PARTIAL-LABEL CONTINUAL LEARNING

Yuxiang Zheng & Tongliang Liu*
University of Sydney

Lei Feng†
Chongqing University

ABSTRACT

Continual learning (CL), a setting that continuously learns from a never-ending stream of data, is a challenging problem that allows a model to constantly update in a class incremental, or domain incremental setting. In the past few years, many different state-of-the-art methods and tricks with different settings and different assumptions have been introduced to address this problem. Despite each method having pushed the accuracy of CL from different perspectives, the performances are still unacceptable in practice. In this work, we prospectively discuss the possibilities of extending CL in the weak-supervised learning setting. Partial label learning (PLL) is a promising weak-supervised learning task that considers using data from a set of ambiguous candidate label sets rather than fully-labelled clean data. We propose a partial label-based CL method that demonstrates the scalability of CL on the partial label and achieves similar performance to the supervised counterpart under a benchmark.

1 INTRODUCTION

Modern artificial intelligence is capable of performing at the human level or even better on specific tasks in many areas.(Krizhevsky et al., 2012) However, the learning process is still inconsistent with that of humans, which can incrementally learn from different tasks in different domains and use previous knowledge to help learn new knowledge. The PCL aims to obtain better machine intelligence from a never-ending stream of non-iid partial label data. One of the key challenges is catastrophic forgetting (Goodfellow et al., 2013), i.e. potential forgetting of old knowledge when learning a new task, which is common in both machines and humans. Several state-of-the-art approaches have been addressed to attempt to solve this problem from a different perspective, including three major ideas, which are regularization-based methods (Kirkpatrick et al., 2016; Li & Hoiem, 2016), memory-based methods(Rebuffi et al., 2016; Chaudhry et al., 2019b; Aljundi et al., 2019a; Prabhu et al., 2020) and parameter-isolation-based methods(Lee et al., 2020).

Despite the promise of CL, the majority of neural networks currently use a supervised learning setting. A large amount of clean annotated data can be time-consuming and expensive, especially for some data that requires expert annotation, such as medical images or protein structure images. Indeed, partial labels are widespread and cheap in reality and can be easily obtained through automatic image annotation(Chen et al., 2017) or data mining(Luo & Orabona, 2010). The most important problem experienced in the training process of PLL is selecting the ground-truth label from the candidate set; a wrong selection may lead to severe ambiguity thus exacerbating the catastrophic forgetting of PCL. For example, In Figure 1 people may incorrectly mistake the Maine Coon for a Norwegian Forest cat or a Siberian cat.

In this work, we assume that the estimator can learn incrementally from a never-ending stream of partial label data. We consider using a memory-based method, as has proved to be effective and trivial to implement in many benchmarks. This method is based on the human learning process inspired by reviewing old tasks following each new task learned(Rolnick et al., 2018). We store part of the data from the previous task in a memory buffer and review it by taking the most interfering samples after training the new task, which implies that we choose a subset of the most forgotten samples from the buffer each time(Aljundi et al., 2019a).

*supervisor
†co-supervisor

A Maine Coon image x_i with
$Y_i = \{\underline{\text{Maine Coon}}, \text{Norwegian forest}, \text{Siberian}\}$

Figure 1: An input image x_i with a candidate set with ground-truth label “Maine Coon".

Inspired by contrastive learning, we use the PiCO strategy of label disambiguation based on contrastive learning for representation(Wang et al., 2022; He et al., 2019). Unlike traditional PLL methods, it invokes contrastive learning for representation, alleviating the representation-disambiguation dilemma: uncertainty in sample labels can severely affect the representation of the neural network, which in turn affects the label disambiguation(Wang et al., 2022).

