Lecture 16

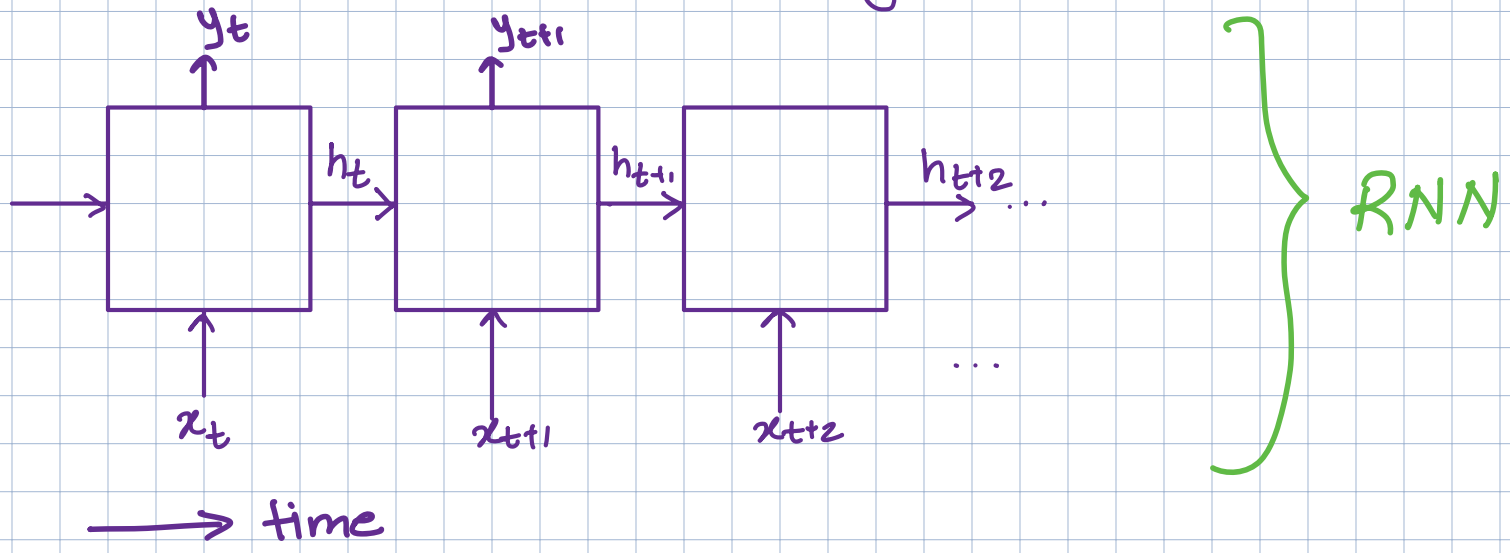
State-space models.

Problem with traditional RNN's

① Hard to parallelize training

② Hard to learn long-range dependancies

Key idea: Eliminate "horizontal" non-linearity



Traditional RNN: $\vec{h}_{t+1} = \sigma\left(A\vec{h}_t + B\vec{x}_t + \vec{b}\right)$

New : $\vec{h}_{t+1} = A\vec{h}_t + B\vec{x}_t$ Just eliminate!

Learned weights A, B, C (not dependant on input)

$$\vec{h}_t = A\vec{h}_{t-1} + B\vec{x}_t$$
$$\vec{y}_t = C\vec{h}_t + D\vec{x}_t$$

Linear-time invariant system?

Control Systems

State space view v/s input output view.

Unroll:

$$\vec{h}_0 = B\cdot\vec{x}_0$$
$$\vec{h}_1 = AB\vec{x}_0 + B\vec{x}_1$$
$$\vec{h}_2 = A^2B\vec{x}_0 + AB\vec{x}_1 + B\vec{x}_2$$
$$\vdots$$
$$\vec{h}_k = A^kB\vec{x}_0 + \cdots + AB\vec{x}_{k-1} + B\vec{x}_k$$
$$\vec{y}_k = CA^kB\vec{x}_0 + \cdots + CB\vec{x}_k + D\vec{x}_k$$

only depends on previous inputs! Not on hidden states!

Given Learned weights A, B and C, this is just a convolution!

$\Rightarrow$ Computing $\vec{y}_k$ can be paralledized.

At position t, there are t terms in the sequence.

