applied sciences

MDPI

Article

Recommendation Algorithm for Multi-Task Learning with Directed Graph Convolutional Networks

Lifeng Yin [1], Jianzheng Lu [2], Guanghai Zheng [1], Huayue Chen [3,*] and Wu Deng [4,5,*]

[1] School of Software, Dalian Jiaotong University, Dalian 116000, China
[2] School of Computer and Communication Engineering, Dalian Jiaotong University, Dalian 116028, China
[3] School of Computer Science, China West Normal University, Nanchong 637002, China
[4] Traction Power State Key Laboratory, Southwest Jiaotong University, Chengdu 610031, China
[5] School of Electronic Information and Automation, Civil Aviation University of China, Tianjin 300300, China
* Correspondence: sunnyxiaoyue20@cwnu.edu.cn (H.C.); wdeng@cauc.edu.cn (W.D.)

Abstract: As an important branch of machine learning, recommendation algorithms have attracted the attention of many experts and scholars. The current recommendation algorithms all more or less have problems such as cold start and single recommended items. In order to overcome these problems and improve the accuracy of personalized recommendation algorithms, this paper proposes a recommendation for multi-task learning based on directed graph convolutional network (referred to as MTL-DGCNR) and applies it to recommended areas for e-commerce. First, the user's micro-behavior is constructed and converted into directed graph structure data for model embedding. It can fully consider the embedding of first-order proximity nodes and second-order proximity nodes, which can effectively enhance the transformation ability of features. Secondly, this model adopts the multi-task learning method, and uses knowledge graph embedding to effectively deal with the one-to-many or many-to-many relationship between users and commodities. Finally, it is verified by experiments that MTL-DGCNR has a higher interpretability and accuracy in the field of e-commerce recommendation than other recommendation models. The ranking evaluation experiments, various training methods comparison experiments, and controlling parameter experiments are designed from multiple perspectives to verify the rationality of MTL-DGCNR.

Keywords: recommendation algorithm; directed graph convolutional network; soft attention mechanism; multi-task learning; knowledge graph embedding

Citation: Yin, L.; Lu, J.; Zheng, G.; Chen, H.; Deng, W. Recommendation Algorithm for Multi-Task Learning with Directed Graph Convolutional Networks. *Appl. Sci.* **2022**, *12*, 8956. https://doi.org/10.3390/app12188956

Academic Editor: Antonio Moreno

Received: 31 July 2022
Accepted: 2 September 2022
Published: 6 September 2022

1. Introduction

With the rapid development of the network, the recommendation system is born, and the recommendation algorithm has also become a research hot-spot. Collaborative filtering recommendation [1] and content-based recommendation [2] are two mainstream recommendation algorithms. The collaborative filtering algorithm has two modes, the neighborhood-based model (NBM) [3] and the latent factor model (LFM) [4], but the collaborative filtering algorithm is affected by key issues such as cold start, sparsity, and constraints. The content-based recommendation algorithm is based on the introduction of the commodity characteristics and the records describing the historical interests of consumers. It can determine what is most in line with consumers' preferences and push them to consumers. Like the collaborative filtering algorithm, it does not require grasping a huge customer base and reviewing records. Even just for a single consumer, it can create a personalized recommendation list for them. However, these algorithms also have limitations, such as a low efficiency, difficult semantic processing, and a low novelty of recommendation results [5–12]. In order to solve the shortcomings of collaborative filtering algorithms and content-based recommendation algorithms, and to increase the accuracy of personalized recommendation results, researchers combine these methods and use hybrid recommendation methods [13] to increase the accuracy of recommendation results.

Reference [14] fused the recommendation algorithm based on collaborative filtering and the recommendation algorithm based on content. Reference [15] used multiple models to calculate the features of users or commodities, respectively, and fused the features. In recent years, more researchers have paid attention to personalized recommendation technology based on user sequence behavior. Oren et al. [16] proposed the Item2vec method based on the user's click sequence information selected by the user. The Item2vec method is concise and effective, but it only examines the local information of the ordering and cannot examine the overall effect of the user behavior sequence on the context. In order to make up for this deficiency, Session-based Recommendation Systems [17] have emerged quietly and have received extensive attention. An et al. [18] proposed a robust method based on dynamic feature weight selection to realize the complex scenes. The other methods are also proposed to realize recommendation technology [19–24].

With the rapid development of deep learning technology and powerful representation capabilities, many conversational recommendation algorithms based on deep learning have been born. In the use of deep learning to deal with the ranking problem, Hidasi et al. [25] first proposed to apply the recurrent neural network (RNN) to the recommendation of established sequences and proposed the GRU4Rec model. Compared with the traditional solution sequence method, this kind of model has a significant improvement in effect, but it only models the single-item transfer relationship between two proximity items, ignoring the problem of other item information in the session. Long short-term memory (LSTM) and gated recurrent units (GRU) networks [26] are often combined with matrix factorization methods to input user sequences into the network, and the user's preference is represented by the hidden state of the network. Zhou et al. [27] provided a set of deep interest setwork (DIN) training modes. DIN not only provides an adaptive understanding of customer preferences from historical activities to specific advertisements through a local activation module, and also provides two practical techniques of mini-batch-aware regularization and statistical adaptive trigger function to assist in training tens of thousands of network parameters. An attention mechanism is applied in a neural network as an efficient auxiliary module to enhance the interpretability of the model. Zhou et al. [28] improved DIN and established a deep interest evolution network (DIEN) model. By adding an interest extraction layer to the network and by adding an attention mechanism in the interest evolution layer to capture the latest evolution process related to the target item, the real-time preference of the user was captured in the sequence feature representation of the target user. Tang et al. [29] proposed a recommendation model SASRec based on a sub-attention mechanism. SASRec can not only capture the long-term semantics of users like the RNN model, but also achieves recommendation with relatively few operations. Chen et al. [30] propose an item-level and component-level attention mechanism in the field of multimedia implicit feedback. Wang et al. [31] proposed an attention mechanism to learn the importance of each neighbor node, and the collaborative filtering signal was obtained by using the attention score to improve the recommendation accuracy. Pereira et al. [32] proposed a Transformers4Rec model, which also achieved excellent performance in the field of e-commerce based on the transformer architecture. Sánchez-Moreno et al. [33] proposed an approach to infer user contextual recommendations from the dynamics of social tag embeddings of music. Song et al. [34] proposed a short-session next-item recommendation based on a session-based recommendation system (SBRS). In addition, some other deep learning models are proposed, which can be applied in the recommendation [35–40].

In practical applications, many data are graph-structured, such as knowledge graphs, social networks, molecular networks, etc., followed by the emergence of graph neural networks (GNNs). Berg et al. [41] firstly applied graph convolution network (GCN) to the contact graph of visits between users and items, and proposed a GC-MC model. Wang et al. [42] deeply researched and summarized the application of graph neural networks in recommendation systems. The IGPL algorithm [43] transforms the score prediction problem into link prediction by extracting the partially closed sub-graphs of the users and items and inputting the sub-graphs into the graph neural network for training. It not

only enhances the explanatory power of the model, but also learns the deep features of each node. Wu et al. [44] proposed a session-based recommendation with graph neural networks (SR-GNN), which models all user sequences through directed graphs, and then uses GNN to learn each the hidden vector representation of the commodity. Then, the embedding of each session is obtained through the attention architecture model. Finally, the global prediction is achieved through the softmax layer. KIM et al. [45] proposed a content-rich graph neural network with attention (CoRGi). It used an attention mechanism. This algorithm considers the contextual nodes of node neighbors and gives a personalized attention mechanism as a message delivery. Experiments show that this method is better at predicting edge values in sparse regions of the graph.

The above recommendation algorithms all have the defects of cold start and single recommendation items, more or less. In order to solve the problem of only modeling the single-item transfer relationship between two proximity items, ignoring the information of other items in the session; in order to effectively deal with the one-to-many or many-to-many relationship between users and items, obtain a clearer session representation and reduce the cold start problem; in order to effectively enhance the changing ability of features, a simpler feature distribution than GCN is obtained. This paper takes advantage of a directed graph convolutional network, a soft attention mechanism, and multi-task learning, integrates their related technologies, and proposes a recommendation for multi-task learning based on directed graph convolutional network (MTL-DGCNR). The main work contributions are as follows:

(1) In order to solve the problem that the recurrent neural network (RNN) only models the single transfer relationship of two adjacent items in the recommendation field and ignores other item information in the session, the MTL-DGCNR model adopts a directed graph convolutional network.
(2) In order to effectively enhance the changing ability of features and obtain a simpler feature distribution than GCN, the MTL-DGCNR model adopts the learning embedding of the directed graph, and fully considers the embedding of the first-order proximity nodes and the embedding of the second-order proximity node.
(3) In order to effectively solve the cold start problem in recommendation and the one-to-many, many-to-one, or many-to-many relationship between users and commodities, the MTL-DGCNR model adopts a multi-task learning method, and uses knowledge graph embedding as auxiliary tasks, and cooperates the DGCNR model to complete the recommendation task.
(4) Experiments verify that the goal effect achieved by integrating two learning tasks in one MTL paradigm is better than the effect of two tasks achieving their respective tasks separately. Utilizing user micro-behavior embedding helps to better capture user preferences at a fine-grained level. Compared with other deep learning models, MTL-DGCNR has a higher interpretability and accuracy in the field of e-commerce recommendation. A more effective commodity recommendation is realized to meet the personalized needs of users.

