A Experiments compute resources

All experiments were conducted on a server running Ubuntu 22.04.4 LTS. The system is equipped with an Intel Xeon Gold 6133 processor featuring 80 logical cores at 3.0GHz, and 352 GB of RAM. The machine is configured with 8 NVIDIA GeForce RTX 4080 GPUs with 16GB of video memory each.

B Detail of Mamba

The following provides a detailed explanation of the internal computations within the Mamba module, beginning with the specification of the corresponding dimensional representations.

$$
\begin{aligned}
B &:= \text{Batch size},\\
S &:= \text{Sequence length},\\
D_m &:= \text{dimension of model},\\
D_{in} &:= \text{dimension of inner model}\\
&= expand \times D_m,\\
N &:= \text{dimension of hidden state},\\
Num_h &:= \text{number of heads},\\
H_d &:= \text{dimension of heads},
\end{aligned}
$$

The computations within the Mamba module can be divided into three components: pre-SSM, SSM, and post-SSM. The following presents a detailed derivation for each submodule.

$$\text{Input } \ u \in \mathbb{R}^{(\mathrm{B},\ \mathrm{S},\ \mathrm{D_m})},$$

Pre-SSM

$$
\begin{aligned}
zxBCdt &= \mathbf{Linear_{proj}}(u) \quad \in \mathbb{R}^{(\mathrm{B},\ \mathrm{S},\ 2\mathrm{D_{in}}+2\mathrm{N}+\mathrm{Num_h})},\\
(z, xBC, dt) &= zxBCdt \quad \in \mathbb{R}^{(\mathrm{B},\ \mathrm{S},(\ \mathrm{D_{in}},\ \mathrm{D_{in}}+2\mathrm{N},\ \mathrm{Num_h}))},\\
x_c B_c C_c &= \mathbf{Conv1d}(xBC) \quad \in \mathbb{R}^{(\mathrm{B},\ \mathrm{S},\ \mathrm{D_{in}}+2\mathrm{N})},\\
x_{cf} B_{cf} C_{cf} &= \mathbf{SiLU}(xBC) \quad \in \mathbb{R}^{(\mathrm{B},\ \mathrm{S},\ \mathrm{D_{in}}+2\mathrm{N})},\\
(x_{cf}, B_{cf}, C_{cf}) &= x_{cf} B_{cf} C_{cf} \quad \in \mathbb{R}^{(\mathrm{B},\ \mathrm{S},(\ \mathrm{D_{in}},\ \mathrm{N},\ \mathrm{N}))},\\
A_{init} &\sim \mathbf{Uniform}(A_{\text{min}}, A_{\text{max}}) \quad \in \mathbb{R}^{\mathrm{Num_h}},\\
A_{\log} &= \log(A_{init}) \quad \in \mathbb{R}^{\mathrm{Num_h}},\\
A &= -\exp(A_{\log}) \quad \in \mathbb{R}^{\mathrm{Num_h}},\\
dtb_{init} &= \exp(\mathbf{rand}(\mathrm{Num_h}) \cdot (\log(\text{dt\_max}) - \log(\text{dt\_min})) + \log(\text{dt\_min})) \quad \in \mathbb{R}^{\mathrm{Num_h}},\\
dt_{bias} &= dtb_{init} + \log(-\exp(-dtb_{init}) + 1) \quad \in \mathbb{R}^{\mathrm{Num_h}},\\
dt_{fb} &= \mathbf{softplus}(dt + dt_{bias}) \quad \in \mathbb{R}^{(\mathrm{B},\ \mathrm{S},\ \mathrm{Num_h})},\\
\widetilde{A} &= A \circ dt_{fb} \quad \in \mathbb{R}^{(\mathrm{B},\ \mathrm{S},\ \mathrm{Num_h})},\\
\widetilde{x} &= x_{cf}.\mathbf{reshape}(B, S, Num_h, H_d) \circ dt_{fb}.\mathbf{reshape}(B, S, Num_h, 1) \quad \in \mathbb{R}^{(\mathrm{B},\ \mathrm{S},\ \mathrm{Num_h},\ \mathrm{H_d})},\\
\widetilde{B} &= B_{cf}.\mathbf{reshape}(B, S, 1, N) \quad \in \mathbb{R}^{(\mathrm{B},\ \mathrm{S},\ 1,\ \mathrm{N})},\\
\widetilde{C} &= C_{cf}.\mathbf{reshape}(B, S, 1, N) \quad \in \mathbb{R}^{(\mathrm{B},\ \mathrm{S},\ 1,\ \mathrm{N})}.
\end{aligned}
$$

Here, $\mathbf{Linear_{proj}}$ denotes a linear transformation, $\mathbf{Conv1d}$ refers to a one-dimensional convolution, and $\mathbf{SiLU}$ represents the SiLU activation function. $\mathbf{Uniform}$ indicates sampling from a uniform distribution, while $\mathbf{softplus}$ denotes the Softplus activation function. $\mathbf{rand}$ refers to drawing a specified number of random values. The $\circ$ symbol denotes element-wise multiplication, and $\mathbf{reshape}$ indicates a dimensional transformation operation.

