Progressive Horizon Learning: Adaptive Long Term Optimization for Personalized Recommendation

CONGRUI YI*, Amazon, USA
DAVID ZUMWALT*, Amazon, USA
ZIJIAN NI, Amazon, USA
SHREYA CHAKRABARTI, Amazon, USA

As E-commerce and subscription services scale, personalized recommender systems are often needed to further drive long term business growth in acquisition, engagement, and retention of customers. However, long-term metrics associated with these goals can require several months to mature. Additionally, deep personalization also demands a large volume of training data that take a long time to collect. These factors incur substantial lead time for training a model to optimize a long-term metric. Before such model is deployed, a recommender system has to rely on a simple policy (e.g. random) to collect customer feedback data for training, inflicting high opportunity cost and delaying optimization of the target metric. Besides, as customer preferences can shift over time, a large temporal gap between inputs and outcome poses a high risk of data staleness and suboptimal learning. Existing approaches involve various compromises. For instance, contextual bandits often optimize short-term surrogate metrics with simple model structure, which can be suboptimal in the long run, while Reinforcement Learning approaches rely on an abundance of historical data for offline training, which essentially means long lead time before deployment. To address these problems, we propose Progressive Horizon Learning Recommender (PHLRec), a personalized model that can progressively learn metric patterns and adaptively evolve from short- to long-term optimization over time. Through simulations and real data experiments, we demonstrated that PHLRec outperforms competing methods, achieving optimality in both deployment speed and long-term metric performances.

CCS Concepts: • **Computing methodologies** → **Learning to rank**; • **Information systems** → **Recommender systems**; **Personalization**.

Additional Key Words and Phrases: long term optimization, deep learning, model deployment, data augmentation, masking

ACM Reference Format:
Congrui Yi, David Zumwalt, Zijian Ni, and Shreya Chakrabarti. 2023. Progressive Horizon Learning: Adaptive Long Term Optimization for Personalized Recommendation. In *Seventeenth ACM Conference on Recommender Systems (RecSys '23), September 18–22, 2023, Singapore, Singapore.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3604915.3608852

1 INTRODUCTION

Continued growth for an E-commerce or subscription service requires focused efforts to drive long-term acquisition, engagement, and retention of customers. They often create tailored experiences targeting different customer interests to achieve these goals. For example, an audio streaming service may consider ads for sports podcasts, music hits or news channels to engage its customers. To match each customer with relevant ads or content, a common approach is building a personalized recommender system, which is often required to meet the following business needs: (i) optimize

*The first two authors contributed equally to this work.

Manuscript submitted to ACM

long-term success metrics, e.g. 2-month retention for subscription services, total viewing minutes within a month for video streaming; (ii) quick to deploy - ideally as soon as the service is launched with minimal time needed for initial training data collection; (iii) enable deep personalization by utilizing high-dimensional customer features.

However, such requirements are at odds with each other, presenting unique scientific challenges. Firstly, modeling a long-term metric incurs substantial lead time in data collection, as the target metric outcomes are unobserved until a few months after recommendations are made. Secondly, while deep personalization maximizes recommendation impact, it also demands a massive amount of customer feedback data for training that can take months to collect. Thirdly, as customer preferences can shift over time, the large temporal gap between input data and outcome metrics can cause a supervised learning approach to fail due to data staleness. The reason is, even if we train a model continuously, we still cannot include fresh data for training, as long as we have to wait for the outcomes to mature.

These challenges represent a critical tradeoff between long-term optimization (LTO), personalization, and model deployment speed. To the best of our knowledge, there is no existing body of work which addresses them simultaneously. Contextual bandits [7, 23] are capable of balancing exploitation with exploration by incorporating model uncertainty, hence widely used for cold-start recommendation with little or no historical data for training. However, works on bandits often rely purely on immediate customer feedback such as clicks or ratings for reward definition and policy update [23, 26, 38]. Since such immediate reward is only a short-term surrogate of some long-term metric targeted by business, this approach can lead to suboptimal recommendations. Besides, bandits often make use of lightweight linear model structure, which is suitable for continuous training but may not be sufficient for deep personalization.

