Density Map Distillation for Incremental Object Counting

Chenshen Wu
Computer Vision Center
Barcelona, Spain
chenshen@cvc.uab.es

Joost van de Weijer
Computer Vision Center
Barcelona, Spain
joost@cvc.uab.es

Abstract

We investigate the problem of incremental learning for object counting, where a method must learn to count a variety of object classes from a sequence of datasets. A naïve approach to incremental object counting would suffer from catastrophic forgetting, *where it would suffer from a dramatic performance drop on previous tasks. In this paper, we propose a new exemplar-free functional regularization method, called Density Map Distillation (DMD). During training, we introduce a new counter head for each task and introduce a distillation loss to prevent forgetting of previous tasks. Additionally, we introduce a cross-task adaptor that projects the features of the current backbone to the previous backbone. This projector allows for the learning of new features while the backbone retains the relevant features for previous tasks. Finally, we set up experiments of incremental learning for counting new objects. Results confirm that our method greatly reduces catastrophic forgetting and outperforms existing methods.*

1. Introduction

Image-based counting aims to infer the number of people, vehicles or any other objects present in images. It has a wide range of applications such as traffic control, environment survey and public safety. Most of existing research focus on learning a model from a single dataset. Only [3] and [20] propose to train a model on multiple datasets simultaneously in a multi-task setting. In this paper, we propose a method to incrementally learn to count new objects or to count in a new domain. This has the advantage that it does not require collecting data on a single server for training. Moreover, annotators can focus on just labelling instances of a single class (typically annotated with a single point), which reduces the annotation effort required for adding new classes.

Continual learning (CL) addresses the problem of training a model from a non-stationary distribution. It is important because data in the real-world might not be jointly available (e.g. due to data privacy or legislation). Moreover, often the previous data cannot be revisited due to the privacy or storage restrictions. Researchers have explored continual learning in many tasks, e.g. classification [15, 16], segmentation [2, 8], and object detection [30]. However, continual learning for counting systems, has to the best of our knowledge, not yet been studied.

One of the main challenges of continual learning is catastrophic forgetting. After training on new data, models tend to forget the knowledge extracted from previous data. In the past few years, people tried to alleviate this issue by using replay examples [26, 38], expand networks [40] and regularization [15, 16]. As one of the most promising methods, regularization can be further categorized as weight regularization [15] and data (or functional) regularization [16]. The former applies regularization on weights to prevent them from drifting too far from the old model, while the latter apply it on the output of the network given the input data. Due to their success for the classification tasks [22], the fact that they do not require exemplars, and because they scale well with the number of tasks, we will here explore data regularization for object counting.

However, these methods are mainly designed for the classification problem, which aims to predict a category for a given sample. For the counting problem, which is a regression problem where the output is a scalar map, we found that directly applying these existing CL method is suboptimal. Therefore, we propose a new method called *Density Map Distillation* (DMD). For each new object, we train a separate counter head that maps the feature extraction backbone to an object-specific density map. After the training of each task, the counter head is fixed and during future tasks only the feature extractor and new counter head is trained. When training a new task, we use the new data to apply distillation on all previous counter heads. Since the feature extractor is drifting when learning new tasks, we propose to use a cross-task adaptor to project the new features to the old features. This mechanism allows us to keep plasticity while maintaining stability (i.e., prevent forgetting).

The contributions of this paper include: (1) We set up

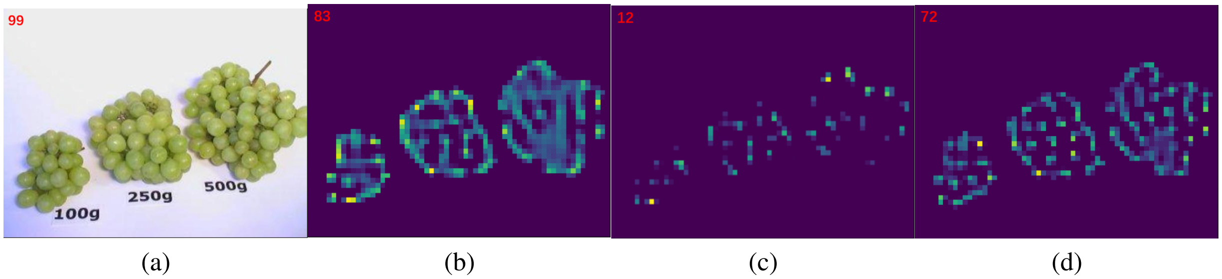


Figure 1. (a) Input Image (b) Density map after training only on the *grapes* data. Density map after learning two additional tasks (*tomatoes* and *strawberries*) with (c) Fine Tuning (FT) and (d) after learning by our proposed method (DMD). In the upper-left corner we show the ground truth number of grapes in (a) and the estimation of the algorithms respectively. Note that naive fine-tuning leads to catastrophic forgetting and the method loses its ability to count *grapes*. (d) Our method manages to get a considerable better count prediction even though there is some performance loss.

experiments for incremental learning for counting new objects. We define metrics for evaluating incremental counting problems. (2) We propose Density Map Distillation (DMD) for the incremental counting problem. The method includes fixing the task-specific counter head and training a cross-task adaptor for the feature extractor. Our method prevents forgetting, while maintaining plasticity to learn new tasks. (3) We adapt several existing methods of incremental learning for incremental object counting. Experiments show that our new methods outperforms these existing methods.

2. Related Work

2.1. Incremental Learning

Incremental learning aims to develop methods that can learn new knowledge from new data while not forgetting previous knowledge learned from the previous training stages. The existing methods can mainly be categorized as three types: distillation based, dynamic model based and rehearsal based [5]. Distillation based methods focus on how to limit the change of the model by applying a loss on the weights directly [1, 15], or on the output features [14] and probabilities [16]. Dynamic model based methods [40] extend the architecture of the network to learn new knowledge from the new incoming data distribution. Rehearsal based methods [26] save a few exemplars from the previous dataset and replay them or use them to constrain the model during the new training sessions.

