Object tracking with deep neural networks

Santeri Salmijärvi

School of Electrical Engineering

Bachelor's thesis
Espoo Work in progress! Compiled: 23:41:05 2017/08/20

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: 23:41:05 2017/08/20 Language: English
Number of pages: 5+18

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

Tekijä: Santeri Salmijärvi

Työn nimi: Kohteenseuranta syvillä neuroverkoilla

Päivämäärä: Work in progress! Compiled: 23:41:05 2017/08/20 Kieli: Englanti
Sivumäärä: 5+18

Koulutusohjelma: Sähkötekniikan kandidaattiohjelma

Vastuuopettaja: TkT Pekka Forsman

Työn ohjaaja: DI Mikko Vihlman

lyhyt tiivistelmä suomeksi

Avainsanat: avainsanat suomeksi

Contents

Abbreviations

CNN Convolutional Neural Network

DLT Deep Learning Tracker

DNN Deep Neural Network

ILSVRC ImageNet Large Scale Visual Recognition Challenge

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 indicate the desired subject in a sequence of images. It can be applied in areas such as human-computer interaction and augmented reality and is related to other image analysis tasks. In the recent years, use of deep neural networks has been researched for object tracking following their adoption in image classification.

Tracking implementations with deep neural networks are investigated because traditional hand-crafted feature models can suffer from drift or tracking failure in challenging circumstances like cluttered backgrounds or large changes illumination. Another issue that hand-crafted features don't generalize well to new target classes, which is one of the observed strengths of the hierarchical features a deep learning model acquires from training. Deep networks are also becoming more viable in real-time tracking as the computational power that is available increases and more efficient specializations of deep feature models are developed for the task. 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 applicable datasets.

This thesis explores object trackers that have used deep neural networks in their architecture. First, deep neural networks and object tracking are familiarized in enough detail to understand the reasoning and designs used for the trackers utilizing deep learning. Datasets for training and evaluating trackers are also briefly discussed as they are important to the research. The main research questions are the following. What kinds of tracker architectures have used deep neural networks? How deep features benefit the trackers in question? What kinds of drawbacks emerge from their use? The work will be done purely as a literature study.

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. The use and complexity of neural networks have greatly increased in the recent years as the massively parallel architecture of GPUs has been used to gain significant increases in speed compared to what is possible on CPU based implementations [1]. 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 consists 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 units can be connected to all 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.

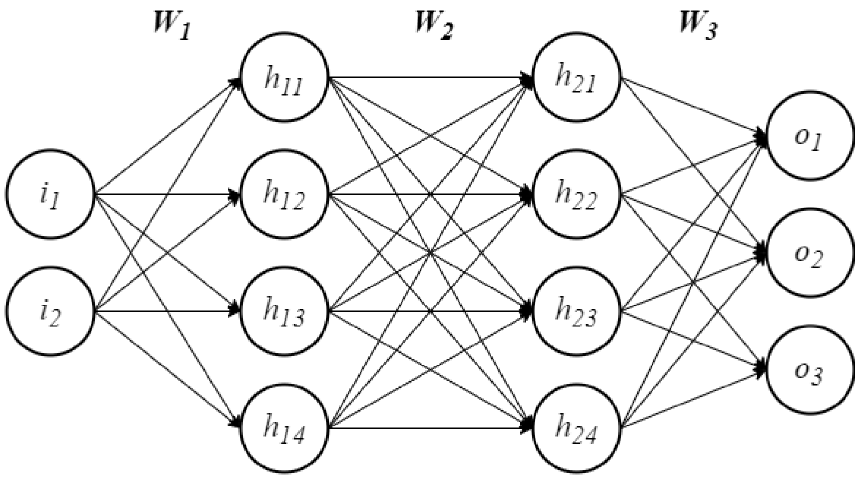

Figure 1: A fully connected network with two inputs i, two hidden layers of four units h and three outputs o. Each set of connections is represented by a weight matrix $\mathbf{W}$ which indicates mapping from one layer to another. Excluding the input, all layers also have an activation function and 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 network's output values compared to the desired ones using a loss function. The function's gradient can then be calculated for example by back-propagation, 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

"Convolutional networks are simply neural networks that use convolution in place of general matrix multiplication in at least one of their layers" [2]. Intuitively, convolution can be viewed as a blending of two functions as it is an integral expressing the overlap of two functions as one is moved over the other. A convolution performed on integers is by definition an infinite summation but it can be implemented on a finite number of elements if the functions are considered zero for all values that are not stored. In a Convolutional Neural Network (CNN), a convolution defined by this property is performed on the input data and a kernel in the form of a vector of weights learned in training.

