**ADER: Adaptively Distilled Exemplar Replay Towards Continual Learning for Session-based Recommendation**

FEI MI, XIAOYU LIN, and BOI FALTINGS, Artificial Intelligence Laboratory, Swiss Federal Institute of Technology Lausanne (EPFL), Switzerland

Session-based recommendation has received growing attention recently due to the increasing privacy concern. Despite the recent success of neural session-based recommenders, they are typically developed in an offline manner using a static dataset. However, recommendation requires continual adaptation to take into account new and obsolete items and users, and requires "continual learning" in real-life applications. In this case, the recommender is updated continually and periodically with new data that arrives in each update cycle, and the updated model needs to provide recommendations for user activities before the next model update. A major challenge for continual learning with neural models is catastrophic forgetting, in which a continually trained model forgets user preference patterns it has learned before. To deal with this challenge, we propose a method called Adaptively Distilled Exemplar Replay (ADER) by periodically replaying previous training samples (i.e., exemplars) to the current model with an adaptive distillation loss. Experiments are conducted based on the state-of-the-art SASRec model using two widely used datasets to benchmark ADER with several well-known continual learning techniques. We empirically demonstrate that ADER consistently outperforms other baselines, and it even outperforms the method using all historical data at every update cycle. This result reveals that ADER is a promising solution to mitigate the catastrophic forgetting issue towards building more realistic and scalable session-based recommenders.

ACM Reference Format:
Fei Mi, Xiaoyu Lin, and Boi Faltings. 2020. ADER: Adaptively Distilled Exemplar Replay Towards Continual Learning for Session-based Recommendation. In The 14th ACM Recommender Systems Conference, Sept. 22–26, 2020, Virtual Event, Brazil. ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/1122445.1122456

1 **INTRODUCTION**

Due to new privacy regulations that prohibit building user preference models from historical user data, utilizing anonymous short-term interaction data within a browser session becomes popular. Session-based Recommendation (SR) is therefore increasingly used in real-life online systems, such as E-commerce and social media. The goal of SR is to make recommendations based on user behavior obtained in short web browser sessions, and the task is to predict the user’s next actions, such as clicks, views, and even purchases, based on previous activities in the same session.

Despite the recent success of various neural approaches [11, 16, 18, 20], they are developed in an offline manner, in which the recommender is trained on a very large static training set and evaluated on a very restrictive testing set in a one-time process. However, this setup does not reflect the realistic use cases of online recommendation systems. In reality, a recommender needs to be periodically updated with new data streaming in, and the updated model is supposed to provide recommendations for user activities before the next update. In this paper, we propose a continual learning setup to consider such realistic recommendation scenarios.
The major challenge of continual learning is catastrophic forgetting [6, 23]. That is, a neural model updated on new data distributions tends to forget old distributions it has learned before. A naive solution is to retrain the model using all historical data every time. However, it suffers from severe computation and storage overhead in large-scale recommendation applications.

To this end, we propose to store a small set of representative sequences from previous data, namely exemplars, and replay them each time when the recommendation model needs to be trained on new data. Methods using exemplars have shown great success in different continual learning [3, 29] and reinforcement learning [2, 32] tasks. In this paper, we propose to select representative exemplars of an item using an herding technique [29, 36], and its exemplar size is proportional to the item frequency in the near past. To enforce a stronger constraint on not forgetting previous user preferences, we propose a regularization method based on the well-known knowledge distillation technique [12]. We propose to apply a distillation loss on the selected exemplars to preserve the model’s knowledge. The distillation loss is further adaptively interpolated with the regular cross-entropy loss on the new data by considering the difference between new data and old ones to flexibly deal with different new data distributions.

