

Title: Transformers

Speakers: Mohamed Hibat Allah

Collection: Navigating Quantum and AI Career Trajectories: A Beginnerâ€™s Mini-Course on Computational Methods and their Applications

Date: May 24, 2024 - 9:30 AM

URL: <https://pirsa.org/24050087>

Lecture 3: Transformers

Energy-based model
(Intractable)

Restricted Boltzmann Machines (RBM)

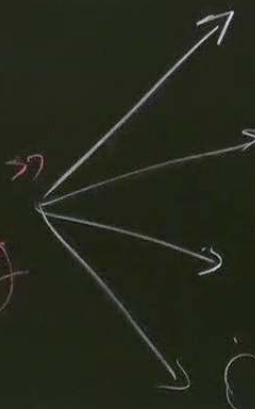
Recurrent Neural Networks (RNN)

Transformers

Other architectures (Not covered in this mini-course)

Last time

Generative modeling



Outlines

- ① Attention mechanism
- ② Transformer architecture
- ③ Transformer Wave function

① Attention mechanism

Most famous paper:

↳ "Attention is All you need" →

Vaswani et al (2017)

* Why it's interesting?

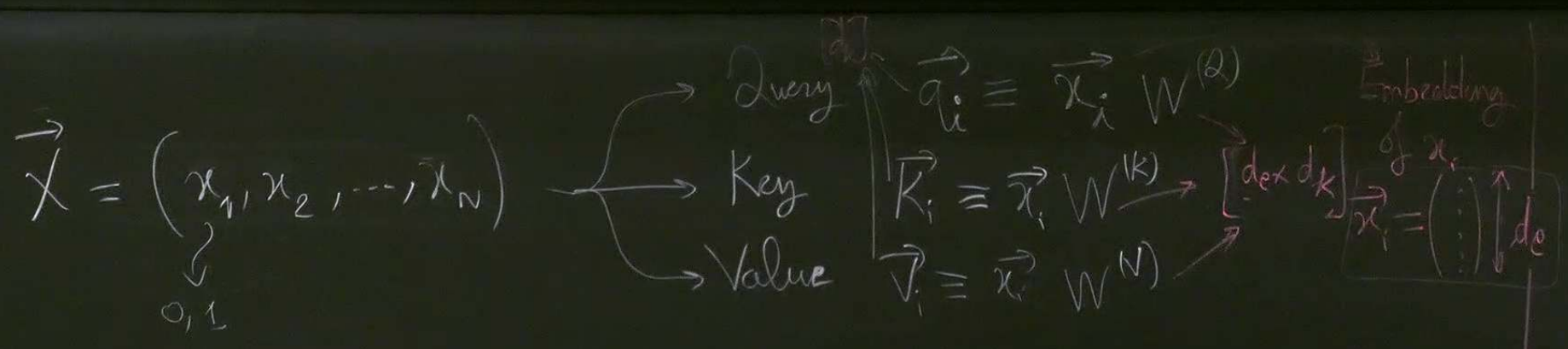
→ Enabled state of the art results in Natural Language processing (NLP)

→ Transformer is the basic building ^{block} of ChatGPT, GPT 4, GPT 4o

GPT = Generative Pre-trained Transformer.

based on the
"attention
mechanism"

→ Other architectures (Not covered in this)



"↓" → $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$
 "↑" → $\begin{pmatrix} 0 \\ 1 \end{pmatrix}$

Example of
Youtube

- ↳ Query: Text on search bar.
- ↳ Key: Videos, titles & descriptions
- ↳ Value: Best video matches.

→ We can use $\vec{q}_i, K_i, \vec{v}_i$ to compute self-attention.

$$\vec{A}_i = \sum_{j=1}^N \underbrace{\alpha(\vec{q}_i, K_j)}_{\text{correlation between site "i" and "j"}} \cdot \vec{v}_j$$

Attention that has information about the most relevant on " x_i "

«Correlation between site "i" and "j"»

«Attention of x_i on x_j »

Solves the vanishing gradient problem in RNNs

attention
mechanism

« I live in France, hence I speak French »

$$\sum_{j=1}^N \alpha(\vec{q}_i, \vec{K}_j) = 1$$

$$\alpha(\vec{q}_i, \vec{K}_j) = \text{Softmax} \left(\frac{\vec{q}_i \cdot \vec{K}_j}{\sqrt{d_k}} \right)$$

For normalization purposes.

$$\vec{q}_i \cdot \vec{K}_j \in \mathbb{R}$$
$$\vec{q}_i \cdot \vec{R} = \begin{pmatrix} \vdots \\ \vdots \end{pmatrix} \uparrow^N$$

mechanism

"I live in France, hence I speak French"

$$\alpha(\vec{q}_i, k_j^+) = \text{Softmax} \left(\frac{\vec{q}_i \cdot \vec{R}_j^+}{\sqrt{d_k}} \right)$$

For normalization purposes

$$\sum_{j=1}^N \alpha(\vec{q}_i, k_j^+) = 1$$

$$\vec{q}_i \cdot \vec{R}_j \in \mathbb{R}$$

$$\vec{q}_i \cdot \vec{R} = \begin{pmatrix} \vdots \\ \vdots \\ \vdots \end{pmatrix} \begin{matrix} \uparrow \\ \uparrow \\ \uparrow \end{matrix} N$$

For normalization purposes

$$[\mathbf{d}_k] \leftarrow \vec{A}_i = \sum_{j=1}^N \text{Softmax} \left(\frac{\vec{q}_i \cdot \mathbf{K}^T}{\sqrt{d_k}} \right)_j \vec{V}_j$$

$$[\mathbf{N} \times \mathbf{d}_k] \leftarrow \mathbf{A} = \underbrace{\text{Softmax} \left(\frac{\mathbf{Q} \cdot \mathbf{K}^T}{\sqrt{d_k}} \right)}_{\mathbf{N} \times \mathbf{N}} \cdot \mathbf{V}$$

→ Taken advantage of GPU parallelization power

↳ value - Best video matches.

