SharpSeq: Empowering Continual Event Detection through Sharpness-Aware Sequential-task Learning

Thanh-Thien Le[1]*, **Viet Dao**[2]*, **Linh Van Nguyen**[2]*, **Thi-Nhung Nguyen**[1], **Linh Ngo Van**[2]† and **Thien Huu Nguyen**[1,3]

[1]VinAI Research [2]Hanoi University of Science and Technology [3]University of Oregon

{v.thienlt3, v.nhungnt89}@vinai.io,

{vietdt200661@sis, linhnv194093@sis, linhnv@soict}@hust.edu.vn,

thien@cs.oregon.edu

Abstract

Continual event detection is a cornerstone in uncovering valuable patterns in many dynamic practical applications, where novel events emerge daily. Existing state-of-the-art approaches with replay buffers still suffer from catastrophic forgetting, partially due to overly simplistic objective aggregation. This oversight disregards complex trade-offs and leads to sub-optimal gradient updates, resulting in performance deterioration across objectives. While there are successful, widely cited multi-objective optimization frameworks for multi-task learning, they lack mechanisms to address data imbalance and evaluate whether a Pareto-optimal solution can effectively mitigate catastrophic forgetting, rendering them unsuitable for direct application to continual learning. To address these challenges, we propose **SharpSeq**, a novel continual learning paradigm leveraging sharpness-aware minimization combined with a generative model to balance training data distribution. Through extensive experiments on multiple real-world datasets, we demonstrate the superior performance of SharpSeq in continual event detection, proving the importance of our approach in mitigating catastrophic forgetting in continual event detection.

1 Introduction

Event Detection (ED) is one of the fundamental topics in NLP (Nguyen et al., 2023a; Man et al., 2022) and aims to classify an event trigger into a predefined event type in an established ontology. Continual event detection (CED), however, introduces a distinct challenge as it deals with a continuously evolving ontology that accommodates previously unseen event types as new data emerges. An effective CED system requires robust mechanisms that not only enable the identification of novel events but also mitigate the phenomenon of catastrophic forgetting, where the detection performance on previously encountered events deteriorates when new events are introduced.

Most state-of-the-art methodologies for effective continual event detection (Cao et al., 2020a; Yu et al., 2021; Liu et al., 2022) are built upon *memory-based* techniques from continual learning (Castro et al., 2018; Aljundi et al., 2018; Chaudhry et al., 2018). These techniques store a fraction of previously learned data in a *replay buffer*, allowing the model to reinforce its performance on past tasks while acquiring new knowledge. However, when employing memory-based techniques in continual learning, the management of multiple objectives associated with previous and current tasks becomes crucial. Naively aggregating these objectives by simple summation overlooks the inherent, complicated trade-offs involved. Thus, a more sophisticated strategy is imperative to address the challenge of multiple-objective optimization (MOO) for memory-based mechanisms. In this context, gradient-based frameworks for MOO, aiming to find a solution on the *Pareto front* (Sener and Koltun, 2018; Navon et al., 2022), have emerged as some of the most successful approaches. Despite their achievements, the effective application of such methods to continual NLP remains largely unexplored. Our empirical observations have shown that directly applying these frameworks for continual event detection leads to degraded performance, as evident in Table 4. This performance degradation can be attributed to *two* key factors.

The first conundrum of utilizing MOO for continual event detection is the highly imbalanced distribution of training data in later tasks, where current-task classes are significantly more prevalent. Failing to adequately address this problem can make the model susceptible to poor generalization (Johnson and Khoshgoftaar, 2019). Second, the existing MOO methods lack a clear criterion to

*Equal contribution.

†Corresponding author.

determine whether a solution on the Pareto front would be ideal for mitigating catastrophic forgetting, as well as a systematic approach to reach such a solution. In the context of continual learning, where tasks are not simultaneously presented at the beginning, an efficient solution must surpass the Pareto-optimality criteria; it should also exhibit remarkable robustness in learning new tasks and minimize performance decline in previously encountered tasks.

Regarding the above criterion, several studies (Mirzadeh et al., 2020) have proposed a relationship between solutions found at flat minima in the loss landscape and their effectiveness in alleviating catastrophic forgetting. "*Flat minima*" refers to regions where the loss function demonstrates a relatively wide and flat basin, fostering more robust models with reduced overfitting risks. Recently, a study (Phan et al., 2022a) has proposed the use of sharpness-aware minimization (SAM) (Foret et al., 2021) for finding Pareto-optimal solutions in flatter regions of the loss landscape to improve MOO. However, in the context of continual learning, the utilization of sharpness-aware minimization necessitates a nuanced approach to fully harness its potential. Specifically, it is crucial to overcome the inherent instability associated with SAM's adversarial nature and prevent it from negatively impacting the performance of the model.

Contribution. To address the above challenges, in this paper, **(i)** we propose **SharpSeq**, a novel approach that enables effective utilizations of multi-task learning frameworks with sharpness-aware minimization (SAM) for continual event detection. It is a meticulously devised adaptation of SAM, tailored for the context of sequential task emergence in continual learning. Our method entails strategically applying SAM exclusively to the objectives affiliated with the current task, while excluding its application to the objectives of previously encountered tasks. This selective integration introduces a layer of objective filter, effectively addressing the unique challenges posed by continual learning. To the best of our knowledge, our work is the first to propose a strategic adaptation of SAM and MOO for enhancing continual learning. **(ii)** To address data imbalance, we propose using a generative model to learn the underlying distribution of event triggers representations from each event type, and thereby synthesize data to alleviate the imbalance between past-task and current-task data during replay.

Our extensive experiments on multiple datasets show that our method outperforms state-of-the-art baselines by significant margins.

2 Background

2.1 Continual Learning

Continual event detection (CED) is a continual learning (i.e., lifelong learning) problem, which is often categorized into three scenarios: task-incremental learning, domain-incremental learning, and class-incremental learning (Ke and Liu, 2022; Van de Ven and Tolias, 2019). CED involves the classification of events in a continuous data stream that encompasses both new data and novel event types (Cao et al., 2020a; Liu et al., 2022). As such, it can be perceived as a class-incremental learning (CIL) problem, where the model learns to adapt and classify new events without forgetting previously learned ones. In CIL, the mention of "tasks" only refers to different stages in training, not distinct prediction tasks. Therefore, during testing, the model has to predict using the accumulated set of encountered labels, without explicit task identity.

