Machine Learning for Physics

Transformers

Lisbon School on Machine Learning for Physics 2025, LIP Lisboa, Portugal

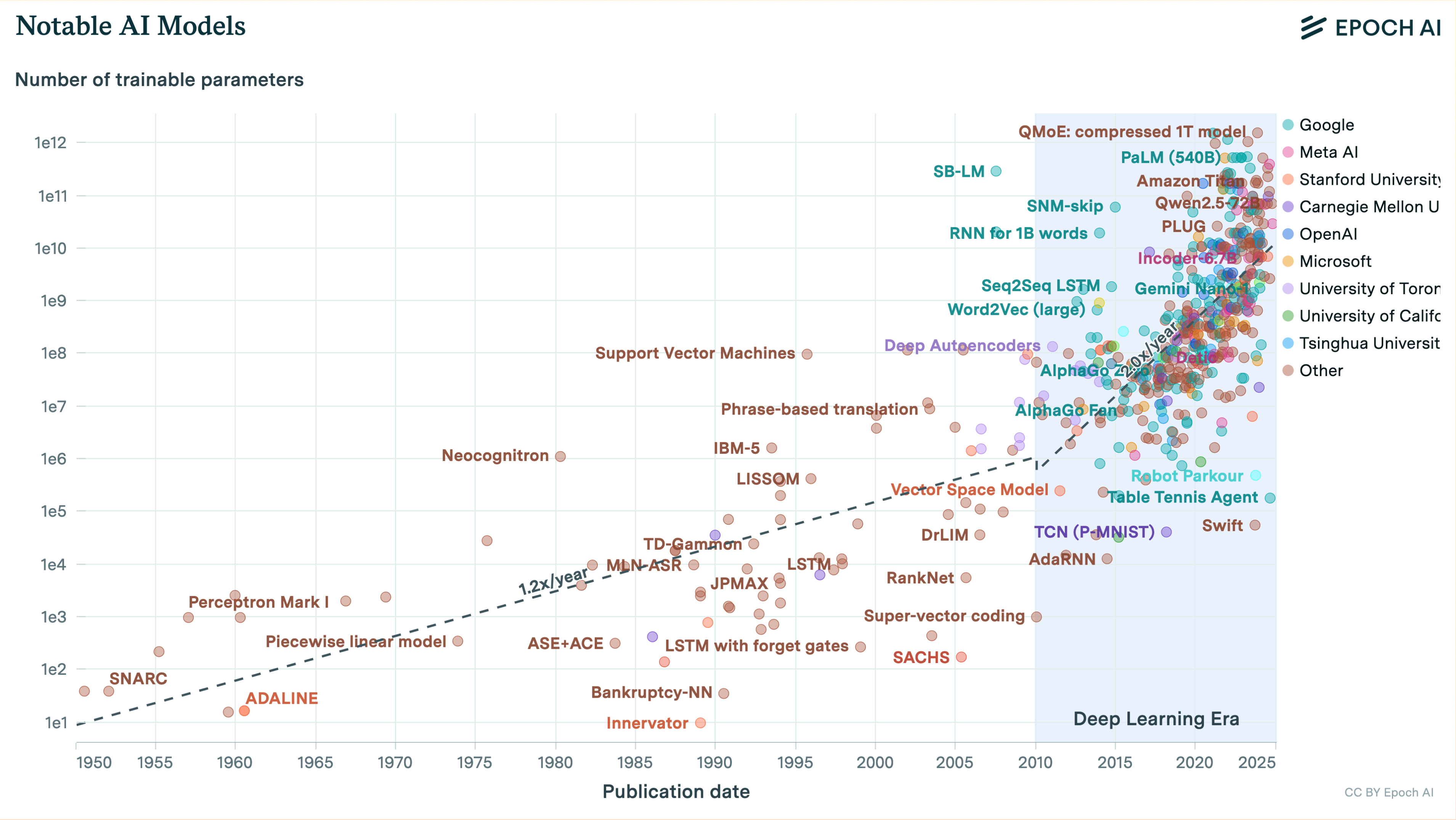
Pietro Vischia
pietro.vischia@cern.ch
@pietrovischia

If you are reading this as a web page: have fun! If you are reading this as a PDF: please visit

https://www.hep.uniovi.es/vischia/persistent/2025-03-12_LisbonMLSchoolPhysics_Transformers.html

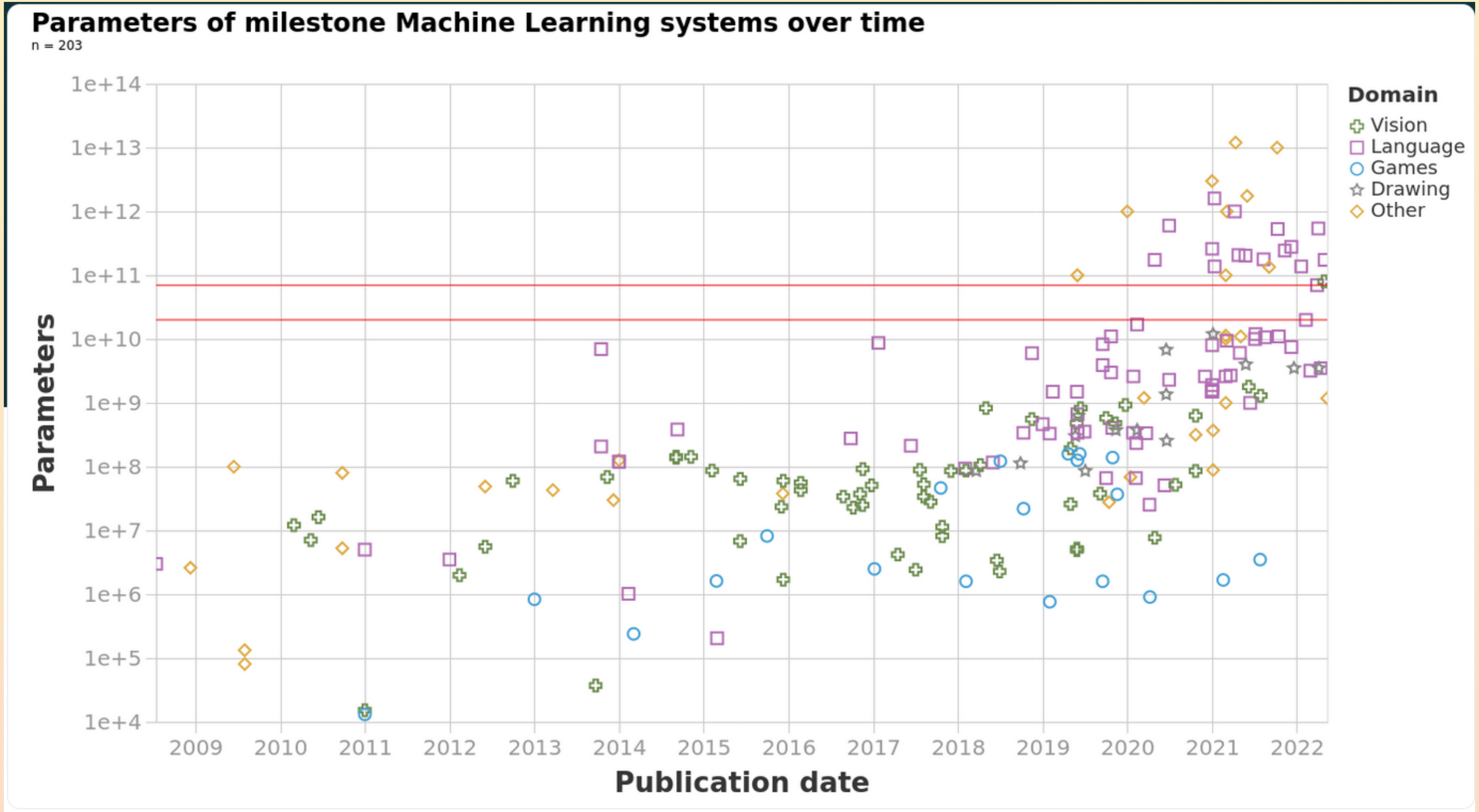
to get the version with working animations

Deep Learning Era



Plot from epoch.ai

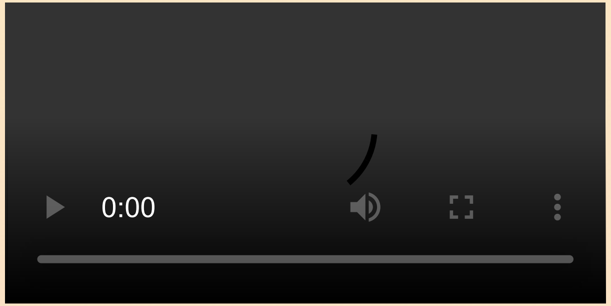
The Gap (induced by GPT-3)



