Context-aware Deep Feature Compression for High-speed Visual Tracking

Jongwon Choi[1] Hyung Jin Chang[2,3] Tobias Fischer[2] Sangdoo Yun[1,4]
Kyuewang Lee[1] Jiyeoup Jeong[1] Yiannis Demiris[2] Jin Young Choi[1]
[1]ASRI, ECE., Seoul National University [2]Personal Robotics Lab., EEE., Imperial College London
[3]School of Computer Science, University of Birmingham [4]Clova AI Research, NAVER Corp.
jwchoi.pil@gmail.com, {hj.chang,t.fischer,y.demiris}@imperial.ac.uk, {yunsd101,kyuewang,jy.jeong,jychoi}@snu.ac.kr

Abstract

We propose a new context-aware correlation filter based tracking framework to achieve both high computational speed and state-of-the-art performance among real-time trackers. The major contribution to the high computational speed lies in the proposed deep feature compression that is achieved by a context-aware scheme utilizing multiple expert auto-encoders; a context in our framework refers to the coarse category of the tracking target according to appearance patterns. In the pre-training phase, one expert auto-encoder is trained per category. In the tracking phase, the best expert auto-encoder is selected for a given target, and only this auto-encoder is used. To achieve high tracking performance with the compressed feature map, we introduce extrinsic denoising processes and a new orthogonality loss term for pre-training and fine-tuning of the expert auto-encoders. We validate the proposed context-aware framework through a number of experiments, where our method achieves a comparable performance to state-of-the-art trackers which cannot run in real-time, while running at a significantly fast speed of over 100 fps.

1. Introduction

The performance of visual trackers has vastly improved with the advances of deep learning research. Recently, two different groups for deep learning based tracking have emerged. The first group consists of online trackers which rely on continuous fine-tuning of the network to learn the changing appearance of the target [25, 30, 35, 36, 40]. While these trackers result in high accuracy and robustness, their computational speed is insufficient to fulfil the real-time requirement of online tracking. The second group is composed of correlation filter based trackers utilising raw deep convolutional features [6, 7, 10, 22, 27]. However, these features are designed to represent general objects contained in large datasets such as ImageNet [28] and therefore are of high dimensionality. As the computational time for the correlation filters increases with the feature dimensionality, trackers within the second group do not satisfy the real-time requirement of online tracking either.

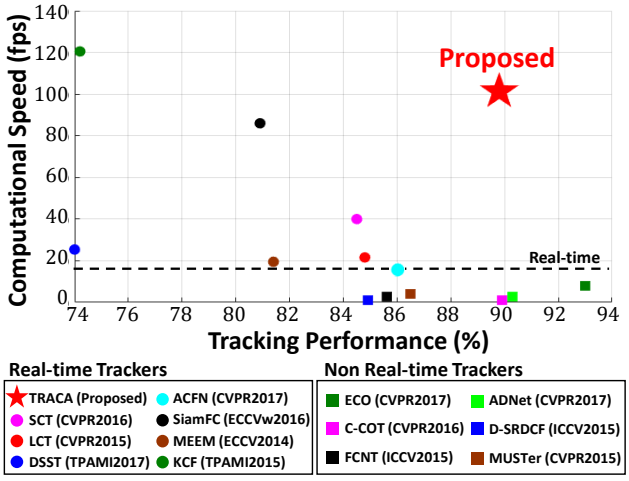


Figure 1. **Comparison of computational efficiency.** This plot compares the performance and computational speed of the proposed tracker (TRACA) with previous state-of-the-art trackers using the CVPR2013 dataset [37]. TRACA shows comparable performance with the best performing non real-time trackers, while running at a fast speed of over 100 fps.

In this work, we propose a correlation filter based tracker using context-aware compression of raw deep features, which reduces computational time, thus increasing speed. This is motivated by the observation that a lower dimensional feature map can sufficiently represent the single target object which is in contrast to the classification and detection tasks using large datasets that cover numerous object categories. Compression of high dimensional features into a low dimensional feature map is performed using autoencoders [11, 24, 32, 39]. More specifically, we employ multiple auto-encoders whereby each auto-encoder specialises in a specific category of objects; these are referred to as *expert auto-encoders*. We introduce an unsupervised approach to find the categories by clustering the training samples according to contextual information, and subsequently train

one expert auto-encoder per cluster. During visual tracking, an appropriate expert auto-encoder is selected by a context-aware network given a specific target. The compressed feature map is then obtained after fine-tuning the selected expert auto-encoder by a novel loss function considering the orthogonality of the correlation filters. The compressed feature map contains reduced redundancy and sparsity, which increases accuracy and computational efficiency of the tracking framework. To track the target, correlation filters are applied to the compressed feature map. We validate the proposed framework through a number of self-comparisons and show that it outperforms other trackers using raw deep features while being notably faster at a speed of over 100 fps (see Fig. 1).

2. Related Works

Online deep learning based trackers: Recent trackers based on online deep learning [25, 30, 35, 36, 40] have outperformed previous low-level feature-based trackers. Wang *et al*. [35] proposed a framework simultaneously utilising shallow and deep convolutional features to consider detailed and contextual information of the target respectively. Nam and Han [25] introduced a novel training method which avoids overfitting by appending a classification layer to a convolutional neural network that is updated online. Tao *et al*. [30] utilised a Siamese network to estimate the similarities between the target's previous appearance and the current candidate patches. Yun *et al*. [40] suggested a new tracking method using an action decision network which can be trained by a reinforcement learning method with weakly labelled datasets. However, trackers based on online deep learning require frequent fine-tuning of the networks, which is slow and prohibits real-time tracking. David *et al*. [16] and Bertinetto *et al*. [1] proposed pre-trained networks to quickly track the target without online fine-tuning, but the performance of these trackers is lower than that of the state-of-the-art trackers.

Correlation filter based trackers: The correlation filter based approach for visual tracking has become increasingly popular due to its rapid computation speed [2, 4, 5, 8, 17, 20, 23]. Henriques *et al*. [17] improved the tracking performance by extending the correlation filter to multi-channel inputs and kernel-based training. Danelljan *et al*. [8] developed a new correlation filter that can detect scale changes of the target. Ma *et al*. [23] and Hong *et al*. [20] integrated correlation filters with an additional long-term memory system. Choi *et al*. [5] proposed a tracker with an attentional mechanism exploiting previous target appearance and dynamics.

