香港中文大學(深圳)
The Chinese University of Hong Kong, Shenzhen

CSC6052/5051/4100/DDA6307/ MDS5110 Natural Language Processing

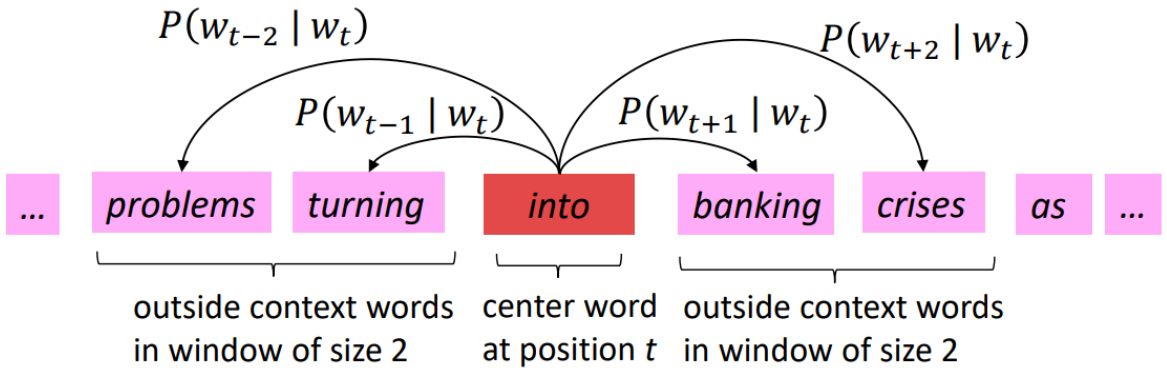
Lecture 4-1: Deep Learning in NLP

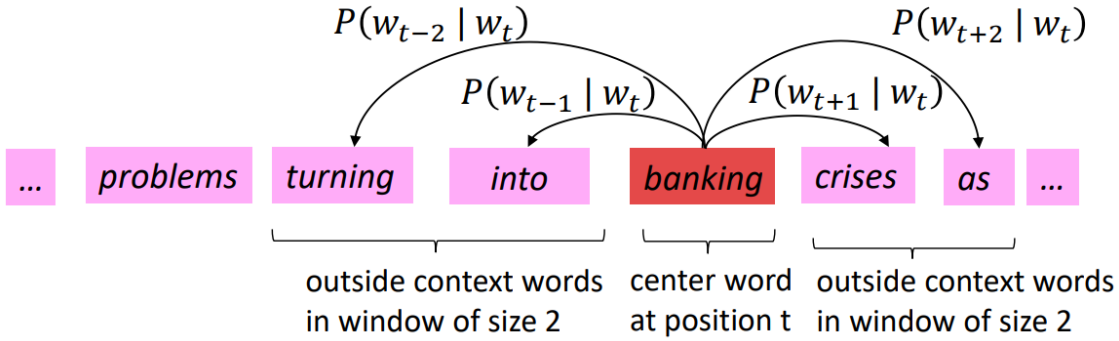
Spring 2025
Benyou Wang
School of Data Science

To recap….

Word2Vec Overview

Word2vec (Mikolov et al. 2013) is a framework for learning word vectors





https://web.stanford.edu/class/cs224n/slides/cs224n-2023-lecture01-wordvecs1.pdf

What's wrong with word2vec?

- One vector for each word type

$$v(\text{bank}) = \begin{pmatrix} -0.224 \\ 0.130 \\ -0.290 \\ 0.276 \end{pmatrix}$$

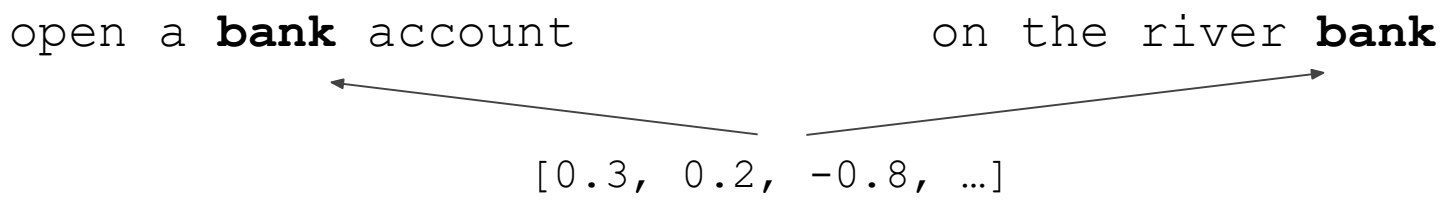
- Complex characteristics of word use: semantics, syntactic behavior, and connotations

- Polysemous words, e.g., bank, mouse

mouse^{1} : .... a *mouse* controlling a computer system in 1968.
mouse^{2} : .... a quiet animal like a *mouse*
bank^{1} : ...a *bank* can hold the investments in a custodial account ...
bank^{2} : ...as agriculture burgeons on the east *bank*, the river ...

Static vs. Contextualized

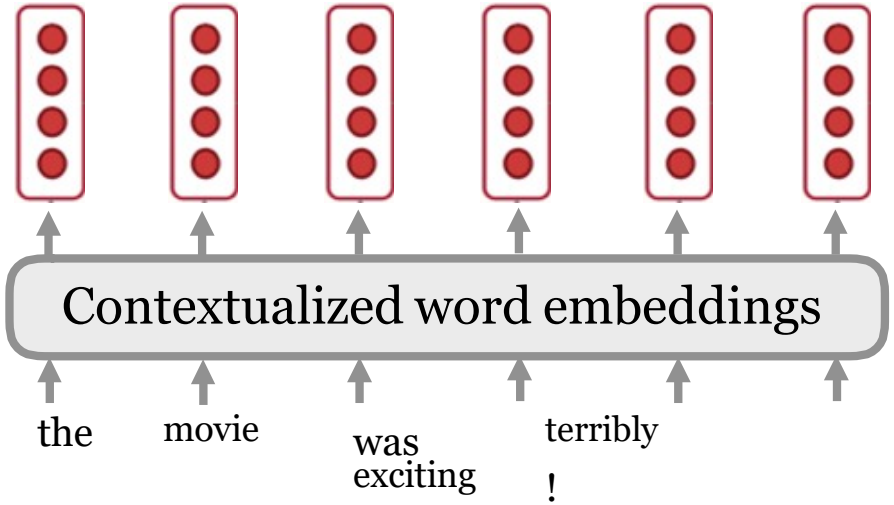
- **Problem**: Word embeddings are applied in a context free manner



- **Solution**: Train *contextual* representations on text corpus

Contextualized word embeddings

Let's build a vector for each word conditioned on its **context**!



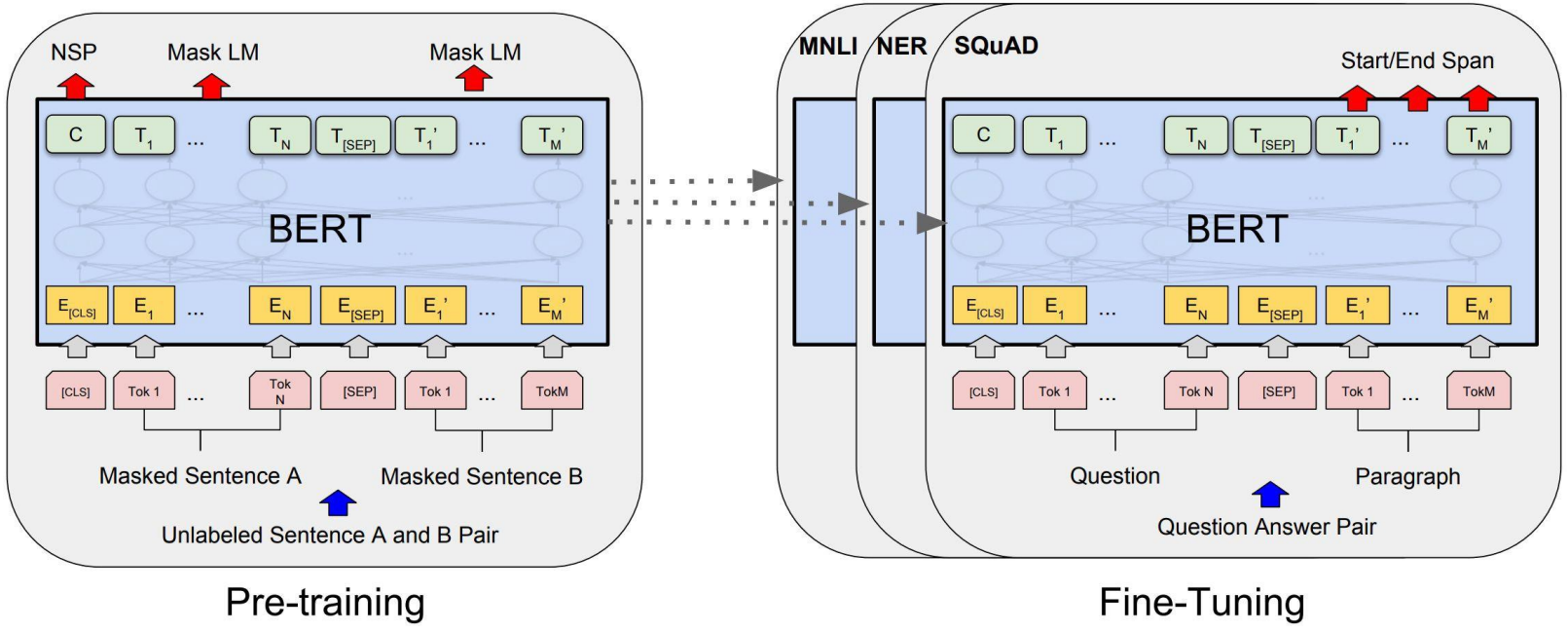
$$f : (w_1, w_2, \ldots, w_n) \longrightarrow \mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$$

Masked language models (MLMs)

- Solution: Mask out 15% of the input words, and then predict the masked words

- Too little masking: too expensive to train
- Too much masking: not enough context

One pre-trained models is adapted everywhere

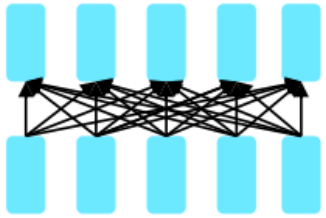


Maybe this is one of the first popular **Foundation model**

On the Opportunities and Risks of Foundation Models. https://arxiv.org/abs/2108.07258

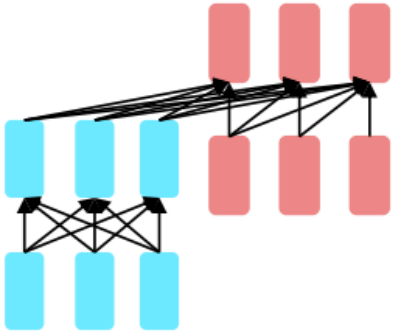
Pretraining for three types of architectures

The neural architecture influences the type of pretraining, and natural use cases.



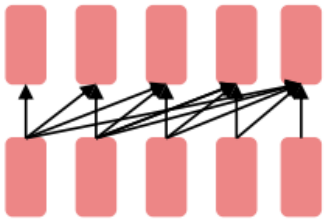
Encoders

- Gets bidirectional context – can condition on future!
- How do we train them to build strong representations?



Encoder-Decoders

- Good parts of decoders and encoders?
- What's the best way to pretrain them?



Decoders

- Language models! What we've seen so far.
- Nice to generate from; can't condition on future words

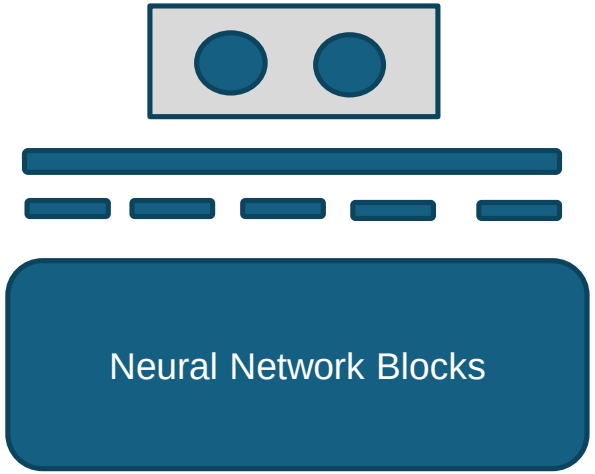
https://web.stanford.edu/class/cs224n/slides/cs224n-2023-lecture9-pretraining.pdf

Today’s Lecture– Big Picture

Typical NLP Applications

- Sentence classification
- Question answering (span prediction)
- NER (sequential labeling)
- Retrieval (document pair classification)

These tasks need either a **sentence-level** or (contextualized) **token-level** representation



Label space

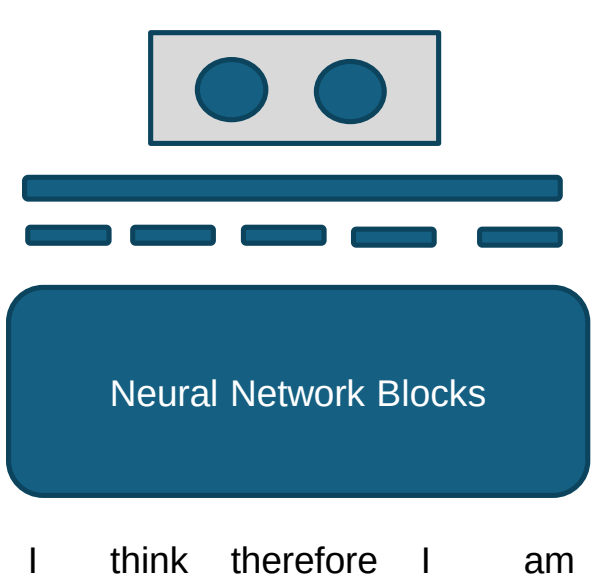
Document-level representation

Token-level representation

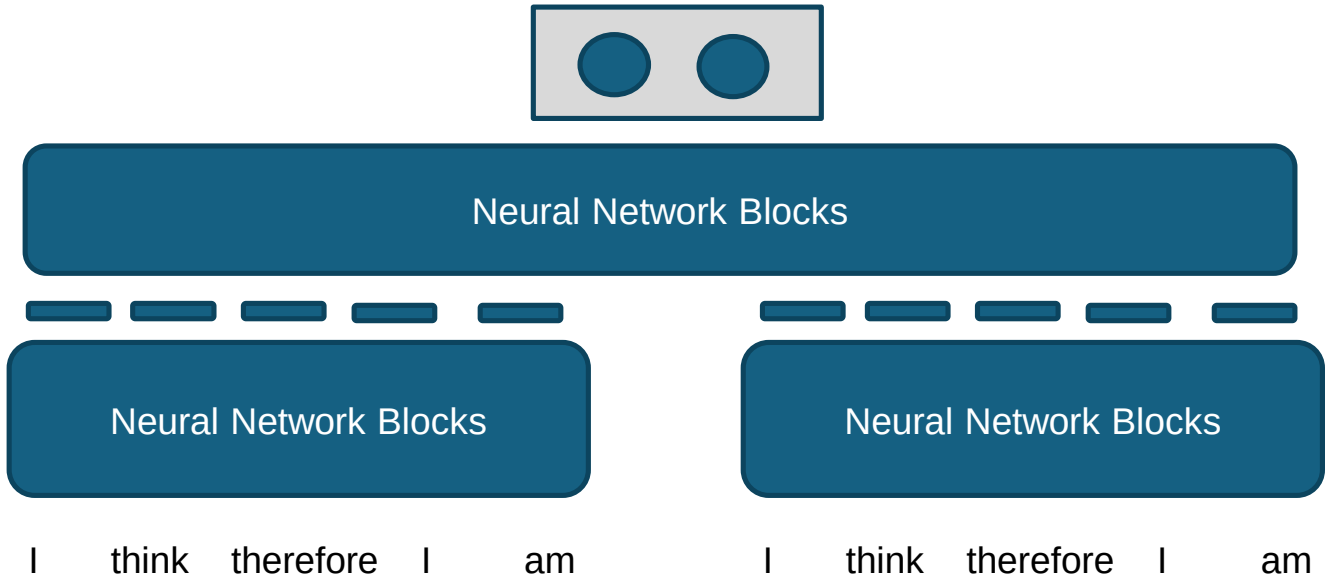
Here we refer a document as a longer “sentence”

Sentence-level tasks

Fusion at input tokens (BERT)
Fusion at the middle (Condenser)
Fusion at output (SimCSE)

