Dynamic Hashtag Assignment: Leveraging Graph Convolutional Networks with Class Incremental Learning

Matteo Kolyszko, Marco Buzzelli, Simone Bianco
Department of Informatics, Systems and Communication
Università degli Studi di Milano-Bicocca
Milan, Italy

2024 IEEE 8th Forum on Research and Technologies for Society and Industry Innovation (RTSI) | 979-8-3503-6213-8/24/$31.00 ©2024 IEEE | DOI: 10.1109/RTSI61910.2024.10761214

***Abstract*—The use of hashtags plays a pivotal role in various applications. They have proven effective in social data mining, aiding information retrieval, sentiment analysis, event detection, and topic tracking. However, many users fail to include hashtags, leaving a vast amount of content unnoticed. As a result, automating hashtag recommendations has become essential. This work introduces a novel class incremental learning approach for personalized hashtag recommendations using Graph Convolutional Networks (GCNs), leveraging image content and trending topics. In order to simulate the dynamic nature of social media trends and to validate the adaptability of our model to changing contexts, we create an extension of the popular HARRISON dataset by adding a temporal component. We investigate our solution's sensitivity to different approches and availability of training samples. The results presented show that our model can effectively adapt in different scenarios, whether old data is available or not at each training iteration, also through the use of different correlation matrices to mitigate computational and memory load. As far as we know, this work is the first incremental learning attempt at hashtag recommendation for real-world images in social networks. We expect this benchmark to accelerate the advancement of hashtag recommendation.**

***Index Terms*—class incremental learning, graph convolutional network, hashtag recommendation**

I. Introduction

Social media platforms have had a significant impact on events and opinions on a wide range of topics. Social media platforms provide a broad view of user behavior. Users tend to converse with others who share their interests. Users can use hashtags to annotate their postings. Hashtags are abstract topic labels that are widely used in a variety of applications, including content categorization, organizing, navigation, and query expansion. Because hashtags are more inclusive and informative, many prior works have proved the effectiveness of hashtags in social data mining, such as information retrieval, sentiment analysis, event detection, and topic tracking. It has been observed that Instagram posts with even just one hashtag can boost user participation by 12.6% [8]. According to the same statistics, 95 million photographs are shared on Instagram daily. Only a few individuals tag their posted work with hashtags. A massive amount of contents is unnoticed due to their unwillingness or lack of knowledge. As a result, it has become critical to automate the hashtag recommendation process in order to identify relevant hashtags for social media posts.

There are two challenges related to automatic hashtag assignment: understanding the context, keeping the model updated. Understanding the context for automatic hashtag assignment involves grasping semantic, temporal, cultural, user-specific, conversational, and sentiment aspects to accurately assign hashtags to images. Keeping the model updated is vital and can be achieved through Class Incremental Learning (CIL), but it presents unique challenges for multi-label classification tasks like hashtag assignment. These challenges include concept drift, managing label combinations, avoiding overfitting, handling sparse labels, and developing continuous evaluation strategies for evolving datasets.

In this paper: I. We propose a deep incremental learning approach for personalized hashtag recommendation. II. We formulate the hashtag recommendation as a incremental classification problem, adopting the use of Graph Convolutional Network (GCN) [1]. III. We propose the use of a different correlation matrix, compared to [1], which only takes into account the new labels at each incremental stage in order to optimize computational and memory resources. IV. We propose a new method for training the model using only new data at each incremental stage, compared to the state-of-the-art methods which use both old and new data.

II. State of the Art

A. Hashtag Recommendation

Hashtag recommendation models can be divided in two main categories:

1) Feature-Based Approach: This approach delves into the content of social media posts to suggest relevant hashtags. It is divided into Content-Based and Personalized Hashtag Recommendation.
   Content-Based recommendation focuses on the text, images, and multimedia aspects of posts. For text, methods like topic and translation models or extracting information from external sources are employed [19].

979-8-3503-6213-8/24/$31.00 ©2024 IEEE

Image-centric approaches utilize Convolutional Neural Networks (CNNs) to extract features and suggest hashtags [18]. Additionally, there are approaches integrating both visual and textual information for more accurate suggestions [17]. However, these approaches might overlook individual user preferences.

