# Lecture 7: Language models (Transformers)

AI in Genetics

*ZOO6927 / BOT6935 / ZOO4926*

# Modeling biological sequences

DNA, protein, RNA sequences

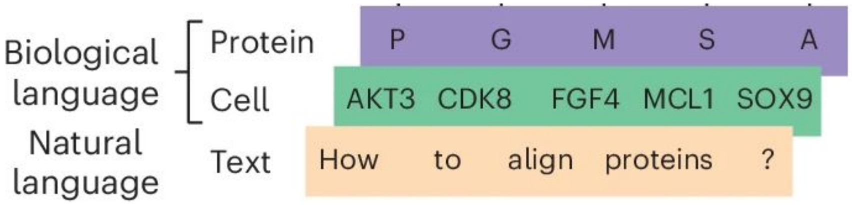
- Learning the grammar of proteins, gene regulation, genomics
- Learning the latent distribution of natural sequences
- Functional prediction and annotation
- Generating novel sequences

# Shortcomings of using CNNs to model biological sequences

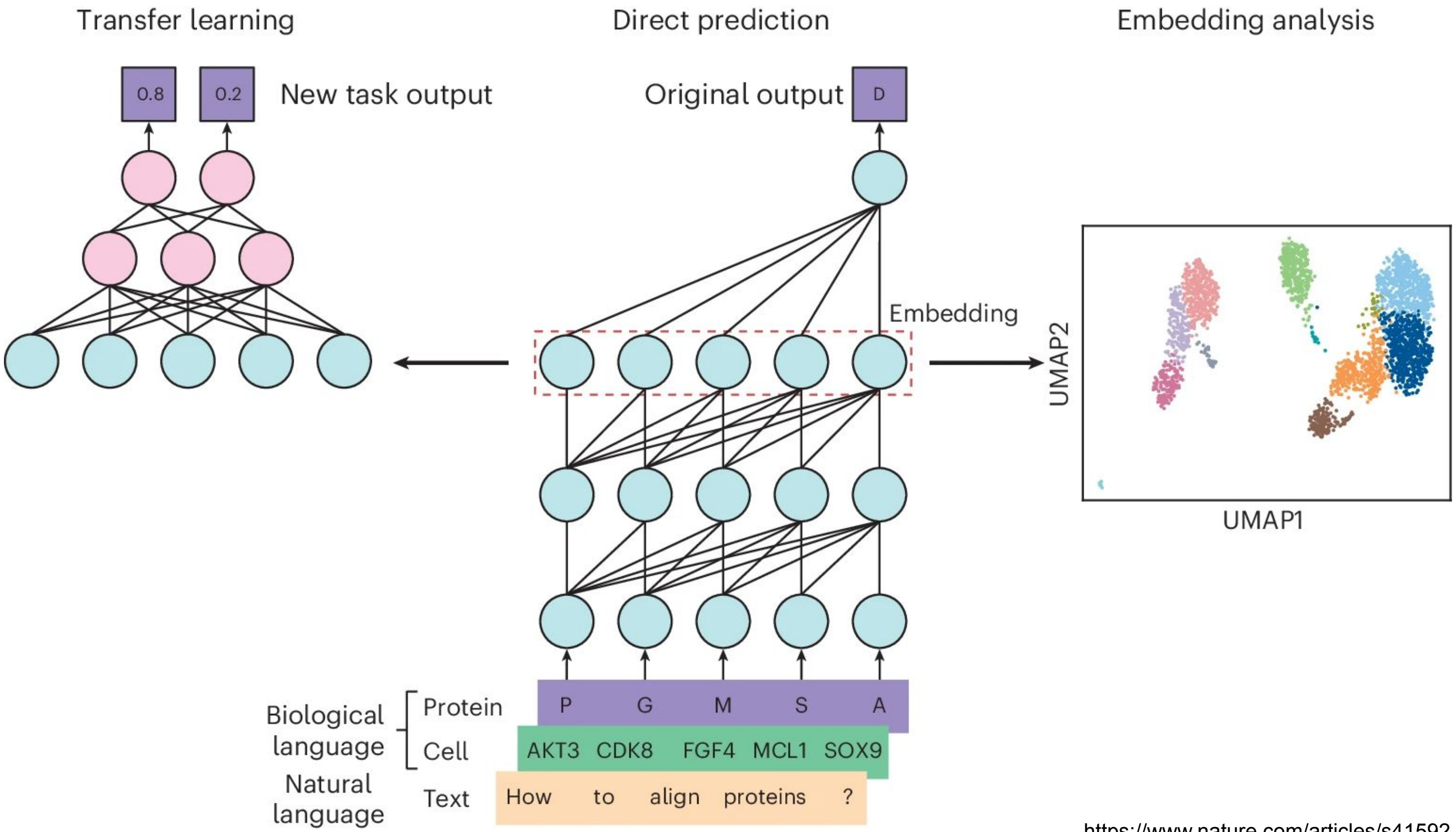
- Bad at capturing long-range dependencies

- CNN itself is not generative

# Biological sequences as a language

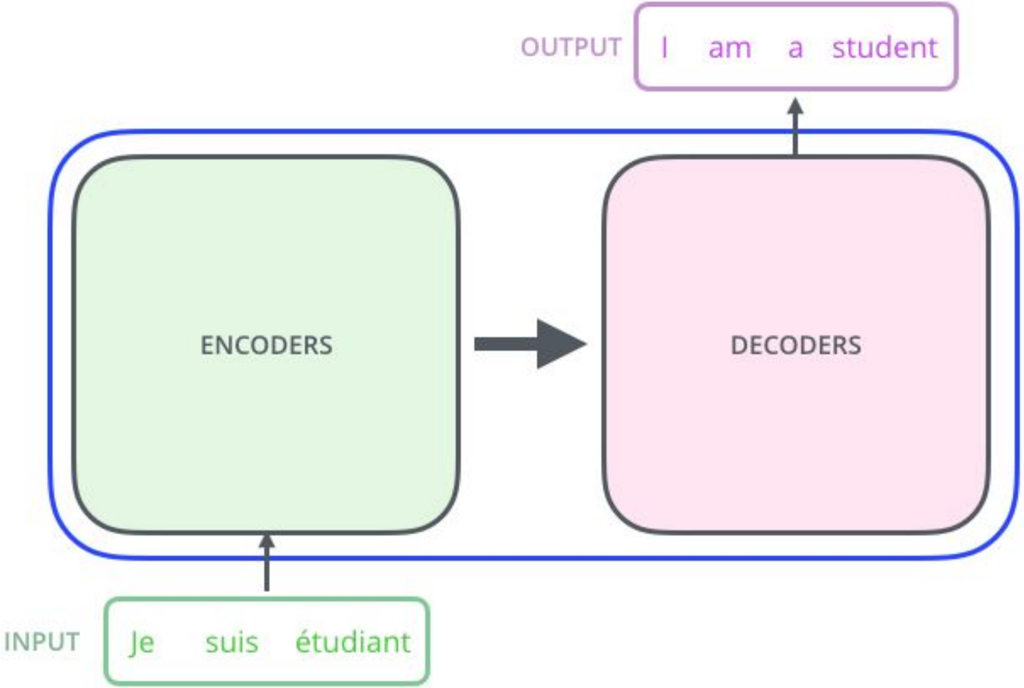


# Applications of language models in biology



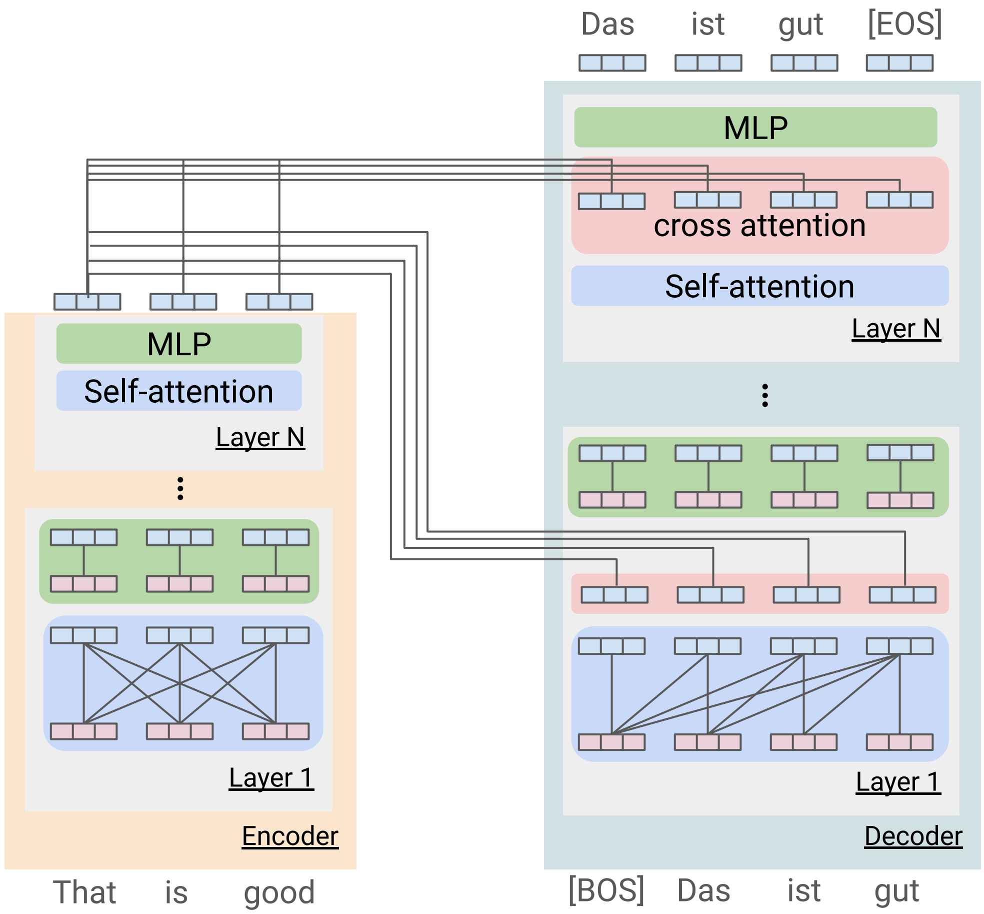

https://www.nature.com/articles/s41592-024-02354-y

# Original encoder-decoder transformer



# Encoder-decoder

**Encoder**

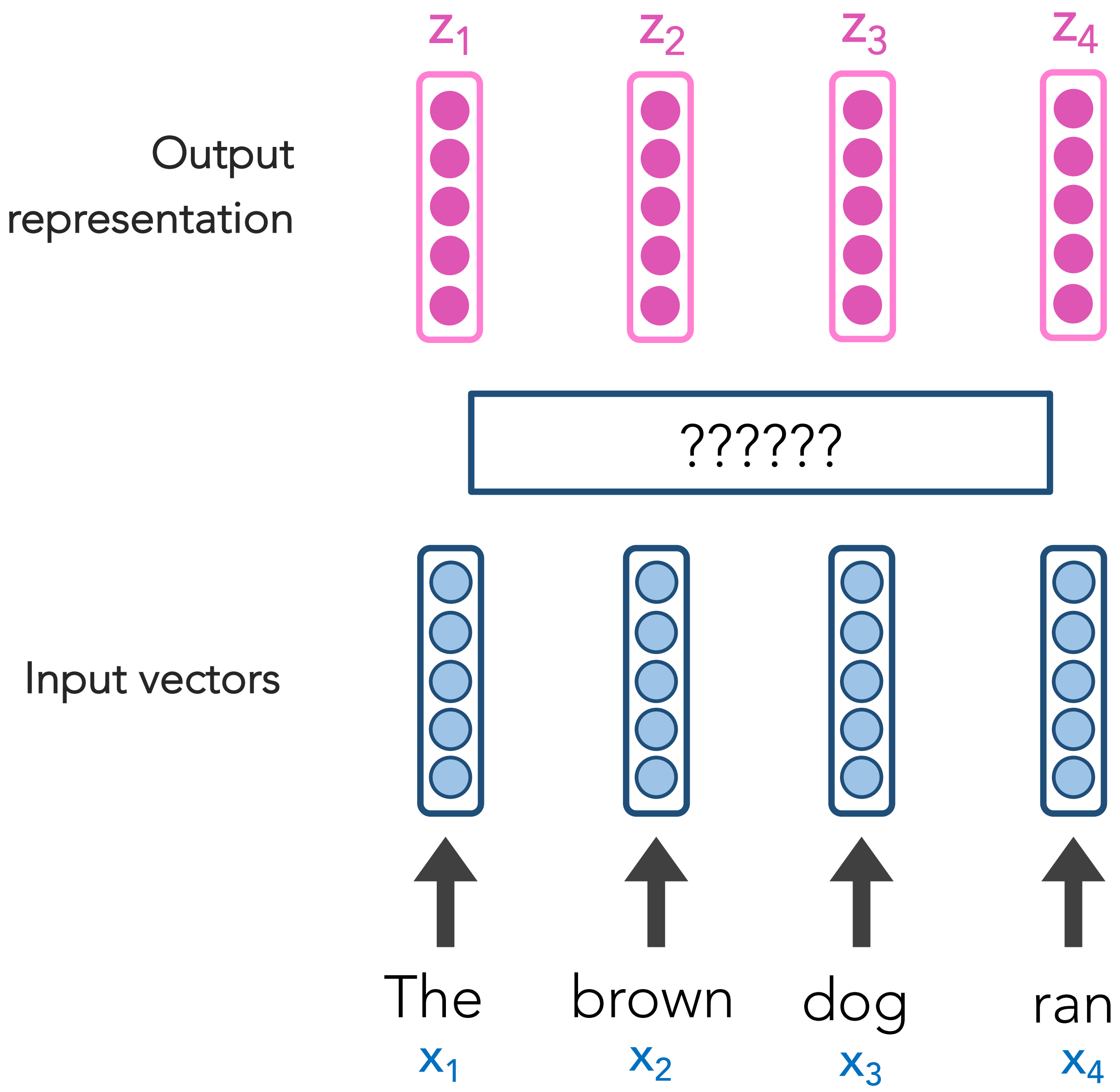



**Decoder**

# Outline

- Attention mechanism
- BERT
- ESM
- DNABert
- GPT

# Self-Attention



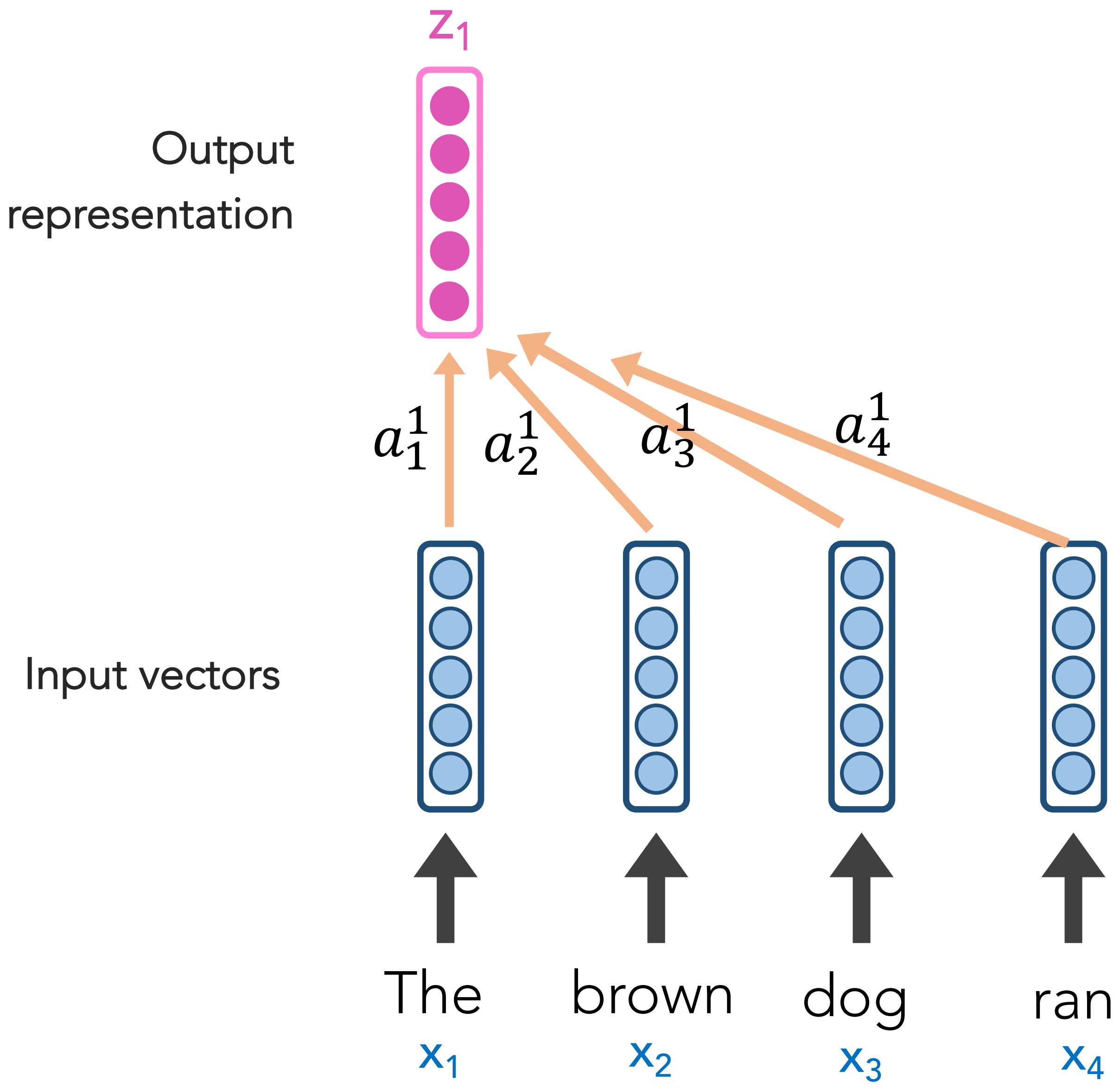

**Self-Attention**'s goal is to create great representations, $z_i$, of the input.

To capture the meaning and relationships between words in different contexts.

- Handling Polysemy (Words with Multiple Meanings)
- Capturing Long-Distance Dependencies
- Preserving Idiomatic Expressions
- Managing Syntax and Grammatical Structures

...

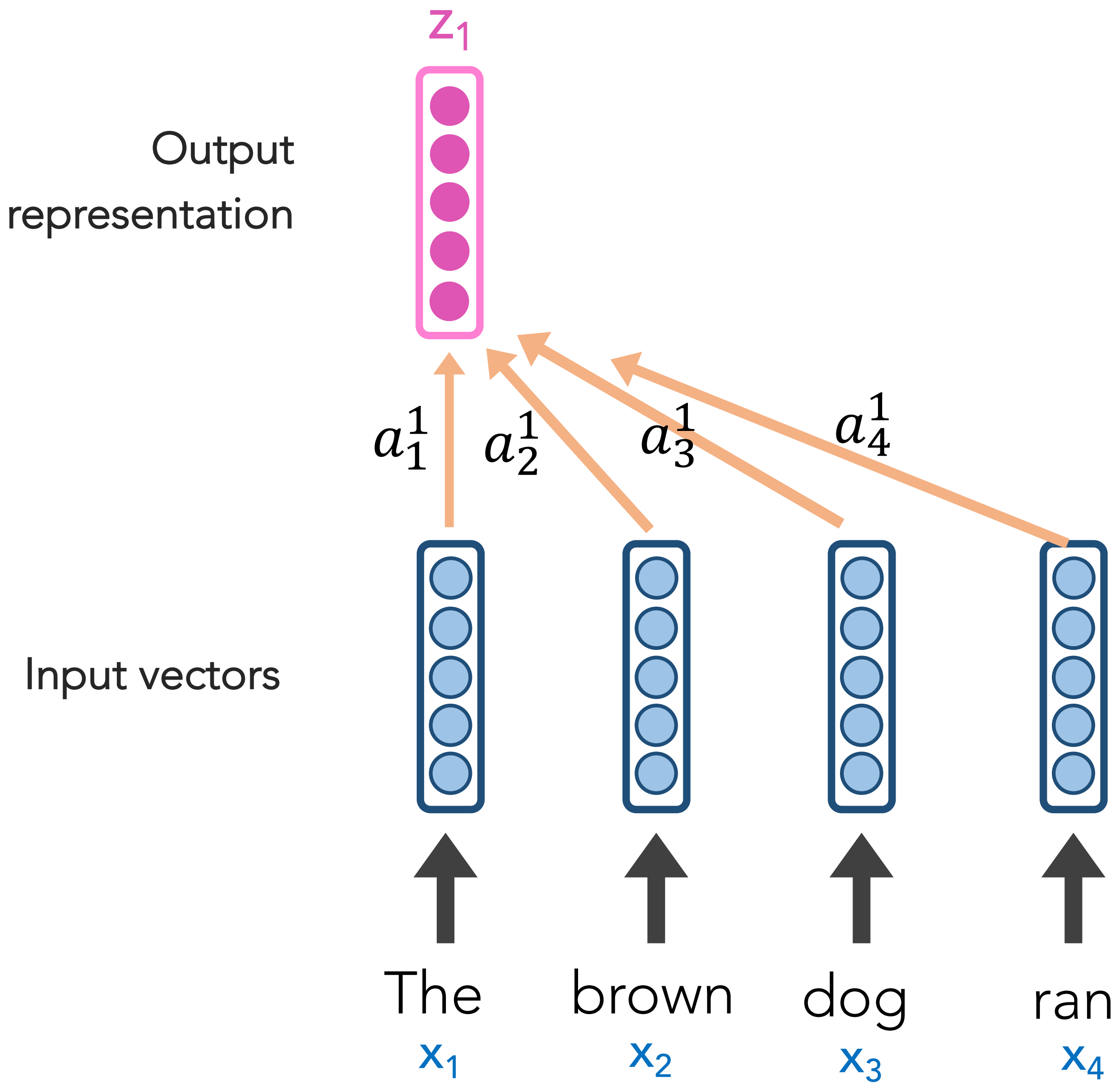
# Self-Attention




**Self-Attention**'s goal is to create great representations, $z_i$, of the input

$z_1$ will be based on a weighted contribution of $x_1$, $x_2$, $x_3$, $x_4$

# Self-Attention

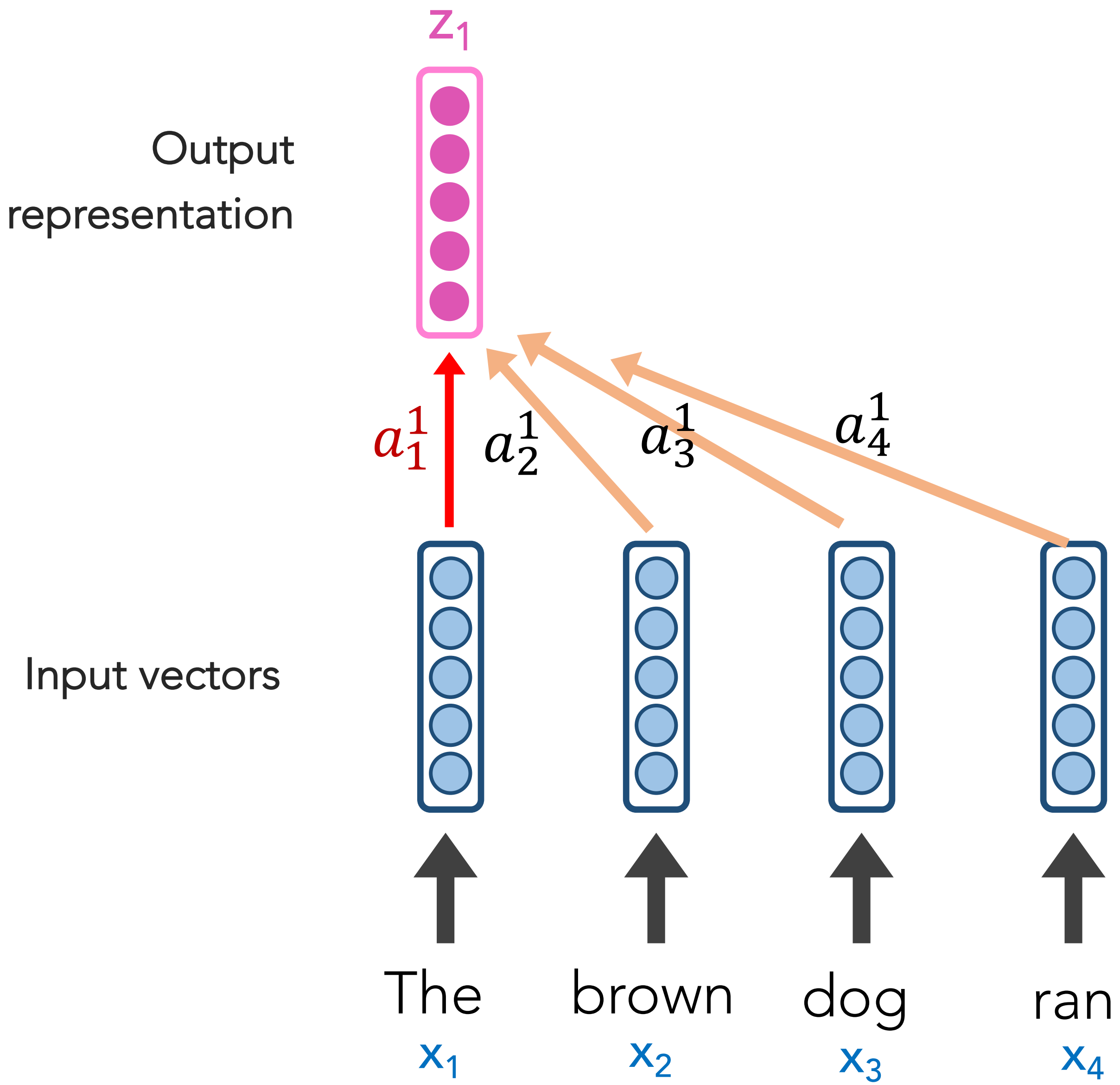



**Self-Attention**'s goal is to create great representations, $z_i$, of the input

$z_1$ will be based on a weighted contribution of $x_1$, $x_2$, $x_3$, $x_4$

$a_i^1$ is "just" a weight. More is happening under the hood, but it's effectively weighting *versions* of $x_1$, $x_2$, $x_3$, $x_4$

# Self-Attention




Under the hood, each $x_i$ has 3 small, associated vectors.

