Benchmarking losses for deep learning laxness detection

Adrien Chan-Hon-Tong

Benchmarking losses for deep learning laxness detection

Adrien CHAN-HON-TONG

July 17, 2017

Abstract

In object detection, classical rule for accepting a match between a ground truth area and a detection is a 0.5 jacard ratio. But does user care about this ? Especially, does deep network training care about this ? And if yes, may user accept some relaxation of the problem if it can help the training ?

In this paper, we add laxness on object detection in remote sensing image by deep learning pipeline. We present several models and relaxation of the classical detection problem. Our preliminary results on different public dataset show that some relaxations are very facilitative to train the pipeline outperforming the same pipeline learned for strict detection.

1 Introduction

Interactions between users and deep learning designers usually go through one classical academic problem. However, users are not aware on the difficulties introduced by this model of their needs, and inversely, deep learning designers are not aware of the exact needs of the users and are not well able to explain what could be relaxed and how it could be facilitative.

Off course, we acknoledge that academics need common problem in order to compare each other. But, when it came to adapt deep network to some real life problems, it may be relevant to relax classical model of the problem.

In this article, we apply this statement to object detection in image (see figure 1), which is one of oldest computer vision problem.

For academic comparison, there is a classical model of this problem. Each testing image is annotated by hand, for each image, ground truth is a set of bounding box (eventually labelled). Detector should take one image as input and product a set of bounding box (eventually labelled and eventually with confidence level). A detection is allowed to match an annotation if intersection over union of the two boxes is greater than 0.5 (and if label are the same when present). A maximization of the number of matchs is computed. Then each non matched annotation is a miss detection and each non matched prediction is a false alarm. Classical measure of the fairness of the detector is given by combining the precision

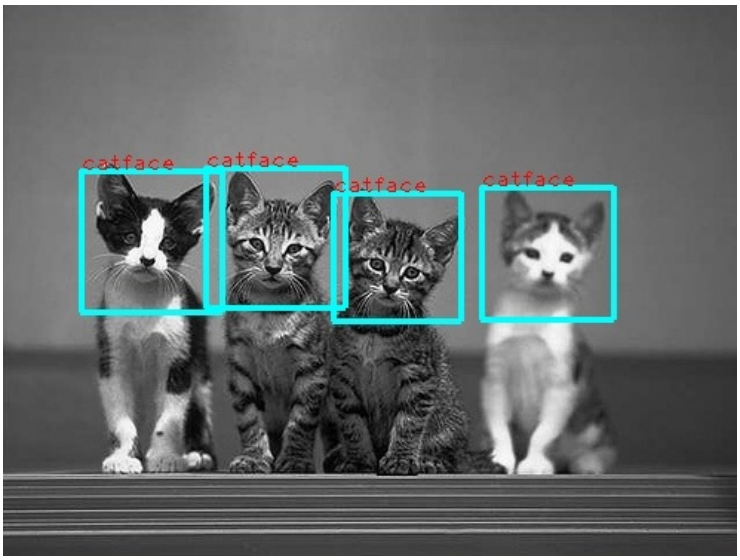


Figure 1: Illustration of object detection: putting bounding boxes on object in the image

(number of correct detections over number of detections) and the recall (number a correct detections over number of annotations), for example with the so called F score or G score. Alternatively, detection are sorted according to the confidence level, and, a elementary score is computed for each rank. All this elemantary score are merged into a global score (for example by considering that elemantary scores form a curve and by computing the area under the curve).

If, for example, the detector is applied to count cars on a parking this matching model is completely relevant because 1 to 1 matching is needed to produce an accurate estimation of the number of cars.

But, there are a lot of application where this strict model of the problem is not relevant. For example, if detections should be checked by an human anyway (for example for search and rescue), the only matter is to produce coarse area of interest because both 1 to 1 matching is useless and false alarm close to correct detections are not an issues. Also, in this contexte computing the extends of the object (i.e. a bouding box) is not needed, a simple dot coarsely centrered on the object will do the job.

Now, these considerations are not new [1]. And, at first side, one just need to ajust the metric to his own problem without changing the detector.

