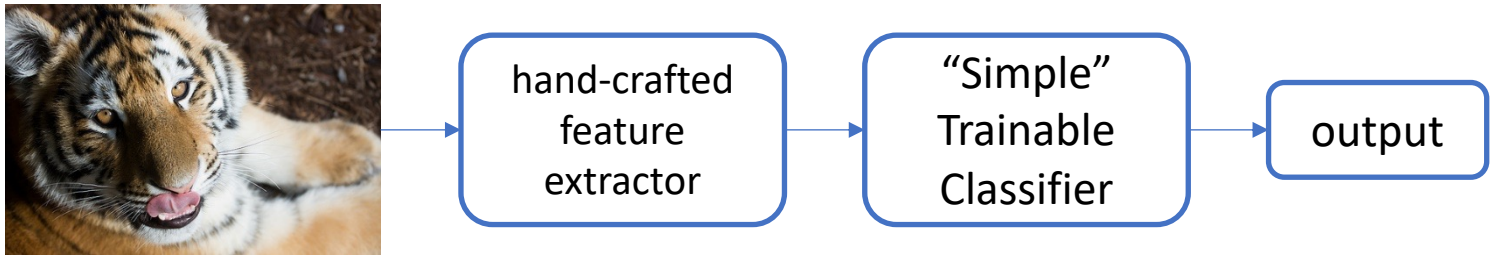
Deep neural networks

Introduction

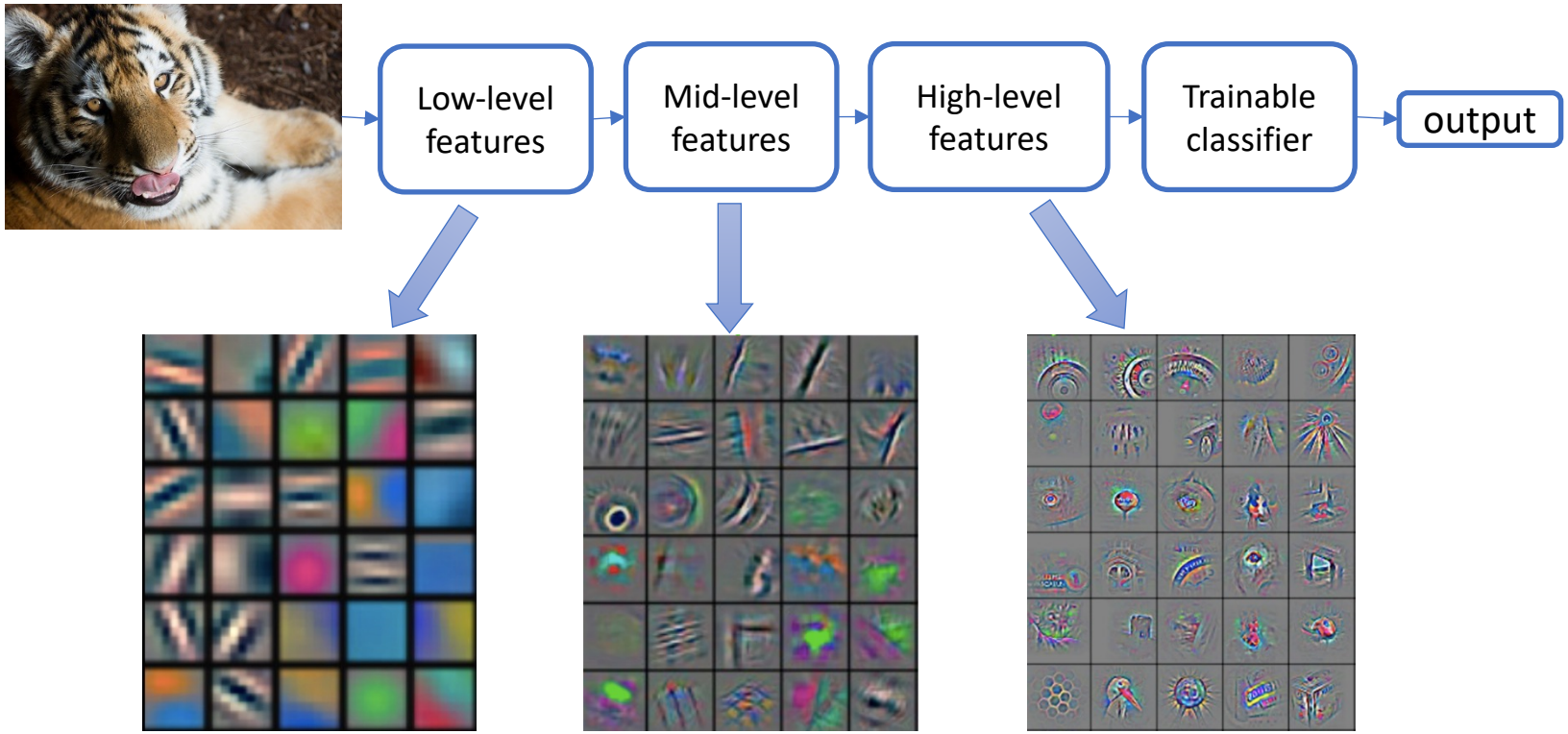
- Traditional machine learning models use hand-crafted features and relatively simple trainable classifier.