In contrast, Personalized Hashtag Recommendation takes into account user tagging behavior, historical posts, and platform interactions to suggest hashtags tailored to individual users, enhancing user experience [16].

2) Method-Based Approach: This approach employs various methods for hashtag recommendation, including Similarity-Based, Topic Modeling-Based, Neural Network-Based, and Joint Modeling of Text and Image. Similarity-Based methods compare new posts with existing ones and suggest similar hashtags [20]. Topic Modeling-Based techniques, such as Latent Dirichlet Allocation (LDA), identify themes in posts to suggest relevant hashtags [21]. Neural Network-Based methods include classification-based, treating recommendation as a classification problem using deep learning models [17], and generation-based, employing recurrent neural networks (RNNs) to generate hashtags sequentially [22].

B. Class incremental Learning

The goal of Class Incremental Learning (CIL) is to create algorithms that can learn new concepts sequentially and finally perform well across all observable classes [10]. To extend a trained model to new classes, a considerable amount of labeled data from both new and old classes is required for network finetuning. Otherwise, if the previous class dataset is no longer available, fine-tuning a deployed model with new classes can result in catastrophic forgetting [11] - [13]. Catastrophic forgetting occurs when a model degrades performance on old classes after being retrained on new ones.

The problem of CIL has been handled using many ways, which can be classified into three major categories:

1) Replay methods: these works save samples of previous classes that are repeated when learning a new activity, hence reducing forgetfulness. The samples are either reused as model inputs for rehearsal or used to constrain the loss optimization for subsequent jobs.
2) Regularization-based methods: These works do not retain raw data, which reduces memory needs. An extra regularization term is inserted into the loss function, allowing you to learn new classes while keeping your previous knowledge.
3) Parameter isolation methods: The methods in this class use separate model parameters for each task.

The taxonomy is based on the works by Liu et al. [14] and Delange et al. [15], where additional references can be found.

III. Proposed Method

The CIL proposed in this work aims to simulate a real-world case in which the model can learn continuously and autonomously from new images and corresponding hashtags. Assuming a scenario where users continuously share new images and assign relevant hashtags to them. With Class Incremental Learning, the model can be updated in real-time, integrating new data without having to start from scratch.

For example, if a user uploads a sunset image with the hashtag #sunset, the model can learn from the image and its context to improve its recommendation capabilities. Furthermore, if new related hashtags are added in the future, such as #landscape or #nature, the model can integrate these new classes of images into its existing dataset without requiring a complete reprocessing.

On the other hand, some hashtags over time can become irrelevant or outdated, which is why it has been decided to maintain a fixed number of classes. Only the hashtags most used by users at a specific moment in time, t, are considered by the model.

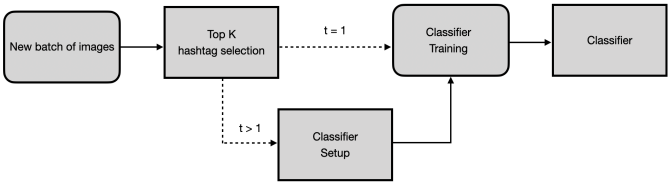

Fig. 1. Training pipeline devised for incremental hashtag assignment.

The CIL pipeline, visible in Fig. 1, is divided into three main stages:

1) Identification of the most frequent hashtags.
2) Classifier Setup.
3) Classifier Training.

A. Identification of the most frequent hashtags

At each iteration, new images are introduced, each with corresponding hashtags. Once the new data is obtained, the most frequent hashtags among the new images are determined.

B. Classifier Setup

Beginning with the second iteration, the model requires configuration. Since new classes may have become the most frequent or the order of those classes may have changed, the model's output layer must be updated accordingly to reflect the newly identified classes. This configuration allows the model to adapt to the new classes and retain the knowledge gained.

After loading the model, we extract the weights of the output layer, which corresponds to the first fully connected layer (fc). These weights represent the learned representations for each class in the previous iteration.

Next, we compute the average weights across all existing classes. This step involves computing the mean of the output layer weights along the class dimension, resulting in a vector of average weights.

We iterate over the new class order of the new iteration obtained from the dataset. For each new class absent in the previous iteration, we append the average weights computed in the previous step. This ensures that the model has initialized weights for any newly introduced classes.

With weights initialized for all classes in the new order, we reorganize the weights based on the updated class order. To achieve this, we align the weights with the model's class-to-index mapping, ensuring consistency between the weights and class labels.

