Towards Automatic Discovering of Deep Hybrid Network Architecture for Sequential Recommendation

Mingyue Cheng[1], Zhiding Liu[1,†], Qi Liu[1,*], Shenyang Ge[2], Enhong Chen[1]
[1]Anhui Province Key Lab of Big Data Analysis and Application, School of Data Science &
School of Computer Science and Technology, University of Science and Technology of China [2] xiaobing.ai
{mycheng,doge}@mail.ustc.edu.cn,{qiliuql,cheneh}@ustc.edu.cn,neuronblack@outlook.com

ABSTRACT

Recent years have witnessed great success in deep learning-based sequential recommendation (SR), which can provide more timely and accurate recommendations. One of the most effective deep SR architectures is to stack high-performance residual blocks, e.g., prevalent self-attentive and convolutional operations, for capturing long- and short-range dependence of sequential behaviors. By carefully revisiting previous models, we observe: 1) simple architecture modification of gating each residual connection can help us train deeper SR models and yield significant improvements; 2) compared with self-attention mechanism, stacking of convolution layers also can cover each item of the whole sequential behaviors and achieve competitive or even superior performance.

Guided by these findings, it is meaningful to design a deeper hybrid SR model to ensemble the capacity of both self-attentive and convolutional architectures for SR tasks. In this work, we aim to achieve this goal in the automatic algorithm sense, and propose NASR, an efficient neural architecture search (NAS) method that can automatically select the architecture operation on each layer. Specifically, we firstly design a Table-like search space, involving both self-attentive and convolutional-based SR architectures in a flexible manner. In the search phase, we leverage weight-sharing supernets to encode the entire search space, and further propose to factorize the whole supernet into blocks to ensure the potential candidate SR architectures can be fully trained. Owning to lacking supervisions, we train each block-wise supernet with a self-supervised contrastive optimization scheme, in which the training signals are constructed by conducting data augmentation on original sequential behaviors. The empirical studies show that the discovered deep hybrid network architectures can exhibit substantial improvements over compared baselines, indicating the practicality of searching deep hybrid network architectures on SR tasks. Notably, we show the discovered architecture also enjoys good generalizability and transferability among different datasets.

[†]Equal contribution, [*]Qi Liu is corresponding author,
Our codes are available at https://github.com/Mingyue-Cheng/NASR.

WWW '22, April 25–29, 2022, Virtual Event, Lyon, France

ACM ISBN 978-1-4503-9096-5/22/04…$15.00
https://doi.org/10.1145/3485447.3512066

CCS CONCEPTS

• **Information systems** → **Recommender systems**.

KEYWORDS

Sequential recommedation, Neural architecture search

ACM Reference Format:
Mingyue Cheng, Zhiding Liu, Qi Liu, Shenyang Ge, Enhong Chen. 2022. Towards Automatic Discovering of Deep Hybrid Network Architecture for Sequential Recommendation. In *Proceedings of the Web Conference 2022 (WWW '22), April 25–29, 2022, Virtual Event, Lyon, France.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3485447.3512066

1 INTRODUCTION

Recommender systems (RS) [29, 45, 48] have become ever important in the increasingly overloaded age of the digital economy where users have to make choices from massive and rapidly increasing contents, products, and services. The key of personalized RS is to model users' preferences on items based on their past interactions, known as collaborative filtering (CF) [7, 30]. In real scenarios, the user interaction records often exist in a form of chronological sequence. In such scenes, sequential recommendation (SR) models, focusing on characterizing users' evolving interest from ordered behaviors, could provide more timely and accurate recommendations compared with traditional CF methods [18, 22, 47].

Massive efforts have been paid to improve the accuracy of SR tasks [42]. Early works on SRS usually capture lower-order sequential dependencies from user historical interactions by using Markov chains (MC) and matrix factorization [37]. Following this line, a series of extensions for higher-order MC have been proposed [18, 22]. Among them, leveraging deep neural networks to capture dynamic user interests from chronological user behaviors has become the core of SR research and achieved substantial improvements [34]. The most prevalent deep SR network architecture usually contains three major modules: two embedding layers for input & output items, and several middle hidden layers [50]. The core part of existing deep SR models is to capture long- and short-term user interests with sequential behaviors via utilizing diversified high-performance architectures [40, 47].

Originally, recurrent architectures and their variants [18, 25, 26] are first utilized to capture infinitely long user interaction sessions. While effective, such methods mainly summarize all previous actions depending on the hidden states that cannot be fully used for parallel computation within a sequence [40, 47]. Thus their speed is limited in both training and evaluating in large-scale recommendation scenarios. Later, numerous advanced efforts [38, 51] have been

proposed to capture evolving user interests by abandoning recurrent structures and achieve competitive or superior performance by stacking many hidden residual blocks. Nowadays, there exist two types of prevalent configurations in capturing user preference from sequential behaviors: 1) stacking multiple convolutional layers to cover all interacted items from the sequential behaviors; 2) leveraging the global reception field of self-attentive architectures to capture dynamic user interests.

Despite their effectiveness, we empirically find that these existing deep SR models (i.e., self-attentive and convolutional networks in [22, 47]), modified with slight linear residual enhancement (see Section 2.2), can be stacked with deeper hidden layers to achieve more promising results. Such surprising results are inconsistent with the sense of most current works [17], holding that only several hidden layers are enough. With rigorous controlled experiments (on the same experimental setting and evaluation protocol), we also find convolutional architectures could achieve competitive results compared with self-attentive alternatives. Even more surprising, convolutional-based SR models can exhibit self-attentive methods in certain scenarios. We believe it is reasonable since the convolutional networks can cover all the items within the whole session by stacking convolutions. Such results remind us that the convolutional architectures retain great strength in modeling ordered user behaviors just as recent very prevalent self-attentive structures. As suggested by both such empirical findings and prior works [12] - using hybrids of self-attentive and convolutional architectures can outperform pure ones, one may ask whether can we design a very deep SR model, which can merge the strengths of both self-attentive and convolutional capacities to boost capturing dynamic user interests. Following this intuition, we explore a new but more challenging research question - designing a deep hybrid SR model in the automatic algorithm sense instead of relying on the human experts' manual efforts.

To achieve the above goals, we propose NASR, an efficient network architecture search method for automatically designing SR models to make accurate recommendations. Specifically, we first propose a Table-like search space, containing both diversified self-attentive and convolutional-based SR architectures in a flexible manner. In the search phase, we apply the weight-sharing techniques, allowing candidate architectures to share the weight parameters from the supernet, for saving the evaluation cost. We further modularize the whole supernet into blocks to ensure the potential candidate SR architecture is fully trained. Due to lacking training signals in optimizing each block supernet, we employ a self-supervised learning scheme, in which we construct the supervisions via performing data augmentation on original sequential behaviors. Thanks to the block-wise search, we can evaluate all candidates within each block. Toward a more fair and efficient estimation, we also propose to select sub-models from Table-like space block-by-block with a newly designed unsupervised evaluation metric to measure candidates. We evaluate the effectiveness and accuracy of NASR on two benchmark datasets. The results show that the searched deep hybrid SR models can achieve superior performance than that achieved by compared competitive baselines.

