Improve training on Convolutional Neural Networks with artificial samples.

João Pedro Knauer de Queiroz Verçosa [a]

[a] Department of Informatics, CTC – Scientific Technical Center, PUC-Rio – Pontifical Catholic University of Rio de Janeiro, Rio de Janeiro, Rio de Janeiro, Brazil, jpkqvercosa@gmail.com.

Abstract. The YOLOv8 algorithm has become a widely-used method for object detection due to its ability to detect objects in real time with high accuracy. However, one of the challenges faced in the field is the necessity of huge labeled datasets with sufficient diversity to train models to recognize objects effectively. In the case of Brazilian traffic signs, for example, such datasets are rare and often inadequate, which can lead to reduced accuracy when detecting these objects. To address this challenge, we propose the use of synthetic image generation to augment the available dataset of Brazilian traffic signs. By generating synthetic images, we aim to provide additional diversity to the dataset and improve object detection accuracy. Specifically, we utilize the YOLOv8 algorithm to train the original manually annotated images and the synthetic images, which were generated using various background images from the Microsoft COCO dataset. The effectiveness of the model is assessed by comparing its mean average precision (mAP) performance on both the real and the augmented datasets. This enables us to measure the impact of the synthetic image generation on the model's performance and determine how much it improves the accuracy of object detection. The results of this study contribute to advancing the field of object detection and provide insights into how synthetic image generation can be utilized to address the challenge of inadequate datasets in other areas as well.

Keywords. convolutional neural networks, Brazilian traffic signs, data augmentation, artificial image generation, deep learning.

1. Introduction

Convolutional Neural Networks (CNN) are an essential element in the fields of Artificial Intelligence (AI) and a powerful tool that play a key role in solving complex problems in computer vision and image recognition. CNNs are a type of deep neural network that is specifically designed to recognize patterns and features within image data. They have revolutionized the field of computer vision, enabling unprecedented levels of accuracy in tasks such as object detection, facial recognition, and scene understanding. [1]

The architecture of a CNN typically consists of several layers, each of which performs a specific function in the image recognition process. The first layer in a CNN is usually a convolutional layer, which applies a set of filters to the input image to extract features. Each filter detects a specific pattern or feature in the image, such as edges, corners, or blobs.

After the convolutional layers, pooling layers are often used to reduce the size of the feature maps and make the network more computationally efficient. This is frequently done by applying a max or average pooling operation to the features maps. Activation layers apply a non-linear function to the output of the previous layer, like *ReLU*, *sigmoid*, or *tanh*, allowing the network to learn more complex representations of the input image. [2]

Using the output of the previous layer, the final layer of a CNN is fully connected. These layers commonly use a *softmax* activation function to output a probability distribution over the possible classes [1][2]. In addition to the core layers, CNNs may also include other types of layers such as normalization layers, dropout layers and skip connections.

Although CNNs are an incredible statistical and mathematic structure, to work effectively, one of the primary requirements is large amounts of data to learn from. The more data the network has to train on, the better it can learn to recognize patterns and generalize to new, unseen images. However, collecting and labeling large amounts of data can be time-consuming and expensive. To address this challenge data augmentation techniques are often used to artificially increase the size of the training set. [3]

In this study, we direct our attention toward a collection of images that depict Brazilian traffic signs. At present, there exists a limited quantity of labeled datasets with this type of information, and thus, our initial dataset comprises only 256 images, annotated across 14 distinct classes [4]. The objective of our research is to expand the number of available images by creating artificial images and evaluating the efficacy of the YOLOv8 model [5] on those different scenarios, the original one, and the augmented dataset. By doing so, we aim to investigate the performance of this model with limited labeled data, and potentially identify avenues for improving its generalization capabilities.

The YOLOv8 is an extension of the previous YOLO models and offers improvements in terms of accuracy and speed. The network takes an input image and divides it into a grid of cells, each of which is responsible for detecting objects that fall within its boundaries [6]. By performing detection at the level of individual cells, the model can achieve real-time performance without sacrificing accuracy. Each cell in the grid predicts a fixed number of bounding boxes, along with a corresponding confidence score that reflects the likelihood of an object being present in that box.

