Multi-Label Continual Learning Using Augmented
Graph Convolutional Network

Kaile Du, Fan Lyu, Member, IEEE, Linyan Li, Fuyuan Hu, Member, IEEE, Wei Feng, Member, IEEE,
Fenglei Xu, Xuefeng Xi, and Hanjing Cheng

Abstract—Multi-Label Continual Learning (MLCL) is a framework designed for class-incremental multi-label image recognition. However, MLCL faces two critical challenges: the construction of label relationships on past-missing and future-missing partial labels of training data, and the problem of catastrophic forgetting, which leads to poor generalization. To address these challenges, this study proposes an enhanced version of the Augmented Graph Convolutional Network (AGCN++), capable of constructing cross-task label relationships and mitigating catastrophic forgetting. First, an Augmented Correlation Matrix (ACM) is constructed across all observed classes, incorporating intra-task relationships derived from hard label statistics. Additionally, inter-task relationships are established by leveraging both hard and soft labels obtained from the data, as well as a constructed expert network. Next, a novel partial label encoder (PLE) is introduced for MLCL, enabling the extraction of dynamic class representations for each partial label image as graph nodes. This PLE also facilitates the generation of soft labels, which contribute to the creation of a more persuasive ACM and effectively mitigate forgetting. Lastly, a relationship-preserving constrainter is proposed to address the issue of forgetting label dependencies across old tasks. In the AGCN++, the label relationships topology can be augmented automatically, thereby generating efficient class representations. The effectiveness of the proposed method is evaluated using two multi-label image benchmarks. The experimental results demonstrate that the proposed approach is highly effective in the context of MLCL image recognition. It can establish compelling correlations across tasks, even in scenarios where the old task labels are missing.

Index Terms—Continual learning, multi-label recognition, partial label encoder, augmented correlation matrix.

Manuscript received 9 November 2022; revised 7 June 2023; accepted 11 August 2023. Date of publication 16 August 2023; date of current version 14 February 2024. This work was supported in part by the Natural Science Foundation of China under Grant 61876121, in part by the Postgraduate Research & Practice Innovation Program of Jiangsu Province under Grant SJKX21-1414, and in part by Suzhou Science and Technology Development Plan (Science and Technology Innovation for Social Development) Project under Grant ss202133. The Associate Editor coordinating the review of this manuscript and approving it for publication was Dr Liang Lin. (Kaile Du and Fan Lyu contributed equally to this work.) (Corresponding author: Fuyuan Hu.)

Kaile Du, Fuyuan Hu, Fenglei Xu, Xuefeng Xi, and Hanjing Cheng are with the Suzhou University of Science and Technology, Suzhou 215009, China, and also with the College of Automation, Southeast University, Nanjing 210000, China (e-mail: kailedu@post.usts.edu.cn; fuyuahu@mail.usts.edu.cn; xufi@mail.usts.edu.cn; xfxi@mail.usts.edu.cn; chj@mail.usts.edu.cn).

Fan Lyu and Wei Feng are with the College of Intelligence and Computing, Tianjin University, Tianjin 300350, China (e-mail: fanlyu@tju.edu.cn; wengf@tju.edu.cn).

Linyan Li is with the Suzhou Institute of Trade & Commerce, Suzhou 215009, China (e-mail: lilinyan@szjm.edu.cn).

Digital Object Identifier 10.1109/TMM.2023.3305871

I. INTRODUCTION

Machine learning approaches have achieved human-level performance on various tasks, such as Atari games [1] or object recognition [2]. However, these approaches typically assume that no new knowledge will be introduced to the models, which is unrealistic in real-world scenarios. To address this limitation, continual learning has been developed to enable intelligent systems to continuously learn new tasks from sequential datasets while retaining the knowledge of previously learned tasks [3].

Recently, class-incremental continual learning [4] has emerged as an approach that focuses on adaptively evolving classifiers for seen classes without access to task identification during inference [5], mirroring real-life applications. Compared to traditional continual learning, class-incremental models face the additional challenge of distinguishing between all seen classes from all tasks, making the problem more demanding. In the context of continual learning, the inability of training data for old tasks when new tasks arrive poses challenges for preserving old knowledge. The phenomenon known as catastrophic forgetting [6] occurs as the model incrementally learns new knowledge, resulting in a decline in performance for previously learned tasks over time. In Multi-Label Continual Learning (MLCL), the primary challenge lies in learning new tasks without catastrophically forgetting the previous tasks.

Existing methods for class-incremental continual learning have made significant progress in addressing this challenge. The rehearsal-based methods [7], [8], [9], [10], [11] stores samples from raw datasets or generates pseudo-samples with a generative model [12], [13], these samples are replayed while learning a new task to prevent forgetting. The regularization-based methods [6], [14], [15], [16], [17] have an additional regularization term introduced in the loss function, consolidating previous knowledge when learning on new tasks. And the parameter isolation methods [18], [19], [20] dedicates different model parameters to each task to alleviate any possible forgetting. Moreover, some recent transformer-based methods [21], [22], [23], [24] have also achieved good performance.

However, most existing methods for class-incremental continual learning focus on single-labeled input images, which we refer to as Single-Label Continual Learning (SLCL). SLCL has limitations in practical applications, such as movie categorization and scene classification, where multi-label data is prevalent. As shown in Fig. 1, an image contains multiple

Multi-Label Continual Learning Using Augmented
Graph Convolutional Network

Kaile Du, Fan Lyu, Member, IEEE, Linyan Li, Fuyuan Hu, Member, IEEE, Wei Feng, Member, IEEE,
Fenglei Xu, Xuefeng Xi, and Hanjing Cheng

Abstract—Multi-Label Continual Learning (MLCL) is a framework designed for class-incremental multi-label image recognition. However, MLCL faces two critical challenges: the construction of label relationships on past-missing and future-missing partial labels of training data, and the problem of catastrophic forgetting, which leads to poor generalization. To address these challenges, this study proposes an enhanced version of the Augmented Graph Convolutional Network (AGCN++), capable of constructing cross-task label relationships and mitigating catastrophic forgetting. First, an Augmented Correlation Matrix (ACM) is constructed across all observed classes, incorporating intra-task relationships derived from hard label statistics. Additionally, inter-task relationships are established by leveraging both hard and soft labels obtained from the data, as well as a constructed expert network. Next, a novel partial label encoder (PLE) is introduced for MLCL, enabling the extraction of dynamic class representations for each partial label image as graph nodes. This PLE also facilitates the generation of soft labels, which contribute to the creation of a more persuasive ACM and effectively mitigate forgetting. Lastly, a relationship-preserving constrainter is proposed to address the issue of forgetting label dependencies across old tasks. In the AGCN++, the label relationships topology can be augmented automatically, thereby generating efficient class representations. The effectiveness of the proposed method is evaluated using two multi-label image benchmarks. The experimental results demonstrate that the proposed approach is highly effective in the context of MLCL image recognition. It can establish compelling correlations across tasks, even in scenarios where the old task labels are missing.

Index Terms—Continual learning, multi-label recognition, partial label encoder, augmented correlation matrix.

Manuscript received 9 November 2022; revised 7 June 2023; accepted 11 August 2023. Date of publication 16 August 2023; date of current version 14 February 2024. This work was supported in part by the Natural Science Foundation of China under Grant 61876121, in part by the Postgraduate Research & Practice Innovation Program of Jiangsu Province under Grant SJKX21-1414, and in part by Suzhou Science and Technology Development Plan (Science and Technology Innovation for Social Development) Project under Grant ss202133. The Associate Editor coordinating the review of this manuscript and approving it for publication was Dr Liang Lin. (Kaile Du and Fan Lyu contributed equally to this work.) (Corresponding author: Fuyuan Hu.)

Kaile Du, Fuyuan Hu, Fenglei Xu, Xuefeng Xi, and Hanjing Cheng are with the Suzhou University of Science and Technology, Suzhou 215009, China, and also with the College of Automation, Southeast University, Nanjing 210000, China (e-mail: kailedu@post.usts.edu.cn; fuyuahu@mail.usts.edu.cn; xufi@mail.usts.edu.cn; xfxi@mail.usts.edu.cn; chj@mail.usts.edu.cn).

Fan Lyu and Wei Feng are with the College of Intelligence and Computing, Tianjin University, Tianjin 300350, China (e-mail: fanlyu@tju.edu.cn; wengf@tju.edu.cn).

Linyan Li is with the Suzhou Institute of Trade & Commerce, Suzhou 215009, China (e-mail: lilinyan@szjm.edu.cn).

Digital Object Identifier 10.1109/TMM.2023.3305871

I. INTRODUCTION

Machine learning approaches have achieved human-level performance on various tasks, such as Atari games [1] or object recognition [2]. However, these approaches typically assume that no new knowledge will be introduced to the models, which is unrealistic in real-world scenarios. To address this limitation, continual learning has been developed to enable intelligent systems to continuously learn new tasks from sequential datasets while retaining the knowledge of previously learned tasks [3].

