Incremental Two-Stage Logo Recognition with Knowledge Distillation

Simone Bianco, Marco Buzzelli, Gianluca Giudice
Department of Informatics, Systems and Communication
University of Milano – Bicocca
Milan, Italy

***Abstract*—The recognition of logos can be useful in developing autonomous checkout systems, or monitoring brand presence and advertisement in shopping malls. The continuous generation and update of new brand logos imposes the definition of a flexible solution to the problem. We therefore define a two-stage logo recognition system composed of an agnostic logo detector, to locate image regions that possess generic logo-like characteristics, and an incremental logo classifier, to progressively update the set of known logo classes. We investigate our solution's sensitivity to regularization and availability of training samples, and we develop two alternative techniques for model compression. Results are presented and compared with state of the art solutions, showing promising results. Our code is made available for public download.**

***Index Terms*—Logo recognition, class incremental learning, knowledge distillation**

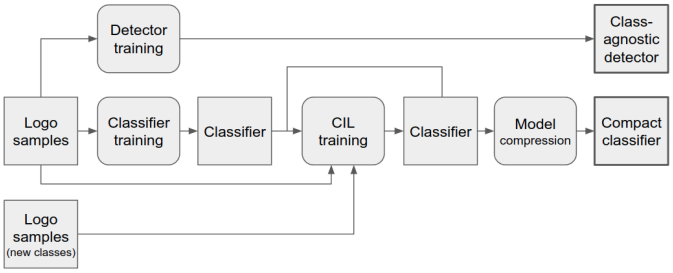

Fig. 1: Training pipeline devised for incremental logo recognition. Two different sources of training data are here exemplified: a first set of logo samples, and an update with "new classes".

I. Introduction

Logo recognition consists in locating and identifying instances of known logo classes in a digital image. The first studies in this field date back to 1993 [1], yet this task is becoming increasingly important in a variety of applications, with works aimed at helping the development of autonomous checkout systems in retail environments [2], developing video advertising systems [3], monitoring brand visibility [4], and protecting the intellectual property [5], while future applications might involve monitoring food items in smart home appliances [6]. There are several challenges in logo recognition, starting from the fact that a symbol composed by text and images can be considered a logo, but there is no formal definition of what a logo is. In fact, logos can be created from text using different typographic styles, any particular graphic consisting of many colors, or even a combination of the two. There is a large variety of logos, and this problem has both high intra-class and inter-class variations, since the same brand can have very different logos (e.g. only a stylized text version and a graphic version) and logos which belong to different brands might look very similar. Since many new brands are constantly being created and each brand has its own logo, it is necessary to develop systems that keep up with the creation of new logos. A method for logo detection and recognition should take into account this particular aspect of the problem and should properly recognize each new logo. For this reason, there is the need to develop a system which is able to adapt to these changes where standard closed-set classification techniques would fail. One possible approach could be to train a new classifier each time a new logo is created. However, this method is very inefficient and unfeasible for large scale datasets of logos. Moreover, retraining a model in such a way would require to store a large quantity of examples, since both the data from the previous logos and the new ones would be needed. Open-set logo recognition allows models to detect logos that are not available during the training phase. In this way, it is possible to overcome the problems discussed above. Open-set logo recognition can be addressed as a distance metric learning problem, where instances are identified by means of an advanced comparison with one or few examples per class, or as an incremental learning problem, where the system continuously learns to address new tasks from new data while preserving knowledge learned from previously learned tasks [7]. This latter way of learning, inspired from natural systems which are intrinsically incremental [8], is the approach that we adopt in this work, as highlighted in Figure 1.

In this paper: I. We define a two-stage logo recognition system, composed of agnostic logo detection and classification. II. We formulate the classification part as a incremental classification problem, adopting the Dynamically Expandable Representation approach [9]. III. We make training more robust by integrating dropout regularization and data augmentation. IV. We simplify the classification model considering both pruning with trainable masks, and knowledge distillation.