Furthermore, the network is capable of predicting the class probability for each box, allowing it to identify the specific type of object contained within. These predicted bounding boxes with associated class probabilities are subsequently used to generate the final object detections. This approach has proven to be highly effective, as it allows YOLO to detect multiple objects in an image with high accuracy and efficiency. Figure 1 shows the outlook of the proposed YOLOv5 architecture, which is similar to YOLOv8.

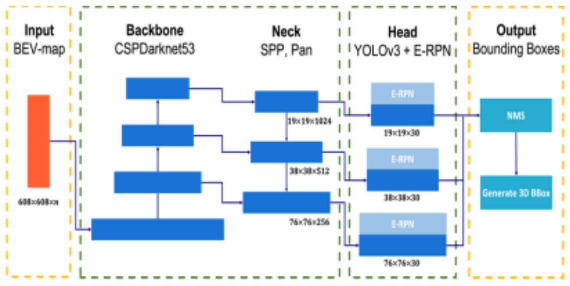

Fig. 1 - Proposed system with YOLOv5 architecture. [7]

2. Methodology

2.1 Generating Artificial Images

This research proposes a novel approach to data augmentation that involves generating synthetic samples to supplement the training dataset. This technique, outlined in [8], offers several advantages over traditional methods that rely on real-world data. One of the primary benefits is that it eliminates the need for costly and time-consuming data acquisition and annotation. Instead, synthetic samples can be generated quickly and easily through cropping and collage, allowing for more efficient model training.

To implement this approach, we follow the steps outlined in [8] and begin by acquiring templates of the traffic signs of interest. In this study, our focus is on the 14 classes that are present in the original Brazilian traffic sign dataset [4]. To obtain these templates, we leveraged the work of [9], who conducted a comprehensive study of CNNs on the detection and classification of Brazilian traffic signs. The templates from this study are available online and are shown in Figure 2. This provides a valuable resource for this research and allows us to proceed with generating synthetic samples to augment the training dataset.

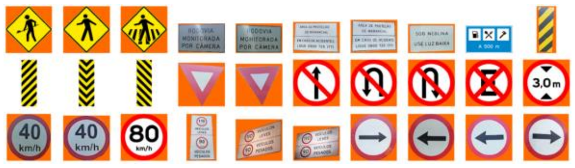

Fig. 2 - Example of templates of Brazilian traffic signs.

The second step requires selecting background images so we can make a collage with the sign's templates. While it may seem logical to choose images from the domain of interest, this isn't the chosen approach. This is because it may introduce unwanted noise in the data. In addition, there are freely available large-scale datasets such as ImageNet and Microsoft COCO with thousands of images that can be used instead as background. For this study, following the methodology outlined in [8], we chose to use Microsoft COCO images as background, except for images containing closely related classes to the domain of interest, to avoid introducing noise in the training data. Once the background images have been chosen, the training samples can be generated.

The last step of the generation of the synthetic images is blending the background images and the templates of the Brazilian traffic signs. This way is not necessary to manually annotate each image, since the position of the traffic sign being pasted is known. If successful, this process is capable of generating a large-scale database, and is possible to balance the number of samples of each class. [8] Figure 3 shows some artificially generated images.

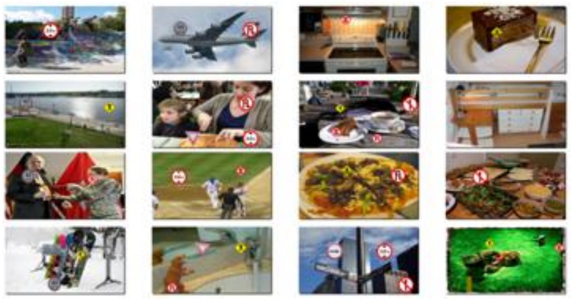

Fig. 3 - Example of generated training samples.

2.2 Model Training

This research utilizes the powerful Google Colab virtual environment with GPU support to train the proposed YOLOv8 model. The training data is divided into two groups: the control group comprising the original data, and the second group containing all images from the first group as well as the augmented training data generated in the previous section.

To adhere to the conventional approach of machine learning methods, we split the control group into three parts: train, validation, and test, with 70%, 20%, and 10% of images respectively. To prevent data leakage from the test split, we remove all test images from the augmented dataset, which contains the synthetically generated images, and split it into two parts: train and validation.