For example, $x_1$ has:

- Query $\mathbf{q}_i$
- Key $\mathbf{k}_i$
- Value $\mathbf{v}_i$

# Self-Attention

Step 1: Our Self-Attention Head I has just 3 weight matrices $W_q$, $W_k$, $W_v$ in total. These same 3 weight matrices are multiplied by each $x_i$ to create all vectors:

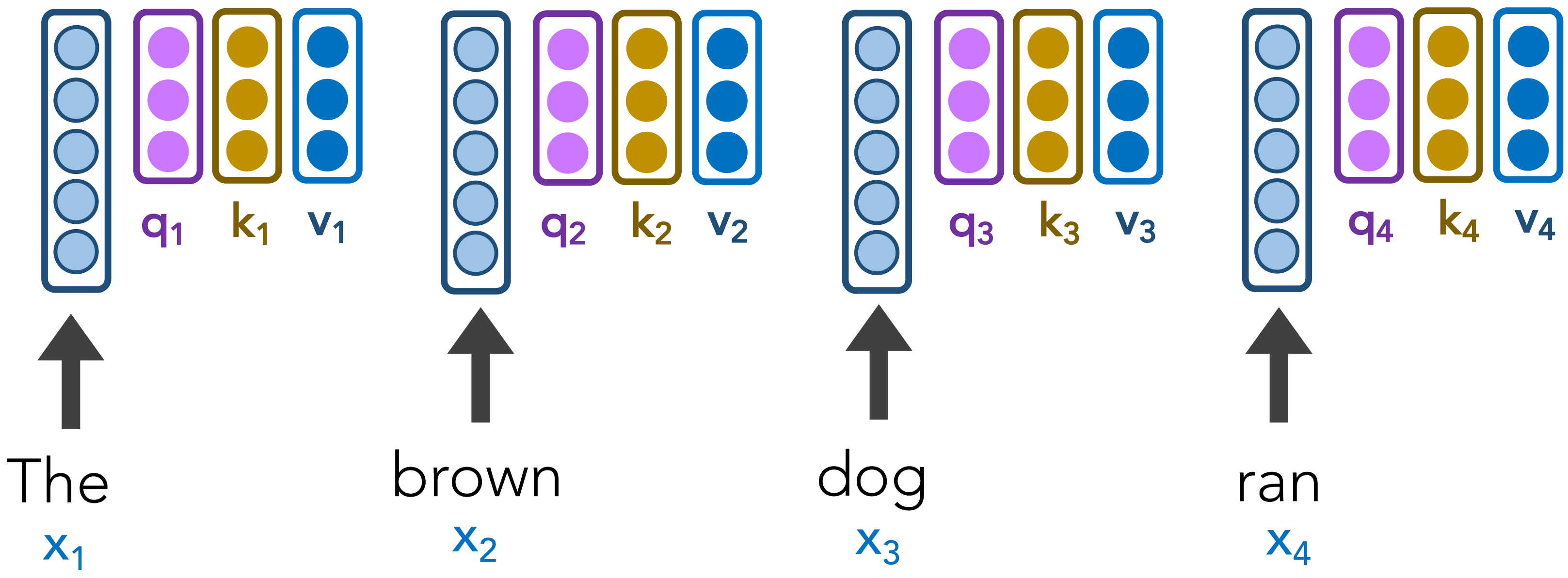
$$q_i = W_q x_i$$

$$k_i = W_k x_i$$

$$v_i = W_v x_i$$

Under the hood, each $x_i$ has 3 small, associated vectors. For example, $x_1$ has:

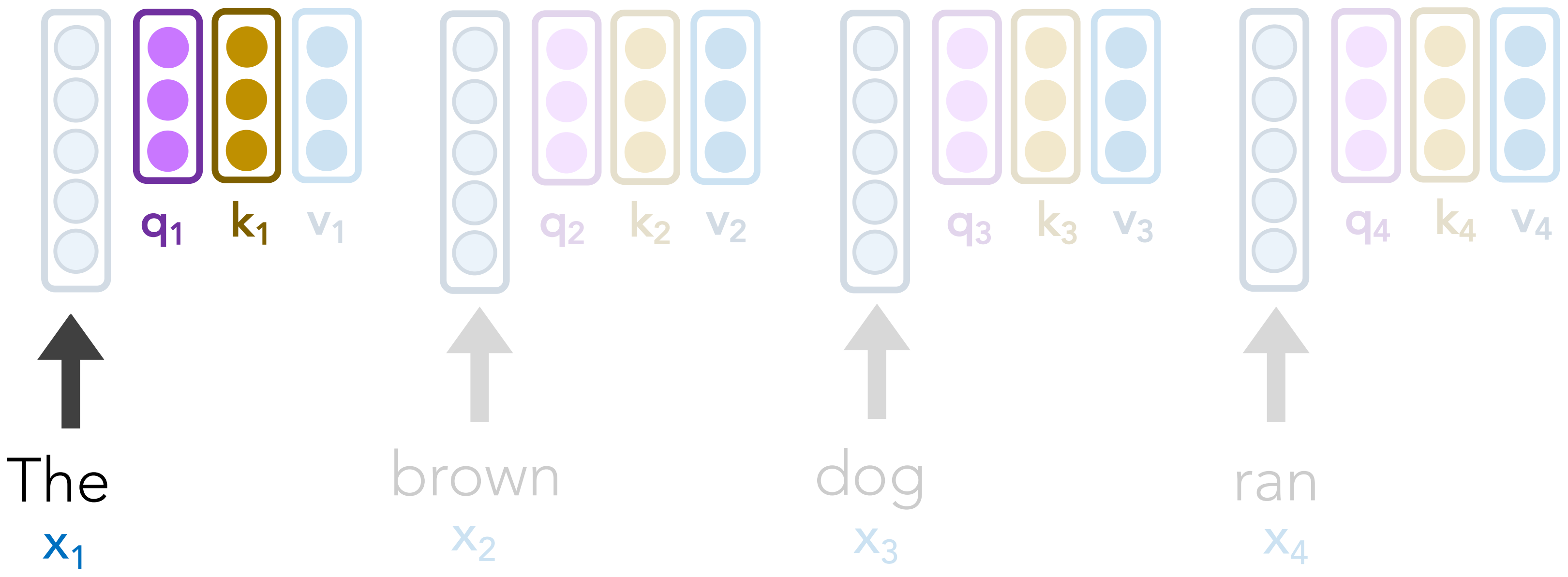
- Query $\mathbf{q_1}$
- Key $\mathbf{k_1}$
- Value $\mathbf{v_1}$



# Self-Attention

Step 2: For word $x_1$, let's calculate the scores $s_1$, $s_2$, $s_3$, $s_4$, which represent how much attention to pay to each respective "word" $v_i$

$s_1 = q_1 \cdot k_1 = 112$

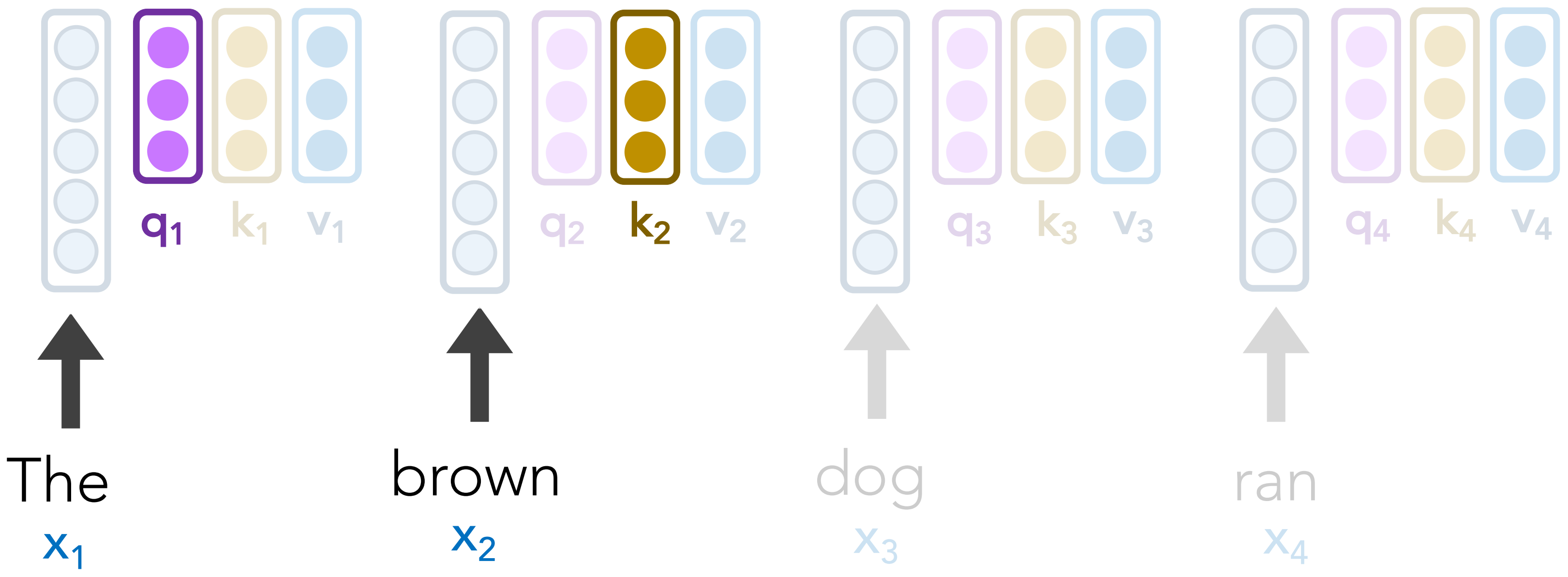


# Self-Attention

Step 2: For word $x_1$, let's calculate the scores $s_1$, $s_2$, $s_3$, $s_4$, which represent how much attention to pay to each respective "word" $v_i$

$s_2 = q_1 \cdot k_2 = 96$

$s_1 = q_1 \cdot k_1 = 112$



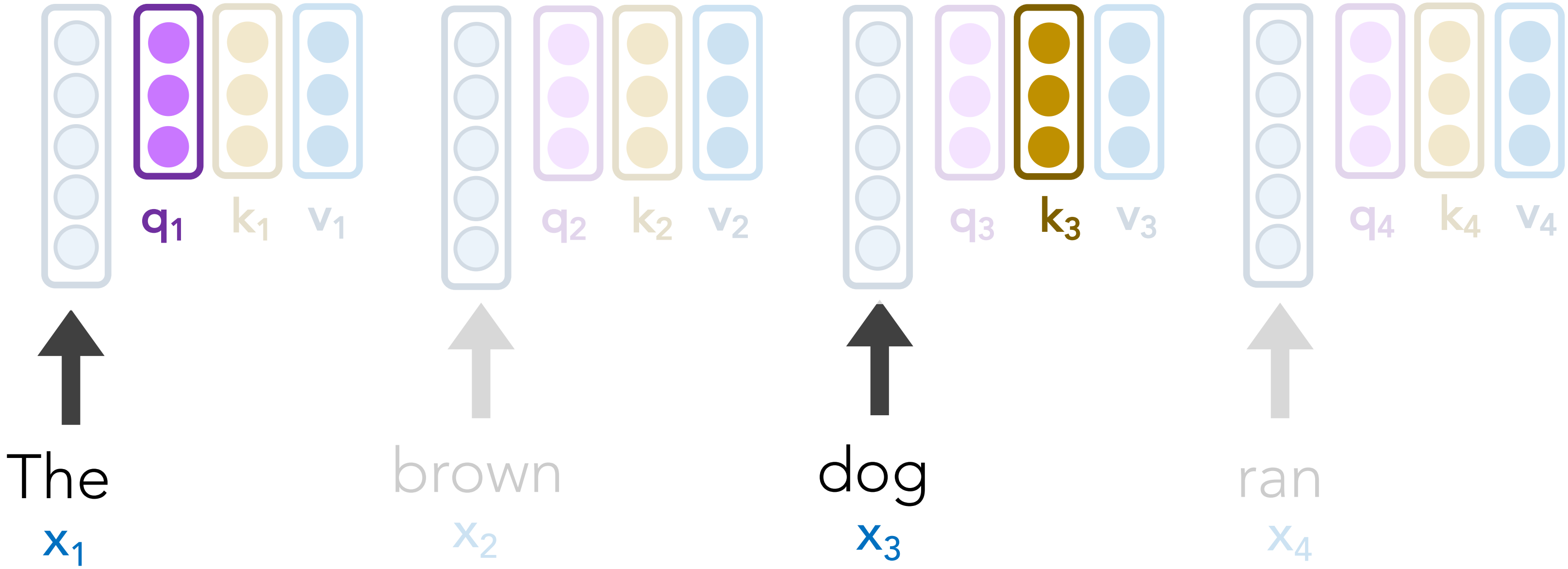
# Self-Attention

Step 2: For word $x_1$, let's calculate the scores $s_1, s_2, s_3, s_4$, which represent how much attention to pay to each respective "word" $v_i$

$s_3 = q_1 \cdot k_3 = 16$

$s_2 = q_1 \cdot k_2 = 96$

$s_1 = q_1 \cdot k_1 = 112$



# Self-Attention

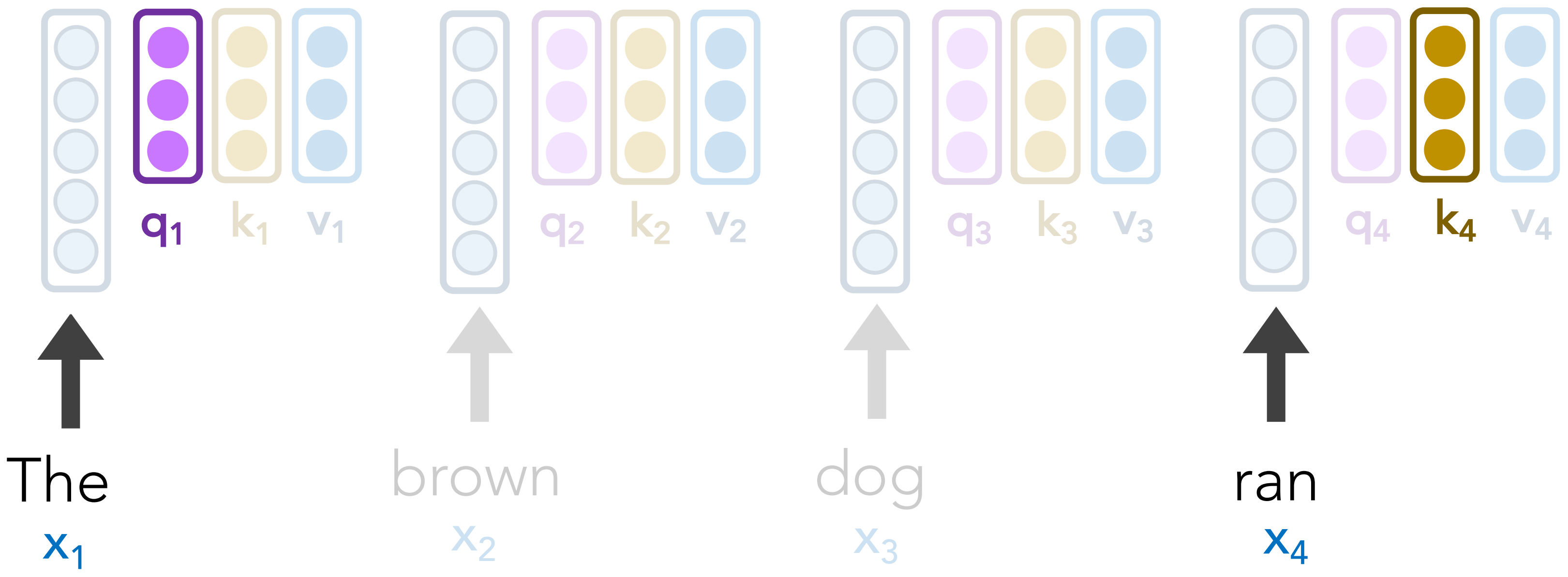
**Step 2:** For word $x_1$, let's calculate the scores $s_1, s_2, s_3, s_4$, which represent how much attention to pay to each respective "word" $v_i$

$s_4 = q_1 \cdot k_4 = 8$

$s_3 = q_1 \cdot k_3 = 16$

$s_2 = q_1 \cdot k_2 = 96$

$s_1 = q_1 \cdot k_1 = 112$



# Self-Attention

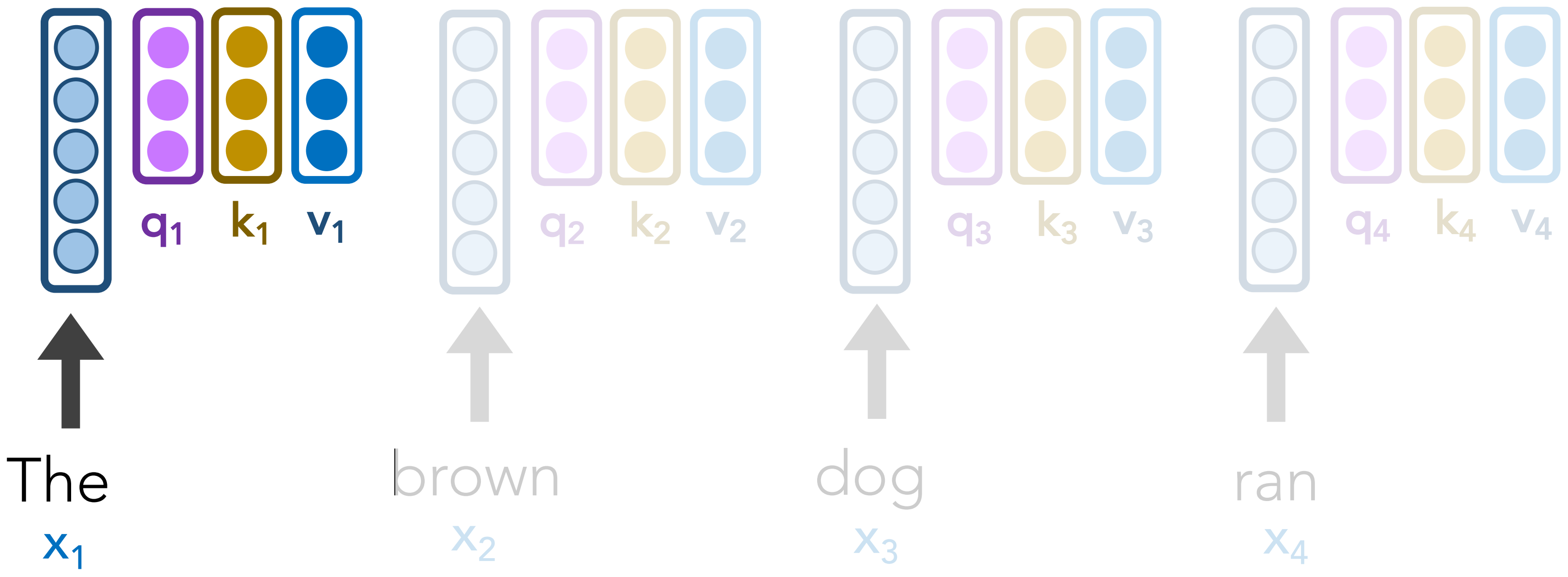
Step 3: Our scores $s_1, s_2, s_3, s_4$ don't sum to 1. Let's divide by $\sqrt{len(k_i)}$ and softmax it

$s_4 = q_1 \cdot k_4 = 8$ $\qquad a_4 = \sigma(s_4/8) = 0$

$s_3 = q_1 \cdot k_3 = 16$ $\qquad a_3 = \sigma(s_3/8) = .01$

$s_2 = q_1 \cdot k_2 = 96$ $\qquad a_2 = \sigma(s_2/8) = .12$

$s_1 = q_1 \cdot k_1 = 112$ $\qquad a_1 = \sigma(s_1/8) = .87$



# Self-Attention

**Step 3:** Our scores $s_1$, $s_2$, $s_3$, $s_4$ don't sum to 1. Let's divide by $\sqrt{len(k_i)}$ and softmax it

$s_4 = q_1 \cdot k_4 = 8$

$s_3 = q_1 \cdot k_3 = 16$

$s_2 = q_1 \cdot k_2 = 96$

$s_1 = q_1 \cdot k_1 = 112$

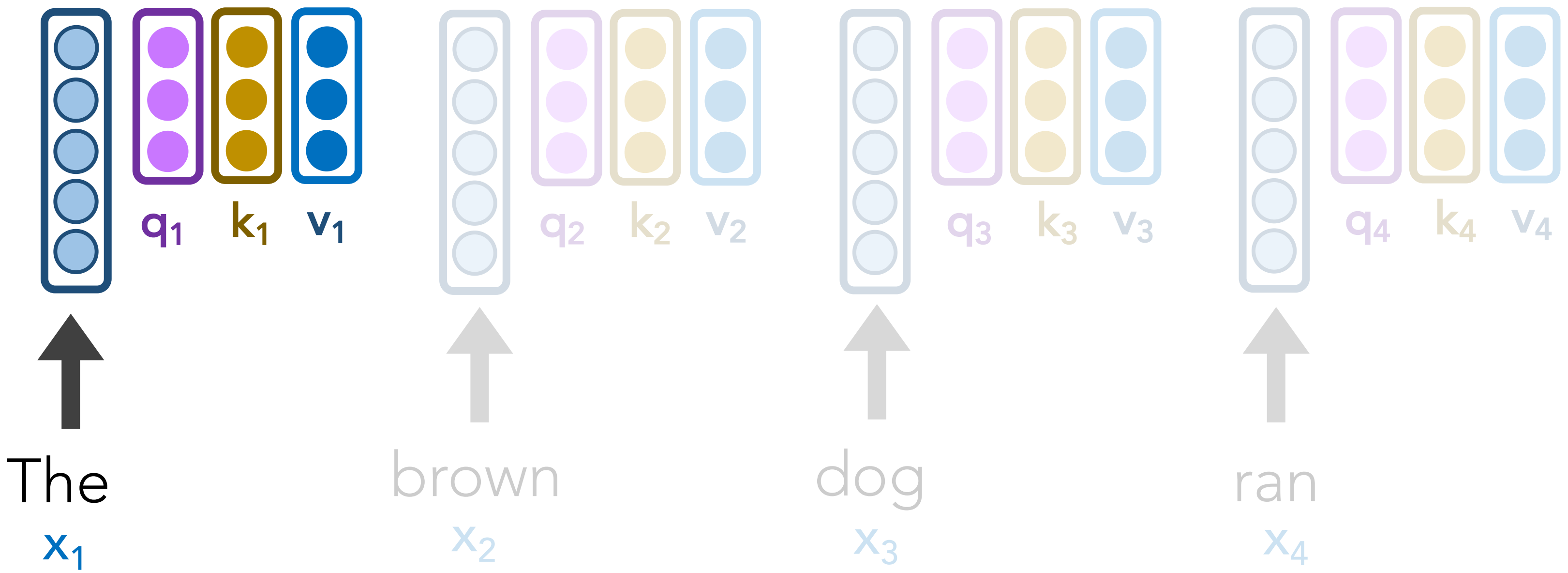
$a_4 = \sigma(s_4/8) = 0$

$a_3 = \sigma(s_3/8) = .01$

$a_2 = \sigma(s_2/8) = .12$

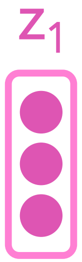
$a_1 = \sigma(s_1/8) = .87$

Instead of these $a_i$ values directly weighting our original $x_i$ word vectors, they directly weight our $v_i$ vectors.