SSM

$$
\begin{aligned}
\widehat{A} &= \mathbf{Mask}_1 \circ \mathbf{repeat}(\widetilde{A}) \quad \in \mathbb{R}^{(\mathrm{B},\ \mathrm{Num_h},\ \mathrm{S},\ \mathrm{S})}, \\
A_{cumsum} &= \mathbf{cumsum}(\widehat{A}) \quad \in \mathbb{R}^{(\mathrm{B},\ \mathrm{Num_h},\ \mathrm{S},\ \mathrm{S})}, \\
L &= \exp(\mathbf{Mask}_2 \circ A_{cumsum}) \quad \in \mathbb{R}^{(\mathrm{B},\ \mathrm{Num_h},\ \mathrm{S},\ \mathrm{S})}, \\
P &= \mathbf{einsum}(\text{"}BSHN, BSHN \to BHSS\text{"}, \widetilde{C}, \widetilde{B}) \quad \in \mathbb{R}^{(\mathrm{B},\ 1,\ \mathrm{S},\ \mathrm{S})}, \\
M &= \mathbf{einsum}(\text{"}BHSS, BHSS \to BHSS\text{"}, L, P) \quad \in \mathbb{R}^{(\mathrm{B},\ \mathrm{Num_h},\ \mathrm{S},\ \mathrm{S})}, \\
y &= \mathbf{einsum}(\text{"}BHSS, BSHP \to BSHP\text{"}, M, \widetilde{x}) \quad \in \mathbb{R}^{(\mathrm{B},\ \mathrm{S},\ \mathrm{Num_h},\ \mathrm{H_d})}.
\end{aligned}
$$

The $\mathbf{Mask}_1$ operation is used to zero out the diagonal and upper-triangular elements of a matrix, retaining only the elements below the diagonal. The $\mathbf{Mask}_2$ sets the elements above the diagonal to negative infinity. The $\mathbf{cumsum}$ operation performs a cumulative sum along the first of the last two dimensions (i.e., across rows) in a matrix of shape (S, S), summing from top to bottom. The $\mathbf{einsum}$ operation denotes the Einstein summation convention, used for concise and flexible tensor contractions.

Post-SSM

$$
\begin{aligned}
D &= \mathbf{1}_{\mathrm{D_{in}}} \quad \in \mathbb{R}^{\mathrm{D_{in}}}, \\
\dot{y} &= y + x_{cf}.\mathbf{reshape}(B, S, Num_h, H_d) \times D.\mathbf{reshape}(D_{in}, 1) \quad \in \mathbb{R}^{(\mathrm{B},\ \mathrm{S},\ \mathrm{Num_h},\ \mathrm{H_d})}, \\
\ddot{y} &= \dot{y}.\mathbf{reshape}(B, S, D_{in}) \quad \in \mathbb{R}^{(\mathrm{B},\ \mathrm{S},\ \mathrm{D_{in}})}, \\
y_z &= \ddot{y} \cdot \mathbf{SiLU}(z) \quad \in \mathbb{R}^{(\mathrm{B},\ \mathrm{S},\ \mathrm{D_{in}})}, \\
y_{norm} &= y_z \cdot \frac{1}{\sqrt{\text{mean}(y_z^2, \text{axis} = -1) + \epsilon}} \cdot \mathbf{w} \quad \in \mathbb{R}^{(\mathrm{B},\ \mathrm{S},\ \mathrm{D_{in}})}, \\
y_{out} &= \mathbf{Linear}_{\mathbf{proj}}(y_{norm}) \quad \in \mathbb{R}^{(\mathrm{B},\ \mathrm{S},\ \mathrm{D_m})}.
\end{aligned}
$$

Here, $\mathbf{1}$ denotes a vector in which all elements are equal to 1.

$$\text{Output} \quad y_{out} \in \mathbb{R}^{(\mathrm{B},\ \mathrm{S},\ \mathrm{D_m})}.$$

C Data setup

C.1 Composite function task

Standard The total dataset comprises 300,000 samples, and each sequence has a fixed length of 8. Each anchor pair in the training set accounts for 5.6% of the total data, while each anchor pair in the test set constitutes 0.6%. The mapping between anchors and their associated functions is as follows: anchor 1 corresponds to a shift of $+5$, anchor 2 to $+1$, anchor 3 to -2, and anchor 4 to -8. During training, all 15 possible anchor pairs are included except for pair 43. Among these, all pairs except 34 are derived from the composition of their corresponding single-anchor functions. Notably, the function for pair 34 is manually set to -6, deviating from the correct compositional result of -10. The test set contains both symmetric and compositional instances of pair 43. The distinction between training and test data is governed by the position of the key token. For sequences of length 8, each position is associated with a congruence class modulo 8. In the training set, the key token is prohibited from appearing in the position whose modulo-8 class matches its own. For example, key 33 cannot appear in the first position, as it belongs to the congruence class 1 mod 8, which is aligned with the first index. Conversely, in the test set, each key token is required to occupy the position that corresponds exactly to its modulo-8 congruence class. This design ensures that the model has access to the full semantic range of tokens during training while preventing it from relying solely on positional memorization to generalize to the test set.

