# STAT 461 Final Project: Self-Supervised Sequential Recommendation with Hard Negative Graph Contrastive Learning

**Sharon Lin**
Northwestern University
Evanston, Illinois
sharonlin2025@u.northwestern.edu

**Faith Cramer**
Northwestern University
Evanston, Illinois
faithcramer2025@u.northwestern.edu

**Zihan Zhao**
Northwestern University
Evanston, Illinois
zihanzhao2024@u.northwestern.edu

**Jeffrey Yuan**
Northwestern University
Evanston, Illinois
jeffreyyuan2026.1@u.northwestern.edu

## Abstract

Sequential recommendation systems aim to capture dynamic user behavior to predict future interactions, often leveraging graph neural networks (GNNs) and self-supervised learning (SSL) to overcome data sparsity and behavioral noise. While SelfGNN has demonstrated strong performance by combining short-term user-item graphs with long-term sequential modeling, it does not explicitly distinguish between interacted items and semantically similar but irrelevant ones. In this work, we propose HardGNN, an extension of SelfGNN that incorporates contrastive learning with hard negative sampling to enhance the precision of short-term user representations. Specifically, we select top-K non-interacted items that are most similar to a user's embedding using cosine similarity, and apply the InfoNCE loss to pull the user embedding closer to the actual next item while pushing it away from hard negatives. We replicate the original SelfGNN results on three real-world datasets (Amazon, Gowalla, and Movielens), integrate and evaluate our HardGNN model, and conduct extensive hyperparameter tuning and ablation studies. Our results show that HardGNN improves recommendation performance on sparse datasets like Gowalla, where hard negatives provide more informative contrast signals, but underperforms on the denser media-based datasets Amazon and Movielens. These findings suggest that contrastive learning with hard negative sampling is most beneficial in domains with higher behavioral noise or limited supervision. The code repository for this work can be found at https://github.com/juanyeffrey/HardGNN.

## 1 Introduction

Recommendation systems have become essential tools for filtering the vast amount of online information, driven by evolving user behavior, increasing demand for personalization, and growing internet accessibility. They serve as a useful tool to predict users' future preferences by modeling their historical interaction sequences. Traditional methods have leveraged Graph Neural Networks (GNNs) to model high-order relations in user-item interaction graphs, enabling more expressive representations of user preferences. For example, NGCF [12] employs Graph Convolutional Networks (GCNs) to explicitly capture collaborative signals in user-item interactions, while LightGCN [5] simplifies the GCN architecture to retain only essential components, achieving improved performance

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

and efficiency. Beyond static graphs, recent work has explored dynamic GNNs for recommendation tasks.

Sequential recommendation systems have emerged as a focal area within recommendation research, aiming to capture users' temporal interaction patterns to accurately predict their future behaviors. Recent advances have turned to self-supervised learning (SSL) to alleviate the limitations of sparse supervision in recommendation systems, such as the RNN-based approach GRU4Rec [6, 10] or attention-based approaches like SASRec [7]. However, (1) effectively modeling collaborative short-term behaviors across users, which are often fragmented and sparse; and (2) mitigating noise in real-world short-term interactions, which may arise from transient intent, misclicks, or superficial engagement signals remains a challenge.

To address these issues, Liu et al proposes SelfGNN [8], a self-supervised framework that captures multi-resolution temporal dynamics through short-term user-item graphs and long-term sequential encoders. It is built upon three key paradigms: (1) It learns short-term user behavior via time-segmented interaction graphs and GNNs. (2) It captures long-term user preferences through multi-level sequential encoders with interval fusion. (3) It denoises short-term representations using long-term signals for supervision. The key contributions proposed by the author are summarized as follows:

- SelfGNN captures dynamic user interests by combining periodic collaborative pattern learning with attentive sequential modeling. It incorporates a personalized self-augmented learning mechanism to denoise sparse short-term interactions and adaptively adjusts to individual user preferences.

**Our Contribution** The authors of the SelfGNN paper highlight that future extensions of SelfGNN should aim to more accurately capture short-term behavioral characteristics in order to improve recommendation performance. While the existing framework reduces user-specific noise by aligning short- and long-term signals, it does not explicitly distinguish between items that users interact with and those that are semantically similar but ultimately irrelevant.

