CV IS Study

Donald Winkelman, Maxwell Tang, Jaime Davila

THE FUTURE!

1 Abstract

Recently, Mamba has emerged as a powerful model for sequence prediction. In particular, Mamba has been shown to have a strong inductive bias for the induction heads task. In this paper, we experiment with multiple model architectures, including [TODO], and find that

2 options:

...[Model Architecture] performs better than established models on text prediction in on the [Insert Dataset].

...Mamba fails to use context information in order to improve accuracy on OCR tasks.

2 Background on State Space Models

In this paper, we use a network architecture called Mamba, which is centered around a type of sequence-to-sequence model called a state space model(SSM).

2.1 State Space Models

State space models(abbreviated SSMs) are a type of sequence-to-sequence model that have recently found promising applications in language modeling[**mamba**], image processing[**medmamba**], and more[**s4**].

The recent wave of SSMs has been discrete SSMs, which, as the name suggests, are discrete versions of continuous SSMs. Continuous SSMs introduced by Kalman[**kalman**] for signal processing, and they assume that the system that they are modeling has some internal state s that is affected by an input signal x, and produces an output signal y based on x and s. The governing equations are as follows:

$$\frac{ds}{dt} = \mathbf{A}\vec{s} + \mathbf{B}x$$
$$y = \mathbf{C}\vec{s} + \mathbf{D}x$$

where x, y, s are vectors, and $\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}$ are matrices. These models can simulate a variety of systems, such as physics equations, financial equations, and more.

A key observation is that all of the relationships are linear. This is one of the key properties used by discrete SSMs.

2.1.1 Discretization

Since we can't store general continuous sequences in hardware, real implementations of SSMs typically approximate the dynamics across discrete timesteps. A common method for discretizing SSMs is used by Gu et al. [**s4**]

$$s_k = \overline{\mathbf{A}} s_{k-1} + \overline{\mathbf{B}} x_k \quad \overline{\mathbf{A}} = \left(I - \frac{\Delta}{2}\mathbf{A}\right)^{-1}\left(I + \frac{\Delta}{2}\mathbf{A}\right)$$

$$y_k = \overline{\mathbf{C}} s_k \qquad \overline{\mathbf{B}} = \left(I - \frac{\Delta}{2}\mathbf{A}\right)^{-1}\Delta\mathbf{B} \qquad \overline{\mathbf{C}} = \mathbf{C}$$

This scheme assumes that the input signal is constant within each timestep and uses the trapezoidal rule for integration. Note that $\mathbf{D}$ is omitted. This is since the "pass-through" function of $\mathbf{D}$ can be easily emulated with this discretization.

2.2 Hyperparameters

In the style of Gu et al. [**s4**], we will use the following notation for dimensions:

- B - The batch size.
- L - The length of the sequence.
- D - The number of individual SSMs to run in parallel. This is the actual number of input channels.
- N - The state size for each SSM.

Note that the input and output sizes are omitted. These are generally both set to 1.

2.3 S4

S4[**s4**] is one implementation of state space models that implements the scheme detailed above. S4 uses an initialization for $\mathbf{A}$ based on the HiPPO framework, which is designed to efficiently represent long-range dependencies. The key development introduced by S4 is that the S4 parametrizes $\mathbf{A}$ as a normal-plus-low-rank(NPLR) matrix. The authors then use the spectral properties of these matrices in an algorithm that brings the training complexity from $O(BLN^2)$ down to $O(BN(N\log N + L\log L) + B(L\log L)N)$[**s4**]. This allows the authors to achieve much higher state sizes than normal. As for parameterization, S4 makes Λ, $\mathbf{D}$(Components of $\mathbf{A} = \Lambda - \mathbf{D}\mathbf{D}^*$), $\mathbf{B}$ and $\mathbf{C}$ trainable parameters. Δ is indirectly trainable by scaling $\mathbf{B}$ and $\mathbf{A}$.

2.4 Mamba

Mamba[**mamba**] is the model architecture that we use in our experiments. Mamba introduces a new SSM architecture, called S6, and it provides a new block design for introducing gating.