Conversely, existing works on LTO usually depend on an abundance of historical data and ignore the business need of quick model deployment for cold start situations. One body of work involves applications of Reinforcement learning (RL) [2, 16, 18, 37, 39], which represent a long-term objective by the cumulative sum of immediate rewards from a sequence of recommendations made to each customer. Although targeting long-term, these works all required a large amount of logged data on recommendation trajectories for simulator or off-policy learning, which essentially means long lead time for data collection before such a model can be deployed. In addition, training these RL models requires numerous interactions between the recommender system and each customer, which limits their applicability to tasks with only sparse interactions. Another line of work involves modeling of customer lifetime values (CLV). Conventional researches on CLV [4, 10, 28] mainly focused on identifying high-value customers for marketing, but some recent works [29, 30] extended CLV to recommendation space by recognizing a conceptual connection between CLV and cumulative rewards used in RL. Unfortunately, these works adopted off-policy learning methods using a considerable amount of data, hence also subject to the aforementioned limitations with RL approaches regarding model deployment. Lastly, delayed feedback modeling (DFM) for online advertising [5, 21, 33] bears some resemblance to LTO conceptually, but the two differ in important aspects. Firstly, DFM exclusively focused on a binary conversion event such as click or purchase as the customer feedback, whereas LTO is concerned with generic metrics that can be binary as well as integers or real-valued. Secondly, such conversion event studied by DFM occurs at a randomly delayed time for which the time to event can be modeled, while the long-term metrics discussed here are essentially summaries of customer behaviors or statuses for a given time duration. Therefore, it may not be feasible to apply DFM directly to LTO for recommendation.

In this paper, we propose a novel technique called Progressive Horizon Learning (PHL) and a PHL-based recommender model (*PHLRec*), which systematically address the aforementioned challenges of lead time, data collection and staleness with LTO. PHLRec is a multi-target deep neural network model that gets trained initially with short-term targets and evolves over time to progressively adapt to longer term predictions, hence eliminating the need to wait for a long term

metric to mature. Additionally, PHLRec incorporates a pretrained customer feature encoder, which largely reduces the amount of training data needed for the model to achieve adequate performance for deep personalization. As shown later in our experiments, PHLRec achieves simultaneous optimality on both deployment speed and optimization of long-term metrics, at a significantly lower opportunity cost to business compared to existing methods.

2 METHODOLOGY

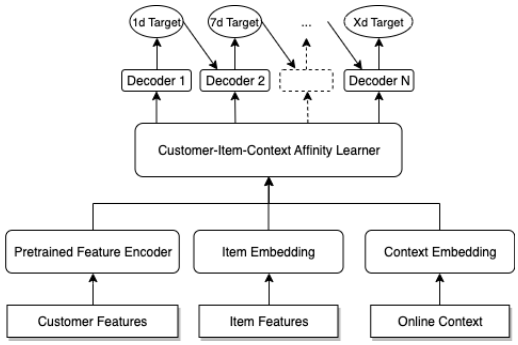

Fig. 1. Overall model architecture of PHLRec.

In this section we present the proposed PHLRec model. As illustrated in Figure 1, PHLRec consumes 3 types of feature inputs: customer features, item features and optional online context, which are respectively processed by a pretrained encoder and two separate embedding components into numerical vectors of comparable dimensions. Then the concatenated vector is fed to a customer-item-context affinity learner to generate an affinity vector. Finally, the model has a set of multi-horizon output decoders with "linkages" between adjacent horizons: each decoder takes the affinity vector and the prediction from the previous horizon as input and outputs prediction at a new horizon. Next we will introduce key designs of PHLRec for resolving the challenges with LTO for personalized recommendation.

2.1 Progressive Horizon Learning

To address the lead time and data staleness issues associated with optimizing long-term metrics, we introduce a conceptually new technique called Progressive Horizon Learning (PHL), which is the cornerstone of PHLRec. Specifically, it consists of three components, multi-horizon augmentation, horizon masking, and dynamic horizon selection.

2.1.1 Multi-Horizon Augmentation. Given a targeted long-term metric, we introduce a set of auxiliary short- and intermediate-term metrics, bridging the temporal gap between the target metric and the predictive features. For example, if 60 day retention is the target metric to be optimized, one can additionally introduce 1, 7, 14, 30 day retention targets that form a progression from 1 to 60. More formally, let T be the horizon to optimize, and Y_T denote the metric outcome value at T after recommendation. Then we define a temporally ordered set of horizons $\mathcal{T} = \{t_i : 1 \leq t_1 < t_2 < \cdots < t_K \leq T\}$ and introduce the associated outcomes $\mathcal{Y} = \{Y_t : t \in \mathcal{T}\}$ to the data as augmented model targets. Accordingly, we consider a "linked" multi-target network as shown in Figure 1. We set a decoder for each horizon which consists of 2 dense layers and outputs a scalar prediction, and connect them together such that the prediction for Y_{t_i} is again used as an input to the decoder for $Y_{t_{i+1}}$, the target of the next horizon. With these between-horizon linkages, perturbation at one horizon can be propagated to longer ones, allowing the model to be adaptive to time trends detectable from

short-term signals. Additionally, it is worth noting that although throughout this paper we discuss multi-horizon augmentation in the sense of introducing extra targets for the same business metric which differ only in time horizons, our approach also applies in situations that one wants to include different kinds of metrics. For instance, to optimize 60-day retention, beside shorter term retention targets, one may consider also adding other types of auxiliary metrics such as membership signup or engagement, which is feasible under the general PHL framework.