Previously, incremental learning mainly focused on image classification problems. Recently, the community also developed incremental learning algorithms for other problems such as image generation [37], segmentation [2], object detection [24], video classification [23]. But to the best of our knowledge, there is no work for incremental learning of counting problems yet.

2.2. Crowd Counting

There are two categories of crowd counting methods, density map based methods [21, 35] and localization based methods [31].

Localization based methods count by locating each individual's position. Some methods [17, 27] are driven by an object detector, and inaccuracy is introduced by estimation of the ground truth bounding boxes. Liu et al. [18] propose RAZ-Net that recurrently detect high density regions and zoom in for re-inspection. The network performs the counting and localization task at the same time, and these two tasks complement each other. Song et al. [31] propose the P2PNet that predicts the localization points directly by introducing a one-to-one matching strategy from the prediction to the ground truth.

For the localization based method, it is hard to predict each location where the crowd density is very high [34]. Most of the research in counting mainly focuses on predicting a density map and then count by the summing it. In [28, 41], they propose to use several parallel CNNs of different sizes to address the problem of scale variation. Another line of research focuses on the loss function. In [21], Ma et al. propose a Bayesian loss to measure the distance between the predicted and the ground truth density map. Wang et al. [35] measure the similarity between the predicted density map to the ground truth density map by solving an Optimal Transport (OT) problem.

In the above methods, the model is always trained with one dataset. In [3] and [20], they propose to train a model on multiple datasets simultaneously. Some other researches [11, 13, 36] focus on counting problem in domain adaptation setting, where the model is trained on the source dataset and the label of the target dataset is limited. In [10], Gao et al. deal with the problem of domain incremental learning of crowd counting, where a model is trained to count people sequencially on several datasets.

In [19, 25, 29], they consider the problem of class-agnostic counting. The aim is to train a network that counts the number of instances in an image by specifying an exemplar patch.

3. Method

Counting is an integral part of many real-life applications. To alleviate the human costs of manual counting, many methods have been developed for the counting of objects [21, 31, 35]. As discussed in the introduction, these methods generally assume that all training data is jointly available. However, for many applications this assumption is not realistic and the algorithm would only be able to have access to a batch of data at each time step.

A naive approach to learning from a sequence of tasks would be to just continue finetuning the model on the available data of consecutive tasks. However, this would lead to the *catastrophic forgetting* phenomenon. An illustration of this is provided in Figure 1 where we show that after learning several tasks with fine-tuning, the method has lost its ability to count the first-task *grapes* class. In this section, we explore distillation-based methods for incremental learning of object counting to prevent the effect of catastrophic forgetting.

3.1. Notation

In a typical counting problem, images X_i are annotated for a single object class $c \in C$, for example annotations of persons, cars, or apples are given. Existing works do not consider counting various classes of objects simultaneously. Typically, objects are annotated with a single point in the center of the object at positions $p_{ij}, j \in 1, ..., N_i$ where there are N_i objects for the image x_i. We will use the notation P_i to refer to the set of locations in image x_i. Object counting learns a model for a single object class that given an input image maps to a density map which predicts the number of object instances per pixel [21, 35] or which directly predicts the object coordinates [31].

In incremental learning for counting problems, the data is split in various tasks, where each task $t \in [1, T]$ arrives sequentially. For each task, the dataset $D_t = \{c_t, \left(\left(x_1, P_1\right), \left(x_2, P_2\right), \cdots, \left(x_M, P_M\right)\right)\}$ contains the class category c_t and images with ground truth position annotations. We consider the scenarios where each task has a single object category c_t different from the other tasks. After training on all T tasks, the model is evaluated on a test set Y that contains images of all objects C seen in the various tasks. The task-ID of the test images is available to the algorithm at inference time (this setting is also known as task-incremental learning) [32].

For the training of the object counting network, we propose to use a network which can be divided into a feature extractor $f : R^{w \times h \times 3} \rightarrow R^{w_d \times h_d \times d}$ where d is the number of output channels of the feature extractor, and an object-specific *counter head* given by $h : R^{w_d \times h_d \times d} \rightarrow R^{w_d \times h_d \times 1}$. The counter head maps from the feature space to a density map. The prediction of a network for an image x is then given by:

$$\hat{y} = \sum_{w=1}^{w_d} \sum_{h=1}^{h_d} \hat{d}(x) = \sum_{w=1}^{w_d} \sum_{h=1}^{h_d} h \circ f(x) \tag{1}$$

where $\hat{d} = h \circ f$ is the predicted density map and the summation is over the spatial coordinates of the density map.

For training the new task, we use the loss proposed by Wang et al. [35]:

$$\mathcal{L}_{\text{train}} = \left| \left\| d \right\|_1 - \left\| \hat{d} \right\|_1 \right| - \lambda_1 \mathcal{W} \left(\frac{d}{\left\| d \right\|_1} - \frac{\hat{d}}{\left\| \hat{d} \right\|_1} \right) + \lambda_2 \frac{1}{2} \left\| \frac{d}{\left\| d \right\|_1} - \frac{\hat{d}}{\left\| \hat{d} \right\|_1} \right\| \tag{2}$$

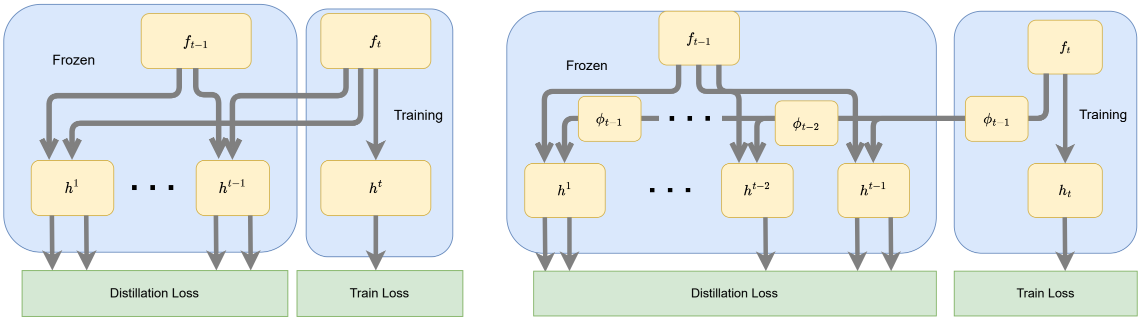
The first term is the counting loss for the final counting number. The second term is the optimal transport loss, where $\mathcal{W}$ is the Monge-Kantorovich's Optimal Transport (OT) cost [33]. The third term is the Total Variation (TV) loss, and λ_1 and λ_2 are the hyperparameters for the OT and TV losses.

