Hyperparameter-free Continuous Learning for Domain Classification in Natural Language Understanding

Ting Hua   **Yilin Shen**   **Changsheng Zhao**   **Yen-Chang Hsu**   **Hongxia Jin**

Samsung Research America

`{ting.hua,yilin.shen,changsheng.z,yenchang.hsu,hongxia.jin}@samsung.com`

Abstract

Domain classification is the fundamental task in natural language understanding (NLU), which often requires fast accommodation to new emerging domains. This constraint makes it impossible to retrain all previous domains, even if they are accessible to the new model. Most existing continual learning approaches suffer from low accuracy and performance fluctuation, especially when the distributions of old and new data are significantly different. In fact, the key real-world problem is not the absence of old data, but the inefficiency to retrain the model with the whole old dataset. Is it potential to utilize some old data to yield high accuracy and maintain stable performance, while at the same time, without introducing extra hyperparameters? In this paper, we proposed a hyperparameter-free continual learning model for text data that can stably produce high performance under various environments. Specifically, we utilize Fisher information to select exemplars that can "record" key information of the original model. Also, a novel scheme called dynamical weight consolidation is proposed to enable hyperparameter-free learning during the retrain process. Extensive experiments demonstrate that baselines suffer from fluctuated performance and therefore useless in practice. On the contrary, our proposed model CCFI significantly and consistently outperforms the best state-of-the-art method by up to 20% in average accuracy, and each component of CCFI contributes effectively to overall performance.

1   Introduction

Catastrophic forgetting is the well-known Achilles' heel of deep neural networks, that the knowledge learned from previous tasks will be forgotten when the networks are retrained to adapt to new tasks. Although this phenomenon has been noticed as early as the birth of neural networks (French, 1999; McCloskey and Cohen, 1989), it didn't attract much attention until deep neural networks have achieved impressive performance gains in various applications (LeCun et al., 2015; Krizhevsky et al., 2012).

Domain classification is the task that mapping the spoken utterances to natural language understanding domains. It is widely used in intelligent personal digital assistant (IPDA) devices, such as Amazon Alexa, Google Assistant, and Microsoft Cortana. As many IPDAs now allow third-party developers to build and integrate new domains (Kumar et al., 2017), these devices are eager to continual learning technologies that can achieve high performance stably (Kim et al., 2018a,b). However, most traditional IPADs only work with well-separated domains built by specialists (Tur and De Mori, 2011) or customized designed for specific datasets (Li et al., 2019).

There is still a lack of continual learning methods that capable of general domain classification. Most previous approaches capable of continual learning focus on the scenario that the new model should be retrained without any access to old data (Li and Hoiem, 2016; Kirkpatrick et al., 2017; Lopez-Paz and Ranzato, 2017). However, these methods often involve more parameters that require extensive efforts in expert tuning. And when data distributions of new tasks are obviously different from the original task (e.g. class-incremental learning), these approaches can hardly maintain good accuracy for both tasks and may suffer from fluctuations in performance. On the other hand, old data are not unavailable in many practical cases. The main concerns arise from the tremendous cost in memory and computation resources, if the model is updated with huge previous datasets. Also, most continual learning approaches are applied to image data that little attention has been paid to text data. Is it possible to develop a desirable model capable of continual learning that satisfies the following qualities? 1) **High accuracy with limited old data**. Compared to the extreme cases that no access or full access to

old data, it is more practical to put models under the setting with a limited amount of old data available (e.g., no more than 10% of original data). In this case, models can achieve good performance without too much additional cost in physical resources and can be conveniently renewed with periodical system updates. 2) **High stability with zero extra parameters**. Many continual learning models can perform well only under restricted experiment settings, such as specific datasets or carefully chosen parameters. However, practical datasets are usually noisy and imbalanced distributed, and inexperienced users can't set suitable parameters in real-world applications. Therefore, it is desirable to develop a hyperparameter-free model that can work stably under various experimental environments.

To achieve these goals, we proposed a *Continous learning model based on weight Consolidation and Fisher Information sampling* (CCFI), with application to domain classification. The main challenge is how to "remember" information from original tasks, not only the representative features from data, but also the learned parameters of the model itself. This is a non-trivial contribution since "uncontrollable changes" will happen to neural network parameters whenever the feature changes. To avoid such "uncontrollable changes", previous work iCarL even discards deep neural networks as its final classifier and turns to k-nearest neighbors algorithm for actual prediction (Rebuffi et al., 2017). Our work demonstrates that these changes are "controllable" with exemplar selected by Fisher information and parameters learned by Dynamical Weight Consolidation. Our contributions can be summarized as follows.

