Breakout YOLO

Vladimir Philipenko

Optimization Class Project. MIPT

Introduction

One of the state-of-the-art models in object detection is YOLO. In 2018 YOLOv3 [1] was proposed showing better mAP score and lower inference time in comparison with other approaches. Taking into account that even this model is still slow on CPU I took its simplified versions — tiny-YOLOv3 and XNOR tiny-YOLOv3. I trained them on custom data to detect 4 gestures which are used to control classic browser Breakout game via webcam. Whole structure of this work can be described by following diagram:

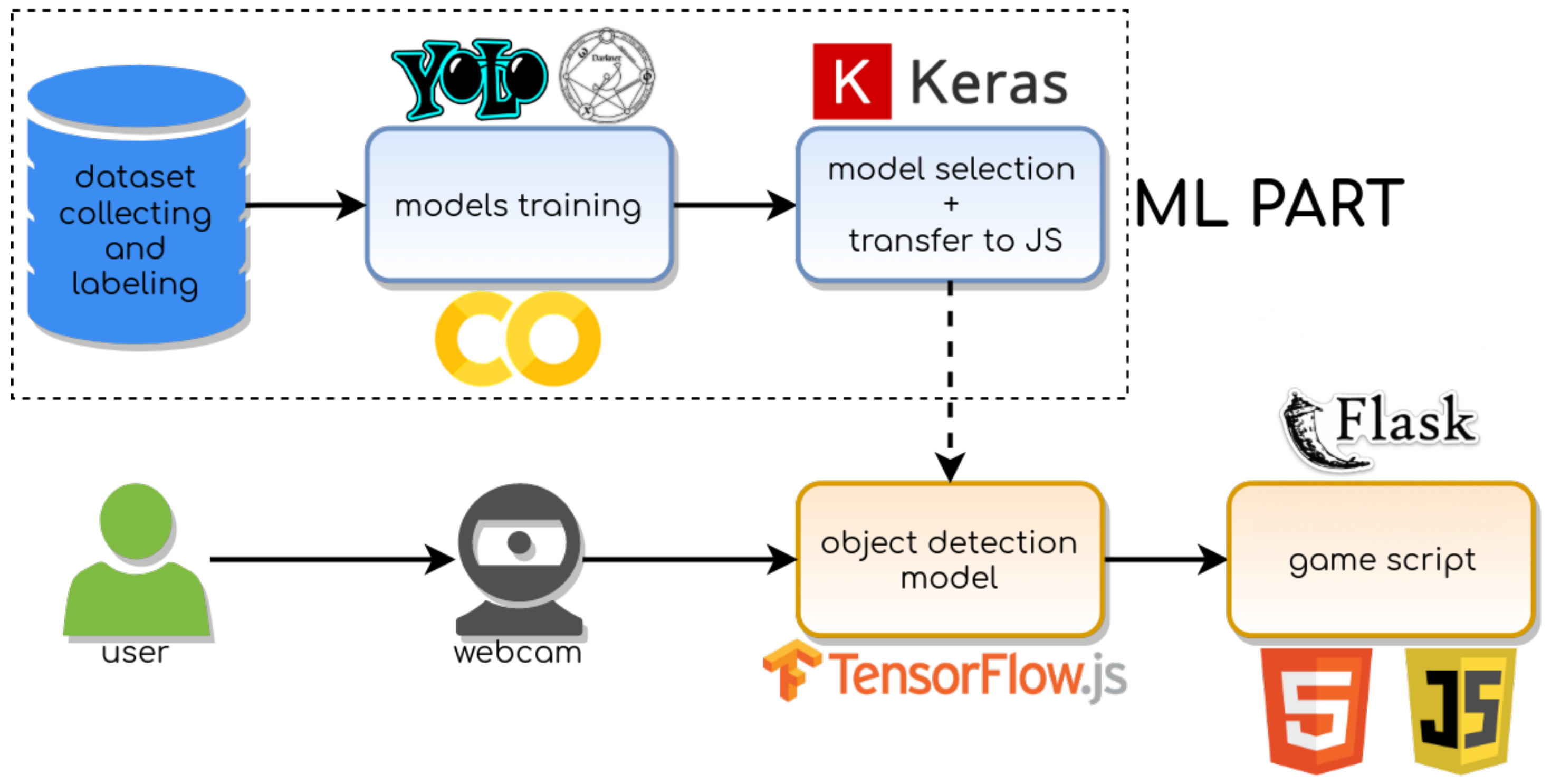

Figure 1: Project map

Model architecture

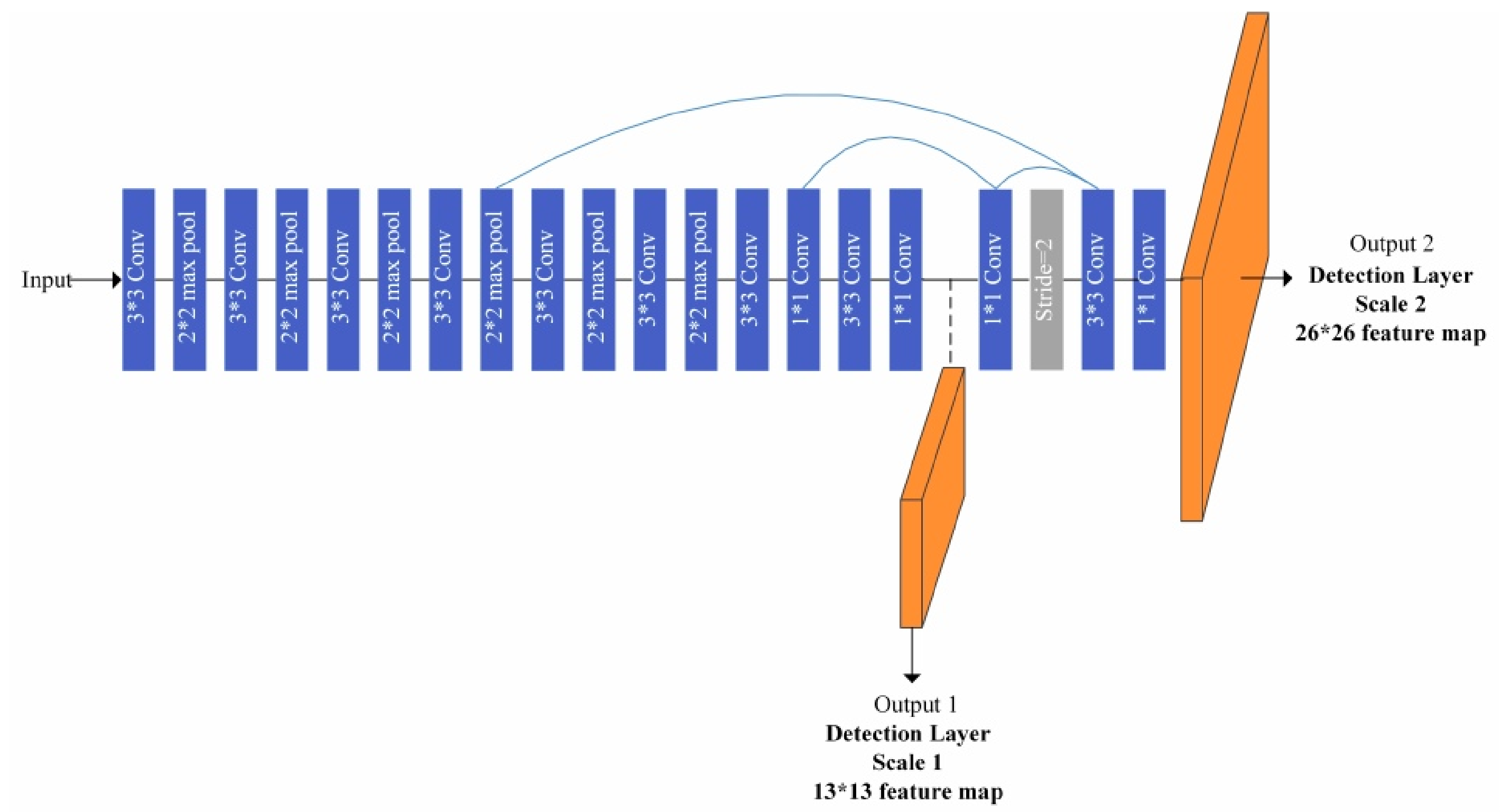

Figure 2: Vanilla tiny-YOLOv3 architecture

The tiny-YOLOv3 model is a smaller version of the YOLOv3 model. tiny-YOLOv3 uses 7 convolutional layers which are composition of Convolution2D BatchNormalization and LeakyRelu, and then features are extracted by using a small number of 1x1 and 3x3 convolutional layers. tiny-YOLOv3 uses the pooling layer instead of YOLOv3's convolutional layer with a stride of 2 to achieve dimensionality reduction. XNOR tiny-YOLOv3 model uses primary binary operations to perform convolutions which accelerates detection.

Data

4 gestures were selected for detection. For better performance custom dataset was created [2]. It contains 7725 images(approximately 2000 images for each class) with bounding box annotations in text files. Images were obtained by framing videos captured by webcams on different laptops. Bounding box annotations in Darknet format were created with help of Yolo mark GUI [3]. Due to not great amount of data I chose 9:1 train-test ratio.