Source code available at: https://github.com/gianlucagiudice/logo-detection-recognition

II. State of the Art

A. Logo Recognition

A typical pipeline for logo detection and recognition consists in logo region proposal followed by a classifier specifically trained for logo classification, as proposed by Bianco et al. [10], [11]. Another approach presented by Wang et al. [12] involves a model based on YOLOv3 [13] to produce both bounding boxes and classification for each detected logo. Their proposed model is called Logo-YOLO, which introduces into YOLOv3 a modification to the loss function and the recomputation of the anchors sizes. The modified loss function utilizes the Focal Loss to solve the problem of the logos which are small objects to the background, and the Complete-IoU loss [14] to obtain more accurate and faster regression of the bounding boxes.

The issue with these approaches is the closed-world assumption which does not apply in the case of logo recognition. Fehérvári et al. [15] propose a method based on Distance Metric Learning (DML) using deep learning techniques called SoftTriple Loss presented in [16]. Another work based on DML has been presented by Li et al. [17] and can be considered as an extension to [15]. In this work the authors enrich the latent space learned by DML with text features contained in the logos. The authors highlight how a large number of logos have remarkable amount of text or (stylized) letters, for this reason, in addition to visual features, they consider text features as relevant information for logo classification.

B. Object Detection

This task can be defined as follows: given an image, determine whether or not there are instances of a predefined set of objects, usually referred to as classes, and, if present, return the location of each instance [18]. The spatial location of an object in an image can be represented using bounding boxes. Object detection was initially addressed using handcrafted features and shallow trainable architectures. As described in [19], the frameworks of object detection methods can mainly be categorized into two groups:

1) Two-stage Object Detection: generates region proposals at first and then classifies each proposal into different object categories.
2) One-stage Object Detection: adopts a unified framework to achieve final results (categories and locations) directly.

In this paper we propose a solution based on two stages.

C. Class Incremental Learning

The problem of Class Incremental Learning (CIL) aims to design algorithms that can learn new concepts in a sequential way and eventually perform well on all observed classes [9].

To extend a trained model on new classes, a large amount of labeled data for both new and old classes is necessary for network finetuning. Otherwise, if the dataset of old classes is no longer available, finetuning a deployed model with new classes can lead to the catastrophic forgetting problem [20]–[22]. Catastrophic forgetting means that a model degrades performance on old classes when retrained on new ones.

The problem of CIL has been addressed using different methods, which can be divided into three main categories:

1) Replay methods: these works store samples of old classes which are replayed while learning a new task, by doing so it is possible to alleviate forgetting. The samples are either reused as model inputs for rehearsal, or to constrain the optimization of the loss on the new tasks.
2) Regularization-based methods: these works do not store raw data, therefore reducing the memory requirements. An extra regularization term is introduced in the loss function, in this way it is possible to learn new classes while maintaining the previous knowledge.
3) Parameter isolation methods: methods in this class use different model parameters for each task.

The taxonomy is based on the works by Liu et al. [23] and Delange et al. [24], where additional references can be found.

III. Proposed method for incremental logo recognition

The proposed system consists of two stages, performed by corresponding deep learning models as shown in Figure 1:

1) Object Proposal for logo detection, performed by a class-agnostic logo detector.
2) Classification for logo recognition, performed by a CIL classifier.

Additionally, "Model compression" is also considered during training of the classification stage.

A. Class-Agnostic Logo Detector

Our class-agnostic logo detector is based on YOLOv5-v6.1 [25], which can be formulated in different versions, depending on the size of the model and the size of the input image. In general, versions with more parameters perform better, with the disadvantage of longer training time, longer inference time and more computational resources required for training. On the other hand, a small model might achieve very low performance, thus misleading the final evaluation of the system that heavily relies on the detector. Therefore, a model that is too small could act as a bottleneck for the whole system. The class-agnostic logo detector used in this paper is based on YOLOv5m6, with an image input size of 512×512px and pre-trained on the COCO dataset [26].

Unlike the classification stage, the detection stage is not strictly dependent on an update to consider new classes. We hypothesize that the concept of what a generic logo is can be learned and approximated adequately by using only an initial set of logos, and that the subsequent introduction of new classes will not disrupt the general idea of a logo. Nonetheless, a degree of dependency on an updated set of classes can be expected, which we explore in Section V-A.

