Carnegie Mellon University
Heinz College

95-865 Unstructured Data Analytics

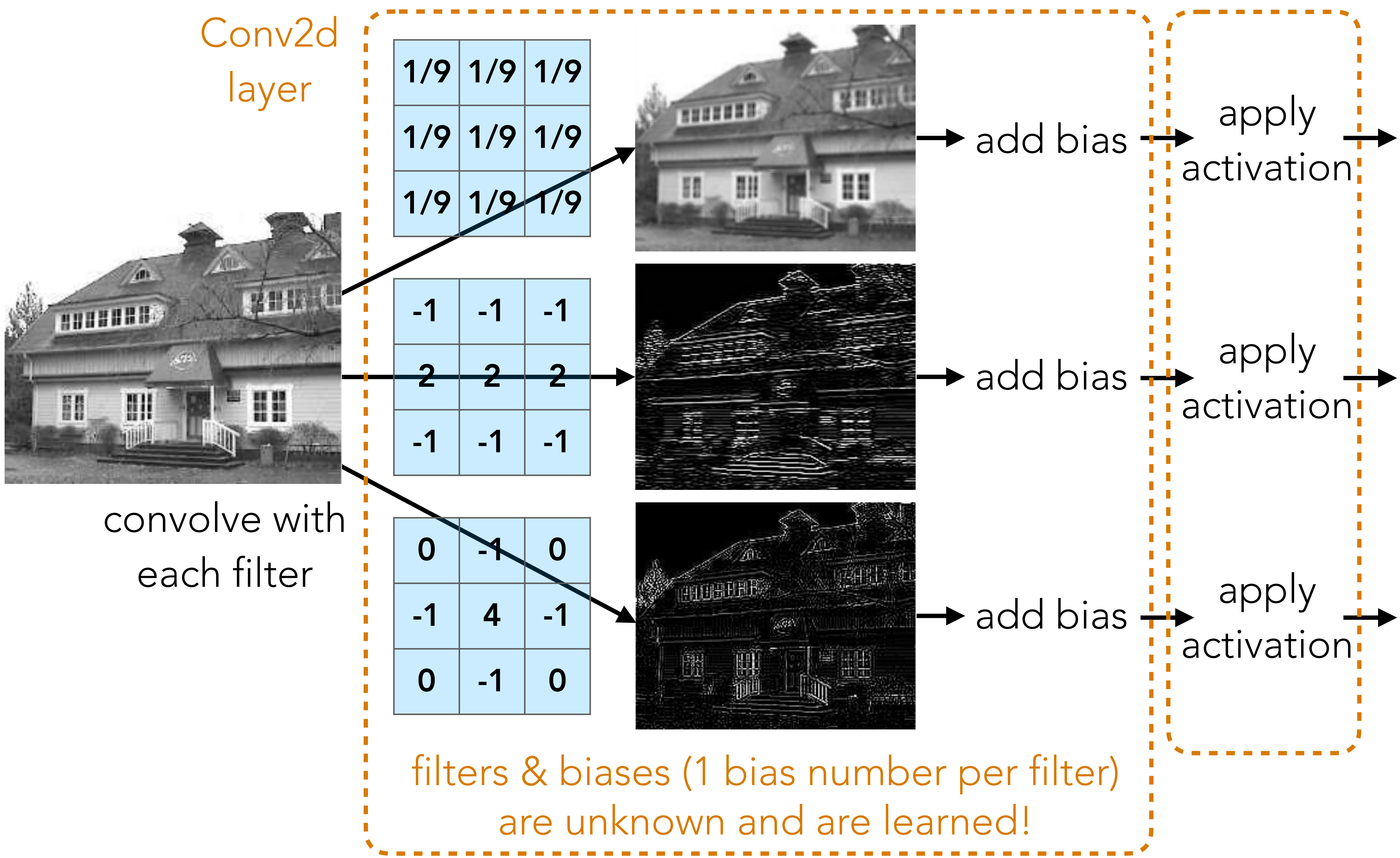
Lecture 13: Wrap up CNNs; time series analysis with recurrent neural nets (RNNs)

Slides by George H. Chen

Administrivia

- HW3 has been released (due Mon Apr 28, 11:59pm)
- There's no questionnaire for HW3 — instead there are official Faculty Course Evaluations (FCEs)!
  - Please fill this out to provide feedback on the course!
  - Your predecessors' feedback greatly improved the course (and *your* feedback could greatly improve the course for your successors, i.e., future 95-865 students)

(Flashback) Convolution Layer

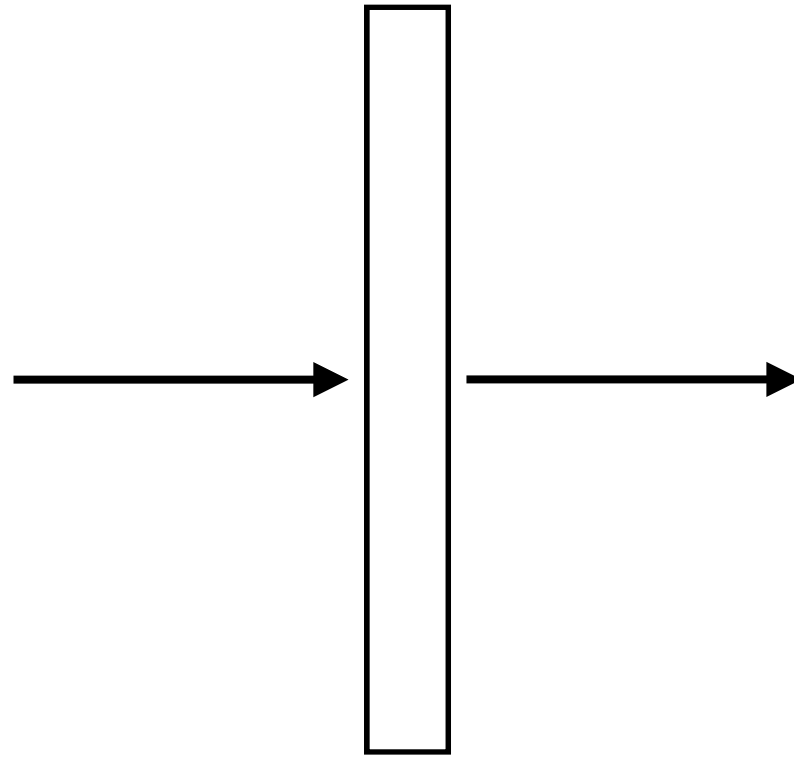


Images from: http://aishack.in/tutorials/image-convolution-examples/

(Flashback) Convolution Layer

Input

shape:
d (# channels)
height,
width

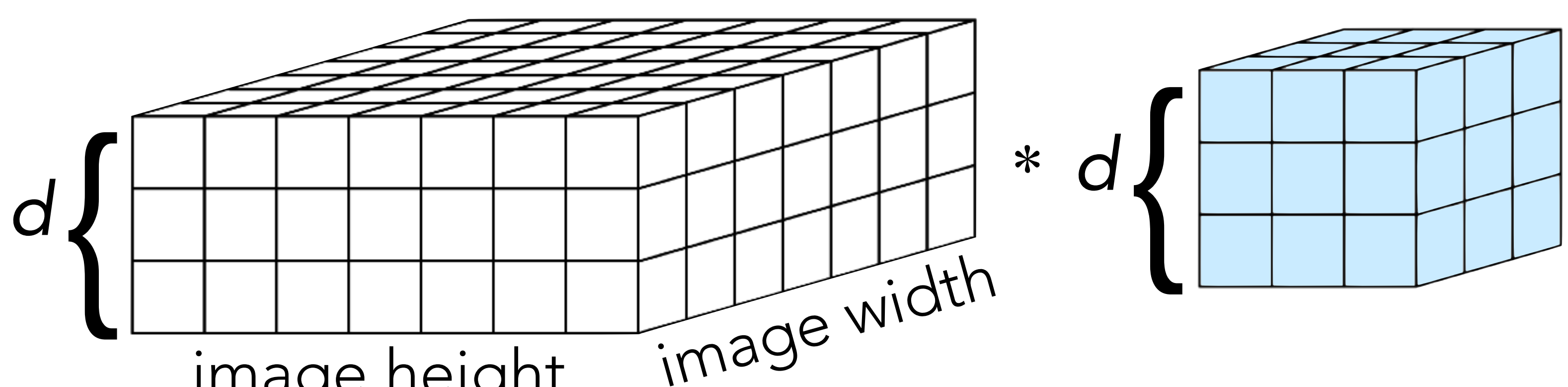


Conv2d
(*k* kernels
each size *d*x3x3),
ReLU activation

Stack output images into a single "output feature map"

shape:
k,
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

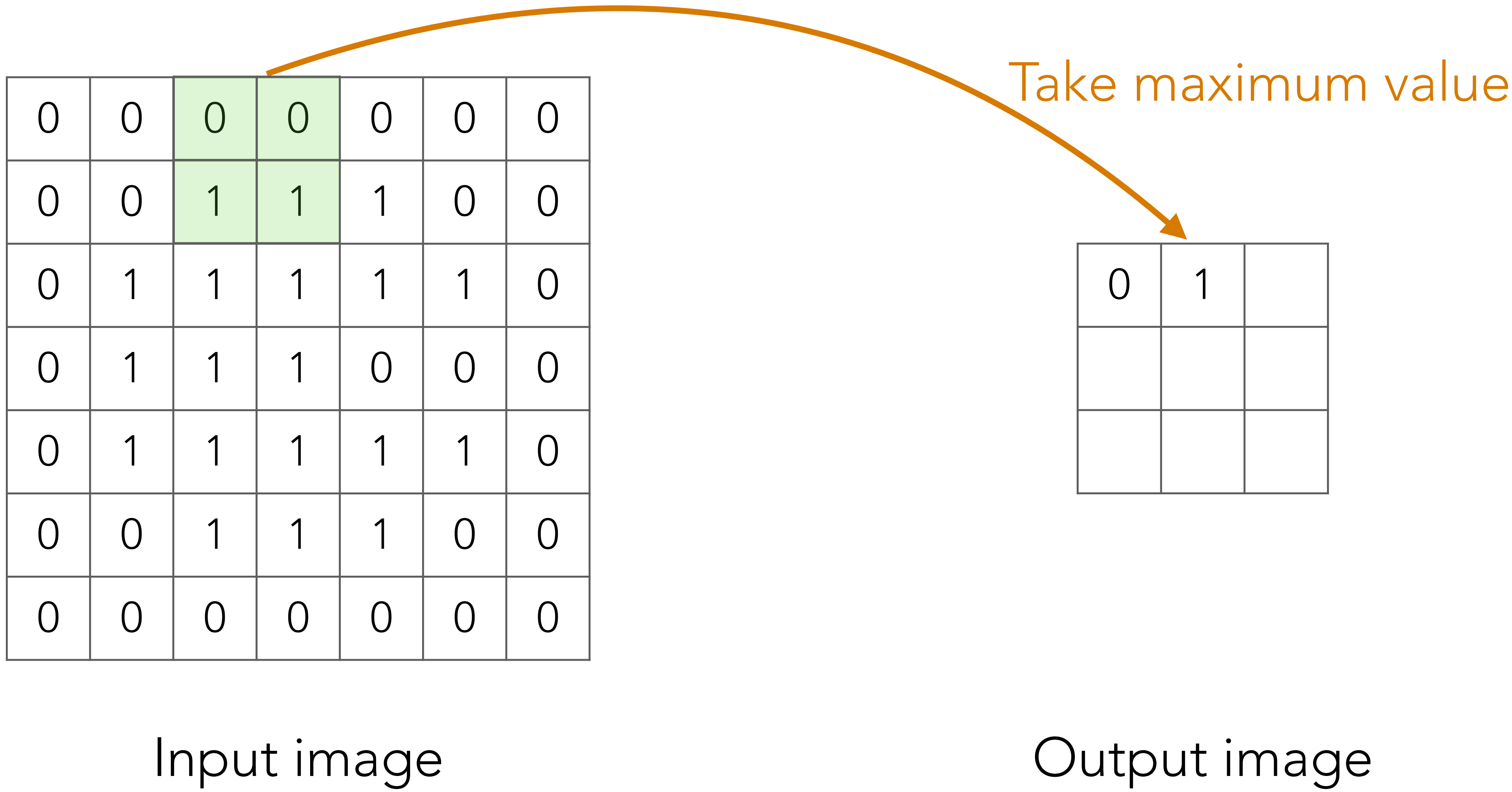
0		

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



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	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