To address this limitation, we extend the SelfGNN framework by integrating contrastive learning with hard negative sampling [9]. Specifically, we select the top-K non-interacted items that are most similar to a user's short-term embedding, computed using cosine similarity, to prioritize semantically informative negatives over random ones. This encourages the model to learn more robust and precise decision boundaries, effectively filtering out misleading short-term signals and improving recommendation accuracy under behavioral noise.

We will refer to the new model as HardGNN and our key contributions can be summarized as follows:

- We used contrastive learning, specifically hard negative sampling, to train the model to distinguish between relevant and similar but irrelevant items, enhancing the precision of short-term user representations.
- Extensive experiments on real-world datasets demonstrate that our enhanced HardGNN outperforms baseline models in some scenarios. Its consistent performance across diverse settings demonstrates the practical effectiveness and strong generalization ability of our framework.
- We conducted ablation studies to analyze the isolated impact of individual hyperparameter settings related to hard negative sampling on recommendation performance.

## 2 Problem Definition

In sequential recommendation tasks, it is essential to simultaneously capture high-order collaborative relations and temporal preference dynamics. Graph neural networks (GNNs) are effective at modeling multi-hop dependencies across users and items, and sequential models are adept at learning temporal shifts in user interests. To unify these strengths, the author considers the problem of predicting future user-item interactions by leveraging both structural and temporal signals in user behavior data, augmented with self-supervised learning (SSL) and contrastive learning techniques to enhance learning ability.

**Our HardGNN** combines the recommendation loss from the original SelfGNN framework with our proposed InfoNCE contrastive loss. This multi-task learning approach allows the model to simultaneously optimize recommendation accuracy and the quality of the learned embedding space. The final loss function, $L_{\text{Total}}$, is a weighted sum of four components: the binary cross-entropy loss for the main recommendation task ($L_{\text{rec}}$), a self-supervised loss across different graph views ($L_{\text{sal}}$), an L2 regularization term ($L_{\text{reg}}$), and the InfoNCE contrastive loss ($L_{\text{InfoNCE}}$).

**Definition 1** (Problem Setting)**.** *Let $\mathcal{U} = \{u_1, u_2, \ldots, u_I\}$ denote the set of users and $\mathcal{V} = \{v_1, v_2, \ldots, v_J\}$ denote the set of items. We are given a temporal sequence of user-item interaction graphs $\{\mathcal{A}_t\}_{t=1}^{T}$, where each $\mathcal{A}_t \in \mathbb{R}^{I \times J}$ represents interactions occurring within the $t$-th time interval. For each $\mathcal{A}_{t,i,j}$:*

$$\mathcal{A}_{t,i,j} = \begin{cases} 1, & \textit{if user } u_i \textit{ interacted with item } v_j \textit{ during time period } t \\ 0, & \textit{otherwise} \end{cases}$$

*The overall time range $[t_b, t_e]$ is uniformly divided into $T$ intervals, where $T$ is a tunable hyperparameter. Each interval captures user behavioral signals at a particular temporal granularity.*

*Given this setup, the goal is to predict the future interaction matrix $\hat{\mathcal{A}}_{T+1}$ for the next time period $(T+1)$, using both short-term sequential signals and long-term user preference representations.*

**Definition 2** (Target Formulation)**.** *The future interaction prediction task is formulated as the following joint optimization problem:*

$$\min_{\Theta_f, \Theta_g} \mathcal{L}_{rec}(\hat{\mathcal{A}}_{T+1}, \mathcal{A}_{T+1}) + \beta \mathcal{L}_{sal}(\mathbf{E}_s, \mathbf{E}_l) + \mathcal{L}_{reg} + \lambda \mathcal{L}_{infoNCE}(e_i^{(u)}, e_i^{(u)+}, \mathbf{e}_j^{(u)-})$$

*where:*

- *$\mathcal{L}_{rec}$ is a supervised loss function that measures the discrepancy between the predicted matrix $\hat{\mathcal{A}}_{T+1}$ and the ground-truth $\mathcal{A}_{T+1}$,*
- *$\mathcal{L}_{sal}$ is a self-supervised loss that encourages alignment between the short-term embedding matrix $\mathbf{E}_s$ and the long-term embedding matrix $\mathbf{E}_l$.*
- *$\mathcal{L}_{reg}$ is defined as the L2-regularized loss to control overfitting.*
- *The contrastive loss is defined using the InfoNCE Loss between the current sample $\mathbf{e}_i^{(u)}$, the next positive sample $\mathbf{e}_i^{(u)+}$ and the previous top-j negative samples $\mathbf{e}_j^{(u)-}$.*

