

Forward Once for All: Structural Parameterized Adaptation for Efficient Cloud-coordinated On-device Recommendation

Kairui Fu

Zhejiang University

Hangzhou, China

fukairui.fkr@zju.edu.cn

Zheqi Lv

Zhejiang University

Hangzhou, China

zheqilv@zju.edu.cn

Shengyu Zhang*

Zhejiang University

Hangzhou, China

Shanghai Institute for Advanced

Study of Zhejiang University

Shanghai, China

sy_zhang@zju.edu.cn

Fan Wu

Shanghai Jiao Tong University

Shanghai, China

fwu@cs.sjtu.edu.cn

Kun Kuang*

Zhejiang University

Hangzhou, China

kunkuang@zju.edu.cn

Abstract

In cloud-centric recommender system, regular data exchanges between user devices and cloud could potentially elevate bandwidth demands and privacy risks. On-device recommendation emerges as a viable solution by performing reranking locally to alleviate these concerns. Existing methods primarily focus on developing local adaptive parameters, while potentially neglecting the critical role of tailor-made model architecture. Insights from broader research domains suggest that varying data distributions might favor distinct architectures for better fitting. In addition, imposing a uniform model structure across heterogeneous devices may result in risking inefficacy on less capable devices or sub-optimal performance on those with sufficient capabilities. In response to these gaps, our paper introduces **Forward-OFA**, a novel approach for the dynamic construction of device-specific networks (**both structure and parameters**). Forward-OFA employs a structure controller to selectively determine whether each block needs to be assembled for a given device. However, during the training of the structure controller, these assembled heterogeneous structures are jointly optimized, where the co-adaption among blocks might encounter gradient conflicts. To mitigate this, Forward-OFA is designed to establish a structure-guided mapping of real-time behaviors to the parameters of assembled networks. Structure-related parameters and parallel components within the mapper prevent each part from receiving heterogeneous gradients from others, thus bypassing the gradient conflicts for coupled optimization. Besides, direct mapping enables Forward-OFA to achieve adaptation through **only one forward pass**, allowing for swift adaptation to changing interests and eliminating the requirement for on-device backpropagation.

*Shengyu Zhang and Kun Kuang are corresponding authors.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

KDD '25, August 3–7, 2025, Toronto, ON, Canada

© 2025 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ACM ISBN 979-8-4007-1245-6/25/08

<https://doi.org/10.1145/3690624.3709178>

Further sophisticated design protects user privacy and makes the consumption of additional modules on device negligible. Experiments on real-world datasets demonstrate the effectiveness and efficiency of Forward-OFA.

CCS Concepts

- Information systems → Recommender systems.

Keywords

Recommender System, On-device Recommendation

ACM Reference Format:

Kairui Fu, Zheqi Lv, Shengyu Zhang, Fan Wu, and Kun Kuang. 2025. Forward Once for All: Structural Parameterized Adaptation for Efficient Cloud-coordinated On-device Recommendation. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.1 (KDD '25), August 3–7, 2025, Toronto, ON, Canada*. ACM, New York, NY, USA, 12 pages. <https://doi.org/10.1145/3690624.3709178>

1 INTRODUCTION

Recent advances in deep learning have significantly enhanced the capabilities of recommender system, particularly in extracting user preferences from intricate sequential data[9, 47]. Traditional *on-cloud recommendation* methods primarily focus on enhancing the scalability and generalizability of models deployed on cloud. In such systems, user requests are processed on cloud, and recommendation lists are subsequently delivered. This process necessitates transmitting user data between remote devices and cloud, which can introduce substantial network overhead, especially in scenarios characterized by frequent user interactions[19, 43, 52]. Moreover, these user requests often include sensitive information, such as recent item interactions(e.g. *item id*), or even user private profiles(e.g. *ages, income, etc.*). Uploading these sensitive data to cloud may result in a potential leakage of user privacy.

Benefiting from the booming computational resources on mobile devices, recommender system is now deploying models directly to mobile devices to better serve users[11, 38]. This paradigm leverages the computational resources of devices to conduct real-time reranking[11], eliminating the need for data uploads to cloud servers for processing. Such an approach not only

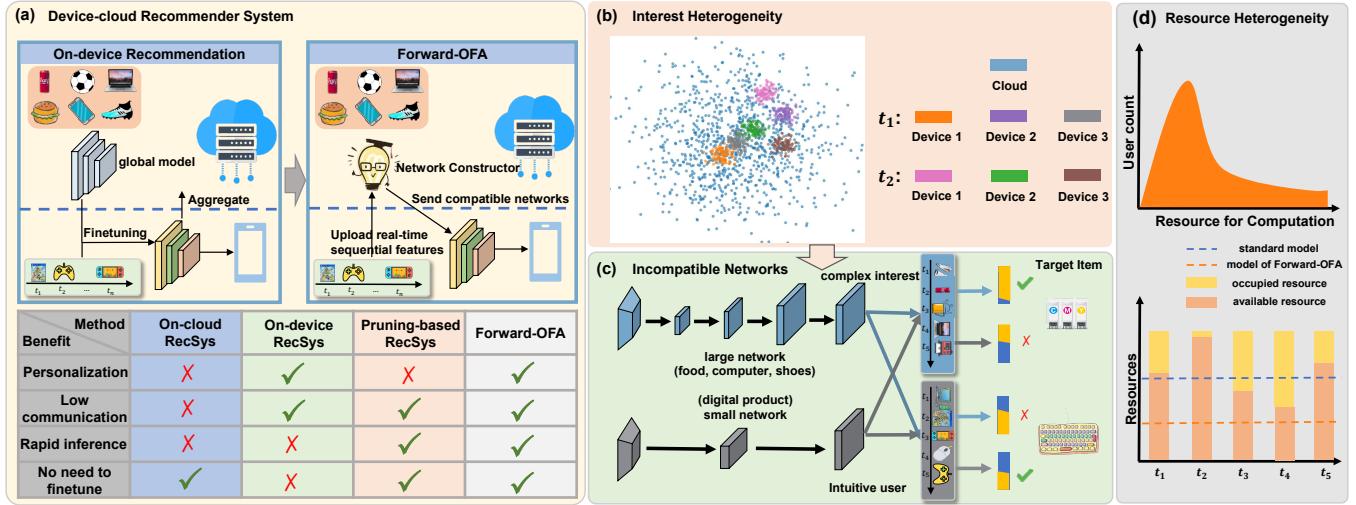


Figure 1: (a): Brief comparison between Forward-OFA and other methods used in on-device recommendation. (b): Each device has its own specific behaviors which change frequently while cloud has access to all the historical data of all devices. Distribution shifts among them and within each device make models trained with data on cloud degrade on some devices. (c): Large networks are conducive to exploring complex user interests, while simple networks are suitable enough for intuitive users. (d): Most users own mobile devices that don't have a lot of computing resources and the computing resources available for the current recommendation task will change in real time due to the presence of other apps.

mitigates network traffic burdens but also significantly enhances user privacy protections. Existing research in this area generally falls into two main categories: personalization-based and cost-aware mechanisms. The former focuses on tailoring device-specific parameters[24, 25, 27, 29, 42] to enhance the modeling of long-tailed users and items[30]. Contrastingly, cost-aware methods prioritize minimizing both the communication overhead involved in synchronizing local models from cloud and the computational cost of on-device inference due to the requirement for continuous adaptation. For instance, some studies[26, 31, 41] explore optimal moments for updating models from cloud to prevent unnecessary data transmission. Other works[6, 10, 33, 40] make an effort to remove redundant parameters for efficient transmission and inference.

While the aforementioned methods have made significant strides in on-device recommendation, their focus has predominantly been on network parameters, potentially overlooking the importance of network structures. Current research demonstrated that the uniqueness of network structures plays a vital role in data distribution modeling[48, 53]. This phenomenon is illustrated in Figure 1(c) of recommender system, where for some intuitive users(the gray one) smaller networks can already accurately model user interests. In contrast, larger networks are more conducive to modeling complex user interests(the blue one), as these users prefer to explore unknown interests. Figure 1(b) reveals the extensive variability of interest between devices themselves and clouds. Consistently deploying identical networks does not ensure satisfactory services for most devices with changing interests. When it comes to **resource heterogeneity** in Figure 1(d), universally applying larger networks will add a substantial burden for devices with limited resources and disrupt the functionality of other applications due to continuous resource usage. On the contrary, applying smaller models for all devices may not fully utilize the capabilities of more robust devices.

Consequently, the adaptive construction of networks has become critical for user-oriented recommender systems. Given its demanding and challenging characteristics, this problem raises four crucial research challenges: (i): How to design the corresponding local structures for each device **accurately**? (ii): How to **efficiently** build adaptive models for a given device to accommodate its local varying interests? (iii): How to protect user **privacy** from exposure to cloud during this process? (iv): How to avoid increasing much **burden** on devices due to the additional modules?

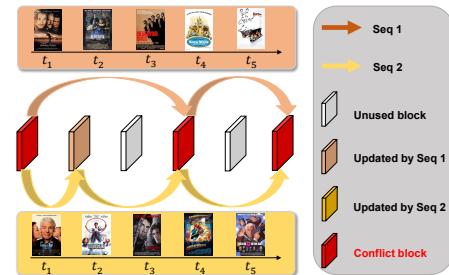


Figure 2: When backpropagation, conflict gradients from two sequences with different interests will prevent shared blocks(red blocks) from being updated correctly.

