Lecture 3: Transformers

Last Time:

- Generative modeling

⇒

Energy-based model

Restricted Boltzmann Machines (RBM)

Recurrent Neural Networks (RNN)

Transformers

Other architectures (Not covered in this mini-course)
Outline:

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 4.5

- GPT = Generative Pre-trained Transformer.
$X = (x_1, x_2, \ldots, x_N)$

- Query: Text on search bar.
- Example query: YouTube
- Key: Videos, title, description
- Value: Best video matches.
We can use $q_i, r_i, p_i$ to compute self-attention.

$$A^i = \sum_{j=1}^{N} \alpha(q^i, r^j) \cdot \bar{v}^j$$

Attention that has information about the most relevant $x_j$ on $x_i$ (Attention of $x_i$ on $x_j$ in $i$)

Solves the vanishing gradient problem in RNNs.
"I live in France, hence I speak French."

\[ \alpha(q_i, K^i_j) = \text{Softmax} \left( \frac{q_i \cdot K^i_j}{\sqrt{d_k}} \right) \]

For normalization purposes.

\[ \sum_{j=1}^{N} \alpha(q_i, K^i_j) = 1 \]

\[ q_i \cdot R_{ij} \in \mathbb{R} \]

\[ q_i \cdot R = (\_\_\_)^N \]
"I live in France, hence I speak French"
\[ \hat{A}_i = \sum_{j=1}^{N} \text{Softmax} \left( \frac{q_i \cdot k_j^T}{V_{dk}} \right) \cdot v_j \]

\[ A = \text{Softmax} \left( \frac{Q \cdot K^T}{V_{dk}} \right) \cdot V \]

For normalization purposes.

- Take advantage of GPU parallelization power.
Positional encoding

\[ x^9, x_2, x_3, x_4, x_5, x_6 \]

\[ x = \text{I'm not sad, I'm happy} \]

\[ \overrightarrow{X} = \text{I'm not happy, I'm sad} \]

Idea

\[ x_i + \overrightarrow{PE(i)} \rightarrow \overrightarrow{x_i} \]
\[
\rightarrow PE(i)_{2j} = \sin \left( \frac{i}{10000 \text{ } \text{m}^2/\text{de}} \right)
\]
\[
PE(i)_{2+1} = \cos \left( \frac{i}{10000 \text{ } \text{m}^2/\text{de}} \right)
\]

Max sequence length

Nearby positions have similar PE at particular component (SK)

Each component of the PE has a different wavelength \( \rightarrow \) short–long range dependencies.
**Multi-Head Attention** allows to learn attention about different features:

\[
\text{Multi-Head Attention} = \text{Concat}(A^1, A^2, \ldots, A^h) W^{(e)}
\]

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

Embedding of inputs
Multi-Head Attention

Scaled Dot-Product Attention

Linear

Concat

$w^{(a)}$

$w^{(u)}$

$w^{(v)}$

Embedding of inputs

New index $\mu = 1 \rightarrow h$

Feature

different self-attention
Multi-Head Attention

- Scaled Dot-Product Attention
  - Linear
  - Concat
- Different self-attention
- Embedding of inputs

Example:
- Group of people → Character, Job, Age, Gender...
- Spirit, Goals, Motivation
Example:

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

In Equation:

\[
\text{Multi-head Attention} = \text{Concat}(A^1, A^2, ..., A^h) W^{(0)}
\]
Adding Attention on inputs $Y$:

At step $i$:

$$x_1, x_2, \ldots, x_i, x_{i+1}, \ldots, x_N$$

$$
\overline{A}_i = \sum_{j<i} \alpha (\overline{q}_j, \overline{r}_j) \cdot \overline{q}_j
$$
Layer Normalization

Adding Attention on inputs Y

\[ \text{At step } i = x_1, x_2, \ldots, x_i, x_{i+1}, \ldots, x_N \]

\[ A_i = \min_{1 \leq j < i} \alpha \left( \overline{a}_j, \overline{a}_i \right) \cdot \overline{a}_i \]
175B parameters

$W(Q), W(K), W(V)$

$W(O), \ldots$

96 Transformer layers

$P(\text{Next word } | Y)$

Source of Stochasticity

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

Human feedback (Reward)
To fine tune the
parameters
(3) **Transformer Wavefunctions**

![Diagram of Transformer Wavefunctions]

\[ \Psi_{TF}(\sigma_1, \sigma_2, ..., \sigma_N) = \sqrt{p_{TF}(\sigma_1, \sigma_2, ..., \sigma_N)} \]

**Cost Function:**

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

K. Sprague & S. Gasic, Comm. Phys., 2024
### Transformer Wavefunctions

**Cost Function:**

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

**Patch size:** e.g., 4 spins

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

For Stochastic Hamiltonians

K. Spragne & S. Cseh, Commun. Phys. 2024
\[ \Psi_{TF}(\sigma_1, \sigma_2, \ldots, \sigma_N) = \sqrt{p_{TF}(\sigma_1, \sigma_2, \ldots, \sigma_N)} \]

\[ \Rightarrow \text{For Stoquastic Hamiltonians} \]

Cost Function:

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

K. Sprague & S. Gierekh, Comm Phys 2024

Another paper: ArXiv:2310.05715
**Rydberg atom Hamiltonian**

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

\[P = \{P(\sigma_1), P(\sigma_2|\sigma_1), \ldots, P(\sigma_N|\sigma_{<N})\}\]

**Rydberg CPT predictions**

Training points

\[
E
\]

\[
\Delta/\hbar
\]

Different parameters

Spin configurations
Transformer wave functions

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

Cloning github repository:

```
> git clone https://github.com/mhibatallah/VMC_with_TF.git
```

Training Arguments:

```
L (int) -- Total lattice size (0x0 would be L=64).
Q (int) -- Number of minibatches per batch.
K (int) -- size of each minibatch.
B (int) -- Total batch size (should be 0*K).
NLOOPS (int) -- Number of loops within the off_diag_labels function. Higher values save ram and generally makes the code run faster (run in 2x). Note: you can only set this
```
approximate the gradients, more efficient but approximate.

At 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=1

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

weights/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':

{'L': 16, 'patch': '2x2', 'Nh': 32, 'dropout': 0.0, 'num_l

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

0.077, Transformer energy variance= 471.61472
```python
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
```

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= 19.876886
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

```

```
Final Remarks:

Complexity:

- 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 NLP and many-body physics is very promising.

Modern applications of machine learning in quantum sciences

Anna Dawid1,2,3*, Julian Arnold4†, Borja Requena2†, Alexander Gresch5,6‡, Marcin Płodzięń2, Kaelan Donatella7, Kim A. Nicoli8,9, Paolo Stornati2, Rouven Koch10, Miriam Büttner11, Robert Okula12,13, Gorka Muñoz–Gil14, Rodrigo A. Vargas–Hernández15,16,17, Alba Cervera-Lierta18, Juan Carraquilla16, Vedran Dunjko19, Marylou Gabrié20, Patrick Huembeli21,22, Evert van Nieuwenburg19,23, Filippo Vicentini21,24, Lei Wang25,26, Sebastian J. Wetzel27, Giuseppe Carleo21, Eliška Greplová28, Roman Krems29, Florian Marquardt30,31, Michał Tomza1, Maciej Lewenstein2,32 and Alexandre Dauphin2,33*