Plot from 2207.02852

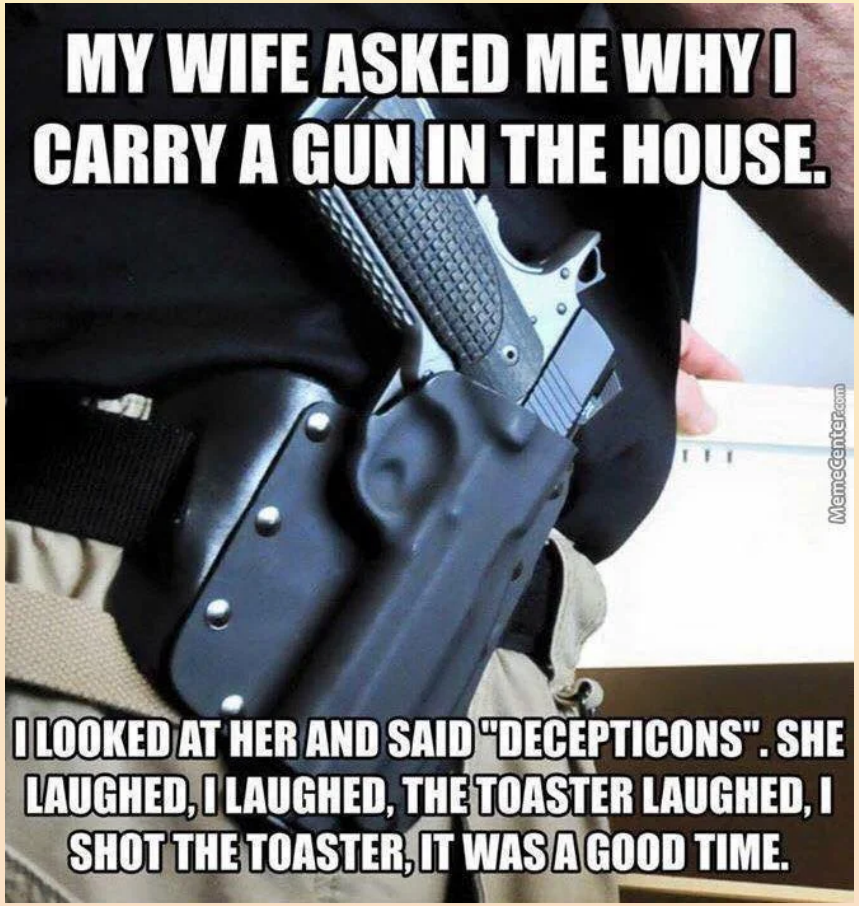
Transformers

- Transform a set of vectors into a corresponding set of vectors in another representation space having the same dimensionality
  - The new representation is designed to make it easier to solve the task at hand
- The architecture is totally general: inputs can be of any kind, or a mixture of them
  - Vectors
  - Text
  - Images
  - Audio
  - ...



—
Image from youtube and FastForward Labs

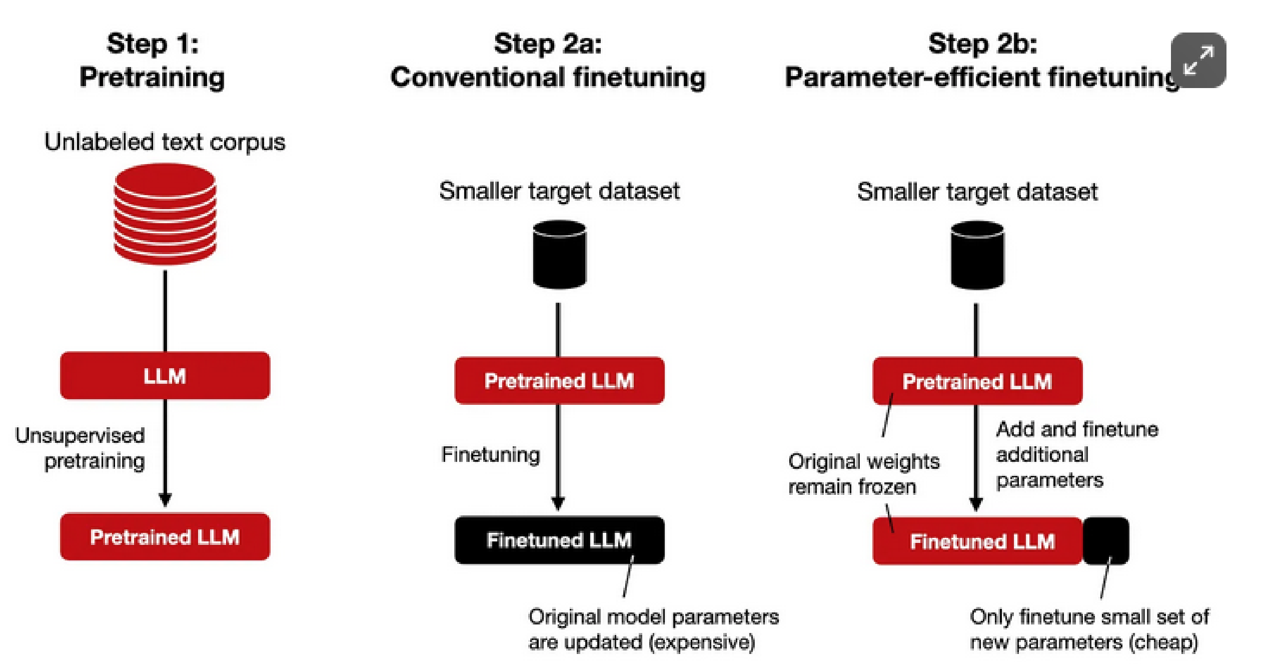
Transformers



—
Image from reddit

Fine tuning

- Train (typically unsupervised) a large model on enormous amounts of data
- Pick the trained model, and resume the training only with the small dataset corresponding to the task at hand
  - Or fix parameters of the large model, and add maybe a trainable layer that does classification, etc



The idea of parameter-efficient finetuning techniques (like prompt, prefix, and adapter tuning) is to add a small set of parameters to a pretrained LLM. Only the newly added parameters are finetuned while all the parameters of the pretrained LLM remain frozen.

—
Image by Sebastian Raschka

Foundation Models

- Scaling hypothesis: by increasing the number of learnable parameters, and training on a commensurately large dataset, the performance of the model will increase significantly without needing to change the architecture of the model iself

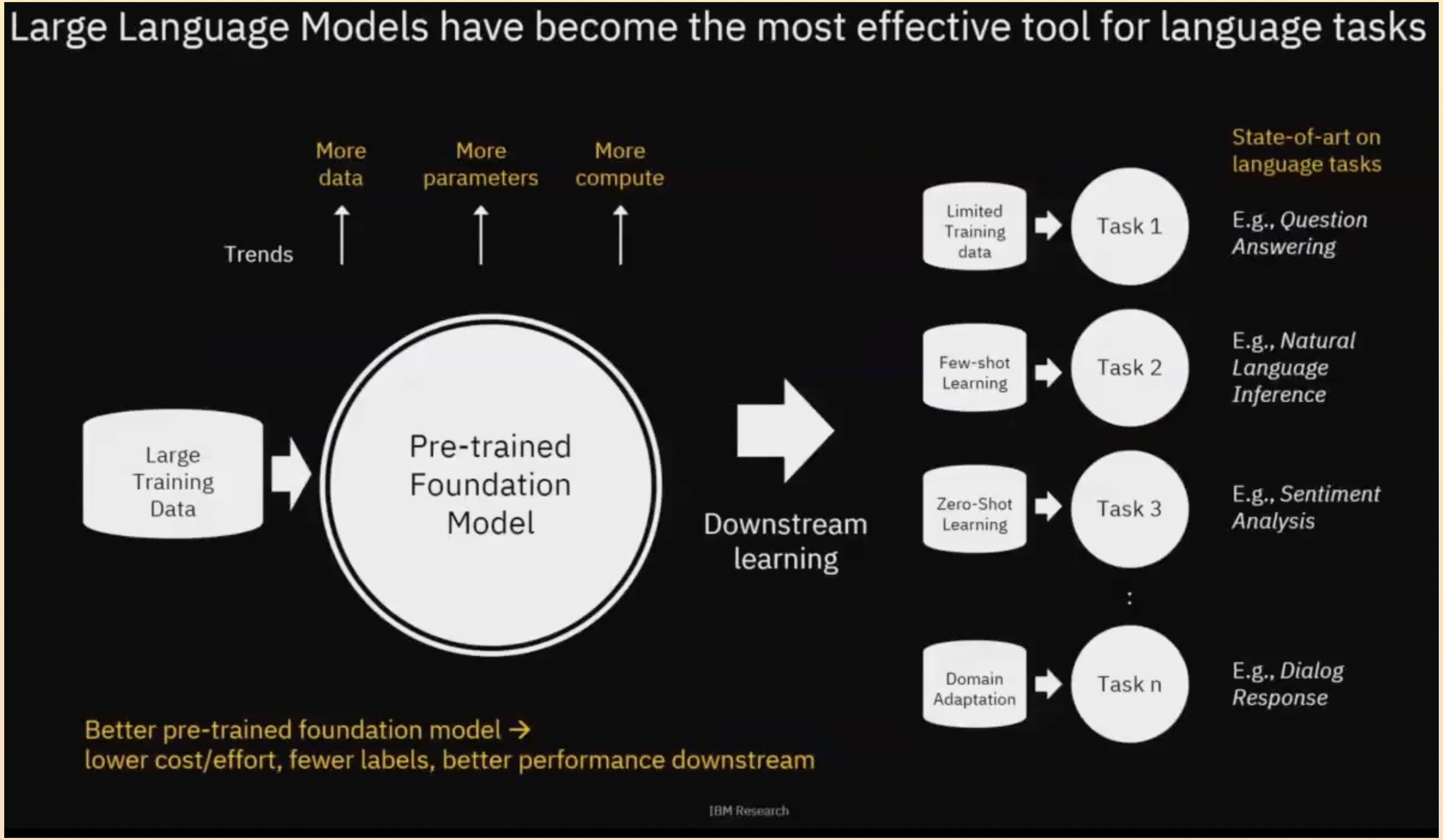
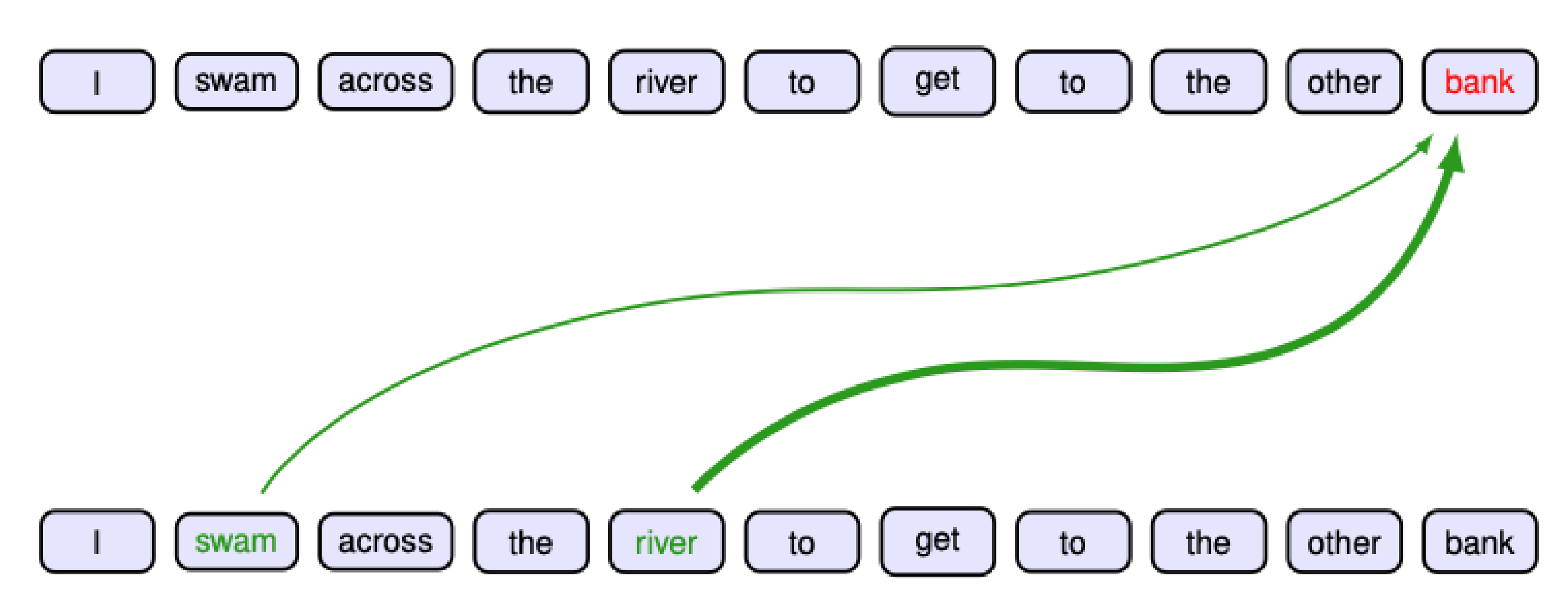


