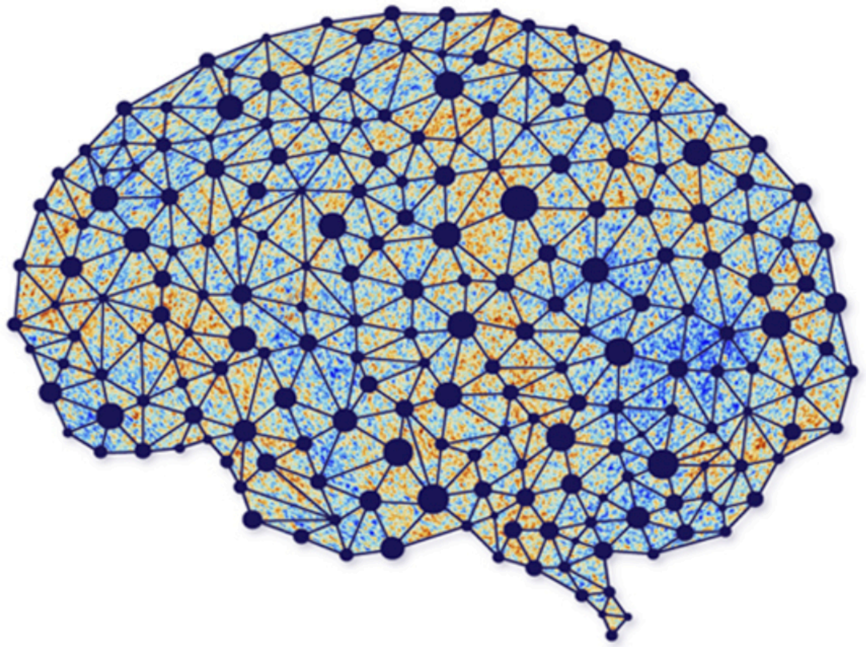
PHY 835: Machine Learning in Physics

Lecture 19: Transformers Part 1

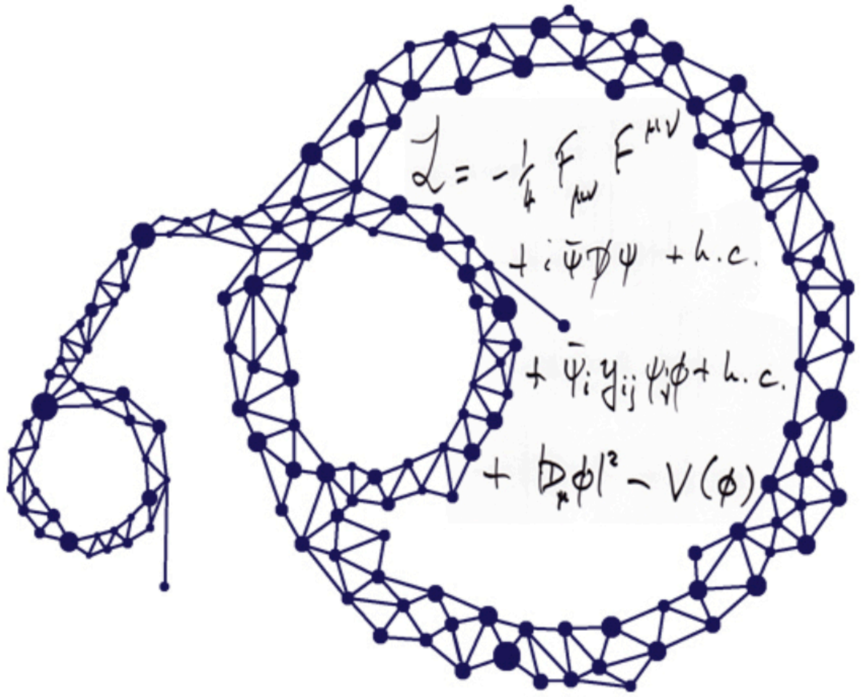
April 2, 2024



AI

$\cap$

Universe


$\mathcal{L} = -\frac{1}{4} F_{\mu\nu} F^{\mu\nu}$
$+ i\bar{\psi}\not{D}\psi + h.c.$
$+ \bar{\psi}_i y_{ij} \psi_j \phi + h.c.$
$+ |D_\mu \phi|^2 - V(\phi)$

