EXPLORING APPLICATIONS OF STATE SPACE MODELS AND ADVANCED TRAINING TECHNIQUES IN SEQUENTIAL RECOMMENDATIONS: A COMPARATIVE STUDY ON EFFICIENCY AND PERFORMANCE

Baderko Makar*
makarbaderko@gmail.com

Kulibaba Stepan*
kulibabast@gmail.com

Obozov Mark*
obozovmark9@gmail.com

Nikolay Kutuzov
kutuzov.nv@phystech.edu

Alexander Gasnikov
gasnikov@yandex.ru

ABSTRACT

Recommender systems aim to estimate the dynamically changing user preferences and sequential dependencies between historical user behaviour and metadata. Although transformer-based models have proven to be effective in sequential recommendations, their state growth is proportional to the length of the sequence that is being processed, which makes them expensive in terms of memory and inference costs. Our research focused on three promising directions in sequential recommendations: enhancing speed through the use of State Space Models (SSM), as they can achieve SOTA results in the sequential recommendations domain with lower latency, memory, and inference costs, as proposed by Liu et al. (2024); improving the quality of recommendations with Large Language Models (LLMs) via Monolithic Preference Optimization without Reference Model (ORPO); and implementing adaptive batch- and step-size algorithms to reduce costs and accelerate training processes.

1 INTRODUCTION

Modern digital products are heavily dependent on the performance of the recommender systems, that are frequently utilized to predict future interactions of customers based on their historical behaviour.

While recurrent neural networks (RNNs) and convolutional neural networks (CNNs) led the way in the use of neural networks in sequential recommendation, they suffer from the forgetting issue Kirkpatrick et al. (2017). Recently, researches were able to introduce the Transformer-based models into the field Li et al. (2023).

Due to the quadratic computational cost, these attention-based approaches typically suffer from the inference inefficiency problem despite their impressive performance.

SSM models have been adopted as a replacement for RNNs and CNNs, with Liu et al. (2024) being an example. We develop this idea further, while introducing a novel state-of-the-art sequential recommender model Hydra4Rec, based on the architecture proposed in **?**.

*These authors contributed equally to this work.

1.1 RELATED STUDY

1.1.1 SEQUENTIAL RECOMMENDATION TASK DEFINITION

Consider the user set $U = \{u_1, u_2, ..., u_N\}$, item set $V = \{v_1, v_2, ..., v_K\}$ and $S_u = \{v_1, v_2, ..., v_{n_u}\}$ as the chronologically ordered interaction sequence for user $u \in U$, where n_u is the the length of the sequence. Given S_u the task is to predict the next interacted item, v_{n_u+1}.

1.1.2 TRANSFORMERS

Recently, transformer models have been shown to be effective in sequential recommendation tasks as the backbone of larger models Kang & McAuley (2018) and as individual LLMs Li et al. (2023), Yue et al. (2023). Despite their success, attention-based methods face inference inefficiencies due to the quadratic computational complexity inherent in attention operators and their rapid state growth, which is proportional to the sequence length. Also, without special mechanisms Liu et al. (2023), transformers can't handle long contexts and, consequently, long user histories.

1.1.3 STATE SPACE MODELS

State Space Model (SSM) is a recent framework for sequence modelling defined by linear ordinary differential equations:

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

Where A, B, C are learnable matrices, $h(t)$ is the latent space, $x(t)$ is the input sequence and $y(t)$ is the output sequence. To compute sequence-to-sequence transformations efficiently, the matrix A must be *structured*, so structured SSMs have been introduced Gu et al. (2022). A general form of a structured SSM is defined by the equations:

$$\begin{aligned} h_t &= Ah_{t-1} + Bx_t \\ y_t &= Ch_t \end{aligned}$$

Where $A \in \mathbb{R}^{(N,N)}, B \in \mathbb{R}^{(N,1)}, C \in \mathbb{R}^{(N,1)}$. They map a 1-dimensional sequence $x \in \mathbb{R}^T \to y \in \mathbb{R}^T$ through an implicit latent state $h \in \mathbb{R}^{(T,N)}$. To operate directly on sequences, the discretisation rule (f_A, f_B) is applied to continuous parameters $(\Delta, \mathring{A}, \mathring{B})$, by $A = f_A(\Delta, \mathring{A})$ and $B = f_B(\Delta, \mathring{B})$, where Δ is the parameterised step size.

1.1.4 MAMBA BLOCK

In order to adaptively focus on relevant information while filtering out noise, the Mamba block Gu & Dao (2024) introduces an extension to structured SSMs by adding a data-dependent selection mechanism. An important feature of Selective SSMs is their ability to be computed efficiently on the GPU using kernel fusion, parallel scanning and recomputation mechanisms.

