Hertie School

Deep Learning

Session 8: Natural language processing (2): Encoder-decoder architecture and attention mechanism

November 6, 2025

This session will be recorded for in-class use.

Logistics

Timeline

- **No class next week**
- PS1 grades out now
- PS3 will likely be out on Nov 7 (optional problem set)
    - Self-select into groups of 3-4
- Any questions on the tutorial (Assignment 3)?

Guest Lecture

Claudius Wehner
Data Scientist
AI Lab at German Environmental Agency (UBA)

Syllabus

Hertie School

Deep learning theory

1. Deep learning in public policy
2. Deep neural networks (1)
3. Deep neural networks (2)

Applications

4. Computer vision (1): Convolutional neural networks
5. Computer vision (2): Modern CNNs, implementation, and applications
6. Sequence methods and time-series analysis

- Midterm exam -

7. Natural language processing (1)
8. **Natural language processing (2): Transformer models, implementation, and applications**
9. Further topics in deep learning

Deep learning and public policy

10. Policy approaches to regulate deep learning and responsible implementation in government
11. Deep learning in practice
12. Tutorial presentations

Learning objectives

Hertie School

Natural language processing (2): Encoder-decoder architecture and attention mechanism

- Encoder - decoder
- Machine translation
- Attention mechanism
- Transformer architecture
- Pretrained models BERT and ViT

Hertie School

Encoder – decoder: Machine translation

Zhang et al. Chapter 10

Machine translation

Overview

- Model is presented with a sentence in one language and must predict the sentence in another language
- Sentences may be of different lengths
- Corresponding words in the two sentences may not occur in the same order:
    - I would like to learn German.
    - Ich möchte Deutsch lernen.
- **Sequence-to-sequence** (seq2seq) problems: mapping between two "unaligned" sequences
- Also question-answer pairs or dialogue are seq2seq problems

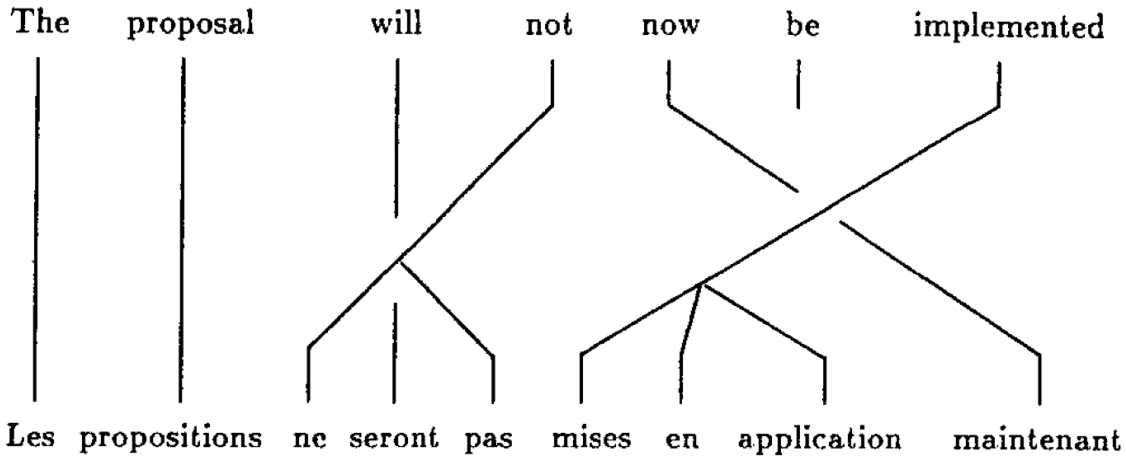


Figure 3 Alignment Example.

Image: Brown et al. (1990)

Machine translation

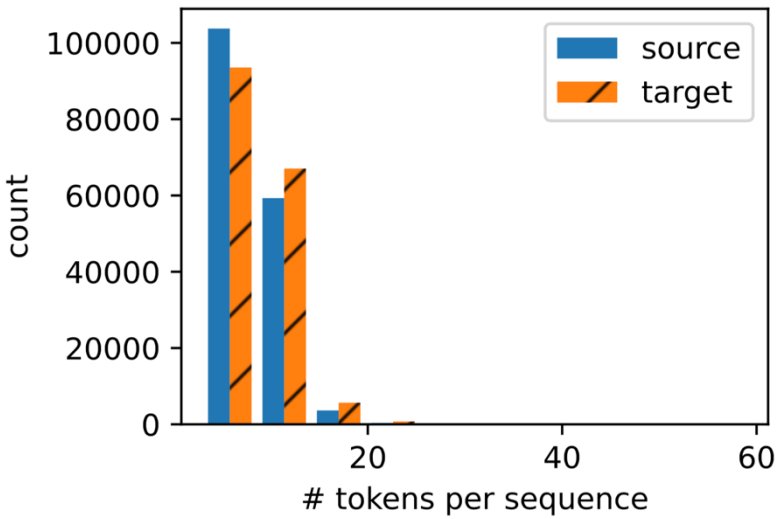
Approach

- In a machine translation problem where English is translated into French, English is called the **source language** and French is called the **target language**.
- Data downloadable for example from Tatoeba project
- Available for many languages (e.g., Malayalam, Polish, Tagalog) mapped to English
- Typically, word-tokenization is used
- Process minibatch of text sequences at one time:
  - by **truncation** (only take fixed number of tokens and discard the rest)
  - and **padding** (adding "<pad>" to end of sequence)

How the Data Looks

English + TAB + The Other Language + TAB + Attribution

```
This work isn't easy.   この仕事は簡単じゃない。     CC-BY 2.0 (France) A
ttribution: tatoeba.org #3737550 (CK) & #7977622 (Ninja)
Those are sunflowers.   それはひまわりです。         CC-BY 2.0 (France) A
ttribution: tatoeba.org #441940 (CK) & #205407 (arnab)
Tom bought a new car.   トムは新車を買った。         CC-BY 2.0 (France) A
ttribution: tatoeba.org #1026984 (CK) & #2733633 (tommy_san)
This watch is broken.   この時計は壊れている。       CC-BY 2.0 (France) A
ttribution: tatoeba.org #58929 (CK) & #221604 (bunbuku)
```



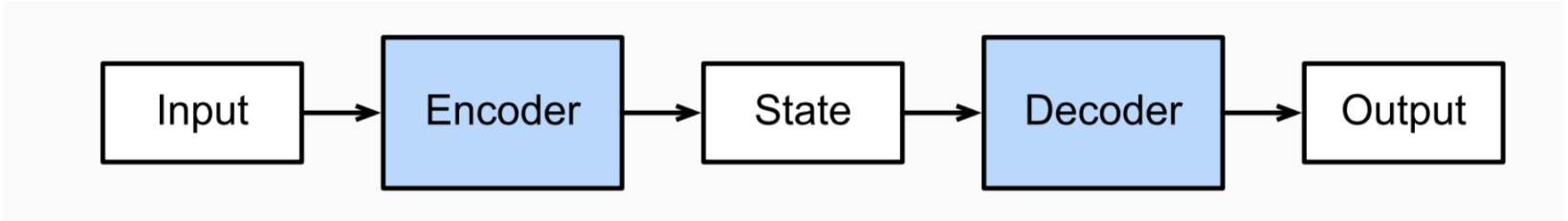
English to French dataset

Image: Anki (top), Zhang et al (bottom)

Encoder-decoder architecture

Deep learning approach to seq2seq problems

- Encoder-decoder architecture is standard approach to seq2seq problems like machine translation
- 2 components:
  - The encoder takes a variable-length sequence as input and transforms it into a state with a fixed shape.
  - The decoder maps the encoded state of a fixed shape to a variable-length sequence.



I would like to learn German.

Ich möchte Deutsch lernen.

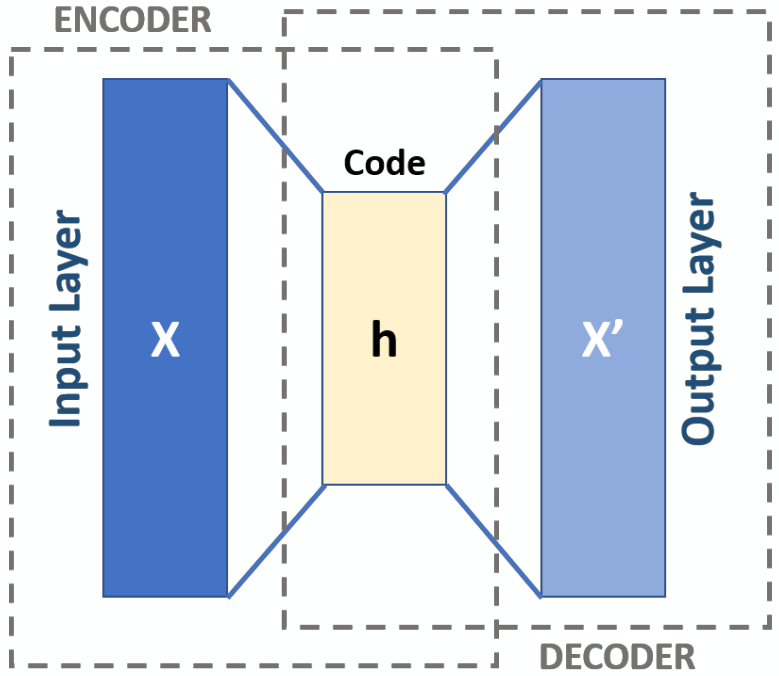
Image: Zhang et al.

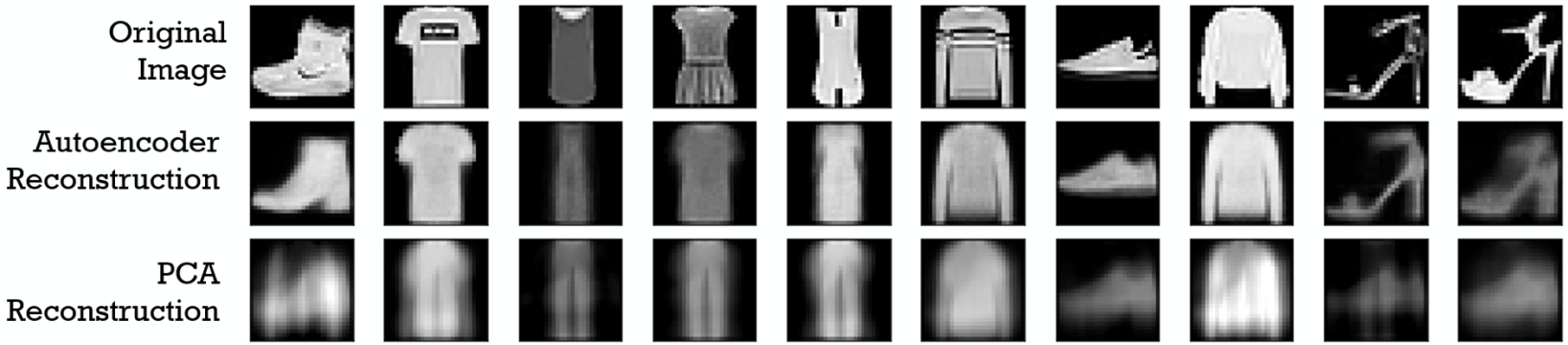
Hertie School

Encoder-decoder architecture

Autoencoder

- Encoder-decoder where source and target domain are the same
- Special case of encoder-decoder models
- Learns efficient encodings of the data (unsupervised)
- Task: Regenerate the input data
- Regularized autoencoders: learning representations for subsequent classification tasks
- Variational autoencoders: Generative models
- Used for dimensionality reduction (e.g. denoising data), anomaly detection, image compression, etc.





