ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)

Vol 11, Issue 3, March 2024

Comparative Analysis of Normalization Techniques in Convolutional Neural Networks for Leaf Counting in Arabidopsis thaliana

[1] Zorana Štaka, [2] Marko Mišić

[1] Faculty of Electrical Engineering, University of East Sarajevo, Bosnia and Herzegovina
[2] School of Electrical Engineering, University of Belgrade, Serbia
Corresponding Author Email: [1] zorana.staka@etf.ues.rs.ba, [2] marko.misic@etf.bg.ac.rs

Abstract*— Normalization techniques have been extensively used in deep learning methods to facilitate the learning process. Normalization decreases the impact of different scales of the input features, improves convergence of the model affected by the changes of input features or trained weights, and positively affects the generalization and robustness of the model. Three main techniques of normalization have been used in the open literature for research in the domain of computer vision: Batch, Layer, and Instance Normalization. New approaches, such as Switchable Normalization, suggest using different normalizers for different normalization layers in a deep neural network. In this work, we evaluate those four normalization techniques in the context of the computer vision problem, leaf counting, in the plant Arabidopsis thaliana. The four normalization techniques were evaluated using the popular evaluation metrics commonly used for leaf counting problem: difference in count, absolute difference in count, accuracy, and mean square error. The results show that the best models are those using Instance, Layer and Switchable normalization.

Index Terms*—deep learning, neural networks, normalization, switchable normalization.

I. INTRODUCTION

Deep learning methods play a key role in many successful applications of machine learning in the domains of image segmentation [1], object tracking [2], natural language processing [3], speech recognition [4], and many others. In numerous cases, convolutional neural networks (CNN) were used for such tasks, as they are designed to extract and classify features from images.

Usually, CNNs are multi-layered neural networks consisting of convolutional, pooling, normalizing, dropout, flattening, and fully connected layers [5]. A convolutional layer is used to extract important features from the input images. It is usually followed by pooling layers, which are aimed at decreasing the computational cost by reducing the size of the convolved feature map. Fully connected layers are used for classification purposes after the input is converted by a flattening layer. Different regularization strategies and layers are used to enhance the performance and learning process and prevent overfitting, such as weight decay, data augmentation, normalization, and dropout [6].

Normalization layers are used in CNN architectures to tackle issues in training, such as stability, optimization efficiency, and generalization ability [7]. The aim of the normalization is to transform the input data to use the same scale, usually between 0 and 1, in order to stabilize the gradient descent step in training. There are numerous normalization approaches used in CNNs, such as Batch Normalization (BN) [8], Instance Normalization (IN) [9], and Layer Normalization (LN) [10]. Usually, the same normalizer is used in all normalization layers in the network. A good general overview of such techniques can be found in [6][7].

However, recent studies [11][12] showed that different normalizers can be used in different normalization layers. Moreover, it is shown that the choice of normalizer can be learned [11]. In that context, the Switchable Normalization (SN) approach [13] has been proposed, which allows to select different normalizers for different normalization layers of a deep neural network by calculating various statistics and learning their importance during training. The authors of [13] showed good performance of SN for image classification and object detection computer vision tasks, where a combination of all BN, IN, and LN is preferred.

Studying plant behavior and traits is important in contemporary precision agriculture [14]. Computer vision, pattern recognition, and machine learning are often employed for such tasks [15]. Following our previous work on the leaf counting problem [16], we adopted the Switchable Normalization strategy in our solutions and evaluated it using several datasets from the CVPPP 2017 challenge based on the Arabidopsis thaliana plant, which is a frequently studied plant in the leaf counting context. The main objective of the study was to compare the impact of different normalization techniques on the generalization and performance of the model for the leaf counting task.

The rest of the paper is structured as follows: In the second section, we give an overview of the different normalization methods used in computer vision tasks with convolutional

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)

Vol 11, Issue 3, March 2024

neural networks. The third section describes our datasets and methods and presents details on how switchable normalization is implemented for a leaf counting task. The results of the study are presented in the fourth section, along with the discussion. The final section concludes the paper with directions for future work.

