Image augmentation for object detection of grapevines

Tatiana Sofía Parlanti[1], Alejandro Martín Lobos[1,2], Luis G. Moyano[2,3], and Emmanuel N. Millán[2,4]

[1] Facultad de Ciencias Exactas y Naturales, Universidad Nacional de Cuyo, Mendoza, Argentina
[2] CONICET, Mendoza, Argentina
[3] Grupo de Física Estadística e Interdisciplinaria, Centro Atómico Bariloche, Bariloche, Argentina
[4] ITIC and ITU, Universidad Nacional de Cuyo, Mendoza, Argentina.
emmanueln@gmail.com

Abstract. Machine Learning methods are widely used for data analysis in various areas. In this work we use Neural Networks for image analysis in order to detect grape fruit clusters. A set of manually tagged images is built and a comparison is made between different data augmentation techniques in order to analyse the best way to expand the image set. The technique presented here obtained up to 13% better detection performance starting with only 100 images for training. The types of transformations and filters that worked the best for these images are discussed. In addition, training and detection times in five different hardware infrastructures, both CPU and GPUs, are briefly discussed.

Keywords: machine learning, neural networks, deep learning, object detection

1 Introduction

Machine Learning (ML) is the field of study that allows computers to have the ability to learn without having been explicitly programmed to do so [10]. ML is often associated with automatic detection of patterns or useful information in data. The use of Machine Learning for data analysis is widely used in various areas, from social to health and engineering [4,7]. This is particulary the case in the area of automatic object recognition, which has direct applications in agricultural processes [11].

A general need for agricultural producers is to have an adequate process for estimating harvest; in particular, in our local region, grape harvest. Traditionally, the most accurate and fastest methodology for estimating the harvest of grapes is given by the count of bunches per plant and weight estimation. The extrapolation of the individual result to all plants determines the projected production value. Direct visual estimation is also used, but is subjective. To automate the analysis of grapevine images with the objective of improving the

estimation of the grape harvest forecast is seen as a problem that can be addressed by ML, especially given the advances seen in recent years. For example, [1] presented an algorithm based on mathematical morphology and pixel classification for grapevine berry counting, however images were taken placing a dark cardboard behind the cluster, which is difficult for an operator to implement. Then, yield prediction based on RGB images acquired "on-the-go" from an all-terrain vehicle was presented in [2], however the capture of images was during the night, which means that operators have to work in an unusual time window. Also, ML has been used to identify apples at different times of growth using YOLOv3 [14], and [3] presented an adaptation of crowd-counting algorithms for the estimation of harvest yield for grape berries.

The relevant steps to automate the forecast process are: first, image acquisition, then automatic cluster identification, and finally, cluster analysis for harvest forecast. In this work we focus on the first stages of this process: image acquisition, labeling, training and detection with Neural Networks.

A common limitation of Neural Networks (NN) is that they typically require hundreds to thousands of training images for the correct detection of objects. To address this, we first capture images on various vineyard farms, which were then hand-tagged. A NN was trained using the images obtained from vineyards, and was later used to generate detection models. A comparison of the resulting detections, obtained from the same set of images made by trained models with 100 and 500 photographs, was made. Afterwards, a series of transformations were applied to 100 images in the first set in order to increase the number of data, which were used to train again and obtain new, improved models. Finally, the detection percentages of these models were compared against the detections of the two initial sets of images.

The work is organized as follows. Section 2 describes the computational tools we use to carry out the analysis (2.1), our images (2.2), and the filters and transformations applied to them (2.3). We then show and discuss our results in Sec. 3. Finally, we elaborate conclusions and state future lines of research in Sec. 4.

2 Materials and Methods

The Python ImageAI library [6] was used to carry out the different analyses, since it allows training custom object detection models on image datasets that are in Pascal VOC annotation format. ImageAI uses the YOLOv3 architecture [8,9], and only requires a few lines of code.

For the training stage, a pre-trained model provided by the same ImageAI library was used, as faster progress is obtained using transfer learning compared to training from scratch. The number of epochs used in the training was 100. Also, batch size and learning rate were set at 2 and 0.0001 respectively. Different models were generated, which were then evaluated by calculating the mAP score (mean Average Precision) of each one in order to pick the most accurate, com-

puted over the validation image set. Source code is available from the authors via email request.