Image by John R. Smith (IBM Research)

Attention

- Sometimes the meaning (and consequently the downstream treatment of the data) depends on other elements of the sequence
  - I swam across the river to get to the other bank.
  - I walked across the road to get cash from the bank.
- Different elements in different positions influence the word each time
  - Regular networks: train to find the weights, then freeze the parameters
  - Attention weights: weight depend on the specific input data



—

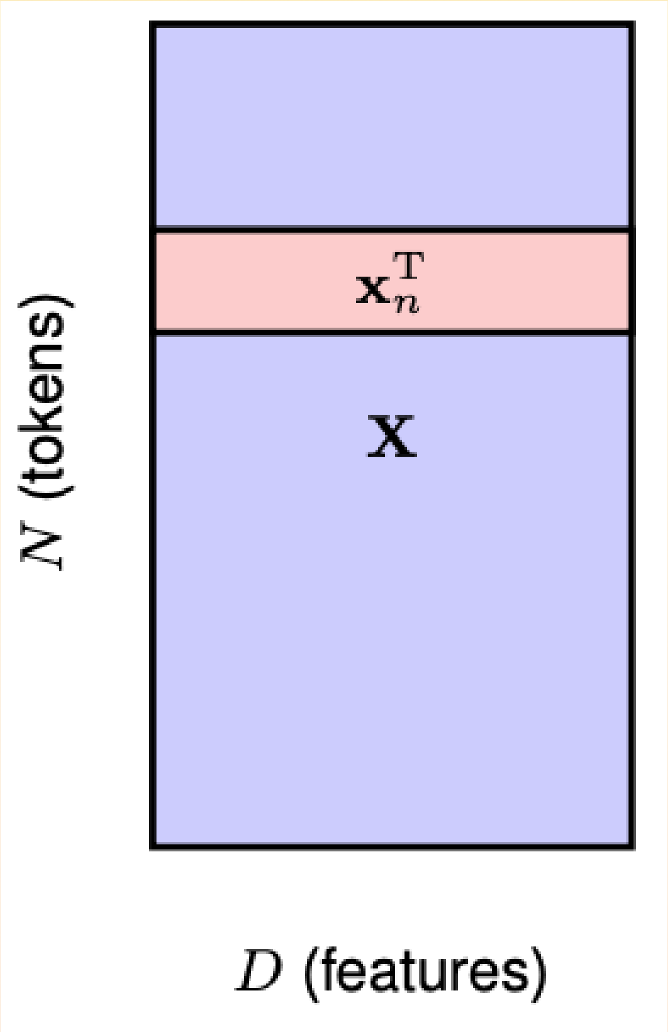
Image and treatment from Bishop and Bishop, Deep Learning Foundations and Concepts, Springer 2024

Transformer input

- Vectors $\mathbf{x}_n$ called tokens. The features are the x_{ni}
  - Data can be combined into a set of tokens, no need to have different architectures
- The data matrix $\mathcal{X}$ is one set of tokens. The full training dataset will be composed by many sets $\mathcal{X}$
- The transformer transforms $\mathcal{X}$ into a representation $\tilde{\mathcal{X}}$ with the same dimension:

$$\tilde{\mathcal{X}} = TransformerLayer[\mathcal{X}]$$

- Each transformer layer has two stages
  - Attention: mixes features across columns
  - Transforms features within each row



—
Image and treatment from Bishop and Bishop, Deep Learning Foundations and Concepts, Springer 2024

Attention

- Tokens $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_n$ to be transformed into $\mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_n$
- Each $\mathbf{y}_n$ should depend on all vectors $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_n$, not only on $\mathbf{x}_n$
- Simplest way (as usual) is linear combination using some attention weights

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

- $a_{mn} \geq 0$ (large value means input token influences output a lot)
- $\sum_{m=1}^{N} a_{mn} = 1$ (ensure that if an input is made more important, the importance of the others decreases)
- One set of weights per each vector of outputs!!!

Image and treatment from Bishop and Bishop, Deep Learning Foundations and Concepts, Springer 2024

Hard attention

- Movie example: choose which movie to watch
  - Attributes of each movie: key
  - The movie itself: value
- Search string with vector of desired attributes: query
  - Compare query vector with all keys, to find best match, then send the value (movie file) to user

Soft attention

- We use continuous values to match queries and keys, and use these values to weight the influence of value vectors on the outputs
  - Also ensures tranformer is differentiable
  - Called self attention because we use the same vector as query, key, and value
- $\mathbf{x}_n$ is a value vector used to create output tokens
- $\mathbf{x}_n$ also used as key vector for the input token n ("use the move itself instead of its characteristics")
- $\mathbf{x}_m$ is the query vector for $\mathbf{y}_m$, that needs to be compared with $\mathbf{x}_n$
- How similar are $\mathbf{x}_n$ and $\mathbf{x}_m$? Scalar product
- Constraints on attention weights : $a_{mn} = Softmax(\mathbf{x}_n \cdot \mathbf{x}_m)$ (here no probabilistic interpretation)

—
Image and treatment from Bishop and Bishop, Deep Learning Foundations and Concepts, Springer 2024

Make it learnable

- $\mathbf{Y} = Softmax[\mathbf{X}\mathbf{X}^T]\mathbf{X}$ is a fixed transformation, and each feature $x_{ni} has the same importance.
- Substitute X with $\tilde{\mathbf{X}} = \mathbf{X}\mathbf{U}$, with $\mathbf{U}$ is linear transform with learnable weights (as in usual neural network)

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

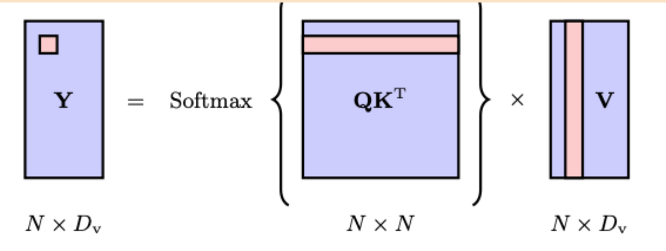
—
Image and treatment from Bishop and Bishop, Deep Learning Foundations and Concepts, Springer 2024

Make it asymmetrical

- $\mathbf{XUU}^T\mathbf{X}^T]$ is very symmetric
  - We want asymmetry: "Peugeot" must be strongly associated with "Car", but "Car" should be more weakly associated with "Peugeot" (there are many brands)
- Separate matrices for the query, key, and value

$$\mathbf{Y} = Softmax[\mathbf{QK}^T]\mathbf{V}$$

- Where each matrix has its own weights
  - $\mathbf{Q} = \mathbf{XW}^{(q)}$
  - $\mathbf{K} = \mathbf{XW}^{(k)}$
  - $\mathbf{V} = \mathbf{XW}^{(v)}$



