Transformers: More Than Meets AI

Brian Cruz
UC Berkeley, Advocate

About Me

- Favorite undergraduate math class:
  - Set theory
- Studied DNA topology using stochastic methods and GPUs
- Worked in industry as a data engineer, AI/ML engineer, and AI evangelist.
- Currently Head of AI Engineering for Advocate
- Came back to school because I missed Evans Hall

“We are all going to be in jeopardy of being replaced by machines” — Fran Drescher, president of SAG-AFTRA

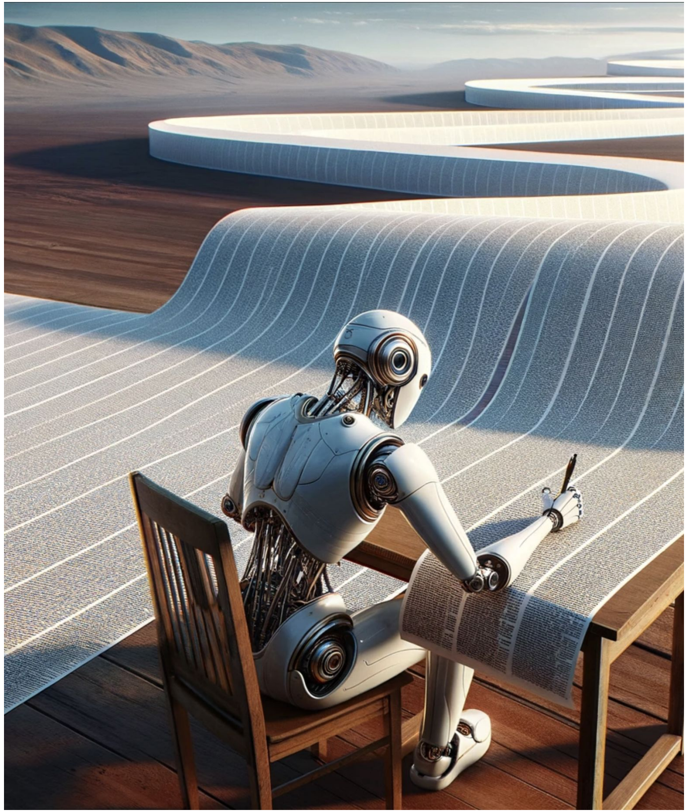
Transformers: More than Meets AI

1. A Brief History of Sequence Modeling
2. Attention Is All You Need
3. Research on LLMs
4. Build Your Own GPT!

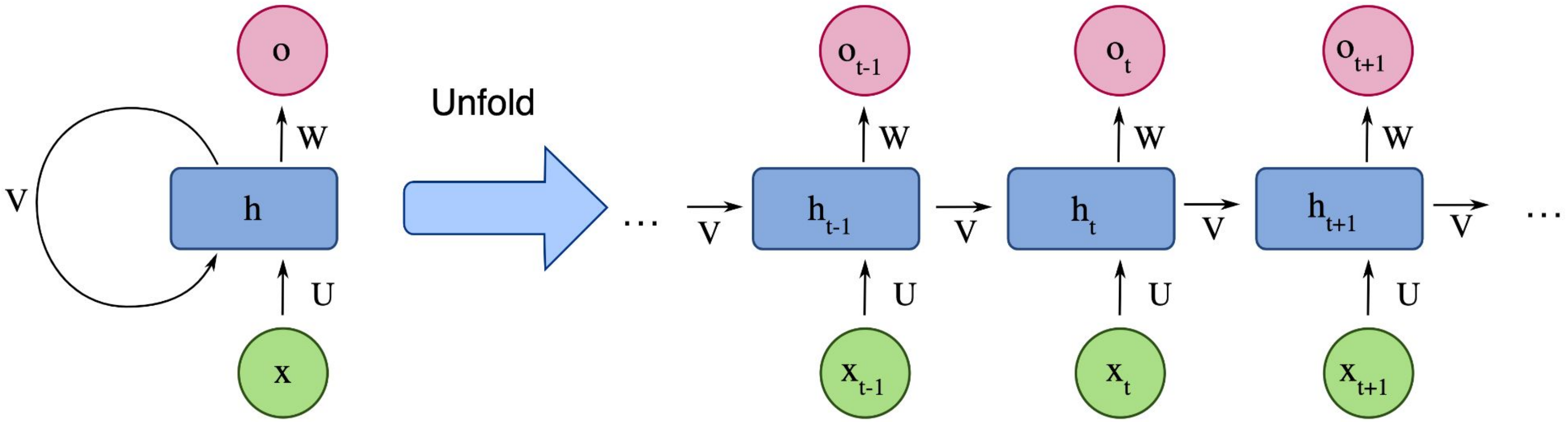
A Brief History of Sequence Modeling

Examples of Sequence Modeling

- Time series prediction
- Classify the sentiment of a sentence
- Translate a passage
- Write a five-paragraph essay



