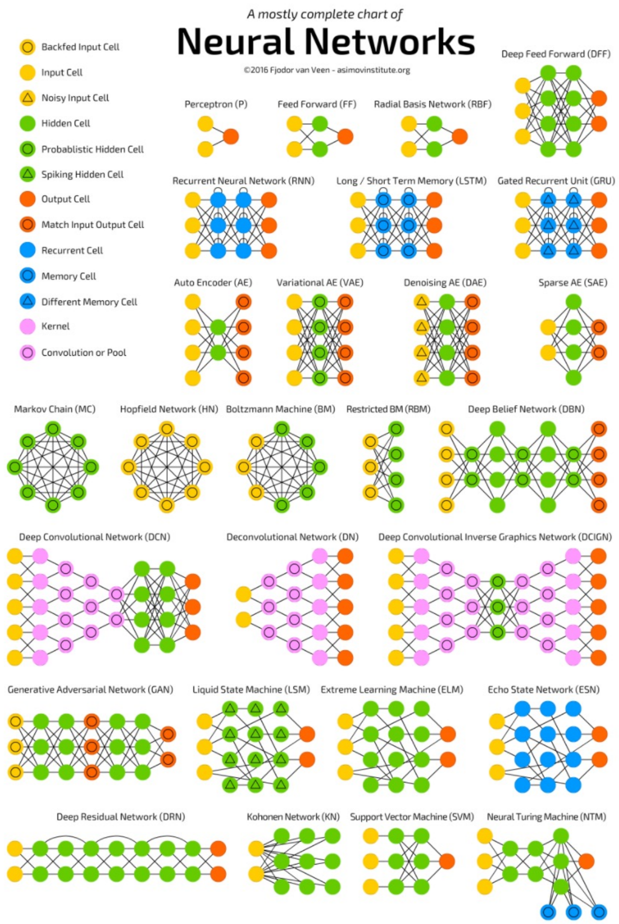
Transformers

CSCI 4360/6360 Data Science II

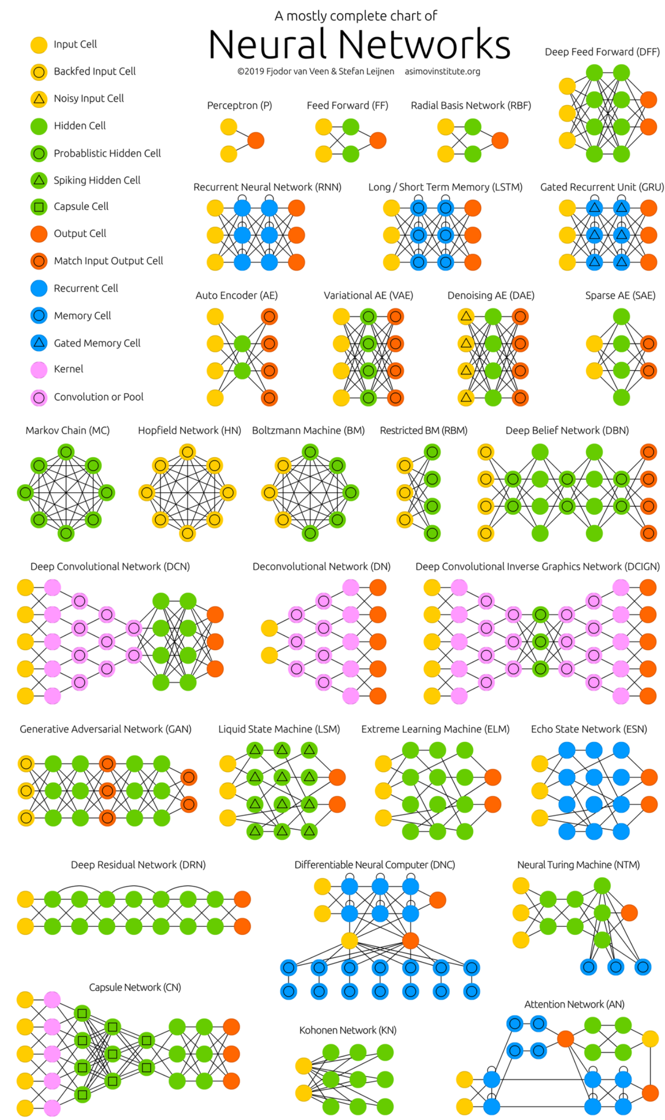
The Neural Network Zoo

- http://www.asimovinstitute.org/neural-network-zoo/



The Neural Network Zoo

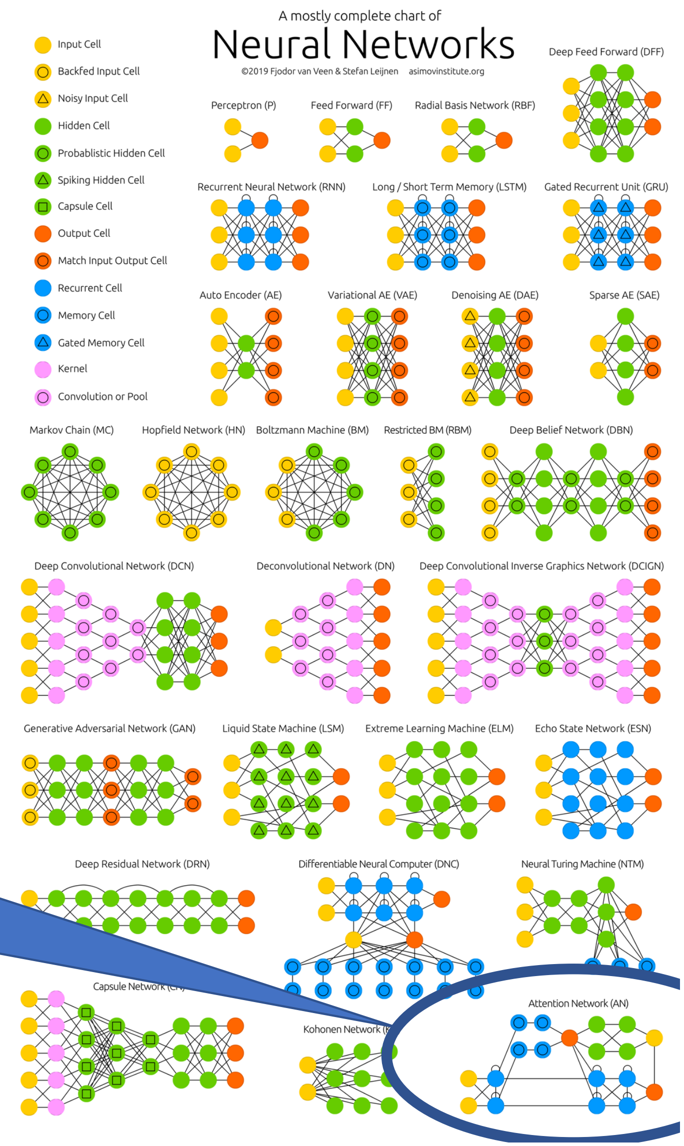
- http://www.asimovinstitute.org/neural-network-zoo/



