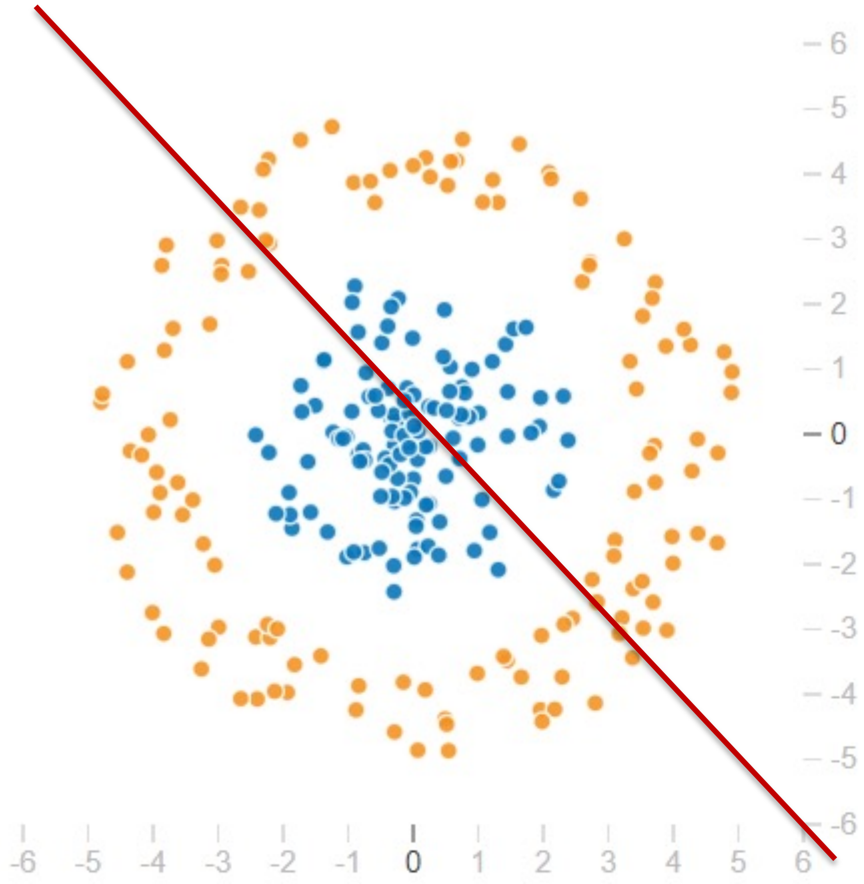


Image Classification

- Limits of linear classifiers

Linear classifiers learn linear decision planes

What if dataset is not linearly separable?



Multilayer Perceptrons (MLPs)

- Linear Model $f = Wx$
- 2-layer MLP $f = W_2 \max(0, W_1 x)$

Multilayer Perceptrons (MLPs)

- Linear Model $f = Wx$
- 2-layer MLP $f = W_2 \max(0, W_1 x)$
- 3-layer MLP $f = W_3 \max(0, W_2 \max(0, W_1 x))$

Multilayer Perceptrons (MLPs)

- Linear Model $f = Wx$
- 2-layer MLP $f = W_2 \max(0, W_1 x)$
- 3-layer MLP $f = W_3 \max(0, W_2 \max(0, W_1 x))$

Non-linearity/activation function between linear layers

Multilayer Perceptrons (MLPs)

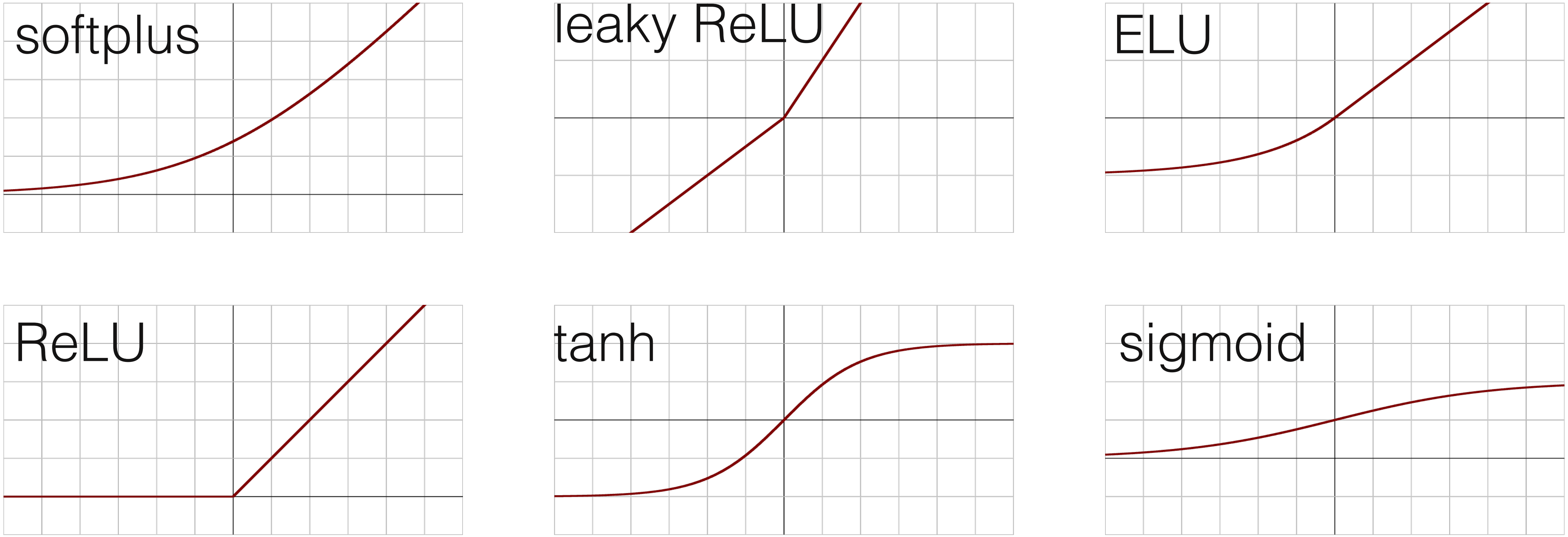
- Linear Model $$f = Wx$$
- 2-layer MLP $$f = W_2 \max(0, W_1 x)$$
- 3-layer MLP $$f = W_3 \max(0, W_2 \max(0, W_1 x))$$

Otherwise we have:

$$f = W_3 W_2 W_1 x$$

Activation Functions

…many to choose from

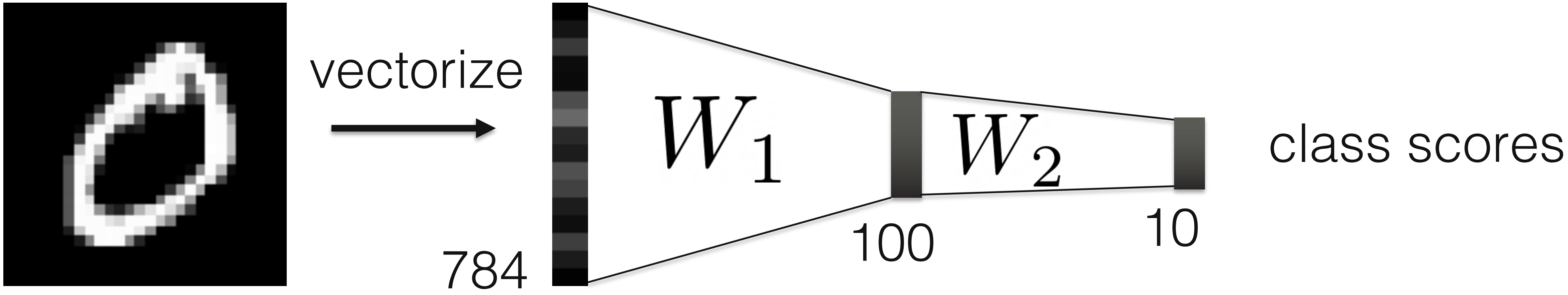


… ReLU is a good general-purpose choice: ReLU(x) = max(0, x)

Multilayer Perceptrons (MLPs)

- Linear Model $f = Wx$
- 2-layer MLP $f = W_2 \max(0, W_1 x)$

Back to our classification example…

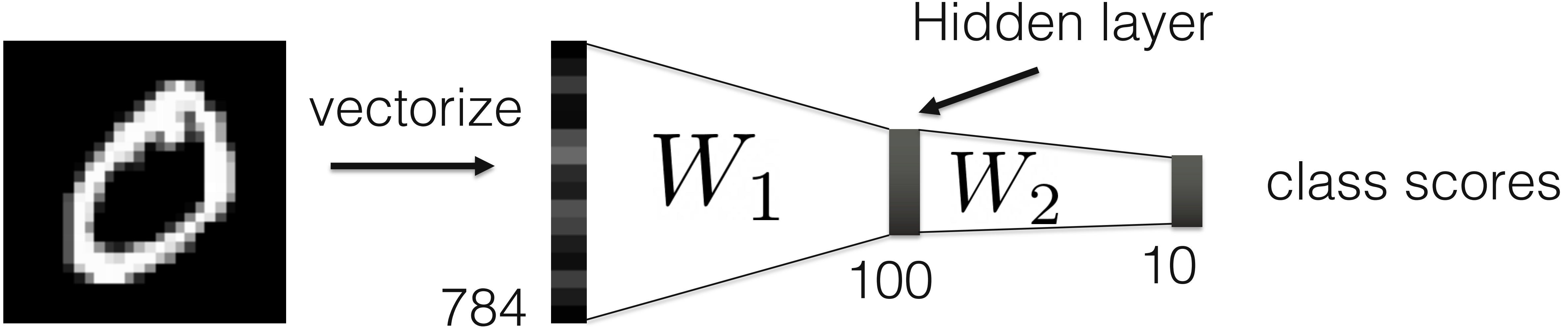


$$x \in \mathbb{R}^D,\ W_1 \in \mathbb{R}^{H \times D},\ W_2 \in \mathbb{R}^{C \times H}$$

Multilayer Perceptrons (MLPs)

- Linear Model $f = Wx$
- 2-layer MLP $f = W_2 \max(0, W_1 x)$

Back to our classification example…

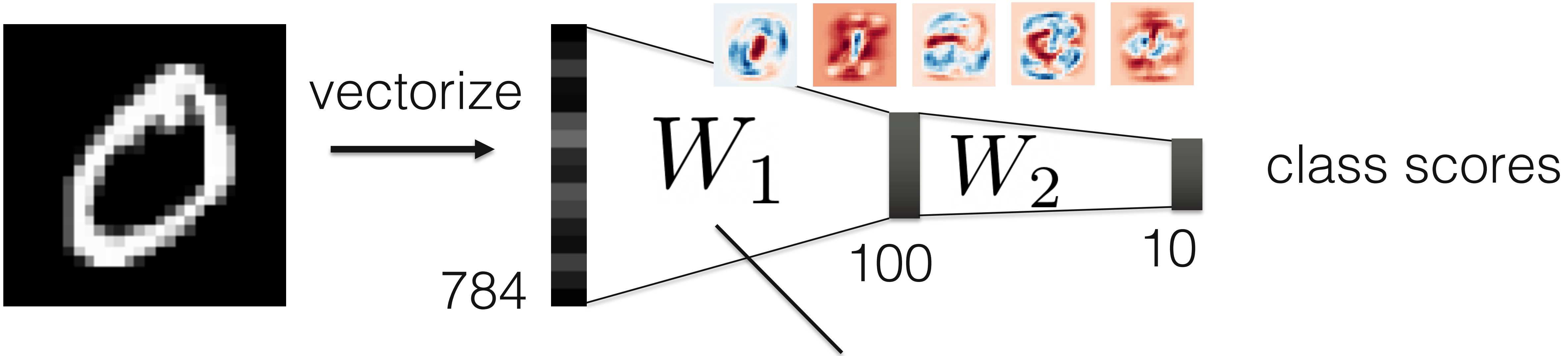


$$x \in \mathbb{R}^D, W_1 \in \mathbb{R}^{H \times D}, W_2 \in \mathbb{R}^{C \times H}$$

Multilayer Perceptrons (MLPs)

- Linear Model $f = Wx$
- 2-layer MLP $f = W_2 \max(0, W_1 x)$

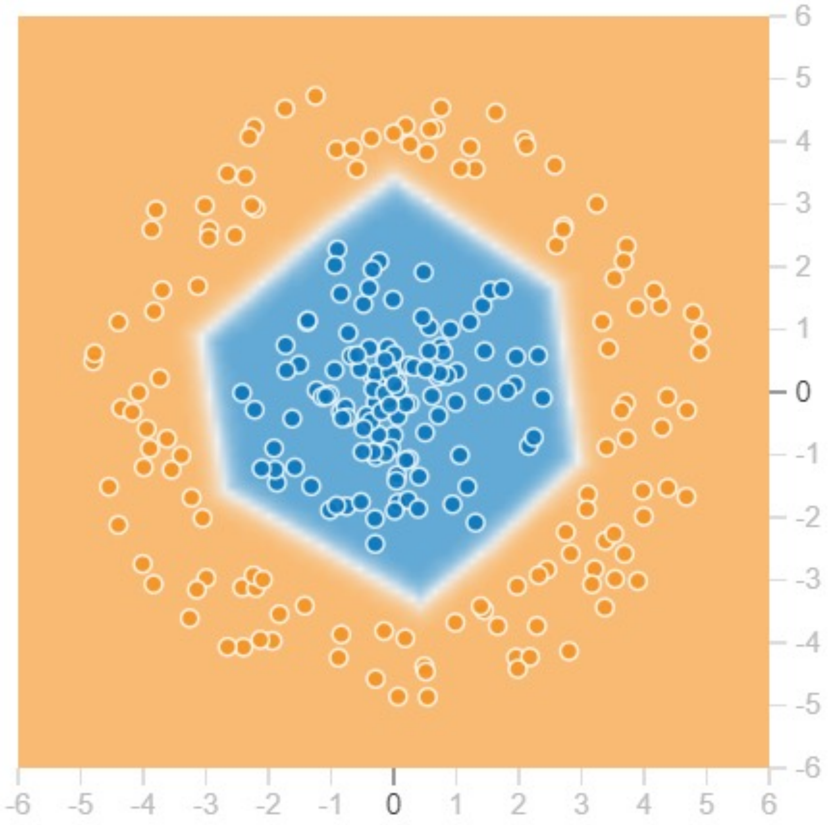
Back to our classification example…



Now we have 100 shape templates, shared between classes

Multilayer Perceptrons (MLPs)

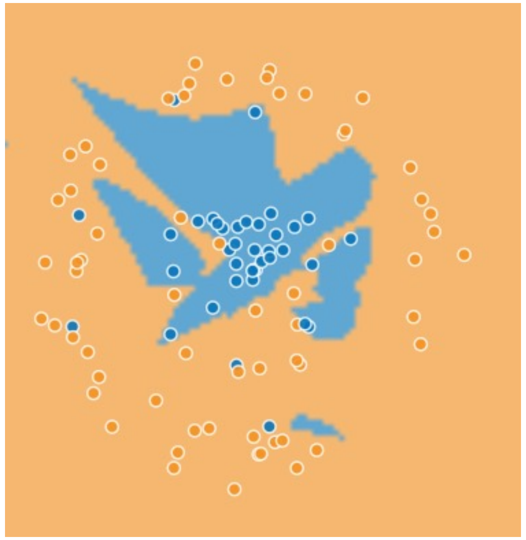
- Overcomes limits of linear classifiers
  - Can learn non-linear decision boundaries
  - Complexity scales with the number of neurons/hidden layers



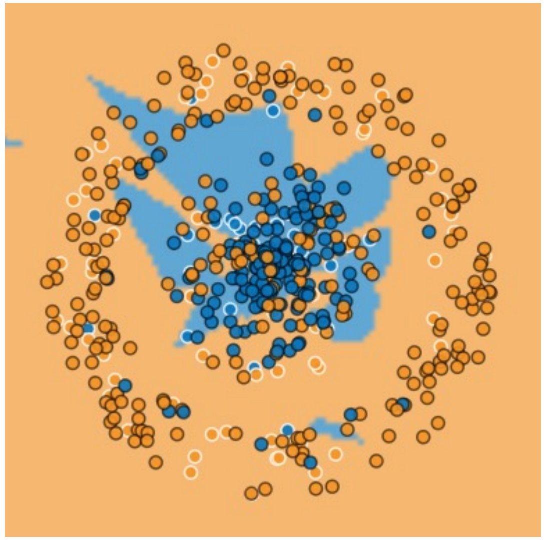
Multilayer Perceptrons (MLPs)

- More parameters is not always better!
  - Can lead to overfitting the training data
  - Performance on test data is worse

train

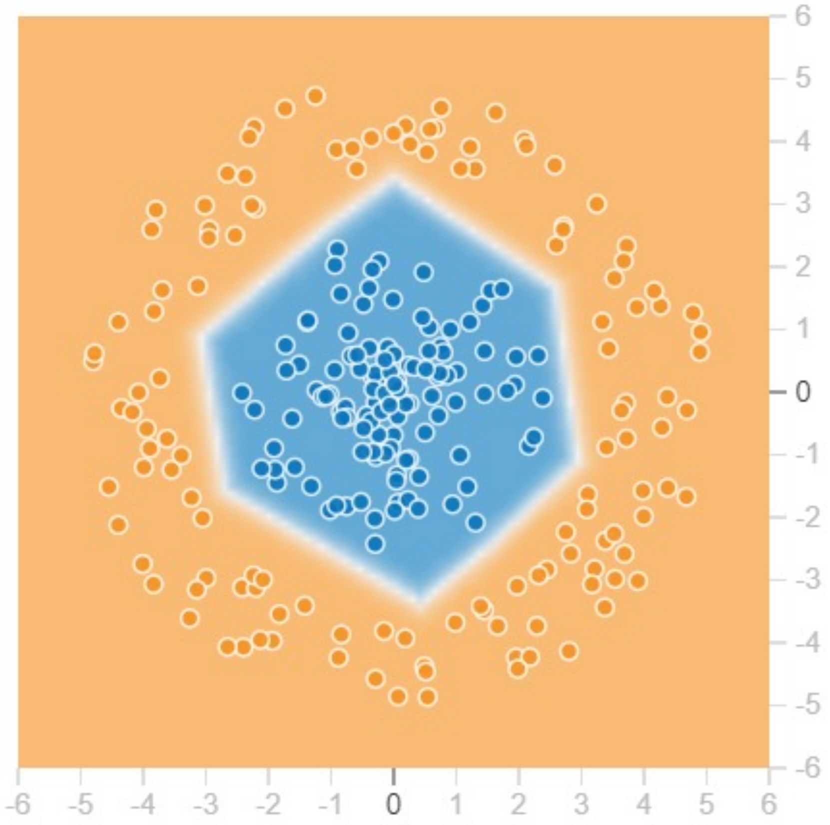


test

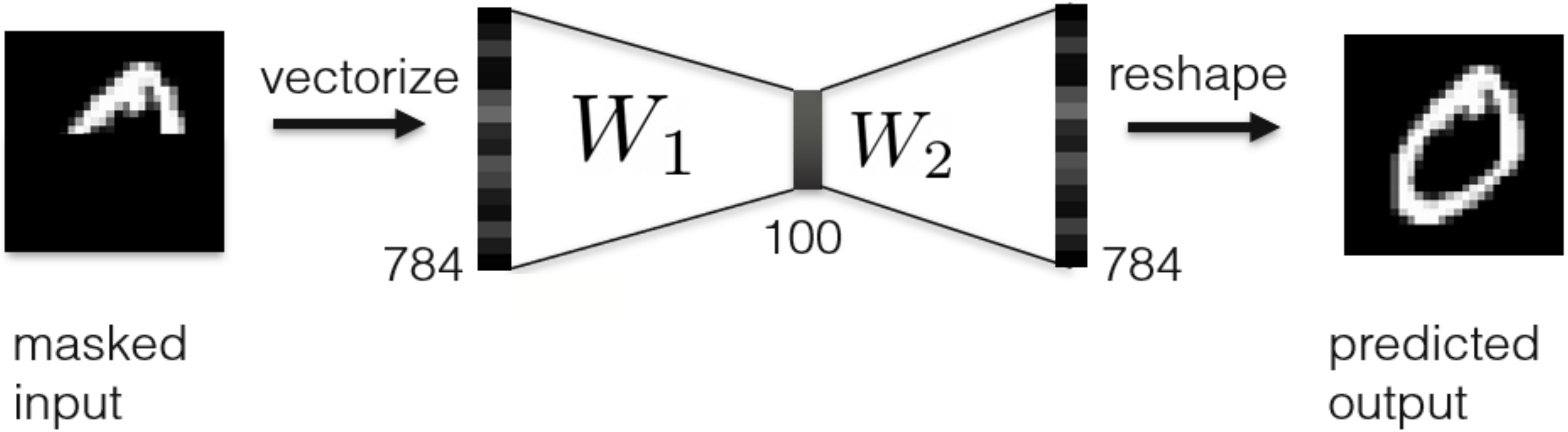


Multilayer Perceptrons (MLPs)

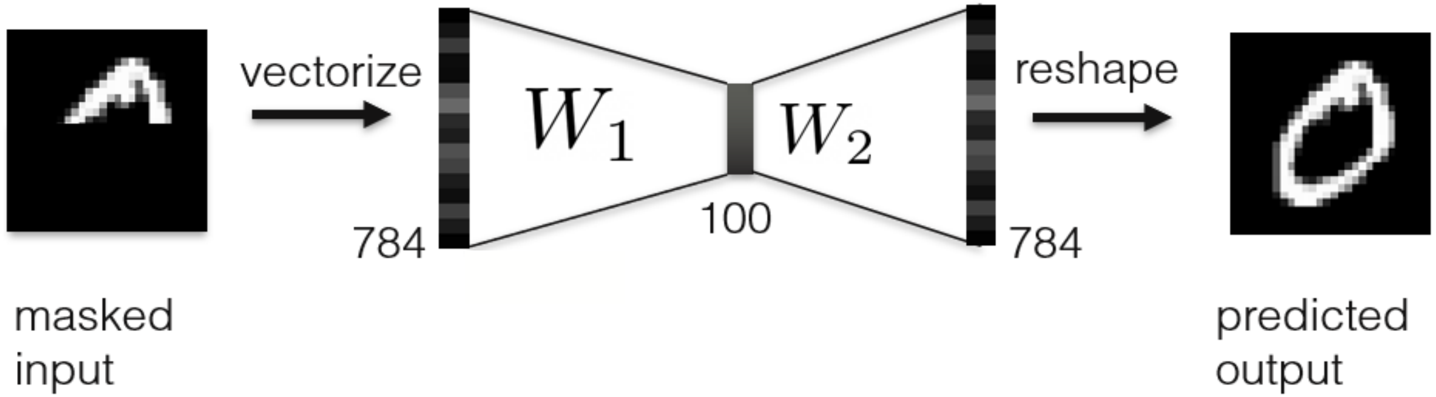
- More on classification…
  - https://cs231n.github.io/linear-classify/
  - https://csc413-uoft.github.io/

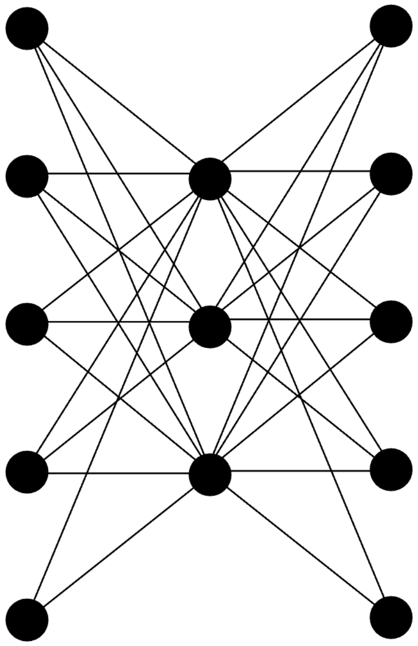


Why “neural” network?

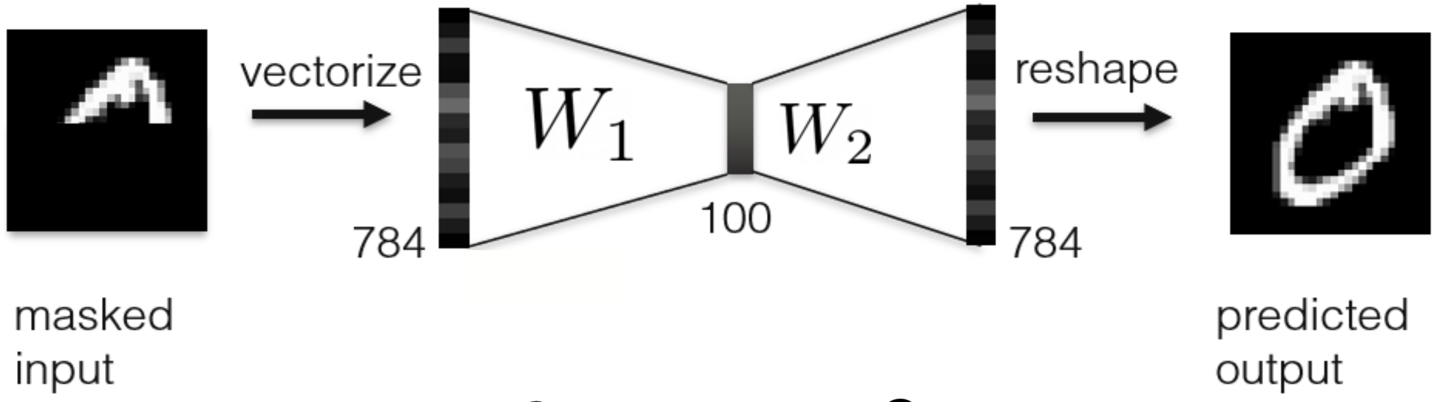


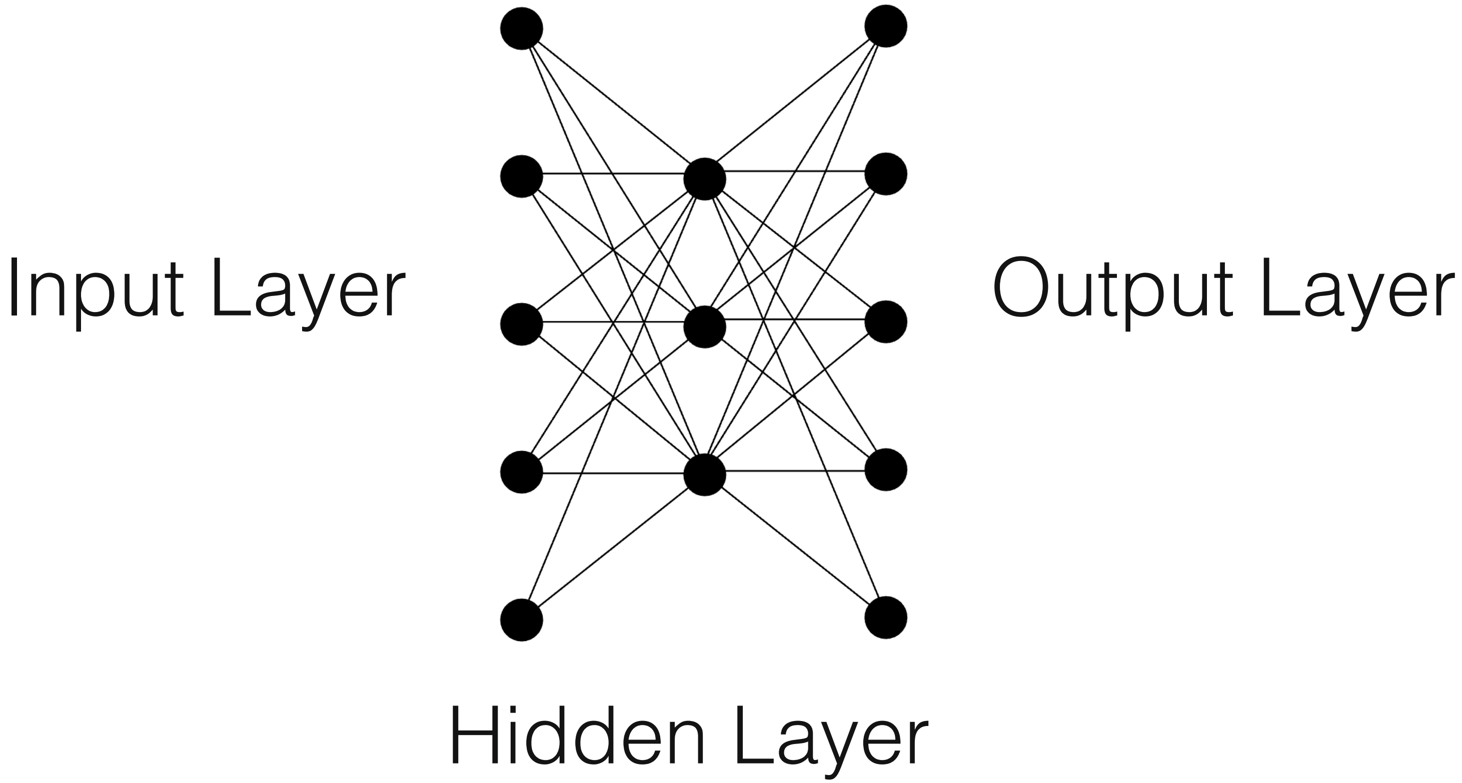
Why “neural” network?



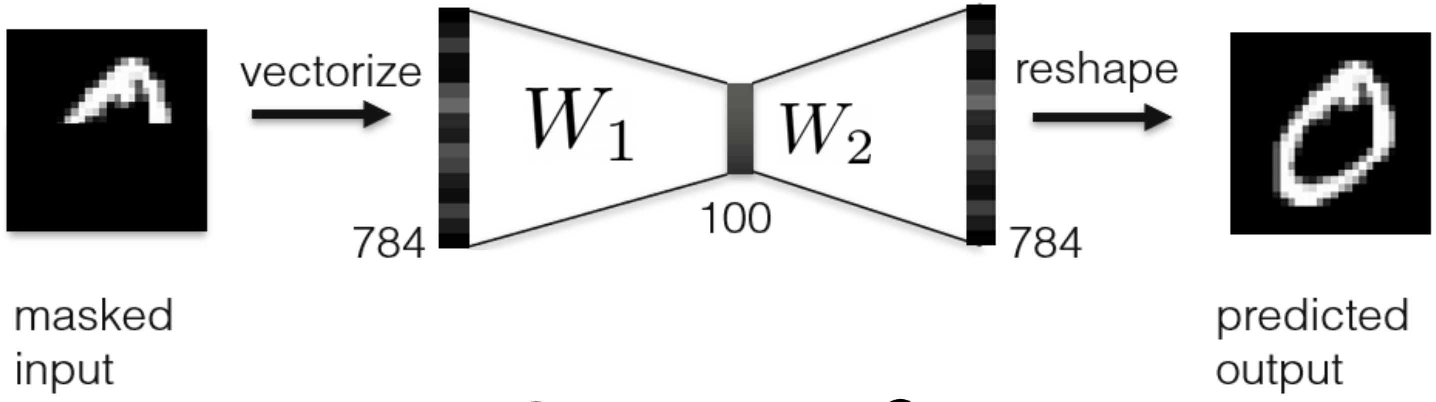


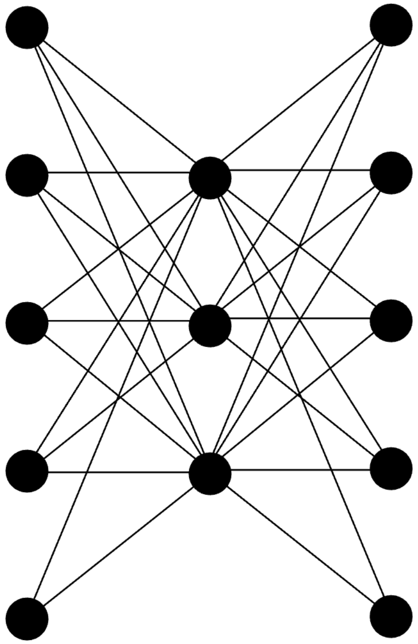
Why "neural" network?





Why “neural” network?





