Pontificia Universidad Católica de Chile
IA Lab

Transformers

Carlos Aspillaga & Felipe del Río

Computer Science Department, PUC

Intro

Motivation

Point the tallest building.

Motivation

To correctly solve the previous tasks:

1. Which elements of the image are buildings.
2. Extract a measure of size.
3. Pointwise compare each building size's between each other.

But when the underlying structure is composed of complex and sparse relationships, deep learning

- Doesn't work well without large amounts of data.
- Usually employ shortcuts.
- Don't generalize correctly.

Motivation

Point the tallest building.

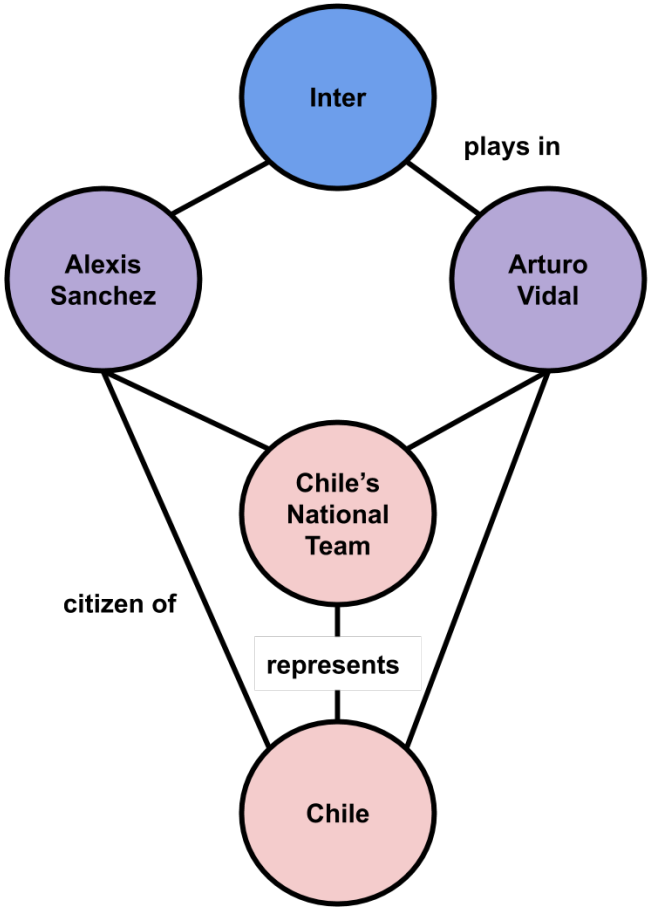
Table of Contents

Relationship Learning

Relational Reasoning

Relational reasoning is a central component of intelligent behavior.

- **Entities.** *Alexis*, *Inter*, etc.
- **Relations.** *citizen of*, *plays in*, etc.
- **Rules.**
  x *plays in Chile's NT* $\rightarrow$ x *citizen of Chile*, etc.



Relational Reasoning

- Symbolic approaches (GOFAI) are intrinsically relational. We would like o replicate this.
- We can reason about entities using powerful methods such as algebra, deduction, arithmetic, etc.
  e.g.

$$x^2 + 3x = y^7$$

- The building example can be thought as

$$\underset{b_i \in Buildings}{\arg\max} \quad size(b_i)$$

Transformer

Motivation

When modeling sequence data, RNNs are a natural choice to model a particular task.

But...

- Doesn't have a explicit mechanism to model hierarchy.
- hidden state $\rightarrow$ bottle neck during training
- Requires various steps to relate far away tokens.

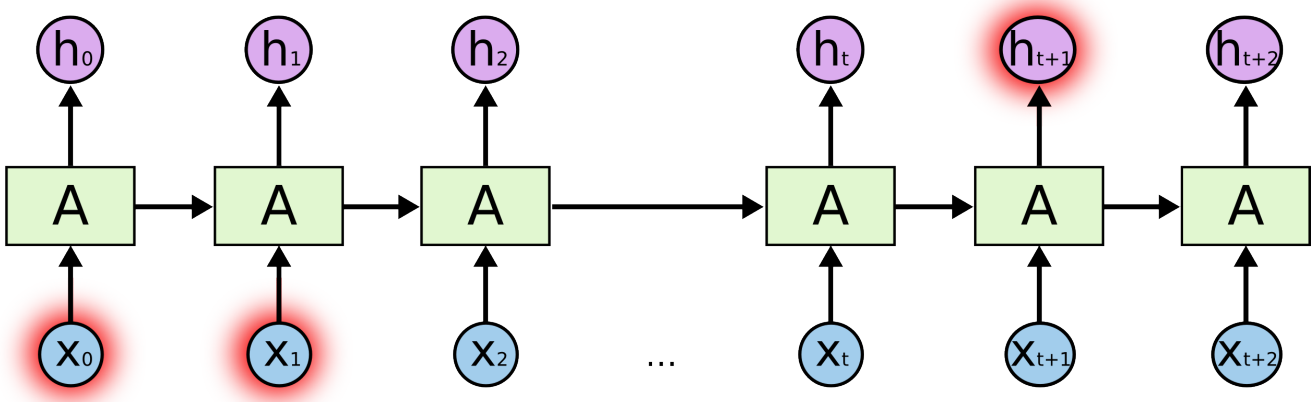


Image from https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com

Noam Shazeer* Google Brain noam@google.com

Niki Parmar* Google Research nikip@google.com

Jakob Uszkoreit* Google Research usz@google.com

Llion Jones* Google Research llion@google.com

Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu

Łukasz Kaiser* Google Brain lukaszkaiser@google.com

Illia Polosukhin* ‡ illia.polosukhin@gmail.com

The Transformer

Paper released on 2017 by people from Google Brain & Google Research.

Seq2seq model based on **self-attention**.

Revolutionize NLP and the whole field.

Key Ingredients:

1. Self-attention
2. Multi-head Attention
3. Hierarchical

Seq2seq Models

First let's remember how seq2seq models work.
Transformers where proposed for machine translation.

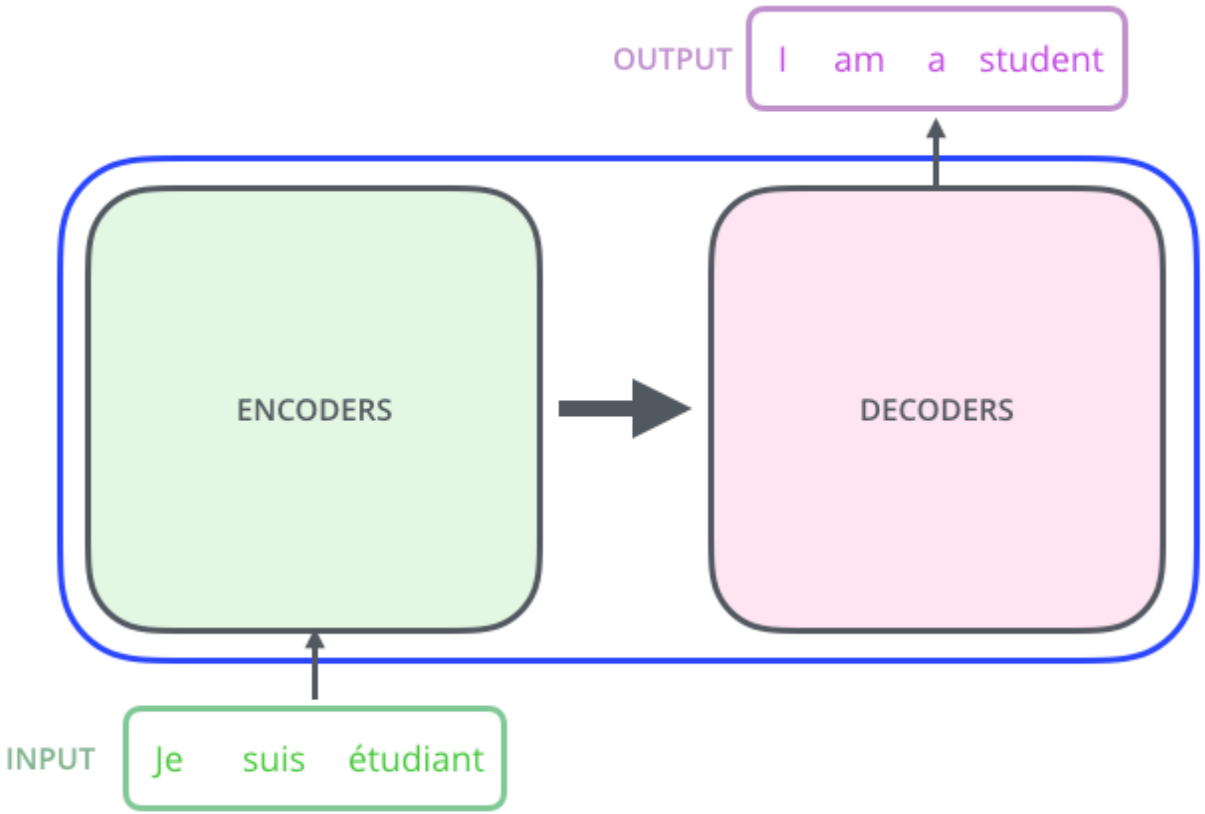
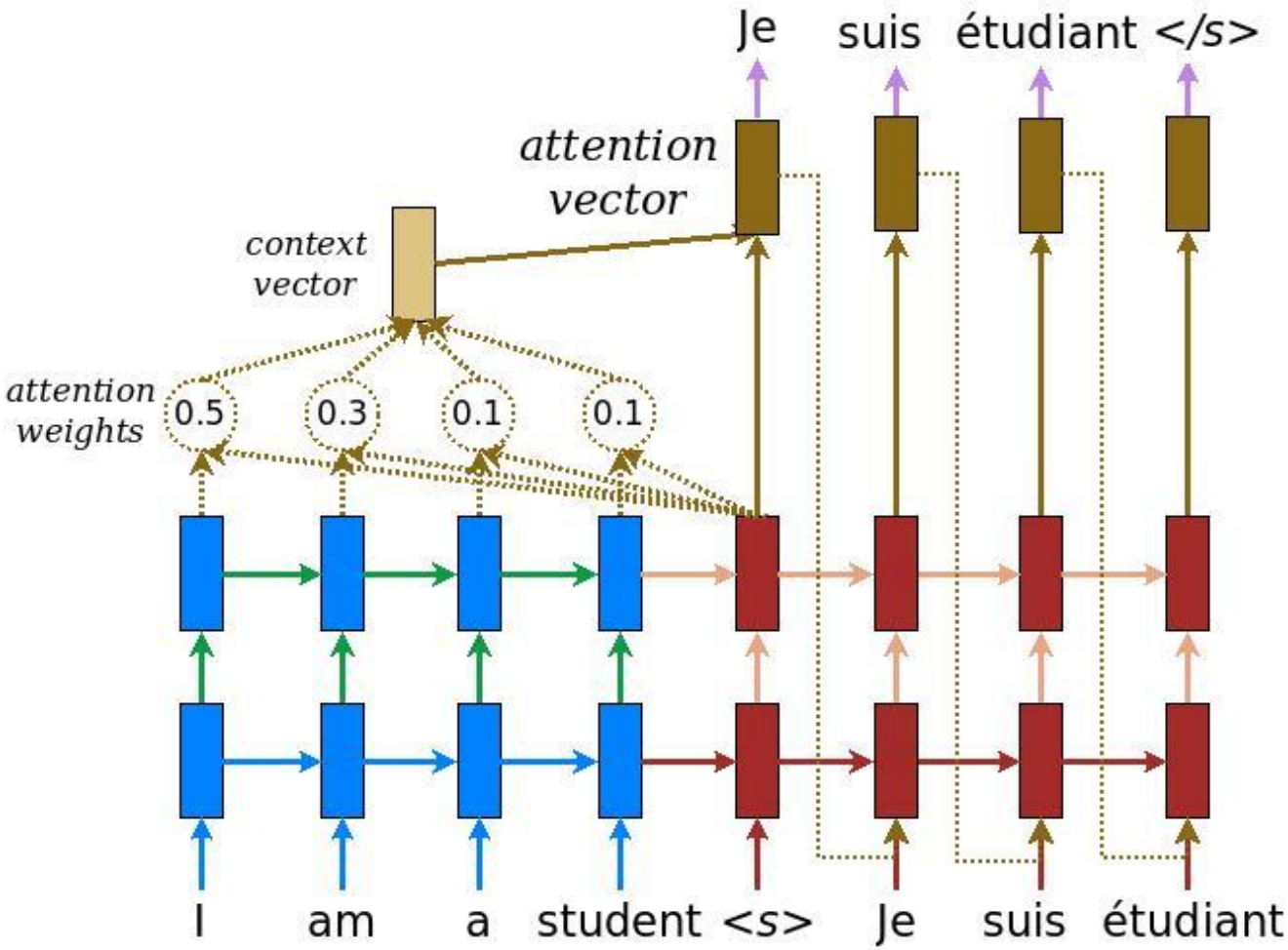


Figure: https://jalammar.github.io/illustrated-transformer/

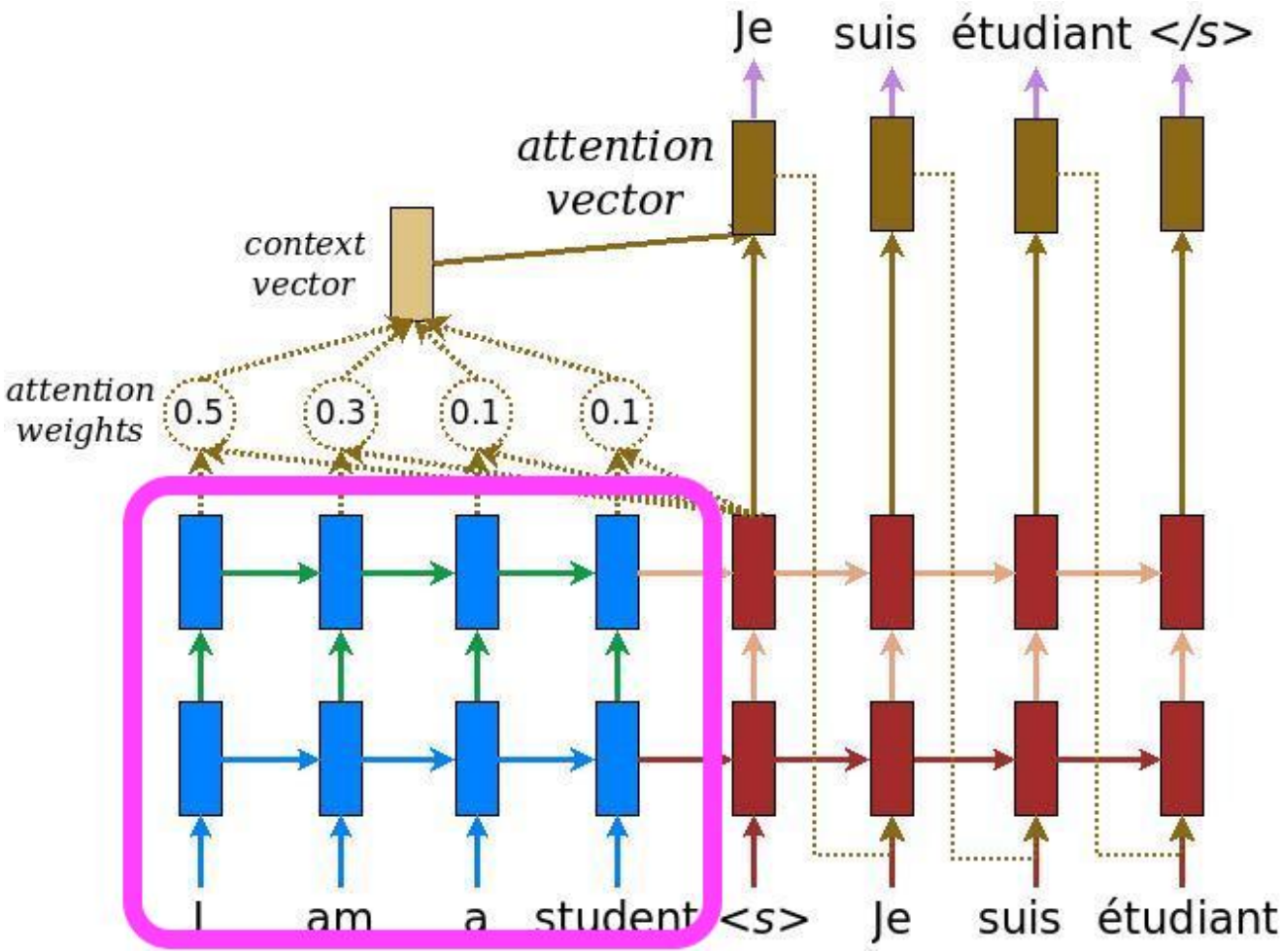
Seq2seq with attention

Review seq2seq with attention.



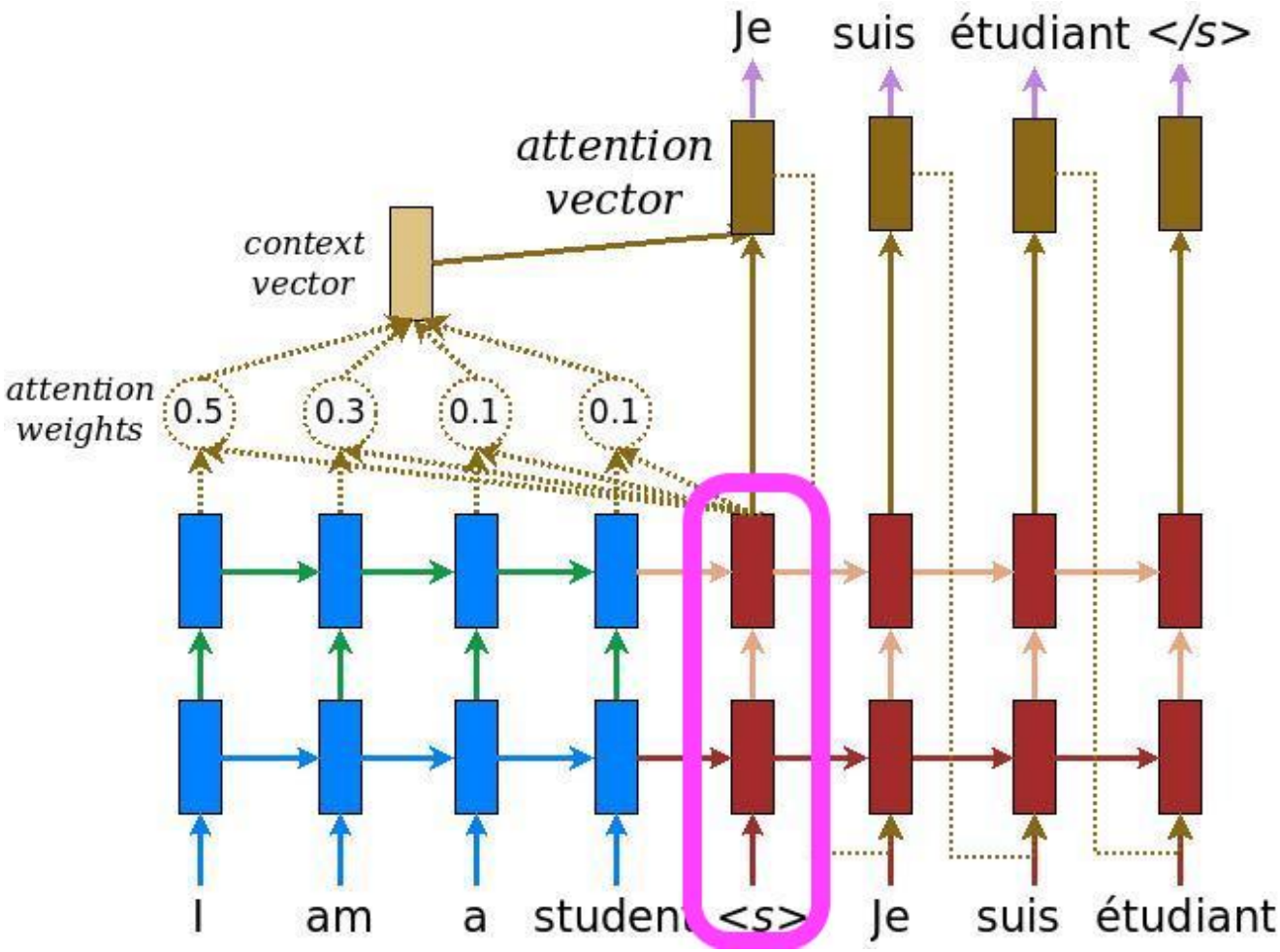
Model review: Seq2seq with Attention

Review seq2seq with attention.



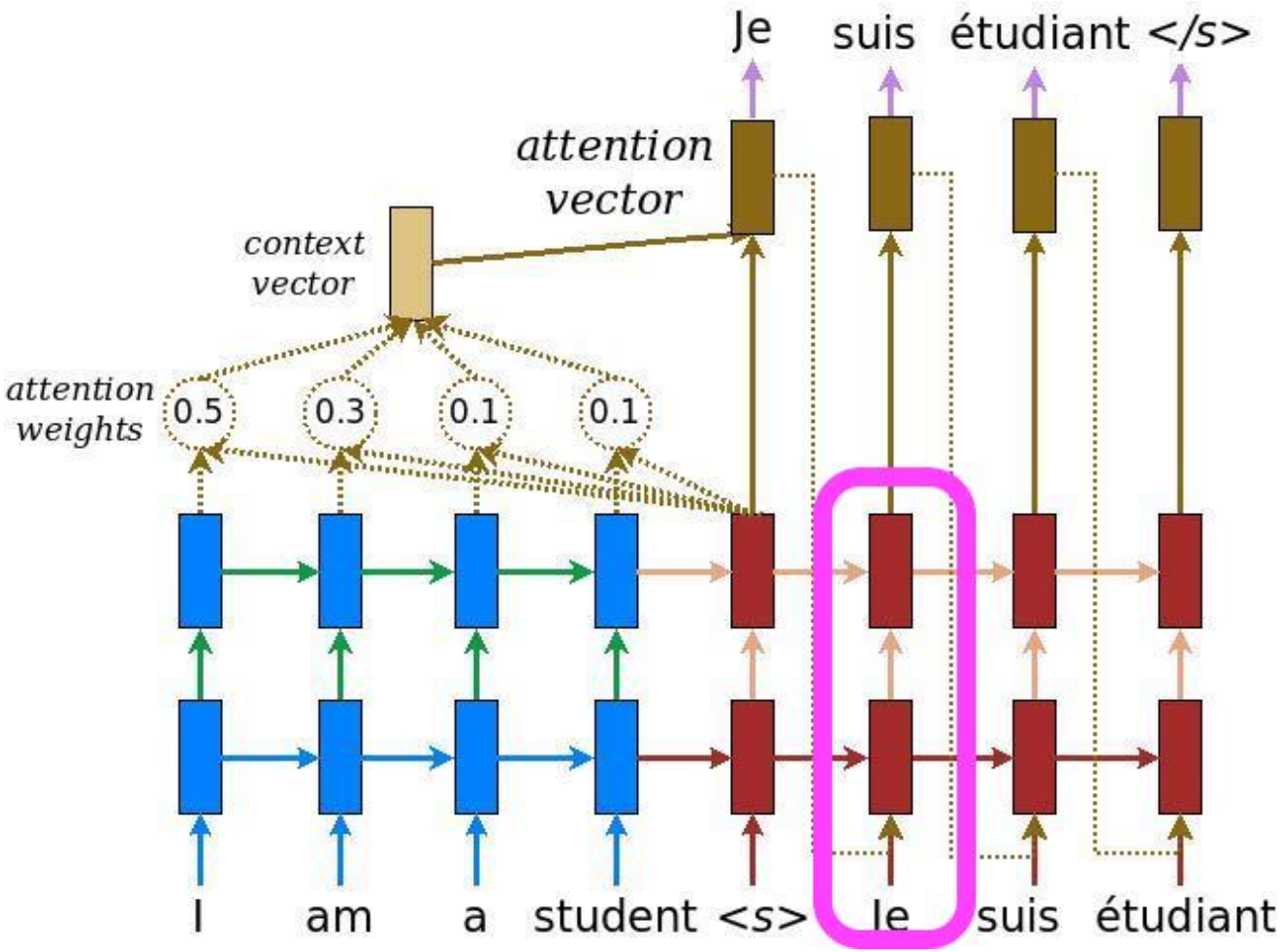
Model review: Seq2seq with Attention

Review seq2seq with attention.



