Large Language Model and Its Applications

LLM02: Introduction to Deep Learning

Tran-Binh Dang

sponsored by **KEPCO KDN Co., Ltd.**
Eco-friendly & Digital Centered Energy ICT Platform Leader

Hanoi, 09/2023

Outline

1. Text classification
2. Deep learning
3. Neural network
4. Convolution neural network
5. Recurrent neural network
6. Attention mechanism

Text classification: Is this spam?

Subject: **Important notice!**
From: Stanford University <newsforum@stanford.edu>
Date: October 28, 2011 12:34:16 PM PDT
To: undisclosed-recipients:;

Greats News!

You can now access the latest news by using the link below to login to Stanford University News Forum.

http://www.123contactform.com/contact-form-StanfordNew1-236335.html

Click on the above link to login for more information about this new exciting forum. You can also copy the above link to your browser bar and login for more information
about the new services.

Text classification: Positive or negative movie review?

1. Unbelievably disappointing
2. Full of zany characters and richly applied satire, and some great plot twists
3. This is the greatest screwball comedy ever filmed
4. It was pathetic. The worst part about it was the boxing scenes.

1. **Negative**
2. **Positive**
3. **Positive**
4. **Negative**

Text classification: Topic

Text classification: definition

Input:

- a document **d**
- a fixed set of classes $C = \{c_1, c_2, \ldots, c_J\}$

Output:

- a predicted class $c \in C$

Text classification: definition

With supervised learning method.

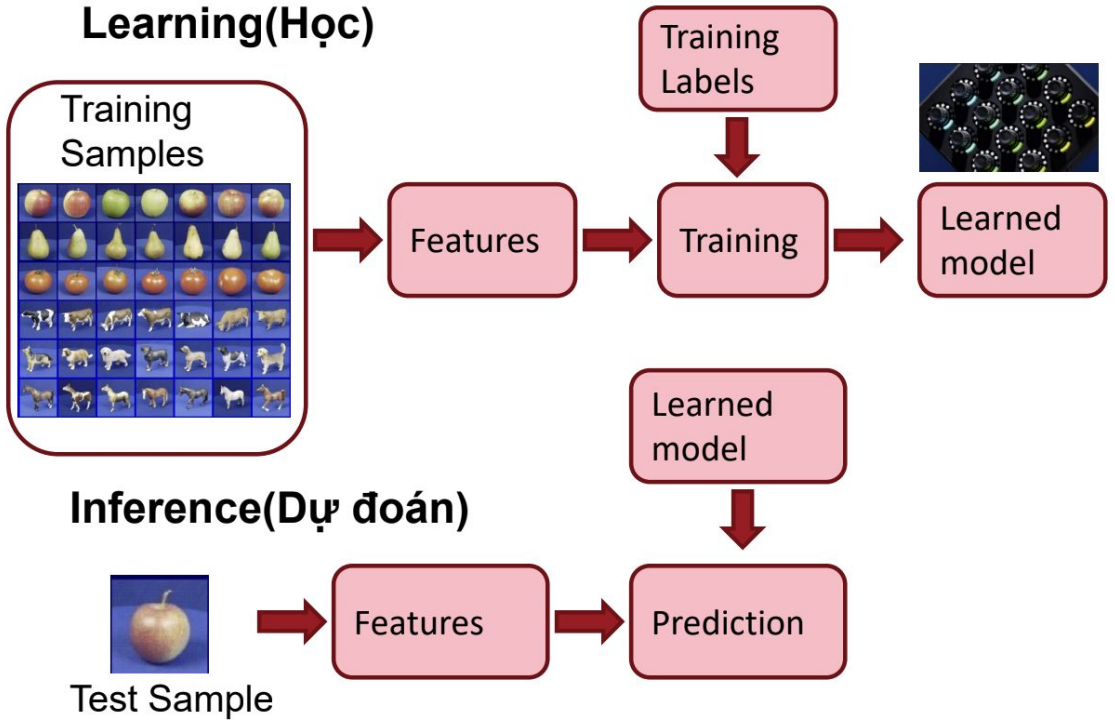
Input:

- a document **d**
- a fixed set of classes **$C = \{c_1, c_2, \ldots, c_J\}$**
- A training set of m hand-labeled documents **$(d_1,c_1),\ldots,(d_m,c_m)$**

Output:

- a learned classifier **$\gamma: d \rightarrow c$**

Text classification: Supervised Learning Architecture



Text classification: Supervised Learning Architecture

Bayes' Rule Applied to Documents and Classes: For a document d and a class c

$$P(c \mid d) = \frac{P(d \mid c)P(c)}{P(d)}$$

$$c_{MAP} = \underset{c \in C}{\operatorname{argmax}}\, P(c \mid d)$$

MAP is "maximum a posteriori" = most likely class

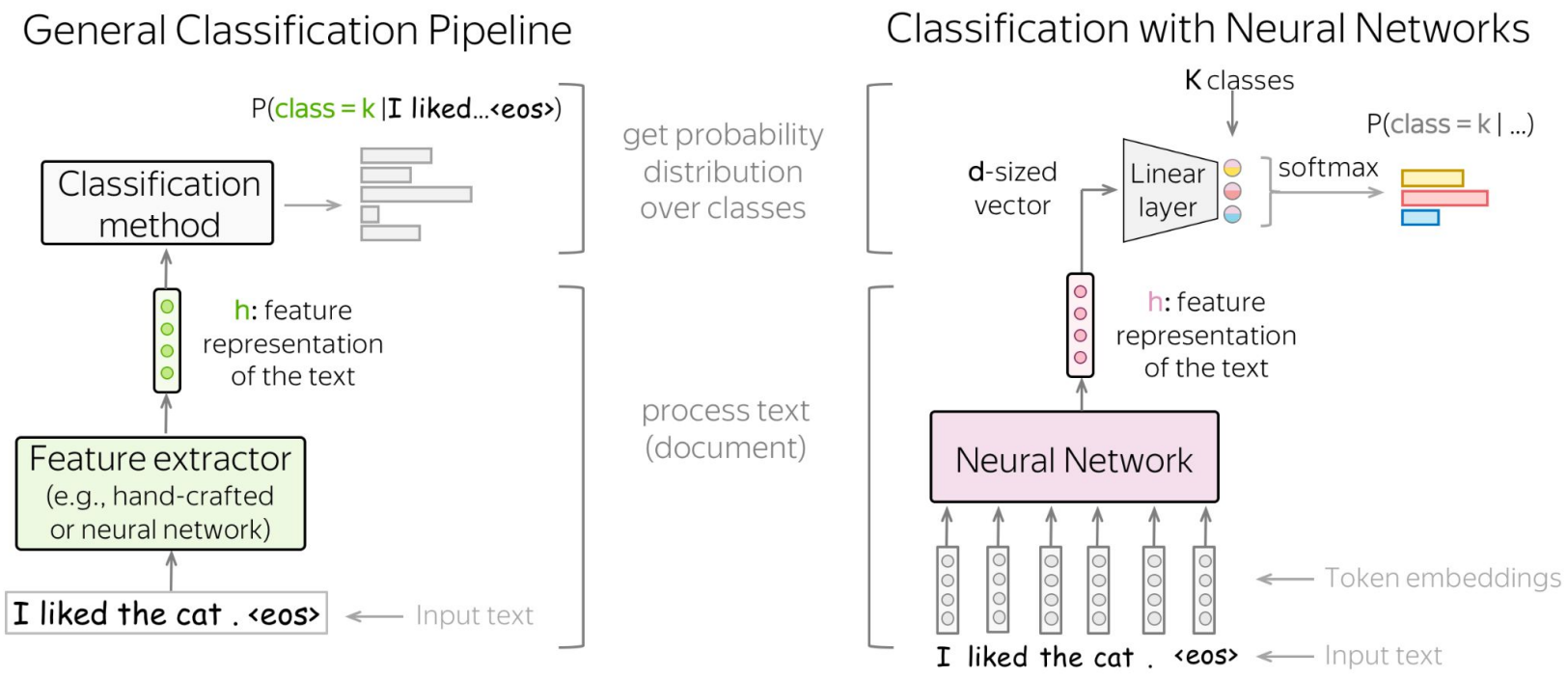
$$= \underset{c \in C}{\operatorname{argmax}}\, \frac{P(d \mid c)P(c)}{P(d)}$$

Bayes Rule

$$= \underset{c \in C}{\operatorname{argmax}}\, P(d \mid c)P(c)$$

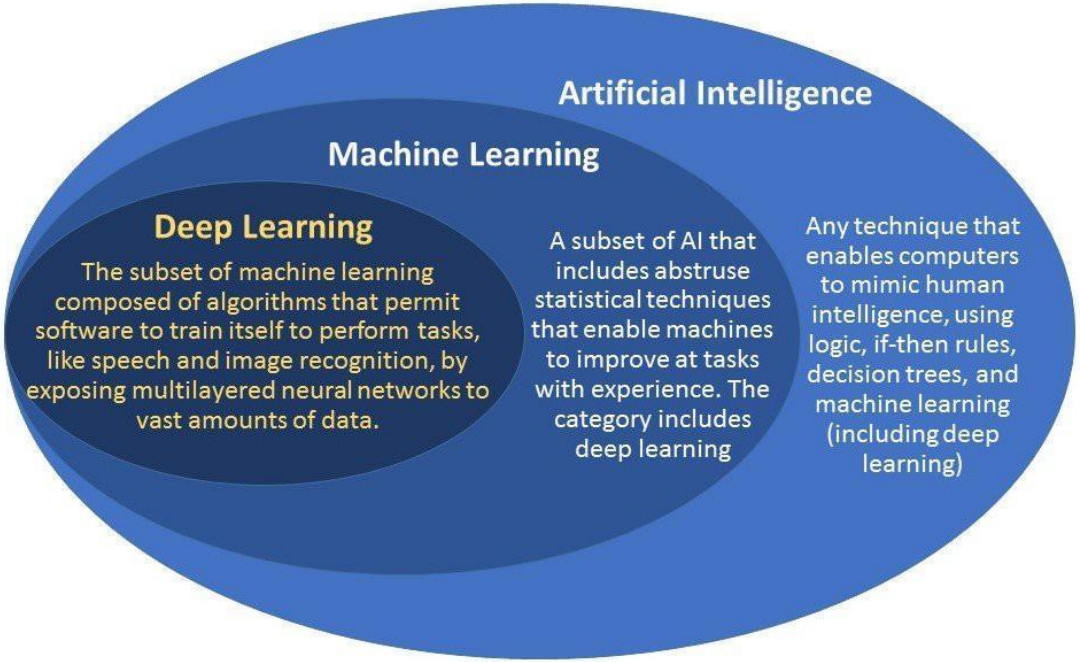
Dropping the denominator

Text Classification with Neural Networks



Deep learning

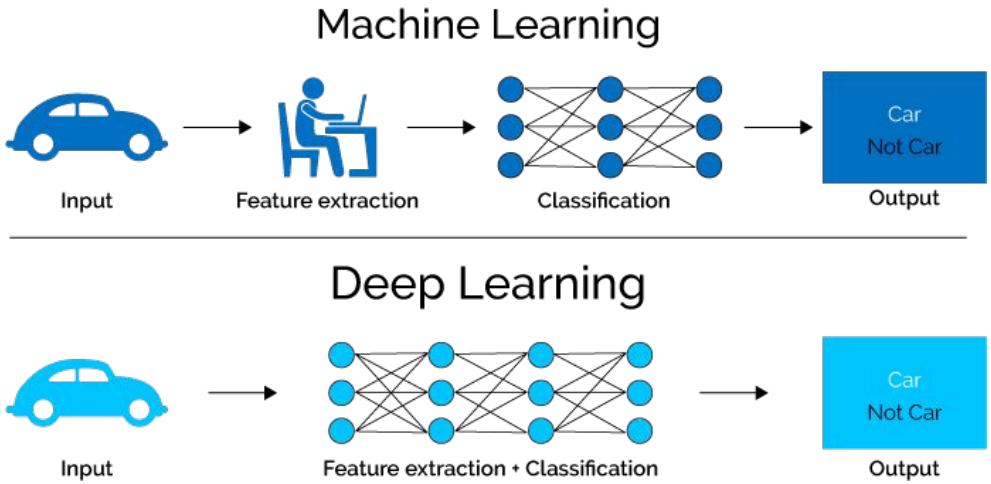
Machine Learning vs. Deep Learning



What is Deep Learning?

