Dynamically Constructing Relation Extraction Network for Continual Learning

Anonymous ACL submission

Abstract

Continual relation extraction aims to continuously learn new relation categories without forgetting the already learned ones. To achieve this goal, two key issues need to be addressed: catastrophic forgetting (CF) of the model and knowledge transfer (KT) of the relations. In terms of CF, there has been a great deal of research work. However, another important challenge of continual learning: knowledge transfer, has hardly been studied in the field of relation extraction. To address this, we propose dynamically constructing relation extraction networks (DCREN) for Continual relation extraction, which dynamically changes the architecture of the model through six designed actions to achieve knowledge transfer of similar relations, and further to combat catastrophic forgetting, an extensible classification module is proposed to expand the new learning space for new tasks while preserving the knowledge of old relations. Experiments show that DCREN achieves state-of-the-art performance through dynamically updating the model structure to learn new relations and transfer old knowledge.

1 Introduction

Relation extraction is an important and fundamental task in natural language processing, which aims to identify semantic relationships between entities from text. In short, relation extraction is to determine, given two or more entities, the type of relation between these entities. Relation extraction has a wide range of applications in the fields of information retrieval (Fan et al., 2022), knowledge graph construction (Ji et al., 2022), and question answering systems (Sarkar et al., 2023). It can help us understand and organize a large amount of textual information so as to provide users with more accurate and useful knowledge. However, new relations in the real world are constantly updating and changing, and traditional relation extraction methods (Tian et al., 2022; Ye et al., 2022; Zhong and Chen, 2021; Wei et al., 2020) are unable to learn new relation knowledge in real time and continuously. Therefore, many researchers (De Lange et al., 2022; Wang et al., 2022; Xia et al., 2023) have turned to continual relation extraction.

Continual relation extraction allows for continual learning of new relation categories without forgetting learned relations, aiming to continually improve the model's relation extraction capabilities. In order to ensure that relation extraction models still have excellent performance under continual learning, there are two important issues that must be taken into account: (1) catastrophic forgetting (CF) of the model, and (2) knowledge transfer (KT) of the relations.

On the one hand, in order to prevent the model from forgetting the knowledge of learned relations, i.e., CF, the simplest way is to store past data and use the historical data to retrain the model when it learns new relations. However, in practice this approach cannot be applied in reality due to computational resources and time costs. Existing work therefore focuses on storing and replaying a small number of typical samples to avoid catastrophic forgetting of the model, but due to the limitation of the number of typical samples, frequent replay can lead to overfitting.

On the other hand, catastrophic forgetting is attributed to the decline of prior knowledge with the emergence of new relations, and thus the transfer of previously learned relation knowledge is critically important, as we empirically study in Appendix A. Although knowledge transfer has been studied in the continual learning, the research on how to transfer previously learned relational knowledge to a new task network and build a continual learning model that allows for long-term learning and rapid adaptation has been limited in the field of continual relation extraction. Existing relation extraction approaches (Zhao et al., 2023; Wu et al.,

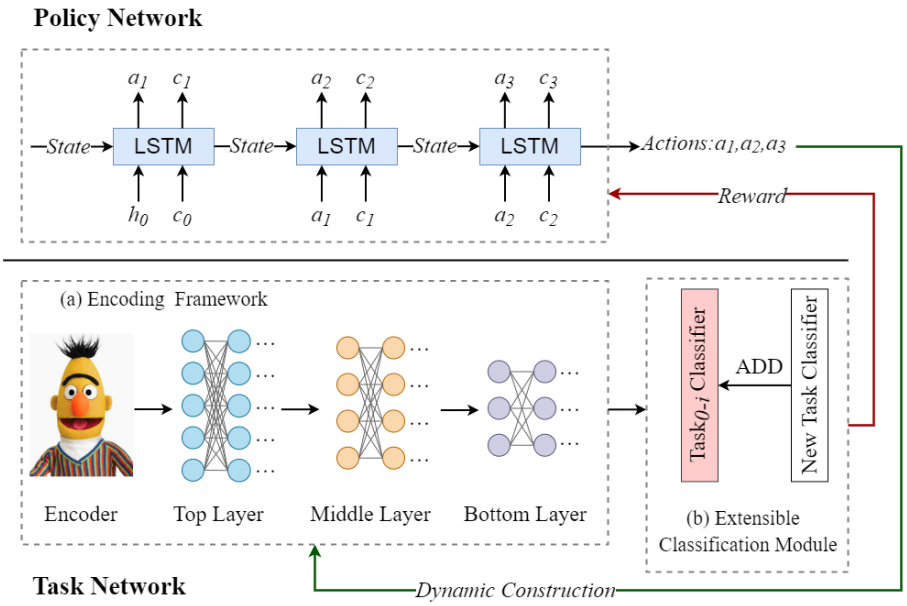

Figure 1: Overview of our proposed model.

2021; Zhao et al., 2022; Cui et al., 2021) focus on catastrophic forgetting for continual learning, ignoring the model's ability to transfer knowledge for similar tasks that have been learned in the past.

To overcome the above problems, we propose dynamically constructing relation extraction networks to enable knowledge transfer of similar relations and thus prevent catastrophic forgetting of the model. The outstanding contribution is that it can dynamically change the model architecture according to the similarity of relations in different learning tasks, so as to utilize the previous knowledge of relations to learn each task in a targeted way. Specifically, 1) we use reinforcement learning to train a policy network that dynamically constructs a corresponding task network for each task by taking a series of policy actions in order to achieve knowledge transfer of similar relations in different tasks. 2) We use a limited number of typical samples to teach the task network how to use previously learned knowledge, similar to the role of examples, with the aim of evoking the task network's learning experience and memory of old relations in order to prevent catastrophic forgetting. The sensitivity of our method to typical samples is explored in Appendix B. 3) To further enhance the memorization ability of the model and retain prior knowledge of relations, we propose an extensible classification module. Unlike previous classification modules for continual relation extraction models (Zhao et al., 2022), our extensible classification module can extend the new learning space for new relations based on the classifier for learning the old relations, instead of retraining a new classifier for the set of visible relations (old relations and new relations). 4) DCREN will create a new task network for each task, so each task network will only experience a typical sample replay, which maximizes the avoidance of model overfitting to typical samples.

