VILNIUS UNIVERSITY
FACULTY OF MATHEMATICS AND INFORMATICS
DEPARTMENT OF COMPUTER SCIENCE

Convolutional neural network features for visual tracking

Master's Thesis

Done by: **Ignas Namajūnas**

(signature)

Advisor: **Dr. Rimantas Kybartas**

(signature)

Reviewer: **Mindaugas Eglinskas**

(signature)

VILNIUS 2016

VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
INFORMATIKOS KATEDRA

Konvoliucinių neuroninių tinklų požymiai vaizdiniui sekimui

Magistro baigiamasis darbas

Atliko: **Ignas Namajūnas**

(parašas)

Darbo vadovas: **Dr. Rimantas Kybartas**

(parašas)

Recenzentas: **Mindaugas Eglinskas**

(parašas)

VILNIUS 2016

Summary

This Thesis analyzed convolutional neural network feature transferability to the problem of visual object tracking. It started with a theoretical analysis of how the tracking problem is approached using convolutional neural networks. Based on the analysis, a methodology was designed to quantify transferability. It included choosing a rich tracking dataset with provided ground-truth annotations, several convolutional neural network architectures with three different layers for each of them as features that are compared in their efficiency in representing a tracked object and a quantitative measure based on the classification accuracy by a Linear Support Vector machine that uses those representations as input. Using the above methodology various qualitative and quantitative measures were acquired that compares the architectures, their layers, different ways to split the data into train and test sets for the SVM and different ways to generate negative examples, that are not of the tracked object, for the classification. Based on the acquired measures a number of recommendations were proposed for methods that use convolutional neural networks for tracking.

Santrauka

Šis darbas tyrė konvoliucinių neuroninių tinklų požymių efektyvumą vaizdiniui sekimui. Pradžioje buvo atlikta literatūros analizė apie tai, kaip konvoliuciniai neuroniniai tinklai yra panaudojami sekimo problemai spręsti. Remiantis šia analize, buvo sukurta detali metodologija kaip įvertinti šį efektyvumą. Buvo parinkta turininga sekimo duomenų bazė, parinktos kelios skirtingos konvoliucinių neuroninių tinklų architektūros su jų trimis sluoksniais, kurie buvo panaudoti kaip požymiai skirti reprezentuoti sekamą objektą. Šie požymiai buvo naudojami ir pritaikant tiesines atraminių vektorių mašinas, kurių klasifikavimo tikslumai ir buvo parinkti kaip požymių efektyvumo įvertinimas. Buvo gauti įvertinimai, palyginantys skirtingas architektūras, jų sluoksnius, skirtingus būdus išskaidyti duomenis į mokymo ir testavimo aibes atraminių vektorių mašinai bei neigiamų pavyzdžių, tai yra tų, kurie nėra sekamas objektas, generavimo būdus. Remiantis gautais įvertinimas buvo pateikta keletas rekomendacijų sekimo metodams, kurie naudoja konvoliucinius neuronų tinklus.

Contents

Introduction

Object tracking is considered to be one of the most important computer vision problems, as it is a fundamental task which has many applications. Deep learning and more specifically convolutional neural networks relatively recently emerged as a dominant method to solve various computer vision tasks. This makes the application of convolutional neural networks to the problem of visual object tracking a promising and important area of research and this was the general direction of this Thesis.

More specifically, the goal was to provide valuable insights and recommendations for methods that apply convolutional neural networks to visual object tracking. Other works have already shown that convolutional neural network features are indeed transferable to other tasks, different from the ones on which the networks were trained on, however, detailed research of transferability to visual object tracking is still lacking. What is more, even though convolutional neural networks work very well in practice, there is still relatively little understanding about what the network actually learns and applying them usually involves heuristics found by trial and error. Understanding these networks in the context of visual object tracking was the motivation for this Thesis.

To achieve the above goal, several tasks had to be done:

- Designing a methodology by which convolutional neural network feature transferability could be analyzed
- Acquiring both qualitative and quantitative measures for transferability using the above methodology
- Giving suggestions for visual object tracking methods that are based on convolutional neural networks based on the above measures

1 Theory

1.1 Deep Learning

The success of machine learning algorithms depends greatly on data representation, namely the features of the data [BCV12]. For this reason, when designing various machine learning applications, a great deal of time and effort is spent on hand-crafting the right features, suited for a specific task with its associated data. It would be very convenient if these features were automatically learned by the learning algorithm and this is what Deep Learning tries to achieve [BCV12]. It takes as input raw data and tries to automatically learn the right features. It does it in a hierarchical fashion, which means it starts with raw data and non-linearly transforms it into a slightly more abstract representation. This slightly more abstract representation is then used as input to another layer, which transforms it into an, even more, abstract representation. With some number of such transformations, very complex functions can be learned that map the raw data into a representation form which is free from irrelevant variations and is only left with the most discriminative information present in the data [LBH15].

There are various architectures and algorithms associated with the name of Deep Learning. Two prominent examples, namely autoencoders and convolutional neural networks, will be presented in the next subsections.

1.1.1 Autoencoders

One of the factors that make solving the problem of visual object tracking hard is a limited amount of labeled data that can be used to train the designed systems. For this reason, unsupervised approaches, that do not require the data to be labeled, might be very useful for an object tracking system. One important example of an architecture that can be trained with unlabeled data is the autoencoder, which is a particular type of a neural network.

An autoencoder tries to learn a representation of the data that retains as much information as possible [VLL+10]. It consists of two parts: an encoder part and a decoder part. The encoder part transforms the input into a hidden representation. Assume that the input to the autoencoder x is a d dimensional vector of real values. Then, the encoder is a mapping [VLL+10]:

$$f_\theta(x) = s(Wx + b) \tag{1}$$

Where θ is the parameter set of the encoder $\{W, b\}$, W is $d' \times d$ matrix of weights and b is a d' dimensional offset vector. $s(x)$ is the sigmoid function:

$$s(x) = \frac{1}{1 + e^{-x}} \tag{2}$$

and if the input is a vector:

$$s(\mathbf{x}) = (s(x_1), s(x_2), ..., s(x_d))^T \tag{3}$$

The decoder part of an autoencoder transforms the hidden representation y back to the input space, attempting to reconstruct it [VLL+10]:

$$g_{\theta'}(y) = s(W'y + b') \tag{4}$$

Where θ' is the parameter set of the decoder part $\{W', b'\}$, similarly to θ.

As mentioned before, the autoencoder tries to retain information that was present in the input. However, only being able to retain information is not a sufficient requirement for learning a useful representation of the input [VLL+10], as the autoencoder could simply learn the identity mapping, which is a perfect reconstruction of the input. Hence, additional constraints are required.

One option is to make the number of hidden neurons smaller than the number of input neurons, or, using our previous notation, to make $d' < d$. The representation learned this way is a compression of the input which tries to retain as much information as possible [VLL+10].

Another option is to use an overcomplete representation, one which is of higher dimension than that of the input, that is to have $d' > d$, but to impose the so-called sparsity constraint [VLL+10]. The sparsity constraint forces the hidden neurons to be inactive most of the time, activating only for a subset of the training samples. Denote the activation of a hidden neuron j as a_j, the ith training sample as x^i and the number of training samples as m. Then, the average activation of the hidden neuron j:

$$\bar{\rho}_j = \frac{1}{m} \sum_{i=1}^{m} a_j(x^i) \tag{5}$$

The sparsity constraint requires that $\bar{\rho}$ would be approximately equal to the sparsity parameter ρ, a small constant, such as 0.05.

The above two options both constrain the representation learned to avoid learning the identity mapping. However, [VLL$^+$10] proposes a different strategy - slightly corrupting the initial data vector before giving it as input to the autoencoder. This strategy is partially based on the following ideas [VLL$^+$10]:

- A higher level representation that is learned by the autoencoder should be robust under corruptions of the input.
- Performing the denoising task, that is reconstructing the uncorrupted version of the input, requires extracting features that capture the structure of the input distribution well.

First of all, the input vector x is corrupted to obtain $\widetilde{x}$. Then, the corrupted version of the input $\widetilde{x}$ is mapped to the hidden representation y via the mapping f_θ defined earlier. The decoder part of the autoencoder is then used to map the hidden representation y back to the input space, attempting to reconstruct the uncorrupted input x, acquiring z. The reconstructed version of the uncorrupted input z is then compared to x and the reconstruction error $L_H(x, z)$ is calculated.

There are various ways the initial input x could be corrupted [VLL$^+$10]:

- Adding isotropic *Gaussian noise*: $\widetilde{x}|x \sim \mathcal{N}(x, \sigma^2 I)$.
- *Masking noise*: a percentage of the coordinates of x are set to be 0, where the coordinates are randomly chosen for each input vector.
- *Salt-and-pepper noise*: a percentage of the coordinates of x are set to be their minimal or maximal possible value, chosen randomly.

Up to this point, the autoencoder was presented as a single layer neural network. However, several layers of autoencoders can be stacked on top of each other to acquire a deep architecture.

After training, the stacked denoising autoencoder can act as a feature extractor for a classification layer, obtaining a deep architecture which could be fine-tuned using labeled data [VLL$^+$10].

1.1.2 Convolutional neural networks

Convolutional neural network is an architecture which is very well suited for data that consists of several multi-dimensional arrays and where the local statistics are translation invariant in those arrays. This means that if it is known that a specific feature is useful for a particular location in the array, it will likely be useful in the other locations [LeC12]. Convolutional neural networks make use of this property by using shared weights, which reduce the number of parameters that need to be learned from data and this reduces data and computational power required to train a convolutional neural network. It also has pooling layers to reduce the dimensionality of data and it also makes the architecture less sensitive to small local variations in the data [LeC12].

Assume that our input consists of images, that is of two-dimensional arrays of data, possibly with multiple color channels. The goal of the first convolutional layer is to extract patterns found within local regions in the input images [ZF13a]. This is achieved by convolving a filter with the input image pixels, resulting in a feature map, one for each filter. Additionally, a non-linear function f is applied point-wise to each of the feature maps. Various non-linear functions can be used for f, such as the hyperbolic tangent function:

$$f(x) = tanh(x) = \frac{e^{2x} - 1}{e^{2x} + 1} \tag{6}$$

Or the logistic function:

$$f(x) = \frac{1}{1 + e^{-x}} \tag{7}$$

Or the linear rectification function:

$$f(x) = max(0, x) \tag{8}$$

The resulting activations a, that is the feature map values after applying the chosen non-linear function, are given as input to the sub-sampling layer. It combines information in each of a set of small local regions R in the input and produces pooled feature maps as output, that is of smaller size than the input [ZF13a]. There is also a variety of ways to sub-sample, such as taking the average value of the values in the region R:

$$s = \frac{1}{|R|} \sum_{i \in R} a_i \tag{9}$$

Or taking the largest value in the region R:

$$s = \max_{i \in R} a_i \tag{10}$$

The above two sub-sampling operations are deterministic. There are also stochastic operations, such as the one presented in [ZF13a].

The process of applying convolution and sub-sampling layers can be repeated, potentially resulting in a very deep architecture.

Finally, after several stages of convolutions and sub-sampling, the resulting features are given as input to a fully connected multi-layer perceptron, which performs classification or some other task.

Convolutional neural networks recently became very popular due to their success in various pattern recognition tasks [Ben09]. One prominent example of those tasks is visual object recognition [KSH12].

1.1.3 Transfering convolutional neural network features to other tasks

Convolutional neural network layer activations have been found to be transferable as features to other tasks, different from the one that was used to train the convolutional neural network [DJV$^+$13]. What is more, different layers provide different features, where it is intuitively understood that features which are closer to the input are more general and the ones closer to the output more specific [YCBL14]. These different features extracted from various layers of a convolutional neural network have differences in their effectiveness for the new task, for which those features are used [HXHZ15].

1.2 Object tracking

Visual object tracking is the estimation of the size and location of an object as it moves in the scene [YOM06]. It is considered to be one of the most important components in various applications of computer vision [WLY13]. Possible applications include [YOM06]:

- Surveillance
- Human-Computer Interaction
- Robotics
- Autonomous Driving

It is a hard problem due to various reasons [YOM06]:

- Complicated object motion
- Partial or full occlusion of the tracked object
- Noisiness of the visual information
- Complicated and flexible form of the object
- Illumination changes of the scene
- Real-time requirement for various applications

Many approaches have been proposed to solve the visual tracking problem, as can be seen from the review [YOM06] or a more recent one [WLY13].

1.2.1 Deep Learning for object tracking

As Deep Learning techniques, especially convolutional neural networks, have been found to be very successful at various computer vision tasks, a variety of methods that use Deep Learning have been proposed to solve the visual tracking problem too. Some of these works will be reviewed in the following paragraphs.

1.2.1.1 Tracking with Deep Neural Networks

[JDB+13] uses a convolutional neural network for tracking a single object, given its location in the first frame. Their architecture also includes a Radial Basis Function Network (RBFN) to produce a confidence map of the object location.