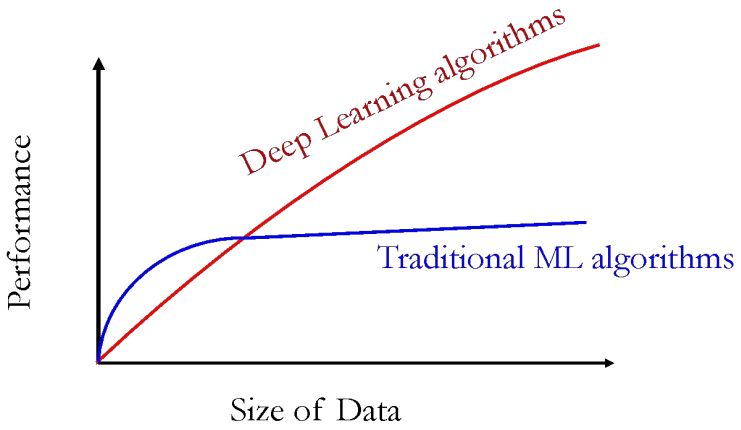
A machine learning subfield of learning **representations** of data. Exceptional effective at **learning patterns**.

Deep learning algorithms attempt to learn (multiple levels of) representation by using a **hierarchy of multiple layers**

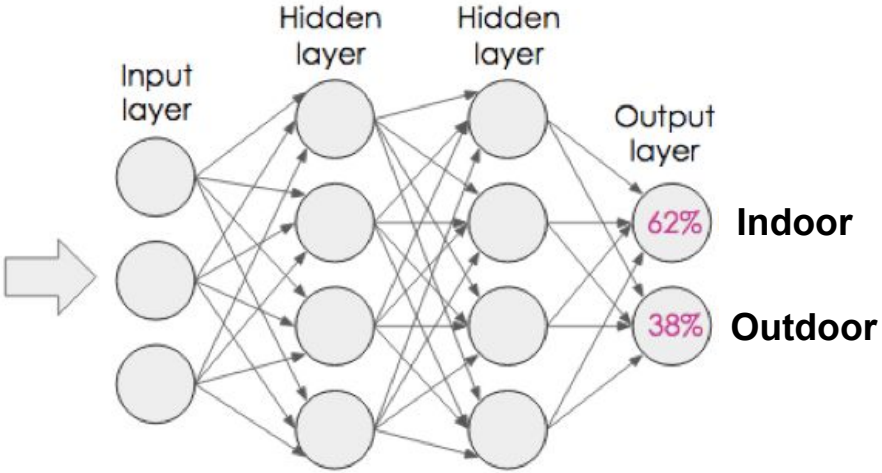


Why is DL useful?

- Manually designed features are often **over-specified**, **incomplete** and take a **long time to design** and validate
- Learned features are **easy to adapt**, **fast** to learn
- Deep learning provides a very **flexible**, (almost?) **universal**, learnable framework for representing world, visual and linguistic information.
- Can learn both unsupervised and supervised
- Effective **end-to-end** joint system learning
- Utilize large amounts of training data



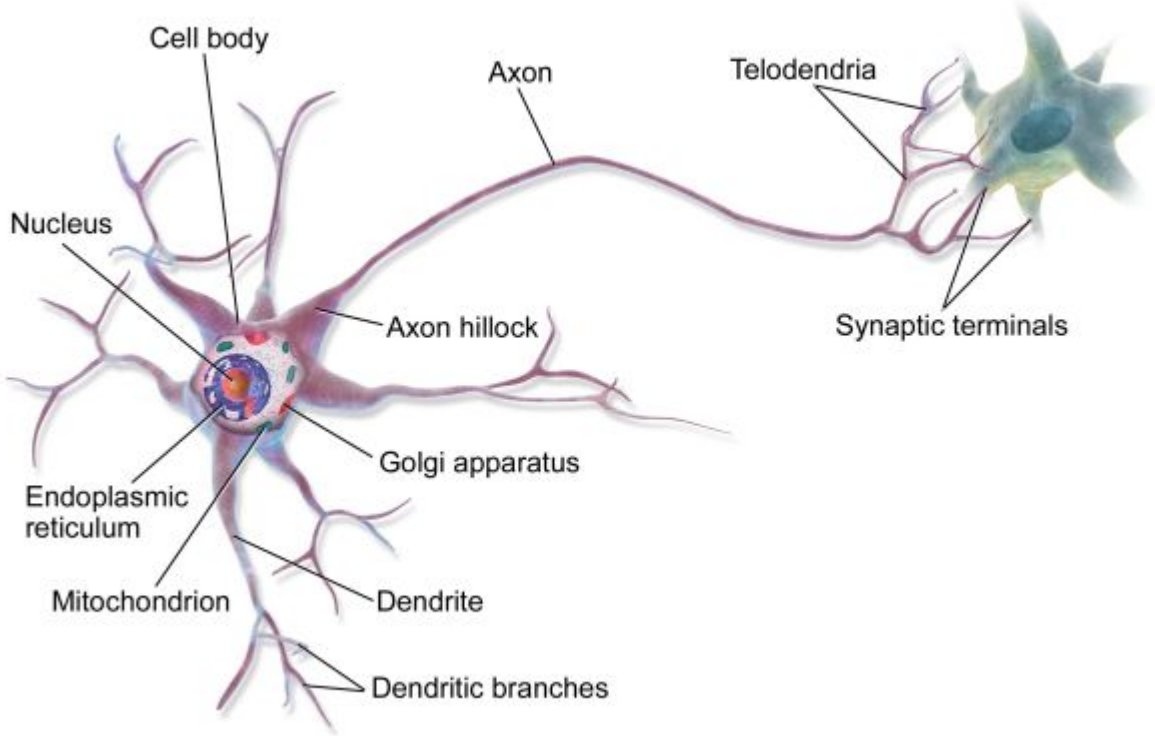
How does DL learn features?



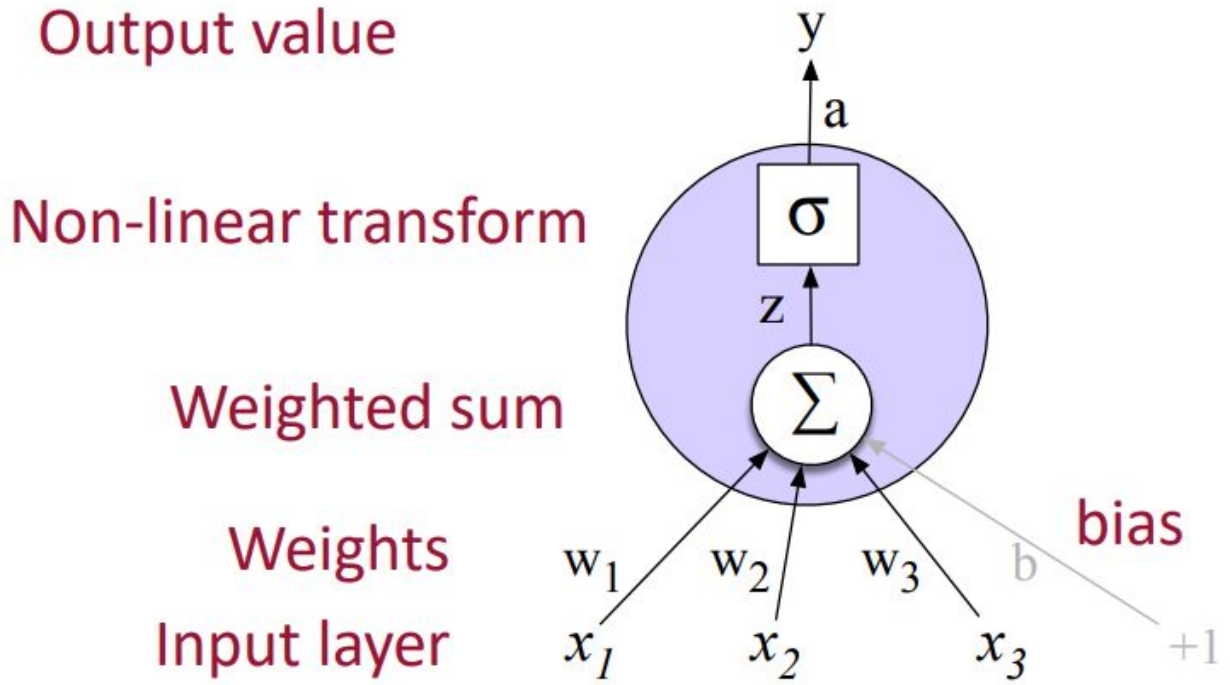
Answer: Indoor

Neural network

This is in your brain



Neural Network Unit



Neural Network Unit

- Take weighted sum of inputs, plus a bias

$$z = b + \sum_i w_i x_i$$

$$z = w \cdot x + b$$

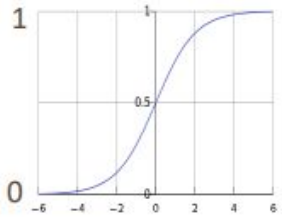
- Instead of just using z, we'll apply a nonlinear activation function f:

$$y = a = f(z)$$

Non-Linear Activation Functions

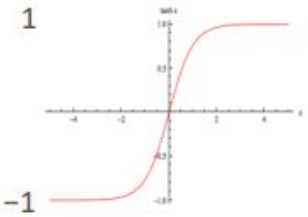
logistic ("sigmoid")

$$f(z) = \frac{1}{1 + \exp(-z)}$$



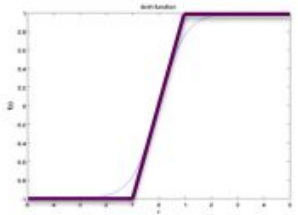
tanh

$$\tanh(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$$



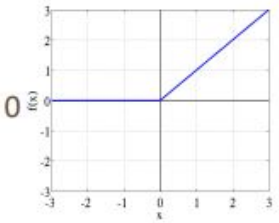
hard tanh

$$\text{HardTanh}(x) = \begin{cases} -1 & \text{if } x < -1 \\ x & \text{if } -1 <= x <= 1 \\ 1 & \text{if } x > 1 \end{cases}$$

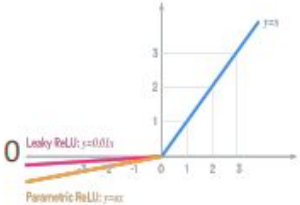


(Rectified Linear Unit)
ReLU

$$\text{ReLU}(z) = \max(z, 0)$$



Leaky ReLU / Parametric ReLU



tanh is just a rescaled and shifted sigmoid (2 × as steep, [−1,1]):

$$\tanh(z) = 2\text{logistic}(2z) - 1$$

Logistic and tanh are still used (e.g., logistic to get a probability)

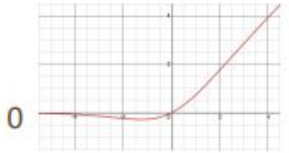
However, now, for deep networks, the first thing to try is ReLU: it trains quickly and performs well due to good gradient backflow.

ReLU has a negative "dead zone" that recent proposals mitigate

GELU is frequently used with Transformers (BERT, RoBERTa, etc.)

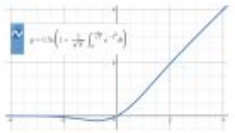
Swish arXiv:1710.05941

$$\text{swish}(x) = x \cdot \text{logistic}(x)$$



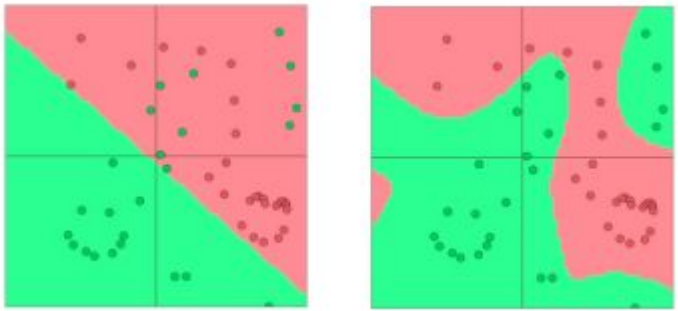
GELU arXiv:1606.08415

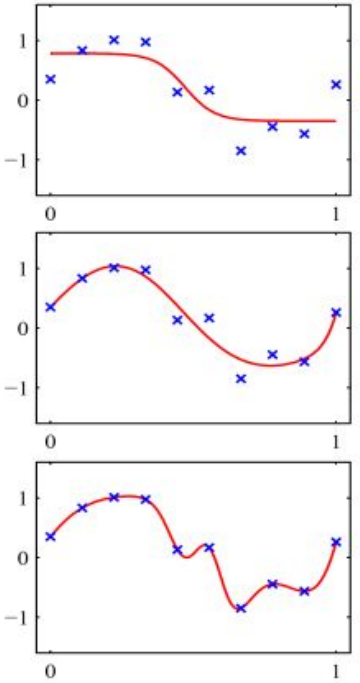
$$\text{GELU}(x) = x \cdot P(X \le x), X \sim N(0,1) \approx x \cdot \text{logistic}(1.702x)$$



Why they're needed

- Neural networks do function approximation, e.g., regression or classification
  - Without non-linearities, deep neural networks can't do anything more than a linear transform
  - Extra layers could just be compiled down into a single linear transform
  - But, with more layers that include non-linearities, they can approximate any complex function!