II. RELATED WORK

The general model of the normalization layer shared between BN, LN, and IN can be expressed with the following equation:

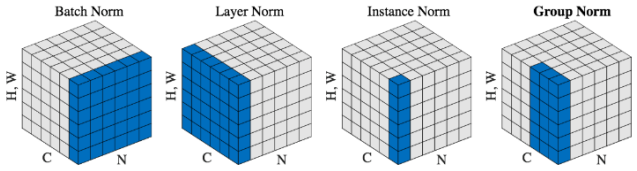

Fig. 1. Different normalization methods and their domains of operation representing a 4D tensor, consisting of N samples, C channels, height H, and width W [17].

$$\hat{x} = \gamma \frac{x - \mu}{\sqrt{\sigma^2 + \varepsilon}} + \beta$$

where x and $\hat{x}$ are pixel values before and after normalization, μ and σ are a mean and a standard deviation, and γ and β are a scale and a shift parameter, while ε is a small constant to ensure numerical stability. While BN, IN, and LN share the same equation, they use different sets of pixels to calculate the mean and standard deviation [13]. On the 4D tensor feature map, Fig. 1 illustrates how mean and standard deviation are calculated, as well as the way pixel values after normalization are derived [17]. The remainder of this section includes describing the three most prominent normalization approaches used in CNNs in modern deep neural networks, and in addition to that, the concepts of the Switchable Normalization technique are explained.

A. Batch Normalization

Batch Normalization is one of the most commonly used techniques in deep neural networks. It was proposed by Ioffe and Szegedy [8] to solve the problem of internal covariate shift during the learning process through the stochastic gradient descent method. It is shown that the training process can be affected by changes in the distribution of input data in deeper layers of the network. The input of deeper layers is affected by the parameters of all previous layers. For that reason, small changes in input distribution tend to become larger as the network gets deeper, affecting performance.

BN is performed between the layers of the given CNN on mini-batches of data of the defined size. The features are normalized based on the mean and variance in the mini-batch, subtracting the mean value and dividing the feature by its mini-batch standard deviation. In that way, all features have zero mean and unit variance, leading to a more stable and efficient learning process. The statistics (mean and variance) are calculated differently during training and inference; throughout the training, mini-batch statistics are used, whereas population statistics are applied during inference. Also, while training, it is important to keep track of the moving mean and moving variance, which will be later used in the inference.

The main problem associated with Batch Normalization is the choice of mini-batch size. Smaller mini-batch sizes are generally avoided, as they might lead to inaccurate calculations of the mean and variance, leading to errors [17]. On the other hand, larger mini-batches lead to implementation problems, especially on the GPUs, as they tend to be overly memory-consuming.

B. Layer Normalization

A similar idea from the BN is used in the Layer Normalization technique [10]. However, the LN technique computes mean and variance from all the summed inputs to the neurons in a layer on a single training sample rather than across the samples for each feature represented in a mini-batch, as shown in Fig. 1.

LN is performed in exactly the same way during training and inference phases. Also, it can be used regardless of the mini-batch size, which is beneficial for applications where batch sizes may vary. The most important advantage over BN is that LN can be more easily integrated with recurrent neural networks (RNN), which are used in popular architectures such as LSTM and ResNet.

C. Instance Normalization

Instance Normalization [9] can be applied both in training and inference phases, similar to Layer Normalization. The main difference is that instance normalization is performed across each channel in each training sample instead of normalizing across input features in a single training sample. If normalization is performed over a group of channels for each training sample, it is called Group Normalization (GN) [17], as shown in Fig. 1.

The main motivation for IN comes from the problem of image stylization, i.e., transferring a style from one image to another. In many real-life applications, the variability of visual styles, including contrast, object textures, lighting conditions, and filter effects, deeply affects image classification [12]. In some more advanced applications, IN is combined with BN to explicitly manipulate style information from the images. That approach is called Batch-Instance Normalization (BIN) [12], and it is able to selectively learn which image styles to normalize and which ones to keep during the training process for object classification and style transfer tasks.

D. Switchable Normalization

The first idea that different normalizers can be used in different normalization layers in CNN was presented in [13].

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)

Vol 11, Issue 3, March 2024

The authors solved a learning-to-normalize problem by computing statistics employed in the normalization process (means and variances) from three different sources: mini-batches, layers, and channels. SN switches between those three normalization strategies depending on the learned importance weights for each layer of the network. For the whole mathematical model, one can refer to [13].