2 BACKGROUND

In the setting of partial-label continual learning (PCL) task we give the following formal definitions. We consider an image classification task in which all samples come from a never-ending stream of non-iid data, in line with the recent CL literature (Aljundi et al., 2019a; Lesort et al., 2019; Mai et al., 2021). We define the data as a set of unknown distributions $\mathbb{D} = \{\mathcal{D}_1, \cdots, \mathcal{D}_N\}$ over $\mathcal{X} \times \mathcal{Y}$, where $\mathcal{X}$ is the sample sapce and $\mathcal{Y} = \{1, 2, \cdots, C\}$ is the target label space. For each task, we consider the dataset $\mathcal{D} = \{(x_i, Y_i)\}_{i=1}^n$, where each sample contains an image $x_i \in \mathcal{X}$ and a candidate label set $Y_i \subset \mathcal{Y}$. As in traditional supervised learning, PCL aims to learn an estimator to classify images. However, the difference is that the partial label of PLL introduces ambiguity into the training process. A common assumption of PLL is that the candidate label set Y_i contains the ground-truth label y_i, i.e. $y_i \in Y_i$, which is invisible to the estimator. The PCL algorithm expects to train a new learner at time t with the help of new data (x_t^i, Y_t^i) and the learner at time $t-1$, which (x_t^i, Y_t^i) is a mini-batch that received in the distribution D_i at time t, and we define the algorithm in the following structure:

$$A_t^{PCL} = \langle f_{t-1}, (x_t, Y_t), M_{t-1} \rangle \mapsto \langle f_t, M_t \rangle , \tag{1}$$

where M_t is a memory buffer for storing a subset of samples of previous tasks. One of the key challenge of PCL is label disambiguation, which is selecting the ground-truth label from the candidate set. We consider giving each image a normalized pseudo label vector $\boldsymbol{s}_i \in [0,1]^C$, where each entry represents the probability that the image is predicted to be of a particular class and all entries sum to 1. The pseudo labels will be constantly updated throughout the training process, and ideally, the probability of eventual pseudo-labels might be concentrated on the ground-truth label. We train a classifier $f : \mathcal{X} \to [0,1]^C$ using cross-entropy loss, which has the following loss function:

$$\mathcal{L}_{cls}(f; x_i, Y_i) = \sum_{j=1}^{C} -s_{i,j} \log(f^j(x_i)) \quad \text{s.t.} \quad \sum_{j \in Y_i} s_{i,j} = 1 \text{ and } s_{i,j} = 0, \forall j \notin Y_i, \tag{2}$$

where j denotes the index of the class and $s_{i,j}$ denotes the probability that the pseudo label predicted sample is of class j. f is the softmax output of the classifier and f^j is its j_{th} entry.

3 METHOD

In this section, we demonstrate a specific method for implementing PCL. We consider the use of a memory-based method to complete the continual learning process(Aljundi et al., 2019a). An

advanced method using contrastive label disambiguation will be used as training for the classifier(Wang et al., 2022). We will describe our method from the perspective of PLL and CL respectively.

3.1 PICO

The ambiguity of the partial labels creates many obstacles to network representation. Inspired by unsupervised learning, PiCO employs a framework of contrastive learning for representation(Wang et al., 2022). A combination of contrastive loss and classification loss from *Equation*2 was used to jointly update the neural network. The MoCo proposed by He et al. (2019) is comparable to supervised learning in many benchmarks and has largely driven the development of unsupervised learning. PiCO uses a similar framework to MoCo and achieves SOTA results in the domain of partial label learning. One of the main challenges is the construction of positive set selection.

3.1.1 PICO: EMBEDDED LOSS

The framework of PiCO and MoCo(He et al., 2019) is similar in that it aims to provide similar representations for the same class of samples by introducing two identical or approximate networks. The approximate network ensures consistency in the input space, and we divide the two networks into a query network $q'(\cdot)$ and a key network $k'(\cdot)$. When inputting a sample x_i, we perform two different image augmentations on it to obtain a query view $\text{Aug}_q(x_i)$ and a key view $\text{Aug}_k(x_i)$. Then fed into the corresponding network we obtain a set of L_2-normalized embedding vectors, i.e. $\boldsymbol{q} = q'(\text{Aug}_q(x_i))$ and $\boldsymbol{k} = k'(\text{Aug}_k(x_i))$. PiCO uses the queue data structure to construct the embedding pool, $A = B_q \cup B_k \cup \text{queue}$. Where B_q refers to the query views in the current mini-batch and B_k refers to the key views, one of the main benefits of using a queue is that it ensures that the first dequeued samples are the most inconsistent with the feature space in the embedding pool. Contrastive learning uses a dictionary query method where for each input x we compare its query embedding with each entry in the embedding pool and use an improved InfoNCE-based supervised contrastive loss(Khosla et al., 2020; van den Oord et al., 2018):