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

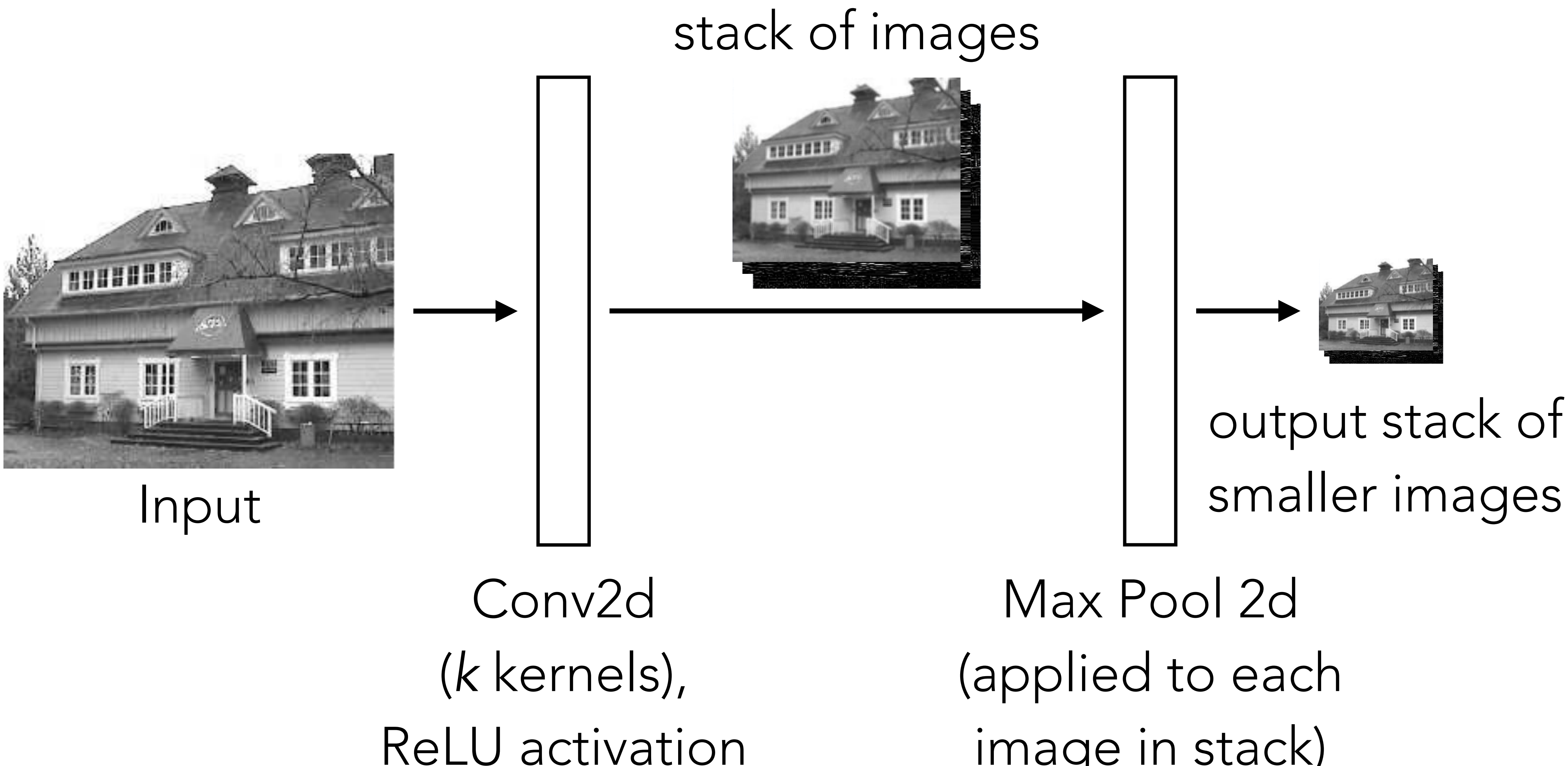
Input image

0	1	1
1	1	1
1	1	1

Output image

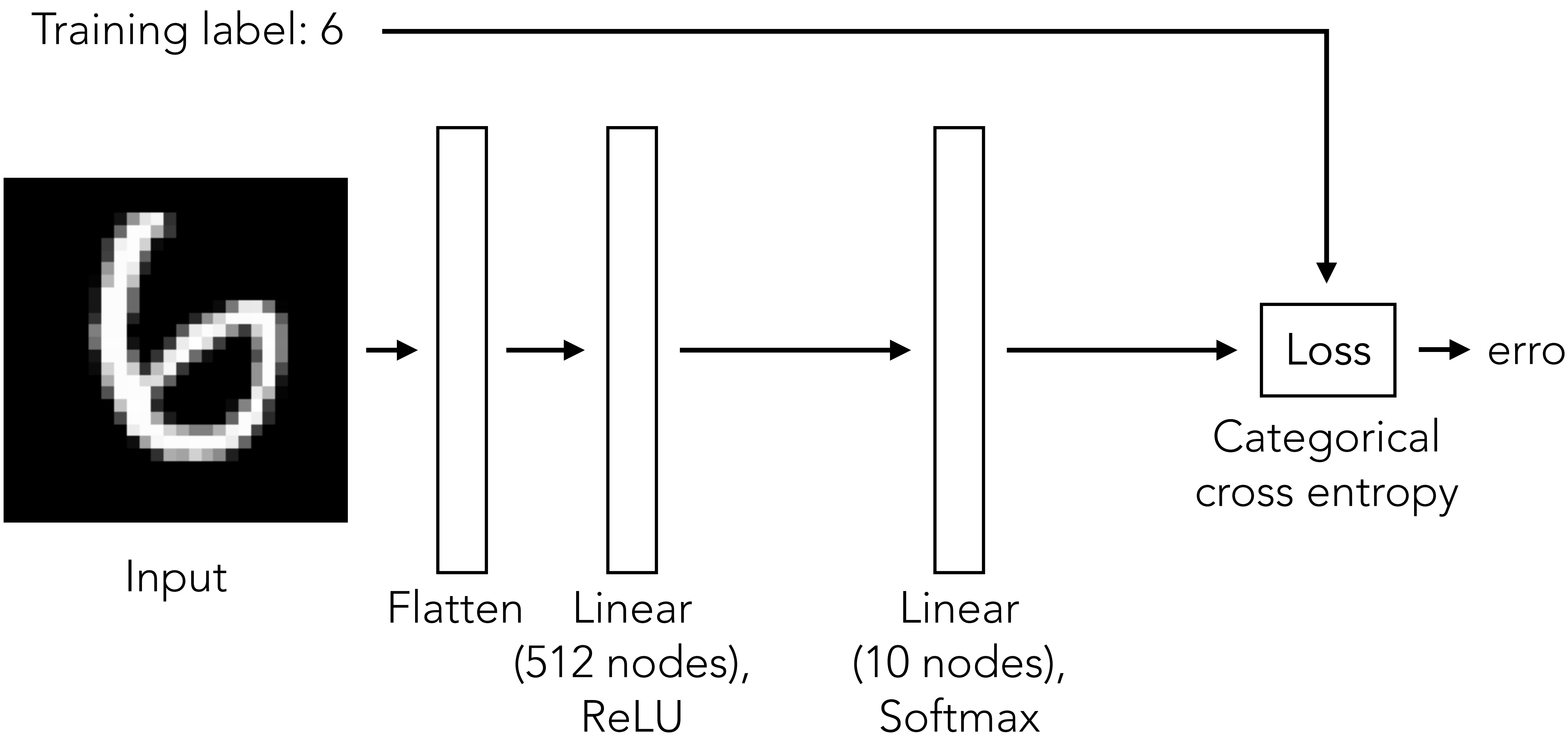
3-by-4 max pooling would mean that the green box is 3 rows by 4 columns, etc

