Large-scale Lifelong Learning of In-context Instructions and How to Tackle It

Jisoo Mok[1*] **Jaeyoung Do**[2†] **Sungjin Lee**[2] **Tara Taghavi**[2]
Seunghak Yu[3] **Sungroh Yoon**[1,4†]
[1] Department of ECE, Seoul National University [2] Amazon Alexa AI [3] NAVER Search US
[4] Interdisciplinary Program in AI, Seoul National University

Abstract

Jointly fine-tuning a Pre-trained Language Model (PLM) on a pre-defined set of tasks with in-context instructions has been proven to improve its generalization performance, allowing us to build a universal language model that can be deployed across task boundaries. In this work, we explore for the first time whether this attractive property of in-context instruction learning can be extended to a scenario in which tasks are fed to the target PLM in a sequential manner. The primary objective of so-called lifelong in-context instruction learning is to improve the target PLM's instance- and task-level generalization performance as it observes more tasks. DYNAINST, the proposed method to lifelong in-context instruction learning, achieves noticeable improvements in both types of generalization, nearly reaching the upper bound performance obtained through joint training.

1 Introduction

A number of recent studies have shown that Pre-trained Language Models (PLMs) only need to undergo a brief fine-tuning process to achieve remarkable instance-level generalization performance within the observed task (Brown et al., 2020). Compared to instance-level generalization within seen tasks, however, the zero-shot cross-task generalization capability of PLMs to unseen tasks is just starting to be explored (Sanh et al., 2021; Wei et al., 2021). In order to build a language model that can generalize across task boundaries and thus be deployed to various task scenarios, in-context instruction learning draws inspiration from how humans can familiarize themselves with a variety of language-related tasks only by following instructions (Mishra et al., 2022). To emulate the human learning process, in-context instruction learning trains the target PLM with both input-output pairs and a set of in-context instructions that contain additional task-specific information.

The introduction of in-context instructions is spearheading a noticeable progress in cross-task generalization within Natural Language Processing (NLP) research. The recent advances in-context instruciton learning (Ouyang et al., 2022; Mishra et al., 2022; Wang et al., 2022) indicate that fine-tuning a PLM with task-specific instructions has a positive impact on its cross-task generalization capability. One common limitation shared by the existing works is that they require a static, pre-defined set of tasks to jointly train the target PLM. Any learning paradigm bound by the assumption that all of the tasks are pre-defined and non-changing not only incurs a huge memory and computational cost but also raises serious data privacy concerns (De Lange et al., 2021). In this paper, we aim to address such concerns by studying whether it is possible to sequentially fine-tune the target PLM on a stream of large-scale instruction-paired tasks in a lifelong manner.

In the same spirit as joint in-context instruction learning, the primary objective of lifelong in-context instruction learning is to gradually improve both instance- and task-level generalization capabilities as the target PLM observes more train tasks. With the help of lifelong in-context instruction learning, deployment of a universal language model on edge devices with limited memory and computation power becomes easier. In such a resource-constrained setting, it is infeasible to jointly train a language model from scratch on all user data every time a new task is introduced. Moreover, it is difficult to deploy a separate model for every task or utilize an extremely large model with multi-task capability. Instead, we can deploy a continuously evolving language model and train it sequentially on a task stream.

To study the problem of lifelong in-context instruction learning within the context of instance-

[*] Work done while interning at Amazon Alexa AI (magicshop1118@snu.ac.kr)
[†] Corresponding Authors

and task-level generalization, we adopt Super-NaturalInstructions (Sup-NatInst) (Wang et al., 2022), which is the largest dataset with in-context instructions to date, and restructure it accordingly. Because we believe that cross-language generalization is beyond the scope of this paper, we use the English subset of Sup-NatInst, and from here on, we use Sup-NatInst to refer to the English subset instead of the entire dataset. More details on the characteristics of Sup-NatInst and the data restructuring process are provided in Section 3.

Our proposed approach to lifelong in-context instruction learning, DYNAINST, combines parameter regularization and experience replay. The regularizer employed by DYNAINST is designed to induce wide local minima in the target PLM. Deep neural networks with wide local minima are known to achieve improved generalization performance and become more robust against task distribution shifts (Cha et al., 2020); these two advantages of wide local minima are well-aligned with the objectives of lifelong instruction learning, making it a particularly attractive choice of regularization. To design a memory- and compute-efficient experience replay framework, we devise Dynamic Instruction Replay, which is comprised of Dynamic Instance Selection (DIS) and Dynamic Task Selection (DTS). DIS and DTS flexibly determine which instances and tasks are stored and replayed, respectively. Our experimental results demonstrate that DYNAINST outperforms strong baselines in both instance- and task-level generalization under various experimental scenarios.

Our contributions can be summarized as follows:

- This is the first work to study the potential of lifelong in-context instruction learning as an efficient framework towards building a continuously evolving universal language model.

- We propose DYNAINST, a hybrid approach to lifelong in-context instruction learning that integrates a wide local minima-inducing regularizer and Dynamic Instruction Replay. With extensive experimental results, we verify that DYNAINST outperforms existing baselines from continual learning.

- We present a series of empirical analyses and ablation studies that offer further insights into lifelong in-context instruction learning and the inner-workings of DYNAINST.

Characteristic	ConTinTin	Ours.
In-Context instructions	✓	✓
Fully continual	✗	✓
Zero-shot generalization	✗	✓
Large-scale	✗	✓
Changing # of instances	✗	✓

Table 1: Comparison of ConTinTin (Yin et al., 2022) *vs.* our framework. Our learning objective and problem definition are clearly distinguishable from ConTinTin.

2 Related Works

2.1 Lifelong Learning

Lifelong learning (De Lange et al., 2021; McCloskey and Cohen, 1989) concerns with the problem of learning from a continuous stream of data (Parisi et al., 2019; Chen and Liu, 2018). Thus, what distinguishes lifelong learning from the conventional paradigm of joint training is the sequential characteristic of the learning process, in which only a subset of input data are fed to the model at once. There are largely three different settings for lifelong learning: class, domain, and task incremental settings (De Lange et al., 2021). Here, we focus on the methods for task incremental setting, which is most relevant to the investigated framework of lifelong in-context instruction learning.

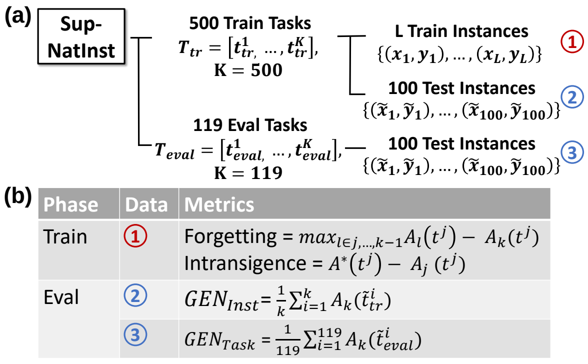
Based on how information from each task is stored and utilized later in the task stream, task incremental methods can be categorized into three: parameter regularization-, rehearsal-, and architecture expansion-based methods. Regularization-based methods (Li and Hoiem, 2017; Aljundi et al., 2018; Liu et al., 2018; Kirkpatrick et al., 2017) discourage re-visiting of inputs from previous tasks and instead introduce an auxiliary regularization term. Rehearsal-based methods (Rebuffi et al., 2017; Lopez-Paz and Ranzato, 2017; Chaudhry et al., 2018b; Shin et al., 2017) store a small number of instances and explicitly reuse the stored instances when training on future tasks. They inevitably result in some memory consumption, but this cost is offset by the clear advantage on the performance side. Lastly, architecture expansion-based methods (Mallya and Lazebnik, 2018; Serra et al., 2018) add new parameters to the backbone architecture each time a new task is presented.