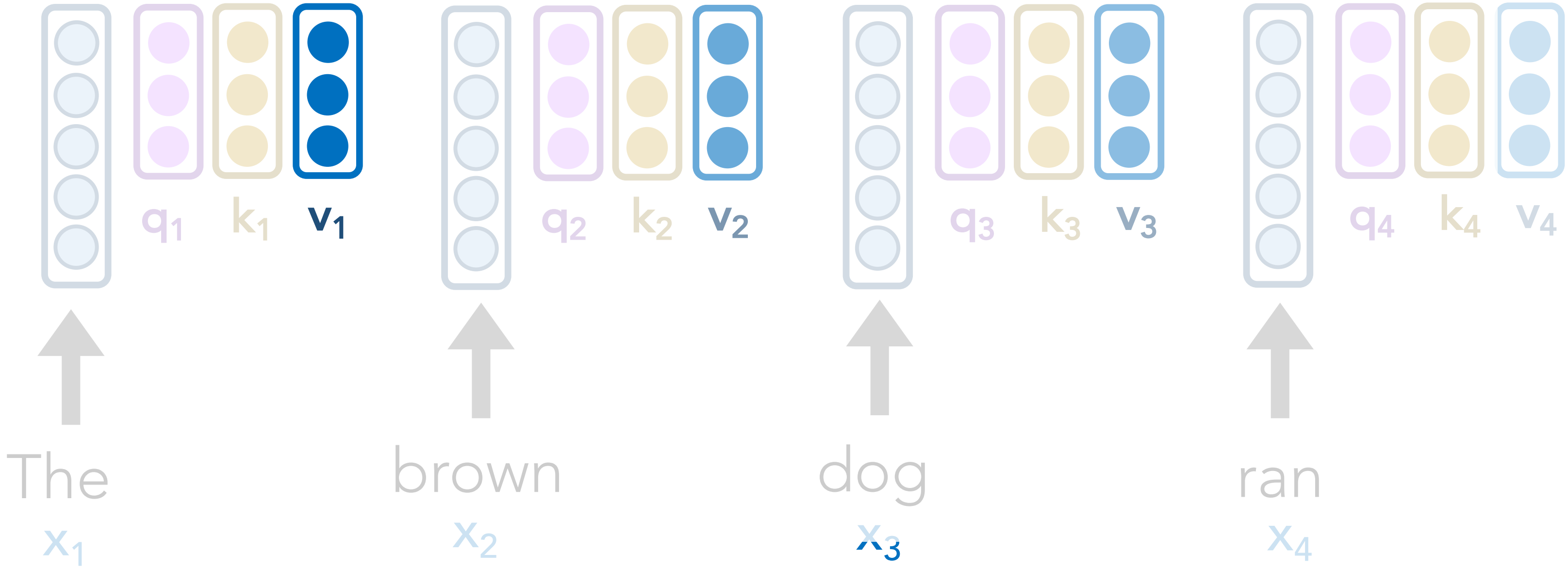
# Self-Attention

Step 4: Let's weight our $v_i$ vectors and simply sum them up!



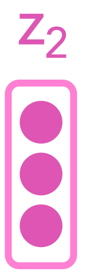

$$z_1 = a_1 \cdot v_1 + a_2 \cdot v_2 + a_3 \cdot v_3 + a_4 \cdot v_4$$

$$= 0.87 \cdot v_1 + 0.12 \cdot v_2 + 0.01 \cdot v_3 + 0 \cdot v_4$$

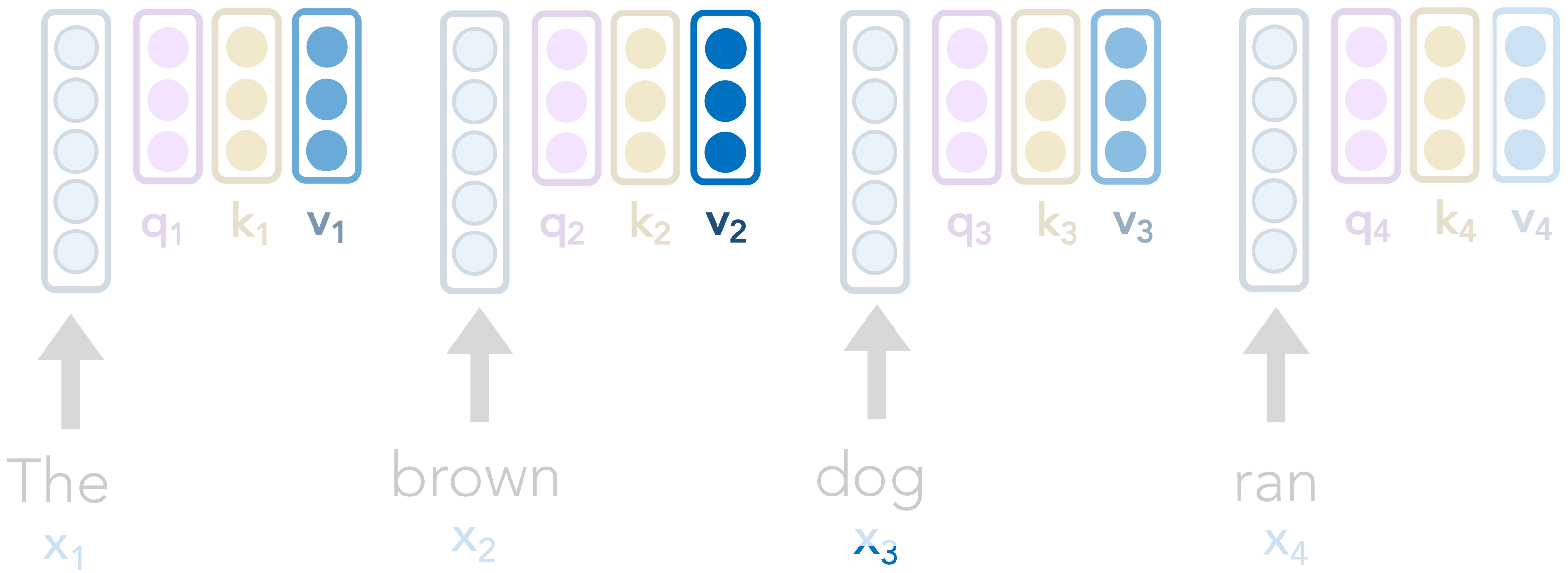


# Self-Attention

**Step 5:** We repeat this for all other words, yielding us with great, new $z_i$ representations!

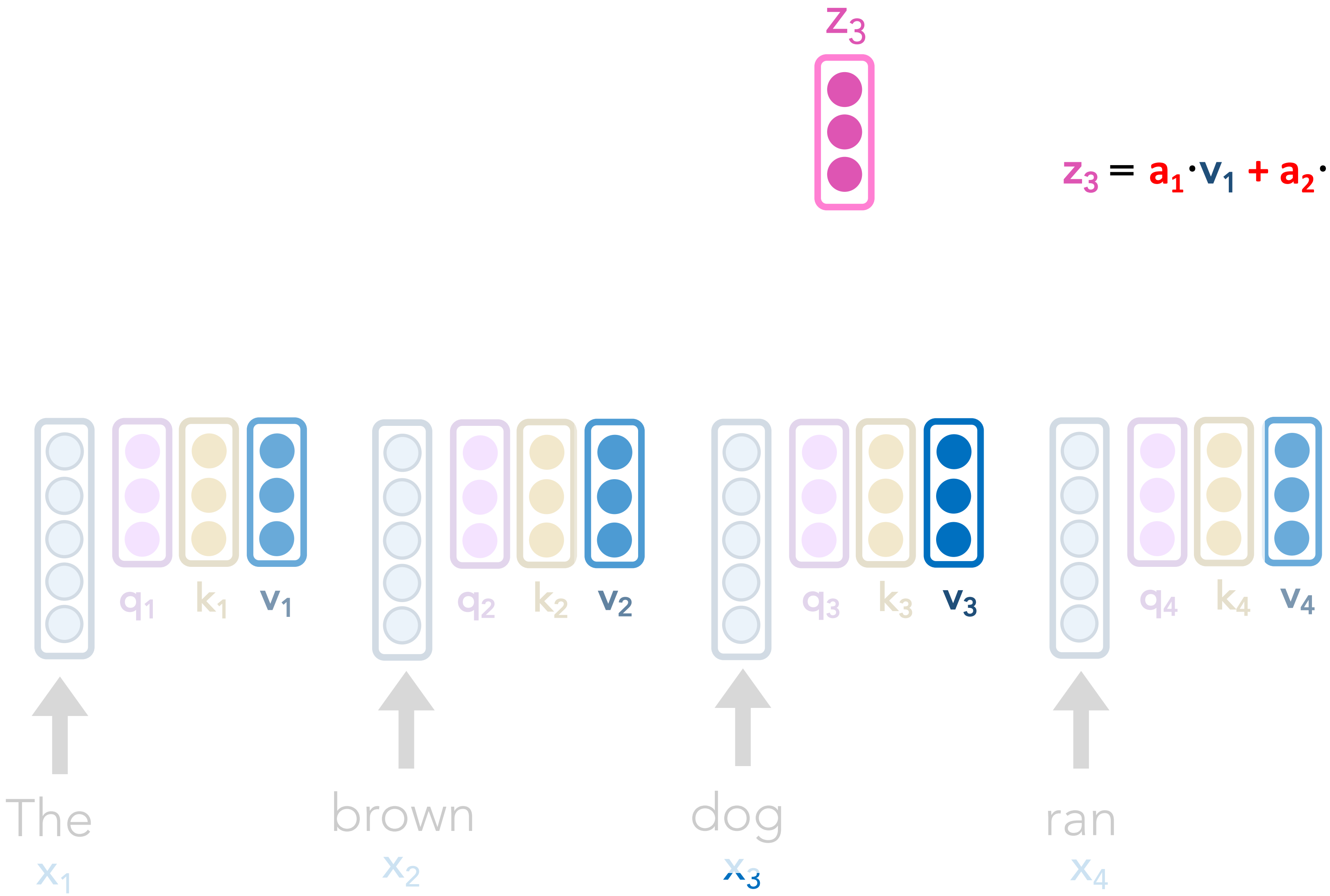



$$z_2 = a_1 \cdot v_1 + a_2 \cdot v_2 + a_3 \cdot v_3 + a_4 \cdot v_4$$



# Self-Attention

Step 5: We repeat this for all other words, yielding us with great, new $z_i$ representations!

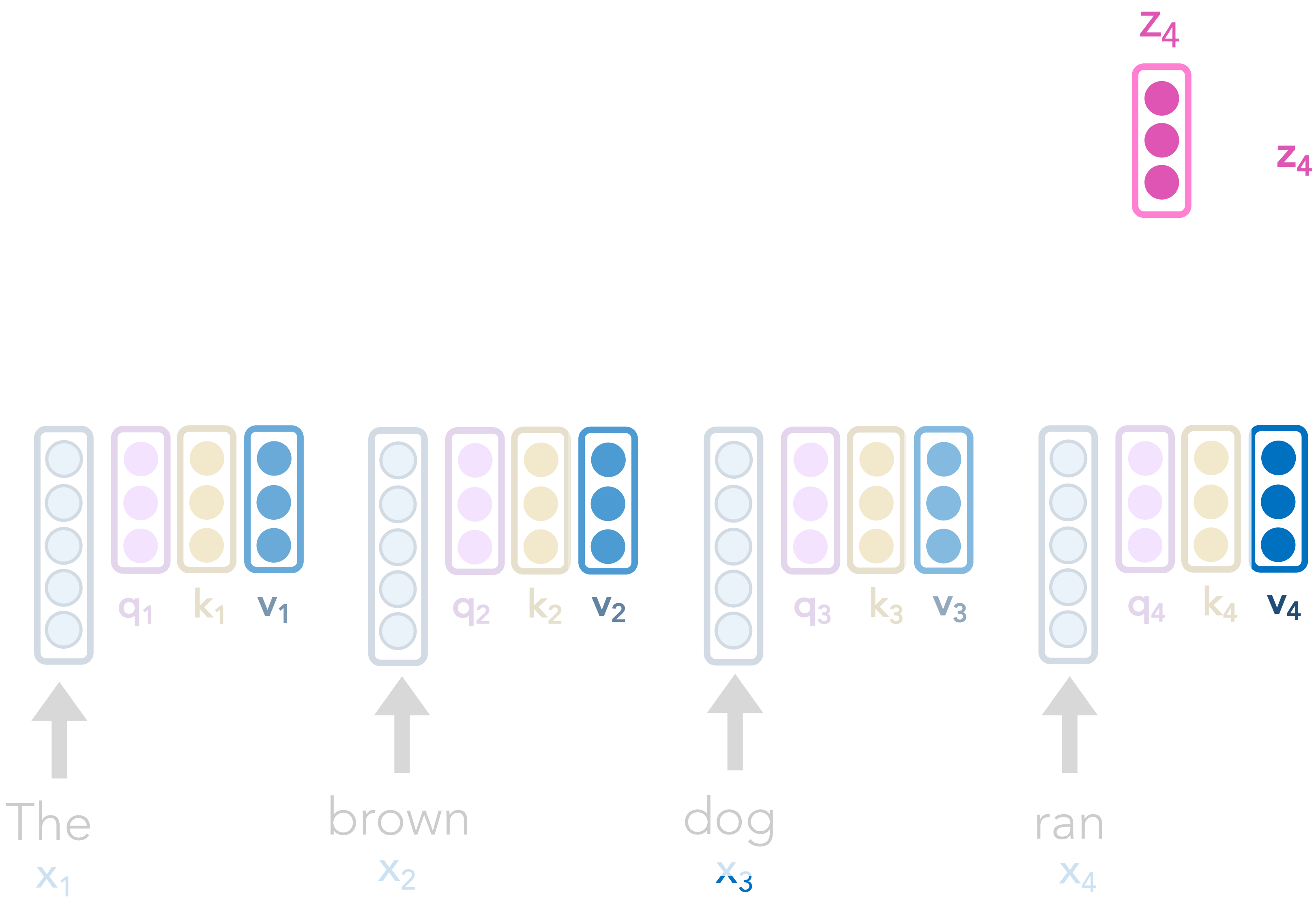



$$z_3 = a_1 \cdot v_1 + a_2 \cdot v_2 + a_3 \cdot v_3 + a_4 \cdot v_4$$

# Self-Attention

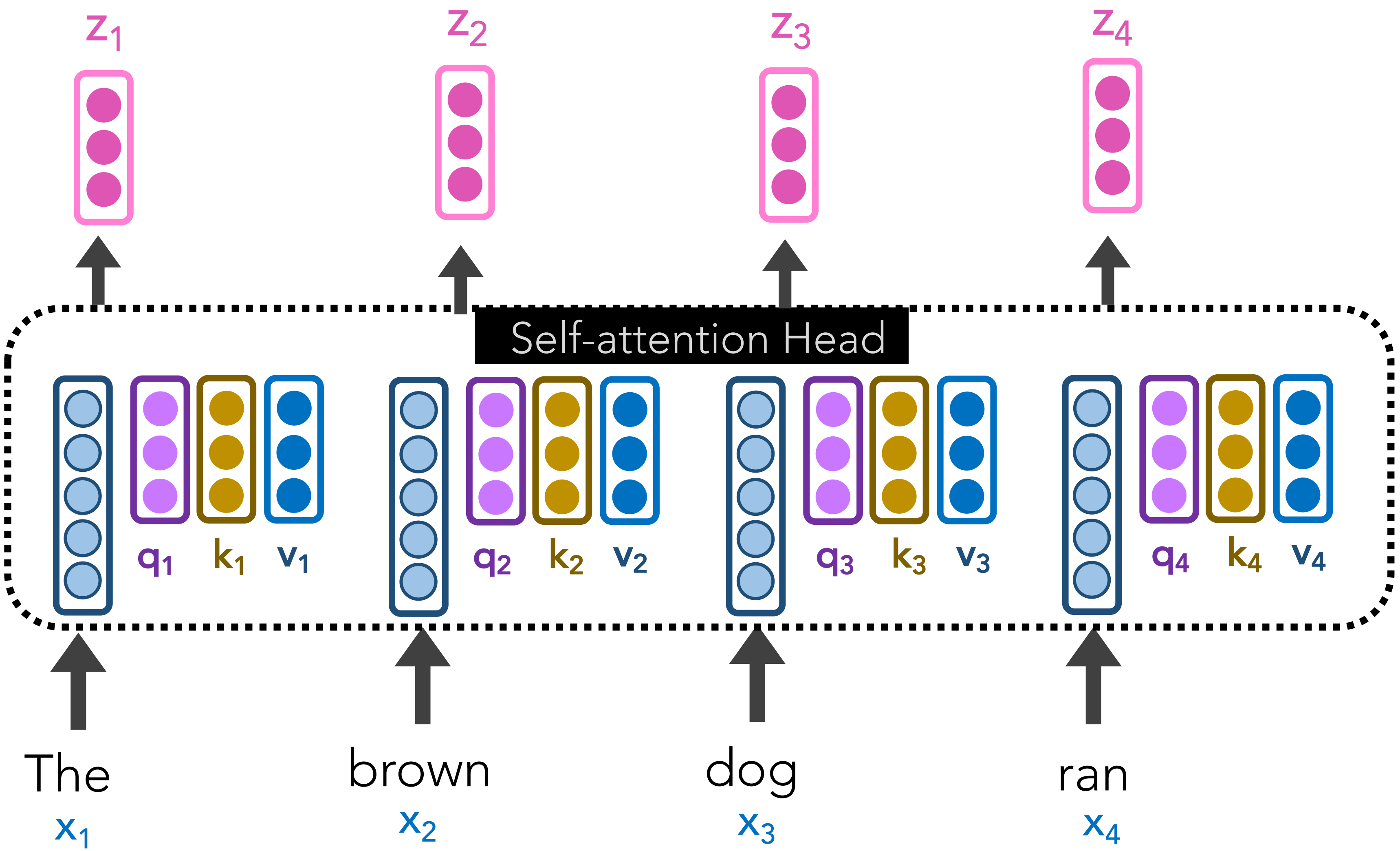
Step 5: We repeat this for all other words, yielding us with great, new $z_i$ representations!

$$z_4 = a_1 \cdot v_1 + a_2 \cdot v_2 + a_3 \cdot v_3 + a_4 \cdot v_4$$



# Self-Attention

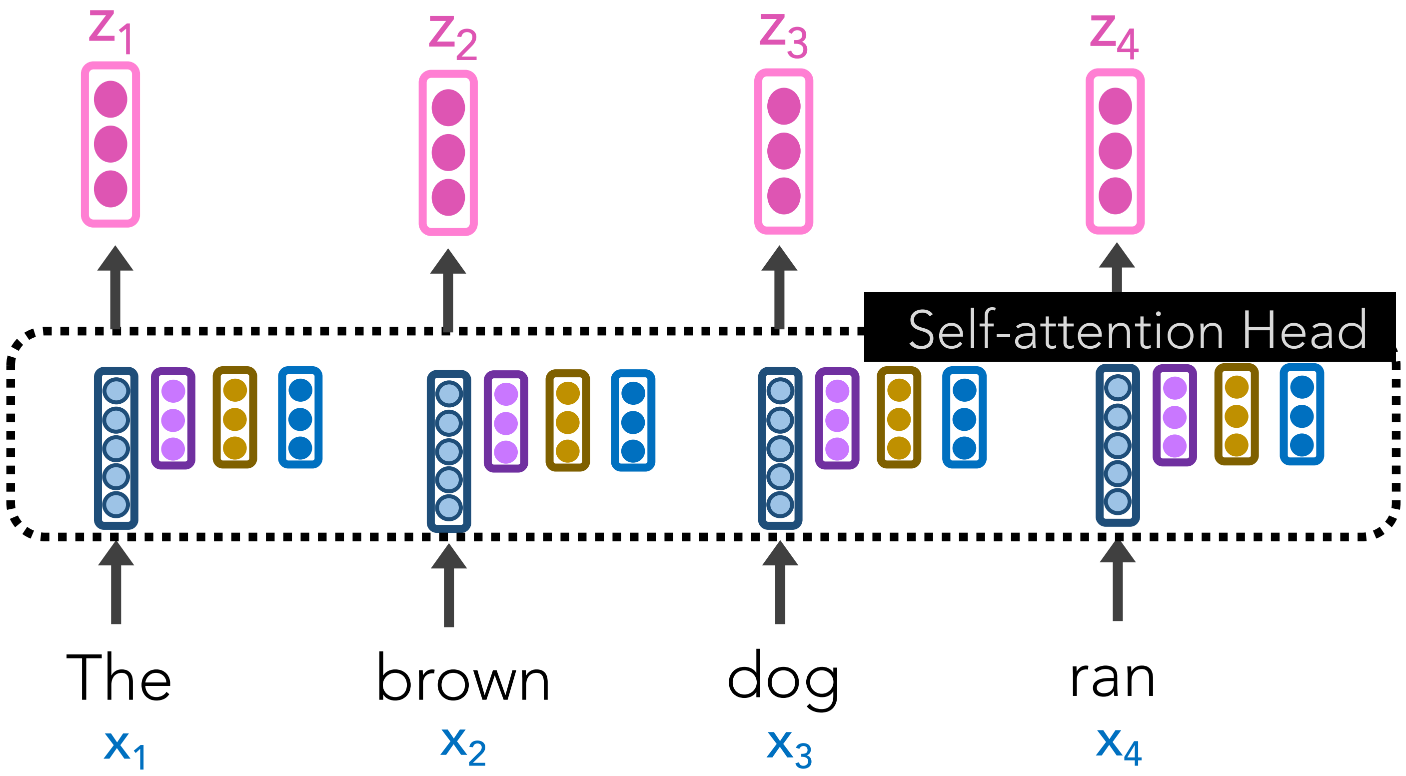
Tada! Now we have great, new representations $z_i$ via a self-attention head



Takeaway:

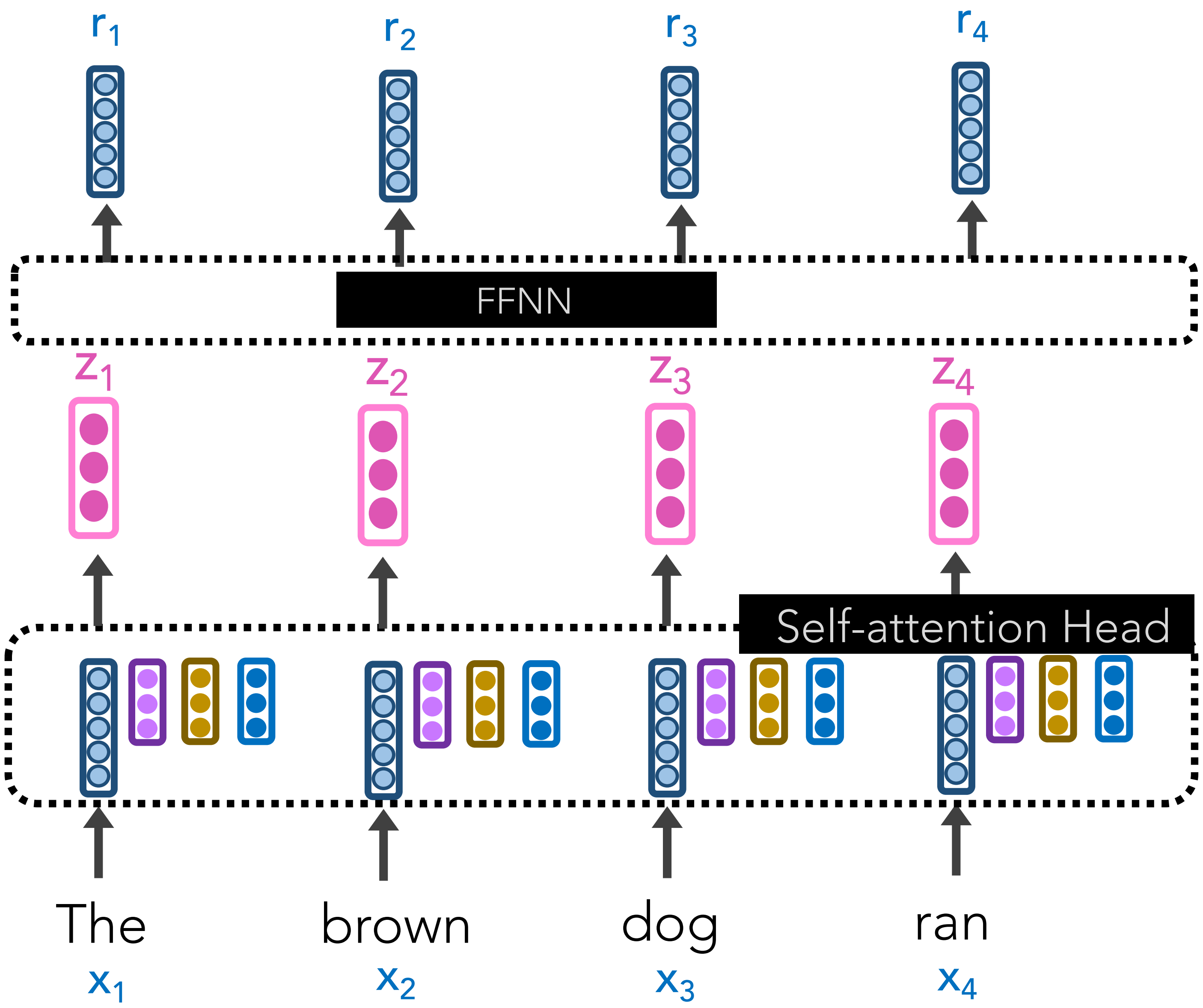
Self-Attention is powerful; allows us to create great, context-aware representations

# Self-Attention



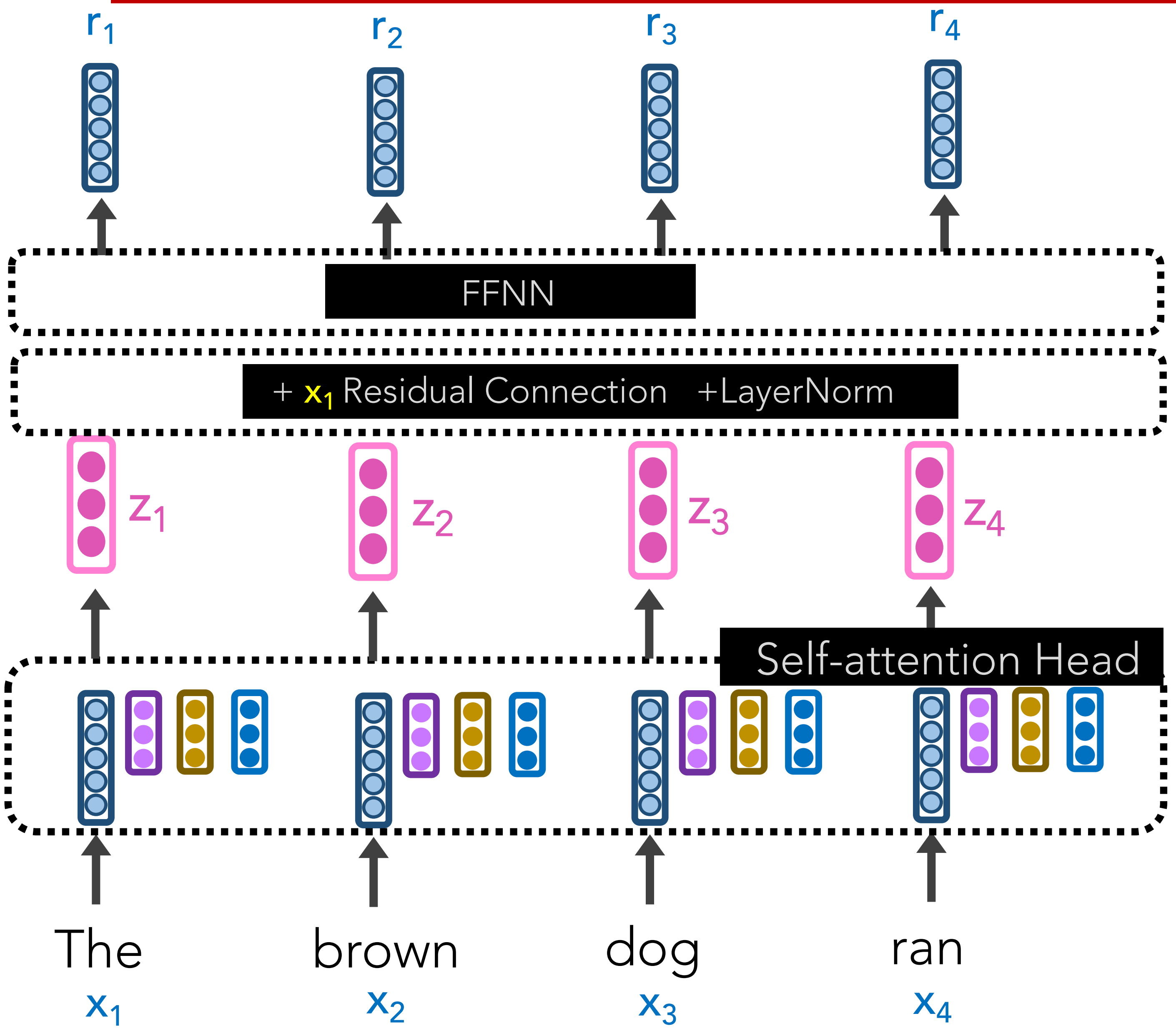

Let's further pass each $z_i$ through a FFNN

# Self-Attention + FFNN




Let's further pass each $z_i$ through a FFNN

# Self-Attention + FFNN

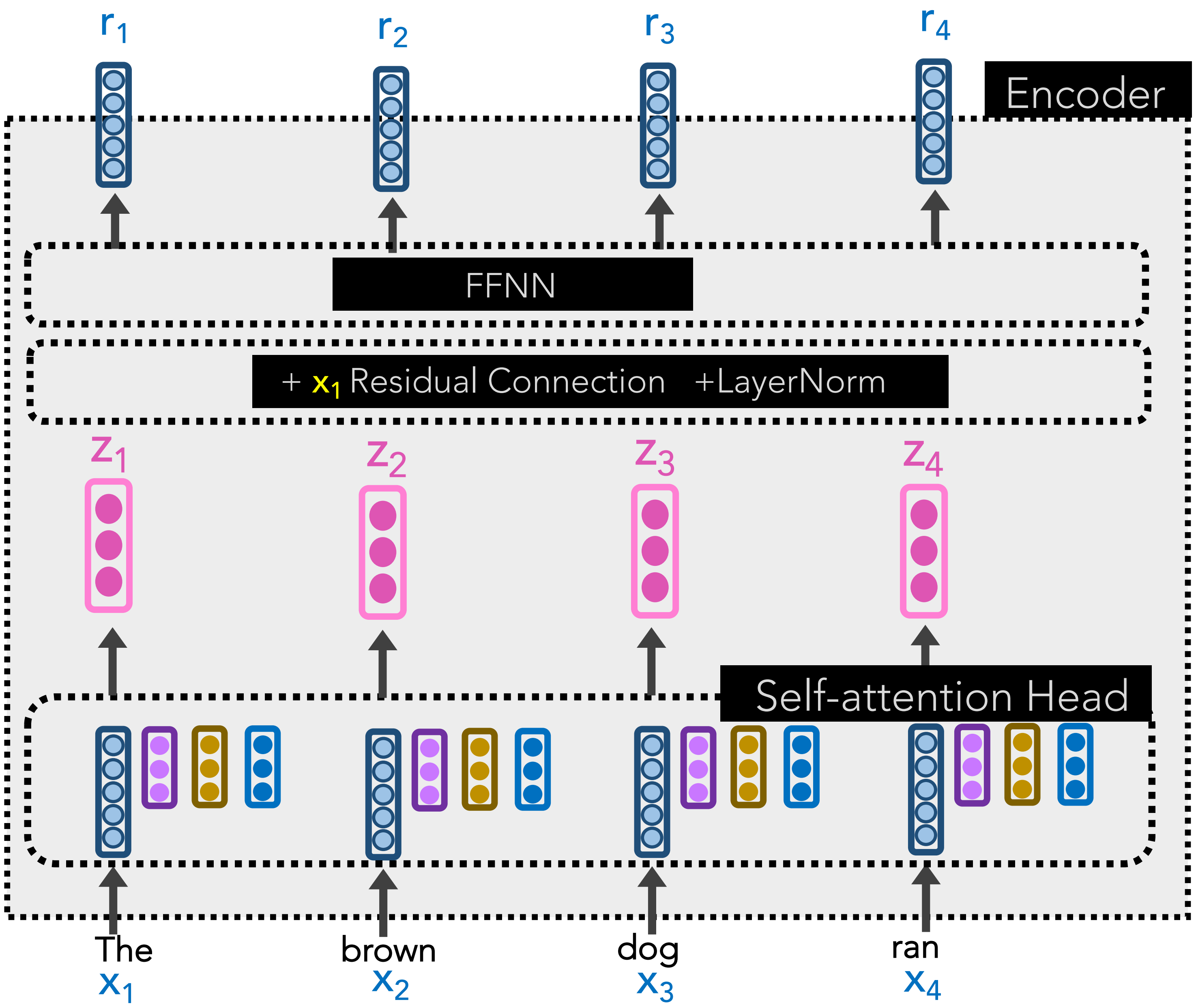



Let's further pass each $z_i$ through a FFNN

We concat w/ a residual connection to help ensure relevant info is getting forward passed.

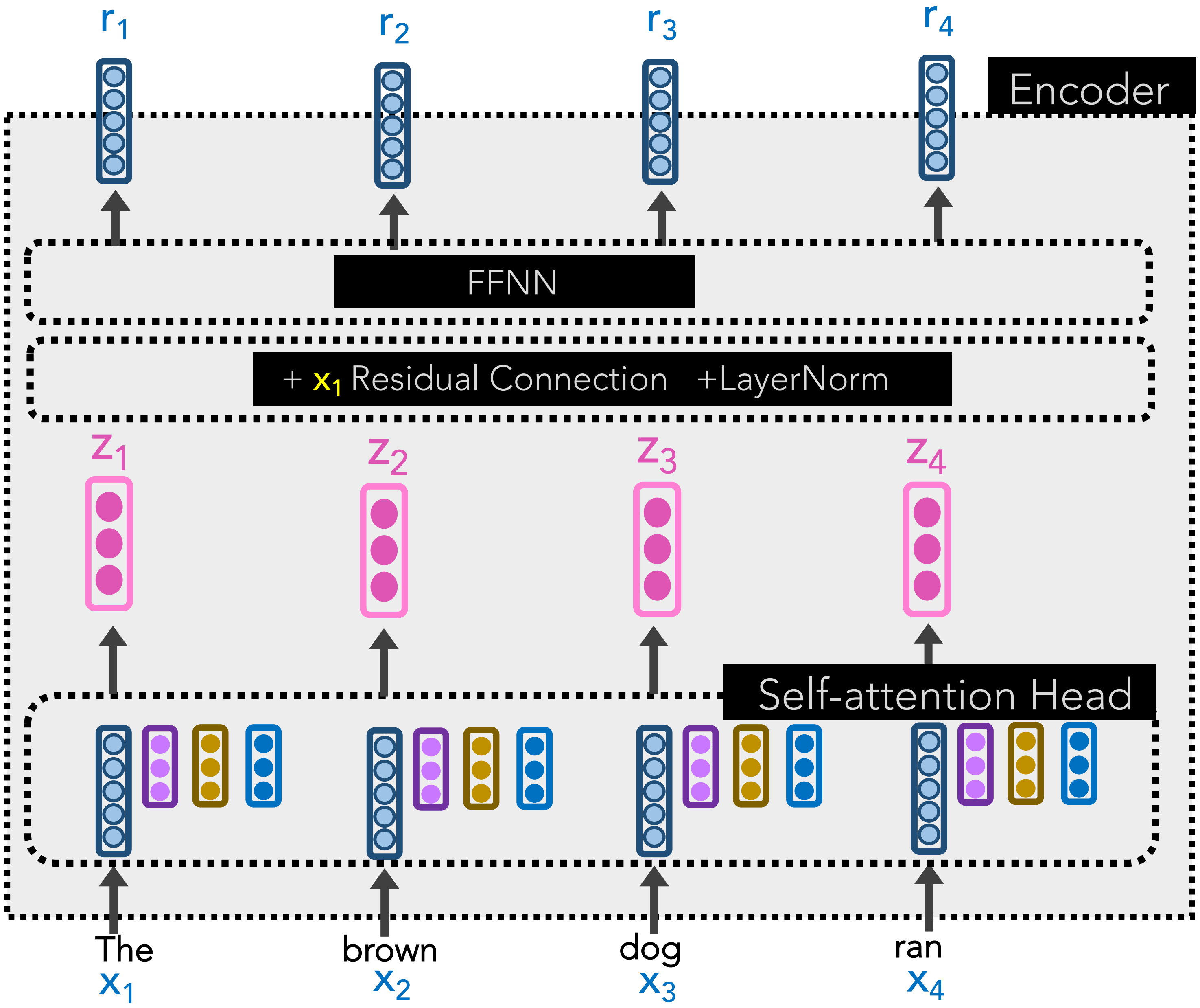
We perform LayerNorm to stabilize the network and allow for proper gradient flow.

# Transformer Encoder




Yay! Our $r_i$ vectors are our new representations, and this entire process is called a **Transformer Encoder**

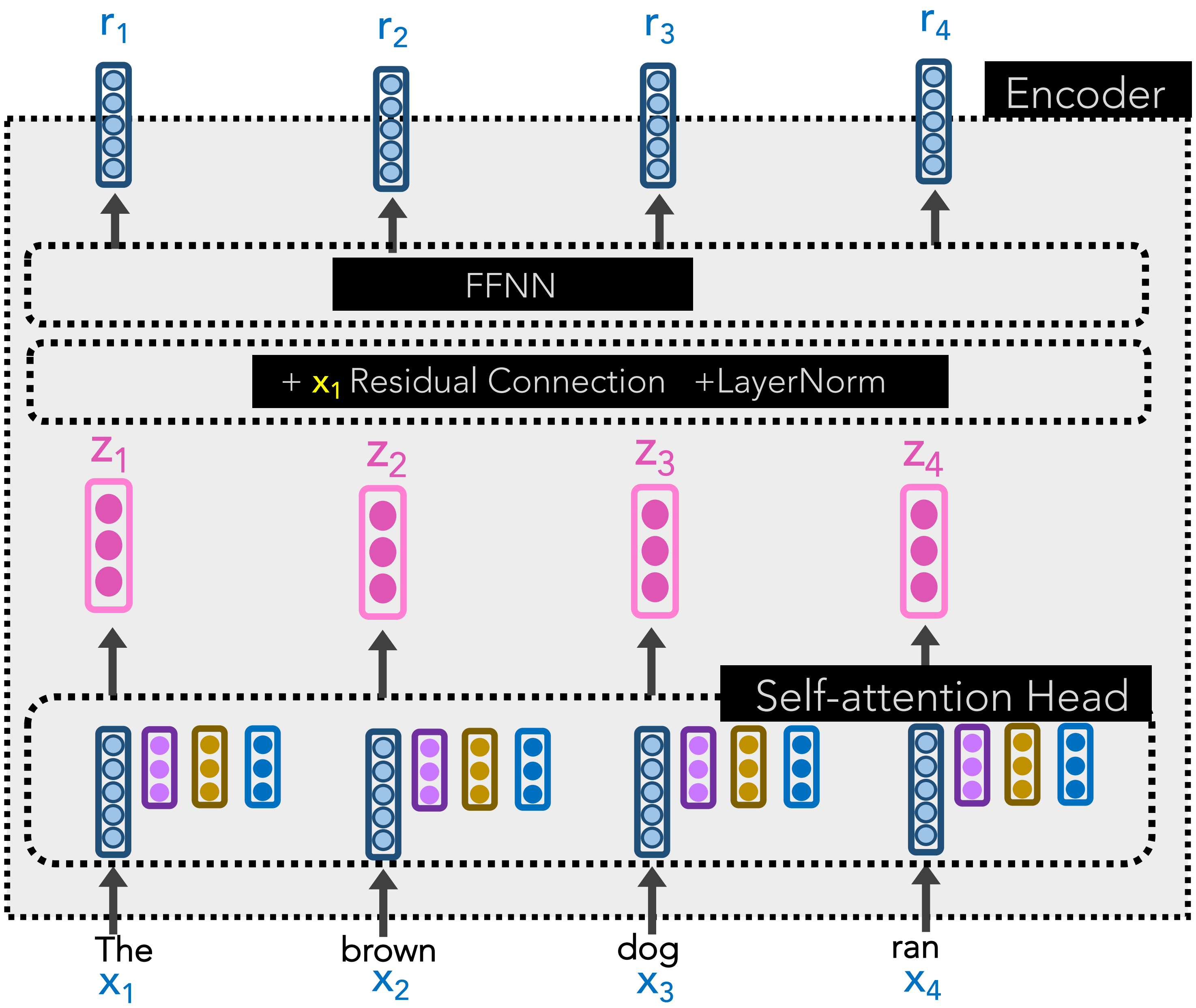
# Transformer Encoder




Yay! Our $r_i$ vectors are our new representations, and this entire process is called a **Transformer Encoder**

**Problem:** there is no concept of positionality. Words are weighted as if a "bag of words"

# Transformer Encoder




Yay! Our $r_i$ vectors are our new representations, and this entire process is called a **Transformer Encoder**

**Problem:** there is no concept of positionality. Words are weighted as if a "bag of words"

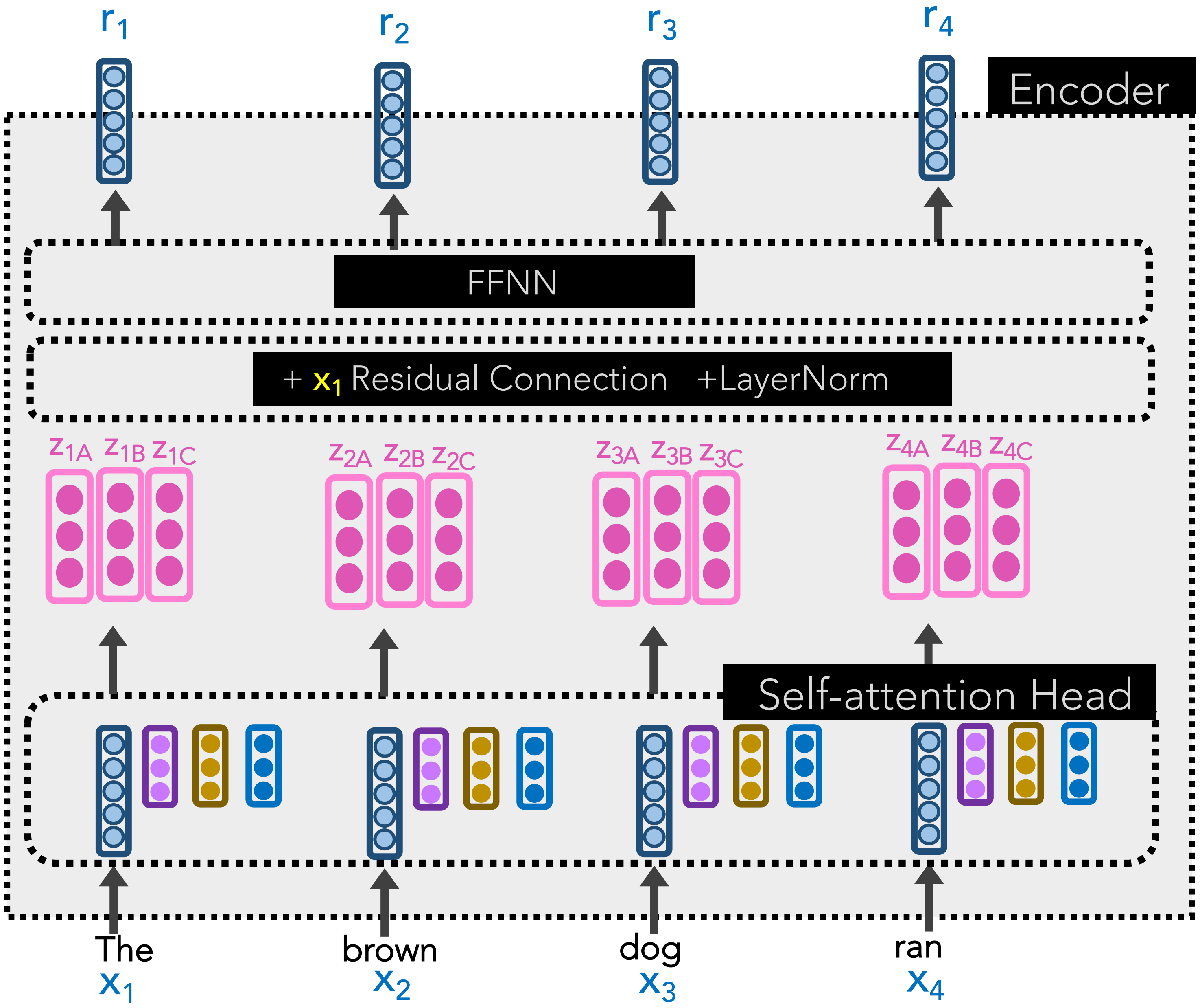
**Solution:** append each input word $x_i$ with a positional encoding: $\sin(i)\cos(i)$

A Self-Attention Head has just one set of query/key/value weight matrices $w_q, w_k, w_v$

Words can relate in many ways, so it's restrictive to rely on just one Self-Attention Head in the system.

