

# Parameter Efficient Continual Automated Knowledge Graph Completion

Janna Omelivanenko<sup>1,2</sup>, Andreas Hotho<sup>1,2</sup>, and Daniel Schlör<sup>1,2</sup>

<sup>1</sup> CAIDAS Center for Artificial Intelligence and Data Science, Würzburg, Germany

<sup>2</sup> Julius-Maximilians-University of Würzburg, Würzburg, Germany  
{omelivanenko, hotho, schloer}@informatik.uni-wuerzburg.de

**Abstract.** Automated Knowledge Graph Completion (KGC) is a crucial task in the semantic web community, focused on discovering missing structured information within Knowledge Graphs (KGs), such as identifying links between entities or classifying relation types. The continually improving performance of pre-trained Large Language Models (LLMs), with their inherent ability to learn world knowledge, has shown significant promise for KGC. However, as KGs evolve with new factual knowledge, continual fine-tuning of such models becomes necessary, making them vulnerable to catastrophic forgetting and incurring significant computational and storage overhead. In this paper, we propose a dynamic Parameter-Efficient Fine-Tuning method that introduces a novel and effective combination of masking and growing strategies for continual KGC, addressing a core limitation of existing methods: static architectures that suffer from parameter saturation over time. Our method enables LLMs to continually adapt to evolving KGs while preserving previously acquired knowledge. It supports knowledge transfer, mitigates catastrophic forgetting, and incrementally expands model capacity as needed. Evaluation is conducted in two continual learning settings, task-incremental learning for link prediction and class-incremental learning for relation extraction. Experimental results show that the proposed method outperforms both rehearsal-based and rehearsal-free baselines, offering an effective and scalable solution for continual KG modeling.

**Keywords:** Knowledge Graphs · Link Prediction · Relation Extraction · Large Language Models · Continual Learning.

## 1 Introduction

Knowledge Graphs (KGs) provide a structured knowledge representation which is crucial for various applications such as question answering or semantic analysis [46,55]. However, their construction and maintenance are often time-consuming and complex, making automated Knowledge Graph Completion (KGC) a central focus of research. Here, the continually improving performance of pre-trained Large Language Models (LLMs), with their inherent ability to learn world knowledge, has shown significant potential for KGC [29]. LLM-based solutions for KGC include prompting approaches, in which LLMs receive an instruction (prompt)

as input to derive new knowledge [29]. However, their performance can be heavily dependent on the design and quality of the prompt. As an alternative, fine-tuning LLMs to specific KG domains has shown promising results, adapting the model’s knowledge to the nuances of a particular graph [8]. Conventional methods for KGC typically provide a static solution, where a model is trained on a dataset that represents only a snapshot of fixed knowledge [8,27,35,41,42]. Continuous training on new data leads to catastrophic forgetting [18], where the learning of new knowledge overwrites previously learned information. To address this limitation, recent research has increasingly turned to the Continual Learning (CL) paradigm. Within this context, several key requirements have been identified: preventing catastrophic forgetting [18], ensuring parameter efficiency for LLM-based approaches [28,31,47], and supporting effective knowledge transfer between previously learned and new tasks [31].

These principles of CL have been successfully applied to KGC, establishing continual KGC as an emerging research direction. Existing approaches for continual KGC can be broadly categorized into regularization-based, rehearsal-based, and parameter isolation-based methods [44]. Although regularization- and rehearsal-based approaches [5,6,9,19,20,26,50,56] show promise in mitigating catastrophic forgetting, they cannot fully prevent it due to their inherent design of updating previously learned parameters. Pure parameter isolation approaches, such as training separate Parameter-Efficient Fine-Tuning (PEFT) modules per task [48], fully prevent catastrophic forgetting, but sacrifice knowledge sharing between tasks. Recent works like MoCL [45] attempt to bridge this gap through similarity-based weight initialization for new tasks, but this sharing is limited to initialization and gets overwritten during training. In the context of continual link prediction, CapsKG [28] introduces a hybrid approach combining relation-specific parameters that are updated through a complex routing mechanism and shared parameters that are separated through learned binary masks. This approach is limited to Task-Incremental Learning (TIL), where task identifiers are known during inference, and faces efficiency challenges due to its routing mechanism, which is known to be over 50 times slower than pure masked PEFT [16].

To address these limitations, we propose Continual Growing Knowledge Graph Completion (CGKGC), illustrated in Figure 1. CGKGC combines masked PEFT with selective backward pass updates to achieve both catastrophic forgetting prevention and continuous knowledge sharing. The core architecture employs a single PEFT module where neurons are dynamically selected through learning task-specific masks. While all active neurons participate in the forward pass, enabling knowledge transfer, gradient updates are selectively blocked for previously reserved neurons, ensuring parameter isolation. To prevent saturation as tasks accumulate, we introduce a growing mechanism that dynamically extends intermediate representations. For Class-Incremental Learning (CIL) scenarios where task identifiers are unknown during inference, we integrate a similarity-based task identification system that leverages maximum likelihood estimation over training embedding distributions. This enables application to both

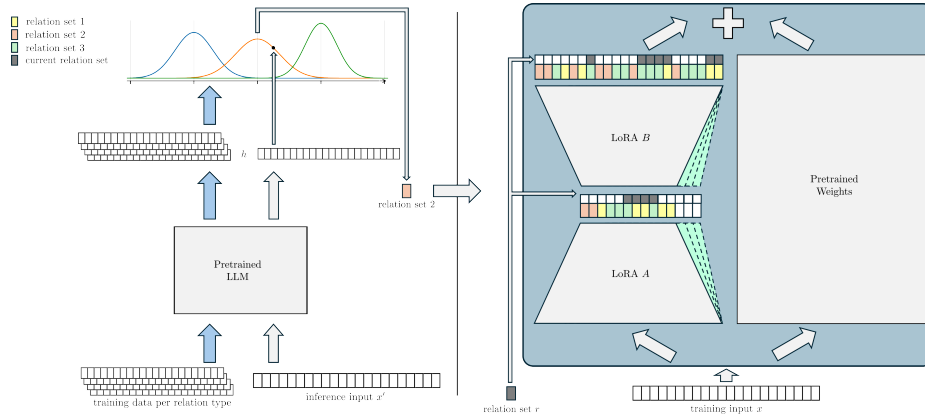


Fig. 1: Architecture overview of our proposed CGKGC approach for continual KGC using a LoRA PEFT module. During inference in the CIL setting (left), relation set identification is performed via embedding-based maximum likelihood estimation to select the appropriate set of relation types. The training process (right) combines relation-set-dependent masked neuron selection with selective gradient updates, while the growing mechanism prevents saturation by extending the intermediate representation space.

continual link prediction and relation extraction while maintaining efficient training characteristics.

In summary, our core contributions are: 1) A novel continual learning approach for KGC that completely prevents catastrophic forgetting while enabling continuous knowledge transfer. 2) A growing mechanism that dynamically expands the model’s capacity while maintaining parameter efficiency and mitigating parameter saturation. 3) Extension of the approach to both TIL and CIL settings, with a comprehensive evaluation of continual link prediction and relation extraction, showing superior performance over state-of-the-art methods. 4) An ablation study which shows that our novel integration of masking and growth strategies makes more efficient use of available parameters, leading to improved performance with a reduced model size.<sup>3</sup>

The remainder of this paper is structured as follows: Section 2 includes related work on CL, PEFT, and growing networks. Section 3 introduces CGKGC in detail. Section 4 describes our experimental setup, while Section 5 presents our results. Finally, Section 6 concludes the paper.