Finally, we replace the original output layer weights with the updated weights. By doing so, the model's output layer is effectively updated to reflect the new class order, enabling it to make predictions according to the revised class distribution.

C. Classifier Training

For this stage we will use the Prediction Learning GCN (P-GCN) which was introduced by Zhao-Min Chen et al. [1] in 2021 to perform multi-label classification. The entire model framework is mainly composed of three modules, namely the Image Representation Learning module, the Label-Aware Modulation module and the GCN Based Image-Level Prediction Score Learning module.

Image Representation Learning. Given an image, a CNN is used to learn the representation of the image and global max-pooling is exploited to obtain at the image level the representation $x \in \mathbb{R}^D$.

Label-Aware Modulation. This module is used to decompose the global image representation x of an image into a set of label-specific characteristics. This is accomplished by first applying plain classifiers, which can be implemented as a fully-connected layer on x.

$$y_{cls} = f_{fc}(x; \theta_{fc}), \tag{1}$$

where $\theta_{fc} \in \mathbb{R}^{C \times D}$ denotes the parameters of the plain classifiers. Each classifier $\theta^c_{fc} \in \mathbb{R}^D$ extracts the information relevant to class c and predicts the emergence of class c in the image. Then, the disentangled label-relevant features are obtained via

$$Z_{lam} = repeat(x) \odot \theta_{fc} \in \mathbb{R}^{C \times D}, \tag{2}$$

where $repeat(x)$ denotes the operator which copies $x \in \mathbb{R}^D$ C times to form$[x, ..., x]^\top \in \mathbb{R}^{C \times D}$ and $\odot$ represents the Hadamard product. By modulating the image feature x in this way, the modulated feature $Z^c_{lam} \in \mathbb{R}^D$ captures the information associated with class c.

GCN Based Image-level Prediction Score Learning. With the disentangled image features associated to different classes, we use GCNs to map these features Z_{lam} into the prediction scores y_{gcn} for this image. Thanks to the information propagation in GCNs guided by the correlation matrix, the prediction scores tend to maintain the inter-class relationships. In details, for the first GCN layer, the inputs are the label-relevant representations Z_{lam}. For the last layer, the outputs are inter-dependent prediction scores $y_{gcn} \in R^C$. Eventually, we combine the prediction scores obtained from both the plain classifiers and the GCNs to serve as the final prediction scores for the image. That is

$$\hat{y} = y_{cls} + y_{gcn} \tag{3}$$

GCNs work by propagating information between nodes based on the correlation matrix. Thus, how to build the correlation matrix is a crucial problem for the P-GCN. The first matrix we built, called Augmented Correlation Matrix (ACM) was proposed by Kaile et al in 2022 [3].

In this configuration the training data D^t_{trn} and the ACM A^t for task t, are passed to the P-GCN classifier. At each completed task the model configuration is saved so that the soft labels $\hat{z}$ for the next task can be derived.
The use of ACM, however, presents some problems:

1) Exponential Growth: As new classes are added, the size of the ACM grows exponentially. This can lead to a significant increase in the memory required to store and manipulate such data.
2) Computational Complexity: Operations on the ACM can become computationally intensive and time-consuming. This can adversely affect the performance of the model.

Therefore, a second Correlation Matrix (CM) is built, inspired by Chen et al. [2]. The use of the CM should solve the two problems related to exponential growth and computational complexity since it will be calculated each time from zero only for the new classes that are present, thus having a fixed size at each task t.

In this second configuration, the training data D^t_{trn} and CM A^t for task t, are fed to the P-GCN classifier. At each new task t the CM will be computed again in order to model the dependence of the new trend classes. Once the training is completed, the weights and biases of the model will be saved for reuse at the next task.

IV. Experimental Setup

A. Dataset

Originally, the HARRISON dataset [7] consists of 57,383 images. Each image has an average of 4.5 associated hashtags (minimum 1 and maximum 10 associated hashtags). The ground truth hashtags for the images are made up of the 1,000 most frequently used hashtags.