In light of this, we propose Forward-OFA(**Forward Once For All**), an approach to building both compatible and compact networks for a given device with just one forward pass and without necessitating further on-device backpropagation after training on cloud. Current sequential recommenders, which consist of stacked blocks[20, 34, 36, 44], guide us to consider learning inference paths containing one or more blocks as specific structures. In particular, we sampled heterogeneous paths from a discrete

distribution, which is parameterized by the output of the *structure controller*, to determine each block's existence. As the sample process is discrete and non-differentiable, we utilize the Gumbel Softmax Straight Through Estimator[18] to achieve joint optimization of the structure controller and the backbone(**Challenge i**). However, this feasible approach may encounter gradient conflicts in the co-adaptation among blocks for the coupled optimization of these heterogeneous assembled structures as depicted in Figure 2. To navigate this, Forward-OFA aims to learn structure-related parameters for each heterogeneous structure, bypassing the gradient conflicts for coupled optimization. In technique, we propose to learn a structure-guided mapping of real-time behaviors to the parameters of assembled networks, where *the structure-related gradients are directly propagated to the mapper whose components operate individually without heterogeneous gradients from each other*. This structural mapper also allows the construction of adaptive networks in a single forward pass, swiftly adapting to changing interests and eliminating the need for backpropagation on devices. An additional compact constraint ensures that local structures are more deterministic and resource-efficient, addressing **Challenge ii**. To prioritize user privacy, the low-level components of Forward-OFA, responsible for extracting user interest, are stored on device. They are designed to be resource-efficient, necessitating far fewer resources compared to traditional recommenders(**Challenge iv**). They only perform inference for adaptive networks when there's a substantial shift of on-device interests[31, 41]. Instead of raw user information, the extracted latent representation is uploaded to cloud for model construction, thereby protecting user privacy(**Challenge iii**).

Experiments on four real-world datasets underscore the effectiveness and efficiency of Forward-OFA. We conduct several in-depth analyses to uncover the workings of each component within Forward-OFA and the impact of its hyperparameters. Additional network visualization and case studies provide a clear analysis of how Forward-OFA constructs networks and the influence of different networks on diverse devices. In a nutshell, the key contributions of this paper are as follows:

- We early attempt to investigate the joint customization of both structure and parameters, analyzing the challenges of interest heterogeneity, network transmission, and on-device inference simultaneously.
- We introduce Forward-OFA to adaptively construct compatible networks that strategically remove unnecessary blocks while preserve those compatible ones for each device.
- We establish a structural mapping from real-time interactions on each device to its adaptive networks, ensuring both efficient and superior adaptation in a single forward pass.
- Extensive experiments demonstrate that Forward-OFA can be applied to various architectures and datasets for better recommendation and a smaller burden.

2 RELATED WORK

2.1 Sequential Recommendation

The field of sequential recommendation has witnessed significant advancements in recent years. Traditional sequential recommenders mainly rely on the Markov assumptions[15, 32], where current interactions only depend on the most recent interaction or interactions,

making it challenging to capture long-term dependencies. After the emergence of neural networks[45], neural sequential recommenders have started to utilize advanced techniques to enhance modeling capabilities and account for long-term dependencies[7]. For instance, GRU4Rec[16], a solution based on recurrent neural networks, can effectively capture temporal dependencies in sequences. Given multiple history items for each user, determining the importance of them to the target item becomes a critical issue. Attention-based mechanisms[3, 20, 34, 49] address it by assigning soft attention weights to items at different time steps, allowing for more flexible integration of information and more precise capture of user interests. Additionally, some CNN-based recommenders[36, 44] utilize a convolutional network to process user sequential information by treating the user's embeddings as a picture. However, all the above methods are essentially served on cloud, necessitating frequent message transmission and raising privacy concerns. Compared with them, Forward-OFA and other on-device recommendation methods, are motivated to handle these problems by performing recommendations locally owing to the increasing on-device resources. These model-agnostic frameworks reduce user response time and protect user privacy from exposure to cloud.

2.2 On-device Recommendation

Coinciding with the continuous upgrading of computing resources on mobile devices, on-device recommendation is widely used to alleviate network congestion and privacy issues. The majority of current research aims to exploit the diversity of models on devices to enrich personalized user services[5, 39]. DCCL[42] incorporates meta-patch into recommenders to avoid the expensive update of finetuning the entire model, the gradient will be passed to cloud server for aggregation. DUET[27] directly sends personalized parameters to device to better adapt to the user's local interest. PEEL[51], on the other hand, groups users on cloud and designs a compact and specific item embedding for each group, which not only enhances the personalization but also reduces the space required by embedding. Another method named DIET[10] aims at discovering the most compatible parameters for device-specific distribution within a random network to quickly adapt to local changes. Moreover, to keep pace with the evolving interests of users, recent studies[31, 41] have introduced methods to detect changes on the device and request new models from cloud[10, 27, 51] when necessary. Nevertheless, current methods mainly focus on parameter adaptation while potentially neglecting the potential of architectures, which is primarily emphasized in our paper.

3 METHOD

3.1 Preliminary

In recommender system, consider an item set $I = \{i_1, i_2, \dots, i_n\}$ and device(user) set $D = \{d_1, d_2, \dots, d_m\}$, where n and m represent the number of items and devices. Cloud has the capability to access the historical interactions of each device, denoted as $\mathbb{D} = \{(x_d^t, y_d^t)\}_{d=1, \dots, m, t=1, \dots, T^d}$, where T^d is the length of the historical interactions of device d and $x_d^t = \{y_d^1, \dots, y_d^{t-1}\}$ denotes all historical interactions before T^d . The objective of recommendation is to train a powerful model M using all the data on cloud, which

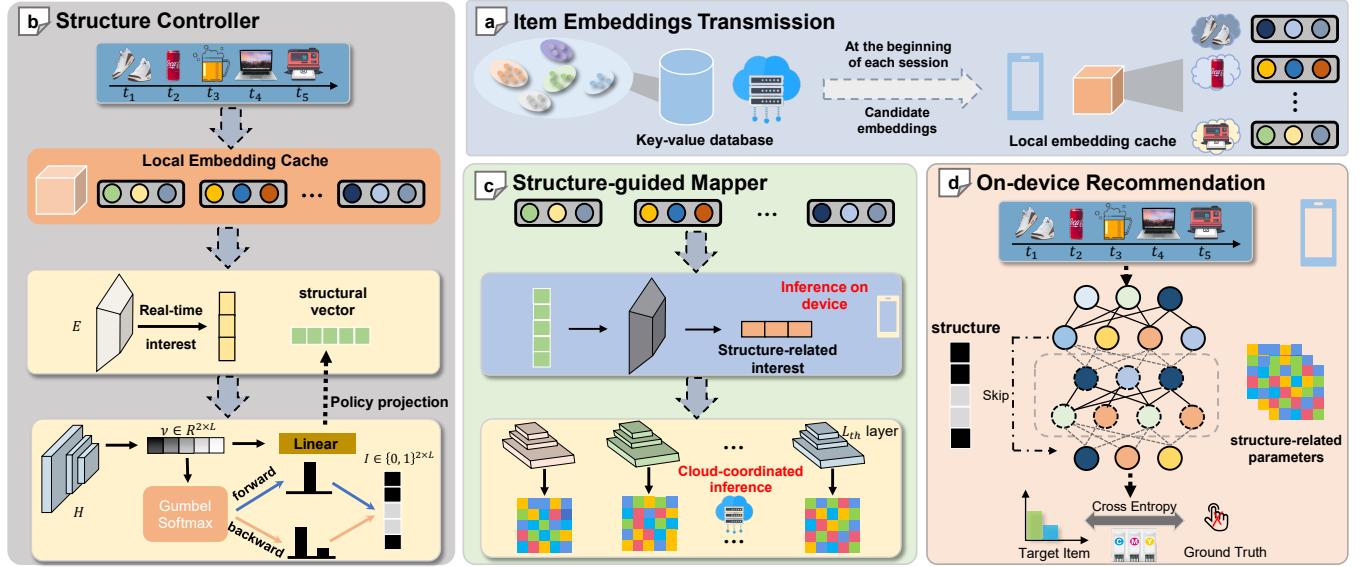


Figure 3: Illustrations of all components in Forward-OFA. (a): At the beginning of each session when interest changes dramatically, cloud sends candidate item embeddings to device and those embeddings will be cached for its recommendation in this session. (b): The structure controller consists of an extractor and a lightweight layer for searching the suitable path(distribution vector). The vector will be used in (c) to make structure-related parameters and alleviate the gradient conflict. (c): A mapper to assign personalized and structural parameters, aiming at removing gradient conflicts during training. (d): Each device does not necessarily own the whole network, but only a sub-model to acquire efficiency.

can be formulated as:

$$\min_{\theta} \sum_{d=1}^{d=m} \sum_{t=1}^{t=T^d} F(y_d^t | \{y_d^1, y_d^2, \dots, y_d^{t-1}\}; \theta), \quad (1)$$

s.t. $\min \mathbb{P} + \mathbb{F}$.