In their architecture, both convolutional layers include 32 kernels of size 7×7. The non-linear function that is applied to the convolution result is hyperbolic tangent *tanh*. As for sub-sampling, they used L2-pooling, which, using our previous notation, is defined as:

$$s = \sqrt{\frac{1}{|R|} \sum_{i \in R} a_i^2} \tag{11}$$

Where the region R is of size 2×2. Once the feature vector was extracted by the two layers of convolution and sub-sampling (pooling), it was given to the RBFN to either construct a positive prototype neuron in the case of the first frame or to compare the feature vector to the positive prototype and estimate the similarity in the case of the other frames. Comparing the extracted feature vector of various locations in the frame to the saved feature vector extracted from the object location in the first frame resulted in a confidence map of the object location in the current frame. The estimated rectangle was chosen from positions in the map where the confidence exceeded a given threshold τ.

1.2.1.2 Learning a Deep Compact Image Representation for Visual Tracking

[WY13] used the architecture of a stacked denoising autoencoder to train it offline, that is before the tracking process began, and then used the encoder part of the autoencoder together with a classification layer to perform online tracking.

They trained the autoencoder using Tiny Images dataset [TFF08], randomly sampling 1 million images of size 32×32 from it.

The autoencoder was trained by minimizing two sums, the first one being:

$$\sum_{i=1}^{k} ||x_i - \hat{x}_i||_2^2 + \lambda(||W||_F^2 + ||W'||_F^2), \tag{12}$$

where W and W' are the weight matrices for the encoder and decoder parts, respectively, x_i is the i-th input vector, $\tilde{x_i}$ is it's corrupted version, $h_i = f(W\tilde{x_i} + b)$ is the hidden representation and $\hat{x} = f(W'h_i + b')$ is the reconstruction of the input. f is a non-linear function, which was either logistic sigmoid or hyperbolic tangent function, defined earlier. By minimizing the above sum, the autoencoder learns to reconstruct the input, while the parameter λ, which was set to 0.0001, controls the penalty of large values of the weights W and W', as $||\cdot||_F$ is the Frobenius norm.

The second sum, due to the imposed sparsity constraint, was the cross-entropy of the target sparsity ρ and the average empirical activation rate $\hat{\rho}$:

$$H(\rho||\hat{\rho}) = -\sum_{j=1}^{m}(\rho_j \log(\hat{\rho}_j) + (1-\rho_j)\log(1-\hat{\rho}_j)), \tag{13}$$

where m was the number of hidden units, ρ_j was a constant set to 0.05 and the $\hat{\rho}_j$ were the average empirical activation rates for each of the hidden units. To minimize this sum, gradient method with momentum was used.

To generate hypotheses about the current location of the object, particle filtering approach was used. With this approach, when a new frame arrived, 1000 particles were drawn and each of the predicted positions was given a weight depending on the likelihood estimated by propagating the hypothesized location in the image through the above network. If the likelihood of all the particles was lower than a constant τ, set to 0.9, the whole network was tuned again to adapt to the significant appearance change, which likely had happened. However, to avoid overfitting, a larger value of λ was used, equal to 0.002. The tracking result for each frame was the particle with the largest weight.

1.2.1.3 Robust Online Visual Tracking with a Single Convolutional Neural Network [LLP15] used a convolutional neural network to use the tracking-by-detection framework, where the three-layer network learned to distinguish the target object from the background in an online manner.

First of all, using the input gray-scale frame, 4 different image cues were generated - 3 locally normalized images were the normalization was carried out using different parameter sets and one gradient image. Each of these image cues was propagated through a convolutional neural network and the final responses were concatenated before giving it as input to the fully-connected classification layer.

When training a convolutional neural network, a particular loss function is minimized. For binary classification problems, a common choice is:

$$\mathcal{L} = \frac{1}{N}\sum_{n=1}^{N}||f(x_n;\Omega) - 1_n||_2, \tag{14}$$

where N is the number of training samples, x_n is the input patch, 1_n is it's ground truth label and f is the response of the network, parametrized by Ω. However, for object localization, [LLP15] propose various modifications of the above equation.

First of all, different training patches are given different weights, that is patches that are considered to be negative cases (not belonging to the tracked target) which are far away from the estimated position of the target and patches that are considered to be positive cases (belonging to the tracked target) and are close to the estimated position are given larger weight than those that are in between. What is more, as only the first estimate of the target's location, which is given as input, is of high confidence, negative samples that are far away from the estimated location, but have a high convolutional neural network score are given lower weights. Finally, to speed up the process of training the convolutional neural network using backpropagation, a truncated l_2 norm is proposed, which means that only patches that have high error are used in the training process.

To train the convolutional neural network online, Stochastic Gradient Descent was used. Positive samples were sampled in a way that would allow the network to learn the long term appearance of the object and it wouldn't overfit to the most recent frame while negative samples were sampled differently - more recent background patches were more likely to be used for training the network.

Finally, to improve the performance of the tracking algorithm, the convolutional neural network was only retrained when the loss function was above some predefined threshold.

When a new frame arrived, hypotheses about the current object location were randomly sampled from a Gaussian distribution centered at the previous object location with standard deviations equal to 10 for the two position parameters and equal to 0.02 for the relative scale parameter.

1.2.1.4 Other works Other significant works which use Deep Learning architectures, possibly fusing them with various other algorithms, for the problem of visual tracking, will be briefly summarized in this section.

[BFL$^+$11] used Restricted Boltzmann Machines, which is another architecture related to Deep Learning, particle filtering, and gaze estimation to simultaneously track and recognize objects. [KLL15] use stacked convolutional autoencoders to learn invariant features offline from unlabeled data. The trained stacked autoencoders are then used in a particle filtering framework to perform online tracking. [WLGY15] pre-trained a convolutional neural network offline and then fine-tuned it online to adapt to the appearance changes of the object. The convolutional neural network was trained to provide a probability map instead of a class label, which is the more usual output.

1.2.2 Transfering convolutional neural network features to tracking

As was mentioned previously, using convolutional neural network layer activations as features for other tasks is more effective than learning that task from scratch [YCBL14]. However, even though there already exist works that use convolutional neural network features as representations for the object to be tracked, there do not seem to be any detailed experiments that compare the effectiveness of various representations for tracking, where those representations come from pre-trained convolutional neural networks with different architectures. One work which is related is that of [WOWL15]. They used the architecture of [SZ14] and looked at the feature maps of two convolutional layers - 10th and 13th. They found that the 10th convolutional layer captures more discriminative information for intra-class variations and as a result helps with discriminating the tracked object from distractors while 13th convolutional layer captures more semantic information and helps distinguish the tracked object category [WOWL15]. The work of [MHYY15], where they also use the feature maps from different convolutional layers for target representation, is similar to [WOWL15] in this respect.

2 Methods

This section explains various methodological details about the experiments that were conducted. The first subsection presents five convolutional neural network architectures that were used to study the convolutional neural network feature transfer to visual tracking. The second subsection describes the data that was used for the experiments. Finally, the third subsection explains how different features that were extracted from those five convolutional neural network architectures were compared to each other in the context of visual object tracking.

2.1 Specific convolutional neural network architectures

Various convolutional neural network architectures have been proposed up to this date, differing in various aspects such as network depth, the number of neurons per each layer, size of the convolutional filters at each convolutional layer, convolution stride, max-pooling downsampling factor and others. As it is uncertain which architectures suit the task of visual object tracking better, several different architectures have been chosen for the experiments of this Thesis. They are the following:

1. The first convolutional neural network architecture is taken from [CSVZ14], in which the authors mention that it is similar to the architecture of [KSH12]. In the Thesis, similarly to the original work of [CSVZ14], this architecture will be named CNN-F. For this network, the input image is of size 224×224. It consists of 8 layers with learnable parameters - 5 convolutional and 3 fully-connected. What makes this architecture fast is a large stride for the first convolutional layer, which is 4 pixels.

2. The second convolutional neural network architecture, which is also taken from [CSVZ14], in which the authors say that is similar to the architecture from [ZF13b]. In the Thesis, similarly to the original work of [CSVZ14], this architecture will be named CNN-M. For this network, the input image is also of size 224×224, and it also consists of 8 layers with learnable parameters - 5 convolutional and 3 fully-connected. Compared to the CNN-F architecture, this architecture has a larger stride and a smaller receptive field in the first convolutional layer. However, for this architecture to remain relatively compu-

tationally nonintensive, a larger stride was used for the second convolutional layer.

3. The third convolutional neural network architecture, once again taken from [CSVZ14], which, according to the authors, is similar to the architecture from [SEZ$^+$13]. In the Thesis, similarly to the original work of [CSVZ14], this architecture will be named CNN-S. For this network, the input image is also of size 224×224, and it also consists of 8 layers with learnable parameters - 5 convolutional and 3 fully-connected. This architecture, similarly to the architecture of CNN-M, has stride equal to two in the first convolutional layer. However, differently from CNN-M, it also has stride equal to one in the second convolutional layer, which makes it slower than CNN-M. To partially compensate for this, CNN-S uses increased max-pooling downsampling factor for the first and fifth convolutional layers.

4. The fourth convolutional neural network architecture, taken from [SZ14], is quite different from the three architectures listed above. First of all, it is much deeper - it consists of 16 layers with learnable parameters, 13 of which are convolutional and 3 are fully-connected. All of the convolutional layers use the same receptive field size, which is 3×3, smaller than the ones used for the above architectures. However, some of these convolutional layers are stacked together without max-pooling layers in between, so, for example, two such layers have an effective receptive field size of 5×5, even though there are non-trivial differences, such as the fact that after each convolutional layer follows a non-linear rectification layer [SZ14]. This architecture will be called CNN-D.

5. The last one, which is fifth, architecture is also taken from [SZ14]. It is quite similar to the fourth one, however, it is even deeper - it consists of 19 layers with learnable parameters, 16 of which are convolutional and 3 of them are fully-connected. This architecture will be called CNN-E.

   As we see from the above, all of the chosen architectures differ in non-trivial ways. Additionally, the last two architectures, namely CNN-D and CNN-E, have much better classification accuracies on ILSRVC 2012 validation dataset [RDS$^+$15], which, according to *caffe Model Zoo* [JSD$^+$14], are as follows:

- CNN-F achieves 16.7% top-5 classification error, which is the percentage of images where the true label was not included among the five labels considered most likely by the convolutional neural network
- CNN-M achieves 13.7% top-5 classification error,
- CNN-S achieves 13.1% top-5 classification error,
- CNN-D achieves 7.5% top-5 classification error,
- CNN-E achieves 7.5% top-5 classification error,

It might be interesting to see if the above classification accuracies correlate to the effectiveness in representing a tracked object.

Aside from differences, they also have important similarities, such as, first of all, they all use the same input image size, which is 224×224. Different input image sizes might change the amount of information that a neural network receives and it might make the comparison of representations acquired by these neural networks more difficult. What is more, they all have the same number of fully-connected layers which all have the same dimensionality, and it is exactly these layers that will be used to acquire feature representations for the experiments of this Thesis. More specifically, three fully connected layers, named fc6, fc7 and fc8, respectively, will be used as representations, where fc6 and fc7 are of dimension 4096 while fc8 is of dimension 1000 for all the chosen convolutional neural networks. Different vector dimensionality could also make the direct comparison of the feature representations harder. Lastly, all of these convolutional neural networks have been trained using the same training data and using very similar training protocols, which is of [KSH12], where the training data consists of 1.2 million images that each has a single label out of 1000 possible. As training on such large amounts of data takes significant computational resources, already trained models of the above architectures were downloaded from *caffe Model Zoo* [JSD+14].

2.2 Dataset used for experiments

As the tracked object can undergo a variety of transformations, a dataset which covers many of them is required for meaningful experiments. After an extensive search for such a dataset, VOT2015 dataset was chosen [KML+15]. It consists of

60 short sequences in which a variety of different objects, one object per sequence, are labeled with a rotated rectangle in every frame. However, as the convolutional neural networks expect input in the form of a 224×224 sized rectangular image, the labeled rotated rectangles were converted to upright ones and then resized to the required size using linear interpolation. Some images, generated this way, are depicted below:

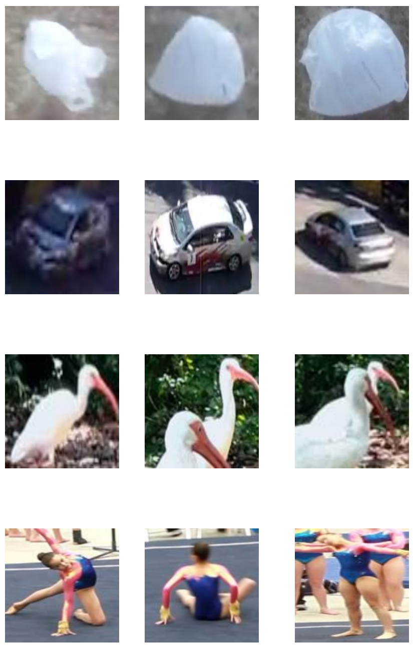


Figure 1: Example images from the generated sequences.

As we can see from the above figure, the sequences do include a variety of different object types undergoing a variety of different transformations. What is more, the third row with a bird shows that there are other similarly looking birds that act as distractors, which indicates that feature representations that capture very fine details are necessary for successful tracking.

Additionally, the original images in which the tracked object was labeled were used to generate samples for negatives - rectangles which either not include the tracked object at all or only include it partially. More specifically, such negative crops were generated:

1) Rectangles shifted to the left or right by 25%, 50%, 75% and 100% of the original object rectangle width value, where if the resulting rectangle moved outside of the original image, it was skipped. Some examples of such images, shifted to the right by 50%, are depicted below:

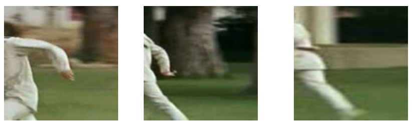


Figure 2: Example images from the sequences, shifted to the right by 50% of the rectangle width value.