Recently, class-incremental continual learning [4] has emerged as an approach that focuses on adaptively evolving classifiers for seen classes without access to task identification during inference [5], mirroring real-life applications. Compared to traditional continual learning, class-incremental models face the additional challenge of distinguishing between all seen classes from all tasks, making the problem more demanding. In the context of continual learning, the inability of training data for old tasks when new tasks arrive poses challenges for preserving old knowledge. The phenomenon known as catastrophic forgetting [6] occurs as the model incrementally learns new knowledge, resulting in a decline in performance for previously learned tasks over time. In Multi-Label Continual Learning (MLCL), the primary challenge lies in learning new tasks without catastrophically forgetting the previous tasks.

Existing methods for class-incremental continual learning have made significant progress in addressing this challenge. The rehearsal-based methods [7], [8], [9], [10], [11] stores samples from raw datasets or generates pseudo-samples with a generative model [12], [13], these samples are replayed while learning a new task to prevent forgetting. The regularization-based methods [6], [14], [15], [16], [17] have an additional regularization term introduced in the loss function, consolidating previous knowledge when learning on new tasks. And the parameter isolation methods [18], [19], [20] dedicates different model parameters to each task to alleviate any possible forgetting. Moreover, some recent transformer-based methods [21], [22], [23], [24] have also achieved good performance.

However, most existing methods for class-incremental continual learning focus on single-labeled input images, which we refer to as Single-Label Continual Learning (SLCL). SLCL has limitations in practical applications, such as movie categorization and scene classification, where multi-label data is prevalent. As shown in Fig. 1, an image contains multiple

1520-9210 © 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
This study focuses on the incremental scenario of multi-task classification, specifically class-incremental classification, and explores the sequential learning of new classes in the context of Multi-label Continual Learning (MLCL). The inference process of MLCL is illustrated in Fig. 1, where the model can progressively recognize multiple labels as new classes are continuously learned. However, in continual learning, the unavailability of data pertaining to future and past classes poses a significant challenge in MLCL compared to Single-Label Continual Learning (SLCL). This challenge is known as the partial label problem, which arises due to overlapping label spaces across different tasks in MLCL. To illustrate, as shown in Fig. 1, the label “sky” is present in all three tasks, serving as an overlapped label. In task 1, “sky” represents a future latent label, while in task 3, it represents a past latent label. If the past latent label is not annotated during current training, it leads to the issue of past-missing partial labels. Similarly, future-missing partial labels arise when the future latent label is not annotated. Consequently, in MLCL, each task cannot be trained independently due to the partial label problem. An MLCL model should be capable of incrementally recognizing multiple labels as new classes are continuously learned.

In practical terms, we address the challenges of MLCL in two real-world labeling scenarios: Continuous Labeling MLCL (CL-MLCL) and Independent Labeling MLCL (IL-MLCL). In CL-MLCL, the current training dataset contains both past and current labels, whereas IL-MLCL only has current labels. IL-MLCL faces the issues of past-missing and future-missing partial labels, while CL-MLCL only deals with the future-missing partial label problem. In both scenarios, the partial label problem presents a significant challenge in establishing multi-label relationships and preventing catastrophic forgetting. Given these challenges, it becomes crucial to explore a viable solution to address the partial label problem in MLCL. This serves as our motivation to develop a unified MLCL solution for the sequential multi-label classification problem. Our approach considers label relationships across tasks in both IL-MLCL and CL-MLCL scenarios, aiming to overcome the challenges posed by the partial label problem in MLCL.

This article serves as an extension of our previous work, Augmented Graph Convolutional Network (AGCN) [32], and aims to address the challenges of MLCL in real-world scenarios (IL-MLCL and CL-MLCL) by proposing an enhanced version called AGCN++. AGCN++ consists of three main components. Firstly, to establish connections between partial labels across tasks, we propose the construction of an Augmented Correlation Matrix (ACM) in a sequential manner for MLCL. We develop a unified ACM constructor that operates differently for CL-MLCL and IL-MLCL. In CL-MLCL, ACM is updated using hard label statistics from new training data at each task. For IL-MLCL, an auto-updated expert network is designed to generate predictions for the old tasks, which are then used as soft labels in constructing the ACM. Secondly, effective class representation becomes challenging due to the partial label problems. In our earlier conference work, the AGCN model relied on pre-existing semantic information (e.g., word embeddings) as class representation. However, this fixed class representation could accumulate errors during the construction of label relationships, leading to skewed predictions and more severe forgetting. To address this, AGCN++ incorporates a Partial Label Encoder (PLE) to decompose the features of each partial image into dynamic class representations. These class-specific representations, varying from image to image, are used as graph nodes in AGCN++. Unlike AGCN, which manually adds graph nodes, PLE automatically generates graph nodes for each partial label image. Utilizing PLE for graph node generation also reduces the impact of low-quality word embeddings, enabling AGCN++ to generate a more accurate ACM and alleviate forgetting. Thirdly, we propose to encode the dynamically constructed ACM and graph nodes. The AGCN++ model employs a convolutional architecture to correlate the label spaces of both old and new tasks, extracting latent correlations between every pair of classes. This information is combined with visual features for prediction. To further mitigate forgetting, we introduce a distillation loss function for class-level forgetting and a relationship-preserving graph loss function for relationship-level forgetting. By incorporating these components, AGCN++ enhances the ability to handle partial label problems, construct an accurate ACM, and mitigate forgetting, ultimately improving the performance of MLCL.
MLCL image recognition. AGCN++ successfully establishes robust correlations across tasks, even when the labels of previous tasks are missing (IL-MLCL) or present (CL-MLCL).

The contributions of this article extend our previous work on AGCN [32] and encompass the following key elements:
1) We complete the real-world scenario of MLCL from IL-MLCL to CL-MLCL scenarios, and a unified AGCN++ model is redesigned to capture label dependencies to improve multi-label recognition in the data stream;
2) We propose a novel partial label encoder (PLE) to decompose the global image features into dynamic graph nodes for each partial label image, which reduces the accumulation of errors in the construction of label relationships and suppresses forgetting;
3) We propose a unified ACM constructor. The ACM is dynamically constructed using soft or hard labels to build label relationships across sequential tasks of MLCL to solve the partial label problem for IL-MLCL and CL-MLCL. The distillation loss and relationship-preserving loss readjust to IL-MLCL and CL-MLCL to mitigate the class- and relationship-level catastrophic forgetting;
4) More experimental results are provided, including extensive comparisons on two different scenarios settings and more ablation studies, etc. More ablation studies and new SOTA MLCL results are provided.

II. RELATED WORK

A. Class-Incremental Continual Learning

Class-incremental continual learning [4] builds a classifier that learns a sequence of new tasks corresponding to different classes. The state-of-art methods for class-incremental continual learning can be categorized into three main branches to solve the catastrophic forgetting problem.

First, the regularization-based methods [6], [14], [15], [16], [17], this line of work introduces additional regularization terms in the loss function to consolidate previous knowledge when learning new tasks. These methods are based on regularizing the parameters corresponding to the old tasks, penalizing the feature drift on the old tasks and avoiding storing raw inputs. Kirkpatrick et al. [6] limits changes to parameters based on their significance to the previous tasks using Fisher information; LwF [16] is a data-focused method, and it leverages the knowledge distillation combined with a standard cross-entropy loss to mitigate forgetting and transfer knowledge by storing the previous parameters. Thusaethan et al. [14] propose an indicator loss, which is associated with the distillation mechanism that preserves the existing upcoming emotion knowledge. Yang et al. [15] introduce an attentive feature distillation approach to mitigate catastrophic forgetting while accounting for semantic spatial- and channel-level dependencies. The regularization-based procedures can protect privacy better because they do not collect samples from the original dataset.

Second, the rehearsal-based methods [4], [8], [9], [13], [33], [34], [35], [36], [37], which sample a limited subset of data from the previous tasks or a generative model as the memory. The stored memory is replayed while learning new tasks to mitigate forgetting. In ER [34], this memory is retrained as the extended training dataset during the current training; RM [8] is a replay method for the blurry setting; iCaRL [4] selects and stores samples closest to the feature mean of each class for replaying; AGEM [35] resets the training gradient by combining the gradient on the memory and training data; Ye et al. [13] propose a Teacher-Student network framework. The Teacher module would remind the Student about the information learnt in the past.

Third, the parameter isolation based methods [18], [19], [20], [38], which generate task-specific parameter expansion or sub-branch. When no limits apply to the size of networks, Expert Gate [20] grows new branches for new tasks by dedicating a model copy to each task. PackNet [18] iteratively assigns parameter subsets to consecutive tasks by constructing binary masks.

Though the existing methods have achieved successes in SLC, they are hardly used in MLCL directly. The overlook of the partial label problem means the inevitability of more serious forgetting in MLCL, let alone the construction of multi-label relationships and reducing the forgetting of the relationships.

