Object Detection within a Robotic Application

Chia-Wen Tsai[1], Felizitas Kunz[2], Christoph Caprano[1], and Oskar Haller[1]

[1]Technical University of Munich [2]Ludwig-Maximilians-University, Munich

Introduction

Our idea was to teach a Braccio Robotic Arm to play the child's game pairs. It should detect the playing cards laying on the table, their position and the motif of a card and find the second one. The robot picks one playing card up, places it on the stack and searches for the second one within the remaining cards. For object detection YOLOv2-Real-Time-Object detection is used for transfer learning with datasets containing images of our playing cards. We trained our network with Google Cloud Service.

Dataset

Datasets for the project could be divided into two parts. Darknet has pre-trained on the Pascal VOC 2007+2012. Our dataset consist of 30 images for each of the 10 classes of playing cards. The labels have a format of: [category number] [object center in X] [object center in Y] [object width in X] [object width in Y].

We devided our dataset in training data and validation data - 90 and 10 percent respectively. Figure 1 illustrates sample images from our dataset. The 10 different classes are named Boat, Girl, Boy, Horse, EuropeanBird, PacificBird, Flower, BlueCard, Tree and Pineapple.

Fig. 1: Sample images from the dataset

Related Work

YOLOv2 is based on the Darknet-19 classification architecture, has 19 convolutional layers and 5 maxpooling layers.

YOLOv2 trains the labels and the bounding boxes at the same time with a joint dataset consisting of one for classification and one for detection. If an image from the detection dataset is detected, it backpropagates normally. If the other case is detected, the network only backpropagates the classification specific parts.

What makes YOLOv2 unique is that it reshapes the input images during training within a range of 320x320 to 640-640. Therefore the detection works for high resolution and low resolution camera streams. A higher resolution yields a more accurate result with longer processing time, while a lower resolution stream behaves the other way arround. Figure 2 illustrates that relation.

