Object tracking with deep neural networks

Santeri Salmijärvi

School of Electrical Engineering

Bachelor's thesis
Espoo Work in progress! Compiled: 09:17:45 2017/08/15

Thesis supervisor:

D.Sc. (Tech) Pekka Forsman

Thesis advisor:

M.Sc. (Tech) Mikko Vihlman

Aalto University
School of Electrical
Engineering

AALTO UNIVERSITY
SCHOOL OF ELECTRICAL ENGINEERING

ABSTRACT OF THE BACHELOR'S THESIS

Author: Santeri Salmijärvi

Title: Object tracking with deep neural networks

Date: Work in progress! Compiled: 09:17:45 2017/08/15 Language: English
Number of pages: 5+13

Degree programme: Bachelor's Program in Electrical Engineering

Supervisor: D.Sc. (Tech) Pekka Forsman

Advisor: M.Sc. (Tech) Mikko Vihlman

abstract in english

Keywords: keywords in english

AALTO-YLIOPISTO
SÄHKÖTEKNIIKAN KORKEAKOULU

KANDIDAATINTYÖN
TIIVISTELMÄ

Tekijä: Santeri Salmijärvi

Työn nimi: Kohteenseuranta syvillä neuroverkoilla

Päivämäärä: Work in progress! Compiled: 09:17:45 2017/08/15 Kieli: Englanti
Sivumäärä: 5+13

Koulutusohjelma: Sähkötekniikan kandidaattiohjelma

Vastuuopettaja: TkT Pekka Forsman

Työn ohjaaja: DI Mikko Vihlman

lyhyt tiivistelmä suomeksi

Avainsanat: avainsanat suomeksi

Contents

Abbreviations

CNN Convolutional Neural Network

DNN Deep Neural Network

FCN Fully Convolutional Network

ILSVRC ImageNet Large Scale Visual Recognition Challenge

LSTM Long Short Term Memory

MLP MultiLayer Perceptron

MOT Multiple Object Tracking challenge

NN Neural Network

ReLU Rectified Linear Unit

SDAE Stacked Denoising Autoencoder

VOT Visual Object Tracking challenge

1 Introduction

Object tracking is a large and actively researched sub-area of computer vision. The main task for a tracker is to find and follow the desired subject in a sequence of images. Object tracking is closely related to other image analysis tasks so the implementations also share elements. In the recent years, use of deep neural networks has been researched for object tracking.

Many of the deep networks tailored to tracking tasks are variations of convolutional networks. Another way used to extract features from a frame is a stacked denoising autoencoder [1]. The training of deep neural networks requires a large amount of training data and their development has been made easier by an increase in the size of applicaple datasets.

Comment by author:
add cites to examples

The goal of this thesis is to study the concepts behind object tracking and deep neural networks. It will present the architectures and principles currently used in deep neural networks tailored to object tracking tasks. The practices and datasets used in training and evaluating such networks are also introduced.

2 Deep Learning

Neural networks are heavily researched for numerous applications and they are loosely based on the way a brain functions. The basic model a Neural Network (NN) consist of inputs, outputs and a connecting layer of neurons. This chapter introduces the basic concepts behind general deep neural networks, convolutional neural networks and stacked denoising autoencoders. 2.1 and 2.2 are based on the book Deep Learning by Goodfellow et. al. [2].

2.1 Deep neural networks

A Deep Neural Network (DNN) is commonly defined as a NN that has a **visible** input and output layer with several **hidden layers** between them. The distinction between visible and hidden layers is important because training of the network only evaluates the output layer's performance. During training, a **learning algorithm** optimizes the individual hidden layers to best approximate the desired output of the whole network.

The input layer takes in the data to be processed, which typically means a vector of color values in the case of object tracking. These are then processed by the hidden layers and finally the output layer produces the target's position in the frame. These models usually come in the form of a **feedforward neural network** or **MultiLayer Perceptron (MLP)**. The name comes from the fact that information flows from the input through computations to the output with no **feedback** connections. Typically this means that connections are only between consecutive layers.

