[Re] Unsupervised Domain Adaptation For Plant Organ Counting

Thong Vo
Ryerson University
thong.vo@ryerson.ca

Abstract

This project is a reproduction of the paper Unsupervised Domain Adaptation For Plant Organs Counting. The original paper discussed an extension of Domain Adaptation for the case of dense object counting where annotated datasets could be limited. Specifically, in the plant organs counting field, the data collecting process is challenging due to intensive labeling requirements and there are significant domain shifts between species images or variations between plant growth stages. Therefore, the authors propose a domain-adversarial learning approach for domain adaptation in such object counting contexts. The approach is applicable for both general object counting and plant organ counting tasks because it does not impose any assumption on the distributions of studied domains. To assess the claims in the paper, we re-implement the counting network with adaptation to reproduce the main results. Following that, we compare the results and discuss key insights obtained from the re-implementation.

1 Introduction

The counting problem is the estimation of the number of objects in an image or video frame. It arises in many real-world applications such as monitoring crowds in surveillance systems, estimate the number of cells in a microscopy image, or counting the number of cars on a street. Object counting is also a key task in image-based plant phenotyping where plant health can be assessed via plant organ counting. In fact, there is an annual leaf counting competition based on the Computer Vision Problems in Plant Phenotyping (CVPPP) dataset [1].

Plant organ objects are typically small and densely distributed making the annotation and detection tasks challenging. Besides, because plant images are significantly different between crops, fields, and growing seasons, directly apply learning from one dataset to another dataset will likely lead to low performances. In other words, due to the domain-shift challenge, a model trained to count objects on one dataset will not achieve high results with other datasets that have different prior distributions. One solution for this problem is to leverage domain adaptation techniques to align the related domains.

The authors of the original article [2] first formulate the object counting problem as a density map estimation problem and then employ a domain adaptation method to transfer learning from the original domain to the new domain. In this way, they present a method that applies an unsupervised domain adaptation technique to jointly train a convolutional neural network to count objects from images with dot annotations and adapt this knowledge to a related set of images where labels are absent. Their experiments' results demonstrate improvements compared to baseline models trained without domain adaptation. Also, the gains in performance are comparable with a previously proposed domain adaptation approach[3].

This work comprises a full re-implementation and reproduction of the model and leaf counting results. Additionally, we propose some modifications in the training loop in an attempt to yield better training

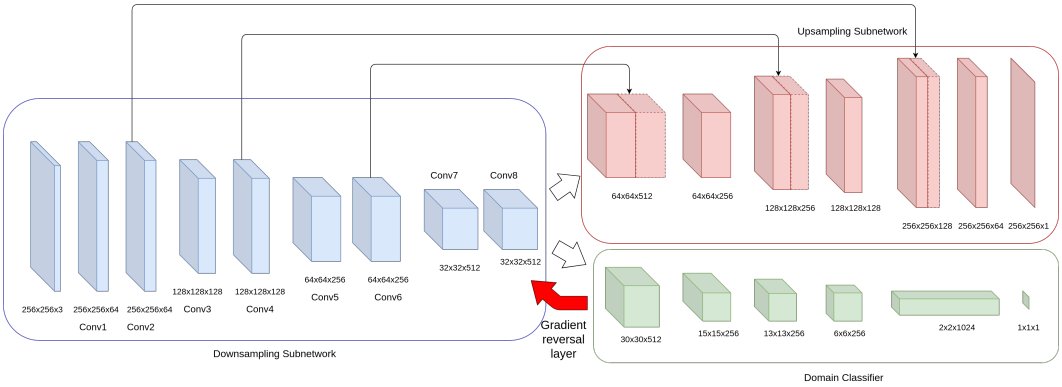


Figure 1: The proposed Domain-Adversarial Neural Network. The three subnetworks are the Downsampling Subnetwork (G_d - in blue), the Upsampling Subnetwork (G_u - in red), and the Domain Classifier (G_c - in green). The red arrow indicates the Gradient Reversal Layer which reverses the gradient during backpropagation for domain adaptation purpose [4].

outcomes. We discuss the main model, present obtained results, analyze findings, and discuss some discrepancies compared to the original results in the follow-up sections.