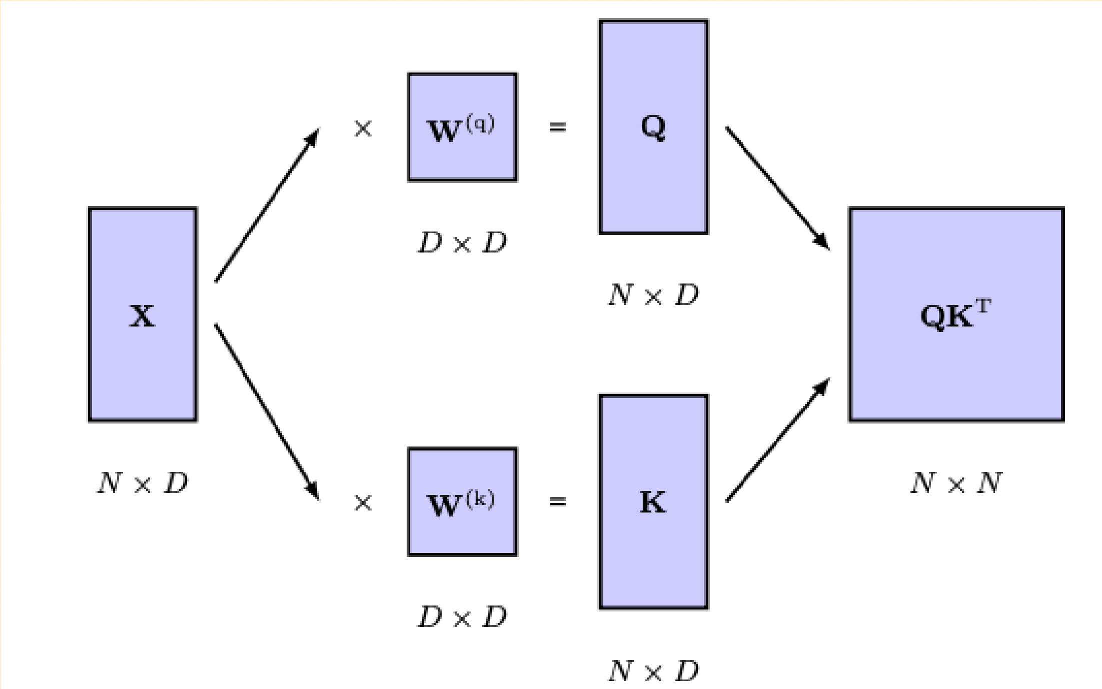


Image and treatment from Bishop and Bishop, Deep Learning Foundations and Concepts, Springer 2024

Information flow

- Normal networks multiply activations by fixed weights
  - If a weight is nearly zero, the network will learn that input or variable for all input vectors
- In Transformers, the activations are multiplied by data-dependent coefficients
  - If a coefficient is nearly zero for a certain input vector, the resulting path will ignore the incoming signal, and the output will not depend on it

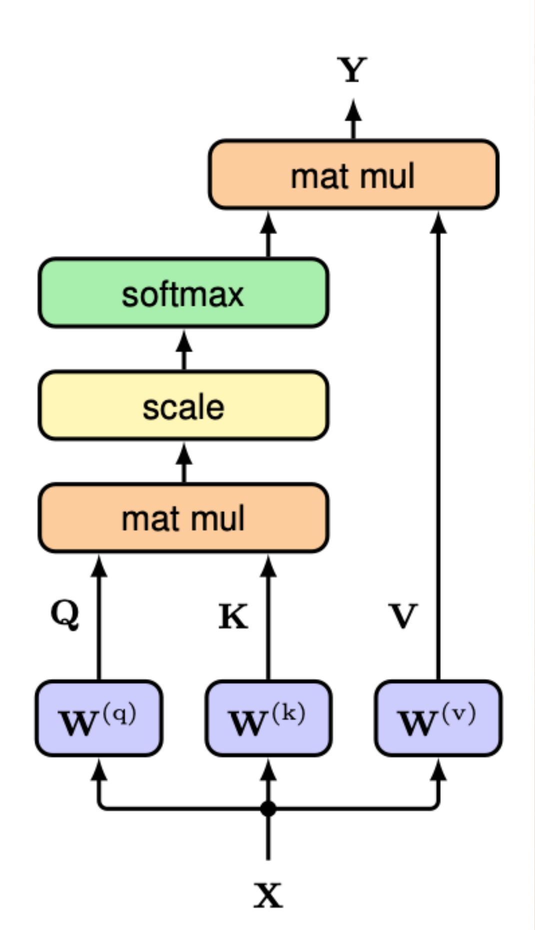
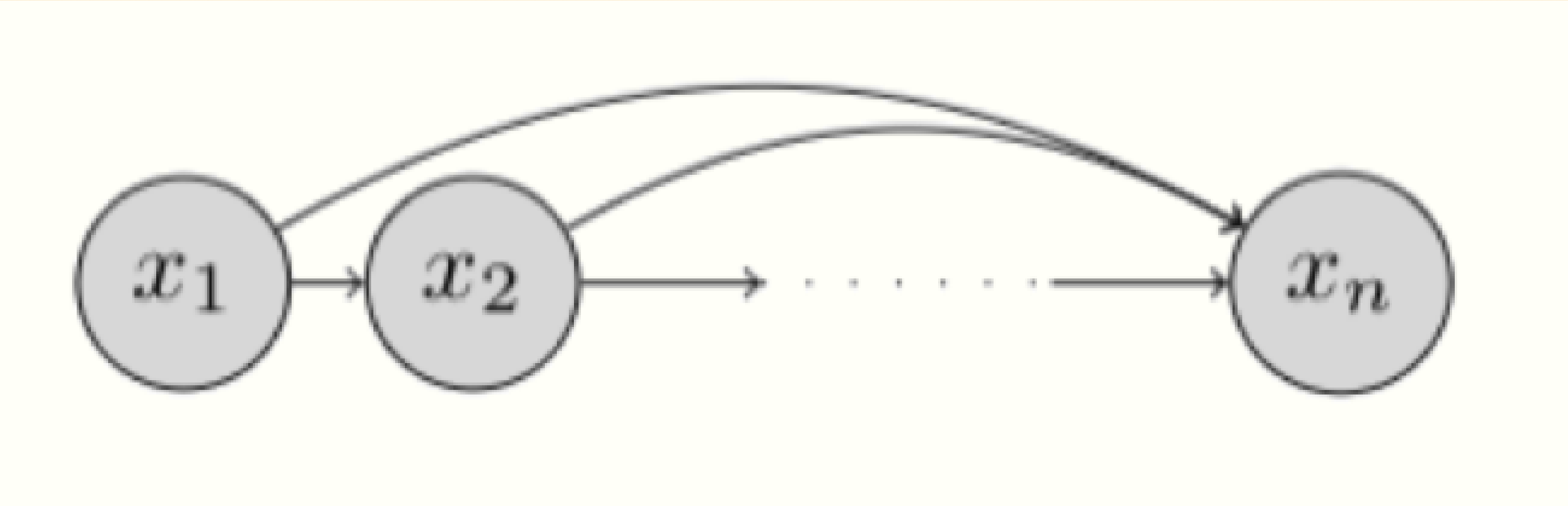


Image and treatment from Bishop and Bishop, Deep Learning Foundations and Concepts, Springer 2024

Gaussian Autoregressive models

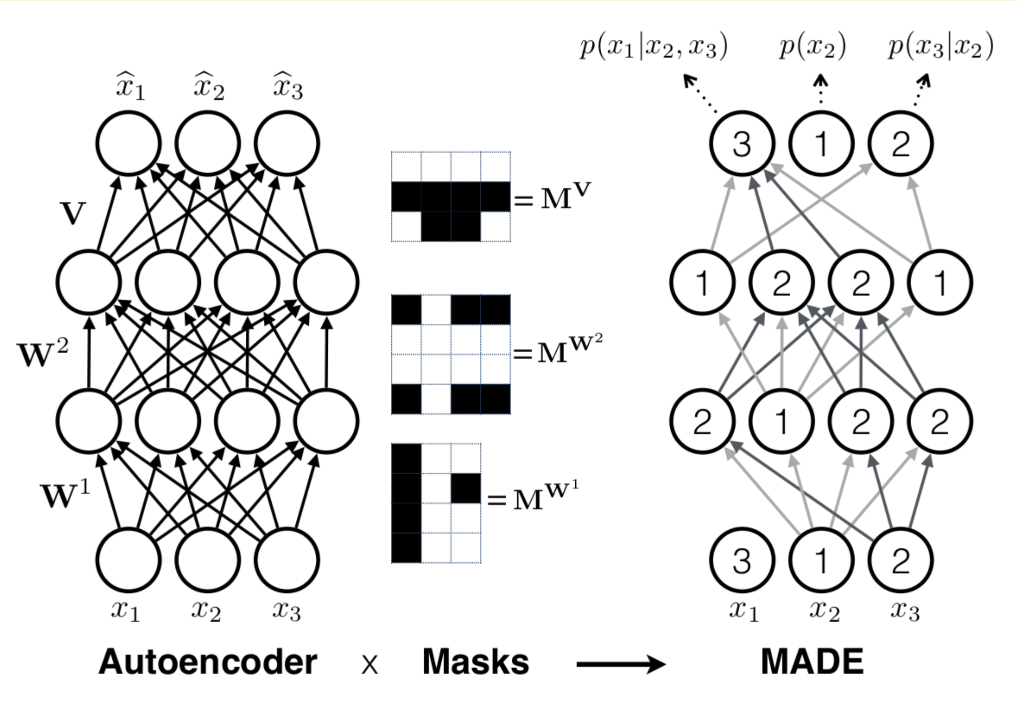
- Decompose a joint density into a product of conditional densities
  - Condition on \textit{previous} variables (time series, or lower-index coordinates for some ordering
  - Predicted value of features depend on past values of the same feature, rather than on other predictors
- Used for density estimation
  - Take some variable with some implicit ordering (e.g. tensor)
  - Output a mean and standard deviation for each element of the input, conditioned on previous elements
- In a sense, a Bayesian network