Here, \mathbb{P} and \mathbb{F} here refer to the number of parameters and FLOPs for inference of M . Besides, each device continue to generate real-time interaction data $X_d = \{y_d^1, \dots, y_d^{t-1}\}$, which may not have appeared on cloud before. Ideally, M should have good generalization ability and show adequate adaptation on these emerging data.

In on-device recommendation, we follow those widely-used paradigms[10, 26, 31, 41] from recent works. *At the start of each session, or when user interests change significantly, a series of candidate item embeddings and the updated model are simultaneously sent to devices for local reranking.* These embeddings are then cached on devices in Figure 3(b). As the recommended items for the user to click are chosen from these candidate items during this session, there is no requirement for any cloud-side involvement during predictions unless local interests change dramatically.

3.2 Structure Controller for Specific Networks

The use of stacked blocks has become prevalent in recent architectures [20, 44], significantly advancing the development of recommender systems. However, those numerous blocks also pose a challenge for on-device recommendation, particularly for resource-sensitive devices where a larger network leads to longer computation time. Additionally, complex structures can potentially result in overfitting, especially on devices with limited data. Hence, it is crucial to precisely allocate appropriate blocks for each device to ensure efficiency and reliability. A straight approach is to determine

where to stop and discard the latter blocks. However, we find that those latter layers might also be crucial for stable prediction as they are closer to the final classifier. In contrast, we tend to predict the presence of each block individually to reserve those later but important blocks. Specifically, we seek a distribution variable $v \in R^{L \times 2}$ for each network, where L is the number of blocks(residual block or transformer block) in a given network. With this variable, for each block F_l with $1 \leq l \leq L$, we assemble it if $v_{l,0} \geq v_{l,1}$ otherwise we would skip it. Formally,

$$h_{l+1} = F_l(h_l) * I_{l,0} + h_l * I_{l,1}, \quad (2)$$

where h_l and h_{l+1} are the input and output of block F_l . Variable $I \in \{0, 1\}^{L \times 2}$ in Equation 2 indicates whether each block F_l is supposed to be executed and its value is determined by v , where for $k = \{0, 1\}$:

$$I_{l,k} = \begin{cases} 1 & \text{if } \operatorname{argmax}\{v_{l,0}, v_{l,1}\} = k \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

Obviously the gradient of h_{l+1} cannot propagate back to vector v due to the non-differential function argmax in Equation 3. This inability to optimize through gradient backpropagation leads to the model consistently choosing a single path for parameter optimization, thus failing to achieve the intended purpose of heterogeneous structures. To overcome this, the Gumbel-Softmax straight-through estimator [18] is employed to address the non-differentiability of argmax . For random variable $G_{i,j} = -\log(-\log(U_{i,j}))$ sampled from a Gumbel distribution, where $U_{i,j}$ is sampled from a uniform distribution $U(0, 1)$, it is used to reparameterize v to obtain the

probability distribution v' :

$$v'_{i,j} = \frac{e^{(\log v_{i,j} + G_{i,j})/\tau}}{\sum_{k=0}^{k=L} e^{(\log v_{i,k} + G_{i,k})/\tau}}, \quad (4)$$

where τ serves as a temperature control for the smoothness of the sampling process. The closer τ is to 0, the more the reparameterized variables v' resemble the discrete distribution(i.e. one hot representation). As illustrated in Figure 3(b), the binary vector I can be sampled during the forward pass, and the gradient of the discrete variable can be approximated by computing the gradient of the continuous softmax relaxation for backpropagation:

$$I_{i,j} + v'_{i,j} - sg(v'_{i,j}), \quad (5)$$

where sg denotes the stop gradient operation, namely, $sg(x) = x$ and $\nabla sg(x) = 0$.

With Equation 5, the selected structures can evolve dynamically while trained with the parameters together. However, such a method requires training an individual distribution vector v for each device due to their diverse interests, which is impractical within recommender systems containing substantial and continuously expanding users. The expenses associated with retraining the distribution variable and network parameters for each device can be notably high. Additionally, the scarcity of local data for each device adds to the difficulty of the training process. In response to these difficulties, we propose the direct learning of a mapping from user interactions X_d to its special distribution vector v_d :

$$X_d = (y_d^{1'}, \dots, y_d^{t-1'}) \rightarrow v_d^{2 \times L}, \quad (6)$$

which constructs local preferred sub-structures adaptively based on real-time interests, thereby alleviating the complicated retraining.

As for the mapping process, considering the simplification within Forward-OFA, we utilize a single GRU E to extract user latent interests, followed by a subsequent fully connected layer denoted as H , projecting the interest vector onto the space of v :

$$\begin{aligned} \{\beta_{i,0}, \beta_{i,1}\}_{i=1,\dots,L} &= \{\log v_{i,0}, \log v_{i,1}\}_{i=0,\dots,L} \\ &= H(E(y_d^{1'}, \dots, y_d^{t-1'})). \end{aligned} \quad (7)$$

In that case, the structure controller can be trained along with the network parameters. This strategy can also be seen as a way to share knowledge among devices with similar interests[1, 21, 35]. In our framework, we place the structure controller on device as it only comprises a single GRU and another fully connected layer. Apart from this, these modules will be utilized to build adaptive sub-structures only when interest changes dramatically or at the beginning of each session[26, 31], which momentarily occupies only a few resources.

3.3 Structural Parameters to Alleviate Conflicts

Despite the promising approach we developed, there are still underlying challenges when training the structure controller and the original parameters simultaneously. Illustrated in Figure 2, different interactions tend to exhibit varying interests, which consequently requires different network configurations. The optimization of heterogeneous configurations thus introduces another critical problem, **the co-adaptation of them with the shared parameters** prevents each subnetwork achieves the optimal performance. In Figure 2, the unshared blocks between the two sequences(e.g., the orange

block owned by Seq 2 in the figure) result in the conflict on both the former block(block 1) and the latter blocks(blocks 4, 5) due to inconsistent gradient directions[2, 17, 50]: **i**) The latter blocks receive input from both block 1 and block 2, necessitating a considerable burden to fit both structures sharing the same parameters. **ii**) Regarding the former block, heterogeneous gradients from blocks 2 and 4 are transmitted to it, leading to a more pronounced impact. These discrepancies ultimately result in suboptimal networks.

In contrast to prior approaches utilizing few-shot learning [17, 50], which still encounter potential conflict issues and introduce significant overhead during training, we seek to **bypass the shared weights when updating the gradients by assigning structure-related parameters for the corresponding networks(conflict i)**. Similar to Section 3.2, given user real-time data X_d , Forward-OFA extracts the latent interests in advance and leverages the powerful *hypernetwork*[12] to dynamically generate parameters for each device. As the structures are also determined by device data X_d and its more potential knowledge than the structure variables $\{\beta_{i,0}, \beta_{i,1}\}_{i=0,\dots,L}$, in our experiment it is adopted to construct structure-related parameters to bypass the weight sharing problem. In technique, only a feature extractor E' and L fully connected layers $\{H'_k\}_{k=1,\dots,L}$ are employed to establish a mapping from user real-time data to the parameters of L blocks.

$$W = \{W_k\}_{k=1,\dots,L} = \{H'_k(E'(y_d^{1'}, \dots, y_d^{t-1'}))\}_{k=1,\dots,L}. \quad (8)$$

Under these circumstances, gradients would be directly propagated back to $\{E'_k\}_{k=1,\dots,L}$ and $\{H'_k\}_{k=1,\dots,L}$. Moreover, the L layers within the mapper are executed in parallel, thereby there won't be heterogeneous gradients from each other like in Figure 2, overcoming **conflict ii**.

Apart from this, as network structures can also have an impact on the compatible parameters, the output of the structure controller in Section 3.2 is used as the initial state of E' :

$$\begin{aligned} W &= \{W_k\}_{k=1,\dots,L} \\ &= \{H'_k(E'(y_d^{1'}, \dots, y_d^{t-1'}, \{\beta_{i,0}, \beta_{i,1}\}_{i=0,\dots,L}))\}_{k=1,\dots,L}. \end{aligned} \quad (9)$$

With structural parameterization, our method constructs specific networks (in both structure and parameters) for each device based on its local interests. Once a device desires to acquire new models, it autonomously builds network paths and sequential characteristics $h = E'(y_d^{1'}, \dots, y_d^{t-1'}, \{\beta_{i,0}, \beta_{i,1}\}_{i=0,\dots,L})$ itself, subsequently uploading them to cloud to acquire compatible and adaptive networks. The whole process is efficient as it only needs one forward inference on cloud without any backpropagation on device.

3.4 Compact Constraint and Loss Function

The structural parameterized networks primarily focus on recommendation performance but may overlook the resource constraints of mobile devices. For instance, two sub-structures might achieve comparable performance for a specific user. As the method above does not consider efficiency, it could mistakenly select sub-structures with longer inference time. Towards this end, we introduce another compact constraint \mathcal{L}_c with coefficient λ to encourage a lower probability of a block being executed:

$$\mathcal{L}_c = \sum_{k=1}^{k=L} -\log \alpha_{k,1}, \quad (10)$$

where

$$\alpha_{k,1} = \frac{e^{\beta_{k,1}}}{e^{\beta_{k,0}} + e^{\beta_{k,1}}}. \quad (11)$$

In addition to reducing model capacity and decreasing on-device response time, \mathcal{L}_c can assist Forward-OFA in identifying the most effective blocks(increasing their scores) and discarding less effective blocks, thus enhancing the performance of the framework.

