Illegal graffiti detection with YOLOv4

Guilherme Andrey Medeiros Ribeiro
guilhermeandrey7@gmail.com

I. Introduction

Graffiti is a widely studied topic covering many scientific fields, from artistic techniques to political participation and vandalism. Nevertheless, illegal graffiti is a recurring problem in our society, and it is one of Brazilian's most common form of vandalism. The visual pollution and the detriment of public heritage require public interference. Thus, identifying and quantifying illegal graffiti in urban areas is very important to public managers to effectively identify and treat this problem.

The present research aims to automatically identify illegal graffiti using YOLOv4 [1], which is state-of-the-art in object detection [2].

II. State of the Art (SOTA)

Computer vision problems had a significant breakthrough with the advances of Deep Learning and Convolutional Neural Networks (CNN). In this regard, the computer vision field has a lot of subdivisions, whose some of the main ones are: classification, object detection, semantic segmentation, instance segmentation and panoptic segmentation.

The neural network architectures within each scenario have different configurations, and the appropriate usage may vary according to the objective. Object detection had many changes over the years, and it is very suitable for live videos since it enables fast inference and good precision. Thus, a common ground for evaluating new object detection models is the COCO dataset [3]. It consists of thousands of labeled images in a variety of scenarios, with 80 different classes of objects. Moreover, the current state-of-the-art relies on the best metrics method in the shortest period. Currently, an implementation of the YOLOv4 architecture, Scaled YOLOv4 [4], is one of the best models within the COCO dataset [2], with others such as EfficientDet [5] and DetectorRS [6].

You Only Look Once (YOLO) [7] was a breakthrough in real-time object detection since it was considerably faster than other algorithms of the time. Since the publication of the YOLO paper, the model got many improvements through further researches and experiments, resulting in the release of three improved versions: YOLO9000 [8], YOLOv3 [9] and YOLOv4.

The algorithm performs detection on the entire image at once, thus the name You Only Look Once. That is the main reason why YOLO is considerably faster than other approaches, since they perform detection by scanning through multiple regions of the image.

Looking forward to a better understanding of YOLO's functioning, a brief explanation of its architecture and the improvements the subsequent models presented will be provided below. For a more detailed explanation of the architectures, internal features and results, refer to [1, 7, 8, 9].

A. *YOLO*

Training begins by re-scaling images to $448x448$ size, dividing them into a SxS grid and performing detection on all grids simultaneously. Each grid cell is responsible for detecting a single object and outputs a set of B bounding boxes predictions and its confidence scores, alongside C conditional class probabilities. The combination of those labels form the prediction vector fed into the neural network.

The confidence score in this implementation is defined as the Intersection over Union (IOU) metric of the predicted bounding box. IOU measures how accurate a prediction is by dividing the area of intersection by the area of union between predicted and ground truth boxes. It is a key concept within object detection field used to measure the accuracy of a prediction.

Since the same object can be identified by more than one grid cell, predictions go through non-max suppression, which consists of eliminating all boxes with a confidence score below a given threshold, taking the from the remaining boxes the one with the highest confidence score and outputting it as a prediction, and finally calculating the IOU of the remainder boxes with the box outputted in the previous step. Then the process is repeated until all predictions are made.

A CNN similar to GoogleLeNet [10] is used for classification, adapting the last layers of the model for detection. The loss function used is the sum-squared error. The model uses pre-trained weights trained on $224x224$ images on the ImageNet 1000-class competition dataset [11]. More details on the network architecture and loss function are presented at [7].

B. *YOLO9000/YOLOv2*

In order to fix some issues with the first model, such as low recall and localization errors, the author changed four major features on the architecture of the model: (I) the previous CNN format got replaced by a new format named Darknet-19, similar to the VGG model [12]; (II) trained the last layers of the pre-trained network on $448x448$ images instead of $224x224$, besides randomly changing the input size throughout the batches, thus making it more robust to images of different sizes; (III) performed batch normalization, improving the convergence of the model [13]; and (IV) used anchor boxes, similar to those present on the Fast R-CNN [14] model. A K-means clustering is applied to boxes on the training set aiming to determine the number of anchor boxes and their shapes. More details on the improvements made can be obtained at [8].

C. *YOLOv3*

The author changed the internal CNN architecture from Darknet-19 to a new Darknet-53, increasing the number of layers from 19 to 53, making the model more robust. Besides that, they added three main new features: (a) each bounding box predicted now generated a objectness score using logistic regression; (b) they replaced the softmax function for class prediction by a simple logistic regression classification with binary cross-entropy loss, which in practice allows the model to make multi-label predictions; and (c) instead of predicting boxes exclusively at the final layer, it also predicts them at 3 intermediate stages of the network. For more details, refer to [9].