2) Rectangles shifted to the top or bottom by 25%, 50%, 75% and 100% of the original object rectangle height value, where if the resulting rectangle moved outside of the original image, it was skipped. Some examples of such images, shifted to the top by 50%, are depicted below:

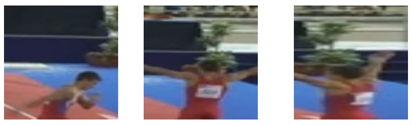


Figure 3: Example images from the sequences, shifted to the top by 50% of the rectangle height value.

3) Rectangles scaled by a factor of 0.5. More precisely, the center of the rectangle was fixed while width and height became two times smaller. Several examples of such images are depicted below:

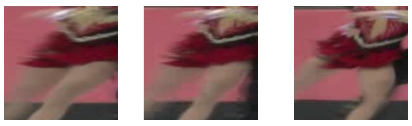


Figure 4: Example images from the sequences, scaled by a factor of 0.5

4) Rectangles scaled by a factor of 2.0. More precisely, the center of the rectangle was fixed while width and height became two times larger. Several examples of such images are depicted below:

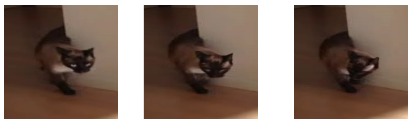


Figure 5: Example images from the sequences, scaled by a factor of 2.0

There was a total of 19 crops, including the non-shifted one. As the number of images for the central crop for all the 60 object sequences was above 20 000, the total number of images that were used to compute convolutional neural network features was around 380 000.

2.3 Quantifying convolutional neural network feature transferability to tracking

It is not clear how one should compare the effectiveness of different representations for visual object tracking. However, using the convolutional neural network layer activations as features that are used to train an online classifier which discriminates object and its surrounding background is currently a popular and successful approach. For example, the work of [NH15], which recently won the VOT2015 tracking challenge [KML+15], follows a similar framework. They use pre-trained shared layers of a convolutional neural network, even though it has a simpler architecture than the ones used in this Thesis, to acquire a target representation and then follows a domain-specific layer, that is trained separately for each sequence, including the testing phase one. The domain specific layer acts as a classifier which is trained online to discriminate the object patches versus background patches.

Another relevant approach is that of [HYKH15]. Their algorithm starts with a collection of image samples that are propagated through a pre-trained convolutional neural network acquiring a representation, using which a SVM classifier classifies the sample as either a positive one, that is of the tracked object, or a negative one, that is belonging to the background.

As using an online trained Support Vector Machine (SVM) to discriminate object versus background seems to be popular more generally, SVM was chosen to be the classifier which uses the convolutional neural network features to learn to discriminate the object from its surrounding background. What is more, even though the chosen dataset might not be the best dataset to study multi-object tracking, as it consists of sequences of individual moving objects, it might be interesting to see how well a multi-class SVM can classify each of the 60 objects using the fully-connected layer activations from the five chosen convolutional neural networks as features.

To summarize, the following experiments have been conducted for this Thesis:

- Testing how well a Linear SVM is able to discriminate a tracked object from its background, varying the distance with which background samples are shifted relative to the true object location by the procedure described in the previous subsection, using the responses of the three fully-connected layers of the chosen five convolutional neural networks, which in total gives 15 different representations, as features. Such a test is three-dimensional, as, first of all, it tests how discriminability of the object from its background varies as background samples gradually move away from the true target location. Secondly, it checks how discriminability changes as you move further in the convolutional neural network, that is as you take the fully-connected layer responses closer to the output of the network. Thirdly, it provides information about which convolutional neural network architectures are more suited towards representing a tracked object with its fully-connected layer activations.

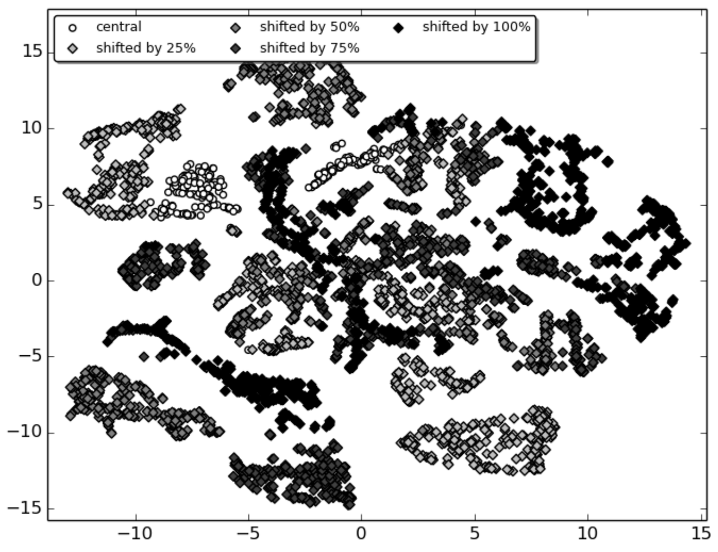
- Testing how well a Linear SVM is able to discriminate a tracked object from its background, where the background patches are either two times smaller or two times larger than the true object patch, using features similar to the description above.
- Testing how well a Linear SVM is able to discriminate a tracked object from other objects, using the 15 different representations.
- Additionally, as there are many ways to split the feature vectors into train and test sets for training and evaluating a Linear Support Vector Machine, several schemes have been chosen. First of all, using the whole feature vector set as training set was chosen to quantify how linearly separable are the vectors in the feature space, as defined by the particular convolutional neural network and its fully-connected layer. Using every odd vector as train sample and every even vector as test sample quantifies something related, however, it avoids using the same set for both training and testing. What is more, in the online tracking scenario, only the locations in the previous frames are known. To simulate such a scenario, some percentage of the first frames are used for training while another percentage of subsequent frames are used for testing.

3 Experiments

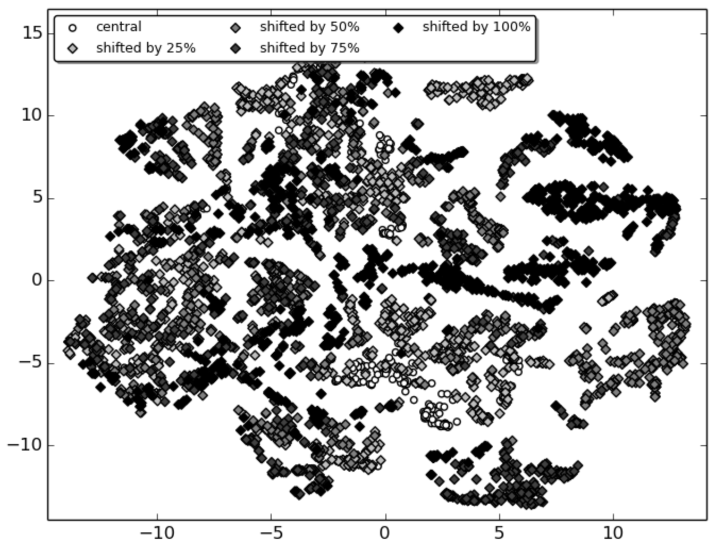
This section provides various results of the experiments, the methodology of which are explained in the previous section. The first subsection deals with the case of discriminating image patches containing a single tracked object from image patches that were shifted or scaled by the procedure described in the previous section. The second subsection provides results about the discriminability of image patches containing a single tracked object from image patches containing other tracked objects.

3.1 Discriminating objects from background

First of all, to visualize how different convolutional neural networks and their various fully-connected layers are able to discriminate objects from background, t-SNE [vdM13] plots were made. t-Distributed Stochastic Neighbor Embedding (t-SNE) is a technique for dimensionality reduction, which helps to visualize large dimensional datasets. In this work, the variant of t-SNE with Barnes-Hut approximations was used to speed up calculations. t-SNE plots for the sequence named "car1" and the CNN-F convolutional neural network are depicted below:



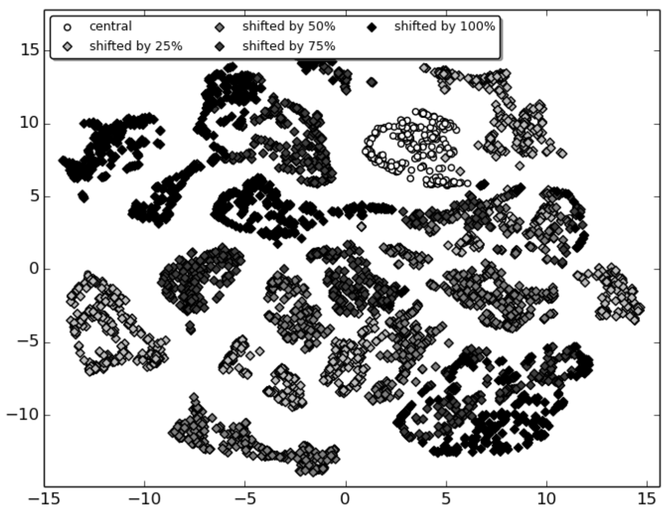
(a) fc6



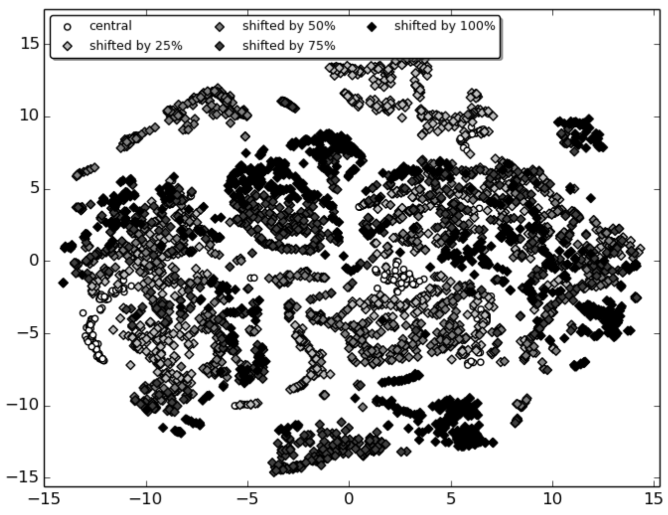
(b) fc8

Figure 6: t-SNE plots for the car1 sequence using CNN-F fully-connected layer activations as representations.

As we can see from the above, central image crops, which are the correctly localized images of the tracked object, and the shifted crops seem to cluster in a more structured way for the fc6 layer, as compared to the fc8 layer. What is more, the central crops, which are depicted as white circles, seem to be more separable from the shifted crops in the fc6 layer, as they seem to cluster in two localized clusters that do not overlap the shifted image representations. For comparison, t-SNE plots for the same sequence, but for the CNN-S convolutional neural network are depicted below:



(a) fc6

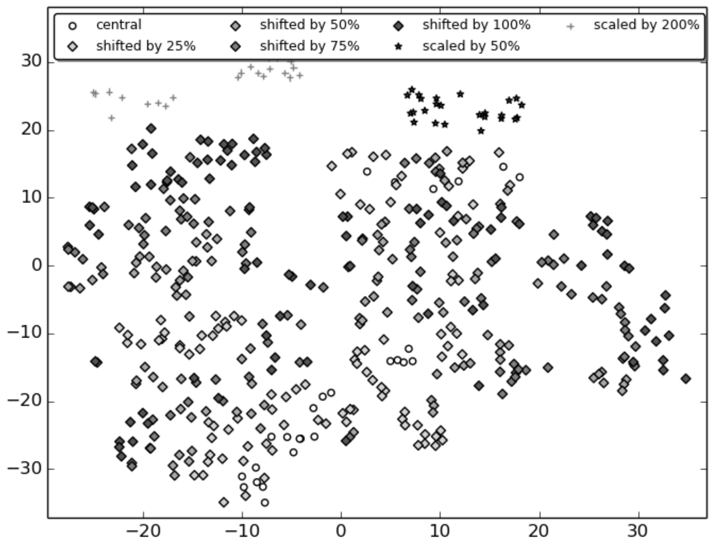


(b) fc8

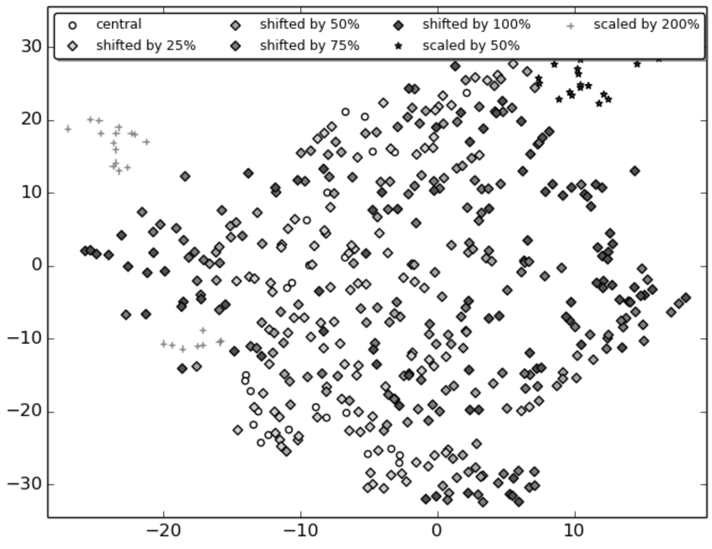
Figure 7: t-SNE plots for the car1 sequence using CNN-S fully-connected layer activations as representations.