B. Incremental Logo Classifier

As a basis for incremental logo classification, we rely on the Dynamically Expandable Representation algorithm (DER) [9], which structures the training procedure in different "tasks", starting from an initial set of classes, and introducing new

classes with each task. The algorithm can be divided as follows:

1) Representation learning: the architecture of the Convolutional Neural Network (CNN) is dynamically expanded at each new incremental learning step, using a limited memory to preserve classes from previous steps.
2) Masking and pruning: the CNN introduced in the last incremental learning step is pruned using a channel-level masked-based method.
3) Classifier learning: the classifier is retrained by integrating a limited number of training examples from previous steps, with samples from the new classes.

A partial implementation of the DER algorithm is provided in the PyCIL repository [27] , and it is used as a starting point for the development of our CIL logo classifier. Starting from the original architecture, a dropout layer [28] is added before the fully-connected layer in order to avoid overfitting.

C. Model Compression

The DER algorithm addresses the problem of class incremental learning by expanding the neural network architecture at each incremental step. The super-feature-extractor Φ_t for images $\mathbf{x}$ at step t of incremental learning is given by the following concatenation:

$$\Phi_t(\mathbf{x}) = [\mathcal{F}_0, \mathcal{F}_1(\mathbf{x}), ..., \mathcal{F}_t(\mathbf{x})]. \quad (1)$$

Each feature extractor $\mathcal{F}_i$ is a CNN, which in this paper is implemented using ResNet-34 [29]. The parameters of the neural network grow linearly as a function of the number of incremental learning steps. For this reason, there is a need to reduce the number of parameters of the neural network.

The authors of DER adapted the concept of model pruning with trainable masks [20], where the super-feature-extractor is pruned after the training of each step t. This is done using a differentiable channel-level mask-based method to prune filters of the extractor $\mathcal{F}_t$, in which the masks are learned jointly with the representation. The mask is then binarized after the learning procedure, and the feature extractor $\mathcal{F}_t$ is pruned using the binary mask, producing as output the pruned network $\mathcal{F}_t^P$. This first solution to model compression was implemented in the context of our work, since it was not originally available within the DER implementation.

A second approach to tackle the problem of the large number of model parameters is Knowledge Distillation (KD) [30], in which a smaller model (labelled "student") is trained to mimic a larger one ("teacher"). The distillation loss used to train the student model is computed with the Kullback-Leibler divergence considering both the logits produced in output by the teacher (soft targets) and the real data labels (hard targets). The probability p_i of each class i, is computed by the teacher model using its logit z_i as follows:

$$p_i^{(\{t,s\})} = \left(exp\left(\frac{z_i^{(\{t,s\})}}{T} \right) \right) \Big/ \left(\sum_j exp\left(\frac{z_i^{(\{t,s\})}}{T} \right) \right), \quad (2)$$

Fig. 2: Number of objects in LogoDet-3K for each class.

where temperature T is used to smooth the probability distribution revealing hidden inter-class relationships. Equation 2 computes both the class probability predicted by the student ($p_i^{(s)}$), and that predicted by the teacher ($p_i^{(t)}$: the soft targets).

IV. Experimental setup

A. Dataset

The LogoDet-3K dataset [12] consists of ∼3,000 logo classes, 158,652 images and 194,261 logo objects. The objects correspond to the bounding boxes in each image, and since an image can contain more than one logo (not necessarily of the same class), the number of objects is greater than the number of images. The dataset is divided into nine categories, where each category contains several brands and for each brand there is a set of images. In this context, a brand is intended as a logo class. In fact, the same brand can have multiple versions of logos, for example, a symbolic logo and a textual logo. These types of logos are treated as different classes in the LogoDet-3K dataset. Figure 2 shows the distribution of the number of objects for each logo class. An evident issue of LogoDet-3K, representative of a real-world situation, is the low number of objects for some classes. Specifically, 25% of the classes have a number of images lower or equal to 27, and half of the classes have a number of images lower or equal to 54. Since several classes have a low number of images, it could be challenging for the model to correctly predict these classes. To this extent, we adopt online data augmentation. The transformations used to augment the original dataset, and to train the CIL classifier are the following:

- Geometric transformations: random affine, and random perspective.
- 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.