To extend the above described method to incremental object counting, we use the following notations. The network contains a feature extractor after learning task t given by f_t. For each of the learned tasks, we have a task specific counter head h_t for each object. At the beginning of the task, the feature extractor f_t is initialized from the previous feature extractor f_{t-1}. The previous feature extractor f_{t-1} is then fixed and stored. Other older feature extractors like f_{t-2} are not stored.

When training task t we use h_t^τ to refer to the previous counter heads for the object that was learned at task τ. At inference time, we combine the last feature extractor with any of the previously learned counter heads, so for example to get the solution for class c_τ after training task t we apply $h_t^\tau \circ f_t$. We also consider fixing the previous task specific counter, i.e. we do not update it when learning new tasks, so $h_\tau^\tau = h_{\tau+1}^\tau = \cdots$, and we simply refer to it as h^τ.

3.2. Data regularization for regression problems

One of the popular approaches to prevent *catastrophic forgetting* in continual learning is by means of regularization methods [5]. Compared with the other two main approaches to continual learning, regularization methods have the advantage over rehearsal methods that they do not require the storage of any data from previous tasks, and they do not have an increased memory footprint when training



(a) Density Map Distillation without Adapter

(b) Density Map Distillation

Figure 2. (a) Density Map Distillation (DMD) without Adaptor. While training new tasks, the distillation loss is applied on the output density map using the previous counter heads, between the previous and the new feature extractors. Different from LwF [16], previous counter heads are fixed when training new task. (b) Density Map Distillation (DMD). In addition to the distillation loss, we train cross-task adaptors (ϕ) to project new features to old features, since the feature extractor is continuously trained.

on larger task sequences like isolation methods typically have. Regularization methods can be differentiated in data (or functional) and weight regularization methods.

Data regularization for classification networks is proposed by [16] and it is one of the most popular methods for exemplar-free continual learning. Different from the weight regularization methods [1, 15] which apply the regularization loss on the parameters of the network, data regularization apply the regularization on the output of network layers. Other than weight regularization, it is dependent upon the data on which the distillation is applied. This idea has been further extended by [6, 14]. The former apply the regularization loss on the feature output and the output after the cosine normalization. The latter apply them on several intermediate layers and study various marginalizations to improve the plasticity of the method.

However, the most popular data regularization method, LwF [16], cannot be applied directly to the counting problem. In LwF [16], Li et al. proposed to apply a knowledge distillation loss between the new and the old output. Given the image from the new dataset as the input, both models give a prediction of the probability and a cross entropy loss is applied as a regularization. However, an object counting network does not output a probability, and therefore the cross entropy loss cannot be applied. An adaption to the counting problem is to apply a L_2 loss on the density map:

$$\mathcal{L}_{\text{reg}} = \sum_{\tau\in[1,t-1]} \left\| h_t^\tau \circ f_t(x) - h_{t-1}^\tau \circ f_{t-1}(x) \right\|_2 . \tag{3}$$

We will identify this method with Learning without Forgetting (LwF) in our results section. However, we found that such an adaptation leads to suboptimal results. We hypothesize that this method suffers from overfitting.

Another typical data regularization method is to apply regularization on the feature level according to:

$$\mathcal{L}_{\text{reg}} = \sum_{\tau\in[1,t-1]} \| f_t(x) - f_{t-1}(x) \|_2 . \tag{4}$$

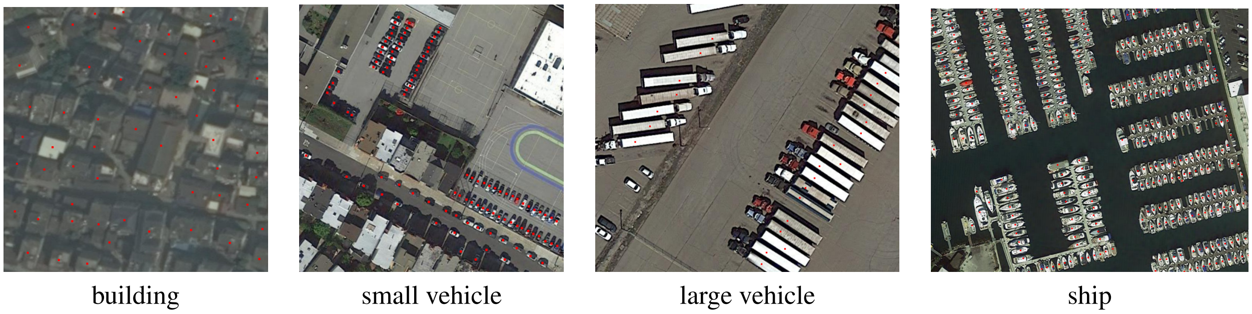
We call this method Feature Distillation (FD) [6, 14]. This prevents the feature extractor from drifting too far from the old one. This regularization is very restrictive since it requires the whole feature map to remain similar. So it is often too rigid so that the model cannot learn from new tasks. This was also observed by PODNet [6].