In contrast to transformers' $O(n^2 \cdot d)$, SSMs provide $O(n \cdot d^2)$ complexity, which makes them a more efficient alternative, especially when operating on long sequences (Dao & Gu (2024)) .

1.1.5 MAMBA APPLICATIONS TO SEQUENTIAL RECOMMENDATIONS

Several applications of Mambas selective SSMs to recommender systems have been introduced. Liu et al. (2024), Wang et al. (2024). As Mamba has already shown efficient results in different areas Zhu et al. (2024) Erol et al. (2024) Chen et al. (2024), we propose a more complex exploration of Mamba applications to recommender systems.

1.2 HYDRA

Hydra fully utilize the matrix mixer framework, to explore a novel bidirectional sequence mixer. To achieve linear complexity, special class of quasiseparable matrices is considered. A matrix $\mathbf{M}$ is quasiseparable if every label element m_{ij} satisfies:

$$m_{ij} = \begin{cases} \overrightarrow{\mathbf{c}}_i^T \overrightarrow{\mathbf{A}^{\times}_{i:j}} \overrightarrow{\mathbf{b}}_j, & \text{if } i > j \\ \delta_i, & \text{if } i = j \\ \overleftarrow{\mathbf{c}}_j^T \overleftarrow{\mathbf{A}^{\times}_{j:i}} \overleftarrow{\mathbf{b}}_i, & \text{if } i < j \end{cases}, \tag{1}$$

where each δ_i is a scalar, $_i, _i \in \mathbb{R}^{N\times 1}$, and $_i \in \mathbb{R}^{N\times N}$ **?**. Quasiseparable matrices can be used as matrix mixers, this framework is named as **Hydra**, achieving $O(L)$ complexity. Based on an important property of quasiseparable matrices - quasiseparable matrices can be expressed as a combination of two semiseparable matrices - a subquadratic matrix multiplication algorithm was developed. Then, as a semiseparable matrix structure includes SSMs Dao & Gu (2024), any SSM could be combined with the Hydra framework, resulting in a new efficient bidirectional SSM architecture.

1.2.1 UNIVERSAL GRADIENT METHOD FOR STOCHASTIC OPTIMIZATION

The release of the Universal Stochastic Gradient Method (USGM) by Rodomanov et al. (2024) represents a noteworthy development in the field of stochastic optimization, that is the backbone of machine learning.

The classical Stochastic Gradient Descent (SGD) by Robbins (1951) is one of the earliest techniques in the field of stochastic optimization. Its novelty at the time of publication helped it gain popularity. However, because hyperparameter values are only defined prior to the start of the training phase, SGD achieves poor convergence rates in real-world applications.

A number of SGD-based adaptive strategies, including Adam by Kingma & Ba (2017) and Adagrad by Duchi et al. (2011), have been put forth to address the issue. Although they are more efficient than SGD, because they modify the step size according to the aggregate of the previous gradients, when applied to non-convex optimization problems, they might still experience difficulty achieving convergence.

The USGM develops the idea of adaptiveness further, as the algorithm can adapt to the curvature of the loss function, thus providing more accurate gradient estimates even in highly noisy environments. Classic optimization algorithms are typically designed to handle either smooth or non-smooth problems. USGM, however, is an algorithm capable of addressing both classes of problems and adapting to oracle noise with unknown variance. The algorithm can adjust to Hölder-Lipschitz coefficients without requiring precise calculation of those. The only hyperparameters used by the algorithm are the upper bound on the diameter of the search hypersphere, the initial learning rate and batch size, what makes it a more practical choice for real-world applications.

Notation

The same mathematical notation as in the original USGM paper will be used.

The loss function is defined as f and the parameters of our model are defined as x.

To characterize the smoothness of f the Hölder constant is introduced for each $\nu \in [0, 1]$:

$$L_\nu := \sup_{\substack{x,y\in \mathrm{Dom} f, x\neq y, \\ g(x)\in\partial f(x), g(y)\in\partial f(y)}} \frac{\|g(x) - g(y)\|_*}{\|x - y\|^\nu}, \tag{1.1}$$

where $g(x)$ and $g(y)$ are the stochastic approximations of subgradients of our loss function f at points x and y respectively.

The dual norm is defined in the standard way:

$$\|s\|_* := \max_{\|x\|=1} \langle s, x\rangle = \langle s, B^{-1}s\rangle^{1/2}, \quad s \in \mathbb{R}^n. \tag{1.2}$$

Of course, for certain values of the exponent $\nu \in [0, 1]$, it may happen that $L_\nu = +\infty$. However, we assume that there exists (at least one) exponent for which the corresponding Hölder constant is finite.

For any $t \in \mathbb{R}$, by $[t]_+ := \max\{t, 0\}$, its positive part is denoted. For random variables X and ξ, by $\mathbb{E}_\xi[X]$ and $\mathbb{E}[X]$, the expectation of X w.r.t. ξ, and the full expectation of X, respectively are denoted.