There are three prominent approaches to alleviate catastrophic forgetting in continual learning: *regularization-based methods* (Phan et al., 2022b; Van et al., 2022; Hai et al., 2024), *architecture-based methods* (Hung et al., 2019; Liu et al., 2021c; Mallya and Lazebnik, 2018), and *replay-based methods* (Farajtabar et al., 2020; Hou et al., 2019; Nguyen et al., 2023b; Le et al., 2024). Among the trio, replay-based methods, which store a small number of previously learned data in a *replay buffer* to facilitate rehearsal learning on old tasks while the model learns new tasks, have been the most effective ones in lifelong NLP learning (de Masson d'Autume et al., 2019).

2.2 Event Detection

In conventional event detection (ED) (Wadden et al., 2019), a typical training dataset D often consists of m pairs of input-label, $D = \{(x_i, y_i)\}_1^M$. An input x includes a context $w_{1:L}$, which is a sentence of length L, and its event trigger span (s, e). The task of event detection is to classify each of these triggers into one of the defined event types y.

One common approach to ED involves utilizing a pretrained language model such as BERT (Devlin et al., 2019). This model encodes the context tokens $w_{1:L}$ to obtain the contextual representations $w'_{1:L}$. For each event trigger span (s, e), the con-

textual representations w'_s and w'_e of the beginning and ending trigger tokens are concatenated to obtain the trigger representation z_i. Subsequently, the trigger representation z_i is passed through a multilayer perceptron (MLP) to derive a feature vector h. This feature vector h is then fed into a linear layer followed by a softmax layer, resulting in a probability distribution p over the predefined event types. The entire process can be described using the following equations:

$$p = Softmax(Linear(h)), \tag{1}$$

where $h = MLP(z)$ and $z = [w'_s, w'_e]$. Let D, m denote the training dataset and the number of instances, respectively. We have the training loss as the cross-entropy loss. However, it is important to note that there is an imbalance between the data of the "unknown" (i.e., Not-Any or NA) type and other event types. To avoid the phenomenon that NA-label instances will dominate the total gradient, we regularize the loss with the introduction of hyperparameter coefficient ν:

$$\begin{aligned} L_{ed} = &- \frac{\nu}{m_{\text{NA}}} \sum_{(w,y)\in D_{\text{NA}}} \log p \\ &- \frac{1-\nu}{m_{\text{non-NA}}} \sum_{(w,y)\in D_{\text{non-NA}}} \log p. \end{aligned} \tag{2}$$

In which, D_{NA} denote the set of all NA-label instances in D; $D_{\text{non-NA}}$ denote the set of all non-NA-label instances in D; m_{NA} and $m_{\text{non-NA}}$ denote their respective sizes.

2.3 Continual Event Detection

The continual event detection problem is formulated as (Yu et al., 2021): a model is trained sequentially on T tasks, each of which has a dataset D_t corresponding to event types set C_t; and $\{C_i\}_1^T$ are disjoint. As a result, the set of all learned types at arbitrary timestep t is $O_t = C_1 \bigcup C_2 \bigcup ... \bigcup C_t$. The Not-Any (NA) label is understood as a currently undefined label; it is included in every task.

Several papers (Wu et al., 2019; Liu et al., 2022; Yu et al., 2021) have demonstrated the effectiveness of replay-based methods (section 2.1) in addressing catastrophic forgetting in both continual learning and continual event detection. In these methods, the selection of data for the replay buffer often employs the herding algorithm, as proposed by Welling (2009). In the context of CED, where the Not-Any (NA) label is present in all tasks, instances labeled as NA are not selected for inclusion in the replay buffer.

There are two techniques often employed in rehearsal when training a CED model: *knowledge distillation* and *knowledge transfer*. The descriptions of these two techniques are as follows, with R denoting the replay buffer.

Knowledge distillation is a well-known method to transfer knowledge from a model to another. We denote the models we have before and after training on the t-th task as $\boldsymbol{\theta}^{t-1}$ and $\boldsymbol{\theta}^t$, respectively. Forwarding an instance z stored in the memory buffer R through these two models, using the process described in Equation (1), we obtain the probability distributions over learned event types p^{t-1} and p^t. From there, we have the *distillation loss*:

$$L_d = -\sum_{z\in R} p^{t-1} \log(p^t). \tag{3}$$

As for **knowledge transfer** (Yu et al., 2021), it makes the prediction probability of the current model p^t close to the prediction probability of the old model scaled by temperature τ, $q \sim (p^{t-1})^{1/\tau}$. To achieve that, it uses the *knowledge transfer loss*:

$$L_{kt} = -\frac{1}{m'} \sum_{z\in D'_t} q^{t-1} \log(p^t), \tag{4}$$

where D'_t denotes the modified training set for the t-th task, and m' denotes its size. Yu et al. (2021) mentioned that the instances that have a high probability of NA label given by the old model should not be used in *knowledge transfer* since these instances have less similarity to the old event types. That is the reason we opt for constructing D'_t from D_t, instead of directly using D_t, within the scope of the *knowledge transfer loss*.

2.4 Gradient-based Multi-Objective Optimization

Multi-task learning (MTL) can be conceptualized as a multi-objective optimization problem. We denote $\boldsymbol{\theta}$ as all model parameters within the feasible set $\boldsymbol{\Theta}$, L_i as the training loss associated with task i, and K as the total number of tasks. We aim to minimize, simultaneously, all K losses:

$$\min_{\boldsymbol{\theta}} [L_1(\boldsymbol{\theta}), L_2(\boldsymbol{\theta}), ..., L_K(\boldsymbol{\theta})]. \tag{5}$$

Given $\boldsymbol{\theta}^1$ and $\boldsymbol{\theta}^2$ as two feasible solutions to problem (5), we state that $\boldsymbol{\theta}^1$ **dominates** $\boldsymbol{\theta}^2$ if and only if $L_i(\boldsymbol{\theta}^1) \leq L_i(\boldsymbol{\theta}^2)\ \forall i \in \{1, ..., K\}$ and

$\exists j \in \{1, ..., K\}\ s.t.\ L_j(\boldsymbol{\theta}^1) < L_j(\boldsymbol{\theta}^2)$. A feasible solution is deemed **Pareto-optimal** if it is not dominated by any other solutions. The set of Pareto-optimal solutions is called the **Pareto front**.

PCGrad (Yu et al., 2020), CAGrad (Liu et al., 2021a), IMTL (Liu et al., 2021b), and Nash-MTL (Navon et al., 2022) are some of the most highlighted papers that propose gradient-based MTL frameworks, aiming to find a solution on the Pareto front. Their common principal idea is to determine the vector updating direction as a linear combination of the individual task gradients, $\Delta\boldsymbol{\theta} = \sum_{i=1}^{K} \alpha_i \boldsymbol{g}_i$. α can be perceived as a dynamic version of weighted loss summation since it can change, suitably to the current state of the model, at each descending step. Their differences lie in their strategies of choosing α.