Full symmetry The total number of datasets is 300,000, and each sequence has a fixed length of 8. Each anchor pair in the training set accounts for 4.5% of the total, while each anchor in the test set accounts for 0.5% of the total. To ensure that the only possible solutions are symmetric,

we simultaneously utilize two sets of correspondences between anchors and functions. This setup prevents the model from simply solving for individual anchor functions and instead forces it to derive symmetric solutions by understanding the symmetry of anchor pairs. There are a total of five anchors: 0, 1, 2, 3, and 4. The 0 anchor is added to balance the data volume between the two function sets, thus avoiding model bias caused by disparities in data quantity. The first set of correspondences between anchors and functions is as follows: 0 corresponds to $+2$, 1 corresponds to $+5$, 2 corresponds to $+1$, 3 corresponds to -2, and 4 corresponds to -8. The second set of correspondences is: 0 corresponds to -9, 1 corresponds to $+6$, 2 corresponds to -7, and 3 corresponds to $+3$. All anchor pair functions are composed of either the first or the second set of correspondences. To facilitate the observation of symmetric solutions, anchor pairs 00, 11, 22, 33, and 44 are excluded from this task. Thus, there are a total of 20 anchor pairs in this task. The anchor pairs following the first set of correspondences are 01, 02, 10, 20, 14, 41, 23, 32, 34, 43, a total of 10 pairs. Among them, there are 4 1s, 4 ones, 4 twos, 4 threes and 4 fours. This design ensures that each anchor has the same amount of data under the two corresponding methods, and the two corresponding methods also have the same amount of data. The anchor pairs following the second set of correspondences are 03, 30, 04, 40, 12, 21, 13, 31, 24, 42. Only 43 is not included in the training set. The purpose of this task is to observe whether the model can discover the symmetry among all anchor pairs and thus output symmetric solutions. The distinction between the training set and the test set in the dataset is based on the position of the key. For sequences of length 8, each position corresponds to a congruence class modulo 8. In the training set, for each key, the key does not fall within the congruence class corresponding to its position. For example, the key 33 cannot appear in the first position of the sequence because it would then be in the congruence class modulo 8 of 1, which corresponds to that position. In contrast, in the test set, each key must exactly fall within the congruence class corresponding to its position. This setup allows the model to learn the meaning of all tokens while preventing it from relying solely on memorization to generalize to the test set.

C.2 Inverse sequence matching task

The total dataset consists of 100,000 samples. The sequence length varies with the number of layers in the Mamba model. Each sequence is generated from a generation set consisting of three distinct elements. It contains five different permutations of the elements in the generation set, where each permutation is followed by a random token that does not belong to the generation set. Each such permutation is referred to as a key sequence.

One of these five sequences is randomly selected and reversed to form the query sequence, which is appended to the end of the original sequence. Between the key sequences and the query sequence, a number of randomly generated tokens—equal to the convolutional receptive field of Mamba are inserted to prevent the model from retrieving information through convolution. Therefore, the total sequence length can be formally expressed as follows:

$$S = 5 \times (3 + 1) + 3 \times \mathbf{Num}_{layer} + 3.$$

Here, $\mathbf{Num}_{layer}$ denotes the number of layers in the Mamba model. For example, in the case of a two-layer Mamba model, the sequence length is 29. To ensure a fair comparison, Mamba and Transformer models with the same number of layers are trained on identical sequence configurations.

The training set constitutes 80% of the total dataset, while the test set and the out-of-distribution (OOD) set each account for 10%. In the training set, the three elements in the generating set do not all belong to the same congruence class modulo 3. In contrast, in the test set, all three elements in the generating set belong to the same congruence class modulo 3. For both the training and test sets, all numerical values in the sequences are drawn from the range 20 to 100. In the OOD set, however, all sequence elements are drawn from the range 101 to 200, and the generation set is not subject to any congruence constraints.

D Training setup

Unless otherwise specified, all tasks in this work adopt the following training parameter settings. The learning rate is initially set to 1e-5 at the start of training, warmed up to 25 times its initial value within 10 epochs, and then decreases to 1e-5 via cosine decay at 200 epochs. The training optimizer is AdamW with parameters set as $\beta_1 = 0.9$, $\beta_2 = 0.999$, $eps = 1e - 8$, and weight decay=1e-2.

Meanwhile, gradient clipping is applied with a maximum norm of 1. For composite function tasks, the batch size is set to 2048, while for inverse sequence matching tasks, the batch size is 1024. The loss function used for training is the cross - entropy loss function, which is calculated only for the last token of the model output sequence.

E Experiment detail and result

E.1 Composite function task

E.1.1 Phase diagram of Mamba on the composite function task

Mamba configurations The dimension of model is set to 32, the dimension of hidden state is set to the default value of 128, and the expansion factor is set to the default value of 2, and the activation function employed throughout the model is the Sigmoid Linear Unit (SiLU). To facilitate clear observation, all experiments are conducted using a single head. The convolution kernel length is set to its default value of 4. The values reported at different positions represent the mean results obtained using three fixed random seeds. Across these positions, the only variations are in the model's depth or initialization.

E.1.2 Experiments with all-one convolution

Mamba configurations To investigate whether the asymmetry of convolution is responsible for Mamba's difficulty in learning the symmetric solution, we select a model configuration under which Mamba successfully learns the compositional solution but fails to learn the symmetric one. Specifically, we use a five-layer Mamba model with small initialization($\gamma = 1$), while all other settings are consistent with those used in the phase diagram experiments.