* Positional encoding

$$\vec{X} = \overset{x_1}{I} \overset{x_2}{m} \overset{x_3}{not} \overset{x_4}{sad}, \overset{x_5}{I'm} \overset{x_6}{happy}$$

$$\vec{X} = I'm \ not \ happy, \ I'm \ sad$$

Idea $\vec{x}_i + \underbrace{\vec{PE}(i)}_{(d)}$ $\rightarrow \vec{x}'_i$

on λ_i

$$\vec{PE}(i)_{2j} = \sin\left(\frac{\lambda}{10000^{2j}/de}\right)$$

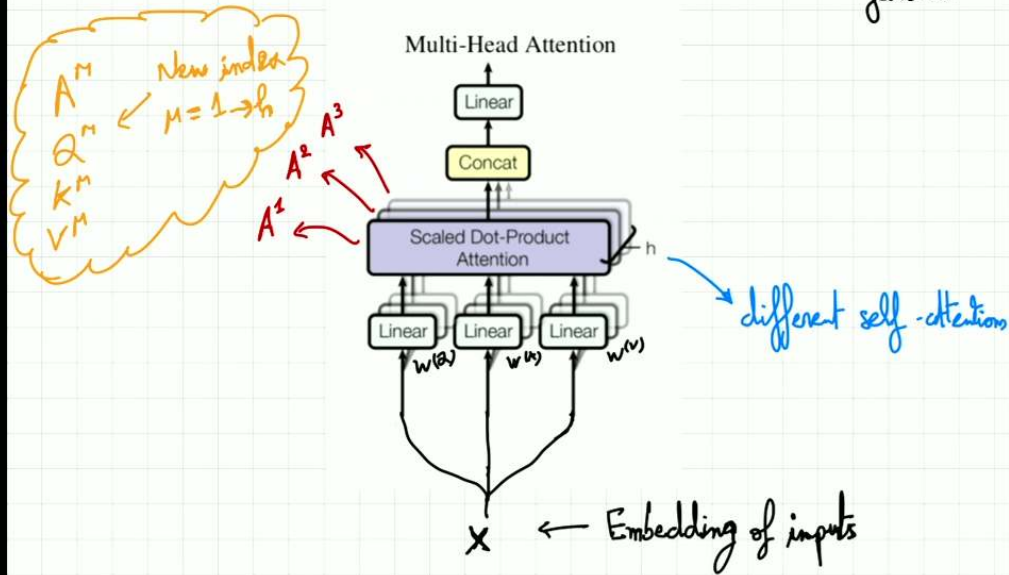
$$\vec{PE}(i)_{2j+1} = \cos\left(\frac{\lambda}{10000^{2j}/de}\right)$$

Max sequence length

↳ Nearby positions have similar PE at particular component $\langle k \rangle$

↳ Each component of the PE has a different wavelength. → short-long range dependencies.

* Multi-head Attention: allows to learn attention about different features



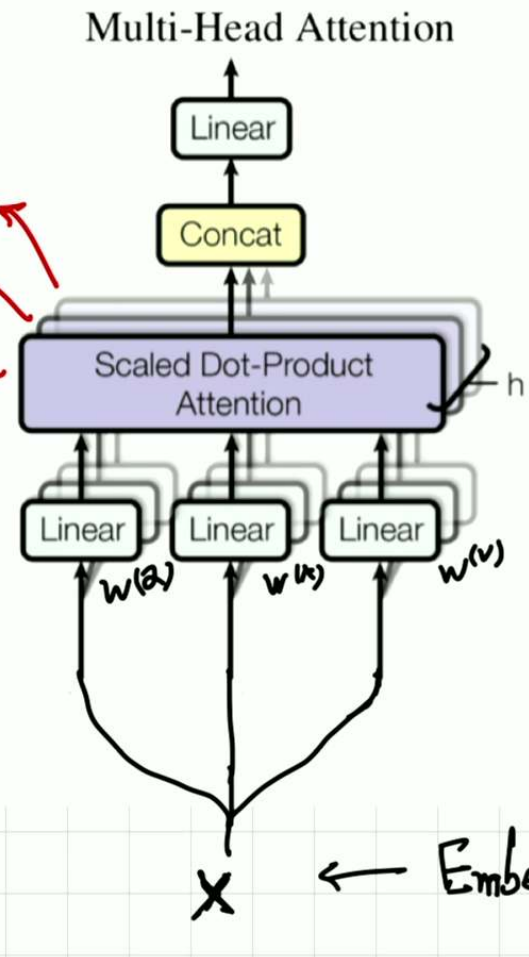
Example: Group of people → Character, Job, Age, Gender...
 Spins → Correlation, magnetization, ...