While these methods possess a strong theoretical foundation, their empirical efficiency in the context of continual learning has been subpar. In continual learning, an effective solution must fulfill more than just the Pareto-optimal requirement; it should also demonstrate robustness in learning newer, unseen tasks without experiencing a drastic performance drop in previously encountered tasks.

3 Proposed Method

To improve the common continual event detection workflow, which we describe in subsections 2.2 and 2.3, we propose the following method. In our proposed method, we use BERT, which is frozen during training, as the pretrained language model. The overview workflow of our proposed method is illustrated in Appendix A.1.

3.1 Balancing Continual Event Detection via Representation Generation

A notable challenge in continual event detection is that the replay buffer size is constrained while the dataset continuously expands throughout the model's lifespan. This scenario poses a dual conundrum: the risk of the model overfitting to the memorized data and the data imbalance when implementing multi-task replay strategy, both of which can diminish the effectiveness of rehearsal over time. To address this issue and enhance the diversity within the memory buffer, we utilize a generative model, such as Variational Autoencoder (VAE) (Kingma and Welling, 2013) or Conditional Variational Autoencoder (cVAE) (Sohn et al., 2015), to synthesize representations for each event type.

It is crucial to note that: generating high-quality natural language samples for event detection presents a more intricate challenge due to the discrete nature of our data. Therefore, we propose to leverage the frozen BERT model to instead learn the distribution of latent trigger representations to capture class-level features. This strategy is not only more feasible than generating explicit text data, where concerns about grammar, structures, and other linguistic factors arise, but it also offers significant benefits for the task of classifying these representations.

After training on task t, for each event type c in C_t, we learn a generative model (GM) to the latent representations of the data of that label and store it for subsequent sampling.

In the next task $(t+1)$, for each event type $c \in D_t$, we use its corresponding GM in the memory to sample $\tilde{n}$ synthetic trigger representations. We denote the generated set as $\tilde{R}$. The event features of instances in replay buffer R are merged with $\tilde{R}$ to get a new set called the augmented set R_a. R_a will replace R in experience replay and distillation tasks. Following that, we can write the replay loss (L_r), which is the cross-entropy loss on the augmented replay set R_a, and rewrite the distillation loss (L_d):

$$L_r = -\frac{1}{m_{R_a}} \sum_{z_a \in R_a} \log\left(p_{z_a}\right), \tag{6}$$

$$L_d = -\frac{1}{m_{R_a}} \sum_{z_a \in R_a} p_{z_a}^{t-1} \log\left(p_{z_a}^{t}\right). \tag{7}$$

In the above equations, m_{R_a} denotes the number of trigger representations in R_a, $h_{z_a} = MLP(z_a)$, and $p_{z_a} = Softmax(Linear(h_a))$.

3.2 Sharpness-Aware Sequential-Task Learning

The training process of our model can be viewed as a multi-objective optimization (MOO), aiming to minimize four objectives simultaneously: L_r, L_d, L_{ed}, and L_{kt}, as represented by equations (6), (7), (2), and (3) respectively. Multi-task learning (MTL) frameworks, such as those discussed in section 2.4, can be employed to address this problem.

Nevertheless, the empirical performance of current multi-task learning frameworks, as noted by Phan et al. (2022a), has been limited, partly due to their tendency to disregard the geometric properties of the loss landscape while solely focusing on minimizing empirical error during optimization. They propose an approach to enhance the robustness of

a multi-task model by seeking the task-based flat regions, via sharpness-aware minimization. However, we should keep in mind the unique nature of continual event detection, and continual learning problems in general, that distinguishes them from traditional multi-task learning problems. Unlike in multi-task learning, not all tasks are available simultaneously in continual event detection. Consequently, naively applying Phan et al.'s (2022a) method to this scenario is ill-advised and might results in declined performance (see Table 4).

Let us assume that after completing task t, we have obtained a solution that satisfies the flat-minima requirement. Therefore, when moving on to $t+1$, the solution already lies within the flat regions of the loss landscapes of the loss associated with task t, namely the replay loss (L_r) and distillation loss (L_d). Based on this observation, to reduce the chance of disturbing the model on old tasks with more noise from SAM, we propose to apply the sharpness-aware multi-task learning approach exclusively to the event detection loss (L_{ed}) and knowledge transfer loss (L_{kt}). Specifically, to achieve a flat minima optimizer for these two losses, we modify each objective L_i with the worst-case loss perturbation in a neighborhood of the model parameter:

$$\min_{\boldsymbol{\theta}} \max_{||\boldsymbol{\epsilon}^i||_2 \leq \rho} L_i(\boldsymbol{\theta} + \boldsymbol{\epsilon}^i), \tag{8}$$

where $||\cdot||_2$ represents the l_2 norm; ρ denotes the neighborhood radius; and L_i is either L_{ed} and L_{kt}. It is assumed that the function L_i is differentiable up to the first order with respect to the model parameter $\boldsymbol{\theta}$. The optimization problem described by equation (8) is known as sharpness-aware minimization (SAM) (Foret et al., 2021). The gradient of the modified loss function with respect to the model parameter, $\boldsymbol{g}_i$, is computed as:

$$\begin{aligned} \boldsymbol{\epsilon}^{*i} &= \rho \frac{\nabla_{\boldsymbol{\theta}} L_i(\boldsymbol{\theta})}{||\nabla_{\boldsymbol{\theta}} L_i(\boldsymbol{\theta})||_2}, \\ \boldsymbol{g}_i &= \nabla_{\boldsymbol{\theta}} L_i(\boldsymbol{\theta} + \boldsymbol{\epsilon}^{*i}). \end{aligned} \tag{9}$$

More details regarding SAM can be found in Appendix A.2.

Once we have obtained the task-specific gradients $\{\boldsymbol{g}i\}_{i=1}^K$, we can subsequently apply one of the gradient-based multi-task learning frameworks (Sener and Koltun, 2018; Yu et al., 2020; Liu et al., 2021a,b) to solve our modified MOO problem. In our approach, we specifically leverage the **Nash-MTL** framework (Navon et al., 2022) to ensure balanced improvements across all tasks. As we have discussed in section 2.4, Nash-MTL models the parameter updating direction at each descending step as a linear combination of the task-specific gradients, i.e., $\Delta\boldsymbol{\theta} = \sum_i^K \alpha_i \boldsymbol{g}_i$. According to Navon et al. (2022), the optimal direction for the gradient $\Delta\boldsymbol{\theta}^*$, which guarantees a balanced improvement across all objectives, is obtained by solving the following equation with respect to α:

$$\mathbf{G}^T\mathbf{G}\alpha = [1/\alpha_1, ..., 1/\alpha_k]^T; \tag{10}$$

$\mathbf{G}$ denotes the matrix whose columns are the task gradients $\boldsymbol{g}_i$. In summary, the essential steps of our proposed paradigm is outlined in Algorithm 1.