Although lifelong learning is studied mostly within computer vision or robotics-related tasks, lifelong learning with NLP data has also been attracting significant interest. Many of the lifelong learning works in NLP focus on learning a sequential stream of data that belong in the same task, such as sentiment classification (Chen and Liu, 2018) or task-oriented dialog systems (Mi et al., 2020; Madotto et al., 2021). Recent research efforts aim to explore a more challenging lifelong learning setting that encompasses more than one task (Sun et al., 2019; Kanwatchara et al., 2021), only uses a limited number of instances per task (*i.e.,* few-shot setting (Qin and Joty, 2021)), or generalizes to out-of-distribution data (Lin et al., 2022).

2.2 Learning with In-Context Instructions

The idea of introducing in-context instructions was first proposed by Goldwasser and Roth (2014) who explored whether an automated agent can understand and execute the instruction that is transformed into a comprehensible expression through semantic parsing. In the NLP community, utilizing in-context language-based instructions continues to rise in popularity as an effective method of improving the generalization capability of PLMs. The core concept of in-context instruction learning is to utilize task-specific instructions that provide some description or hint to the task at hand. For instance, the NaturalInstructions-v1 (NI-v1) dataset (Mishra et al., 2022), a predecessor to Sup-NatInst used in our work, contains 64 NLP tasks, all paired with real-world instructions from Amazon Mechanical Turk (AMT) (Paolacci et al., 2010). The instruction schema of NI-v1 and Sup-NatInst resemble each other in that they include the following: task-specific definition, positive examples, negative examples, and some explanation for why an example is positive or negative.

To the best of our knowledge, this is the first work to explore whether a PLM can be trained sequentially on a stream of instruction-paired tasks. While Yin et al. (2022) seemingly consider a similar problem, their framework, ConTinTin, and ours fundamentally differ in several aspects. The key differences between ConTinTin and our framework are summarized in Table 1. Because ConTinTin includes the joint training step prior to sequentially adapting the jointly-trained model, it is not fully continual. During the adaptation step, ConTinTin observes evaluation tasks and thus ignores the trained model's generalization capacity to unseen tasks, an important aspect of instruction-based learning. Lastly, it is not as large-scale as ours, and it assumes that for all of the pre-train tasks, the


(a)
Sup-NatInst
500 Train Tasks
$T_{tr} = [t^1_{tr}, \ldots, t^K_{tr}]$, K = 500
L Train Instances
$\{(x_1, y_1), \ldots, (x_L, y_L)\}$
①
100 Test Instances
$\{(\tilde{x}_1, \tilde{y}_1), \ldots, (\tilde{x}_{100}, \tilde{y}_{100})\}$
②
119 Eval Tasks
$T_{eval} = [t^1_{eval}, \ldots, t^K_{eval}]$, K = 119
100 Test Instances
$\{(\tilde{x}_1, \tilde{y}_1), \ldots, (\tilde{x}_{100}, \tilde{y}_{100})\}$
③
(b)

Phase	Data	Metrics

Train	①	Forgetting = $max_{l \in j,\ldots,k-1} A_l(t^j) - A_k(t^j)$
		Intransigence = $A^*(t^j) - A_j(t^j)$

Eval	②	$GEN_{Inst} = \frac{1}{k}\sum_{i=1}^{k} A_k(\tilde{t}^i_{tr})$
	③	$GEN_{Task} = \frac{1}{119}\sum_{i=1}^{119} A_k(\tilde{t}^i_{eval})$

Figure 1: (a) visualizes how Sup-NatInst is restructured for our purpose. (b) summarizes the data types and metrics used for train and evaluation phases.

same number of instances are available. To tackle the ConTinTin framework, Yin et al. (2022) propose a method dubbed InstructionSpeak, which we consider as a baseline approach in Section 5.2.

3 Problem Definition

In this section, we describe Sup-NatInst and how it is restructured for the purpose of lifelong in-context instruction learning. Then, we formulate evaluation metrics for quantifying the instance- and task-level generalization capabilities. The visual summary of problem definition is provided in Figure 1.

3.1 Data

Dataset Sup-NatInst, which contains 757 train tasks and 119 evaluation tasks, is the largest and most comprehensive among existing datasets for in-context instruction learning. For the sake of computational efficiency, we randomly sample 500 out of 757 train tasks. The results reported in the original paper indicate that only minor performance change occurs after the model observes approximately 400 train tasks; therefore, it is reasonable to assume that the results obtained after 500 tasks are sufficient for analyzing the characteristics of large-scale lifelong in-context instruction learning.

The default instruction scheme of Sup-NatInst includes four components: task definition, positive examples, negative examples, and explanation. Unless specified otherwise, all explored approaches leave out negative examples because they have been shown to deteriorate the generalization performance of the target model (Wang et al., 2022). An example of instruction and instance in Sup-NatInst can be found in Section A1 of Appendix.

Data Restructuring From here on, we refer to some arbitrary target PLM model as F. We use t to

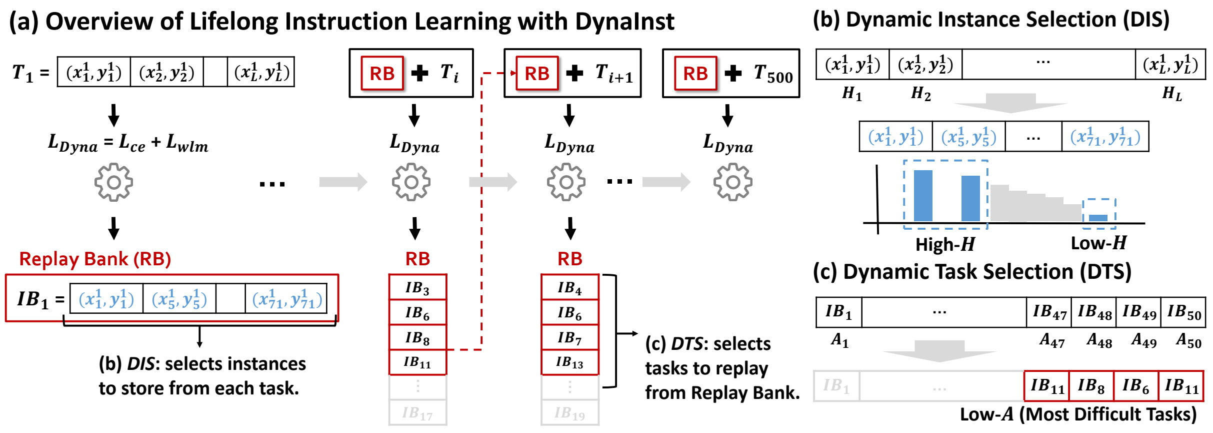

(a) Overview of Lifelong Instruction Learning with DynaInst
T_1 = (x_1^1, y_1^1) (x_2^1, y_2^1) (x_L^1, y_L^1)
$L_{Dyna} = L_{ce} + L_{wlm}$
Replay Bank (RB)
IB_1 = (x_1^1, y_1^1) (x_5^1, y_5^1) (x_{71}^1, y_{71}^1)
(b) DIS: selects instances to store from each task.
RB + T_i
RB + T_{i+1}
RB + T_{500}
L_{Dyna}
RB
IB_3 IB_6 IB_8 IB_{11} IB_{17}
IB_4 IB_6 IB_7 IB_{13} IB_{19}
(c) DTS: selects tasks to replay from Replay Bank.
(b) Dynamic Instance Selection (DIS)
H_1 H_2 H_L
High-H
Low-H
(c) Dynamic Task Selection (DTS)
IB_1 … IB_{47} IB_{48} IB_{49} IB_{50}
A_1 A_{47} A_{48} A_{49} A_{50}
IB_{11} IB_8 IB_6 IB_{11}
Low-A (Most Difficult Tasks)

Figure 2: (a) depicts the overall lifelong instruction learning process with DYNAINST that encompasses $\mathcal{L}_{\text{wlm}}$ and Dynamic Instruction Replay (DIR), which consists of Dynamic Instance Selection (DIS) and Dynamic Task Selection (DTS). (b) Based on the model's predictive entropy $\mathcal{H}$, DIS selects a mix of high and low entropy instances. (c) Based on the Rouge-L score A computed with the selected instances, DTS identifies most difficult tasks.

denote a task, which implicitly includes an instruction, and (x, y) to denote the input and output of a task-specific instance. During the lifelong learning process, the 500 train tasks with L number of labeled instances per task ($T_{\text{tr}} = [t_{\text{tr}}^i]_{i=1}^{500}$, where $t_{\text{tr}}^i = \{(x_1^i, y_1^i), ..., (x_L^i, y_L^i)\}$) are sequentially fed into F. In this work, we study two different settings for the choice of L: the static instance setting, where the same L number of instances per task are used for training, and the random instance setting, where a changing number of instances are used for each task. Because in real life, it is hard to guarantee that the same number of instances are available for each task, the random instance setting may be considered more realistic. In the random instance setting, we use a random integer value between 1 and L for each train task.

