



Attention in Deep Learning

Alex Smola (smola@) and Aston Zhang (astonz@)

Amazon Web Services

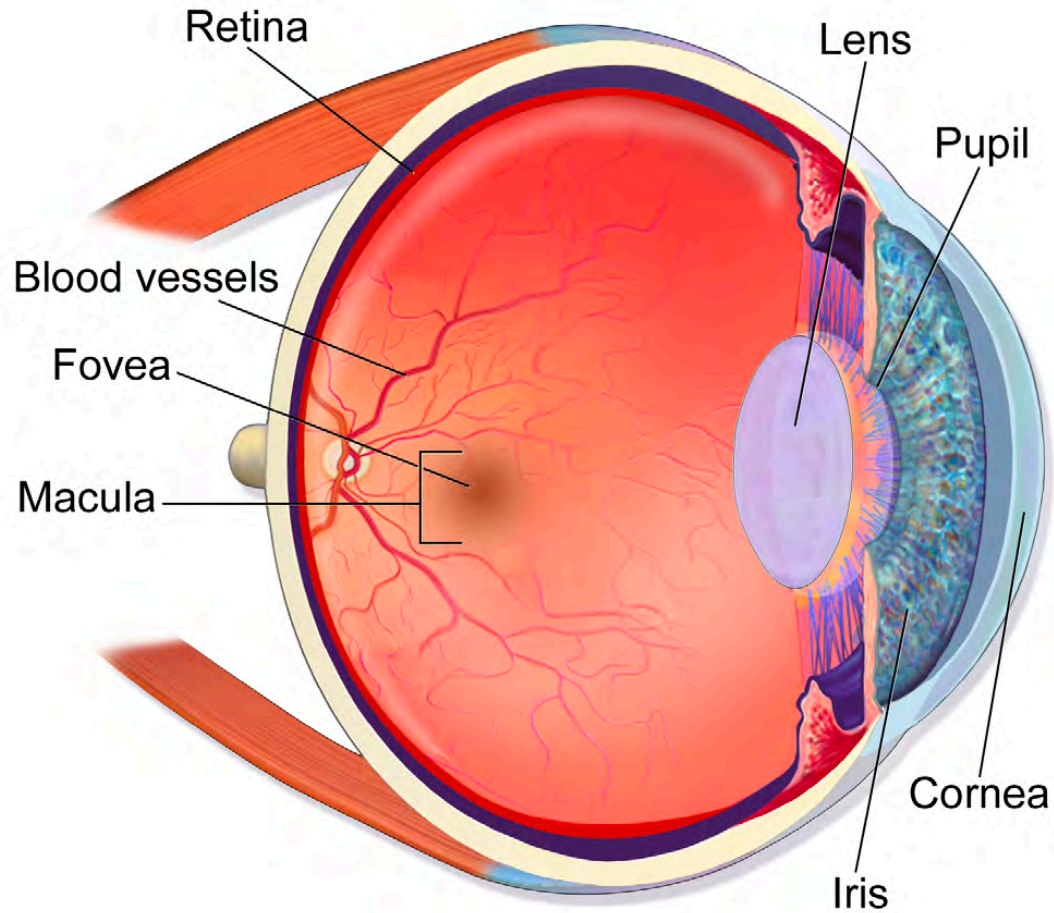
ICML 2019, Long Beach, CA

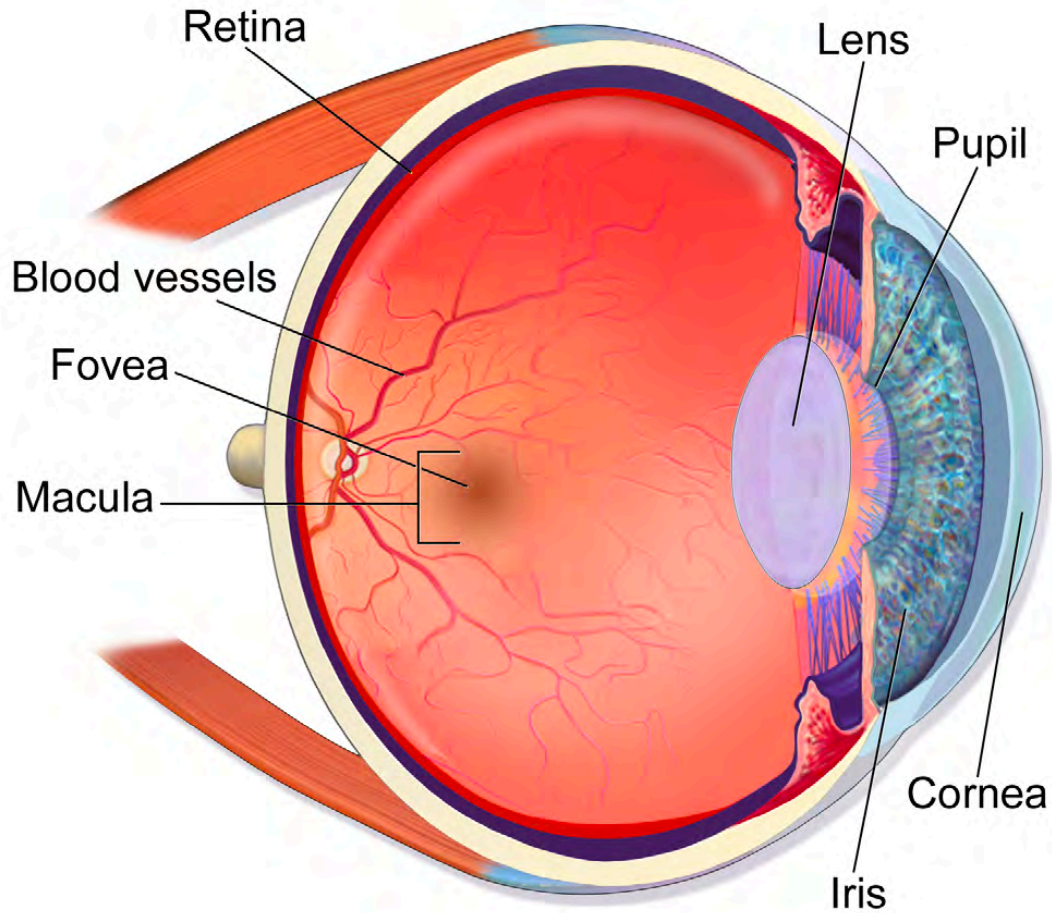
bit.ly/2R10hTu

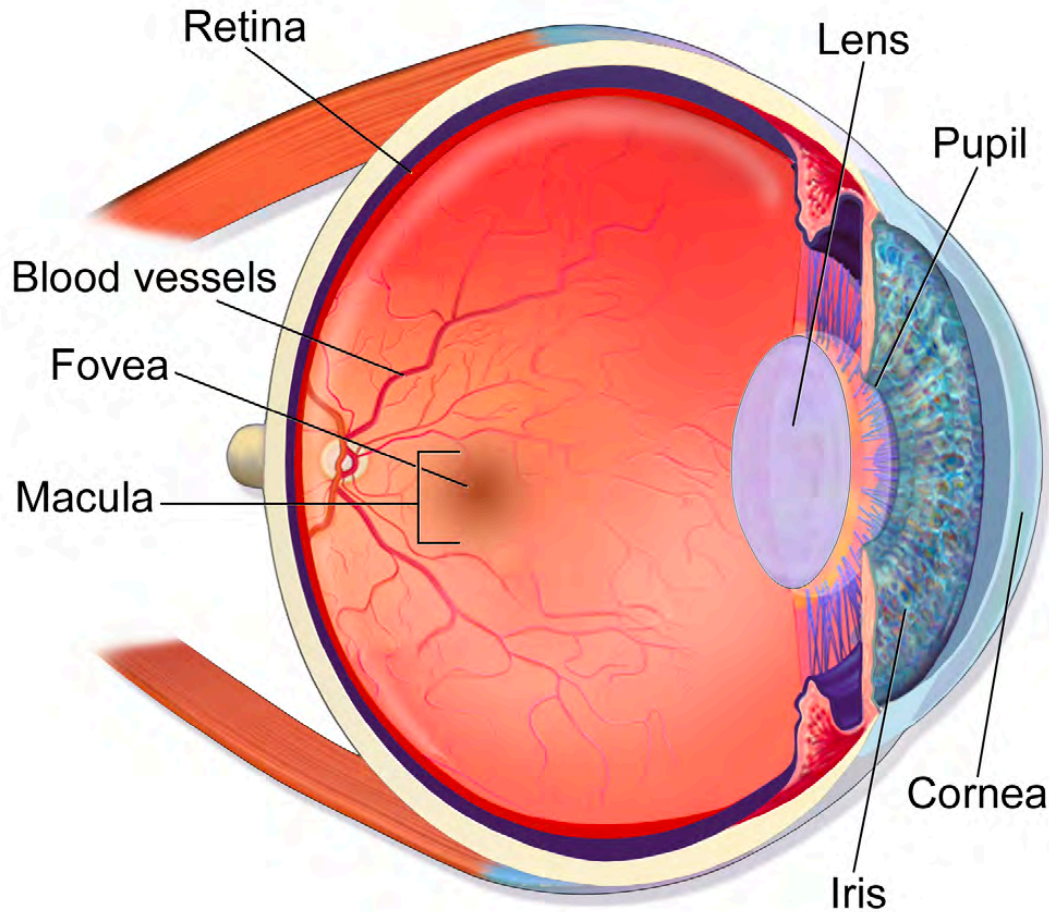
alex.smola.org/talks/ICML19-attention.key

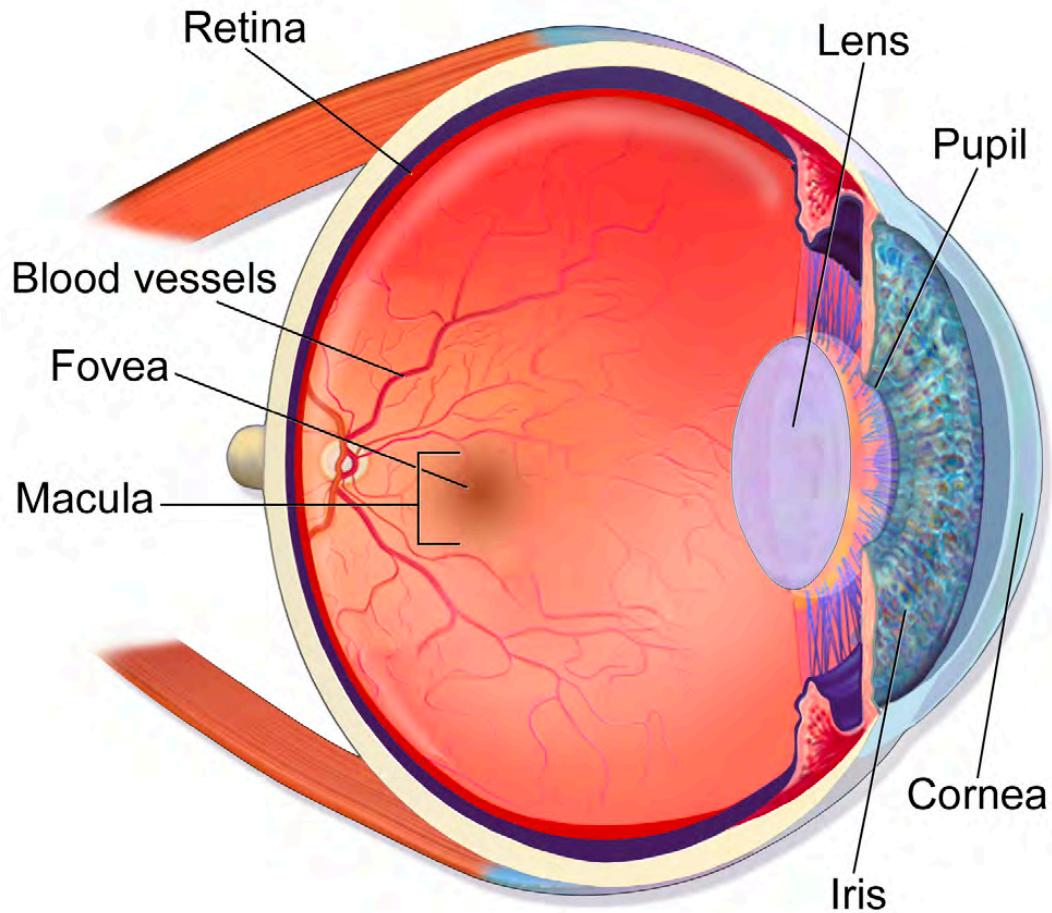
alex.smola.org/talks/ICML19-attention.pdf

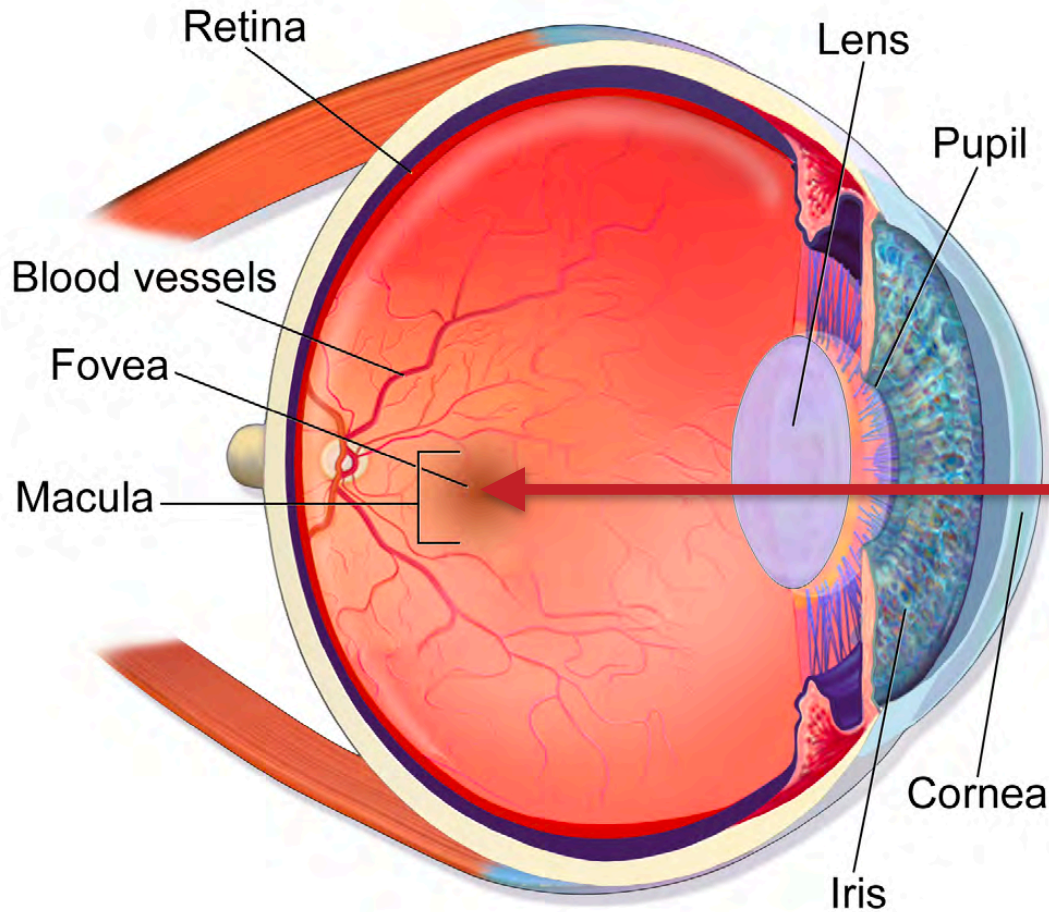


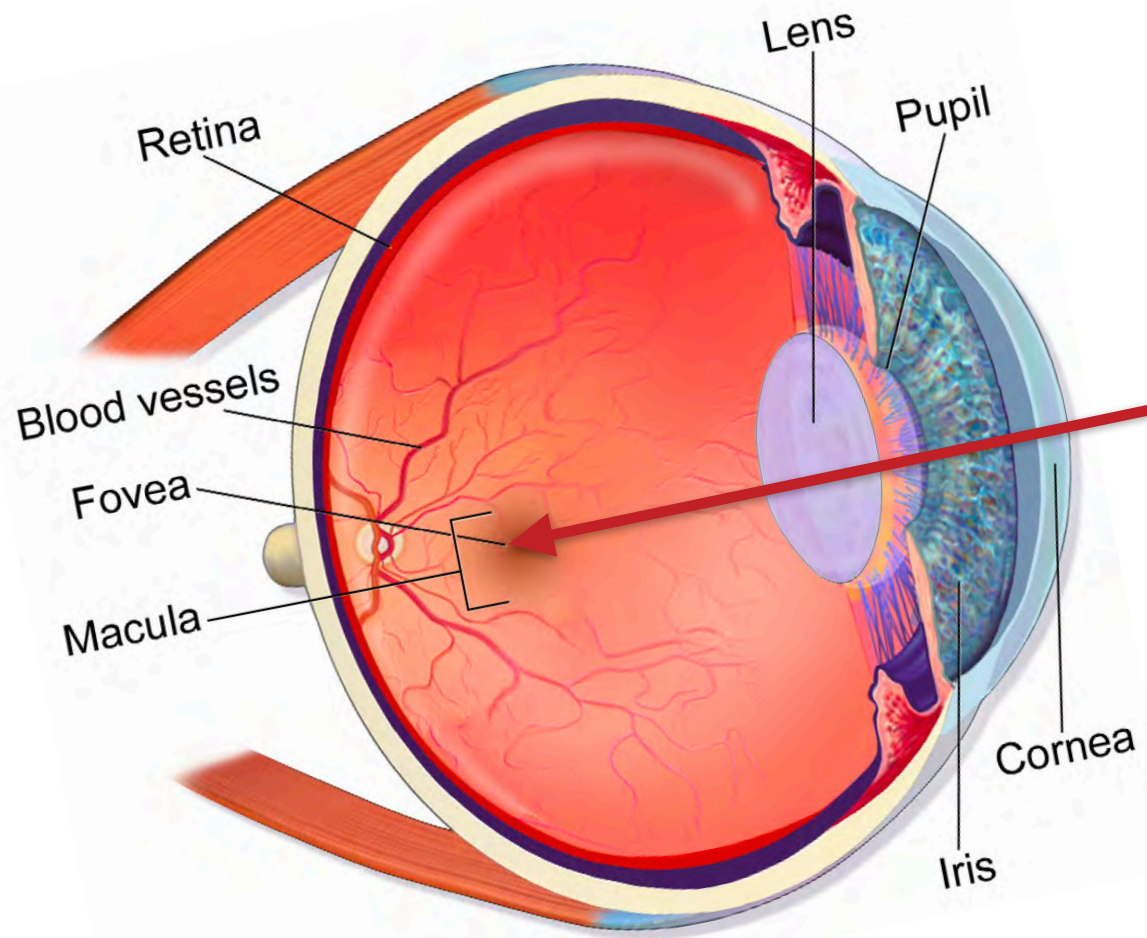






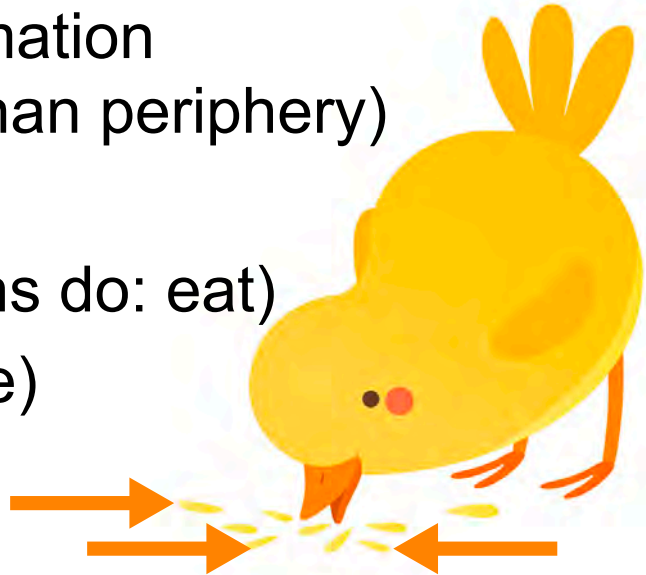






Attention in Animals

- **Resource saving**
 - Only need **sensors** where relevant bits are (e.g. fovea vs. peripheral vision)
 - Only **compute** relevant bits of information (e.g. fovea has many more 'pixels' than periphery)
- **Variable state manipulation**
 - Manipulate environment (for all grains do: eat)
 - Learn modular subroutines (not state)
- **In machine learning - nonparametric**



Outline

1. Watson Nadaraya Estimator

2. Pooling

- Single objects - Pooling to attention pooling
- Hierarchical structures - Hierarchical attention networks