Altogether, (1) we are the first to study the practical continual learning setting for the session-based recommendation task; (2) we propose a method called Adaptively Distilled Exemplar Replay (ADER) for this task, and benchmark it with state-of-the-art continual learning techniques; (3) experiment results on two widely used datasets empirically demonstrate the superior performance of ADER and its ability to mitigate catastrophic forgetting.¹

2 RELATED WORK

2.1 Session-based Recommendation

Session-based recommendation (SR) can be formulated as a sequence learning problem to be solved by recurrent neural networks (RNNs). The first work (GRU4Rec, [11]) uses a gated recurrent unit (GRU) to learn session representations from previous clicks. Based on GRU4Rec, [10] proposes new ranking losses on relevant sessions, and [34] proposes to augment training data. Attention operation is first used by NARM [18] to pay attention to specific parts of the sequence. Based on NARM, [20] proposes STAMP to model users’ general and short-term interests using two separate attention operations, and [30] proposes RepeatNet to predict repetitive actions in a session. Motivated by the recent success of Transformer [35] and BERT [5] for language model tasks, [16] proposed SASRec using Transformer, and [33] proposed BERT4Rec to model bi-directional information. Despite the broad exploration and success, the above methods are all studied in a static and offline manner. Recently, the incremental and streaming nature of SR is pointed out by [9, 27].

Besides neural approaches, several non-parametric methods have been proposed. [15] proposed SKNN to compare the current session with historical sessions in the training data. Lately, variations [8, 21] of SKNN have been proposed to consider the position of items in a session or the timestamp of a past session. [7, 24–26] applies a non-parametric structure called context tree. Although these methods can be efficiently updated, the realistic continual learning setting and the corresponding forgetting issue remain to be explored.

2.2 Continual Learning

The major challenge for continual learning is catastrophic forgetting [6, 23]. Methods designed to mitigate catastrophic forgetting fall into three categories: regularization [17, 19, 38], Exemplar Replay [3, 4, 29] and dynamic architectures [22, 31]. Methods using dynamic architectures increase model parameters throughout the training process, which leads to an unfair comparison with other methods. In this work, we focus on the first two categories.

¹Code is available at: https://github.com/DoubleMuL/ADER
Towards Continual Learning for Session-based Recommendation RecSys 2020, Sept. 22–26, 2020, Virtual Event, Brazil

Fig. 1. An visualization of the continual learning setup. At each update cycle $t$, the model is trained with data $D_t$, and the updated model $f(\theta_t)$ is evaluated w.r.t. to data $D_{t+1}$ before the next model update.

**Regularization** methods add specific regularization terms to consolidate knowledge learned before. [19] introduces knowledge distillation [12] to penalize model logit change, and it is widely employed by [3, 14, 29, 37, 39]. [1, 17, 38] propose to penalize changes on parameters that are crucial to old knowledge according to various importance measures. **Exemplar Replay** methods store past samples, a.k.a. exemplars, and replay them periodically to prevent model forgetting previous knowledge. Besides selecting exemplars uniformly, [29] incorporates the Herding technique [36] to select exemplars, and it soon becomes popular [3, 14, 37, 39].

3 METHODOLOGY

In this section, we first introduce some background in Section 3.1 and a formulation of the continual learning setup in Section 3.2. In Section 3.3, we propose our method called “Adaptively Distilled Exemplar Replay” (ADER).

3.1 Background on Neural Session-based Recommenders

A user action in SR is a click or view on an item, and the task is to predict the next user action based on a sequence of user actions in the current web-browser session. Existing neural models $f(\theta)$ typically contain two modules: a feature extractor $\phi(x)$ to compute a compact sequence representation of the sequence $x$ of previous user actions, and an output layer $\omega(\phi(x))$ to predict the next user action. Various recurrent neural networks [10, 11] and attention mechanisms [16, 18, 20] have been proposed for $\phi$, and the common choices for the output layer $\omega$ is fully-connect layers[11] or bi-linear decoders [16, 18]. In this paper, we base our comparison on SASRec [16], and we refer readers to model details in the original paper to avoid verbosity. Nevertheless, the techniques proposed and compared in this paper are agnostic to $f(\theta)$, therefore, a more thorough comparison using different $f(\theta)$ are left for interesting future work.

3.2 Formulation of Continual Learning for Session-based Recommendation

In this section, we formulate the continual learning setting for the session-based recommendation task to simulate the realistic use cases of training a recommendation model continually. To be specific, at an update cycle $t$, the recommendation model $f(\theta_{t-1})$ obtained until the last update cycle $t-1$ needs to be updated with new incoming data $D_t$. After $f(\theta_{t-1})$ is trained on $D_t$, the updated model $f(\theta_t)$ is evaluated w.r.t. the incoming data $D_{t+1}$ before the next update cycle $t+1$. A visualization of the continual learning setup is illustrated in Fig. 1, where a recommendation model is continually trained and tested upon receiving data in sequential update cycles.