Finally, the overall loss function \mathcal{L} is defined as:

$$\mathcal{L} = \mathcal{L}_{rec} + \lambda \mathcal{L}_c, \quad (12)$$

where \mathcal{L}_{rec} denotes the cross-entropy loss defined in Equation 1 between the prediction and the ground truth(next-clicked item). **A detailed pseudocode is used in the Appendix B to clarify the workflow of Forward-OFA better.**

4 EXPERIMENT

In this section, we conduct various experiments on two widely used sequential recommenders using four real-world datasets, aiming to address the following research questions:

- **RQ1:** How does Forward-OFA perform compared to other on-device recommendation methods in terms of resource consumption and recommendation?
- **RQ2:** How do the proposed modules of Forward-OFA and different hyperparameters affect the final performance?
- **RQ3:** Do the additional modules for adaptive network construction in Forward-OFA impose significant burdens on devices?
- **RQ4:** What impact does an adaptive network have on different user distributions?

4.1 Experimental Setup

4.1.1 Datasets. We adopt MovieLens-1M, MovieLens-10M[13], Amazon-Game, and Amazon-food[28], four real-world datasets in the experiment. The data preprocessing can be found in Appendix A.1.

4.1.2 Evaluation metrics. Through our experiments, we employ the widely adopted metrics[23, 46] NDCG and Hit in our experiments to assess the accuracy of recommendations provided by each method for devices. In terms of resource constraints, we primarily focus on the inference time on devices, calculated using FLOPs (floating point operations) of local networks during a single inference. Additionally, users are required to update the latest models from the cloud as needed to meet their requirements. Therefore, the size of network transmission, which is reflected by the number of bits in the recommenders(*param*), should also be taken into consideration. It is important to note that **higher NDCG and Hit mean better performance while smaller FLOPs and params indicate less dependence on resources**. Consistently, we utilize NDCG@10 and Hit@10 throughout the experiment for uniformity.

4.1.3 Baseline. To demonstrate the effectiveness of Forward-OFA, we adopt two widely used sequential recommender *SASRec*[20] and *NextItNet*[44] as base models, as they are made up of transformer and convolutional blocks, separately. We select 6 methods(*DeviceRec*, *Finetune*, *DUET*[27], *Rare Gem*[33], *STTD*[40] and *Gater*[6]) as baselines. **More descriptions can be found in the Appendix.**

4.1.4 Implementation Details. The *SASRec* and *NextItNet* we used in the experiment are 6 and 12 blocks, respectively. We follow consistent training epochs for all models, except for *Finetune* and *Rare Gem*, which require additional on-device tuning and sparse network retraining. The learning rate is set to 0.001 for *SASRec* and 0.002 for *NextItNet*, respectively. Specifically for *SASRec*, the coefficient for resource constraint λ is 0.005 for MovieLens-1M and Amazon-food, 0.001 for Amazon-game, and 0.01 for MovieLens-10M. For *NextItNet*, all datasets adopt 0.001 as the coefficient. In *SASRec*, the temperature τ is set to 5, 6, 5, and 7 for MovieLens-1M, MovieLens-10M, Amazon-game, and Amazon-food, respectively. In *NextItNet*, the temperature values are set to 12, 7, 6, and 6 for the corresponding datasets. To ensure a fair comparison, *we have adjusted the hyperparameters of compression-based methods and compared their performance with Forward-OFA under the same device constraints*.

4.2 Overall Analysis(RQ1)

To demonstrate the potential of Forward-OFA, we conduct experiments on four real-world benchmarks, comparing them to a range of baselines. The results are shown in Table 1. From these results, we can conclude that although deploying models on devices reduces the frequency of transmission when requesting, *DeviceRec* demands the same number of floating-point operations for inference, which is not practical for most devices with limited computing resources. Additionally, consistent networks for all devices may produce ambiguous recommendations for some users compared to those with adaptive networks, owing to the distinct distribution between them and cloud. It also requires enormous bandwidth to request the latest models from cloud in Figure 1(a).

STTD and *Rare Gem* aim to compress the model parameters to reduce the transmission cost during the cloud-coordinated model updating process. However, neither of them effectively mitigates the overhead caused by on-device inference. In fact, *STTD* even increases the computational cost due to the additional computation required for the semi-tensor product's matrix multiplication. Additionally, the random distribution of 0s in the sparse matrix obtained by *Rare Gem* still necessitates calculations once there are non-zero elements in each layer. *Gater* speeds up local inference but fails to achieve a satisfactory compression ratio without ensuring comparable performance. Furthermore, these methods still exhibit degradation on certain datasets, notably in MovieLens-10M or *NextItNet*, where *Rare Gem* reduced the NDCG and Hit by 16%.

Finetune leverages limited interactions on devices to adjust the parameters. However, as depicted in Table 1, it does not yield provide satisfying results compared to *DeviceRec*. We attribute this phenomenon to the overfitting problem for devices with few interactions and changing interest between training and test data. Another personalized method *DUET* achieves relatively large improvements in both NDCG and Hit. Personalized parameters for each user can effectively capture the latent interests of each user in the behavior sequence. Nevertheless, as mentioned in Section 1, the consistent model structures in *DUET* may not be effective enough to serve some users. The same inference overhead as the original model also limits its practical application.

Table 1: Overall Performance towards both recommendation and resource cost. We use bold font to denote the best-performing model. ↑ and ↓ denote that larger and smaller metrics lead to better performance, respectively.

Model	Method	Dataset															
		MovieLens-1M				MovieLens-10M				Amazon-Game				Amazon-Food			
		NDCG ↑	Hit ↑	FLOPs ↓	Param ↓	NDCG ↑	Hit ↑	FLOPs ↓	Param ↓	NDCG ↑	Hit ↑	FLOPs ↓	Param ↓	NDCG ↑	Hit ↑	FLOPs ↓	Param ↓
SASRec	DeviceRec	0.0969	0.1816	0.6244	3.9936	0.0718	0.1308	0.6244	3.9936	0.0359	0.0541	0.6244	3.9936	0.0526	0.0620	0.6244	3.9936
	Finetune	0.0939	0.1793	0.6244	3.9936	0.0681	0.1272	0.6244	3.9936	0.0343	0.0537	0.6244	3.9936	0.0523	0.0623	0.6244	3.9936
	DUET	0.1140	0.2050	0.6244	3.9936	0.0871	0.1511	0.6244	3.9936	0.0456	0.0676	0.6244	3.9936	0.0582	0.0701	0.6244	3.9936
	STTD	0.0974	0.1823	3.5735	3.7478	0.0718	0.1314	3.5735	3.7478	0.0371	0.0588	3.5735	3.7478	0.0541	0.0648	3.5735	3.7478
	Rare Gem	0.0970	0.1838	0.6244	3.9936	0.0741	0.1344	0.6244	3.9936	0.0373	0.0577	0.6244	3.7696	0.0533	0.0635	0.6244	3.6364
	Gater	0.0980	0.1821	0.5738	3.9936	0.0733	0.1398	0.5863	3.9936	0.0393	0.0600	0.5858	3.9936	0.0535	0.0644	0.6050	3.9936
	Forward-OFA	0.1182	0.2081	0.5699	3.6384	0.0905	0.1566	0.4795	3.0522	0.0487	0.0754	0.5720	3.7216	0.0590	0.0713	0.5667	3.6160
NextItNet	DeviceRec	0.0975	0.1846	1.5053	9.5846	0.0638	0.1166	1.5053	9.5846	0.0275	0.0440	1.5053	9.5846	0.0401	0.0467	1.5053	9.5846
	Finetune	0.0879	0.1715	1.5053	9.5846	0.0635	0.1165	1.5053	9.5846	0.0279	0.0422	1.5053	9.5846	0.0399	0.0467	1.5053	9.5846
	DUET	0.1175	0.2058	1.5053	9.5846	0.0815	0.1369	1.5053	9.5846	0.0448	0.0656	1.5053	9.5846	0.0515	0.0606	1.5053	9.5846
	STTD	0.0920	0.1725	4.1595	4.5957	0.0612	0.1108	4.1595	4.5957	0.0254	0.0421	4.1595	4.5957	0.0394	0.0454	4.1595	4.5957
	Rare Gem	0.1035	0.1921	1.5053	5.4784	0.0524	0.0998	1.5053	5.4784	0.0328	0.0496	1.5053	5.4784	0.0417	0.0477	1.5053	5.4784
	Gater	0.0981	0.1849	0.7141	3.8656	0.0613	0.1119	0.6107	7.7760	0.0304	0.0426	0.6924	8.8224	0.0385	0.0438	0.5718	7.2832
	Forward-OFA	0.1226	0.2140	0.6167	3.8656	0.0826	0.1429	0.4324	2.6816	0.0477	0.0708	0.6228	3.9072	0.0523	0.0613	0.3126	1.9104
	Improv.↑	21.97%	14.59%	× 1.0	× 1.0	26.03%	19.68%	× 1.30	× 1.31	35.61%	39.28%	× 1.09	12.29%	14.90%	× 1.10	× 1.10	

On the contrary, *Forward-OFA* constructs both local adaptive parameters and structures for each device and adopts a structural mapper to facilitate efficient adaptation. This approach not only achieves satisfying performance but also minimizes resource consumption, encompassing both transmission delay and inference. Notably, *Forward-OFA* outperforms other baselines on all four metrics, underscoring the effectiveness of our framework. In particular, *Forward-OFA* improved by nearly 20% on the MovieLens-1M and MovieLens-10M datasets, and by 30% on Amazon-food of *NextItNet*. It's worth noting that the *NextItNet* utilized in our experiments has more blocks than *SASRec* (6 vs 12), consequently resulting in relatively smaller FLOPs on *NextItNet* compared to those on *SASRec*.