Images: Wikipedia

Hertie School

Encoder-Decoder with RNN

Encoder-Decoder with RNN

- Vanilla model for encoder-decoder seq2seq (e.g., for machine translation)
- **Encoder**: encoder RNN will take a variable-length sequence as input and transform it into a fixed-shape hidden state
- **Decoder**: RNN that will predict each successive target token given both the input sequence and the preceding tokens in the output
- Ignoring the encoder, the decoder in a seq2seq architecture behaves just like a normal language model

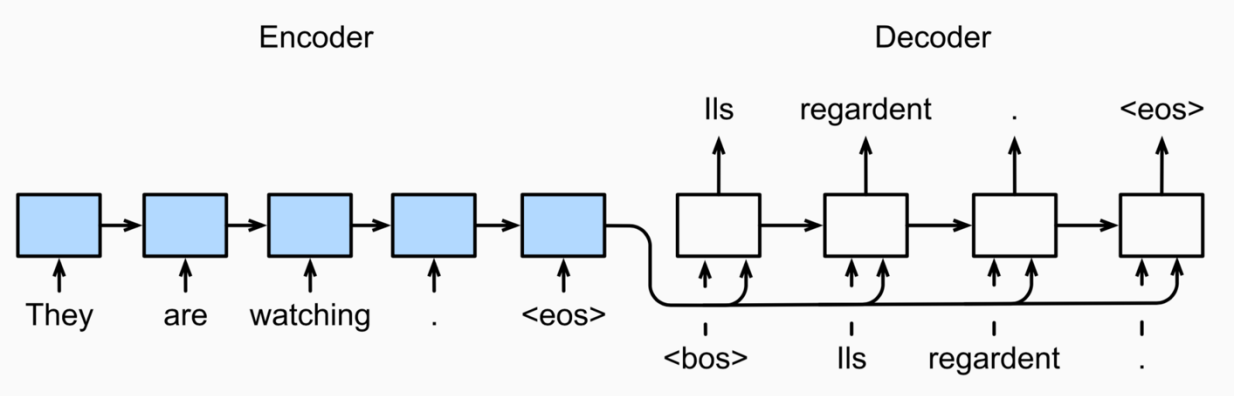


Image: Zhang et al.

Hertie School

Encoder-Decoder with RNN

Encoder

Encoder RNN takes a variable-length sequence as input and transform it into a fixed-shape hidden state:

Given input sequence of tokens $x_1, \ldots, x_T$, for time step t the encoder's recurrent layer is $\boldsymbol{h}_t = f(\boldsymbol{x}_t, \boldsymbol{h}_{t-1})$.

Encoder transforms the hidden state into a **context variable** used by the decoder through some customized function q of the hidden states:

$$\boldsymbol{c} = q(\boldsymbol{h}_1, \ldots, \boldsymbol{h}_T)$$

Example displayed below is taking just the last hidden state: $\boldsymbol{c} = \boldsymbol{h}_T$

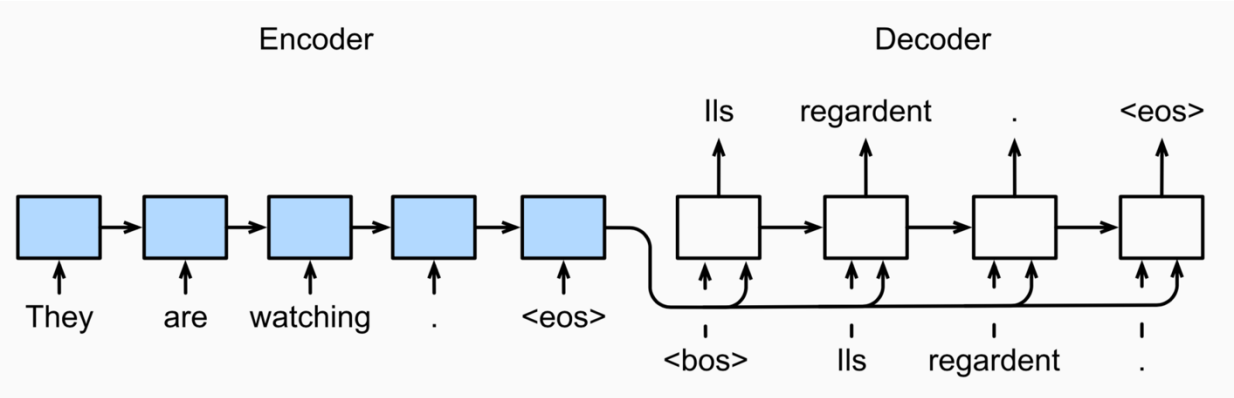


Image: Zhang et al.

Hertie School

Encoder-Decoder with RNN

Decoder

Decoder: RNN that will predict each successive target token given both the input sequence and the preceding tokens in the output:
Given taget output sequence $y_1, y_2, \ldots, y_T$, with time step t', and context variable $\boldsymbol{c}$, decoder assigns a probability to each possible token to occur at $t' + 1$

$$P(y_{t'+1}|y_1, \ldots, y_{t'}, \boldsymbol{c})$$

Decoder hidden state:

$$\boldsymbol{s}_{t'} = g(y_{t'-1}, \boldsymbol{c}, \boldsymbol{s}_{t'-1})$$

We can use an output layer and softmax to compute the predictive distribution $P(y_{t'+1}|y_1, \ldots, y_{t'}, \boldsymbol{c})$ over output token at $t' + 1$.

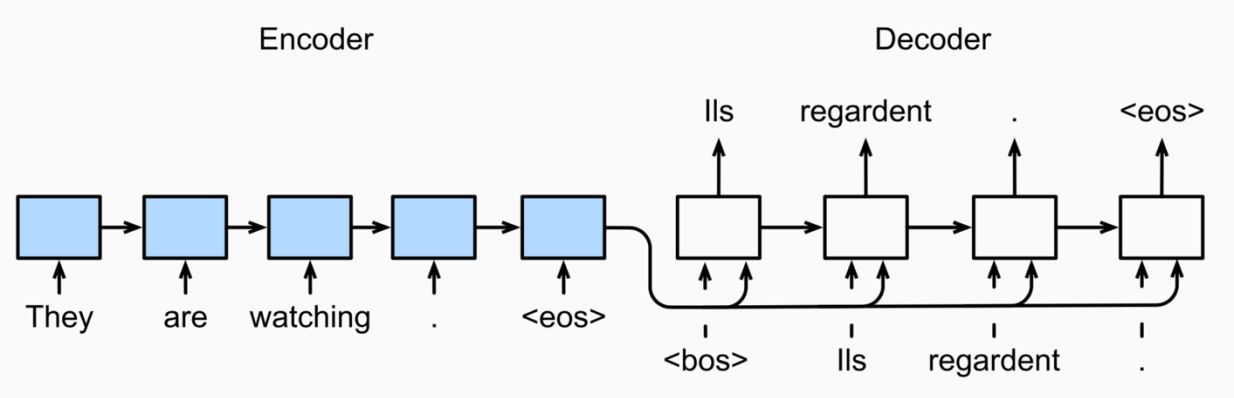


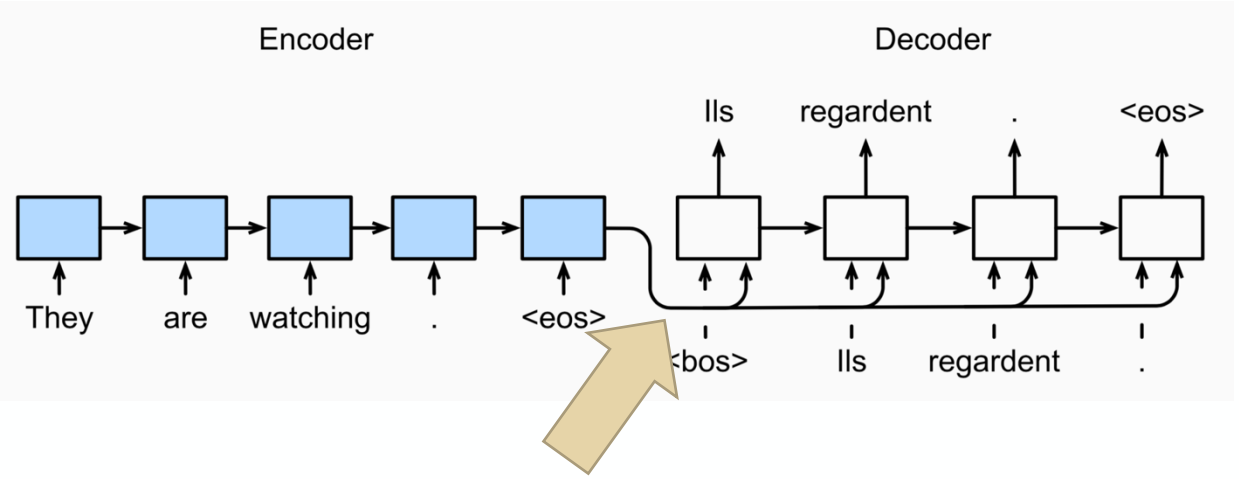
Image: Zhang et al.

Hertie School

Encoder-Decoder with RNN

Full model

- We directly use the hidden state at the final time step of the encoder to initialize the hidden state of the decoder
- To incorporate the encoded input sequence information, the context variable $\boldsymbol{c}$ is concatenated with the decoder input at all the time steps
- To predict the probability distribution of the output token, we use a fully connected layer to transform the hidden state at the final layer of the RNN decoder.



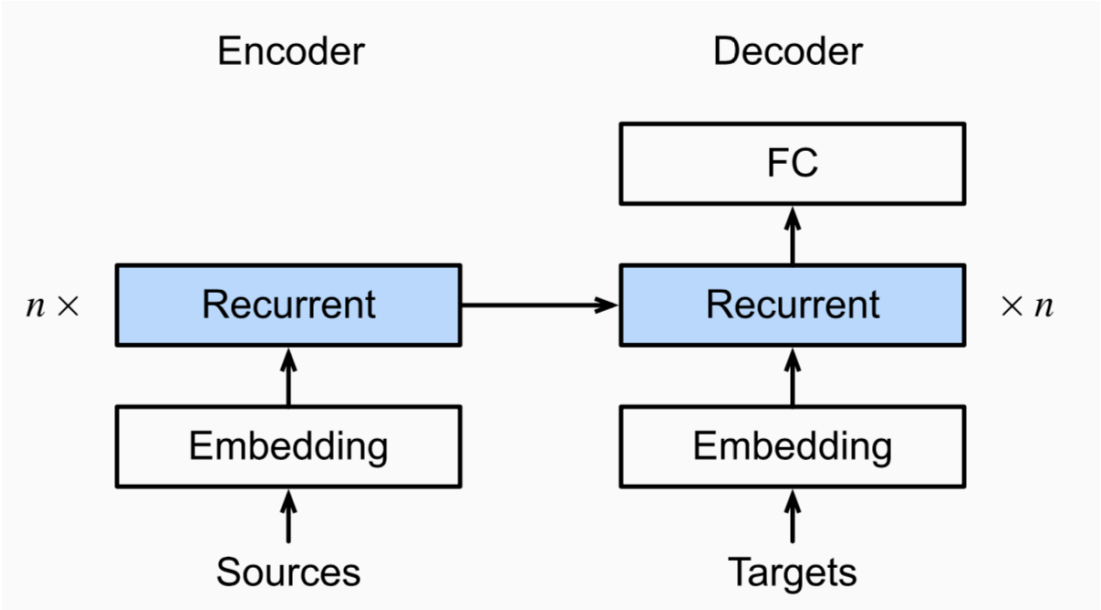


Image: Zhang et al.