Algorithm 1 Universal Stochastic Gradient Method

1: **Initialize:** $x_0 \in \operatorname{dom} f$, $D > 0$, $H_0 := 0$, $g_0 \sim \hat{g}(x_0)$.
2: **for** $k = 0, 1, \ldots$ **do**
3: $\quad x_{k+1} = \arg\min_{x \in \operatorname{dom} f}\{\langle g_k, x\rangle + \frac{H_k}{2}\|x - x_k\|^2\}$.
4: $\quad g_{k+1} \sim \hat{g}(x_{k+1})$.
5: $\quad H_{k+1} := H_k + \frac{\left[\hat{\beta}_{k+1} - \frac{1}{2}H_k r_{k+1}^2\right]_+}{D^2 + \frac{1}{2}r_{k+1}^2}$,
6: $\quad$ where $r_{k+1} = \|x_{k+1} - x_k\|$, $\hat{\beta}_{k+1} = \langle g_{k+1} - g_k, x_{k+1} - x_k\rangle$.
7: **end for**

The authors of the paper also define the upper bound for the stochastic approximation of the symmetrized Bregman distance for points x_k and x_{k+1} ($\hat{\beta}_{k+1}$) as follows:

$$\hat{\beta}_{k+1} = \langle f'(x_{k+1}) - f'(x_k) + \Delta_{k+1}, x_{k+1} - x_k\rangle \leq L_\nu r_{k+1}^{1+\nu} + \sigma_{k+1} r_{k+1}, \tag{1.3}$$

where $f'(x_k) := \mathbb{E}_{\xi_k}[g_k] \in \partial f(x_k)$, $\Delta_{k+1} := \delta_{k+1} - \delta_k$ with $\delta_k := g_k - f'(x_k)$ being the error of the stochastic gradient (such that $\mathbb{E}\|\delta_k\|^2 \leq \sigma^2$), and $\sigma_{k+1} := \|\Delta_{k+1}\|$.

2 Odds Ratio Preference Optimization

Odds Ratio Preference Optimization (ORPO), Hong et al. (2024) - is novel preference optimization framework that consolidates an odds ratio-based penalty.

The odds of generating the output sequence y given an input sequence x are defined by:

$$\log P_\theta(y|x) = \frac{1}{m}\sum_{t=1}^{m} \log P_\theta(y_t|x, y < t) \tag{2}$$

$$\mathbf{odds}_\theta(y|x) = \frac{P_\theta(y|x)}{1 - P_\theta(y|x)} \tag{3}$$

The odds ratio of the chosen response y_w over the rejected response y_l, $\mathbf{OR}_\theta(y_w, y_l)$, indicates how much more likely it is for the model θ to generate y_w than y_l given input x.

$$\mathbf{OR}_\theta(y_w, y_l) = \frac{\mathbf{odds}_\theta(y_w|x)}{\mathbf{odds}_\theta(y_l|x)} \tag{4}$$

2.1 Objective Function of ORPO

The objective function of ORPO consists of two components: 1) supervised fine-tuning (SFT) loss ($\mathcal{L}_{SFT}$); 2) relative ratio loss ($\mathcal{L}_{OR}$).

$$\mathcal{L}_{ORPO} = \mathbb{E}_{(x, y_w, y_l)}\left[\mathcal{L}_{SFT} + \lambda \cdot \mathcal{L}_{OR}\right] \tag{5}$$

$\mathcal{L}_{SFT}$ follows the conventional causal language modelling negative log-likelihood (NLL) loss function to maximise the probability of generating the reference tokens. The $\mathcal{L}_{OR}$ in the equation maximises the odds ratio between the likelihood of generating the unfavourable response y_w and the favourable response y_l.

$$\mathcal{L}_{OR} = -\log \sigma \left(\log \frac{\mathbf{odds}_\theta(y_w|x)}{\mathbf{odds}_\theta(y_l|x)} \right) \tag{6}$$

Together, $\mathcal{L}_{SFT}$ and $\mathcal{L}_{OR}$ weighted with λ tailor the pre-trained language model to adapt to the specific subset of the desired domain and disfavor generations in the rejected response sets.