However, in order to simulate the concept described at the beginning of Chapter III, we have randomly divided the dataset into 9 distinct subsets, one for each iteration during which the model will be trained. With each new subset, we have computed the top 50 most used hashtags, based on the frequency of occurrence within that subset, in order to mimic the concept of trending topics. Therefore, we are observing which hashtags are most popular within each subset we examine. With this new configuration the first iteration will use the subset consisting of 10,000 images. Each subsequent iteration will use subsets composed of approximately 5,900 images. Each image has an average of 4.2 associated hashtags (minimum 1 and maximum 8 associated hashtags).

The transformations used to augment the datasets, and to train the classifier are the following:

1) Geometric transformations: random affine, and random perspective.

2) Color transformations: sharpness adjustment, posterization, and color jitter.

The original image is transformed using a combination of one random geometric transformation and one random color transformation.

The training, validation and test sets for the classification problem are built from the individual classes with stratified sampling following this ratio: training set 70%, validation set 10%, test set 20%.

B. Comparative Analysis

Two types of comparisons were conducted. In the first comparison, the goal is to determine whether using the ACM yields better performance than using the CM. The second comparison assesses the performance of the model in two practical scenarios. The first scenario involves excluding all images used from previous training sessions in all subsequent sessions. Instead, the second scenario uses all available images.

a) Choice of correlation matrix: ACM vs CM: The purpose of this test was to determine whether it is worthwhile to use the ACM, since as the number of classes grows, both computational complexity and memory requirements increase exponentially. Therefore, if the results are equivalent or show only a slight improvement, the basic CM is preferred. Having chosen between the two matrices, we want to observe how the model performs as the number of iterations increases. Therefore, we wanted to see if there are any differences in the results between the first and the last iteration.

b) Real-world scenarios: NOD vs YOD: The purpose of this test was to simulate two real-world scenarios. In the first scenario, called No Old Data (NOD), it is assumed that the photos from iteration $t-1$ are no longer available at iteration t. This may be due to the fact that certain photos have been deleted for security, and privacy reasons, or events that have altered people's sensitivity to those images. In the second scenario, called Yes Old Data (YOD), it is assumed that at each iteration t, the images from iteration $t-1$ are also used in the model training. Unlike the previous test where the two matrices could be considered in competition to select the better model, the objective here is to observe how the model performs in the two aforementioned cases and whether it can be considered a flexible model. In the case of NOD, at iteration $t = 1$, there will be a batch of 10000 images available, and at each iteration t, 5900 new images will be available and only the new ones will be used. Similarly, in the case of YOD, at iteration $t =$, there will be an initial batch of 10000 images, but in this case, an additional 5900 images will be added at each iteration t.

C. Training configuration

The training pipeline consists of two main steps: the visual feature extractor, and the multi-label classifier. During the visual extraction step, two types of visual features are extracted from the input image. Next, the extracted features are used as the inputs for the Prediction Learning GCN, and the score of each class of hashtag is obtained. The scores are sorted, and the top-ranked hashtags are recommended for the input images. During the visual feature extraction phase, it is necessary to have high-level feature representation and diverse visual information to allow the model to recognize objects in a broader range of environments.

Our feature extractor, the Res-Net101 [4], was firstly trained on the 1.2 million images of the ImageNet dataset [5]. The visual features extracted from the final bottleneck layer of the previously mentioned model have 2048 dimensions, called ResNet-Object.

Subsequently, the feature extractor was trained on 2,5 million images of the Places-365 dataset [6]. The visual features extracted from the final bottleneck layer of the previously mentioned model have 2048 dimensions, called ResNet-Scene.

Lastly, inspired by [7] we decided to combine both the object-based and the scene-based visual features. The concatenation of the previous two visual features (N = 2048 × 2) is used as the input of the P-GCN. The visual extractor step took 21 minutes and 36 seconds to complete. The P-GCN step was trained for 100 iterations with 256 samples per batch using the MultiLabelSoftMarginLoss as Loss function. The learning rate was initially set to 10^{-3}, and then decreased by a factor of 10 every 30 iterations. Both steps, the visual extractor and the classifier, were trained using the open-source PyTorch framework on a single NVIDIA GeForce RTX 2070 Super and the training time was 47 minutes and 19 seconds in total.

D. Evaluation Measures

The parameters widely employed for assessing the performance of hashtag recommendation systems are hit rate, precision, recall, and F1-score. Hence, in this paper, we will use these evaluation metrics.