$$\mathcal{L}_{\text{cont}}(q; \boldsymbol{x}, \tau, A) = -\frac{1}{|P(\boldsymbol{x})|} \sum_{\boldsymbol{k}_+ \in P(\boldsymbol{x})} \log \frac{\exp(\boldsymbol{q}^\top \boldsymbol{x}_+ / \tau)}{\sum_{\boldsymbol{k}' \in A(\boldsymbol{x})} \exp(\boldsymbol{q}^\top \boldsymbol{k}' / \tau)}, \tag{3}$$

where $P(\boldsymbol{x})$ is a positive set, $A(\boldsymbol{x}) = A \setminus \{\boldsymbol{q}\}$ and τ is a temperature parameter ≥ 0. Finally, we share the convolutional layer with the query network and add an extra fully-connected mapping to the classifier for prediction.

3.1.2 PICO: POSITIVE SET SELECTION

In contrastive learning the construction of a positive set is essential. The accuracy of the positive set greatly affects the representational effect on contrastive learning. Unlike MoCo and many other contrastive learning algorithms that use various pretext tasks, PiCO constructs positive sets $P(\boldsymbol{x})$ directly using the predictions of the classifier(He et al., 2019; Ye et al., 2019). Given a sample x we use the classifier to predict the class of the sample, $\tilde{y} = \arg\max_{j \in Y} f^j(\text{Aug}_q(\boldsymbol{x}))$, and construct the positive set:

$$P(\boldsymbol{x}) = \{k' \mid k' \in A(\boldsymbol{x}), \tilde{y}' = \tilde{y}\}, \tag{4}$$

where k' is the sample in the embedding pool and $\tilde{y}'$ is the prediction of the classifier for k'. We restrict the prediction of sample $\boldsymbol{x}$ to the candidate set to ensure compliance with the assumption. This strategy is simple and easy to implement, but our experiments have shown its effectiveness. Some more sophisticated methods using the threshold have also been widely used in the semi-supervised domain, maximise $f^j(\text{Aug}_q(\boldsymbol{x})) \leq \delta$ ($\delta = 0.95$)(Sohn et al., 2022). With a good positive set, we can then update the query network using a combination of classification loss and contrastive loss,

$$\mathcal{L} = \mathcal{L}_{\text{cls}} + \lambda \mathcal{L}_{\text{cont}}, \tag{5}$$

where $\lambda \in [0, 1]$ is a loss weight. We update the key network slowly using a momentum method, which ensures that the features in the embedding pool are as consistent as possible. $\theta_k = m \cdot \theta_{k-1} + (1 - m) \cdot \theta_q$, usually we set the momentum parameter large enough, e.g. $m = 0.9999 \mid m \in [0, 1)$, to ensure that the query network does not affect the key network too

much. Misclassification of the positive set can significantly affect the representation of contrastive learning, and PiCO uses a prototype-based disambiguation strategy that can effectively suppress the probability of false positive samples.(Wang et al., 2022).

3.1.3 PiCO: PROTOTYPE-BASED LABEL DISAMBIGUATION

Contrastive learning aims to achieve a similar effect to clustering by constructing the correct positive sets which bring samples of the same class closer together in feature space and maximise the distance between different classes. To this end, PiCO proposes a novel prototype-based method that maintains a prototype embedding vector μ_c for each class $c \in \{1, 2, \cdots, C\}$. Where each prototype can represent the embedding vector of a set of samples, i.e. the centroids in the clustering. If an accurate prototype is available, the naive idea is to directly select the closest prototype embedding output. Indeed, the prototype embedding vector has limited information at the beginning of training. PiCO uses a moving-average strategy to gradually update the pseudo label vector and the prototype vector.