*The prediction is computed as:*

$$\hat{\mathcal{A}}_{T+1} = f(\mathbf{E}_l, \mathbf{E}_s; \Theta_f; e), \quad \mathbf{E}_l, \mathbf{E}_s = g(\{\mathcal{A}_t\}_{t=1}^{T}; \Theta_g) \quad \{\mathbf{e}_i^{(u)}, \mathbf{e}_i^{(u)+} \mathbf{e}_j^{(u)-}\} \subseteq e$$

*Here, $g$ is an encoder that extracts temporal graph features, and $f$ is a prediction function that fuses short- and long-term signals. Parameters $\Theta_g$ and $\Theta_f$ are learned jointly during training.*

## 3 SelfGNN Methodology

Our methodology consists of two main components. The first focuses on reproducing the SelfGNN framework proposed by Liu et al. [8], shown in Figure 1. The second presents our extension, which incorporates a hard negative sampling strategy and contrastive learning into the model in detail.

### 3.1 SelfGNN Architecture

To model the evolving nature of user interests, our approach jointly leverages interval-level collaborative graph representations, covering both short-term and long-term dynamics, and fine-grained instance-level sequential modeling. Below, we detail the mathematical formulations of each module.

### 3.1.1 Short-Term Collaborative Graph Encoding

To capture recent collaborative behavior, user-item interaction graphs are constructed for each interval $t$. Inspired by the LightGCN [5], the model employs a simplified GCN to derive user and item embeddings:

$$z_{t,i}^{(u)} = \sigma(A_{t,i,*}E_t^{(v)}), \quad z_{t,j}^{(v)} = \sigma(A_{t,j,*}E_t^{(u)}) \tag{1}$$

where $z_{t,i}^{(u)}, z_{t,j}^{(v)} \in \mathbb{R}^d$ denote the information aggregated from neighboring nodes to the central nodes $u_i$ and $v_j$, and $\sigma(\cdot)$ denotes the LeakyReLU activation function. At the $l$-th layer, the **message-passing** mechanism is formally defined as follows:

$$e_{t,i,l}^{(u)} = z_{t,i,l}^{(u)} + e_{t,i,l-1}^{(u)}, \quad e_{t,j,l}^{(v)} = z_{t,j,l}^{(v)} + e_{t,j,l-1}^{(v)} \tag{2}$$

where $e_{t,i,l}^{(u)}, e_{t,j,l}^{(v)} \in \mathbb{R}^d$ represent the embeddings of user $u_i$ and item $v_j$ at the $l$-th GNN layer for the $t$-th time interval.

### 3.1.2 Multi-Level Long-Term Sequence Modeling

To model long-term user interests, we adopt a two-level approach that integrates both temporal and fine-grained behavioral dynamics: (1) interval-level sequential pattern modeling incorporates short-term features into long-term embeddings using temporal attention to capture transitions across time periods; and (2) instance-level sequential pattern modeling directly learns pairwise relationships between item instances to represent detailed user preferences.

**Interval-Level Sequential Modeling**: To model evolving preferences, Gated Recurrent Unit (GRU) [1] networks are used to process the short-term embeddings across intervals. For each user $u_i$ and item $v_j$, we define their chronological embedding sequences as:

$$S_i^{\text{intv}} = (\mathbf{h}_{1,i}^{(u)}, \ldots, \mathbf{h}_{T,i}^{(u)}), \quad S_j^{\text{intv}} = (\mathbf{h}_{1,j}^{(v)}, \ldots, \mathbf{h}_{T,j}^{(v)}) \tag{3}$$

Each hidden state is updated using the GRU:

$$\mathbf{h}_{t,i}^{(u)} = \text{GRU}(\mathbf{e}_{t,i}^{(u)}, \mathbf{h}_{t-1,i}^{(u)}), \quad \mathbf{h}_{t,j}^{(v)} = \text{GRU}(\mathbf{e}_{t,j}^{(v)}, \mathbf{h}_{t-1,j}^{(v)}) \tag{4}$$

