Incremental Event Detection via Knowledge Consolidation Networks

Pengfei Cao[1,2], **Yubo Chen**[1,2], **Jun Zhao**[1,2], and **Taifeng Wang**[3]
[1] National Laboratory of Pattern Recognition, Institute of Automation,
Chinese Academy of Sciences, Beijing, 100190, China
[2] School of Artificial Intelligence, University of Chinese Academy of Sciences,
Beijing, 100049, China
[3] Ant Group, Hangzhou, 310013, China
{pengfei.cao, yubo.chen, jzhao}@nlpr.ia.ac.cn, taifeng.wang@antgroup.com

Abstract

Conventional approaches to event detection usually require a fixed set of pre-defined event types. Such a requirement is often challenged in real-world applications, as new events continually occur. Due to huge computation cost and storage budge, it is infeasible to store all previous data and re-train the model with all previous data and new data, every time new events arrive. We formulate such challenging scenarios as incremental event detection, which requires a model to learn new classes incrementally without performance degradation on previous classes. However, existing incremental learning methods cannot handle semantic ambiguity and training data imbalance problems between old and new classes in the task of incremental event detection. In this paper, we propose a **K**nowledge **C**onsolidation **N**etwork (**KCN**) to address the above issues. Specifically, we devise two components, *prototype enhanced retrospection* and *hierarchical distillation*, to mitigate the adverse effects of semantic ambiguity and class imbalance, respectively. Experimental results demonstrate the effectiveness of the proposed method, outperforming the state-of-the-art model by 19% and 13.4% of whole F1 score on ACE benchmark and TAC KBP benchmark, respectively.

1 Introduction

Event detection (ED) is an important task of information extraction, which aims to identify event triggers and classify them into specific types (Ahn, 2006; Chen et al., 2015). For instance, in the sentence "*A man died when a tank fired on the hotel*", an ED model should be able to recognize two events: a *Die* event triggered by the word "*died*" and an *Attack* event triggered by the word "*fired*".

Existing ED methods can only handle a fixed number of event classes (also called event types) and perform once-and-for-all training on a fixed dataset. Although the evaluation of this setting is straightforward, it clearly limits the usage of these methods in practical applications, as new events continually emerge in the real world. A practical ED system should be able to incrementally learn new classes, instead of requiring a fixed set of pre-defined event classes. Therefore, we consider a more realistic *incremental learning* setting (also called continual learning or lifelong learning) (Ring, 1994; Thrun, 1998), where a learning system learns from class-incremental data streams in which examples of different event classes arrive at different times, which can be shown in Figure 1. In such scenarios, it is often infeasible to combine the new data with all previous data to re-train the model, due to various issues such as huge computation cost, storage budget and data privacy (McMahan et al., 2017).

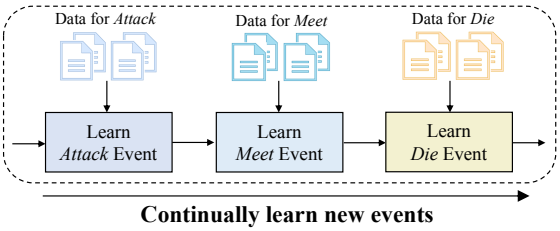


Figure 1: Incremental event detection: a model continually learns new event classes, when data for them becomes available. At any time, the model is able to perform prediction for all event classes observed so far.

Alternatively, a natural approach to incremental event detection is to simply finetune a pre-trained model on new data. However, this approach faces a serious challenge – *catastrophic forgetting* (McCloskey and Cohen, 1989; French, 1999). To be more specific, a learned system usually suffers from significant performance drop on old classes when it adapts to new classes. For example in Figure 2, after finetuning on the training data of new class (i.e., *Attack*), the updated model can recog-

Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, pages 707–717,
November 16–20, 2020.

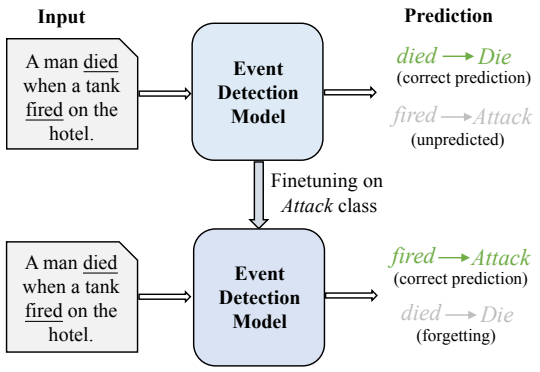


Figure 2: Catastrophic forgetting in incremental event detection. Top: An event detection model originally trained for *Die* class, successfully recognizes the *Die* event, and cannot predict *Attack* event, due to no training on *Attack* class. Bottom: After finetuning on *Attack* class, the updated model can recognize the *Attack* event, but fails to recognize the *Die* event, due to forgetting previous knowledge.

nize the *Attack* event, but fails to detect the *Die* event, which it was capable of originally.