E.1.3 Experiments with positional encoding

Mamba configurations To examine whether Mamba's bias for asymmetry in the composite function task arises from its lack of explicit positional encoding, thus encouraging reliance on convolution. We select a configuration under which Mamba struggles to learn both the composite and symmetric solutions. Specifically, we use a two-layer Mamba model with standard initialization($\gamma = 0.5$). All other settings are kept consistent with those used in the phase diagram experiments.

E.1.4 Experiment under full symmetry

Mamba configurations To validate Mamba's difficulty in solving the composite function task under fully symmetric settings, we set dimension of model to 128, while keeping all other settings consistent with those used in the phase diagram experiments. The number of layers is varied across 2, 3, 4, 5, and 6, and standard initialization($\gamma = 0.5$) is applied.

Extended results For each configuration, experiments are conducted using three fixed random seeds. The results are shown in the Fig. 1 and Fig. 2. As can be observed, under all configurations, Mamba consistently struggles to solve the composite function task in the fully symmetric setting.

E.2 Inverse sequence matching task

Mamba configurations The dimension of model is set to 128, the dimension of hidden state is set to the default value of 128, and the expansion factor is set to the default value of 2. The activation function used is SiLU. To clearly isolate the structural differences between Transformer and Mamba and ensure a fair comparison, the Transformer model also adopts SiLU as its activation function. The convolution kernel length is set to its default value of 4, and the number of heads is fixed at 1.

Transformer configurations The dimension of model is set to 128, consistent with the Mamba configuration. To ensure a fair comparison, the Transformer also adopts a $2\times$ dimensional expansion when generating value vectors, mirroring Mamba's setup. Specifically, the dimensions of the query and key vectors are set to 128, while the value vectors have a dimension of 256. Additionally, to ensure a fair comparison and maintain a comparable number of parameters between the two models,

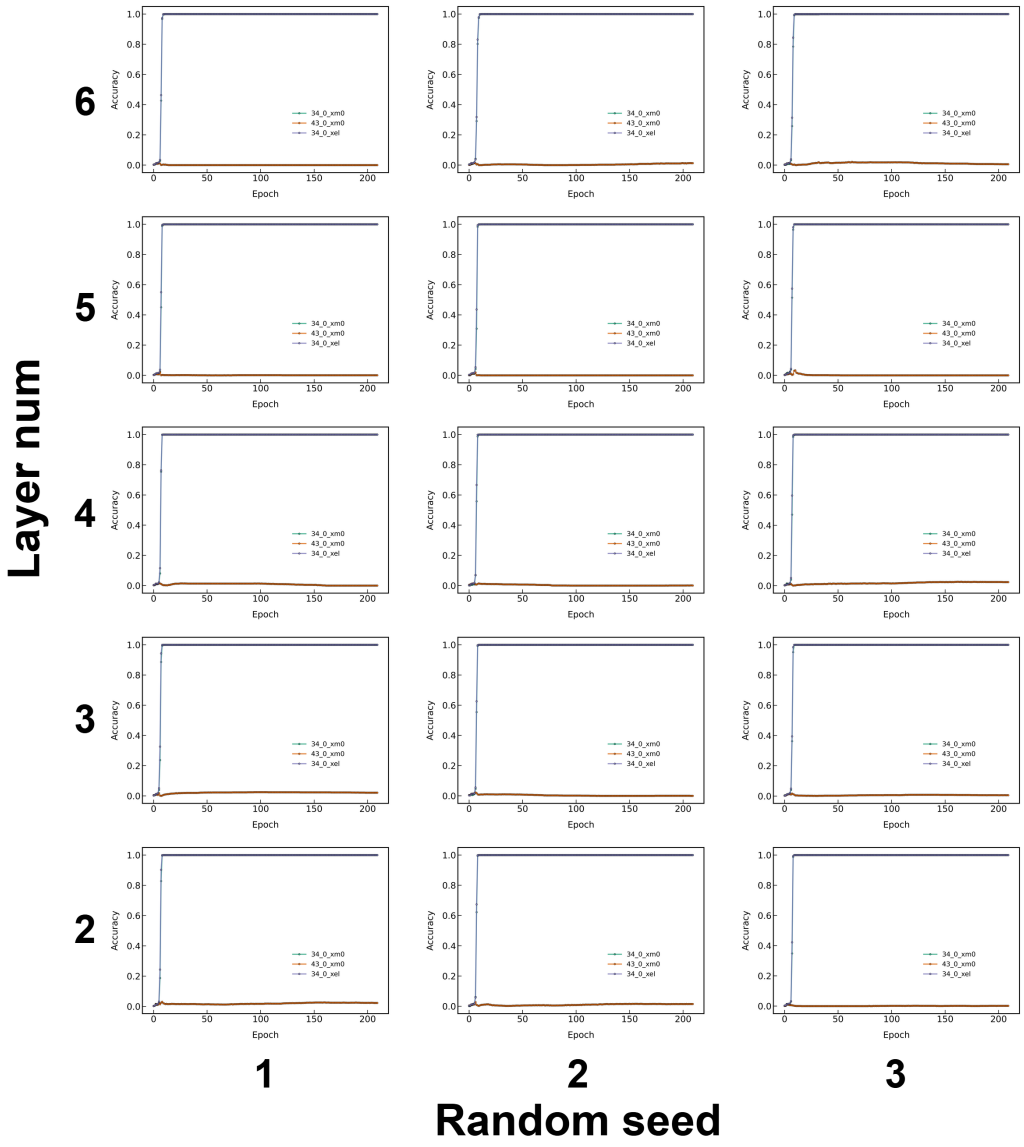

Figure 1: Accuracy of Mamba under fully symmetric setting. The horizontal axis represents the random seed, while the vertical axis corresponds to the number of layers in the Mamba model.

the Transformer's feedforward network (FNN) uses a hidden dimension of 128, and, like Mamba, it is configured with only a single attention head throughout.