2. Related Concepts

2.1. Directed Graph Convolutional Networks

The graph convolutional network (referred to as GCN) has been widely used for its excellent performance in processing graph result data. However, undirected graphs limit the application range of GCNs, and the directed graph convolutional networks (referred to as DGCNs) [46] are proposed for this problem. The graph convolution is extended to the directed graph by using the first-order and second-order degrees, which not only preserves the connected property of the directed graph, but also enlarges the receptive field of the convolution operation. Compared with a traditional GCN, DGCN has a stronger feature conversion ability. It can obtain simpler feature distribution than GCN, and thus facilitates the analysis of samples. DGCN takes the directed graph and item feature matrix as input in the preprocessing stage, constructs the first-order and second-order proximity, and embeds

them into the proximity convolutional layer. The node feature matrix of the first-order proximity nodes, second-order in-degree proximity, and second-order out-degree proximity after convolution operation are obtained. The node feature matrix is aggregated into the node feature map and input to the fully connected layer, and finally the model output is obtained through the softmax classifier.

2.2. TransH Algorithm

The TransE algorithm is simple, efficient, and popular in the citation of knowledge embedding, but it has certain flaws. There are certain deficiencies in dealing with one-to-many, many-to-one, many-to-many, and reflexive relationships, and the TransH algorithm [47] was proposed to make up for the deficiencies of the TransE algorithm. Given a triple <h,r,t>, the TransH algorithm introduces a hyperplane W_r for each r and a relation vector d_r on W_r, $h\perp$ and $t\perp$ are the projections of h, t on W_r, where the correct triplet is required to satisfy $h_r + d_r = t_r$. Only in this way can the difference in meaning of the same entity in different associations be determined, and the meanings of each entity in the same association are also consistent. Equivalent entities have a distributed representation, it can help the model to deal with one-to many, many-to-one, or many-to-many relational model problems.

2.3. Multi-Task Learning

Multi-task Learning (MTL) is a machine learning technique that integrates several related tasks for learning based on shared embedding. Multi-task learning is based on several related tasks with shareable embedding. When there are correlations between different learning tasks, multi-tasking can improve learning efficiency based on information sharing between tasks.

3. Multi-Task Learning Recommendation Model for Directed Graph Convolutional Networks

This section takes advantage of the respective advantages of the directed graph convolutional network, the TransH algorithm, and multi-task learning to propose a multi-task learning recommendation model for directed graph convolutional network (MTL-DGCNR), the overall framework of which is shown in Figure 1, including the DGCNR layer, the knowledge graph embedding layer, and the MTL learning layer. The DGCNR layer is further divided into an input layer, a preprocessing layer, a DGCN layer, and an output layer. In the input layer, the user's micro-behavior m_i and feature matrix X are defined through the interaction information between the user and the product in the user session; the data source of the preprocessing layer is m_i and X, and the first-order proximity and second-order proximity of each node are defined according to the generated directed graph. The first-order proximity, the second in-degree proximity, and the second-order out-degree proximity are used as the input of the DGCN layer; the DGCN layer includes the proximity convolution module, the mapping Z module, and the attention mechanism module. The proximity convolution module performs the convolution operation on the output of the preprocessing layer. The local preference Z_L of each node is obtained by Z module's aggregating, and the attention mechanism is used to assign appropriate weights to each node, and finally the global session s is generated; the output layer is composed of a multi-layer perceptron (MLP) and a softmax classifier. The global session s and the user candidate item set i_j are embedded into a multi-layer perceptron (MLP), which calculates the final score through the softmax classifier to obtain the loss value of the DGCNR model. DGCNR completes the training of commodity information and operation behaviors involved in the training samples. In the knowledge graph embedding layer, the knowledge graph of the product is defined as a triplet <entity, relationship, entity> or <entity, attribute, attribute value>, and the TransH algorithm is used for learning them. In the MTL learning layer, the knowledge graph embedding is introduced into the MTL paradigm as an auxiliary task to assist the recommendation task of DGCNR. Each part is described in detail below.

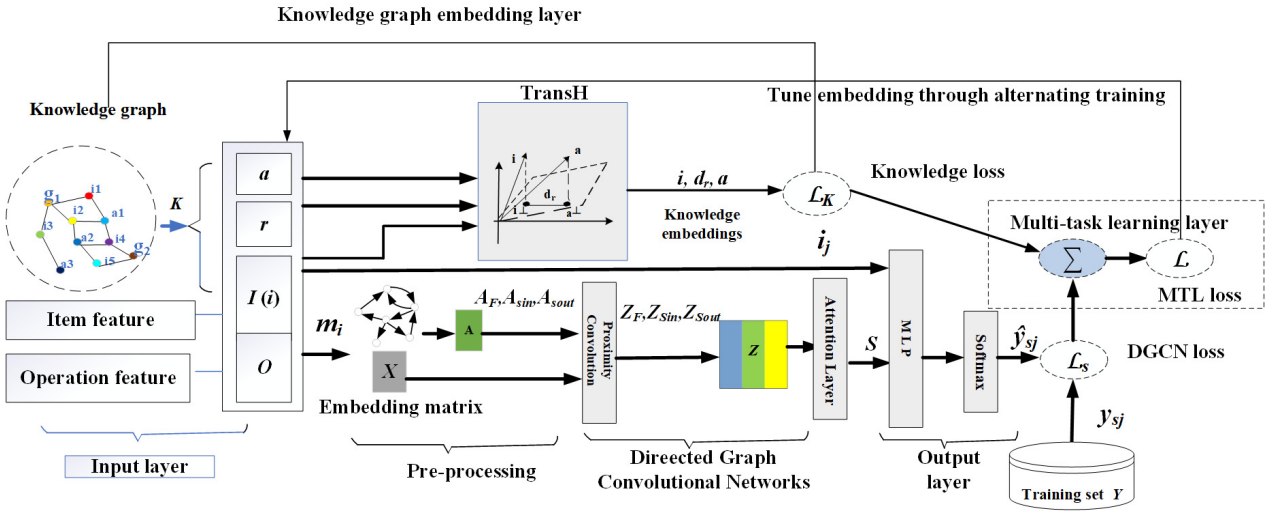

Figure 1. Overall architecture of the MTL-DGCNR model.

3.1. Input Layer of the DGCNR Layer

According to the definition of user micro-behaviors proposed by Wang et al. [47], a session of an object in a sequence is a micro-behavior, that is, a micro-behavior is a combination of an item and an operation committed to that item. First, the session sequence $s = \{m_1, m_2, ..., m_N\}$ of the user's micro-behavior is constructed, where m_i represents the user's i-th behavior and is sorted in chronological order. Each user's micro-behavior includes a user–item interaction, such as clicking, browsing, purchasing, etc. m_i can be described as a pair of $< item_i, operation_i >$, where $item_i$ represents the commodity corresponding to the i-th behavior operation, and $operation_i$ describes the specific characteristics of the operation behavior. The flow chart of user micro-behavior construction is shown in Figure 2.

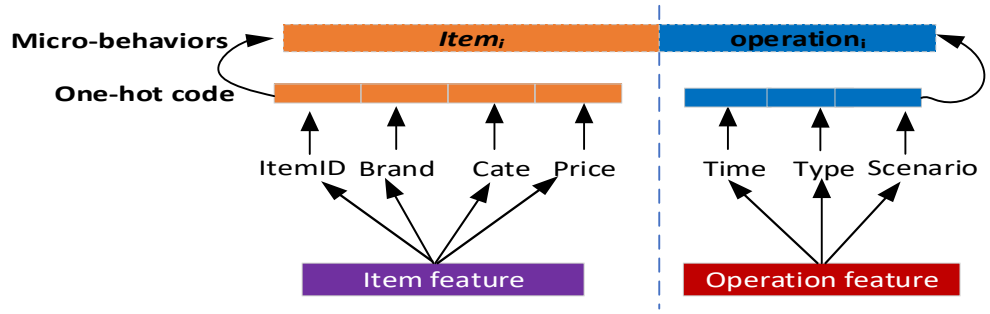

Figure 2. User micro-behavior flow chart.

The statistical feature of the commodity is defined as *item feature*, and the user's behavior attribute is defined as *operation feature*. The statistical features of commodities (*item feature*) describe the characteristics of commodities corresponding to the user's operation behavior, such as commodity ID, commodity category, commodity brand, and commodity price. The user's behavior attribute describes the user's behavior details for a certain commodity, including behavior type, behavior scene, and behavior time. Behavior types include click, browse, add to cart, favorite, and purchase. Behavioral scenarios represent scenarios in which user actions occur, such as searches or advertisements. Behavior time indicates the time corresponding to the user's current behavior. The commodity features and user behavior attribute features are modeled separately, and then connected to the user's micro-behavior. Each user behavior m_i is represented by a one-hot vector $\{m_i^1, m_i^2, \ldots, m_i^F\}$ to represent commodity features (commodity ID, commodity category, commodity brand, etc.) and behavior attributes (scene, type, time, etc.), where F is the

number of features for each behavior. All the one-hot vectors corresponding to the micro-behaviors of all users constitute the user micro-behavior feature matrix, denoted by X.