- These approaches have the following limitations:
  - It is very tedious and costly to develop hand-crafted features
  - The hand-crafted features are usually highly dependent on one application, and cannot be transferred easily to other applications

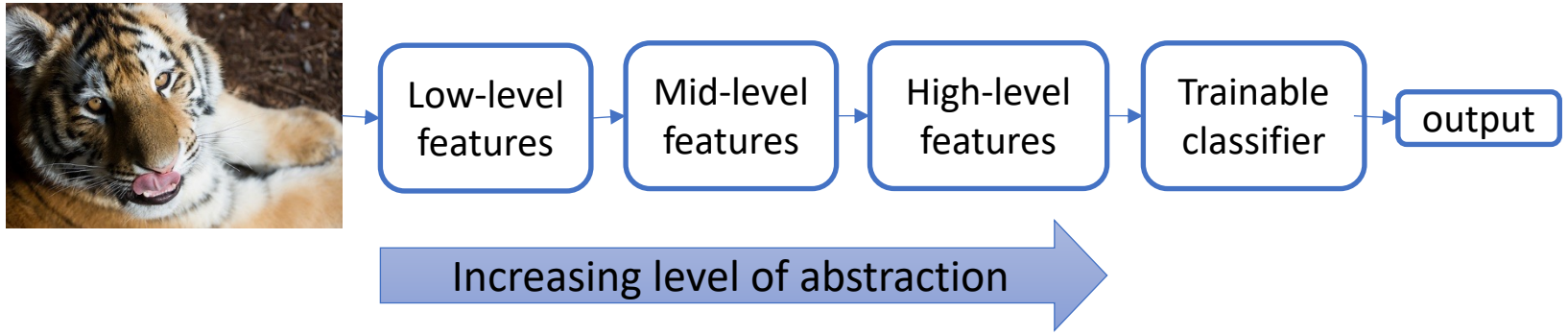
Deep Learning

- Deep learning (a.k.a. representation learning) seeks to learn rich hierarchical representations (i.e. features) automatically through multiple stages of feature learning process.



Feature visualization of convolutional net trained on ImageNet (Zeiler and Fergus, 2013)

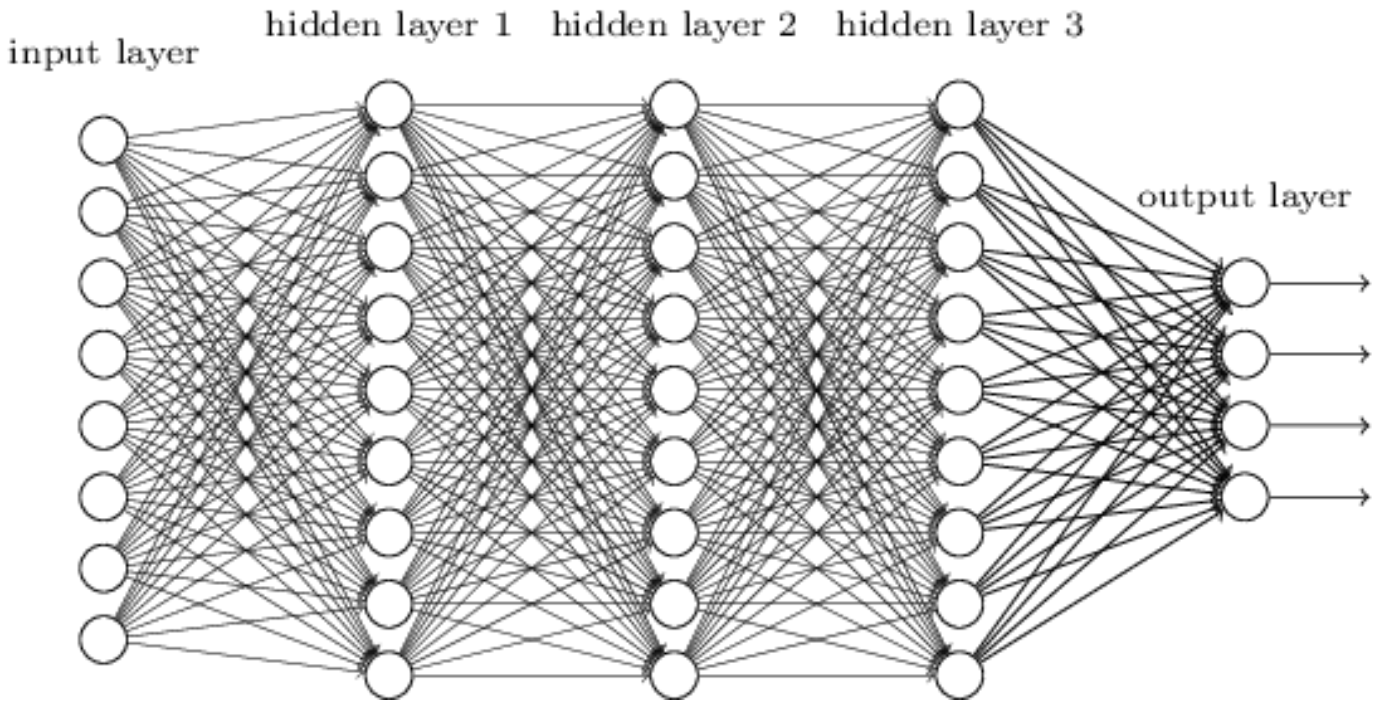
Learning Hierarchical Representations



- Hierarchy of representations with increasing level of abstraction. Each stage is a kind of trainable nonlinear feature transform
- Image recognition
  - Pixel → edge → texton → motif → part → object
- Text
  - Character → word → word group → clause → sentence → story