Correlation filter based trackers showed state-of-the-art performance when deep convolutional features were utilised [6, 7, 10, 27]. Danelljan *et al*. [7] extended the regularised correlation filter [9] to use deep convolutional features. Danelljan *et al*. [10] also proposed a novel correlation filter to find the target position in the continuous domain to incorporate features of various resolutions. Ma *et al*. [27] estimated the target position by fusing the response maps obtained from convolutional features of various resolutions. However, even though each correlation filter works fast, raw deep convolutional features have too many channels to be handled in real-time. A first step towards decreasing the feature space was made by Danelljan *et al*. [6] by considering the linear combination of raw deep features, however the method still cannot run in real-time, and the deep feature redundancy was not fully suppressed.

Multiple-context deep learning frameworks: Our proposed tracking framework benefits from the observation that the performance of deep networks can be improved using contextual information to train multiple specialised deep networks. Indeed, there are several works utilizing such a scheme. Li *et al*. [21] proposed a cascaded framework detecting faces through multiple neural networks trained by samples divided according to the degree of their detection difficulty. Vu *et al*. [33] integrated the head detection results from two neural networks, one specialising in local information and the other one in global information. Neural networks specialising in local and global information have also been utilised in the saliency map estimation task [34, 43]. In crowd density estimation, many works [26, 29, 42] have increased their performance by using multiple deep networks with different receptive fields to cover various scales of crowds.

3. Methodology

The proposed TRAcker based on Context-aware deep feature compression with multiple Auto-encoders (TRACA) consists of multiple expert auto-encoders, a context-aware network, and correlation filters as shown in Fig. 2. The expert auto-encoders robustly compress raw deep convolutional features from VGG-Net [3]. Each of them is trained according to a different context, and thus performs context-dependent compression (see Sec. 3.1). We propose a context-aware network to select the expert auto-encoder best suited for the specific tracking target, and only this auto-encoder is running during online tracking (see Sec. 3.2). After initially adapting the selected expert auto-encoder for the tracking target, its compressed feature map is utilised as an input of correlation filters which track the target online. We introduce the general concept of correlation filters in Sec. 3.3 and then detail the tracking processes including the initial adaptation and the online tracking in Sec. 3.4.

3.1. Expert Auto-encoders

Architecture: Auto-encoders have shown to be suitable for unsupervised feature learning [18, 19, 32]. They offer a way to learn a compact representation of the input while retaining the most important information to recover the input given the compact representation. In this paper, we propose to use a set of N_e expert auto-encoders of the same structure,

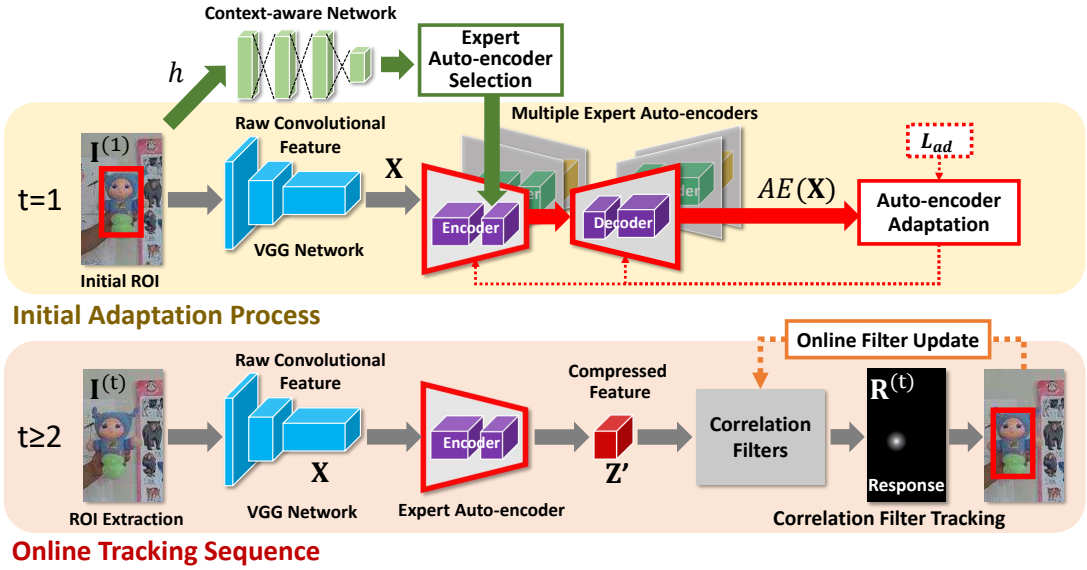


Figure 2. **Proposed algorithm scheme.** The expert auto-encoder is selected by the context-aware network and fine-tuned once by the ROI patch at the initial frame ($\mathbf{I}^{(1)}$). For the following frames, we first extract the ROI patch ($\mathbf{I}^{(t)}$) centred at the previous target position. Then, a raw deep convolutional feature ($\mathbf{X}$) is obtained through VGG-Net, and is compressed by the fine-tuned expert auto-encoder. The compressed feature ($\mathbf{Z}'$) is used as the feature map for the correlation filter, and the target's position is determined by the peak position of the filter response. After each frame, the correlation filter is updated online by the newly found target's compressed feature.

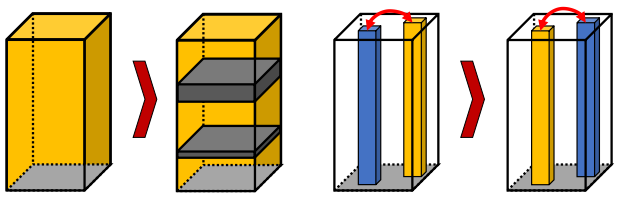
each covering a different context. The inputs to be compressed are raw deep convolutional feature maps obtained from one of the convolution layers in VGG-Net [3].

To achieve a high compression ratio, we stack N_l encoding layers which are followed by N_l decoding layers in the auto-encoder. The l-th encoding layer f_l is a convolutional layer working as $f_l : \mathbb{R}^{w\times h\times c_l} \rightarrow \mathbb{R}^{w\times h\times c_{l+1}}$, thus reducing the channel dimension c_l of the input to latent channel dimension c_{l+1} while preserving the resolution of the feature map. The output of f_l is provided as input to f_{l+1} such that the channel dimension c decreases as the feature maps pass through the encoding layers. More specifically, in our proposed framework one encoding layer reduces the channel dimension in half, *i.e.* $c_{l+1} = c_l/2$ for $l \in \{1, \cdots, N_l\}$. By denoting the $(N_l - k + 1)$-th decoding layer by g_k in the adverse way of f_l, $g_k : \mathbb{R}^{w\times h\times c_{k+1}} \rightarrow \mathbb{R}^{w\times h\times c_k}$ expands the input channel dimension c_{k+1} into c_k to restore the original dimension c_1 of $\mathbf{X}$ at the last layer of the decoder, where $k \in \{1, \cdots, N_l\}$. Then, the auto-encoder AE can be expressed as $AE(\mathbf{X}) \equiv g_1(\cdots(g_{N_l}(f_{N_l}(\cdots(f_1(\mathbf{X})))))) \in \mathbb{R}^{w\times h\times c_1}$ for a raw convolutional feature map $\mathbf{X} \in \mathbb{R}^{w\times h\times c_1}$, and the compressed feature map in the auto-encoder is defined as $\mathbf{Z} \equiv f_{N_l}(\cdots(f_1(\mathbf{X}))) \in \mathbb{R}^{w\times h\times c_{N_l+1}}$. All convolution layers are followed by the ReLU activation function, and the size of their convolution filters is set to 3×3.