## 2 Related Work

**PEFT** PEFT approaches introduce distinct trainable parameters at various points in the LM while keeping base parameters frozen. Housby et al. [12] de-

<sup>3</sup> Our source code is available here: <https://professor-x.de/code-cgkgc>.

velop Adapters that insert trainable layers between the model’s existing layers. Hu et al. [13] introduce Low Rank Adaptation (LoRA) which implements parameter updates through additive transformations. These methods all enable efficient model adaptation to new knowledge and tasks while preventing catastrophic forgetting of the initial weights. While PEFT approaches can maintain performance on previous tasks by using dedicated parameter sets for each task, this prevents knowledge sharing between tasks.

**Growing networks** Neural growth strategies have been explored in various works. Rusu et al. [33] develop a lifelong learning approach that expands the full network while copying previous parameters, leading to high parameter counts. Additional research optimizes PEFT module dimensions [40,53] but does not jointly address CL and knowledge transfer.

**Continual Learning (CL)** CL is commonly divided into regularization-based, rehearsal-based, and parameter isolation-based methods [45].

*Regularization-based* CL approaches augment the loss function with regularization to reduce changes to parameters for previous tasks [1,14,18,24,36]. Within CRE, Nguyen et al. [26] reduce catastrophic forgetting through a specific regularization optimizing class-wise eigenvalues. In CLP, Cui et al. [6] integrate regularization into their graph-based link prediction approach. While these regularizations mitigate the effect of catastrophic forgetting, they do not fully prevent it due to updates to the full network.

*Rehearsal-based* CL methods use a memory buffer to repeatedly train on old and new tasks simultaneously [3,4,25,30,32]. Implementations in CRE and CLP include the direct use of training samples for repeated training [50]. Since retaining training data can cause privacy and security issues and run the risk of overfitting to the kept data, many pseudo-rehearsal methods have been created that replace the direct training data with pseudo-data through using relation prototypes or a generator network [5,9,19,20,43,52,56]. These methods require repeated updates to already fully trained past relations. In contrast, our approach does not require additional training effort after the relation-specific training phase and fully prevents catastrophic forgetting by design.

*Parameter isolation-based* CL aims to combat catastrophic forgetting by isolating the parameters of individual tasks, e.g., by freezing previous parameters and adding new parameters for each task [33] or reserving task-specific neurons through masking [34]. Further, there exist approaches that purely learn a binary mask for each task without training the base model [49]. Other works use a PEFT module for each task to remain parameter efficient [48]. To enable knowledge transfer between tasks, Razdaibiedina et al. [31] introduce a prepending strategy for showing previous frozen PEFT modules to the current task. Wang et al. [45] use similarity-based weighting instead to identify relevant previous tasks and initialize the new task module with the weights from relevant tasks. Within CRE, Zhou et al. [57] decompose the goal into task identification and

within-task prediction. For within-task prediction, they leverage LoRA with additional data augmentations. For task identification, they propose an ensemble of the within-task prediction models and aggregate them through a temporal voting strategy. As this voting strategy fully separates the individual models, this approach does not allow knowledge transfer. Within CLP, Omeliyanenko et al. [28] combine relation-specific and shared parameters within a PEFT module. While relation-specific parameters are reused via a routing mechanism, the shared parameters are masked to prevent catastrophic forgetting. Liu et al. [21] introduces FastKGE, a continual knowledge graph embedding framework that utilizes an Incremental Low-Rank Adapter (IncLoRA) mechanism to efficiently manage evolving knowledge, while preserving existing information by freezing previous parameters. New information is captured by training isolated LoRA modules with adaptive rank allocation. While parameter isolation-based CL can prevent catastrophic forgetting by design, full isolation impedes knowledge transfer between previous and current relations.

Although various Continual Knowledge Graph Embedding (CKGE) methods [6,21,50] effectively update structural vector representations, they are limited to graph topology and lack the semantic reasoning and world knowledge of pre-trained LLMs.

### 3 Methodology

In this section, we first introduce the architectures our solution builds upon. We then present CGKGC, which adapts pretrained LLMs for continual KGC. While our approach can be quickly adapted to different PEFT methods, we showcase CGKGC with the popular low rank adapters [13] in this work. We first introduce CGKGC on an individual task and then generalize to knowledge sharing between tasks.

#### 3.1 Adapter Module

The concept of adapter modules introduces additional parameters into a pre-trained LLM, to adapt it to new knowledge or tasks, such as link prediction, without modifying the original model weights. Besides design goals such as parameter efficiency and minimizing storage overhead, a major advantage of adapter modules is their ability to avoid or reduce the risk of catastrophic forgetting in continuous knowledge integration scenarios, an aspect highly relevant for KGC. For this, each knowledge source or task is associated with a distinct set of trainable adapter parameters, enabling modular and continual updates.

Several adapter variants have been proposed in recent work, including Bottleneck Adapters [12] and Low-Rank Adaptation (LoRA) [13]. These approaches differ in structure and in how they are integrated into the underlying LM.

### 3.2 Bottleneck Adapter

An early adapter variant is the Hously or bottleneck adapter [12]. As the name suggests, it builds on the bottleneck principle commonly used in autoencoders, comprising a down-projection feed-forward layer  $f_d$ , a non-linear activation  $\sigma$ , and an up-projection feed-forward layer  $f_u$  that restores the original dimensionality. These modules are integrated twice within each Transformer layer  $l$ : once after the multi-head self-attention sub-layer and once after the feed-forward sub-layer. For a given relation type (link prediction) or set of relation types (relation extraction) noted as  $(r)$ , a bottleneck adapter can be defined as follows:

$$h^{(r)} = f_u^{(r)}(\sigma(f_d^{(r)}(h^{(r)}))) + h^{(r)}. \quad (1)$$

where  $h \in \mathbb{R}^{z \times e}$  is the input representation from a preceding sub-layer, with  $z$  denoting the sequence length and  $e$  the hidden dimensionality of the model. The residual connection ensures that the adapter output  $h^{(r)}$  preserves the original information while integrating relation-specific transformations.

### 3.3 Low-Rank Adaptation (LoRA)

LoRA [13] introduces a parameter-efficient strategy for task-specific fine-tuning by adapting a frozen pre-trained weight matrix of the LLM with low-rank trainable components. For a given Transformer layer  $l$  with pre-trained weight matrix  $W_l \in \mathbb{R}^{d \times k}$ , LoRA adds two task-specific trainable matrices,  $A_l^{(r)} \in \mathbb{R}^{d \times a}$  and  $B_l^{(r)} \in \mathbb{R}^{a \times k}$ , with rank  $a \ll \min(d, k)$ . In the context of task-incremental learning for knowledge graph construction, each *task*  $(r)$  refers to a relation type in link prediction or a set of relation types in relation extraction.

$$h_l^{(r)} = W_l x_l^{(r)} + B_l^{(r)} A_l^{(r)} x_l^{(r)} \quad (2)$$

where  $x_l^{(r)} \in \mathbb{R}^k$  is the input vector at layer  $l$  for relation context  $(r)$ . The low-rank component  $B_l^{(r)} A_l^{(r)} x_l^{(r)}$  introduces relation-specific adaptation while leaving the core LLM parameters untouched. This structure enables continual KGC, where isolating updates to task-relevant parameters helps to preserve prior knowledge. While this approach can be adopted for all pre-trained layers in the LM, it is typically only applied to attention layers for parameter-efficiency [13].