It is shown that SN is robust for different deep learning tasks, where BN is preferred for image classification and object detection, LN is more often used in the box and mask methods, and IN is used for image style transfer [13]. It is also observed that IN is preferred at the beginning of the network, BN is more often used in the middle layers, and LN is used more frequently towards the end. Also, SN is more robust to mini-batch size, which is a problem for BN in the case of smaller mini-batches. The authors have proven the superiority of the approach for several key applications and datasets in their subsequent study [11]. Those include image classification with the ImageNet dataset and ResNet50 architecture, object detection and instance segmentation in the COCO dataset, semantic image segmentation and parsing in the ADE20K and Cityscapes datasets, and video recognition in the Kinetics dataset.

Fig. 2. Example images from the CVPPP 2017 challenge Arabidopsis thaliana images from A1, A2, and A4 directory

III. DATASETS AND METHODS

The A1, A2, and A4 directories of the CVPPP 2017 challenge Arabidopsis thaliana images are used as the training and testing dataset for trained neural networks. In total, there are 128 images (500 pixels × 530 pixels), 31 images (530 pixels x 565 pixels), and 624 images (441 pixels x 441 pixels) in the A1, A2, and A4 directory, respectively. In these images, several challenges are present that make recognizing and counting leaves a difficult task: a layer of water that causes reflection in some trays, a changing background that may contain some other things except the plant itself, and overlapping leaves as the plant grows that result in the presence of occlusion. Example images from A1, A2, and A4 directory are shown in Fig. 2 having 11, 4 and 28 leaves respectively.

In addition to the images in the folders, the ground truth of leaf count is also provided. To get an appropriate overview of how well one model is good for predicting the number of leaves, one cannot look only at the accuracy (the exact number of images where the number of leaves was correctly predicted). The difference between the prediction and the ground truth is also very essential; that is, even if the prediction is incorrect, it is crucial how far it is from the truth. For that reason, we use other metrics, such as difference in count, absolute difference in count, and mean squared error, in addition to accuracy. Therefore, evaluation metrics used in this work are difference in count (DiC), absolute difference in count (|DiC|), accuracy, and Mean Square Error (MSE). A detailed overview of these metrics can be found in [16][18][19].

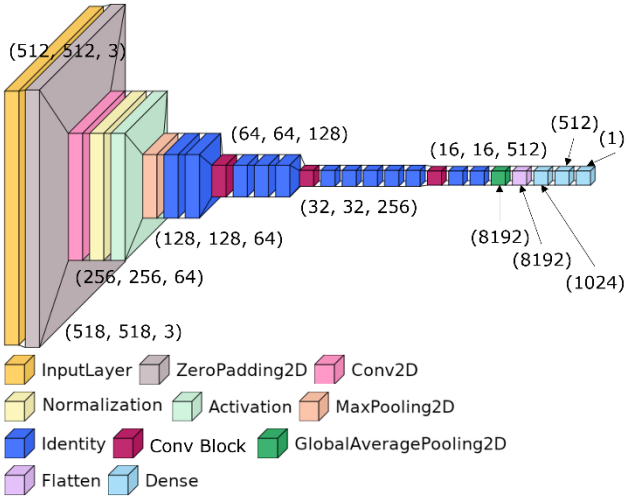

Fig. 3. Shematic diagram of a modified ResNet model for leaf counting from our previous work [16]

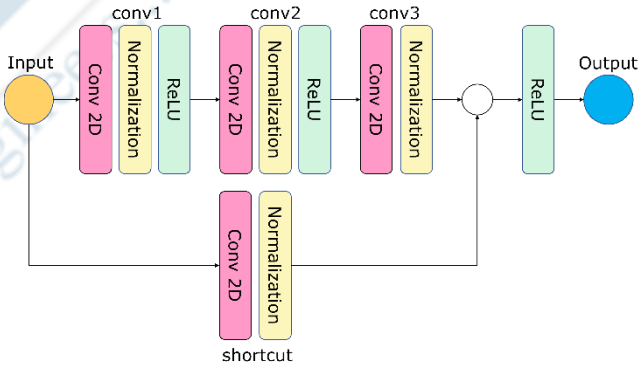

Fig. 4. Shematic diagram of a Conv block [16]