3. Iterative Pooling

Question answering / memory networks

4. Iterative Pooling and Generation

Neural machine translation

5. Multiple Attention Heads

- Transformers / BERT
- Lightweight, structured, sparse

6. Resources

1. Watson Nadaraya Estimator '64

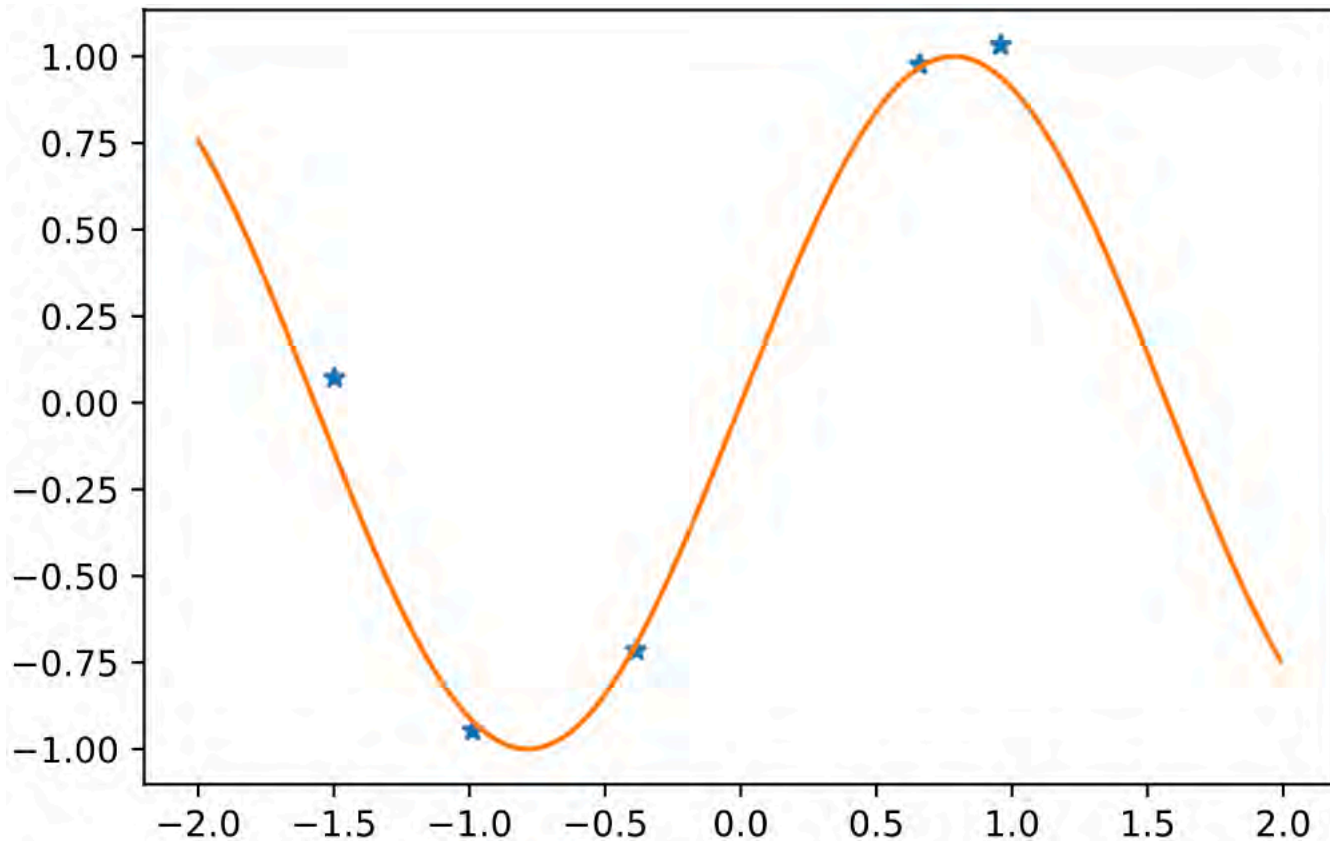


Geoffrey Watson



Elizbar Nadaraya

Regression Problem



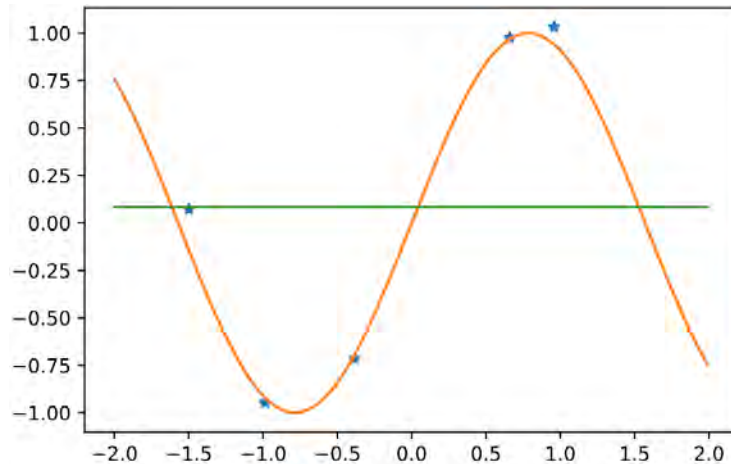
Solving the regression problem

- Data $\{x_1, \dots, x_m\}$ and labels $\{y_1, \dots, y_m\}$
- Estimate label y at new location x
- **The world's dumbest estimator**
Average over all labels

$$y = \frac{1}{m} \sum_{i=1}^m y_i$$

- **Better idea (Watson, Nadaraya, 1964)**
Weigh the labels according to location

$$y = \sum_{i=1}^m \alpha(x, x_i) y_i$$



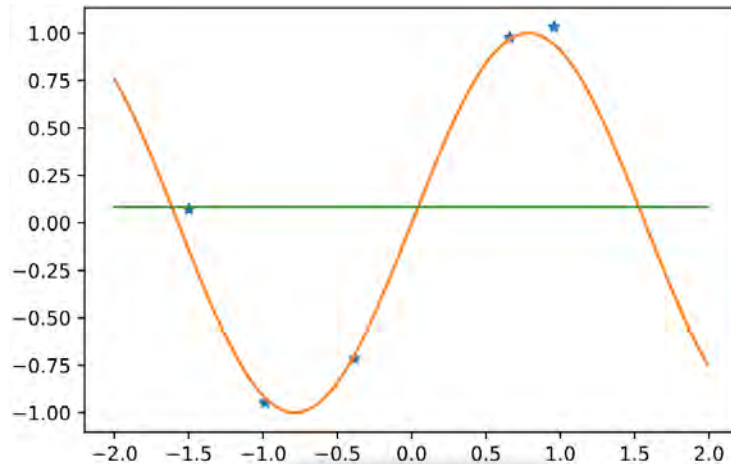
Solving the regression problem

- Data $\{x_1, \dots, x_m\}$ and labels $\{y_1, \dots, y_m\}$
- Estimate label y at new location x
- **The world's dumbest estimator**
Average over all labels

$$y = \frac{1}{m} \sum_{i=1}^m y_i$$

- **Better idea (Watson, Nadaraya, 1964)**
Weigh the labels according to location

$$y = \sum_{i=1}^m \alpha(x, x_i) y_i$$



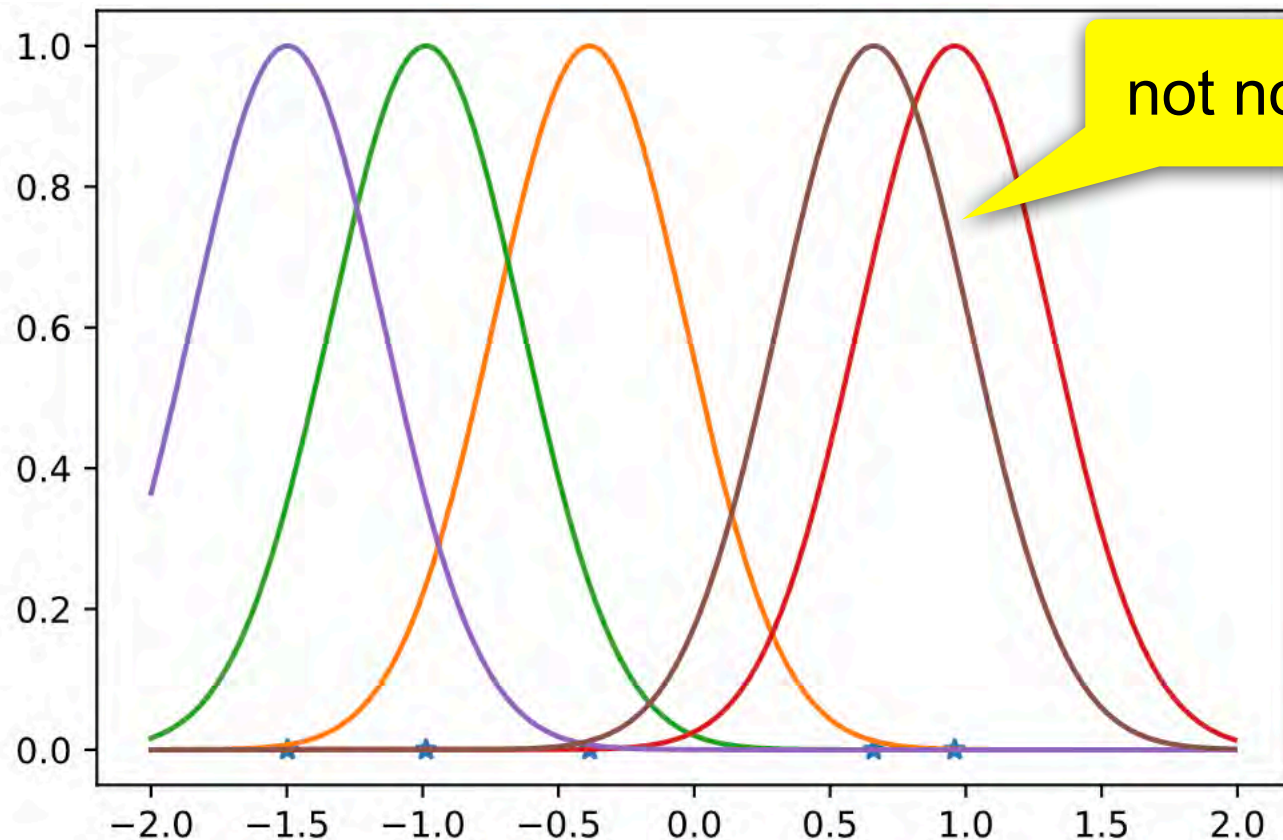
key

$\alpha(x, x_i) y_i$

query

value

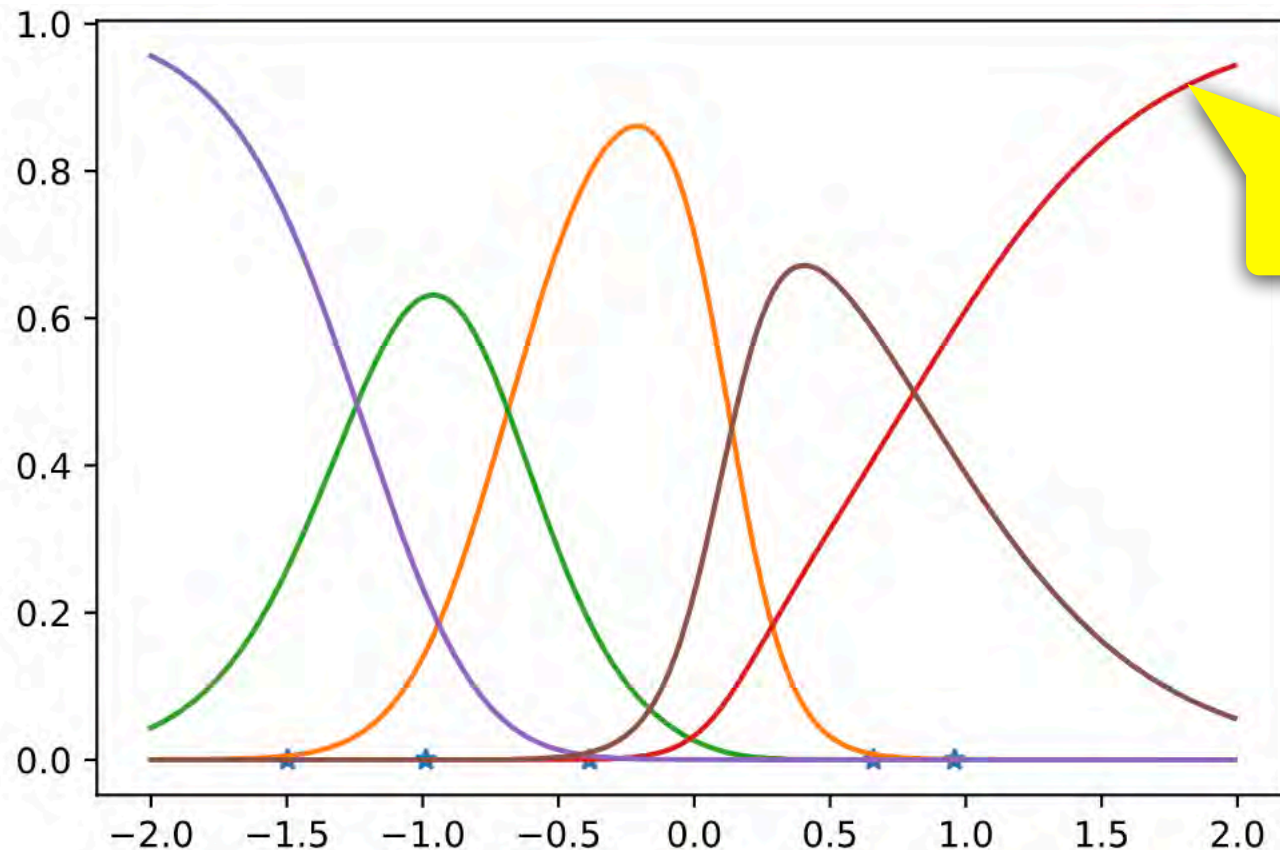
Weighing the locations (e.g. with Gaussians)



not normalized

$$\alpha(x, x_i) \propto k(x_i, x)$$

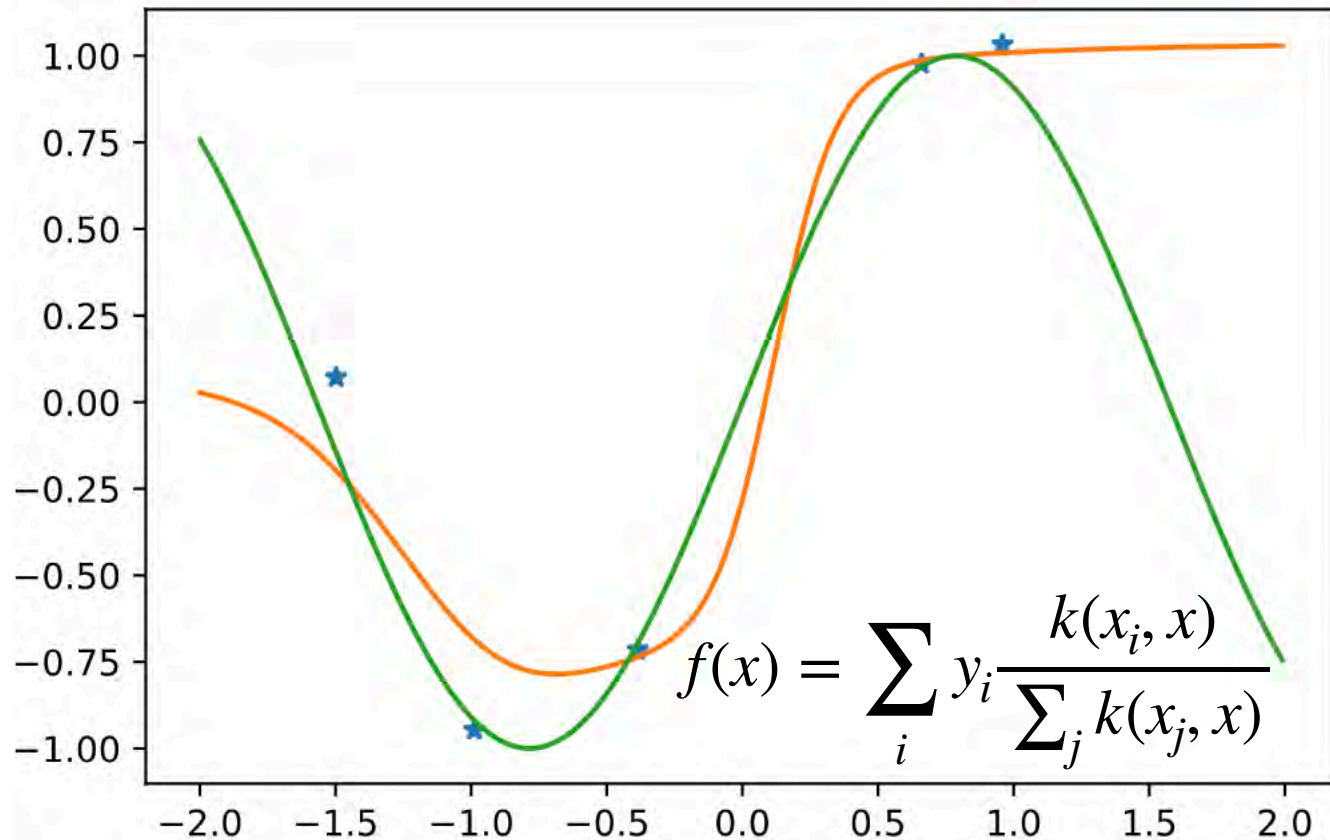
Weighing the locations (e.g. with Gaussians)



normalized

$$\alpha_i(x) = \frac{k(x_i, x)}{\sum_j k(x_j, x)}$$

Weighted regression estimate



Why bother with a 55 year old algorithm?

- **Consistency**

Given enough data this algorithm converges to the optimal solution (can your deep net do this?)

- **Simplicity**

No free parameters - information is in the data not weights (or very few if we try to learn the weighting function)

Why bother with a 55 year old algorithm?

- **Consistency**

Given enough data this algorithm converges to the optimal solution (can your deep net do this?)

- **Simplicity**

No free parameters - information is in the data not weights (or very few if we try to learn the weighting function)

- **Deep Learning Variant**

- Learn weighting function
- Replace averaging (pooling) by weighted pooling

An aerial view of a swimming pool with several lanes. The lanes are separated by dark blue lane lines and white lane lines. The water is a vibrant blue. The text "2. Pooling" is overlaid in the center of the image in a white, bold, sans-serif font.

2. Pooling

Deep Sets (Zaheer et al. 2017)

- Deep (Networks on) Sets $X = \{x_1, \dots, x_n\}$
 - Need permutation invariance for elements in set (e.g. LSTM doesn't work to ingest elements)
 - Theorem - all functions are of the form*

$$f(X) = \rho \left(\sum_{x \in X} \phi(x) \right)$$

*or some combination thereof

- Applications - point clouds, set extension, red shift for galaxies, text retrieval, tagging, etc.

Deep Sets (Zaheer et al. 2017)

Outliers in sets - learn function $f(X)$ on set such that

$$f(\{x\} \cup X) \geq f(\{x'\} \cup X) + \Delta(x, x')$$



Deep Sets with Attention aka Multi-Instance Learning (Ilse, Tomczak, Welling, '18)

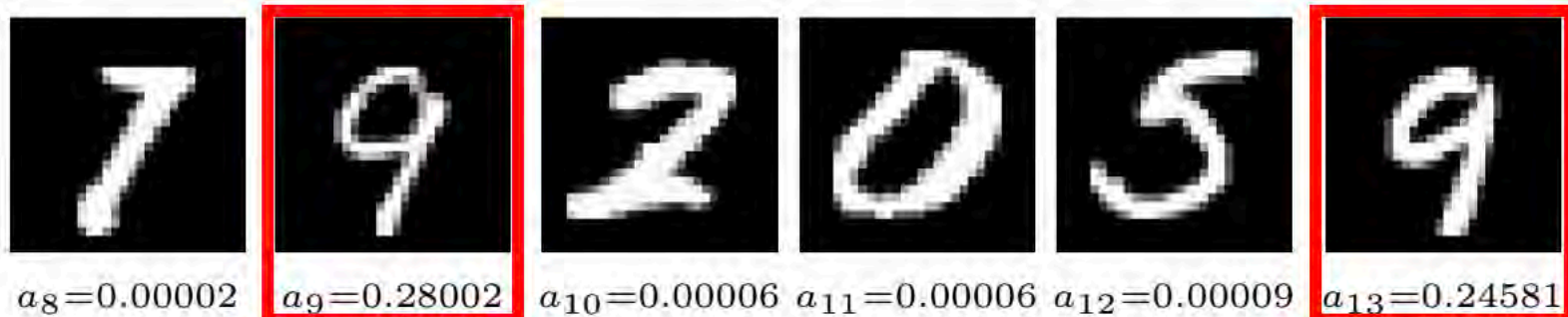
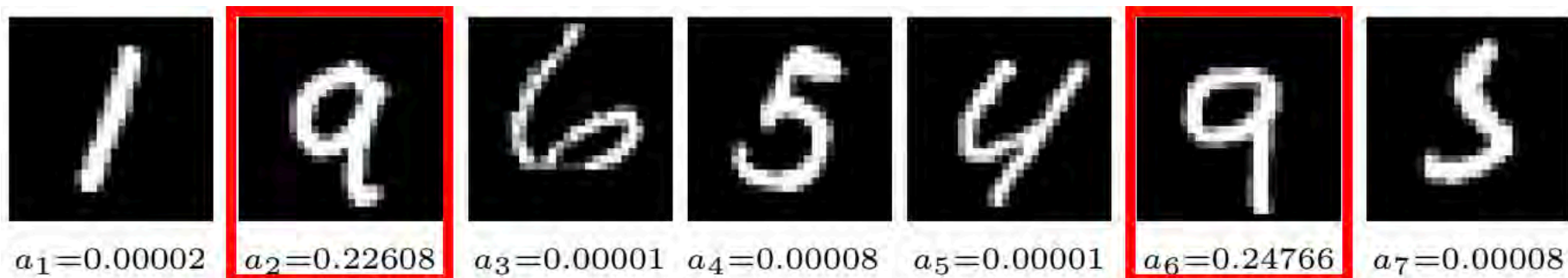
- Multiple Instance Problem
Set contains one (or more) elements with desirable property (drug discovery, keychain). Identify those sets.
- Deep Sets have trouble focusing, hence weigh it

$$f(X) = \rho \left(\sum_{x \in X} \phi(x) \right) \longrightarrow f(X) = \rho \left(\sum_{x \in X} \alpha(w, x) \phi(x) \right)$$

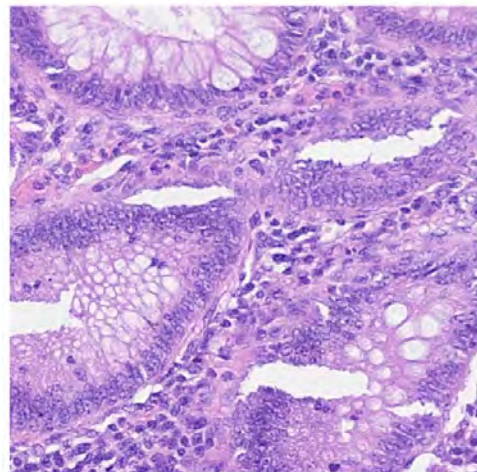
- Attention function e.g. $\alpha(w, x) \propto \exp(w^\top \tanh Vx)$

Deep Sets with Attention aka Multi-Instance Learning (Ilse, Tomczak, Welling, '18)

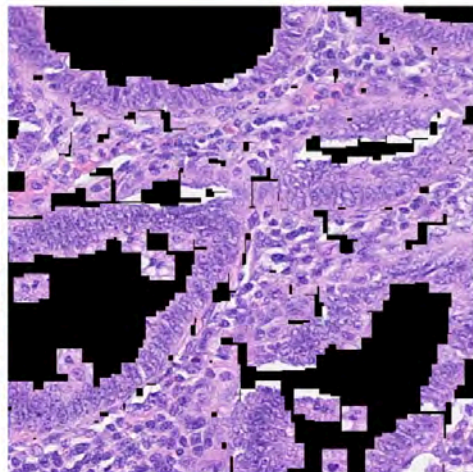
Identifying sets that contain the digit '9'



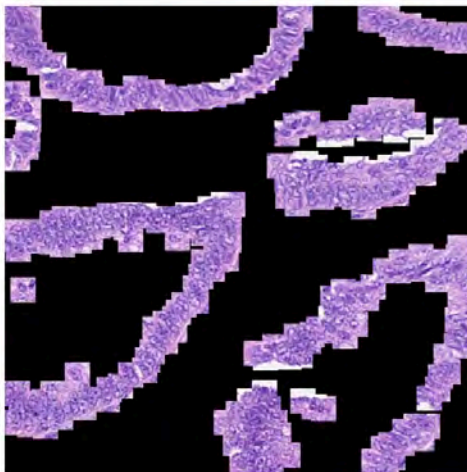
Deep Sets with Attention aka Multi-Instance Learning (Ilse, Tomczak, Welling, '18)



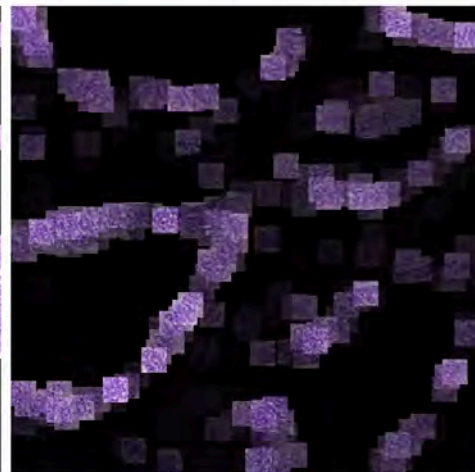
tissue
sample



windowed
cell nuclei



cancerous
cells



attention
weights

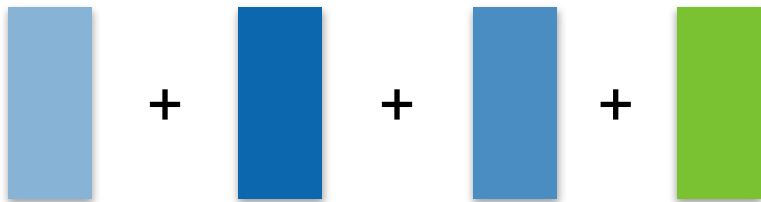
Bag of words (Salton & McGill, 1986)

Word2Vec (Mikolov et al., 2013)

- Embed each word in sentence (word2vec, binary, GRU ...)
- Add them all up
- Classify

$$f(X) = \rho \left(\sum_{i=1}^n \phi(x_i) \right)$$

The tutorial is awesome.