### 3.4 Masked PEFT

In standard PEFT-based approaches, a new PEFT module is added to the backbone model for each new task or dataset. These modules are trained independently and do not share parameters, which prevents any form of knowledge transfer across tasks. To address this limitation, we draw inspiration from [16,34] and propose a *single* PEFT module that is shared across all tasks (relations, relation sets) encountered during CL. By maintaining a single PEFT module, we

enable cross-task knowledge transfer while significantly reducing the parameter footprint. However, continual updates to a shared module risk catastrophic forgetting. To mitigate this, we introduce a gating mechanism with masking that selectively controls the activation of parameters and gradient flow during training, preserving previously learned knowledge while adapting to new data.

For each new task ( $r$ ), we compute a trainable binary mask  $m^{(r)}$  with the same dimensionality as the activated neurons within the PEFT module. This binary mask is derived from task-specific embeddings  $e^{(r)}$ , representing a numerical identifier for each task in latent space. The embeddings are passed through a sigmoid activation function and scaled by a hyperparameter  $s$  to produce a pseudo-gating function

$$m^{(r)} = \sigma(se^{(r)}). \quad (3)$$

The hyperparameter  $s$  is a positive scalar that is gradually increased during training, typically to values  $\gg 1$ . This annealing process drives the learned mask  $m^{(r)}$  to converge toward binary values close to 0 or 1. These pseudo-binary masks  $m^{(r)}$  are applied to the corresponding activated neurons in the PEFT module through element-wise multiplication, gating their contribution during training.

*Gated LoRA* In LoRA, we apply task-specific gating to the shared low-rank adaptation weights as

$$h_l^{(r)} = W_l x_l^{(r)} + \left( B_l (A_l x_l^{(r)} \otimes m_{A_l}^{(r)}) \otimes m_{B_l}^{(r)} \right) \quad (4)$$

Here,  $m_{A_l}^{(r)} \in \mathbb{R}^r$  and  $m_{B_l}^{(r)} \in \mathbb{R}^k$  are the learned gating masks for relation context ( $r$ ), applied to the intermediate outputs of the LoRA weight matrices  $A_l$  and  $B_l$ , respectively. Note that  $A_l$  and  $B_l$  themselves are shared across all tasks and are not conditioned on ( $r$ ). Only the gating masks are task-specific.

For integration into LoRA, the following steps are applied to both gating masks  $m_{A_l}^{(r)}$  and  $m_{B_l}^{(r)}$  for each layer  $l$ . For brevity, we omit this distinction in the notation. Since the pseudo-gating function typically produces continuous values, the resulting masks may remain non-binary and lead to catastrophic forgetting. To address this, we adopt the approach proposed by [15] and binarize the task-specific masks after training on each task ( $r$ ) is complete:

$$m_{\text{eval}}^{(r)} = \begin{cases} 1 & \text{if } \sigma(se^{(r)}) > 0.5 \\ 0 & \text{otherwise.} \end{cases} \quad (5)$$

Through this post-training binarization, the explicit activation and deactivation of neurons is achieved, allowing tasks to directly draw upon or isolate knowledge from previous tasks. However, weight updates to neurons already used before could lead to catastrophic forgetting of previous tasks  $r^{(prev)}$  that have been learned prior to task ( $r$ ). To prevent this, a mask  $m^{(prev)}$  that aggregates all neurons used for all previous tasks, i.e., relations or relation-sets, is calculated through

$$m^{(prev)} = \text{MaxPool}\left(\{m_{eval}^{(r')}, r' \in \{r^{(prev)}\}\}\right). \quad (6)$$

The resulting mask  $m^{(prev)}$  is then used to restrict gradient updates during backpropagation for the current task ( $r$ ).

$$g^{(r)} = g^{(r)} \otimes (1 - m^{(prev)}). \quad (7)$$

Therefore, only neurons that were not previously activated are updated, thereby preserving the knowledge learned from earlier tasks.

To prevent early exhaustion of the shared PEFT module’s capacity, [34] emphasize the importance of maintaining sparsity in the task-specific masks  $m^{(r)}$ . To encourage such sparsity, we incorporate a regularization term into the loss function  $\mathcal{L}$  during training for the current task ( $r$ ). This term penalizes the activation of neurons that have not yet been used by previous tasks through iterating over all masks with

$$\mathcal{L}' = \mathcal{L} + \lambda \cdot \frac{\|m^{(r)} \otimes (1 - m^{(prev)})\|_1}{\|1 - m^{(prev)}\|_1} \quad (8)$$

where  $\lambda$  is a weighting hyperparameter. Note that this is computed per LoRA layer and per matrix (e.g.,  $A_l$  and  $B_l$ ), as described earlier.

### 3.5 Growing PEFT

Using a single masked PEFT module across all tasks allows the model to access knowledge acquired from previous tasks while mitigating catastrophic forgetting. However, since capacity must be reserved for future learning, only a subset of the remaining unallocated neurons can be used for training on each new task. As tasks accumulate and more neurons are committed, the risk of exhausting the capacity of the module increases. To address this limitation, we propose an *expandable* PEFT module that dynamically increases its capacity according to the demands of the CL process. This approach preserves knowledge transfer and catastrophic forgetting prevention, while allowing the model to scale its capacity.

*Expandable LoRA* Given a LoRA module with weights  $A \in \mathbb{R}^{d \times a}$ ,  $B \in \mathbb{R}^{a \times k}$ , we dynamically expand the capacity by introducing additional neurons  $A_{\text{exp}} \in \mathbb{R}^{d \times a_{\text{exp}}}$ ,  $B_{\text{exp}} \in \mathbb{R}^{a_{\text{exp}} \times k}$  by

$$A' = [A^\top | A_{\text{exp}}^\top]^\top, \quad B' = [B | B_{\text{exp}}], \quad (9)$$

where  $[\cdot | \cdot]$  denotes the concatenation operation along the first dimension,  $a$  represents the current LoRA rank, and  $a_{\text{exp}}$  denotes the number of neurons added to expand the layer size. Note that we only extend the intermediate dimension  $a$ , as both  $d$  and  $k$  are fixed by the frozen LLM. When using our proposed gating approach in conjunction with expandable LoRA, the dimensionality of all masks  $m^{(r)}$  is adjusted by extending the task embedding  $e^{(r)}$  with  $e_{\text{exp}}^{(r)} \in \mathbb{R}^{a_{\text{exp}}}$  through

$$e'^{(r)} = [e^{(r)} | e_{\text{exp}}^{(r)}]. \quad (10)$$

*Expansion Strategy* While various strategies of differing complexity exist for determining when to expand neural networks, including predefined schedules [7], we adopt a simpler and widely used approach that triggers expansion when the validation loss reaches a plateau [17,51]. Once stagnation is detected, an expansion step is performed by adding additional trainable parameters to the PEFT module, after which training continues on the same task. This is repeated until no further improvement in validation performance is observed after expansion. The validation loss is evaluated at the end of each epoch and early stopping is applied with a patience of five epochs, following [45].