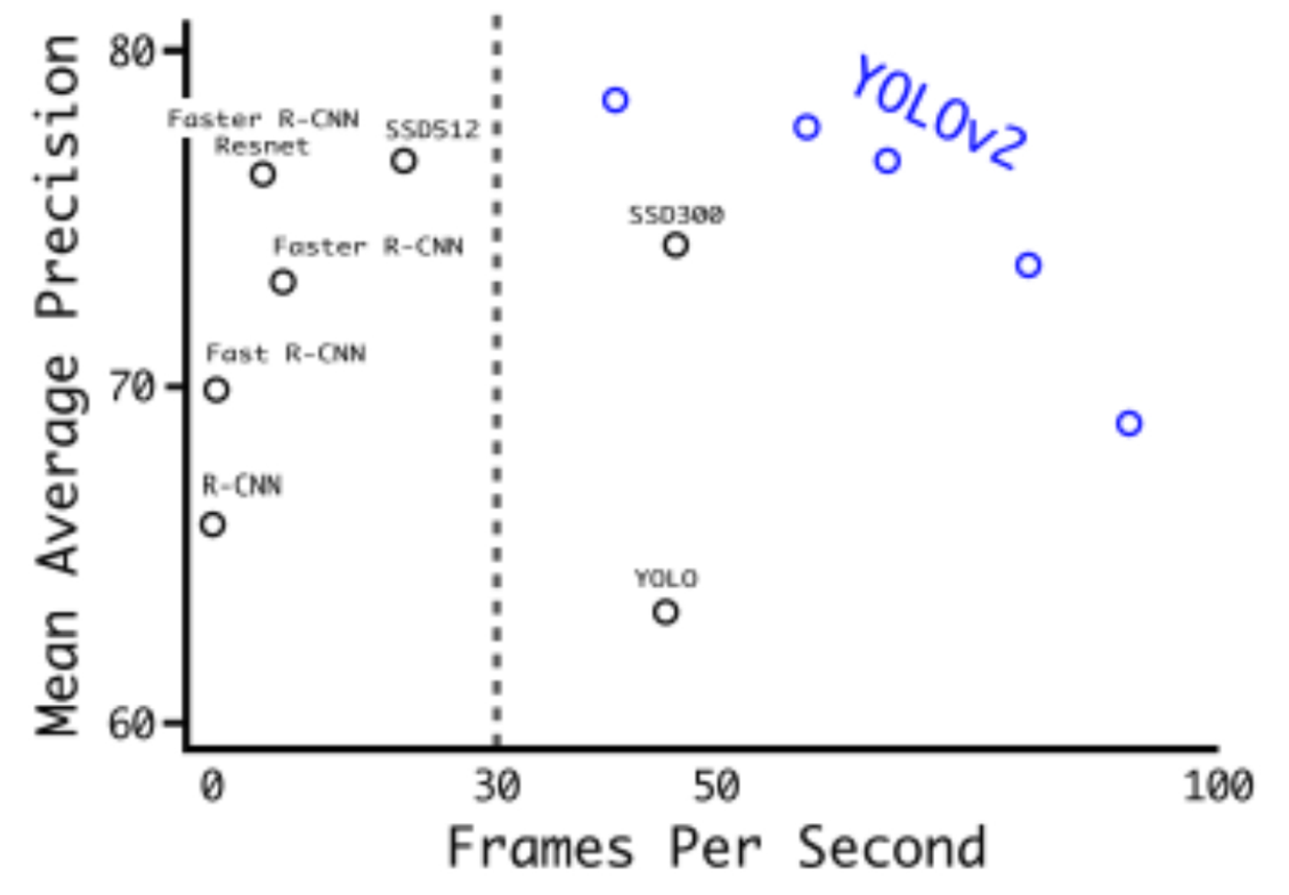

Fig. 2: Accuracy and speed of YOLOv2 on VOC 2007

Methodology

Network Architecture

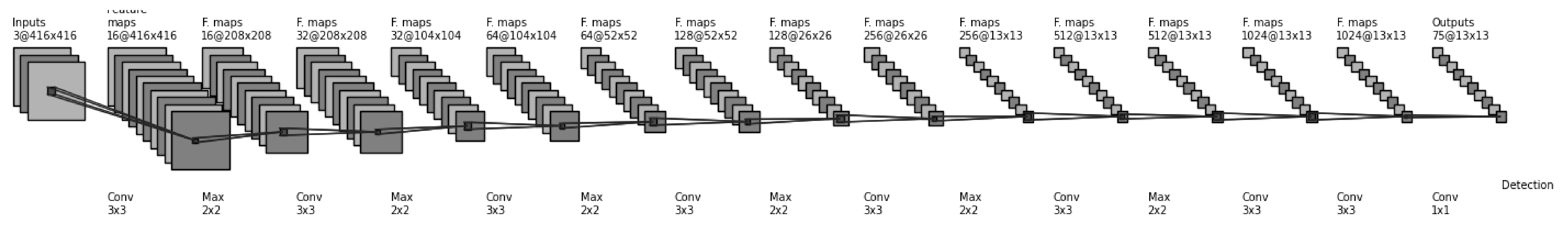

Fig. 3: Network architecture.

We used a pre-trained Tiny YOLO network (see figure 3). It has 15 layers (Yolo has 30), which makes it much faster, but also less accurate. However, our dataset only comprises of 10 instead of 1000 classes. Thus, using 15 instead of 30 layers seems reasonable. The architecture consist of 9 convolutional and 6 pooling layers. Leaky ReLUs are used for activation.

For transfer learning we only trained the last two convolutional layers for 5 epochs, therefore making use of the basic and mid level filters from tiny yolo and only changing the output mapping.

For optimization we used stochastic gradient descent with a starting learning rate of 0.1, polynomial rate decay with a power of 4, weight decay of 0.0005 and momentum of 0.9.