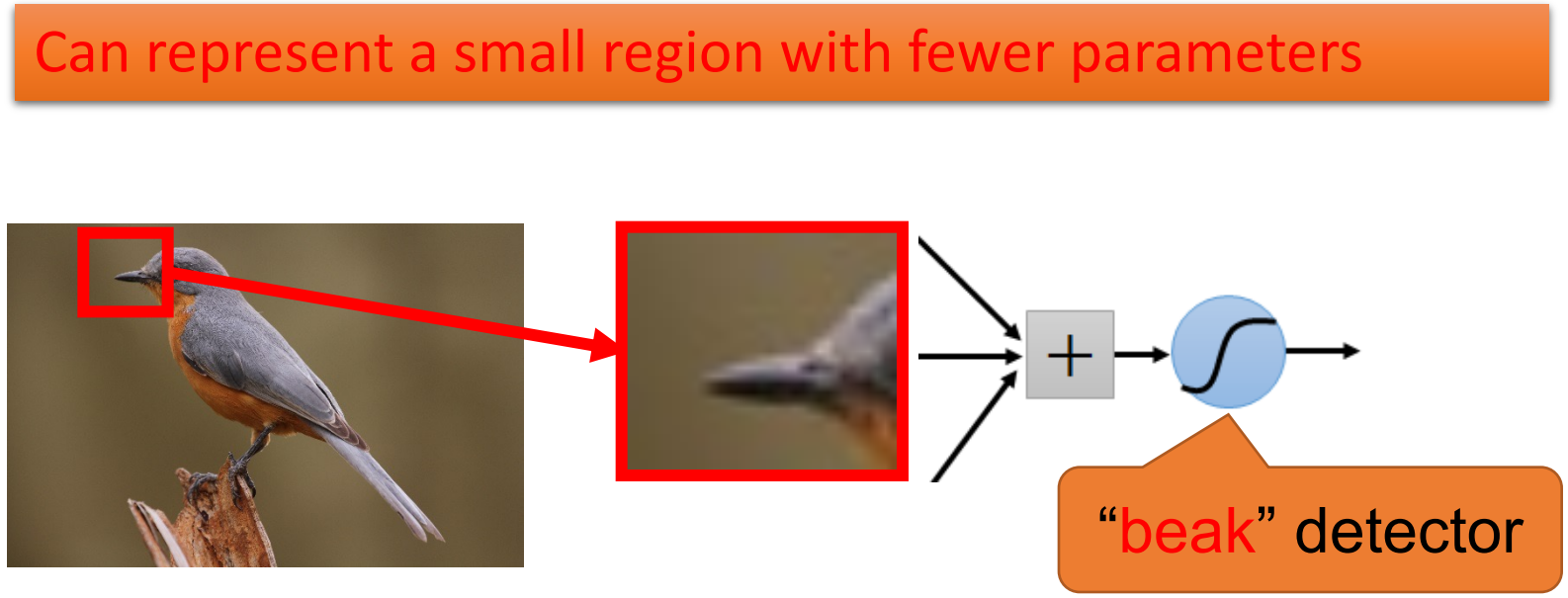
Convolutional Neural Networks

- From this fully connected model, do we really need all the edges?
- Can some of these be shared?

