Teacher-Forced Selective Self-Distillation for Uncurated Replay Data

Anonymous ACL submission

Abstract

Continual fine-tuning involves incrementally training a language model to acquire knowledge of new tasks. This learning paradigm introduces the challenge of catastrophic forgetting, where models tend to forget previously learned tasks as they adapt to new ones. Several techniques have been proposed to address this issue, including regularization, parameter-isolation, and replay-based approaches. Among these, replay-based methods have gained wider adoption due to their less invasive nature and ease of integration into existing continual learning pipelines. However, in real-world settings, replay-based methods face the practical challenge of curating ideal replay samples. This leads to the use of noisy replay data from the task owner, which is often suboptimal for improving task performance. To address this crucial real-world challenge, we introduce Teacher-Forced Selective Self-Distillation (TF-SSD) a novel method that employs self-distillation of the labels from the task stage model and refine the less effective samples using mixture of teachers framework. Our experiments involving challenging 16 task continual learning setting demonstrate that TF-SSD outperforms best-performing baseline by ∼2.7 points in task performance and ∼2.8 points in mitigating catastrophic forgetting across 2 model families: Llama2 7B and Granite3.3 2B. We are planning to open-source the code of TF-SSD.

1 Introduction

Large Language Models (LLMs) have demonstrated promising performance (Dubey et al., 2024; Brown et al., 2020) across a wide range of Natural Language Processing (NLP) tasks. In practice, LLMs must be continually updated to remain effective on newly emerging tasks. To facilitate this, Continual Learning (CL) (Chen and Liu, 2018) is employed—a paradigm in which models are incrementally trained on incoming data, enabling them

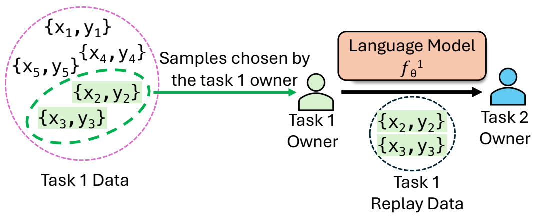

Figure 1: Demonstration of the problem setting.

to remain relevant and effective with time. Continual Fine-Tuning (CFT[1]), a variant of CL, involves incrementally training models on new fine-tuning task data. However, this paradigm introduces a major challenge known as Catastrophic Forgetting (CF) (McCloskey and Cohen, 1989), where performance on previously learned tasks deteriorates as new tasks are acquired.

Numerous techniques have been proposed to mitigate CF, including regularization-based approaches (Aljundi et al., 2018; Kirkpatrick et al., 2017; Li and Hoiem, 2018), architecture-level methods involving task-specific parameter isolation (Wang et al., 2023b; Ke et al., 2021; Satapara and Srijith, 2024; Wang et al., 2023a; Razdaibiedina et al., 2023; Wang et al., 2024a), and data replay strategies that combine previous and current task data during training (Chaudhry et al., 2019; Rebuffi et al., 2017; Buzzega et al., 2020; He et al., 2024; M'hamdi and May, 2024; Lopez-Paz and Ranzato, 2017). Among these, replay-based methods have gained broader adoption due to their ease of integration into existing CL pipelines and their less invasive nature compared to architectural or regularization-based methods.

While replay-based approaches are less invasive, they typically require storing a subset of past task data. In real-world settings, often future task owners do not have control over the replay data stored by past task owners. As shown in Figure 1, the task 1 owner just exposes a small part of the data (while keeping most of it private) to help task 2 owner to

[1]Terms CL and CFT will be used interchangeably.

preserve task 1 performance. It is then upto the task 2 owner to effectively curate the available replay data to retain task 1 performance.

In this paper, we introduce TF-SSD (see Figure 2), a novel replay-based CFT method designed to address this challenge. TF-SSD aims to curate the available replay data. Our method distills labels for each task from its corresponding stage model within the CL pipeline. These samples are subsequently refined: noisy samples selectively undergo teacher-forcing, where teacher (T) tokens are derived from the responses of an ensemble of open-source LLMs. The number of teacher tokens used is dynamically chosen based on the degree of defectiveness in the sample.

Our main contributions are:

1. We propose a novel replay-based method for continual learning that is less invasive than existing approaches, effectively addressing a real-world setting with uncurated replay data. Key features involve: (1) Label Distillation: distillation of labels from respective task stage model. (2) Refinement: teacher-forced selective self-distillation based on the degree of defectiveness in the sample with teacher tokens from ensemble of teachers.

2. We evaluate our method on the SuperNI multi-task dataset (Wang et al., 2022), comprising 48 subtasks across 16 tasks encompassing code, language and mathematics, using two model families: Granite3.3(Mishra et al., 2024) and Llama2(Touvron et al., 2023).

3. We conduct extensive analysis and ablation studies to validate the effectiveness of our approach.

2 Related Work

Consistent with recent CL studies (Wang et al., 2024a; Chen and Zeng, 2025), we classify CL strategies into three distinct categories.

Regularization. These methods (Aljundi et al., 2018; Kirkpatrick et al., 2017; Li and Hoiem, 2018) aim to mitigate forgetting of previous tasks by adding regularization terms to the loss function. However, the inclusion of multiple regularization terms may degrade model performance (Parisi et al., 2019). To address this, some works (Mok et al., 2023) combine regularization with other approaches.

Notation	Description
n	Denotes number of stages in CL pipeline. Since, each stage is associated with a task, n denotes number of tasks as well
$\{1,...,n\}$	Series of tasks in CL pipeline
f_θ^0	Base model with parameters θ
f_θ^i	Model finetuned at stage i
d^i	Training data for task i
r^i	Replay data for task i from task owner i
r_p^i	Replay data from TF-SSD for task i
$r_{\theta(i)}^i$	Replay data with labels distilled from model f_θ^i
v	Scorer module
s	A sample in the dataset
x	Input text in the sample s
y	Label in the sample s
y_l	Label with l number of tokens
m	Number of teachers
$y^1 \dots y^m$	Intermediate labels generated by m teachers
y^{MoT}	Label generated by Mixture of Teachers precisely coming from the Aggregator
y_k^{MoT}	First k tokens of y^{MoT}
y^{TF}	Label generated by teacher-forcing the stage model
t_θ^i	i^{th} teacher model

Table 1: Various notations with descriptions used in TF-SSD.

Architecture. Some approaches (Wang et al., 2023b) isolate task-specific parameters via parameter isolation, while others adopt parameter-efficient fine-tuning techniques by introducing task-specific modules (Ke et al., 2021; Satapara and Srijith, 2024; Wang et al., 2023a), learning soft prompts (Razdaibiedina et al., 2023), or applying prefix tuning (Wang et al., 2024a) for newly arriving tasks in the CL pipeline.

Replay. Early methods (Chaudhry et al., 2019; Rebuffi et al., 2017) tackle forgetting by replaying a portion of past data. Later work (Rolnick et al., 2019; Buzzega et al., 2020), improved this by generating replay data with distilled labels from previous models. He et al. (2024) introduced an architecture-level approach that distills attention from selected attention heads instead of labels. Additionally, some studies (M'hamdi and May, 2024; Lopez-Paz and Ranzato, 2017) propose techniques for selectively sampling candidates to populate a fixed-size replay data.