By doing so, we ensure a fair comparison of results between both groups, as the test sets are identical and data leakage is prevented. The hyperparameters used in the training of the first group are the same as the ones used in the second group.

To optimize the accuracy of the YOLOv8 algorithm, various training settings were tested. Transfer learning was implemented from YOLOv8m, and many of the default configurations for the model were utilized. Batch sizes 1, 8, 12, 16, and 20 were tested, as well as epoch numbers 50, 80, 120, and 150. Additionally, learning rates of 0.1, 0.01, and 0.001 were tried. The input sizes tested included 640x640 and 900x900. After testing, the most effective training setup was determined to be a batch size of 16, 120 epochs, a learning rate of 0.01, and an input size of 900x900.

2.3 Datasets

The first group of data comprises 275 manually annotated Brazilian traffic signs sourced from [4]. These images belong to 14 distinct classes, as seen before, and serve as the foundational component of the training dataset. In addition to this, we generated 248 artificial images, thereby bringing the total number of training images to 496 in the second group. It is worth noting that 27 of the original images were excluded and reserved for the test set, which was discussed earlier.

2.4 Performance Metrics

Usually, to evaluate the model performance, one commonly used metric for classification tasks is accuracy, but it can be misleading in cases where the classes are imbalanced. In object detection tasks, mAP (mean Average Precision) is mostly used, which measures the accuracy of object detection and object localization.

The metric is calculated by first defining a threshold for the intersection over union (IoU) between predicted and ground-truth bounding boxes, typically set to 0.5. IoU measures the overlap between two bounding boxes, and a threshold of 0.5 means that a predicted box is considered a match if its IoU with the ground-truth box is greater than or equal to 0.5. Figure 4 shows how IoU is calculated.

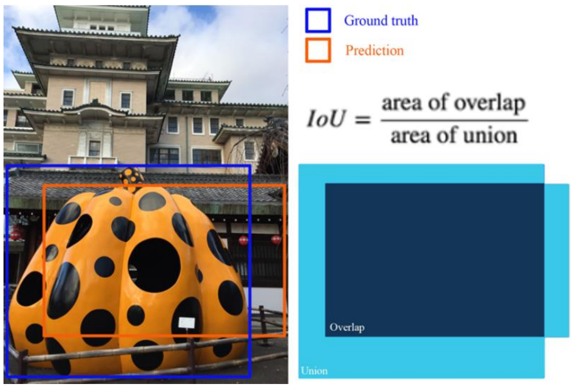

Fig. 4 – Definition of IoU. [10]

For each class, the precision-recall curve is calculated by varying the confidence threshold of the model and counting true positive (TP), false positive (FP), and false negative (FN) detections at each threshold. Precision is defined as the ratio of TP over the sum of TP and FP, while Recall is defined as the ratio of TP over the sum of TP and FN. The average precision (AP) for a given class is the area under the precision-recall curve. [10] Figure 5 shows the definitions of *precision* and *recall*.

$$Precision = \frac{TP}{TP + FP} \qquad Recall = \frac{TP}{TP + FN}$$

$$TP = \text{True positive}$$
$$TN = \text{True negative}$$
$$FP = \text{False positive}$$
$$FN = \text{False negative}$$

Fig. 5 – Definition of *precision* and *recall*. [10]

Figure 6 shows the formula to calculate mAP. This metric takes into account both precision and recall for each class and is the mean of AP values across all classes.

$$\mathrm{mAP} = \frac{1}{N}\sum_{i=1}^{N} \mathrm{AP}_i$$

Fig. 6 – mAP formula [9]

3. Results

In this section, we present the performance of the model's ability to detect Brazilian traffic signs and compare its performance on two sets of images. The first set had 248 images (the control group), while the second set had 496 images (the augmented group). As we expected, training the model in the augmented group took nearly twice as long as training it in the control group. The control set was trained in 0.857 hours over 120 epochs, while the augmented set was trained in 1.581 hours over the same number of epochs. Both models had the same architecture, with a count of floating-point operations (FLOPs) of 78.7 GFLOPs, and nearly 26 million parameters spread across 218 layers.