Example

Suppose a unit has:

w = [0.2, 0.3, 0.9]

b = 0.5

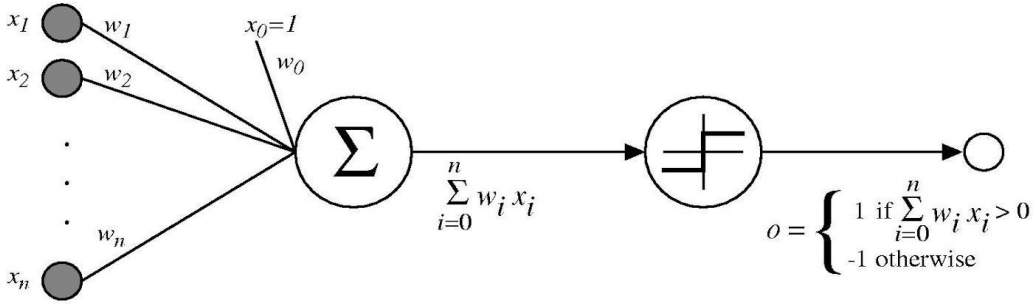
What happens with input x:

x = [0.3, 0.4, 0.2]

y = σ (w.x+b)

Note: using **sigmoid** and **tanh**

Perceptron



$$o(x_1, \ldots, x_n) = \begin{cases} 1 & \text{if } w_0 + w_1 x_1 + \cdots + w_n x_n > 0 \\ -1 & \text{otherwise.} \end{cases}$$

Sometimes we'll use simpler vector notation:

$$o(\vec{x}) = \begin{cases} 1 & \text{if } \vec{w} \cdot \vec{x} > 0 \\ -1 & \text{otherwise.} \end{cases}$$

Perceptron Training Rule

$$w_i \leftarrow w_i + \Delta w_i$$

where

$$\Delta w_i = \eta(t - o)x_i$$

Where:

- $t = c(\vec{x})$ is target value
- o is perceptron output
- η is small constant (e.g., 0.1) called *learning rate*

Can prove it will converge if

- Training data is linearly separable
- η sufficiently small

Gradient Descent

To understand, consider simpler *linear unit*, where

$$o = w_0 + w_1 x_1 + \cdots + w_n x_n$$

Let's learn w_i's that minimize the squared error

$$E[\vec{w}] \equiv \frac{1}{2} \sum_{d \in D} (t_d - o_d)^2$$

Where D is set of training examples

Gradient Descent

Gradient:

$$\nabla E[\vec{w}] \equiv \left[\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \cdots \frac{\partial E}{\partial w_n} \right]$$

Training rule:

$$\Delta \vec{w} = -\eta \nabla E[\vec{w}]$$

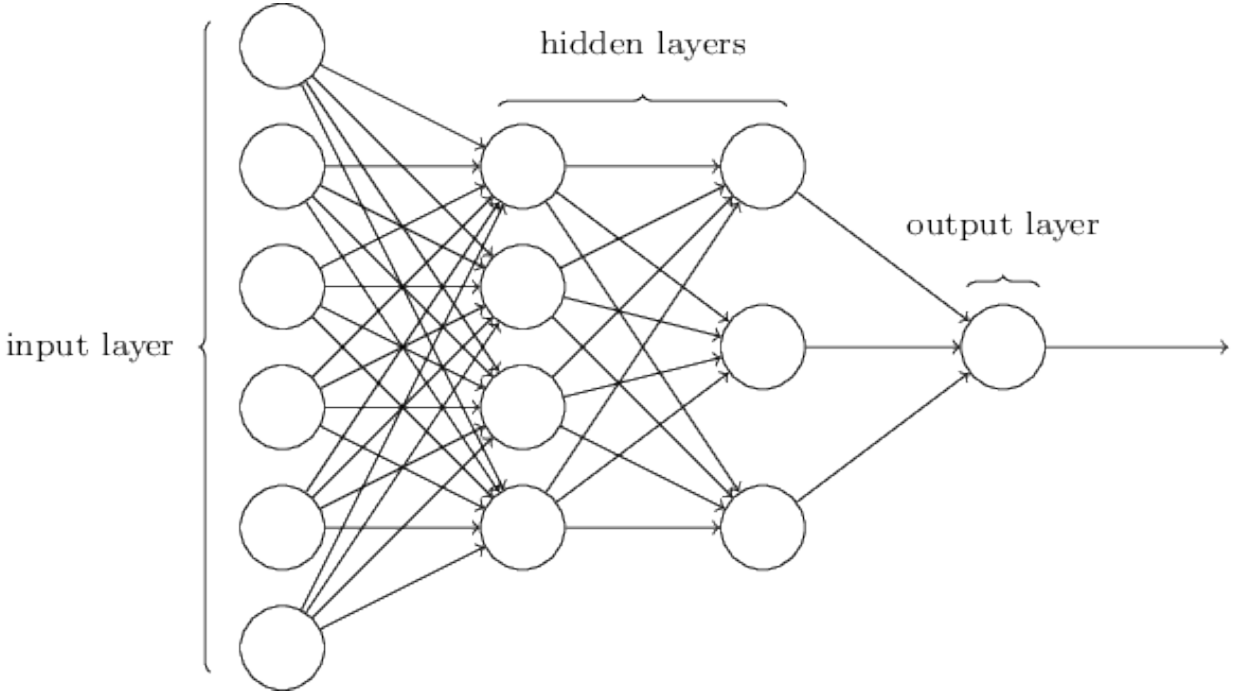
I.e.:

$$\Delta w_i = -\eta \frac{\partial E}{\partial w_i}$$

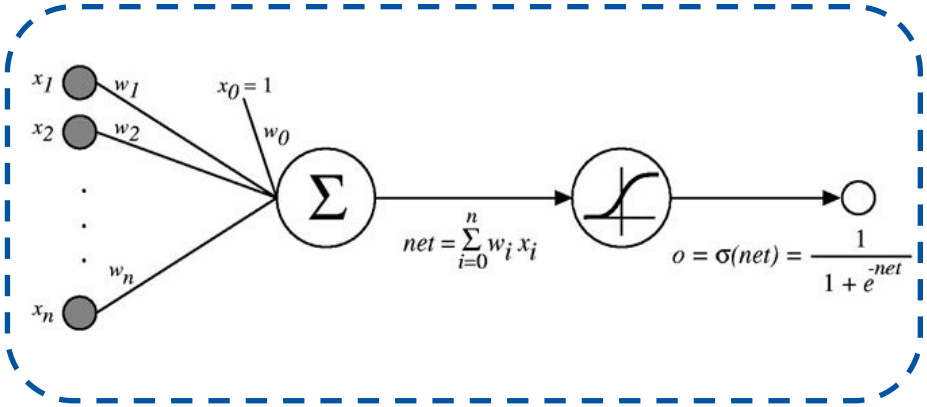
Gradient Descent

$$
\begin{aligned}
\frac{\partial E}{\partial w_i} &= \frac{\partial}{\partial w_i}\frac{1}{2}\sum_d (t_d - o_d)^2 \\
&= \frac{1}{2}\sum_d \frac{\partial}{\partial w_i}(t_d - o_d)^2 \\
&= \frac{1}{2}\sum_d 2(t_d - o_d)\frac{\partial}{\partial w_i}(t_d - o_d) \\
&= \sum_d (t_d - o_d)\frac{\partial}{\partial w_i}(t_d - \vec{w}\cdot\vec{x_d}) \\
\frac{\partial E}{\partial w_i} &= \sum_d (t_d - o_d)(-x_{i,d})
\end{aligned}
$$

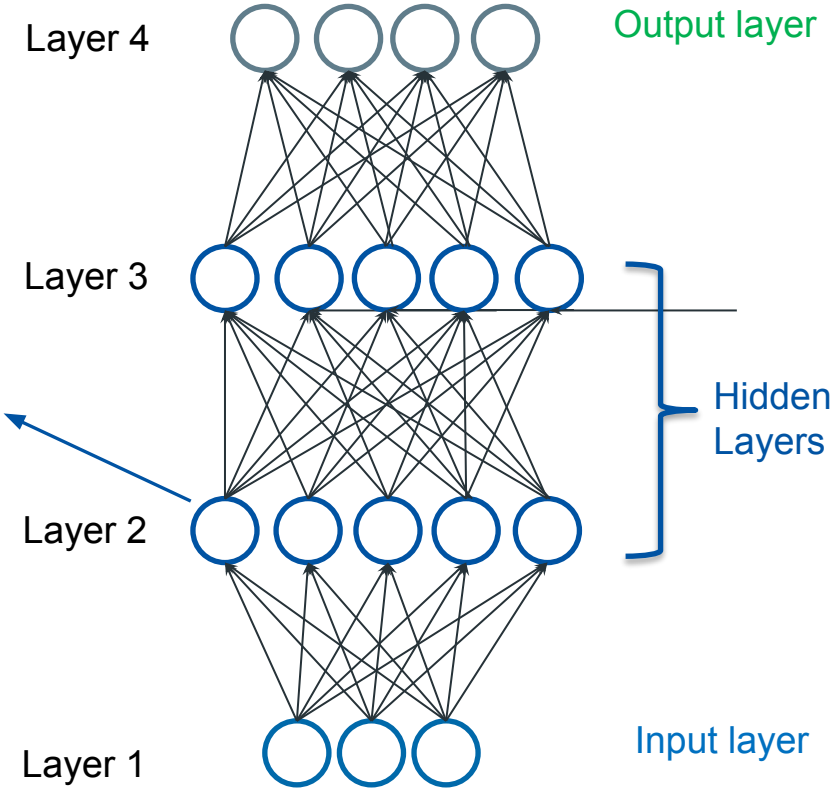
Multi Layer Perceptron



Multi-layer Neural Networks with Sigmoid



Each node (“neuron”) is the sigmoid unit



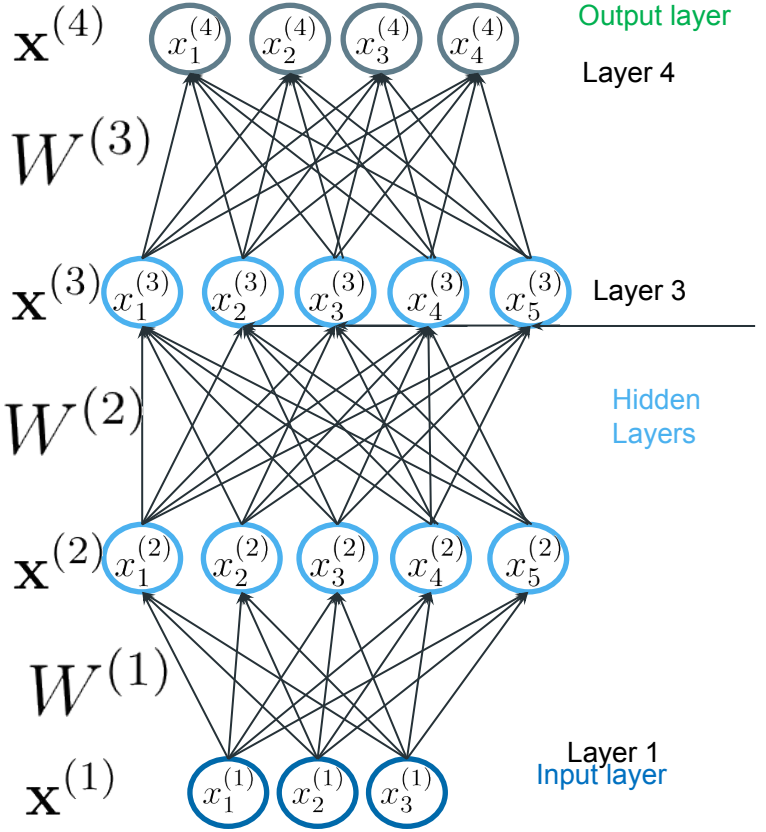
Forward Propagation

- Notations
  - Input vector at level l $\mathbf{x}^{(l)}$
  - Weight matrix of level l $W^{(l)}$
- Forward-propagation

$$\underset{5\times 1}{\mathbf{x}^{(2)}} = \sigma\left(\underset{5\times 3}{W^{(1)}}\underset{3\times 1}{\mathbf{x}^{(1)}}\right)$$

$$\underset{5\times 1}{\mathbf{x}^{(3)}} = \sigma\left(\underset{5\times 5}{W^{(2)}}\underset{5\times 1}{\mathbf{x}^{(2)}}\right)$$

$$\underset{4\times 1}{\mathbf{x}^{(4)}} = \sigma\left(\underset{4\times 5}{W^{(3)}}\underset{5\times 1}{\mathbf{x}^{(3)}}\right)$$



Back Propagation

- Error at level *l* $\boldsymbol{\delta}^{(l)}$
- Error at last level (L=4)