3.3 Proposed Solution: Adaptively Distilled Exemplar Replay (ADER)

3.3.1 Exemplar Replay. To alleviate the widely-recognized catastrophic forgetting issue in continual learning, the model needs to preserve old knowledge it has learned before. To this end, we propose to store past samples, a.k.a...
exemplars, and replay them periodically to preserve previous knowledge. To maintain a manageable memory footprint, we only store a fixed total number of exemplars throughout the entire continual learning process. Two decisions need to be made at each cycle $t$: (1) how many exemplars should be stored for each item/label? (2) what is the criterion for selecting exemplars of an item/label?

First, we design the number of exemplars of each appeared item in $I_t$ (i.e. the set of appeared items until cycle $t$) to be proportional to its appearance frequency. In other words, more frequent and popular items contribute a larger portion of selected exemplars to be replayed to the next cycle. Suppose we store $N$ exemplars in total, the number of exemplars $m_{t,i}$ at cycle $t$ for a item $i \in I_t$ is:

$$m_{t,i} = N \cdot \frac{|\{(x, y = i) \in D_t \cup E_{t-1}\}|}{|D_t \cup E_{t-1}|},$$

where the second term is the probability that item $i$ appears in the current update cycle, as well as in the exemplars $E_{t-1}$ we kept from the last cycle. Therefore, the exemplar sizes of different items to be select in the cycle $t$ can be encoded as a vector $M_t = [m_1, m_2, ..., m_{|I_t|}]$.

Second, we need to decide which samples to select as exemplars for each item. We propose to use a herding technique [29, 36] to select the most representative sequences of an item in an iterative manner based on the distance to the mean feature vector of the item. In each iteration, one sample from $D_t \cup E_{t-1}$ that best approximates the average feature vector ($\mu$) over all training examples of this item ($y$) is selected to $E_t$. The details are presented in Algorithm 1.

3.3.2 Adaptive Distillation on Exemplars. The number of exemplars should be reasonably small to reduce memory overhead. As a consequence, the constraint to prevent the recommender forgetting previous user preference patterns is not strong enough. To enforce a stronger constrain on not forgetting old user preference patterns, we propose to use a knowledge distillation loss [12] on exemplars to better consolidate old knowledge.

At a cycle $t$, the set of exemplars to be replayed is $E_{t-1}$ and the set of items till the last cycle is $I_{t-1}$, the proposed knowledge distillation (KD) loss is written as:

$$L_{KD}(\theta_t) = -\frac{1}{|E_{t-1}|} \sum_{(x, y) \in E_{t-1}} \sum_{i=1}^{|I_{t-1}|} \hat{p}_i \cdot \log(p_i),$$

where $[\hat{p}_1, \ldots, \hat{p}_{|I_{t-1}|}]$ is predicted distribution (softmax of logits) over $I_{t-1}$ generated by $f(\theta_{t-1})$, and $[p_1, \ldots, p_{|I_{t-1}|}]$ is the prediction of $f(\theta_t)$ over $I_{t-1}$. $L_{KD}$ measures the difference between the outputs of the previous model and the current model on exemplars, and the idea is to penalize prediction changes on items in previous update cycles.

---

**Algorithm 1** ADER: ExemplarSelection at cycle $t$

**Input:** $S = D_t \cup E_{t-1}$; $M_t = [m_1, m_2, ..., m_{|I_t|}]$

**for** $y = 1, ..., |I_t|$ **do**

$\mathcal{P}_y \leftarrow \{x : (x, y) \in S\}$

$\mu \leftarrow \frac{1}{|\mathcal{P}_y|} \sum_{x \in \mathcal{P}_y} \phi(x)$

**for** $k = 1, ..., m_y$ **do**

$x^k \leftarrow \arg\min_{x \in \mathcal{P}_y} \|\mu - \frac{1}{k} (\phi(x) + \sum_{j=1}^{k-1} \phi(x^j))\|$