The Neural Network Zoo

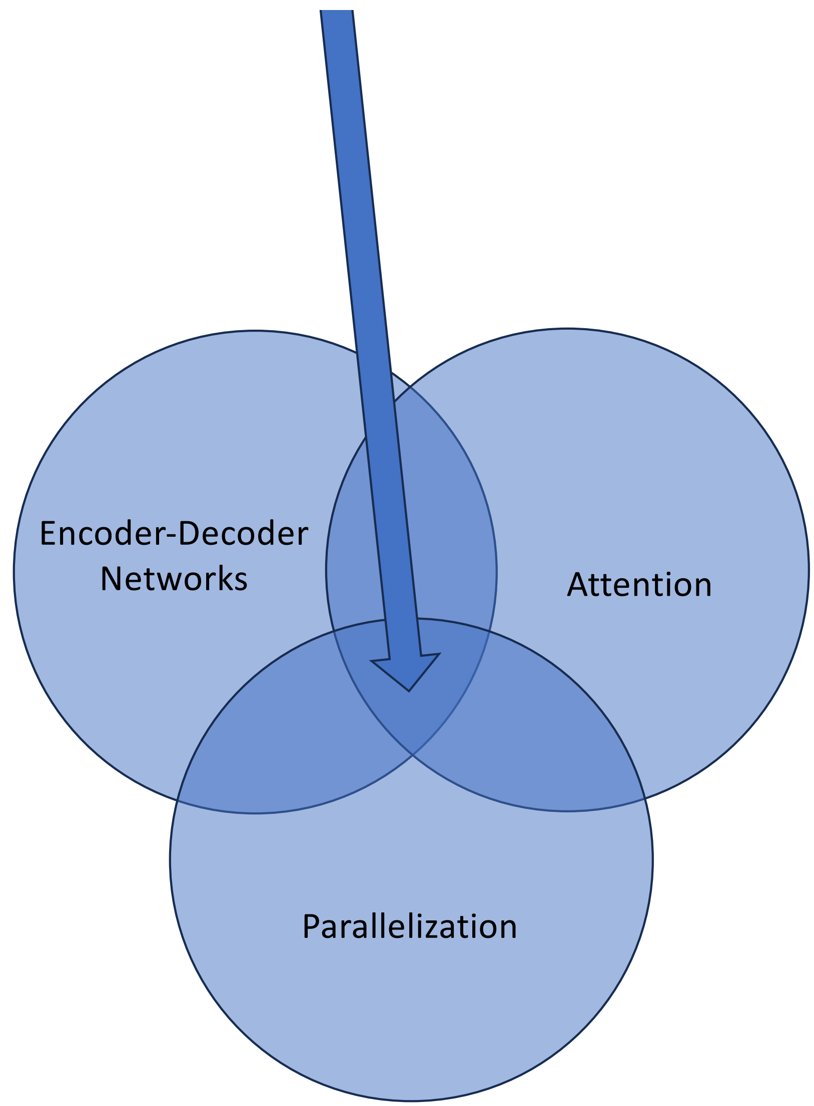
- http://www.asimovinstitute.org/neural-network-zoo/

Today



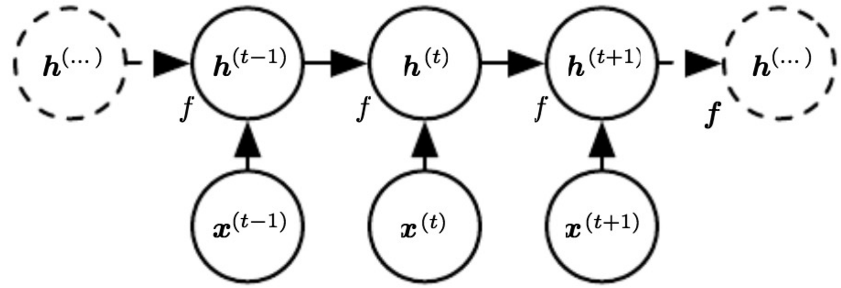
Transformers

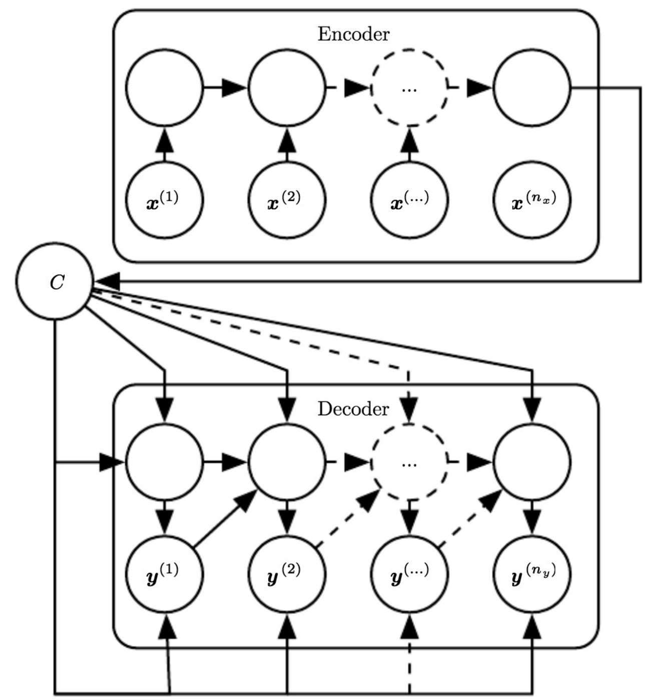
- A confluence of multiple technologies and theories



RNNs revisited

- Recurrences
  - Input + hidden state
  - LSTM “forget” gate
  - Encoder-decoder
- Sequential architecture precludes parallelization
- Lot of information crammed into h_t
- “Catastrophic forgetting”





Transformers

- Transductive model
- Relies entirely on self-attention mechanisms
- No sequence-aligned RNNs or convolutions

Transformer architecture

- Encoder
  - Maps input sequence ($x_1,\ldots,x_n$) to a representation sequence ($z_1,\ldots,z_n$)
- Decoder
  - From sequence **z**, generates output sequence ($y_1,\ldots,y_m$)
  - Model is autoregressive
- Stacked self-attention and point-wise fully connected layers
- Positional encodings allow for fully parallelized encodings

