# Real-Time Excavation Detection at Construction Sites using Deep Learning

Bas van Boven[1(✉)], Peter van der Putten[1], Anders Åström[2], Hakim Khalafi[3], and Aske Plaat[1]

[1] LIACS, Leiden University, Leiden, The Netherlands
bas@basvanboven.nl, {p.w.h.van.der.putten, a.plaat}@liacs.leidenuniv.nl
[2] Accenture, Singapore, Singapore
anders.astrom@accenture.com
[3] Amsterdam, The Netherlands
hakimkse@gmail.com

**Abstract.** In this paper we present a robust approach to real world, real time action classification. It relies on a convolutional network based object detector to extract relevant shape and motion features and uses these features as input for an action classifier. Using a sequence of localization and classification information of various objects deemed relevant to an action, the model recognizes predefined actions in a reliable manner, and can localize these actions in camera footage in real time. Without loss of generalization, we study our approach within the context of a construction company that wants to prevent unauthorized excavation activities happening at their construction sites. We differentiate four excavation activities, two of which we detect on the basis of actions because the target pattern contains temporal features, and two of which we detect on the basis of object presence only. The system needs to operate in real time, on basic on-site hardware and under varying image conditions.

**Keywords:** Video analysis · Action classification · Convolutional neural networks · Feature engineering

## 1 Introduction

The detection and classification of specific actions in video is a difficult task, especially if computing resources and data are limited. Convolutional nets can be a powerful tool, especially to detect concepts without having to engineer specific features. They require a relatively large amount of labelled data though, and more importantly, when applied end to end there is limited ability to leverage domain knowledge to engineer features that might be useful for the task at hand.

In this paper we present a two staged approach that combines the benefits of deep learning with the flexibility and control of feature engineering. We apply it to a new real world problem, the detection of unwanted human or mechanical


W. Duivesteijn et al. (Eds.): IDA 2018, LNCS 11191, pp. 340–352, 2018.
https://doi.org/10.1007/978-3-030-01768-2_28

excavation at construction sites. We approach this problem from scratch, from gathering camera footage, labeling the data, training classifiers, integrating these into an end to end pipeline, and deploying and running the system in the real world. An additional constraint is that the system should monitor the site in real time, with minimal latency, processing feeds from four cameras using a laptop without internet connectivity. We first detect objects of interest, such as various parts of excavators, and then use domain knowledge to enrich this information with a variety of engineered features, and feed sequences of this data into a subsequent more standard classifier. The experiments demonstrate the validity and real world flexibility of the approach, which makes it a valid approach to explore for a range of real world problems with similar needs.

This paper is structured as follows. We will first introduce the business problem (Sect. 2) and related background (Sect. 3). Then we will describe our methods and approach for the end to end pipeline in Sect. 4, and report on experimental results (Sect. 5), followed by a discussion (Sect. 6) and conclusion (Sect. 7).

## 2 Problem Description

A construction company is facing losses due to unauthorized excavation activities at their building sites. Because of various reasons, such as misinterpretation of work instructions and erroneous reading of site maps, their workers often excavate at wrong locations. This can in turn lead to damage to important infrastructure like sewage pipes and power lines, and repairing such accidental damage can be expensive. In order to limit the cost of such damage, they need a system that can automatically detect unapproved excavation events.

Using the system, one should be able to deploy four different cameras on tripods, which are linked to the same laptop workstation by means of a wireless connection. On this workstation, one can configure the zones in which excavation is prohibited by means of a user interface. When excavation activities are detected within this zone, a SMS alert should be sent and an on-site alarm should be triggered. All excavation activities need to be detected before major damage has been done. This means processing needs to happen in real time with minimal latency. As internet connectivity cannot be guaranteed, all video processing and analysis needs to happen locally on a laptop (i7-6820HK processor, 32GB of DDR4-memory, a 512GB SSD and a NVIDIA GeForce GTX1080 GPU with 8GB of GDDR5X video memory).

The system should be designed to detect four different kinds of excavation activities. First of all, there is mechanical excavation by digging: in this situation, a mechanical excavator with a bucket attachment at the end of its arm is performing an action that will directly lead to it moving around ground or retracting the arm of the excavator in order to reach into the ground. Second, there is mechanical excavation by breaking: in this situation, a mechanical excavator with a breaker attachment at the end of its arm is performing an action

that will directly lead to it putting its piercer into the surface. As in the previous digging example, this definition does not encompass riding or turning, but it does include extending or retracting the arm in order to reach into the ground.