**end for**

$E_y \leftarrow \{(x^1, y), ..., (x^{m_y}, y)\}$

**end for**

**Output:** exemplar set $E_t = \bigcup_{y=1}^{|I_t|} E_y$

---

**Algorithm 2** ADER: UpdateModel at cycle $t$

**Input:** $D_t, E_{t-1}, I_t, I_{t-1}$

Initialize $\theta_t$ with $\theta_{t-1}$

**while** $\theta_t$ not converged **do**

Train $\theta_t$ with loss in Eq. (4)

**end while**

Compute $E_t$ using Algorithm 1 with $\theta_t$ and $M_t$ computed by Eq. (1)

**Output:** updated $\theta_t$ and new exemplar set $E_t$
Table 1. Statistics of the two datasets; “new actions” indicate the percentage of actions on new items in this update cycle; week/day 0 is only used for training, while week/day 16 is only used for testing.

|          | week 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|----------|--------|---|---|---|---|---|---|---|---|
| total actions | 70,739 | 37,586 | 31,089 | 32,687 | 30,419 | 57,913 | 52,225 | 57,100 | 69,042 |
| new actions | 100.00% | 18.25% | 13.26% | 11.29% | 10.12% | 9.08% | 6.64% | 6.35% | 5.42% |

|          | week 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
|----------|--------|----|----|----|----|----|----|----|
| total actions | 82,834 | 82,935 | 50,037 | 63,133 | 70,050 | 71,670 | 56,959 | 77,065 | 993,483 |
| new actions | 5.22% | 3.02% | 3.01% | 1.78% | 1.83% | 0.78% | 0.45% | 0.27% | / |

|          | day 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|----------|------|---|---|---|---|---|---|---|---|
| total actions | 219,389 | 209,219 | 218,162 | 162,637 | 177,943 | 307,603 | 232,887 | 178,076 | 199,615 |
| new actions | 100.00% | 3.04% | 1.74% | 1.29% | 0.95% | 0.57% | 0.50% | 1.09% | 0.74% |

|          | day 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
|----------|------|----|----|----|----|----|----|----|
| total actions | 179,889 | 123,750 | 153,565 | 300,830 | 259,673 | 187,348 | 154,316 | 105,676 | 3,370,578 |
| new actions | 0.81% | 1.08% | 0.56% | 0.56% | 0.29% | 0.41% | 0.38% | 0.35% | / |

In practice, the size of incoming data and the number of new items varies in different cycles, therefore, the degree of need to preserve old knowledge varies. To this end, we propose an adaptive weight \( \lambda_t \) to combine \( L_{KD} \) with \( L_{CE} \): \( L_{ADER} = L_{CE} + \lambda_t \cdot L_{KD} \).

4 EXPERIMENTS

4.1 Dataset

Two widely used dataset are adopted: (1). DIGINETICA: This dataset contains click-streams data on a e-commerce site over a 5 months, and it is used for CIKM Cup 2016 (http://cikm2016.cs.iupui.edu/cikm-cup). (2). YOOCHOOSE: It is another dataset used by RecSys Challenge 2015 (http://2015.recsyschallenge.com/challenge.html) for predicting click-streams on another e-commerce site over 6 months.

As in [11, 16, 18, 20], we remove sessions of length 1 and items that appear less than 5 times. To simulate the continual learning scenario, we split the model update cycle of DIGINETICA by weeks and YOOCHOOSE by days as its volume is much larger. Different time spans also resemble model update cycles at different granulates. In total, 16 update cycles are used to continually train the recommender on both datasets. 10% of the training data of each update cycle is randomly selected as a validation set.

In general, \( \lambda_t \) increases when the ratio of the number of old items to that of new items increases, and when the ratio of the exemplar size to the current data size increases. The idea is to rely more on \( L_{KD} \) when the new cycle contains fewer new items or fewer data to be learned. The overall training procedure of \( ADER \) is summarized in Algorithm 2.