To extract temporal dynamics across intervals, the model applies a multi-head self-attention mechanism:

$$\bar{\mathbf{H}}_i^{(u)} = \text{Self-Att}(S_i^{\text{intv}}), \quad \bar{\mathbf{H}}_j^{(v)} = \text{Self-Att}(S_j^{\text{intv}}) \tag{5}$$

**Instance-Level Sequential Modeling**: To model fine-grained user preferences, SelfGNN applies a multi-layer self-attention network over the sequence of item instances that each users has interacted with. For a user $u_i$, we define the interaction sequence as:

$$S_{i,0}^{\text{inst}} = \left(\bar{e}_{v_{i,1}}^{(v)} + p_1, \ldots, \bar{e}_{v_{i,M}}^{(v)} + p_M\right) \tag{6}$$

where $\bar{e}_{v_{i,m}}^{(v)} \in \mathbb{R}^d$ is the embedding of the $m$-th interacted item and $p_m \in \mathbb{R}^d$ is the learnable positional embedding for position $m$.

The sequence is processed through $L_a$ layers of self-attention with residual connections:

$$S_{i,l}^{\text{inst}} = \sigma(\text{Self-Att}(S_{i,l-1}^{\text{inst}})) + S_{i,l-1}^{\text{inst}} \tag{7}$$

The instance-level user embedding is obtained by summing the representations from the last layer:

$$\tilde{e}_i^{(u)} = \sum S_{i,L_a}^{\text{inst}} \tag{8}$$

## 3.2 Multi-view Aggregation and Prediction

The user's final embedding combines interval-level ($\bar{e}_i^{(u)}$) and instance-level ($\tilde{e}_i^{(u)}$) views:

$$\hat{e}_i^{(u)} = \bar{e}_i^{(u)} + \tilde{e}_i^{(u)}, \quad \hat{A}_{T+1,i,j} = \hat{e}_i^{(u)\top} \cdot \bar{e}_j^{(v)} \tag{9}$$

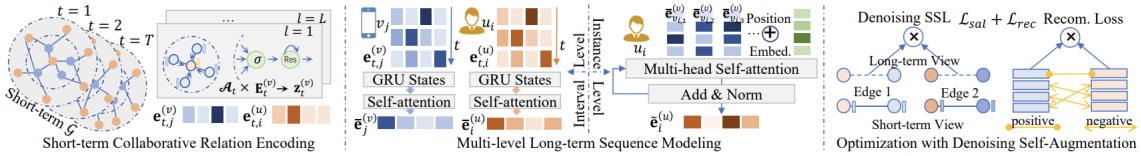



Figure 1: Overall framework of the proposed SelfGNN model

### 3.3 Personalized Self-Augmented Learning

To denoise noisy behaviors, a self-supervised contrastive loss is introduced to align short- and long-term predictions. User-specific weights $w_{t,i}$ reflect their reliability:

$$\mathcal{L}_{sal} = \sum_{i=1}^{N} \sum_{t=1}^{T} \max\left(0, 1 - d_1 \cdot d_2\right) \tag{10}$$

## 4 HardGNN Methodology

### 4.1 Implement Hard Negative Sampling

In order to improve the discriminative power of our model, we employed a hard negative sampling strategy during model training. The goal of this sampling strategy is to select negative items that are challenging for the model to distinguish from positive items, based upon their respective representations. Doing so encourages the model to learn more robust user and item representations. For a given user $u$ who has interacted with a set of positive items $i$, our goal is to find a set of hard negative items $j$ that the user has not interacted with, but are semantically close to their preferences.