3.2. Preprocessing Layer of the DGCNR Layer

The session sequence of the user's micro-behavior is $s = \{m_1, m_2, ..., m_N\}$. Let $V = \{m_1, m_2, ..., m_N\}$ is the set of all items involved in all user session information with duplicates removed. Each session sequence s is modeled as a directed graph $G_s = (V_s, E_s)$. Each node represents an item $m_{s,i} \in V$, and each edge $(m_{s,i-1}, m_{s,i}) \in E_s$ represents the user's operation $m_{s,i}$ after the operation $m_{s,i-1}$ in session s. For example, Figure 3 describes the user behavior sequence, there are three users in total, namely U1, U2, and U3, and the corresponding user session graph is shown in Figure 4.

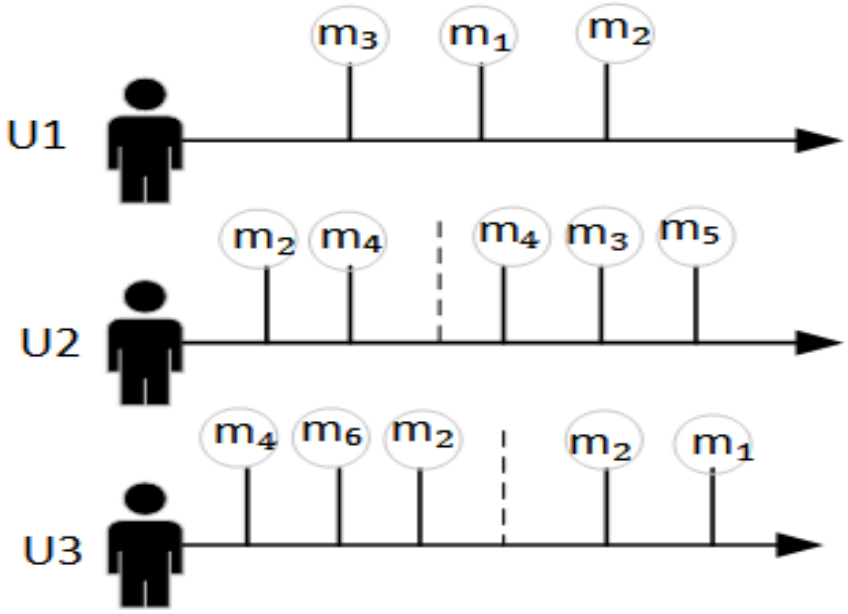

Figure 3. User behavior sequences.

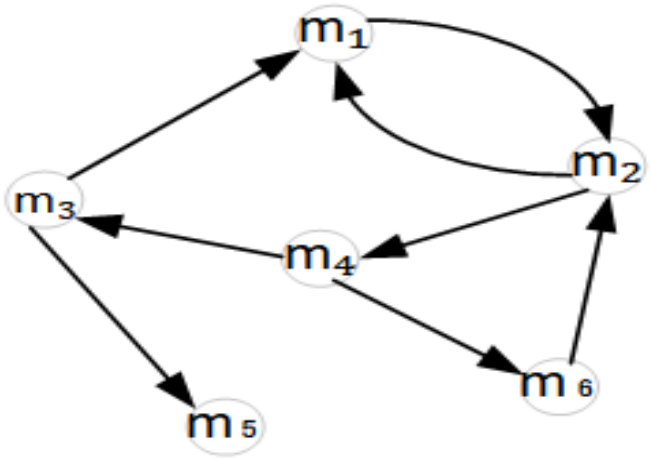

Figure 4. User session graph.

In the directed graph, we should not only consider the neighbor nodes, but also the proximity nodes of each node. In Figure 4, the definition [46] of first-order proximity nodes and second-order proximity nodes is cited, and the corresponding proximity nodes representation graph is shown in Figure 5. Node m_1 and m_2 are directly connected by an edge, and node m_2 (m_1) is the neighbor node of m_1 (m_2), as shown in Figure 5a. Node m_1 and node m_4 have no edge directly connected, but they have a common neighbor m_2, and

edges <m_2, m_1> and <m_2, m_4> exist, m_4 (m_1) is the second-order in-degree neighbor node of m_1 (m_4), as shown in Figure 5b; node m_2 and node m_6 are not directly connected by edges, but they have a common neighbor m_2, and edges <m_1, m_2> and <m_6, m_2> exist, m_6(m_1) is the second-order out-degree proximity node of m_1 (m_6), as shown in Figure 5c.

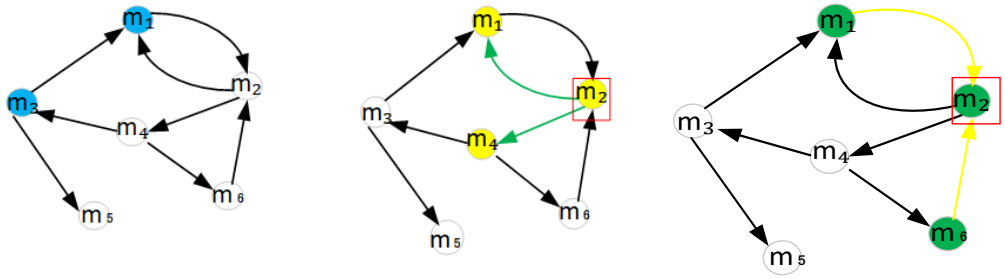

Figure 5. Proximity nodes representation graph. (**a**) First-order proximity; (**b**) In degree proximity; and (**c**) Out degree proximity.

The proximity relationship between nodes in a directed graph can be described by a matrix. The first-order proximity relationship corresponds to the first-order proximity matrix, and the second-order proximity relationship corresponds to the second-order in-degree proximity matrix and the second-order out-degree proximity matrix, they are all symmetric matrices.

The first-order proximity matrix is represented by A_F, and the value rule for the elements at (i, j) is as follows. If there is no edge from node m_i to node m_j or from node m_j to node m_i, then $A_F(i, j) = 0$. If there is only one edge from node m_i to node m_j, then $A_F(i, j) = 1$. If there is only one edge from node m_j to node m_i, then $A_F(i, j) = -1$.

There may be several items that appear repeatedly in the sequence, and a normalized weight needs to be assigned to the bidirectional edge. The elements of A_F are not limited to 0, 1, and -1, Taking the directed graph of Figure 4 as an example, the first-order proximity matrix A_F can be expressed as:

$$\begin{bmatrix} 0 & 1/2 & -1 & 0 & 0 & 0 \\ 1/2 & 0 & 0 & 1 & 0 & -1 \\ 1 & 0 & 0 & -1 & 1 & 0 \\ 0 & -1 & 1 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 & 0 \end{bmatrix}$$

The element value at (*i,j*) in the second-order in-degree proximity matrix A_{Sin} is calculated by Formula (1), and the element value at (*i,j*) in the second-order out-degree proximity matrix A_{Sout} is calculated by Formula (2), where $A_{k,i}$ represents the element at adjacent matrix *A of directed graph* and k, $m \in V_s$.

$$A_{Sin}(i, j) = \sum_k \frac{A_{k,i} A_{k,j}}{\sum_m A_{k,m}} \quad (1)$$

$$A_{Sout}(i, j) = \sum_k \frac{A_{i,k} A_{j,k}}{\sum_m A_{m,k}} \quad (2)$$

The output of the input layer is the session sequences of the user's micro-behavior, which is then input into the preprocessing layer. In the preprocessing layer, the session sequences are first converted into a directed graph, and then the first-order proximity matrix, the second-order in-degree proximity matrix, and the second-order out-degree proximity matrix are used to describe the various proximity relationships of each node in

the directed graph. The proximity matrices A_F, A_{Sin}, and A_{Sout} are used as the input data for directed graph convolutional network layer.

3.3. DGCN Layer of the DGCNR Layer

This section draws on DGCN proposed by Z Tong et al. [46] to define the learning process of the node vector in the graph. The user micro-behavior feature matrix X obtained by the input layer and the proximity matrices A_F, A_{Sin}, and A_{Sout} of the directed graph obtained by the preprocessing layer are used as the input data of the DGCN layer. The convolution operation are performed on X, A_F, A_{Sin}, and A_{Sout} is shown in the Formulas (3)–(5).

$$Z_F = \widetilde{D}_F^{-\frac{1}{2}} A_F \widetilde{D}_F^{-\frac{1}{2}} X \ominus \tag{3}$$

$$Z_{Sin} = \widetilde{D}_{Sin}^{-\frac{1}{2}} A_{Sin} \widetilde{D}_{Sin}^{-\frac{1}{2}} X \ominus \tag{4}$$

$$Z_{Sout} = \widetilde{D}_{Sout}^{-\frac{1}{2}} A_{Sout} \widetilde{D}_{Sout}^{-\frac{1}{2}} X \ominus \tag{5}$$

Among them, $\ominus$ is the linear transformation operation based on the parameters $\{W, b\}$ (there is no expression under the activation function). $\widetilde{D}_F$, $\widetilde{D}_{Sin}$, $\widetilde{D}_{Sout}$ represent the augmented matrices of the first-order degree matrix, the second-order in-degree degree matrix, and the second-order out-degree degree matrix. $\widetilde{A}_F$, $\widetilde{A}_{Sin}$, and $\widetilde{A}_{Sout}$ represent the augmented matrices of the first-order proximity matrix, the second-order in-degree proximity matrix, and the second-order out-degree proximity matrix. X represents the micro-behavior feature matrix. Then, the augmented matrix $\widetilde{A}_x$ of the proximity matrix A_x and the augmented matrix $\widetilde{D}_x$ of the degree matrix D_x are defined, as shown in Equations (6) and (7).

$$\widetilde{A}_x = A_x + \beth I \tag{6}$$

$$\widetilde{D}_x = D_x + \beth I \tag{7}$$

where $x \in \{F, Sin, Sout\}$, $\beth$ is a parameter, I is the identity matrix. The degree matrix corresponding to the proximity matrix $A_x \in R^{n \times n}$ is D_x, its calculation rule is shown in Formula (8).

$$D_x(i,i) = \sum_{j}^{n} A_x(i,j) \tag{8}$$

It can be seen that Z_F, Z_{Sin}, and Z_{Sout} not only obtain rich first-order and second-order proximity feature information, but Z_{Sin} and Z_{Sout} also retain the directed graph structure information. Based on these facts, we further design a fusion method to integrate the three signals together, so as to retain the characteristics of the directed structure while obtaining the surrounding information.