A first random selection of 1,000 classes is used for the initial classification task, followed by 8 steps of incremental learning, each adding 250 new classes. 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%. The splits for the detection problem are defined coherently, by considering the training labels for the detector only if that bounding box corresponds to a Region of Interest (ROI) used as a training example for the classifier, otherwise the bounding box is used in the validation or test set.

B. Training configuration

The class-agnostic logo detector is trained on the LogoDet-3K dataset for 30 epochs with a batch size of 16 using the Stochastic Gradient Descent (SGD) optimizer with Nesterov momentum set to 0.937. L2 regularization is used to avoid overfitting, with the weight decay parameter set to $5 \cdot 10^{-4}$. As training data, only those 1,000 classes available for the first task are used to train the detector.

The training of the CIL model on LogoDet-3K is performed for 200 epochs at the first step, and for 150 at subsequent incremental steps. Early stopping is used to avoid overfitting of the model: the number of epochs with no improvement in accuracy on the validation set after which training will be stopped is set to 30 (i.e. patience); the minimum change in the monitored accuracy to be considered as an improvement is set to 0.5%. To update the model weights two different optimizers are tested: SGD and Adam [31]. In both cases, an exponential decay scheduling is applied to the learning rate: once training reaches one of the predetermined milestone epochs, the learning rate of the optimizer is multiplied by a factor $\gamma = 0.1$. The milestones are set to 60, 100 and 150 for the initial learning step, and to 40, 75 and 100 for each incremental learning step.

V. Experimental results

The problem formulation consists of 1,000 classes used as the initial task, then at each incremental step 250 classes are added for the corresponding task.

A. Detection experiments

The detector is trained on the first 1,000 classes of the initial task, then evaluation metrics are computed on the test set of these 1,000 classes combined with the remaining 1,993, thus covering the entire dataset. Results are shown in Table I, offering a reference comparison between the detector trained on 1,000 classes and the one trained on all the 2,993 classes. This is done to asses the generalization capabilities obtained by the detector trained on 1,000 classes. Mean Average Precision (mAP) is used to evaluate object detection models based on Intersection over Union (IoU), Recall, and Precision. It is calculated as the weighted average of precision values at each threshold of accepted IoU, while the weight is the increase in recall from the prior threshold. Given k different classes, mAP is the average of the AP values among each class.

Although the performance of the detector based on 1,000 classes can be considered acceptable, the gap in mAP@.5 and mAP@.5:.95 between the detector trained on 1,000 classes and the one trained on 2,993 classes is not negligible. This result is an important aspect for the proposed system, and suggests the need to also investigate the topic of incremental logo detection in future developments.

TABLE I: Precision, Recall, mAP@.5 and mAP@.5:.95 obtained by the detector trained on 1,000 classes, and that trained on 2,993 classes. The test set is composed of all 2,993 classes.

# Training classes	Precision	Recall	mAP@.5	mAP@.5:.95
1,000	0.704	0.666	0.706	0.465
2,993	0.833	0.846	0.890	0.648

B. Classification experiments

We performed preliminary experiments on a subset of 100 classes, where 30 are used for the initial task, then the remaining 70 classes are incrementally added 10 at a time. From these experiments we selected the following optimal configuration: a ResNet-34 backbone pretrained on the ImageNet dataset, data augmentation, 0.5 dropout rate (tested against 0.3 and 0.1), and Adam otpimizer (tested against SGD).

Table II reports results on the full dataset composed of 2,993 classes. Top-k accuracy is used to evaluate the classification performance, focusing on cases where $k = 1$ and $k = 5$. We provide a baseline having the same computational complexity (in terms of architecture and number of parameters) of the last step of CIL, but trained to directly classify all 2,993 classes, which achieves 90.22% top-1 accuracy.