The hard negative sampling process consists of three steps. First, we obtain the learned embeddings for the user and all candidate items. The user's preference is captured by a short-term user embedding $e_i^{(u)}$, which is derived from their sequence of recent interacted items using a multi-head self-attention mechanism. To model long-term user preference representations, a history consisting of up to 200 of the most recent interactions was used. To model short-term preference representations, a history of the 50 most recent interactions was used. The negative item embeddings $e_j^{(u)}$ are learned through the GNN component of our model. Second, we find the similarity between the user's preference embedding $e_i^{(u)}$ and the embedding of every negative item $e_j^{(u)}$. This similarity is computed using the cosine similarity metric, as defined below:

$$\text{sim}(e_i^{(u)}, e_j^{(u)}) = \frac{e_i^{(u)} \cdot e_j^{(u)}}{\|e_i^{(u)}\| \|e_j^{(u)}\|} \tag{11}$$

This metric quantifies the similarity between all the negative items with the user's learned preferences, where higher values indicate higher similarity. Finally, we select the top-K hardest negatives. All the negative items are ranked based on their similarity scores. Any item that the user has previously interacted with are excluded. The top-K items are then selected as the hard negatives for the current training instance. By training on these challenging examples, the model is forced to learn finer-grained distinctions between items, leading to improved recommendation accuracy.

### 4.2 InfoNCE Contrastive Loss

The original SelfGNN framework reduces user-specific noise through temporal and collaborative modeling. However, it does not explicitly distinguish between items that users interact with and those that are highly similar but ultimately irrelevant. To address this limitation, we augment the model with a contrastive learning objective based on the previous hard negative sampling, in order to enhance the discriminative capacity of short-term user representations. We specifically employ the InfoNCE loss [11] to encourage the alignment between a user's current preference representation $e_u$

and the embedding of the next interacted item $e_i$, while pushing it away from the top-$K$ most similar but non-interacted (hard negative) items $e_j$. The contrastive loss is formulated as:

$$\mathcal{L}_{\text{InfoNCE}}(e_i^{(u)}, \mathbf{e}_i^{(u)+}, \mathbf{e}_j^{(u)-}) = -\log \frac{\exp(\text{sim}(e_i^{(u)}, \mathbf{e}_i^{(u)+})/\tau)}{\exp(\text{sim}(e_i^{(u)}, \mathbf{e}_i^{(u)+})/\tau) + \sum_{j=1}^{N} \exp(\text{sim}(e_i^{(u)}, \mathbf{e}_j^{(u)-})/\tau)} \tag{12}$$

The variables of this loss function are defined as follows:

- Anchor ($e_i^{(u)}$): This is the user's preference embedding for item i, $e_i^{(u)}$, which represents the users current interests based on their previous sequence of interactions.
- Positive Key ($\mathbf{e}_s^{(u)+}$): This is the embedding of the next positive item that the use interacted with, $e_i$.
- Negative Keys ($\mathbf{e}_j^{(u)-}$): These are the embeddings of the hard negative items selected in the previous section, $e_j$.
- Temperature ($\tau$): A hyperparameter that controls the sharpness of the distribution. Lower values of $\tau$ increase the penalty of the hard negatives, making the optimization more focused on these challenging negatives.

The numerator calculates the similarity between the user preference (anchor) and their respective positive item, while the denominator represents the sum of the similarities between the user and the positive and hard negative items in the set. Minimizing this negative log-likelihood, the model learns to maximize the similarity of the positive items relative to all negative items, learning a more discriminative representations.

### 4.3 Final Learning Objective

The complete learning objective for HardGNN combines the recommendation loss from the original SelfGNN framework with our proposed InfoNCE contrastive loss. This multi-task learning approach allows the model to simultaneously optimize recommendation accuracy and the quality of the learned embedding space. The final loss function, $L_{\text{Total}}$, is a weighted sum of four components: the binary cross-entropy loss for the main recommendation task ($L_{\text{BCE}}$), a self-supervised loss across different graph views ($L_{\text{SSL}}$), an L2 regularization term ($L_{\text{reg}}$), and the InfoNCE contrastive loss ($L_{\text{InfoNCE}}$). The full objective is formulated as:

$$\mathcal{L}_{\text{Total}} = \mathcal{L}_{\text{REC}} + \beta \cdot \mathcal{L}_{\text{SSL}} + \mathcal{L}_{\text{reg}} + \lambda \cdot \mathcal{L}_{\text{InfoNCE}} \tag{13}$$

Where:

- $L_{\text{REC}}$ is the primary binary cross-entropy loss, that evaluates the model's ability to predict user-item interactions correctly.
- $L_{\text{SSL}}$ is the self-supervised loss inherited from SelfGNN, that ensures consistency across different relational views of the data.
- $\beta$ is the hyperparameter controlling the weight of $L_{\text{SSL}}$.
- $L_{\text{reg}}$ is the standard L2 regularization term prevent overfitting.
- $L_{\text{InfoNCE}}$ is the contrastive loss component described above.
- $\lambda$ is the hyperparameter controlling the weight of the $L_{\text{InfoNCE}}$.