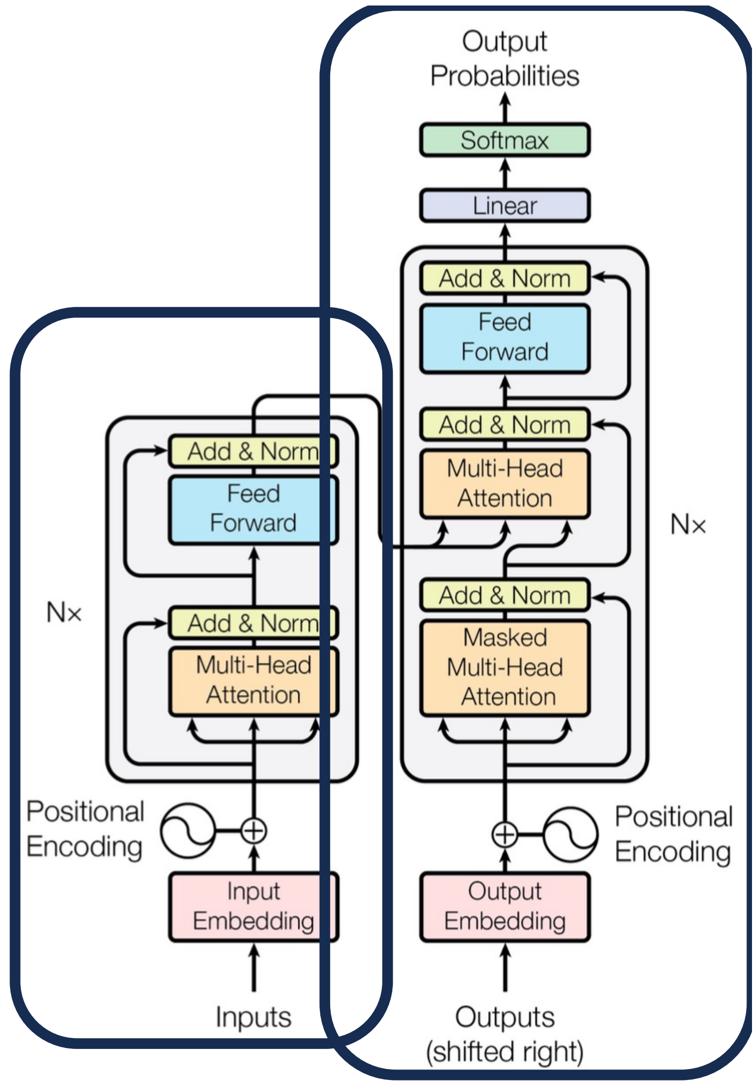


Figure 1: The Transformer - model architecture.

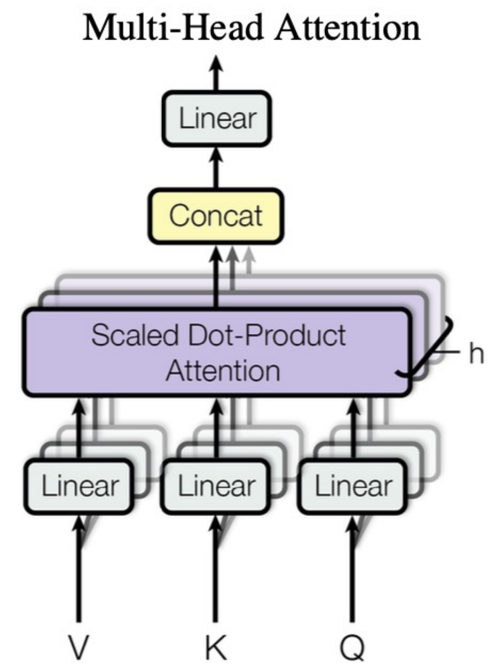
Attention

- Maps:
  - A query
  - A set of key-value pairs
- An output
  - Weighted sum of values
  - Weight assigned to each value is output of a "compatibility function" of query with corresponding key
- Query, Key, Value
- Vectors (in theory)
  - Matrices in practice (i.e., many queries/keys/values in parallel)

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

Attention

- In 2017 paper, Attention = "Scaled Dot-Product Attention"
- Multi-head attention
  - Linearly project Q, K, and V *h* times with different learned projections
  - Attention performed in parallel on each projection
  - Concatenated to compute final attention values



$$\mathrm{MultiHead}(Q, K, V) = \mathrm{Concat}(\mathrm{head_1}, ..., \mathrm{head_h})W^O$$
$$\text{where } \mathrm{head_i} = \mathrm{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

Attention and Self-Attention

- Encoder-decoder attention layers
  - Q from previous decoder layer
  - K, V from output of encoder
  - Every position in the decoder can attend to all input positions
- Encoder contains self-attention layers
  - Q, K, V all come from the same place (output of previous encoder layer)
  - Each position in encoder can attend to all positions in previous encoder layer
- Decoder contains self-attention layers
  - Each position in decoder can attend to all positions in previous decoder layer **up to and including the current position** (but not past it!)

Connections to Support Vector Machines

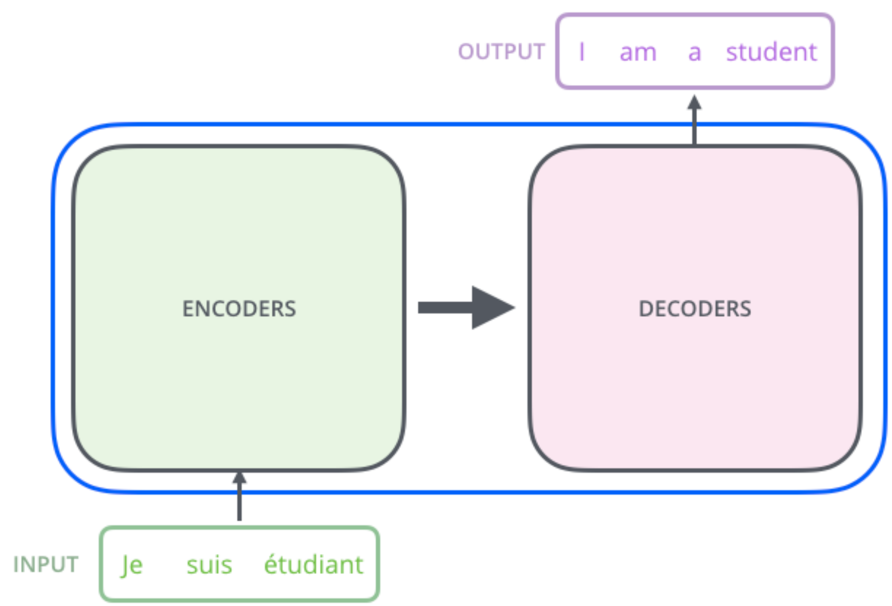
- SVMs were all the rage in the late 90s and early 2000s
  - Neural networks had been a “dead end” since early 90s
- 1-layer self-attention Transformers = hard-margin SVM
- Multilayer transformers = hierarchy of SVMs