Replay techniques are generally less invasive than architecture-based methods. TF-SSD falls under replay-based category and we attempt to study the crucial challenge of minimizing CF in a setting where replay data from the task owner is uncurated (see Figure1) and yet essential for maintaining task performance. Since our approach belongs to the

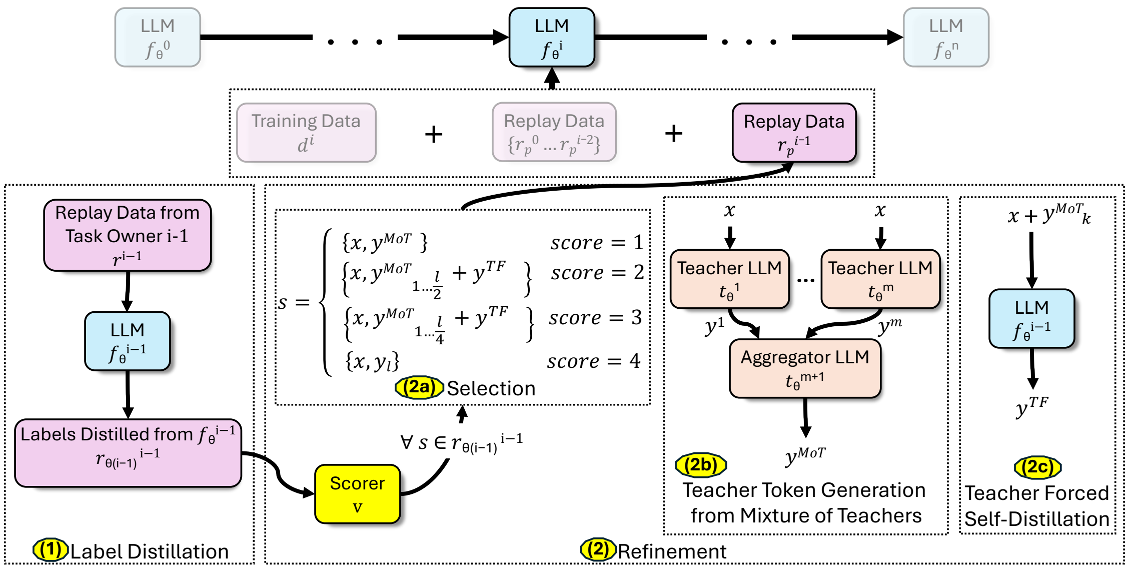
LLM f_θ^0
LLM f_θ^i
LLM f_θ^n
Training Data d^i
Replay Data $\{r_p^0 \ldots r_p^{i-2}\}$
Replay Data r_p^{i-1}
Replay Data from Task Owner i-1 r^{i-1}
LLM f_θ^{i-1}
Labels Distilled from f_θ^{i-1} $r_{\theta(i-1)}^{i-1}$
(1) Label Distillation
Scorer v
$\forall s \in r_{\theta(i-1)}^{i-1}$
$s = \{x, y^{MoT}\}$ score = 1; $\{x, y^{MoT}_{1\ldots\frac{l}{2}} + y^{TF}\}$ score = 2; $\{x, y^{MoT}_{1\ldots\frac{l}{4}} + y^{TF}\}$ score = 3; $\{x, y_l\}$ score = 4
(2a) Selection
x
Teacher LLM t_θ^1
Teacher LLM t_θ^m
y^1
y^m
Aggregator LLM t_θ^{m+1}
y^{MoT}
(2b) Teacher Token Generation from Mixture of Teachers
$x + y^{MoT}_k$
LLM f_θ^{i-1}
y^{TF}
(2c) Teacher Forced Self-Distillation
(2) Refinement

Figure 2: Overview of our method TF-SSD.

replay-based class, we include prior work - real data as replay (Chaudhry et al., 2019) and pseudo data distilled from a model as replay (Rolnick et al., 2019; Buzzega et al., 2020) that can be adapted to this setting in our evaluation. We exclude replay-based methods that require access to the model architecture (He et al., 2024) or rely on curating (M'hamdi and May, 2024; Lopez-Paz and Ranzato, 2017) the replay data from training data.

3 Method

3.1 Notations

The notations used in TF-SSD are provided in Table 1 for clarity.

3.2 Problem Formulation.

CFT aims to incrementally train the model f_θ^i over a sequence of fine-tuning tasks $\{1, \ldots, n\}$, while mitigating catastrophic forgetting and maximizing performance across all tasks. We use the widely adopted decoder class of language models with a causal language modeling loss defined as:

$$\mathcal{L}(x, y; \theta) = -\sum_{t=1}^{T} \log P_\theta(y_t \mid x, y_{<t}) \quad (1)$$

The objective is to minimize the negative log-likelihood of each label token y_t, conditioned on the input x and the previous label tokens $y_{<t}$. Where T is total number of tokens in the label y.

3.3 Overview.

TF-SSD is a replay-based method that aims to generate a refined version (r_p^i) of the uncurated replay data (r^i) provided by the task owner, aiming to mitigate catastrophic forgetting and improving overall performance across tasks. This is achieved through two key submethods: (1) Label Distillation ((1) in Figure 2), and (2) Refinement ((2) in Figure 2). The Refinement submethod further incorporates sample scoring and score-based selection (($2a$) in Figure 2), teacher forced self-distillation (($2c$) in Figure 2) using tokens sampled from a mixture of teacher (MoT) models (($2b$) in Figure 2).

3.4 Label Distillation

The training data for fine-tuning the model f_θ^i is composed of current task data d^i, replay data from past stages $\{r_p^0, \ldots, r_p^{i-2}\}$ and the immediate previous stage r_p^{i-1}. Replay data $\{r_p^0, \ldots, r_p^{i-2}\}$ is reused from past curation while r_p^{i-1} is prepared via TF-SSD for the current stage i.

This submethod is shown as (1) in Figure 2. Task owner $i-1$ provides a portion of the uncurated task data for replay. The goal of this submethod is to generate labels y for each input x using the $i-1^{th}$ stage model f_θ^{i-1}, thereby producing the distilled replay data $r_{\theta(i-1)}^{i-1}$, defined as:

$$y = f_\theta^{i-1}(x) \quad (2)$$

3.5 Refinement

Scoring. Each sample from $r_{\theta(i-1)}^{i-1}$ is passed through the Scorer v (see Section 4.3), which assigns a quality score in the range of 1-4. The score reflects the severity of defects in the sample's label y, with higher scores indicate fewer defects. The score guides the degree of teacher-forcing required to improve defective samples. We adopt 1-4 scale since it covers required levels of defectiveness buckets where samples can belong to. The Scorer follows a LLM-as-a-Scorer paradigm and details on prompt template are in Appendix A.1.

Selection. Based on the score assigned by v, the sample's label is selectively curated (($2a$) in Figure 2). If the score is 1, the full y^{MoT} is used; if 2, first half ($l/2$) of the tokens; and if 3, first quarter ($l/4$), where l is number of tokens in y^{MoT}. In the cases of score 1 and 2, remaining label tokens are from y^{TF} as described later in this section.

The intuition behind this label token selection strategy is to achieve a balanced curation of label tokens from both the MoT and the stage model f_θ^{i-1}, where the proportion of MoT tokens increases with the degree of f_θ^{i-1} deficiency, as indicated by the score. A score of 2 or 3 suggests that the label generated by the stage model has partial merit but includes minor errors; thus, MoT serves as a corrective signal through prompt-level teacher-forcing. A score of 1 indicates that f_θ^{i-1} generates flawed label, so y^{MoT} is fully retained. Conversely, a score of 4 implies that the stage model's label is optimal, and is used without modification.

Teacher Tokens Generation through MoT. As illustrated in ($2b$) of Figure 2, each input x is passed through a set of Teacher models (denoted as T) to generate corresponding outputs y, following Equation (2). These outputs are then aggregated by a separate Aggregator model (denoted as A) to produce the final label y^{MoT} for each x. Details on prompt templates used for teacher and aggregator are provided in Appendix A.1.

The paradigm of ensembling multiple open-source LLMs has been explored in latest prior work (Wang et al., 2025; Tian et al., 2025) for applications such as generating improved responses and teacher-forced chain-of-thought reasoning for smaller models. In our setup, we use MoT to selectively teacher-force label tokens by leveraging MoT tokens in the prompt (explained in the later part of the section), based on the degree of defectiveness in each sample.

Teacher-Forced Self-Distillation. This submethod is demonstrated by ($2c$) in Figure 2. Teacher-forcing is applied only to samples with scores 2 and 3. Each input x is augmented with k teacher tokens from y^{MoT}. Number of tokens k is explained in Selection submethod in Section 3.5. The concatenated input $x + y_k^{\text{MoT}}$ is then passed to the previous stage model f_θ^{i-1} to self-distill the remaining label tokens y^{TF}.

Through these submethods, the curated replay data r_p^{i-1} is constructed and used to train the current stage model f_θ^i.

4 Experimental Setup

4.1 Baselines

We evaluate our method against the following challenging baselines:

Base Model. The base LLM is evaluated without any additional fine-tuning. It is indicated by the corresponding model name in the results.