The primary objective of the lifelong learning process is to gradually improve the trained model's instance- and task-level generalization performance, as more train tasks are visited by F. To measure the instance-level generalization performance, we leave out 100 instances in each train task for evaluation process and treat them as test instances within train tasks: $\tilde{t}_{\text{tr}}^i = \{(\tilde{x}_1^i, \tilde{y}_1^i), ..., (\tilde{x}_{100}^i, \tilde{y}_{100}^i)\}$. To measure the task-level generalization performance, we utilize 100 test instances in each one of 119 evaluation tasks ($T_{\text{eval}} = [\tilde{t}_{\text{eval}}^i]_{i=1}^{119}$, where $\tilde{t}_{\text{eval}}^i = \{(\tilde{x}_1^i, \tilde{y}_1^i), ..., (\tilde{x}_{100}^i, \tilde{y}_{100}^i)\}$).

3.2 Evaluation Metrics

All metrics are measured with the Rouge-L score (Lin, 2004), which quantifies the sentence-level structural similarity; thus, a high Rouge-L score corresponds the to improved performance of a language model. The Rouge-L score is used as the default metric in the original Sup-NatInst paper as well. We denote the Rouge-L score of the j-th task after training on the k-th task as: $A_k(t^j)$.

GEN$_{\text{Inst}}$ measures the degree of instance-level generalization and is formulated as: $\frac{1}{k}\sum_{i=1}^{k} A_k(\tilde{t}_{\text{tr}}^i)$. This is equivalent to the Rouge-L score averaged across test instances of observed train tasks.

GEN$_{\text{Task}}$ measures the degree of task-level generalization and is formulated as: $\frac{1}{119}\sum_{i=1}^{119} A_k(\tilde{t}_{\text{eval}}^i)$. This is equivalent to the Rouge-L score averaged across test instances of unseen evaluation tasks.

4 Methodology

We introduce DYNAINST, our approach to lifelong in-context instruction learning. DYNAINST is a hybrid method that combines parameter regularization and experience replay. In Section 4.1, we elaborate on the use of wide local minima-inducing regularizer for lifelong instruction learning. Then, in Section 4.2, we describe how instances and tasks are dynamically stored and replayed through Dynamic Instruction Replay. The lifelong instruction learning process with DYNAINST is illustrated in Figure 2. We also provide line-by-line description of DYNAINST in Algorithm 1.

4.1 Wide Local Minima

Promoting wide local minima in neural networks has been widely accepted as an effective way of achieving improved generalization performance (Pereyra et al., 2017). In addition, in (Cha et al., 2020), it is shown that not only does wide local minima help with generalization performance, but it can also be used to combat task distribution shifts in a sequential learning process. The multi-faceted benefits of wide local minima are well-aligned with the objectives of lifelong instruction learning. Therefore, we incorporate an implementation of wide local minima-inducing regularizer proposed in Cha *et al.* (Cha et al., 2020) by modifying the plain cross entropy loss as follows:

$$\mathcal{L}_{\text{Dyna}} = \mathcal{L}_{\text{ce}} + \gamma \cdot \mathcal{L}_{\text{wlm}},$$
$$\text{where } \mathcal{L}_{\text{wlm}} = \frac{1}{L} \cdot \sum_{i=1}^{L} D_{\text{KL}}(\hat{y}_i \,||\, \text{Unif.}). \tag{1}$$

$\hat{y}_i$ is the softmax output of the model ($F(x_i)$), and γ is the coefficient used to control the strength of $\mathcal{L}_{\text{wlm}}$. D_{KL} and Unif. are the Kullback-Leiber (KL) divergence of two distributions and the uniform distribution, respectively. Essentially, $\mathcal{L}_{\text{wlm}}$ is designed to drive $\hat{y}_i$ closer to the uniform distribution. By doing so, $\mathcal{L}_{\text{wlm}}$ effectively discourages the model output from becoming overconfident. Because penalizing overconfident model outputs allows F to avoid overfitting, the resulting F trained with $\mathcal{L}_{\text{Dyna}}$ becomes more robust to distribution shifts and thus obtains higher generalization (Pereyra et al., 2017). In practice, this regularization term is implemented by maximizing the entropy of the model predictions. We additionally discuss its relationship to maximum entropy regularization used in Soft Actor-Critic from reinforcement learning in Section A2 of Appendix.

4.2 Dynamic Instruction Replay

Dynamic instruction replay (DIR) can largely be divided into two processes: Dynamic Instance Selection (DIS) and Dynamic Task Selection (DTS). To implement DIS and DTS, we introduce Replay Bank that consists of N number of task-specific Instance Banks with known task boundaries. The size of Replay Bank N is thus equal to the number of task-specific Instance Banks. Each Instance Bank of size M in Replay Bank contains M number of train instances per task. The maximum cap on the size of each memory bank is enforced to limit the memory consumption of DYNAINST.

DIS: Storing all the train instances within each task in Instance Bank leads to an excessive amount of memory consumption. Therefore, after learning each task, it is preferable to selectively store instances that will be revisited later down the task

Algorithm 1: DYNAINST

```
Require: Target model $F$, Number of Tasks
  $K$, Instance Bank of Size $M$, Replay Bank
  of Size $N$, Number of Replayed Tasks $R_t$,
  Number of Replayed Instances $R_I$,
  Number of Train Epochs per Task $E$
for $k = 1 : K$ do
    if $k == 1$ then
        Train $F$ on $t^k$ with $\mathcal{L}_{ce} + \mathcal{L}_{wlm}$ for
          $E$ epochs → WLM
    else
        Select $R_t$ most difficult tasks from
          Replay Bank → DTS
        Sample $R_I$ number of instances
          from each selected task
        Finetune $F$ on the sampled instances
          of selected tasks and $t^k$ with
          $\mathcal{L}_{\text{ce}} + \mathcal{L}_{\text{wlm}}$ for $E$ epochs → WLM
    Replay Bank.push($t^k$)
    Select $M/2$ instances with lowest $\mathcal{H}$
    Select $M/2$ instances with highest $\mathcal{H}$
    Store the selected instances in Instance
      Bank of $t^k$ → DIS
    if | Replay Bank | $> N$ then
        Replay Bank.pop($t^{k-N-1}$)
```

stream. In DYNAINST, the stored instances are 1) used to determine which tasks must be prioritized for replay and 2) replayed with future tasks. As a criterion for instance selection, we adopt the entropy of model predictions as defined below:

$$\mathcal{H}(\hat{y}_i) = -\sum_i p(\hat{y}_i|x_i;F)\log(p(\hat{y}_i|x_i;F)) \tag{2}$$

From here on, we refer to this quantity as the model's predictive entropy. Predictive entropy is commonly adopted for sample selection across various research fields (*e.g.,* active learning (Gal et al., 2017) and neural architecture search (Na et al., 2021) that can benefit by identifying a subset of instances that best represents the dataset as a whole.