4.3 In-depth Analysis(RQ2)

4.3.1 Block visualization. To gain a deeper understanding of the impact of our framework, we conduct visualizations of block distributions and experiments to explore the influence of each component on adaptive networks. In Figure 4, we plot the distribution of local adaptive networks and usage of blocks in the test set, which can provide insight into the functioning of our method. Based on the figure, we have the following observations: 1) Different blocks exhibit significant variations in the number of times they are selected. Some blocks are infrequently chosen, while others are deemed more crucial for most users, reflecting the diverse interests of users on the device. Most users tend to include the first and last blocks in their local networks as the former mainly extract low-level knowledge of user interactions and the latter is responsible for the final sequential representation in recommender system. 2) Not all users require all blocks within their local network. On the contrary, only a small fraction of users tend to retain the entire network, demonstrating the significance of building adaptive networks for each user.

4.3.2 Influence of adaptive networks. Table 2 presents the performance of the ablation models and the base recommender *DeviceRec*. *+controller* denotes the augmentation of the base model with a structural controller to predict the network structure for devices. From the table, we can conclude that in this way this approach does not always lead to satisfactory performance, despite the compact model it selects as explained in Figure 4. As introduced in Section 3.3, gradient conflicts result in incorrect updates within

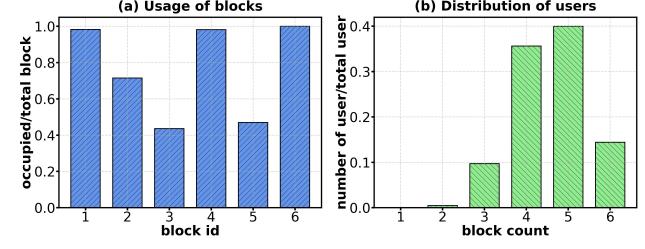


Figure 4: Analysis conducted on MovieLens-10M using a 6-layer SASRec. (a) The number of times each block is selected by the device in the test set. (b) Distribution of users with different numbers of blocks.

Table 2: Analysis of the structure controller and structural mapper involved in constructing adaptive networks.

Model	Method	Dataset					
		MovieLens-1M		MovieLens-10M		Amazon-Game	
		NDCG	Hit	NDCG	Hit	NDCG	Hit
SASRec	DeviceRec	0.0969	0.1816	0.0718	0.1308	0.0359	0.0541
	+controller	0.0946	0.1765	0.0703	0.1289	0.0369	0.0573
	+mapper	0.1142	0.2043	0.0868	0.1505	0.0452	0.0678
NextItNet	DeviceRec	0.0975	0.1846	0.0638	0.1166	0.0275	0.0425
	+controller	0.0973	0.1841	0.6114	0.1117	0.0316	0.0458
	+mapper	0.1193	0.2088	0.0816	0.1422	0.0461	0.0692

the shared blocks. With fewer blocks in *SASRec*, more users opt to share blocks, making it more pronounced in such scenarios. However, we can still observe some improvements on Amazon-Game where the conflict problem is relatively small and *+controller* consistently performs better than the base recommender. This finding demonstrates heterogeneous structures are indeed beneficial for a precise recommendation. On the other hand, *+mapper* bypasses the updates of shared blocks by associating parameters with user interests and the network structure, thus significantly enhancing the model's effectiveness.

4.3.3 Influence of sparsity constraint. We are interested in understanding how the sparsity constraint \mathcal{L}_c in Section 3.4 influences our method, therefore we adjust the coefficient and plot the result in Figure 5. First, when we increase the coefficient λ , the average

FLOPs and params in the test set will continue to decline as higher λ forces the probability of unnecessary blocks being selected to decrease. Furthermore, even without the sparsity constraint, the local models still require fewer resources than the original recommender in Table 1, thus proving the effectiveness of our structure controller. Second, an appropriate constraint could help our choice to be more deterministic and trust those more effective blocks, thereby enabling the learning of better subnetworks. Figures 5 (a) and (b) show that the sparsity constraint improves the recommendation, but it drops significantly when λ is so large that even useful blocks are ignored. Therefore, choosing the balance between real performance and resources becomes crucial for Forward-OFA.

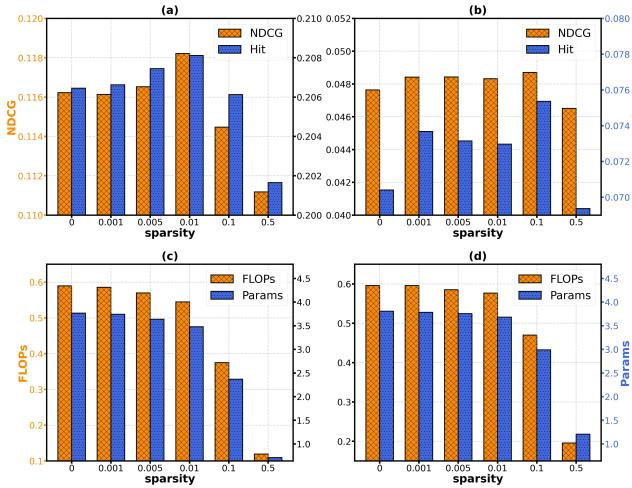


Figure 5: The influence of the coefficient λ of the sparsity constraint. The first and second columns are the experimental results on MovieLens and Amazon-game respectively. (a) and (b) report the results on NDCG and Hit while (c) and (d) represent the results on FLOPs and params.

4.3.4 Influence of structure-based parameters. Here we take an analysis to figure out how structure-related parameters benefit adaptive networks in our framework. Therefore we ablate some components in our method and compare their performance on three datasets. The ablated models are as follows:

- **w.o. structural vector** does not take the distribution vector(Equation 7) as an auxiliary input for the adaptive parameters. This allows the parameters to bypass conflicting gradient updates but ignore the structural information.
- **random block** means we only decides how many blocks will be used, but which blocks are used is random.
- **first k block** uses the first k blocks in the adaptive networks, where k is the number of blocks.
- **last k block** consistently uses the last k blocks in the adaptive networks.

The detailed performance is shown in Table 3, from which we have the following observations: 1) Parameters containing the structural information perform better than those without on all datasets, proving the necessity of the structure we generated as input. 2) Not using model-specified blocks will yield poor results, whether using the first few or the last few blocks. We believe that not only the

exact number of blocks, but the combination of blocks is crucial. Therefore, only by considering both the structural parameters and the adaptive structure can the model that best suits the interests of the device be obtained.

Table 3: Analysis of Structural Parameters

Model	Method	Dataset					
		MovieLens-1M		MovieLens-10M		Amazon-Game	
		NDCG		Hit		NDCG	
SASRec	DeviceRec	0.0969	0.1816	0.0718	0.1308	0.0359	0.0541
	random block	0.0714	0.1325	0.0564	0.1045	0.0462	0.0696
	first k block	0.0174	0.0371	0.0030	0.0067	0.0384	0.0546
	last k block	0.0699	0.1285	0.0836	0.1464	0.0481	0.733
	w.o. structural vector	0.1166	0.2068	0.0872	0.1504	0.0472	0.0714
	Forward-OFA	0.1182	0.2081	0.0905	0.1566	0.0487	0.0754
NextItNet	DeviceRec	0.0975	0.1846	0.0638	0.1166	0.0275	0.0425
	random block	0.0346	0.0644	0.0306	0.0571	0.0343	0.0468
	first k block	0.0969	0.1795	0.0812	0.1425	0.0455	0.0663
	last k block	0.0082	0.0187	0.0132	0.0294	0.0184	0.0214
	w.o. structural vector	0.1209	0.2141	0.0812	0.1412	0.0456	0.0662
	Forward-OFA	0.1226	0.2141	0.0824	0.1429	0.0477	0.0708

4.3.5 Comparision between Forward-OFA and smaller baselines. As discussed above, our framework can obtain better performance with smaller models. Inspired by this, we begin to question whether this improvement is simply due to using small models rather than adaptive networks. To further clarify the necessity of adaptive networks, we replace all the baselines in Table 1 with smaller models, each of which is $\frac{1}{3}$ the size of the backbones in Table 1. We conduct experiments on three datasets and list the results in Table 4. The first observation is that larger models may not always provide better performance. While most larger models achieve better results, on some datasets, the smaller SASRec performs better, which is consistent with what we introduced in Figure 1. Another observation is that despite smaller models used by other baselines, our method still achieves the best performance, especially on MovieLens-1M and MovieLens-10M where our approach has improved by approximately 20% to 30%. It makes sense that users' latent interests can only be well modeled with adaptive models instead of uniformly large or small models.

