Self-attention as an attractor network: transient memories without backpropagation

Francesco D'Amico, Matteo Negri

SAPIENZA UNIVERSITÀ DI ROMA

Dipartimento di Fisica

October 11, 2024

Contents

One year ago...

PHYSICAL REVIEW RESEARCH **6**, 023057 (2024)

Mapping of attention mechanisms to a generalized Potts model

Riccardo Rende, Federica Gerace, Alessandro Laio, and Sebastian Goldt*
Scuola Internazionale Superiore di Studi Avanzati (SISSA), Via Bonomea 265, 34136 Trieste, Italy

(Received 27 April 2023; revised 14 December 2023; accepted 4 March 2024; published 16 April 2024)

What are transformers?

- Most successful architecture of last years in many domains

What are transformers?

- Most successful architecture of last years in many domains
- Data: sequence of vectors $x \in (\mathbb{R}^d)^N$, so $x = (\vec{x}_1, ..., \vec{x}_N)$

What are transformers?

- Most successful architecture of last years in many domains
- Data: sequence of vectors $x \in (\mathbb{R}^d)^N$, so $x = (\vec{x}_1, ..., \vec{x}_N)$
- At their core: **self-attention**

$$\vec{x}_i^{t+1} = \sum_j \alpha_{i \leftarrow j} W_V \vec{x}_j^t + \vec{x}_i^t \tag{1}$$

- **Attention weights**:

$$\alpha_{i \leftarrow j} = \text{softmax}_j[\lambda(\vec{x}_i^T W_K^T, W_Q \vec{x}_j)] \tag{2}$$

What are transformers?

- Most successful architecture of last years in many domains
- Data: sequence of vectors $x \in (\mathbb{R}^d)^N$, so $x = (\vec{x}_1, ..., \vec{x}_N)$
- At their core: **self-attention**

$$\vec{x}_i^{t+1} = \sum_j \alpha_{i \leftarrow j} W_V \vec{x}_j^t + \vec{x}_i^t \qquad (1)$$

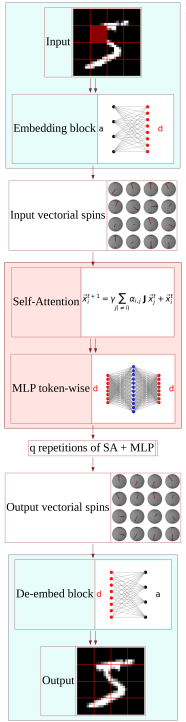
- **Attention weights**:

$$\alpha_{i \leftarrow j} = \text{softmax}_j [\lambda (\vec{x}_i^T W_K^T, W_Q \vec{x}_j)] \qquad (2)$$

- **Autoregressive task**: predict one missing token given the others.

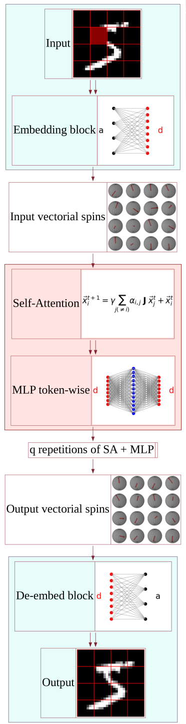
Standard transformer vs recycling

- **Standard**: *q different* SA+MLP blocks
- **Recycling**: *one single layer* iterated q times



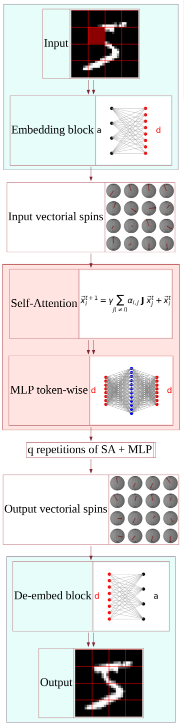
Standard transformer vs recycling

- **Standard**: *q different* SA+MLP blocks
- **Recycling**: *one single layer* iterated q times
- Why recycling? No cheaper training