Hertie School

Encoder-Decoder with RNN

Inference

- During training, in the decoder the ground truth of the previous time step is fed into the next time step as an input
- During inference, to predict the output sequence at each step, the predicted token from the previous time step is used
- Whichever token the decoder has assigned the highest probability is the prediction at each step

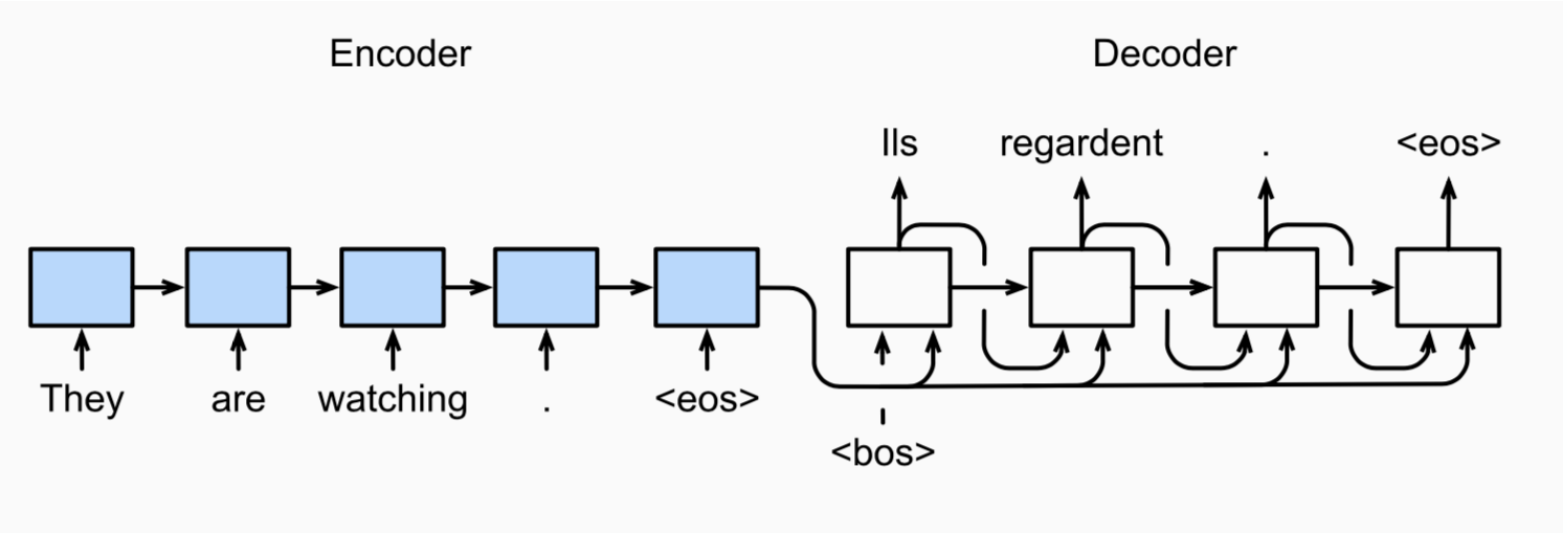


Image: Zhang et al.

Evaluation metric

BLEU (Bilingual Evaluation Understudy)

- Compare the predicted sequence with the ground truth sequence
- For any n-grams in the predicted sequence, BLEU evaluates whether this n-gram appears in the target sequence.
- **Precision** of n-grams p_n is the ratio of the number of matched n-grams in the predicted and target sequence to the number of n-grams in the predicted sequence
- Example:

  Predicted sequence A, B, B, C, D

  Target sequence A, B, C, D, E, F

  What is the precision p_1?

  What about p_2?

Evaluation metric

BLEU (Bilingual Evaluation Understudy)

- BLUE is defined as

$$\exp\left(\min\left(0, 1 - \frac{len_{label}}{len_{pred}}\right)\right) \prod_{n=1}^{k} p_n^{1/2^n}$$

where len_{label} is # tokens in target sequence, len_{pred} is # tokens in predicted sequence, and k is the longest n-gram that can be matched

- BLUE is 1 if predicted sequence is same as target sequence
  - In that case: $p_n = 1$ (full match) and $len_{label} = len_{pred}$
- BLUE assigns higher weights to larger n-grams (which are harder to match)

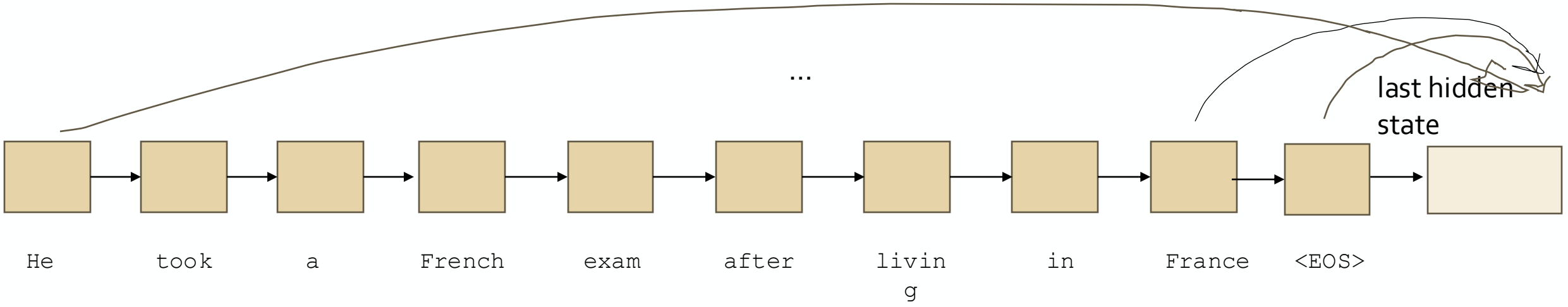
Hertie School

Attention mechanism

Zhang et al. Chapter 11

Long sequences

He took at French exam after living in France.



Hertie School

Attention

Attention in biology

- Our optical nerve receives massive sensory input, far exceeding what the brain can fully process
- We can concentrate on objects of interest, a small fraction of the information, freeing up more capacity
- This attention can be directed cognitively and volitionally
- Attention is a scarce resource: your attention is being paid with an opportunity cost.
- Attention mechanism is powerful approach in deep learning

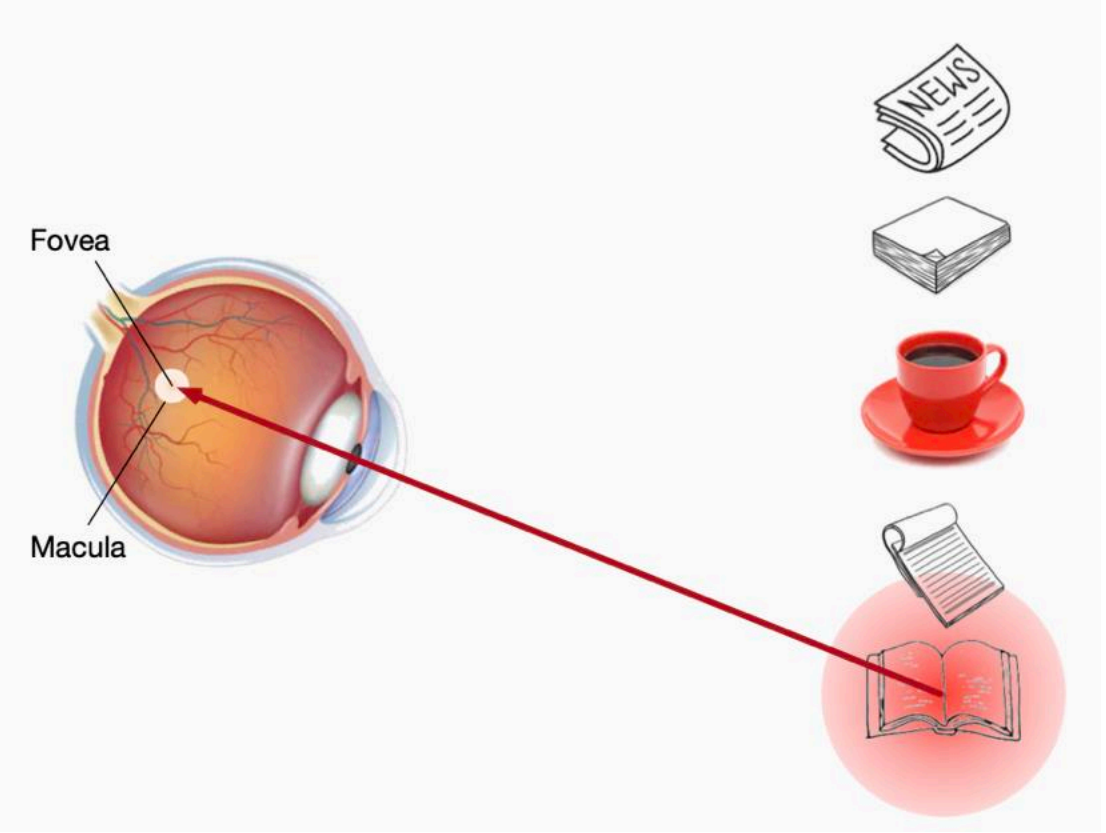


Image: Zhang et al.

Hertie School

Attention

Attention used in deep learning

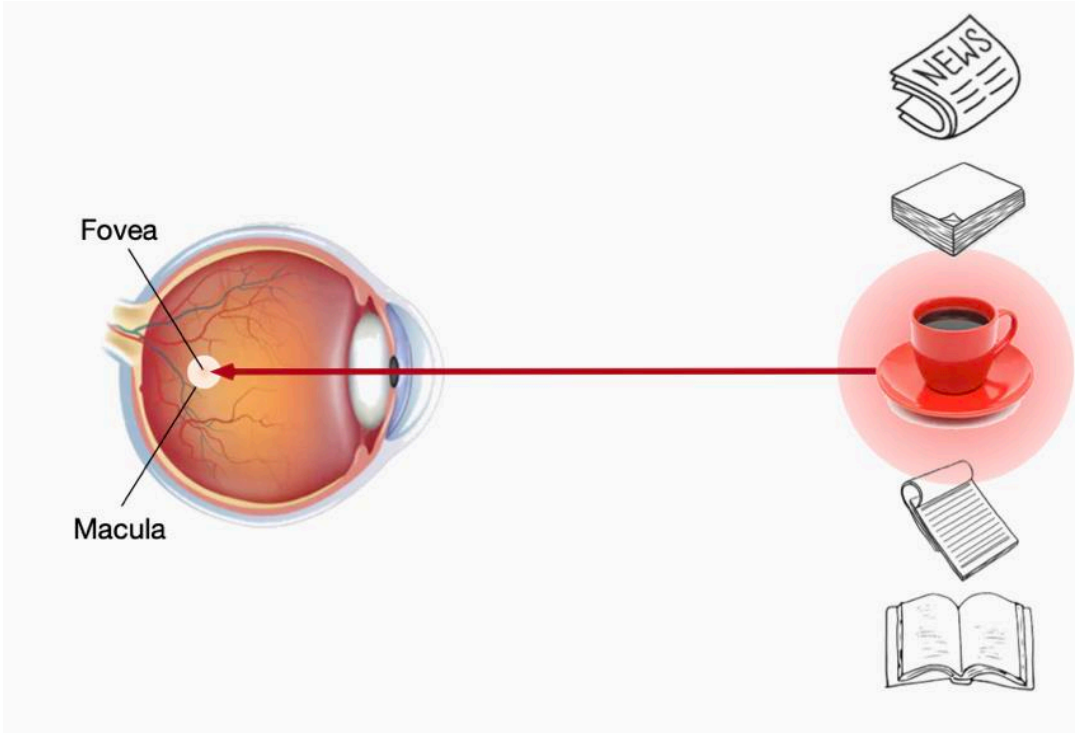
- Transformer models are built on attention mechanism
- Superior scalability of transformers make them widely used in large-scale pre-training
- High performance of large-scale pretraining with transformers in areas as diverse as language, vision, speech, and reinforcement learning that benefits from larger models, more training data, and more compute
- NLP: State-of the art language models are almost all transformers ("large language models (LLMs)")
- CV: When training very large models on very large datasets (e.g., 300 million images), vision transformers outperform ResNets significantly in image classification
- Different tasks: encoder-only (e.g., BERT), encoder-decoder (e.g., T5), and decoder-only (e.g., GPT series)

Hertie School

Attention cues

Nonvolitional cue

Based on the saliency and conspicuity of objects in the environment



Volitional cue

Based on variable selection criteria, more deliberate

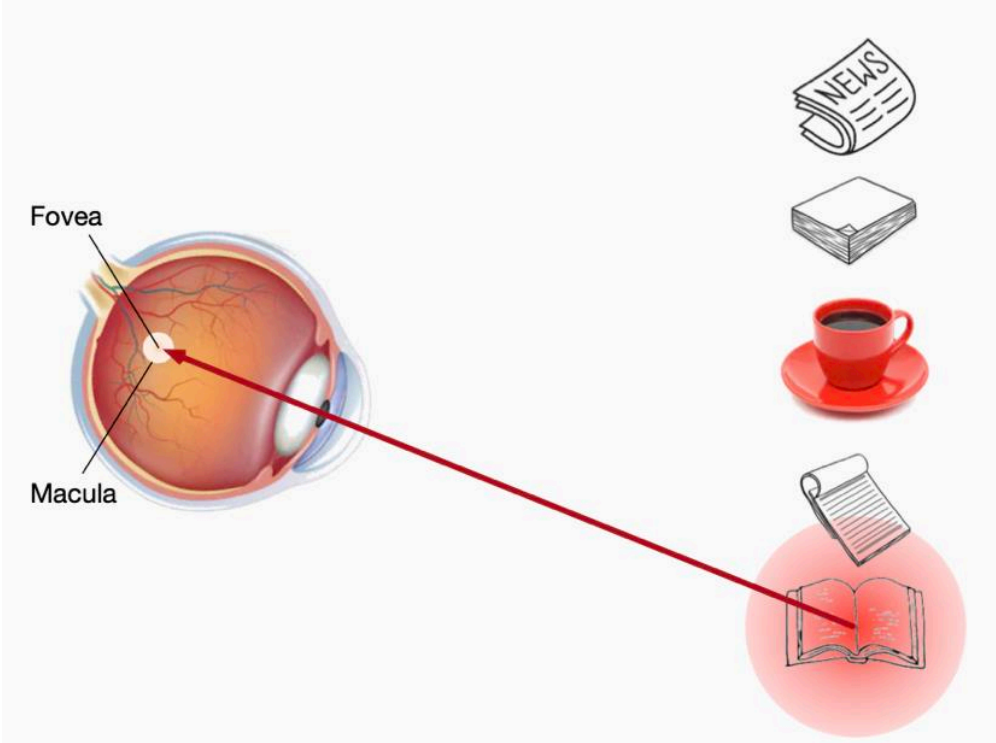


Image: Zhang et al.