### 3.6 Extension to CIL

The Masked PEFT and Expandable PEFT architectures described in Sections 3.4 and 3.5 inherently require knowledge of the current relation type or set of relation types ( $r$ ) to apply the appropriate task-specific gating mask  $m^{(r)}$ . To adapt our method to the more challenging CIL scenario for continual relation extraction, where the task of an incoming sample is unknown, we must first infer ( $r$ ). To achieve this, we integrate a parameter-free task identification mechanism inspired by the work of Wang et al. [48]. Their approach uses a pre-trained LLM to distinguish between tasks without requiring additional trainable parameters for the identification step itself. In our context, for an incoming test sample, we first obtain its embedding representation, denoted as  $h(x)$ , by processing it through the frozen backbone LLM without any PEFT modules. During the training phase for each relation type set ( $r$ ), we compute and store the mean representation  $\mu_c^{(r)}$  for each class  $c$  within that relation type set, based on the  $h(x)$  vectors of its training instances. Furthermore, a single covariance matrix  $\Sigma$  is estimated and shared across all classes and tasks. At inference time, given a new sample and its representation  $h(x)$ , we calculate the Mahalanobis distance between  $h(x)$  and each stored class mean  $\mu_c^{(r)}$ :

$$\text{Dist}(h(x), \mu_c^{(r)}) = (h(x) - \mu_c^{(r)})^T \Sigma^{-1} (h(x) - \mu_c^{(r)}) \quad (11)$$

$$r^* = \arg \min_r \text{Dist}(h(x), \mu_c^{(r)}) \quad (12)$$

The relation type set ( $r^*$ ) corresponding to the class  $c$  that yields the minimum Mahalanobis distance is selected as the identified relation type set for the current sample. This inferred relation type set ( $r^*$ ) is then used to retrieve and apply the relevant gating mask  $m^{(r^*)}$  within our Masked PEFT module, processing the sample with the appropriate task-specific configuration in a CIL setting.

## 4 Experimental Setup

We evaluate our proposed solution on two core automated KGC tasks: continual link prediction and relation extraction. This section details formal task definitions, the datasets used, and the baseline models for comparison.

#### 4.1 Task- and Class-Incremental Learning

In our experiments, we use two fundamental paradigms of CL.

*Task-Incremental Learning (TIL)* In TIL, the model is trained on a sequence of tasks, each associated with its own set of classes. Here, the task identifier is known during both training and inference. This allows the model to exploit task-specific decision boundaries. Formally, the model is trained sequentially on a series of tasks  $R = \{(r_1), (r_2), \dots, (r_N)\}$ . Each task  $(r_t)$  corresponds to learning a specific relation type or set of relation types and is associated with a dataset  $D^{(r_t)} = \{(x_j, c_j)\}_{j=1}^{|D^{(r_t)}|}$ , where  $x_j$  is an input sample with its corresponding class label  $c_j$ . At inference time, the model is provided with both an input sample  $x$  and the identifier  $(r)$  of the task it belongs to. The objective is then to predict the correct label  $c \in C^{(r)}$ .

*Class-Incremental Learning (CIL)* In CIL, training is conducted as in TIL. However, at inference time, the model is not provided with task identifiers and must instead classify among all previously encountered classes. Formally, given an input sample  $x$ , the model must predict its correct class label  $c$  from the union of all classes encountered across all tasks seen so far,  $\bigcup_{t=1}^N C^{(r_t)}$  without access to the task identifier  $(r)$ .

#### 4.2 Continual Link Prediction (CLP)

Knowledge Graphs (KGs) represent facts as triples  $\langle h_{KG}, r_{KG}, t_{KG} \rangle$ , linking head  $h_{KG}$  and tail  $t_{KG}$  entities via a relation type  $r_{KG}$ . Since KGs are often incomplete, the link prediction task aims to infer missing entities. Formally, given an incomplete query such as  $\langle h_{KG}, r_{KG}, ? \rangle$ , the objective is to learn a function  $f_{LP}$  that predicts the plausible missing tail entity  $t_{KG}$ :

$$f_{LP}(h_{KG}, r_{KG}) \rightarrow t_{KG}. \quad (13)$$

In our experiments, we frame link prediction within the TIL setting, where the specific relation type  $r_{KG}$  directly serves as the task identifier  $(r)$ .

#### 4.3 Continual Relation Extraction (CRE)

CRE aims to extract fact triples  $\langle h_{KG}, r_{KG}, t_{KG} \rangle$  from natural language. Formally, given an input text  $x$  containing mentions of two entities,  $h_{KG}$  and  $t_{KG}$ , the objective is to learn a function  $f_{RE}$  that predicts the specific relation type  $r_{KG}$  holding between them:

$$f_{RE}(x, h_{KG}, t_{KG}) \rightarrow r_{KG}. \quad (14)$$

In our experiments, relation extraction is conducted in a CIL setting, as the input text  $x$  typically does not provide explicit relation information.

#### 4.4 Datasets

For our CLP experiments, we follow [28] and use three well-established datasets WN18 [2], YAGO3-10 [23,38], and FB15k [37]. WN18 has a hierarchical structure, with most triples representing hypernym and hyponym relations. YAGO3-10 primarily contains descriptive facts about individuals, such as their place of birth or profession. FB15k covers a broad range of factual knowledge across various domains, including people, locations, movies, and more. To ensure direct comparability, we apply the same data preprocessing steps as in [28]. For our CRE experiments, we employ two widely used benchmark datasets FewRel [10] and TACRED [54]. FewRel contains 80 relation types with 700 instances each. To ensure comparability, we follow the methodology of [5], splitting the data into 10 subsets, each containing 8 distinct relation types. TACRED, in contrast, is an imbalanced dataset containing 42 relation types. Consistent with [5], we exclude the no\_relation class and partition the dataset into 10 subsets, each with 4 relation types. We also use the task order and data splits as in [5].

#### 4.5 Input Prompts

For CLP, to adjust the LLM to the new KG relation type, we use the fixed heuristics from [28] to convert each KG triple into natural language. For instance, the triple  $\langle \text{Leonardo\_da\_Vinci wasBornIn ?} \rangle$  is converted into *Leonardo da Vinci was born in*. This input is then passed to the model for text completion. Completions are then evaluated through exact match accuracy.

For CRE, we follow the strategy from [39]. Given a sentence and an entity pair  $\langle h_{KG}, t_{KG} \rangle$ , the input prompt is constructed by concatenating the sentence with the natural language question “*The relation between  $h_{KG}$  and  $t_{KG}$  is? Answer:*”. Results are evaluated with the standard continual learning protocol, reporting classification accuracy after the completion of each newly learned task.