As it is visible from the above plots, once again fc6 layer seems to provide representations that are better suited for discriminating object images from background images. What is more, the CNN-M fc6 representation only makes a single cluster for the correctly localized object patches, as compared to the two clusters of CNN-F fc6 representation, which might be important for being able to track an object based on this representation.

Secondly, similar t-SNE plots for a different object, but with additional negative crops, that are centered on the true object location while being scaled either 0.5 or 2.0 times, are presented below. For a better visualization, only every 10th frame was used.



(a) fc6



(b) fc8

Figure 8: t-SNE plots for the sheep sequence using CNN-F fully-connected layer activations as representations.

We can see from the above figure that negative crops which are scaled versions of the true object patch seem to cluster together and isolate from the central patches more than the shifted crops, at least the ones which still partially overlap the tracked object. A similar trend was also seen for other object t-SNE visualizations, which might suggest that discriminating a tracked object from its scaled versions is relatively easier than discriminating it from slightly shifted versions, at least using the tracking framework analyzed in this Thesis.

Another trend, showing that discriminating an object from its scaled version by a factor of 0.5 is easier than discriminating it from a similarly scaled version, but with a factor of 2.0, may be visible in the below t-SNE plot, which was acquired using the fc7 feature of the CNN-M network for a sequence called "graduate":

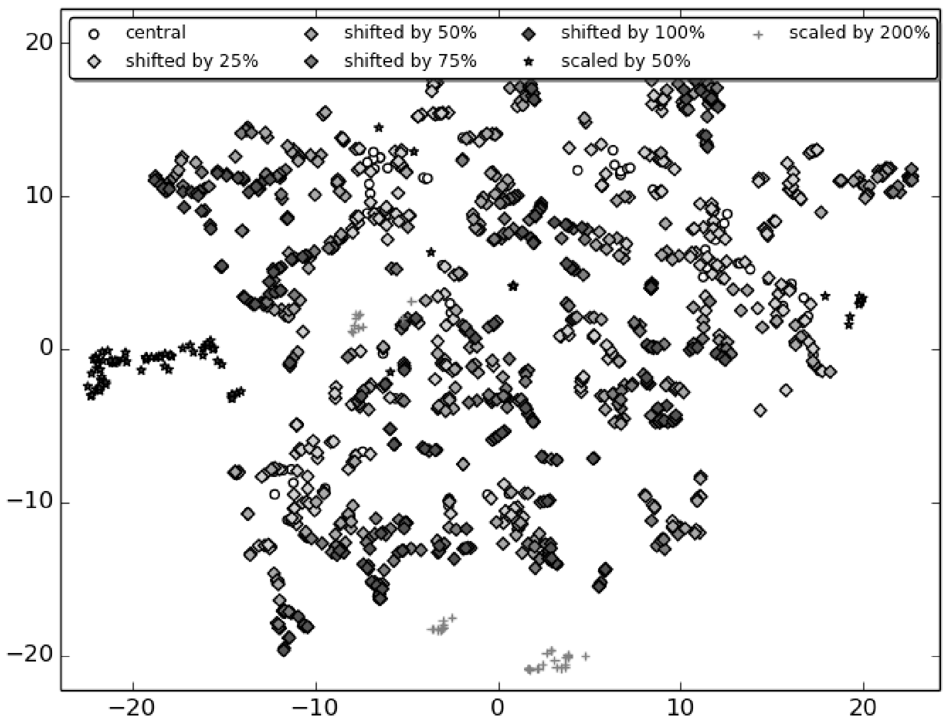
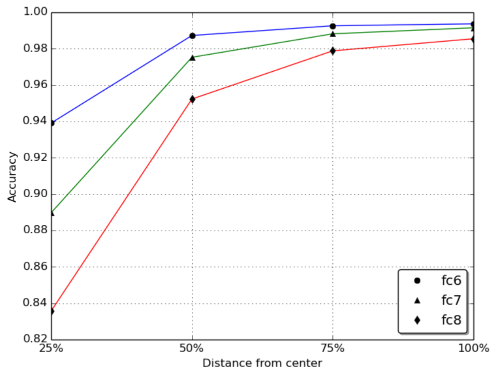


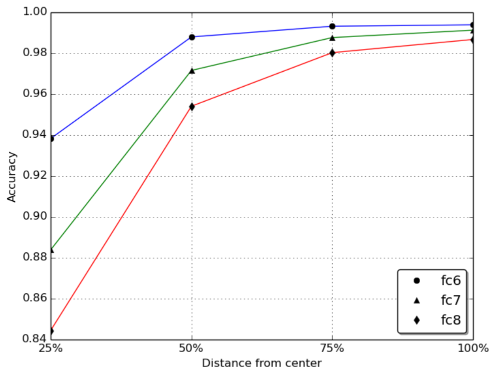
Figure 9: t-SNE plots for the graduate sequence using CNN-M fully-connected layer activations as representations.

Previously depicted figures showing t-SNE projections do seem to suggest a few hypotheses, however, interpreting them is quite subjective and some more objective criteria are necessary. As was mentioned in the previous section, using a Linear Support Vector Machine (SVM) to separate objects from the background is quite popular in the tracking literature. For this reason, the classification accuracies of a SVM were used to quantify how effective a representation is for visual object tracking.

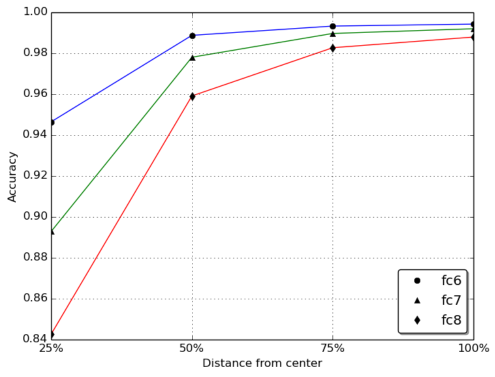
First of all, all of the samples were used as training data and the training accuracy was used as a quantification of effectiveness. Training accuracy quantifies how linearly separable are the representations of the object and background. As was mentioned in the previous section, this analysis will be three dimensional. The first dimension is how the SVM accuracy depends on the distance by which the background crops are shifted relative to the true object location. The distances are quantified by percentages of height or width of the object, depending on whether the object was shifted up/down or left/right, respectively. How SVM training accuracy, averaged for all the object sequences, depends on this distance for the convolutional neural network fully-connected layer representations is depicted below:



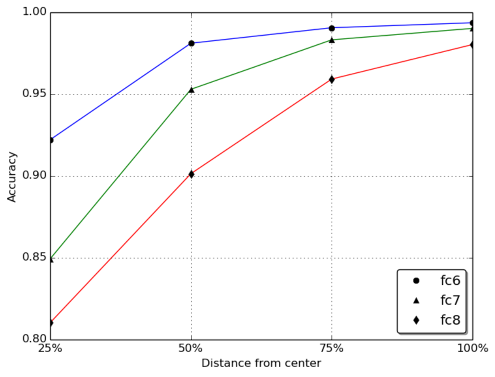
(a) CNN-F



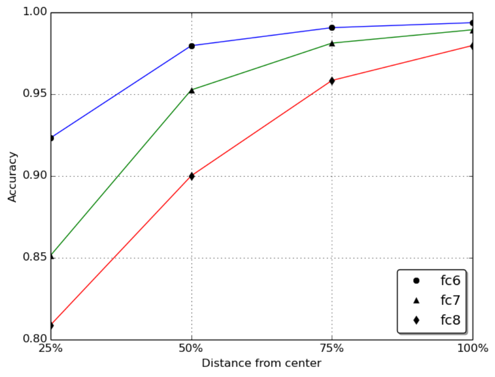
(b) CNN-M



(c) CNN-S



(d) CNN-D

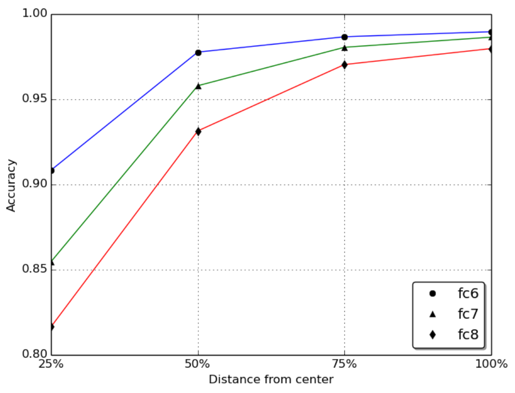


(e) CNN-E

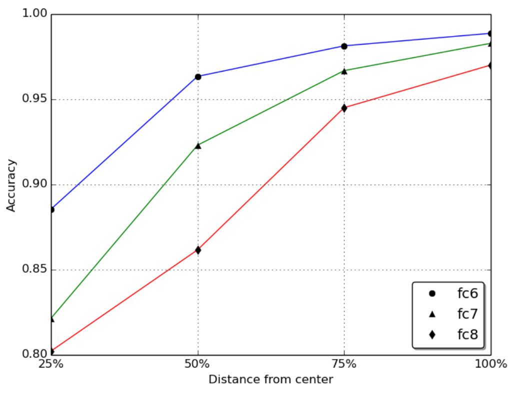
Figure 10: SVM classification training accuracy dependance on background image patch distances to the true object location.

As we see from the above figures, for all the convolutional neural network architectures the pattern is similar - as the distance increases, the training accuracy moves from, depending on the layer, about 85% for background patches that are only shifted by 25% of rectangle height or width value and hence still include a large portion of the object to around 98-99% for patches that do not include the object at all. The accuracies seem to be quite high, which might indicate that in the fully-connected feature space, the problem is almost linearly separable, at least for the data in question. What is more, even though the difference in accuracies between different layer representations is relatively high for 25% distance, it seems to diminish as the classification problem becomes easier and accuracies start to approach 100%.

Secondly, instead of using all the data for SVM training and reporting the training accuracy, another approach was used. In this approach, half of the data, that is every even frame, was used for training and the rest of the data, that is every odd frame, was used for testing. As the general pattern seems to remain similar, except that the accuracies now become lower, only results for CNN-F and CNN-E are depicted:



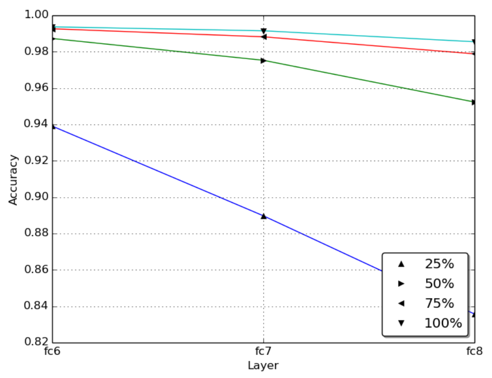
(a) CNN-F



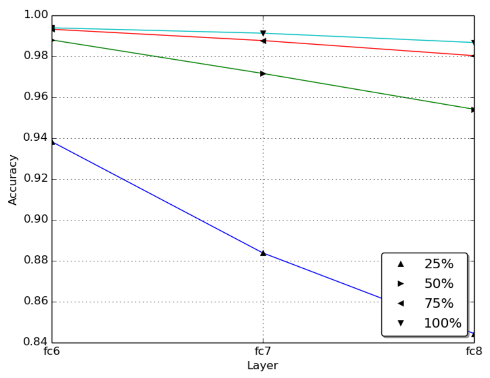
(b) CNN-E

Figure 11: SVM classification testing accuracy dependance on background image patch distances to the true object location, using half of the data for training.

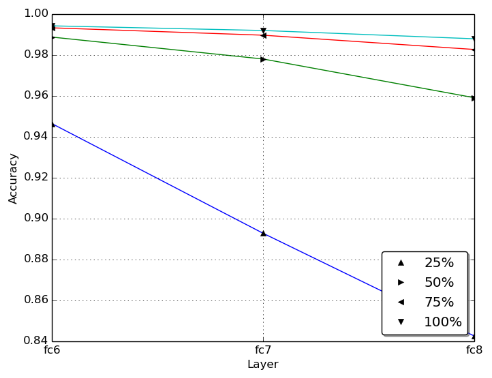
As for the second dimension of these experiments, it was tested how the SVM accuracies depend on which fully-connected layer was used as the representation. Even though the above figures seem to already suggest the answer, below are the results for all the chosen convolutional neural network architectures:



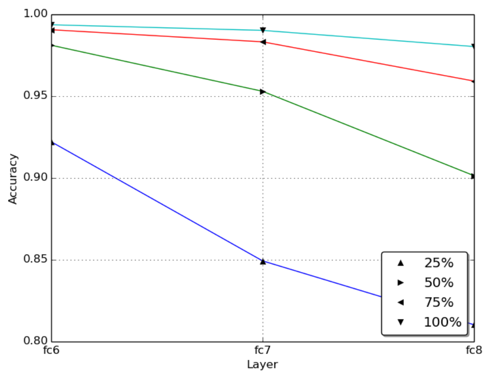
(a) CNN-F



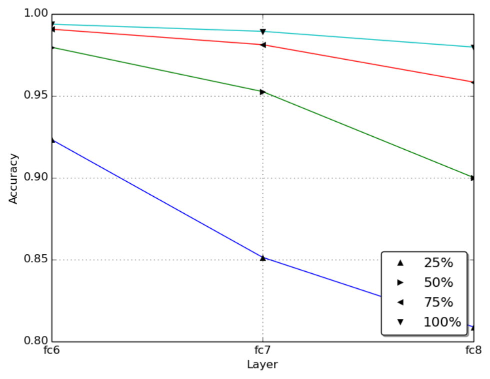
(b) CNN-M



(c) CNN-S



(d) CNN-D



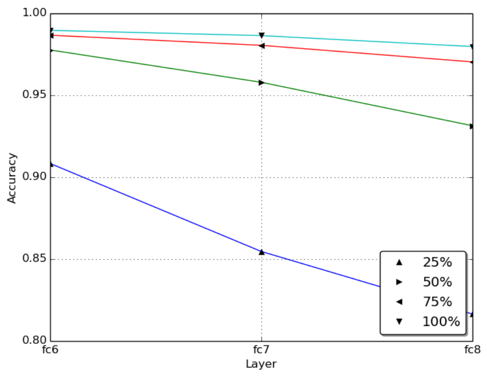
(e) CNN-E

Figure 12: SVM classification training accuracy dependance on which fully-connected layer was chosen as the representation.