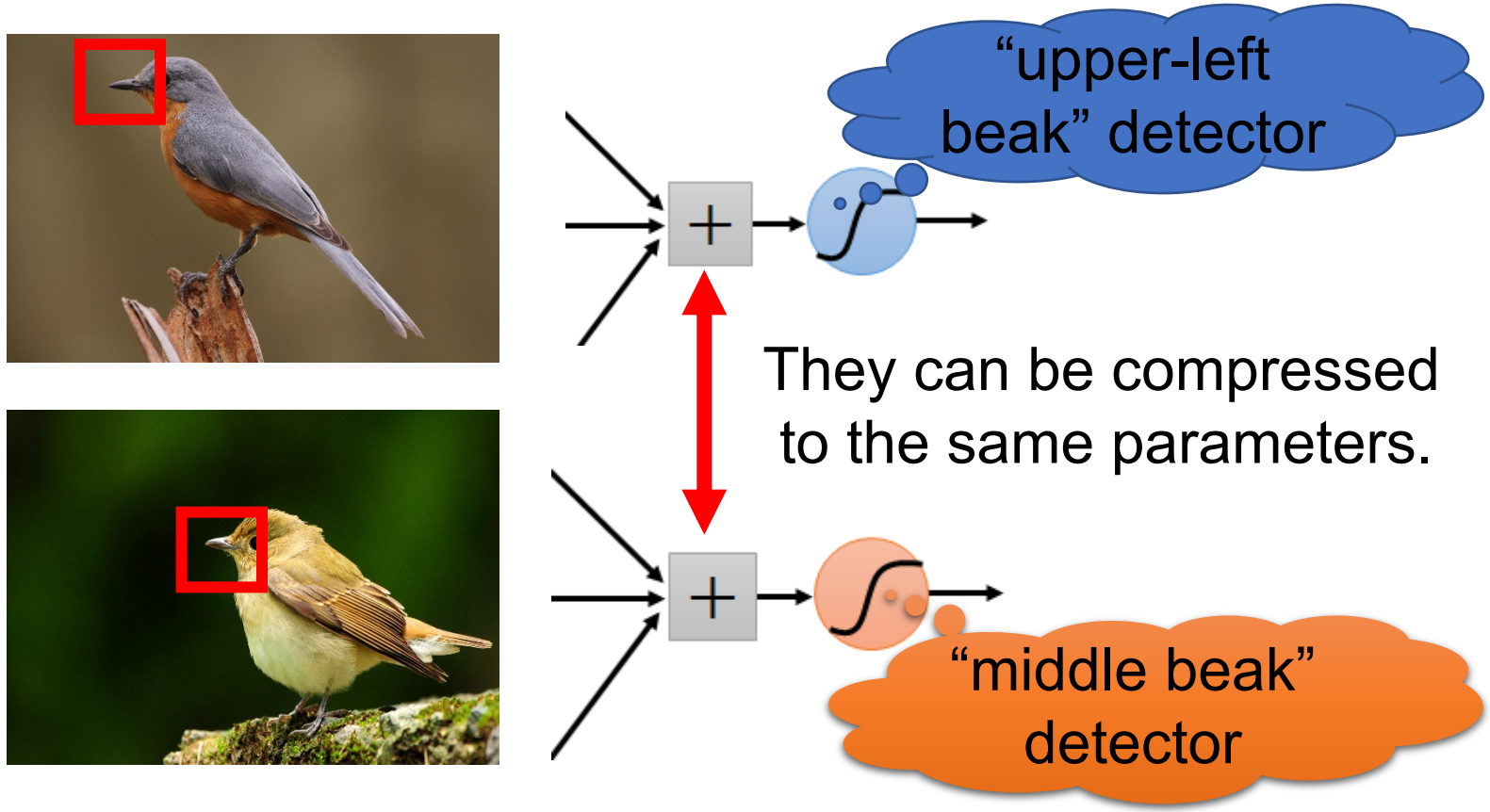


Consider learning an image:

- Some patterns are much smaller than the whole image

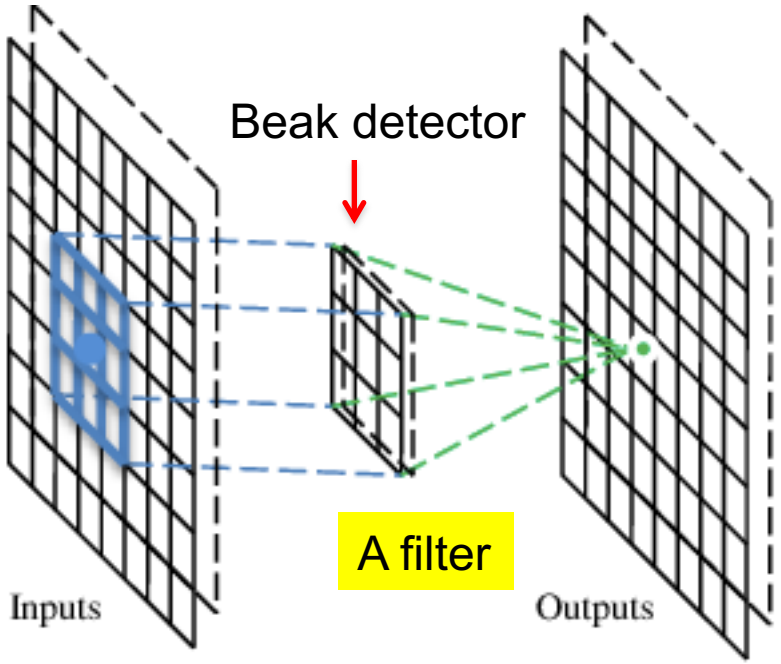


Same pattern appears in different places:
They can be compressed!
What about training a lot of such “small” detectors
and each detector must “move around”.



A convolutional layer

A CNN is a neural network with some convolutional layers (and some other layers). A convolutional layer has a number of filters that does convolutional operation.



Main CNN idea for text:
Compute vectors for n-grams and group them afterwards

Example: “This takes too long” compute vectors for:
This takes, takes too, too long, this takes too, takes too long, this takes too long

Convolution

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

These are the network parameters to be learned.

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

-1	1	-1
-1	1	-1
-1	1	-1

Filter 2

⋮ ⋮

Each filter detects a small pattern (3 x 3).