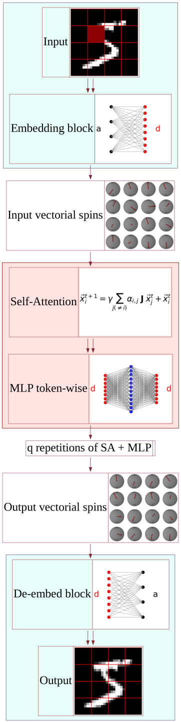
Standard transformer vs recycling

- **Standard**: *q different* SA+MLP blocks
- **Recycling**: *one single layer* iterated q times
- Why recycling? No cheaper training
- $\Rightarrow$ Pro (1) fundamental in some relevant cases such as **Alphafold2**



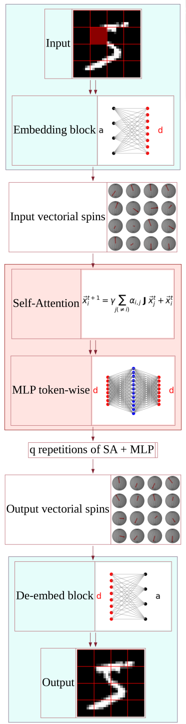
Standard transformer vs recycling

- **Standard**: *q different* SA+MLP blocks
- **Recycling**: *one single layer* iterated q times
- Why recycling? No cheaper training
- $\Rightarrow$ Pro (1) fundamental in some relevant cases such as **Alphafold2**
- $\Rightarrow$ Pro (2) iterative dynamical system instead of different layers $\rightarrow$ **explainability**



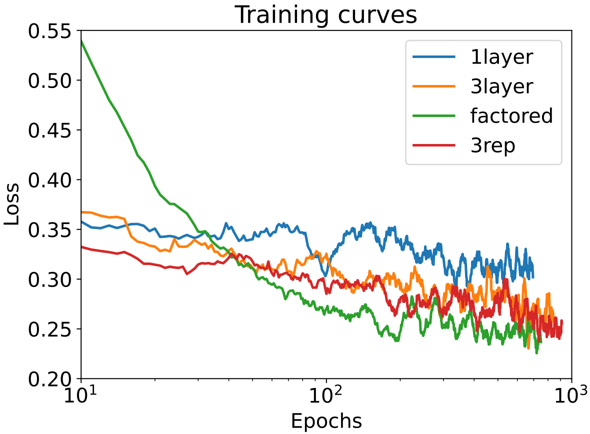
Standard transformer vs recycling

- **Standard**: *q different* SA+MLP blocks
- **Recycling**: *one single layer* iterated q times
- Why recycling? No cheaper training
- $\Rightarrow$ Pro (1) fundamental in some relevant cases such as **Alphafold2**
- $\Rightarrow$ Pro (2) iterative dynamical system instead of different layers $\rightarrow$ **explainability**
- Recycling training issue: dependence on q.
- $\Rightarrow$ Solution: $q_{train} \in [q_{min}, q_{max}]$.

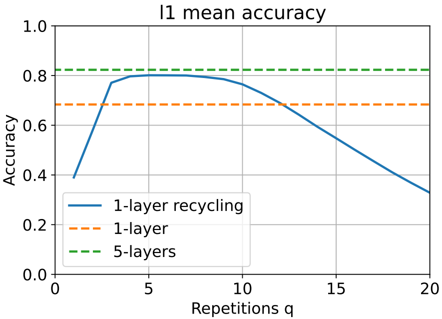


Recycling on generalized Potts model

- Potts dataset of Riccardo et al, MLM task
- 3rep model: one layer as factored attention but same performance as 3 layers



Recycling on CIFAR10



Contents

Our work: (1) summary

1. Transformers can be used as **dynamical systems**
2. Can the SA map be written as the **derivative of a cost**?
   Is it an energy E?