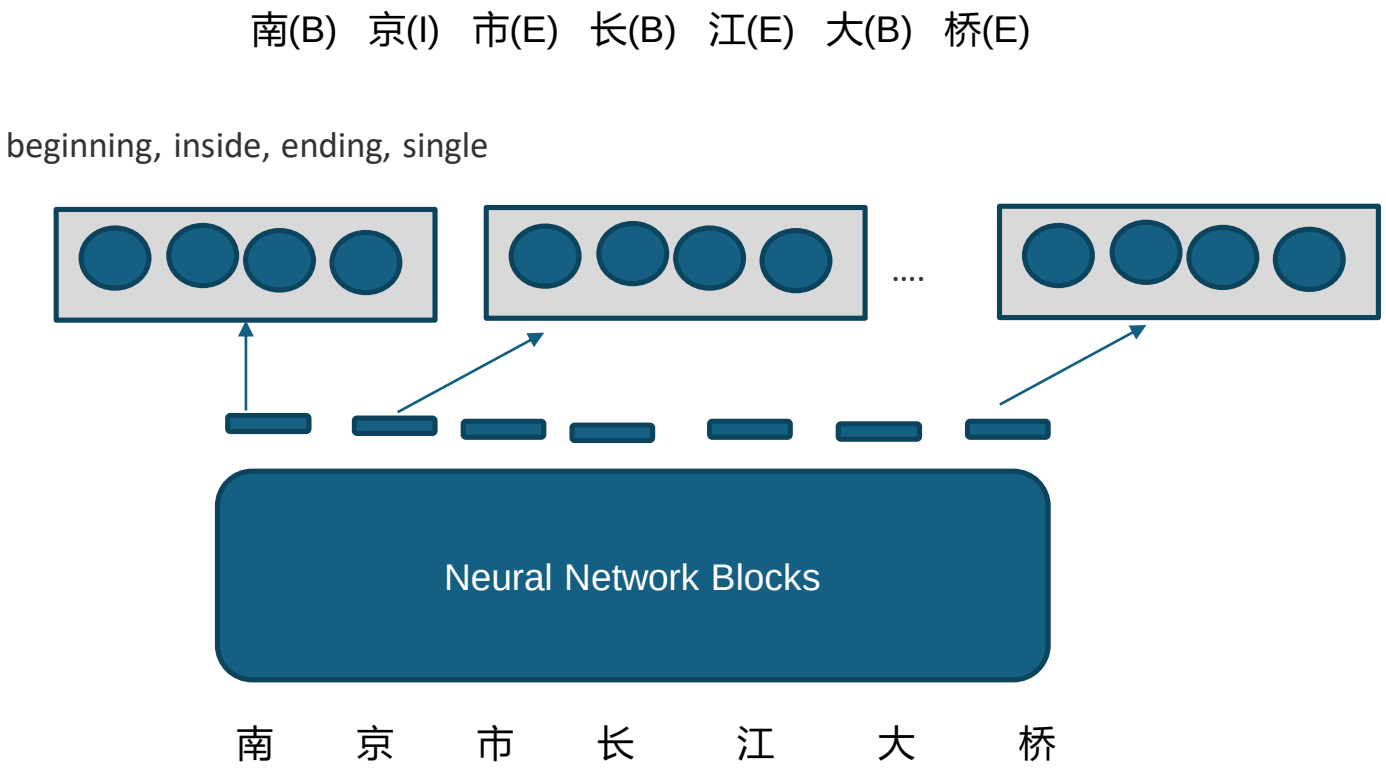


Single Sentence

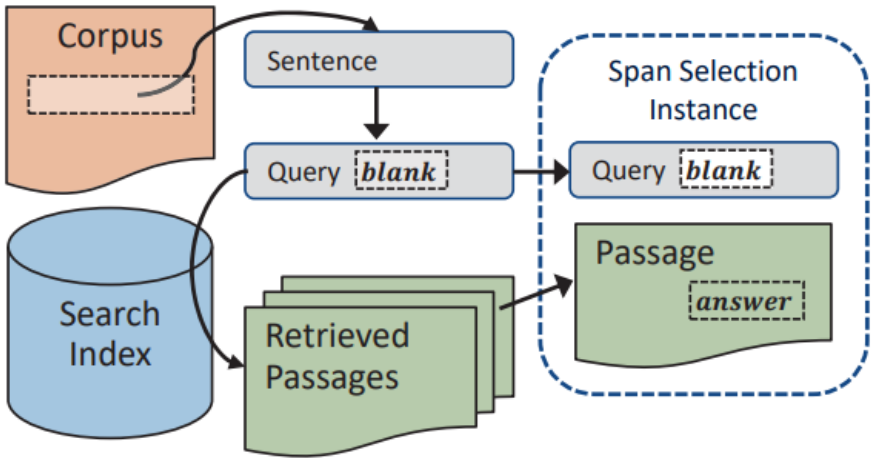


Sentence pair classification

Token-level tasks, e.g. word segmentation



Token-level tasks, e.g. span prediction



Example:

Query "In a station of the metro" is an Imagist poem by [BLANK] published in 1913 in the literary magazine Poetry

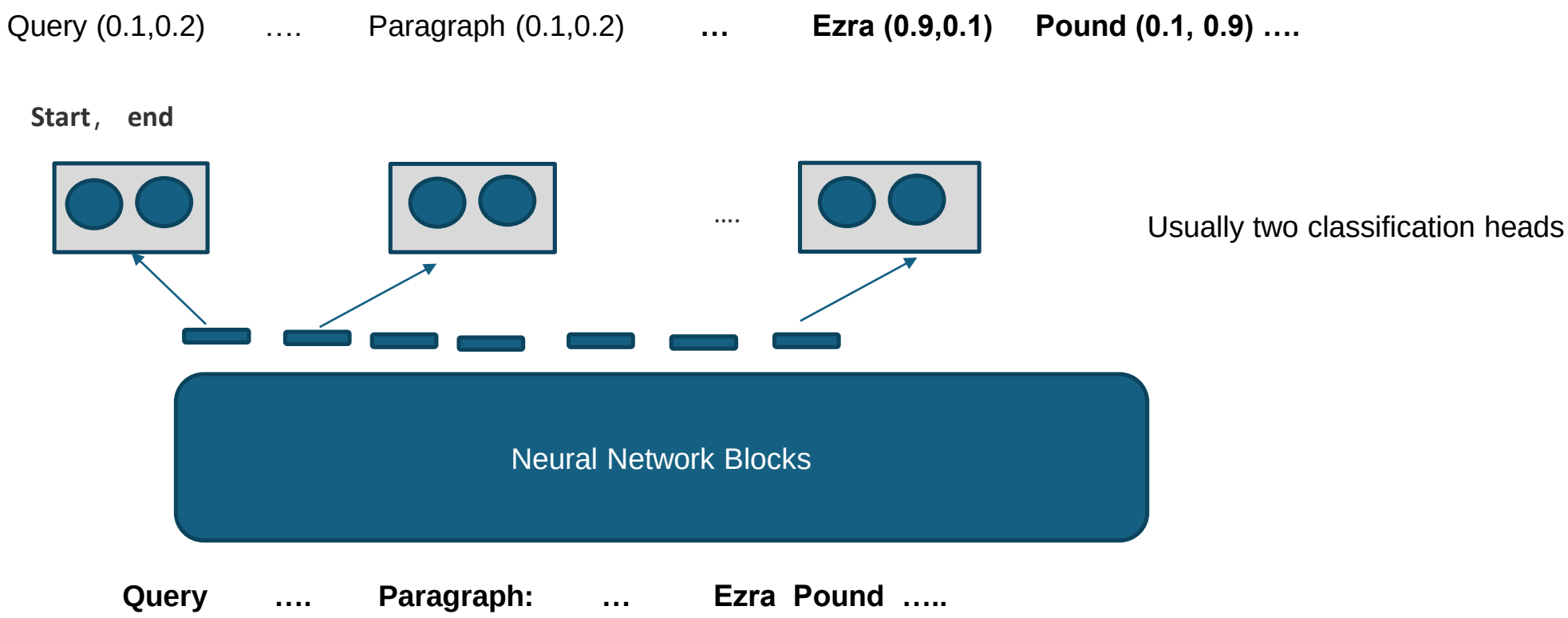
Passage . . . Ezra Pound 's famous Imagist poem, "In a station of the metro", was inspired by this station . . .

Answer Ezra Pound

Background

https://aclanthology.org/2020.acl-main.247.pdf

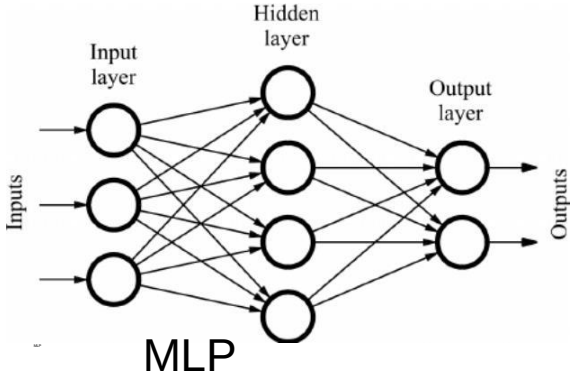
Token-level tasks, e.g. span prediction



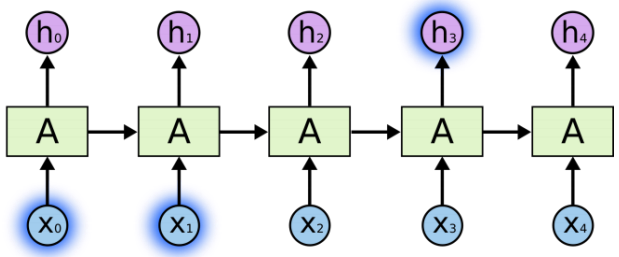
https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1194/reports/default/15792151.pdf

Which neural networks should be used for the NLP neural network backbone?

- ✔ Multilayer Perceptron (MLP)
- ✔ Convolutional neural network
- ✔ Recurrent neural network
- ✔ **Transformer**

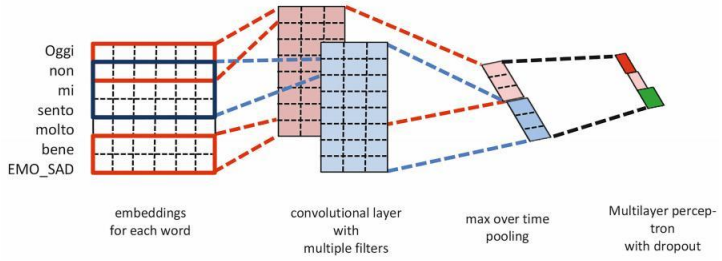


MLP

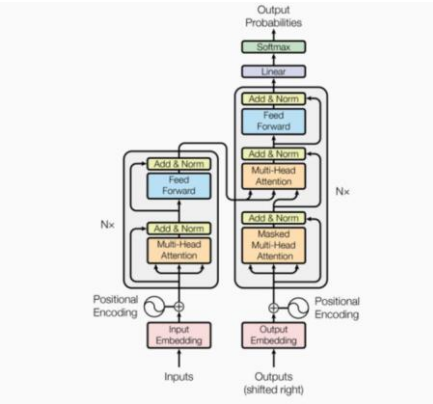


Recurrent NNs

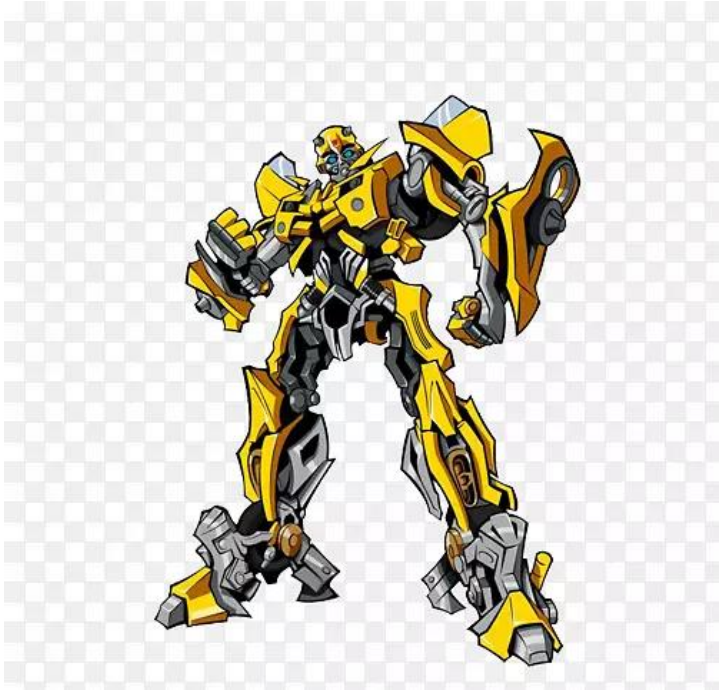
Convolutional NNs



Transformer



Why Transformer is so powerful?



Today’s lecture

- **MLP**
  +: Strongest inductive bias: if all words are concatenated
  +: Weakest inductive bias: if all words are averaged
  - : The interaction at the token-level is too weak
- CNN & RNN
  +: The interaction at the token-level is slightly better.
  CNN: Bringing the global token-level interaction to the window-level
  - : Make simplifications, its global dependencies are limited
  RNN: An ideal method for processing token sequences
  - : Its recursive nature has the problem of disaster forgetting.
- Transformer
  +: Achieve **global dependence** at the **token-level** by **decoupling** token-level interaction and feature-level abstraction into two components, in **SAN** and **FNN**.

- Scaling law and emergent ability

Semantic Abstraction and Semantic composition

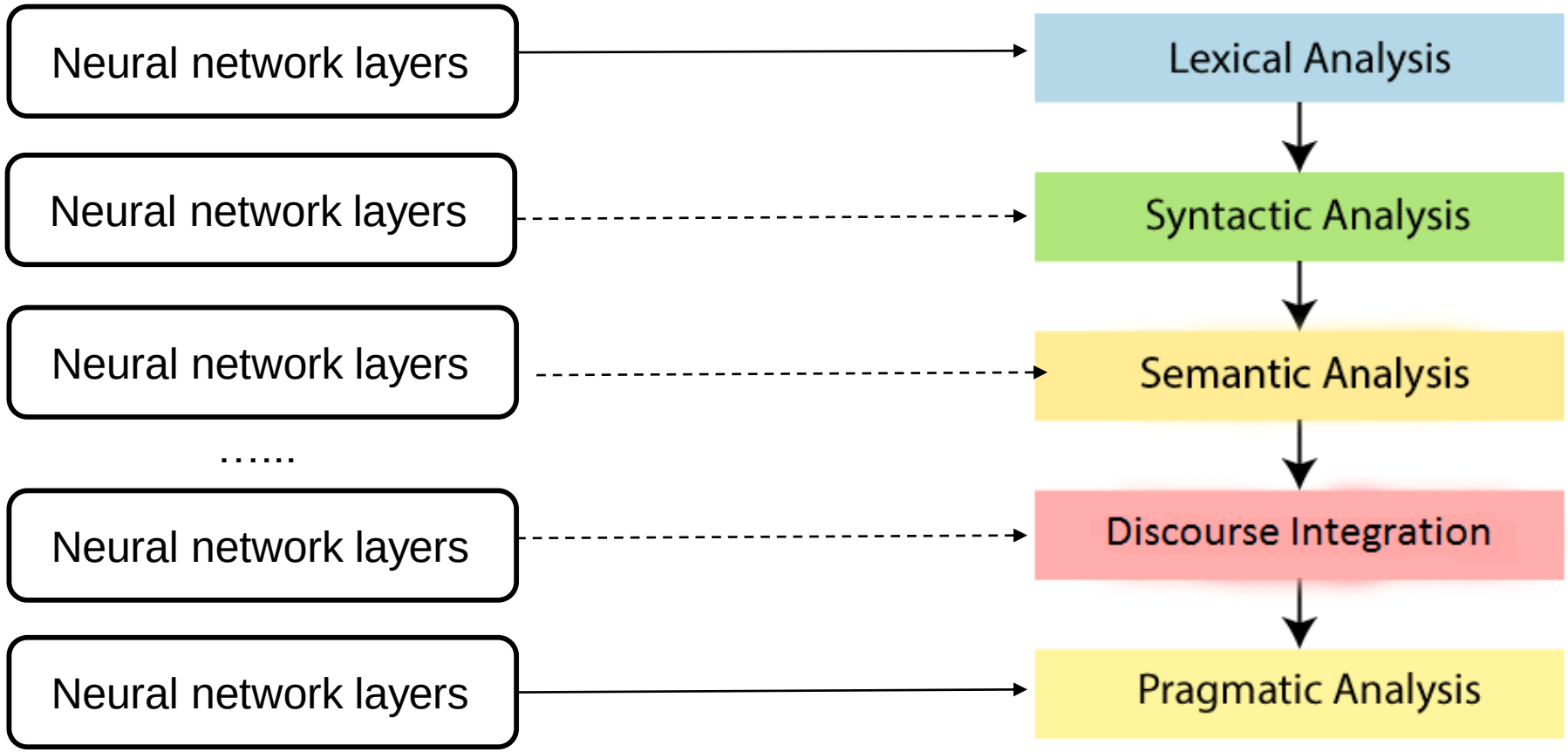
(I) What is Semantic abstraction?

Pixel -> texture -> region -> object -> relation -> semantics-> …

Chao Ma, Jia-Bin Huang, Xiaokang Yang, and Ming-Hsuan Yang. https://arxiv.org/pdf/1707.03816. ICCV 2015

Higher-level layers deal with higher-degree abstraction

Input: I think therefore I

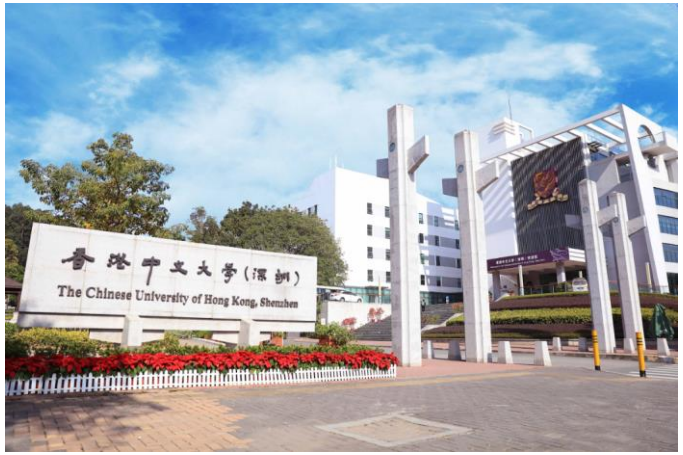


Output: am

(II) What is Semantic composition?