As a first set of experiments for CIL, an important factor for testing the scalability of the model is the number of examples stored for old classes. In particular 50, 20 and 10 are the memory sizes taken into account. Since some classes do not have enough training samples to reach the available budget, we also report for reference the total memory across all classes. The results quantify the relatively small impact that memory size has on overall accuracy, going from 87.20% for 29K samples, to 89.22% for 102K samples. The top-1 accuracy of the baseline is only slightly better than that of the best CIL model, proving that the DER algorithm is an effective approach for achieving CIL.

The second set of experiments is relative to model compression via pruning. When pruning is used in combination with Weight Aligning (WA), the performance drops dramatically. Therefore, only for models using pruning, WA is disabled. Despite producing promising results in the preliminary experiments with 100 classes, in the case of 2,993 classes pruning yields poor performance: even considering the case where 50 samples are used for the pruning model, performance deteriorates by almost 20%.

The subsequent experiment is relative to adopting a KD approach to model compression, transferring the knowledge of a teacher network trained in a CIL setup to a student network. Here, a ResNet-50 model pretrained on ImageNet is used as the backbone for the student, with a dropout layer before the fully-connected layer. In order to replicate a realistic situation, the student model is trained under the supervision of the teacher, but only the samples stored by the teacher are used as training data. As can be seen in table, the number of samples

TABLE II: Logo classification performance with class-incremental learning on the 2,993 classes of the LogoDet-3K dataset, for different training configurations. Symbols ♢ ♣ ♠ ♡ are used to identify experiments that are referenced later on.

Solution	Memory (class)	Memory (total) (K)	Weight align.	Compression	# Params (M)	Top-1 acc. (%)	Top-5 acc. (%)
Baseline	(all)	(all)	no	none	205.0	90.22%	94.36%
#1 ♢	10	29	yes	none	205.0	87.20%	92.19%
#2	20	53	yes	none	205.0	88.47%	92.58%
#3 ♣	50	102	yes	none	205.0	89.22%	92.97%
#4	50	102	yes	pruning	70.7	7.60%	10.67%
#5	10	29	no	pruning	68.8	51.69%	62.56%
#6	20	53	no	pruning	71.7	63.20%	73.97%
#7 ♠	50	102	no	pruning	68.3	70.37%	81.85%
#8	10	29	yes	KD	29.6	73.52%	82.49%
#9 ♡	50	102	yes	KD	29.6	88.96%	93.57%

TABLE III: Performance of the full-recognition system using the class-agnostic logo detector and different CIL classifiers. Precision, Recall, mAP@.5 and mAP@.5:.95 are computed on the test set composed of all the 2,993 classes.

Solution	Precision	Recall	mAP@.5	mAP@.5:.95
CIL-10 ♢	0.545	0.591	0.583	0.397
CIL-50 ♣	0.578	0.615	0.614	0.417
CIL-50-pruned ♠	0.496	0.485	0.489	0.347
CIL-50-KD ♡	0.558	0.604	0.590	0.403
Logo-Yolo [12]	-	-	0.523	-
SeeTek [17]	-	-	0.705	-

used to train the student heavily affects the final performance. Using 50 samples per class to train the student results in top-1 accuracy of 88.96%, i.e. an increase of 15% compared to using only 10 samples per class. KD clearly outperforms the pruning compression, both in terms of parameters and accuracy. More importantly, we are able to achieve very similar performance to the baseline that requires availability of the full training set, at a fraction of the network parameters: 29.6 million against 205.0 million (i.e. about 14% of the parameters).

C. Full recognition experiments

In this last section, the whole system is tested. This consists in combining the two stages: the class agnostic logo detection which generates ROIs, and the actual classification performed by the CIL classifier. The final performance is evaluated considering the Precision, Recall, mAP@.5 and mAP@.5:.95 on the test set composed of all the 2,993 classes.

The model used for the first stage of the system is the class-agnostic logo detector, trained with only the 1,000 classes used for the first task by the CIL classifier. For what concerns the CIL classifier the following models are compared, with references from Table II:

1) CIL classifier with memory 10 (♢).
2) CIL classifier with memory 50 (♣).
3) CIL classifier with memory 50, pruned (♠).
4) Student trained with KD with memory 50 (♡).