2.1.2 Horizon Masking. Multi-horizon augmentation alone is insufficient for reducing lead time, since the network still includes the long-term metric as a model target, and ordinarily training such multi-target network would still require us to wait long enough to collect all outcomes. Hence, we further devise a technique called "horizon masking". For every data point, each horizon-indexed target of it is associated with a binary mask, which is initialized to 0, and later changed to 1 when the actual outcome is observed, or in other words, "matured" at the horizon. Then the PHLRec model is trained by minimizing an objective which is a sum of masked losses on all horizons with loss weight λ_t and horizon mask m_{it}, for data point $i \in \{1, \ldots, N\}$ and horizon $t \in \mathcal{T}$ defined above:

$$\text{Objective} = \frac{1}{N} \sum_{i=1}^{N} \sum_{t \in \mathcal{T}} \lambda_t \cdot m_{it} \cdot f(y_{it}, \widehat{y}_{it}) \tag{1}$$

where y_{it} is the model target at horizon t, $\widehat{y}_{it}$ is the corresponding model prediction, λ_t's are tunable hyper-parameters, and f is the loss function for each target-prediction pair. f is selected according to the target type, typically cross-entropy for binary targets and probabilistic predictions, and squared or Huber loss [17] for real valued ones. It is noteworthy that the masks m_{it} are defined for each data point i and horizon t, which ensure that unobserved y_{it}'s (marked with $m_{it} = 0$) do not contribute to model training, allowing the model to dynamically evolve through retraining based on outcome maturity. In particular, the decoder weights for a long horizon are not updated until there are some data points with matured outcomes at that horizon. For any given horizon, the more matured outcomes are used for supervision, the more accurate the model becomes at making prediction for that horizon.

Binary masking has been mainly applied to self-attention decoders of Transformer model [32] and some extensions, e.g. GPT [1, 27] for natural language understanding, and SASRec [19] for sequential recommendation. The purpose of masking in those works was to preserve auto-regressive property for sequential prediction and prevent information leak from future time steps to influence prediction at the current step. In comparison, the problem we consider here is not necessarily formulated as sequential prediction or optimization. Instead, we have motivated horizon masking as part of PHL to address the lead time issue for optimizing a long term outcome. Despite its simplicity, horizon masking allows us to train on a natural data mix of recency and outcome maturity, thereby eliminating the need to wait for the longest term outcome to mature. To the best of our knowledge, this is the first time that masking is designed for such purpose.

2.1.3 Dynamic Horizon Selection and Recommendation Inference. When PHLRec is launched to a brand new task, data collected initially will only contain target signals for the shortest horizon, as longer ones remain unobserved, and the corresponding decoder weights are not trained due to horizon masking. Therefore, instead of directly using predictions at the longest horizon for recommendation inference, we designed a dynamic mechanism for horizon selection: (i) set a volume threshold (e.g., 10000) and keep a horizon only if the number of data points matured at this horizon exceeds it; (ii) compute testing performance metrics for each horizon and filter them based on some guardrails (e.g., AUC > 0.7, or estimated metric mean from off-policy evaluation methods such as Inverse Propensity Scores [15] or Doubly Robust Estimator [9]); (iii) pick the longest horizon from the ones satisfying (i) (ii). Given the selected horizon, we consider two

PHLRec-based recommendation policies to balance exploitation and exploration: ϵ-greedy, and Boltzmann Exploration (i.e. softmax policy $p(a) \propto \exp(\eta\widehat{y}(a))$ for each item a [3]). Typically, the hyperparameters ϵ, η can be chosen based on data volume and item space complexity. Overall, this mechanism enables us to leverage predictions of a short-term surrogate until the next, longer-term horizon has matured with reliable performance, and the recommender system will automatically evolve from short- to long-term optimization. Lastly, although throughout this paper we mainly consider recommendation tasks where each customer is shown one item at a time, we argue that PHLRec is also applicable to top-K recommendation, for which the ranked list can be formed via an iterative application of ϵ-greedy or Boltzmann Exploration at each position of the list upon the pool of remaining items.

2.2 Pretraining A Customer Feature Encoder

To resolve the conflict between the desire to launch quickly and the substantial data volume needed to train a personalized model, we employ a "pretrained customer feature encoder". The technique is motivated by an observation that given any recommendation task the input features we use can be grouped into two: (1) task-specific item and online context features, which tend to be low dimensional vectors that include candidate item index and associated attributes, as well as other real-time context information like webpage location or device type when the recommendation happens. (2) task-agnostic customer features, which are behavioral and engagement features. The dimension of customer features can often be very high, depending on granularity of data logging system. While the item/context features are pertinent to a specific recommendation task, the customer features can be collected independently over a span of years.

