Applying New architecture to an existing deep leaf counting problem

ASAF NEISI MINAEI, University of Saskatchewan

abstract

Additional Key Words and Phrases: deep learning, leaf counting, Xception

ACM Reference Format:
Asaf Neisi Minaei. 2020. Applying New architecture to an existing deep leaf counting problem. 1, 1 (May 2020), 8 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Nowadays, one of the challenges that we humans are facing is "expanding global demand for food". Plant phenotyping is identified as a critical sector to increase plant resistance and productivity. Biologists consider the number of leaves in a plant as one of the metrics of phenotyping [6]. **Phenotype** is the functional plant body, and it is the result of dynamic interaction between **Genotype** and **environment**. These interactions determine plant performance and productivity. Phenotyping is the quantitative analysis of the actual plant traits [3].

The leaf count problem can be solved with different approaches in deep learning, including counting via leaf segmentation, and image to count regressor model. Our focus will be on regression approaches. The task is going to be performed on images of single plants of Arabidopsis and tobacco. The input of the task is top-down images of an Arabidopsis or a tobacco plant, and the output is the number of leaves of those plants.

In this research, we tried to train a fully connected architecture to set a baseline for our measurements. Afterward, we modified a convolutional architecture in order to exploit it to solve the leaf counting problem. Furthermore, we used data augmentation techniques as well to measure its effect on the training procedure.

The rest of this paper is organized as the following. Section 2 is about the methodologies used in training the models and processing datasets. In section 3 we explain the environment of the tests and the several experiments that we ran. Section 4 is expressing the results of our experiments. Section 5 will present a summary.

2 METHODOLOGY

This section introduces the models utilized in the experiments. Alongside that, it determines the algorithm for training, hyper-parameters, and metrics. Also, there is a brief introduction and analysis of the dataset we used to train our model.

2.1 The Network Architectures

The Architecture we will use for baseline will be a model with three fully connected layers. The first layer will have 1024 nodes, the second one will have 512 nodes, and the last layer will have just one in which the output will be the prediction of the leaf count. The method for parameter initialization is Glorot initialization [9]. As the

Author's address: Asaf Neisi Minaei, asn586@usask.ca, University of Saskatchewan.

XXXX-XXXX/2020/5-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

regularization method, we will use dropout with a 0.2 rate before layers 2 and 3. An L2 method will be in the second layer, as well. All activation functions are ReLU in all three layers.

Another model that will be exploited is a modified Xception [5]. The classification layers are going to be removed, and the same layers as our base architecture will be added, and every setting will be the same. Xception is a deep convolutional architecture developed by google. It uses a depthwise convolution, followed by a pointwise convolution [8]. For initializing weights of this part of the model will be the pre-trained weight on ImageNet. The original architecture of Xception is depicted in figure 1.