- **Fisher information sampling**. Good exemplars are required to "remember" key information of old tasks. Unlike previous work using simple mean vectors to remember the information of old data, exemplars selected by Fisher information record both the features of data and the information of the original neural network.
- **Dynamical weight consolidation**. The need for hyperparameter is an inherited problem of regularization-based continual learning. Previous works search for this hyperparameter by evaluating the whole task sequence, which is supposed not to be known. This work provides a simple auto-adjusted weighting mechanism, making the regularization strategy possible for a practical application. Also, traditional weight consolida-

tion methods such as EWC (Kirkpatrick et al., 2017) are designed for sequential tasks with similar distributions. We extend it to the incremental learning scenario and add more regularity to achieve better stability.

- **Extensive experimental validation**. Most existing continual learning methods are designed for image data, while few previous attempts working on text data are often limited to specific usage scenarios and rely on fine-tuned parameters. Our proposed CCFI model is a general framework that can be efficiently applied to various environments with the least efforts in parameter tuning. Our extensive experimental results demonstrate the proposed CCFI can outperform all state-of-the-art methods, and provide insights into the working mechanism of methods.

2 Related Work

Although most of the existing approaches are not directly applicable to our problem, several main branches of research related to this work can be found: exemplar selection, regularization of weights, and feature extraction or fine-tune method based on pre-trained models.

Our problem is closest to the setting of exemplar selection methods (Rebuffi et al., 2017; Li et al., 2019). These approaches store examples from original tasks, and then combine them together with the new domain data for retraining. iCarL (Rebuffi et al., 2017) discards the classifier based on neural network to prevent the catastrophic forgetting, and turns to traditional K-nearest neighbor classifier.

To avoid large changes of important parameters, regularization models (Kirkpatrick et al., 2017; Li and Hoiem, 2016; Zenke et al., 2017) add constraints to the loss function. They usually introduce extra parameters requiring careful initialization. And it has been shown that their performance will drop significantly if the new tasks are drawn from different distributions (Rebuffi et al., 2017). On the contrary, our proposed CCFI is a parameter-free model that can produce stable performance under various experimental environments.

Feature extraction methods utilize pre-trained neural networks to calculate features of input data (Donahue et al., 2014; Sharif Razavian et al., 2014). They make little modifications to the original network but often result in a limited capacity for learning new tasks. Fine-tuning models (Girshick et al., 2014) can modify all the parameters in order to

achieve better performance in new tasks. Although starting with a small learning rate can indirectly preserve the knowledge learned from the original task, fine-tuning method will eventually tend to new tasks. Adapter tuning (Houlsby et al., 2019) can be viewed as the hybrid of fine-tune and feature extraction. Unlike our model that makes no changes to the backbone model, the Adapter tuning method increases the original model size and introduces more parameters by inserting adapter modules to each layer.

3 Our CCFI Model

Given data stream $D = \{x_i, y_i\}_{i=1}^N$, the classification tasks in deep learning neural networks are equal to find the best parameter set Θ that can maximize the probability of the data $p(\mathcal{D}|\Theta)$. Namely, the classifier can make predictions $\hat{Y}$ that best reproduce the ground truth labels Y. Under the continual learning setting, new data $\mathcal{D}^n$ of additional classes will be added to the original data stream $\mathcal{D}^o$ in an incremental form. Our goal is to update the old parameters Θ^o (trained on original data stream $\mathcal{D}^o$) to the new parameters Θ^n, which can work well on both new data $\mathcal{D}^n$ and old data $\mathcal{D}^o$.

In this paper, the initial model is trained with the original data set $\mathcal{D}^o$, and will output the trained model Θ^o. In this training process, **Fisher Information Sampling** is conducted to select the most representative examples that can help to "remember" the parameters of the initial trained model. In the retraining process, the renewed model is learned based on **Dynamical Weight Consolidation**, and evaluated on the training set consisted of new classes and the old exemplars.

3.1 Fisher Information Sampling

The critical problem of exemplar set selection is: what are good examples that can "maintain" the performance of old tasks? The state-of-the-art method iCaRL (Rebuffi et al., 2017) selects examples close to mean feature representation, and CoNDA (Li et al., 2019) follows the same scheme to domain adaptation on text data. To utilize the advantage of the mean feature and avoid catastrophic forgetting, iCaRL chooses k-nearest neighbors algorithm as the classifier rather than deep neural networks, although the latter is proved to be a much better performer. Is there any exemplar selection method that can utilize the powerful deep learning models as the classifier, and at the same time, "remember"

the key information of old tasks?