## 5 Evaluation

Our experiments are designed to answer the following questions:

**RQ1**: How does HardGNN perform with resepect to top-k recommendation as compared with SelfGNN?

**RQ2**: How do contrastive loss weight and number of hard negative samples impact HardGNN?

### 5.1 Experimental Datasets

The experiments were conducted on three open-source datasets.

- **Amazon-book** [4]: User ratings of Amazon books from 2014.
- **Gowalla** [2]: Check-in data from Gowalla, a location-based social network, from 2010.
- **Movielens** [3]: Movie ratings collected from 2002 to 2009.

While included in the original SelfGNN Paper, a dataset consisting of Yelp venue review data was omitted due to processing issues while using the provided dataset and code in the SelfGNN Github repository.

All 3 of the datasets were processed with a 5-core setting to ensure that all users and items had at least 5 interactions. The temporal train-test split was maintained from the SelfGNN paper, holding the last interaction of each user for test, the second-to-last for validation, and the remaining for training. Negative sampling was only implemented during the training process.

### 5.2 Baseline Testing

To establish a baseline for HardGNN testing, the results for Hit Rate (HR), and Normalized Discounted Cumulative Gain (NDCG) reported in the SelfGNN paper for the three datasets were first verified using the official code from the SelfGNN GitHub repository. In order to ensure that our sampling strategy was working as expected, we isolated the contrastive loss component and measured how the model is able to distinguish the short-term anchor, its next positive item, and hard negatives with InfoNCE loss. This validated that the selected negatives were close in embedding space and that the model can learn to separate them. Once the sampling strategy was confirmed, the contrastive loss objective function was integrated into the overall training objective. After integrating the HardGNN components, the new codebase was validated by setting the number of hard negative samples to zero, effectively reverting to the original SelfGNN setup, to ensure correct implementation within the HardGNN framework. While the original SelfGNN verification involved training for 150 epochs to align with the published implementation, the validation using the new HardGNN codebase was limited to 100 training epochs due to computational constraints. This also allowed for a fair comparison with the final HardGNN results, which were obtained using the best hyperparameters selected via grid search and trained for 100 epochs. As a result, the reproduced SelfGNN baseline scores in our comparison section are slightly lower than those reported in the original paper, but they remain highly consistent with the SelfGNN paper results and validate the correctness of the HardGNN implementation. During verification, all original hyperparameter settings and configuartions remained consistent with those used by the original authors [8].

### 5.3 Hyperparameter Tuning

Two hyperparameters related to hard negative sampling were tuned using a grid search to identify the best configuration for each dataset. All other hyperparameters were kept consistent with those used in the SelfGNN paper. Specifically, three values for the number of hard negative samples were evaluated: $k \in \{3, 5, 7\}$, along with three values for the contrastive loss weight: $\lambda \in \{0.01, 0.1, 0.2\}$. Each combination was trained for up to 25 epochs per dataset, and the best-performing configuration for each dataset as measured by HR and NDCG on top-10 recommendations was selected.

- **Amazon-book**: $k = 3$, $\lambda = 0.1$
- **Gowalla**: $k = 3$, $\lambda = 0.2$
- **Movielens**: $k = 3$, $\lambda = 0.01$

### 5.4 Evaluation and Comparison

Using the best combination of k and $\lambda$ for each dataset, a HardGNN model was trained and evaluated on top-10 and top-20 recommendation tasks. Each model was trained for 100 epochs and compared against the HardGNN baseline performance (k = 0, $\lambda$ = 0). The experimental results are presented in Table 1.