Gary Shiu

Introduction

- Transformer is one of the most talked about ML architecture (e.g. ChatGPT).
- Initially targeted at natural language processing (NLP) problems, transformers are now being used quite generally on unstructured data representations (texts, images, audio, video, and their combo).
- These ML models are known as transformers because they transform a set of vectors in some representation space into a corresponding set of vectors, having the same dimensionality, in some new space.
- The new space has a richer internal representation that is better suited to solving downstream tasks.
- Reference: “Deep learning: Foundations and Concepts” by Chris Bishop with Hugh Bishop, Chapter 12: https://www.bishopbook.com/

Why should you care?

- Math and Physics problems are language problems, expressed in terms of formulae. Your tasks are to translate questions to answers.

- Numerous applications of transformers in math and theoretical physics. Applications of ML are not limited to experimental areas.

- Some success in solving college level physics and math problems (see talks by Guy Gur-Ari and Francois Charton at http://www.physicsmeetsml.org/)

- AI Does Math as Well as Math Olympians: https://www.scientificamerican.com/article/ai-matches-the-abilities-of-the-best-math-olympians/

- Examples of research level problems:

  - https://deepmind.google/discover/blog/funsearch-making-new-discoveries-in-mathematical-sciences-using-large-language-models/

  - https://nips.cc/virtual/2023/76132

Foundational Model

- A large-scale model that can be adapted to solve multiple different tasks is known as a foundation model, e.g., https://polymathic-ai.org/
- Transformers can be trained in a self-supervised way using unlabeled data, which is especially effective with language models since there are vast quantities of text available from the internet.
- The scaling hypothesis asserts that simply by increasing the number of learnable parameters and training on a commensurately large data set, significant improvements in performance can be achieved.
- Transformers are quite suited for massively parallel processing hardware, e.g., GPU. Models with 10^{12} parameters can be trained in reasonable time.
- The pre-trained models can then be fine-tuned for specific tasks, achieving artificial general intelligence (AGI).

Natural Language Processing

- Language datasets share some similarities with image data:
    - The number of input variables can be very large.
    - The statistics are similar at every position; not sensible to re-learn the meaning of dog at every possible position in a body of text.
- These are the reasons for introducing CNN: instead of fully connected NN, a CNN employs parameter sharing.
- However, language datasets have varying lengths in text sequences. There is no easy way to resize them.

An Illustrative Example

- Consider the following restaurant review

  > The restaurant refused to serve me a ham sandwich because it only cooks vegetarian food. In the end, they just gave me two slices of bread. Their ambiance was just as good as the food and service.

- How to process texts like this into a representation suitable for downstream tasks (positive/negative review? is steak served?)

- Three problems to overcome:
  - Inputs are large: 37 words represented by an embedding vector of length 1024 has a 37x1024 = 37888 dimensional input.
  - Inputs have different lengths: not obvious how to apply fully connected NNs; how to share parameters across words at different positions?
  - Language is ambiguous: it refers to the restaurant and not to ham sandwich. A successful ML model should pay *attention* to the word restaurant. There are connections between words and the strength of these connections depends on the words themselves. The word their also refers to the restaurant.

Attention is all you need

https://arxiv.org/abs/1706.03762

- Originally developed as an enhancement to RNNs for machine translation: https://arxiv.org/abs/1409.0473
- https://arxiv.org/abs/1706.03762) showed that the RNN structure can be eliminated; instead focus exclusively on the attention mechanism.
- Consider the following two sentences:

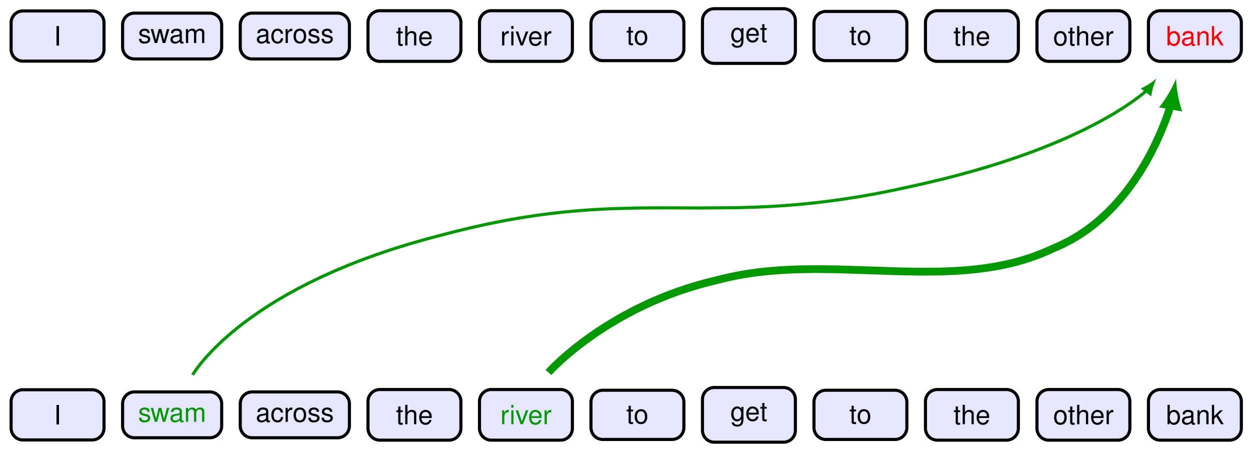
I swam across the river to get to the other bank.

I walked across the road to get cash from the bank.

- The word “bank” has different meanings which can be detected by looking at other words in the sentence.
- In the first sentence, the words “swam” and “river” most strongly indicate that “bank” refers to the side of a river, while in the second sentence, the word “cash” is a strong indicator that “bank” refers to a financial institution.

Attention is all you need

- A NN processing a sentence should attend to specific words from the rest of the sequence:



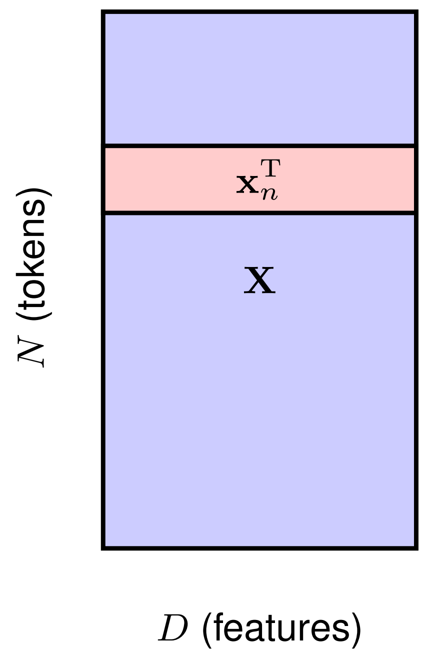
- The specific locations that should receive more attention depends on the input sequence itself.
- In a standard NN, once a network is trained, the weights are independent on the input data.
- By contrast, attention uses weighting factors whose values depend on the specific input data.

Word Embedding

- Words are mapped into vectors in an embedding space.
- Words with similar meanings are mapped to nearby locations in the embedding space.
- A transformer is a richer form of embedding in which a given vector is mapped to a location that depends on other vectors in the sequence.
- The vector representing “bank” is mapped to a location close to “water” in the embedding space in the first sentence, and close to “money” in the second sentence.
- Not only for words: a protein is a 1d sequence of amino acids (22 possibilities). A protein can comprise hundreds or thousands of such amino acids. Amino acids that are widely separated in the 1d sequence can be physically close in 3d space if the proton folds. A transformer model allows distant amino acids to attend to each other for modeling 3d structure.
- For similar reasons, transformers have been used for modeling molecular dynamics.