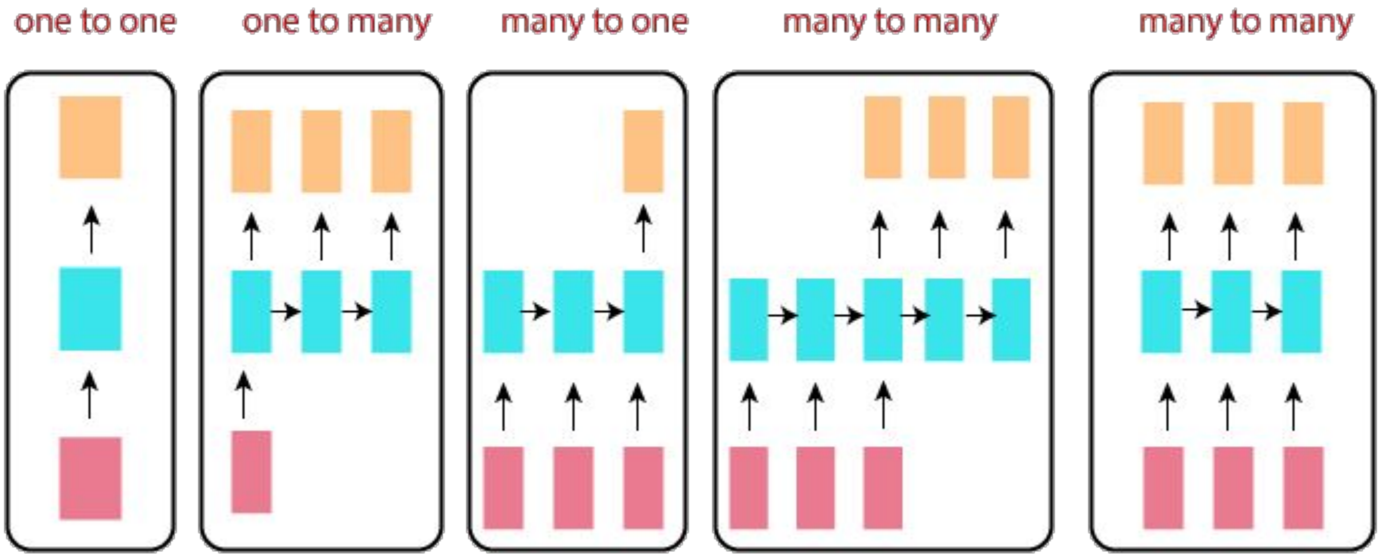
RNNs



RNN Code

```
import torch.nn as nn

class RNN(nn.Module):
    def __init__(self, input_size, hidden_size, output_size):
        super(RNN, self).__init__()

        self.hidden_size = hidden_size

        self.i2h = nn.Linear(input_size + hidden_size, hidden_size)
        self.h2o = nn.Linear(hidden_size, output_size)
        self.softmax = nn.LogSoftmax(dim=1)

    def forward(self, input, hidden):
        combined = torch.cat((input, hidden), 1)
        hidden = self.i2h(combined)
        output = self.h2o(hidden)
        output = self.softmax(output)
        return output, hidden

    def initHidden(self):
        return torch.zeros(1, self.hidden_size)

n_hidden = 128
rnn = RNN(n_letters, n_hidden, n_categories)
```

Sequences in, Sequences out



Problems with RNNs

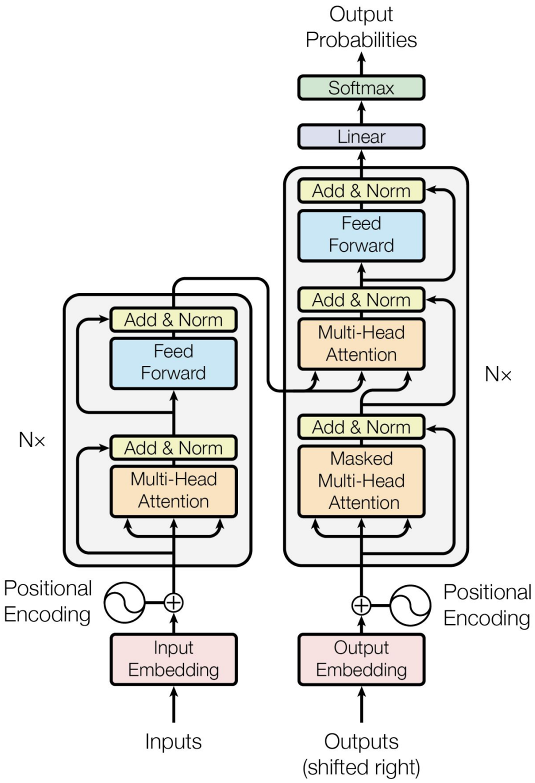
- Vanishing and exploding gradients*
- Handling long-term dependencies*
- Sequential computation during training*