Fisher information measures how much information that an observable random variable carries about the parameter. For a parameter θ in the network Θ trained by data $\mathcal{D}$, its Fisher information is defined as the variance of the gradient of the log-likelihood:

$$I(\theta) = Var\left(s(\theta)\right) \tag{1}$$
$$= E\left[\left(\frac{\partial}{\partial\theta}\log p(\mathcal{D}|\theta)\right)\left(\frac{\partial}{\partial\theta}\log p(\mathcal{D}|\theta)\right)^T\right].$$

Fisher information can be calculated directly, if the exact form of $\log p(\mathcal{D}|\theta)$ is known. However, the likelihood is often intractable. Instead, empirical Fisher information $\hat{I}(\theta)$ is computed through data $d_i \in \mathcal{D}$ drawn from $p(\mathcal{D}|\theta)$:

$$\hat{I}(\theta) = \frac{1}{N}\sum_{i=1}^N \left(\frac{\partial}{\partial\theta}\log p(d_i|\theta)\right)\left(\frac{\partial}{\partial\theta}\log p(d_i|\theta)\right)^T. \tag{2}$$

From another point of view, when $\log p(\mathcal{D}|\theta)$ is twice differentiable with respect to θ, Fisher information can be written as the second derivative of likelihood:

$$I(\theta) = -E\left[\frac{\partial^2}{\partial\theta^2}\log p(\mathcal{D}|\theta)\right]. \tag{3}$$

According to Equation 3, three equivalent indications can be made to a high value in Fisher information $I(\theta)$:
- a sharp peak of likelihood with respect to θ,
- θ can be easily inferred from data $\mathcal{D}$,
- data $\mathcal{D}$ can provide sufficient information about the correct value for parameter θ.

Jointly thinking about the calculation form introduced by empirical Fisher information (Equation 2) and the physical meaning of Fisher information revealed by the second derivative form (Equation 3), we can find a way to measure how much information each data d_i carries to the estimation of parameter θ, which we call as empirical Fisher information difference:

$$\Delta\hat{I}_i(\theta) = \left(\frac{\partial}{\partial\theta}\log p(d_i|\theta)\right)\left(\frac{\partial}{\partial\theta}\log p(d_i|\theta)\right)^T. \tag{4}$$

Instead of simple mean feature vectors used in previous work (Rebuffi et al., 2017; Li et al., 2019), we use the empirical Fisher information difference to select exemplar set. Specifically, CCFI model makes use of BERT (Devlin et al., 2019) for text

classification. The base BERT model is treated as feature extractor $\Phi : X \to H$, which takes input token sequences X, and outputs the hidden representations H. To predict the true label Y, a softmax classifier is added to the top of BERT:

$$p(\hat{Y}|X, \Theta) = \sigma(W^T \cdot H) = \sigma(W^T \cdot \Phi(X)), \quad (5)$$

where W is the task-specific parameter matrix for classification. The trained parameters Θ can therefore be split into the fixed feature extraction part Φ and variable weight parameter W. In continual learning setting, we denote $W^k \in R^{h \times k}$ as the most up-to-date weight matrix, where k is the number of classes that have been so far observed, and h is the size of the final hidden state H.

Remember that, for the classification task, the best parameters that can maximize the probability of the data $p(\mathcal{D}|\Theta)$ are also the parameters that make predictions $\hat{Y}$ closest to the ground truth label Y. Therefore, we can take Equation 5 into Equation 4, in order to get the practical computation of empirical Fisher information difference for data d_i on parameter θ. Since the parameters of feature extractor Φ are fixed, only empirical Fisher information difference of parameters in weight matrix $w_j \in W$ are calculated:

$$\Delta \hat{I}_i(w_j) = \left(\frac{\partial}{\partial w_j} \log \left[\sigma(W^T \cdot \Phi(x_i)) \right]^{(y_i)} \right)^2, \quad (6)$$

where the likelihood $p(d_i|\theta)$ is calculated via the log-probability value of the correct label y_i of input x_i. And the total empirical Fisher information difference data d_i carrying is the sum over all $w_j \in W$:

$$\Delta \hat{I}_i = \sum_{j=1}^{h \times k} \Delta \hat{I}_i(w_j). \quad (7)$$

Algorithm 1 describes the exemplar selecting process. Within each class k, the samples top ranked by empirical Fisher information difference are selected as exemplars, till the targeted sample rate (e.g., 1%) is met.

3.2 Dynamical Weight Consolidation

The main goal of retraining process is: how to achieve good performance for both new and old tasks? EWC (Kirkpatrick et al., 2017) is proved to be a good performer that can balance the performance of old and new task. However, rather

---

Algorithm 1: Construction of exemplar set

Input: original data stream $\mathcal{D}^o$
Input: trained neural network $\Theta^o = \{\Phi, W^o\}$
Input: sample rate r
1 **for** *each data d_i* **do**
2   **for** *each parameter $w_j \in W^o$* **do**
3   calculate $\Delta \hat{I}_i(w_j)$ using Equation 6
4   calculate $\Delta \hat{I}_i$ using Equation 7
5 **for** *each class k* **do**
6   rank the samples d_i by $\Delta \hat{I}_i$
7   select the top $|\mathcal{D}_k| \times r$ examples as E_k
Output: exemplar set $E \leftarrow \{E_1, ..., E_k\}$

---

than incremental learning problem studied in this paper, EWC is designed for the tasks with same class number but different in data distributions. Furthermore, EWC requires careful hyperparameter setting, which is unrealistic to be conducted by inexperienced users. In this section, we introduce a scheme named Dynamical Weight Consolidation, which can avoid the requirement of such hyperparameter setting. Also, this scheme is demonstrated to perform more stably than traditional EWC in the experimental part.