Pre-training: The pre-training phase for the expert auto-encoders is split into three parts, each serving a distinct purpose. First, we train the base auto-encoder AE^o using all training samples to find context-independent initial compressed feature maps. Then, we perform contextual clustering on the initial compressed feature maps of AE^o to find N_e context-dependent clusters. Finally, these clusters are used to train the expert auto-encoders initialised by the base auto-encoder with one of the sample clusters.

The purpose of the base auto-encoder is twofold: Using the context-independent compressed feature maps to cluster the training samples and finding good initial weight parameters from which the expert auto-encoders can be fine-tuned. The base auto-encoder is trained by raw convolutional feature maps $\{\mathbf{X}_j\}_{j=1}^m$ with a batch size m. The $\mathbf{X}_j$ is obtained as the output from a convolutional layer involved in VGG-Net [3] fed by randomly selected training images $\mathbf{I}_j$ from a large image database such as ImageNet [28].

To make the base auto-encoder more robust to appearance changes and occlusions, we use two denoising criteria which help to capture distinct structures in the input distribution (illustrated in Fig. 3). The first denoising criterion is a *channel corrupting process* where a fixed number of feature channels is randomly chosen and the values for these channels is set to 0 (while the other channels remain unchanged), which is similar to the destruction process of denoising auto-encoders [32]. Thus all information for these channels is removed and the auto-encoder is trained to recover this information. The second criterion is an *exchange process*, where some spatial feature vectors of the convolutional feature are randomly interchanged. Since the receptive fields of the feature vectors cover different regions within an image, exchanging the feature vectors is similar to interchanging regions within the input image. Thus, interchanging feature vectors that cover the background region and target region



(a) Channel corrupting process (b) Feature vector exchange process

Figure 3. **Extrinsic denoising criteria.** To increase robustness of the compressed feature map in the pre-training, two extrinsic denoising criteria are applied to the raw deep feature map which is the input of the auto-encoder. (a) In the channel corrupting process, some randomly selected channels are set to zero. (b) In the exchange process, randomly chosen feature vectors are interchanged.

respectively leads to a similar effect as the background occluding the target. Therefore, the auto-encoders are trained to be robust against occlusions. We denote $\{\check{\mathbf{X}}_j\}_{j=1}^m$ as the mini-batch after performing the two denoising processes. Then, the base auto-encoder AE^o can be trained by minimising the distance between the input feature map $\mathbf{X}_j$ and its output $AE^o(\check{\mathbf{X}}_j)$ with the noisy sample $\check{\mathbf{X}}_j$.

However, when we only consider the distance between the input and the final output of the base auto-encoder, we frequently observed an overfitting problem and unstable training convergence. To solve these problems, we design a novel loss based on a multi-stage distance which consists of the distances between the input and the outputs obtained by the partial auto-encoders. The partial auto-encoders $\{AE_i(\mathbf{X})\}_{i=1}^{N_l}$ contain only a portion of the encoding and decoding layers of their original auto-encoder $AE(\mathbf{X})$, while the input and output sizes match that of the original auto-encoder, *i.e.* $AE_1(\mathbf{X}) = g_1(f_1(\mathbf{X}))$, $AE_2(\mathbf{X}) = g_1(g_2(f_2(f_1(\mathbf{X}))))$, $\cdots$ when $AE(\mathbf{X}) = g_1(\cdots(g_{N_l}(f_{N_l}(\cdots(f_1(\mathbf{X}))))))$. Thus, the loss based on the multi-stage distance can be described as:

$$L_{ae} = \frac{1}{m}\sum_{j=1}^{m}\sum_{i=1}^{N_l}\|\mathbf{X}_j - AE_i^o(\check{\mathbf{X}}_j)\|_2^2, \tag{1}$$

where $AE_i^o(\mathbf{X})$ is the i-th partial auto-encoder of $AE^o(\mathbf{X})$, and recall that m denotes the mini batch size.

Then, we cluster the training samples $\{\mathbf{I}_j\}_{j=1}^N$ according to their respective feature maps compressed by the base auto-encoder, where N denotes the total number of training samples. To avoid overfitting of the expert auto-encoders due to a too small cluster size, we introduce a two-step clustering algorithm which avoids small clusters.

In the first step, we find $2N_e$ samples which are chosen randomly from the feature maps compressed by the base auto-encoder (note that this is twice the amount of desired clusters). We repeat the random selection 1000 times and find the samples which have the largest Euclidean distance amongst them as initial centroids. Then, all training samples are clustered by k-means clustering with $k = 2N_e$ using the compressed feature maps. In the second step, among the resulting $2N_e$ centroids, we remove the N_e centroids of the clusters with the smallest number of included samples. Then, N_e centroids remain, and we cluster the training samples again using these centroids, which results in N_e clusters including enough samples to avoid the overfitting problem. We denote the cluster index for $\mathbf{I}_j$ as $d_j \in \{1, ..., N_e\}$.

The d-th expert auto-encoder AE^d is then found by fine-tuning the base auto-encoder using the training samples with contextual cluster index d. The training process (including the denoising criteria) differs from the base auto-encoder only in the training samples.

3.2. Context-aware Network

Architecture: The context-aware network selects the expert auto-encoder which is most contextually suitable for a given tracking target. We adopt a pre-trained VGG-M model [3] for the context-aware network since it contains a large amount of semantic information from pre-training on ImageNet [28]. Given a 224×224 RGB input image, our context-aware network consists of three convolutional layers $\{$*conv1*, *conv2*, *conv3*$\}$ followed by three fully connected layers $\{$*fc4*, *fc5*, *fc6*$\}$, whereby $\{$*conv1*, *conv2*, *conv3*, *fc4*$\}$ are identical to the corresponding layers in VGG-M. The weight parameters of *fc5* and *fc6* are initialised randomly with zero-mean Gaussian distribution. *fc5* is followed by a ReLU function and contains 1024 output nodes. Finally *fc6* has N_e output nodes and is combined with a softmax layer to estimate the probability for each of the expert auto-encoders to be suited for the tracking target.