D. *YOLOv4*

The fourth version aimed towards making detection faster and more precise, thus many changes to the architecture and feature additions were made. The main improvements result from the use of many data augmentation techniques entitled Bag-of-freebies (BoF), the addition of the Bag-of-Specials (BoS), which is part of the post-processing phase that enhances accuracy, besides the inclusion of skip-connections in the network. The model uses CSPDarknet53 [15] network as its backbone and the same head architecture of YOLOv3. Detailed information on the new features can be found at [1].

III. Project contribution

The rapid detection of illegal graffiti applies to many governmental and private companies. The usage of object detection networks enables a fast and precise localization in many areas, guiding proper authorities to better decision making in this problem. Thus, this project's contributions are: (a) creating a database for identifying illegal graffiti, which can be augmented by other researchers; (b) applying SOTA YOLOv4 architecture for this purpose; and (c) comparing the results with and without pre-processing techniques.

IV. Methodology

The present research presented the following methodology: (A) image acquisition and labeling; (B) image preprocessing; (C) configuring and training YOLOv4; and (D) accuracy analysis. Details for each step are presented below.

A. *Image acquisition and labeling*

Image acquisition used publicly open images obtained from Google Images, and the selection was automatic with the posterior manual exclusion of low-quality and duplicated images. The total number of images used in this dataset was 101. The labeling procedure used LabelMe framework [16], where the annotations are

obtained by drawing bounding boxes on the objects of interest and designing a class (see example in Figure 1). The framework automatically outputs their coordinates, which are used in training and testing stages.

Fig. 1: LabelMe annotation example

B. *Image pre-processing*

Image pre-processing is a crucial step that enhances results in most Deep Learning algorithms. The most common approach is data augmentation, where filters, rotations, distortions and many other techniques are applied to increase training data and make it more robust to unseen data. This stages used Roboflow framework [17] to apply data augmentation to the images and export both images and annotation in the YOLO Darknet format, needed for the implementation steps. The resulting set has 243 images total.

After performing data augmentation on the original set of images, a second dataset was created by applying unsharp mask on all images. Unsharp mask is a technique used in image processing to enhance details, sharpness and contrast of images by applying a high-pass filter on the image and subtracting the result from the original image, thus obtaining the edges of the image, then adding them, multiplied by a factor $k \leq 1$, to the original image. Unsharp mask is ideal for this problem, since graffiti on walls consist mostly of edges, they are regions of the image with high-frequency. An example of the result of high-pass filtering on an image from the training set can be visualized in Figure 2.

Training and testing are performed separately on each set of images, and their results compared in order to evaluate the effect of simple image pre-processing on the quality of results.

Fig. 2: High-pass filtering result example

C. *Configuring YOLOv4*

The implementation used in this work was obtained from Roboflow's GitHub repository [18] using Joseph Redmon's (the creator of YOLO) open-source implementation of Darknet [19]. The configuration parameters are based on a jupyter notebook created by Roboflow [20], changing some parts to adapt to the graffiti images and in order to perform the evaluations this research aims for. Full implementation details and code can be found at [21].

This initial research maintains the original architecture, parameters and hyperparameters of the network provided during training and testing time. The model used the weights obtained from the original implementation of YOLOv4 as its initial weights.

Images on both sets were reshaped to the standard of size $416x416$ for training and testing purposes. The model initially trained for 1000 epochs with an input size of $416x416$ and batch size equal to 64. After this first training stage, the model's input size changed to $608x608$ aiming towards getting more precision on small objects detection, followed by an extra 1000 epochs of training using the weights obtained from the first stage. The process of changing parameters of the model and train it for more epochs is called fine-tuning.

D. *Accuracy analysis*

After a few hours of model training, the best weights obtained were used to make predictions and calculate accuracy on the test sets. there are two main metrics

used for evaluation of the quality of an object detector: mean average precision (mAP) and inference time.

mAP is the mean of the average precision (AP) between all classes of objects. In this research $mAP = AP$, since the detection problem only has one class. Average precision is defined as the area under curve (AUC) of the precision-recall function, that is:

$$AP = \int_0^1 p(r)\, dr \tag{1}$$

where the precision-recall curve is the curve obtained by changing the cutoff value for classification to values between 0 and 1. The metric is calculated through an approximation of equation 1, since the implementation uses discrete values and thus cannot be integrated.

Most metrics used to evaluate detection models use the AP metric for different configurations. The main ones are AP, AP50, AP75, APsmall, APmedium and APlarge. AP refers to the average of mAP scores for multiple IOU thresholds between certain given range, AP50 is the mAP for $IOU = 50\%$, AP75 for $IOU = 75\%$ while APsmall, APmedium and APlarge refer to the mAP for different sizes of objects (small, medium and large respectively). This research uses the AP50 metric in order to make its evaluations and conclusions.