Convolution

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

stride=1

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

Dot product →

3 -1

6 x 6 image

Convolution

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

If stride=2

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

3 -3

Convolution

-1	1	-1
-1	1	-1
-1	1	-1

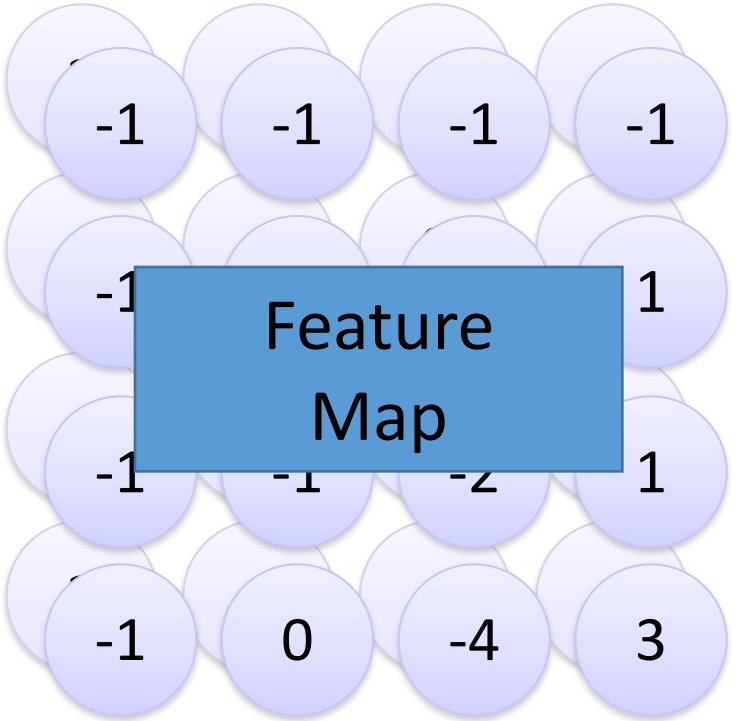
Filter 2

stride=1

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

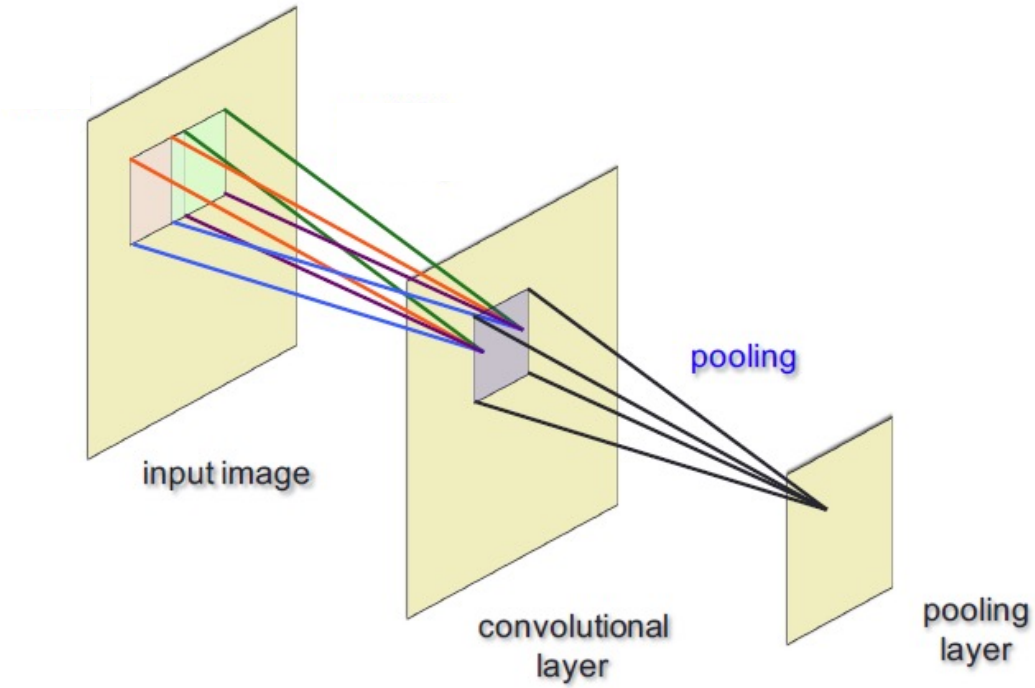
6 x 6 image

Repeat this for each filter

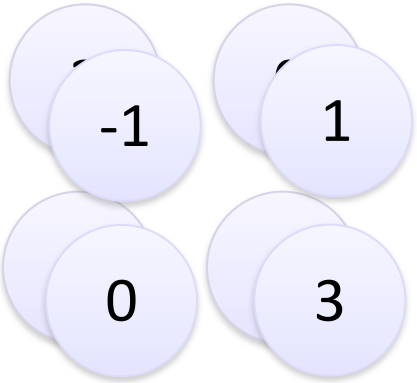


Pooling

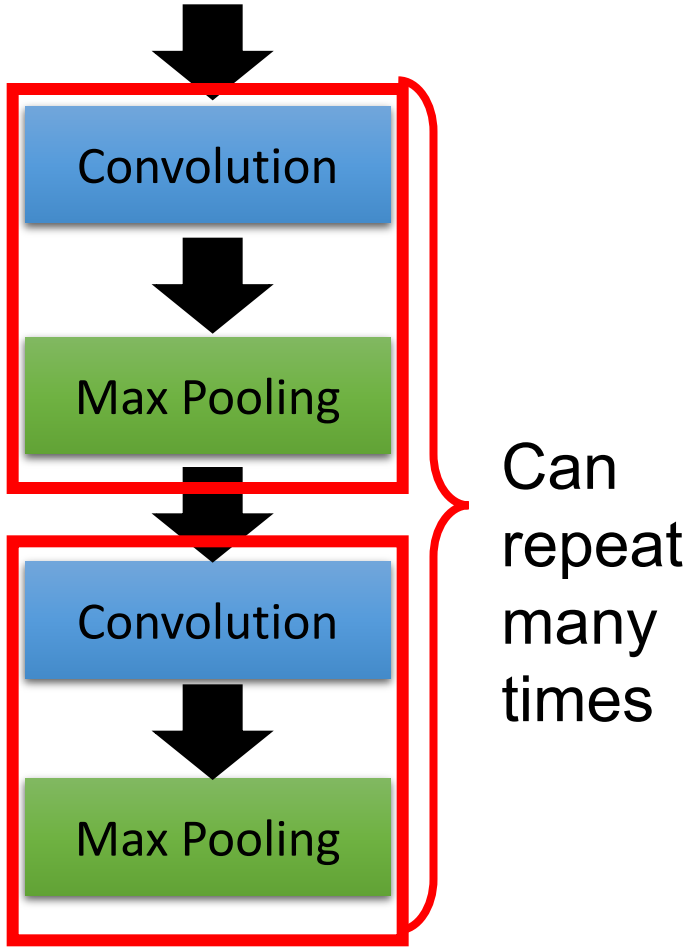
- Common pooling operations:
  - Max pooling: reports the maximum output within a rectangular neighborhood.
