Carnegie Mellon University
Heinz College

# 94-775 Unstructured Data Analytics

## Lecture 12: Wrap up neural net basics; brief overview of word embeddings; image analysis with convolutional neural nets

Nearly all slides by George H. Chen
with one slide by Phillip Isola

# Outline

Last few lectures teach fundamental concepts underlying some state-of-the-art technologies currently used in unstructured data analysis

- Today:
  - Wrap up neural net & deep learning basics
  - High-level idea of word embeddings

You'll see some practical examples in tomorrow's recitation!

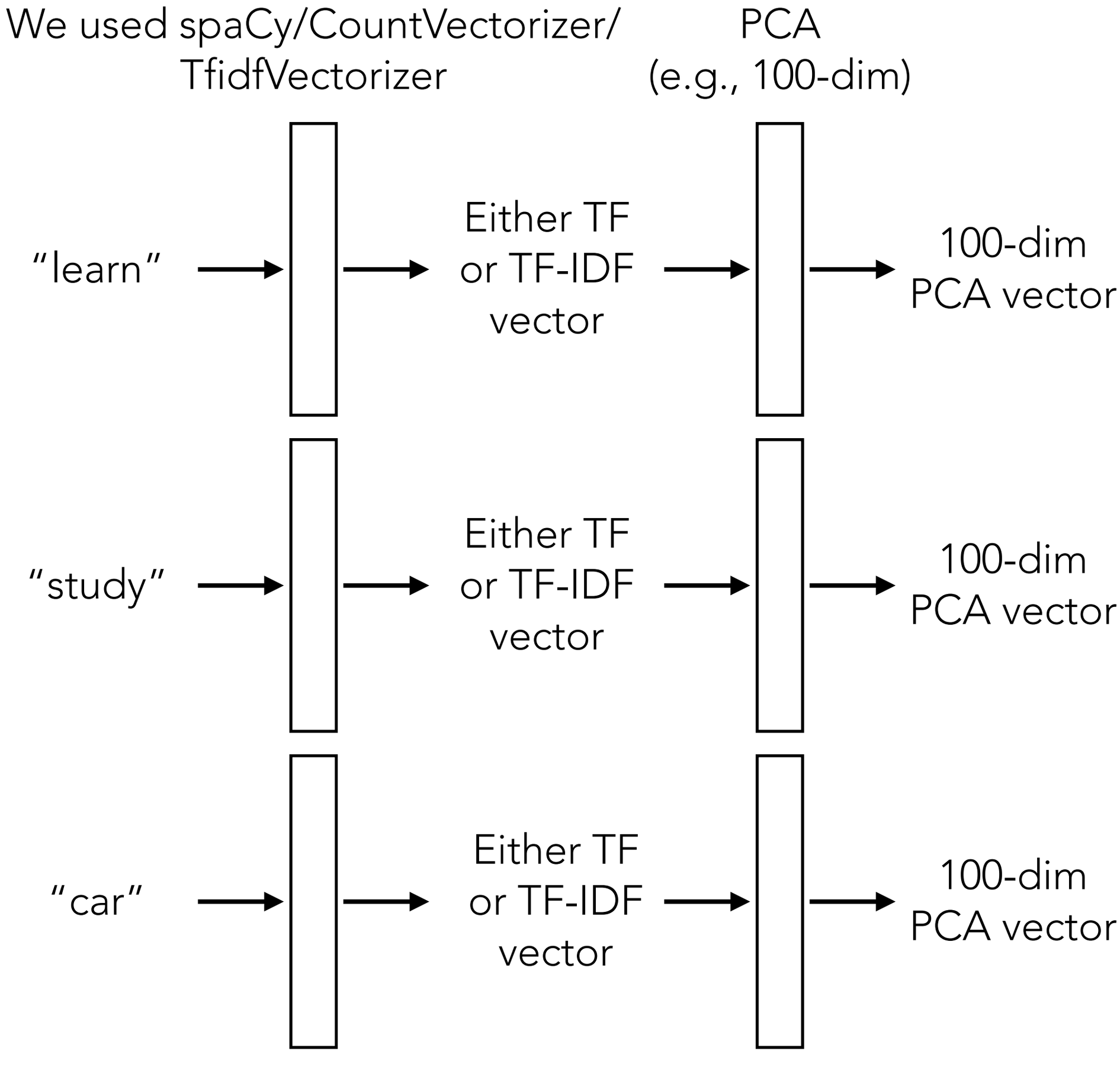
Next week I'll give a little bit more intuition for these in our coverage of transformers (for handling time series)

  - Analyzing images using convolutional neural nets (CNNs)
- Next week:
  wrap up CNNs (if we don't finish talking about them today),
  text generation using generative pretrained transformers

# Handwritten Digit Recognition

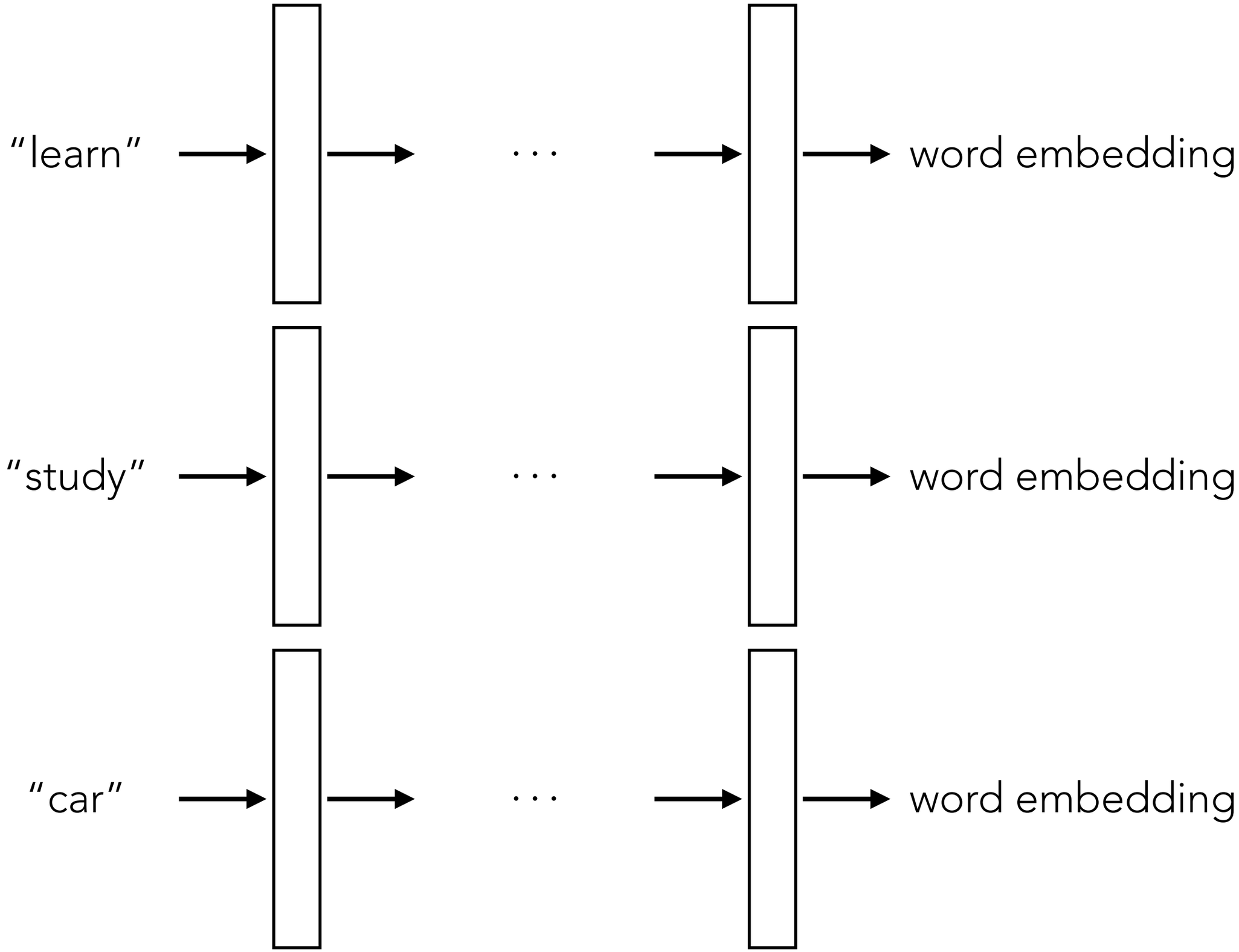
Demo

# A brief glimpse at word embeddings


We used spaCy/CountVectorizer/
TfidfVectorizer
PCA
(e.g., 100-dim)
"learn"
Either TF
or TF-IDF
vector
100-dim
PCA vector
"study"
Either TF
or TF-IDF
vector
100-dim
PCA vector
"car"
Either TF
or TF-IDF
vector
100-dim
PCA vector