In equation:

$$\text{Multi-head Attention} = \text{Concat}(A^1, A^2, \dots, A^h) W^{(o)}$$

features

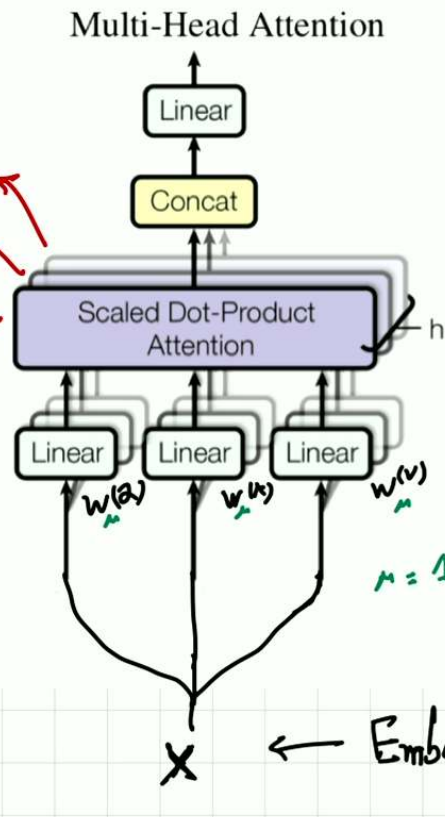
A^M
 Q^M
 K^M
 V^M
New index
 $M=1 \rightarrow h$



different self-attentions

A^m
 Q^m
 K^m
 V^m

New index $m=1 \rightarrow h$



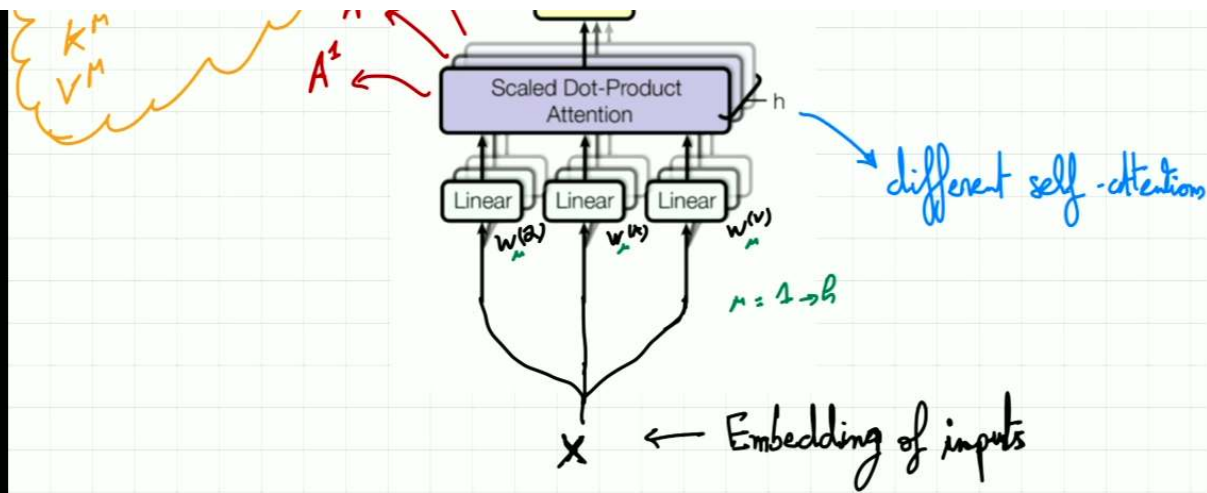
A^1
 A^2
 A^3

different self-attention

Example:

Group of people \rightarrow Character, Job, Age, Gender...

Since people may st. in.

