# FLASH4Rec: A Lightweight and Sparsely Activated Transformer for User-Aware Sequential Recommendation

YaChen Yan, Liubo Li

August 6, 2023

# Summary

# Introduction

# Transformer Architecture for Recommendation

- SASRec: The model is trained to predict the next item in the sequence, and during inference, it can recommend a list of items that the user is likely to interact with next.
- BERT4Rec: BERT4Rec applies the masked language modeling technique from BERT to recommendation systems.
- etc.

# Transformer Architecture for Recommendation Cont.

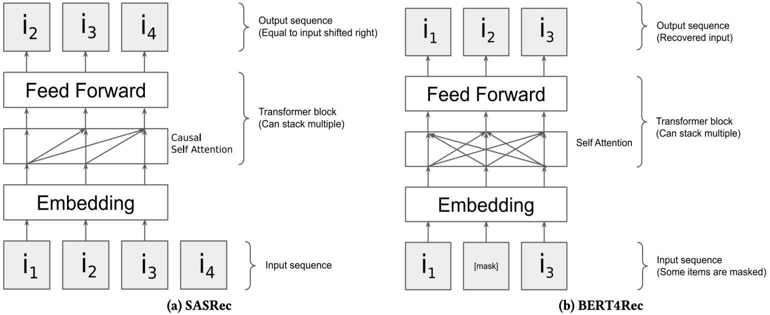


Figure: SASRec vs. BERT4Rec[1]

[1] **Petrov et al.**

# Motivation

- User-Aware Recommendation
  - Leveraging user demographics and profiles to generate dynamic item sequence representation.
- Lightweight but still performant Transformer Layer
  - A more efficient alternative to multi-head self-attention.
  - Linear Attention: $O(N^2) \rightarrow O(N)$.
  - An alternative to FFN having higher modeling capacity

# Proposed Methods

# The Architecture of FLASH4Rec

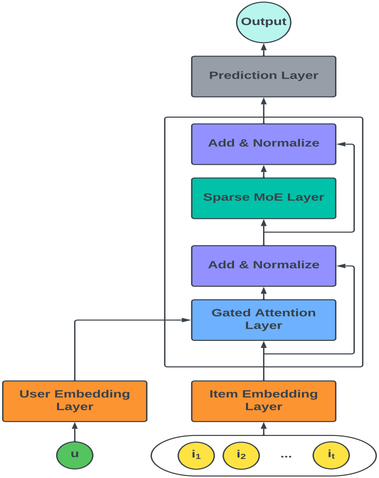


Figure: The Architecture of FLASH4Rec

# FLASH4Rec

We introduce a new architecture called FLASH4Rec, which efficiently models item dependencies in users' historical behavior sequences.

- The architecture consists of a Gated Attention Layer and a Sparsely-Gated Mixture-of-Experts Layer.
- The Gated Attention Layer computes user-aware item sequence representations, while the SparseMoE Layer increases the model's capacity without increasing computational costs.
- To prevent overfitting, we include a Top-K Dropout mechanism that encourages the model to learn from long-tail attention positions.

# Gated Attention Layer Cont.

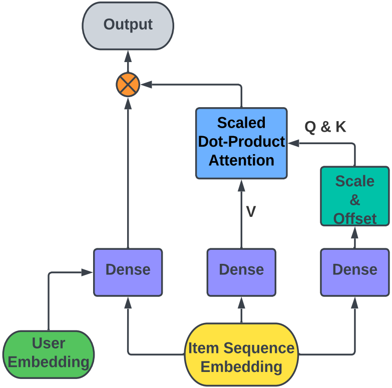


Figure: The Architecture of Gated Attention Layer

# Gated Attention Layer Cont.

QKV:

$$Z = \sigma(X_I W_z) \qquad \in \mathbb{R}^{T\times k} \tag{1}$$

$$V = \sigma(X_I W_v) \qquad \in \mathbb{R}^{T\times k} \tag{2}$$

Gating:

$$U = \sigma(\mathrm{Concat}(X_I, X_U) W_u) \qquad \in \mathbb{R}^{T\times k} \tag{3}$$

Attention:

$$A = \mathrm{softmax}\left(\frac{\mathcal{Q}(Z)\mathcal{K}(Z)^\top}{\sqrt{d_k}}\right) \qquad \in \mathbb{R}^{T\times T} \tag{4}$$

$$A = 1 + \left(\frac{\mathcal{Q}(Z)}{\|\mathcal{Q}(Z)\|}\right)^\top \left(\frac{\mathcal{K}(Z)}{\|\mathcal{K}(Z)\|}\right) \qquad \in \mathbb{R}^{T\times T} \tag{5}$$