  - Average pooling: reports the average output of a rectangular neighborhood (possibly weighted by the distance from the central pixel).



The whole CNN



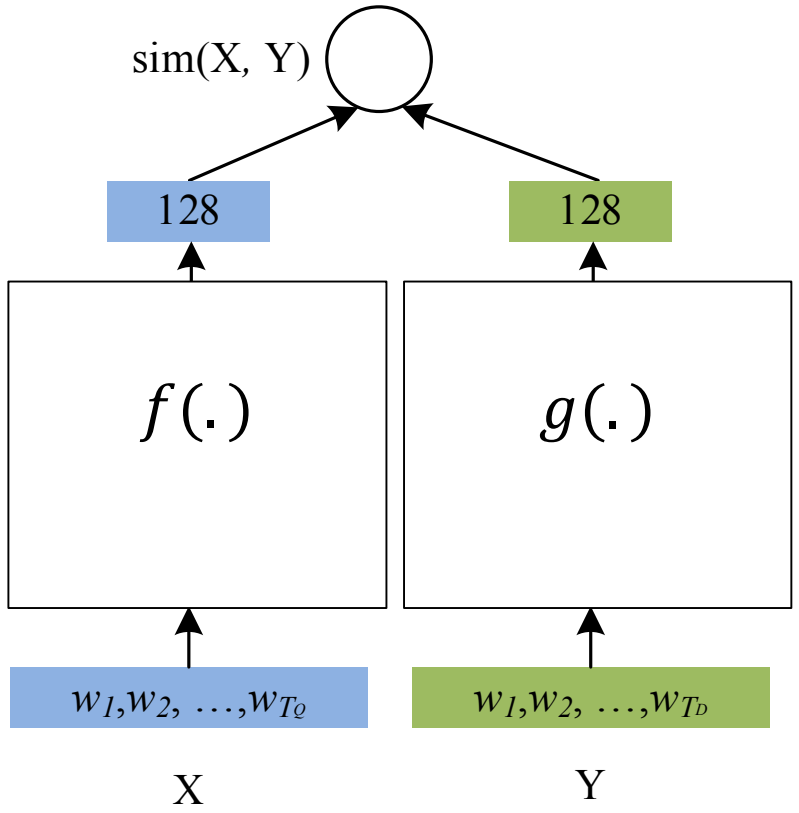
Smaller than the original image



Can repeat many times

Deep Structured Semantic Models (DSSM)

Relevance measured by cosine similarity



Word sequence x_t

Learning: maximize the similarity between X (source) and Y (target)

Representation: use DNN to extract abstract semantic features, f or g is a

- Multi-Layer Perceptron (MLP) if text is a bag of words [Huang+ 13]
- **Convolutional Neural Network (CNN) if text is a bag of chunks** [Shen+ 14]
- Recurrent Neural Network (RNN) if text is a sequence of words [Palangi+ 16]

Deep Structured Semantic Models (DSSM)

