Machine Learning for Graphs and Sequential Data

Sequential Data – Neural Network Approaches

lecturer: Prof. Dr. Stephan Günnemann
www.daml.in.tum.de

Summer Term 2024

Data Analytics and
Machine Learning

Roadmap

- Chapter: Temporal Data / Sequential Data
  1. Autoregressive Models
  2. Markov Chains
  3. Hidden Markov Models
  4. **Neural Network Approaches**
     a) Word Vectors
     b) RNNs
     c) Non-Recurrent Models (ConvNets, Transformer)
     d) Structured State Space Models
  5. Temporal Point Processes

Introduction

- Sometimes when modeling a sequence we do not need the complete history to produce the output
- Example: **generating speech**
  – Raw audio has many data points (16000 per second)
  – Important relations on many time scales
- Recall an autoregressive model:

$$X_t = c + \sum_{i=1}^{p} \varphi_i X_{t-i} + \varepsilon_t$$

- Uses a fixed window of p previous inputs and performs regression
- Can we use neural networks to capture more complex behavior?
  – RNNs share the parameters across time steps, but **depend on full history**
  – We can instead use **Convolutional Neural Networks (ConvNets)**

Recap: Definition

- The convolution $f * g$ of functions f and g is

$$(f * g)(\mathrm{t}) = \int_{-\infty}^{+\infty} f(\tau)g(t-\tau)d\tau$$

- In image processing, given an image I and a kernel K, both 2-D matrices, the convolution can be writen as:

$$(K * I)(i,j) = \sum_m \sum_n I(i-m, j-n)K(m,n)$$

- Output is again a 2-D matrix (transformed image), where an element (pixel) is a sum of its neighbors, weighted by a kernel

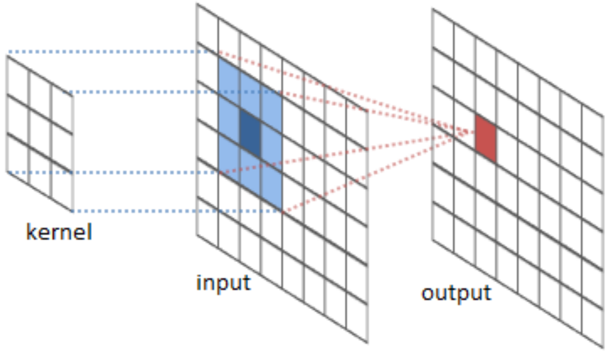
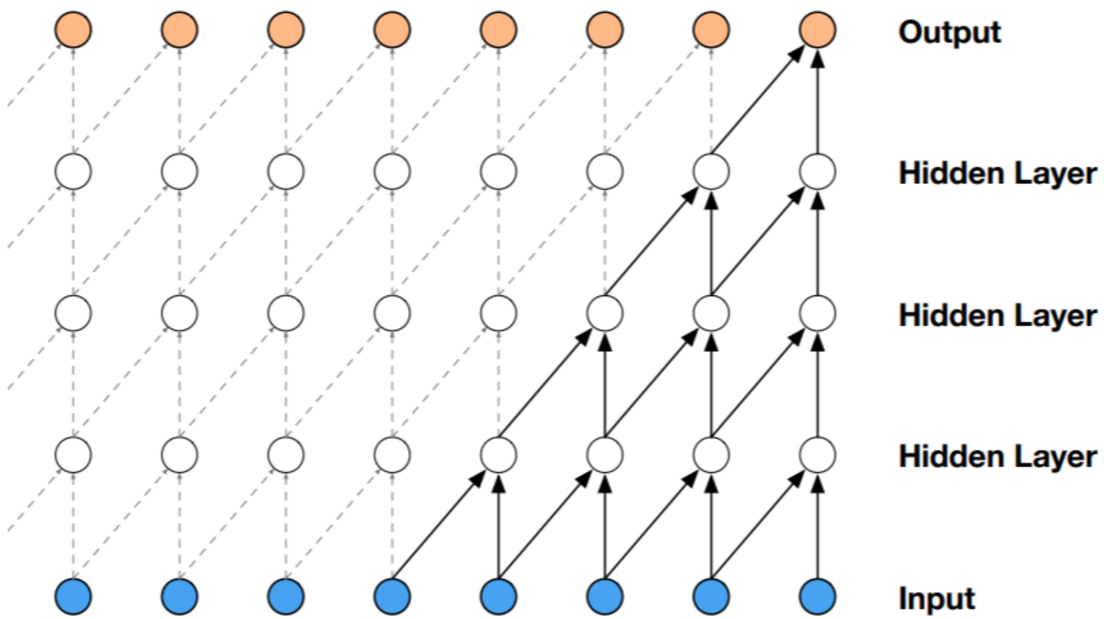