Model review: Seq2seq with Attention

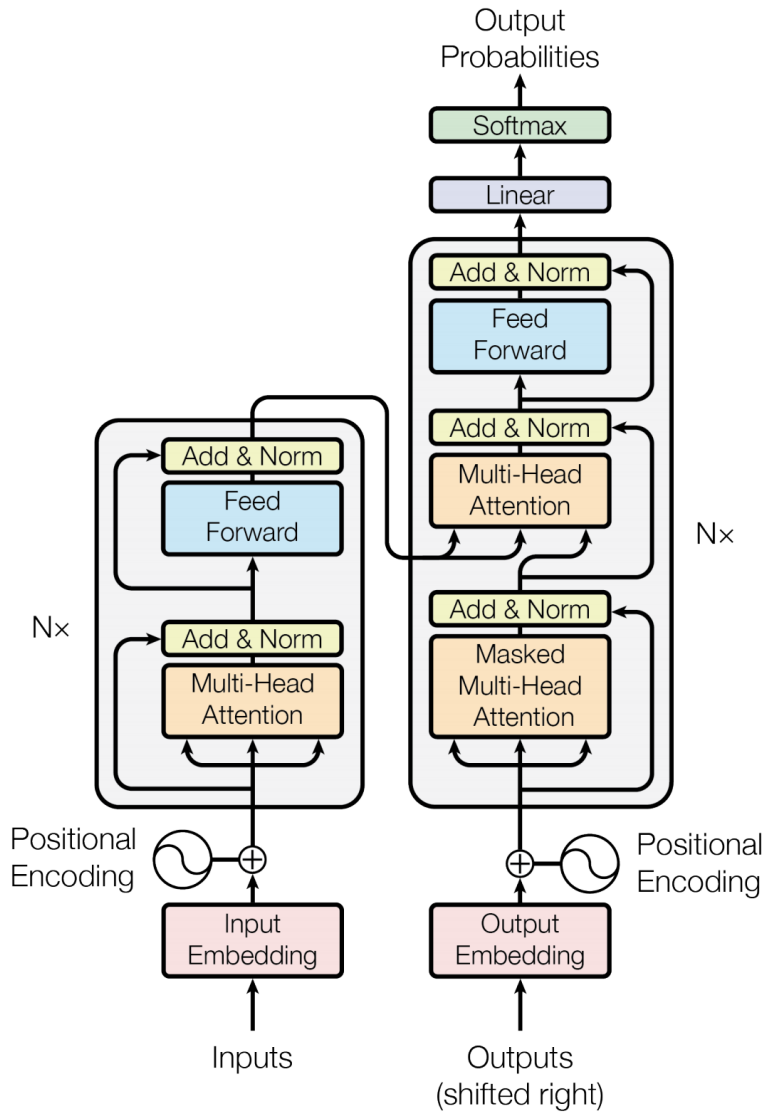
Review seq2seq with attention.



The Transformer

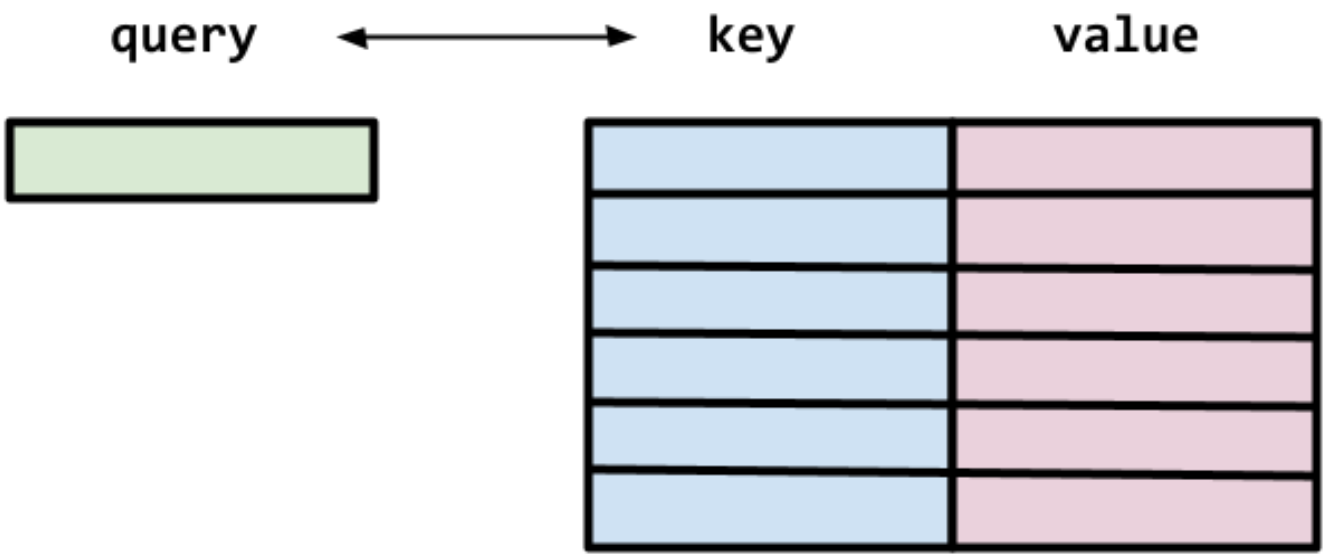
You've use it before.

Let's peek inside it's inner workings.