Transformer Processing

- Input data is a set of vectors $\{\mathbf{x}_n\}$ of dimensionality D, $n = 1,\ldots,N$.
- These data vectors are known as **tokens** (e.g., a word within a sentence, a patch within an image, or an amino acid within a protein).
- The elements x_{ni} of the tokens are called **features**.
- Transformers can handle a mix of different data types by combining the data variables into a joint set of tokens.
- Combining the data vectors into a matrix X of dimensions $N \times D$.



$$\widetilde{\mathbf{X}} = \mathrm{TransformerLayer}\left[\mathbf{X}\right]$$

same dimensionality as X

Apply multiple transformer layer
to learn rich internal representations.

Attention Coefficients

- A set of input tokens $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ is mapped to a set of output tokens $\{\mathbf{y}_1, \ldots, \mathbf{y}_N\}$.
- With attention, this dependence should be stronger for those inputs $\mathbf{x}_m$ that are particularly important for determining $\mathbf{y}_n$.
- Consider the map:

$$\mathbf{y}_n = \sum_{m=1}^{N} a_{nm}\mathbf{x}_m$$

where a_{nm} are called attention weights. $a_{nm} \approx 0$ for input tokens $\mathbf{x}_m$ that have little influence on the output $\mathbf{y}_n$ and large otherwise.

- The attention weights satisfy two constraints:

$$a_{nm} \geqslant 0$$

avoid cancellation from large coefficients of opposite signs.

$$\sum_{m=1}^{N} a_{nm} = 1.$$

normalize the total attention.

Self-attention

- Consider the problem of choosing which movie to watch on Netflix.
- Associate each movie with a list of attributes: genre, names of leading actors, length of movie, etc.
- Search thru a catalogue to find a movie that matches preferences.
- Encode the attributes of each movie in a vector called the **key**.
- The corresponding movie file is called a **value**.
- The user's personal vector of attributes is called the **query**.
- Netflix compares the query vector with all the key vectors to find the best match, and send the user the corresponding movie (value) file.
- Hard attention: a single value vector is returned.

Dot-Product Self-attention

- For transformer, we generalize this info retrieval to **soft attention**.
- Use continuous variables to measure the degree of match between queries and keys, then use these variables to weight the influence.
- Transformer function is differentiable, trainable by gradient descent.
- To satisfy the two constraints on the attention weights, we define:

$$a_{nm} = \frac{\exp(\mathbf{x}_n^{\mathrm{T}}\mathbf{x}_m)}{\sum_{m'=1}^{N}\exp(\mathbf{x}_n^{\mathrm{T}}\mathbf{x}_{m'})}$$

- In matrix notation:

$$\mathbf{Y} = \mathrm{Softmax}\left[\mathbf{X}\mathbf{X}^{\mathrm{T}}\right]\mathbf{X}$$

where $\mathrm{Softmax}[\mathbf{L}]$ is an operator that takes the exponential of every element of a matrix $\mathbf{L}$ then normalizes each row independently to sum to 1.

- **Dot-product self-attention** (using the same sequence to determine the queries, keys, and values; measure of similarity is given by dot product).

Network Parameters

- Transformation from $\{\mathbf{x}_n\}$ to $\{\mathbf{y}_n\}$ is fixed, with no capacity to learn from data because it has no adjustable parameters.
- Each feature within a token vector $\{\mathbf{x}_n\}$ plays an equal role in determining a_{nm}. Want flexibility to focus on some features vs others.
- We can address both issues if we define modified feature vectors:

$$\widetilde{\mathbf{X}} = \mathbf{X}\mathbf{U}$$

- $\mathbf{U}$ is a $D \times D$ matrix of learnable weight parameters, analogous to a layer in a standard NN. This gives a modified transformation:

$$\mathbf{Y} = \mathrm{Softmax}\left[\mathbf{X}\mathbf{U}\mathbf{U}^{\mathrm{T}}\mathbf{X}^{\mathrm{T}}\right]\mathbf{X}\mathbf{U}$$

- This has more flexibility, but still the matrix $\mathbf{X}\mathbf{U}\mathbf{U}^{\mathbf{T}}\mathbf{X}^{\mathbf{T}}$ is symmetric.