The *Hit Rate*@K is defined as 1 if at least one match between the top K-ranked hashtags and the ground truth hashtags exists:

$$\textit{Hit Rate}@K = \begin{cases} 1, & \text{if } \textit{Result(K)} \cap GT \neq 0 \\ 0, & \text{if } \textit{Result(K)} \cap GT = 0 \end{cases} \tag{4}$$

Precision@K is defined as the portion of hashtags in the top K-ranked hashtags that match with the ground truth hashtags.

Recall@K is defined as the portion of hashtags in the ground truth hashtags that match with the top K-ranked hashtags.

F1-score@K is defined as the harmonic mean of *Precision*@K and *Recall*@K.

In our experiments, we set K to 1 for precision and 5 for recall and hit rate, considering the average number of associated hashtags per image in the HARRISON dataset is 4.5.

Since in the incremental learning phase, the same model will be trained for 9 iterations, it is necessary to add two additional metrics for model evaluation: the mean and the standard deviation.

The *Mean*@K is calculated by adding up all the values @K and then dividing that sum by the total number of values in that set.

The *Standard Deviation*@K is calculated as the square root of the variance, which is the average of the squared differences between each data point and the mean.

The *Mean*@K and the *Standard Deviation*@K will be calculated for each previously described metric: *Hit Rate*@K, *Precision*@K, *Recall*@K and *F1-score*@K.

V. Experimental Results

A. ACM vs CM

Firstly, we evaluated the matrices' result on *Mean Hit Rate*@5, *Mean Precision*@1, *Mean Recall*@5, and *Mean F1-score*@5 by averaging over all images in the test set and across all nine iterations. We also have the standard deviation for each metric in order to understand the dispersion of the data. Tables I, II, and Fig. 2 show the evaluation results for both matrices on the HARRISON dataset. As shown in Table I, there is no substantial difference between the use of the two matrices, with the ACM obtaining only 2.5% more in terms of Hit rate, 1.5% more in terms of Recall, just 0.3% and 0.6% in terms of Precision and F1-score respectively. It is also important to note that, in the case of the ACM, the values exhibit a higher standard deviation and, therefore, greater variability. The base Correlation Matrix can be considered the best choice in terms of computational complexity and accuracy ratio.

TABLE I
MEAN RESULTS USING CM AND ACM ACROSS THE ITERATION.

Matrix	M Hit Rate@5	M Recall@5	M Precision@1	M F1-score@5
CM	0.495	0.306	0.144	0.122
ACM	0.520	0.321	0.147	0.128

TABLE II
STANDARD DEVIATION RESULTS USING CM AND ACM ACROSS THE ITERATION.

Matrix	SD Hit Rate@5	SD Recall@5	SD Precision@1	SD F1-score@5
CM	0.041	0.070	0.049	0.066
ACM	0.043	0.073	0.050	0.070

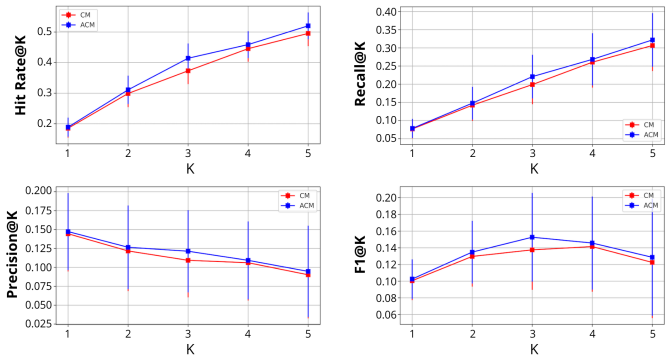

Fig. 2. Evaluation results using CM and ACM across the iteration as K increases.

Once we have identified the best matrix to use, we will focus on analyzing how our model's results vary based on the number of iterations. Therefore we will compare the results of training at iteration $t = 1$ with those of the last iteration, which is $t = 9$.