Table 1: Performance Comparison of SelfGNN and HardGNN on Recommendation Tasks

| **Dataset** | **Amazon** | | | | **Gowalla** | | | | **Movielens** | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Top N** | Top 10 | | Top 20 | | Top 10 | | Top 20 | | Top 10 | | Top 20 | |
| **Metric** | HR | NDCG | HR | NDCG | HR | NDCG | HR | NDCG | HR | NDCG | HR | NDCG |
| *SelfGNN* | **0.371** | **0.218** | **0.486** | **0.250** | 0.568 | 0.370 | 0.703 | 0.407 | **0.224** | **0.121** | **0.326** | **0.145** |
| *HardGNN* | 0.223 | 0.120 | 0.401 | 0.203 | **0.624** | **0.408** | **0.736** | **0.434** | 0.206 | 0.111 | 0.308 | 0.137 |

SelfGNN outperforms HardGNN on the Amazon and Movielens datasets, while HardGNN outperforms SelfGNN on the Gowalla Dataset.

### 5.5 Results Discussion

HardGNN's superior performance on the Gowalla dataset suggests that incorporating contrastive learning through hard negative sampling into the original SelfGNN framework can improve performance in certain cases. We hypothesize that this improvement, as measured by Hit Rate (HR) and Normalized Discounted Cumulative Gain (NDCG), is observed only on the Gowalla dataset due to its distinct characteristics compared to Amazon and MovieLens. Both Amazon and MovieLens involve media data, where recommendations correspond to books and movies respectively. In contrast, Gowalla contains geolocation data, where recommendations involve predicting the next location a user might check into. Hard negatives may be comparatively more informative for geolocation checkins, as some random negatives could be extremley uniformative, corresponding to locations a user would never realistically visit, and hard negatives may offer more infomrative information that wasn't already present with only random samples. Although some random negatives may also be uninformative in the Amazon and MovieLens datasets, they are generally more meaningful, as they still represent plausible items a user might engage with. While hard negative sampling is expected to improve performance across all datasets, it may instead be confusing the model on Amazon and MovieLens datasets, as suggested by the inferior performance for these datasets.

Anther important charactherisitc to look at is sparsity, as measured by the density of the user-item interaction matrix. The Gowalla datset is the sparsest of the three, as reported in the original SelfGNN paper. This may make it more sensitive to highly informative negative samples.

To explore these hypotheses, future work could include testing on a broader range of datasets with varying characteristics, especally those involving media and geolocation data. Additionally, more extensive hyperparameter tuning could better adapt the model to the specific properties of each dataset. This may help to clarify the conditions under which hard negative sampling is beneficial.

### 5.6 Ablation Studies

The performance of HardGNN was evaluated by varying values of k while keeping $\lambda$ constant, and vice versa, to analyze the individual impact of each hyperparameter on the model's recommendation performance. These evaluations were based on the results from the previously completed grid search, with k and $\lambda$ held constant at each dataset's respective optimal value during the variation of the other parameter. Visualiztions of perfromance are displayed in Figure 2.

As $\lambda$ varies with K held constant, the performance trends differ across datasets. This highlights the importance of carefully tuning $\lambda$, and suggests that future work could benefit from exploring a wider range of $\lambda$ values.

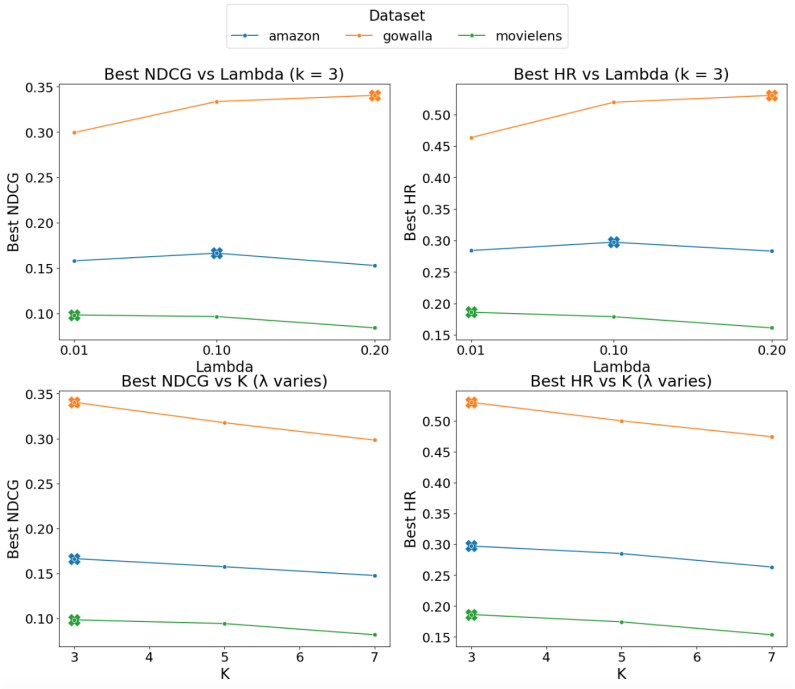
As k varies with $\lambda$ held constant, all three datasets achieved their best performance at k = 3, the lowest value included in the grid search. This is consistent with earlier observations where hard negative sampling negatively impacted performance on the Amazon and MovieLens datasets. For the Gowalla dataset, the best results were achieved with fewer hard negatives combined with a relatively higher weighting of the contrastive loss term.