The contribution of this paper are listed as:

- By performing a comprehensive empirical evaluation over existing self-attentive and convolutional SR models, we make two important observations, specifically, we find that 1) SR models equipped with simple linear residual enhancement can be stacked with deeper hidden layers to achieve their optimal performance; 2) training convolutional SR models are competitive and even promising alternatives in certain scenarios, compared to self-attentive structures.
- We propose NASR, a neural architecture search method to automatically discover more expressive deep hybrid models for SR tasks. In the NASR, we propose a Table-like search space, which can organize the candidate operations in a flexible manner. Furthermore, we apply several key techniques, including weight-sharing supernet, block-wise search, and self-supervised training scheme, to reduce the search cost and improve the search accuracy.
- We empirically compare NASR and baseline architectures by following the same setting and demonstrate substantial improvements, indicating the practicality of searching deep hybrid SR models. We also find that the searched models can be transferrable between datasets, indicating the good generalizability and transferability of searched architectures.

2 PRELIMINARIES

We first formalize the sequential recommendation (SR) and introduce the notations. Then, we briefly introduce the deep SR architecture and show a linear residual modification. After that, we perform insightful empirical studies. The novel contribution of this part is to highlight the necessity to search hybrid networks for SR tasks.

2.1 Definitions and Notations

Suppose that there exist $|\mathcal{U}|$ unique users in the user set $\mathcal{U} = \{u_1, ..., u_{|\mathcal{U}|}\}$ and $|\mathcal{I}|$ unique items in the item set $\mathcal{I} = \{i_1, ..., i_{|\mathcal{I}|}\}$ in total. The goal of SR is to predict the next item $x^u_{t+1} \in \mathcal{I}$, that user might interact in time $t+1$. The key constraint is only to use the historical interaction records in chronological order $\mathcal{X} = [x^u_1, x^u_2, ..., x^u_t]$ for user $u \in \mathcal{U}$ in the current session. In this work, we omit the discussion of learning manner in previous sequential recommendation models [18, 38] and consistently adopt auto-regressive (a.k.a self-supervised learning in [46]) optimization to train all deep sequential recommendation models.

2.2 Revisiting Deep Sequential Recommenders

2.2.1 ***Basic Deep Sequential Recommendation Models.*** As is shown in Figure 1, modern SR model usually contains three major modules. Specifically, each input item x^u_i is first mapped into an embedding vector by embedding lookup operation, and then the sequential behavior input $\mathcal{X}^u$ is represented by an embedding matrix $\mathrm{E}^u = [e^u_1, ..., e^u_t]$. Next, the sequence embedding matrix E^u is fed into a stack of hidden layers, which are expected to capture long- and short-range dependence of sequential behaviors. Finally, the extracted hidden sequence representation is further fed to softmax classier layer [8, 47] for generating the user preference distribution over candidate items. Actually, the hidden layer lies the core part of capturing sequence dependence. A large body of research has provided efforts in leveraging diversified architectures (e.g., recurrent network in [18]) to conduct non-linear transformation. In the literature, training self-attentive and convolutional SR models [22, 40]

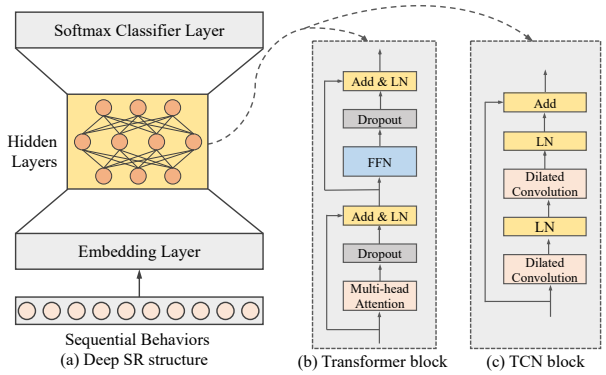

Figure 1: Illustration of sandwich-like deep SR models.

become the research hotspot due to: (1) these networks can leverage modern parallel computation, such as GPU/TPU, (2) these models can stack more hidden layers with residual block structure [16].

In fact, leveraging residual block to boost SR has become very prevalent in a series of recently proposed works [38, 39, 51]. The basic idea of residual learning is to stack multiple non-linear layers together as a residual block and then employ a skip connection scheme that passes the previous layers' transformation information to its posterior layer. Formally, we describe the expression of residual connection as $\mathcal{H}_L^u = \mathcal{H}_{L-1}^u + F_L((\mathcal{H}_{L-1}^u); \mathcal{W}_L)$, where $H_{L-1}^u \in \mathbb{R}^{t \times d}$ and $H_L^u \in \mathbb{R}^{t \times d}$ are the input and output of $L-$th residual block respectively while F_L denotes the residual mapping to be learned. Here, d is the embedding (hidden) size, and $\mathcal{W}_L$ indicates all the parameters in the $L-$th block.

2.2.2 ***Linear Residual Enhancement.*** Despite the effectiveness, by revisiting current deep SR models, we notice that most recommendation works [22, 38, 47] only stack limited layers depth in SR task. It is easy to understand since the neural work becomes more difficult to training with the increasing of depth layer [16]. To solve such a dilemma, taking inspirations from recently proposed work [2, 43], we add one trainable parameter λ_{L-1} to the vanilla residual connection for each layer. In this way, the residual connection can be re-formulated as

$$\mathcal{H}_L^u = \mathcal{H}_{L-1}^u + \lambda_{L-1} \cdot F_L((\mathcal{H}_{L-1}^u); \mathcal{W}_L), \tag{1}$$

where we initialize the λ_{L-1} with 0 at the beginning of training since such a simple modification can help current deep SR models obtain faster convergence and better accuracy.

In the following, we instantiate the linear residual modification over two well-known deep SR models including self-attentive based SASRec [22] and convolutional based NextItNet [47]. The former is composed of repeated Transformer structure for capturing long-range dependence, while the later relies on repeated temporal convolution neural network (TCN) architecture to obtain the global reception field to cover each item within the user session. The Transformer architecture modified with our linear residual modification can be expressed as

$$\mathcal{H}_{\mathrm{L}}^{\mathrm{u}} = \mathrm{LN}(\mathcal{H}_{\mathrm{L-1}}^{\mathrm{u}} + \lambda_{\mathrm{L-1}} \cdot \mathrm{sublayer}(\mathcal{H}_{\mathrm{L-1}}^{\mathrm{u}})), \tag{2}$$

where $\mathrm{sublayer} \in \{\mathrm{Multi-head\ Self-attention}, \mathrm{Position-wise\ Feed-forward\ Network(FFN)}\}$, where LN denotes the layer normalization [1], as is illustrated in the middle of Figure 1. NextItNet [47] is

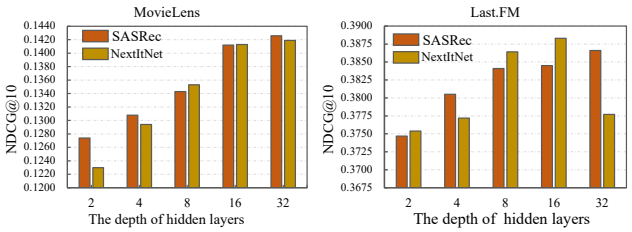

Figure 2: Illustration of the SR results w.r.t. the model depth of self-attentive [22] and convolutional [47] based methods.

composed a stack of temporal convolutional network (TCN) blocks, each two of dilated convolution are wrapped with a residual TCN block, as is shown the right of Figure 1. We formally represent the corresponding block as,

$$\mathcal{H}_{\mathrm{L}}^{\mathrm{u}} = \mathcal{H}_{\mathrm{L-1}}^{\mathrm{u}} + \lambda_{\mathrm{L-1}} \cdot \mathrm{LN}(\mathrm{sublayer}(\mathcal{H}_{\mathrm{L-1}}^{\mathrm{u}})) \tag{3}$$

where $\mathrm{sublayer} \in \{\mathrm{Dilated\ Convolution}\}$.