Ivory (象牙)

tower (塔)

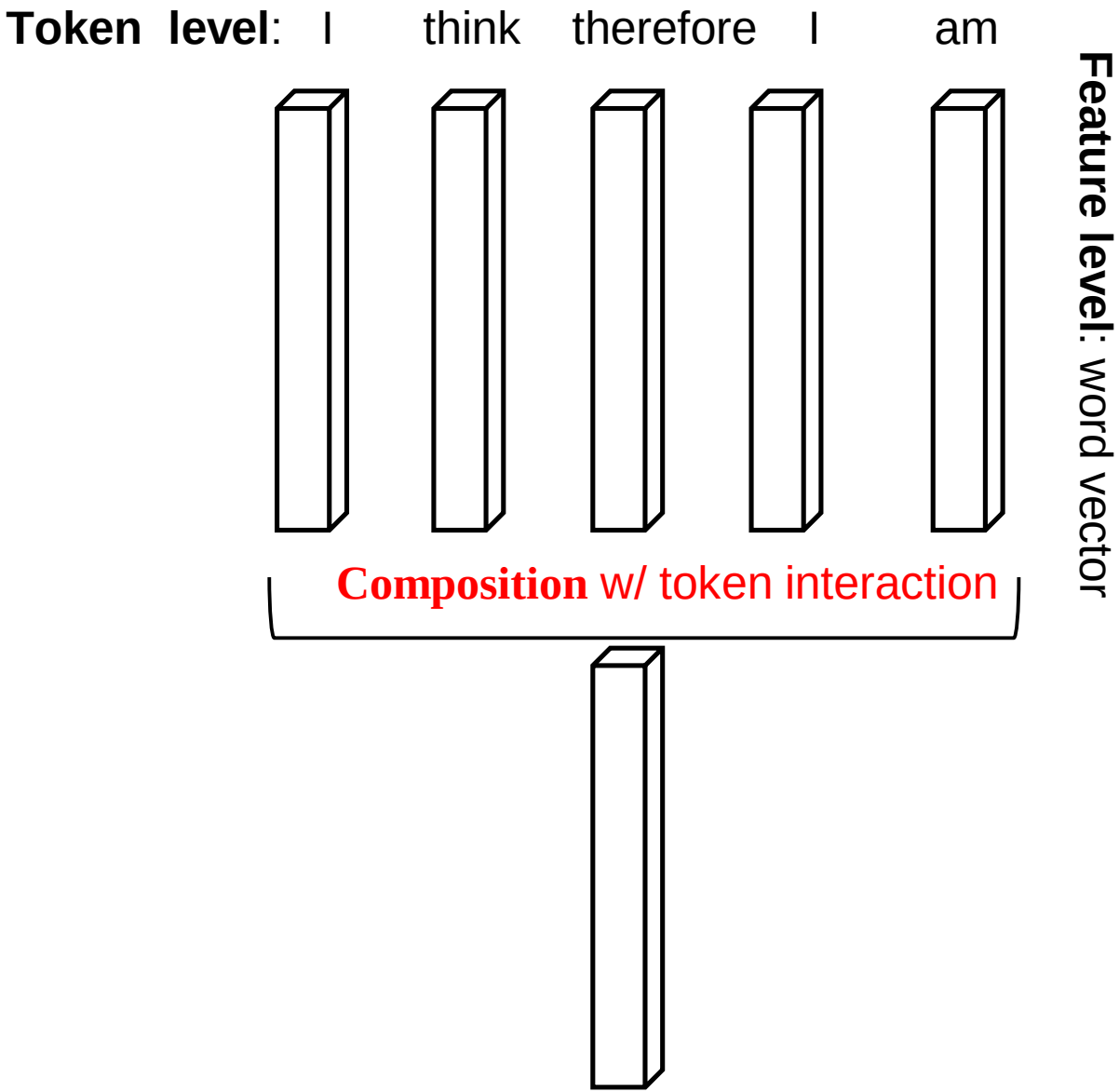


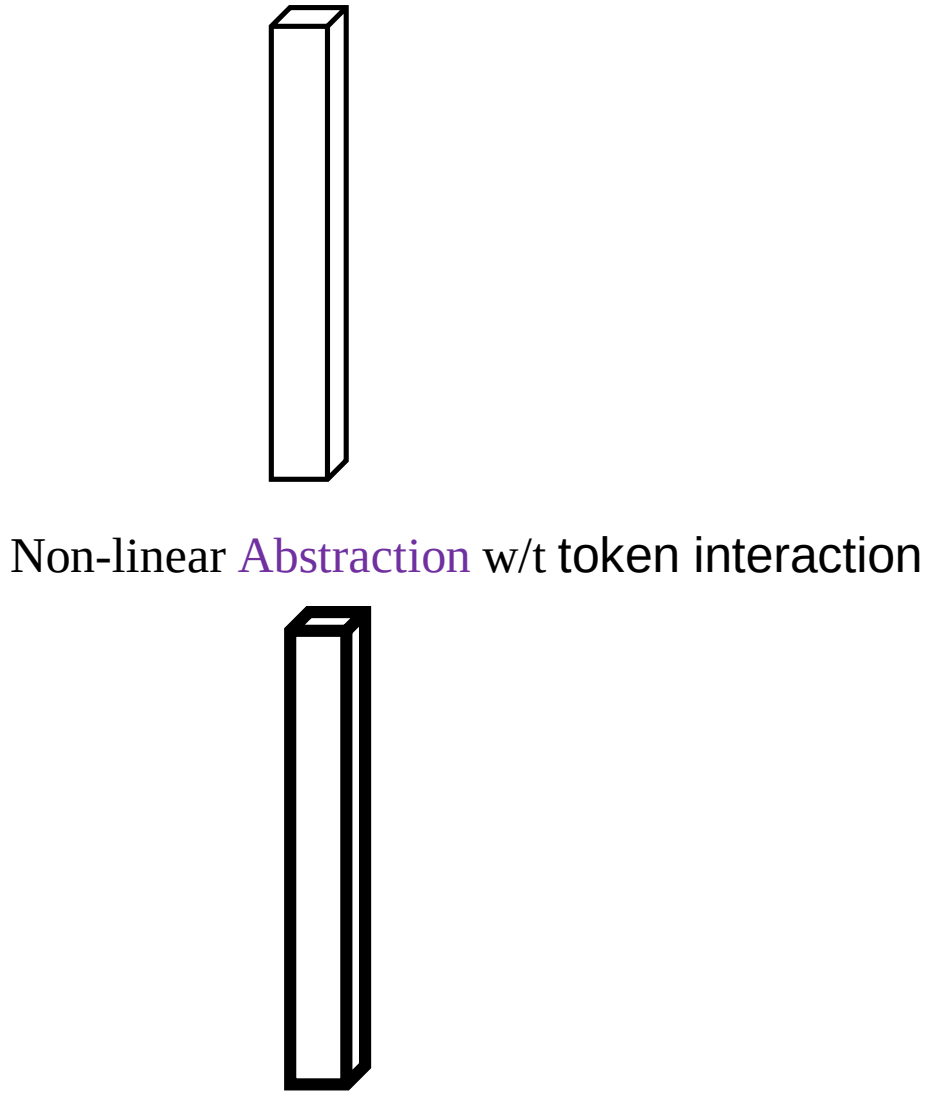
Ivory tower(象牙塔)

Semantic composition is the task of understanding the meaning of text by composing the meanings of the individual words in the text.

It involves **token interaction**

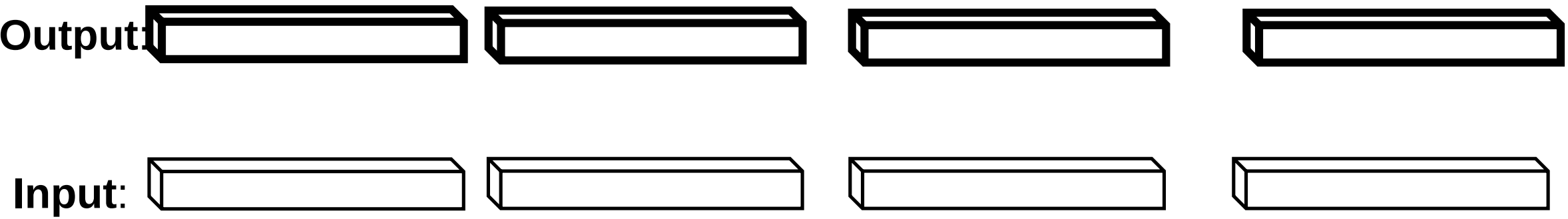
Semantic composition vs. Semantic Abstraction





How to combine composition and Abstraction

A flatten solution: MLP (e.g. NNLM)

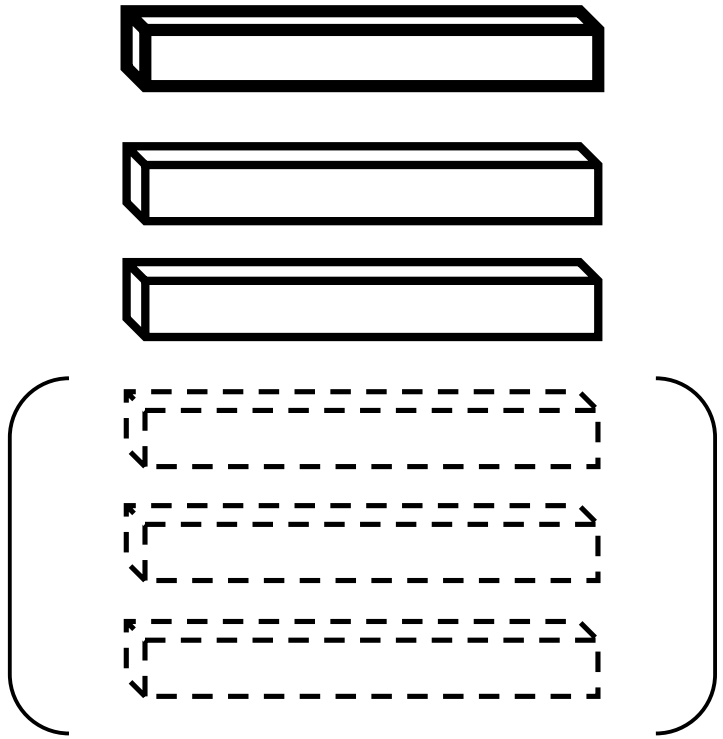


Complexity: O(D^2L^2)

Yoshua Bengio et.al A Neural Probabilistic Language Model. NIPS 2003

How to combine composition and Abstraction

A variant of MLP (e.g. CBoW**)**



Remove token interaction in deeper layers

Mean pooling (token interaction) in the first layer

Complexity: O(D^2)

T Mikolov et.al Efficient Estimation of Word Representations in Vector Space. https://arxiv.org/abs/1301.3781

Inductive bias of composition

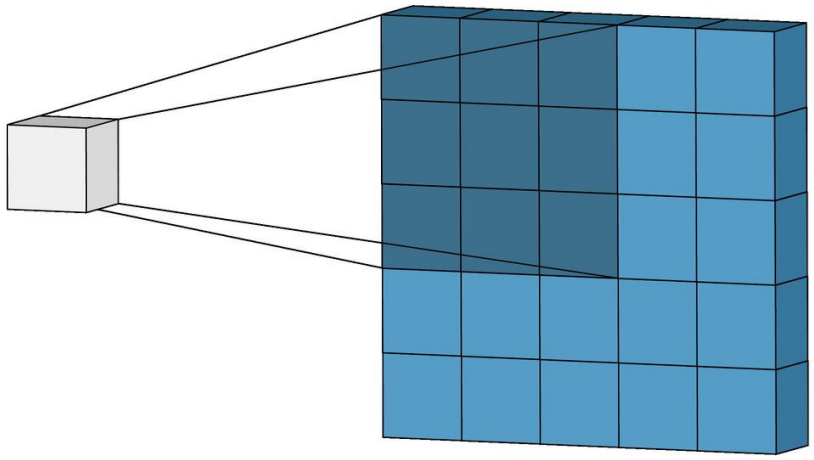
How we believe **tokens should be interacted** as the inductive bias, also considering semantic abstraction simultaneously?

Definition: The inductive bias (a.k.a learning bias) of a learning algorithm is the set of assumptions that a machine learning algorithm makes about the relationship between input variables (features) and output variables (labels) based on the training data.

Inductive bias of composition

CNN: **local** composition within a window

RNN: **recurrently** compose tokens from left to right or right to left.



Issues of CNN and RNN

CNN: **local** composition：

unfriendly to long-term/global token interaction

RNN: **recurrent** composition

what if we forget tokens checked 10 timestamp ago?

Long-term token interaction

上下文：随后他立即想到自己为什么如此气愤——他之所以气愤，是因为他害怕了。在他个人处于巨大危险的情况下，贝思抛弃了他。在海底深处只剩下他们三个人，他们互相需要——他们得互相依靠。
目标句：然而贝思不可信赖，这使他感到害怕，而且
目标词：气愤

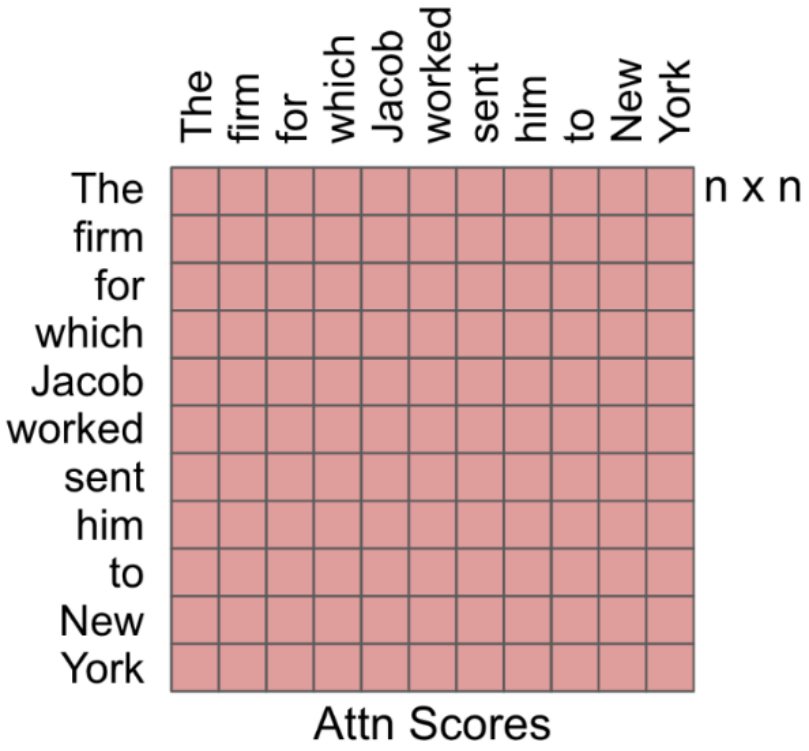
上下文：公主喜欢出宫打猎，雷章采便和公主从中原武功谈到中土风物，不但显示风流文采，还极尽体贴周到，公主就对他心生好感了。与此同时，他又物色了一个美貌妓女去勾引准驸马，让公主亲眼见了准驸马的下流丑态。
目标句：公主一怒之下，回宫禀告，就此废了这个
目标词：驸马

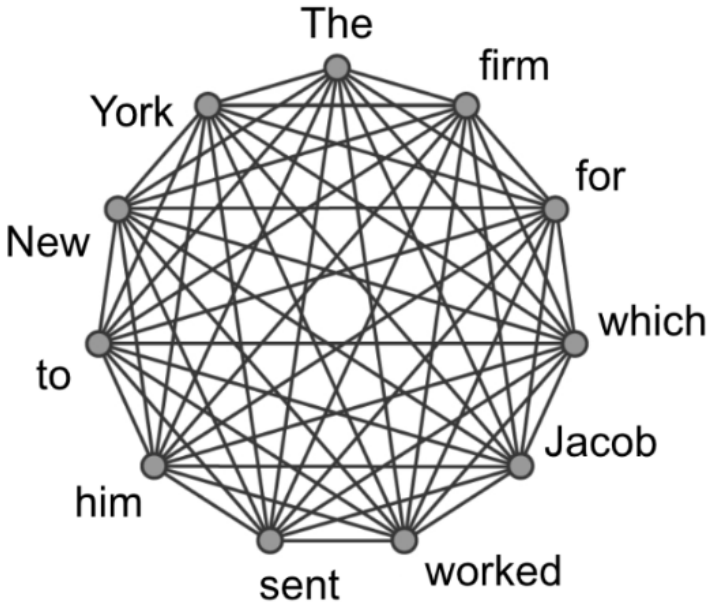
Huibin Ge, Chenxi Sun, Deyi Xiong, Qun Liu. Chinese WPLC: A Chinese Dataset for Evaluating Pretrained Language Models on Word Prediction **Given Long-Range Context**. https://aclanthology.org/2021.emnlp-main.306/

How can we freely compose tokens without constraints (weaker inductive bias) ?

The modern deep learning is just using weaker inductive biases and make more data-driven instead of prior-driven.

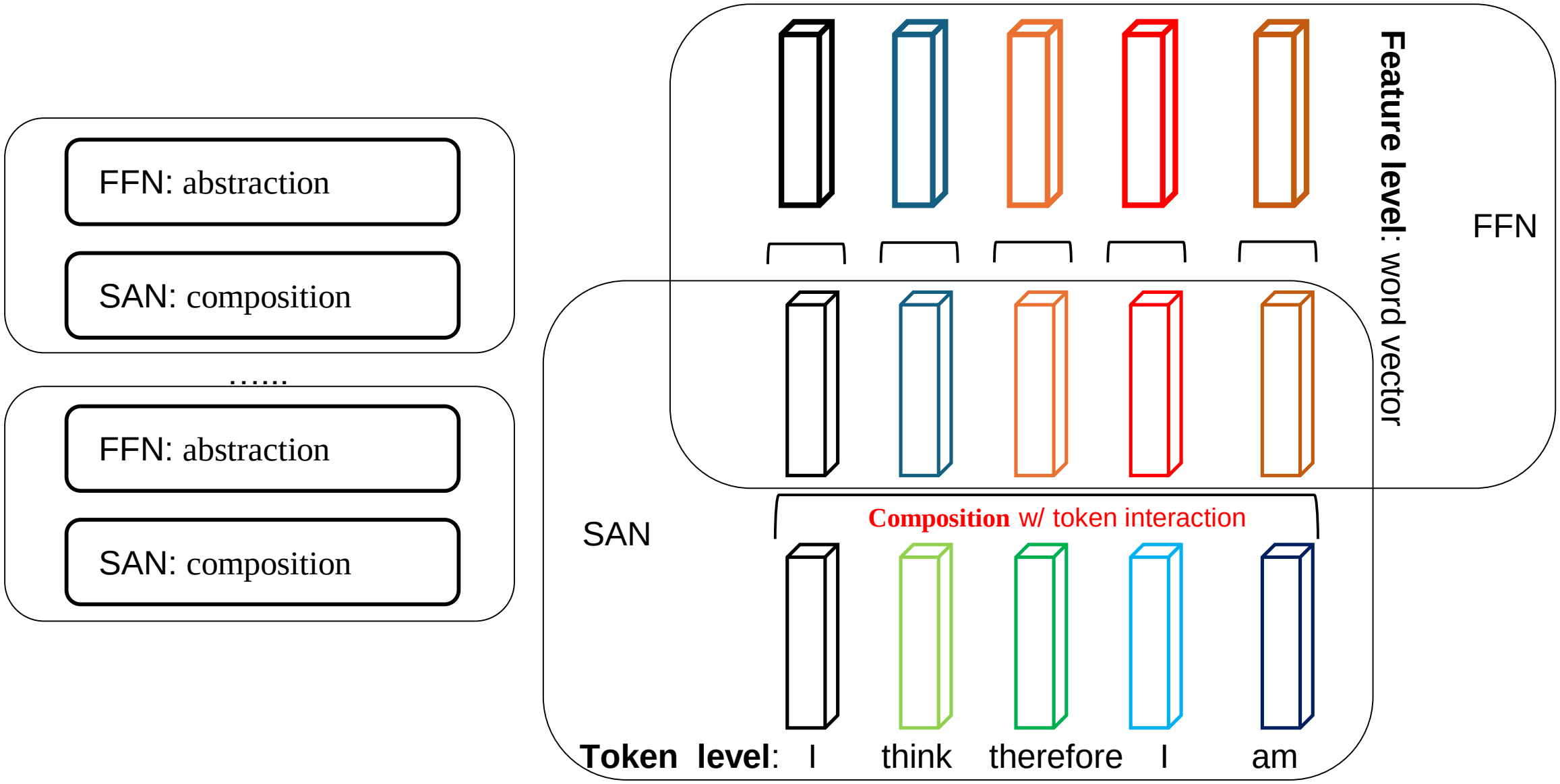
Make each token to see every other token





Difference between BERT and

Efficiency: Decompose abstraction and composition



General picture

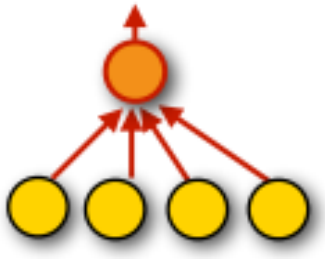
- **MLP**
  +: Strongest inductive bias: if all words are concatenated
  +: Weakest inductive bias: if all words are averaged
  - : The interaction at the token-level is too weak
- CNN & RNN
  +: The interaction at the token-level is slightly better.
  CNN: Bringing the global token-level interaction to the window-level
  - : Make simplifications, its global dependencies are limited
  RNN: An ideal method for processing token sequences
  - : Its recursive nature has the problem of disaster forgetting.
- Transformer
  +: Achieve **global dependence** at the **token-level** by **decoupling** token-level interaction and feature-level abstraction into two components, in **SAN** and **FNN**.