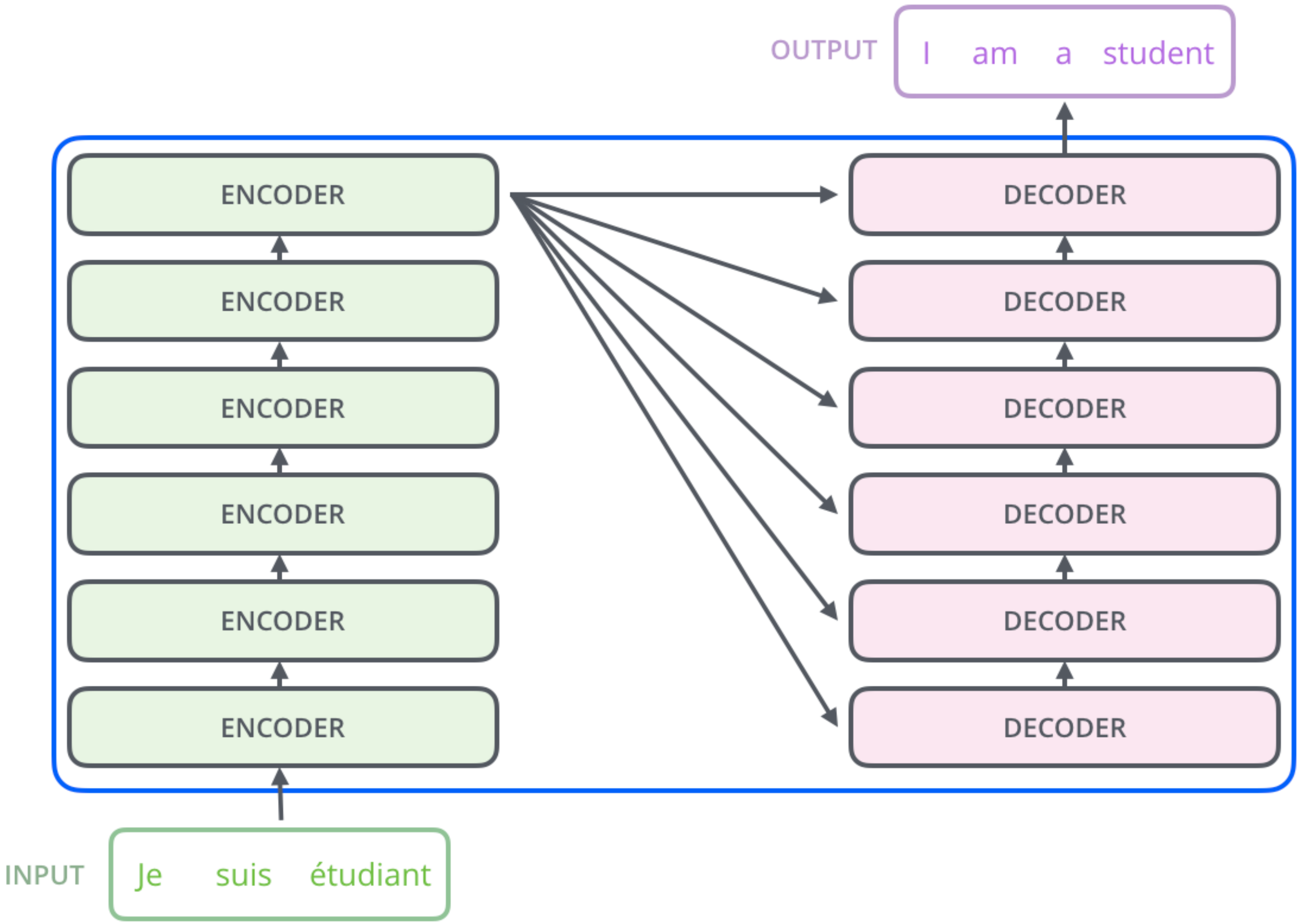
Transformers as Support Vector Machines

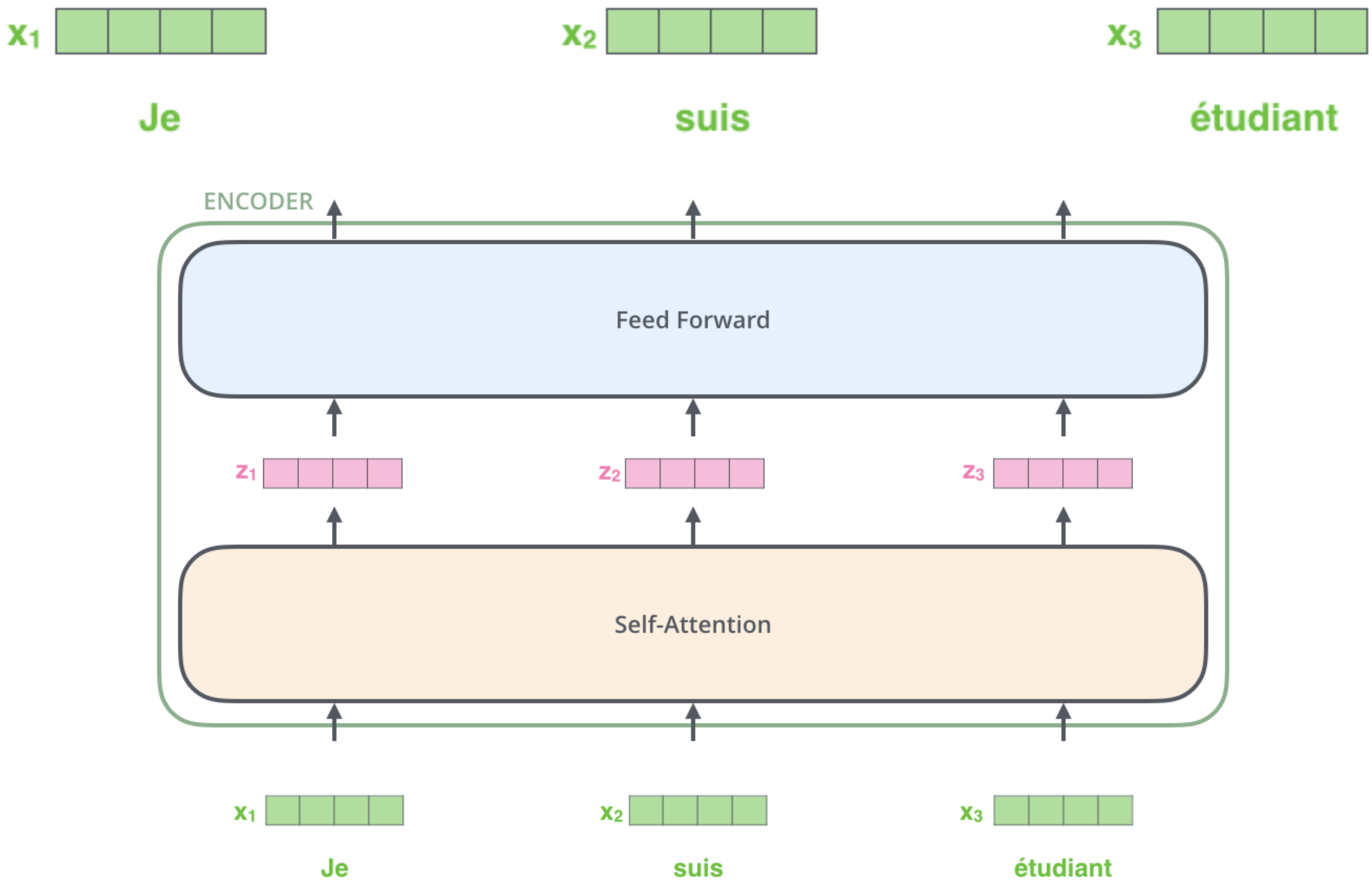
Davoud Ataee Tarzanagh[1⋆] Yingcong Li[2⋆] Christos Thrampoulidis[3] Samet Oymak[4†]