Great efforts have been devoted to overcoming the catastrophic forgetting on the image classification task, which can be mainly divided into parameter-based methods which try to identify and preserve significant parameters of the original model (i.e., the model learned on old classes) (Kirkpatrick et al., 2017; Aljundi et al., 2018), and replay-based methods which reserve a small amount of data from each old class. When new classes arrive, the stored data and new data are combined to re-train the model (Rebuffi et al., 2017; Hou et al., 2018, 2019). Since replay-based methods are very simple and effective, such methods have dominated the research.

However, when applying replay-based methods to incremental event detection, we find two challenges: 1) *Semantic Ambiguity* problem and 2) *Class Imbalance* problem. ***Semantic Ambiguity***: Compared with simple image classification task, event detection has more serious ambiguity problem (Chen et al., 2018). For example, in the sentence "*He left the company*", the word "*left*" can not only trigger *End-Position* event, but also trigger *Transport* event. By contrast, the example "*He resigned his position as manager*" can more accurately express *End-Position* event and is more representative for the *End-Position* class. The ambiguous examples could confuse the classifier, affecting the generalization ability (Liu et al., 2014). It indicates that re-training the model with ambiguous old examples isn't conducive to retaining previous knowledge. Therefore, to alleviate semantic ambiguity, we need to reserve the most representative examples for each class. ***Class Imbalance***: Since only a small number of old data is stored while the number of new classes data is usually large, there is a serious training data imbalance between old and new classes. Under this circumstance, the focus of the training process is significantly biased towards new classes (He and Garcia, 2009; Wu et al., 2019), which severely misleads the classifier. It is a crucial cause of catastrophic forgetting.

In response to the above problems, we propose a **K**nowledge **C**onsolidation **N**etwork (**KCN**) for preserving previous knowledge. Specifically, to address the semantic ambiguity issue, we devise a *prototype enhanced retrospection* module to reserve the most representative examples for each old class. To reduce the adverse effect of class imbalance, we propose a *hierarchical distillation* module which makes the current model mimic the behaviors of the original model in feature level and event class prediction level, respectively. Experimental results demonstrate that our method outperforms previous state-of-the-art models. In summary, the contributions of this paper are listed as follows:

- To our best knowledge, we are the first to introduce the incremental event detection task and we construct two incremental event detection benchmarks by using two widely used ED datasets, ACE 2005 and TAC KBP 2017.
- We propose a knowledge consolidation network for incremental event detection task, which can efficiently alleviate catastrophic forgetting problem via prototype enhanced retrospection and hierarchical distillation.
- Experimental results show that our method outperforms previous state-of-the-art models, achieving 19% and 13.4% improvements of whole F1 score on the ACE benchmark and TAC KBP benchmark, respectively. The source code of this paper is available at `https://github.com/CPF-NLPR/IncrementalED`.

2 Problem Definition

Event detection (ED) is a subtask of event extraction (EE). We first introduce some Automatic Content Extraction (ACE) terminologies to facilitate the understanding of ED task: **Event trigger** refers

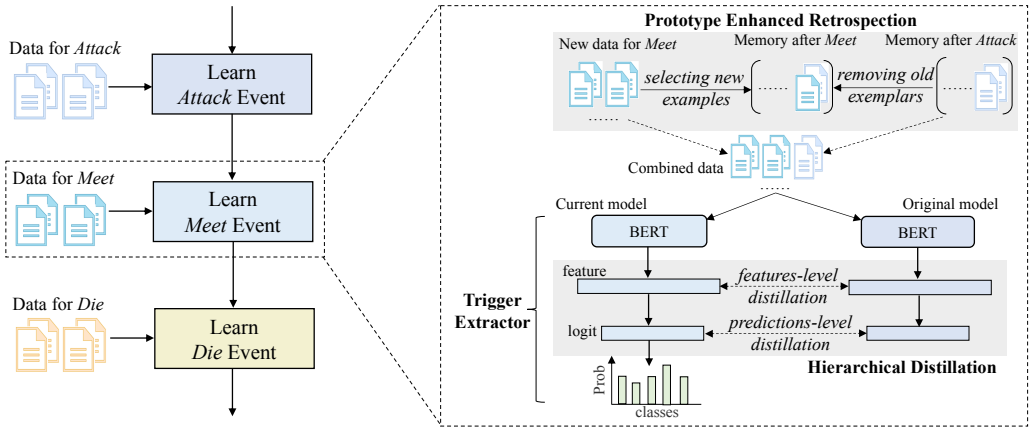


Figure 3: Illustration of **K**nowledge **C**onsolidation **N**etwork (**KCN**). When learning *Meet* event, the model is first updated with the combination of training data for *Meet* event and stored data in memory (i.e., memory after *Attack*). Then the model selects the representative examples to reserve. Since the size of memory is limited, the memory need to remove some reserved exemplars of old classes to allocate space for *Meet* event. Note that there is only one memory. For better understanding, we show the memory states after learning *Attack* and *Meet*, respectively.

to a word that most clearly expresses the occurrence of an event. **Event arguments** are participants of the event. **Event mention** refers to a phrase or sentence within which an event is described.