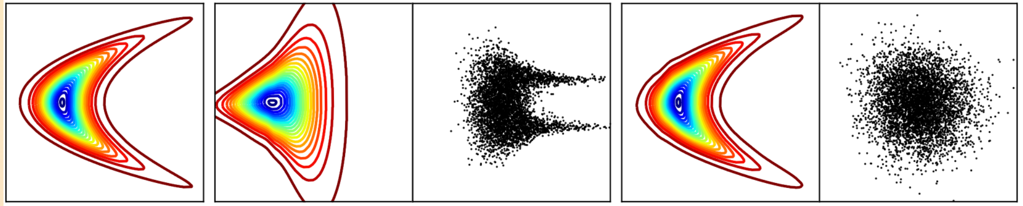


—
Image from https://deepgenerativemodels.github.io

MADE and MAF

- MADE (Masked Autoencoders for Distribution Estimation) 1502.03509
  - Vectorized architecture for density estimation based on autoencoders
  - Fast and reliable
  - Masked: deactivate inputs to enforce autoregressive model!
- Masked Autoregressive Flows (MAF), 1705.07057
  - If you stack several autoregressive models that each learn a conditional density, you obtain a normalizing flow!

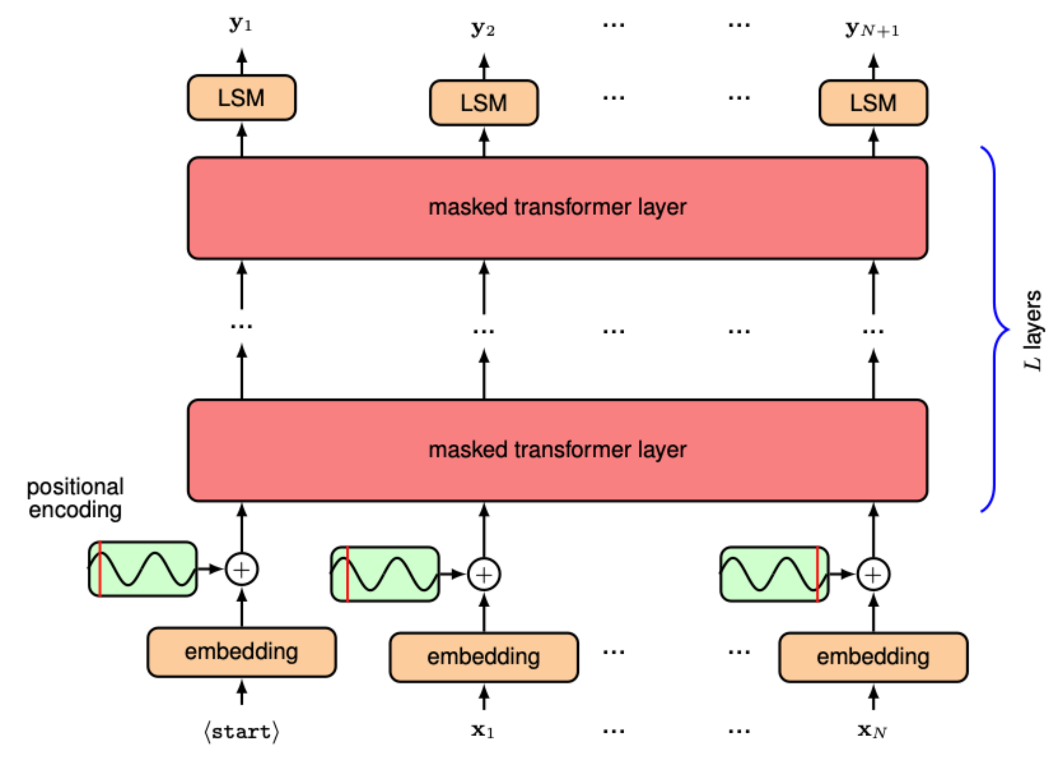




(a) Target density (b) MADE with Gaussian conditionals (c) MAF with 5 layers

Transformer

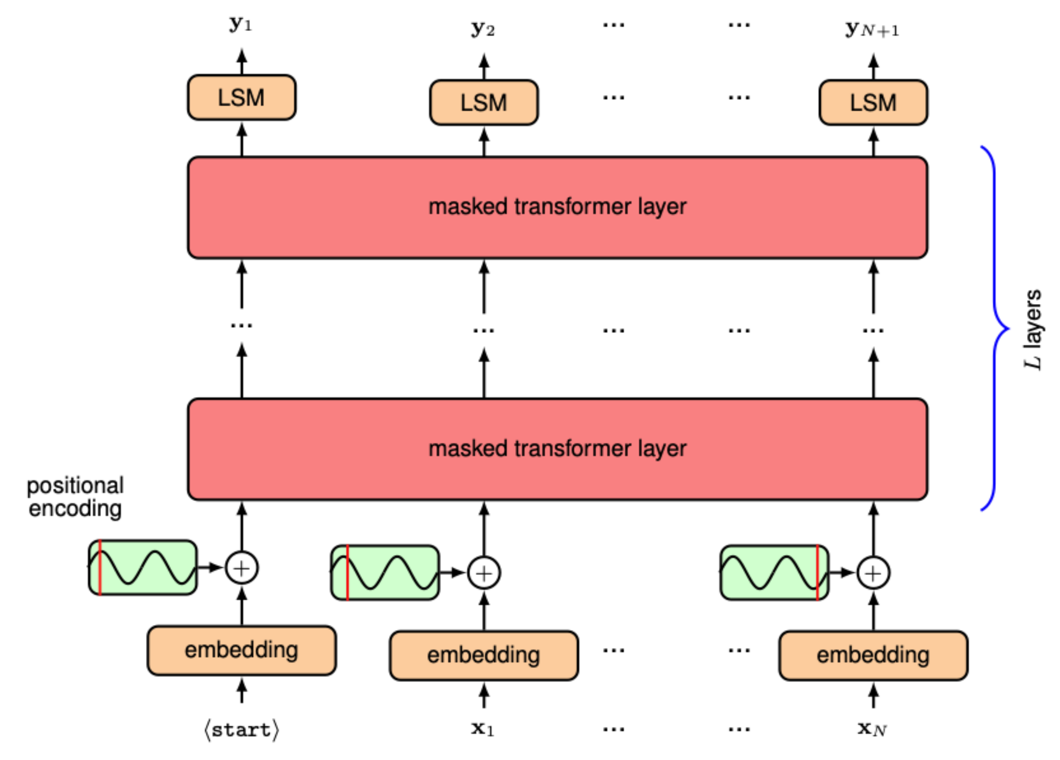
- Autoregressive model where the conditional distributions are expressed using a transformer network learned from data
- ...
- ...
  - ...



—
Image and treatment from Bishop and Bishop, Deep Learning Foundations and Concepts, Springer 2024

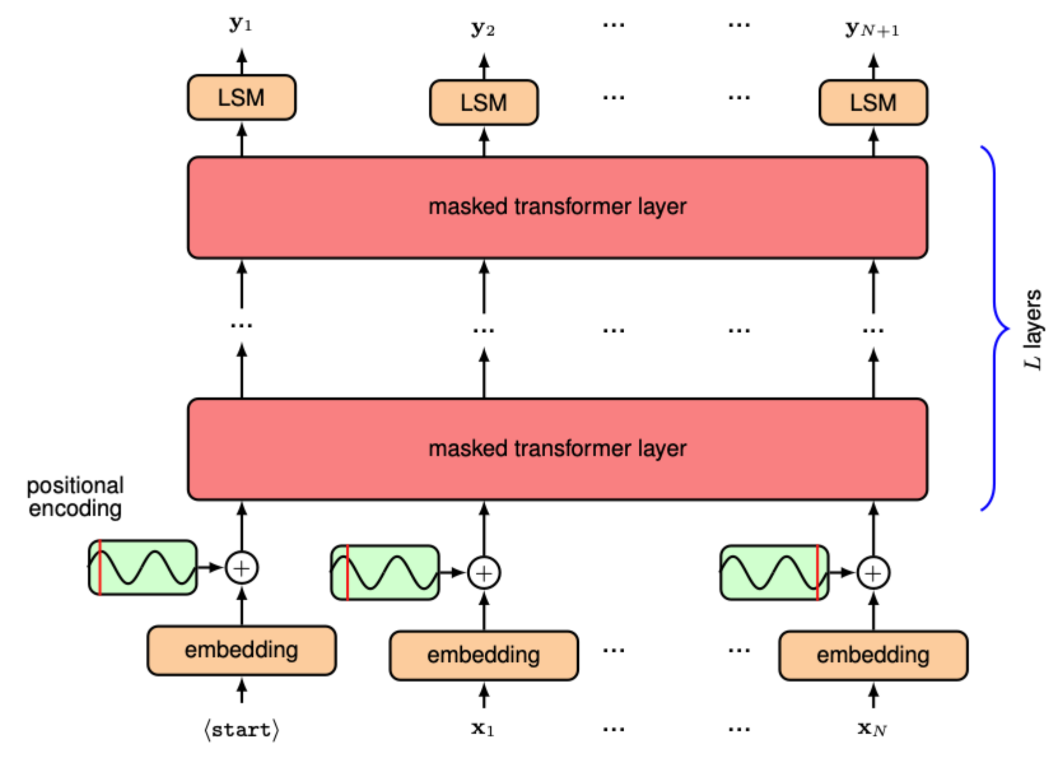
Pretrained Transformer

- Autoregressive model where the conditional distributions are expressed using a transformer network learned from data
- Train using lots of sequences of tokens
- Output: probability distribution, over the space of tokens, representing probability of the next token given the current token sequence
    - ...