Requires $O(t)$ compute. $\quad 1, 2, \cdots, t, \cdots T$

Max sequence length is T (the largest that fits in memory).

$$1 + 2 + 3 + \cdots + T = \frac{T(T+1)}{2} = O(T^2)$$

$\Rightarrow$ Total $O(T^2)$ compute.

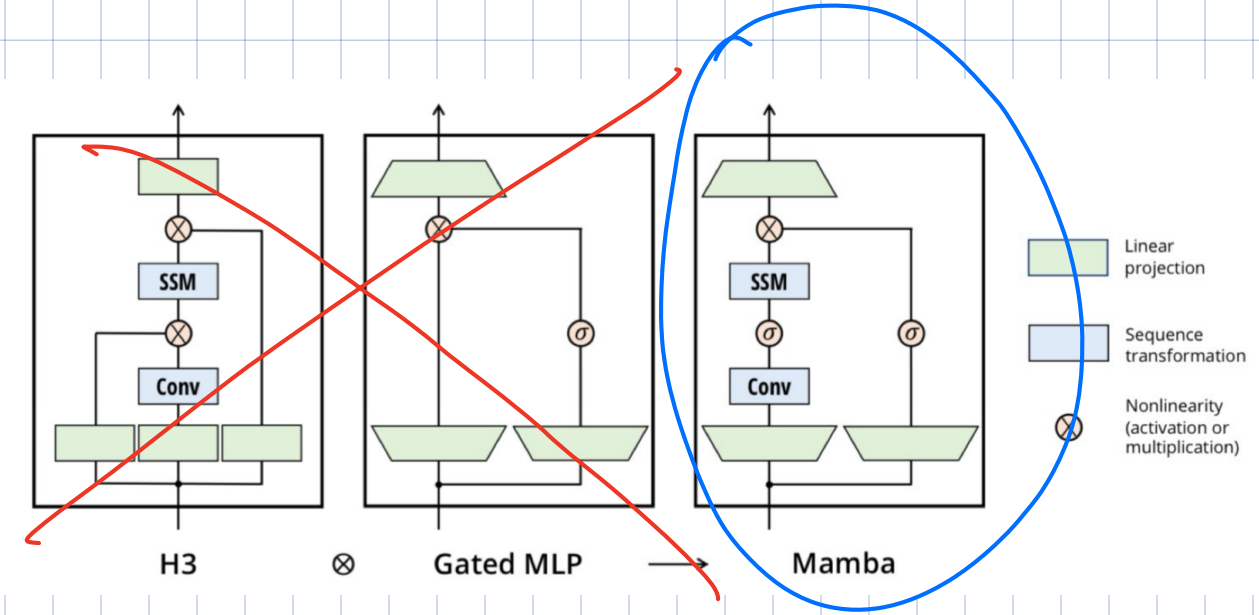
Can we improve this? $\rightarrow$ Yes.

Fast Fourier Transform! $\quad O(T \log T)$

↳ Dynamic Programming

But what about expressive power?

→ Add a "vertical" non-linearity. i.e. an MLP after the state-space layer.



Summary:

Traintime: Convolutions. No sequential non-linearity.

Inference: All past information is contained in the state.

What about CA^kB?

$A^k \rightarrow$ adds depth to computation graph.

$\rightarrow$ must do sequentially, can't parallelize.

What can we do to speed it up?

$\rightarrow$ Add structure!

$\rightarrow$ Easiest A to exponentiate? Diagonal A

Coming back to state-space representation.

$$\vec{h}_t = A\vec{h}_{t-1} + B\vec{x}_t$$
$$\vec{y}_t = C\vec{h}_t + D\vec{x}_t$$

$\vec{y}_t$: output

$\vec{x}_t$: input.

$\vec{y}_t$: d_y dimensions

$\vec{x}_t$: d_x dimensions.

h_t : d_h dimensions

This is our "memory".

$$\vec{y}_t = \begin{bmatrix} y_t[1] \\ y_t[2] \\ \vdots \\ y_t[d_y] \end{bmatrix}$$

$\vec{y}_t$ has d_y components.

Think of time-evolution / sequence on a component by component basis.

$y[k]$: $y_1[k]$, $y_2[k]$, $\cdots$ $y_t[k]$ k^{th} component.

time 1 time 2 time t.