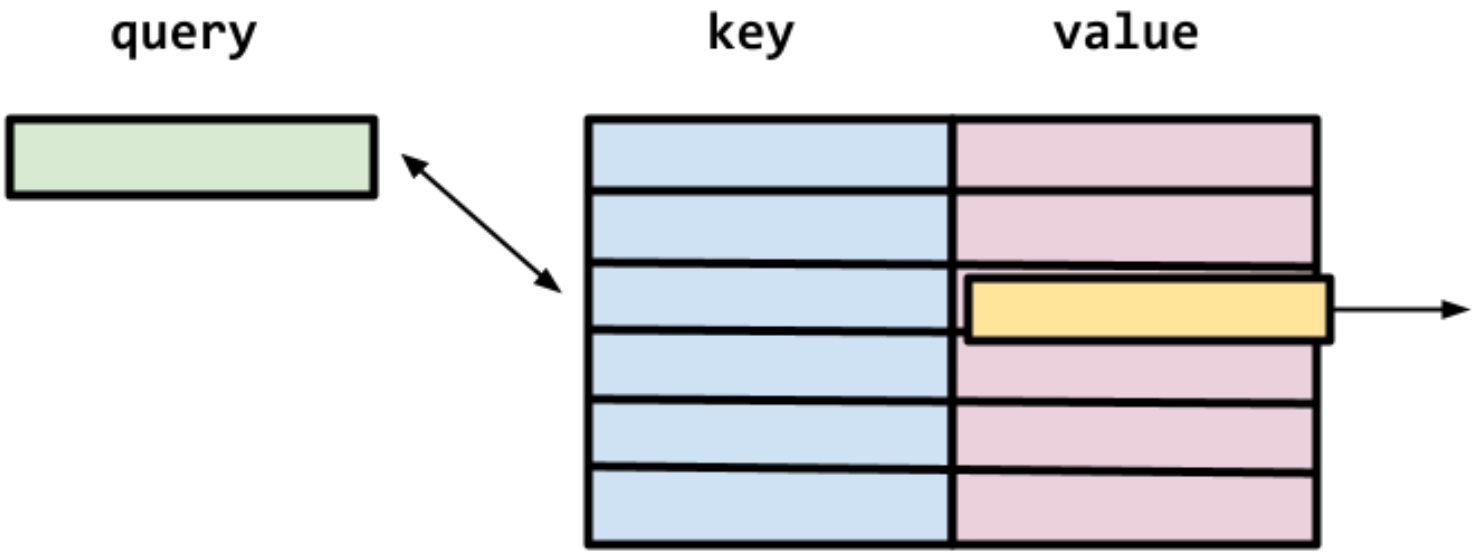
Attention

$$Attention(query, key, value)$$



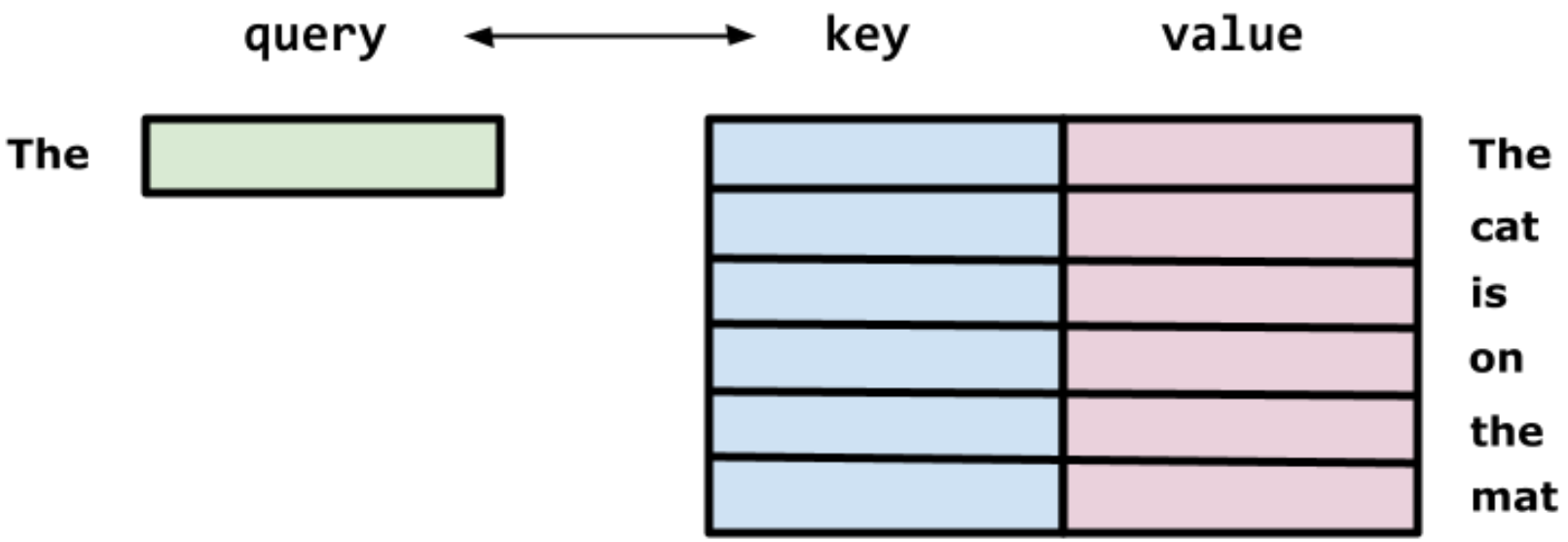
Attention

$$Attention(Q, K, V)$$



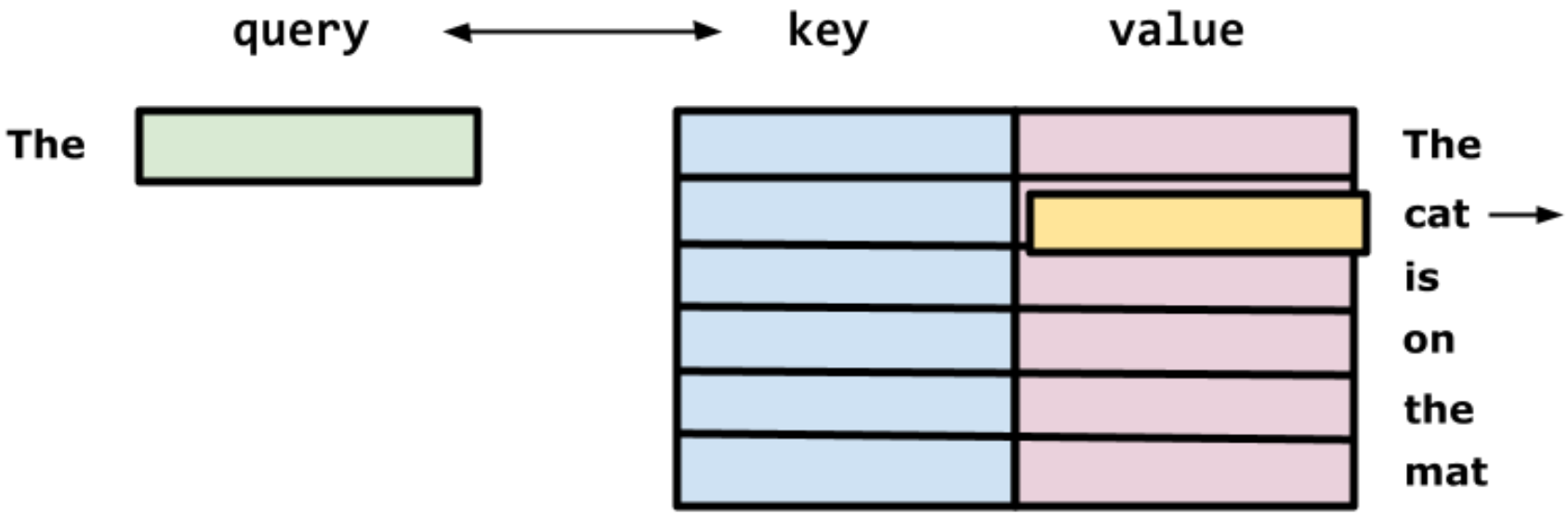
Self-Attention

$$Attention(Q, K, V)$$



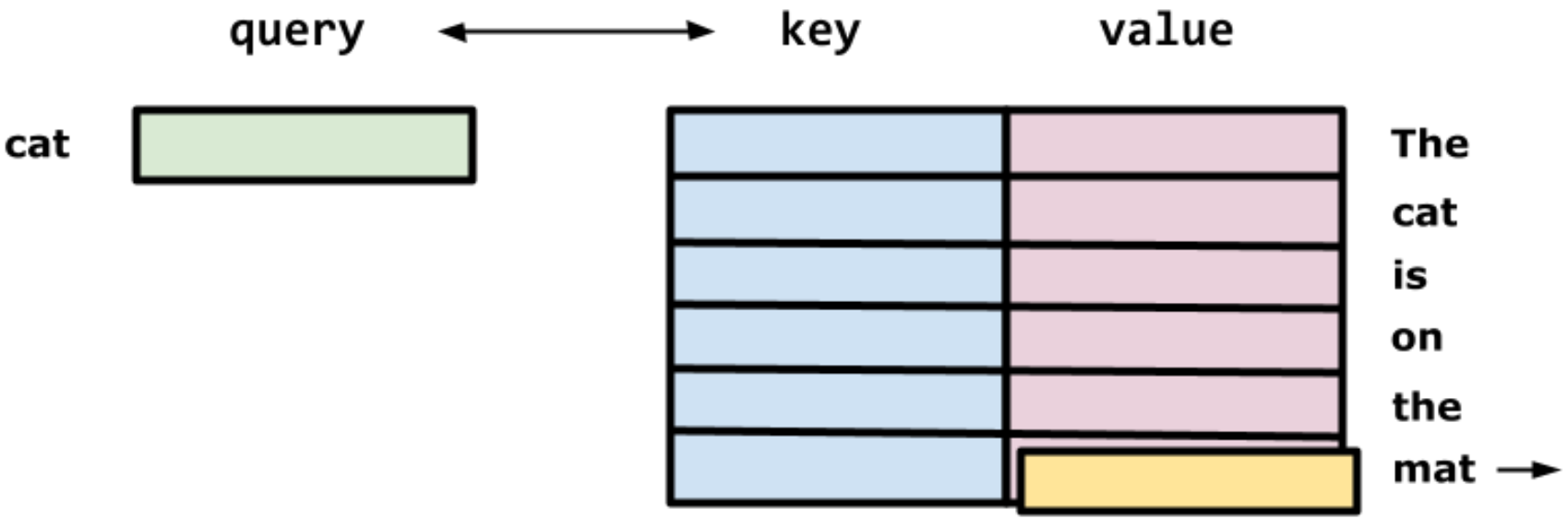
Self-Attention

$$Attention(Q, K, V)$$



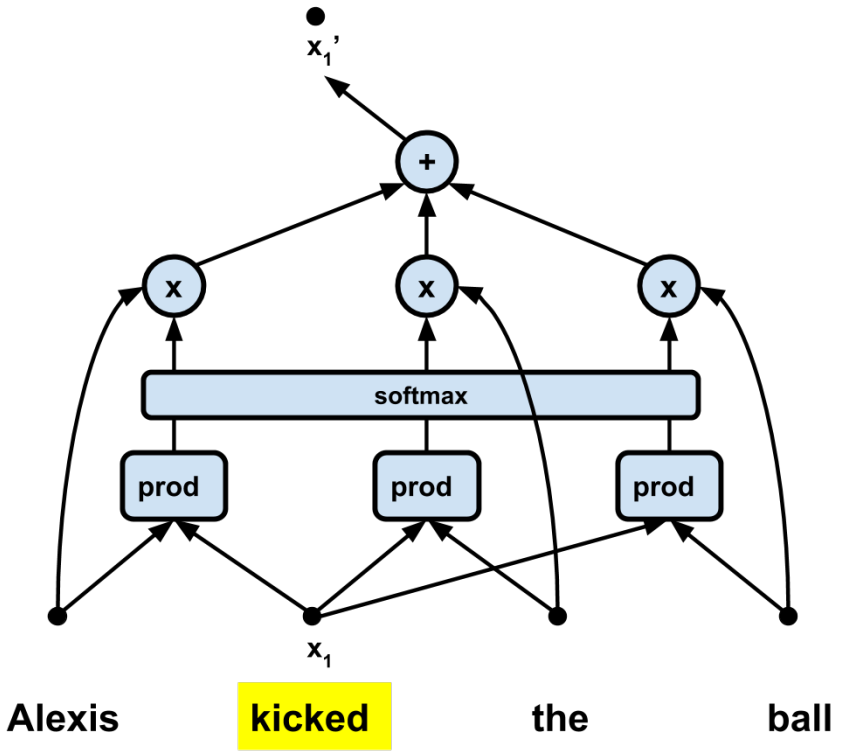
Self-Attention

$$Attention(Q, K, V)$$



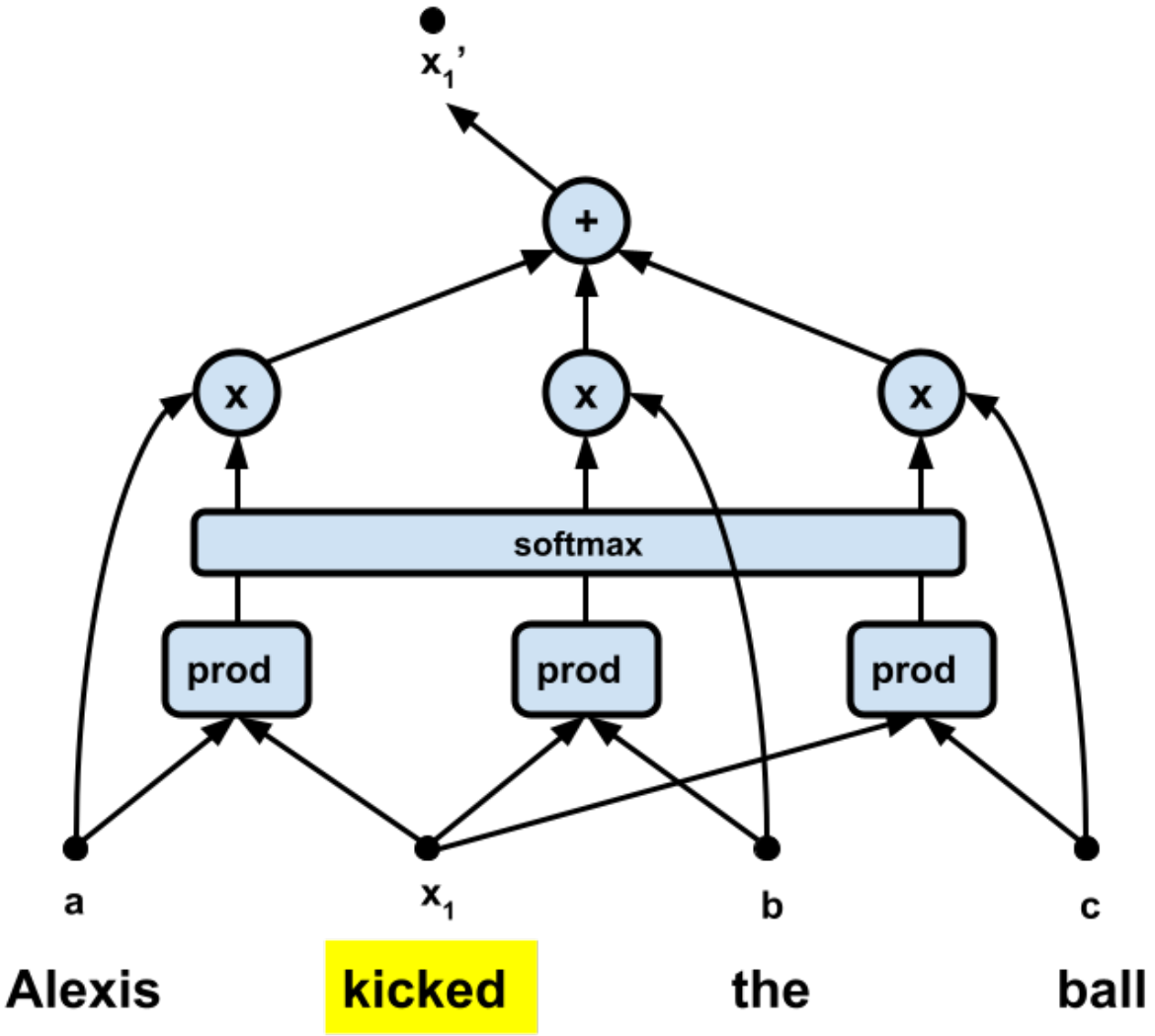
Self-Attention

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



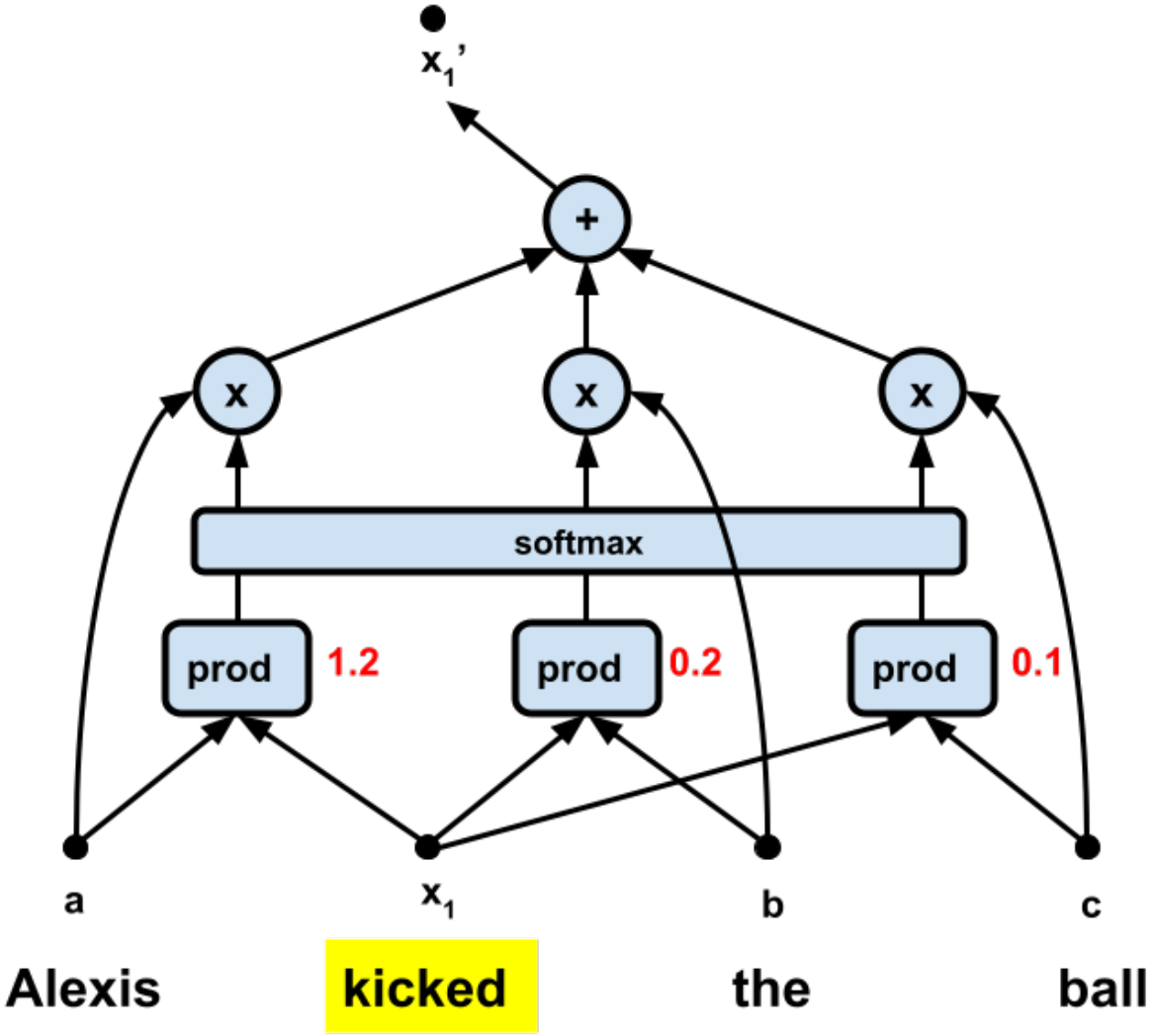
Self-Attention

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



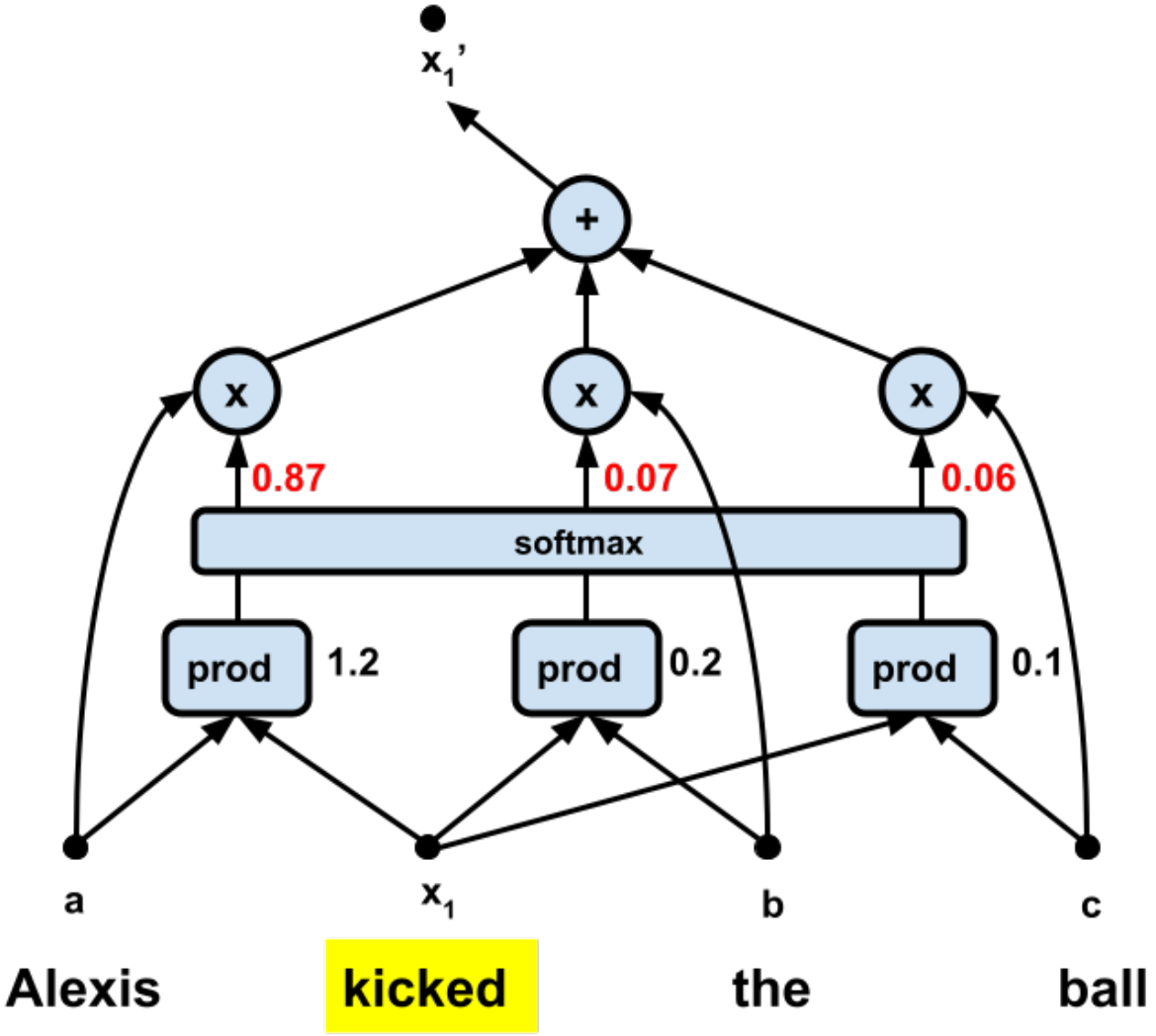
Self-Attention

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



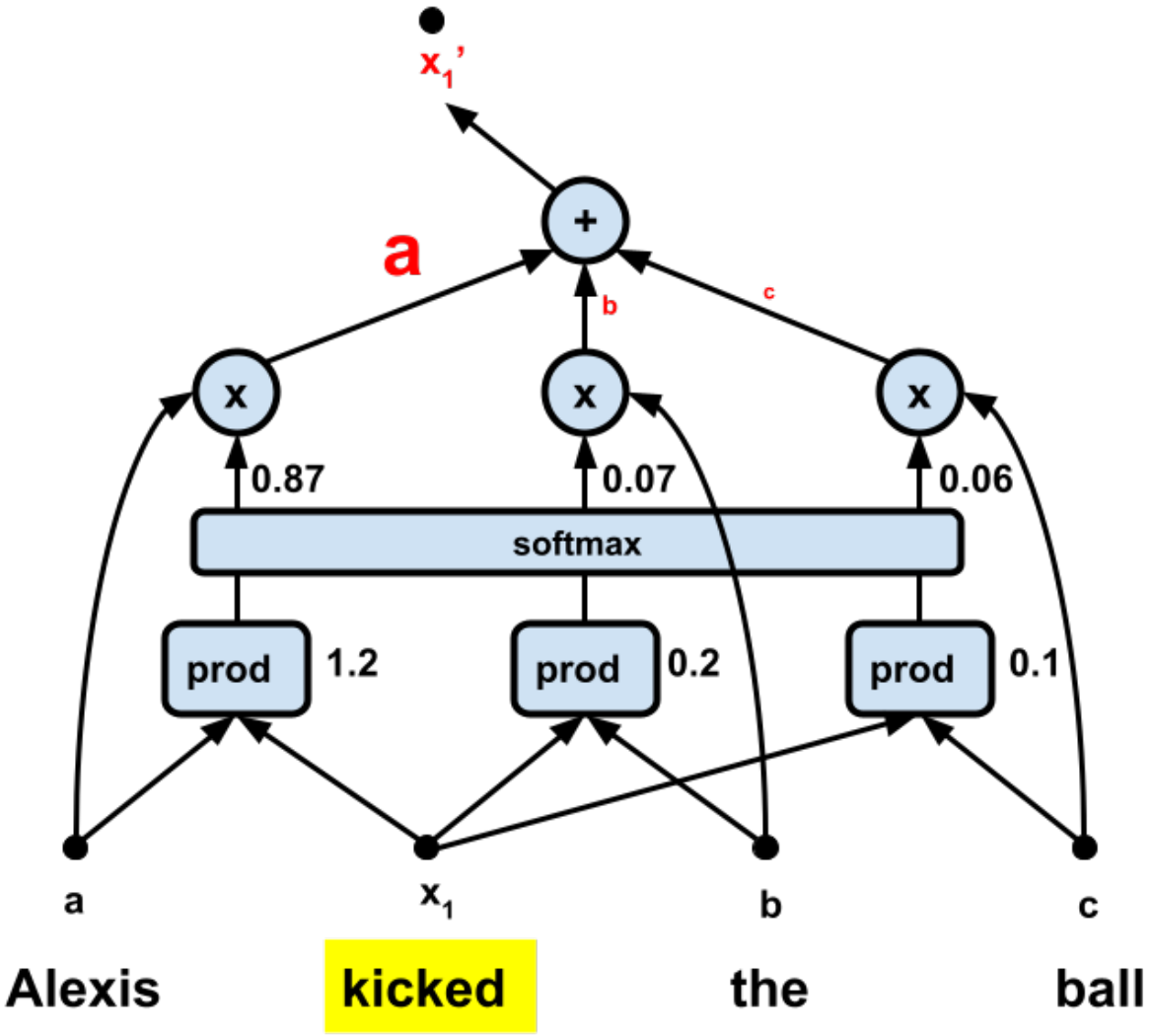
Self-Attention

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



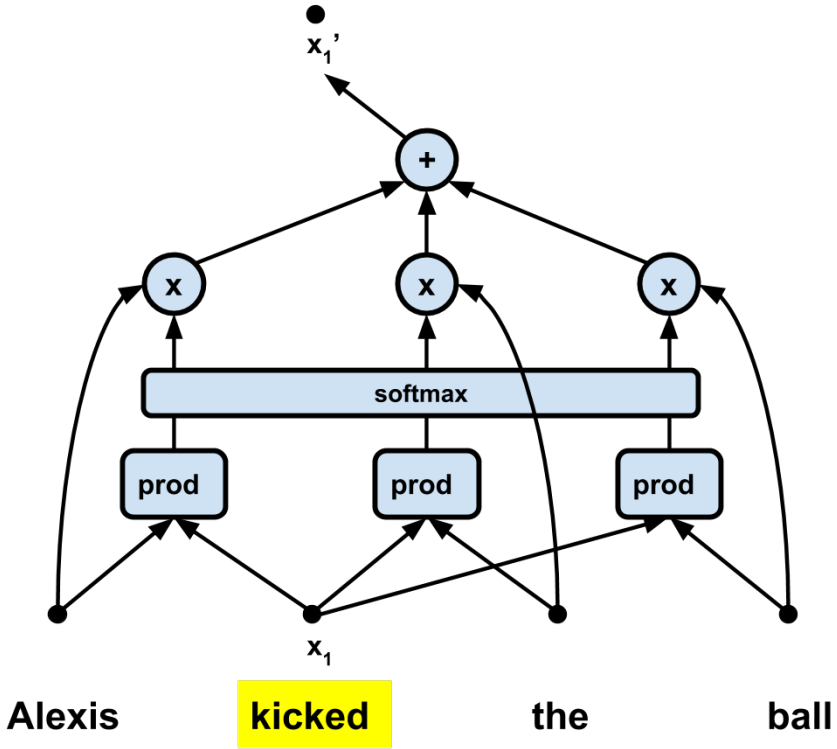
Self-Attention

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



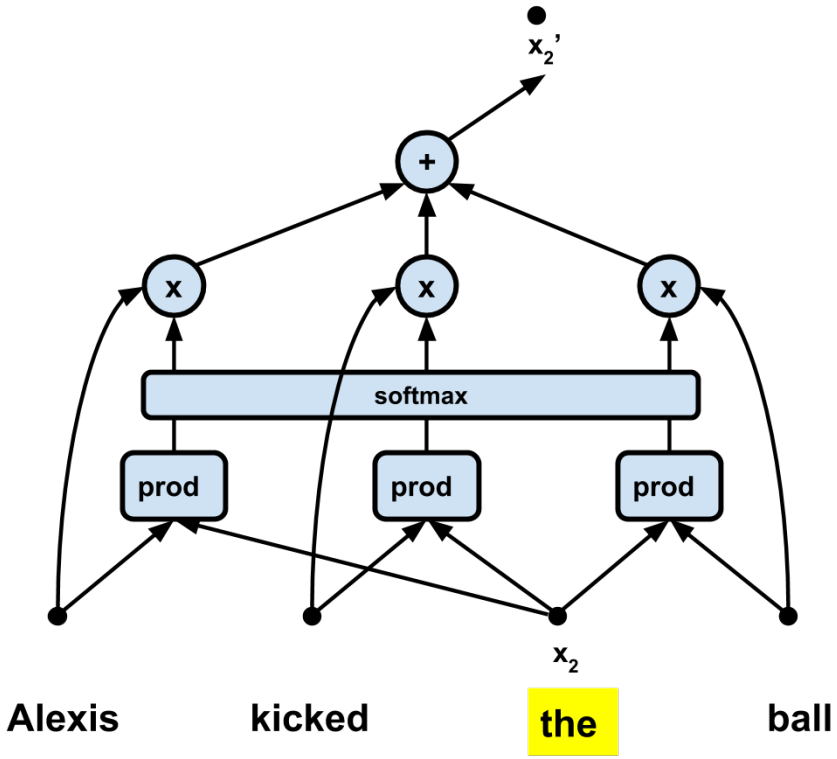
Self-Attention

Re express yourself in terms of a weighted combination of your neighbourhood.



Self-Attention

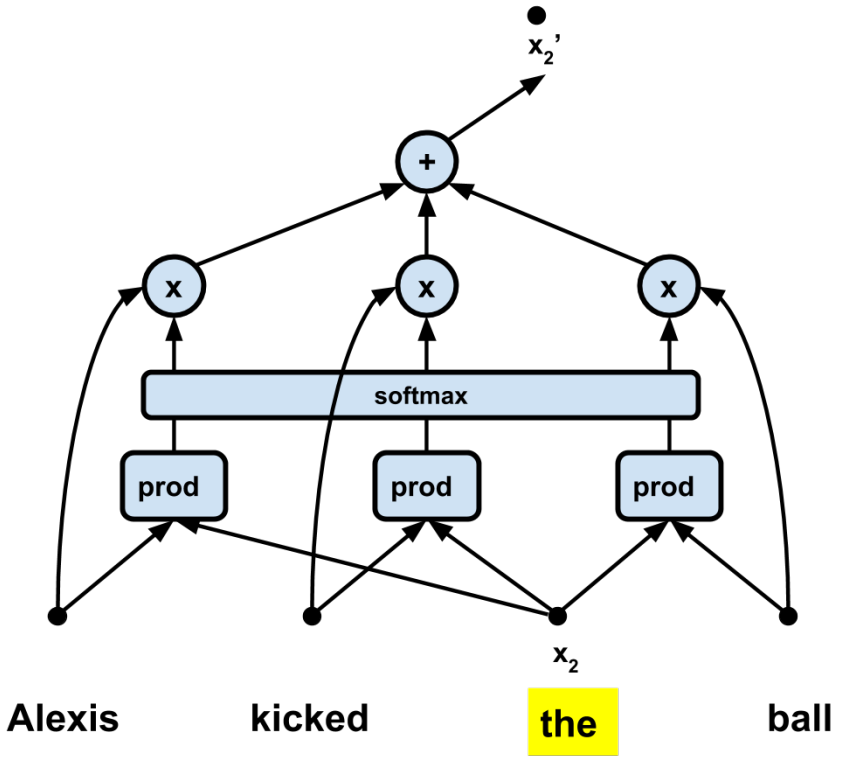
Contextualize every token representation.



Why Self-Attention?

Advantages of self-attention

- Can be easily parallelizable.
- One step distance to every other position.
- Can be stacked forming a hierarchy.



Neuron View

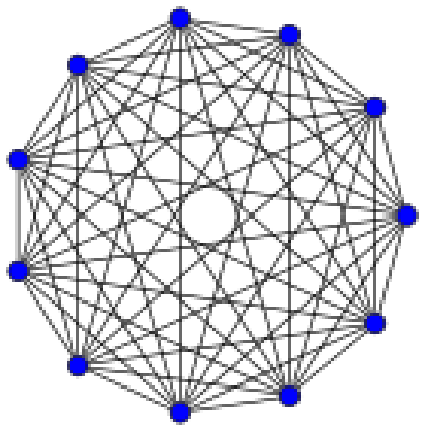
Let's peek the lab for a clearer picture.

Self-Attention as a Relational Bias

We are computing relationships from all tokens (entities) with a given one.

Relational bias

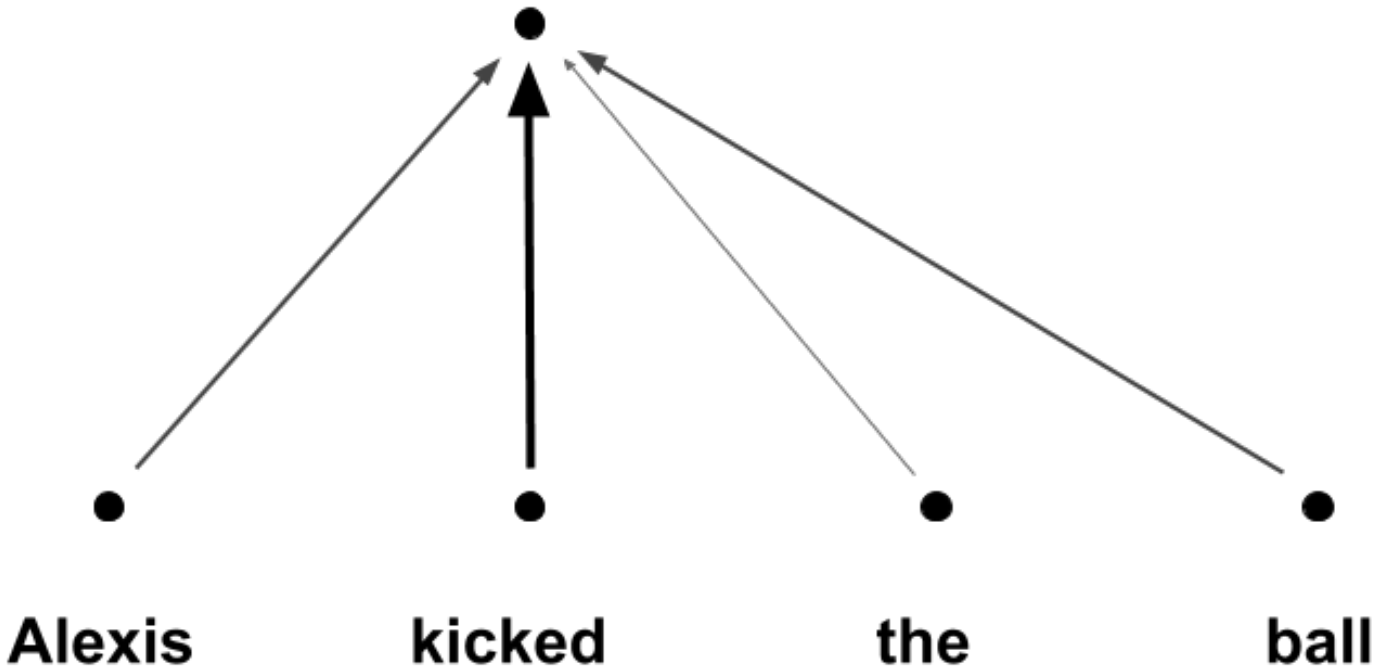
- Tokens are the entities and attention models the relationship between them.
- Impose a fully connected graph.



Multi-Head Attention

Why do we have to restrict what a given word is focusing on?

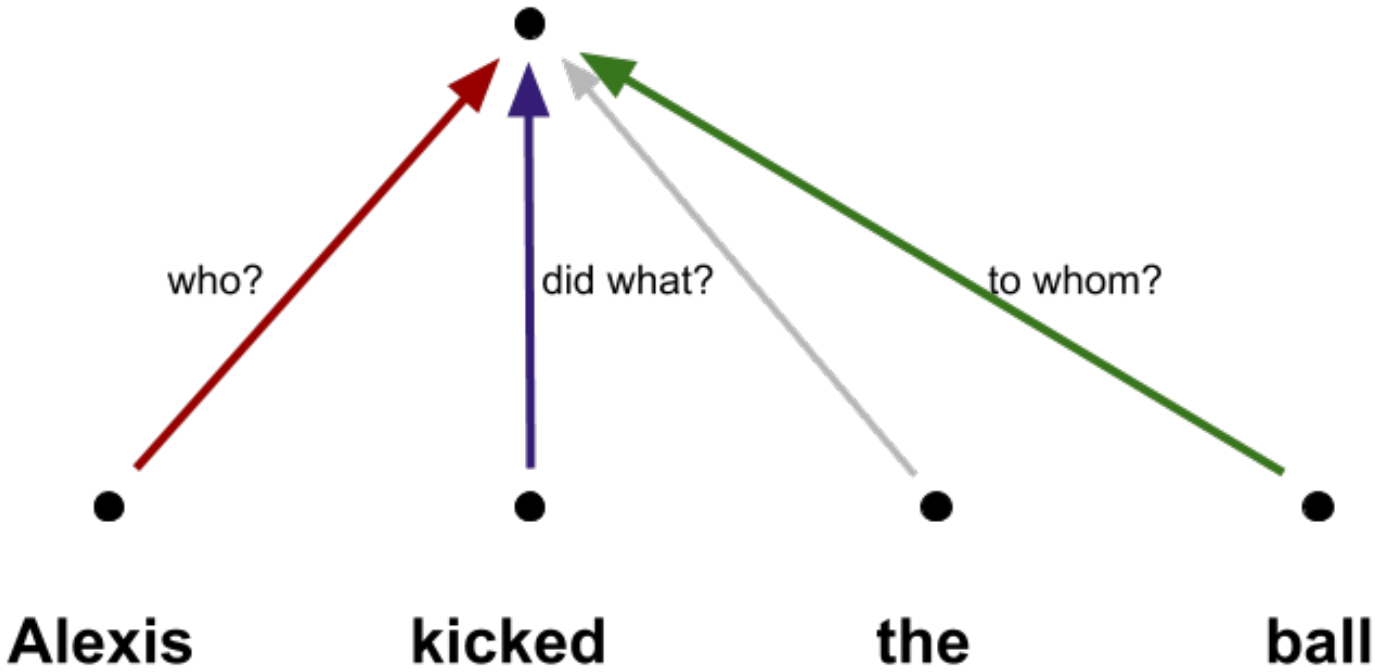
E.g. the verb embedding might want to focus on subject as well as the object.



Multi-Head Attention

Why do we have to restrict what a given word is focusing on?

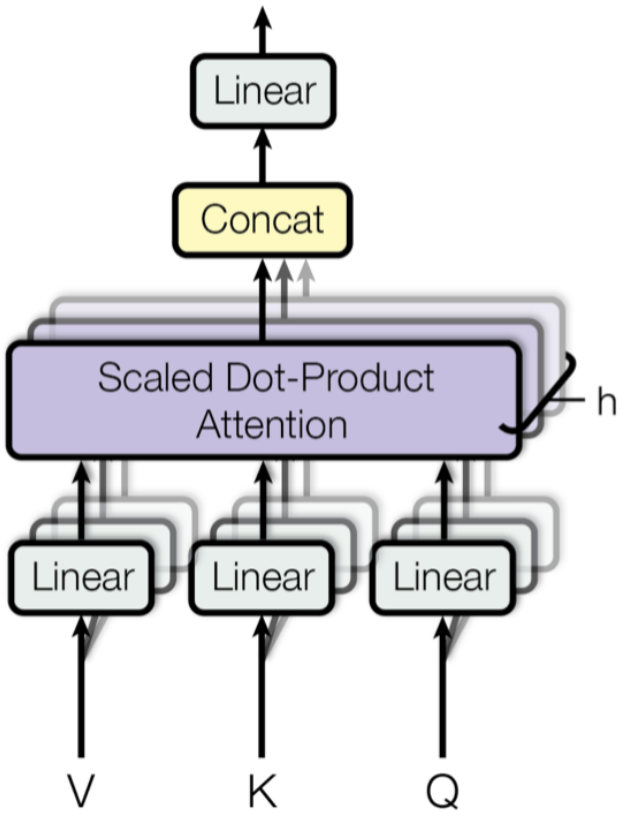
E.g. the verb embedding might want to focus on subject as well as the object.



