Project Outline

https://github.com/anyfruit/mamba-sentiment-dataset.git

0.1 Title:

Mamba–the First Linear-time Sequence Model that Achieves Transformer-quality Performance[1]

0.2 Who:

Ruoyun Yang: ruoyun_yang@brown.edu cs_login: kyan28
Kejing Yan: kejing_yan@brown.edu cs_login: ryang104
Xiaoke Song: xiaoke_song@brown.edu cs_login: xsong40

0.3 Introduction:

In this project, we aim to re-implement the Mamba: Linear-Time Sequence Modeling with Selective State Spaces model with TensorFlow and evaluate its performance on a natural language sentiment classification task using the *Sentiment140 dataset*[2] on Kaggle. The dataset consists of 1.6 million tweets labeled as positive, negative, or neutral, and the task involves predicting sentiment based on tweet content. Sentiment analysis is a classic problem in natural language processing that requires models to understand context, sarcasm, and shifts in meaning, which makes it an ideal challenge for testing sequence models. Our focus is on evaluating Mamba's ability to model textual data both efficiently and effectively. Given its advancements, we aim to compare Mamba directly to Transformer models to see whether it can match their performance while being more computationally efficient as stated in the paper.

We selected Mamba for its innovative approach to sequential modeling, which combines the basic structure of state space models (specifically structured state space sequence models known as S4) with input-dependent selection mechanism and hardware-aware efficiency. Unlike Transformers, which suffer from quadratic scaling with respect to the window length due to self-attention, Mamba processes sequences in linear time because of the hardware-aware algorithm that computes the model recurrently with a scan instead of convolution, but does not materialize the expanded state in order to avoid IO access between different levels of the GPU memory hierarchy(p.2). Better than earlier SSMs, which often underperformed on tasks with discrete and information-dense data such as text, Mamba introduces the ability to efficiently select data in an input-dependent manner that improves its performance on such data. These advancements endow Mamba a relatively good balance of the efficiency vs. effectiveness tradeoff of sequence models and therefore a promising alternative for tasks involving long or information-dense sequences. Our objective is to explore how well Mamba performs on sentiment analysis, how it compares to Transformer models in terms of accuracy and speed, and what trade-offs emerge

[1] Gu, A., Dao, T., Ermon, S., Ré, C., & Rudra, A. (2023). *Mamba: Linear-Time Sequence Modeling with Selective State Spaces*. arXiv:2312.00752. https://arxiv.org/abs/2312.00752

[2] Sentiment140 dataset. Available at: https://www.kaggle.com/datasets/kazanova/sentiment140.

from its innovative architecture. We believe our results may provide insights for practical deployment of more efficient NLP models.

0.4 Related Work:

(Albert Gu, Karan Goel, and Christopher Ré. "Efficiently Modeling Long Sequences with Structured State Spaces". In: The International Conference on Learning Representations (ICLR). 2022)[3]

We took a close look at the Structured State Space (S4) model by Gu et al. (2022) because it forms the foundation upon which Mamba is built. The S4 model introduced a breakthrough in sequence modeling by enabling state space models (SSMs) to efficiently handle long-range dependencies (LRD). Traditional SSMs suffered from computational inefficiencies and instability, especially when dealing with long sequences. S4 resolves this by re-parameterizing the core SSM matrices with a combination of both normal and low-rank terms, which would allow the model to be diagonalized stably and reduce the SSM to the computation of a Cauchy kernel. This innovative method reduces the complexity of training and inference to near-linear time. S4 achieves strong empirical results and outperforms prior models on long-sequence benchmarks such as the Long Range Arena and speech classification. It's proved to have both speed and accuracy advancements. This model laid the foundation for Mamba by showing that SSMs could be competitive with Transformers across a range of tasks, but also drew our attention to the potential limitations in adapting to discrete, text-heavy tasks, which is what Mamba addresses

List of public implementations:

- https://github.com/state-spaces/mamba
- https://github.com/alxndrTL/mamba.py
- https://github.com/VuBacktracking/mamba-text-classification
- https://github.com/myscience/mamba

0.5 Data:

We will train and evaluate our Mamba re-implementation on *Sentiment140 dataset*[4], a public corpus of 1600000 tweets extracted using twitter API. The tweets have been annotated (0 as negative, 2 as neutral, and 4 as positive). The data were collected via Twitter's Search API in 2009 and released by Go, Bhayani & Huang[5] (ICWSM '09).
Pre-processing pipeline:

1. Stratified split: 1.4M train / 100k val / 100k test (50%-50% class balance preserved). Text cleaning:

[3]Gu, A., Goel, K., & Ré, C. (2022). *Efficiently Modeling Long Sequences with Structured State Spaces*. In: Proceedings of the International Conference on Learning Representations (ICLR).

[4]Sentiment140 dataset. Available at: https://www.kaggle.com/datasets/kazanova/sentiment140.