The S6 architecture is an improvement on the S4 architecture that introduces timesteps that vary within sequences. This breaks the algorithm introduced in S4, so the authors fall back on more general, existing optimizations in order to maintain performance with large state sizes. Variable timesteps allow the model to selectively "forget" and "ignore" different values. If the timestep for a given input is set to a high value, the existing state will decay to close to 0, allowing the model to "forget" all previous information. If the timestep for a given input is set to a low value, the state has very little time to be influences, and in the case of the timestep being 0 or almost 0, it is as if the model "skips over" the given input token.

The block model introduced by the authors is an improvement on the existing H3[**h3**] architecture, and adds explicit selection functionality.

This allows the model to perform well on copying tasks, which we hypothesize is crucial for in-context character recognition.

3 Exploratory testing with Mamba

The inductive biases and thoeretical capabilities of Mamba are an active area of research. In this section we run a multitude of mini-experiments on the Mamba architecture in order to understand it's capabilities better. This section(and the background on SSMs and Mamba) constitute Maxwell's independent portion. In this section, we attempt to recreate some

3.1 Known Results For State-Space Models

It is known that state space models are only capable of recognizing a subset of the regular languages[**ssmformal**], called the star-free languages. The authors define the star-free languages are as the languages that can be defined with empty set, the empty string, individual symbols, concatenation, and Boolean combinations. One important subset of the star free languages is a language class that we will denote SEARCH.

Definition 1. *SEARCH is the class of all $SEARCH_s$, where $SEARCH_s$ is defined as the set of all strings containing s as a substring.*

For instance, $\text{SEARCH}_{\text{xyz}}$ is defined as $\Sigma^* xyz \Sigma^*$. Mamba has been found to perform as well as transformers on finite n-gram memorization tasks, but fails to generalize on multiple n-gram memorization [**mambangram**].