Extended results All experiments are conducted using the same set of random seeds to ensure fairness in comparison.

To evaluate Mamba's difficulty in solving the inverse sequence matching task, we vary the model depth across 2, 3, 4, and 5 layers, and consider different initialization strategies ranging from standard to small initialization($\gamma = 0.5, 0.6, 0.7, 0.8, 0.9, 1.0$). The detailed results are shown in the Fig. 3 and Fig. 4. It can be observed that Mamba fails to solve the inverse sequence matching task under nearly all configurations.

For the two-layer Transformer, results under different initialization schemes($\gamma = 0.5, 0.6, 0.7, 0.8, 0.9, 1.0$) are shown in the Fig. 5 and Fig. 6. It can be seen that even with a small number of layers, the Transformer outperforms Mamba.

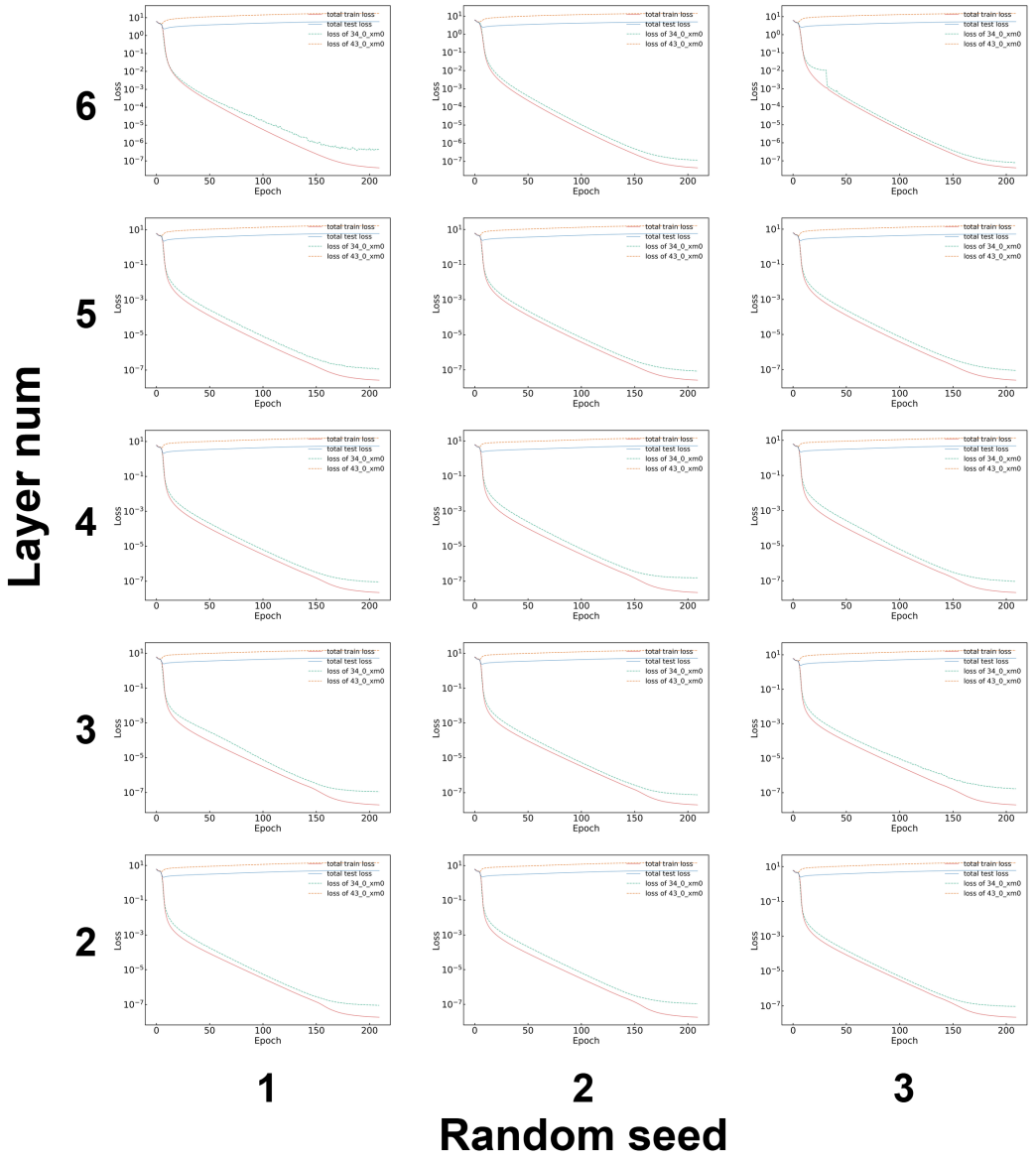

Figure 2: Loss of Mamba under fully symmetric setting. The horizontal axis represents the random seed, while the vertical axis corresponds to the number of layers in the Mamba model.