This is probably true for large hand crafted detector pipeline where only small blocks rely on training because changing the metric will have low influence on the complete pipeline. But, on end to end deep learning network, metric is the only prior on the problem, thus changing the

[1] these considerations are not new and not restricted to our field, for example it is well known in optimization that different models can exist for a common problem [4]

metric - and so the network loss - may have a critical influence on the performance.

The contribution of this paper is to show that losses have a critical influence on the performance for deep learning detector.

More precisely, we add some user oriented relaxations on detection, and given these relaxations, we show that similar pipelines using same training data can behave highly differently on testing data depending on the way we model the detection problem.

Our preliminary results show that modeling the problem as a simple grid classification is very facilitative for network training probably because it match the pooling structure of the classical networks.

In the next section, we will present related work, then we will present the model we selected to the benchmarking. Section 4 will present different datasets and results before the conclusion.

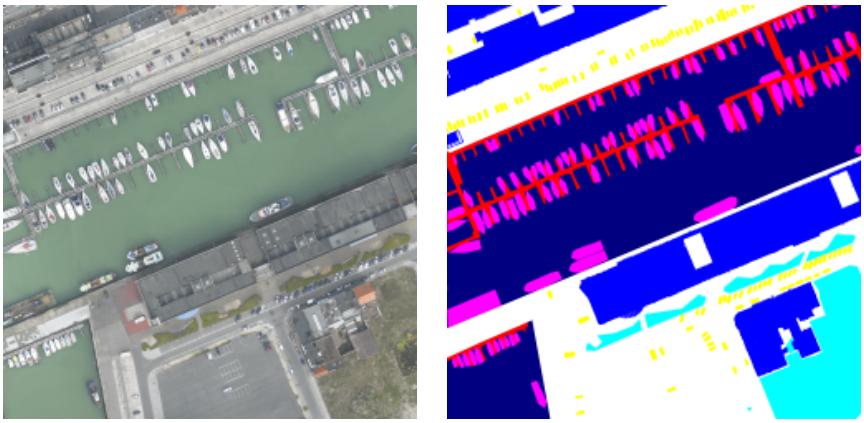
2 Related works

When the deep learning [8] appear in classification, object detections community does not directly follow the move: at this time, even an efficient implementation of [8] could not processed more that 100 small images per second. As the dominant framework for object detection was based on sliding windows processing only 100 windows per second was definitely not enough.

At this time, a typical detector was [5]. In [5], hand crafted features are extracted from all the images (train and test). Then a learning problem is created by putting in one side features extracted from ground truth boxes and on the other side features extracted from random boxes which do not match ground truth. This problem is solved by svm (with an implementation close to [3]). At test time, the image is explored by sliding windows, the svm model is applied on the features of each windows. The set of windows selected by the svm are then pruned by non maximal suppression (svm will behave likely on two very close windows but by keeping the two boxes one will likely become a false alarme). Finaly, the learning step can be done multiple times to look for hard negative instead of random ones, this is the so called bootstrapping method.

The real apparition of deep learning in detection was with [6] which both highly increases performance of the previous state of the art and on the same time breaks the sliding windows framework by enlighting region proposal framework. In [6], an ad hoc algorithm extract bounding box in the image folding the detection problem into a classification one without exploring all the boxes of the sliding way. Classification in [6] is then done by [8].

Current state of the art of detections [12] are still based on [6] but the adhoc region proposal is now a deep network based on the same first layers than the classifier. Also, as convolution can take image from arbitrary spatial dimension, first layers of the network are computed on the complete image and not on windows, fastening the first layers computation by the size of the windows compared to the classical way.



(a) a remote sensing images (b) a semantic mask

Each color in the semantic mask is expected to correspond to a kind of object in the image. For example, boats in the image are in pink in the semantic mask while cars are yellow.

Figure 2: Example of semantic segmentation of a remote sensing image

This last idea of convolution taking arbitrary image gives rise to a lot of semantic segmentation works like [2, 13] (or [9] for remote sensing). Semantic segmentation (see 2) is the goal of producing semantic mask of an images. Or, with others words, the goal of deciding a semantic label for each pixel of an image.

Optimizing deep learning is straightforward in semantic segmentation: output is a highly structured image with predefined size and a number of channels corresponding to the number of type of objects, on which error with the desired manual annotation can be easily measured. Also, on dense semantic segmentation, there are often multiple labels with large number of instance. Thus, small object pixel are unbalanced toward surface pixels (e.g. car vs building in 2), but there are multiples surface labels forbidding the deep network to produce a mono label map.