This is a sequence.

d_y such sequences.

Similarly, for x, we have d_x such sequences.

$y_t[k]$ is a linear combination of all components of all past times of x

ie. of $x_1[1]$ $x_2[1]$ $\cdots$ $x_t[1]$ ← each a sequence.

$x_1[2]$ ⋮ ⋮ ⋮

$x_1[d_x]$ $x_2[d_x]$ $x_t[d_x]$

# of sequences → d_x

$$Seq(y[k]) = \sum_{l=1}^{d_x} F_{k,l} * Seq(x[l])$$

convolution ← $F_{k,l}$

appropriate filter

Example:

$$\vec{h}_{t+1} = \begin{bmatrix} 1 & 0 \\ 0 & \alpha \end{bmatrix} \vec{h}_t + \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \vec{x}_{t+1} \qquad \vec{h}_0 = \vec{0}$$

$$\vec{y}_t = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ 0 & 1 \end{bmatrix} \vec{h}_t$$

$$h_{t+1}[1] = h_t[1] + x_{t+1}[1] + x_{t+1}[2] \rightarrow \text{running sum of all components}$$

$$= \sum_{i=1}^{t+1} x_i[1] + x_i[2]$$

$$h_{t+1}[2] = \alpha h_t[2] + x_{t+1}[2] \rightarrow \text{exp. moving avg of second component.}$$

$$= \sum_{i=1}^{t+1} \alpha^{t-i+1} x_i[2]$$

$$y_t[1] = \frac{1}{2} h_t[1] + \frac{1}{2} h_t[2]$$

$$= \sum_{i=1}^{t} \frac{1}{2} x_i[1] + (1+\alpha^{t-i+1}) x_i[2]$$

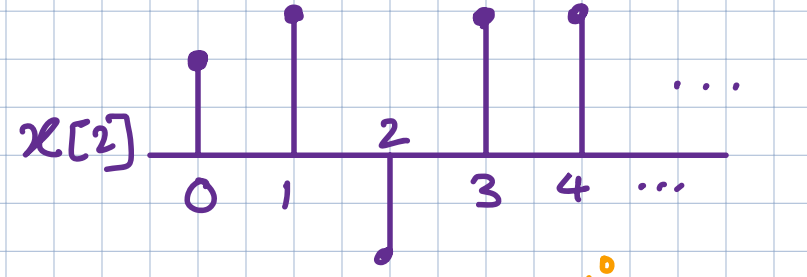
$$= \sum_{i=1}^{t} \frac{1}{2} x_i[1] + \sum_{i=1}^{t} (1+\alpha^{t-i+1}) x_i[2]$$

$$y_t[2] = h_t[2] = \sum_{i=1}^{t} \alpha^{t-i+1} x_i[2]$$

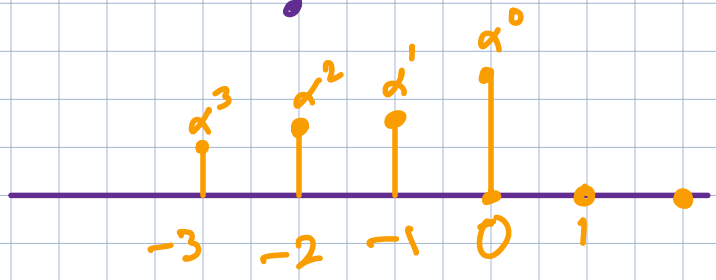
Why are these convolutions?

Start with $y_t[2]$.

$$y_t[2] = \sum_{i=1}^{t} \alpha^{t-i+1} x_i[2] = \sum_{i=1}^{t} \alpha^{t-i+1} x_i[2]$$



Filter $F_{2,2}$



only uses the past + current input

Now $y_t[1]$

$$y_t[1] = \underbrace{\sum_{i=1}^{t-1} \frac{1}{2} x_i[1]}_{\text{Convolution 1}} + \underbrace{\sum_{i=1}^{t-1} \left(\frac{1}{2} + \alpha^{t-i}\right) x_i[2]}_{\text{Convolution 2.}}$$

Can we further simplify?