Third, we detect manual excavation by people who are crouching: in this situation, a worker is using a tool to manually dig into the ground, for which it is necessary for them to assume a crouching position. There are three reasons that the definition is formulated as such. First is the observation that it would be very difficult to detect all the individual tools used for manual excavation. The second reason is that for most of these tools, assuming a crouching position is necessary. Finally, local observations confirmed that workers are seldom crouching, if not for excavating. Thus, crouching people are a good indicator of manual excavation activities. Fourth, we detect manual excavation by an earth rod tool: in this situation, a worker is using an earth rod tool to manually drill a hole into the ground. An earth rod is a tool that consists of two parts: a long thin stick that can be beaten into the ground, and a hammer that can be used for this. As workers are generally not crouching when using this tool, it is necessary to detect this type of excavation separately.

## 3 Related Work

Both object detection and action classification are problems of interest within domains like surveillance, automatic video classification and video retrieval.

The objective of object detection is to identify instances of certain predefined object classes in an image. Up until recently, solutions to this problem that produced state-of-the-art accuracy relied on machine learning methods like SVMs to produce their results [4]. However, recent developments in the field of deep learning have enabled more accurate object detection methods like Fast(er) R-CNN [9,20] and R-FCN [2], which use deep convolutional neural networks as the basis of their architecture. Even more recent architectures like You-Only-Look-Once [18], YOLOv2 [19] and the Single-Shot Detector [13] are only marginally less accurate then the current state-of-the-art, but their architectures are fast enough to provide inference in real-time. All of these architectures rely on a convolutional base network to provide the first network layers: popular choices are VGG-16 [22], Resnet-101 [26], Inception v3 [24] and Inception Resnet v2 [23].

The objective of action classification is to determine which kind of predefined action is undertaken in a series of images. A distinction can be made between approaches which feed hand-crafted features into a trainable classifier [5,11], and approaches which use a trainable feature extractor [8,25] to select the most useful features for classification [21]. Furthermore, the type of action one tries to classify and the environment in which one tries to accomplish this are also major factors in the success of the undertaken approach [21]. On datasets of less controlled environments like Hollywood2 [21], trainable feature extractors (CNNs [12]) have started to outperform hand-crafted features [6]. Recent state-of-the-art CNN-based approaches feed their output through SVMs [8] or computationally more expensive architectures like LSTMs [25] to produce a classification, but

these studies were based on readily available data, not developed and tested from scratch in a real world field setting as in our work.

## 4 Approach

We combine the ability of a CNN-based approach to extract features from uncontrolled environments with the information density hand-crafted features can provide. Our approach consists of two models: an object detector and an action classifier. The object detector predicts which pretrained objects are are present in the video frame, along with a location (bounding box) and a confidence value, given a JPEG video frame as input. This bounding box data is used directly to detect manual excavation, and is also fed into the action classifier, which predicts if mechanical excavation is taking place in a given sequence of bounding box data. The output from both models can be thresholded to find the right balance between sensitivity and specificity.

### 4.1 Data Collection and Preparation

We have constructed a dataset of video footage of excavation activities, from which we have extracted training and test data for both our object detector and action classifier. Most videos are shot by the project team on different days and time of days, in different locations, with different equipment, from different angles and by various people. We were constrained by the fact that self-captured data is not necessarily as heterogeneous as one would like it to be. One way to prevent overfitting [1] on domain-specific variables is to augment the dataset with videos from external sources, which we have done for the object detector ground truth. The videos were also randomly split between a training set (80% of videos) and a test set (20% of videos).

Both actions and objects have to be labeled. In our case, the process of gathering and labeling real-world data has proven to be very time consuming. On average, a person could label around 300 images per hour for the object detector, or 20 min of video for the action classifier. In our case, this has resulted in around 50 h of continuous labeling for both models combined, on top of the time it took to capture the footage.