Let’s add some pictures

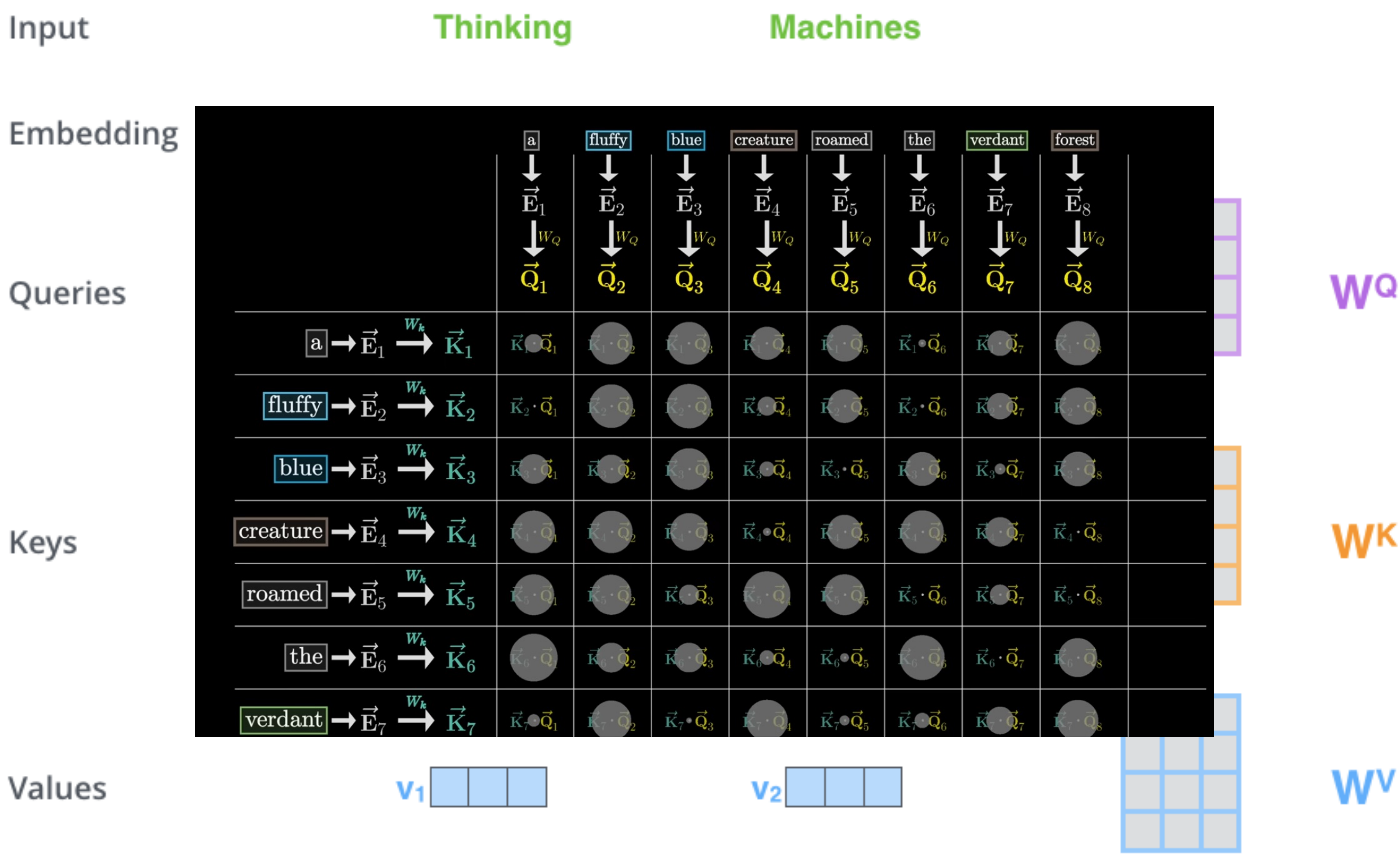
- Courtesy of The Illustrated Transformer



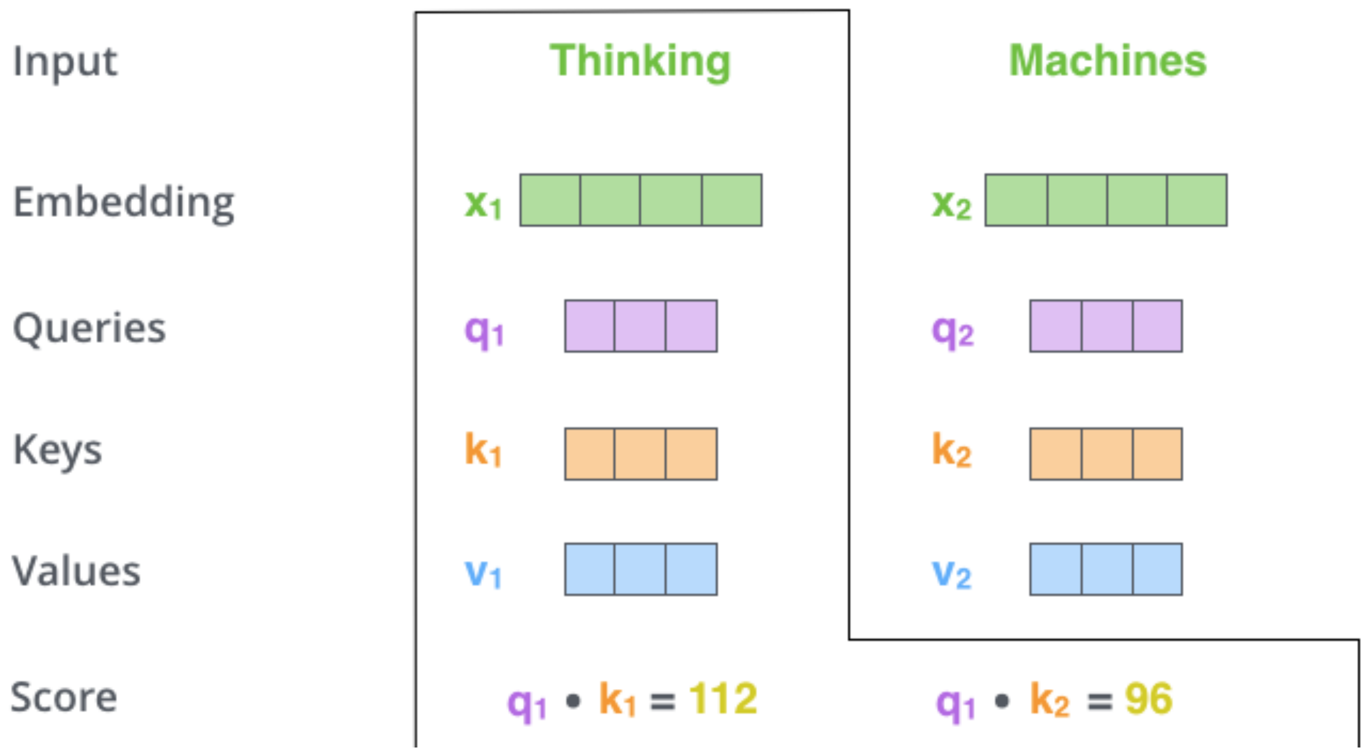

OUTPUT
I am a student
ENCODER
ENCODER
ENCODER
ENCODER
ENCODER
ENCODER
DECODER
DECODER
DECODER
DECODER
DECODER
DECODER
INPUT
Je suis étudiant