#### 4.6 Baselines

For CLP, we compare against several approaches. **LLaMa<sub>notuning</sub>** serves as a zero-shot baseline without any fine-tuning on the underlying LLaMA 2-7B model. **Per-Task-Adapter** and **Per-Task-LoRA** represent parameter isolation strategies where a separate Adapter or LoRA module is trained for each relation type, freezing the backbone LLM. In contrast, **Seq-FT-Adapter** and **Seq-FT-LoRA** sequentially fine-tune a single, shared Adapter or LoRA module across all relations, risking catastrophic forgetting. We also include methods specifically designed for continual KG completion: **CapsKG** [28] leverages capsule-based routing, and **MoCL** [45] use LoRA modules which are initialized with a weighted sum of LoRAs from previously learned tasks [45]. For CRE, we include the following baselines. **EA-EMR** [43] utilizes memory replay combined with embedding alignment. **EMAR** [9] employs a memory activation and re-consolidation strategy. **CML** [52] leverages curriculum-meta learning to handle

task order sensitivity and forgetting. **RP-CRE** [5] uses relation prototypes to refine sample representations and preserve knowledge. **CREST** [20] mitigates catastrophic forgetting by managing the disparate gradient signals of different learning objectives. **EoE** [57] is a rehearsal-free method that enhances task identification through discriminative training and uses a cascade voting mechanism to learn new relations. Finally, we also use **MoCL** as parameter isolation-based CRE baseline, as it is inherently capable of CIL.

#### 4.7 Hyperparameter and implementation details

We use the AdamW optimizer [22]. For CLP, following [28], and for CRE, following [20], we use a batch size of 16 for all experiments. The initial LoRA rank is set to 4, based on prior work [45]. We initialize LoRA’s  $A$  and  $A_{exp}$  matrices using the Kaiming uniform distribution [11], and LoRA’s  $B$  and  $B_{exp}$  with zeros, following [13]. For dynamic LoRA, we extend the LoRA’s intermediate rank. Early stopping is applied with a patience of 5 steps on the development set. The supplementary material and our code repository provide detailed hyperparameter settings for CGKGC on CLP and CRE across datasets to foster reproducibility. Training is conducted using the LLaMA 2-7B pre-trained LLM on a single NVIDIA A100 GPU.

## 5 Results

### 5.1 CRE Results

In these experiments, we evaluate CGKGC on the two established benchmark datasets TACRED and FewRel, comparing it against a range of rehearsal-based and rehearsal-free baselines, as well as state-of-the-art (SoTA) methods. Notably, we report the established average accuracy after each trained relation set  $r$ , which re-evaluates the accuracy of all trained relation sets after training on each new relation set.

Tables 1 and 2 present CRE results on TACRED and FewRel, respectively, with both results showing similar observations. All rehearsal-based methods, including the rehearsal-based SoTA method CREST, consistently underperform compared to CGKGC. Results on later relation sets also show strong performance decreases, indicating that these methods suffer from significant catastrophic forgetting. The current rehearsal-free SoTA method EoE outperforms existing rehearsal-based approaches but still shows decreasing average accuracies in continued training. MoCL, originally designed for continual text classification and known for its strong performance, yields promising results that surpass existing CRE-specific SoTA solutions. Yet, our CGKGC outperforms all baselines. While the relation-type classification of our basic CIL implementation identifies the correct LoRA modules in 70% of cases, final CRE accuracies are considerably higher, indicating that even incorrect LoRA modules are capable of correct CRE due to inherent LM knowledge and knowledge transfer between LoRA modules.

Table 1: Average accuracy of CIL on the TACRED dataset evaluated after training each relation set  $r$ , averaged over five sequences. Highest accuracies in bold.

Model	$r_1$	$r_2$	$r_3$	$r_4$	$r_5$	$r_6$	$r_7$	$r_8$	$r_9$	$r_{10}$
EA-EMR	47.5	40.1	38.3	29.9	24	27.3	26.9	25.8	22.9	19.8
EMAR	73.6	57.0	48.3	42.3	37.7	34.0	32.6	30.0	27.6	25.1
CML	57.2	51.4	41.3	39.3	35.9	28.9	27.3	26.9	24.8	23.4
EMAR+BERT	96.6	85.7	81.0	78.6	73.9	72.3	71.7	72.2	72.6	71.0
RP-CRE	97.8	92.3	91.0	87.3	84.2	82.7	79.8	78.8	78.6	77.3
CREST	97.3	91.4	82.3	82.5	79.2	75.8	78.8	77.4	78.6	79.4
EoE	<b>98.7</b>	94.7	90.6	87.8	87.2	85.9	84.3	83.2	82.7	81.5
MoCL	96.4	<b>96.6</b>	95.4	95.0	94.9	94.6	94.6	94.5	94.6	94.8
CGKGC	96.7	<b>96.6</b>	<b>96.1</b>	<b>96.1</b>	<b>96.2</b>	<b>96.0</b>	<b>96.1</b>	<b>96.2</b>	<b>96.3</b>	<b>96.3</b>

Table 2: Average accuracy of CIL on the FewRel dataset evaluated after training each relation set  $r$ , averaged over five sequences. Highest accuracies in bold.

Model	$r_1$	$r_2$	$r_3$	$r_4$	$r_5$	$r_6$	$r_7$	$r_8$	$r_9$	$r_{10}$
EA-EMR	89.0	69.0	59.1	54.2	47.8	46.1	43.1	40.7	38.6	35.2
EMAR	88.5	73.2	66.6	63.8	55.8	54.3	52.9	50.9	48.8	46.3
CML	91.2	74.8	68.2	58.2	53.7	50.4	47.8	44.4	43.1	39.7
EMAR+BERT	98.8	89.1	89.5	85.7	83.6	84.8	79.3	80.0	77.1	73.8
RP-CRE	98.4	95.2	93.1	91.4	90.8	88.8	87.6	86.8	85.2	83.9
CREST	<b>98.7</b>	93.6	93.8	92.3	91.0	89.9	87.6	86.7	86.0	84.8
EoE	97.8	95.0	93.6	92.5	91.6	90.0	88.9	87.9	86.9	85.5
MoCL	96.9	93.9	93.5	93.7	94.2	94.0	94.1	94.2	94.6	94.4
CGKGC	97.0	<b>96.3</b>	<b>96.7</b>	<b>96.8</b>	<b>96.9</b>	<b>97.0</b>	<b>96.9</b>	<b>96.9</b>	<b>96.9</b>	<b>96.9</b>

Compared to MoCL, CGKGC achieves higher average accuracies in later stages of training, suggesting that continuous access to parameters reserved by previous relation types allows efficient knowledge transfer.

## 5.2 CLP results

In this experiment we evaluate CGKGC on CLP. Following Omelivanenko et al. [28] we fine-tune CGKGC sequentially on individual relation types, reporting average accuracy after all relation types are trained.

Results can be found in Table 3. Without fine-tuning, the pretrained LLaMA 2-7B base model is unable to identify suitable target entities. Fully parameter-isolated PEFT modules for each relation achieve higher scores, fully preventing catastrophic forgetting by design. Here, LoRA outperforms the use of simple Adapters. Without full parameter isolation, sequentially training on a single

Table 3: Average accuracies of CLP on the WN18, YAGO3-10, and FB15k datasets. Results are averaged over five random seeds, with the highest accuracies highlighted in bold.