2.2 GRADIENT OF ORPO

The gradient of $\mathcal{L}_{Ratio}$ further justifies the use of the odds ratio loss. It consists of two terms: one that penalises wrong predictions and one that contrasts between selected and rejected answers, for $d = (x, y_l, y_w) \sim D$.

$$\nabla_\theta \mathcal{L}_{OR} = \delta(d) \cdot h(d) \tag{7}$$

$$\delta(d) = \left[1 + \frac{\mathbf{odds}_\theta P(y_w|x)}{\mathbf{odds}_\theta P(y_l|x)} \right]^{-1} \tag{8}$$

$$h(d) = \frac{\nabla_\theta \log P_\theta(y_w|x)}{1 - P_\theta(y_w|x)} - \frac{\nabla_\theta \log P_\theta(y_l|x)}{1 - P_\theta(y_l|x)} \tag{9}$$

If the probabilities of the favoured responses are relatively higher than the disfavoured responses, $\delta(d)$ in the equation will converge to 0. This indicates that the $\delta(d)$ will play the role of a penalty term, speeding up the parameter updates when the model is more likely to generate the disfavoured responses.

Meanwhile, $h(d)$ implies a weighted contrast of the two gradients from the chosen and rejected responses. Specifically, $1 - P(y|x)$ in the denominators amplifies the gradients when the corresponding side of the likelihood $P(y|x)$ is low. As the likelihood increases, the model adapts more quickly to the distribution of selected responses.

3 OUR RESULTS

3.1 2MAMBA4REC

To compare the Mamba2 model more fairly with its predecessor (Mamba1) we used the model architecture from the Mamba4Rec article by Liu et al. (2024), replacing the Mamba block with the Mamba2 block introduced by Dao & Gu (2024).

3.2 MAMREC

We implemented the standard GPT4Rec architecture from Li et al. (2023) while replacing the GPT-2 model Radford et al. (2019) with a Mamba2 Dao & Gu (2024) model in order to achieve better performance.

3.3 HYDRA LAYER

To apply the sequence-to-sequence potential of the Hydra block Hwang et al. (2024), we provide a custom Hydra layer that combines the Hydra block with a standard feed-forward network. The main part of our standard architecture consists of Hydra layers. We have found that Hydra layers work more effectively than standard Mamba layers in several cases. Then we use the same PFFN that was introduced in Mamba4Rec Liu et al. (2024)

$$PFFN(H) = GELU(HW^{(1)} + h^{(1)})W^{(2)} + b^{(2)}$$

Where $W^{(1)} \in \mathbb{R}^{D \times 4D}$, $W^{(2)} \in \mathbb{R}^{D \times 4D}$, $b^{(1)} \in \mathbb{R}^D$ are parameters of two dense layers and we use GELU activation.

3.3.1 PREDICTION LAYER

Prediction layer is adopted from SASRec and Mamba4Rec, last item embedding is used to generate the final prediction scores:

$$\hat{y} = Softmax(hE^T) \in \mathbb{R}^{|V|}$$

where $h \in \mathbb{R}^D$ is the last item embedding from the Hydra layer and $\hat{y} \in \mathbb{R}^{|V|}$ represents the probability distribution over the next item in the item set V.

3.4 BENCHMARKS

We tested our models on 3 benchmarks: Amazon Reviews '23 Beauty and Personal care, Amazon Reviews '23 Video Games and MovieLens-1M.

All experiments were conducted using the same initial hyperparameters for better interpretability.

The number listed in parentheses after the number of parameters of each models is the number of parameters after deducting the size of all embedding layers if present.

3.4.1 LATENCY

For the LLMs, the measure tokens-per-second was selected, whilst the average time to generate suggestions for a single user was chosen to compare the others.

Since all of the latency values that we are dealing with are relatively small, it is important to ascertain the measurement's margin of error in order to comprehend the experimental results that were produced.

Consequently, the bootstrap sampling method was applied. 30 samples comprising 1500 users randomly chosen from the original dataset were used to individually measure the latency values. Following the removal of outliers, the distribution of the latencies was examined, and the confidence intervals were computed.

The latency measurements were performed on one Nvidia A100-80GB graphics card.

3.4.2 INTERPRETATION OF RESULTS

It is evident that Hydra shows metrics that are similar, sometimes better to those of Mamba, but its latency is four to five times lower. Simultaneously, models such as LlamaRec exhibit superior performance; however, their substantial parameter count may constrain their applicability in real-world situations. Metrics-wise and latency-wise, SSM-based models outperform the baseline SASRec model on average.

According to the study, SSMs can outperform LLMs while still operating at a greater speed, since they require less number of parameters.