In NNs, each layer consist of several **units** with an activation function and a weight for each of their input connections. The weights of the layer's inputs are commonly represented by a matrix by which the input vector is multiplied as each row represents a unit's input weights. All of the units can be connected to all of the inputs (fig.1) forming a fully connected layer or just some of them (fig.2) utilizing sparse connections. Sparse layers can be implemented by defining unique input vectors for the units. Units in a layer have a common activation function that is fed by the sum of its weighted inputs. The **Rectified Linear Unit (ReLU)** is a commonly used unit type and is defined by the activation function $g(z) = \max\{0, z\}$. It provides a nonlinear transformation while being comparable to linear models in terms of generalizing well and being easy to optimize. A bias-term can also be defined for each unit and a vector containing the layer's biases is summed to the outputs of the activation function before passing the results to the next layer.

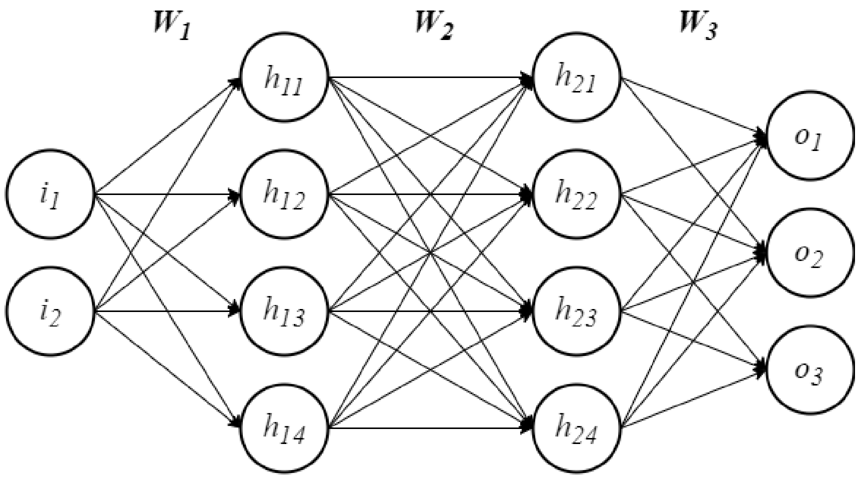


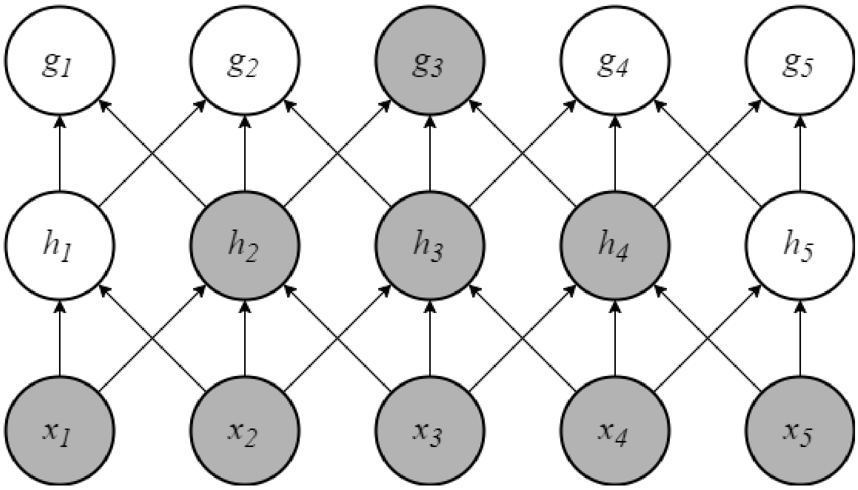
Figure 1: A fully connected network with two inputs i, two hidden layers of four units h and three outputs o. Each set of connections is represented by a weight matrix $\mathbf{W}$ which indicates mapping from one layer to another. Excluding the input, all layers also have an activation function and their units can be assigned individual weights.

