Gate detection algorithm for the AlphaPilot Challenge

Marina González Álvarez (5044030)
Individual assignment AE4317 - TU Delft (Netherlands)

ABSTRACT

This work deals with the design of a gate detection algorithm based on image segmentation in the context of the AlphaPilot Challenge. Three different alternatives based on the complexity of the model are designed and extensively analyzed to study the trade off between accuracy and computational effort and finally determine the most efficient solution.

1 INTRODUCTION

Deep neural networks have outperformed the state of the art techniques in many visual recognition tasks in recent years. Rapid and remarkable advancements have been achieved within the field lately, with the development of high-performance networks such as the AlexNet, GoogleNet, ResNet, YOLO and so on. However, the tight weight and hardware constraints present in the context of autonomous drone racing are incompatible with the great model size and computational effort inherent to these approaches, so that a simpler strategy must be selected.

In this work, a gate detection algorithm based on the U-Net [1], a considerably novel CNN originally used for biomedical image segmentation purposes, is proposed. This approach offers significant high-precision results with very few training images and a much faster performance than the previous alternatives, making it a very suitable strategy for drone racing. Section 2 delineates the learning process and architecture of the network. Then, the qualitative and quantitative results obtained are analyzed in Section 3, together with the computational effort spent by the algorithm.

2 ALGORITHM OVERVIEW

The architecture of the U-Net is illustrated in Figure 1. It obeys a contracted path with 3×3 convolutions (blue rectangles) and a 2×2 max pooling operation (red arrows) followed by an expansive path with an upsampling of the feature map and a 2×2 up-convolution (black arrows). The loss of border pixels in every convolution is fixed by means of a copying and cropping mechanism (grey arrows).

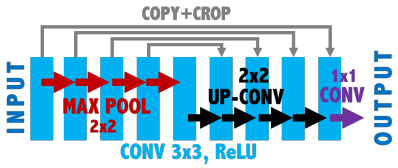


Figure 1: Network architecture.

A sigmoid activation function is used in the last layer, so that the output image has a *grayscale* format, with each pixel taking a value between $0 \leq p \leq 1$ (0: black and 1: white).

Regarding the learning process of the CNN, three different cases have been considered based on the size of the input image after the corresponding preprocessing, as it can be seen in Table 1. Therefore, the dimensions of each layer in Figure 1 and, consequently, the number of parameters of the model have been specifically adapted for each case. This allows to study the trade off between image resolution, computational speed and global performance. The optimizer and loss function used for training are *Adam* [2] and binary cross-entropy, respectively. Table 1 also shows the utilized hyperparameters for each case, which have been selected so that the training ends when the optimal capacity of the model is reached: no underfitting or overfitting, but the network fits *just right*. An example of this is illustrated in Figure 2, where the loss history obtained for the 64×64 input image case can be observed. Finally, the reason for such a small value of the batch size is due to hardware constraints from the available computer used for training.

Size	Epoch	Batch	α	β_1	β_2
256×256	15				
128×128	25	2	10^{-4}	0.9	0.99
64×64	50				

Table 1: Cases and learning hyperparameters.

The proportion of samples split among the training, validation and test set is presented in Table 2. This allocation has been performed randomly.

Set	Training	Validation	Test
Images	222 (72%)	40 (13%)	46 (15%)

Table 2: Data set split

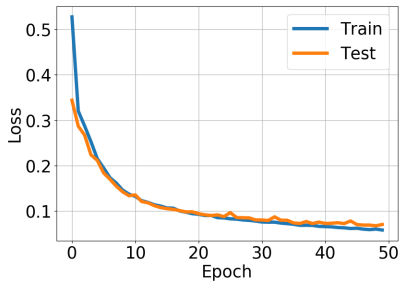


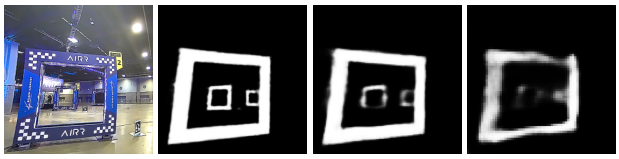
Figure 2: History loss per epoch (64×64 case).