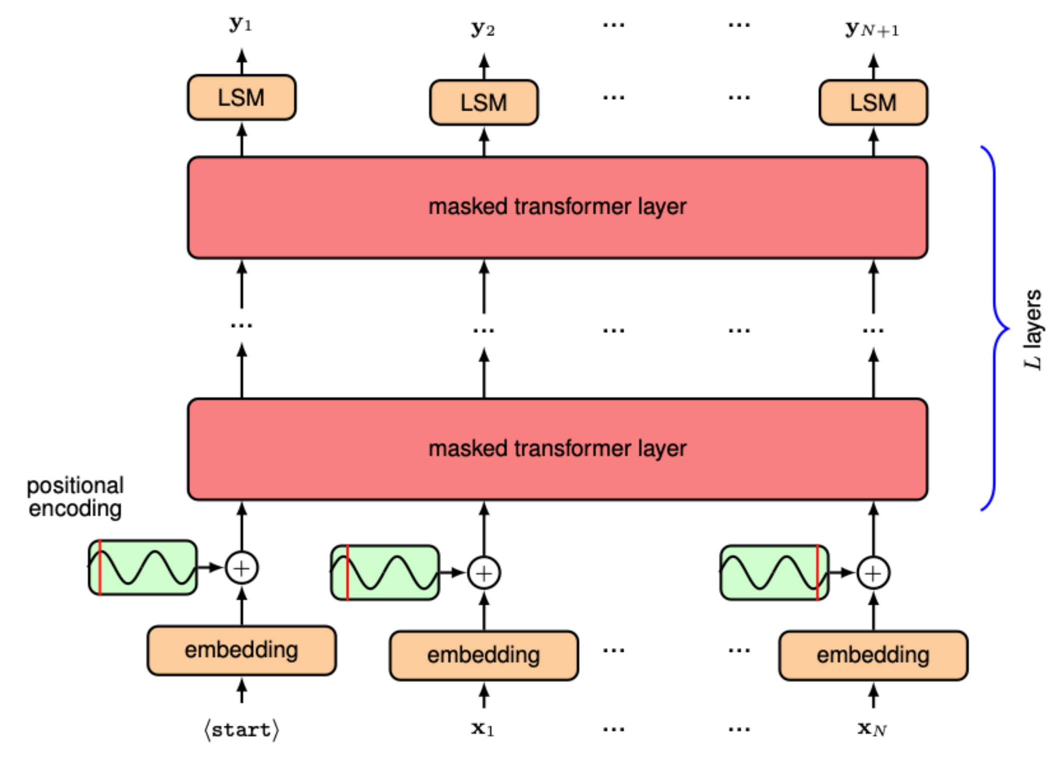
Generative

- Autoregressive model where the conditional distributions are expressed using a transformer network learned from data
- Train using lots of sequences of tokens
- Output: probability distribution, over the space of tokens, representing probability of the next token given the current token sequence
  - Use to generate sentences!



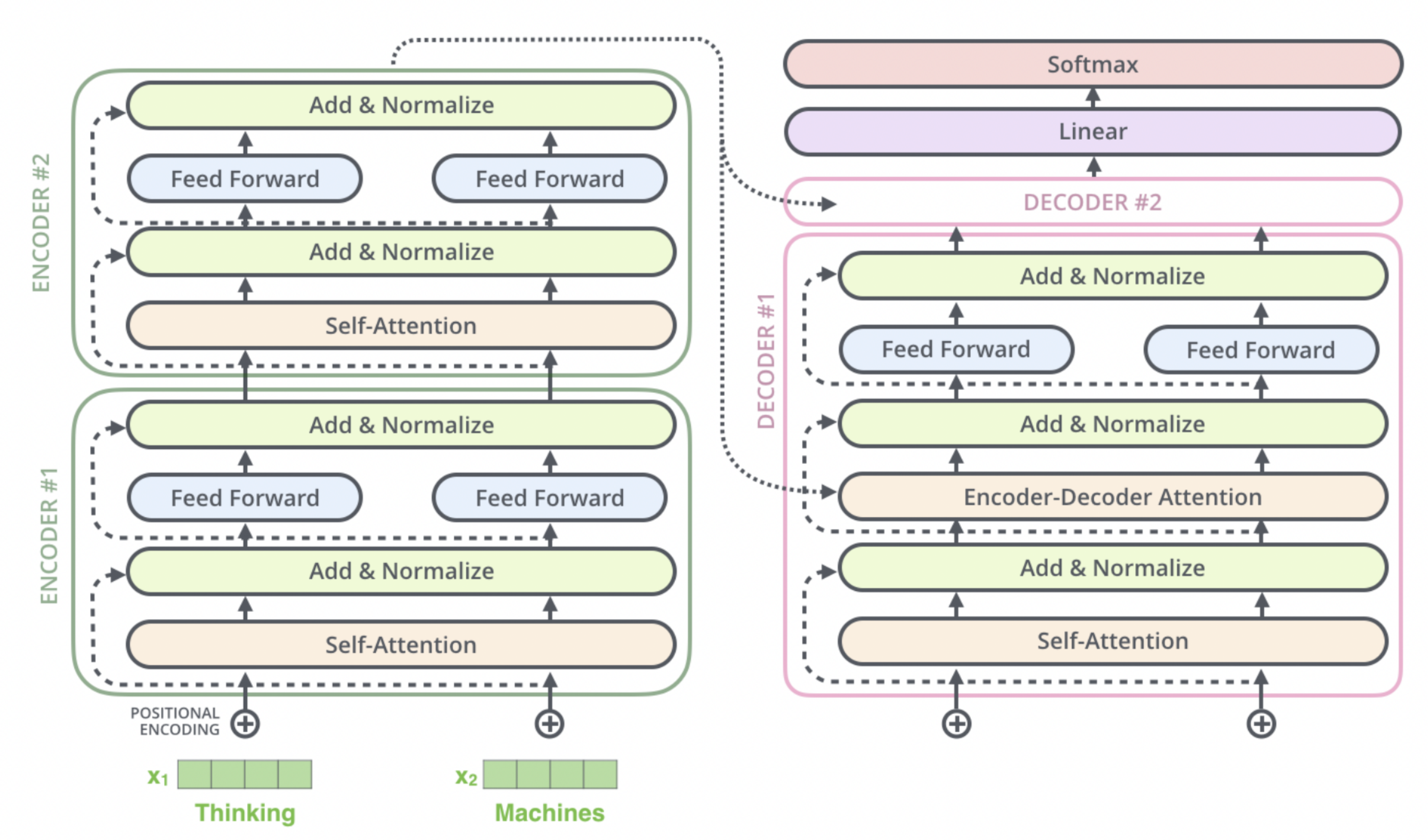
GPT (Generative Pretrained Transformer)

- Autoregressive model where the conditional distributions are expressed using a transformer network learned from data
- Train using lots of sequences of tokens
- Output: probability distribution, over the space of tokens, representing probability of the next token given the current token sequence
    - Use to generate sentences!