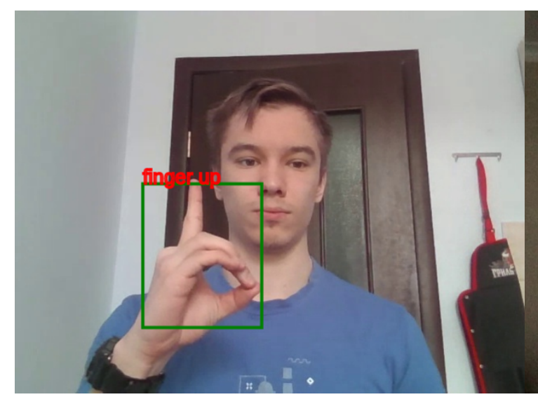

Figure 3: finger up

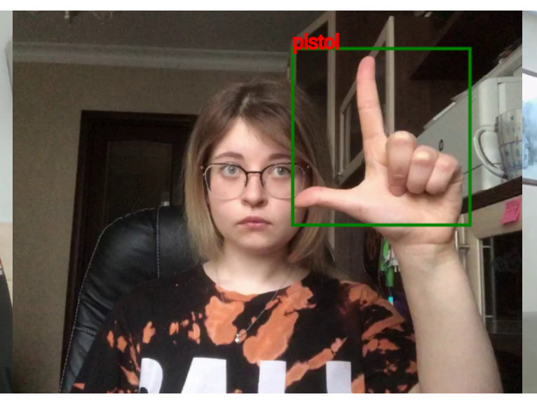

Figure 4: pistol

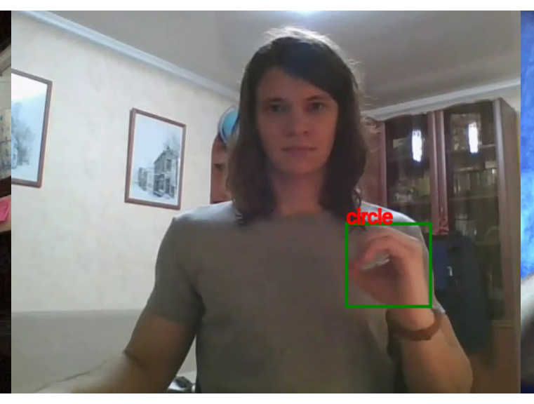

Figure 5: circle

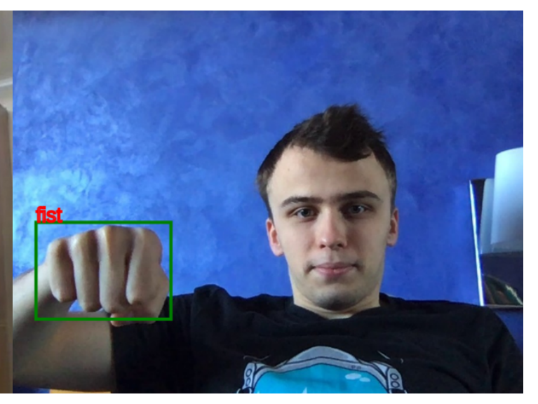

Figure 6: fist

Training and model transfer

In order to improve detection performance anchors were recalculated using k-means clustering. Models were trained in Darknet [4] with initial weights pre-trained on COCO dataset and ADAM optimizer with momentum=0.9, decay=0.0005, β_1=0.9, β_2=0.999 ε=0.000001. They were trained on Tesla P100-PCIE-16GB GPU kindly provided by Google Colab with batch size=64. Training of each model took $\approx$ 3 hours.