Figure from https://intellabs.github.io/RiverTrail/tutorial/

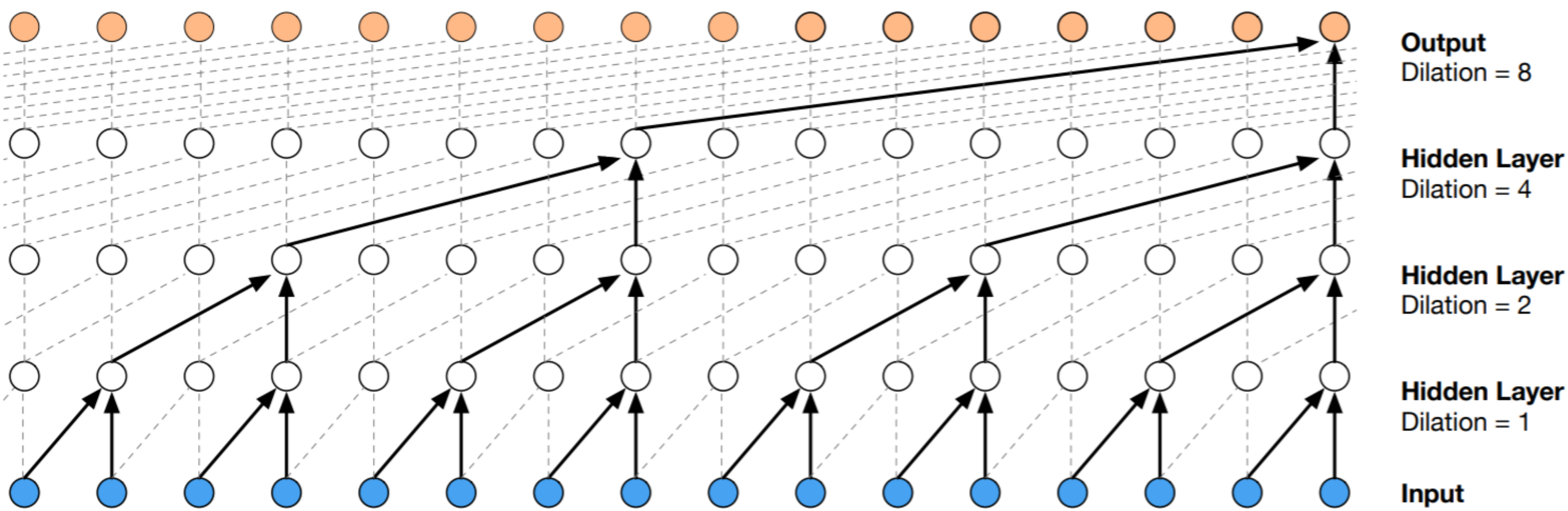
WaveNet

- Sequences are 1-D so we can use a 1-D version of ConvNets
- **WaveNet** [2] is an architecture that uses 1-D ConvNets to model speech
  - In addition, it uses special convolutions to ensure causality and increase receptive field
- **Causal convolutions** – ensure that the output only depends on the past



WaveNet

- **Dilated convolutions** – skip some inputs to increase the receptive field
  - Dilation of 1 gives standard convolution
  - If we start with dilation of 1 in the first layer and double it with every layer (2,4,8...) the receptive field will be the exponential of the number of layers
  - E.g. with 4 layers we use 16 inputs in the first layer



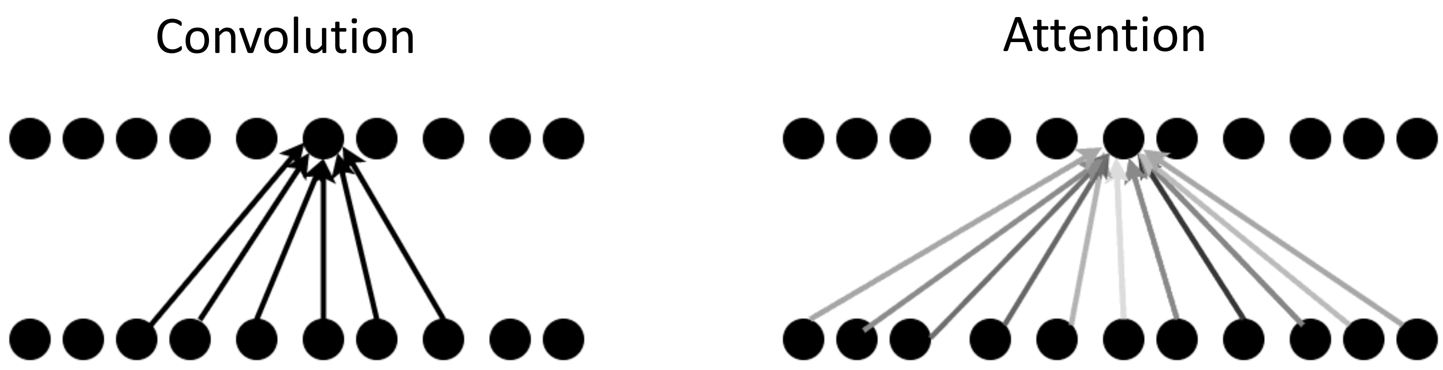
Transformers & Attention

Transformers:

- Transfomers [3] are fast models using attention mechanisms
  - Like WaveNet, it is not a recurrent neural network → parallelizable and fast

Attention:

- Attention is a learned weighting over the elements x_j (given element x_i)

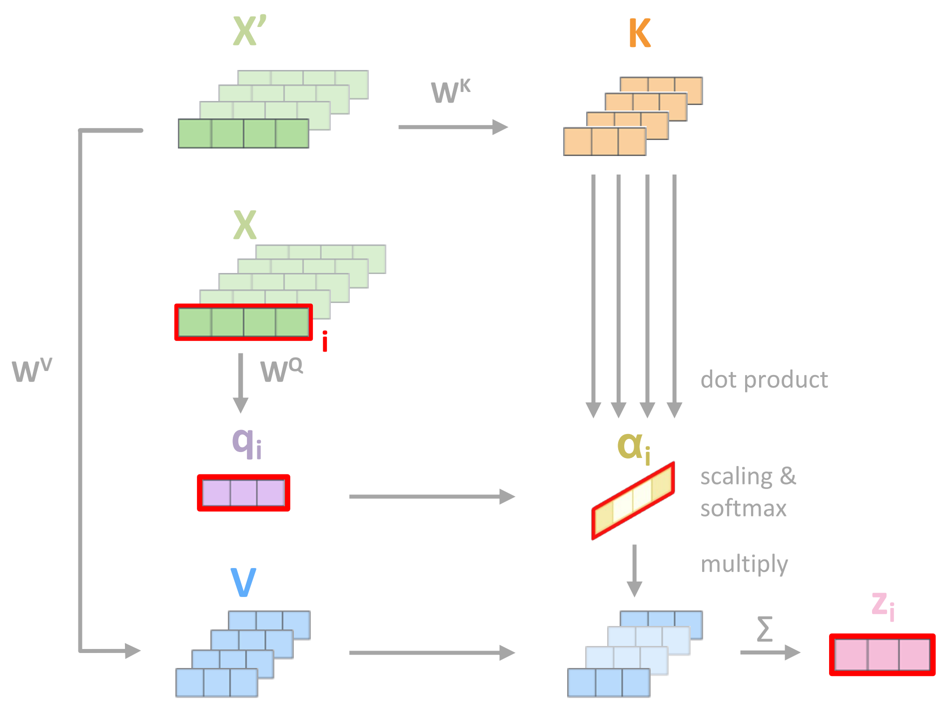


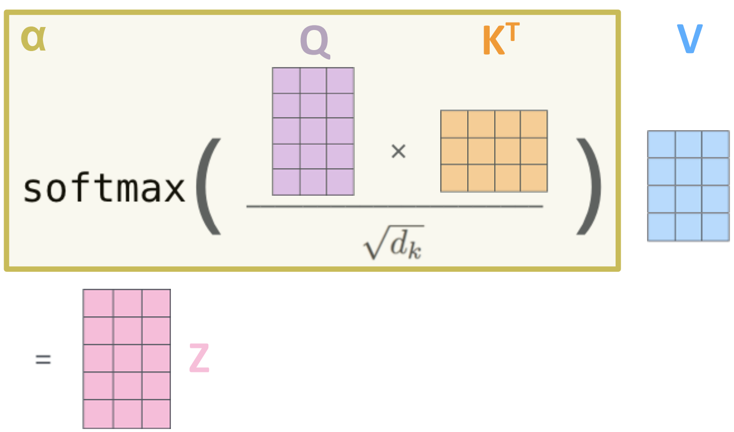
Cross-Attention

- Cross-attention:
  - Two input sources X, X‘
  - Key / query product decides which entries of X‘ “attend to” which entries of X

- Weighting mechanism:
  - The weighting is computed by applying softmax to query/key scores
  - Query depends on x_i; key on x'_j
  - The weight indicates how much of v_j we use (the “value” of x'_j)

- Attention is easily computable in a matrix formulation.