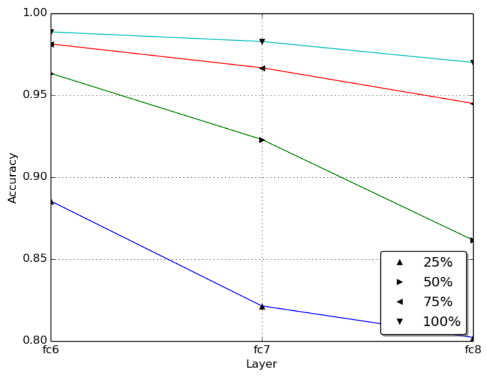
The above figures again demonstrate that the fully-connected layers that are closer to the output are less effective for discriminating an object from the background around him. This might be because the layers fc7 and fc8 are too specialized towards classification, the original problem on which the convolutional neural networks were trained on.

What is more, fc8 is of more than 4 times lower dimensionality than that of fc6 and fc7, which might additionally explain the large performance drop when going from fc7 to fc8.

Once again as the general pattern seems to remain the same when using only half of the data for training and using the other half for testing, only the results for CNN-F and CNN-E are shown:



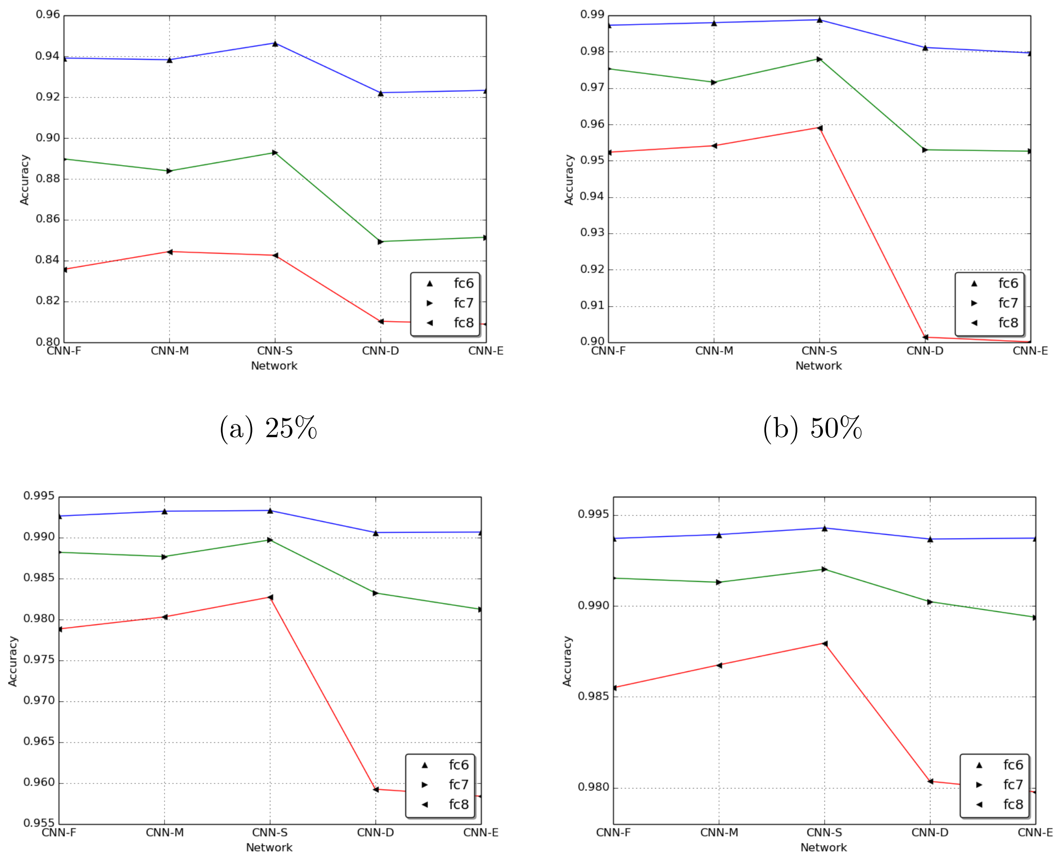
(a) CNN-F



(b) CNN-E

Figure 13: SVM classification testing accuracy dependance on which fully-connected layer was chosen as the representation, using half of the data for training.

The third dimension of these experiments is that of comparing different convolutional neural network architectures. Plots that show how accuracies change as we vary the architecture in the order of their top-5 classification accuracy on the ILSRVC 2012 validation dataset [RDS$^+$15] are shown below:



(a) 25% (b) 50%

(c) 75% (d) 100%

Figure 14: SVM classification training accuracy dependance on the convolutional neural network architecture for different distances between the object and background patches, quantified as percentages of the object rectangle width or height.

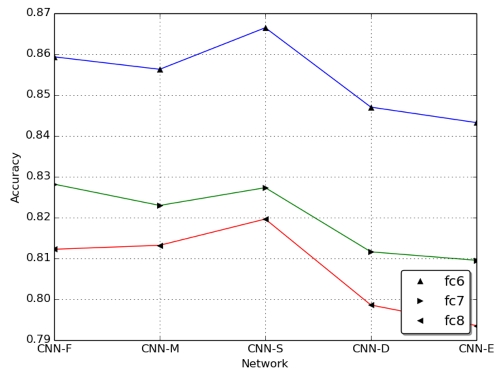
As we can see from the above figures, the first three convolutional neural network architectures seem to provide better representations, especially for the fc8 layer, even though the last two architectures have much better accuracies on the task on which the networks were trained. This might be explained by the fact that the last two architectures are also much deeper, so the representations that they provide at their fully-connected layers are too specialized for the object classification task and they do not generalize too well to this new task of discriminating object from shifted patches around the object.

Accuracy plots generated using the approach that only uses half of the data for training provided similarly looking results and they are not presented here.

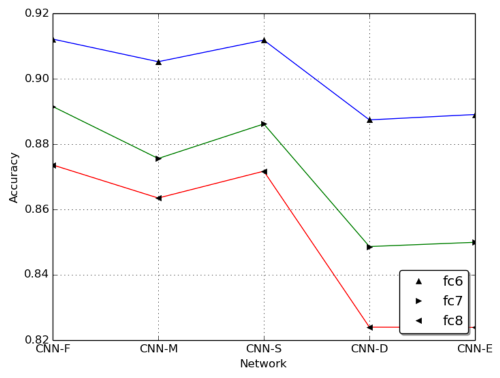
Lastly, as was mentioned in the previous section, in the online tracking scenario only the location in some number of previous frames is known, where the number of frames is usually as small as one while it is necessary to predict the object location in the subsequent frames. To simulate such a scenario at least to some degree, the following train and test sets for the Linear SVM were used:

1) Training on a small number of first frames, which was set to be 10 percent and testing on a small number of subsequent frames, which was also 10 percent of the total frames for the object sequence.

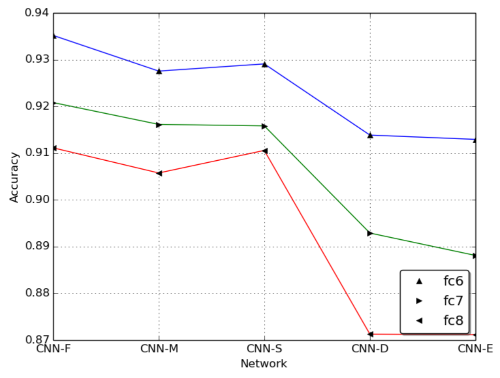
First of all, the figures below depict Linear SVM testing accuracies for discriminating between the true object location patches and variously translated patches, but keeping the scale fixed:



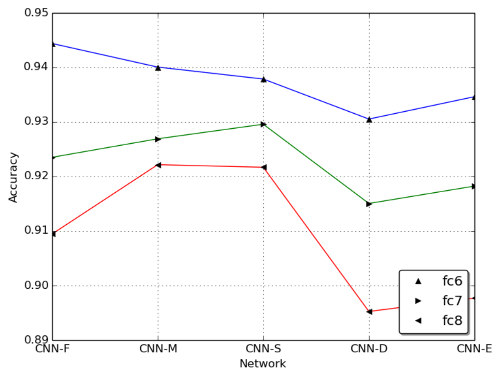
(a) 25%



(b) 50%



(c) 75%

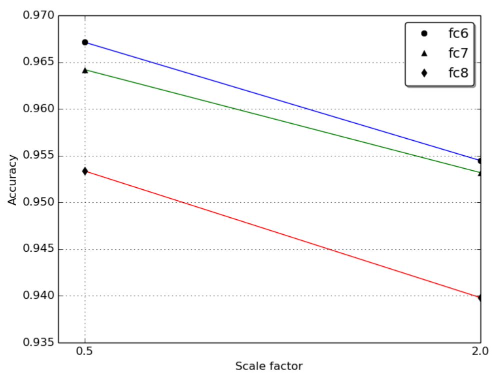


(d) 100%

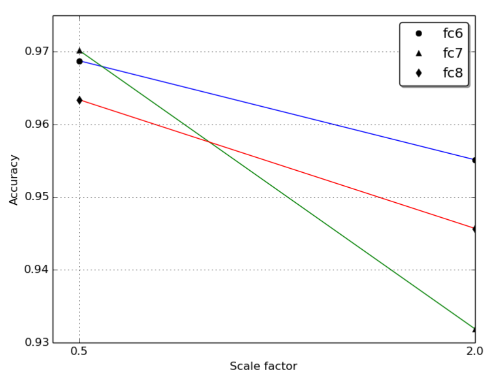
Figure 15: SVM classification testing accuracy dependance on the convolutional neural network architecture for different distances between the object and background patches, quantified as percentages of the object rectangle width or height.

Uninterestingly, the trends are somewhat similar to the ones seen above.

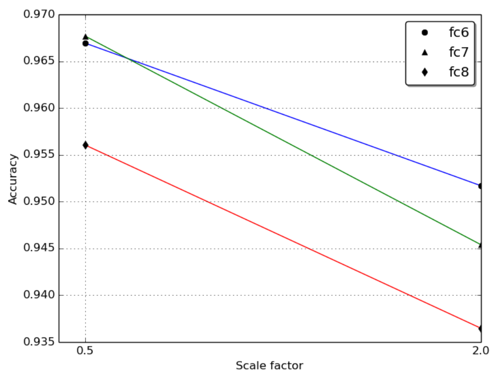
Secondly, the figures below depict Linear SVM testing accuracies for discriminating between the true object location patches and patches scaled by a factor of either 0.5 or 2.0:



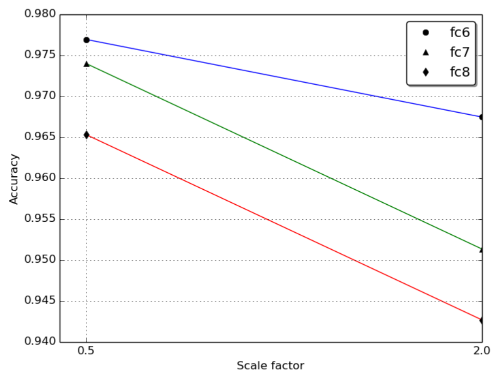
(a) CNN-F



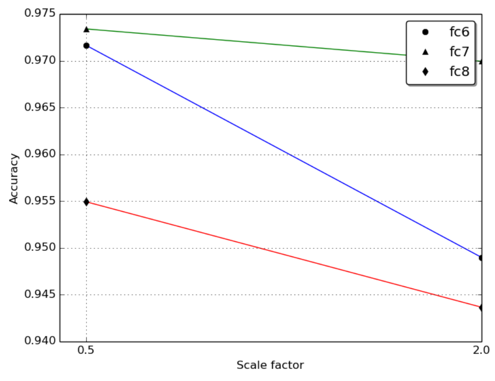
(b) CNN-M



(c) CNN-S



(d) CNN-D



(e) CNN-E

Figure 16: SVM classification testing accuracy dependance on the scale factor by which negative patches were scaled for different convolutional neural network architectures and their fully-connected layers.