The ground truth for the object detector consists of the set of bounding boxes of all predefined objects within a set of JPEG-frames; we have targeted objects that support both manual and mechanical excavation detection. For manual excavation detection, we detect workers who are either crouching or have an earth rod in their hands. For mechanical excavation detection, we detect various parts of the excavator, the underlying assumption being that the positions and movements of excavator parts are good indicators of said activities. We have defined eight predefined objects: “cabin” (the part of an excavator that contains the driver), “upper arm” (the arm section of an excavator directly connected to the cabin), “forearm” (the arm section of an excavator that can be connected to an attachment), “wheelbase” (the caterpillar wheels of an excavator), “bucket”

**Table 1.** Overview of ground truth data for the object detector.

| | Total | Filtered | Training | Test |
|---|---|---|---|---|
| Videos | 230 | n/a | 184 | 46 |
| Frames | 8,629 | 6,251 | 4,939 | 1,312 |
| cabin | 6,436 | 4,579 | 3,563 | 1,016 |
| upper arm | 6,182 | 4,373 | 3,368 | 1,005 |
| forearm | 6,190 | 4,368 | 3,407 | 961 |
| wheelbase | 6,603 | 4,750 | 3,708 | 1,042 |
| bucket | 5,163 | 3,512 | 2,860 | 652 |
| breaker | 1,284 | 1,029 | 688 | 341 |
| crouching | 1,751 | 1,324 | 1,109 | 215 |
| earth rod | 1,698 | 1,267 | 1,026 | 241 |

**Table 2.** Overview of ground truth data for the action classifier for various window sizes.

| | Total | Training | Test |
|---|---|---|---|
| Videos | 117 | 95 | 22 |
| Frames: exc. | 9,703 | 7,977 | 1,726 |
| Frames: no exc. | 7,574 | 5,892 | 1,682 |
| Window 3: exc. | 3,218 | 2,432 | 786 |
| Window 3: no exc. | 2,488 | 2,003 | 485 |
| Window 5: exc. | 1,929 | 1,595 | 334 |
| Window 5: no exc. | 1,475 | 1,185 | 290 |
| Window 7: exc. | 1,388 | 1,142 | 246 |
| Window 7: no exc. | 1,010 | 821 | 189 |
| Window 9: exc. | 1,072 | 818 | 254 |
| Window 9: no exc. | 774 | 618 | 156 |
| Window 11: exc. | 890 | 726 | 164 |
| Window 11: no exc. | 607 | 509 | 98 |

(the scooping attachment of an excavator), "breaker" (the drilling attachment of an excavator), "crouching" (a worker who is crouching to excavate) and "earth rod" (a worker who is excavating with an earth rod).

Some videos were captured by a camera man walking around the construction site, thus enlarging the intra video variety by capturing the same scene from different viewpoints. From 230 recorded videos, we have extracted frames at a rate of one frame per two seconds, which resulted in a pool of 8,629 frames. All of the occurrences of the eight objects listed above were then manually labeled by seven different people, although each frame was labeled by one person only. The generated ground truth was then filtered: first, frames that were very similar to the previous video frame (when all of the labeled bounding boxes overlap for at least 80% with a bounding box of the same object in the previous frame) were removed. Besides that, we have limited the number of extracted frames per video to 50, and took a random subset when this threshold was exceeded. Also, we manually removed frames that were heavily distorted by video artefacts such as ghosting or synchronization jitter [17]. Table 1 provides an overview of the ground truth data used for training and testing our object detector.

The ground truth for the action classifier consists of a classification ("mechanical excavation", "no mechanical excavation" or "unusable") for each second of video, and was generated from 117 videos of variable length (between 16 and 753 s each, with a total running time of 6 h and 35 min) but filmed from a fixed point-of-view, which is relevant because the temporal aspect of actions is combined with the assumption that cameras for the production system are always

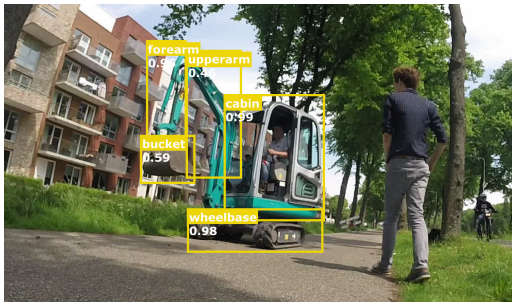


**Fig. 1.** Sample output from the object detector model, detections with highest confidence of each object are shown.

mounted on tripods. Besides this classification, each second of video is associated with a list of detections from an extracted video frame, as provided by the object detector. The videos were then split into non-overlapping windows of a length of an odd number of frames, and sequences which contained at least one "unusable" classification were removed from the window creation process. The remaining windows got assigned a target classification based on a majority vote over the contained frames. Finally, we have balanced the number of "excavation" and "no excavation" samples to be exactly the same, by randomly discarding some of the "excavation" windows. Table 2 provides an overview of the ground truth data used for training and testing our action classifier.