Algorithm 1 Sequential Sharpness Minimization for Continual Event Detection

Input: Model parameters $\boldsymbol{\theta}$, pertubation radius ρ, learning rate η and differentiable loss functions L_r, L_d, L_{ed}, and L_{kt}.
Output: Updated parameter $\boldsymbol{\theta}^*$

for each $i \in [r, d, ed, kt]$ **do**
  Compute gradient $\boldsymbol{g}_i^{\text{loss}} = \nabla_{\boldsymbol{\theta}} L_i(\boldsymbol{\theta})$
  if $i \in \{ed, kt\}$ **then**
    Worst-case perturbation direction:

$$\boldsymbol{\epsilon}^{*i} = \rho \cdot \boldsymbol{g}_i^{\text{loss}} / ||\boldsymbol{g}_i^{\text{loss}}||$$

    Approximate SAM's gradient:

$$\boldsymbol{g}_i = \nabla_{\boldsymbol{\theta}} L_i(\boldsymbol{\theta} + \boldsymbol{\epsilon}^{*i})$$

  else $\boldsymbol{g}_i \leftarrow \boldsymbol{g}_i^{\text{loss}}$
  end if
end for
Calculate α:

$$\alpha = \text{MOO\_algorithm}(\boldsymbol{g}_r, \boldsymbol{g}_d, \boldsymbol{g}_{ed}, \boldsymbol{g}_{kt})$$

Update model parameter:

$$\boldsymbol{\theta}^* = \boldsymbol{\theta} - \eta \sum_{i \in \{r,d,ed,kt\}} \alpha_i \boldsymbol{g}_i$$

4 Experiments

In this section, we present empirical results and analysis to demonstrate the effectiveness of our contributions in improving continual event detection, in comparison against several state-of-the-art baselines.

4.1 Datasets and Experimental Settings

Datasets Our methods is evaluated on two datasets: ACE 2005 (Walker et al., 2006) and MAVEN (Wang et al., 2020); both are preprocessed as in Yu et al. (2021). However, the preprocessing of dataset **ACE** is affected by random factors, leading to differences when splitting documents. Therefore, to ensure fairness, we rerun all baselines on the same preprocessed datasets. Results of the baselines as reported in their original papers are kept to compare directly with our methods. The detailed statistics of these two datasets can be found in Table 1.

Experimental Settings We use the Oracle negative setting (Yu et al., 2021) to assess all methods in continual learning scenario. In this setting, the learned types of previous tasks are excluded from the training set of the new task except for the NA type. Labels of future tasks are considered as NA type. We use the task permutations in (Yu et al., 2021) and report the average F1 on each task of 5 permutations.

Baselines Besides the aforementioned works in continual event detection – KCN (Cao et al., 2020a), KT (Yu et al., 2021), and EMP (Liu et al., 2022) – the following continual learning methods are applied to event detection tasks and used as baselines. First, the model is **finetuned** sequentially, task by task. In this case, the learned model suffers from catastrophic forgetting. **iCaRL** (Rebuffi et al., 2017) uses an exemplar set to perform classification, combined with knowledge distillation. **EEIL** (Castro et al., 2018) utilizes representative memory to retain the old knowledge; the most representative samples of new labels are chosen to train with the old data to mitigate imbalanced data. **BIC** (Wu et al., 2019) alleviates the model's bias towards new labels by using an affine transformation. **KCN** (Cao et al., 2020b) uses a limited set to store data for replay; then, knowledge distillation and prototype-enhanced retrospection are used to mitigate catastrophic forgetting. **KT** (Yu et al., 2021) follows a memory-based approach; it combines knowledge distillation with knowledge transfer. New-label samples are used to remind the model of old knowledge, and old-label samples are used to initialize representations for new-label data in the classification layer. More details are presented in section 2.3. **EMP** (Liu et al., 2022) also uses knowledge distillation but adds straight prompts into the input text to retain the old knowledge. **Nash-MTL** (Navon et al., 2022) refers to directly applying this MTL framework to simultaneously optimize the four losses mentioned in section 3.2. Finally, **Upperbound** is the case that the model is trained on all tasks at the same time. We derive the implementations of **iCaRL**, **EEIL**, **BIC**, and **KCN** from the source code published by Yu et al. (2021).

Regarding our proposed method, we experiment with the following versions: **SharpSeq**, **SharpSeq-G**, and **SharpSeq-G-A**. Nash-MTL is the default MOO algorithm for SharpSeq unless we explicitly discuss otherwise. **SharpSeq** is described in section 3.2. **SharpSeq-G** is SharpSeq without Representation Generation (RG). **SharpSeq-G-A** is a version of SharpSeq-G when we use both losses of the current task and the old tasks for sharpness-aware minimization as in (Phan et al., 2022a). The implementation details are shown in Appendix A.5.

4.2 Experimental Results

We can observe from Table 2 that SharpSeq-G-A achieved significant improvements in F1 scores across most tasks on both datasets, outperforming other baselines. Notably, compared to EMP, the final F1 score of SharpSeq-G-A increased by 4.15% in MAVEN and 4.56% in ACE. This observation demonstrates the effectiveness of finding a flat minimum in continual learning. Moreover, the consistent performance superiority of SharpSeq-G over SharpSeq-G-A highlights the efficiency and necessity of our sharpness-aware continual learning paradigm. Furthermore, Representation Generation (RG) enhanced the F1 scores of SharpSeq-G from 59.11% to 60.27% at the fifth task of MAVEN, and from 56.85% to 62.60% at the fifth task of ACE. These findings are concrete evidence of the effectiveness of our methods. RG synthesizes old-label data to improve balance in the training set, benefiting multi-objective optimization algorithms. Our optimization framework, specifically tailored for continual learning, achieves a minimizer at flat regions while effectively alleviating noise due to Sharpness-Aware Minimization's (SAM) adversarial nature. These are the foundations that enable our methods to outperform current state-of-the-art approaches in continual event detection.

4.3 Ablation Study

In this section, we explore the impact of the generative model and multi-objective optimization

	MAVEN				ACE			
	#Doc	#Sentence	#Mention	#Negative	#Doc	#Sentence	#Mention	#Negative
Train	2522	27983	67637	280151	501	18246	4088	261027
Dev	414	4432	10880	46318	41	1846	433	53620
Test	710	8038	18904	79699	55	689	790	93159

Table 1: Statistics of two datasets. #Doc stands for the total number of documents.