2.1 Machine Learning Algorithms

You Only Look Once (YOLOv3) algorithm [8,9] is a real-time object detection system, which resizes the input images to 416×416, has 53 convolutional layers in its NN architecture and thresholds the resulting detections by the model's confidence.

For its operation the NN divides the image into a grid, and predicts bounding boxes and the level of certainty of each box for each region. The confidence scores quantify the likeliness that a given region contains an object, as well as its accuracy.

2.2 Image dataset

The set of images used was assembled from photographs taken of espaliers from different vineyards, two of which are in the Valle de Uco. Additionally, bunches from espaliers belonging to the Instituto Nacional de Tecnología Agropecuaria (INTA), in Lujan de Cuyo were photographed. Two smartphones were used to capture the images (with 25- and 13-megapixel sensors). The varieties photographed included Malbec and Cabernet Sauvignon, both dark-colored, used to make red wine. Close-ups of the bunches and general plans of the espaliers were made.

Each image was then annotated in Pascal VOC format, with the labelImg tool [15], which is straightforward to apply, and generates files with *.xml* extension that store the coordinates of the upper left and lower right vertices of each manually generated annotation. In this way, more than 800 images were tagged, corresponding to more than 3000 grape bunches.

These images, and their respective annotations, were separated into three sets, which we will call *bunch*600, *bunch*120 and *detect*, detailed below:

- **bunch600**: consists of 500 images used as a training set (2253 annotated bunches), and 100 images used for the validating set (472 annotated bunches). These sets include photographs taken with bunches in the foreground and in the background;
- **bunch120**: consists of 100 images used as a training set (593 annotated bunches), and 20 images used for the validation stage (102 annotated bunches). In this case all the images used were in the foreground;
- **detect**: consists of 100 images (641 annotated bunches), which were used after the training to test the chosen model, generated in the first stage.

2.3 Image filters and transformations

Increasing the data present in the dataset (known as *image augmentation*) is a common technique that seeks to increase the number of images in a sparse training set, based on transformations such as rotation, translation, noise addition,

etc. [5,12]. Image augmentation relies on how NNs interpret image characteristics: small modifications are interpreted as completely different images by the NN.

In this sense, a series of filters and transformations were applied to the images of *bunch*120 using the ImageMagick software [13]. Two types of tests were performed. On the one hand, tests the training and validation set was duplicated with the reflection transformation, the solarize filter and the contrast limited adaptive histogram equalization were used, as detailed below.

- **mirror**: horizontal mirroring;
- **solarize**: when solarizing, the color of the pixels whose luminosity exceeds a certain limit is inverted, in this case the limit was set at 60%;
- **clahe**: adaptive histogram equalization improves local contrast and enhances edge definitions in each region of an image. To apply clahe, a grid of defined pixel width and height tiles is used, where each box contains a fixed number of classes or bins, and a limit is established for localized contrast changes. In this case the boxes were 5% of the width and height of the image, and contained 128 bins each. In addition the contrast limit was set at 4.

On the other hand, the training and validation set was quintupled by applying four different values of the same type of filter:

- **laplacian noise**: with attenuation values 1.5, 3.0, 4.5 y 6.0;
- **paint**: blurred from "drops" of paint that blends the colors present in an environment and turns them into a single-color area. Used radius values: 3, 5, 7 y 9.

The *noise*, *paint*, *solarize* and *clahe* filters modify the images, but not the labels of the grape clusters. On the other hand, the *mirror* filter requires a transformation to be applied to *.xml* files, so labels are also mirrored. Thus, from the 100 images of *bunch*120, five new training sets were obtained, three around 200 images each, and two around 500 images each (matching, in number, the *bunch*600 dataset). Similarly, the same transformations were applied to the validation images, to maintain the proportions between both sets. Fig. 1 shows the types of filters applied to the original image (a).