B. Multi-Label Image Classification

Compared with the traditional single-label classification problem, multi-label classification is more practical in the real world. Earlier multi-label learning methods [39] prefer to build the model with the help of extra-label localisation information, which is assumed to include all possible foreground objects. And they aggregate the features from proposals to incorporate local information for multi-label prediction. However, extra localisation information is costly, preventing the models from applying to end-to-end training approaches. More recent advances are mainly by constructing label relationships. Some works [27], [28] use the recurrent neural network (RNN) for multi-label recognition under a restrictive assumption that the label relationships are in order, which limits the complex relationships in label space. Furthermore, some works [29], [30], [31] build label relationships using graph structure and use graph convolutional network (GCN) to enhance the representation. The standard limit of these methods is that they can only construct the intra-task correlation matrix using the training data from the current task and fail to capture the inter-task label dependencies in a continual data stream. They rely on prior knowledge to construct the correlation matrix, which is the key of GCN that aims to gain the label dependencies. These methods utilize the information of the whole training dataset to capture the co-occurrence patterns of objects in an offline way. Some recent methods focus on the partial label problem [40], [41], [42] for offline multi-label learning. Compared with this offline way, we construct the correlations and use soft label statistics to solve the partial label problem for MLCL. Kim et al. [10] propose to extend the ER [34] algorithm using an improved reservoir sampling strategy to study the imbalanced problem on multi-label datasets. However, the label dependencies are ignored in this work [10]. In contrast, we propose to model label relationships sequentially in MLCL and consider mitigating the relationship-level forgetting in MLCL.
TABLE I
TRAINING AND TESTING LABEL SETS OF TASK \( t \) IN TWO SCENARIOS, CL-MLCL AND IL-MLCL.

|       | CL-MLCL          | IL-MLCL          |
|-------|------------------|------------------|
| Train | \( C_{\text{seen}}^t = C_{\text{seen}}^{t-1} \cup C^t \) | \( C^t \)        |
| Test  | \( C_{\text{seen}}^t = C_{\text{seen}}^{t-1} \cup C^t \) | \( C_{\text{seen}}^t = C_{\text{seen}}^{t-1} \cup C^t \) |

III. MULTI-LABEL CONTINUOUS LEARNING

A. Definition of MLCL

Given \( T \) tasks with respect to training datasets \( \{D_{\text{tm}}^1, \ldots, D_{\text{tm}}^T\} \) and test datasets \( \{D_{\text{ts}}^1, \ldots, D_{\text{ts}}^T\} \), the total class numbers increase gradually with the sequential tasks in MLCL and the model is constantly learning new knowledge. A continual learning system trains on the training sets from \( D_{\text{tm}}^1 \) to \( D_{\text{tm}}^T \) sequentially and evaluate on all seen test sets at any time. For the \( t \)-th new task, the new and task-specific classes are to be trained, namely, \( C^t \). MLCL aims at learning a multi-label classifier to discriminate the increasing number of classes in the continual learning process. We denote \( C_{\text{seen}}^{t-1} = \bigcup_{n=1}^{t-1} C^n \) as seen classes at task \( t \), which contains old class set \( C_{\text{seen}}^{t-1} \) and new class set \( C^t \), that is, \( C_{\text{seen}}^{t-1} = C_{\text{seen}}^{t-1} \cup C^t \), and \( C_{\text{seen}}^{t-1} \cap C^t = \emptyset \).

B. MLCL Scenarios

In this section, considering academic and practical requirements, we introduce the two scenarios in MLCL. In one scenario, we adopt continual learning and cannot obtain the old class like most single-Label continual learning methods [9], [10], [12], [13]. In the other scenario, we consider the real-world setting. IL-MLCL setting has hard task boundaries, so the old classes are unavailable. Conversely, similar to the settings in [8] and [43], CL-MLCL setup makes the task boundaries faint. It is closer to the real world, where new classes do not show up exclusively. The difference between the two scenarios is the training label space for old classes.

1) Continuous Labelling (CL-MLCL): CL-MLCL is a more realistic scenario where the data distribution shifts gradually without hard task boundaries. The annotator of CL-MLCL needs to label all seen classes. The class numbers of training data increase gradually with the sequential tasks, i.e., \( C_{\text{seen}}^t \) for training data of task \( t \). As shown in Table I, the label space \( Y \subseteq C_{\text{seen}}^t \). The past latent label is annotated, so the old and new labels coexist for a current sample in CL-MLCL. This scenario is labor-costly, especially when the class number is large. Because the past latent label is annotated, only the future-missing partial label problem will occur in CL-MLCL, and no past-missing partial label problem will occur.

2) Independent Labelling (IL-MLCL): In this scenario, the annotator only labels the new classes in \( C^t \) for training data in task \( t \), as shown in Table I. This means the training label space is independently labelled with sequential class-incremental tasks. The old and new labels do not overlap in new task samples in IL-MLCL. The training label space \( Y \) of IL-MLCL at task \( t \) is right the task-specific label (new labels) set \( C^t \). IL-MLCL can reduce the labelling cost, but due to the lack of old labels in the IL-MLCL label space, the past latent label is not annotated, so a past-missing partial label problem will be caused together with future-missing partial label.

3) Test Phase and the Goal: During the test phase, the ground truth for each data point contains all the old classes \( C_{\text{seen}}^{t-1} \) and task-specific classes \( C^t \) for both CL-MLCL and IL-MLCL. That is, as shown in Table I, the label space in the test phase is the all seen classes \( C_{\text{seen}}^t \). This article aims to propose a unified approach to solve the MLCL problem in both IL-MLCL and CL-MLCL scenarios.

IV. METHODOLOGY

A. Overview of the Proposed Method

In multi-label learning, label relationships are verified effective to improve the recognition [29], [30], [31]. However, it is challenging to construct convincing label relationships in MLCL image recognition because of the partial label problem. The partial label problem results in difficulty in constructing the inter-task label relationships. Moreover, forgetting happens not only at the class level but also at the relationship level, which may damage performance.

For effective multi-label recognition, we propose an AGCN++ to construct and update the intra- and inter-task label relationships during the training process. As shown in Fig. 2(a), AGCN++ model is mainly composed of three parts: 1) Partial label encoder (PLE) decomposes the image feature extracted by the CNN into a group of class-specific representations, these representations are used as graph nodes to feed the GCN model. 2) Augmented Correlation Matrix (ACM) provides the label relationships among all seen classes \( C_{\text{seen}}^t \) and is augmented to capture the intra- and inter-task label dependencies. 3) Graph Convolutional Network encodes ACM and graph nodes \( H \) into label representations \( y_{\text{gph}} \) for label relationships. We followed [16] and [17], after each task has been trained, the AGCN++ model is saved as the expert model. Our AGCN++ is mainly composed of CNN block and GCN block. After the AGCN++ model is saved into the expert model, the expert model is composed of CNN_{gph} and GCN_{gph}. And after each task is trained, the expert model is updated by AGCN++ which has been trained. As shown in Fig. 2(a) and (b), the most significant difference between AGCN and AGCN++ is that AGCN++ can extract graph nodes from the original image through PLE. GCN encodes ACM and graph nodes to get \( y_{\text{gph}} \). By adding \( y_{\text{cal}} \) and \( y_{\text{gph}} \), the soft label generated by the model can better replace the past-missing partial label, more convincing ACM (see Fig. 8) can be developed for IL-MLCL, and the forgetting of CL-MLCL and IL-MLCL can be reduced through knowledge distillation. These can improve the performance of the model.

B. Partial Label Encoder

Due to the partial label problems in MLCL, effective class representation is difficult to build. In AGCN, the model utilized pre-given word embedding as fixed class representation, which will lead to the accumulation of errors in the construction of
denotes the class-incremental prediction scores. (b) Compared with AGCN++, AGCN directly uses word embedding as graph nodes. Graph \( = \Theta \) are multiple copies \( \otimes \in [44] \Theta \) \( R \) \( \text{Duplicate} \) is copied \( t \) times to get \( \text{R} \) \( \text{fc} \) \( (2) \) \( = \) \( \odot \in \text{R} \) \( \text{represents the class-specific fc layer parameters. The } \in \text{Duplicate} \text{q} \in \text{to} \text{is the Hadamard Product.} \text{R} \in \text{fc} \) \( (2) \) \( = \) \( \odot \in \text{R} \) \( \text{represents the image feature dimensionality.} \text{We use a fully connected layer } \text{fc}( \cdot ) \text{ to achieve two goals. One is to get the prediction without adding label dependencies } y^\text{cal} \).

\[
\hat{y}^\text{cal} = \text{fc}(\text{CNN}(x)) \in \mathbb{R}^{|\text{C}_\text{seen}|},
\]

(1)

The other is to make the image feature aware of class information by doing Hadamard Product with its parameters.

\[
\text{H}^t = \Theta \otimes \text{cat}(p, q) \in \mathbb{R}^{|\text{C}_\text{seen}| \times D},
\]

(2)

where \( \otimes \) is the Hadamard Product. \( p \) and \( q \) are multiple copies of respective image features. \( p = \text{Duplicate}(\text{CNN}_{\text{xp}}(x)) \in \mathbb{R}^{|\text{C}_\text{seen}| \times D}, q = \text{Duplicate}(\text{CNN}(x)) \in \mathbb{R}^{t \times D} \). For example, \( \text{CNN}(x) \in \mathbb{R}^{|C| \times D} \) is copied \( |C| \) times to get \( q \in \mathbb{R}^{t \times D}, \Theta \in \mathbb{R}^{|\text{C}_\text{seen}| \times D} \) represents the class-specific fc layer parameters. The dimension of \( \Theta \) is continuously expanded to accommodate the class-incremental characteristic in continual learning. In (2), \( \text{H}^t \) represents the class-aware graph node and automatically augments as the new task progresses. We then encode \( \text{H}^t \) by Graph Convolutional Network (GCN) to get graph representation \( \tilde{y}^\text{gph} \).

\[
\tilde{y}^\text{gph} = \text{GCN}(\text{A}^t, \text{H}^t) \in \mathbb{R}^{|\text{C}_\text{seen}|},
\]

(3)

where \( \text{A}^t \) denotes the Augmented Correlation Matrix (ACM, see the next section for details). GCN is a two-layer stacked graph model similar to ML-GCN [29], [31]. ACM \( \text{A}^t \) and graph node \( \text{H}^t \) can be augmented as the class number increments. With the established ACM, GCN provides dynamic label relationships to CNN for prediction.