The TensorFlow (version 2.15) Python library was used to implement neural networks for the leaf counting task. The neural network architecture was based on a modified ResNet model that showed best performance in our previous work [16]. Its schematic diagram is shown in Fig. 3. Model also encompasses the *Identity* and *Conv* blocks, whose schematic diagrams are given in [16]. Such an architecture results in 52 layers, of which 49 are convolutional layers and three are dense layers. The following hyperparameters and their respective values are used: activation function (*ReLU*), stride (2x2), learning rate (1e-5), kernel size (1x1, 3x3, 7x7), optimizer (*Adam*) and loss function (MSE).

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)

Vol 11, Issue 3, March 2024

The model (including *Identity* and *Conv* blocks) uses BN, IN, LN, or SN in place of all the normalization layers, resulting in four different versions of the ResNet model whose performances are compared to each other. Each of these four different model versions is trained and tested with batch sizes of 2, 4, 8, 16 and 24, and with a number of epochs ranging from 10 to 100.

The construction of SN involves computing all the statistics (mean and variance) for the batch, instance, and layer normalization, and then utilizing the linear combination of these to derive the final mean and variance. Switchable Normalization is implemented as a class inheriting *tf.keras.layers.Layer*. The trainable parameters added to the layer were scale, offset, as well as mean weight and variance weight (for the above-mentioned linear combinations of mean and variance). Since BN also needs moving mean and variance, those are added but are not trainable. The implementation also takes into account whether the network is in the state of training or inference, since the mean and variance of BN are calculated differently in those two cases.

IV. RESULTS AND DISCUSSION

In this section, we provide the results of our analysis. The versions of ResNet model using different normalization techniques are compared here in terms of time needed for training, as well as evaluation metrics DiC, |DiC|, accuracy and MSE.

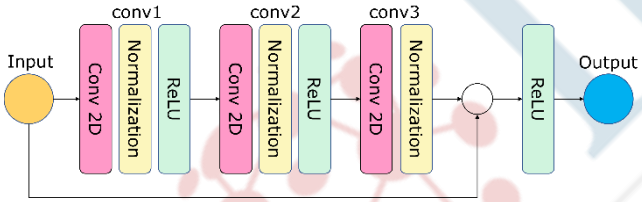

Fig. 5. Shematic diagram of a *Identity* block [16]

Table I Training time of all batch sizes for different normalization techniques used in the model, rounded to the nearest second.

Norm. technique	Batch size					
	2	4	8	16	24	Average
BN	1303	933	836	897	809	956
IN	1307	960	861	923	831	977
LN	1304	998	931	875	875	997
SN	1389	1064	1050	925	920	1070

Table I gives us an overview of the time required to train models with the four normalization techniques for all the batch sizes used (2, 4, 8, 16, and 24). On average, models with BN require the shortest training time, whereas models with SN require the longest. This one was expected because models using SN need to compute all the statistics for BN, IN, and LN. Models using SN take 6-25% more time to be trained than models using BN. Models utilizing IN and LN generally need less time than models using SN and more time than models using BN, whereas in some certain cases, e.g., a batch size 16 model using LN is faster than one using BN. Overall, all models take less than 25 minutes to train, and the differences between them at these ranges of execution time are almost negligible.

For the metrics of DiC, |DiC|, accuracy, and MSE, the best values for different batch sizes were chosen for each of the normalization techniques and then compared to each other. A summary of the DiC metric is given in Fig. 6. It can be observed that the values in the case of using SN fluctuate the most, while the model employing IN and LN changes the least, remaining at two units around zero through all the epochs.

The best value for DiC is zero and is achieved with BN after 50 epochs for a batch size of 24. This does not imply that the model's accuracy on the test set in this case is 100%, but rather that the overestimation and underestimation of the number of leaves cancel each other out. This scenario also does not attain the best accuracy for the BN model, but rather batch size 16 (where accuracy is 39.74% as opposed to 33.33%). Similarly, a model using LN with a batch size of 2 achieves zero for DiC, and essentially the same reasoning applies. At the end of the training (after 100 epochs), all the values are under 1.1 (with IN having the lowest of 0.04 and LN having the highest of 1.06).