Results are shown in Table III. As a comparison, the performance reported in the literature by Wang et al. in Logo-Yolo [12] and by Li et al. in SeeTek [17] are also presented. The proposed approach using KD outperforms Logo-Yolo but achieves lower performance than SeeTek. For reference, SeeTek is an open-set retrieval approach and not an incremental learning approach. In an open-set retrieval approach, the classification of an input logo is performed by assigning the class of the nearest logo in a latent space. By doing so, if an input logo is similar, but does not actually match the training images, it is still recognized, but the system will not be able to integrate new knowledge to recognize new logos.

As shown from the results, the full recognition performance obtained by SeeTek (70.46% mAP@.5) is very similar to that obtained by the proposed class-agnostic logo detector evaluated without taking into account the class (70.60% mAP@.5 from Table I). This further suggests that, even without the errors introduced by the classification stage, the performance obtained by the classification stage are suboptimal, and can be considered a current bottleneck of our solution.

Figure 3 presents some visual examples of the full system for logo recognition. The tested images are not from LogoDet-3K, in order to simulate an out-of-dataset real world scenario. The ZARA example in particular shows the correct detection (and classification) of two instances of the logo, while the instance in the middle is not detected. This is possibly due to its lack of contrast, which we can assume was learned by the class-agnostic logo detector to be a discriminative feature. The precision-oriented nature of the detector allowed to avoid false positives, such as the label in the shop window.

VI. Conclusions

We addressed the task of logo recognition by formulating the problem with a class-incremental learning approach, motivated by the observation that new logo classes are being constantly generated, thus requiring a flexible solution. Our two-stage system for logo recognition is composed of a class-agnostic logo detector, and a classifier trained using the incremental learning approach, later compressed via knowledge distillation. Rigorous experiments show that we are able to achieve classification performance close to a baseline that requires availability of the full training set, at a fraction of the network parameters (i.e. about 1/7). In terms of full-recognition results, we outperform an existing solution based

Fig. 3: In-the-wild visual examples of the proposed two-stage logo recognition system with knowledge distillation.

on the YOLO one-stage detector, while underperforming against a solution based on distance-metric learning based on textual image features. This type of image description could be considered for integration in our own solution as a direction for future developments. Our experiments also suggest that the current bottleneck lies in the detection stage. This could be expanded in terms of computational complexity, as well as introducing incremental learning capabilities.

REFERENCES

[1] David S Doermann, Ehud Rivlin, and Isaac Weiss, "Logo recognition using geometric invariants," in *Proceedings of 2nd International Conference on Document Analysis and Recognition (ICDAR'93)*. IEEE, 1993, pp. 894–897.

[2] Cristina Mata, Nick Locascio, Mohammed Azeem Sheikh, Kenny Kihara, and Dan Fischetti, "Standardsim: A synthetic dataset for retail environments," in *International Conference on Image Analysis and Processing*. Springer, 2022, pp. 65–76.

[3] Zhi-Qi Cheng, Xiao Wu, Yang Liu, and Xian-Sheng Hua, "Video ecommerce++: Toward large scale online video advertising," *IEEE transactions on multimedia*, vol. 19, no. 6, pp. 1170–1183, 2017.

[4] Yue Gao, Yi Zhen, Haojie Li, and Tat-Seng Chua, "Filtering of brand-related microblogs using social-smooth multiview embedding," *IEEE Transactions on Multimedia*, vol. 18, no. 10, pp. 2115–2126, 2016.

[5] Xuan Jin, Wei Su, Rong Zhang, Yuan He, and Hui Xue, "The open brands dataset: Unified brand detection and recognition at scale," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 4387–4391.

[6] Marco Buzzelli, Federico Belotti, and Raimondo Schettini, "Recognition of edible vegetables and fruits for smart home appliances," in *2018 IEEE 8th International Conference on Consumer Electronics-Berlin (ICCE-Berlin)*. IEEE, 2018, pp. 1–4.

[7] Marc Masana, Xialei Liu, Bartlomiej Twardowski, Mikel Menta, Andrew D Bagdanov, and Joost van de Weijer, "Class-incremental learning: survey and performance evaluation on image classification," *arXiv preprint arXiv:2010.15277*, 2020.

