DroGate : Lightweight Real-Time Drone Perception System

Zeryab Moussaoui, **Yacine Benameur**, **Houssem Meghnoudj**, **Nabil Tchoulak**, **Merouane Guettache**

École Nationale Polytechnique - Alger
École Centrale d'Électronique - Paris

I. Introduction

Rapidity, Computational efficiency, Precision and Robustness have been our core objectives when designing for the Perception and Machine Vision test.

Though widely popular and very efficient, Region based object detector like Faster R-CNN [1] ,or single shot object detectors like YOLO V3 [2] and SSD [3] are not well suited for real-time onboard computation [4], neither do pre-trained deep Convolutional neural networks on large Dataset (e.g. VGG [10], ResNet [11]).

For the purpose of Autonomous Drone Racing in general, and in AlphaPilot challenge more specifically where computations must run fully onboard, we decided to adopt a shallow CNN architecture.

II. DroGate Architecture

DroGate is built on the efficient ResNet-based architecture of Loquercio et al. [5]

The input of the network is a grayscale 300x300 gate preprocessed image. The convolution stage of the architecture consists of a fast ResNet-8 with 3 residual blocks, followed by dropout and ReLU non-linearity.

The extracted features are then processed by two separates multilayer perceptrons (MLPs) (fig. 1):

- Linear layer that regresses the 4 (x,y) pixel coordinates of the gate in the image plan.
- Sigmoid layer to carry out the confidence score [13].

DroGate architecture has a proven efficiency as a drone perception system, because a similar architecture was used by [6] and [7] for autonomously navigating a drone and showcased a perfect balance between precision and rapidity of execution.

III. EDA & data cleaning

The data contains a lot of mislabeled gates and getting rid of these outliers is crucial as for every machine learning project.

We used several methods to detect these outliers:

- Trapezoidal heuristic : After observation, a strong hypothesis can be stated, which is that the projection of the gate on the image plan is in the major cases a trapeze. Images that deviate strongly from this rule are mainly outliers. Here are some examples of these outliers :

Fig. 2: Outliers detected using the trapeze heuristic

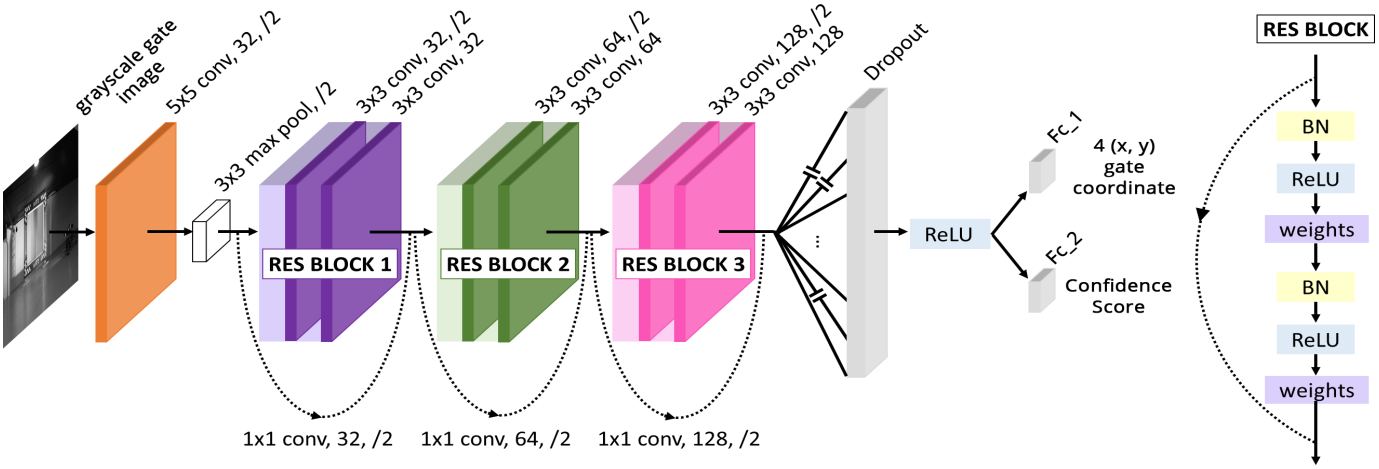

Fig. 1: DroGate Architecture [5]

- Autoencoders : Poorly reconstructed labels by the trained autoencoder are diagnosed as outliers [9]. many types of outliers are then detected (e.g. labels order inversion, partially occluded gates ...)