V. Results

Table I shows the AP50 scores and inference time obtained for every model trained. There were four different training stages: (a),(b): the initial training for 1000 epochs with input size of $416x416$ on both sets of images; and (c),(d): the fine-tuning stage, training for an extra 1000 epochs with input size of $608x608$ on both sets of images.

TABLE I: Performance Analysis

Image set		AP50	Time(ms)
Initial training	Original	28.13	44
	Unsharp Mask	34.78	44
Fine-tuned	Original	32.37	79
	Unsharp Mask	39.04	79

From those results, it is clear that fine-tuning the model to train on larger images and for more epochs considerably increases its precision. The downside is that, since the input images are larger, it takes considerably more time to perform detection. Nonetheless, an inference time of 79 milliseconds is still acceptable.

Fig. 3: Example of a successful graffiti detection on the unsharp masked image after fine-tuning

Fig. 4: Example of a slightly inaccurate graffiti detection on the unsharp masked image after fine-tuning

Other important conclusion taken is that a simple pre-processing step such as applying unsharp mask on images can produce substantial improvements on a model, even for a simple model with few images to train on.

Figure 3 shows that the model can easily identify where graffiti is located and accurately draws the bounding box for small and medium objects. However, as seen on Figure 4, although it successfully locates the object, on larger objects it struggles to draw the bounding box covering the whole area.

The AP50 scores are lower than most object detec-

tion tasks published, however it is due to the fact that there was not enough data, time and resources to train the model for more epochs and tune the parameters and hyperparameters of the network, which should be able to improve accuracy on the model. Notwithstanding, it still accomplishes good qualitative results, as the one observed on Figure 3.

VI. Conclusion and future works

This project presents a reliable algorithm on identifying graffiti on walls with moderate precision and good inference time. It also demonstrates the impact that a simple image pre-processing technique has on the accuracy of a model, showing that pre-processing images before getting into a computer vision task is crucial in achieving reliable results.

For future works, adding images of walls and scenes with no graffiti on them (negative samples) could be considered in order to accomplish better and more reliable results. Exploring different image pre-processing techniques is also important to find better ways of preparing images for training.

Moreover, this research did not explore the possibilities of YOLOv4's network architecture and training parameters and hyperparameters. Changing the number of layers and bounding boxes shapes, tuning regularization and dropout and testing different backbone structures for the classification part are possibilities to be explored in the future aiming to reach better precision on the detector.

An interesting alternative to be explored is to change the model in order to detect more than one class of objects. For example, differentiating illegal from legal graffiti and possibly being able to identify famous graffiti artists by their signature styles. Possibilities are endless for this task.

VII. Acknowledgements

This report was kindly supported by my co-workers and friends Osmar Luiz Ferreira de Carvalho, who guided me throughout the object detection area of computer vision, helping me with my implementation and theoretical background, and Isabel Caroline Gomes Giannecchini, who helped me on grammar review.

References

[1] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection, 2020.

[2] Papers with code - coco test-dev benchmark (object detection) 2020. https://paperswithcode.com/sota/object-detection-on-coco, 2020.

[3] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[4] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. Scaled-yolov4: Scaling cross stage partial network. *arXiv preprint arXiv:2011.08036*, 2020.

[5] Mingxing Tan, Ruoming Pang, and Quoc V Le. Efficientdet: Scalable and efficient object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10781–10790, 2020.

[6] Siyuan Qiao, Liang-Chieh Chen, and Alan Yuille. Detectors: Detecting objects with recursive feature pyramid and switchable atrous convolution. *arXiv preprint arXiv:2006.02334*, 2020.

[7] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2016.

[8] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017.

[9] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[10] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer*

vision and pattern recognition, pages 1–9, 2015.

[11] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

[12] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[13] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[14] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[15] Chien-Yao Wang, Hong-Yuan Mark Liao, Yueh-Hua Wu, Ping-Yang Chen, Jun-Wei Hsieh, and I-Hau Yeh. Cspnet: A new backbone that can enhance learning capability of cnn. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 390–391, 2020.

[16] Labelbox: The leading training data platform for data labeling. https://labelbox.com/, 2020.

[17] Roboflow: Go from raw images to a trained computer vision model in minutes. https://roboflow.com/, 2020.

[18] roboflow ai. roboflow-ai/darknet. https://github.com/roboflow-ai/darknet, May 2020.

[19] Joseph Redmon. Darknet: Open source neural networks in c. http://pjreddie.com/darknet/, 2013–2016.

[20] YOLOv4 Darknet Object Detection Model. Yolov4 darknet. https://models.roboflow.com/object-detection/yolov4, 2020.

[21] Guilherme A.M. Ribeiro. Illegal graffiti detection with yolov4. https://colab.research.google.com/drive/1RoG9njaAtb6JEASSPcqFqKPdXja9tpx0?usp=sharing, 2020.