No Replay (NoRep). The base LLM is sequentially fine-tuned across all tasks in the CL pipeline, without the use of any replay data.

Replay (Rep) (Chaudhry et al., 2019). The base LLM is sequentially fine-tuned with replay data consisting of all previous task replays having labels from ground-truth.

Distilled Replay from Base Model (DR$_{\text{base}}$) (Bhushan et al., 2025). The base LLM is sequentially fine-tuned with replay data with labels distilled from the base model.

Distilled Replay from Task Stage Model (DR$_{\text{task}}$) (Rolnick et al., 2019; Buzzega et al., 2020). The base LLM is sequentially fine-tuned using replay data with labels distilled from the corresponding task stage model.

Multi-Task Learning (MTL). Multi-task learning involves training the model on all the tasks at once and CL is not involved. Many works (Huang et al., 2024; He et al., 2024) consider MTL as a target topmost performance to achieve for CL.

4.2 Ablations

We conduct the following ablation studies to isolate the contributions of MoT and teacher-forced selective self-distillation:

DR$_{\text{task}}$ + MoT (2 T + 1 A)[2]. The same setup as DR$_{\text{task}}$, with an additional refinement submethod. Each replay sample is passed through the Scorer

[2]Two teacher (T) models and one aggregator (A) model.

Task	Given a sentence, generate a new sentence by performing small changes on the sentence. Here, make sure that the changes are semantically related and syntactically similar to the input. And the generated sentence should have high commonsense plausibility, that is to have reasonable probability of it being true.		
Input	You would write a story because you have a spontaneous idea .		
Ground Truth	You would write a story because you have a detailed idea.		
	Response	**ROUGE-L**	**LaJ**
$\mathbf{DR}_{\text{task}}$	You would write a story because you have a spontaneous idea.	0.91	0
TF-SSD (2 T + 1 A)	You might decide to write a story because you have a compelling idea.	0.75	2

Table 2: An example from test set illustrating the limitation of the ROUGE-L score. The top baseline $\mathbf{DR}_{\text{task}}$ simply repeats the input text without following the task instructions, whereas TF-SSD (2 T + 1 A) adheres to the task but is penalized by ROUGE-L due to its low overlap with the ground truth. LaJ accurately fills this gap in evaluation.

scoring samples either 1 (accept) or 0 (reject). If rejected, the label is fully replaced with the MoT's output; if accepted, the original label is retained.

TF-SSD (1 T)[3]**.** As described in Section 3, the MoT component uses a single teacher for token generation. In other words, its an extension to DR_{task} + MoT (2 T + 1 A) baseline with 1-4 scale Scorer and teacher-forced selective self-distillation of labels, instead of 0-1 scoring.

TF-SSD (2 T + 1 A). This setting includes two teachers and one aggregator model. TF-SSD and TF-SSD (2 T + 1 A) are used interchangeably in the paper.

4.3 Models

Exact sources to the models used in this paper are provided in Appendix A.3.

Models in CL Pipeline. We choose the widely adopted Llama 2 7B (Touvron et al., 2023) and Granite 3.3 2B (Mishra et al., 2024) models to demonstrate our method on 2 diverse families and sizes. We choose the specific model versions to comply with their respective terms.

Teacher Models. We choose the models Mixtral 8x22B Instruct (MixIns) (Jiang et al., 2024) and WizardLM 2 8x22B(Wiz) (Xu et al., 2024) as teachers. Since teachers are employed in a form of synthetic data generation (SDG), we make sure to choose this specific set since we comply with their respective SDG terms. Details on prompt template used for teacher is provided in Appendix A.1.

Aggregator Model. Similarly, we use WizardLM 2 8x22B as the aggregator model. Aggregator prompt template can be found at Appendix A.1.

[3]Only one teacher (T) model and no aggregator model.

Scorer. We employ Prometheus 8X7B (Kim et al., 2024) as the Scorer module in our approach. Scorer prompt template can be found at Appendix A.1.

LLM-as-a-Judge. We also adopt the open-source Prometheus 8X7B model as our LLM-as-a-Judge (LaJ). This model demonstrates state-of-the-art evaluation capabilities, enabling a more rigorous assessment of our method. Using the same model as both Scorer and Judge does not introduce conflict, as the Scorer does not contribute data that directly influences model training, thereby leaving no scope for data leakage. LaJ prompt template and details on the scoring can be found at Appendix A.1.

4.4 Dataset

We use the widely adopted(Huang et al., 2024; Wang et al., 2024b) and challenging SuperNI dataset (Wang et al., 2022), which comprises 16 tasks with 3 subtasks per category.We are testing our method in 2 random orders. First order of tasks used is: Classification $\rightarrow$ Text Quality Evaluation $\rightarrow$ Code $\rightarrow$ Detection $\rightarrow$ Sentiment Analysis $\rightarrow$ Comprehension $\rightarrow$ Closed QA $\rightarrow$ Extraction $\rightarrow$ Dialogue $\rightarrow$ Program Execution $\rightarrow$ Rewriting $\rightarrow$ Open QA $\rightarrow$ Misc. $\rightarrow$ Generation $\rightarrow$ Summarization $\rightarrow$ Mathematics. Second order results and additional dataset details are provided in Appendix A.4.

4.5 Evaluation Metrics

We adopt the widely used (Huang et al., 2024) evaluation metrics for CL. We use ROUGE-L (Lin, 2004) and LaJ score (Zheng et al., 2023) to assess the model's task performance. Let $a_j^{(i)}$ denote the ROUGE-L score of stage i model evaluated on task j, similarly $s_j^{(i)}$ denotes LaJ score. We report scores in the following derived evaluation metrics to assess overall task performance and catastrophic

Task	Craft one correct answer to the question given in input. To make it more interesting, try to use non-stereotypical language if possible.Make sure your correct answer is reasonably long, consistent with the context, and requires common sense (instead of explicit extraction from the context.) In your answer, use as few words as possible from the given context.Use a response that is uncommon/non-stereotypical, so that it is less predictable. To be less repetitive, please vary your language for each question.	
Context	A friend of this blog , and someone some of you know (virtually) died recently. I only received word recently and found it hard to believe . I was going to post this several days ago , but there was a contretemps on the blog and I did n't want to get that confused with this . Now that such issues are largely over , I feel able to post .	
Question	Why is it so hard to believe that someone died ?	
Ground Truth	It was sudden.	
	Response	**LaJ**
MTL	Because they were young and healthy.	1
TF-SSD (2 T + 1 A)	It's hard to accept that someone you knew, even if it was only virtually, is no longer alive.	2

Table 3: An example from test set is provided showcasing TF-SSD(2 T + 1 A) outperforming MTL though it is considered as a reference to topmost performance.

forgetting in CL. Higher values denote better performance for the all evaluation metrics.

Average ROUGE-L (AR): This metric captures the average performance of the model across all n tasks at the end of the training sequence in CL, defined as:

$$\mathbf{AR} = \frac{1}{n}\sum_{i=1}^{n} a_i^{(n)} \tag{3}$$

Average LaJ (ALaJ): Similar to AR, ALaJ is defined as:

$$\mathbf{ALaJ} = \frac{1}{n}\sum_{i=1}^{n} s_i^{(n)} \tag{4}$$

Backward Transfer using ROUGE-L ($\mathbf{BWT_r}$): This metric measures how learning new tasks affects the performance on previously learned tasks. Average over all the stages except the last one where at each stage the difference between the final stage model $a_i^{(n)}$ and associated stage model $a_i^{(i)}$ task i performance is computed. A negative BWT_r value indicates catastrophic forgetting, while positive value indicates emergent/enhanced performance. Such enhanced performance can be attributed to the replay data used for latest tasks aiding older tasks. $\mathbf{BWT_r}$ is defined as:

$$\mathbf{BWT_r} = \frac{1}{n-1}\sum_{i=1}^{n-1}\left(a_i^{(n)} - a_i^{(i)}\right) \tag{5}$$

Backward Transfer using LaJ ($\mathbf{BWT_{LaJ}}$):

$$\mathbf{BWT_{LaJ}} = \frac{1}{n-1}\sum_{i=1}^{n-1}\left(s_i^{(n)} - s_i^{(i)}\right) \tag{6}$$

Limitation of ROUGE-L. Since the tasks in our setup are highly diverse and open-ended, traditional metrics such as ROUGE-L may not fully capture (see Table 2) the quality and correctness of the generated outputs. Moreover, we observe that some samples contain defective ground truths, which can weaken evaluation reliability. Additionally, the replay data labels obtained from TF-SSD via teacher-forced selective self-distillation often include paraphrased variants. This can lead the model to generate paraphrased yet valid responses at evaluation time, which may be penalized by ROUGE-L. To address these limitations, we include LaJ to provide a more holistic and robust evaluation.