Directed graph fusion operation Γ is a signal fusion function of the first-order proximity convolution output Z_F, the second-order in-degree proximity convolution output Z_{Sin}, and the second-order out-degree proximity convolution output Z_{Sout}. L is the sum of the nodes.

$$Z_L = \Gamma(Z_F, Z_{Sin}, Z_{Sout}) \tag{9}$$

For a given user micro-behavior feature matrix X and a directed adjacency matrix A, after taking all the steps above, we can get the final directed graph result Z_L = f (X, A).

A good model should consider global preference and local preference [48]. After the convolution operation, the aggregation result Z_L of the proximity nodes of each node is obtained as the local preference of the session, in order to represent the global preference of the session and consider that the information in the node embedding may have different priorities, this paper adopts a soft-attention mechanism [49] (soft-attention) to assign

appropriate weights to each proximity node embedded in the session. Specifically, for each node, $Z_t (1 \leq t \leq L)$, its attention weight calculation formula is shown in (10).

$$a_t = \beta^{\intercal} \sigma(W_1 Z_L + W_2 Z_t + b_a) \tag{10}$$

b_a is the bias, $\beta \in R^{3d}$, W_1, $W_2 \in R^{3d \times 3d}$ are the weight parameters, then the global session representation is shown in Formula (11).

$$s_g = \sum_{t=1}^{L} a_i Z_t \tag{11}$$

a_i is the attention weight, and finally, the final representation of the session node is shown in Formula (12). Among them, $W_3 \in R^{d \times 6d}$.

$$s = W_3 \left[Z_L ; s_g \right] \in R^d \tag{12}$$

3.4. Output Layer of the DGCNR Layer

After obtaining the embedding of the global node session s, this paper embeds s and the candidate item set i_j into the MLP layer. Then, the final score $\hat{y}_{sj}$ is calculated by the softmax classifier, so the calculation formula of $\hat{y}_{sj}$ is (13).

$$\hat{y}_{sj} = softmax\left(MLP\left(s \oplus i_j \right) \right) \tag{13}$$

In order to obtain a better model representation, it must be represented by training enough training samples to represent <s,j,y_{sj}>. If m_j is the user's next interaction item after session s, then $y_{sj} = 1$, otherwise $y_{sj} = 0$. The binary cross-entropy is used as the loss function of the session recommendation task, and its loss function is defined, as shown in Formula (14).

$$\mathcal{L}_S = -\sum_{s \in S} \sum_{j \in I} \{ y_{sj} log(\hat{y}_{sj}) + (1 - y_{sj}) log(1 - \hat{y}_{sj}) \} \tag{14}$$

where S and I are the session set and the candidate set of items in the training sample.

3.5. Knowledge Graph Embedding Layer

Knowledge graph [47] is a knowledge base with a directed graph structure, and there are one-to-many, many-to-one, or many-to-many relationships between nodes. Use $G = (E, R, K)$ to represent the complete knowledge graph, where $E = \{e_1, e_2, ..., e_{|E|}\}$ represents the entity set, $R = \{r_1, r_2, ..., r_{|R|}\}$ denotes the relation set, K denotes the triplet set, $|E|$ and $|R|$ denote the number of entities and relations. In triples K = <i,r,a>, $i \in E$ denotes the head entity, $a \in E$ represents the tail entity, and $r \in R$ represents the relationship between i and a.

In order to effectively learn the many-to-one, one-to-many, or many-to-many relationships in the knowledge graph, this paper uses the TransH algorithm [47] to learn the knowledge graph embedding. TransH introduces two new definitions: hyperplane and relation vector. Given a triple <i,r,a> in the knowledge graph embedding, for each relation r, the hyperplane w_r of relation r is defined, and relation vector d_r on the hyperplane w_r is defined. The embedding of i and the embedding of a are projected onto the hyperplane w_r, respectively, denoted as $i\perp$ and $a\perp$. If $< i, r, a >$ is defined correctly, and vector $i\perp$ and $a\perp$ can be connected in a low-error manner by the translation vector d_r on the hyperplane w_r. A scoring function $\left|\left| i\perp + d_r - a\perp \right|\right|_2^2$ to measure the correctness of triples is established. Suppose $||w_r|| = 1$, $i\perp$ is expressed by w_r and i, and is shown in Formula (15). $a\perp$ is expressed by w_r and a, and is shown in Formula (16).

$$i\perp = i - w_r^{\intercal} i w_r \tag{15}$$

$$a\perp = a - w_r^{\intercal} a w_r \tag{16}$$

The loss function formula of knowledge graph embedding is shown in Formula (17).

$$\mathcal{L}_K = \sum_{<i,r,a>} ||(i - w_r^{\intercal} i w_r) + d_r - (a - w_r^{\intercal} a w_r)||_2^2 \tag{17}$$

3.6. Multi-T0061sk Learning Layer (MTL)

In MTL, the learning result of one task can be used as an auxiliary task to guide the better learning of another task. The MTL-DGCNR model combines the knowledge graph embedding task and the prediction recommendation task of DGCNR into one MTL paradigm, and the knowledge graph embedding is used as an auxiliary task to assist DGCNR completing the recommended task.

In the scenario of this paper, the goal of MTL is to maximize the posterior probability of the parameter Φ of the model given the knowledge triple K and the training set Y of DGCNR. The goal formula is obtained according to the Bayes rule as Formula (18) shown.

$$max p(\Phi|K,Y) = max \frac{p(\Phi,K,Y)}{p(K,Y)} = max p(\Phi) p(K|\Phi) p(Y|\Phi,K) \tag{18}$$

where $p(\Phi)$ is the prior probability of Φ, which is set to follow a Gaussian distribution with a mean of 0 and a standard deviation of 1. $p(K|\Phi)$ is the probability of observing K given Φ. $p(Y|\Phi,K)$ is the probability of observing Y given K and Φ, and they are defined as the product of Bernoulli distributions. The comprehensive loss function of the MTL target is shown in Equation (19).

$$\mathcal{L} = \mathcal{L}_S + \lambda_1 \mathcal{L}_K + \lambda_2 \left|\left|\Phi\right|\right|_2^2 \tag{19}$$

where $\left|\left|\Phi\right|\right|_2^2$ is the regularization term to prevent overfitting, and λ_1, λ_2 are control parameters. MTL has two training strategies of alternating training and joint training [29]. The loss function for alternate training is shown in Formula (20).

$$\begin{aligned} \mathcal{L}_{alter} = -\sum_{s\in S}\sum_{j\in I} & \left[y_{sj} log(\hat{y}_{sj}) + (1 - y_{sj}) log(1 - \hat{y}_{sj})\right] \\ & + \lambda_1 \sum_{<i,r,a>} ||(i - w_r^{\intercal} i w_r) + d_r - (a - w_r^{\intercal} a w_r)||_2^2 + \lambda_2 ||\Phi||_2^2 \end{aligned} \tag{20}$$

where S and I represent the session set and candidate item set in the training set Y, respectively. The loss function for joint training is shown in Formula (21).

$$\begin{aligned} \mathcal{L}_{joint} = \sum_{s\in S}\{-\sum_{j\in I} & \left[y_{sj} log(\hat{y}_{sj}) + (1 - y_{sj}) log(1 - \hat{y}_{sj})\right] \\ & + \lambda_1 \sum_{<i,r,a>} ||(i - w_r^{\intercal} i w_r) + d_r - (a - w_r^{\intercal} a w_r)||_2^2 + ||\Phi||_2^2 \end{aligned} \tag{21}$$

3.7. Time Complexity Analysis of MTL-DGCNR

The time spent in this model is mainly determined by DGCN, the time spent on a layer is related to the following quantities:

M represents the output feature matrix size of each convolution kernel;

K represents the side length of each convolution kernel;

C_{in} represents the number of channels of each convolution kernel, that is, the number of input channels, that is, the number of output channels of the previous layer;

C_{out} represents the number of convolution kernels that this convolution layer has, that is, the number of output channels. The output feature matrix size M is determined by the input matrix size X, the convolution kernel size K, *Padding*, and *Stride*, which are expressed as follows: $M = (X - K + 2 \times Padding)/Stride + 1$.

It can be seen that the time complexity of each convolution layer is completely determined by the square of the feature matrix output size of each convolution kernel (M^2), the area of the convolution kernel (K^2), the number of input channels (C_{in}) and the number of output channels (C_{out}). The time complexity of this layer is: $O(M^2 \times K^2 \times C_{in} \times C_{out})$.