Table 1: Amazon Reviews '23 Beauty and Personal care

Model	HT@10	NDCG@10	MRR@10	Latency	# Parameters
SASRec	0.048	0.028	0.022	0.117	14M (100k)
Mamba4Rec	0.048	0.030	0.025	0.0075	14M (80k)
MamRec	0.031	0.020	0.017	2.51	130M
GPT4Rec	0.030	0.025	0.015	2.32	117M
2Mamba4Rec	0.048	0.031	0.028	0.0110	14M (80k)
LlamaRec	**0.093**	0.040	**0.040**	-	7B
Hydra4Rec	0.070	**0.043**	0.035	**0.0042**	840k

Table 2: Amazon Reviews '23 Video Games

Model	HT@10	NDCG@10	MRR@10	Latency	# Parameters
SASRec	0.119	0.073	0.059	0.129	1.8M (100k)
Mamba4Rec	0.107	0.062	0.048	0.0088	1.8M (80k)
MamRec	0.083	0.033	0.025	2.51	130M
GPT4Rec	0.080	0.042	0.026	2.32	117M
2Mamba4Rec	0.118	0.061	0.048	0.0103	1.8M (80k)
LlamaRec	**0.150**	**0.098**	**0.064**	-	7B
Hydra4Rec	0.112	0.059	0.044	**0.0037**	750k

Table 3: MovieLens-1M

Model	HT@10	NDCG@10	MRR@10	Latency	# Parameters
SASRec	0.224	0.117	0.084	0.128	320k (100k)
Mamba4Rec	0.303	0.178	0.139	0.0020	300k (80k)
MamRec	0.201	0.072	0.064	2.51	130M
GPT4Rec	0.212	0.074	0.060	3.23	117M
2Mamba4Rec	0.340	0.193	0.148	0.0034	300k (80k)
Hydra4Rec	**0.308**	**0.179**	**0.140**	**0.0005**	290k
LlamaRec	0.148	0.067	-	-	7B

3.5 Application of ORPO to recommender systems

To improve quality and incorporate human preference into the LLM, we used the ORPO technique described above. Main motivation is ability to slightly improve quality of the large model, showing the potential of preference optimization techniques in recommeder systems.

3.5.1 Baseline

As a baseline, we considered the standard LlamaRec pipeline with Llama 7B LLM. Both retrieval and LLM were fine-tuned as described in the LlamaRec paper prior to the ORPO procedure.

3.5.2 Data

We used the standard LlamaRec format. To construct the text input, we prepend an instruction to describe the task, followed by both history and candidate items, represented by their titles. Our prompt template is: " *Instruction*: Given the user's history in chronological order, recommend an item from the candidate pool with its index letter; *Input*: User history: history ; *Candidate pool*: candidates ; *Answer*: label " where history, candidates and label are replaced by history item titles, candidate item titles and the labeled of each data item. For inference, the label position is left empty for the model to provide predictions.

We then created a pair data set, where each pair consists of a winner and a loser. It consists of 2 parts. The first part was created from the last 2 items from the user history, where the winner is the item with the higher rating and the loser the opposite. To create the second part, we used negative sampling training LightFM model and took the last user's movie as the winner and the worst recommendation of LightFM as the loser.

3.5.3 Experimental setup

For the ORPO procedure we used lr = 8e-6, ORPO uses very low learning rates compared to traditional SFT or even DPO. This value of 8e-6 comes from the original paper. beta = 0.1 An appendix from the original paper shows how it's been selected with an ablation study. We use paged adamw optimizer with gradient accumulation steps = 4. We train for 1 training epoch and evaluate each 0.2

steps, also we set warmup steps to 10. We use linear scheduler type. Other parameters are dependant on capabilities of your resources. Training was proceed on 1 A100 80GB videocard.

3.5.4 ORPO RESULTS

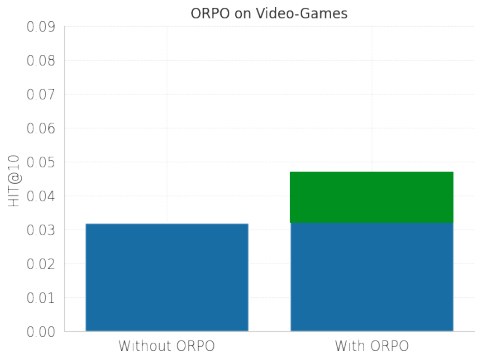


Figure 1: Video-Games

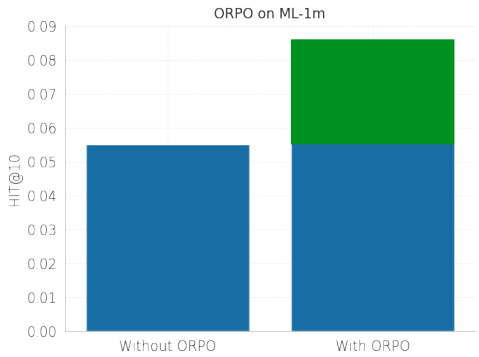


Figure 2: ML-1m

ORPO procedure provide slight impact on LLM perfomace not affecting original latency. See Figure 2 and 3.

3.6 ADAPTIVE BATCHING

3.6.1 ALGORITHM

We were able to apply the concept of adaptive batching to the aforementioned algorithm (USGM) and enhance its performance in training SSMs in the sequential recommendations domain. The approach presented in this section is novel, since adaptive batching techniques have never been applied to the Rodomanov et al. (2024).