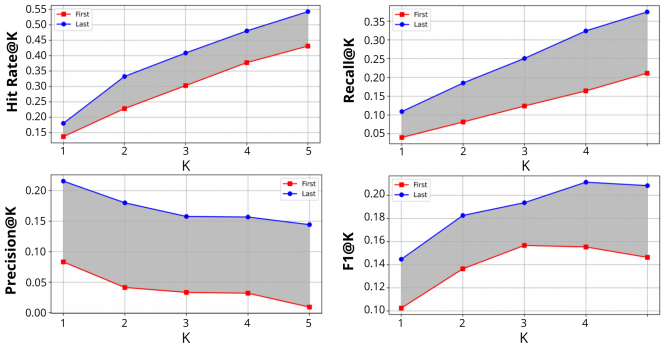

Fig. 3. Evaluation results for the first and last iteration with Hit Rate@K, Recall@K, Precision@K, and F1-score@K as K varies.

We evaluated the two iteration results on *Hit Rate*@5, *Precision*@1, *Recall*@5, and *F1-score*@5 by averaging over all images in the test set. Table III and Fig 3 show the evaluation results for the first and last iteration on the HARRISON dataset. As shown in Table III, the last iteration achieved better performance across all metrics, with an average Hit Rate@K of 54.2% compared to 43% of the first iteration. Also, in terms of Recall@K the last iteration obtained 16.3% more than the first iteration, 13.2% and 6% more in terms of Precision@K and F1-score@K respectively.

TABLE III
EVALUATION RESULTS FOR THE FIRST AND LAST ITERATION WITH HIT RATE@K, RECALL@K, PRECISION@K, AND F1-SCORE@K.

Iteration	Hit Rate@5	Recall@5	Precision@1	F1-score@5
First	0.430	0.211	0.083	0.146
Last	0.542	0.374	0.215	0.208

B. NOD vs YOD

As a final test, we want to verify how the model performs in case the data from iteration $t-1$ is available. We evaluated the YOD and NOD results on *Mean Hit rate*@5, *Mean Precision*@1, *Mean Recall*@5, and *Mean F1-score*@5 by averaging over all images in the test set and across all nine iterations. We also have the standard deviation for each metric to understand the variability of the data. Tables IV, V and Fig. 4 show the evaluation results for both approaches on the HARRISON dataset. As we can see in Table IV, there is no substantial difference in terms of results, both approaches can be considered effective in assigning hashtags. The only difference can be found in the sparsity of the result, as can we see in Table V, YOD results across iterations tend to be more stable having a lower standard deviation.

VI. Conclusion

We addressed the task of hashtag assigment by leveraging Graph Convolutional Networks with class-incremental learn-

TABLE IV
MEAN RESULTS USING NOD AND YOD APPROACHES ACROSS THE ITERATIONS.

Matrix	M Hit Rate@5	M Recall@5	M Precision@1	M F1-score@5
NOD	0.495	0.306	0.144	0.122
YOD	0.475	0.294	0.140	0.117

TABLE V
STANDARD DEVIATION RESULTS USING NOD AND YOD APPROACHES ACROSS THE ITERATIONS.

Matrix	SD Hit Rate@5	SD Recall@5	SD Precision@1	SD F1-score@5
NOD	0.041	0.070	0.049	0.066
YOD	0.021	0.036	0.024	0.036

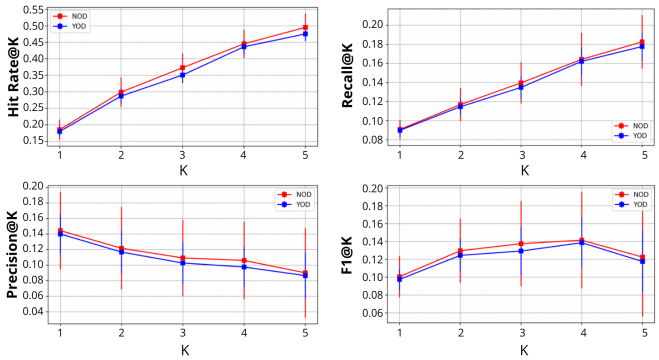

Fig. 4. Evaluation results using NOD and YOD approaches across the iteration as K increases.

ing, motivated by the observation that new hashtags are being constantly generated, thus requiring a flexible solution.

During the training phase, several experiments were conducted. The first experiment involved comparing the base correlation matrix to the augmented correlation matrix. The latter yielded slightly higher values compared to the former, but due to its computational complexity and exponential growth as the number of classes increased, the base correlation matrix was preferred. The second experiment compared using all available images at each iteration, known as Yes Old Data (YOD), to using only the images from the current iteration, known as No Old Data (NOD). Based on the conducted tests, the model performed well in both cases, producing similar results in both experiments. The only difference observed was that the model trained with YOD exhibited less variability in results between iterations. In any case, the model can be considered effective in both real-life scenarios.