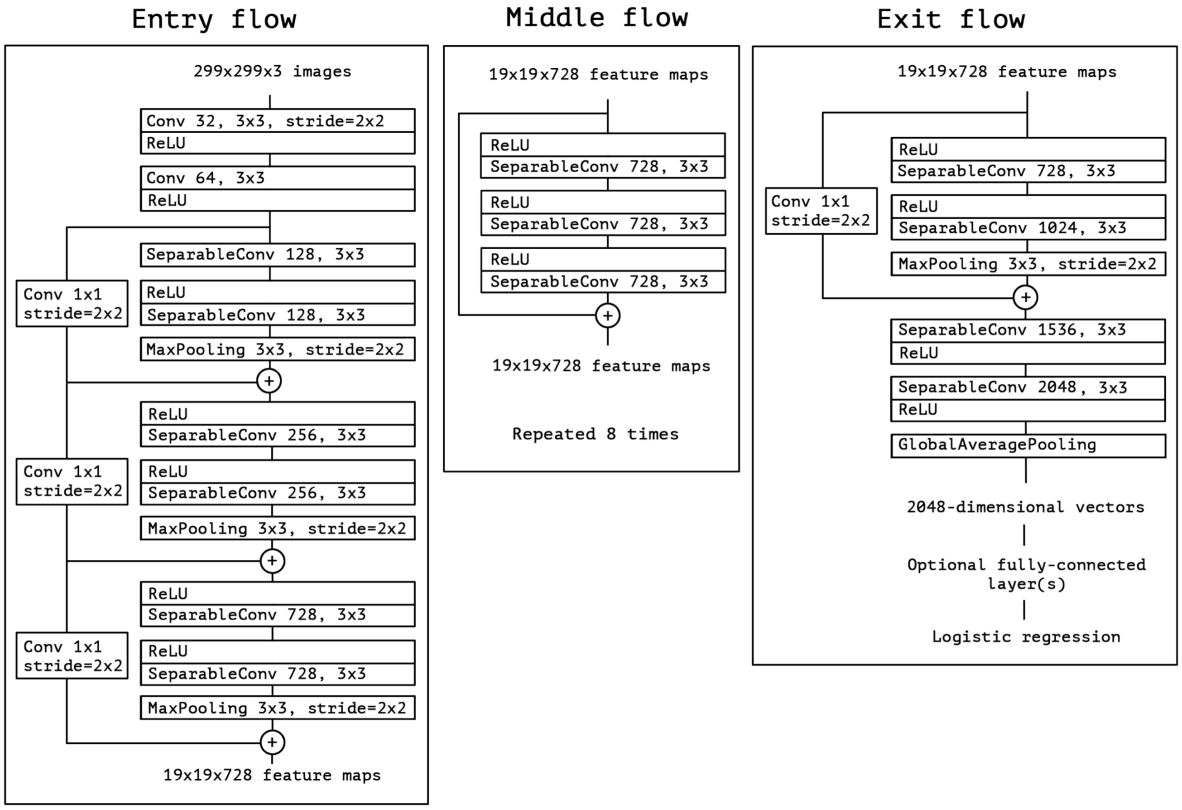

Fig. 1. The original Xception architecture[5]

2.2 Datasets

The data type that we are working on is top-view images from rosette plants (tobacco plants and Arabidopsis plants). The dataset is comprised of four sub-datasets A1 to A4 [2, 4, 10]. A1 and A4 are images of wild wild-type Arabidopsis plants, A2 comprise of four mutant types of Arabidopsis plants, and A3 contain images of young Tobacco plants [6]. We add all these datasets together to form a bigger dataset called "Ac".

The image size for each Sub-dataset is different because they are produced in different labs. All images are RGB color; however, they are different in size. A1 available images are cropped, and the size is reduced base on the size of the actual plant in the picture. The same applies to A2; the photos are cropped to contain only the

plant and the soil background. A3 remains unchanged; The size of the images is 2448*2048. A4 has an image size of 441*441. The number of samples is 120, 165, 62, and 624 for A1 to A4 [1], respectively. Each sample is labeled with the amount of the leaves that appeared in the picture.

The figure 2 present the distribution of all sub-datasets. It is quite important to notice the partial similarity between the distributions of A1 and A4. By scaling up the A1 curve, we match it with the left part of the A4. Also, we should consider the resemblance between A3 and A4 as well. In the result section, we will see the effect of these similarities when we combine the training set to reach Lower MSE. A2 has a distribution more like a normal distribution and slightly different from other sub-datasets.