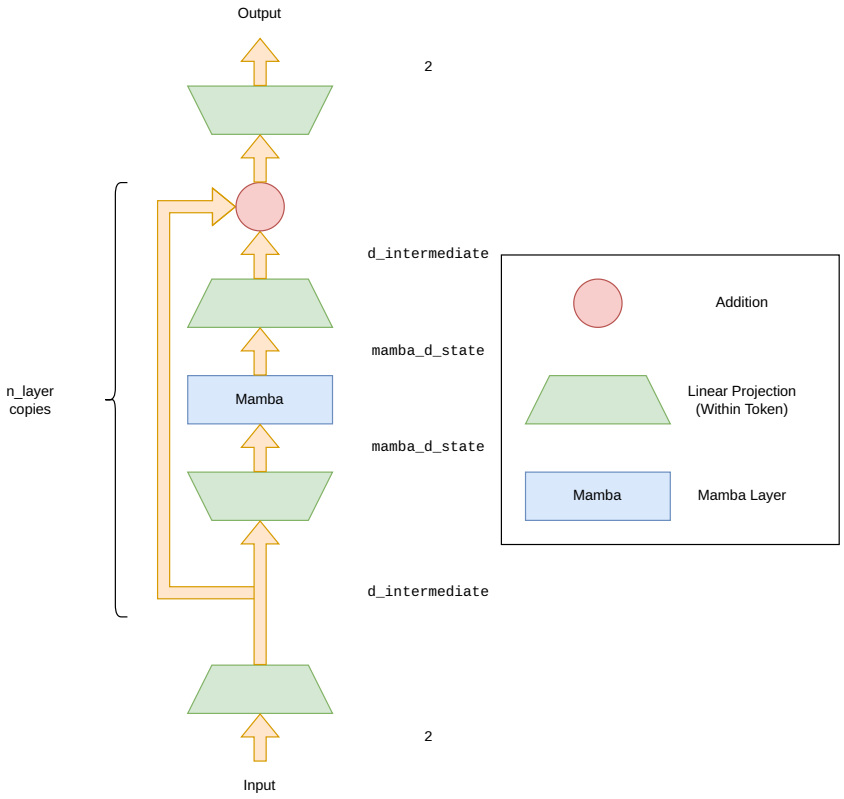

Figure 1: Simple Mamba Stack Architecture

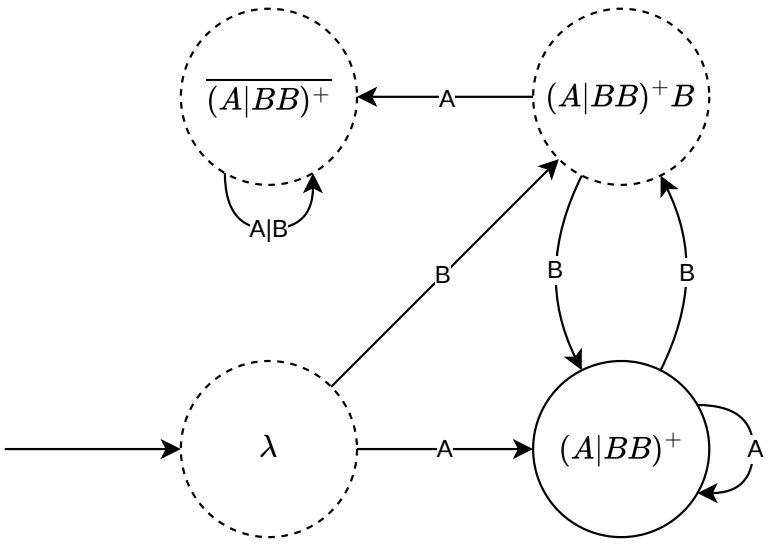

Figure 2: DFA for `(a|bb)+`

3.2 Star-Free Approximation with Mamba

Despite their inability to perfectly predict non-star-free regular languages, we find that Mamba is capable of recognizing certain non-star-free regular languages with limited accuracy.

Preliminary testing on the regular language `(a|bb)+` shows a surprisingly strong ability to recognize instances of the language, with lower accuracy on medium-length strings. If we construct the DFA for the language(see Figure 2), it is clear that recognizing the language requires recognizing odd-length runs of `b` We hypothesize that Mamba is searching for specific odd-length runs of `b`.

Dataset The dataset that we train Mamba to recognize is a synthetic dataset. The instances are first randomly and uniformly assigned a length(from 1-64 inclusive) and a label(either in the language or not). They are then randomly sampled given these constraints. For instance, the data generator might first choose a length of 37, and a label of true(in the language). Then, it uniformly samples from the set of all length 37 strings in `(a|bb)+` The sampling is done with an efficient dynamic programming algorithm.

Model The model that we train is a stack of Mamba layers with linear projection in-between(see Figure 1). The inputs and outputs also have linear projections to make the channel numbers match the task. The primary hyperparameters are `n_layer = 2`, `mamba_d_model = 16`, and `mamba_d_state = 16`.

The rest of the hyperparameters can be found on Github[1].

Results We find that the model quickly attains nearly perfect performance on positive cases, with the trained model only making errors with negative cases. This strongly suggests that our hypothesis is true. The model is correctly identifying that positive cases lack the forbidden strings that the model memorized, and the model is failing to recognize some negative cases, since they contain odd-length forbidden strings that were not memorized.

3.3 Formation of "Bad Habits" with Mamba

Given Mamba's limited ability to recognize regular languages, it would be useful to place it alongside more formally powerful models such as LSTMs, which can trivially model general regular languages given large enough parameters[**lstmformal**]. Unfortunately, preliminary testing found that the inclusion of Mamba in series with LSTMs leads to a degradation of generalization ability.

We suspect that this explains the inclusion of drop-path in many ML architectures. Drop-path is a normalization layer introduced by FractalNet[**fractalnet**] in which the outputs of entire layers in an architecture are "zeroed-out". As stated by the authors, this trains the model to do the task without certain components. In the literature, we find that machine learning architectures like MedMamba[**medmamba**], VMamba[**vmamba**], and MambaND[**mamband**] apply drop-path to the output of Mamba layers. We suspect this is done since Mamba learns certain tasks very quickly, leading to the derivatives produced by Mamba to "drown out" the derivatives coming from the other layers. Drop-path would allow the other layers to then learn to solve the tasks without being influenced by the Mamba layer.

3.4 Weak Generalization on Counter CFGs

Another interesting finding from the literature is that

4 Methodology

In this section, we explain our setup for determining Mamba's ability to learn from context. In our paper, we test 2 different models which attempt to use Mamba for its in-context learning abilities against a variety of datasets, elaborated on below.