Multi-Head Attention

$$MultiHeadAttn(Q, K, V) = W^O Concat(head_1, ..., head_h)$$

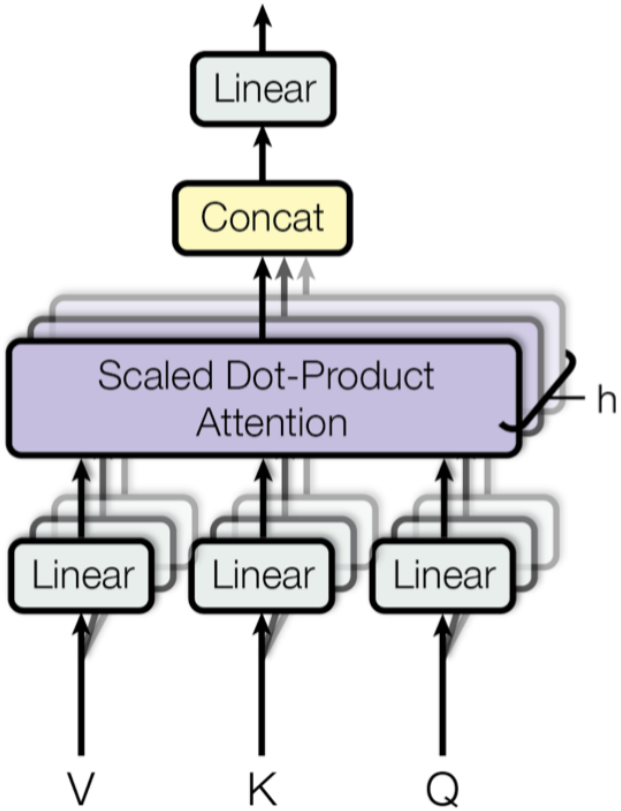
$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$



Multi-Head Attention

$$MultiHeadAttn(Q, K, V) = W^O Concat(head_1, ..., head_h)$$

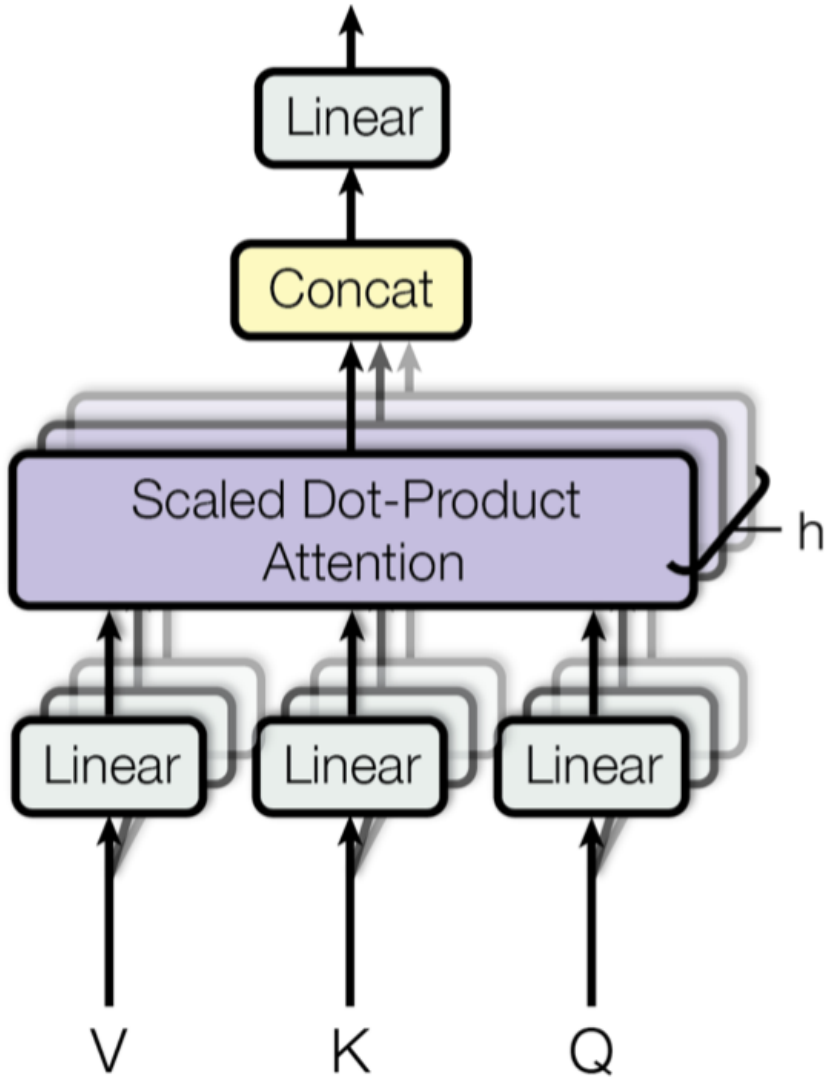
$$head_i = Attention(Q\,W_i^Q, K\,W_i^K, V\,W_i^V)$$



Multi-Head Attention

Allow the model to have multiple attention distributions per embedding.

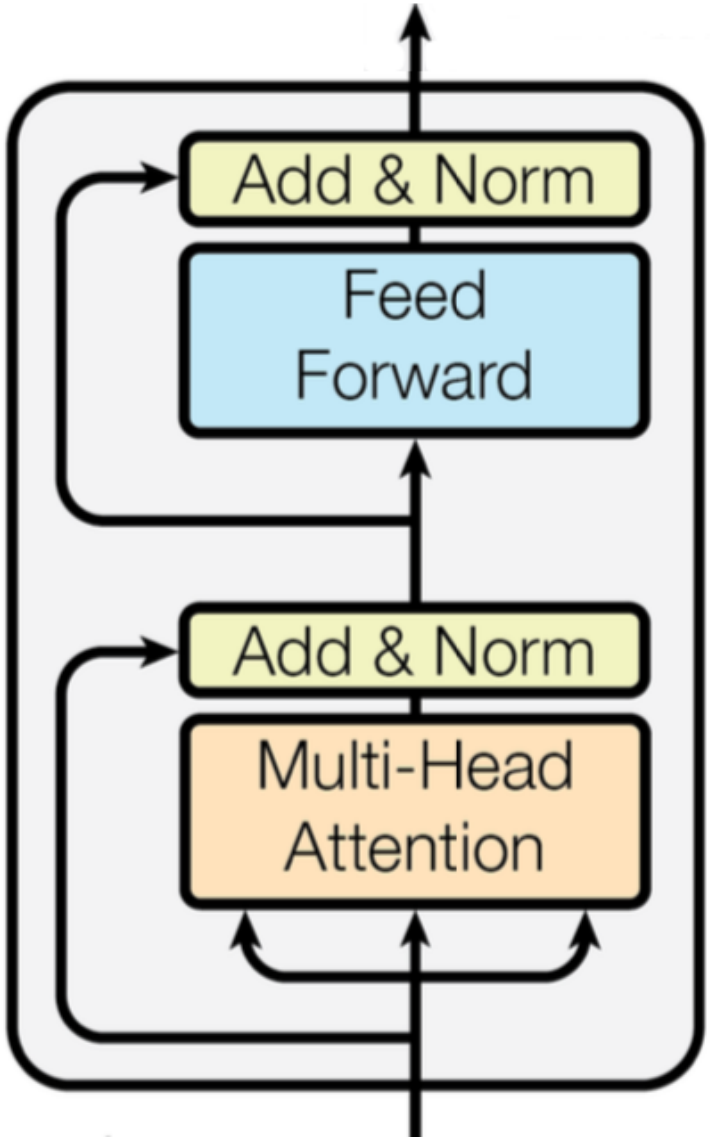
Compose them to create new representations.



Transformer Layer

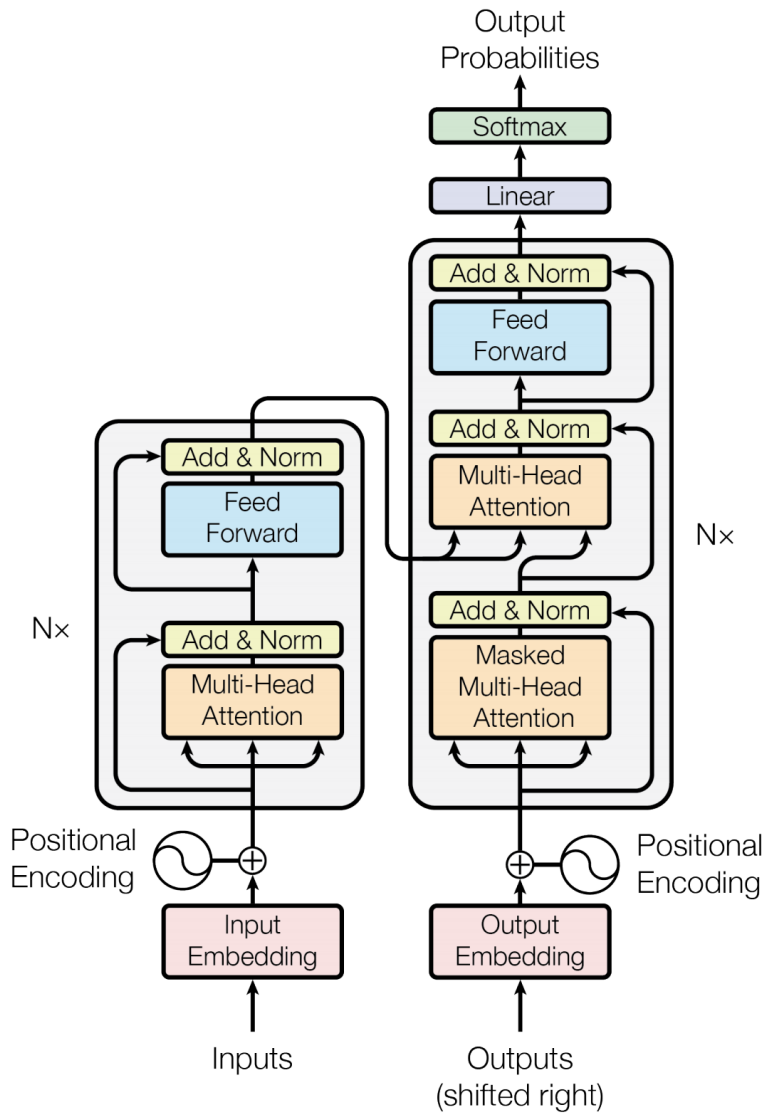
MultiHead Attention +

- Residual connections.
- Layer normalization
- Position-wise Feed-forward Layer ($\sim$ Fully connected layer).



Full Architecture

Stack of N Transformer layers



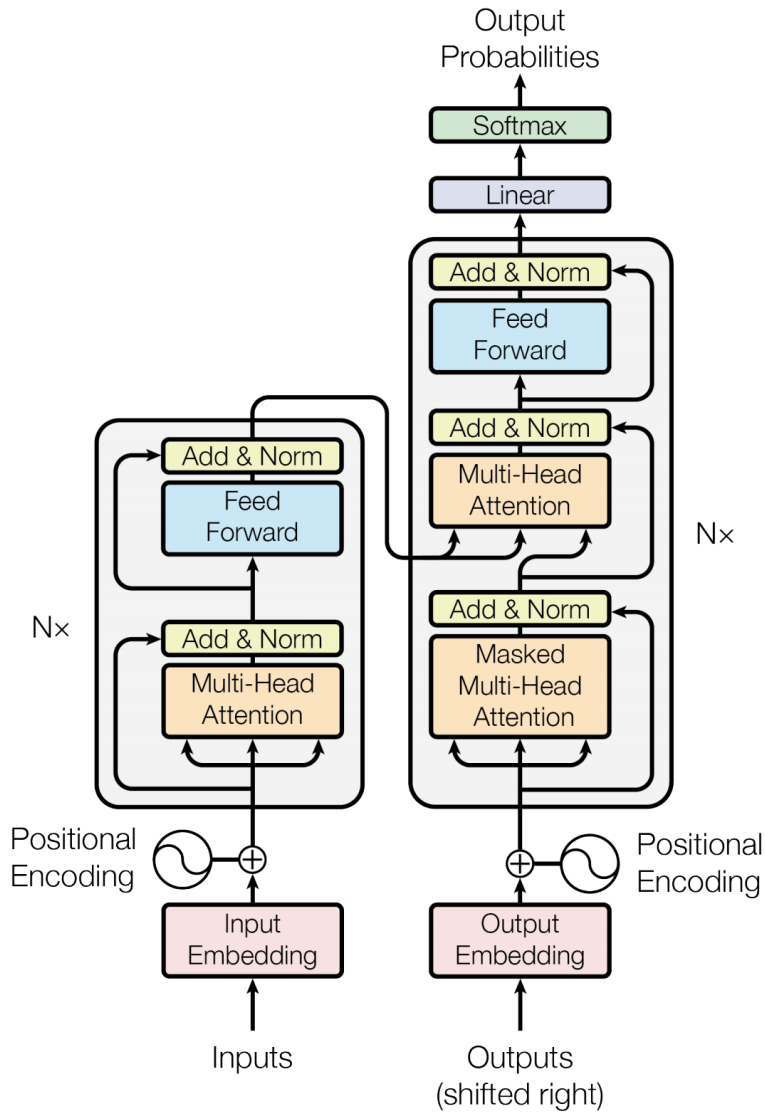
Full Architecture

Positional embeddings

$$PE(p, 2i) = sin(p/10000^{2i/d_{model}})$$

$$PE(p, 2i+1) = cos(p/10000^{2i/d_{model}})$$

Or learned.

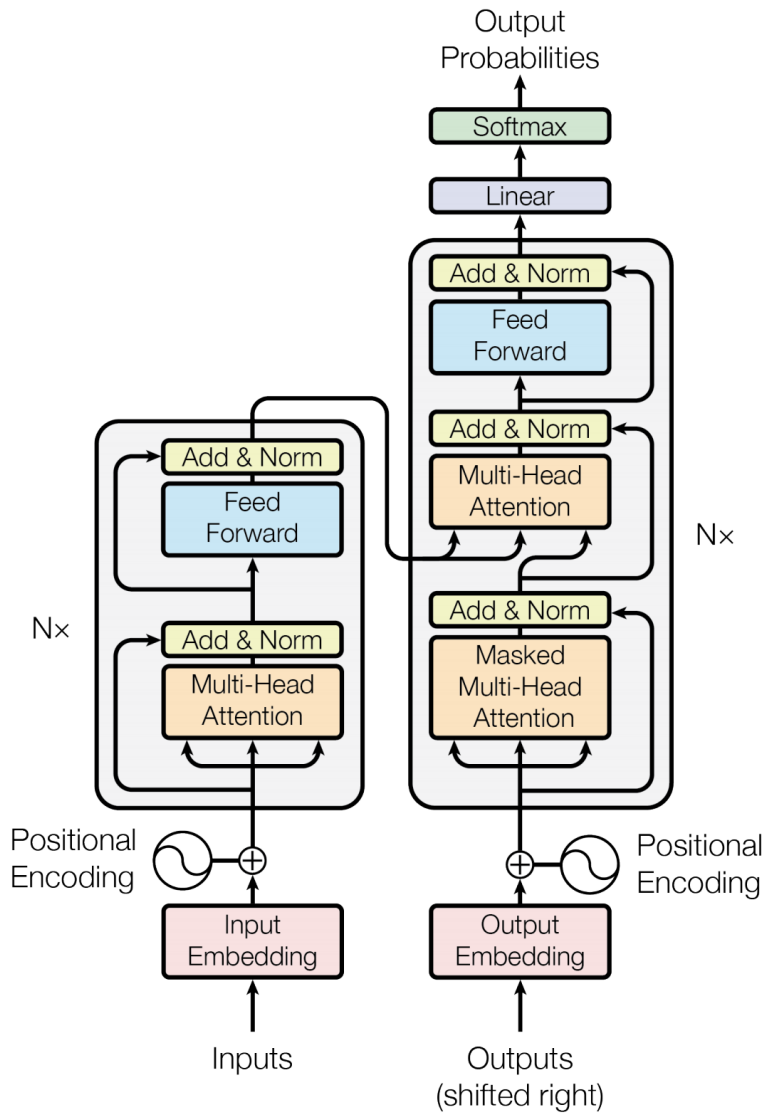


Full Full Architecture

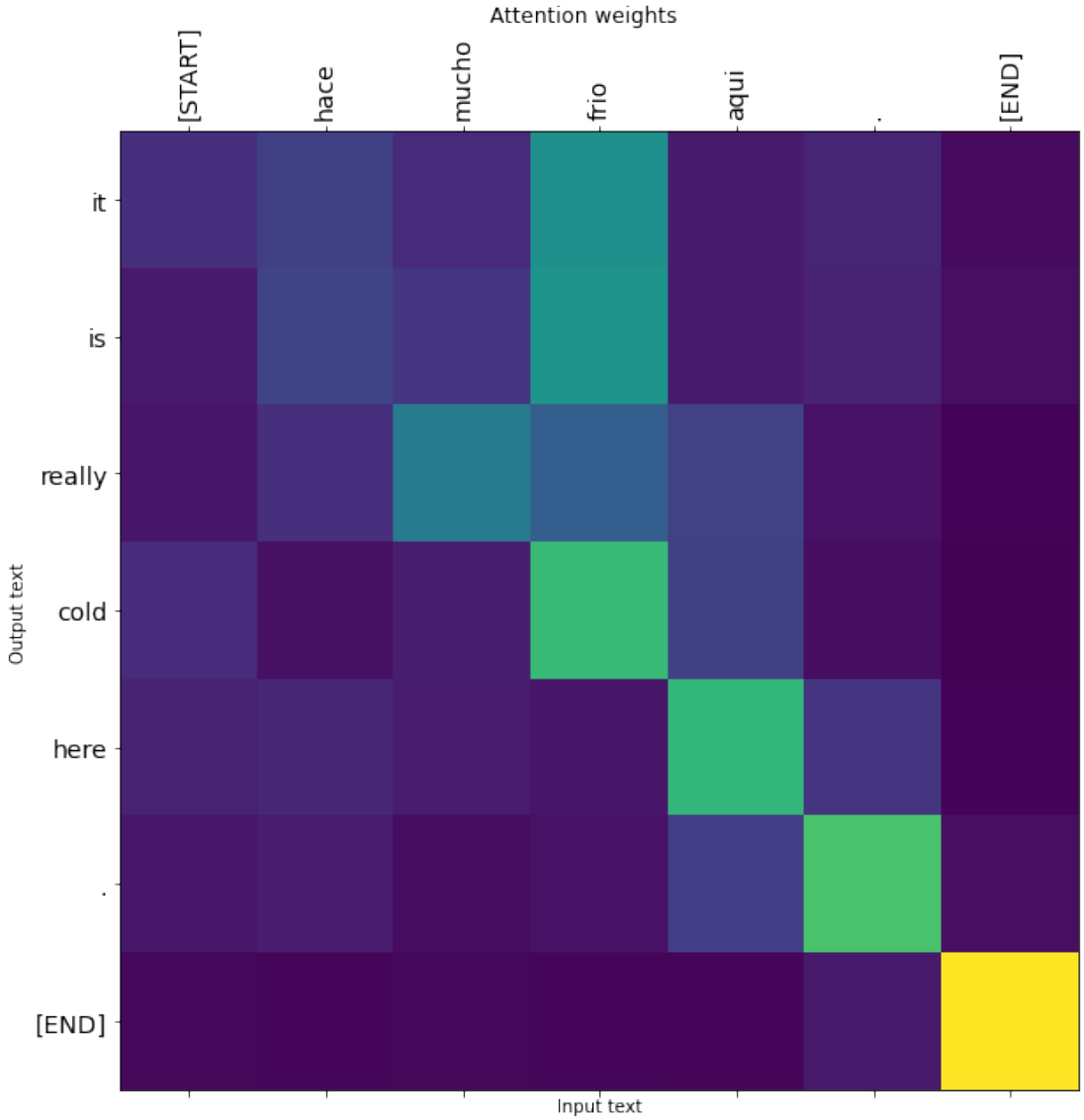
Encoder + Decoder

Decoder

- Similar to encoder with some considerations.
- Cross-attention.
- Masking of the future.



Cross-Attention



Full Architecture

Figure: Click me!

Transformer for Multi-modal Learning

Roadmap

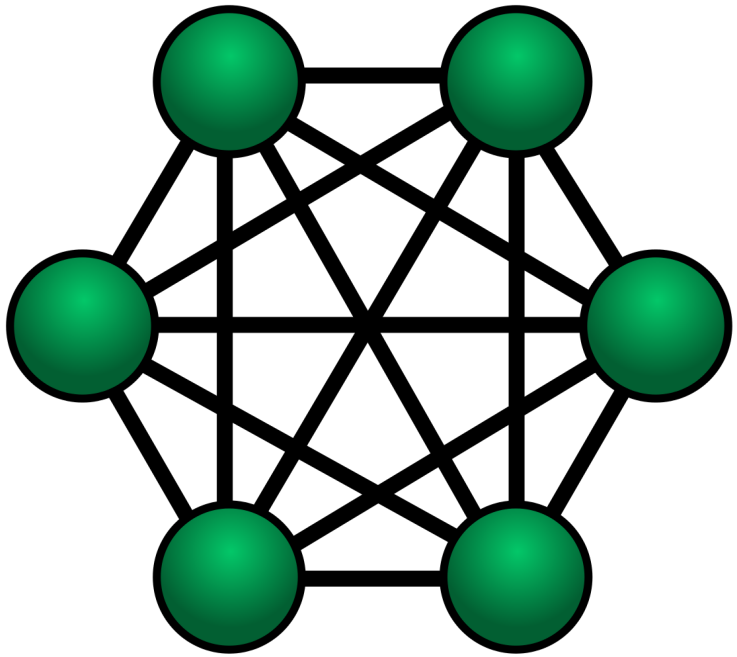
Vision Transformers + CLIP

1. **Vision Transformers**: Model
2. **CLIP**: Training Scheme

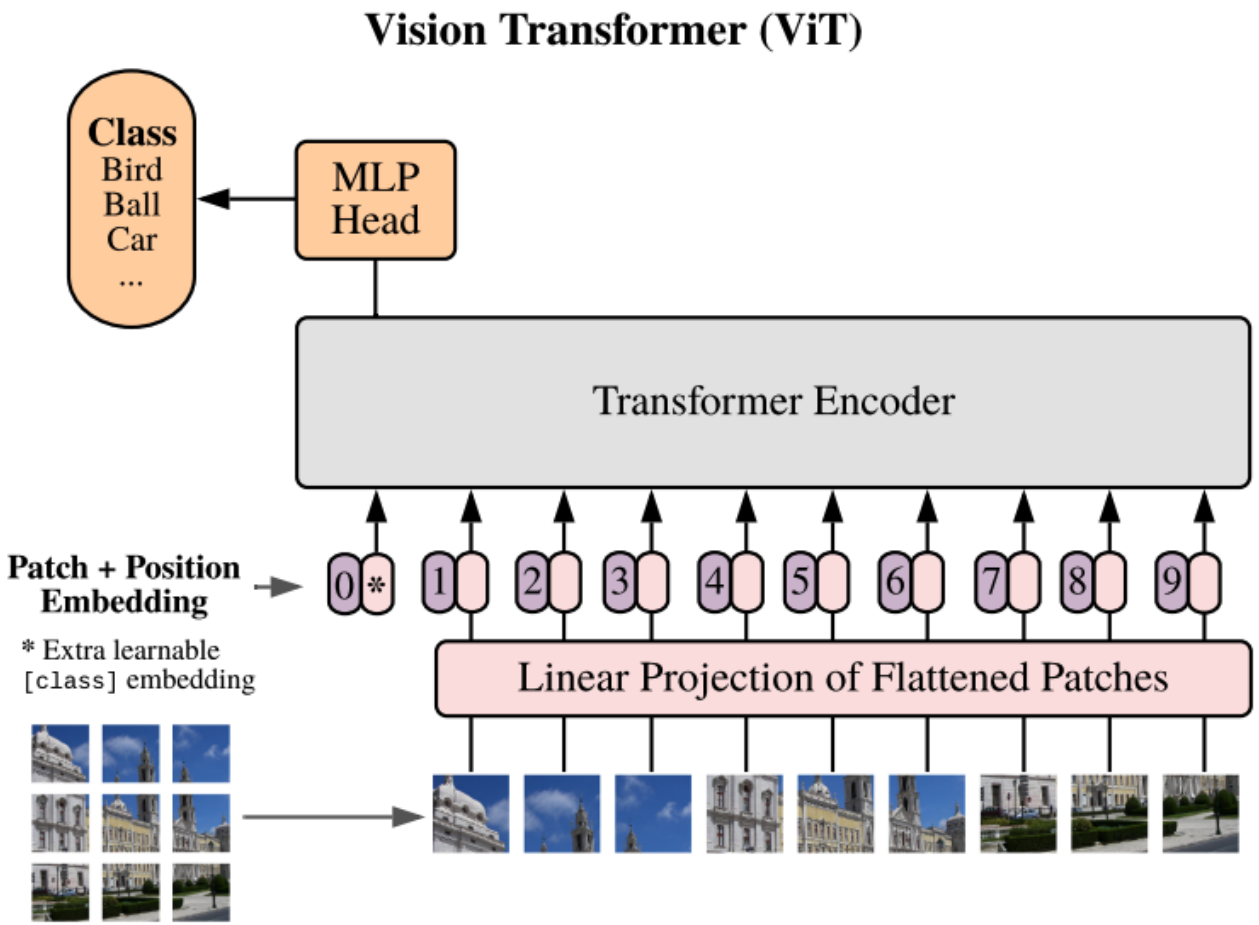
Transformer in Vision

Issues with the transformer for images?