Our work: (1) summary

1. Transformers can be used as **dynamical systems**
2. Can the SA map be written as the **derivative of a cost**? Is it an energy E?
3. $\Rightarrow$ **Yes, but it is not an energy.** **It is formally a Negative log Pseudo Likelihood (NLP)**

Our work: (1) summary

1. Transformers can be used as **dynamical systems**
2. Can the SA map be written as the **derivative of a cost**?
   Is it an energy E?
3. $\Rightarrow$ **Yes, but it is not an energy.**
   It is formally a Negative log Pseudo Likelihood (NLP)
4. We can train a very simple SA ("bare SA") directly minimizing the NLP
5. We compare result with a recycled transformer in MLM and denoising tasks

Our work: (2) pros and cons

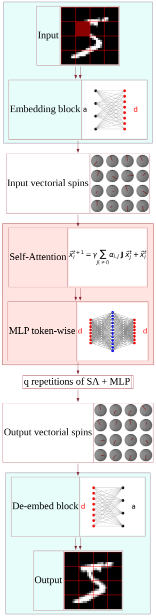
1. PROs: training is
   - Fast (no backpropagation)
   - Local (relevant for biological networks)
   - More explainable: planted model

Our work: (2) pros and cons

1. PROs: training is
   - Fast (no backpropagation)
   - Local (relevant for biological networks)
   - More explainable: planted model

2. CONs:
   - Sensible to hyperparameters
   - Worse test performances
   - High memory usage

The simplifications

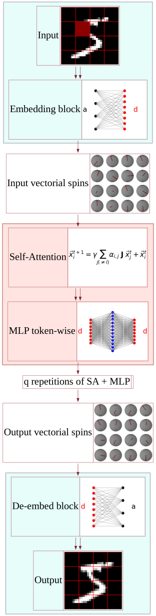


Again, self-attention:

$$\vec{x}_i^{t+1} = \sum_j \alpha_{i \leftarrow j} W_V \vec{x}_j^t + \vec{x}_i^t$$

$$\alpha_{i \leftarrow j} = \text{softmax}_j \left[\lambda \left(\vec{x}_i^T W_K^T, W_Q \vec{x}_j \right) \right]$$

The simplifications



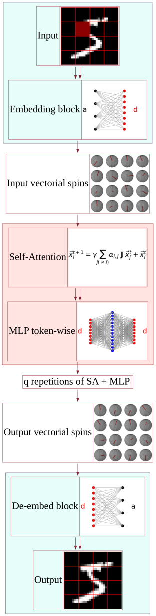
Again, self-attention:

$$\vec{x}_i^{t+1} = \sum_j \alpha_{i\leftarrow j} W_V \vec{x}_j^t + \vec{x}_i^t$$

$$\alpha_{i\leftarrow j} = \text{softmax}_j \left[\lambda \left(\vec{x}_i^T W_K^T, W_Q \vec{x}_j \right) \right]$$

1. Take a fixed embedding

The simplifications



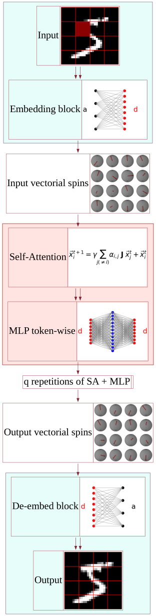
Again, self-attention:

$$\vec{x}_i^{t+1} = \sum_j \alpha_{i\leftarrow j} W_V \vec{x}_j^t + \vec{x}_i^t$$

$$\alpha_{i\leftarrow j} = \mathrm{softmax}_j \left[\lambda \left(\vec{x}_i^T W_K^T, W_Q \vec{x}_j \right) \right]$$

1. Take a fixed embedding
2. Get rid of the MLP

The simplifications



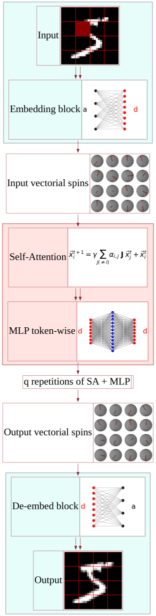
Again, self-attention:

$$\vec{x}_i^{t+1} = \sum_j \alpha_{i\leftarrow j} W_V \vec{x}_j^t + \vec{x}_i^t$$

$$\alpha_{i\leftarrow j} = \text{softmax}_j \left[\lambda \left(\vec{x}_i^T W_K^T, W_Q \vec{x}_j \right) \right]$$