A typical convolutional layer consists of three stages: a convolution stage, detector stage and pooling stage. These operations can be implemented by individual layers. First, the multiplication of the kernel and the input data is performed in positions separated by a step size. The result is then fed to a linear activation function. 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 activations as the final output.

The main motivations in using CNNs are sparse interactions, parameter sharing and equivariant interactions. Units in traditional layers are connected to the whole input so an input sized m and output of size n form a computational complexity of $m \times n$. Convolutional layers' units typically only connect to a small portion of the input, which can be a significant decrease in computation: a kernel of size k results in a complexity $k \times n$ and k can be kept several orders of magnitude smaller than m. It is also possible to share the same kernel for all positions in the input to reduce the number of weights stored from $m \times n$ to just k. Parameter sharing in convolution also results in equivariance to translation. It is a useful property in processing 2D data as a shift in the input results in a similar shift in the output. Equivariance to some other transformations is not inherent to convolution so other mechanisms are required for handling them.

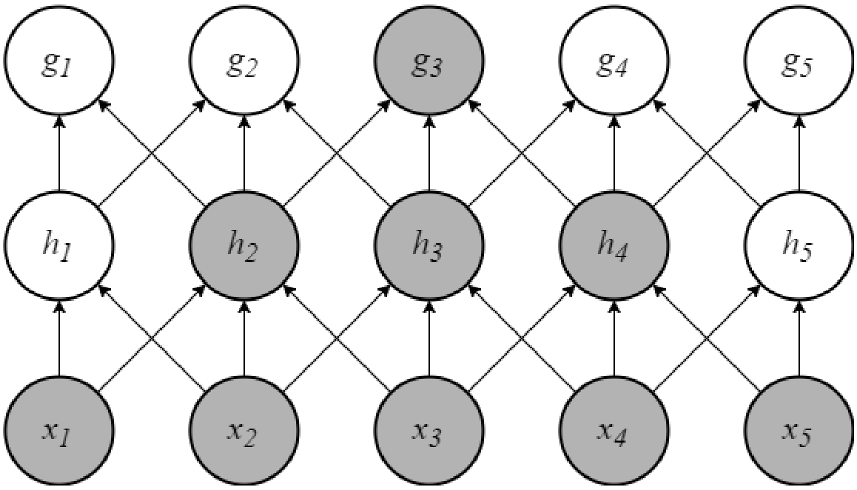

Figure 2: Stacking convolutions can provide deeper layers indirect connections to most or all of the input data even though their direct connections are sparse. This forms hierarchies of features that are useful for capturing larger concepts and the effect increases if a strided convolution or pooling is used. [2] Source: Recreated fig. 9.4 from Deep Learing [2]