3.3. Density Map Regularization with Cross-Task adaptors

To address the shortcomings of data regularization for regression tasks, and to prevent the overfitting of the previous counter heads, we propose a further adaptation. After training of each task, the counter head for this task will be fixed. So the notation h_t^τ (the counter head for task τ during or after the learning of the task t) can be simplified as h^τ, because the counter head is not changed after the training, and hence $h_\tau^\tau = h_{\tau+1}^\tau = \cdots$. We also store the previous feature extractor f_{t-1} as a reference for the regularization loss. Earlier feature extractor are not needed, so the memory requirement does not scale linearly with the number of tasks. Then we apply the following regularization loss on the density map output from the old and new models:

$$\mathcal{L}_{\text{reg}} = \sum_{\tau\in[1,t-1]} \| h^\tau \circ f_t(x) - h^\tau \circ f_{t-1}(x) \|_2 . \tag{5}$$

This method is an exemplar-free method, since images from previous tasks are not used. As shown in Figure 2.a, both old and new feature extractors use the same image $x \in D_t$ as input and extract a feature $f_{t-1}(x)$ and $f_t(x)$. We use L_2 distance as the regularization loss. It is applied to the output of each counter head for all previous tasks $h_1, \cdots, h_{t-1}$,



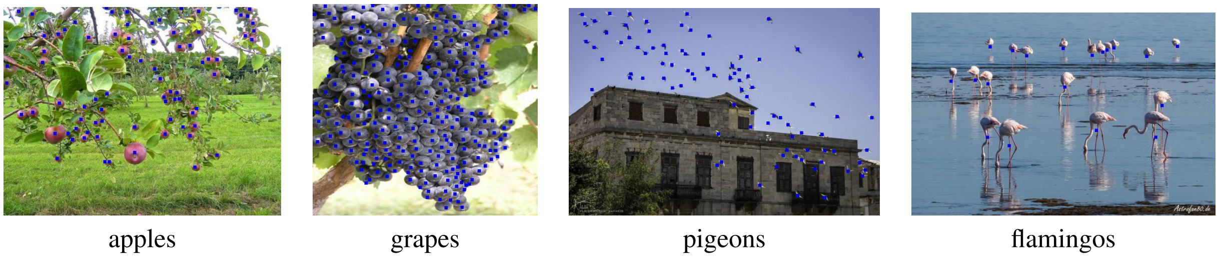
building

small vehicle

large vehicle

ship

Figure 3. Sample images from RSOC dataset [9].



apples

grapes

pigeons

flamingos

Figure 4. Sample images from FSC147 dataset [25].

which encourages the new model output to yield the same result when counting previous objects.

As we fixed the previous counter head, this might prevent the feature extractor from learning new knowledge. Therefore, in addition, we propose to train an adaptor ϕ to project the features from the new feature extractor to the old one. The adaptor is trained together with the feature extractor using the distillation loss. As illustrated in Figure 2.b, when training task t, the adaptor ϕ_{t-1} projects the features generated by f_t to approximate those by f_{t-1}. Similarly, by cascading several previous adaptors $\phi_{t-2}, \cdots, \phi_1$, the features can be projected to those in earlier stages. So the distillation loss with adaptor is given by:

$$\begin{aligned}\mathcal{L}_{\text{reg}} = \sum_{\tau \in [1, t-1]} & \| h^{\tau} \circ \phi_{\tau} \circ \cdots \circ \phi_{t-1} \circ f_t(x) \\ & - h^{\tau} \circ \phi_{\tau} \circ \cdots \circ \phi_{t-2} \circ f_{t-1}(x) \|_2 . \end{aligned} \tag{6}$$

We call our method *density map distillation* (DMD). To identify, the version defined by Eq. 5 without the adaptor, we will use the name *DMD w/o Adapt*).

The final loss is given by:

$$\mathcal{L} = \mathcal{L}_{\text{train}} + \lambda \mathcal{L}_{\text{reg}}, \tag{7}$$

where λ is the hyperparameter to balance the training loss and the regularization loss.

During the inference, the feature is extract by the new feature extractor f_t. To count the object c_τ, the feature needs to be adapted through all the adaptors learned after that task, $\phi_{t-1}, \phi_{t-2}, \cdots, \phi_\tau$. Then counter head h^τ uses the adapted feature to predict the density map for the given object according to:

$$\hat{d}(x) = h^{\tau} \circ \phi_{\tau} \circ \cdots \circ \phi_{t-1} \circ f_t(x). \tag{8}$$

The learning of adaptors between backbone networks in continual learning has been studied recently for continual self-supervised learning [7, 12], and more recently for supervised tasks in [4]. Other than them, we here study their usage for a regression task.

4. Experimental Results

4.1. Dataset and evaluation

Datasets. The RSOC dataset is a counting dataset of aerial images proposed by [9] involving *buildings*, *small vehicles*, *large vehicles*, and *ships*. In this paper, we will consider learning to count these classes incrementally in the before mentioned order. The images of buildings are collected from Google Earth, while the rest are from the DOTA dataset [39]. The DOTA dataset is an object detection dataset of aerial images. The original labels of bounding boxes are replaced by their central location for the counting problem. There are 2468 images for buildings, 280 images for small vehicles, 172 images for large vehicles and 137 images for ships

The FSC147 [25] dataset is a counting dataset for few-shot learning, containing 147 categories. For most categories, there are less than 100 images per category. For

Dataset:	**building**			**small vehicle**			**large vehicle**			**ship**			**Avg**
Metric:	MSE	MAE	NMAE	MSE	MAE	NMAE	MSE	MAE	NMAE	MSE	MAE	NMAE	NMAE
FT	31.10	27.05	0.926	1266.89	430.81	0.609	52.34	38.70	0.624	86.40	59.51	0.296	0.614
LwF	14.21	10.70	0.360	1326.12	505.10	1.000	79.65	62.78	1.000	137.76	108.27	0.499	0.715
FD	**10.79**	**7.48**	**0.263**	1108.06	356.01	0.379	39.16	27.17	0.423	122.86	88.16	0.382	0.362
EWC	10.94	7.58	0.268	1150.12	360.78	0.383	39.75	27.33	0.418	117.46	80.90	0.345	0.352
MAS	11.07	7.71	0.271	1068.67	333.95	0.380	40.23	27.75	0.419	117.94	85.17	0.375	0.361
DMD w/o Adapt	13.36	9.90	0.336	**929.75**	**291.62**	**0.288**	33.92	22.52	0.345	130.56	87.13	0.384	0.338
DMD	12.63	9.25	0.315	988.63	320.97	0.315	**25.78**	**16.53**	**0.269**	**107.84**	**76.40**	**0.367**	**0.316**

Table 1. Performance of several incremental learning methods after learning four tasks of RSOC dataset. In **bold** we show the best results for each column excluding the FT method.