Given a sentence or document, an ED system aims to locate event triggers and categorize their types. For example, in the sentence "*A man died in the hospital*", an ED system is expected to detect a *Die* event along with the trigger word "*died*". Following previous work (Chen et al., 2015; Nguyen et al., 2016; Liu et al., 2018a), we formulate ED as a token-level multi-class classification task. Namely, given a sentence, we treat every token in it as a trigger candidate, and we aim to classify each candidate into pre-defined event classes.

In real world, new event classes continually occur. Therefore, a practical ED system should enable to incrementally learn new event classes. We introduce a new problem, *incremental event detection*. Suppose that there a class-incremental data stream, denoted as $\{\mathcal{X}^{(1)}, \mathcal{X}^{(2)}, \ldots, \mathcal{X}^{(M)}\}$. Each $\mathcal{X}^{(k)}$[1] contains training/validation/testing data $(\mathcal{X}^{(k)}_{\text{train}}, \mathcal{X}^{(k)}_{\text{valid}}, \mathcal{X}^{(k)}_{\text{test}})$ and its own event class set $\mathcal{C}^{(k)}$. Any two event class sets are disjoint (i.e., $\mathcal{C}^{(i)} \bigcap \mathcal{C}^{(j)} = \emptyset$, $i \neq j$). At step k, the event detection model optimizes its parameters using the training data $\mathcal{X}^{(k)}_{\text{train}}$ and the updated model should still perform well on all previous observed classes. That is to say, during testing stage at step k, we evaluate the updated model on the testing data of all observed classes (i.e., $\bigcup_{i=1}^{k} \mathcal{X}^{(i)}_{\text{test}}$). Given an input from $\mathcal{X}^{(j)}$ $(j \leq k)$, the model ought to give a prediction from $\bigcup_{i=1}^{k} \mathcal{C}^{(i)}$, instead of $\mathcal{C}^{(j)}$.

To alleviate catastrophic forgetting, a bounded memory is allowed to store a small amount of old classes data. Therefore, every time new classes arrive, the event detection model exploits the reserved old classes data and the training data of new classes to update parameters.

3 Method

We propose a **K**nowledge **C**onsolidation **N**etwork (**KCN**) for incremental event detection, which is illustrated in Figure 3. The model consists of three important components: 1) *Trigger Extractor*, 2) *Prototype Enhanced Retrospection* and 3) *Hierarchical Distillation*. When new classes occur, the proposed KCN model updates the parameters with the combination of reserved old classes data and training data of new arriving classes. The prototype enhanced retrospection retains previous knowledge via reserving the most representative examples. The hierarchical distillation transfers the previous knowledge from the original model to the current model. We will detail these three components.

3.1 Trigger Extractor

Our trigger extractor is based on the Transformer architecture (Vaswani et al., 2017). We use the state-of-the-art text encoder BERT (Devlin et al., 2019) to encode the input sentence. BERT is a multi-layer bidirectional Transformer, pre-trained on a large-scale unlabeled corpus, which has achieved the state-of-the-art performance for event detection

[1] In general, $\mathcal{X}^{(k)}$ can contain one or more new classes.

(Wang et al., 2019b; Yang et al., 2019).

When new classes $\mathcal{C}^{(k)}$ arrive, the training data of the new classes is denoted as $\mathcal{X}^{(k)}_{\text{train}} = \{(X_i, Y_i), 1 \leq i \leq K\}$, where K is the number of training examples, X_i is the sentence and Y_i denotes the label of each token in the sentence X_i. The memory reserves the representative examples for old m classes (i.e., $m = |\bigcup_{i=1}^{k-1} \mathcal{C}^{(i)}|$), denoted as $\mathcal{P} = \{\mathcal{P}^{(1)}, \ldots, \mathcal{P}^{(m)}\}$, where $\mathcal{P}^{(i)}$ is the set of stored examples for i-th class. We combine the stored old data and training data of new classes data, denoted as $\mathcal{N} = \mathcal{P} \bigcup \mathcal{X}^{(k)}_{\text{train}}$, to train the current model. The target label set contains all observed event types, denoted as $\mathcal{C}^o = \bigcup_{i=1}^{k} \mathcal{C}^{(i)}$. The token [CLS] and [SEP][2] are placed at the start and end of the sentence, respectively. We first leverage BERT to obtain the contextual representation for each token. Then a multi-classifier (i.e., softmax classifier) is added on BERT to predict event types. As general, we adopt cross entropy as the loss function to train the event detection model:

$$\mathcal{L}_{ce} = -\frac{1}{|\mathcal{N}|} \sum_{X \in \mathcal{N}} \sum_{x \in X} \boldsymbol{y} \cdot \log(\boldsymbol{p}) \qquad (1)$$

where $\boldsymbol{y}$ is the one-hot ground-truth label for token x and $\boldsymbol{p}$ is the corresponding class probabilities obtained by softmax. The size of them is the number of all observed classes, i.e., $|\mathcal{C}^o|$.