Moreover, we introduce the prediction \( \hat{y}^\text{cal} \) without adding label dependencies, which is combined with \( \tilde{y}^\text{gph} \) as the final multi-label prediction \( \hat{y} \in \mathbb{R}^{|\text{C}_\text{seen}|} \) of our model:

\[
\hat{y} = \sigma (\hat{y}^\text{cal} + \tilde{y}^\text{gph}),
\]

(4)

where \( \sigma(\cdot) \) represents the Sigmoid function.
ACM represents the auto-updated dependency among all seen classes in the MLCL image recognition system. The next section will introduce how to establish and augment ACM in AGCN++.

C. Augmented Correlation Matrix

Most existing multi-label learning algorithms [29], [30], [31] rely on constructing the inferring label correlation matrix $A$ by the hard label statistics among the class set $C$: $A_{ij} = P(C_i|C_j)_{i 
eq j}$. The correlation matrix represents a fully-connected graph. When a new task comes, the graph should be augmented automatically. However, in MLCL, the label correlation matrix is hard to infer directly by statistics because of the partial label problem.

To tackle the problem, as shown in Fig. 4(b), we introduce an auto-updated expert network inspired by [16] and [17], which is used to provide missing past labels. We take the expert model based on knowledge distillation as the basis of our method. We use the expert model to generate soft labels instead of missing hard labels from the dataset. As the soft label for the $t$-th task, we have $\mathbf{\hat{z}} = \exp(x)$. Based on the soft labels, as shown in Fig. 4(a), we construct an Augmented Correlation Matrix (ACM) $A^t$ in IL-MLCL and CL-MLCL:

$$A^t = \begin{bmatrix} A^{t-1} & R^t \\ Q^t & B^t \end{bmatrix} \iff \begin{bmatrix} \text{Old-Old} & \text{Old-New} \\ \text{New-Old} & \text{New-New} \end{bmatrix},$$

in which we take four block matrices including $A^{t-1}$ and $B^t$, $R^t$ and $Q^t$ to represent intra- and inter-task label relationships between old and old classes, new and new classes, old and new classes as well as new and old classes respectively. For the first task, $A^1 = B^1$. For $t > 1$, $A^t \in \mathbb{R}^{(|C|_t \times |C|_t)}$. It is worth noting that the block $A^{t-1}$ (Old-Old) can be derived from the old task, so we focus on how to compute the other three blocks in the ACM.

New-New block ($B^t \in \mathbb{R}^{(|C^t|_t \times |C^t|_t)}$). As shown in Fig. 4(a), this block computes the intra-task label relationships among the new classes, and the conditional probability in $B^t$ can be calculated using the hard label statistics from the training dataset similar to the common multi-label learning:

$$B^t_{ij} = P(C^t_i \in C^t | C^t_j \in C^t) = \frac{N_{ij}}{N_j}, \quad (6)$$

where $N_{ij}$ is the number of examples with both class $C^t_i$ and $C^t_j$, $N_j$ is the number of examples with class $C^t_j$. Due to the data stream, $N_{ij}$ and $N_j$ are accumulated and updated at each step of the training process. This block is shared by both IL- and CL-MLCL, because the new class data is always available.

Old-New block ($R^t \in \mathbb{R}^{(|C^t|_t \times |C^t-1|_t)}$). As shown in Fig. 4(b), for CL-MLCL, this block can be directly obtained by the hard label statistics. For IL-MLCL, given an image $x$, for old classes, $\mathbf{\hat{z}}_i$ (predicted probability) generated by the expert can be considered as the soft label for the $i$-th class. Thus, the product $\mathbf{\hat{z}}_i y_j$ can be regarded as an alternative of the cooccurrences of $C^t_{i-1}$ and $C^t_j$. Thus, $\sum_{x} \mathbf{\hat{z}}_i y_j$ means the online mini-batch accumulation:

$$R^t_{ij} = P(c_{i-1}^{t-1} \in c_{i}^{t-1} | c_{i}^{t-1} \in C^t) = \frac{N_{ij}}{N_j}, \quad \text{if CL-MLCL},$$

$$= \frac{\sum_{x} \mathbf{\hat{z}}_i y_j}{N_j}, \quad \text{if IL-MLCL}, \quad (7)$$

where $N_{ij}$ is the accumulated number of examples with both class $C^t_{i-1}$ and $C^t_j$, $N_j$ is the accumulated number of examples with class $C^t_j$.

New-Old block ($Q^t \in \mathbb{R}^{(|C^t-1|_t \times |C^t|_t)}$). As shown in Fig. 4(b), for CL-MLCL, the inter-task relationship between new and old classes can be computed using hard label statistics. For IL-MLCL, based on the Bayes’ rule, we can obtain this block
Algorithm 1: Training procedure of AGCN++.

\textbf{Input:} $\mathcal{D}_{\text{trn}}^t$

\begin{algorithmic}[1]
\FOR {$t = 1 : T$}
\FOR {$(x, y) \sim \mathcal{D}_{\text{trn}}^t$}
\IF {$t = 1$}
\STATE Compute $A^1$ with $y$ using Eq. (6);
\STATE $H^1, \hat{y}_{\text{cal}} = \text{PLE}(\text{CNN}(x))$;
\STATE $y_{\text{cal}} = \text{GCN}(A^1, H^1)$;
\STATE $\hat{y} = \sigma(\hat{y}_{\text{cal}} \oplus y_{\text{gph}})$;
\STATE $\ell = \ell_{\text{cls}}(y, \hat{y})$;
\ELSE
\STATE $\hat{z} = \text{expert}(x)$;
\STATE // get soft labels.
\STATE Compute $B^t$ with $y$ using Eq. (6);
\STATE Compute $R^t$ and $Q^t$ using Eq. (7) and (8);
\STATE $A^t = [A^{t-1} \quad R^t]$;
\STATE // compute ACM of task $t$.
\STATE $H^t, \hat{y}_{\text{cal}} = \text{PLE}(\text{CNN}(x))$;
\STATE $y_{\text{cal}} = \text{GCN}(A^t, H^t)$;
\STATE // get new graph representation.
\STATE $y_{\text{gph}} = \text{GCN}_{\text{gph}}(A^{t-1}, H^{t-1})$;
\STATE // get target representation.
\STATE $\hat{y} = \sigma(\hat{y}_{\text{cal}} \oplus y_{\text{gph}})$;
\STATE // get class prediction.
\STATE $\ell = \lambda_{1}\ell_{\text{cls}}(y, \hat{y}) + \lambda_{2}\ell_{\text{cal}}(\hat{y}, \hat{y}_{\text{old}})$
\STATE $+ \lambda_{3}\ell_{\text{gph}}(y_{\text{gph}}^t, \hat{y}_{\text{gph}}^t)$;
\STATE // compute the final loss.
\ENDFOR
\STATE Update AGCN++ model by minimizing $\ell$
\STATE // save parameters to the expert model.
\ENDFOR
\end{algorithmic}

where $A^1$ is constructed by the statistics of hard labels $y$. After the input $x$ is fed to the CNN, the class-specific feature $\hat{y}_{\text{cal}}$ and the graph nodes $H^1$ is obtained by PLE. Then the GCN encodes $A^1$ and $H^1$ to get graph representation $\hat{y}_{\text{gph}}$. The prediction score $\hat{y}$ is generated by $\hat{y}_{\text{cal}}$ and $\hat{y}_{\text{gph}}$ (Line 4–8).

2) When $t > 1$, the ACM $A^t$ is augmented via soft labels $\hat{z}$ and the Bayes’ rule. Based on the $A^t$, GCN model can capture both intra- and inter-task label dependencies. Then, $\hat{z}$ and $y_{\text{gph}}^t$ as target features to build $\ell_{\text{cal}}$ and $\ell_{\text{gph}}$ respectively (Line 9–20).

3) The AGCN++ and expert models are updated respectively (Line 21–23).
V. EXPERIMENTS

A. Datasets

1) Dataset Description: We use two datasets, Split-COCO and Split-WIDE, to evaluate the effectiveness of the proposed method.

Split-COCO. We construct Split-COCO by selecting the 40 most frequent concepts from the 81 classes of MS-COCO [45]. This dataset contains 65,082 examples for training and 27,173 examples for validation. The 40 classes are divided into ten different and non-overlapping tasks, with each task consisting of four classes. Split-WIDE. NUS-WIDE [46] is a raw web-crawled multi-label image dataset. We further curate a sequential class-incremental dataset from NUS-WIDE. Following [47], we choose the 21 most frequent concepts from 81 classes of NUS-WIDE to construct the Split-WIDE, which has 144,858 examples for training and 41,146 examples for validation. Split-WIDE has a larger scale than Split-COCO. We split the Split-WIDE into 7 tasks, where each task contains 3 classes.