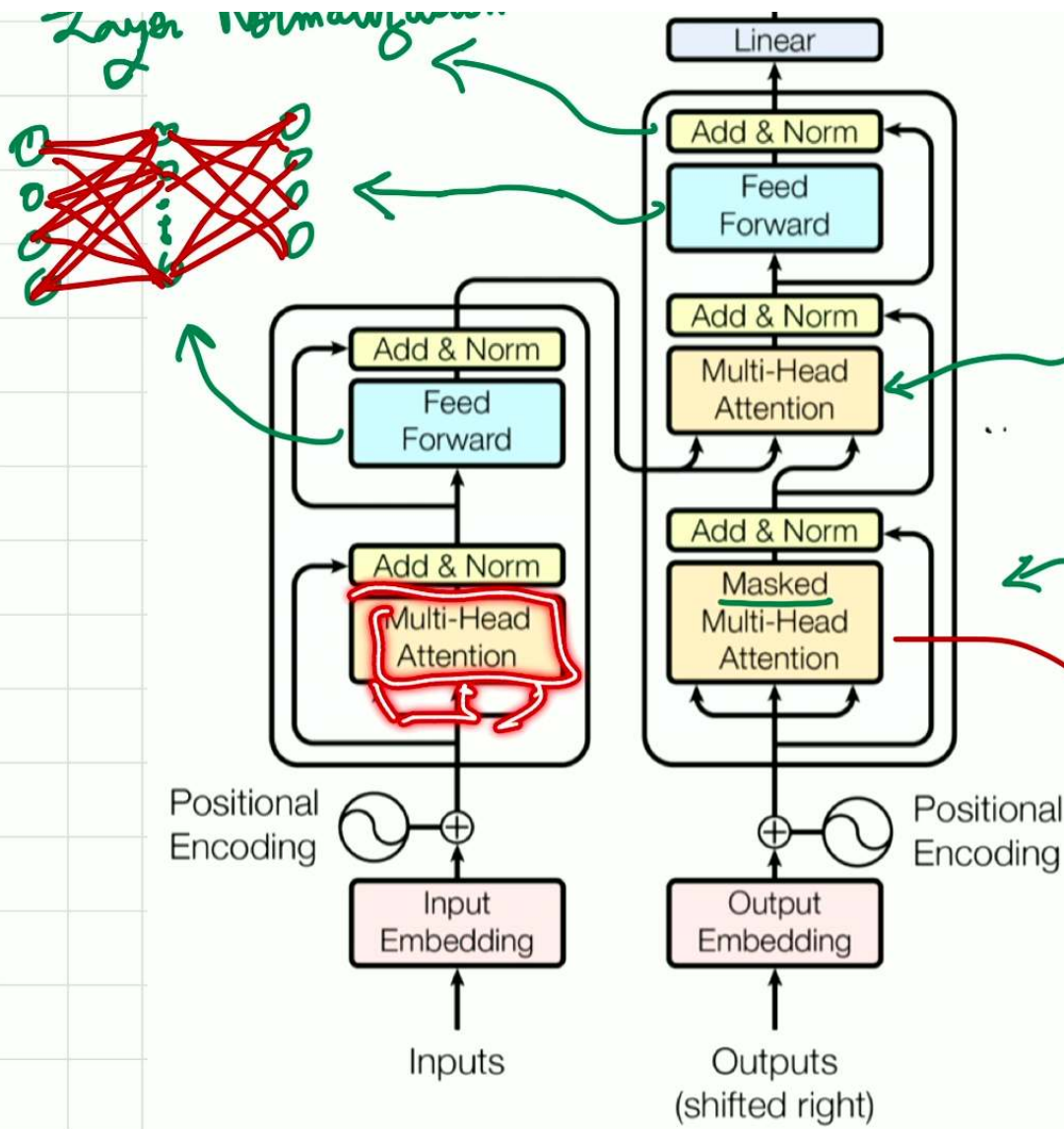


Example: Group of people \rightarrow Character, Job, Age, Gender...
 Spins \rightarrow Correlation, magnetization, ...

In equation:

$$\text{Multi head Attention} = \text{Concat}(A^1, A^2, \dots, A^h) W^{(o)}$$

Learnable weights



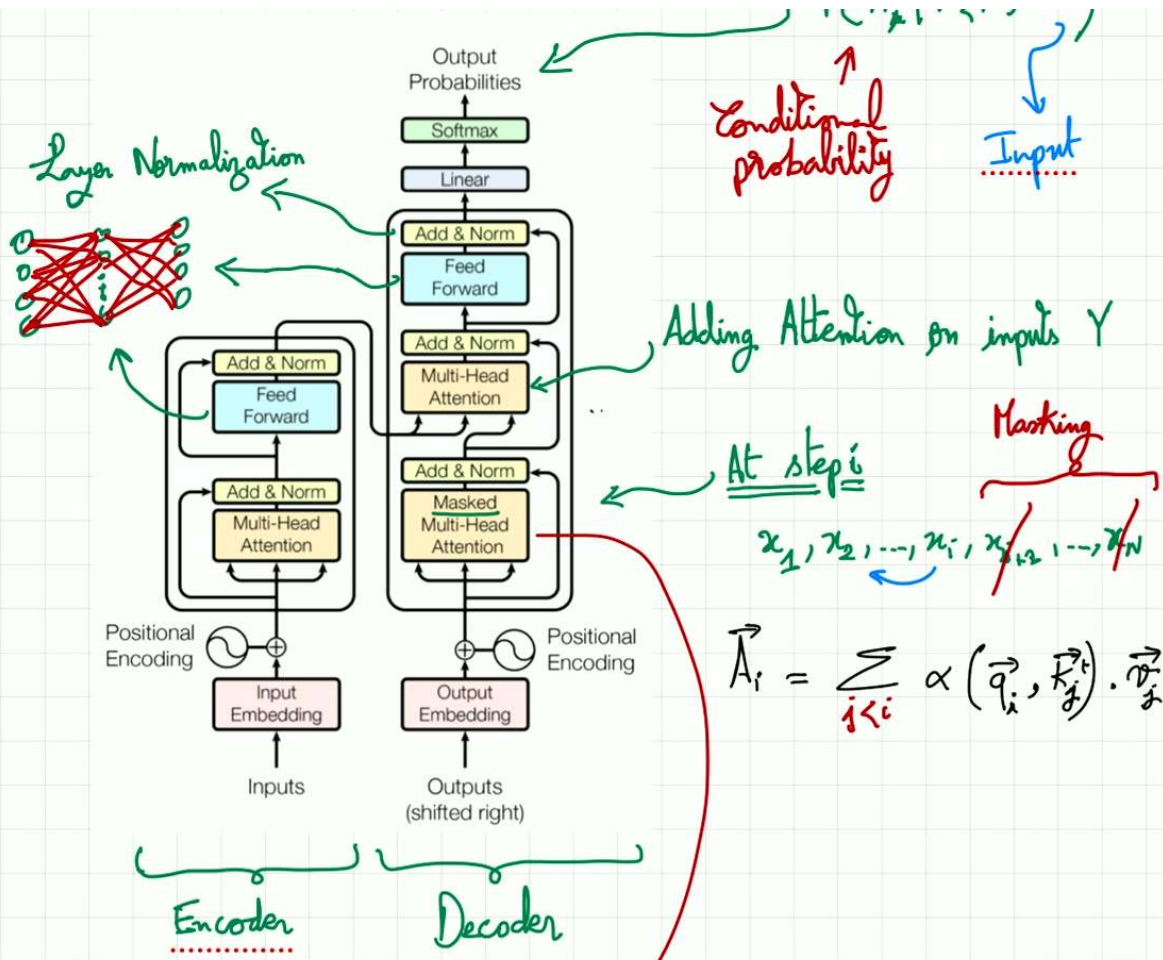
proceeding

.....