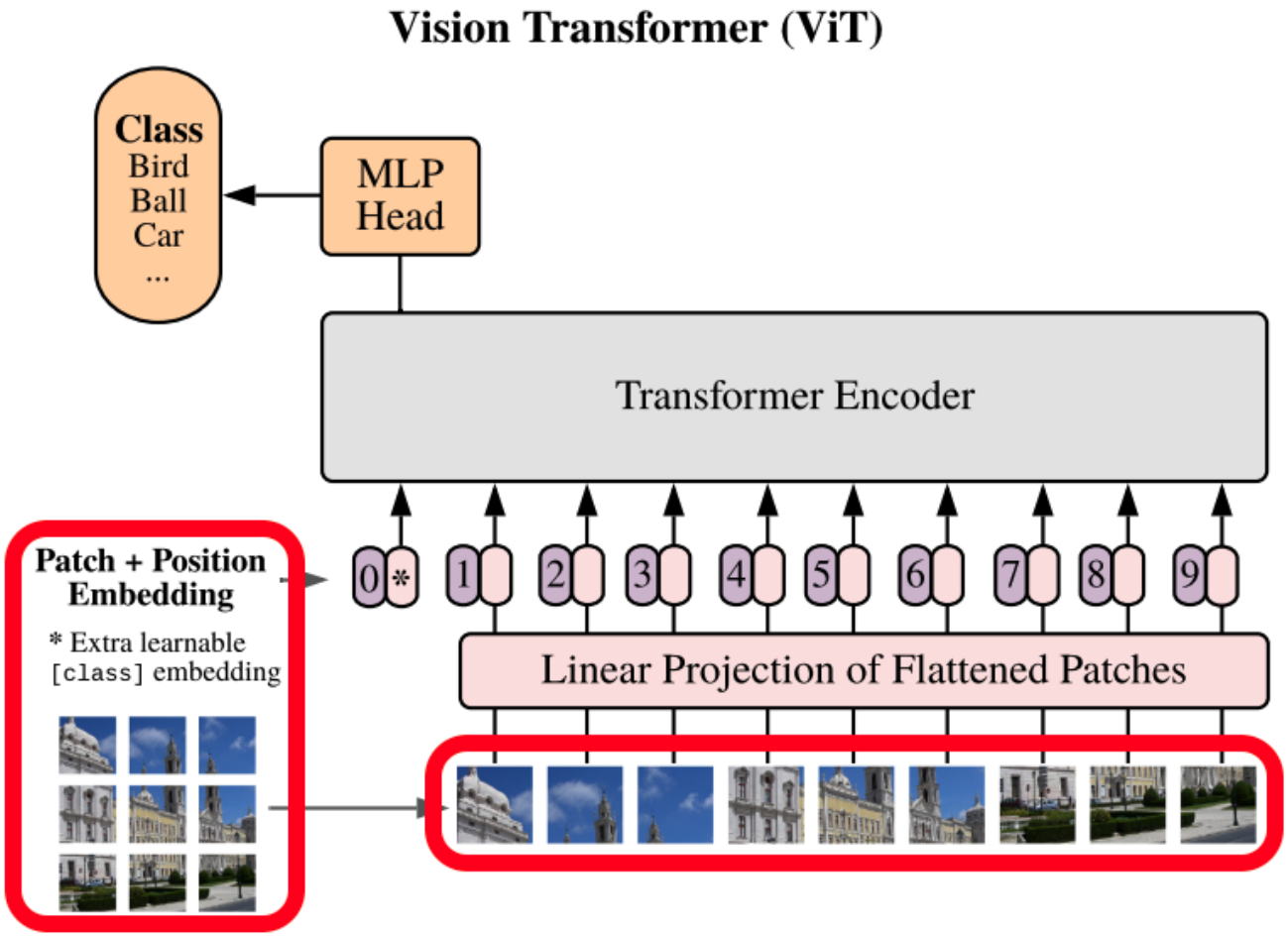
- Pixel space is much larger.
- Complexity scales O^2



Overview

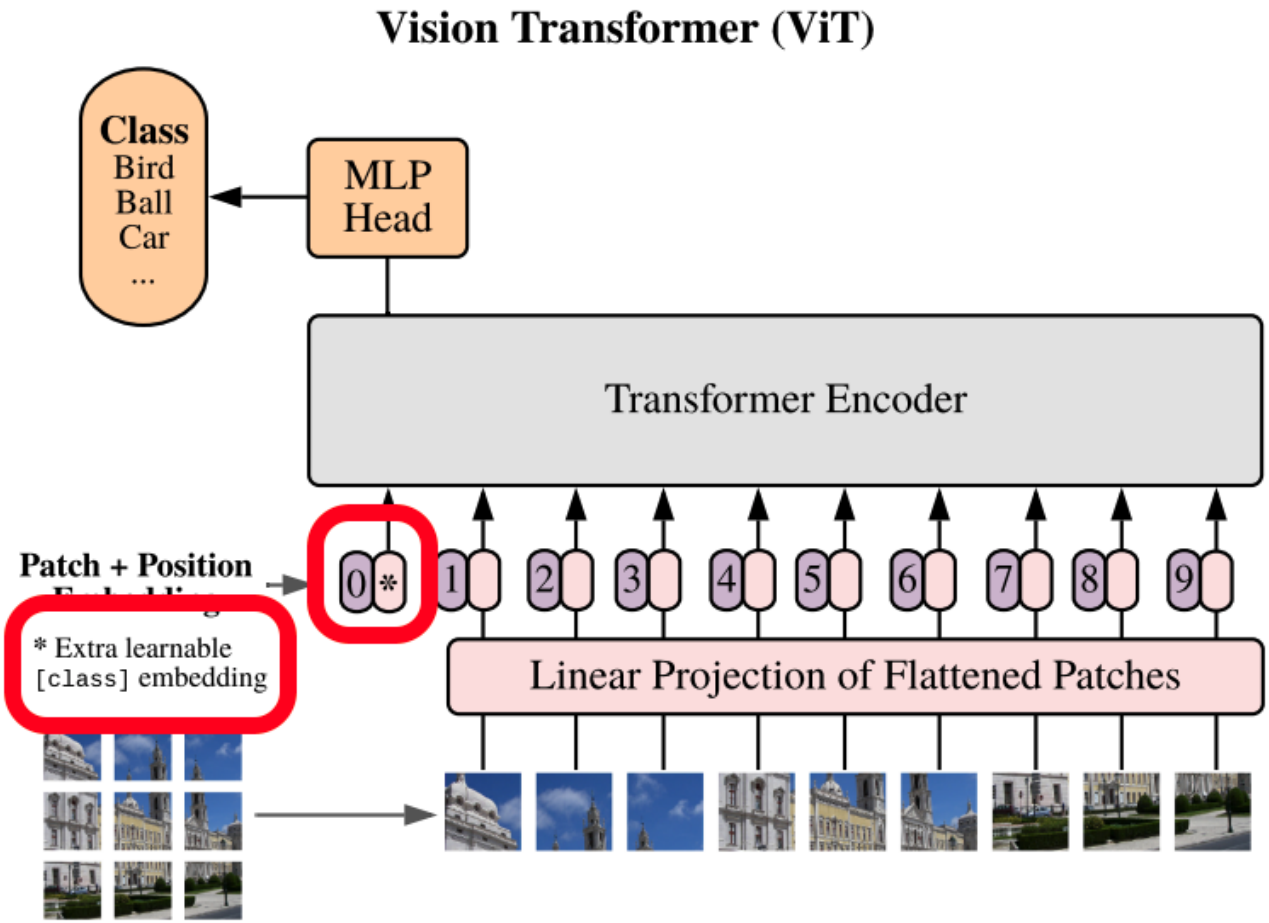

Vision Transformer (ViT)
Class
Bird
Ball
Car
...
MLP Head
Transformer Encoder
Patch + Position Embedding
* Extra learnable [class] embedding
0
*
1
2
3
4
5
6
7
8
9
Linear Projection of Flattened Patches

Divide image in patches (originally 16x16)

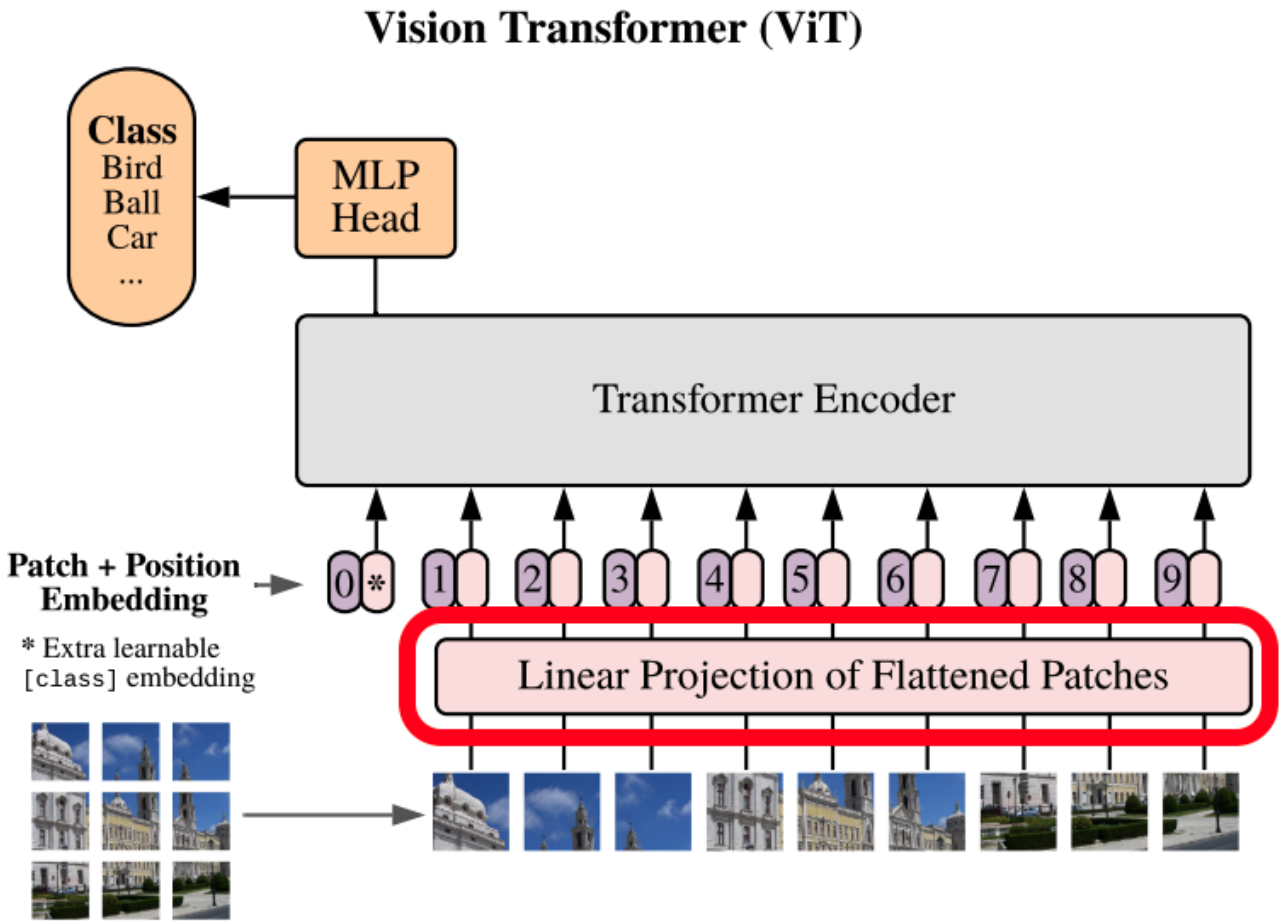


Vision Transformer Model

Add a special token for classification (`[class]`).

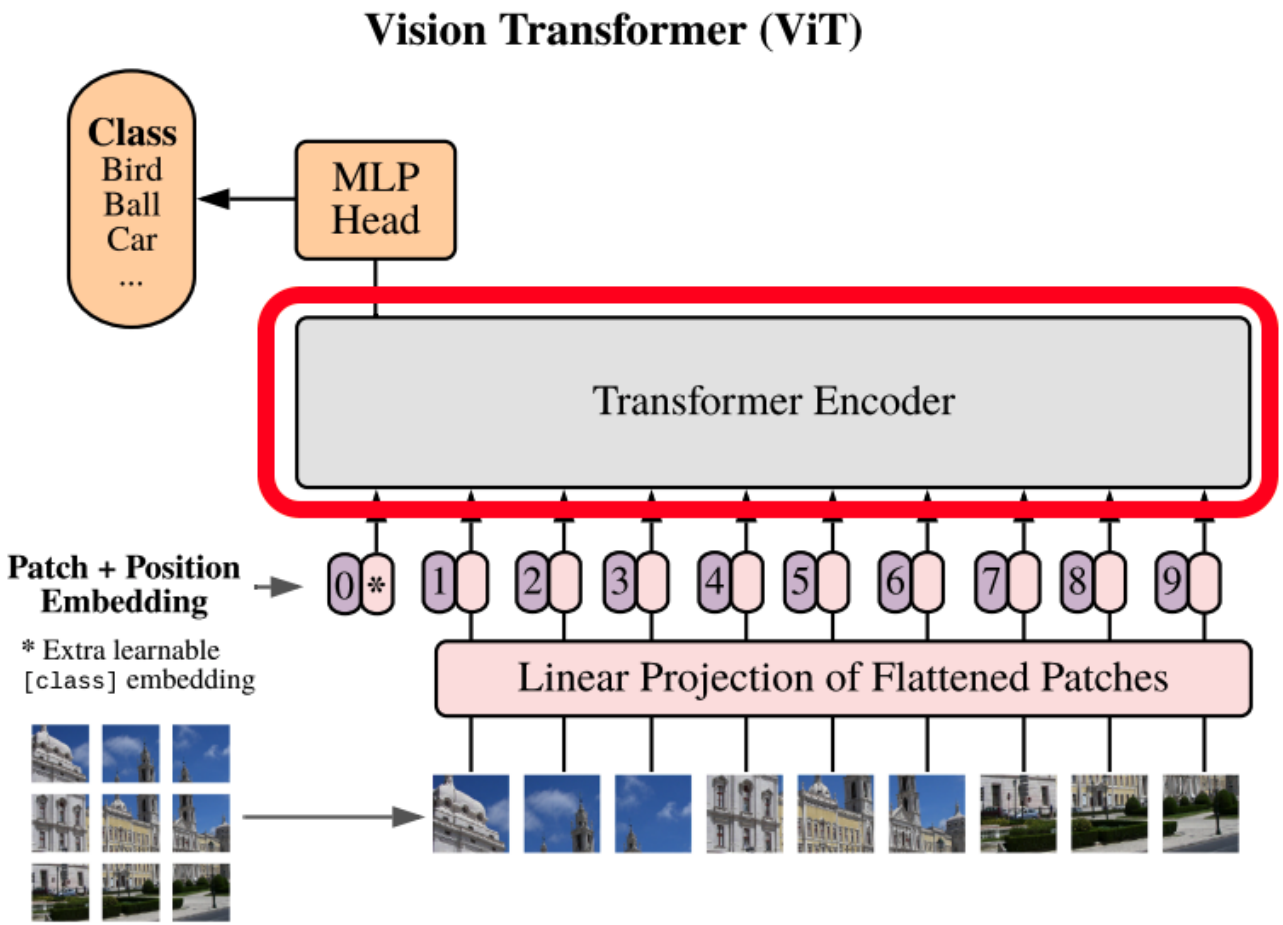


Linearly project to embedding space.

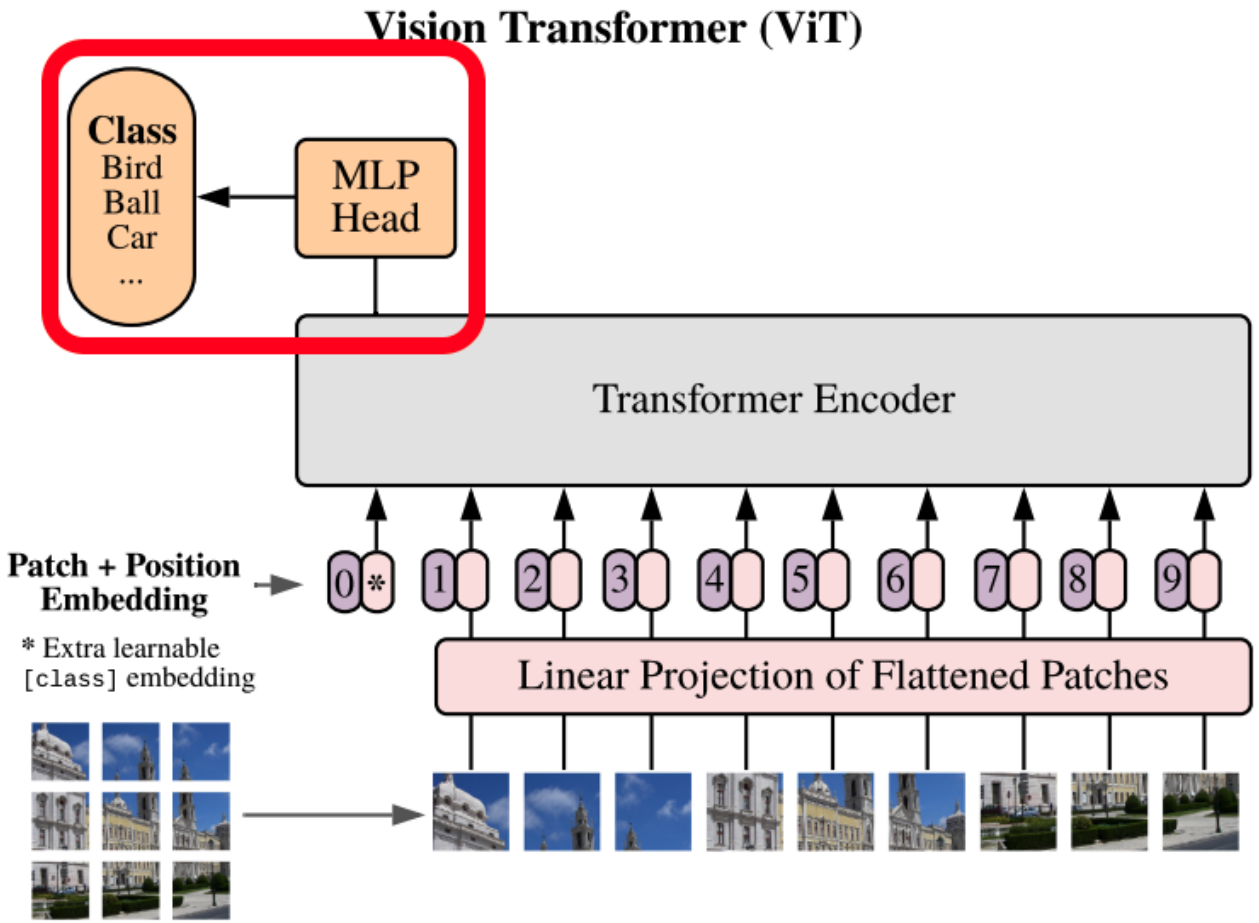


Vision Transformer Model

Based on the Transformer encoder.

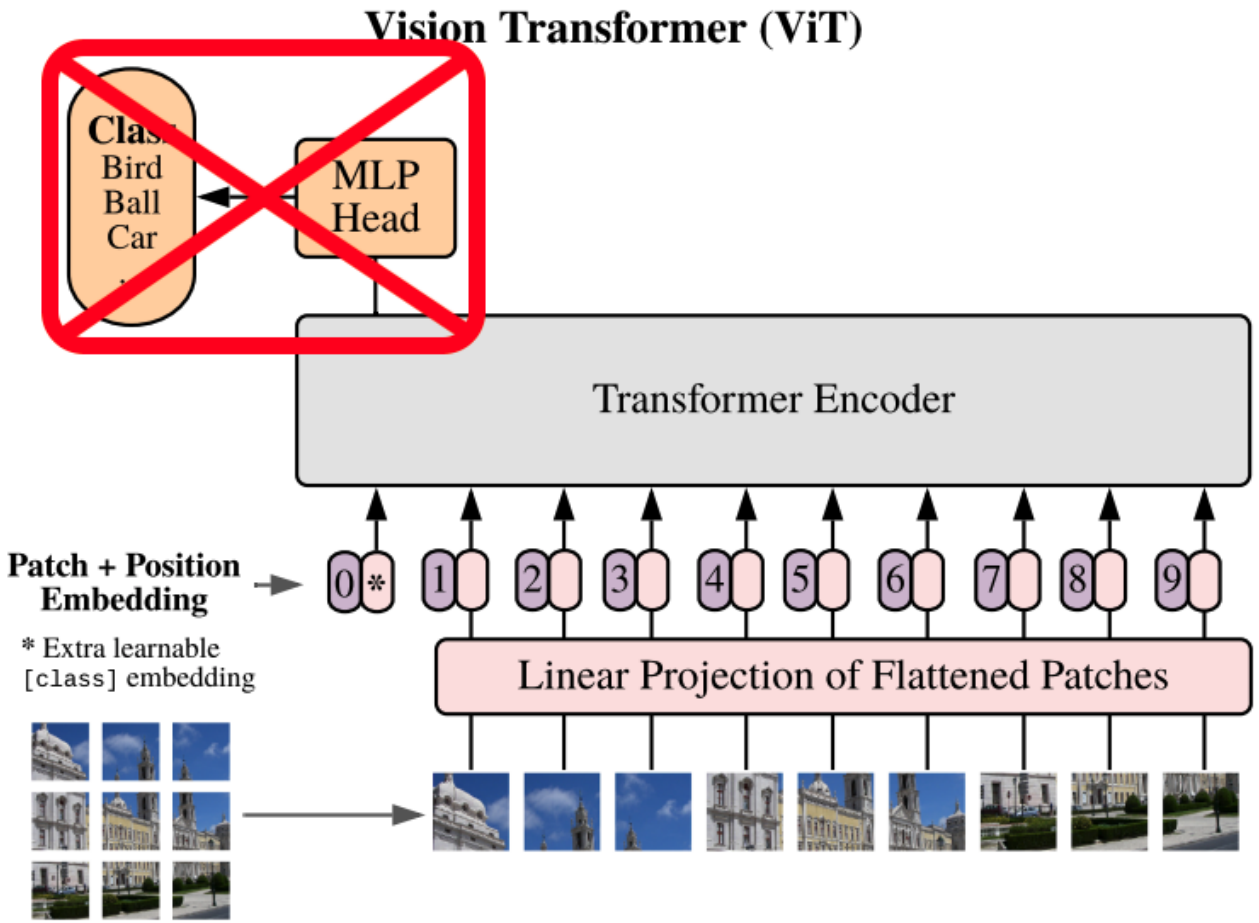


Classify using the special `[class]` token.



Vision Transformer Model

In clip we will not use it for classification.