The results of the modified two-layer Mamba under different initialization schemes($\gamma = 0.5, 0.6, 0.7, 0.8, 0.9, 1.0$) are shown in the Fig. 5 and Fig. 6. It can be observed that the modified Mamba not only significantly outperforms standard Mamba, but also substantially surpasses the performance of the Transformer.

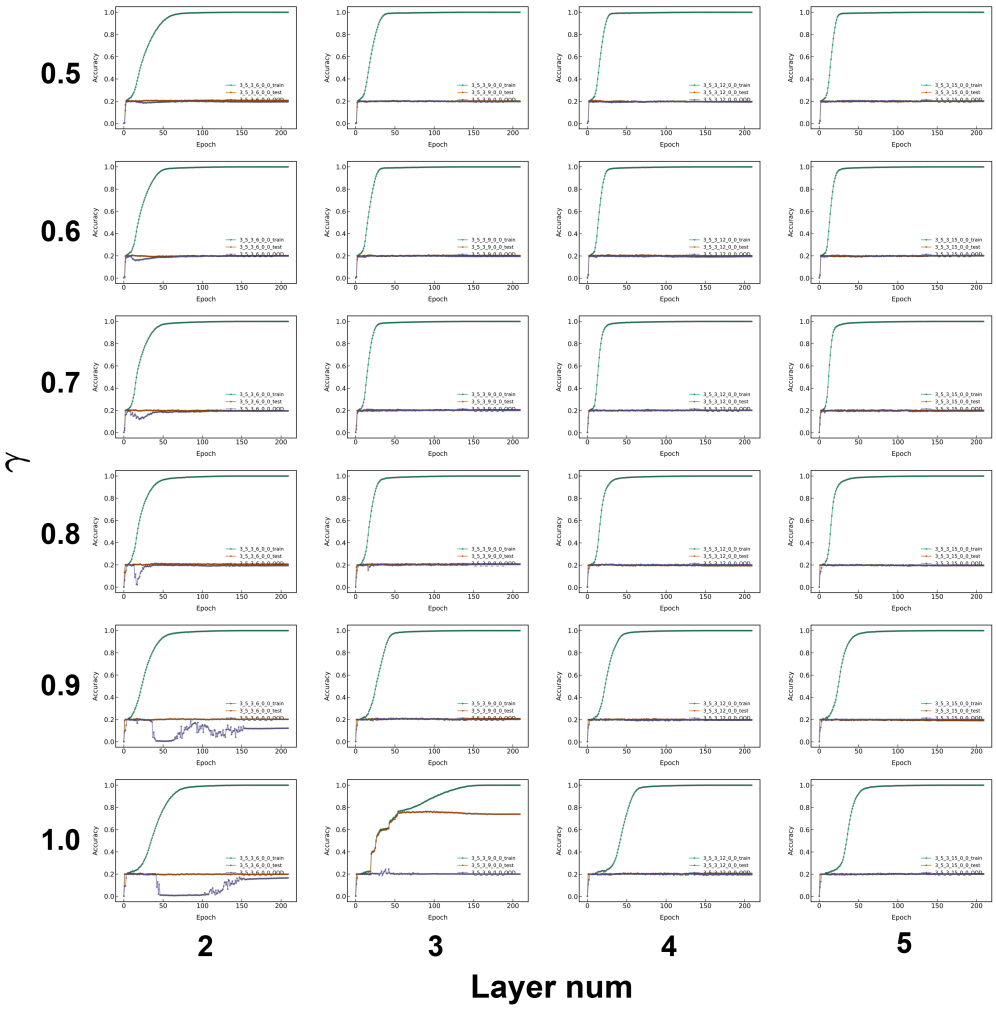

Figure 3: Accuracy of Mamba under different configurations on the inverse sequence matching task. The horizontal axis represents the number of layers in the Mamba model, while the vertical axis corresponds to the initialization scheme.