Recently [1] shows that this semantic segmentation way can be postraited to produce accurate detection.

At this point, there are two possible strategies to handle detections: a way based on region proposal + classifier (R-CNN framework) and a way based on semantic segmentation.

Now, for semantic segmentation, [7] shows the relevancy of decreasing the weight given to pixels close to a boundary of the ground truth segmentation. At first view, [7] seems to offer the opposite of bootstrapping: bootstrapping consists in looking for hard negative while [7] offer to discard pixel close to a boundary which are typically hard. However, pixel close to a boundary are more *ambigious* than *hard* (typically, pixel close to boundary are likely to be wrongly annotated (see figure 2 because manual annotation is coarse due to the difficulty to realize it). In other word, hard example are clear errors from the current detector while hard

boundaries examples are discutable errors inherent to the problem.

Again, considering boundaries or not may depend of the finality of the algorithm. If the goal is to produce a precise map from an image, having a pixelwise segmentation is required, and in this case, border pixel have to be considered, otherwise the segmentation will be like coarse and unsuitable for making a map. But if the goal is to produce coarse segmentation as a first step toward detection, it is much more important to focus on non border hard negatives (which will create false detections) and not on border hard negatives that may not forbid the match with the annotation (and thus may not lead to a false detection).

So, after this fast review of related works, there are at least three strategies to handle detection: classifier based, segmentation bases and borderless segmentation based.

In the next section, we will offer models for laxness detections, we will also formalize the different strategies for detection and finally we will introduce a new one.

3 Different models for laxness detections

3.1 Laxness detections

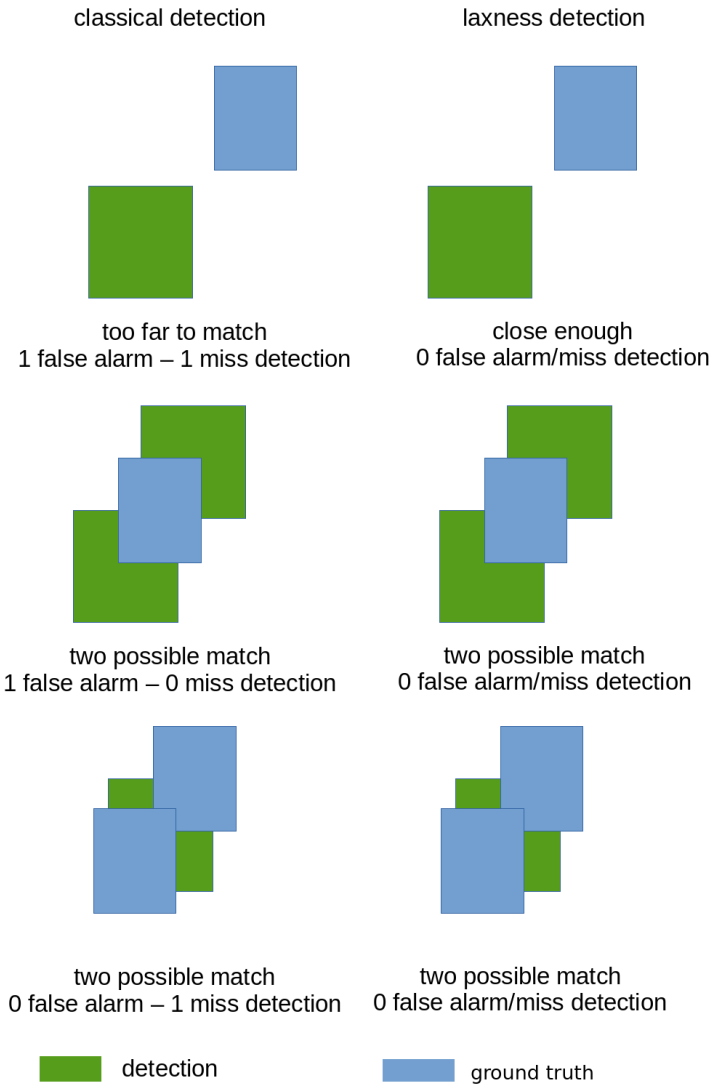
The problem we try to handle is to produce an *output* that will be postraited by a person to finally lead to detections. The use case is to fasten detections in very large remote sensing images (e.g. 20km x 20km at 50cm - or in pixel 40000x40000 with target not exceding 16x16)

This way, we do not really care about *false alarmes* in classical sense that are sufficiantly close to ground truth - the person checking the detection will be able to filter.