Since event detection has serious ambiguity problems and the capacity of memory is bounded, we need to reserve the most representative examples for old classes. We devise a prototype enhanced retrospection to achieve the objective.

3.2 Prototype Enhanced Retrospection

In our model, all classes are treated equally, i.e., when m classes have been learned so far and B is the total number of examples that can be reserved in a memory, our model will store $n = B/m$ examples for each class. Since the size of memory is limited, when new classes arrive, the memory unit performs two operations: one to select representative examples to reserve for new classes and one to remove some stored examples of old classes to allocate space for new classes.

3.2.1 Selecting Representative Examples

At step k, when the training data of new classes $\mathcal{C}^{(k)}$ is added to the model, we compute the *prototype* for each class in $\mathcal{C}^{(k)}$:

$$\boldsymbol{\mu}_c = \frac{1}{N_c} \sum_{i=1}^{N_c} \boldsymbol{z}_i \qquad (2)$$

where N_c is the number of training samples for class $c \in \mathcal{C}^{(k)}$ and $\boldsymbol{z}_i$ is representation of sentence X_i belonging to class c. Expressing an event usually requires the whole of information of a sentence, therefore, we use the representation of [CLS] token as the sentence representation. We refer to $\boldsymbol{\mu}_c$ as the *prototype* of class c.

Inspired by the recent works about prototype learning (Snell et al., 2017; Yang et al., 2018) which refer to the prototype as the class representative point in feature space, we devise a prototype-based selection algorithm. For each class, the algorithm computes the distance between each training example and the corresponding prototype, and then produces a sorted list of samples of one class based on the distance to the prototype of that class. Intuitively, the closer the example to the prototype, the more representative the example is for the class. Based on the sorted list of examples, the first n examples of the list are selected as exemplars to store in a bounded memory.

3.2.2 Removing Reserved Exemplars

Since the storage size of memory is constant, when new classes arrive, the memory needs to remove some reserved examples of old classes to allocate space for the examples from new classes. Suppose the number of old and new classes is m and t ($t = |\mathcal{C}^{(k)}|$), respectively. The memory needs to remove $B/m - B/(m+t)$ stored examples of each old class. We adopt a data-independent removal strategy. For each old class, we remove examples that are far from prototype according to the sorted list of examples. In this way, the most representative examples of old classes are still reserved.

3.3 Hierarchical Distillation

Although storing a small number of old data is very useful to improve performance, the number of training samples between old and new classes is very imbalanced, which makes the model have a bias towards the new classes, resulting in severely forgetting previous knowledge (Wu et al., 2019; Hou et al., 2019).

Since the original model isn't trained on the new classes data, it is less biased towards the new classes compared with the current model. Therefore, if the knowledge of the original model is well

[2] [CLS] and [SEP] are special tokens of BERT.

preserved, the adverse effects of the imbalance will be efficiently mitigated. Knowledge distillation is an effective way to transfer knowledge from one network to another (Hinton et al., 2015). Inspired by it, we propose a hierarchical distillation to learn the previous knowledge from the original model. The core idea is as follows: for the same input, if the current model can extract similar features, and give similar prediction distributions with the original model on previous classes, it can be assumed that the current model can efficiently preserve the previous knowledge.

3.3.1 Features-level Distillation

We use the BERT as feature extractor, denoted as $f(\cdot)$. For the input x, the extracted features of original model and current model are $f^*(x)$ and $f(x)$, respectively. To enforce extracted features not to bias towards new classes, we propose a features-level distillation loss function:

$$\mathcal{L}_{fd} = \frac{1}{|\mathcal{N}|} \sum_{X \in \mathcal{N}} \sum_{x \in X} 1 - \langle \bar{f}^*(x), \bar{f}(x) \rangle \quad (3)$$

where $\bar{f}^*(x)$ and $\bar{f}(x)$ are l_2-normalized features extracted by the original model and current model, respectively. $\langle \bar{f}^*(x), \bar{f}(x) \rangle = \bar{f}^*(x)^T \bar{f}(x)$ measures the cosine similarity between two normalized feature vectors. The features-level distillation loss is computed for all samples from the new classes and reserved exemplars.

If the extracted features of the current model don't greatly deviate from the ones of the original model, the feature extractor can effectively preserve previous knowledge.

3.3.2 Predictions-level Distillation

Besides features-level distillation loss, we also propose a predictions-level distillation loss, which preserves the previous knowledge of classifier by encouraging the current predictions on old classes to match the soft labels by the original model. At step k, t ($t = |\mathcal{C}^{(k)}|$) new classes arrive, and the model has observed m old classes. For token x, the output logits (i.e., the results before softmax) of the original and current model are $\boldsymbol{o}^* = [o_1^*, o_2^*, \ldots, o_m^*]$ and $\boldsymbol{o} = [o_1, o_2, \ldots, o_{m+1}, \ldots, o_{m+t}]$, respectively. The predictions-level distillation loss is formulated as follows:

$$\begin{aligned} \mathcal{L}_{pd} &= -\frac{1}{|\mathcal{N}|} \sum_{X \in \mathcal{N}} \sum_{x \in X} \sum_{i=1}^{m} \tau_i^* \log(\tau_i), \\ \tau_i^* &= \frac{e^{o_i^*/T}}{\sum_{j=1}^{m} e^{o_j^*/T}}, \quad \tau_i = \frac{e^{o_i/T}}{\sum_{j=1}^{m} e^{o_j/T}} \end{aligned} \quad (4)$$

where T is the temperature scalar, which is usually set to be greater than 1 (e.g., $T = 2$ in our experiments) to increase the weights of small values. The predictions-level distillation loss is also computed for all samples from the new classes and reserved exemplars from the old classes.