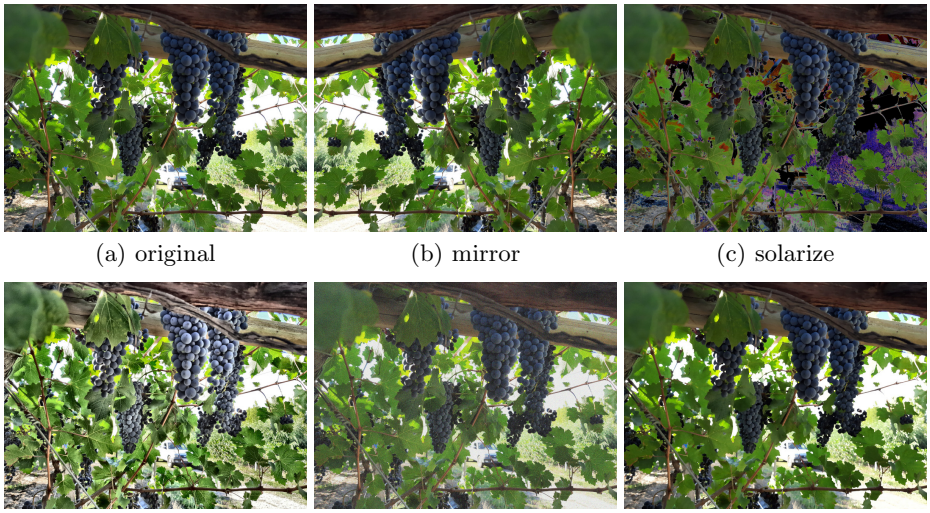
3 Results and Discussion

Results of the detections from chosen models obtained by the NN are presented.

3.1 Hardware and Software Description

The analysis of the images is carried out mainly using GPUs. The following equipment belonging to the Toko cluster of the FCEN UNCuyo was used:

- Toko computing node with four AMD **Opteron** 6376 CPU, with 16 CPU cores at 2.3GHz each (64 cores total), 128 GB of RAM.



(a) original (b) mirror (c) solarize

(d) clahe (e) laplacian noise (f) paint

Fig. 1. Example of an original image and its transformations.

- Toko computing node with two AMD **EPYC** 7281 CPU, with 16 CPU cores at 2.1GHz each (32 cores total), 128 GB of RAM.
- Toko computing node with one AMD **Ryzen** 7 2700 with 8 cores and 16 threads running at 3.2 GHz with 64 GB of DDR4 RAM memory.
- Two AMD FX-8350 nodes with 8 cores running at 4 GHz and 32GB of DDR3 RAM. One node with an NVIDIA GeForce GTX **Titan X** (Maxwell GM200 architecture) with 12 GB of memory, and another node with an NVIDIA GeForce GTX **Titan Xp** (Pascal GP102) with 12 GB of memory.

The Toko cluster is running Slackware Linux current 2020 64bit with kernel 5.4.28, gcc 9.2, NVIDIA driver 410.73 and CUDA 10. The software used to perform the analysis was developed around the imageAI 2.1.5 library with tensorflow-gpu 1.13.1, keras 2.2.4, and python 3.7.2. In the section 3.3 we discuss the performance of the NN running in this hardware.

3.2 Image detection

Frequently, the same cluster is divided at detection and reported as two or three different objects, depending on whether there are leaves or branches partially obstructing it. In this situation, it was considered that each one of these detections corresponded to the real cluster if the intersection between both rectangles (the detection and the hand-labeled rectangles) was greater than 25%.

Also, a manually tagged cluster is considered “well detected” if the rectangle detected by the NN and the tagged rectangle match 60% or more. If a labeled cluster corresponds to several smaller detected rectangles, it is considered “well

detected" if the sum of the surfaces in the small rectangles represents at most 60% of the real surface of the labeled rectangle.

As explained in Sec. 2.2, *detect* is the set of images that was used to test the models. It consists of 100 images, containing 641 labeled clusters in total. These images were tagged in order to compare them against NN detections, but the NN did not had access to the tags at any time during training. The Table 1 shows the difference in detections between *bunch*120 and *bunch*600, which have 100 and 500 training images, respectively. The name "originals" means that the images do not have any type of filter or transformation applied.