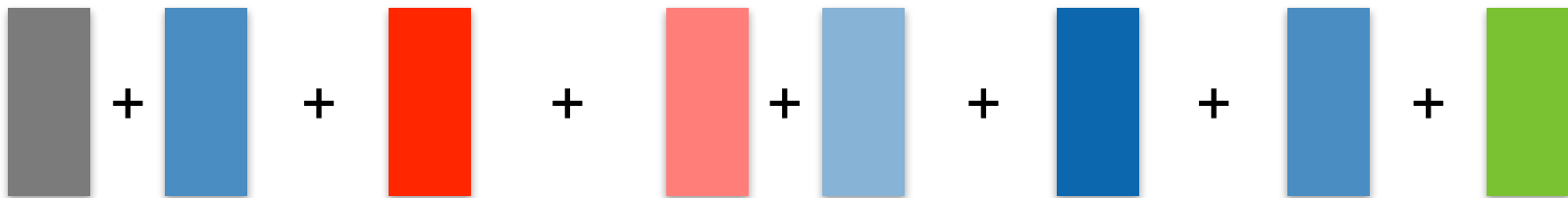
Bag of words (Salton & McGill, 1986)

Word2Vec (Mikolov et al., 2013)

- Embed each word in sentence (word2vec, binary, GRU ...)
- Add them all up
- Classify

$$f(X) = \rho \left(\sum_{i=1}^n \phi(w_i) \right)$$

Alex is obnoxious but the tutorial is awesome.

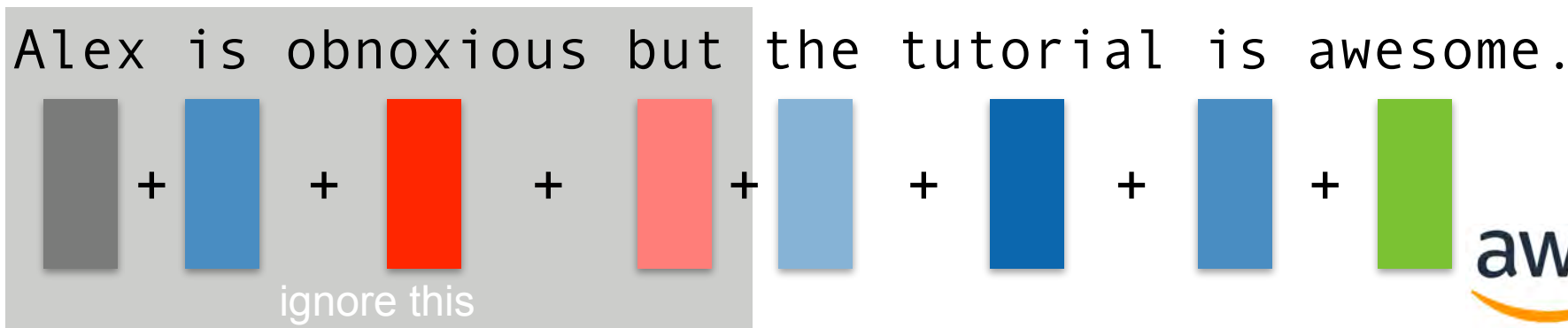


Bag of words (Salton & McGill, 1986)

Word2Vec (Mikolov et al., 2013)

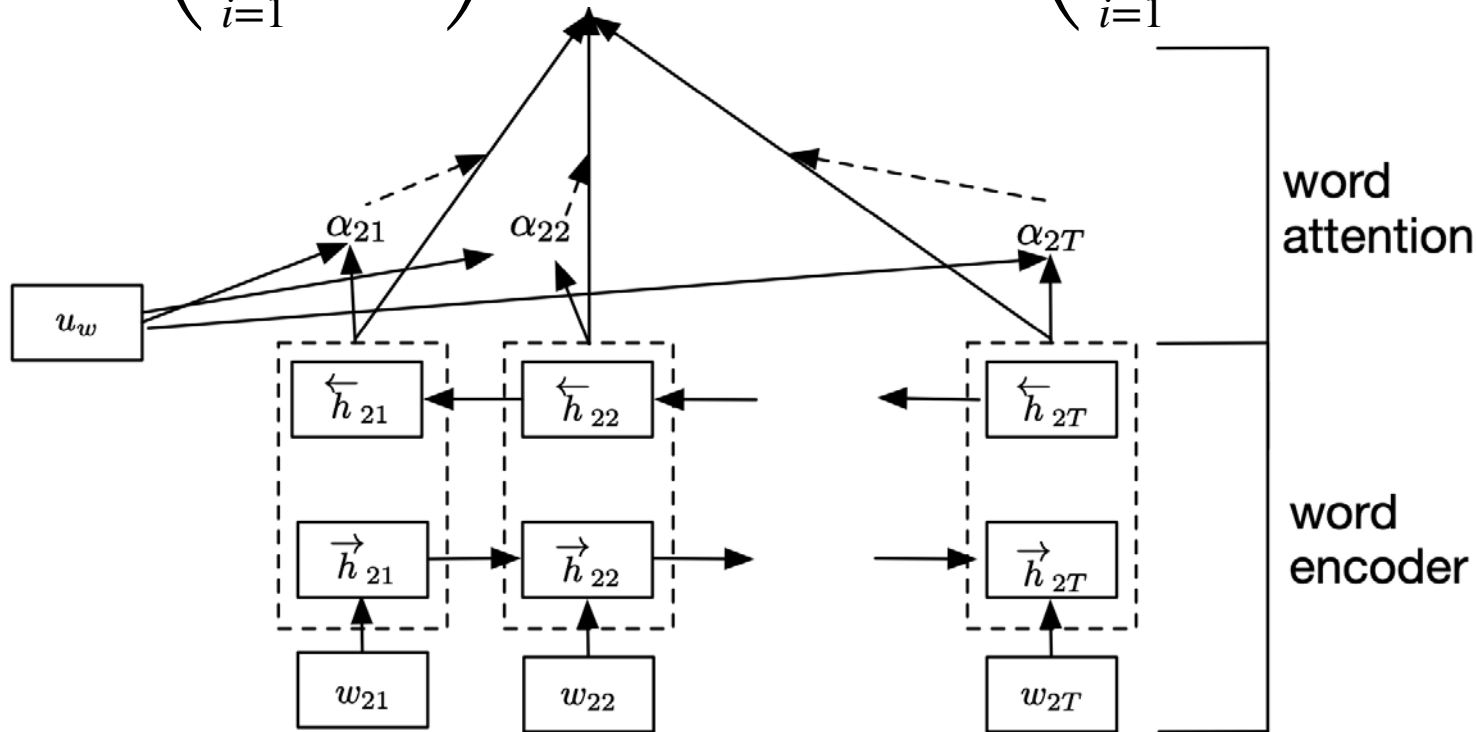
- Embed each word in sentence (word2vec, binary, GRU ...)
- Add them all up
- Classify

$$f(X) = \rho \left(\sum_{i=1}^n \phi(w_i) \right)$$



Attention weighting for documents (Wang et al, '16)

$$f(X) = \rho \left(\sum_{i=1}^n \phi(w_i) \right) \longrightarrow f(X) = \rho \left(\sum_{i=1}^n \alpha(w_i, X) \phi(w_i) \right)$$



Hierarchical attention weighting (Yang et al. '17)

Some sentences are more important than others ...

GT: 4 Prediction: 4

pork belly = delicious .
scallops ?
i do n't .
even .
like .
scallops , and these were a-m-a-z-i-n-g .
fun and tasty cocktails .
next time i 'm in phoenix , i will go
back here .
highly recommend .

GT: 0 Prediction: 0

terrible value .
ordered pasta entree .
.
\$ 16.95 good taste but size was an
appetizer size .
.
no salad , no bread no vegetable .
this was .
our and tasty cocktails .
our second visit .
i will not go back .

Hierarchical attention

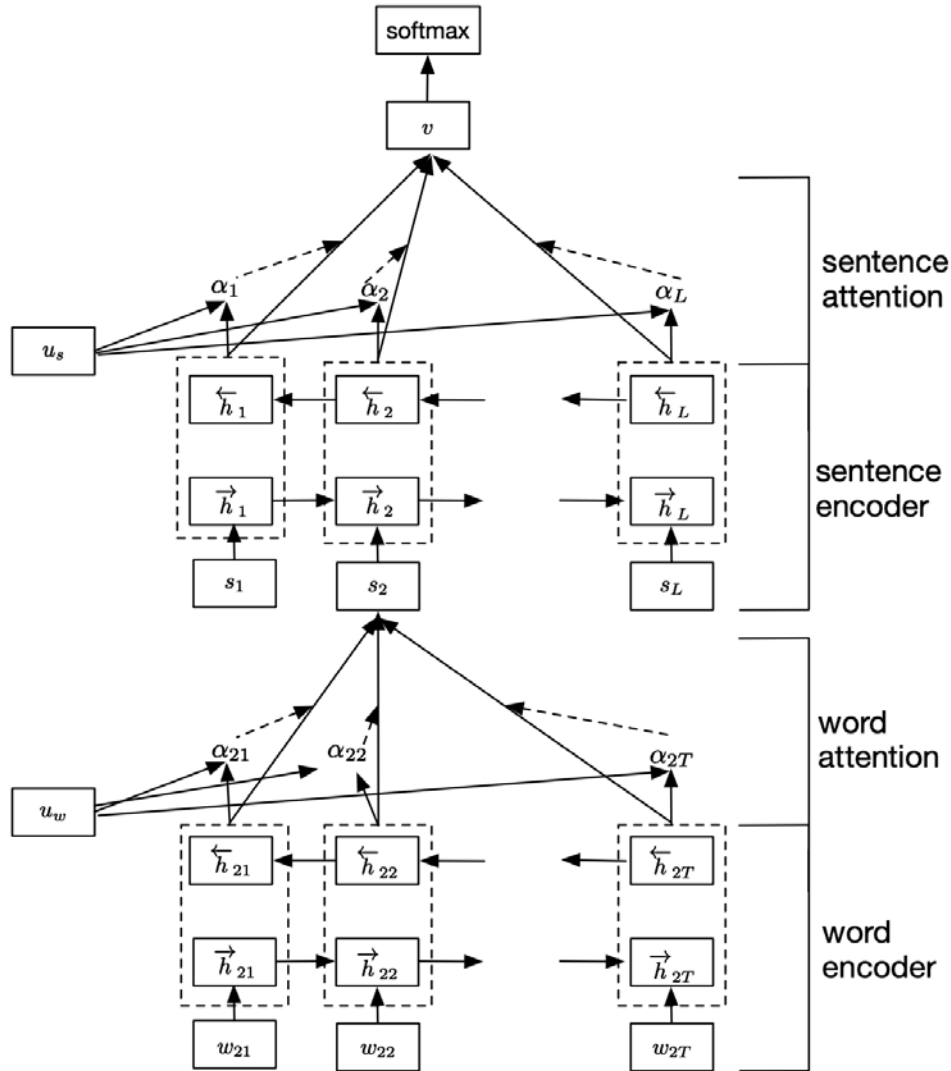
- Word level

$$f(s_i) = \rho \left(\sum_{j=1}^{n_i} \alpha(w_{ij}, s_i) \phi(w_{ij}) \right)$$

- Sentence level

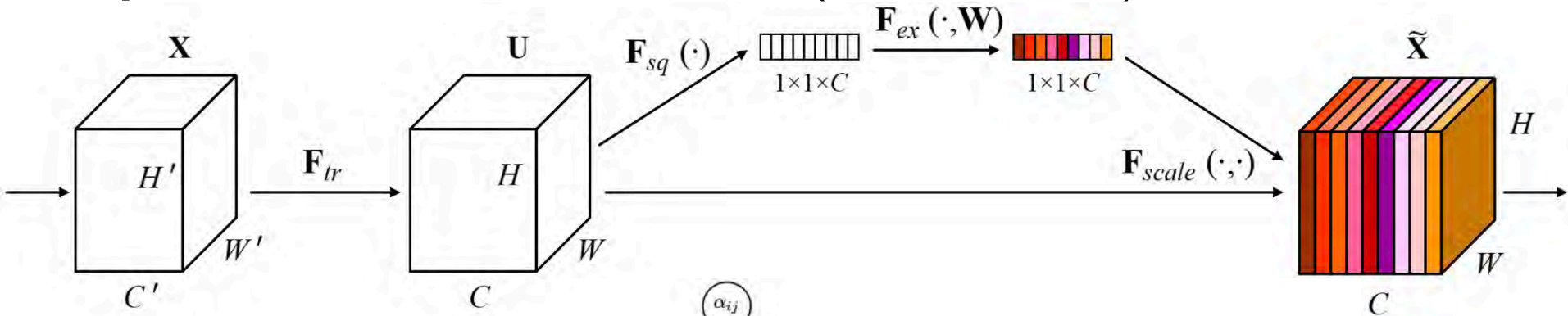
$$g(d) = \rho \left(\sum_{i=1}^n \alpha(s_i, d) \phi(f(s_i)) \right)$$

- Embeddings e.g. via GRU



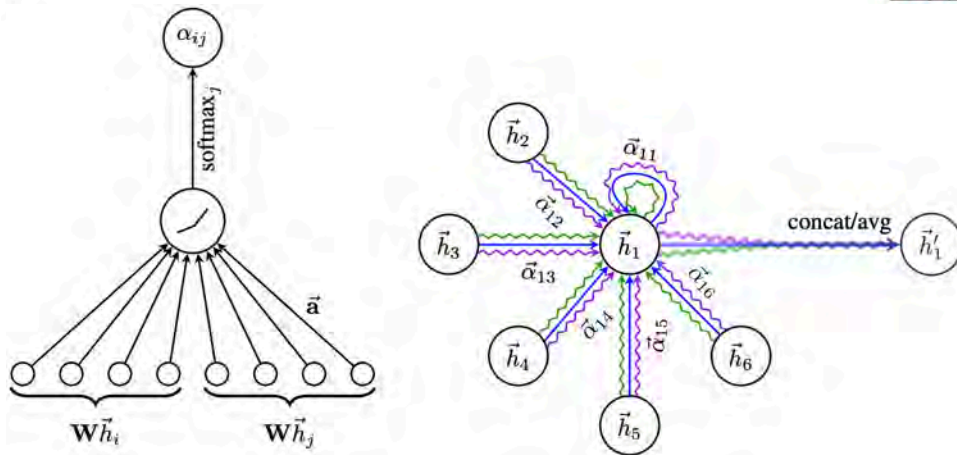
More Applications

Squeeze Excitation Networks (Hu et al., '18)



Graph Attention Networks

(Velickovic et al., '18)



Attention Summary

- Pooling

$$f(X) = \rho \left(\sum_{x \in X} \phi(x) \right)$$

Query w can depend on context

- Attention pooling

$$f(X) = \rho \left(\sum_{x \in X} \alpha(x, w) \phi(x) \right)$$

- Attention function (normalized to unit weight) such as

$$\alpha(x, X) \propto \exp(w^\top \tanh Ux)$$

3. Iterative Pooling



original
image

first attention
layer

second attention
layer



Question Answering

Joe went to the kitchen.

Fred went to the kitchen.

Joe picked up the milk.

Joe travelled to the office.

Joe left the milk.

Joe went to the bathroom.

Where is the milk?

Question Answering

Joe went to the kitchen.

Fred went to the kitchen.

Joe picked up the milk.

Joe travelled to the office.

Joe left the milk.

Joe went to the bathroom.

Where is the milk?

Question Answering

Joe went to the kitchen.

Fred went to the kitchen.

Joe picked up the milk.

Joe travelled to the office.

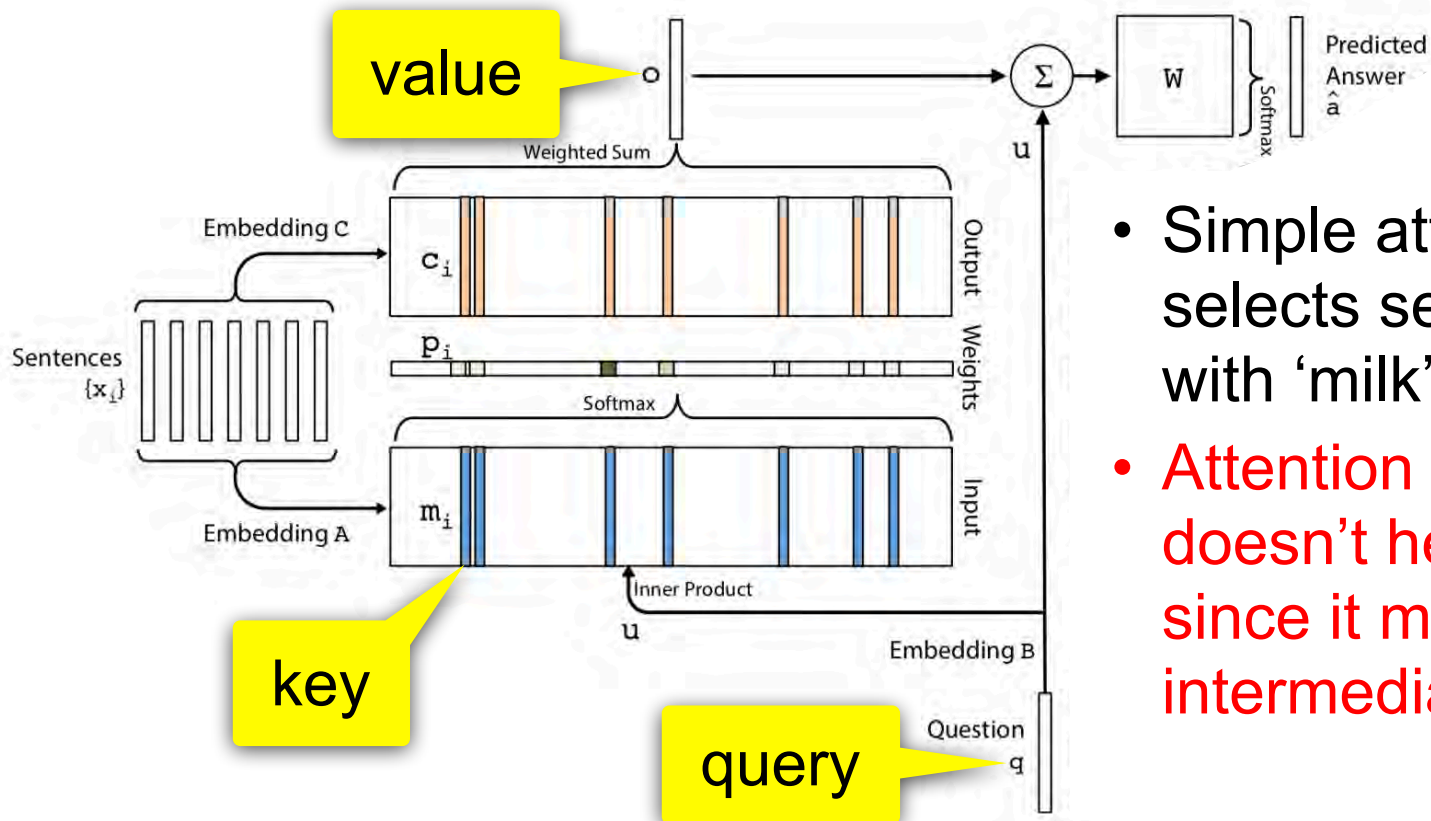
Joe left the milk.

Joe went to the bathroom.

Where is the milk?