[5]Go, A., Bhayani, R., & Huang, L. (2009). *Twitter Sentiment Classification using Distant Supervision*. Technical report, Stanford University. Available at: http://help.sentiment140.com/for-students.

- replace URLs & user mentions with special tokens <url>, <user>;
- keep emojis (they carry sentiment);
- lowercase, NFC normalize.

2. Tokenization – 32k-size byte-pair encoding (SentencePiece); maximum sequence length= 64 tokens (approximately 3× the median tweet).

3. Label smoothing $-\epsilon = 0.05$ to soften the hard 0/4 labels.

0.6 Methodology:

Model Architecture

1. First, a linear layer expands the input token embedding from 64 to 128 dimensions. This transformation enhances the network's representational capacity and enables the separation of features that might be inseparable in the original lower-dimensional space, following a core principle that higher dimensionality often allows for better feature discrimination.

2. Then a 1-D convolution layer processes the expanded representation, facilitating information exchange across feature dimensions while preserving the sequence's temporal structure. Then a SiLU activation controls information flow before the selective SSM processes the convolution output in a linear-time recurrent fashion, similar to an RNN but with state space dynamics.

3. Then Mamba performs gated multiplication, where the input passes through a parallel linear layer and SiLU activation before being multiplied with the selective SSM output. This multiplication effectively measures similarity between the current token embedding and the SSM output (containing previous context), creating a content-aware mechanism for information flow control.

4. Then a linear layer reduces the dimensionality back from 128 to 64, completing the information processing cycle.

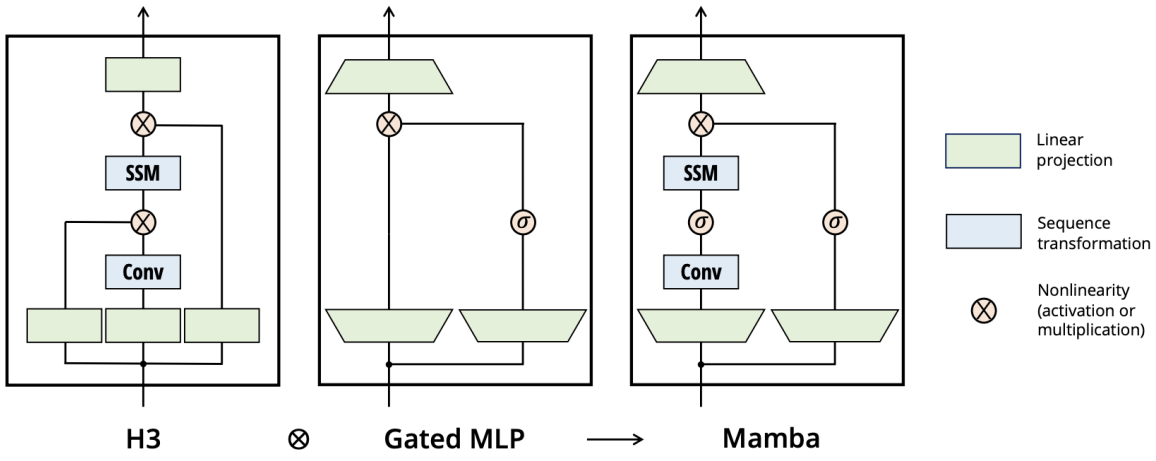

Figure 1: Architecture

This single Mamba layer serves as the fundamental building block of the architecture, with the complete model formed by stacking multiple such layers sequentially to capture increasingly complex representations. Their design combines elements from H3 (the foundation of most SSM architectures) and standard MLP blocks, creating a homogeneous structure of repeated Mamba blocks rather than interleaving different block types. The key modification from H3 is replacing the first multiplicative gate with an activation function, streamlining the architecture.

Training Our Model

We will train our Mamba model on the Sentiment140 dataset containing 1.6 million tweets with three sentiment labels (negative, neutral, positive). Following preprocessing—which includes stratified splitting (1.4M/100k/100k for train/val/test), cleaning text while preserving sentiment-carrying emojis, applying 32k-size byte-pair encoding with SentencePiece, and truncating to 64 tokens per sequence—we'll employ cross-entropy loss with label smoothing (ϵ) to prevent overconfidence, as we mentioned in **Data** section.

Potential Hardest Part while Implementing

The most challenging aspect will be translating Mamba's selective scan operation from PyTorch to TensorFlow while maintaining its computational efficiency. This involves correctly implementing the hardware-aware parallel scan algorithm, ensuring proper handling of input-dependent parameter selection, managing recurrent state computations, and addressing framework-specific differences in gradient calculation. Debugging numerical stability issues that might arise during the translation process will require extensive testing and validation against the original implementation.