Hertie School

Attention cues

Attention in deep learning: Queries

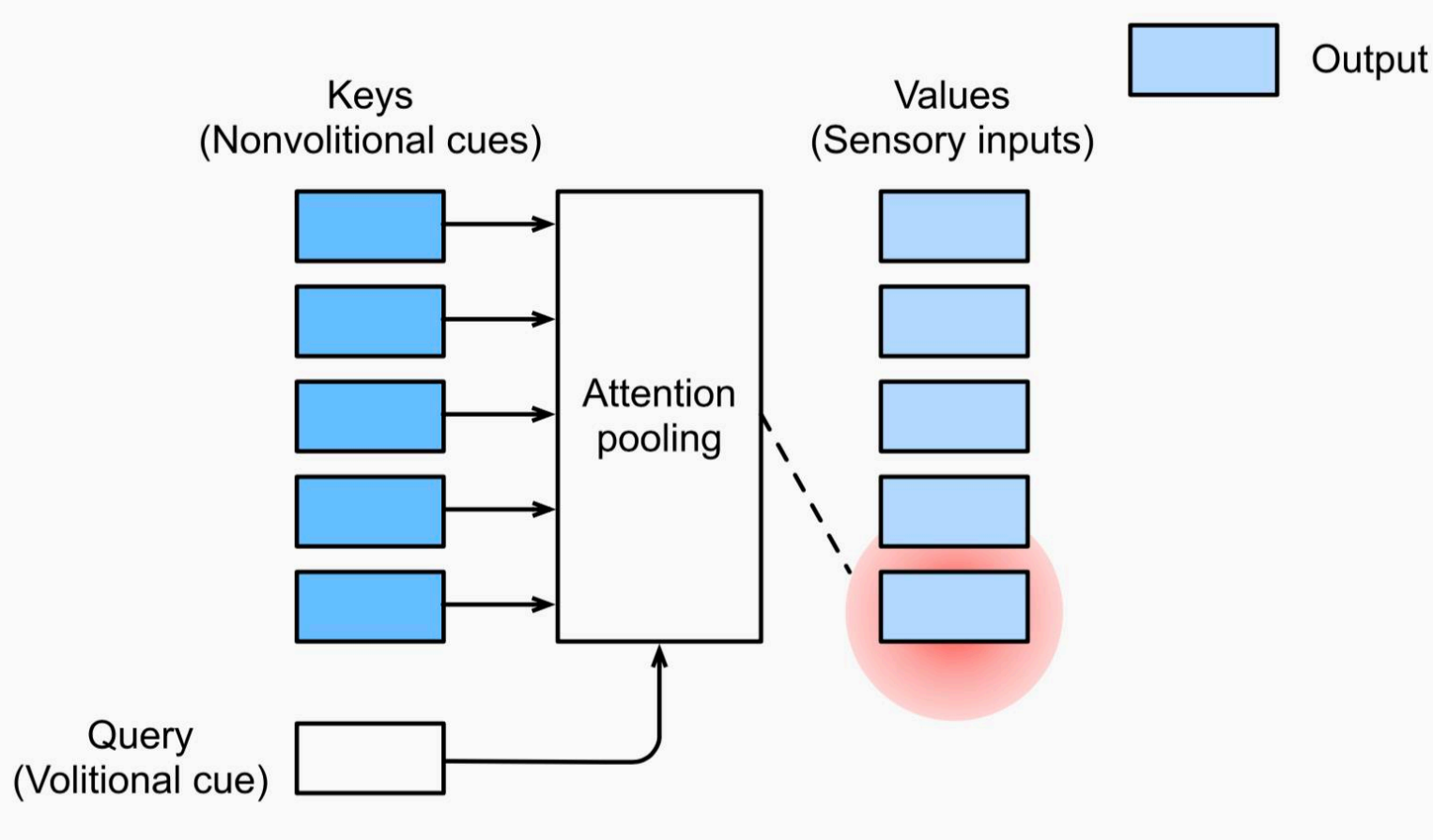
Case where only nonvolitional cues are available:

Selecting between sensory inputs (e.g. features) can be done by a parameterized fully connected layer or even non-parameterized max or average pooling

Attention mechanism: **We use also volitional cues!**

Cues are called **queries** in deep learning.

Attention cues



Attention in deep learning: Queries, keys, and values

- Given any query, attention mechanisms bias selection over sensory inputs (e.g., intermediate feature representations) via **attention pooling**.
- Sensory inputs are called **values**
- Every value is paired with a **key** (nonvolitional cue of that sensory input)

Attention pooling so that the given query (volitional cue) can interact with keys (nonvolitional cues), which guides bias selection over values (sensory inputs).

Image: Zhang et al.

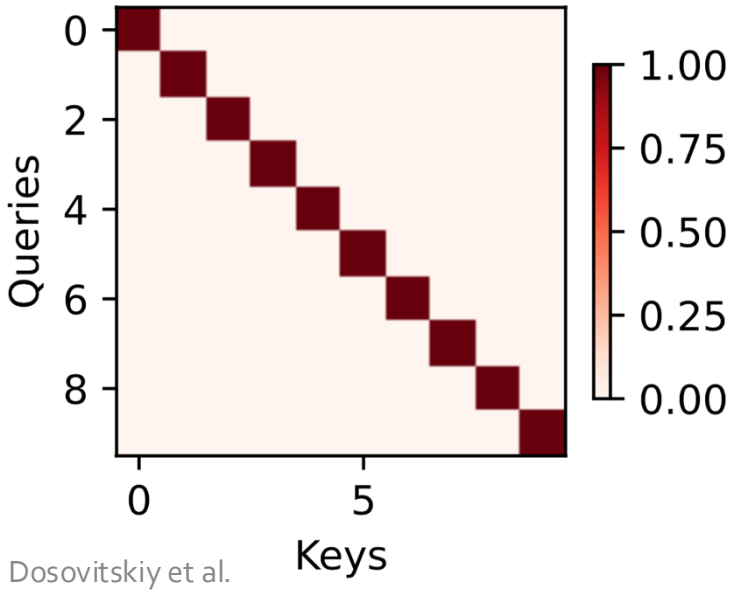
Hertie School

Attention cues

Visualizing attention weights

Attention pooling aggregates values using a weighted average, where weights are computed between the given query and different keys

Simple case where the attention weight is one only when the query and the key are the same; otherwise it is zero:



Examples from NLP and CV:

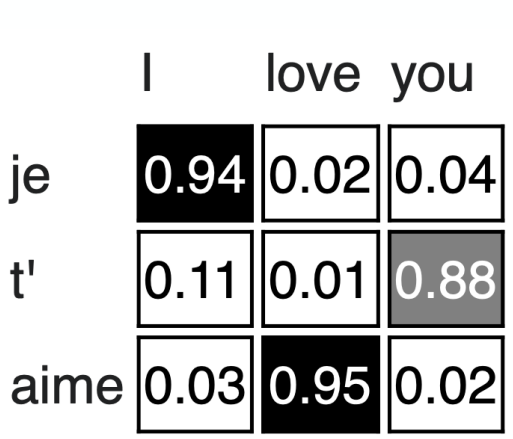


Image: Zhang et al., Wikipedia, Dosovitskiy et al.

Hertie School

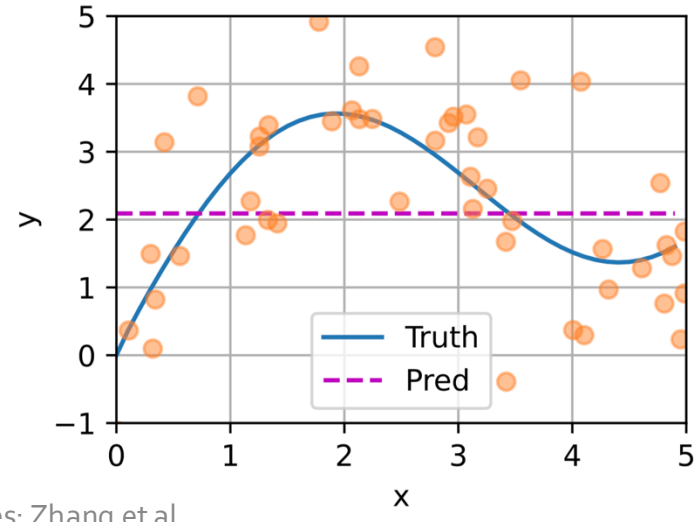
Attention pooling

Non-parametric attention pooling

Using a simple regression example and non-parametric estimators

Average pooling estimator:

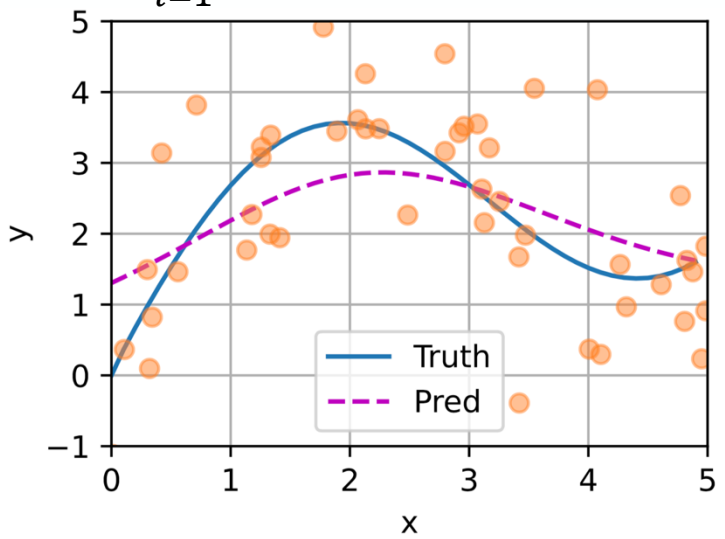
$$f(x) = \frac{1}{n}\sum_{i=1}^{n} y_i$$



Nadaraya-Watson kernel regression:

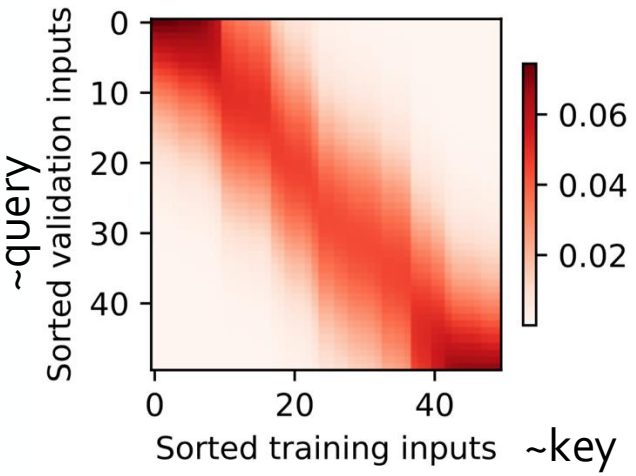
$$f(x) = \sum_{i=1}^{n} \frac{K(x - x_i)}{\sum_{j=1}^{n} K(x - x_j)} y_i$$

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



Nadaraya-Watson kernel regression is example of nonparametric attention pooling:

- x is the query
- (x_i, y_i) is the key-value pair
- $\alpha(x, x_i)$ is the **attention weight** assigned to value y_i



Images: Zhang et al.