PSEUDO LABEL UPDATING

A moving-average mechanism is used to update the pseudo-labels. At the beginning of training, we initialise the prototype embedding vector as a zero vector $\mu_c = [0]^C$ and the pseudo-labels as a uniform vector $s_j = \frac{1}{|Y|}\mathbb{I}(j \in Y)$, where we restrict j to the candidate set to ensure that samples outside the candidate set do not interfere with the prediction. We update the pseudo-label using the following strategy:

$$\boldsymbol{s} = \phi \boldsymbol{s} + (1-\phi)z, \quad z_c = \begin{cases} 1 & \text{if } c = \arg\max_{j \in Y} \boldsymbol{q}^\top \boldsymbol{\mu}_j \\ 0 & \text{else} \end{cases}, \tag{6}$$

where $\phi \in (0, 1)$ is a weight parameter, which controls the rate of updating of pseudo-labels. $\boldsymbol{\mu}_j$ refers to the prototype embedding vector of class j, and $\boldsymbol{q}^\top \boldsymbol{\mu}_j$ refers to the projection distances between the query view and the prototype. $\boldsymbol{s}$ is intended to be updated in the direction of the closest prototype, and in general, we use a small ϕ for slow updating since intuitively the prototype embedding at the initial stage is not reliable.

PROTOTYPE UPDATING

The most straightforward method is that for each epoch we compute centroids for each class of samples, but this leads to an extra computational toll that might be unacceptable. In addition, the accuracy of the sample prediction can severely affect the update of the prototype. For this reason, PiCO proposes to use the moving-average strategy again to update the prototype embedding vector.:

$$\boldsymbol{\mu}_c = \text{Normalize}(\gamma \boldsymbol{\mu}_c + (1-\gamma)\boldsymbol{q}), \quad \text{if } c = \arg\max_{j \in Y} f^j(\text{Aug}_q(\boldsymbol{x})), \tag{7}$$

where γ is a momentum parameter. When a sample $\boldsymbol{x}$ is predicted as category c, we update the prototype embedding vector along the direction of the query view of $\boldsymbol{x}$. The use of normalisation is intended to be consistent with the embedding pool for contrastive learning.

The subtlety of PiCO lies in the fact that contrastive learning and prototype-based label disambiguation strategy can mutually work together. The use of contrastive learning brings a clustering effect to the feature space, which is effective in helping to obtain accurate prototype embeddings. More accurate prototypes can help the classifier to make better predictions. An accurate prediction can in turn contribute to more accurate positive sets for the contrastive learning module, which can help contrastive learning to obtain a more accurate representation.

3.2 MIR

After we have solved the training of the partial label learning task, we need to consider how PCL can avoid catastrophic forgetting in the presence of class increment. In the PCL setting, we obtain a set of data $(\boldsymbol{x}_t^i, Y_t^i)$ at time t from an never-ending stream of data in a new domain $\mathcal{D}_t$. $\mathcal{D}_t$ can be a different non-iid domain from $\mathcal{D}_{t-1}$ or it can be identical. We aim to train a classifier f with parameters θ which minimizes the loss $\mathcal{L}$ of the current samples and wishes to preserve the loss

of past samples as unaffected as possible. Many studies have been conducted in the last few years to demonstrate that repeated training of partial samples from past tasks can prevent catastrophic forgetting to some extent(Chaudhry et al., 2019b; Aljundi et al., 2019b; Lopez-Paz & Ranzato, 2017; Chaudhry et al., 2019a). An intuitive idea is that we can store some of the samples from the past and then randomly select a portion to retrain each time we train a new task. MIR selects the part of the memory buffer that has maximum interference with the loss for retraining(Aljundi et al., 2019b). A key idea of this criterion is that some past samples might have almost no effect on the model. An intuitive understanding is that, like humans, machines need to review the most forgotten knowledge.