1. Take a fixed embedding
2. Get rid of the MLP
3. We call $W_k^T W_q =: J$. Get rid of positional encoding, instead $J = J_{ij}$

The simplifications



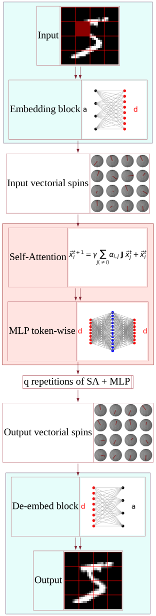
Again, self-attention:

$$\vec{x}_i^{t+1} = \sum_j \alpha_{i\leftarrow j} W_V \vec{x}_j^t + \vec{x}_i^t$$

$$\alpha_{i\leftarrow j} = \text{softmax}_j \left[\lambda \left(\vec{x}_i^T W_K^T, W_Q \vec{x}_j \right) \right]$$

1. Take a fixed embedding
2. Get rid of the MLP
3. We call $W_k^T W_q =: J$. Get rid of positional encoding, instead $J = J_{ij}$
4. Set $W_v \equiv J$.

The simplifications



Again, self-attention:

$$\vec{x}_i^{t+1} = \sum_j \alpha_{i\leftarrow j} W_V \vec{x}_j^t + \vec{x}_i^t$$

$$\alpha_{i\leftarrow j} = \text{softmax}_j \left[\lambda \left(\vec{x}_i^T W_K^T, W_Q \vec{x}_j \right) \right]$$

1. Take a fixed embedding
2. Get rid of the MLP
3. We call $W_k^T W_q =: J$. Get rid of positional encoding, instead $J = J_{ij}$
4. Set $W_v \equiv J$.

Our objective is a proof-of-work.
(2) and (4) have theoretical implications
(1) and (3) are practical simplifications

The bare self-attention

The only parameters tensor is $J \in \mathbb{R}^{N\times N\times d\times d}$.

The bare self-attention

The only parameters tensor is $J \in \mathbb{R}^{N\times N\times d\times d}$.

$\Rightarrow$ Dynamical system of vectors

$$\vec{x}_i^{t+1} = \sum_{j(\neq i)} \alpha_{i\leftarrow j} J_{ij} \vec{x}_j^t + \vec{x}_i^t \tag{3}$$

$$\alpha_{i\leftarrow j} = \text{softmax}_j \left[\lambda \vec{x}_i \cdot J_{ij} \vec{x}_j \right] \tag{4}$$

Contents

Self-attention from a variational approach

In general **energy models**

- Update:

$$x^{t+1} = f_\theta(x) + x^t \tag{5}$$

- Energy $F_\theta(x)$ is such that

$$x^{t+1} = -\nabla_x F_\theta(x) + x^t \tag{6}$$

Self-attention from a variational approach

In general **energy models**

- Update:

$$x^{t+1} = f_\theta(x) + x^t \tag{5}$$

- Energy $F_\theta(x)$ is such that

$$x^{t+1} = -\nabla_x F_\theta(x) + x^t \tag{6}$$

In **self-attention**:

- Update:

$$\vec{x}_i^{t+1} = \sum_{j(\neq i)} \alpha_{i\leftarrow j} J_{ij} \vec{x}_j^t + \vec{x}_i^t \tag{7}$$

- "Energy"

$$F(x;J) = -\frac{1}{\lambda}\sum_i \log\Big[\sum_{j(\neq i)} \exp\big(\lambda \vec{x}_i \cdot J_{ij}\vec{x}_j\big)\Big] = \sum_i e_i(x;J) \tag{8}$$

Self-attention from a variational approach

In general **energy models**

- Update:

$$x^{t+1} = f_\theta(x) + x^t \tag{5}$$

- Energy $F_\theta(x)$ is such that

$$x^{t+1} = -\nabla_x F_\theta(x) + x^t \tag{6}$$

In **self-attention**:

- Update:

$$\vec{x}_i^{t+1} = \sum_{j(\neq i)} \alpha_{i\leftarrow j} J_{ij} \vec{x}_j^t + \vec{x}_i^t \tag{7}$$

- "Energy"

$$F(x;J) = -\frac{1}{\lambda}\sum_i \log\left[\sum_{j(\neq i)} \exp\left(\lambda \vec{x}_i \cdot J_{ij}\vec{x}_j\right)\right] = \sum_i e_i(x;J) \tag{8}$$

- An "energy" individually decreased by each token:

$$f_i(x;J) = -\nabla_{\vec{x}_i} e_i(x;J) + \vec{x}_i \tag{9}$$

Self-attention from a variational approach

In general **energy models**

- Update:

$$x^{t+1} = f_\theta(x) + x^t \tag{5}$$

- Energy $F_\theta(x)$ is such that

$$x^{t+1} = -\nabla_x F_\theta(x) + x^t \tag{6}$$

In **self-attention**:

- Update:

$$\vec{x}_i^{t+1} = \sum_{j(\neq i)} \alpha_{i \leftarrow j} J_{ij} \vec{x}_j^t + \vec{x}_i^t \tag{7}$$

- "Energy"

$$F(x;J) = -\frac{1}{\lambda} \sum_i \log\left[\sum_{j(\neq i)} \exp\left(\lambda \vec{x}_i \cdot J_{ij} \vec{x}_j\right)\right] = \sum_i e_i(x;J) \tag{8}$$

- An "energy" individually decreased by each token:

$$f_i(x;J) = -\nabla_{\vec{x}_i} e_i(x;J) + \vec{x}_i \tag{9}$$

The self-attention cost is formally a pseudo-likelihood

1. Pseudo-likelihood approximation: $p(x) \approx \prod_i p(x_i|x_{\setminus i})$, with

$$p(x_i|x_{\setminus i}) = \frac{\exp(\beta H_i(x))}{\int d\mu(x_i) \exp(\beta H_i(x))} \tag{10}$$

The self-attention cost is formally a pseudo-likelihood

1. Pseudo-likelihood approximation: $p(x) \approx \prod_i p(x_i|x_{\setminus i})$, with

$$p(x_i|x_{\setminus i}) = \frac{\exp(\beta H_i(x))}{\int d\mu(x_i) \exp(\beta H_i(x))} \tag{10}$$

2. NLP compared to cost of SA:

$$\text{NLP}(x,J) = -\sum_i \log(p(x_i|x_{\setminus i})) = \sum_i e_i \tag{11}$$

$$F(x;J) = -\frac{1}{\lambda}\sum_i \log\Big[\sum_{j(\neq i)} \exp\big(\lambda \vec{x}_i \cdot J_{ij}\vec{x}_j\big)\Big] = \sum_i e_i \tag{12}$$

Training via max-NLP

We learn J directly from NLP: no forward and backpropagation.

Training via max-NLP

We learn J directly from NLP: no forward and backpropagation.

- Take some datapoints x^μ, select a random i:

$$J_{ij}^{t+1} = J_{ij}^{t} - \eta \frac{\partial}{\partial J_{ij}^{t}} \sum_{\mu} e_i(x^\mu, J^t) \tag{13}$$

Training via max-NLP

We learn J directly from NLP: no forward and backpropagation.

- Take some datapoints x^μ, select a random i:

$$J_{ij}^{t+1} = J_{ij}^t - \eta \frac{\partial}{\partial J_{ij}^t} \sum_\mu e_i(x^\mu, J^t) \tag{13}$$

- Or directly with GD or SGD using $\sum_\mu e_i(x^\mu, J^t)$ as a Loss

Training via max-NLP

We learn J directly from NLP: no forward and backpropagation.