For the control group, we can see the evolution of loss function by Loss and by Classes for training and validation sets, in Figure 7 and Figure 8.

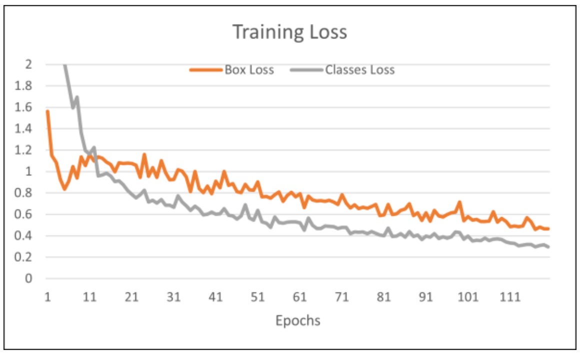

Fig. 7 – Training Loss on the original dataset

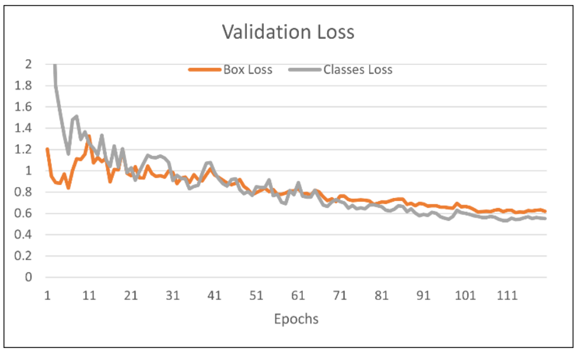

Fig. 8 – Validation Loss on the original dataset

The same can be seen, in Figure 9 and Figure 10, for the augmented group.

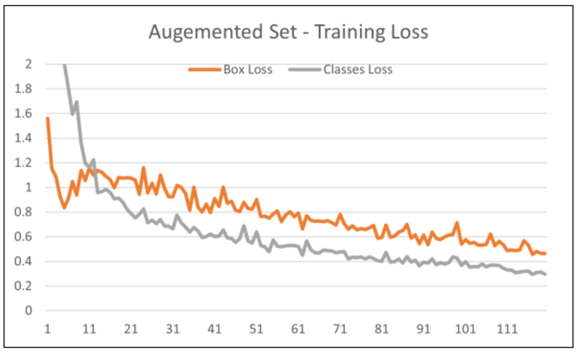

Fig. 9 – Training Loss on the augmented dataset

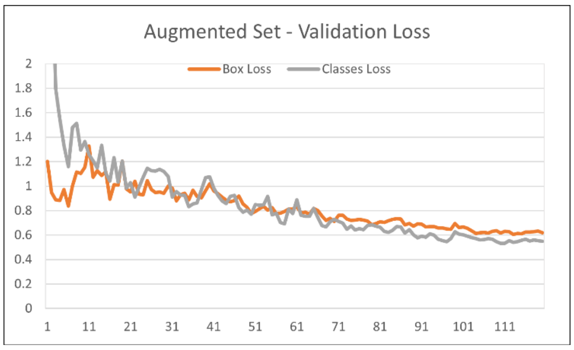

Fig. 10 – Validation Loss on the original dataset

Analyzing the graphs from the loss function is possible to see that the graphs are well-behaved, with a steady decrease in the loss over time. This is a positive sign that the model is learning as the training progresses. The loss function is a good metric to monitor during training, as it reflects how well the model is fitting the training data.

Figure 11 shows the evolution of $mAP^{IoU=0.5}$ while training on the original dataset.

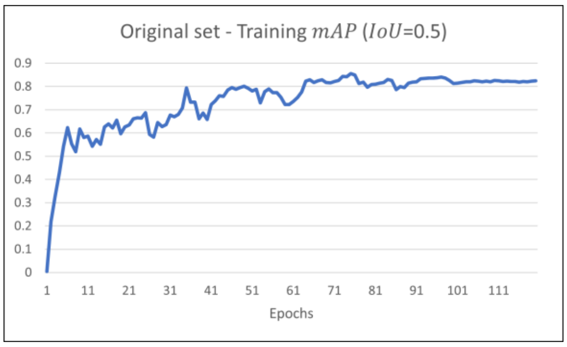

Fig. 11 – $mAP^{IoU=0.5}$ while training on the original dataset

And Figure 12 shows the evolution of $mAP^{IoU=0.5}$ while training on the augmented dataset.