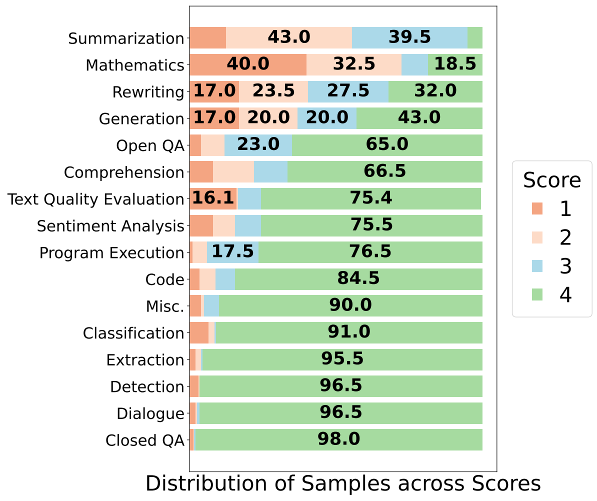

Figure 3: Distribution of the samples across Scorer model's scores for 16 tasks for Llama 2 7B. The numbers annotated are the percentage of samples belonging to the score and insignificant percentages (below 15.0%) are not shown.

4.6 Training and Inference Details

Details on the training and inference stack, hardware and hyperparameters are in Appendix A.2.

	Llama 2 7B				Granite 3.3 2B			
	AR	ALaJ	$\mathbf{BWT_r}$	$\mathbf{BWT_{LaJ}}$	AR	ALaJ	$\mathbf{BWT_r}$	$\mathbf{BWT_{LaJ}}$
Baselines								
Base Model	1.65	23.69	–	–	45.11	63.67	–	–
NoRep	21.57	61.58	-55.78	-20.88	32.33	66.64	-45.59	-15.85
Rep	**69.09**	78.40	**-4.85**	-2.51	**69.89**	78.29	**-5.38**	-3.34
$\mathbf{DR}_{\text{base}}$	**69.87**	78.44	**-4.23**	-2.44	**70.95**	78.70	**-4.55**	-3.11
$\mathbf{DR}_{\text{task}}$	**70.04**	78.69	**-3.70**	-2.13	**71.01**	79.08	**-4.32**	-2.73
Ablations								
$\mathbf{DR}_{\text{task}}$ + MoT (2 T + 1 A)	66.45	**81.27**	-7.72	**0.13**	67.66	80.94	-7.93	-0.70
TF-SSD (1 T - MixIns)	67.54	81.06	-6.58	-0.11	67.74	**81.59**	-7.77	**0.04**
TF-SSD (1 T - Wiz)	67.16	**81.46**	-6.90	**0.36**	67.33	**81.58**	-8.23	**0.00**
TF-SSD (2 T + 1 A)	66.97 (-3.07)	**81.51 (+2.82)**	-7.16 (-3.46)	**0.47 (+2.6)**	66.65 (-4.36)	**81.74 (+2.66)**	-8.86 (-4.54)	**0.29 (+3.02)**
MTL	73.65	81.16	–	–	74.45	81.20	–	–

Table 4: Evaluation scores of various baselines, ablations and MTL are shown. **First-**, **second-**, and **third-** best performing methods are highlighted over all baselines and ablations. MTL is omitted from ranking since it is considered as an ideal target performance reference for a CL model to achieve. Performance difference of TF-SSD (2T + 1A) with top method ($\mathbf{DR}_{\text{task}}$) in the baselines is shown in parentheses.

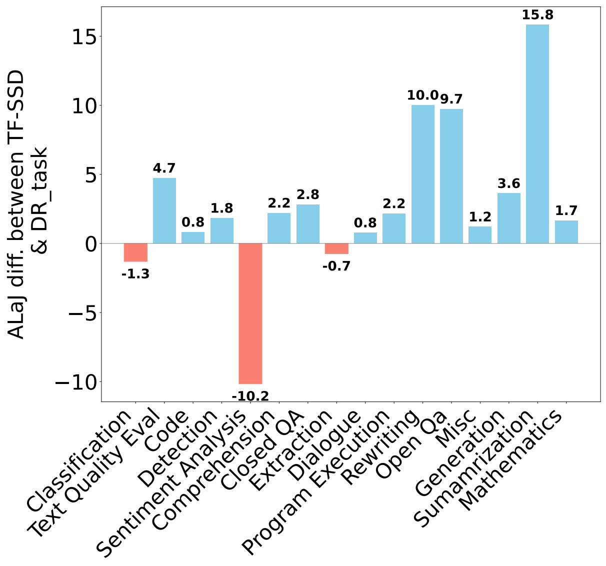

Figure 4: Task-wise difference in ALaJ performance between TF-SSD and $\mathbf{DR}_{\text{task}}$ using Llama2 7B

5 Results and Discussion

5.1 Task Performance

Performance via AR. We identify a shortcoming of the ROUGE-L score, where correct answers are sometimes penalized (see Table 2), primarily due to low overlap with the ground truth though semantically correct according to the task instruction. Since TF-SSD employs a form of teacher forcing, where a portion of the tokens is generated from a set of open models, it is more susceptible to such penalties (see example in Table 2). This explains the observed decline of ∼3.1 and ∼4.4 points in AR (Table 4) across models compared to the top-performing baseline ($\mathbf{DR}_{\text{task}}$). To address this, we adopt ALaJ, which evaluates responses more holistically by considering both the ground truth and the nature of the task. The higher MTL scores validate the fairness of LaJ.

Performance via ALaJ. Our method, TF-SSD, consistently outperforms all baselines (see Table 4), showing an improvement of ∼2.7 points in ALaJ over the best-performing baseline ($\mathbf{DR}_{\text{task}}$) across both model families. Moreover, TF-SSD meets the ideal target performance of MTL and slightly exceeds it by ∼0.45 points on average across models.

Performance compared to MTL. The reason behind the higher performance of TF-SSD over MTL (see Table 4) is attributed to the refinement submethod which involves teacher forced self-distillation using tokens coming from an ensemble of open models outside the distribution of the replay dataset. An example is provided in the Table 3 where the response of the TF-SSD is well-formed and accurate while MTL provides an incorrect response not covered in the context.

5.2 Catastrophic Forgetting.

BWT_r and BWT_{LaJ} are not applicable to the MTL and Base Model baselines since these models do not undergo CL.

CF via BWT_r. As BWT_r is a derived metric from ROUGE-L, the same limitations apply. Similarly, this explains the average drop of ∼4 points (see Table 4) of TF-SSD compared to $\mathbf{DR}_{\text{task}}$ across models.

CF via BWT_{LaJ}. BWT_{LaJ} derived from the ALaJ score, addresses this gap. TF-SSD consistently outperforms all baselines (see Table 4), with an average improvement of ∼2.81 points over $\mathbf{DR}_{\text{task}}$.

Positive BWT_{LaJ} score for TF-SSD. The last-stage model outperforms the task-stage model, suggesting it has not only retained prior knowledge but also gained enhanced capabilities. We attribute this behavior to the refinement submethod, which provides an improved replay set that was not available in training dataset of the task-stage model.

5.3 Ablation Study

Effect of self-distillation. Baselines $\mathbf{DR}_{\text{base}}$ and $\mathbf{DR}_{\text{task}}$ perform self-distillation by omitting ground-truth labels, resulting in average ALaJ improvements of ∼0.23 points for $\mathbf{DR}_{\text{base}}$ and ∼0.54 points for $\mathbf{DR}_{\text{task}}$ over the Rep baseline, which uses ground-truth labels. Distilling from the respective task-stage model leads to a 2X improvement compared to distillation from the base model. Our method, TF-SSD, which uses teacher-forced selective self-distillation, achieves an improvement of ∼3.28 points over the Rep baseline that is roughly a 6X gain over the improvement from $\mathbf{DR}_{\text{task}}$. A similar trend is observed for BWT_{LaJ} as well.

