PlayCDC - Playing Card Detection

Learning to detect suits and ranks of playing cards

Daniel Gonzalez & Frank Gabel
Ruprecht-Karls-Universität Heidelberg

Contact Information:
Daniel Gonzalez
Email: d.gonzalez@stud.uni-heidelberg.de

Frank Gabel
Email: kpazrael2@gmail.com

Abstract

With the capabilities of upcoming small video capturing devices in, for example, smart contact lenses with built-in camerass, whole new ways of cheating in certain cardgames emerge. In order to help facilitate these cheating endeavours, we implement an algorithm that detects the suits and ranks of playing cards in the field of view of a camera using the latest iteration of the YOLO object recognition algorithm.

Main Objectives

The objectives of this project are summarized as follows:

Create a general dataset of a standard, 52-card deck of playing cards in different poses, brightness situations and blurring levels annotated with bounding boxes around the ranks and suits and corresponding class information.

Train an object detection algorithm on these synthesized data that performs bounding box localization and regression for classification. In particular, we train the latest iteration of the YOLO object detection algorithm [RF18] end-to-end.

Evaluate the algorithm on a hold-out validation dataset covering all classes. As a performance metric, mean Average Precision (mAP) is used.

Deploy the model on a smartphone camera as a proof of concept.

Methods

Dataset creation pipeline

The dataset was created in mainly 2 big steps.

Prepare raw card images We took two photos for each card of a deck of 52 cards, manually crop the cards to a selection and rescale the result by 900×600 pixels using GIMP. After this, we detect the convex hulls of the cards suits and ranks using the Python libraries SciPy and OpenCV.

Synthesize a general dataset First, we pasted the images in canvances provided by DTD [CMK+14] in order to have different textures around the images. In order to generate a considerable amount of training data, we apply different transformations to each image (blurring, sharpening, change of lighting, rotation, small shears). As convex hull information changes during operations such as rotation, we used the Python library imgaug to keep track of the correct coordinates at all times.
The final datasets consisted of up to 25000 images.

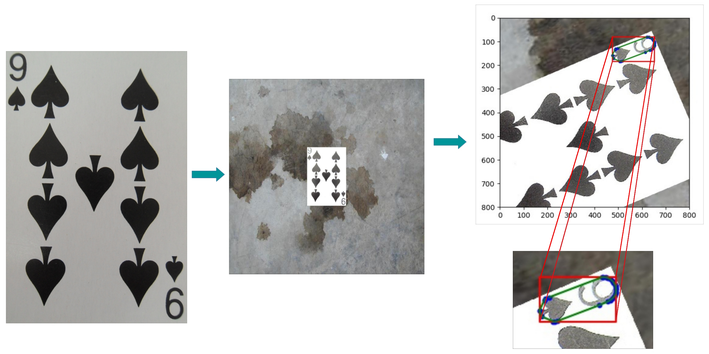


Figure 1: The dataset creation pipeline: paste photographs of images onto textures and apply random transformations to both images and corresponding convex hulls/bounding boxes

The YOLO approach to object detection

The YOLO approach works by modelling detection as a **regression problem**, where the image is divided into an $S \times S$ grid and for each grid cell B **bounding boxes** are predicted along with the **confidence** and C **class probabilities** for these boxes.
These predictions are performed simultaneously by a single CNN.

tiny YOLO-v3

The particular architecture we used for training on our dataset is **tiny YOLOv3** which essentially is a smaller version of YOLO for constrained environments.

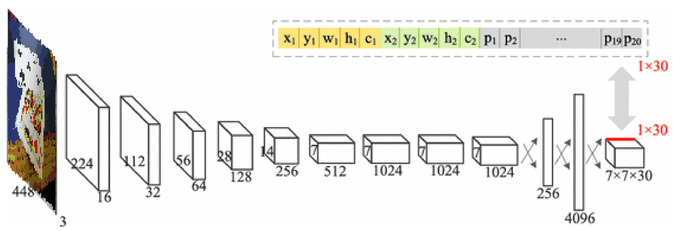


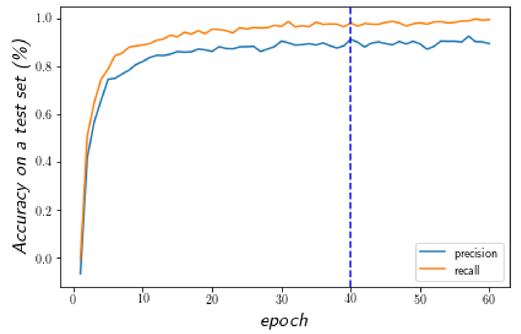
Figure 2: tiny YOLO-v3 feature extractor

Evaluation strategy

Evaluation of object detection algorithms is not easy as two tasks need to be evaluated: localization and classification. Therefore, often, mean Average Precision (mAP) is used in literature where mean classification performance (as measured by the precision) is averaged across predictions using different IOU thresholds.

Results

We consecutively trained on increasingly general datasets, starting with cards pasted on a white canvas to cards pasted on varying textures as backgrounds. For the most general dataset, we reached a mAP of **97.30%**. Using a dataset consisting of "easier" transformations (in particular, less blurring) resulted in a mAP of **99.11%**, but did considerably worse on random images of cards not from the same deck.



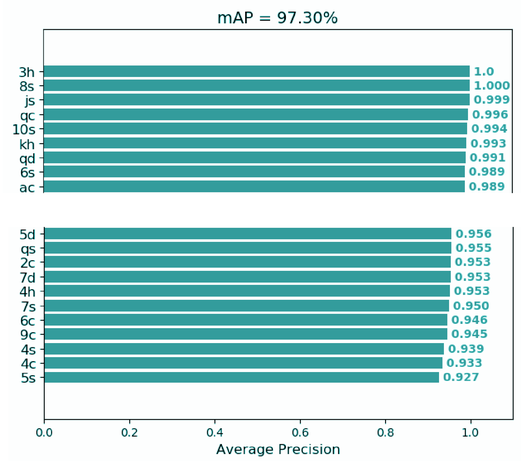


Figure 3: Training performance on holdout datasets across epochs (above) and performance of the final, trained model across different classes (below)

Conclusions

- Using an artificially created dataset, we achieve a mAP score of 97.30% on a holdout dataset.
- The task of object detection on ranks/suits of playing cards appears to be easy as it can be thought of as 2D rather than 3D.

References

[CMK+14] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, , and A. Vedaldi. Describing textures in the wild. In *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[RF18] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *CoRR*, abs/1804.02767, 2018.