After finetuning F on the k-th task, DYNAINST first measures $\mathcal{H}$ of all train instances in the k-th task. Based on $\mathcal{H}$, DYNAINST stores a mixture of high and low entropy instances in the task-specific Instance Bank. Given Instance Bank of size M, we split it into two and allocate each half to high and low entropy instances. This hybrid approach to DIS allows easier and more difficult examples to be represented evenly in Instance Bank within a fixed memory budget. After determining which

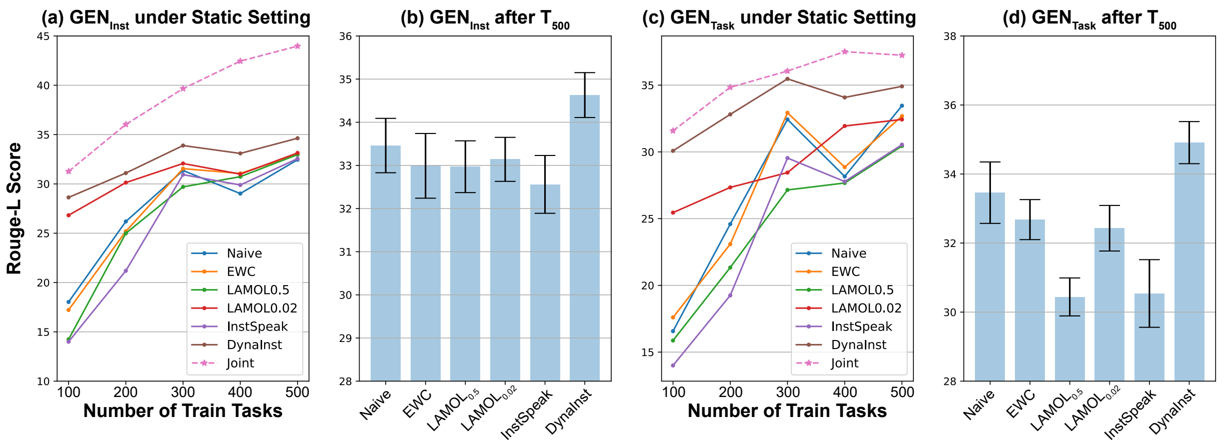


Figure 3: (a) and (c) visualize instance- and task-level generalization performances throughout the lifelong in-context instruction learning process under the **static** instance setting. (b) and (d) report means and standard deviations of final performances computed over five different random seeds under the **static** instance setting.

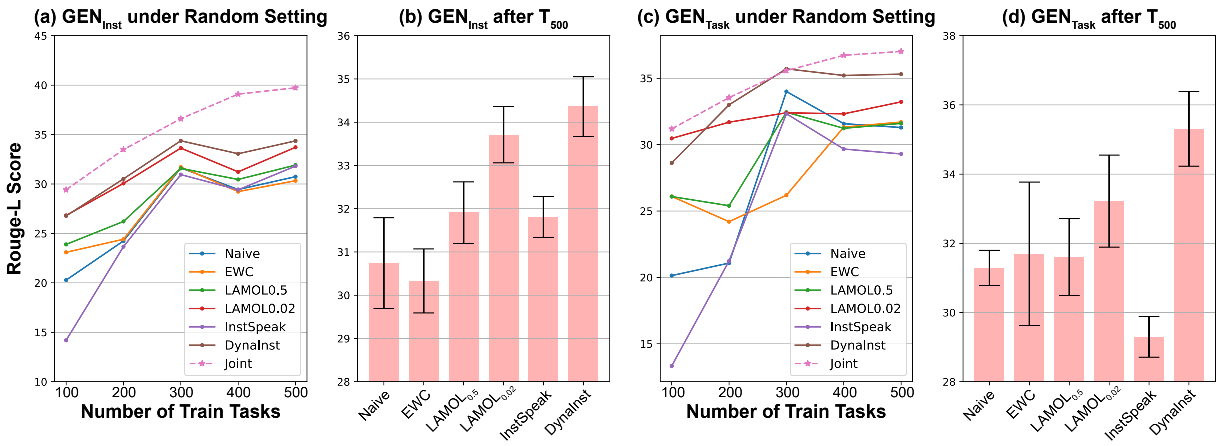


Figure 4: (a) and (c) visualize instance- and task-level generalization performances throughout the lifelong in-context instruction learning process under the **random** instance setting. (b) and (d) report means and standard deviations of final performances computed over five different random seeds under the **random** instance setting.

instances to store in Instance Bank, Instance Bank of the k-th task is added to Replay Bank. Once the number of Instance Banks exceeds the pre-set size of Replay Bank N, Instance Bank of the oldest task is removed from Replay Bank.

DTS: Instead of replaying all stored tasks in Replay Bank, we only replay the more difficult tasks that the model struggles to learn. To quantify the difficulty of a task, we rely on the instances stored in the corresponding Instance Bank. Based on the Rouge-L score of a task measured using the stored instances, R_t number of tasks with bottom Rouge-L scores are replayed with the current task. When replaying a task, we randomly sample R_i number of instances from its Instance Bank to replay. In essence, within DYNAINST, DIS and DTS complement each other to identify which tasks and instances should be replayed to maximize the generalization performance of the target model in a memory- and compute-efficient manner.

5 Experiments

5.1 Experimental Set-up

For all experiments, BART-base model (Lewis et al., 2020) is used as the target model F. All of the implementation is done through Huggingface (Wolf et al., 2019) and PyTorch (Paszke et al., 2019), and NVIDIA V100 GPU is used to run the experiments. The following hyperparameters are shared across all baselines and DYNAINST: AdamW optimizer with the learning rate of 5e-5 and constant learning rate scheduling, 2 epochs of training per task, and effective batch size of 16. As for the hyperparameters specific to DYNAINST, the Replay Bank size (N) is set to 50, and the Instance Bank size (M) is adjusted to store 50% of train instances depending on the value of L. The values of R_t and R_i are set to 10 and 2, respectively. All approaches are run using five different random seeds to create different task and instance streams.

Setting	Method	Evaluation Categories											
		TE	CEC	CR	DAR	AC	WA	OE	KT	QR	TG	DT	GEC
Static	DYNAINST	35.29	54.13	34.52	27.63	49.45	8.48	17.62	36.63	52.16	21.44	28.17	78.66
	Joint	34.98	53.39	39.32	32.14	52.57	10.74	25.96	43.29	58.49	24.29	30.95	80.56
Random	DYNAINST	38.72	53.06	33.34	30.22	50.69	8.10	15.41	37.15	50.34	19.13	27.24	73.59
	Joint	36.22	53.70	41.16	35.96	52.26	10.46	24.31	43.72	56.43	23.79	32.56	78.89

Table 2: Comparison of DYNAINST *vs.* Joint on 12 separate evaluation categories. We run each method on 5 different random seeds under the static and random instance settings and report the average Rouge-L score.

5.2 Baselines

- **Naive:** sequentially finetunes the target model on a stream of tasks with no additional technique.
- **Elastic Weight Consolidation (EWC)** (Kirkpatrick et al., 2017): is a benchmark parameter regularization method for continual learning. EWC retains past knowledge by preventing significant changes in important parameters.
- **LAMOL$_{0.5,0.02}$** (Sun et al., 2019): is a benchmark continual learning method in NLP that relies on experience replay. The sampling ratio in LAMOL $(0.5, 0.02)$ denotes the percentage of instances replayed from each one of previous tasks.
- **InstructionSpeak** (Yin et al., 2022): is a continual learning method designed for the ConTinTin framework. InstructionSpeak consists of two main processes: History Training and Negative Training. History Training replays the past two tasks with the current task, and Negative Training utilizes negative samples as if they were positive ones.