Common Building Block of CNNs

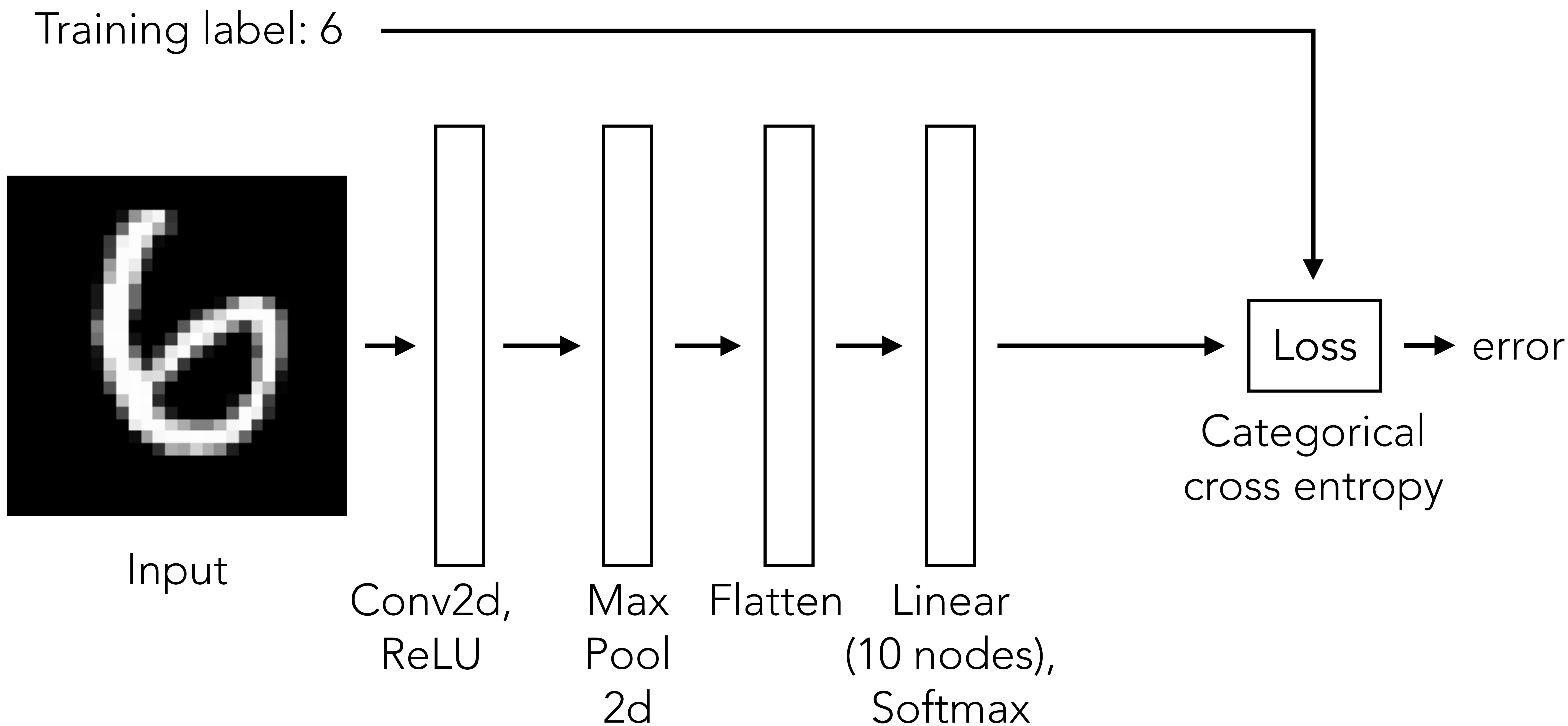


Images from: http://aishack.in/tutorials/image-convolution-examples/

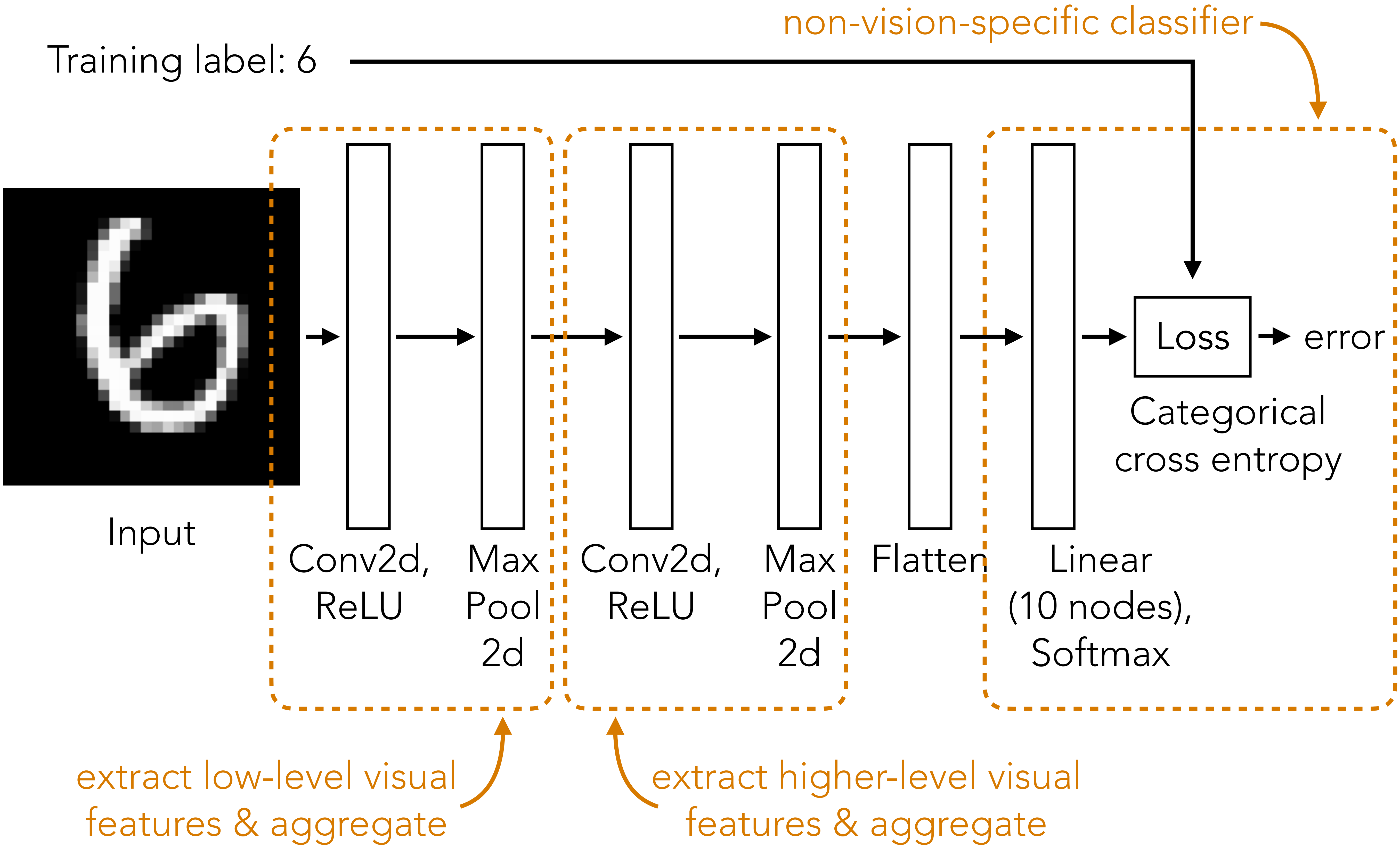
Handwritten Digit Recognition



Handwritten Digit Recognition



Handwritten Digit Recognition



CNNs

Demo

Recap

- A convolution filter processes an input image to produce an output image by taking weighted sums
(examples: blurring an image, finding edges in an image)

- Max pooling produces a *smaller* summary output
  - Max pooling can sometimes produce unexpected behavior when an input image shifts by a small amount:
see Richard Zhang's fix for max pooling (supplemental materials)

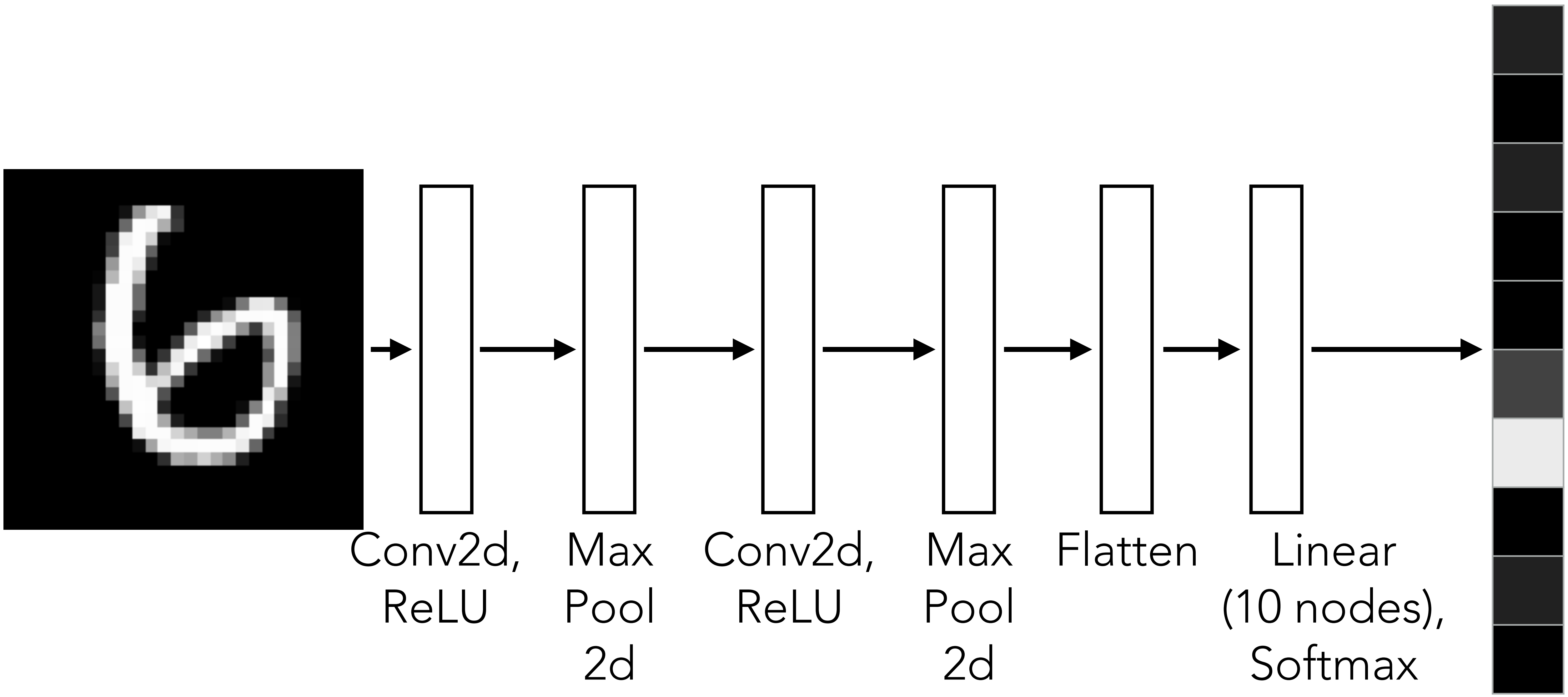
- Repeat convolution→nonlinear activation→pooling to learn increasingly higher-level features

CNNs Encode Semantic Structure for Images

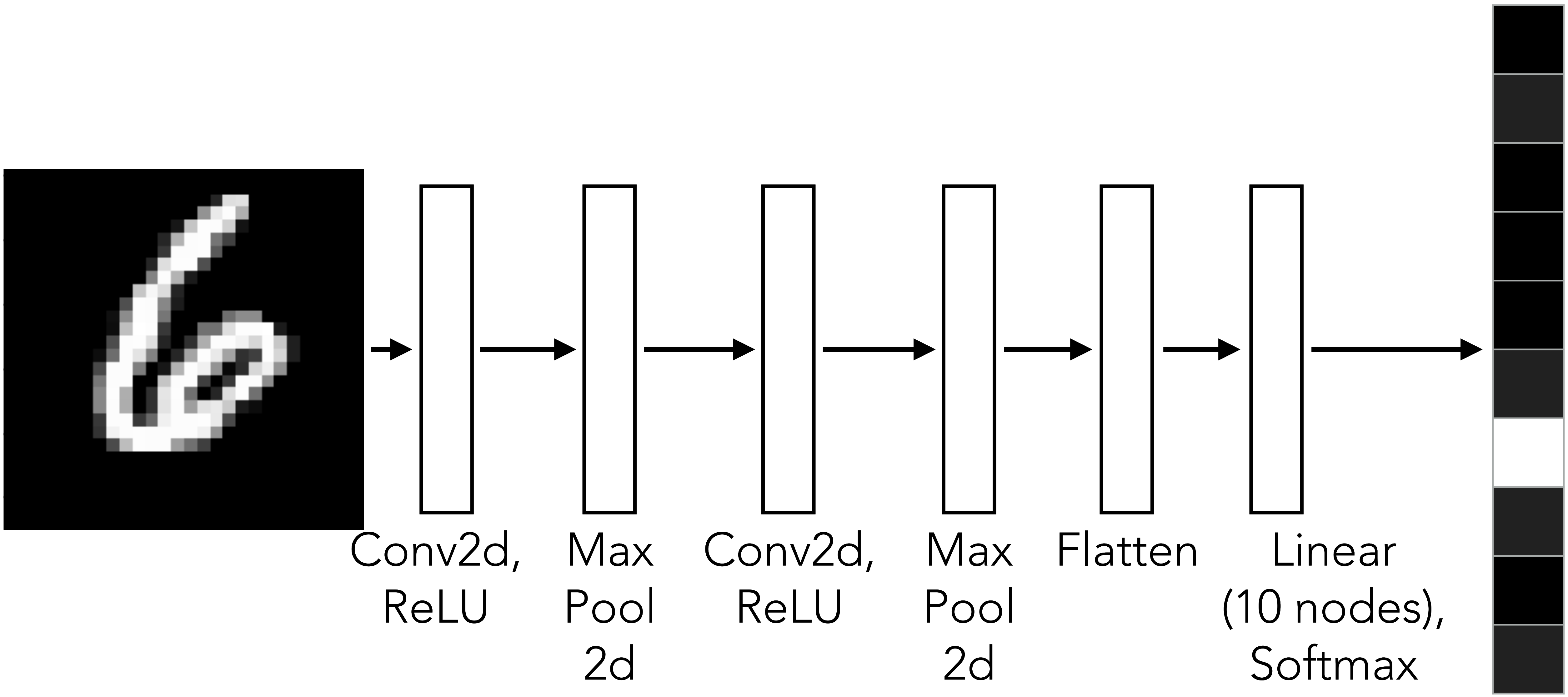
Remember how back in the text clustering & topic modeling demos, 100-dimensional PCA space captured semantic structure of words (such as “study” and “learn” being similar)?

CNNs capture semantic structure for images

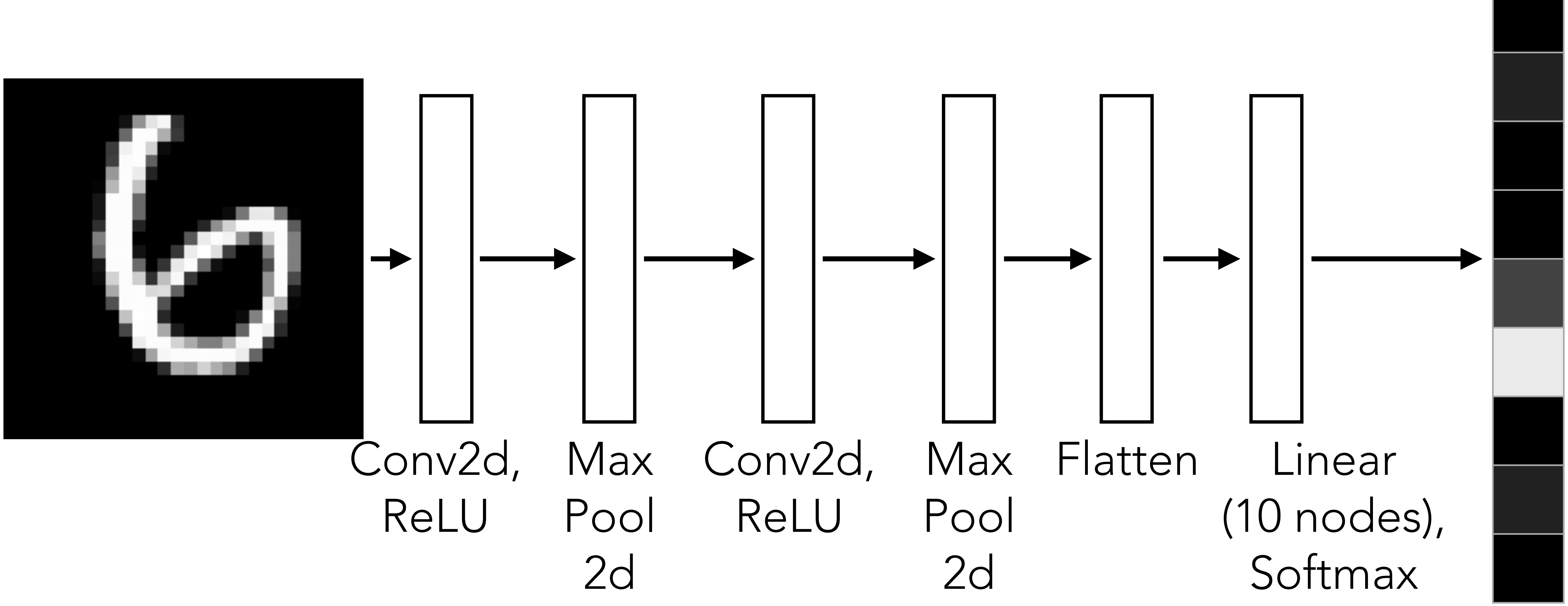
CNNs Encode Semantic Structure for Images