Output:

$$O = U \otimes AV \tag{6}$$

# Sparse Mixture-of-Experts Layer

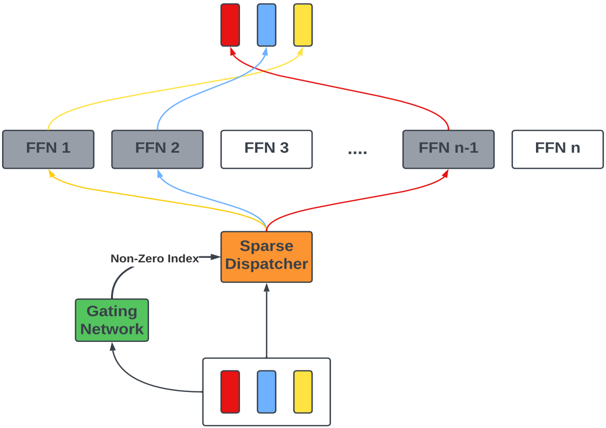


Figure: The Architecture of Sparse Mixture-of-Experts Layer Layer

# Noisy Gating Network

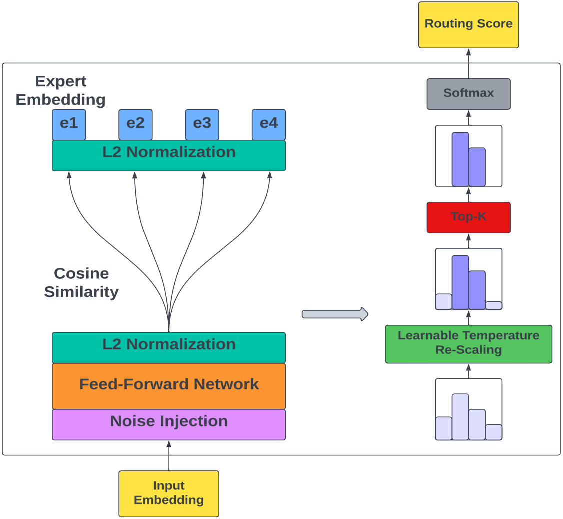


Figure: The Noisy Gating Network within Sparse Mixture-of-Experts Layer

# SparseMoE Components

- Experts: Feed-Forward Network (FNN)
- Noisy Gating Network:
  - A neural network selecting the Top-1 experts per item embedding.
  - Load Balance Regularization.
- Sparse Dispatcher
  - Dispatch input and sparsely activate corresponding experts.
  - Combine each expert's output.

# Top-K Dropout

- Item Popularity: Power-Law Distribution
- Bias the model to overly rely on on popular item's embedding
- Balance: short-tail item embeddings vs. long-tail item embeddings

# Top-K Dropout

Formally given a self-attention weight matrix $A \in \mathbb{R}^{T \times T}$, we firstly compute its Top-K position indicator $S_A$, in which each element $S_{i,j}$ is defined as:

$$S_{i,j} = \begin{cases} 1 & \text{if } A_{i,j} \text{ is in the top } k \text{ elements of } A_{i,}. \\ 0 & \text{otherwise.} \end{cases} \tag{7}$$

Next, we want to randomly dropout self-attention weights within the Top-K positions to produce the Top-K mask matrix $M_A$ with dropout rate $p$:

$$M_{i,j} = \begin{cases} 0 & \text{if } s_{i,j} * Bernoulli(p) = 1 \\ 1 & \text{otherwise,} \end{cases} \tag{8}$$

# Top-K Dropout

After the dropout is applied, we re-scale the self-attention weights by scaling factor $f$:

$$f = \frac{1}{1.0 - (\sum_{i=1}^{T} \sum_{j=1}^{T} A_{i,j} * M_{i,j} / \sum_{i=1}^{T} \sum_{j=1}^{T} A_{i,j})} \tag{9}$$

# Experiment

# Model Performance Comparison

Table: Performance Comparison of Different Algorithms on ML-1M, ML-20M and Yelp Dataset.

| | ML-1M | | ML-20M | | Yelp | |
|---|---|---|---|---|---|---|
| Model | Recall@10 | NDCG@10 | Recall@10 | NDCG@10 | Recall@10 | NDCG@10 |
| MF-BPR | 0.0740 | 0.0377 | 0.0807 | 0.0407 | 0.0191 | 0.0092 |
| GRU4Rec | 0.2132 | 0.1093 | 0.1544 | 0.0839 | 0.0113 | 0.0048 |
| SASRec | 0.1993 | 0.1078 | 0.1439 | 0.0724 | 0.0146 | 0.0076 |
| BERT4Rec | 0.2584 | 0.1392 | 0.2393 | 0.1310 | 0.0149 | 0.0079 |
| FLASH4Rec | **0.2841** | **0.1568** | **0.2554** | **0.1487** | **0.0151** | **0.0081** |

# Evaluation on Efficiency

Table: Efficiency Comparison of BERT4Rec and FLASH4Rec on ML-1M Dataset.

| | Params | FLOPs |
|---|---|---|
| BERT4Rec | 3.08M | 74.12M |
| FLASH4Rec | 3.16M | 63.10M |

# Ablation Study

Table: Abalation Study about key componenets of FLASH4Rec on ML-1M Dataset.

| | Recall@10 | NDCG@10 |
|---|---|---|
| FLASH4Rec | 0.2841 | 0.1568 |
| w/o Gated Attention | 0.2690 | 0.1488 |
| w/o SparseMoE Layer | 0.2787 | 0.1535 |
| w/o Top-K Dropout | 0.2765 | 0.1512 |

# Conclusions

# Conclusions

- We introduce FLASH4Rec, a Transformer variant for sequential recommendation, employing a Gated Attention Layer and Sparsely-Gated Mixture-of-Experts Layer for efficient and effective user-aware item sequence representation.
- The Top-K Dropout technique is designed to facilitate model learning from low-attention positions, thereby reducing over-fitting.

# End

Thank You!