Let's create Multi-headed Self-Attention

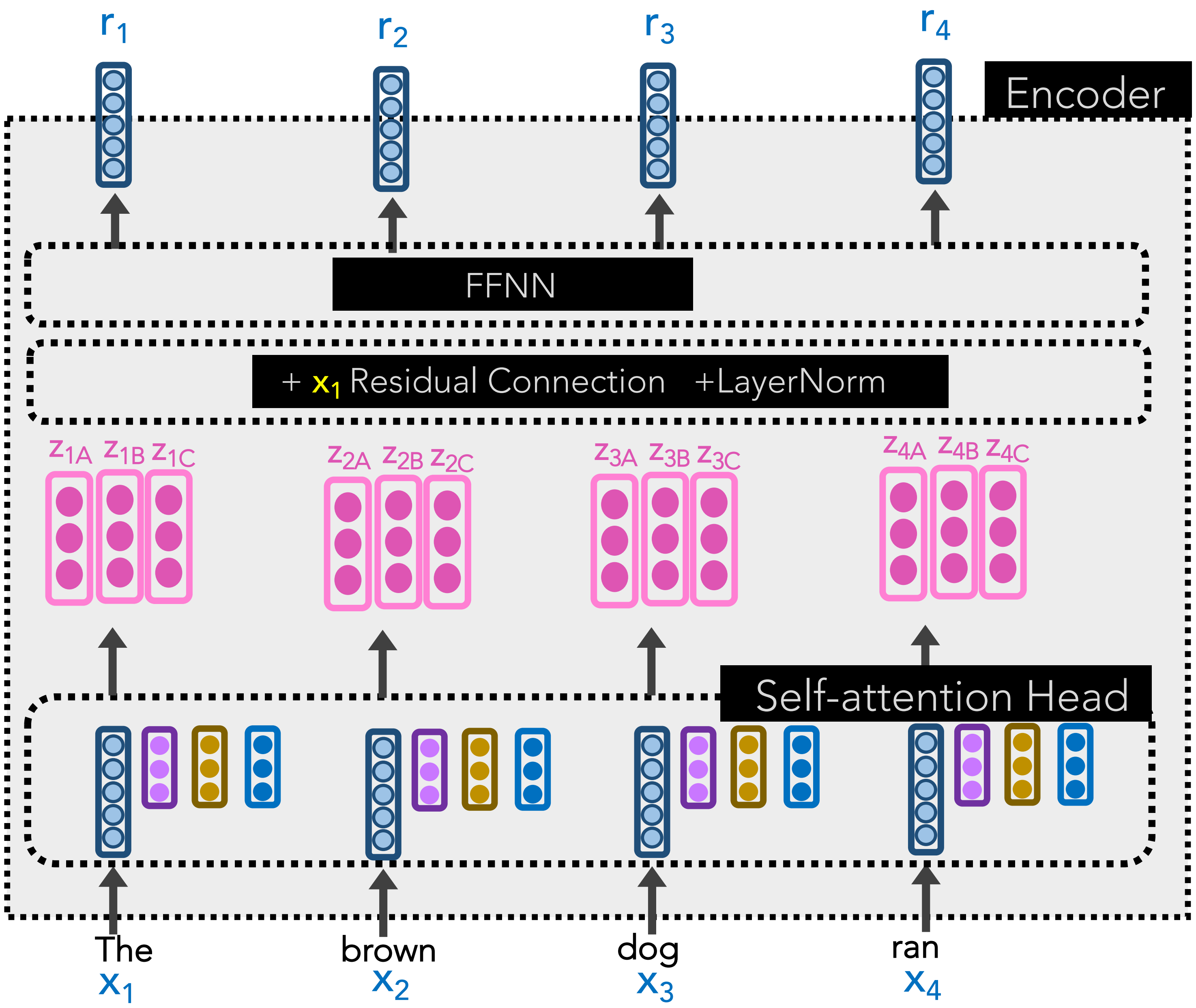
# Transformer Encoder




Each Self-Attention Head produces a $z_i$ vector.

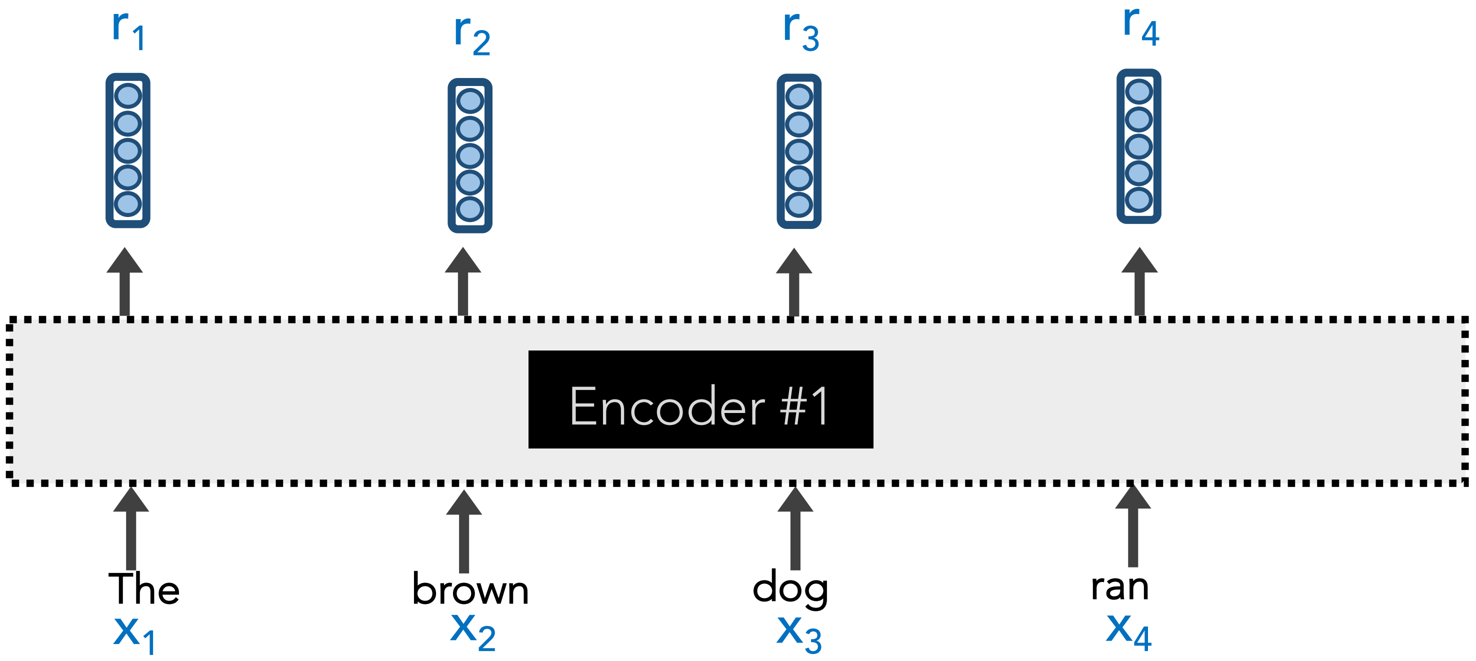
We can, in parallel, use multiple heads and concat the $z_i$'s.

# Transformer Encoder



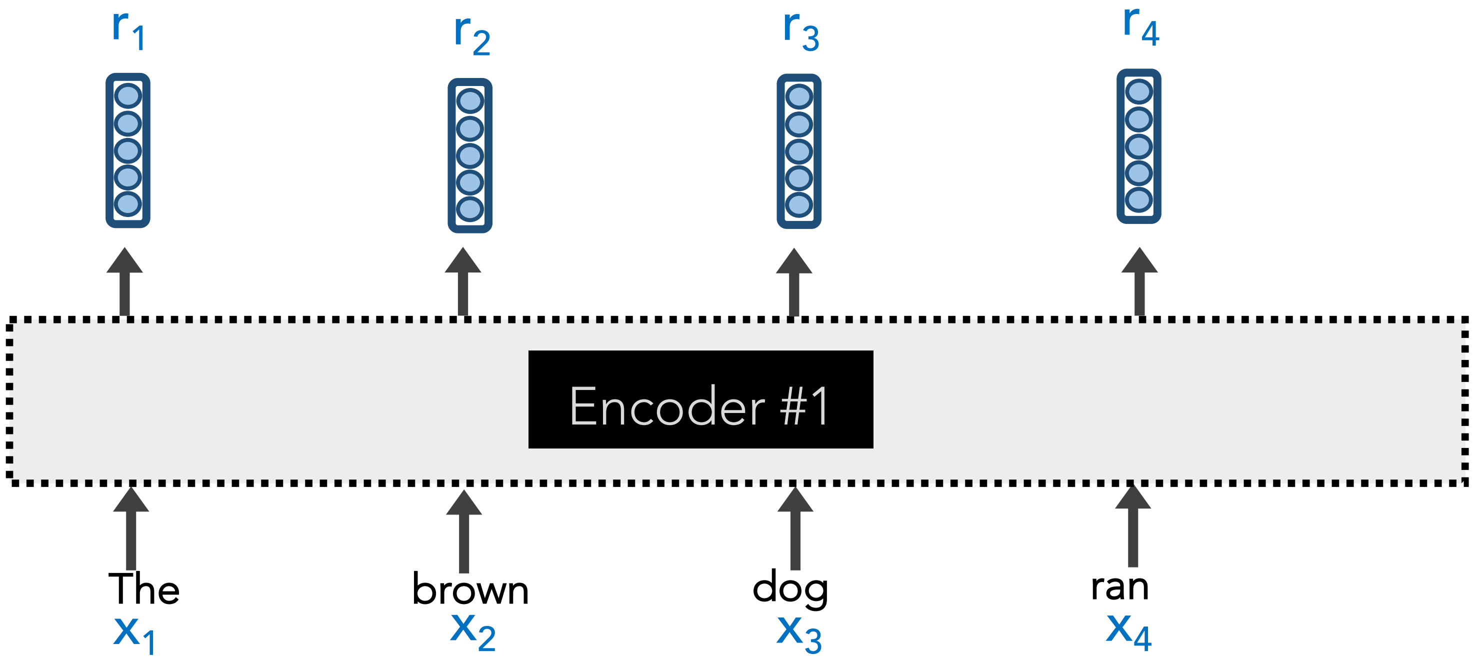

**To recap:** all of this looks fancy, but ultimately it's just producing a very good contextualized embedding $r_i$ of each word $x_i$

# Transformer Encoder




**To recap:** all of this looks fancy, but ultimately it's just producing a very good **contextualized embedding** $r_i$ of each word $x_i$

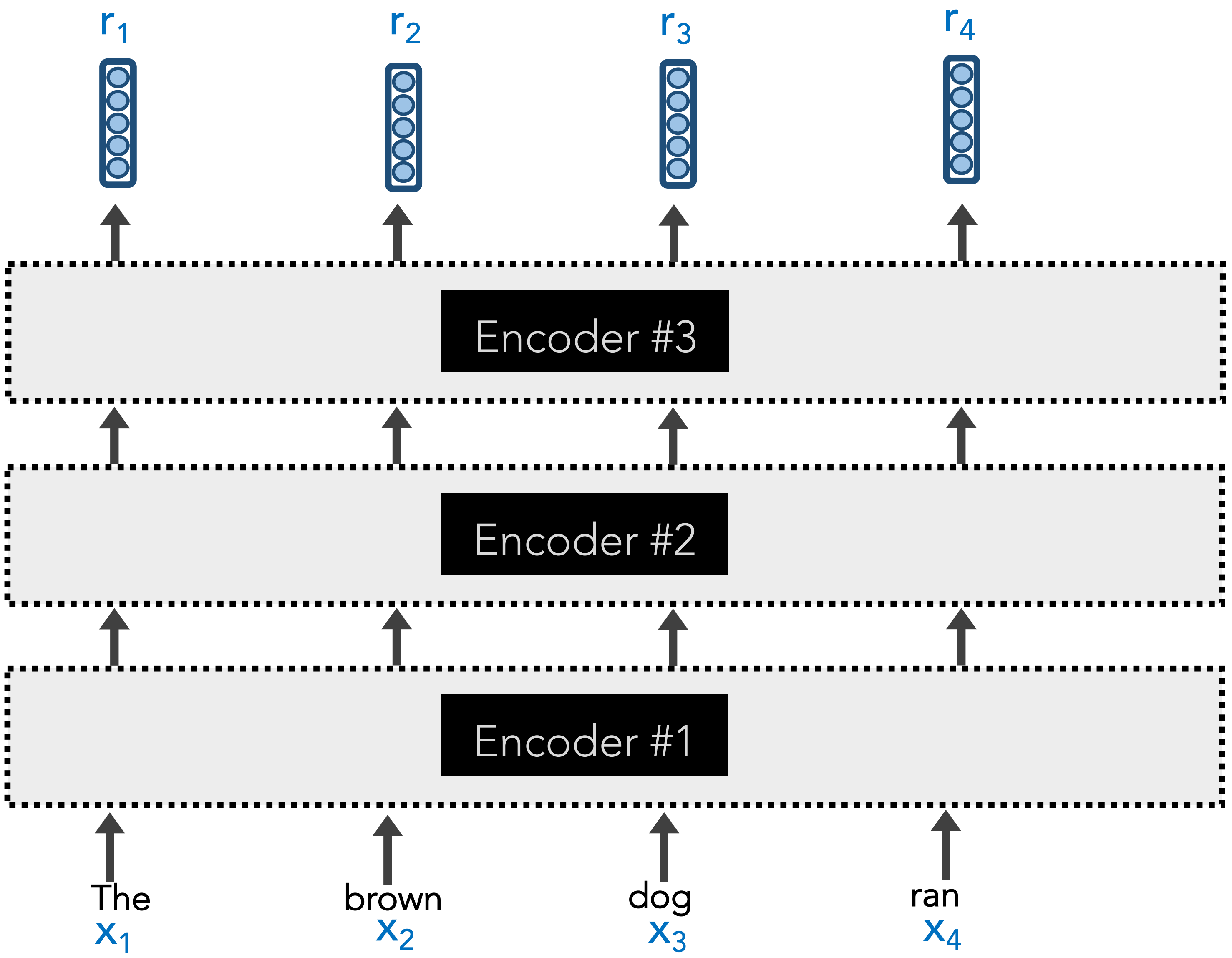
# Transformer Encoder




**To recap:** all of this looks fancy, but ultimately it's just producing a very good **contextualized embedding** $r_i$ of each word $x_i$

**Why stop with just 1 Transformer Encoder? We could stack several!**

# Transformer Encoder

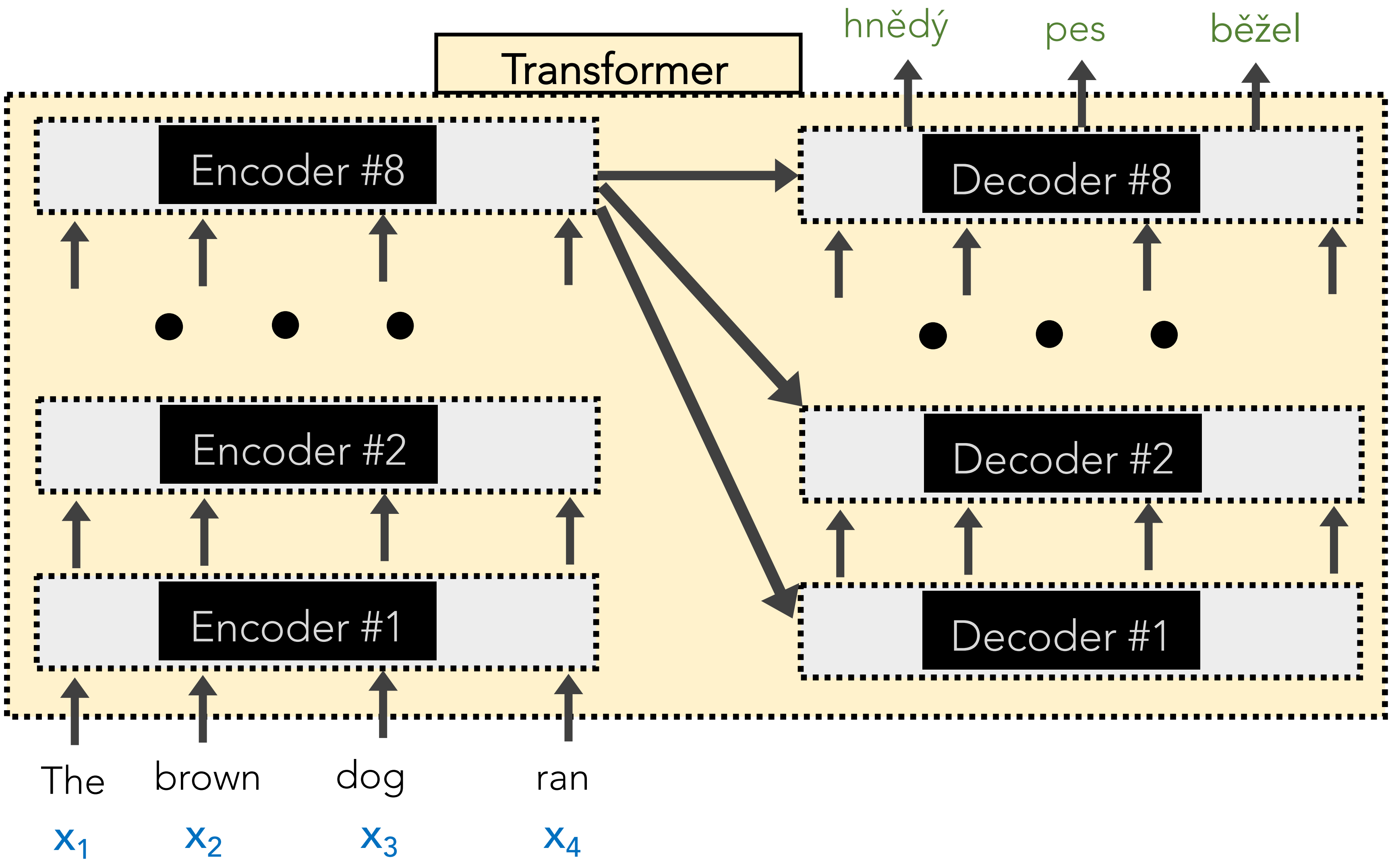



**To recap:** all of this looks fancy, but ultimately it's just producing a very good **contextualized embedding** $r_i$ of each word $x_i$

**Why stop with just 1 Transformer Encoder? We could stack several!**

The original Transformer model was intended for Machine Translation, so it had Decoders, too

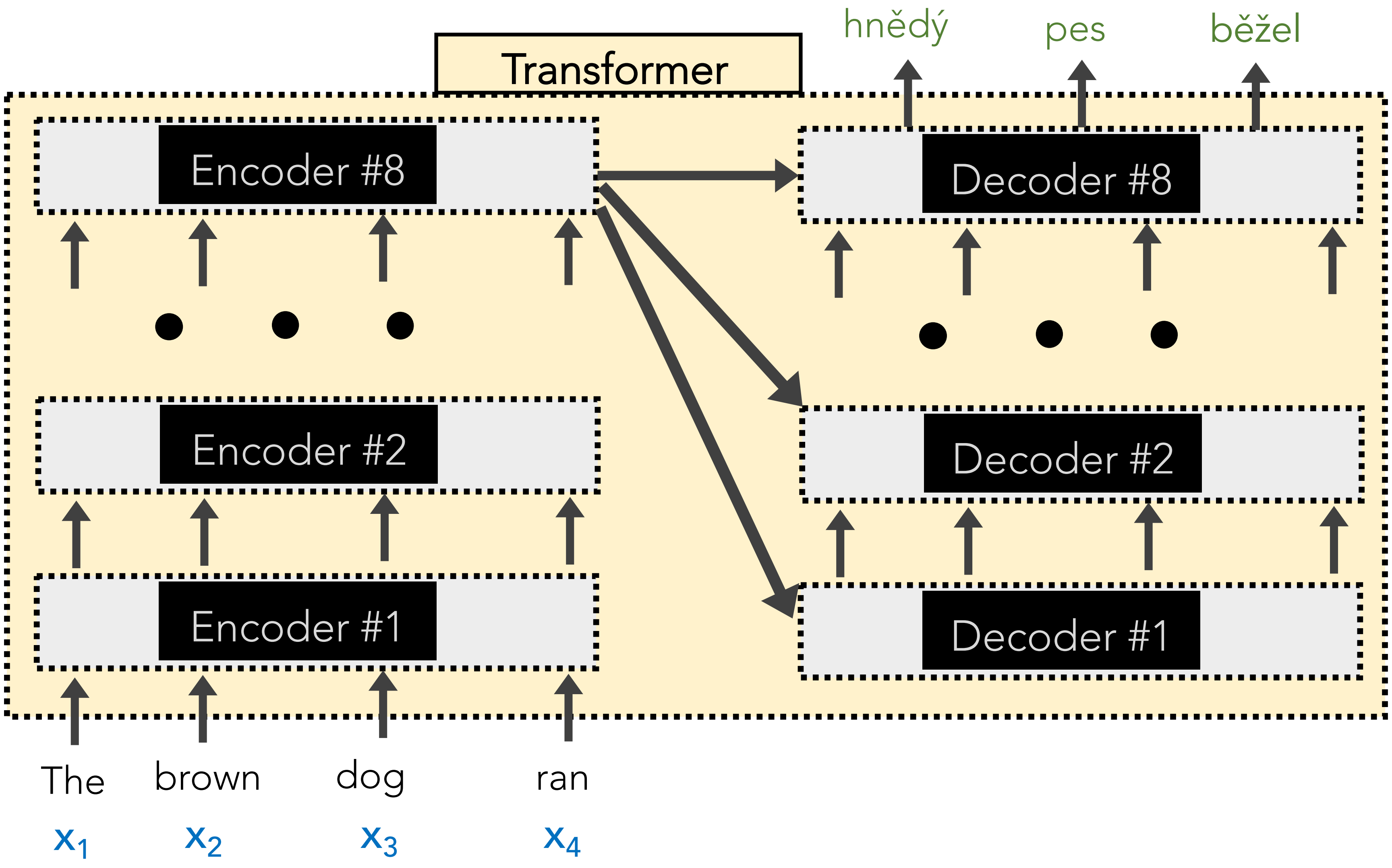
# Transformer Encoders and Decoders




Transformer Encoders produce contextualized embeddings of each word

Transformer Decoders generate new sequences of text

# Transformer Encoders and Decoders

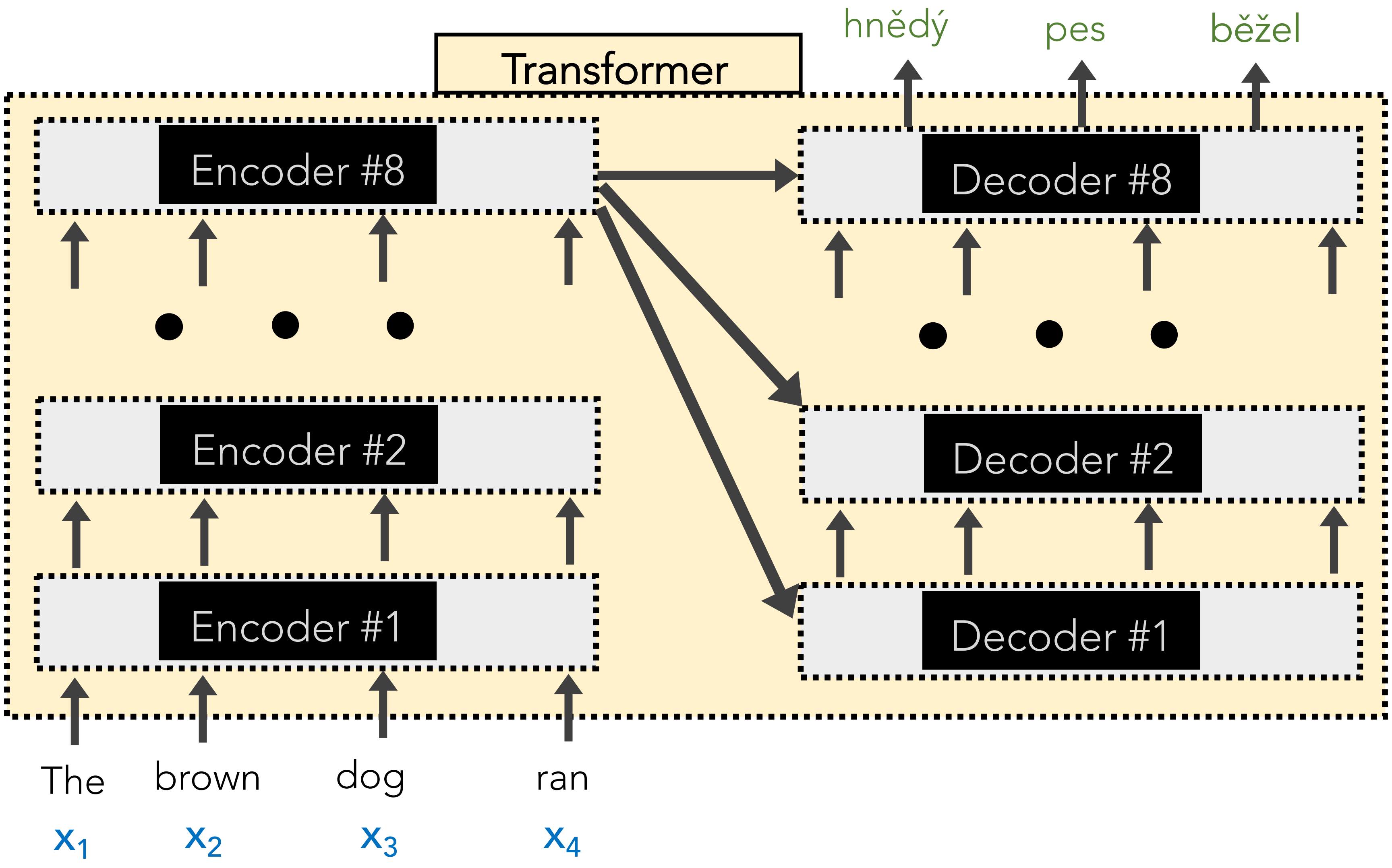



NOTE

Transformer Decoders are identical to the Encoders, except they have an additional Attention Head in between the Self-Attention and FFNN layers.