Algorithm 1 Training Procedures of KCN

Input: training data $\mathcal{X}_{\text{train}}^{(k)}$ at step k, the number of new classes $t = |\mathcal{C}^{(k)}|$
Require: memory capacity B
Require: current model parameters Θ
Require: current reserved exemplar sets $\mathcal{P} = (\mathcal{P}^{(1)}, \ldots, \mathcal{P}^{(m)})$, m classes occured

```
Form combined training set: X_train^(k) ⋃ P
Update the model parameters Θ with loss function L = L_ce + αL_fd + βL_pd
Compute the number of exemplars per class: n ← B/(m + t)
for c = 1, ..., m do
    Remove some reserved exemplars for old class c until the number of exmplars is n
end for
for c = m + 1, ..., m + t do
    Compute the prototype for new class c via Equation (2)
    Select the examples to store in a memory according to the distance to prototype
end for
```

3.4 Training

Our approach addresses the catastrophic forgetting problem by storing the most representative examples for each class and reducing the impact of class imbalance. Combining the losses presented above, we reach a total loss comprised of three terms, given as:

$$\mathcal{L} = \mathcal{L}_{ce} + \alpha \mathcal{L}_{fd} + \beta \mathcal{L}_{pd} \quad (5)$$

where α and β are adjustment coefficients. If α and β are very large, the model will place more emphasis on the old classes, hurting the performance on

new classes. The coefficients are used to balance the performance on old classes and new classes.

The overall training procedures is outlined in Algorithm 1.

4 Experiments

4.1 Incremental Event Detection Benchmarks

So far, there is no benchmark for evaluating incremental event detection models. Therefore, we propose the following construction method: for a given event detection dataset, the classes are arranged in a fixed order. Each method is then trained in a class-incremental way on the available training data. Based on two widely used datasets, ACE 2005[3] and TAC KBP 2017[4], we introduce two instantiations of the above construction method. 1) *ACE benchmark*: We use the dataset split of ACE dataset as suggested in (Li et al., 2013; Chen et al., 2015; Lu et al., 2019; Liu et al., 2019). Since the ACE dataset has long-tail frequency distribution, we exploit the data of the top 10 most frequent classes. 2) *TAC KBP benchmark*: We exploit the official training and testing data of TAC KBP. Same as the ACE benchmark, we also use the data of the top 10 most frequent classes. For both of two benchmarks, one new class is available for the model at each time.

4.2 Evaluation Metrics and Implementation Details

For conventional event detection, Precision (P), Recall (R) and F1 score are used as evaluation metrics. For incremental event detection, every time the model finishes training on the new classes data, we report the F1 score on the test data of all observed classes. For example, after time step k, the result is denoted as $F1_k$. Therefore, these results can be plotted as a curve. In addition, if a single number is preferable, we report the ***Average F1*** which is the average of these F1 scores (i.e., $\frac{1}{k}\sum_{i=1}^{k} F1_i$), and ***Whole F1*** which is the F1 score on the whole testing data of all classes.

Our method uses the HuggingFace's Transformers library[5] to implement BERT base model. The learning rate is set to 2e-5. The batch size is 16. The adjustment coefficients α and β are both 0.5. For the two benchmarks, the capacity of memory is 100 and 500, respectively.

4.3 Baselines

We compare our approach KCN with the following baselines:

EWC (Kirkpatrick et al., 2017). The method is proposed to keep the network parameters close to the optimal parameters for the previous classes while training new classes data. It is the representative work of parameter-based methods.

EMR (Wang et al., 2019a). The method alleviates the impact of catastrophic forgetting via randomly storing some samples of old classes, which is the representative work of replay-based methods.

LwF (Li and Hoiem, 2017). The method aims to match the softmax output of the network of previous models for old classes, which is also a widely used incremental learning method.

Finetune. A straightforward method which simply finetunes the pre-trained model on new arriving data. We exploit the state-of-the-art model (i.e., BERT) for finetuning.

UpperBound. A model is trained using all training samples from all observed classes, which can provide the upper bound of the benchmark.