CNN vs Transformer in Vision

CNNs

- Locality
- Two-dimensional neighborhood structure
- Translation equivariance

vs

ViT

- Two-dimensional neighborhood structure: cutting the image into patches
- MLP layers: local and translationally equivariant.
- Self-attention layers are global.

How do we train it?

Motivations for CLIP

Vision datasets:

- Are labor intensive and costly to create.
- Test only a narrow set of visual concepts.

Standard vision models:

- Are good at one task and one task only.
- Require significant effort to adapt to a new task.
- Perform well on benchmarks have disappointingly poor performance on stress tests.

CLIP

Contrastive Language–Image Pre-training

Ideas:

- Train model using vast data from the web pairing images and text.
- Using text as more flexible label to address other tasks.
- Perform "zero-shot" task using natural language.
- Learn a representation that's connected to language.

WIT Dataset

WebImageText (WIT) Dataset

- 400 million image-text pairs
- Collected form a variety of sources on the Internet
- Enhance by searching for image-text pairs from a set of queries.
- Build query list with all words occurring at least 100 times in Wikipedia.
- Including up to 20,000 image-text pairs per query.

Training

Given a batch of N image-text pairs, CLIP is trained to predict which of the N $\times$ N possible image-text pairings across a batch actually occurred.

Contrastive objective:

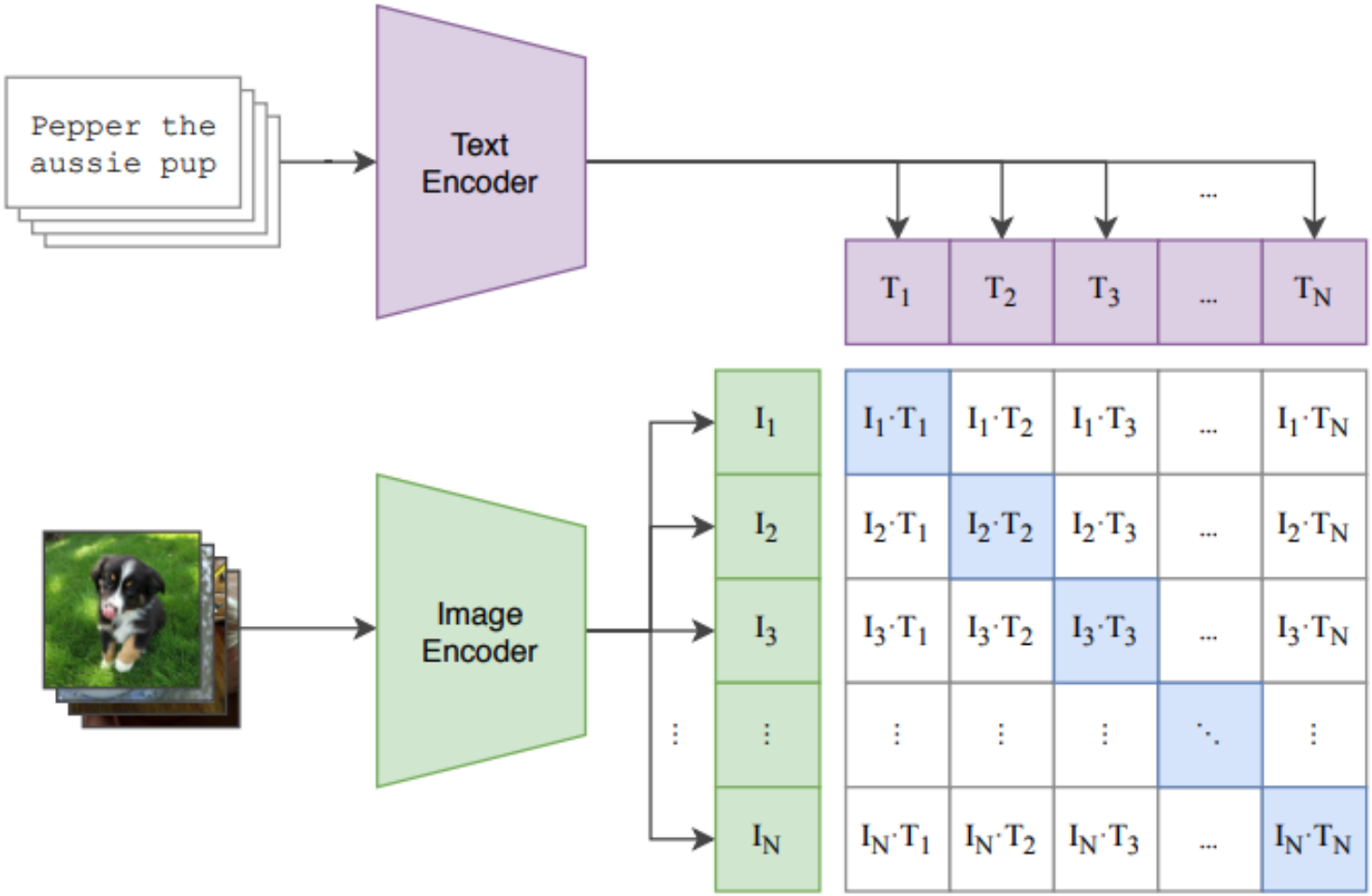
- Related items should be closer together in the embedding space than unrelated ones.

Why contrastive?

- Avoid trying to predict the exact words.
- More efficient.

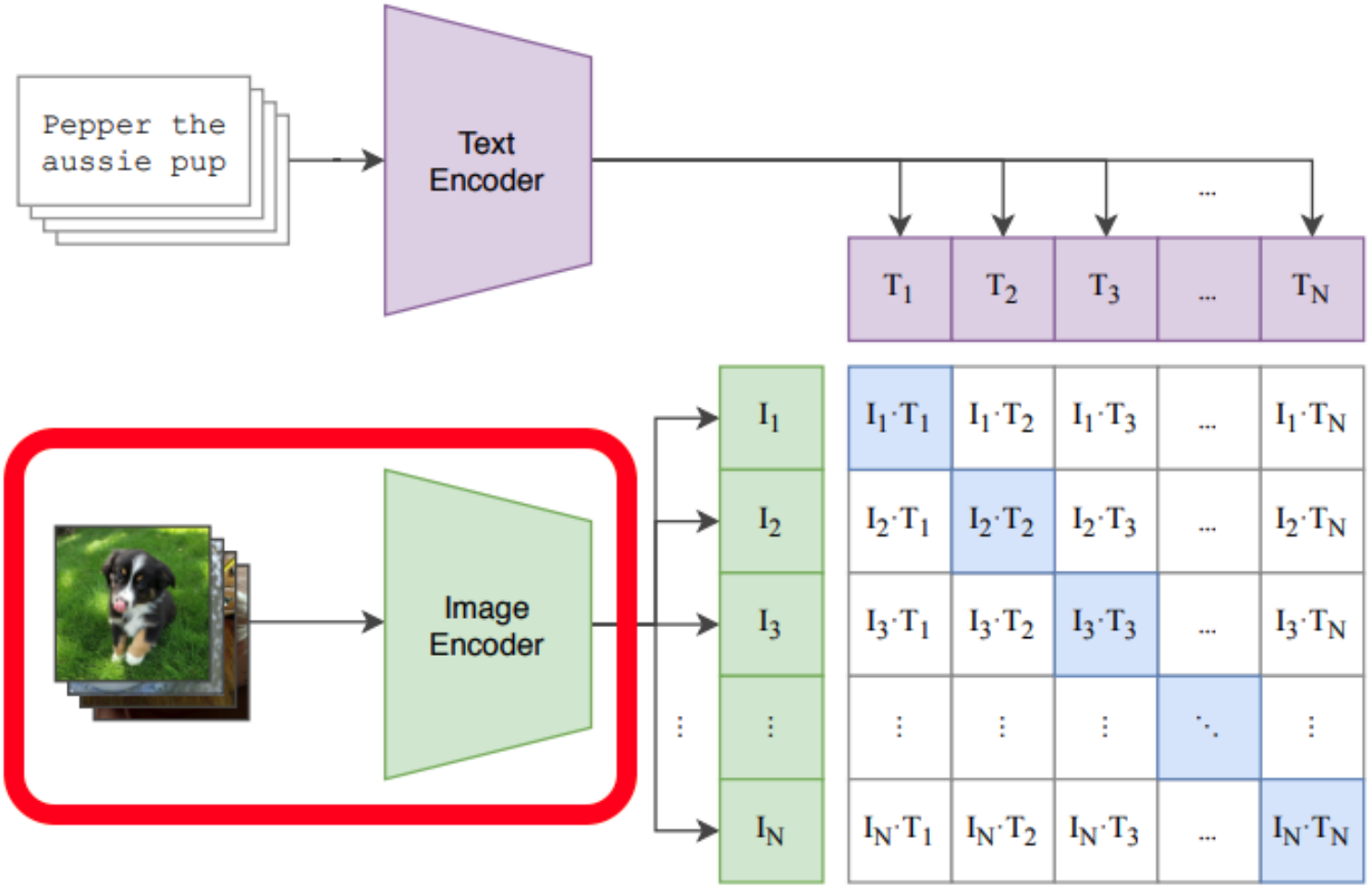
CLIP: Training

Overview.



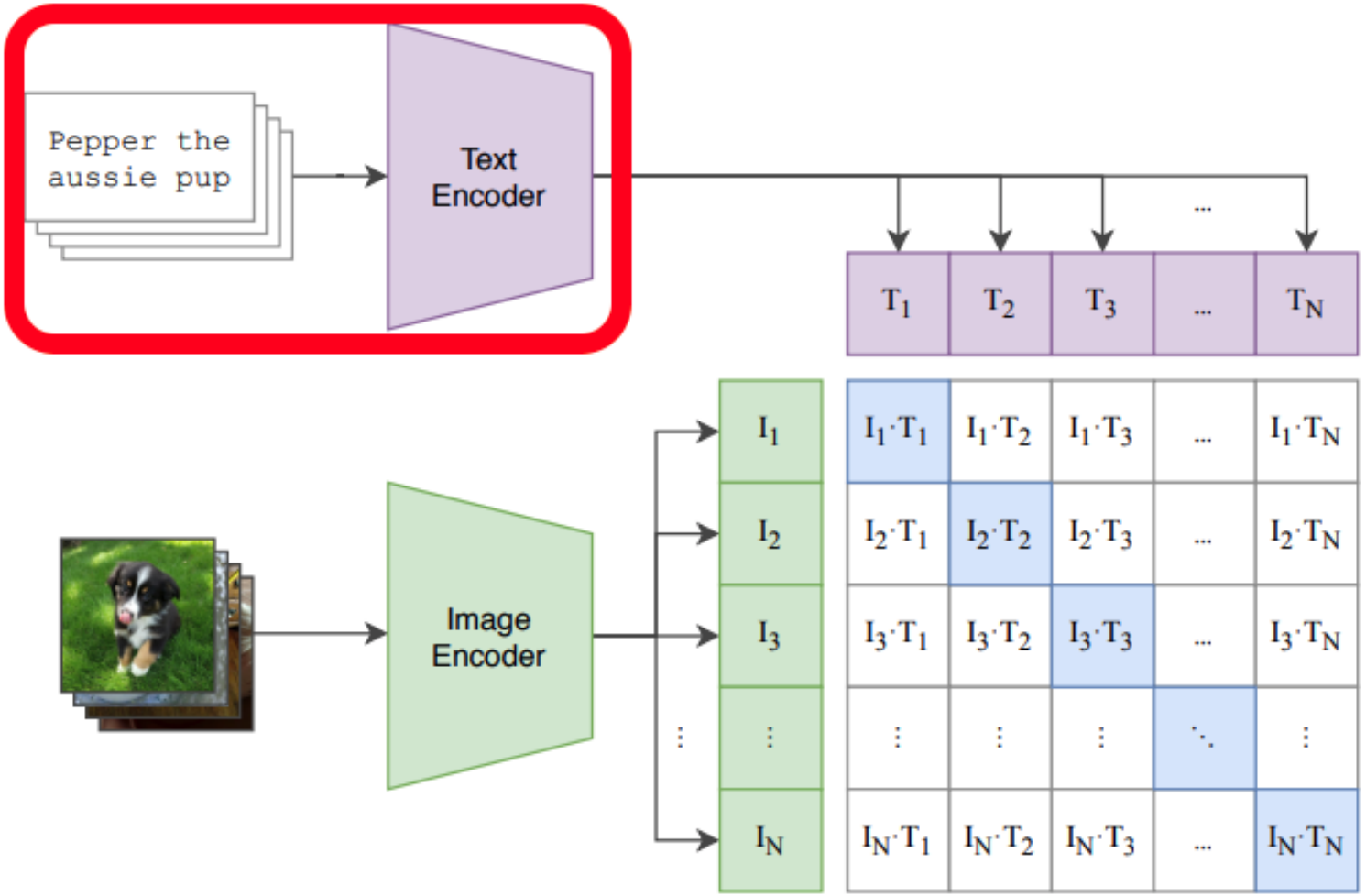
CLIP: Training

Encode image using Vision Transformer (ViT).



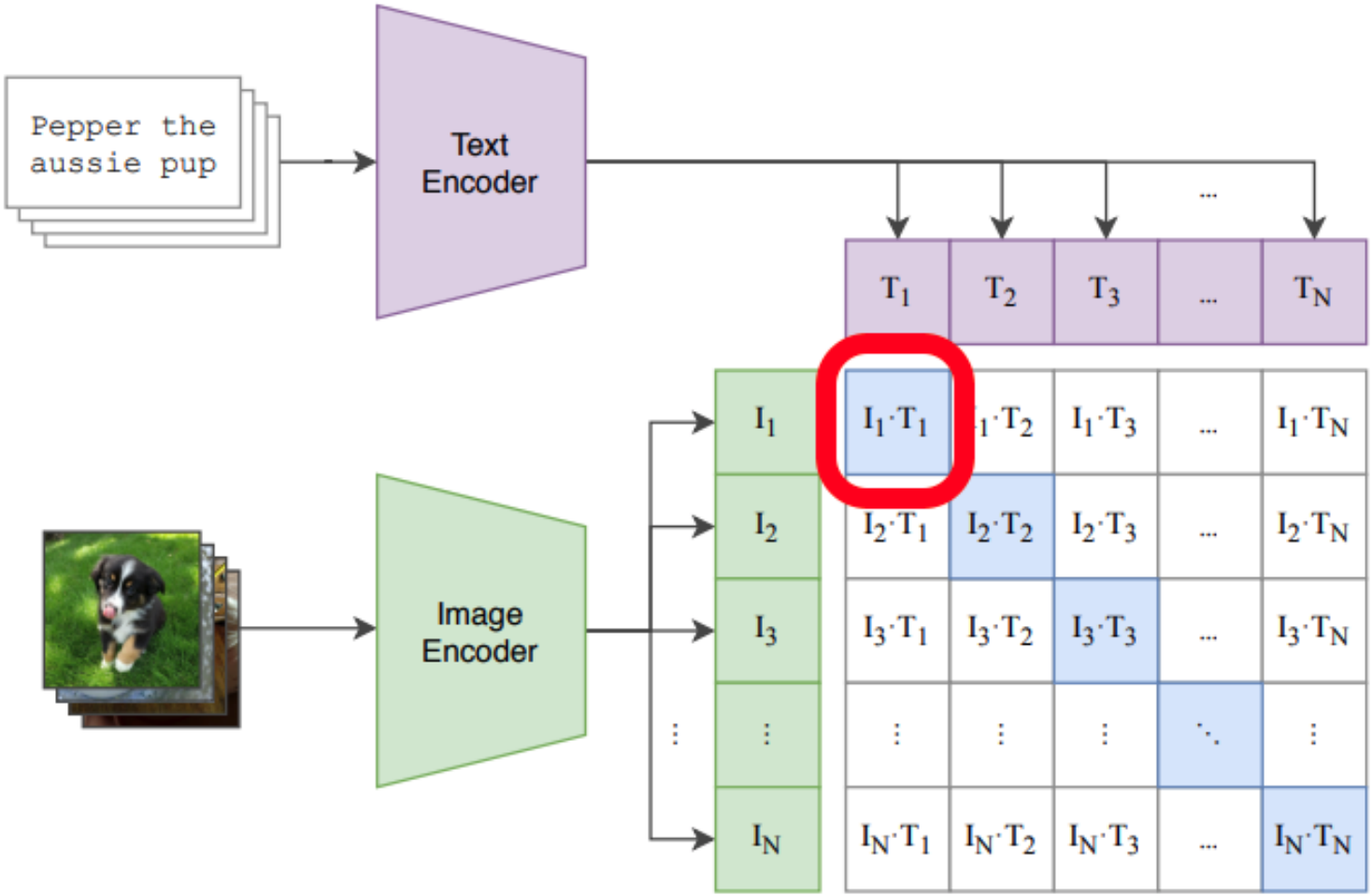
CLIP: Training

Encode text using Transformer encoder.



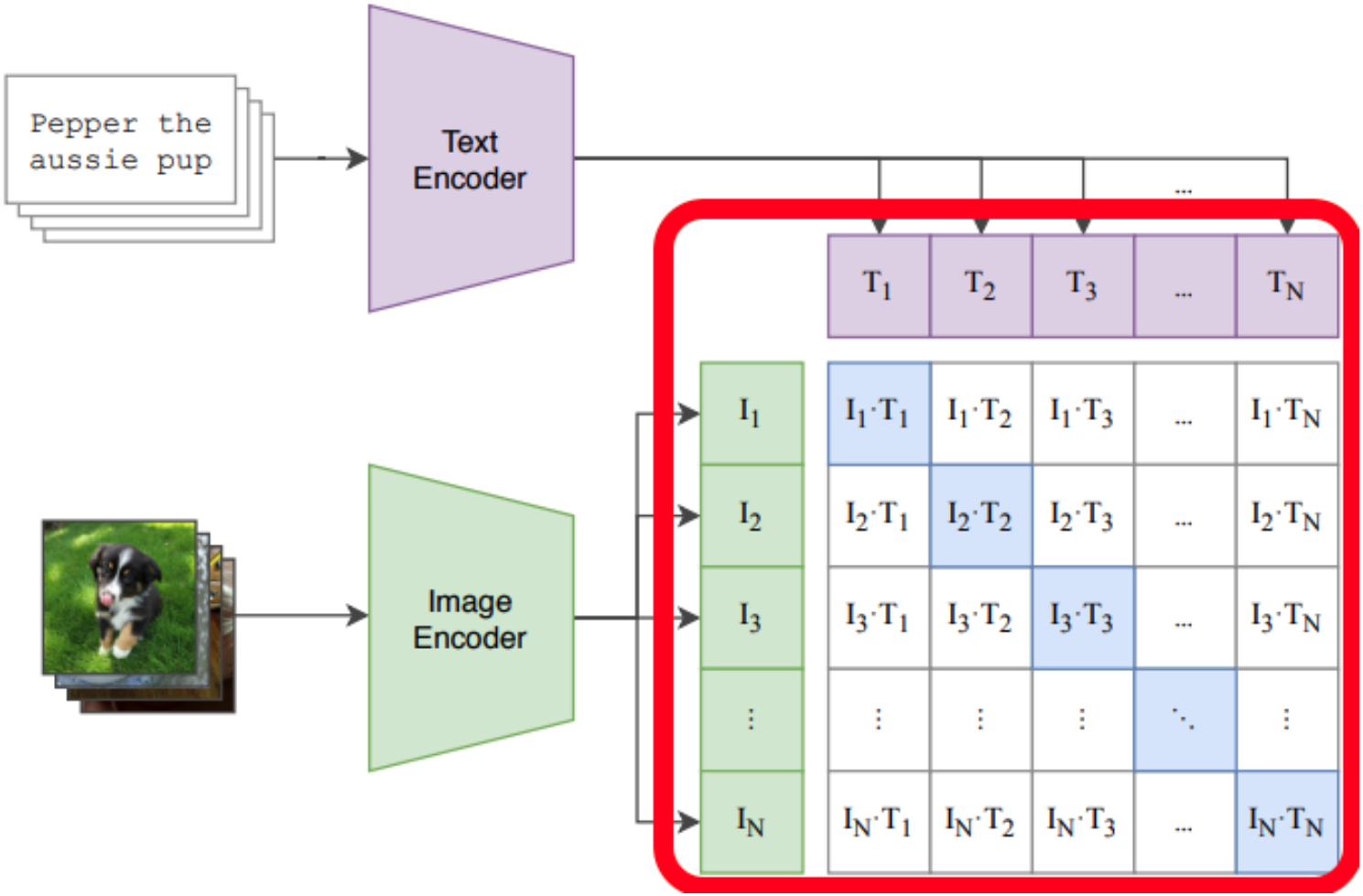
CLIP: Training

Cosine similarity between the embeddings.



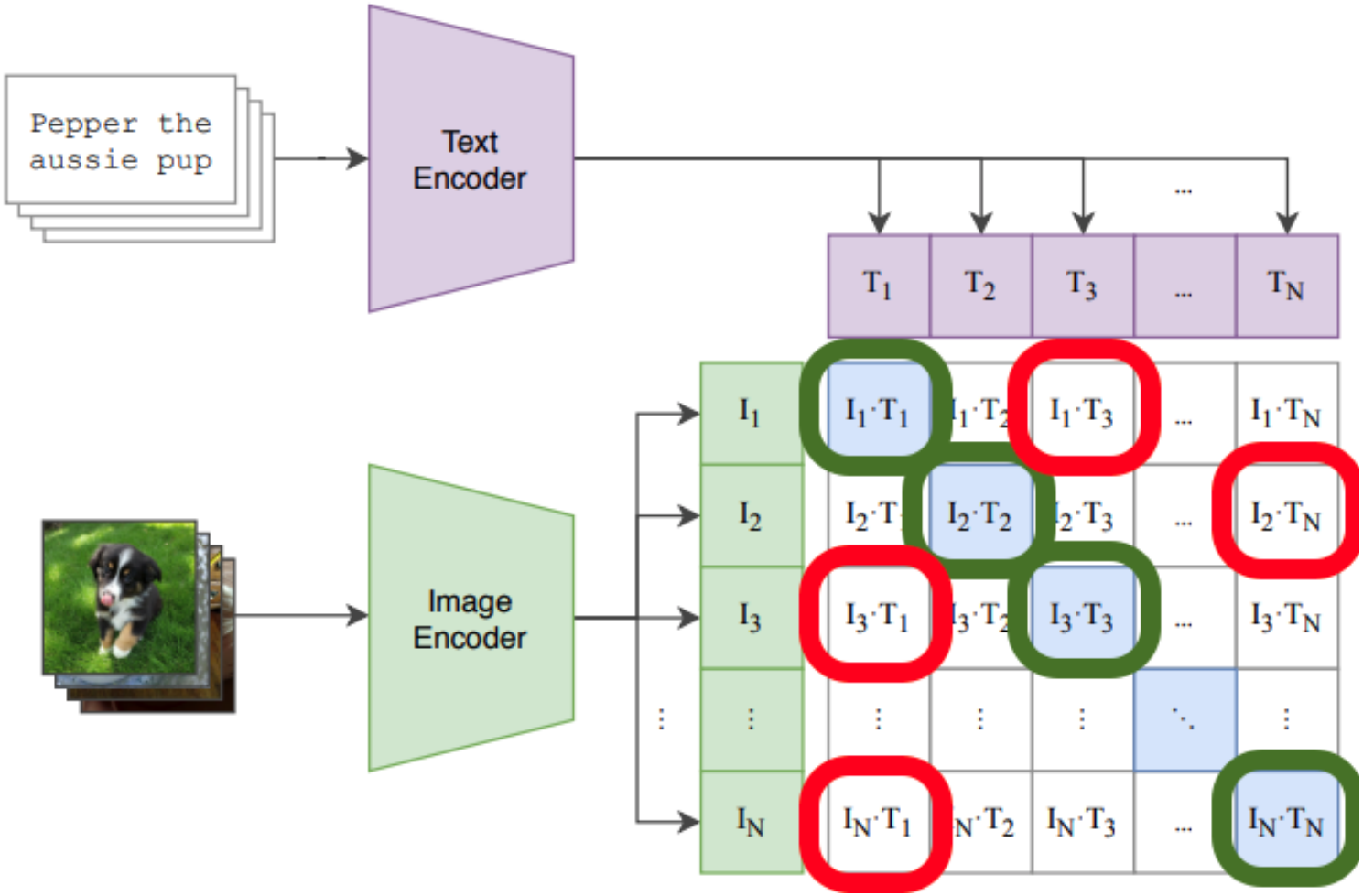
CLIP: Training

Use the whole batch for positive and negative examples.