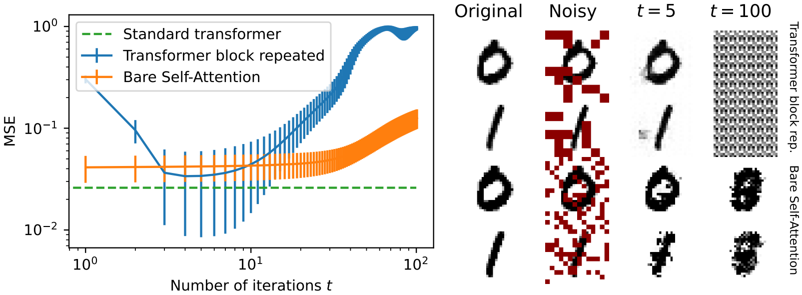
- Take some datapoints x^μ, select a random i:

$$J_{ij}^{t+1} = J_{ij}^t - \eta \frac{\partial}{\partial J_{ij}^t} \sum_\mu e_i(x^\mu, J^t) \tag{13}$$

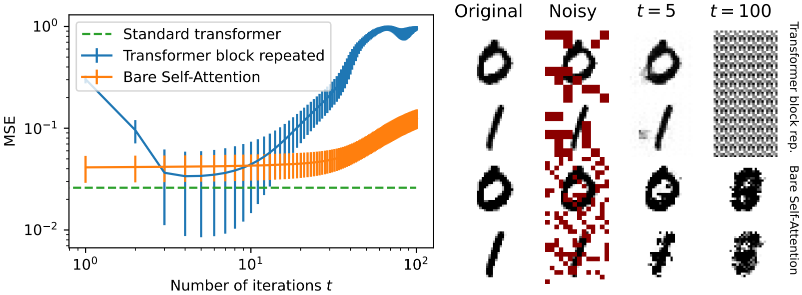
- Or directly with GD or SGD using $\sum_\mu e_i(x^\mu, J^t)$ as a Loss
- We plant the dataset in the cost of the model

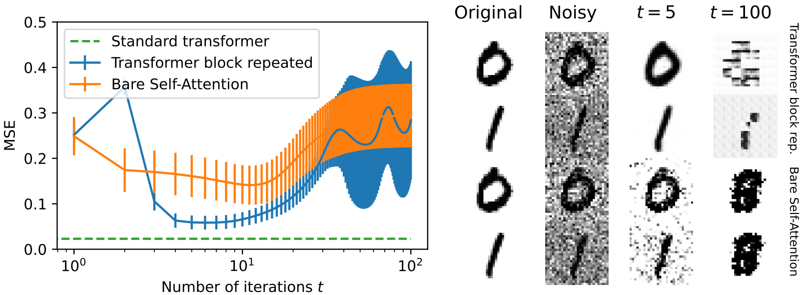
  $\Rightarrow$ Hebb rule and **Hopfield models**

Results on real data



Results on real data





Contents

Conclusions

1. The cost of self-attention is a pseudo-likelihood

$$F(x;J) = -\frac{1}{\lambda}\sum_i \log\Big[\sum_{j(\neq i)} \exp\big(\lambda \vec{x}_i \cdot J_{ij}\vec{x}_j\big)\Big] = \sum_i e_i \tag{14}$$

2. A bare self-attention can be trained via max-pseudo-likelihood

$$J_{ij}^{t+1} = J_{ij}^t - \eta \frac{\partial}{\partial J_{ij}^t} \sum_\mu e_i(x^\mu, J^t) \tag{15}$$

3. It works qualitatively as recycled transformers

Advancements (1): NPL as a different training method

- Can we improve it? For ex. with Contrastive Divergence

$$\Delta J_{ij} \propto -\frac{\partial}{\partial J_{ij}} \langle e_i(x^\mu, J) \rangle_{data} + \frac{\partial}{\partial J_{ij}} \langle e_i(x^\mu, J) \rangle_{model} \tag{16}$$

Advancements (1): NPL as a different training method

- Can we improve it? For ex. with Contrastive Divergence

$$\Delta J_{ij} \propto -\frac{\partial}{\partial J_{ij}} \langle e_i(x^\mu, J) \rangle_{data} + \frac{\partial}{\partial J_{ij}} \langle e_i(x^\mu, J) \rangle_{model} \tag{16}$$

- Can we train a **recycled 1-layer transformer** with NLP?
  Encoder $\Rightarrow$ SA $\Rightarrow$ MLP $\Rightarrow$ Decoder