At step i

$x_1, x_2, \dots, x_i, x_{i+2}, \dots, x_N$

$$\vec{A}_i = \sum_{j < i} \alpha(\vec{q}_i, \vec{k}_j) \cdot \vec{v}_j$$



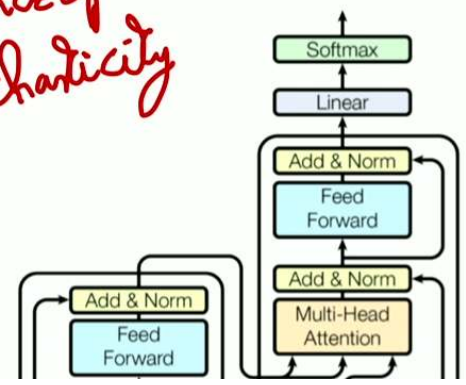
| GPT-3 |

175 B parameters
 $W^{(Q)}, W^{(K)}, W^{(V)}$
 $W^{(O)}, \dots$
96 Transformer layers

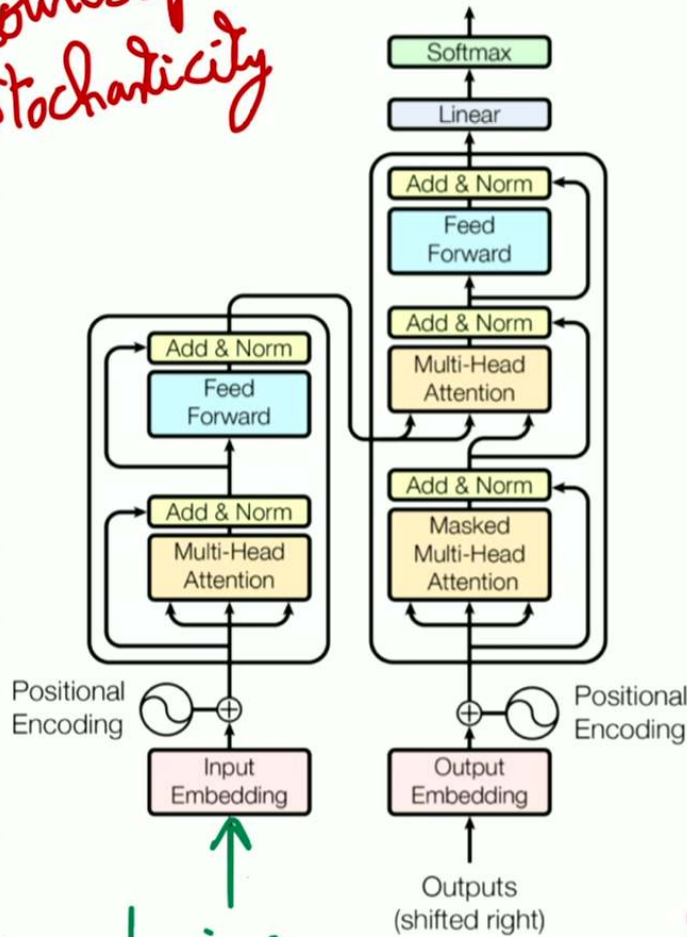
→ Training using massive
Dataset $D \sim 300$ B words

↓
Human feedback
(Reward)
to fine tune the
parameters

Source of Stochasticity ← $P(\text{Next word} | Y)$



Source of Stochasticity



Human feedback
(Reward)
to fine tune the
parameters

1) Training cost:
~ 4.6 million \$
to train.

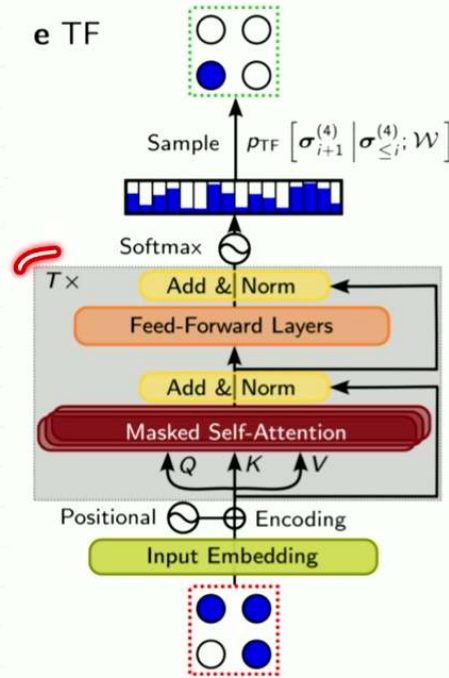
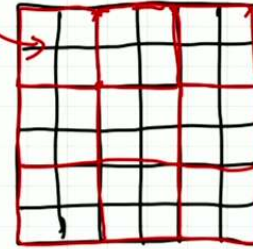


Sentence beginning
v

550 tons of CO₂ → Climate Change

(3) Transformer Wavefunctions

patch
eg: 4 spins



$$\Psi_{TF}(\sigma_1, \sigma_2, \dots, \sigma_N)$$

$$= \sqrt{P_{TF}(\sigma_1, \sigma_2, \dots, \sigma_N)}$$

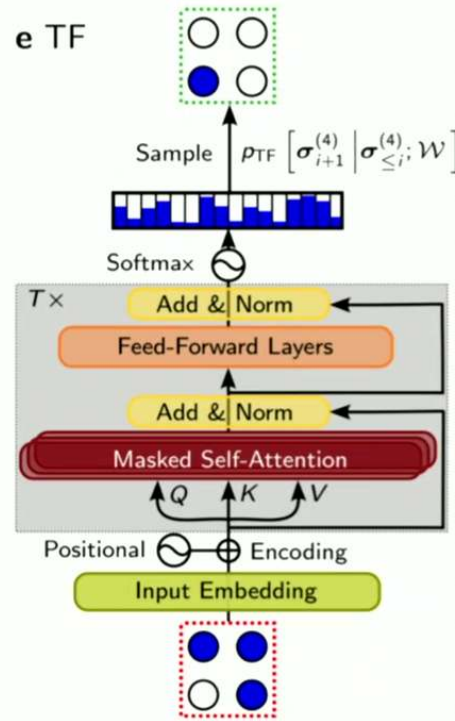
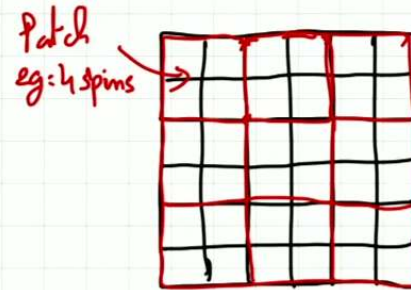
↳ For Stochastic
Hamiltonians