2 Methods

In this section, we aim to provide a sufficient background of the counting with adaptation model for the reader to be able to follow the reproduced results. Additionally, we outline which experiments and results that we intend to reproduce.

2.1 Counting with Adaptation model

The task of domain adaptation in this object counting framework is modeled as an unsupervised domain adaptation problem. Data are sampled from two domains: a source domain ($\mathcal{D}^s$) and a target domain ($\mathcal{D}^t$). There is one assumption that labels only exist for images from the source domain. In that way, the source domain includes a set of images ($\mathcal{X}^s$) and their corresponding labels ($\mathcal{Y}^s$), while the target domain has only images set ($\mathcal{X}^t$) without labels.

The proposed architecture follows the Domain-Adversarial Neural Network (DANN) design [4] for the domain adaptation purpose. It consists of three sub-networks that perform the following tasks: the feature extraction - downsampling network (G_d), counting prediction - upsampling network (G_u), and the domain classification network (G_c). The feature extraction layers take RGB images as inputs and produce feature maps for downstream tasks. Following that, the counting prediction branch up-samples these features and generate a density estimation map that is equal to the size of the input image. Together, the extraction layers and the counting prediction layers form a U-Net style model. The remaining branch classifies whether the input image is sampled from the source domain or the target domain. The gradients from this network are passed through a Gradient Reversal Layer prior to going back to the feature extractor. The purpose of this step is to encourage the network to find domain-invariant features common to both the source and the target domain[4]. A negative constant ($-\lambda$) controls the rate of the reverse gradient. This rate varies through the training time to fine-tune the performance of the model.

The loss of the model is a combination of density estimation loss and domain classification loss. The details of the two-part loss function are described as follows:

$$\mathcal{L}_{density} = \log(\frac{1}{N}\sum_{i=0}^{N}(G_u(G_d(x_i^s)) - y_i^s)^2) \tag{1}$$

$$\mathcal{L}_{domain} = -\mathbb{E}_{x^s \sim \mathcal{X}^s}[\log(G_c(G_d(x_i^s))] - \mathbb{E}_{x^t \sim \mathcal{X}^t}[1 - \log(G_c(G_d(x_i^t))] \tag{2}$$

$$\mathcal{L} = \mathcal{L}_{density} + \mathcal{L}_{domain} \tag{3}$$

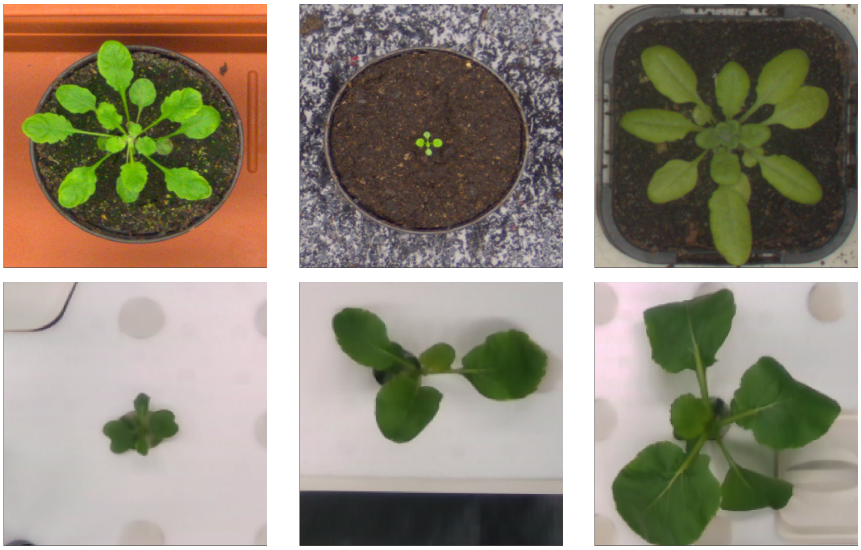


Figure 2: Example images for leaf counting experiments. Top: samples from Source dataset, CVPPP Dataset. Bottom: samples from Target dataset, KOMATSUNA Dataset [5]