Our main contributions are summarized as follows:

(1) We clarify the important role of knowledge transfer for continual relation extraction through an empirical study and propose a novel continual relation extraction network, termed DCREN, that can dynamically construct the model structure to realize knowledge transfer for similar relations.

(2) In order to enhance the model's memory capability and prevent catastrophic forgetting, we propose an extensible classification module. This module extends the new learning space for new relations based on the classifiers that have finished learning old relations, preserving the knowledge of previously learned relations.

(3) The task network dynamically constructed by DCREN only replays the typical sample once, thus minimizing the overfitting problem in continual relation extraction.

(4) The experimental results on two benchmark datasets show that our model achieves state-of-the-art accuracy compared to existing work and takes into account both catastrophic forgetting and

knowledge transfer, two important issues faced in continual learning. Our source code can be found at https://github.com/Anonymous-acl2025/DCREN.

2 Related Work

The purpose of continual learning (Wang et al., 2023) is to consistently learn new tasks while maintaining a high level of accuracy on tasks that have been learned before. The main challenges of this study are (1) avoiding catastrophic forgetting of learned knowledge while learning a new task and (2) beneficial knowledge transfer to subsequent tasks based on the experience accumulated in previous tasks. In order to solve the above problems, current research in continual learning focuses on three aspects: (a) Regularization-based methods (Li and Hoiem, 2018; Kirkpatrick et al., 2016; Adel et al., 2019; Kemker and Kanan, 2017) to limit the parameter updates of neural networks can control the balance of the model between old and new tasks to enhance the generalization ability of the model. (b) Dynamic model architecture-based methods (Véniat et al., 2020; Yoon et al., 2017; Mallya and Lazebnik, 2018; Qin et al., 2021) dynamically update the network as each new task is learned, allowing the model to efficiently integrate new knowledge while retaining old knowledge as it learns new tasks. (c) Memory-based methods (Lopez-Paz and Ranzato, 2017; Chaudhry et al., 2018; Rolnick et al., 2018; de Masson d'Autume et al., 2019) additionally equip the model with a memory repository with a fixed size storage space for important historical data. The current task data is then used to train the model in conjunction with the historical data, thus reducing catastrophic forgetting of old knowledge.

Specifically, with respect to continual relation extraction (CRE), there has been substantial research work on challenge (1) catastrophic forgetting. The memory-based models are the mainstream choice because they show better performance than other methods in avoiding catastrophic forgetting. For example, Wang et al. (2019) et al. proposed a simple memory replay method to mitigate the catastrophic forgetting problem using embedding alignment to alleviate the rapid change of embedding space in continual learning. Han et al. (2020) et al. inspired by the mechanisms of long-term memory formation in humans, introduced situational memory activation and reconsolidation into continual learning of relations to reconsolidate the prototype of old relations. Wu et al. (2021) et al. proposed a new curriculum-meta learning method to address the problems of order sensitivity and catastrophic forgetting in continual relation extraction. Cui et al. (2021) et al. use an attention-based memory network refining sample embeddings to obtain better memory prototypes and enhance performance . Zhao et al. (2022) et al. proposed a consistent representation learning method that employs contrast learning and knowledge distillation to maintain the stability of relation embeddings during replay memory. Zhao et al. (2023) et al. designed a memory-insensitive relation prototype to overcome the overfitting problem, introducing integrated training and focal knowledge distillation during training to improve similar relation performance. Recently, Le et al. (2024) et al. proposed a novel continual relation extraction approach to address the balance of the objective function of the CRE model on new and old tasks by customizing a multi-task learning framework for continual learning.

Several previous works have demonstrated that memory-based methods can be effective in avoiding model forgetting. However, memory-based models have great difficulties in addressing challenge (2) knowledge transfer due to the working mechanism of sample replay. The knowledge transfer capability is crucial for constructing continual learning models that can learn over time and adapt quickly, which is clearly ignored by existing continual relation extraction studies. In this regard, we propose the DCREN method, which dynamically constructs relation extraction models based on the similarity of different tasks to achieve knowledge transfer of similar relations, thereby preventing catastrophic forgetting of the model.

3 Task Definition

Given a sequence of relation extraction tasks $\{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_T\}$, each task $\mathcal{T}_t$ is associated with a dataset $\mathcal{D}^t = \{(x_i, y_i)\}_{i=1}^{N_t}$, where x_i represents a sentence and $y_i \in \mathcal{R}^t$ denotes a relation label from the relation space $\mathcal{R}^t$. The relation spaces of different tasks are disjoint. The goal of continual relation extraction is to learn a model that incrementally adapts to new tasks while maintaining performance on previously learned tasks. Formally, a continual relation extraction model must demonstrate the capability to detect all previously encountered relations $\tilde{\mathcal{R}}^t = \bigcup_{i=1}^{t} \mathcal{R}^i$ and will be comprehensively evaluated across the test sets of all observed tasks

$\tilde{\mathcal{D}}_{test}^{t} = \bigcup_{i=1}^{t} \mathcal{D}_{test}^{i}$.

4 Methodology

4.1 Overview

The overview diagram of our method is shown in Figure 1 and consists of two networks. The first one is the policy network $P(\cdot;\bar{\theta})$, which is in the environment of a given task and makes a policy decision based on a certain state in that environment, taking a series of actions to dynamically construct a task-specific network suitable for the task. The second one is the task network $T(\cdot;\tilde{\theta})$, which consists of two parts: the encoding framework and the extensible classification module, and the purpose of this network is to learn a new task based on memorizing the knowledge of the previous task. A similar work is BNS (Qin et al., 2021), which we discuss in detail in Appendix C.