2) Dataset Collection: We enlist the curation details of Split-COCO and Split-WIDE. In the previous continual learning methods, Shmelkov et al. [48] selects 20 out of 80 classes to create 2 tasks for SLCL, each with 10 classes. Nguyen et al. [49] tailor MSCOCO for continual learning of captioning. They select 24 out of 80 classes to create two tasks. PRS [10] needs more low-frequency classes to study the imbalanced problem. They curate four tasks with 70 classes using MSCOCO. Compared to these previous splitting, on the one hand, we set more tasks to test the robustness of the algorithm over more tasks to create a continual setting. On the other hand, we selected more frequent concepts from the original dataset to reduce the long-tail effect of the original data. Multi-label datasets inherently have intersecting concepts among the data points. Hence, a naive splitting strategy may lead to a dangerous amount of data loss. This motivates us to minimize data loss during the split. Moreover, to test diverse research environments, the second objective is to keep the size of the splits balanced optionally. To split the well-known MS-COCO and NUS-WIDE into several different tasks fairly and uniformly, we introduce two kinds of labelling in the datasets. 1) Specific-labelling: If an image only has the labels that belong to the task-special class set \(C^t\) of task \(t\), we regard it as a specific-labelling image for task \(t\); 2) Mixed-labelling: If an image not only has the task-specific labels but also has the old labels belonging to the class set \(C^{t-1}\), we regard it as a mixed-labelling image.

In IL-MLCL, because the model learns from the task-specific labels \(C^t\), the training data is labelled without old labels, so IL-MLCL will suffer from the partial label problem, which mainly appears in the mixed-labelling image. The IL-MLCL and CL-MLCL share the same training images. A randomly data-splitting approach may lead to the imbalance of specific-labelling and mixed-labelling images for each task. We split two datasets into sequential tasks with the following strategies to ensure a proper proportion. We first count the number of labels for each image. Then, we give priority to leaving specific-label images for each task. The mixed-labelling images are then allocated to other tasks. The dataset construction is presented in Fig. 5.

B. Evaluation Metrics

Multi-label evaluation. Following these multi-label learning methods [29], [30], [31], 7 metrics are leveraged in MLCL. (1) the average precision (AP) on each label and the mean average precision (mAP) over all labels; (2) the per-class F1-measure (CF1); (3) the overall F1-measure (OF1). The mAP, CF1 and OF1 are relatively more important for multi-label performance evaluation. Moreover, we adopt 4 other metrics: per-class precision (CP), per-class recall (CR), overall precision (OP) and overall recall (CR).

\[
\begin{align*}
OP &= \frac{\sum_i N^c_i}{\sum_i N^p_i}, \\
CP &= \frac{1}{C} \sum_i \frac{N^c_i}{N^p_i}, \\
OR &= \frac{\sum_i N^c_i}{\sum_i N^g_i}, \\
CR &= \frac{1}{C} \sum_i \frac{N^c_i}{N^g_i}, \\
OF1 &= \frac{2 \times OP \times OR}{OP + OR}, \quad CF1 = \frac{2 \times CP \times CR}{CP + CR}
\end{align*}
\]

where \(i\) is the class label and \(C\) is the number of labels. \(N^c_i\) is the number of correctly predicted images for class \(i\), \(N^p_i\) is the number of predicted images for class \(i\) and \(N^g_i\) is the number of ground-truth for class \(i\).

Forgetting measure [50]. This metric denotes the above multi-label metric value difference for each task between testing when it was first trained, and the last task was trained. For example, the forgetting measure of mAP for a task \(t\) can be computed by its performance difference between task \(T\) and \(t\) was trained. \(F_t\), average forgetting after the model has been trained continuously up till task \(t \in \{1, \ldots, T\}\) is defined as:

\[
F_t = \frac{1}{t-1} \sum_{j=1}^{t-1} f^t_j,
\]

where \(f^t_j\) is the forgetting on task \(j\) after the model is trained up till task \(t\) and computed as

\[
f^t_j = \max_{\ell \in \{1, \ldots, k-1\}} a_{\ell,j} - a_{t,j},
\]

where \(a\) denotes accuracy in single-label continual learning. In multi-label continual learning, \(a\) denotes every metric in multi-label continual learning, specifically, mAP, CP, CR, CF1, OP, OR and OF1. We evaluate the final forgetting \(F_T\) after training the final task.
TABLE II
WE REPORT 7 METRICS (%) FOR MULTI-LABEL CLASSIFICATION AFTER THE WHOLE DATA STREAM IS SEEN ONCE ON SPLIT-WIDE IN BOTH IL-MLCL AND CL-MLCL SCENARIOS. THE MULTI-TASK IS OFFLINE TRAINED AS THE UPPER BOUND, AND FINE-TUNING IS THE LOWER BOUND

| Method          | Split-WIDE IL-MLCL |                  | Split-WIDE CL-MLCL |                  |
|-----------------|---------------------|------------------|---------------------|------------------|
|                 | mAP    | CP     | CR   | CF1   | OP   | OR   | OP   | OF1   | mAP    | CP     | CR   | CF1   | OP   | OR   | OP   | OF1   |
| Multi-Task      | 66.17  | 69.15  | 55.30 | 61.45 | 77.74 | 66.30 | 71.57 | 69.19  | 60.60  | 43.96 | 50.33 | 78.45 | 59.39 | 66.67 |
| Fine-Tuning     | 40.85  | 35.96  | 24.70 | 36.77 | 53.22 | 33.12 | 50.37 | 36.54  | 32.60  | 18.20 | 24.87 | 32.17 | 21.80 | 29.66 |
| Forgetting      | 22.03  | 13.99  | 39.53 | 28.27 | 54.29 | 32.97 | 53.70 | 34.04  | 53.33  | 18.73 | 38.18 | 53.72 | 25.17 | 39.09 |
| EWC [6]         | 34.86  | 35.51  | 24.23 | 38.18 | 54.81 | 39.55 | 54.17 | 14.73 | 38.18 | 53.72 | 25.17 | 39.09 | 4.06  |
| LwF [10]        | 29.46  | 21.65  | 49.66 | 23.64 | 50.77 | 69.70 | 42.09 | 46.44 | 51.05  | 33.01 | 40.09 | 54.24 | 40.40 | 60.01 |
| AGEM [35]       | 32.47  | 23.26  | 58.44 | 33.28 | 26.36 | 74.40 | 38.93 | 46.83  | 50.48  | 27.67 | 35.75 | 47.93 | 35.18 | 40.58 |
| Forgetting      | 16.42  | 20.09  | 8.67  | 15.71 | 26.55 | 6.95  | 9.73  | 11.91  | 10.05  | 48.36 | 33.55 | 11.21 | 21.56 | 17.22 |
| ER [34]         | 34.03  | 24.64  | 60.02 | 34.94 | 26.62 | 73.57 | 39.37 | 48.08  | 53.33  | 31.16 | 39.61 | 53.40 | 38.84 | 44.98 |
| Forgetting      | 15.15  | 26.18  | 7.14  | 11.80 | 26.45 | 6.25  | 8.61  | 9.24   | 7.13   | 43.32 | 28.78 | 27.58 | 5.96  | 14.53 |
| PRS [10]        | 39.70  | 52.77  | 83.24 | 63.48 | 60.81 | 14.05 | 22.19 | 51.42  | 58.26  | 37.64 | 55.73 | 55.66 | 48.90 | 52.06 |
| Forgetting      | 11.24  | 4.08   | 43.22 | 34.48 | 23.84 | 55.73 | 43.76 | 7.86   | 2.17   | 31.72 | 16.68 | 11.36 | 7.13  |
| SCR [7]         | 35.34  | 28.33  | 54.34 | 35.47 | 32.31 | 70.28 | 41.92 | 49.23  | 53.87  | 36.86 | 43.77 | 50.16 | 47.58 | 48.84 |
| Forgetting      | 14.26  | 21.29  | 9.56  | 10.17 | 23.09 | 7.26  | 8.04  | 8.34   | 7.89   | 39.22 | 20.56 | 6.62  | 13.56 | 10.78 |
| AGCN [32]       | 45.73  | 33.08  | 61.57 | 48.04 | 57.62 | 45.26 | 57.07 | 54.20  | 54.26  | 54.20  | 54.26  | 49.03 | 74.96 | 59.29 |
| AGCN++          | 8.32   | 17.28  | 7.44  | 2.13  | 22.11 | 5.52  | 3.98  | 4.45   | 3.56   | 19.48 | 10.64 | 6.13  | 1.88  | 1.02  |

The boldface indicates SOTA results.

C. Implementation Details
Following existing multi-label image classification methods [29, 30, 31], we employ ResNet101 [51] as the image feature extractor pre-trained on ImageNet [52]. We adopt Adam [53] as the optimizer of network with \( \beta_1 = 0.9, \beta_2 = 0.999, \) and \( \epsilon = 10^{-8} \). Following [29, 30], our AGCN++ consists of two GCN layers with output dimensionality of 1024 and 2048, respectively. The input images are randomly cropped and resized to 448 \times 448 with random horizontal flips for data augmentation. The network is trained for a single epoch like most continual learning methods done [4, 8, 9, 33].