The overall time complexity of DGCN is related to the following quantities:

D represents the depth of the neural network;

l represents the l-th convolutional layer of the neural network;

C_l represents the number of output channels C_{out} of the l-th convolutional layer, that is, the number of convolution kernels in this layer.

C_{l-1} represents the number of input channels C_{in} of the l-th convolutional layer. Then, the time complexity of the entire DGCN is: $O(\sum_{l=1}^{D} M_l^2 \times K_l^2 \times C_{l-1} \times C_l)$.

So, the time complexity of the entire model MTL-DGCNR is the same as above.

4. Experimental Results and Analysis

In this section, the datasets and experimental settings are first described, and then experiments are designed to validate the performance of the model MTL-DGCNR, and the results under different experimental settings are analyzed in detail.

4.1. Experimental Datasets

This paper uses the JData and Retailrocket e-commerce website datasets, and the JData datasets with cold start items to evaluate the MTL-DGCNR model. The JData dataset is provided by the JD.com website. It contains the user's operation behavior of items on this website within two months, including 3,708,785 behavior times of 105,180 users. Action types include clicks, purchases, reviews, add to cart, and collect. Item attributes used in the experiment include id, brand, type, and price. Retailrocket e-commerce website behavior data is real e-commerce website user behavior data, which includes the behavior data of users visiting the website within 4.5 months, including 2,756,101 behavior events of 1,407,580 users. User behavior operations are divided into three categories: click, add to cart, and purchase. Among them, there are 2,664,312 browsing behaviors, 69,332 adding shopping cart behaviors, and 22,457 purchasing behaviors.

Before the experiment starts, the datasets are briefly processed. The threshold for the duration of the sessions in the two e-commerce datasets is set to 1 h to separate different sessions. Sessions with length one and items appearing less than three times in the datasets are also filtered out. The two datasets are divided into training and test sets in chronological order, taking the first 90% of user behaviors as the training set and the last 10% of the user behaviors as the test set. In the model's test session, the model first computes the matching function for all items, and then generates a top-k recommendation list based on the scores.

In order to alleviate the cold start problem of commodities, this paper adopts a different method from the previous sequential recommendation model (SR). This paper adds an additional operation to the data set to retain the commodities that only appear in the test set, namely cold start commodities. This results in a larger proportion of cold start items in sparse datasets. In contrast to previous sequential recommendation models, these cold start items are randomly initialized with embedding, and since they do not involve any training samples, they cannot be adjusted during model training. As a result, recommendations for these items are sometimes less than ideal. The data set statistics table is shown in Table 1, where ("N") represents the data set with cold start items, and "new%" represents the proportion of behaviors designed with cold start items to all behaviors in the test set. "#" indicates the number.

Table 1. Data set statistics table.

Data	Session #	Session Length	Item # (New%)	Item Frequency	Operation #
Retailrocket	28,376	5.172	10,485	16.236	3
JData	41,852	5.372	13,454	18.186	5
JData(N)	42,600	5.383	13,909 (2.63%)	17.654	5

4.2. Compared Models

This paper compares with the following models one by one and verifies the superiority of the algorithm proposed in this paper.

FPMC [50] is a sequence recommendation method based on Markov chains and matrix factorization.

STAMP [51] captures the user's general interest and current interest in the current session using an attention network.

GRU4REC [25] uses RNN to model user sequences for session-based proposals.

NARM [52] uses an RNN with an attention mechanism to capture the user's main purpose and sequential behavior, and adds an attention mechanism to capture the user's general interest and current interest in the current session.

S-POP recommends items with the most user interactions to users, that is, recommend Top-N frequent items in the training set and the current session.

Item-KNN [53] uses cosine similarity to define the degree of similarity between sessions and recommends items that are similar to previous clicks.

SR-GNN [46] uses graph neural network to capture complex item information transformation and uses attention mechanism to capture long-term and short-term interests.

MKM-SR [34] proposes user micro-behavior, that is, the separation of commodity sequence and operation sequence of user interaction. The commodity sequence is constructed as a conversation graph and embedded in a gated graph neural network (GGNN). The operation sequence is embedded in a recurrent network of a gating mechanism (GRU) and merges the two embedding sequences into user micro-behaviors. At the same time, the knowledge graph is constructed and the TranH algorithm is used to process triples as auxiliary information to assist the recommendation of the SR model.

4.3. Experimental Parameters

For a fair comparison, the model in this paper adopts the same baseline standard, which is to set the vector embedding dimension d = 100 for the three datasets. In addition, all parameters are initialized with a Gaussian distribution with a mean of 0 and a standard deviation of 1. The DGCNR step size H is set to 1. This model adopts the Adam [54] optimizer. The learning rate is set to 0.001, and the batch size is set to 128. For other models, this paper uses their default hyper parameter settings except for the embedding dimension. The control parameter λ_1, with respect to Formula (19), is set to 0.0001, and the penalty λ_2 for the $\mathcal{L}$ is set as 10^{-5}.

4.4. Experimental Evaluation Metrics

In order to accurately evaluate the performance of each model in the recommendation task, this paper adopts three evaluation indicators commonly used in the recommendation field, which are *Recall@K*, *MRR@K*, and *AUC*. This paper sets the parameter as K = 20. The recall rate (*Recall@K*) refers to the ratio of the number of relevant results retrieved in the top K results to the number of all relevant results in the library, and it measures the recall rate of the retrieval system, the formula is shown as (22).

$$Recall@K = \frac{1}{M} \sum I_{hit} \tag{22}$$

Among them, M represents the total number of recommendations, and I_{hit} represents the indicator function. If the target item appears in the current recommendation list, the value of the indicator function is 1, otherwise it is 0.

The mean reciprocal rank (*MRR*) reflects whether the items we found are placed in a more obvious position for users, emphasizing the positional relationship and order, the formula is shown as Formula (23).

$$MRR@K = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{rank_i} \tag{23}$$

where N is the total number of users, and $rank_i$ is the position of the i-th user's real visit value in the recommendation list. If this value does not exist in the recommendation list, then $rank_i \to \infty$.

A positive sample and a negative sample are randomly given, the true representative meaning of AUC is the probability that the predicted score of the positive sample is greater than the predicted score of the negative sample. The larger the AUC, the higher the probability that the model predicts the sample to be a positive sample than the probability that the model predicts that the sample is a negative sample, the formula is shown as Formula (24).

$$AUC = \frac{\sum_{i \in PositiveClass} rank_i - \frac{M(1+M)}{2}}{M \times N} \tag{24}$$

Among them, M represents the number of positive samples. N represents the number of negative samples, and $rank_i$ represents the serial number of the i-th sample.

4.5. Evaluation Index Values Analysis

In order to prove the practical significance of the model MTL-DGCNR, this paper will carry out comparison experiments on the same eight existing recommendation models listed in Section 4.2 on three different datasets. The evaluation index values of the ten models on the three datasets obtained through experiments are shown in Table 2.

Table 2. Evaluation index values for ten models on three datasets.

Model	Retailrocket		JData		JData(N)	
	Recall@20	*MRR*@20	*Recall*@20	*MRR*@20	*Recall*@20	*MRR*@20
FPMC	0.2372	0.1104	0.3548	0.1253	0.2647	0.1454
STAMP	0.3524	0.1275	0.4165	0.1554	0.3855	0.1479
GRU4REC	0.6048	0.2982	0.7076	0.4381	0.6382	0.3115
NARM	0.3417	0.1202	0.4217	0.1596	0.3769	0.1388
S-POP	0.5036	0.2081	0.5913	0.2284	0.5458	0.2247
Item-KNN	0.0671	0.0165	0.1528	0.0955	0.1357	0.0165
SR-GNN	0.6802	0.3584	0.7242	0.4114	0.7154	0.4013
MKM-SR	0.7024	0.3692	0.7346	0.4116	0.7187	0.4117
MTL-DGCNR	0.7096	0.3788	0.7513	0.4206	0.7376	0.4155

Overall, according to the evaluation index values in Table 2, the MTL-DGCNR model proposed in this paper achieves the best results on three different datasets, which shows that the method proposed in this paper is feasible and effective.

4.6. Visualization of Experimental Results

This paper visualizes the training results of each model's *Recall*@20 in order to observe the change process of the model recall rate (*Recall@K*) more intuitively. The recall rate change curve of each model in the JData datasets is shown in Figure 6. The recall rate variation curve of the model in the Retailrocket datasets is shown in Figure 7. The recall rate variation curve of each model in the JData (N) datasets with cold start items is shown in Figure 8.

Observing Figures 6 and 7, it can be seen that the iteration speed of the four models MTL-DGCNR, MKM-SR, SR-GNN, and GRU4Rec is significantly higher than other models. This also verifies that MTL-DGCNR has a stronger representation learning ability than other models. Potential node features are continuously mined through iterations at each layer to obtain a more complete feature representation. This learning method is obviously superior to models that utilize hand-extracted features, such as Item-KNN, FPMC, NARM, etc. In addition, the late learning ability of the MTL-DGCNR model is better than other neural network hybrid models. This shows that the importance of considering the second-order proximity nodes in the directed graph. After continuous iteration, the characteristics of proximity nodes between each node are constantly enhancement. This makes it easier for samples to be classified, thereby obtaining better iterative effects.