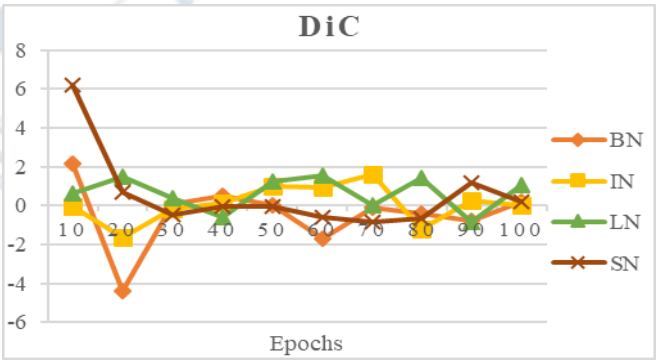

Fig. 6. The results for the DiC across 100 epochs for different normalization techniques

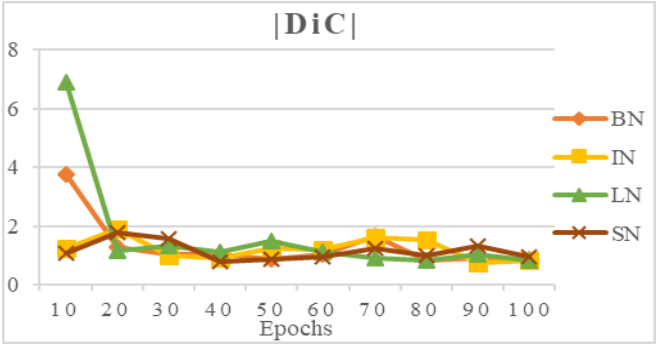

Fig. 7. The results for the |DiC| metric across 100 epochs for different normalization techniques

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)

Vol 11, Issue 3, March 2024

Fig. 7 depicts the values of metric |DiC| changing over epochs for the models with different normalization techniques. After training for the 100 epochs all the models have comparable value for |DiC| ranging from 0.81 (IN and LN) to 0.95 (SN). The best values of |DiC| in case SN and IN are achieved in case of batch size 8, BN requires 16, while LN works best in case of 24 images in a batch. For model with IN the values for |DiC| range from 0.74 (after 90 epochs) to 1.88 (after 20 epochs). When we consider values for DiC and |DiC| for IN it can be observed that the model at the beginning is overestimating the number of leaves, then starts to underestimate the number of leaves, while also getting closer to the true number of leaves.

The trend of SN being dominant in the first 50 epochs when it comes to the accuracy of the model can be observed in Fig. 8. In the second part of the training performances of model with SN are decreasing significantly, going from 41.03% (maximum in the first 50 epochs) to even only 15.38% (after 60 epochs) and to 21.79% after the training is completed. The low accuracy at the very beginning is observed in case of BN and LN and in the end the models utilizing BN and IN have the highest accuracy (38.46% for both). The highest accuracy of 43.59% in all the training epochs is accomplished with LN (after 70 epochs) and IN (after 90 epochs). SN has the highest minimum of the four models for all the epochs, always staying above 15%, while on the other hand the accuracy of just few percentages can be observed in models using different normalization techniques (but at the very beginning of the training process).

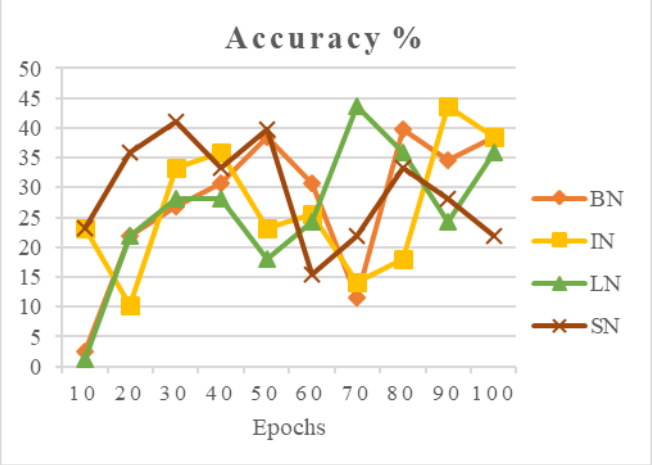

Fig. 8. The results for the accuracy metric across 100 epochs for different normalization techniques