So we adopt a very laxness matching criterion: output is expected to be a set of middle size areas (32x32 in the remote sensing example) grid spaced with half overlapping (or pruned by non maximal suppression). An area is considered like a false alarm only if it contains no center of any annotations. An annotation is considered like a miss detection if it is not contained by any areas.

We can see in figure 3 that this model is much more laxness that the classical detection problem. Now, the idea is not just to see if performances increase when relaxing the metric (which is a trivial expectation) but to see if the relaxation allow new ways to handle the problem - new ways which may be much more adapted to train deep network and thus providing performance gap.

As a example, in strict detection, all pipelines performs non maximal suppression to remove redondant detections. But, learning deep network behind a non maximal suppression is not easy. So, by allowing to remove the non maximal suppression, we do not just adapt a pipeline to a new metric (which will produce only small increase of performance), we allow a end to end training of the deep learning block which can skyrock the performance of the pipeline. Other example, in [7], testing performances measured with strict metric increase when relaxing metric on training.



In laxness detection, we allow a much larger distance to match (2 times the size of the object) and we do not require a 1 to 1 matching (i.e. one detection can count for several annotation and two detections can be allowed for a common annotation). However, we require detections to be grid spaced (with a step corresponding to the size of an object) or pruned by non maximal suppression to forbid too much detections on an easy target (which could bias precision evaluation).

Figure 3: Illustration of relaxation added to the classical detection problem

So, in this case, the increase is only due to a good relaxation to learn and not to a lower metric.

Such protocol (hard metric on test - weak metric on train) could be good to evaluate if learning behaves better and not just exploit better the weakness of the metric. However, such evaluation is only possible if the training relaxation does not change the algorithm output format which will be the case in the following.

3.2 R-CNN

In R-CNN, a first algorithm is applied to produce candidate. The precision of this first algorithm is expected to be very low and the main requirement for such algorithm is to be training free. Candidates are pruned by non maximal suppression.

A training time, candidates are matched with the ground truth and a classifier (a deep network) is learnt on the positive/negative (positive are candidate which match the ground truth - negative are candidate which not match).

At testing time, the classifier is applied on all candidate and accepted candidates form the detections.

3.3 Segment before detect

In segment before detect, we convert the bounding boxes annotations into a binary mask (background for pixel outside all boxes and foreground for pixel inside one or more boxes).

At training time, we learn a fully convolutionnal deep network in segmentation fashion (we want each pixel be well classified).

At test time, we produce detection by putting a bounding box around each connected component when using strict detection rules.

3.4 Coarse segment before detect

The third model to handle detection is like segment before detect but instead of producing a binary mask by labelling background all pixels outside all boxes and foreground all pixels inside one or more boxes, we produce a trinary mask: all pixels outside all boxes are background, all centers of all boxes are foreground and all other pixels are border.

When training border pixels are discarded from the optimization, thus the network only has to predict correctly the center of object and not the ground truth bounding boxes of the object.

3.5 Pooled detection

The last model to handle detection is the simpler and can only handle laxness detection: we explore the image by spaced sliding windows (with step corresponding for example to the average size of objects), each window containing an object should be classified as positive and each window containing no object should be classified as negative.

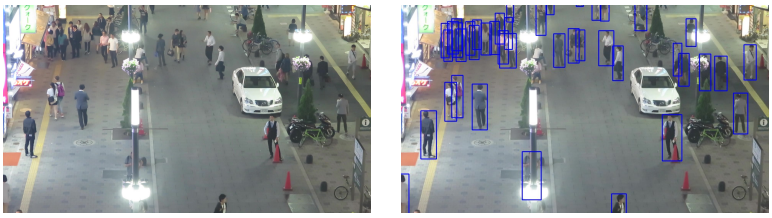


Figure 4: Illustration of the MOT 16 dataset.