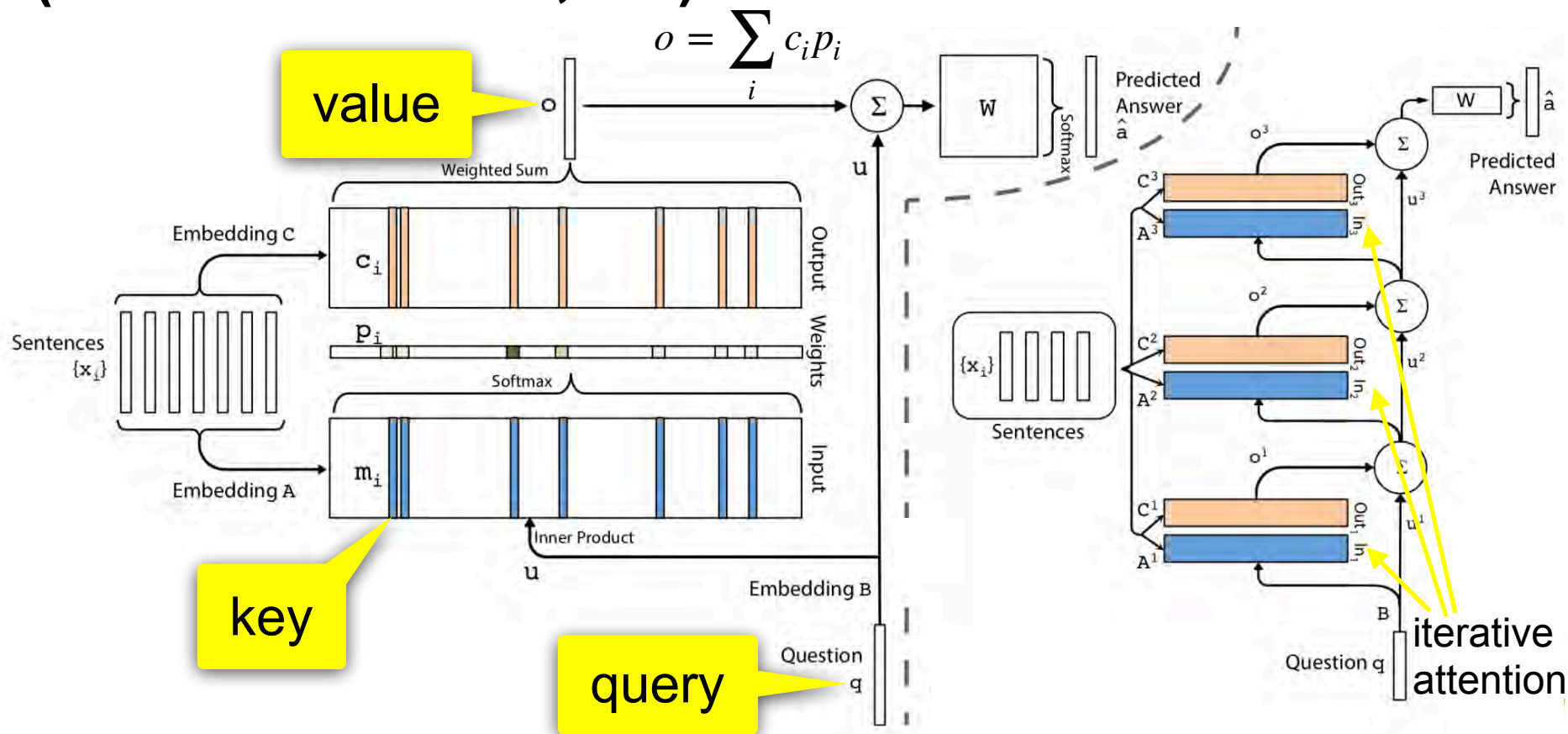
- Simple attention selects sentences with 'milk'.
- Attention pooling doesn't help much since it misses intermediate steps.

Question Answering with Pooling (Sukhbaatar et al., '15)



- Simple attention selects sentences with 'milk'.
- Attention pooling doesn't help much since it misses intermediate steps.

Question Answering with Pooling and Iteration (Sukhbaatar et al., '15)



Question Answering with Pooling and Iteration (Sukhbaatar et al., '15)

Sam walks into the kitchen.
Sam picks up an apple.
Sam walks into the bedroom.
Sam drops the apple.

Q: Where is the apple?

A. Bedroom

Brian is a lion.
Julius is a lion.
Julius is white.
Bernhard is green.

Q: What color is Brian?

A. White

Mary journeyed to the den.
Mary went back to the kitchen.
John journeyed to the bedroom.
Mary discarded the milk.

Q: Where was the milk before the den?

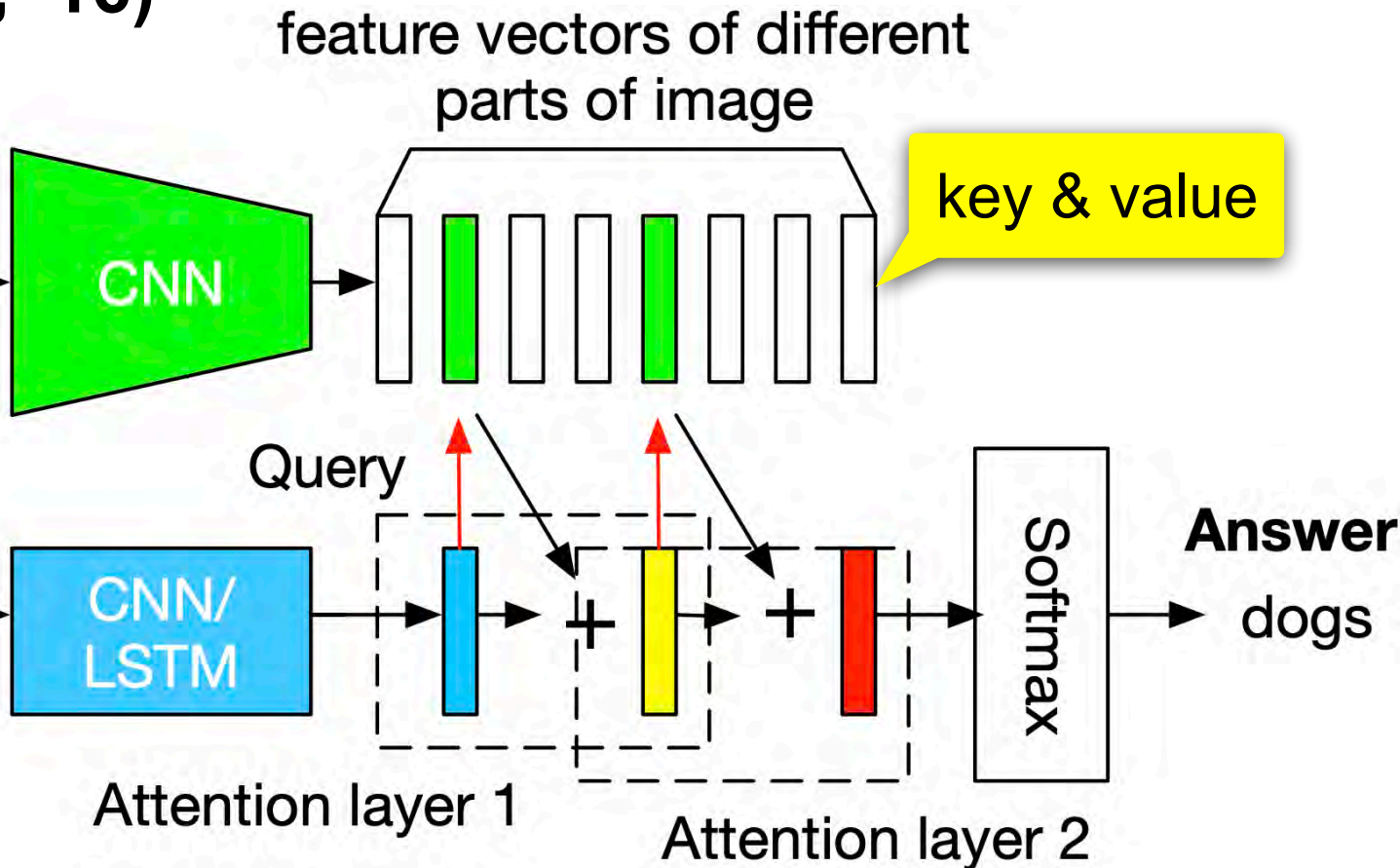
A. Hallway

Question Answering with Pooling and Iteration

(Yang et al., '16)



Question:
What are sitting
in the basket on
a bicycle?

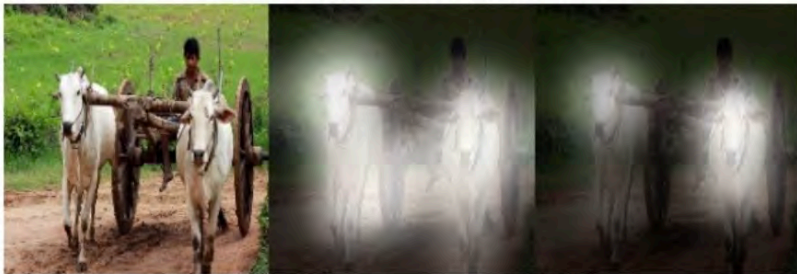


Question Answering with Pooling and Iteration (Yang et al., '16)

- Encode image via CNN
- Encode text query via LSTM
- Weigh patches according to attention and iterate

- Improving it (2019 tools)
 - Convolutionalize CNN (e.g. ResNet)
 - BERT for query encoding
 - Convolutional weighting (a la SE-Net)

(a) What are pulling a man on a wagon down on dirt road?
Answer: horses Prediction: horses



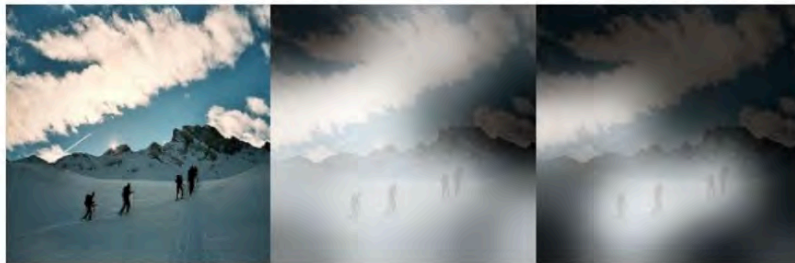
(b) What is the color of the box ?
Answer: red Prediction: red



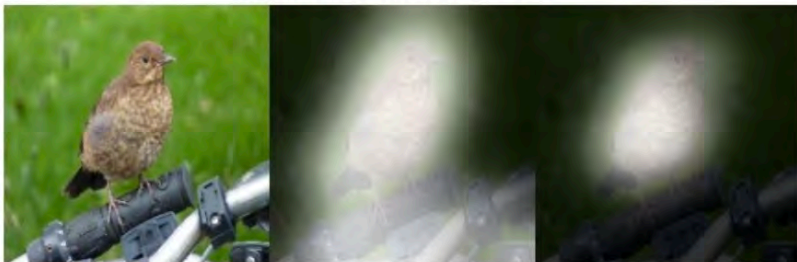
(c) What next to the large umbrella attached to a table?
Answer: trees Prediction: tree



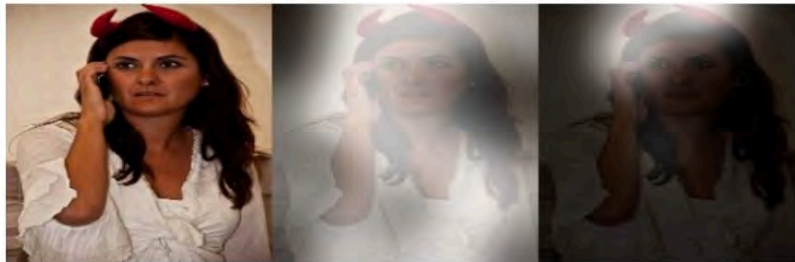
(d) How many people are going up the mountain with walking sticks?
Answer: four Prediction: four



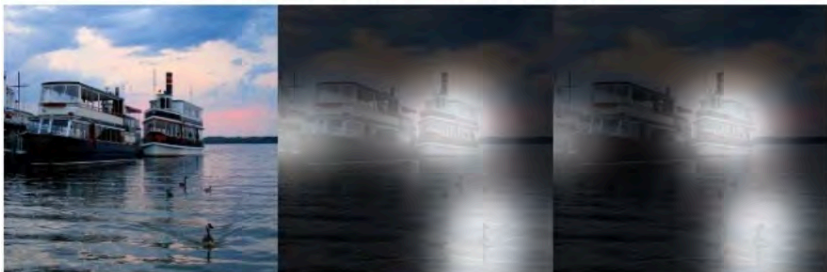
(e) What is sitting on the handle bar of a bicycle?
Answer: bird Prediction: bird



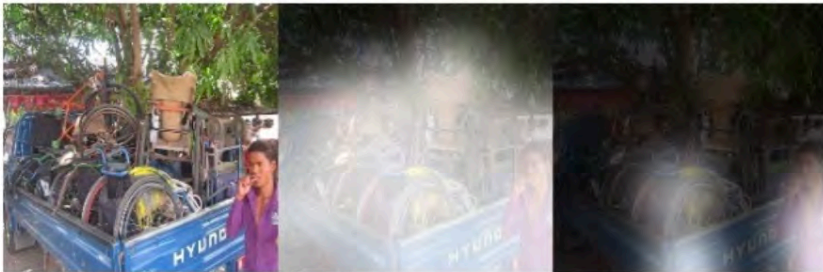
(f) What is the color of the horns?
Answer: red Prediction: red



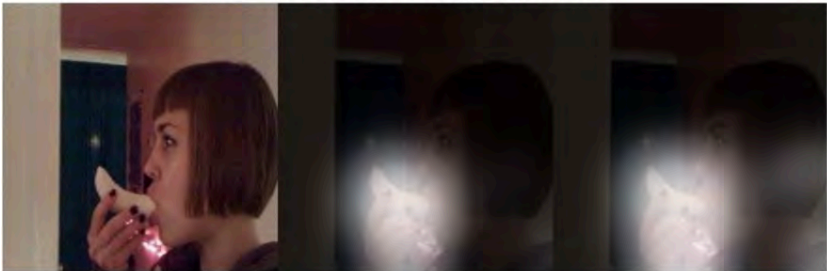
(a) What swim in the ocean near two large ferries?
Answer: ducks Prediction: boats



(b) What is the color of the shirt?
Answer: purple Prediction: green



(c) What is the young woman eating?
Answer: banana Prediction: donut



(d) How many umbrellas with various patterns?
Answer: three Prediction: two



(e) The very old looking what is on display?
Answer: pot Prediction: vase



(f) What are passing underneath the walkway bridge?
Answer: cars Prediction: trains



Iterative Attention Summary

- Pooling

$$f(X) = \rho \left(\sum_{x \in X} \phi(x) \right)$$

- Attention pooling

$$f(X) = \rho \left(\sum_{x \in X} \alpha(x, w) \phi(x) \right)$$

- Iterative Attention pooling

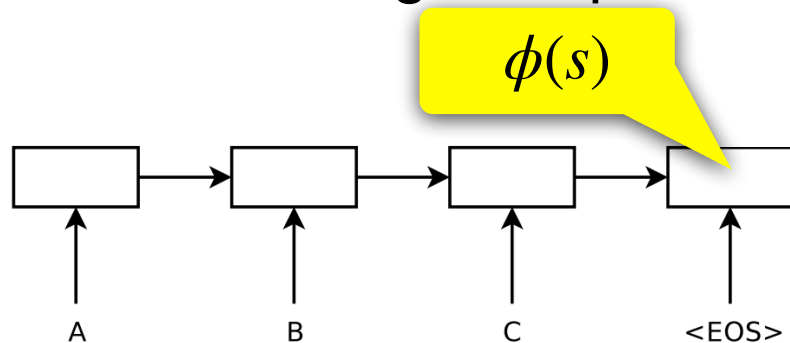
Repeatedly update
internal state

$$q_{t+1} = \rho \left(\sum_{x \in X} \alpha(x, q_t) \phi(x) \right)$$



Seq2Seq for Machine Translation, Sutskever, Vinyals, Le '14

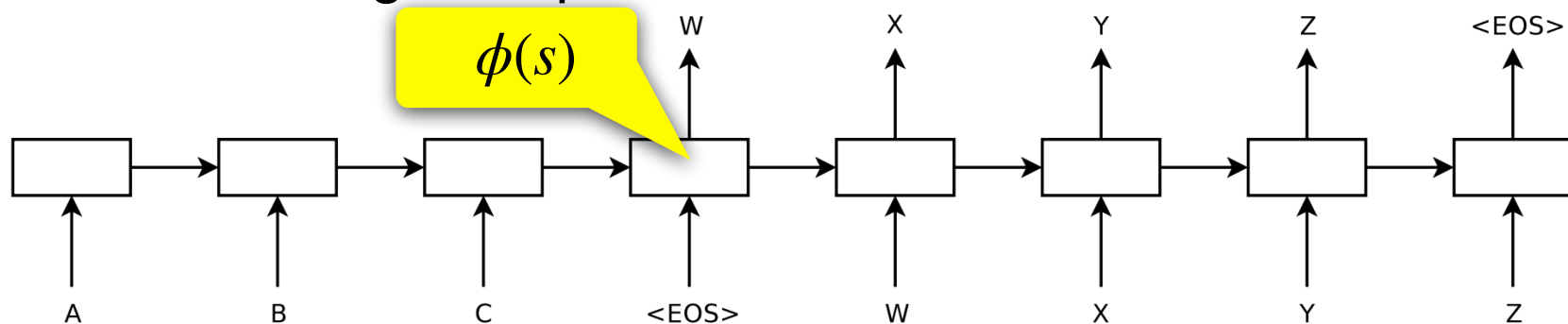
- Encode source sequence s via LSTM to representation $\phi(s)$
- Decode to target sequence one character at a time



- 'The table is round.' - 'Der Tisch ist rund.'
- 'The table is very beautiful with many inlaid patterns, blah blah blah blah' - 'Error ...'

Seq2Seq for Machine Translation, Sutskever, Vinyals, Le '14

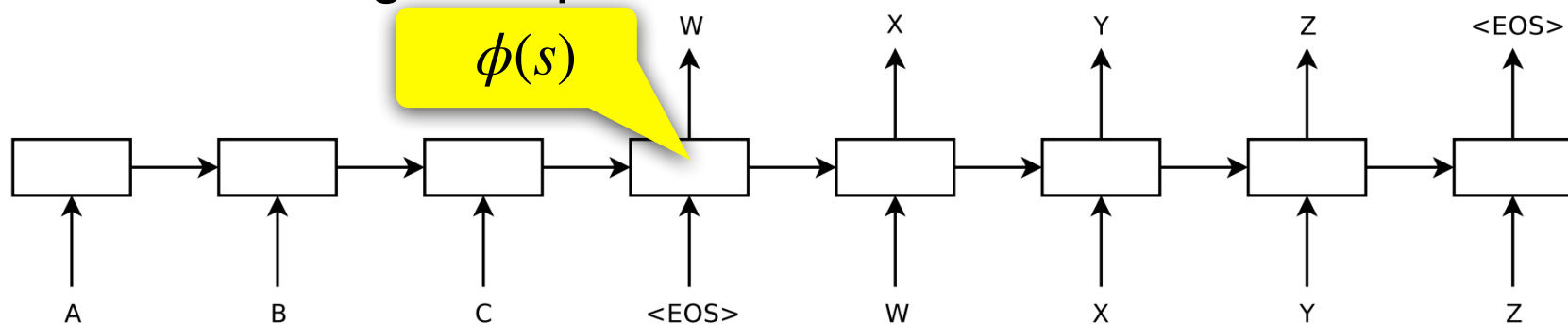
- Encode source sequence s via LSTM to representation $\phi(s)$
- Decode to target sequence one character at a time



- 'The table is round.' - 'Der Tisch ist rund.'
- 'The table is very beautiful with many inlaid patterns, blah blah blah blah' - 'Error ...'

Seq2Seq for Machine Translation, Sutskever, Vinyals, Le '14

- Encode source sequence s via LSTM to representation $\phi(s)$
- Decode to target sequence one character at a time



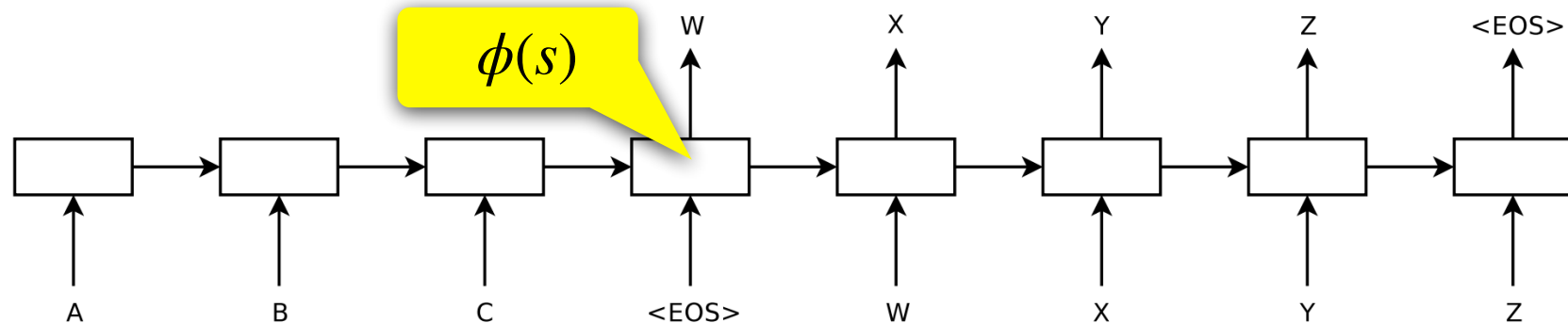
• 'The table is round.' - 'Der Tisch ist rund.'

• 'The table is very beautiful with n
blah blah blah blah' - 'Error ...'