2.2.3 ***Empirical Explorations.*** In this subsection, we are interested in conducting empirical studies from two aspects: 1) validate the performance of residual-style SR architectures equipped with our slight linear residual modification; 2) fairly evaluate the representation capacity of self-attentive and convolutional-based architectures in the SR task. To keep the comparison as fair as possible, the experimental setting strictly obeys the hyper-parameter rules. With rigorous controlled experiments, we report all compared results by 1) adopting auto-regressive regression accompanied by cross-entropy loss as optimization objective [47], 2) generating the probability of the next item with a linear softmax classifier without leveraging shared item embedding layer [22], 3) training two SR models until achieving the best results. More implementation details w.r.t. hyper-parameter setting see section 4.1.5. We report all these experimental results in Figure 2.

By analyzing the experimental results, we can conclude that:

- By leveraging the linear residual enhancement, we can stack both the well-known Transformer and TCN architectures with deeper layers (e.g., up to 32 layers on MovieLens datasets). Notably, we find that SR can enjoy a very deep network and yield substantial gains (e.g., obtaining 15% improvements than shallow models on MovieLens datasets). Actually, some recent SR models only achieve a stacking of only several layers (e.g., SASRec [22], BERT4Rec [38], S^3Rec [51]).
- Convolutional-based architectures, e.g., TCN structures consisting of repeated dilated convolutions, actually are competitive and even achieve superior performance compared with prevalent self-attentive style structures (e.g., Transformer) for SR tasks. Such experimental phenomenon reflects the same potential ability for using convolutional recommenders for capturing user interests [41].

Combining recent hotpot in computer vision (CV) [12] - using hybrids of CNN and Transformer can outperform pure alternatives - and these insightful empirical findings claimed above, we are naturally motivated to think: whether can we ensemble the strengths of both convolutional and self-attentive architectures in a single model to obtain a more powerful SR method. Instead of relying on human experts' manual efforts, we are interested in designing such a deep hybrid SR network in the automatic algorithm sense.

In the next section, we would introduce our proposed NASR, an efficient neural architecture method for automatically discovering deep hybrid architectures for SR.

3 METHODOLOGY

In this section, we describe NASR, which performs neural architecture search to find a powerful sequential recommendation (SR) model with an ensemble of self-attentive and convolutional operations, simultaneously. To the best of our knowledge, few attempts are devoted to boosting recommendation performance, especially for SR tasks. Generally, designing the NAS algorithms requires the specification of three main steps, including *search space* (Section 3.1), *search algorithms* (Section 3.2), *model selection strategies* (Section 3.3). We first design a hybrid search space, which can organize high-performance diversified candidate operations well together to form a Table-like space. After that, we will respectively introduce several vital techniques (including weight-sharing, block-wisely search, self-supervised training) to reduce the whole search cost and improve the search accuracy.

3.1 Search Space

While designing the search space, we release two simple yet useful principles based on the insightful empirical studies in Section 2.2.3. On the one hand, we hope the search space can be scalable to the *depth* dimension so as to allow the SR model to enjoy the benefits of deeper architecture. On another hand, we hope the search space can ensemble the reception field of both self-attentive and convolutional architecture for further capturing user sequential dependence. Obeying this two basic rules, we design a novel Table-like search space as is shown in Figure 3, which flexibly takes the candidate operations of self-attentive and convolutional based architectures into account.

In this work, instead of designing more powerful candidate network operations, we are primarily interested in showing readers the expressiveness ability of hybrid SR models consisting of prevalent self-attentive and convolutional building blocks in current works. Specifically, we mainly choose the Transformer structure in SASRec [22, 38] and temporal convolutional network (TCN) architecture in NextItNet [4, 47]. It is worth mentioning that while a large body of self-attentive and convolutional SR architectures have been proposed recently, we leave it for future explorations.

3.2 Search Algorithms

Despite the large advance brought by NAS, automatically designing well-optimized deep hybrid architecture can be challenging, especially as the number of layer depth and choice operations increases. In this subsection, we would successively introduce three vital techniques adopted in our NASR methods to solve the tremendous search inefficiency and ineffectiveness issue caused by the depth of the SR model, including weight-sharing supernet, block-wise search, and self-supervised training.

Weight-sharing Supernet. Considering the depth of our hybrid search space in Figure 3, rating all candidates by training them from scratch, making it nearly impossible in real recommendation scenarios. Here, we follow the prevalent weight-sharing NAS solutions [33] to avoid repeated training the weight of candidate networks $\mathcal{W}$. Specifically, the entire search space $\mathcal{A}$ is encoded

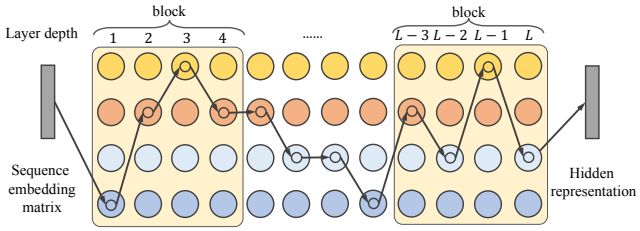

Figure 3: Illustration of our Table-like search space.

into a weight-sharing supernet $\mathcal{S}(\mathcal{W}, \mathcal{A})$, in which all candidate architectures inherit the shared network weights $\mathcal{W}^*$ over the same network operations. In this way, the search cost can be largely reduced since these candidate networks avoid repeated training by inheriting the weights of the supernet. The search phrase can be formulated as: $\alpha^* = \min_{\forall \alpha \in \mathcal{A}} \mathcal{L}_{val}(\mathcal{W}^*, \alpha; \mathcal{X})$. Here, $\mathcal{L}_{val}$ denotes the evaluation metric. The optimization process of shared weights can be expressed as: $\mathcal{W}^* = \min_{\mathcal{W}} \mathcal{L}_{train}(\mathcal{W}, \mathcal{A}; \mathcal{X})$. The loss $\mathcal{L}_{train}$ denotes the loss function while training supernets.