Table 2 compares the detections achieved by the combination of *bunch*120 (original images) and the application of a transformation type, which increases the number of images to 200 of training. On the other hand, the Table 3 compares the combinations of four filters, reaching 500 training images per test.

Table 1. *bunch*120 vs. *bunch*600

	originals *bunch*120	originals *bunch*600
Total clusters detected	455	593
Total clusters not detected	186	48
Total other detections	125	211
Percentage of correct detections	**70.98**%	**92.51**%
Average certainties	50.3	52.45
Standard deviation certainties	9.73	9.8

Table 2. Dataset detections that duplicate the training set

	originals + clahe	originals + mirror	originals + solarize
Total clusters detected	458	480	536
Total clusters not detected	183	161	105
Total other detections	80	123	217
Percentage of correct detections	**71.45**%	**74.88**%	**83.61**%
Average certainties	51.38	55.92	54.45
Standard deviation certainties	11.38	11.25	13.82

Table 3. Dataset detections that quintuple the training set

	originals + laplacian	originals + paint
Total clusters detected	454	467
Total clusters not detected	187	174
Total other detections	108	86
Percentage of correct detections	**70.82**%	**72.85**%
Average certainties	46.87	48.07
Standard deviation certainties	8.57	13.29

Regarding the level of certainty, a distinction is made between detections that correspond to clusters, from detections of other objects that could —or not— be partial-view clusters, not considered in the labeling. The different tables, in addition, report the average and standard deviation of the certainties of the detections that correspond to real clusters. Figure 2 shows the percentages of certainty of the detections obtained after training with images of *bunch*120 and *bunch*600, without the application of transformations or filters. On the left,

graphs (a) and (c) take into account the certainties for labeled clusters, while graphs (b) and (d) on the right take into account detections from other objects.

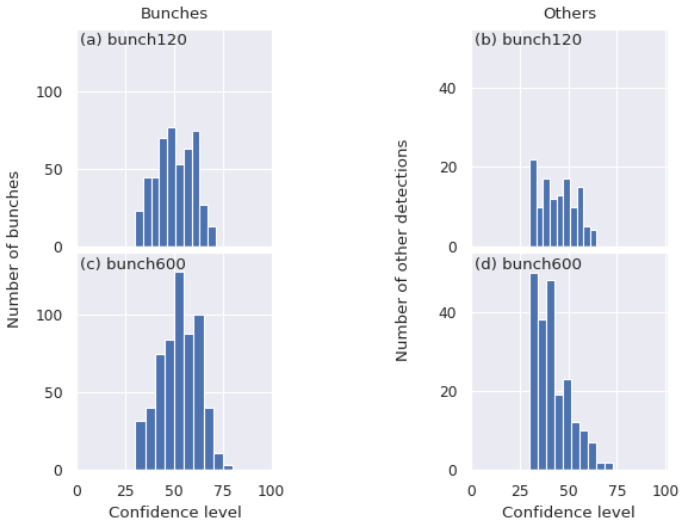


Fig. 2. Percentages of certainty of the detections obtained after training with the original images of *bunch*120 (above) and *bunch*600 (below).

Similarly, Fig. 3 has on its left the cluster detections, obtained from the training carried out with the images of *bunch*120 together with the reflection transformation (a), next to the solarize filter (c), and next to the clahe equalization (e). On the right, instead, are represented the detections of other objects. Likewise, Fig. 4 shows the graphics corresponding to the training with images with Laplacian filter and paint.

As expected, the NN reaches a higher detection of 92% from the dataset *bunch*600, while detection from *bunch*120 only reaches 70%. However, the application of some *image augmentation* techniques leads to an increase in the number of detections. Duplicating the original dataset using filters such as mirroring or solarizing achieves an increase from 4% to 13%. However, quintupling the original dataset only accounts for 2% more detection when using the paint filter.

It is remarkable that a higher detection rate was obtained among the original images plus their mirrors (200 images), than the original ones plus four paint filter applications (500 images). Simply duplicating the data using this particular transformation can achieve better results than quintupling the data, with the addition of less computer time.

On the other hand, the tests of solarize filter, mirrored images, clahe equalization and application of four values of paint are the ones that present higher percentages of certainty, even higher than those of *bunch*600 . However, the use of solarize is also the one that leads to additional “other detections”.