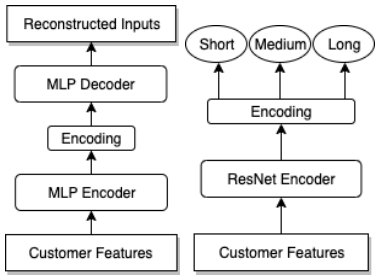

Fig. 2. Two types of pretrained feature encoders. Left: autoencoder, right: supervised encoder.

Given such separation, we consider pretraining a feature encoder with historical customer data to compress the high-dimensional task-agnostic customer features to a low-dimensional vector to be shared across recommendation tasks. Such feature compression allows the downstream model to be very lightweight yet performant. As shown later in Section 3.2, compared to an end-to-end neural network that takes raw features as inputs, PHLRec model equipped with a pretrained encoder component (Figure 1) can have 90% fewer trainable weights, and consequently requires roughly 90% less training data to achieve equivalent performances, enabling a quick launch with low data volume to start with. We have implemented two types of encoders (Figure 2), an unsupervised autoencoder, and a supervised encoder with common business metrics as targets, which are motivated for different application scenarios. For the autoencoder model, multi-layer perceptron (MLP) is considered for its encoder and decoder components, which are trained together to learn a low-dimensional encoding vector of input customer features by minimizing reconstruction error, thus the encodings can be versatile for arbitrary tasks. In comparison, the supervised encoder model is a deep residual network (aka. ResNet [13]), constructed by stacking "residual blocks" that each consist of 2 fully-connected

layers with ReLU activation, Batch normalization, and a skip-layer residual connection. It is trained to learn an encoding vector (i.e. last hidden layer before output) specialized in predicting the given set of targets, so it is suitable primarily for applications with same or correlated targets, and potentially outperforms autoencoders in such cases. The utility of pretrained encoders for transfer learning and training data reduction has been thoroughly researched in NLP [8, 22] and multimodal learning [24]. For instance, a pretrained BERT encoder [8] achieved state-of-the-art performances on a wide range of NLP tasks with transfer learning, where only a small set of task-specific data is required for model fine-tuning. In the area of recommender systems, however, applications of pretraining mainly focused on improving prediction accuracy by bringing in the additional encodings as features [25, 35, 36], and its utility for training data reduction is not well explored. To the best of our knowledge, our work is the first one within the domain of recommender systems to incorporate pretraining for this purpose and demonstrate its efficacy. Lastly, it should be noted that the pretraining idea adopted here is not necessarily limited to customer features, but applicable to any features regarded as task-agnostic. For instance, if each item is associated with a text description, a pretrained BERT encoder can be considered.

2.3 Customer-Item-Context Affinity Learner

The customer-item-context affinity learner that connects the processed inputs with the targets is responsible for learning the affinities between any given combination of customer, item, and optional context information. It is also a deep residual network similar to supervised encoder, except that it takes the concatenated vector of customer encoding, item and context embeddings as input and outputs a learned affinity vector of the same dimension. We chose this structure due to the ability of residual blocks to mitigate the "vanishing gradient" issue [14] commonly encountered within deep networks. Wide & Deep [6] is an earlier work on recommender model which adopted MLP for its "deep" component, hence roughly speaking our affinity learner can be perceived as an upgrade of that. Additionally, we may further extend its current structure by incorporating an extra Cross Network [34] or DeepFM [12] component to better characterize high-order interactions among the inputs.

3 EXPERIMENTS

3.1 Simulation Experiments for Progressive Horizon Learning

We studied the utility of PHL with simulation experiments. Binary rewards were simulated with Weibull survival probability $S(t|x,a)$ for time horizon (in days) t given customer context vector x and item a, formulated as

$$\begin{aligned} &\beta_i(x,a,t) = \lambda_{ia} \cdot f_i(x,a) \cdot g_{ia}(t), \;\; i = 1,2, \\ &S(t|x,a) = \exp\left(-(\beta_1(x,a,t)\cdot t)^{\beta_2(x,a,t)}\right), \end{aligned} \tag{2}$$

where each $f_i(x,a)$ is a two-tower neural network characterizing reward contextualization on the interaction of x and a, and each $g_{ia}(t) = 1 + c_{ia1}\log(t+1) + c_{ia2}\left[(t+1)^{c_{ia3}} - 1\right]$ defines a time trend to simulate nonstationarity. Base parameters λ_{ia}'s and coefficients of $f_i(x,a)$, $g_{ia}(t)$ were determined by sampling from uniform distributions.