4.2 Policy Network

The policy network will be in different task environments in different learning tasks. In general, the environment of task $\mathcal{T}_t$ contains (1) A set $\tilde{M} = \bigcup M^{r_i}$ of typical samples of relations that have appeared in all previous tasks , where M^{r_i} denotes typical samples of the stored relation r_i. (2) The task network $T(\cdot;\tilde{\theta}_{t-1})$ for the previous task $\mathcal{T}_{t-1}$. (3) The dataset $\mathcal{D}^t = \left\{\mathcal{D}_{train}^t, \mathcal{D}_{val}^t, \tilde{\mathcal{D}}_{test}^t\right\}$ for the current task $\mathcal{T}_t$. (4) The knowledge $K = \left\{\tilde{\theta}_1, \ldots, \tilde{\theta}_{t-1}\right\}$ of the relations learned by all previous task networks.

4.2.1 Agent

Our policy network $P(\cdot;\bar{\theta})$ uses a parameterized LSTM as agent, as shown in Figure 1. When learning a task t, the agent receives the state S_t in the current task environment and samples a series of actions sequentially to construct each layer of the task network $T(\cdot;\tilde{\theta})$ specific to the current task. Specifically, the agent receives the state S_t and samples the action a_i , and then a_i and the state S_t are input together to the next layer of the LSTM to sample the action a_{i+1}. Benefiting from the influence of the prior action a_i , the next action a_{i+1} sampled by the agent has sequential forward and backward dependence on the former action.

4.2.2 State

In every learning task, it usually contains N relations that need to be learned, so when learning the current task $\mathcal{T}_t$, we separately compute the similarity s_i^t between each relation r_i^t in task $\mathcal{T}_t$ and all the relations appearing in the old task, to get the state representation $S_t = \left[s_1^t, \ldots, s_N^t\right]$ of the task $\mathcal{T}_t$. Specifically, in the environment of the current task $\mathcal{T}_t$, the task network $T(\cdot;\tilde{\theta}_{t-1})$ is used to encode the training set $\mathcal{D}_{train}^t$ of the current task and all typical samples of previous tasks stored in the set $\tilde{M}$. For each visible (old task relations and current task relations) relation r_i, we compute the average of its corresponding sentence embeddings as the prototypical representation of the relation, expressed as $\hat{r}_i$. Then, the cosine function is used to compute the similarity between the relation prototype $\hat{r}_i^t$ in the current task $\mathcal{T}_t$ and the relation prototype $\hat{r}_j^{\tilde{M}}$ in the previous tasks. The formula is expressed as follows:

$$\hat{r}_i^t = \frac{1}{n_i^t} \sum_{k=1}^{n_i^t} T\left(x_{i,k}; \tilde{\theta}_{t-1}\right), \tag{1}$$

$$\hat{r}_j^{\tilde{M}} = \frac{1}{n_j^{\tilde{M}}} \sum_{k=1}^{n_j^{\tilde{M}}} T\left(m_{j,k}; \tilde{\theta}_{t-1}\right), \tag{2}$$

$$s_i^t = \frac{1}{|r|} \sum_{j=1}^{|r|} \frac{\hat{r}_i^t \cdot \hat{r}_j^{\tilde{M}}}{||\hat{r}_i^t|| \times ||\hat{r}_j^{\tilde{M}}||}, \tag{3}$$

where $x_{i,k} \in \mathcal{D}_{train}^t$, represents the k_{th} sample in the training set that belongs to relation r_i. n_i^t represents the total number of samples of relation r_i in the training set. $m_{j,k} \in \tilde{M}$, represents the k_{th} sample in the typical sample set that belongs to relation r_j. $n_j^{\tilde{M}}$ represents the total number of samples of relation r_j in the typical sample set. $|r|$ represents the total number of relations that appeared in previous tasks.

4.2.3 Action

Due to intuitive priors, the policy network has six actions to build the task network, including *"reload"*, *"fuse"*, *"add"*, *"remove"*, *"reset"*, and *"protect"*. Among them, *"reload"*, *"fuse"* and *"protect"* are categorized as Class I actions, which serve to transfer important relation knowledge for the task network, and *"add"*, *"reset"* and *"remove"* are categorized as Class II actions, which serve to add new learning space or remove redundant neural units from the task network. In conclusion, they work together to determine how to transfer relation knowledge learned from previous task networks

into the new task network or how to add new learning space to the new task network. A detailed description of each of these actions is given as follows.

The *"reload"* action reloads the parameters of the last task network, with the aim of utilizing the knowledge learned in the previous task to facilitate the learning of the new task. When the l_{th} layer of the new task network $T(\cdot;\tilde{\theta}_t)$ executes the action, the parameters of this layer will be initialized to the parameters of the l_{th} layer of the last task network, i.e., $T(\cdot;\tilde{\theta}_t^l \leftarrow \tilde{\theta}_{t-1}^l)$.

The *"fuse"* action fuses the parameters of all previous task networks, with the aim of referring to the knowledge learned in all previous tasks to help the learning of the new task. When the l_{th} layer of the new task network $T(\cdot;\tilde{\theta}_t)$ executes the action, the parameters of this layer will be initialized to the average of the parameters of the l_{th} layer of all previous task networks, i.e., $T(\cdot;\tilde{\theta}_t^l \leftarrow \theta^l)$, $\theta^l = \frac{(\tilde{\theta}_1^l + \cdots \tilde{\theta}_{t-1}^l)}{t-1}$.

The *"add"* action adds a new neural unit (e.g., a linear layer) based on the hidden dimensions of the current layer, with the aim of generating a new layer of learnable parameter space for the new task network. When the l_{th} layer of the new task network $T(\cdot;\tilde{\theta}_t)$ executes the action, a parameter space with the same number of neurons as the l_{th} layer is added after it, i.e., $T(\cdot;\tilde{\theta}_t^l + w^l)$.