Attention pooling

Parametric attention pooling

Introducing learnable parameters w into attention pooling.

Example: Modifying the Nadaraya-Watson kernel regression with Gaussian Kernel by multiplying with parameter w that acts like a "width" of the Kernel

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

$$= \sum_{i=1}^{n} \frac{\exp\left(-\frac{1}{2}[(x - x_i)w]^2\right)}{\sum_{j=1}^{n} \exp\left(-\frac{1}{2}\left[(x - x_j)w\right]^2\right)} y_i$$

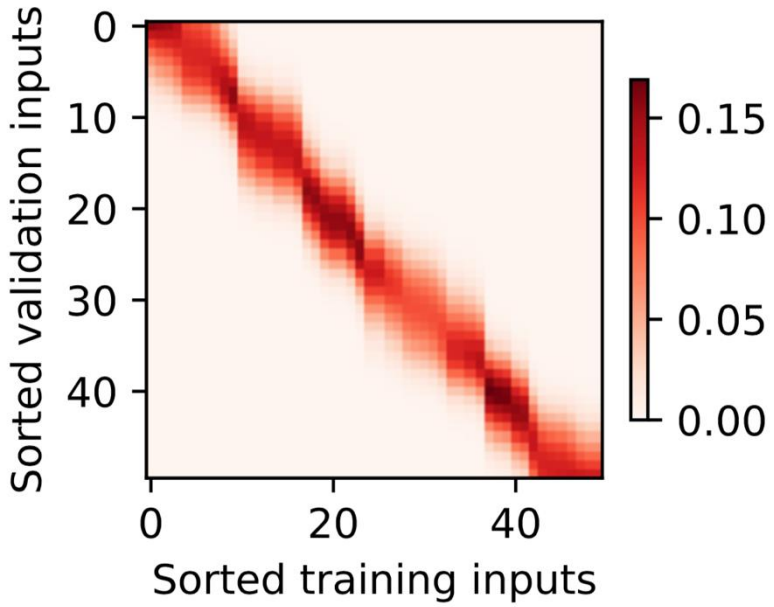
$$= \sum_{i=1}^{n} softmax\left(-\frac{1}{2}\left[(x - x_j)w\right]^2\right) y_i$$

Hertie School

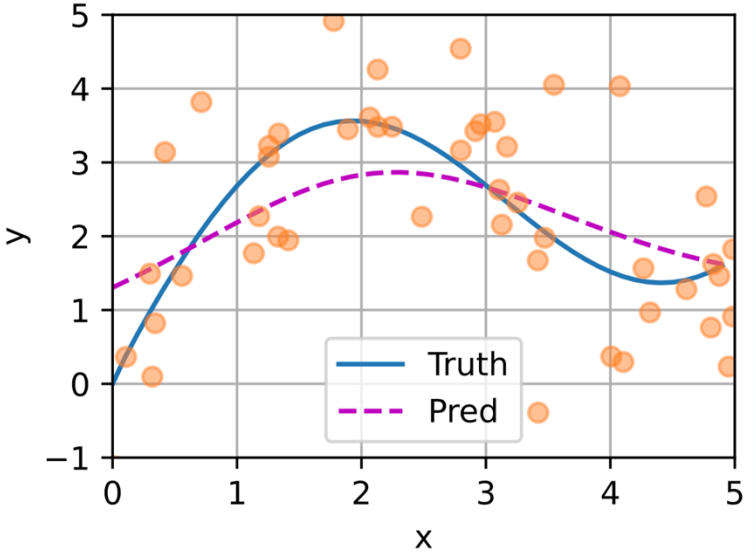
Attention pooling

Parametric attention pooling

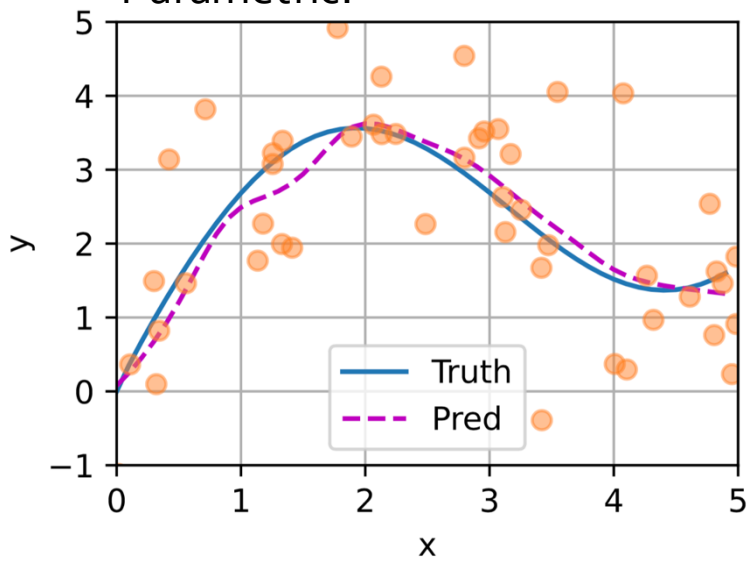
- Training on data with noise: predicted line is less smooth than its nonparametric counterpart
- Comparing with nonparametric attention pooling, the region with large attention weights becomes sharper



Non-parametric:



Parametric:



Attention pooling

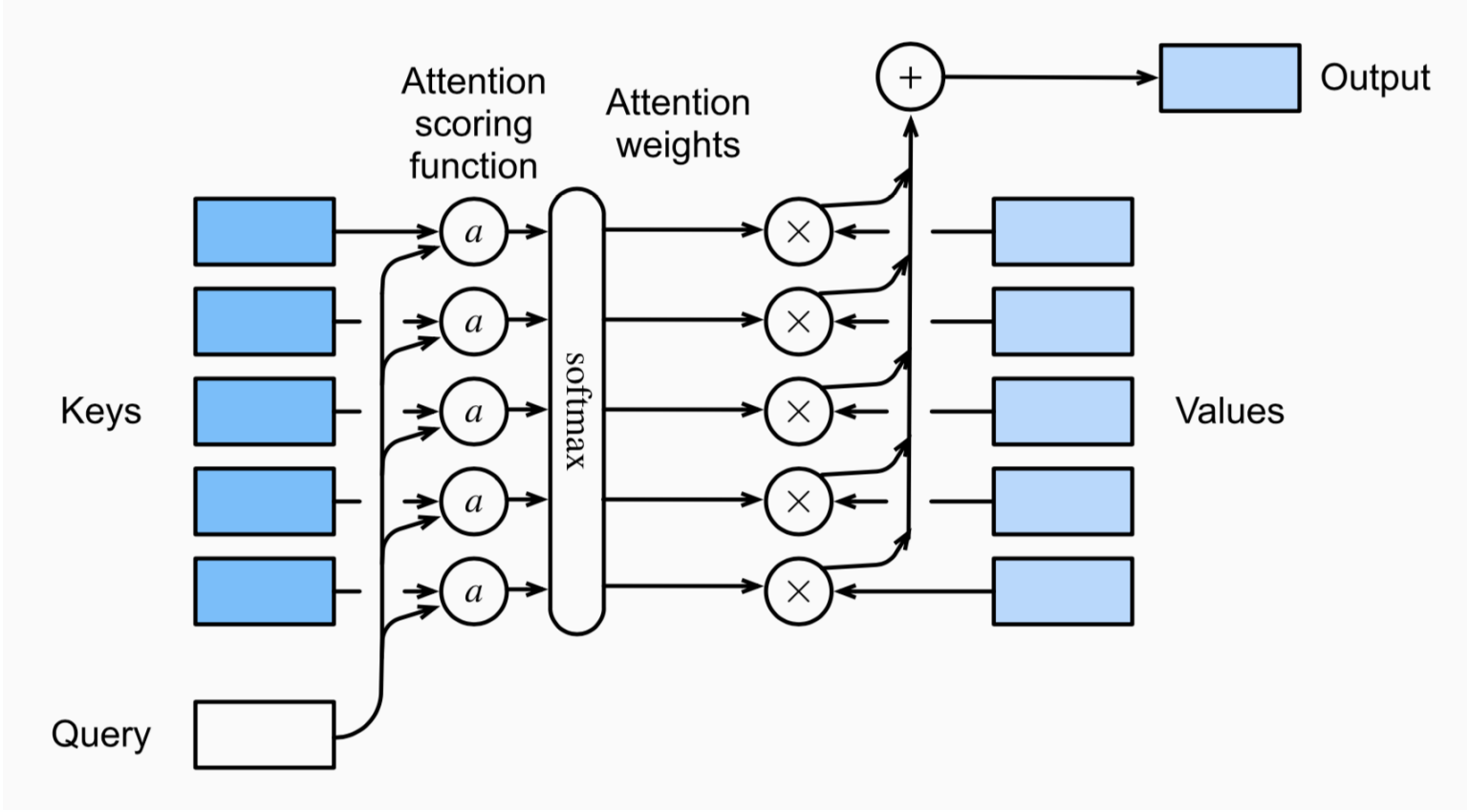


Image: Zhang et al.

Attention scoring function

Example of parametric attention pooling:

- Results of attention scoring function (a) are fed into softmax
- Resulting attention weights are like a probability distribution
- Output of attention pooling is weighted average

Simple example, in reality attention scoring functions more complex!

Attention scoring functions

Additive attention scoring fn

When queries and keys are vectors of **different lengths**.
Given a query $\boldsymbol{q} \in \mathbb{R}^q$ and a key $\boldsymbol{k} \in \mathbb{R}^k$, we have

$$a(\boldsymbol{q}, \boldsymbol{k}) = \boldsymbol{w}_v^T \tanh(\boldsymbol{W}_q \boldsymbol{q} + \boldsymbol{W}_k \boldsymbol{k}) \in \mathbb{R}$$

with learnable parameters $\boldsymbol{W}_q \in \mathbb{R}^{h \times q}$, $\boldsymbol{W}_k \in \mathbb{R}^{h \times k}$, and $\boldsymbol{w}_v^T \in \mathbb{R}^h$

Read as: Query and the key are concatenated and fed into an MLP with a single hidden layer whose number of hidden units is h, a hyperparameter.
This is then fed into softmax.

Scaled dot-product attention scoring fn

When queries and keys are vectors of the **same lengths**.
Given query and key size d, we have

$$a(\boldsymbol{q}, \boldsymbol{k}) = softmax\left(\frac{q^T k}{\sqrt{d}}\right)$$

Weights can be included as well.

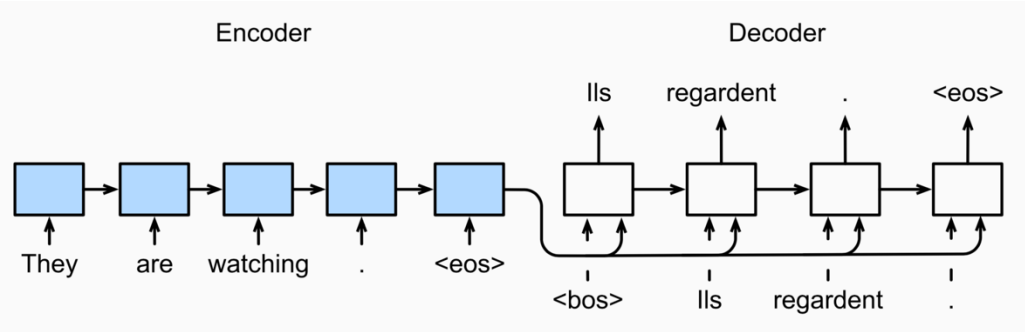
- We can process minibatches efficiently.
- Scaled dot-product attention scoring function is more computationally efficient.
- Used in transformers!