where (x_i^s, y_i^s) represents the ith image and density map label sampled from the source domain ($\mathcal{D}^s$). The Density loss ($\mathcal{L}_{density}$) is calculated as the logarithm of mean squared errors between the density map estimation $G_u(G_d(x_i^s))$ for samples taken from the source domain, and its ground truth labels y_i^s . The logarithm term is added to balance the multi-part loss function (3) . The Domain loss ($\mathcal{L}_{domain}$) is a Binary Cross-Entropy loss for domain classification.

2.2 What we Reproduce

The authors assess their solutions with two distinct tasks: leaf counting and wheat spikelet counting. In this work, we verify the proposed model with the leaf counting task because the related datasets are readily available. The reported metrics are similar to the CVPPP 2017 competition's metrics. These metrics are Difference in Count (DiC), absolute Difference in Count ($|DiC|$), mean squared error (MSE) and percentage agreement($\%$). The details for these metric are presented in the results summary.

3 Experiments

3.1 Datasets

The two datasets for our reproducibility experiments are:

- The source dataset: *CVPPP 2017 Leaf Counting Dataset* [1]. This dataset is collected for the CVPPP leaf counting competition, comprising *Arabidopsis thaliana* and tobacco plants. The dataset includes various types of annotations such as leaf segmentation masks, leaf bounding boxes and leaf center dot annotations. The leaf center annotations were employed in this work and groud truth density maps were generated by filtering the center points with a 2D gaussian filter with $\sigma = 3$. We contacted the author of the paper [2] in order to get the dataset with the generated density maps.
- The target dataset *KOMATSUNA Dataset* [5]. Top-view images of Komatsuna plants developed for 3D plant phenotyping tasks. The RGB images of the plant are used in this experiment. The counting labels are only used for prediction benchmark purposes.

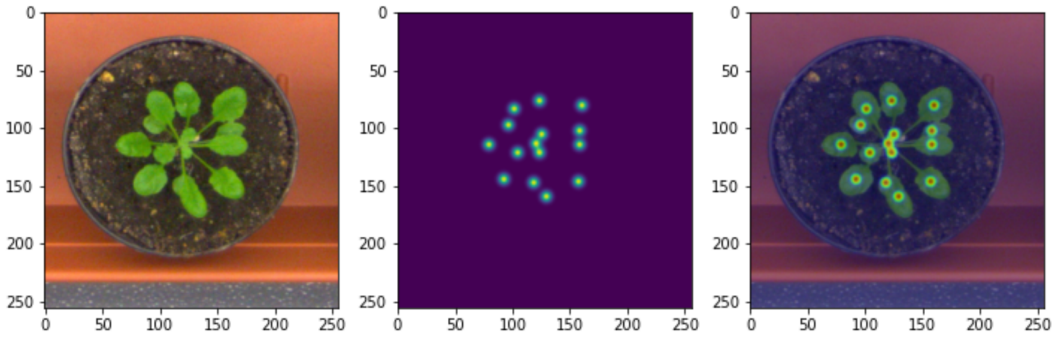


Figure 3: A sample of dot annotation image with 2D gaussian filters.

3.2 Implementation details

We re-implement the leaf counting experiments in Pytorch framework. Dataloaders generate batches of data by mapping predefined transformation functions on a batch of training data. The dataloaders read the jpeg file, then resize them to a standard 256x256 pixels, and transform pixel values them from 0-255 range to 0-1 range. In this implementation, every dataset has its own dataloader. In the author's reference implementation, there is only one dataloader for both source and target dataset. While this approach is convenient for the training loop, there are some drawbacks to this method. More details are discussed in the analysis section.

In the train loop, batches of data from the source data set and the target dataset take turns to feed into the network. For data from the source dataset, both related losses in the density esimation branch and the domain classification branch are considered. For data from the target dataset, only the loss from the domain classification branch is taken into account because our assumption is that the label for target dataset is not available. The learning rates follow the recommendations from the original paper, 10^{-3} for the downsampling and upsampling subnetworks, and 10^{-4} for the domain classifier. The Optimizer is Adam with AMSGrad [6]. The model is trained for 50 epochs.