The *"remove"* action removes the current layer of the task network, with the aim of removing redundant neurons that are not useful for the old task. When the l_{th} layer of the new task network $T(\cdot;\tilde{\theta}_t)$ executes the action, then this layer will be removed in the task network, i.e., $T(\cdot;\tilde{\theta}_t - \tilde{\theta}_t^l)$.

The *"reset"* action resets the parameters of the current task network, with the aim of introducing a completely new learning space for new tasks. When the l_{th} layer of the new task network $T(\cdot;\tilde{\theta}_t)$ executes the action, the parameters of that layer are randomly initialized and conform to a uniform distribution, i.e., $T(\cdot;uniform(\tilde{\theta}_t^l))$.

The *"protect"* action protects the parameters of the current task network, with the aim of keeping the relational knowledge from being updated and forgotten during training. When the l_{th} layer of a new task network $T(\cdot;\tilde{\theta}_t)$ executes the action, the parameters of that layer are frozen and not modified during training, i.e., $T(\cdot;|\tilde{\theta}_t^l|)$.

4.2.4 Reward

Once the policy network constructs a task network based on the sampled actions, we use accuracy as a measure of the performance of the current task network. In the current task $\mathcal{T}_t$, we stipulate the accuracy improvement of the current training epoch i over the previous training epoch i-1 as the base reward, and the improvement of the current training epoch i over the highest accuracy in all previous training epochs as the advanced reward. The total rewards obtained by the current task network $T(\cdot;\tilde{\theta}_t)$ at training epoch i is shown as follows:

$$R_t^i = \left(\alpha\left(acc_t^i - acc_t^{i-1}\right) + (1-\alpha)\left(acc_t^i - acc_t^{\max}\right)\right)\gamma, \quad (4)$$

where α is a weight parameter to regulate the proportion of basic and advanced rewards , and γ is a scale parameter to normalize the rewards to a reasonable range.

4.3 Task Network

In different task environments, the structure of the task network is dynamically modified based on the policy network's decisions in order to better adapt to new task learning. Once the task network is constructed, it is put into new task learning.

4.3.1 Encoding Framework

Encoding framework consists of four parts: encoder based on pre-trained model, i.e., BERT (Devlin et al., 2019) and the top layer, middle layer and bottom layer that can be dynamically modified. Among them, the number of neurons (i.e., hidden dimensions) in the top, middle, and bottom layers decreases hierarchically, and a nonlinear activation function exists between each layer, forming a special kind of funnel structure. The advantage of this structure is that the rich features extracted by encoder are abstracted and integrated step by step, and the hierarchical feature extraction contributes to the model's ability to capture the key semantics of the sentence. The initial structure of the top, middle, and bottom layers all consist of a linear layer and an activation function, and in subsequent learning of new tasks, the policy network dynamically extends and updates the three layers to adapt to different learning tasks.

4.3.2 Extensible Classification Module

In contrast to traditional classifiers with a fixed parameter space, Extensible Classification Module

can dynamically add new learnable parameters to the existing parameter space, making it more suitable for continual learning that requires constant learning of new tasks. Specifically, in each learning task, there are m learnable relations. When learning a new task $\mathcal{T}_t$, we first specify the relation knowledge of the previous $\mathcal{T}_{t-1}$ old tasks, i.e., the parameter matrix of the old task is defined as $w_{old} \in R^{h\times(t-1)m}$, and then we will generate a parameter matrix $w_{new} \in R^{h\times m}$ for the new task $\mathcal{T}_t$ as the learning space of the new task. Finally, the parameter matrix w_{old} of the old task and the parameter matrix w_{new} of the new task are spliced together as a classifier for task $\mathcal{T}_t$. The formula is represented as follows:

$$w_{cls} = w_{old} \oplus w_{new}, \tag{5}$$

where $w_{cls} \in R^{h\times tm}$ and $\oplus$ represents matrix concatenation.

4.4 Model Training

4.4.1 Training the Policy Network

The goal of the policy network is to make rational decisions based on the similarities between different tasks, sampling a series of actions to construct a task network suitable for the task. We train the policy network using the popular A3C (Mnih et al., 2016) algorithm, which is an Actor Critic method, so that when our intelligent agent outputs a sequence of actions based on the state, it will also output a critical value of its own decision-making ability synchronously. For each iteration of training, it contains $\mathcal{T}_T$ non-intersecting learning tasks, and when each iteration is completed, we then collect $\mathcal{T}_T$ times of learning data to update the parameters of the policy network with the help of critical values. To enable the policy network to more accurately distinguish between actions that should be encouraged and those that should be suppressed, thereby improving policy optimization efficiency, we employ the advantage function to update the policy gradient during training instead of relying solely on rewards. The advantage function is typically defined as the sum of the reward at the current step and the value function estimate at the next step, minus the value function estimate at the current step. The loss of the policy network is calculated as follows:

$$J(\bar{\theta}) = -\sum_{t=0}^{T} log\pi_{\bar{\theta}}(a_t|s_t)A(s_t,a_t)+ \frac{1}{2}\sum_{t=0}^{T}(R_t - V(s_t;\bar{\theta}_v))^2 - H(\pi_{\bar{\theta}}(a_t|s_t)), \tag{6}$$

$$A(s_t,a_t) = R_t + V(s_{t+1};\bar{\theta}_v) - V(s_t;\bar{\theta}_v), \tag{7}$$

where $A(\cdot)$ is the advantage function, R_t is the reward for the current task $\mathcal{T}_t$, $V(\cdot)$ is the critical function, and $H(\cdot)$ is the entropy of the policy network, which is used to adjust the policy and encourage the intelligent agent to explore.