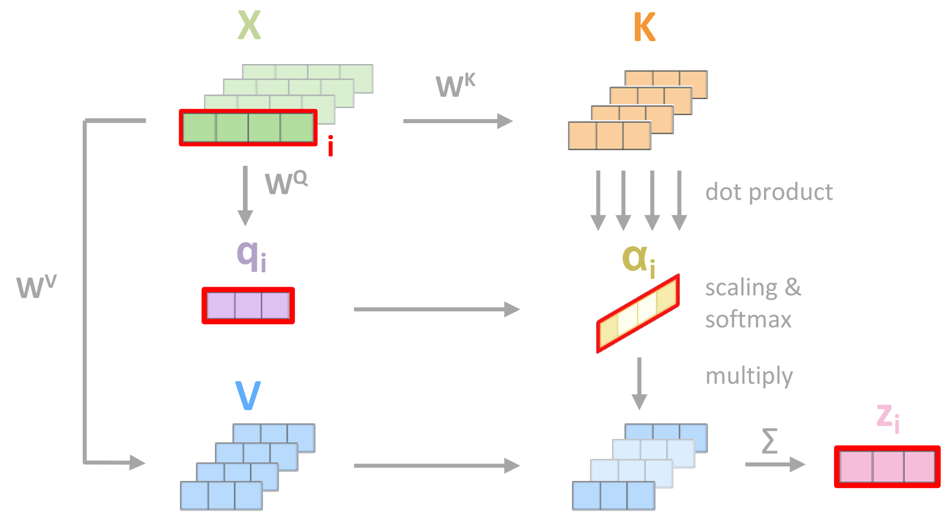


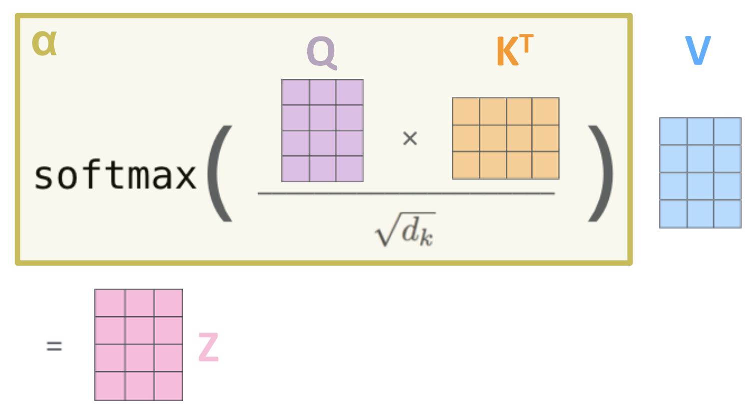
Self-Attention

- Self-attention: the attention is on the input signal X itself

- Weighting mechanism:
  - The weighting is computed by applying softmax to query/key scores
  - Query depends on x_i; key on x_j
  - The weight indicates how much of v_j we use (the “value” of x_j)

- Attention is easily computable in a matrix formulation.

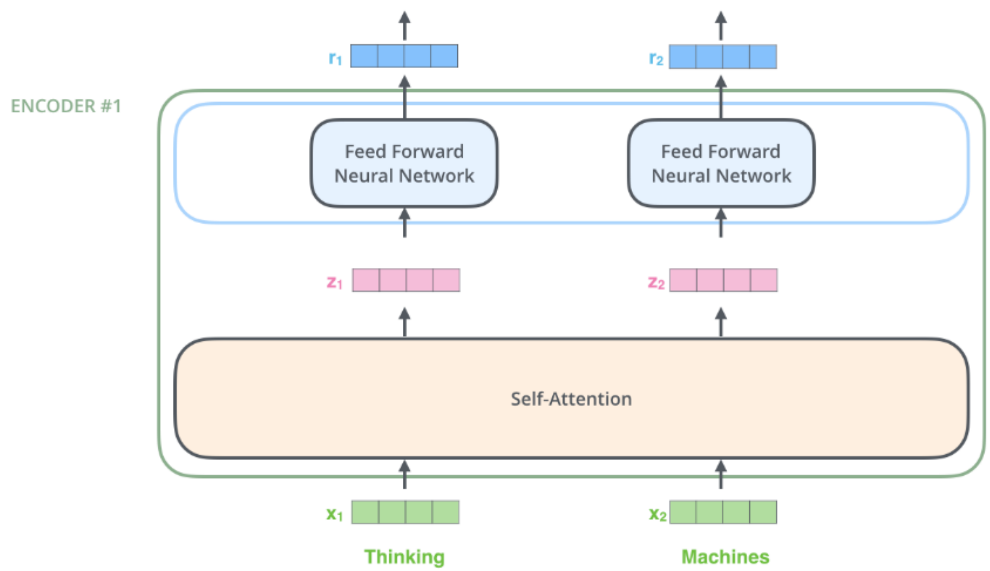




“Self-attention allows the model to look at other positions in the input sequence for clues that can help lead to a better encoding for this word” [4]

Encoder Block

- Tokens (e.g. words) are represented with **embeddings**
- The self-attention layer “couples” the embeddings
- The rest handles the embeddings independently