Fig. 3: Outliers detected using the Autoencoder

These methods allowed us to flag m any outliers, however they suffer from false positives detection, and other kinds of outliers remain non-detected. Therefore we decided to manually check for them. More details about data cleaning can be found in the **source code.**

IV. Training procedure

The network is trained in two stages:

1) First, the network is trained to output only the 4 (x,y) coordinates of the gate by minimizing a standard mean-squared-error (MSE). We then calculate the *COCO_map* [12] score of each image over groundtruth coordinates: these scores are the desired confidence scores that the network shall predict.
2) In the second stage, a sigmoid layer is added to the network and only the associated weights are trained. This layer aims to predict the calculated confidence scores. The loss used for training is a Mean Squared Logarithmic Error as experiments showed that the MSE loss suffers from convergence issues due to small gradients.

V. Results

The results showed that on average DroGate misses the groundtruth coordinate by 4.6 pixels.

This error is significant only when the drone is at a reasonable distance from the gate. Therefore, we estimate that this precision is sufficient for the purpose of drone navigation.

TABLE1. DroGate performance measurement

	Training Data	Leaderboard Data
Mean average precision (%)	81.2%	80.3%
Average MAE (pixels)	4.6	-

TABLE 2. Inference time comparison.

Model Hardware	DroGate i5-7440HQ	DroGate GTX 1050Ti	SSD [4] GTX 1050Ti
Inference time (ms)	9.01	**1.4**	24.16

VI. DroGate Improvements

Due to time limitations, we couldn't enhance the precision of Drogate predictions. However we believe this could be achieved through :

- Post-processing: based on a prioris about the gate projection on the image plan (trapeze hypothesis)
- Data augmentation: Rotate and flip the training images, change lighting conditions, add gaussian noise.
- Test deeper architecture depending on the limits defined by the onboard allowable computations.

During the final AlphaPilot challenge, the images rendered by the drone camera will contain multiple gates or no gates at all. Simple strategies are considered to handle these situations :

- Retrain Drogate on synthetic images (or real images if they are provided) to detect the closest gate.
- Knowing roughly the relative location of the next gate to the drone, simply ignore DroGate output if it strongly deviates from the expected output [6].
- Train DroGate on images with no gates to output "0" confidence score.

VII. Perspectives

DroGate will serve in the AlphaPilot challenge as the main resource for the perception system.

Depending on the retained approach, DroGate can be used either as a feature extractor to detect the official gates (fine-tuning), image-based visual servoing techniques can then be applied [4]; or repurposed to output intermediate informations like waypoints or carrying out directly heading angles and desired velocities in an end-to-end fashion [8] using primarily "Imitation Learning".

REFERENCES

[1] K. E. van de SandeT. GeversA. W. Smeulders. Faster R - CNN : Towards real - time object detection with region proposal networks 2015.

[2] Joseph Redmon, Ali Farhadi. YOLOv3: An Incremental Improvement 2018.

[3] Wei Liu et al. SSD: Single Shot MultiBox Detector 2015.

[4] Jung et al. Perception, Guidance and Navigation for Indoor Autonomous Drone Racing using Deep Learning 2017.

[5] Loquercio et al. Dronet: Learning to fly by driving 2017.

[6] Kaufmann et al. Beauty and the Beast: Optimal Methods Meet Learning for Drone Racing 2018.

[7] Kaufmann et al. Deep Drone Racing: Learning Agile Flight in Dynamic Environments 2017.

[8] Li , Muller et al. OIL: Observational Imitation Learning 2018.

[9] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press.

[10] Karen Simonyan, Andrew Zisserman : Very Deep Convolutional Networks for Large-Scale Image Recognition 2014.

[11] Xiangyu Zhang, Jian Sun et al. : Deep Residual Learning for Image Recognition. 2015.

[12] Tsung-Yi Lin, Michael Maire et al. Microsoft COCO: Common Objects in Context. 2014.

[13] Amit Mandelbaum, Daphna Weinshall : Distance-based Confidence Score for Neural Network Classifiers 2017