Model/Dataset	WN18	YAGO3-10	FB15k
LLaMa <sub>notuning</sub>	0.4 ± 0.0	0.1 ± 0.0	0.3 ± 0.0
Per-Task-Adapter	18.8 ± 0.2	44.9 ± 0.7	32.2 ± 0.6
Per-Task-LoRA	26.8 ± 0.8	56.8 ± 0.6	35.6 ± 0.6
Seq-FT-Adapter	13.6 ± 0.4	26.1 ± 1.1	6.7 ± 1.1
Seq-FT-LoRA	16.6 ± 0.2	31.3 ± 5.4	13.9 ± 4.1
CapsKG <sub>forward</sub>	19.7 ± 0.4	46.4 ± 0.2	33.7 ± 0.1
CapsKG <sub>backward</sub>	19.1 ± 0.4	46.3 ± 0.2	32.2 ± 0.2
MoCL	26.8 ± 0.7	57.1 ± 0.4	36.4 ± 0.5
CGKGC	<b>27.4 ± 1.2</b>	<b>58.5 ± 0.5</b>	<b>36.7 ± 0.2</b>

PEFT module yields considerably lower accuracy due to catastrophic forgetting, as only the original model parameters are protected by the PEFT module while knowledge of previously learned relation types is overwritten. CapsKG improves on the Adapter solution it builds upon through its knowledge sharing approach while preventing most catastrophic forgetting. The continual text classification approach MoCL outperforms the other baselines, even offering slight improvements over individual per-task PEFT modules due to its knowledge transfer at initialization. Our CGKGC further improves on these results, outperforming all baselines on all datasets. Calculating forward transfer scores in Section A.3 shows knowledge sharing improvements over MoCL without catastrophic forgetting. For closer inspection of the learned task masks for each relation we provide similarity heatmaps of learned task masks in the supplementary code repository. Our investigations reveal that CGKGC is able to share more neurons between semantically similar relations.

### 5.3 Ablation Study

To obtain a deeper understanding of the behavior of CGKGC, we conduct additional experiments on all three CLP datasets by varying the sparsity regularization parameter  $\lambda$  and the growth size per growing step (see Equations (8) to (10)).

We report performances and final parameter usage across multiple hyperparameter configurations for the FB15k dataset in Tables 4 and 5, respectively. For the FB15k dataset with the highest number of relations, without growth, a certain level of regularization is required to prevent early saturation. For excessively large regularizations  $\lambda$ , performance decreases considerably even with growing enabled, with the used parameter quota showing that the model refrains from using the available parameters. Increasing growth rate directly results in lower final quotas of used parameters. Similar patterns can be observed in the

ablations on the WN18 and YAGO3-10 datasets available in the supplementary code repository. Throughout all datasets, high performance is achieved with low regularizations  $\lambda$  and varying growth rates, suggesting that a suitable combination of both parameters, while dataset dependent, requires a low regularization to allow the efficient use of available parameters.

Our solution utilizes iterative training loops until convergence. In our experiments, this approach required approximately 21% more training time than MoCL and 47% more than Per-Task-LoRA. However, this trade-off results in models with up to four times fewer parameters, as detailed in our detailed final parameter comparison in Appendix A.4. CGKGC achieves consistently higher parameter efficiency compared to the final parameter counts of MoCL and can adapt to dataset complexity instead of growing purely with increasing numbers of tasks. Additionally, while we maintained a fixed batch size for comparability in our demonstrations, our higher parameter efficiency can be leveraged with a dynamic batch size for even faster training.

Table 4: Performance of CGKGC with varying  $\lambda$  and growth rate on FB15k. Results show high accuracies for low regularization rates and variable growth rates. High or no regularization causes considerable accuracy decrease.

Avg. $\lambda \backslash$ Growth	Average Accuracy				
	0%	25%	50%	75%	100%
0	18.22	36.66	36.12	36.27	36.07
0.1	33.55	36.23	35.97	36.28	36.26
1	34.45	36.29	36.37	36.21	36.45
10	34.22	35.96	36.06	35.82	36.02
100	32.91	31.19	31.58	31.34	32.87

Table 5: Parameter usage of CGKGC with varying  $\lambda$  and growth rate on FB15k. Results show high parameter usage and minor increases in final model size for slow growth rate, indicating that low regularization is sufficient to enforce high parameter usage.

Avg. $\lambda \backslash$ Growth	Used-Parameter Quota (Final LoRA Rank)				
	0%	25%	50%	75%	100%
0	100 (4.0)	82.11 (13.8)	76.52 (20.4)	65.67 (34.6)	56.39 (46.4)
0.1	100 (4.0)	81.65 (14.2)	73.11 (25.6)	65.20 (35.8)	52.68 (51.2)
1	99.9 (4.0)	81.80 (14.4)	78.07 (19.2)	70.78 (28.6)	62.57 (39.2)
10	99.7 (4.0)	80.40 (16.0)	72.49 (26.0)	70.78 (28.0)	60.10 (41.6)
100	99.1 (4.0)	81.03 (15.2)	72.18 (26.4)	67.53 (31.6)	49.59 (55.2)

## 6 Conclusion

We present CGKGC, a novel approach for continual knowledge graph completion that combines masked parameter-efficient fine-tuning with dynamic growth mechanisms. Our solution entirely prevents catastrophic forgetting by design while enabling continuous knowledge transfer between tasks. The growing mechanism successfully mitigates parameter saturation, allowing efficient adaptation to new knowledge domains. Experimental results demonstrate superior performance over the state-of-the-art in both task-incremental and class-incremental learning settings for continual link prediction and relation extraction.

While CGKGC leverages parameter-efficient fine-tuning modules to remain highly parameter-efficient even on longer task sequences, investigations to further reduce the necessary parameter growth on new tasks is of interest. Additionally, we employ a simple, parameter-free approach to task identification in the class-incremental setting to showcase the feasibility of CGKGC through strong results. A promising direction for future work is the exploration of other task identification mechanisms to further improve performance. Similarly, we plan to explore more growth strategies beyond the simple loss-based convergence strategy that already provided state-of-the-art results.

*Acknowledgements:* This work is funded by the Bavarian Ministry of Economic Affairs, Regional Development and Energy through DEMAnD-LM project (grant no. DIK-2506-0011 // DIK0887/03).

*Supplemental Material Statement:* Our source code is available here: <https://professor-x.de/code-cgkgc>. The used CLP datasets are publicly available at <https://everest.hds.utc.fr/doku.php?id=en:transe> for WN18 and FB15k, and <https://pykeen.readthedocs.io/> for YAGO3-10. The used CRE datasets can be found at <https://nlp.stanford.edu/projects/tacred/> for TACRED and at <https://github.com/thunlp/FewRel> for FewRel. Hyperparameters, dataset statistics, convenience overviews on final parameter counts and forward transfer scores are reported in the appendix for easy reproducibility.