Advancements (1): NPL as a different training method

- Can we improve it? For ex. with Contrastive Divergence

$$\Delta J_{ij} \propto -\frac{\partial}{\partial J_{ij}} \langle e_i(x^\mu, J) \rangle_{data} + \frac{\partial}{\partial J_{ij}} \langle e_i(x^\mu, J) \rangle_{model} \tag{16}$$

- Can we train a **recycled 1-layer transformer** with NLP?
Encoder $\Rightarrow$ SA $\Rightarrow$ MLP $\Rightarrow$ Decoder

- **What differs** in tensor J trained *via backpropagation or via pseudo-likelihood*?
Can we use **NLP for explainability in standard transformers**?

Advancements (2): theoretical

- Hebb rule is the minimum of NLP in two-body models?

  What is the relation between **self-attention and Hopfield models**?

Advancements (2): theoretical

- Hebb rule is the minimum of NLP in two-body models?

  What is the relation between **self-attention and Hopfield models**?

- NLP training is local, so biologically plausible

  $\Rightarrow$ **biological networks with self-attention**

Self-attention as an attractor network: transient memories without backpropagation

Francesco D’Amico, Matteo Negri

Chimera group, physics department, Sapienza

Contacts:

francesco.damico@uniroma1.it
matteo.negri@uniroma1.it

Self attention in transformers

- Data are sequence of tokens: token $\vec{x}_i \in \mathbb{R}^d$ with $i = 1, \ldots, N$, and sequence $x \in (\mathbb{R}^d)^N$
- Update:

$$\vec{x}_i^{t+1} = \sum_{j(\neq i)} \alpha_{i \leftarrow j} V \vec{x}_j^t + \vec{x}_i^t \tag{17}$$

- Attention weights:

$$\alpha_{i \leftarrow j} = \text{softmax}_j[\lambda (K \vec{x}_i) \cdot (Q \vec{x}_j)] \tag{18}$$

- $Q, K \in \mathbb{R}^{d_h \times d}$, meanwhile $V \in \mathbb{R}^{d \times d}$
- In transformers, V, K, Q do not depend on (i,j) $\rightarrow$ permutational invariance (p.i.)
- Positional encoding to break p.i. $\rightarrow \vec{x}_i$ is a mix of positional and semantical information

Our choices

1. V, K, Q depends on (i, j).

   So for us $Q, K \in \mathbb{R}^{N\times N\times d_h\times d}$, $V \in \mathbb{R}^{N\times N\times d\times d}$

   $\Rightarrow$ to get rid of positional encoding

2. Defining

$$J = \sum_{\mu=1}^{h_d} (K^T)^\mu Q^\mu \tag{19}$$

   we constraint $V \equiv J$

   $\Rightarrow$ to write the update as a derivative of an energy

In the end, the only parameters tensor is $J \in \mathbb{R}^{N\times N\times d\times d}$.

$\Rightarrow$ an update

$$\vec{x}_i^{t+1} = \sum_{j(\neq i)} \alpha_{i\leftarrow j} J_{ij} \vec{x}_j^t + \vec{x}_i^t \tag{20}$$

This model energy compared to pseudo-likelihood

- This "energy" is individually decreased by each token, because update rule reads

$$f_{J,i}(x) = -(1-\gamma)\nabla_{\vec{x}_i} e_{J,i}(x) + \gamma\vec{x}_i \tag{21}$$

$$x_i^{\alpha,t+1} = -(1-\gamma)\frac{\partial}{\partial x_i^\alpha} e_{J,i}(x^t) + \gamma x_i^\alpha = (1-\gamma)\sum_{j\neq i}\alpha_{i\leftarrow j}\sum_\beta J_{ij}^{\alpha\beta} x_j^\beta + \gamma x_i^\alpha \tag{22}$$