### 4.2 Design of the Object Detector

We select the SSD-512 architecture for the object detector, mainly for its speed [13]: our solution needs to process frames of four different cameras, and it is not the only software that should run on the production system: we also have to reserve resources for the action classifier, an orchestration service and some rule-based logic. The SSD-architecture is based on the traditional feed-forward Convolutional Neural Network [12], where the first layers of the network are initialized from a pretrained base network (in our case VGG-16 [23]): the benefit of this being that such a base network is able to extract higher-level features from images, which reduces training time. After this base network, five additional sets of feature map layers are defined, which are all implemented as a convolutional layer and responsible for object detections on a different scale. Finally, all detections of feature maps are concatenated in the network output layer, from which all detections with a confidence lower than 1% are filtered out. After that, the non-maximum suppression algorithm [15] is applied in order to merge different overlapping detections of the same object into one. Finally, for each processed frame we store the 200 detections with the highest associated confidence. See Fig. 1 for an example of the object detector applied to a public domain image.

Our training function is taken from the original SSD-512 implementation [13], which in turn is based on minimizing the MultiBox loss function [3], although

this loss function is adapted slightly to handle detections of multiple different classes [20].

We also deploy a method to augment our training set, analog to the original SSD-512 implementation [13]. First of all, a sub sample is selected from the original image, out of one of the following three options: the original input image, a sub sample covering at least 10%, 30%, 50%, 70% or 90% of the original input image, or a sub sample of random size. Now, on this sub sample, three translations are applied: a 50% chance of a horizontal flip, a 50% chance of the image canvas being randomly expanded with a maximum of 400%, and a 50% chance of hue, saturation, brightness and contrast adjustments, respectively. Furthermore, the balance between positive and negative examples can be significantly unbalanced in favor of the negative ones. Therefore, we pick at most three times as many negative as positive examples, selecting the ones with the highest associated confidence loss, as per the original SSD-512 implementation [13].

### 4.3 Design of the Action Classifier

The action classifier is mainly based on AdaBoost [7], which combines a boosting algorithm with a multitude of decision trees to arrive at a prediction. We use 500 estimators and the SAMME.R real boosting algorithm [27].

In order to make the action classifier as robust to real-life variety as possible, we have experimented with generating 10 permutations for each training set sample, on which a number of translations are applied. Analogue to the object detector, permutations have a 50% chance of getting flipped horizontally. Furthermore, all detections are scaled to a random size between 70% and 130%, keeping the aspect ration intact and preventing detections to move outside image bounds. Finally, all detections are moved to a random place on the image canvas, again keeping the distance between all detections intact.

Input features are based on object detections, which we limit to the strongest detection of each of the six mechanical excavator parts per frame, if any. All of the input features are normalized between 1 and −1. For each of the six parts, we first distinguish five base features, namely the x- and y-coordinate of the center of the predicted bounding box, the bounding box width and height and the confidence of the prediction. Because this representation of the features is not necessarily the most useful representation for discerning excavation actions from non-excavation actions, we augment these features with a set of engineered features, which are all different representations of these base features. In order to express movement of parts, we define the difference between each set of consecutive frames for each of the five attributes for each of the six excavator parts. Also, we calculate a motility score for each object over the whole window, based on the relative movements of the center point of the corresponding bounding box. Besides that, we define relative arm motility, representing the motility of both arm parts compared to the motility of the cabin and wheelbase objects in order to detect rotation. Besides input features related to movement, we also define features related to distance. For each object except for the cabin object in each frame,

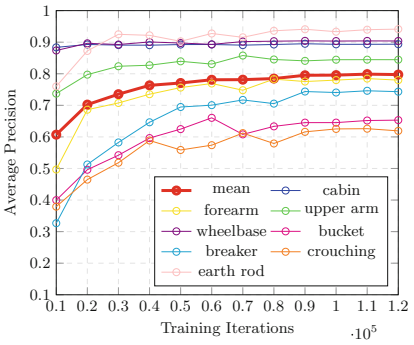


**Fig. 2.** Accuracy of the object detector in AP after training for a certain number of iterations. The thick red line denotes the mean accuracy over all the classes. After 90,000 iterations the mAP reaches 0.80.