Tokens/words

Neural net model



# (Flashback) Do Data Actually Live on Manifolds?

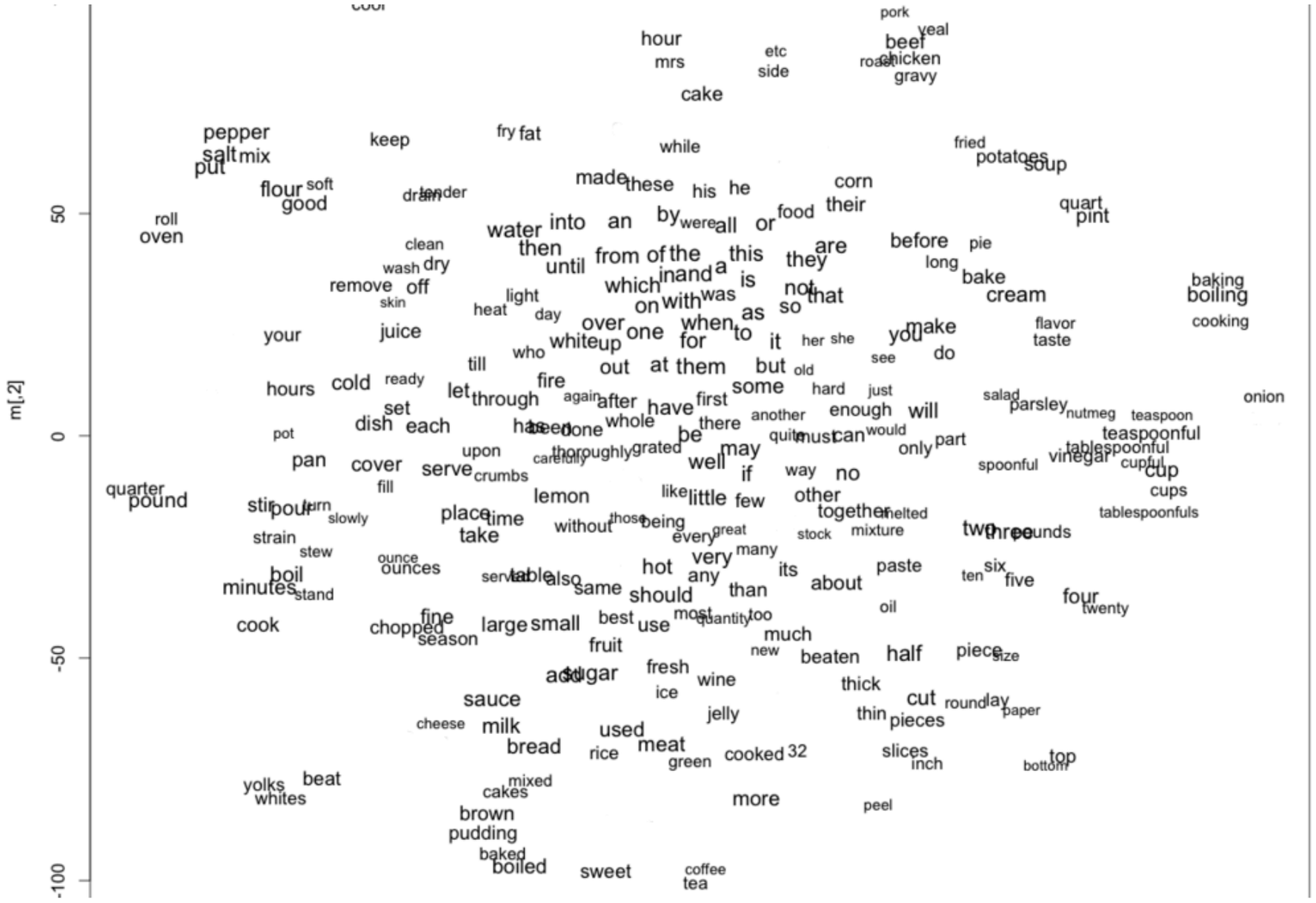



Image source: http://www.adityathakker.com/wp-content/uploads/2017/06/word-embeddings-994x675.png

# Word Embeddings:
# Even without labels, we can set up a prediction problem!

*Hide* part of training data and try to predict what you've hid!

# Word Embeddings: word2vec (2013)

Can solve tasks like the following:

Man is to King as Woman is to ???

# Word Embeddings: word2vec (2013)

Can solve tasks like the following:

Man is to King as Woman is to Queen

# Word Embeddings: word2vec (2013)

Can solve tasks like the following:

Man is to King as Woman is to Queen

Which word doesn't belong?
blue, red, green, crimson, transparent

# Word Embeddings: word2vec (2013)

Can solve tasks like the following:

Man is to King as Woman is to Queen

Which word doesn't belong?

blue, red, green, crimson, transparent

# Word Embeddings: word2vec (2013)

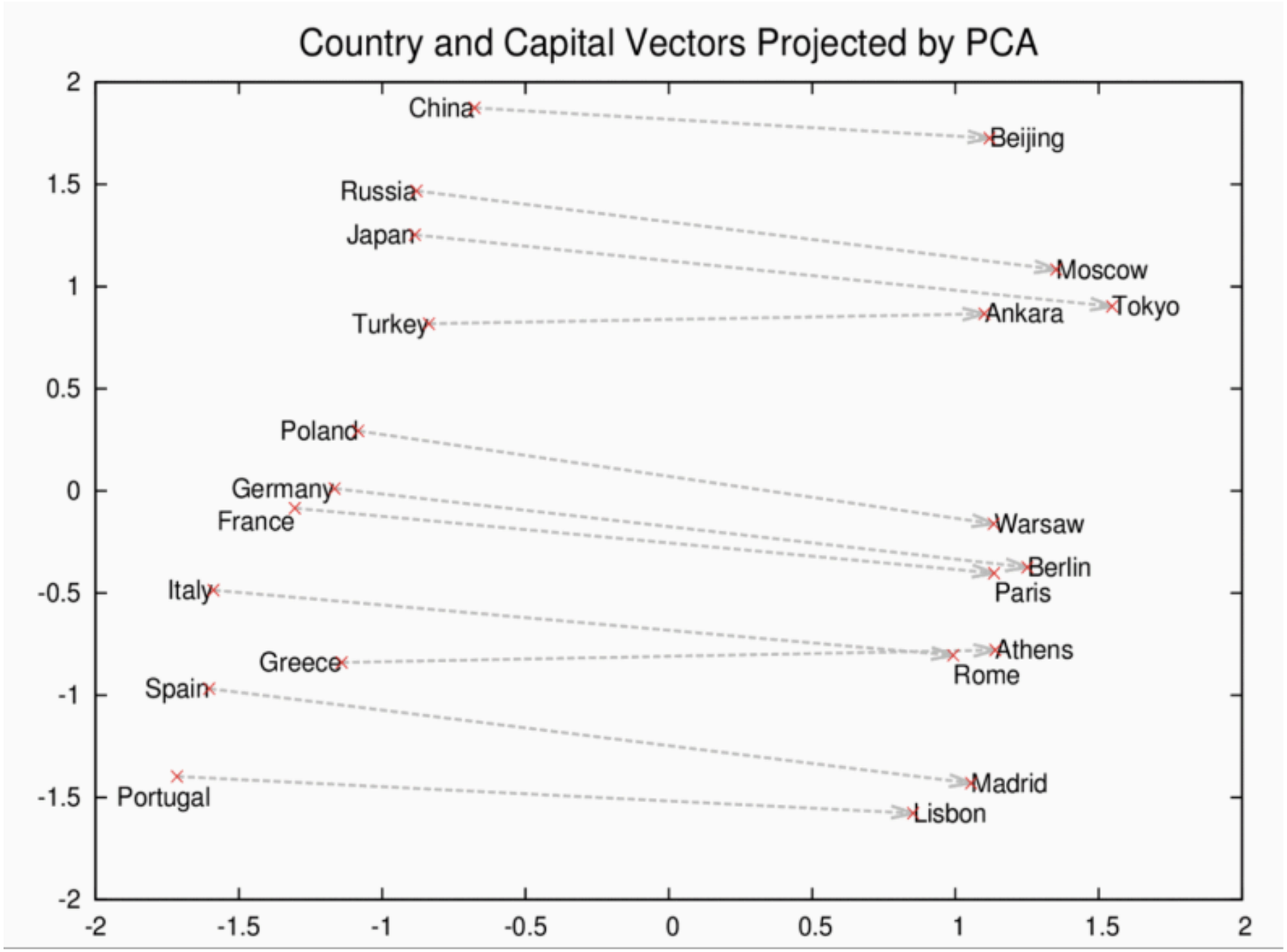



Image source: https://deeplearning4j.org/img/countries_capitals.png

# Word Embeddings: word2vec (2013)

The opioid epidemic or opioid crisis is the rapid increase in the use of prescription and non-prescription opioid drugs in the United States and Canada in the 2010s.

Predict context of each word!