- This is how Negative Pseudo log-Likelihood (NPL) works:

$$\text{NPL}(x,J) = -\sum_i \log(p(x_i|x_{/i})) = \sum_i e_i \tag{23}$$

- To be compared to

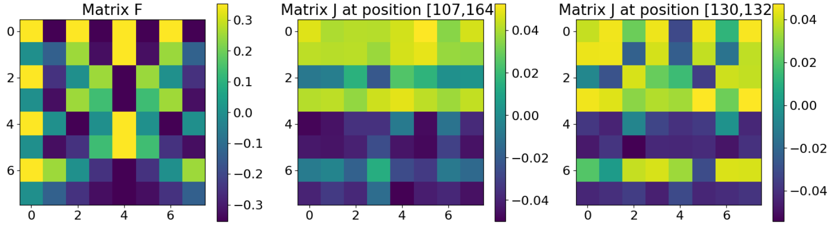
$$F_J(x) = -\frac{1}{\lambda}\sum_i \log\Big[\sum_{j(\neq i)} \exp\big(\lambda\vec{x}_i\cdot J_{ij}\vec{x}_j\big)\Big] = \sum_i e_{J,i}(x)$$

Complete pipeline

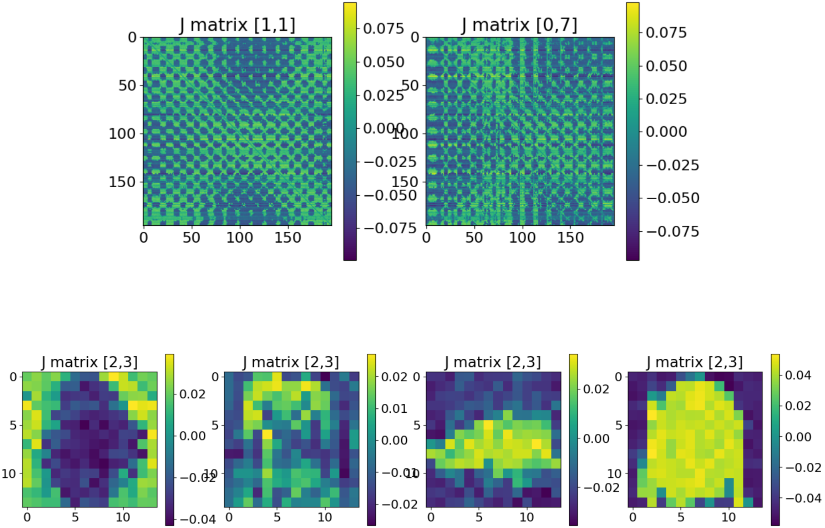
1. Spherical embedding: every pixel channel
   $x_i \in \mathbb{R} \to \vec{x}_i \in \mathbb{R}^2, |\vec{x}_i| = 1$
2. Patches: non overlapping squares of p pixels grouped together
   $\Rightarrow \vec{x}_i \in \mathbb{R}^{2p}$, $d_i = 2p$.
3. A non trainable $F \in \mathbb{R}^{d,d_i}$ matrix: $\vec{x}_i \to F\vec{x}_i$
4. Training from

$$F_J(x) = -\frac{1}{\lambda}\sum_i \log\Big[\sum_{j(\neq i)} \exp\big(\lambda \vec{x}_i \cdot F^T J_{ij} F \vec{x}_j\big)\Big] = \sum_i e_{J,i}(x)$$

F matrix


Matrix F
Matrix J at position [107,164
Matrix J at position [130,132
0
2
4
6
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
0.04
0.02
0.00
−0.02
−0.04

Visualizing J matrix



Logsumexp energy models

$$E(\vec{x}) = -\frac{1}{\lambda}\log\sum_{\mu=1}^{P} e^{\lambda\,\vec{x}\cdot\vec{\xi}^{\mu}} + \frac{1}{2}|\vec{x}|^2 \tag{24}$$

$$\mathbf{x}_{t+1} = \sum_{\mu} a_t^{\mu}\,\xi^{\mu} \tag{25}$$

$$a_t^{\mu} = \frac{e^{\lambda_{x_t}\cdot\xi^{\mu}}}{\sum_{\nu} e^{\lambda_{x_t}\cdot\xi^{\nu}}} \tag{26}$$