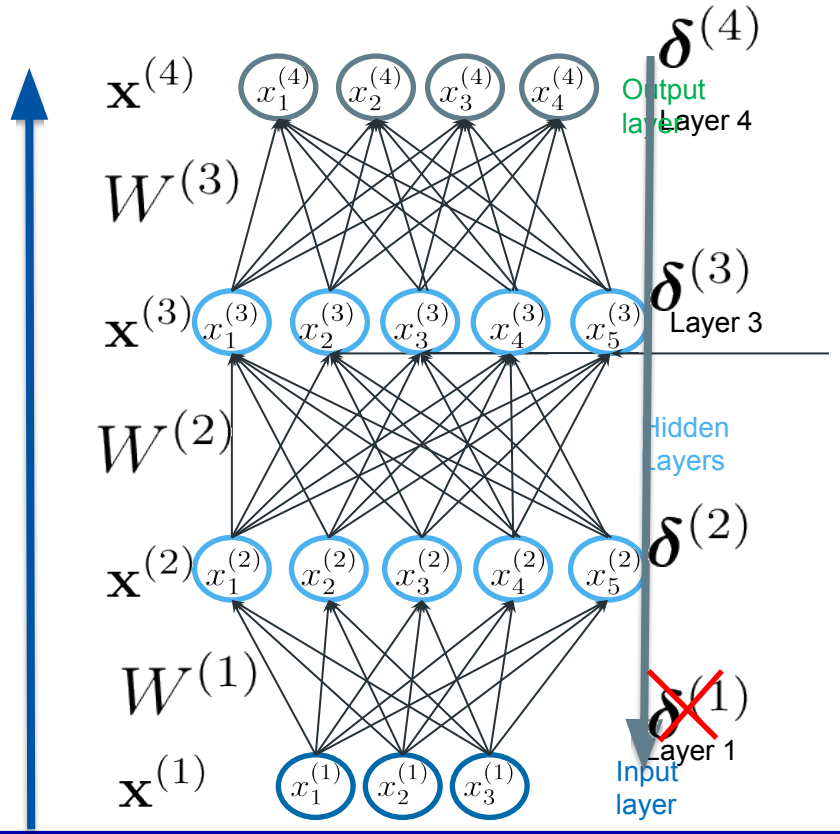
$$\boldsymbol{\delta}^{(4)} = (\mathbf{y} - \mathbf{x}^{(4)}) .* \mathbf{x}^{(4)} .* (1 - \mathbf{x}^{(4)})$$

$$\boldsymbol{\delta}^{(3)} = (W^{(3)})^T \boldsymbol{\delta}^{(4)} .* \mathbf{x}^{(3)} .* (1 - \mathbf{x}^{(3)})$$

$$\boldsymbol{\delta}^{(2)} = (W^{(2)})^T \boldsymbol{\delta}^{(3)} .* \mathbf{x}^{(2)} .* (1 - \mathbf{x}^{(2)})$$

$$\Delta W^{(l)} = \eta \boldsymbol{\delta}^{(l+1)} (\mathbf{x}^{(l)})^T$$

$$\mathbf{W}^{(l)} := \mathbf{W}^{(l)} + \Delta W^{(l)}$$



Convolution neural networks

Convolutional networks

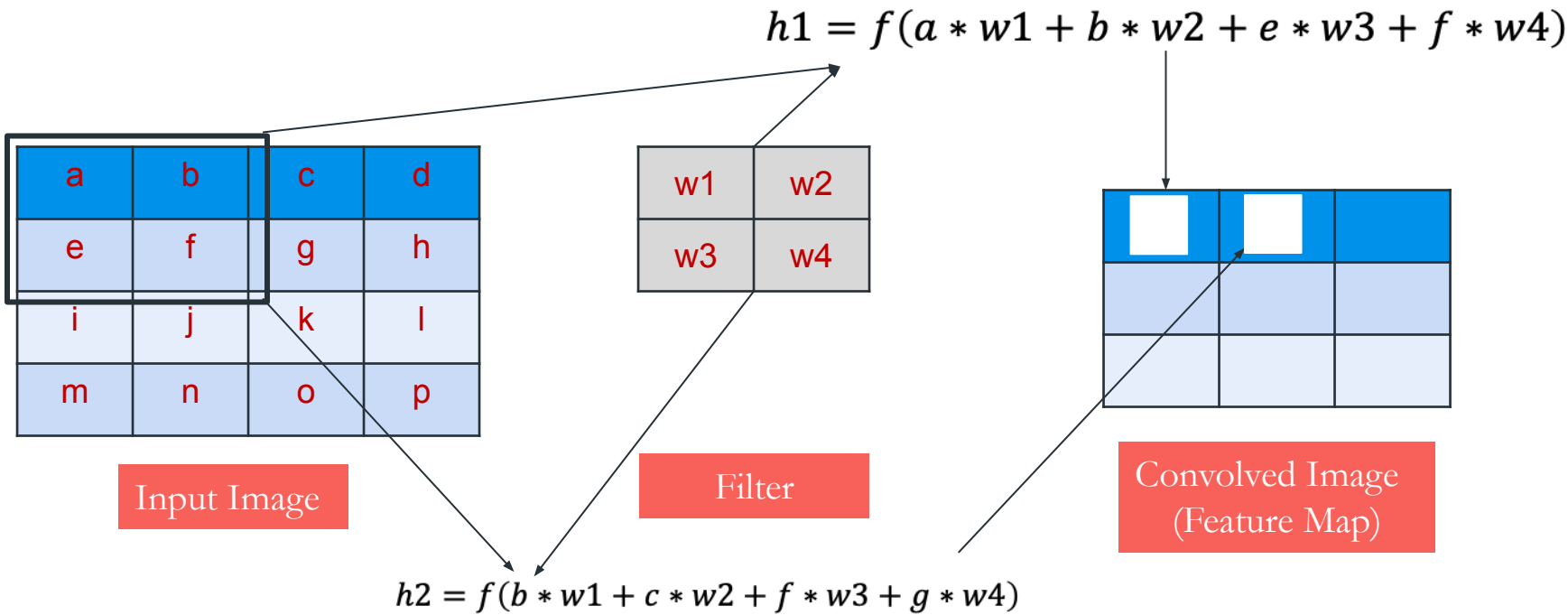
Convolutional networks were originally developed for computer vision tasks. Therefore, let's first understand the intuition behind convolutional models for images.

Input Image

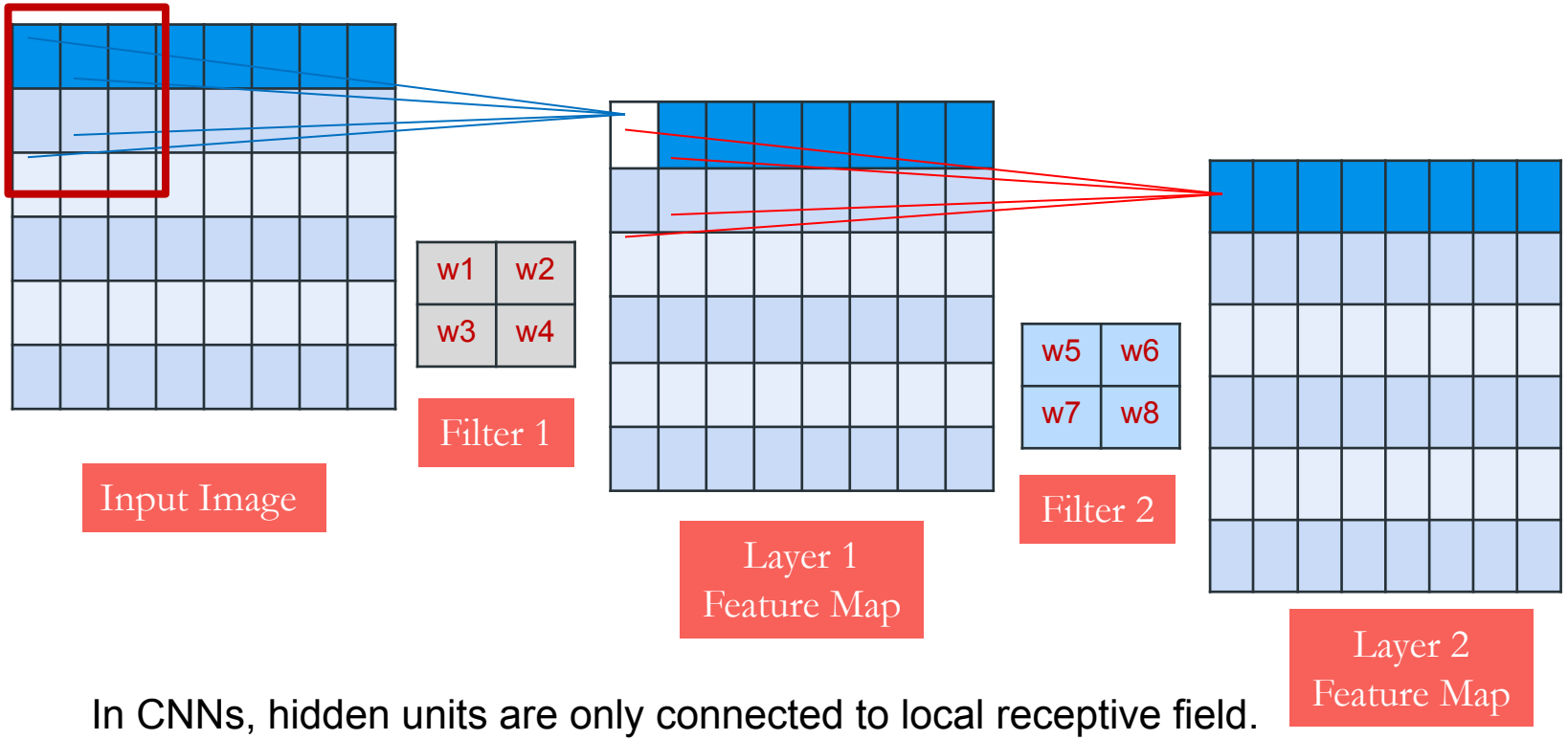
$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

Convoluted Image

Convolution Layers



Lower Level to More Complex Features



In CNNs, hidden units are only connected to local receptive field.

Pooling

- Max pooling: reports the maximum output within a rectangular neighborhood.
- Average pooling: reports the average output of a rectangular neighborhood.