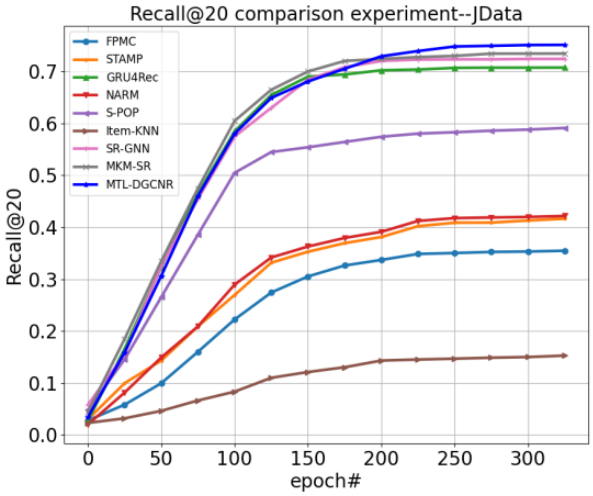

Figure 6. *Recall*@20 change graph on JData.

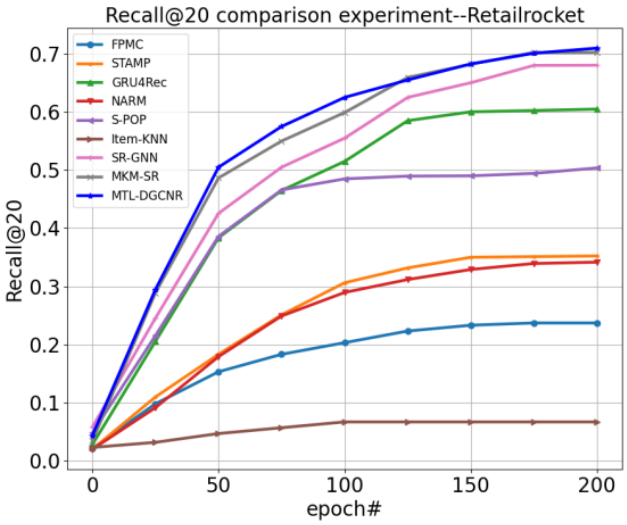

Figure 7. *Recall*@20 change graph on Retailrocket.

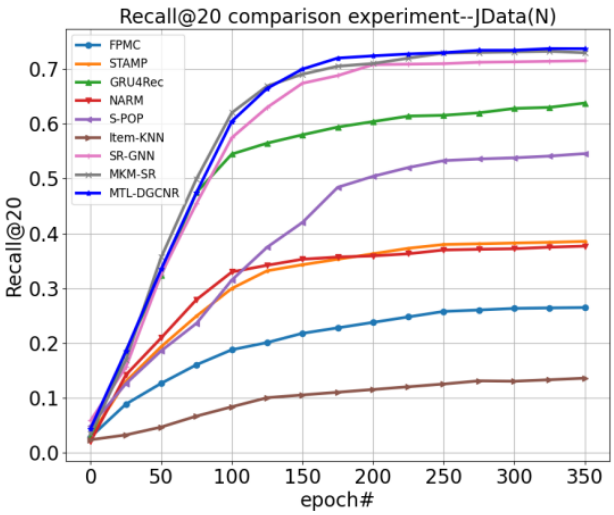

Figure 8. *Recall*@20 change graph on JData (N).

JData (N) is a dataset with cold-start items added. When dealing with data with cold-start items, it can be seen from Figure 7 that the iterative effect of MTL-DGCNR

and MKM-SR is better than that of SR-GNN. This shows that adding knowledge graph learning as auxiliary training can effectively alleviate the cold start problem. In addition, although the iteration of the GRU4Rec model is faster in the early stage, there are obvious shortcomings in the later stage. This means that the GRU model only models the single-item transfer relationship between two proximity items, easily ignoring other items in the session, and through the multi-layer neural network, the front hidden layer will affect the later hidden layer, and the propagation of the multi-layer neural network to the front hidden layer will affect the subsequent hidden layers. Therefore, it is difficult to accurately estimate each user representation from each session. In summary, the RNN model is not as effective as the GNN model in dealing with the cold start problem.

4.7. Comparative Analysis of Alternating Training and Joint Training

In order to prove the superiority of using alternating training in this paper, this paper uses the learning curves of the two training strategies to compare the applicability of the MTL-DGCNR model. In addition, this paper also includes pretrained variants for comparison. The item embedding is first pretrained by TransH, then they are input into the MTL-DGCNR model and are done parameter tuning only by the loss L_S of the equation. This paper measures the pros and cons of the three training methods through the changes of the *MRR*@20 curve on the Retailrocket and JData datasets, because *MRR*@20 can reflect a better ranking than *Recall*@20. The learning curves of the three specific training strategies are shown in Figures 8 and 9.

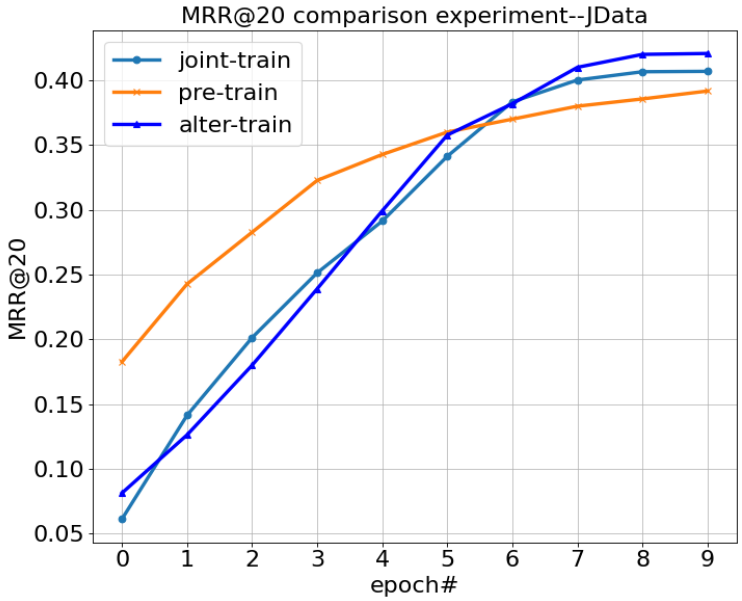

Figure 9. *MRR*@20 curve change chart—JData.

The curves in Figures 9 and 10 show that although the pretraining model achieves better results at the beginning of training, it is significantly inferior to alternating and joint training at the convergence stage. This also proves that multi-task learning is significantly superior to single-task knowledge graph pre-training. In addition, according to Formula (20), the commodities that appear multiple times in the training set session will adjust the loss L_K multiple times in each epoch of alternate training so that the knowledge learning embedding is gradually biased towards the auxiliary task L_K. This method can reduce the loss (L_S) of the main task DGCNR and improve the learning performance of the overall model.

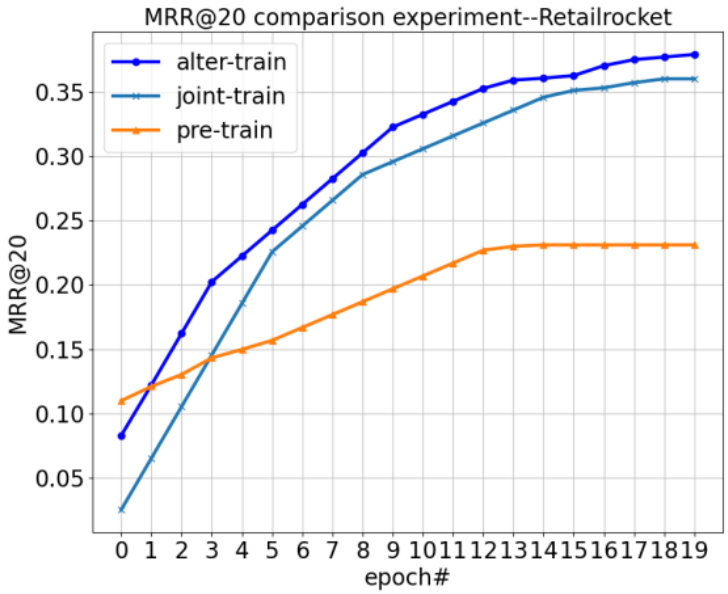

Figure 10. *MRR*@20 curve change chart—Retailrocket.

4.8. The Influence of Different Control Parameter λ_1 Values on the Model

The value of the control parameter λ_1 will have a direct impact on the MTL loss and will also affect the final recommendation result of the model. When the interval of parameter λ_1 is set at [0.00001, 0.1], the performance of the model changes very little. After continuous adjustment, this paper finds that when λ_1 = 0.0001, the MTL-DGCNR model can obtain the highest score and the model performance is optimal. The effect diagram of different values of the control parameter λ_1 is shown in Figure 11.