Hertie School

Bahdanau Attention

Seq2seq problems (machine translation)

Problem with vanilla encoder-decoder model using RNN (from before): not all the input tokens are useful for decoding a certain token, but the same context variable $\boldsymbol{c}$ that encodes the entire input sequence is still used at each decoding step



Idea: Model attends only to those parts of the input sequence that are relevant to the current prediction

Solution: Treating the context variable $\boldsymbol{c}$ as an output of attention pooling:

$\boldsymbol{c} = q(\boldsymbol{h}_1, \ldots, \boldsymbol{h}_T)$ now becomes dependent on time step t': $\boldsymbol{c}_{t'} = \sum_{t=1}^{T} \alpha(\boldsymbol{s}_{t'-1}, \boldsymbol{h}_t)\, \boldsymbol{h}_t$

Hertie School

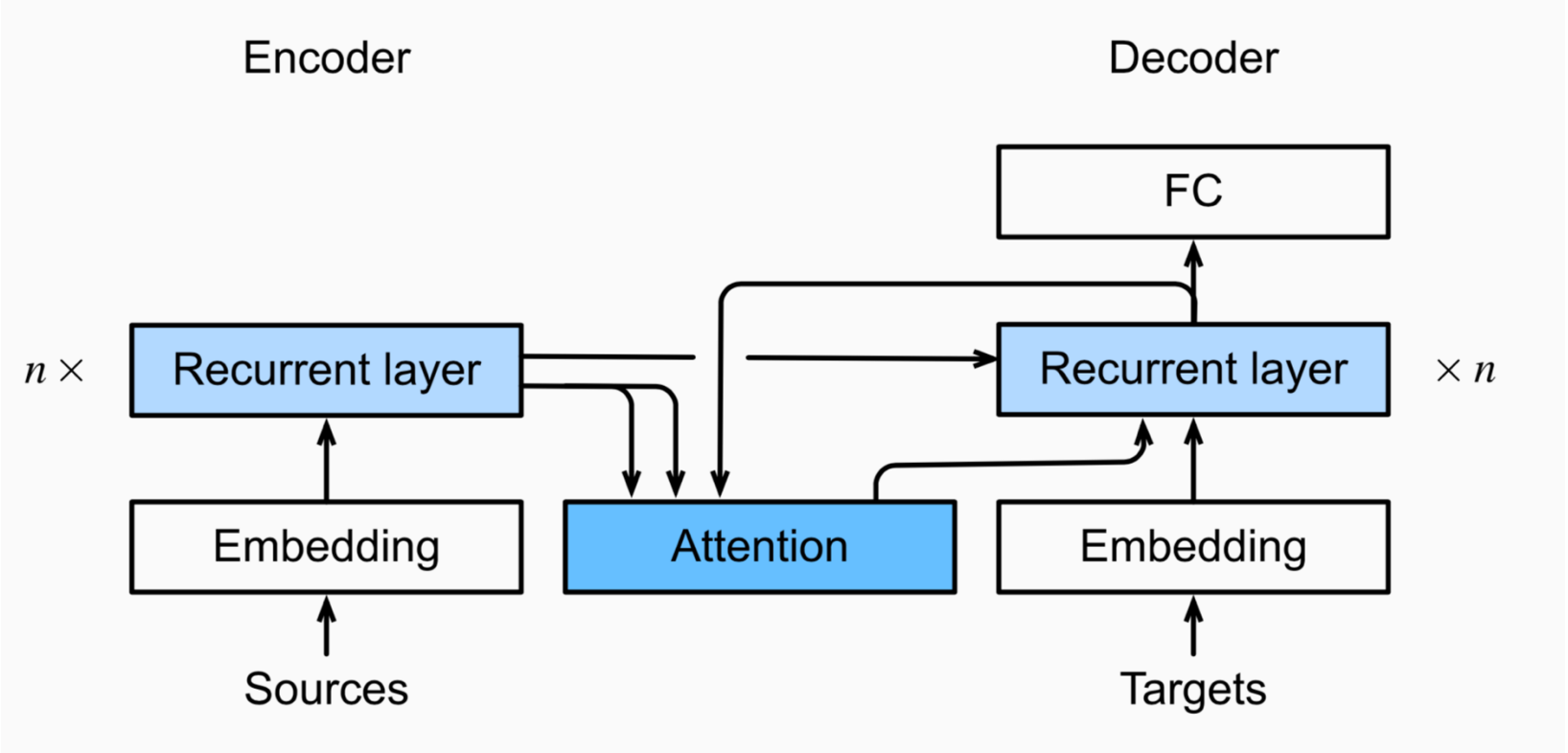
Bahdanau Attention

Seq2seq problems with RNNs (machine translation)

Example of RNN encoder-decoder model with Bahdanau attention

$$\boldsymbol{c}_{t'} = \sum_{t=1}^{T} \alpha(\boldsymbol{s}_{t'-1}, \boldsymbol{h}_t)\, \boldsymbol{h}_t$$

query keys values



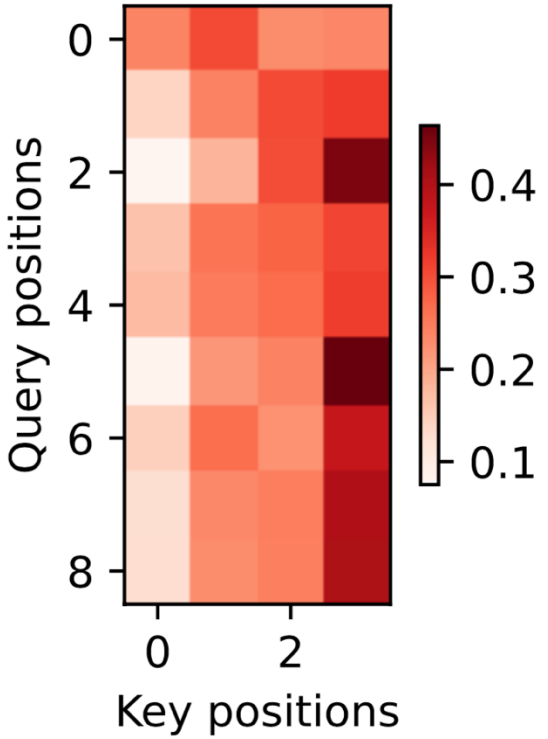


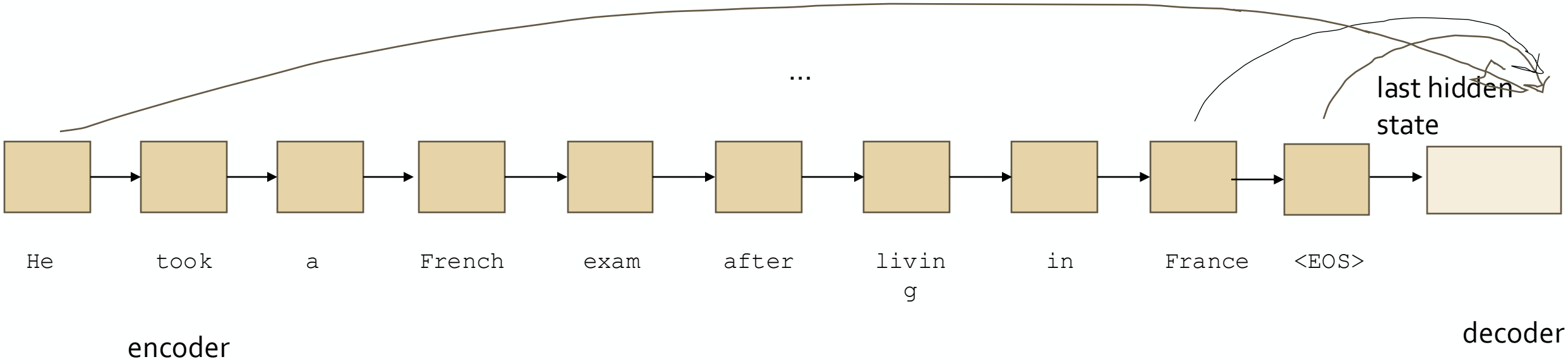
Image: Zhang et al.

Hertie School

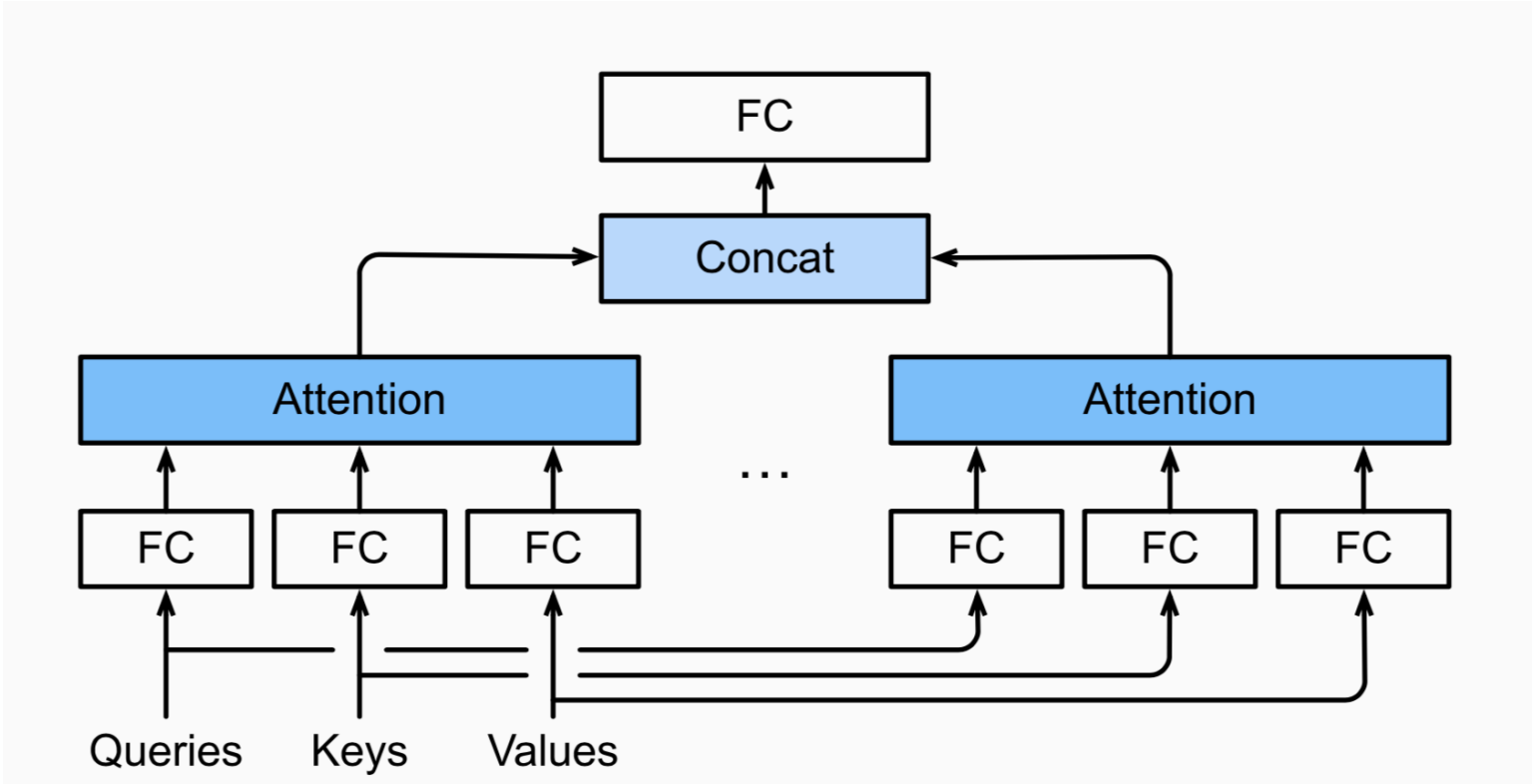
Long sequences

RNN with attention

He took at French exam after living in France.