4.4 Complexity and Privacy Analysis(RQ3)

As *Forward-OFA* takes user historical embeddings as input, uploading the sequence or embeddings to the cloud can lead to privacy issues by potentially leaking the device's privacy issues. In contrast, we address the privacy problem by placing the structure generator (a GRU and a fully connected layer) and the sequence extractor of the structural mapper(a GRU) on device. At the beginning of each session or when user interests dramatically change[31, 41], device would extract its own interest, select network paths itself and send them to cloud, which can protect user privacy because cloud can only get the sequence features extracted by devices. **Other modules will be saved on cloud to prevent much burden for devices.** Besides, as mentioned in Section 3.1 the candidate embeddings of this session will be stored in the local cache, there is no need to communicate with cloud in this session anymore.

For the complexity, we show the parameters and FLOPs of each component in Table 5. *Forward-OFA* only adds a small fraction of parameters(0.012 \times of SASRec and 0.005 \times of NextItNet), which is tolerable for resource-sensitive devices as **once deployed, these**

Table 4: Comparison between our model and others based on small models. We use a two-layer SASRec and a four-layer NextItNet for each baseline in this experiment. * denotes the baseline uses small models compared to those in Table 1

Model	Method	Dataset					
		MovieLens-1M		MovieLens-10M		Amazon-Game	
		NDCG	Hit	NDCG	Hit	NDCG	Hit
SASRec	DeviceRec*	0.0998	0.1897	0.0733	0.1351	0.0434	0.0656
	finetune*	0.1000	0.1902	0.0705	0.1324	0.0430	0.0647
	DUET*	0.1128	0.2018	0.0867	0.1499	0.0467	0.0717
	STTD*	0.0964	0.1836	0.0722	0.1331	0.0455	0.0694
	Rare Gem*	0.0993	0.1877	0.0733	0.1353	0.0450	0.0683
	Gater*	0.0991	0.1869	0.0728	0.1350	0.0467	0.0701
	Forward-OFA	0.1182	0.2081	0.0905	0.1566	0.0487	0.0754
NextItNet	DeviceRec*	0.0947	0.1798	0.0638	0.1166	0.0298	0.0471
	finetune*	0.0883	0.1755	0.0610	0.1136	0.0303	0.0474
	DUET*	0.1177	0.2081	0.0815	0.1369	0.0455	0.0664
	STTD*	0.0915	0.1753	0.0621	0.1149	0.0262	0.0422
	Rare Gem*	0.1025	0.1902	0.0684	0.1243	0.0329	0.0513
	Gater*	0.0973	0.1834	0.0630	0.1153	0.0309	0.0453
	Forward-OFA	0.1226	0.2141	0.0826	0.1429	0.0477	0.0708
	Improv.	29.42%	19.06%	29.05%	22.55%	59.81%	50.38%

modules won't be updated from the cloud frequently like parameters. Moreover, the extra modules on device take much less time to do inference than the original model. The inference of *Forward-OFA* will only happen when interests on device change dramatically or at the beginning of each session, occasional lightweight inference prevents it from occupying a large number of the device's resources for a long time. As for modules on cloud, even if they consume more parameters, they are shared among all devices and **can be executed in parallel**. Additionally, abundant computing resources on cloud make it efficient to hold these modules.

Table 5: FLOPs and Param of each component on devices and cloud. Backbone denotes the original sequential recommender, while Forward-device and Forward-cloud denote the extra component included by Forward-OFA, the former is stored on device while the latter is stored on cloud.

Model	Metric	Component		
		Backbone	Forward-device	Forward-cloud
SASRec	FLOPs	0.6244	0.2544	0.2621
	Param	3.9936	0.0503	8.1120
NextItNet	FLOPs	1.5053	0.2548	0.7864
	Param	9.5846	0.0507	19.2691

4.5 Case Study(RQ4)

To better understand the importance of adaptive networks for recommendation, we present the case study results from the MovieLens-10M dataset. Specifically, we sample four individual interactions and generate the corresponding networks through *Forward-OFA* as shown in Figure 6. The adaptive path(structure) is presented in Figure 6 (b), where we can observe that different user interests correspond to different paths. Apart from this, some sequences(1, 3, 4) only require part of the blocks to finish inference, while sequence 2 needs all blocks. Without compatible networks, the final recommendation list will be a deviation from expected results.

For a clearer understanding, we use the **orange** sequence(denoted as P) as the input of recommender and consider the following three networks: (1) the adaptive network generated from *Forward-OFA*. (2) the dense model trained with data on cloud. (3) incompatible networks generated using *Forward-OFA* and **yellow** interactions(denoted as Y). Figure 6(a) demonstrates that the adaptive network has the least parameters and provides the best result(with the next-predicted item and relevant items). The trained network gives moderate results, which contain relevant items but not the ones that users are most interested in. Conversely, when utilizing Y to generate masks, which exhibit disparate preferences compared to P , the outcomes are unfavorable, yielding few relevant movie recommendations. These results further demonstrate that only networks learned by *Forward-OFA* can best fit user requirements.

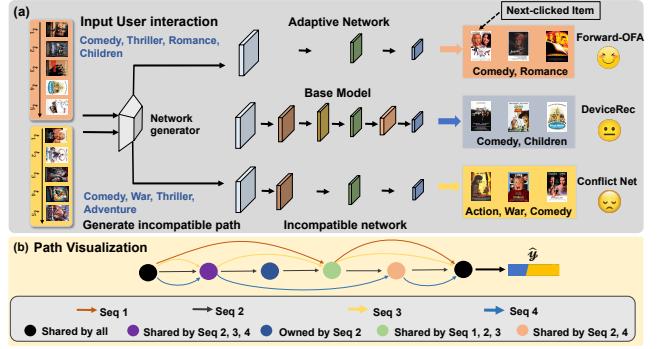


Figure 6: (a) The influence of compatible networks for Forward-OFA. The compatible networks(orange one) obtain the best performance compared to the trained model or conflict network generated with another sequence(yellow one). (b) Visualization of assigned paths of these interactions.

5 CONCLUSION

In this paper, we propose an efficient framework, *Forward-OFA* to construct local adaptive networks in only one forward pass. The compatible model provides better performance and lowers computational costs, as only a small fraction of blocks are necessary and beneficial for each device. Extensive experiments and comprehensive analysis of various real-world datasets and widely used sequential recommenders demonstrate the feasibility and effectiveness of the *Forward-OFA*. Furthermore, this work can be viewed as an initiative to explore the possibility of searching for compatible and lightweight networks for specific tasks.

Acknowledgement

This work was supported by the 2030 National Science and Technology Major Project (2022ZD0119100), the National Natural Science Foundation of China (62376243, 62402429, 62441605, 62037001), the Starry Night Science Fund at Shanghai Institute for Advanced Study (Zhejiang University), the Key Research and Development Program of Zhejiang Province(2024C03270), ZJU Kunpeng&Ascend Center of Excellence. This work was also supported by Ant group. The author gratefully acknowledges the support of Zhejiang University Education Foundation Qizhen Scholar Foundation.