D. Baseline Methods
MLCL is a new paradigm of continual learning. We compare our method with several essential and state-of-art continual learning methods, including (1) EWC [6], which regularizes the training loss to avoid catastrophic forgetting; (2) LwF [16], which uses the distillation loss by saving task-specific parameters; (3) ER [34], which saves a few training data from the old tasks and re-trains them in the current training; (4) AGEM [35], resets the training gradient by combining the gradient on the memory and training data; (5) PRS [10], which uses an improved reservoir sampling strategy to study the imbalanced problem. PRS studies similar problems with us. Still, they focus more on the imbalanced problem but ignore the label relationships and the problem of partial labels for MLCL image recognition. (6) SCR [7], which proposes the NCM classifier to improve SLCL performance. SCR is an algorithm designed to improve the top-1 accuracy of single-label recognition. Similar to [7], [10], [17], we use a Multi-Task baseline, which is trained on a single pass over shuffled data from all tasks. It can be seen as the performance upper bound. We also compare with the Fine-Tuning, which performs training without any continual learning technique. Thus, it can be regarded as the performance lower bound.

Note that, to extend some SLCL methods to MLCL, we turn the final Softmax layer in each of these methods into a Sigmoid. Other details follow their original settings.

E. Main Results

1) Split-WIDE Results: In Table II, with the establishment of relationships and inhibition of class-level and relationship-level forgetting using distillation and relationship-preserving loss, our method shows better performance than the other state-of-art performances in both IL-MLCL and CL-MLCL scenarios. In particular, AGCN and AGCN++ perform better than other comparison methods on three more essential evaluation metrics, including mAP, CF1 and OF1, which means the effectiveness in multi-label classification. Also, in the forgetting value evaluated after task 7, we achieve a better forgetting measure, which means the stability of the proposed method in MLCL. In the IL-MLCL scenario, because we use soft labels to replace hard labels in the old task label space and establish and remember the label relationships, the AGCN++ outperforms the most state-of-art performances by a large margin: 45.73\% vs. 42.15\% (\( + 3.58\%)\) on mAP, 43.04\% vs. 37.99\% (\( + 5.05\%\)) on CF1 and 45.26\% vs. 43.70\% (\( + 1.56\%\)) on OF1, as shown in Table II. Like IL-MLCL, CL-MLCL still needs to model complete label dependencies between label relationships and reduce forgetting. The AGCN++ shows better performance than the others in CL-MLCL: 57.07\% vs. 54.20\% (\( + 2.87\%\)) on mAP, 54.66\% vs. 46.13\% (\( + 8.53\%\)) on CF1 and 45.26\% vs. 43.70\% (\( + 1.56\%\)) on OF1, as demonstrated in Table II, which suggests that AGCN++ is effective in a large-scale multi-label dataset.

2) Split-COCO Results: Split-COCO is split into ten tasks, as mentioned in [54], compared with methods PRS [10] and ER [34], our approach can protect privacy better because AGCN++ does not collect data from the original dataset. As shown in Table III, in IL-MLCL and CL-MLCL, AGCN++ achieves better performance than the others in most metrics.
AGCN++ also has a low rate of forgetting old knowledge. With the AGCN++ combining intra- and inter-task label relationships, the proposed AGCN++ outperforms the most state-of-art performances in IL-MLCL: 38.23% vs. 34.11% (+ 4.12%) on mAP, 41.38% vs. 35.49% (+ 5.89%) on CF1 and 45.26% vs. 42.37 (+ 2.89%) on OF1. This means soft labels can effectively replace hard labels in the old task label space to alleviate the partial label problem. AGCN++ is also better in CL-MLCL: 53.49% vs. 48.82% (+ 4.67%) on mAP, 49.55% vs. 39.18% (+ 10.37%) on CF1 and 59.32% vs. 56.76% (+ 2.56%) on OF1. This means ACM is effective for both IL-MLCL and CL-MLCL scenarios in Split-COCO. As illustrated above, AGCN++ can be a uniform MLCL method for IL-MLCL and CL-MLCL.

F. More MLCL Settings

To demonstrate the robustness of our method, we conducted additional experiments to evaluate the effectiveness of AGCN and AGCN++ under different MLCL settings. These settings include assigning more classes to each task, assigning a random number of classes to each task, and using random task orders. First, we increased the number of classes in each task to examine the capability of our proposed PLE and ACM in handling a larger number of label relationships within a task. Specifically, 8-way for Split-COCO and 7-way for Split-WIDE. As shown in Table IV, our AGCN and AGCN++ can still achieve better results in three more important metrics, mAP, CF1 and OF1. Take the mAP, for example. For Split-COCO, 65.92% vs. 62.60% (+ 3.32%) in IL 8-way and 73.41% vs. 70.24% (+ 3.17%) in CL 8-way. For Split-WIDE, 56.63% vs. 54.11% (+ 2.52%) in IL 7-way and 61.04% vs. 58.98% (+ 2.06%) in CL 7-way.

Second, considering the variability in the number of classes among different tasks in real-world scenarios, we randomly assigned a different number of classes to each task. Specifically, the task setting is “7 : 4 : 1 : 6 : 2 : 2 : 5 : 7 : 3 : 3”. As shown in Fig. 6, our method consistently outperformed the comparison methods in every task, further supporting the effectiveness of AGCN++ across various task configurations.

Third, in order to assess the robustness of a continual learning system in relation to task order, we conducted experiments using three distinct task orders: one positive order and two random orders. As shown in Table V, our method consistently outperformed the comparison methods in every task, further supporting the effectiveness of AGCN++ across various task configurations.

AGCN++ also has a low rate of forgetting old knowledge. With the AGCN++ combining intra- and inter-task label relationships, the proposed AGCN++ outperforms the most state-of-art performances in IL-MLCL: 38.23% vs. 34.11% (+ 4.12%) on mAP, 41.38% vs. 35.49% (+ 5.89%) on CF1 and 45.26% vs. 42.37 (+ 2.89%) on OF1. This means soft labels can effectively replace hard labels in the old task label space to alleviate the partial label problem. AGCN++ is also better in CL-MLCL: 53.49% vs. 48.82% (+ 4.67%) on mAP, 49.55% vs. 39.18% (+ 10.37%) on CF1 and 59.32% vs. 56.76% (+ 2.56%) on OF1. This means ACM is effective for both IL-MLCL and CL-MLCL scenarios in Split-COCO. As illustrated above, AGCN++ can be a uniform MLCL method for IL-MLCL and CL-MLCL.

F. More MLCL Settings

To demonstrate the robustness of our method, we conducted additional experiments to evaluate the effectiveness of AGCN and AGCN++ under different MLCL settings. These settings include assigning more classes to each task, assigning a random number of classes to each task, and using random task orders.
outperformed other approaches across all three task orders. This serves as evidence of the robustness of our approach in both IL-MLCL and CL-MLCL settings, as it consistently delivers superior performance regardless of the task order.

These additional experiments provide further evidence of the effectiveness and robustness of AGCN and AGCN++ in various MLCL settings, validating their performance from multiple perspectives.

### G. mAP Curves

Similar to [7], [8], [17], we show the mAP trends of different methods in Fig. 7 for sequential learning. These curves indicate the performance along the MLCL progress. In two MLCL scenarios, Fig. 7 illustrates the mAP changes as tasks are being learned on two benchmarks. The mAP curves show that AGCN and AGCN++ can perform better through the MLCL process. In addition, our algorithm is applied after the first task for most continual learning methods. Our AGCN and AGCN++ have modeled the label dependencies from the first task. As distillation loss and relationship-preserving loss are applied to subsequent tasks, the algorithm’s performance exceeds other methods in each task.

### H. Ablation Studies

1) **ACM and Soft Labels Effectiveness**: To assess the effectiveness of the intra- and inter-task relationships in AGCN++, we conducted ablation experiments on ACM. Soft labels were generated using an expert model, and we evaluated the impact of soft labels in constructing multi-label relationships by comparing the performance of our method in IL-MLCL and CL-MLCL scenarios.

As described in Section IV-C, R1 (Old-New) and Q1 (New-Old) are used to model inter-task label dependencies across old and new tasks, B1 (New-New) is used to model intra-task label dependencies, and while R1 and Q1 are unavailable, neither are R1−1 and Q1−1 in block A1−1, the A1−1 that inherit from the old task only build intra-task relationships. Table VI demonstrates the results when label relationships across old and new tasks (w/o R1 & Q1) are not established. AGCN++ (Line 1 and 3) already outperforms most non-AGCN methods in terms of performance. For example, the comparison between AGCN++ (w/o R1 & Q1) and LwF (w/o A1−1 & B1 & R1 & Q1) on mAP is: 42.25% vs. 29.46% (Split-WIDE, IL-MLCL), 54.72% vs. 46.44% (Split-WIDE, CL-MLCL), 39.19% vs. 19.95% (Split-COCO, IL-MLCL) and 51.51% vs. 40.87% (Split-COCO, CL-MLCL), as shown in Tables VI, II and III. AGCN++ also outperforms most non-AGCN methods in terms of CF1 and OF1. These findings suggest that intra-task label relationships alone are effective for MLCL image recognition. When the inter-task block matrices Rt and Qt are available, AGCN++ incorporating both intra- and inter-task relationships (Line 2 and 4) achieves even better performance across all three evaluation metrics. For example, mAP comparisons of AGCN++ (w/o A1−1 & B1 & R1 & Q1) and PRS on two datasets in two scenarios: 45.73% vs. 39.70% (Split-WIDE, IL-MLCL), 57.07% vs. 51.42% (Split-WIDE, CL-MLCL), 38.23% vs. 31.08% (Split-WIDE, CL-MLCL), as shown in Tables VI, II and III. These results indicate that incorporating inter-task relationships can enhance multi-label recognition.