3 GATE DETECTION

This section comprises the results achieved by the algorithm from a qualitative, quantitative and computational effort perspective.

3.1 Qualitative image segmentation analysis

Figure 3 illustrates an example of the predictions of the model for each of the considered input image size cases. Concretely, the results obtained for `img_185.png` are shown.



(a) Original (b) 256x256 (c) 128x128 (d) 64x64

Figure 3: Predictions on the test set (`img_185.png`).

It can be observed that the accuracy of the predictions is highest for the 256×256 instance, which can almost be confused with the actual ground truth image. Then, although the 128×128 case is slightly less accurate, the three gates present are still clearly detected and differentiated from the rest of the image. Finally, the 64×64 can still accurately predict gates that are at a close distance from the observer, but shows less precise results for those that are far away.

3.2 Receiver operating characteristic (ROC) curves

It is important to recall at this point that the prediction images shown in Figure 3 are in grayscale format. In order to quantitatively evaluate the accuracy of these predictions by comparing them to the actual masks, they must be converted to a binary format by means of a threshold.

(a) $t = 0.0$ (b) $t = 0.4$ (c) $t = 0.9$

Figure 4: Predictions after threshold, t (128×128 case).

Figure 4 shows the results obtained for the 128×128 case for three different values of the threshold conversion, t. Concretely, the effect of applying a threshold on the grayscale predictions is the following: all pixels having a value $p > t$ are converted to $p = 1$ and, the rest, to $p = 0$. With this information in mind, the ROC curves that compare the true and false positive rate of pixels with respect to the ground truths for all 46 images in the test set can be generated for the three cases under consideration, as shown in Figure 5. To generate these curves and perform the comparison, all predictions and masks have been resized to 315×315 (the smallest dimension found on the original ground truths data set).

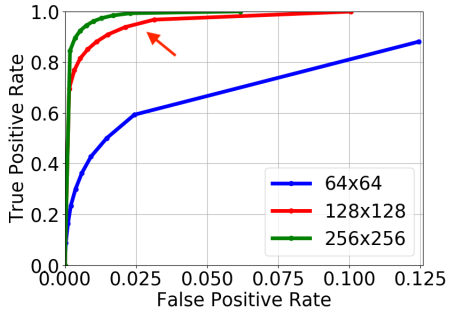


Figure 5: ROC curve for each considered case.

The points with the highest value of the FPR correspond to $t = 0$ and those with the lowest TPR to $t = 1$. These curves confirm that halving the image resolution from 256×256 to 128×128 does not make a big difference in the accuracy of the results, which is remarkably high for both cases for $0.1 < t < 0.5$, with a ratio of $\sim 95\%$ TPR vs. $\sim 2\%$ FPR. The precision of the predictions does diminish significantly for the 64×64 instance, although this case could still be considered acceptable for a threshold of $t = 0$.

3.3 Computational effort

A computer Intel Core i5-8265U, QuadCore, 1.6-3.9GHz, 8GB RAM, 512GB SSD and Intel UHD Graphics 620 was used to train the model and compute the predictions. The computational speed achieved by the final algorithm is illustrated in Table 3. It can be seen that the 256×256 is too slow but the other two cases are notably fast. Taking into account that the *AIRR racing drone* hardware is considerably more powerful than that available for this project, it would be able to easily run the 128×128 model instance, obtaining notably precise results at a fast rate. The 64×64 could also even be considered in the context of MAV racing.

Size	64×64	128×128	256×256
fps	92	44	4

Table 3: Frames per second depending on image size.

4 CONCLUSION

In this work, a computationally efficient gate detection method based on an image segmentation CNN is implemented in the context of autonomous drone racing. Three different models depending on the size of the input image are evaluated with the 128×128 case leading to the best results.

REFERENCES

[1] Philipp Fischer Olaf Ronneberger and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *Medical Image Computing and Computer-Assisted Intervention (MICCAI), Springer*, 9351:234 – 241, 2015.

[2] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2015.