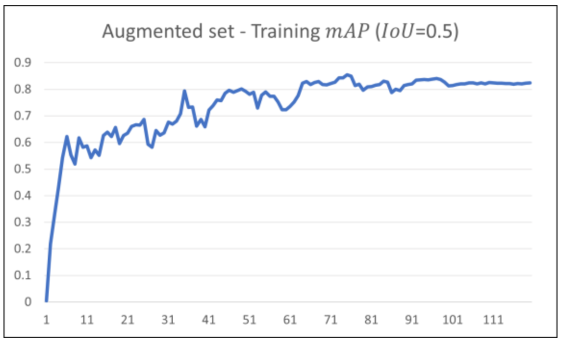

Fig. 11 – $mAP^{IoU=0.5}$ while training on the augmented dataset

Table 1 shows the results of model's performance when inferencing on unseen images from the testing set.

Tab. 1 - Performance on test set

Dataset	Precision	Recall	$mAP^{IoU=0.5}$
Original	0.884	0.807	0.901
Augmented	**0.934**	0.807	**0.923**

4. Conclusion

The main task of this study is to propose that is viable to create artificial images to augment a poor dataset. Such a problem is relevant to scientists because CNNs models often need a huge amount of data to become robust, and often is costly to label the images. So was built a system that can take templates and paste them into out-of-domain backgrounds. The original dataset has been doubled in size and the results shows an increase of 2,5% on $mAP^{IoU=0.5}$ of the model. This approach is particularly relevant for researchers and developers working with CNN models, as it provides a cost-effective solution for obtaining more data without the need for extensive labeling efforts.

Furthermore, the approach of using templates and out-of-domain backgrounds can be easily adapted for use with other object detection tasks, beyond just traffic sign recognition. This can help address the challenge of limited training data in other domains, including robotics, autonomous vehicles, and medical imaging.

In conclusion, this study highlights the potential for synthetic image generation as a viable approach for augmenting training datasets and improving the performance of object detection models. Future work in this area could explore the use of more advanced image generation techniques, such as generative adversarial networks, to further improve the quality and diversity of the generated images.

5. References

[1] Data Science Academy. *Deep Learning Book.* 2022; < www.deeplearningbook.com.br>

[2] O'Shea K., Nash R. An Introduction to Convolutional Networks. 2015; <https://arxiv.org/pdf/1511.08458.pdf>

[3] Bansal A., Sharma R., Kathuria M. A Systematic Review on Data Scarcity Problem in Deep Learning: Solution and Applications. *ACM Computing Surveys.* 2015; 54:1-29. <https://doi.org/10.1145/3502287>

[4] Barcia R. L. Automating the recognition of traffic signs on roads using artificial neural network. 2021; <https://doi.org/10.17771/PUCRio.acad.56926>

[5] Ultralytics. YOLOv8 Documentation. 2022; <https://docs.ultralytics.com/>

[6] Redmon J. Divvala S., Girshick R. You Only Look Once: Unified, Real-Time Object Detection. *CoRR.* 2016; <http://arxiv.org/abs/1506.02640.pdf>

[7] Mylvaganam P., Dissanayake M. B., Deep Learning for Arbitrary Shaped Water Pooling Region Detection on Aerial Images. *IEEE.* 2022; <https://ieeexplore.ieee.org/document/9906204>

[8] Torres L. T., Paixão T. M., Berriel R. F., Souza A. F., Badue C., Sebe N. Oliveira T. S. Effortless Deep Training for Traffic Sign Detection Using Templates and Arbitrary Natural Images. *CoRR.* 2019; <https://arxiv.org/pdf/1907.09679v1.pdf>

[9] Verçosa J. P. K. de Q. Redes Neurais Convolucionais na detecção e classificação de placas de trânsito brasileiras. 2022; <https://github.com/JPVercosa/Final-Thesis-Dataset>

[10] Hui J. mAP (mean Average Precision) for Object Detection. *Medium Article.* 2018; <https://jonathan-hui.medium.com/map-mean-average-precision-for-object-detection-45c121a31173>