In stochastic optimization, the variance of the gradient estimates reduces as the batch size increases. Specifically, the variance decreases proportionally to $\frac{1}{B}$. Concurrently, the standard deviation (σ) of the gradient estimates decreases at the rate of $\frac{1}{\sqrt{B}}$. From here 1.3 , we can derive the equation 2.1 Thus, the variance and standard deviation of the gradient estimations decrease as the batch size increases. But as the batch size increases, the impact of this reduction decreases. Additional bigger increases in batch size result in insignificant decreases in variance, as accuracy gains have landed on a plateau. Therefore, increasing the batch size further at this stage is not effective, since it will not provide any improvements in accuracy, but will require additional computational resources.

$$\hat{\beta}_{k+1} \leq L_\nu r_{k+1}^{1+\nu} + \frac{\sigma_{k+1} r_{k+1}}{\sqrt{B}}, \tag{2.1}$$

Further into the paper we will consider this upper estimate an equality to indirectly calculate the values of $L_\nu r_{k+1}^{1+\nu}$ and $\sigma_{k+1} r_{k+1}$, since we know the exact value of $\hat{\beta}_{k+1}$ on each iteration of the algorithm.

As the coefficient $\hat{\beta}_{k+1}$ is computed on every iteration of the algorithm and the batch size is known, we can use the Weighted Least Squares method (with loss defined by the equation 2.2) in linearized axes ($\frac{1}{\sqrt{B}}$ as the X axis and the $\hat{\beta}_{k+1}$ as the Y axis) to compute the values of $L_\nu r_{k+1}^{1+\nu}$ and $\sigma_{k+1} r_{k+1}$, the intercept and the slope of the linearized function respectively.

$$L = \sum_{k=0}^{K} (F(c_1, c_2, B_k) - \hat{\beta}_{k+1})^2 \cdot (1-\alpha)^{K-k}, \tag{2.2}$$

where F is the sought function $F = c_1 + \frac{c_2}{\sqrt{B}} \approx \hat{\beta}_{k+1}$, c_1 and c_2 are the values of $L_\nu r_{k+1}^{1+\nu}$ and $\sigma_{k+1} r_{k+1}$ respectively, predicted by the WLS algorithm. The value of alpha was empirically set to $\alpha = 0.01$, s.t. the first points almost diminish.

In the proposed algorithm, we will increase the batch size on each iteration ($B_k := B_{k-1} + B_0$) if $c_1 \neq \frac{c_2}{sqrt(B)}$ and recalculate the values of c_1 and c_2 using WLS ($L \underset{c_1, c_2}{\rightarrow} \min$).

When we reach the plateau ($c_1 = \frac{c_2}{\sqrt{B}}$), we fix the current batch size until the end of the epoch. We consider this batch size perfect w.r.t the current conditions and label it as B_i^*.

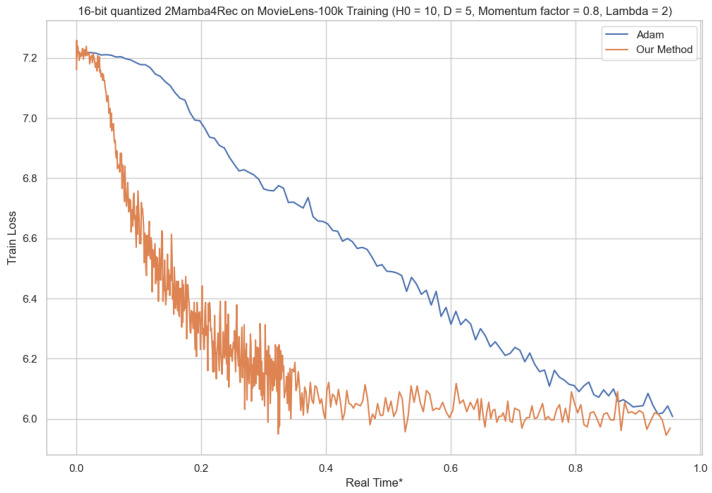


Figure 3: Training results

When the epoch ends, we lower the batch size ($B_i := \frac{B_{i-1_k}}{\lambda}$), since the perfect batch size (B_i^*) could have changed since the last epoch. However, if it's bigger than $\frac{B_{i-1_k}}{\lambda}$, the algorithm will increase it during the next iterations. By doing so, we give the algorithm the ability to not continue training with excessive resource if the calculated batch size turns out to be too big in the current circumstances or to increase it even more if needed.

Since $B_k := B_{k-1} + B_0$, $B_k = m_k B_0 : \forall k \ (m_k \in \mathbb{Z})$. That means we can use the original Dataloader class from PyTorch (Paszke et al. (2019)) wihout any modifications and take m_k batches of initial size (B_0) on each iteration.