As mentioned in Table VI, “H” refers to hard labels, “S” refers to soft labels, and “S-H” denotes the combination of soft and hard labels. As explained in Section IV-C, in IL-MLCL, A1−1 & B1 is built with soft and hard labels, while R1 & Q1 is built with soft labels. In CL-MLCL, both A1−1 & B1 & R1 & Q1 are built using hard labels. Table VI further demonstrates that, similar to hard labels (Line 3-4), soft labels can also contribute to building label relationships and improve the overall performance of MLCL models (Line 1-2).

2) **PLE Effectiveness**: As shown in Table VII, AGCN++ (w/ PLE) outperforms AGCN++ (w/o PLE) in all three

### TABLE V

| Method        | IL-MLCL mAP (%) | CL-MLCL mAP (%) | IL-MLCL CF1 (%) | CL-MLCL CF1 (%) |
|---------------|----------------|----------------|----------------|----------------|
| Multi-Task    |                |                |                |                |
| Fine-Tuning   | 65.85          | 61.79          | 66.27          | 60.68          |
| EWC [6]       | 12.20          | 12.50          | 29.67          | 33.83          |
| LwF [16]      | 19.95          | 21.69          | 40.68          | 48.87          |
| PRS [16]      | 31.08          | 32.77          | 22.25          | 46.39          |
| AGCN [32]     | 34.11          | 35.49          | 42.37          | 48.82          |
| AGCN++        | 38.23          | 41.38          | 45.26          | 53.49          |

The boldface indicates SOTA results.
metrics, which can prove the effectiveness of PLE. For mAP, 45.73% vs. 42.15% (Split-WIDE, IL-MLCL), 57.07% vs. 54.20% (Split-WIDE, CL-MLCL), 38.23% vs. 34.11% (Split-COCO, IL-MLCL), 53.49% vs. 48.82% (Split-COCO, CL-MLCL).

3) Hyperparameter Selection: Then, we analyze the influences of loss weights and relationship-preserving loss on two benchmarks, as shown in Table VIII. When the relationship-preserving loss is unavailable, loss weight $\lambda_3$ is set to 0. The loss weights of others: $\lambda_1 = 0.10$, $\lambda_2 = 0.90$ for Split-WIDE in IL-MLCL, $\lambda_1 = 0.70$, $\lambda_2 = 0.30$ for Split-WIDE in CL-MLCL, $\lambda_1 = 0.15$, $\lambda_2 = 0.85$ for Split-COCO in IL-MLCL and $\lambda_1 = 0.40$, $\lambda_2 = 0.60$ for Split-COCO in CL-MLCL. By adding the relationship-preserving loss $\ell_{\text{gph}}$, the performance gets more gains, and the values of forgetting are also lower, which means the mitigation of relationship-level catastrophic forgetting is quite essential for MLCL image recognition, and the relationship-preserving loss is effective. We select the best $\lambda_3$ as the hyper-parameters, i.e., $\lambda_3 = 10^4$ for Split-WIDE in IL-MLCL, $\lambda_3 = 10^3$ for Split-WIDE in CL-MLCL, $\lambda_3 = 10^4$ for Split-COCO in IL-MLCL and $\lambda_3 = 10^5$ for Split-COCO in CL-MLCL.

![Fig. 7. mAP (%) on two benchmarks in both IL- and CL-MLCL.](image)

### Table VI

| A $^{t-1}$ & B $^t$ | R$^t$ & Q$^t$ | Split-WIDE | Split-COCO |
|-----------------|-----------------|-------------|-------------|
|                 |                 | mAP↑ CF1↑ OF1↑ | mAP↑ CF1↑ OF1↑ | mAP↑ CF1↑ OF1↑ |
| **IL-MLCL**     |                 |             |             |             |
| S-H             | x               | 42.25 40.47 42.98 | 35.19 39.98 37.62 |
| S-H             | S               | 45.73 43.04 45.26 | 38.23 41.38 45.26 |
| **CL-MLCL**     |                 |             |             |             |
| H               | x               | 54.72 30.41 48.84 | 51.47 47.13 56.97 |
| H               | H               | 57.07 54.66 59.29 | 53.49 49.55 59.32 |

The boldface indicates SOTA results.

### Table VII

| A $^{t-1}$ & B $^t$ | R$^t$ & Q$^t$ | Split-WIDE | Split-COCO |
|-----------------|-----------------|-------------|-------------|
|                 |                 | mAP↑ CF1↑ OF1↑ | mAP↑ CF1↑ OF1↑ | mAP↑ CF1↑ OF1↑ |
| **IL-MLCL**     |                 |             |             |             |
| w/o PLE         | 42.15 37.99 43.70 | 54.20 46.13 55.35 | 34.11 35.49 42.37 | 48.82 39.18 56.76 |
| w/ PLE          | **45.73** 43.04 45.26 | **57.07** 54.66 59.29 | **38.23** 41.38 45.26 | **53.49** 49.55 59.32 |

The boldface indicates SOTA results.

### Table VIII

| Loss Weights and Relationship-Preserving Loss | Split-WIDE | Split-COCO |
|---------------------------------------------|-------------|-------------|
| $\lambda_1$ & $\lambda_2$ & $\lambda_3$ | mAP↑ CF1↑ OF1↑ | mAP↑ CF1↑ OF1↑ |
| **IL-MLCL**                                |             |             |
| 0.10 & 0.90 & 0                            | 42.04 38.76 42.12 | 36.77 39.95 39.10 |
| Forgetting                                  | 10.48 5.24 6.42 | 22.34 12.73 13.32 |
| 0.10 & 0.90 & $10^4$                       | **45.73** 43.04 45.26 | **38.23** 41.38 45.26 |
| Forgetting                                  | 8.32 2.13 3.98 | 20.12 11.34 6.78 |
| **CL-MLCL**                                |             |             |
| 0.70 & 0.30 & 0                            | 55.68 51.58 49.24 | 50.98 46.82 54.70 |
| Forgetting                                  | 5.04 12.56 10.24 | 12.87 22.80 6.62 |
| 0.70 & 0.30 & $10^4$                       | **57.07** 54.66 59.29 | **53.49** 49.55 59.32 |
| Forgetting                                  | 4.45 10.64 1.02 | 7.24 14.52 1.24 |

The boldface indicates SOTA results.
I. **Visualization of ACM**

As shown in Fig. 8, we provide visualizations of the constructed ACM to validate its effectiveness in Split-WIDE and Split-COCO datasets for both IL-MLCL and CL-MLCL scenarios. We introduce the oracle augmented correlation matrix (oracle ACM) as the upper bound, which is constructed offline using hard label statistics from all tasks in the corresponding datasets. The Euclidean distance between the matrix and the Oracle ACM, denoted as $d$, is used as a measure of proximity. A smaller value of $d$ indicates that the matrix is closer to the oracle ACM, thus demonstrating superior construction.

As observed in Fig. 8, the proposed ACM in both scenarios exhibits close proximity to the oracle ACM. This implies that constructing the ACM using either soft or hard label statistics is effective. It should be noted that in the CL-MLCL scenario, the ACM is constructed solely using hard labels from the dataset. Consequently, the ACMs of AGCN++ and AGCN are the same under the same dataset. In contrast, in IL-MLCL, the ACM is constructed using soft labels generated by the model, leading to better construction in AGCN++. Additionally, in IL-MLCL, it can be observed that the ACM built in AGCN++ is closer to the Oracle ACM than in AGCN, with distances of 4.01% vs. 4.18% (Split-COCO) and 4.28% vs. 4.54% (Split-WIDE). This finding demonstrates that the PLE has mitigated the accumulation of errors during the construction of label relationships, resulting in improved accuracy.

VI. **CONCLUSION**

Multi-Label Continual Learning (MLCL) tackles the challenges of constructing label relationships and reducing forgetting in the context of continual learning for multi-label classification. The main difficulty lies in dealing with the partial label problem. In this article, we propose an innovative approach called AGCN++ that addresses the issues associated with MLCL. Our AGCN++ leverages an auto-updated expert mechanism to construct label relationships in a partial label data stream and effectively mitigate catastrophic forgetting, thereby improving overall performance. We investigate MLCL in both IL-MLCL and CL-MLCL scenarios. In the process of relationship construction, we demonstrate the effectiveness of utilizing soft or hard label statistics to update the correlation matrix, even when dealing with partial label data. Furthermore, we show the effectiveness of the PLE in reducing the accumulation of errors during the construction of label relationships and mitigating forgetting. To address forgetting, we propose an effective distillation loss and a novel relationship-preserving loss to alleviate class-level and relationship-level forgetting. Through extensive experiments, we validate that AGCN++ can capture label dependencies, resulting in improved MLCL performance in both IL-MLCL and CL-MLCL scenarios. In future work, we aim to explore methods to enhance the construction of the old-old block by utilizing the correlation of only soft labels instead of inheriting the previously constructed ACM. We believe that this approach will further enhance the performance of our model.