We consider a classical experience replay (ER) algorithm, which has been shown to be effective in many studies(Chaudhry et al., 2019a). We first construct a finite-size memory buffer $\mathcal{M}$. When we finish training task t, we randomly draw a fixed number of samples into the buffer. Some other algorithms also provide strategies for adding samples into the buffer, such as weighting the samples(Aljundi et al., 2019b; Sperl et al., 2016). When we receive a mini-batch of $\boldsymbol{x}_t^i$, the objective is to obtain the least loss parameter $\min_\theta \mathcal{L}(f_\theta(\boldsymbol{x}_t^i), Y_t^i)$. For this purpose, we first calculate the possible parameters under the current batch, $\theta^v = \theta - \alpha\nabla\mathcal{L}(f_\theta(\boldsymbol{x}_t^i), Y_t^i)$, where α is the learning rate. We then select the top-k samples of maximum interference, $\boldsymbol{x} \in \mathcal{M}$, using the following criterion:

$$s_{MI-1}(\boldsymbol{x}) = \mathcal{L}(f_{\theta^v}(\boldsymbol{x}), Y) - \mathcal{L}(f_\theta(\boldsymbol{x}), Y). \tag{8}$$

In addition to the maximum interference samples in the memory buffer, we can also consider the global maximum interference samples. We additionally store the minimum loss $\mathcal{L}(f_\theta(\boldsymbol{x}), Y)$ in the memory, denoted $\mathcal{L}(f_{\theta^*}(\boldsymbol{x}), Y)$. Then we can use this criterion instead:

$$s_{MI-2}(\boldsymbol{x}) = \mathcal{L}(f_{\theta^v}(\boldsymbol{x}), Y) - \min(\mathcal{L}(f_\theta(\boldsymbol{x}), Y), \mathcal{L}(f_{\theta^*}(\boldsymbol{x}), Y)). \tag{9}$$

Some experiments have shown that the larger buffer size is more effective, but with the attendant large computational toll(Aljundi et al., 2019a). To this end, we consider first sampling a random budget $\mathcal{B}$ from the memory buffer and then rehearsal k samples from $\mathcal{B}$ according to the criterion.

4 EXPERIMENTS

4.1 SETUP

Our experiment uses a widely used dataset, CIFAR10(Krizhevsky, 2009), which in order to satisfy the CL setting we split into 5 disjoint tasks, each with 2 classes(Aljundi et al., 2019c). Since we are using a memory-based algorithm, we set the memory buffer size $n(\mathcal{M}) = 500$ and the budget size $n(\mathcal{B}) = 50$, and for each step, we select the same size samples from the buffer as the training batch. Each of these tasks contains 9,750 training samples and 250 validation samples and keeps the identical PLL setting as in previous work(Wen et al., 2021). We generate false positive samples according to a fixed flipping probability $q = P(\bar{y} \in Y \mid \bar{y} \neq y)$, i.e. all non-ground-truth labels have the same flipping probability. We then combine all the false positive labels with the ground-truth label to construct the candidate set. In our experiments, we took a flipping probability of $q = 0.3$ and the average number of labels in the candidate set was 3.7018.

We compared the performance of five classical CL algorithms on Split CIFAR10: 1) **ER-MIR**(Aljundi et al., 2019a) is a fully supervised MIR algorithm based on rehearsal method; 2) **iCaRL**(Sperl et al., 2016) uses a nearest-class-mean classifier and knowledge distillation loss to decouple representation and classification; 3) **GEM**(Lopez-Paz & Ranzato, 2017) uses a smart way of combining the gradients of old and new tasks to update the network together; 4) **GSS**(Aljundi et al., 2019b) gives a score to each sample in the memory buffer and keeps the lower scores when the buffer is full; 5) **Naive** retrains a fixed number of random samples directly from the memory buffer.

4.2 MAIN EXPIRICAL RESULTS

We used two different batch sizes for our experiments, respectively 10 and 24. From Table 1, we can see that PCL has approximated the results of the fully supervised learning algorithm to some extent. PCL-24 achieves **SOTA** levels among the five benchmark algorithms. Despite the additional computational toll that partial label imposes on CL, we believe that increasing the size of the memory buffer and batch size will result in better performance.