1	3	5	3
4	2	3	1
3	1	1	3
0	1	0	4

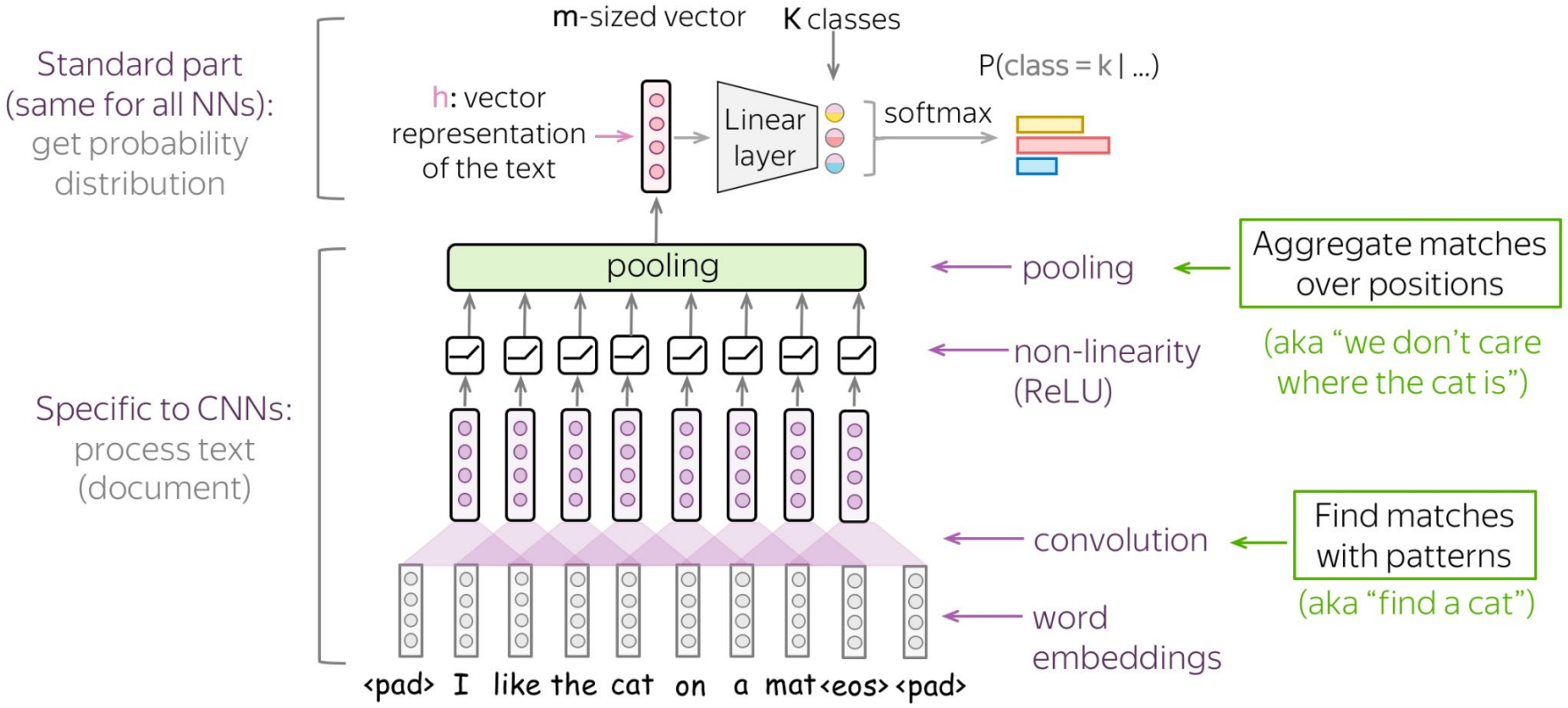
Input Matrix

MaxPool with 2X2 filter with stride of 2

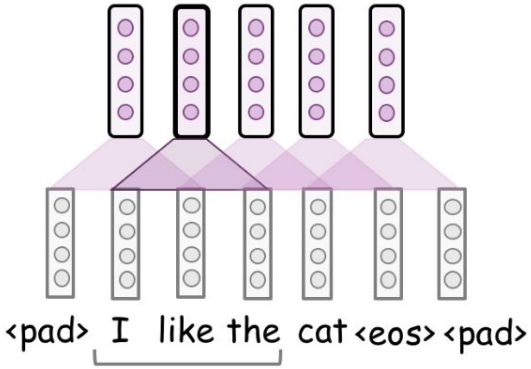
4	5
3	4

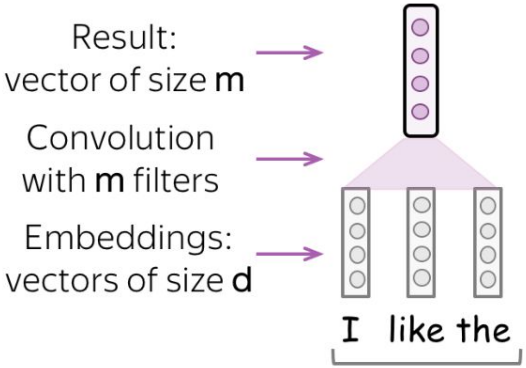
Output Matrix

Convolutional Neural Networks



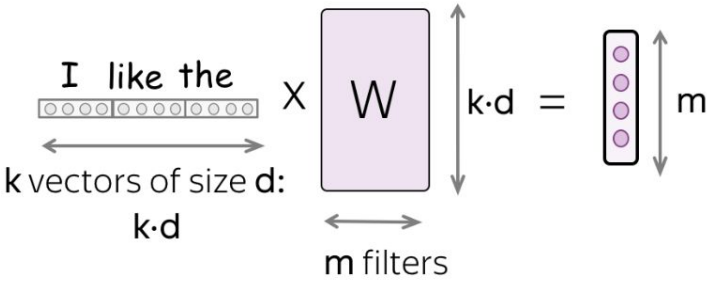
Convolutional Neural Networks



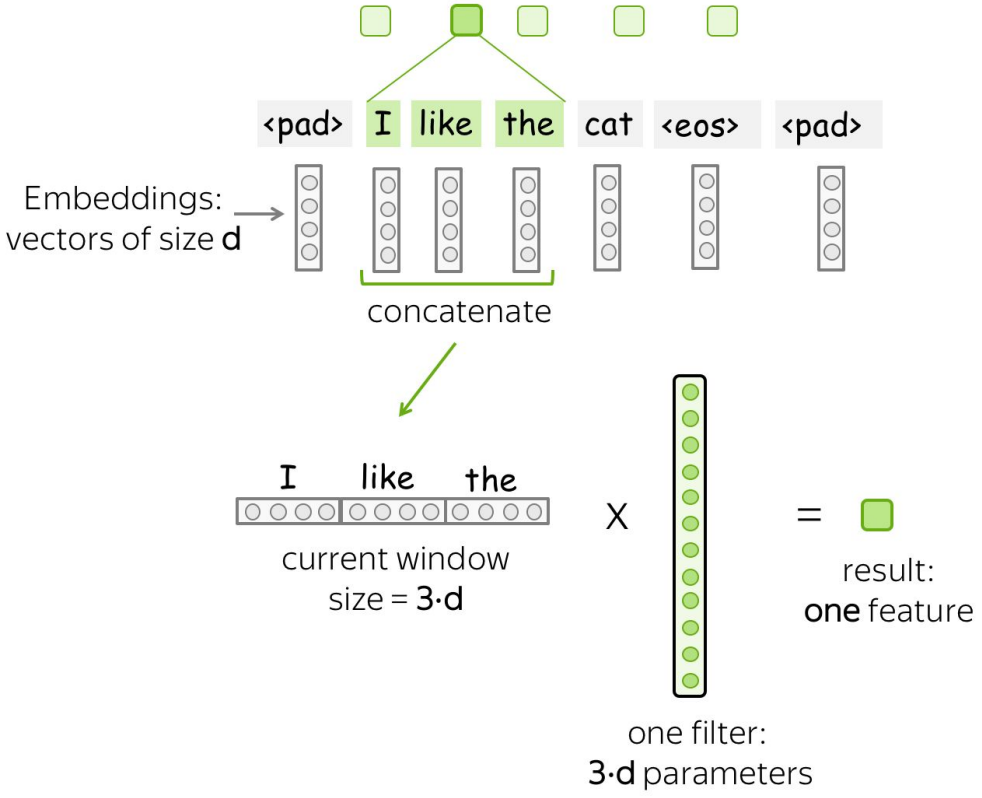


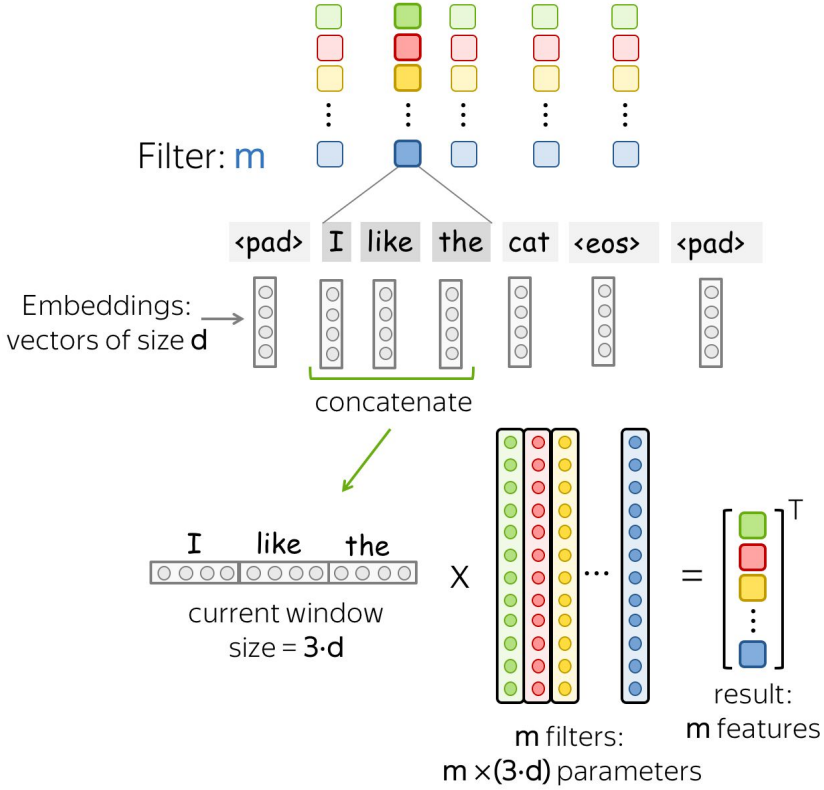
Kernel size k (k=3)
(convolution window size)