- Scaling law and emergent ability

Multilayer Perceptron (MLP)

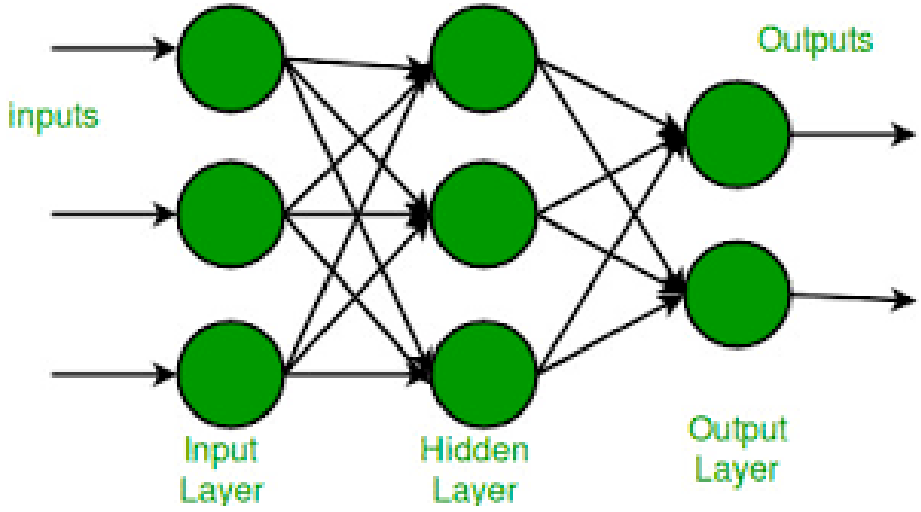
Definition: The Multilayer Perceptron (MLP) is a type of artificial neural network (ANN) that consists of multiple layers of interconnected artificial neurons or perceptrons.

A **perceptron** can be seen as a single neuron
(one output unit with a vector or **layer** of input units):



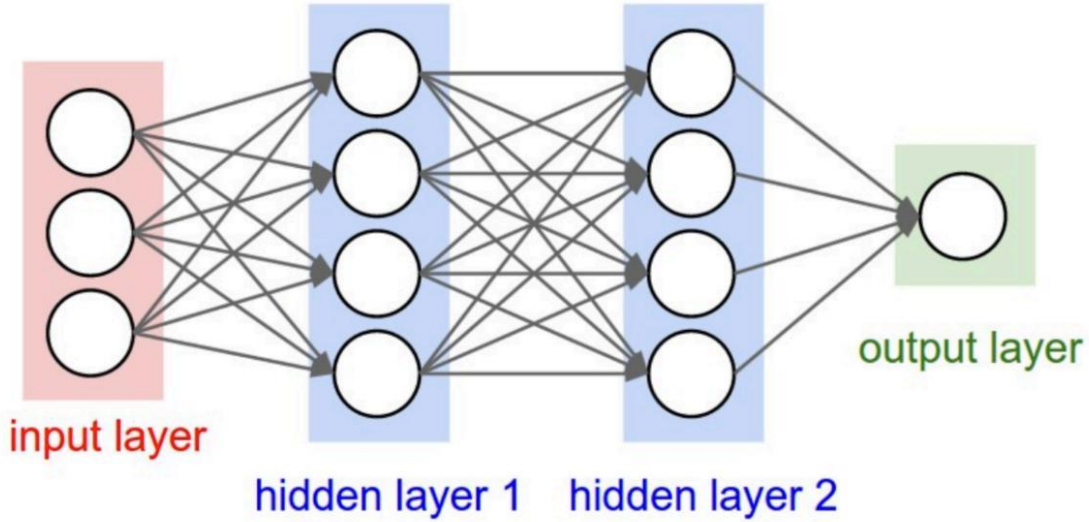
Output unit: scalar $y = f(\mathbf{x})$

Input layer: vector $\mathbf{x}$



Feed-forward NNs

- The units are connected with no cycles
- The outputs from units in each layer are passed to units in the next higher layer. No outputs are passed back to lower layers



Fully-connected (FC) layers:

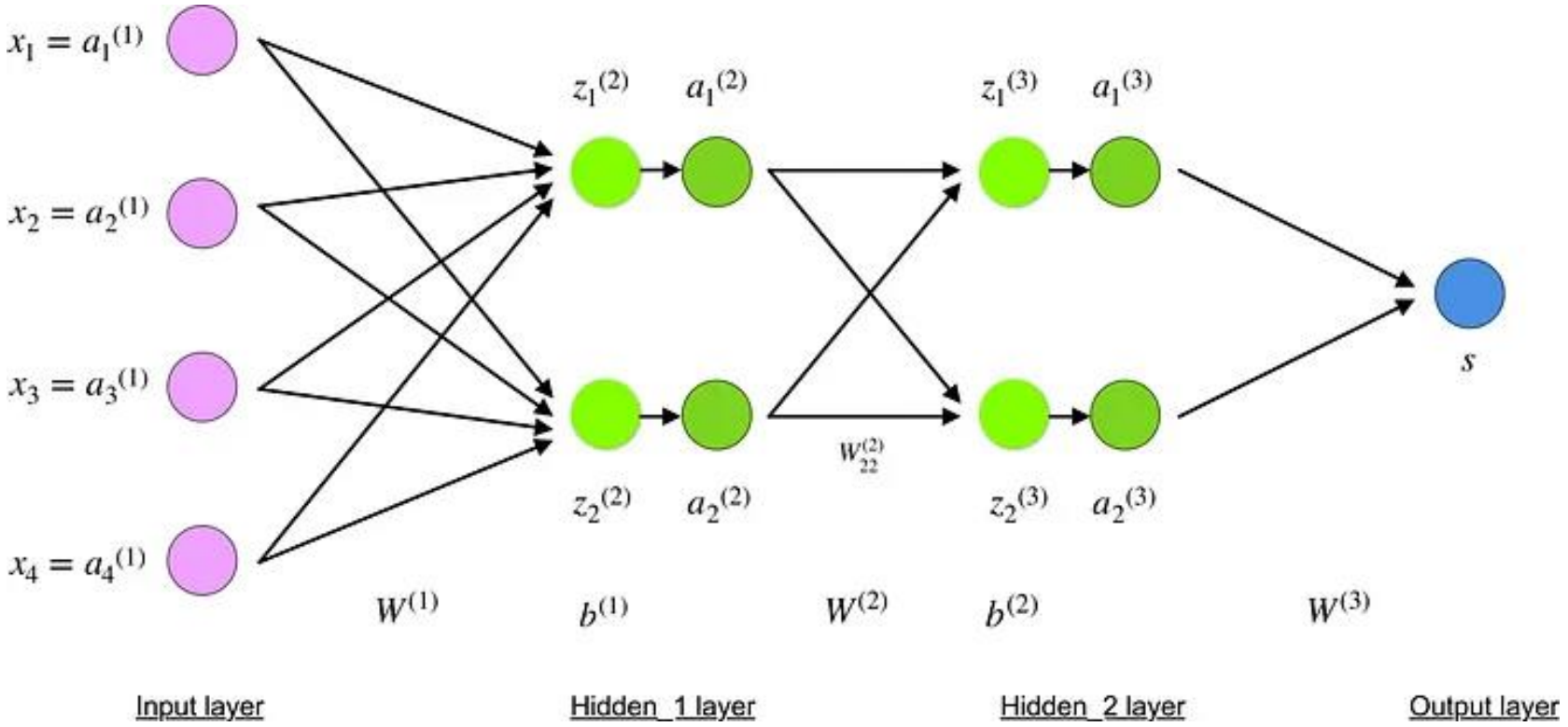
All the units from one layer are fully connected to every unit of the next layer.

```
# forward-pass of a 3-layer neural network:
f = lambda x: 1.0/(1.0 + np.exp(-x)) # activation function (use sigmoid)
x = np.random.randn(3, 1) # random input vector of three numbers (3x1)
h1 = f(np.dot(W1, x) + b1) # calculate first hidden layer activations (4x1)
h2 = f(np.dot(W2, h1) + b2) # calculate second hidden layer activations (4x1)
out = np.dot(W3, h2) + b3 # output neuron (1x1)
```

Backpropagation

Definition:

Backpropagation, short for "backward propagation of errors," is a supervised learning algorithm used for training artificial neural networks, including deep learning models like Multilayer Perceptrons (MLPs).



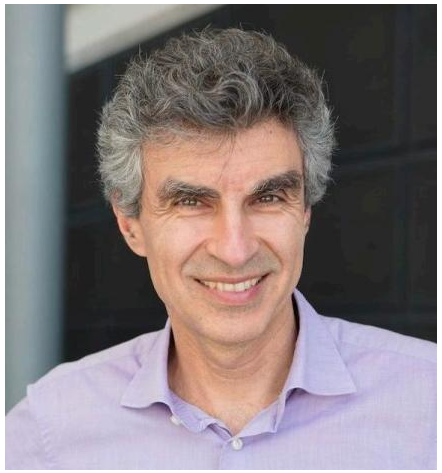
https://towardsdatascience.com/understanding-backpropagation-algorithm-7bb3aa2f95fd

Common Challenges in Backward Propagation

- Vanishing Gradients
- Exploding Gradient
- Overfitting
- Local Minima
- Poor Initialization

Summary:
- Backward propagration is a critical but challenging step in training neural networks
- Addressing these issues requires a combination of architectural choices, optimization techniques, and regularization methods.

Feedforward neural language models



A Neural Probabilistic Language Model (Bengio et al., 2003)

Yoshua Bengio BENGIOY@IRO.UMONTREAL.CA
Réjean Ducharme DUCHARME@IRO.UMONTREAL.CA
Pascal Vincent VINCENTP@IRO.UMONTREAL.CA
Christian Jauvin JAUVINC@IRO.UMONTREAL.CA

Yoshua Bengio

Probabilistic models of sequences: In the 1990s, Bengio combined neural networks with probabilistic models of sequences, such as hidden Markov models. These ideas were incorporated into a system used by AT&T/NCR for reading handwritten checks, were considered a pinnacle of neural network research in the 1990s, and modern deep learning speech recognition systems are extending these concepts.

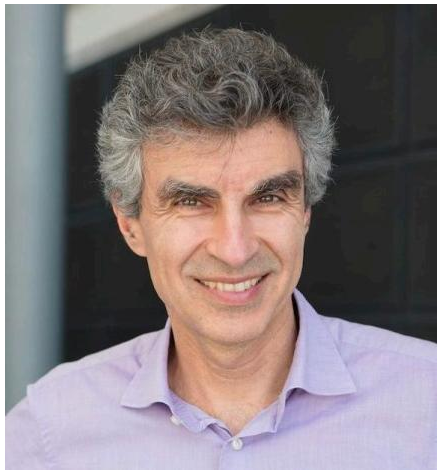
High-dimensional word embeddings and attention: In 2000, Bengio authored the landmark paper, "A Neural Probabilistic Language Model," that introduced high-dimension word embeddings as a representation of word meaning. Bengio's insights had a huge and lasting impact on natural language processing tasks including language translation, question answering, and visual question answering. His group also introduced a form of attention mechanism which led to breakthroughs in machine translation and form a key component of sequential processing with deep learning.

Generative adversarial networks: Since 2010, Bengio's papers on generative deep learning, in particular the Generative Adversarial Networks (GANs) developed with Ian Goodfellow, have spawned a revolution in computer vision and computer graphics. In one fascinating application of this work, computers can actually create original images, reminiscent of the creativity that is considered a hallmark of human intelligence.

https://awards.acm.org/about/2018-turing

Feedforward neural language models

A Neural Probabilistic Language Model (Bengio et al., 2003)



Yoshua Bengio BENGIOY@IRO.UMONTREAL.CA
Réjean Ducharme DUCHARME@IRO.UMONTREAL.CA
Pascal Vincent VINCENTP@IRO.UMONTREAL.CA
Christian Jauvin JAUVINC@IRO.UMONTREAL.CA

Key idea: Instead of estimating raw probabilities, let's use a

neural network to fit the **probabilistic distribution of language**!

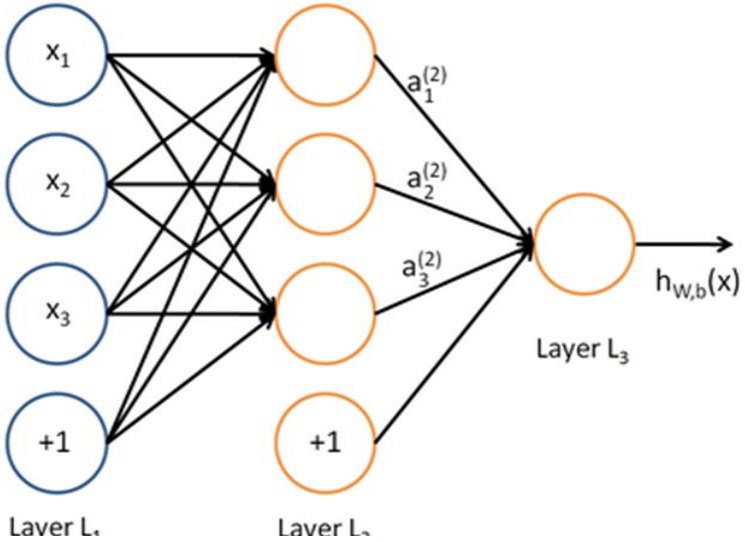
$P(w \mid \text{I am a good})$ $P(w \mid \text{I am a great})$

Key ingredient: word embeddings **e(good)** $\approx$ **e(great)**

Hope: this would give us similar distributions for similar contexts!

Review: MLP

1. Brief introduction of MLP;
2. Forward propagation and backward propagation;



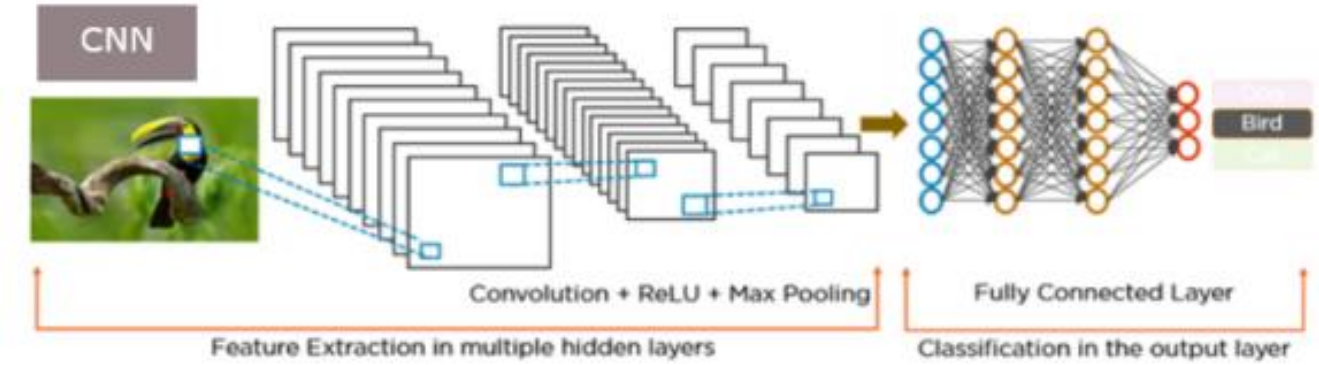
Limitations of MLP:

1. Limited Spatial Invariance (vs. CNNs)
2. Sequential Information Handling (vs. RNNs)
3. Positional Encoding (vs. Transformers)
4. Attention Mechanism (vs. Transformers)
5. Hierarchical Feature Extraction (vs. CNNs and Transformers)
6. Parameter Efficiency (vs. Transformers)
7. Pre-training Efficiency (vs. Transformers)
8. Structured Input Bias (vs. CNNs and Transformers)

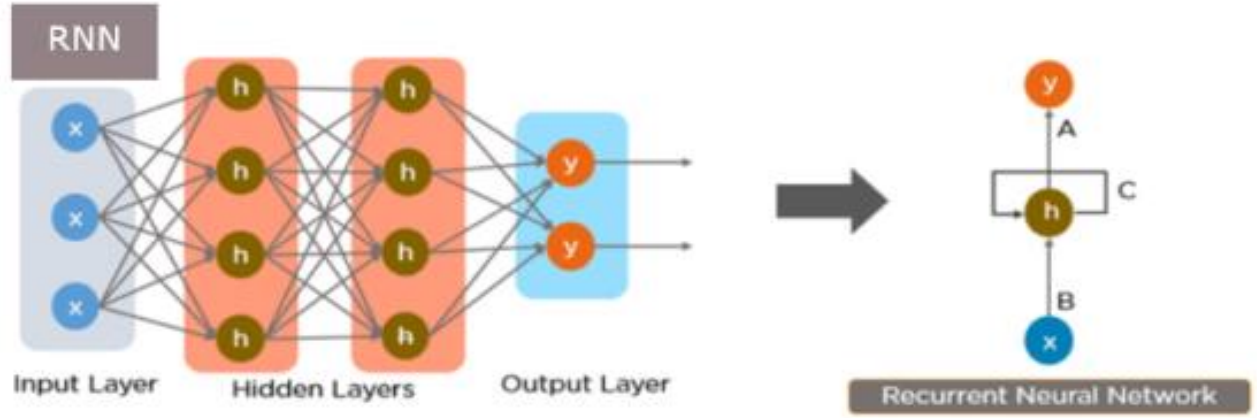
CNN&RNN

- Convolutional Neural Network (CNN)
- Recurrent Neural Network (RNN)

Convolutional Neural Network



Recurrent Neural Network

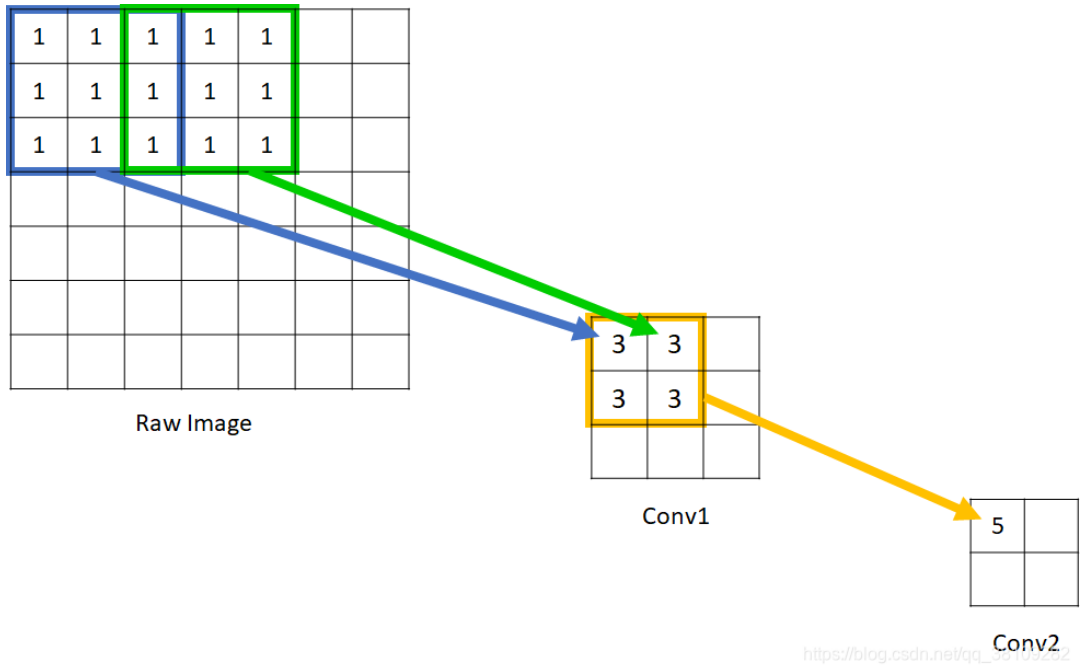


Today’s lecture

- MLP
  +: Strongest inductive bias: if all words are concated
  +: Weakest inductive bias: if all words are averaged
  - : The interaction at the token-level is too weak
- **CNN & RNN**
  +: The interaction at the token-level is slightly better.
  CNN: Bringing the global token-level interaction to the window-level
  - : Make simplifications, its global dependencies are limited
  RNN: An ideal method for processing token sequences
  - : Its recursive nature has the problem of disaster forgetting.
- Transformer
  +: Achieve **global dependence** at the **token-level** by **decoupling** token-level interaction and feature-level abstraction into two components, in **SAN** and **FNN**.