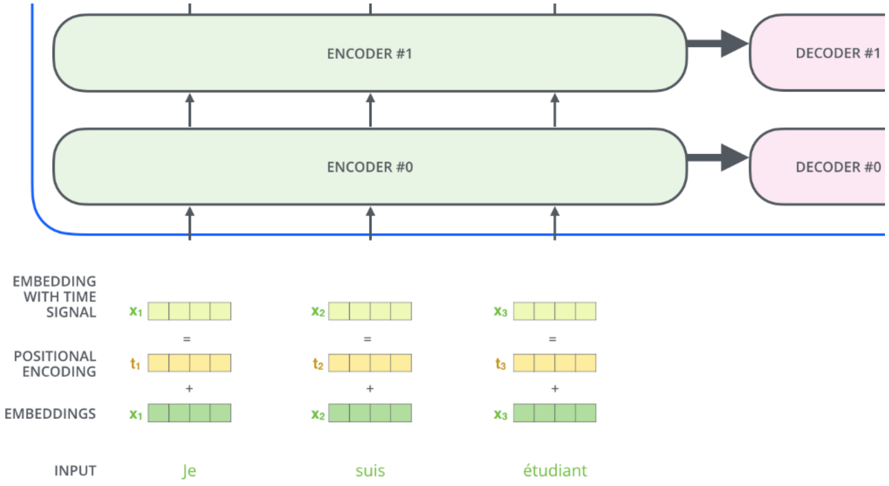


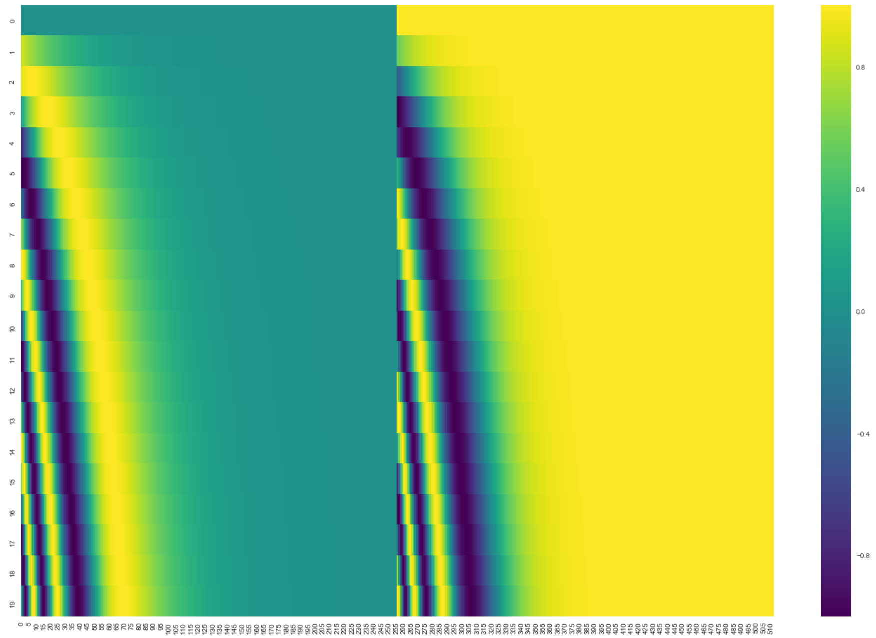
The following images are taken from [4] Jalammar blog

Positional Encoding

- **Note**: Attention mechanisms do not care about the order of tokens / words, i.e. the architecture itself is not aware of the non-i.i.d. nature

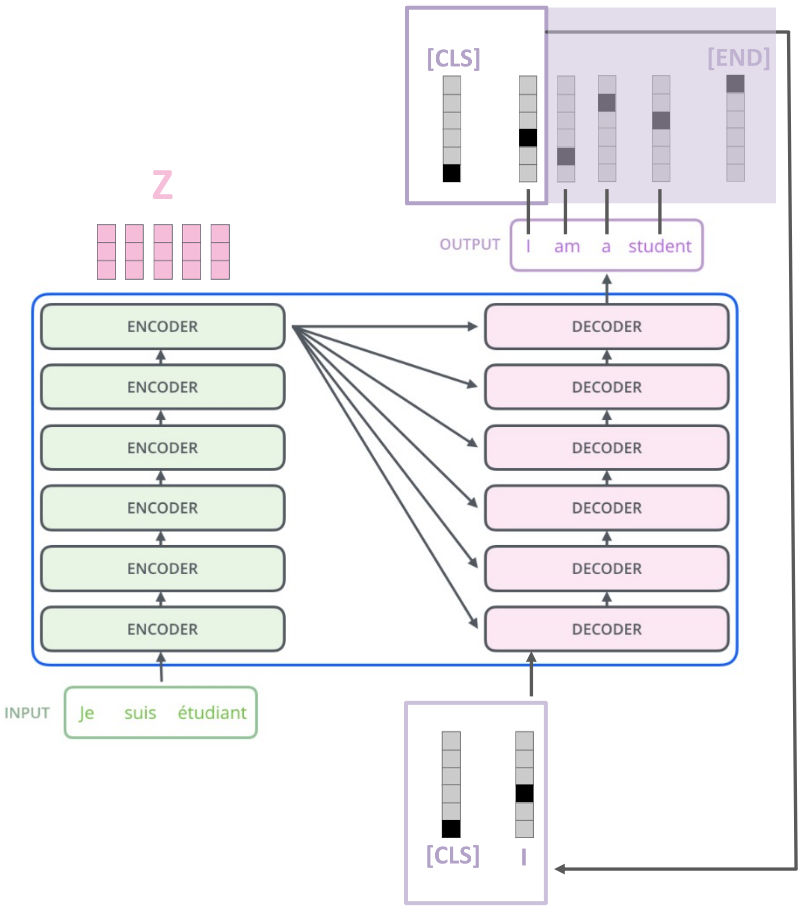
➢ Standard workaround: **positional encoding** to represent the token order. Positional encoding: meaningful static vectors which are concatenated with the token embeddings.

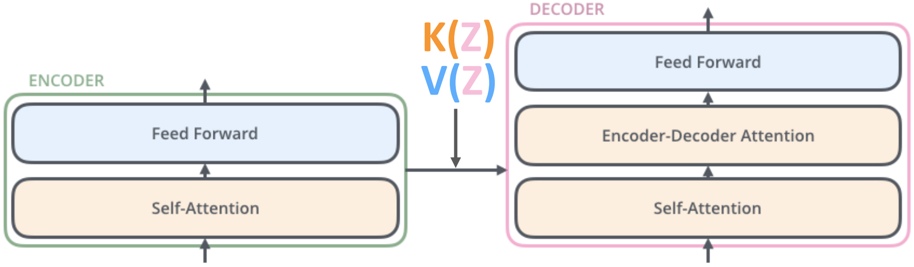




Transformers

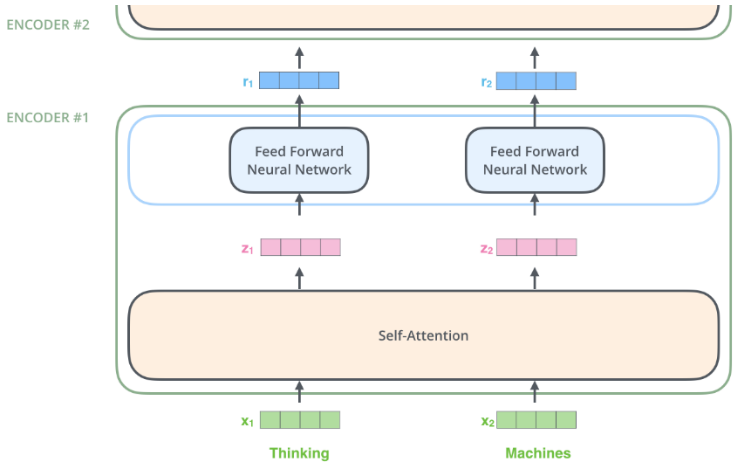
- Transformers are composed of a stack of encoders and decoders using (self-)attention