Effect of the Scorer. Overall, $\mathbf{DR}_{\text{task}}$ + MoT (2 T + 1 A) and TF-SSD demonstrate the effectiveness of the Scorer. For $\mathbf{DR}_{\text{task}}$ + MoT (2 T + 1 A), the Scorer's classification of samples into accept (use as-is) or reject (replace labels with MoT response) improves ALaJ by ∼2.5 points and ∼2.2 points in BWT_{LaJ} over $\mathbf{DR}_{\text{task}}$. Similarly, improvements from TF-SSD over $\mathbf{DR}_{\text{task}}$ are detailed in Sections 5.1 and 5.2.

We further conduct task-level analysis to understand the Scorer's impact. While overall performance improves, $\mathbf{DR}_{\text{task}}$ performs better than TF-SSD on 3 tasks—Classification, Sentiment Analysis, and Extraction (see Figure 4). The drop in performance is proportional to the number of samples with scores below 4 (see Figure 3) specifically for these 3 tasks. This suggests that while the Scorer benefits most tasks, its score distribution may negatively impact a few. Our study treats tasks as black boxes, in the sense that TF-SSD does not perform optimizations tailored to a task. However, future work could incorporate task identity and nature, enabling task-specific optimizations to further improve scoring effectiveness.

Effect of teacher forcing of the labels. The performance improvement of TF-SSD over $\mathbf{DR}_{\text{task}}$ + MoT (2 T + 1 A) highlights the positive impact of selective teacher-forcing of labels. TF-SSD outperforms $\mathbf{DR}_{\text{task}}$ + MoT (2 T + 1 A) by ∼0.52 points in ALaJ and ∼0.67 points in BWT_{LaJ}.

Effect of teachers in MoT. Ablations TF-SSD (1T - MixIns), TF-SSD (1T - Wiz) and TF-SSD (2T + 1A) demonstrate the effectiveness of the MoT paradigm. Individual teacher ablations fail to outperform 2T + 1A configuration (see Table 4) which is consistent across the model families. 2T + 1A configuration shows an average improvement of ∼0.25 points in ALaJ and ∼0.31 points in BWT_{LaJ} over single teacher configurations.

5.4 Task Order, Diversity, and Replay Data Size Robustness

Effect of task order. TF-SSD is robust to task order in CL consistently outperforming (see Table 8) $\mathbf{DR}_{\text{task}}$ for 2 distinct random task orders (orders detailed in Appendix A.7).

Effect of replay data size. Table 9 demonstrates TF-SSD is stable to different replay data sizes and consistently outperforms $\mathbf{DR}_{\text{task}}$ (details in Appendix A.8).

Effect of task diversity. TF-SSD consistently outperforms (see Table 10) best performing baseline $\mathbf{DR}_{\textbf{task}}$ in an additional CL setting having 10 tasks (details in Appendix A.6) different from the 16 task setting demonstrating robustness to task diversity.

6 Conclusion

We present TF-SSD, a novel method designed to mitigate catastrophic forgetting in continual learning, improve overall task performance, and address a critical real-world scenario of task ownership and noisy replay - not tailored for task performance. TF-SSD comprises 2 key submethods: (1) Label Distillation: where labels are distilled from the respective task-stage model, and (2) Refinement: applies teacher-forced selective self-distillation to improve label quality. These submethods curate noisy replay in to a more effective replay. We validate TF-SSD through extensive experiments, including 4 ablations and comparisons against 6 baselines across 2 model families in a 16 stage CL setting. TF-SSD consistently outperforms the baselines and multi-task learning in task performance and mitigating catastrophic forgetting. We hope our work would motivate further research catering to various real-world constraints in continual learning.

Limitations

Though TF-SSD outperforms other baselines, we identify two key limitations. First, we restrict the Mixture of Teachers component in our approach to open-source instruction-tuned models. We acknowledge that leveraging stronger proprietary model could potentially yield even greater performance improvements than those reported. Due to restrictive synthetic data generation terms of some model families both in open and closed source, we confine to a specific set of open models. However, we perform extensive ablations showcasing the contributions of the teachers in MoT. Given the observations from the ablations, choosing better teachers would further improve the performance of the TF-SSD in proprietary or licensed settings. Second, scorer plays a central role in determining the extent of teacher forcing from the MoT but from our results, some tasks are negatively affected. Thus, using a stronger proprietary or licensed models would potentially overcome the negative effect and may benefit all tasks.

Ethics Statement

We have taken deliberate steps to filter unsafe or harmful data while selecting tasks for training data from SuperNI dataset. Due to scale and diversity of the resources, the data may still contain sensitive informations.

References

Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. 2018. Memory aware synapses: Learning what (not) to forget. In *Computer Vision – ECCV 2018: 15th European Conference, Munich, Germany, September 8–14, 2018, Proceedings, Part III*, page 144–161, Berlin, Heidelberg. Springer-Verlag.

Kushagra Bhushan, Yatin Nandwani, Dinesh Khandelwal, Sonam Gupta, Gaurav Pandey, Dinesh Raghu, and Sachindra Joshi. 2025. Systematic knowledge injection into large language models via diverse augmentation for domain-specific rag.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. 2020. Dark experience for general continual learning: a strong, simple baseline. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA. Curran Associates Inc.

Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet Kumar Dokania, Philip H. S. Torr, and Marc'Aurelio Ranzato. 2019. Continual learning with tiny episodic memories. *CoRR*, abs/1902.10486.

Xi Chen and Min Zeng. 2025. Prototype conditioned generative replay for continual learning in NLP. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 12754–12770, Albuquerque, New Mexico. Association for Computational Linguistics.

Zhiyuan Chen and Bing Liu. 2018. *Continual Learning and Catastrophic Forgetting*, pages 55–75. Springer International Publishing, Cham.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurélien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Rozière, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Graeme Nail, Grégoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel M. Kloumann, Ishan Misra, Ivan Evtimov, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, and et al. 2024. The llama 3 herd of models. *CoRR*, abs/2407.21783.

Jinghan He, Haiyun Guo, Kuan Zhu, Zihan Zhao, Ming Tang, and Jinqiao Wang. 2024. SEEKR: Selective attention-guided knowledge retention for continual learning of large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 3254–3266, Miami, Florida, USA. Association for Computational Linguistics.

Jianheng Huang, Leyang Cui, Ante Wang, Chengyi Yang, Xinting Liao, Linfeng Song, Junfeng Yao, and Jinsong Su. 2024. Mitigating catastrophic forgetting in large language models with self-synthesized rehearsal. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1416–1428, Bangkok, Thailand. Association for Computational Linguistics.

Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2024. Mixtral of experts.

Zixuan Ke, Hu Xu, and Bing Liu. 2021. Adapting BERT for continual learning of a sequence of aspect sentiment classification tasks. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4746–4755, Online. Association for Computational Linguistics.

Seungone Kim, Juyoung Suk, Shayne Longpre, Bill Yuchen Lin, Jamin Shin, Sean Welleck, Graham Neubig, Moontae Lee, Kyungjae Lee, and Minjoon Seo. 2024. Prometheus 2: An open source language model specialized in evaluating other language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 4334–4353, Miami, Florida, USA. Association for Computational Linguistics.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, SOSP '23, page 611–626, New York, NY, USA. Association for Computing Machinery.

Zhizhong Li and Derek Hoiem. 2018. Learning without forgetting. *IEEE Trans. Pattern Anal. Mach. Intell.*, 40(12):2935–2947.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

David Lopez-Paz and Marc'Aurelio Ranzato. 2017. Gradient episodic memory for continual learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6470–6479, Red Hook, NY, USA. Curran Associates Inc.

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

Meryem M'hamdi and Jonathan May. 2024. Leitner-guided memory replay for cross-lingual continual learning. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 7808–7821, Mexico City, Mexico. Association for Computational Linguistics.