Convolution is a linear layer
mapping from k·d to m

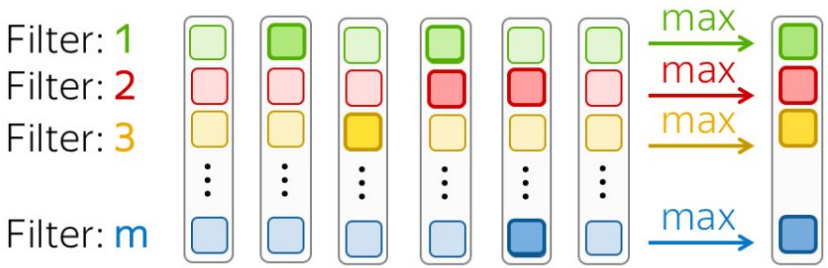


Convolutional Neural Networks



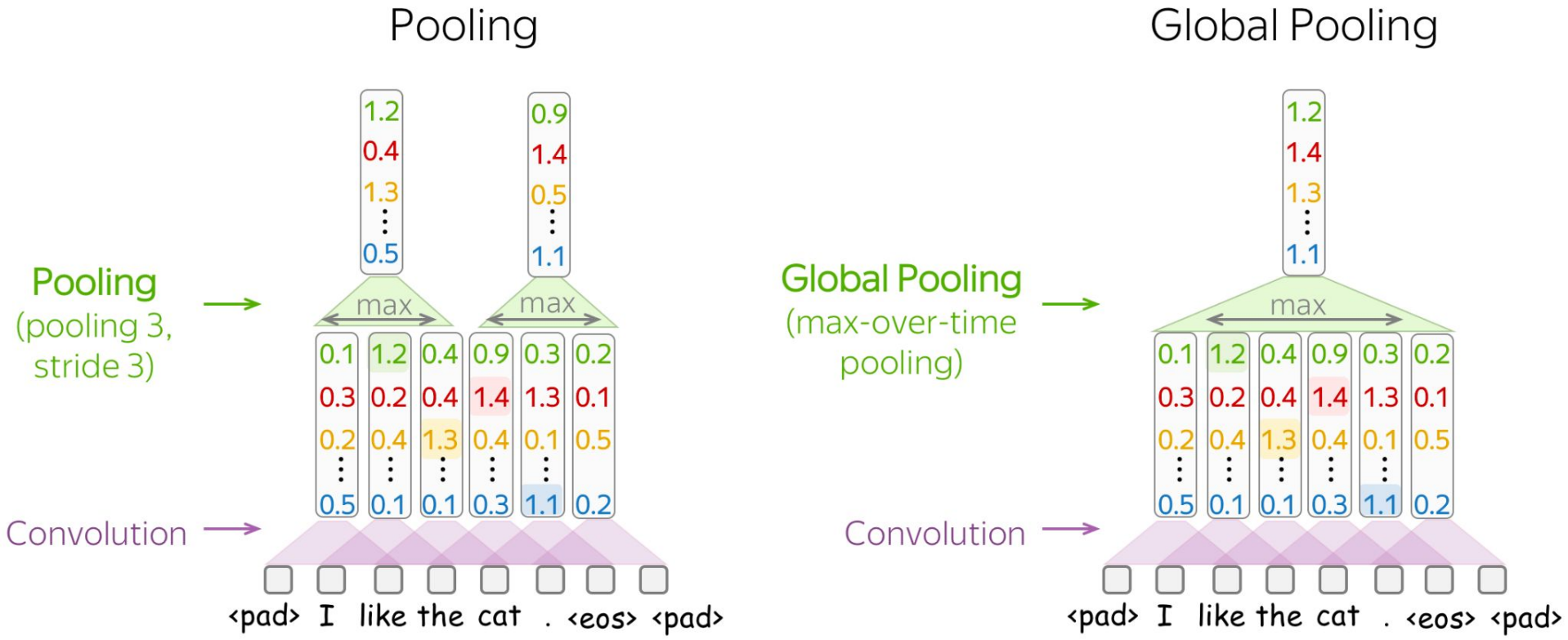


Convolutional Neural Networks

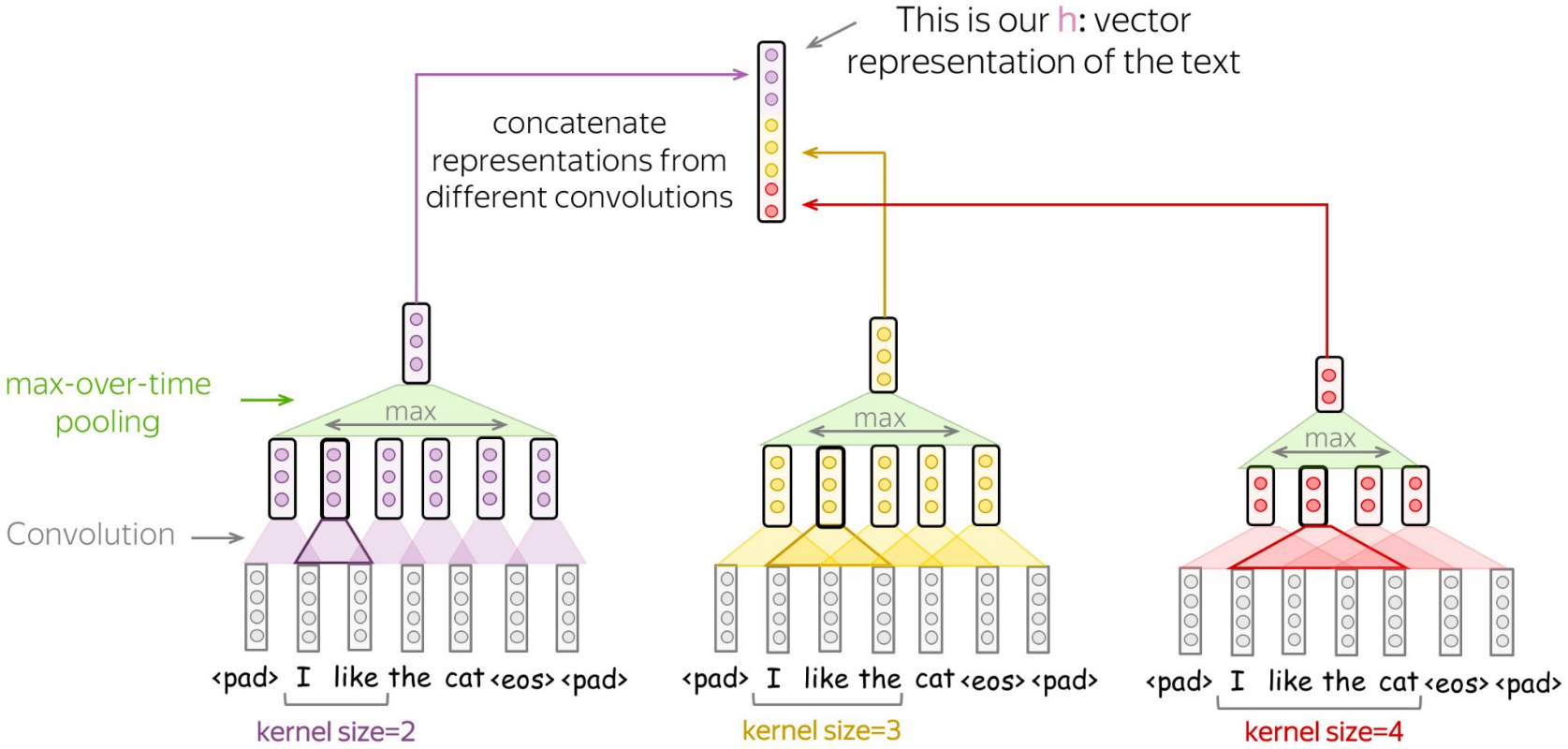


Max pooling: maximum for each dimension (feature)

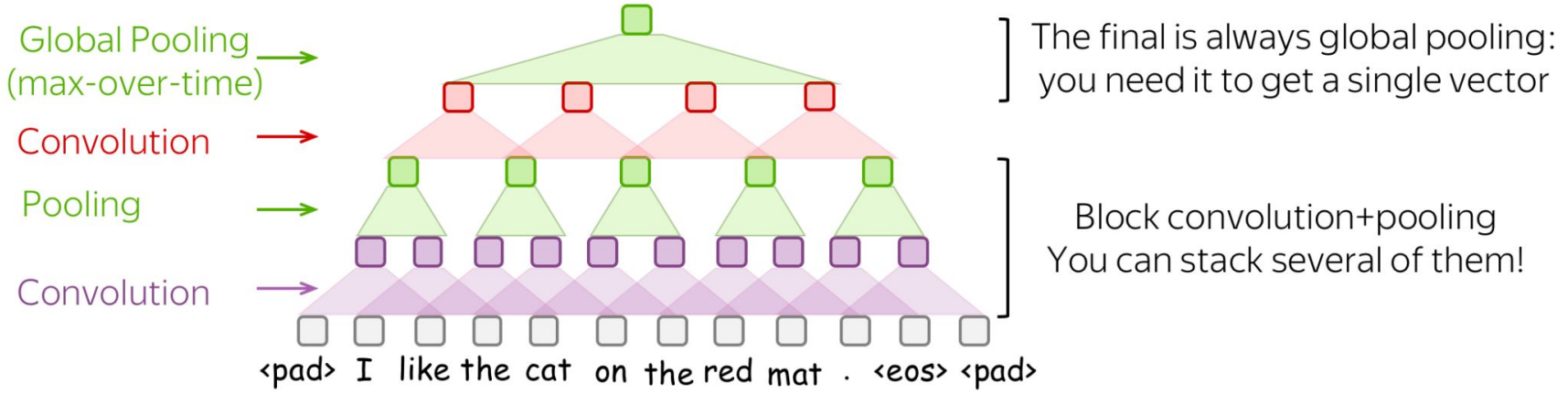
Convolutional Neural Networks



Convolutional Neural Networks



Convolutional Neural Networks



Recurrent neural networks

Why RNN?

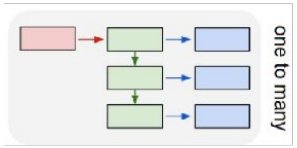
The **limitations** of the **Convolutional Neural Networks**

- Take fixed length vectors as input
- And produce fixed length vectors as output.
- Allow fixed amount of computational steps.

We need to model the data with **temporal or sequential structures** and **varying length of inputs/outputs**

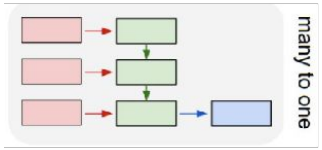
E.g., This movie is very slow in the beginning but picks up pace later on and has some great action sequences and comedy scenes.

Modeling Sequences



A person riding a motorbike on dirt road

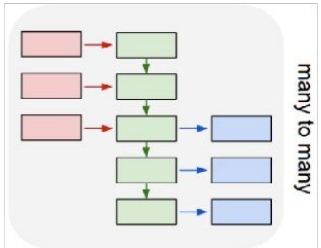
Image Captioning



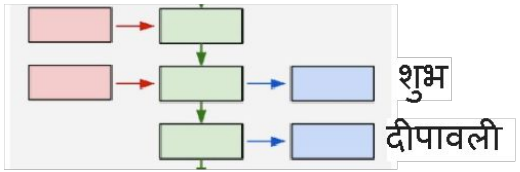
Awesome tutorial.

Positive

Sentiment Analysis



Happy Diwali



Machine Translation

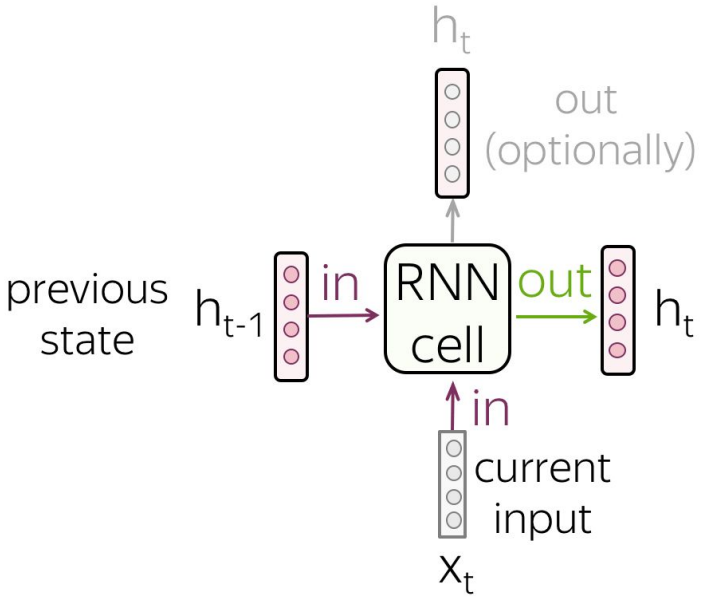
Sequence to Sequence Basics

Machine translation task:

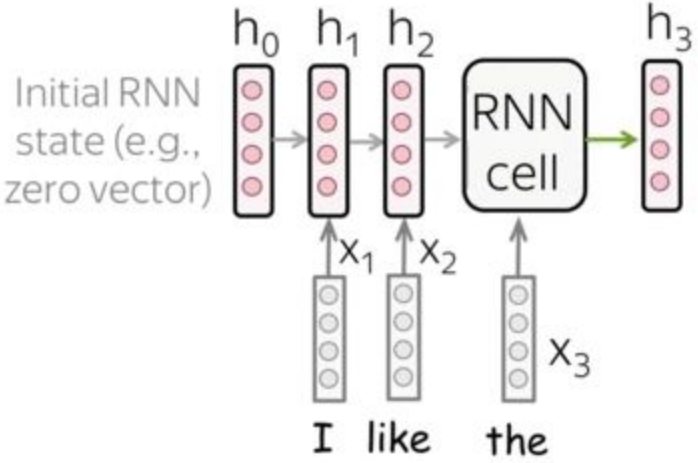
- Input sequence: $\mathbf{x_1, x_2, \ldots, x_m}$
- Output sequence: $\mathbf{y_1, y_2, \ldots, y_n}$
- Translation can be thought of as finding the target sequence that is **the most probable** given the input; formally, the target sequence that **maximizes** the conditional probability:

What is RNN?

- Recurrent neural networks are **connectionist models** with the ability to **selectively pass information across sequence steps**, while processing sequential data one element at a time.
- Allows **a memory of the previous inputs** to **persist in the model's internal state** and **influence the outcome**.



RNN reads a sequence



Get new state from RNN

Text: I like the cat on a mat <eos>

not read yet

we are here

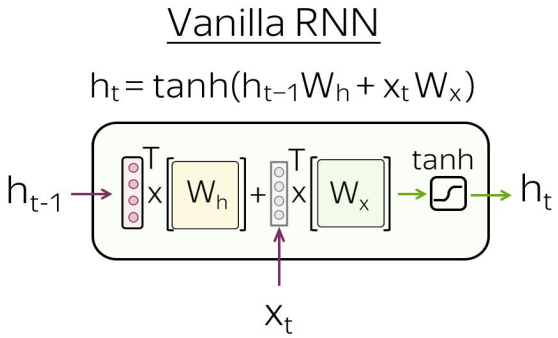
RNN reads a sequence

Vanilla RNN

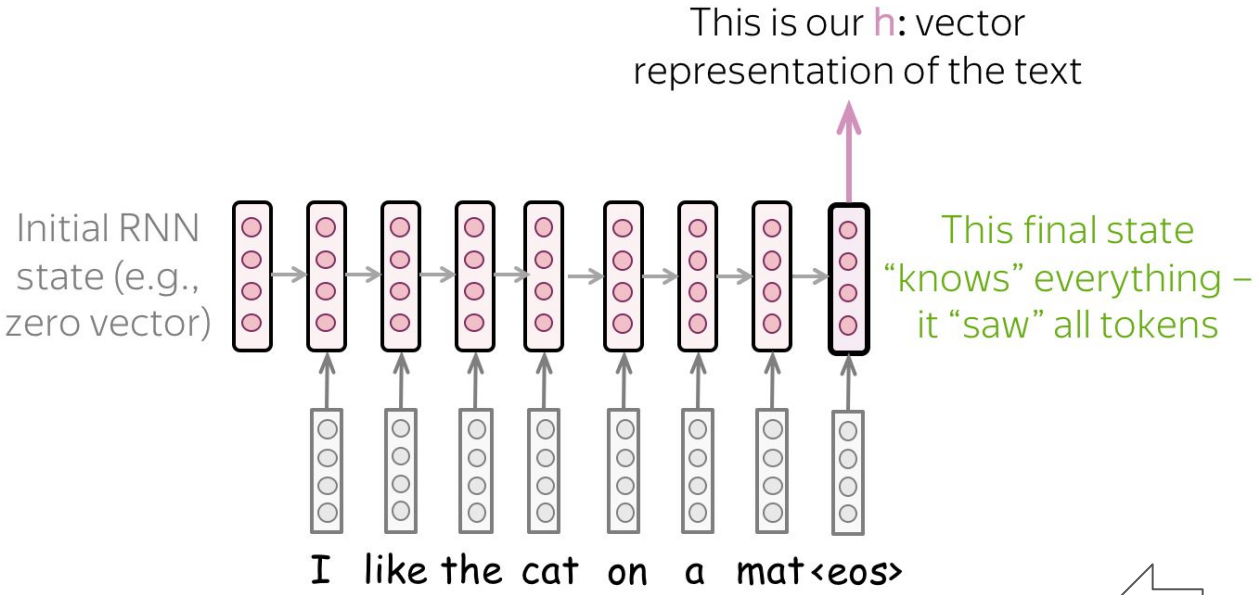
- The simplest recurrent network, Vanilla RNN, transforms $\mathbf{h}_{t-1}$ and $\mathbf{x}_t$ linearly, then applies a non-linearity (most often, the **tanh** function):

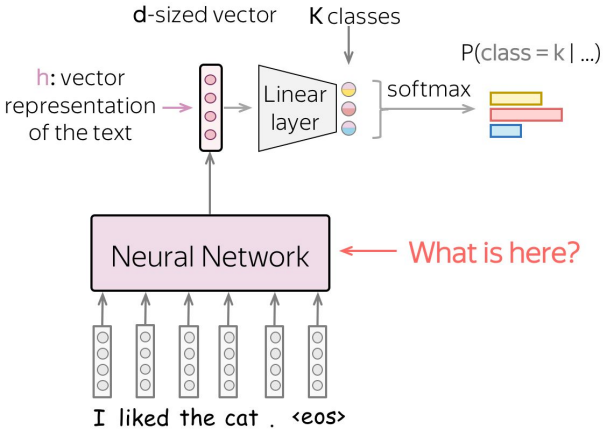
$$h_t = \tanh(h_{t-1}W_h + x_t W_t)$$

- Vanilla RNNs suffer from the vanishing and exploding gradients problem.