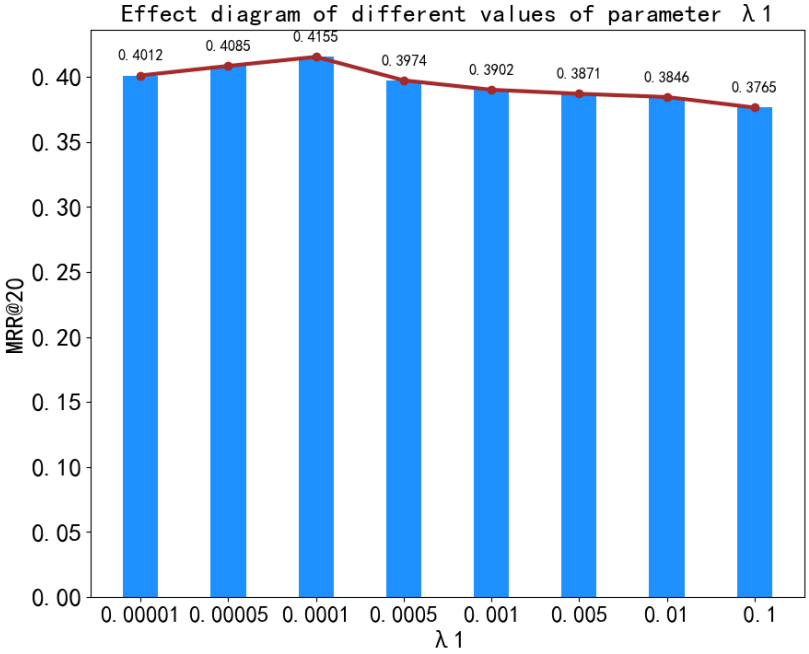

Figure 11. Effect diagram of different values of parameter λ_1.

When the value of λ_1 is too small, it means that the knowledge learning embedding as an auxiliary task is given too little weight. This results in that the auxiliary task has less influence on the main task, and the model cannot be optimized. If the value of λ_1 is too large, it means that the auxiliary task is given too much weight. This will also cause the auxiliary task to interfere with the learning of the main task, which will also cause the

model to fail to achieve optimality. Therefore, after continuous adjustment, this paper finds that when $\lambda_1 = 0.0001$, the model can reach the optimal level.

5. Conclusions

The current recommendation models all have the defects of cold start and single recommendation items more or less. This paper mainly studies a multi-task learning recommendation model with a directed graph convolution network (MTL-DGCNR) for the recommendation field of e-commerce. This model integrates related technologies such as a directed graph convolutional network, a soft attention mechanism, a TransH algorithm, and multi-task learning. The working principle of MTL-DGCNR is expounded and the superiority of this model is verified experimentally. However, there are still some deficiencies in this model, and further research is planned in two aspects in the future.

(1) In the multi-task learning mode, the data preprocessing stage not only needs to build the graph structure data of the user-commodity interaction information, but also needs to build the knowledge map of the user and the commodity. Therefore, a relatively large amount of work and parameters are undoubtedly increased in the preparatory stage. Even if multi-task learning proves to be effective in improving the performance of the model, the complexity of the model is still high. It is hoped that the complexity and parameter quantity of the model can be reduced through more in-depth research without reducing the performance of the model.

(2) In data sets with dense user behaviors or many types of user operation behaviors, the workload of the MTL-DGCNR model in processing each node and each directed edge will become larger, and the speed of model convergence may also be affected accordingly. If the gated graph sequence neural networks (GGNN) are used as the main recommendation task, and the gating mechanism is used to relatively reduce the influence of the front hidden nodes on the back hidden nodes, the convergence and performance of the model should be better. However, corresponding algorithm improvements are needed to adapt to the processing of directed graph structured data.

Author Contributions: Conceptualization, L.Y. and J.L.; methodology, L.Y.; software, J.L.; validation, LY., H.C. and J.L.; formal analysis, G.Z.; investigation, G.Z.; resources, H.C.; data curation, J.L.; writing–original draft preparation, J.L.; writing–review and editing, L.Y.; visualization, J.L.; supervision, L.Y.; project administration, W.D.; funding acquisition, L.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China under grant number U2133205 and 61771087, the Natural Science Foundation of Sichuan Province under Grant 2022NSFSC0536, the Traction Power State Key Laboratory of Southwest Jiaotong University under Grant TPL2203, the Research Foundation for Civil Aviation University of China under Grant 3122022PT02 and 2020KYQD123.

Informed Consent Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

RNN	Recurrent Neural Network
GCN	Graph Convolutional Network
GNN	Graph Neural Networks
DGCN	Directed Graph Convolutional Networks
MTL	Multi-task Learning
MTL-DGCNR	Multi-Task Learning Recommendation Model for Directed Graph Convolutional Networks

MLP	Multi-Layer Perceptron
DGCNR	Recommendation Model for Directed Graph Convolutional Networks
s	session sequence
m_i	user's i-th micro-behaviors
A_F	the first-order proximity matrix
A_{Sin}	the second-order in-degree proximity matrix
A_{Sout}	the second-order out-degree proximity matrix
$\widetilde{A}_F$	the augmented matrix of the first-order proximity matrices.
$\widetilde{A}_{Sin}$	the augmented matrix of second-order in-degree proximity matrices.
$\widetilde{A}_{Sout}$	the augmented matrix of second-order out-degree proximity matrices.
D_x	the degree matrix
$\widetilde{D}_F$	the augmented matrix of degree matrices of first-order
$\widetilde{D}_{Sin}$	the augmented matrix of degree matrices of second-order in-degree
$\widetilde{D}_{Sout}$	the augmented matrix of degree matrices of second-order out-degree.
X	the micro-behavior feature matrix
Z_L	the local preference of a session
a_t	the attention weight
s_g	the global session representation
i_j	the candidate item set
$\hat{y}_{sj}$	the final score calculated by the softmax classifier
S	the session set
I	the candidate set of items
w_r	the hyperplane
$i\perp$	the embedding of i is projected onto the hyperplane w_r
$a\perp$	the embedding of a is projected onto the hyperplane w_r
$\mathcal{L}_S$	the loss function of the session recommendation task
$\mathcal{L}_K$	the loss function of knowledge graph embedding
$\mathcal{L}$	the comprehensive loss function of the MTL target
$\mathcal{L}_{alter}$	the loss function for alternate training
$\mathcal{L}_{joint}$	the loss function for joint training

References

1. Xiao, S.; Shao, Y.; Li, Y.; Yin, H.; Shen, Y.; Cui, B. LECF: Recommendation via learnable edge collaborative filtering. *Sci. China Inf. Sci.* **2022**, *65*, 112101. [CrossRef]
2. Yu, B. Research on Recommendation System Based on Massive Data Content. Master's Thesis, Beijing University of Posts and Telecommunications, Beijing, China, 2013.
3. Shen, Y.; Jin, R. Learning personal social latent factor model for social recommendation. In Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Beijing, China, 12–16 August 2012.
4. Kumar, R.; Verma, B.K.; Rastogi, S.S. Social Popularity based SVD++ Recommender System. *Int. J. Comput. Appl.* **2014**, *87*, 33–37. [CrossRef]
5. Zhou, X.B.; Ma, H.J.; Gu, J.G.; Chen, H.L.; Deng, W. Parameter adaptation-based ant colony optimization with dynamic hybrid mechanism. *Eng. Appl. Artif. Intell.* **2022**, *114*, 105139. [CrossRef]
6. Wu, D.; Wu, C. Research on the Time-Dependent Split Delivery Green Vehicle Routing Problem for Fresh Agricultural Products with Multiple Time Windows. *Agriculture* **2022**, *12*, 793. [CrossRef]
7. Tian, C.; Jin, T.; Yang, X.; Liu, Q. Reliability analysis of the uncertain heat conduction model. *Comput. Math. Appl.* **2022**, *119*, 131–140. [CrossRef]
8. Zhao, H.M.; Liu, J.; Chen, H.Y.; Chen, J.; Li, Y.; Xu, J.J.; Deng, W. Intelligent diagnosis using continuous wavelet transform and gauss convolutional deep belief network. *IEEE Trans. Reliab.* **2022**, 1–11. [CrossRef]
9. Wei, Y.Y.; Zhou, Y.Q.; Luo, Q.F.; Deng, W. Optimal reactive power dispatch using an improved slime Mould algorithm. *Energy Rep.* **2021**, *7*, 8742–8759. [CrossRef]
10. Jin, T.; Xia, H.; Deng, W.; Li, Y.; Chen, H. Uncertain fractional-order multi-objective optimization based on reliability analysis and application to fractional-order circuit with Caputo type. *Circ. Syst. Signal Pract.* **2021**, *40*, 5955–5982. [CrossRef]
11. Deng, W.; Ni, H.C.; Liu, Y.; Chen, H.L.; Zhao, H.M. An adaptive differential evolution algorithm based on belief space and generalized opposition-based learning for resource allocation. *Appl. Soft Comput.* **2022**, *127*, 109419. [CrossRef]
12. Li, T.; Qian, Z.; Deng, W.; Zhang, D.; Lu, H.; Wang, S. Forecasting crude oil prices based on variational mode decomposition and random sparse Bayesian learning. *Appl. Soft Comput.* **2021**, *113*, 108032. [CrossRef]