We considered 2 experiment scenarios, stationary (i.e. $g_{ia}(t) = 1$), and nonstationary (i.e. $g_{ia}(t)$ is not constant). Both scenarios have 10 hypothetical items, 30-day survival as the target metric to optimize(i.e. reward), and represent a customer with a simplified 15-dim feature vector sampled from internal data. We simulated rewards over a 70-day period with 5000 incoming data points per day for each of the following policies: (1) **Random**: uniform random policy as a baseline. (2) **PHLRec-EG**: a 5% ϵ-greedy (EG) policy based on a PHLRec model which is trained daily using Adam optimizer [20] with progressive horizon learning on simulated survivals over 6 horizons 1, 3, 7, 14, 21, 30, with

corresponding daily horizon mask update. Since only 15 customer features were involved, we did not pretrain a customer feature encoder as described in previous section. Rather, the PHLRec model we considered here was simplified to have an affinity learner that consists of 2 residual blocks and takes the concatenated vector of raw customer features and item embedding as input, so compared to other methods it did not borrow strength from pretraining with extra data. Also, we chose a linear weighting scheme $\lambda_t = t$ for horizon t to encourage the daily training processes to focus more on the supervision by matured longer term outcomes when they became available, so that the policy could transition to long term optimization quickly. (3) **PHLRec-BE**: same as (2) except that EG was replaced by Boltzmann Exploration (BE) with $\eta = 500$, which was chosen to constrain its randomness. (4) **STNN-EG**: a 5% EG policy based on a neural network with the same structure as PHLRec described above, except using 30-day survival as a single target. (5) **STNN-BE**: BE policy with $\eta = 500$ for STNN. (6) **BLIP-TS-30D**: a policy based on a widely used contextual linear bandit model called Bayesian Linear Probit Regression (BLIP) [11], which predicts 30-day survival, the targeted metric to optimize, given the 15-dim customer features and recommended item as input. Our BLIP implementation assumes independent Gaussian priors on model weights and a Probit link for modeling the conditional probability of a binary reward given the inputs. For our simulations, when the BLIP model was trained on each simulated day, items were then ranked with Thompson Sampling (TS) [31], i.e. ranking based on the predicted reward probabilities per item given the model weights drawn from their posteriors. (7) **BLIP-TS-1D**: variant of **BLIP-TS-30D** that optimizes 1-day survival, an aggressive short-term proxy which minimized lead time but risked suboptimality for long term. (8) **TS-30D**: noncontextual variant of **BLIP-TS-30D**, which had an intercept in replacement of the customer features. (9) **TS-1D**: same as **TS-30D** except that it optimizes 1-day survival. Since (2) (3) (7) (9) used 1-day survival as a supervisory signal, they only required 2-day random policy warmup to kick-start model training. On the contrary, since (4) (5) (6) (8) all used 30-day survival as the single model target, they were each started with a much longer 31-day random warmup. With the exception of **Random** that required no training, each policy had its own independent feedback loop: recommendations were made with the policy, and then context-item-reward data were used for model training and policy update.

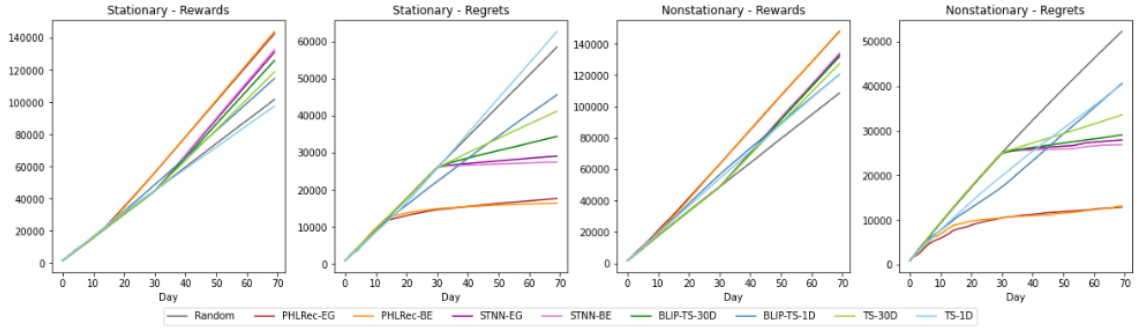

Fig. 3. Performances of policies for the stationary and nonstationary simulation experiments.