- Scaling law and emergent ability

Local receptive field

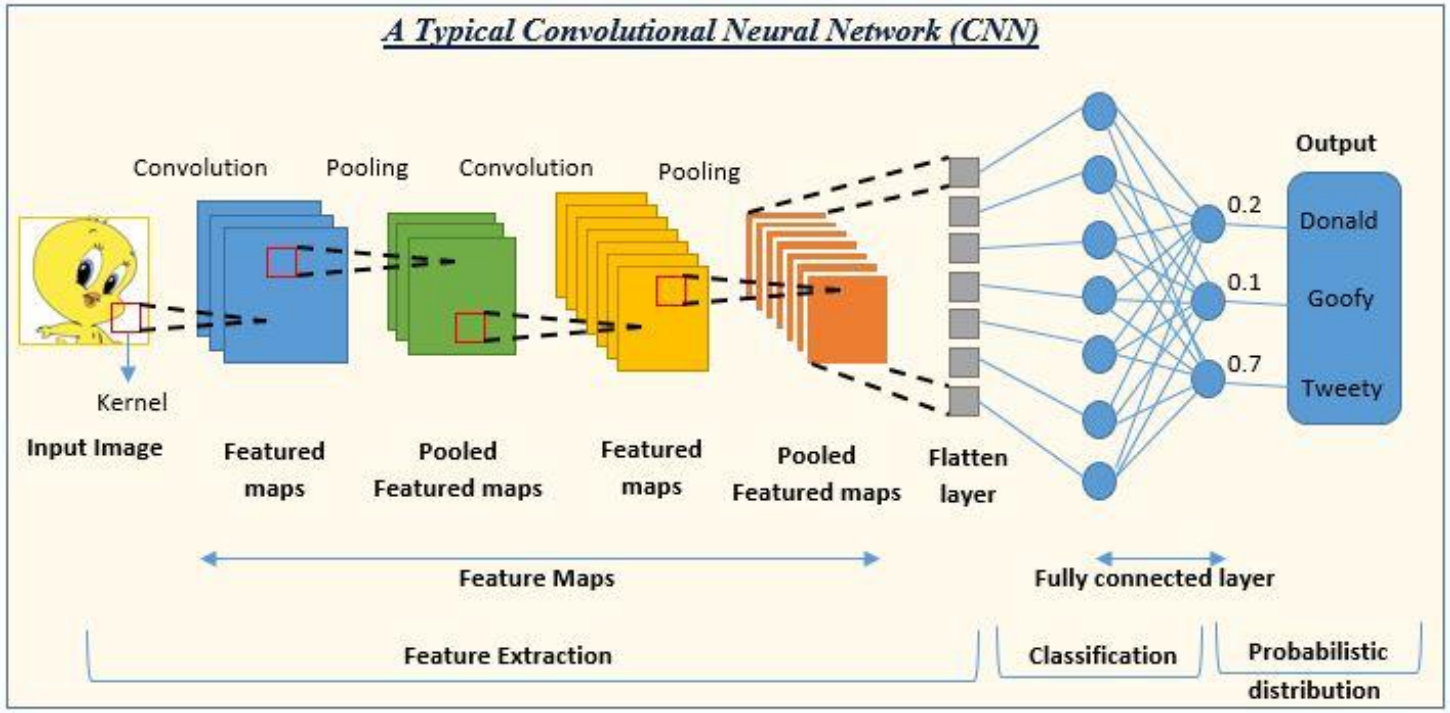


At each time， it only perceives part of the input.

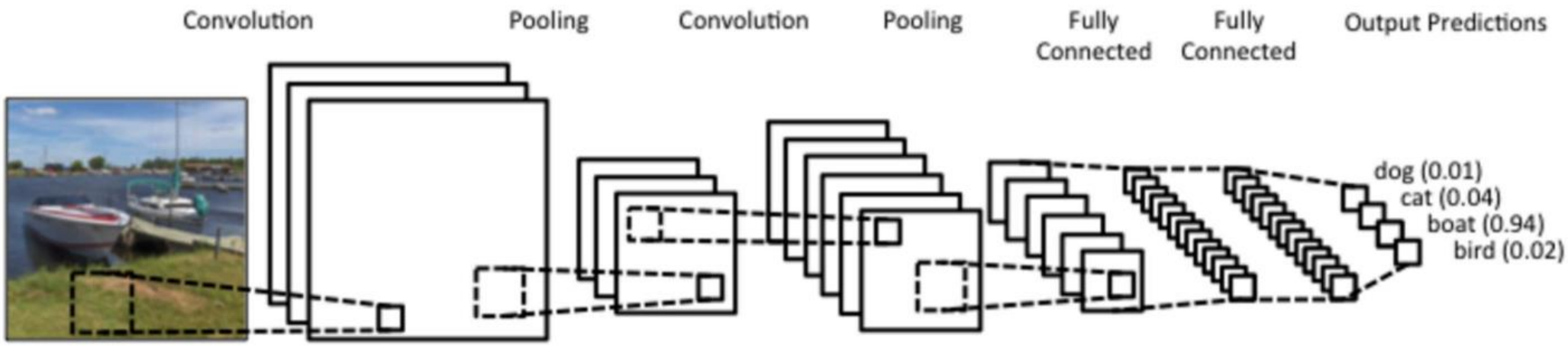
CNN

Convolutional Neural Network

- What is CNN?
- Motivation: Image Processing
- Key Components
  - Convolutional Layers
  - Pooling Layers
  - Fully Connected Layers
- Hierarchical Feature Extraction

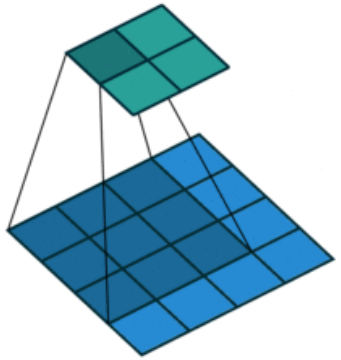


Convolutional NNs in image classification

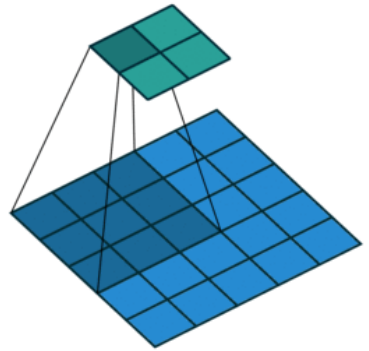


Key components: 1) convolution; 2) pooling; 3) multiple channels (feature maps)

Convolution

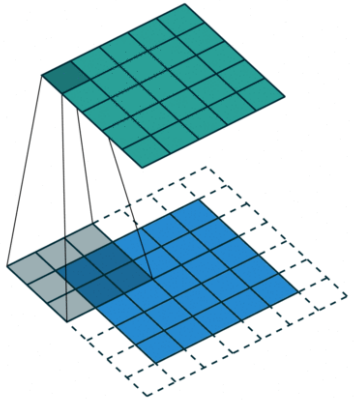


Stride =1

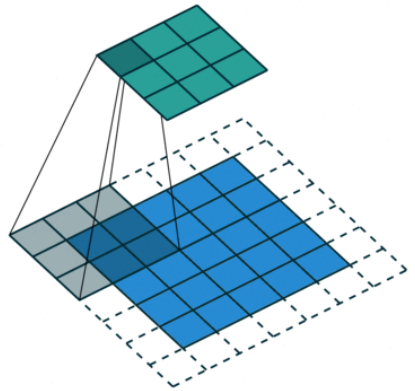


Padding = 0

Padding =1



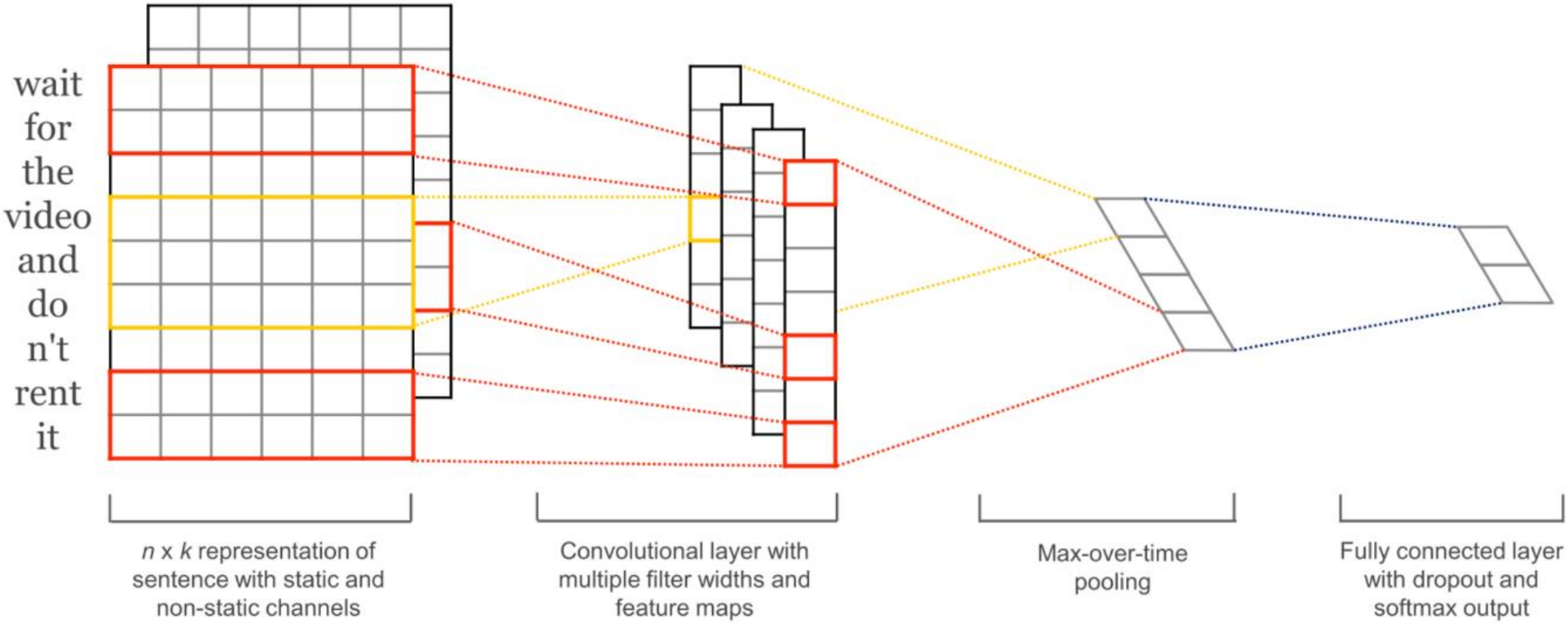
Stride =2



CNN Visualization

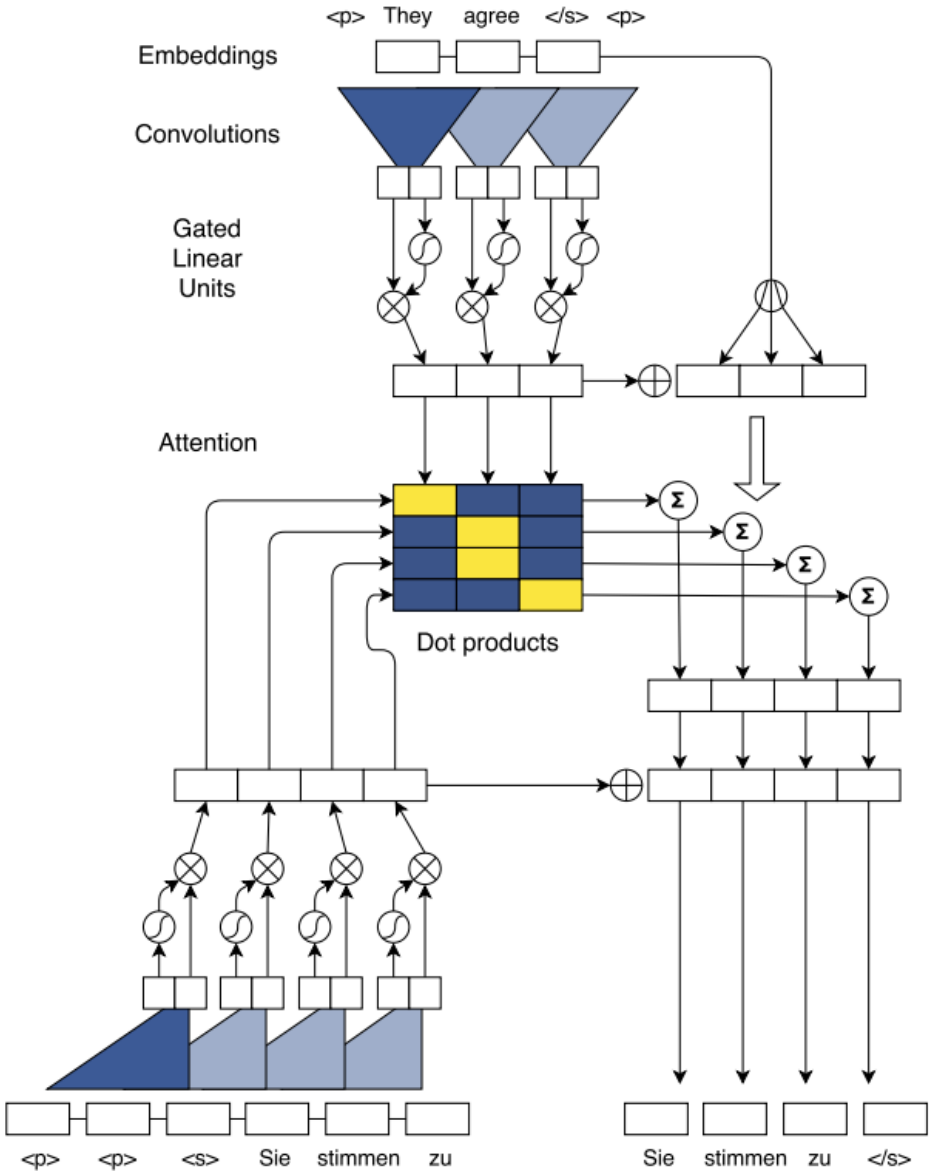
https://poloclub.github.io/cnn-explainer/
https://adamharley.com/nn_vis/cnn/3d.html

Convolutional NNs for text classification



(Kim 2014): Convolutional Neural Networks for Sentence Classification

Convolutional Sequence to Sequence Learning



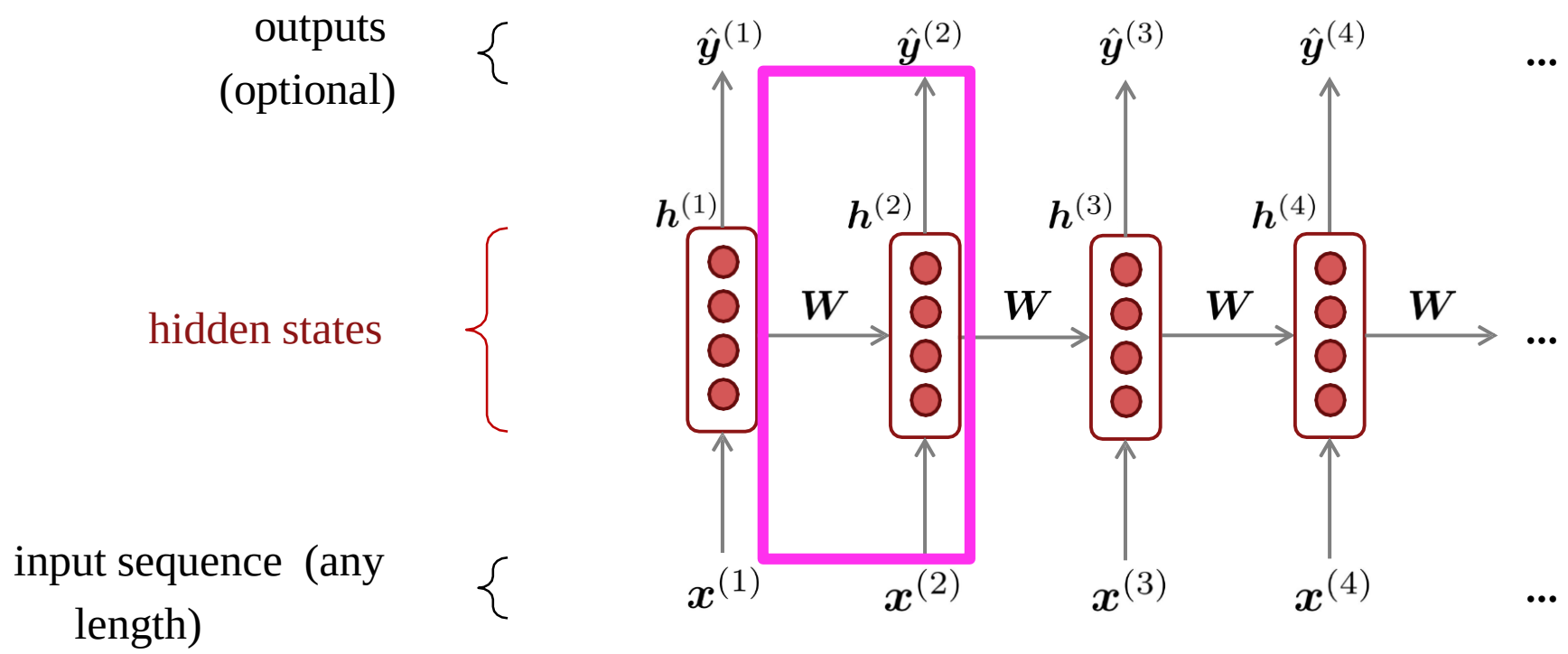
- Encoder and decoder are simple blocks of convolution operation followed by nonlinearity on fixed size of input.
- Introduce a concept of order preservation as a positional vectors p = (p_1,p_2 ...,p_m). In combination of both input elements are represented as E = (e_1=w_1+p_1, e_2=w_2+p_2, ...,e_m=w_m+p_m).
- Adds a linear mapping to project between the embedding size f and the convolution outputs that are size 2d.
- Computes a distribution over the T possible next target elements y_i+1 by transforming the top decoder output h_i_l via a linear layer with weights and bias.

https://browse.arxiv.org/pdf/1705.03122.pdf This paper inspires Transformer!