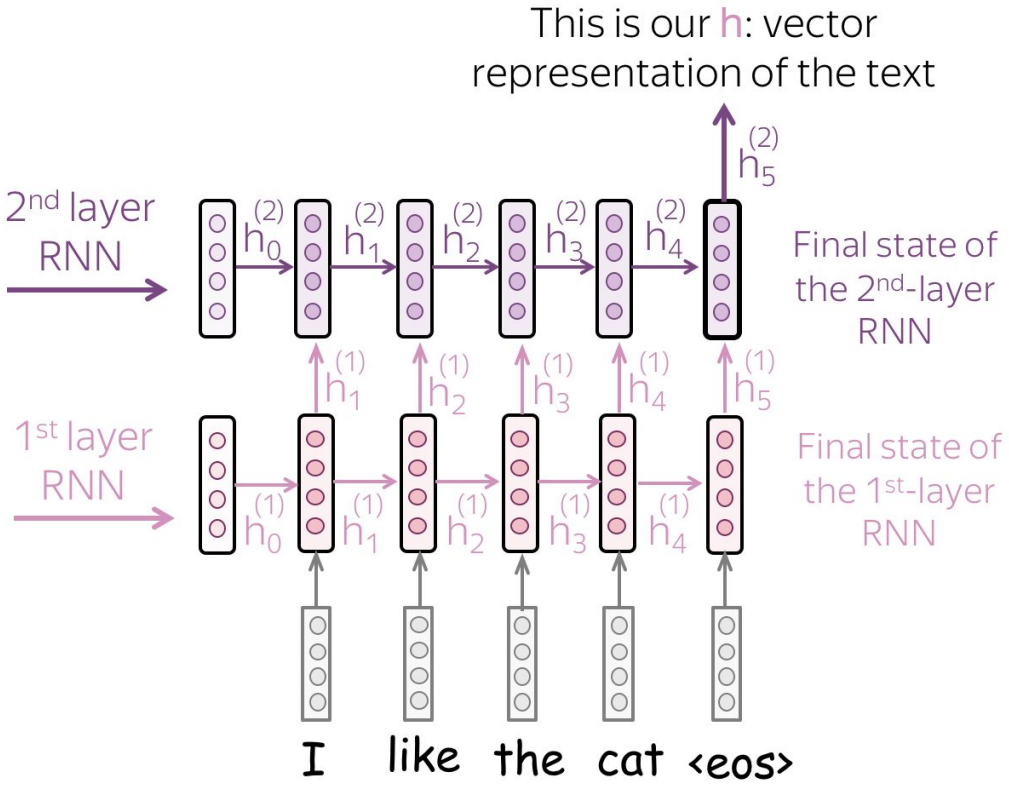


Recurrent Neural Networks for Text Classification

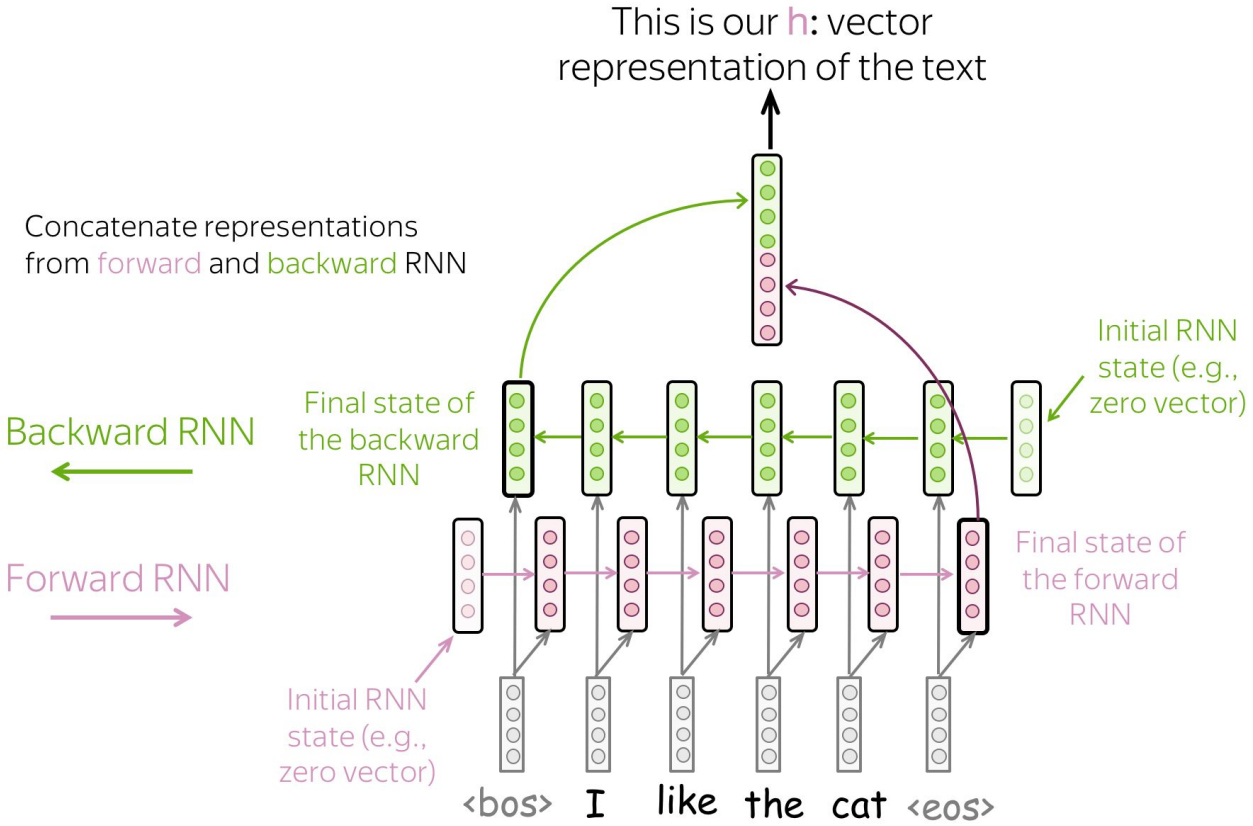




Recurrent Neural Networks for Text Classification



Recurrent Neural Networks for Text Classification

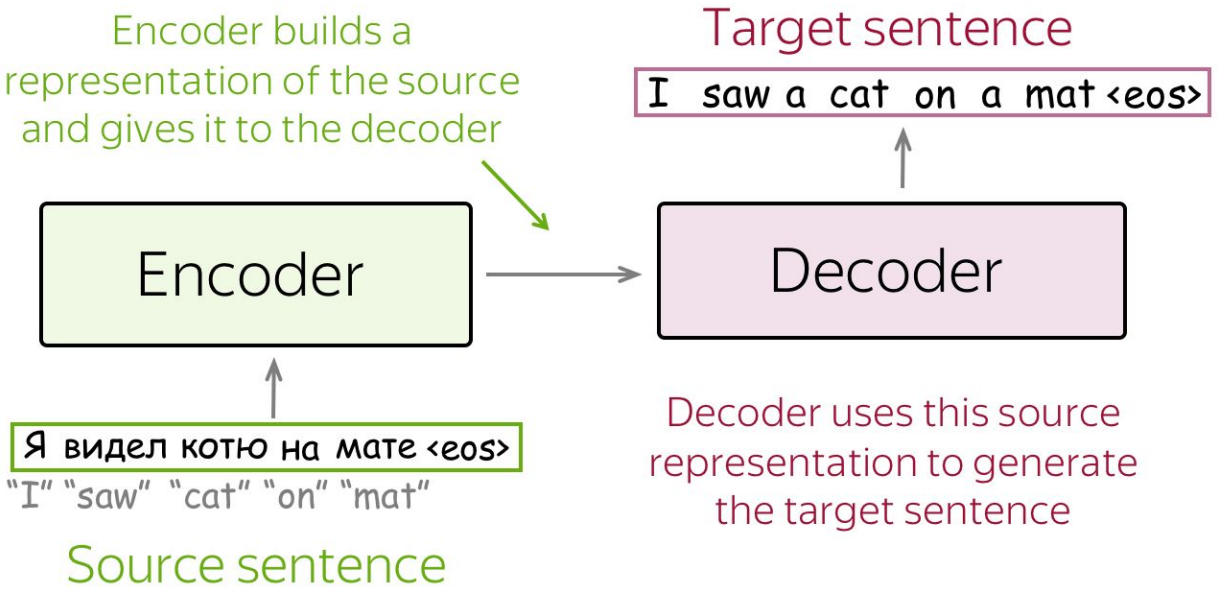


Attention Mechanism

Encoder-Decoder Framework

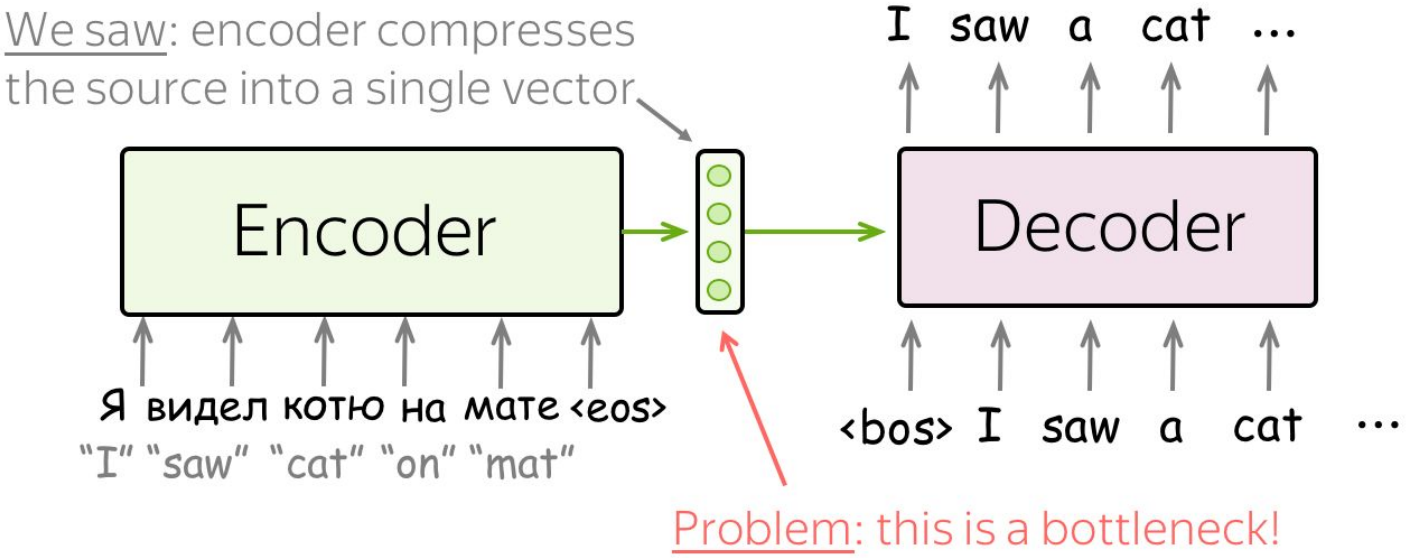
Encoder - reads source sequence and produces its representation;

Decoder - uses source representation from the encoder to generate the target sequence.