Depthwise | Channel-wise convolutions?

$$\vec{h}_{t+1} = A\vec{h}_t + B\vec{x}_t$$

$$\text{Seq}(Y[k]) = \sum_{l=1}^{d_x} F_{k,l} * \text{Seq}(x[l])$$

# of sequences

convolution

appropriate filter

If $d_x = d_y = 100$, we have 10,000 convolutions!

To reduce convolutions, in conv nets we did depthwise (channelwise) convs!

Can we do this here?

A diagonal. What is the structure for B and C?.

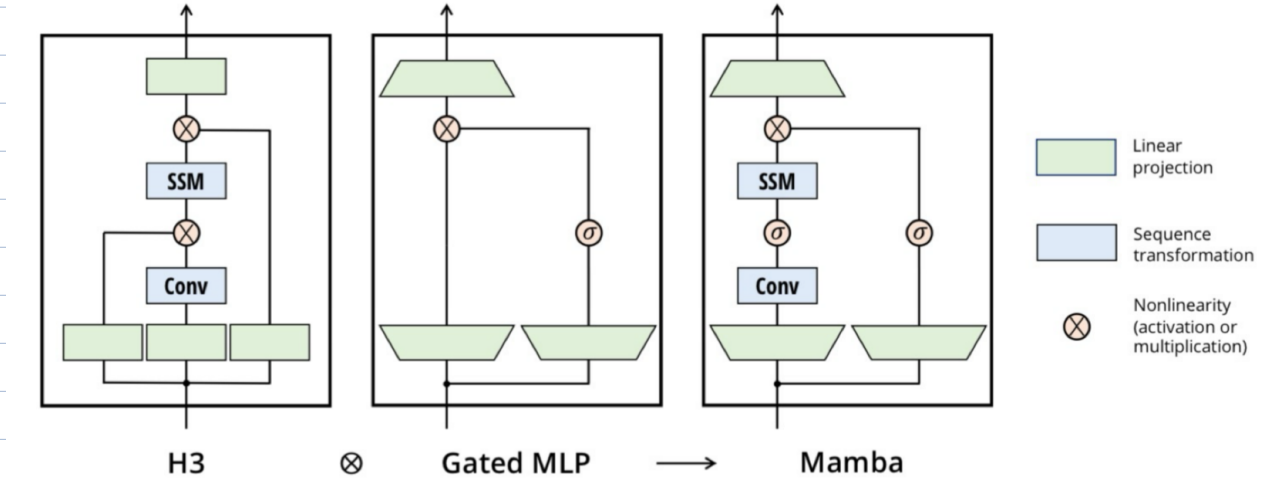
Block diagonal!

