Supplementary Material

September 14, 2024

A Justification for Using Mamba Architecture

State Space Models (SSMs) are commonly employed as linear time-invariant systems that transform a one-dimensional input stimulus $x(t) \in \mathbb{R}^L$ through intermediary implicit states $h(t) \in \mathbb{R}^N$ to an output $y(t) \in \mathbb{R}^L$. In mathematical terms, SSMs are typically described by linear ordinary differential equations (ODEs) (Equation 1), where the system is characterized by a set of parameters including the state transition matrix $A \in \mathbb{C}^{N\times N}$, the projection parameters $B, C \in \mathbb{C}^N$, and the skip connection $D \in \mathbb{C}^1$.

$$\begin{aligned} h'(t) &= Ah(t) + Bx(t) \\ y(t) &= Ch(t) + Dx(t) \end{aligned} \tag{1}$$

If we directly integrate $\dot{x}(t) = Ax(t) + Bu(t)$, the result is as follows:

In this case, the integral term contains the term $x(\tau)$ itself. Since we are dealing with a discrete system, we cannot obtain all the values of $x(\tau)$ within a continuous time interval $0 \to t$, hence we cannot compute the integral result.

A.1 Discretization

For discrete systems, we aim to transform the integral expression (4) into the following form:

$$x(k+1) = x(k) + \sum_{i=0}^{k} \left(Ax(i) + Bu(i)\right)\Delta t$$

To eliminate the $x(t)$ term in the expression for $\dot{x}(t)$, we typically construct a new function $\alpha(t)x(t)$, and simplify the corresponding derivative terms by differentiating this new function.

We differentiate $\alpha(t)x(t)$ as follows:

$$\frac{d}{dt}\left[\alpha(t)x(t)\right] = \alpha(t)\dot{x}(t) + x(t)\frac{d\alpha(t)}{dt}$$

Substituting equation 1 into the above and replacing $\dot{x}(t)$:

$$\frac{d}{dt}\left[\alpha(t)x(t)\right] = \alpha(t)\left(Ax(t) + Bu(t)\right) + x(t)\frac{d\alpha(t)}{dt}$$

Further rewriting, combining terms related to $x(t)$:

$$\frac{d}{dt}\left[\alpha(t)x(t)\right] = \left(A\alpha(t) + \frac{d\alpha(t)}{dt}\right)x(t) + B\alpha(t)u(t)$$

Since our goal is to eliminate the $x(t)$ term in the derivative, we set the coefficient of $x(t)$ to zero:

$$A\alpha(t) + \frac{d\alpha(t)}{dt} = 0$$

We can then obtain the expression for $\alpha(t)$:

$$\alpha(t) = e^{-At}$$

Substituting the expression for $\alpha(t)$ into the previous equation, we get:

$$\frac{d}{dt}\left[e^{-At}x(t)\right] = Be^{-At}u(t)$$

Now, we have achieved the goal of eliminating $x(t)$ from the derivative term. Integrating $e^{-At}x(t)$:

$$e^{-At}x(t) = x(0) + \int_0^t e^{-A\tau}Bu(\tau)d\tau$$

Rearranging:

$$x(t) = e^{At}x(0) + \int_0^t e^{A(t-\tau)}Bu(\tau)d\tau$$

A.2 Parameter Definition

Define the sampling instants t_k and t_{k+1}, where k is the sampling index, and T is the sampling interval, i.e., $T = t_{k+1} - t_k$.

A.3 Discretization of the Integral Interval

In continuous-time integration, we typically have an integral interval, for example, from t to $t + \Delta t$. In discrete-time systems, we need to divide this interval into k equal-length subintervals, each with a length of T.

Within a particular subinterval, equation A.3 takes the form:

$$x(t_{k+1}) = e^{A(t_{k+1}-t_k)}x(t_k) + \int_{t_k}^{t_{k+1}} e^{A(t_{k+1}-\tau)}Bu(\tau)d\tau$$

A.4 Approximate Integration

For each subinterval, consider using numerical integration methods to approximate the integral. Here, we apply zero-order hold to $u(t)$, assuming it is constant between sampling instants t_k and t_{k+1}. Then, we can factor out $u(t_k)$ from the integral term as follows:

$$\int_{t_k}^{t_{k+1}} e^{A(t_{k+1}-\tau)}Bu(\tau)d\tau = \int_{t_k}^{t_{k+1}} e^{A(t_{k+1}-\tau)}d\tau \cdot Bu(t_k)$$

A.5 Construction of Discrete-time State Equation

Substituting the integral result into the previous equation and simplifying using $T = t_{k+1} - t_k$, we obtain:

$$x(t_{k+1}) = e^{AT}x(t_k) + \int_{t_k}^{t_{k+1}} e^{A(t_{k+1}-\tau)}d\tau \cdot Bu(t_k)$$

Introducing a new variable $\lambda = t_{k+1} - \tau$, we simplify the original integral to obtain:

$$x(t_{k+1}) = e^{AT}x(t_k) + Bu(t_k)\int_0^T e^{A\tau}d\tau$$

Here, the integral involves the matrix exponential, and the result is obtained from some references:

$$\int_0^T e^{A\tau}d\tau = A^{-1}(e^{AT} - I)$$

Finally, we arrive at the discrete-time state equation:

$$x(t_{k+1}) = e^{AT}x(t_k) + (e^{AT} - I)A^{-1}Bu(t_k)$$

$$\begin{aligned}
h_k &= \bar{A}h_{k-1} + \bar{B}x_k, \\
y_k &= \bar{C}h_k + \bar{D}x_k, \\
\bar{A} &= e^{\Delta A}, \\
\bar{B} &= (e^{\Delta A} - I)A^{-1}B, \\
\bar{C} &= C
\end{aligned}$$

The equation $x(t_{k+1}) = e^{AT}x(t_k) + (e^{AT} - I)A^{-1}Bu(t_k)$ implies that $x(t_k)$ represents a set of variables describing the current state y_k, while $u(t_k)$ represents the current input. However, there are two issues:

1. When using shallow networks, if $u(t_k)$ is too frequent, $x(t_k)$ may not adequately describe the state and may fluctuate severely, making convergence difficult.

2. If $x(t_k)$ is comprehensive, it will lead to an excessively large network size.

A.6 Solution Approach

The solution approach is as follows:

1. First, employ Convolutional Neural Networks (CNNs) for semantic compression to remove duplicate items and compress semantics to some extent.

2. Utilize SCConv for further feature reconstruction to make semantic features more distinct.

3. After compression through the above steps, the new input $u(t_k)$ should meet the following criteria:

   (a) Low repetition rate.

   (b) Decrease in $R[u(t_{k+1}) - u(t_k)]$ but without shrinking the feature space description. This allows $x(t_k)$ to better learn in a finite environment.

A.7 Conclusion

By integrating CNN-based semantic compression and SCConv-based feature reconstruction, the Mamba architecture effectively addresses the challenges of state representation and network scalability in discrete-time SSMs. This ensures efficient learning and convergence, making it ideal for deployment on UAV edge systems.