This additional Attention Head **focuses on parts of the encoder's representations.**

# Transformer Encoders and Decoders

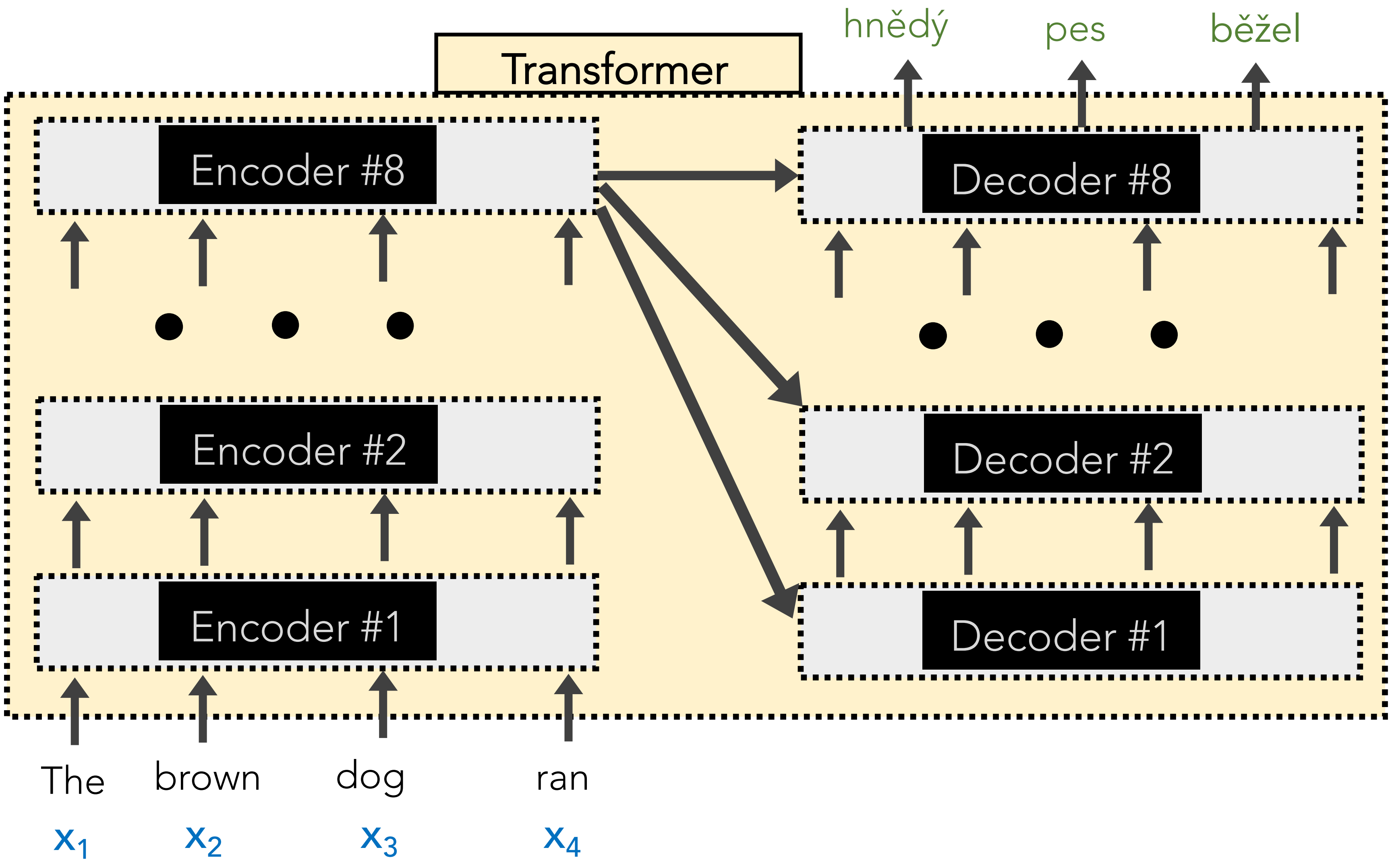



**NOTE**

The query vector for a Transformer Decoder's Attention Head (not Self-Attention Head) is from the output of the previous decoder layer.

However, the key and value vectors are from the Transformer Encoders' outputs.

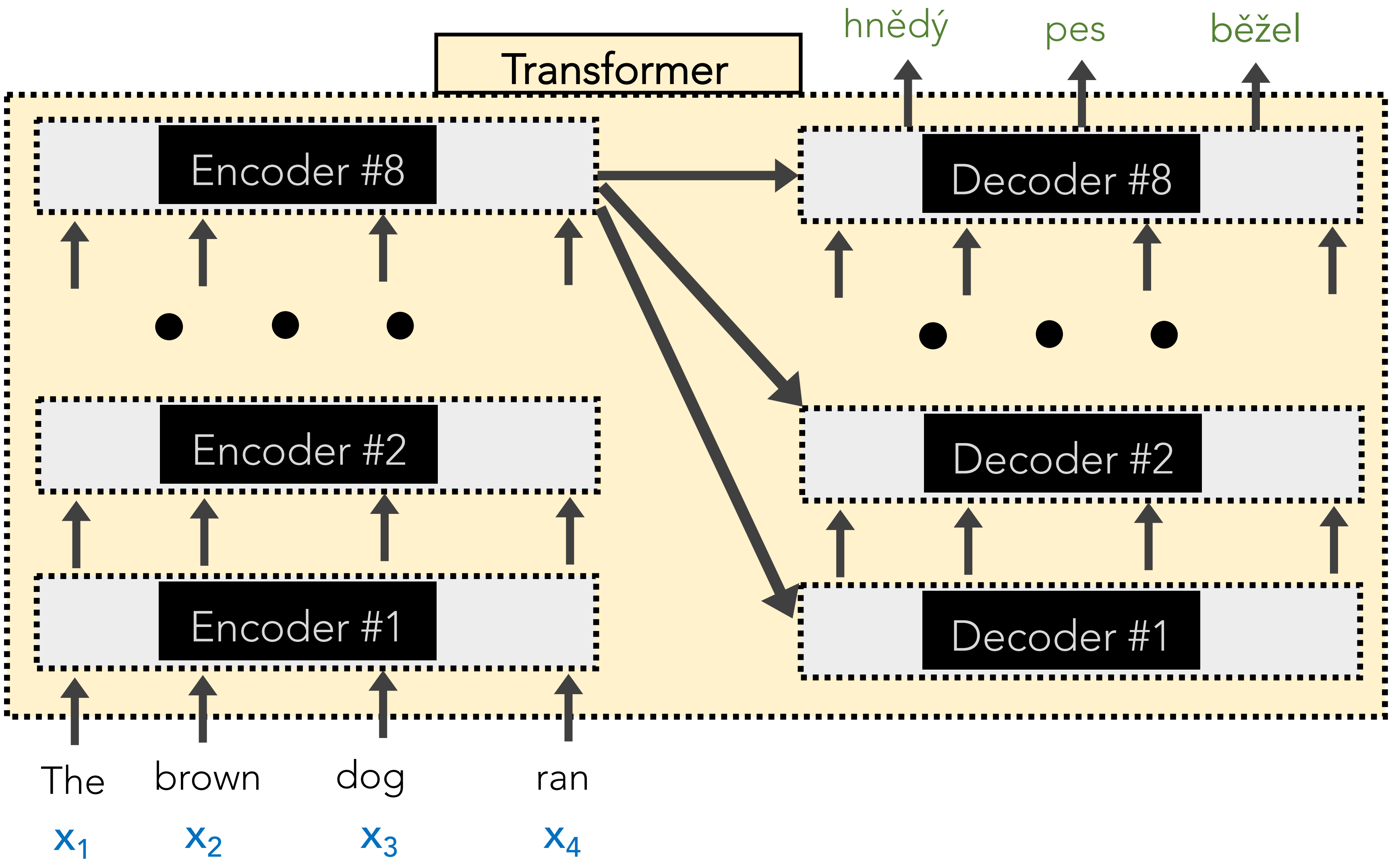
# Transformer Encoders and Decoders




NOTE

The query, key, and value vectors for a Transformer Decoder's **Self-Attention Head** (not Attention Head) are all from the output of the previous decoder layer.

# Transformer Encoders and Decoders




IMPORTANT

The Transformer Decoders have positional embeddings, too, just like the Encoders.

Critically, each position is only allowed to attend to the previous indices. This *masked* Attention preserves it as being an auto-regressive LM.

Loss Function: cross-entropy (predicting translated word)

Training Time: ~4 days on (8) GPUs

| Layer Type | Complexity per Layer | Sequential Operations | Maximum Path Length |
|---|---|---|---|
| Self-Attention | $O(n^2 \cdot d)$ | $O(1)$ | $O(1)$ |
| Recurrent | $O(n \cdot d^2)$ | $O(n)$ | $O(n)$ |
| Convolutional | $O(k \cdot n \cdot d^2)$ | $O(1)$ | $O(log_k(n))$ |
| Self-Attention (restricted) | $O(r \cdot n \cdot d)$ | $O(1)$ | $O(n/r)$ |

# Machine Translation results: state-of-the-art (at the time)

| Model | BLEU | | Training Cost (FLOPs) | |
|---|---|---|---|---|
| | EN-DE | EN-FR | EN-DE | EN-FR |
| ByteNet [18] | 23.75 | | | |
| Deep-Att + PosUnk [39] | | 39.2 | | $1.0 \cdot 10^{20}$ |
| GNMT + RL [38] | 24.6 | 39.92 | $2.3 \cdot 10^{19}$ | $1.4 \cdot 10^{20}$ |
| ConvS2S [9] | 25.16 | 40.46 | $9.6 \cdot 10^{18}$ | $1.5 \cdot 10^{20}$ |
| MoE [32] | 26.03 | 40.56 | $2.0 \cdot 10^{19}$ | $1.2 \cdot 10^{20}$ |
| Deep-Att + PosUnk Ensemble [39] | | 40.4 | | $8.0 \cdot 10^{20}$ |
| GNMT + RL Ensemble [38] | 26.30 | 41.16 | $1.8 \cdot 10^{20}$ | $1.1 \cdot 10^{21}$ |
| ConvS2S Ensemble [9] | 26.36 | **41.29** | $7.7 \cdot 10^{19}$ | $1.2 \cdot 10^{21}$ |
| Transformer (base model) | 27.3 | 38.1 | $\mathbf{3.3 \cdot 10^{18}}$ | |
| Transformer (big) | **28.4** | **41.8** | $2.3 \cdot 10^{19}$ | |

## Machine Translation results: state-of-the-art (at the time)

You can train to translate from Language A to Language B.

Then train it to translate from Language B. to Language C.

Then, without training, it can translate from Language A to Language C

- What if we don't want to decode/translate?
- Just want to perform a particular task (e.g., classification)
- Want even more robust, flexible, rich representation!
- Want positionality to play a more explicit role, while not being restricted to a particular form (e.g., CNNs)

# BERT

Bidirectional Encoder Representations from Transformers

# BERT

**B**idirectional **E**ncoder **R**epresentations from **T**ransformers

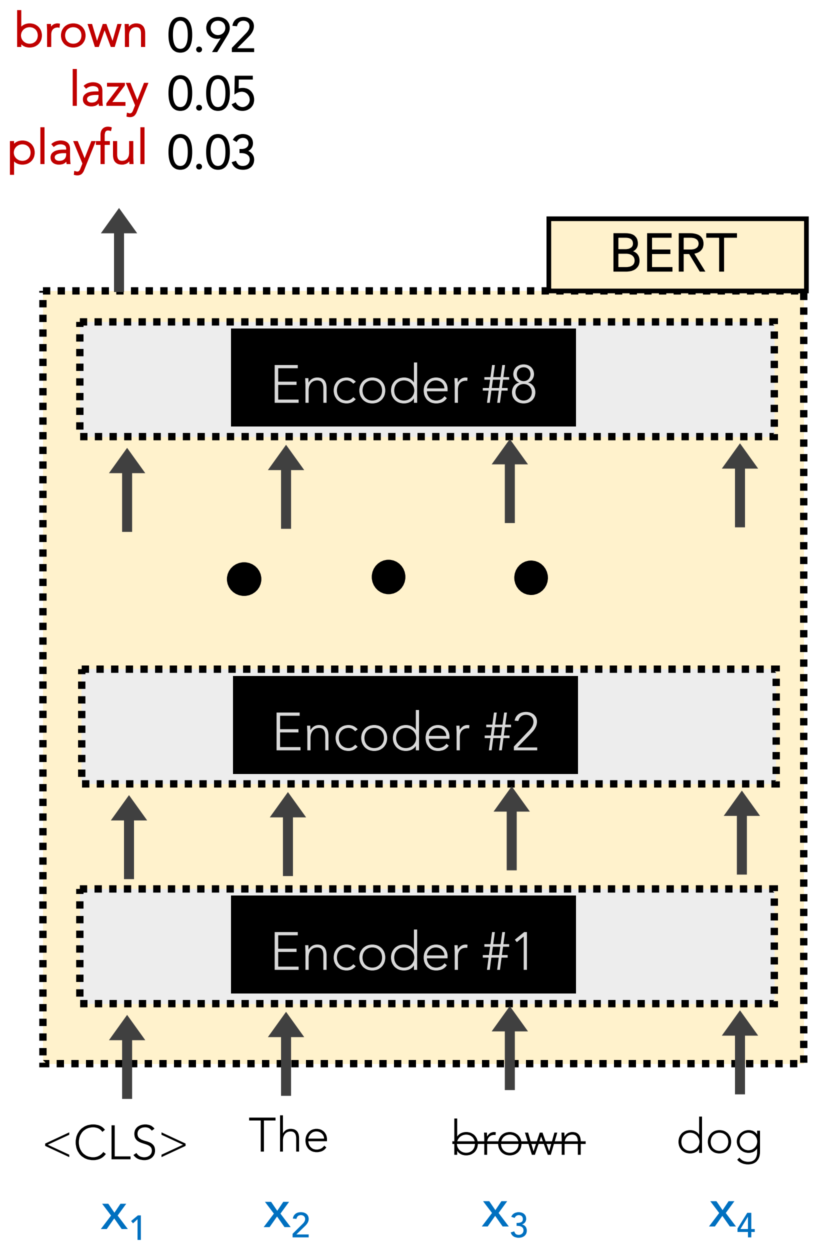
Let's only use Transformer *Encoders*, no Decoders

# BERT

Bidirectional Encoder Representations from Transformers

It's a language model that builds rich representations

# BERT



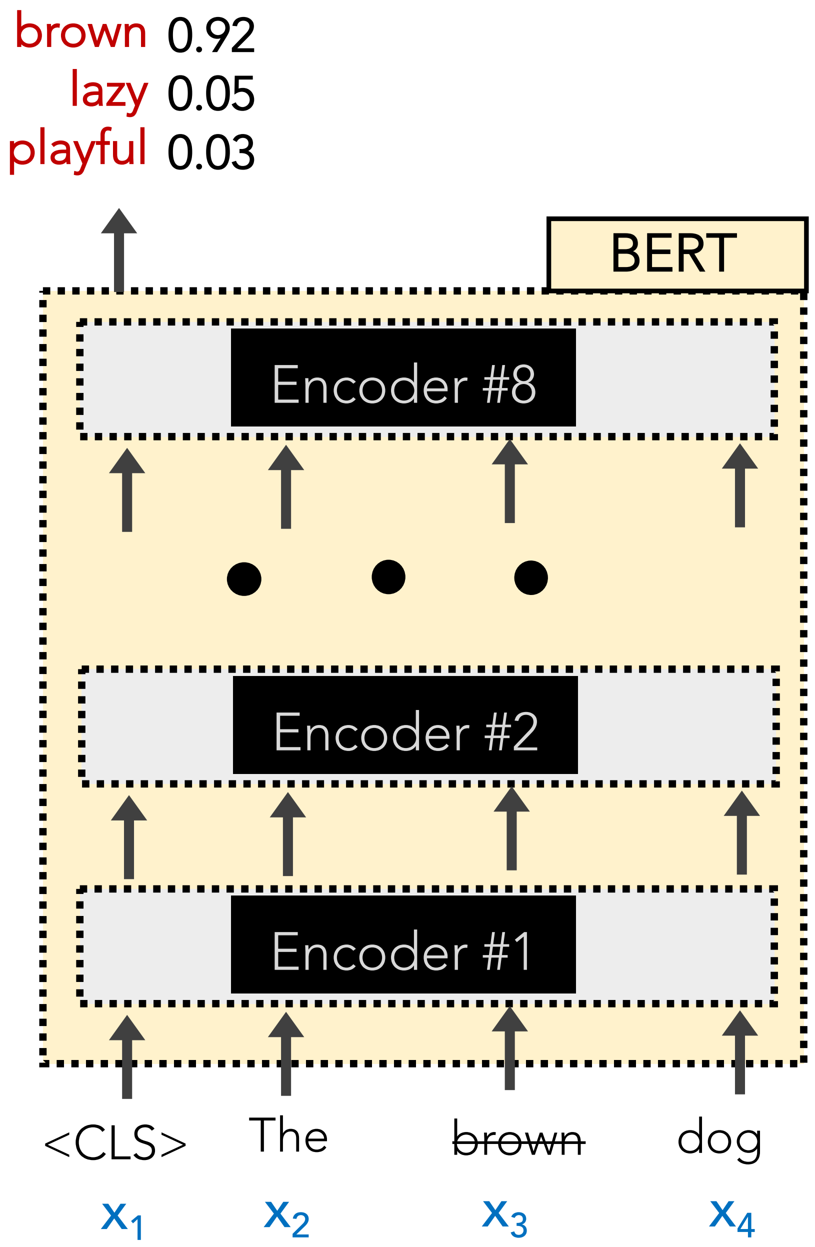

BERT has 2 training objectives:

1. Predict the **Masked word** (a la CBOW)

15% of all input words are randomly masked.

- 80% become [MASK]
- 10% become revert back
- 10% become are deliberately corrupted as wrong words

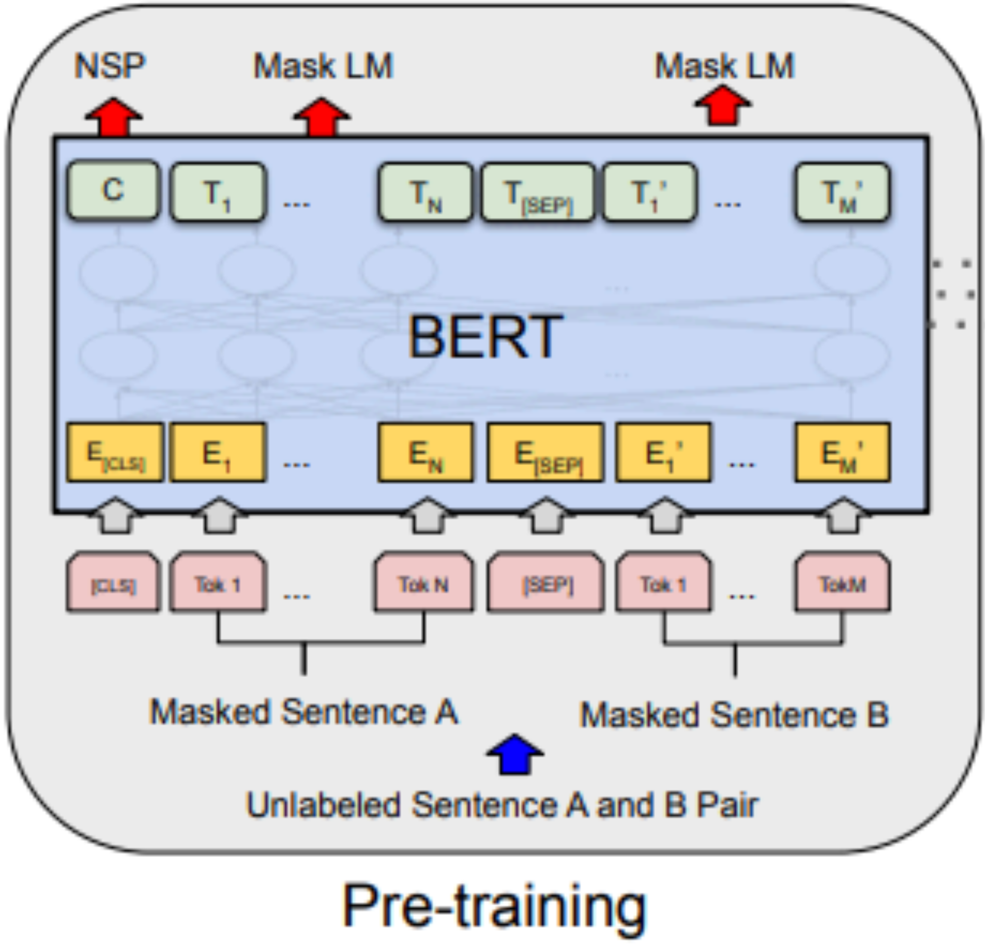
# BERT




BERT has 2 training objectives:

2. Two sentences are fed in at a time. Predict the if the second sentence of input truly follows the first one or not.

# BERT




Every two sentences are separated by a <SEP> token.

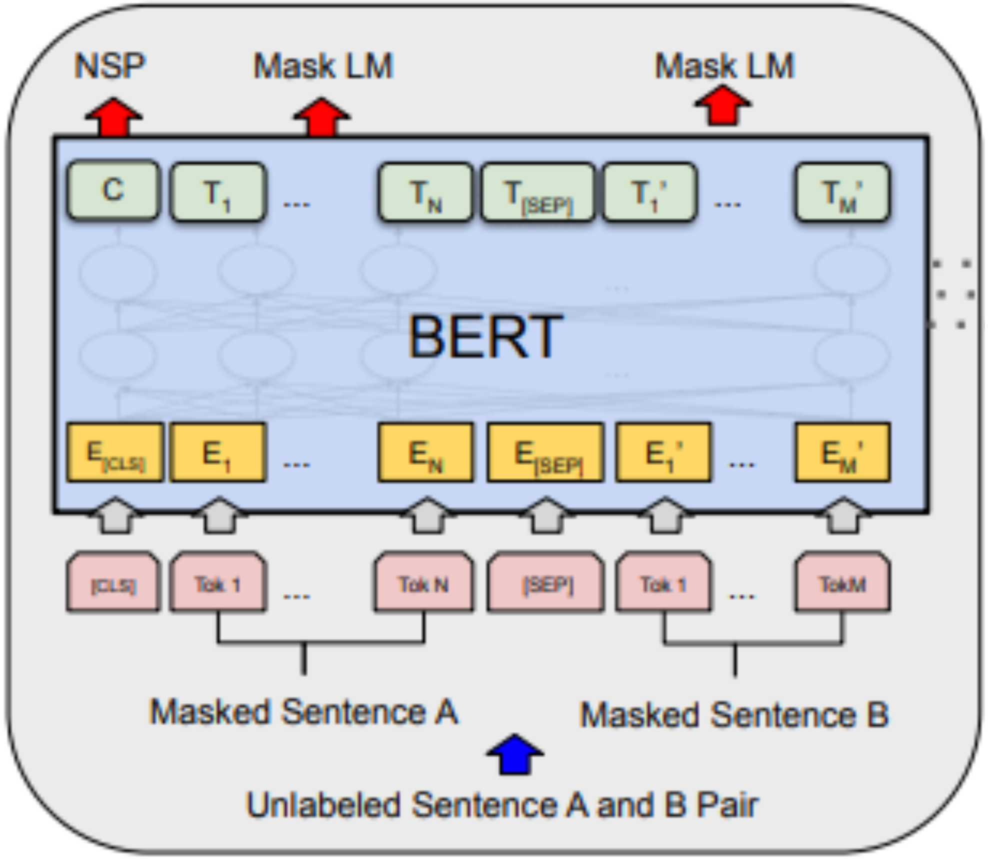
50% of the time, the 2nd sentence is a randomly selected sentence from the corpus.

50% of the time, it truly follows the first sentence in the corpus.



**Source**: original BERT paper: https://arxiv.org/pdf/1810.04805.pdf

# BERT




Pre-training

NOTE: BERT also embeds the inputs by their WordPiece embeddings.