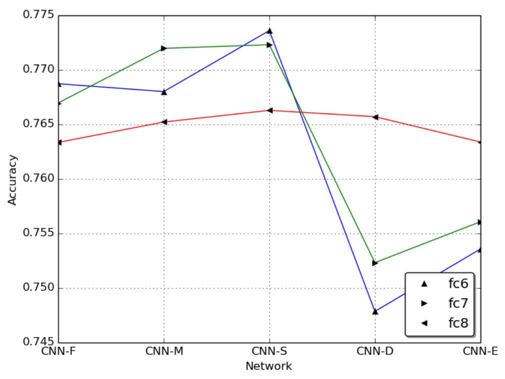
Firstly, we can see that as was suggested by the t-SNE plots, patches that are scaled by a factor of 0.5 do seem to be easier to distinguish from the true object patches than are the patches scaled by a factor of 2.0. This may be the result of the original dataset on which the networks were trained on having images of objects that either take up most of the image or are only centered in the middle. Training on such images may have resulted in features that are partially invariant to scaling by a factor of 2.0 and this shows up in reduced accuracies in the above plots.

Secondly, the overall accuracies seem to be higher than in the case of discriminating against translated patches. This was also suggested by the t-SNE plots made in the previous subsection.

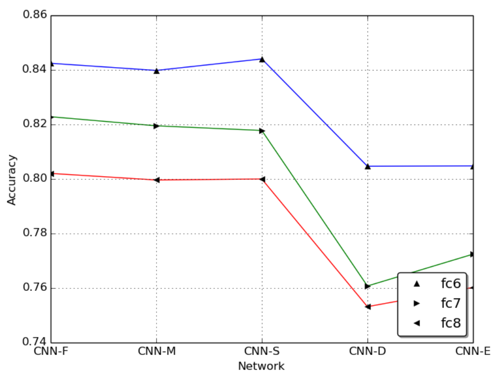
Thirdly, even though fc6 remains the most accurate representation for most of the networks, for two of them fc7 is actually more accurate for the case of the scale factor of 2.0. This might show that fc6 and fc7 might be more effective in different scenarios and could complement each other, especially when computing fc7 is cheap from a computational perspective.

2) Training on a small number of first frames, which was set to be 10 percent and testing on a large number of subsequent frames, which was set to 50 percent of the total frames for the object sequence.

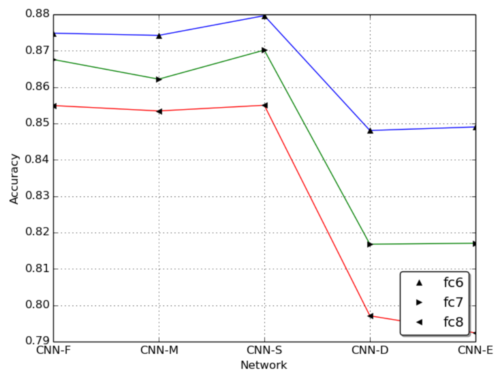
First of all, the figures below depict Linear SVM testing accuracies for discriminating between the true object location patches and variously translated patches, but keeping the scale fixed:



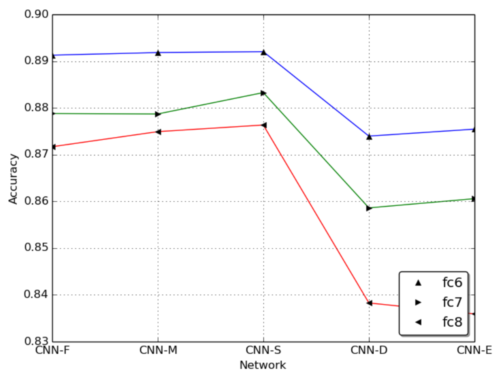
(a) 25%



(b) 50%



(c) 75%

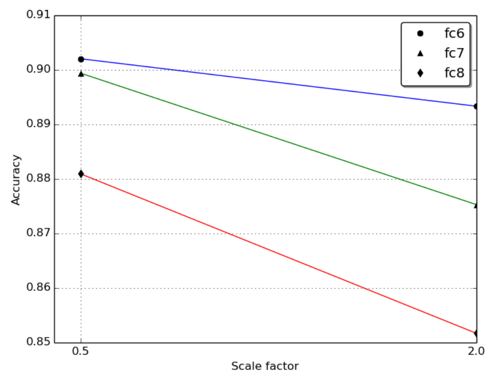


(d) 100%

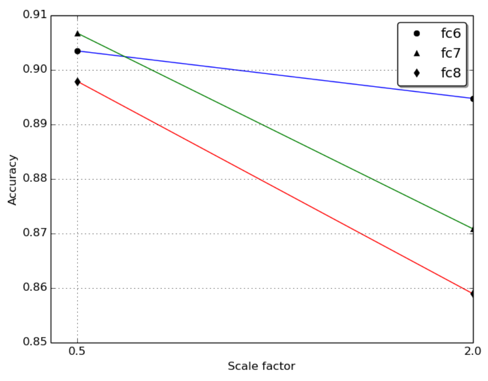
Figure 17: SVM classification testing accuracy dependance on the convolutional neural network architecture for different distances between the object and background patches, quantified as percentages of the object rectangle width or height.

As we can see from the above, the trends are mostly similar to the ones seen above. However, there are some interesting variations seen in the figure (a) which again suggest that different feature representations could complement each other.

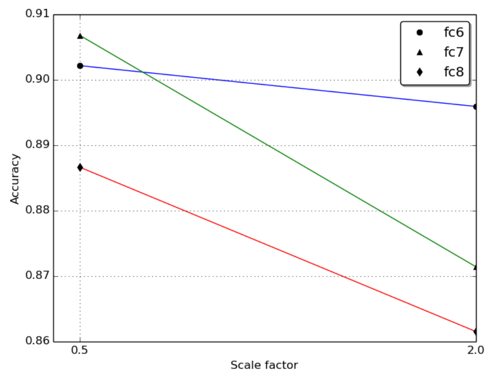
Secondly, the figures below depict Linear SVM testing accuracies for discriminating between the true object location patches and patches scaled by a factor of either 0.5 or 2.0:



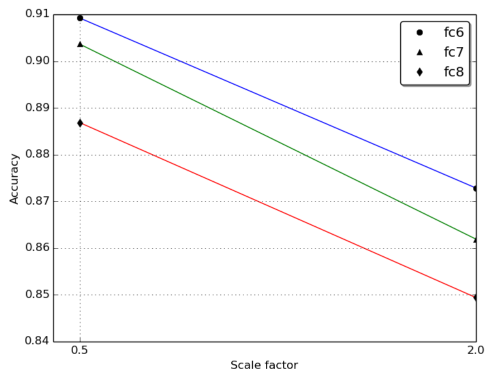
(a) CNN-F



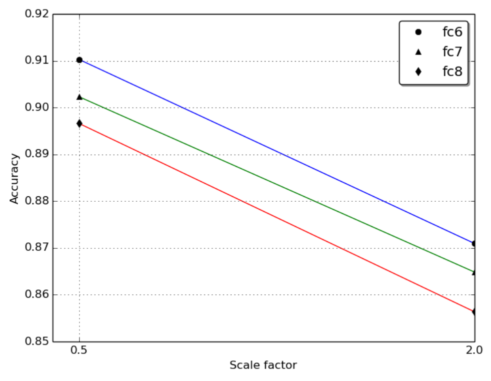
(b) CNN-M



(c) CNN-S



(d) CNN-D



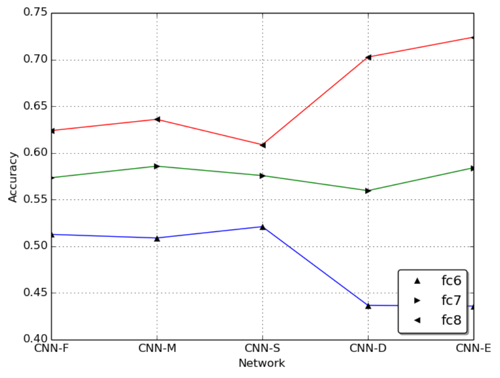
(e) CNN-E

Figure 18: SVM classification testing accuracy dependance on the scale factor by which negative patches were scaled for different convolutional neural network architectures and their fully-connected layers.

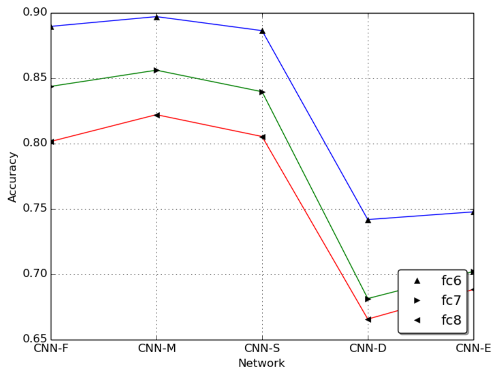
Here we see tendencies that are similar to the ones seen before.

3) Training on a large number of first frames, which was set to be 50 percent and testing on a small number of subsequent frames, which was set to 10 percent of the total frames for the object sequence.

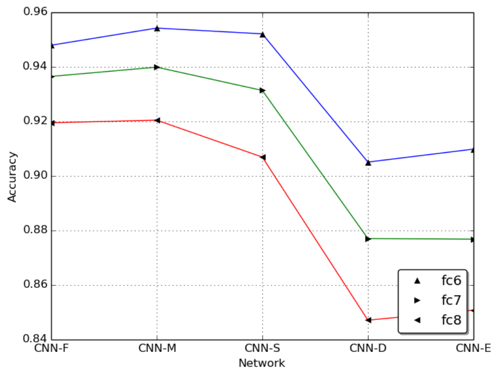
First of all, the figures below depict Linear SVM testing accuracies for discriminating between the true object location patches and variously translated patches, but keeping the scale fixed:



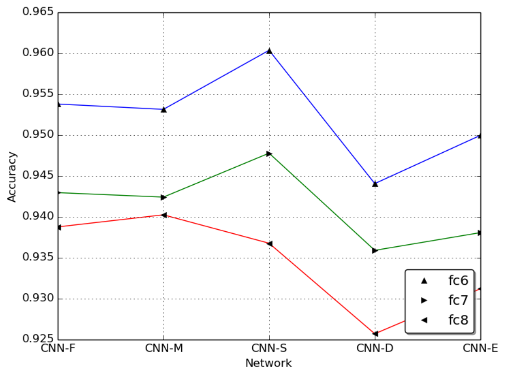
(a) 25%



(b) 50%



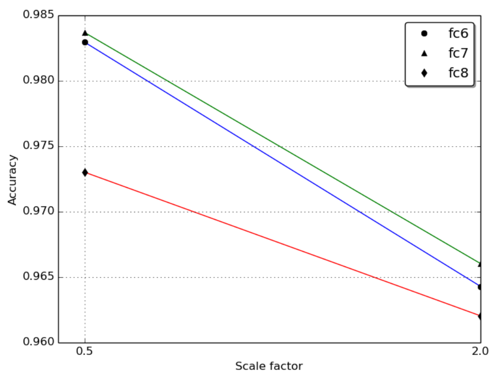
(c) 75%



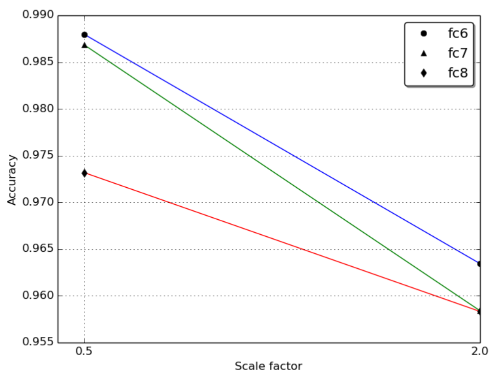
(d) 100%

Figure 19: SVM classification testing accuracy dependance on the convolutional neural network architecture for different distances between the object and background patches, quantified as percentages of the object rectangle width or height.