Pre-training: The context-aware network takes a training sample $\mathbf{I}_j$ as input and outputs the estimated probabilities of that sample belonging to cluster index d_j. It is being trained by a batch $\{\mathbf{I}_j, d_j\}_{j=1}^{m'}$ of image/cluster-index pairs where m' is the mini-batch size for the context-aware network. We fix the weights of $\{$*conv1*, *conv2*, *conv3*, *fc4*$\}$, and train the weights for $\{$*fc5*, *fc6*$\}$ by minimising the multi-class loss function L_{pr} using stochastic gradient descent, where

$$L_{pr} = \frac{1}{m'}\sum_{j=1}^{m'} H(d_j, h(\mathbf{I}_j)), \tag{2}$$

H denotes the cross-entropy loss, and $h(\mathbf{I}_j)$ is the predicted cluster index of $\mathbf{I}_j$ by the context-aware network h.

3.3. Correlation Filter

Before detailing the tracking process of TRACA, we briefly introduce the functionality of conventional correlation filters using a single-channel feature map. Based on the property of the circulant matrix in the Fourier domain [13], correlation filters can be trained quickly which leads to high-performing trackers under low computational load [17]. Given the vectorised single-channel training feature map $\mathbf{z} \in \mathbb{R}^{wh\times 1}$ and the vectorised target response map $\mathbf{y}$ obtained from a 2-D Gaussian window with size $w \times h$ and

variance σ_y^2 as in [17], the vectorised correlation filter $\mathbf{w}$ can be estimated by:

$$\mathbf{w} = \mathcal{F}^{-1}\left(\frac{\widehat{\mathbf{z}} \odot \widehat{\mathbf{y}}}{\widehat{\mathbf{z}} \odot \widehat{\mathbf{z}}^* + \lambda}\right), \tag{3}$$

where $\hat{\mathbf{y}}$ and $\hat{\mathbf{z}}$ represent the Fourier-transformed vector of $\mathbf{y}$ and $\widehat{\mathbf{z}}$ respectively, $\widehat{\mathbf{z}}^*$ is the conjugated vector of $\mathbf{z}$, $\odot$ denotes an element-wise multiplication, $\mathcal{F}^{-1}$ stands for an inverse Fourier transform function, and λ is a predefined regularisation factor.

For the vectorised single-channel test feature map $\mathbf{z}' \in \mathbb{R}^{wh\times 1}$, the vectorised response map $\mathbf{r}$ can be obtained by:

$$\mathbf{r} = \mathcal{F}^{-1}\left(\widehat{\mathbf{w}} \odot \widehat{\mathbf{z}}'^*\right). \tag{4}$$

Then, after re-building a 2-D response map $\mathbf{R} \in \mathbb{R}^{w\times h}$ from $\mathbf{r}$, the target position is found from the maximum peak position of $\mathbf{R}$.

3.4. Tracking Process

To track a target in a scene, we rely on a correlation filter based algorithm using the compressed feature map of the expert auto-encoders as selected by the context-aware network. We describe the initial adaptation of the selected expert auto-encoder in Sec. 3.4.1 followed by a presentation of the correlation filter based tracking algorithm in Sec. 3.4.2.

3.4.1 Initial Adaptation Process

The initial adaptation process contains the following parts. We first extract a region of interest (ROI) including the target from the initial frame, and the expert auto-encoder suitable for the target is selected by the context-aware network. Then, the selected expert auto-encoder is fine-tuned using the raw convolutional feature maps of the training samples augmented from the ROI. When we obtain the compressed feature map from the fine-tuned expert auto-encoder, some of its channels represent background objects rather than the target. Thus, we introduce an algorithm to find and remove the channels which respond to the background objects.

Region of interest extraction: The ROI is centred around the target's initial bounding box, and is 2.5 times bigger than the target's size to cover the area nearby. We then resize the ROI of width W and height H to 224×224 in order to match the expected input size of the VGG-Net. This results in the resized ROI $\mathbf{I}^{(1)} \in \mathbb{R}^{224\times 224\times 3}$ for the rgb domain. For grey-scale images, the grey value is replicated three times to obtain $\mathbf{I}^{(1)}$. The best expert auto-encoder for the tracking scene is selected according to the contextual information of the initial target using the context-aware network h, and we can denote this auto-encoder as $AE^{h(\mathbf{I}^{(1)})}$.

Initial sample augmentation: Even though we use two denoising criteria as described earlier, we found that the compressed feature maps of the expert auto-encoders show a deficiency for targets which become blurry or are flipped. Thus, we augment $\mathbf{I}^{(1)}$ in several ways before fine-tuning the selected auto-encoder. To tackle the blurriness problem, four augmented images are obtained by filtering $\mathbf{I}^{(1)}$ with Gaussian filters with variances $\{0.5, 1.0, 1.5, 2.0\}$. Two more augmented images are obtained by flipping $\mathbf{I}^{(1)}$ around the vertical and horizontal axes respectively. Then, the raw convolutional feature maps extracted from the augmented samples can be represented by $\{\mathbf{X}_j^{(1)}\}_{j=1}^{7}$.

Fine-tuning: The fine-tuning of the selected auto-encoder differs from the pre-training process for the expert auto-encoders. As there is a lack of training samples, the optimisation rarely converges when applying the denoising criteria. Instead, we employ a correlation filter orthogonality loss L_{ad} which considers the orthogonality of the correlation filters estimated from the compressed feature map of the expert auto-encoder, where L_{ad} is defined as:

$$L_{ad} = \sum_{j=1}^{7}\sum_{i=1}^{N_l}\left\{\left\|\mathbf{X}_j^{(1)} - AE_i(\mathbf{X}_j^{(1)})\right\|_2^2 + \lambda_\Theta \sum_{k,l=1}^{c^{i+1}} \Theta(\mathbf{w}_{jik}, \mathbf{w}_{jil})\right\}, \tag{5}$$

where $\Theta(\mathbf{u}, \mathbf{v}) = (\mathbf{u}\cdot\mathbf{v})^2/(\|\mathbf{u}\|_2^2\|\mathbf{v}\|_2^2)$ and $\mathbf{w}_{jik}$ defines a vectorised correlation filter estimated by Eq.(3) using the vectorised k-th channel of the compressed feature map $f_i(\cdots(f_1(\mathbf{X}_j^{(1)})))$ from the selected expert auto-encoder. The correlation filter orthogonality loss allows increasing the interaction among the correlation filters as estimated from the different channels of the compressed feature maps. The fine-tuning is performed by minimising L_{ad} using stochastic gradient descent. The differentiation of L_{ad} is described in Appendix A of the supplementary material.