Table 1: Comparison of the average accuracy of different algorithms in Split CIFAR10

MATHOD	T1	T2	T3	T4	T5	AVERAGE
Navie	0.03	0.08	0.04	0.15	0.93	0.246
ER-MIR	0.07	0.17	0.00	0.14	0.90	0.256
iCaRL	0.21	0.12	0.11	0.17	0.35	0.192
GEM	0.00	0.01	0.02	0.07	0.91	0.202
GSS	0.06	0.05	0.20	0.65	0.92	0.376
PCL-10(ours)	0.04	0.01	0.02	0.15	0.75	0.194
PCL-24(ours)	0.32	0.21	0.35	0.46	0.54	**0.376**

5 Related Works

Continual learning (CL), also known as lifelong learning or class-incremental learning, allows the same model to incrementally learn new tasks or add new classes. Some regularisation-based methods use a weighting of the neurons of a neural network, making some of the neurons more sensitive to a specific task(Kirkpatrick et al., 2016). Li & Hoiem (2016) propose to preserve the old knowledge from the previous tasks by using knowledge distillation(Hinton et al., 2015). However, this method is more dependent on the relevance between tasks. Most of the memory-based methods store a subset of the data through a buffer or use a generative network to reconstruct the old data, and the different algorithms mainly focus on designing the criteria for adding and removing from the buffer(Aljundi et al., 2019a;b). The dynamic architecture methods can expand dynamically based on the representation of new tasks by neurons, and adjust the network architecture when the current neurons are insufficient to represent new tasks(Lee et al., 2020; Yoon et al., 2017).

Partial label learning (PLL) task assumes that the candidate set comprises a ground-truth label and helps the network to converge through different disambiguation strategies. Some intuitive average-based methods aim to treat all labels in a candidate set equally, using the average of the model's output of all candidate labels as a prediction(Hüllermeier & Beringer, 2005; Lv et al., 2020; Cour et al., 2011). Some identification-based methods also receiving widespread attention in recent years, which treat the ground-truth label in a candidate set as a latent variable and constantly identify this latent variable to disambiguate. These include methods based on the maximum likelihood criterion(Jin & Ghahramani, 2002), graph-based methods(Zhang et al., 2016; Lyu et al., 2019) or methods based on class activation values(Zhang et al., 2022).

6 Future Work

Our work demonstrates the possibilities of weak-supervised in continual learning. In future work, we wish to explore more different applications of weak-supervised learning tasks or unsupervised learning to continual learning and to theoretically prove the convergence of weak-supervised continual learning. We believe that this new learning paradigm can be easily extended to more computer vision tasks, such as object detection or semantic segmentation.

7 Discussion & Conclusion

In this work, we propose partial-label continual learning, which is a new paradigm for weak-supervised continual learning. The experiments show that continual learning scalability on partial label tasks is possible and competitive with fully supervised continual learning on many benchmarks. A good weak-supervised learning representation can help reduce catastrophic forgetting, and a good continuous learning strategy can in turn help label disambiguation. However, weak-supervised tasks also bring an unavoidable drawback to continual learning: for an online (potentially infinite) stream of data partial label learning increases severely the training time, and how to reduce the training time is crucial.

ACKNOWLEDGEMENTS

The authors acknowledge Professors Tongliang Liu and Lei Feng for their helpful tutoring and support throughout. The authors acknowledge the Sydney Informatics Hub and the University of Sydneys high performance computing cluster, Artemis, for providing the computing resources that have contributed to the results reported herein. Acknowledgements also thanks for GPUs support from `AutoDL.com`, which focus on providing various professional and economical hardware solutions for deep learning research.

REFERENCES

Rahaf Aljundi, Eugene Belilovsky, Tinne Tuytelaars, Laurent Charlin, Massimo Caccia, Min Lin, and Lucas Page-Caccia. Online continual learning with maximal interfered retrieval. *neural information processing systems*, 2019a.

Rahaf Aljundi, Min Lin, Baptiste Goujaud, and Yoshua Bengio. Gradient based sample selection for online continual learning. *neural information processing systems*, 2019b.

Rahaf Aljundi, Min Lin, Baptiste Goujaud, and Yoshua Bengio. Online continual learning with no task boundaries. 2019c.

Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K. Dokania, Philip H. S. Torr, and Marc'Aurelio Ranzato. Continual learning with tiny episodic memories. 2019a.

Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K. Dokania, Philip H. S. Torr, and Marc'Aurelio Ranzato. On tiny episodic memories in continual learning. *arXiv: Learning*, 2019b.