Cost function:

$$E = \langle \Psi_{TF} | \hat{H} | \Psi_{TF} \rangle$$

K. Sprague & S. Coish, Comm. Phys. 2024

③ Transformer Wavefunctions



$$\Psi_{TF}(\sigma_1, \sigma_2, \dots, \sigma_N)$$

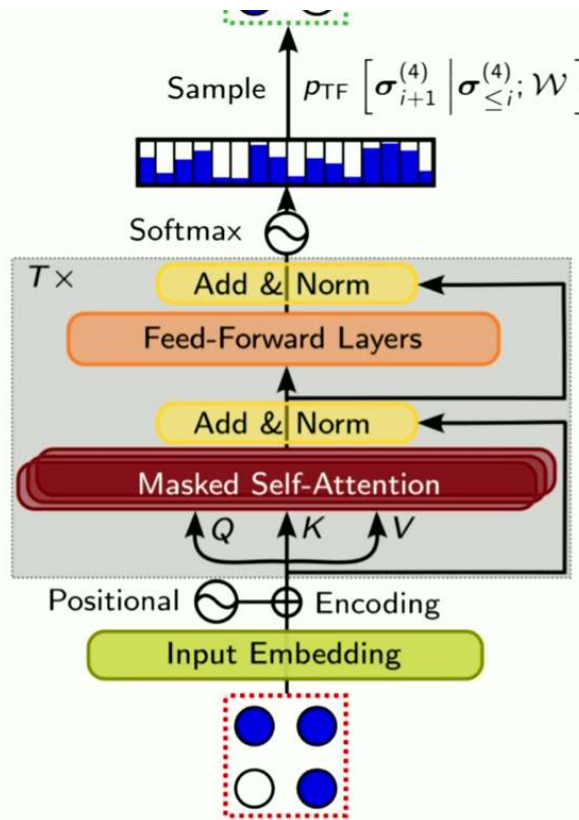
$$= \sqrt{P_{TF}(\sigma_1, \sigma_2, \dots, \sigma_N)}$$

↳ For Stoquastic Hamiltonians

Cost function:

$$E = \langle \Psi_{TF} | \hat{H} | \Psi_{TF} \rangle$$

K. Sprague & S. Coish, Comms Phys 2024



$$\Psi_{TF}(\sigma_1, \sigma_2, \dots, \sigma_N)$$

$$= \sqrt{P_{TF}(\sigma_1, \sigma_2, \dots, \sigma_N)}$$

For Stochastic
Hamiltonians

Cost function:

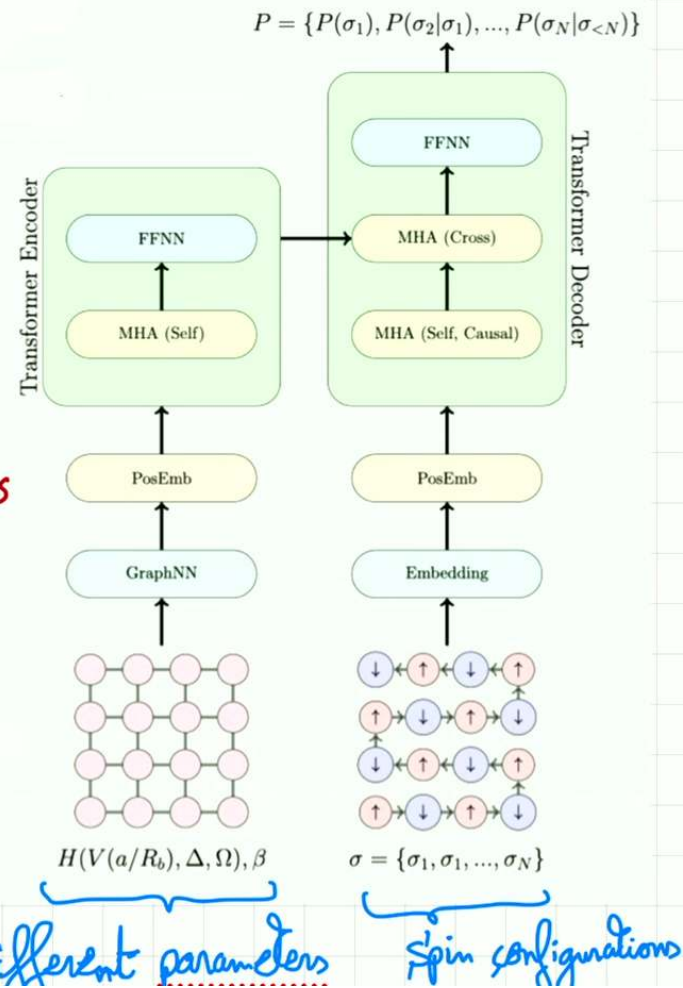
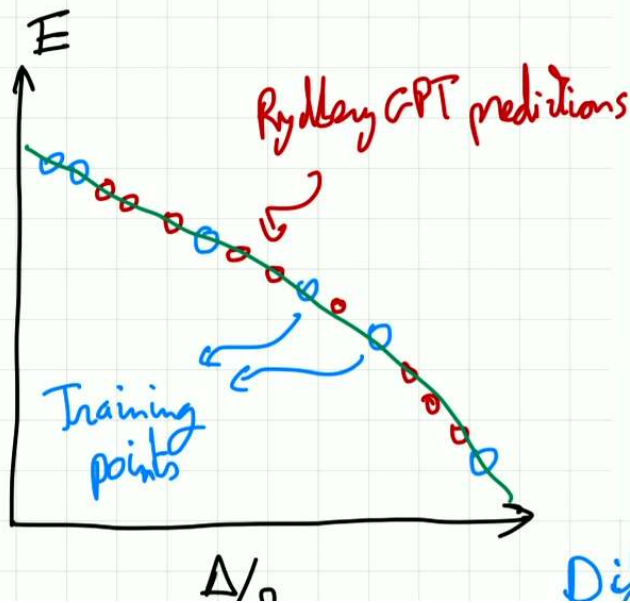
$$E = \langle \Psi_{TF} | \hat{H} | \Psi_{TF} \rangle$$