FSC-fruits													
Dataset:	**grapes**			**tomatoes**			**strawberries**			**apples**			**Avg**
Metric:	MSE	MAE	NMAE	MSE	MAE	NMAE	MSE	MAE	NMAE	MSE	MAE	NMAE	NMAE
FT	67.11	53.25	0.671	28.49	20.28	0.362	28.11	21.18	0.312	12.85	7.89	0.119	0.366
LwF	27.29	19.74	0.260	63.09	49.54	1.000	73.45	61.02	1.000	**16.50**	**9.77**	**0.135**	0.599
FD	17.26	11.99	0.149	16.44	12.76	0.319	19.83	13.79	0.225	29.35	15.19	0.179	0.218
EWC	17.91	12.40	0.154	17.86	14.07	0.328	21.42	15.16	0.252	22.03	13.69	0.190	0.231
MAS	**16.12**	**11.37**	**0.142**	15.79	12.19	0.298	19.23	13.31	0.212	27.32	15.61	0.211	0.216
DMD w/o Adapt	25.35	18.28	0.240	14.29	11.50	0.260	**16.09**	11.25	**0.177**	23.96	12.33	0.141	0.207
DMD	26.63	18.83	0.250	**11.35**	**8.86**	**0.221**	16.27	**11.09**	0.178	18.56	10.84	0.140	**0.197**
FSC-birds													
Dataset:	**flamingos**			**pigeons**			**cranes**			**geese**			**Avg**
Metric:	MSE	MAE	NMAE	MSE	MAE	NMAE	MSE	MAE	NMAE	MSE	MAE	NMAE	NMAE
FT	74.55	37.77	0.587	41.54	27.12	0.617	12.79	7.63	0.218	7.90	3.54	0.102	0.381
LWF	66.35	27.73	0.326	60.12	42.93	1.000	54.75	32.30	1.000	10.59	5.21	0.153	0.620
FD	64.01	24.85	0.246	27.42	15.30	0.350	13.14	7.09	0.192	13.19	6.78	0.198	0.247
EWC	**62.90**	**23.94**	0.233	23.33	12.06	0.284	12.76	6.49	0.163	12.25	6.41	0.193	0.217
MAS	63.38	24.10	**0.226**	25.07	13.70	0.308	12.96	6.64	0.168	12.04	6.03	0.178	0.220
DMD w/o Adapt	67.35	28.19	0.330	25.96	11.60	0.204	7.78	4.41	0.128	10.72	5.56	0.166	0.207
DMD	67.21	28.66	0.354	**22.52**	**10.56**	**0.198**	**7.23**	**4.16**	**0.123**	**9.01**	**4.37**	**0.133**	**0.202**

Table 2. Performance of several incremental learning methods after learning four tasks on FSC-fruits and FSC-birds. In **bold** we show the best results for each column excluding the FT method.

our incremental learning, we chose several categories that contain a significant number of images. To better share the knowledge across the learning process, we select similar categories for the learning sequence. We consider two sequences of counting of four tasks. The first sequence, called *FSC-fruits*, contains *grapes*, *tomatoes*, *strawberries* and *apples*, containing 116, 117, 126 and 165 images, respectively. The second sequence, called *FSC-birds*, is *flamingos*, *pigeons*, *cranes* and *geese*, containing 76, 81, 108 and 162 images, respectively.

Evaluation Metric. Following previous methods, we use rooted Mean Squared Error (MSE), Mean Absolute Errors (MAE) and mean Normalized Absolute Errors (NAE) as metric to evaluate the performance of the model.

Mean Squared Error (MSE) is defined as:

$$\mathrm{MSE} = \frac{1}{N}\sum_{i=1}^{N} \|\hat{y} - y\|_2 \,, \tag{9}$$

where $\hat{y}$ is the predicted count number, y is the ground truth count number and N is the size for the testset.

Mean Absolute Errors (MAE) is defined as:

$$\mathrm{MAE} = \frac{1}{N}\sum_{i=1}^{N} \|\hat{y} - y\|_1 \,, \tag{10}$$

and mean Normalized Absolute Errors (NAE) is defined as:

$$\mathrm{NAE} = \frac{1}{N}\sum_{i=1}^{N} \frac{\|\hat{y} - y\|_1}{y}. \tag{11}$$

When evaluating the average performance of all the dataset, we use NAE because its values can be compared over datasets which have varying number of objects in them.

4.2. Implementation Details

Our implementation is based on the official code of DM-Count [35], and we follow its setting for training on a single dataset. The feature extractor is the convolutional layers of VGG19 with 512 output channels. The counter head contains of two 3×3 convolutional layers with 256 and 128 output channels respectively and a 1×1 convolutional layers with 128 output channels. The adapter is a one-layer 1×1 convolutional layer with the same number of channels. We train the model with the Adam optimizer, using batch size 10, learning rate 1e-5, weight-decay 1e-4 and beta 0.9 and 0.999. For each stage, we train the model for 1000 epochs and for the next stage the training is started from the previous model. The hyperparameters $\lambda = 100$ for RSOC dataset and $\lambda = 10$ for both FSC-fruits and FSC-birds.