Experiments have shown that the use of values of beta and the batch size only from the current epoch in WLS almost do not change the desired Batch size (B_i) in comparison with using all of the previous values recorded during the training procedure. This proves that the values of $c1$ and $c2$ are, indeed, true for all parts of the loss landscape as we can see during the training procedure. This fact goes along well with the proposed theoretical assumptions, as these coefficients in part consist of values, that are constant for the whole function (the Hölder-Lipschitz constant L_ν) and therefore should not change when we adjust the values of model's parameters x during optimization.

Since the number of iterations of the algorithm with adaptive batching and of those, with which it was compared, differs, since the equality $N_{\text{iterations}} = \frac{\text{Dataset sise}}{\text{Batch size}} = const$ is not true for the proposed algorithm, a scaled measure, which is proportional to the epoch number is used as the X axis in the comparison plot, displayed in the Figure 3.

The experiments conducted using the 2Mamba4Rec pipeline, introduced in this paper, on the MovieLens-100k dataset have shown promising results regarding the innovative adaptive batching algorithm, applied to the USGM optimization algorithm by Rodomanov et al. (2024), when compared with the well-known Adam optimizer by Kingma & Ba (2017).

4 DISCUSSION

Limitations. ORPO and LLM related procedures require a lot of computational resources, especially for larger models. LLM inference itself also requires optimization, but this does not make them unusable in real tasks. SSM based models might have poor CPU performance due to their structure and implementation. Furthermore, optimizer that we proposed has poor robustness among its hyperparameters.
Potential impact. We believe that ideas and obtained results from our work can inspire the community to continue the research in the field of recommender systems.
Broader impact. The goal of this paper is to advance the field of Machine Learning. There are many potential societal consequences of our work, none of which we feel must be specifically highlighted here.

ACKNOWLEDGMENTS

The paper was written from July 1 to July 24 as a part of the Sirius "Big Challenges" project programme in the "Big Data, Artificial Intelligence, Financial Technologies, and Machine Learning" category. We express our appreciation to MIPT, the Moscow Institute of Physics and Technology, for providing the computational resources required to finish this study.

REFERENCES

Guo Chen, Yifei Huang, Jilan Xu, Baoqi Pei, Zhe Chen, Zhiqi Li, Jiahao Wang, Kunchang Li, Tong Lu, and Limin Wang. Video mamba suite: State space model as a versatile alternative for video understanding, 2024. URL https://arxiv.org/abs/2403.09626.

Tri Dao and Albert Gu. Transformers are ssms: Generalized models and efficient algorithms through structured state space duality, 2024. URL https://arxiv.org/abs/2405.21060.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159, 07 2011.

Mehmet Hamza Erol, Arda Senocak, Jiu Feng, and Joon Son Chung. Audio mamba: Bidirectional state space model for audio representation learning, 2024. URL https://arxiv.org/abs/2406.03344.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces, 2024. URL https://arxiv.org/abs/2312.00752.

Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces, 2022. URL https://arxiv.org/abs/2111.00396.

Jiwoo Hong, Noah Lee, and James Thorne. Orpo: Monolithic preference optimization without reference model, 2024. URL https://arxiv.org/abs/2403.07691.

Sukjun Hwang, Aakash Lahoti, Tri Dao, and Albert Gu. Hydra: Bidirectional state space models through generalized matrix mixers, 2024. URL https://arxiv.org/abs/2407.09941.

Wang-Cheng Kang and Julian McAuley. Self-attentive sequential recommendation, 2018. URL https://arxiv.org/abs/1808.09781.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017. URL https://arxiv.org/abs/1412.6980.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13): 3521–3526, March 2017. ISSN 1091-6490. doi: 10.1073/pnas.1611835114. URL http://dx.doi.org/10.1073/pnas.1611835114.

Jinming Li, Wentao Zhang, Tian Wang, Guanglei Xiong, Alan Lu, and Gerard Medioni. Gpt4rec: A generative framework for personalized recommendation and user interests interpretation, 2023. URL https://arxiv.org/abs/2304.03879.

Chengkai Liu, Jianghao Lin, Jianling Wang, Hanzhou Liu, and James Caverlee. Mamba4rec: Towards efficient sequential recommendation with selective state space models, 2024. URL https://arxiv.org/abs/2403.03900.

Hao Liu, Matei Zaharia, and Pieter Abbeel. Ring attention with blockwise transformers for near-infinite context, 2023. URL https://arxiv.org/abs/2310.01889.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library, 2019. URL https://arxiv.org/abs/1912.01703.

Alec Radford, Jeff Wu, Rewon Child, D. Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners, 2019. URL https://www.semanticscholar.org/paper/Language-Models-are-Unsupervised-Multitask-Learners-Radford-Wu/9405cc0d6169988371b2755e573cc28650d14dfe.