4.4 Compared with State-of-the-art Methods

We conduct experiments on ACE benchmark and KBP benchmark to compare our proposed methods with the above baselines. The F1 scores over all observed classes during the whole incremental learning process are presented in Figure 4. We also list the results at the last step in Table 1. From the results, we can observe that:

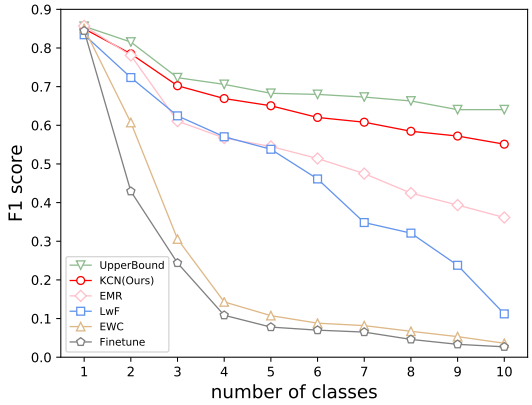
(1) Our method outperforms all the baselines by a large margin. For example, compared with the state-of-the-art model EMR, our method achieves 19% and 13.4% improvements of whole F1 score on the ACE benchmark and TAC KBP benchmark, respectively. It indicates that our proposed method KCN is very effective for the incremental event detection task.

(2) EMR and LwF can achieve competitive performance at the beginning. However, the gap between the two baselines and our method KCN becomes wider as more new classes arrive. The reason is that data imbalance becomes more serious as the incremental step increases, since the total capacity of memory is bounded. It demonstrates

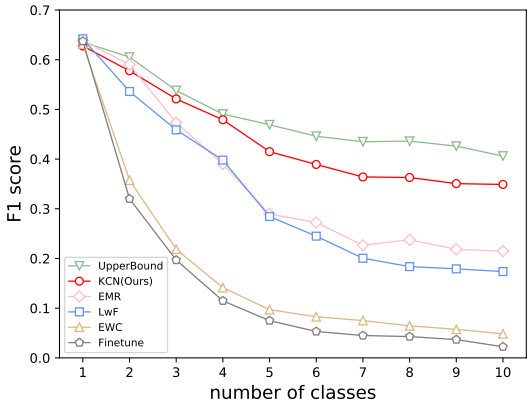
[3] https://catalog.ldc.upenn.edu/LDC2006T06

[4] https://tac.nist.gov/2017/KBP/data.html

[5] https://github.com/huggingface/transformers



(a) ACE benchmark



(b) TAC KBP benchmark

Figure 4: The performance on (a) ACE benchmark and (b) TAC KBP benchmark. Our approach achieves better performance than other incremental learning methods.

Method	ACE		TAC KBP	
	Avg	Whole	Avg	Whole
Finetune	19.5	2.7	15.4	2.2
EWC	23.4	3.6	17.7	4.8
LwF	47.7	11.2	33.1	17.4
EMR	55.3	36.1	35.5	21.5
KCN(Ours)	**65.9**	**55.1**	**44.3**	**34.9**

Table 1: The average F1 (%) on all observed classes ("Avg" column), and whole F1 (%) on the whole testing data ("Whole" column) after the last time step.

Models	ACE		TAC KBP	
	Avg	Whole	Avg	Whole
KCN	**65.9**	**55.1**	**44.3**	**34.9**
w/o PS	62.1	51.6	41.8	32.2
w/o FD	64.1	53.2	42.4	32.7
w/o PD	59.7	50.6	38.2	27.4
w/o HD	58.2	44.3	36.8	24.0
w/o PS and HD	55.3	36.1	35.5	21.5

Table 2: Ablation studies by removing the main components, where "w/o" indicates without. "PS", "FD", "PD" and "HD" refer to "prototype-based selection", "features-level distillation", "predictions-level distillation" and "hierarchical distillation", respectively. Actually, "HD" is the combination of "FD" and "PD".

that data imbalance between old and new classes are well handled in our approach.

(3) Finetuning always achieves the worst results on the two benchmarks, confirming that catastrophic forgetting is indeed a major problem in incremental event detection. In addition, the gap between these incremental learning methods and the UpperBound model indicates that incremental event detection is a very challenging task.

4.5 Ablation Experiment

To investigate the effectiveness of the prototype enhanced retrospection and hierarchical distillation, we conduct the ablation studies. The experimental results are listed in Table 2. Overall, we can observe that:

(1) Effectiveness of Prototype Enhanced Retrospection. Compared with the model removed prototype-based selection (PS), i.e., randomly selecting examples, our model KCN improves the average F1 score from 62.1% to 65.9% on the ACE benchmark. It indicates the prototype enhanced retrospection is able to effectively select the most representative examples for each class.

(2) Effectiveness of Hierarchical Distillation. We can see that removing any submodule of hierarchical distillation (features-level distillation (FD) or predictions-level distillation (PD)) brings performance degradation. If we remove the entire hierarchical distillation module (HD), the performance further declines. It demonstrates the hierarchical distillation is very effective to alleviate the class imbalance problem.