To compare these policies, we computed two metrics for each scenario: (i) cumulative rewards defined as the total 30-day survivals; (ii) (expected) cumulative regrets, where regret per data point is defined as the difference between expected reward of the best item and the item recommended. In Fig 3, we observe that **PHLRec-EG** and **PHLRec-BE** consistently outperformed the other policies in both scenarios with higher rewards and lower regrets, and their lead was more substantial in the nonstationary case. Among themselves, **PHLRec-BE** slightly outperformed **PHLRec-EG**

with lower daily regrets towards the end, which was also observed between **STNN-EG** and **STNN-BE**. Additionally, we can examine the policies more closely based on two perspectives. From personalization perspective, the plots show that personalized models were better than noncontextual ones in the long run, with a rate of lower regrets per day. Among the personalized models, **PHLRec-EG**, **PHLRec-BE**, **STNN-EG** and **STNN-BE**, the four based on deep neural networks, beat **BLIP-TS-30D** and **BLIP-TS-1D** which have a simpler linear model structure. From horizon perspective, we can see the two models optimizing 1D reward, **BLIP-TS-1D** and **TS-1D**, were eventually surpassed by the models optimizing 30D reward, despite the fact that they could benefit from 2-day random warmup to win in the short term. And even this short-term win is not guaranteed, as we see in the stationary case **TS-1D** was almost consistently worse than the **Random** baseline. Conversely, while **STNN-EG**, **STNN-BE**, **BLIP-TS-30D** and **TS-30D** showed better performances than **BLIP-TS-1D**, **TS-1D** towards the end, they were subject to long lead time of 30-day random warmup and accumulated high regrets during that period. Thanks to Progressive Horizon Learning, **PHLRec-EG** and **PHLRec-BE** were able to enjoy the advantages of both worlds. On one hand, they had a short lead time of 2-day random warmup, the same as **BLIP-TS-1D**, **TS-1D**. On the other hand, they were designed to eventually optimize the long-term 30-day survival when data become available, the same as **STNN-EG**, **STNN-BE**, **BLIP-TS-30D** and **TS-30D**. And since they automatically and gradually evolved from short to long-term optimization during the first 31 days, they accumulated relatively low regrets and started to dominate the other competitors around Day 15 or even earlier, way before any 30-day survival outcome data were available.

3.2 Training Data Reduction Through the Use of A Pretrained Encoder

To verify the effectiveness of a pretrained encoder for training data reduction, we conducted the following experiments on historical data collected from an existing recommendation task targeting member retention for a duration of 60 days, a binary metric for a membership subscription service. Holding out a fixed testing dataset for performance comparison, we experimented with different combinations of training dataset size and model structure. We trained models to predict retention probabilities for each customer at three horizons, 1, 30, 60 days, given the customer features and the recommendation, and compared their testing AUC performances.

Table 1. Testing AUC performances for membership retention metric at three horizons.

Training Data Size	Trainable Weights	Testing AUCs		
Encoder-Free		1D	30D	60D
650 k	492 k	0.845	0.840	0.831
65 k	492 k	0.750	0.747	0.730
PHLRec				
65 k	57 k	0.847	0.850	0.847
6.5 k	57 k	0.828	0.815	0.751

We considered 2 models, PHLRec and "Encoder-Free". PHLRec model contained a pretrained supervised encoder component of 5 residual blocks, and an affinity learner of 10 blocks. In contrast, Encoder-Free model was almost identical with PHLRec, except that its customer feature encoding layers were not pretrained but rather trained alongside other parts of it. As a result of this single distinction, we can observe from Table 1 that PHLRec had 88% fewer trainable weights than Encoder-Free. We started by training Encoder-Free model on a dataset of size 650 k, and logged testing

AUC performances. Next, we lowered its training data size by 90% to 65 k, which saw a significant drop of AUC by over 0.09 on all horizons (see Table 1 "Encoder-Free"). Then we replaced Encoder-Free by PHLRec and repeated the experiment by training PHLRec using the 65 k sized subsample, and observed that its AUC performances were at least as good as the Encoder-Free model trained on the original dataset of size 650 k. Lastly, for PHLRec we further reduced the training data to a smaller subsample of size 6.5 k, only 1% of the original, which still yielded serviceable performances across all horizons, even outperforming the Encoder-Free model trained on 65 k by some margin. These experiments demonstrate that with help from the dense representations learned through a pretrained encoder, training PHLRec is turned into a transfer learning task that requires considerably fewer data points than an Encoder-Free model to perform adequately with deep personalization.

3.3 Performances of PHLRec for Real World Recommendation Tasks

We tested PHLRec on two recommendation tasks that respectively target long-term subscription retention and user engagement. In the first experiment, we compared PHLRec to a BLIP policy, which produced recommendations via Thompson Sampling against a short-term surrogate metric for subscription retention, and a business-configured Control. In the second one, a PHLRec policy was compared to a Random policy and Control, as no other ML policy was set up.