Block-wisely Search. Although the weight-sharing strategies largely improve the evaluation efficiency, the used Table-like search space accompanied by layer-level granularity grow exponentially with the increase of network depth. Badly, a series of recent research works [23] show that such a huge search space has become one inevitable obstacles to the evaluation of weight-sharing candidate networks. Inspired by recent proposed methods [24, 49], we adopt the block-wise search techniques to further factorize the whole search space. Specifically, we first uniformly divide the supernet $\mathcal{A}$ into $|k|$ blocks $(\mathcal{A}_1, \mathcal{A}_2, ..., \mathcal{A}_{|k|})$ according to the depth layer dimension. Mathematically, we formalize the supernet $\mathcal{S}(\mathcal{W}, \mathcal{A}) = \{\mathcal{S}_k(\mathcal{W}_k, \mathcal{A}_k)\}$, denoting a supernet is composed of $|k|$ blocks. Here, $\mathcal{W} = \{\mathcal{W}_k\}$ indicates its network weights of supernet of $k-$th block and $\mathcal{A} = \{\mathcal{A}_k\}$ denotes corresponding architectures.

Self-supervised Training. To separately train each supernet block, lacking supervision for each hidden block becomes a vital technical barrier. We notice existing works [23, 44] mainly leverage knowledge distillation (KD) framework [8, 19] to provide block-wise supervisions, in which a pre-trained teacher supernet generates the data for both training and searching. Considering the goal of obtaining hybrid CNN-Transformer recommendation architecture, it is a big barrier to finding a well-optimal hybrid SR model to perform as a teacher network. Also, recently proposed work [24] claims that such KD scheme is likely to be highly correlated with the teacher architecture and the student network inevitably inherits the architecture bias from the teacher model. In the supernet training of NASR methods, we thus abandon such KD optimization scheme and focus on performing self-supervised learning with constructing the supervision signals from original sequential behaviors [9].

Inspired by recent works [5, 9, 14, 15], we find that different compositions of network architectures can generate user representations with various views. Hence, we propose to perform contrastive learning [5, 15] to train each block-wise supernet, in which minimizing the distance of different views of the same sequence records is the main optimization objective. To achieve this goal, we adopt the simple dual Siamese (weight-sharing) network, containing student and target supernet (respectively denoted as $\mathcal{S}(\mathcal{W}, \mathcal{A})$ and

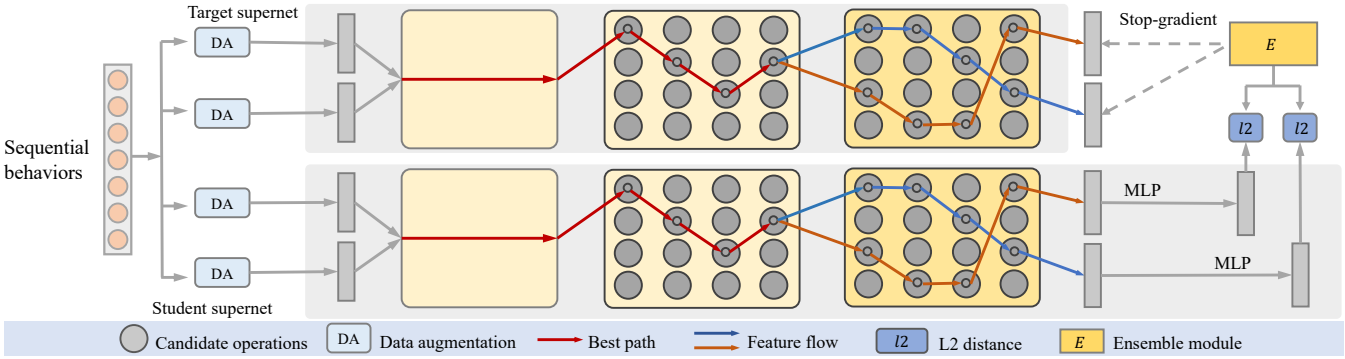

Figure 4: Illustration of supernet training with self-supervised contrastive learning.

$\mathcal{T}(\mathcal{W}^{\mathcal{T}}, \mathcal{A}^{\mathcal{T}})$), as the self-supervised contrastive training framework, analogous to the simple yet powerful self-supervised works widely adopted in computer vision [6]. To generate the self-supervised training signals, we adopt three simple but effective data augmentation (DA) strategies (destroying the partial user sequential behaviors by masking, cropping, reordering, detail descriptions will be provided in Section 4.1.2) for contrastive training by following previous works [9, 11]. As for more effective DA strategies, we leave them for future works.

To improve the stabilization of training supernet, we allow the student supernet to imitate the *ensemble target representations* during each training iteration, which indicates the ensemble of $|p|$ sampled sub-networks, indicated by $\{\alpha_p\} \subset \mathcal{A}_k, p = 1, 2, ..., |p|$. To be more specific, the optimization of each sampled sub-network of the student supernet is to predict the probability ensemble of $|p|$ sampled sub-models in the target supernet:

$$\widehat{\mathcal{T}}_k(\{\alpha_p; \{\mathcal{X}_p^{'}\}\}) = \frac{1}{|p|} \sum_{p=1}^{|p|} \mathcal{T}_k(\mathcal{W}, \alpha_p; \mathcal{X}_p^{'}), \tag{4}$$

where $\{\mathcal{X}_p^{'}\}$ are generated for each sampled sub-network of the Siamese supernets. Next, we formalize the optimization process of the block-wisely Siamese supernet as follows:

$$\begin{aligned} &\alpha^* = \{\alpha\}^* = \arg\min_{\mathrm{W_k}} \sum_{\mathrm{p=1}}^{\mathrm{|p|}} \mathcal{L}_{\mathrm{train}}(\mathrm{W_k^*}, \{\alpha_\mathrm{p}\}; \mathcal{X}) \\ &\text{where } \mathcal{L}_{\mathrm{train}}(\{\mathcal{W}_\mathrm{k}\}\,\{\alpha_\mathrm{p}\}; \mathcal{X}) = \\ &\| \mathcal{S}_\mathrm{k}(\mathcal{W}_\mathrm{k}, \alpha_\mathrm{p}; \mathcal{X}_\mathrm{p}^{'}) - \widehat{\mathcal{T}}_\mathrm{k}(\mathcal{W}_\mathrm{k}, \{\alpha_\mathrm{p}; \{\mathcal{X}_\mathrm{p}^{'}\}\}) \| \,. \end{aligned} \tag{5}$$

in which $\| \cdot \|$ indicates l_2 distance.

3.3 Model Selection

In the following, we introduce the efficient process of model selection from two parts: (1) greedily search the candidates block-by-block; (2) design an unsupervised evaluation metric for estimating candidate architectures.

Greedily Search Strategies. After training the supernet with self-supervised learning, the candidate architecture can be ranked and searched based on the weight of the supernets. Considering the typical supernet contain tremendous sub-models (e.g., hybrid search space with 32 layers depth and 4 candidate architectures contains about 4^{32} sub-networks), which stop us from evaluating all of them. To solve the large size of sub-networks, greedy search strategy usually is utilized to progressively shrink the search space by selecting the top-performing *partial* models layer by layer. Since that we adopt the block-wise search techniques, we here propose to estimate the performance of all sub-models according to their block-wise performance and traverse all the sub-models to select the top-performing partial architectures block by block. Thanks to our block-wise search, traversal evaluation of all the candidate architectures are affordable.