4.4.2 Training the Task Network

The goal of the task network is to learn the current task and achieve excellent performance. After the policy network samples a series of actions to construct the top, middle, and bottom layers of the task network, it is trained using the dataset $\mathcal{D}^t_{train}$ to learn the relation knowledge of the current task $\mathcal{T}_t$. The loss of the task network at the current task $\mathcal{T}_t$ is calculated as follows:

$$L_{crr}(\tilde{\theta}) = -\frac{1}{|\mathcal{D}^t_{train}|}\sum_{(x_i,y_i)\in\mathcal{D}^t_{train}} \delta_{y_i,r_j} logP(r_j|x_i;\tilde{\theta}), \tag{8}$$

where r_j is the prediction and y_i is the true label, when $r_j = y_i$, $\delta_{y_i,r_j} = 1$, otherwise $\delta_{y_i,r_j} = 0$.

In order to awaken the task network's learning experience on previous tasks and to better utilize the relation knowledge transferred from different tasks, we will use a small number of typical samples to guide the task network's memory on the old relations. More specifically, after the task network has been trained on the current training set $\mathcal{D}^t_{train}$, it is then guide using the typical samples stored in the set $\tilde{M}$ to wake up the model's previous learning memory. The loss of the task network performing guided training is calculated as follows:

$$L_{gt}(\tilde{\theta}) = -\frac{1}{|\tilde{M}|}\sum_{(x_i,y_i)\in\tilde{M}} \delta_{y_i,r_j} logP(r_j|x_i;\tilde{\theta}). \tag{9}$$

5 Experiments

5.1 Datasets

We evaluated our method DCREN on two popular relation extraction benchmarks — FewRel and TACRED.

FewRel	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10
EA-EMR	89.0	69.0	59.1	54.2	47.8	46.1	43.1	40.7	38.6	35.2
EMAR	**98.8**	89.1	89.5	85.7	83.6	84.8	79.3	80.0	77.1	73.8
CML	91.2	74.8	68.2	58.2	53.7	50.4	47.8	44.4	43.1	39.7
RP-CRE	97.9	92.7	91.6	89.2	88.4	86.8	85.1	84.1	82.2	81.5
CRL	98.2	94.6	92.5	90.5	89.4	87.9	86.9	85.6	84.5	83.1
CEAR	98.5	95.4	93.0	91.6	90.5	89.0	87.7	86.6	85.2	83.6
CREST	98.7	93.6	**93.8**	**92.3**	**91.0**	**89.9**	87.6	86.7	86.0	**84.8**
Ours	98.4	**95.6**	93.5	92.0	**91.0**	89.4	**88.5**	**87.4**	**86.2**	84.5
TACRED	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10
EA-EMR	47.5	40.1	38.3	29.9	28.4	27.3	26.9	25.8	22.9	19.8
EMAR	96.6	85.7	81.0	78.6	73.9	72.3	71.7	72.2	72.6	71.0
CML	57.2	51.4	41.3	39.3	35.9	28.9	27.3	26.9	24.8	23.4
RP-CRE	97.6	90.6	86.1	82.4	79.8	77.2	75.1	73.7	72.4	72.4
CRL	97.7	93.2	89.8	84.7	84.1	81.3	80.2	79.1	79.0	78.0
CEAR	97.9	93.7	91.4	87.5	85.1	83.2	81.1	79.5	79.3	79.0
CREST	97.3	91.4	82.3	82.5	79.2	75.8	78.8	77.4	78.6	**79.4**
Ours	**98.1**	**95.0**	**92.5**	**88.0**	**85.5**	**83.7**	**81.3**	**80.1**	**79.4**	**79.4**

Table 1: Experimental results after learning each task. The best results are marked in bold and the second best results are underlined.

FewRel (Han et al., 2018) contains 100 relation categories with 700 instances per category, containing 70,000 instances in total. The instances are extracted from Wikipedia and Wikidata and are intended for use in relation extraction tasks in natural language processing. We follow the previous settings (Zhao et al., 2022) and use 80 relations with 700 samples each and divide them into 10 subsets to simulate 10 disjoint tasks.

TACRED (Zhang et al., 2017) is a large-scale relation extraction dataset. The dataset contains 106,264 instances covering 42 relation categories. The instances are constructed from English newswire and web texts. Following the settings of previous work (Zhao et al., 2022), the number of relations for continual learning is limited to 40, and the maximum number of training samples per relation is 320.

5.2 Implementation Details and Baseline Models

For the evaluation metrics, we follow the existing evaluation scheme using accuracy (Hu et al., 2022), for the currently learned K tasks, the average accuracy over 5 experiments is reported in the paper. For the selection of typical samples, we follow previous work (Zhang et al., 2022) using the K-means algorithm to cluster the samples for each relation r_i. The number of clusters is defined as the memory size $M^{r_i} = 10$. For hyperparameter settings, α = 0.5, γ = 10, train batch size = 16, test batch size = 64, task network learning rate = 1e-5, policy network learning rate = 1e-3, and extensible classification module learning rate = 1e-3. For the experimental environment, the IDE used for the experiments in this paper is Pycharm2021 Professional, PyTorch version 1.9.1, and CUDA version 12.1. The model training and inference are performed on an NVIDIA GeForce RTX4090 with 24GB GPU memory.

For the baseline model, we selected representative models in recent years, including EA-EMR (Wang et al., 2019), EMAR (Han et al., 2020), CML (Wu et al., 2021), RP-CRE (Cui et al., 2021), CRL (Zhao et al., 2022), CEAR (Zhao et al., 2023) and CREST (Le et al., 2024). see Section 2 for a detailed description.

5.3 Results and Analyses

5.3.1 Main Results

Table 1 shows the results of the main experiment of our DCREN with all baseline models. From these results, we can draw the following conclusions:

(1) Our proposed method achieves overall state-of-the-art performance on two different datasets. This is due to the fact that our method dynamically changes the network structure for each task, enabling the transfer of relation knowledge across tasks, a capability not found in other baseline models.

(2) Although the performance of all comparative models decreases as the continual learning task increases, our model is still able to maintain a high performance, which suggests that our proposed extensible classification module plays an indispensable role in combating catastrophic forgetting.

(3) On the TACRED benchmark, the performance of all models is degraded because the dis-

tribution of the number of relation categories is extremely unbalanced, but our model still shows a huge advantage with up to 10.2% performance improvement compared to the CREST baseline.