Network Parameters

- The attention mechanism should support significant asymmetry, e.g., “chisel” is strongly associated with “tool”, but not the other way round.
- Although the softmax function means the attention weights matrix is not symmetric (NB normalization), we can create more flexibility by allowing queries & keys to have independent parameters.
- Define query, key, & value matrices each w/ different transformations:

$$\mathbf{Q} = \mathbf{X}\mathbf{W}^{(\mathrm{q})}$$
$$\mathbf{K} = \mathbf{X}\mathbf{W}^{(\mathrm{k})}$$
$$\mathbf{V} = \mathbf{X}\mathbf{W}^{(\mathrm{v})}$$

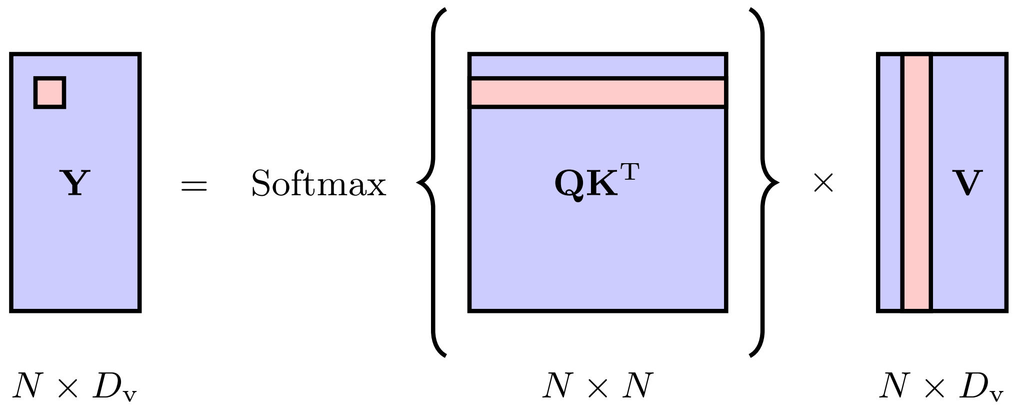
  the weight matrices $\mathbf{W}^{(q)}, \mathbf{W}^{(k)}, \mathbf{W}^{(v)}$ represent parameters that will be **learned** during the training of the transformer architecture.

- $\mathbf{W}^{(q)}, \mathbf{W}^{(k)}, \mathbf{W}^{(v)}$ are matrices of dim. $D \times D_k, D \times D_q, D \times D_v$. Setting $D_k = D_q$ allows for dot-products between query and key while $D_v = D$ allows multiple transformer layers to be stacked. We set $D_k = D_q = D_v = D$.