**Table 3.** Average Precision (AP) accuracy on the test set of excavator pictures from various websites.

| Class | Average Precision | Samples |
|---|---|---|
| Cabin | 0.99 | 49 |
| Forearm | 0.92 | 49 |
| Upper Arm | 0.89 | 49 |
| Wheelbase | 0.99 | 49 |
| Bucket | 0.83 | 41 |

we supply the horizontal, vertical and Pythagorean distance between that object and the cabin. In order to determine which type of excavator is pictured, we also supply the difference in confidence between the two types of attachments in each frame, along with the difference in horizontal, vertical and Pythagorean distance between both attachments and the forearm object. Finally, we define features related to object size. We calculate the total width, height and surface area of each object over all frames, the relative size of arm boxes compared to cabin boxes as an indicator for the camera angle and the cumulative horizontal, vertical and Pythagorean change in size of each object over all frames as an additional indicator of rotation.

## 5 Results

Our experiments are performed on the object detector and action classifier separately.

### 5.1 Object Detector

The accuracy of the object detector over a number of training iterations is displayed in Fig. 2. Each iteration, a batch of 8 training examples is passed both forwards and backwards. We have used a SGD-solver combined with a multistep learning rate policy and an initial learning rate of $1 \times 10^{-6}$ [13]. Compared to the other objects, the accuracy of the cabin and wheelbase object detection does not improve as much with more iterations. A likely reason is that the base network VGG-16 is trained on an object that is visually similar to these objects: a car [22]. Furthermore, we have observed that the model incorrectly classifies certain

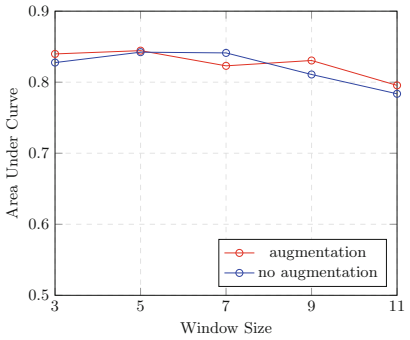


**Fig. 3.** The influence of data augmentation with AdaBoost on the AUC-score of the action classifier.

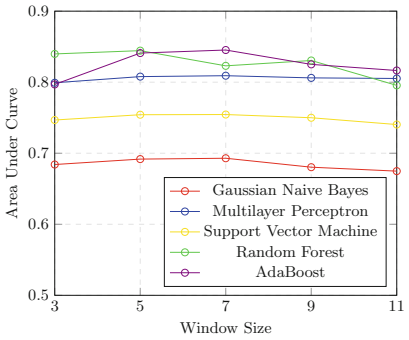


**Fig. 4.** The influence of window size and classifier choice on the AUC-score of the action classifier.

**Table 4.** Action classifier confusion matrix. Ground truth in rows, predictions in columns.

| | Excavation | No excavation |
|---|---|---|
| Excavation | 224 | 57 |
| No excavation | 48 | 369 |

**Table 5.** Action classifier confusion matrix (1 frame-per-window version).

| | Excavation | No excavation |
|---|---|---|
| Excavation | 1,017 | 451 |
| No excavation | 391 | 1,703 |

patches of dirt as a bucket, probably because the color is similar and there is often dirt attached to the bucket attachment in our training set.

We have also constructed a small dataset consisting of excavator images crawled from the web in order to determine if our object detector could also be useful outside the boundaries of our testing environment. This dataset was manually labeled using the same approach as we used for constructing our train and test sets. The results of running the model on this test set are given in Table 3. We can see that our model was able to generalize well to these different types of excavators.

### 5.2 Action Classifier

The optimal parameters to fit our use case are determined experimentally, two of which are window size and classifier type. We have established the influence of these parameters by training 25 different models: 5 different classifiers combined with 5 different window sizes. After training these 25 models, we compare them on the basis of their AUC-score [10]. The results of these experiments are plotted in Fig. 4. Top performers are AdaBoost on a window size of 5/7 and Random Forest on a window size of 3/5, resulting in an AUC-score of 0.84-0.85.