One of the possible future developments will be to test how different configurations of dataset partitions can influence the results compared to the one used in this specific set of temporal experiments.

Acknowledgments

This work was partially supported by the MUR under the grant "Dipartimenti di Eccellenza 2023-2027" of the Department of Informatics, Systems and Communication of the University of Milano-Bicocca, Italy.

REFERENCES

[1] CHEN, Zhao-Min, et al. Learning graph convolutional networks for multi-label recognition and applications. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2021, 45.6: 6969-6983.
[2] CHEN, Zhao-Min, et al. Multi-label image recognition with graph convolutional networks. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2019. p. 5177-5186.
[3] DU, Kaile, et al. Class-incremental lifelong learning in multi-label classification. arXiv preprint arXiv:2207.07840, 2022.
[4] HE, Kaiming, et al. Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2016. p. 770-778.
[5] RUSSAKOVSKY, Olga, et al. Imagenet large scale visual recognition challenge. International journal of computer vision, 2015, 115: 211-252.
[6] ZHOU, Bolei, et al. Places: A 10 million image database for scene recognition. IEEE transactions on pattern analysis and machine intelligence, 2017, 40.6: 1452-1464.
[7] PARK, Minseok; LI, Hanxiang; KIM, Junmo. HARRISON: A benchmark on hashtag recommendation for real-world images in social networks. arXiv preprint arXiv:1605.05054, 2016.
[8] ZHANG, Suwei, et al. Hashtag recommendation for photo sharing services. In: Proceedings of the AAAI conference on artificial intelligence. 2019. p. 5805-5812.
[9] KIPF, Thomas N.; WELLING, Max. Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907, 2016.
[10] YAN, Shipeng; XIE, Jiangwei; HE, Xuming. Der: Dynamically expandable representation for class incremental learning. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2021. p. 3014-3023.
[11] SERRA, Joan, et al. Overcoming catastrophic forgetting with hard attention to the task. In: International conference on machine learning. PMLR, 2018. p. 4548-4557.
[12] ZHANG, Chi, et al. Few-shot incremental learning with continually evolved classifiers. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2021. p. 12455-12464.
[13] MCCLOSKEY, Michael; COHEN, Neal J. Catastrophic interference in connectionist networks: The sequential learning problem. In: Psychology of learning and motivation. Academic Press, 1989. p. 109-165.
[14] LIU, Yaoyao; SCHIELE, Bernt; SUN, Qianru. Adaptive aggregation networks for class-incremental learning. In: Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition. 2021. p. 2544-2553.
[15] DE LANGE, Matthias, et al. A continual learning survey: Defying forgetting in classification tasks. IEEE transactions on pattern analysis and machine intelligence, 2021, 44.7: 3366-3385.
[16] TRAN, Van Cuong; HWANG, Dosam; NGUYEN, Ngoc Thanh. Hashtag recommendation approach based on content and user characteristics. Cybernetics and Systems, 2018, 49.5-6: 368-383.
[17] ZHANG, Qi, et al. Hashtag Recommendation for Multimodal Microblog Using Co-Attention Network. In: IJCAI. 2017. p. 3420-3426.
[18] SIGURBJÖRNSSON, Börkur; VAN ZWOL, Roelof. Flickr tag recommendation based on collective knowledge. In: Proceedings of the 17th international conference on World Wide Web. 2008. p. 327-336.
[19] DING, Zhuoye, et al. Learning topical translation model for microblog hashtag suggestion. In: Twenty-third international joint conference on artificial intelligence. 2013.
[20] ZANGERLE, Eva; GASSLER, Wolfgang; SPECHT, Günther. On the impact of text similarity functions on hashtag recommendations in microblogging environments. Social network analysis and mining, 2013, 3: 889-898.
[21] GODIN, Fréderic, et al. Using topic models for twitter hashtag recommendation. In: Proceedings of the 22nd international conference on world wide web. 2013. p. 593-596.
[22] LI, Yang, et al. Hashtag recommendation with topical attention-based LSTM. In: Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers. 2016. p. 3019-3029.