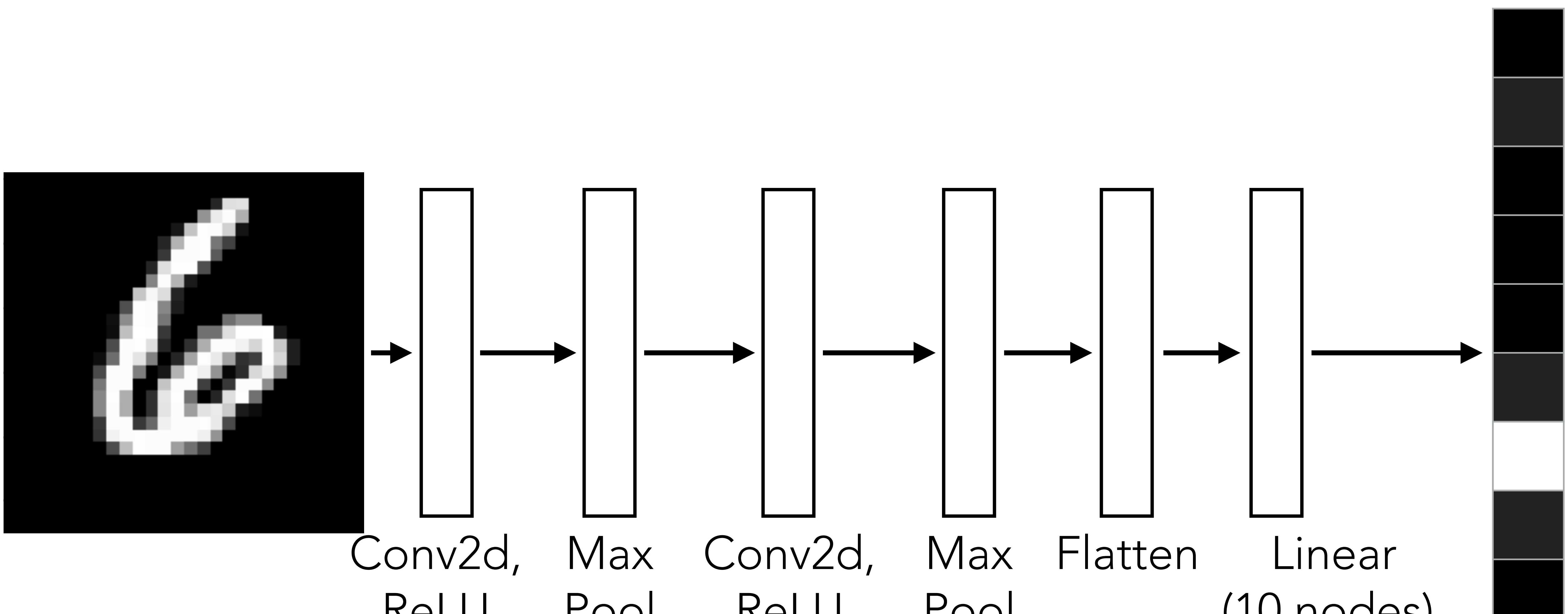


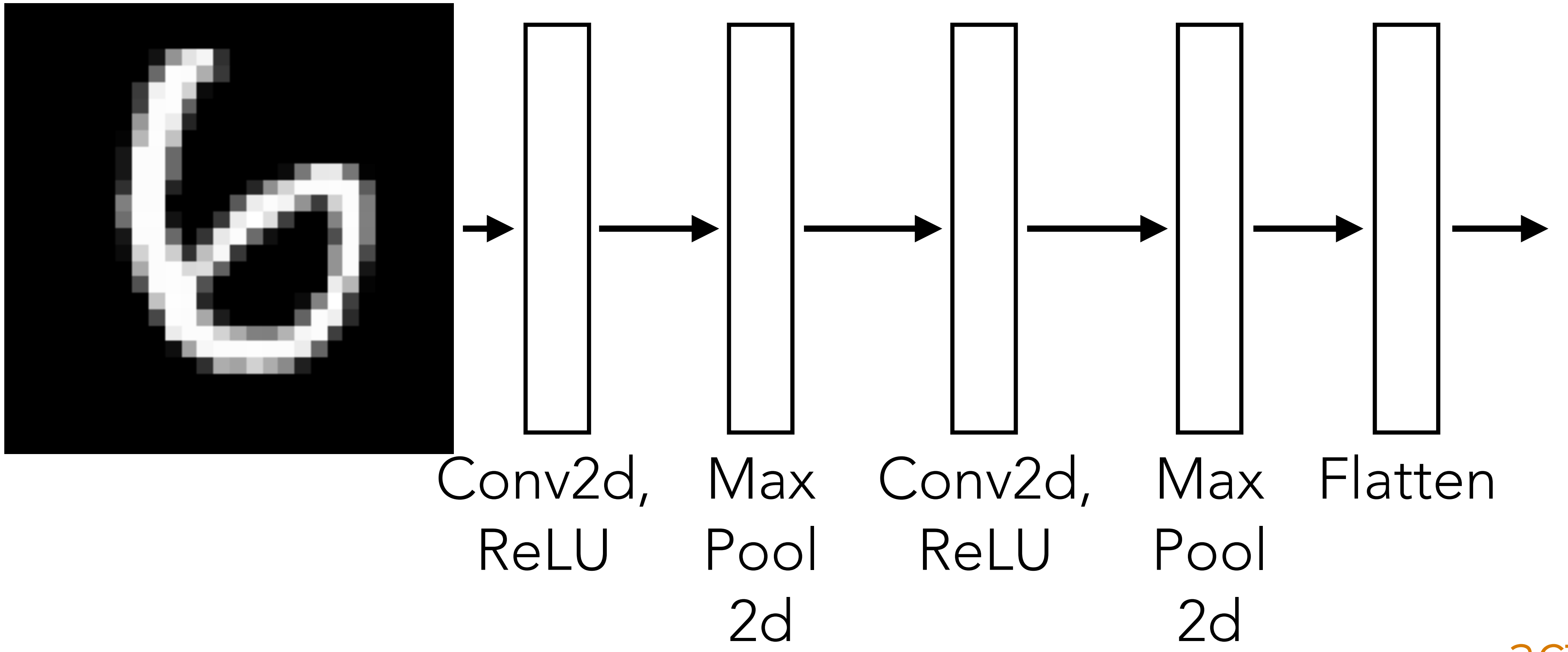
CNNs Encode Semantic Structure for Images



final output for different input 6's is similar

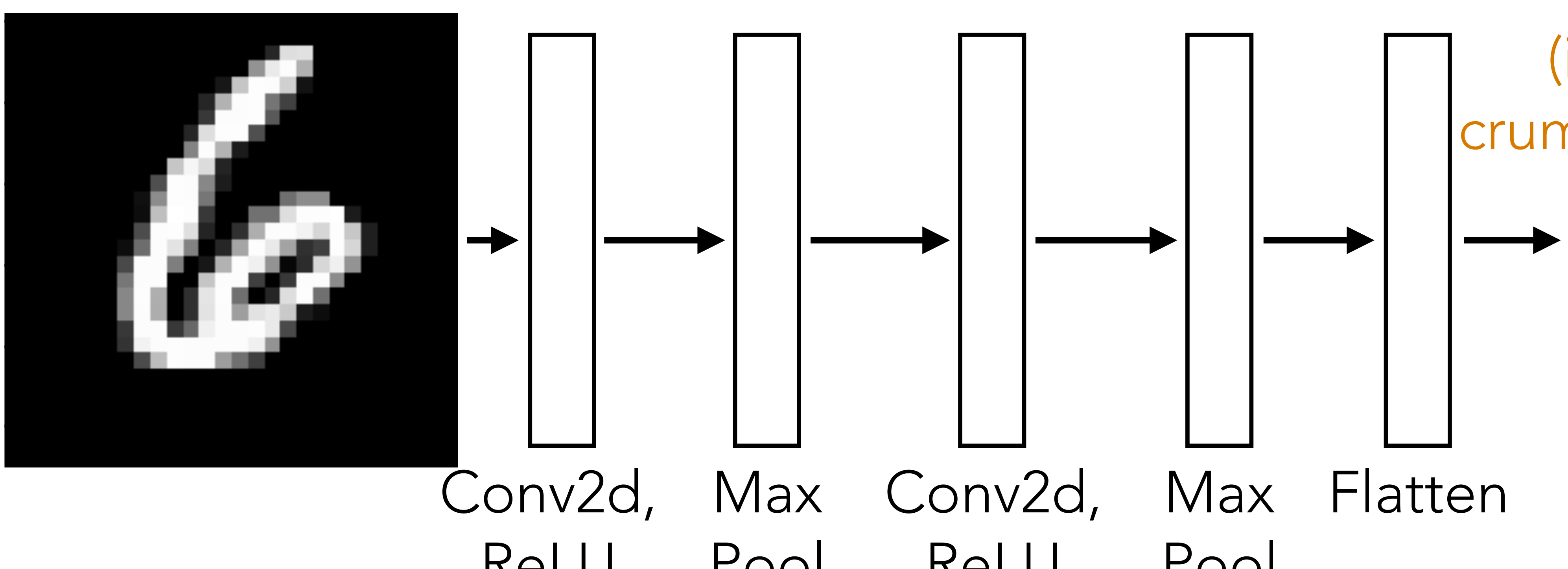

Conv2d,
ReLU
Max
Pool
2d
Conv2d,
ReLU
Max
Pool
2d
Flatten
Linear
(10 nodes),
Softmax


Conv2d,
Max
Conv2d,
Max
Flatten
Linear



actually, intermediate representations close to the last layer are also similar!

(intuition: recall the crumpled paper analogy!)



One more PyTorch thing…

Constructing PyTorch Models with `nn.Module`

(we'll need this level of detail in the next demo)

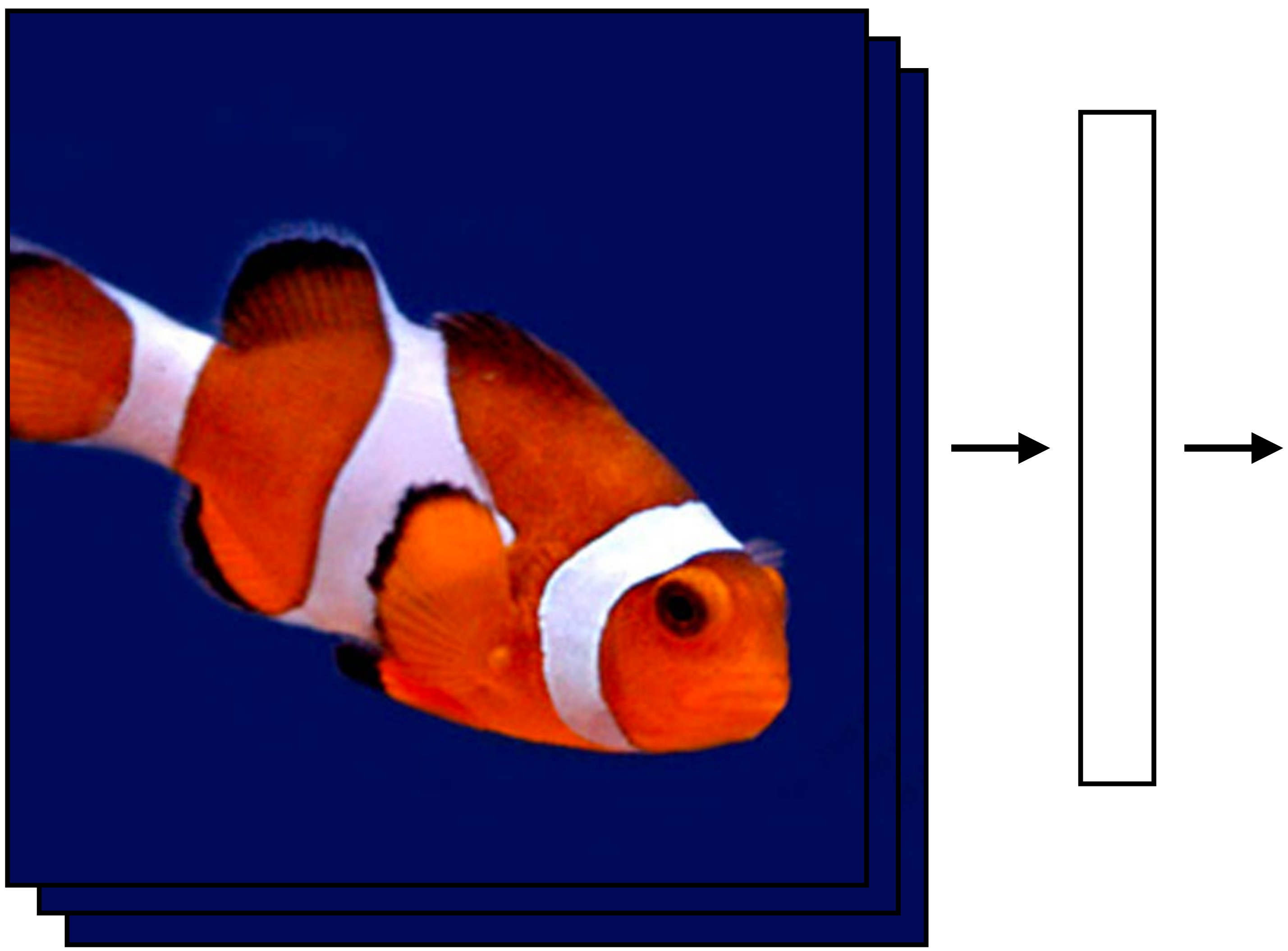
```
deeper_model = nn.Sequential(nn.Flatten(),
                             nn.Linear(in_features=784, out_features=512),
                             nn.ReLU(),
                             nn.Linear(in_features=512, out_features=10))
```