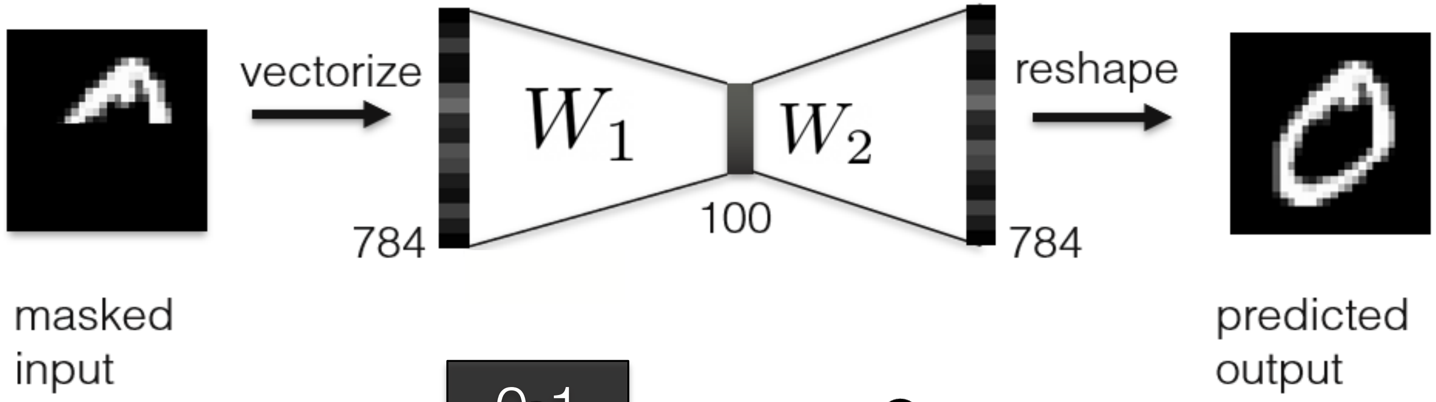
Input Layer

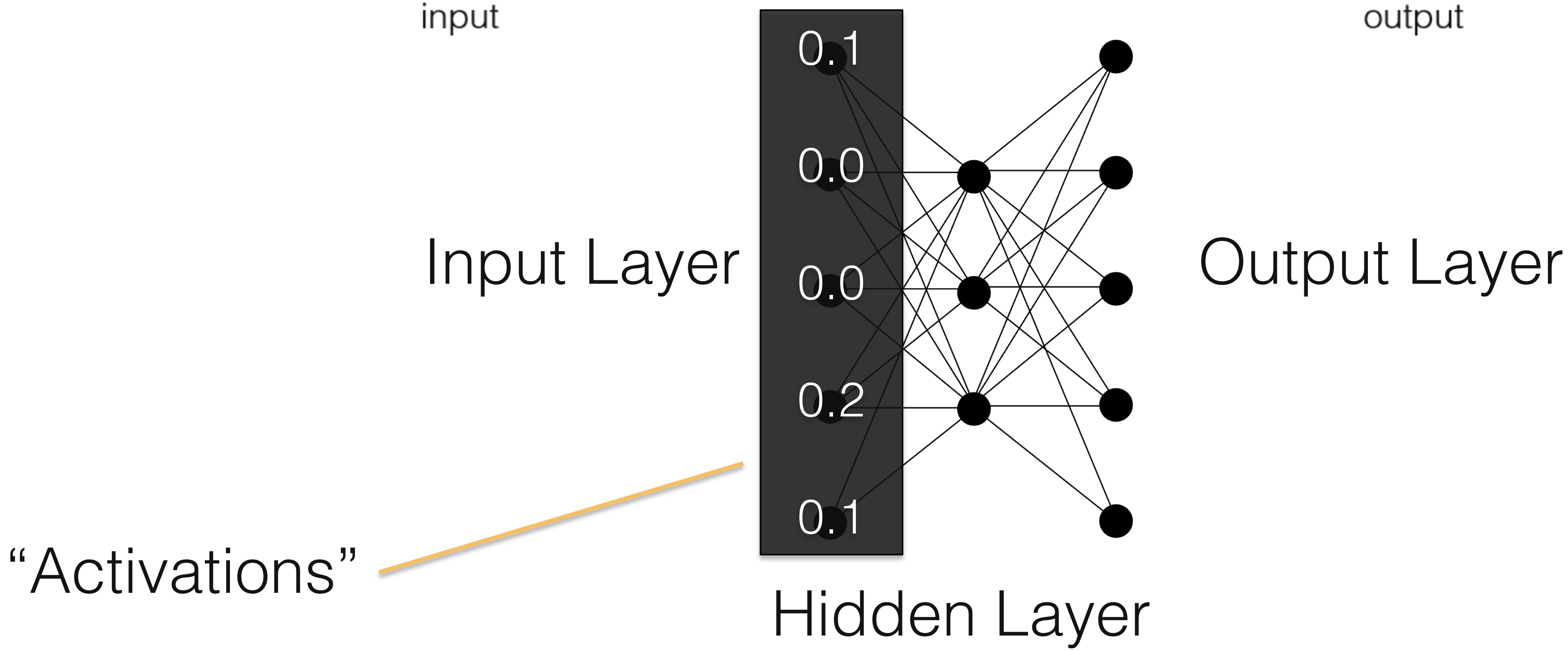
Output Layer

“Neuron”

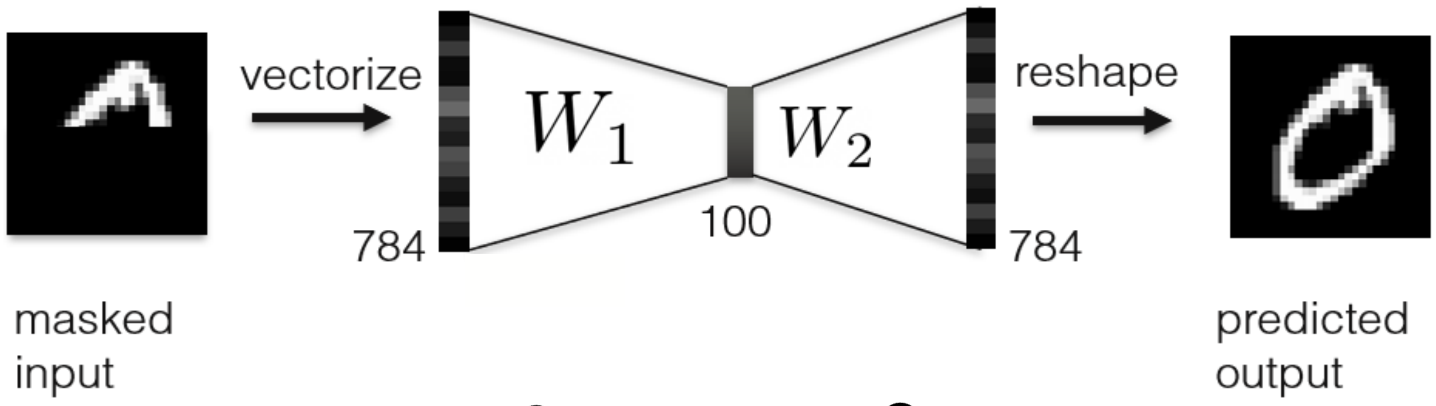
Hidden Layer

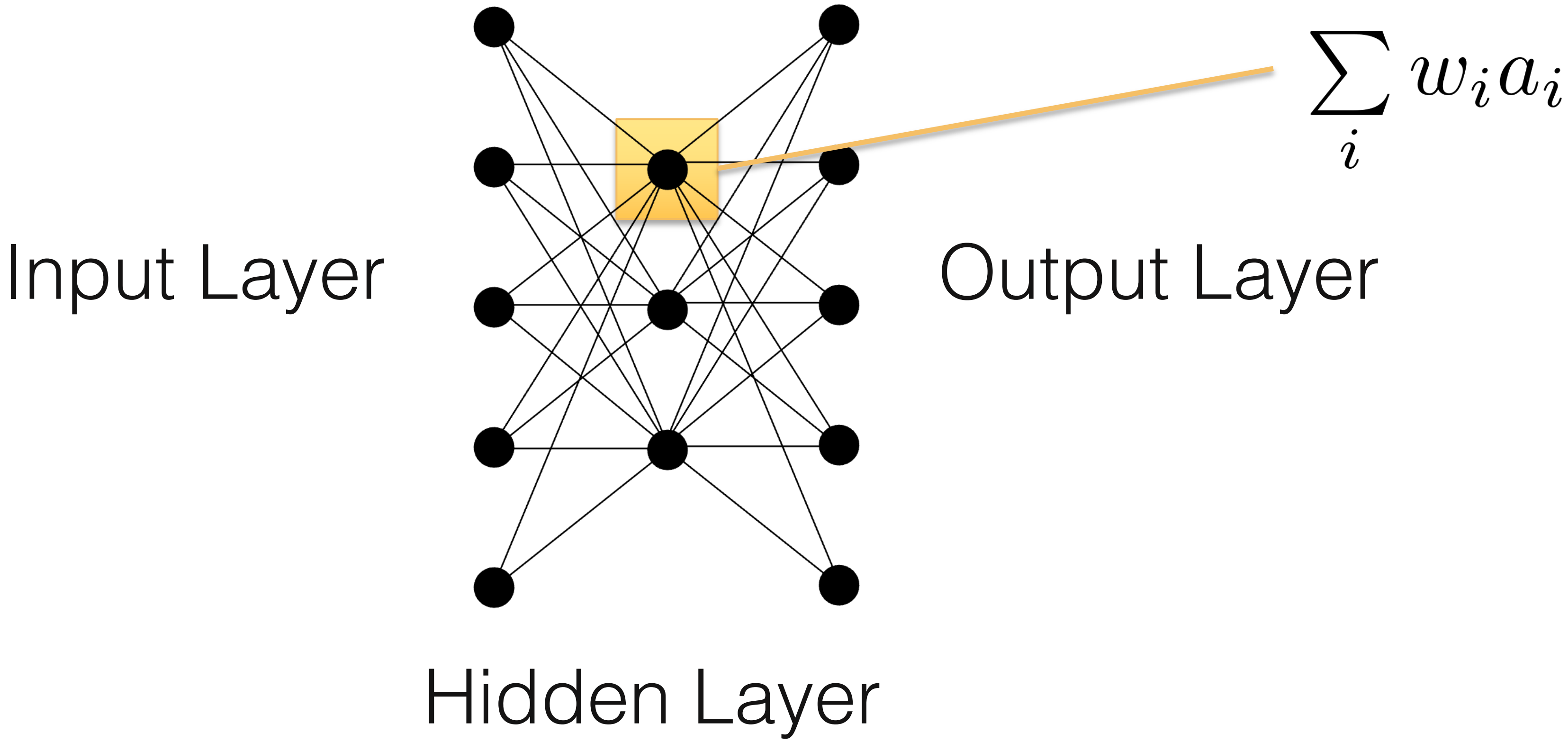
Why “neural” network?

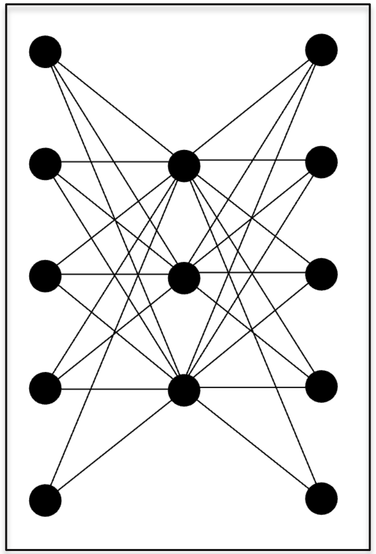


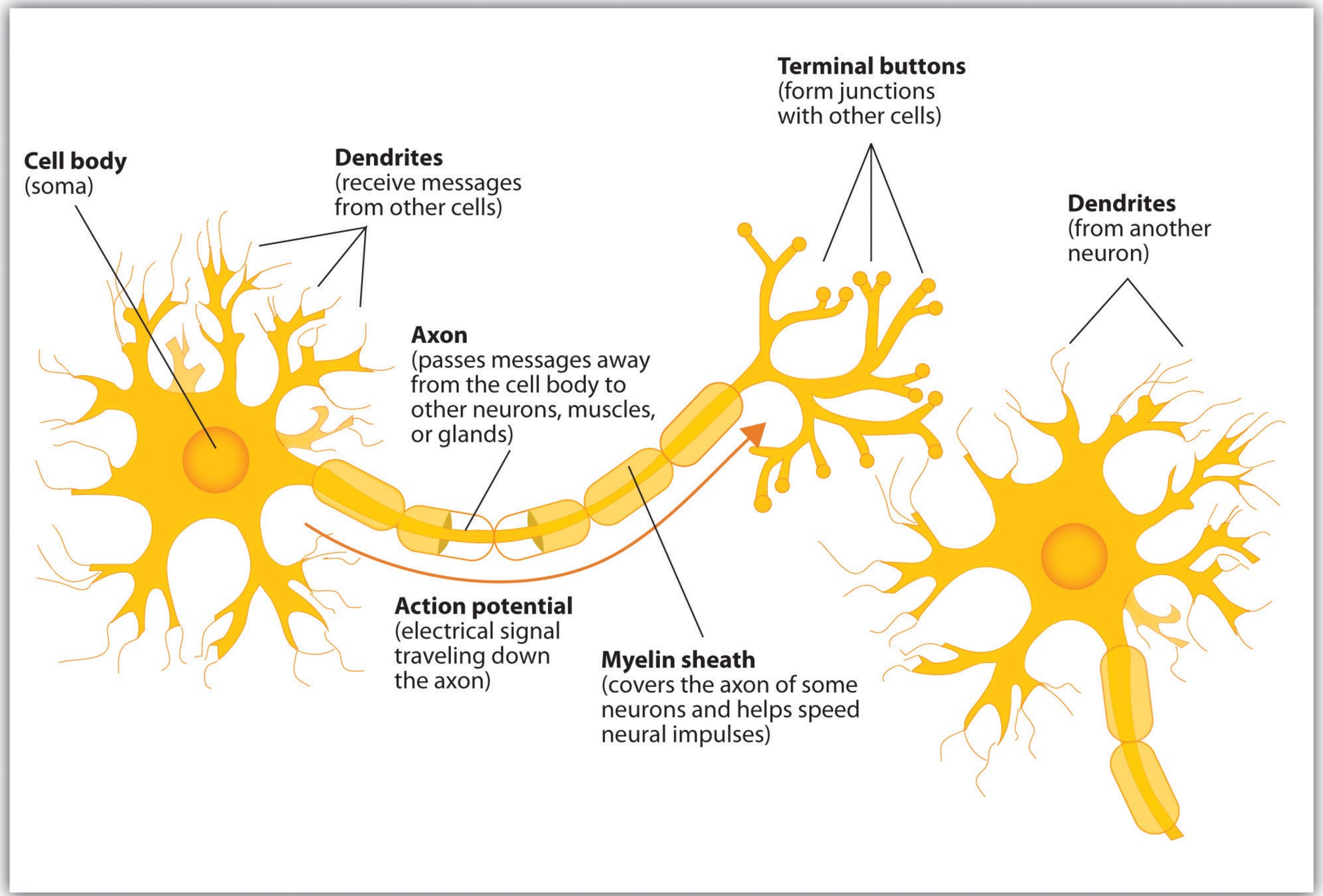


Why “neural” network?









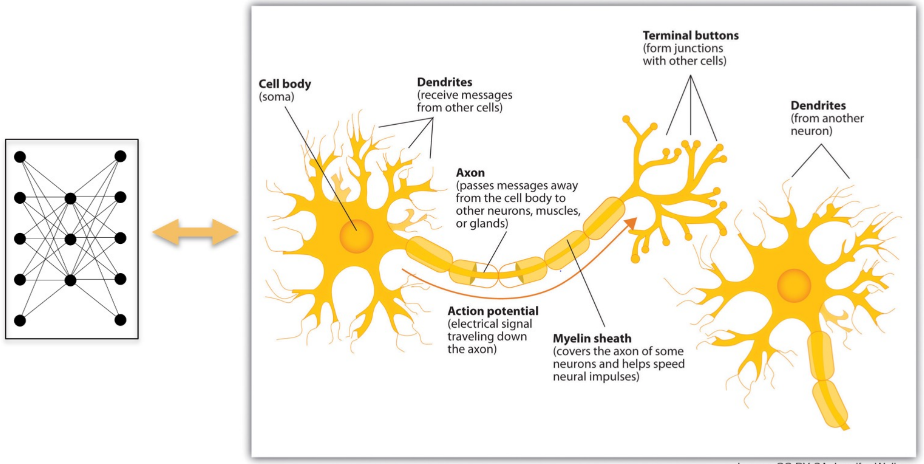
Cell body
(soma)
Dendrites
(receive messages
from other cells)
Terminal buttons
(form junctions
with other cells)
Dendrites
(from another
neuron)
Axon
(passes messages away
from the cell body to
other neurons, muscles,
or glands)
Action potential
(electrical signal
traveling down
the axon)
Myelin sheath
(covers the axon of some
neurons and helps speed
neural impulses)

Loose analogy!

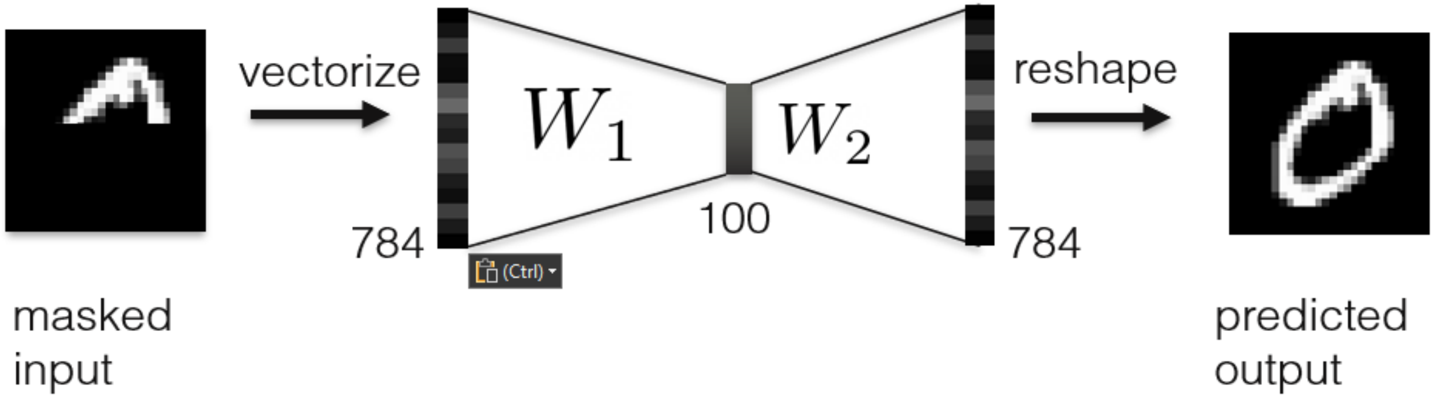
- Neurons have activation potentials, all-or-none firing behavior

- Interconnectivity between actual neurons is dense and complicated

- Connection between neurons is complex non-linear dynamical system



Drawbacks of fully-connected networks



- spatial structure is destroyed
- fully-connected weights do not scale

Overview

- Motivation
- Fully-connected Networks
- Convolutional Neural Networks
- Training networks

Convolutional Neural Networks

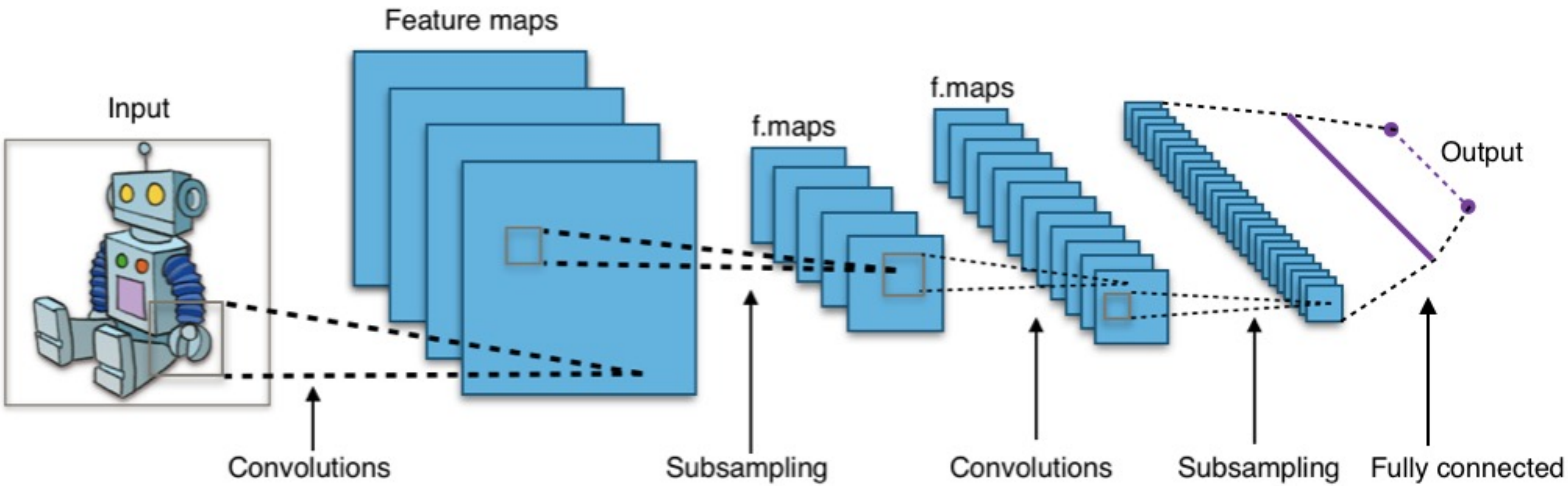


Image: CC Aphex34

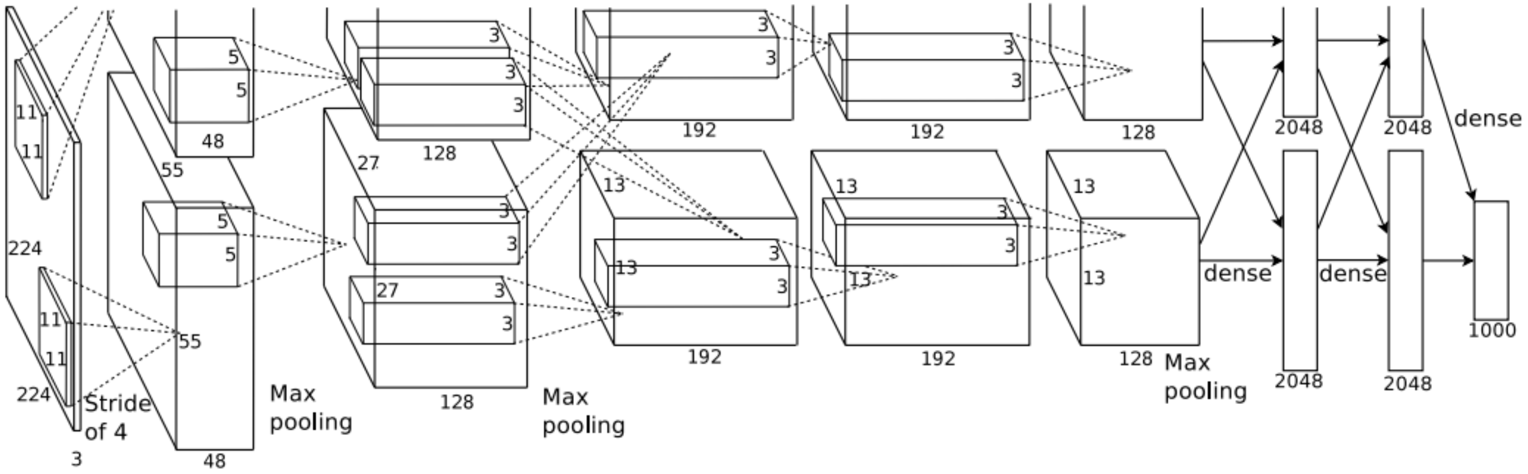
- Exploit spatial structure
- Scale to large inputs with fewer parameters
- Remarkable performance for processing visual data

AlexNet & surge in popularity

2010: ImageNet Large Scale Visual Recognition Challenge

- 14 million labeled images

First convolutional network for image classification

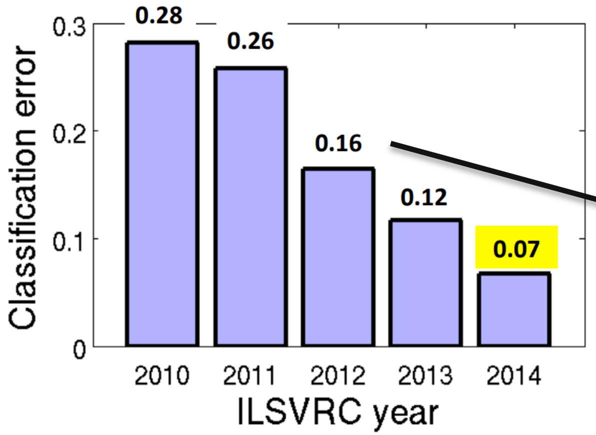


AlexNet [Krizhevsky ‘12]

AlexNet & surge in popularity

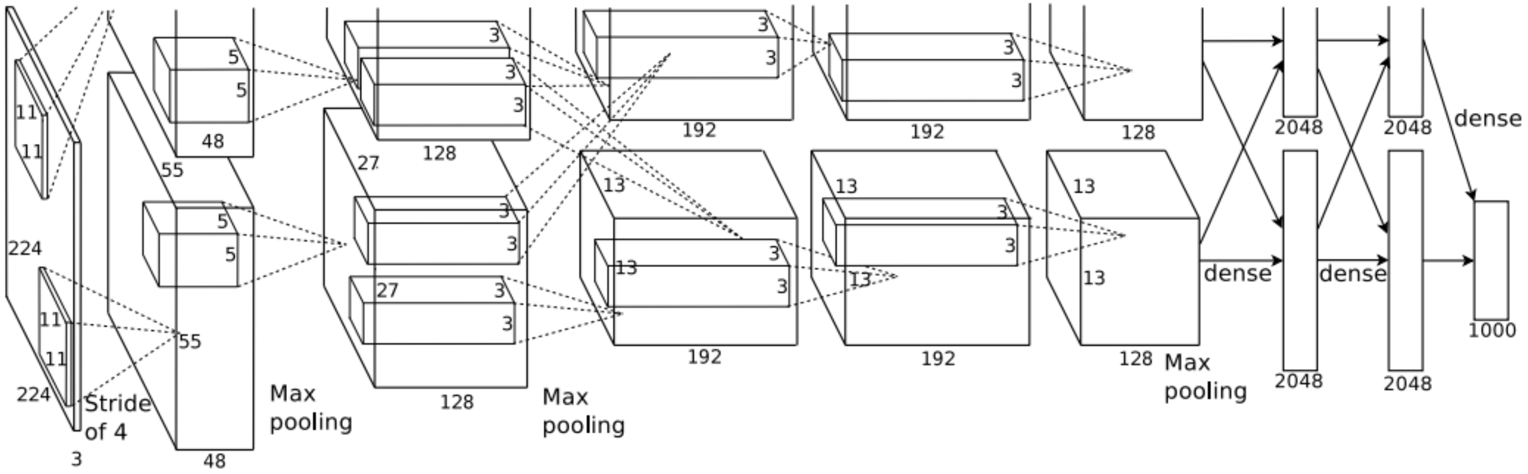
2010: ImageNet Large Scale Visual Recognition Challenge
- 14 million labeled images

First convolutional network for image classification



[Russakovsky '15]

CNNs

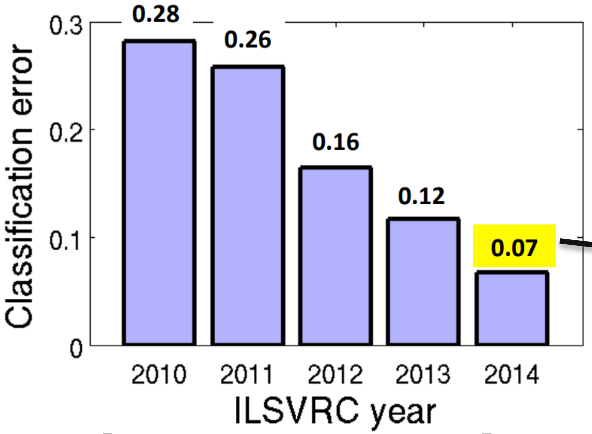


AlexNet [Krizhevsky '12]

AlexNet & surge in popularity

2010: ImageNet Large Scale Visual Recognition Challenge

- 14 million labeled images



[Russakovsky '15]

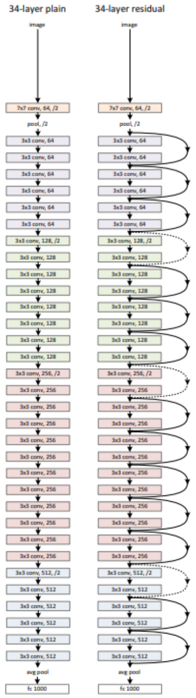
Deep networks

VGG

[Simonyan '14]

GoogLeNet

[Szegedy '14]

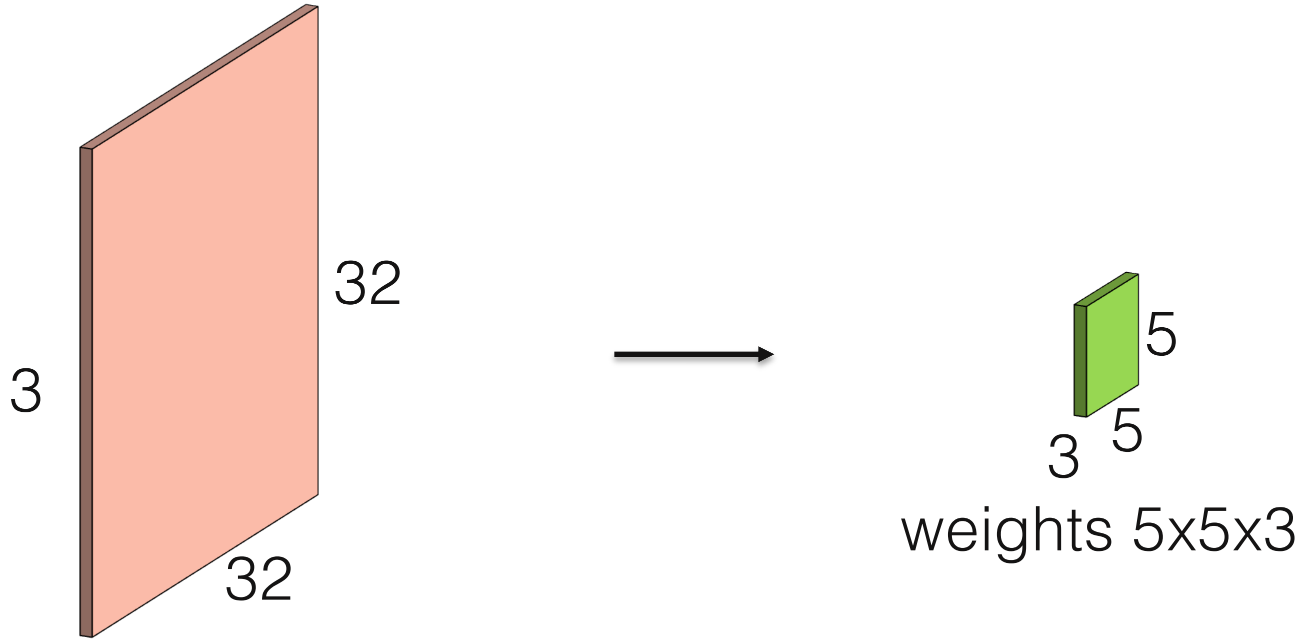


ResNet

[He '15]