Multi-head attention

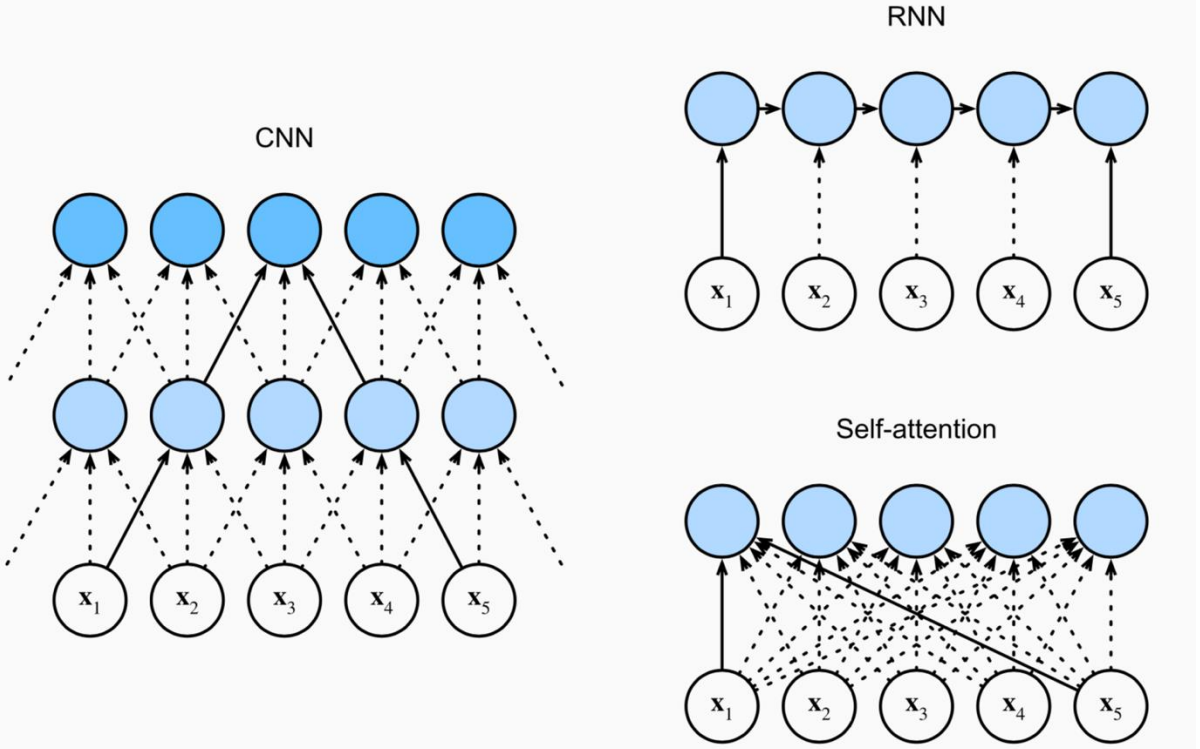


Concatenating different attention outputs

- Same set of queries, keys, and values can be transformed with different independently learned linear projections and fed into attention pooling in parallel
- Every attention pooling output is a **head**
- Attention outputs are then concatenated and transformed with another learned linear projection

Image: Zhang et al.

Self-attention



$$\boldsymbol{y}_i = \sum_{j=1}^{n} a(\boldsymbol{q}, \boldsymbol{k})\boldsymbol{v}_j$$

Image: Zhang et al.

Self-attention (for transformers)

Same set of tokens act as queries, keys, and values

Input sequence of n tokens $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ that are each a vector dimension d.

Attention output is a sequence of length n:

$$\boldsymbol{y}_i = f\left(\boldsymbol{x}_i, (\boldsymbol{x}_1, \boldsymbol{x}_1), \ldots, (\boldsymbol{x}_n, \boldsymbol{x}_n)\right)$$

$$= \sum_{j=1}^{n} \alpha\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right)\boldsymbol{x}_j \in \mathbb{R}^d$$

- Both CNNs and self-attention enjoy parallel computation
- Self-attention lends itself to learn long-range dependencies
- Quadratic computational complexity with respect to the sequence length makes self-attention prohibitively slow for very long sequences

Self-attention

Example

We calculate the attention weight for each word in the sequence.
The output is a vector of dimension d (like the initial embedding) but with **contextual embedding**.

$$\boldsymbol{y}_i = \sum_{j=1}^{n} a(\boldsymbol{q}_i, \boldsymbol{k}_j)\boldsymbol{v}_j = \sum_{j=1}^{n} \alpha'(\boldsymbol{x}_i, \boldsymbol{x}_j)\boldsymbol{x}_j$$

Attention output

$$\boldsymbol{y}_1 \quad \boldsymbol{y}_2 \quad \boldsymbol{y}_3 \quad \boldsymbol{y}_4$$

$$\boldsymbol{y}_3 = \sum_{j=1}^{n} a(\boldsymbol{q}_3, \boldsymbol{k}_j)\boldsymbol{v}_j = \sum_{j=1}^{n} a'(\boldsymbol{x}_3, \boldsymbol{x}_j)\boldsymbol{W}_v\boldsymbol{x}_j$$

$$\boldsymbol{q}_1, \boldsymbol{k}_1, \boldsymbol{v}_1 \quad \boldsymbol{q}_2, \boldsymbol{k}_2, \boldsymbol{v}_2 \quad \boldsymbol{q}_3, \boldsymbol{k}_3, \boldsymbol{v}_3 \quad \boldsymbol{q}_4, \boldsymbol{k}_4, \boldsymbol{v}_4$$

$$\boldsymbol{q}_1 = \boldsymbol{W}_q\boldsymbol{x}_1$$
$$\boldsymbol{k}_1 = \boldsymbol{W}_k\boldsymbol{x}_1$$
$$\boldsymbol{v}_1 = \boldsymbol{W}_v\boldsymbol{x}_1$$

Input (e.g. embedded token)

$\boldsymbol{x}_1$	$\boldsymbol{x}_2$	$\boldsymbol{x}_3$	$\boldsymbol{x}_4$
Ich	möchte	Deutsch	lernen.

Self-attention

Exercise

Turn to your partner to answer the following question:

What could the final attention output $\boldsymbol{y}_3$ look like if you were to write it in terms of the initial embeddings x_i and the weights W? (Recall the attention scoring functions, variations exist).

Positional encoding

Positional encoding

- Because self-attention attends to all words at the same time, we need positional information.
- We can inject absolute or relative positional information by adding **positional encoding** to the input representations.
- We define a matrix $\boldsymbol{P} \in \mathbb{R}^{n \times d}$ that has the same dimension as the d-dimensional embeddings for n tokens of the input sequence $\boldsymbol{X} \in \mathbb{R}^{n \times d}$.
- The values of $\boldsymbol{P}$ can for example be given by trigonometric functions as float numbers, which makes them computationally more efficient.
- The positional encoding outputs are then $\boldsymbol{X} + \boldsymbol{P}$

Hertie School

Transformers

Zhang et al. etc.

Transformers

Overview

- Transformer models are solely based on attention mechanisms without any convolutional or recurrent layer
- Originally proposed for sequence-to-sequence learning on text data (emerged 2017)
- Transformers are omnipresent in a wide range of modern deep learning applications (language, vision, speech, and reinforcement learning).
- BERT, GPT-4, Vision Transformer, BLOOM, etc. are all transformers
- Encoder-decoder architecture, and also either the encoder or the decoder can be used individually in practice
- Using self-attention, the transformer processes the entire input sequence at once
- Transformers process the input in a bidirectional way

Transformers

Architecture

- Based on encoder-decoder architecture
- Uses positional encoding

Encoder:

- Multi-head self-attention pooling and positionwise feed-forward network
- Transformer encoder outputs a d-dimensional vector representation for each position of the input sequence

Decoder:

- Decoder is allowed to only attend to all positions in the target sequence up to that position → Masked multi-head attention
- Queries are from the outputs of the previous decoder layer, and the keys and values are from the transformer encoder outputs

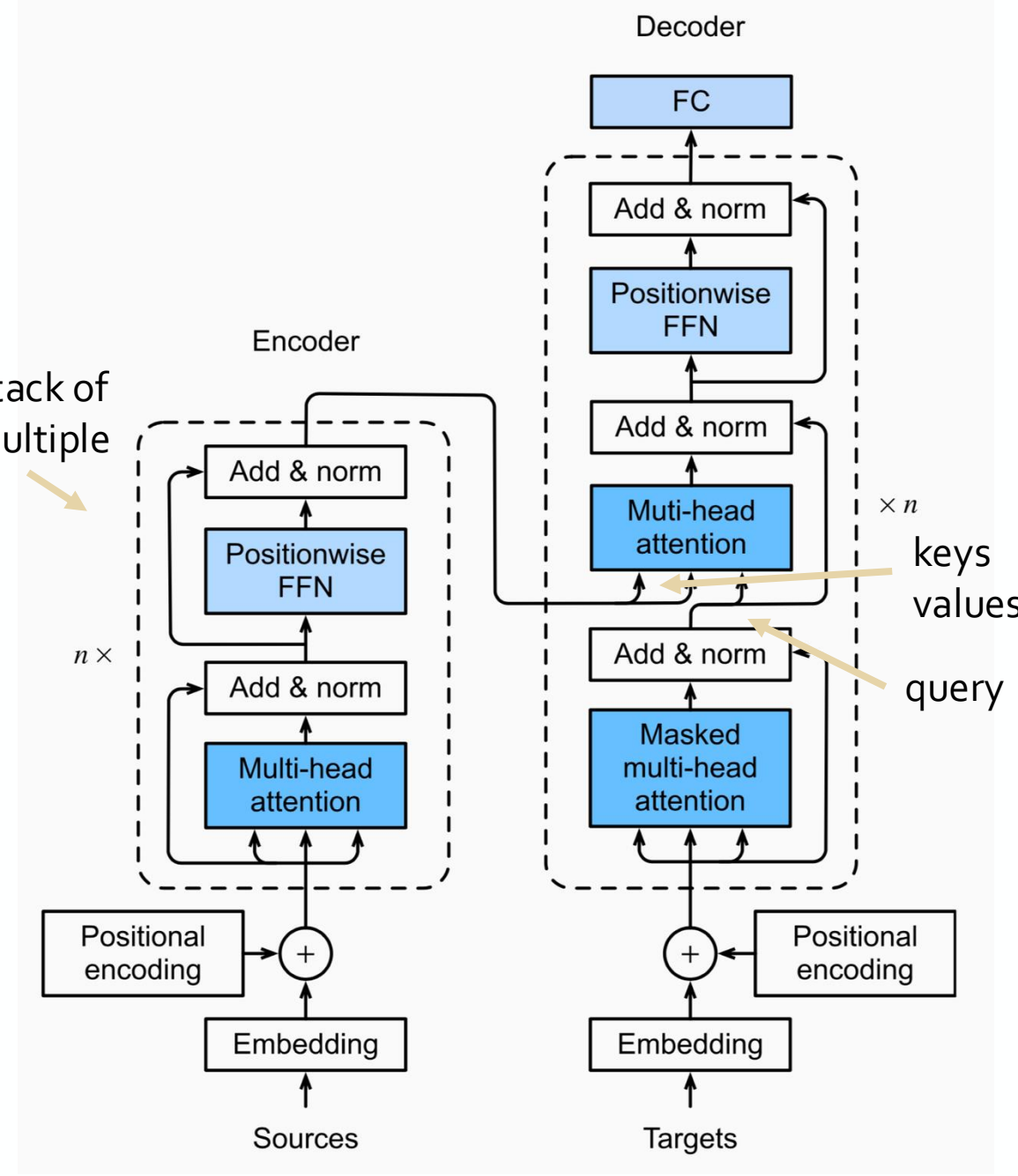


Image: Zhang et al.

BERT (Encoder-Only)

BERT (Bidirectional Encoder Representations from Transformers)

- Only the transformer encoder is used: a sequence of input tokens is converted into the same number of representations that can be further projected into output (e.g., classification)
- BERT pretrained with masked language modeling: input text with randomly masked tokens is fed into a transformer encoder to predict the masked tokens
- Large-scale text data from books and Wikipedia can be used for pretraining BERT with masking, no need for manual labeling (figure above)
- Fine-tuning of BERT for sentiment analysis. The transformer encoder is a pretrained BERT, which takes a text sequence as input and feeds the representation into a fully connected layer to predict the sentiment (figure below).