(4) On the FewRel benchmark, as the number of task increases, it can be noticed that the gap between the performance of CEAR model and ours is increasingly large, and from the first incremental learning T2 to T10, the gap between the CEAR and our DCREN improves from +0.2 to +1.0. The reason for this situation is that CEAR consider only the catastrophic forgetting problem and rely excessively on stored typical samples, and overfitting occurs during the frequent replay of the samples. On the contrary, DCREN, our model dynamically updates the structure in each task, so that the network will only experience typical samples replay once in each task, which minimizes the problem of model overfitting to typical samples. Most importantly, our model transferred the knowledge learned in previous tasks, avoiding catastrophic forgetting.

5.3.2 Ablation Study

We performed ablation studies to verify the effectiveness of individual modules in our model. Specifically, for "w/o Policy", we remove the action decision of the policy network, i.e., DCREN cannot dynamically change its encoding framework part in each task. For "w/o Ecm", we remove the extensible classification module of the task network, i.e., DCREN cannot extend the new learning space for the new task on top of the old one, but instead use the stored typical samples to retrain a classifier that contains all the visible relations. For "w/o Both", we perform the above two ablation operations simultaneously.

The experimental results are shown in Table 2, where it can be observed that the performance of the model is impaired when any of the modules are removed individually, indicating that all modules are beneficial and are an essential part of DCREN. In particular, the "Policy" dominates the knowledge transfer capability of DCREN, which helps DCREN to utilize the relation knowledge of previous tasks to guide the model to learn new tasks quickly, the "Ecm" dominates knowledge memorization ability of DCREN and it helps DCREN to fight against catastrophic forgetting. Further, when both were removed, the model shows a significant decrease in performance, indicating that the two are complementary and both have a huge positive effect on the model.

		T2	T4	T6	T8	T10
FewRel	w/o Policy	95.2	91.5	88.6	86.2	83.7
	w/o Ecm	95.3	91.9	89.1	86.8	83.9
	w/o Both	94.7	89.0	86.2	84.2	81.1
	Ours	**95.6**	**92.0**	**89.4**	**87.4**	**84.5**
TACRED	w/o Policy	93.9	86.4	82.8	78.9	77.7
	w/o Ecm	93.6	87.8	83.7	79.9	77.6
	w/o Both	91.6	84.2	77.9	74.7	72.5
	Ours	**95.0**	**88.0**	**83.7**	**80.1**	**77.9**

Table 2: Experimental results of ablation studies.

6 Conclusion

In this work, we propose DCREN, a continual relation extraction model that can dynamically change its model structure to address two important challenges in continual learning: catastrophic forgetting and knowledge transfer. DCREN can dynamically transfer knowledge of similar relations or add new learning spaces for new relations to the task network based on the similarities and differences of relations in different tasks. Specifically, DCREN will take more Class I (*reload*, *fuse*, *protect*) actions to transfer beneficial knowledge for the task network when the relations are similar, whereas when the relations are dissimilar, DCREN will take Class II actions (*add*, *reset*, *remove*) to expand new learning spaces for the task network. Experimental results show that DCREN achieves state-of-the-art performance and outperforms current mainstream memory-based models. Further, we conducted additional empirical research to demonstrate the important role of knowledge transfer in continual learning as a capability that a continual learning model should have. In future work, we will explore more possibilities of dynamic models in continual learning.

Limitations

Although DCREN effectively mitigated catastrophic forgetting in continual relation extraction through knowledge transfer capabilities, there are still some limitations: (1) Our model still needs to consume additional space to store a small number of typical samples to teach the task network how to use previously learned relational knowledge. (2) Our approach used reinforcement learning to train the model, which resulted in slower convergence of our model compared to mainstream memory-based continual relation extraction models. (3) Large language models may improve the performance of continual relation extraction due to their powerful parameters, which has not been explored in this

paper.

References

Tameem Adel, Han Zhao, and Richard E. Turner. 2019. Continual learning with adaptive weights (claw). *ArXiv*, abs/1911.09514.

Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. 2018. Efficient lifelong learning with a-gem. *ArXiv*, abs/1812.00420.

Li Cui, Deqing Yang, Jiaxin Yu, Chengwei Hu, Jiayang Cheng, Jingjie Yi, and Yanghua Xiao. 2021. Refining sample embeddings with relation prototypes to enhance continual relation extraction. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 232–243, Online. Association for Computational Linguistics.

Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Aleš Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. 2022. A continual learning survey: Defying forgetting in classification tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(7):3366–3385.

Cyprien de Masson d'Autume, Sebastian Ruder, Lingpeng Kong, and Dani Yogatama. 2019. Episodic memory in lifelong language learning. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Yixing Fan, Xiaohui Xie, Yinqiong Cai, Jia Chen, Xinyu Ma, Xiangsheng Li, Ruqing Zhang, and Jiafeng Guo. 2022.

Xu Han, Yi Dai, Tianyu Gao, Yankai Lin, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. 2020. Continual relation learning via episodic memory activation and reconsolidation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6429–6440, Online. Association for Computational Linguistics.

Xu Han, Hao Zhu, Pengfei Yu, Ziyun Wang, Yuan Yao, Zhiyuan Liu, and Maosong Sun. 2018. FewRel: A large-scale supervised few-shot relation classification dataset with state-of-the-art evaluation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4803–4809, Brussels, Belgium. Association for Computational Linguistics.

Chengwei Hu, Deqing Yang, Haoliang Jin, Zhen Chen, and Yanghua Xiao. 2022. Improving continual relation extraction through prototypical contrastive learning. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 1885–1895, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and Philip S. Yu. 2022. A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE Transactions on Neural Networks and Learning Systems*, 33(2):494–514.

Ronald Kemker and Christopher Kanan. 2017. Fearnet: Brain-inspired model for incremental learning. *ArXiv*, abs/1711.10563.