Fully-Connected Layer

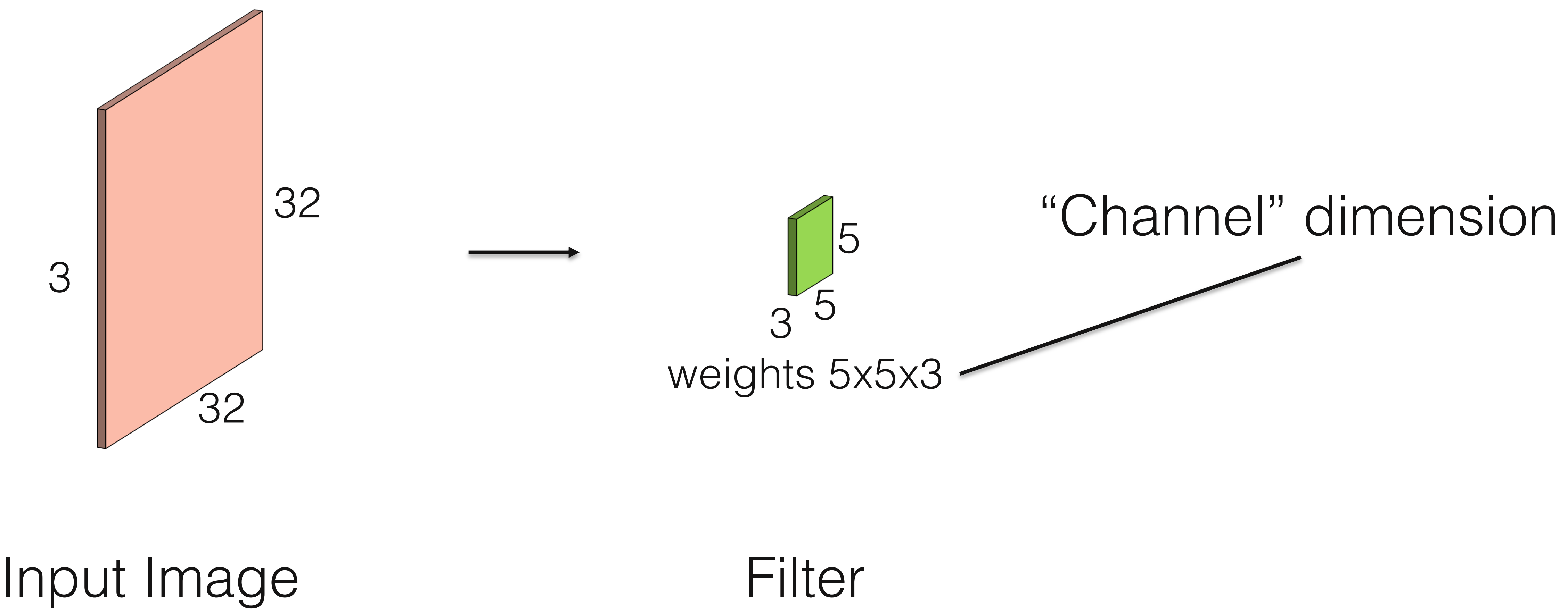
Convolutional Layer



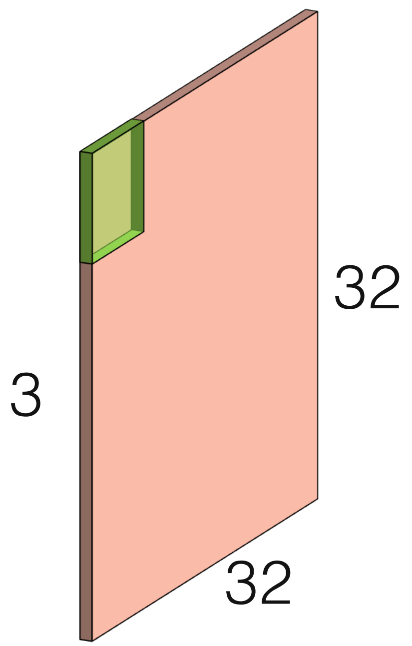
Input Image

Filter

Convolutional Layer

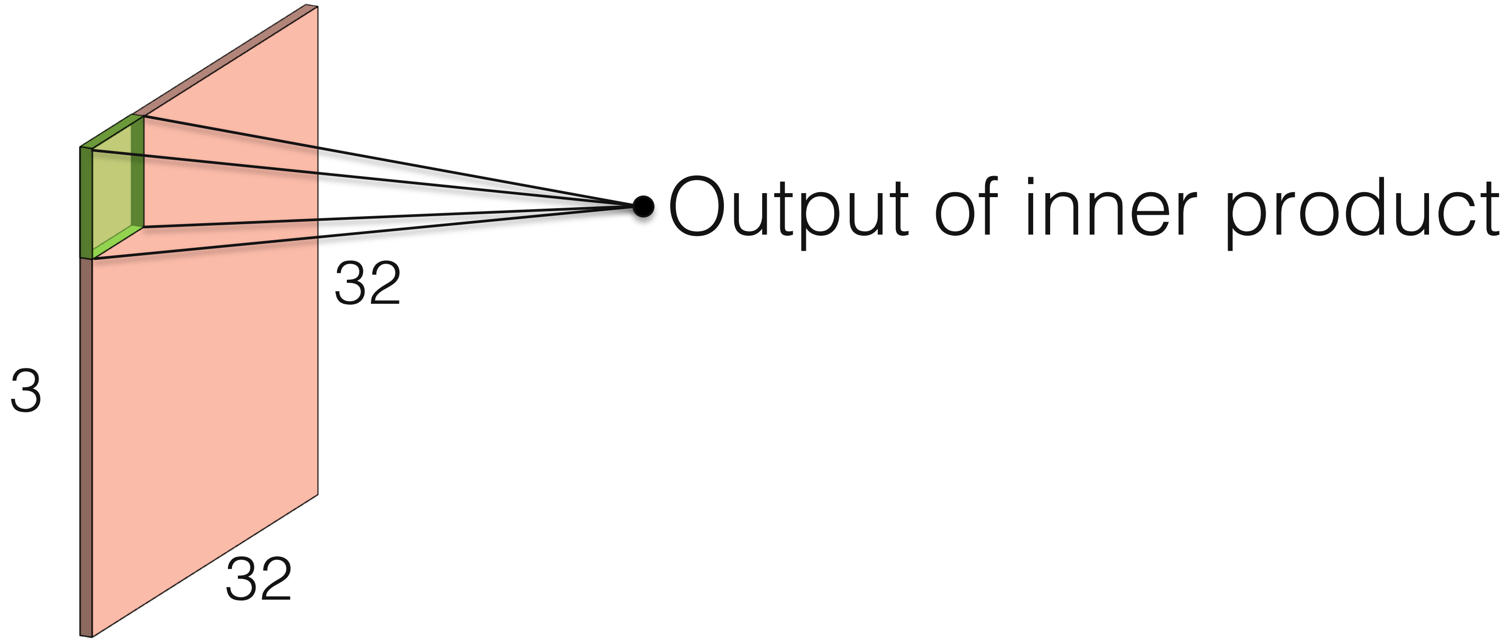


Convolutional Layer



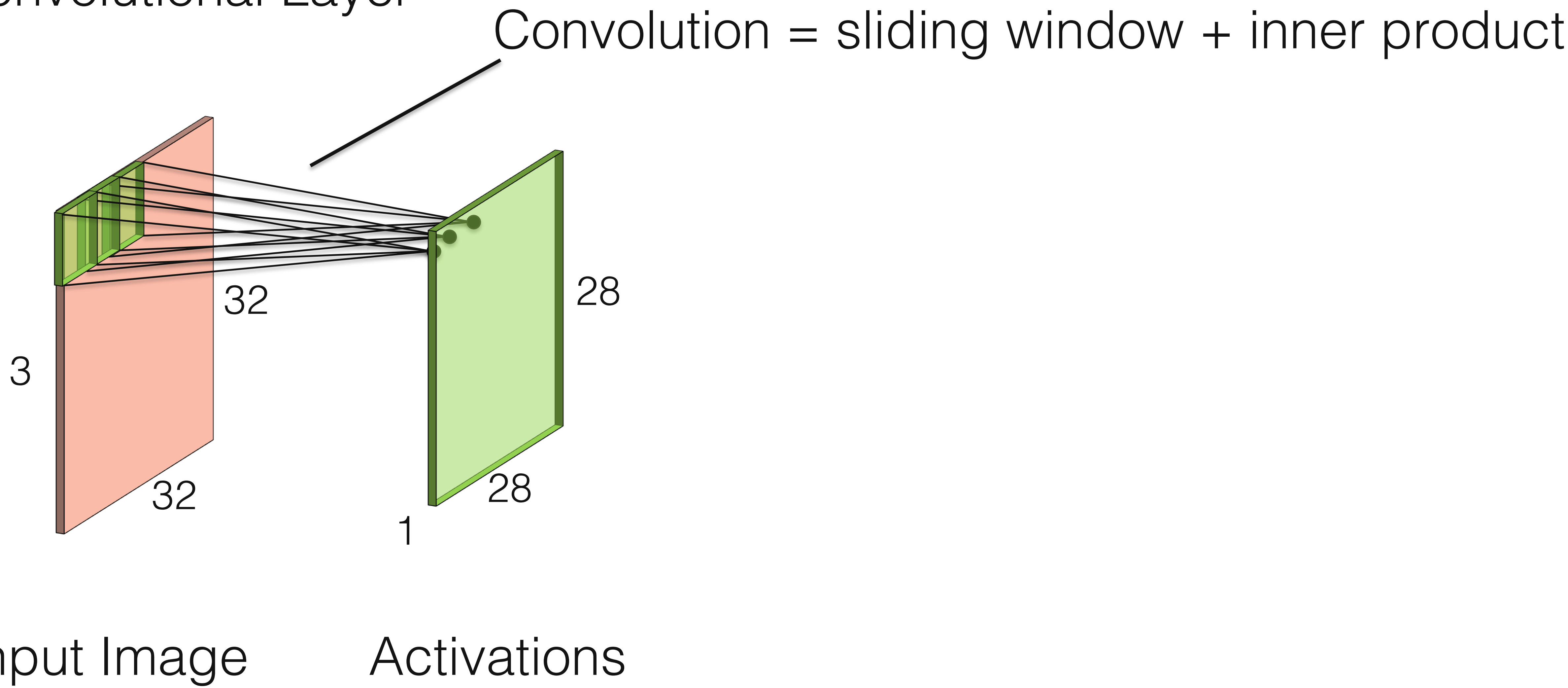
Input Image

Convolutional Layer

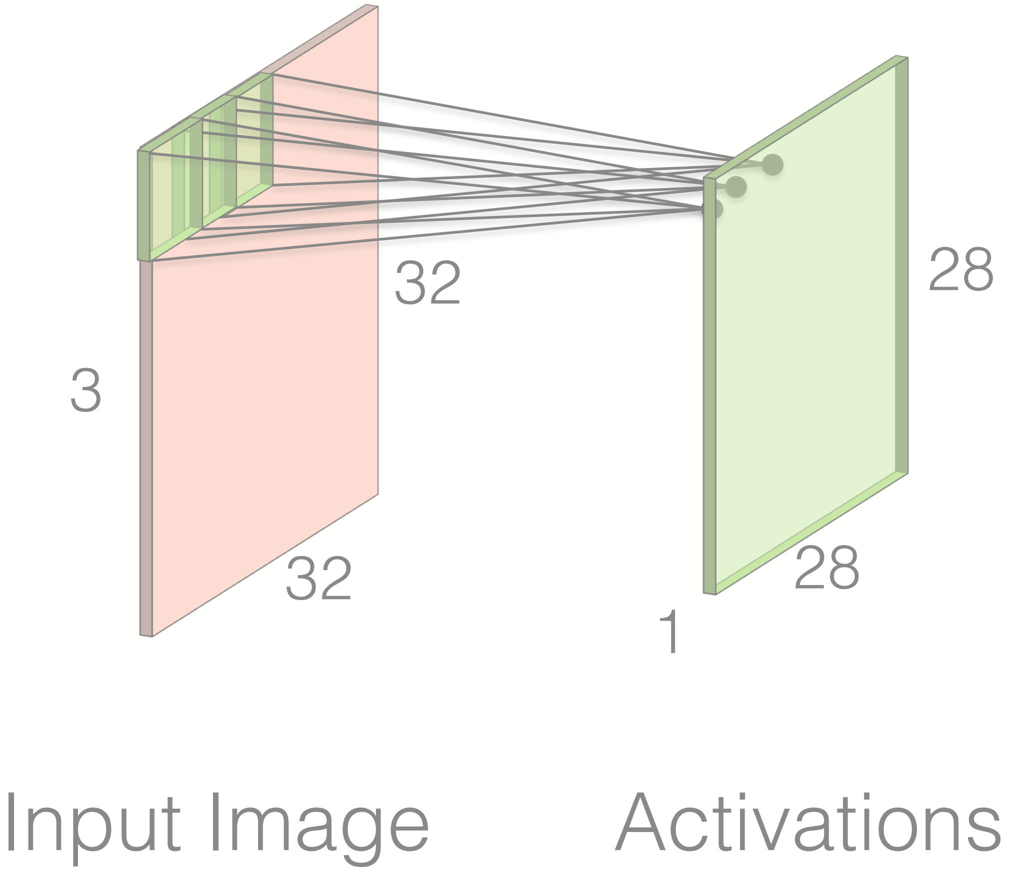


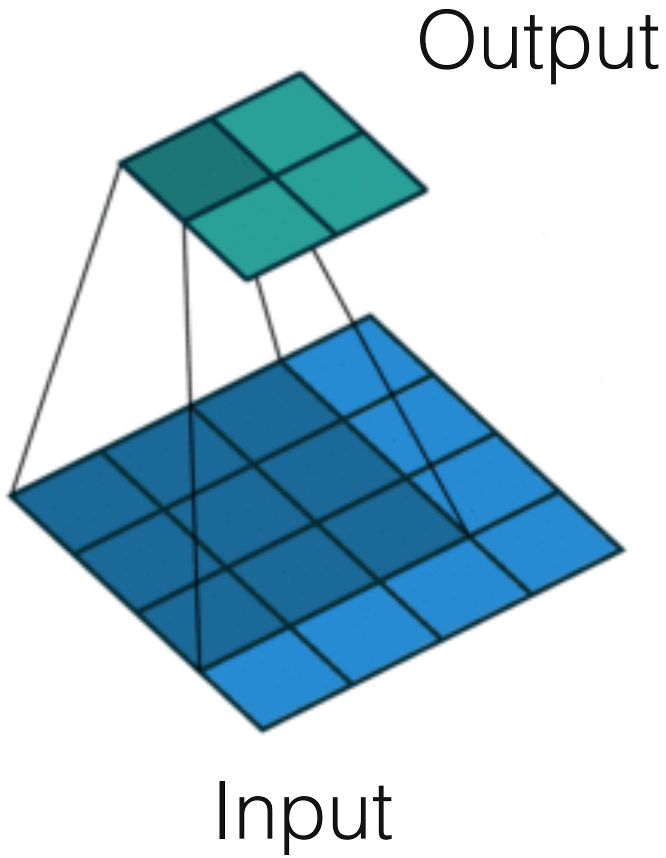
Input Image

Convolutional Layer



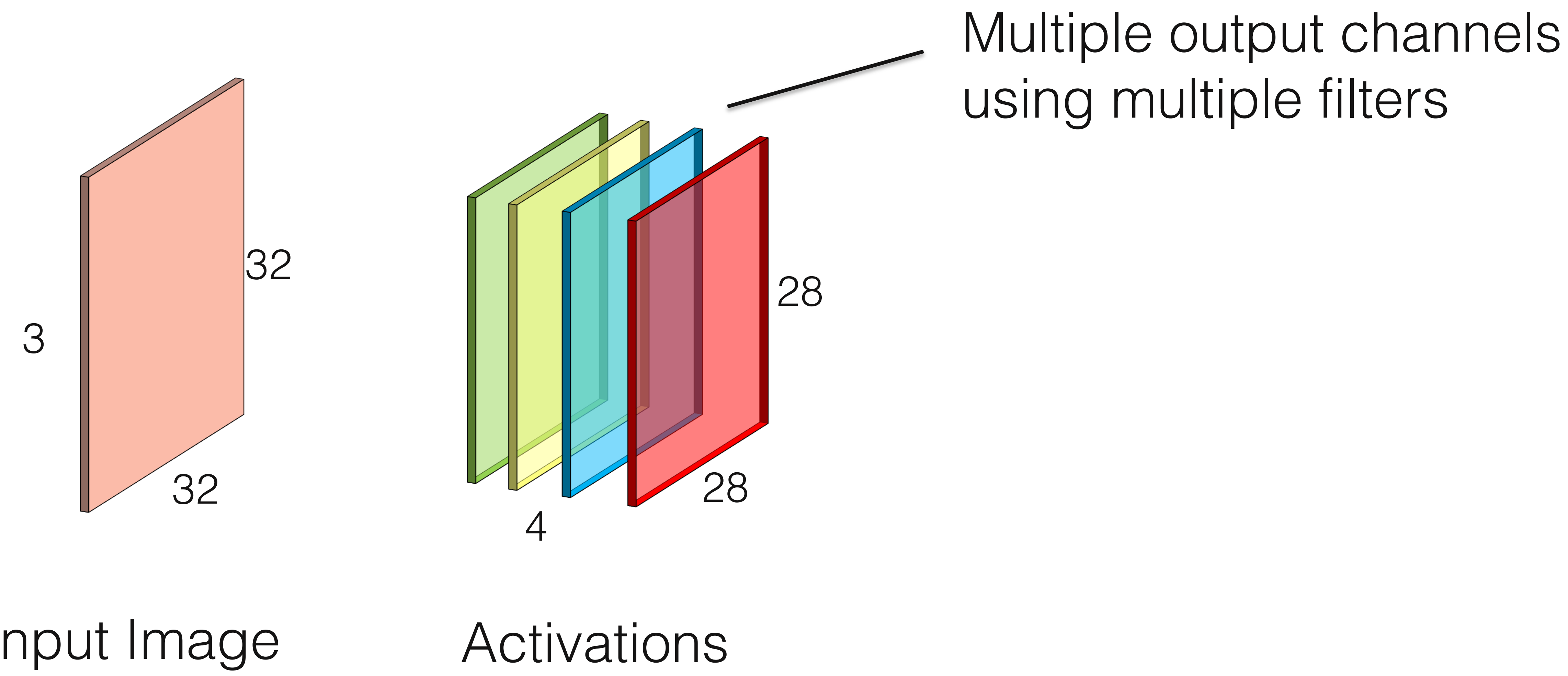
Convolutional Layer



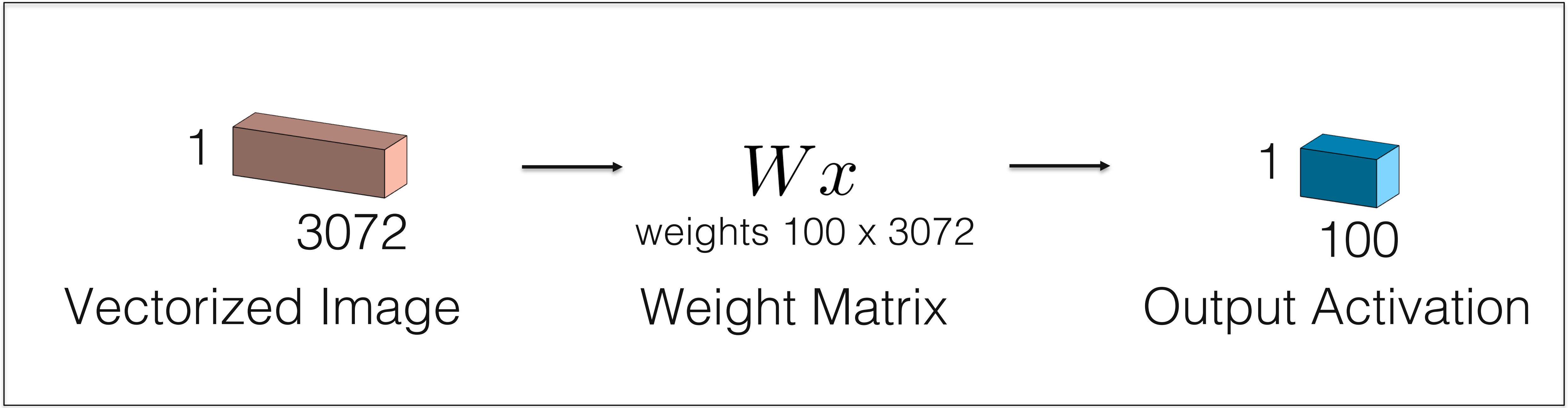


https://github.com/vdumoulin/conv_arithmetic

Convolutional Layer

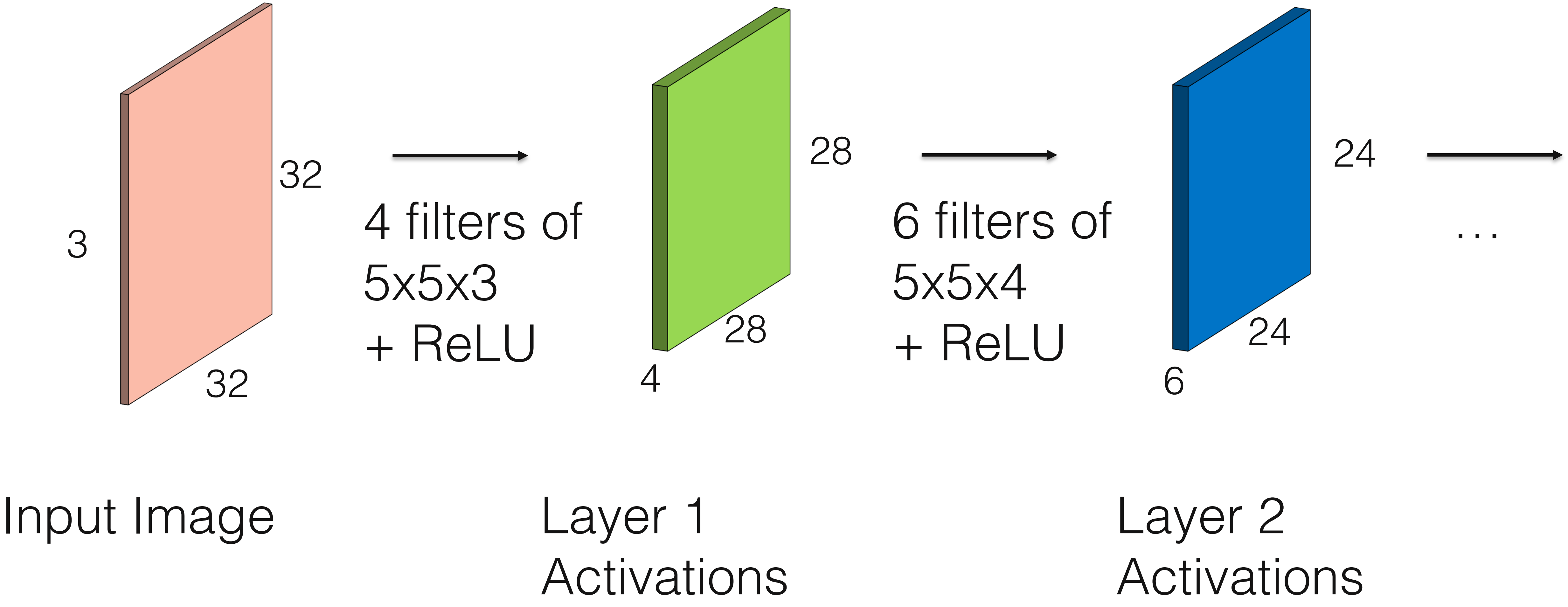


Fully-Connected Layer



Special case of convolutional layer when filter size = input size!

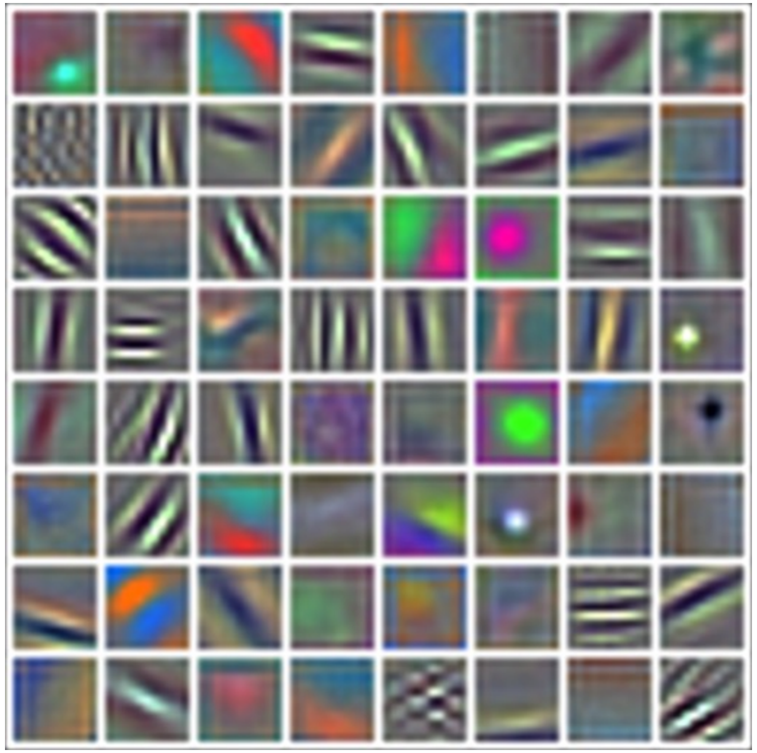
Convolutional Neural Network



Case Study: AlexNet

Input Image

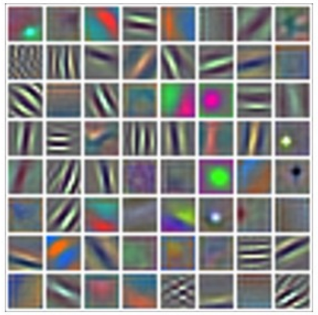
First-layer Filters

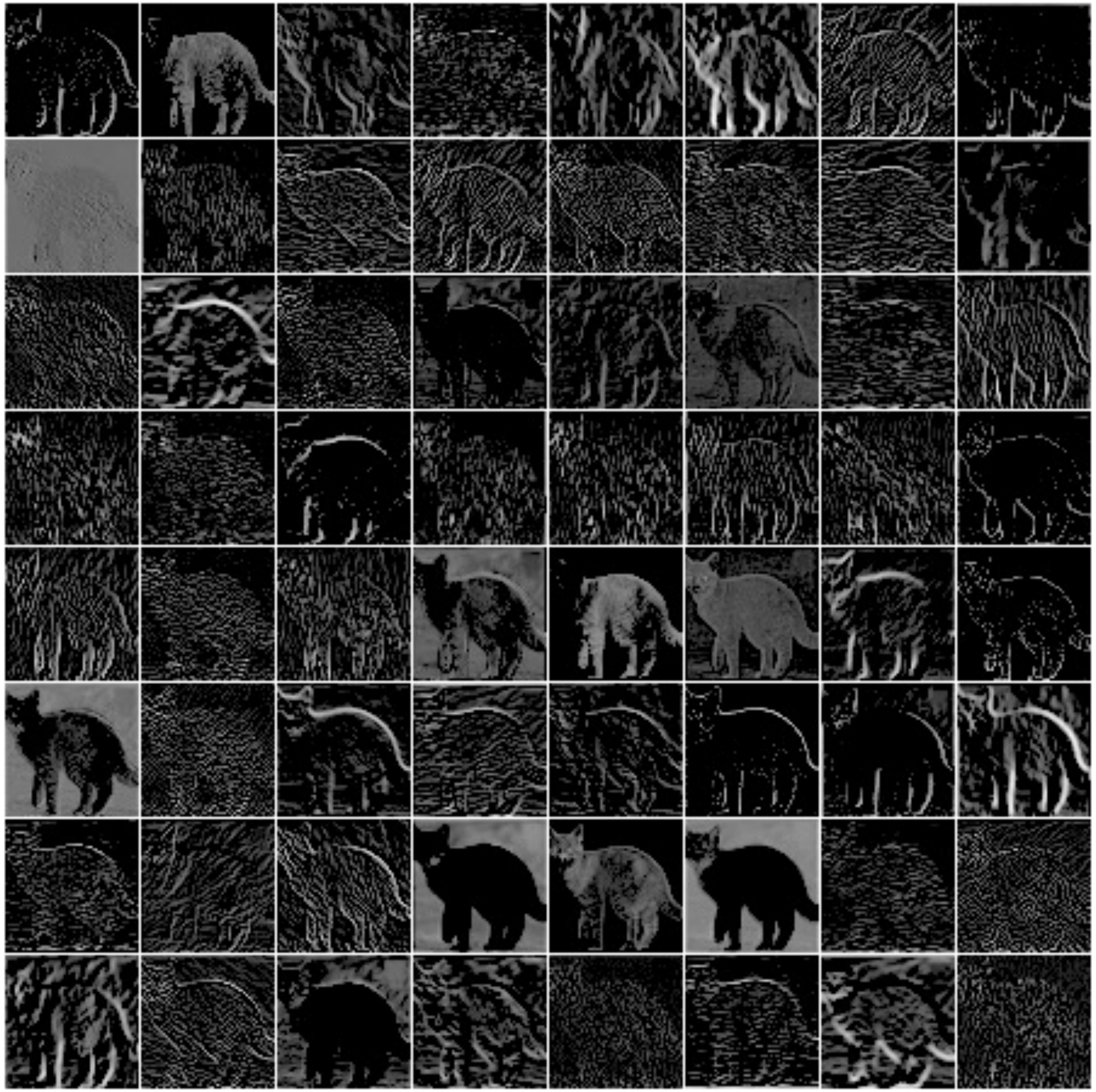


Case Study: AlexNet

Input Image

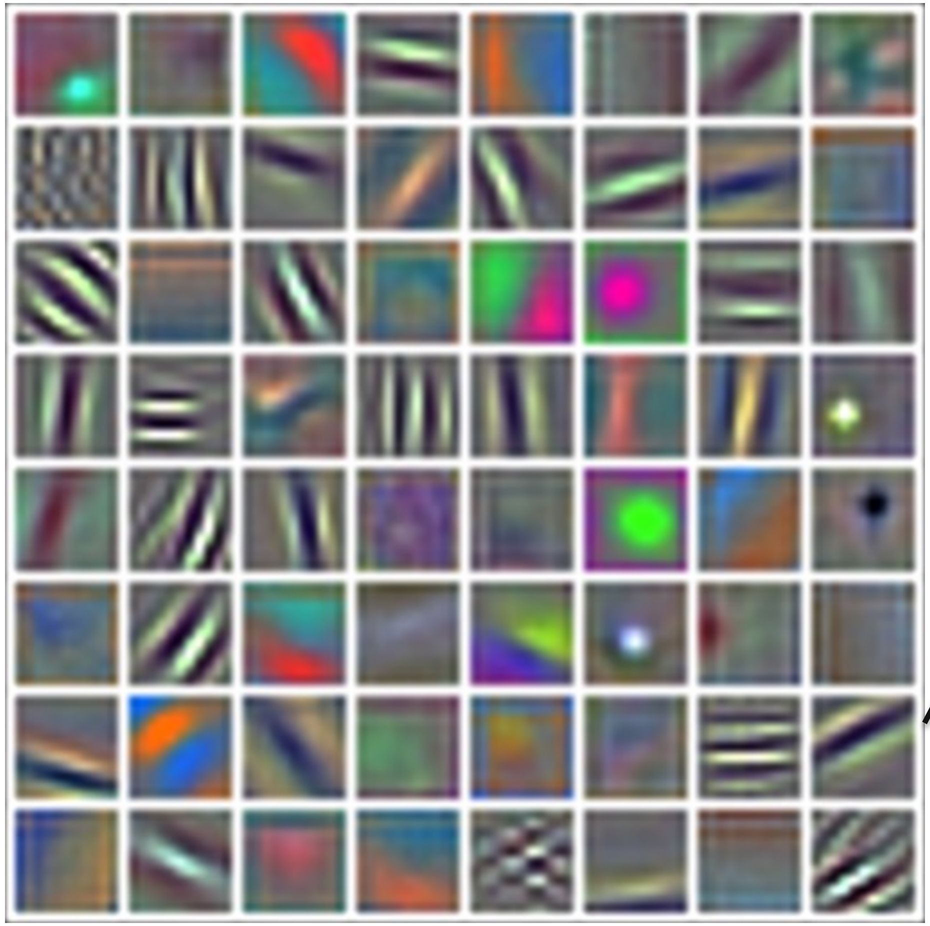
First-layer Filters





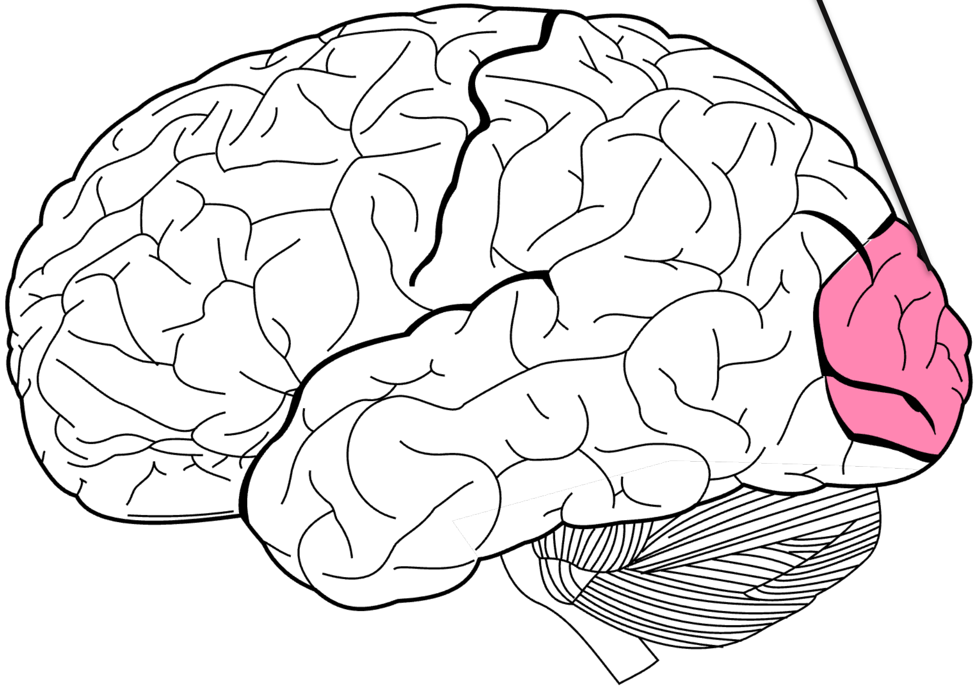
Activations

Case Study: AlexNet



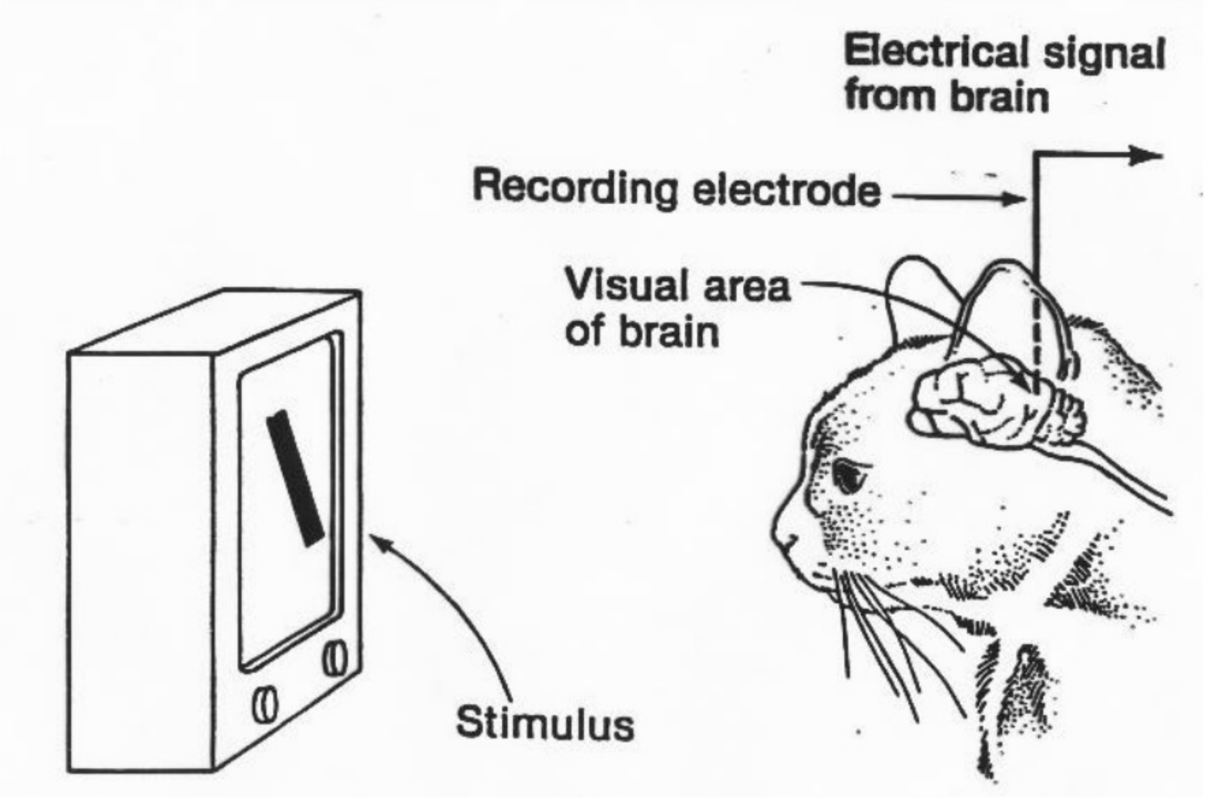
First-layer Filters

Similar to simple cells in visual cortex!
- Edge detectors



Case Study: AlexNet

[Hubel & Wiesel 1959]



Simple cells in visual cortex detect edges, complex cells compose earlier responses

CNN higher layer filters

[Olah '17]

Dataset examples that maximize neuron outputs

CNN Building Blocks

Design choices:

- filter size
- number of filters
- padding
- stride

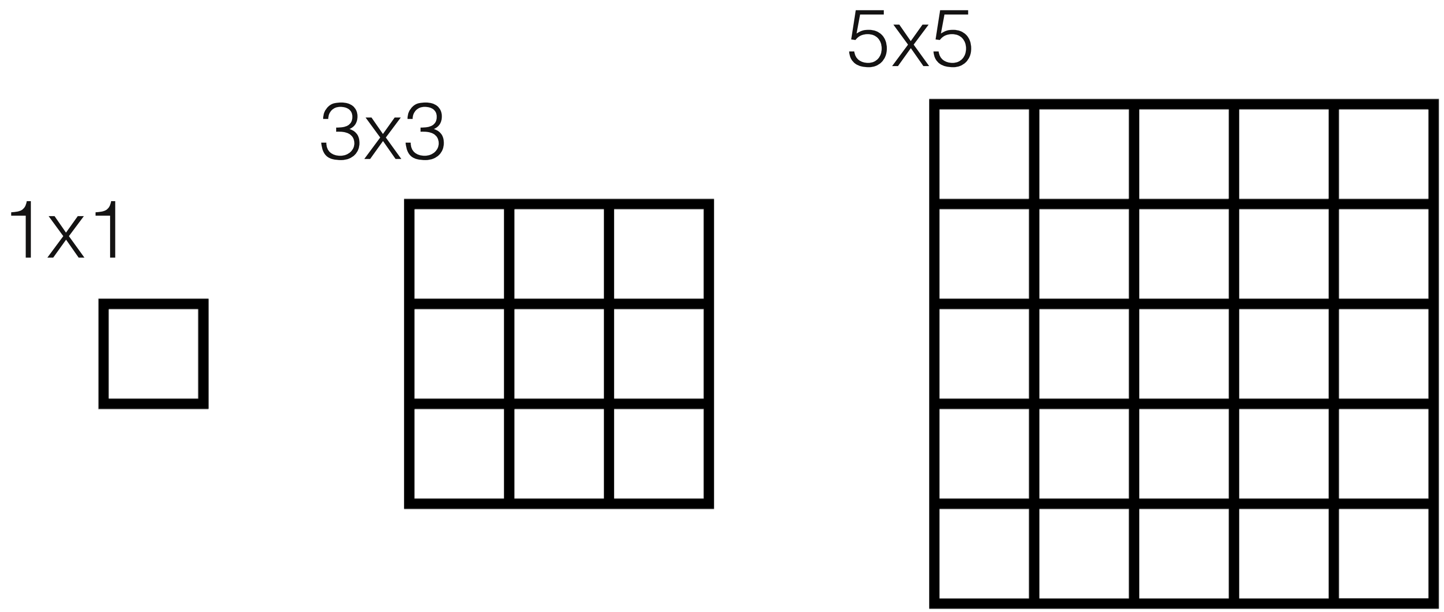
Layer types:

- pooling
- transpose convolutions
- upsampling layers*
- batch normalization*
- softmax layers*

**no time to cover all of these layers! Check out PyTorch docs for details…*
e.g., https://pytorch.org/docs/stable/generated/torch.nn.BatchNorm2d.html