Training data point: epidemic

“Training labels”: the, opioid, or, opioid

# Word Embeddings: word2vec (2013)


Predict context of each word!

Training data point: or

"Training labels": opioid, epidemic, opioid, crisis

# Word Embeddings: word2vec (2013)


Predict context of each word!

Training data point: opioid

"Training labels": epidemic, or, crisis, is

These are "positive" (correct) examples of what context words are for "opioid"

Also provide "negative" examples of words that are *not* likely to be context words (by randomly sampling words elsewhere in document)

# Word Embeddings: word2vec (2013)


randomly sampled word

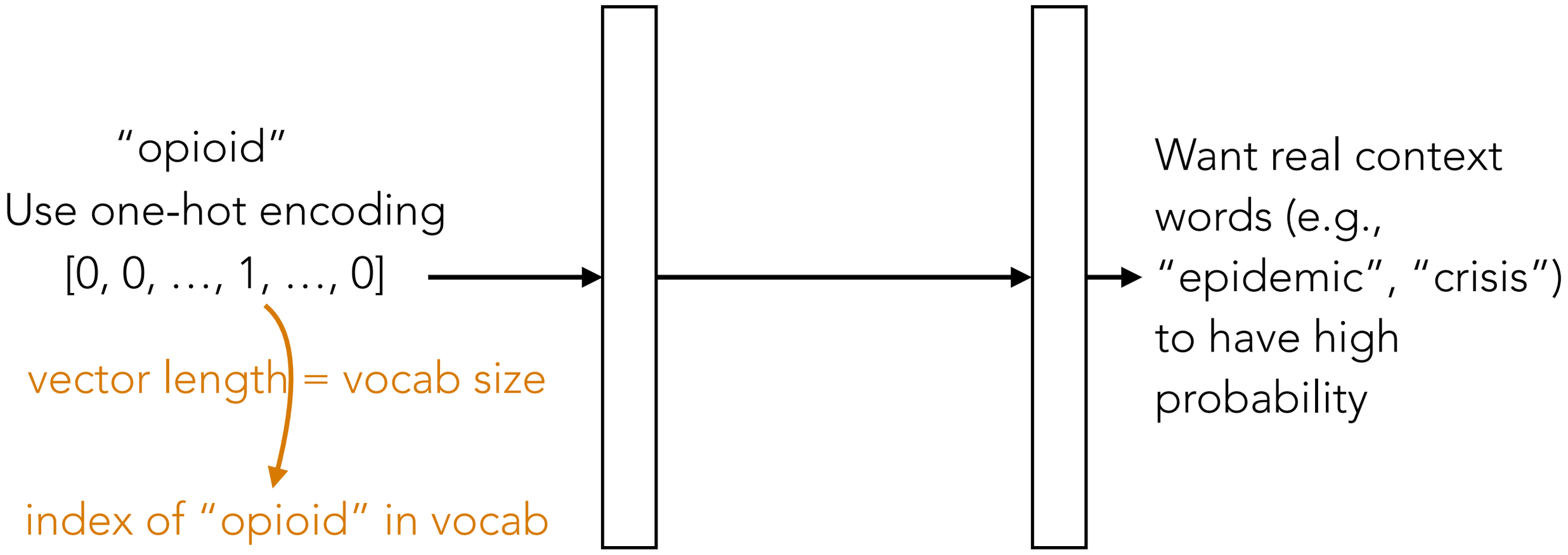
Predict context of each word!

Training data point: opioid

"Negative training label": 2010s

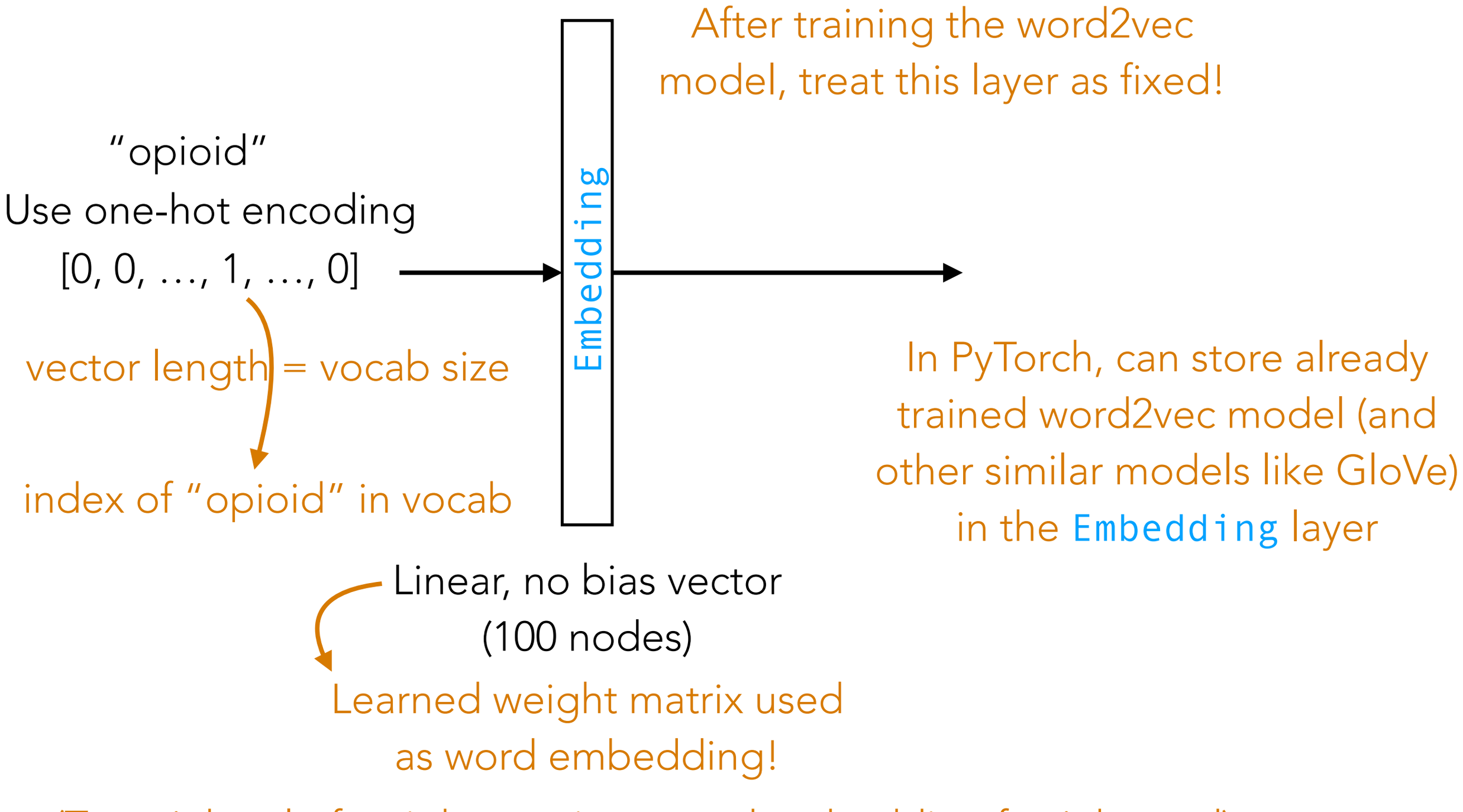
Also provide "negative" examples of words that are *not* likely to be context words (by randomly sampling words elsewhere in document)