CLIP: Training

Using cross-entropy. Positive examples get closer, negative further apart.

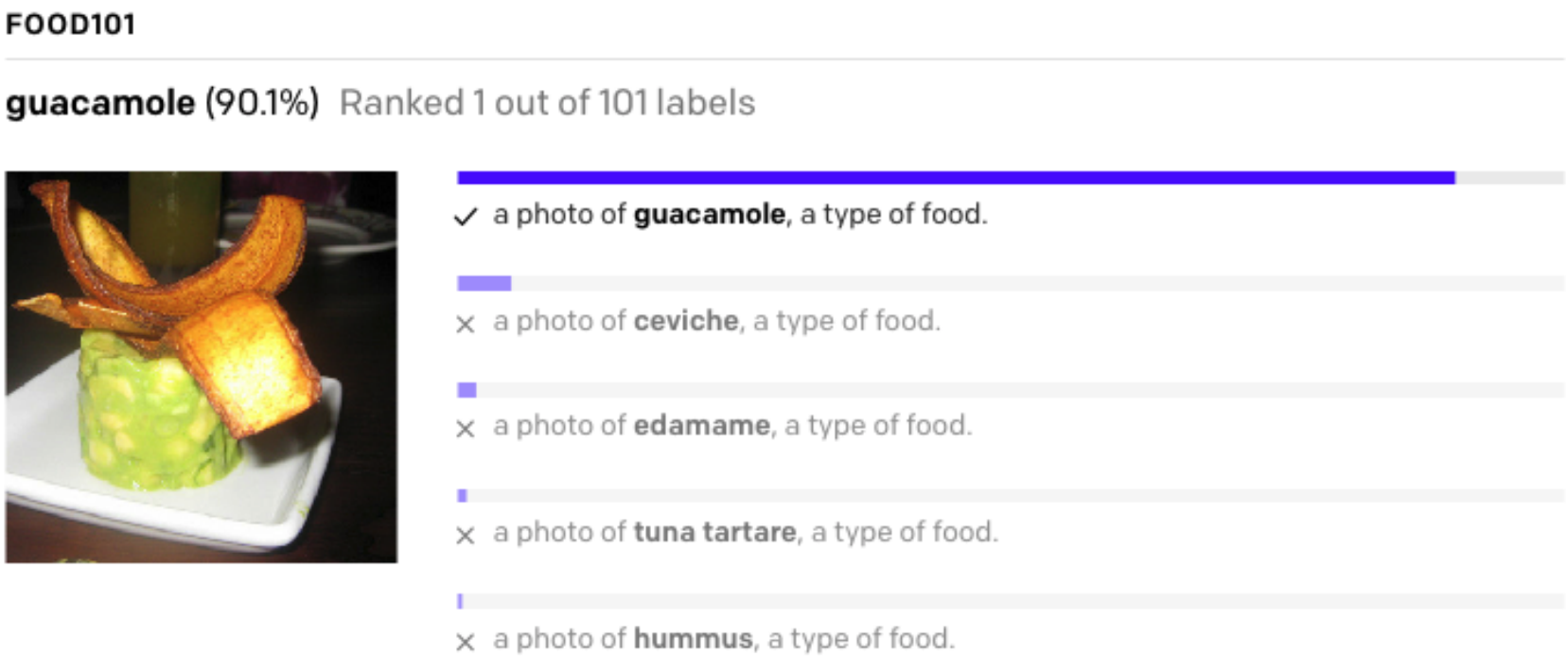


CLIP Usage

Clips compares a text and an image, giving a similarity score.

Zero-shot Image Classification

- Classify using query "A photo of a **{class}**".

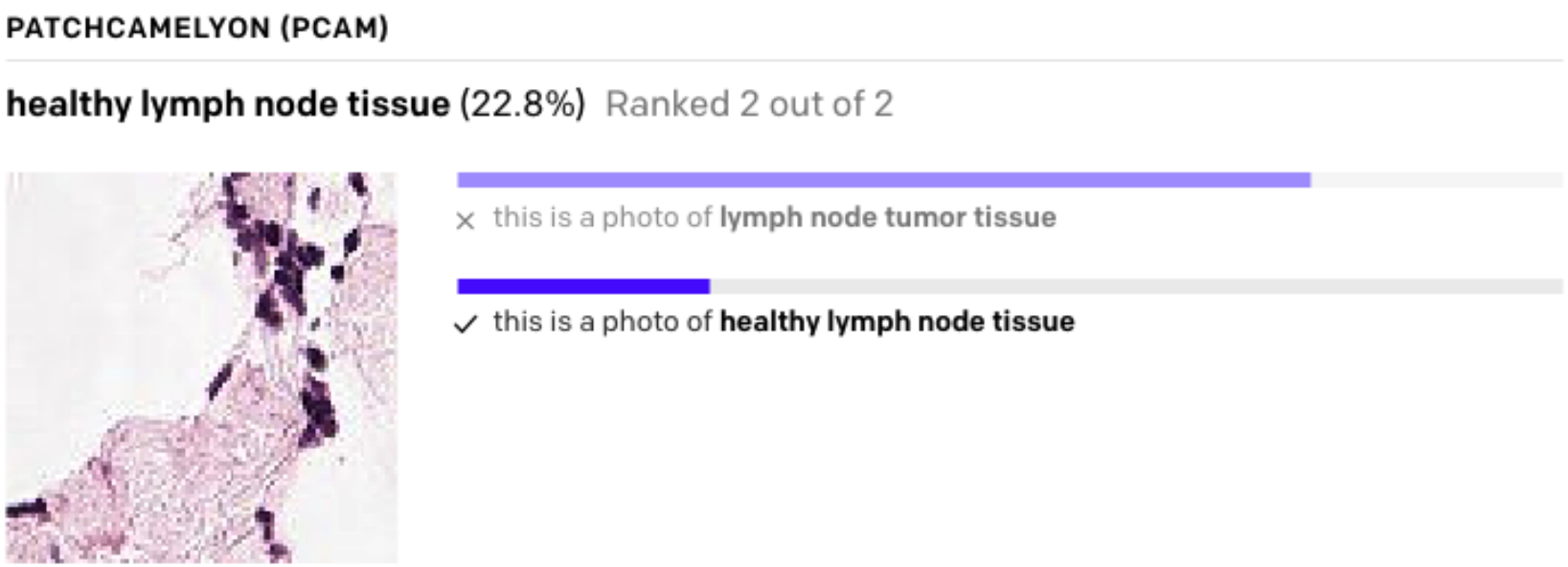


CLIP Usage

Clips compares a text and an image, giving a similarity score.

Zero-shot Image Classification

- Classify using query "A photo of a **{class}**".



Final words

Good Resources

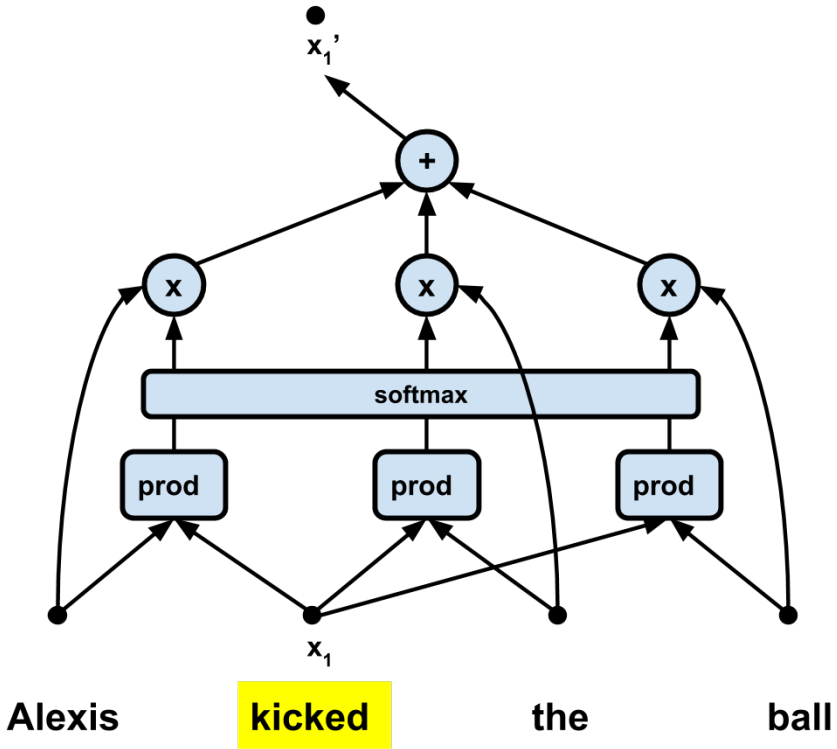
- BETO's repository.
- Harvard's Annotated Transformer.
- Stanford's NLP course session about Transformers.
- Original Transformer paper.
- Original BERT paper.
- *Analysis of BETO's attention patterns (Bertology).
- CLIP's repository.

Questions?

Appendix

Why normalize?

When d_k grows, the dot product grows too pushing the softmax to small gradient regions.



Transformers for NLP: BERT

Motivations for BERT

Replicate the success CNN have had in lot's of task just by pre-training on Imagenet.

Model's architecture must support a large amount of tasks.

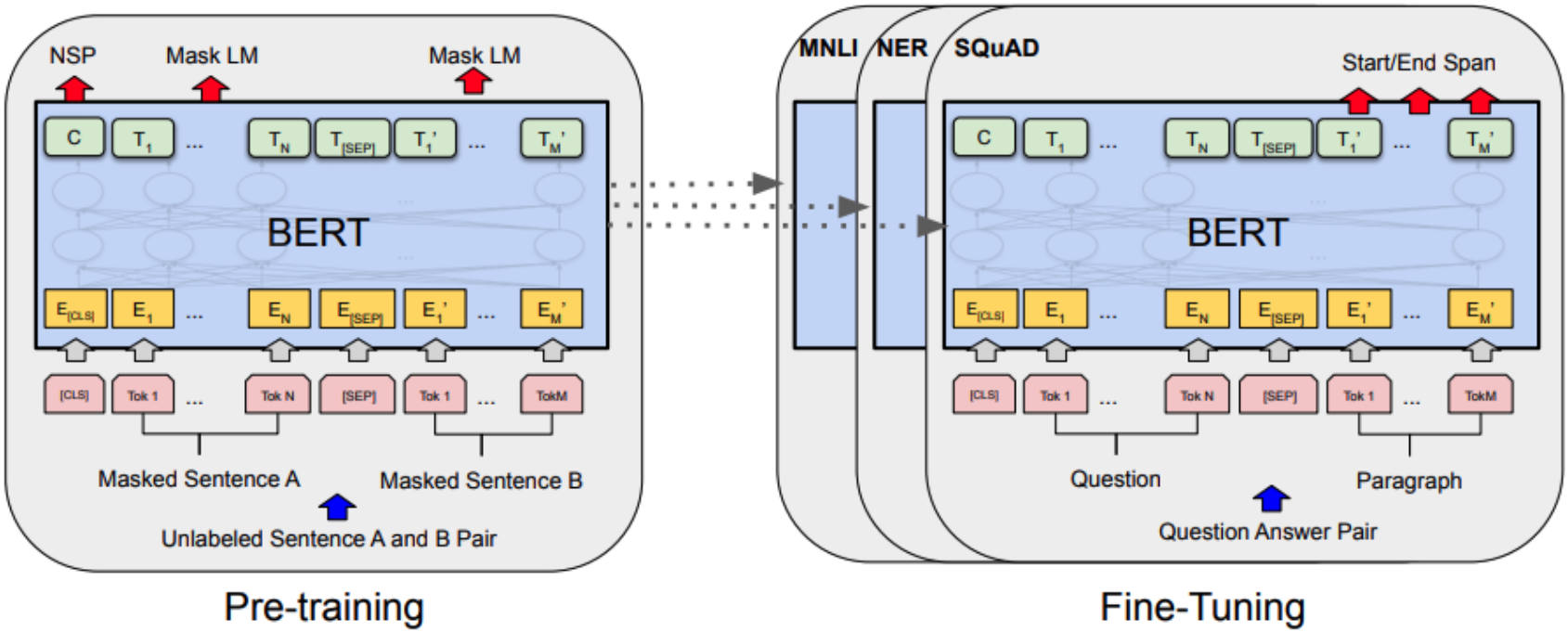
- Token classification tasks.
- Sentence classification tasks.
- Single sentence input.
- Pair of sentence input.

BERT

"BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding" by Devlin et al. 2018.

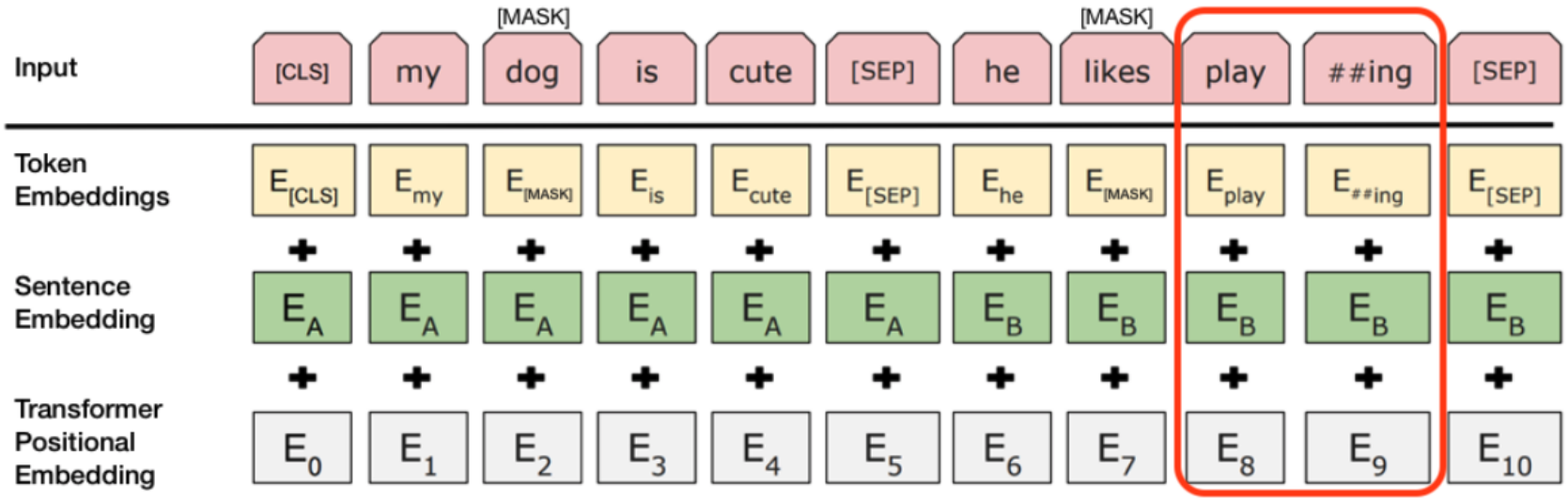
Multi-layer bidirectional Transformer encoder.

Unified architecture across tasks.



BERT Input Representation

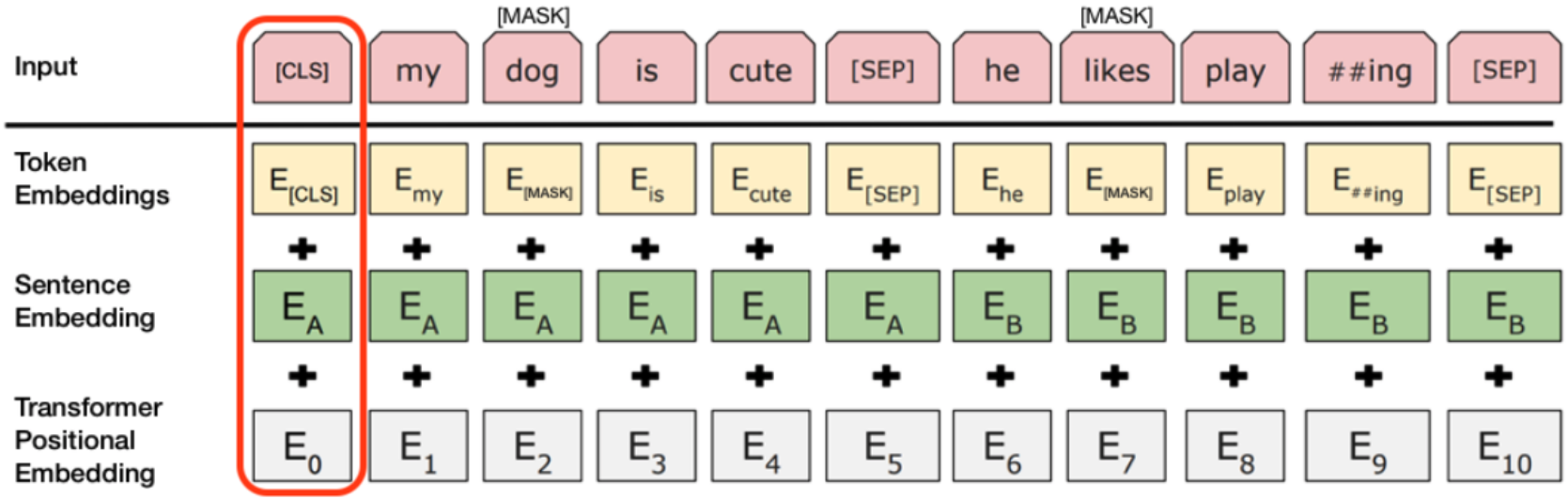
Word piece embeddings.



BERT Input Representation

First token: special classification token `[CLS]`

Used as the aggregate sequence representation for classification tasks.

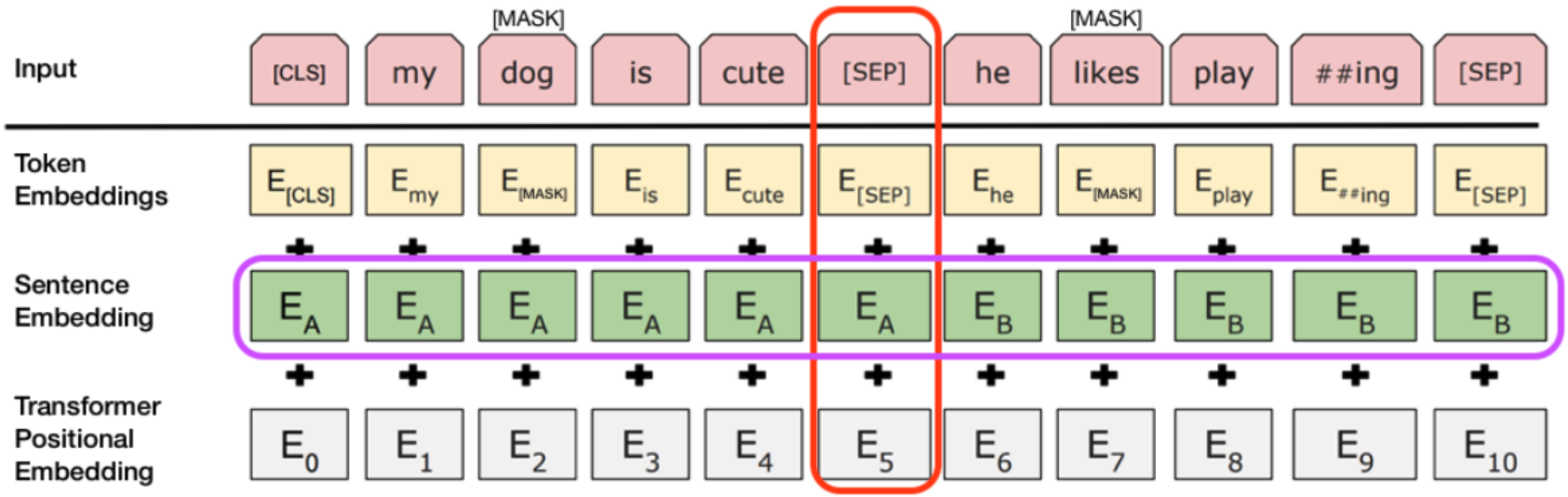


BERT Input Representation

Unambiguously represent single and pair of sentences.

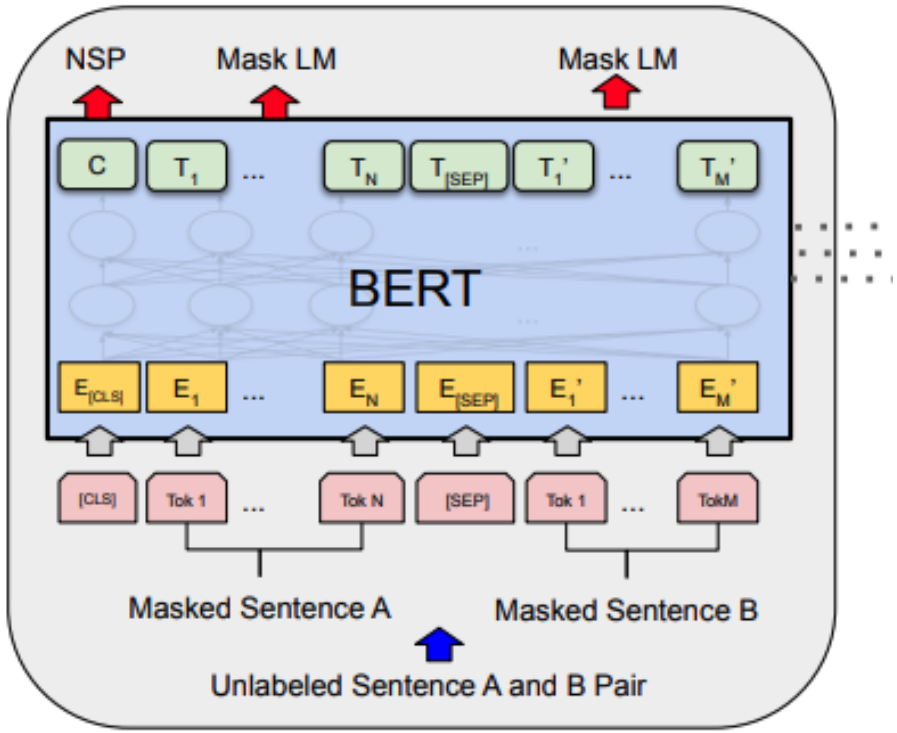
Differentiate the sentences in two ways.

1. Separate them with a special token, `[SEP]`.
2. Add a learned embedding indicating which sentence it belongs to.



Masked language model.

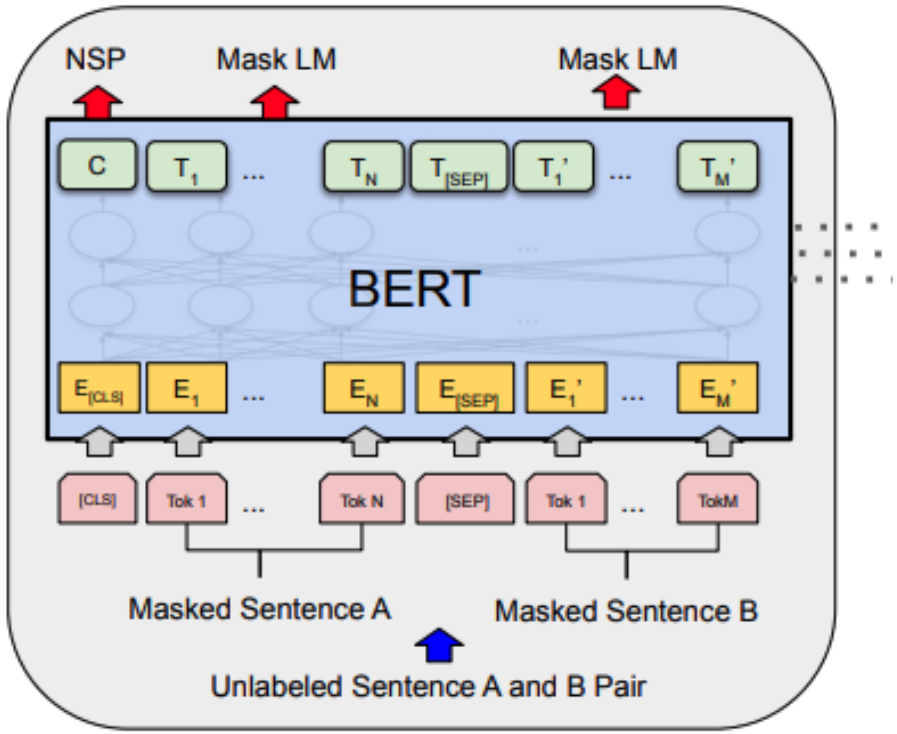
- Mask 15% of all tokens at random to predict that place token.
- Only 80% of the time use the `[MASK]` token.
- 10% with a random token.
- 10% without replacement.