Ching-Hui Chen, Vishal M. Patel, and Rama Chellappa. Learning from ambiguously labeled face images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

Timothee Cour, Ben Sapp, and Ben Taskar. Learning from partial labels. *Journal of Machine Learning Research*, 2011.

Ian Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv: Machine Learning*, 2013.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. *arXiv: Computer Vision and Pattern Recognition*, 2019.

Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. *arXiv: Machine Learning*, 2015.

Eyke Hüllermeier and Jürgen Beringer. Learning from ambiguously labeled examples. *Intelligent Data Analysis*, 2005.

Rong Jin and Zoubin Ghahramani. Learning with multiple labels. *neural information processing systems*, 2002.

Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. *neural information processing systems*, 2020.

James Kirkpatrick, Razvan Pascanu, Neil C. Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences of the United States of America*, 2016.

Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *Communications of The ACM*, 2012.

Soochan Lee, Junsoo Ha, Dongsu Zhang, and Gunhee Kim. A neural dirichlet process mixture model for task-free continual learning. *Learning*, 2020.

Timothée Lesort, Vincenzo Lomonaco, Andrei Stoian, Davide Maltoni, David Filliat, and Natalia Díaz-Rodríguez. Continual learning for robotics: Definition, framework, learning strategies, opportunities and challenges. *Information Fusion*, 2019.

Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2016.

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

Jie Luo and Francesco Orabona. Learning from candidate labeling sets. *neural information processing systems*, 2010.

Jiaqi Lv, Miao Xu, Lei Feng, Gang Niu, Xin Geng, and Masashi Sugiyama. Progressive identification of true labels for partial-label learning. *international conference on machine learning*, 2020.

Gengyu Lyu, Songhe Feng, Tao Wang, Congyan Lang, and Yidong Li. Gm-pll: Graph matching based partial label learning. *IEEE Transactions on Knowledge and Data Engineering*, 2019.

Zheda Mai, Ruiwen Li, Jihwan Jeong, David Quispe, Hyun-Woo Kim, and Scott Sanner. Online continual learning in image classification: An empirical survey. *arXiv: Learning*, 2021.

Ameya Prabhu, Philip H. S. Torr, and Puneet K. Dokania. Gdumb: A simple approach that questions our progress in continual learning. *european conference on computer vision*, 2020.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. icarl: Incremental classifier and representation learning. *computer vision and pattern recognition*, 2016.

David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy P. Lillicrap, and Greg Wayne. Experience replay for continual learning. *neural information processing systems*, 2018.

Kihyuk Sohn, David Berthelot, Chun-Liang Li, Zizhao Zhang, Nicholas Carlini, Ekin Cubuk, Alex Kurakin, Han Zhang, and Colin Raffel. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. *neural information processing systems*, 2022.

G.T. Sperl, Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. icarl: Incremental classifier and representation learning. *computer vision and pattern recognition*, 2016.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv: Learning*, 2018.

Haobo Wang, Ruixuan Xiao, Yixuan Li, Lei Feng, Gang Niu, Gang Chen, and Junbo Zhao. Pico: Contrastive label disambiguation for partial label learning. *Learning*, 2022.

Hongwei Wen, Hongwei Wen, Jingyi Cui, Hanyuan Hang, Jiabin Liu, Yisen Wang, and Zhouchen Lin. Leveraged weighted loss for partial label learning. *international conference on machine learning*, 2021.

Mang Ye, Xu Zhang, Pong C. Yuen, and Shih-Fu Chang. Unsupervised embedding learning via invariant and spreading instance feature. *computer vision and pattern recognition*, 2019.

Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. Lifelong learning with dynamically expandable networks. *Learning*, 2017.

Fei Zhang, Lei Feng, Bo Han, Tongliang Liu, Gang Niu, Tao Qin, Masashi Sugiyama, and Microsoft Research. Exploiting class activation value for partial-label learning. *Learning*, 2022.

Min-Ling Zhang, Bin-Bin Zhou, and Xu-Ying Liu. Partial label learning via feature-aware disambiguation. *knowledge discovery and data mining*, 2016.