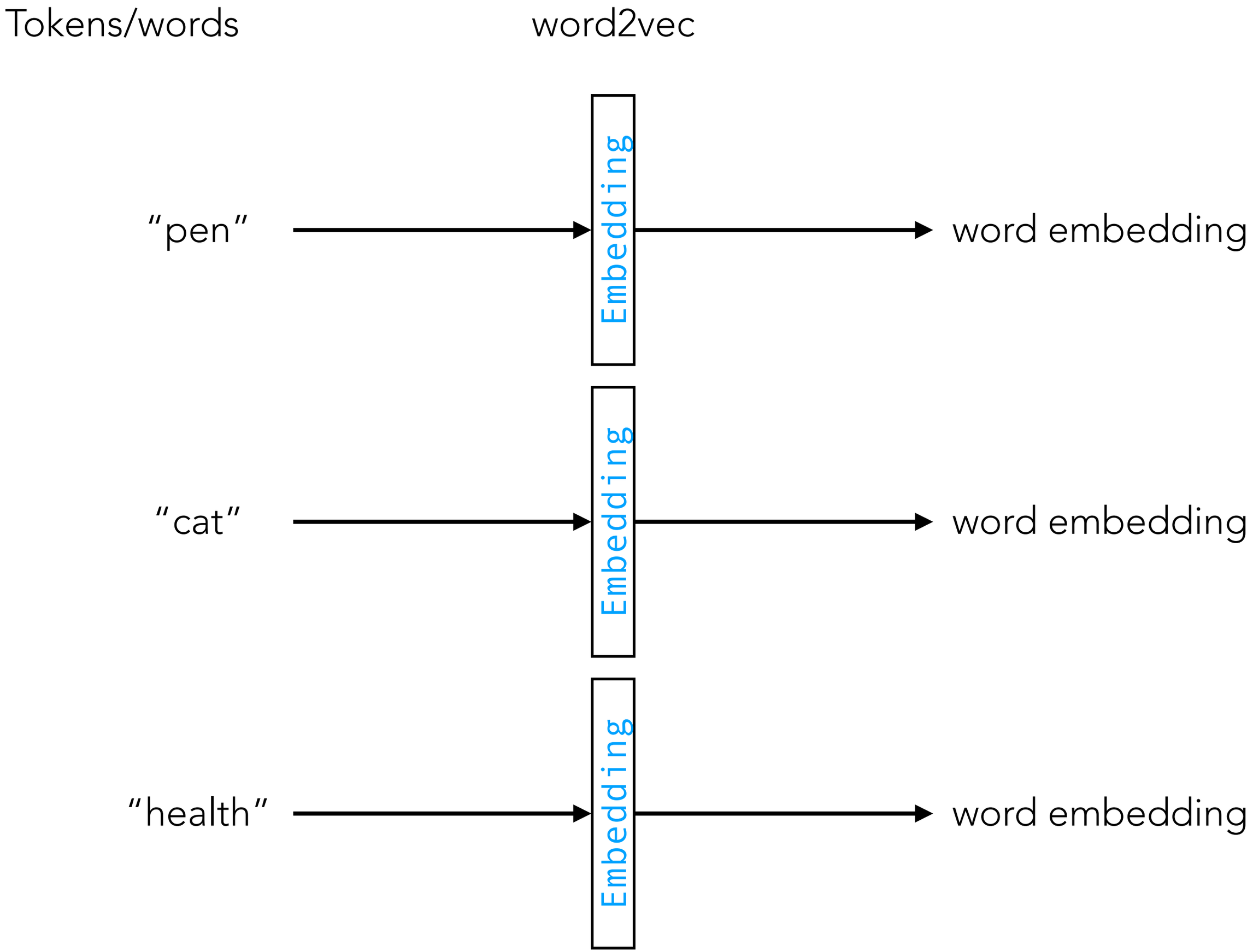
# Word2vec Neural Net




(Treat *i*-th col of weight matrix as word embedding for *i*-th word)

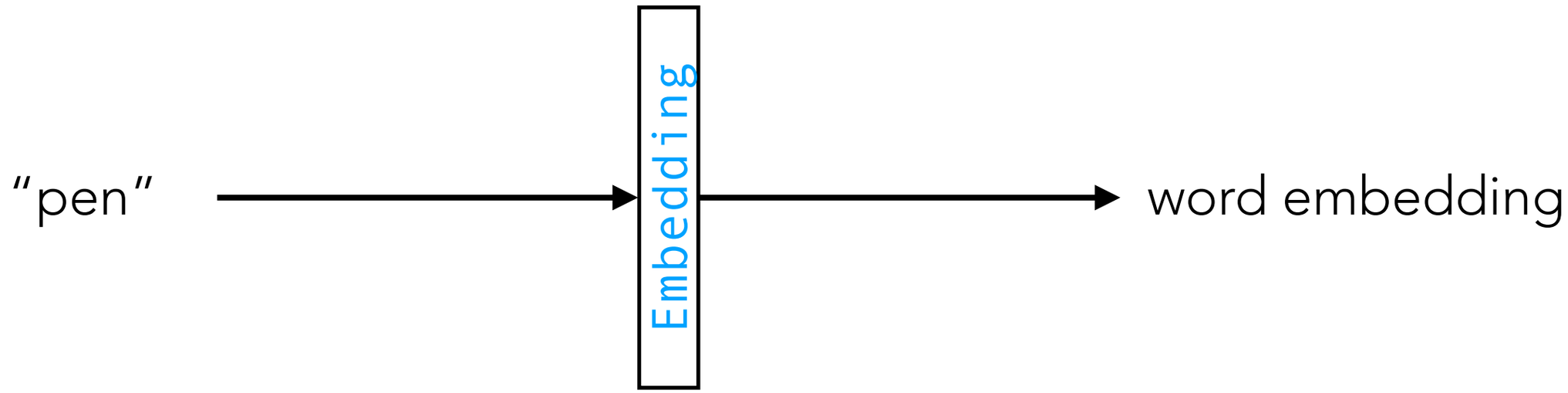
# Word2vec Neural Net




Tokens/words
word2vec
“pen”
Embedding
word embedding
“cat”
Embedding
word embedding
“health”
Embedding
word embedding

Tokens/words

word2vec




Even though "pen" has multiple meanings
(e.g., what you write with vs a play pen),
word2vec would produce the same word embedding for "pen"

(Flashback)

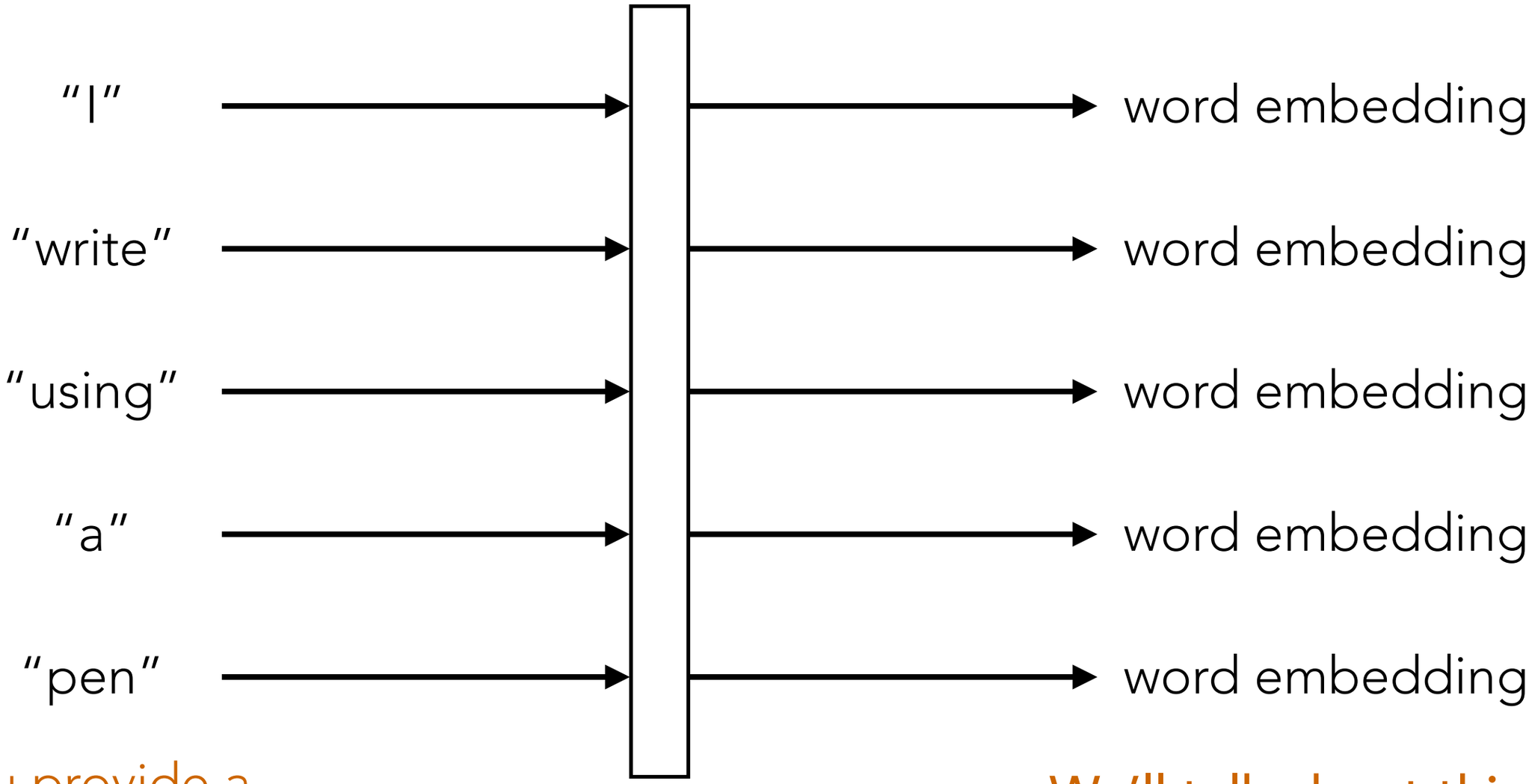
# What about a word that has multiple meanings?