Sequence-to-sequence: the bottleneck problem

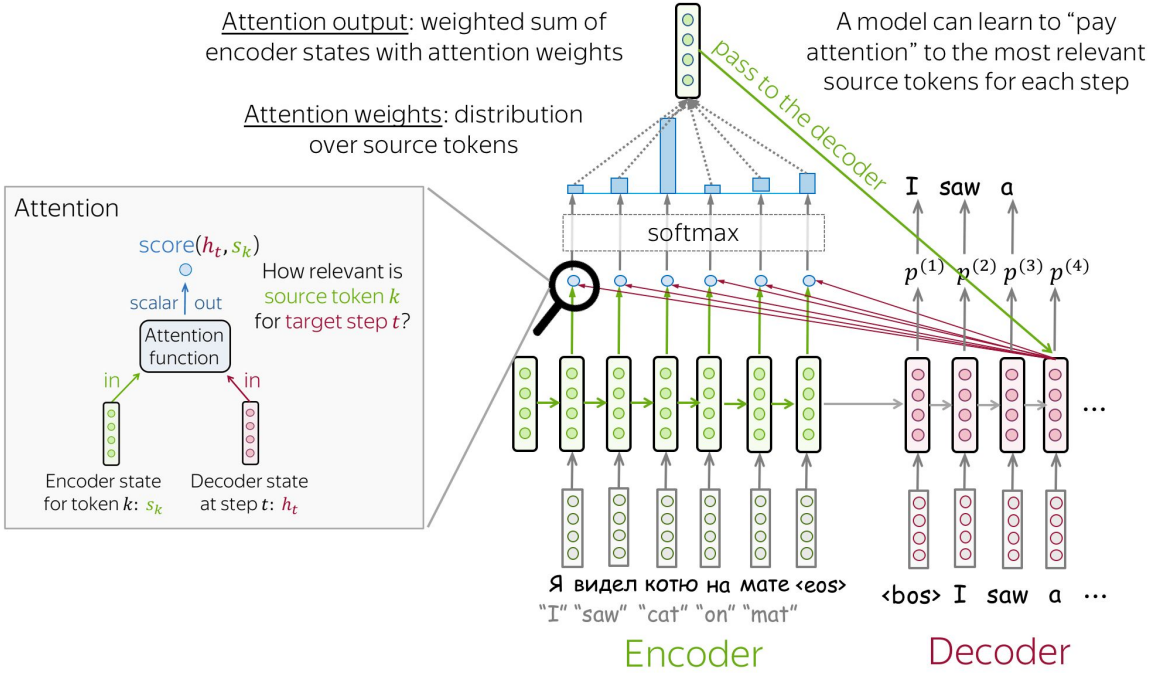
Problem: Fixed source representation is suboptimal: (i) for the encoder, it is hard to compress the sentence; (ii) for the decoder, different information may be relevant at different steps.



Attention: A High-Level View

Attention: At different steps, let a model "focus" on different parts of the input.

- At each decoder step, it decides which source parts are more important.



Attention: A High-Level View

Attention output

$$c^{(t)} = a_1^{(t)} s_1 + a_2^{(t)} s_2 + \cdots + a_m^{(t)} s_m = \sum_{k=1}^{m} a_k^{(t)} s_k$$

"source context for decoder step t"

↑ (weighted sum)

Attention weights

$$a_k^{(t)} = \frac{\exp(\text{score}(h_t, s_k))}{\sum_{i=1}^{m} \exp(\text{score}(h_t, s_i))}, \text{k} = 1..\text{m}$$

"attention weight for source token k at decoder step t"

↑ (softmax)

Attention scores

$$\text{score}(h_t, s_k), \text{k} = 1..\text{m}$$

"How relevant is source token k for target step t?"

↑

Attention input

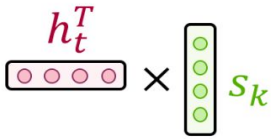
$s_1, s_2, \ldots, s_m$ — all encoder states

h_t — one decoder state

Attention Score

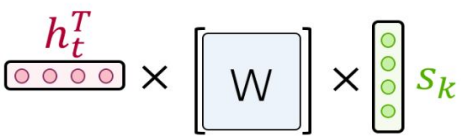
- You can apply any function you want - even a very complicated one.
- The most popular ways to compute attention scores are:
  - **dot-product** - the simplest method;
  - **bilinear function** (aka "Luong attention")
  - **multi-layer perceptron** (aka "Bahdanau attention")

Dot-product



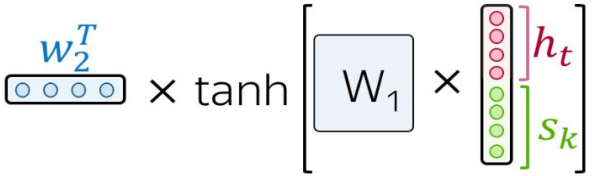
$$\text{score}(h_t, s_k) = h_t^T \; s_k$$

Bilinear



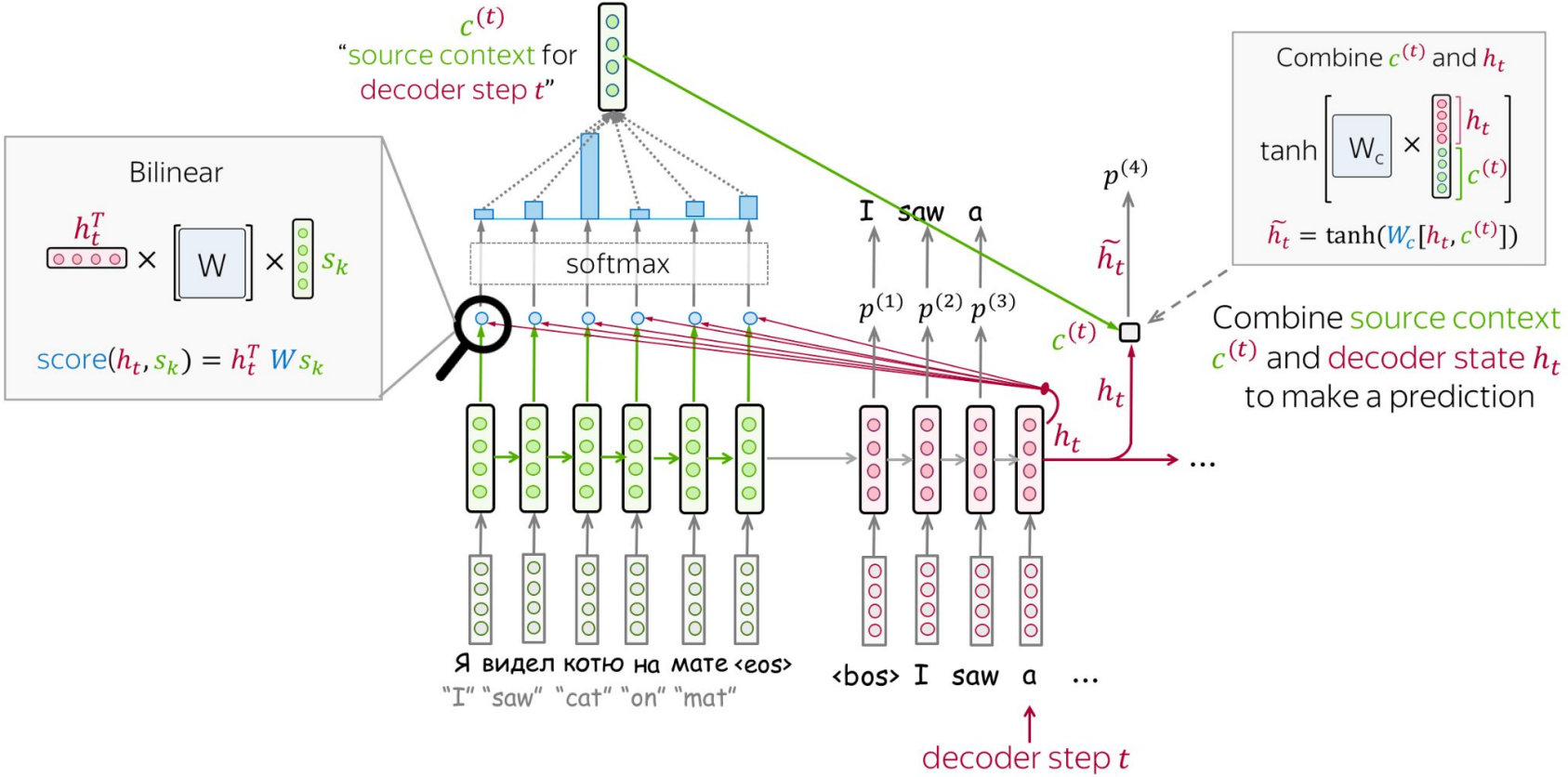
$$\text{score}(h_t, s_k) = h_t^T \; W s_k$$

Multi-Layer Perceptron

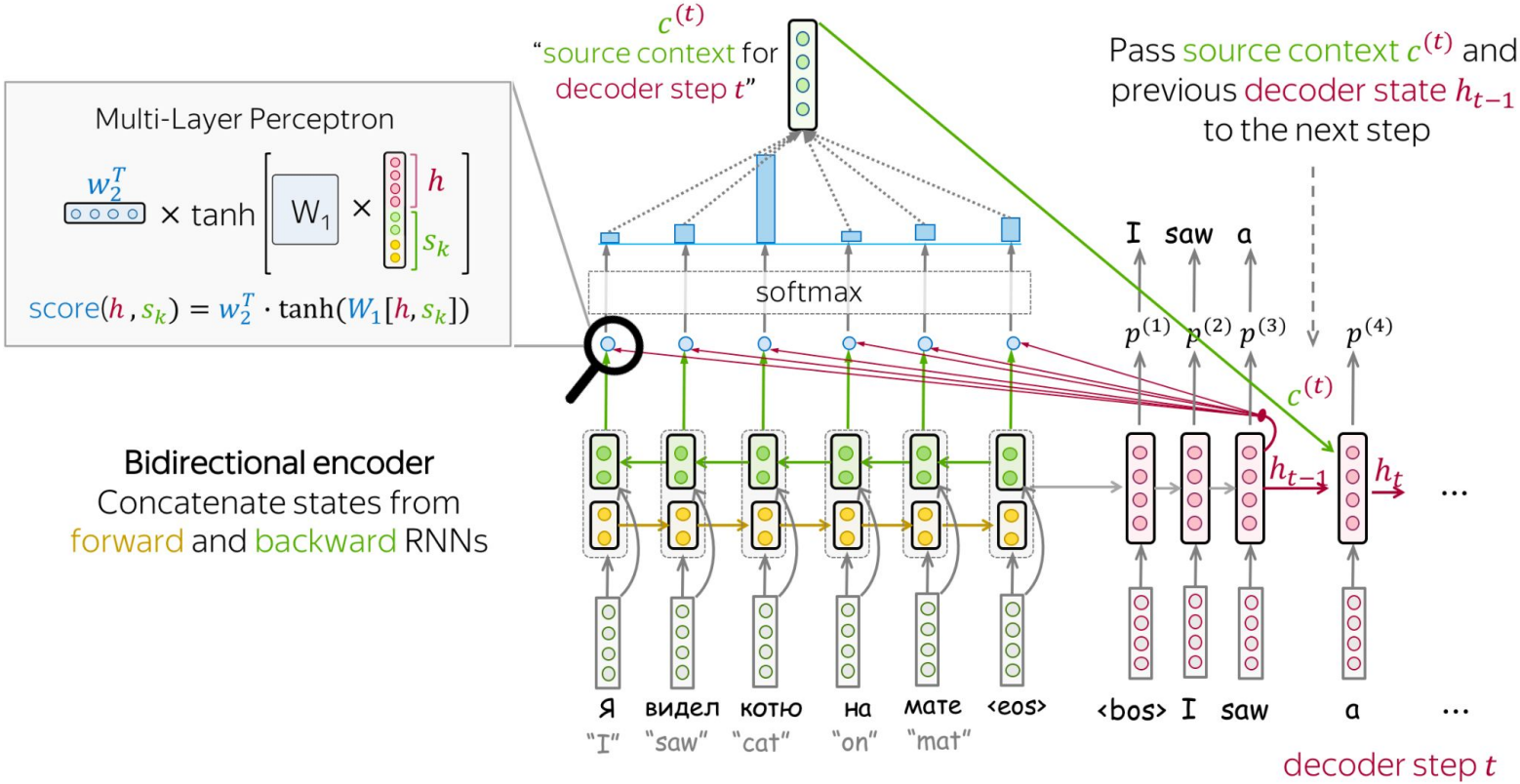


$$\text{score}(h_t, s_k) = w_2^T \cdot \tanh(W_1[h_t, s_k])$$

Attention Mechanism



Attention Mechanism



Summary

- Text classification
- Deep learning
- Neural network
- Convolution neural network
- Recurrent neural network
- Attention mechanism

Thank you

Email me
binhdt@vnu.edu.vn

Course: Large Language Models and Its Applications
sponsored by **KEPCO KDN Co., Ltd.**
Eco-friendly & Digital Centered Energy ICT Platform Leader