Transformers

- The engine behind GPT3



—

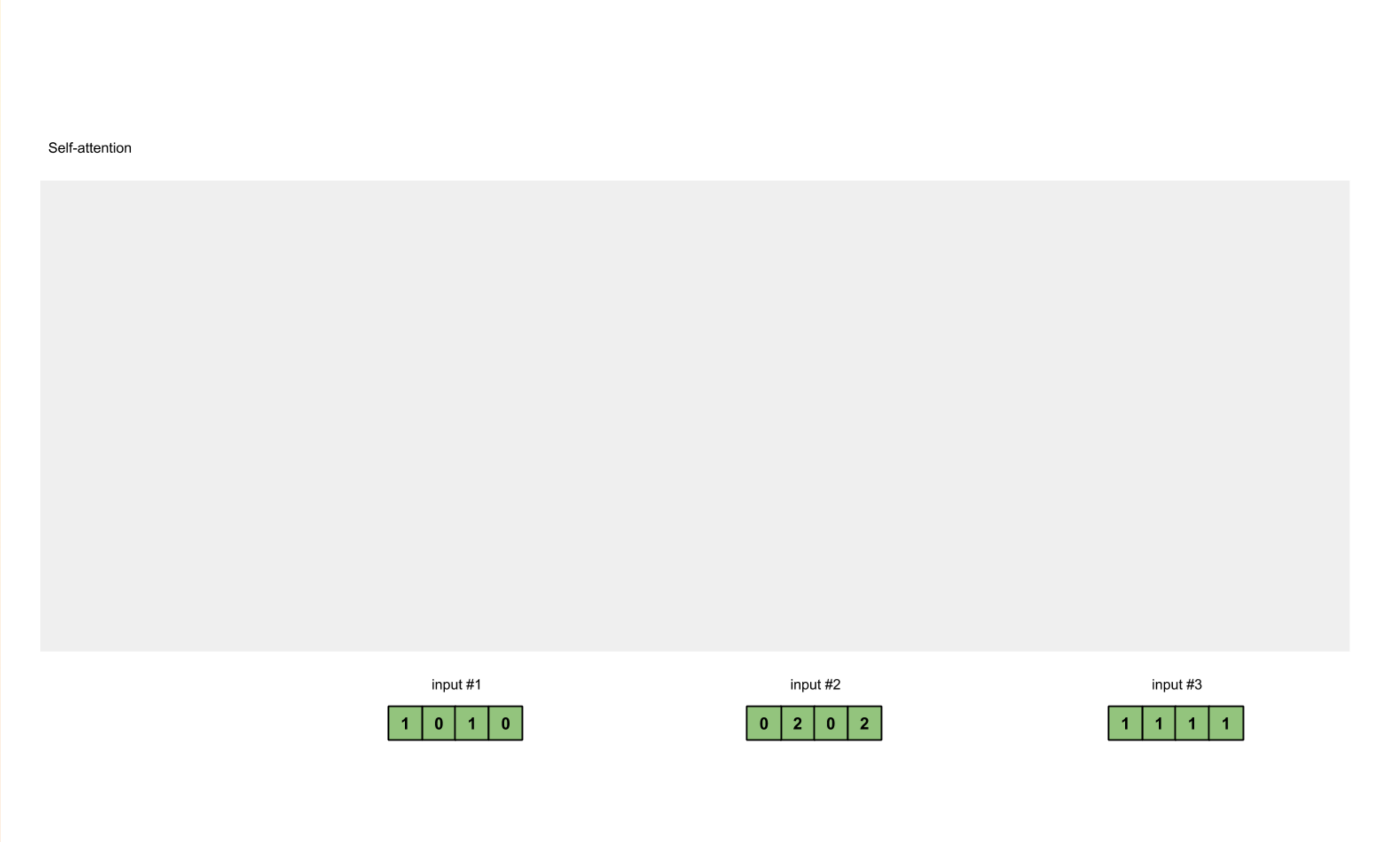
Image by Jay Alammar

Self-attention: a graphical illustration

- Capture dependencies and relationships within inputs
  - Mostly in natural language processing and computer vision
- N inputs, N outputs
  - Allow inputs to interact with eacho other and find out which ones to pay attention to
  - Output is an aggregate of interactions and attention scores
- Useful for:
  - Long-range dependencies: understand complex patterns and dependencies
  - Contextual understanding: assign appropriate weights to important elements in the sequence
  - Parallel computation: can be computed in parallel $\rightarrow$ efficient and scalable for large datasets.

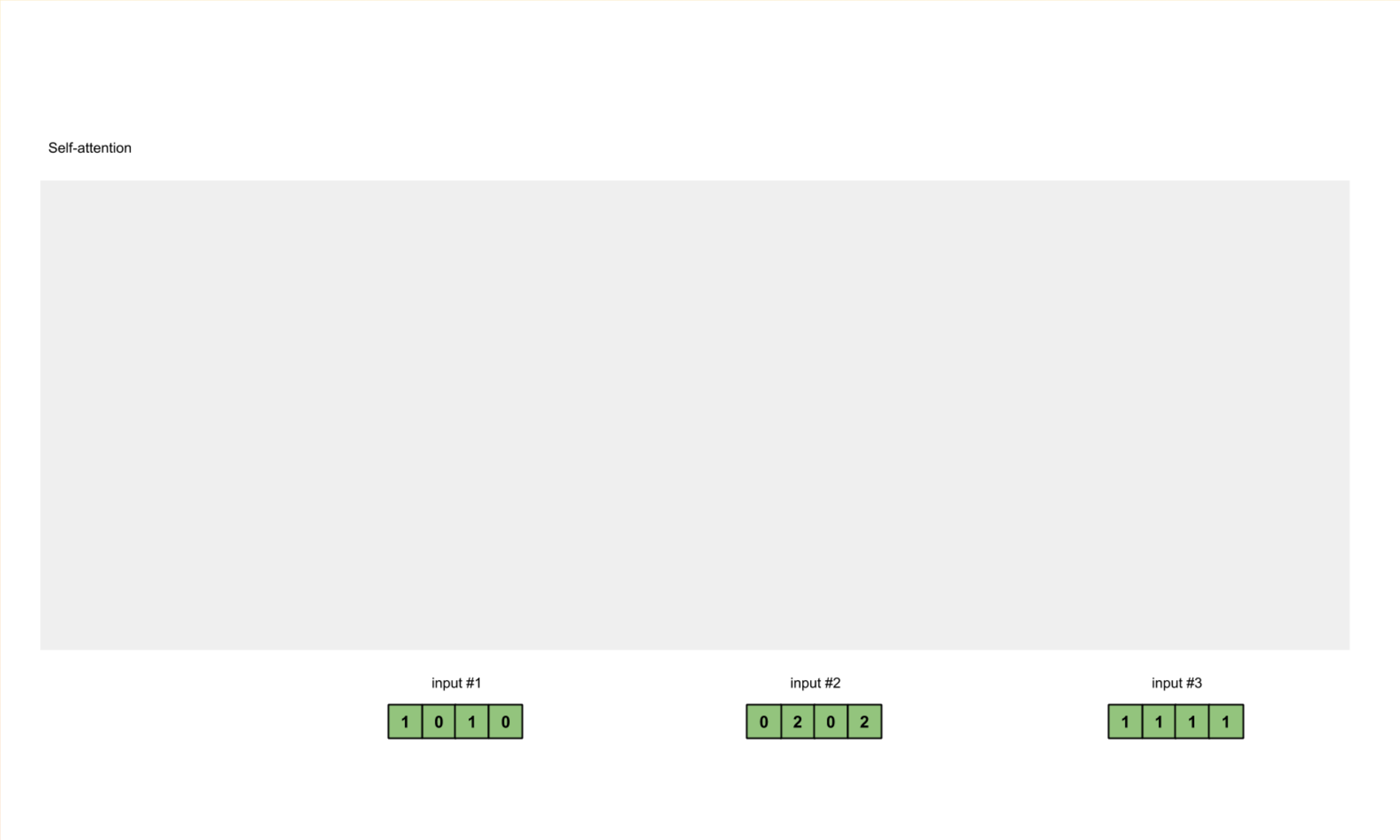
Self-attention: a graphical illustration

- Inputs (green) must be represented as: key (orange), query (red), value (purple)
  - Initially, by random reweighting of inputs themselves



Self-attention: a graphical illustration

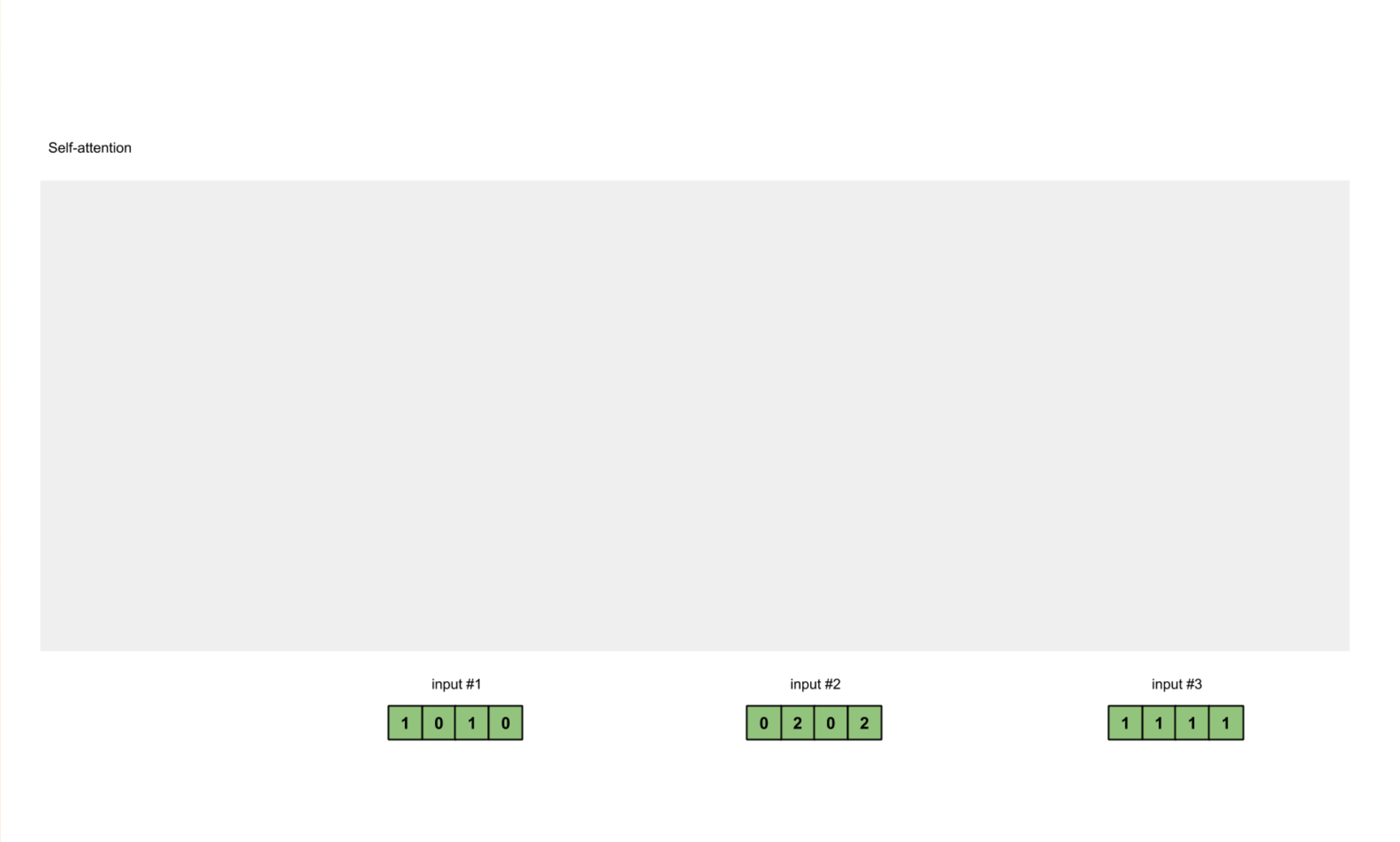
- Calculate attention score
  - Multiply (dot product) each query with all keys
  - For each query: N keys $\to N$ attention scores