[8] Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu, "Large scale incremental learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 374–382.

[9] Shipeng Yan, Jiangwei Xie, and Xuming He, "Der: Dynamically expandable representation for class incremental learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 3014–3023.

[10] Simone Bianco, Marco Buzzelli, Davide Mazzini, and Raimondo Schettini, "Logo recognition using cnn features," in *International Conference on Image Analysis and Processing*. Springer, 2015, pp. 438–448.

[11] Simone Bianco, Marco Buzzelli, Davide Mazzini, and Raimondo Schettini, "Deep learning for logo recognition," *Neurocomputing*, vol. 245, pp. 23–30, 2017.

[12] Jing Wang, Weiqing Min, Sujuan Hou, Shengnan Ma, Yuanjie Zheng, and Shuqiang Jiang, "Logodet-3k: A large-scale image dataset for logo detection," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 18, no. 1, pp. 1–19, 2022.

[13] Joseph Redmon and Ali Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[14] Zhaohui Zheng, Ping Wang, Wei Liu, Jinze Li, Rongguang Ye, and Dongwei Ren, "Distance-iou loss: Faster and better learning for bounding box regression," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020, vol. 34, pp. 12993–13000.

[15] István Fehérvári and Srikar Appalaraju, "Scalable logo recognition using proxies," in *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2019, pp. 715–725.

[16] Qi Qian, Lei Shang, Baigui Sun, Juhua Hu, Hao Li, and Rong Jin, "Softtriple loss: Deep metric learning without triplet sampling," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 6450–6458.

[17] Chenge Li, István Fehérvári, Xiaonan Zhao, Ives Macedo, and Srikar Appalaraju, "Seetek: Very large-scale open-set logo recognition with text-aware metric learning," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2022, pp. 2544–2553.

[18] Li Liu, Wanli Ouyang, Xiaogang Wang, Paul Fieguth, Jie Chen, Xinwang Liu, and Matti Pietikäinen, "Deep learning for generic object detection: A survey," *International journal of computer vision*, vol. 128, no. 2, pp. 261–318, 2020.

[19] Zhong-Qiu Zhao, Peng Zheng, Shou-tao Xu, and Xindong Wu, "Object detection with deep learning: A review," *IEEE transactions on neural networks and learning systems*, vol. 30, no. 11, pp. 3212–3232, 2019.

[20] Joan Serra, Didac Suris, Marius Miron, and Alexandros Karatzoglou, "Overcoming catastrophic forgetting with hard attention to the task," in *International Conference on Machine Learning*. PMLR, 2018, pp. 4548–4557.

[21] Chi Zhang, Nan Song, Guosheng Lin, Yun Zheng, Pan Pan, and Yinghui Xu, "Few-shot incremental learning with continually evolved classifiers," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 12455–12464.

[22] Michael McCloskey and Neal J Cohen, "Catastrophic interference in connectionist networks: The sequential learning problem," in *Psychology of learning and motivation*, vol. 24, pp. 109–165. Elsevier, 1989.

[23] Yaoyao Liu, Bernt Schiele, and Qianru Sun, "Adaptive aggregation networks for class-incremental learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 2544–2553.

[24] Matthias Delange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Ales Leonardis, Greg Slabaugh, and Tinne Tuytelaars, "A continual learning survey: Defying forgetting in classification tasks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[25] Glenn Jocher et. al., "ultralytics/yolov5: v6.1 - TensorRT, TensorFlow Edge TPU and OpenVINO Export and Inference," Feb. 2022, https://github.com/ultralytics/yolov5 (Accessed on July 15th, 2022).

[26] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*. Springer, 2014, pp. 740–755.

[27] Fu-Yun Wang, "G-U-N/PyCIL: PyCIL: A Python Toolbox for Class-Incremental Learning," 2021, https://github.com/G-U-N/PyCIL (Accessed on July 15th, 2022).

[28] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[29] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[30] Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al., "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, vol. 2, no. 7, 2015.

[31] Diederik P Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.