Another way to write this:

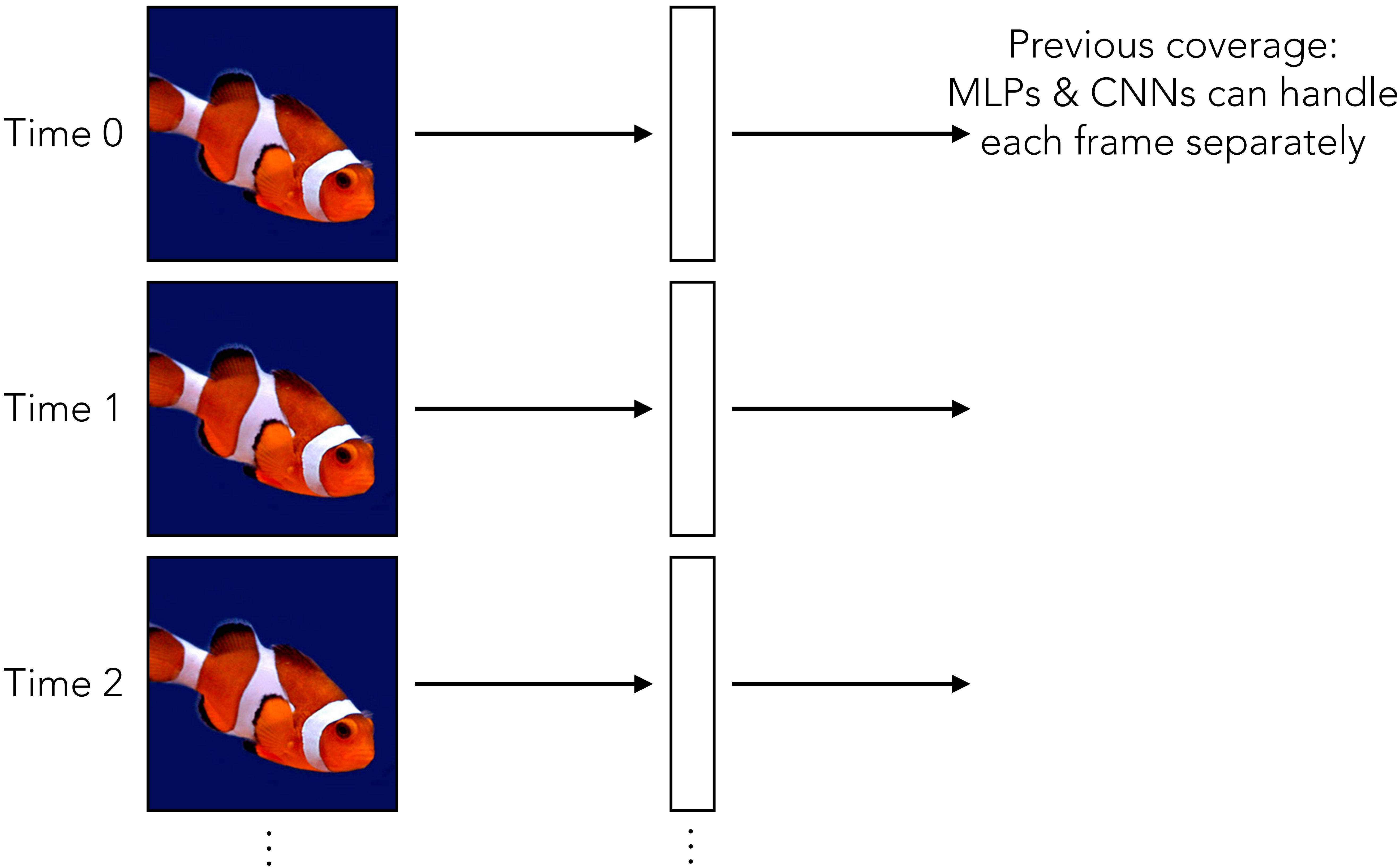
```
class DeeperModel(nn.Module):
    def __init__(self, num_in_features, num_intermediate_features, num_out_features):
        super().__init__()
        self.flatten = nn.Flatten()
        self.linear1 = nn.Linear(num_in_features, num_intermediate_features)
        self.relu = nn.ReLU()
        self.linear2 = nn.Linear(num_intermediate_features, num_out_features)

    def forward(self, inputs):
        flatten_output = self.flatten(inputs)
        linear1_output = self.linear1(flatten_output)
        relu_output = self.relu(linear1_output)
        linear2_output = self.linear2(relu_output)
        return linear2_output

deeper_model = DeeperModel(784, 512, 10)
```

Accounting for time series structure using recurrent neural networks (RNNs)

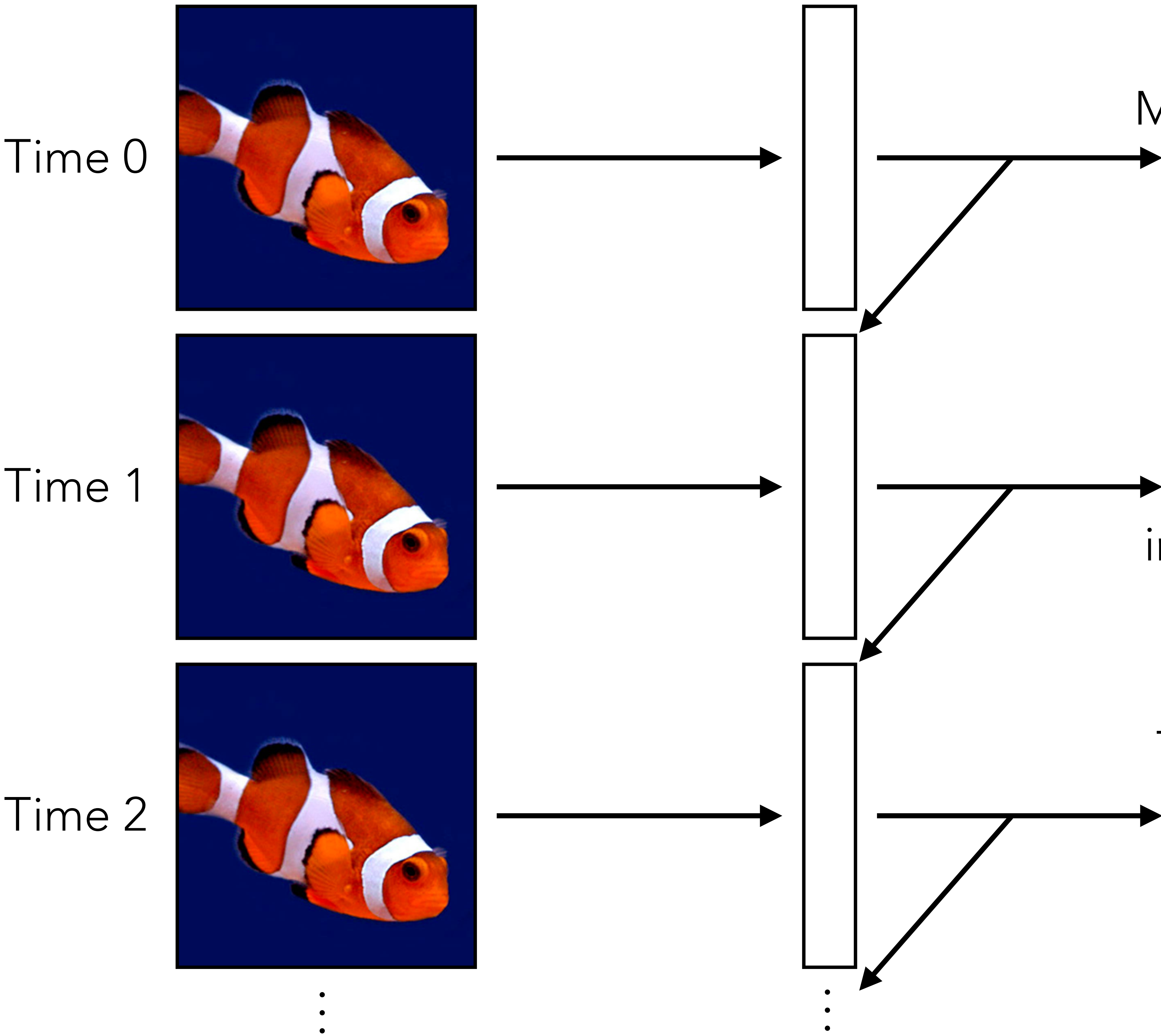
Time Series Data



Each data point is a video


Time 0
Time 1
Time 2
Previous coverage:
MLPs & CNNs can handle
each frame separately

Recurrent Neural Nets



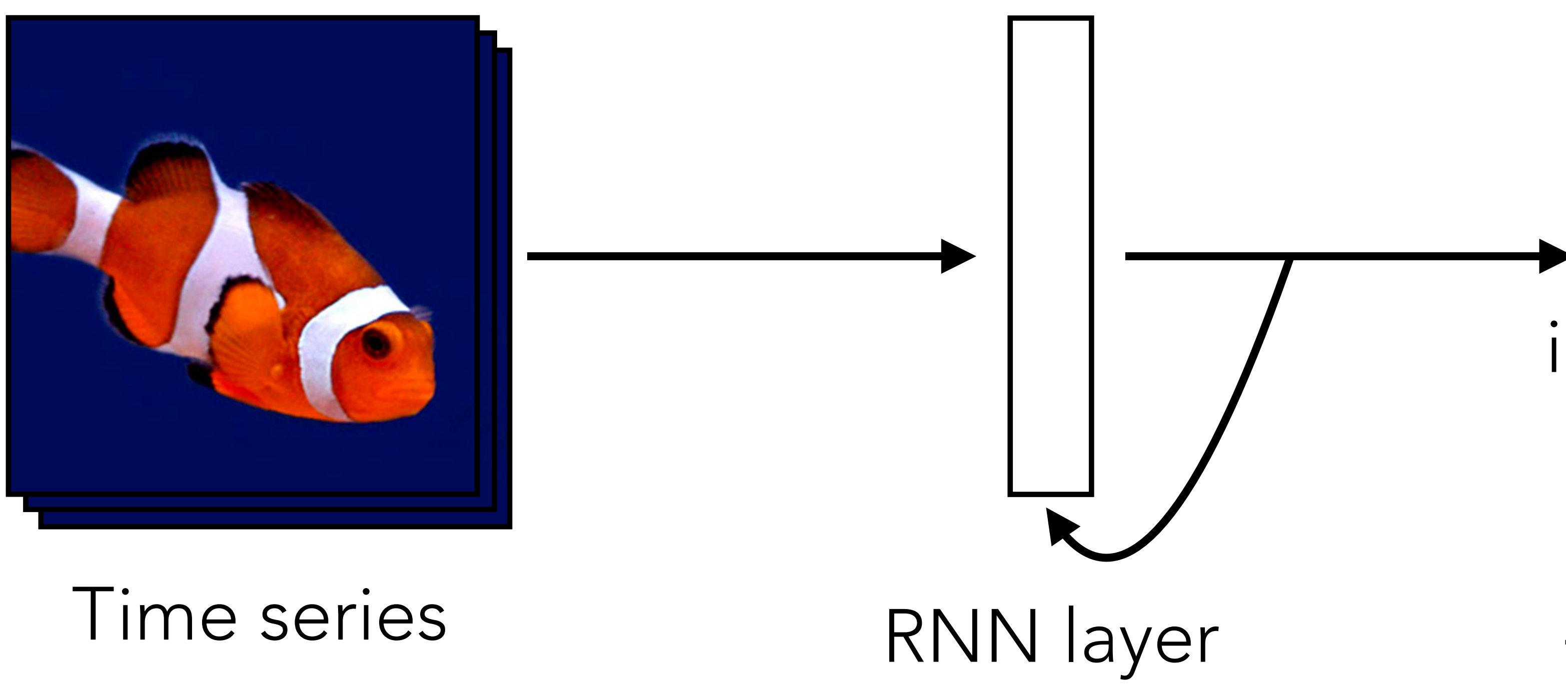
Previous coverage:
MLPs & CNNs can handle each frame separately