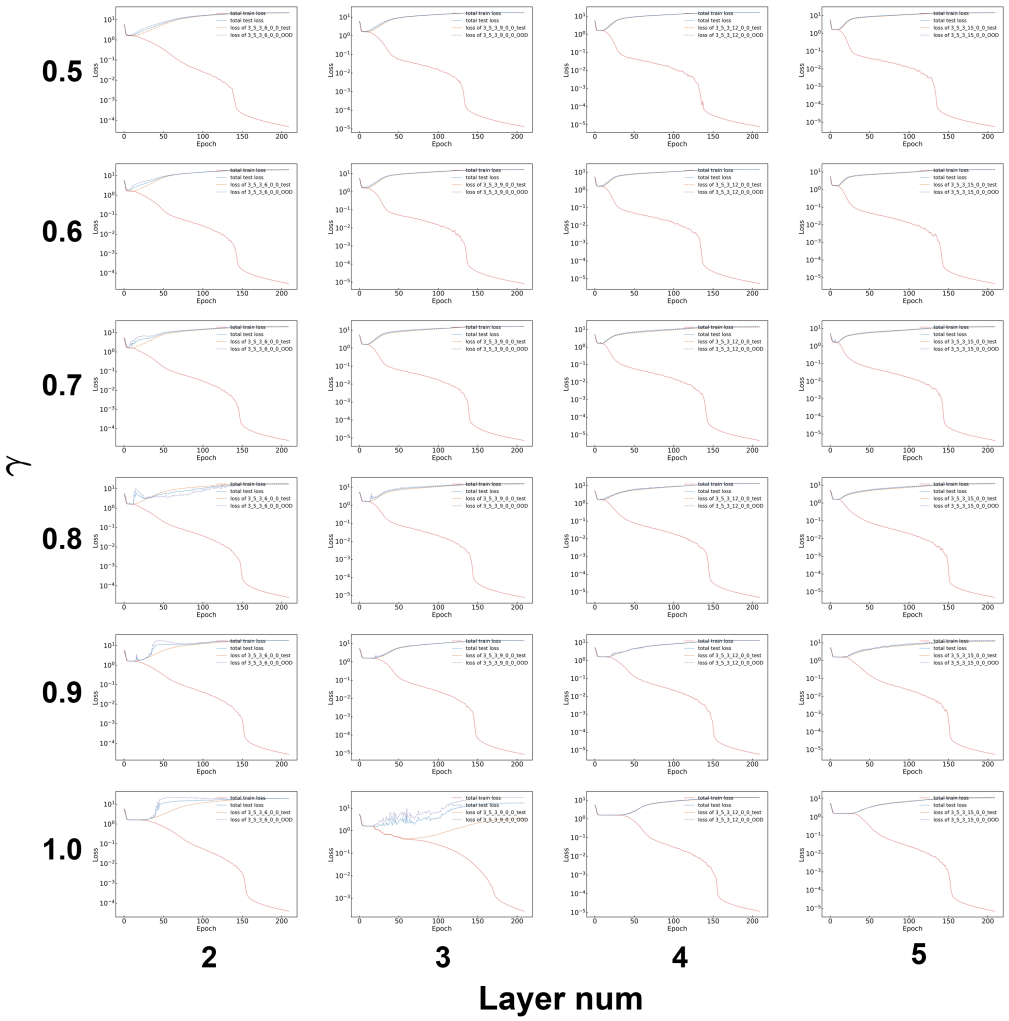

Figure 4: Loss of Mamba under different configurations on the inverse sequence matching task. The horizontal axis represents the number of layers in the Mamba model, while the vertical axis corresponds to the initialization scheme.