Network Parameters

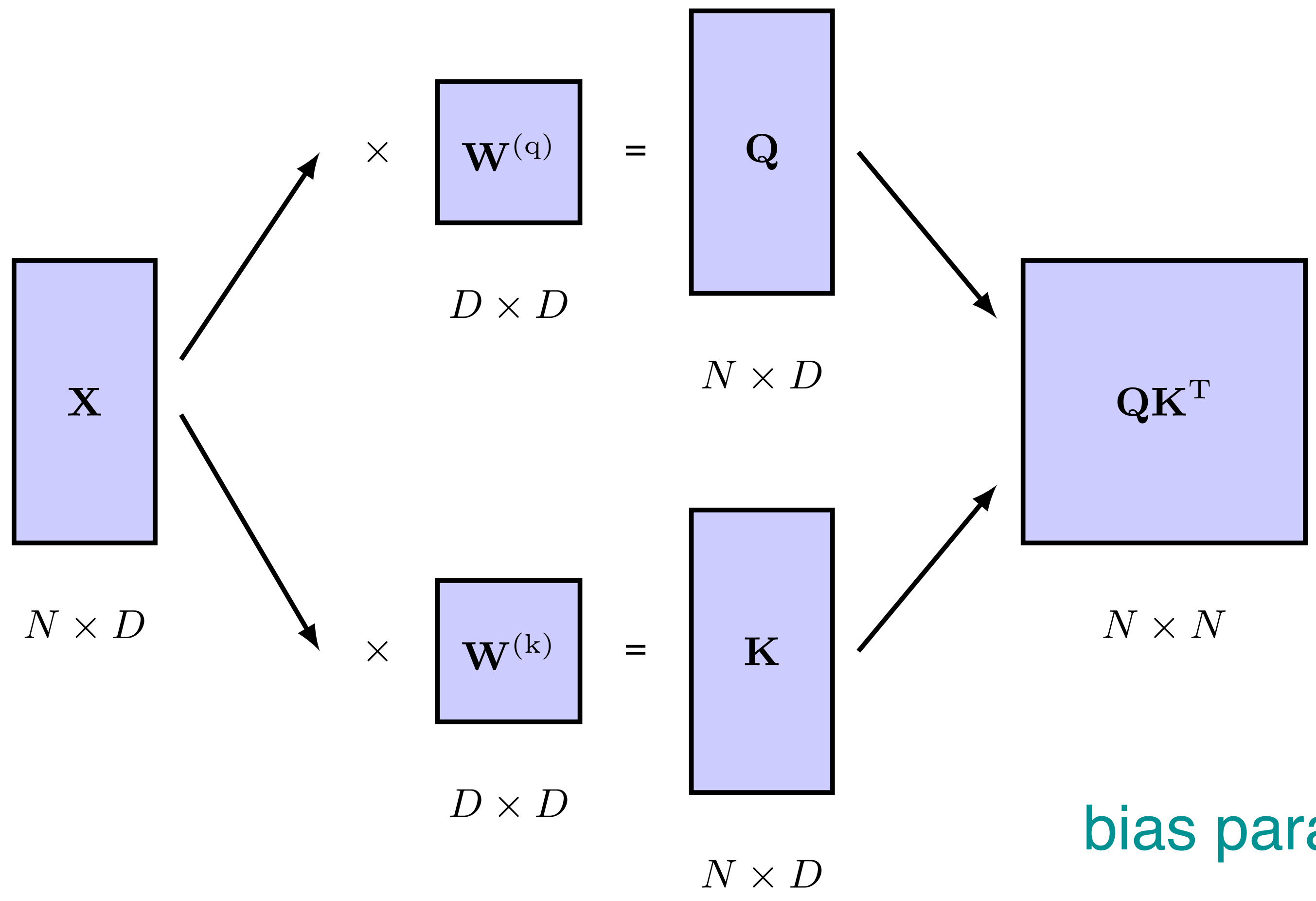
- The transformation is now generalized to:

$$\mathbf{Y} = \text{Softmax}\left[\mathbf{Q}\mathbf{K}^{\mathrm{T}}\right]\mathbf{V}$$



$\mathbf{Y}$
=
Softmax
$\mathbf{QK}^{\mathrm{T}}$
×
$\mathbf{V}$
$N \times D_{\mathrm{v}}$
$N \times N$
$N \times D_{\mathrm{v}}$

whereas the dot-product can be computed by:



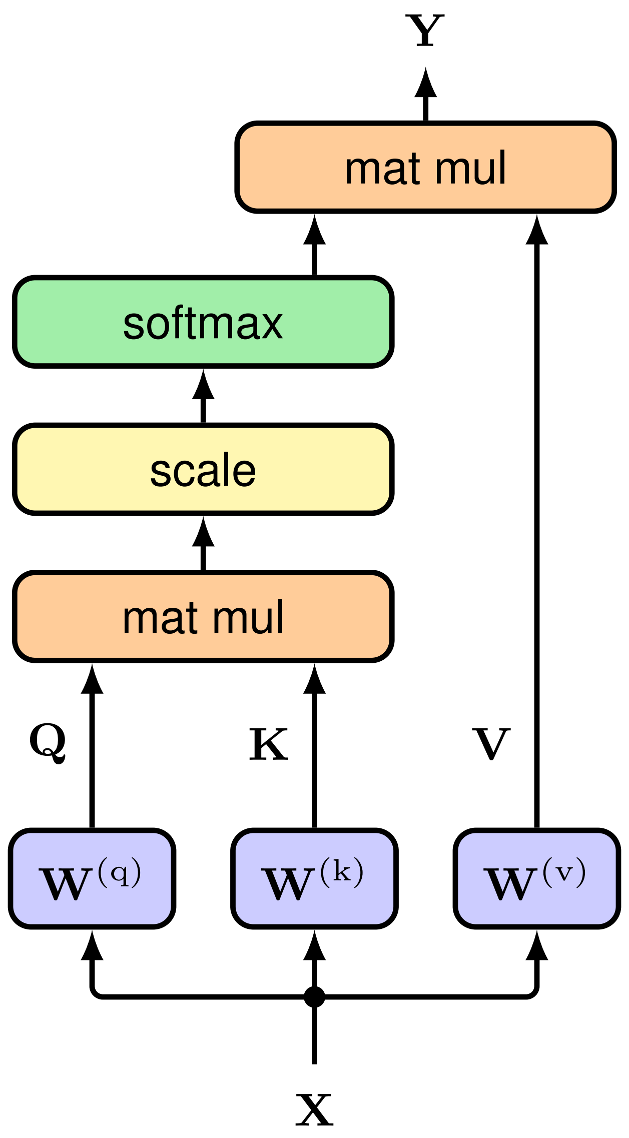
bias parameters are implicit

Scaled self-attention

- Gradient of Softmax becomes exponentially small for inputs of high magnitude, c.f. $\tanh$ or sigmoid activation; trouble with grad descent.
- Rescale the product of the query and key vectors before Softmax.
- If the elements of the query and key vectors were all independent random numbers with zero mean and unit variance, then the variance of the dot product would be D_k.
- Normalizing the argument to the softmax using the standard deviation given by $\sqrt{D_k}$:

$$\mathbf{Y} = \text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) \equiv \text{Softmax}\left[\frac{\mathbf{Q}\mathbf{K}^{\text{T}}}{\sqrt{D_{\text{k}}}}\right]\mathbf{V}.$$

- This is the **scaled dot-product self-attention**.



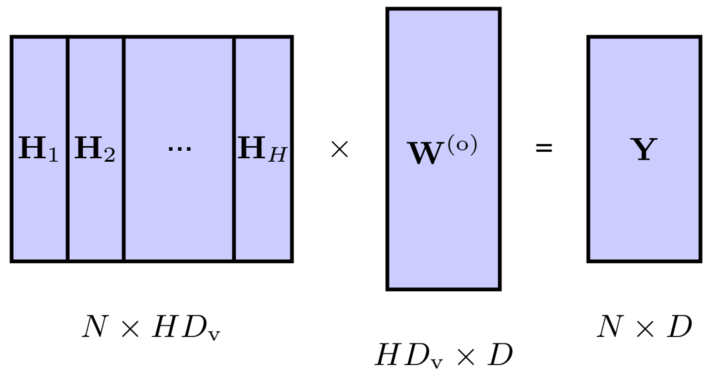
Multi-head attention

- There might be multiple patterns of attention relevant at the same time, e.g., some associated with tenses, some with vocabulary.
- Single "attention head" averages out these effects. Instead use multiple attention heads in parallel; analogous to channels in CNN.
- Suppose we have H heads indexed by $h = 1,\ldots,H$:

$$\mathbf{H}_h = \mathrm{Attention}(\mathbf{Q}_h, \mathbf{K}_h, \mathbf{V}_h)$$

- The heads are concatenated into a single matrix, and the result is then linearly transformed to give a combined output:

$$\mathbf{Y}(\mathbf{X}) = \mathrm{Concat}\left[\mathbf{H}_1, \ldots, \mathbf{H}_H\right] \mathbf{W}^{(\mathrm{o})}$$



- The matrix $\mathbf{W}^{(\mathrm{o})}$ is learned along with the weight matrices $\mathbf{W}^{(q)}, \mathbf{W}^{(k)}, \mathbf{W}^{(v)}$.