Before training, the weights of a MLP are initialized to small random values and biases to zero or small positive values. Then an algorithm called **stochastic gradient descent** is commonly applied alongside a training dataset. The basic procedure is to calculate the error of the netwok's output values compared to the desired ones using a **loss function**. The function's gradient can then be calculated for example by **back-porpagation**, which feeds the errors back through the network to assign a contribution value to each unit. These values are then used to calculate the gradient of the loss function relative to the weights. Each weight is adjusted slightly to the opposite sign to minimize the loss function.

2.2 Convolutional neural networks

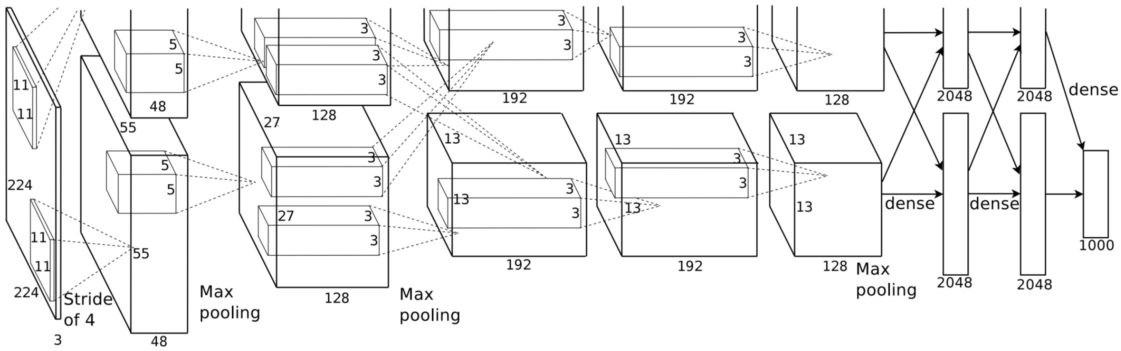
A Convolutional Neural Network (CNN) is simply a NN that uses convolution instead of general matrix multiplication in at least one of its layers. The main benefits of convolution in NNs are that it's dramatically more efficient in terms of memory requirements, it reduces the amount of computation needed and it makes it possible to work with variable input sizes.

A typical convolutional layer consists of three stages: a convolution stage, detector stage and pooling stage. These can be implemented by individual layers. First, a **kernel** is applied to the input data in positions separated by a stepsize. This means that a linear activation function is fed by the matrix product of the input location and the kernel's weight matrix. In the detector stage, the results are then run through a non-linear activation, for example a ReLU. Finally, a **pooling function** is used to combine the results of multiple nearby outputs as the final output.



Source: Recreated fig. 9.4 from Deep Learing [2]

Figure 2: Deeper layers in a convolutional network are connected to a larger window of the input data than shallow layers. This means that deeper layers can be indirectly connected to most or all of the input data even though their direct connections are sparse. [2]



Source: Krizhevsky et. al. page 5. [3]

Figure 3: The network used in [3] is a good example of a typical CNN architecture. The first layer evaluates 11×11×3 kernels of pixels from the input and feeds into layers of increasing depth with the final convolutional layer working on 3×3×192 kernels. The final layers are fully connected with 4096 neurons and produce the networks output as a vector of probabilities over the 1000 trained subject classes.

2.3 Stacked denoising autoencoders

Autoencoders consist of an encoder, a decoder and a loss function. They first encode the given data to a hidden representation and then reconstruct it while the loss function is used in training. Reconstructing the exact input data is not useful and denoising autoencoders avoid that by learning to encode a corrupted version of the input and decode the result into useful features of the clean input. A stacked denoising autoencoder utilizes a sequence of encoders followed by matching decoders trained this way. Corrupted input data is only used to train the individual layers to find useful features as a trained Stacked Denoising Autoencoder (SDAE) works on clean input. [4]

Comment by author:
motivation for using sdaes?

3 Object tracking

Object tracking in video sequences has been researched for decades using different approaches for defining the target and adapting to changes in its shape or orientation. The situations most likely to cause tracking failure have also been identified.

Comment by author:
Define the problem in more depth

3.1 Target representation

Early influential works in the field have used target models including subspaces [5] and representing the target as a curve [6]. Modern tracking methods can be roughly divided to generative and discriminative, but combinations of them have also been proposed.