Unsupervised Evaluation Metric. Since the process of training supernet without leveraging supervision signals, in the following, we aim to design a fair and effective unsupervised rating metric to measure the representation ability of candidate architectures. In our proposed NASR, we aim to imitate the behavior target network in each block. Thus, we estimate the learning ability of a student sub-model by our designed measure in each block. Specifically, we first ensemble the population $\{\alpha_p\}$ as the *common targets* to provide fair rating for each architecture α_p. Then, one pair of views for each validation user sequence behaviors $\mathcal{X}$ are generated and fixed. We describe the ensemble of the architecture population as

$$\widehat{\mathcal{S}}_k(\{\alpha_p\}; \mathcal{X}') = \frac{1}{|p|} \sum_{p=1}^{|p|} \mathcal{T}_k(\mathcal{W}, \alpha_p; \mathcal{X}_p^{'}). \tag{6}$$

In this case, the architecture population $\{\alpha_p\}$ is expanded to the whole block-wise search space $\mathcal{A}_k$, and the whole searching process is finished in a single step:

$$\begin{aligned} &\alpha^* = \min_{\forall \alpha \subset \mathcal{A}} \sum_{\mathrm{p=1}}^{\mathrm{|p|}} \mathcal{L}_{\mathrm{val}}(\alpha; \mathcal{X}_\mathrm{k}), \\ &\text{where } \mathcal{L}_{\mathrm{val}}(\alpha; \mathcal{X}) = \| \mathcal{S}_\mathrm{k}(\alpha; \mathcal{X}^{'}) - \widehat{\mathcal{S}_\mathrm{k}}(\mathcal{A}_\mathrm{k}; \{\mathcal{X}^{'}\}) \| \,. \end{aligned} \tag{7}$$

4 EXPERIMENTS

4.1 Experimental Setup

4.1.1 ***Datasets Description.*** To verify the effectiveness of our proposed NASR, we conduct extensive experiments on two widely used datasets: MovieLens[1] and Last.FM[2]. In the data processing step, to alleviate the impact of cold users and items, we perform the basic pre-processing by filtering out interactions with less than 5 users and users with less than 10 items. We summarize the statics of datasets in Table 1.

[1] http://files.grouplens.org/datasets/movielens/
[2] http://www.dtic.upf.edu/ocelma/MusicRecommendationDataset/lastfm-1K.html

Table 1: Statistics of the datasets.

DATA	# actions	# sequences	# items	length
MovieLens	25,417,546	876,162	23,515	30
Last.FM	10,699,640	534,982	199,013	20

- **MovieLens**: We generate the interaction sequence of a user according to the chronological order. We define the maximum length of the interaction sequence as 30. Sequences shorter than 30 will be padded with zero at the beginning of the sequence to reach 30.
- **Last.FM**: We define the session length as 20, and extract 20 successive items as the input sequence. This is done by sliding a window of both size and stride of 20 over the whole data. We ignore sessions in which the time span between the last two items is longer than 2 hours.

4.1.2 ***Search Space and Supernet Setup.*** We perform our search on the designed Table-like search space as demonstrated in Figure 3. The supernet consists of $L = 16$ layers. We search among Transformer architecture in [22] with multi-head number of $\{4, 8\}$ and TCN structure [47] with dilated factor of $\{[1, 2], [1, 4]\}$, four operations in total. We random sample 4 paths to obtain corresponding sub-networks. It should be noted that we assume that it is consistent for all hidden embedding sizes among the 4 blocks. We separately train each block in the supernet for 2 epochs under the guidance of self-supervised signals. The self-supervised learning hyper-parameter setting follows those used in SimSiam [6]. Not loss of generality, we employ three simply data augmentation strategies to generate the self-supervised training signals including: 1) randomly **masking** a certain percent of the whole sequence behaviors with [MASK] token; 2) randomly **clipping** fragments of the entire sequence; 3) randomly **reordering** the partial sub-sequence of the whole sequence. All of these data augmentation strategies destroy 25% of the whole sequence behaviors.

4.1.3 ***Compared Methods.*** To evaluate the effectiveness of NASR, we prepare several baselines to compare with our design. To keep the results of all recommendations as fair as possible, we strictly obey the hyper-parameter control rules to train these SR models as is illustrated in Section 2.2.3. In addition, all residual-style SR models are further enhanced by leveraging our newly introduced linear residual learning enhancement.

- GRU4Rec [18] is a well-known session-based recommendation method by using the hidden state to memory users' dynamic interest to the evolving of sequential behaviors.
- SASRec [22] is a self-attentive-based recommender model, which uses Transformer structure to capture user preference to the evolving of interactions in chronological order.
- NextItNet [47] employs the residual block of dilated convolution to increase the reception field for modeling long-range dependence over user interactions with chronological order.
- Random search: randomly sample candidate architecture operations to form deep hybrid network architectures for SR tasks, where the reported results are obtained with the mean value of repeating experiments three times.
- NASR denotes the searched SR models with automatic designing methods, i.e., NASR. Note that all searched SR network architectures are **retrained from scratch** on the original sequence behavior datasets.

Here, we respectively use SASRec-1 and SASRec-2 to denote the self-attentive-based SR model with the multi-head numbers of 4 and 8. Similarly, we use NextItNet-1 and NextItNet-2 to represent the convolutional-based SR methods with different settings of dilated factors including $[1, 2]$ and $[1, 4]$.

4.1.4 ***Evaluation Protocols of Sequential Recommendation.*** To quantify the performance of compared methods, we evaluate all models by employing two popular top-K metrics, namely NDCG@K (Normalized Discounted Cumulative Gain), and Recall@K. K is set to 10 and 20 for comparison. Following previous works [22, 38], we apply the *leave-one-out* strategy for evaluation, i.e., only considering the last item in each interaction for the SR task. It should be noted that since we have only a test set for each user, Recall@M is equivalent to HR@K and is proportional to Precision@K. In addition, we evaluate each method on the whole item set rather than the sampled metrics.

4.1.5 ***Hyper-parameter Settings.*** We implement all models by PyTorch. All models were trained on the device of a single NVIDIA V100 GPU. For comparison purposes, the embedding dimension d is set as 64 for all models. The hidden dimensions are set the same value as embedding dimension. The learning rate of all models is set to 0.001 on all datasets. The batch size is set to 256 on MovieLens, while it is set to 64 on Last.FM. For all models, we use Adam as optimizer. Other hyper-parameters of baseline methods are empirically tuned according to performance on validation sets by following original works [18, 22, 47]. For all compared results, the default depth layer is 16. In search phase, we use AdamW optimizer with 0.001 initial learning rate and exponential decay schedule, weight decay is set to 0.

4.2 Experimental Results Analysis

4.2.1 ***Recommendation Performance Analysis.*** We report the overall results in Table 2. By carefully analyzing the experimental results, we find GRU4Rec yields much worse results than Transformer and TCN based SR models. The results indicate self-attentive and convolutional SR models via stacking deeper hidden layers can be more expressive than shallow GRU4Rec architecture in capturing sequential patterns in user behaviors. Meanwhile, training a deeper model from scratch requires much more computational costs than a shallower model. Second, SR models discovered by the NASR consistently perform better than pure self-attentive and convolutional ones. Such results demonstrate the powerful capacity of deep hybrid SR models in capturing dynamic user interests via the global reception field of attention mechanism and local view of convolutional architecture. We believe that this is explainable since the effectiveness of using hybrids of using self-attentive and convolutional architecture have been fully evidenced in other domains [10, 12]. It worst noting that naively using hybrids of Transformer and TCN seems to perform a bit worse than baselines. To some extent, this indicates the importance of employing NAS techniques to heuristically search for effective SR architectures.