CNN Building Blocks

Filter size



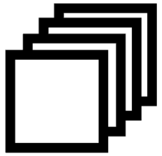
CNN Building Blocks

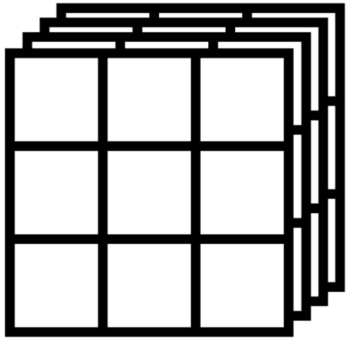
Number of channels

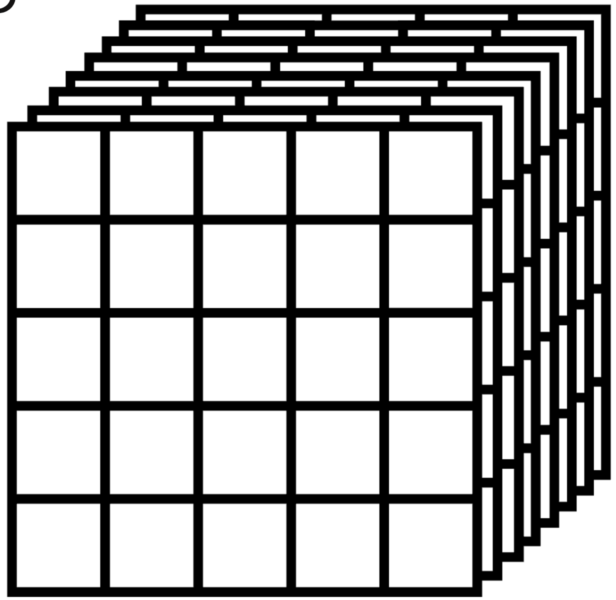
N_out x N_in x 5 x 5

N_out x N_in x 3 x 3

N_out x N_in x 1 x 1

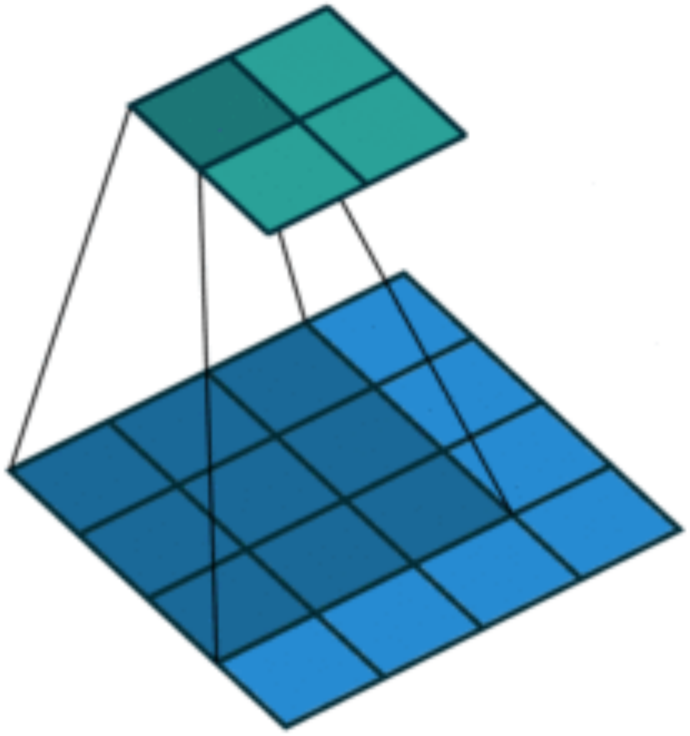




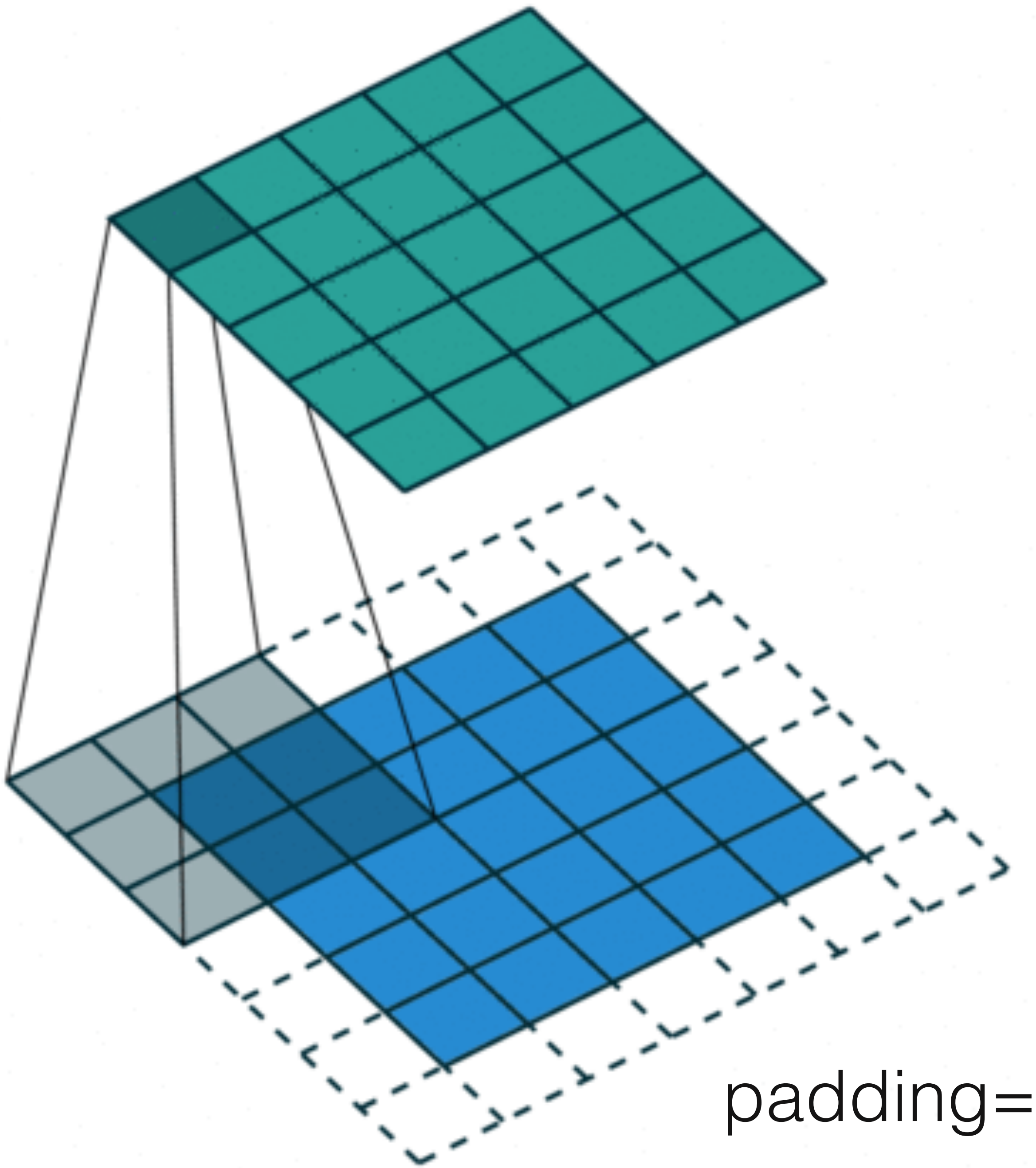


CNN Building Blocks

padding



No padding

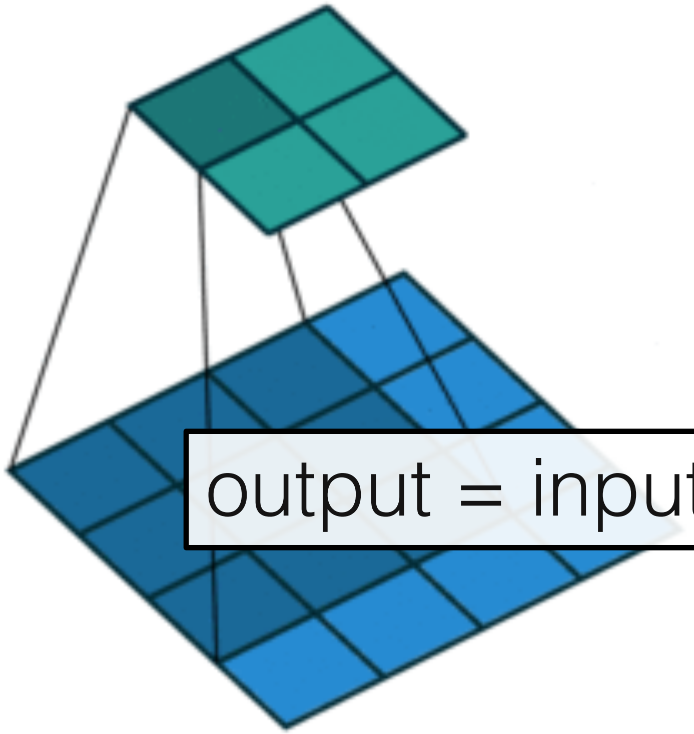


padding=1

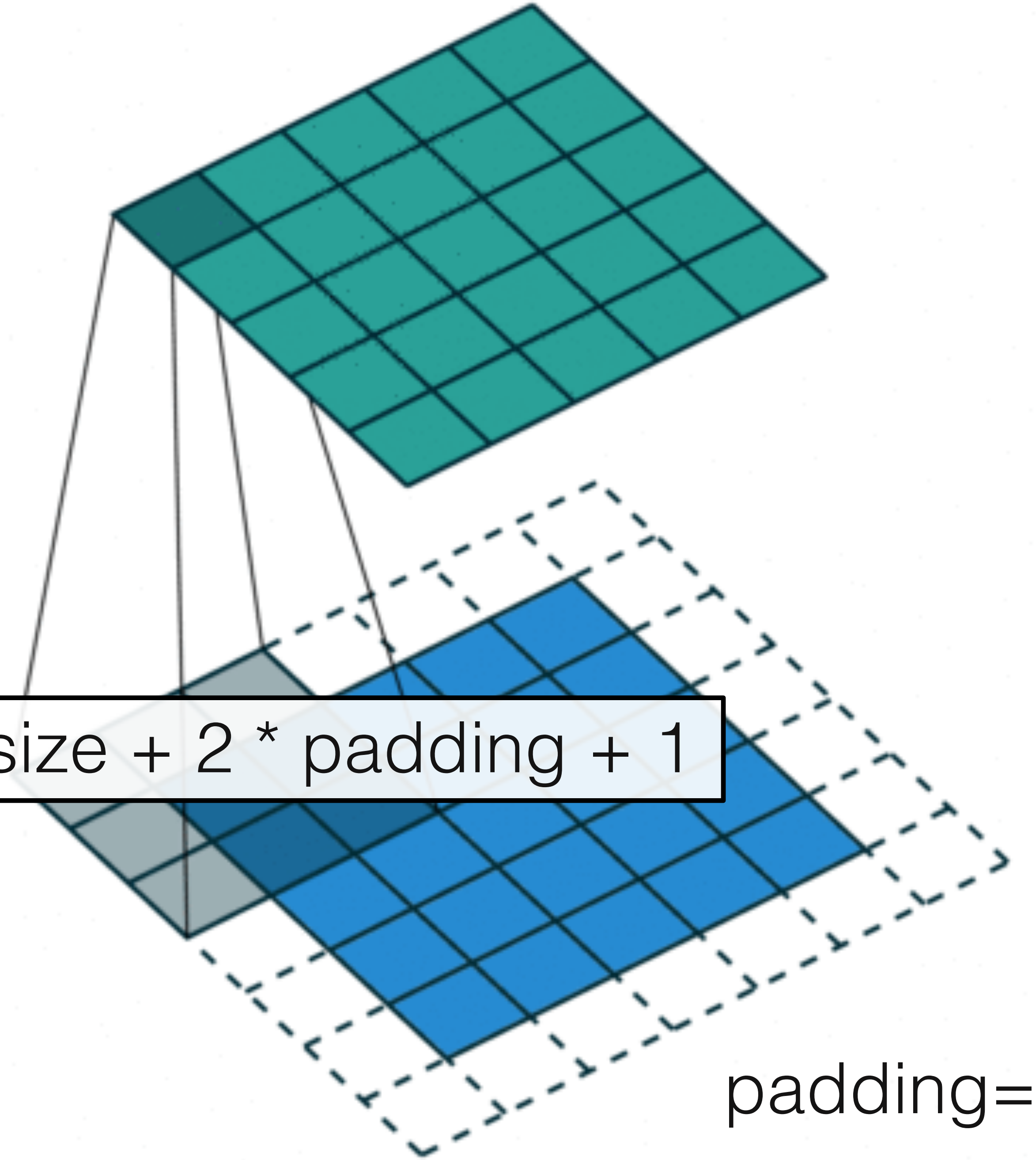
https://github.com/vdumoulin/conv_arithmetic

CNN Building Blocks

padding



No padding



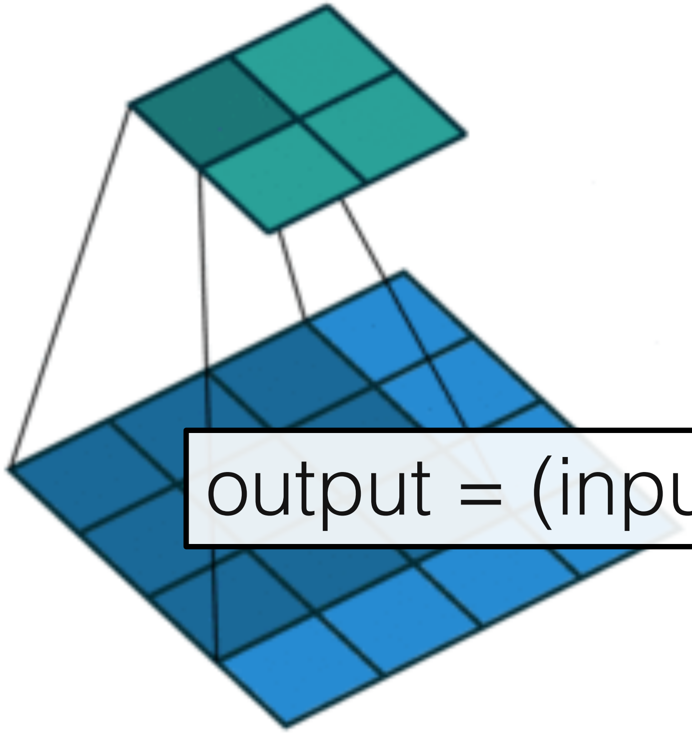
padding=1

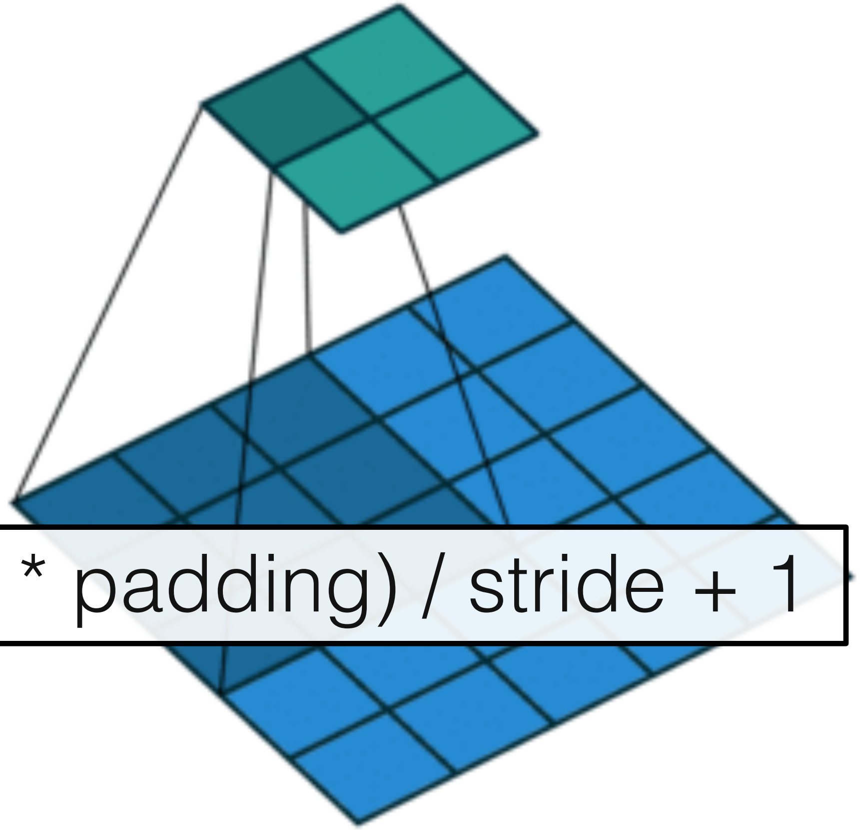
output = input – filtersize + 2 * padding + 1

https://github.com/vdumoulin/conv_arithmetic

CNN Building Blocks

stride





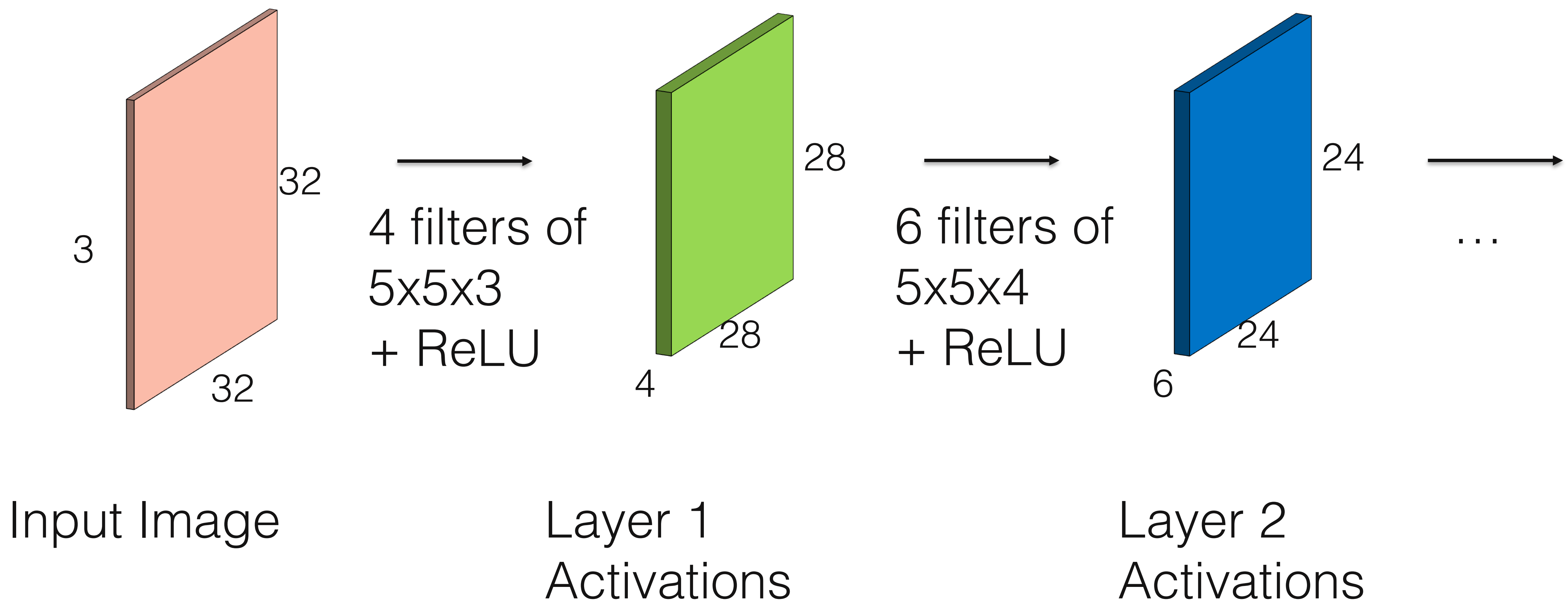
output = (input – filtersize + 2 * padding) / stride + 1

stride = 1

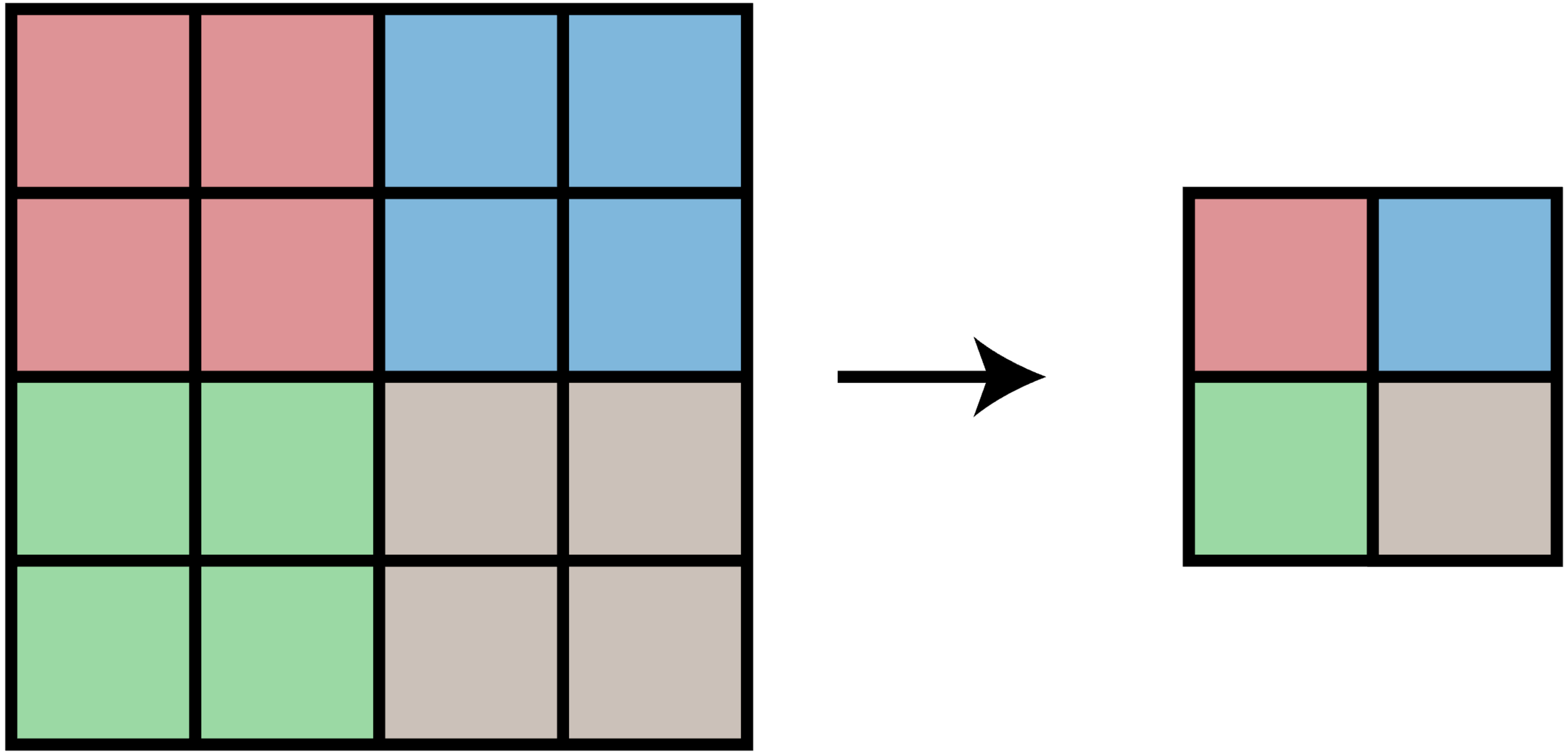
stride = 2

https://github.com/vdumoulin/conv_arithmetic

Convolutional Neural Network



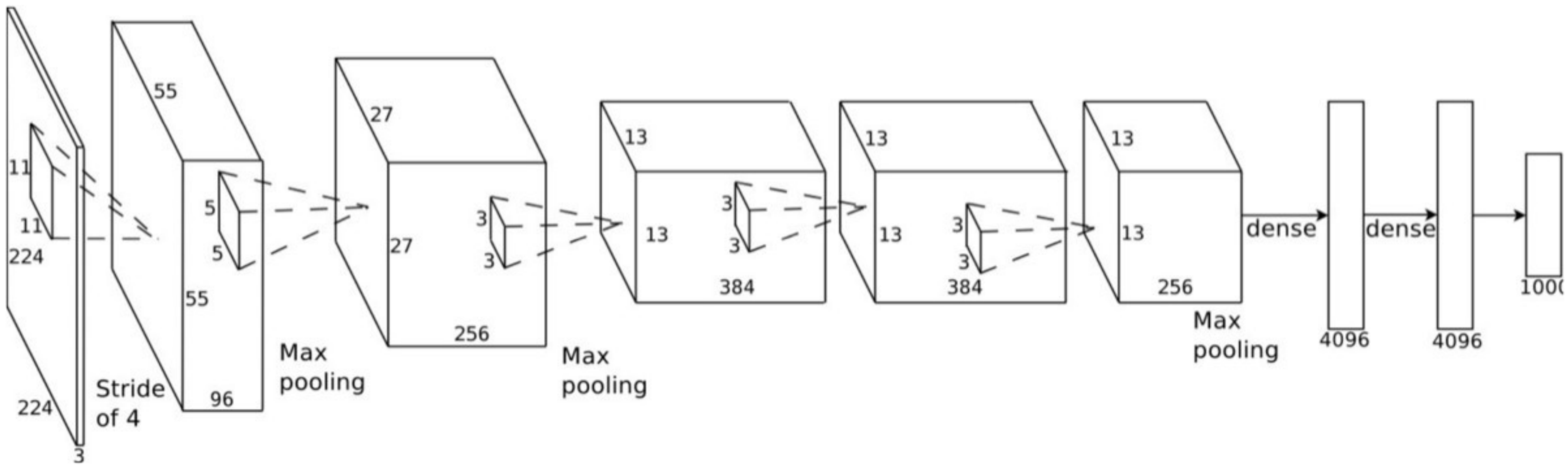
Layer types: Pooling



e.g., max pool size=2, stride=2

Convolutional Neural Networks (CNN)

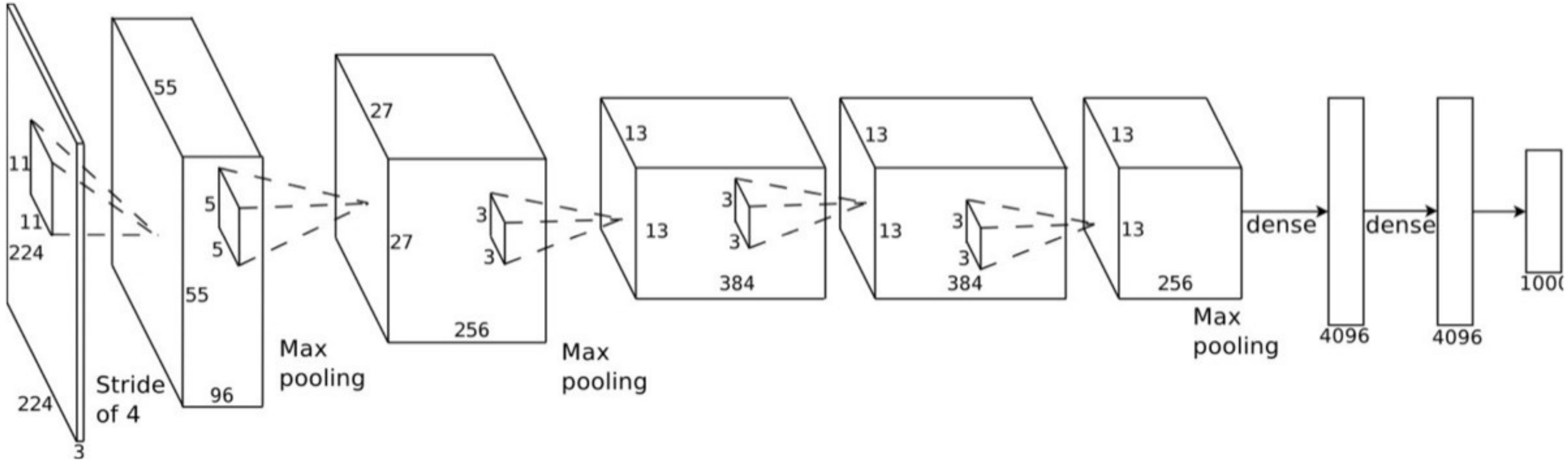
- AlexNet (from UofT!): A. Krizhevsky, I. Sutskever, G. E. Hinton, ImageNet Classification with Deep Convolutional Neural Networks, NeurIPS 2012. This network won the Imagenet Challenge of 2012, and revolutionized computer vision.



[Pic adopted from: A. Krizhevsky]

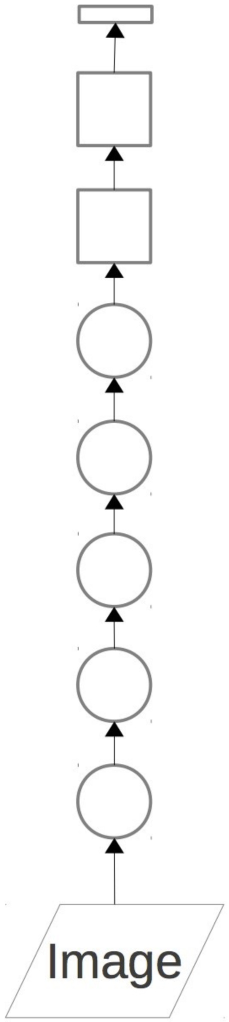
Convolutional Neural Networks (CNN)

- AlexNet (from UofT!): A. Krizhevsky, I. Sutskever, G. E. Hinton, ImageNet Classification with Deep Convolutional Neural Networks, NeurIPS 2012. This network won the Imagenet Challenge of 2012, and revolutionized computer vision.
- How many parameters (weights) does this network have?



[Pic adopted from: A. Krizhevsky]

Convolutional Neural Networks (CNN)



- Trained with stochastic gradient descent on two NVIDIA GPUs for about a week
- 650,000 neurons
- 60,000,000 parameters
- 630,000,000 connections
- **Final feature layer:** 4096-dimensional

Convolutional layer: convolves its input with a bank of 3D filters, then applies point-wise non-linearity

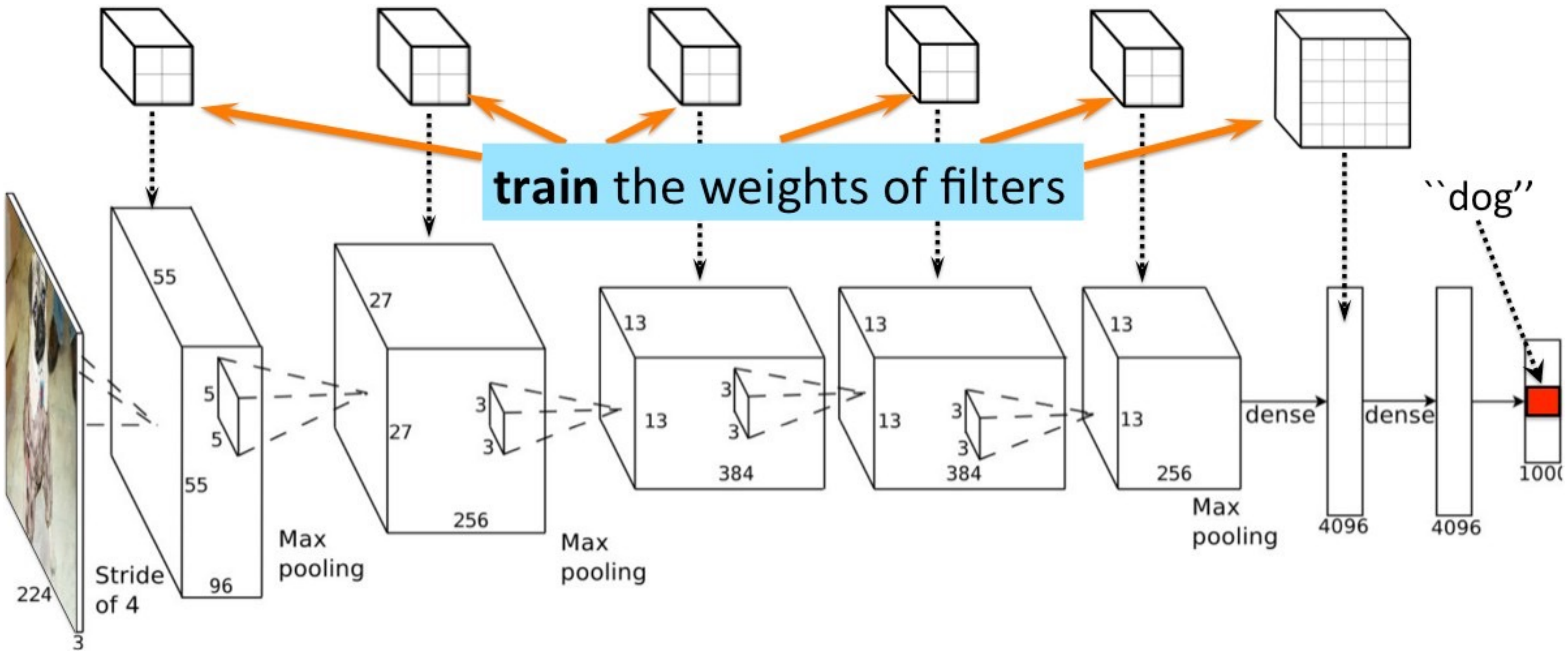
Fully-connected layer: applies linear filters to its input, then applies point-wise non-linearity

Figure: From http://www.image-net.org/challenges/LSVRC/2012/supervision.pdf

[Pic adopted from: A. Krizhevsky]

Convolutional Neural Networks (CNN)

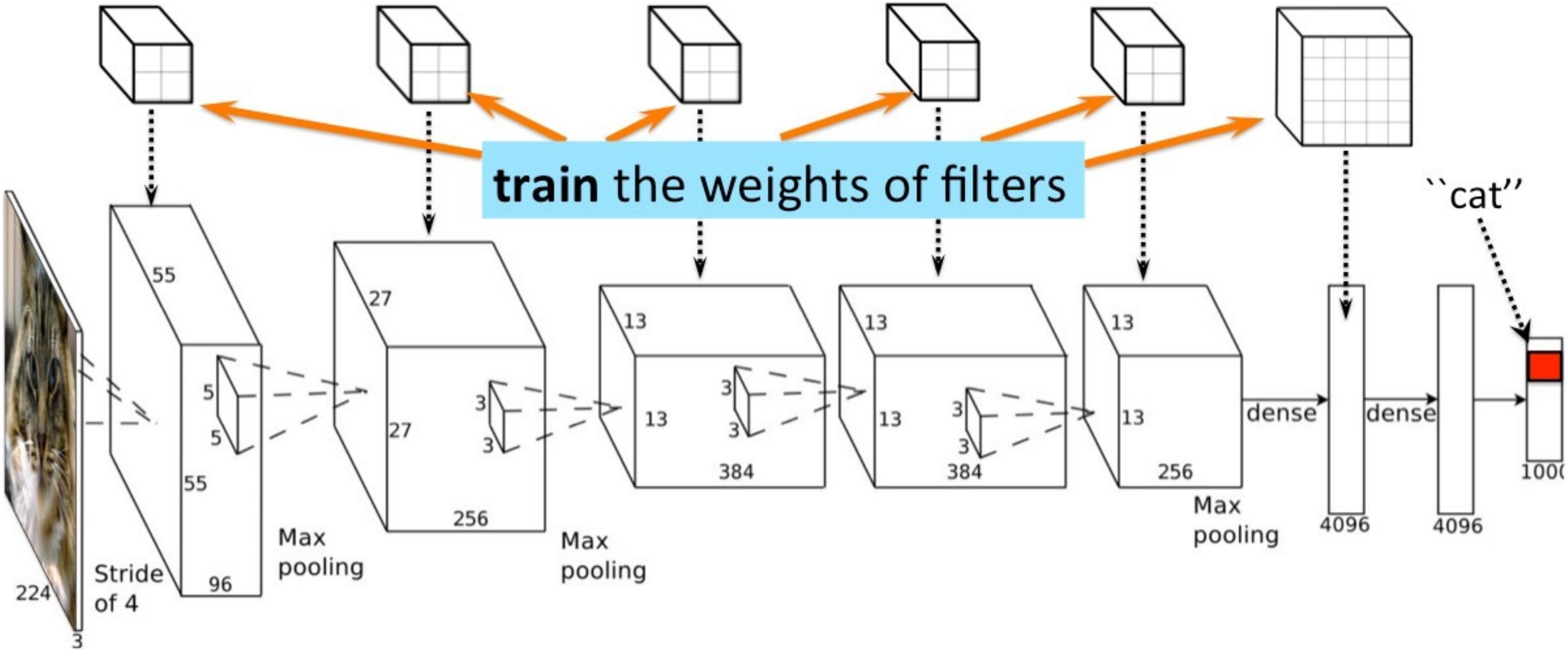
- The trick is to not hand-fix the weights, but to train them. Train them such that when the network sees a picture of a dog, the last layer will say "dog".