Specifically, our loss function during the retraining process can be viewed the sum of two parts: loss ℓ_n calculated by the correct class indicators of new data and loss ℓ_e to reproduce the performance of old model:

$$\ell_n = - \sum_{y \in Y^n} y \log \hat{y}, \quad (8a)$$

$$\ell_e = - \sum_{y \in E} y \log \hat{y} + \frac{\lambda}{2} \sum_{j=1}^{h \times (k_o + k_n)} \hat{I}(w_j)(\hat{w}_j^n - w_j^o)^2. \quad (8b)$$

The loss function ℓ_e can be further divided into two parts: the cross entropy of exemplar set, and the consolidation loss caused by modifying parameters with high Fisher information. In traditional EWC model, the weight λ that balances cross entropy and consolidation loss is a fixed value. In our CCFI model, λ is updated dynamically based on current values of cross entropy and consolidation loss:

$$\lambda = \left\lfloor \lg \frac{- \sum_{y \in Y^n} y \log \hat{y}}{\sum_{j=1}^{h \times (k_o + k_n)} \hat{I}(w_j)(\hat{w}_j^n - w_j^o)^2} \right\rfloor. \quad (9)$$

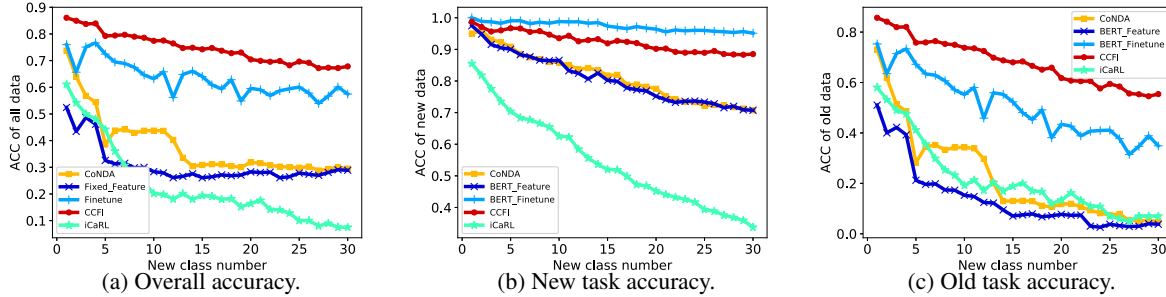Note that, the $\hat{I}(w_j)$ is the element in the updated

Figure 1: Performance of class-incremental learning on the 73-domain dataset. The observable ratio of old data is kept as 1%, 43 out of 73 domains are used for initial training, and then 30 domains are added one-by-one for retraining.

parameter information table T^n. The details can be found in the section 3.3.2.

3.3 Overall Process

This part describes the overall process of the CCFI model. First we list the outputs of the old tasks, then we introduce the detailed implements of retraining.

3.3.1 Initial training

After training the model with old data (k_o classes), the outputs of the old task include:1) trained model Θ^o; 2) exemplars E of old data, and 3) parameter information table T^o. Each element in the parameter information table T^o is the empirical Fisher information $\hat{I}(w_j^o)$ of the old task, which can be computed through Equation 2 during the initial training process.

3.3.2 Retraining

The retraining process can be described as follows:
1. **Load freeze feature extractor**: The feature extractor Φ is kept unchanged, which means the BERT encoder with transformer blocks and self-attention heads are freezed.
2. **Update variable weight matrix**: To adopt the new classes data X^n, the original variable weight matrix W^{k_o} is extended to $W^{k_o+k_n} \in R^{h\times(k_o+k_n)}$, where the first k_o columns are kept the same with the original model and the new k_n columns are initialized with random numbers.
3. **Update parameter information table**: Similar to variable weight matrix, parameter information table T^o is a matrix with dimension $h \times k_o$. To adopt the new data, it is extended to the new matrix T^n with dimension $h \times (k_o + k_n)$, where the first k_o columns are same with T^o and the new k_n columns are initialized with zero. In this way, the new model can freely update the the new k_n columns to lower classification loss,

but will receive penalty when modifying important parameters in the original k_o columns.

4 Experiment

In this section, the CCFI model is first compared with the state-of-the-art methods under a continual setting. And further evaluations are conducted to examine the effectiveness of the individual components within CCFI model.

4.1 Experiment Settings

Datasets. We evaluated our proposed CCFI and comparison methods on public available 150-class dataset (Larson et al., 2019) and real-world (even product) 73-domain dataset The details of datasets can be found in Appendix A.1.