References

- [1] Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. 2016. Neural module networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 39–48.
- [2] Gabriel Bender, Pieter-Jan Kindermans, Barret Zoph, Vijay Vasudevan, and Quoc Le. 2018. Understanding and simplifying one-shot architecture search. In *International conference on machine learning*. PMLR, 550–559.
- [3] Yukun Cen, Jianwei Zhang, Xu Zou, Chang Zhou, Hongxia Yang, and Jie Tang. 2020. Controllable multi-interest framework for recommendation. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2942–2951.
- [4] Jingfan Chen, Guanghui Zhu, Haojun Hou, Chunfeng Yuan, and Yihua Huang. 2022. AutoGSR: Neural architecture search for graph-based session recommendation. In *Proceedings of the 45th international ACM SIGIR conference on research and development in information retrieval*. 1694–1704.
- [5] Tong Chen, Hongzhi Yin, Yujia Zheng, Zi Huang, Yang Wang, and Meng Wang. 2021. Learning elastic embeddings for customizing on-device recommenders. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 138–147.
- [6] Zhourong Chen, Yang Li, Samy Bengio, and Si Si. 2019. You look twice: Gaternet for dynamic filter selection in cnns. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 9172–9180.
- [7] Liu Chong, Xiaoyang Liu, Rongqin Zheng, Lixin Zhang, Xiaobo Liang, Juntao Li, Lijun Wu, Min Zhang, and Leyu Lin. 2023. CT4Rec: Simple yet Effective Consistency Training for Sequential Recommendation. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 3901–3913.
- [8] Chengquan Feng, Li Lyra Zhang, Yuanchi Liu, Jiahang Xu, Chengruidong Zhang, Zhiyuan Wang, Ting Cao, Mao Yang, and Haisheng Tan. 2024. {LitePred}: Transferable and Scalable Latency Prediction for {Hardware-Aware} Neural Architecture Search. In *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*. 1463–1477.
- [9] Kairui Fu, Qiaowei Miao, Shengyu Zhang, Kun Kuang, and Fei Wu. 2023. End-to-End Optimization of Quantization-Based Structure Learning and Interventional Next-Item Recommendation. In *CAAI International Conference on Artificial Intelligence*. Springer, 415–429.
- [10] Kairui Fu, Shengyu Zhang, Zheqi Lv, Jingyuan Chen, and Jiwei Li. 2024. DIET: Customized Slimming for Incompatible Networks in Sequential Recommendation. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 816–826.
- [11] Yu Gong, Ziwen Jiang, Yufei Feng, Binbin Hu, Kaiqi Zhao, Qingwen Liu, and Wenwu Ou. 2020. EdgeRec: recommender system on edge in Mobile Taobao. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 2477–2484.
- [12] David Ha, Andrew M. Dai, and Quoc V. Le. 2017. HyperNetworks. In *International Conference on Learning Representations*.
- [13] F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. *ACM Trans. Interact. Intell. Syst.* 5, 4, Article 19 (Dec. 2015), 19 pages. <https://doi.org/10.1145/2827872>
- [14] F. Maxwell Harper and Joseph A. Konstan. 2015. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)* 5, 4 (2015), 1–19.
- [15] Ruining He, Wang-Cheng Kang, Julian J McAuley, et al. 2018. Translation-based Recommendation: A Scalable Method for Modeling Sequential Behavior. In *IJCAI*. 5264–5268.
- [16] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2016. Session-based Recommendations with Recurrent Neural Networks. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, Yoshua Bengio and Yann LeCun (Eds.). <http://arxiv.org/abs/1511.06939>
- [17] Shoukang Hu, Ruochen Wang, HONG Lanqing, Zhenguo Li, Cho-Jui Hsieh, and Jia Shi Feng. 2022. Generalizing Few-Shot NAS with Gradient Matching. In *International Conference on Learning Representations*.
- [18] Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144* (2016).
- [19] Yangqin Jiang, Chao Huang, and Lianghao Huang. 2023. Adaptive Graph Contrastive Learning for Recommendation. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (Long Beach, CA, USA) (KDD '23)*. Association for Computing Machinery, New York, NY, USA, 4252–4261. <https://doi.org/10.1145/3580305.3599768>
- [20] Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *2018 IEEE international conference on data mining (ICDM)*. IEEE, 197–206.
- [21] Louis Kirsch, Julius Kunze, and David Barber. 2018. Modular networks: Learning to decompose neural computation. *Advances in neural information processing systems* 31 (2018).
- [22] Himabindu Lakkaraju, Julian McAuley, and Jure Leskovec. 2013. What's in a name? understanding the interplay between titles, content, and communities in social media. In *Proceedings of the international AAAI conference on web and social media*, Vol. 7. 311–320.
- [23] Jiacheng Li, Ming Wang, Jin Li, Jinmiao Fu, Xin Shen, Jingbo Shang, and Julian McAuley. 2023. Text is all you need: Learning language representations for sequential recommendation. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 1258–1267.
- [24] Shuchang Liu, Shuyuan Xu, Wenhai Yu, Zuohui Fu, Yongfeng Zhang, and Amelie Marian. 2021. FedCT: Federated collaborative transfer for recommendation. In *Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval*. 716–725.
- [25] Zheqi Lv, Shaoxuan He, Tianyi Zhan, Shengyu Zhang, Wenqiao Zhang, Jingyuan Chen, Zhao Zhou, and Fei Wu. 2024. Semantic Codebook Learning for Dynamic Recommendation Models. In *Proceedings of the 32nd ACM International Conference on Multimedia*. 9611–9620.
- [26] Zheqi Lv, Wenqiao Zhang, Zhengyu Chen, Shengyu Zhang, and Kun Kuang. 2024. Intelligent model update strategy for sequential recommendation. In *Proceedings of the ACM on Web Conference 2024*. 3117–3128.
- [27] Zheqi Lv, Wenqiao Zhang, Shengyu Zhang, Kun Kuang, Feng Wang, Yongwei Wang, Zhengyu Chen, Tao Shen, Hongxia Yang, Beng Chin Ooi, et al. 2023. Duet: A tuning-free device-cloud collaborative parameters generation framework for efficient device model generalization. In *Proceedings of the ACM Web Conference 2023*. 3077–3085.
- [28] Jianmo Ni, Jiacheng Li, and Julian McAuley. 2019. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*. 188–197.
- [29] Chaoyue Niu, Fan Wu, Shaojie Tang, Lifeng Hua, Rongfei Jia, Chengfei Lv, Zhihua Wu, and Guihai Chen. 2020. Billion-scale federated learning on mobile clients: A submodel design with tunable privacy. In *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking*. 1–14.
- [30] Yoon-Joo Park and Alexander Tuzhilin. 2008. The long tail of recommender systems and how to leverage it. In *Proceedings of the 2008 ACM conference on Recommender systems*. 11–18.
- [31] Xufeng Qian, Yue Xu, Fuyu Lv, Shengyu Zhang, Ziwen Jiang, Qingwen Liu, Xiaoyi Zeng, Tat-Seng Chua, and Fei Wu. 2022. Intelligent request strategy design in recommender system. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 3772–3782.
- [32] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing personalized markov chains for next-basket recommendation. In *Proceedings of the 19th international conference on World wide web*. 811–820.
- [33] Kartik Sreenivasan, Jy-yong Sohn, Liu Yang, Matthew Grinde, Alliot Nagle, Hongyi Wang, Eric Xing, Kangwook Lee, and Dimitris Papailiopoulos. 2022. Rare gems: Finding lottery tickets at initialization. *Advances in Neural Information Processing Systems* 35 (2022), 14529–14540.
- [34] 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 *Proceedings of the 28th ACM international conference on information and knowledge management*. 1441–1450.
- [35] Ximeng Sun, Rameswar Panda, Rogerio Feris, and Kate Saenko. 2020. Adashare: Learning what to share for efficient deep multi-task learning. *Advances in Neural Information Processing Systems* 33 (2020), 8728–8740.
- [36] Jiaxi Tang and Ke Wang. 2018. Personalized top-n sequential recommendation via convolutional sequence embedding. In *Proceedings of the eleventh ACM international conference on web search and data mining*. 565–573.
- [37] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).
- [38] Chuhan Wu, Fangzhao Wu, Tao Qi, Yongfeng Huang, and Xing Xie. 2022. FedAttack: Effective and covert poisoning attack on federated recommendation via hard sampling. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 4164–4172.
- [39] Yunjia Xi, Weiwen Liu, Yang Wang, Ruiming Tang, Weinan Zhang, Yue Zhu, Rui Zhang, and Yong Yu. 2023. On-device integrated re-ranking with heterogeneous behavior modeling. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 5225–5236.
- [40] Xin Xia, Hongzhi Yin, Junliang Yu, Qinyong Wang, Guandong Xu, and Quoc Viet Hung Nguyen. 2022. On-device next-item recommendation with self-supervised knowledge distillation. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 546–555.
- [41] Jiangchao Yao, Feng Wang, Xichen Ding, Shaohu Chen, Bo Han, Jingren Zhou, and Hongxia Yang. 2022. Device-cloud collaborative recommendation via meta controller. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 4353–4362.
- [42] Jiangchao Yao, Feng Wang, Kunyang Jia, Bo Han, Jingren Zhou, and Hongxia Yang. 2021. Device-cloud collaborative learning for recommendation. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 3865–3874.

- [43] Jiangchao Yao, Shengyu Zhang, Yang Yao, Feng Wang, Jianxin Ma, Jianwei Zhang, Yunfei Chu, Luo Ji, Kunyang Jia, Tao Shen, et al. 2022. Edge-cloud polarization and collaboration: A comprehensive survey for ai. *IEEE Transactions on Knowledge and Data Engineering* 35, 7 (2022), 6866–6886.
- [44] 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.
- [45] Shengyu Zhang, Tan Jiang, Tan Wang, Kun Kuang, Zhou Zhao, Jianke Zhu, Jin Yu, Hongxia Yang, and Fei Wu. 2020. Devlibert: Learning deconfounded visiolinguistic representations. In *Proceedings of the 28th ACM International Conference on Multimedia*. 4373–4382.
- [46] Shengyu Zhang, Dong Yao, Zhou Zhao, Tat-Seng Chua, and Fei Wu. 2021. Causerec: Counterfactual user sequence synthesis for sequential recommendation. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 367–377.
- [47] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. 2019. Deep learning based recommender system: A survey and new perspectives. *ACM computing surveys (CSUR)* 52, 1 (2019), 1–38.
- [48] Tunhou Zhang, Dehua Cheng, Yuchen He, Zhengxing Chen, Xiaoliang Dai, Liang Xiong, Feng Yan, Hai Li, Yiran Chen, and Wei Wen. 2023. NASRec: weight sharing neural architecture search for recommender systems. In *Proceedings of the ACM Web Conference 2023*. 1199–1207.
- [49] Yipeng Zhang, Xin Wang, Hong Chen, and Wewu Zhu. 2023. Adaptive disentangled transformer for sequential recommendation. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 3434–3445.
- [50] Yiyang Zhao, Linnan Wang, Yuandong Tian, Rodrigo Fonseca, and Tian Guo. 2021. Few-shot neural architecture search. In *International Conference on Machine Learning*. PMLR, 12707–12718.
- [51] Ruiqi Zheng, Liang Qu, Tong Chen, Kai Zheng, Yuhui Shi, and Hongzhi Yin. 2024. Personalized Elastic Embedding Learning for On-Device Recommendation. *IEEE Transactions on Knowledge and Data Engineering* (2024).
- [52] Guorui Zhou, Xiaoqiang Zhu, Chenru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. 2018. Deep interest network for click-through rate prediction. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*. 1059–1068.
- [53] Barret Zoph and Quoc Le. 2016. Neural Architecture Search with Reinforcement Learning. In *International Conference on Learning Representations*.