Challenging: try to split up word into multiple words depending on meaning (requires inferring meaning from context)

This problem is called word sense disambiguation (WSD)

# Modern Word Embeddings Use Context

(such as BERT, which came out in 2018)




You provide a whole sentence (or a longer document)

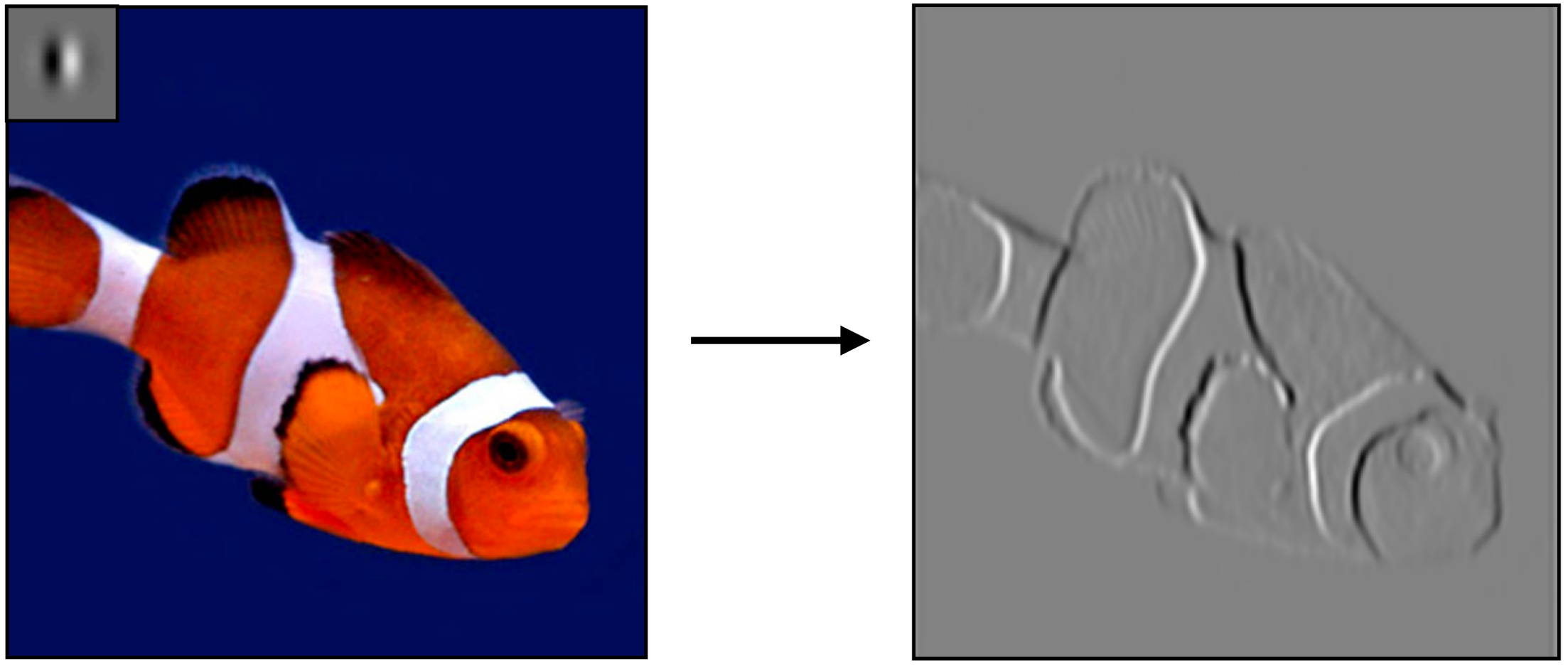
More complicated neural net

We'll talk about this sort of idea more next week

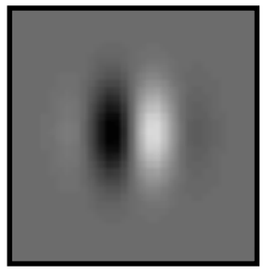
Recitation tomorrow goes over how to use this technology!

# Accounting for image structure: convolutional neural nets (CNNs or convnets)

# Convolution



filter



Slide by Phillip Isola

# Convolution

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Input image

| | | |
|---|---|---|
| -1 | -1 | -1 |
| 2 | 2 | 2 |
| -1 | -1 | -1 |

Filter
(also called “kernel”)

# Convolution

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Input image

| | | |
|---|---|---|
| **-1** | **-1** | **-1** |
| **2** | **2** | **2** |
| **-1** | **-1** | **-1** |

Filter
(also called “kernel”)

# Convolution

0(-1) + 0(-1) + 0(-1) + 0(2) + 0(2) + 1(2) + 0(-1) + 1(-1) + 1(-1)

= 2 - 1 - 1 = 0

Take dot product!

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 **-1** | 0 **-1** | 0 **-1** | 0 | 0 | 0 | 0 |
| 0 **2** | 0 **2** | 1 **2** | 1 | 1 | 0 | 0 |
| 0 **-1** | 1 **-1** | 1 **-1** | 1 | 1 | 1 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Input image

| | | | | |
|---|---|---|---|---|
| 0 | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

Output image

# Convolution

Take dot product!

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 -1 | 0 -1 | 0 -1 | 0 | 0 | 0 |
| 0 | 0 2 | 1 2 | 1 2 | 1 | 0 | 0 |
| 0 | 1 -1 | 1 -1 | 1 -1 | 1 | 1 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Input image

| | | | | |
|---|---|---|---|---|
| 0 | 1 | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

Output image

# Convolution

Take dot product!

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 (-1) | 0 (-1) | 0 (-1) | 0 | 0 |
| 0 | 0 | 1 (2) | 1 (2) | 1 (2) | 0 | 0 |
| 0 | 1 | 1 (-1) | 1 (-1) | 1 (-1) | 1 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Input image

| | | | | |
|---|---|---|---|---|
| 0 | 1 | 3 | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

Output image

# Convolution

Take dot product!

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Input image

| | | | | |
|---|---|---|---|---|
| 0 | 1 | 3 | 1 | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

Output image

# Convolution

Take dot product!

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 -1 | 0 -1 | 0 -1 |
| 0 | 0 | 1 | 1 | 1 2 | 0 2 | 0 2 |
| 0 | 1 | 1 | 1 | 1 -1 | 1 -1 | 0 -1 |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Input image

| | | | | |
|---|---|---|---|---|
| 0 | 1 | 3 | 1 | 0 |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

Output image

# Convolution

Take dot product!

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Input image

| | | | | |
|---|---|---|---|---|
| 0 | 1 | 3 | 1 | 0 |
| 1 | | | | |
| | | | | |
| | | | | |
| | | | | |

Output image

# Convolution

Take dot product!

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Input image

| | | | | |
|---|---|---|---|---|
| 0 | 1 | 3 | 1 | 0 |
| 1 | 1 | | | |
| | | | | |
| | | | | |
| | | | | |

Output image

# Convolution

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Input image

*

| | | |
|---|---|---|
| **-1** | **-1** | **-1** |
| **2** | **2** | **2** |
| **-1** | **-1** | **-1** |

=

| | | | | |
|---|---|---|---|---|
| 0 | 1 | 3 | 1 | 0 |
| 1 | 1 | 1 | 3 | 3 |
| 0 | 0 | -2 | -4 | -4 |
| 1 | 1 | 1 | 3 | 3 |
| 0 | 1 | 3 | 1 | 0 |

Output image

Note: output image is smaller than input image

# Convolution

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Input image

$$* \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} = \frac{1}{9} \begin{bmatrix} 3 & 5 & 6 & 5 & 3 \\ 5 & 8 & 8 & 6 & 3 \\ 6 & 9 & 8 & 7 & 4 \\ 5 & 8 & 8 & 6 & 3 \\ 3 & 5 & 6 & 5 & 3 \end{bmatrix}$$

Output image

# Convolution

Very commonly used for:

- Blurring an image

*

| | | |
|---|---|---|
| 1/9 | 1/9 | 1/9 |
| 1/9 | 1/9 | 1/9 |
| 1/9 | 1/9 | 1/9 |

=

- Finding edges

*

| | | |
|---|---|---|
| -1 | -1 | -1 |
| 2 | 2 | 2 |
| -1 | -1 | -1 |

=

(this example finds horizontal edges)

Images from: http://aishack.in/tutorials/image-convolution-examples/

# Convolution Layer

Activation layer (such as ReLU)

Conv2d layer

convolve with each filter

| | | |
|---|---|---|
| 1/9 | 1/9 | 1/9 |
| 1/9 | 1/9 | 1/9 |
| 1/9 | 1/9 | 1/9 |

add bias 42

apply activation

| | | |
|---|---|---|
| -1 | -1 | -1 |
| 2 | 2 | 2 |
| -1 | -1 | -1 |

add bias -17

apply activation

| | | |
|---|---|---|
| 0 | -1 | 0 |
| -1 | 4 | -1 |
| 0 | -1 | 0 |

add bias 99

apply activation

filters & biases (1 bias number per filter) are unknown and are learned!