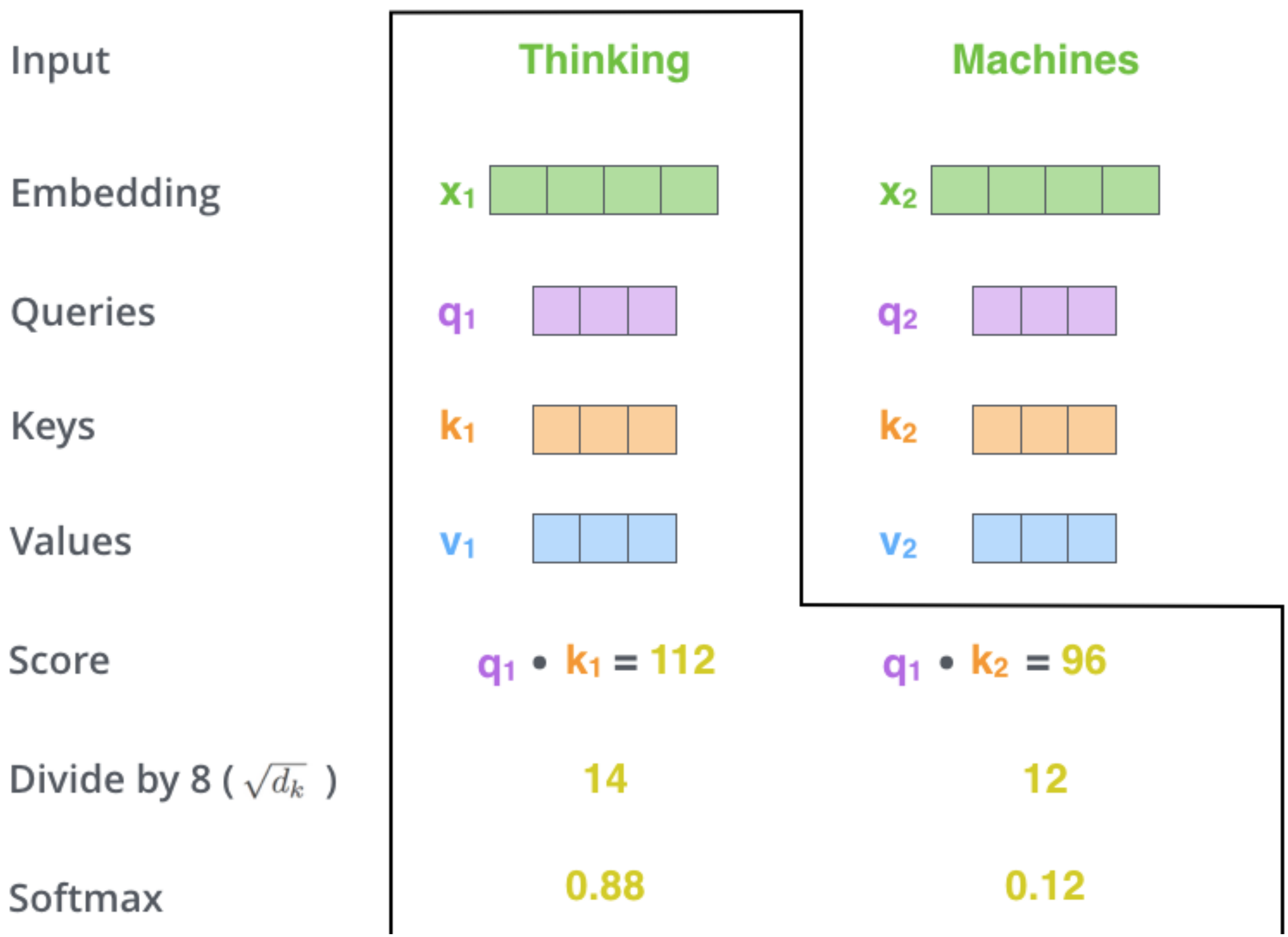
x_1
Je
x_2
suis
x_3
étudiant
ENCODER
Feed Forward
z_1
z_2
z_3
Self-Attention
x_1
Je
x_2
suis
x_3
étudiant