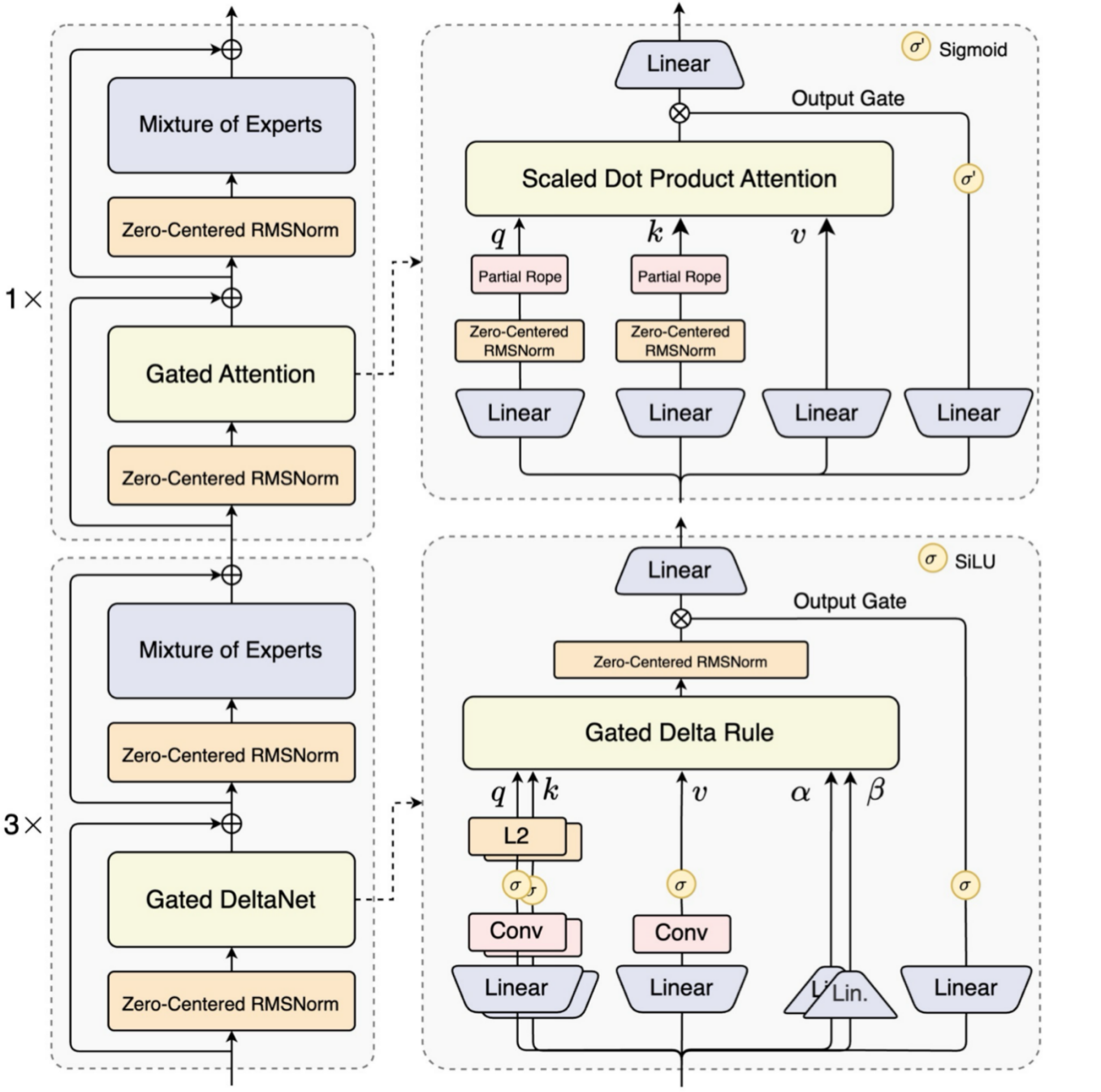
$$\vec{h}_{t+1} = \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_N \end{bmatrix} \vec{h}_t + \begin{bmatrix} b_{11} & 0 \\ b_{21} & 0 \\ 0 & 0 \\ & b_{33} \; 0 \\ & b_{43} \; 0 \\ & \vdots \end{bmatrix} \vec{x}_t$$