Relevance measured by cosine similarity

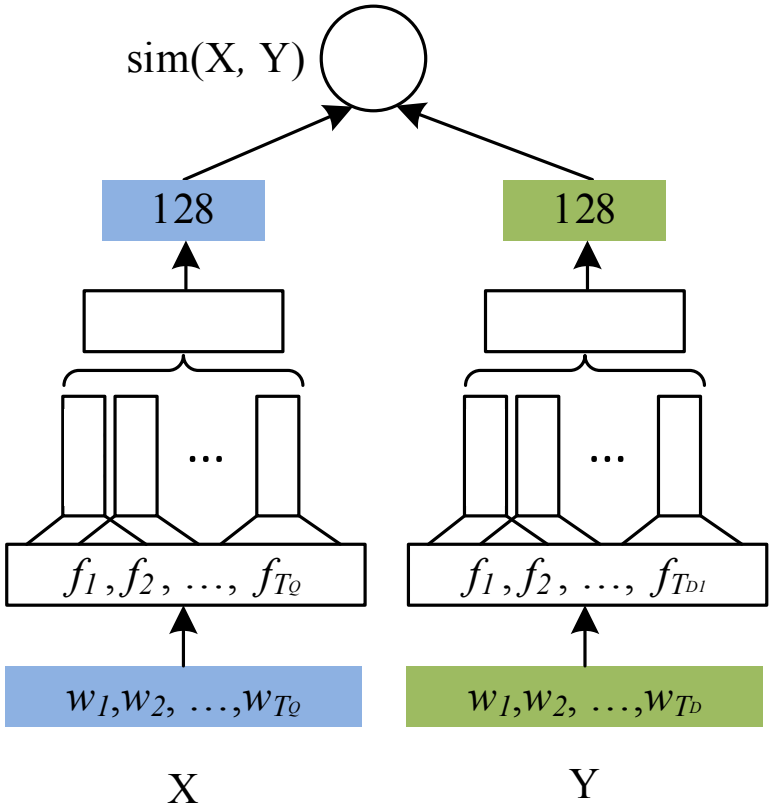
Semantic layer h

Max pooling layer v

Convolutional layer c_t

Word hashing layer f_t

Word sequence x_t



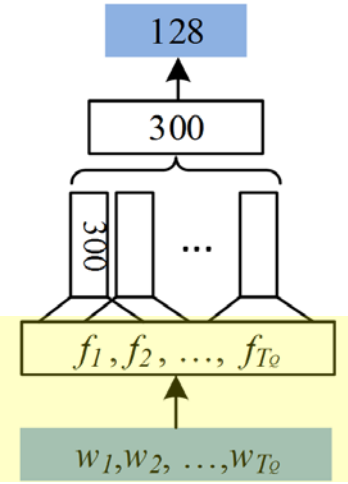
Learning: maximize the similarity between X (source) and Y (target)

Representation: use DNN to extract abstract semantic representations

Convolutional and Max-pooling layer: identify key words/concepts in X and Y

Word hashing: use sub-word unit (e.g., letter n-gram) as raw input to handle very large vocabulary

Deep Structured Semantic Models (DSSM)



- Letter-trigram Representation
- Control the dimensionality of the input space
  - e.g., cat → #cat# → #-c-a, c-a-t, a-t-#
  - Only ~50K letter-trigrams in English
- Capture sub-word semantics (e.g., prefix & suffix)
- Words with small typos have similar raw representations
- Collision: different words with same letter-trigram representation?

Vocabulary size	# of unique letter-trigrams	# of Collisions	Collision rate
40K	10,306	2	0.0050%
500K	30,621	22	0.0044%
5M	49,292	179	0.0036%

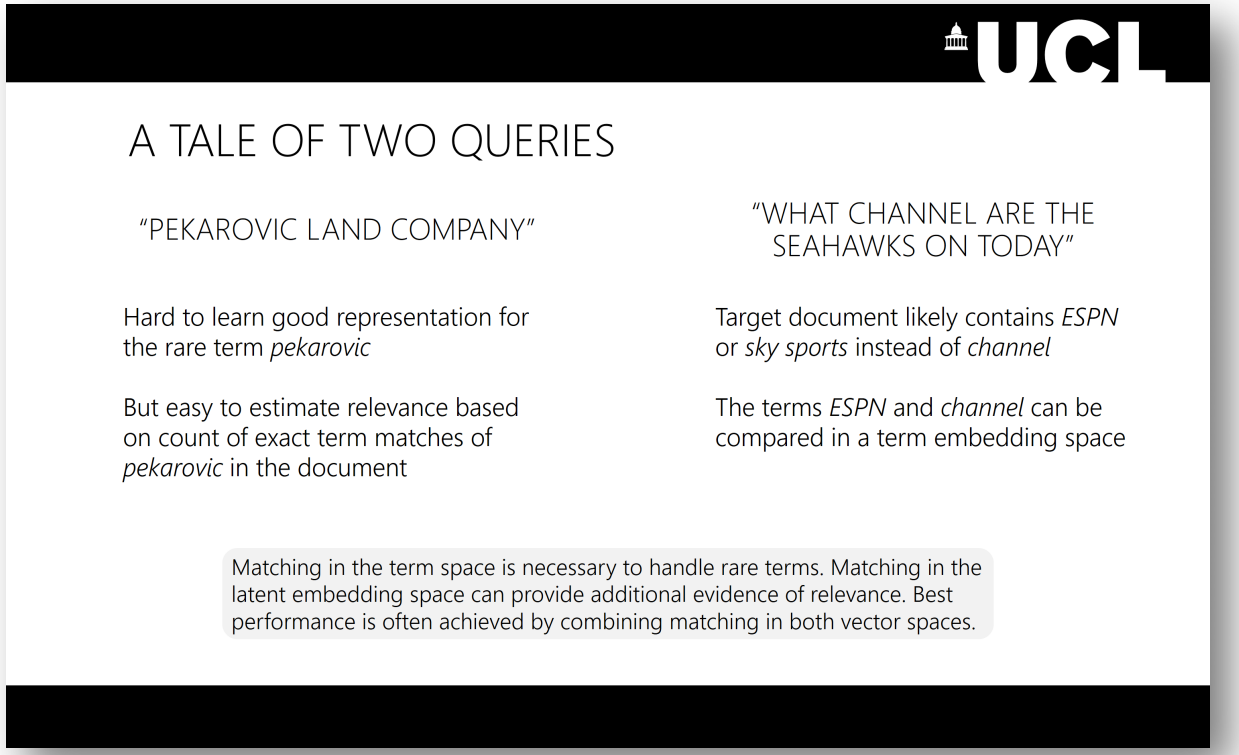
Deep Structured Semantic Models (DSSM) Variants