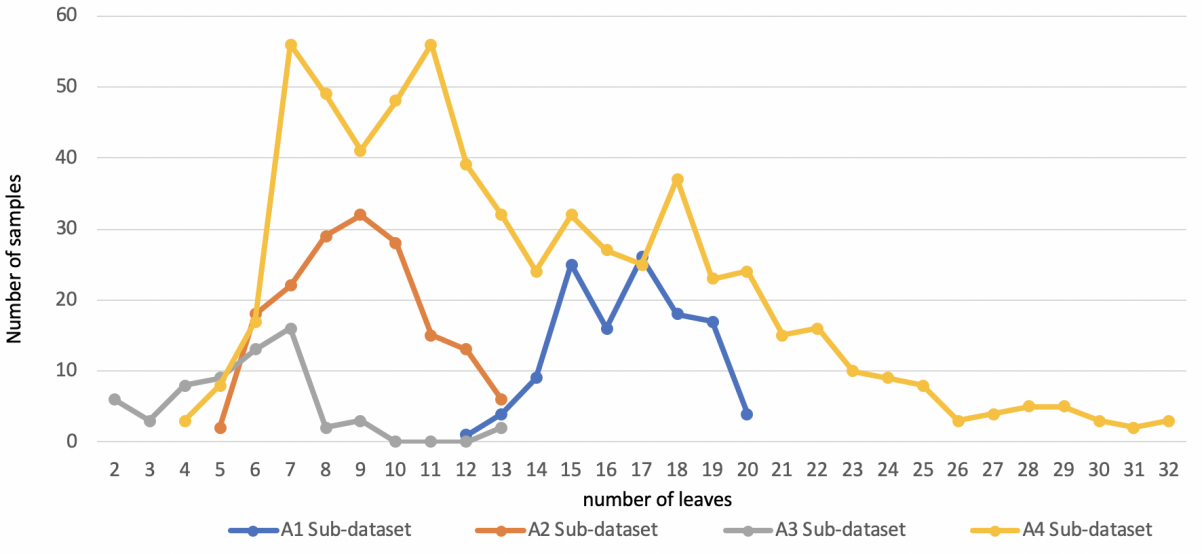

Fig. 2. The quantification of distribution of all sub-datasets

2.3 Dataset Pre-processing and Augmentation

The Xception architecture has a fixed number of input channels for the pre-trained version. We used the standard input size for that model and resize every image. The standard input size for Xception is 299*299*3. Furthermore, the contrast stretching was applied to the images to increase the contrast on darker ones. Other techniques of histogram stretching (including histogram equalization and contrast stretching) were not proper because they had a huge negative effect in the value of the metrics The split of the dataset is 75% for training, and 25% for testing. We utilize measures to augment the dataset we possess. It extenda the training part of the dataset to 12 times. Random rotation between 0° to 170° is applied to images. 0% to 10% random zoom is used on pictures, and there were a random horizontal or a vertical flip as well.

2.4 Training Methods, Metrics, and Hyperparameters

The training algorithm that we used is Adam [9]. Adam utilizes the momentum of first and second derivation. It also uses an average of past gradients, which exponentially decaying. The advantages of using this method

[1]Due to some problems, it was not possible to reach the original A4 dataset. However, we used a version available from CVPPP 2017 challenge training set. We still call that A4 for making it simple to read.

are being fast and rectifying the vanishing learning rate. However, this algorithm might be computationally expensive [7]. The learning rate was 0.001. The loss function is MSE (mean squared error). Furthermore, for evaluating the final result, we used the difference in counts (*DiC*) and the absolute difference in counts (|*DiC*|) to compare the architecture of the original paper and our architectures as well. In the next paragraph we will define these metrics. Early Stopping has been applied; the step limit was 5. Also, the batch size would be set to 6.

To measure the results, we used *DiC* as it used in the CVPPP LCC challenge [1]. The measure would be *DiC* for each image. It is calculated by the output of the counting algorithm minus the ground truth. For a set of images, we would measure the average and the standard deviations. The goal of this metric is to estimate overall bias. The |*DiC*| is going to be calculated similarly. The only difference is that the final result would be the absolute difference. The goal is to estimate absolute error.

3 EXPERIMENTS

Due to some problem[2] we had to run our experiments on a virtual environment provided by Google Colab[3]. Underlying hardware were used for calculation was CPU and GPUs. Because of the google policy it is not guaranteed that during multiple experiments we have access to the same hardware. But base on the FAQ[4] page of the Colab, we can name some available hardware. The are often Nvidia K80s, T4s, P4s and P100s.