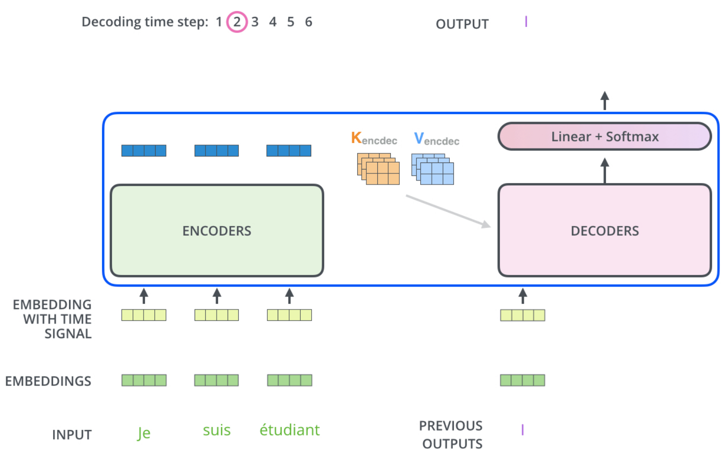




Transformers: Notes on Training and Inference

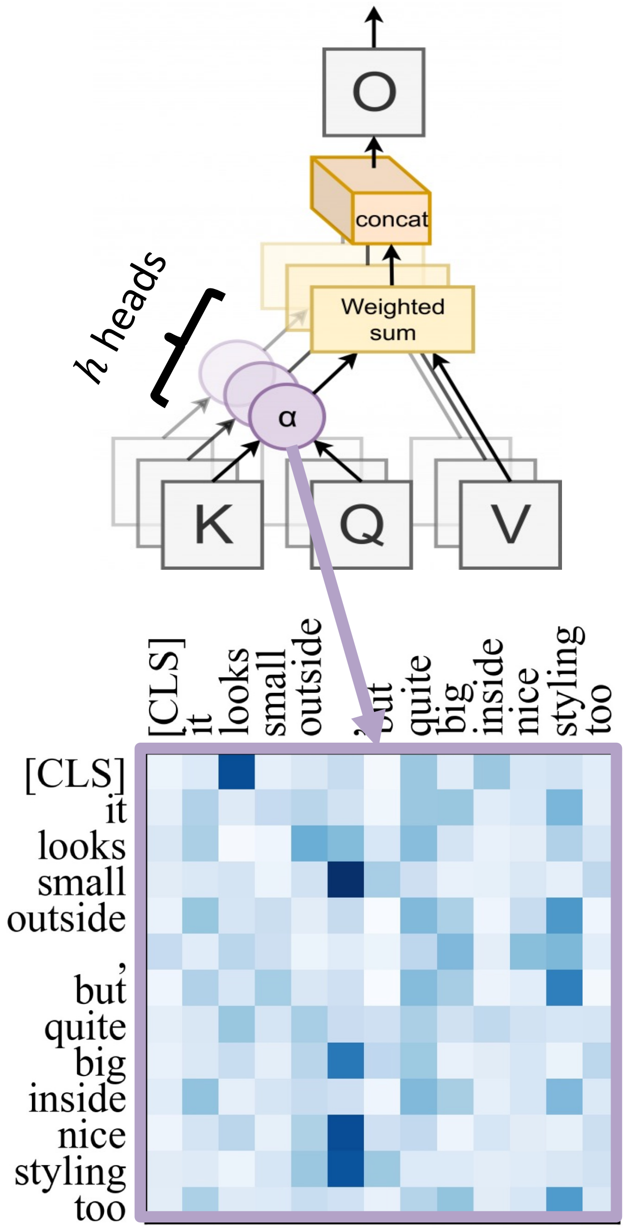
- During training, the embeddings flow all through the transformer at the same time/in parallel (attention coefficients are masked for future tokens)
  - This enables efficient training on very large datasets; crucial for the success of recent models
- At inference time, decoding is done one step after the other until the end of sentence symbol is reached.





Transformers: Complexity

- Multi-Head Attention: Combine the output of h self-attention blocks
- Each self-attention block computes n^2 attention weights
- Intractable for long sequences
- Many possible solutions:
  - Fix structure of attention weights
  - Low-Rank approximations
  - Downsampling the sequence length n



Images taken from [5, 6]

Transformers: GPT Models

- **G**enerative **P**re-Trained **T**ransformers:
  - First, unsupervised pre-training on predicting the next token in a sequence

$$L = \sum_i \log P_\theta\left(\boldsymbol{x}^{(i)} \middle| \boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(i-k)}\right)$$

  - Second, task-specific fine-tuning (classification, chatbots, …)
- GPT-n models use large text corpora (Data crawled from the internet, books, …)
- Strong performance because of large models and datasets: Loss and datasets compute follow a power-law
  - e.g. GPT-3 has ~**175B** parameters

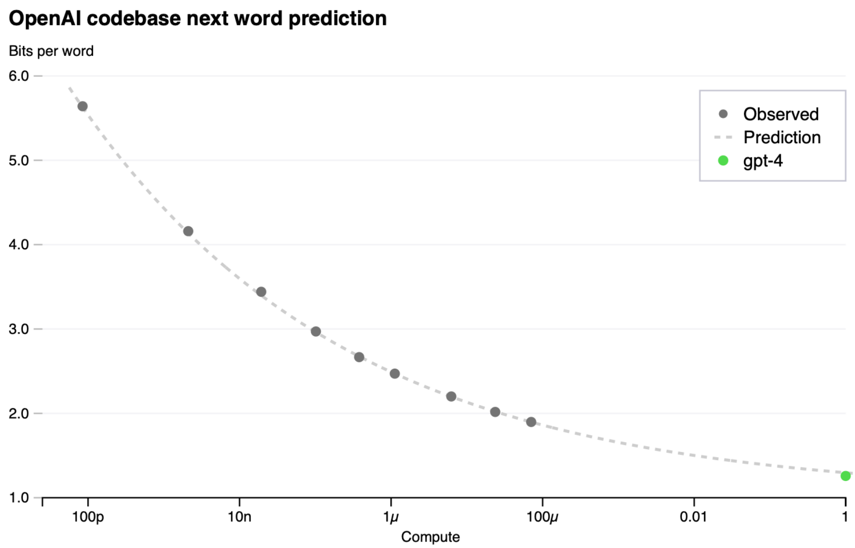


Image taken from [7]