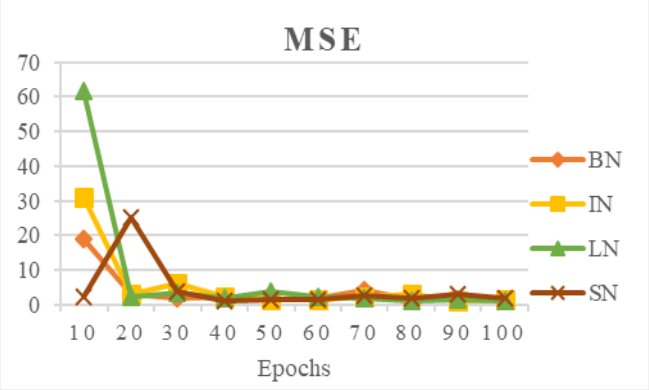

Fig. 9. The results for the MSE metric across 100 epochs for different normalization techniques

For the metric of MSE the overview is given in Fig. 9. The lowest value is achieved in case of SN after only 40 epochs, while the lowest value for BN, IN, and LN are achieved after 80, 90, and 100 epochs respectively. For SN this is achieved when using only 8 images per batch, while for other the batch size is larger. In Fig. 9, it can be observed that at the very beginning (after 10 epochs), BN, IN, and LN have a significantly larger number for the MSE metric compared to the later stages of the training process. For SN, the same thing happens just after 20 epochs.

To provide an overall performance overview for all the models and all the batch sizes, Table II presents the top 5 models with the highest accuracy, and Table III shows the top 5 models with the lowest MSE.

The highest accuracy on the test set is obtained in the case of modles using IN, LN, LN, IN, and SN, with batch sizes of 8, 24, 4, 2, and 4, respectively (Table II). The lowest value for accuracy, only 25.64%, is in the case of BN with two images per batch, and a batch size of 24 for BN produces only 33.33% accuracy, which is among the lowest. For batch size 16, however, an accuracy of 39.74% is obtained. This is consistent with the previous remark that BN is affected by the choice of batch size.

The lowest values for MSE are achieved in models using SN, IN, LN, IN, and SN with batch sizes of 8, 24, 24, 8, and 16 respectively (Table III). Model with BN and batch size of 2 takes the last place here as well, having an MSE of 4.59.

The obtained results of the four models are consistent with the results obtained in [16], where only BN was used for normalization. In the case of BN with a batch size of 8 in [16], the accuracy of 30.77% is achieved after 50 epochs, while here in the same case, the accuracy is 32.05% (the MSE is 1.45 in [16] as opposed to the 1.79 obtained here). From Table II and Table III, it can be observed that the same model as one used in [16] but with different normalization techniques and batch sizes can outperform the model with BN and a batch.

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)

Vol 11, Issue 3, March 2024

Table II Accuracy obtained on the test set for top 5 normalization techniques and batch sizes

Normalization technique	Batch size	Accuracy (%)
IN	8	43.59
LN	24	43.59
LN	4	42.31
IN	2	41.03
SN	4	41.03

Table III MSE values achieved on the test set for top 5 normalization techniques and batch sizes

Normalization technique	Batch size	MSE
SN	8	1.12
IN	24	1.24
LN	24	1.24
IN	8	1.26
SN	16	1.27

Size of 8. In the context of SN, the corresponding model shows similar performance in terms of DiC, |DiC|, accuracy, and MSE. However, it can be noted that accuracy drops after 50 epochs. Similar trends can be observed for other techniques, as accuracy is not stable over epochs. That effect should be further investigated in future work.

V. CONCLUSION

The advent of new normalization techniques can lead to more robust and stable machine learning models with better generalization. One of the recent ideas is to use different normalizers for different normalization layers in convolutional neural networks through a technique called switchable normalization.

Following our previous work, we tried different normalization techniques as well as adopted switchable normalization for the problem of leaf counting in the Arabidopsis thaliana plant, which is a computer vision problem from the domain of object detection. Our results suggest that among the best models are those using Instance, layer, or Switchable Normalization, placing themselves in front of the Batch Normalization. However, based on our experience, we cannot conclude that Switchable Normalization is always the best option for leaf counting tasks.