* from the Transformers marketing team

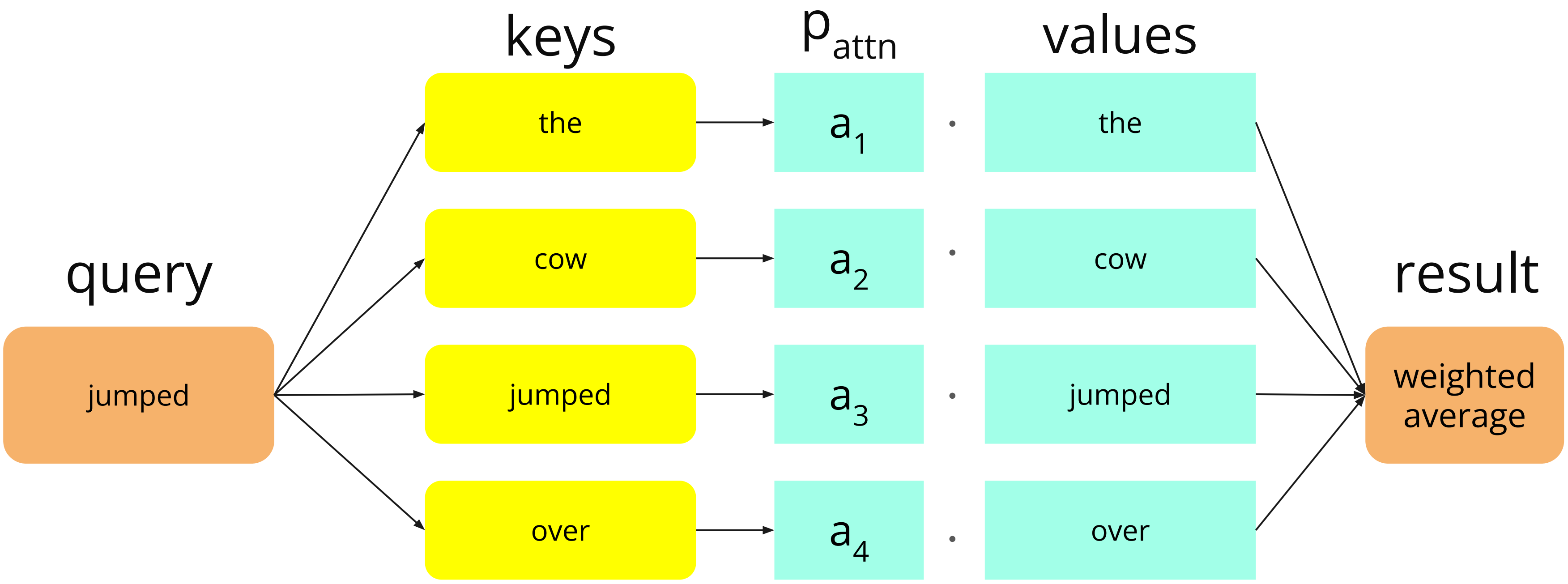
Attention Is All You Need

Attention is All You Need

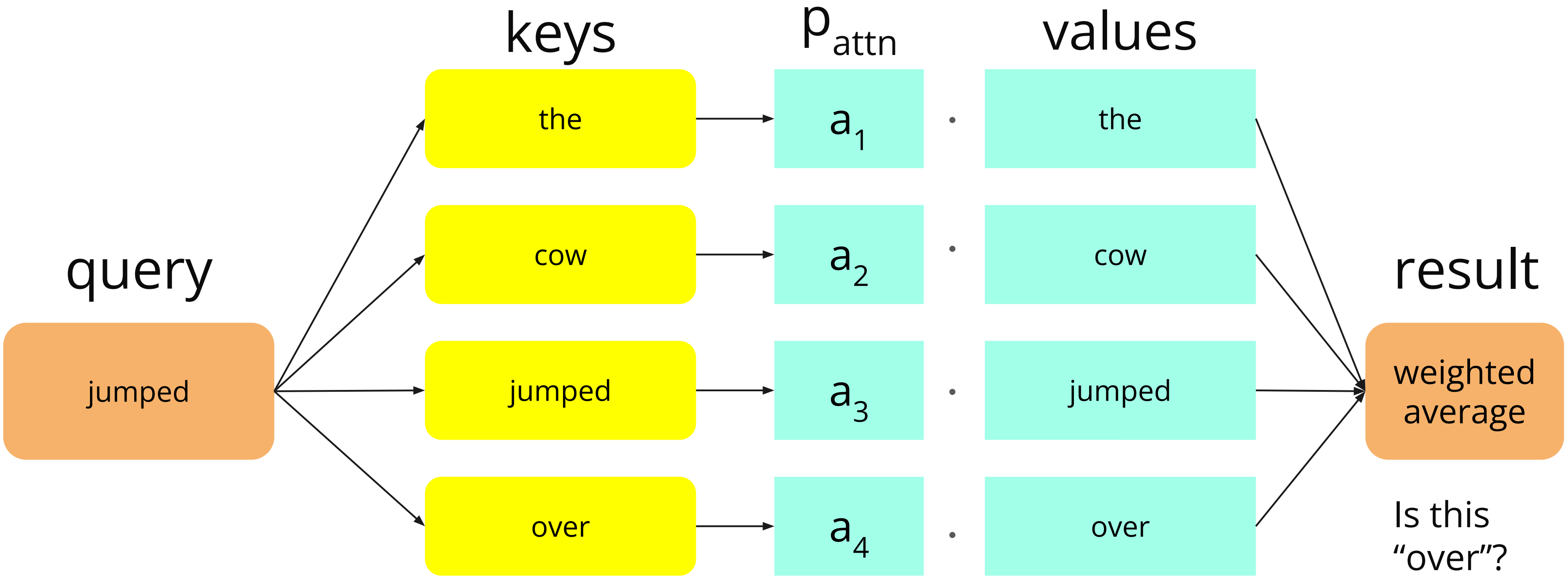
- Attention is All Your Need (Vaswani et al. 2017)
- The image on the right is an “encoder-decoder architecture”
- BERT is just the encoder
- ChatGPT is just the decoder