Images from: http://aishack.in/tutorials/image-convolution-examples/

# Convolution Layer

Input

Conv2d
(3 kernels,
each size 3x3),
ReLU activation

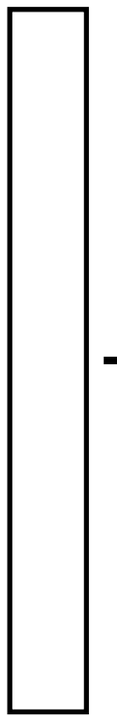
Output images

Images from: http://aishack.in/tutorials/image-convolution-examples/

# Convolution Layer

Input

shape:
1 (# channels),
height,
width



Conv2d
(3 kernels,
each size 3x3),
ReLU activation

Stack output images into a single "output feature map"

shape:
3,
height-2,
width-2

Images from: http://aishack.in/tutorials/image-convolution-examples/

# Convolution Layer

Input

shape:
1 (# channels),
height,
width

Conv2d
(*m* kernels
each size 3x3),
ReLU activation

Stack output images into a single "output feature map"

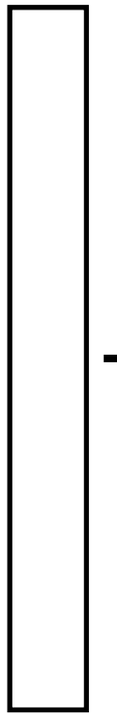
shape:
*m*,
height-2,
width-2

Images from: http://aishack.in/tutorials/image-convolution-examples/

# Convolution Layer

Input

shape:
*d* (# channels)
height,
width



Conv2d
(*m* kernels
each size *d*x3x3),
ReLU activation

Stack output images into a single "output feature map"

shape:
*m*,
height-2,
width-2

Images from: http://aishack.in/tutorials/image-convolution-examples/

# Convolution Layer

Input

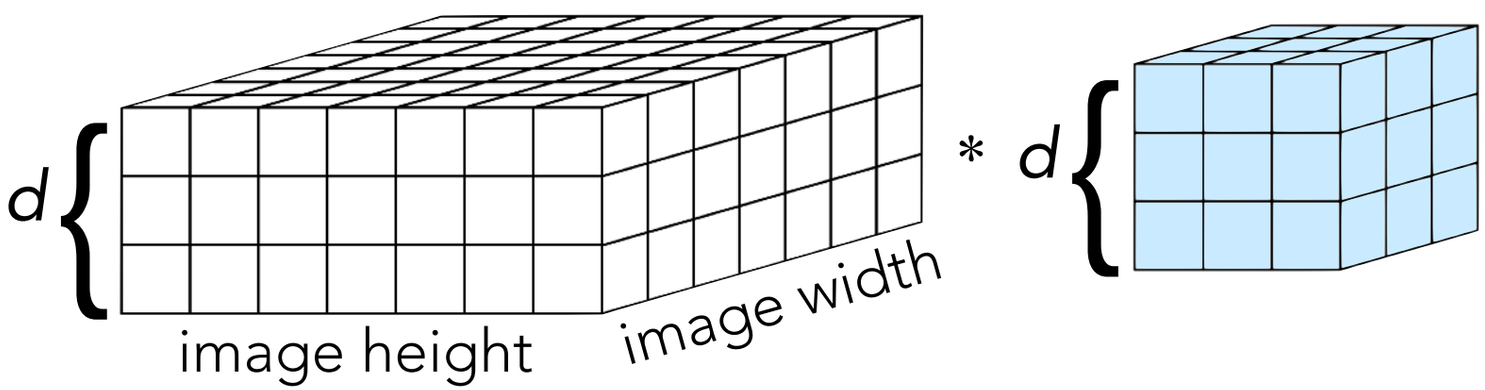
shape:
*d* (# channels)
height,
width

Conv2d
(*m* kernels
each size *d*x3x3),
ReLU activation

Stack output images into a single "output feature map"

shape:
*m*,
height-2,
width-2

Each filter:



# Pooling

- Produces smaller image summarizing original larger image

- To produce this smaller image, need to aggregate or “pool” together information

# Max Pooling

Called “2-by-2” max pooling since this green box is 2 rows by 2 columns

Take maximum value

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Input image

| | | |
|---|---|---|
| | | |
| | | |
| | | |

Output image

3-by-4 max pooling would mean that the green box is 3 rows by 4 columns, etc

# Max Pooling

Called “2-by-2” max pooling since this green box is 2 rows by 2 columns

Take maximum value

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Input image

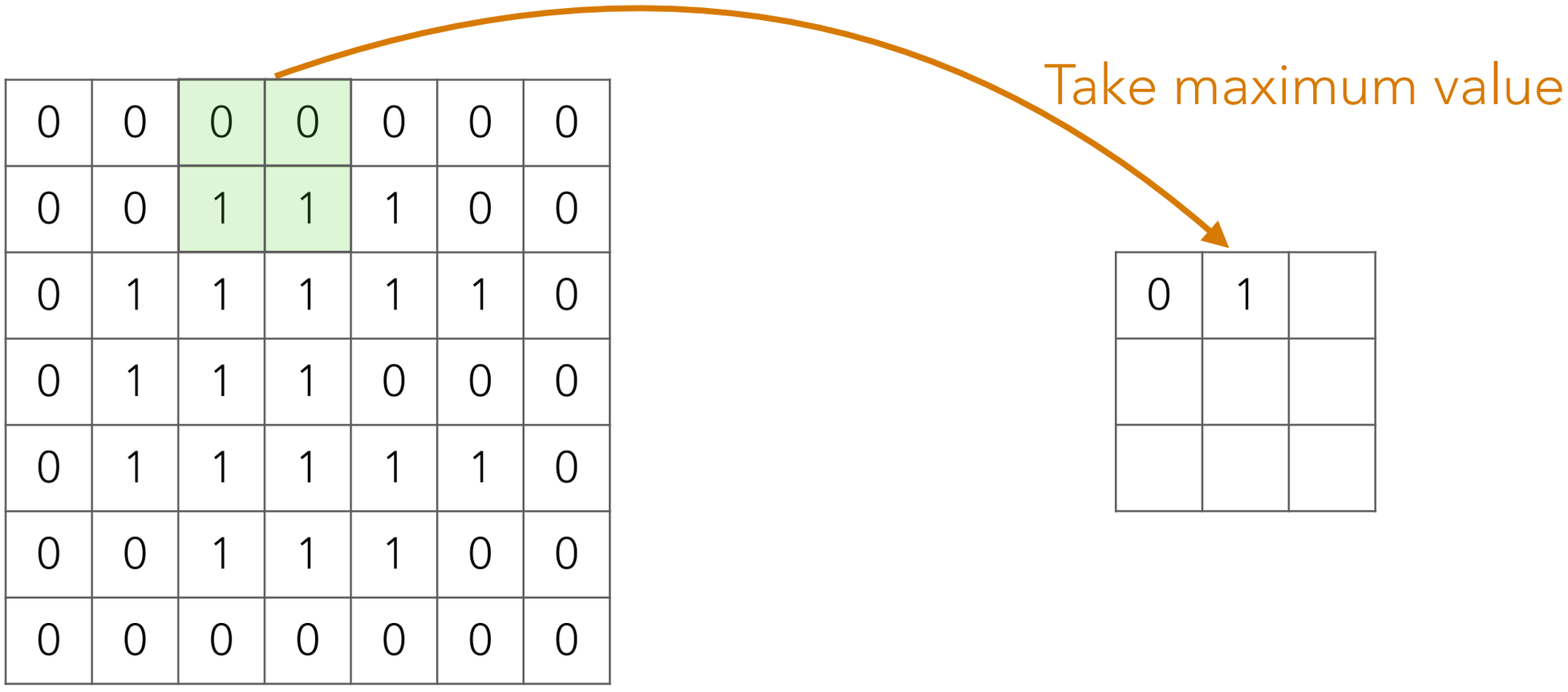
| | | |
|---|---|---|
| 0 | | |
| | | |
| | | |