Background channel removal: The compressed feature map $\mathbf{Z}^\forall$ can be obtained from the fine-tuned expert auto-encoder. Then, we remove the channels within $\mathbf{Z}^\forall$ which have large responses outside of the target bounding box. Those channels are found by estimating the channel-wise ratio of foreground and background feature responses. First, we estimate the channel-wise ratio of the feature responses for channel k as

$$ratio^k = \|vec(\mathbf{Z}_{bb}^{k,\forall})\|_1 / \|vec(\mathbf{Z}^{k,\forall})\|_1, \tag{6}$$

where $\mathbf{Z}^{k,\forall}$ is the k-th channel feature map of $\mathbf{Z}^\forall$ and $\mathbf{Z}_{bb}^{k,\forall}$ is obtained from $\mathbf{Z}^{k,\forall}$ by setting the values out of the target bounding box to 0 while the other values are untouched. Then, after sorting all channels according to $ratio^k$ in descending order, only the first N_c channels of the compressed feature map are utilised as input to the correlation filters. We denote the resulting feature map as $\mathbf{Z} \in \mathbb{R}^{S\times S\times N_c}$, where S is the feature size.

3.4.2 Online Tracking Sequence

Correlation filter estimation & update: We first obtain the resized ROI for the current frame t using the same

method as in the initial adaptation, *i.e.* the resized ROI is centred at the target's centre and its size is 2.5 times the target's size and resized to 224×224. After feeding the resized ROI to the VGG-Net, we obtain the compressed feature map $\mathbf{Z}^{(t)} \in \mathbb{R}^{S \times S \times N_c}$ by inserting the raw deep convolutional feature map of the VGG-Net into the adapted expert auto-encoder.

Then, using Eq.(3), we estimate independent correlation filters $\mathbf{w}^{k,(t)}$ for each feature map $\mathbf{Z}^{k,(t)}$, where $\mathbf{Z}^{k,(t)}$ denotes the k-th channel of $\mathbf{Z}^{(t)}$. Following [17], we suppress background regions by multiplying each $\mathbf{Z}^{k,(t)}$ with cosine windows of the same size. For the first frame, we can estimate the correlation filters $\bar{\mathbf{w}}^{k,(1)}$ with Eq.(3) given by $\mathbf{Z}^{k,(1)}$. For the following frames ($t > 1$), the correlation filters are updated as follows:

$$\bar{\mathbf{w}}^{k,(t)} = (1-\gamma)\bar{\mathbf{w}}^{k,(t-1)} + \gamma \mathbf{w}^{k,(t)}, \tag{7}$$

where γ is an interpolation factor.

Tracking: After estimating the correlation filter, we need to find the position $[x^t, y^t]$ of the target in frame t. As we assume that $[x^t, y^t]$ is close to the target position in the previous frame $\left([x^{t-1}, y^{t-1}]\right)$, we extract the tracking ROI from the same position as the ROI for the correlation filter estimation of the previous frame. Then, we can obtain the compressed feature map $\mathbf{Z}^{'(t)}$ for tracking using the adapted expert auto-encoder. Inserting $\mathbf{Z}^{'(t)}$ and $\bar{\mathbf{w}}^{k,(t-1)}$ to Eq.(4) then provides the channel-wise response map $\mathbf{R}^{k,(t)}$ (we apply the multiplication of cosine windows in the same manner as for the correlation filter estimation).

We then need to combine $\mathbf{R}^{k,(t)}$ to the integrated response map $\mathbf{R}^{(t)}$. We use a weighted averaging scheme, where we use the validation score s^k as weight factor with

$$s^k = \exp\left(-\lambda_s \|\mathbf{R}^{k,(t)} - \mathbf{R}_o^{k,(t)}\|_2^2\right), \tag{8}$$

and $\mathbf{R}_o^{k,(t)} = \mathcal{G}(\mathbf{p}^{k,(t)}, \sigma^2)_{S \times S}$ is a 2-D Gaussian window with size $S \times S$ and variance σ_y^2 centred at the peak point $\mathbf{p}^{k,(t)}$ of $\mathbf{R}^{k,(t)}$. Then, the integrated response map is calculated as:

$$\mathbf{R}^{(t)} = \sum_{k=1}^{N_c} s^k \mathbf{R}^{k,(t)}. \tag{9}$$

Following [5], we find the sub-pixel target position $\mathbf{p}^{(t)}$ by interpolating the response values near the peak position. Finally, the target position $[x^t, y^t]$ is found as:

$$[x^t, y^t] = [x^{t-1}, y^{t-1}] + round([W, H] \odot \mathbf{p}^{(t)}/S). \tag{10}$$

Scale changes: To handle scale changes of the target, we extract two additional ROI patches scaled from the previous ROI patch size with scaling factors 1.015 and 1.015^{-1} respectively in the tracking sequence. The new target scale is chosen as the scale where the respective maximum value of the response map (from the scaled ROI) is the largest.

Full occlusion handling: To handle full occlusions, a re-detection algorithm is adopted. The overall idea is to introduce a so-called re-detection correlation filter which is not being updated and applied to the position of the target at the time where an occlusion has been detected. A full occlusion is assumed when a sudden drop of the maximum response value $R_{max}^{(t)} \equiv \max(\mathbf{R}^{(t)})$ is detected as described by $R_{max}^{(t)} < \lambda_{re} \bar{R}_{max}^{(t-1)}$ with $\bar{R}_{max}^{(t)} = (1-\gamma)\bar{R}_{max}^{(t-1)} + \gamma R_{max}^{(t)}$ and $\bar{R}_{max}^{0} = R_{max}^{1}$ (note that this is the same γ as in Eq.(7)). If that condition is fulfilled, the correlation filter at time $(t-1)$ is used as re-detection correlation filter. During the next N_{re} frames, the target position as determined by the re-detection correlation filter is being used if the maximum value of the re-detection filter's response map is larger than the maximum value of the response map obtained by the normal correlation filter. Furthermore, $\bar{\mathbf{w}}^{k,(t)}$ are replaced by the ones of the re-detection correlation filter, and the target scale is reset to the scale when the occlusion was detected.