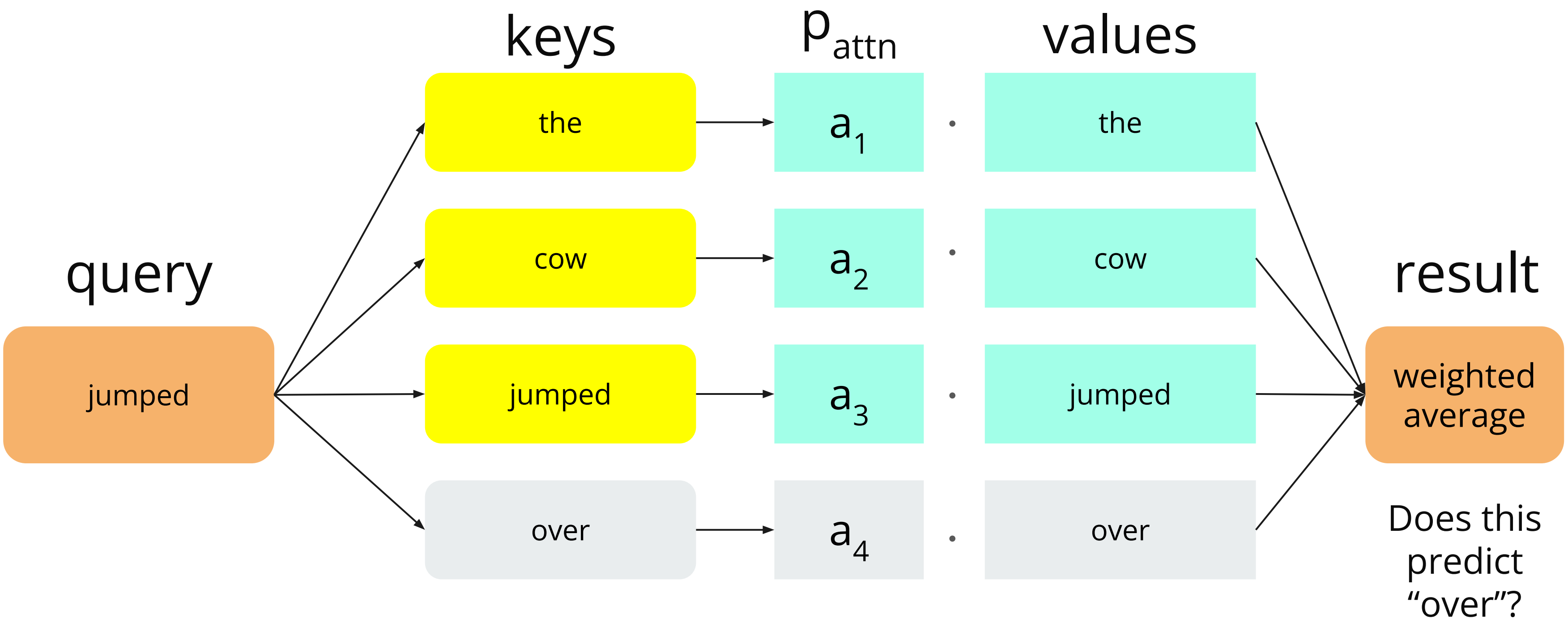
Attention is a weighted average



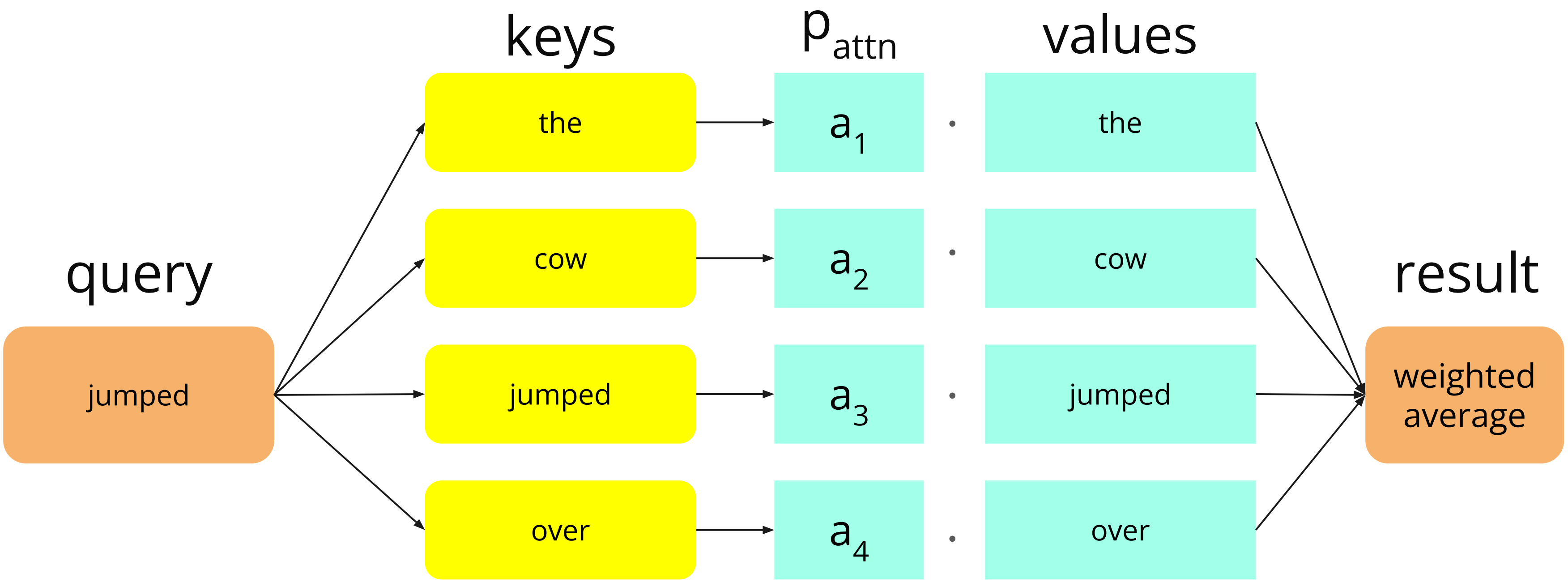
Auto-regressive pre-training objective predicts next word



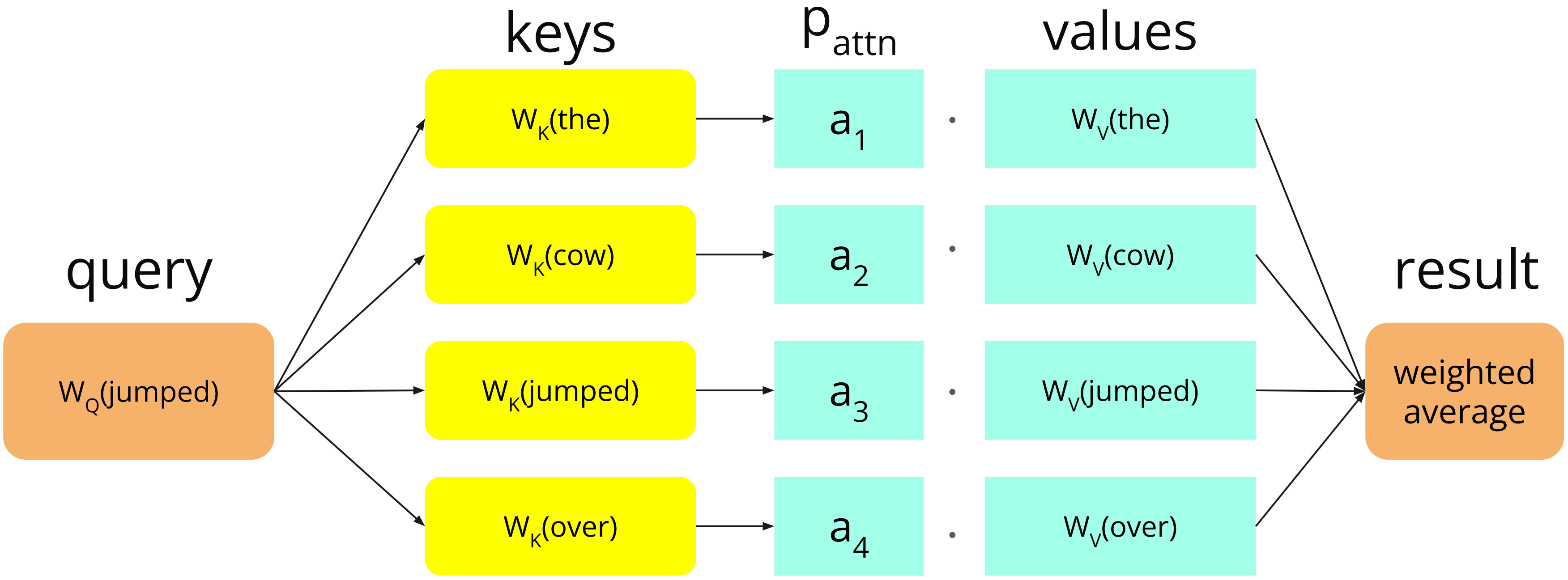
Masked attention hides the next word when we try to predict it



