Project Reflection

Xiaoke Song, Ruoyun Yang, Kejing Yan
https://github.com/anyfruit/mamba-sentiment-dataset.git

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 on Kaggle, comparing to transformer, SSM and LSTM. The dataset consists of 1.6 million tweets labeled as positive, negative, or neutral, and the task involves predicting sentiment.

Data Preprocessing

We have collected all the data and preprocessed it. The cleaned tweets were converted to integer token sequences using BERT's bert-base-uncased tokenizer with a fixed maximum length of 64 tokens. The datasets are stored in `.csv` format and loaded efficiently via `tf.data.Dataset`. A key insight is that BERT-based tokenization (with padding and truncation) significantly improves downstream model compatibility and generalization compared to naive integer encoding. This preprocessing pipeline now supports all four model types.

Model Implementation

We have successfully completed the model implementation phase. This includes translating the original Mamba model from PyTorch to TensorFlow. To benchmark Mamba's performance and runtime efficiency, we also implemented our own Transformer, LSTM, and a simple SSM model from scratch. Currently, we have successfully trained each model on a small fraction of our chosen dataset using a complete pipeline that reports both accuracy and runtime. The results of these initial evaluations are discussed in the following section.

Challenges

The most challenging part existed when translating the Mamba model from PyTorch to TensorFlow, Mamba relies on selective scan operations and dynamic state updates that are implemented using custom CUDA kernels in PyTorch, which are not directly available in TensorFlow. To address this, we carefully reconstructed each part using TensorFlow-native operations such as `tf.einsum`, `tf.cumsum`, and custom reshaping logic to simulate the behavior of the original Mamba components. We encountered several shape mismatches, especially when handling multi-dimensional tensors in our state-space updates. We resolved these by thoroughly inspecting tensor shapes at each step and debugging was aided by unit tests and synthetic data.

Insights

From our preliminary evaluations on a subset of the tweet sentiment dataset, we observed that SSM and Transformer models outperformed Mamba and LSTM in accuracy, with both achieving comparable top scores. Interestingly, LSTM seemed to have the slowest inference time, while SSM and Transformer were the fastest, and Mamba fell in between. This aligns only partially with claims from the Mamba paper, which states that Mamba should match or exceed Transformer-level performance with significantly better runtime efficiency. We suspect this discrepancy is due to multiple factors: 1) *Data Simplicity*, 2) *Subset Training*.

Plan

We are on track with our project. The implementation for all four models (Mamba, SSM, Transformer, and LSTM) has been completed, and subset testing was successfully conducted with results saved and evaluated (generated runtime and accuracy comparisons across models).All group members contributed evenly to both the coding and the reflection write-ups, and we feel confident about moving forward with full-scale training and final analysis in the next stage. Possible further changes include different hyperparameters to better reflect the paper's settings.

Supplementary Sources

Model	Depth	Main Layers	Hidden Units	Dropout
Mamba	2	HighwayBlock (Selective Scan)	64 internal	0.2
LSTM	2	LSTM	128 units	0.2
Transformer	2	MHA + FFN	64 embed, 128 FF	0.1
SSM	2	SimpleSSMLayer	64	0.1

Table 1: Consistent Models' Structure for Comparison

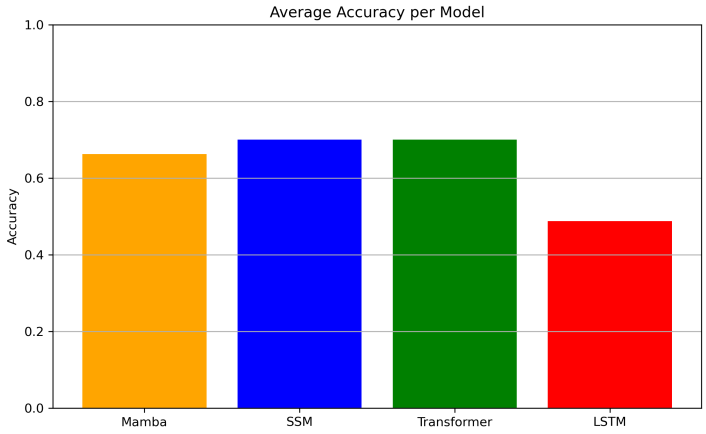

Figure 1: Average Accuracy Summary

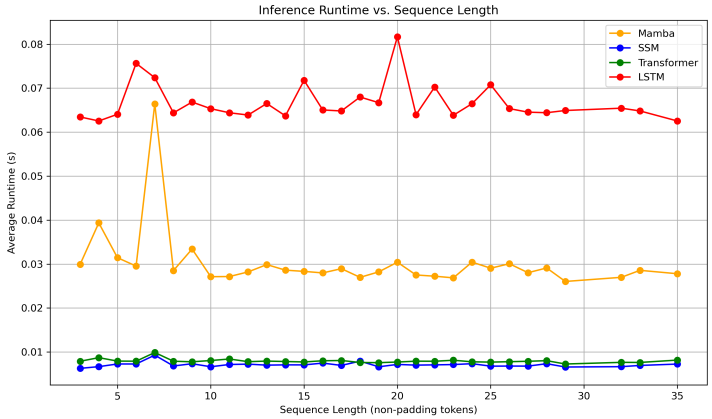

Figure 2: Inference Runtime versus Sequence Length