This can be seen as a segmentation pooled by a factor corresponding to the selected sliding step. This can also be seen as a R-CNN with R outputing all windows as candidate.

At testing time, we just output the center of the windows accepted as positive.

4 Experiments

4.1 Datasets

We use 3 datasets to evaluate the influence of these different strategies for detection: MOT 2016 (MOT16) [10], VIRAT aerial dataset [11] (not video surveillance data - we use only the aerial videos). As no annotation has been released, we annotate it ourselves using our tool (annotations are planned to be released). We also rely on SACLAY which is a private dataset (but planned to be released).

MOT16 (https://motchallenge.net) is a multi objects tracking dataset: detection are provided, the goal is to keep temporal id on the detection. Here, we use this dataset only to learn to produce a semantic mask corresponding to the detections (we only keep person detection). The training part of MOT16 is composed of 6 HD videos taken from a pedestrian or car or surveillance camera (see figure 4).

For MOT16, we will use accuracy, gscore and iou score to evaluate all algorithms. Accuracy is the stablest measure, gscore is the product of precision per recall and iou score is the same as for the CITYSCAPE leaderboard (this is not related to the IoU of 2 boxes in detection to know if a matching is possible, this is a score computable from the confusion matrix that aim to give the same weight on positive and negative while acknowledging both two types of error).

The VIRAT aerial dataset is a set of videos which are low resolution and contain camera motions, highly textured background and very small object. As no public annotation has been released for this dataset, we annotated a subset of the frames in a person detection setting. In order to provide a diversity of situations, we chose to annotate about thirty sub videos of 400 frames containing at least one person distributed over the dataset (but discarded infrared images). Figure 5 shows examples of images from this sub dataset.

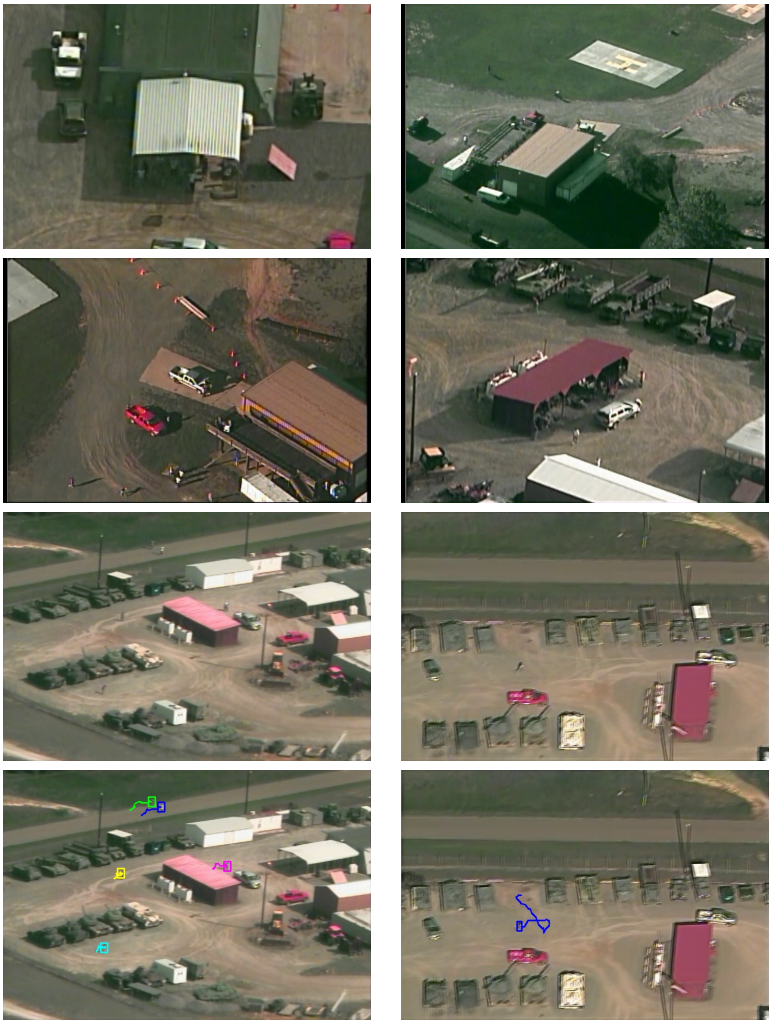


Figure 5: Illustration of the subset of VIRAT (plus annotation).

model	iou	gscore
R-CNN	30	19
segmentation	26	31
coarse	2	1

(a) strict detection

model	iou	gscore
R-CNN	45	44
segmentation	49	42
coarse	4	3
pooled	67	56

(b) laxness detection

Score are given in percentage. A 0.3 Jacard is required in strict detection while a 32 distances is allowed in laxness detection (grid spaced by 16).