Comment by author:
cite a combination

Generative methods search the frame for the best matches to a template of an appearance model of the subject. Template methods based on pixel intensity and color histograms perform well with no drastic changes in object appearance and non-cluttered backgrounds. Appearance models learned from training can be less affected by appearance variations and adaptive schemes provide added flexibility, while sparse models handle occlusion and image noise better. [7]

Comment by author:
add citations

Discriminative methods consider tracking as a binary classification problem. They take the background also into account to separate the target from it. Used approaches include refining the initial guess with a support vector machine [8] or utilizing a relevance vector machine [9].

During tracking, the appearance of the target may change for example due to changes in orientation. Some trackers adapt the tracking model online to be robust against such changes, but care must be taken in designing the update algorithm as it could result to drift.

Comment by author:
Give some examples, cite

3.2 Datasets

The datasets used for training are equally important as the actual network design

Comment by author:
citation

.

Research on networks working with image data has been made easier by larger sets of both hand-labeled sets and ones obtained by simple keyword searches from online image services. These kinds of sets can be used to pre-train useful target features to tracking networks.

VOC was a yearly competition for object recognition and VOC 2012 [10] is the last challenge in the series. The datasets of the challenges are still used for pre-training features for detection stages in tracking networks. There are four major subsets of hand-labeled VOC data: classification, segmentation action classification, person layout. Classification datasets consist of images annotated with the objects contained and bounding boxes for the objects drawn in the image itself while image segmentation sets provide additional mask images of the objects and classes in each shot. Action classification sets contain descriptions and bounding boxes of actions the subjects are performing and person layout sets contain bounding boxes for the subjects head, hands and feet.

Comment by author:
image examples of the datasets, descriptions to annotations

The ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [11] is another recognition challenge running since 2010. The most recent dataset consists of subsets of object localization, object detection and object detection from video. The last subset is especially beneficial object tracking tasks as it provides data for training on actual tracking data. The other two sets are also substantially larger than the respective VOC sets as their labeling has been crowd sourced.

There has also been an increase in resources solely devoted to tracking data with the TB-100 -set [12] being a good example. It contains a hundred tracking sequences with reference positions for the target on each frame. Because some of the targets are similar or less challenging, a subset of 50 sequences considered challenging is also provided as TB-50. [13]

The datasets used for the Visual Object Tracking challenge (VOT) [14] can also be used for training networks. The competition is run yearly with updated evaluation sets which can be used for training as training data for a network but the challenge itself prohibits training on tracking datasets for participants.

There is also the yearly Multiple Object Tracking challenge (MOT) [15] for testing multiple object trackers but its unique sequences can also be used to train single object trackers one object at a time.

3.3 Evaluation

Evaluation of proposed trackers is a vital part of the research. It also limits the use of annotated tracking sequences in training as training and evaluation should be done on different data.

Comment by author:
present the metrics used in evaluation

The Visual Tracker Benchmark [12] is a commonly used resource for comparing performance to other trackers. It consists of the TB-datasets, a code library containing implementations of 31 publicly available trackers and ready benchmark results for the included trackers. The code library is implemented using MATLAB and all included trackers have been modified to use unified input and output formats. A Python-based testing suite is also in development. The original benchmark was compiled in 2015 so doesn't include more recent trackers in the suite. The VOT [14] and MOT [15] challenges also publish both the yearly challenge suite and results, which can used to compare new networks against the participants.

4 Deep neural networks in tracking

This chapter presents some effective trackers using DNNs.

4.1 Trackers using convolutional neural networks

CNNs are predominant as feature extractors and an early implementation of a CNN-based tracker [16] pre-dates the work of Krizhevsky et. al. [3]. It takes modifies the architecture used for detection to make the network less affected by sifts in the objects position in the frame. Shift-invariancy is a non-desirable quality in tracking while using previous positions as a as it might result to mixups with objects similar to the target [16]

[17] Learns generic features from video sequences with tracked objects and focuses on features robust to complex motion patterns. An adaptation module integrates the target's shape and texture to the pre-trained features. This is done using the first 20 frames of the online tracking sequence as training data.