	MAVEN					ACE				
Task	1	2	3	4	5	1	2	3	4	5
finetune	63.16	40.30	33.00	23.86	22.34	55.88	40.25	37.04	20.71	25.01
iCaRL	18.08	27.03	30.78	31.26	29.77	5.05	6.42	7.05	6.93	9.44
EEIL	63.16	48.17	44.17	40.35	37.75	55.88	48.63	57.14	50.45	52.68
BIC	63.16	55.51	53.96	50.13	49.07	55.88	58.16	61.23	59.72	59.02
KCN	63.16	55.73	53.69	48.86	47.44	55.88	58.55	61.40	59.48	58.64
KT	62.76	58.49	57.46	55.38	54.87	55.88	57.29	61.42	60.78	59.82
EMP	**66.82**	58.02	58.19	55.07	54.52	**59.05**	57.14	55.80	53.42	52.97
Nash-MTL	62.76	60.39	59.47	56.09	52.84	55.88	57.92	62.08	59.11	58.17
SharpSeq-G-A	62.28	61.52	62.55	60.52	58.67	56.47	**59.08**	63.46	59.23	57.53
SharpSeq-G	62.28	61.57	62.48	60.54	59.11	56.47	58.51	63.37	59.54	56.85
SharpSeq	62.28	**61.85**	**62.92**	**61.31**	**60.27**	56.47	56.99	**64.44**	**62.47**	**62.60**
Upperbound	/	/	/	/	*63.46*	/	/	/	/	*67.95*

Table 2: Classification F1-scores (%) on 2 datasets MAVEN and ACE.

	MAVEN					ACE				
Task	1	2	3	4	5	1	2	3	4	5
KT	62.76	58.49	57.46	55.38	54.87	55.88	57.29	61.42	60.78	59.82
GMMs	**62.76**	59.30	59.55	58.25	57.70	55.88	56.97	62.48	60.88	62.01
GMMs+SS	62.28	**61.85**	**62.92**	**61.31**	**60.27**	56.47	56.99	**64.44**	**62.47**	**62.60**
VAE	62.76	60.06	59.82	57.14	55.68	55.88	57.37	61.52	59.38	59.51
VAE+SS	62.28	61.63	62.63	60.62	59.30	**56.47**	**59.61**	63.16	60.3	61.56
CVAE	62.76	59.62	59.60	56.56	54.88	55.88	57.97	64.24	60.46	61.81
CVAE+SS	62.28	61.68	62.38	60.33	58.77	56.47	57.47	63.77	60.60	61.87

Table 3: Ablation results of generation methods. "SS" is the abbreviation of SharpSeq.

	MAVEN					ACE				
Task	1	2	3	4	5	1	2	3	4	5
KT	62.76	58.49	57.46	55.38	54.87	55.88	57.29	61.42	60.78	59.82
Nash-MTL	**63.69**	60.56	59.12	55.56	52.40	55.19	58.12	62.77	58.46	55.69
Nash-MTL + SS	62.28	**61.85**	**62.92**	**61.31**	**60.27**	**56.47**	56.99	**64.44**	**62.47**	**62.60**
PCGrad	63.69	58.91	54.71	48.70	45.58	55.19	57.99	62.47	58.40	55.56
PCGrad + SS	62.28	61.45	61.52	57.78	55.48	56.47	**58.36**	63.62	61.53	61.58
IMTL	63.69	53.02	51.92	50.33	48.06	55.19	56.96	58.62	59.72	56.23
IMTL + SS	62.28	58.43	58.47	56.40	55.93	56.47	54.76	59.20	56.70	56.98

Table 4: Ablation result on MOO methods. "SS" is the abbreviation of SharpSeq.

method choices. Additional ablation studies on the number of GMM components and the ratio of synthesized representations are provided in Appendices A.4 and A.3.

Generative model We examine three generative methods to synthesize samples for old labels: GMMs, VAE (Kingma and Welling, 2013), and cVAE (Sohn et al., 2015); the results are presented in Table 3. We can see that all generation methods result in better performance compared to the baseline model KT. Considering the generation method in isolation, GMMs achieved the best performance, substantially improving KT by 3.83% and 3.19% on MAVEN and ACE, respectively, after the fifth task. This outcome proves the effectiveness of employing generative models to relieve the imbalance problem. Especially, when combined with SharpSeq, all of them gained significant additional enhancements across most tasks. On MAVEN, GMMs+SS was the best method with the F1-Score of 60.27% after the fifth task, better than standalone GMMs by 1.57%. GMMs+SS were also the best method on ACE with a score of 62.60%. From these findings, we can see that the choice of the generation method for SharpSeq is consequential and needs to be selected carefully: GMMs' learning process can be perceived as a soft-clustering process, which makes them excel in preserving the inherent separability within the latent trigger representations of different labels. Conversely, VAEs are trained such that their encoders can map the data into a continuous, latent probabilistic space, which allows smooth interpolations during reconstruction. This is a strength of VAEs in generating continuous-in-nature data types such as images and sound. However, in the context of our work, additional mapping of the latent trigger representations to a different latent space can result in unnecessary information loss. As such, the expected benefit of VAEs, which is to achieve smooth interpolation between two latent spaces, is not significant for the replay process of our Continual Event Detection model. Moreover, GMMs offer advantages in training and storage efficiency compared to VAE or cVAE. While VAE requires the creation and training of a new network for every event type, and cVAE requires that for every task, GMMs eliminate the need for excessive network proliferation, resulting in shorter training and inference times.

Multi-objective optimization algorithm The results in table 4 show the performances of different MOO methods, with and without SharpSeq. When we used MOO methods in isolation, Nash-MTL achieved the best performance at most tasks on both datasets. Although PCGrad and Nash-MTL's results on the ACE datasets were comparable, Nash-MTL outperformed PCGrad by a significant margin of 6.82% after five tasks on MAVEN.

However, directly applying MOO methods to KT without any adjustments can even cause a downgrade in performance. For instance, Nash-MTL worsened KT's performance by 2.47% and 4.13% on MAVEN and ACE, respectively. The main reasons for these decreases are the oversight of training data imbalance and the inherent differences between multi-task learning and continual learning. When the MOO methods were combined SharpSeq, their performances improved clearly. Considering these methods when combined with SharpSeq (SS), Nash-MTL still had the best F1 score at most tasks, on both datasets. Particularly, on MAVEN, Nash-MTL+SS achieved 60.27% F1 score, better than Nash-MTL (52.4%), and better than PCGrad+SS (55.48%). On the ACE dataset, the results have the same pattern: Nash-MTL+SS yielded the best outcome, eclipsing the performance of KT by 2.78%. The effectiveness of Nash-MTL comes from its ability to mitigate the detrimental effects originating from the disparity in magnitudes between objectives in MOO.