Representation
not rich enough

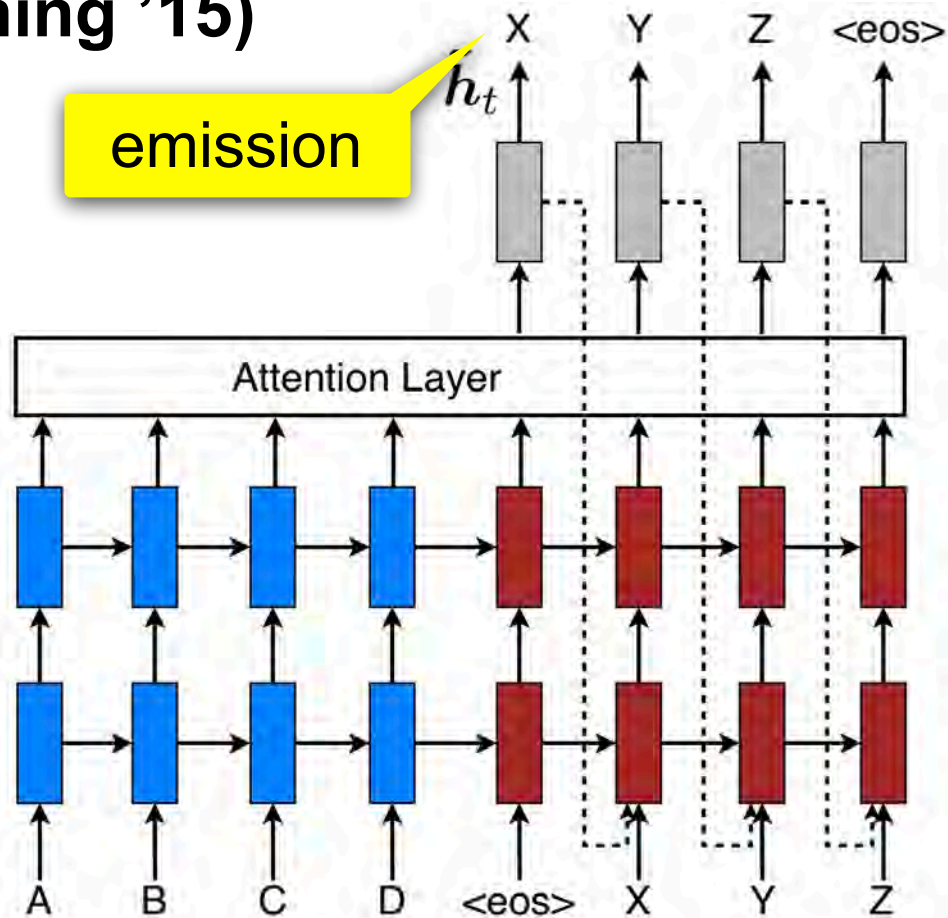
Seq2Seq for Machine Translation, Sutskever, Vinyals, Le '14

- Encode source sequence s via LSTM to latent representation $\phi(s)$
- Decode to target sequence one character at a time



- Need memory for long sequences
- Attention to iterate over source
(we can look up source at any time after all)

Seq2Seq with attention (Bahdanau, Cho, Bengio '14) (Pham, Luong, Manning '15)



Seq2Seq with attention (Bahdanau, Cho, Bengio '14) (Pham, Luong, Manning '15)

$$\alpha_{ij} \propto \exp(a(\tilde{h}_{i-1}, h_j))$$

query

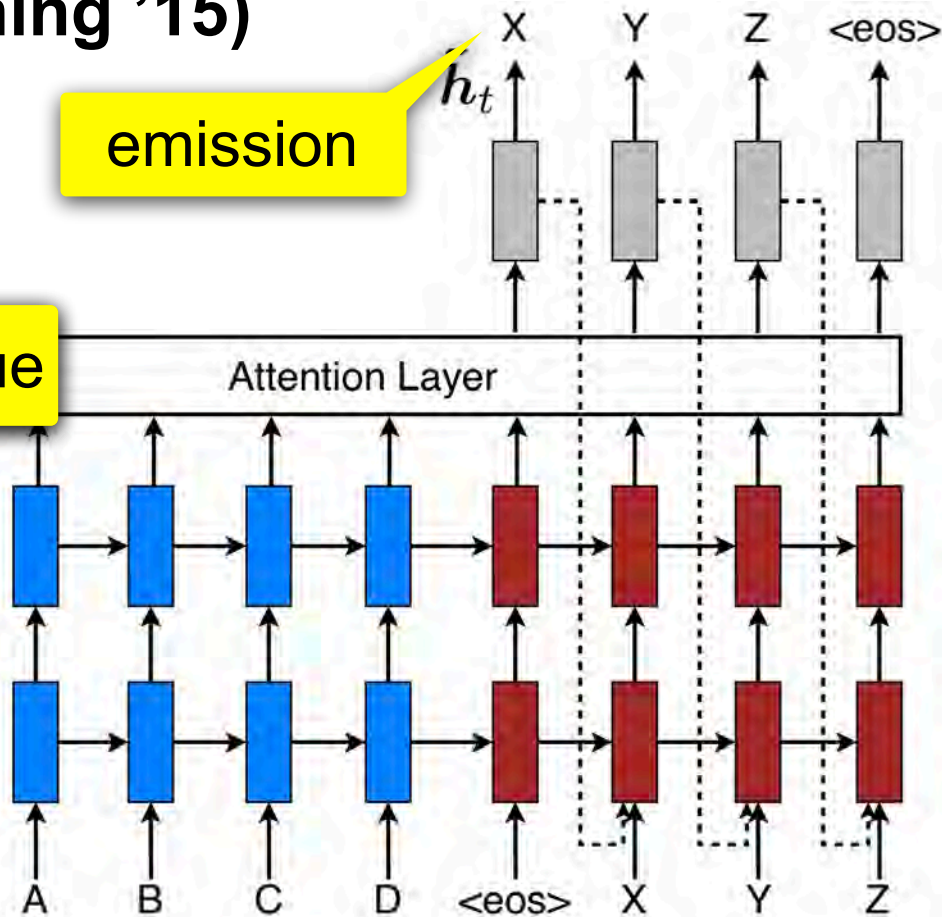
key & value

$$c_i = \sum_{j=1}^n \alpha_{ij} h_j$$

$$\tilde{h}_i = f(s_{i-1}, y_{i-1}, c_i)$$

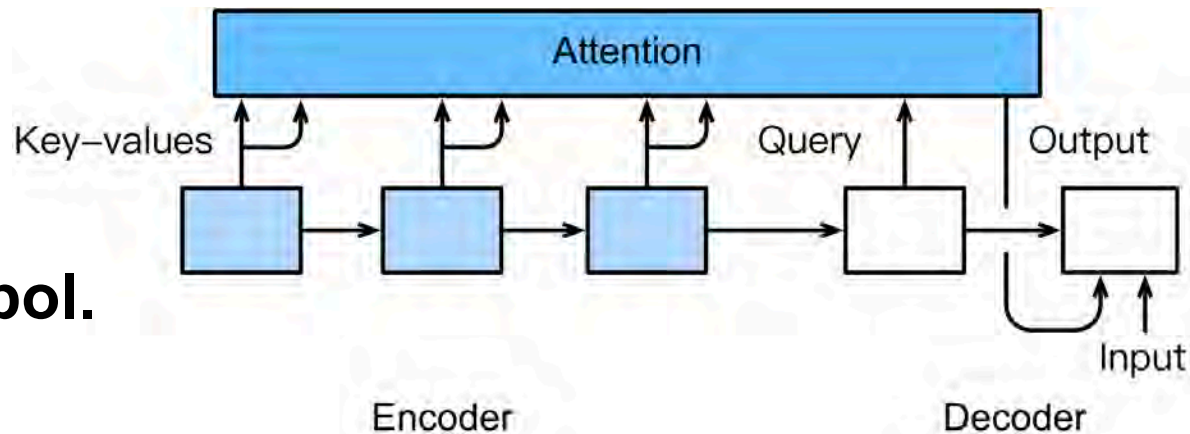


y_i

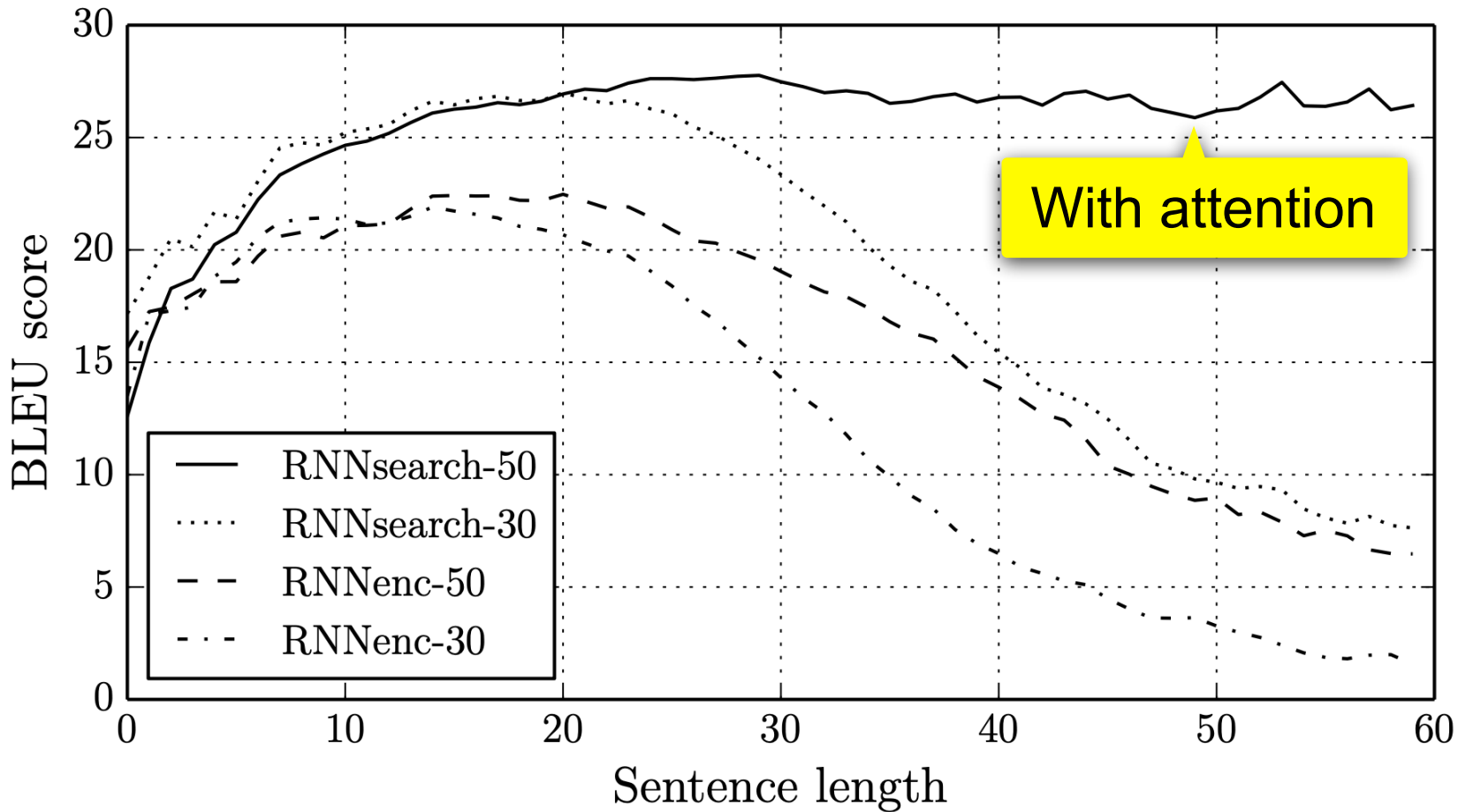


Seq2Seq with attention (Bahdanau, Cho, Bengio '14) (Pham, Luong, Manning '15)

- Iterative attention model
 - Compute (next) attention weights
 - Aggregate next state
 - Emit next symbol
- Repeat
- **Memory networks emit only one symbol.**
- **NMT with attention emits many symbols.**



Seq2Seq with attention (Bahdanau, Cho, Bengio '14)



Variations on a Theme

BWV 988

(PART I)

J.S. Bach (1685-1750)

Aria

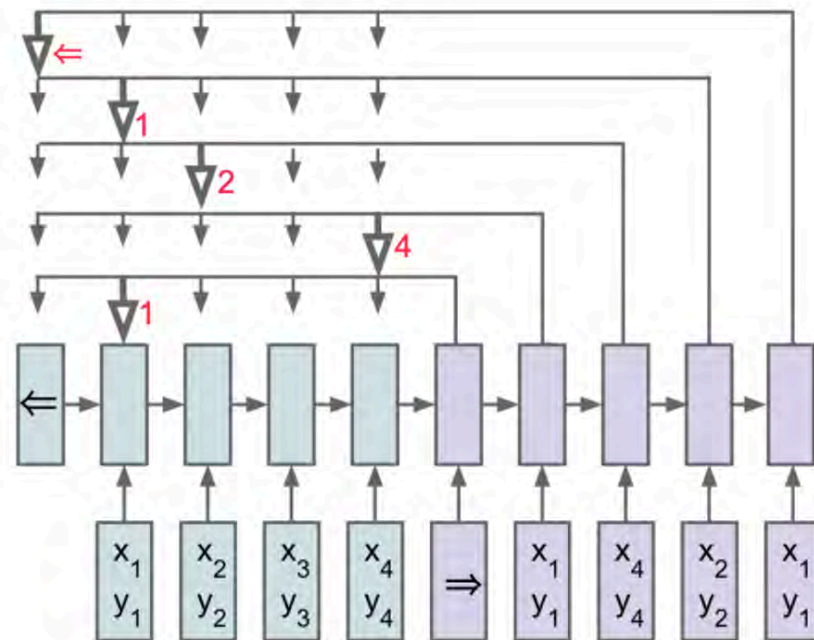
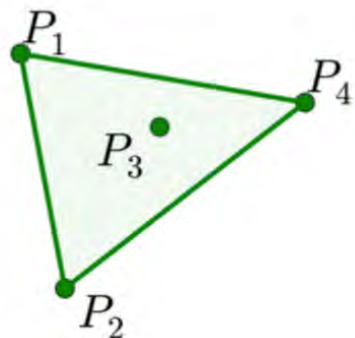
The image shows the first four measures of the Aria from BWV 988. The music is written in G major (one sharp) and 3/4 time. The right hand (treble clef) plays a melodic line with various ornaments and a trill. The left hand (bass clef) provides a harmonic accompaniment with chords and moving bass lines. The first measure starts with a quarter note G4, followed by a quarter note A4, and a quarter note B4. The second measure has a quarter note A4, a quarter note G4, and a quarter note F#4. The third measure has a quarter note E4, a quarter note D4, and a quarter note C4. The fourth measure has a quarter note B3, a quarter note A3, and a quarter note G3. The bass line in the first measure has a half note G2, and in the second measure, a half note F#2. The third measure has a half note E2, and the fourth measure has a half note D2. The piece is marked 'Aria'.

The image shows the first four measures of the second variation of the Aria from BWV 988. The music is written in G major (one sharp) and 3/4 time. The right hand (treble clef) plays a melodic line with various ornaments and a trill. The left hand (bass clef) provides a harmonic accompaniment with chords and moving bass lines. The first measure starts with a quarter note G4, followed by a quarter note A4, and a quarter note B4. The second measure has a quarter note A4, a quarter note G4, and a quarter note F#4. The third measure has a quarter note E4, a quarter note D4, and a quarter note C4. The fourth measure has a quarter note B3, a quarter note A3, and a quarter note G3. The bass line in the first measure has a half note G2, and in the second measure, a half note F#2. The third measure has a half note E2, and the fourth measure has a half note D2. The piece is marked 'Aria'.

Pointer networks for finding convex hull (Vinyals et al., '15)

Input $P = \{P_1, \dots, P_4\}$

Output $O = \{1, 4, 2, 1\}$



Pointer networks for finding convex hull (Vinyals et al., '15)

Input $P = \{P_1, \dots, P_4\}$

Output $O = \{1, 4, 2, 1\}$

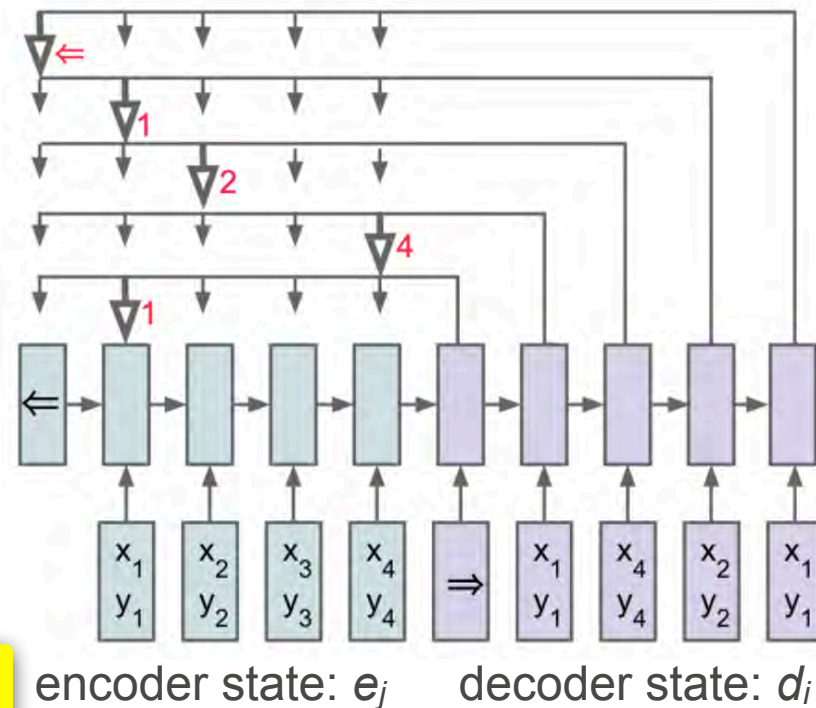
key

query

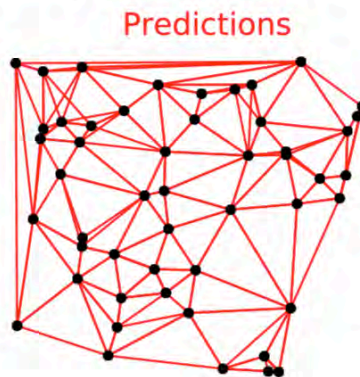
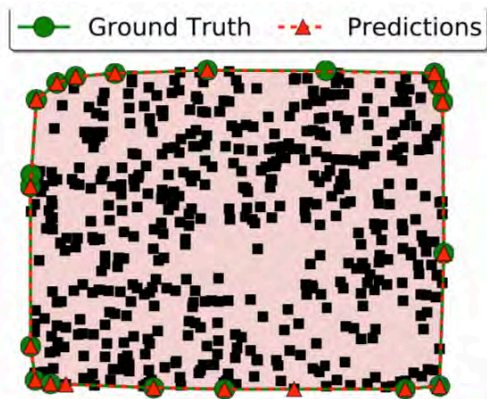
$$u_{ij} = v^T \tanh(W[e_j, d_i])$$

$$p(C_i | C_{[1:i-1]}, P) = \text{softmax}(u_i)$$

attention weight as
prediction distribution



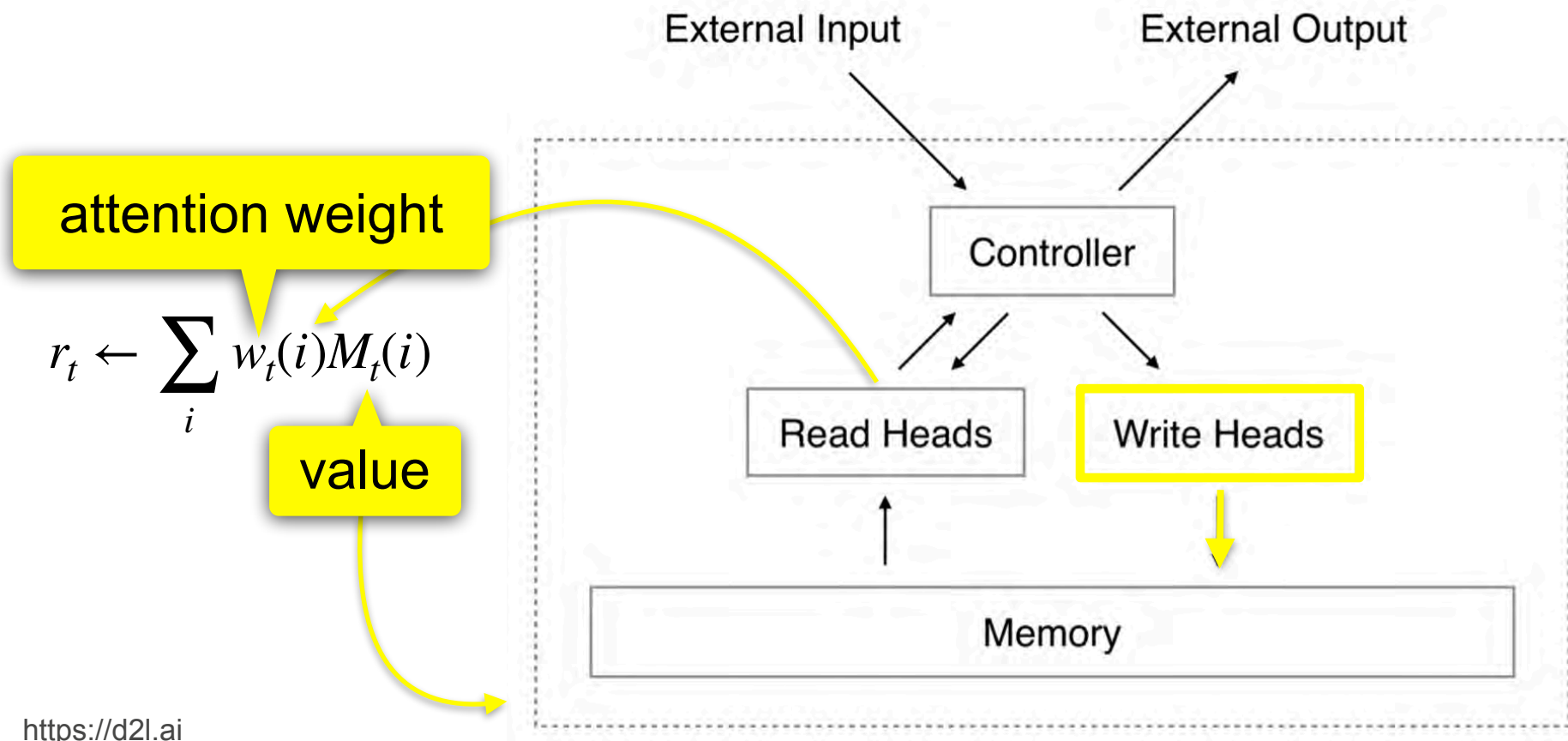
Convex hulls, Delaunay triangulation, and TSP