We ran multiple experiments with different settings. To have a baseline for building up comparisons we first trained our base model described in subsection 2.1 without any augmentation and with every sub-dataset and the total dataset comprised with all sub-datasets. After that the same experiments was conducted. The Only difference was the model. We used Xception as underlying architecture for training. In the next set of experiments we tried use dataset augmentation on the training part of the sub-datasets and used Xception as the model.

We continued our experiments by using Ac's training part as our training set and trained our Xception model using each of sub-datasets testing part separately. These series of experiments was held with and without dataset augmentation. The results of all experiments are presented in the next section.

4 RESULTS AND DISCUSSION

This section will discuss the results first by each sub-datasets then present an overall analysis on the matter. Note that the "+" sign in the dataset columns in the tables showing the results is used in cases we applied dataset augmentation based on what we discussed in subsection 2.3. The information in the rows with ResNet50 as their model is from a research conducted on the same dataset [6]. We mentioned them here for better comparison and analysis.

4.1 Sub-dataset A1

Table 1 shows the results of experiments that we used the A1 test part as the test set. All the models are biased to estimate less number of leaves based on the *DiC* metric. However, the Xception model trained with Ac without data augmentation has the best result among our experiments in this metric. The average of *DiC* for this experiment was pretty close to zero. Its standard deviation[5] is around 1.35, which is the minimum in all experiments on A1. Considering |*DiC*| metric, the best reach by Xception model when trained on A1 without dataset augmentation. Two test results closely follow it . the first ran with the same dataset with data augmentation and the second one trained Ac dataset without data augmentation.

[2]COVID-19 pandemic and lockdown
[3]www.colab.research.google.com
[4]https://research.google.com/colaboratory/faq.html
[5]which form now on we referred it as *std*

Table 1. Test result on test set of A1

Train Set	Model	*DiC*	$\|DiC\|$	*MSE*	Best Epoch
A1	Base	-0.17(1.66)	1.43(1.08)	3.63	10
A1	Xception	-0.30(1.39)	1.10(0.96)	2.37	7
$A1^+$	Xception	-0.30(1.46)	1.10(1.01)	2.23	2
$A1^+$	ResNet50	-0.81(0.85)	0.94(0.70)	1.38	≈ 50
Ac	Xception	-0.03(1.35)	1.17(0.78)	2.23	7
Ac^+	Xception	-0.27(1.59)	1.33(0.90)	2.60	6
Ac^+	ResNet50	-0.28(0.80)	0.53(0.66)	0.72	≈ 50

Table 2. Test result on test set of A2

Train Set	Model	*DiC*	$\|DiC\|$	*MSE*	Best Epoch
A2	Base	0.60(1.54)	1.55(1.04)	3.60	4
A2	Xception	0.38(1.46)	1.43(1.03)	3.24	4
$A2^+$	Xception	-0.05(1.99)	1.74(1.06)	4.18	3
$A2^+$	ResNet50	-2.38(2.69)	2.38(2.69)	12.88	≈ 50
Ac	Xception	0.38(1.53)	1.47(1.12)	3.62	13
Ac^+	Xception	-0.03(1.83)	1.53(1.05)	3.47	6
Ac^+	ResNet50	-0.28(1.11)	0.88(0.78)	1.38	≈ 50

Best *MSE* is reached by to tests with training sets $A1^+$ and Ac with a value of 2.23. This similarity between $A1^+$ and Ac in the metrics values may well be caused by the plant type of the dataset A1 and A4. As we mention in the subsection 2.2, these datasets have the same plant type, So when we combine all datasets into Ac, the most part has the plant type of A1 and A4. Xception training on Ac^+ had performed better than the base model; However, not as good as the others. The reason might be the overfitting caused by our expansion scale, which was 12.