Baselines. iCaRL (Rebuffi et al., 2017) and CoNDA (Li et al., 2019), are the closest continual learning approaches to CCFI, which are designed for the scenario with access to old data. We also add fine-tune and the fixed feature method as baselines. To make fair comparisons, CCFI and all the baselines use the same BERT backbone (Devlin et al., 2019), and observe the same amount of old data in all learning tasks. The implementation details can be found in Appendix A.2.

In the main body of experiments, we report the results with the framework consisted of BERT backbone and one-layer linear classifier. We also conducted experiments with a multiple-layer classifier, which can be found in Appendix A.3.

4.2 Quantitative Evaluation

Two key factors play in the performance of continual learning: 1) the number of new classes for retraining, and 2) the amount of old observable data. In this section, we first validate our model through a class-incremental learning task, by keep-
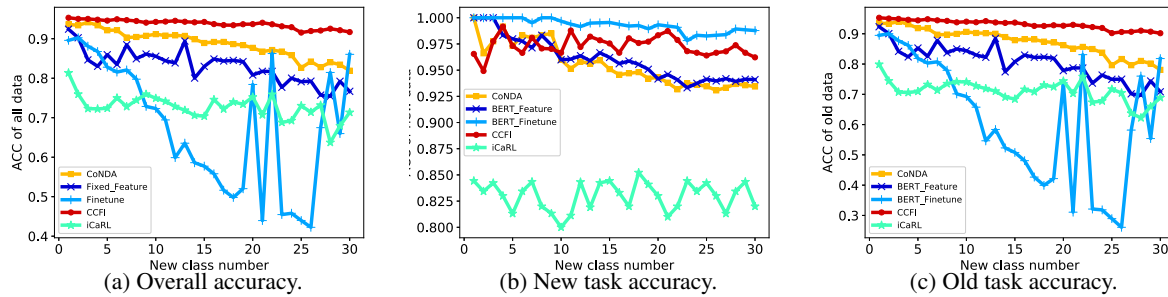
Figure 2: Performance of class-incremental learning on the 150-class dataset. The observable ratio of old data is kept as 10%, 90 out of 150 classes are used for initial training, and then 30 domains are added as additional classes.

ing the amount of old observable data fixed and changing the number of new classes. We also study the effects of different exemplars by keeping the number of new classes unchanged but varying old observable data.

Class-incremental Learning. In this part, we conduct the class-incremental learning evaluation on both 150-class and 73-domain dataset. Class-incremental learning can be viewed as the benchmark protocol for continual learning with access to old data (Rebuffi et al., 2017; Li et al., 2019). Specifically, after the initial training, new domains will be added in random order. After adding each batch of new data, the results are evaluated on the current data set, considering all classes have been trained so far.

Figure 1 and Figure 2 show the performance of class-incremental learning on 73-domain dataset and 150-class dataset. CCFI outperforms other methods in all tasks on both datasets. Specifically, several observations can be made as follows.

- **Overall performance**. Under the same new class number, CCFI always achieves the best overall accuracy among all methods. And this performance gap is enlarged with more new classes added for retraining.
- **Performance fluctuations**. Fine-tune method is unstable in performance. It is the second performer on the 73-domain dataset. However, it quickly drops to almost zero and displays fluctuations on the 150-class dataset, even if the experiment conducted on the 150-class dataset is set with a higher observable ratio of old data.
- **Accuracy stage**. Both the fixed feature method and CoNDA display the pattern of "performance stage" with more new classes added, and CoNDA enjoys a "larger" stage than the fixed feature method. For example, as shown in Figure 1a, CoNDA maintains stable performance with

5 to 12 newly added classes varying and then suddenly drops.

- **Predictable performance**. Both CCFI and iCaRL display linear patterns in overall performance. It means the possibility to predict and estimate class-incremental learning performance, which is a preferable feature in many applications. But iCaRL starts at a lower accuracy and drops much faster than CCFI, probably because it discards the neural network and tunes to the simple k-nearest neighbors algorithm as the final classifier. This phenomenon also confirms that CCFI can enjoy the excellent performance of neural network classifiers and overcome its drawback of catastrophic forgetting.

4.2.1 Different Exemplar Size

To provide insight into the working mechanism of models capable of continual learning, we conduct experiments by varying the exemplar set's size with the number of new classes fixed. Figure 3 shows the model performance under the effect of different exemplar sizes by changing the observable ratios of old data.