4. Experimental Result

4.1. Implementation

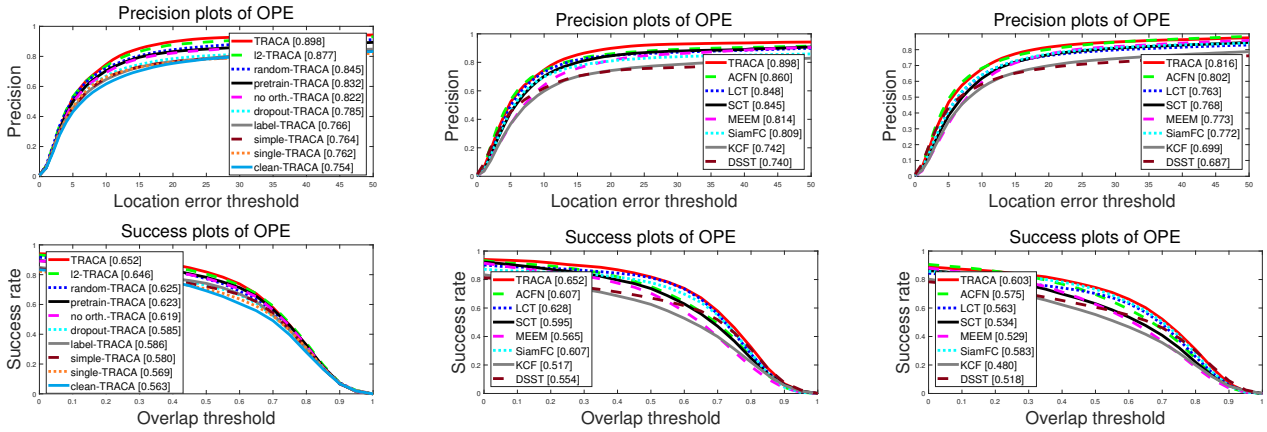
The feature response after the second convolution layer ($conv2$) of VGG-M [3] was given to the auto-encoders as raw convolutional feature input. The number of expert auto-encoders was set to $N_e = 10$, and their depth to $N_l = 2$. The mini-batch size for all auto-encoders was set to $m = 10$. The learning rate for the base auto-encoder was set to 10^{-10}, and for expert auto-encoders to 10^{-9}. The base auto-encoder was trained for 10 epochs, and the expert auto-encoders were fine-tuned for 30 epochs. The proportions for the two extrinsic denoising processes were set to 10% respectively. For training the context-aware network, the mini-batch size and the learning rate were set to $m' = 100$ and 0.01, respectively. The weight for the orthogonality loss term was set to $\lambda_\Theta = 10^3$, and the reduced channel dimension after the background channel removal was $N_c = 25$. The parameters for the correlation filter based tracker were set to $\sigma_g = 0.05$, $\lambda = 1.0$, and $\gamma = 0.025$, and $\lambda_s = 50$. The parameters for full occlusion handling, λ_{re} and N_{re}, were experimentally determined to 0.7 and 50 using scenes with occlusions.

We used MATLAB and MatConvNet [31] to implement the proposed framework. The computational environment had an Intel i7-2700K CPU @ 3.50GHz, 16GB RAM, and an NVIDIA GTX1080 GPU. The computational speed was 101.3 FPS in the CVPR2013 dataset [37]. We release the source code along with the attached experimental results[1].

4.2. Dataset

The classification image samples included in VOC2012 [12] were used to pre-train the expert auto-

[1] https://sites.google.com/site/jwchoivision/



(a) Self-comparison on CVPR2013 (b) Evaluation plots on CVPR2013 (c) Evaluation plots on TPAMI2015

Figure 4. **Evaluation results.** TRACA showed the best performance within the self-comparison, and the state-of-the-art performance amongst real-time trackers in CVPR2013 [37] and TPAMI2015 [38] datasets. The numbers within the legend are the average precisions when the centre error threshold equals 20 pixels (top row), or the area under the curve of the success plot (bottom row).

encoders and the context-aware network. To evaluate the proposed framework, we used the CVPR2013 [37] (51 targets, 50 videos) and TPAMI2015 [38] (100 targets, 98 videos) datasets, which contain the ground truth of the target bounding box at every frame. These datasets have been frequently used [5, 10, 15, 17, 20, 25, 41] as they include a large variety of environments to evaluate the general tracking performance.

4.3. Evaluation Measure

As performance measure, we used the average precision curve of one-pass evaluation (OPE) as proposed in [37]. The average precision curve was estimated by averaging the precision curves of all sequences, which was obtained using two sources: location error threshold and overlap threshold. As representative scores of trackers, the average precisions when the location error threshold equals 20 pixels and the area under the curve of the success plot were used.

4.4. Self-comparison

To analyse the effectiveness of TRACA, we compare TRACA with nine variants: no orth.-TRACA, pretrain-TRACA, clean-TRACA, dropout-TRACA, random-TRACA, l_2-TRACA, label-TRACA, simple-TRACA, and single-TRACA. In no orth.-TRACA, the weight factor λ_Θ for the orthogonality loss term is set to zero. Pretrain-TRACA skipped the initial adaptation step and directly utilised the pre-trained expert auto-encoder. Clean-TRACA used the expert auto-encoders which were pre-trained without any extrinsic denoising process. Dropout-TRACA adopted a dropout scheme instead of the proposed dimension corrupting process, while keeping the feature vector exchange process. Random-TRACA randomly selected the suitable expert auto-encoder. l_2-TRACA selected the best suitable expert auto-encoder according to the smallest l_2 generation error estimated from the initial target. Label-TRACA utilised 20 class labels of the pre-training dataset (VOC2012 [12]) as the contextual clusters. The expert auto-encoders of simple-TRACA were trained and fine-tuned by minimising the Euclidean distance between their inputs and final outputs, *i.e.* without using the multi-stage distance. Single-TRACA utilised the compressed feature map of the base auto-encoder[2].