The relative lifts of each policy over the respective Controls are shown in Table 2 with standard error bars included. In both experiments, the PHLRec policies yielded a statistically significant improvement above the comparison policies as well as Control, with lift values considered substantial for those applications. In Experiment 2, we also include an estimated performance lift (over Control) of a hypothetical Best-Single-Item (BSI) policy, i.e. a policy that would recommend the item with best overall performance to all customers. This number serves as an "oracle" upper bound of the best possible performance of a non-contextual bandit model, since in reality such bandit would take time to explore before they converged to an item with high confidence, not to mention the possibility of suboptimal convergence. Given this, the estimated lift for BSI policy, 2.8%, was still substantially lower than 4.1%, the lift with PHLRec, which demonstrated the efficacy of personalization offered by PHLRec.

Table 2. Policy performance comparisons in two business applications.

Relative Performance Lift over Control			
Experiment	Objective	Policy	Relative Lift
1	30D Retention	PHLRec	0.68% ± 0.12%
		BLIP	0.21% ± 0.12%
2	28D Engagement	PHLRec	4.1% ± 0.5%
		Random	2.5% ± 0.5%
		BSI	2.8% (est.)

4 CONCLUSION

We proposed PHLRec to address challenges of LTO including lead time, data collection and staleness, in the domain of personalized recommendation. PHLRec is a multi-target neural network constructed with a new technique called Progressive Horizon Learning (PHL) and pretraining of a customer feature encoder. With simulation studies, we demonstrated that PHL allows the model to be deployed quickly with minimal lead time needed to start continuous training and personalized recommendation, while still progressively moving towards long-term optimization to achieve

best overall performances. In real data experiments, we showed the use of a pretrained customer feature encoder empowers PHLRec to reach high predictive performance with 90% less training data than what is needed by an Encoder-Free model, thus making PHLRec suitable for cold-start scenarios with little data available. Finally, we also showed that PHLRec achieved significant wins in optimizing long-term metrics for two real-world recommendation tasks.

REFERENCES

[1] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems* 33 (2020), 1877–1901.

[2] Qingpeng Cai, Shuchang Liu, Xueliang Wang, Tianyou Zuo, Wentao Xie, Bin Yang, Dong Zheng, Peng Jiang, and Kun Gai. 2023. Reinforcing User Retention in a Billion Scale Short Video Recommender System. *arXiv preprint arXiv:2302.01724* (2023).

[3] Nicolò Cesa-Bianchi, Claudio Gentile, Gabor Lugosi, and Gergely Neu. 2017. Boltzmann Exploration Done Right. In *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.), Vol. 30. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2017/file/b299ad862b6f12cb57679f0538eca514-Paper.pdf

[4] Benjamin Paul Chamberlain, Angelo Cardoso, CH Bryan Liu, Roberto Pagliari, and Marc Peter Deisenroth. 2017. Customer Lifetime Value Prediction Using Embeddings. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 1753–1762.

[5] Olivier Chapelle. 2014. Modeling delayed feedback in display advertising. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. 1097–1105.

[6] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. 2016. Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems*. 7–10.

[7] Wei Chu, Lihong Li, Lev Reyzin, and Robert Schapire. 2011. Contextual Bandits with Linear Payoff Functions. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. JMLR Workshop and Conference Proceedings, 208–214.

[8] 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).

[9] Miroslav Dudík, John Langford, and Lihong Li. 2011. Doubly robust policy evaluation and learning. *arXiv preprint arXiv:1103.4601* (2011).

[10] Peter S Fader, Bruce GS Hardie, and Ka Lok Lee. 2005. "Counting Your Customers" the Easy Way: An Alternative to the Pareto/NBD Model. *Marketing science* 24, 2 (2005), 275–284.

[11] T. Graepel, C. Quinonero, T. Borchert, and R. Herbrich. 2010. Web-scale Bayesian click-through rate prediction for sponsored search advertising in Microsoft's Bing search engine. In *Proceedings of the Twenty-Seventh International Conference on Machine Learning (ICML-10),*. 13–20.

[12] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: a factorization-machine based neural network for CTR prediction. *arXiv preprint arXiv:1703.04247* (2017).

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 770–778.

[14] Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, Jürgen Schmidhuber, et al. 2001. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies.

[15] Daniel G Horvitz and Donovan J Thompson. 1952. A generalization of sampling without replacement from a finite universe. *Journal of the American statistical Association* 47, 260 (1952), 663–685.

[16] Liwei Huang, Mingsheng Fu, Fan Li, Hong Qu, Yangjun Liu, and Wenyu Chen. 2021. A deep reinforcement learning based long-term recommender system. *Knowledge-Based Systems* 213 (2021), 106706.