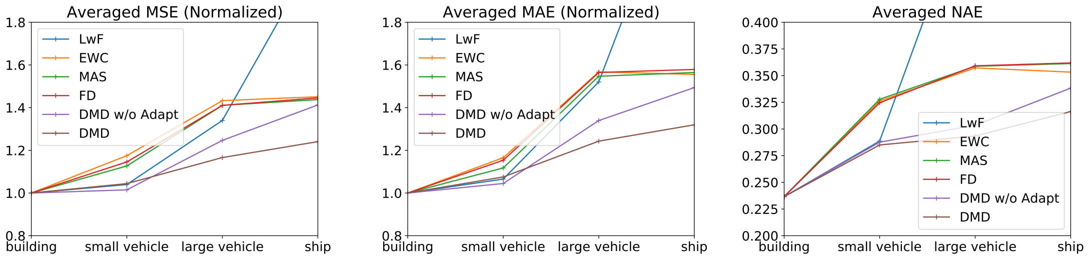


Figure 5. Result for RSOC (satellite) Dataset. The performances are evaluated after training of each task. We report the averaged value for all the previously seen tasks. The value is normalized to remove the dataset-scale. Lower value indicate better performance.

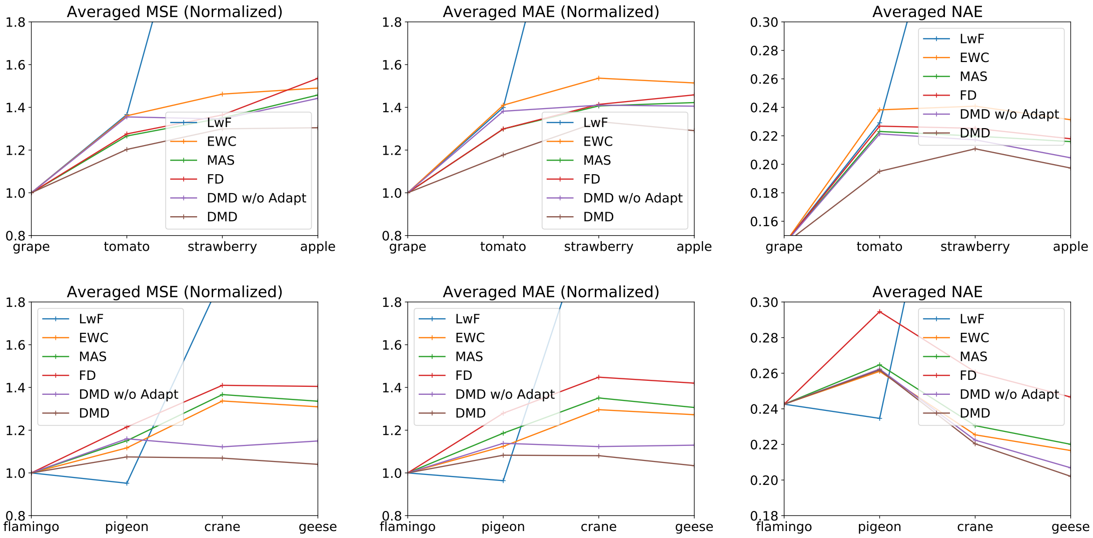


Figure 6. Averaged performance for FSC147 Dataset. The performances are evaluated after training of each task. We report the averaged value for all the previously seen tasks. The value is normalized to remove the dataset-scale. Lower value indicate better performance.

4.3. Results on satellite images

For satellite images, we train our model with four classes *buildings*, *small vehicles*, *large vehicles* and *ships* in sequence from the RSOC dataset. Table 1 shows the performance at the end of the incremental learning process, after training all four classes. The performance is evaluated in three metric: MSE, MAE and NMAE. Smaller values indicates better performance. The average performance of the four classes is evaluated with NMAE because of dataset-scale invariance of the NMAE metric.

Finetuning (FT) achieves the best performance on the last task and worst on the first task, as expected. Feature Distillation (FD), EWC [15] and MAS [1] show a similar pattern: they are good at remembering the first task, but have difficulties to learn subsequent tasks. However, they also often perform good in the last task. This might be because the *ships* class is more similar to the first task of *buildings* when comparing to the middle *vehicle* tasks. LwF [16] performs good on the first task. But it fails in the second and third task due to its very flexible counter head.

Our method DMD w/o Adapt improved the result compared with existing methods. The good performance in the second and the third task shows that it can learn new task while not forgetting the previous one. After adding the adaptor for the feature extractor, our method DMD further improved the performance, especially on the last task. In conclusion, the proposed density map distillation obtains around a 4% improvement over the best weight regularization method (EWC).

Figure 5 presents additional results, including the averaged MSE and MAE after learning each task, in addition to Table 1. For example, the scores reported at task 2 in the graph are the average of normalized MSE obtained on *building* and *small vehicle*) based on the network after training task 2. In the figure, we can observe that the parameter regularization methods EWC and MAS significantly outperform the FT baseline. Next, we observe that our method

Dataset:	**grapes**			**tomatoes**			**strawberries**			**apples**			**Avg**
Metric:	MSE	MAE	NMAE	MSE	MAE	NMAE	MSE	MAE	NMAE	MSE	MAE	NMAE	NMAE
DMD w/o Adapt(1)	24.35	15.85	0.200	17.15	13.38	0.302	18.82	14.70	0.266	31.26	16.47	0.195	0.241
DMD w/o Adapt(3)	25.35	18.28	0.240	14.29	11.50	0.260	16.09	11.25	0.177	23.96	12.33	0.141	0.207
DMD(1)	22.82	15.38	0.196	15.40	11.70	0.266	17.76	13.52	0.243	32.43	17.01	0.192	0.224
DMD(3)	26.63	18.83	0.250	11.35	8.86	0.221	16.27	11.09	0.178	18.56	10.84	0.140	0.197

Table 3. Ablation of varying number of layers in the counter head on the FSC-fruits sequence.