The ResNet model in both $A1^+$ and Ac^+ cases had better results, but with substantially lot more epochs. Also, it didn't perform very well when it ran without any augmentation. It reached 30.59 in *MSE* metric [6] which is way of the charts.

4.2 Sub-dataset A2

We can see the test result on A2's test part in the table 2. By looking at the *DiC* Column, an exciting fact would appear. Whenever dataset augmentation has not been used, models tend to have overestimations. On the contrary, using dataset augmentation causes underestimations or biases close to zero. Best result in $|DiC|$ and *MSE* within our experiments reached by the Xception model trained on A2 without any augmentation. It is closely followed by the result of the training on Ac^+ with the same model.

It is not very obvious why the training $A2^+$ with ResNet reached such a high MSE. It is even more than the base model result. It is noteworthy that we should appreciate the fact that devices utilized to run the experiment were different. Consequently, the result might be slightly different. However, with that scale of difference, it still not usual. ResNet50 training on Ac^+ has the best result among all tests on A2.

Table 3. Test result on test set of A3

Train Set	Model	*DiC*	$\|DiC\|$	*MSE*	Best Epoch
A3	Base	0.42(2.31)	1.97(1.27)	7.75	9
A3	Xception	-0.08(2.47)	2.03(1.44)	7.58	9
$A3^+$	Xception	-0.81(3.00)	2.31(2.08)	9.68	2
$A3^+$	ResNet50	-0.57(1.50)	1.43(0.73)	2.57	≈ 50
Ac	Xception	0.38(1.53)	1.47(1.12)	7.00	2
Ac^+	Xception	0.38(2.8)	2.00(2.00)	8.00	6
Ac^+	ResNet50	0.71(1.03)	0.71(1.03)	1.57	≈ 50

Table 4. Test result on test set of A4

Train Set	Model	*DiC*	$\|DiC\|$	*MSE*	Best Epoch
A4	Base	-0.13(1.63)	1.40(1.03)	3.63	9
A4	Xception	0.36(1.60)	1.33(1.04)	3.23	9
$A4^+$	Xception	0.21(1.80)	1.40(1.15)	3.33	7
$A4^+$	ResNet50	0.10(1.14)	0.91(0.73)	1.54	≈ 50
Ac	Xception	-0.18(1.63)	1.37(1.01)	3.17	13
Ac^+	Xception	-0.24(1.94)	1.43(1.34)	3.90	11
Ac^+	ResNet50	0.12(0.99)	0.69(0.73)	1.01	≈ 50

4.3 Sub-dataset A3

As it is presented in table 3, in all cases, when we used A3 for training a convolutional model, the models were biased with underestimation. On the other hand, when we used Ac for training, models had overestimation. The exception model trained on Ac without augmentation reached the best result in $|DiC|$ and *MSE* metrics among our experiments. When we used augmentation, its results were worse even than the base model.

The difference between this sub-dataset with other sub-datasets additional to the plant type is the resolution of the images. Also, the number of samples was quite low. Hence the high *MSE* in our experiments seems to be reasonable. However, the ResNet50 had the best performance on the A3 training set in both $A3^+$ and Ac^+ datasets trains.

4.4 Sub-dataset A4

In model training with A4's testing part, the best performance in *DiC* among our experiments belongs to the base model. The second best is the Xception model trained on Ac without augmentation. In $|DiC|$, the best average belongs to the Xception model trained on A4 without augmentation, but the best std reached by Xception trained on Ac without Augmentation. The best in *MSE* is the result of Xception training on Ac with no dataset augmentation. All results for this series of test are shown in table 4.

Like most cases, the best performances trained by A4 An Ac are reached by ResNet50 Model. In all three metrics, ResNet50 shows compelling results comparing to all of our training settings. Both *DiC* and $|DiC|$ are at their minimum when trained by the ResNet50 and when the Ac^+ used, *MSE* is just 1.01. Considering the distribution of the A4, as we mentioned in section 2.2, the result is outstanding.