K. Sprague & S. Geisler, Comm Phys 2024

Another paper: "ArXiv:2310.05715"

*Rydberg atoms Hamiltonian:

$$H = \Omega \sum_i \sigma_i^x - \Delta \sum_i n_i + \sum_{i < j} V_{ij} n_i n_j$$



LLM Visualization

Home

Chapter: Overview

GPT-2 (small) nano-gpt GPT-2 (XL) GPT-3

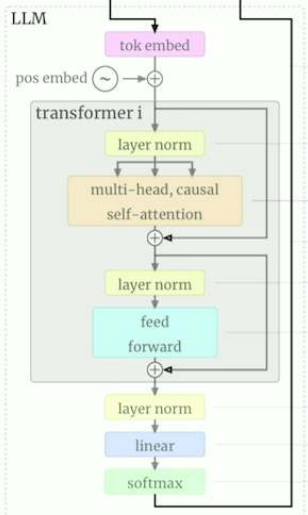
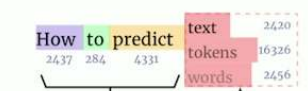
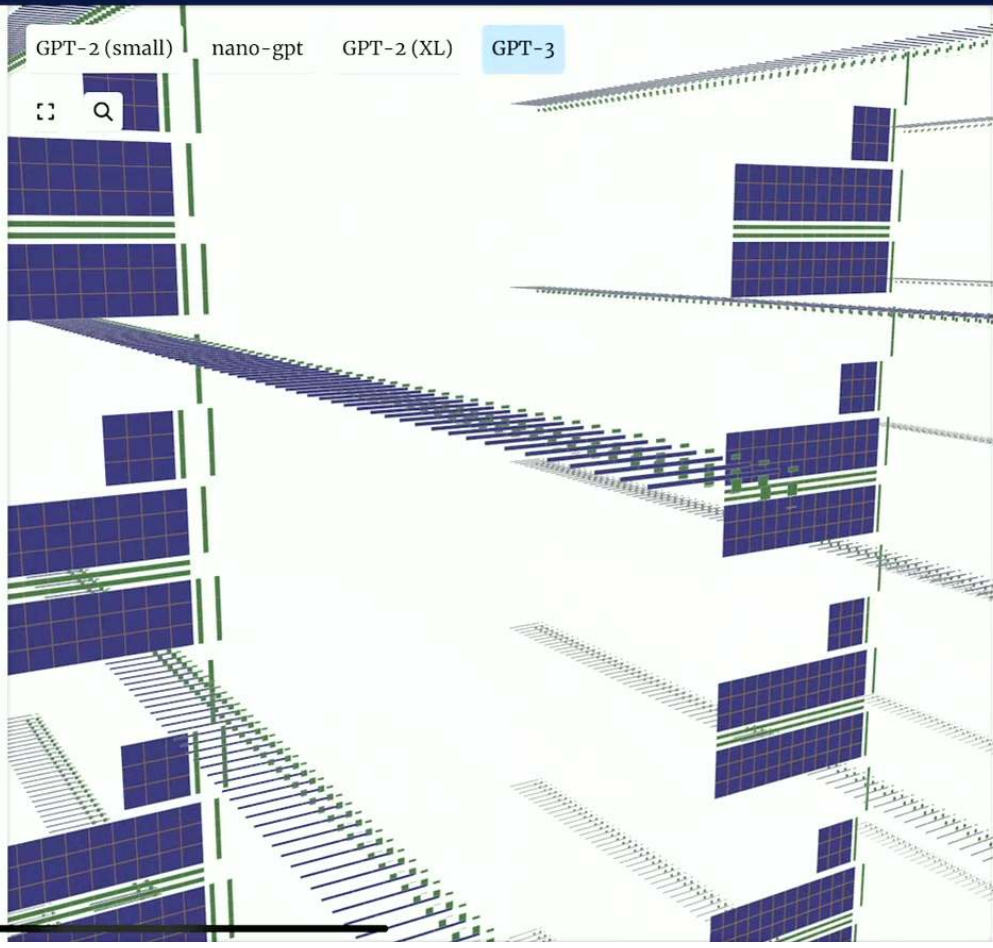


Table of Contents

- Intro
- Introduction
- Preliminaries
- Components
- Embedding
- Layer Norm
- Self Attention
- Projection
- MLP
- Transformer
- Softmax
- Output

Welcome to the walkthrough of the GPT large language model! Here we'll explore the model *nano-gpt*, with a

Continue Skip



Transformer wave functions

This notebook is based on code implemented on this repository: https://github.com/APRIQuOt/VMC_with_LPTFs based on this paper: <https://www.nature.com/articles/s42005-024-01584-y>

