Risiko! Army Detector: Synthetic2Real

Bresolin Paolo, Fonsato Andrea

Abstract

Risiko! is a widespread strategy game which involves two main types of pieces: flags and tanks. In the original version of the game, there are six colors for tanks and six colors for flags: yellow, red, blue, green, black and purple. This paper explains the development of an object detection system for the game Risiko!, aimed at the detection of twelve possible objects: six tanks and six flags, based on their colors. The implemented deep learning model is based on YOLO, the version 5 in particular. The pre-trained *YOLOv5* model is fine-tuned on two datasets: a dataset containing only synthetic images and another containing both synthetic and real images, but with a high predominance of the of the first ones with respect to the latter. As we will see, performances are very good for synthetic test sets, but poor for real images.

Introduction

Object detection plays a crucial role in various applications, especially in the computer vision domain: in the context of the Risiko! game, it can enhance the gameplay experience by automating the identification of game pieces. Our goal was to implement a method capable of detecting and localizing game pieces using a YOLO style model. The primary challenge in developing this system lied in the limited availability of annotated training data. To overcome this problem, we generated synthetic images.

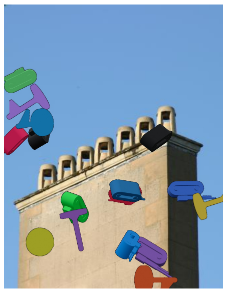
Figure 1: Example of synthetic image.

Then we focused on what solution to adopt to develop an efficient object detection system and, after several evaluations, we decided to fine-tune a YOLO model called *Ultralytics YOLOv5*.
To conclude, we evaluated the performance of our model on both a synthetic dataset and a provided real images dataset to understand the impact of incorporating real images in the training process. To facilitate the comparison and have some benchmark, we used a pre-trained *YOLOv7* model.

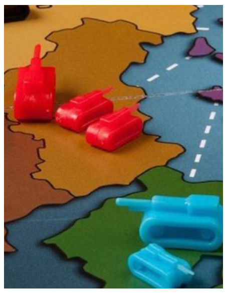
Figure 2: Example or real image.

We can say that the real objective of this work was understanding how performances change if the balance between synthetic images and real images used at training time changes. So, we were able to understand the impact of synthetically generated samples in a real world case in which the availability of real labelled patterns is limited and its production too expensive in terms of time.
In the following sections, we detail the methodology employed for synthetic dataset generation, the architecture design for object detection and the experimental setup for training and evaluation. We present the results of our model's performance on the synthetic dataset and on the provided real images dataset.

Method overview

In this section we describe every notebook we created, with the associated explanation of what it

does and the specific of some parameters and decisions taken.
The synthetic dataset generation process was carried out using the notebook *Risiko!_Synthetic_Dataset_Creator.ipynb*. It facilitated the generation of synthetic images, with their associated labels. After having set up some parameters (especially regarding colors), by running this script, a new set of synthetic images representative of the Risiko! game flags and tanks is created, putting them on some random background images among a provided set. As a first attempt, we generated 1000 synthetic images. However, after conducting some training experiments on our network, we realized that the results obtained were not optimal. This especially due to the very large imbalance between synthetic images (~1100) and real ones (~40). Therefore, we decided to create less images, precisely 150, which corresponds to approximately three times the number of provided real images. However, since we use also other ~120 synthetic images provided in the *synthetic_images* dataset, we still are using a total of ~270 synthetic images and ~40 real ones.
To ensure proper data organization and facilitate subsequent training and evaluation steps, the *Split_train_test_val.ipynb* notebook was used. It splits the images from three different sources, namely *real_images*, *synthetic_images*, and *synthetic_dataset* folders, into distinct sets for training, validation and testing. As a result, a comprehensive folder named *datasets* is created: it contains the divided images and their corresponding labels. The division between synthetic and real images is still preserved, because we need to train two times the model: before only on synthetic images, then also on real ones.
Specifically, we decided to divide the images into the respective subfolders as follows:

synthetic_dataset is split into
- *train* 70%
- *validation* 15%
- *test* 15%

synthetic_images is split into
- *train* 70%
- *validation* 15%
- *test* 15%

real_images is split into
- *train* 70%
- *test* 30%

For the purpose of training our object detection model, *Tanks_flags_detection_training.ipynb* notebook is used. It is focused on fine-tuning the *Ultralytics YOLOv5* model performing two different steps.
In the first phase, it uses solely synthetic images to fine-tune its weights and parameters. By exclusively using synthetic data during this training phase, the model could learn to detect and localize tanks and flags within a wider range of real-world scenarios.
The second phase of training incorporates both synthetic and real images. By combining these two sources of data, the model's performance and generalization capabilities were expected to improve, since it would have exposure to the real Risiko! game scenario too.
The results of both training experiments were saved in the dedicated *training_results* folder, enabling analysis and comparison.
Originally, we performed a training of 300 epochs but, since we wanted to try to obtain better results, we decided to run for 500 epochs for both trainings, knowing that in this way the risk of overfitting would increase.
To evaluate the performances of trained models, *Test_detection.ipynb* is used. Several evaluations have been tried, but in the end we decided to compute IoU, precision, recall and F1 score for each class and also averaged over all classes.
As a first thing, we evaluated the model trained only on synthetic images on three test sets: only synthetic images, only real images, both of them. Secondly, the second model (trained on both synthetic and real images) was evaluated on the same test sets.
To provide a benchmark for comparison, a *YOLOv7* model trained for the same task was evaluated using the *Risiko! Test.ipynb* notebook. This aimed to assess the performance of the *YOLOv7* network specifically trained for the Risiko! object detection task. By comparing the results of the two models implemented earlier with the performance of the this one, an analysis of the effectiveness and potential improvements of the developed models could be lead.

Results

It is important to note that within the *training_results* folder, there are two subfolders corresponding to the trainings and tests conducted on our model. One subfolder is about the training performed using only synthetic images, while the other subfolder contains data of training runs on

both real and synthetic images. Additionally, there is another subfolder containing the results produced by the *YOLOv7* model, called *runs*.
In these subfolders, various curves related to metrics and performance such as precision, recall, and F1 score can be found. Furthermore, several graphs illustrate the progress and performance of the models on the validation set over training iterations.
Regarding the results obtained by the fine-tuning of YOLOv5, we want to highlight that we achieved excellent results during training, with a precision of approximately 96% and a recall of around 87% on the validation set. This is true for both the training experiments: the one with only synthetic examples and the one with both synthetic and real images. When evaluating the models on the test sets, as expected, it is possible to see that still very good performances are achieved if the test set contains only synthetic images or a mix with real ones, but the results obtained when testing our models using only real images are not that good, with the model trained on also real images performing better. It's important to consider this is not due to a poor implementation or incorrect choices we made, but rather caused by the great imbalance between the number of synthetic and real images, with the availability of the latter significantly smaller compared to the quantity of synthetic images used to train the model. We also tried to train the models on less synthetic images, but performances were even worse, due to the small number of training examples and the consequent high performances on training set, but very low on both validation and test sets.
To conclude, given a model that successfully detects tanks and flags in the Risiko! game, it is possible to include it into a wider system of Risiko! gameplaying. This can be achieved, for example, by using a deep reinforcement learning strategy, that uses the object detection model to build an internal representation of the environment that can be used as state for the RL algorithm.