[Pic adopted from: A. Krizhevsky]

Convolutional Neural Networks (CNN)

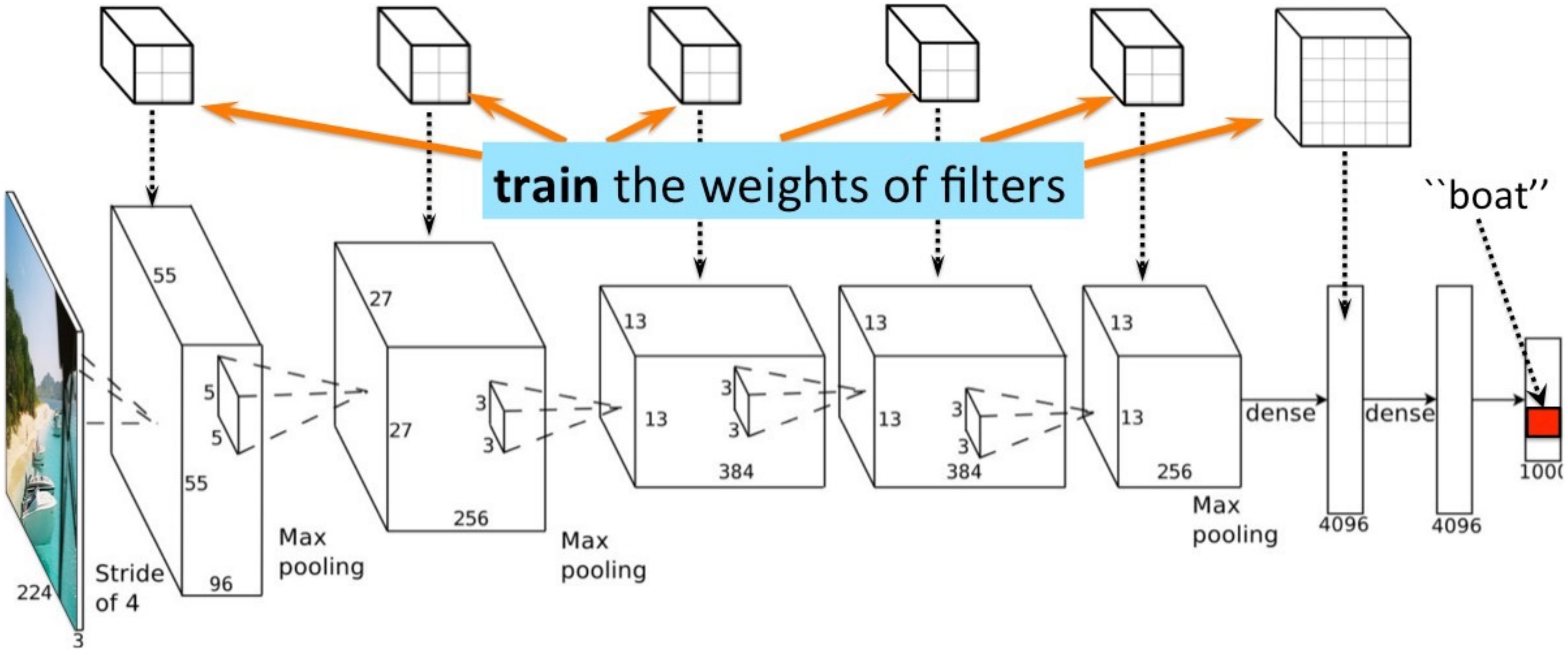
- Or when the network sees a picture of a cat, the last layer will say "cat".



[Pic adopted from: A. Krizhevsky]

Convolutional Neural Networks (CNN)

- Or when the network sees a picture of a boat, the last layer will say "boat"... The more pictures the network sees, the better.



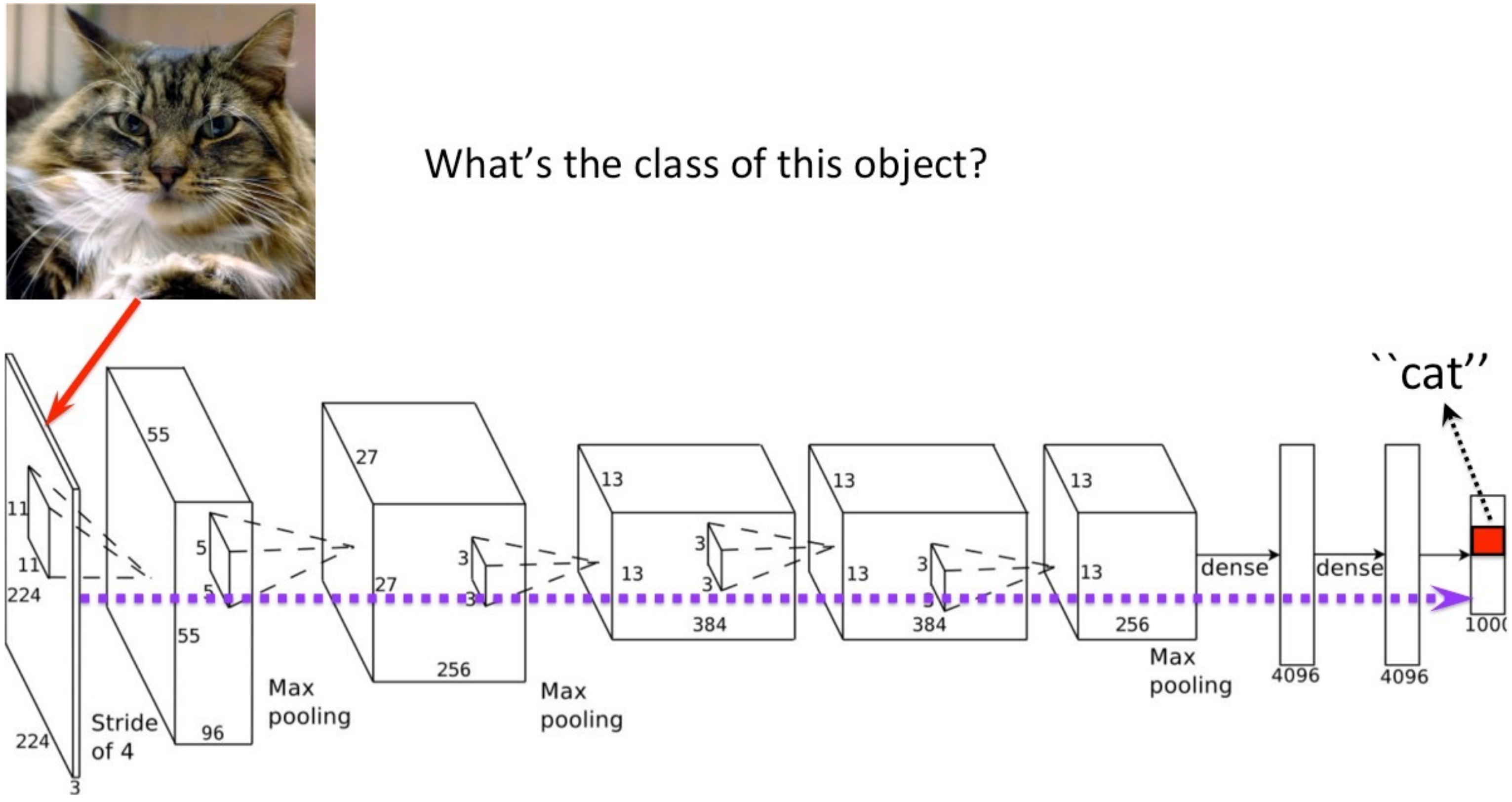
Train on **lots** of examples. Millions. Tens of millions. Wait a week for training to finish.

Share your network (the weights) with others who are not fortunate enough with GPU power.

[Pic adopted from: A. Krizhevsky]

Classification

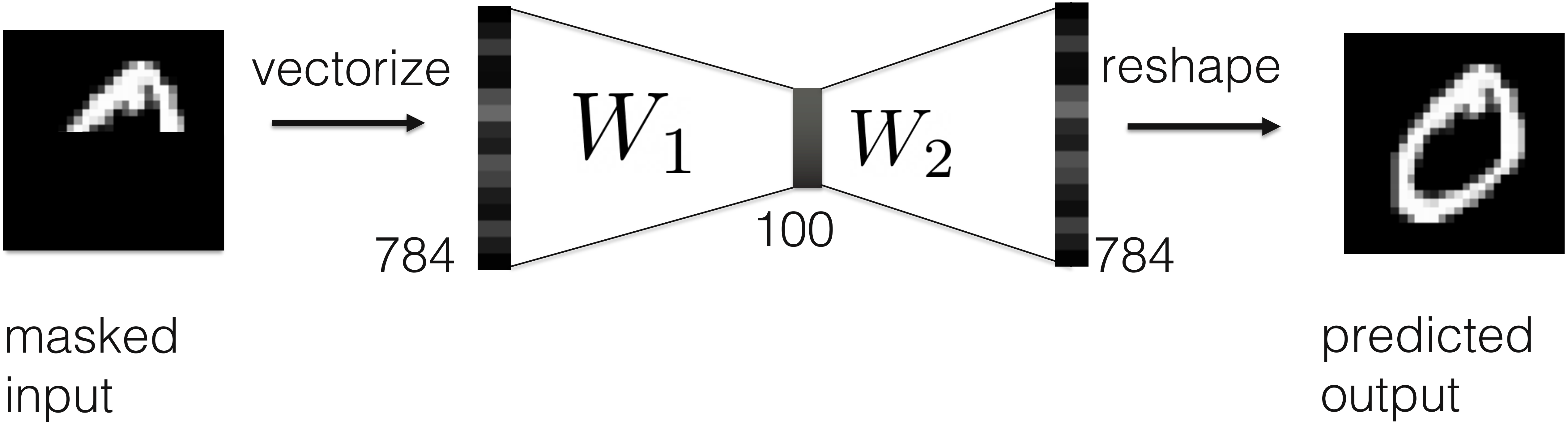
- Once trained we can do classification. Just feed in an image or a crop of the image, run through the network, and read out the class with the highest probability in the last (classification) layer.



Overview

- Motivation
- Fully-connected Networks
- Convolutional Neural Networks
- Training networks

Image Inpainting



Training the MLP

masked images

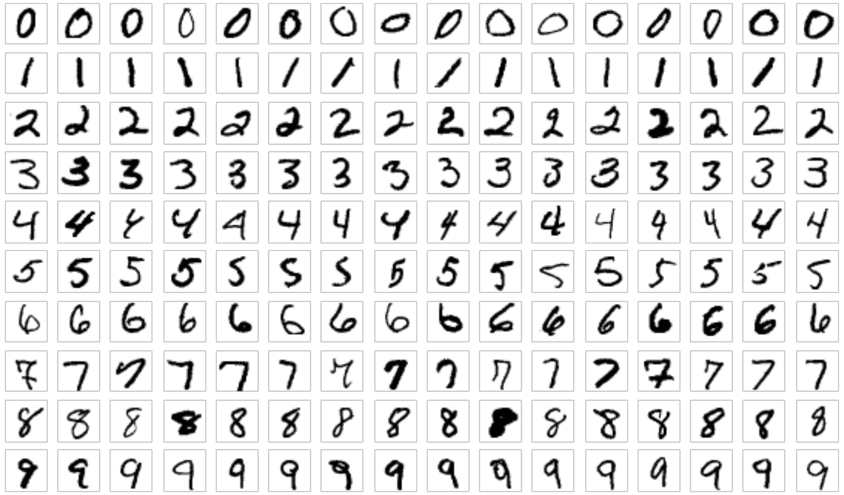
Image inpainting example

Training dataset:

- masked and complete image pairs

- train network to predict the complete image

ground truth



Training the MLP

Train the network to minimize the loss function

$$\mathcal{L}_\theta = \frac{1}{2}||y - \hat{y}||_2^2$$

network
parameters

$\theta = \{W_1, W_2\}$

Training the MLP

Train the network to minimize the loss function

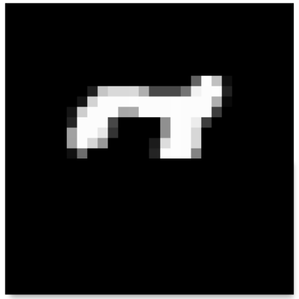
$$\mathcal{L}_\theta = \frac{1}{2}||y - \hat{y}||_2^2$$

network parameters

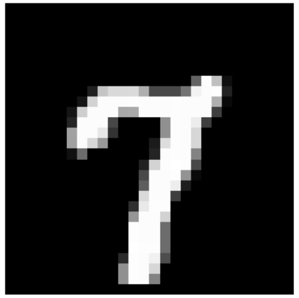
$\theta = \{W_1, W_2\}$

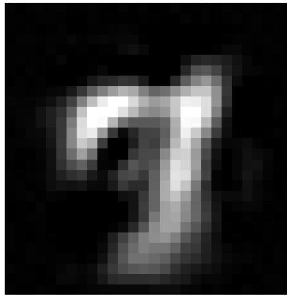
ground truth image

network prediction



input





Training the MLP

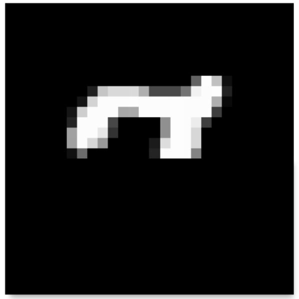
How do we figure out θ?

$$\mathcal{L}_\theta = \frac{1}{2}||y - \hat{y}||_2^2$$

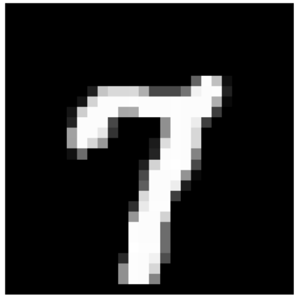
network
parameters
$\theta = \{W_1, W_2\}$

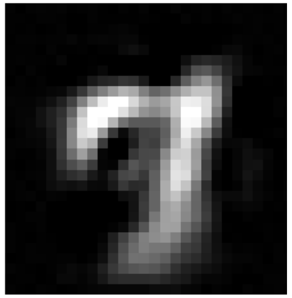
ground truth image

network prediction



input

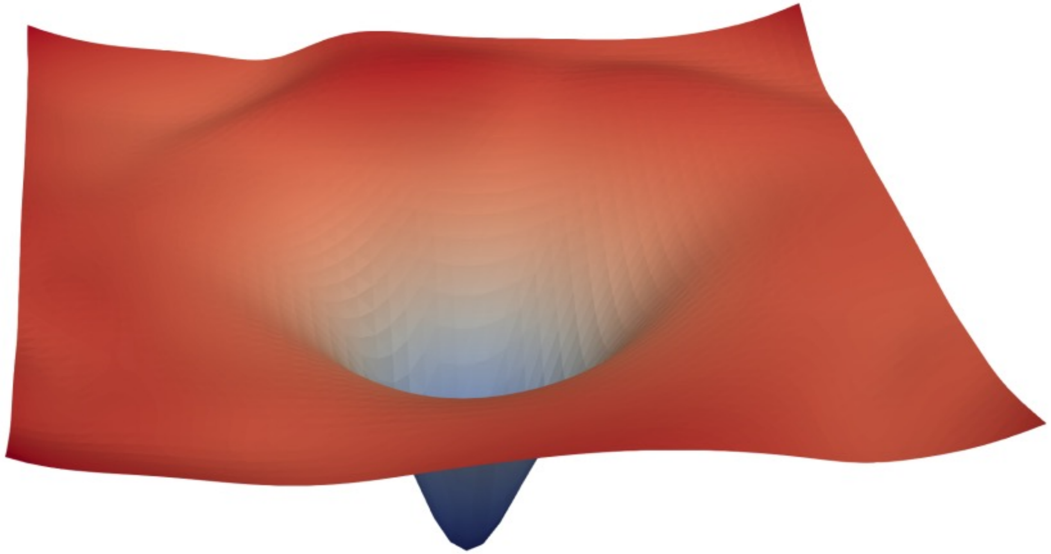




Training the MLP

Gradient-based optimization

$$\theta^{(k+1)} = \theta^{(k)} - \nabla_\theta \mathcal{L}_\theta$$



Loss Landscape

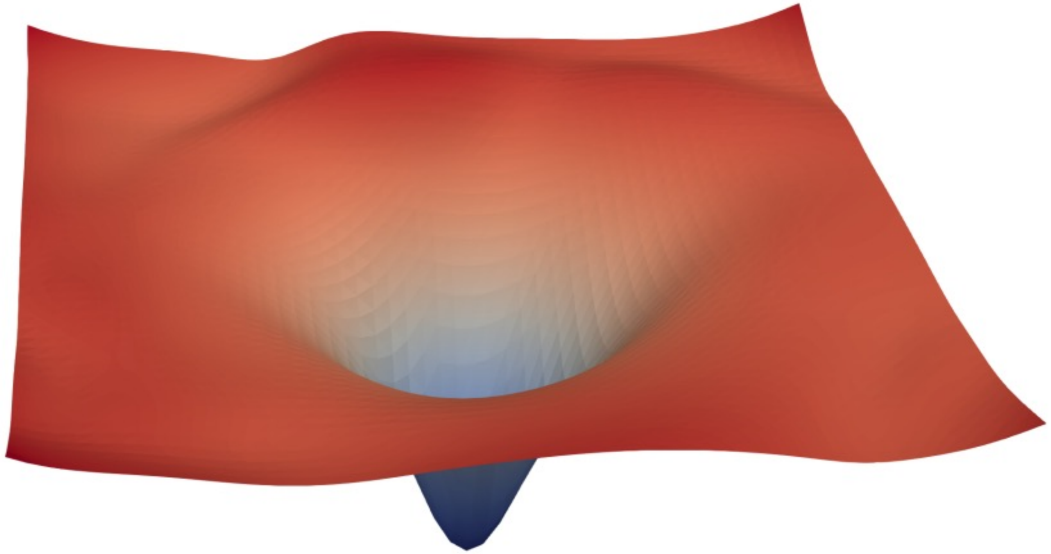
[Li et al. '18]

Training the MLP

Gradient-based optimization

$$\theta^{(k+1)} = \theta^{(k)} - \nabla_\theta \mathcal{L}_\theta$$

Need to calculate the partial derivative with respect to each parameter



Loss Landscape

[Li et al. '18]

Training the MLP

Generally there are 3 options

1. Numerical differentiation
2. Symbolic differentiation
3. “Automatic” differentiation

Numerical Differentiation

$$\frac{\partial f(x)}{\partial x} \approx \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

Not very accurate, computationally expensive

Easy to implement! Can be used to check your analytical answers..

Symbolic Differentiation

$$
\begin{aligned}
\frac{\partial \mathcal{L}_\theta}{\partial W_1} &= \frac{\partial}{\partial W_1} \frac{1}{2} \|y - \hat{y}\|_2^2 \\
&= \frac{\partial}{\partial W_1} \frac{1}{2} \left(W_2 \sigma(W_1 x)\right)^T \left(W_2 \sigma(W_1 x)\right) \\
&= \frac{\partial}{\partial W_1} \frac{1}{2} \sigma(W_1 x)^T W_2^T W_2 \sigma(W_1 x) \\
&= \dots \quad \text{chain rule, product rule…}
\end{aligned}
$$

Accurate, but must be manually calculated for each term
Tedious!

Automatic Differentiation

Think about the problem as a "computational graph"

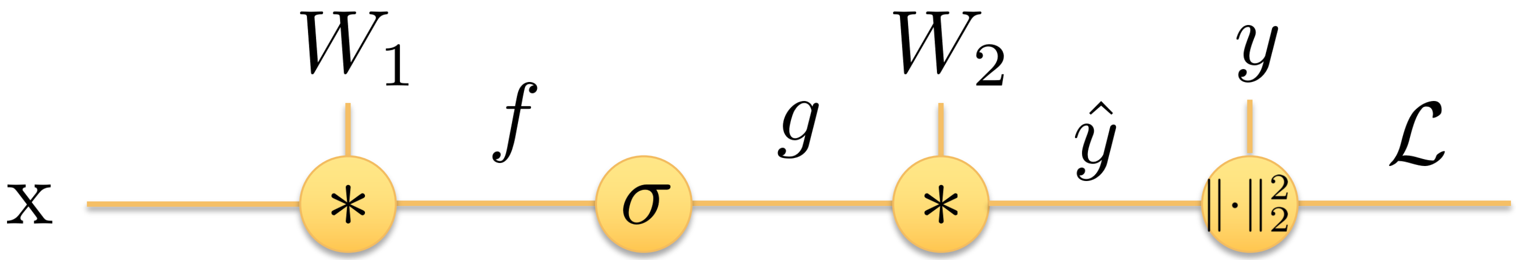
Divide and conquer using the chain rule

Enables "backpropagation" – an efficient way to take derivatives of all parameters in a computational graph

Automatic Differentiation

Think about the problem as a "computational graph"

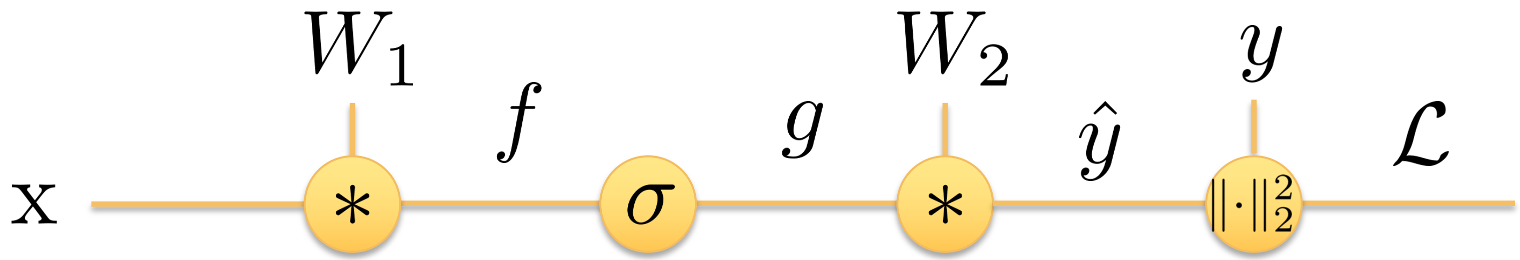
Divide and conquer using the chain rule



Automatic Differentiation

Think about the problem as a "computational graph"

Divide and conquer using the chain rule

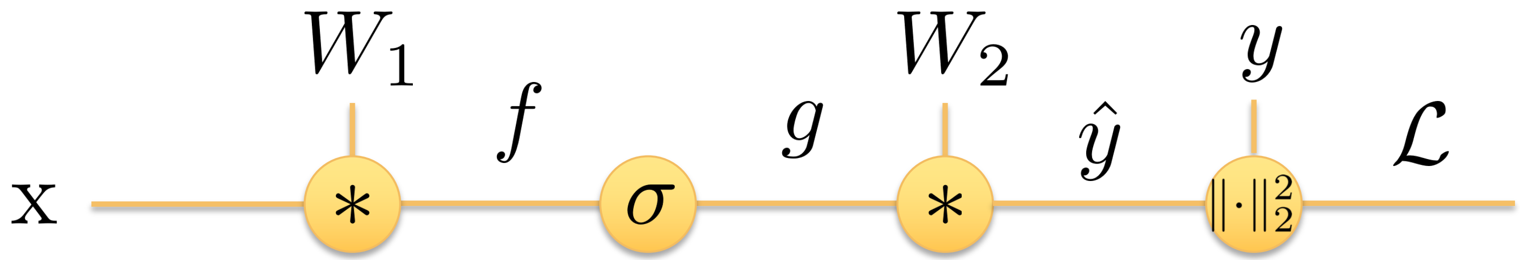


$$\frac{\partial \mathcal{L}}{\partial W_2} = \frac{\partial \hat{y}}{\partial W_2} \frac{\partial \mathcal{L}}{\partial \hat{y}}$$

Automatic Differentiation

Think about the problem as a "computational graph"

Divide and conquer using the chain rule

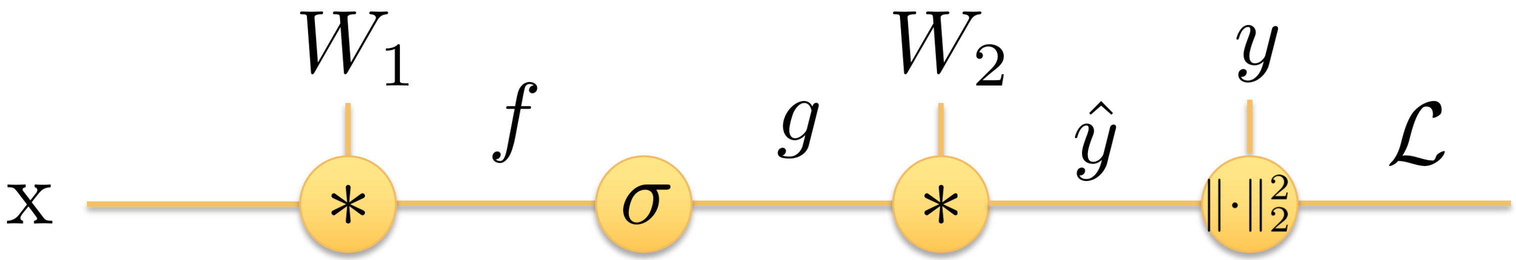


$$\frac{\partial \mathcal{L}}{\partial W_1} = \frac{\partial f}{\partial W_1} \frac{\partial g}{\partial f} \frac{\partial \hat{y}}{\partial g} \frac{\partial \mathcal{L}}{\partial \hat{y}}$$

Automatic Differentiation

Think about the problem as a "computational graph"

Divide and conquer using the chain rule

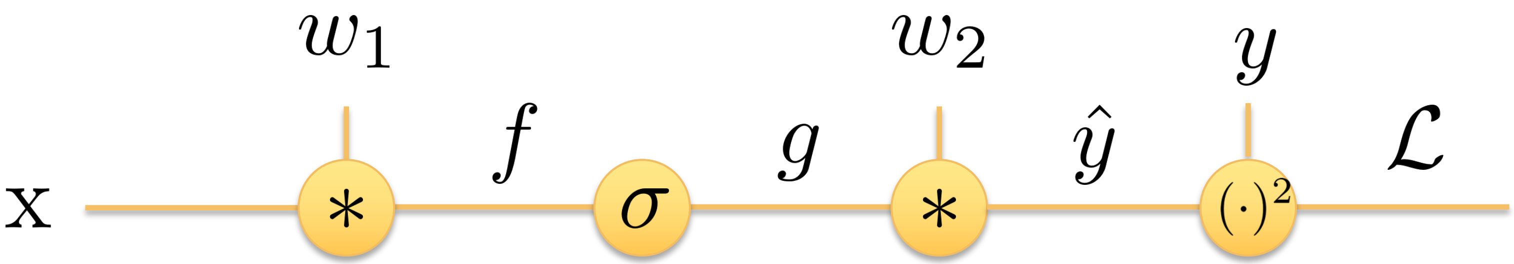


$$\frac{\partial \mathcal{L}}{\partial W_1} = \frac{\partial f}{\partial W_1} \frac{\partial g}{\partial f} \frac{\partial \hat{y}}{\partial g} \frac{\partial \mathcal{L}}{\partial \hat{y}}$$

We can calculate analytical expressions for each of these terms and then plug in our values

Autodiff Example

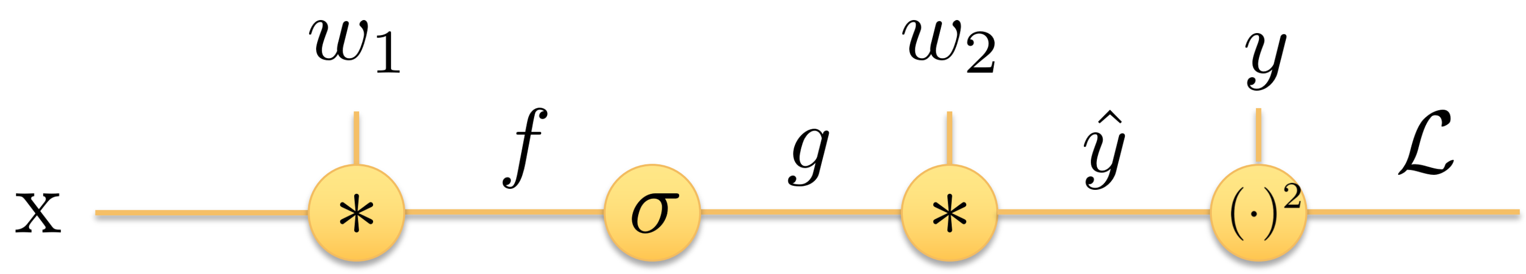
(assume scalar values for now)



$$\frac{\partial \mathcal{L}}{\partial w_1} = \frac{\partial f}{\partial w_1} \frac{\partial g}{\partial f} \frac{\partial \hat{y}}{\partial g} \boxed{\frac{\partial \mathcal{L}}{\partial \hat{y}}}$$

Autodiff Example

(assume scalar values for now)

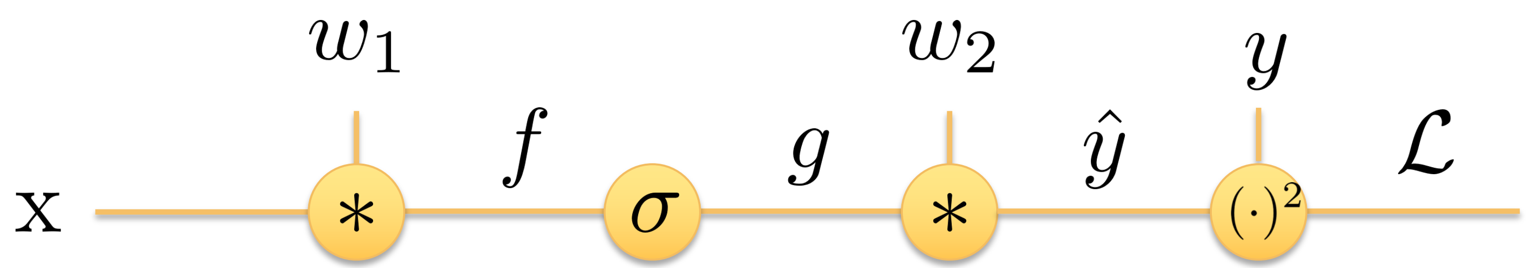


$$\frac{\partial \mathcal{L}}{\partial w_1} = \frac{\partial f}{\partial w_1} \frac{\partial g}{\partial f} \frac{\partial \hat{y}}{\partial g} \boxed{\frac{\partial \mathcal{L}}{\partial \hat{y}}}$$

$$\frac{\partial \mathcal{L}}{\partial \hat{y}} = \frac{\partial}{\partial \hat{y}} \frac{1}{2} (\hat{y} - y)^2 = \hat{y} - y$$

Autodiff Example

(assume scalar values for now)

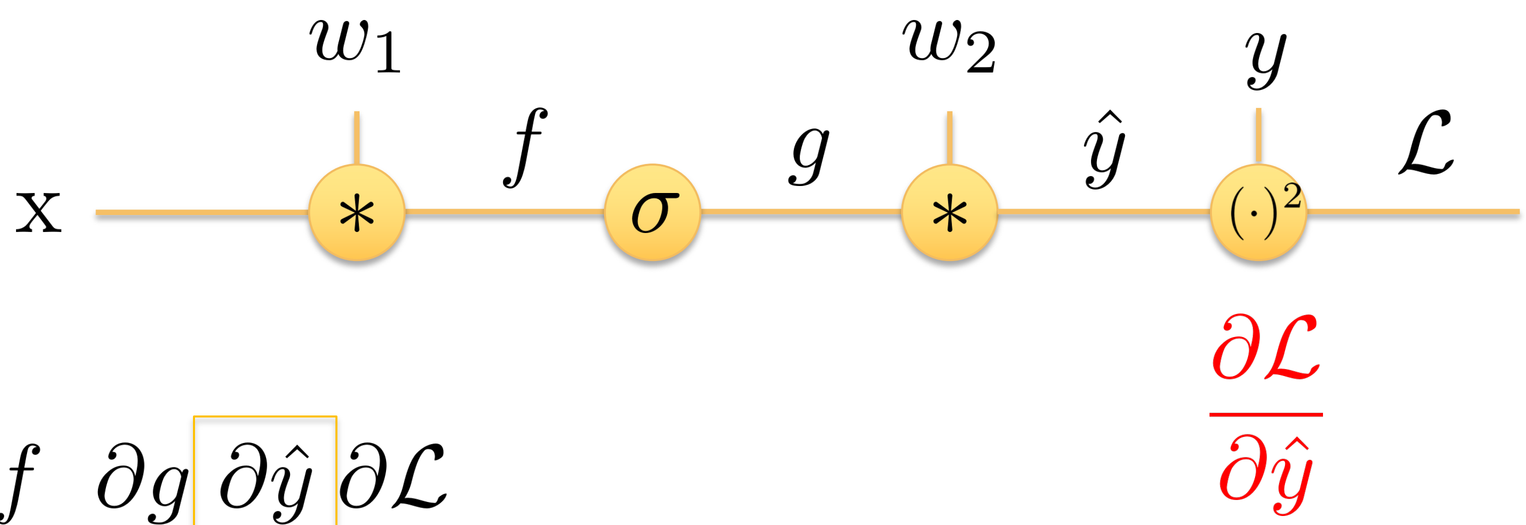


$$\frac{\partial \mathcal{L}}{\partial \hat{y}}$$

$$\frac{\partial \mathcal{L}}{\partial w_1} = \frac{\partial f}{\partial w_1} \frac{\partial g}{\partial f} \boxed{\frac{\partial \hat{y}}{\partial g}} \frac{\partial \mathcal{L}}{\partial \hat{y}}$$

Autodiff Example

(assume scalar values for now)

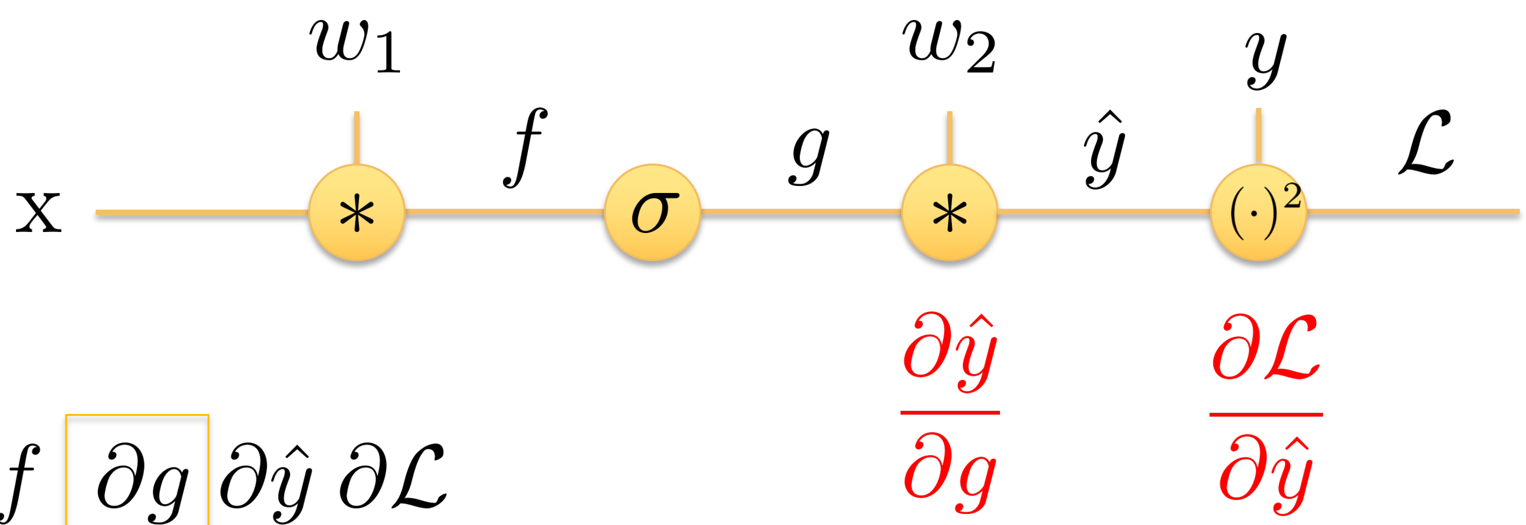


$$\frac{\partial \mathcal{L}}{\partial w_1} = \frac{\partial f}{\partial w_1} \frac{\partial g}{\partial f} \boxed{\frac{\partial \hat{y}}{\partial g}} \frac{\partial \mathcal{L}}{\partial \hat{y}}$$

$$\frac{\partial \hat{y}}{\partial g} = \frac{\partial}{\partial g} w_2 \cdot g = w_2$$