Cloning github repository:

```
1 !git clone https://github.com/mhibatallah/VMC_with_TF.git
```

```
Cloning into 'VMC_with_TF'...
remote: Enumerating objects: 96, done.
remote: Counting objects: 100% (96/96), done.
remote: Compressing objects: 100% (94/94), done.
remote: Total 96 (delta 45), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (96/96), 47.67 KiB | 185.00 KiB/s, done.
Resolving deltas: 100% (45/45), done.
```

Training Arguments:

```
L      (int)  -- Total lattice size (8x8 would be L=64).

Q      (int)  -- Number of minibatches per batch.

K      (int)  -- size of each minibatch.

B      (int)  -- Total batch size (should be Q*K).

NL00PS (int)  -- Number of loops within the off_diag_labels function. Higher values save ram and
                generally makes the code run faster (up to 2x). Note: you can only set this
```

proximate the gradients, more efficient but approximate.

the end of the output directory (inside a subfolder).

```
python L=16 steps=1000 --ptf patch=2x2 Nh=32 nhead=1 --rydberg V=7 delta=
```

```
tf', 'patch=2x2', 'Nh=32', 'nhead=1', '--rydberg', 'V=7', 'delta=1',
```

```
s/torch/nn/modules/transformer.py:306: UserWarning: enable_nested_tensor
```

```
is True, but self.use_nested_tensor is False because {why_not_sparse}
```

```
  {'L': 16, 'Q': 1, 'K': 256, 'B': 256, 'NLOOPS': 1, 'steps': 1000}
```

```
  {'L': 16, 'patch': '2x2', 'Nh': 32, 'dropout': 0.0, 'num_layers': 16}
```

```
  {'Lx': 4, 'Ly': 4, 'V': 7, 'Omega': 1, 'delta': 1, 'L': 16}
```

```
077 , Transformer energy variance= 471.61472
```

```
1 !python VMC_with_TF/train.py --train L=16 steps=1000 --ptf patch=2x2 Nh=32 nhead=8 --rydberg V=7 delta=1 Omega=1
```

```
... [ '--train', 'L=16', 'steps=1000', '--ptf', 'patch=2x2', 'Nh=32', 'nhead=8', '--rydberg', 'V=7', 'delta=1', 'Omega=1']  
/usr/local/lib/python3.10/dist-packages/torch/nn/modules/transformer.py:306: UserWarning: enable_nested_tensor is True, but self.use_  
warnings.warn(f"enable_nested_tensor is True, but self.use_nested_tensor is False because {why_not_sparsity_fast_path}")  
train {'L': 16, 'Q': 1, 'K': 256, 'B': 256, 'NLOOPS': 1, 'steps': 1000, 'dir': 'PTF', 'lr':  
model {'L': 16, 'patch': '2x2', 'Nh': 32, 'dropout': 0.0, 'num_layers': 2, 'nhead': 8, 'ref'  
hamiltonian {'Lx': 4, 'Ly': 4, 'V': 7, 'Omega': 1, 'delta': 1, 'L': 16, 'name': 'RYDBERG'}
```

Output folder path established

Step : 0 , Transformer energy = 31.8925 , Transformer energy variance= 543.7189

Step : 50 , Transformer energy = 7.497976 , Transformer energy variance= 179.69516

Step : 100 , Transformer energy = -2.9836862 , Transformer energy variance= 35.55628

Step : 150 , Transformer energy = -5.044218 , Transformer energy variance= 19.876806

Step : 200 , Transformer energy = -6.4999714 , Transformer energy variance= 2.637269

Step : 250 , Transformer energy = -6.633377 , Transformer energy variance= 3.255067

Step : 300 , Transformer energy = -6.476509 , Transformer energy variance= 8.979716

Step : 350 , Transformer energy = -6.746655 , Transformer energy variance= 1.1118166

Step : 400 , Transformer energy = -6.842925 , Transformer energy variance= 1.4557284

```
[ ] 1 !python VMC_with_TF/train.py --train L=16 steps=1000 --ptf patch=2x2 Nh=128 nhead=8 --rydberg V=7 delta=1 Omega=1
```

↪ Final Remarks:

* Complexity:

Computationally more expensive

* Regular Transformer: $O(N^3)$ sampling, $O(N^2)$ inference

* RNN: $O(N)$ for sampling and inference

↪ Linear transformers: $O(N)$

One example: «Transformers are RNNs»

* The intersection of $\langle \text{NLP} \mid \text{Many body physics} \rangle$ is
very promising.

Modern applications of machine learning in quantum sciences

Anna Dawid^{1,2,3*}, Julian Arnold^{4†}, Borja Requena^{2†}, Alexander Gresch^{5,6†}, Marcin Płodzień²,
Kaelan Donatella⁷, Kim A. Nicoli^{8,9}, Paolo Stornati², Rouven Koch¹⁰, Miriam Büttner¹¹,
Robert Okuła^{12,13}, Gorka Muñoz-Gil¹⁴, Rodrigo A. Vargas-Hernández^{15,16,17}, Alba
Cervera-Lierta¹⁸, Juan Carrasquilla¹⁶, Vedran Dunjko¹⁹, Marylou Gabrié²⁰,
Patrick Huembeli^{21,22}, Evert van Nieuwenburg^{19,23}, Filippo Vicentini^{21,24}, Lei Wang^{25,26},
Sebastian J. Wetzel²⁷, Giuseppe Carleo²¹, Eliška Greplová²⁸, Roman Krems²⁹, Florian
Marquardt^{30,31}, Michał Tomza¹, Maciej Lewenstein^{2,32} and Alexandre Dauphin^{2,33*}