A final ablation study included varying the maximum number of previous interactions used as positive samples for each user. In the main HardGNN experiments, the value of this parameter was kept at 200 for each dataset, consistent with the value used in the SelfGNN paper. This high value typically retains a very large portion of, if not all of, each user's history. To examine whether limiting the interaction history used as positive samples impacted model performance, thus focusing more on short-term information, each model was retrained with the maximum number of positive samples set

Table 2: Performance Comparison of HardGNN with Varying Positive Sample Lengths on Recommendation Tasks

| **Dataset** | **Amazon** | | **Gowalla** | | **Movielens** | |
|---|---|---|---|---|---|---|
| **Metric** | HR | NDCG | HR | NDCG | HR | NDCG |
| *HardGNN (pos_length = 200)* | **0.223** | **0.120** | **0.624** | **0.408** | 0.206 | 0.111 |
| *HardGNN (pos_length = 50)* | 0.207 | 0.111 | 0.609 | 0.402 | **0.207** | **0.112** |

Figure 2: Hard Negative Sampling Parameter Effects on top-10 Recommendation Performance




to 50 for top-10 recommendation. All other hyperparameters were kept constant. A comparison of performance is displayed in Table 2. Performance metrics remained similar, with a slight decline observed for the Amazon and Gowalla datasets. This suggests that providing broader context by incorporating both short-term and long-term user interactions when selecting hard negative samples is generally more beneficial.

## 6 Conclusion and Future Work

In this paper, we built on the framework provided in the SelfGNN paper by incorporating contrastive learning through hard negative sampling, which can help the model better distinguish meaningful signals from misleading ones. Experiments showed mixed results: the addition improved performance on the Gowalla dataset but reduced performance on the Amazon and MovieLens datasets. These initial findings are promising. Future work could focus on improving computational efficiency, as our experimentation was significantly constrained by limited computing resources. Additional areas for future work include extending the number of training epochs, further tuning the two hard negative sampling parameters and other hyperparameters introduced in the SelfGNN paper, and evaluating performance on additional datasets, particularly those involving geolocation data.

## References

[1] Mingxiao An, Fangzhao Wu, Chuhan Wu, Kun Zhang, Zheng Liu, and Xing Xie. Neural news recommendation with long-and short-term user representations. In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pages 336–345, 2019.

[2] Eunjoon Cho, Seth A Myers, and Jure Leskovec. Friendship and mobility: user movement in location-based social networks. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1082–1090, 2011.

[3] F Maxwell Harper and Joseph A Konstan. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)*, 5(4):1–19, 2015.

[4] Ruining He and Julian McAuley. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *proceedings of the 25th international conference on world wide web*, pages 507–517, 2016.

[5] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, pages 639–648, 2020.

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

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

[8] Yuxi Liu, Lianghao Xia, and Chao Huang. Selfgnn: Self-supervised graph neural networks for sequential recommendation. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1609–1618, 2024.

[9] Joshua Robinson, Ching-Yao Chuang, Suvrit Sra, and Stefanie Jegelka. Contrastive learning with hard negative samples. *arXiv preprint arXiv:2010.04592*, 2020.

[10] Yong Kiam Tan, Xinxing Xu, and Yong Liu. Improved recurrent neural networks for session-based recommendations. In *Proceedings of the 1st workshop on deep learning for recommender systems*, pages 17–22, 2016.

[11] Petar Veličković, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. Deep graph infomax. *arXiv preprint arXiv:1809.10341*, 2018.

[12] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. Neural graph collaborative filtering. In *Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval*, pages 165–174, 2019.