## References

1. Aljundi, R., Babiloni, F., Elhoseiny, M., Rohrbach, M., Tuytelaars, T.: Memory aware synapses: Learning what (not) to forget. In: Proceedings of the European conference on computer vision (ECCV). pp. 139–154 (2018)
2. Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems* **26** (2013)
3. Chaudhry, A., Gordo, A., Dokania, P., Torr, P., Lopez-Paz, D.: Using hindsight to anchor past knowledge in continual learning. In: Proceedings of the AAAI conference on artificial intelligence. vol. 35, pp. 6993–7001 (2021)
4. Chaudhry, A., Ranzato, M., Rohrbach, M., Elhoseiny, M.: Efficient lifelong learning with a-gem. arXiv preprint arXiv:1812.00420 (2018)

5. Cui, L., Yang, D., Yu, J., Hu, C., Cheng, J., Yi, J., Xiao, Y.: 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). pp. 232–243 (2021)
6. Cui, Y., Wang, Y., Sun, Z., Liu, W., Jiang, Y., Han, K., Hu, W.: Lifelong embedding learning and transfer for growing knowledge graphs. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 37, pp. 4217–4224 (2023)
7. Evci, U., van Merriënboer, B., Unterthiner, T., Vladymyrov, M., Pedregosa, F.: Gradmax: Growing neural networks using gradient information. arXiv preprint arXiv:2201.05125 (2022)
8. Fichtel, L., Kalo, J.C., Balke, W.T.: Prompt Tuning or Fine-Tuning - Investigating Relational Knowledge in Pre-Trained Language Models. In: 3rd Conference on Automated Knowledge Base Construction (Sep 2021)
9. Han, X., Dai, Y., Gao, T., Lin, Y., Liu, Z., Li, P., Sun, M., Zhou, J.: Continual relation learning via episodic memory activation and reconsolidation. In: Proceedings of the 58th annual meeting of the association for computational linguistics. pp. 6429–6440 (2020)
10. Han, X., Zhu, H., Yu, P., Wang, Z., Yao, Y., Liu, Z., Sun, M.: 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. pp. 4803–4809 (2018)
11. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: Proceedings of the IEEE international conference on computer vision. pp. 1026–1034 (2015)
12. Houlshby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., De Laroussilhe, Q., Gesmundo, A., Attariyan, M., Gelly, S.: Parameter-efficient transfer learning for nlp. In: International conference on machine learning. pp. 2790–2799. PMLR (2019)
13. Hu, E.J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., Chen, W.: Lora: Low-rank adaptation of large language models. arXiv preprint arXiv:2106.09685 (2021)
14. Huang, Y., Zhang, Y., Chen, J., Wang, X., Yang, D.: Continual learning for text classification with information disentanglement based regularization. arXiv preprint arXiv:2104.05489 (2021)
15. Ke, Z., Lin, H., Shao, Y., Xu, H., Shu, L., Liu, B.: Continual training of language models for few-shot learning. In: Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing. pp. 10205–10216 (2022)
16. Ke, Z., Liu, B., Ma, N., Xu, H., Shu, L.: Achieving forgetting prevention and knowledge transfer in continual learning. *Advances in Neural Information Processing Systems* **34**, 22443–22456 (2021), supplementary material
17. Kilcher, Y., Bécigneul, G., Hofmann, T.: Escaping flat areas via function-preserving structural network modifications (2018)
18. Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A.A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al.: Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences* **114**(13), 3521–3526 (2017)
19. Le, M., Luu, T.N., Le, T.T., Nguyen, T., Nguyen, T.T., Van, L.N., Nguyen, T.H., et al.: Adaptive prompting for continual relation extraction: A within-task variance perspective. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 39, pp. 24384–24392 (2025)

20. Le, T.T., Nguyen, M., Nguyen, T.T., Van, L.N., Nguyen, T.H.: Continual relation extraction via sequential multi-task learning. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 38, pp. 18444–18452 (2024)
21. Liu, J., Ke, W., Wang, P., Wang, J., Gao, J., Shang, Z., Li, G., Xu, Z., Ji, K., Li, Y.: Fast and continual knowledge graph embedding via incremental lora. CoRR (2024)
22. Loshchilov, I.: Decoupled weight decay regularization. arXiv preprint arXiv:1711.05101 (2017)
23. Mahdisoltani, F., Biega, J., Suchanek, F.M.: Yago3: A knowledge base from multilingual wikipedias. In: CIDR (2013)
24. de Masson D’Autume, C., Ruder, S., Kong, L., Yogatama, D.: Episodic memory in lifelong language learning. *Advances in Neural Information Processing Systems* **32** (2019)
25. Mirzadeh, S.I., Farajtabar, M., Gorur, D., Pascanu, R., Ghasemzadeh, H.: Linear mode connectivity in multitask and continual learning. arXiv preprint arXiv:2010.04495 (2020)
26. Nguyen, H., Nguyen, C., Ngo, L., Tuan, L.A., Nguyen, T.: A spectral viewpoint on continual relation extraction. In: Findings of the Association for Computational Linguistics: EMNLP 2023. pp. 9621–9629 (2023)
27. Nguyen, T.H., Grishman, R.: Relation extraction: Perspective from convolutional neural networks. In: Proceedings of the 1st workshop on vector space modeling for natural language processing. pp. 39–48 (2015)
28. Omeliyanenko, J., Zehe, A., Hotho, A., Schlör, D.: Capskg: enabling continual knowledge integration in language models for automatic knowledge graph completion. In: International Semantic Web Conference. pp. 618–636. Springer (2023)
29. Petroni, F., Rocktäschel, T., Riedel, S., Lewis, P., Bakhtin, A., Wu, Y., Miller, A.: Language models as knowledge bases? In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP). pp. 2463–2473 (2019)
30. Qin, C., Joty, S.: Lfpt5: A unified framework for lifelong few-shot language learning based on prompt tuning of t5. In: International Conference on Learning Representations (2022)
31. Razdaibiedina, A., Mao, Y., Hou, R., Khabsa, M., Lewis, M., Almahairi, A.: Progressive prompts: Continual learning for language models. arXiv preprint arXiv:2301.12314 (2023)
32. Rebuffi, S.A., Kolesnikov, A., Sperl, G., Lampert, C.H.: icarl: Incremental classifier and representation learning. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 5533–5542. IEEE (2017)
33. Rusu, A.A., Rabinowitz, N.C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., Pascanu, R., Hadsell, R.: Progressive neural networks (2016)
34. Serra, J., Suris, D., Miron, M., Karatzoglou, A.: Overcoming catastrophic forgetting with hard attention to the task. In: International conference on machine learning. pp. 4548–4557. PMLR (2018)
35. Soares, L.B., FitzGerald, N., Ling, J., Kwiatkowski, T.: Matching the blanks: Distributional similarity for relation learning. arXiv preprint arXiv:1906.03158 (2019)
36. Sun, F.K., Ho, C.H., Lee, H.Y.: Lamol: Language modeling for lifelong language learning. In: International Conference on Learning Representations
37. Toutanova, K., Chen, D.: Observed versus latent features for knowledge base and text inference. In: Proceedings of the 3rd workshop on continuous vector space models and their compositionality. pp. 57–66 (2015)