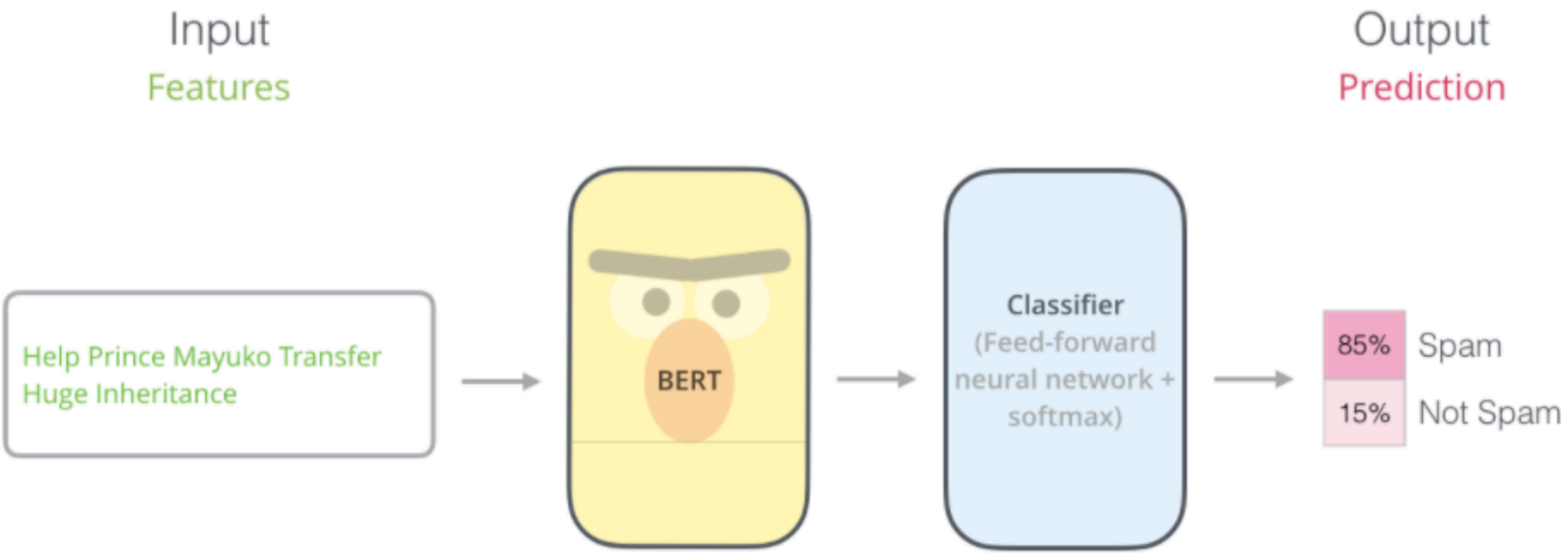
**WordPiece** is a <u>sub-word tokenization</u> learns to merge and use characters based on which pairs maximize the likelihood of the training data if added to the vocab.



Source: original BERT paper: https://arxiv.org/pdf/1810.04805.pdf

# BERT

Most of the time, people use BERT to fine-tune on a separate task

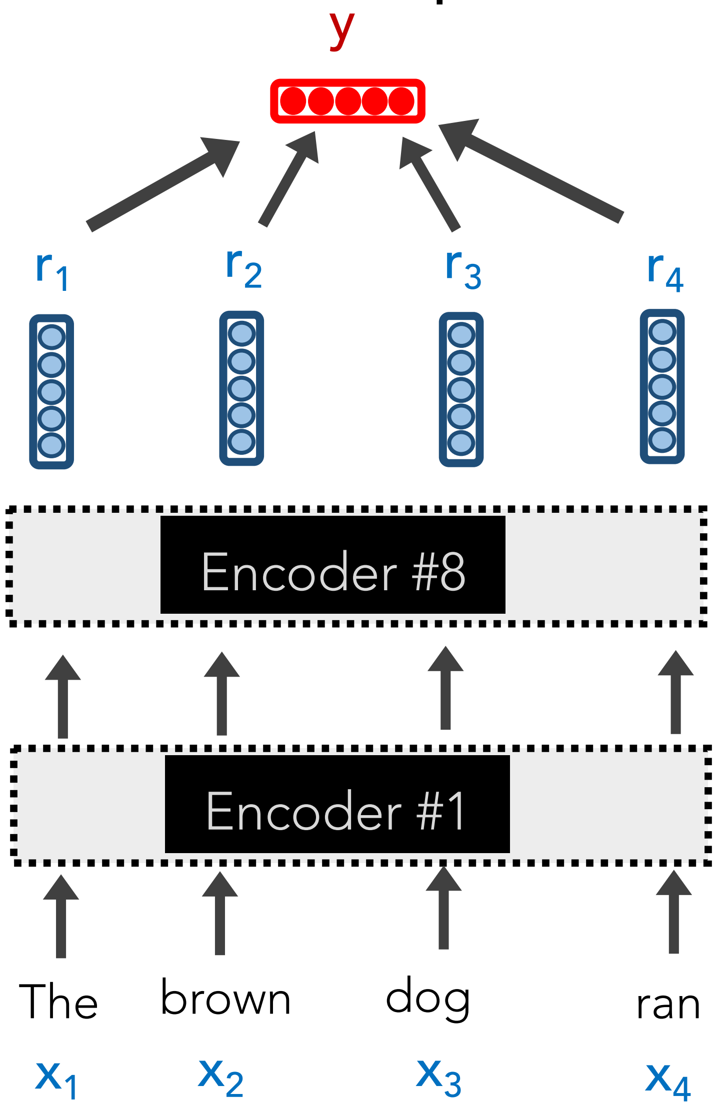





Picture: https://jalammar.github.io/illustrated-bert/

# BERT

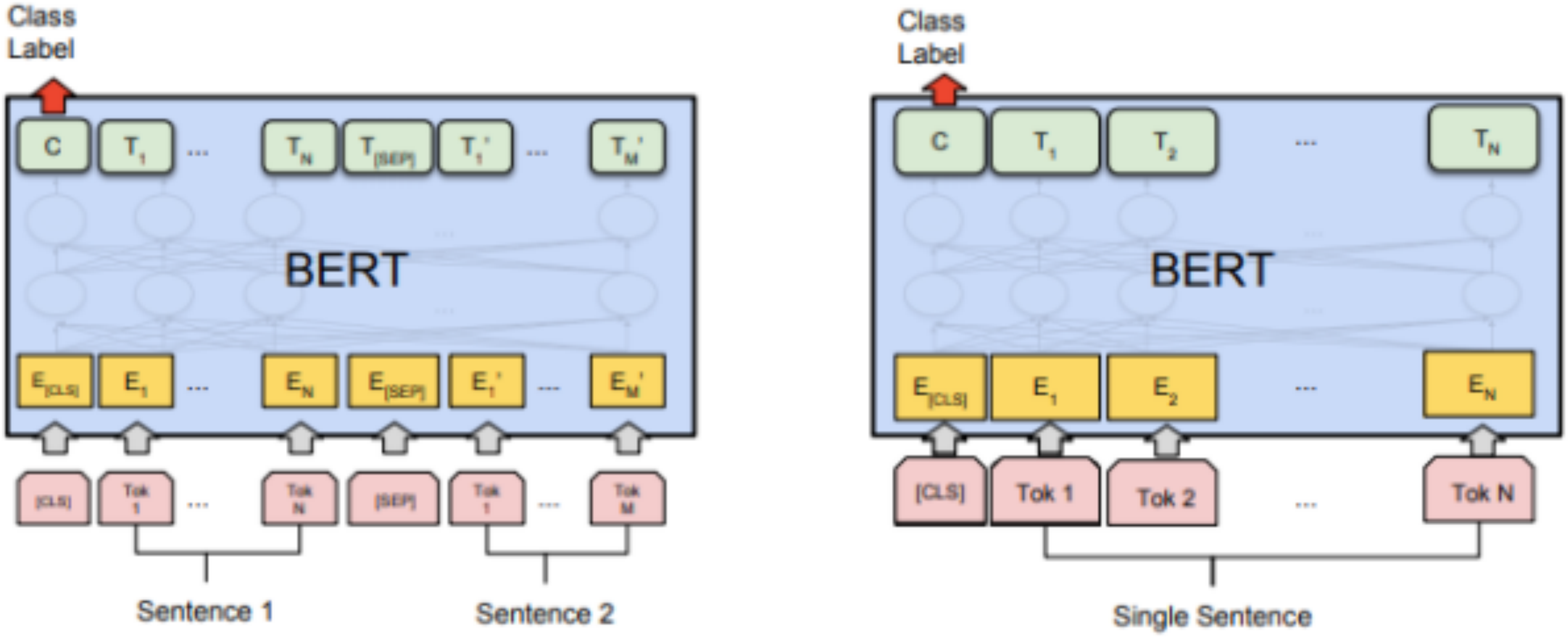
Most of the time, people use BERT to fine-tune on a separate task



# BERT

Most of the time, people use BERT

to fine-tune on a separate task




(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG

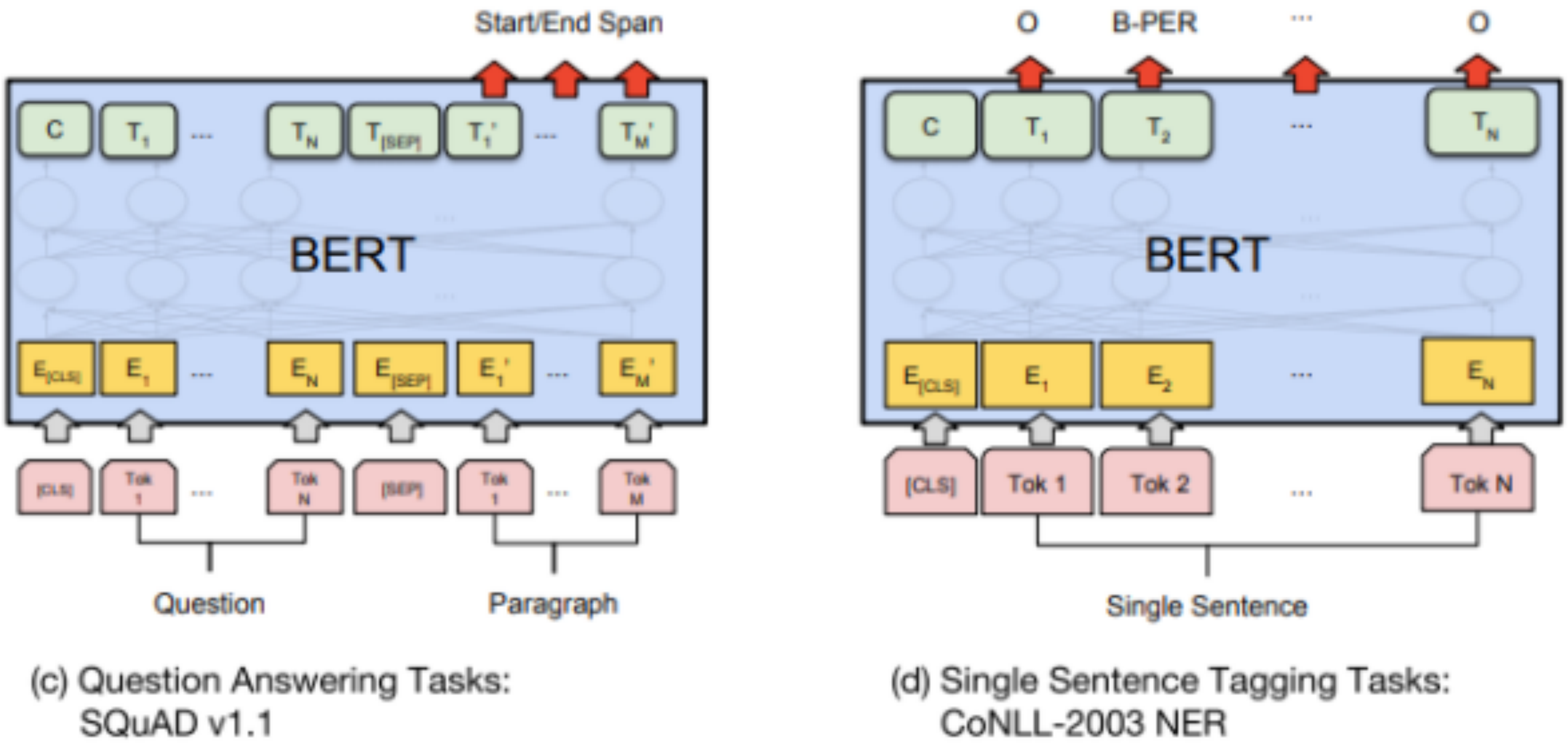
(b) Single Sentence Classification Tasks: SST-2, CoLA



**Picture**: https://jalammar.github.io/illustrated-bert/

# BERT

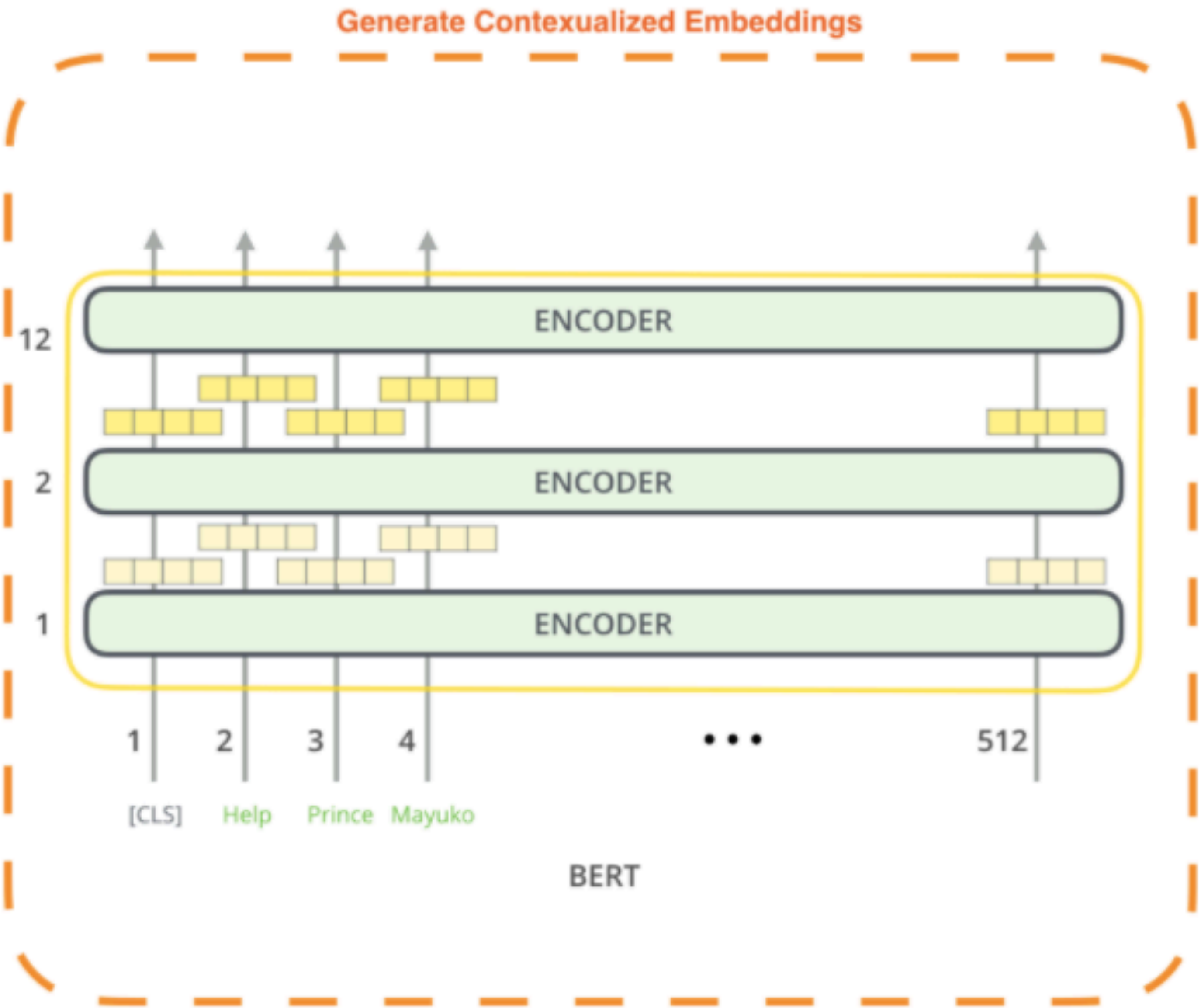
Most of the time, people use BERT to fine-tune on a separate task






Picture: https://jalammar.github.io/illustrated-bert/

# One could also extract the contextualized embeddings




The output of each encoder layer along each token's path can be used as a feature representing that token.

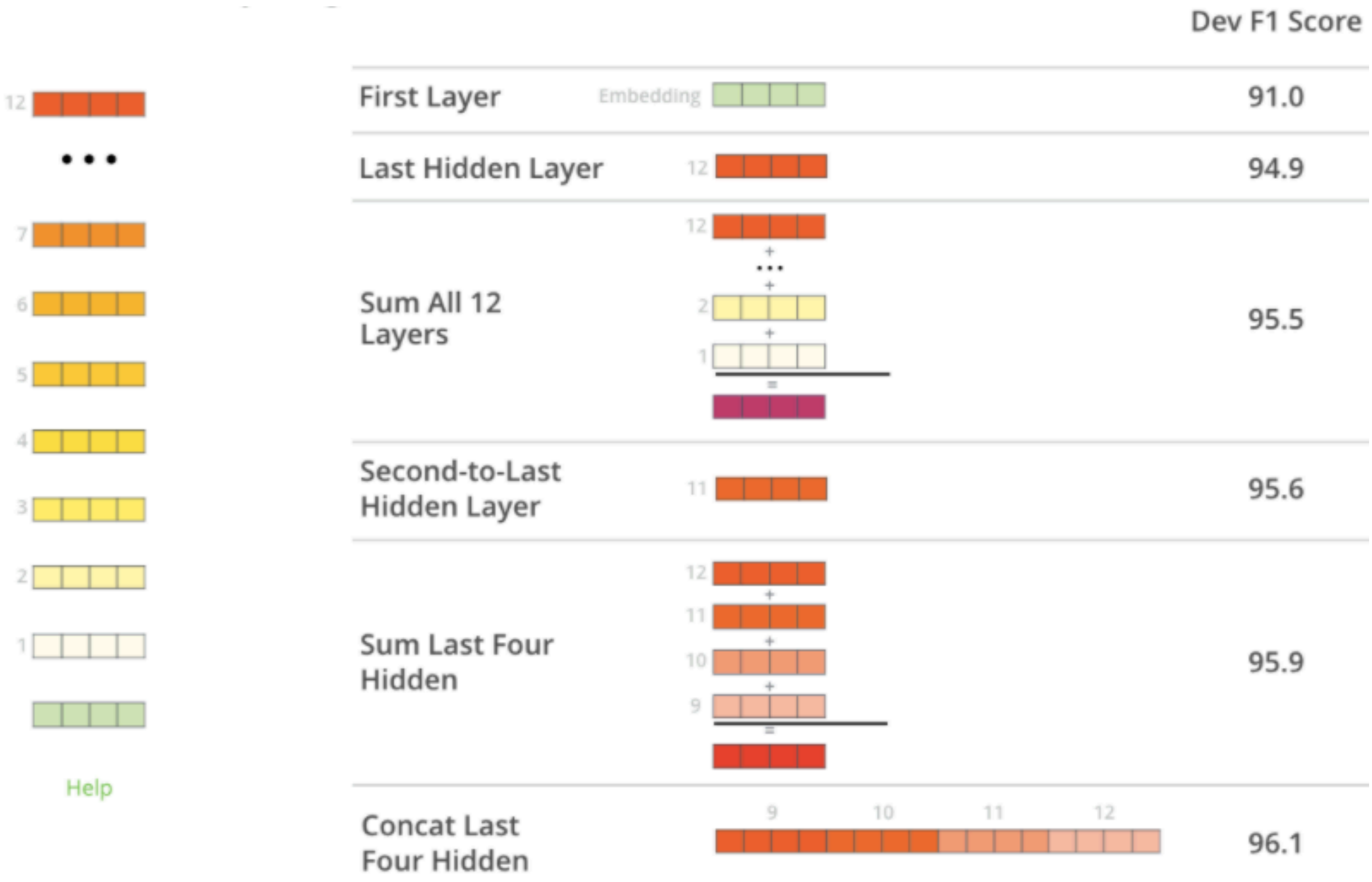

But which one should we use?



Picture: https://jalammar.github.io/illustrated-bert/

# BERT

## Later layers have the best contextualized embeddings




Picture: https://jalammar.github.io/illustrated-bert/

# BERT

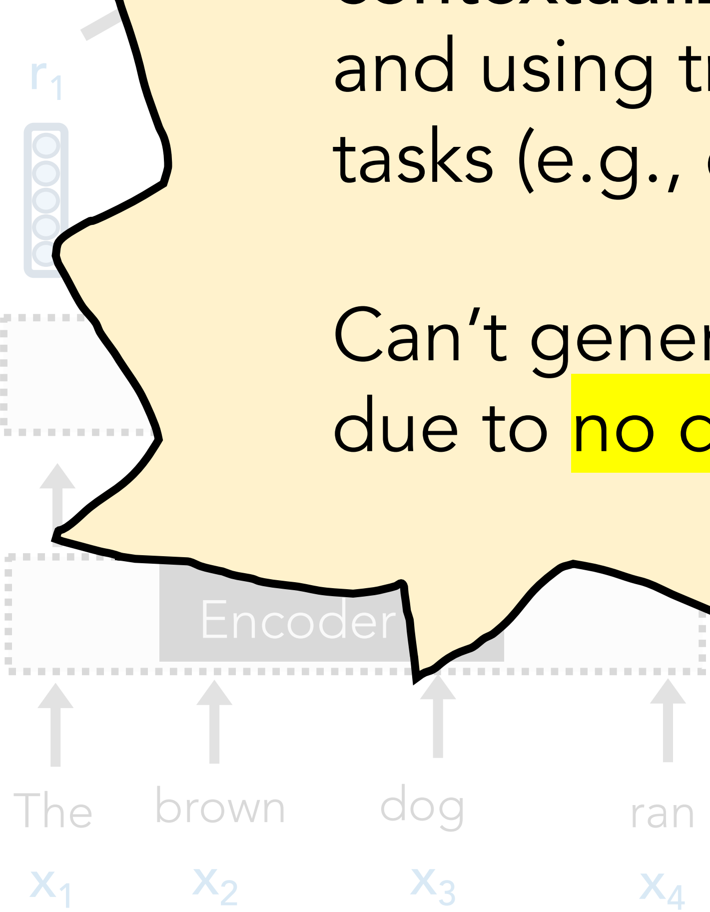
## BERT yields state-of-the-art (SOTA) results on many tasks

| System | MNLI-(m/mm) 392k | QQP 363k | QNLI 108k | SST-2 67k | CoLA 8.5k | STS-B 5.7k | MRPC 3.5k | RTE 2.5k | **Average** - |
|---|---|---|---|---|---|---|---|---|---|
| Pre-OpenAI SOTA | 80.6/80.1 | 66.1 | 82.3 | 93.2 | 35.0 | 81.0 | 86.0 | 61.7 | 74.0 |
| BiLSTM+ELMo+Attn | 76.4/76.1 | 64.8 | 79.8 | 90.4 | 36.0 | 73.3 | 84.9 | 56.8 | 71.0 |
| OpenAI GPT | 82.1/81.4 | 70.3 | 87.4 | 91.3 | 45.4 | 80.0 | 82.3 | 56.0 | 75.1 |
| BERT$_{\text{BASE}}$ | 84.6/83.4 | 71.2 | 90.5 | 93.5 | 52.1 | 85.8 | 88.9 | 66.4 | 79.6 |
| BERT$_{\text{LARGE}}$ | **86.7/85.9** | **72.1** | **92.7** | **94.9** | **60.5** | **86.5** | **89.3** | **70.1** | **82.1** |

Table 1: GLUE Test results, scored by the evaluation server (https://gluebenchmark.com/leaderboard).