5.3 Main Results

In Figure 3, we compare GEN$_{\text{Inst}}$ and GEN$_{\text{Task}}$ of compared approaches under the static setting. Likewise, in Figure 4, we report the results of using random number of instances. The default number of instances for the static setting is set to 100, and for the random setting, the number of instances per task is randomly sampled from 1 to 100. In addition to the lifelong learning approaches, we visualize the performance obtained by jointly training on all of the tasks, equivalent to the upper bound performance for lifelong in-context instruction learning, with pink dotted lines with star markers.

In both figures, (a) and (c) show visualize how mean GEN$_{\text{Inst}}$ and GEN$_{\text{Task}}$ averaged over five different random seeds change over time. Under the static setting, DYNAINST outperforms all baselines no matter how many tasks are used for lifelong learning. Under the random setting, DYNAINST comes in as a close second to LAMOL$_{0.02}$ in the beginning when only 100 tasks are used, but soon outperforms all baselines as more tasks are added. These results clearly demonstrate that DYNAINST better utilizes the increased number of train tasks. DYNAINST appears to be particularly effective at improving the task-level generalization performance, outperforming the second-best baseline by a significant margin on the Rouge-L score under both settings.

In (b) and (d), we report means and standard deviations of GEN$_{\text{Inst}}$ and GEN$_{\text{Task}}$ after observing all 500 train tasks. Changes in random seeds seem to have a similar amount of effect on all compared methods, with no single method having a particularly small or large error bar. Under the random setting, we observe a slight increase in performance variation. We believe that this occurs because in the random setting, there are two sources of variation: changing number of instances and task ordering.

In Table 2, we report the performance of the model trained with DYNAINST and that trained with joint training on separate evaluation categories. The evaluation categories as defined by Wang et al. (2022) are in Section A1 Appendix. Due to the page constraint, from here on, we only report the results after training on all 500 tasks. It appears that the models trained with DYNAINST and joint training struggle with similar evaluation categories. Thus, it is reasonable to assume that the performance discrepancy among tasks is caused by the inherent task distribution skew within Sup-NatInst.

In Section A3 of Appendix, we report the results of using up to 20 instances per task. Once again, DYNAINST achieves the highest Rouge-L score, closest to the upper bound performance. In addition, we provide forgetting and intransigence analyses and an example of instances identified through hybrid DIS in Sections A4 and A5 of Appendix, respectively.

	Static		Random	
	GEN_{Inst}	GEN_{Task}	GEN_{Inst}	GEN_{Task}
Naive	33.66	34.08	30.08	32.39
+ WLM	33.02	34.98	32.22	33.12
+ DIR	33.24	35.48	32.17	35.31
DYNAINST	34.44	35.85	34.51	36.14

Table 3: Effect of separately applying each technical component in DYNAINST to the "Naive" baseline approach. Both the WLM regularizer and DIR process lead to a significant improvement in performance.

	Static		Random	
	GEN_{Inst}	GEN_{Task}	GEN_{Inst}	GEN_{Task}
$\gamma = 0.3$	33.85	35.55	32.02	34.86
$\gamma = 0.7$	34.03	34.92	32.89	34.01
$R_t = 15$	33.86	32.74	33.30	35.88
$R_t = 20$	34.09	35.06	32.88	35.69
$M = 10$	33.94	36.32	33.83	36.87
Default	34.44	35.85	34.51	36.14

Table 4: Sensitivity of DYNAINST to changes in various hyperparameters. Default refers to DYNAINST implemented with the default set of hyperparameters.

6 Ablation Studies

6.1 Separate Components

To validate the efficacy of each technical component in DYNAINST, we perform a component-wise analysis of DYNAINST and report the results in Table 3. It is apparent that parameter regularization with $\mathcal{L}_{\text{wlm}}$ and experience replay with DIR each contributes to improving the generalization performance of the target model trained with DYNAINST.

6.2 Hyperparameter Sensitivity

We now analyze the sensitivity of DYNAINST to the following hyperparameters: the strength of $\mathcal{L}_{\text{wlm}}$ (γ), the number of replayed tasks (R_t), and the size of the Instance Bank (M). We test out one hyperparameter at a time and fix the rest of them as default values. The results are reported in Table 4. What is particularly noteworthy is that reducing M to 10 preserves the performance of DYNAINST; this result indicates that DYNAINST is capable of achieving high generalization performance even with a limited number of stored instances.

In addition, we observe that increasing R_t does not necessarily improve the performance of DYNAINST. We conjecture that the reason behind this phenomenon is that replaying relatively easier tasks by increasing R_t may hinder the target model from learning more difficult tasks. On the contrary, joint training, which uses all train tasks at once, does not experience performance degradation as the number of train tasks increases. Note that in joint training, all instances are shuffled in a task-agnostic manner, effectively blurring the task boundaries. Therefore, we would expect the discrepancy in task difficulties to have less influence on the generalization performance of the model.

		Static		Random	
		GEN_{Inst}	GEN_{Task}	GEN_{Inst}	GEN_{Task}
	Rand	33.08	34.70	32.44	34.08
	Min	33.86	33.36	33.12	34.51
DIS	Max	33.76	34.69	34.19	35.89
	Hyb	34.44	35.85	34.51	36.14
	All	34.69	36.45	34.44	37.18
DTS	Rel	33.65	34.71	32.91	34.04
	Abs	34.44	35.85	34.51	36.14

Table 5: Effect of altering the main design choices in DIS and DTS. The default settings used in DYNAINST (hybrid DIS and DTS with the absolute Rouge-L score) achieve the best performance out of potential choices.

6.3 DIS and DTS Design Choices

Lastly, we study how different design choices for DIS and DTS influence the pefroamcne of DYNAINST. The results can be found in Table 5. For DIS, we investigate three additional entropy-based instance selection methods - random, minimum, and maximum instance selection - as well as the upper bound performance obtained by storing all of the train instances. It is clear that the hybrid DIS best approximates the upper bound performance. Such a result validates that the hybrid selection is most capable of identifying instances that are representative of the task as a whole.

The default criterion for task selection in DTS is the absolute Rouge-L score per task. One alternative approach to DTS is to utilize the relative change in the Rouge-L score, effectively replaying the tasks that are forgotten the most by the target model. The results in Table 5 show that using the relative change in the Rouge-L score leads to a meaningful degree of performance drop compared to default DTS, consolidating the effectiveness of DTS based on the absolute Rouge-L score.

7 Conclusion

In this work, a fully lifelong learning of in-context instructions was investigated for the first time. We proposed DYNAINST, a novel hybrid approach to lifelong in-context instruction learning, and verified its superiority to existing baselines under various experimental scenarios. Potential directions for future research include extending our investigation to blurred or unknown task boundaries and analyzing whether DYNAINST outputs biased predictions.

Acknowledgements

This work was supported in part by the Institute of Information & communications Technology Planning & Evaluation (IITP) and the National Research Foundation of Korea (NRF) grants funded by the Korean government (MSIT) (2022-0-00959, No. 2022R1A3B1077720, No. 2022R1A5A708390811).

Limitations and Potential Risks

The two limitations of DYNAINST are that it requires known task boundaries, and that it does not concern with corrupted or noisy training instances. In a realistic industry setting where the task definition is quite ambiguous, and a non-negligible amount of human bias and noise are introduced during the data collection process, these limitations of DYNAINST may degrade its performance. However, considering that this is the first time lifelong instruciton learning has been studied, these limitations can be considered interesting directions for future research.

Like any language model, the model trained with DYNAINST may output unfair and/or offensive predictions due to the bias embedded in the dataset. Improving the fairness of instruction-tuned language models is beyond the scope of this paper; nonetheless, if these problems remain neglected, we will risk deploying language models that are heavily biased and discriminatory.