(3) Effectiveness of Prototype Enhanced Retrospection and Hierarchical Distillation. When we remove the prototype-based selection and hierarchical distillation, the performance drops significantly. The whole F1 score drops from 34.9% to 21.5% on the TAC KBP benchmark. It indicates simultaneously exploiting the prototype enhanced retrospection and hierarchical distillation is also

	KCN(Ours)		EMR	
	Avg	Whole	Avg	Whole
10	64.2	53.7	47.5	34.3
20	65.4	55.3	53.6	36.4
30	66.5	56.8	58.3	45.5
40	67.4	57.7	59.6	48.6
50	68.2	59.1	63.1	51.1
60	**69.1**	**60.8**	**65.2**	**53.4**

Table 3: The effect of the number of reserved samples. We compare our method KCN with the replay-based method EMR on the ACE benchmark.

KCN	① Thousands could die. ② He was killed in action in Iraq. ③ And they sent him to Baghdad and killed. ④ 16 people were killed.
EMR	① The baby fell 80 feet. ② A Oh , I 'm sorry to hear that. ③ A Viet Cong soldier dropped dead. ④ He said Sharif drowned.

Figure 5: Sampling some reserved examples of *Die* class for case study to qualitatively illustrate the effectiveness of our method.

very effective.

4.6 Discussion

4.6.1 The effect of the number of reserved samples

To reserve a few samples has proven much helpful to maintain the performance on old classes (Rebuffi et al., 2017; Wang et al., 2019a). Table 3 shows the comparison of our approach with EMR reserving different number of samples per class. From the results, we can observe that:

(1) The more samples reserved, the better performance for both EMR and our approach KCN. While in each case, the results of our approach are superior to those of EMR. It demonstrates the effectiveness of our proposed method.

(2) Even reserved fewer samples, our method still achieves better performance than EMR, which indicates that the prototype enhanced retrospection can select the most representative examples and hierarchical distillation enables to reduce the impact of class imbalance.

4.6.2 Case Study

We sample some reserved examples of *Die* event type of our approach and EMR during training to conduct a case study for qualitatively analyzing the effects of our method. It is shown in Figure 5. Although the reserved sentence "*A Oh, I 'm sorry to hear that*" of EMR may express the *Die* event, it is very obscure and implicit. By contrast, the reserved examples of our method can more accurately express the type and are more representative. The example qualitatively demonstrates our method is able to select the most representative examples.

5 Related Work

In this section, we briefly review two related topics: event detection and incremental learning.

5.1 Event Detection

Event detection (ED) is a very important task in information extraction. The proposed models can be divided into feature-based methods (Ahn, 2006; Ji and Grishman, 2008; Liao and Grishman, 2010; Li et al., 2013; Li and Ji, 2014) and neural network-based methods (Chen et al., 2015; Nguyen and Grishman, 2015; Liu et al., 2017, 2018a,b; Lu et al., 2019; Ding et al., 2019).

In feature-based methods, a diverse set of features is exploited to predict event. Ahn (2006) leverage lexical features, syntactic features and external knowledge features to extract the event. Hong et al. (2011) propose a cross-event and cross-entity inference method to capture more clues. However, feature-based methods rely heavily on handcrafted features, limiting the scalability and robustness. In recent years, neural network-based methods have dominated the research. Chen et al. (2015) exploit the convolutional neural network for event extraction. Nguyen et al. (2016) leverage the recurrent neural network to model the dependency of triggers and arguments. Wang et al. (2019b) and Yang et al. (2019) exploit the BERT for event detection.

Despite the vast progress that event detection has made in recent years, these existing methods cannot address incremental event detection task.

5.2 Incremental Learning

Incremental learning has been a long standing problem in machine learning (Cauwenberghs and Poggio, 2001; Kuzborskij et al., 2013). Existing methods can be divided into two categories, parameter-based methods (Kirkpatrick et al., 2017; Aljundi et al., 2018) and replay-based methods (Rebuffi et al., 2017; Hou et al., 2019; Wang et al., 2019a).

In parameter-based methods, these methods try

to identify and preserve significant parameters of the original model (Kirkpatrick et al., 2017; Zenke et al., 2017; Aljundi et al., 2018). However, it is difficult to design a reasonable metric to evaluate all the parameters. In replay-based methods, these methods preserve the previous knowledge via storing a small number of old data (Castro et al., 2018; Wang et al., 2019a; Han et al., 2020). Wang et al. (2019a) propose an episodic memory replay (EMR) method which randomly selects examples to reserve in a memory.

Despite the effectiveness of these methods on image classification tasks, these methods cannot handle semantic ambiguity problem and class imbalance problem in incremental event detection.

6 Conclusion

In this paper, we introduce incremental learning into event detection and propose a knowledge consolidation network to preserve previously learned knowledge. To alleviate semantic ambiguity, we devise the prototype enhanced retrospection to reserve the most representative examples. Moreover, to mitigate the adverse effect of class imbalance, we propose the hierarchical distillation to learn the previous knowledge from the original model. Experimental results demonstrate that our model outperforms previous state-of-the-art methods.

Acknowledgments

We thank the anonymous reviewers for their insightful comments. This work is supported by the Natural Key R&D Program of China (No.2017YFB1002101), the National Natural Science Foundation of China (No.61533018, No.61976211, No.61806201) and the Key Research Program of the Chinese Academy of Sciences (Grant NO. ZDBS-SSW-JSC006). This work is also supported by the CCF-Tencent Open Research Fund, a grant from Ant Group and independent research project of National Laboratory of Pattern Recognition.