Autodiff Example

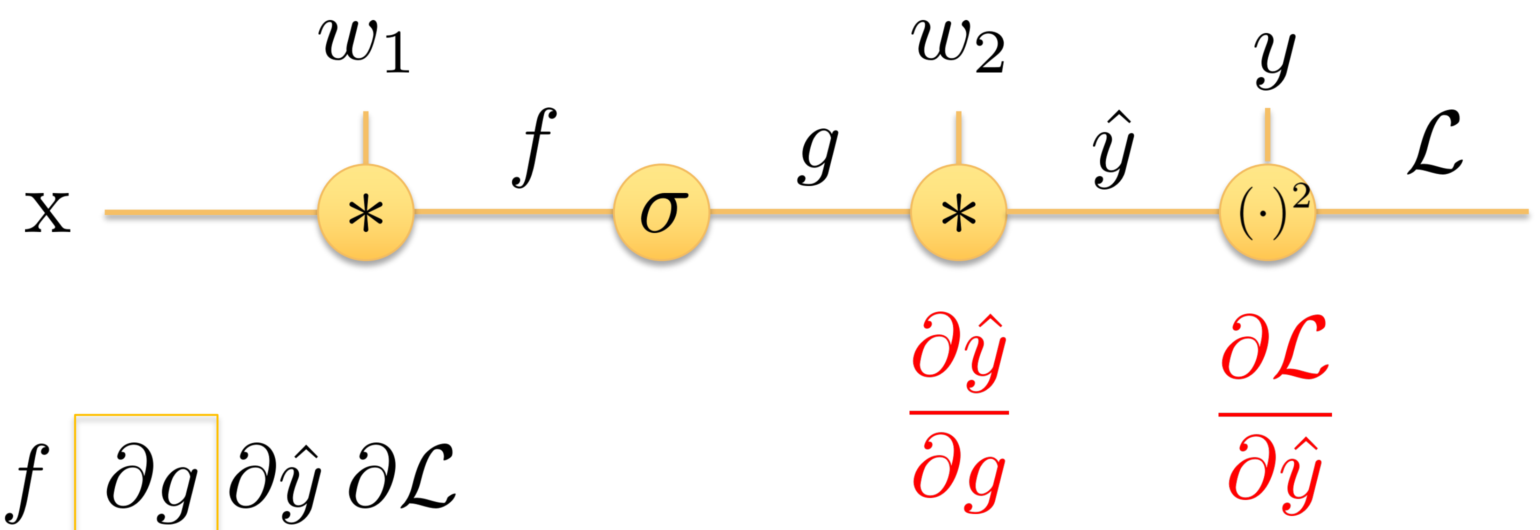
(assume scalar values for now)



$$\frac{\partial \mathcal{L}}{\partial w_1} = \frac{\partial f}{\partial w_1} \boxed{\frac{\partial g}{\partial f}} \frac{\partial \hat{y}}{\partial g} \frac{\partial \mathcal{L}}{\partial \hat{y}}$$

Autodiff Example

(assume scalar values for now)

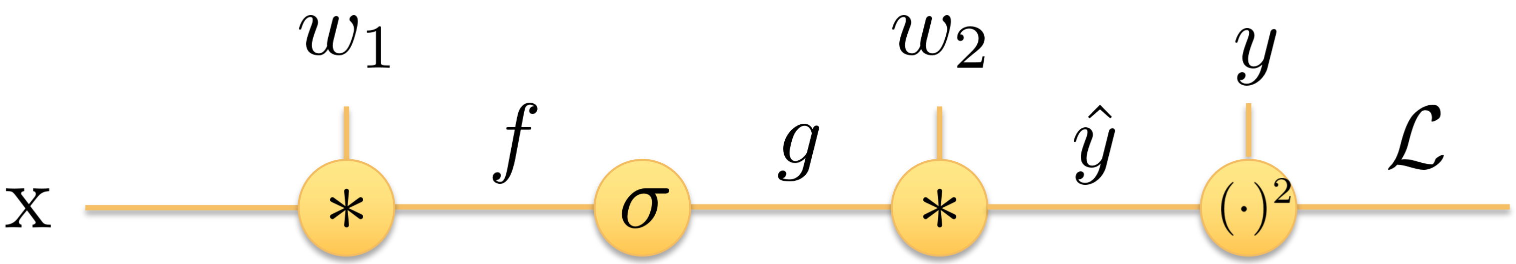


$$\frac{\partial \mathcal{L}}{\partial w_1} = \frac{\partial f}{\partial w_1} \boxed{\frac{\partial g}{\partial f}} \frac{\partial \hat{y}}{\partial g} \frac{\partial \mathcal{L}}{\partial \hat{y}}$$

$$\frac{\partial g}{\partial f} = \frac{\partial}{\partial f}\sigma(f) = \frac{\partial}{\partial f}\max(0, f) = \begin{cases} 0, & f < 0 \\ 1 & \text{else} \end{cases}$$

Autodiff Example

(assume scalar values for now)

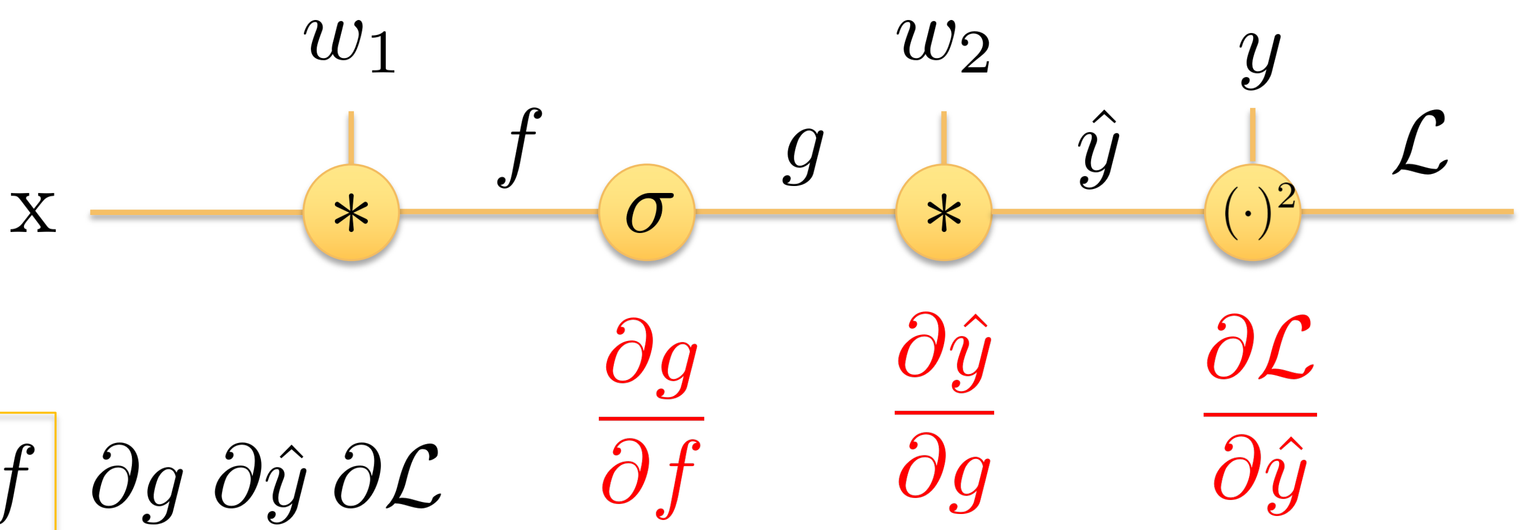


$$\frac{\partial g}{\partial f} \qquad \frac{\partial \hat{y}}{\partial g} \qquad \frac{\partial \mathcal{L}}{\partial \hat{y}}$$

$$\frac{\partial \mathcal{L}}{\partial w_1} = \boxed{\frac{\partial f}{\partial w_1}} \frac{\partial g}{\partial f} \frac{\partial \hat{y}}{\partial g} \frac{\partial \mathcal{L}}{\partial \hat{y}}$$

Autodiff Example

(assume scalar values for now)

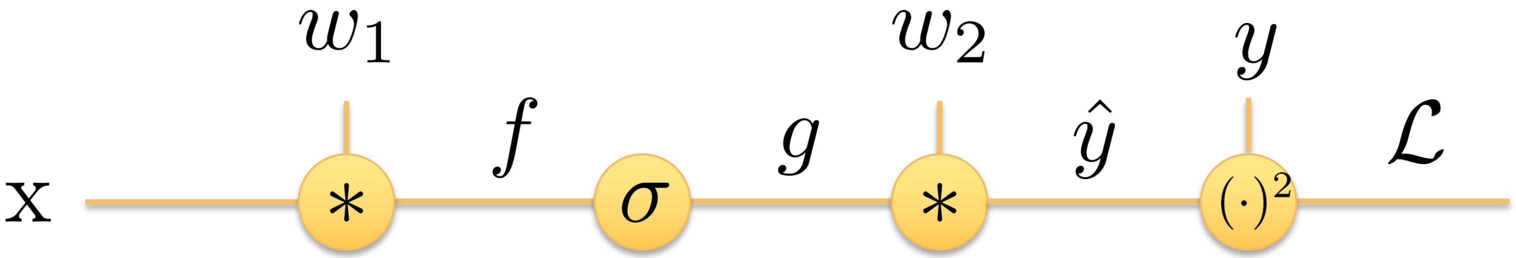


$$\frac{\partial \mathcal{L}}{\partial w_1} = \boxed{\frac{\partial f}{\partial w_1}} \frac{\partial g}{\partial f} \frac{\partial \hat{y}}{\partial g} \frac{\partial \mathcal{L}}{\partial \hat{y}}$$

$$\frac{\partial f}{\partial w_1} = \frac{\partial}{\partial w_1} w_1 \cdot x = x$$

Autodiff Example

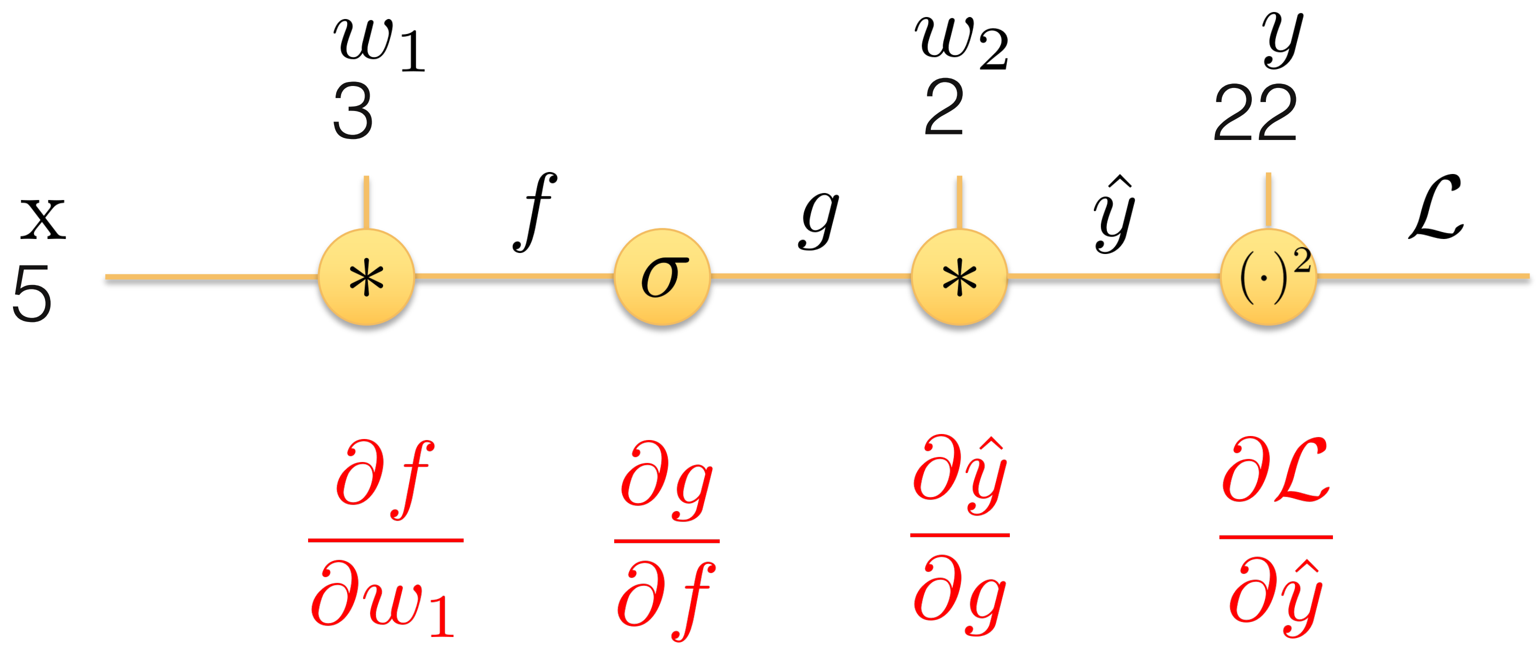
(assume scalar values for now)



$$\frac{\partial f}{\partial w_1} \qquad \frac{\partial g}{\partial f} \qquad \frac{\partial \hat{y}}{\partial g} \qquad \frac{\partial \mathcal{L}}{\partial \hat{y}}$$

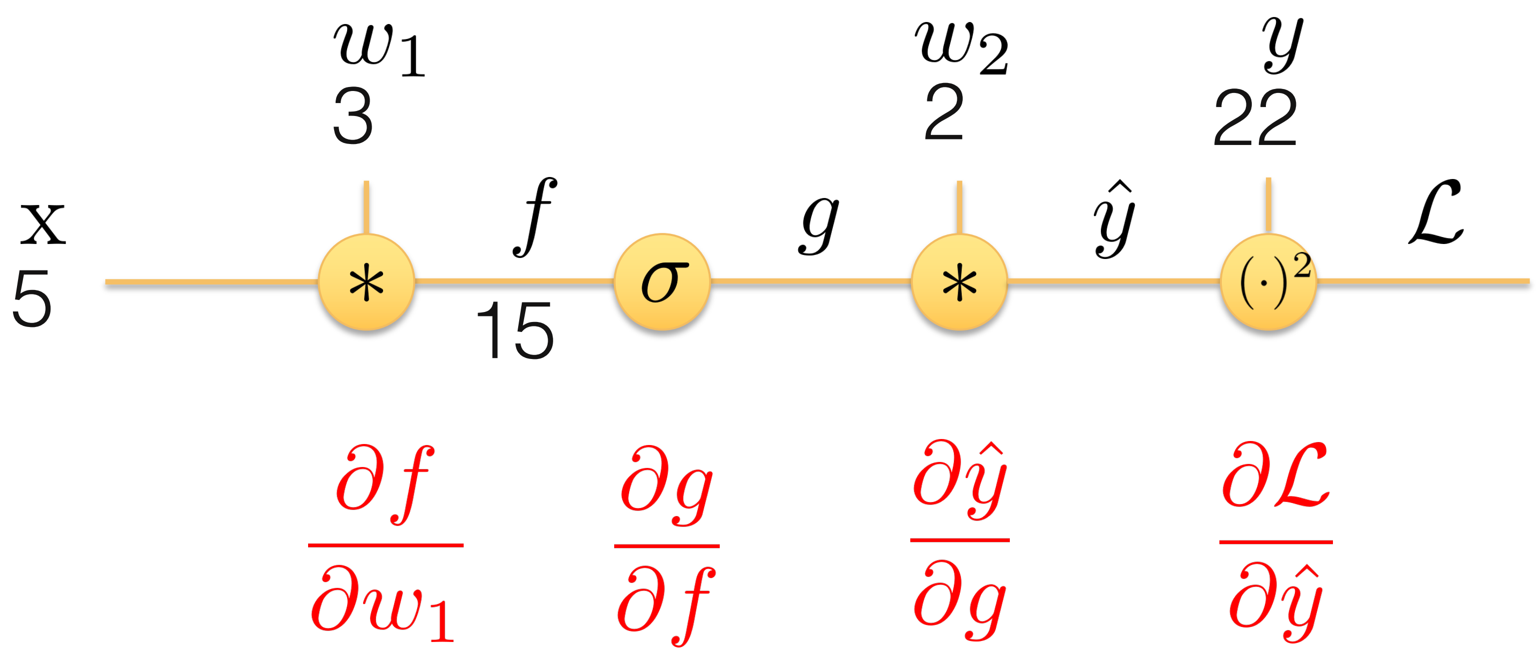
Autodiff Example

Let's plug in the values now…



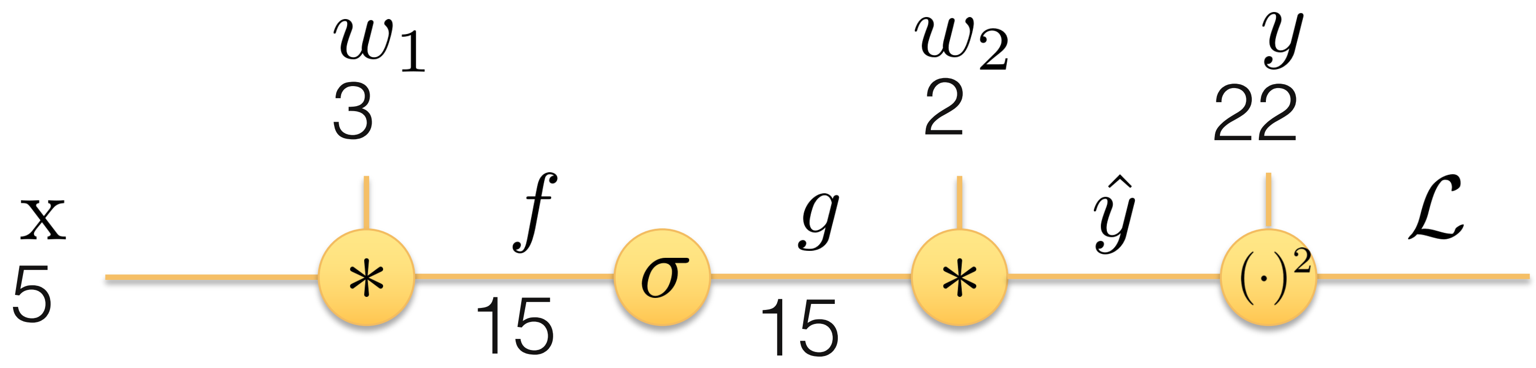
Autodiff Example

Let's plug in the values now…



Autodiff Example

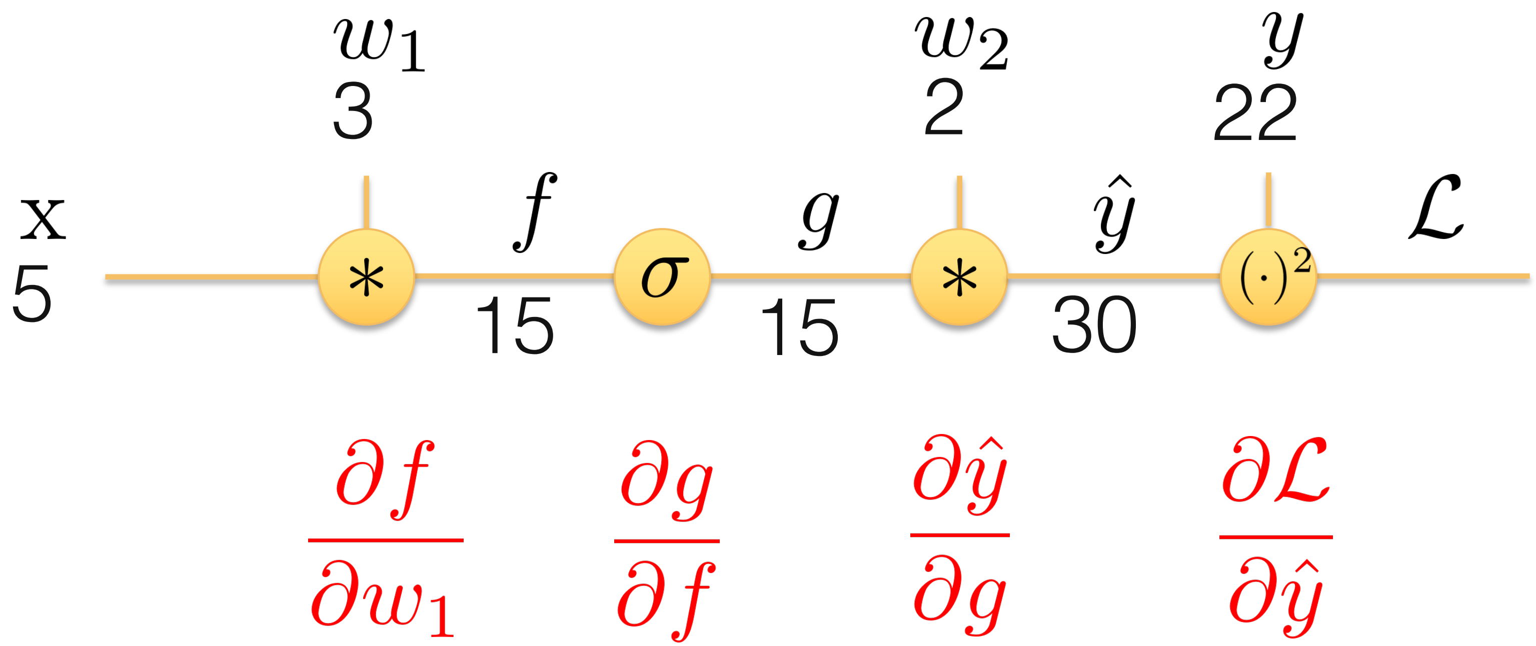
Let's plug in the values now…



$$\frac{\partial f}{\partial w_1} \qquad \frac{\partial g}{\partial f} \qquad \frac{\partial \hat{y}}{\partial g} \qquad \frac{\partial \mathcal{L}}{\partial \hat{y}}$$

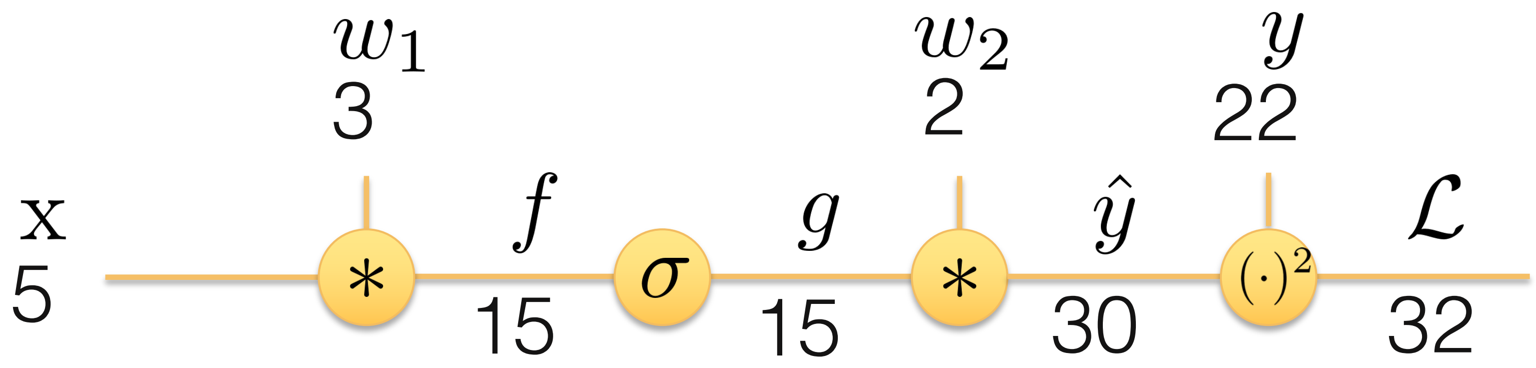
Autodiff Example

Let's plug in the values now…



Autodiff Example

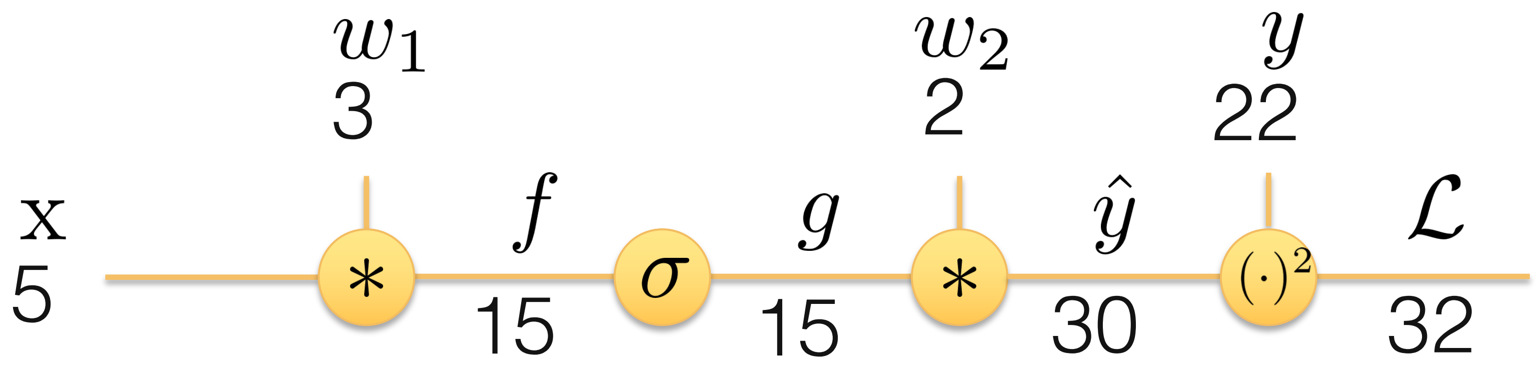
Let's plug in the values now…



$$\frac{\partial f}{\partial w_1} \qquad \frac{\partial g}{\partial f} \qquad \frac{\partial \hat{y}}{\partial g} \qquad \frac{\partial \mathcal{L}}{\partial \hat{y}}$$

Autodiff Example

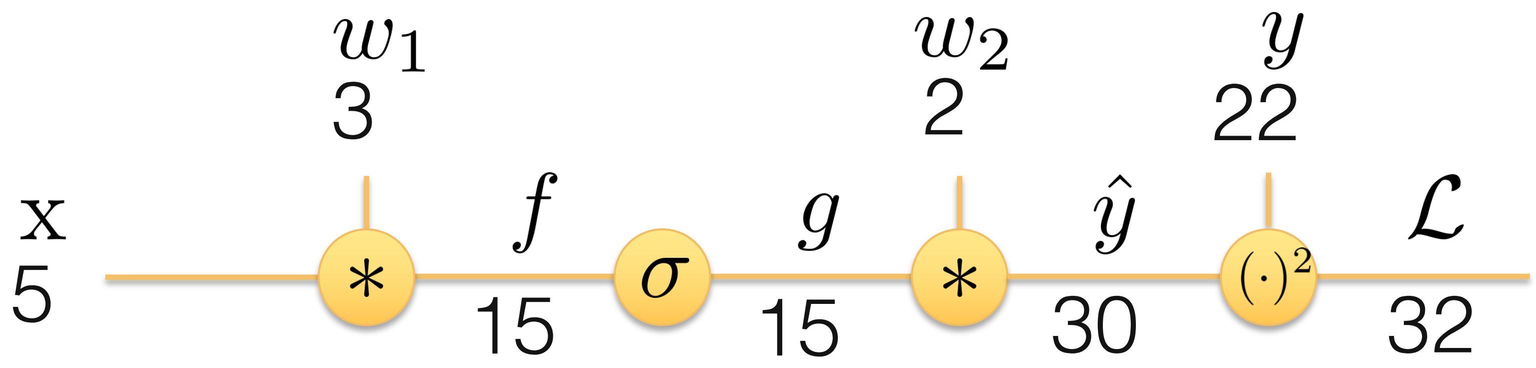
Let's plug in the values now…



$$\frac{\partial f}{\partial w_1} \qquad \frac{\partial g}{\partial f} \qquad \frac{\partial \hat{y}}{\partial g} \qquad \frac{\partial \mathcal{L}}{\partial \hat{y}} = \hat{y} - y$$

Autodiff Example

Let's plug in the values now…

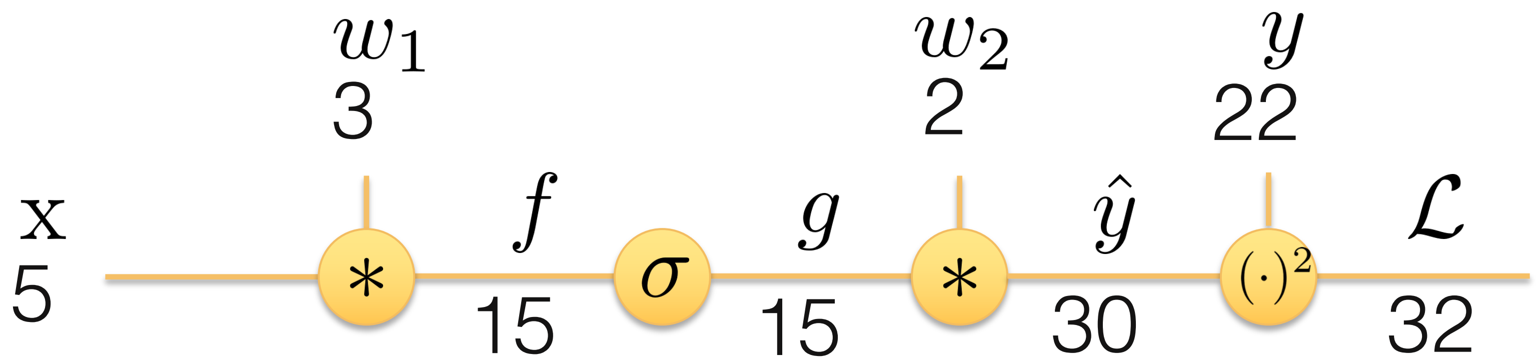


$$\frac{\partial f}{\partial w_1} \qquad \frac{\partial g}{\partial f} \qquad \frac{\partial \hat{y}}{\partial g} \qquad \frac{\partial \mathcal{L}}{\partial \hat{y}} = \hat{y} - y$$

8

Autodiff Example

Let's plug in the values now…



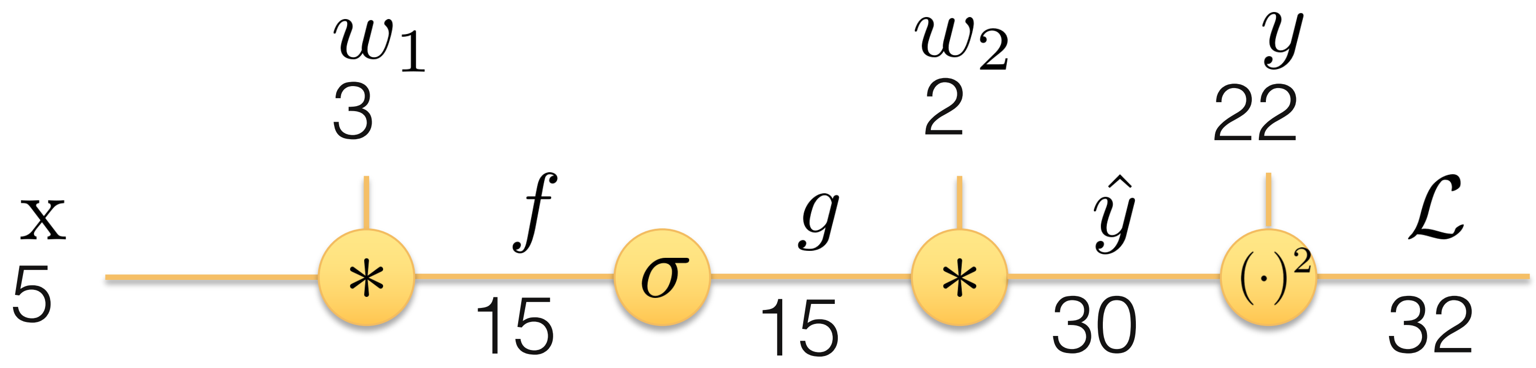
$$\frac{\partial f}{\partial w_1} \qquad \frac{\partial g}{\partial f} \qquad \frac{\partial \hat{y}}{\partial g} \qquad \frac{\partial \mathcal{L}}{\partial \hat{y}} = \hat{y} - y$$

2

8

Autodiff Example

Let's plug in the values now…

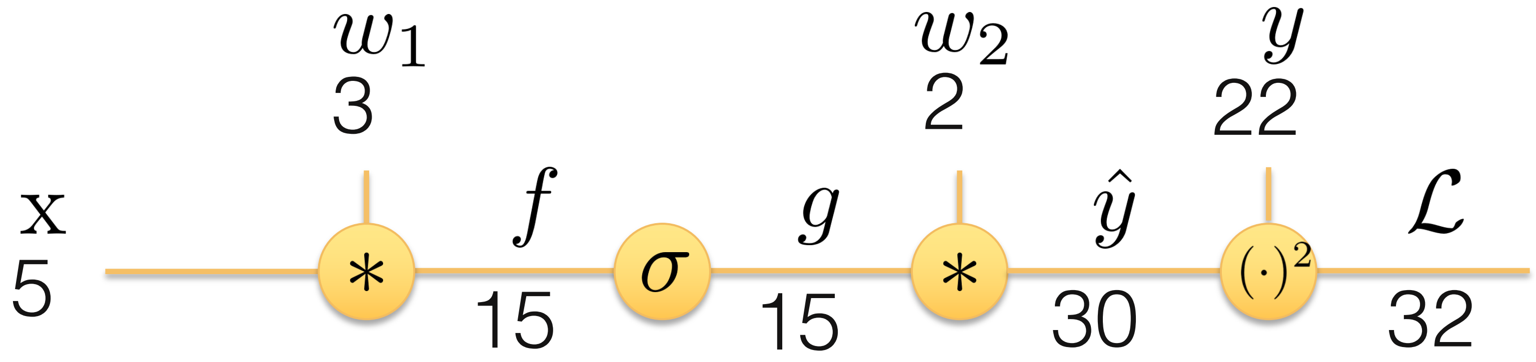


$$\frac{\partial f}{\partial w_1} \qquad \frac{\partial g}{\partial f} \qquad \frac{\partial \hat{y}}{\partial g} \qquad \frac{\partial \mathcal{L}}{\partial \hat{y}} = \hat{y} - y$$

1 2 8

Autodiff Example

Let's plug in the values now…

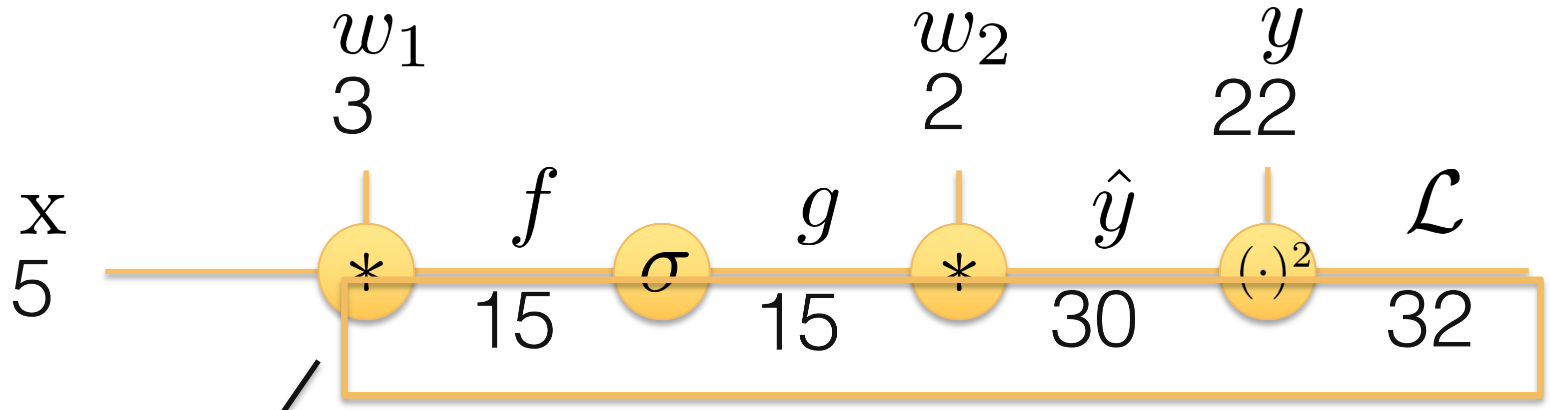


$$\frac{\partial f}{\partial w_1} \qquad \frac{\partial g}{\partial f} \qquad \frac{\partial \hat{y}}{\partial g} \qquad \frac{\partial \mathcal{L}}{\partial \hat{y}} = \hat{y} - y$$

5 1 2 8

Autodiff Example

What is backpropagation?