Dataset:	**grapes**			**tomatoes**			**strawberries**			**apples**			**Avg**
Metric:	MSE	MAE	NMAE	MSE	MAE	NMAE	MSE	MAE	NMAE	MSE	MAE	NMAE	NMAE
DMD w/o Adapt(10)	25.35	18.28	0.240	14.29	11.50	0.260	16.09	11.25	0.177	23.96	12.33	0.141	0.207
DMD w/o Adapt(100)	22.47	15.00	0.190	14.98	11.25	0.274	18.14	12.35	0.210	23.65	14.23	0.194	0.217
DMD(10)	26.63	18.83	0.250	11.35	8.86	0.221	16.27	11.09	0.178	18.56	10.84	0.140	0.197
DMD(100)	22.03	15.09	0.193	14.93	11.26	0.272	17.56	11.69	0.196	27.44	15.50	0.190	0.213

Table 4. Ablation of different regularization hyperparameter for regularization on FSC-fruits.

DMD w/o Adapt obtains significantly better results, especially for averaged NAE. Next, we see that for only two tasks, the proposed DMD method does perform similarly to DMD w/o Adapt. However, for more tasks, DMD does significantly better, and outperforms all methods after four tasks.

4.4. Results of counting fruits and birds

We consider two incremental learning sequences based on the FSC147 dataset. The first one, FSC-fruits, contains the following tasks *grapes*, *tomatoes*, *strawberries*, and *apples*. The second one, FSC-birds, considers the consecutive tasks of *flamingos*, *pigeons*, *cranes*, and *geese*. Table 2 summarize the results on FSC-fruits and FSC-birds.

Similar to the result in RSOC dataset, Finetuning (FT) achieves the best performance on the last task and forgets previous tasks. LwF [16] gives relatively good result in the first and the last task, but failed in the second and third task. MAS [1] and EWC [15] give the best result in the first task in FSC-fruits and FSC-birds respectively, but they fail to learn new tasks. Feature Distillation (FD) also performs similarly. FD and MAS work slightly better in FSC-fruits, and EWC works better in FSC-birds.

Our method DMD w/o Adapt improves the result over the above-mentioned methods. Especially, it gets better performance in the new tasks, on both the FSC-fruit and FSC-bird sequence. DMD further improves the result than DMD w/o Adapt, with the feature translation by the adaptor. In FSC-fruits, the performance drops slightly in the first task *grapes* and improves by a large margin in the second task *tomatoes*, compared with DMD w/o Adapt. In FSC-birds, the performance improves in both *pigeons* (second) and *geese* (last) tasks.

Figure 6 shows the averaged performance after training of each task. In FSC-fruits, our method DMD outperforms other methods. In FSC-birds, both DMD and DMD w/o Adapt outperform other existing method with a large margin after the third task.

4.5. Ablation Study

Layers of the counter head. We study the effect of using different layers for the counter head in FSC-fruits benchmark. For the comparison, the size of the total network is fixed, so to increase the size of the counter head means that we move few layers from the feature extractor to the counter head. Table 3 shows the result of our methods, DMD w/o Adapt and DMD, with 1 or 3 counter head layers. It shows that compared with 1 layer, using 3 layers for the counter head achieves a better performance on newer tasks and a better overall performance.

Hyper parameter for regularization. We study the effect of the hyperparameter for regularization in FSC (fruits) benchmark. Table 3 shows the result of DMD w/o Adapt and DMD. With higher regularization, the performance for the latter tasks drop because it is too rigid for learning the new task and the model remembers the first task better.

5. Conclusions

We studied incremental learning for the object counting problem. The main challenge is to prevent forgetting while learning to count new object categories for new tasks. We propose an exemplar-free method, called Density Map Distillation (DMD). For counting each object, we train a new counter head and all tasks share a feature extractor. We propose to fix the task counter and apply a distillation loss computed with new data on the output of the old counter head. To adapt the changed feature extractor for the fixed counter head, we introduced an adaptor to project the new output feature to the old one. Experiments shows that our method DMD w/o Adapt outperforms those methods adapted from continual learning for classification problems. And with the adaptor, our DMD further improves the performance.

Acknowledgement. We acknowledge the support of the Spanish Government funding for projects PID2019-104174GB-I00 and TED2021-132513B-I00.

References

[1] Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *ECCV*, pages 139–154, 2018. 2, 4, 7, 8

[2] Fabio Cermelli, Massimiliano Mancini, Samuel Rota Bul'o, Elisa Ricci, and Barbara Caputo. Modeling the background for incremental learning in semantic segmentation. In *CVPR*, 2020. 1, 2

[3] Binghui Chen, Zhaoyi Yan, Pengyu Li, Biao Wang, Wangmeng Zuo, and Lei Zhang. Variational attention: Propagating domain-specific knowledge for multi-domain learning in crowd counting. In *ICCV*, 2021. 1, 2

[4] Marco Cotogni, Fei Yang, Claudio Cusano, Andrew D Bagdanov, and Joost van de Weijer. Gated class-attention with cascaded feature drift compensation for exemplar-free continual learning of vision transformers. *arXiv preprint arXiv:2211.12292*, 2022. 5

[5] 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 TPAMI*, 2021. 2, 3

[6] Arthur Douillard, Matthieu Cord, Charles Ollion, Thomas Robert, and Eduardo Valle. Podnet: Pooled outputs distillation for small-tasks incremental learning. In *ECCV*, pages 86–102. Springer, 2020. 4

[7] Enrico Fini, Victor G Turrisi da Costa, Xavier Alameda-Pineda, Elisa Ricci, Karteek Alahari, and Julien Mairal. Self-supervised models are continual learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9621–9630, 2022. 5

[8] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Int. Conf. Machine Learning*, pages 1126–1135, 2017. 1

[9] Guangshuai Gao and Qingjie Liu. Counting from sky: A large-scale dataset for remote sensing object counting and a benchmark method. *IEEE Transactions on geoscience and remote sensing*, 2020. 5