- **Overall pattern**. CCFI continues to beat baselines with obvious advantages in performance. Especially, CCFI can achieve high accuracy with a minimal amount (e.g.,1%) of old data, although all methods can obtain performance improvement by increasing the ratio of old observable data. A dramatic performance gain can also be observed from all models when the observable ratio of old data increases from zero to non-zero values. This phenomenon further confirms that our experimental setting with limited access to old data is practically useful.
- **Consistent improvement**. CCFI, CoNDA, and iCarL obtain consistent improvements when increasing the ratio of old observable data. How-
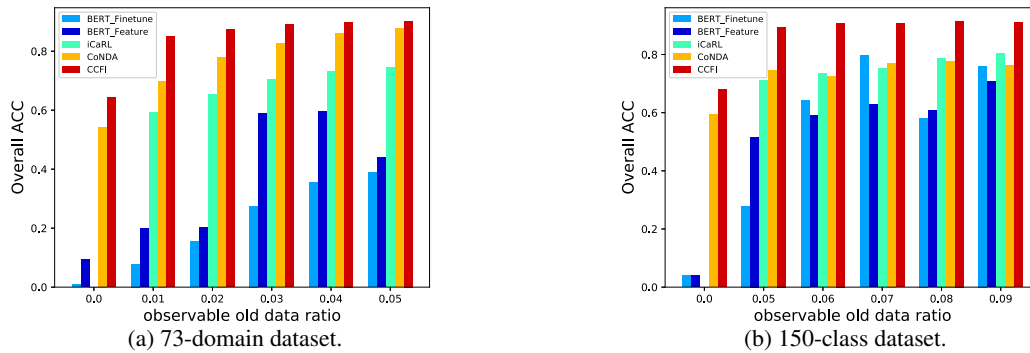
| (a) 73-domain dataset. | (b) 150-class dataset. |

Figure 3: Effect of Differential Exemplar on the 73-domain dataset and 150-class dataset. For the 73-domain dataset, the observable ratio of old data is changed from zero to 5%, while the number of new classes is fixed: 68 domains are used for initial training, and five domains are used as the additional new classes. For the 150-class dataset, the observable ratio of old data is changed from zero to 10%, while the number of new classes is fixed: 90 domains are used for initial training, and 30 domains are used as the additional new classes.

ever, the fixed feature method doesn't get apparent benefits with more old data. This phenomenon indicates more observations of old data are not the guarantee for better performance. And it further confirms the necessity of developing continual learning methods that can effectively utilize the information learned from exemplars.

4.3 Ablation Study

Our proposed CCFI outperforms all the state-of-the-art methods. To provide further insights into its working mechanism, additional experiments are conducted to examine individual aspects of CCFI.

4.3.1 Dataset and Experimental Setting

In order to avoid the occasionalities introduced by data and model complexity, components are examined on a synthetic data set by simple neural networks with fixed weight initialization.

Specifically, we generate a synthetic dataset of 10 completely separable classes, and each class includes 1,000 examples. As the setting for continual learning, we use six classes for initial training, and four classes as additional new classes for retraining. The neural network used in this section is a simple network with two fully-connected layers. The first layer is served as a feature extractor, which is fixed after the initial training. The second layer is used as a classifier that can be tuned during retrain. To ensure other parts won't affect the component to be validated, the neural networks are initialized with the same weight matrix generated by a fixed random seed. With these settings, the results can best reflect the true contribution of components.

4.3.2 Dynamical Weight Consolidation

First, we analyze the effectiveness of the dynamical weight consolidation component. Figure 4 plots the consolidation loss (second part in Equation 8b) of model using traditional fixed weight and our proposed dynamical weight consolidation. Several observations can be made as follows.

- **Fixed weight with big value**. When the weight (λ in Equation 8b) is set by a big value (e.g., 10^{25} in Figure 4a), the consolidation loss is hard to converge and suffers from fluctuations.
- **Fixed weight with small value**. Oppositely, if the weight is initialized with a relatively small value (e.g., 10^2 in Figure 4b), the consolidation loss is too small to be effective. In fact, as can be observed from Figure 4b, under the small weight setting, the consolidation loss even experiences an increase first before it slowly decreases. The increase in consolidation indicates that the neural network tends to sacrifice consolidation loss to lower the overall loss. Furthermore, this phenomenon happens when the new model modifies the important parameters learned by the original model, which are supposed to be kept with the least changes for the continual learning purpose.
- **Dynamical weight consolidation**. In contrast to the unstable performance of the traditional method, as shown in Figure 4c, consolidation loss converges fast and stable by using our proposed dynamical weight consolidation.

4.3.3 Fisher Information Sampling

The second set of experiments validate whether Fisher information sampling is indeed beneficial to the overall performance, compared with using
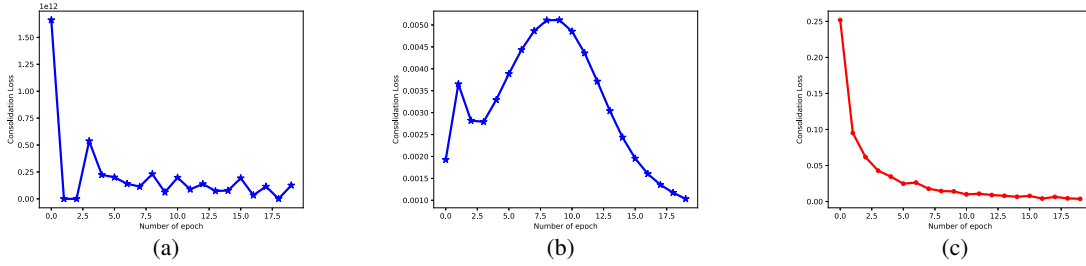
Figure 4: Consolidation loss under different weight strategies. Figure 4a: Fixed weight strategy initiated by a big value 10^{25}. Figure 4b: Fixed weight strategy initiated by a small value 100. Figure 4c: Dynamic weight strategy using weight calculated by Equation 9.