We have also tested our data augmentation routine on a variety of window sizes. The results of this experiment can be found in Fig. 3. Overall, data augmentation does not improve the accuracy of the action classifier, indicating that either the amount of ground truth data we feed to the model initially is already

**Table 6.** The 10 most important features, ranked on information gain.

| | |
|---|---|
| Attachment closest to forearm | 1.6% |
| Motility of cabin | 1.4% |
| Motility of upper arm | 1.2% |
| Cumulative horizontal distance between cabin and upper arm | 1.2% |
| Difference in vertical location of bucket, frame 1–2 | 1.2% |
| Difference in vertical location of bucket, frame 3–4 | 1.2% |
| Difference in vertical location of upper arm, frame 3–4 | 1.2% |
| Horizontal location of forearm, frame 5 | 1.2% |
| Difference in horizontal location of cabin, frame 3–4 | 1.0% |
| Difference in vertical location of upper arm, frame 2–3 | 1.0% |

sufficient, or that the variety the data augmentation is providing is not meaningful.

Eventually we have implemented an AdaBoost classifier and a window size of 5 in our production model, with data augmentation turned off. See Table 4 for the confusion matrix belonging to this action classifier. The corresponding AUC-score is 0.84. 85% of the samples are classified correctly.

A benefit of our action classifier is that we are able to take not only shape information, but also motion information into account. In order to prove the usefulness of this approach, we have constructed a variant of our action classifier where windows consist of one frame only, thus eliminating all motion information from training data. As can be derived from Table 5, such a model classifies 76% of the samples correctly. This is 9% less then the accuracy our 5 frames-per-window version achieves on the same data set. The corresponding AUC-score decreased to 0.75.

In order to further understand the characteristics of mechanical excavation, we also determine the features the model considers most important to make a distinction between excavating and non-excavating actions on the basis of information gain. The 10 features associated with the highest information gain are listed in Table 6. With the exception of the horizontal location of the forearm and the confidence of the cabin detection, all of the 15 most important features are engineered features, highlighting the importance of feature engineering.

## 6 Discussion

Our two stage approach provides a good balance between more assumption free, data driven learning at a lower level and using domain knowledge at a higher level, for improved results and better control. In the first object detection step we leave the task of figuring out the best features to the convolutional net, though we already guide it to learn the right concepts by specifying specific object classes. The subsequent action classification step then allows for a lot of

flexibility to engineer use case specific features based on detected objects, and the experiments have demonstrated that these are most predictive. The experiments confirmed as well that using sequences as input, i.e. more than one frame, gave superior results, providing further support for a sequence based approach. From an application perspective, the production pilot lived up to the expectations of the construction company, and the intent is to keep using the system for a prolonged period of time. Also, a range of other companies have shown interest in the pilot, further demonstrating the relevancy of this problem.

In terms of future work, there are various methods in which our approach could be improved further. First of all, one could incorporate object tracking into the object detector model, to provide more reliable detection results to the action classifier [14]. Also, pose estimation could be used in order to differentiate various angles of the same action, as the mechanical excavation action looks very different viewed from different angles [16]. Another possible improvement is the incorporation of online learning methods, as we have already implemented collection methods for incorrectly classified alerts.

## 7 Conclusion

We have demonstrated an approach to real time action classification based on object detection, under difficult real world conditions and with limited hardware. We have applied this approach to the practical problem of detecting unauthorized excavation activities on construction sites. Our system is capable of classifying actions in real-time on a laptop workstation: we are able to analyze the output of four different cameras simultaneously without performance issues. To best balance assumption free learning with application of problem domain knowledge, we use a neural network based object detector to extract relevant shape and motion features, and then use these features as well as problem specific, engineered features derived from this as input for an action classifier. A major benefit of this approach is that it is insensitive to stray objects and movements, and thus is able to function in uncontrolled environments. A second benefit is that we can use localization information originating from the object detector to localize actions within a video frame.