James Kirkpatrick, Razvan Pascanu, Neil C. Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. 2016. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114:3521 – 3526.

Thanh-Thien Le, Manh Nguyen, Tung Thanh Nguyen, Ngo Van Linh, and Thien Huu Nguyen. 2024. Continual relation extraction via sequential multi-task learning. In *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*, pages 18444–18452. AAAI Press.

Zhizhong Li and Derek Hoiem. 2018. Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(12):2935–2947.

David Lopez-Paz and Marc'Aurelio Ranzato. 2017. Gradient episodic memory for continual learning. In *Neural Information Processing Systems*.

Arun Mallya and Svetlana Lazebnik. 2018. Packnet: Adding multiple tasks to a single network by iterative pruning. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7765–7773.

Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Tim Harley, Timothy P. Lillicrap, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous methods for deep reinforcement learning. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, page 1928–1937. JMLR.org.

Qi Qin, Wenpeng Hu, Han Peng, Dongyan Zhao, and Bing Liu. 2021. Bns: Building network structures dynamically for continual learning. In *Advances in*

Neural Information Processing Systems, volume 34, pages 20608–20620. Curran Associates, Inc.

David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy P. Lillicrap, and Greg Wayne. 2018. Experience replay for continual learning. In *Neural Information Processing Systems*.

Sagnik Sarkar, Pardeep Singh, Namrata Kumari, and Poonam Kashtriya. 2023. The task of question answering in nlp: A comprehensive review. In *Proceedings of International Conference on Recent Innovations in Computing*, pages 603–611, Singapore. Springer Nature Singapore.

Yuanhe Tian, Yan Song, and Fei Xia. 2022. Improving relation extraction through syntax-induced pretraining with dependency masking. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1875–1886, Dublin, Ireland. Association for Computational Linguistics.

Tom Véniat, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2020. Efficient continual learning with modular networks and task-driven priors. *ArXiv*, abs/2012.12631.

Hong Wang, Wenhan Xiong, Mo Yu, Xiaoxiao Guo, Shiyu Chang, and William Yang Wang. 2019. Sentence embedding alignment for lifelong relation extraction. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 796–806, Minneapolis, Minnesota. Association for Computational Linguistics.

Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. 2023. A comprehensive survey of continual learning: Theory, method and application. *IEEE transactions on pattern analysis and machine intelligence*, PP.

Peiyi Wang, Yifan Song, Tianyu Liu, Binghuai Lin, Yunbo Cao, Sujian Li, and Zhifang Sui. 2022. Learning robust representations for continual relation extraction via adversarial class augmentation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 6264–6278, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Zhepei Wei, Jianlin Su, Yue Wang, Yuan Tian, and Yi Chang. 2020. A novel cascade binary tagging framework for relational triple extraction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1476–1488, Online. Association for Computational Linguistics.

Tongtong Wu, Xuekai Li, Yuan-Fang Li, Reza Haffari, Guilin Qi, Yujin Zhu, and Guoqiang Xu. 2021. Curriculum-meta learning for order-robust continual relation extraction. *ArXiv*, abs/2101.01926.

Heming Xia, Peiyi Wang, Tianyu Liu, Binghuai Lin, Yunbo Cao, and Zhifang Sui. 2023. Enhancing continual relation extraction via classifier decomposition. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 10053–10062, Toronto, Canada. Association for Computational Linguistics.

Deming Ye, Yankai Lin, Peng Li, and Maosong Sun. 2022. Packed levitated marker for entity and relation extraction. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4904–4917, Dublin, Ireland. Association for Computational Linguistics.

Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. 2017. Lifelong learning with dynamically expandable networks. *ArXiv*, abs/1708.01547.

Han Zhang, Bin Liang, Min Yang, Hui Wang, and Ruifeng Xu. 2022. Prompt-based prototypical framework for continual relation extraction. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 30:2801–2813.

Yuhao Zhang, Victor Zhong, Danqi Chen, Gabor Angeli, and Christopher D. Manning. 2017. Position-aware attention and supervised data improve slot filling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 35–45, Copenhagen, Denmark. Association for Computational Linguistics.

Kang Zhao, Hua Xu, Jiangong Yang, and Kai Gao. 2022. Consistent representation learning for continual relation extraction. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 3402–3411, Dublin, Ireland. Association for Computational Linguistics.

Wenzheng Zhao, Yuanning Cui, and Wei Hu. 2023. Improving continual relation extraction by distinguishing analogous semantics. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1162–1175, Toronto, Canada. Association for Computational Linguistics.

Zexuan Zhong and Danqi Chen. 2021. A frustratingly easy approach for entity and relation extraction. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 50–61, Online. Association for Computational Linguistics.

A Empirical Study of knowledge transfer

Knowledge transfer is a desirable and extremely important capability for models that require continual learning. Extracting knowledge from previously learned tasks not only facilitates the learning of new tasks, but also effectively mitigates catastrophic forgetting. In order to verify the essential role of knowledge transfer and how our DCREN specifically transfers knowledge about similar relations, we designed the following experiments: The average of the embeddings of all instances of a relation as a prototype of itself and represents the overall representation of the relation. The relations in the dataset are equally divided into different sets based on the similarity of each relation prototype to the other relation prototypes, and then we observe the performance of the strongest baseline CEAR and our DCREN in different sets of relations.

Figure 2 illustrates the experimental results on both FewRel and TACRED datasets. It can be observed that the performance of the models all decrease as the similarity of the relations increases, due to the fact that the severe decay of knowledge in similar relations is a key factor in catastrophic forgetting. However, our model is still able to maintain a relatively excellent performance in a highly similar set of relations and is ahead of CEAR by a large gap, which is due to the fact that our model is able to transfer the knowledge of similar relations and effectively alleviate catastrophic forgetting.