[18] Learns branching and domain specific fully connected layers and the online tracking substitutes them with a single new layer that is fine-tuned online with the shared layers. Both long- and short-term updates are utilized to provide both robust and adaptive tracking.

[19] No pre-training as the network learns features online. Model is updated if training loss is above a certain threshold.

[20] Combines a pre-trained feature descriptor CNN and a SVM that creates a saliency map from the extracted features. This map is used as a filter to extract the position of the target in each frame.

[21] Applies a siamese architecture of two convolutional networks to object tracking. A candidate image is compared to an exempla image and is scored based on their similarities.

[22] Combines an efficient feature extractor (YOLO) to spatial and temporal constraints. The network's layers are first pre-trained with a traditional CNN for general feature learning. YOLO is then adopted as the detection module and the Long Short Term Memory (LSTM) is added before training it as part of the whole network. The LSTM is provide robust access to long-range context and is fed with the output of the detection stage converted to a 32x32 heatmap linked with the learned visual features.

[23] Uses the conv4-3 and conv5-3 layers of the VGG network for selecting feature maps that are fed to two different networks: a general network to capture category information and a specific network to discriminate the target from the background. Both networks output heatmaps which are used for final detection. The general network's result is used by default while the specific network is used to determine the target location if a distracter is detected in the background. Both networks are initialized on the first frame, but only the specific network is updated online to avoid noise.

[24] Instead of using just the final output of a sequence of convolutional layers, the proposed algorithm uses multiple layers to find the target's position. This is done

by going through the outputs coarse-to-fine to regularize the search for the maximum value in the finer response maps. All the layers' correlation filter numerator and denominator are also updated each frame to get a robust and computationally lighter approximation of minimizing output error.

[25] Views the traditionally fully connected layers at the end of a CNN based tracking network as convolutional layers and uses upscaling with skip connections to previous layers. This Fully Convolutional Network (FCN) is computationally lighter than a sliding window based network as it only requires a single feedforward [connection?]. The network was pre-trained on the VOC2012 dataset to learn features for targets of 20 categories in the dataset. The tracker is only able to detect objects in those categories. It also only allows single object tracking but the target can be identified in the first probability map if the sequence contains multiple targets to permit multi-object scenarios and increase accuracy in single-object tracking. (However, the method is currently not efficient enough for tracking in real time.)

4.2 Other approaches

[26] Uses a SDAE fine-tuned with SURF features gotten from matching the current frame's to the first one's.

[27] Tackles the issue of motion blur as it is common in actual applications of object tracking. Deblurring the images online is not computationally viable so the work proposes a blur-invariant object tracker. It uses a deep hierarchial appearance model pre-trained with unlabeled data that is blurred with varying kernel sizes to make the model more robust.

5 Conclusions

Summarize the current state of object tracking with DNNs with possibly some insight to future developments.

References

[1] N. Wang and D.-Y. Yeung. Learning a deep compact image representation for visual tracking. In *Advances in Neural Information Processing Systems 26*, pages 809–817. Curran Associates, Inc., 2013. ISBN 9781632660244.

[2] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, volume 2, pages 1097–1105, 2012. URL http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

[4] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11:3371–3408, 2010. URL http://www.jmlr.org/papers/v11/vincent10a.html.

[5] M. J. Black and A. D. Jepson. Eigentracking: Robust matching and tracking of articulated objects using a view-based representation. *International Journal of Computer Vision*, 26(1):63–84, 1998. doi: 10.1023/A:1007939232436.

[6] M. Isard and A. Blake. Condensation - conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29(1):5–28, 1998. doi: 10.1023/A:1008078328650.

[7] Q. Wang, F. Chen, W. Xu, and M. H Yang. Object tracking via partial least squares analysis. *IEEE Transactions on Image Processing*, 21(10):4454–4465, 2012. doi: 10.1109/TIP.2012.2205700.

[8] S. Avidan. Support vector tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(8):1064–1072, 2004. doi: 10.1109/TPAMI.2004.53.