Mayank Mishra, Matt Stallone, Gaoyuan Zhang, Yikang Shen, Aditya Prasad, Adriana Meza Soria, Michele Merler, Parameswaran Selvam, Saptha Surendran, Shivdeep Singh, Manish Sethi, Xuan-Hong Dang, Pengyuan Li, Kun-Lung Wu, Syed Zawad, Andrew Coleman, Matthew White, Mark Lewis, Raju Pavuluri, Yan Koyfman, Boris Lublinsky, Maximilien de Bayser, Ibrahim Abdelaziz, Kinjal Basu, Mayank Agarwal, Yi Zhou, Chris Johnson, Aanchal Goyal, Hima Patel, Yousaf Shah, Petros Zerfos, Heiko Ludwig, Asim Munawar, Maxwell Crouse, Pavan Kapanipathi, Shweta Salaria, Bob Calio, Sophia Wen, Seetharami Seelam, Brian Belgodere, Carlos Fonseca, Amith Singhee, Nirmit Desai, David D. Cox, Ruchir Puri, and Rameswar Panda. 2024. Granite code models: A family of open foundation models for code intelligence.

Jisoo Mok, Jaeyoung Do, Sungjin Lee, Tara Taghavi, Seunghak Yu, and Sungroh Yoon. 2023. Large-scale lifelong learning of in-context instructions and how to tackle it. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12573–12589, Toronto, Canada. Association for Computational Linguistics.

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.

Anastasia Razdaibiedina, Yuning Mao, Rui Hou, Madian Khabsa, Mike Lewis, and Amjad Almahairi. 2023. Progressive prompts: Continual learning for language models. In *The Eleventh International Conference on Learning Representations*.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. 2017. icarl: Incremental classifier and representation learning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5533–5542.

David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. 2019. Experience replay for continual learning. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Shrey Satapara and P. K. Srijith. 2024. TL-CL: Task and language incremental continual learning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 12123–12142, Miami, Florida, USA. Association for Computational Linguistics.

Yijun Tian, Yikun Han, Xiusi Chen, Wei Wang, and Nitesh V. Chawla. 2025. Beyond answers: Transferring reasoning capabilities to smaller llms using multi-teacher knowledge distillation. In *Proceedings of the Eighteenth ACM International Conference on Web Search and Data Mining*, WSDM '25, page 251–260, New York, NY, USA. Association for Computing Machinery.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models.

Junlin Wang, Jue WANG, Ben Athiwaratkun, Ce Zhang, and James Zou. 2025. Mixture-of-agents enhances large language model capabilities. In *The Thirteenth International Conference on Learning Representations*.

Mingyang Wang, Heike Adel, Lukas Lange, Jannik Strötgen, and Hinrich Schuetze. 2024a. 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)*, pages 469–480, Mexico City, Mexico. Association for Computational Linguistics.

Xiao Wang, Tianze Chen, Qiming Ge, Han Xia, Rong Bao, Rui Zheng, Qi Zhang, Tao Gui, and Xuanjing Huang. 2023a. Orthogonal subspace learning for language model continual learning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 10658–10671, Singapore. Association for Computational Linguistics.

Yifan Wang, Yafei Liu, Chufan Shi, Haoling Li, Chen Chen, Haonan Lu, and Yujiu Yang. 2024b. InsCL: A data-efficient continual learning paradigm for fine-tuning large language models with instructions. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 663–677, Mexico City, Mexico. Association for Computational Linguistics.

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*.

Zhicheng Wang, Yufang Liu, Tao Ji, Xiaoling Wang, Yuanbin Wu, Congcong Jiang, Ye Chao, Zhencong Han, Ling Wang, Xu Shao, and Wenqiu Zeng. 2023b. Rehearsal-free continual language learning via efficient parameter isolation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10933–10946, Toronto, Canada. Association for Computational Linguistics.

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, Qingwei Lin, and Daxin Jiang. 2024. WizardLM: Empowering large pre-trained language models to follow complex instructions. In *The Twelfth International Conference on Learning Representations*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems*, 36:46595–46623.

A Appendix

A.1 Prompts

LLM-as-a-Judge. We use the following prompt template (Listing A.1) for LLM-as-a-Judge.

LLM-as-a-Judge prompt.

```
You are a helpful and fair evaluator trained to judge the quality of AI-generated answers. Your job is to assess whether the model's response is correct and appropriate for the given instruction and input.
Sometimes the provided reference answer (ground truth) might be incomplete, imprecise, or incorrect. So do **not assume** it is always correct. Instead, evaluate the model's answer based on reasoning and factual correctness.

Please follow these steps:

1. Read the instruction and input carefully.
2. Analyze the model's response: Is it logically valid, factually correct, and aligned with the instruction?
3. Compare it with the reference answer (if useful), but do not blindly trust it.
4. Rate the model's response on a scale from 0 to 2, where:
- 2 = **Correct**
- 1 = **Acceptable but with minor issues**
- 0 = **Incorrect and imperfect**
5. Give score within <score> and </score> tags only

You must explain your judgment briefly.
—
Now evaluate the following:

**Instruction**:{instruction}
**Input**:{input}
**Model's Response**:{predicted response}
**Reference Answer**:{golden response}
**Your Evaluation**:
```

Scorer prompt for TF-SSD. We use the following prompt template (Listing A.1) for Scorer model when used in TF-SSD.

Scorer prompt for TF-SSD.

```
You are given a question, the corresponding ground-truth answer, and a prediction from a model. Analyse and categorise the given prediction into one of the four labels.

There should be no contradicting statements in a good prediction. The prediction may contain extra information. If the prediction states something as a possibility, treat it as a definitive answer.
A good prediction must contain all the important information presented in the ground truth, but doesn't have to fully match it word by word.

To make your decision, first read the question and ground-truth answer carefully. Then compare the given prediction with the ground-truth answer in the light of the question.

Start with an explanation and reasoning for your evaluation within '<explanation>' and '</explanation>' tags.
Then, within '<category>' and '</category>' tags, output one of the following four labels number only:

- 4: Fully correct, complete, and fluent.
- 3: Understandable, but lacks minor details.
- 2: Understandable, but lacks major details.
- 1: Contains factual errors, hallucinations, or misleading info.

The output format should as follow: <explanation> (the explanation) </explanation> <category> (category the prediciton belongs to) </category>

Question: {full prompt} Ground-truth answer: {golden response} Prediction: {predicted response}
Output:
```

Scorer prompt for DR_{task} + MoT (2 T + 1 A) baseline. We use the following prompt template (Listing A.1) for Scorer model when used in DR_{task} + MoT (2 T + 1 A) baseline.

Scorer prompt when used in DR_{task} + MoT (2 T + 1 A) baseline.

You are given a question, the corresponding ground-truth answer and a prediction from a model. Compare the "Ground-truth answer" and the "Prediction" to determine whether the prediction correctly answers the question.
There should be no contradicting statements in a good prediction. The prediction may contain extra information. If the prediction states something as a possibility, treat it as a definitive answer.
A good prediction must contain all the important information presented in the ground truths, but doesn't have to fully match it word by word.
To make your decision, first read the question and Ground-truth answer carefully. Then compare the given Prediction with the Ground-truth answer in the light of the question. Start with an explanation and reasoning for your evaluation within <explanation> and </explanation> tags.
Then, within <score> and </score> tokens, generate "1" if the Prediction is correct in the light of Ground-truth and question. Otherwise generate "0" if it is incomplete or incorrect.

Question: {full prompt}
Ground-truth answer: {golden response}
Prediction: {predicted response}
Answer:

Teacher prompt. We use the following prompt template (Listing A.1) for the teacher.

Teacher prompt

Instruction:
{instruction}.Give only final answer within <response> and </response> tags without any explanation.

Input:
{input}
Response:

Aggregator prompt. We use the following prompt template (Listing A.1) for the aggregator.

Aggregator prompt

You have been provided with a set of responses from various open-source models along with user task instruction and input for which the response is generated. Your task is to synthesize these responses into a single, high-quality response. It is crucial to critically evaluate the information provided in these responses, recognizing that some of it may be biased or incorrect. Your response should not simply replicate the given answers but should offer a refined, accurate, and comprehensive reply to the instruction. Ensure your response is well-structured, coherent, and adheres to the highest standards of accuracy and reliability.

Response: {teacher1}
Response: {teacher2}

Task Instruction:
{instruction}.Give only final answer within <response> and </response> tags without any explanation.