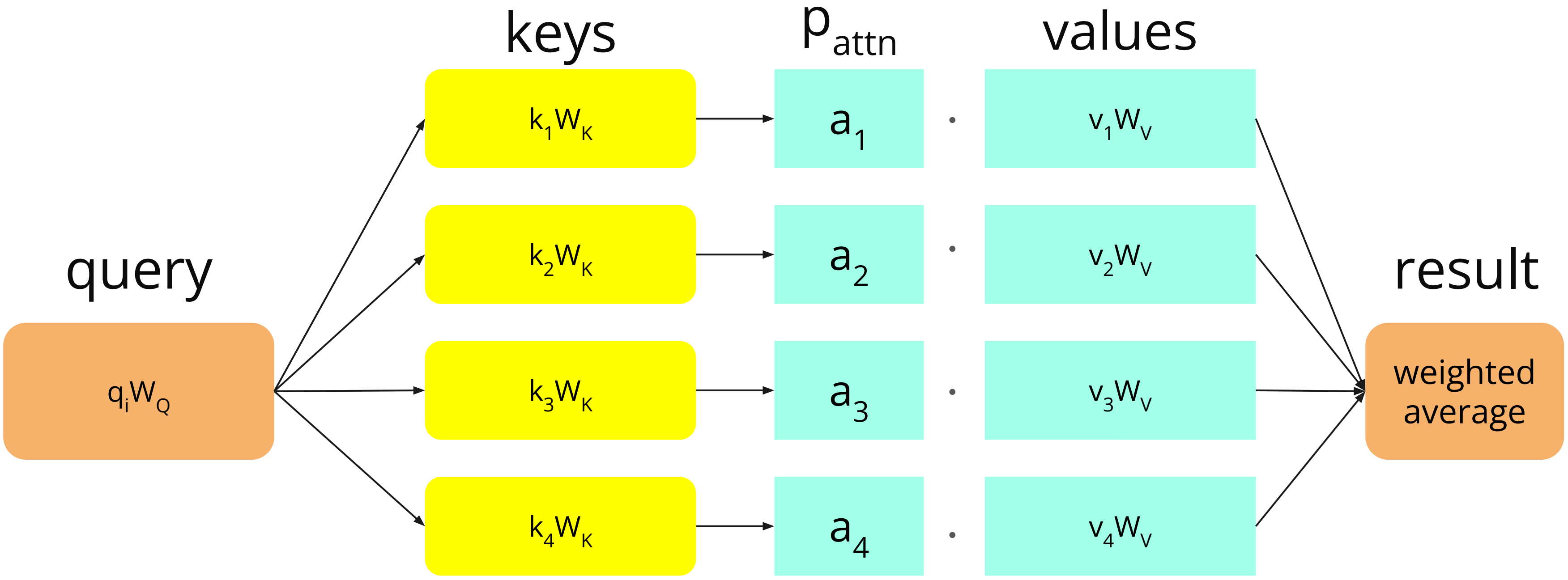
General attention



Linear projections W_Q, W_K, and W_V first applied



Linear projections W_Q, W_K, and W_V first applied

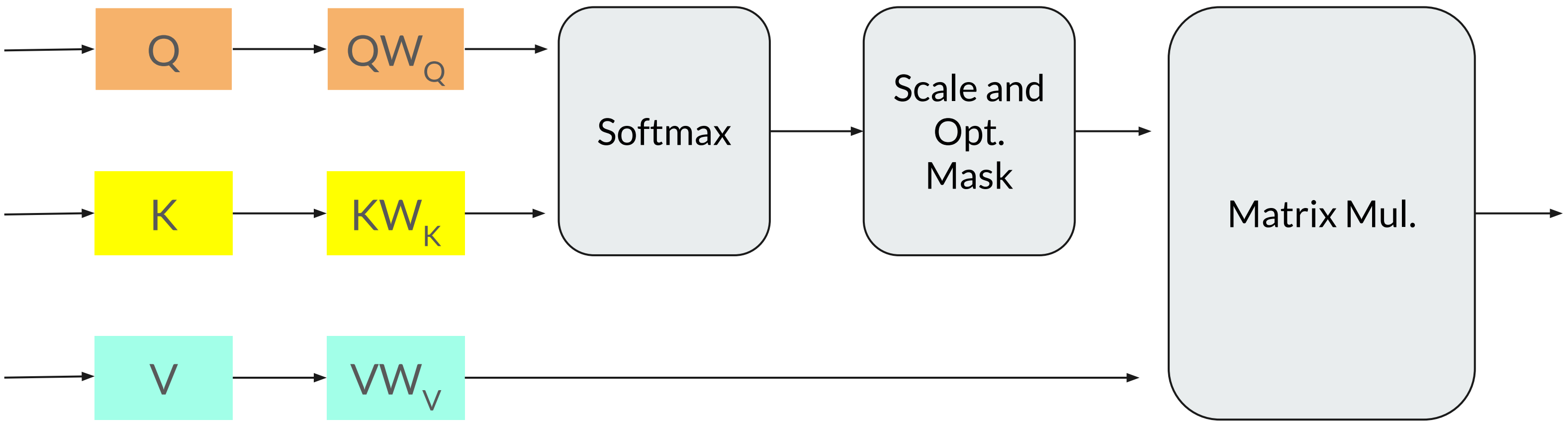


Scaled Dot-Product Attention

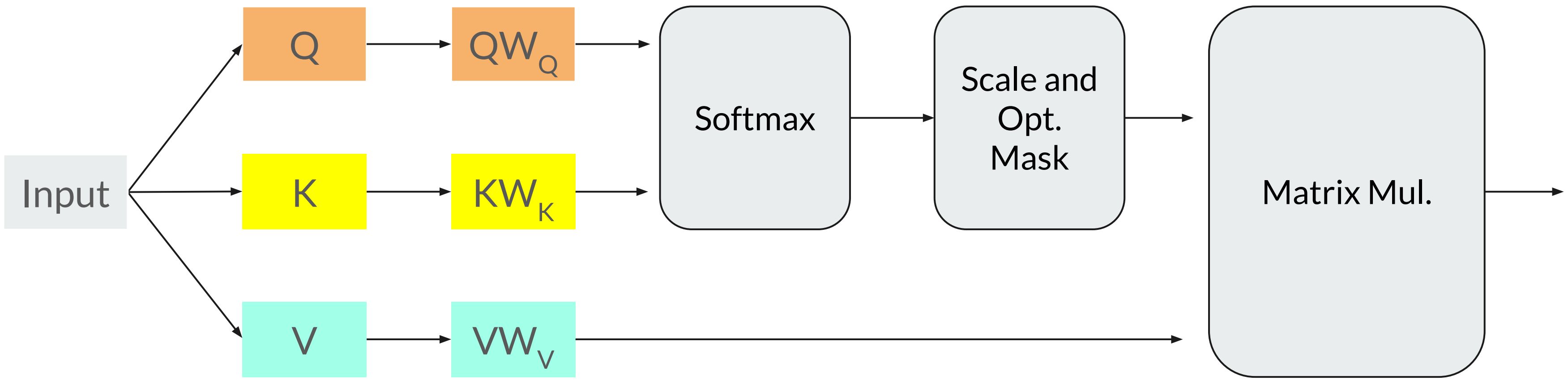
“In practice, we compute the attention function on a set of queries simultaneously, packed together into a matrix Q. The keys and values are also packed together into matrices K and V . We compute the matrix of outputs as:”