RNNs:
include output at previous time step as input to current time step

There are different kinds of RNNs, such as:
`RNN` (vanilla),
`LSTM`, `GRU`

Recurrent Neural Nets

Previous coverage:
MLPs & CNNs can handle each frame separately



RNNs:
include output at previous time step as input to current time step

There are different kinds of RNNs, such as:
`RNN` (vanilla),
`LSTM`, `GRU`

Vanilla ReLU RNN

input is a 1D table: num_features entries

Python list that can have any nonzero length!

memory that evolves over time; we want to learn how it changes

```
current_state = np.zeros(num_nodes)
for input in input_sequence:
    linear = np.dot(input, W.T) + b \
             + np.dot(current_state, U.T)
    output = np.maximum(0, linear) # ReLU
    current_state = output
```

W is a 2D table: # rows: num_nodes, # cols: num_features

b is a 1D table: num_nodes entries

U is a 2D table: num_nodes by num_nodes

linear is a 1D table: num_nodes entries

Parameters: weight matrices W & U, and bias vector b

How memory changes from one time step to the next is determined by an operation that looks like a linear layer followed by a nonlinear activation

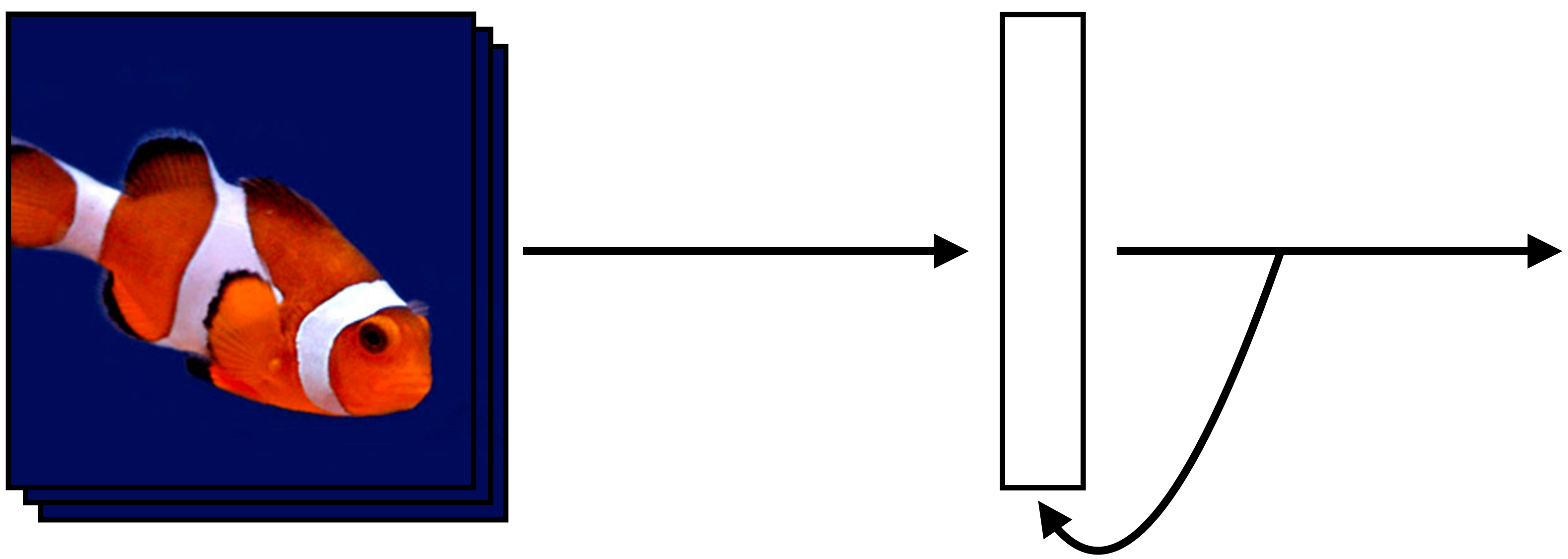
Vanilla ReLU RNN

```
current_state = np.zeros(num_nodes)
outputs = []
for input in input_sequence:
  linear = np.dot(input, W.T) + b   \
           + np.dot(current_state, U.T)
  output = np.maximum(0, linear) # ReLU
  outputs.append(output)
  current_state = output
```

In general: there is an output at every time step

For simplicity, in today's lecture, we only use the very last time step's output

Recurrent Neural Nets



Time series

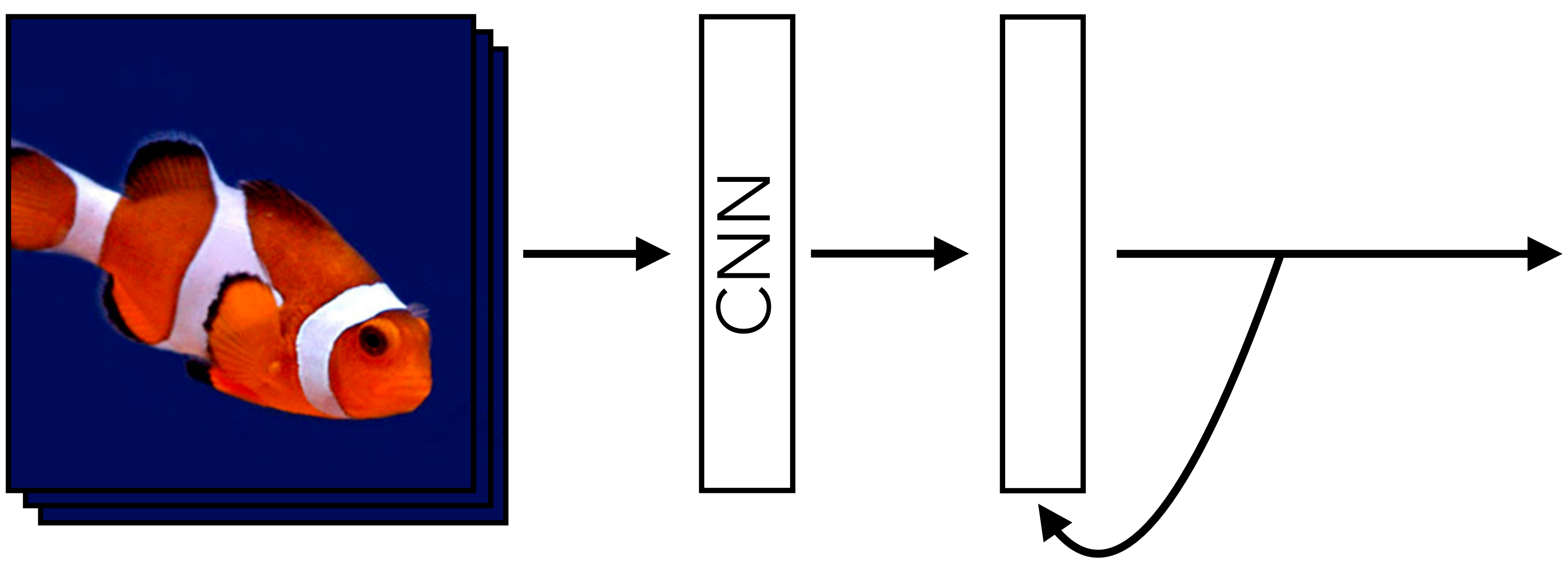
RNN layer

models how output changes over time but *does not know image or text structure!!!*

⇒ combine with other neural net layers

Recurrent Neural Nets

apply CNN to each video frame to extract semantically meaningful representation

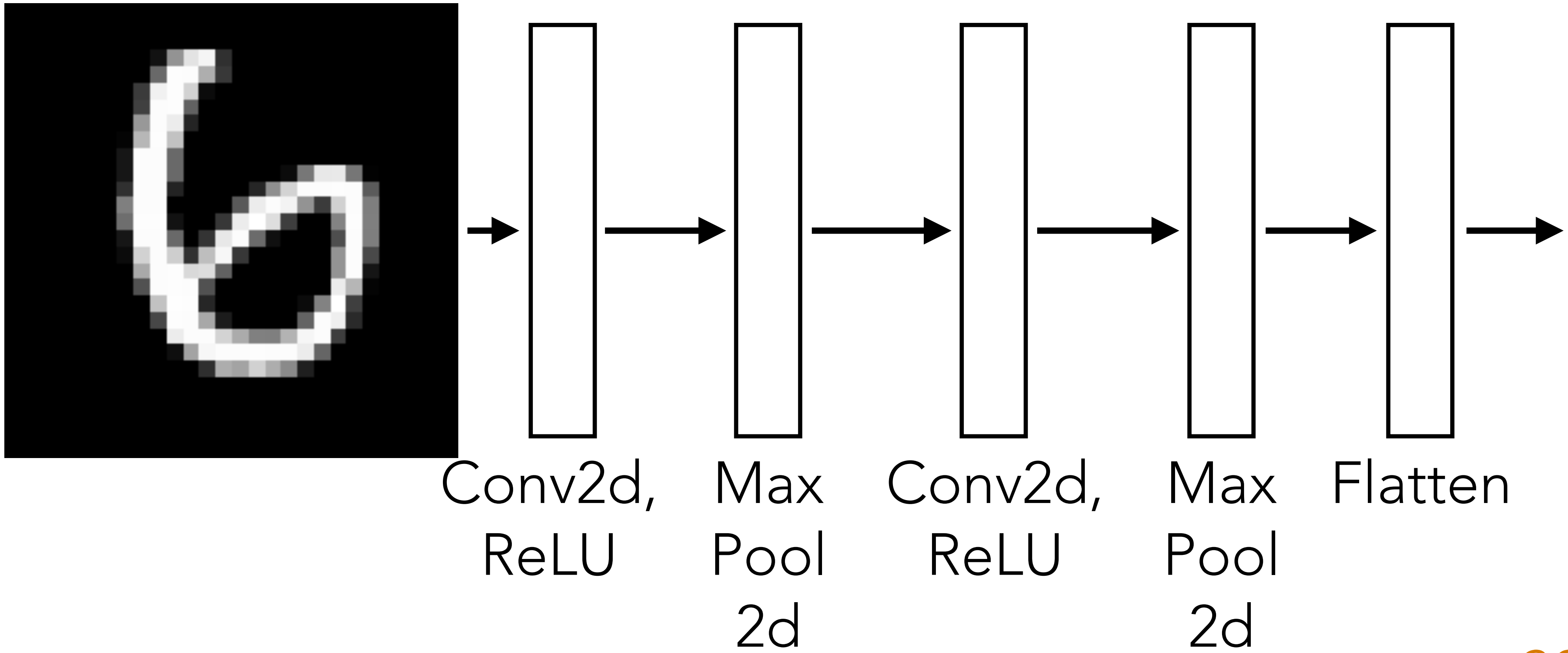


Time series

RNN layer

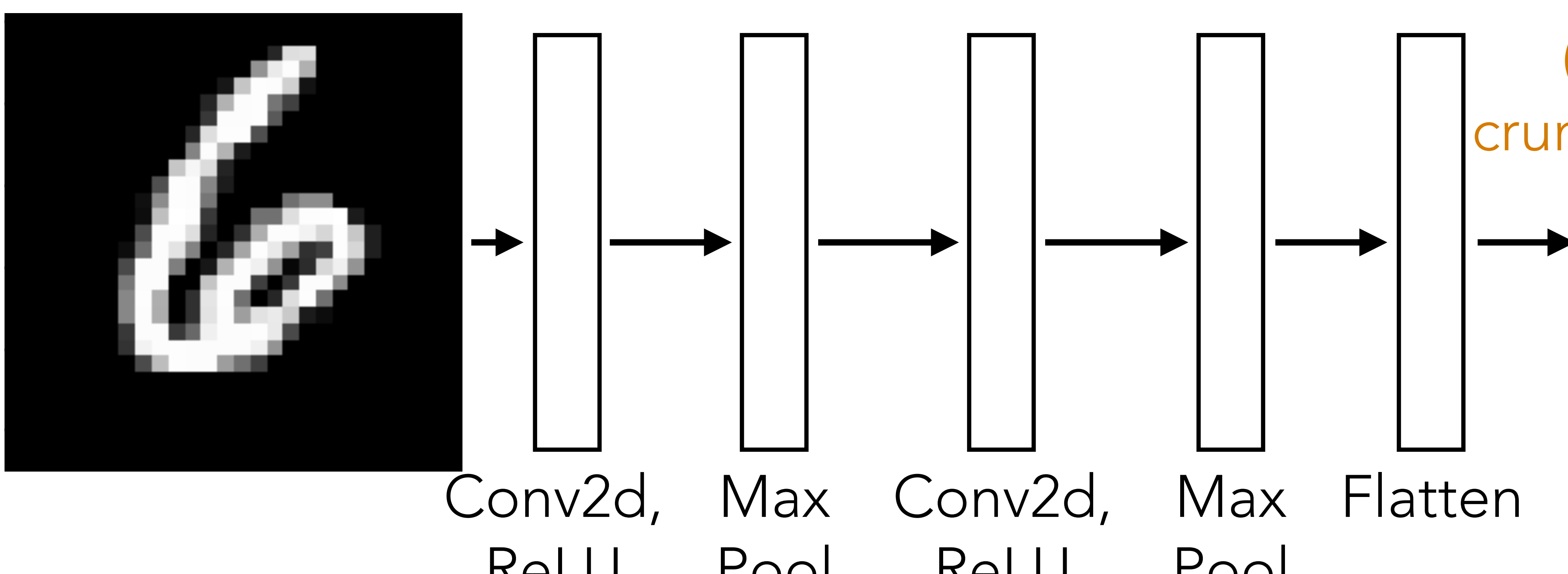
models how output changes over time but *does not know image or text structure!!!*