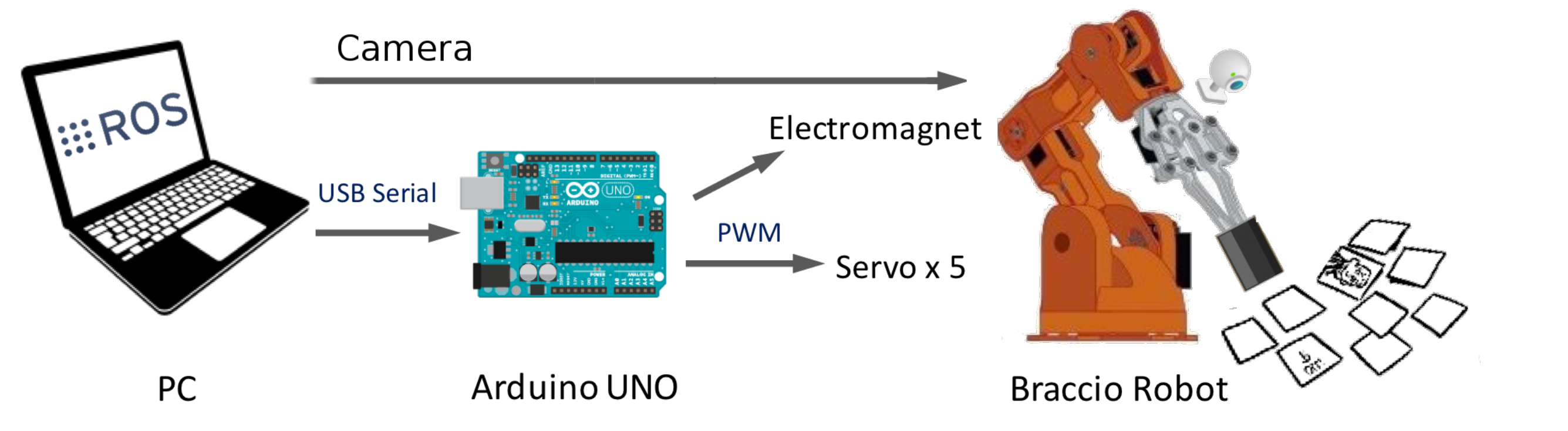

Fig. 4: Hardware setup

Hardware Architecture

Our hardware architecture consists of a PC, an Arduino UNO, an electromagnet, a camera and the Braccio Robot Arm. The communication between the devices and the search algorithm for the playing cards are implemented in ROS.