RNN

Recurrent Neural Network

Core idea: Apply the same weights *W repeatedly*



A Simple RNN Language Model

output distribution

$$\hat{\boldsymbol{y}}^{(t)} = \text{softmax}\left(\boldsymbol{U}\boldsymbol{h}^{(t)} + \boldsymbol{b}_2\right) \in \mathbb{R}^{|V|}$$

hidden states

$$\boldsymbol{h}^{(t)} = \sigma\left(\boldsymbol{W}_h\boldsymbol{h}^{(t-1)} + \boldsymbol{W}_e\boldsymbol{e}^{(t)} + \boldsymbol{b}_1\right)$$

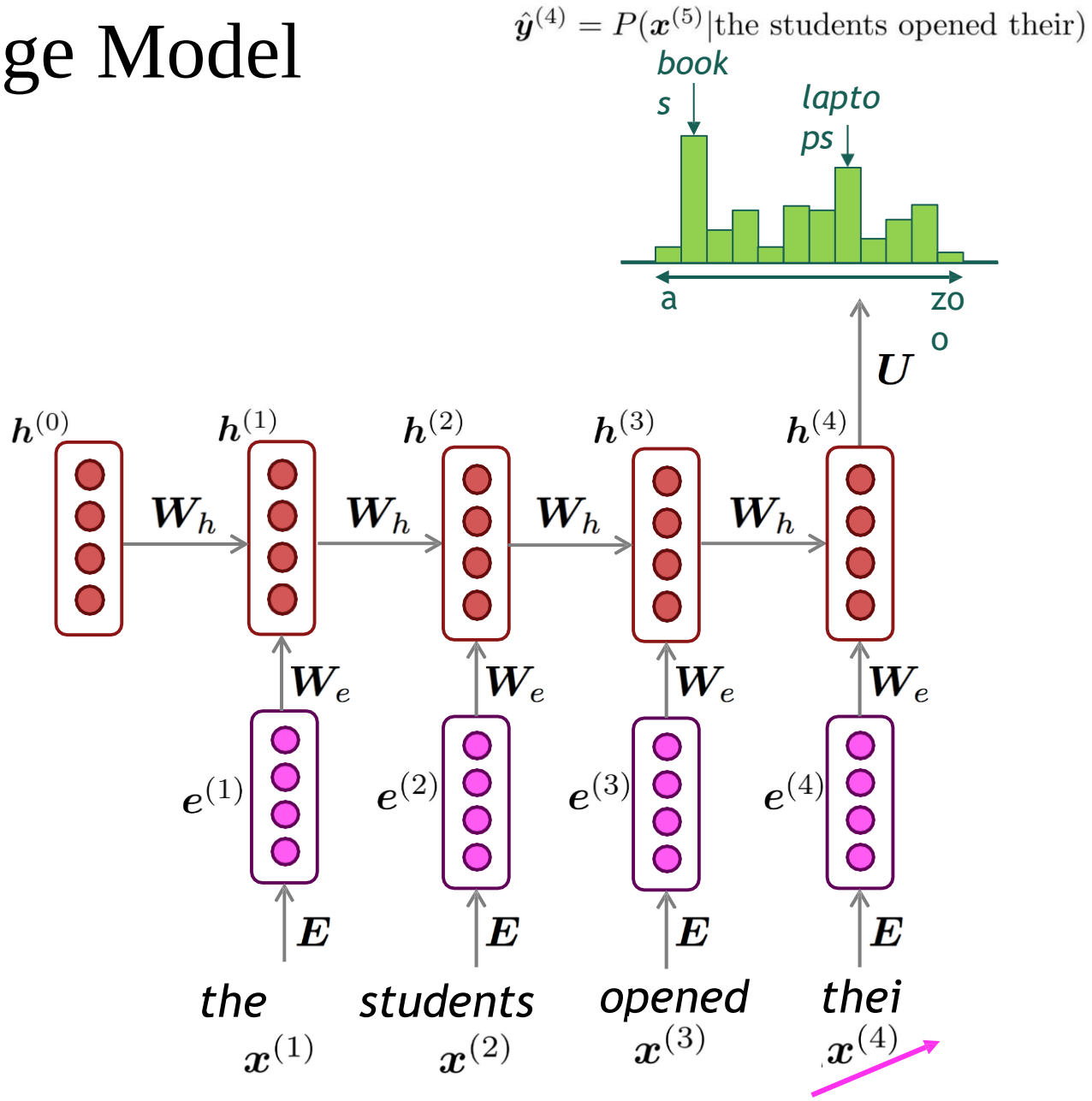
$\boldsymbol{h}^{(0)}$ is the initial hidden state

word embeddings

$$\boldsymbol{e}^{(t)} = \boldsymbol{E}\boldsymbol{x}^{(t)}$$

words / one-hot vectors

$$\boldsymbol{x}^{(t)} \in \mathbb{R}^{|V|}$$



***Note**: this input sequence could be much longer now!*

RNN Language Models

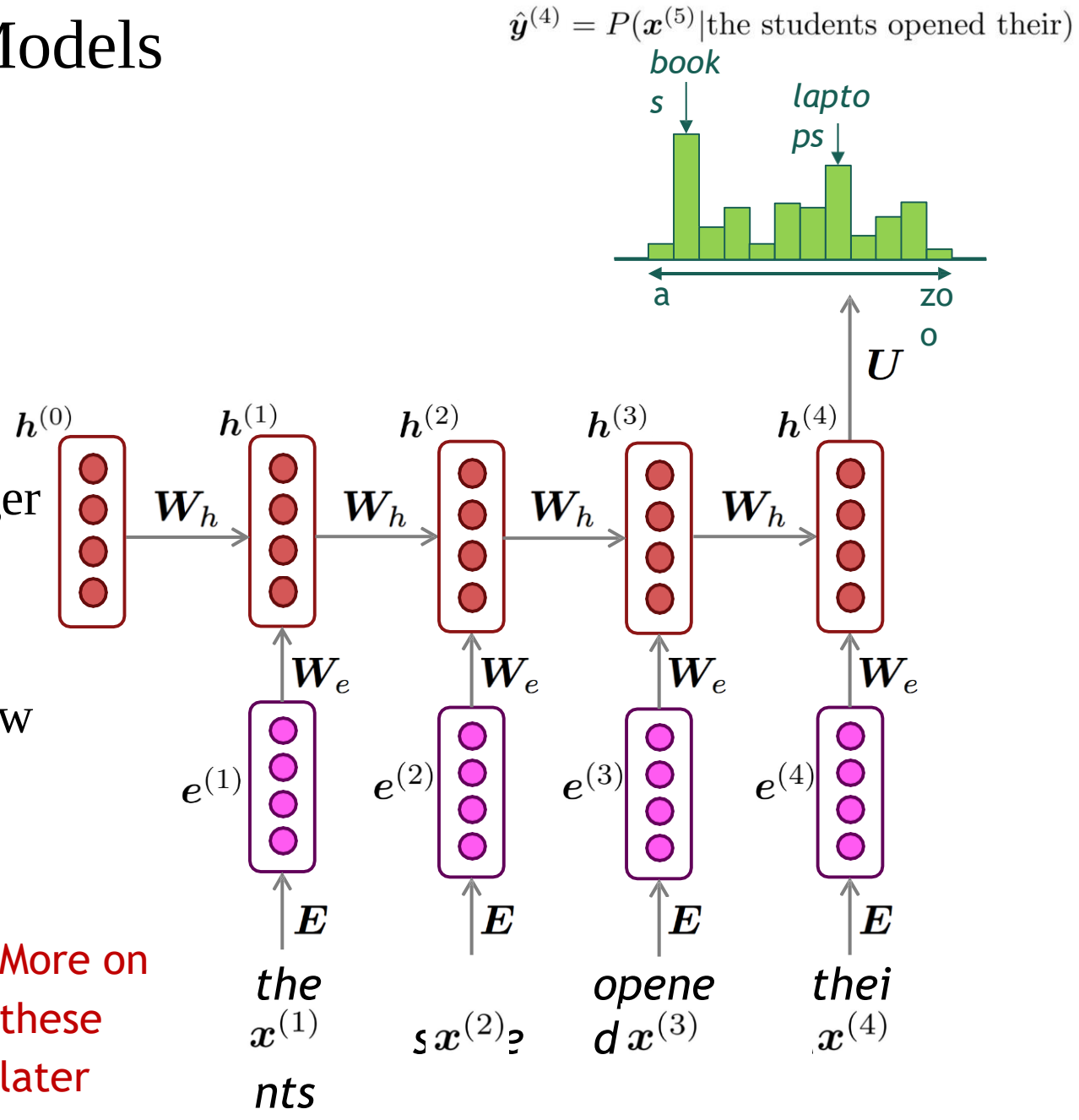
RNN **Advantages**:

- Can process any length input
- Computation for step *t* can (in theory) use information from many steps back
- Model size doesn't increase for longer input context
- Same weights applied on every timestep, so there is symmetry in how inputs are processed.

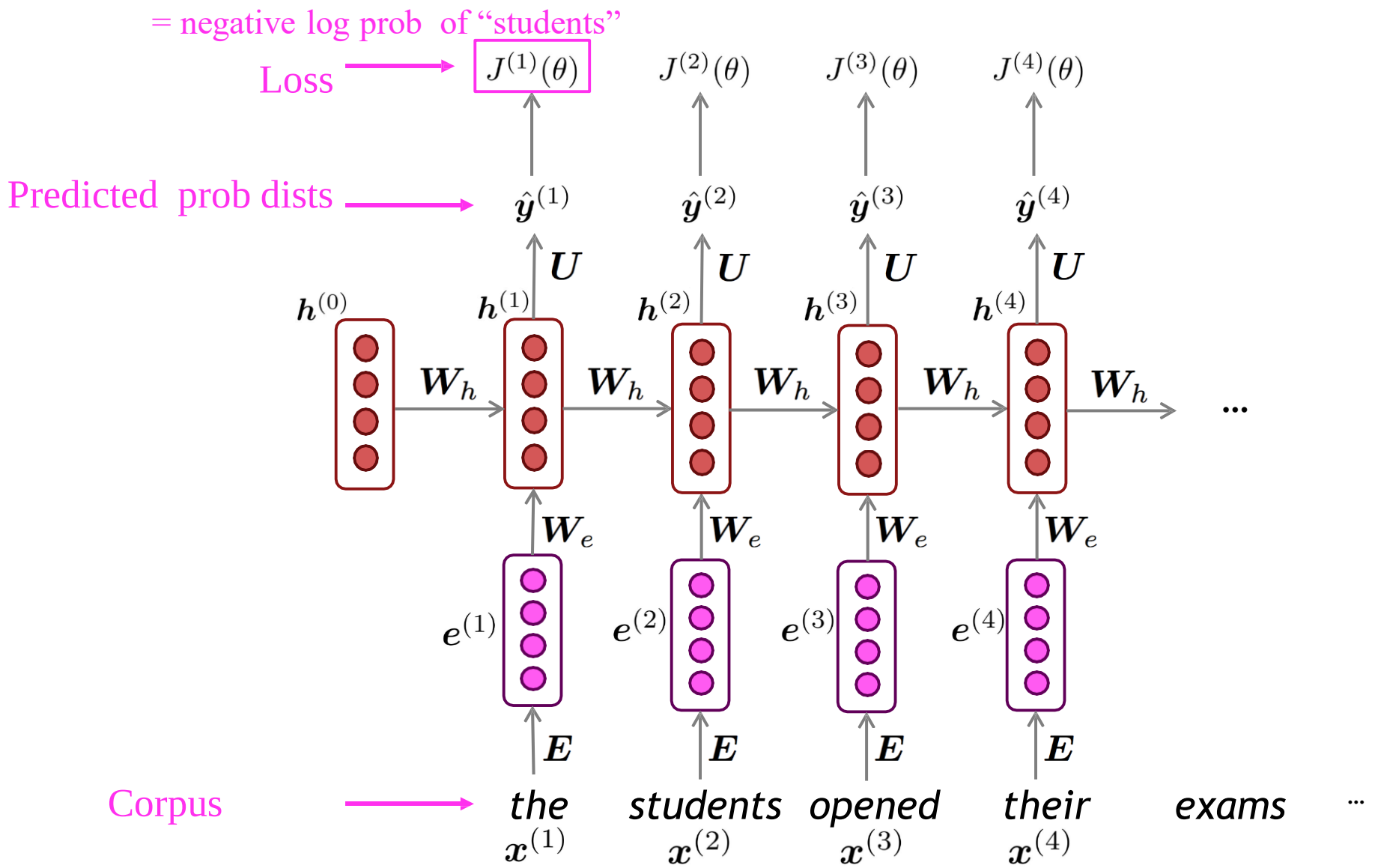
RNN **Disadvantages**:

- Recurrent computation is slow
- In practice, difficult to access information from many steps back