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

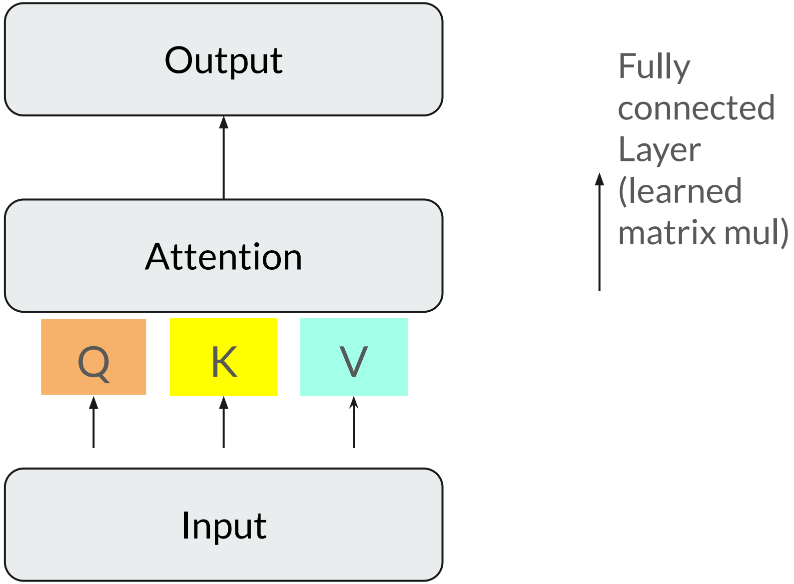
Scaled Dot-Product Attention

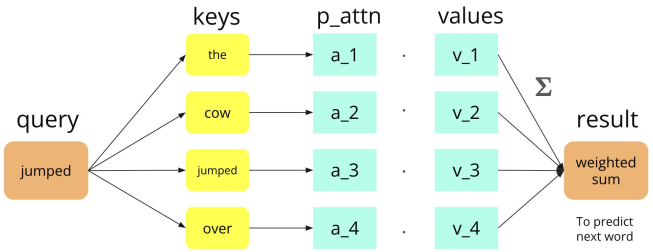


For Self-Attention Q=K=V



Self-Attention

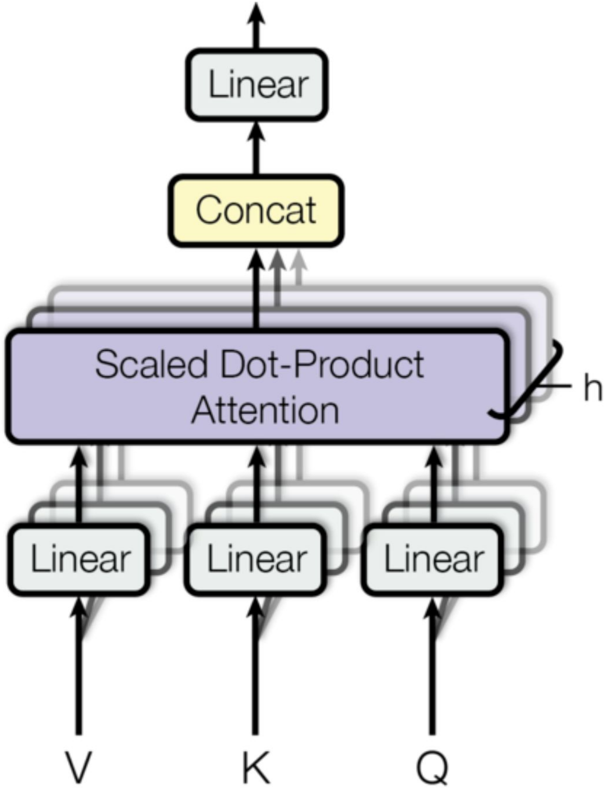




- Input and Output can be thought of as sequences of embeddings (matrices), like Q, K, and V themselves
- Self-attention uses the same input for Q, K, and V
- NOTE: Q, K, and V sometimes refers to the inputs to the attention function the "Input"

Multi-Head Attention

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



```python
def forward(self, x):
    # batch size, sequence length (in tokens), embedding dimensionality
    B, T, C = x.size()
    hs = C // self.n_head  # head size

    k = self.key(x).view(B, T, self.n_head, hs).transpose(1, 2)
    q = self.query(x).view(B, T, self.n_head, hs).transpose(1, 2)
    v = self.value(x).view(B, T, self.n_head, hs).transpose(1, 2)

    k_t = k.transpose(-2, -1)
    d_k = k.size(-1)