3.3 Summary of the results

In this experiment, we adapt the leaf counting from the CVPPP dataset to KOMATSUNA dataset. Assuming $\epsilon_i = y_i - \hat{y}_i$ to be the difference between the ground truth and the prediction, the evaluation metrics[3] are listed below:

- Difference in Count [DiC]: $\frac{1}{N}\sum_{i=1}^{N}\epsilon_i$
- Absolute Difference in Count[|DiC|]: $\frac{1}{N}\sum_{i=1}^{N}|\epsilon_i|$
- Mean Squared Error [MSE]: $\frac{1}{N}\sum_{i=1}^{N}\epsilon_i^2$
- Percentage Agreement [%]: $\frac{1}{N}\sum_{i=1}^{N}1\left[\epsilon_i = 0\right]$

The first line of the results in Table 1 shows the metrics for the model performance if none of the adaptation methods are applied. The next line indicates reported results from a comparable method [7]. The last two lines present the results with adaptation applied during the training process, one from the original article and the second one from the re-implemented results.

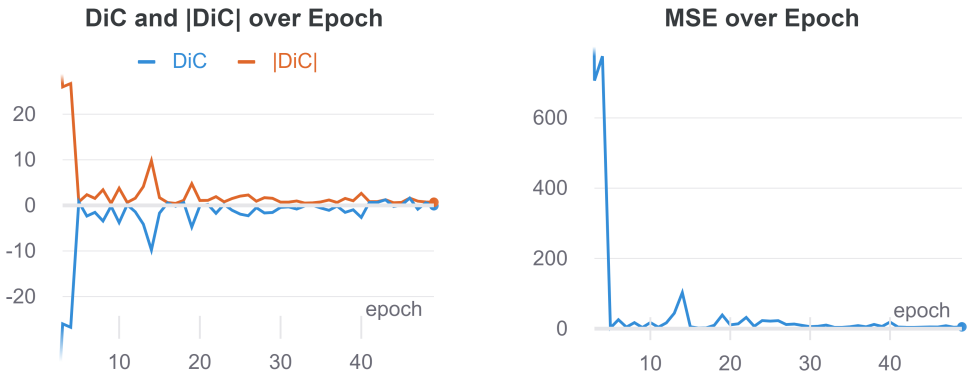
4 Discussion

Overall, we were able to reproduce results that are close to the claims in the article. The adaptation branch significantly improves the performance of the model with a new dataset. The MSE loss quickly settles to a small range after roughly 5 epochs.

Besides, we notice some considerable challenges of this model. Firstly, the variances of the results are very high, so it is difficult to select an early stopping criteria. For example, as reported in the article

Table 1: Domain adaptation results for the leaf counting tasks. XE is cross-entropy loss, LS is least-squares loss, ↓ denotes lower is better, ↑ denotes higher is better.

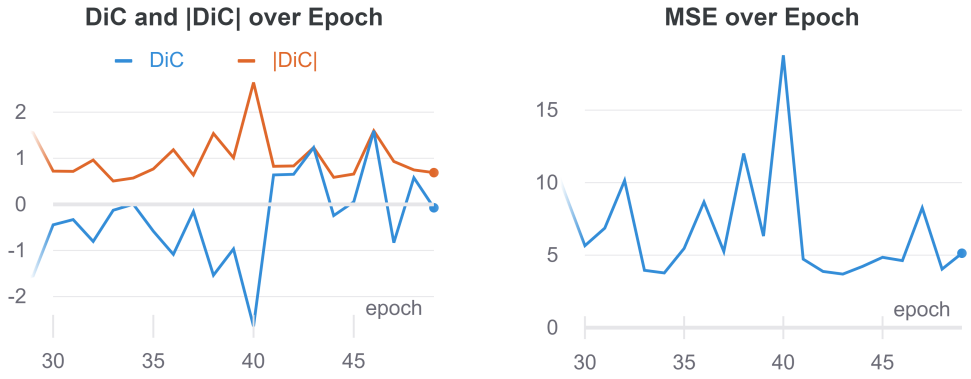
	$DiC \downarrow$	$\lvert DiC \rvert \downarrow$	$\% \uparrow$	$MSE \downarrow$
CVPPP to KOMATSUNA				
No Adaptation-from article	4.09	4.09	0	18.49
With Adaptation - Giuffrida, et al. [7]	-0.78	1.04	26	1.84
With Adaptation - from article	-0.95	1.56	29.33	5.26
With Adaptation - from code	0.69	-0.07	16.67	5.13