More on these later

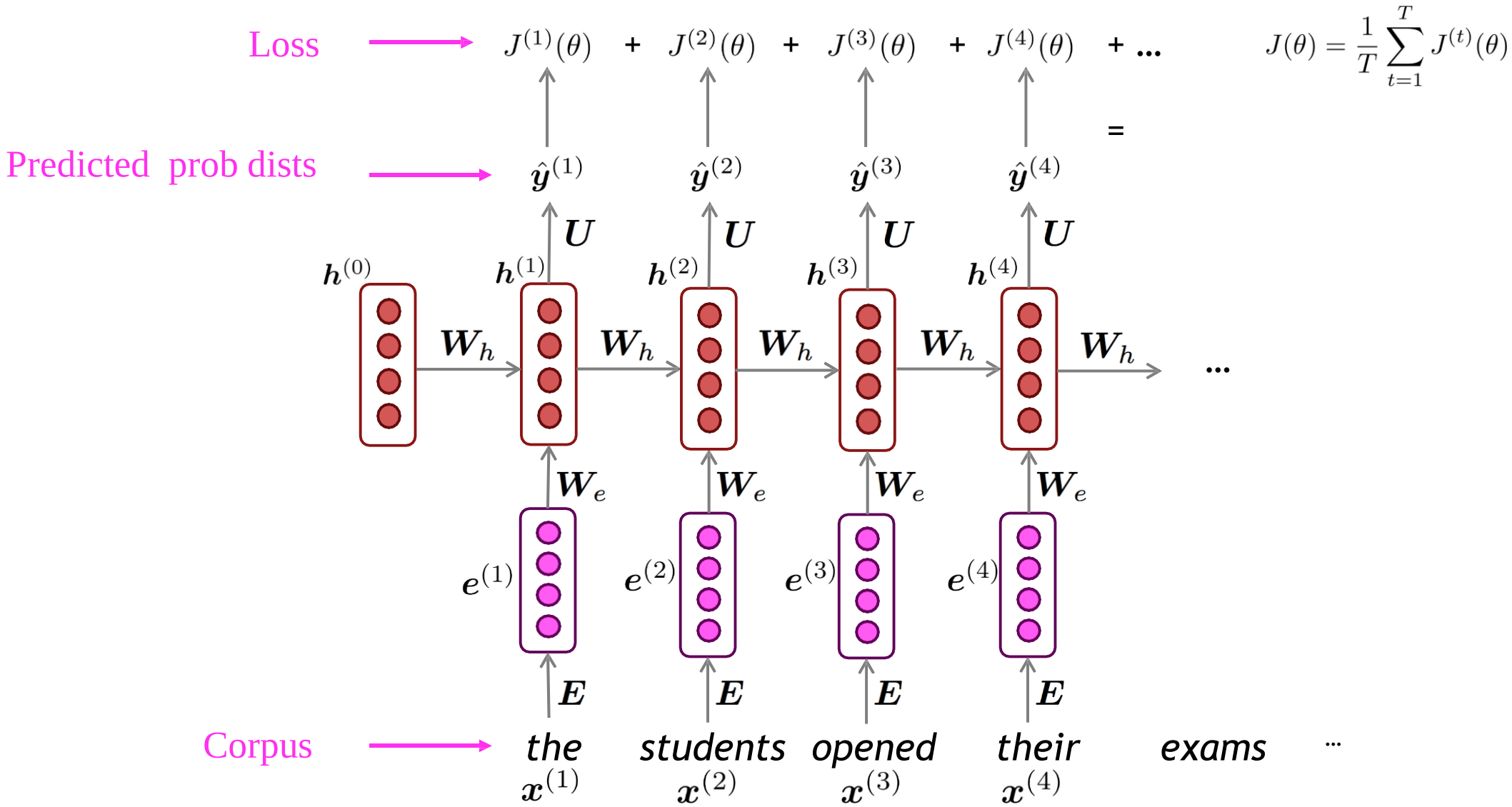


Training an RNN Language Model

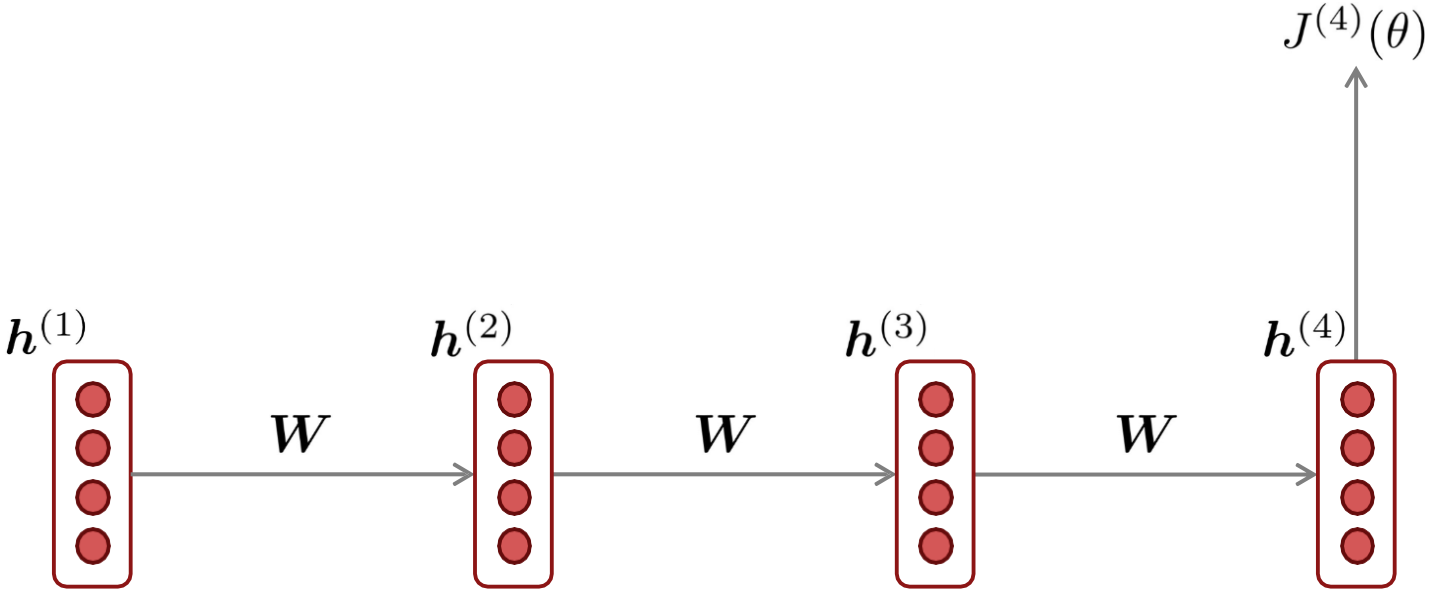


Training an RNN Language Model

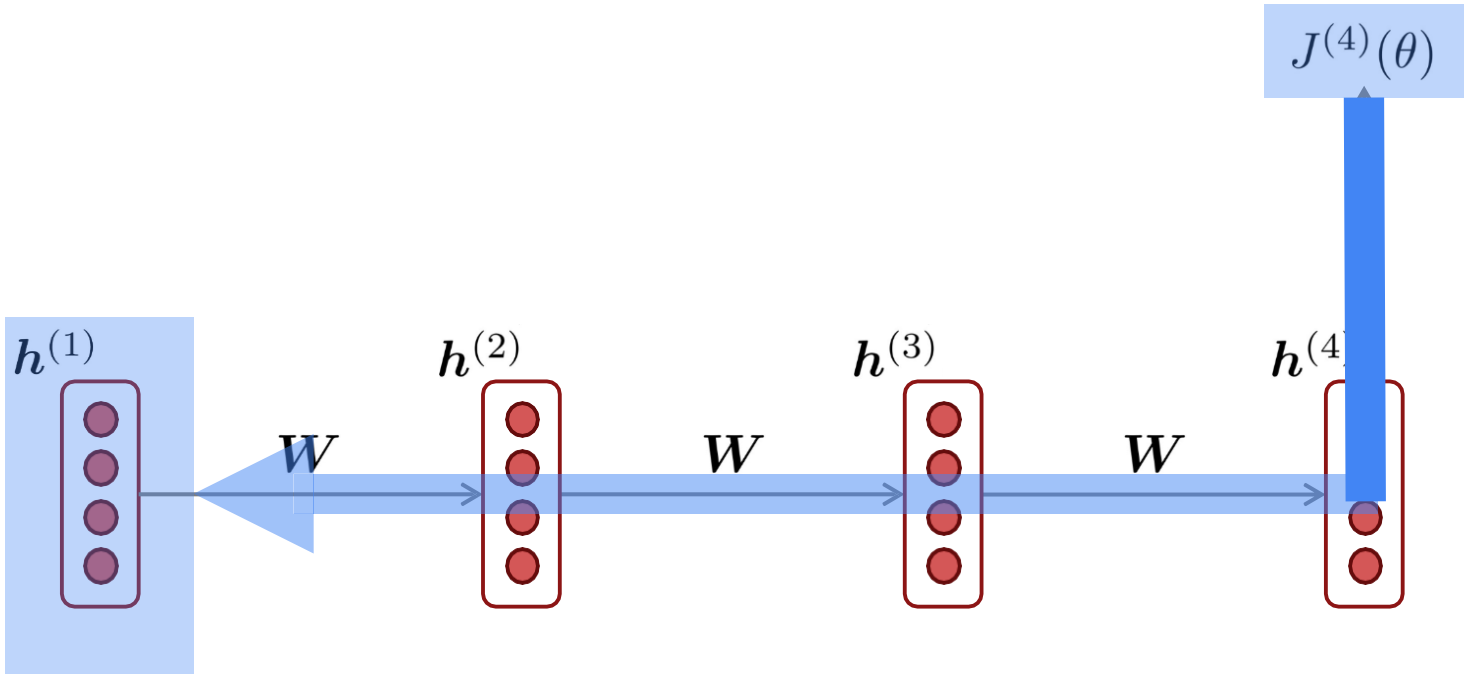
"Teacher forcing"



Problems with RNNs: Vanishing and Exploding Gradients

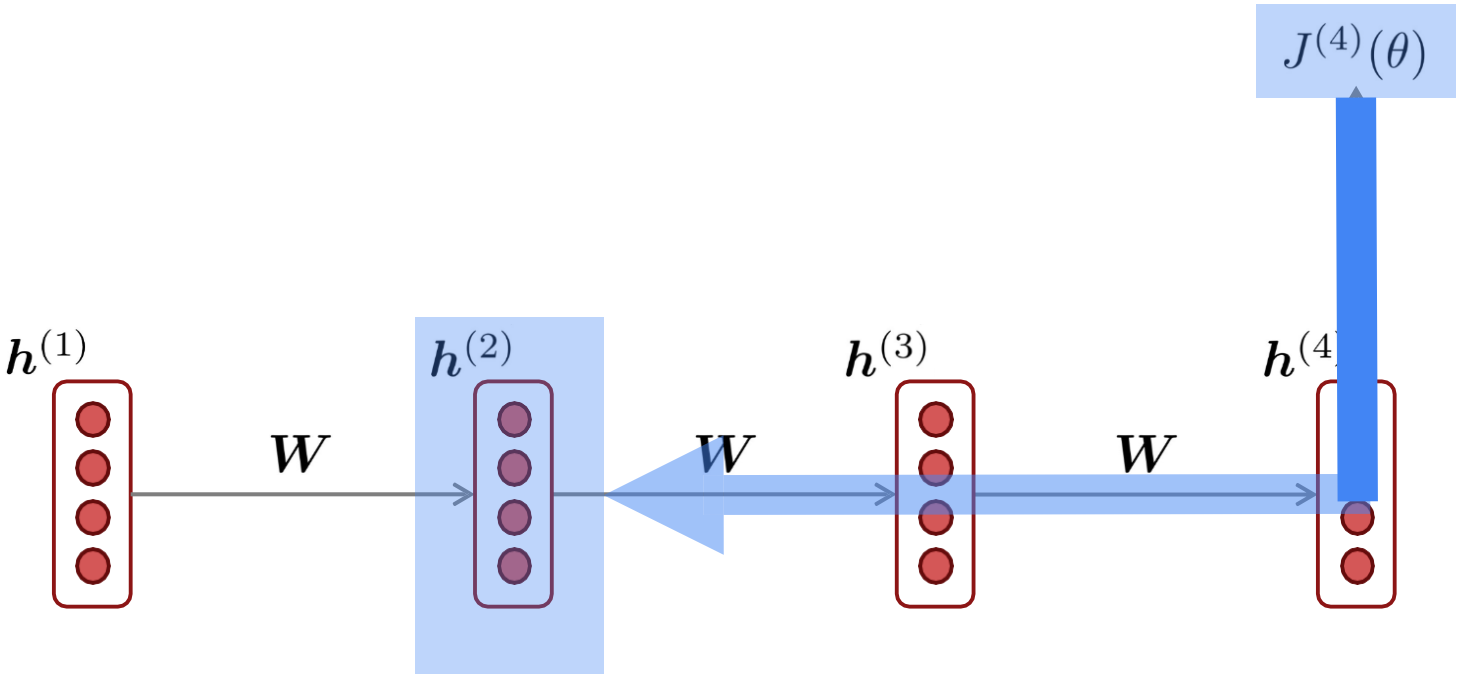


Vanishing gradient intuition



$$\frac{\partial J^{(4)}}{\partial \boldsymbol{h}^{(1)}} = ?$$

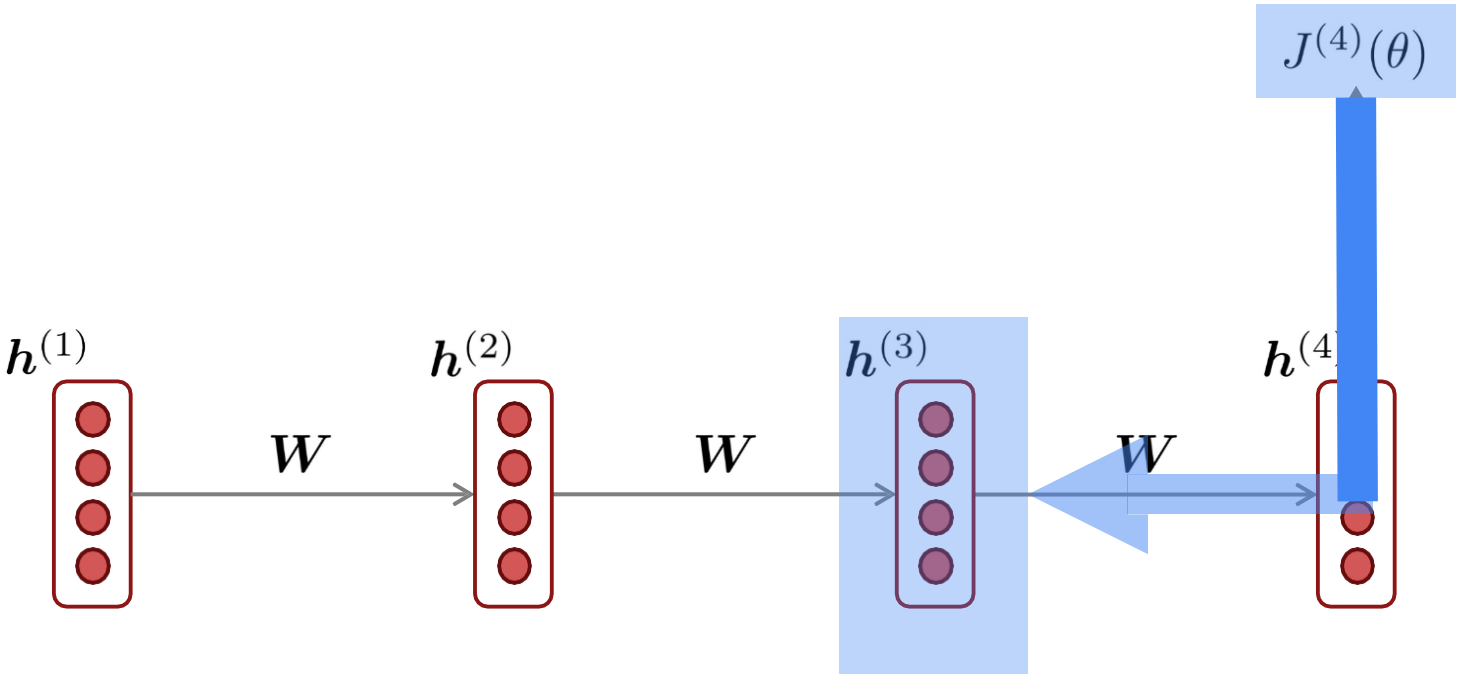
Vanishing gradient intuition



$$\frac{\partial J^{(4)}}{\partial \boldsymbol{h}^{(1)}} = \frac{\partial \boldsymbol{h}^{(2)}}{\partial \boldsymbol{h}^{(1)}} \times \frac{\partial J^{(4)}}{\partial \boldsymbol{h}^{(2)}}$$

chain rule!

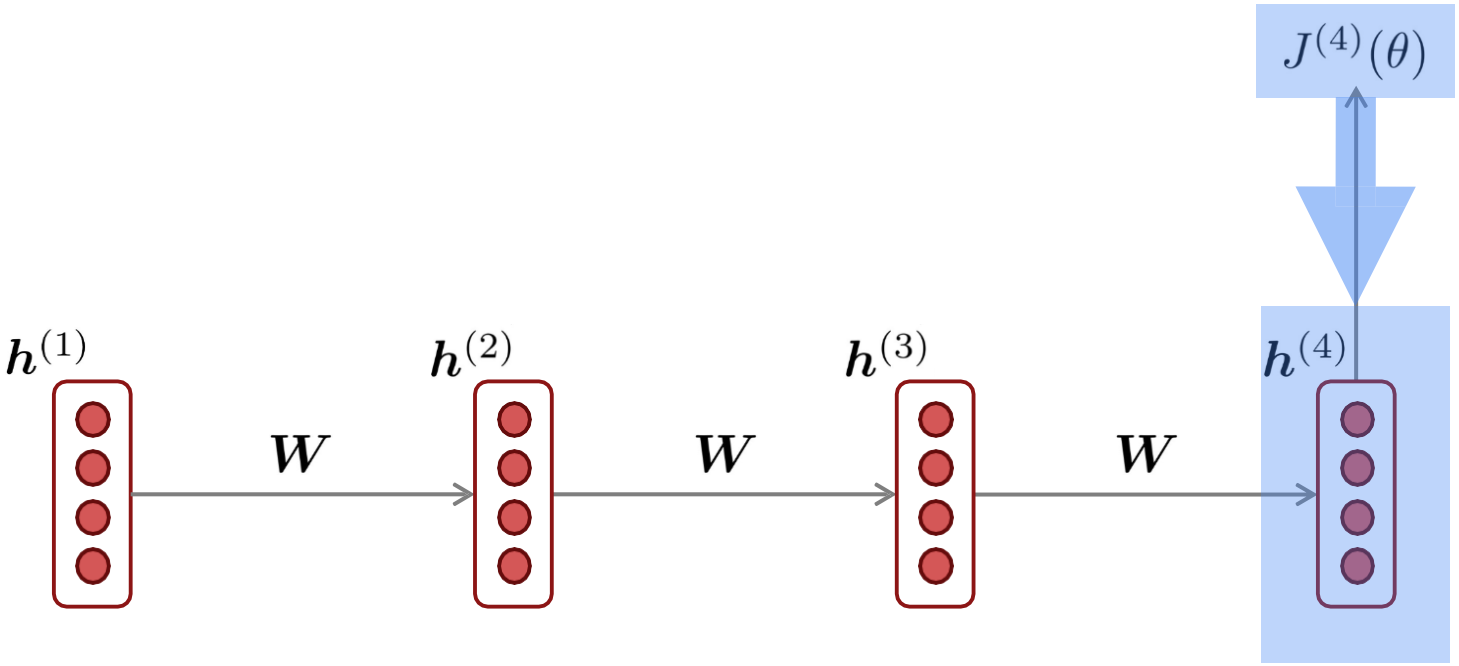
Vanishing gradient intuition



$$\frac{\partial J^{(4)}}{\partial \boldsymbol{h}^{(1)}} = \frac{\partial \boldsymbol{h}^{(2)}}{\partial \boldsymbol{h}^{(1)}} \times \frac{\partial \boldsymbol{h}^{(3)}}{\partial \boldsymbol{h}^{(2)}} \times \frac{\partial J^{(4)}}{\partial \boldsymbol{h}^{(3)}}$$

chain rule!