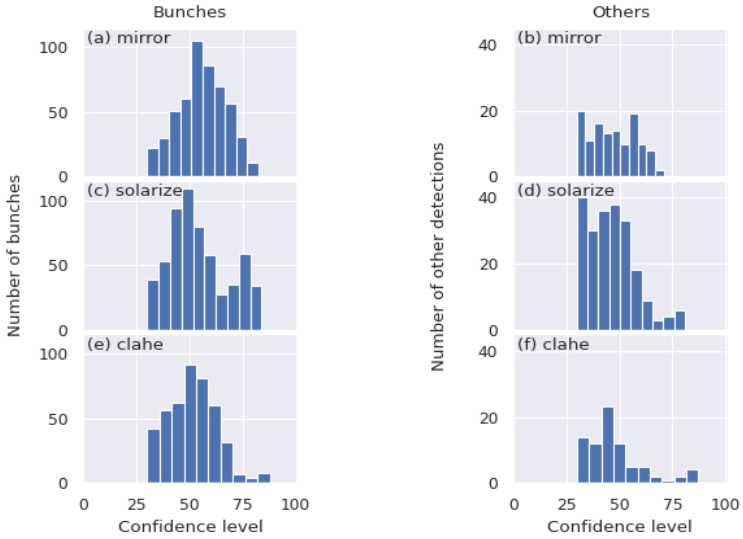


Fig. 3. Percentages of certainty of the detections obtained after training with the original images of *bunch*120 together with their reflections (above), together with the images with solarization (middle), and together with the clahe equalization (below).

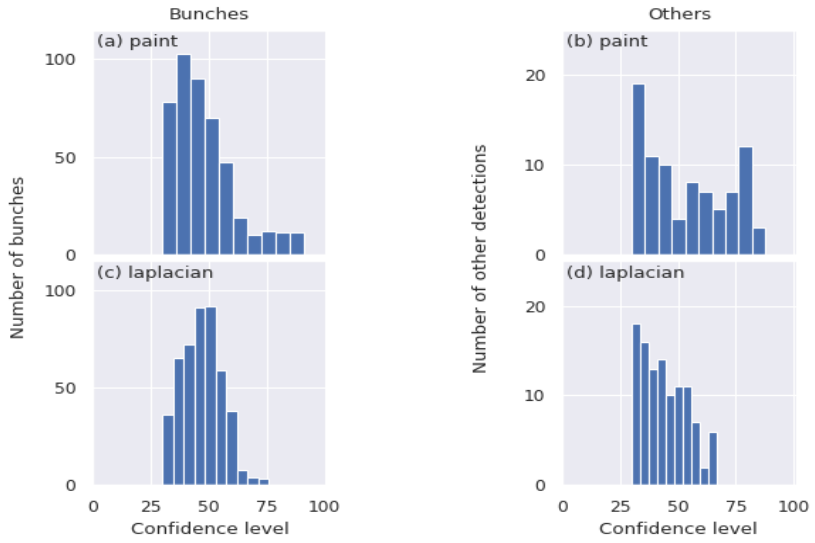


Fig. 4. Percentages of certainty of the detections obtained after training with the original images of *bunch*120 together with four values of paint (above) and four values of Laplacian noise (below).

3.3 Computational performance

The estimation of performance was a secondary objective of this work. The analysis was performed in five different hardware infrastructures, both CPUs and GPUs. For this analysis the data set *bunch*120 was used and the time is equivalent to one experiment. Table 4 shows that the Titan Xp GPU, as expected, has the best performance being ∼3 times faster than 64 cores of the Opteron 6376 node and ∼1.4 times faster than the previous generation Titan X GPU. The two GPUs have superior performance to the fastest CPU, the AMD Epyc 7281.

The training time for 100 experiments (section results 3.2) on the Titan Xp GPU, from the original *bunch*120 image set and mirror (200 images) took ∼11 hours and original *bunch*120 and 4 filters paint (500 images) took ∼31 hours. In the case of *bunch*600 (500 images) the training took ∼30 hours. Detection time is ∼1.5 seconds per image using the model generated by *bunch*120 + mirror on the same Titan Xp GPU. Due to these training times the number of filters that could be tested were limited as the use of the Toko cluster is shared amongst other users.