References

Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. 2018. Memory aware synapses: Learning what (not) to forget. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 139–154.

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

Sungmin Cha, Hsiang Hsu, Taebaek Hwang, Flavio Calmon, and Taesup Moon. 2020. Cpr: Classifier-projection regularization for continual learning. In *International Conference on Learning Representations*.

Arslan Chaudhry, Puneet K Dokania, Thalaiyasingam Ajanthan, and Philip HS Torr. 2018a. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 532–547.

Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. 2018b. Efficient lifelong learning with a-gem. In *International Conference on Learning Representations*.

Zhiyuan Chen and Bing Liu. 2018. Lifelong machine learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 12(3):1–207.

Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Aleš Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. 2021. A continual learning survey: Defying forgetting in classification tasks. *IEEE transactions on pattern analysis and machine intelligence*, 44(7):3366–3385.

Yarin Gal, Riashat Islam, and Zoubin Ghahramani. 2017. Deep bayesian active learning with image data. In *International Conference on Machine Learning*, pages 1183–1192. PMLR.

Dan Goldwasser and Dan Roth. 2014. Learning from natural instructions. *Machine learning*, 94(2):205–232.

Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. 2018. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*.

Kasidis Kanwatchara, Thanapapas Horsuwan, Piyawat Lertvittayakumjorn, Boonserm Kijsirikul, and Peerapon Vateekul. 2021. Rational lamol: A rationale-based lifelong learning framework. 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 2942–2953.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880.

Zhizhong Li and Derek Hoiem. 2017. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947.

Bill Yuchen Lin, Sida I Wang, Xi Lin, Robin Jia, Lin Xiao, Xiang Ren, and Scott Yih. 2022. On continual model refinement in out-of-distribution data streams. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3128–3139.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Xialei Liu, Marc Masana, Luis Herranz, Joost Van de Weijer, Antonio M Lopez, and Andrew D Bagdanov. 2018. Rotate your networks: Better weight consolidation and less catastrophic forgetting. In *2018 24th International Conference on Pattern Recognition (ICPR)*, pages 2262–2268. IEEE.

David Lopez-Paz and Marc'Aurelio Ranzato. 2017. Gradient episodic memory for continual learning. *Advances in neural information processing systems*, 30.

Andrea Madotto, Zhaojiang Lin, Zhenpeng Zhou, Seungwhan Moon, Paul Crook, Bing Liu, Zhou Yu, Eunjoon Cho, Pascale Fung, and Zhiguang Wang. 2021. Continual learning in task-oriented dialogue systems. In *EMNLP 2021-2021 Conference on Empirical Methods in Natural Language Processing, Proceedings*.

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

Michael McCloskey and Neal J Cohen. 1989. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier.

Fei Mi, Liangwei Chen, Mengjie Zhao, Minlie Huang, and Boi Faltings. 2020. Continual learning for natural language generation in task-oriented dialog systems. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3461–3474.

Swaroop Mishra, Daniel Khashabi, Chitta Baral, and Hannaneh Hajishirzi. 2022. Cross-task generalization via natural language crowdsourcing instructions. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3470–3487.

Byunggook Na, Jisoo Mok, Hyeokjun Choe, and Sungroh Yoon. 2021. Accelerating neural architecture search via proxy data. *arXiv preprint arXiv:2106.04784*.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *arXiv preprint arXiv:2203.02155*.

Gabriele Paolacci, Jesse Chandler, and Panagiotis G Ipeirotis. 2010. Running experiments on amazon mechanical turk. *Judgment and Decision making*, 5(5):411–419.

German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. 2019. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:54–71.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32.

Gabriel Pereyra, George Tucker, Jan Chorowski, Łukasz Kaiser, and Geoffrey Hinton. 2017. Regularizing neural networks by penalizing confident output distributions. *arXiv preprint arXiv:1701.06548*.

Chengwei Qin and Shafiq Joty. 2021. Lfpt5: A unified framework for lifelong few-shot language learning based on prompt tuning of t5. In *International Conference on Learning Representations*.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. 2017. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 2001–2010.

Victor Sanh, Albert Webson, Colin Raffel, Stephen Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Arun Raja, Manan Dey, et al. 2021. Multitask prompted training enables zero-shot task generalization. In *International Conference on Learning Representations*.

Joan Serra, Didac Suris, Marius Miron, and Alexandros Karatzoglou. 2018. Overcoming catastrophic forgetting with hard attention to the task. In *International Conference on Machine Learning*, pages 4548–4557. PMLR.

Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. 2017. Continual learning with deep generative replay. *Advances in neural information processing systems*, 30.

Fan-Keng Sun, Cheng-Hao Ho, and Hung-Yi Lee. 2019. Lamol: Language modeling for lifelong language learning. In *International Conference on Learning Representations*.

Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Anjana Arunkumar, Arjun Ashok, Arut Selvan Dhanasekaran, Atharva Naik, David Stap, et al. 2022. Super-naturalinstructions:generalization via declarative instructions on 1600+ tasks. In *EMNLP*.

Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. 2021. Finetuned language models are zero-shot learners. In *International Conference on Learning Representations*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Wenpeng Yin, Jia Li, and Caiming Xiong. 2022. Contintin: Continual learning from task instructions. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3062–3072.

A Appendix

A1 Sup-NatInst Instruction Schema and Evaluation Categories

A1.1 Instruction Schema

The instructions in Sup-NatInst are contributions of 88 NLP practitioners, and the collected instructions are reviewed through Amazon Mechanical Turk (AMT) (Paolacci et al., 2010). All instructions in the Sup-NastInst dataset follow the same instruction schema that consists of: task definition, positive examples, negative examples, and explanation. The description and example of each component can be found in Figure A1.

A1.2 Evaluation Categories

All of the evaluation tasks in the Sup-NatInst dataset fall into one of the following 12 categories: Textual Entailment (TE), Cause Effect Classification (CEC), Coreference Resolution (CR), Dialogue Act Recognition (DAR), Answerability Classification (AC), Word Analogy (WA), Overlap Extraction (OE), Keyword Tagging (KT), Question Rewriting (QR), Title Generation (TG), Data to Text (DT), and Grammar Error Correction (GEC).

A2 Similarities in DYNAINST and Soft Actor-Critic

Soft Actor-Critic (Haarnoja et al., 2018) from reinforcement learning and DYNAINST both employ maximum entropy regularization. Interestingly enough, while DYNAINST and Soft Actor-Critic are used in two different domains, their motivations behind utilizing maximum entropy regularization bear some resemblance. By maximizing the expected reward and the entropy of the actor simultaneously, Soft Actor-Critic incentives the actor to improve exploration in the exploration-exploitation trade-off. The resulting model becomes more robust against estimation errors. Similarly, with the WLM regularizer, DYNAINST drives the target model towards a flatter or wider minima, effectively making the model more robust against task and data distribution shifts.

A3 Results of Using 20 Instances ($L = 20$)

The instance- and task-level generalization performance of models trained by using up to 20 instances per task are reported in Table A3 (static) and Table A4 (random). Even with fewer train instances, DYNAINST achieves the best generalization performance among compared approaches.

A4 Forgetting and Intransigence Analyses for 100 Instances ($L = 100$) and 20 Instances ($L = 100$)

In this section, we analyze the lifelong learning process through the following measures borrowed from the continual learning literature (Chaudhry et al., 2018a). For these measures, it is preferable to obtain lower numbers.

Forgetting measures the stability of the lifelong learning process by quantifying the degree of catastrophic forgetting. When $S_k(t^j)$ is the stability of j-th task after training on k-th task, $S_k(t^j)$ is defined as: $S_k(t^j) = \max_{l \in j, \ldots k-1} A_l(t^j) - A_k(t^j)$.