4.2 Evaluation Metrics

Two commonly used evaluation metrics are used: (1). Recall@k: The ratio when the desired item is among the top-k recommended items. (2). MRR@k: Recall@k does not consider the order of the items recommended, while MRR@k measures the mean reciprocal ranks of the desired items in top-k recommended items. For easier comparison, we reported the mean value of these two metrics averaged over all 16 update cycles.
4.3 Baseline Methods

Several widely adopted baselines in continual learning literature are compared:

- **Finetune**: At each cycle, the recommender trained till the last task is trained with the data from the current task.
- **Dropout** [28]: Dropout [13] is recently found by [28] that it effectively alleviates catastrophic forgetting. Based on Finetune, we applied dropout to every self-attention and feed-forward layer.
- **EWC** [17]: It is a well-known method to alleviate forgetting by regularizing parameters important to previous data estimated by the diagonal of a Fisher information matrix computed w.r.t. exemplars.
- **ADER** (c.f. Algorithm 2): The proposed method using adaptively distilled exemplars in each cycle with dropout.
- **Joint**: In each cycle, the recommender is trained (with dropout) using data from the current and all historical cycles. This is a common performance “upper bound” for continual learning.

The above methods are applied on top of the state-of-the-art base SR recommender SASRec [16] using 150 hidden units and 2 stacked self-attention blocks. During continual training, we set the batch size to be 256 on DIGINETICA and 512 on YOOCHOOSE. We use Adam optimizer with a learning rate of 5e-4. A total of 100 epochs are trained, and early stop is applied if validation performance (Recall@20) does not improve for 5 consecutive epochs. Other hyper-parameters are tuned to maximize Recall@20. The dropout rate of Dropout, ADER, and Joint is set to 0.3; 30,000 exemplars are used by default for EWC and ADER; $\lambda_{\text{base}}$ of ADER is set to 0.8 on DIGINETICA and 1.0 on YOOCHOOSE.

4.4 Overall Results on Two Datasets

Results averaged over 16 update cycles are presented in Table 2, and several interesting observations can be noted:

- **Finetune** already works reasonably well, especially on the less dynamic YOOCHOOSE dataset. The performance gap between Finetune and Joint is less significant than typical continual learning setups [14, 19, 29, 37]. The reason is that catastrophic forgetting is not severe since old items can frequently reappear in recommendation tasks.
- **EWC** only outperforms Finetune marginally, and it performs worse than Dropout.
- **Dropout** is effective, and it notably outperforms Finetune, especially on the more dynamic DIGINETICA dataset.
- **ADER** significantly outperforms other methods, and the improvement margin over other methods is larger on the more dynamic DIGINETICA dataset. Furthermore, it even outperforms Joint. This result empirically reveals that ADER is a promising solution for the continual recommendation setting by effectively preserving user preference patterns learned before.

Detailed disentangled performance at each update cycle is plotted in Figure 2. We can see that the advantage of ADER is significant on the more dynamic DIGINETICA dataset. On the less dynamic YOOCHOOSE dataset, the gain of ADER mainly comes from the more dynamic starting cycles with relatively more actions on new items. At later stable cycles with few new items, different methods show comparable performance, including the vanilla Finetune.

4.5 In-depth Analysis

In following experiments, we conducted an in-depth analysis of the results on the more dynamic DIGINETICA dataset.
Fig. 2. Disentangled Recall@20 (Top) and MRR@20 (Bottom) at each continual learning update cycle on two datasets.

|       | 10k  | 20k  | 30k  |
|-------|------|------|------|
| Recall@20 | 49.59% | 50.05% | 50.21% |
| Recall@10 | 36.92% | 37.40% | 37.52% |
| MRR@20  | 17.04% | 17.29% | 17.32% |
| MRR@10  | 16.17% | 16.42% | 16.45% |

Table 3. Different exemplar sizes for ADER.

|       | ER$_{random}$ | ER$_{loss}$ | ER$_{herding}$ | ADER$_{equal}$ | ADER$_{fix}$ | ADER |
|-------|---------------|-------------|----------------|----------------|--------------|------|
| Recall@20 | 49.14% | 49.31% | 49.34% | 49.92% | 50.09% | 50.21% |
| Recall@10 | 36.61% | 36.65% | 36.78% | 37.23% | 37.41% | 37.52% |
| MRR@20   | 16.79% | 16.90% | 16.85% | 17.23% | 17.29% | 17.32% |
| MRR@10   | 15.92% | 16.02% | 16.98% | 16.35% | 16.41% | 16.45% |

Table 4. Ablation study for ADER.

