Human Trajectory Prediction in First-Person View

Advanced Deep Learning for Computer Vision

Nikita Kister, Marc Katzenmaier

Abstract

This work covers a new approach to predict human trajectories in first person view using a combination of the YOLO object detector, a convolutional LSTM, a flow network and a convolutional decoder LSTM. This was done with the data provided by the Multiple Object Tracking Benchmark (MOT17).

YOLO

In this work the multiclass object detector YOLO (You Only Look Once) was adapted for person detection and fine tuned on our dataset. Therefore, the anchor boxes were recalculated to fit tighter to the ground truth boxes. Conceptually the YOLO network can be separated into an encoder part and some convolutional layers which then produce the final output of the network as seen below.

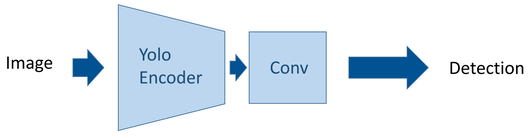

Figure 1: YOLO architecture

During fine tuning a drastic performance gap between input resolutions was discovered as shown in tab. 1. It can be explained by the higher pixel count per person, this effects especially small boxes with humans far away. The problem with small objects is a known drawback of the YOLO v2 architecture compared to other object detectors like the Single Shot Detector (SSD) or YOLO v3 which both detects persons on different feature levels.

Figure 2: Results of YOLO detection, the rows corresponds to rows in Tab. 1

Method	mAP0.5
Yolo train 416 val 416	0.305
Yolo train 416 val 832	0.479
Yolo train 832 val 416	0.278
Yolo train 832 val 832	0.577

Table 1: YOLO results where the first number refers to the training resolution and the second to the validation resolution

YOLO LSTM

The second step to our goal was to use video data for our detection. Therefore, the last two convolutional layers of the network were exchanged with a convolutional LSTM and a convolutional layer, as displayed in fig. 3. For using the capabilities of the LSTM our network was trained with sequences containing 20 frames. Due to memory limits it wasn't possible to train our YOLO LSTM end to end and therefore it was decided to freeze all layers up to the Convolutional LSTM, which reduced the memory consumption drastically.

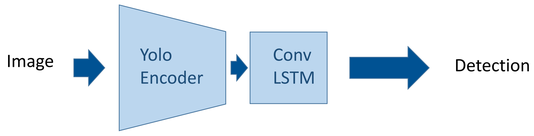

Figure 3: YOLO LSTM architecture

Method	mAP0.5
Yolo832	0.577
YoloLSTM832	0.559

Table 2: YOLO LSTM results

One can see comparable performance compared to our YOLO. By comparing the results as shown in Tab. 2 it can be seen slightly less jittering of the boxes, especially for the man with the blue shirt in front of Fig. 4. This effect only applies if the box stays roughly the same size and therefore is generated with the same anchor box. As a drawback one can see some slightly reduced box alignment compared to YOLO which is also displayed in the mAP0.5 value.

Figure 4: YOLO LSTM results, first row YOLO, second row YOLO LSTM

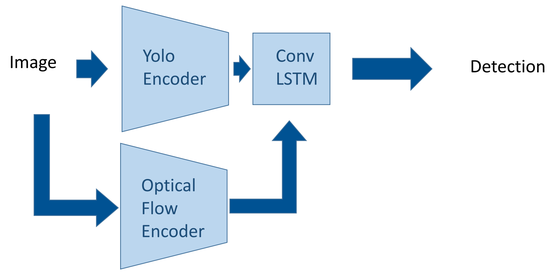

Figure 5: YOLO LSTM Flow architecture

Incorporating Flow

The next step is to condition the detection with optical flow. We used a pretrained FlowNetS to calculate the optical flow. Instead of using the calculated flow, we use the feature maps produced by the early decoder parts of FlowNetS. We resize them with a simple convolutional layer and concatenate them with the feature maps of the YOLO detector as shown in fig. 6. The results are presented in the tab. 3 below.

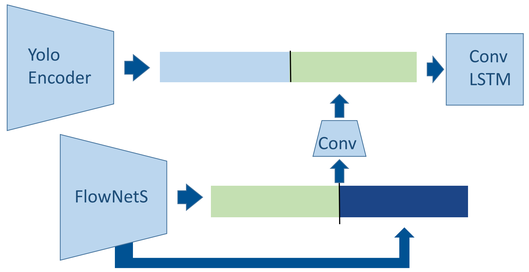

Figure 6: Conditioning with optical flow.

Method	mAP0.5
Yolo832	0.577
YoloLSTM832	0.559
YoloLSTMFlow832	0.584

Table 3: Comparing the performance of the final three detectors.

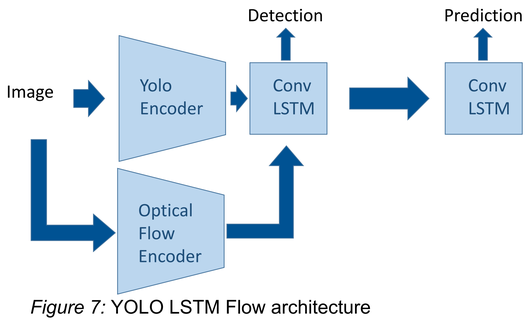

Figure 7: YOLO LSTM Flow architecture

Predicting Future Frames

The decoder is initialized with the hidden states of the encoder. It uses the transformed output of the YOLO model as the input.This allows for a joint prediction of all pedestrians. In each step we predict the difference in x, y, w, h.To predict further into the future we use the sum of the old input with the current output as the next input.

To accelerate the training we used teacher forcing in different variations:

1. Pretrain the network with the real boxes as input and then train it with the self generated input.
2. Alternate between generated and real input for each batch.
3. Shift the labels by the difference between the last detection output and the ground truth boxes.

Figure 8: Prediction results.

Method	Final Displacement	Average Displacement
pretraining tf (416x416)	31.35	24.20
alternating tf (416x416)	39.32	32.18
label correction (416x416)	37,66	29.96
big images (832x832)	25,57	17,8

Table 4: Comparing the performance of different training approaches. The first three were trained on 416 by 416 images Evaluation is done on 832 by 832 images.