Output image

3-by-4 max pooling would mean that the green box is 3 rows by 4 columns, etc

# Max Pooling

Called “2-by-2” max pooling since this green box is 2 rows by 2 columns



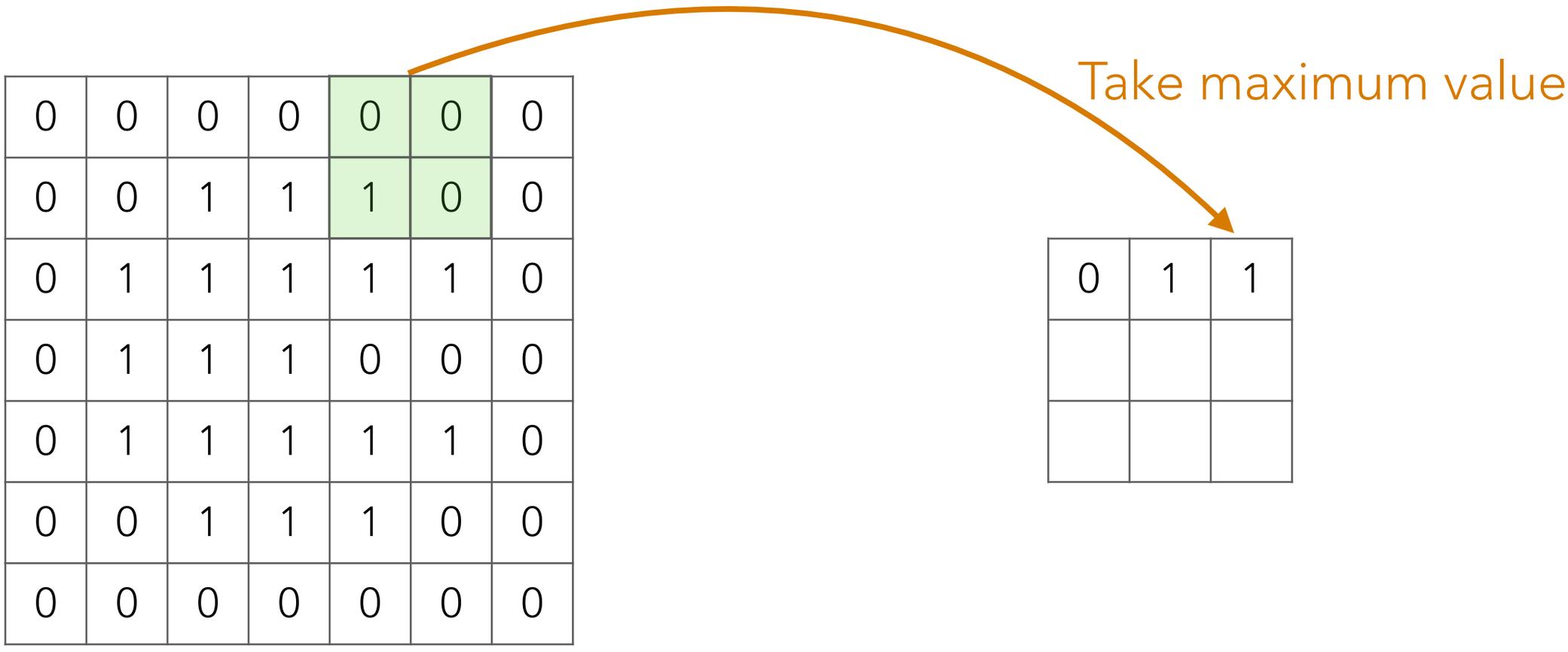

Input image

Output image

3-by-4 max pooling would mean that the green box is 3 rows by 4 columns, etc

# Max Pooling

Called “2-by-2” max pooling since this green box is 2 rows by 2 columns




Input image

Output image

3-by-4 max pooling would mean that the green box is 3 rows by 4 columns, etc

# Max Pooling

Called “2-by-2” max pooling since this green box is 2 rows by 2 columns

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Input image

Take maximum value

| | | |
|---|---|---|
| 0 | 1 | 1 |
| 1 | | |
| | | |

Output image

3-by-4 max pooling would mean that the green box is 3 rows by 4 columns, etc

# Max Pooling

Called “2-by-2” max pooling since this green box is 2 rows by 2 columns

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |

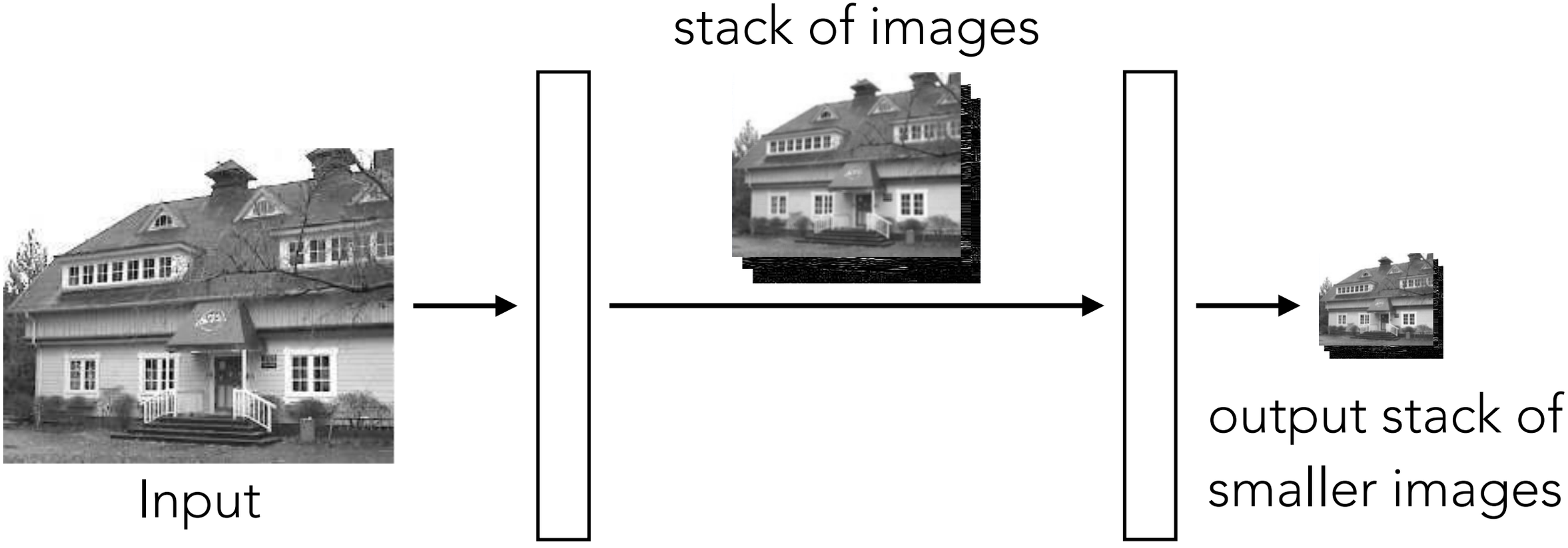
Input image

| | | |
|---|---|---|
| 0 | 1 | 1 |
| 1 | 1 | 1 |
| 1 | 1 | 1 |

Output image

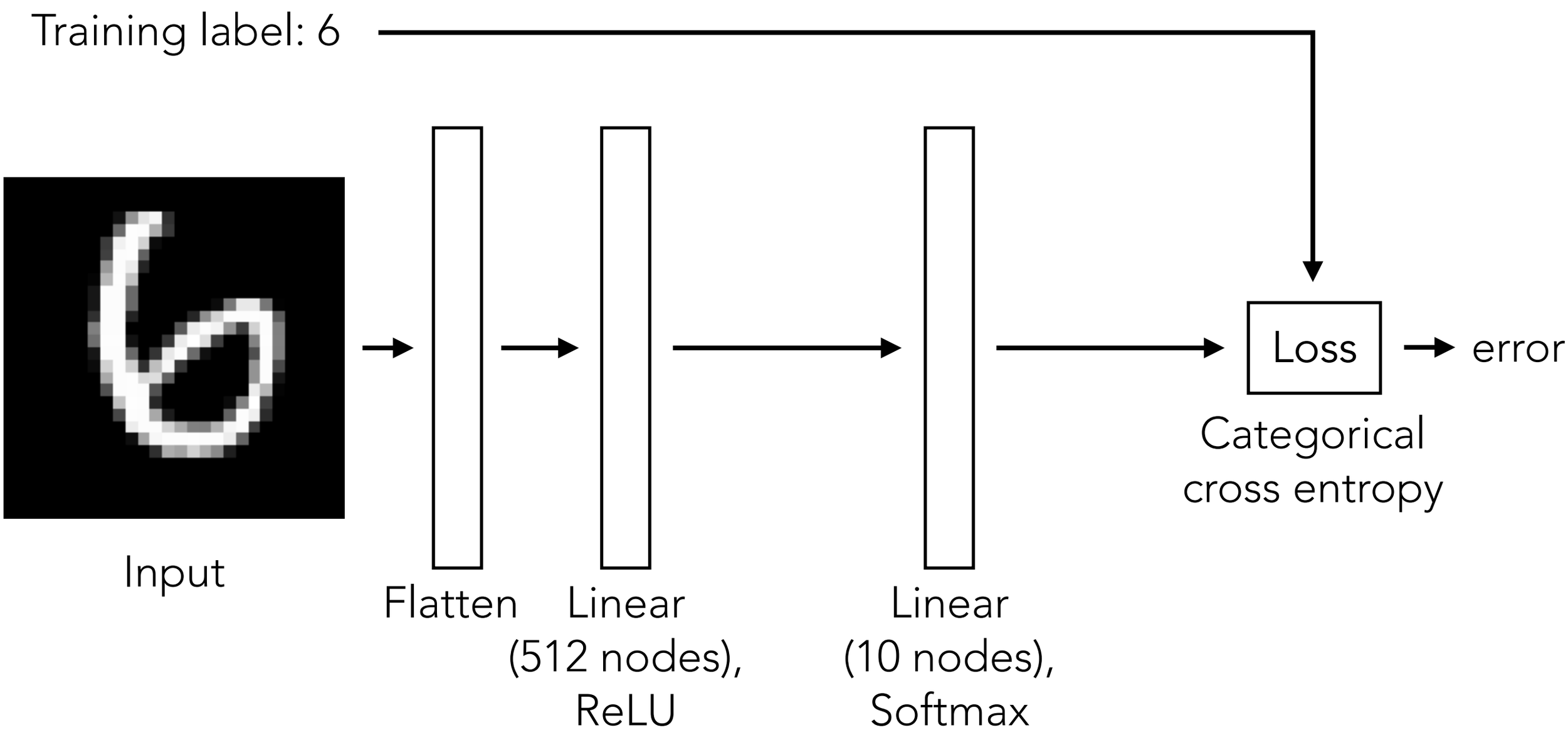
3-by-4 max pooling would mean that the green box is 3 rows by 4 columns, etc