[10] Jiaqi Gao, Jingqi Li, Hongming Shan, Yanyun Qu, James Z. Wang, Fei-Yue Wang, and Junping Zhang. Forget less, count better: a domain-incremental self-distillation learning benchmark for lifelong crowd counting. *Frontiers of Information Technology &amp Electronic Engineering*, 24(2):187–202, feb 2023. 2

[11] Junyu Gao, Yuan Yuan, and Qi Wang. Feature-aware adaptation and density alignment for crowd counting in video surveillance. *IEEE Transactions on Cybernetics*, 2020. 2

[12] Alex Gomez-Villa, Bartlomiej Twardowski, Lu Yu, Andrew D Bagdanov, and Joost van de Weijer. Continually learning self-supervised representations with projected functional regularization. In *Conference on Computer Vision and Pattern Recognition Workshop (CVPRW)*, 2022. 5

[13] Tao Han, Junyu Gao, Yuan Yuan, and Qi Wang. Focus on semantic consistency for cross-domain crowd understanding. In *ICASSP*, 2020. 2

[14] Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. Learning a unified classifier incrementally via rebalancing. In *CVPR*, pages 831–839, 2019. 2, 4

[15] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017. 1, 2, 4, 7, 8

[16] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE TPAMI*, 40(12):2935–2947, 2017. 1, 2, 4, 7, 8

[17] Dongze Lian, Jing Li, Jia Zheng, Weixin Luo, and Shenghua Gao. Density map regression guided detection network for rgb-d crowd counting and localization. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1821–1830, 2019. 2

[18] Chenchen Liu, Xinyu Weng, and Yadong Mu. Recurrent attentive zooming for joint crowd counting and precise localization. In *CVPR*, June 2019. 2

[19] Erika Lu, Weidi Xie, and Andrew Zisserman. Class-agnostic counting. In *ACCV*, 2018. 3

[20] Zhiheng Ma, Xiaopeng Hong, Xing Wei, Yunfeng Qiu, and Yihong Gong. Towards a universal model for cross-dataset crowd counting. In *ICCV*, 2021. 1, 2

[21] Zhiheng Ma, Xing Wei, Xiaopeng Hong, and Gong Yihong. Bayesian loss for crowd count estimation with point supervision. In *ICCV*, 2019. 2, 3

[22] Marc Masana, Xialei Liu, Bartlomiej Twardowski, Mikel Menta, Andrew D Bagdanov, and Joost van de Weijer. Class-incremental learning: survey and performance evaluation. *IEEE TPAMI*, 2022. 1

[23] Jaeyoo Park, Minsoo Kang, and Bohyung Han. Class-incremental learning for action recognition in videos. In *ICCV*, pages 13698–13707, October 2021. 2

[24] Juan-Manuel Pérez-Rúa, Xiatian Zhu, Timothy Hospedales, and Tao Xiang. Incremental few-shot object detection. In *CVPR*, 2020. 2

[25] Viresh Ranjan, Udbhav Sharma, Thu Nguyen, and Minh Hoai. Learning to count everything. In *CVPR*, 2021. 3, 5

[26] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *CVPR*, pages 2001–2010, 2017. 1, 2

[27] Deepak Babu Sam, Skand Vishwanath Peri, Mukuntha Narayanan Sundararaman, Amogh Kamath, and R. Venkatesh Babu. Locate, size and count: Accurately resolving people in dense crowds via detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020. 2

[28] Deepak Babu Sam, Shiv Surya, and R Venkatesh Babu. Switching convolutional neural network for crowd counting. In *CVPR*, 2017. 2

[29] Min Shi, Hao Lu, Chen Feng, Chengxin Liu, and Zhiguo Cao. Represent, compare, and learn: A similarity-aware framework for class-agnostic counting. In *CVPR*, 2022.

[30] K. Shmelkov, C. Schmid, and K. Alahari. Incremental learning of object detectors without catastrophic forgetting. In *ICCV*, pages 3420–3429, 2017. 1

[31] Qingyu Song, Changan Wang, Zhengkai Jiang, Yabiao Wang, Ying Tai, Chengjie Wang, Jilin Li, Feiyue Huang, and Yang Wu. Rethinking counting and localization in crowds: A purely point-based framework. In *ICCV*, 2021. 2, 3

[32] Gido M. van de Ven and Andreas S Tolias. Three scenarios for continual learning. In *NeurIPS CL Workshop*, 2019. 3

[33] Cédric Villani. *Optimal transport: old and new*, volume 338. Springer Science & Business Media, 2008. 3

[34] Jia Wan, Ziquan Liu, and Antoni B Chan. A generalized loss function for crowd counting and localization. In *CVPR*, 2021. 2

[35] Boyu Wang, Huidong Liu, Dimitris Samaras, and Minh Hoai. Distribution matching for crowd counting. In *NeurIPS*, 2020. 2, 3, 6

[36] Qi Wang, Junyu Gao, Wei Lin, and Yuan Yuan. Learning from synthetic data for crowd counting in the wild. In *CVPR*, 2019. 2

[37] Chenshen Wu, Luis Herranz, Xialei Liu, Yaxing Wang, Joost Van De Weijer, and Bogdan Raducanu. Memory replay gans: learning to generate images from new categories without forgetting. In *NeurIPS*, 2018. 2

[38] Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. Large scale incremental learning. In *CVPR*, pages 374–382, 2019. 1

[39] Gui-Song Xia, Xiang Bai, Jian Ding, Zhen Zhu, Serge Belongie, Jiebo Luo, Mihai Datcu, Marcello Pelillo, and Liangpei Zhang. Dota: A large-scale dataset for object detection in aerial images. In *CVPR*, 2018. 5

[40] Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. Lifelong learning with dynamically expandable networks. *arXiv preprint arXiv:1708.01547*, 2017. 1, 2

[41] Yingying Zhang, Desen Zhou, Siqin Chen, Shenghua Gao, and Yi Ma. Single-image crowd counting via multi-column convolutional neural network. In *CVPR*, volume 2016-Decem, pages 589–597, 2016. 2