    att = F.softmax(q @ k_t / math.sqrt(d_k), dim=-1)

    y = att @ v

    # re-assemble all head outputs side by side
    y = y.transpose(1, 2).contiguous().view(B, T, C)

    # output projection
    y = self.proj(y)
    return y
```

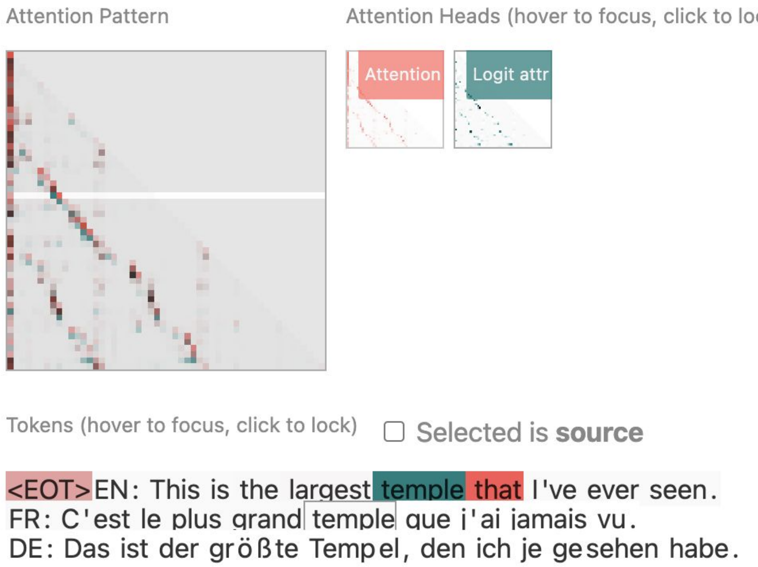
Check out “The Annotated Transformer”

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

```
def attention(query, key, value, mask=None, dropout=None):
    "Compute 'Scaled Dot Product Attention'"
    d_k = query.size(-1)
    scores = torch.matmul(query, key.transpose(-2, -1)) / math.sqrt(d_k)
    if mask is not None:
        scores = scores.masked_fill(mask == 0, -1e9)
    p_attn = scores.softmax(dim=-1)
    if dropout is not None:
        p_attn = dropout(p_attn)
    return torch.matmul(p_attn, value), p_attn
```

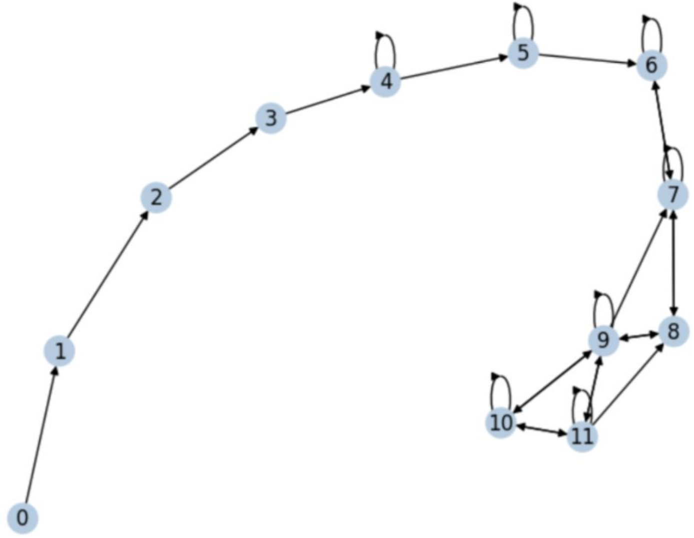
Research on LLMs

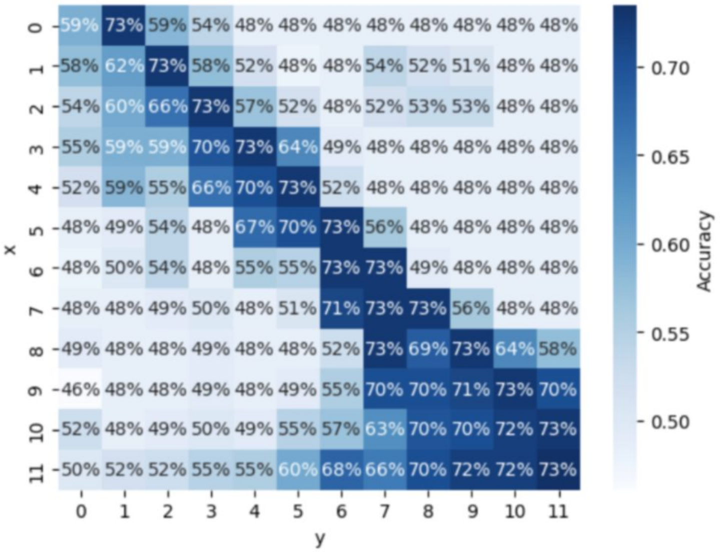
Model Understanding: Induction Heads



https://transformer-circuits.pub/2022/in-context-learning-and-induction-heads/index.html