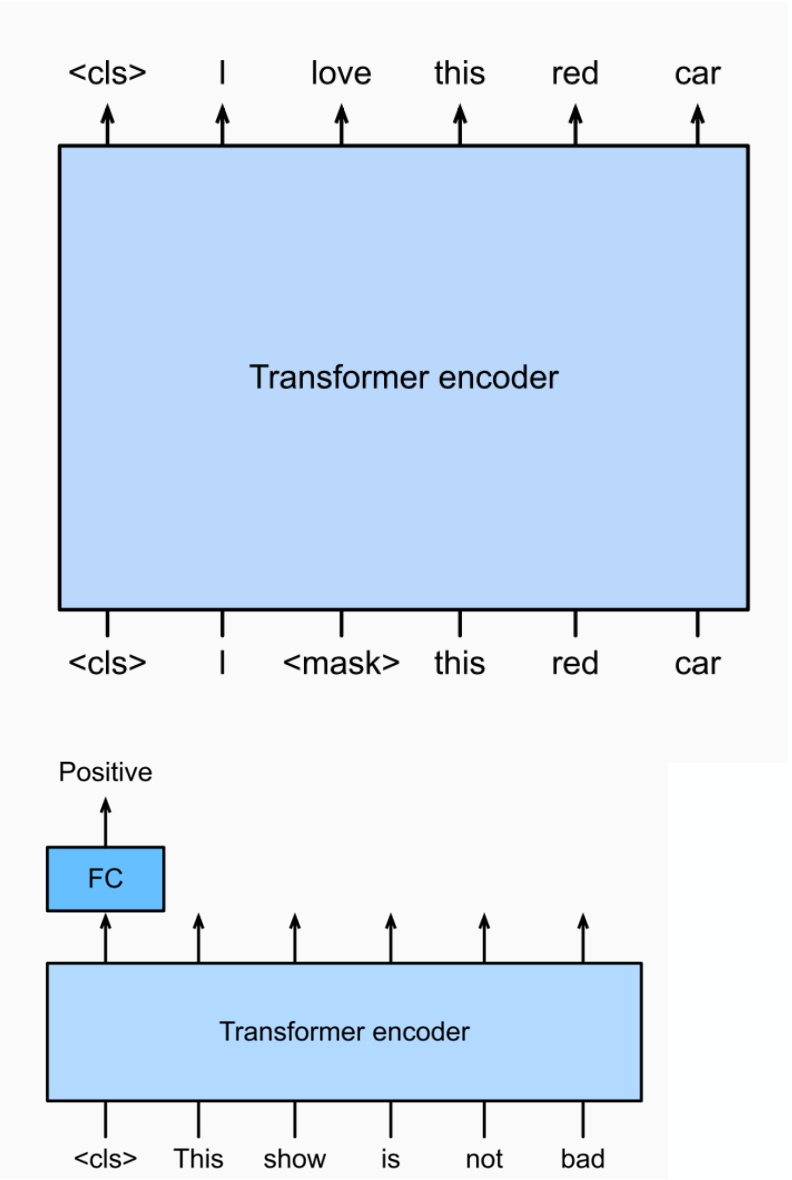


Image: Zhang et al.

BERT (Encoder-Only)

BERT's variants

- RoBERTa: a BERT variant of the same size trained on more data (2000 billion tokens)
- ALBERT: enforcing parameter sharing
- SpanBERT: representing and predicting spans of text
- DistilBERT: lightweight variant via knowledge distillation
- diseaseBERT is a domain-specific LM that can distinguish different aspects of diseases
- Pre-trained biomedical LMs (BlueBERT, ClinicalBERT, SciBERT, BioBERT)
- ClimateBERT: trained on language related to climate change

Hertie School

ViT (Encoder-Only)

Vision Transformer (ViT)

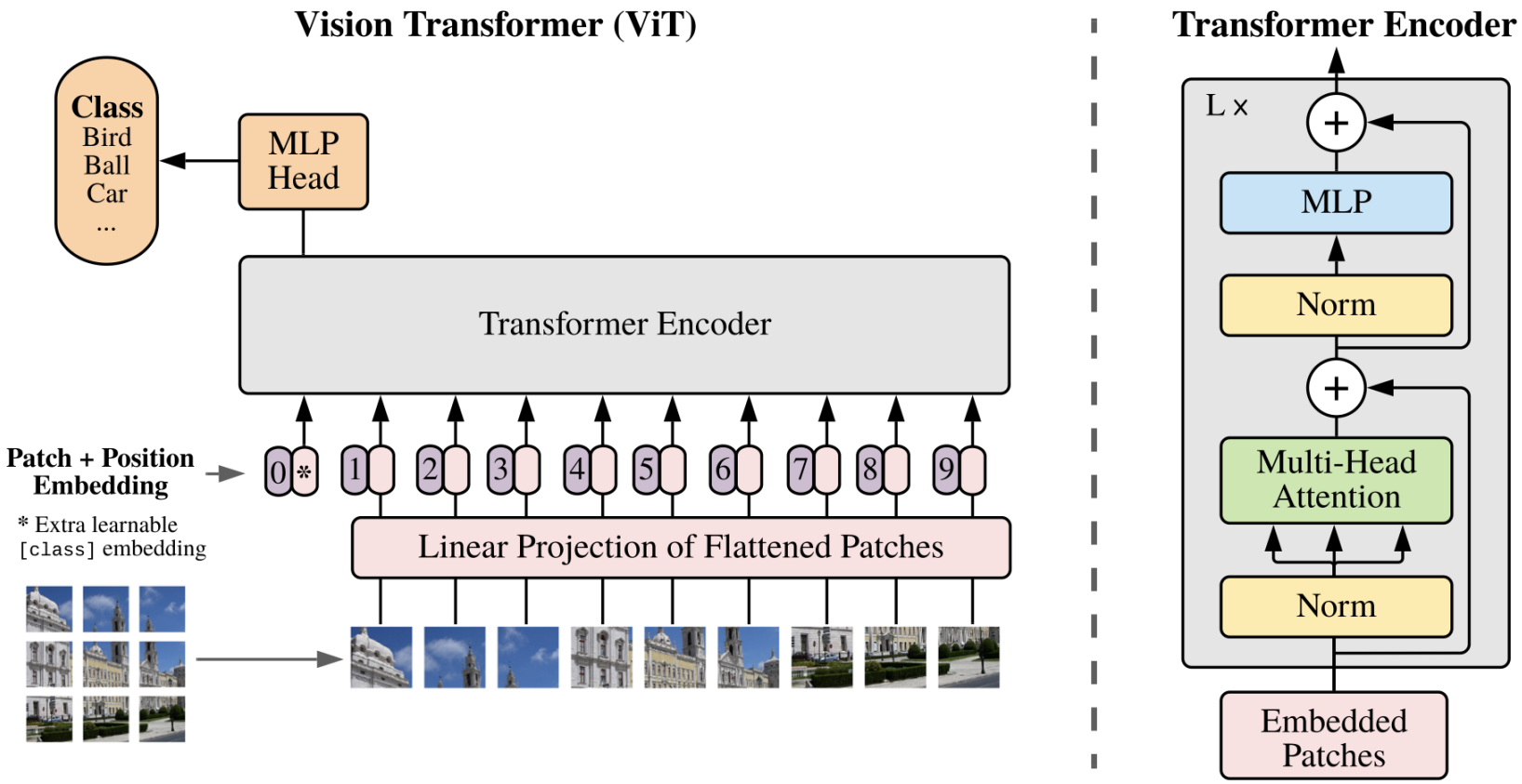


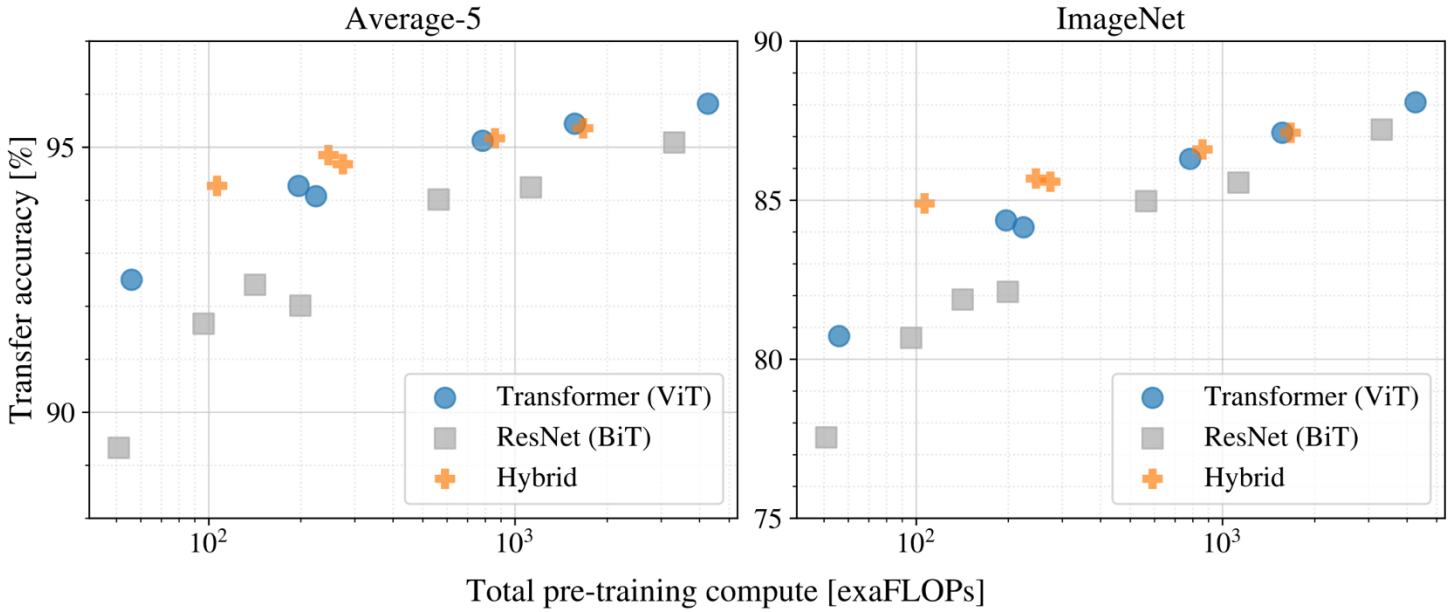
Image: Dosovitskiy et al. (2021)

Hertie School

ViT (Encoder-Only)

Vision Transformer (ViT)

ViT generally outperform ResNets with the same computational budget.
ViT do not saturate in performance → potential for scaling



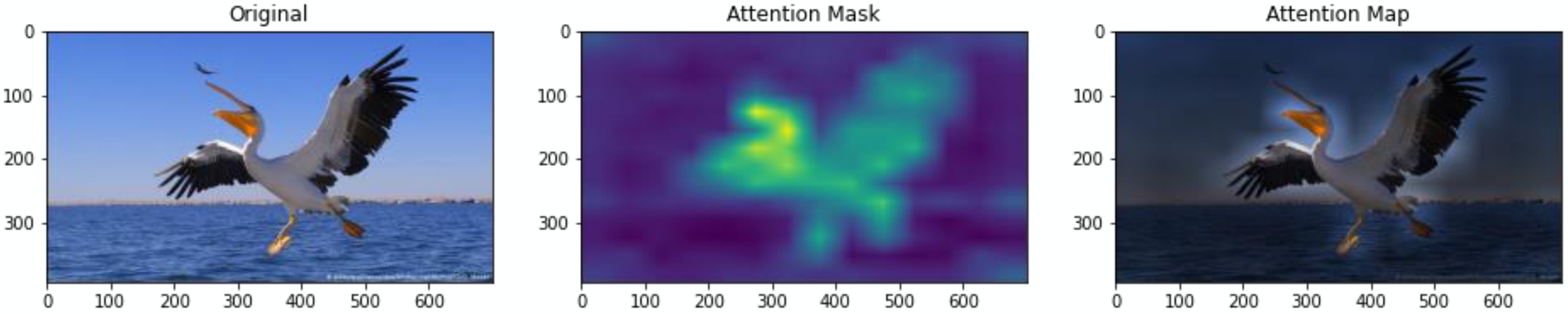


Image: Dosovitskiy et al. (2021), jo1jun/Vision_Transformer

Hertie School

Hertie School
Friedrichstraße 180
10117 Berlin, Germany
T +49 (0)30 259219-0
F +49 (0)30 259219-11
info@hertie-school.org
www.hertie-school.org