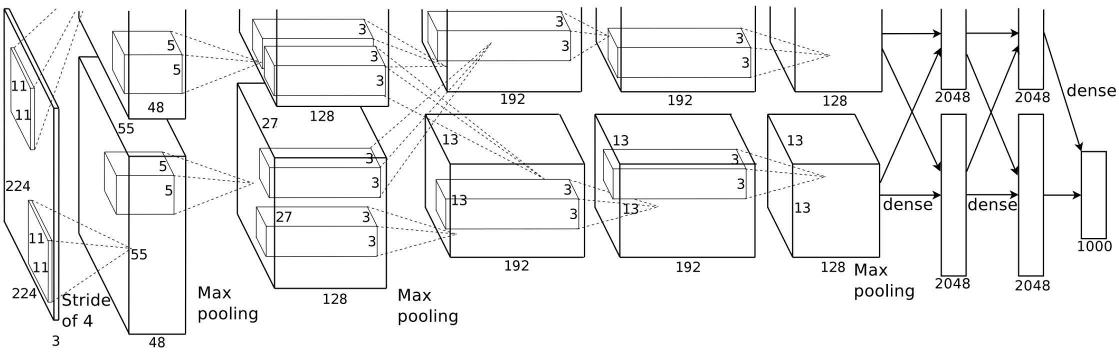

Figure 3: A branch in the network of Krizhevsky et. al. [1] is a good example of a basic CNN architecture. The first layer uses a 11×11×55 kernel and feeds into layers of increasing depth with the final convolutional layer working with a 3×3×192 kernel. The final layers are fully connected and produce the networks output as a vector of probabilities over the 1000 trained subject classes. Source: Krizhevsky et. al. page 5 [1]

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. The loss function is used in training to guide the result towards the desired output. 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 each encoding the input further with the final encoder feeding a series of matching decoders. Corrupted input data is only used in training the autoencoder to find useful features as a trained Stacked Denoising Autoencoder (SDAE) works on clean input. [3] The encoder halves of SDAEs were used for extracting features from tracking sequences especially before research was done on shift-variant CNNs.

3 Object tracking

Object tracking in video sequences has been researched for decades. The task of a tracker is to follow a target through a video sequence with the target's location initial location commonly indicated in the first frame. It is important to make the distinction between object tracking and object detection like the face detection that's available for many smartphones and cameras. Detection searches for objects matching a set class of objects from the image, but a tracker must keep track of an individual object based on its texture and other unique characteristics. In this chapter, the common target representations used in tracking are discussed along with some datasets typically used for training or evaluation. Methodology for evaluating trackers is also introduced.

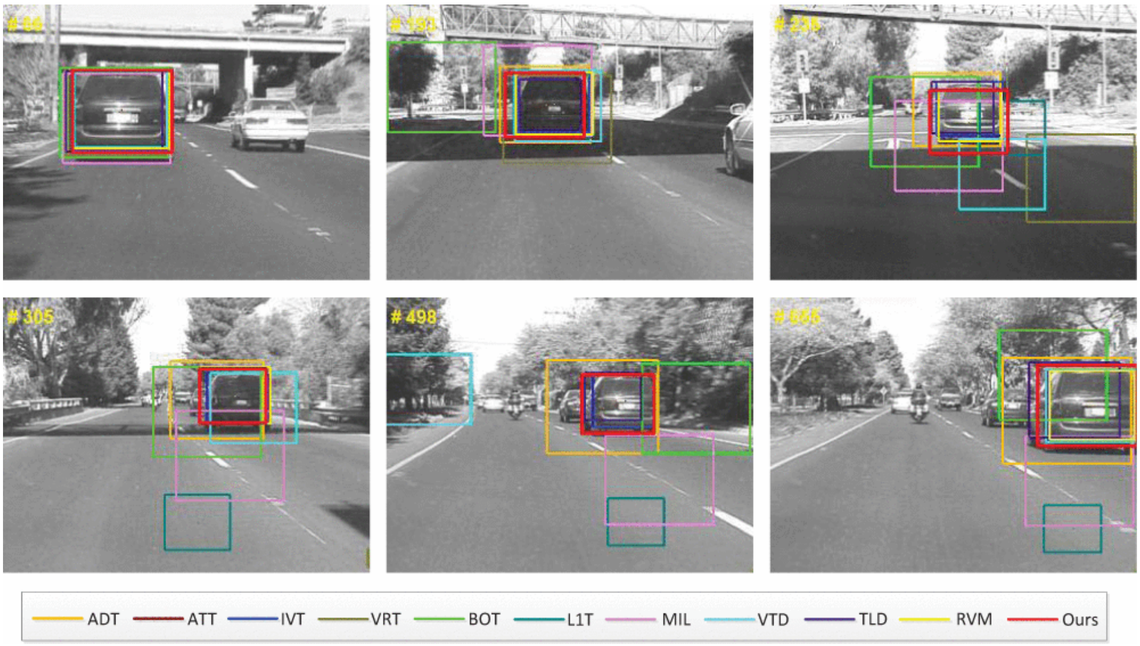

Figure 4: Frames from a tracking sequence featuring the bounding boxes indicating results of several tracking algorithms. Some examples of drift from the target and complete tracking failure can also be observed from some of the boxes. Source: Wang et. al. fig. 6 [4]

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. [7]

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. [4]

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. Models using online updates have implemented it for example with results of previous successful tracking results [10].