Roadmap

- Chapter: Temporal Data / Sequential Data
  1. Autoregressive Models
  2. Markov Chains
  3. Hidden Markov Models
  4. **Neural Network Approaches**
     a) Word Vectors
     b) RNNs
     c) Non-Recurrent Models (ConvNets, Transformer)
     d) Structured State Space Models
  5. Temporal Point Processes

Introduction

- Seen two types of neural sequence models so far: **RNN** vs. **non-recurrent**
- Computationally, they have complementary advantages:

	Training	Inference
Transformers	**Fast!** (parallelizable)	**Slow...** (scales **quadratically** with sequence length)
RNNs	**Slow...** (not parallelizable)	**Fast!** (scales **linearly** with sequence length)

- Is it somehow possible to get the best of both worlds?

Image taken from [8]

Recurrence as a convolution

Consider a simple, purely linear RNN-type architecture

with input $u_k \in \mathbb{R}$, output $y_k \in \mathbb{R}$, and $\overline{\boldsymbol{A}} \in \mathbb{R}^{N\times N}, \overline{\boldsymbol{B}} \in \mathbb{R}^{N\times 1}, \overline{\boldsymbol{C}} \in \mathbb{R}^{1\times N}$

(“**state-space model**”):

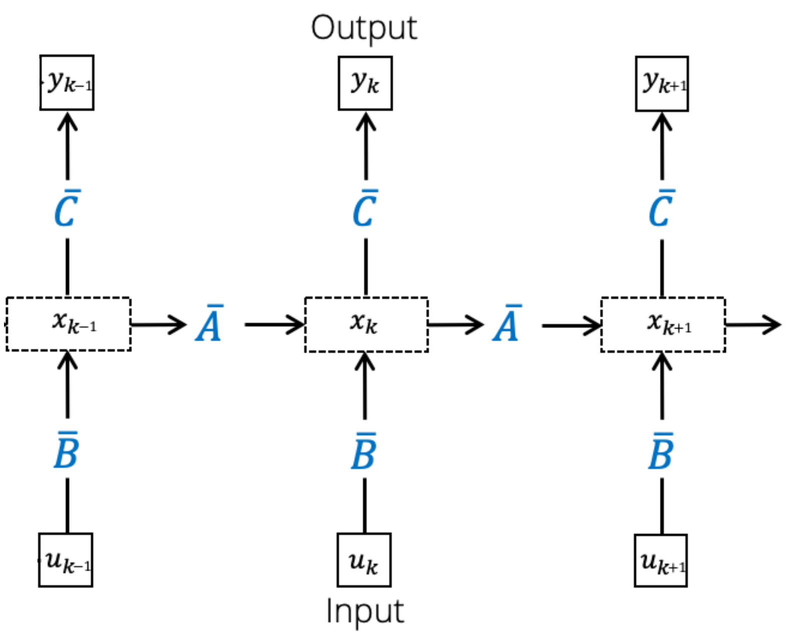


Image taken from [9]

$$x_0 = \overline{\boldsymbol{B}}u_0, \quad x_1 = \overline{\boldsymbol{A}}\overline{\boldsymbol{B}}u_0 + \overline{\boldsymbol{B}}u_1, \dots$$

$$\Downarrow$$

$$y_k = \overline{\boldsymbol{C}}\overline{\boldsymbol{A}}^k\overline{\boldsymbol{B}}u_0 + \cdots + \overline{\boldsymbol{C}}\overline{\boldsymbol{A}}\overline{\boldsymbol{B}}u_{k-1} + \overline{\boldsymbol{C}}\overline{\boldsymbol{B}}u_k$$

$$\Downarrow$$

Equivalent to convolution with (sequence-long) kernel $\overline{\boldsymbol{K}} = [\overline{\boldsymbol{C}}\overline{\boldsymbol{B}}, \overline{\boldsymbol{C}}\overline{\boldsymbol{A}}\overline{\boldsymbol{B}}, \dots, \overline{\boldsymbol{C}}\overline{\boldsymbol{A}}^{L-1}\overline{\boldsymbol{B}}]$

“Naïve” Idea:

Train in convolution representation,

do inference in recurrent representation!

Efficient parametrization of $\overline{\boldsymbol{K}}$

$$\overline{\boldsymbol{K}} = [\overline{\boldsymbol{C}}\overline{\boldsymbol{B}}, \overline{\boldsymbol{C}}\overline{\boldsymbol{A}}\overline{\boldsymbol{B}}, \dots, \overline{\boldsymbol{C}}\overline{\boldsymbol{A}}^{\boldsymbol{L-1}}\overline{\boldsymbol{B}}]$$

- Problem with "naïve" approach: parametrizing $\overline{\boldsymbol{K}}$ directly via $\overline{\boldsymbol{K}}(\overline{\boldsymbol{A}}, \overline{\boldsymbol{B}}, \overline{\boldsymbol{C}})$ requires taking $L-1$ powers of $\overline{\boldsymbol{A}}$ ($\mathcal{O}(N^2L)$ steps), effectively unrolling the recurrence ⇒ **no** computational benefit just yet…
- Precomputing once + caching not possible as $\overline{\boldsymbol{A}}, \overline{\boldsymbol{B}}, \overline{\boldsymbol{C}}$ are trainable
- SSMs are an old concept – efficient parametrizations for changing between recurrent / convolutional representations contribute to recent successes:
  - [10]: Express $\overline{\boldsymbol{A}}$ in another basis to get $\overline{\boldsymbol{K}}$ in only $\mathcal{O}(N+L)$ steps (+ log factors)
  - [11]: Extract approximate SSM from explicitly parametrized convolution layer

Optimal initialization of $\overline{A}$ – continuous-time view

- RNN model with generic $\overline{\boldsymbol{A}}$ may suffer from vanishing / exploding gradients
- [10]: same output as continuous-time SSM,

$$x'(t) = \boldsymbol{A}x(t) + \boldsymbol{B}u(t)$$
$$y'(t) = \boldsymbol{C}x(t)$$

