Transferring Rich Feature Hierarchies for Robust Visual Tracking

mingzailao

2016-09-08 Thu

Outline

Introduction

Goal

Given this single (labeled) instance, the goal is to track the movement of the object in an online manner.

The Contributions

- To alleviate the overfitting and drifting problems during online tracking, we pre-train the CNN to distinguish objects from non-objects instead of simply reconstructing the input or performing categorical classification on large-scale datasets with object-level annotations.

Introduction

The Contributions

- The output of the CNN is a pixel-wise map to indicate the probability that each pixel in the input image belongs to the bounding box of an object. The key advantages of the pixel-wise output are its induced structured loss and computational scalability.
- We evaluate our proposed method on an open benchmark as well as a challenging non-rigid object tracking dataset and obtain very remarkable results. In particular, we improve the area under curve (AUC) metric of the overlap rate curve from 0.529 to 0.602 for the open benchmark.

Overview

Two Stages

- offline pre-training stage
- online fine-tuning and tracking stage

Pre-training Stage

We train a CNN to learn generic object features for distinguishing objects from non-objects.

Tracking Stage

Fine-tuning the parameters so that CNN can adapt to the target.

Overview

For Robustness

Running two CNNs concurrently during online tracking to account for possible mistakes caused by model update.

Objectness Pre-training

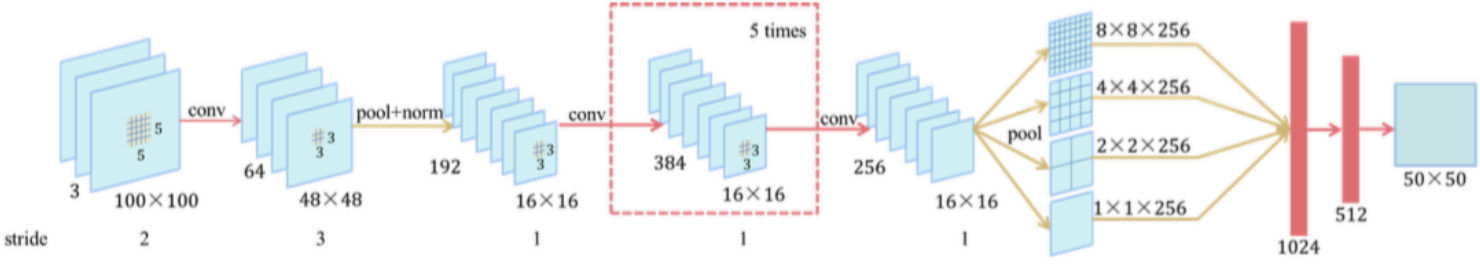

Figure: Structure of the pre-training CNN

Objectness Pre-training

Difference between the tracking model CNN and the classification CNN

The output of the CNN is a 50×50 probability map rather than a single number. Each output pixel corresponds to a 2×2 region in the original input,with its value representing the probability that the corresponding input region belongs to an object

Objectness Pre-training

Mathematically Loss Function

$$\min_{p_{i,j}} \sum_{i=1}^{50} \sum_{j=1}^{50} -(1 - t_{ij}) \log(1 - p_{ij}) - t_{ij} \log(p_{ij}) \quad (1)$$

1. p_{ij} : prediction of (i, j) position
2. t_{ij} : binary variable denotes the ground truth of (i,j) position

Online Tracking

Before

Fine-tuning the network using the annotation in the first frame.

Online Tracking

Basic online tracking pipeline

Two CNNs which use different model update strategie.
After fine-tuning using the annotation in the first frame, we crop some image patches from each new frame based on the estimation of the previous frame.

- Input : the estimation of the previou frame
- Output : some image patched from each new frame

CNN forward out : probability map for each of the image patches.
The final estimation is then determined by searching for a proper bounding box

Bounding Box Determination

The first step of the tracker when a new frame comes

Determine the best location and scale of the target.

1. specify the possible regions that may contain the target and feed the regions into the CNN.
2. decide the most probable location of the bounding box based on the probability map.

Bounding Box Determination

Search Mechanism

Using too small search regions makes it easy to lose track of a target under fast motion, but using too large search regions may include salient distractors in the background.

1. First, all the cropped regions are centered at the estimation of the previous frame.
2. Then, we start searching with the smallest scale.
    1. If the sum over the output probability map is below a threshold, then we proceed to the next larger scale.
    2. If we cannot find the object in all scales, we report that the target is missing.

Bounding Box Determination

Generating Bounding Box

After we have selected the best scale, we need to generate the final bounding box for the current frame.

1. determine the center of the bounding box.
2. estimate its scale change with respect to the previous frame.

Bounding Box Determination

How to determine the center of the bounding box : a density based method

1. First we set a threshold τ_1 for the corresponding probability map and finds a bounding box with all probability values inside above the threshold.
2. Next, the bounding box location under the current scale is estimated by taking an average over the different values of τ_1.

Bounding Box Determination

How to find a proper scale after we get the center of the bounding box

Let P denote the output probability map and p_{ij} the (i, j) th element in P, the score of the bounding box for top-left corner (x, y), width w and height h:

$$c = \sum_{i=x}^{x+w-1} \sum_{j=y}^{y+h-1} (p_{ij} - \epsilon) \cdot w \cdot h \tag{2}$$

where balance the scale of the bounding box. we also repeat with several ϵ values and average their results for robust estimation.

Differentially-paced Fine-tuning

Key :using two CNNs during online tracking

1. CNN_S account for short-term appearance.
2. CNN_L account for long-term appearance.

First

CNN_S and CNN_L are fine-tuned in the first frame of a video.

Afterwords

CNN_L is tuned conservatively while CNN_S is tuned aggressively.

Final

The final estimation is then determined by the more confident one.

Differentially-paced Fine-tuning

Notes for determined the CNN_S or CNN_L

There exist more advanced model update methods, the simple scheme works quite well in practice.

Details

CNN_S is updated when there exists a negative example such that

$$\sum_{i=1}^{50}\sum_{j=1}^{50} p_{ij} > \tau_2 \tag{3}$$

Differentially-paced Fine-tuning

Details

CNN_L is updated if :

$$\sum_{i=x}^{x+w-1} \sum_{j=y}^{y+h-1} p_{ij} > \tau_3 \cdot w \cdot h \tag{4}$$

where (x, y, w, h) denotes the output target bounding box in the current frame.

Differentially-paced Fine-tuning

Details

In each update, we need to collect both positive and negative examples.

- For positive examples, we sample them in four scales based on the estimation of the previous frame, Random translation is also introduced to eliminate the learning bias to the center location.
- For negative examples, we crop eight non-overlapping bounding boxes around the target in different directions in two scales

Implementation Details

For a Pre-training of CNN

- Start with a learning rate of 10^{-7} with momentum 0.9 and decrease the learning rate once every 5 epochs. We train for about 15 epochs in total.
- To alleviate overfitting, a weight decay of 5×10^4 is used for each layer and the first fully connected layer is regularized with a dropout rate of 0.5.

Implementation Details

For Fine-tuning

- we use a larger learning rate of 2 Œ 107with a smaller momentum of 0.5.
- For the first frame, we fine-tune each CNN for 20 iterations.
- For subsequent frames, we only fine-tune for one iteration.

Implementation Details

Others

- τ_1 ranges from 0.1 to 0.7 with a step size 0.05.
- τ_2 (the threshold of sum of confidence for negative examples) :100.
- τ_3 (the threshold for CNN_L) :0.8
- ϵ (the normalized constant for searching proper scales) range from 0.55 to 0.6 with a step size of 0.025.