3.2 Datasets

Research on networks working with image data has been made easier by larger sets of both hand-labeled data and ones obtained by simple keyword searches from online image services. With the adoption of unsupervised training and architectures not working as classifiers unlabeled data can also be used to increase the size of the available training set. The labeled resources introduced here consist of sets labeled with object classes contained in the image or frame and ones for tracking with the target marked in each frame.

VOC was a yearly competition for object recognition and VOC 2012 [11] 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 subject's head, hands and feet.

The ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [12] 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 [13] 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. [14]

The datasets used for the Visual Object Tracking challenge (VOT) [15] 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) [16] 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.

The Visual Tracker Benchmark [13] 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 [15] and MOT [16] challenges also publish both the yearly challenge suite and results that can used to compare new networks against the participants.

Precision and success plots are common metrics for comparing trackers against others. Precision plot is the average center location error over the tracking sequence. This error is calculated as the distance between the centers of the tracking location and the hand-labeled ground truth. Success plot represents the average amount of overlap and the bounding box to the ground truth in relation to their sizes. The overlap score for a single frame is defined as the union of the boxes divided by their intersection. [14] The raw errors can also be used in calculating other indicators. Used examples of these are precision as the percentage of frames with a center distance error below a set value and success rate as the percentage of frames with an overlap score above some threshold [17].

4 Deep neural networks in tracking

Trackers based on deep learning have been researched because of DNNs' ability to capture hierarchies of features from raw data with minimal earlier domain specific knowledge. They provide greater versatility compared to traditional trackers that are based on hand-crafted sets of features. This chapter reviews both early research done in the field and more recent trackers that have made use of DNNs.

4.1 Early works

Multilayered CNNs are currently common as feature extractors, but an implementation of a CNN-based tracker [18] pre-dates the work of Krizhevsky et. al. [1] that sparked the current research on DNNs for classification tasks. Fan et. al. [18] proposed a shift-variant architecture utilizing the previous tracking result each frame. They recognized crowded scenes containing multiple objects similar to the target to be especially challenging as false positives could result in drift from the intended target. A reference position from the previous frame was used to provide additional information for locating the target in the current one. They also considered the shift-invariance of conventional detection CNNs to present its own challenges. Shift-invariance means that the position of an object does not affect the network's output, which is beneficial for classification tasks as it is desirable to recognize the objects in an image regardless of their position. However, a tracker is expected to identify the location of the target, which motivated the adoption of a shift-variant architecture. [18]