Herbert E. Robbins. A stochastic approximation method. *Annals of Mathematical Statistics*, 22: 400–407, 1951. URL https://api.semanticscholar.org/CorpusID:16945044.

Anton Rodomanov, Ali Kavis, Yongtao Wu, Kimon Antonakopoulos, and Volkan Cevher. Universal gradient methods for stochastic convex optimization, 2024. URL https://arxiv.org/abs/2402.03210.

Yuda Wang, Xuxin He, and Shengxin Zhu. Echomamba4rec: Harmonizing bidirectional state space models with spectral filtering for advanced sequential recommendation, 2024. URL https://arxiv.org/abs/2406.02638.

Zhenrui Yue, Sara Rabhi, Gabriel de Souza Pereira Moreira, Dong Wang, and Even Oldridge. Llamarec: Two-stage recommendation using large language models for ranking, 2023. URL https://arxiv.org/abs/2311.02089.

Lianghui Zhu, Bencheng Liao, Qian Zhang, Xinlong Wang, Wenyu Liu, and Xinggang Wang. Vision mamba: Efficient visual representation learning with bidirectional state space model, 2024. URL https://arxiv.org/abs/2401.09417.

A Appendix A. Datasets

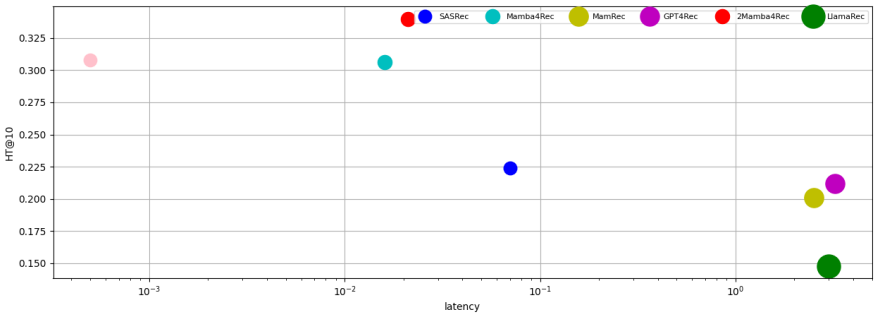
A table regarding the information about the datasets using in the paper is provided for scale. 4

Table 4: Datasets

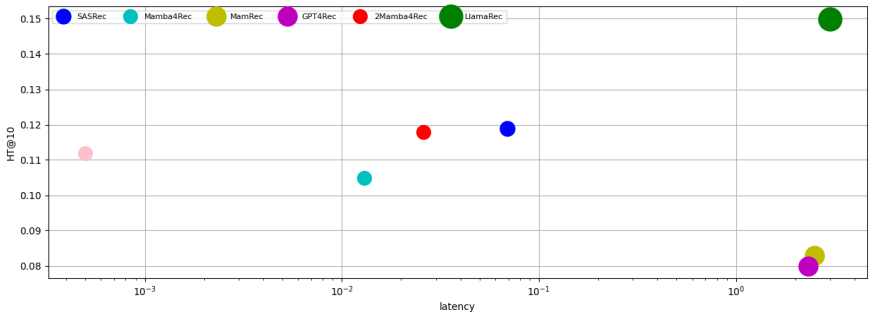
Datasets	Users	Items	Reviews
Beauty and P.C.	750,835	211,452	6,860,059
MovieLens-1M	6,041	3,417	999,611
Video Games	98,907	26,355	857,505

B Appendix B. Benchmarks plots

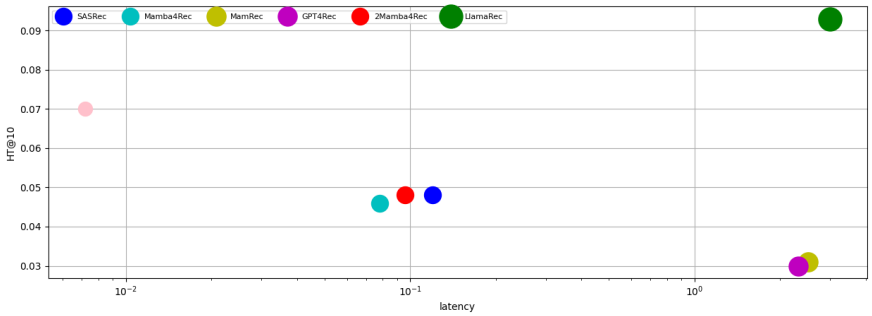
For a better understanding of our results, we propose plots with latency and the HIT@10 metric. Note that for better visualization, a logarithmic scale for latency is used.



(a) Comparison of performance of different models on the Beauty dataset.



(b) Comparison of performance of different models on the MovieLens-1M dataset.



(c) Comparison of performance of different models on the VideoGames dataset.