A PCL ALGORITHM

Algorithm 1 Partial-label Continual Learning Algorithm (start from second task)

Input: Training dataset $\mathcal{D}_t$, classifier f, query network q', key network k', momentum queue, pseudo-labels $\boldsymbol{s}_i$ of $\boldsymbol{x}_i$ in $\mathcal{D}_t$, prototype embeddings $\boldsymbol{\mu}_j$ $(1 \leq j \leq C)$, Memory buffer $\mathcal{M}$, subset size $\mathcal{N}$, Budget $\mathcal{B}$, learning rate α

for $t \in 1 \cdots T$ **do**
  for $epoch = 1, 2, \cdots,$ **do**
    // a mini-batch B_i from D_t
    //would-be parameters
    $\theta^w \leftarrow SGD(B_i, \alpha)$
    // randomly select $\mathcal{N}$ samples from $\mathcal{M}$
    $B_{\mathcal{N}} \sim \mathcal{M}$
    // sort $B_{\mathcal{N}}$ based on MIR criteria
    $S \leftarrow sort(s_{MI}(B_{\mathcal{N}}))$
    $B_{\mathcal{M}_{\mathcal{N}}} \leftarrow \{S_i\}_{i=1}^{\mathcal{B}}$
    $B = B_i \cup B_{\mathcal{M}_{\mathcal{N}}}$
    // get query and key embedding
    $B_q = \{\boldsymbol{q}_i = q'(\text{Aug}_q(\boldsymbol{x}_i)) \mid \boldsymbol{x}_i \in B\}$
    $B_k = \{\boldsymbol{k}_i = k'(\text{Aug}_k(\boldsymbol{x}_i)) \mid \boldsymbol{x}_i \in B\}$
    $A = B_q \cup B_k \cup \text{queue}$
    for $\boldsymbol{x}_i \in B$ **do**
      // get the prediction
      $\tilde{y}_i = \arg\max_{j \in Y_i} f^j(\text{Aug}_q(\boldsymbol{x}_i))$
      // update prototype
      $\boldsymbol{\mu}_c = \text{Normalize}(\gamma \boldsymbol{\mu}_c + (1-\gamma)\boldsymbol{q}_i)$, if $\tilde{y}_i = c$
      // construct positive set
      $P(\boldsymbol{x}_i) = \{\boldsymbol{k}' \mid \boldsymbol{k}' \in A(\boldsymbol{x}_i), \tilde{y}' = \tilde{y}_i\}$
    end for
    // update pesudo-labels
    for $\boldsymbol{q}_i \in B_q$ **do**
      $\boldsymbol{z}_i = \text{OneHot}(\arg\max_{j \in Y_i} \boldsymbol{q}_i^\top \boldsymbol{\mu}_j)$
      $\boldsymbol{s}_i = \phi \boldsymbol{s}_i + (1+\phi)\boldsymbol{z}_i$
    end for
    // Contrastive loss
    $\mathcal{L}_{\text{cont}}(q; \tau, A) = \frac{1}{|B_q|} \sum_{\boldsymbol{q}_i \in B_q} \left\{ -\frac{1}{|P(\boldsymbol{x}_i)|} \sum_{\boldsymbol{k}_+ \in P(\boldsymbol{x}_i)} \log \frac{\exp(\boldsymbol{q}^\top \boldsymbol{k}_+/\tau)}{\sum_{\boldsymbol{k}' \in A(\boldsymbol{x}_i)} \exp(\boldsymbol{q}^\top \boldsymbol{k}'/\tau)} \right\}$
    // Classification loss
    $\mathcal{L}_{\text{cls}}(f; B) = \frac{1}{|B|} \sum_{x_i \in B} \sum_{j=1}^{C} -s_{i,j} \log(f^j(\text{Aug}_q(\boldsymbol{x}_i)))$
    minimize loss $\mathcal{L} = \mathcal{L}_{\text{cls}} + \lambda \mathcal{L}_{\text{cont}}$
    momentum update k' by using q'
    enqueue B_k and classifier predictions and dequeue
  end for
  $M \leftarrow UpdateMemory(B_i)$
end for

B ALGORITHM COMPARISON