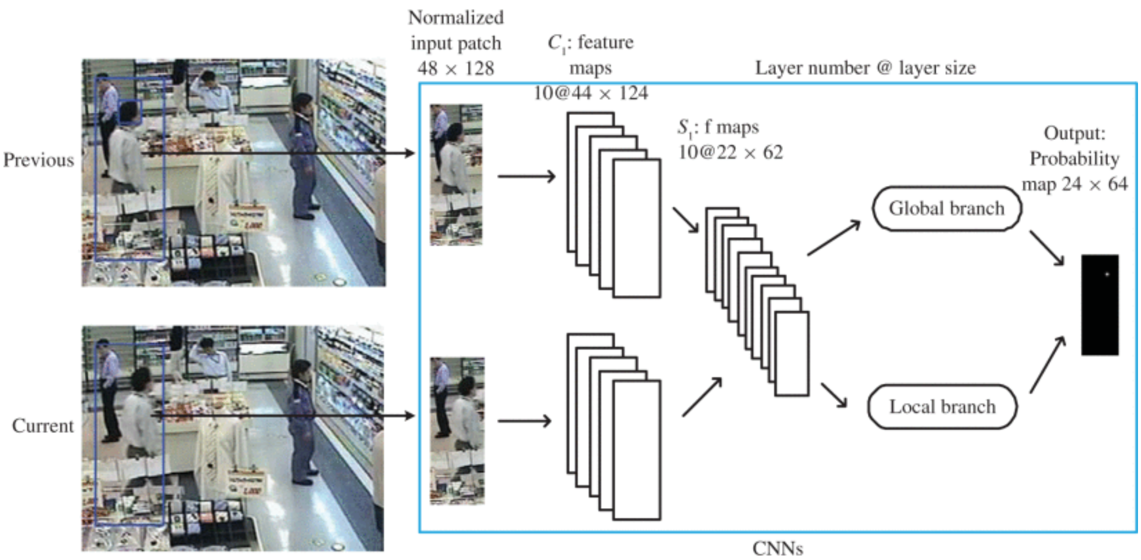

Figure 5: The network Fan et. al. [18] featured a branching desing to include two consecutive frames in the extracted features as well as to detect structures of features on both a local and global scale. Source: Fan et. al. fig. 1 [18]

The tracker Fan et. al. presented had separate input layers for extracting feature maps from the current and previous frame (fig. 5) [18]. These were then downsampled together before splitting the rest of the convolutions into two branches to obtain both global and local structures of features. Global structures were extracted with a series of convolutions, while local structures were discovered by sampling the branching output with a single convolution. The network's final layer took the outputs of both branches and produced the final probability map based on them. [18]

Training was performed on a set of 20,000 images obtained from surveillance videos and it was supervised by comparing the probability map resulting from a pair of frames to a target map. Online tracking was done on a fixed model to avoid the drift adaptive models can induce. The proposed tracker performed especially well when the arget's position or the view changed as the tracker was trained to track humans specifically. Even so, only being able to track a single class of targets is a major limitation. [18]

Wang and Yeung [7] proposed Deep Learning Tracker (DLT), a general object tracker consisting of a pre-trained SDAE and an additional classification layer. Their goal was to combine philosophies of generative and discriminative trackers by developing a discriminative tracker that learns and uses an effective image representation. The SDAE was trained to extract generic image features using 1 million randomly sampled images from the Tiny Images dataset [19] of 80 million images labelled by their main subject. After training, a sigmoid classification layer was added to the encoder part of the SDAE to complete the tracker. The model could also be tuned during tracking if a significant change in the target's appearance is detected. Tracking could achieve an average frame rate of 15 frames/s on a GPU, which is sufficient for many real-time applications. The tracker also compared favorably to then state of the art trackers on a set of 10 video sequences. [7]

4.2 Current state of the art

The ideas behind DLT were developed further by Wang et. al. [20]. They observed that DLT couldn't obtain deep features with temporal invariance due to training on unrelated images instead of videos. Another remark was that DLT doesn't have an integrated objective function to adapt the encoder to a target as the weights are only updated if the target appearance seems to have changed during tracking. The new CNN based feature learning method was integrated into an existing tracking system called ASLSA [21], which originally used raw pixel values as its representations. [20]

The two layer feature model learned features capable of handling complicated motion transformations based on the work of Zou et. al. [22]. It was trained on auxiliary video sequences and the goal was to learn features invariant between two frames, which results high-level features strong against non-linear motion patterns. These generic features didn't include appearance information of specific target objects so a domain adaptation module was added to both layers to readjust them according to a specific target. [20]

Evaluation on some challenging sequences showed that the tracker Wang et. al. [20] proposed could track targets that underwent non-rigid object deformation and in- or out-of-plane rotations. It was also observed to perform better than the baseline trackers and beat DLT in 5 of the 8 sequences tested. However, the tracker was implemented as a single-threaded CPU program and it only reached 0.6 frames/s in comparison to the 15 frames/s of the GPU implementation of DLT [7].