4.1 Data Format

In plain English, the task we train is

> Each instance of our task is based on a **context image**, which is a photograph of a scene containing one or more words(e.x. photograph

[1] `https://github.com/maxwell3025/CV-IS-Fall-2024/blob/main/mamba_formal/config/experiment_a_or_bb.yaml`

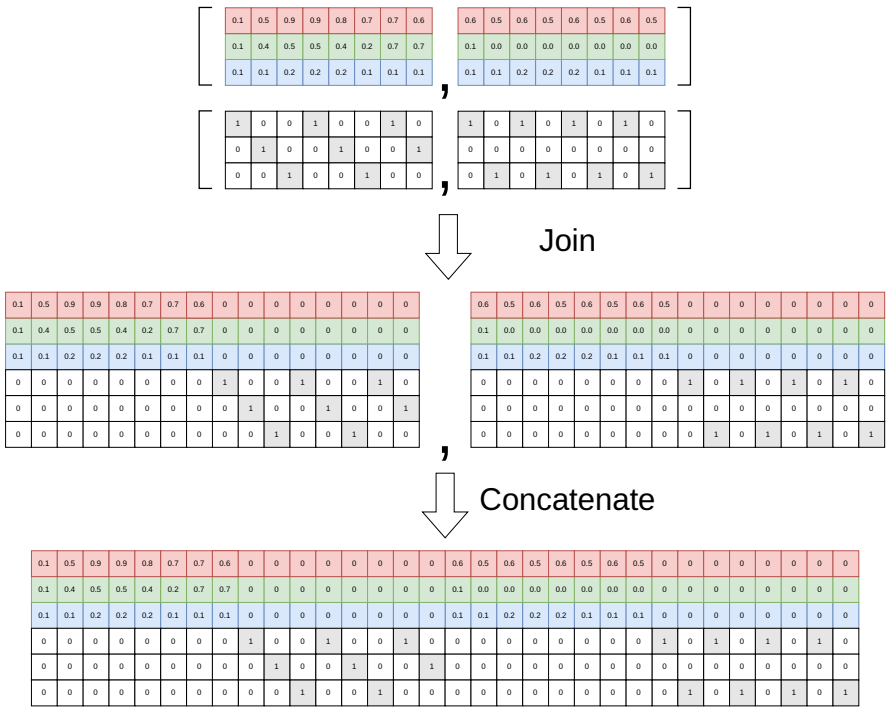

Figure 3: In-Context Encoding

of a train station). We are also provided a set of AABBs surrounding each English word/token in the image(e.x. `the`, `,`). Our task is to predict the text associated with each word image.

Since are trying to make Mamba, a sequence-to-sequence model, learn this task, we propose an encodings of contexts as images. As it stands, there already exists a method for transforming raster images into sequences for Mamba, introduced in [**vmamba**]. We are trying to demonstrate improvement based on context, however, so we need to add context to the sequences. We do this using 2 methods: Positional Encoding and Text Injection

4.1.1 Position Encoding

An issue with our sequence transformation is that we destroy all 2-D positional data. Thus, before the sequence transformation, we append new channels to the image, similar to the positional encodings mentioned in [**attention**].

4.1.2 Text Injection

In order to supply context to the models, we encode our dataset using a scheme that involves what we will call in-context encoding. Before in-context encoding, all of the images within a context need to be transformed into sequences. We will call these sequences the image matrices, where each column corresponds to a pixel in the image(with positional encodings and other per-pixel metadata) Then, we encode the labels as one-hot tokens such that each label is represented as a matrix of one-hot columns. We now apply the in-context encoding transformation on these 2 lists of tensors. First, we diagonally append the label tensors to the image sequence tensors such that they are in separate channel spaces. Then, we append all of the combined tensors lengthwise to get the final tensor. This process is shown in Figure 3.

With this encoding scheme, we assume that the data is being processed by an auto-regressive model, where each output token is evaluated based on the previous (correct) output tokens, and previous predictions in context.

We hypothesize that Mamba's ability to do in-context learning[**mamba**] will allow model architectures containing Mamba to perform well on in-context image prediction.

4.2 Mamba Stack

The first model architecture that we test is a stack of Mamba layers. In our previous testing, we found these to perform well on finite memorization tasks, so we expect this model to do relatively well.

4.3 Mamba With CNN

Another model architecture we look at they run CNN layers parallel to Mamba layers.

4.4

5 Results

6 Conclusion