4.5.1 Different number of Exemplars. We studied the effect of a varying number of exemplars for ADER. Besides using 30k exemplars, we tested using only 10k/20k exemplars, and results are shown in Table 3. We can see that the performance of ADER only drops marginally as exemplar size decreases from 30k to 10k. This result reveals that ADER is insensitive to the number of exemplars, and it works reasonably well with smaller number of exemplars.

4.5.2 Ablation Study. In this experiment, we compared ADER to several simplified versions to justify our design choices. (i). $ER_{herding}$: A vanilla exemplar replay different from ADER by using a regular $L_{CE}$, rather than $L_{KD}$, on exemplars. (ii). $ER_{random}$: It differs from $ER_{herding}$ by selecting exemplars of an item at random. (iii). $ER_{loss}$: It differs from $ER_{herding}$ by selecting exemplars of an item with smallest $L_{CE}$. (iv). $ADER_{equal}$: This version differs from ADER by selecting equal number of exemplars for each item, that is, the assumption that more frequent items should be stored more is removed. (v). $ADER_{fix}$: This version differs from ADER by not using the adaptive $\lambda_t$ in Eq. (4), but a fixed $\lambda$.

Comparison results are presented in Table 4, and several observations can be noted: (1). Herding is effective to selected exemplars, indicated by the better performance of $ER_{herding}$ over $ER_{random}$ and $ER_{loss}$. (2). The distillation loss in Eq. (2) is helpful, indicated by the better performance of three versions of ADER over three vanilla $ER$ methods. (3). Selecting exemplars proportional to item frequency is helpful, indicated by the better performance of ADER over $ADER_{equal}$. (4). The adaptive $\lambda_t$ in Eq. (2) is helpful, indicated by the better performance of ADER over $ADER_{fix}$.

5 CONCLUSION

In this paper, we studied the practical and realistic continual learning setting for session-based recommendation tasks. To prevent the recommender forgetting user preferences learned before, we propose ADER by replaying carefully chosen exemplars from previous cycles and an adaptive distillation loss. Experiment results on two widely used datasets empirically demonstrate the promising performance of ADER. Our work may inspire researchers working from similar continual learning perspective for building more robust and scalable recommenders.
REFERENCES

[1] Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. 2018. Memory aware synapses: Learning what (not) to forget. In Proceedings of the European Conference on Computer Vision (ECCV). 139–154.

[2] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. 2017. Hindsight experience replay. In Advances in neural information processing systems. 5048–5058.

[3] Francisco M Castro, Manuel J Marin-Jimenez, Nicolás Guil, Cordelia Schmid, and Karteek Alahari. 2018. End-to-end incremental learning. In Proceedings of the European Conference on Computer Vision (ECCV). 233–248.

[4] Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K Dokania, Philip HS Torr, and Marc’Aurelio Ranzato. 2019. Continual Learning with Tiny Episodic Memories. arXiv preprint arXiv:1902.10486 (2019).

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018).

[6] Robert M French. 1999. Catastrophic forgetting in connectionist networks. Trends in Cognitive Sciences (1999), 128–135.

[7] Florent Garcia, Christos Dimitrakakis, and Boi Faltings. 2013. Personalized news recommendation with context trees. In RecSys. ACM, 105–112.

[8] Diksha Garg, Priyanka Gupta, Pankaj Malhotra, Lovekesh Vig, and Gautam Shroff. 2019. Sequence and time aware neighborhood for session-based recommendations: Stan. In SIGIR. 1069–1072.

[9] Lei Guo, Hongzhi Yin, Qinyong Wang, Tong Chen, Alexander Zhou, and Nguyen Quoc Viet Hung. 2019. Streaming session-based recommendation. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. 1569–1577.

[10] Balázs Hidasi and Alexandros Karatzoglou. 2018. Recurrent neural networks with top-k gains for session-based recommendations. In Proceedings of the 27th ACM International Conference on Information and Knowledge Management. 843–852.

[11] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2016. Session-based recommendations with recurrent neural networks. In ICLR.

[12] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531 (2015).

[13] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. arXiv preprint arXiv:1207.0580 (2012).

[14] Saitui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahiu Lin. 2019. Learning a unified classifier incrementally via rebalancing. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 831–839.