(a) Sampling rate 0.5% (b) Sampling rate 1%

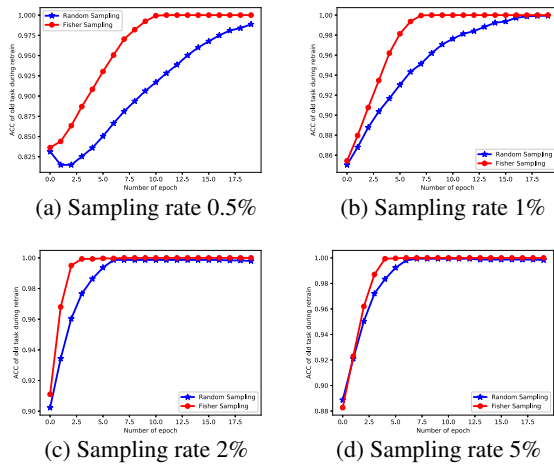(c) Sampling rate 2% (d) Sampling rate 5%

Figure 5: Accuracy of the old task during retraining. Blue lines denote the accuracy using exemplars randomly selected, while red lines reflect the performance of exemplars generated by Fisher sampling.

randomly selected examples.

To examine how much improvement can be obtained by Fisher sampling alone, we remove the weight consolidation component in this section. Thus the results reported here are outputs of the simple two-layer model by using exemplars during retraining. From another point of view, these results show the amount of information the exemplars carrying from the original model.

Figure 5 plots the accuracy of the old task during the retraining process. Although the network is retrained with only a small set of old data, the accuracy is computed over all old data to fully examine the quality of exemplars. Since the classes in synthetic data are fully separable, the accuracy will be 100% eventually. Therefore, the quality of exemplars is demonstrated by the converging speed. A faster converge speed provided by an exemplar set is of great significance in three aspects:

- **Better computational efficiency**. With the same amount of old data for retraining, the most obvious benefit indicated by the faster converge speed is, the better computational efficiency since the model requires less retraining time.
- **Less "damage" to original model**. A faster converge speed indicates less "damage" to the original model. All weight consolidation schemes act like "buffers" for the old parameter. With these schemes, old parameters will slow down their changes when new tasks come. To best cope with the consolidation schemes, good exemplars should achieve comparable good performance with fewer retraining epochs, since more retraining epochs mean that the new model has modified more parameters from the original network.
- **More information of original dataset**. As mentioned above, under the synthetic data is fully separable, the accuracy will be 100% eventually. In this case, a faster speed can be "converted" to more information, as experiments with more data always require fewer epochs to reach the states of convergence. For example, as shown in Figure 5, much more epochs are needed under sampling rate 0.5% than that of 1%.

Figure 5 shows, exemplars generated by Fisher sampling can achieve much faster converge speed than randomly selected exemplars, which proves Fisher sampling alone can contribute contribution effectively to the overall performance.

5 Conclusion

This paper proposes a hyperparameter-free model called CCFI for continuous domain classification. CCFI can record information of old models via exemplars selected by Fisher information sampling, and conduct efficient retraining through dynamical weight consolidation. The comparison against the existing models reveals CCFI is the best performer under various experimental environments, without additional efforts in hyperparameter searching.

classifier	Finetune		Feature Extraction		CoNDA		CCFI	
	Initial training	Retrain	Initial training	Retrain	Initial training	Retrain	Initial training	Retrain
One-layer	0.9671	0.7232	0.9671	0.8556	0.9566	0.9116	0.9614	0.9426
Two-layer	0.9592	0.2387	0.9592	0.6495	0.9522	0.9366	0.9541	0.9408

Table 1: Effect of classifier layer number on the 150-class dataset. 90 out of 150 classes are used for initial training, and then additional data of 10 domains plus 10% of old data (90-class) are used for retraining.

classifier	Finetune		Feature Extraction		CoNDA		CCFI	
	Initial training	Retrain	Initial training	Retrain	Initial training	Retrain	Initial training	Retrain
One-layer	0.9561	0.6322	0.9561	0.2839	0.9349	0.4375	0.9587	0.7378
Two-layer	0.9422	0.0751	0.9422	0.1193	0.9314	0.6868	0.9472	0.7134

Table 2: Effect of classifier layer number on the 73-domain dataset. 43 out of 73 classes are used for initial training, and then additional data of 10 domains plus 1% of old data (43-class) are used for retraining.