4.2.2 ***Searcher Efficiency Analysis.*** Beyond accuracy, we also record the time cost for training searcher and model selection. In our experiments, NASR was trained on with batch of 256 on MovieLens and 64 on Last.FM equipped with the device of GPU (1

Table 2: The experimental results on two benchmark datasets, where the best performance methods are denoted in bold.

Model	MovieLens					Last.FM				
	NDCG@20	Recall@20	NDCG@10	Recall@10	step time	NDCG@20	Recall@20	NDCG@10	Recall@10	step time
GRU4Rec	0.1587	0.3344	0.1347	0.2390	126ms	0.3931	0.4772	0.3848	0.4452	75ms
Nextitnet-1	0.1610	0.3342	0.1375	0.2411	338ms	0.3963	0.4807	0.3883	0.4490	188ms
Nextitnet-2	0.1653	0.3421	0.1413	0.2471	329ms	0.3848	0.4703	0.3764	0.4368	198ms
SASRec-1	0.1648	0.3392	0.1412	0.2453	247ms	0.3916	0.4836	0.3845	0.4493	183ms
SASRec-2	0.1665	0.3428	0.1428	0.2485	257ms	0.3943	0.4886	0.3855	0.4538	182ms
Random search	0.1605	0.3395	0.1361	0.2426	313ms	0.3932	0.4885	0.3837	0.4508	200ms
NASR	**0.1793**	**0.3648**	**0.1548**	**0.2676**	280ms	**0.4068**	**0.4965**	**0.3978**	**0.4611**	191ms

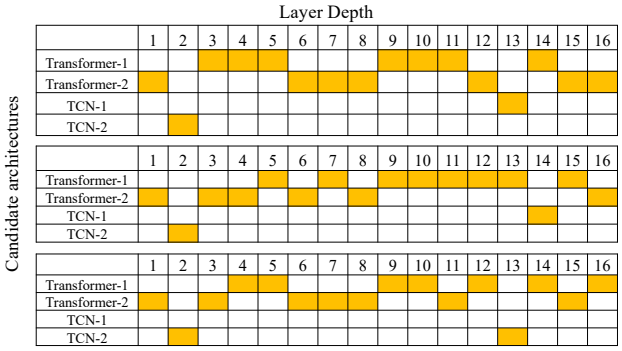

Figure 5: Visualization of architectures discovered by NASR.

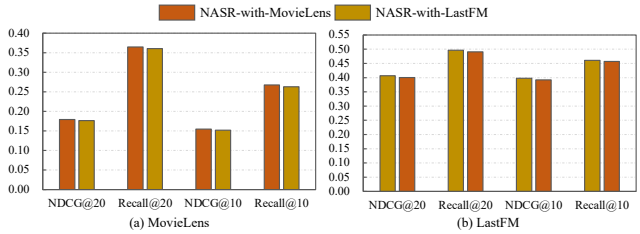

Figure 6: Transferability of architectures found by NASR.

NVIDIA V100). As a result, discovering the SR models with NASR spends about less than 3 hours (9, 128 seconds) over the MovieLens dataset and 5 hours (15, 429 seconds) for Last.FM dataset. It should be noted that the spend time contains the time of both training supernet and model selection stage. Compare to previous works - training even one architecture costs a long time (e.g., more than 10 GPU days) [52], it can be found such search cost is affordable in real recommendation applications. This is mainly because 1) we adopt weight-sharing NAS techniques, which allow the sampled sub-models to inherit the weight parameters from the supernet; 2) we employ block-wise search methods to divide the deep supernet into sub-blocks and perform a greedy search strategy to rank all candidate architectures.

4.2.3 ***Architecture Visualization Analysis.*** We visualize three of the network architectures found by NASR in Figure 5, in which we repeat the experiments of NASR three times with the making data augmentation strategies over MovieLens datasets. It shows these three searched architecture ensembles multiple different network operations and prefer to adopt a higher percent of Transformer architectures while transforming sequence embedding matrix to hidden representations. Such phenomenon largely reflects the self-attentive-based Transformer architecture is necessary for sequence dependences in the next item recommendation. Such results are consistent with the empirical study in Section 2.2. Surprisingly, we notice the searched network architecture tends to adopt TCN structure in the beginning and ending position of the whole model to the dimension of depth layer. Such results reflect a similar empirical phenomenon in recently proposed work [36].

4.2.4 ***Architecture Transferability Analysis.*** Since the human-crafted architectures are not designed for a specific dataset, we explore the transferability of the searched architectures across different datasets. Here, select the best architectures discovered by NASR on MovieLens, and apply them on Last.FM, and vice versa. We report the compared experimental results in Figure 6. As shown, while a bit worse, the architecture searched from one SR dataset could still achieve superior performance compared with baselines while applying on other datasets. Such experimental phenomenon largely reflects that the Table-like space could be well suitable for capturing sequence dependence. We guess a possible reason is that the two benchmark datasets share common characteristics of user interest extraction. We believe such interesting experimental results can inspire more work to be proposed for discovering adaptive architecture in certain scenes, e.g., cold-start recommendation [46].

4.2.5 ***Study on Data Augmentation Strategies.*** In the NASR, we train the supernet with a self-supervised contrastive learning manner, in which the training signals are constructed from the original user behavior information. According to previously claimed views [5, 9], DA strategies are crucial to the contrastive optimization manner. Here, we perform extensive ablation studies by controlling the composition of three different DA strategies, including *masking*, *clipping*, and *reordering* strategies. For limit space, we omit the results using Last.FM and only report the results in Figure 7. We can observe that masking strategies perform a bit better than another two types of DA manners, but there exists a minor difference between the three DA strategies. Despite the effectiveness of these DA strategies, it is still necessary to develop a more effective and simple DA manner [13] for further boosting and simplifying the training process of supernet.

4.2.6 ***Hyper-parameter Sensitivity Analysis.*** We also study the hyper-parameters how to affect the performance of searched models, including the number of training epochs for optimizing supernet, the number of sampling paths, and the number of building blocks involved within each divided block. Here, the default setting denotes remaining each block supernet with layers of 4

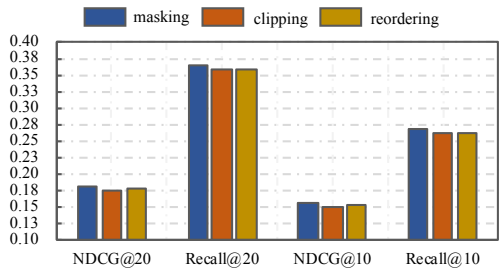

Figure 7: The ablation study w.r.t. data augmentation (DA) strategies in training supernets of our NASR.

building blocks, sample 4 paths on the supernet, and training each block supernet with 1 epoch. Presenting all results on all datasets is redundant and space unacceptable, we mainly report results of hyper-parameter influence on MovieLens datasets in Figure 8. Interestingly, the proposed NAS methods are slightly sensitive to these several parameters. Such results largely reflect the robustness of our NASR methods in discovering the SR models for releasing accurate recommendations. Such experimental results reflect that it is more important to rank these sampled candidate architectures rather than measure the absolute performance of candidate models.