[15] Dietmar Jannach and Malte Ludewig. 2017. When recurrent neural networks meet the neighborhood for session-based recommendation. In RecSys. ACM, 306–310.

[16] Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In 2018 IEEE International Conference on Data Mining (ICDM). IEEE, 197–206.

[17] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. 2017. Overcoming catastrophic forgetting in neural networks. Proceedings of the national academy of sciences 114, 13 (2017), 3521–3526.

[18] Jing Li, Pengjie Ren, Zhumin Chen, Zhaocun Ren, Tao Lian, and Jun Ma. 2017. Neural attentive session-based recommendation. In Proceedings of the 2017 ACM on Conference on Information and Knowledge Management. 1419–1428.

[19] Zhizhong Li and Derek Hoiem. 2017. Learning without forgetting. IEEE transactions on pattern analysis and machine intelligence 40, 12 (2017), 2935–2947.

[20] Qiao Liu, Yifu Zeng, Refuoe Mokhosi, and Haibin Zhang. 2018. STAMP: short-term attention/memory priority model for session-based recommendation. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. 1831–1839.

[21] Malte Ludewig and Dietmar Jannach. 2018. Evaluation of session-based recommendation algorithms. User Modeling and User-Adapted Interaction 28, 4-5 (2018), 331–390.

[22] Davide Maltoni and Vincenzo Lomonaco. 2019. Continuous learning in single-incremental-task scenarios. Neural Networks 116 (2019), 56–73.

[23] Michael McCloskey and Neal J Cohen. 1989. Catastrophic interference in connectionist networks: The sequential learning problem. In Psychology of learning and motivation. Vol. 24. Elsevier, 109–165.

[24] Fei Mi and Boi Faltings. 2016. Adaptive Sequential Recommendation Using Context Trees. In IJCAI. 4018–4019.

[25] Fei Mi and Boi Faltings. 2017. Adaptive sequential recommendation for discussion forums on MOOCs. In Proceedings of the 10th international conference on educational data mining.

[26] Fei Mi and Boi Faltings. 2018. Context Tree for Adaptive Session-based Recommendation. arXiv preprint arXiv:1806.03733 (2018).

[27] Fei Mi and Boi Faltings. 2020. Memory Augmented Neural Model for Incremental Session-based Recommendation. arXiv preprint arXiv:2005.01573 (2020).

[28] Seyed-Iman Mirzadeh, Mehrdad Farajtabar, and Hassan Ghasemzadeh. 2020. Dropout as an Implicit Gating Mechanism For Continual Learning. arXiv preprint arXiv:2004.11545 (2020).

[29] Sylvester-Alviso Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. 2017. icarl: Incremental classifier and representation learning. In Proceedings of the IEEE conference on Computer Vision and Pattern Recognition. 2001–2010.
[30] Pengjie Ren, Zhumin Chen, Jing Li, Zhaochun Ren, Jun Ma, and Maarten de Rijke. 2019. RepeatNet: A Repeat Aware Neural Recommendation Machine for Session-based Recommendation. In AAAI.

[31] Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. 2016. Progressive neural networks. arXiv preprint arXiv:1606.04671 (2016).

[32] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. 2016. Prioritized experience replay. (2016).

[33] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019. BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer. In CIKM. 1441–1450.

[34] Yong Kiam Tan, Xinxing Xu, and Yong Liu. 2016. Improved recurrent neural networks for session-based recommendations. In Proceedings of the 1st Workshop on Deep Learning for Recommender Systems. 17–22.

[35] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In Advances in neural information processing systems. 5998–6008.

[36] Max Welling. 2009. Herding dynamical weights to learn. In Proceedings of the 26th Annual International Conference on Machine Learning. 1121–1128.

[37] Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. 2019. Large scale incremental learning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 374–382.

[38] Friedemann Zenke, Ben Poole, and Surya Ganguli. 2017. Continual learning through synaptic intelligence. In Proceedings of the 34th International Conference on Machine Learning. JMLR. org, 3987–3995.

[39] Bowen Zhao, Xi Xiao, Guojun Gan, Bin Zhang, and Shutao Xia. 2019. Maintaining Discrimination and Fairness in Class Incremental Learning. arXiv preprint arXiv:1911.07053 (2019).