References

David Ahn. 2006. The stages of event extraction. In *Proceedings of the Workshop on Annotating and Reasoning about Time and Events*, pages 1–8.

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*, pages 139–154.

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*, pages 233–248.

Gert Cauwenberghs and Tomaso Poggio. 2001. Incremental and decremental support vector machine learning. In *Advances in neural information processing systems*, pages 409–415.

Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. 2015. Event extraction via dynamic multi-pooling convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 167–176.

Yubo Chen, Hang Yang, Kang Liu, Jun Zhao, and Yantao Jia. 2018. Collective event detection via a hierarchical and bias tagging networks with gated multi-level attention mechanisms. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1267–1276.

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*, pages 4171–4186.

Ning Ding, Ziran Li, Zhiyuan Liu, Haitao Zheng, and Zibo Lin. 2019. Event detection with trigger-aware lattice neural network. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 347–356.

Robert M French. 1999. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135.

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

Haibo He and Edwardo A Garcia. 2009. Learning from imbalanced data. *IEEE Transactions on knowledge and data engineering*, 21(9):1263–1284.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Yu Hong, Jianfeng Zhang, Bin Ma, Jianmin Yao, Guodong Zhou, and Qiaoming Zhu. 2011. Using cross-entity inference to improve event extraction. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Volume 1*, pages 1127–1136. Association for Computational Linguistics.

Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. 2018. Lifelong learning via progressive distillation and retrospection. In *Proceedings of the European Conference on Computer Vision*, pages 437–452.

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

Heng Ji and Ralph Grishman. 2008. Refining event extraction through cross-document inference. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics*, pages 254–262.

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

Ilja Kuzborskij, Francesco Orabona, and Barbara Caputo. 2013. From n to n+ 1: Multiclass transfer incremental learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3358–3365.

Qi Li and Heng Ji. 2014. Incremental joint extraction of entity mentions and relations. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, Volume 1*, pages 402–412.

Qi Li, Heng Ji, and Liang Huang. 2013. Joint event extraction via structured prediction with global features. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, Volume 1*, pages 73–82.

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

Shasha Liao and Ralph Grishman. 2010. Using document level cross-event inference to improve event extraction. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 789–797. Association for Computational Linguistics.

Jian Liu, Yubo Chen, Kang Liu, and Jun Zhao. 2019. Neural cross-lingual event detection with minimal parallel resources. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 738–748.

Shaobo Liu, Rui Cheng, Xiaoming Yu, and Xueqi Cheng. 2018a. Exploiting contextual information via dynamic memory network for event detection. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1030–1035.

Shulin Liu, Yubo Chen, Kang Liu, and Jun Zhao. 2017. Exploiting argument information to improve event detection via supervised attention mechanisms. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, Volume 1*, pages 1789–1798.

Wei Liu, Gang Hua, and John R Smith. 2014. Unsupervised one-class learning for automatic outlier removal. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3826–3833.

Xiao Liu, Zhunchen Luo, and He-Yan Huang. 2018b. Jointly multiple events extraction via attention-based graph information aggregation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1247–1256.

Yaojie Lu, Hongyu Lin, Xianpei Han, and Le Sun. 2019. Distilling discrimination and generalization knowledge for event detection via delta-representation learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4366–4376.

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

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, pages 1273–1282.

Thien Huu Nguyen, Kyunghyun Cho, and Ralph Grishman. 2016. Joint event extraction via recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 300–309.

Thien Huu Nguyen and Ralph Grishman. 2015. Event detection and domain adaptation with convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing Volume 2*, pages 365–371.

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.

Mark Bishop Ring. 1994. Continual learning in reinforcement environments.

Jake Snell, Kevin Swersky, and Richard Zemel. 2017. Prototypical networks for few-shot learning. In *Advances in neural information processing systems*, pages 4077–4087.

Sebastian Thrun. 1998. Lifelong learning algorithms. In *Learning to learn*, pages 181–209. Springer.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

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

Xiaozhi Wang, Xu Han, Zhiyuan Liu, Maosong Sun, and Peng Li. 2019b. Adversarial training for weakly supervised event detection. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1*, pages 998–1008.

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 Conference on Computer Vision and Pattern Recognition*, pages 374–382.

Hong-Ming Yang, Xu-Yao Zhang, Fei Yin, and Cheng-Lin Liu. 2018. Robust classification with convolutional prototype learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3474–3482.

Sen Yang, Dawei Feng, Linbo Qiao, Zhigang Kan, and Dongsheng Li. 2019. Exploring pre-trained language models for event extraction and generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5284–5294.

Friedemann Zenke, Ben Poole, and Surya Ganguli. 2017. Continual learning through synaptic intelligence. In *Proceedings of the 34th International Conference on Machine Learning, Volume 70*, pages 3987–3995. JMLR. org.