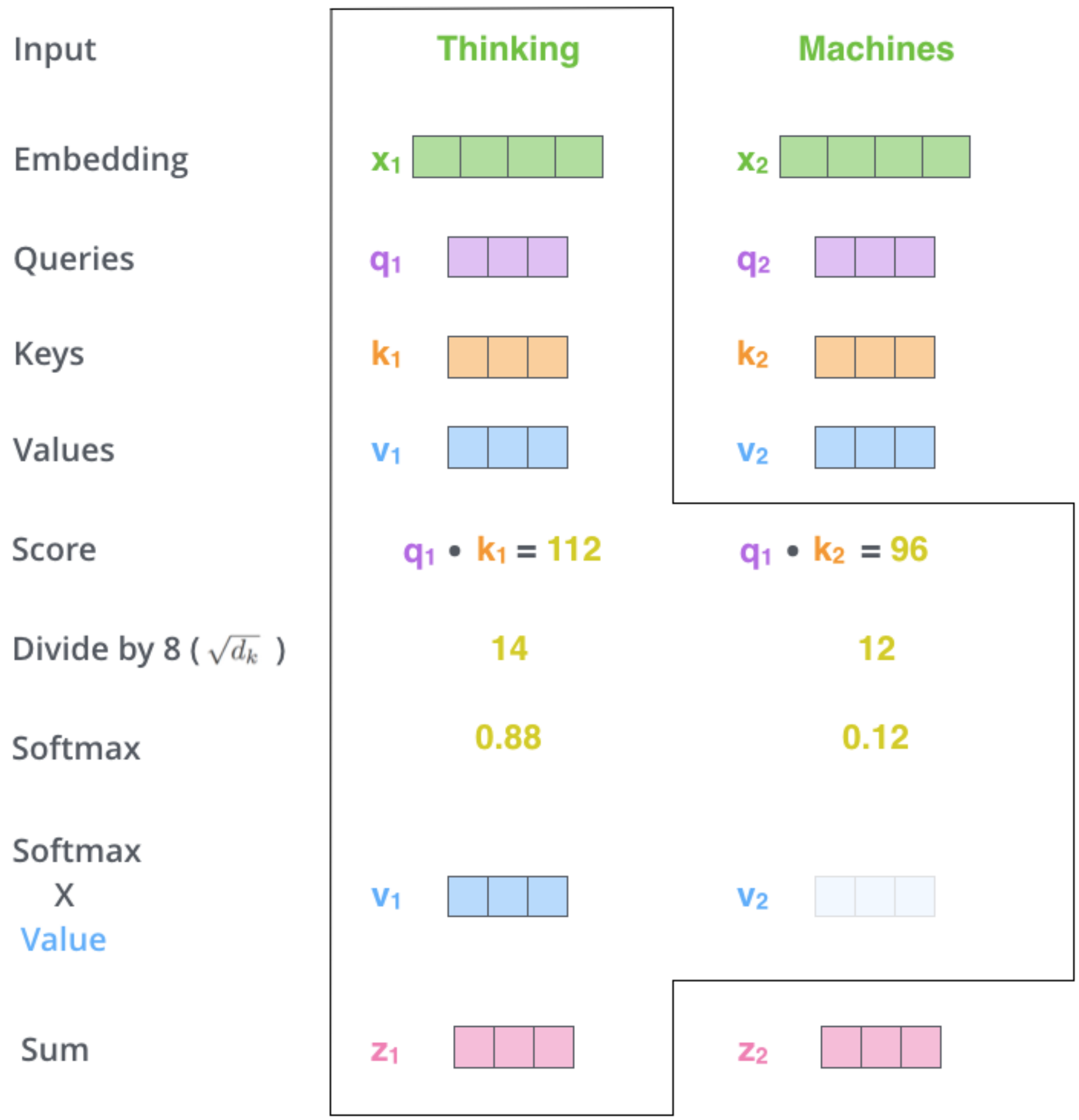


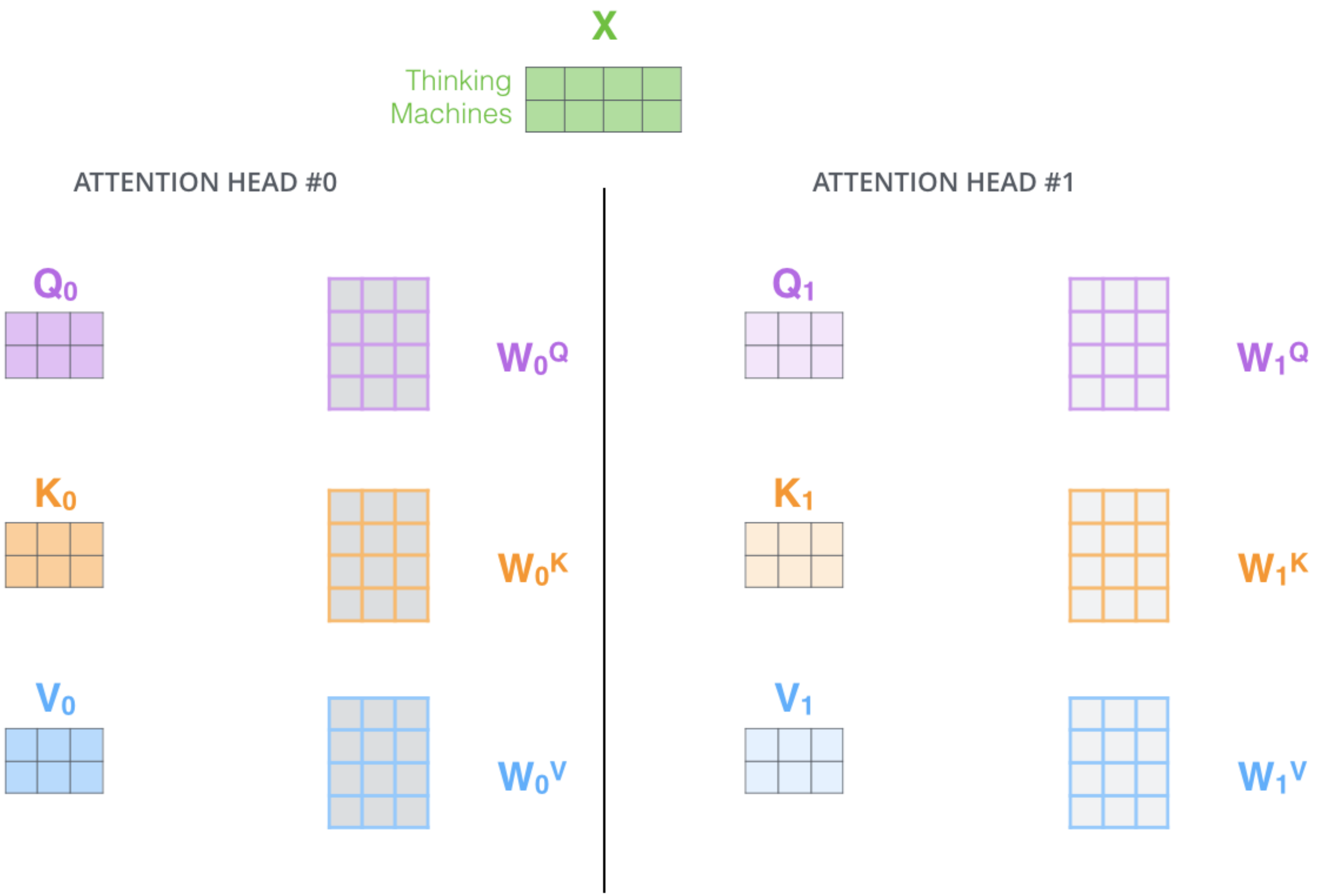
Input
Thinking
Machines
Embedding
Queries
Keys
Values
a
fluffy
blue
creature
roamed
the
verdant
forest
W^Q
W^K
W^V
v_1
v_2



Input
Thinking
Machines
Embedding
x1
x2
Queries
q1
q2
Keys
k1
k2
Values
v1
v2
Score
q1 • k1 = 112
q1 • k2 = 96


Input
Thinking
Machines
Embedding
x1
x2
Queries
q1
q2
Keys
k1
k2
Values
v1
v2
Score
q1 • k1 = 112
q1 • k2 = 96
Divide by 8 ($\sqrt{d_k}$)
14
12
Softmax
0.88
0.12

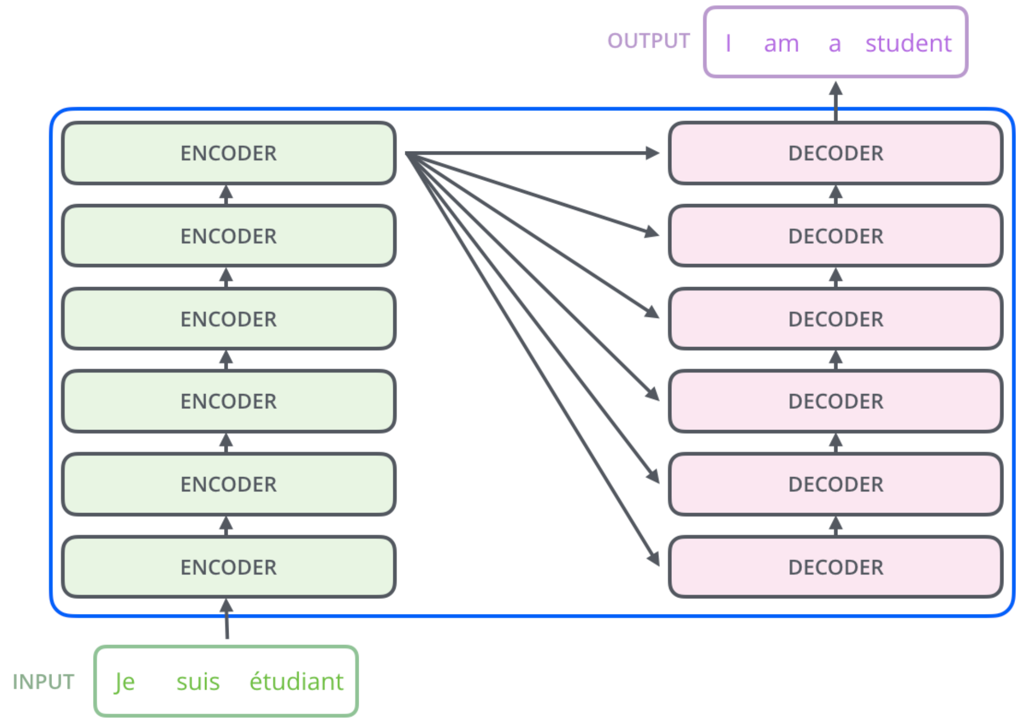

Input
Thinking
Machines
Embedding
x1
x2
Queries
q1
q2
Keys
k1
k2
Values
v1
v2
Score
q1 • k1 = 112
q1 • k2 = 96
Divide by 8 ($\sqrt{d_k}$)
14
12
Softmax
0.88
0.12
Softmax
X
Value
v1
v2
Sum
z1
z2



X
Thinking
Machines
ATTENTION HEAD #0
ATTENTION HEAD #1
Q_0
W_0^Q
K_0
W_0^K
V_0
W_0^V
Q_1
W_1^Q
K_1
W_1^K
V_1
W_1^V