0.7 Metrics:

In the original paper, the authors evaluated the model across a range of modalities including language, audio, and genomics, and used modality-specific quantitative metrics to assess the performance of the models. For language modeling, they focused primarily on perplexity (PPL) for pretraining and accuracy (%) on downstream zero-shot benchmarks such as LAMBADA, HellaSwag, PIQA, and WinoGrande. Mamba achieved amazing results on these tasks while being much more efficient in terms of inference speed (tokens/second) and memory usage, than Transformers of similar or even larger sizes.

In our project, we aim to simulate this evaluation philosophy in the context of sentiment classification, a supervised NLP task. Our primary metrics will be *accuracy* for measuring classification performance, and we will also compare training time and inference latency against a Transformer model as baseline. We are particularly interested in understanding whether Mamba's claimed efficiency and modeling power would bring advantages in this setting.

We plan to run the following experiments:

- Train a Mamba-based classifier on Sentiment140 and evaluate performance on a held-out test set.
- Train a baseline Transformer model under the same conditions.

- Compare classification performance (accuracy) and computational efficiency (training time, inference speed).
- (Potential experiment) Conduct an ablation study by modifying Mamba's state size and disabling input-dependent parameters to observe effects on performance.

Base Goal: Successfully implement Mamba for sentiment classification and achieve $\geq 90\%$ accuracy on the test set.

target Goal: Achieve competitive or improved performance compared to a baseline Transformer, and demonstrate lower inference/training cost and higher efficiency.

Stretch Goal: Run an ablation study to compare Mamba's performance with and without input-dependent parameters, in order to better understand how important this feature is for the model's effectiveness.

0.8 Ethics:

1. **Dataset Concerns: Collection, Labeling, and Bias:**
   Sentiment140 relies on automatic emoticon-based labeling, meaning tweets with ":)" (or variants) are tagged as positive, and those with ":(" are tagged as negative. This method introduces potential label noise because:

   - Irony & sarcasm: People often use emoticons in sarcastic or ambivalent contexts, so the auto-label may be incorrect.
   - Evolving language: Social media slang and cultural references differ across demographics, regions, and time. By focusing on English tweets from 2009, the dataset might underrepresented global opinions or non-English sentiments.
   - Demographic skew: Twitter users at that time may overrepresent younger or tech-savvy groups, introducing biases if the model is deployed on broader populations.

   Furthermore, Twitter text often includes slang, abbreviations, or potentially offensive content. These artifacts can shape the model's learned representation, potentially leading to biased or inappropriate responses when analyzing modern tweets or out-of-domain data.

   Possible Mitigation Steps:

   - Perform a manual audit on a random sample to approximate label correctness.
   - Apply basic offensive-language filtering during training if especially toxic content appears.
   - Document the dataset's time-period limitation and the likely drift in modern sentiment usage.

2. **Stakeholders & Consequences of Misclassification:**

Stakeholder	How They Use Sentiment Analysis	Consequences of Errors
Businesses & Marketing Teams	Track brand reputation, measure consumer satisfaction, guide marketing strategies	Overly positive or negative misreadings can lead to misguided product changes or PR responses
Researchers & Social Scientists	Study public opinion on social/political issues, track language changes	Bias in data or model predictions may skew research findings, misrepresenting public sentiment
End-Users / Consumers	Real-time monitoring of news or product feedback (e.g., integrated into social apps)	Misclassification could amplify toxic content or suppress valid user feedback

While incorrect predictions on individual tweets may seem minor, at scale, systematic errors could distort important decisions—for example, a brand might incorrectly assume a product's success or failure, or policy analysts might misjudge public opinion on critical issues.

Risk Mitigation:

- Confidence Scores & Human Oversight: Provide confidence estimates so high-impact decisions (e.g., adjusting budgets or policy stances) can be deferred to human analysts when the model is uncertain.
- Periodic Re-training: Language on social media evolves rapidly; scheduling model updates or domain adaptation helps maintain reliability.
- Transparency: Publish model cards detailing performance across different demographic slices (e.g., tweets referencing certain communities, or from different time ranges) to identify potential biases.

0.9 Division of labor:

Ruoyun Yang: Contributed to all phases of the project, including coding, experiment setup, and documentation. Took primary responsibility for writing the Introduction and Related Work sections, and assisted in the interpretation of results. Will also help lead experiments such as ablation studies and performance evaluations.

Kejing Yan: Participated in the full project workflow, including implementation, testing, and analysis. Led the Results and Challenges sections of the final report and contributed to experimental evaluations, particularly focusing on the comparative analysis and insights.

Xiaoke Song: Led the technical implementation, including developing the model architecture, writing training and evaluation scripts, and debugging. Authored the Methodology section of the final report and supported the design and execution of experiments.

All three members will collaborate on the final poster and presentation, including visual design and messaging. The team will jointly rehearse and present during Deep Learning Day, with each member prepared to discuss specific aspects of the project. Everyone will contribute to the Reflection section based on shared input and insights gained throughout the project.