[17] Peter J. Huber. 1964. Robust Estimation of a Location Parameter. *The Annals of Mathematical Statistics* 35, 1 (1964), 73 – 101.

[18] Luo Ji, Qi Qin, Bingqing Han, and Hongxia Yang. 2021. Reinforcement learning to optimize lifetime value in cold-start recommendation. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 782–791.

[19] Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *2018 IEEE international conference on data mining (ICDM)*. IEEE, 197–206.

[20] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[21] Sofia Ira Ktena, Alykhan Tejani, Lucas Theis, Pranay Kumar Myana, Deepak Dilipkumar, Ferenc Huszár, Steven Yoo, and Wenzhe Shi. 2019. Addressing delayed feedback for continuous training with neural networks in CTR prediction. In *Proceedings of the 13th ACM conference on recommender systems*. 187–195.

[22] Md Tahmid Rahman Laskar, Cheng Chen, Jonathan Johnston, Xue-Yong Fu, Shashi Bhushan TN, and Simon Corston-Oliver. 2022. An auto encoder-based dimensionality reduction technique for efficient entity linking in business phone conversations. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 3363–3367.

[23] Lihong Li, Wei Chu, John Langford, and Robert E Schapire. 2010. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*. 661–670.

[24] Kazuki Miyazawa, Yuta Kyuragi, and Takayuki Nagai. 2022. Simple and effective multimodal learning based on pre-trained transformer models. *IEEE Access* 10 (2022), 29821–29833.

[25] Changhua Pei, Yi Zhang, Yongfeng Zhang, Fei Sun, Xiao Lin, Hanxiao Sun, Jian Wu, Peng Jiang, Junfeng Ge, Wenwu Ou, et al. 2019. Personalized re-ranking for recommendation. In *Proceedings of the 13th ACM conference on recommender systems*. 3–11.

[26] Lijing Qin, Shouyuan Chen, and Xiaoyan Zhu. 2014. Contextual combinatorial bandit and its application on diversified online recommendation. In *Proceedings of the 2014 SIAM International Conference on Data Mining*. SIAM, 461–469.

[27] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. 2018. Improving language understanding by generative pre-training. (2018).

[28] David C Schmittlein, Donald G Morrison, and Richard Colombo. 1987. Counting Your Customers: Who-Are They and What Will They Do Next? *Management science* 33, 1 (1987), 1–24.

[29] Georgios Theocharous and Assaf Hallak. 2013. Lifetime value marketing using reinforcement learning. *RLDM 2013* (2013), 19.

[30] Georgios Theocharous, Philip S Thomas, and Mohammad Ghavamzadeh. 2015. Ad recommendation systems for life-time value optimization. In *Proceedings of the 24th international conference on world wide web*. 1305–1310.

[31] William R Thompson. 1933. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika* 25, 3-4 (1933), 285–294.

[32] 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).

[33] Claire Vernade, Alexandra Carpentier, Tor Lattimore, Giovanni Zappella, Beyza Ermis, and Michael Brueckner. 2020. Linear bandits with stochastic delayed feedback. In *International Conference on Machine Learning*. PMLR, 9712–9721.

[34] Ruoxi Wang, Bin Fu, Gang Fu, and Mingliang Wang. 2017. Deep & cross network for ad click predictions. In *Proceedings of the ADKDD'17*. 1–7.

[35] Chi-Man Wong, Fan Feng, Wen Zhang, Chi-Man Vong, Hui Chen, Yichi Zhang, Peng He, Huan Chen, Kun Zhao, and Huajun Chen. 2021. Improving conversational recommender system by pretraining billion-scale knowledge graph. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE, 2607–2612.

[36] Chaojun Xiao, Ruobing Xie, Yuan Yao, Zhiyuan Liu, Maosong Sun, Xu Zhang, and Leyu Lin. 2021. Uprec: User-aware pre-training for recommender systems. *arXiv preprint arXiv:2102.10989* (2021).

[37] Wanqi Xue, Qingpeng Cai, Zhenghai Xue, Shuo Sun, Shuchang Liu, Dong Zheng, Peng Jiang, and Bo An. 2022. PrefRec: Preference-based Recommender Systems for Reinforcing Long-term User Engagement. *arXiv preprint arXiv:2212.02779* (2022).

[38] Chunqiu Zeng, Qing Wang, Shekoofeh Mokhtari, and Tao Li. 2016. Online context-aware recommendation with time varying multi-armed bandit. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. 2025–2034.

[39] Lixin Zou, Long Xia, Zhuoye Ding, Jiaxing Song, Weidong Liu, and Dawei Yin. 2019. Reinforcement learning to optimize long-term user engagement in recommender systems. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2810–2818.