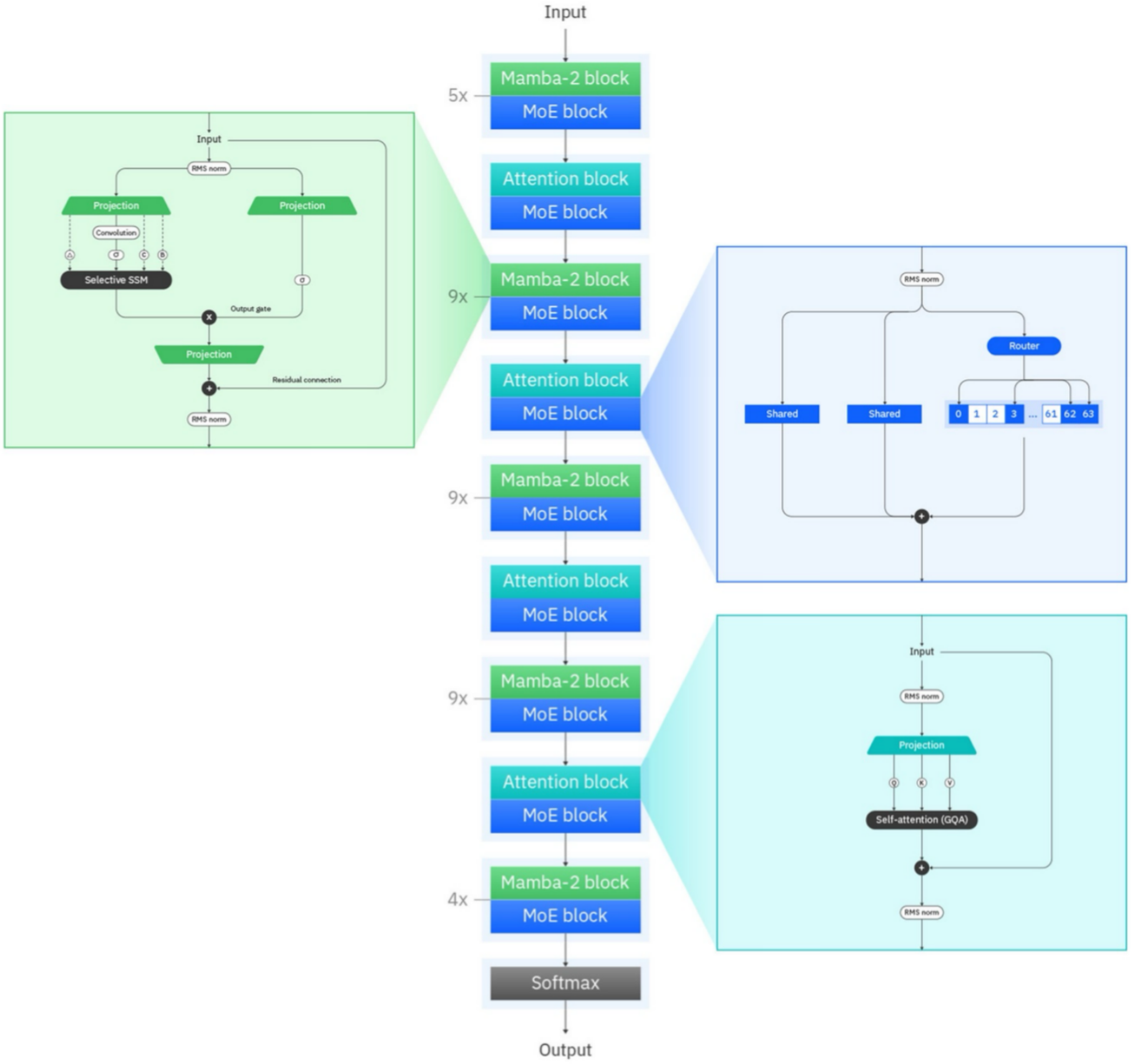
$$\begin{bmatrix} y_t[1] \\ y_t[2] \\ \vdots \\ y_t[d_y] \end{bmatrix} = \square \; \vec{x}_0 + \square \; \vec{x}_1 + \square + \cdots$$

$$\vec{y}_t = \begin{bmatrix} y_t[1] \\ y_t[2] \\ \vdots \\ y_t[d_y] \end{bmatrix} \qquad \vec{x}_t = \begin{bmatrix} x_t[1] \\ \vdots \\ x_t[d_x] \end{bmatrix}$$

$$\vec{y}_t = CA^t B\vec{x}_0 + \cdots + CB\vec{x}_t + D\vec{x}_t$$


SSM
Conv
SSM
Conv
Linear projection
Sequence transformation
Nonlinearity (activation or multiplication)
H3
⊗
Gated MLP
⟶
Mamba


1×
Mixture of Experts
Zero-Centered RMSNorm
Gated Attention
Zero-Centered RMSNorm
3×
Mixture of Experts
Zero-Centered RMSNorm
Gated DeltaNet
Zero-Centered RMSNorm
Linear
Output Gate
Sigmoid
Scaled Dot Product Attention
q
k
v
Partial Rope
Partial Rope
Zero-Centered RMSNorm
Zero-Centered RMSNorm
Linear
Linear
Linear
Linear
Linear
Output Gate
SiLU
Zero-Centered RMSNorm
Gated Delta Rule
q
k
v
α
β
L2
Conv
Conv
Linear
Linear
Lin.
Linear


Input
5x
Mamba-2 block
MoE block
Attention block
MoE block
9x
Mamba-2 block
MoE block
Attention block
MoE block
9x
Mamba-2 block
MoE block
Attention block
MoE block
9x
Mamba-2 block
MoE block
Attention block
MoE block
4x
Mamba-2 block
MoE block
Softmax
Output
Input
RMS norm
Projection
Projection
Convolution
Selective SSM
Output gate
Projection
Residual connection
RMS norm
RMS norm
Router
Shared
Shared
0 1 2 3 ... 61 62 63
Input
RMS norm
Projection
Q K V
Self-attention (GQA)
RMS norm