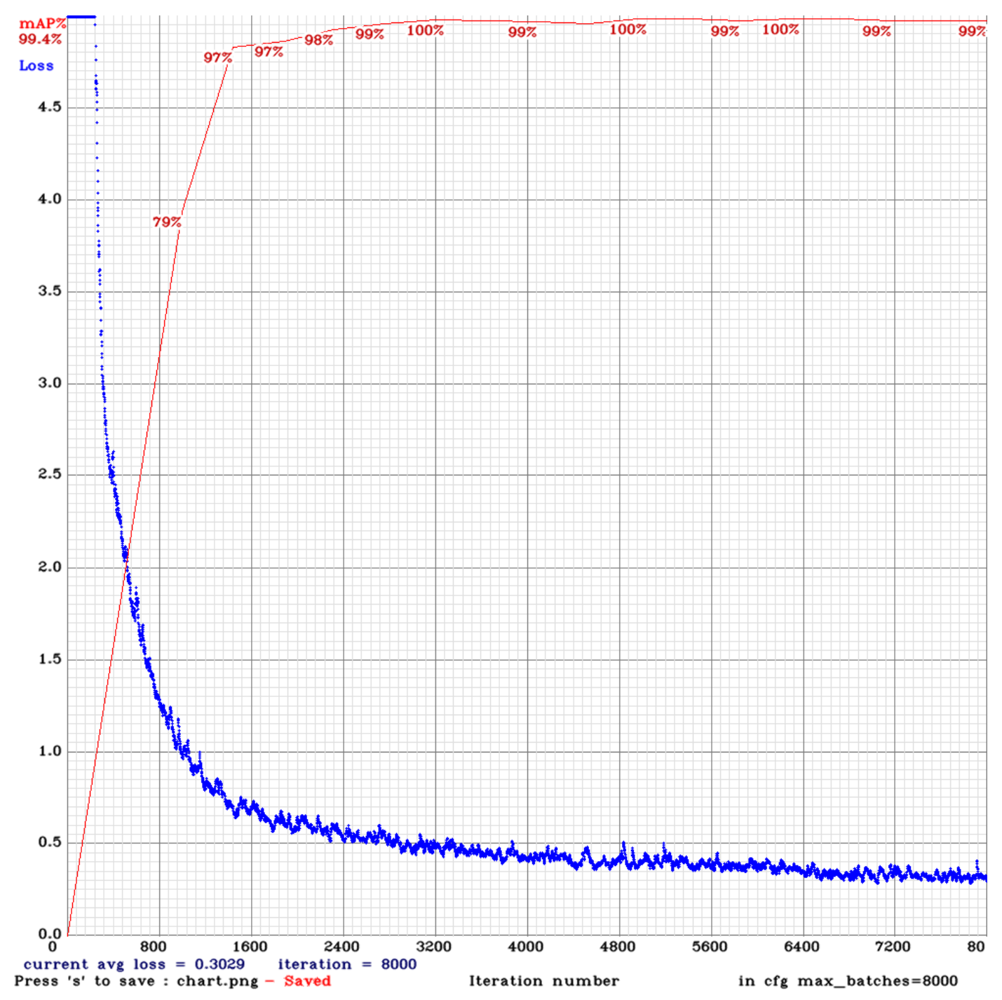

Figure 7: tiny-YOLOv3-416

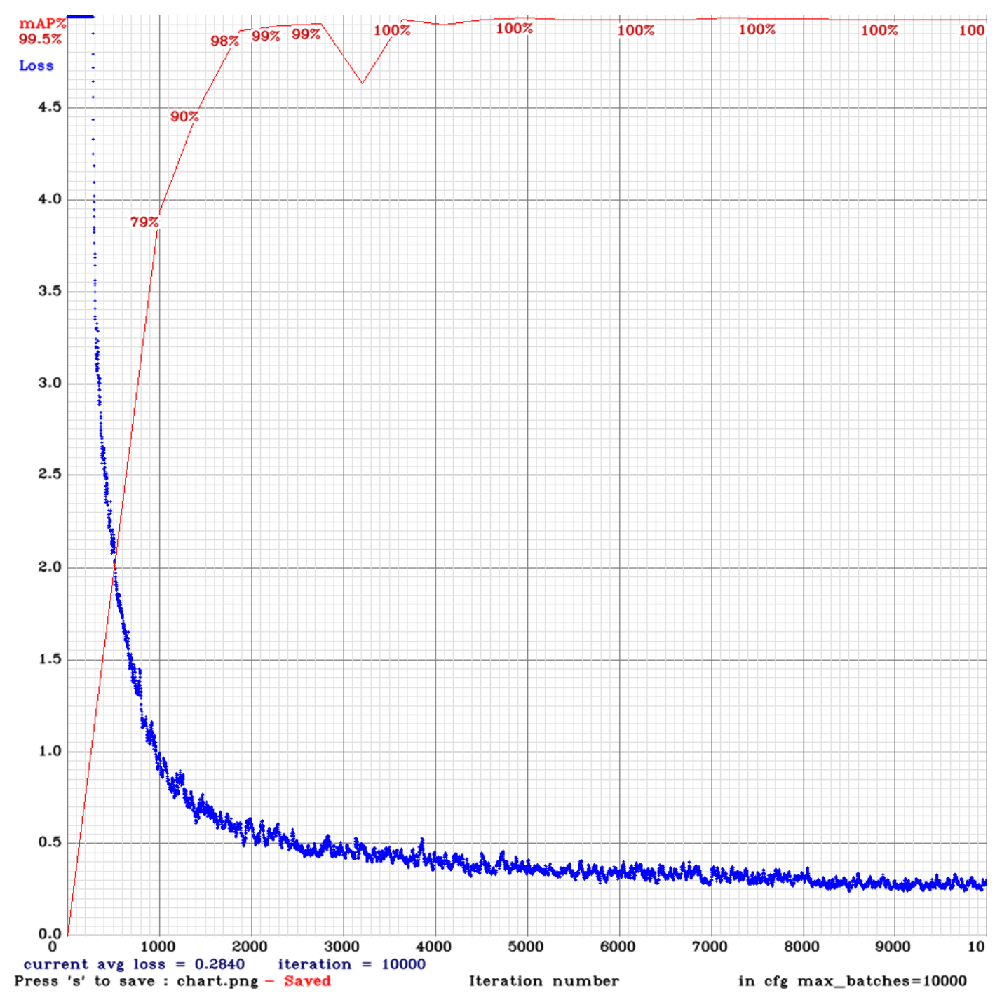

Figure 8: tiny-YOLOv3-256

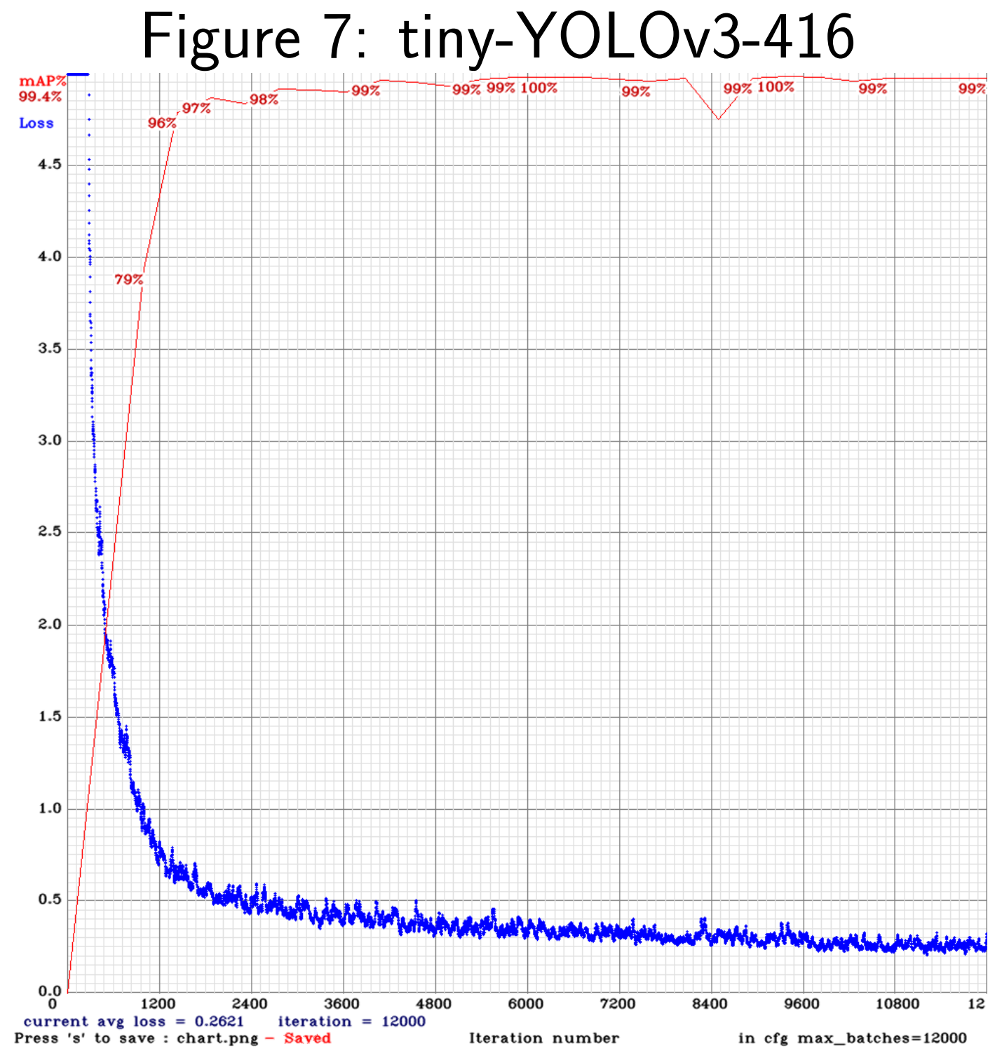

Figure 9: tiny-YOLOv3-192

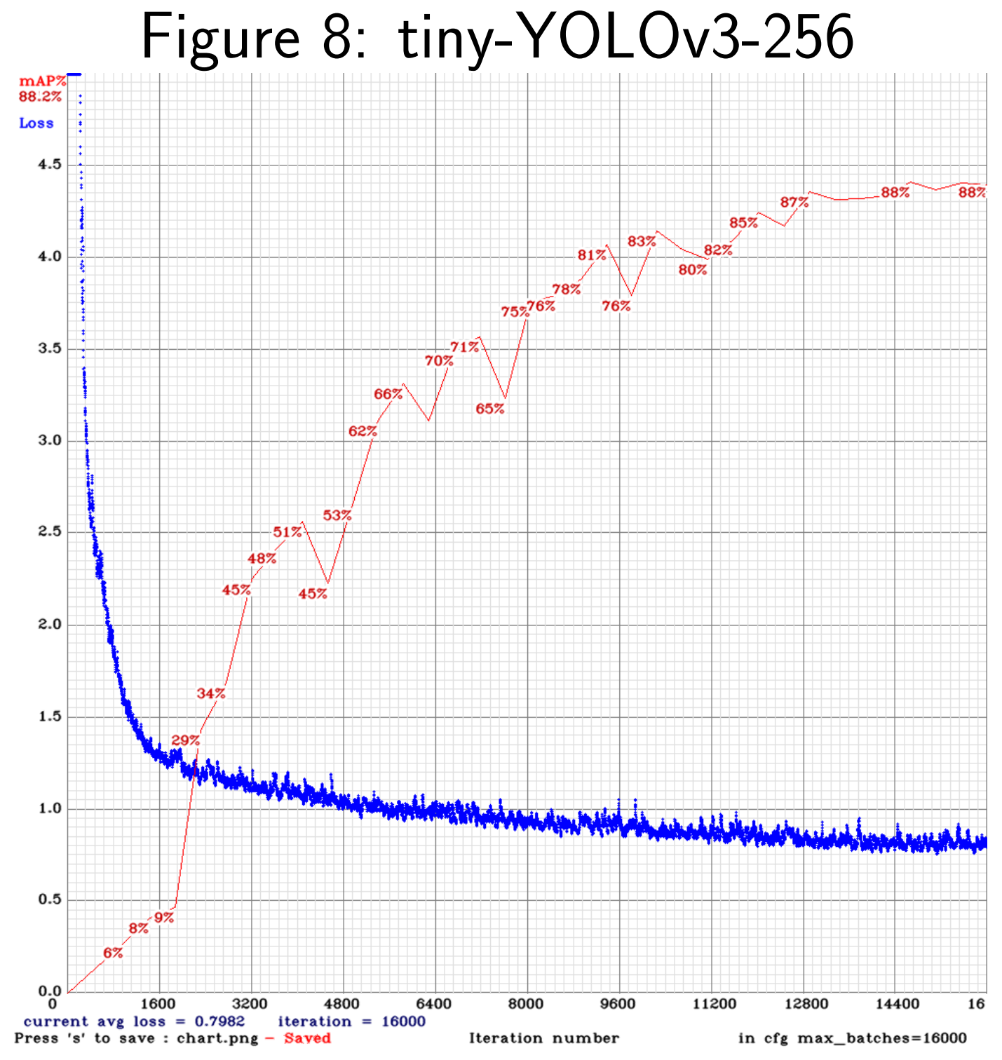

Figure 10: xnor tiny-YOLOv3-416

In order to reach good trade-off between accuracy and FPS I trained tiny-YOLOv3 models with 416x416, 256x256 and 192x192 input size. Obtained models were at first converted to Keras [5] and then to TensorflowJS [6] for browser usage. You can observe validation results in the table below.

model	number of iterations	$mAP_{.50}$	FPS
tiny-YOLOv3-416	8000	99.4	14
tiny-YOLOv3-256	10000	**99.5**	27
★ tiny-YOLOv3-192	12000	99.3	**35**
XNOR tiny-YOLOv3-416	**16000**	88.1	15

Let's play!

Final TensorFlowJS model(★) was used to control browser Breakout game via webcam. Game was written on JS and HTML.

Controls:

- Use "circle" gesture to move mouse cursor
- Use "finger up" gesture to click on buttons
- Use "fist" gesture to move the paddle
- Use "pistol" gesture to detach the ball from sticky paddle