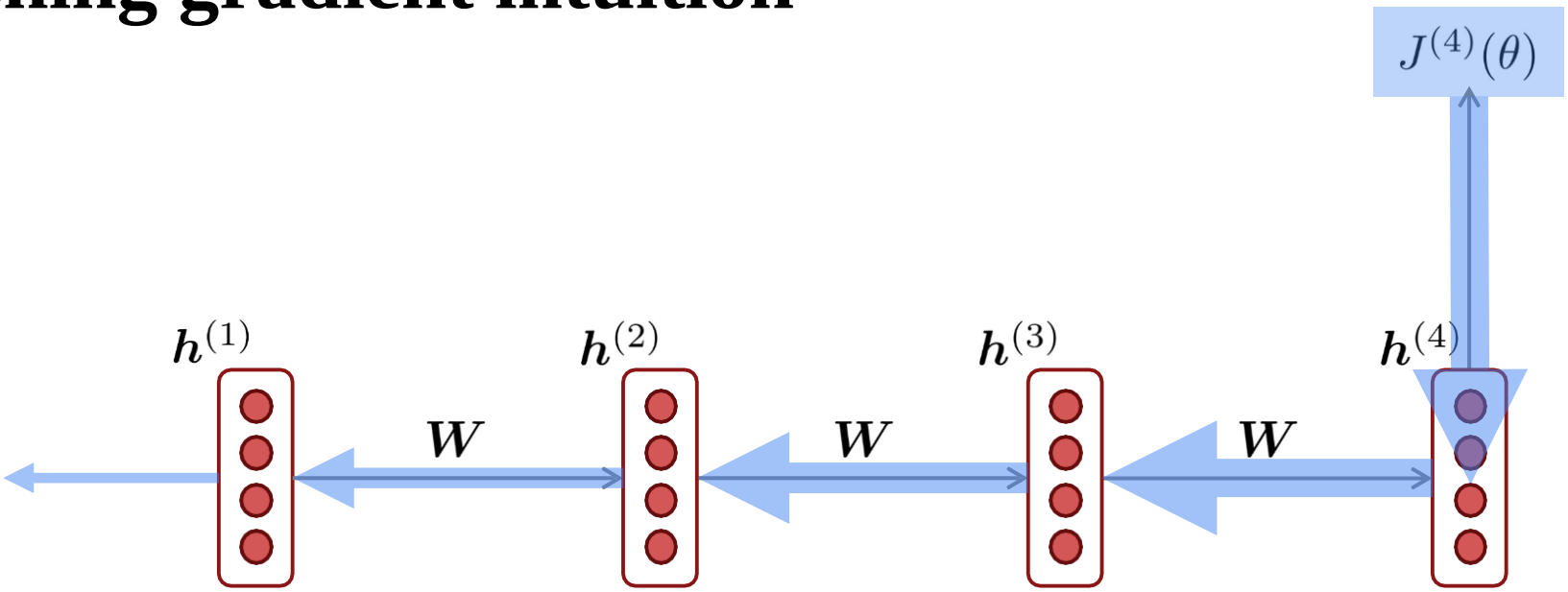
Vanishing gradient intuition



$$\frac{\partial J^{(4)}}{\partial \boldsymbol{h}^{(1)}} = \frac{\partial \boldsymbol{h}^{(2)}}{\partial \boldsymbol{h}^{(1)}} \times \frac{\partial \boldsymbol{h}^{(3)}}{\partial \boldsymbol{h}^{(2)}} \times \frac{\partial \boldsymbol{h}^{(4)}}{\partial \boldsymbol{h}^{(3)}} \times \frac{\partial J^{(4)}}{\partial \boldsymbol{h}^{(4)}}$$

chain rule!

Vanishing gradient intuition

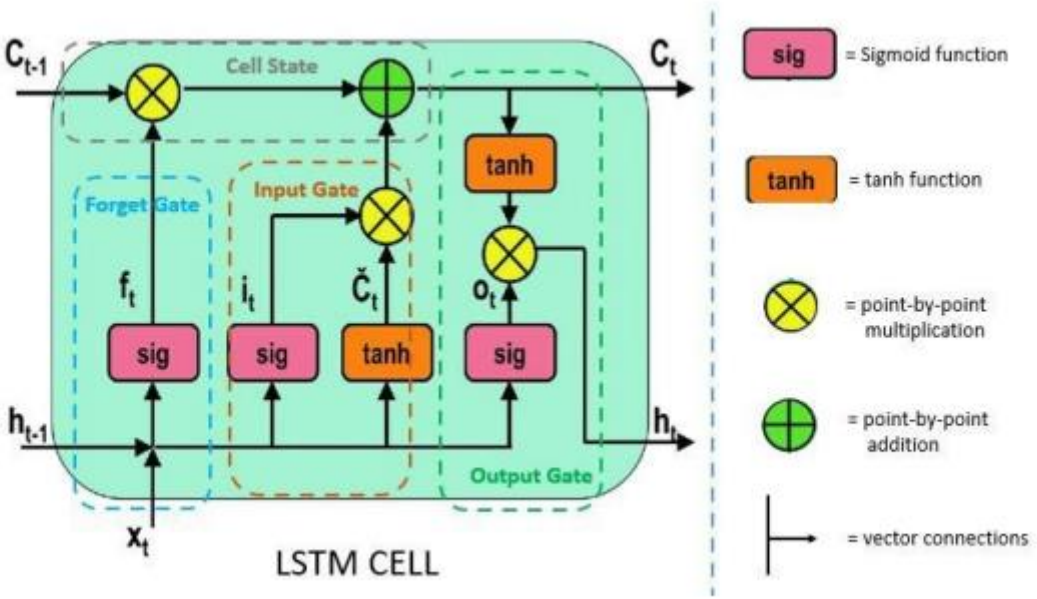


$$\frac{\partial J^{(4)}}{\partial h^{(1)}} = \frac{\partial h^{(2)}}{\partial h^{(1)}} \times \frac{\partial h^{(3)}}{\partial h^{(2)}} \times \frac{\partial h^{(4)}}{\partial h^{(3)}} \times \frac{\partial J^{(4)}}{\partial h^{(4)}}$$

What happens if these are small?

Vanishing gradient problem:
When these are small, the gradient signal gets smaller and smaller as it backpropagates further

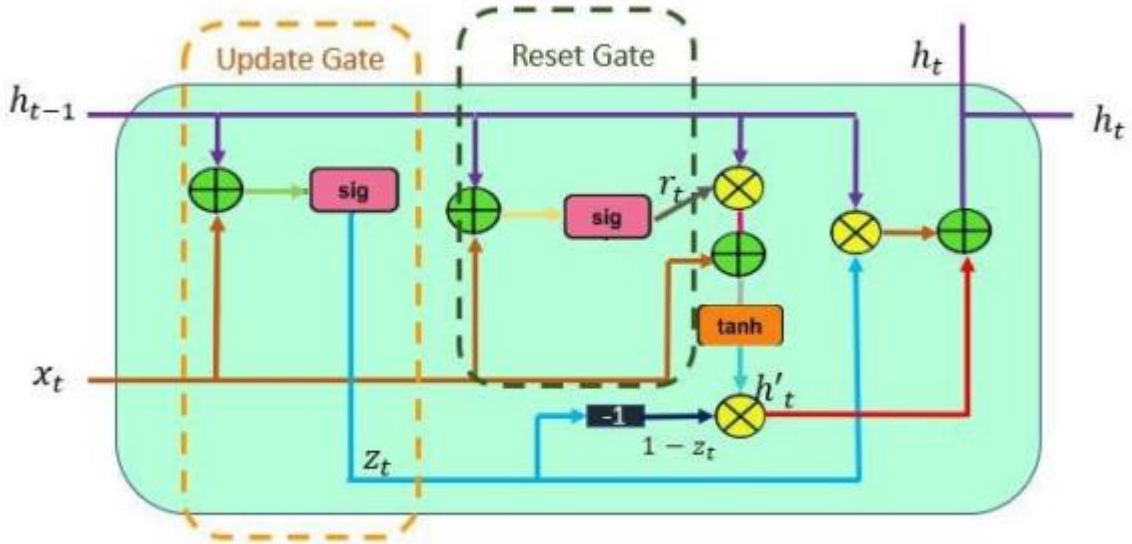
Special case of RNN - LSTM



Forget Gate: – This gate decides what information should be carried out forward or what information should be ignored. Information from previous hidden states and the current state information passes through the sigmoid function. Values that come out from sigmoid are always between 0 and 1. – if the value is closer to 1 means information should proceed forward and if value closer to 0 means information should be ignored.
Input Gate: – After deciding the relevant information, the information goes to the input gate, Input gate passes the relevant information, and this leads to updating the cell states. simply saving updating the weight. Input gate adds the new relevant information to the existing information by updating cell states.
Output Gate: – After the information is passed through the input gate, now the output gate comes into play. – Output gate generates the next hidden states. and cell states are carried over the next time step.

https://mitu.co.in/wp-content/uploads/2022/12/8.-GRU-and-LSTM.pdf

Special case of RNN - GRU



GRU (Gated Recurrent Units) are similar to the LSTM networks. GRU is a kind of newer version of RNN. However, there are some differences between GRU and LSTM.

– GRU doesn't contain a cell state
– GRU uses its hidden states to transport information
– It Contains only 2 gates(Reset and Update Gate)
– GRU is faster than LSTM – GRU has lesser tensor's operation that makes it faster

Is vanishing/exploding gradient just an RNN problem?

- No! It can be a problem for all neural architectures (including feed-forward and convolutional), especially very deep ones.
  - Due to chain rule / choice of nonlinearity function, gradient can become vanishingly small as it backpropagates
  - Thus, lower layers are learned very slowly (i.e., are hard to train)
- Another solution: lots of new deep feedforward/convolutional architectures add more direct connections (thus allowing the gradient to flow)

For example:

- Residual connections aka "ResNet"
- Also known as skip-connections
- The identity connection preserves information by default
- This makes deep networks much easier to train

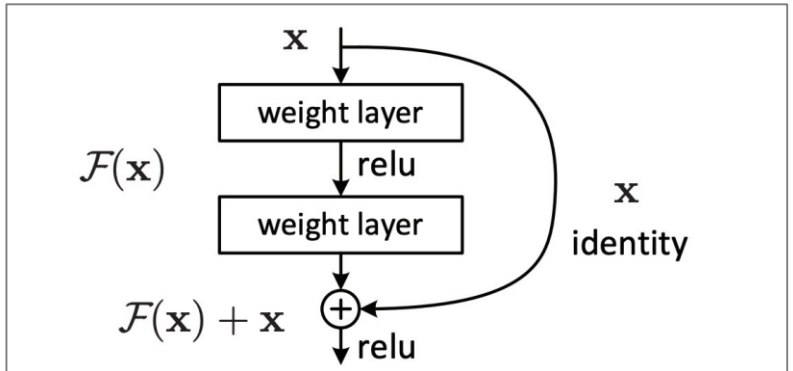


Figure 2. Residual learning: a building block.

"Deep Residual Learning for Image Recognition", He et al, 2015.
https://arxiv.org/pdf/1512.03385.pdf

CNN vs. RNN

			performance	lr	hidden	batch	sentLen	filter_size	margin
TextC	SentiC (acc)	CNN	82.38	0.2	20	5	60	3	–
		GRU	**86.32**	0.1	30	50	60	–	–
		LSTM	84.51	0.2	20	40	60	–	–
	RC (F1)	CNN	68.02	0.12	70	10	20	3	–
		GRU	**68.56**	0.12	80	100	20	–	–
		LSTM	66.45	0.1	80	20	20	–	–
SemMatch	TE (acc)	CNN	77.13	0.1	70	50	50	3	–
		GRU	**78.78**	0.1	50	80	65	–	–
		LSTM	77.85	0.1	80	50	50	–	–
	AS (MAP & MRR)	CNN	**(63.69,65.01)**	0.01	30	60	40	3	0.3
		GRU	(62.58,63.59)	0.1	80	150	40	–	0.3
		LSTM	(62.00,63.26)	0.1	60	150	45	–	0.1
	QRM (acc)	CNN	**71.50**	0.125	400	50	17	5	0.01
		GRU	69.80	1.0	400	50	17	-	0.01
		LSTM	71.44	1.0	200	50	17	-	0.01
SeqOrder	PQA (hit@10)	CNN	54.42	0.01	250	50	5	3	0.4
		GRU	**55.67**	0.1	250	50	5	–	0.3
		LSTM	55.39	0.1	300	50	5	–	0.3
ContextDep	POS tagging (acc)	CNN	94.18	0.1	100	10	60	5	–
		GRU	93.15	0.1	50	50	60	–	–
		LSTM	93.18	0.1	200	70	60	–	–
		Bi-GRU	94.26	0.1	50	50	60	–	–
		Bi-LSTM	**94.35**	0.1	150	5	60	–	–

Table 1: Best results or CNN, GRU and LSTM in NLP tasks

https://arxiv.org/pdf/1702.01923

Lessons for RNN/CNNs in NLP (before Transformer)

- Multi-task?
- Without pre-training, highly specialized;
- Cannot beats Rule-based systems, too much.
- More discrimination tasks, less generative.

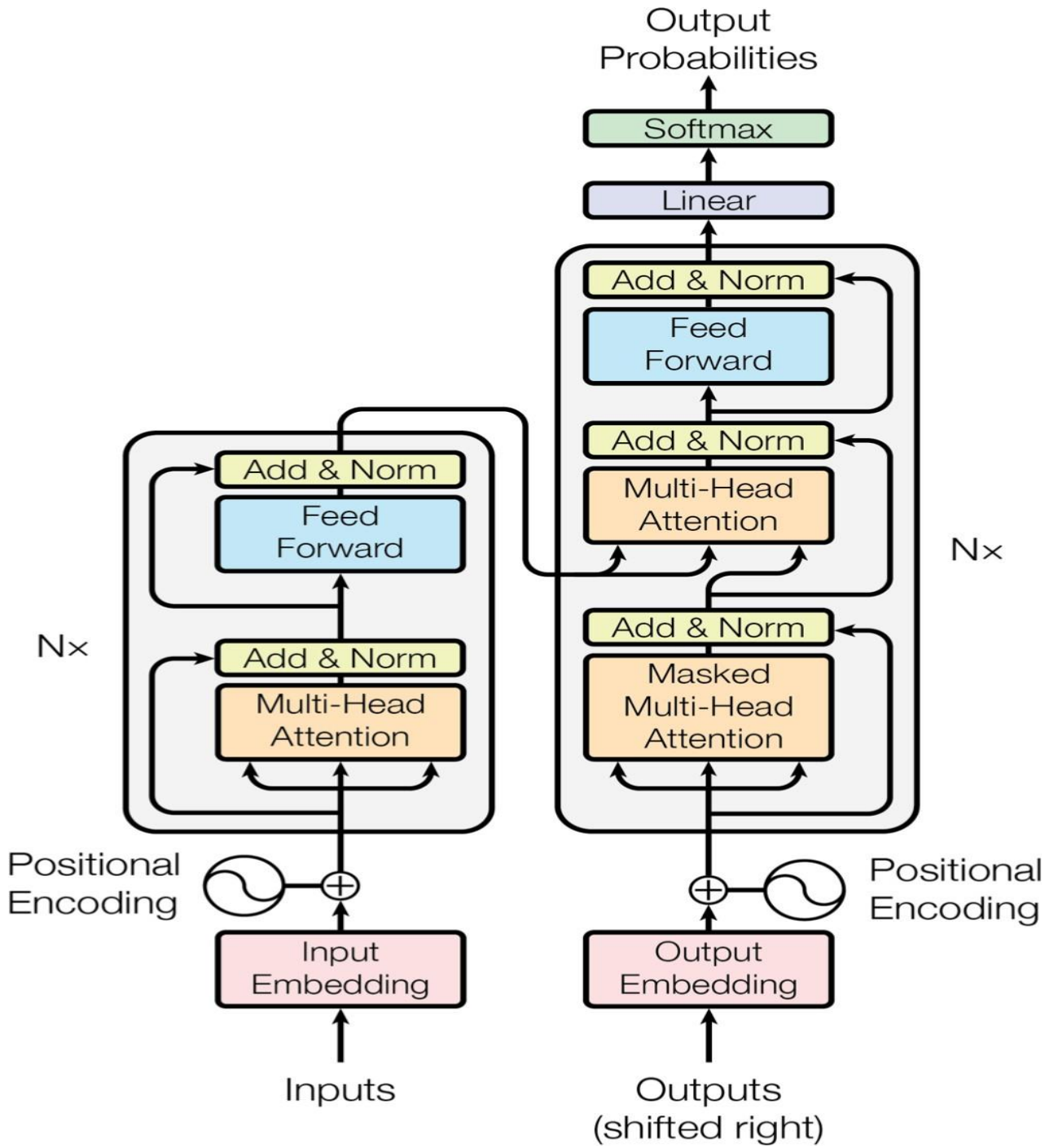
Transformer

- **Encoder**
- **Decoder**
- **Self-attention**
- **Multi-head self-attention**
- **Positional Encoding**

Today’s lecture

- MLP
  +: Strongest inductive bias: if all words are concated
  +: Weakest inductive bias: if all words are averaged
  - : The interaction at the token-level is too weak
- CNN & RNN
  +: The interaction at the token-level is slightly better.
  CNN: Bringing the global token-level interaction to the window-level
  - : Make simplifications, its global dependencies are limited
  RNN: An ideal method for processing token sequences
  - : Its recursive nature has the problem of disaster forgetting.
- **Transformer**
  +: Achieve **global dependence** at the **token-level** by **decoupling** token-level interaction and feature-level abstraction into two components, in **SAN** and **FNN**.

- Scaling law and emergent ability


Output Probabilities
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Nx
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Nx
Positional Encoding
Positional Encoding
Input Embedding
Output Embedding
Inputs
Outputs (shifted right)

Why transformer

Why Pretraining + Transformers

- 1.Because transformers are more efficient?

Transformers are shower comparing to LSTM with same amount parameters

Credits from 杨植麟, Recurrent AI

Why Pretraining + Transformers

- 1.Because transformers are more efficient?

Transformers are shower comparing to LSTM with same amount parameters

- 2. Because transformers are better on machine translation?

RNNs and CNNs are equally good in machine translations

Why Pretraining + Transformers

- 1.Because transformers are more efficient?

Transformers are shower comparing to LSTM with same amount parameters

- 2. Because transformers are better on machine translation?

RNNs and CNNs are equally good in machine translations

- 3. Because transformers use nothing but attention?

So what?

Why Pretraining + Transformers

- 1.Because transformers are more efficient?

Transformers are shower comparing to LSTM with same amount parameters

- 2. Because transformers are better on machine translation?

RNNs and CNNs are equally good in machine translations

- 3. Because transformers use nothing but attention?

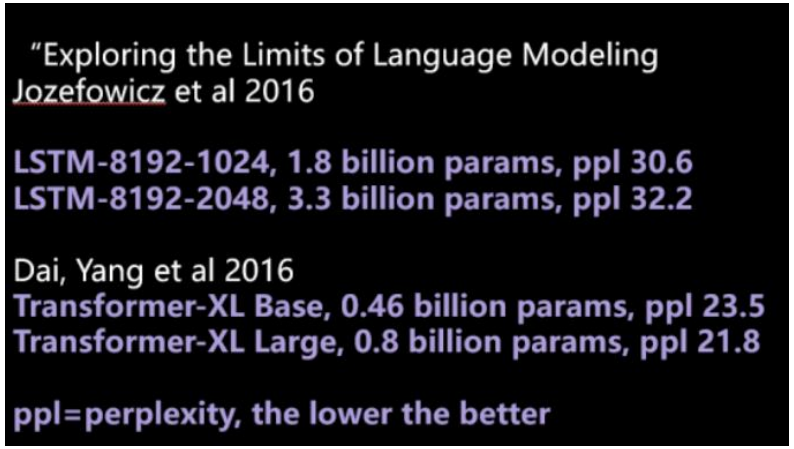
So what?

- 4. Because transformers learns contextualised word embeddings?

RNN also can learn contextualised word embeddings

Why Pretraining + Transformers

- **Capacity**: The model has sufficient expressive capabilities
- **Optimization**: Can optimize and obtain better solutions in a huge expression space
- **Generalization**: Better solutions can generalize on test data



Scalability: Transformers scale much better with more parameters