The work of Ding et. al. [10] focused on the issue of motion blur. They noted that generic trackers assume to have a blur free video to work with, while motion blur is very common in real videos. It was stated that the performance of such trackers may drop significantly if applied to videos with severe motion blur and two challenges were noted in such situations: the appearance features of the object are damaged by blur and abrupt motion of it is difficult to estimate. Deblurring the input was dismissed as too computationally costly and potentially prone to change appearance features. [10]

A SDAE trained on blurred images was proposed as a solution for capturing blur-invariant features. The layers were first pre-trained in sequence and in an unsupervised setting, but fine-tuning of the model was done on all layers simultaneously. Only the encoding layers were used after the joint training. [10]

When initializing the online tracker, the positive samples retrieved by sampling around the target are blurred using kernels simulating combinations of different magnitudes and directions of blur to produce a model less affected by blur effects. Not blurring the training samples might result in tracking failure or drift if abrupt and severe motion blur occurs in the beginning of the sequence. The background is sampled around the initial bounding box for negative samples and those are used without transformations. Tracking results are stored during tracking to update the model if a significant appearance change is detected by the maximum confidence dropping below a set threshold. [10]

Evaluation was done in two parts. First, severely blurred videos were used to test performance with severe blur and abrupt motion. After that, commonly used challenging sequences were used to evaluate general performance in difficult conditions. The new tracker performed well in both scenarios as it placed in the top two trackers in several categories and good results were shown overall. Real-time tracking seems also feasible as the tracker ran at 5–10 frames/s on a quad-core CPU and moderately powerful GPU while the implementation was said to have room for optimization. [10]

DeepTrack [17] is a CNN based tracker capable of competitive performance with training occurring entirely during tracking. They attributed the difficulties of adopting CNNs in tracking to a limited amount of positive training samples available in the tracking sequences, a tendency to overfit to the most recent observation and pure computational intensity. Their method employed a special type of loss function that consisted of a structural term and a truncated norm. The structural term was included to positive samples with different significance levels depending on the uncertainty of the object location and the truncated norm helped to reduce the number of samples needed in back-propagation to accelerate training. [17]

The network consisted of two convolutional layers and two fully connected layers 6. It was trained on multiple low-level features extracted from the image such as normalized gray-scale image and image gradient. Each cue had its own branch of layers. These layers were trained before substituting the fully connected layers of all channels with a common fusion layer for joint training. The fusion layer was responsible for final labeling in the tracking process. [17]

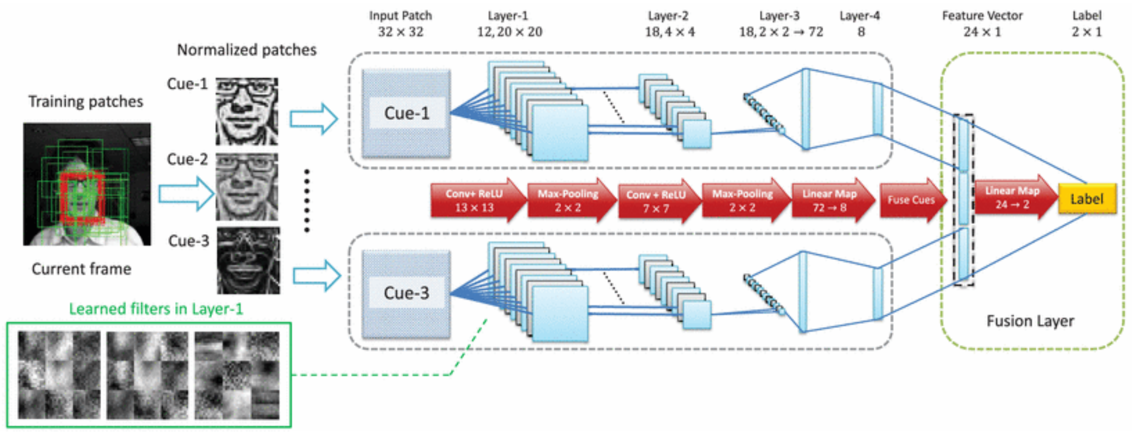

Figure 6: DeepTrack extacted three cues from each frame and each of them were passed to their own branch of two convolutional layers and two fully connected ones. A fusion layer then combined their outputs to the final output. [17] Source: Li et. al. fig. 1 [17]

The model was only updated when a significant change appearance change was detected and the fusion layer was updated using a slower rate than the preceding feature layers. The feature representations were assumed to change fast and each image cue to contribute more stably. Update was also constrained by having the window of positive samples in memory be longer than that of negative samples. This was done to reduce overfitting. The positive samples past the first frame were also judged less than reliable and were assigned a label noise value that was taken into account when the model was updated. DeepTrack was shown to outperform other trackers in two benchmarking challenges spanning 60 video sequences and it reached framerates of 2–4 frames/s on powerful hardware. [17] While not exactly real-time for all applications, the framerates are comparable to other methods.