Table 1. Quantitative results on the CVPR2013 dataset [37]

	Algorithm	Pre. score	Mean FPS	GPU
Proposed	**TRACA**	**89.8%**	101.3	Y
	l_2-TRACA	87.7%	101.2	Y
	random-TRACA	84.4%	99.5	Y
	pretrain-TRACA	83.2%	98.8	Y
	no orth.-TRACA	82.2%	101.2	Y
	dropout-TRACA	78.5%	97.5	Y
	label-TRACA	76.6%	97.2	Y
	simple-TRACA	76.4%	94.1	Y
	single-TRACA	76.2%	100.9	Y
	clean-TRACA	75.4%	92.9	Y
Real-time	ACFN [5]	86.0%	15.0	Y
	LCT [23]	84.8%	21.6	N
	SCT [4]	84.5%	40.0	N
	MEEM [41]	81.4%	19.5	N
	SiamFC [1]	80.9%	86.0	Y
	KCF [17]	74.2%	**120.5**	N
	DSST [8]	74.0%	25.4	N
Non Real-time	ECO [6]	**93.0%**	**8.0**	Y
	ADNet [40]	90.3%	2.9	Y
	C-COT [10]	89.9%	0.5	N
	MUSTer [20]	86.5%	3.9	N
	FCNT [35]	85.6%	3.0	Y
	D-SRDCF [7]	84.9%	0.2	N

The results of the comparison with these nine trackers are shown in Table 1 and Fig. 4 (a). The results of random-TRACA and l_2-TRACA show decreased performance which reflects the importance of the context-aware network. In the

[2]For fair comparison, we train the base auto-encoder for 20 epochs in the case of single-TRACA.

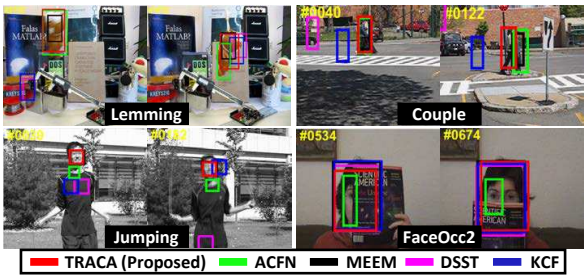


Figure 5. **Qualitative results.** The used sequences are Lemming, Couple, Jumping, FaceOcc2, CarDark, and Soccer from the left-top.

result of pretrain-TRACA, the performance was reduced by 6.6% when the expert auto-encoder was not adapted initially. The initial adaptation ignoring the orthogonality loss term (no orth.-TRACA) further decreased the performance by 1% compared to pretrain-TRACA. When the extrinsic denoising processes were not applied, the tracking performance reduced dramatically (14.3%) according to the result of clean-TRACA. Similarly, as shown in the result of dropout-TRACA, the proposed dimension corrupting process made the expert auto-encoders more robust than a dropout scheme (11.3% performance reduction). When the multi-stage distance was not used, the performance was reduced by 13.4% as shown in the result of simple-TRACA. Single-TRACA showed a dramatic reduction in the tracking performance (13.6%), which demonstrates that the multiple-context scheme was effective to compress the raw deep convolutional feature for a specific target. Finally, the tracking performance was reduced dramatically in label-TRACA (13.2%), which shows that clustering in feature space is beneficial when training the expert auto-encoders.

4.5. Comparison with State-of-the-art Trackers

The results of the state-of-the-art methods, including ECO [6], ADNet [40], ACFN [5], C-COT [10], SiamFC [1], FCNT [35], D-SRDCF [7], SCT [4], LCT [23], and DSST [8] were obtained from the authors. In addition, the results of MUSTer [20], MEEM [41], and KCF [17] were estimated using the authors' implementations[3].

In Table 1, the precision scores of the algorithms on the CVPR2013 dataset are presented along with the computational speed and whether they make use of a GPU. Fig. 4 (b-c) compares the performances of the real-time trackers, where TRACA demonstrates state-of-the-art performance in both the CVPR2013 and TPAMI2015 datasets while running at over 100 fps. Some qualitative results are shown in Fig. 5.

4.6. Further Analysis

The context in the proposed framework refers to a coarse category of the compressed feature maps encoding the target object appearance. To illustrate the context in practice, we extracted the five samples with the highest scores within the context-aware network for each contextual category using Caltech256 [14]. As shown in Fig. 6, the contextual clusters categorise the samples according to appearance patterns.

[3]For fair comparison, the computational time was estimated by the same computer as TRACA and included the image resizing time.

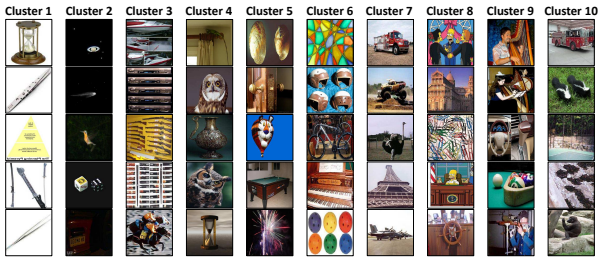


Figure 6. **Top-5 images for each contextual cluster.** Each column represents one context category and consists of the five samples within Caltech256 [14] that have the highest scores of the context-aware network for this category. The results confirm that the contextual clusters represent the category of appearance patterns.

In Appendix B, we evaluate the impact of the chosen target layer of VGG-Net and the number of contextual clusters on the proposed framework. In Appendix C, we analyse the correlation matrix among various computer vision datasets, which was obtained by estimating the correlation among the histograms of the results from the context-aware network.

5. Conclusion

In this paper, a new visual tracking framework based on context-aware deep feature compression using multiple auto-encoders was proposed. Our main contribution is to introduce a context-aware scheme which includes expert auto-encoders specialising in one context, and a context-aware network which is able to select the best expert auto-encoder for a specific tracking target. This leads to a significant speed-up of the correlation filter based tracker utilising deep convolutional features. Our experiments lead to the compelling finding that our framework achieves a high-speed tracking ability of over 100 fps while our framework maintains a competitive performance compared to the state-of-the-art trackers. We expect that embedding our context-aware deep feature compression scheme will be integrated with other trackers utilising raw deep features, which will increase their robustness and computational efficiency. In addition, the scheme can be utilised as a way to avoid the overfitting problem in other computer vision tasks where only few training samples are available, such as in image k-shot learning and image domain adaptation. As a future work, we will jointly train the expert auto-encoders and the context-aware network to potentially further increase the performance due to the correlation between the contextual clustering and the feature compression.

Acknowledgements: This work was supported by ICT R&D program MSIP/IITP [2017-0-00306, Outdoor Surveillance Robots], Next-Generation ICD Program through NRF funded by Ministry of S&ICT [2017M3C4A7077582], and BK21^{+}.