A Experimental Setup

A.1 Datasets

We evaluate our framework on four public benchmarks in recommender system[14, 22] *Movielens-1M*¹, *Movielens-10M*², *Amazon-Food*, and *Amazon-Game*³. The statistics of which are presented in table 6. Consistent with prior research, all user-item pairs with positive ratings in the dataset are considered positive samples. Owing to variations in sparsity across the datasets, we exclude users and items with fewer than 20 interactions to provide reliable results. To construct sequential scenarios of user interaction in practical applications, we order each user’s interactions by their interaction time. The last interaction of each user will be used for test and others will be used for training.

Table 6: Statistics of the datasets.

Dataset	#Users	#Items	#Interactions	#SeqLen	#Sparsity
ML-1M	6,040	3,012	994,852	164.71	94.52%
ML-10M	69,878	8,882	9,983,342	142.87	98.39%
Food	27,149	51,885	517,548	19.06	99.97%
Game	11,292	68,308	223,696	19.81	99.91%

¹<https://grouplens.org/datasets/movielens/1m>

²<https://grouplens.org/datasets/movielens/10m>

³<https://nijianmo.github.io/amazon/index.html>

A.2 Baseline

Base Models. Given that current sequential recommenders predominantly consist of Transformer and CNN, we choose *SASRec*[20] and *NextItNet*[44] as the base model, upon which we incorporate different methods:

- **SASRec**[20] integrates a self-attention mechanism [37] into recommender system, thereby enhancing the capture of a user’s interests by considering the individual impact of each item in the historical sequence on the target item.
- **NextItNet**[44] employs a convolutional neural network to introduce local perception and parameter sharing. It stands as the pioneering work to utilize residual learning in recommender system, effectively modeling long-range and intricate dependencies within historical sequences.

Algorithm 1 Forward-OFA on Cloud

Module 1: ▷ Structure-related Parameters Mapper

Target: Latent Interests h , L Parameter Mappers $\{H'_k\}_{k=1,\dots,L}$
 \mapsto Structure-related Parameters W
 ▷ Executed in parallel
Input: Latent Interests h , L Parameter Mappers $\{H'_k\}_{k=1,\dots,L}$
Output: Structure-related Parameters W

Module 2: ▷ Model Assembler

Target: Structure Representation $\{\beta_{i,0}, \beta_{i,1}\}$, Structure-related Parameters $W \mapsto$ Device-specific Networks M_d
Input: Structure Representation $\{\beta_{i,0}, \beta_{i,1}\}$, Structure-related Parameters W
Output: Device-specific Networks M_d

Base Methods.

- **Device-Rec** trains a unified network on cloud and directly sends it to each device. Once deployed, these networks are not updated any further.
- **Finetune** initially trains a model with all available data on cloud. Each device subsequently fine-tunes its model with local data to better align with its interests.
- **DUET**[27] requests different parameters from cloud for each device, thereby achieving significant progress in device-cloud collaborative learning.
- **Rare Gem**[33] proposes identifying less important elements within a given network and subsequently training the sparse network to achieve comparable performance while reducing model size.
- **STTD**[40] represents each layer with small matrices through multiplication to reduce parameters. However, additional computation is required to reconstruct the original layers.
- **Gater**[6] employs structural pruning to remove redundant filters, thereby enhancing the efficiency of inference.

B Pseudocode of Forward-OFA

We briefly introduce Forward-OFA on cloud and device in Algorithm 1 and 2. First, as shown in Algorithm 2, the compatible network path $\{\beta_{i,0}, \beta_{i,1}\}$ is calculated locally based on the real-time interaction, and the path is then used to encode latent interests h

Algorithm 2 Forward-OFA on Device

Module 1: ▷ *Structure Controller*

Target: Real Time Interaction X_d , Sequence Extractor E , Structure Controller $H \mapsto$ Structure Representation $\{\beta_{i,0}, \beta_{i,1}\}$

Input: Real Time Interaction X_d , Sequence Extractor E , Structure S

ture Controller H

Module 2: ▷ Privacy-focused Interest Extraction

Target: Real Time Interaction X_d , Structure Representation $\{\beta_{i,0}, \beta_{i,1}\}$, Sequence Extractor $E' \mapsto$ Latent Interests h

Input: Real Time Interaction X_d , Structure Representation

$\{\beta_{i,0}, \beta_{i,1}\}$, Sequence Extractor E'

Module 3: ▷ Compatible Network Requests

Target: Structure Representation $\{\beta_{i,0}, \beta_{i,1}\}$, Latent Interests $h \mapsto$ Device-specific Networks M_d

- ▶ Send requests to cloud.

Input: Structure Representation $\{\beta_{i,0}, \beta_{i,1}\}$, Latent Interests h

Output: Device-specific Networks M_d

Module 4: ▷ *On-device Recommendation*

Target: Real Time Interaction X_d , Device-specific Networks
 $M_d \mapsto$ Next-clicked Item $y_d^{t'}$

Input: Real Time Interaction X_d , Device-specific Networks

$$M_d \quad \quad \quad t'$$

to protect privacy. This process is both memory and calculation-efficient and does not occupy much resource on device. Then device will send its latent interest to cloud to finish the network assembly. In Algorithm 1, parameter mapping for each layer on cloud can be executed in parallel. Compatible networks will be sent back to finish later recommendations.

C Association between Neural Architecture Search and Forward-OFA

Smilarity. Both NAS (Neural Architecture Search) and Forward-OFA aim to identify suitable networks for devices that adapt to specific data distributions while minimizing resource consumption. They both address an often overlooked research problem: the fundamental significance of network structures to various data distributions. Forward-OFA is also partially motivated by those methods to detect light subnetworks for each device.

Difference. However, NAS-based methods[4, 50, 53] have to search appropriate structures in advance and retrain local data to get device-specific networks. The whole subnet-constructing process takes a long time, making it impractical for billions of users. For example, the recent method LitePred[8] matches and finetunes models with device data, requiring substantial resources and leading to longer responses. Besides, limited on-device interactions may cause overfitting or suboptimal performance. In contrast, Forward-OFA achieves adaptation through a single forward pass, directly

mapping local interests to networks. We also conducted an additional comparable experiment on LitePred, where the results with ours are shown in Table 7. Owing to the accurate matching process, *LightPred* successfully outperforms *DeviceRec* and *Finetune*, demonstrating the necessity of discovering valuable subnetworks. However, as mentioned above, limited on-device interactions and various latent device interests restrict its potential to learn compatible parameters, leading to a large performance drop compared to Forward-OFA which directly builds networks from real-time interactions.

Table 7: Comparation between NAS-based method LitePred and Forward-OFA.

Model	Method	Dataset					
		MovieLens-1M		MovieLens-10M		Amazon-Game	
		NDCG ↑	Hit ↑	NDCG ↑	Hit ↑	NDCG ↑	Hit ↑
SASRec	DeviceRec	0.0969	0.1816	0.0718	0.1308	0.0359	0.0541
	Finetune	0.0939	0.1793	0.0681	0.1272	0.0343	0.0537
	LitePred	0.0990	0.1877	0.0735	0.1347	0.0419	0.0635
	Forward-OFA	0.1182	0.2081	0.0905	0.1566	0.0487	0.0754
NextItNet	DeviceRec	0.0975	0.1846	0.0638	0.1166	0.0275	0.0440
	Finetune	0.0879	0.1715	0.0635	0.1165	0.0279	0.0422
	LitePred	0.0993	0.1828	0.0652	0.1184	0.0310	0.0484
	Forward-OFA	0.1226	0.2140	0.0826	0.1429	0.0477	0.0708

D Adaptation Time to Changing Interests

To build insights into the efficient adaptation of Forward-OFA, in this section, we compare both the adaptation time once interests shift and the communication time for the model update. In this section, we will further validate the efficiency of Forward-OFA in quickly adapting to changing user interests. As shown in Table 8, we found that on-cloud fine-tuning on an Nvidia RTX 3090 GPU(35.58 TFLOPS) takes 1000 times longer than Forward-OFA. For mobile devices like the iPhone 16(1789.4 GFLOPS), on-device fine-tuning can exceed 94 seconds, which is about 10,000 times slower than Forward-OFA.

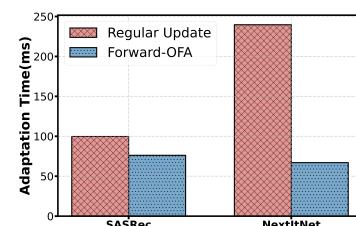


Figure 7: Communication time for the model update of Forward-OFA and Regular Strategy.

Moreover, Forward-OFA requires significantly less bandwidth as only necessary components are transmitted compared to those enumorous blocks in the original backbones. This results in quicker responses(4G network with 40Mbps) as shown in Figure 7.

Table 8: Adaptation time of Forward-OFA and On-device/cloud Finetune.

Model	Forward-OFA	On-cloud Finetune	On-device Finetune
SASRec	0.0011s	4.6259s	$\geq 94.1878s$
NextItNet	0.0051s	4.8764	$\geq 99.2882s$