Intransigence measures how much knowledge from past tasks is utilized by the model when learning the current task. Otherwise known as plasticity, the intransigence measure of j-th task is defined as: $I_j = A^*(t^j) - A_j(t^j)$ where $A^*(t^j)$ denotes the Rouge-L score of F trained only on the j-th task.

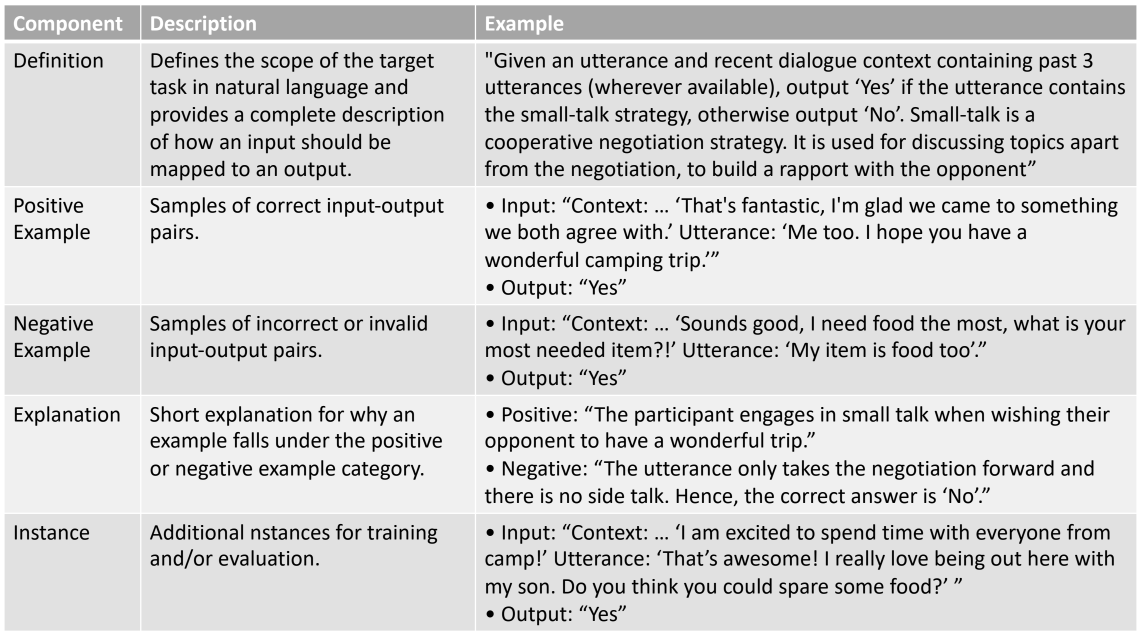
The forgetting and intransigence analyses when using $L = 100$ instances can be found in Table A1 (static) and Table A2 (random). The intransigence measure seems to have little correlation with the generalization performance of the model. For instance, EWC achieves relatively high intransigence under both settings but exhibits poor generalization performance. Such results may imply that a high level of forward transfer could be preventing the model from learning to generalize across tasks because it only promotes rapid learning of the current task, thus overfitting the target model to a single task. While the forgetting measure seems to be indicative of the model's generalization performance under the static instance setting, it again becomes less reliable under the random instance setting.

We conjecture that this conflict between the generalization metrics and continual learning metrics is a result of the difference in learning objective between conventional continual learning and lifelong in-context instruction learning. The objective of conventional continual learning is to battle catastrophic forgetting within observed train tasks, and thus forgetting and intransigence metrics focus on quantifying how much knowledge is retrained and transferred among train instances. On the contrary, lifelong in-context instruction learning wishes to improve instance- and task-level generalization performance on unobserved instances and tasks. Because the two learning paradigms and their evaluation metrics prioritize different aspects of the trained model, the two sets of metrics end up showing little correlation with each other. The forgetting and intransigence analyses under the 20-shot setting in Table A5 (static) and Table A6 (random) show similar tendencies.

A5 Analysis of Stored Instances

Here, we analyze which examples are being stored in Instance Bank and whether the difficulty of an instance quantified with the predictive entropy is aligned with human perception of difficulty. In Toxic Language Classification task, which is a binary classification task that identifies whether a tweet contains toxic language or not, the following examples were selected by DIS. Prior to discussing the examples, we apologize for their vulgarity.

- **High-entropy (difficult) instance:** Input: “Monkey see monkey doo,” | Output: “No.” It appears that the model found this example to be difficult to classify because of the following reasons. First, it does not contain explicitly toxic language. Although the word “monkey” may be used as a derogatory term against certain races, within the above input sentence, it appears innocuous. Second, it is difficult to determine the intent of the writer in the first place. Because the input, which is an idiomatic expression, reads rather incoherently, the intent of the writer remains rather ambiguous without more context.
- **Low-entropy (easy) instance:** Input: "Then the Mexican bitch wanna speak some other fuckin language & then give attitude. Can’t speak right ass! Tf" | Output: “Yes.” It is rather clear why the model found this to be an easy example. The input sentence contains explicitly negative words such as “bitch, “fucking,” and “ass," and is openly expressing the writer’s racist views.



Component	Description	Example
Definition	Defines the scope of the target task in natural language and provides a complete description of how an input should be mapped to an output.	"Given an utterance and recent dialogue context containing past 3 utterances (wherever available), output 'Yes' if the utterance contains the small-talk strategy, otherwise output 'No'. Small-talk is a cooperative negotiation strategy. It is used for discussing topics apart from the negotiation, to build a rapport with the opponent"
Positive Example	Samples of correct input-output pairs.	• Input: "Context: … 'That's fantastic, I'm glad we came to something we both agree with.' Utterance: 'Me too. I hope you have a wonderful camping trip.'" • Output: "Yes"
Negative Example	Samples of incorrect or invalid input-output pairs.	• Input: "Context: … 'Sounds good, I need food the most, what is your most needed item?!' Utterance: 'My item is food too'." • Output: "Yes"
Explanation	Short explanation for why an example falls under the positive or negative example category.	• Positive: "The participant engages in small talk when wishing their opponent to have a wonderful trip." • Negative: "The utterance only takes the negotiation forward and there is no side talk. Hence, the correct answer is 'No'."
Instance	Additional nstances for training and/or evaluation.	• Input: "Context: … 'I am excited to spend time with everyone from camp!' Utterance: 'That's awesome! I really love being out here with my son. Do you think you could spare some food?' " • Output: "Yes"

Figure A1: Description of Sup-NatInst (Wang et al., 2022) instruction schema and sample instance.

Methods	Forgetting					Intransigence				
	T_{100}	T_{200}	T_{300}	T_{400}	T_{500}	T_{100}	T_{200}	T_{300}	T_{400}	T_{500}
Naive	<u>49.24</u>	51.66	52.95	52.84	<u>52.68</u>	-3.57	-4.44	-4.32	-4.38	-4.18
EWC	49.26	<u>51.34</u>	<u>52.85</u>	<u>53.01</u>	52.92	<u>-3.58</u>	-4.26	-4.29	<u>-4.79</u>	<u>-4.98</u>
LAMOL$_{0.5}$	51.99	55.49	55.49	54.87	54.42	-2.50	-4.36	-4.11	-3.61	-3.35
LAMOL$_{0.02}$	52.43	56.26	55.99	55.24	54.81	**-4.83**	**-6.88**	**-6.45**	**-5.89**	**-5.76**
InstSpeak	51.14	53.34	53.49	53.19	53.02	-3.56	<u>-4.94</u>	<u>-4.66</u>	-4.49	-4.48
DYNAINST	**46.29**	**50.89**	**51.85**	**52.36**	**51.59**	-2.88	-3.37	-2.69	-2.18	-1.63

Table A1: Forgetting and intransigence analysis when using $L = 100$ instances per task. Unlike generalization performance, it is preferable to achieve lower numbers for both forgetting and intransigence metrics. The best and second-best metrics in each column are marked in bold and underline, respectively.