# Common Building Block of CNNs

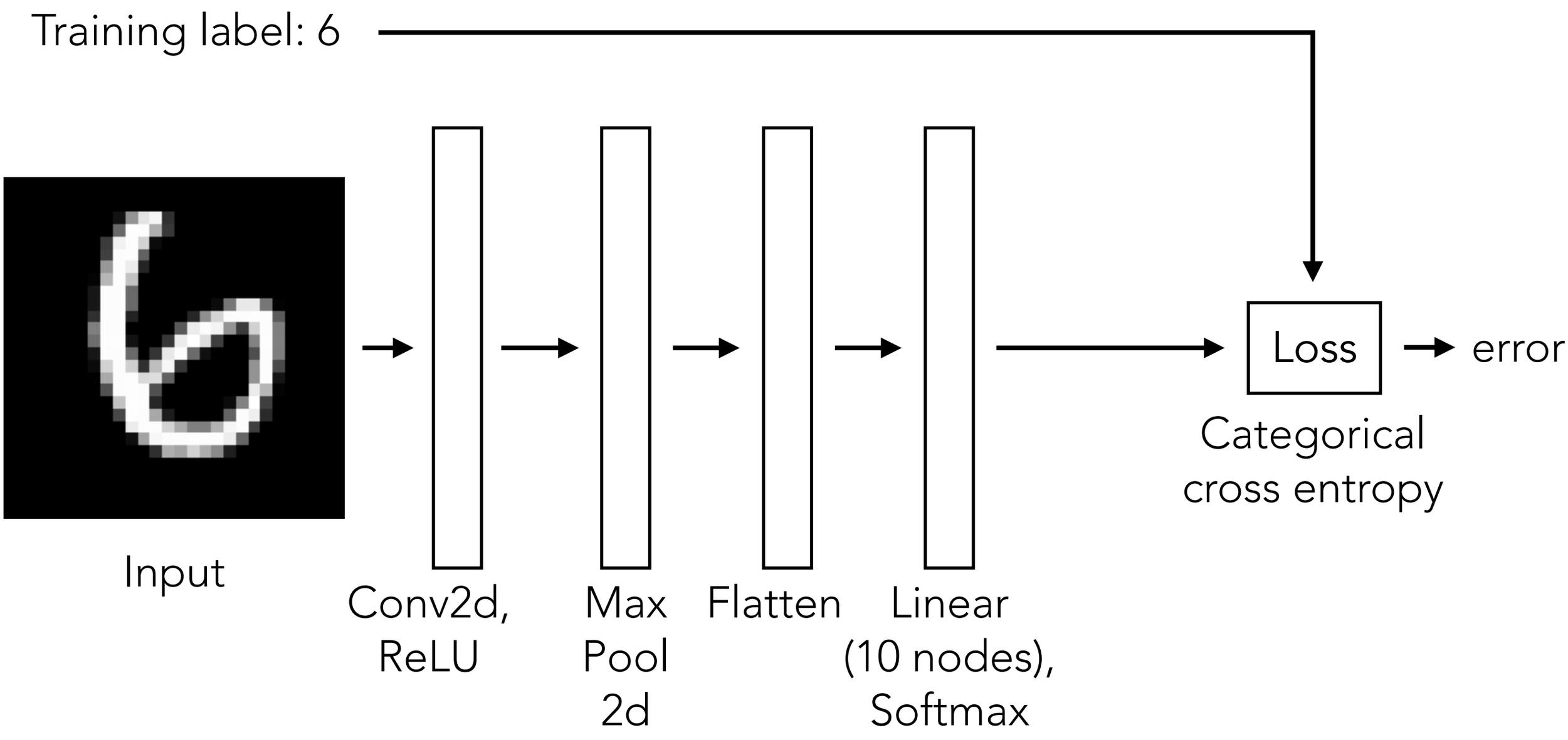



Images from: http://aishack.in/tutorials/image-convolution-examples/

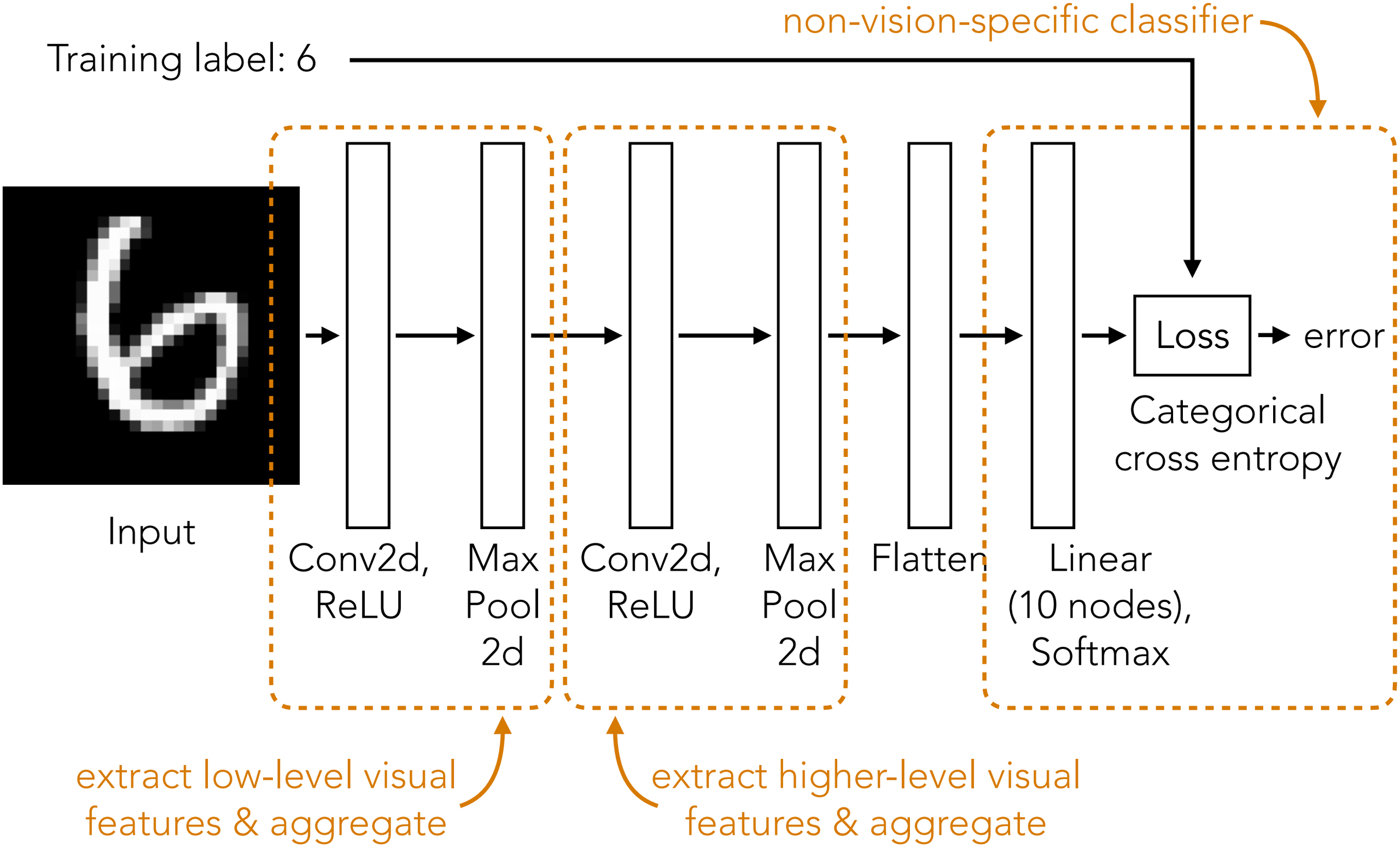
# Handwritten Digit Recognition



# Handwritten Digit Recognition



# Handwritten Digit Recognition



# CNNs

Demo