Figure 3 illustrates the distribution of actions performed by our DCREN in different sets of similarity relations. It can be observed that (1) the probability that DCREN performs class I actions (*reload*, *fuse*, *protect*) grows progressively larger as the similarity increases, suggesting that in a collection of relations with higher similarity, the more relation knowledge needs to be transferred to maintain the model's performance against catastrophic forgetting. (2) As can be seen from Figure 3(a), when the set of relations has less similarity and less transferable beneficial knowledge, our DCREN will more often choose class II actions (*add*, *reset*, *remove*) to create a new learning space for new relations, so as to cope with the situation where the distribution of relations is more varied.

In summary, knowledge transfer is effective in mitigating catastrophic forgetting and is a capability that a continual learning model deserves and must have. Our DCREN is able to selectively transfer relation knowledge according to the similarity of the relations. With relations that are highly similar, DCREN transfers more knowledge to maintain model performance. With relations that are not similar, DCREN adds new learning spaces to accommodate different relation knowledge.

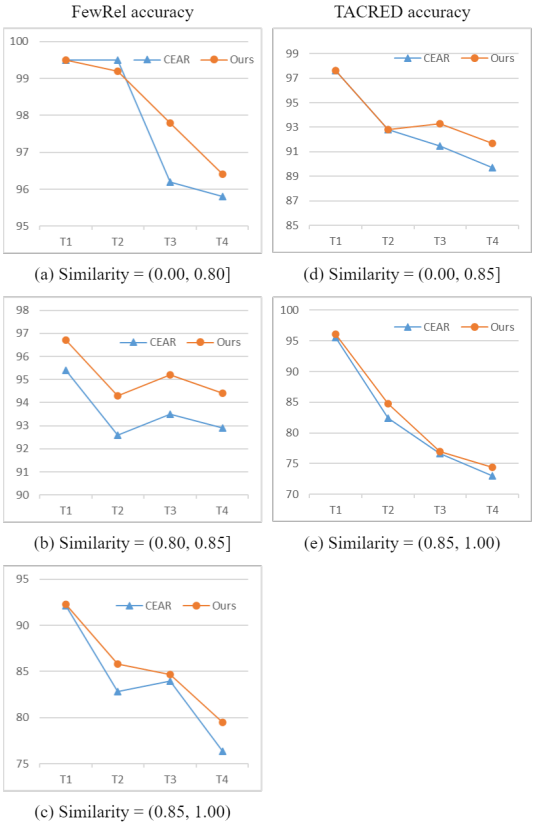

Figure 2: Experimental results of CEAR and our method in relation sets with different similarities. Due to the smaller number of TACRED relations, it is divided into only two sets of relations (0.00, 0.85] and (0.85,1.00).

B Influence of Memory Size

In the former section, we have demonstrated the important role of knowledge transfer. Further, in order to verify the effect of memory size on DCREN, we conducted experiments with the strongest baseline model of memory-based models, CEAR, as a reference.

Figure 4 shows the experimental results of our DCREN and CEAR with different memory size settings. It can be noticed that (1) our model achieves the best performance under different memory settings in both datasets, which indicates that our model has strong generalization ability and is not as sensitive to memory size compared to the memory-based model. (2) As can be seen from Figure 4(b) and (c), with a balanced distribution of relations, our model leads the performance of

(a) FewRel

(b) TACRED

Figure 3: Distribution of actions performed by our method in different similarity sets.

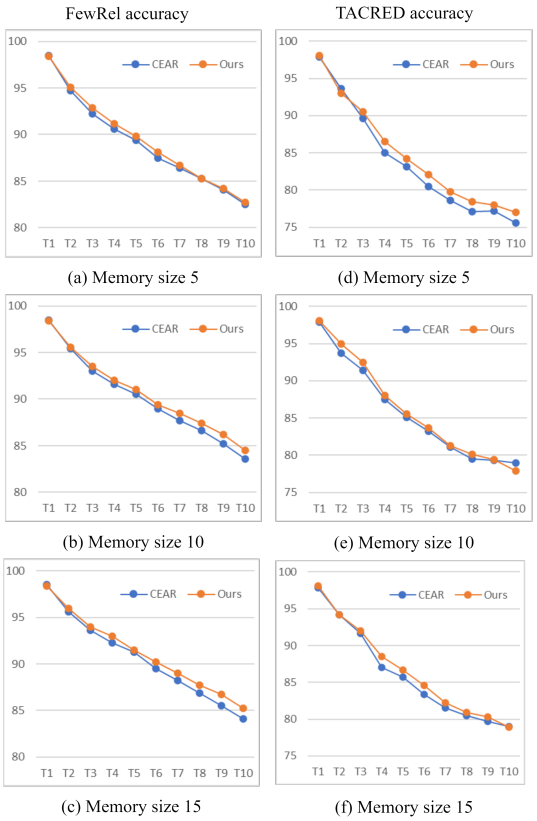

Figure 4: Experimental results of CEAR and our method with different memory sizes.

CEAR increasingly as the tasks increase, which reveals that the utilization of typical samples in memory for DCREN is also completely ahead of the memory-based model.

C Comparison with BNS

Similar to our work is the BNS proposed by Qin et al. (2021) et al. This approach also dynamically builds network architectures in continual learning. However, our approach is significantly different from BNS in the following ways: **(1) Differences in Continual Learning Types.** The BNS focuses on task-incremental learning (Task-CL) within continual learning, while our DCREN focuses on class-incremental learning (Class-CL), which is more challenging. **(2) Field Differences.** The BNS is used to solve image recognition problems in computer vision, whereas our DCREN is designed to address relation extraction problems in natural language processing. **(3) Methodological Differences.** The BNS stipulates a fixed number of hidden layers for the dynamic model. When facing a new task, its dynamism is reflected in the different actions performed for each hidden layer to rebuild the network, i.e., the process of destruction and reconstruction. On the other hand, our DCREN does not specify a fixed number of hidden layers but instead features three different dimensional encoding layers (top, middle, and bottom). Within these layers, the hidden layers can be optimized (Class I actions) or expanded/removed (Class II actions) arbitrarily, i.e., the process of optimization and expansion (or removal).