5 Conclusion

In this work, we introduce **SharpSeq**, a novel framework that enables the seamless integration of state-of-the-art gradient-based multi-objective optimization methods into continual event detection systems. By addressing the challenges of imbalanced training data and the unique nature of continual learning, our method significantly enhances the performance of continual event detection. Through rigorous empirical benchmarks, we demonstrate the effectiveness and versatility of our contributions, extending beyond the realm of continual event detection and showcasing the potential for leveraging multi-objective optimization in solving various continual learning problems across various domains. This work sets a solid foundation and paves the way for future research in this exciting and rapidly evolving field.

Limitations

While our proposed methods have brought a great improvement to continual event detection, it is necessary to point out certain limitations of this research. One notable limitation of SharpSeq is that it is still susceptible to some degree of catastrophic forgetting. Although Representation Generation alleviates the imbalance problem, this method might introduces some level of noise. However, this limitation can be managed through further study to control the quality of generated data. Additionally, at each task, SharpSeq has to compute backpropagation four times, one for each objective, to compute α, resulting in higher training cost. In conclusion, our proposed methods are not a definitive solution for continual event detection, and future research can focus on improving multi-objective optimization and data balancing to further enhance the method's effectiveness.

Acknowledgements

In this research, Linh Ngo Van is supported by NAVER Corporation within the framework of collaboration with the International Research Center for Artificial Intelligence (BKAI), School of Information and Communication Technology, HUST under project NAVER.2024.DA03. Thien Huu Nguyen is supported by the Army Research Office (ARO) grant W911NF-21-1-0112, the NSF grant CNS-1747798 to the IUCRC Center for Big Learning, and the NSF grant # 2239570.

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.

Pengfei Cao, Yubo Chen, Jun Zhao, and Taifeng Wang. 2020a. Incremental event detection via knowledge consolidation networks. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 707–717, Online. Association for Computational Linguistics.

Pengfei Cao, Yubo Chen, Jun Zhao, and Taifeng Wang. 2020b. Incremental event detection via knowledge consolidation networks. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 707–717.

Francisco M Castro, Manuel J Marín-Jiménez, Nicolás Guil, Cordelia Schmid, and Karteek Alahari. 2018. End-to-end incremental learning. In *Proceedings of the European conference on computer vision (ECCV)*, pages 233–248.

Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. 2018. Efficient lifelong learning with a-gem. *arXiv preprint arXiv:1812.00420*.

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

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

Mehrdad Farajtabar, Navid Azizan, Alex Mott, and Ang Li. 2020. Orthogonal gradient descent for continual learning. In *International Conference on Artificial Intelligence and Statistics*, pages 3762–3773. PMLR.

Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. 2021. Sharpness-aware minimization for efficiently improving generalization. In *International Conference on Learning Representations*.

Nam Le Hai, Trang Nguyen, Linh Ngo Van, Thien Huu Nguyen, and Khoat Than. 2024. Continual variational dropout: a view of auxiliary local variables in continual learning. *Machine Learning*, 113(1):281–323.

Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. 2019. Learning a unified classifier incrementally via rebalancing. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 831–839.

Ching-Yi Hung, Cheng-Hao Tu, Cheng-En Wu, Chien-Hung Chen, Yi-Ming Chan, and Chu-Song Chen. 2019. Compacting, picking and growing for unforgetting continual learning. In *Advances in Neural Information Processing Systems*, pages 13647–13657.

Justin M Johnson and Taghi M Khoshgoftaar. 2019. Survey on deep learning with class imbalance. *Journal of Big Data*, 6(1):1–54.

Zixuan Ke and Bing Liu. 2022. Continual learning of natural language processing tasks: A survey. *arXiv preprint arXiv:2211.12701*.

Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Thanh-Thien Le, Manh Nguyen, Tung Thanh Nguyen, Linh Ngo Van, and Thien Huu Nguyen. 2024. Continual relation extraction via sequential multi-task learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 18444–18452.

Bo Liu, Xingchao Liu, Xiaojie Jin, Peter Stone, and Qiang Liu. 2021a. Conflict-averse gradient descent for multi-task learning. *Advances in Neural Information Processing Systems*, 34:18878–18890.

Liyang Liu, Yi Li, Zhanghui Kuang, Jing-Hao Xue, Yimin Chen, Wenming Yang, Qingmin Liao, and Wayne Zhang. 2021b. Towards impartial multi-task learning. In *International Conference on Learning Representations*.

Minqian Liu, Shiyu Chang, and Lifu Huang. 2022. Incremental prompting: Episodic memory prompt for lifelong event detection. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 2157–2165, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Yaoyao Liu, Bernt Schiele, and Qianru Sun. 2021c. Adaptive aggregation networks for class-incremental learning. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition*, pages 2544–2553.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Arun Mallya and Svetlana Lazebnik. 2018. Packnet: Adding multiple tasks to a single network by iterative pruning. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 7765–7773. Computer Vision Foundation / IEEE Computer Society.

Hieu Man, Nghia Trung Ngo, Linh Ngo Van, and Thien Huu Nguyen. 2022. Selecting optimal context sentences for event-event relation extraction. In *Proceedings of the AAAI conference on artificial intelligence*, volume 36, pages 11058–11066.

Seyed Iman Mirzadeh, Mehrdad Farajtabar, Razvan Pascanu, and Hassan Ghasemzadeh. 2020. Understanding the role of training regimes in continual learning. *Advances in Neural Information Processing Systems*, 33:7308–7320.

Aviv Navon, Aviv Shamsian, Idan Achituve, Haggai Maron, Kenji Kawaguchi, Gal Chechik, and Ethan Fetaya. 2022. Multi-task learning as a bargaining game. *arXiv preprint arXiv:2202.01017*.

Chien Nguyen, Linh Ngo, and Thien Nguyen. 2023a. Retrieving relevant context to align representations for cross-lingual event detection. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 2157–2170.

Huy Nguyen, Chien Nguyen, Linh Ngo, Anh Luu, and Thien Nguyen. 2023b. A spectral viewpoint on continual relation extraction. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 9621–9629.

Hoang Phan, Lam Tran, Ngoc N Tran, Nhat Ho, Dinh Phung, and Trung Le. 2022a. Improving multi-task learning via seeking task-based flat regions. *arXiv preprint arXiv:2211.13723*.