[9] O. Williams, A. Blake, and R. Cipolla. Sparse bayesian learning for efficient visual tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(8):1292–1304, 2005. doi: 10.1109/TPAMI.2005.167.

[10] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes challenge 2012 (voc2012) results. URL http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[11] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

[12] Visual tracker benchmark. URL http://cvlab.hanyang.ac.kr/tracker_benchmark/index.html. Accessed: 2017-06-25.

[13] Y. Wu, J. Lim, and M. H Yang. Object tracking benchmark. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9):1834–1848, 2015. doi: 10.1109/TPAMI.2014.2388226.

[14] Matej Kristan, Jiri Matas, Aleš Leonardis, Tomas Vojir, Roman Pflugfelder, Gustavo Fernandez, Georg Nebehay, Fatih Porikli, and Luka Čehovin. A novel performance evaluation methodology for single-target trackers, Jan 2016. URL http://arxiv.org/abs/1503.01313.

[15] A. Milan, L. Leal-Taixé, I. Reid, S. Roth, and K. Schindler. MOT16: A benchmark for multi-object tracking. *arXiv:1603.00831 [cs]*, March 2016. URL http://arxiv.org/abs/1603.00831. arXiv: 1603.00831.

[16] J. Fan, W. Xu, Y. Wu, and Y. Gong. Human tracking using convolutional neural networks. *IEEE Transactions on Neural Networks*, 21(10):1610–1623, 2010. doi: 10.1109/TNN.2010.2066286.

[17] L. Wang, T. Liu, G. Wang, K. L. Chan, and Q. Yang. Video tracking using learned hierarchical features. *IEEE Transactions on Image Processing*, 24(4): 1424–1435, 2015. doi: 10.1109/TIP.2015.2403231.

[18] H. Nam and B. Han. Learning multi-domain convolutional neural networks for visual tracking. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2016-January, pages 4293–4302, 2016.

[19] H. Li, Y. Li, and F. Porikli. Deeptrack: Learning discriminative feature representations online for robust visual tracking. *IEEE Transactions on Image Processing*, 25(4):1834–1848, 2016. doi: 10.1109/TIP.2015.2510583.

[20] S. Hong, T. You, S. Kwak, and B. Han. Online tracking by learning discriminative saliency map with convolutional neural network. In *32nd International Conference on Machine Learning, ICML 2015*, volume 1, pages 597–606, 2015.

[21] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. S. Torr. Fully-convolutional siamese networks for object tracking. 9914 LNCS:850–865, 2016. doi: 10.1007/978-3-319-48881-3_56.

[22] Guanghan Ning, Zhi Zhang, Chen Huang, Zhihai He, Xiaobo Ren, and Haohong Wang. Spatially supervised recurrent convolutional neural networks for visual object tracking. *CoRR*, abs/1607.05781, 2016. URL http://arxiv.org/abs/1607.05781.

[23] L. Wang, W. Ouyang, X. Wang, and H. Lu. Visual tracking with fully convolutional networks. In *Proceedings of the IEEE International Conference on*

Computer Vision, volume 11-18-December-2015, pages 3119–3127, 2016. doi: 10.1109/ICCV.2015.357.

[24] C. Ma, J. B Huang, X. Yang, and M. H Yang. Hierarchical convolutional features for visual tracking. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 11-18-December-2015, pages 3074–3082, 2016. doi: 10.1109/ICCV.2015.352.

[25] J. Lee, B. K. Iwana, S. Ide, and S. Uchida. Globally optimal object tracking with fully convolutional networks. *CoRR*, abs/1612.08274, 2016. URL http://arxiv.org/abs/1612.08274.

[26] N. Ren, J. Du, S. Zhu, L. Li, D. Fan, and J. M. Lee. Robust visual tracking based on scale invariance and deep learning. *Frontiers of Computer Science*, 11 (2):230–242, 2017. doi: 10.1007/s11704-016-6050-0.

[27] J. Ding, Y. Huang, W. Liu, and K. Huang. Severely blurred object tracking by learning deep image representations. *IEEE Transactions on Circuits and Systems for Video Technology*, 26(2):319–331, 2016. doi: 10.1109/TCSVT.2015.2406231.