⇒ combine with other neural net layers



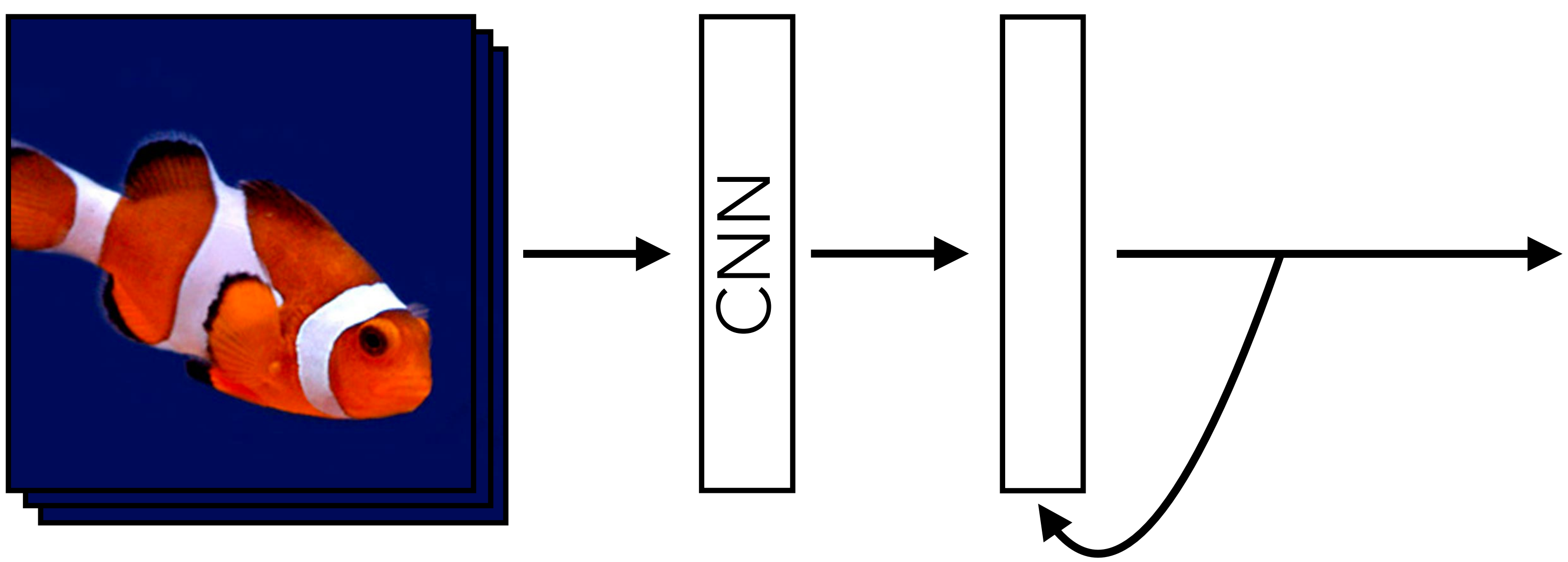
actually, intermediate representations close to the last layer are also similar!

(intuition: recall the crumpled paper analogy!)



Recurrent Neural Nets

apply CNN to each video frame to extract semantically meaningful representation



Time series

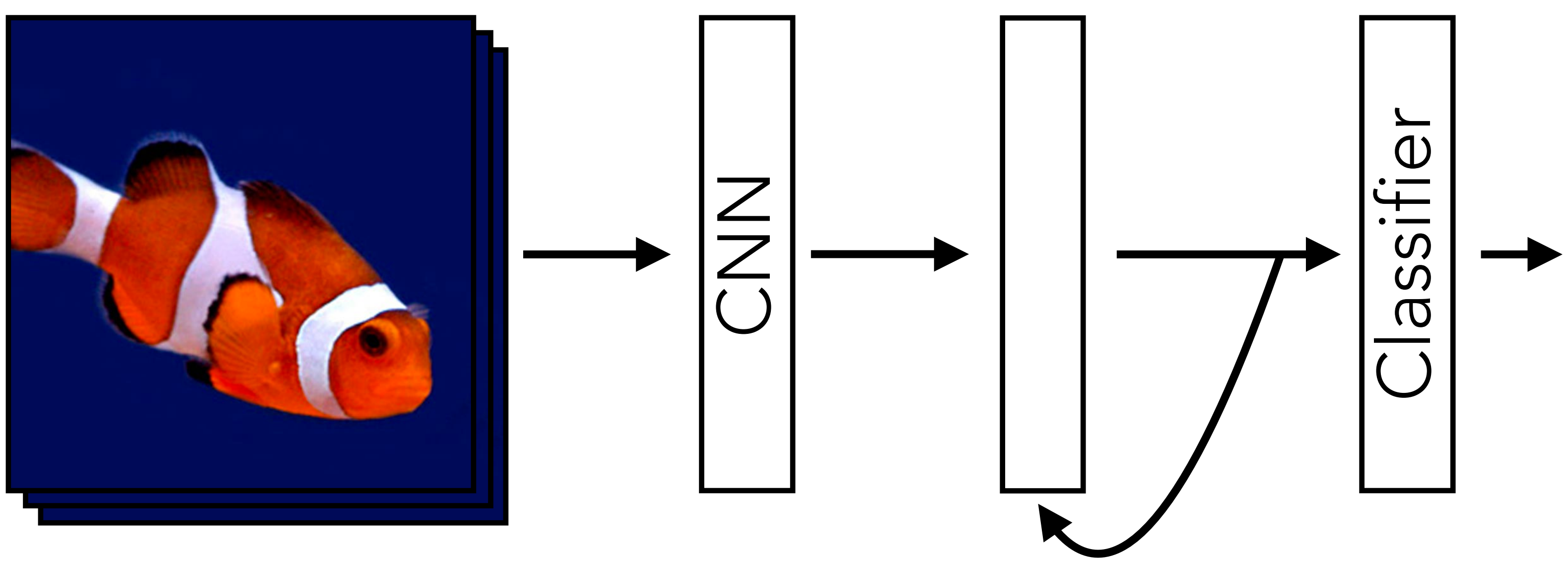
RNN layer

models how output changes over time but *does not know image or text structure!!!*

⇒ combine with other neural net layers

Recurrent Neural Nets

apply CNN to each video frame to extract semantically meaningful representation



Time series

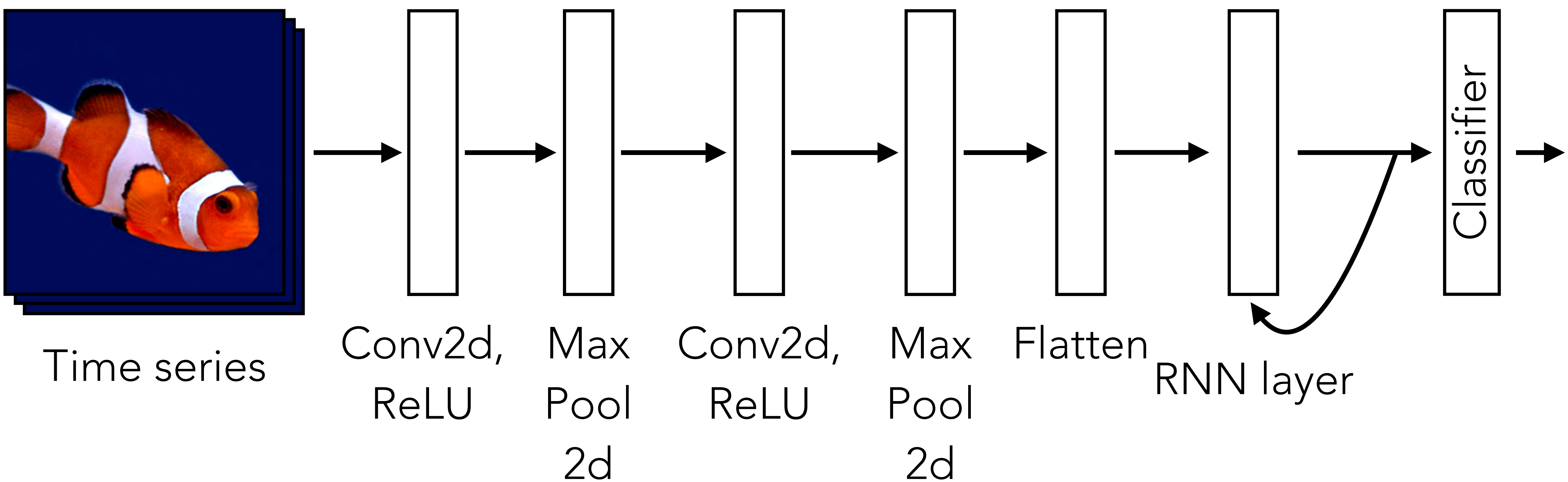
RNN layer

models how output changes over time but *does not know image or text structure!!!*

