Transformers

- powerful memory, we can access specific parts of the input via KV
- parallelize during training
- slow at test time
  - quadratic complexity in L RNNs:
  - memory bottleneck
  - not parallelizable at training time
  - fast at test time

RNN:
\[ h_t = f (W_h h_{t-1} + W_e c_t) \]

\([h\text{idden state}], [c\text{urrent input token}]\)

Linear RNN:
\[ h_t = A h_{t-1} + B c_t \]

- no nonlinearity \(\Rightarrow\) lower expressivity
- what happens if we put a nonlinear feed-forward layer on top?
\[ z_t = \mathbf{f}(C h_t + D c_t) \]

**Ideas:** all nonlinearities are above the linear recurrent layer.

**Linear RNN:**
\[ h_1 = B c_1 \]
\[ h_2 = A h_1 + B c_2 \]
\[ h_3 = A h_2 + B c_3 \]

\[ h_2 = A B c_1 + B c_2 \]
\[ h_3 = A^2 B c_1 + A B c_2 + B c_3 \]
\[ h_4 = A^3 B c_1 + A^2 B c_2 + \ldots \]
\[ h_t = \sum_{k=0}^{t-1} A^k B c_{t-k} \]
how do we parallelize this?

⇒ treat this as a convolution, and design kernel

\[ [B, AB, A^2B, \ldots, A^{k-1}B] \]

slide over sequence

⇒ in modern papers: parallel scan

let's consider the simple case of the cumulative sum operation

\[ Y_k = \sum_{i=1}^{k} x_i \]

3, 1, 7, 0, 4, 1, 6, 3
3, 4, 11, 11, 15, 16, 22, 25

sequentially:

\[ y_t = x_t + y_{t-1} \]
Consider:

- **25**
- **14**
- **5**
- **9**

**upsweep**

- **11**
- **7**
- **4**
- **6**

**downsweep**: root starts at 0

- Left child gets the value of the root.
- Right child gets the sum of the root + upswing(left child).
Let's consider the operation:

\[(a, b) \odot (a', b') = (a' a, a' b + b')\]
$(A^4, \ A^3Bc_o + A^2Bc_1 + ABc_2 + Bc_3)$

$(A^2, \ Abc_o + Bc_1)$

$(A, \ Bc_o)$  $(A, \ Bc_1)$

$(A^2, \ Abc_2 + Bc_3)$

$(A, \ Bc_2)$  $(A, \ Bc_3)$