Methods	Forgetting					Intrasigence				
	T_{100}	T_{200}	T_{300}	T_{400}	T_{500}	T_{100}	T_{200}	T_{300}	T_{400}	T_{500}
Naive	<u>49.19</u>	52.63	53.39	53.49	53.56	**-8.48**	**-9.31**	**-8.89**	**-8.34**	**-8.46**
EWC	**48.17**	<u>52.34</u>	**52.98**	**52.65**	**53.16**	<u>-7.59</u>	<u>-8.64</u>	<u>-7.68</u>	<u>-6.89</u>	<u>-7.28</u>
LAMOL$_{0.5}$	50.46	54.06	54.34	53.96	53.88	-7.66	-8.16	-7.22	-6.08	-6.14
LAMOL$_{0.02}$	52.01	55.69	56.18	55.39	55.15	-7.15	-7.95	-7.52	-6.07	-6.23
InstSpeak	49.46	53.45	54.13	53.60	53.44	-5.63	-6.89	-6.08	-5.55	-5.43
DYNAINST	50.75	**52.30**	<u>53.25</u>	<u>53.27</u>	<u>53.34</u>	-3.41	-4.63	-4.02	-2.81	-3.32

Table A2: Forgetting and intransigence analysis when using $L \in [1, 100]$ instances per task. Unlike generalization performance, it is preferable to achieve lower numbers for both forgetting and intransigence metrics. The best and second-best metrics in each column are marked in bold and underline, respectively.

Methods	GEN$_{\text{Inst}}$					GEN$_{\text{Task}}$				
	T_{100}	T_{200}	T_{300}	T_{400}	T_{500}	T_{100}	T_{200}	T_{300}	T_{400}	T_{500}
Naive	23.73	28.60	31.77	30.25	33.09	25.01	30.85	37.63	34.96	36.59
EWC	24.48	27.48	31.71	30.87	32.27	25.78	29.62	36.88	34.40	35.14
LAMOL$_{0.5}$	24.52	30.42	30.39	30.66	32.05	27.20	33.85	35.02	35.43	37.56
LAMOL$_{0.02}$	27.12	29.62	32.78	30.65	33.34	28.27	31.71	34.85	32.75	32.17
InstSpeak	24.62	26.49	30.98	30.62	33.68	26.84	25.85	34.17	34.51	35.09
DYNAINST	**29.36**	**31.09**	**32.95**	**32.64**	**34.55**	**31.23**	**34.35**	**37.79**	**35.75**	**37.96**
Joint	29.93	36.29	39.62	42.50	43.41	32.28	34.44	37.49	38.93	38.25

Table A3: Comparison of zero-shot instance- and task-level generalization performance when using $L = 20$ instances per task (static instance setting). The best and second-best Rouge-L scores in each column are marked in bold and underline, respectively.

Methods	GEN$_{\text{Inst}}$					GEN$_{\text{Task}}$				
	T_{100}	T_{200}	T_{300}	T_{400}	T_{500}	T_{100}	T_{200}	T_{300}	T_{400}	T_{500}
Naive	20.07	26.07	30.58	29.68	30.56	20.11	26.26	33.60	31.73	32.12
EWC	23.74	26.48	30.78	29.61	31.04	25.17	27.76	35.74	34.25	32.07
LAMOL$_{0.5}$	**26.38**	29.05	31.62	30.51	33.43	28.63	31.53	35.83	35.12	36.09
LAMOL$_{0.02}$	26.12	29.95	31.99	32.05	31.44	30.92	33.25	35.04	34.48	36.65
InstSpeak	24.11	25.89	31.37	30.81	32.55	25.89	27.01	35.23	33.25	34.53
DYNAINST	25.28	**30.29**	**32.45**	**33.00**	**33.44**	**31.29**	**34.09**	**36.99**	**36.07**	**37.55**
Joint	27.33	31.76	34.51	34.93	35.79	30.34	34.61	37.68	37.24	38.58

Table A4: Comparison of zero-shot instance- and task-level generalization performance when using $L \in [1, 20]$ instances per task (random instance setting). The best and second-best Rouge-L scores in each column are marked in bold and underline, respectively.

Methods	Forgetting					Intransigence				
	T_{100}	T_{200}	T_{300}	T_{400}	T_{500}	T_{100}	T_{200}	T_{300}	T_{400}	T_{500}
Naive	43.88	46.89	49.99	51.02	50.89	-11.44	-10.39	**-12.20**	**-11.53**	**-11.73**
EWC	44.20	47.09	49.38	50.25	50.48	-11.73	-10.19	-11.65	-10.81	-11.35
LAMOL$_{0.5}$	47.27	49.01	50.77	50.88	51.16	-7.84	-5.64	-7.76	-6.99	-7.58
LAMOL$_{0.02}$	49.11	51.25	52.38	52.60	52.55	**-14.03**	**-10.96**	-12.17	-10.37	-10.55
InstSpeak	46.11	49.18	50.69	51.18	51.25	-8.74	-8.19	-9.49	-9.01	-9.36
DYNAINST	**42.69**	**45.36**	**46.53**	**47.07**	**47.64**	-9.71	-8.05	-9.31	-8.16	-8.56

Table A5: Forgetting and intransigence analysis when using $L = 20$ instances per task (static instance setting). Unlike generalization performance, it is preferable to achieve lower numbers for both forgetting and intransigence metrics. The best and second-best metrics in each column are marked in bold and underline, respectively.

Methods	Forgetting					Intrasigence				
	T_{100}	T_{200}	T_{300}	T_{400}	T_{500}	T_{100}	T_{200}	T_{300}	T_{400}	T_{500}
Naive	48.38	**51.45**	**52.25**	**52.14**	**52.57**	-15.12	-15.45	-15.49	-14.53	-15.65
EWC	50.97	54.61	57.11	58.25	58.72	**-17.82**	**-17.99**	**-17.83**	**-17.31**	**-18.91**
LAMOL$_{0.5}$	50.33	53.25	54.30	55.76	56.31	-10.85	-10.64	-10.42	-10.32	-11.89
LAMOL$_{0.02}$	49.22	54.57	55.94	56.01	56.56	-7.63	-8.09	-9.17	-8.70	-9.42
InstSpeak	50.69	55.99	56.56	56.90	57.25	-12.42	-14.75	-14.78	-14.50	-15.69
DYNAINST	**43.19**	53.56	55.45	55.56	56.07	-4.70	-8.22	-9.16	-8.27	-8.69

Table A6: Forgetting and intransigence analysis when using $L \in [1, 20]$ instances per task (random instance setting). Unlike generalization performance, it is preferable to achieve lower numbers for both forgetting and intransigence metrics. The best and second-best metrics in each column are marked in bold and underline, respectively.

ACL 2023 Responsible NLP Checklist

A For every submission:

☐ A1. Did you describe the limitations of your work?
Left blank.

☐ A2. Did you discuss any potential risks of your work?
Left blank.

☐ A3. Do the abstract and introduction summarize the paper's main claims?
Left blank.

☐ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☐ Did you use or create scientific artifacts?

Left blank.

☐ B1. Did you cite the creators of artifacts you used?
Left blank.

☐ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
Left blank.

☐ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
Left blank.

☐ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
Left blank.

☐ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
Left blank.

☐ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
Left blank.

C ☐ Did you run computational experiments?

Left blank.

☐ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
Left blank.

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☐ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
Left blank.

☐ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
Left blank.

☐ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
Left blank.

D ☐ Did you use human annotators (e.g., crowdworkers) or research with human participants?
Left blank.

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
Left blank.

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
Left blank.

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
Left blank.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
Left blank.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
Left blank.