⇒ combine with other neural net layers

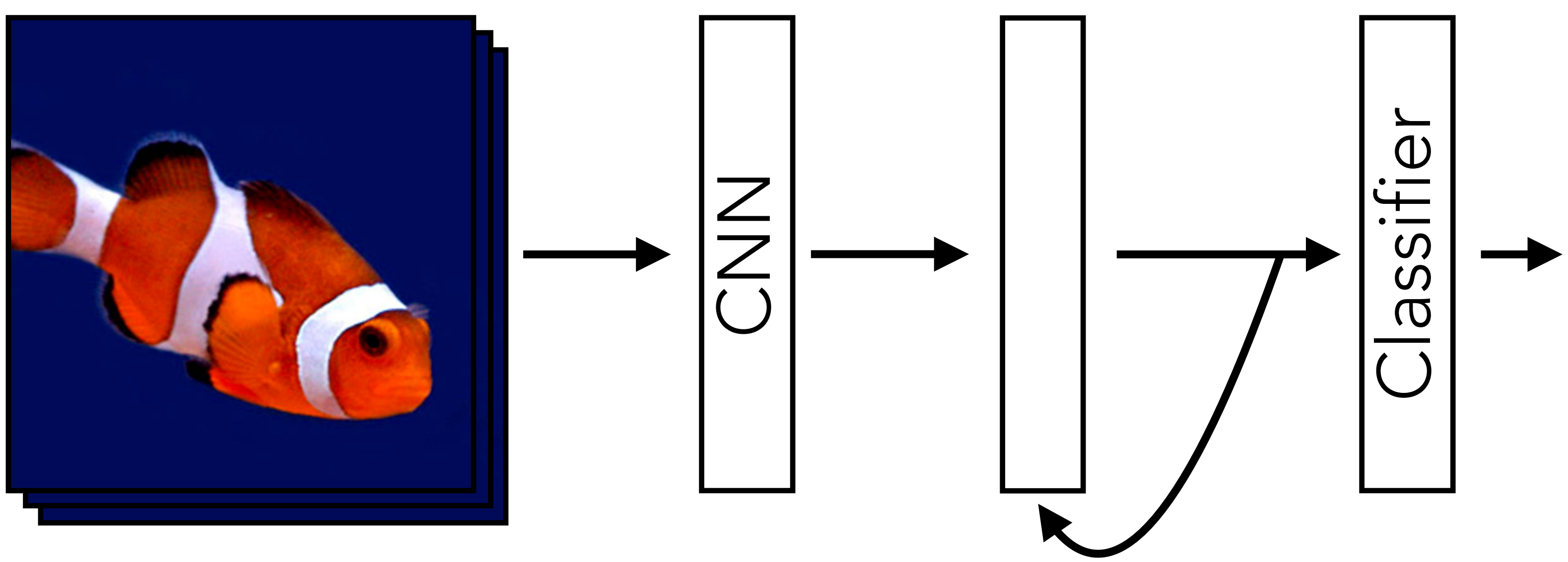
Recurrent Neural Nets

apply CNN to each video frame to extract semantically meaningful representation



Recurrent Neural Nets

apply CNN to each video
frame to extract
semantically meaningful
representation



Time series

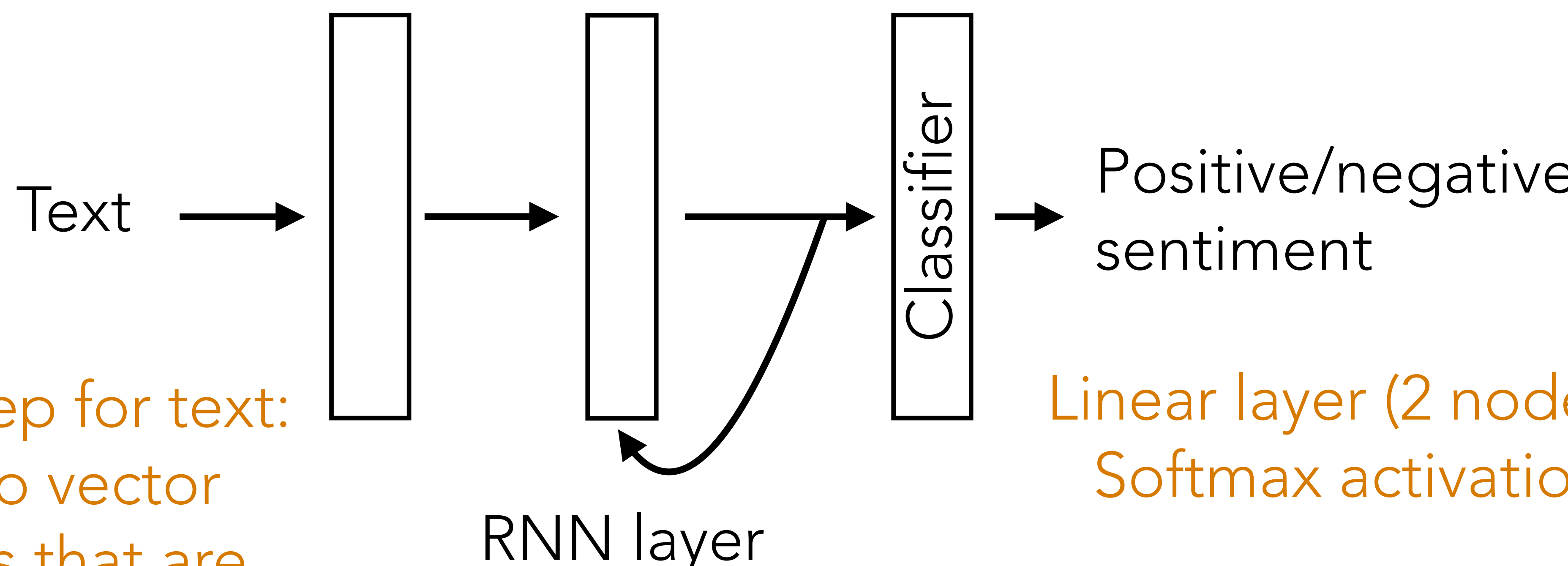
RNN layer

models how output changes
over time but *does not know*
image or text structure!!!

⇒ combine with other neural net layers

Recurrent Neural Nets

Example: Given text (e.g., movie review, Tweet), figure out whether it has positive or negative sentiment (binary classification)



Common first step for text: turn words into vector representations that are semantically meaningful

Linear layer (2 nodes), Softmax activation

label 0: negative sentiment

label 1: positive sentiment

(Flashback) Do Data Actually Live on Manifolds?

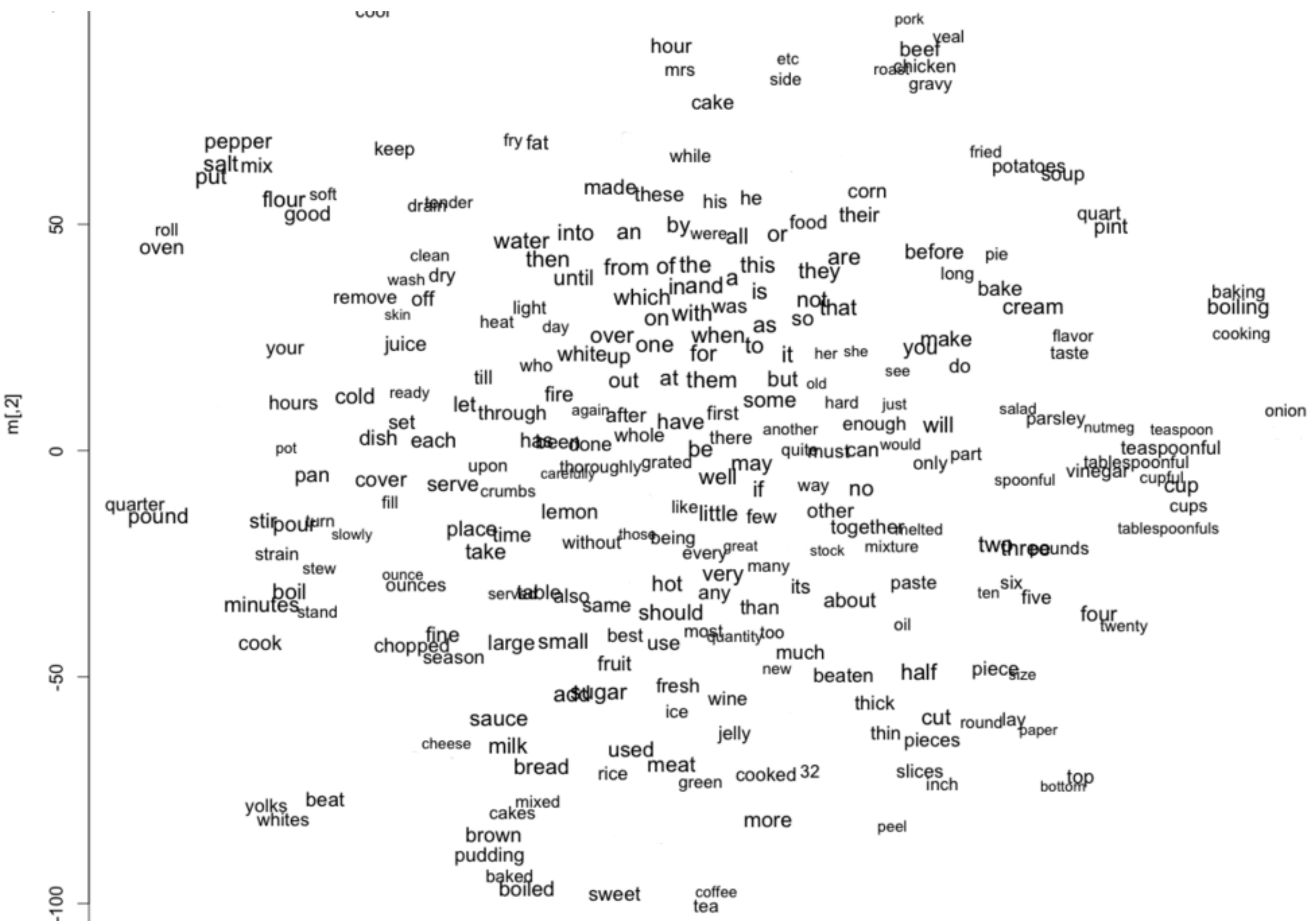
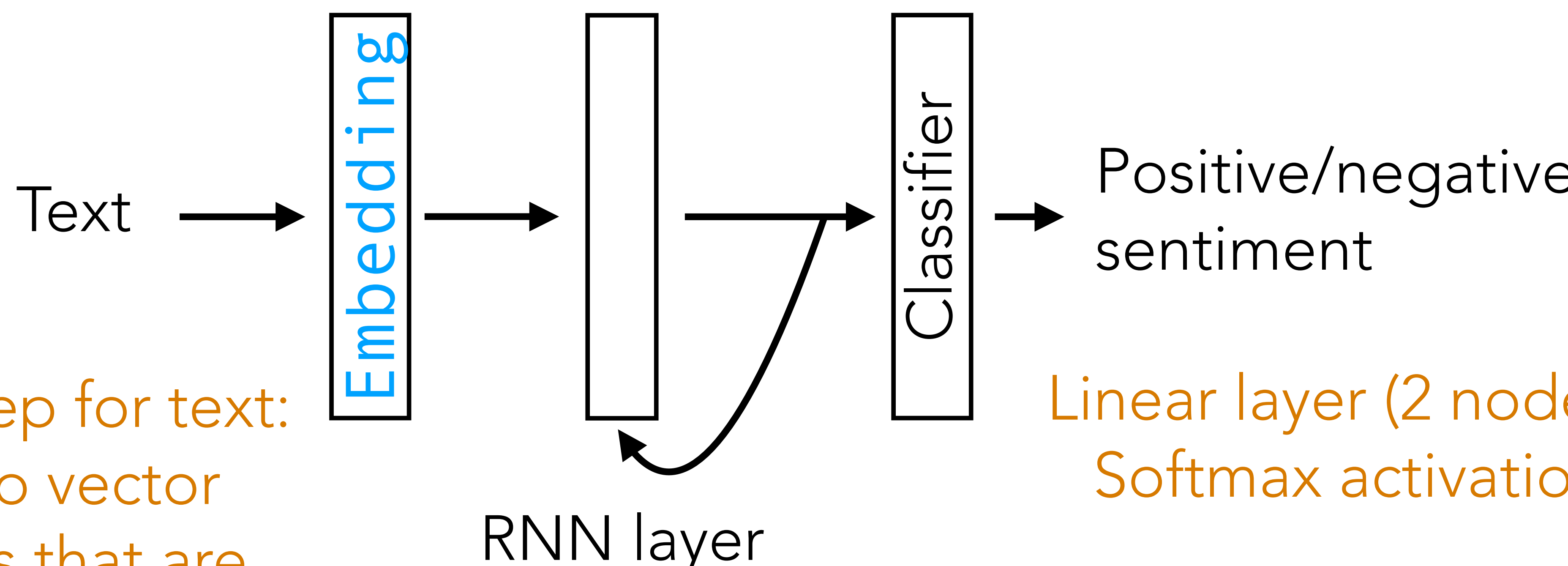


Image source: http://www.adityathakker.com/wp-content/uploads/2017/06/word-embeddings-994x675.png

Recurrent Neural Nets

Example: Given text (e.g., movie review, Tweet), figure out whether it has positive or negative sentiment (binary classification)



Common first step for text: turn words into vector representations that are semantically meaningful

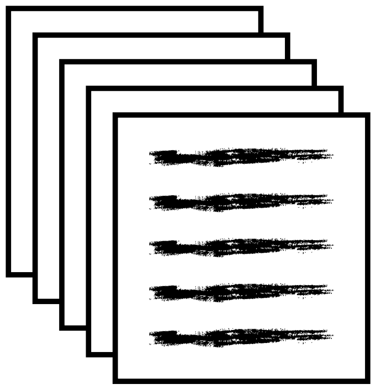
In PyTorch, use the `Embedding` layer

Linear layer (2 nodes), Softmax activation

label 0: negative sentiment

label 1: positive sentiment

Sentiment Analysis with IMDb Reviews



Training reviews

Step 1: Tokenize & build vocabulary

Word index	Word	2D Embedding
0	this	[-0.57, 0.44]
1	movie	[0.38, 0.15]
2	rocks	[-0.85, 0.70]
3	sucks	[-0.26, 0.66]

Step 2: Encode each review as a sequence of word indices into the vocab

Ordering of words matters

Different reviews can have different lengths

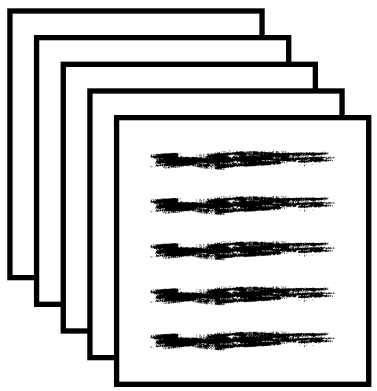
"this movie rocks" → 0 1 2

"this movie sucks" → 0 1 3

"this sucks" → 0 3

Step 3: Use word embeddings to represent each word

Sentiment Analysis with IMDb Reviews



Training reviews

Step 1: Tokenize & build vocabulary

Word index	Word	2D Embedding
0	this	[-0.57, 0.44]
1	movie	[0.38, 0.15]
2	rocks	[-0.85, 0.70]
3	sucks	[-0.26, 0.66]

Step 2: Encode each review as a sequence of word indices into the vocab

"this movie sucks" → 0 1 3

Step 3: Use word embeddings to represent each word

[-0.57, 0.44]
[0.38, 0.15]
[-0.26, 0.66]