- To train the model we can use any of the loss functions we learned about in the last lecture
  - Cross-entropy loss against randomly sampled negative documents (in addition to relevant documents) is commonly used

- Input could be letter-trigram (original version), one-hot encoding of the words (terms) or word embeddings

Remember…

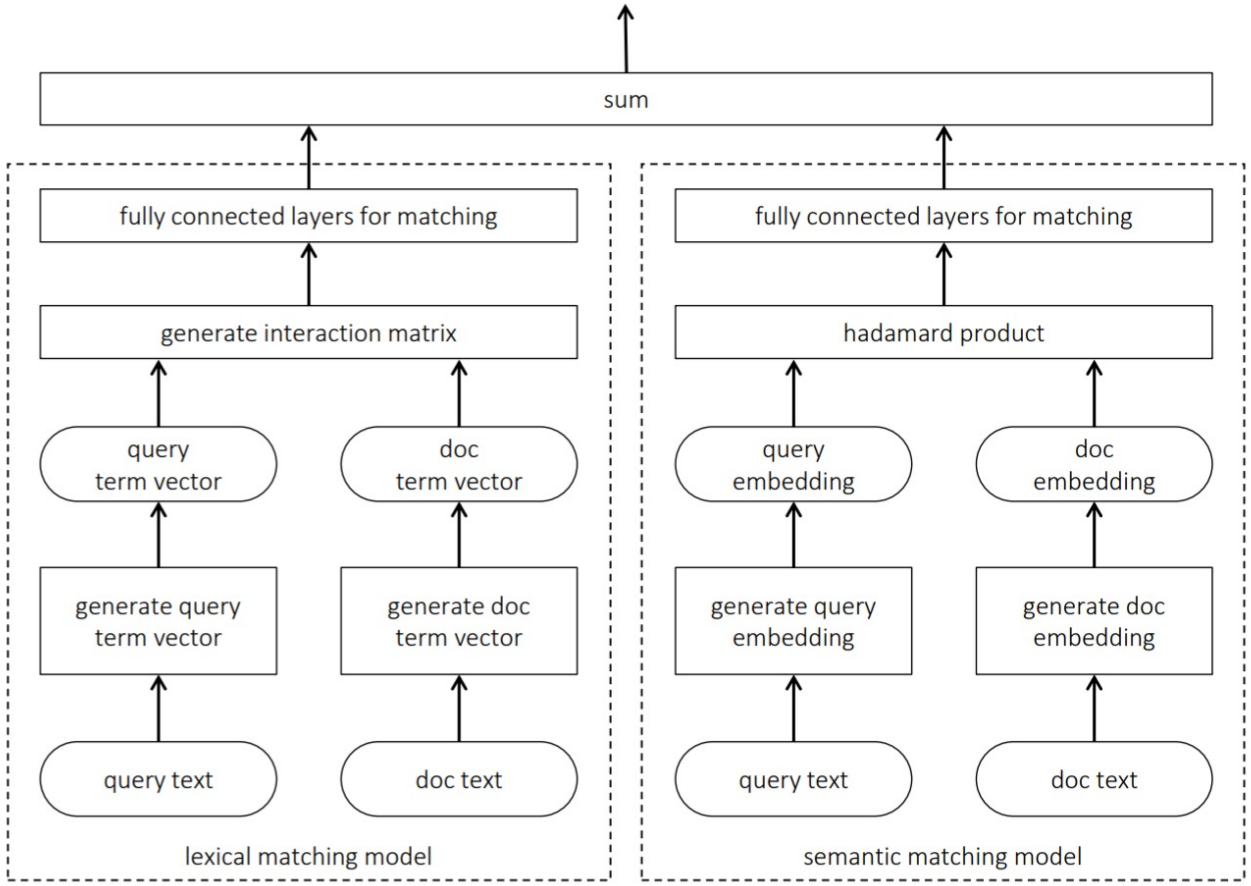
…the importance of incorporating exact term matches as well as matches in the latent space for estimating relevance?



Lexical and semantic matching networks

Mitra et al. [2017] argue that both lexical and semantic matching is important for document ranking

Duet model is a linear combination of two DNNs—focusing on lexical and semantic matching, respectively—jointly trained on labelled data



Bhaskar Mitra, Fernando Diaz, and Nick Craswell. Learning to match using local and distributed representations of text for web search. In WWW, 2017.

Conclusions

- Introduction to Neural Networks
- Term embeddings for IR
- Learning to Rank
- Deep learning
- Deep Structured Semantic Models