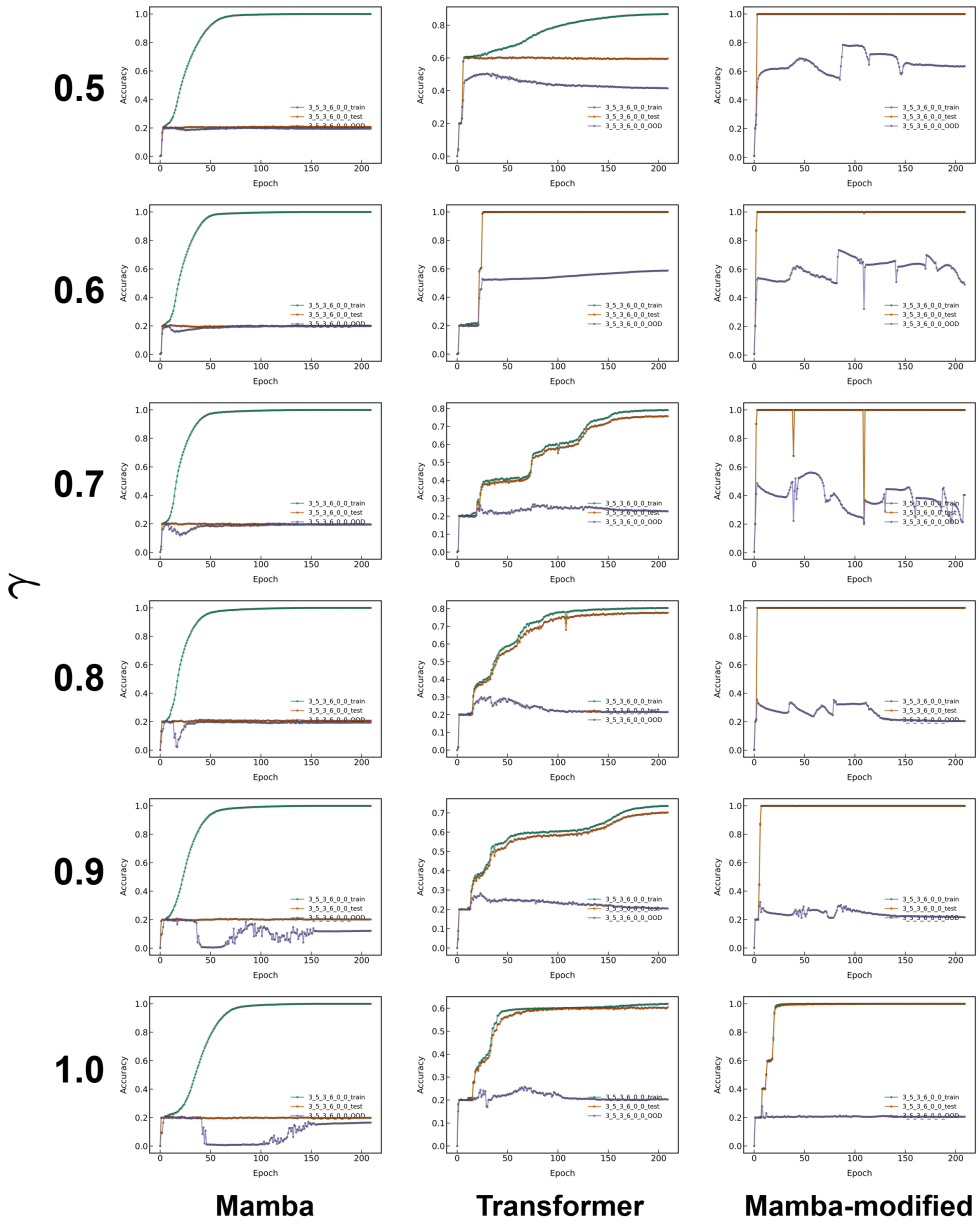

Figure 5: Accuracy of Mamba, Transformer, and modified Mamba on the inverse sequence matching task. Left: Mamba (2-layer), Center: Transformer (2-layer), Right: Modified Mamba (2-layer); all under varying initialization schemes.

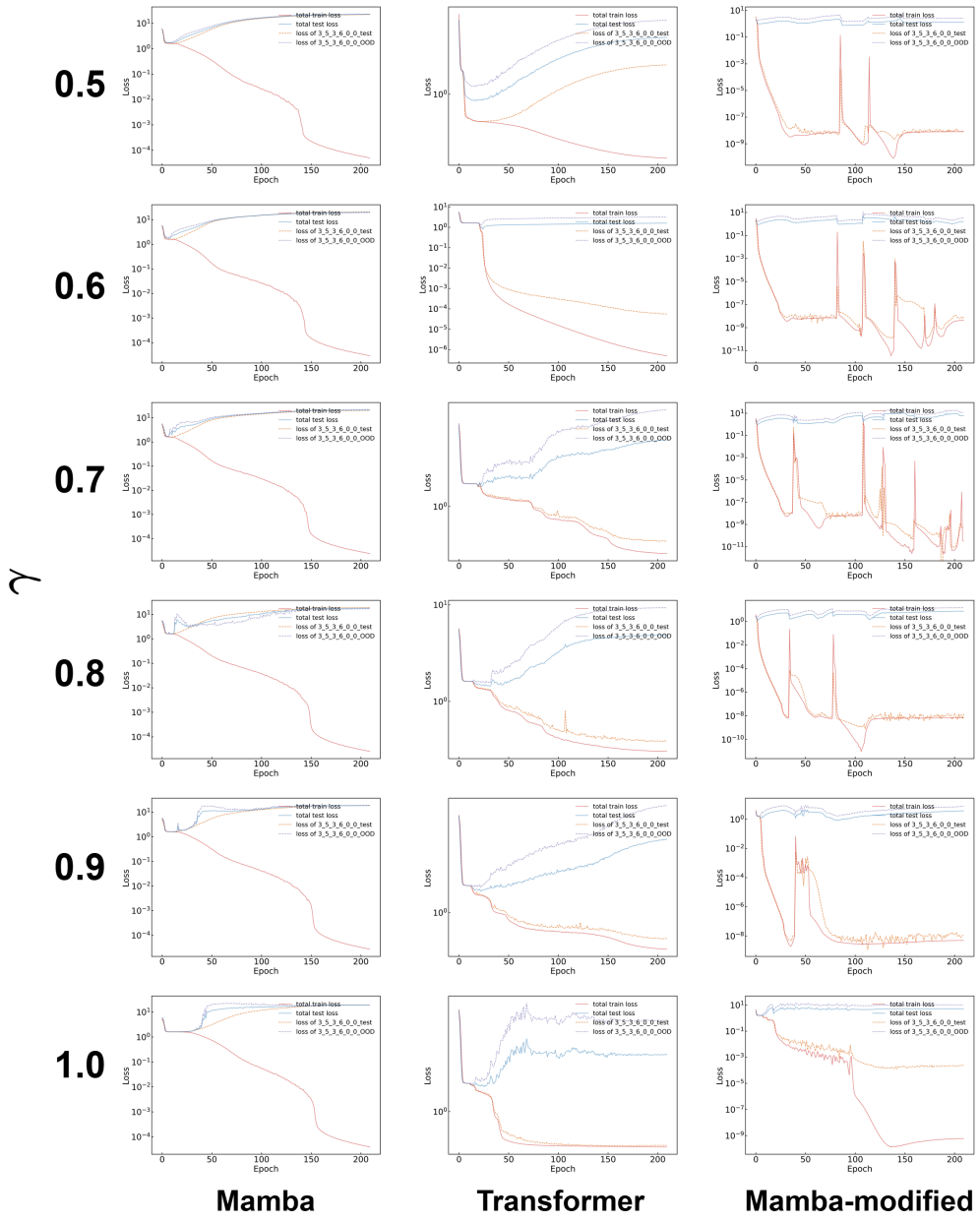

Figure 6: Loss of Mamba, Transformer, and modified Mamba on the inverse sequence matching task. Left: Mamba (2-layer), Center: Transformer (2-layer), Right: Modified Mamba (2-layer); all under varying initialization schemes.