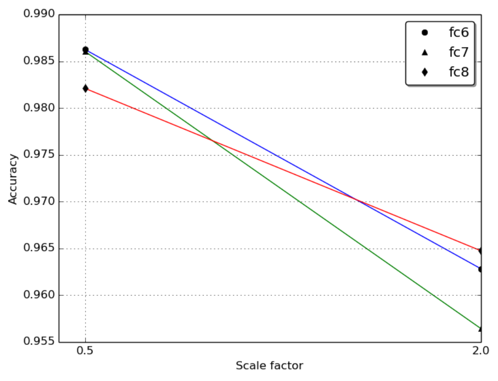
Secondly, the figures below depict Linear SVM testing accuracies for discriminating between the true object location patches and patches scaled by a factor of either 0.5 or 2.0:



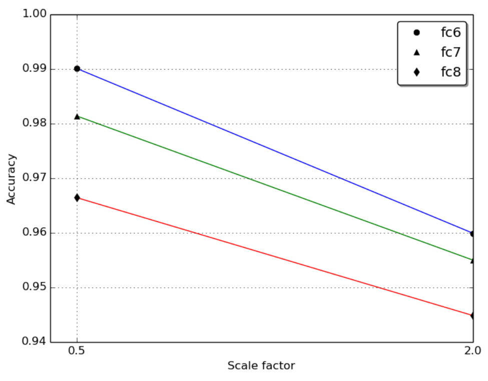
(a) CNN-F



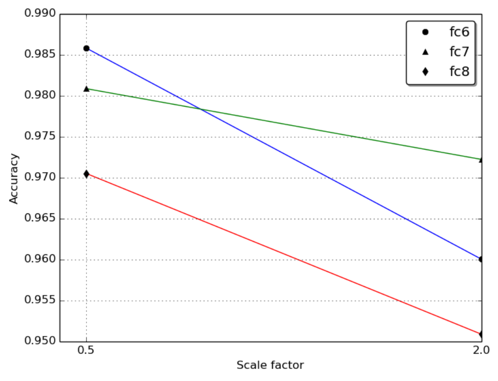
(b) CNN-M



(c) CNN-S



(d) CNN-D



(e) CNN-E

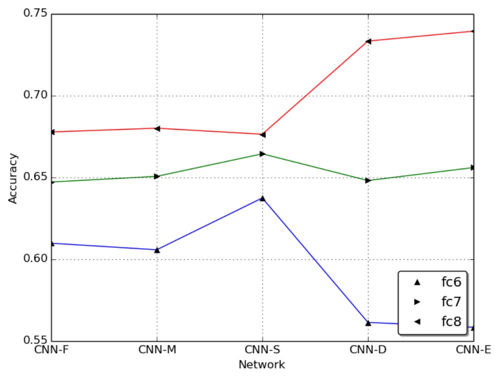
Figure 20: SVM classification testing accuracy dependance on the scale factor by which negative patches were scaled for different convolutional neural network architectures and their fully-connected layers.

In the figure above, the subfigure (c) shows something new - fc8 achieved the highest accuracy, even though it was usually the inferior representation in the scenarios already presented.

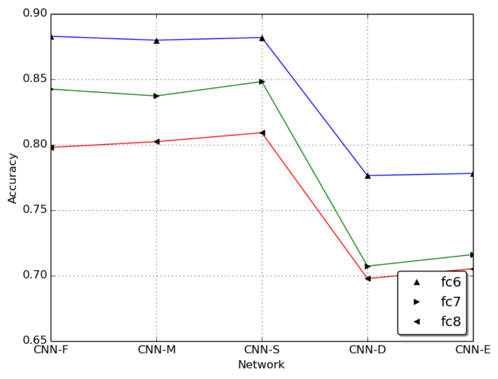
4) Training on a large number of first frames, which was set to be 50 percent

and testing on a large number of subsequent frames, which was also 50 percent of the total frames for the object sequence.

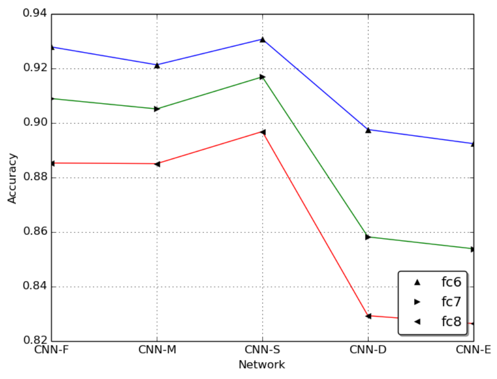
First of all, the figures below depict Linear SVM testing accuracies for discriminating between the true object location patches and variously translated patches, but keeping the scale fixed:



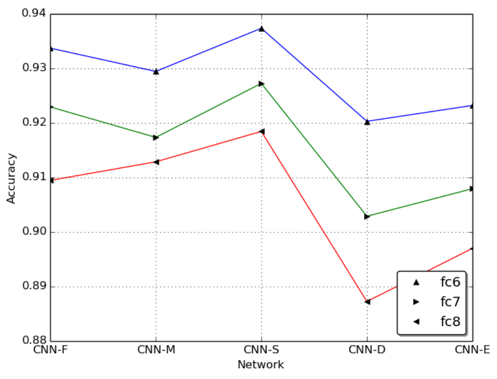
(a) 25%



(b) 50%



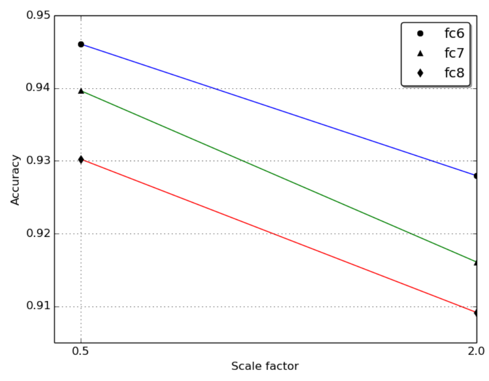
(c) 75%



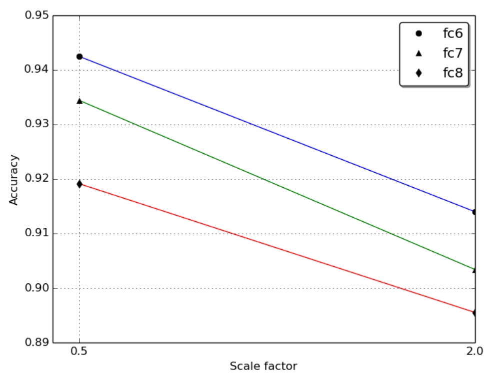
(d) 100%

Figure 21: SVM classification testing accuracy dependance on the convolutional neural network architecture for different distances between the object and background patches, quantified as percentages of the object rectangle width or height.

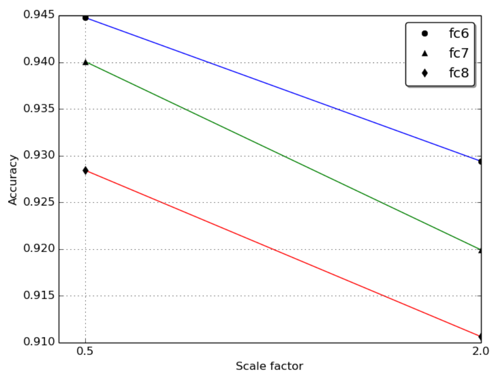
Secondly, the figures below depict Linear SVM testing accuracies for discriminating between the true object location patches and patches scaled by a factor of either 0.5 or 2.0:



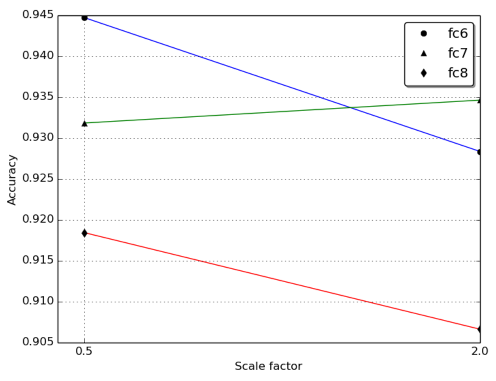
(a) CNN-F



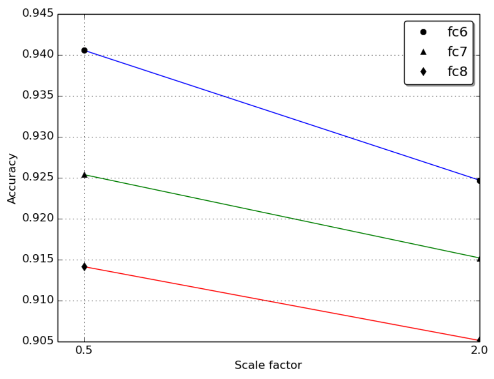
(b) CNN-M



(c) CNN-S



(d) CNN-D



(e) CNN-E

Figure 22: SVM classification testing accuracy dependance on the scale factor by which negative patches were scaled for different convolutional neural network architectures and their fully-connected layers.

3.2 Discriminating objects from other objects

As it was mentioned before, the dataset that was used for experiments is better suited for single object tracking. However, it might still be useful to see how the 15 different representations compare in this respect - how effective they are at representing an object so it would be easy to discriminate it against other objects.

As in the previous section, we start with presenting t-SNE plots to visually see how different representations are able to cluster together instances of the same object. To speed up computations only every 10th frame was used for every object, so in total around 2000 points will be drawn. Below is the t-SNE plot for CNN-F fc6 representation:

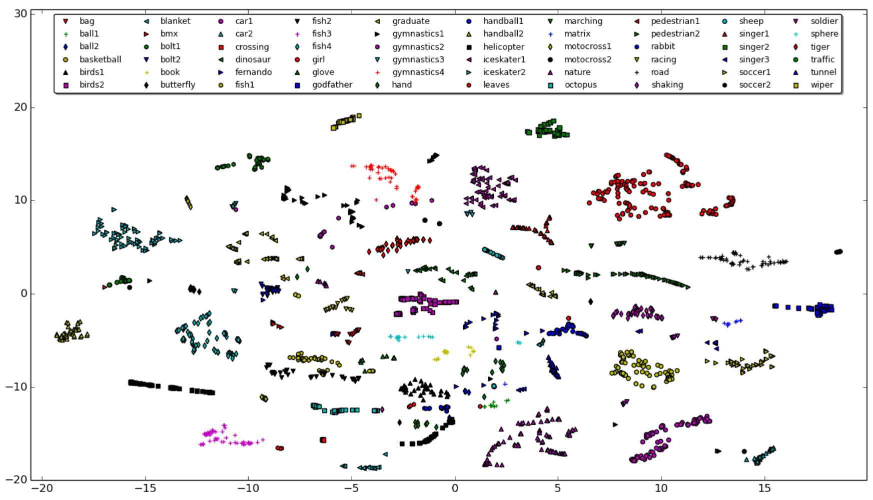


Figure 23: t-SNE plot of all the object sequences for CNN-F fc6 representation. Legend shows names of the sequences used.

As we can see from above, different object sequences do separate from each other, even though there are some overlaps. For comparison, the figure below depicts similar t-SNE plot for the same convolutional neural network, but for its fc8 representation:

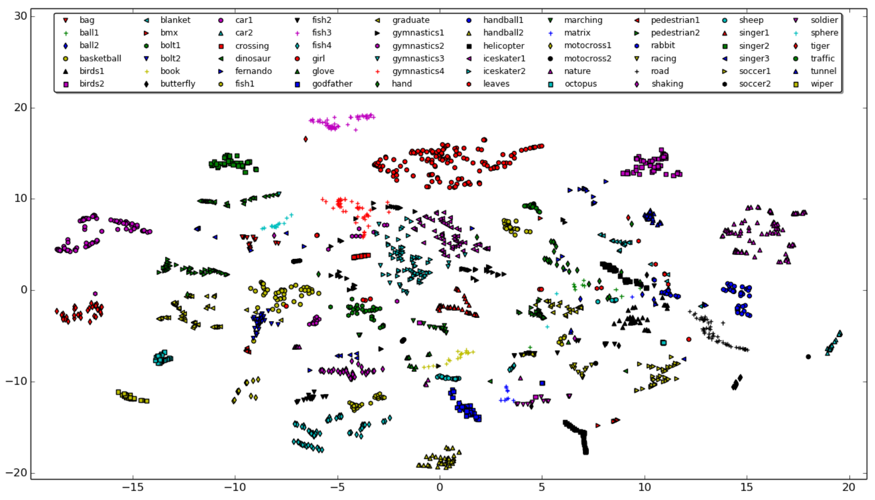
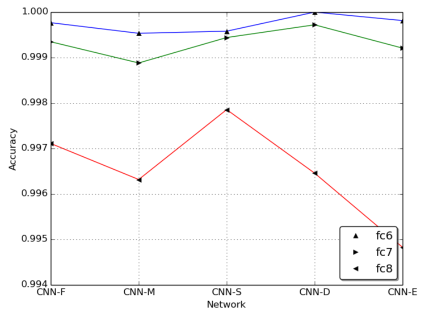


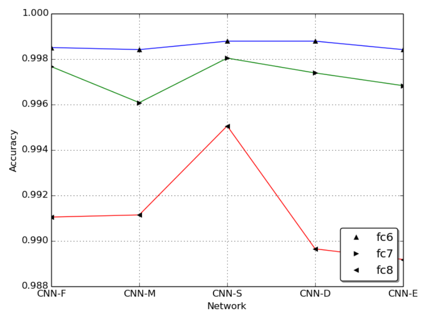
Figure 24: t-SNE plot of all the object sequences for CNN-F fc8 representation. Legend shows names of the sequences used.

Here we see much more overlap between different objects, for example between hand and helicopter sequences, which is a bit surprising, as these classes are very different semantically while fc8 layer should be able to capture semantic properties of images.

Now, using a similar approach as was used before, we look at the SVM training accuracy when varying the convolutional neural network architecture used for representing object images. This is depicted in the two figures below:



(a) Training accuracy

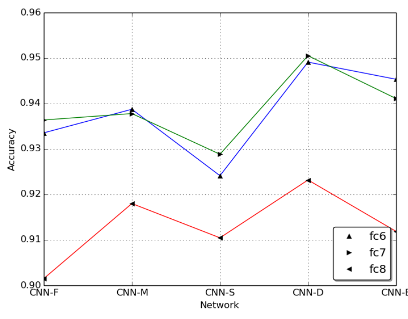


(b) Testing accuracy, when trained on half of the data

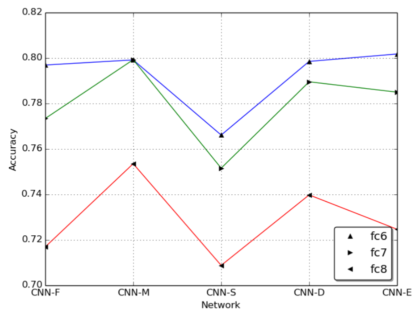
Figure 25: SVM classification accuracy of different tracked objects using various representations.