Source: original BERT paper: https://arxiv.org/pdf/1810.04805.pdf




**Takeaway**

BERT is incredible for learning **contextualized embeddings** of words and using transfer learning for other tasks (e.g., classification).

Can't generate new sentences though, due to no decoders.

# ESM

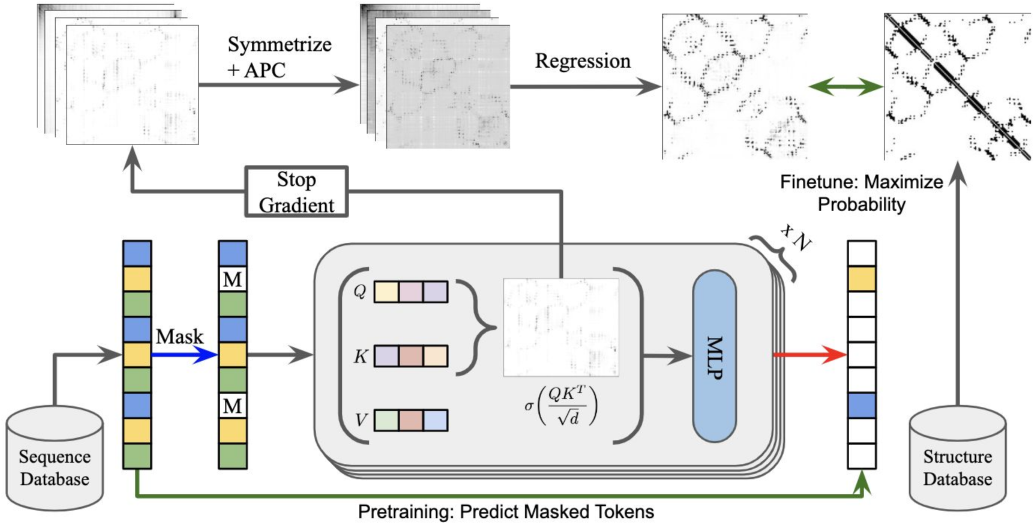
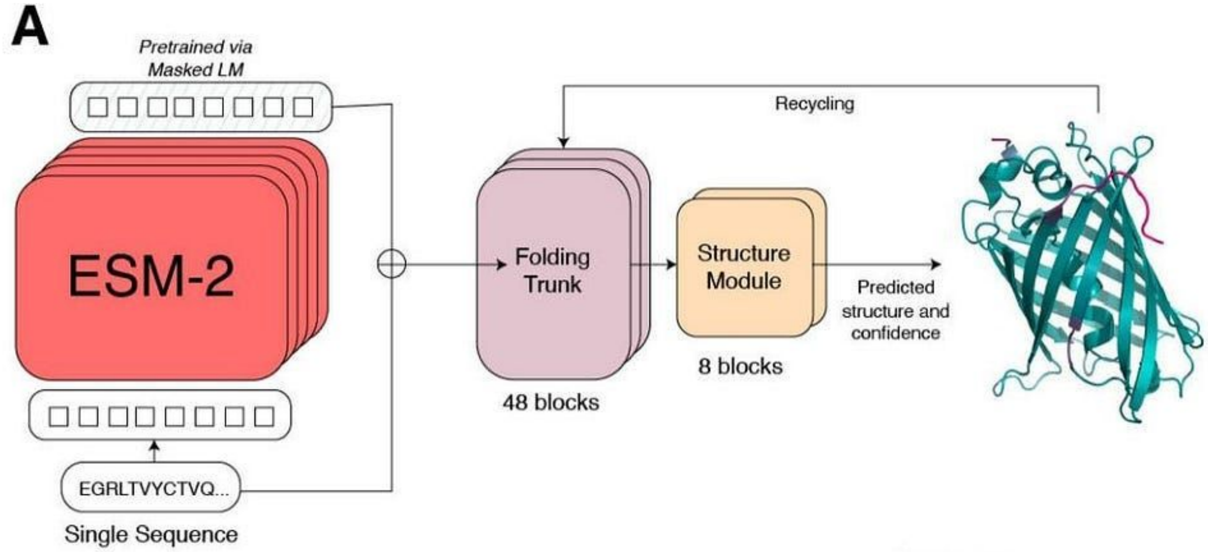


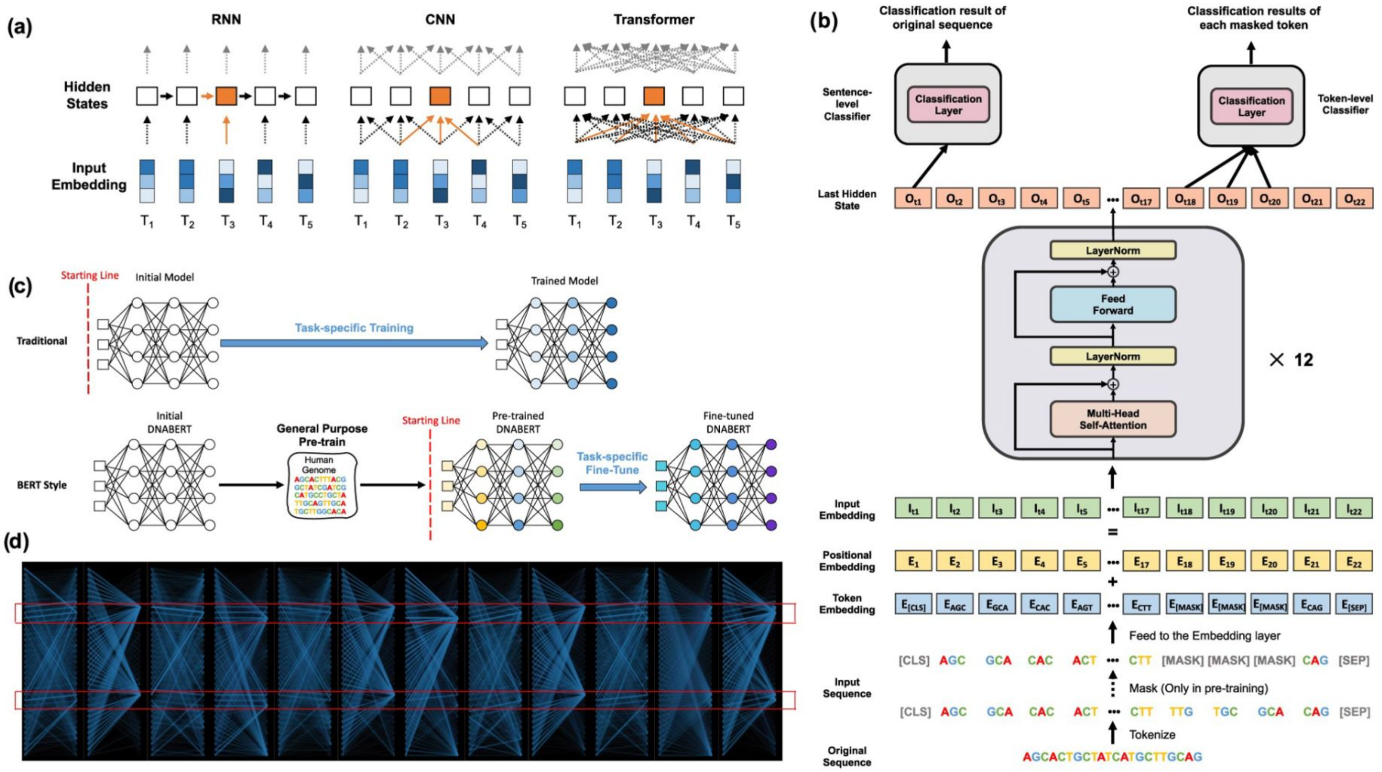

Figure 1: Contact prediction pipeline. The Transformer is first pretrained on sequences from a large database (Uniref50) via Masked Language Modeling. Once finished training, the attention maps are extracted, passed through symmetrization and average product correction, then into a regression. The regression is trained on a small number ($n \leq 20$) of proteins to determine which attention heads are informative. At test time, contact prediction from an input sequence can be done entirely on GPU in a single forward pass.

# ESM




https://doi.org/10.1101/2022.07.20.500902

# DNABert




https://doi.org/10.1093/bioinformatics/btab083

# Transformer

What if we want to generate a new output sequence?

GPT-2 model to the rescue!

**G**enerative **P**re-trained **T**ransformer 2

# GPT-2 (a Transformer variant)

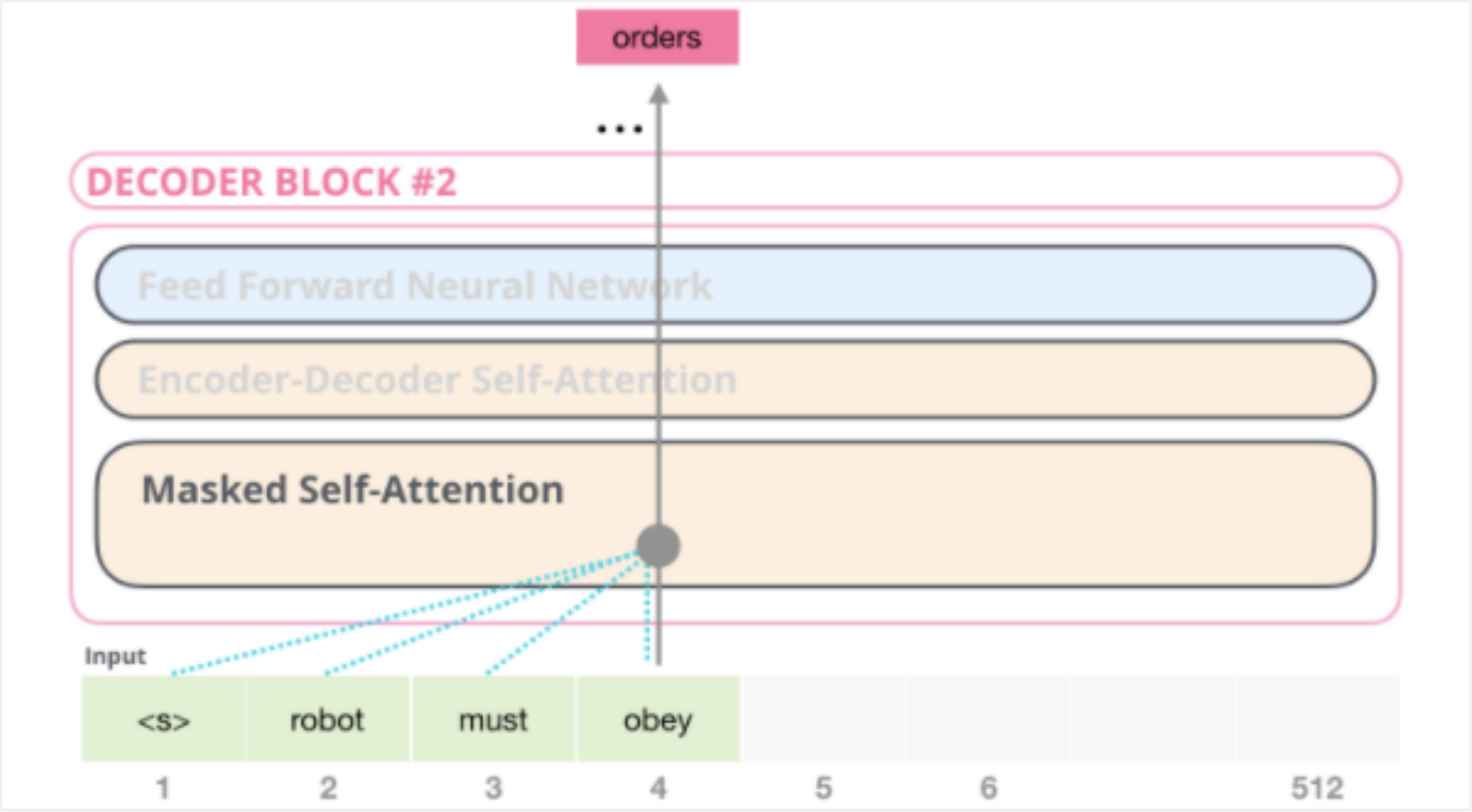
- GPT-2 uses only Transformer Decoders (no Encoders) to generate new sequences (from scratch or from a starting sequence)

There is only Self-Attention.

- As it processes each word/token, it cleverly **masks** the "future" words and conditions itself on the previous words

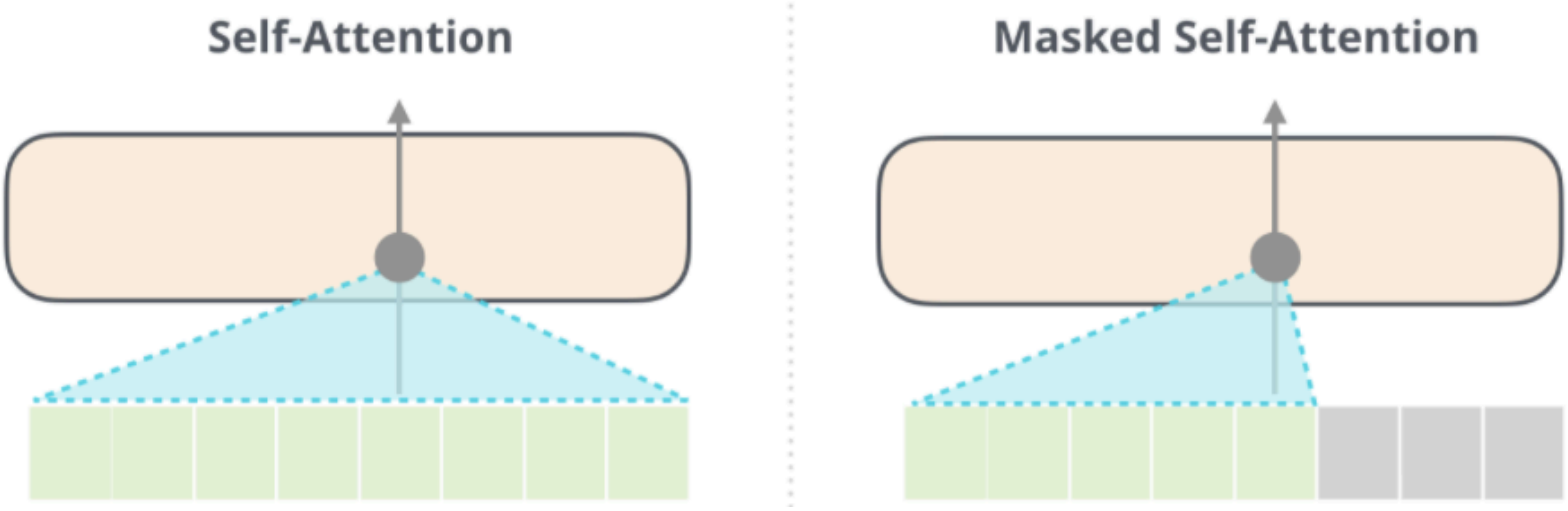
# GPT-2 (a Transformer variant)

As it processes each word/token, it cleverly masks the "future" words and conditions itself on the previous words



# GPT-2 (a Transformer variant)

As it processes each word/token, it cleverly **masks** the "future" words and conditions itself on the previous words



# GPT-2 (a Transformer variant)

- Technically, it doesn't use words as input but Byte Pair Encodings (sub-words), similar to BERT's WordPieces.
- Includes positional embeddings as part of the input, too.
- Easy to fine-tune on your own dataset (language)

# GPT-2 Results

## Easy to fine-tune on your own dataset (language)

SYSTEM PROMPT (HUMAN-WRITTEN)

A train carriage containing controlled nuclear materials was stolen in Cincinnati today. Its whereabouts are unknown.

MODEL COMPLETION (MACHINE-WRITTEN, FIRST TRY)

The incident occurred on the downtown train line, which runs from Covington and Ashland stations.

In an email to Ohio news outlets, the U.S. Department of Energy said it is working with the Federal Railroad Administration to find the thief.

“The theft of this nuclear material will have significant negative consequences on public and environmental health, our workforce and the economy of our nation,” said Tom Hicks, the U.S. Energy Secretary, in a statement. “Our top priority is to secure the theft and ensure it doesn’t happen again.”
The stolen material was taken from the University of Cincinnati’s Research Triangle Park nuclear research site, according to a news release from Department officials.
The Nuclear Regulatory Commission did not immediately release any information.

# GPT-2 Results

| Question | Generated Answer | Correct | Probability |
|---|---|---|---|
| Who wrote the book the origin of species? | Charles Darwin | ✓ | 83.4% |
| Who is the founder of the ubuntu project? | Mark Shuttleworth | ✓ | 82.0% |
| Who is the quarterback for the green bay packers? | Aaron Rodgers | ✓ | 81.1% |
| Panda is a national animal of which country? | China | ✓ | 76.8% |
| Who came up with the theory of relativity? | Albert Einstein | ✓ | 76.4% |
| When was the first star wars film released? | 1977 | ✓ | 71.4% |
| What is the most common blood type in sweden? | A | ✗ | 70.6% |
| Who is regarded as the founder of psychoanalysis? | Sigmund Freud | ✓ | 69.3% |
| Who took the first steps on the moon in 1969? | Neil Armstrong | ✓ | 66.8% |
| Who is the largest supermarket chain in the uk? | Tesco | ✓ | 65.3% |
| What is the meaning of shalom in english? | peace | ✓ | 64.0% |
| Who was the author of the art of war? | Sun Tzu | ✓ | 59.6% |
| Largest state in the us by land mass? | California | ✗ | 59.2% |
| Green algae is an example of which type of reproduction? | parthenogenesis | ✗ | 56.5% |
| Vikram samvat calender is official in which country? | India | ✓ | 55.6% |
| Who is mostly responsible for writing the declaration of independence? | Thomas Jefferson | ✓ | 53.3% |

# GPT-2 Results

Language Models are Unsupervised Multitask Learners

| | LAMBADA (PPL) | LAMBADA (ACC) | CBT-CN (ACC) | CBT-NE (ACC) | WikiText2 (PPL) | PTB (PPL) | enwik8 (BPB) | text8 (BPC) | WikiText103 (PPL) |
|---|---|---|---|---|---|---|---|---|---|
| SOTA | 99.8 | 59.23 | 85.7 | 82.3 | 39.14 | 46.54 | 0.99 | 1.08 | 18.3 |
| 117M | **35.13** | 45.99 | **87.65** | **83.4** | **29.41** | 65.85 | 1.16 | 1.17 | 37.50 |
| 345M | **15.60** | 55.48 | **92.35** | **87.1** | **22.76** | 47.33 | 1.01 | **1.06** | 26.37 |
| 762M | **10.87** | **60.12** | **93.45** | **88.0** | **19.93** | **40.31** | **0.97** | **1.02** | 22.05 |
| 1542M | **8.63** | **63.24** | **93.30** | **89.05** | **18.34** | **35.76** | **0.93** | **0.98** | **17.48** |

*Table 3.* Zero-shot results on many datasets. No training or fine-tuning was performed for any of these results. PTB and results are from (Gong et al., 2018). CBT results are from (Bajgar et al., 2016). LAMBADA accuracy result is from (Hoang and LAMBADA perplexity result is from (Grave et al., 2016). Other results are from (Dai et al., 2019).

## GPT-2

- GPT-2 u…

  new…

- As it p…

  wor…

- Can g…

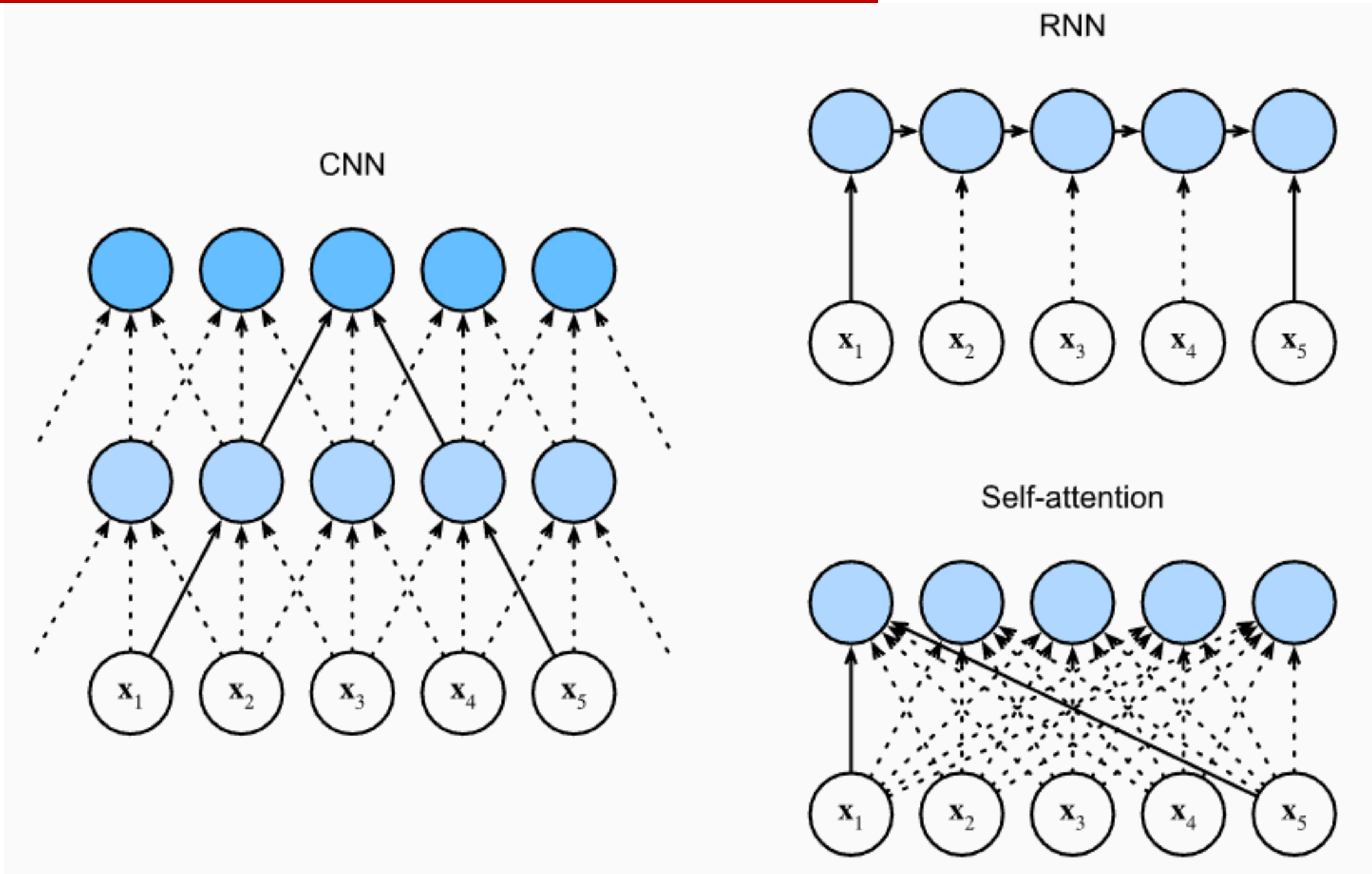
- Easy…

- GPT-3 is an eve… bigger version of GP…, but isn't open-source

**Takeaway:**
GPT-2 is astounding for generating realistic-looking new text

Can fine-tune toward other tasks, too.

# Transformer vs CNN vs RNN




**Source**: https://d2l.ai/chapter_attention-mechanisms/self-attention-and-positional-encoding.html#subsec-cnn-rnn-self-attention

# BERT (a Transformer variant)

BERT is trained on a lot of text data:

- BooksCorpus (800M words)
- English Wikipedia (2.5B words)

Yay, for transfer learning!

BERT-Base model has 12 transformer blocks, 12 attention heads,

110M parameters!

BERT-Large model has 24 transformer blocks, 16 attention heads,

340M parameters!

# GPT-2 (a Transformer variant)

GPT-2 is:

- trained on 40GB of text data (8M webpages)!
- 1.5B parameters

GPT-3 is an even bigger version (175B parameters) of GPT-2, but isn't open-source

Yay, for transfer learning!

# Concerns

There are several issues to be aware of:

- It is very costly to train these large models. The companies who develop these models easily spend an entire month training one model, which uses incredible amounts of electricity.
- BERT alone is estimated to cost over $1M for their final models

- $2.5k - $50k (110 million parameter model)
- $10k - $200k (340 million parameter model)
- $80k - $1.6m (1.5 billion parameter model)

**Source**: https://arxiv.org/pdf/2004.08900.pdf