**REFERENCES**

[1] D. Silver et al., “A general reinforcement learning algorithm that masters chess, shogi, and go through self-play,” *Science*, vol. 362, no. 6419, pp. 1140–1144, 2018.
[2] O. Russakovsky et al., “Imagenet large scale visual recognition challenge,” *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, 2015.
[3] Z. Chen and B. Liu, “Lifelong machine learning,” *Synth. Lectures Artif. Intell. Mach. Learn.*, vol. 12, no. 3, pp. 1–207, 2018.
[4] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert, “ICARL: Incremental classifier and representation learning,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 2001–2010.
[5] G. M. van de Ven and A. S. Tolias, “Three scenarios for continual learning,” in Proc. NeurIPS Continual Learn. Workshop, 2018, pp. 1–18.

[6] J. Kirkpatrick et al., “Overcoming catastrophic forgetting in neural networks,” Nat. Acad. Sci., vol. 114, no. 13, pp. 3521–3526, 2017.

[7] Z. Mai, R. Li, H. Kim, and S. Sanner, “Supervised contrastive replay: Revisiting the nearest class mean classifier in online class-incremental continual learning,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2021, pp. 3589–3599.

[8] J. Bang, H. Kim, Y. Yoo, J.-W. Ha, and J. Choi, “Rainbow memory: Continual learning with a memory of diverse samples,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2021, pp. 8218–8227.

[9] F. Lyu et al., “Multi-domain multi-task rehearsal for lifelong learning,” in Proc. AAAI Conf. Artif. Intell., 2021, pp. 8819–8827.

[10] C. D. Kim, J. Jeong, and G. Kim, “Imbalanced continual learning with partitioning reservoir sampling,” in Proc. Eur. Conf. Comput. Vis., 2020, pp. 411–428.

[11] Q. Sun, F. Lyu, F. Shang, W. Feng, and L. Wan, “Exploring example influence in continual learning,” Adv. Neural Inf. Process. Syst., pp. 27075–27086, 2022.

[12] H. Li, W. Dong, and B.-G. Hu, “Incremental concept learning via online generative memory recall,” IEEE Trans. Neural Netw. Learn. Syst., vol. 32, no. 7, pp. 3206–3216, Jul. 2021.

[13] F. Ye and A. G. Bors, “Lifelong teacher-student network learning,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 44, no. 10, pp. 6280–6296, Oct. 2022.

[14] S. Thuseethan, S. Rajasegarar, and J. Yearwood, “Deep continual learning for emerging emotion recognition,” IEEE Trans. Multimedia, vol. 24, pp. 4367–4380, 2022.

[15] G. Yang et al., “Continual attentive fusion for incremental learning in semantic segmentation,” IEEE Trans. Multimedia, early access, Apr. 14, 2022, doi: 10.1109/TMM.2022.3167555.

[16] Z. Li and D. Hoiem, “Learning without forgetting,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 40, no. 12, pp. 2935–2947, Dec. 2018.

[17] D.-W. Zhou, H.-J. Ye, and D.-C. Zhan, “Few-shot class-incremental learning by visiting the nearest class mean classifier in online class-incremental continual learning,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2021, pp. 3589–3599.

[18] R. Aljundi, P. Chakravarty, and T. Tuytelaars, “Expert gate: Lifelong learning with a network of experts,” in Proc. Eur. Conf. Comput. Vis., 2022, pp. 9285–9295.

[19] A. Mallya, D. Davis, and S. Lazebnik, “Piggyback: Adapting a single network to multiple tasks by learning to mask weights,” in Proc. Eur. Conf. Comput. Vis., 2020, pp. 631–648.

[20] R. Aljundi, P. Chakravarty, and T. Tuytelaars. “Expert gate: Lifelong learning with a network of experts,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2017, pp. 3366–3375.

[21] K. Zhou, J. Yang, C. C. Loy, and Z. Liu, “Learning to prompt for vision-language models,” Int. J. Comput. Vis., vol. 130, no. 9, pp. 2337–2348, 2022.

[22] A. Douillard, A. Ramé, G. Couairon, and M. Cord, “Dytox: Transformers for continual learning with dynamic token expansion,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2022, pp. 9285–9295.

[23] Z. Wang et al., “Dualprompt: Complementary prompting for rehearsal-free continual learning,” in Proc. Eur. Conf. Comput. Vis., 2022, pp. 631–648.

[24] Y. Wang, Z. Huang, and X. Hong, “S-prompts learning with pre-trained transformers: An Ocean’s razor for domain incremental learning,” in Conf. Neural Inf. Process. Syst., 2022, pp. 5682–5695.

[25] F. Zhu, H. Li, W. Ouyang, N. Yu, and X. Wang, “Learning spatial regularization with image-level supervisions for multi-label image classification,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2017, pp. 5513–5522.

[26] Y. Li, Y. Song, and J. Luo, “Improving pairwise ranking for multi-label image classification,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2017, pp. 3617–3625.

[27] J. Wang et al., “CNN-RNN: A unified framework for multi-label image classification,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2016, pp. 2285–2294.

[28] F. Lyu, Q. Wu, F. Hu, Q. Wu, and M. Tan, “Attend and imagine: Multi-label image classification with visual attention and recurrent neural networks,” IEEE Trans. Multimedia, vol. 21, pp. 1971–1981, 2019.

[29] Z.-M. Chen, X.-S. Wei, P. Wang, and Y. Guo, “Multi-label image recognition with graph convolutional networks,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2019, pp. 5177–5186.

[30] T. Chen, L. Lin, X. Hui, R. Chen, and H. Wu, “Knowledge-guided multi-label few-shot learning for general image recognition,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 44, no. 3, pp. 1371–1384, Mar. 2022.

[31] Z. Chen, X.-S. Wei, P. Wang, and Y. Guo, “Learning graph convolutional networks for multi-label recognition and applications,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 45, no. 6, pp. 6969–6983, Jun. 2023.
Fan Lyu (Member, IEEE) received the B.S. and M.S. degrees in electronic and information engineering, Suzhou University of Science and Technology, Suzhou, China, in 2015 and 2018. He is currently working toward the Ph.D. degree with the College of Intelligence and Computing, Tianjin University, Tianjin, China. His research interests include continual learning, vision-language understanding, and low-shot learning.

Linyan Li received the master’s degree from Wuhan University, Wuhan, China, in 2007. She is currently an Associate Professor with the Suzhou Institute of Trade and Commerce, Suzhou, China. She is working in machine learning, neural information processing, and pattern recognition.

Fuyuan Hu (Member, IEEE) was a Postdoctoral Researcher with Vrije Universiteit Brussel, Brussels, Belgium, a Ph.D. student with Northwestern Polytechnical University, Xian, China, and a Visiting Ph.D. student with the City University of Hong Kong, Hong Kong. He is a Professor with the Suzhou University of Science and Technology, Suzhou, China. His research interests include machine learning, graphical models, structured learning, and tracking.

Wei Feng (Member, IEEE) received the B.S. and M.Phil. degrees in computer science from Northwestern Polytechnical University, Xian, China, in 2000 and 2003, respectively and the Ph.D. degree in computer science from City University of Hong Kong, Hong Kong, in 2008. From 2008 to 2010, he was a Research Fellow with the Chinese University of Hong Kong and City University of Hong Kong. He is currently a Full Professor with the School of Computer Science and Technology, Tianjin University, Tianjin, China. His research interests include active robotic vision and visual intelligence, specifically including active camera relocalization and lighting recurrence, general Markov Random Fields modeling, energy minimization, active 3D scene perception, SLAM, and generic pattern recognition. He focuses on solving preventive conservation problems of cultural heritages via computer vision and machine learning. He is the Associate Editor for Neurocomputing and the Journal of Ambient Intelligence and Humanized Computing.

Fenglei Xu received the B.Sc. degree in 2014 and the Ph.D. degree in 2020 from Nanjing University of Science and Technology, Nanjing, China. Since 2020, he has been a Lecturer with Suzhou University of Science and Technology, Suzhou, China. His research interests include 2D/3D object detection and semantic segmentation in driving scenes.

Xuefeng Xi received the bachelor’s degree from Jiangsu University, Zhenjiang, China, in 2000. He received the M.S. and Ph.D. degrees from Southeast University, Nanjing, China and Soochow University, Suzhou, China, in 2004 and 2017, respectively. Since 2012, he has been an Associate Professor with the College of Electronic and Information Engineering, Suzhou University of Science and Technology, Suzhou. His research interests include natural language processing, machine learning, and human-computer intelligent interaction in the smart home.

Hanjing Cheng was born in Anhui, China, in 1994. She received the B.Sc. and Ph.D. degrees in software engineering and control science and engineering from the Nanjing University of Science and Technology, Nanjing, China, in 2013 and 2022, respectively. From 2015 to 2016, she was a Visiting Ph.D. student with the Department of Computer Science, Brunel University London, Uxbridge, U.K. for six months. She is currently a Lecturer with the School of Electronic and Information Engineering, Suzhou University of Science and Technology, Suzhou, China. Her research interests include network pruning, multi-objective optimization, and deep learning.