—
Animation from towardsdatascience

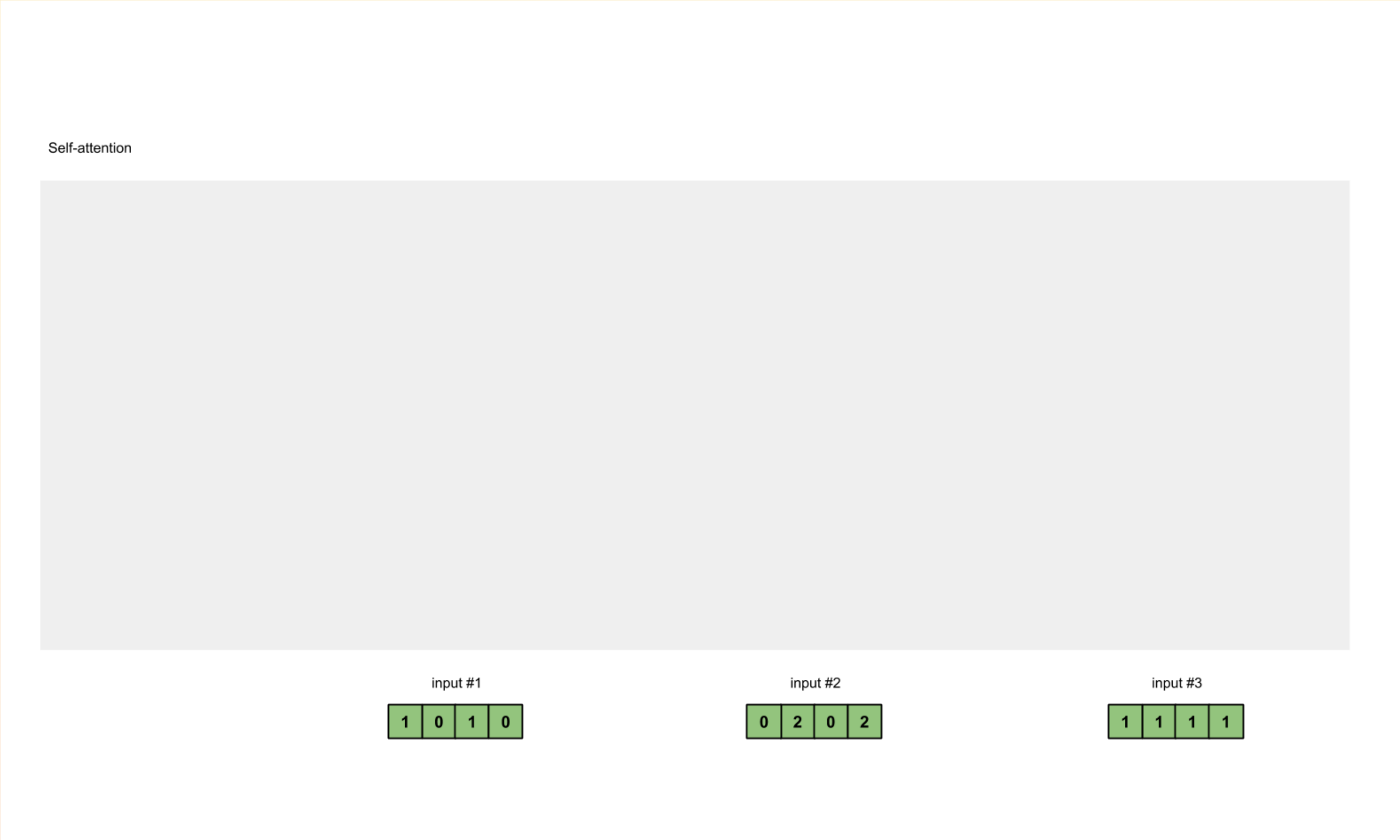
Self-attention: a graphical illustration

- Activation function (softmax) of attention scores



Self-attention: a graphical illustration

- Calculate alignment vectors (yellow), i.e. weighted values
  - Multiply each attention score (blue) by its value (purple)
- Sum alignment vectors to get input for output 1, repeat for 2 and 3

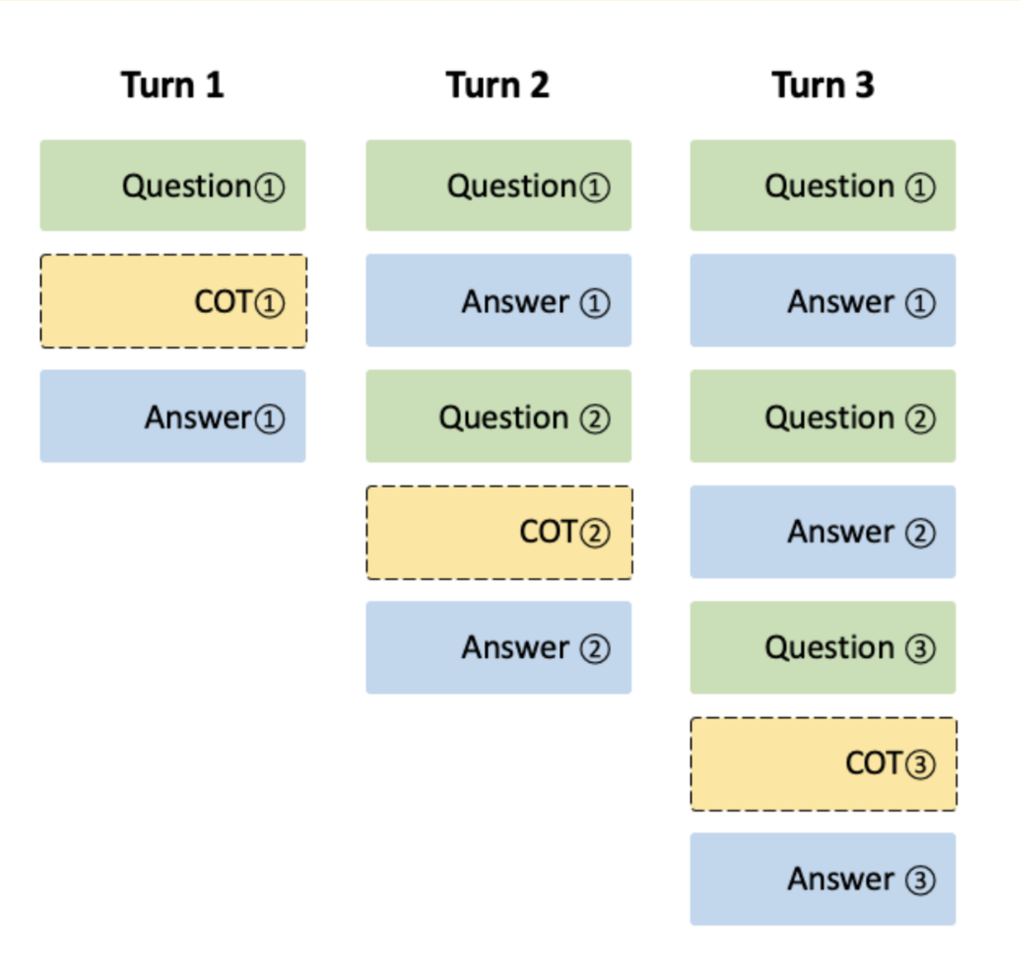


—
Animation from towardsdatascience

DeepSeek: improvements by software and hardware codesign

- Innovative optimization of the computing infrastructure
    - 5 millions to fully train, vs the hundreds of millions quoted e.g. by OpenAI
    - about 2000 GPU training time instead of tens of thousands
- Chain-of-Thought model to autocorrect the answer before providing it to the user

Lastly, we emphasize again the economical training costs of DeepSeek-V3, summarized in Table 1, achieved through our optimized co-design of algorithms, frameworks, and hardware. During the pre-training stage, training DeepSeek-V3 on each trillion tokens requires only 180K H800 GPU hours, i.e., 3.7 days on our cluster with 2048 H800 GPUs. Consequently, our pre-training stage is completed in less than two months and costs 2664K GPU hours. Combined with 119K GPU hours for the context length extension and 5K GPU hours for post-training, DeepSeek-V3 costs only 2.788M GPU hours for its full training. Assuming the rental price of the H800 GPU is $2 per GPU hour, our total training costs amount to only $5.576M. Note that the aforementioned costs include only the official training of DeepSeek-V3, excluding the costs associated with prior research and ablation experiments on architectures, algorithms, or data.



—
Images from deepkeek API guide and 2412.19437

Next: Exercise on transformers!