Input:
{input}
Response:

A.2 Inference and Training Setup Details

Hardware. We use $8 \times A100$ $80GB$ GPUs in a single node with 32 cpus and $400Gi$ memory.

Inference software. We use vLLM[4] (Kwon et al., 2023) with tensor parallelism and greedy decoding with temperature set to 0.

Training software. We built our codebase on top of the SSR framework (Huang et al., 2024), which in turn extends the LLaMA Factory—a lightweight and efficient framework for training and fine-tuning large language models. For each task in our continual learning setup, we employed parameter-efficient fine-tuning using LoRA, implemented via the peft library. The training flow is carried using bash scripts, enabling sequential learning of tasks in a continual fashion.

[4] github.com/vllm-project/vllm

Training hyperparameters. We use a learning rate of 2e-4 and a batch size of 32 with gradient accumulation set to 1 for all experiments. LoRA is applied with a rank of 8 and a dropout rate of 0.1 to the Q and V projection matrices. The model is trained for 3 epochs in each continual learning task.

A.3 Model Details

We use several language models within the paper. All the models are used from their respective HuggingFace source: Llama 2 7B[5] (Touvron et al., 2023), Granite 3.3 2B[6] (Mishra et al., 2024), Mixtral 8x22B Instruct (MixIns)[7] (Jiang et al., 2024), WizardLM 2 8x22B(Wiz)[8] (Xu et al., 2024), Prometheus 8X7B[9] (Kim et al., 2024).

Task Name	Size	Task ID
Classification	18k	50, 1712, 65
Program Execution	17k	63, 93, 370
Mathematics	17k	85, 87, 90
Generation	16k	1, 67, 1730
Summarization	16k	589, 668, 1290
Open QA	15k	2, 24, 1731
Sentiment Analysis	15k	195, 293, 843
Rewriting	14k	402, 413, 1340
Text Quality Evaluation	12k	616, 675, 1283
Code	10k	77, 211, 869
Detection	9k	88, 209, 318
Miscellaneous	9k	305, 383, 700
Comprehension	9k	27, 1664, 223
Dialogue	7.5k	362, 766, 1500
Extraction	6.5k	39, 180, 1568
Closed QA	3k	73, 296, 667

Table 5: The subtask IDs within each task taken from the SuperNI dataset.

A.4 Dataset Details

SuperNI (Wang et al., 2022) is a collection of diverse NLP tasks with natural language instructions, released under the Apache-2.0 license. Each task is stored in a separate file, identified by a unique task ID. Table 5 summarizes the data size and subtask IDs under each task category. Table 6 reports the average and maximum input length in words. We randomly hold out 20% of the dataset for evaluation and 200 samples as buffer replay data.

[5]hf.co/meta-llama/Llama-2-7b
[6]hf.co/ibm-granite/granite-3.3-2b-base
[7]hf.co/mistralai/Mixtral-8x22B-Instruct-v0.1
[8]huggingface.co/alpindale/WizardLM-2-8x22B
[9]hf.co/prometheus-eval/prometheus-8x7b-v2.0

Task	Max Length	Avg Length
Classification	710	47.17
Text Quality Evaluation	60	17.55
Code	216	41.97
Detection	44	11.75
Sentiment Analysis	71	22.10
Comprehension	1809	117.13
Closed QA	88	42.35
Extraction	152	21.22
Dialogue	653	72.11
Program Execution	31	10.18
Rewriting	414	42.50
Open QA	1160	188.84
Misc.	149	26.46
Generation	1143	141.71
Summarization	7329	212.51
Mathematics	17	10.61

Table 6: Maximum and average length (words) per task

Task Setup. In our experiments, we consider a 16-task setup grouped under broader task categories, as shown in Table 5. Each category includes 3 subtasks, and their corresponding task IDs are listed in the same table. To illustrate the diversity of subtasks, consider the Mathematics category. Below are the natural language instructions used for its three subtasks:

Task ordering We adopt the following continual learning orders:

- **Order 1**:Classification → Text Quality Evaluation → Code → Detection → Sentiment Analysis → Comprehension → Closed QA → Extraction → Dialogue → Program Execution → Rewriting → Open QA → Misc. → Generation → Summarization → Mathematics

- **Order 2**:Generation →Mathematics →Extraction →Comprehension →Text Quality Evaluation →Dialogue →Classification →Code →Misc. →Summarization →Program Execution →Rewriting →Closed QA →Detection →Sentiment Analysis →Open QA

Subtask Instructions in Mathematics Task

1. *In this task, you will be given an arithmetic operation and you have to find its answer. The symbols of the operators '+' and '-' have been swapped, i.e., you need to perform subtraction when you see a '+' symbol and addition when you see a '-' symbol.*

2. *In this task, you will be given an arithmetic operation and you have to find its answer. The operators '+' and '-' have been replaced with new symbols. Specifically, '+' has been replaced with the symbol '@' and '-' with the symbol '#'. You need to perform the operations in the given equation and return the answer.*

3. *A polynomial equation is a sum of terms. Each term is either a constant number or consists of the variable x raised to a power and multiplied by a coefficient (called the weight). For example, in the polynomial $2x^2+3x+4$, the weights are [2, 3, 4]. A polynomial with weights [6, 4] represents the equation $6x + 4$, while [1, 3, 4] represents $1x^2 + 3x + 4$. In this task, you are given the list of weights and a value for x, and your goal is to compute the result of the polynomial expression.*

A.5 Additional Experimentation

Figures 5 and 6 show the individual LaJ scores for all 16 tasks across the baseline, ablations and MTL for Llama2 7b and Granite3.3 2B.

A.6 10 Tasks Setting

We also evaluate our method, TF-SSD, in a 10-task continual learning setting using both LLaMA2-7B and Granite 3.3B models. Unlike the 16-task setup, which includes a mix of classification and generation tasks, the 10-task configuration includes majorly generation tasks. The continual leanring order adopted: QA $\rightarrow$ QG $\rightarrow$ SA $\rightarrow$ Sum. $\rightarrow$ Trans. $\rightarrow$ DSG $\rightarrow$ Expl. $\rightarrow$ Para. $\rightarrow$ PE $\rightarrow$ POS. Table 10 shows AR, ALaJ, BWT and $\mathbf{BWT_{LaJ}}$ for both the models on baselines, ablations and multitask settings.

Task performance As shown in Table 10, our method TF-SSD consistently outperforms all baselines with an improvement of ∼8 points in ALaJ for Llama2 7B and by ∼6 points in case of Granite 3.3 2B over best-performing baseline ($\mathbf{DR_{task}}$). Moreover, TF-SSD performs better than MTL by ∼0.62 points of ALaJ for Llama2 7B and ∼5.8 points for ALaJ for Granite 3.3 2B. Tables 10 and 4 confirms that our proposed method is robust to diversity in tasks in CL.

Catastrophic forgetting Similar to 16 tasks results reported in Section 5.2, for 10 tasks our method - TF-SSD outperforms all baselines, with an improvement of ∼9.54 points of BWT_{LaJ} for Llama2 7B and ∼8.93 points of BWT_{LaJ} for Granite3.3 2B over best-performing baseline $\mathbf{DR_{task}}$.

A.7 16 Task Setting Additional Order

The two random orders experimented are mentioned in A.4 section. Table 8 shows results for order 2 is consistent with order 1 (see complete results in table 4). Thus confirming our model is robust to task ordering.

A.8 16 Task Setting Additional Replay buffer size

We additionally varied the buffer size for each task instead of keeping it constant, by randomly sampling a buffer size between 100 and 1000. The randomly selected buffer sizes for each task are provided in Table 7. The table 9 shows our method is robust to task buffer data size.