Another novel tracker, Multi-Domain Network [23], had domain specific classification layers 8. Each video was treated as a separate domain and had their own branch after the generic layers. The generic layers consisted of three convolutions and two fully connected layers, while the branching domain specific classifiers were all a single fully connected layer for solving the binary classification problem between the target and its background. Training was done iteratively by training on a single domain per iteration but alternating between domains. Through this process, domain independent information was modelled to obtain generic feature representations that were resilient to common variations such as illumination changes and scale variations. [23]

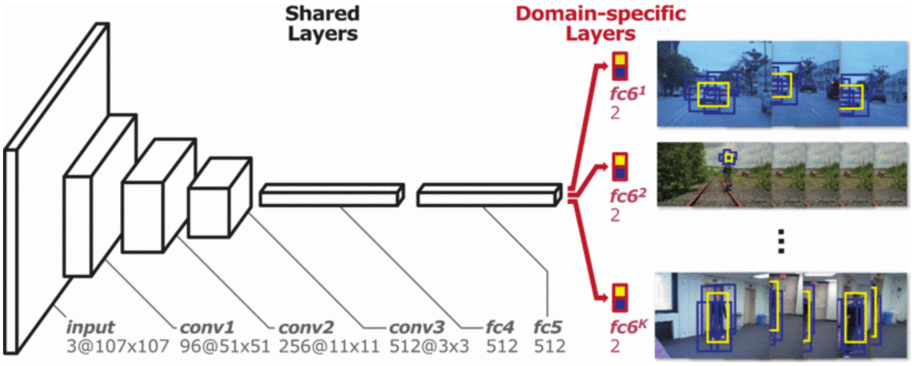

Figure 7: A depiction of the Multi-Domain Network. This form was used during training to train generic features with a domain specific final classification layer. The online version replaced the branches with a single randomized classifier to be tuned during tracking. [23] Source: Nam and Han fig. 1 [23]

In online tracking, the domain specific branches are substituted by a single randomly initialized layer that is tuned during tracking along with the shared fully connected layers. Model updates were handled by separate mechanisms for long and short term updates. Short term updates were only done if tracking failure was detected, while long term updates were performed at regular intervals using positive samples gathered over a long period. Both methods used negative samples from a short period since old negative samples are redundant or irrelevant to the current frame. Evaluation of the tracker showed good performance in two benchmarks with other tracking algorithms. A tracking speed of 1 frame/s was reached on a quad-core CPU and moderately powerful GPU. [23]

Wang et. al. [24] observed that many of the earlier works treated DNNs as black-box classifiers. They took a different approach and studied the properties of CNN features from the perspective of online visual tracking. Two realizations emerged from the results. First, different properties from different depths of features fit the problem. High-level features help distinguishing object classes while lower layers provide features for discerning the target from distracters. The new tracker was designed to switch between the two types based on the current distracters. Second, the features trained on classification data are for distinguishing generic objects and some of them might serve as noise in tracking. Thus, a method was proposed for discarding noisy or unrelated feature maps for the target. [24]

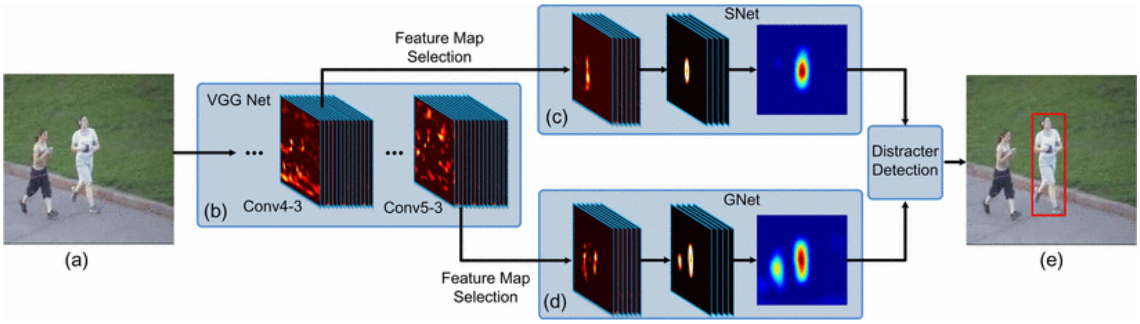

Figure 8: Two branches of convolutions evaluated the target position from different levels of features and the branch used in determining the position was decided based on the current scene. Source: Wang et. al. fig. 5 [24]