Permuting the Layer Structure of RoBERTa (RTE Accuracy)

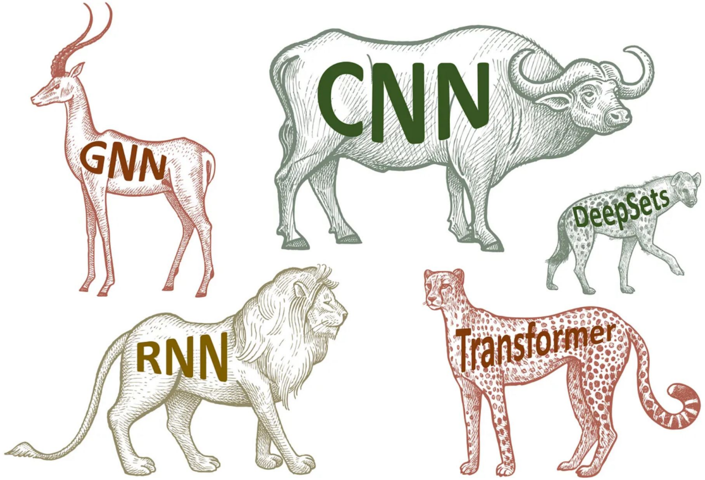




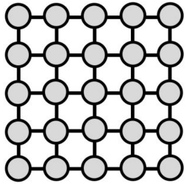
Left: Directed graph of the transitions for models with accuracy ≥ 70%
Right: Transitions Heatmap

Geometric Deep Learning

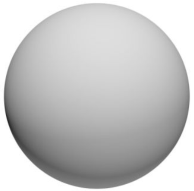
- E.g., extends CNNs to curved manifolds
- Theory to unify different neural networks architecture families based on **message passing** (generalization of convolution kernels)



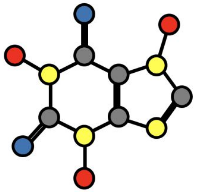
Deep learning today: a zoo of architectures, few unifying principles. Animal images: ShutterStock.



Grids



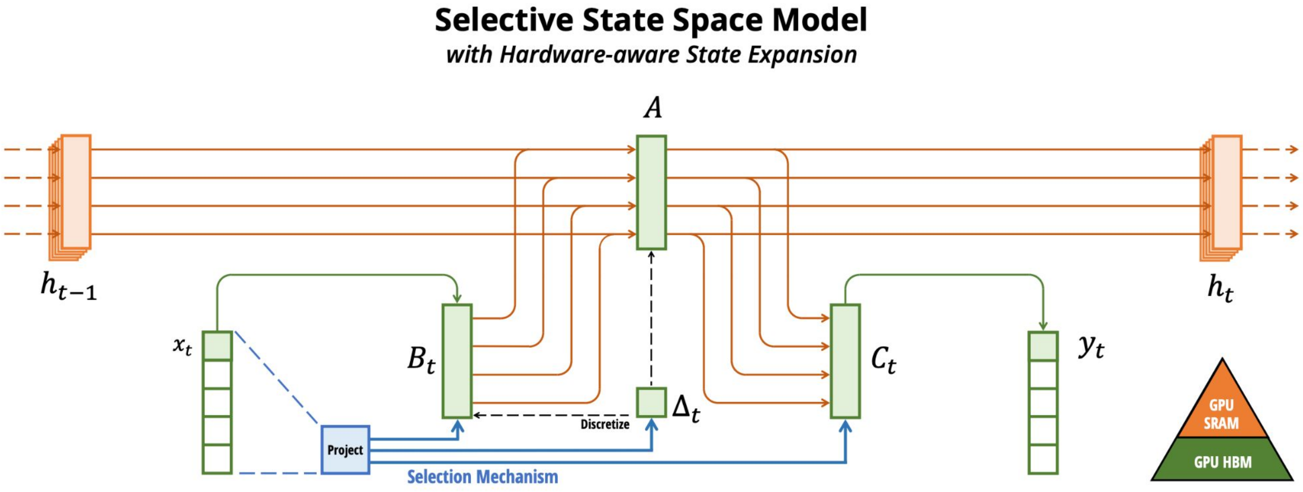
Groups



Graphs

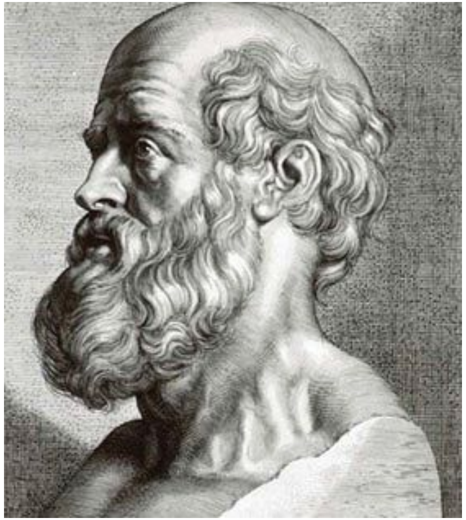
Geodesics & Gauges

[Mamba: Linear-Time Sequence Modeling with Selective State Spaces]
(Gu & Dao 2023)



What is Responsible AI?

- A Human Rights-Based Approach to Responsible AI (Prabhakaran et al 2022)
- Universal Declaration of Human Rights



Train Your Own Transformer!

Thanks!