References

[1] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. Torr. Fully-convolutional siamese networks for object tracking. In *ECCV workshops*, 2016. 2, 7, 8

[2] D. S. Bolme, J. R. Beveridge, B. A. Draper, and Y. M. Lui. Visual object tracking using adaptive correlation filters. In *CVPR*, 2010. 2

[3] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: Delving deep into convolutional nets. In *BMVC*, 2014. 2, 3, 4, 6

[4] J. Choi, H. J. Chang, J. Jeong, Y. Demiris, and J. Y. Choi. Visual tracking using attention-modulated disintegration and integration. In *CVPR*, 2016. 2, 7, 8

[5] J. Choi, H. J. Chang, S. Yun, T. Fischer, Y. Demiris, and J. Y. Choi. Attentional correlation filter network for adaptive visual tracking. In *CVPR*, 2017. 2, 6, 7, 8

[6] M. Danelljan, G. Bhat, F. S. Khan, and M. Felsberg. ECO: efficient convolution operators for tracking. In *CVPR*, 2017. 1, 2, 7, 8

[7] M. Danelljan, G. Hager, F. S. Khan, and M. Felsberg. Convolutional features for correlation filter based visual tracking. In *ICCV workshops*, 2016. 1, 2, 7, 8

[8] M. Danelljan, G. Hager, F. S. Khan, and M. Felsberg. Discriminative scale space tracking. *IEEE Trans. on PAMI*, 39(8):1561–1575, 2016. 2, 7, 8

[9] M. Danelljan, G. Häger, F. Khan, and M. Felsberg. Learning spatially regularized correlation filters for visual tracking. In *ICCV*, 2015. 2

[10] M. Danelljan, A. Robinson, F. S. Khan, and M. Felsberg. Beyond correlation filters: Learning continuous convolution operators for visual tracking. In *ECCV*, 2016. 1, 2, 7, 8

[11] B. Du, W. Xiong, J. Wu, L. Zhang, L. Zhang, and D. Tao. Stacked convolutional denoising auto-encoders for feature representation. *IEEE Trans. on Cybernetics*, 47(4):1017–1027, 2017. 1

[12] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html. 6, 7

[13] R. M. Gray. Toeplitz and circulant matrices: A review. *Foundations and Trends® in Communications and Information Theory*, 2(3):155–239, 2006. 4

[14] G. Griffin, A. Holub, and P. Perona. Caltech-256 object category dataset. In *Caltech Technical Report*. California Institute of Technology, 2007. 8

[15] S. Hare, S. Golodetz, A. Saffari, V. Vineet, M. M. Cheng, S. L. Hicks, and P. H. S. Torr. Struck: Structured output tracking with kernels. *IEEE Trans. on PAMI*, 38(10):2096–2109, 2016. 7

[16] D. Held, S. Thrun, and S. Savarese. Learning to track at 100 fps with deep regression networks. In *ECCV*, 2016. 2

[17] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista. High-speed tracking with kernelized correlation filters. *IEEE Trans. on PAMI*, 37(3):583–596, 2015. 2, 4, 5, 6, 7, 8

[18] G. E. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, 2006. 2

[19] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006. 2

[20] Z. Hong, Z. Chen, C. Wang, X. Mei, D. Prokhorov, and D. Tao. Multi-store tracker (MUSTer): a cognitive psychology inspired approach to object tracking. In *CVPR*, 2015. 2, 7, 8

[21] H. Li, Z. Lin, X. Shen, J. Brandt, and G. Hua. A convolutional neural network cascade for face detection. In *CVPR*, 2015. 2

[22] C. Ma, J.-B. Huang, X. Yang, and M.-H. Yang. Hierarchical convolutional features for visual tracking. In *ICCV*, 2015. 1

[23] C. Ma, X. Yang, C. Zhang, and M.-H. Yang. Long-term correlation tracking. In *CVPR*, 2015. 2, 7, 8

[24] R. Memisevic. Gradient-based learning of higher-order image features. In *ICCV*, 2011. 1

[25] H. Nam and B. Han. Learning multi-domain convolutional neural networks for visual tracking. In *CVPR*, 2016. 1, 2, 7

[26] D. Oñoro-Rubio and R. J. López-Sastre. Towards perspective-free object counting with deep learning. In *ECCV*, 2016. 2

[27] Y. Qi, S. Zhang, L. Qin, H. Yao, Q. Huang, J. Lim, and M.-H. Yang. Hedged deep tracking. In *CVPR*, 2016. 1, 2

[28] 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. *IJCV*, 115(3):211–252, 2015. 1, 3, 4

[29] D. B. Sam, S. Surya, and R. V. Babu. Switching convolutional neural network for crowd counting. In *CVPR*, 2017. 2

[30] R. Tao, E. Gavves, and A. W. Smeulders. Siamese instance search for tracking. In *CVPR*, 2016. 1, 2

[31] A. Vedaldi and K. Lenc. MatConvNet – Convolutional Neural Networks for MATLAB. In *ACM MM*, 2015. 6

[32] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. Extracting and composing robust features with denoising autoencoders. In *ICML*, 2008. 1, 2, 3

[33] T.-H. Vu, A. Osokin, and I. Laptev. Context-aware CNNs for person head detection. In *ICCV*, 2015. 2

[34] L. Wang, H. Lu, X. Ruan, and M.-H. Yang. Deep networks for saliency detection via local estimation and global search. In *CVPR*, 2015. 2

[35] L. Wang, W. Ouyang, X. Wang, and H. Lu. Visual tracking with fully convolutional networks. In *ICCV*, 2015. 1, 2, 7, 8

[36] L. Wang, W. Ouyang, X. Wang, and H. Lu. STCT: Sequentially training convolutional networks for visual tracking. In *CVPR*, 2016. 1, 2

[37] Y. Wu, J. Lim, and M.-H. Yang. Online object tracking: A benchmark. In *CVPR*, 2013. 1, 6, 7

[38] Y. Wu, J. Lim, and M.-H. Yang. Object tracking benchmark. *IEEE Trans. on PAMI*, 37(9):1834–1848, 2015. 7

[39] X. Yan, J. Yang, K. Sohn, and H. Lee. Attribute2image: Conditional image generation from visual attributes. In *ECCV*, 2016. 1

[40] S. Yun, J. Choi, Y. Yoo, K. Yun, and J. Y. Choi. Action-decision networks for visual tracking with deep reinforcement learning. In *CVPR*, 2017. 1, 2, 7, 8

[41] J. Zhang, S. Ma, and S. Sclaroff. Meem: Robust tracking via multiple experts using entropy minimization. In *ECCV*, 2014. 7, 8

[42] Y. Zhang, D. Zhou, S. Chen, S. Gao, and Y. Ma. Single-image crowd counting via multi-column convolutional neural network. In *CVPR*, 2016. 2

[43] R. Zhao, W. Ouyang, H. Li, and X. Wang. Saliency detection by multi-context deep learning. In *CVPR*, 2015. 2