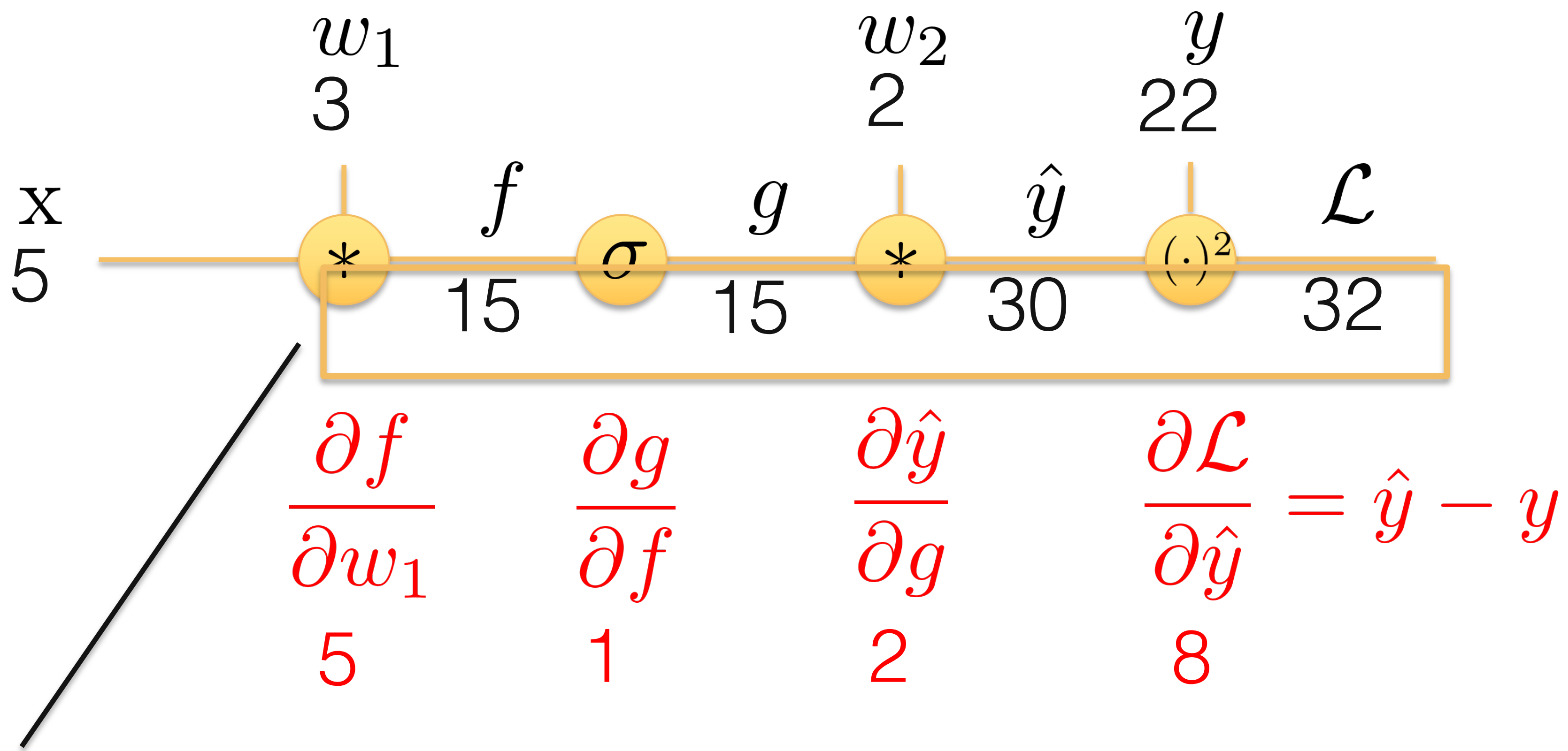
$$\frac{\partial f}{\partial w_1} \qquad \frac{\partial g}{\partial f} \qquad \frac{\partial \hat{y}}{\partial g} \qquad \frac{\partial \mathcal{L}}{\partial \hat{y}} = \hat{y} - y$$

5 1 2 8

Save these intermediate values during forward computation

Autodiff Example

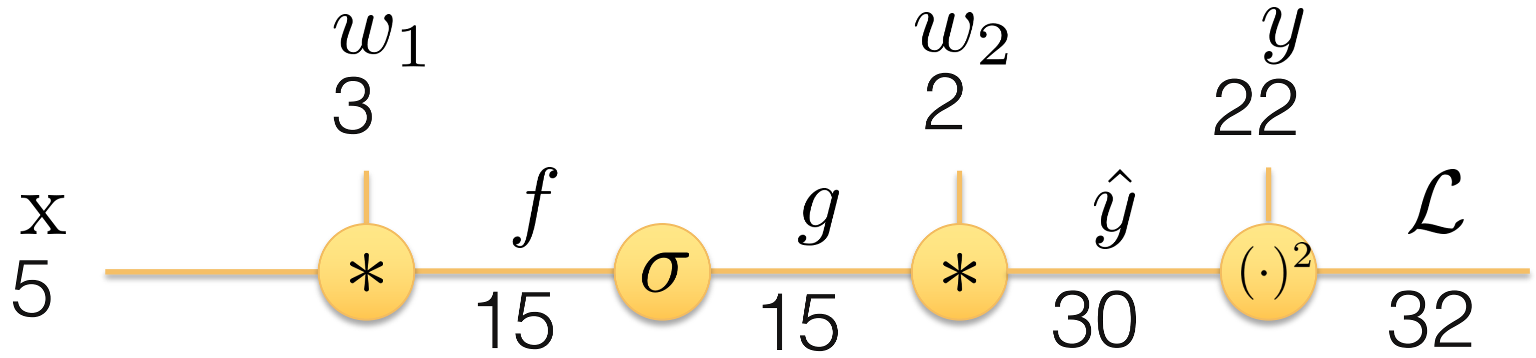
What is backpropagation?



Then we perform a “backward pass”

Autodiff Example

What is backpropagation?



$$\frac{\partial f}{\partial w_1}$$
5

$$\frac{\partial g}{\partial f}$$
1

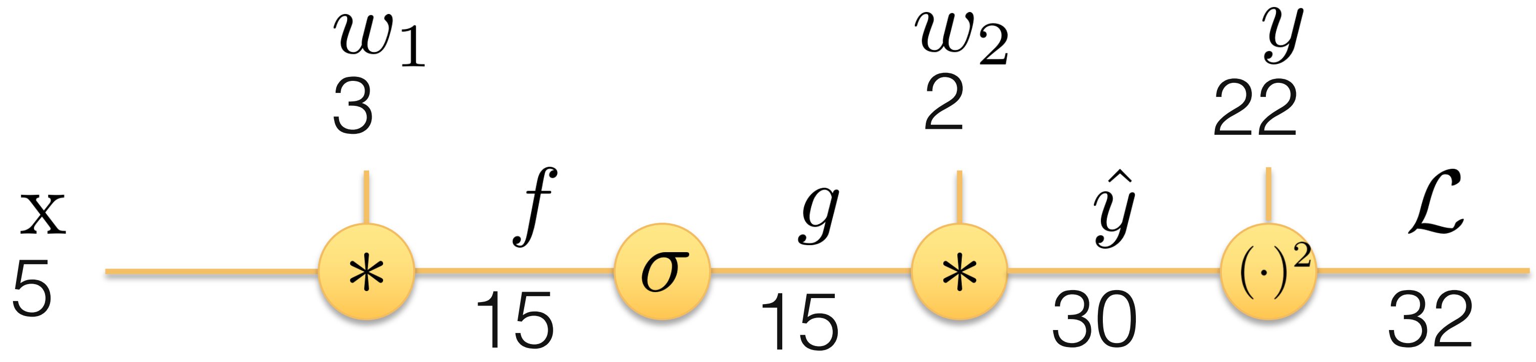
$$\frac{\partial \hat{y}}{\partial g}$$
2

$$\frac{\partial \mathcal{L}}{\partial g}$$

$$\frac{\partial \mathcal{L}}{\partial \hat{y}} = \hat{y} - y$$
8

Autodiff Example

What is backpropagation?



$$\frac{\partial f}{\partial w_1}$$
5

$$\frac{\partial g}{\partial f}$$
1

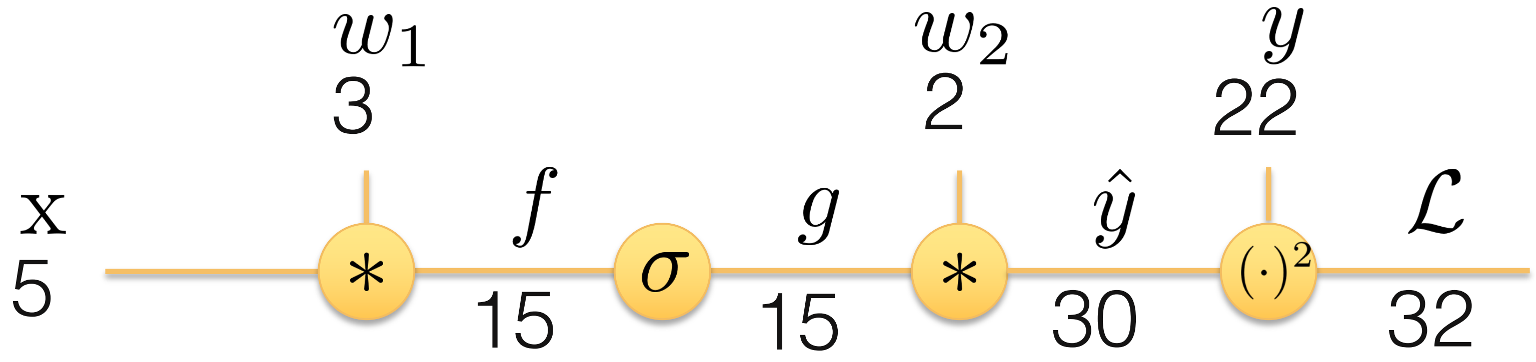
$$\frac{\partial \hat{y}}{\partial g}$$
2

$$\frac{\partial \mathcal{L}}{\partial \hat{y}} = \hat{y} - y$$
8

$$\frac{\partial \mathcal{L}}{\partial g}$$
16

Autodiff Example

What is backpropagation?



$$\frac{\partial f}{\partial w_1} \quad 5$$

$$\frac{\partial g}{\partial f} \quad 1$$

$$\frac{\partial \mathcal{L}}{\partial f}$$

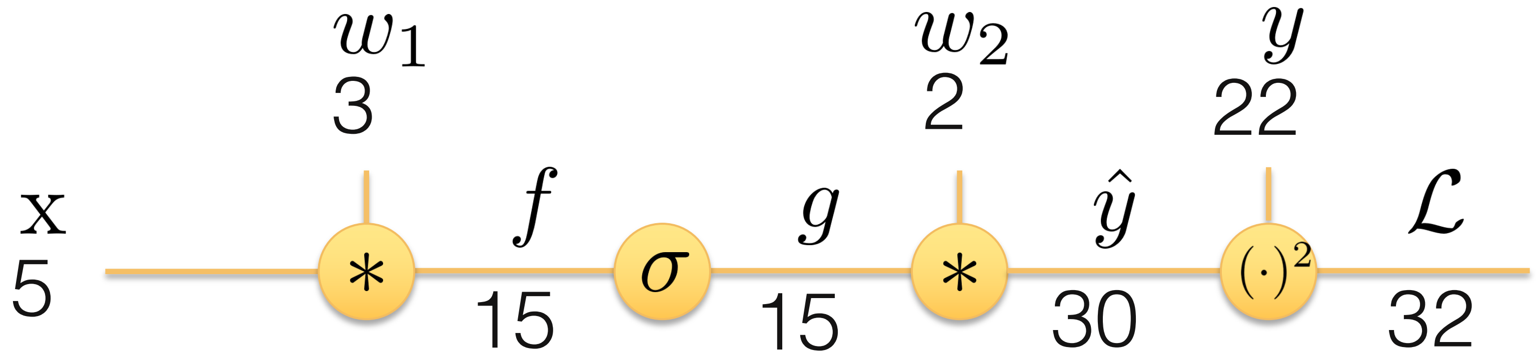
$$\frac{\partial \hat{y}}{\partial g} \quad 2$$

$$\frac{\partial \mathcal{L}}{\partial g} \quad 16$$

$$\frac{\partial \mathcal{L}}{\partial \hat{y}} = \hat{y} - y \quad 8$$

Autodiff Example

What is backpropagation?



$$\frac{\partial f}{\partial w_1}$$
5

$$\frac{\partial g}{\partial f}$$
1

$$\frac{\partial \hat{y}}{\partial g}$$
2

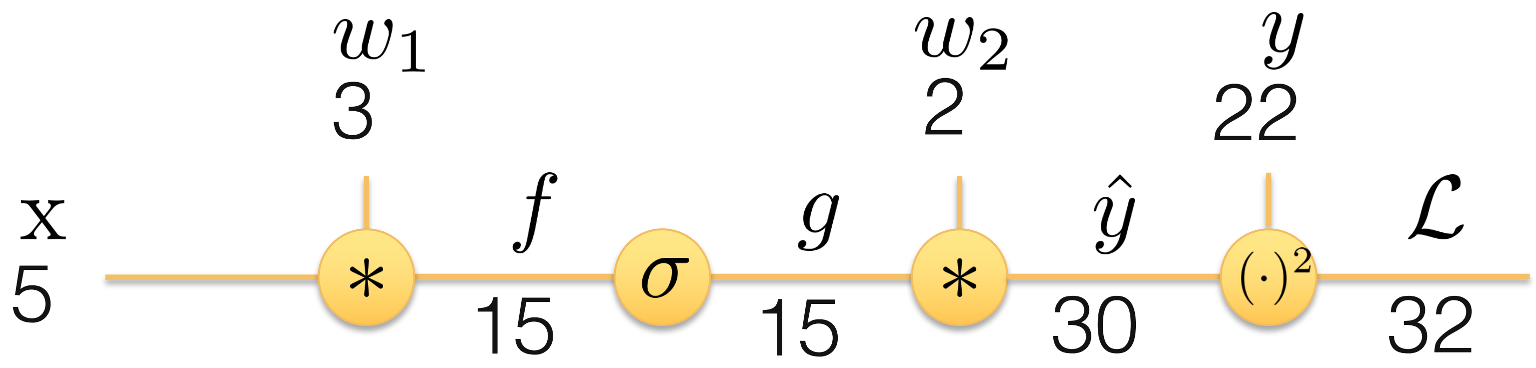
$$\frac{\partial \mathcal{L}}{\partial \hat{y}} = \hat{y} - y$$
8

$$\frac{\partial \mathcal{L}}{\partial f}$$
16

$$\frac{\partial \mathcal{L}}{\partial g}$$
16

Autodiff Example

What is backpropagation?



$$\frac{\partial f}{\partial w_1} \qquad \frac{\partial g}{\partial f} \qquad \frac{\partial \hat{y}}{\partial g} \qquad \frac{\partial \mathcal{L}}{\partial \hat{y}} = \hat{y} - y$$

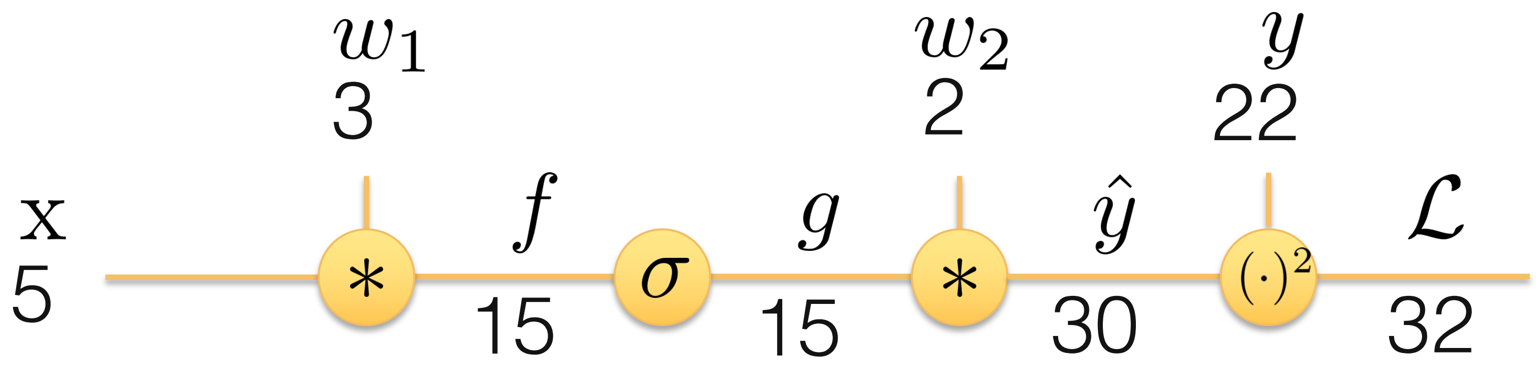
5   1   2   8

$$\frac{\partial \mathcal{L}}{\partial w_1} \qquad \frac{\partial \mathcal{L}}{\partial f} \qquad \frac{\partial \mathcal{L}}{\partial g}$$

16   16

Autodiff Example

What is backpropagation?



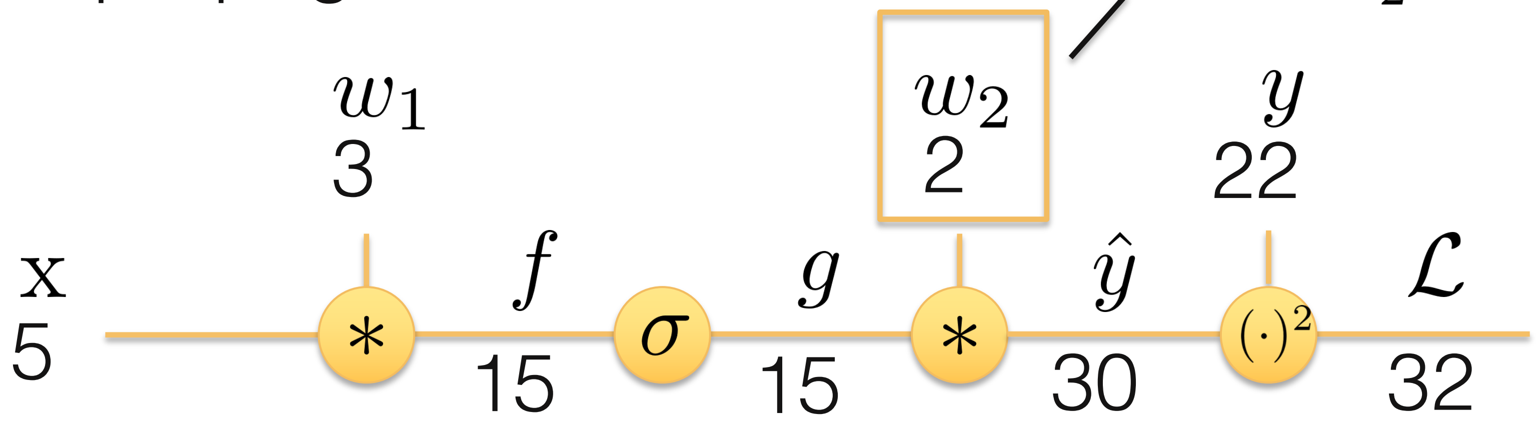
$$\frac{\partial f}{\partial w_1} \quad 5 \qquad \frac{\partial g}{\partial f} \quad 1 \qquad \frac{\partial \hat{y}}{\partial g} \quad 2 \qquad \frac{\partial \mathcal{L}}{\partial \hat{y}} = \hat{y} - y \quad 8$$

$$\frac{\partial \mathcal{L}}{\partial w_1} \quad 80 \qquad \frac{\partial \mathcal{L}}{\partial f} \quad 16 \qquad \frac{\partial \mathcal{L}}{\partial g} \quad 16$$

Autodiff Example

What is backpropagation?

What about $\frac{\partial \mathcal{L}}{\partial w_2}$?



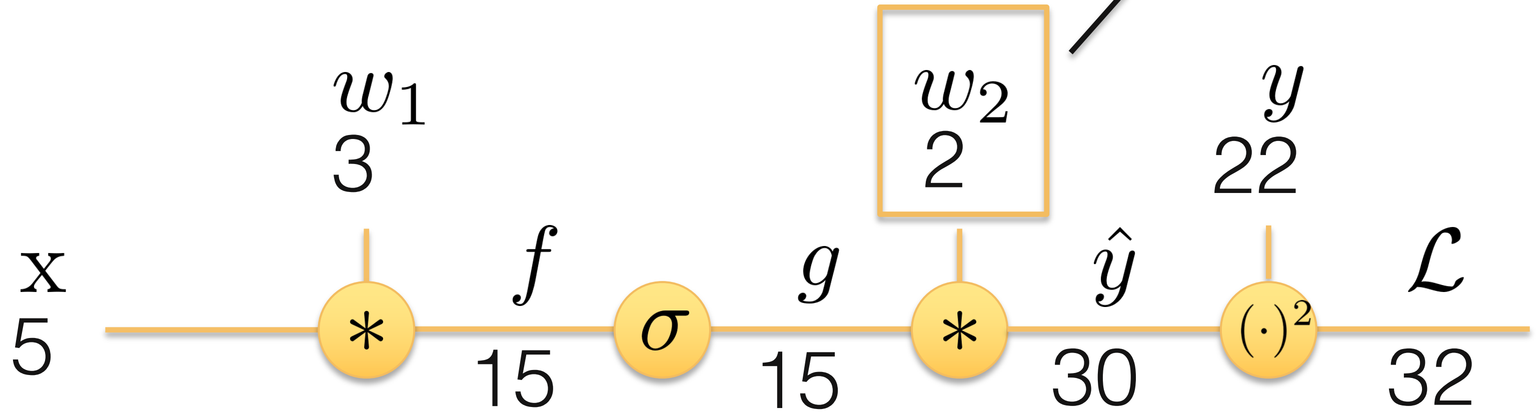
$\frac{\partial f}{\partial w_1}$ 5 $\quad \frac{\partial g}{\partial f}$ 1 $\quad \frac{\partial \hat{y}}{\partial g}$ 2 $\quad \frac{\partial \mathcal{L}}{\partial \hat{y}} = \hat{y} - y$ 8

$\frac{\partial \mathcal{L}}{\partial w_1}$ 80 $\quad \frac{\partial \mathcal{L}}{\partial f}$ 16 $\quad \frac{\partial \mathcal{L}}{\partial g}$ 16

Autodiff Example

What is backpropagation?

What about $\frac{\partial \mathcal{L}}{\partial w_2}$?



$\frac{\partial f}{\partial w_1}$ 5 | $\frac{\partial g}{\partial f}$ 1 | $\frac{\partial \hat{y}}{\partial g}$ 2

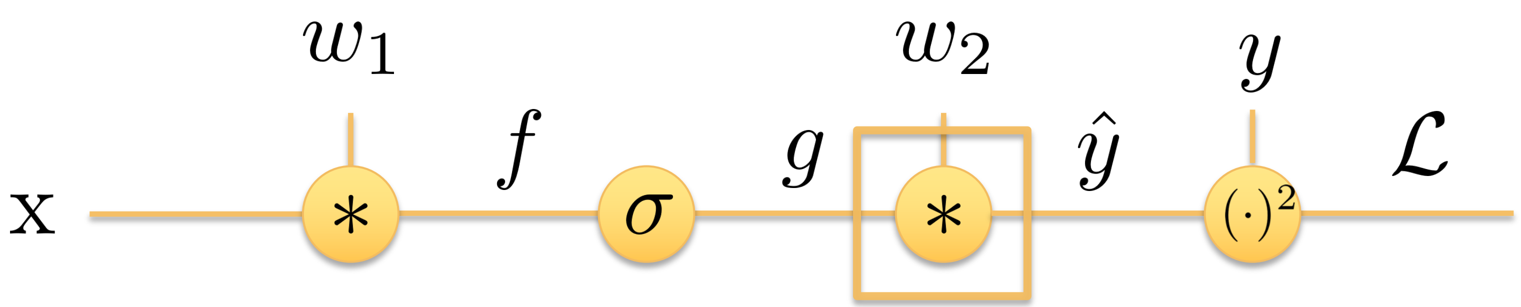
$\frac{\partial \mathcal{L}}{\partial w_1}$ 80 | $\frac{\partial \mathcal{L}}{\partial f}$ 16 | $\frac{\partial \mathcal{L}}{\partial g}$ 16

$$\boxed{\frac{\partial \mathcal{L}}{\partial \hat{y}}} = \hat{y} - y$$

We can re-use computation!

$$\frac{\partial \mathcal{L}}{\partial w_2} = \frac{\partial \hat{y}}{\partial w_2} \boxed{\frac{\partial \mathcal{L}}{\partial \hat{y}}}$$

Autodiff Example



PyTorch Code:

```
class Multiply(torch.autograd.Function):
  @staticmethod
  def forward(ctx, x, y):
    ctx.save_for_backward(x, y)
    z = x * y
    return z
  @staticmethod
  def backward(ctx, grad_z):
    x, y = ctx.saved_tensors
    grad_x = y * grad_z    # dz/dx * dL/dz
    grad_y = x * grad_z    # dz/dy * dL/dz
    return grad_x, grad_y
```

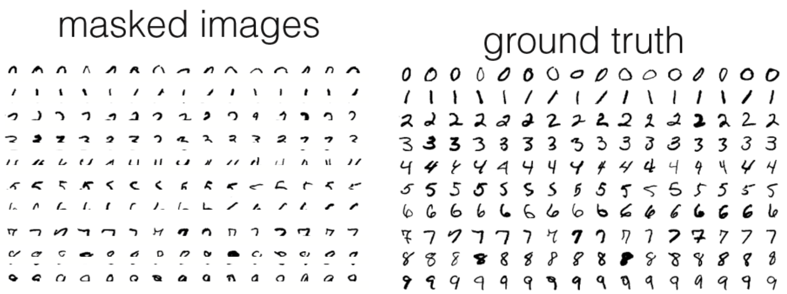
Need to stash some values for use in backward

Upstream gradient

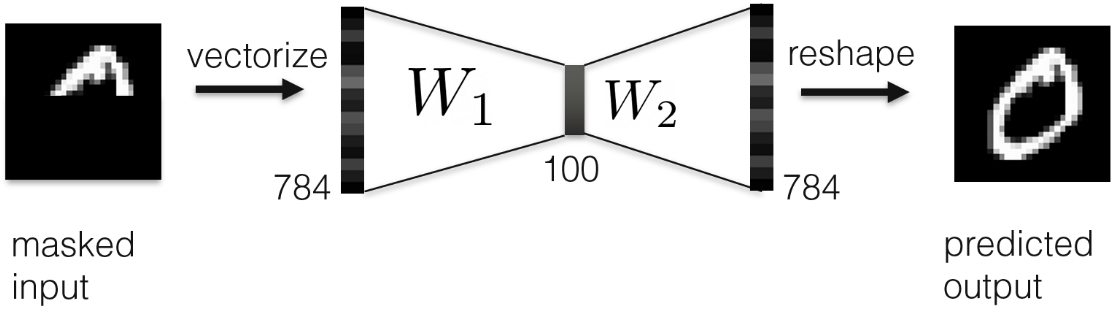
Multiply upstream and local gradients

Image Inpainting Training Loop

1. Sample batch of images from dataset



2. Run forward pass to calculate network output for each image



3. Run backward pass to calculate gradients with backpropagation

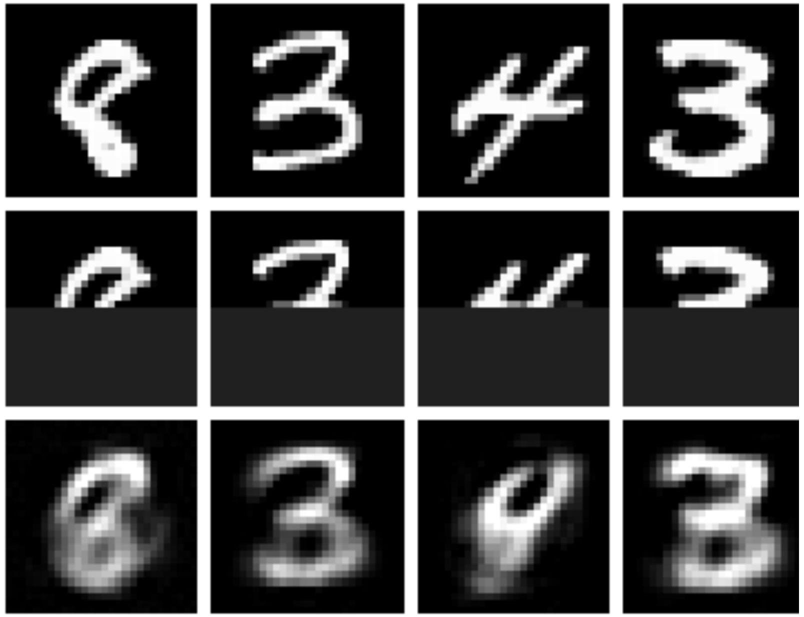
4. Update parameters with stochastic gradient descent

4. Update parameters with stochastic gradient descent

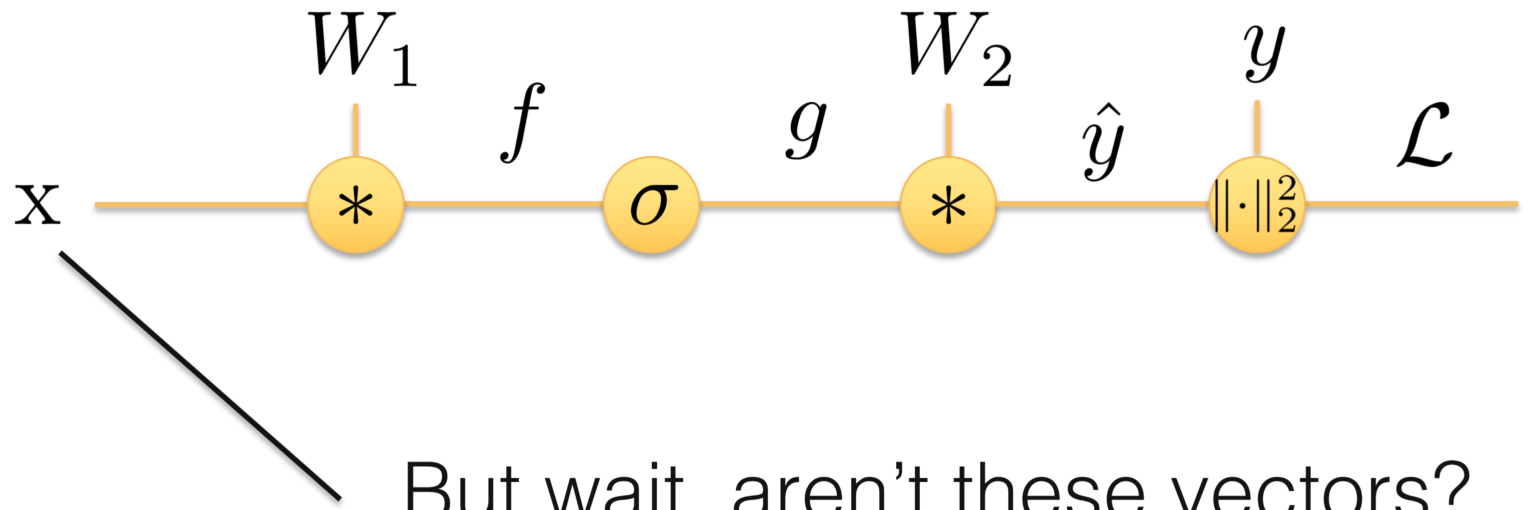
$$\mathcal{L}_\theta = \|y - \hat{y}\|_2^2$$

$$W_2^{(k+1)} = W_2^{(k)} - \alpha \frac{\partial \mathcal{L}}{\partial W_2}$$

$$W_1^{(k+1)} = W_1^{(k)} - \alpha \frac{\partial \mathcal{L}}{\partial W_1}$$



Vector Differentiation



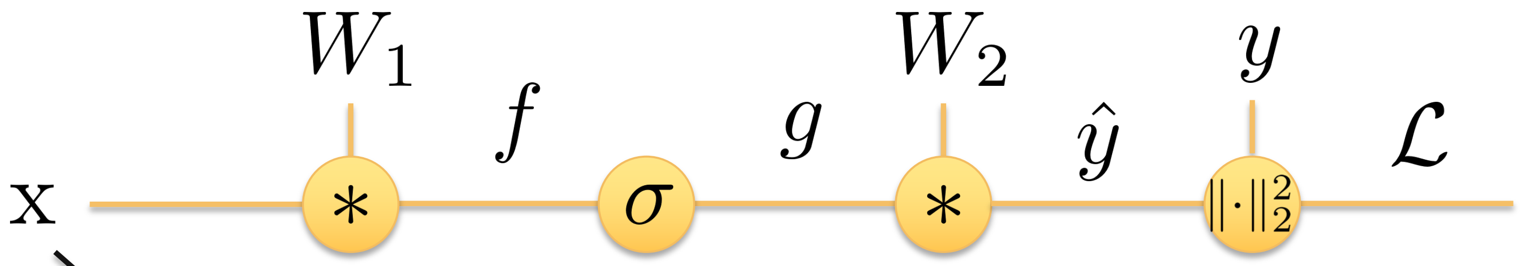
But wait, aren't these vectors?
(see supplemental slides for vector differentiation)

Next Time

- Embedded ethics lecture!