A   Appendix

A.1   Dataset Statistics

The general statistics of the 150-class dataset [1] and 73-domain dataset are listed below.

- 150-class dataset: balanced dataset with 150 intents that can be grouped into 10 general domains. Each intent has 100 training queries, 20 validation, and 30 testing queries.
- 73-domain dataset: imbalanced dataset with 73 domains. Each domain contains 512 examples on average. However, this dataset is highly imbalanced that the largest domain includes 1,771 examples, while the smallest domain only has 27 examples.

In both datasets, we split examples of each class into 90% for training, 5% for validation, and 5% for testing. All classification accuracy results are evaluated on the test set.

A.2   Implement Details

Specific settings. In our implement of CoNDA, we pick up hyperprameter $\Delta_{pos} = 0.5$ and $\Delta_{neg} = 0.3$. The fixed-feature method freezes 12 layers of BERT after the initial training. Only the parameters in the new classifier layer are allowed for tuning, which in a way provides the function of continual learning. Fine-tune method can modify parameters in all 12 layers of BERT, which can be viewed as the network with little prevention of catastrophic forgetting.

General settings. Adam optimizer is used in all learning processes, and the learning rate is always set to be 0.00001. All runs are trained on 4 V100 GPUs with a batch size of 32. Our exam-

ple code can be found at: `https://github.com/tinghua-code/CCFI`

A.3   Multi-layer Classifier Results

To examine the effect of classifier layer number (amount of retrainable parameters), we run experiments under two frameworks. The first framework is the same as the one used in the main experimental part, which consists of a 12-layer BERT feature extractor and a one-layer linear classifier. The second framework keeps the BERT feature extractor unchanged and adds one more layer to the classifier. The results are listed in Table 1 and 2, and several observations can be made as follows.

- CCFI still remains the best performer. Our proposed CCFI model produces good performance regardless of the number of layers in the classifier. This phenomenon further confirms its effectiveness and stability.
- CoNDA is the second-best performer in both frameworks. Notably, the retraining performance of CoNDA increases when we increase the number of layers.
- BERT finetune and feature extraction method become worse when increasing the number of layers. These two baselines are sensitive to the structure of the classifier, which indicates the superficial variations of pre-trained models are not enough for continual learning.
- One-layer classifier works well enough with BERT. As can be seen from Table 1 and 2, the initial training results of all methods degrade when increasing the number of classifier layers. Therefore, we report the results based on a one-layer linear classifier in the main body of the paper.

---

[1]https://github.com/clinc/oos-eval

References

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.

Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. 2014. Decaf: A deep convolutional activation feature for generic visual recognition. In *International conference on machine learning*, pages 647–655.

Robert M French. 1999. Catastrophic forgetting in connectionist networks. volume 3, pages 128–135. Elsevier.

Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, pages 2790–2799.

Young-Bum Kim, Dongchan Kim, Joo-Kyung Kim, and Ruhi Sarikaya. 2018a. A scalable neural shortlisting-reranking approach for large-scale domain classification in natural language understanding. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 3 (Industry Papers)*, pages 16–24.

Young-Bum Kim, Dongchan Kim, Anjishnu Kumar, and Ruhi Sarikaya. 2018b. Efficient large-scale neural domain classification with personalized attention. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2214–2224.

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. volume 114, pages 3521–3526. National Acad Sciences.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

Anjishnu Kumar, Arpit Gupta, Julian Chan, Sam Tucker, Bjorn Hoffmeister, Markus Dreyer, Stanislav Peshterliev, Ankur Gandhe, Denis

Filiminov, Ariya Rastrow, et al. 2017. Just ask: building an architecture for extensible self-service spoken language understanding.

Stefan Larson, Anish Mahendran, Joseph J. Peper, Christopher Clarke, Andrew Lee, Parker Hill, Jonathan K. Kummerfeld, Kevin Leach, Michael A. Laurenzano, Lingjia Tang, and Jason Mars. 2019. An evaluation dataset for intent classification and out-of-scope prediction. 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 1311–1316, Hong Kong, China. Association for Computational Linguistics.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. volume 521, pages 436–444. Nature Publishing Group.

Han Li, Jihwan Lee, Sidharth Mudgal, Ruhi Sarikaya, and Young-Bum Kim. 2019. Continuous learning for large-scale personalized domain classification. In *Proceedings of NAACL-HLT*, pages 3784–3794.

Zhizhong Li and Derek W Hoiem. 2016. Learning without forgetting. In *Computer Vision-14th European Conference, ECCV 2016, Proceedings*, pages 614–629. Springer-Verlag.

David Lopez-Paz and Marc'Aurelio Ranzato. 2017. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems*, pages 6467–6476.

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.

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.

Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. 2014. Cnn features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 806–813.

Gokhan Tur and Renato De Mori. 2011. *Spoken language understanding: Systems for extracting semantic information from speech*. John Wiley & Sons.

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.