## References

1. Babyak, M.A.: What you see may not be what you get: a brief, nontechnical introduction to overfitting in regression-type models. Psychosom. Med. **66**(3), 411–421 (2004)
2. Dai, J., Li, Y., He, K., Sun, J.: R-FCN: Object detection via region-based fully convolutional networks. Advances in Neural Information Processing Systems, pp. 379–387 (2016). http://arxiv.org/abs/1605.06409
3. Erhan, D., Szegedy, C., Toshev, A., Anguelov, D.: Scalable object detection using deep neural networks. In: 2014 IEEE Conference on Computer Vision and Pattern Recognition, pp. 2155–2162 (2014)

4. Felzenszwalb, P.F., Girshick, R.B., Mcallester, D., Ramanan, D.: Object detection with discriminatively trained part based models. IEEE Trans. Pattern Anal. Mach. Intell. **32**(9), 1–20 (2009)
5. Fernando, B., Gavves, E., José Oramas, M., Ghodrati, A., Tuytelaars, T.: Modeling video evolution for action recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 5378–5387 (2015)
6. Fernando, B., Gavves, E., Oramas, J., Ghodrati, A., Tuytelaars, T.: Rank pooling for action recognition. IEEE Trans. Pattern Anal. Mach. Intell. **PP**(99), 1–14 (2016)
7. Freund, Y., Schapire, R.E.: A decision-theoretic generalization of on-line learning and an application to boosting. In: Vitányi, P. (ed.) Computational Learning Theory, pp. 23–37. Springer, Berlin Heidelberg, Berlin, Heidelberg (1995)
8. Girdhar, R., Ramanan, D., Gupta, A., Sivic, J., Russell, B.: ActionVLAD: learning spatio-temporal aggregation for action classification. In: CVPR, vol. 2, p. 3 (2017)
9. Girshick, R.: Fast R-CNN. In: Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition, pp. 1440–1448 (2015)
10. Hanley, A., McNeil, J.: The meaning and use of the area under a receiver operating characteristic (ROC) Curve. Radiology **143**, 29–36 (1982)
11. Hoai, M., Zisserman, A.: Improving human action recognition using score distribution and ranking. In: Cremers, D., Reid, I., Saito, H., Yang, M.-H. (eds.) ACCV 2014. LNCS, vol. 9007, pp. 3–20. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-16814-2_1
12. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. Advances In Neural Information Processing Systems, pp. 1–9 (2012)
13. Liu, W., et al.: SSD: single shot multibox detector. In: European Conference on Computer Vision, pp. 21–37 (2016)
14. Milan, A., Leal-Taixe, L., Reid, I., Roth, S., Schindler, K.: MOT16: a benchmark for multi-object tracking. arXiv preprint arXiv:1603.00831, pp. 1–12 (2016). http://arxiv.org/abs/1603.00831
15. Neubeck, A., Van Gool, L.: Efficient non-maximum suppression. Proc. Int. Conf. Pattern Recognit. **3**, 850–855 (2006)
16. Poirson, P., Ammirato, P., Fu, C.Y., Liu, W., Kosecka, J., Berg, A.C.: Fast single shot detection and pose estimation. In: 2016 Fourth International Conference on 3D Vision (3DV), pp. 676–684
17. Punchihewa, A., Bailey, D.G.: Artefacts in image and video systems: classification and mitigation. In: Proceedings of Image and Vision Computing New Zealand, pp. 197–202 (2002)
18. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: unified, real-time object detection. CVPR **2016**, 779–788 (2016). https://doi.org/10.1016/j.nima.2015.05.028
19. Redmon, J., Farhadi, A.: YOLO9000: better, faster, stronger. arXiv preprint arXiv:1612.08242 (2016). http://arxiv.org/abs/1612.08242
20. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: towards real-time object detection with region proposal networks. In: NIPS, pp. 1–10 (2015)
21. Sargano, A., Angelov, P., Habib, Z.: A comprehensive review on handcrafted and learning-based action representation approaches for human activity recognition. Appl. Sci. **7**(1), 110 (2017)
22. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: International Conference on Learning Representations (ICRL), pp. 1–14 (2015)

23. Szegedy, C., Ioffe, S., Vanhoucke, V.: Inception-v4, Inception-ResNet and the impact of residual connections on learning. In: AAAI, pp. 4278–4284 (2017)
24. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. In: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2818–2826 (2016)
25. Torabi, A., Sigal, L.: Action classification and highlighting in videos. arXiv preprint arXiv:1708.09522 (2017)
26. Wu, S., Zhong, S., Liu, Y.: Deep residual learning for image steganalysis. Multimedia Tools and Applications, pp. 1–9 (2017)
27. Zhu, J., Rosset, S., Zou, H., Hastie, T.: Multi-class Adaboost. Ann. Arbor **1001**(48109), 1612 (2006)