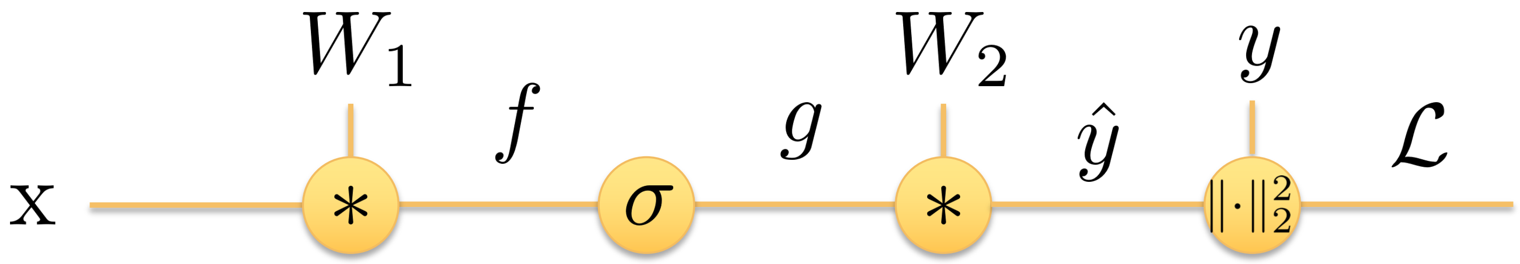
Supplemental Slides

Vector Differentiation



But wait, aren't these vectors?
(see supplemental slides for vector differentiation)

Vector Differentiation



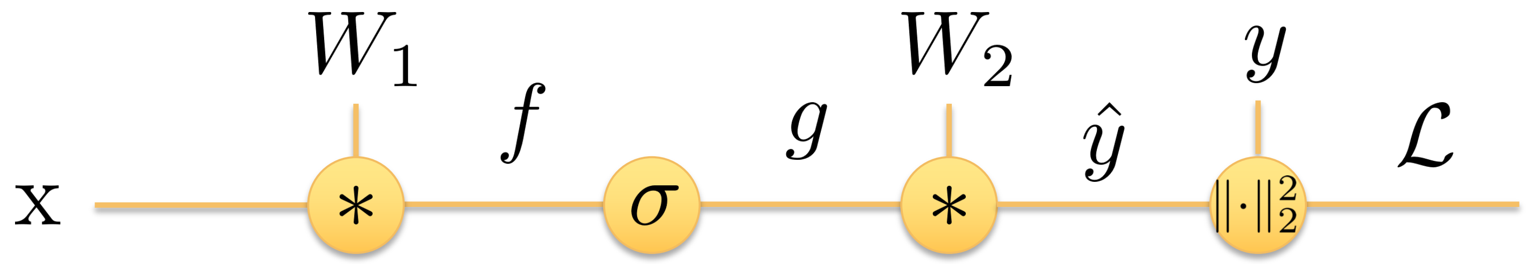
Recap: vector differentiation

Scalar by Scalar

$$x, y \in \mathbb{R}$$

$$\frac{\partial y}{\partial x} \in \mathbb{R}$$

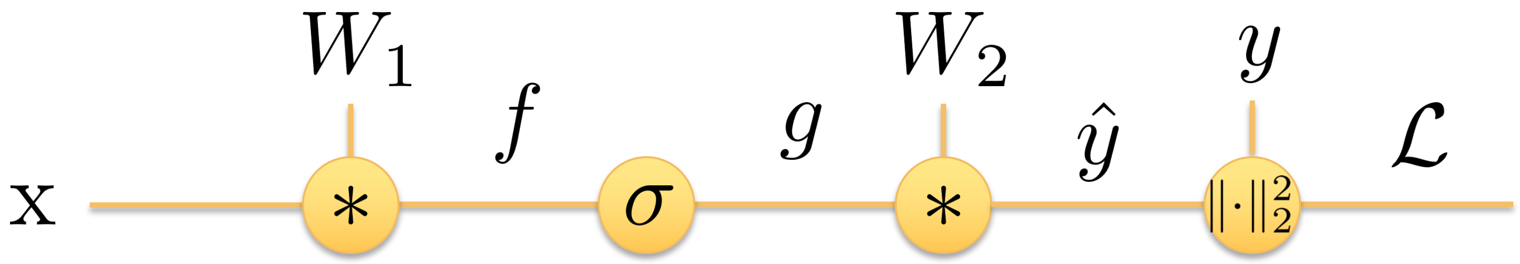
Vector Differentiation



Recap: vector differentiation

Scalar by Scalar	Scalar by Vector
$x, y \in \mathbb{R}$	$x \in \mathbb{R}^N, y \in \mathbb{R}$
$\dfrac{\partial y}{\partial x} \in \mathbb{R}$	$\dfrac{\partial y}{\partial x} \in \mathbb{R}^N$

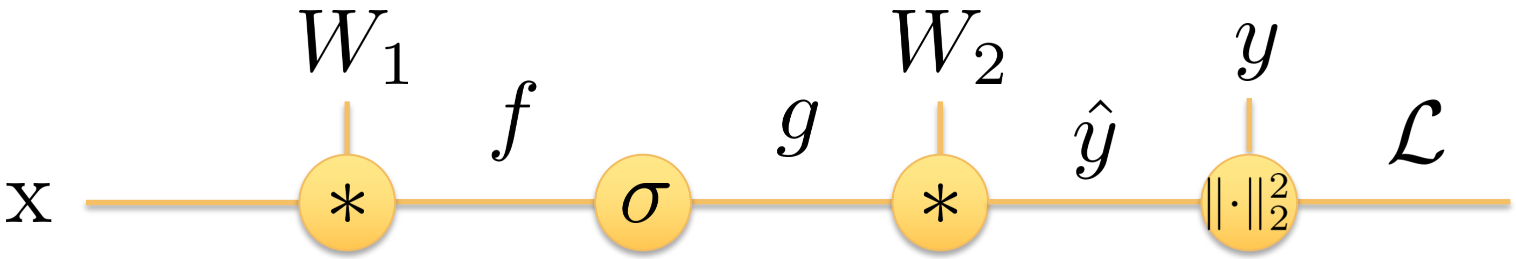
Vector Differentiation



Recap: vector differentiation

Scalar by Scalar	Scalar by Vector	Vector by Vector
$x, y \in \mathbb{R}$	$x \in \mathbb{R}^N, y \in \mathbb{R}$	$x \in \mathbb{R}^N, y \in \mathbb{R}^M$
$\frac{\partial y}{\partial x} \in \mathbb{R}$	$\frac{\partial y}{\partial x} \in \mathbb{R}^N$	$\frac{\partial y}{\partial x} \in \mathbb{R}^{N \times M}$

Recap: vector differentiation



Example 1: matrix multiply

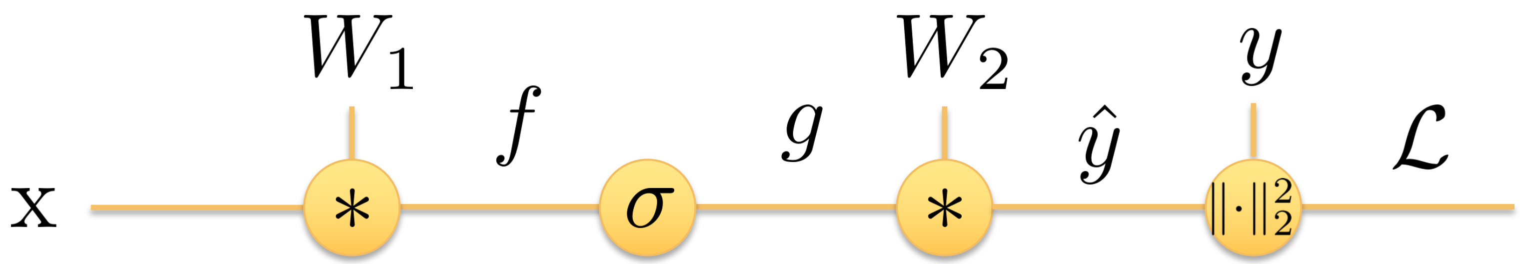
$$\frac{\partial \hat{y}}{\partial g} = \frac{\partial}{\partial g} W_2 g$$

$$W_2 \in \mathbb{R}^{M \times N}$$

$$g \in \mathbb{R}^{N}$$

$$\frac{\partial \hat{y}}{\partial g} \in \mathbb{R}^{N \times M}$$

Recap: vector differentiation



Example 1: matrix multiply

$$\frac{\partial \hat{y}}{\partial g} = \frac{\partial}{\partial g} W_2 g$$

$$W_2 \in \mathbb{R}^{M \times N}$$

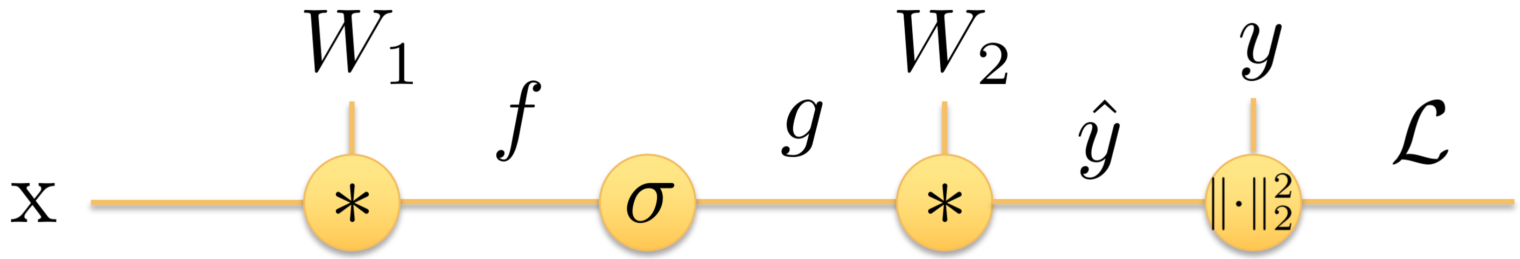
$$g \in \mathbb{R}^{N}$$

$$\frac{\partial \hat{y}}{\partial g} \in \mathbb{R}^{N \times M}$$

$$\frac{\partial}{\partial g} W_2 g$$

$$= \frac{\partial}{\partial g} \begin{bmatrix} w_{11}g_1 + \cdots + w_{1n}g_n \\ \vdots \quad \ddots \quad \vdots \\ w_{m1}g_1 + \cdots + w_{mn}g_n \end{bmatrix}$$

Recap: vector differentiation



$$\frac{\partial \hat{y}}{\partial g} = \begin{bmatrix} \frac{\partial \hat{y}_1}{\partial g_1} & \cdots & \frac{\partial \hat{y}_m}{\partial g_1} \\ \vdots & \ddots & \vdots \\ \frac{\partial \hat{y}_1}{\partial g_n} & \cdots & \frac{\partial \hat{y}_m}{\partial g_n} \end{bmatrix}$$

Example 1: matrix multiply

$$\frac{\partial \hat{y}}{\partial g} = \frac{\partial}{\partial g} W_2 g$$

$$W_2 \in \mathbb{R}^{M \times N}$$

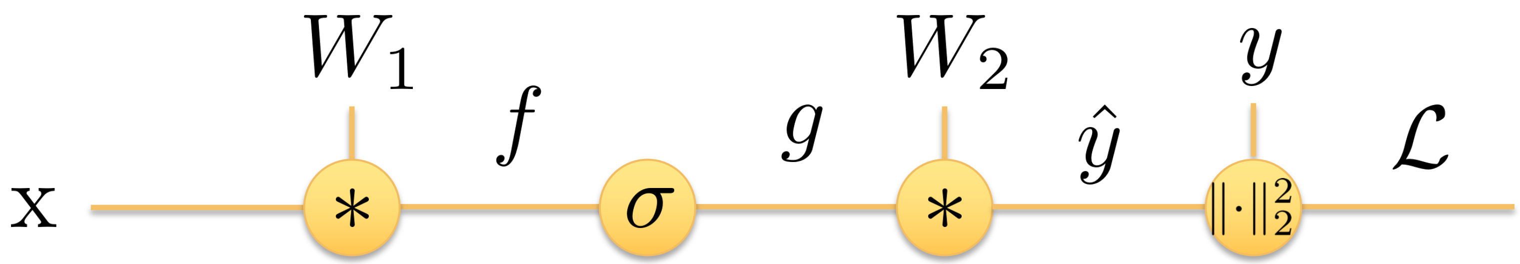
$$g \in \mathbb{R}^{N}$$

$$\frac{\partial \hat{y}}{\partial g} \in \mathbb{R}^{N \times M}$$

$$\frac{\partial}{\partial g} W_2 g$$

$$= \frac{\partial}{\partial g} \begin{bmatrix} w_{11} g_1 + \cdots + w_{1n} g_n \\ \vdots \quad \ddots \quad \vdots \\ w_{m1} g_1 + \cdots + w_{mn} g_n \end{bmatrix} = \begin{bmatrix} w_{11} & \cdots & w_{m1} \\ \vdots & \ddots & \vdots \\ w_{1n} & \cdots & w_{mn} \end{bmatrix}$$

Recap: vector differentiation



$$\frac{\partial \hat{y}}{\partial g} = \begin{bmatrix} \frac{\partial \hat{y}_1}{\partial g_1} & \cdots & \frac{\partial \hat{y}_n}{\partial g_1} \\ \vdots & \ddots & \vdots \\ \frac{\partial \hat{y}_1}{\partial g_n} & \cdots & \frac{\partial \hat{y}_n}{\partial g_n} \end{bmatrix}$$

Example 1: matrix multiply

$$\frac{\partial \hat{y}}{\partial g} = \frac{\partial}{\partial g} W_2 g$$

$$W_2 \in \mathbb{R}^{M \times N}$$

$$g \in \mathbb{R}^N$$

$$\frac{\partial \hat{y}}{\partial g} \in \mathbb{R}^{N \times M}$$

$$\frac{\partial}{\partial g} W_2 g$$

$$= \frac{\partial}{\partial g} \begin{bmatrix} w_{11} g_1 + \cdots + w_{1n} g_n \\ \vdots \quad \ddots \quad \vdots \\ w_{n1} g_1 + \cdots + w_{nn} g_n \end{bmatrix} = \begin{bmatrix} w_{11} & \cdots & w_{n1} \\ \vdots & \ddots & \vdots \\ w_{1n} & \cdots & w_{nn} \end{bmatrix}$$

$$= \boxed{W_2^T}$$

Recap: vector differentiation

Example 2: elementwise functions

$$h = f \odot g$$

$$f \in \mathbb{R}^N$$

$$g \in \mathbb{R}^N$$

$$\frac{\partial h}{\partial f} \in \mathbb{R}^{N \times N}$$

Recap: vector differentiation

Example 2: elementwise functions

$$\frac{\partial h}{\partial f} = \begin{bmatrix} \frac{\partial h_1}{\partial f_1} & \cdots & \frac{\partial h_n}{\partial f_1} \\ \vdots & \ddots & \vdots \\ \frac{\partial h_1}{\partial f_n} & \cdots & \frac{\partial h_n}{\partial f_n} \end{bmatrix}$$

$$h = f \odot g$$

$$f \in \mathbb{R}^N$$

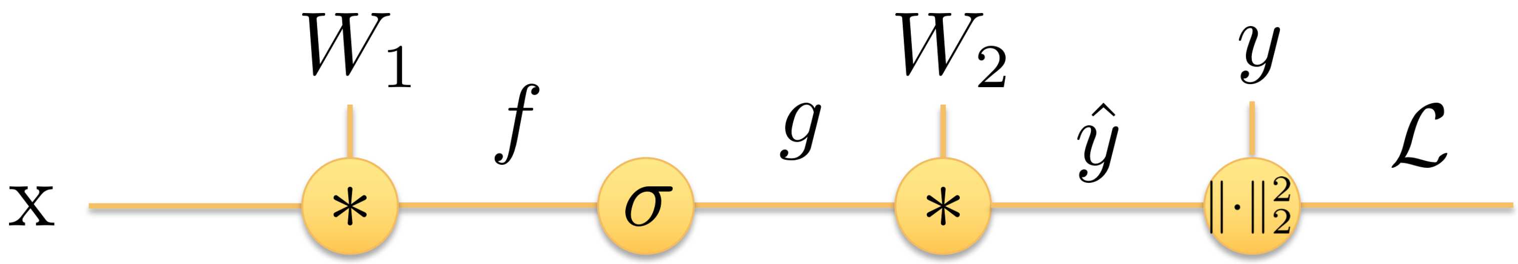
$$g \in \mathbb{R}^N$$

$$\frac{\partial h}{\partial f} \in \mathbb{R}^{N \times N}$$

$$\frac{\partial h}{\partial f} = \begin{bmatrix} g_1 & & 0 \\ & \ddots & \\ 0 & & g_n \end{bmatrix} = \mathrm{diag}(g)$$

Recap: vector differentiation

Final hint: dimensions should always match up!



$$\frac{\partial \mathcal{L}}{\partial W_1} = \frac{\partial f}{\partial W_1} \frac{\partial g}{\partial f} \frac{\partial \hat{y}}{\partial g} \frac{\partial \mathcal{L}}{\partial \hat{y}}$$

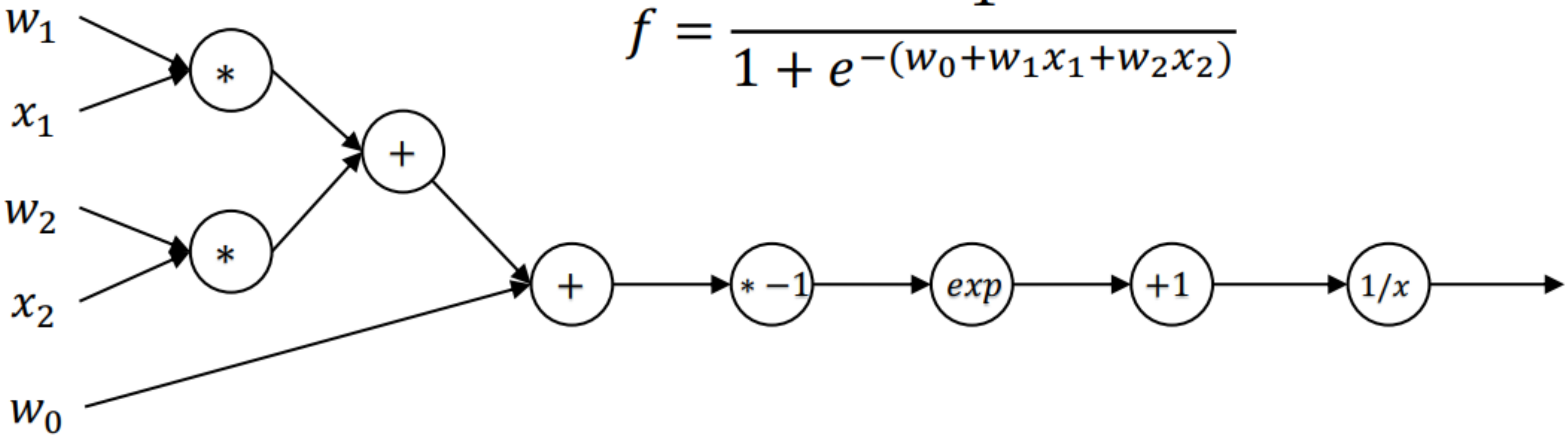
You should be able to calculate derivatives of each of these terms and then perform matrix multiplications without issues

Extra backpropagation example (adapted from Stanford CS231n)

$$f = \frac{1}{1 + e^{-(w_0 + w_1 x_1 + w_2 x_2)}}$$

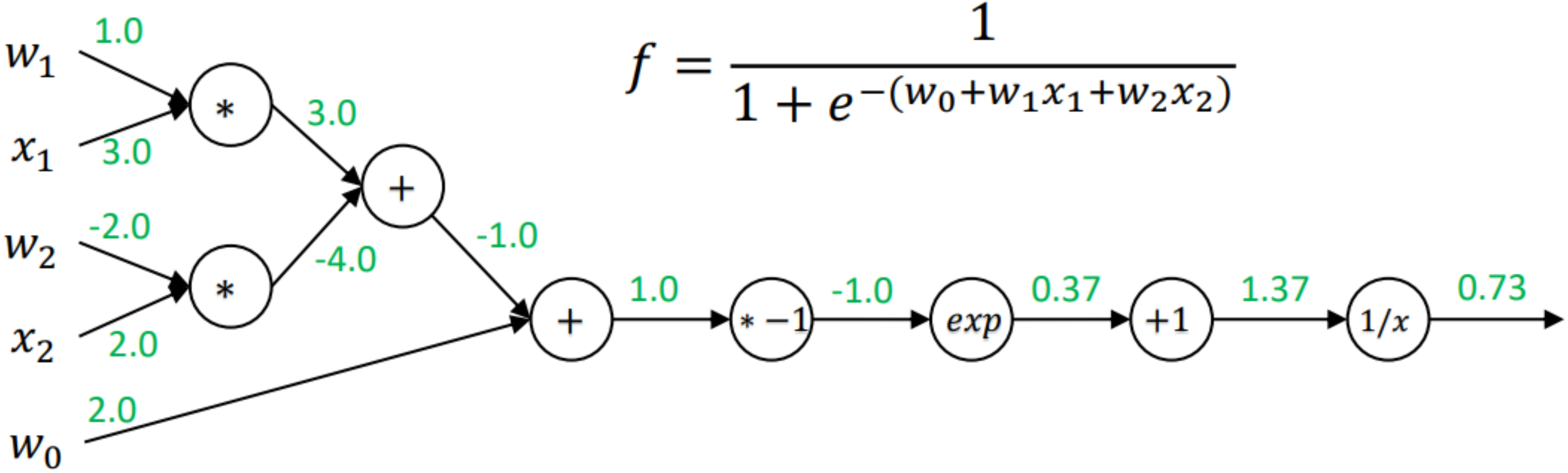
Extra backpropagation example

$$f = \frac{1}{1 + e^{-(w_0 + w_1 x_1 + w_2 x_2)}}$$



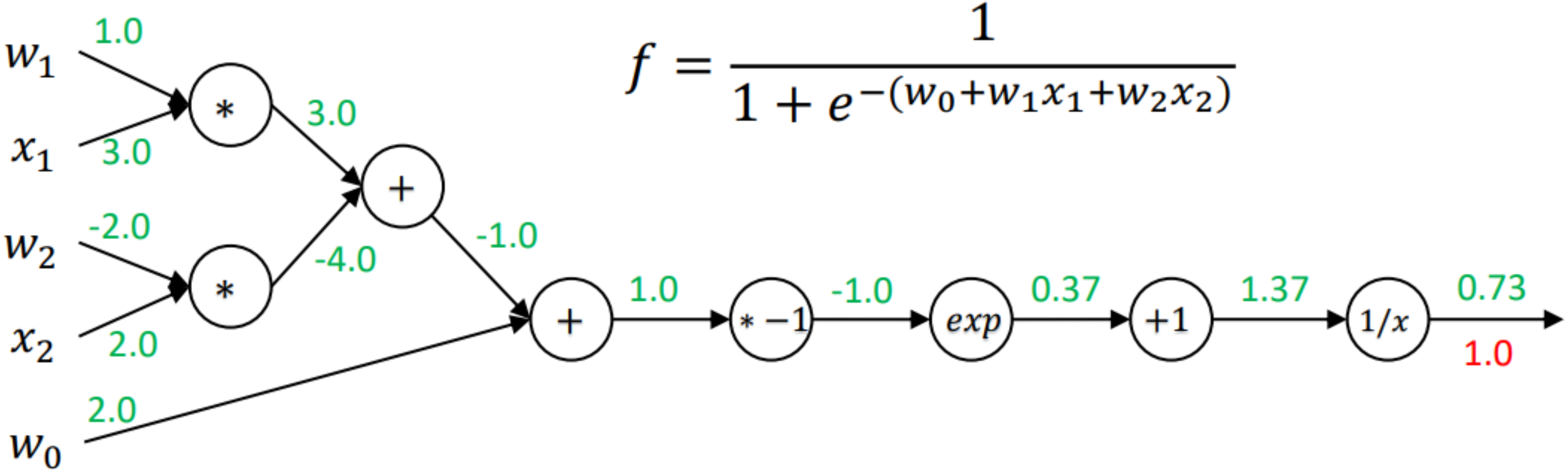
Extra backpropagation example

$$f = \frac{1}{1 + e^{-(w_0 + w_1 x_1 + w_2 x_2)}}$$



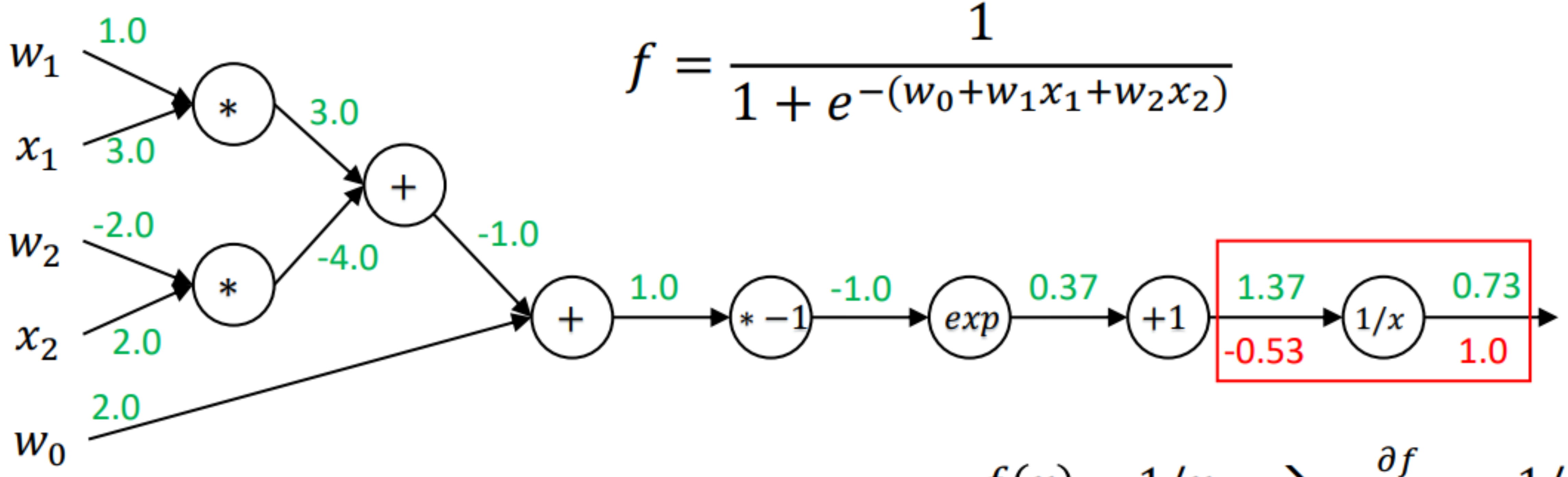
Extra backpropagation example

$$f = \frac{1}{1 + e^{-(w_0 + w_1 x_1 + w_2 x_2)}}$$



Extra backpropagation example

$$f = \frac{1}{1 + e^{-(w_0 + w_1 x_1 + w_2 x_2)}}$$

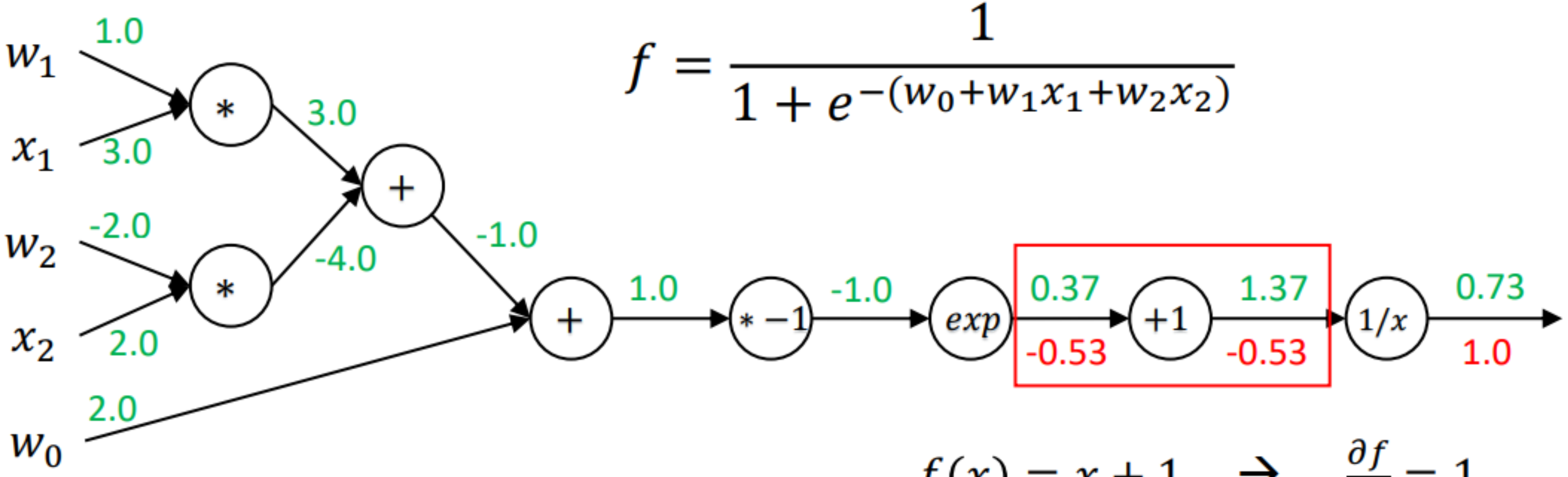


$$f(x) = 1/x \quad \rightarrow \quad \frac{\partial f}{\partial x} = -1/x^2$$

$$\frac{\partial J}{\partial x} = \frac{\partial J}{\partial f}\frac{\partial f}{\partial x} = -1/x^2$$

Extra backpropagation example

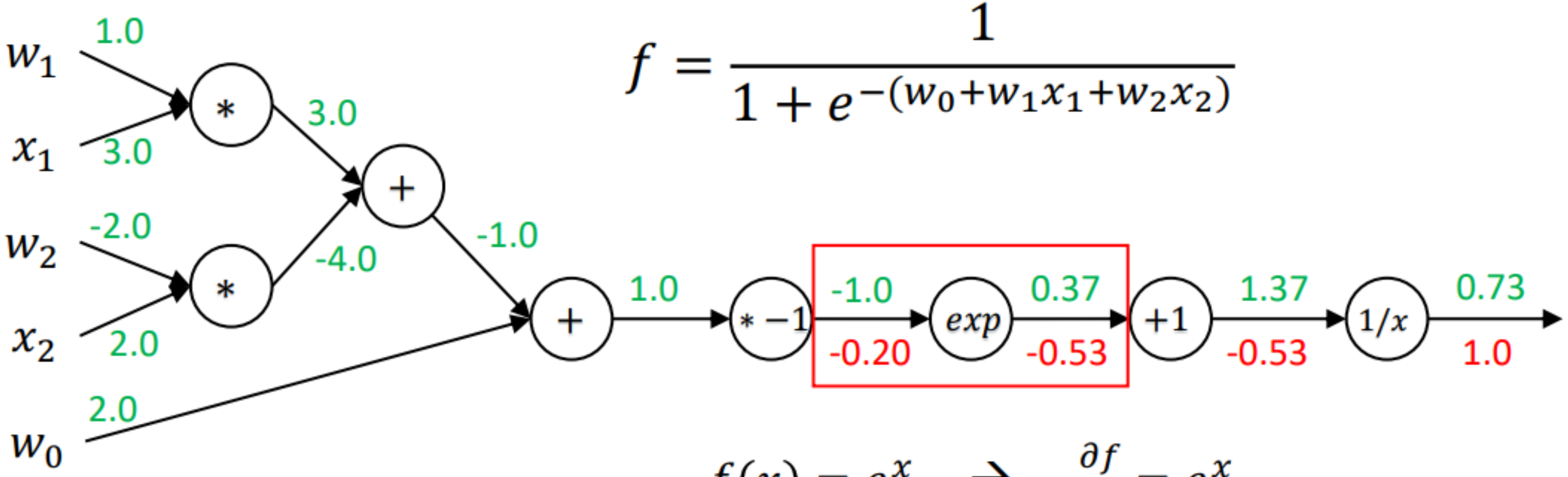
$$f = \frac{1}{1 + e^{-(w_0 + w_1 x_1 + w_2 x_2)}}$$



$$f(x) = x + 1 \quad \rightarrow \quad \frac{\partial f}{\partial x} = 1$$

Extra backpropagation example

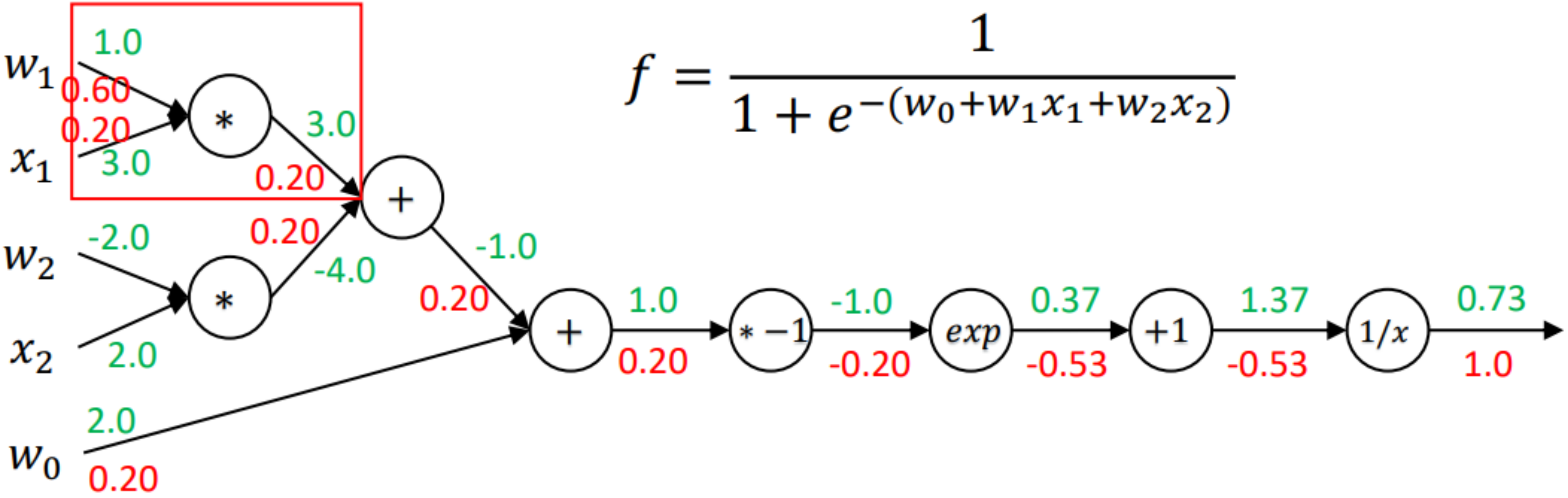
$$f = \frac{1}{1 + e^{-(w_0 + w_1 x_1 + w_2 x_2)}}$$



$$f(x) = e^x \quad \rightarrow \quad \frac{\partial f}{\partial x} = e^x$$

$$\frac{\partial J}{\partial x} = \frac{\partial J}{\partial f}\frac{\partial f}{\partial x} = \frac{\partial J}{\partial f} \cdot e^x$$

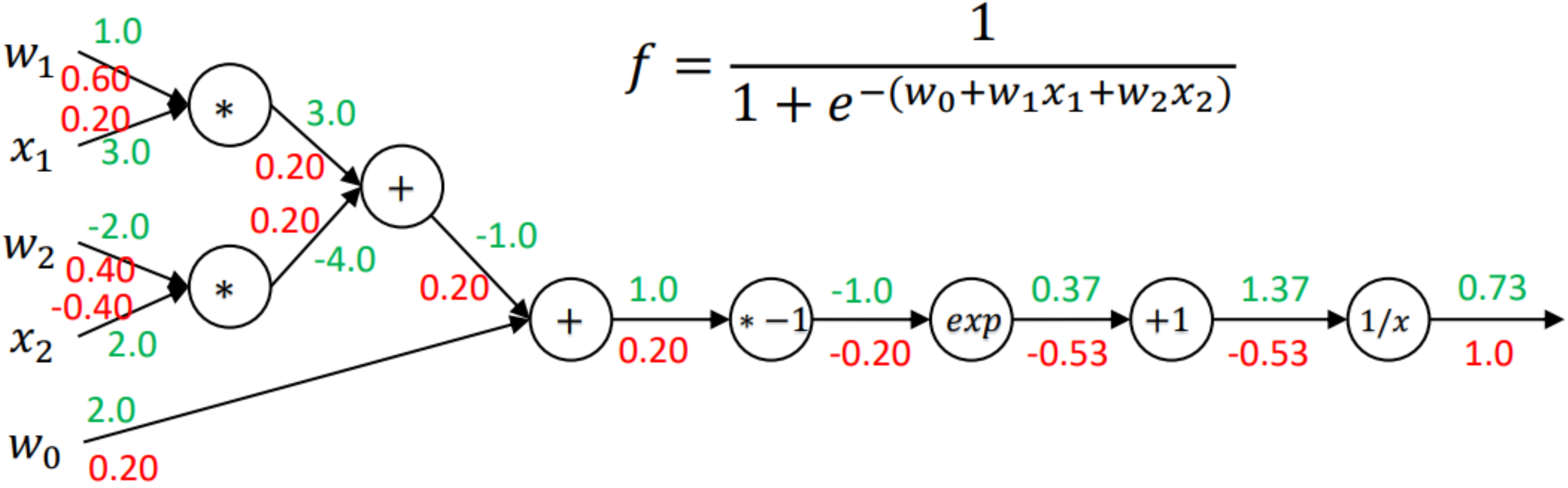
Extra backpropagation example



$$f = \frac{1}{1 + e^{-(w_0 + w_1 x_1 + w_2 x_2)}}$$

$$f(x, w) = xw \quad \rightarrow \quad \frac{\partial f}{\partial x} = w, \; \frac{\partial f}{\partial w} = x$$

Extra backpropagation example



$$f = \frac{1}{1 + e^{-(w_0 + w_1 x_1 + w_2 x_2)}}$$