5 RELATED WORK

Deep Sequential Recommenders. A large body of research provides efforts in deep sequential recommendation techniques since their effectiveness in capturing complex sequential patterns from user behaviors [9, 18, 22]. Originally, recurrent architectures gated recurrent units (GRU) [18, 25, 26] are natural choices in modeling sequential behaviors. While effective, such methods mainly summarize all previous actions depending on employing the hidden state, in which the modern parallel processing resources cannot be fully leveraged in practice. Thus their speed is limited in both training and evaluation in large-scale recommendation scenes. Numerous advanced efforts for sequential recommendation with abandoning RNN architectures have been proposed. A series of works have been proposed by training convolutional neural networks (CNN) [40, 47] or self-attentive networks [22, 25, 38] for SR tasks, in which these methods allow parallelization over each entity within a sequence. These newly proposed convolutional or self-attentive-based models can be able to achieve comparable or even superior performance to the popular RNNs structures since more hidden layers can be stacked by the residual block architectures.

Neural Architecture Search. Recently, neural architecture search (NAS) [31, 52] is hoped to replace the effort of human experts in network design by leveraging machines. Early works adopt reinforcement learning (RL) [52] or evolutionary algorithms (EA) to sample architecture and get its performance through training it from scratch. However, this type of NAS is computationally expensive. To enhance the efficiency of NAS, recently proposed ENAS [33] encode the entire search space as a weight-sharing supernet. After training the supernet, sub-models are sampled and evaluated with the weights inherited from the supernet. To solve the large size of the weight-sharing space, some works [23, 44] proposed to factorize the supernet into independently optimized blocks. Meanwhile, unsupervised NAS performs architecture search without access to

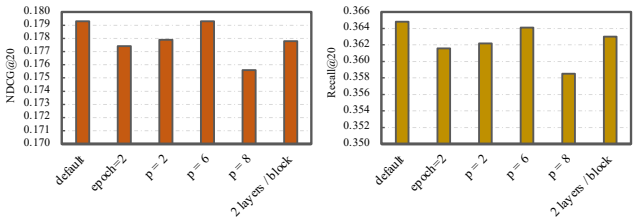

Figure 8: Hyper-parameter sensitivity study on MovieLens.

any supervision signals. Representative work like [28] introduced unsupervised pretext tasks to train supernets.

Self-supervised Learning. Self-supervised learning [5, 20] recently becomes prevalent since the parameters optimized by the self-supervised loss can be easily utilized to benefit other tasks. Most mainstream approaches fall into one of two classes: generative or discriminative. For generative-based methods, in NLP area, the language model becomes a popular self-supervised objective that learns to predict the next word given the previous contextual information [32, 35]. Also, the cloze prediction task is widely adopted in [11]. Though these fine-grained prediction tasks have achieved promising results, some researchers hold that its computation is expensive and might not be necessary for representation learning [5]. Recently, discriminative-based methods, such as contrastive learning, have been well studied in visual representation. The core idea of contrastive learning is to pull the positive example pairs closer and push the negative pairs apart. Simple and effective methods have been developed using Siamese network [3].

Although some automated based methods have been proposed before, most of these research focus on click-through rate (CTR) tasks by: (1) automatically searching the optimal hyper-parameter (e.g., embedding size in [21]); (2) conducting automated feature interaction learning (a.e., feature learning in [27])). To the best of our knowledge, it still remains under-explored in terms of designing effective deep hybrids of self-attentive and convolutional architectures for SR tasks.

6 CONCLUSIONS

In this work, we perform a pilot study of automatically designing deep architectures for SR, and proposed NASR, a very efficient neural architecture search (NAS) method for discovering deep hybrid SR models to provide more expressive ability. In the proposed NASR, we first designed a Table-like search space, containing self-attentive and convolutional architectures, for finding more powerful SR models. After that, NASR could efficiently discover top-performing optimal SR architectures by applying weight-sharing supernet, block-wise search, and self-supervised learning. We conducted extensive experiments on two benchmark datasets to demonstrate both the strength of SR models discovered by our NASR, indicating the practicality of conducting NAS for SR tasks. We hoped our NASR can inspire more NAS efforts and deep hybrid networks of self-attentive and convolutional architecture to be proposed for recommendation.

ACKNOWLEDGMENTS

This research was partially supported by grants from the National Natural Science Foundation of China (Grants No. 61922073 and U20A20229). Qi Liu gratefully acknowledges the support of the

Youth Innovation Promotion Association of CAS (No. 2014299) and WK5290000002.

REFERENCES

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450* (2016).

[2] Thomas Bachlechner, Bodhisattwa Prasad Majumder, Huanru Henry Mao, Garrison W Cottrell, and Julian McAuley. 2020. Rezero is all you need: Fast convergence at large depth. *arXiv preprint arXiv:2003.04887* (2020).

[3] Philip Bachman, R Devon Hjelm, and William Buchwalter. 2019. Learning representations by maximizing mutual information across views. In *Advances in Neural Information Processing Systems*. 15535–15545.

[4] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. 2018. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271* (2018).

[5] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*. PMLR, 1597–1607.

[6] Xinlei Chen and Kaiming He. 2021. Exploring simple siamese representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 15750–15758.

[7] Mingyue Cheng, Runlong Yu, Qi Liu, Vincent W Zheng, Hongke Zhao, Hefu Zhang, and Enhong Chen. 2019. Alpha-Beta Sampling for Pairwise Ranking in One-Class Collaborative Filtering. In *2019 IEEE International Conference on Data Mining (ICDM)*. IEEE, 1000–1005.

[8] Mingyue Cheng, Fajie Yuan, Qi Liu, Shenyang Ge, Zhi Li, Runlong Yu, Defu Lian, Senchao Yuan, and Enhong Chen. 2021. Learning recommender systems with implicit feedback via soft target enhancement. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 575–584.

[9] Mingyue Cheng, Fajie Yuan, Qi Liu, Xin Xin, and Enhong Chen. 2021. Learning Transferable User Representations with Sequential Behaviors via Contrastive Pre-training. In *2021 IEEE International Conference on Data Mining (ICDM)*. IEEE, 51–60.

[10] Stéphane d'Ascoli, Hugo Touvron, Matthew Leavitt, Ari Morcos, Giulio Biroli, and Levent Sagun. 2021. Convit: Improving vision transformers with soft convolutional inductive biases. *arXiv preprint arXiv:2103.10697* (2021).

[11] 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).

[12] Alexey et al Dosovitskiy. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929* (2020).

[13] Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. SimCSE: Simple Contrastive Learning of Sentence Embeddings. *arXiv preprint arXiv:2104.08821* (2021).

[14] Jean-Bastien Grill and Florian et al Strub. 2020. Bootstrap your own latent: A new approach to self-supervised learning. *arXiv preprint arXiv:2006.07733* (2020).

[15] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. 2020. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 9729–9738.