Hoang Phan, Anh Phan Tuan, Son Nguyen, Ngo Van Linh, and Khoat Than. 2022b. Reducing catastrophic forgetting in neural networks via gaussian mixture approximation. In *Advances in Knowledge Discovery and Data Mining: 26th Pacific-Asia Conference, PAKDD 2022, Chengdu, China, May 16–19, 2022, Proceedings, Part I*, pages 106–117. Springer.

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.

Ozan Sener and Vladlen Koltun. 2018. Multi-task learning as multi-objective optimization. *Advances in neural information processing systems*, 31.

Kihyuk Sohn, Xinchen Yan, and Honglak Lee. 2015. Learning structured output representation using deep conditional generative models. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'15, page 3483–3491, Cambridge, MA, USA. MIT Press.

Linh Ngo Van, Nam Le Hai, Hoang Pham, and Khoat Than. 2022. Auxiliary local variables for improving regularization/prior approach in continual learning. In *Advances in Knowledge Discovery and Data Mining: 26th Pacific-Asia Conference, PAKDD 2022, Chengdu, China, May 16–19, 2022, Proceedings, Part I*, pages 16–28. Springer.

Gido M Van de Ven and Andreas S Tolias. 2019. Three scenarios for continual learning. *arXiv preprint arXiv:1904.07734*.

David Wadden, Ulme Wennberg, Yi Luan, and Hannaneh Hajishirzi. 2019. Entity, relation, and event extraction with contextualized span representations. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5784–5789, Hong Kong, China. Association for Computational Linguistics.

Christopher Walker, Stephanie Strassel, Julie Medero, and Kazuaki Maeda. 2006. ACE 2005 multilingual training corpus LDC2006T06. Web Download. Philadelphia: Linguistic Data Consortium.

Xiaozhi Wang, Ziqi Wang, Xu Han, Wangyi Jiang, Rong Han, Zhiyuan Liu, Juanzi Li, Peng Li, Yankai

Lin, and Jie Zhou. 2020. Maven: A massive general domain event detection dataset. *arXiv preprint arXiv:2004.13590*.

Max Welling. 2009. Herding dynamical weights to learn. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, page 1121–1128, New York, NY, USA. Association for Computing Machinery.

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*.

Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. 2019. Large scale incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 374–382.

Pengfei Yu, Heng Ji, and Prem Natarajan. 2021. Lifelong event detection with knowledge transfer. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5278–5290, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. 2020. Gradient surgery for multi-task learning. *Advances in Neural Information Processing Systems*, 33:5824–5836.

A Appendix

A.1 Workflow of Proposed Methods

The overview workflow of our proposed method is illustrated in Figure 1.

A.2 Sharpness-Aware Minimization

The traditional training process concentrates only to minimize the empirical loss, thus lead to overfitting problems where the model can not generalize the training data well and failed on the unseen data. Foret et al. (2021) introduce a procedure to mitigate this problem by minimizing the worst-case loss instead of directly optimizing the training losses. The new optimization problem is as the following:

$$\min_{\boldsymbol{\theta}} \max_{||\boldsymbol{\epsilon}||_2 \leq \rho} L(\boldsymbol{\theta} + \boldsymbol{\epsilon}), \tag{11}$$

where $||.||_2$ is the l_2 norm and ρ is the radius of the neighborhood; $\boldsymbol{\theta}$ denotes the model's parameters. To solve the inner maximization in problem 11, Foret et al. (2021) first estimate ϵ by using the first-order Taylor expansion to estimate $L(\boldsymbol{\theta} + \boldsymbol{\epsilon})$.

$$\begin{aligned}\boldsymbol{\epsilon}^* \in \underset{||\boldsymbol{\epsilon}||_2 \leq \rho}{\operatorname{argmax}} L(\boldsymbol{\theta} + \boldsymbol{\epsilon}) &\approx \underset{||\boldsymbol{\epsilon}||_2 \leq \rho}{\operatorname{argmax}} \boldsymbol{\epsilon}^T \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}) \\ &\approx \rho \frac{\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta})}{||\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta})||_2}\end{aligned}$$

Once $\boldsymbol{\epsilon}$ is approximated, the gradient of worst-case loss will be used to update the parameter $\boldsymbol{\theta}$:

$$\boldsymbol{g}^{SAM} := \nabla_{\boldsymbol{\theta}} \max_{||\boldsymbol{\epsilon}||_2 \leq \rho} L(\boldsymbol{\theta}+\boldsymbol{\epsilon}) \approx \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}+\boldsymbol{\epsilon})|_{\boldsymbol{\theta}+\boldsymbol{\epsilon}^*}$$

A.3 The effects of the quantity of generated samples

As shown in table 2, Representation Generation pushes the performance of SharpSeq by synthesizing data for old labels. To further inspect how it affects SharpSeq, we experiment SharpSeq with different ratios r between the number of generated samples and the replay set. The results of four values of r are presented in table 5. For MAVEN, $r = 10$ gains the highest performance with 60.27% F1 score in the fifth task. while for the fifth task of ACE, the best value of r is 20 with 62.08% score. The effect of r on the early tasks is relatively low but it is significant in the late tasks. We can observe that increasing the value of r does not guarantee a better performance of SharpSeq. The problem with Representation Generation is that the synthesized samples contain noise from random processes. The noise can affect the value of $\boldsymbol{\epsilon}$ in SharpSeq and navigate the MOO algorithm to optimize the model with wrong labels. Thus, when we generate more samples, we need to take into consideration how to remove noise from generated samples to avoid the bad effect.

A.4 Number of GMMs components

Since we use GMMs to synthesize new samples for old labels, it is important to understand how the number of Gaussian components affects the eventual continual event detection ability. Table 6 shows experimental results on hyperparameter g. For MAVEN, the differences in performance between values of g were very small, the best value of g in MAVEN was 6, but it was lower than $g = 4$ by 0.04% on the fifth task. In contrast, the F1-scores of SharpSeq on the ACE dataset varied a lot when g changes. The best value of g on ACE was 8 with an F1-score of 60.60% after the fifth task. However, the increase of g does not guarantee