Table 1: Preliminary results on VIRAT.

For VIRAT, we will use only gscore and IoU score to evaluate all algorithms. Accuracy is not relevant as 99,2% of the pixel are background pixel.

The SACLAY dataset is a set of aerial images at 20cm of resolution. Annotation is a set of point localized on cars. As for VIRAT, we will use only gscore and IoU score for SACLAY

4.2 Results

Currently, we have only preliminary results computed on a subset of VIRAT in table 1. The metrics are a 0.3 Jacard is required in strict detection while a 32 distances is allowed in laxness detection (grid spaced by 16). A HOG detector is used as region proposal for R-CNN. We use a VGG for pooled segmentation and a UNET for segmentation. We adapt the VGG in order that we can extract a layer after several pooling with the exact grid dimension.

From this preliminary results, we can notice that coarse segmentation perform poorly. As a matter of fact, coarse segmentation increase the unbalance (only 1 pixel positive per object all other are either discarded or background) which is not the case for pooled (1 cell positive per object when their are distant but the number of negative cell is globally the number of pixel divided by the size of the cell). But, pooled segmentation seems outperforms other strategies.

5 Partial conclusion

Even if our benchmarking is currently done on a small part of public datasets, the current trends seems clear: losses used to learning deep learning detectors have a dramatic influence over the results.

This invite to strengthen our current benchmark as the persistence of the current features will be an interesting knowledge for the community.

Acknoledgment

We want to thank Adle Koeniguer for providing the pointing ground truth of Saclay dataset.

References

[1] Nicolas Audebert, Bertrand Le Saux, and Sbastien Lefvre. Segment-before-detect: Vehicle detection and classification through semantic segmentation of aerial images. *Remote Sensing*, 9(4):368, Apr 2017.

[2] Vijay Badrinarayanan, Ankur Handa, and Roberto Cipolla. Segnet a deep convolutional encoder decoder architecture for robust semantic pixelwise labelling. In *arXiv preprint*, 2015.

[3] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

[4] Christoph Dürr and Mathilde Hurand. Finding total unimodularity in optimization problems solved by linear programs. *Algorithmica*, 59(2):256–268, 2011.

[5] Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *IEEE transactions on pattern analysis and machine intelligence*, 32(9):1627–1645, 2010.

[6] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Computer Vision and Pattern Recognition*, 2014.

[7] Mingyuan Jiu, Christian Wolf, Graham Taylor, and Atilla Baskurt. Human body part estimation from depth images via spatially-constrained deep learning. *Pattern Recognition Letters*, 50:122–129, 2014.

[8] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[9] Adrien Lagrange, Bertrand Le Saux, Anne Beaupere, Alexandre Boulch, Adrien Chan-Hon-Tong, Stéphane Herbin, Hicham Randrianarivo, and Marin Ferecatu. Benchmarking classification of earth-observation data: from learning explicit features to convolutional networks. In *International Geoscience and Remote Sensing Symposium*, 2015.

[10] Anton Milan, Laura Leal-Taixé, Ian D. Reid, Stefan Roth, and Konrad Schindler. MOT16: A benchmark for multi-object tracking. *CoRR*, abs/1603.00831, 2016.

[11] Sangmin Oh, Anthony Hoogs, Amitha Perera, Naresh Cuntoor, Chia-Chih Chen, Jong Taek Lee, Saurajit Mukherjee, JK Aggarwal, Hyungtae Lee, Larry Davis, et al. A large-scale benchmark dataset for event recognition in surveillance video. In *conference on Computer Vision and Pattern Recognition*, 2011.

[12] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*, 2015.

[13] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. *U-Net: Convolutional Networks for Biomedical Image Segmentation*. Springer International Publishing, 2015.