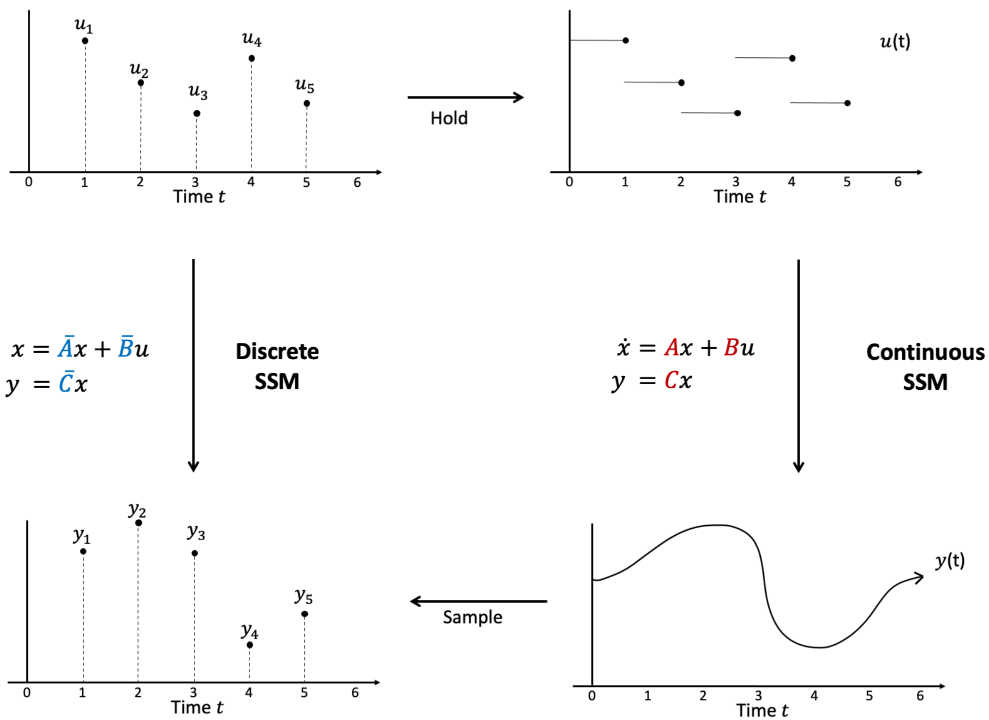


Image taken from [9]

$$\overline{\boldsymbol{A}} = \left(\boldsymbol{I} - \frac{\Delta}{2}\cdot\boldsymbol{A}\right)^{-1}\left(\boldsymbol{I} + \frac{\Delta}{2}\cdot\boldsymbol{A}\right),$$
$$\overline{\boldsymbol{B}} = \left(\boldsymbol{I} - \frac{\Delta}{2}\cdot\boldsymbol{A}\right)^{-1}\Delta\boldsymbol{B},\quad \overline{\boldsymbol{C}} = \boldsymbol{C},$$

if we convert inputs $\{u_k\}$ to piecewise-constant signal $t \mapsto u(t)$ with step-size Δ and subsample $y(t)$

Advantage: admits new framework to analyze input memorization [12]

Optimal initialization of $\bar{A}$ – “Structured” SSMs

- RNN model with generic $\bar{A}$ may suffer from vanishing / exploding gradients
- [12] show (in continuous SSM representation): **HiPPO matrix** produces a hidden state that **memorizes** its history by tracking Legendre polynomial coefficients → stable gradient flow

HiPPO Matrix $$\mathbf{A}_{nk} = \begin{cases} (2n+1)^{1/2}(2k+1)^{1/2} & \text{everything below the diagonal} \\ n+1 & \text{the diagonal} \\ 0 & \text{everything above the diagonal} \end{cases}$$

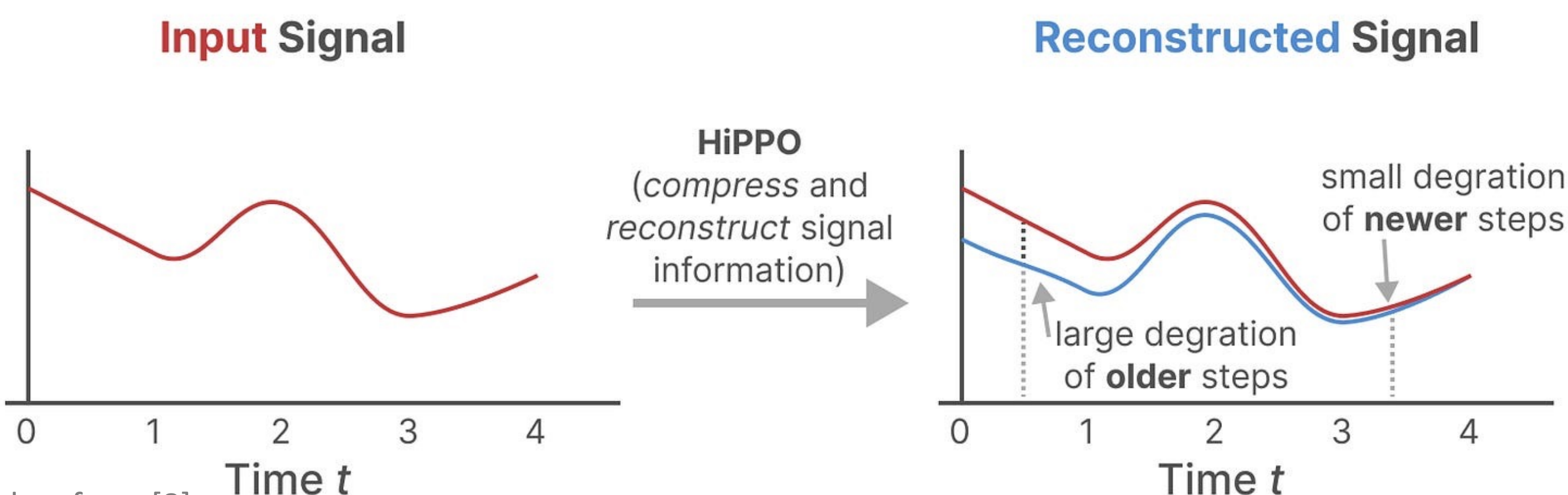


Image taken from [8]

Putting things together: S4 model

- Three equivalent representations of SSMs:
  - Continuous (justifies HiPPO matrices $\boldsymbol{A}$; get matrix $\overline{\boldsymbol{A}}$ from discretization)
  - Recurrent (unlimited context window, linear-time inference)
  - Convolutional (parallelizable training)
- S4 (**S**tructured **S**tate **S**pace for **S**equences) model [10] = underlying continuous SSM + HiPPO matrices (long-range modeling) + efficient parametrization of discrete (i.e., convolutional + recurrent) representations