As we can see from the above two figures, the last two architectures, which are the most accurate on the original task on which the networks were trained, seem to provide inferior results for fc8 layers. However, differently than in the case of discriminating object from its background, fc6 representation provided by these two networks seems to be more effective than provided by other architectures, or at least as effective. One possible explanation for this might be that discriminating an object from other objects is very close to the original problem of object classification, hence the learned features transfer more efficiently.

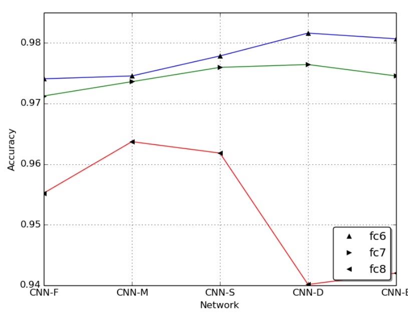
Additionally, similarly as in the section above, different training and testing splits was used to simulate slightly different online tracking scenarios:



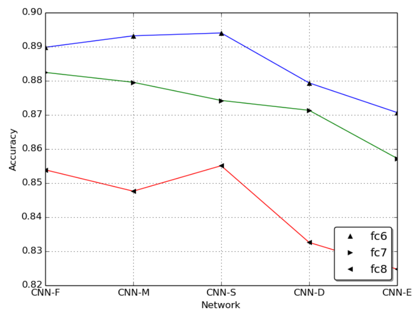
(a) Testing on 10% of the data, trained on 10% of the data



(b) Testing on 50% of the data, trained on 10% of the data



(c) Testing on 10% of the data, trained on 50% of the data



(d) Testing on 50% of the data, trained on 50% of the data

Figure 26: SVM classification accuracy of different tracked objects using various representations and different training and testing sets.

As we can see from the subfigures (c) and (d) above, deeper architectures, namely CNN-D and CNN-E, perform better than shallow architectures when tested on a relatively small amount of future frames while they are worse when tested on a relatively large amount. This might indicate that two different representations could be used for effective tracking - one for long term tracking and one for short-term tracking.

3.3 Summary of experimental results

A relatively large number of experiments have been conducted for this Thesis, comparing different architectures, layers and other factors for convolutional neural network feature transferability.

First of all, a comparison of different convolutional neural network architectures has shown that the first three architectures that are shallower than the second two are actually more effective for visual tracking, even though they were less accurate for the original problem on which they were trained on. This may suggest that using simpler and even less accurate architectures may be better when transferring between different tasks.

Secondly, the features coming from the fc6 layer were usually the most effective for the majority of the scenarios considered, even though for some of them fc7 and fc8 were more efficient. This may suggest that features from different layers could complement each other in a tracking method that is based on convolutional neural networks.

Thirdly, it seemed that discriminating the true object location from its translated variants was harder than discriminating it from its scaled variants. This might suggest using denser sampling for various locations as compared to various scales when designing a tracking method based on convolutional neural networks.

Lastly, when comparing between different test and train splits for training a Linear SVM and when considering the multi-object tracking scenario, where the goal is to discriminate between different objects instead of one object against its background, it has been found that the deeper architectures are more effective in some scenarios. This might suggest using different architectures or representations for short-term and long-term tracking.

Conclusion

The goal of this Thesis was to provide valuable insights and recommendations for methods that apply convolutional neural networks to visual object tracking. Several tasks have been completed.

First of all, a methodology was designed to quantify feature transferability. Tracking was reduced to a classification problem, that of discriminating an object from its background, which is a popular and a successful approach. Fully-connected layers of five different convolutional neural network architectures, trained on an object classification task, were chosen as a representation using which linear classifiers were trained on some subset of the frames for a tracked object.

Secondly, using the above methodology many experiments were conducted and qualitative, namely t-SNE plots, and quantitative, that were Linear Support Vector Machine testing accuracies averaged for all the objects in the used dataset, measures have been acquired.

Both the qualitative and quantitative measures do indicate that convolutional neural network features are indeed successfully transferable to the visual object tracking task, as they provide relatively high, in the range from 60 to 99, classification accuracies when the tracking problem is reduced to a classification problem. The variation in these accuracies was provided for a variety of important tracking scenarios, revealing strengths and weaknesses of different fully-connected layers and different convolutional neural network architectures.

Lastly, using the measures acquired from experiments and analyzing their results, several recommendations for the design of methods using convolutional neural networks have been proposed:

- Using simpler and even less accurate (on the original task) architectures may be better when transferring to the visual object tracking task, as the architecture is less specialized to the task on which it was trained on and the features are more generalizable to other tasks as a result
- Fully-connected layers closer to the input are usually more effective for representing a tracked object. Layers close to the network output are probably overspecialized for the original task

- Discriminating an object from its translated patches is harder than discriminating it from its scaled variants. This might be explained by the fact that convolutional neural networks for object classification are specifically designed to be translation-invariant and hence discriminating an object from its slightly shifted version is especially hard. Some architectural modifications may be required to alleviate this issue
- There is a difference in which architecture is more effective when testing on a large or a small amount of future frames. This may be related to the fact that different representations are required for short-term and long-term tracking

References

[BCV12] Yoshua Bengio, Aaron C. Courville, and Pascal Vincent. Unsupervised feature learning and deep learning: A review and new perspectives. *CoRR*, abs/1206.5538, 2012.

[Ben09] Yoshua Bengio. Learning deep architectures for AI. *Found. Trends Mach. Learn.*, 2(1):1–127, January 2009.

[BFL$^+$11] Loris Bazzani, Nando Freitas, Hugo Larochelle, Vittorio Murino, and Jo-Anne Ting. Learning attentional policies for tracking and recognition in video with deep networks. In Lise Getoor and Tobias Scheffer, editors, *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, ICML '11, pages 937–944, New York, NY, USA, June 2011. ACM.

[CSVZ14] Ken Chatfield, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Return of the devil in the details: Delving deep into convolutional nets. *CoRR*, abs/1405.3531, 2014.

[DJV$^+$13] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. *CoRR*, abs/1310.1531, 2013.

[HXHZ15] Fan Hu, Gui-Song Xia, Jingwen Hu, and Liangpei Zhang. Transferring deep convolutional neural networks for the scene classification of high-resolution remote sensing imagery. *Remote Sensing*, 7(11):14680, 2015.

[HYKH15] Seunghoon Hong, Tackgeun You, Suha Kwak, and Bohyung Han. Online tracking by learning discriminative saliency map with convolutional neural network. *CoRR*, abs/1502.06796, 2015.

[JDB$^+$13] Jonghoon Jin, A. Dundar, J. Bates, C. Farabet, and E. Culurciello. Tracking with deep neural networks. In *Information Sciences and Systems (CISS), 2013 47th Annual Conference on*, pages 1–5, March 2013.

[JSD$^+$14] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[KLL15] Jason Kuen, Kian Ming Lim, and Chin Poo Lee. Self-taught learning of a deep invariant representation for visual tracking via temporal slowness principle. *Pattern Recognition*, 48(10):2964 – 2982, 2015. Discriminative Feature Learning from Big Data for Visual Recognition.

[KML$^+$15] Matej Kristan, Jiri Matas, Aleš Leonardis, Michael Felsberg, Luka Čehovin, Gustavo Fernandez, Tomas Vojir, Gustav Häger, Georg Nebehay, Roman Pflugfelder, Abhinav Gupta, Adel Bibi, Alan Lukežič, Alvaro Garcia-Martin, Amir Saffari, Alfredo Petrosino, Andres Solis Montero, Anton Varfolomieiev, Atilla Baskurt, Baojun Zhao, Bernard Ghanem, Brais Martinez, ByeongJu Lee, Bohyung Han, Chaohui Wang, Christophe Garcia, Chunyuan Zhang, Cordelia Schmid, Dacheng Tao, Daijin Kim, Dafei Huang, Danil Prokhorov, Dawei Du, Dit-Yan Yeung, Eraldo Ribeiro, Fahad Shahbaz Khan, Fatih Porikli, Filiz Bunyak, Gao Zhu, Guna Seetharaman, Hilke Kieritz, Hing Tuen Yau, Hongdong Li, Honggang Qi, Horst Bischof, Horst Possegger, Hyemin Lee, Hyeonseob Nam, Ivan Bogun, Jae chan Jeong, Jae il Cho, Jae-Yeong Lee, Jianke Zhu, Jianping Shi, Jiatong Li, Jiaya Jia, Jiayi Feng, Jin Gao, Jin Young Choi, Ji-Wan Kim, Jochen Lang, Jose M. Martinez, Jongwon Choi, Junliang Xing, Kai Xue, Kannappan Palaniappan, Karel Lebeda, Karteek Alahari, Ke Gao, Kimin Yun, Kin Hong Wong, Lei Luo, Liang Ma, Lipeng Ke, Longyin Wen, Luca Bertinetto, Mahdieh Pootschi, Mario Maresca, Martin Danelljan, Mei Wen, Mengdan Zhang, Michael Arens, Michel Valstar, Ming Tang, Ming-Ching Chang, Muhammad Haris Khan, Nana Fan, Naiyan Wang, Ondrej Miksik, Philip H S Torr, Qiang Wang, Rafael Martin-Nieto, Rengarajan Pelapur, Richard Bowden, Robert Laganiere, Salma Moujtahid, Sam Hare, Simon Hadfield, Siwei Lyu, Siyi Li, Song-Chun Zhu, Stefan Becker, Stefan Duffner, Stephen L Hicks, Stuart Golodetz, Sunglok Choi, Tianfu Wu, Thomas Mauthner, Tony Pridmore, Weiming Hu, Wolfgang Hübner, Xiaomeng Wang, Xin Li, Xinchu Shi, Xu Zhao, Xue Mei, Yao Shizeng, Yang Hua, Yang Li, Yang Lu, Yuezun Li, Zhaoyun Chen, Zehua Huang, Zhe Chen, Zhe

Zhang, and Zhenyu He. The visual object tracking vot2015 challenge results, Dec 2015.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with Deep Convolutional Neural Networks. In F. Pereira, C.J.C. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[LBH15] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521:436–444, May 2015.

[LeC12] Yann LeCun. Learning invariant feature hierarchies. In Andrea Fusiello, Vittorio Murino, and Rita Cucchiara, editors, *Computer Vision – ECCV 2012. Workshops and Demonstrations*, volume 7583 of *Lecture Notes in Computer Science*, pages 496–505. Springer Berlin Heidelberg, 2012.

[LLP15] Hanxi Li, Yi Li, and Fatih Porikli. Robust online visual tracking with a single convolutional neural network. In Daniel Cremers, Ian Reid, Hideo Saito, and Ming-Hsuan Yang, editors, *Computer Vision – ACCV 2014*, volume 9007 of *Lecture Notes in Computer Science*, pages 194–209. Springer International Publishing, 2015.

[MHYY15] Chao Ma, Jia-Bin Huang, Xiaokang Yang, and Ming-Hsuan Yang. Hierarchical convolutional features for visual tracking. In *Proceedings of the IEEE International Conference on Computer Vision*, 2015.

[NH15] Hyeonseob Nam and Bohyung Han. Learning multi-domain convolutional neural networks for visual tracking. *CoRR*, abs/1510.07945, 2015.

[RDS$^+$15] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[SEZ$^+$13] Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *CoRR*, abs/1312.6229, 2013.

[SZ14] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[TFF08] A. Torralba, R. Fergus, and W.T. Freeman. 80 million tiny images: A large data set for nonparametric object and scene recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 30(11):1958–1970, Nov 2008.

[vdM13] Laurens van der Maaten. Barnes-hut-sne. *CoRR*, abs/1301.3342, 2013.

[VLL$^+$10] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion. *J. Mach. Learn. Res.*, 11:3371–3408, December 2010.

[WLGY15] Naiyan Wang, Siyi Li, Abhinav Gupta, and Dit-Yan Yeung. Transferring rich feature hierarchies for robust visual tracking. *CoRR*, abs/1501.04587, 2015.

[WLY13] Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang. Online Object Tracking: A Benchmark. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 2411–2418, June 2013.

[WOWL15] Lijun Wang, Wanli Ouyang, Xiaogang Wang, and Huchuan Lu. Visual tracking with fully convolutional networks. In *IEEE International Conference on Computer Vision (ICCV)*, 2015.

[WY13] Naiyan Wang and Dit-Yan Yeung. Learning a deep compact image representation for visual tracking. In C.J.C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 809–817. Curran Associates, Inc., 2013.

[YCBL14] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? *CoRR*, abs/1411.1792, 2014.

[YOM06] A. Yilmaz, O.Javed, and M.Shah. Object tracking: A survey. *ACM Computing Surveys (CSUR)*, 38, 2006.

[ZF13a] Matthew D. Zeiler and Rob Fergus. Stochastic pooling for regularization of deep convolutional neural networks. *CoRR*, abs/1301.3557, 2013.

[ZF13b] Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. *CoRR*, abs/1311.2901, 2013.