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

[17] Xiangnan He and Tat-Seng Chua. 2017. Neural factorization machines for sparse predictive analytics. In *Proceedings of the 40th International ACM SIGIR*. 355–364.

[18] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2015. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939* (2015).

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

[20] Min Hou, Chang Xu, Yang Liu, Weiqing Liu, Jiang Bian, Le Wu, Zhi Li, Enhong Chen, and Tie-Yan Liu. 2021. Stock Trend Prediction with Multi-granularity Data: A Contrastive Learning Approach with Adaptive Fusion. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 700–709.

[21] Manas R Joglekar, Cong Li, Mei Chen, Taibai Xu, Xiaoming Wang, Jay K Adams, Pranav Khaitan, Jiahui Liu, and Quoc V Le. 2020. Neural input search for large scale recommendation models. In *ACM SIGKDD*. 2387–2397.

[22] Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *2018 IEEE ICDM*. IEEE, 197–206.

[23] Changlin Li, Jiefeng Peng, Liuchun Yuan, Guangrun Wang, Xiaodan Liang, Liang Lin, and Xiaojun Chang. 2020. Block-wisely supervised neural architecture search with knowledge distillation. In *CVPR*. 1989–1998.

[24] Changlin Li, Tao Tang, and Guangrun et al Wang. 2021. Bossnas: Exploring hybrid cnn-transformers with block-wisely self-supervised neural architecture search. *arXiv preprint arXiv:2103.12424* (2021).

[25] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. 2017. Neural attentive session-based recommendation. In *ACM CIKM*. 1419–1428.

[26] Zhi Li, Hongke Zhao, Qi Liu, Zhenya Huang, Tao Mei, and Enhong Chen. 2018. Learning from history and present: Next-item recommendation via discriminatively exploiting user behaviors. In *ACM SIGKDD*. 1734–1743.

[27] Bin Liu and Chenxu et al Zhu. 2020. Autofis: Automatic feature interaction selection in factorization models for click-through rate prediction. In *ACM SIGKDD*. 2636–2645.

[28] Chenxi Liu, Piotr Dollár, Kaiming He, Ross Girshick, Alan Yuille, and Saining Xie. 2020. Are labels necessary for neural architecture search?. In *European Conference on Computer Vision*. Springer, 798–813.

[29] Qi Liu, Enhong Chen, Hui Xiong, Chris HQ Ding, and Jian Chen. 2011. Enhancing collaborative filtering by user interest expansion via personalized ranking. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 42, 1 (2011), 218–233.

[30] Qi Liu, Yong Ge, Zhongmou Li, Enhong Chen, and Hui Xiong. 2011. Personalized travel package recommendation. In *2011 IEEE 11th international conference on data mining*. IEEE, 407–416.

[31] Renqian Luo, Fei Tian, Tao Qin, Enhong Chen, and Tie-Yan Liu. 2018. Neural architecture optimization. *NeurIPS* 31 (2018).

[32] Matthew E. Peters and Mark Neumann et al. 2018. Deep contextualized word representations. In *NAACL-HLT*.

[33] Hieu Pham, Melody Guan, Barret Zoph, Quoc Le, and Jeff Dean. 2018. Efficient neural architecture search via parameters sharing. In *International Conference on Machine Learning*. PMLR, 4095–4104.

[34] Massimo Quadrana and Paolo et al Cremonesi. 2018. Sequence-aware recommender systems. *ACM Computing Surveys (CSUR)* 51, 4 (2018), 1–36.

[35] A. Radford. 2018. Improving Language Understanding by Generative Pre-Training.

[36] Maithra Raghu, Thomas Unterthiner, Simon Kornblith, Chiyuan Zhang, and Alexey Dosovitskiy. 2021. Do Vision Transformers See Like Convolutional Neural Networks? *arXiv preprint arXiv:2108.08810* (2021).

[37] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing personalized markov chains for next-basket recommendation. In *WWW*. 811–820.

[38] 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 *ACM CIKM*. 1441–1450.

[39] Yang Sun, Fajie Yuan, Min Yang, Guoao Wei, Zhou Zhao, and Duo Liu. 2020. A generic network compression framework for sequential recommender systems. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1299–1308.

[40] Jiaxi Tang and Ke Wang. 2018. Personalized top-n sequential recommendation via convolutional sequence embedding. In *ACM WSDM*. 565–573.

[41] Mariya Toneva, Alessandro Sordoni, Remi Tachet des Combes, Adam Trischler, Yoshua Bengio, and Geoffrey J Gordon. 2018. An empirical study of example forgetting during deep neural network learning. *ICLR* (2018).

[42] Shoujin Wang, Longbing Cao, Yan Wang, Quan Z Sheng, Mehmet A Orgun, and Defu Lian. 2021. A survey on session-based recommender systems. *ACM Computing Surveys (CSUR)* 54, 7 (2021), 1–38.

[43] Lechao Xiao and Yasaman et al Bahri. 2018. Dynamical isometry and a mean field theory of cnns: How to train 10,000-layer vanilla convolutional neural networks. In *ICML*. PMLR, 5393–5402.

[44] Jin Xu, Xu Tan, Renqian Luo, Kaitao Song, Jian Li, Tao Qin, and Tie-Yan Liu. 2021. NAS-BERT: Task-Agnostic and Adaptive-Size BERT Compression with Neural Architecture Search. *arXiv preprint arXiv:2105.14444* (2021).

[45] Runlong Yu, Qi Liu, Yuyang Ye, Mingyue Cheng, Enhong Chen, and Jianhui Ma. 2020. Collaborative list-and-pairwise filtering from implicit feedback. *IEEE Transactions on Knowledge and Data Engineering* (2020).

[46] Fajie Yuan, Xiangnan He, Alexandros Karatzoglou, and Liguang Zhang. 2020. Parameter-efficient transfer from sequential behaviors for user modeling and recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1469–1478.

[47] Fajie Yuan, Alexandros Karatzoglou, Ioannis Arapakis, Joemon M Jose, and Xiangnan He. 2019. A simple convolutional generative network for next item recommendation. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*. 582–590.

[48] Kai Zhang, Hao Qian, Qing Cui, Qi Liu, Longfei Li, Jun Zhou, Jianhui Ma, and Enhong Chen. 2021. Multi-interactive attention network for fine-grained feature learning in ctr prediction. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*. 984–992.

[49] Man Zhang, Yong Zhou, Jiaqi Zhao, Shixiong Xia, Jiaqi Wang, and Zizheng Huang. 2021. Semi-supervised blockwisely architecture search for efficient lightweight generative adversarial network. *Pattern Recognition* 112 (2021), 107794.

[50] Xiangyu Zhao, Chong Wang, Ming Chen, Xudong Zheng, Xiaobing Liu, and Jiliang Tang. 2020. Autoemb: Automated embedding dimensionality search in streaming recommendations. *arXiv preprint arXiv:2002.11252* (2020).

[51] Kun Zhou, Hui Wang, and Wayne Xin et al Zhao. 2020. S3-rec: Self-supervised learning for sequential recommendation with mutual information maximization. In *Proceedings of the 29th ACM CIKM*. 1893–1902.

[52] Barret Zoph and Quoc V Le. 2016. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578* (2016).