Task Name	Size
Classification	581
Program Execution	609
Mathematics	741
Generation	742
Summarization	634
Open QA	451
Sentiment Analysis	518
Rewriting	753
Text Quality Evaluation	235
Code	922
Detection	610
Miscellaneous	161
Comprehension	431
Dialogue	837
Extraction	771
Closed QA	716

Table 7: Additional buffer replay size 2 experiment buffer size per task

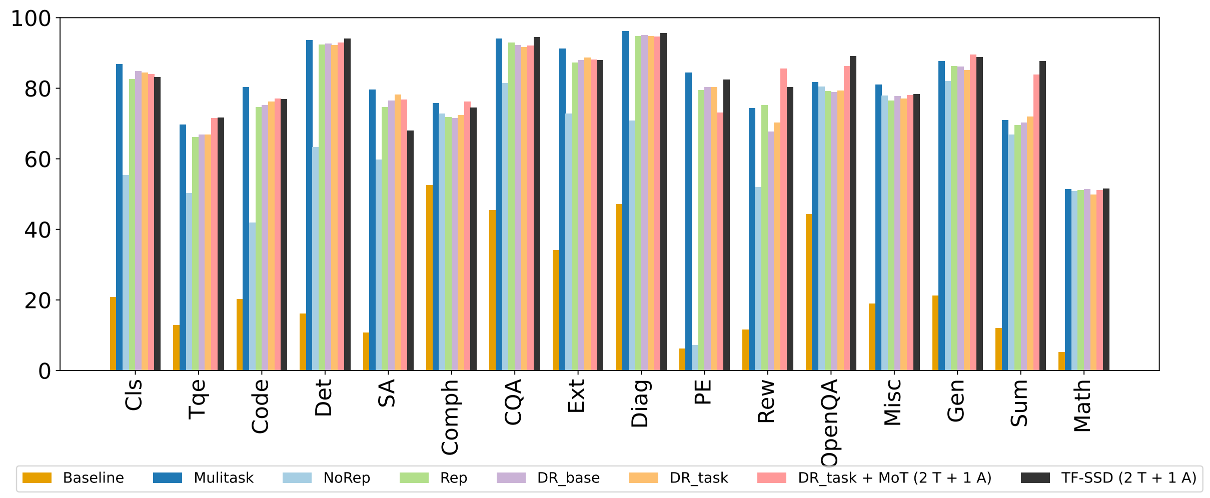

Figure 5: Individual LaJ scores for all 16 tasks for each experiment conducted with LLaMA-2 7B.

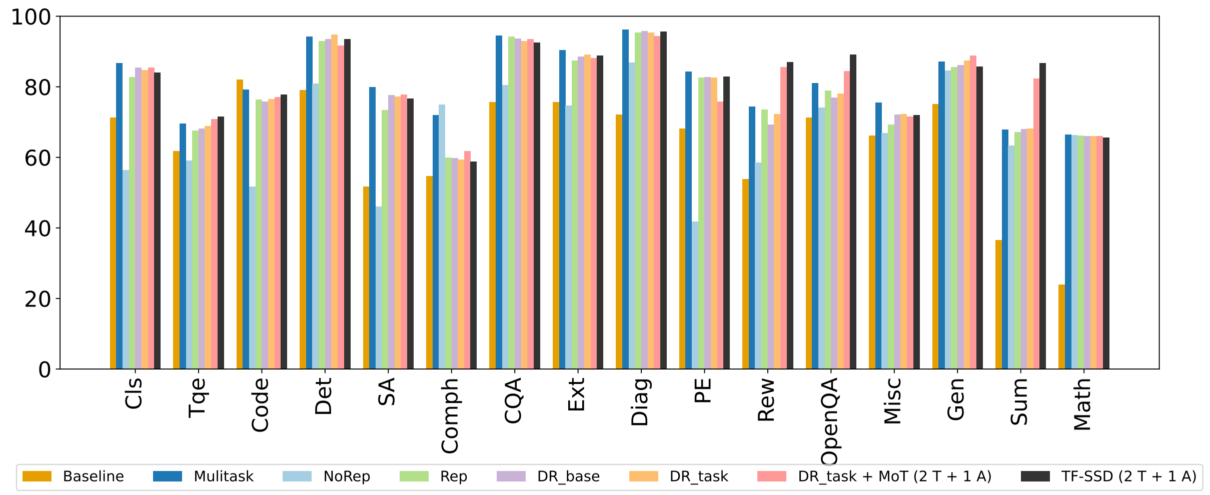

Figure 6: Individual LaJ scores for all 16 tasks for each experiment conducted with Granite 3.3 2B

	Llama 2 7B				**Granite 3.3 2B**			
	Order 1		Order 2		Order 1		Order 2	
	DR_{task}	TF-SSD(2 T + 1 A)	DR_{task}	TF-SSD(2 T + 1 A)	DR_{task}	TF-SSD(2 T + 1 A)	DR_{task}	TF-SSD(2 T + 1 A)
AR	70.03	66.97	69.59	66.23	71.01	66.64	69.36	65.80
ALaJ	78.68	**81.51**	78.63	**79.18**	79.07	**81.74**	77.79	**79.13**
BWT_r	-3.69	-7.16	-4.46	-6.89	-4.31	-8.86	-5.86	-9.55
BWT_{ALaJ}	-2.13	**0.46**	-2.82	**-1.86**	-2.73	**0.29**	-3.77	**-2.33**

Table 8: Comparative evaluation scores of our method TF-SSD on two random continual learning order with our best-performing baseline $\mathbf{DR}_{task}$. **First-** , best performing method.

	Llama 2 7B				**Granite 3.3 2B**			
	Replay size 1		Replay size 2		Replay size 1		Replay size 2	
	DR_{task}	TF-SSD(2 T + 1 A)	DR_{task}	TF-SSD(2 T + 1 A)	DR_{task}	TF-SSD(2 T + 1 A)	DR_{task}	TF-SSD(2 T + 1 A)
AR	70.03	66.97	69.26	66.11	71.01	66.64	71.15	67.52
ALaJ	78.68	**81.51**	80.14	**82.30**	79.07	**81.74**	79.80	**82.25**
BWT_r	-3.69	-7.16	-2.34	-5.55	-4.31	-8.86	-2.94	-6.86
BWT_{ALaJ}	-2.13	**0.46**	-1.19	**1.16**	-2.73	**0.29**	-1.87	**0.89**

Table 9: Comparative evaluation scores of our method TF-SSD on two buffer replay data size for each task with our best-performing baseline $\mathbf{DR}_{task}$. **First-** , best performing method.

	Llama 2 7B				Granite 3.3 2B			
	AR	ALaJ	$\mathbf{BWT_r}$	$\mathbf{BWT_{LaJ}}$	AR	ALaJ	$\mathbf{BWT_r}$	$\mathbf{BWT_{LaJ}}$
Baselines								
Base Model	5.59	32.39	–	–	44.79	60.56	–	–
NoRep	10.64	46.08	-61.70	-38.68	45.17	68.38	-23.41	-13.62
Rep	**61.19**	77.64	**-3.90**	-1.55	**61.66**	78.30	**-3.89**	-1.20
$\mathbf{DR}_{\text{base}}$	**64.74**	78.31	**-0.96**	-2.04	**65.03**	79.63	**-1.29**	-0.52
$\mathbf{DR}_{\text{task}}$	**64.84**	78.40	**-0.57**	-2.07	**65.03**	80.54	**-0.91**	-1.23
Ablations								
$\mathbf{DR}_{\text{task}}$ + MoT (2 T + 1 A)	59.96	**82.95**	-5.45	**3.14**	59.93	83.28	-6.33	3.85
TF-SSD (1 T - MixIns)	59.13	**83.09**	-6.06	**3.96**	59.49	**85.91**	-6.79	**6.23**
TF-SSD (1 T - Wiz)	58.54	**85.29**	-6.68	**5.91**	57.79	**86.38**	-8.31	**6.90**
TF-SSD (2 T + 1 A)	57.23(-7.61)	**86.48(+8.08)**	-8.64(-8.07)	**7.47(+9.54)**	57.51(-7.52)	**86.41(+5.87)**	-9.09(-8.18)	**7.7(+8.93)**
MTL	64.71	85.86	–	–	66.10	80.64	–	–

Table 10: Evaluation scores of various baselines, ablations and MTL for 10 tasks setting are shown. **First-**, **second-**, and **third-** best performing methods are highlighted over all baselines and ablations. MTL is omitted from ranking since it is considered as an ideal target performance reference for a CL model to achieve. Performance difference of TF-SSD (2T + 1A) with top method ($\mathbf{DR}_{\text{task}}$) in the baselines is shown in parentheses.