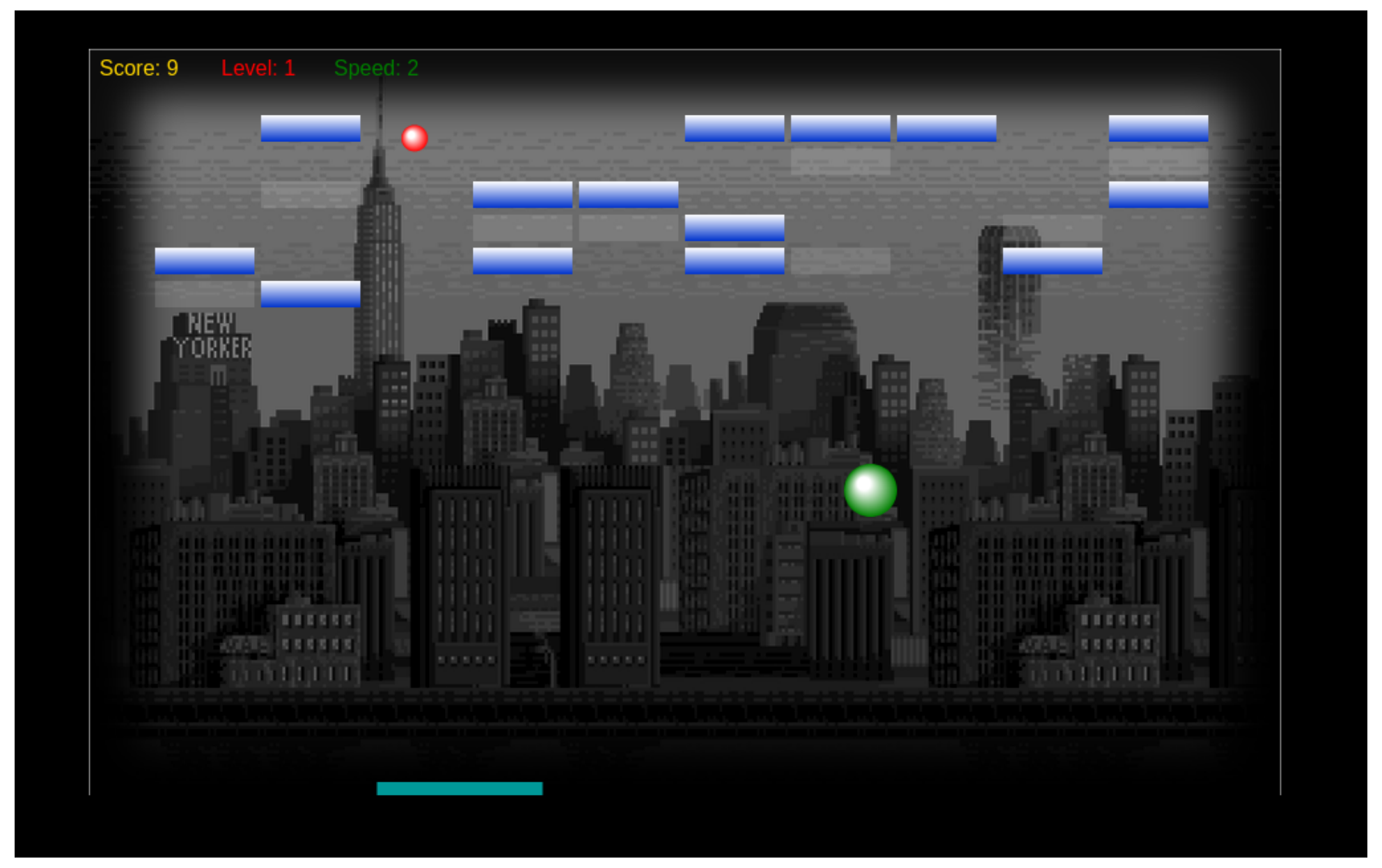

Figure 11: In-game footage

Results

Tiny-YOLOv3 models were trained on custom dataset. Fastest model was selected and converted to TensorFlowJS to perform object detection in the browser to control Breakout game. Code is available here: `https://github.com/vovaf709/Breakout-YOLO`.

References

[1] A. Farhadi J. Redmon. Yolov3: An incremental improvement. *https://arxiv.org/pdf/1804.02767.pdf*, 2018.

[2] People who shared their videos with gestures.

[3] Yolo mark. *https://github.com/AlexeyAB/Yolo_mark*.

[4] Darknet. *https://github.com/AlexeyAB/darknet*.

[5] Keras yolov3. *https://github.com/qqwweee/keras-yolo3*.

[6] Keras to tfjs. *https://github.com/tensorflow/tfjs/tree/master/tfjs-converter*.