The tracker was built on top of the 16-layer VGG network [25] and the layers conv4_3 (10th) and conv5_3 (13th) were chosen as the two levels of features used. Conv4_3 was connected to a “specific network” that discriminated the target from the background and Conv5_3 fed a “general network” that captured category information. The VGG network had been pre-trained but the branches were initialized on the first frame of the tracking sequence. The two branches both output a heatmap and a distracter detection scheme determined which was to be used. After the first frame, updates were only done to the specific network using a setup similar to that of the Multi-Domain Network [23]. Framerates of 3 frames/s were reached on a powerful GPU with competitive performance in a benchmark. [24]

5 Conclusions

This thesis first presented the basic theory of DNNs and object tracking to provide basis for understanding the motivations to apply DNN architectures to tracking. Several novel and successful trackers were then reviewed with an emphasis on their architectures and the reasoning used in the design. Common advantages and drawbacks compared to traditional tracking methods were also noted.

DNNs are seen as a potential alternative to hand-crafted low-level models in object tracking. The traditional methods work well in well controlled environments but can have difficulties tracking in difficult conditions like partial occlusion of the target or motion blur. The hiearchical generic features a DNN learns are more resilient against such challenges which is a common motivation for developing a tracker using DNNs. Learning feature representations from training is also seen as a potential alternative to hand-tuning a traditional model which typically requires domain specific knowledge. [7]

Recent work has used both CNN [17] [23] [25] and SDAE [10] based methods for feature extraction but CNN is clearly the dominant approach. This is likely due to the wide use of them in classification and detection tasks since their good performance in the ImageNet challenge [1]. A noticeable trend in some trackers was the use of branching in the network to extract different levels of features from sequences [23] [24].

The research on using DNNs in trackers is still very young and The number of publications is relatively small. Good comparisons have not been made between many of the more recent methods and the publications themselves have contained benchmarks on different sets of test sequences or slighty differing representations of performance. While certainly interesting, the process of benchmarking is beyond the scope of this thesis because it could require modifying the trackers to work with input and output [14]. All the publications did themselves present evaluation results to show improvements over other methods on the selected test sets. Tracking speed fell mostly in the area of only a few frames/s with only one tracker claiming a speed of up to 10 frames/s [10]. It should be noted that the implementations were not said to be optimized and some of them only ran on the CPU [20] so advanced optimization techniques and the utilization of a GPU could result in more viable real-time tracking.

In conclusion, the research of DNN based object tracking is still relatively young and trackers have been proposed with emphasis on different challenges. Better generalization and being less affected by challenging conditions are common motivations for choosing deep features over traditional hand-crafted ones and trackers using DNNs have shown competitive performance when put against traditional trackers. However, real-time performance is still difficult to attain even with the advance in the hardware available.

References

[1] 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.

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

[3] 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.

[4] 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.

[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] 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.

[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] 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.

[11] 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.

[12] 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.

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

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] A. Torralba, R. Fergus, and W. T. Freeman. 80 million tiny images: A large data set for nonparametric object and scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(11):1958–1970, 2008. doi: 10.1109/TPAMI.2008.128.

[20] 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.

[21] X. Jia, H. Lu, and M. H Yang. Visual tracking via adaptive structural local sparse appearance model. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1822–1829, 2012. doi: 10.1109/CVPR.2012.6247880.

[22] W. Y. Zou, S. Zhu, A. Y. Ng, and K. Yu. Deep learning of invariant features via simulated fixations in video. In *Advances in Neural Information Processing Systems*, volume 4, pages 3203–3211, 2012.

[23] 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 January, pages 4293–4302, 2016.

[24] 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.

[25] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. URL http://arxiv.org/abs/1409.1556.