Table 5. Test result on test set of Ac

Train Set	Model	*DiC*	$\|DiC\|$	*MSE*	Best Epoch
Ac	Base	0.53(1.83)	1.70(1.21)	5.03	8
Ac	Xception	-0.12(1.67)	1.51(1.03)	3.88	7
Ac^+	Xception	-0.26(1.97)	1.53(1.27)	4.11	4

4.5 All datasets Ac

Ac is comprised of all other datasets and contains all the images available form all data sets. It has a significant value, which is the diversity in plant type. Although most pictures are from A4 and A1 with the same type, the other datasets play a considerable role in diversity and not be monotonic. Hence, the performance of the models is valuable to the cause of the research.

The best result in all metrics is reached by the Xception model train on Ac. It is followed by the same model trained on Ac^+. The worst case belongs to the base model.

4.6 Discussion

In our experiments, dataset augmentation had not the promising effect we expected on the results based on the experiences in [6]. Either the data augmentation does not work on Xception as same as ResNet50, or it should be done with different setups more proper to Xception. The settings of augmentation were similar to the ResNet50 model.

The Ac dataset without augmentation trained on the Xception model had the best performance in almost all the cases we ran. The apparent reason is that when we have a more diverse and higher quantity of the samples, the results would be more precise. One of our models' advantages is they reach their best in a low number of epochs, and consequently in less time compare to the ResNet50 model.

5 CONCLUSION AND FUTURE WORK

In this work, we set up two models, base three-layer fully connected network and modified Xception. The base model was set to provide a baseline for our experiments. In our experiments, the convolutional model Xception almost all the time had better performance over the base model except in some cases that dataset augmentation caused overfitting.

In future researches, we can alter the setting of the dataset augmentation and run several tests to find the best setting for the Xception method. Another suggestion is to implement multiple convolutional models and find their accuracy over multiple datasets divisions, then used a weighted average of their output as the ultimate output. We can run a test on the train set and test set distributions to base a more analytical cornerstone for our setting of hyperparameters. All in all, Xception showed a promising performance with lower epochs, which with better adjustment of hyperparameters, may well result in an acceptable outcome.

REFERENCES

[1] 2017. DATA DESCRIPTION AND FURTHER DETAILS. http://www.plant-phenotyping.org/CVPPP2017-challenge
[2] 2020. INTRODUCTION: PLANT PHENOTYPING DATASETS. http://www.plant-phenotyping.org/datasets
[3] 2020. Plant Phenotyping. https://www.helmholtz-muenchen.de/biop/plant-phenotyping/plant-phenotyping/index.html
[4] Jonathan Bell and H Dee. 2016. Aberystwyth leaf evaluation dataset. 168158, 17-36 (2016), 2. https://doi.org/10.5281/zenodo.168158
[5] F. Chollet. 2017. Xception: Deep Learning with Depthwise Separable Convolutions. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 1800–1807.

[6] Andrei Dobrescu, Mario Valerio Giuffrida, and Sotirios A. Tsaftaris. 2017. Leveraging Multiple Datasets for Deep Leaf Counting. https://doi.org/10.1109/iccvw.2017.243
[7] Sanket Doshi. 2019. Various Optimization Algorithms For Training Neural Network. https://medium.com/@sdoshi579/optimizers-for-training-neural-network-59450d71caf6J.
[8] Maël Fabien. 2019. XCeption Model and Depthwise Separable Convolutions. https://maelfabien.github.io/deeplearning/xception/#
[9] D Kingma and J Ba. 2015. Adam: A method for stochastic optimization In: Proceedings of International Conference on Learning Representations. (2015).
[10] Massimo Minervini, Andreas Fischbach, Hanno Scharr, and Sotirios A. Tsaftaris. 2016. Finely-grained annotated datasets for image-based plant phenotyping. 81 (2016), 80–89. https://doi.org/10.1016/j.patrec.2015.10.013