38. Tran, H.N., Takasu, A.: Meim: Multi-partition embedding interaction beyond block term format for efficient and expressive link prediction. In: Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence. International Joint Conferences on Artificial Intelligence Organization (2022)
39. Tran, Q., Thanh, N.X., Anh, N.H., Hai, N.L., Le, T., Ngo, L.V., Nguyen, T.H.: Preserving generalization of language models in few-shot continual relation extraction. In: Al-Onaizan, Y., Bansal, M., Chen, Y.N. (eds.) Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing. pp. 13771–13784. Association for Computational Linguistics, Miami, Florida, USA (Nov 2024)
40. Valipour, M., Rezagholizadeh, M., Kobzyev, I., Ghodsi, A.: Dylora: Parameter-efficient tuning of pre-trained models using dynamic search-free low-rank adaptation. In: Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics. pp. 3274–3287 (2023)
41. Veyseh, A., Deroncourt, F., Dou, D., Nguyen, T.: A joint model for definition extraction with syntactic connection and semantic consistency. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 34, pp. 9098–9105 (2020)
42. Veyseh, A.P.B., Deroncourt, F., Dou, D., Nguyen, T.H.: Exploiting the syntax-model consistency for neural relation extraction. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. pp. 8021–8032 (2020)
43. Wang, H., Xiong, W., Yu, M., Guo, X., Chang, S., Wang, W.Y.: 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). pp. 796–806 (2019)
44. Wang, L., Zhang, X., Su, H., Zhu, J.: A comprehensive survey of continual learning: theory, method and application. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2024)
45. Wang, M., Adel, H., Lange, L., Strötgen, J., Schütze, H.: Rehearsal-free modular and compositional continual learning for language models. In: Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 2: Short Papers). pp. 469–480 (2024)
46. Wang, R., Tang, D., Duan, N., Wei, Z., Huang, X., Ji, J., Cao, G., Jiang, D., Zhou, M.: K-Adapter: Infusing Knowledge into Pre-Trained Models with Adapters (Dec 2020)
47. Wang, X., Chen, T., Ge, Q., Xia, H., Bao, R., Zheng, R., Zhang, Q., Gui, T., Huang, X.: Orthogonal subspace learning for language model continual learning. In: The 2023 Conference on Empirical Methods in Natural Language Processing
48. Wang, Z., Liu, Y., Ji, T., Wang, X., Wu, Y., Jiang, C., Chao, Y., Han, Z., Wang, L., Shao, X., Zeng, W.: Rehearsal-free continual language learning via efficient parameter isolation. In: Rogers, A., Boyd-Graber, J., Okazaki, N. (eds.) Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 10933–10946. Association for Computational Linguistics, Toronto, Canada (Jul 2023)
49. Wortsman, M., Ramanujan, V., Liu, R., Kembhavi, A., Rastegari, M., Yosinski, J., Farhadi, A.: Supermasks in superposition. *Advances in Neural Information Processing Systems* **33**, 15173–15184 (2020)
50. Wu, H., Yin, J., Bui, T.D., Rajaratnam, B.: Continual knowledge graph link prediction: Beyond experience replay
51. Wu, L., Wang, D., Liu, Q.: Splitting steepest descent for growing neural architectures. *Advances in neural information processing systems* **32** (2019)

52. Wu, T., Li, X., Li, Y.F., Haffari, G., Qi, G., Zhu, Y., Xu, G.: Curriculum-meta learning for order-robust continual relation extraction. In: Proceedings of the AAAI conference on artificial intelligence. vol. 35, pp. 10363–10369 (2021)
53. Zhang, Q., Chen, M., Bukharin, A., He, P., Cheng, Y., Chen, W., Zhao, T.: Adaptive budget allocation for parameter-efficient fine-tuning. In: The Eleventh International Conference on Learning Representations
54. Zhang, Y., Zhong, V., Chen, D., Angeli, G., Manning, C.D.: Position-aware attention and supervised data improve slot filling. In: Conference on empirical methods in natural language processing (2017)
55. Zhao, A., Yu, Y.: Knowledge-enabled bert for aspect-based sentiment analysis. *Knowledge-Based Systems* **227**, 107220 (2021)
56. Zhao, W., Cui, Y., Hu, W.: 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). pp. 1162–1175 (2023)
57. Zhou, S., Li, Y., Miao, X., Qian, T.: An ensemble-of-experts framework for rehearsal-free continual relation extraction. In: Findings of the Association for Computational Linguistics ACL 2024. pp. 1410–1423 (2024)

## A Supplementary Material

### A.1 Dataset Characteristics

The characteristics of the continual link prediction and relation extraction datasets used in our experiments are summarized in Table 6 and Table 7, respectively.

Table 6: CLP datasets used, following the preprocessing steps in [28].

Dataset	relation types	entities	train triples	dev triples	test triples	total
WN18	5	23324	27045	9017	9018	45080
YAGO3-10	8	92991	75774	25260	25262	126296
FB15k	18	10754	38178	12734	12736	63648

Table 7: Relation extraction datasets used in this study, following [5].

Dataset	relation types	train	dev	test	total
TACRED	42	68124	22631	15509	106264
FewRel	80	33600	11200	11200	56000

### A.2 Hyperparameters

The detailed hyperparameters used for training CGKGC on each dataset contained in our experiments are listed in Table 8.

Table 8: CGKGC hyperparameters for each dataset.

Hyperparameters	WN18	YAGO3-10	FB15k	TACRED	FewRel
Epochs	40	40	40	40	40
Early stop patience	5	5	5	5	5
Learning rate	1e-3	1e-4	1e-4	1e-4	1e-4
Max. sequence len.	32	32	32	256	256
LoRA scaling factor $\alpha$	16	16	16	16	16
Batch size	16	16	16	16	16
Growth %	0.75	1	0.25	0.25	0.25
$\lambda$	0.1	0	0	1	1

Table 9: Forward transfer scores according to [45]. Higher is better. Best results highlighted in bold.

Model/Dataset	WN18	YAGO3-10	FB15k
MoCL	0.1423	0.3327	0.9195
CGKGC	<b>0.9114</b>	<b>1.7935</b>	<b>1.1799</b>

### A.3 Forward Transfer Scores

To evaluate the average influence of previous tasks on the performance of the current task, Wang et al. [45] calculate forward transfer scores by computing the average percent point improvement of each task within a TIL sequence compared to the performance achieved by training the same task without previous tasks. We compute these forward transfer scores in for the CLP setting in Table 9. While the previous SOTA method MoCL achieves non-negative scores on all datasets, indicating that it is able to avoid catastrophic forgetting, our CGKGC is able to consistently outperform MoCL on all datasets.

### A.4 Paramter Count Comparison

We calculate the final parameter count of both MoCL and CGKGC in Table 10. Here, parameter counts are taken after training on the final task to directly compare final model size. MoCL grows to large parameter sizes on datasets with many tasks, reaching over 43M parameters on the FB15k dataset that contains 18 tasks. Conversely, our CGKGC uses less parameters on all datasets. Additionally, the growth mechanism of CGKGC appears to dynamically adapt the parameter needs to the complexity of the dataset instead of growing only through the number of tasks: CGKGC uses more parameters on the YAGO3-10 dataset with 8 relations than on the FB15k dataset with 18 relations. This highlights both the adaptability and parameter efficiency of CGKGC.

Table 10: Final parameter count comparison of MoCL and CGKGC on all datasets, taken after final task training, with considerably higher parameter efficiency of CGKGC. Lower is better. Best results in bold.

Model/Dataset	TACRED	FewRels	WN18	YAGO3-10	FB15k
MoCL	24 005 120	24 005 120	11 900 160	19 138 560	43 799 040
CGKGC	<b>6 505 907</b>	<b>8 290 662</b>	<b>6 556 800</b>	<b>15 532 032</b>	<b>12 074 752</b>