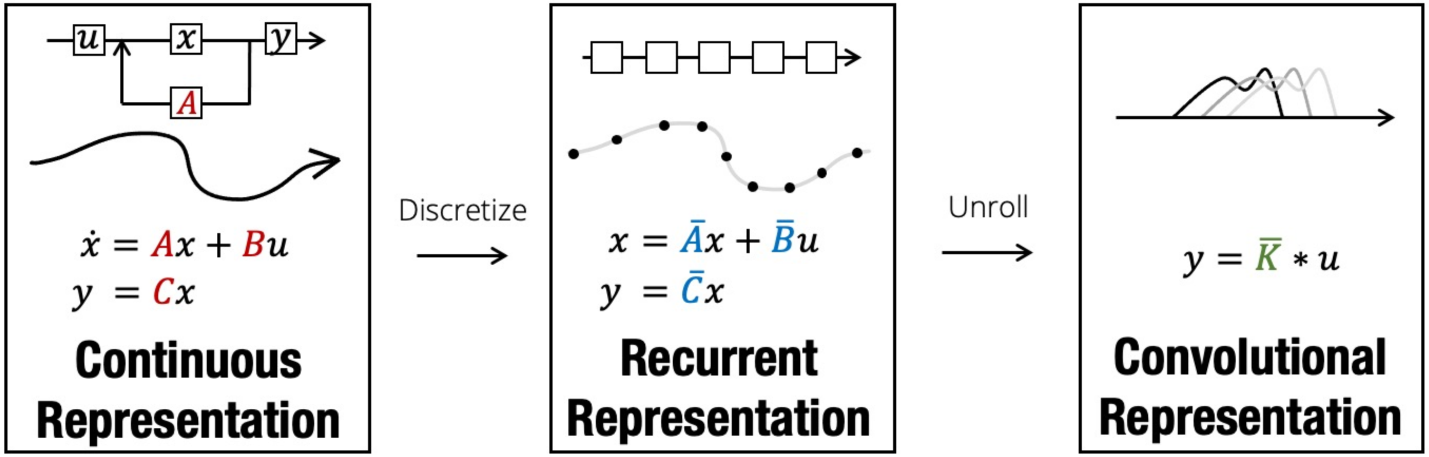


Image taken from [9]

Putting things together: S4 model

- General S4 layer also includes skip connection (with $\boldsymbol{D}, \overline{\boldsymbol{D}} \in \mathbb{R}^{1\times 1}$):

$$x_k = \overline{\boldsymbol{A}} x_{k-1} + \overline{\boldsymbol{B}} u_k \qquad\qquad x'(t) = \boldsymbol{A}x(t) + \boldsymbol{B}u(t)$$
$$y_k = \overline{\boldsymbol{C}} x_k + \overline{\boldsymbol{D}} u_k \qquad\qquad y'(t) = \boldsymbol{C}x(t) + \boldsymbol{D}u(t)$$

(discrete) (continuous)

- So far, this only defines map between 1D sequences $\mathbb{R}^L \to \mathbb{R}^L$
- Given H hidden features, simply run H parallel SSMs, then mix features with position-wise linear layer (sharing parameters across positions)
- Stack this + apply position-wise nonlinearities in between to obtain deep S4 architecture: **nonlinear** sequence-to-sequence map of shape `(batch size, sequence length, hidden dimension)`
  → Same interface as for CNN, RNN, Transformer
- Deep S4 is essentially a CNN, but parametrizing **global** (i.e., sequence-length) filters acting separately along each hidden dimension

Putting things together: S4 model

- Set a substantial SOTA in long-range (1K-16K tokens) sequence modeling

Model	ListOps	Text	Retrieval	Image	Pathfinder	Path-X	Avg
Random	10.00	50.00	50.00	10.00	50.00	50.00	36.67
Transformer	36.37	64.27	57.46	42.44	71.40	✗	53.66
Local Attention	15.82	52.98	53.39	41.46	66.63	✗	46.71
Sparse Trans.	17.07	63.58	59.59	44.24	71.71	✗	51.03
Longformer	35.63	62.85	56.89	42.22	69.71	✗	52.88
Linformer	35.70	53.94	52.27	38.56	76.34	✗	51.14
Reformer	37.27	56.10	53.40	38.07	68.50	✗	50.56
Sinkhorn Trans.	33.67	61.20	53.83	41.23	67.45	✗	51.23
Synthesizer	36.99	61.68	54.67	41.61	69.45	✗	52.40
BigBird	36.05	64.02	59.29	40.83	74.87	✗	54.17
Linear Trans.	16.13	65.90	53.09	42.34	75.30	✗	50.46
Performer	18.01	65.40	53.82	42.77	77.05	✗	51.18
FNet	35.33	65.11	59.61	38.67	77.80	✗	54.42
Nyströmformer	37.15	65.52	79.56	41.58	70.94	✗	57.46
Luna-256	37.25	64.57	79.29	47.38	77.72	✗	59.37
S4	**58.35**	**76.02**	**87.09**	**87.26**	**86.05**	**88.10**	**80.48**

Image taken from [9]

Image taken from [9]

Questions – NN

1. In an RNN, the hidden state at a given time influences all hidden states into the future. However, an RNN cannot model long-term dependencies. Why?

2. What is the receptive field of a causal convolution and dilated convolution with n layers ?

References

- [1] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." In *Advances in NIPS*, pp. 1097-1105. 2012.
- [2] Van Den Oord, Aäron, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W. Senior, and Koray Kavukcuoglu. "WaveNet: A generative model for raw audio." *SSW* 125 (2016).
- [3] Ashish Vaswani et al., "Attention Is All You Need" NIPS 2017
- [4] Jalammar blog, http://jalammar.github.io/illustrated-transformer/
- [5] Cui, Baiyun, et. al. "Fine-tune BERT with sparse self-attention mechanism." *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*. 2019.
- [6] Paul Michel, https://blog.ml.cmu.edu/2020/03/20/are-sixteen-heads-really-better-than-one/
- [7] OpenAI. GPT-4 Technical Report. arXiv preprint

References

- [8] Maarten Grootendorst, https://newsletter.maartengrootendorst.com/p/a-visual-guide-to-mamba-and-state
- [9] Hazy Research Blog, https://hazyresearch.stanford.edu/blog/2022-01-14-s4-3
- [10] Gu, Albert, Goel, Karan, Ré, Christopher. Efficiently Modeling Long Sequences With Structured State Spaces, ICLR 2022
- [11] Massaroli et al. Laughing Hyena Distillery: Extracting Compact Recurrences From Convolutions, NeurIPS 2023
- [12] Gu, Albert, Dao, Tri, Ermon, Stefano, Rudra, Atri, Ré, Christopher. HiPPO: Recurrent Memory with Optimal Polynomial Projections, NeurIPS 2020