(a) Different in Count metrics over Epoch

(b) Mean Squared Error over Epoch

Figure 4: The model achieves favorable results after 20 Epochs.



(a) Different in Count metrics over Epoch

(b) Mean Squared Error over Epoch

Figure 5: Unstable testing results between epochs. The variances of key performance metrics are significant between epochs even after the training loss is settled.

the adapted model has the Different in Count values with $\mu = -0.95, \sigma = 2.09$. The adaptation results fluctuate considerably between epochs. Figure 5 illustrates the variances of the results when the training loss is stable. Secondly, the key metrics such as Difference in Count (DiC), mean squared error (MSE) and percentage agreement($\%$) do not appear to strongly correlate to each other. While it is true that most of the metrics improve with more training, optimizing for one metric might not achieve favorable results for others.

One key difference between the original implementation and this version is the data feeding process. The original implementation combines data from the source domain and the target domain into the

same batch for the training loop, while our implementation splits them to separated batches. Our approach follows the tips from [8] to improve GAN-like model performance. The conjecture for the differences in the performance is the underlying interactions between the batch-norm layers, the batch statistics and the performance of the discriminator.

Another observation for this paper is that while the main model was described in detail, the evaluation methods part could be improved. For instance, the process to convert from the predicted density map to a final counting number was not clear. Also, whether to round the predicted number to an integer value or not was not mentioned. Though, the reference code from the author was really helpful to reproduce the results.

5 Contribution Statements

The author confirms sole responsibility for the following: literature review, re-implementation, interpretation of results, and manuscript preparation. The author would like to acknowledge the first author of the original article for his email responses to provide the dataset and clarify some implementation details.

References

[1] M. Minervini, A. Fischbach, H. Scharr, and S. A. Tsaftaris, "Finely-grained annotated datasets for image-based plant phenotyping," *Pattern Recognition Letters*, vol. 81, pp. 80–89, 2016.

[2] T. W. Ayalew, J. R. Ubbens, and I. Stavness, "Unsupervised Domain Adaptation for Plant Organ Counting," in *Computer Vision – ECCV 2020 Workshops* (A. Bartoli and A. Fusiello, eds.), (Cham), pp. 330–346, Springer International Publishing, 2020.

[3] M. V. Giuffrida, A. Dobrescu, P. Doerner, and S. A. Tsaftaris, "Leaf counting without annotations using adversarial unsupervised domain adaptation," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, vol. 2019-June, pp. 2590–2599, 2019.

[4] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky, "Domain-adversarial training of neural networks," *Advances in Computer Vision and Pattern Recognition*, vol. 17, no. 9783319583464, pp. 189–209, 2017.

[5] H. Uchiyama, S. Sakurai, M. Mishima, D. Arita, T. Okayasu, A. Shimada, and R. Taniguchi, "An easy-to-setup 3d phenotyping platform for komatsuna dataset," in *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*, pp. 2038–2045, 2017.

[6] S. Reddi, S. Kale, and S. Kumar, "On the convergence of adam and beyond," in *International Conference on Learning Representations*, 2018.

[7] M. V. Giuffrida, A. Dobrescu, P. Doerner, and S. A. Tsaftaris, "Leaf counting without annotations using adversarial unsupervised domain adaptation," in *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 1–8, 2019.

[8] S. Chintala, E. Denton, M. Arjovsky, and M. Mathieu, "How to Train a GAN?," *NIPS*, 2016.