There are several directions for future work, including training the same models on a larger dataset to account for differences in plant structure and leaves changing over time, as well as trying out different models of neural networks to see how they compare with each other on the task of leaf counting. Also, one interesting thing would be to try to generalize the model to be able to count not only the leaves of Arabidopsis thaliana but some other plants as well.

REFERENCES

[1] S. Minaee, Y. Boykov, F. Porikli, A. Plaza, N. Kehtarnavaz, D. Terzopoulos, “Image segmentation using deep learning: A survey”, IEEE Trans. on pattern analysis and machine intelligence, vol. 44, no. 7, pp. 3523-3542, 2021.

[2] G. Ciaparrone, F. L. Sánchez, S. Tabik, L. Troiano, R. Tagliaferri, and F. Herrera, F., “Deep learning in video multi-object tracking: A survey”, Neurocomp., vol. 381, pp. 61-88, 2021.

[3] D. W. Otter, J. R. Medina, J. K. Kalita, “A survey of the usages of deep learning for natural language processing”, IEEE trans. on neural networks and learning systems, vol. 32, no. 2, pp 604-624, 2020.

[4] M. Malik, M. K. Malik, K. Mehmood, I. Makhdoom, “Automatic speech recognition: a survey”, Multimed. Tools and App., vol. 80, pp. 9411-9457, 2021.

[5] I. Goodfellow, Y. Bengio, and A. Courville, “Deep learning”, MIT press, 2016.

[6] R. Moradi, R. Berangi, and B, Minaei, “A survey of regularization strategies for deep models”, Artif Intell Rev, vol. 53, pp. 3947–3986, 2020.

[7] L. Huang, J. Qin, Y. Zhou, F. Zhu, L. Liu and L. Shao, “Normalization Techniques in Training DNNs: Methodology, Analysis and Application”, IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 45, no. 8, pp. 10173-10196, Aug. 2023

[8] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,”, ICML, 2015.

[9] D. Ulyanov, A. Vedaldi, and V. Lempitsky. “Instance normalization: The missing ingredient for fast stylization”, arXiv:1607.08022, 2016.

[10] J. L. Ba, J. R. Kiros, and G. E. Hinton, “Layer normalization”, arXiv:1607.06450, 2016.

[11] P. Luo, Z. Peng, J. Ren, R. Zhang, “Do normalization layers in a deep ConvNet really need to be distinct?”, arXiv preprint arXiv:1811.07727, 2018.

[12] H. Nam, H. E. Kim, “Batch-instance normalization for adaptively style-invariant neural networks”, Advances in Neural Information Processing Systems, vol. 31, 2018.

[13] P. Luo, J. Ren, Z. Peng, R. Zhang, J. Li, “Differentiable learning-to-normalize via switchable normalization”. arXiv preprint arXiv:1806.10779, 2018.

[14] Y. Jiang, C. Li, “Convolutional Neural Networks for Image-Based High-Throughput Plant Phenotyping: A Review”, Plant Phenomics, vol. 2020, 2020.

[15] Z. Li, R. Guo, M. Li, Y. Chen, G. Li, “A review of computer vision technologies for plant phenotyping”, Computers and Electronics in Agriculture, vol. 176, 105672, 2020.

[16] Z. Štaka, M. Mišić, “Leaf counting in the presence of occlusion in Arabidopsis thaliana plant using convolutional neural networks”, Journal of Electronic Imaging, vol. 32, no. 5, 052407-052407, 2023.

[17] Y. Wu, K. He, “Group normalization”, Proceedings of the European conference on computer vision (ECCV), pp. 3-19, 2018.

[18] B. Chaudhury et al., "Study on deep convolutional neural networks for leaf counting," https://pal.iitpkd.ac.in/static/pdfs/wspa_chaudhury.pdf (accessed 26-12-2023).

[19] A. Dobrescu, M. Valerio Giuffrida, and S. A. Tsaftaris, "Leveraging multiple datasets for deep leaf counting,"in Proc. IEEE Int. Conf. Comput. Vision Workshops, pp. 2072–2079 (2017)