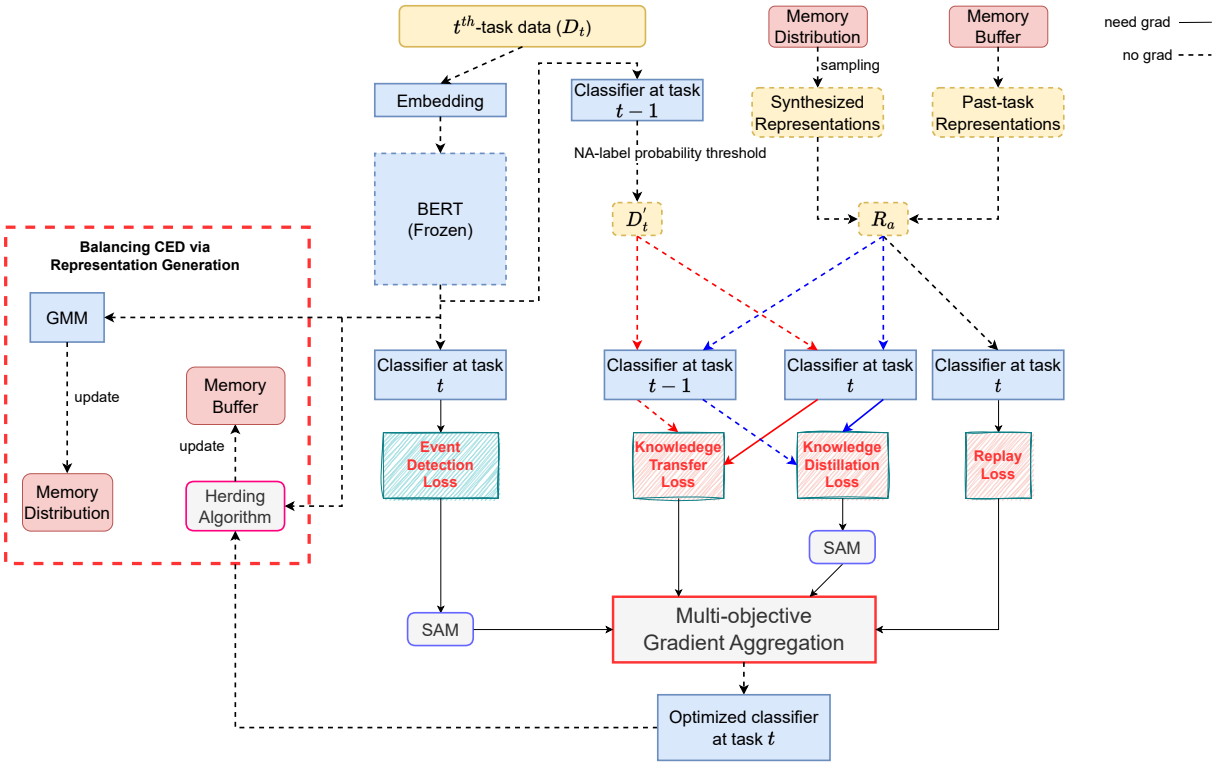

Figure 1: Overview of SharpSeq's workflow.

	MAVEN					ACE				
Task	1	2	3	4	5	1	2	3	4	5
$r = 20$	62.28	61.78	62.79	61.25	60.02	56.47	57.62	62.40	62.55	**62.08**
$r = 10$	62.28	61.85	**62.92**	**61.31**	**60.27**	56.47	57.76	64.03	61.34	60.78
$r = 5$	62.28	61.73	62.78	61.03	60.03	56.47	**58.95**	65.38	**62.64**	61.16
$r = 2$	62.28	**61.86**	62.73	60.78	59.38	56.47	58.78	**64.68**	61.75	60.11

Table 5: Ablation results for the number of generated representations in the SharpSeq method.

	MAVEN					ACE				
Task	1	2	3	4	5	1	2	3	4	5
$g = 2$	62.28	61.69	62.66	60.98	59.78	56.47	57.09	64.42	62.02	60.54
$g = 4$	62.28	61.80	62.56	61.20	**60.24**	56.47	**57.12**	64.18	61.63	**62.34**
$g = 6$	62.28	**61.83**	**62.92**	**61.28**	60.20	56.47	56.31	64.37	**62.84**	61.54
$g = 8$	62.28	61.82	62.75	61.20	60.20	56.47	56.99	**64.44**	62.47	62.60

Table 6: Ablation results on the number of GMMs components in SharpSeq.

an increase in performance: when $g = 4$, the F1-score after the fifth task was 62.34%, which was better than the results corresponding to $g = 6$ and was worse than $g = 8$ by only 0.26%. Thus, the number of components g should be tuned carefully to avoid the detrimental effect of noise and get the best result in the continual learning setting.

A.5 Implementation details

In the training phase, we use AdamW optimizer (Loshchilov and Hutter, 2017) with a learning rate of $1e-4$ and weight decay of $1e-2$. For both datasets, we set the batch size to 128 and the training process will stop after 5 epochs if the performance on the development set does not improve. The setting for KT (Yu et al., 2021) is kept with the number of instances per label in the replay set q to be 20 and the number of instances for initialization of new label h to be 20. For each method, the best results are reported by using grid search. The search ranges of each hyperparameter are as follows:

- the generation ratio r is in $[2, 5, 10, 20]$
- the number of epochs is in $[15, 30]$
- the balancing coefficient ν to balance NA label and valid labels is in $[\frac{4}{5}, \frac{10}{11}, \frac{20}{21}, \frac{30}{31}, \frac{40}{41}]$
- the number of component of GMMs g is in $[2, 4, 6, 8]$. In our experiments, we double the value of g with labels that have more than 600 instances.

All implementations are written using PyTorch; all experiments were conducted on an NVIDIA A100 and an NVIDIA V100. The source code will be published as soon as this paper is accepted.

A.6 Reproducibility Checklist

- **Source code with the specification of all dependencies, including external libraries**: The source code and the necessary documentation for reproducibility is submitted together with this paper via ACL Rolling Review submission system.
- **Description of computing infrastructure used**: In this work, since the number of experiments is very large, we use a Tesla V100-SXM2 GPU with 32GB memory operated by Ubuntu Server 18.04.3 LTS and a Tesla A100-SXM GPU with 40GB memory operated by Ubuntu 20.04. PyTorch 1.9.1 and Huggingface-Transformer 4.23.1 (Apache License 2.0) (Wolf et al., 2019) are used to implement the models.
- **Average runtime for each approach**: Each epoch of the proposed model on average takes 20 minutes for MAVEN dataset and 12 minutes for ACE dataset. We train the model for maximum 30 epochs. The best epoch is chosen based on F1 score over development data.
- **Number of parameters in the model**: The total number of parameters of our model is 335M parameters. Since we freeze the BERT model; the number of trainable parameters is thus only 1.4M.
- **Explanation of evaluation metrics used, with links to code**: We use the same performance measures (average F1-scores on 5 permutations of task orders) as in previous work (Yu et al., 2021) for fair comparisons.
- **Bounds for each hyper-parameter**: Please refer to Section A.5
- **The method of choosing hyper-parameter values and the criterion used to select among them**: The hyperparameters are tuned using random search. Hyper-parameters are chosen based on F1 scores on the development set.