Outcome

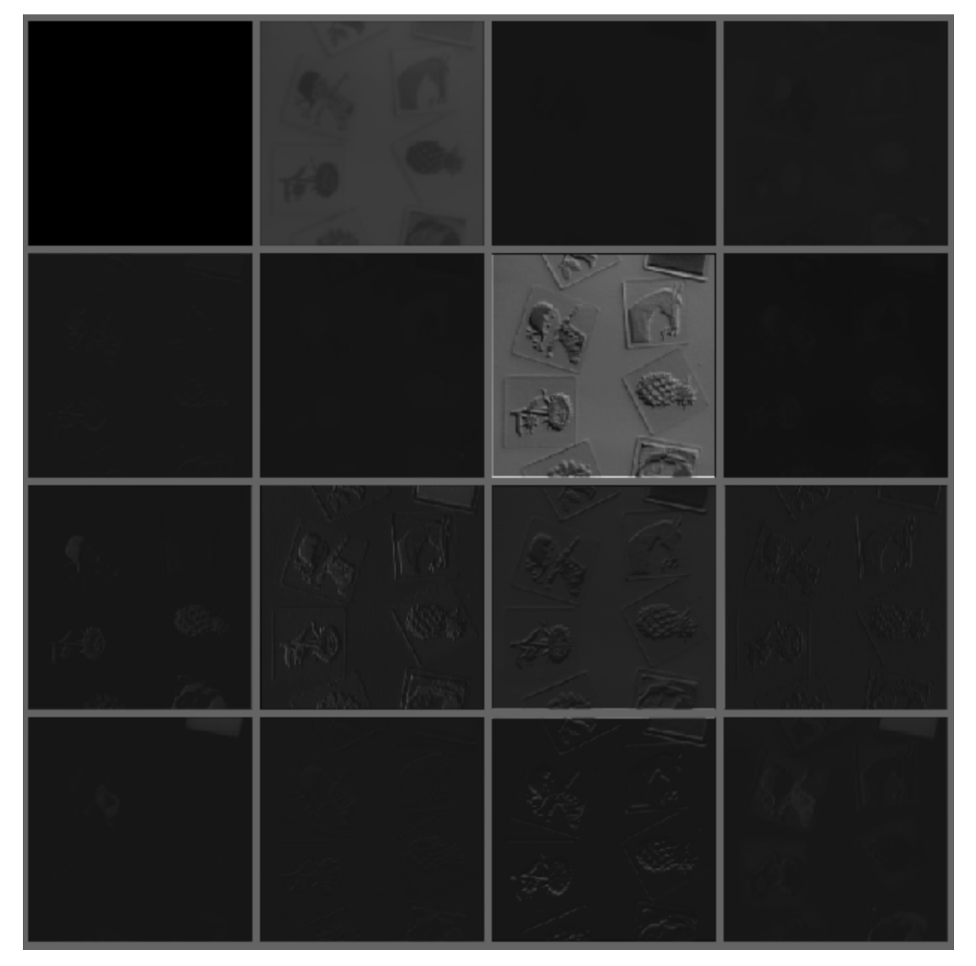

Fig. 5: Activation maps of first CONV layer

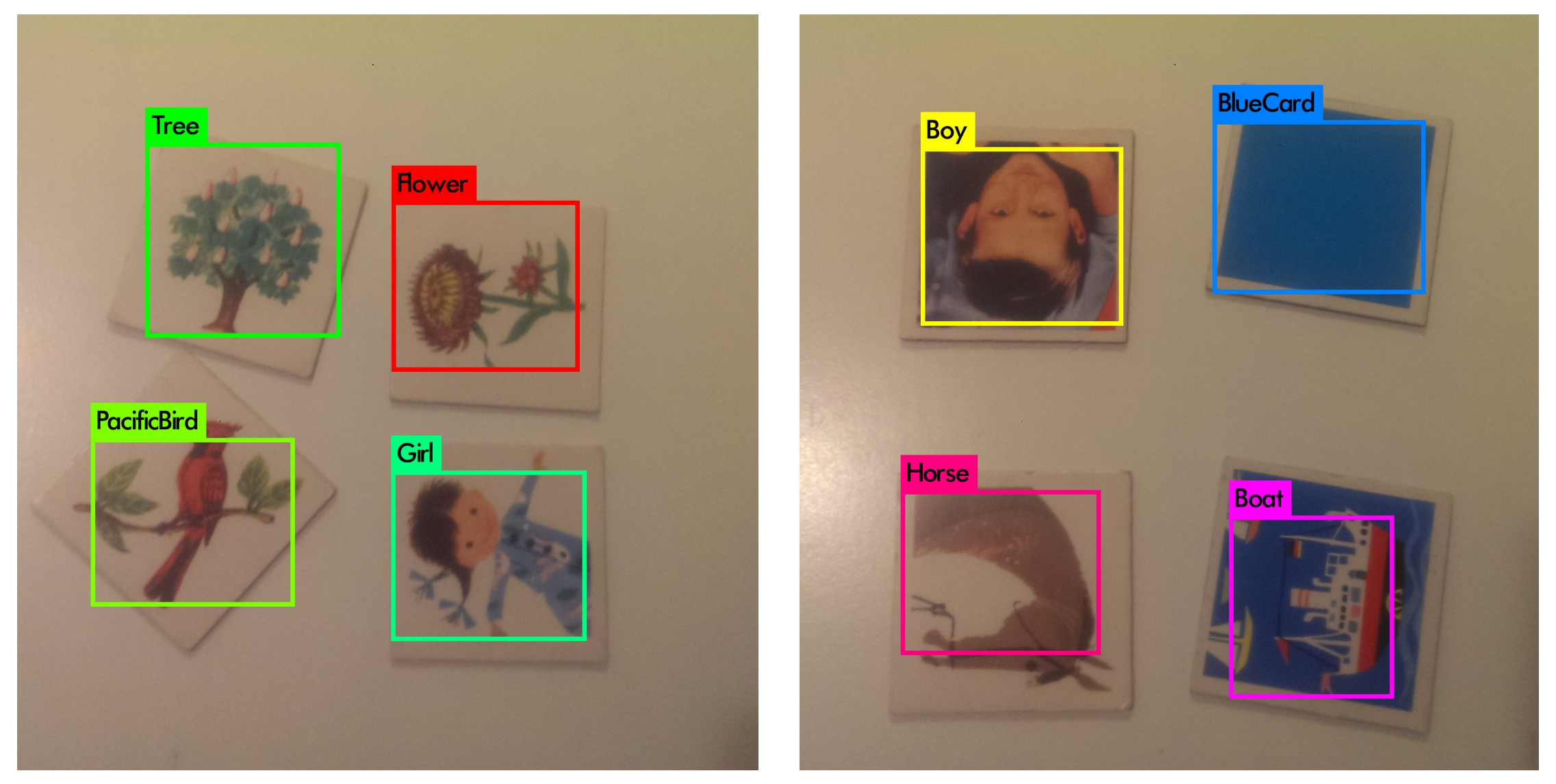

Fig. 6: Sample output

On a GPU a camera stream with input dimensions of 480 x 480 and 30 fps, can be processed in real time.

The robot starts grabbing the card as soon the card arrives within the center of its perception. If no card is the detected it searches for cards within a range of 0 - 150°. Figure 5 illustrates the activations of the 16 filters of the first convolutional layer. Figure 6 shows a sample output for two unseen images with 4 playing cards per image. The confidence of the detector can be adjusted with a threshold.