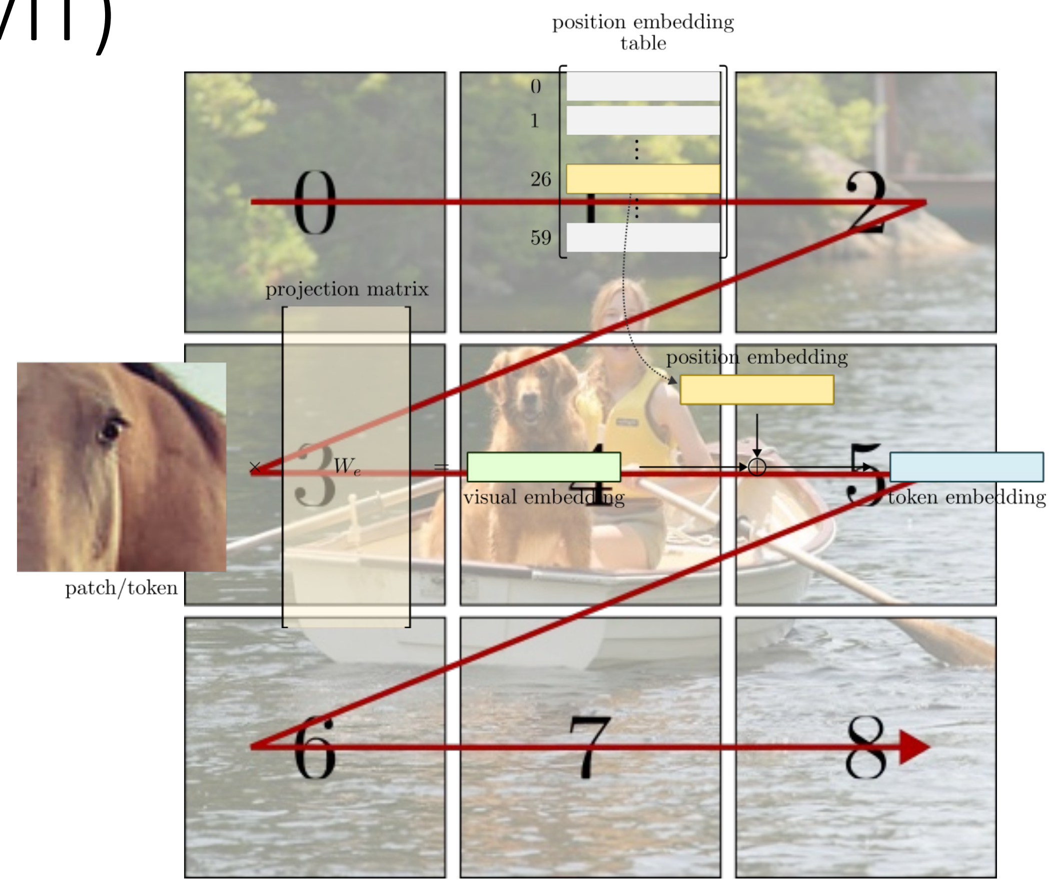
What about the decoder?

- Final K, V from encoder sent to each decoder
- Each decoder focuses its attention on “correct” positions of encoder



Vision Transformers (ViT)

- Built on the same principles
- Patches = tokens
  - Still have positional encodings
  - Are still embedded in the first encoder step
- Attention = dictionary lookup
  - dictionary[query] = value
  - If key==query, return value
  - “Soft” selection
- Everything else is the same!

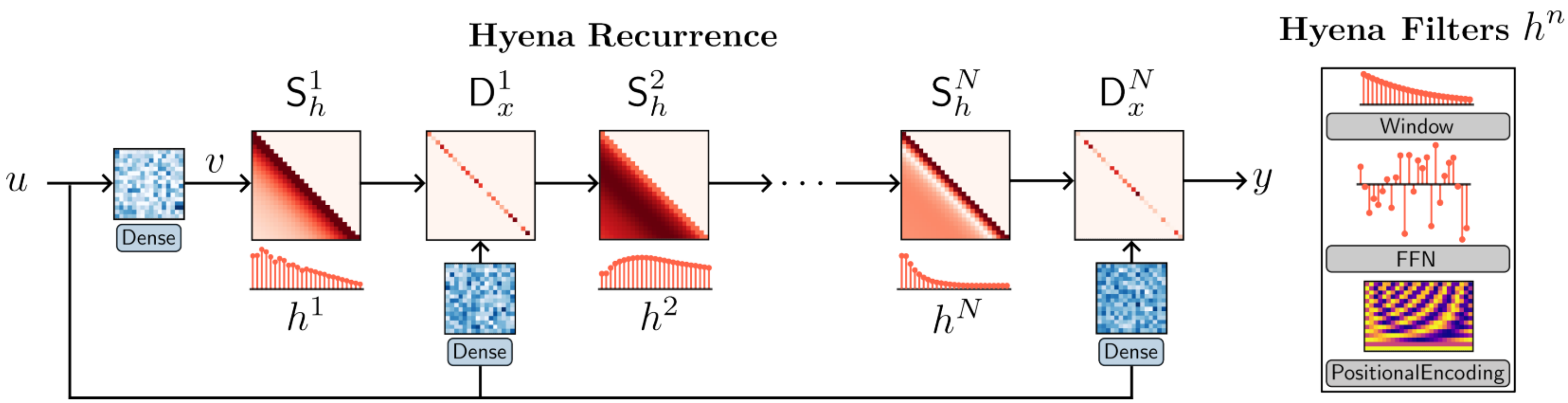


Transformer limitation

- Attention mechanism is still O(n^2)
  - Each token compared to each other token
  - Subquadratic methods exist but rely on low-rank / sparse approximations, and require dense Attention layers
  - **Ultimately limits the possible sequence length *n* (context window)**

Hyena

- Subquadratic drop-in Attention replacement
  - Hyena operator
- Long convolutions
  - filter sizes as long as the input
- Data-controlled gating (element-wise multiplication)
  - Convolutions in FFT (i.e., frequency) space are element-wise multiplications!



0 response submitted

The attention mechanism is

- x * Wq
- x * Wk
- key * query
- (key * query) * value
- this in-class quiz

Treemap Bar

1 of 1

Conclusions

- Transformer architecture for modeling sequences (of text or images)
  - Throws out recurrences of RNNs for more parallel training
  - Ditching recurrences also allows for arbitrary context windows
- Still use the encoder-decoder architecture
  - Input embeddings are critical to the overall performance
- Attention
  - Transformer allows for all tokens to “attend” to all other tokens
  - Can model extremely long-distance dependencies (spatially or sequentially)
  - Only drawback is quadratic computation time
- Hyena operator
  - Clever use of FFT-based convolutions and Toeplitz matrices to accelerate standard computations and produce subquadratic performance

References

- “Attention is All You Need”, https://arxiv.org/abs/1406.6909
- Mechanics of seq2seq with Attention https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/
- The Illustrated Transformer https://jalammar.github.io/illustrated-transformer/
- An Intuitive Introduction to the Vision Transformer https://sthalles.github.io/an-intuitive-introduction-to-the-vision-transformer/
- Transformers as support vector machines. https://arxiv.org/pdf/2308.16898.pdf
- Hyena Hierarchy: Towards Larger Convolutional Language Models https://arxiv.org/abs/2302.10866
- 3Blue1Brown: Transformers: How LLMs work, explained visually https://www.youtube.com/watch?v=wjZofJX0v4M
- 3Blue1Brown: Attention in Transformers, step-by-step https://www.youtube.com/watch?v=eMlx5fFNoYc

Up next

- Final Project Update #2 is due **TONIGHT!**
- Homework 5 is due next Tuesday! (April 15)
- Next Tuesday & Thursday (April 15 & 17): last lectures of the class
- April 22, 23, and 24: **Final Project Presentations**
  - If you have given me a preference for presentation day, yay!
  - If you have not, your time will be **randomized & determined each day!**
  - Presentations are 20 minutes, with 2-3 minutes for questions
  - **Please come support your classmates even after you have presented**