Table 4. Mean time in seconds for five separate training executions in five different CPUs and GPUs infrastructures.

Hardware	Mean time (s)	Standard deviation
Opteron	1652	35
Ryzen	1080	29
Epyc	962	21
Titan X	787	7
Titan Xp	563	12

4 Conclusions and Future Work

Cluster detection models were trained from new constructed datasets of 100 and 500 images each. Their detection percentages were compared, as well as the models obtained after using data augmentation techniques. With the methodology presented in this work, it was possible to increase the number of detections by 13%, although it was not possible to match the detection percentage obtained from the training with 500 images. Despite expectations, not all possible transformations lead to better results, as in the case of Laplacian noise application. Although a greater number of detections is achieved from a large training set, the difference that can be obtained from the application of some specific modifications to a smaller dataset is remarkable.

As future work, we expect to generate new models from the application of different transformations in the images, such as new translations or rotations, and apply adversarial learning techniques to re-train the NN. The next step is to increase the number and variety of images in the *bunch*600 dataset to increase the number of images present in the dataset; we expect that such a step will decrease the number of “other detections”.

Acknowledgments

We acknowledge Ing. Marcos Montoya and Ing. Julieta Dalmasso from INTA, Scattareggia family and Coletto Vineyard for giving us access to their grape fields. We acknowledge support from CONICET and SIIP Type 4 UNCuyo grants. This work was performed in the Toko Cluster from FCEN-UNCuyo, that is part of SNCAD, MinCyT, Argentina.

References

1. Aquino, A., Diago, M.P., Millán, B., Tardáguila, J.: A new methodology for estimating the grapevine-berry number per cluster using image analysis. Biosystems Engineering 156, 80–95 (apr 2017)
2. Aquino, A., Millan, B., Diago, M.P., Tardaguila, J.: Automated early yield prediction in vineyards from on-the-go image acquisition. Computers and Electronics in Agriculture 144, 26–36 (jan 2018)
3. Coviello, L., Cristoforetti, M., Jurman, G., Furlanello, C.: In-field grape berries counting for yield estimation using dilated cnns. arXiv preprint arXiv:1909.12083 (2019)
4. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature 521(7553), 436–444 (may 2015)
5. Mikolajczyk, A., Grochowski, M.: Data augmentation for improving deep learning in image classification problem. In: 2018 International Interdisciplinary PhD Workshop (IIPhDW). IEEE (may 2018)
6. Moses, Olafenwa, J.: Imageai, an open source python library built to empower developers to build applications and systems with self-contained computer vision capabilities (mar 2018–), `https://github.com/OlafenwaMoses/ImageAI`
7. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Müller, A., Nothman, J., Louppe, G., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Édouard Duchesnay: Scikit-learn: Machine learning in python. Journal of Machine Learning Research (2012)
8. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: Unified, real-time object detection. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE (jun 2016)
9. Redmon, J., Farhadi, A.: Yolov3: An incremental improvement (2018), `http\://arxiv.org/pdf/1804.02767v1:PDF`
10. Samuel, A.L.: Some studies in machine learning using the game of checkers. IBM Journal of Research and Development 3(3), 210–229 (jul 1959)
11. Shai Shalev-Shwartz, S.B.D.: Understanding Machine Learning. Cambridge University Pr. (2014), `https://www.ebook.de/de/product/22370006/shai_shalev_shwartz_shai_ben_david_understanding_machine_learning.html`
12. Shorten, C., Khoshgoftaar, T.M.: A survey on image data augmentation for deep learning. Journal of Big Data 6(1) (jul 2019)
13. The ImageMagick Development Team: Imagemagick, `https://imagemagick.org`
14. Tian, Y., Yang, G., Wang, Z., Wang, H., Li, E., Liang, Z.: Apple detection during different growth stages in orchards using the improved YOLO-v3 model. Computers and Electronics in Agriculture 157, 417–426 (feb 2019)
15. Tzutalin: LabelImg. Git code (2015), `https://github.com/tzutalin/labelImg`