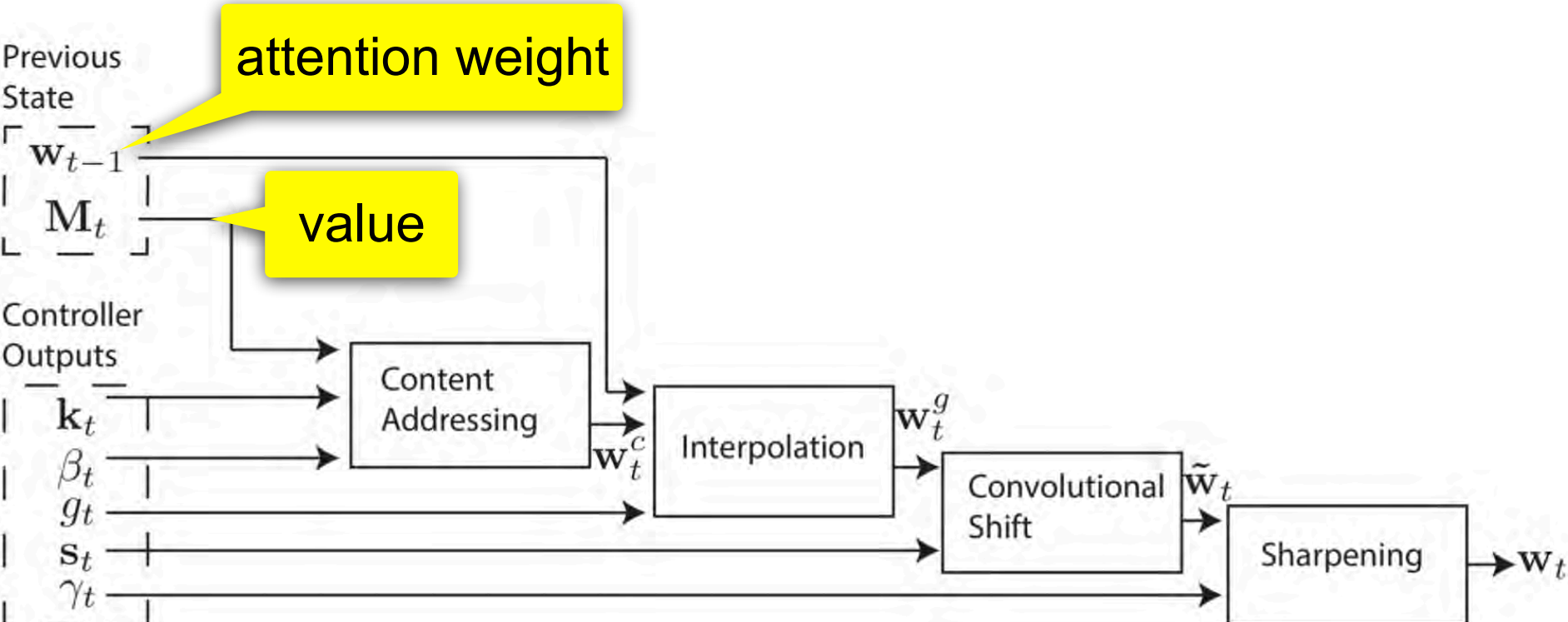
2019 style improvements

- Transformer to encode inputs (and outputs)
- Graph neural networks for local interactions

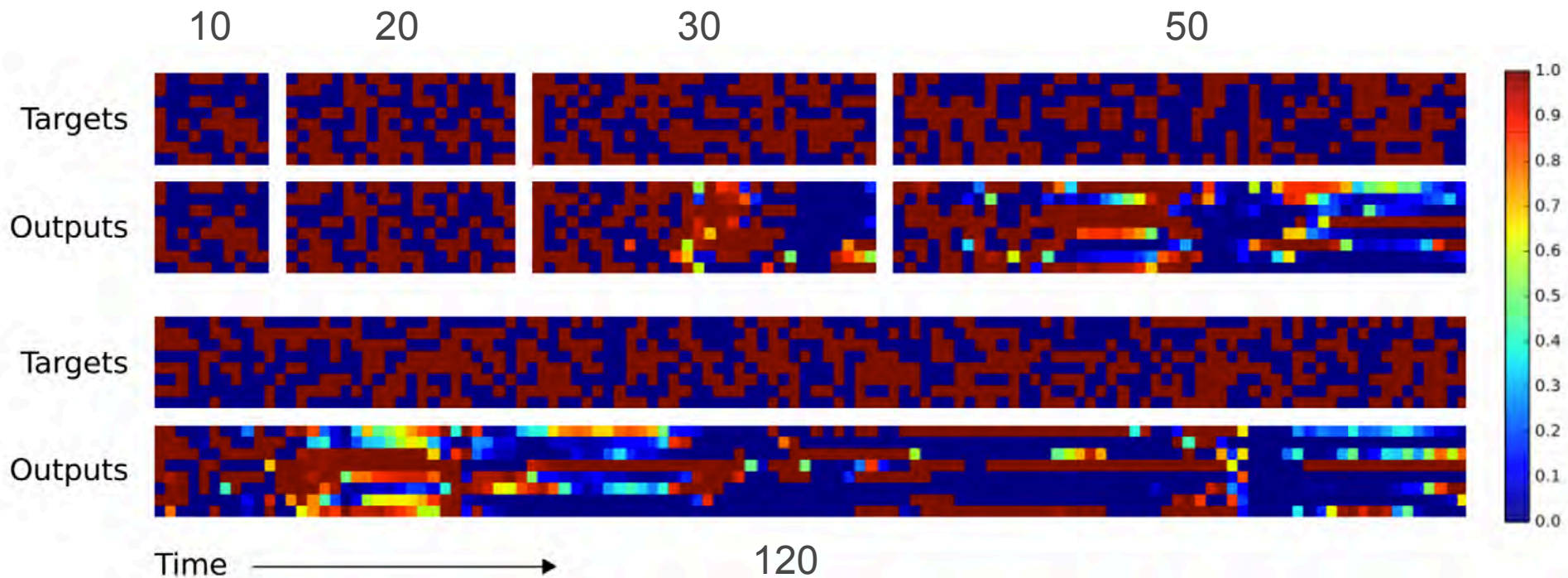
Neural Turing Machines (Graves et al., '14)



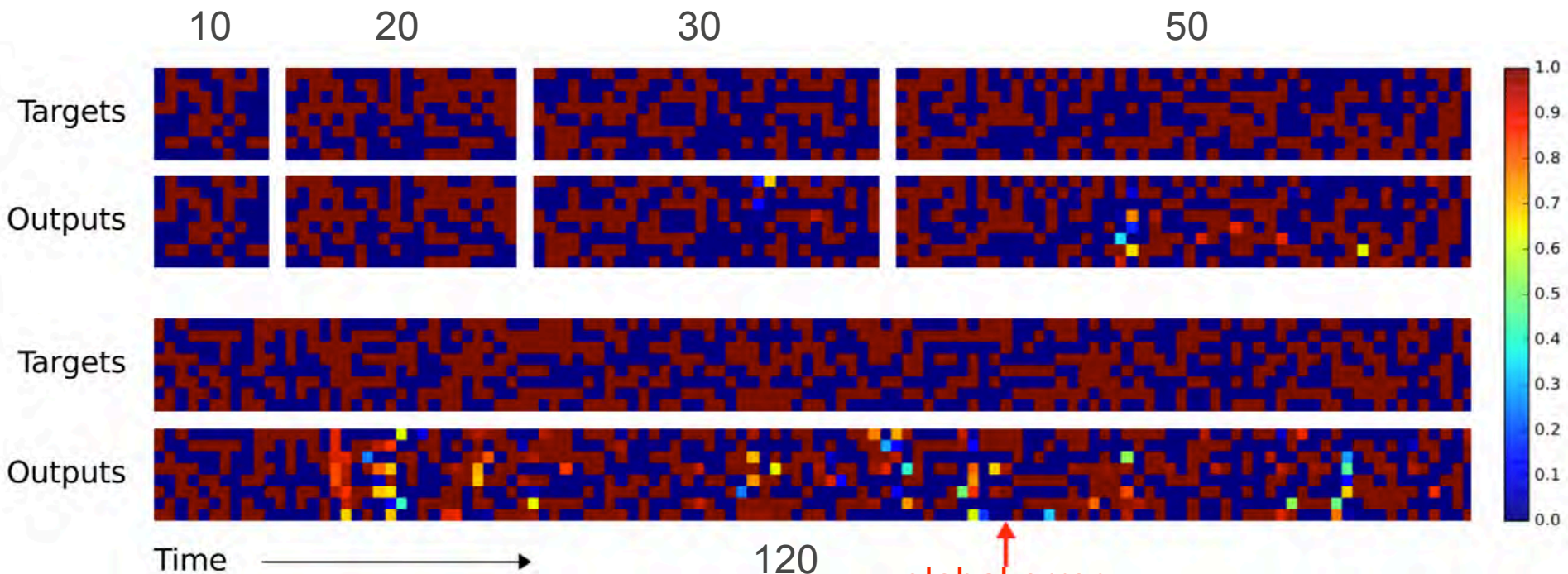
Attention weights can be **stateful (values, too)**



Copying a sequence (with LSTM)



Copying a sequence (with NTM)



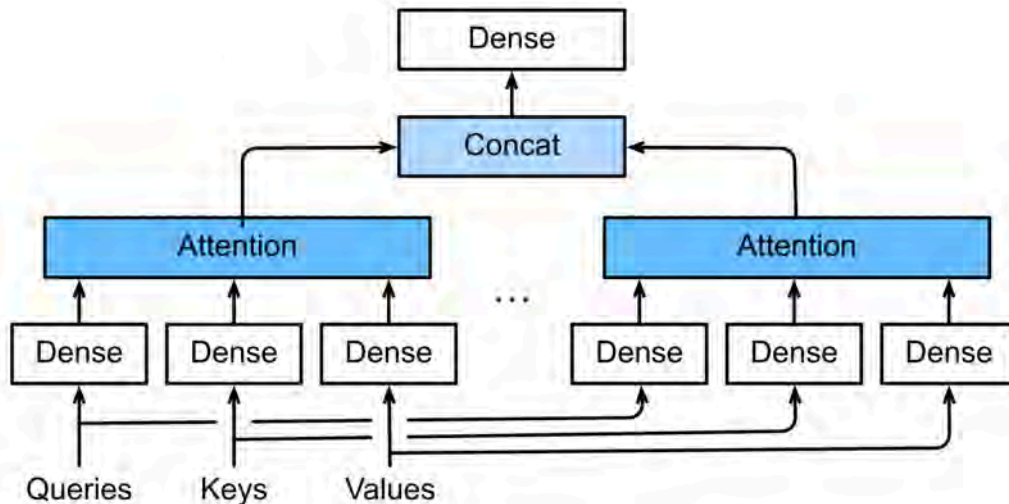
global error:
push all subsequent
vectors one step back



5. Multiple Heads

Multi-head attention (Vaswani et al., '17)

Q: query
K: key
V: value



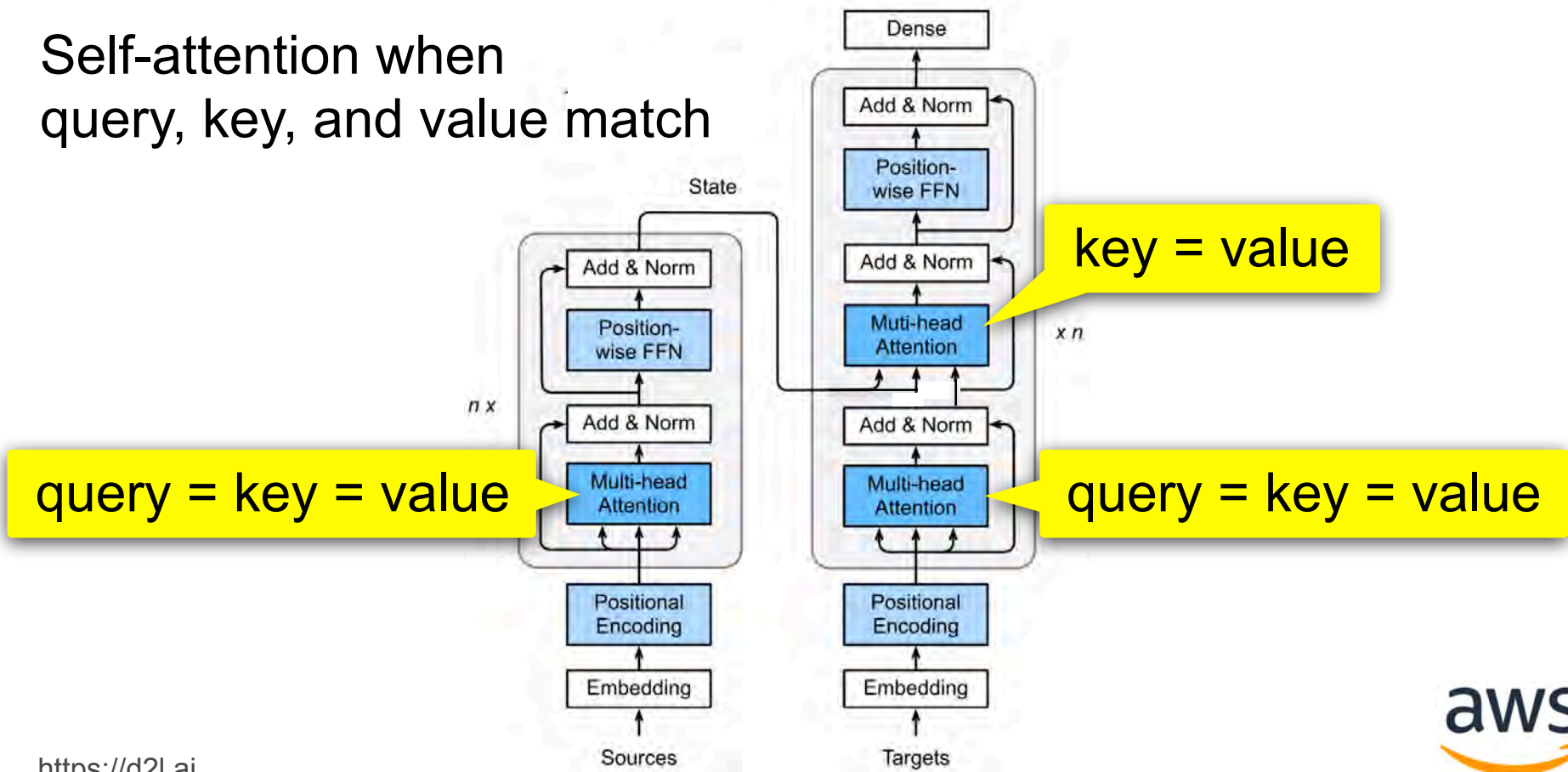
$$\text{Attention}(Q, K, V) = \text{softmax} \left(\frac{QK^T}{\sqrt{d_k}} \right) V$$

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \dots, \text{head}_h) W^O$$

$$\text{where head}_i = \text{Attention} \left(QW_i^Q, KW_i^K, VW_i^V \right)$$

Transformer with multi-head attention (Vaswani et al., '17)

Self-attention when query, key, and value match



Semantic Segmentation

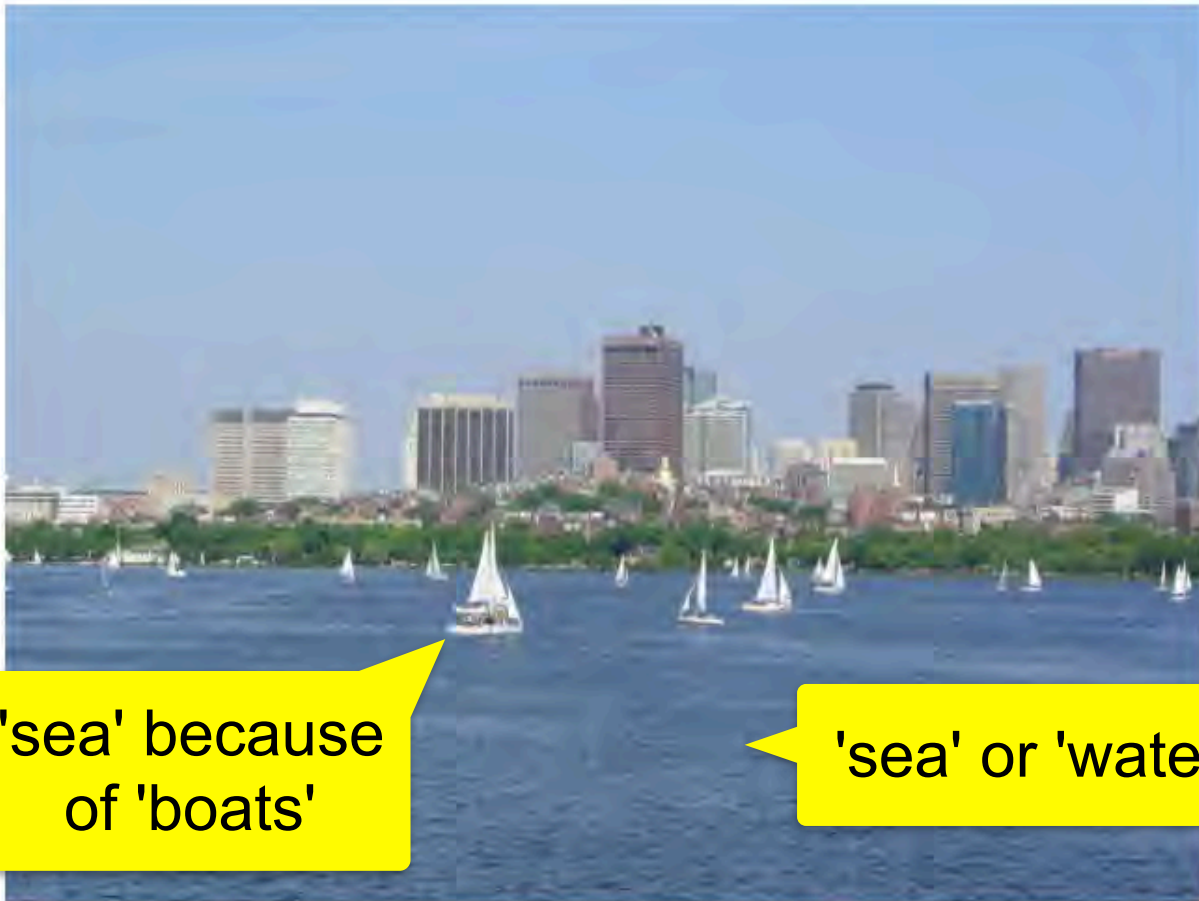


'sea' or 'water'?

Semantic Segmentation



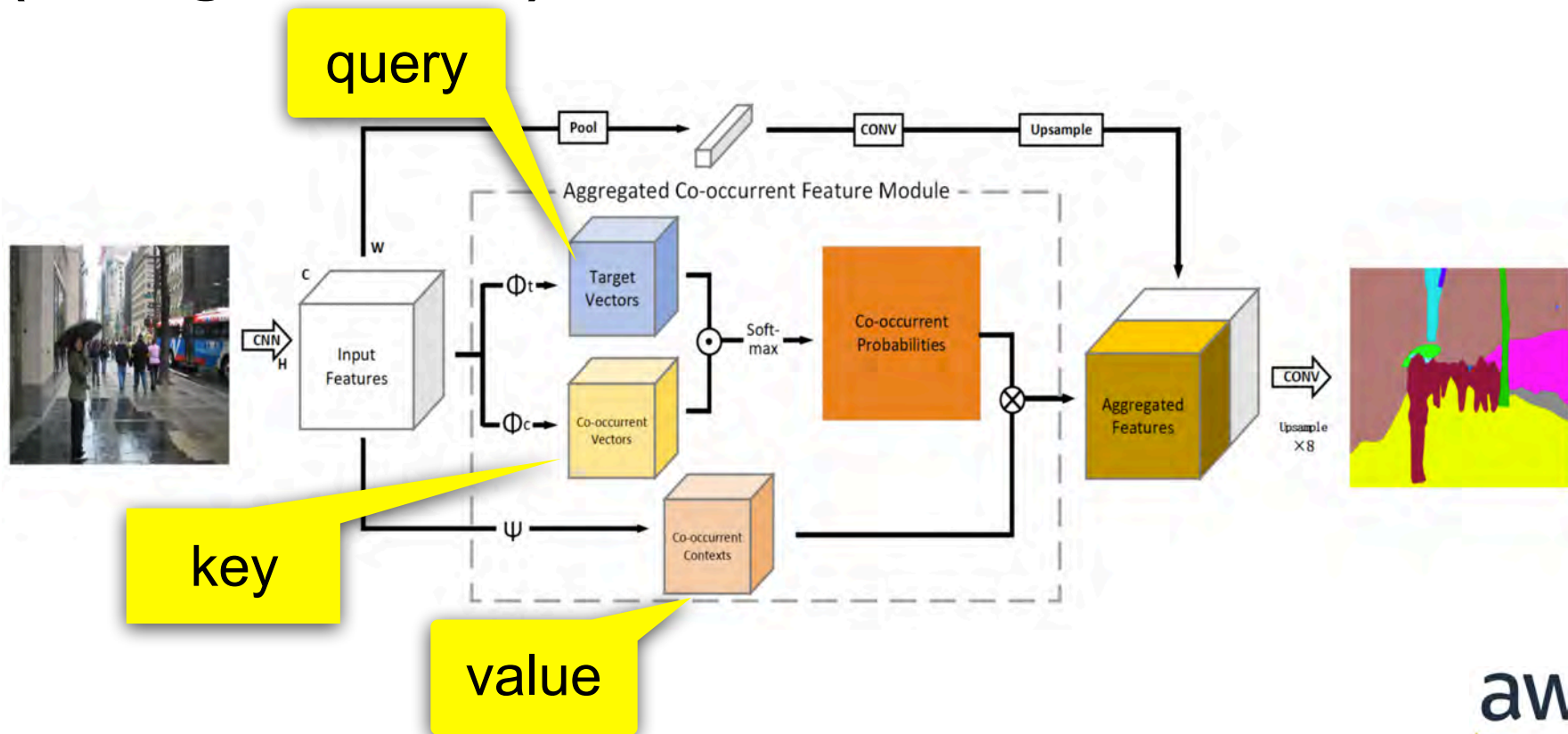
Semantic Segmentation



'sea' because
of 'boats'

'sea' or 'water'?