Pre-training

Next sentence prediction.

- Binary task of weather sentence B follows sentence A.
- 50% of times is and 50% isn't.
- Classified using the `[CLS]` special token.

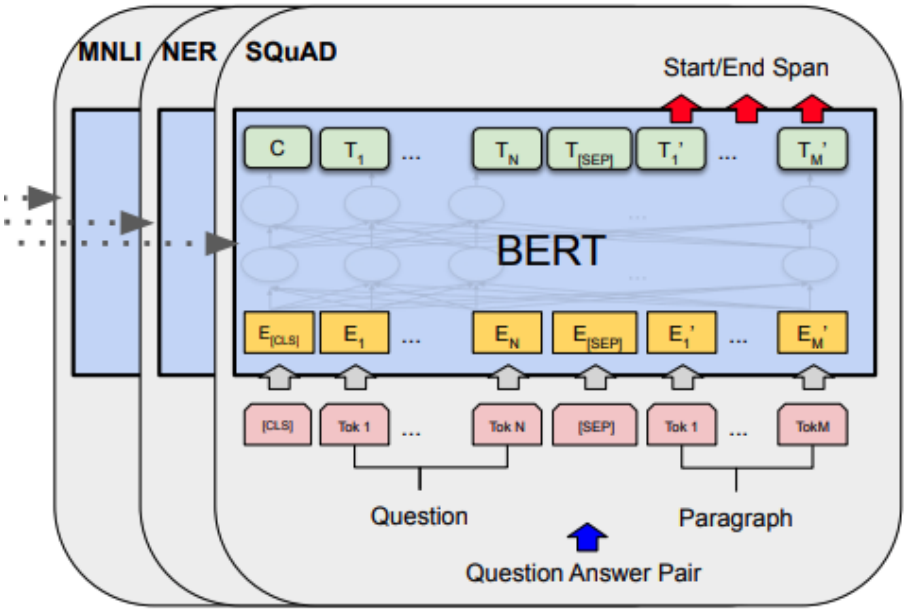


Pre-training

Input

Sentences A and B are used as follows in the following tasks.

- Sentence pairs in paraphrasing.
- Hypothesis-premise pairs in entailment.
- Question-passage pairs in question answering.
- No text pair for text classification or sequence tagging.

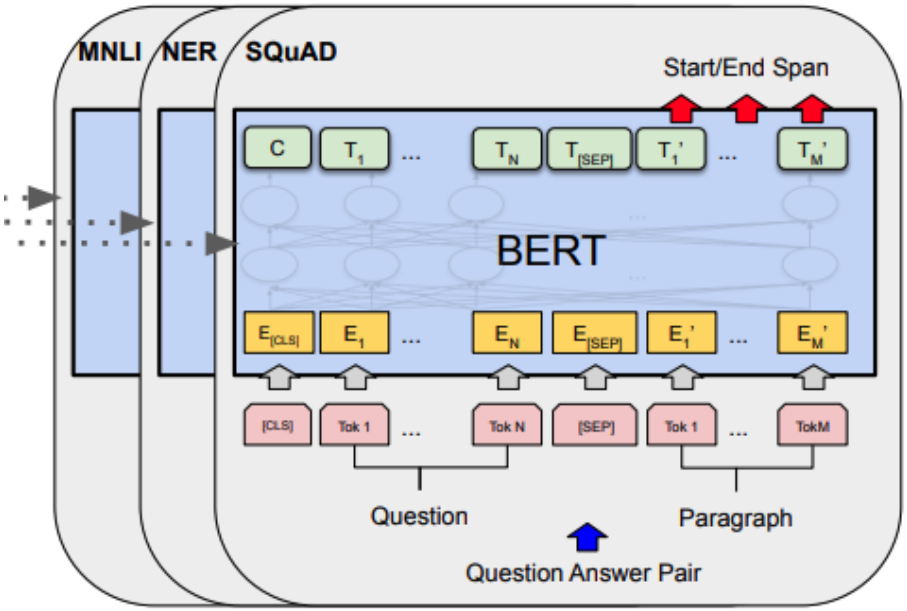


Fine-Tuning

Output

The token representations are fed into an output layer for token level tasks, i.e. sequence tagging for question answering.

The [CLS] final representation is fed into an output layer for classification.



Fine-Tuning

CLIP

CLIP: Training

Pseudo-código de Entrenamiento

```
# image_encoder - ResNet or Vision Transformer
# text_encoder  - CBOW or Text Transformer
# I[n, h, w, c] - minibatch of aligned images
# T[n, l]       - minibatch of aligned texts
# W_i[d_i, d_e] - learned proj of image to embed
# W_t[d_t, d_e] - learned proj of text to embed
# t             - learned temperature parameter

# extract feature representations of each modality
I_f = image_encoder(I) #[n, d_i]
T_f = text_encoder(T)  #[n, d_t]

# joint multimodal embedding [n, d_e]
I_e = l2_normalize(np.dot(I_f, W_i), axis=1)
T_e = l2_normalize(np.dot(T_f, W_t), axis=1)

# scaled pairwise cosine similarities [n, n]
logits = np.dot(I_e, T_e.T) * np.exp(t)

# symmetric loss function
labels = np.arange(n)
loss_i = cross_entropy_loss(logits, labels, axis=0)
loss_t = cross_entropy_loss(logits, labels, axis=1)
loss   = (loss_i + loss_t)/2
```

Relation Networks

A simple neural network module for relational reasoning

Adam Santoro*, David Raposo*, David G.T. Barrett, Mateusz Malinowski,
Razvan Pascanu, Peter Battaglia, Timothy Lillicrap

adamsantoro@, draposo@, barrettdavid@, mateuszm@,
razp@, peterbattaglia@, countzero@google.com

DeepMind
London, United Kingdom

Relation Networks

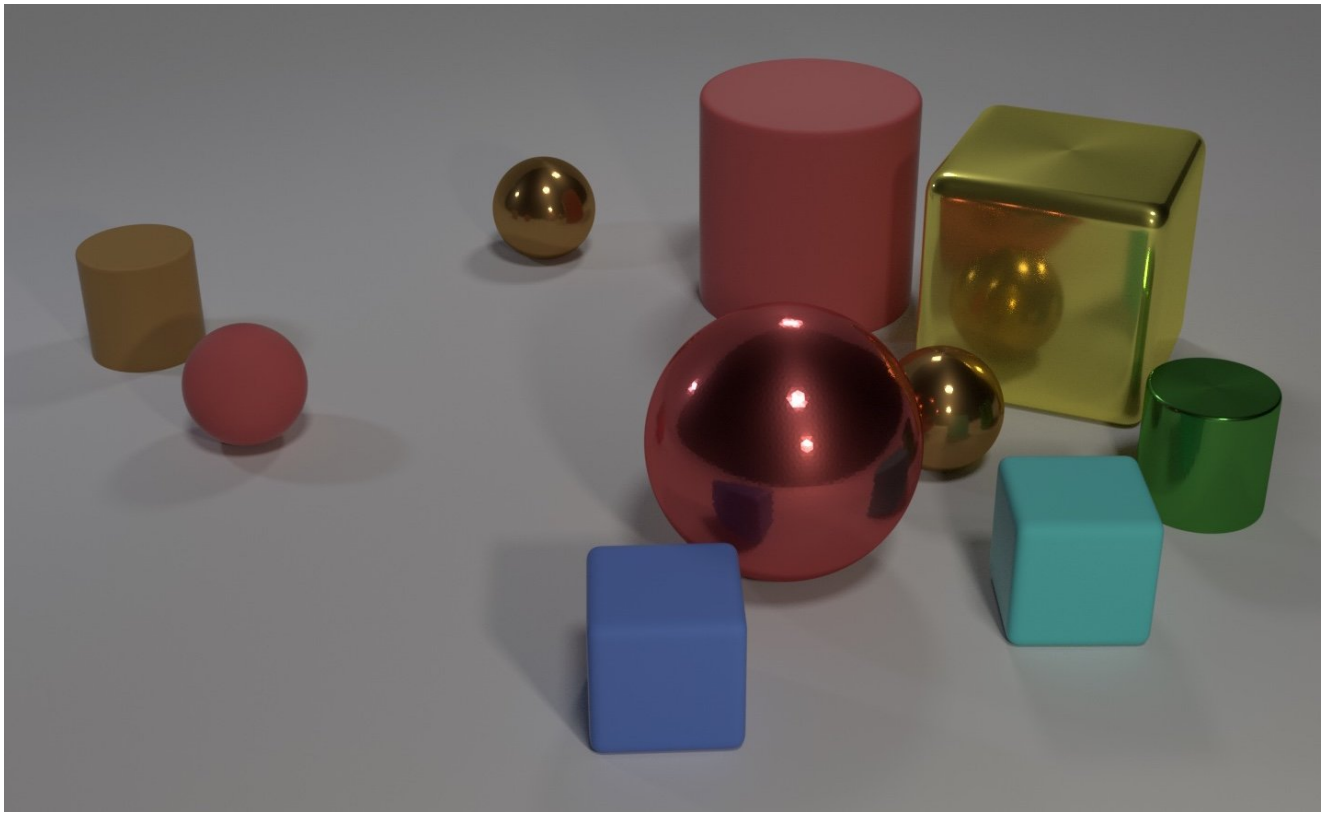
"RNs are simple, plug-and-play, and are exclusively focused on flexible relational reasoning".

$$RN(O) = f_\phi\left(\sum_{i,j} g_\theta(o_i, o_j)\right)$$

Learn functions g_θ and f_ϕ

- g_θ encodes the relationship between the entities $o_i, o_j \in O$.
- f_ϕ aggregates all relationships.

Clevr Task



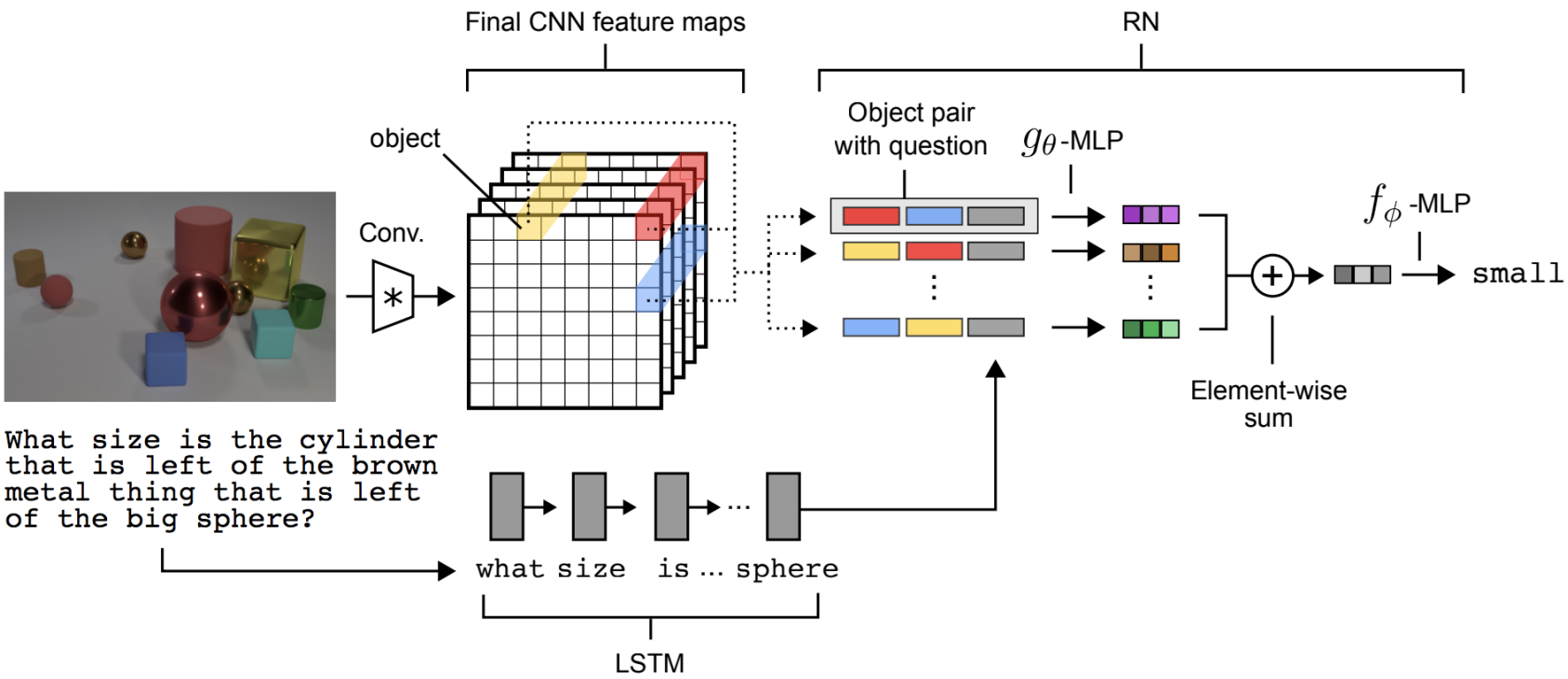
What size is the cylinder that is left of the brown metal thing that is left of the big sphere?

Answer: small

Model for the CLEVR Task

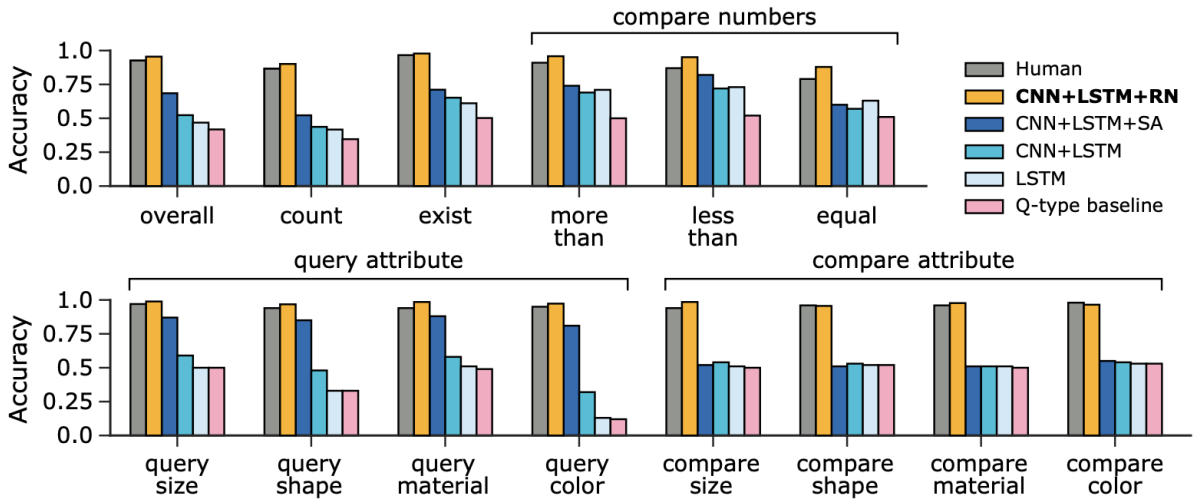
Pixel input & question embedding

$$RN(O) = f_\phi\left(\sum_{i,j} g_\theta(o_i, o_j, \mathbf{q})\right)$$



Results on CLEVR

Results on the CLEVR task.



bAbi Task

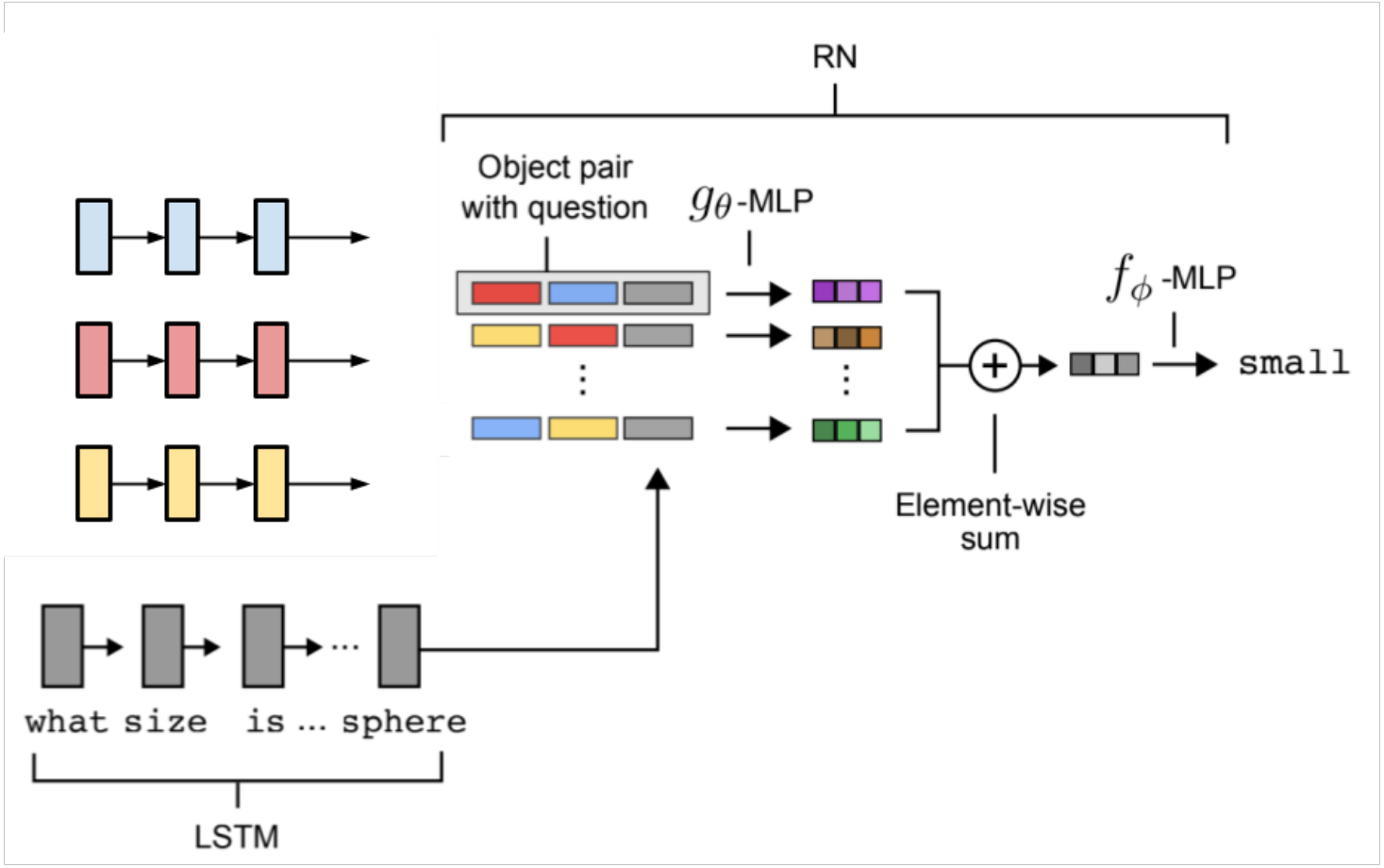
```
1 Mary moved to the bathroom.
2 John went to the hallway.
3 Where is Mary?  bathroom 1
4 Daniel went back to the hallway.
5 Sandra moved to the garden.
6 Where is Daniel?  hallway 4
7 John moved to the office.
8 Sandra journeyed to the bathroom.
9 Where is Daniel?  hallway 4
10 Mary moved to the hallway.
11 Daniel travelled to the office.
12 Where is Daniel?  office 11
13 John went back to the garden.
14 John moved to the bedroom.
15 Where is Sandra?  bathroom 8
```

Model for bAbi Task

Each sentence is considered an object.

They have their position tagged.

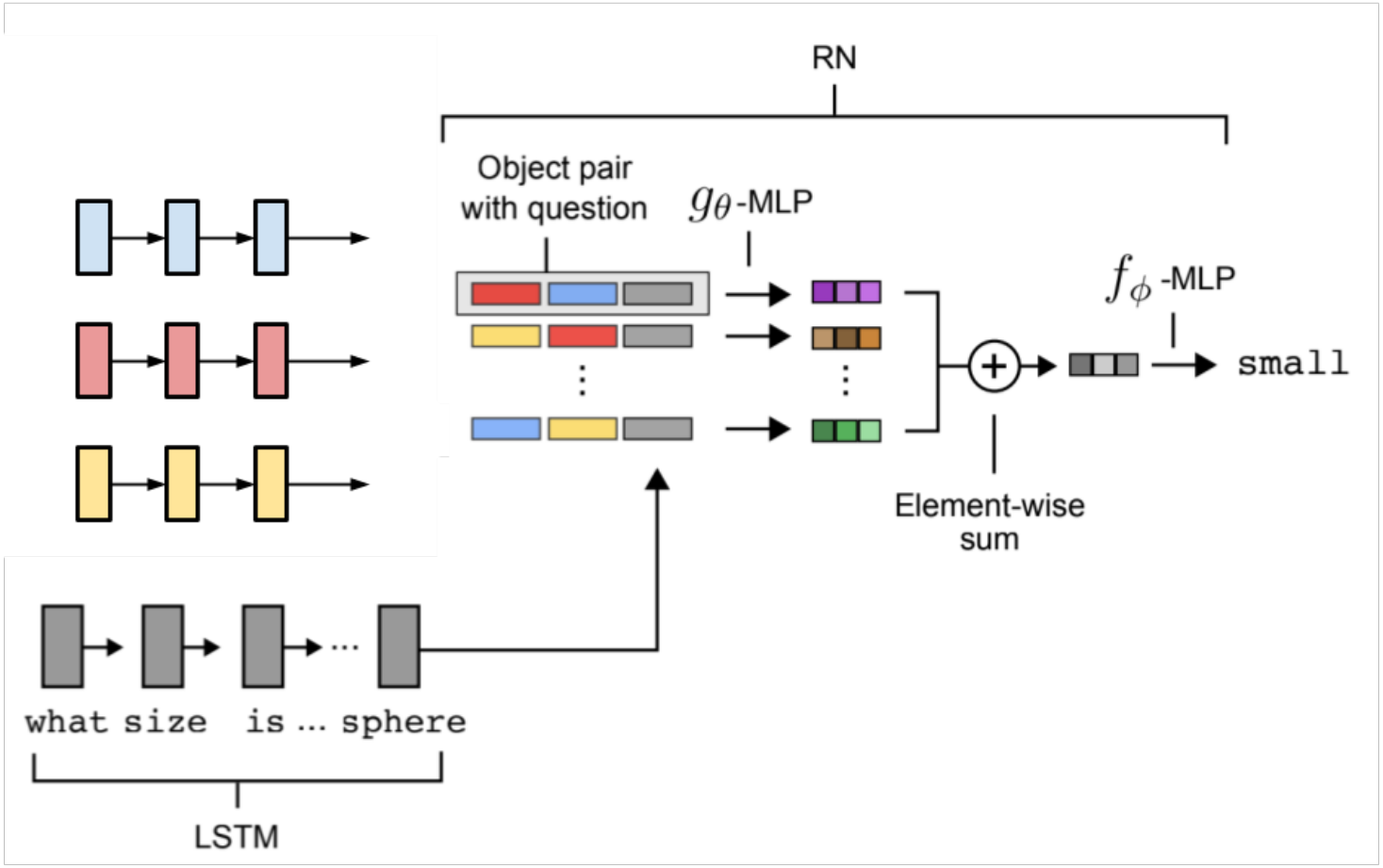
The input to the RN module is the last hidden state of the encoding LSTM.



Model for bAbi Task

Results. Passed 18/20 tasks.

Other such as Memory Networks pass 14/20, DNC 18/20, Sparse DNC 19/20, and EntNet 16/20.



Benefits

- Flexible.
- Learn to infer relationships.
- Data efficient.
- Invariant to object order (operates over the object set).