13. Song, R. Research on Hybrid Recommendation Algorithms. Master's Thesis, Lanzhou University, Lanzhou, China, 2015.
14. Geetha, G.; Safa, M.; Fancy, C.; Saranya, D. A Hybrid Approach using Collaborative filtering and Content based Filtering for Recommender System. *J. Phys. Conf. Ser.* **2018**, *1000*, 012101. [CrossRef]
15. Schein, A.I.; Popescul, A.; Ungar, L.H.; Pennock, D.M. Methods and Metrics for Cold-Start Recommendations. In Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Tampere, Finland, 11–15 August 2002.
16. Barkan, O.; Koenigstein, N. Item2vec: Neural item embedding for collaborative filtering. *arXiv* **2017**, arXiv:1603.04259v2.
17. Deng, W.; Zhang, L.; Zhou, X.; Zhou, Y.; Sun, Y.; Zhu, W.; Chen, H.; Deng, W.Q.; Cheng, H.; Zhao, H. Multi-strategy particle swarm and ant colony hybrid optimization for airport taxiway planning problem. *Information Sciences*, 2022.
18. An, Z.; Wang, X.; Li, B.; Xiang, Z.L.; Zhang, B. Robust visual tracking for UAVs with dynamic feature weight selection. *Appl. Intell.* **2022**, 1–14. [CrossRef]
19. Li, T.Y.; Shi, J.Y.; Deng, W.; Hu, Z.D. Pyramid particle swarm optimization with novel strategies of competition and cooperation. *Appl. Soft Comput.* **2022**, *121*, 108731. [CrossRef]
20. Liu, Q.; Jin, T.; Zhu, M.; Tian, C.; Li, F.; Jiang, D. Uncertain currency option pricing based on the fractional differential equation in the Caputo sense. *Fractal Fract.* **2022**, *6*, 407. [CrossRef]
21. Li, G.; Li, Y.; Chen, H.; Deng, W. Fractional-order controller for course-keeping of underactuated surface vessels based on frequency domain specification and improved particle swarm optimization algorithm. *Appl. Sci.* **2022**, *12*, 3139. [CrossRef]
22. Li, X.; Zhao, H.; Yu, L.; Chen, H.; Deng, W.; Deng, W. Feature extraction using parameterized multisynchrosqueezing transform. *IEEE Sens. J.* **2022**, *2*, 14263–14272. [CrossRef]
23. Deng, W.; Li, Z.; Li, X.; Chen, H.; Zhao, H. Compound fault diagnosis using optimized MCKD and sparse representation for rolling bearings. *IEEE Trans. Instrum. Meas.* **2022**, *71*, 3508509. [CrossRef]
24. Yao, R.; Guo, C.; Deng, W.; Zhao, H.M. A novel mathematical morphology spectrum entropy based on scale-adaptive techniques. *ISA Trans.* **2022**, *126*, 691–702. [CrossRef] [PubMed]
25. Hidasi, B.; Karatzoglou, A.; Baltrunas, L.; Tikk, D. Session-based Recommendations with Recurrent Neural Networks. *arXiv* **2016**, arXiv:1511.06939v4.
26. Tang, S.; Wu, Z.; Chen, K. Movie recommendation via BLSTM. In *International Conference on Multimedia Modeling*; Springer: Cham, Switzerland, 2017.
27. Zhou, G.; Zhu, X.; Song, C.; Fan, Y.; Zhu, H.; Ma, X.; Yan, Y.; Jin, J.; Li, H.; Gai, K. Deep interest network for click-through rate prediction. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, London, UK, 19–23 August 2018.
28. Zhou, G.; Mou, N.; Fan, Y.; Pi, Q.; Bian, W.; Zhou, C.; Zhu, X.; Gai, K. Deep interest evolution network for click-through rate prediction. *Proc. AAAI Conf. Artif. Intell.* **2019**, *33*, 5941–5948. [CrossRef]
29. Tang, J.; Wang, K. Personalized Top-N Sequential Recommendation via Convolutional Sequence Embedding. In Proceedings of the 11th ACM International Conference on Web Search and Data Mining, Los Angeles, CA, USA, 5–9 February 2018.
30. Chen, J.; Zhang, H.; He, X.; Nie, L.; Liu, W.; Chua, T.S. Attentive Collaborative Filtering: Multimedia Recommendation with Item- and Component-Level Attention. *Int. Acm Sigir Conf.* **2017**, *51*, 335–344.
31. Wang, X.; He, X.; Cao, Y.; Liu, M.; Chua, T.S. Kgat: Knowledge graph attention network for recommendation. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Anchorage, AK, USA, 4–8 August 2019.
32. De Souza Pereira Moreira, G.; Rabhi, S.; Lee, J.M.; Ak, R.; Oldridge, E. *Transformers4Rec: Bridging the Gap between NLP and Sequential/Session-Based Recommendation*; ACM Conference on Recommender Systems (RecSys 21): Amsterdam, The Netherlands, 2021.
33. Sánchez-Moreno, D.; Murciego, Á.L.; Batista, V.F.L.; Vicente, M.D.M.; Moreno-García, M.N. Dynamic inference of user context through social tag embedding for music recommendation. In Proceedings of the 15th ACM Conference on Recommender Systems-Workshop on Context-Aware Recommender Systems (RECSYS 2021-CARS), Amsterdam, The Netherlands, 27 September–1 October 2021.
34. Song, W.; Wang, S.; Wang, Y.; Wang, S. Next-Item Recommendations in Short Sessions. *arXiv* **2021**, arXiv:2107.07453v2.
35. Deng, W.; Xu, J.; Gao, X.; Zhao, H. An enhanced MSIQDE algorithm with novel multiple strategies for global optimization problems. *IEEE Trans. Syst. Man Cybern. Syst.* **2022**, *52*, 1578–1587. [CrossRef]
36. Chen, H.Y.; Miao, F.; Chen, Y.J.; Xiong, Y.J.; Chen, T. A hyperspectral image classification method using multifeature vectors and optimized KELM. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2021**, *14*, 2781–2795. [CrossRef]
37. He, Z.; Shao, H.; Wang, P.; Lin, J.; Cheng, J. Deep transfer multi-wavelet auto-encoder for intelligent fault diagnosis of gearbox with few target training samples. *Knowl. Based Syst.* **2020**, *191*, 105313. [CrossRef]
38. Li, X.; Shao, H.; Lu, S.; Xiang, J.; Cai, B. Highly-efficient fault diagnosis of rotating machinery under time-varying speeds using LSISMM and small infrared thermal images. *IEEE Trans. Syst. Man Cybern. Syst.* **2022**, *50*, 1–13. [CrossRef]
39. Cao, H.; Shao, H.; Zhong, X.; Deng, Q.; Yang, X.; Xuan, J. Unsupervised domain-share CNN for machine fault transfer diagnosis from steady speeds to time-varying speeds. *J. Manuf. Syst.* **2022**, *62*, 186–198. [CrossRef]
40. Cui, H.; Guan, Y.; Chen, H. Rolling element fault diagnosis based on VMD and sensitivity MCKD. *IEEE Access* **2021**, *9*, 120297–120308. [CrossRef]
41. van den Berg, R.; Kipf, T.N.; Welling, M. Graph Convolutional Matrix Completion. *arXiv* **2017**, arXiv:1706.02263v2.

42. Wang, S.; Hu, L.; Wang, Y.; He, X.; Sheng, Q.Z.; Orgun, M.; Cao, L.; Wang, N.; Ricci, F.; Yu, P.S. Graph Learning Approaches to Recommender Systems: A Review. *arXiv* **2020**, arXiv:2004.11718.
43. Zhang, M.; Chen, Y. Inductive Graph Pattern Learning for Recommender Systems Based on a Graph Neural Network. *arXiv* **2019**, arXiv:1904.12058.
44. Wu, S.; Tang, Y.; Zhu, Y.; Wang, L.; Xie, X.; Tan, T. Session-based Recommendation with Graph Neural Networks. *arXiv* **2019**, arXiv:1811.00855v4. [CrossRef]
45. Kim, J.; Lamb, A.; Woodhead, S.; Jones, S.P.; Zhang, C.; Allamanis, M. CORGI: Content-Rich Graph Neural Networks with Attention. *arXiv* **2021**, arXiv:2110.04866v1.
46. Tong, Z.; Liang, Y.; Sun, C.; Rosenblum, D.S.; Lim, A. Directed Graph Convolutional Network. *arXiv* **2020**, arXiv:2004.13970.
47. Wang, Z.; Zhang, J.; Feng, J.; Chen, Z. Knowledge Graph Embedding by Translating on Hyperplanes. In Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, Québec City, QC, Canada, 27–31 July 2014.
48. Wu, S.; Tang, Y.; Zhu, Y.; Wang, L.; Xie, X.; Tan, T. Session-based recommendation with graph neural networks. In Proceedings of the AAAI Conference on Artificial Intelligence (AAAI-19), Honolulu, HI, USA, 27 January–1 February 2019; Volume 33, pp. 346–353.
49. Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; Bengio, Y. Graph attention network. *arXiv* **2018**, arXiv:1710.10903,2017.
50. Rendle, S.; Freudenthaler, C.; Schmidt-Thieme, L. Factorizing personalized Markov chains for next-basket recommendation. In Proceedings of the 19th International Conference on World Wide Web, Raleigh, NC, USA, 26–30 April 2010.
51. Liu, Q.; Zeng, Y.; Mokhosi, R.; Zhang, H. STAMP: Short Term Attention/Memory Priority Model for Session-based Recommendation. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, London, UK, 19–23 August 2018.
52. Li, J.; Ren, P.; Chen, Z.; Ren, Z.; Lian, T.; Ma, J. Neural Attentive Session-based Recommendation. *arXiv* **2017**, arXiv:1711.04725.
53. Linden, G.; Smith, B.; York, J. Amazon.com Recommendations: Item-to-Item Collaborative Filtering. *IEEE Internet Comput.* **2003**, *7*, 76–80. [CrossRef]
54. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster r-cnn: Towards real-time object detection with region proposal networks. *Adv. Neural Inf. Process. Syst.* **2015**, *28*, 91–99. [CrossRef]