Multi-head attention for semantic segmentation (Zhang et al., '19)



Classify pixels co-occurring with boat as **sea** rather than **water**



(a) Image



(b) Ground Truth



(c) FCN (baseline)



(d) CFNet (**ours**)



(e) legend

BERT

Bidirectional Encoder Representations from Transformers

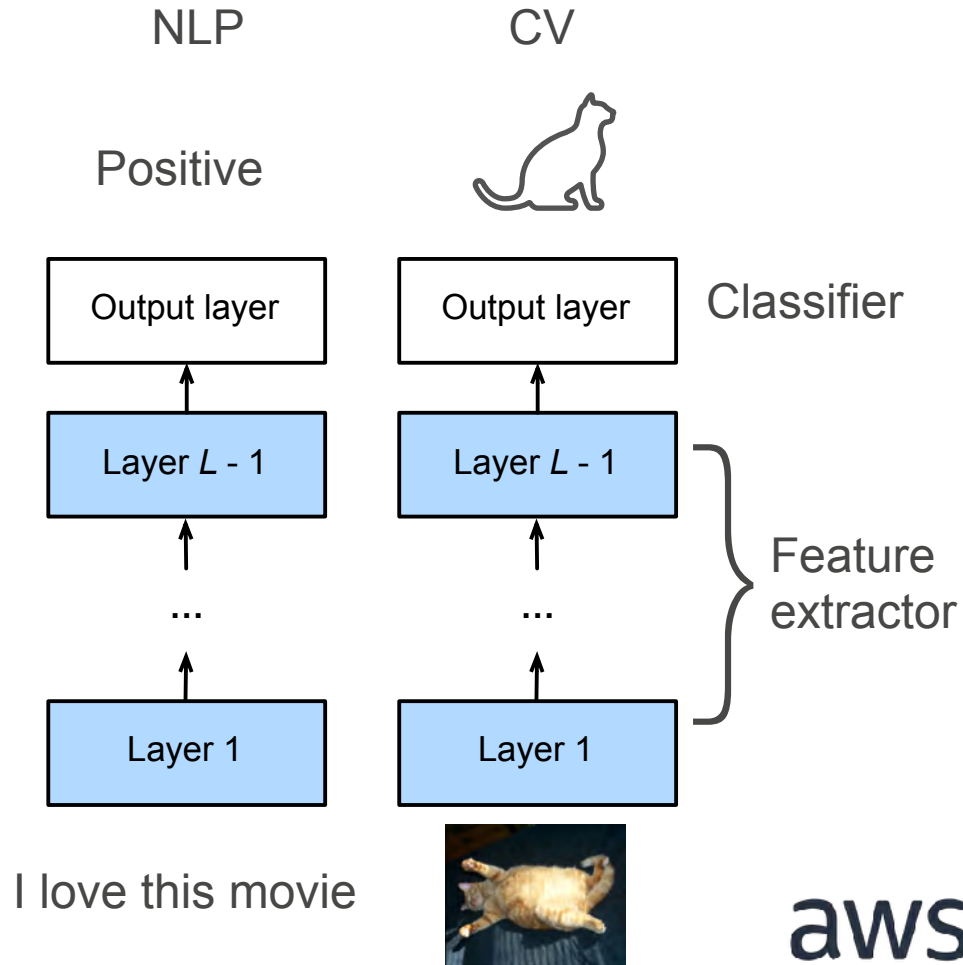
(Devlin et al, 2018)

SOTA on 11 NLP tasks



Motivation

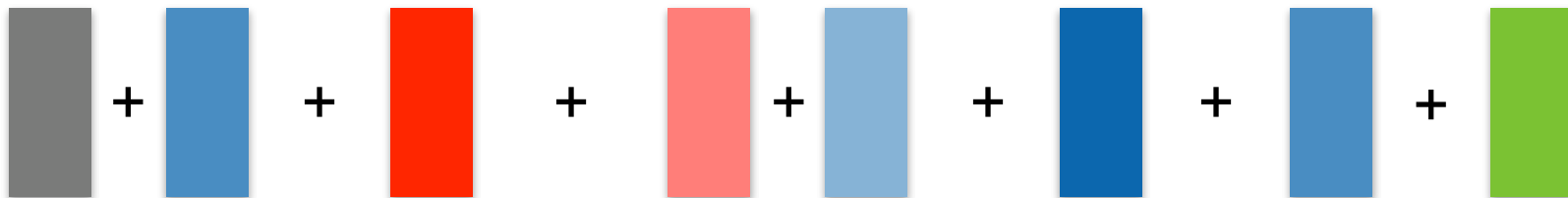
- Fine-tuning for NLP (learning a prior for NLP)
- Pre-trained model captures prior
- Only add one (or more) output layers for new task



Transfer Learning with Embeddings

- Pre-trained embeddings for new models (e.g. word2vec)

Alex is obnoxious but the tutorial is awesome.



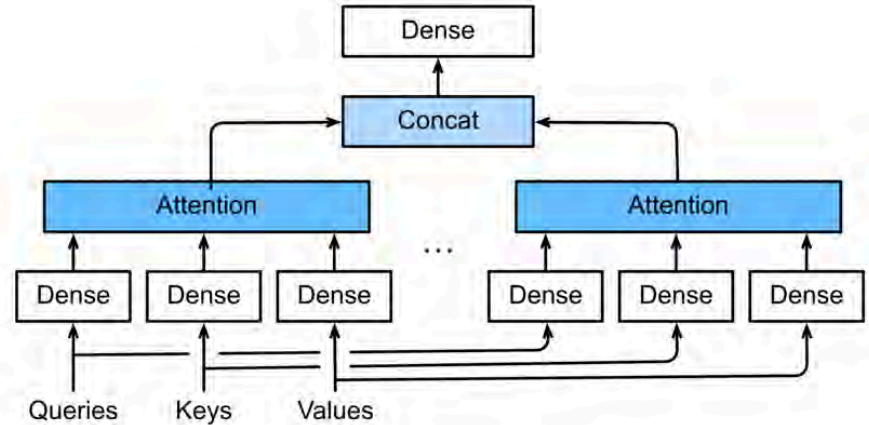
- Word2vec ignores sequential information entirely

GPT uses Transformer **Decoder** (Radford et al., '18)

- Pre-train language model, then fine-tune on each task
- **Trained on full length documents**
- 12 blocks, 768 hidden units, 12 heads
- **SOTA for 9 NLP tasks**
- Language model only looks **forward**
 - I went to the **bank** to deposit some money.
 - I went to the **bank** to sit down.

Architecture

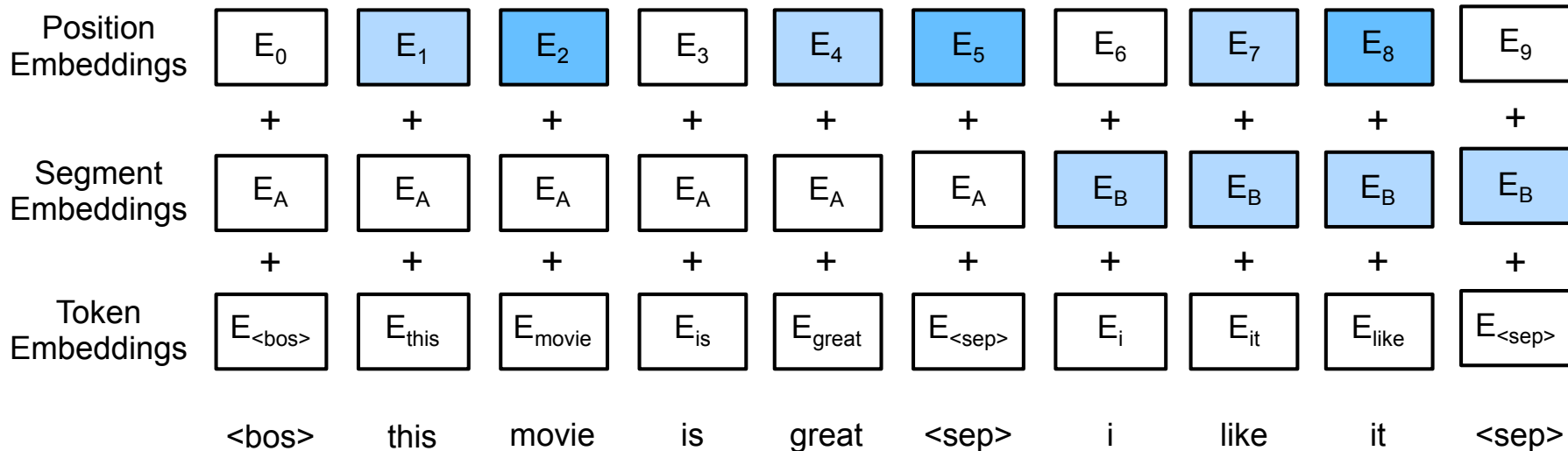
- (Big) transformer encoder
- Train on large corpus (books, wikipedia) with > 3B words



	blocks	hidden units	heads	parameters
small	12	768	12	110M
large	24	1024	16	340M

Input Encoding

- Each example is a pair of sentences
- Add segment embedding and position embedding



Task 1 - Masked Language Model

- Estimate $p(x_i | x_{[1:i-1]}, x_{[i+1:n]})$ rather than $p(x_i | x_{[1:i-1]})$
 - Randomly mask 15% of all tokens and predict token
 - 80% of them - replace token with <mask>
 - 10% of them - replace with <random token>
 - 10% of them - replace with <token>

Alex is obnoxious but the **tutorial** is awesome.

Alex is obnoxious but the <mask> is awesome.

Alex is obnoxious but the <banana> is awesome.

Alex is obnoxious but the <tutorial> is awesome.

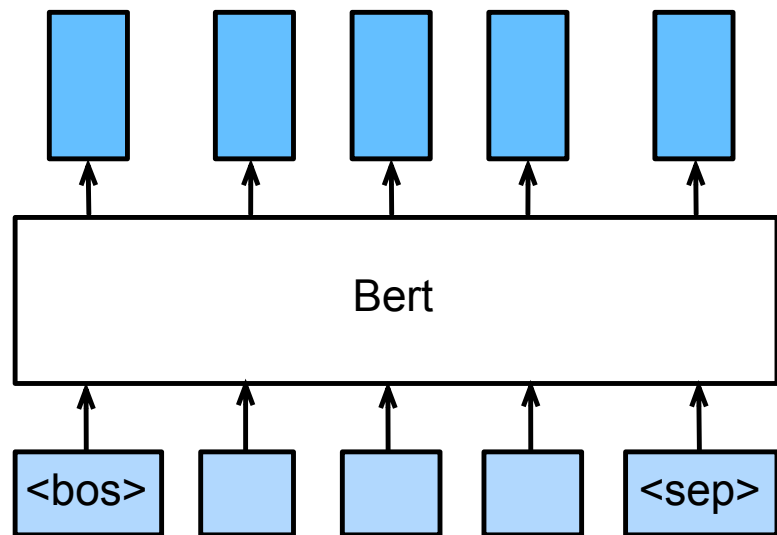
Task 2 - Next Sentence Prediction

- Predict next sentence
 - 50% of the time, replace it by random sentence
 - Feed the Transformer output into a dense layer to predict if it is a sequential pair.
- **Learn logical coherence**

<BOS> Alex is obnoxious <SEP> I don't like his shirt
<BOS> Alex is obnoxious <SEP> Look a Martian

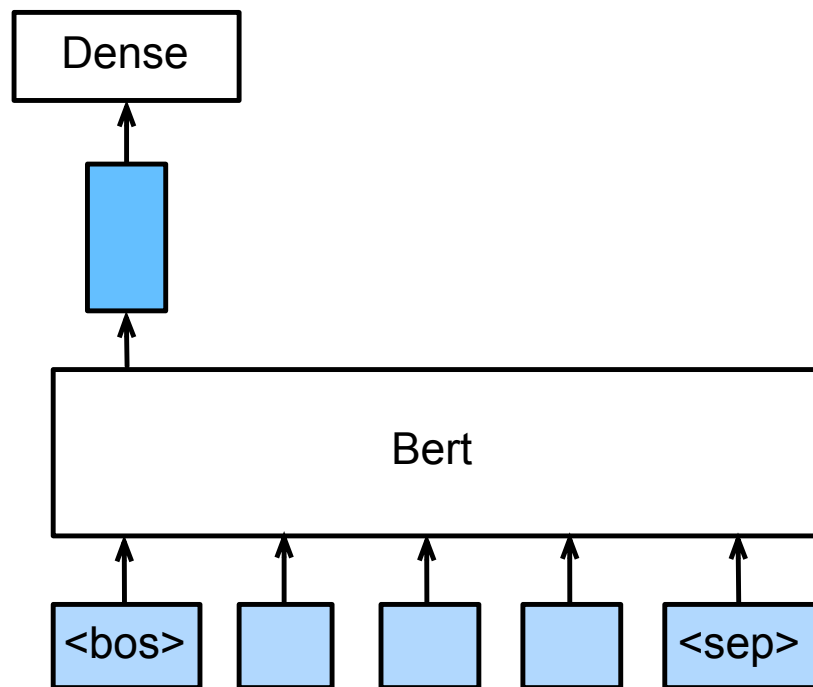
Using BERT

- BERT returns a feature vector for each token.
- Embedding captures context



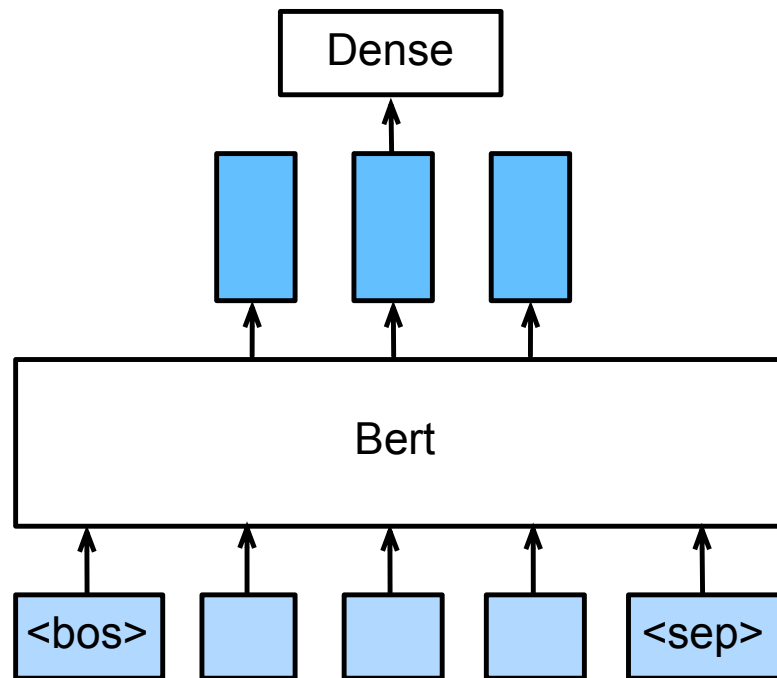
Using BERT - Sentence Classification

- BERT returns a feature vector for each token.
- Embedding captures context
- Feed <bos> embedding into dense layer
- Works for pairs, too



Using BERT - Named Entity Recognition

- BERT returns a feature vector for each token.
- Embedding captures context
- Identify if token is an entity
- Use embedding for each non-special token and classify via dense layer.

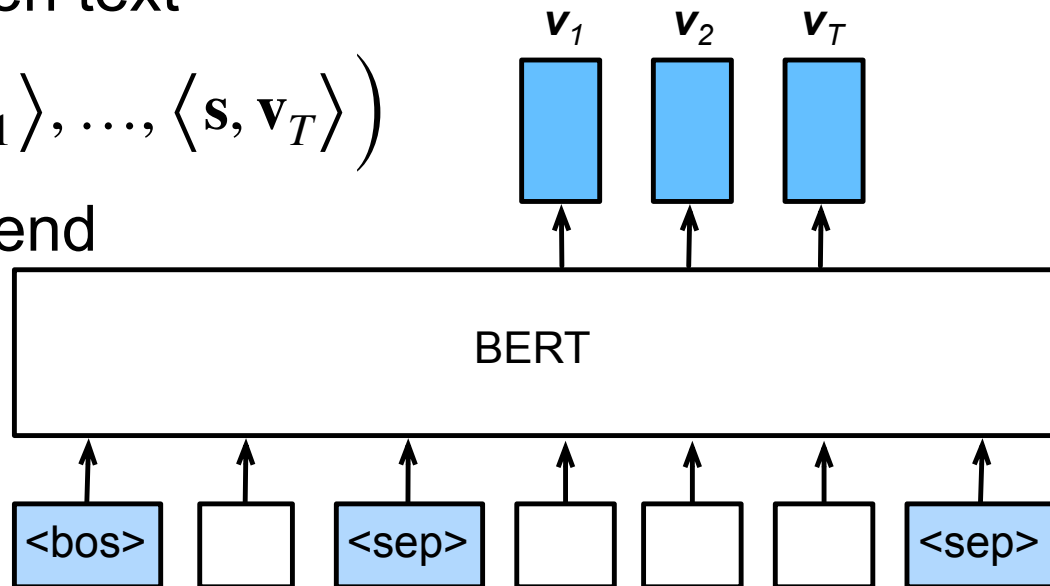


Using BERT - Question Answering

- Given question, find answer as segment of text
- Encode question first, then text

$$p_1, \dots, p_T = \text{softmax} \left(\langle \mathbf{s}, \mathbf{v}_1 \rangle, \dots, \langle \mathbf{s}, \mathbf{v}_T \rangle \right)$$

- Model sequence start & end probability for answer.



GPT2 (it gets even bigger, Radford et al., '19)

- Pretrained on 8M webpages (WebText, 40GB)
- Without fine-tuning **SOTA** on 7 language models

	blocks	hidden units	parameters
small	12	768	110M
large	24	1024	340M
GPT2	48	1600	1.5B

GPT2 Demo (gluon-nlp.mxnet.io)

```
$python sampling_demo.py --model 117M
```

Please type in the start of the sentence

```
>>> average human attention span is even shorter than that of a  
goldfish
```

```
----- Begin Sample 0 -----
```

```
average human attention span is even shorter than that of a  
goldfish strutting its way down the jaws. An estimate by the USA  
TODAY Science team of 80 human-sized models reveals that a complex  
jaw becomes a grandiose mitesaur in 100 million years, less than an  
exothermic Holocene huge sea lion, and towering 500 meters tall.
```

Similar mitesaur-sized jaws would burden as trillions

Scientists would expect a lost at least four million times as much
time in the same distances ocean as other mammals



**Sparse
Structured
Lightweight**

Heavy parameterization in multi-head attention

9. Attention Mechanism > 9.3. Transformer

In practice, we often use $p_q = p_k = p_v = d_o/h$. The hyper-parameters for a multi-head attention, feature size d_o .

```
class MultiHeadAttention(nn.Block):
    def __init__(self, units, num_heads, dropout, **kwargs): # units = d_o
        super(MultiHeadAttention, self).__init__(**kwargs)
        assert units % num_heads == 0
        self.num_heads = num_heads
        self.attention = d2l.DotProductAttention(dropout)
        self.W_q = nn.Dense(units, use_bias=False, flatten=False)
        self.W_k = nn.Dense(units, use_bias=False, flatten=False)
        self.W_v = nn.Dense(units, use_bias=False, flatten=False)

# query, key, and value shape: (batch_size, num_items, dim)
# valid_length shape is either (batch_size, ) or (batch_size, num_items)
def forward(self, query, key, value, valid_length):
    # Project and transpose from (batch_size, num_items, units) to
    # (batch_size * num_heads, num_items, p), where units = p * num_heads.
    query, key, value = [transpose_qkv(X, self.num_heads) for X in (
        self.W_q(query), self.W_k(key), self.W_v(value))]
```

parameterization of
fully connected
(dense) layers

Quaternion Transformer - 75% fewer parameters (Tay et al., '19)

Quaternion is 4D hypercomplex number

$$W = W_r + W_x \mathbf{i} + W_y \mathbf{j} + W_z \mathbf{k}$$

$$Q = r + x \mathbf{i} + y \mathbf{j} + z \mathbf{k}$$

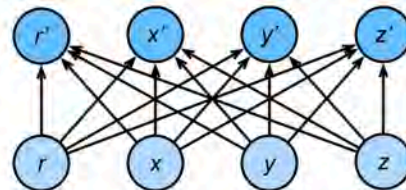
Hamilton product

$$\begin{bmatrix} W_r & -W_x & -W_y & -W_z \\ W_x & W_r & -W_z & W_y \\ W_y & W_z & W_r & -W_x \\ W_z & -W_y & W_x & W_r \end{bmatrix} \begin{bmatrix} r \\ x \\ y \\ z \end{bmatrix}$$

only 4 degrees of freedom
(16 for real-valued matrix)

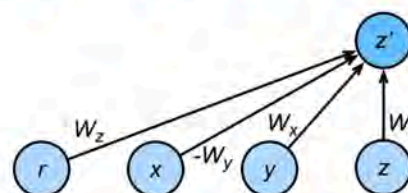
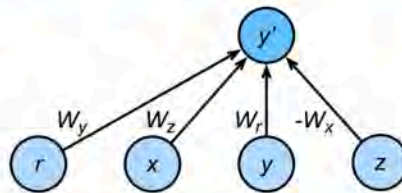
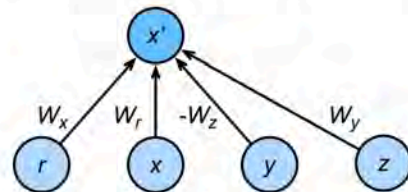
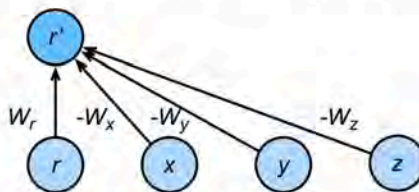
fully connected

components of the output Quaternion Q' :



components of the input Quaternion Q :

pairwise connections with weight parameter variables:



High computational cost for a long sequence

9. Attention Mechanism > 9.1. Attention Mechanism

Assume $\mathbf{Q} \in \mathbb{R}^{m \times d}$ contains m queries and $\mathbf{K} \in \mathbb{R}^{n \times d}$ has all n keys. We can compute all mn scores

$$\alpha(\mathbf{Q}, \mathbf{K}) = \mathbf{Q}\mathbf{K}^T / \sqrt{d}.$$

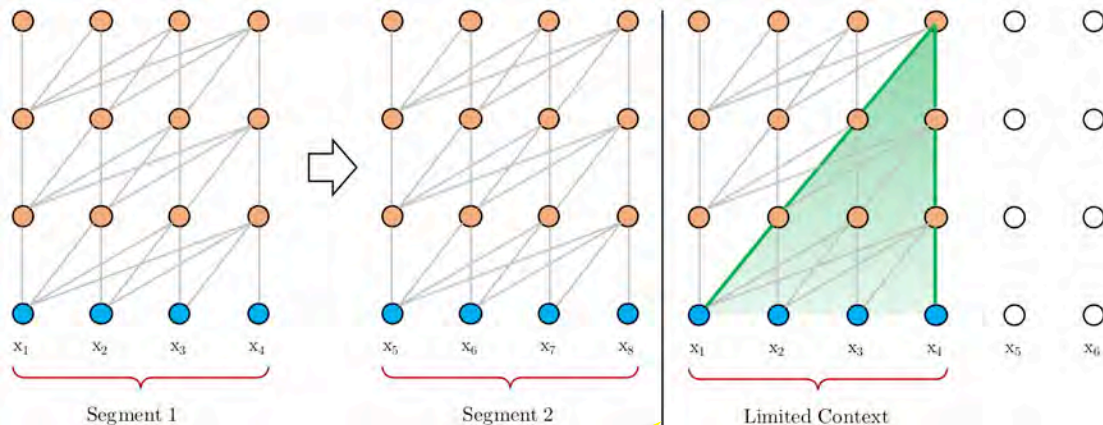
Now let's implement this layer that supports a batch of queries and key-value pairs. In addition, it supports attention weights as a regularization.

```
class DotProductAttention(nn.Block): # This class is saved in d2l.
    def __init__(self, dropout, **kwargs):
        super(DotProductAttention, self).__init__(**kwargs)
        self.dropout = nn.Dropout(dropout)

    # query: (batch_size, #queries, d)
    # key: (batch_size, #kv_pairs, d)
    # value: (batch_size, #kv_pairs, dim_v)
    # valid_length: either (batch_size, ) or (batch_size, xx)
    def forward(self, query, key, value, valid_length=None):
        d = query.shape[-1]
        # set transpose b=True to swap the last two dimensions of key
        scores = nd.batch_dot(query, key, transpose_b=True) / math.sqrt(d)
        attention_weights = self.dropout(masked_softmax(scores, valid_length))
        return nd.batch_dot(attention_weights, value)
```

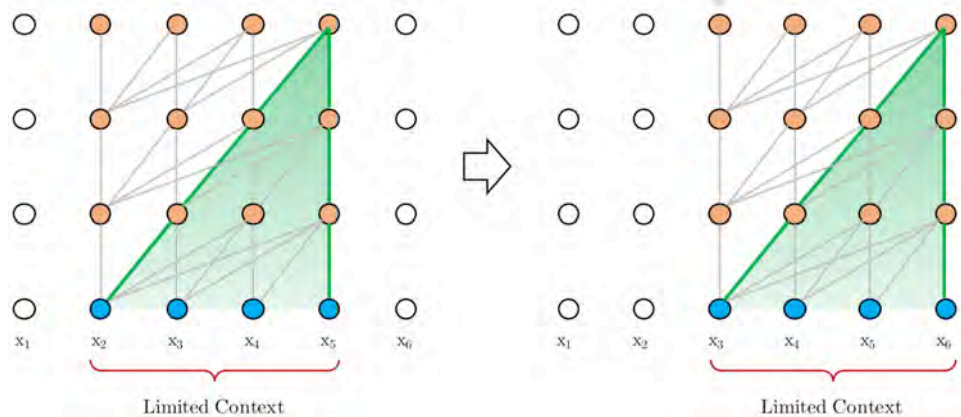
$O(n^2d)$ in self attention
(sequence length n)
(hidden size d)

Structured attention on long sequences (Al-Rfou et al., '18)

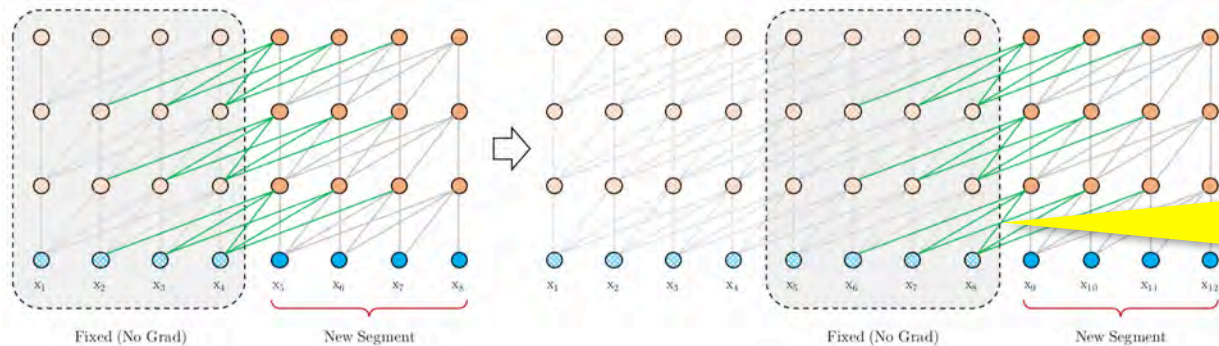


training (context fragmentation)

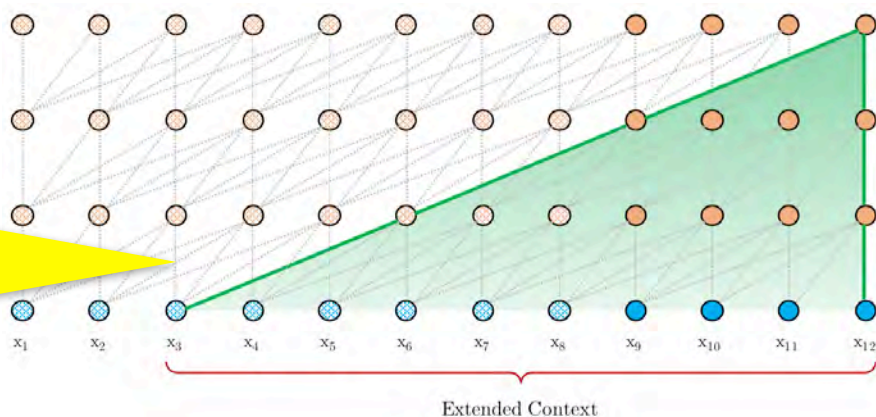
testing (segment is shifted by 1 position then evaluated)



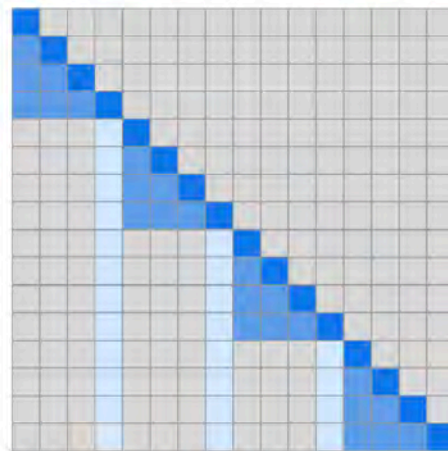
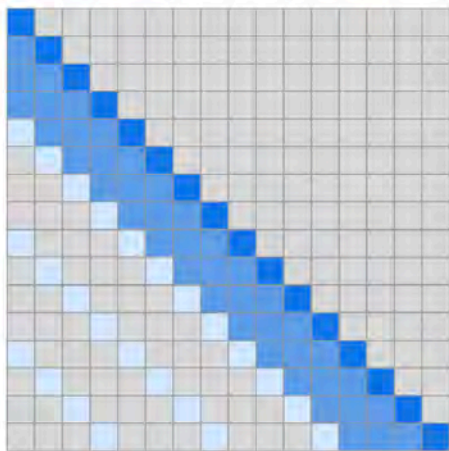
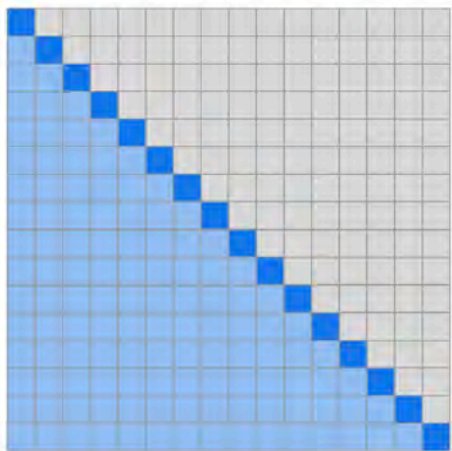
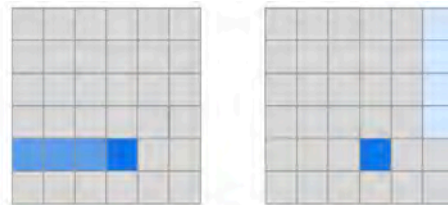
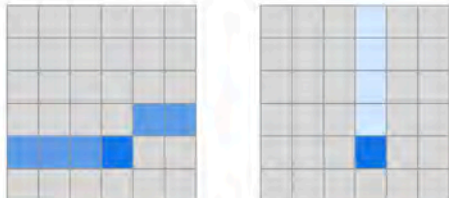
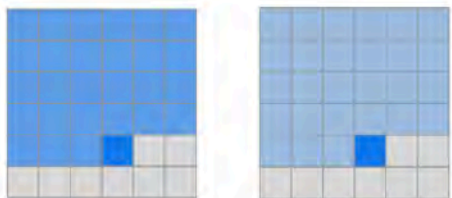
Transformer-XL with recurrence (Dai et al., '19)



testing - reuse previous segments (like in RNN)



Sparse Transformer (Child et al., '19)



Transformer

strided
(for images)

fixed
(for text)

Open Questions

- **Theory**

- Function complexity
(design complex function via simple attention mechanism)
- Convergence analysis for mechanism vs. parameters
(similar to Watson-Nadaraya estimator)
- Regularization

- **Interpretation**

- Attention vs. meaning
(e.g. Hewitt & Manning, '19; Coenen et al., '19 for BERT)
- Multiple steps of reasoning
Can we guide it? Structure it? Can we learn from it?

Open Questions

- **Large State Spaces**

- Factorizing space
(design automatically rather than manually per head)
- Pseudorandom dense (beyond quaternions)
- Learn sparse structure (transfer for attention?)

- **Computation**

- Avoid computation when no attention
- Memory footprint

- **Low Hanging Fruit**

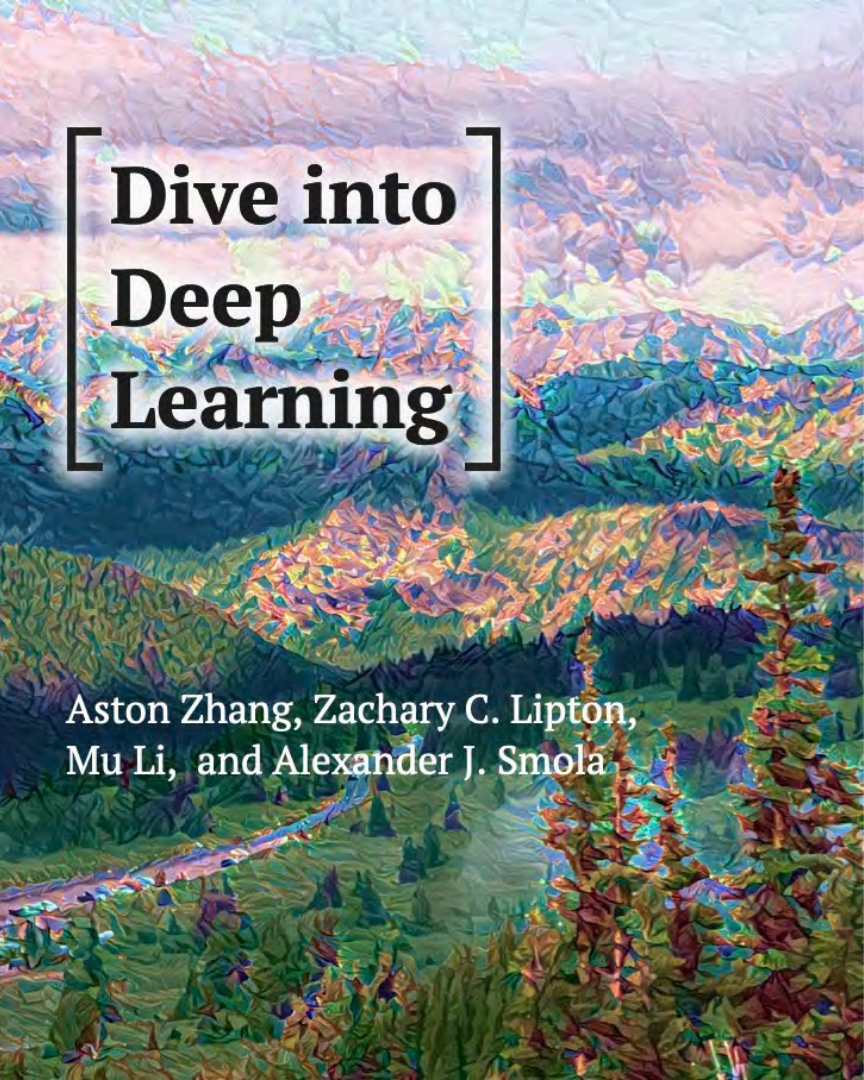
Rewrite papers with attention / Transformers / BERT



Dive into Deep Learning

Aston Zhang, Zachary C. Lipton,
Mu Li, and Alexander J. Smola

6. Resources



Dive into Deep Learning

Aston Zhang, Zachary C. Lipton,
Mu Li, and Alexander J. Smola

- Self-contained tutorials
- Statistics, linear algebra, optimization
- Machine learning basics
- 150+ Jupyter Notebooks, fully executed
- GPU and parallel examples
- Ready to use for applications
- Teachable content
- Adopted as a textbook or reference book at Berkeley, CMU, UCLA, UIUC, Gatech, Shanghai Jiao Tong, Zhejiang U, USTC
- Slides, videos from Berkeley class courses.d2l.ai
- Multilingual content EN, ZH (in progress: KO, JA, FR, TR)

One Code - Multiple Formats & Devices

22:30

d2l.ai

7.1. Deep Convolutional ...

7.1.2.3. Capacity Control and Preprocessing

AlexNet controls the model complexity of the fully-connected layer by dropout ([Section 4.6](#)), while LeNet only uses weight decay. To augment the data even further, the training loop of AlexNet added a great deal of image augmentation, such as flipping, clipping, and color changes. This makes the model more robust and the larger sample size effectively reduces overfitting. We will discuss data augmentation in greater detail in [Section 12.1](#).

```
import sys
sys.path.insert(0, '..')

import d2l
from mxnet import gluon, init, nd
from mxnet.gluon import data as gdata, nn
import os
import sys

net = nn.Sequential()
```

Mobile
friendly

Jupyter
Notebook

localhost:8888/notebooks/chapter_convolutional-mode... jupyter alexnet (unsaved changes) Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted Kernel

Activation Functions

Second, AlexNet changed the sigmoid activation function to a simpler ReLU activation function. On the one hand, the computation of the ReLU activation function is simpler. For example, it does not have the exponentiation operation found in the sigmoid activation function. On the other hand, the ReLU activation function makes model training easier when using different parameter initialization methods. This is because, when the output of the sigmoid activation function is very close to 0 or 1, the gradient of these regions is almost 0, so that back propagation cannot continue to update some of the model parameters. In contrast, the gradient of the ReLU activation function in the positive interval is always 1. Therefore, if the model parameters are not properly initialized, the sigmoid function may obtain a gradient of almost 0 in the positive interval, so that the model cannot be effectively trained.

Capacity Control and Preprocessing

AlexNet controls the model complexity of the fully-connected layer by dropout ([:numref: chapter_dropout](#)), while LeNet only uses weight decay. To augment the data even further, the training loop of AlexNet added a great deal of image augmentation, such as flipping, clipping, and color changes. This makes the model more robust and the larger sample size effectively reduces overfitting. We will discuss data augmentation in greater detail in [:numref: chapter_image_augmentation](#).

```
In [1]: import sys
sys.path.insert(0, '..')

import d2l
from mxnet import gluon, init, nd
from mxnet.gluon import data as gdata, nn
import os
import sys

net = nn.Sequential()
# Here, we use a larger 11 x 11 window to capture objects. At the same time,
# we use a stride of 4 to greatly reduce the height and width of the output.
# Here, the number of output channels is much larger than that in LeNet
net.add(nn.Conv2D(96, kernel_size=11, strides=4, activation='relu'),
        nn.MaxPool2D(pool_size=3, strides=2),
        # Make the convolution window smaller, set padding to 2 for consistent
        # height and width across the input and output, and increase the
```

Open Source

Active
Development

The screenshot shows the GitHub repository page for `d2l-ai/d2l-en`. The repository has 1,669 stars and 434 forks. It contains 1,173 commits, 2 branches, 2 releases, and 59 contributors. The commit history is visible, showing recent updates to various chapters and the preface.

Commit	Message	Time
astonzhang	Update preface.md	Latest commit d5011a4 7 hours ago
	chapter_appendix	Fix some warnings, improve PDF 9 days ago
	chapter_attention-mechanism	D2lbook (#265) 6 days ago
	chapter_computational-performance	D2lbook (#265) 6 days ago
	chapter_computer-vision	D2lbook (#265) 6 days ago
	chapter_convolutional-modern	resolved conflict in batch norm re numref 12 hours ago
	chapter_convolutional-neural-networ...	D2lbook (#265) 6 days ago
	chapter_crashcourse	Fix some warnings, improve PDF 9 days ago
	chapter_deep-learning-computation	Remove repetition (#277) 7 hours ago
	chapter_install	Merge branch 'master' into master 5 days ago
	chapter_introduction	Updating some descriptions in Introduction Chapter (#263) 6 days ago
	chapter_linear-networks	Update softmax-regression-scratch.md 4 days ago

PDF

The screenshot shows the PDF viewer for the book "Dive into Deep Learning, Release 0.7" by Aston Zhang, Zack C. Lipton, Mu Li, and Alex J. Smola. The page number 1/614 is visible in the top right corner.

Dive into Deep Learning
Release 0.7

Aston Zhang, Zack C. Lipton, Mu Li, Alex J. Smola

<https://d2l.ai>

Syllabus

Assignments

Projects

Units

FAQ

Date	Topics
1/22	Logistics, Software, Linear Algebra
1/24	Probability and Statistics (Bayes Rule, Sampling Naive Bayes, Sampling)
1/29	Gradients, Chain Rule, Automatic differentiation
1/31	Linear Regression, Basic Optimization
2/5	Likelihood, Loss Functions, Logistic Regression, Information Theory
2/7	Multilayer Perceptron
2/12	Model Selection, Weight Decay, Dropout
2/14	Numerical Stability, Hardware
2/19	Environment
2/21	Layers, Parameters, GPUs
2/26	Convolutional Layers

UC Berkeley
Spring '19

120+ Videos on YouTube (+20 slide decks)

- Home
- Trending
- Subscriptions
- Originals
- Library



▶ PLAY ALL

Deep Learning UC Berkeley STAT-157 2019

128 videos • 17,182 views • Updated 4 days ago

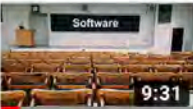


Alex Smola

EDIT

1  **L1/1 Logistics**
Alex Smola Added by Alex Smola

2  **L1/2 Deep Learning Overview**
Alex Smola Added by Alex Smola

3  **L1/3 Software**
Alex Smola Added by Alex Smola

4  **L1/4 Linear Algebra**
Alex Smola Added by Alex Smola

5  **L1/5 Linear Algebra in Jupyter**
Alex Smola Added by Alex Smola



Dive into Deep Learning

Aston Zhang, Zachary C. Lipton,
Mu Li, and Alexander J. Smola

gluon-cv.mxnet.io
Computer Vision

gluon-nlp.mxnet.io
Natural Language

gluon-ts.mxnet.io
Time Series Prediction

tvm.ai
Deep Learning
Compiler

mxnet.io
Imperative & Symbolic

dgl.ai
Deep Learning on
Graphs

d2l.ai



References

- Zaheer, Manzil, et al. "Deep sets." *Advances in neural information processing systems*. 2017.
- Ilse, Maximilian, Jakub M. Tomczak, and Max Welling. "Attention-based deep multiple instance learning." *arXiv preprint arXiv:1802.04712* (2018).
- Salton, Gerard, and Michael J. McGill. "Introduction to modern information retrieval." (1986).
- Mikolov, Tomas, et al. "Distributed representations of words and phrases and their compositionality." *Advances in neural information processing systems*. 2013.
- Wang, Yequan, Minlie Huang, and Li Zhao. "Attention-based LSTM for aspect-level sentiment classification." *Proceedings of the 2016 conference on empirical methods in natural language processing*. 2016.
- Yang, Zichao, et al. "Hierarchical attention networks for document classification." *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 2016.
- Hu, Jie, Li Shen, and Gang Sun. "Squeeze-and-excitation networks." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018.
- Veličković, Petar, et al. "Graph attention networks." *arXiv preprint arXiv:1710.10903* (2017).
- Sukhbaatar, Sainbayar, Jason Weston, and Rob Fergus. "End-to-end memory networks." *Advances in neural information processing systems*. 2015.
- Yang, Zichao, et al. "Stacked attention networks for image question answering." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.
- Sutskever, Ilya, Oriol Vinyals, and Quoc V. Le. "Sequence to sequence learning with neural networks." *Advances in neural information processing systems*. 2014.
- Tay et al. "Lightweight and Efficient Neural Natural Language Processing with Quaternion Networks", Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL), 2019

References

- Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate." *arXiv preprint arXiv:1409.0473* (2014).
- Luong, Minh-Thang, Hieu Pham, and Christopher D. Manning. "Effective approaches to attention-based neural machine translation." *arXiv preprint arXiv:1508.04025* (2015).
- Vinyals, Oriol, Meire Fortunato, and Navdeep Jaitly. "Pointer networks." *Advances in Neural Information Processing Systems*. 2015.
- Graves, Alex, Greg Wayne, and Ivo Danihelka. "Neural turing machines." *arXiv preprint arXiv:1410.5401* (2014).
- Vaswani, Ashish, et al. "Attention is all you need." *Advances in neural information processing systems*. 2017.
- Zhang et al. Co-occurrent Features in Semantic Segmentation. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019
- Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." *arXiv preprint arXiv:1810.04805* (2018).
- Radford, Alec, et al. "Improving language understanding by generative pre-training." URL https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/languageunsupervised/language_understanding_paper.pdf (2018).
- Radford, Alec, et al. "Language models are unsupervised multitask learners." *OpenAI Blog* 1.8 (2019).
- Al-Rfou, Rami, et al. "Character-level language modeling with deeper self-attention." *arXiv preprint arXiv:1808.04444* (2018).
- Dai, Zihang, et al. "Transformer-xl: Attentive language models beyond a fixed-length context." *arXiv preprint arXiv:1901.02860* (2019).
- Child, Rewon, et al. "Generating long sequences with sparse transformers." *arXiv preprint arXiv:1904.10509* (2019).