High Performance Soccer Player Tracking System with Convolutional Neural Networks

University of Massachusetts Amherst
COMPSCI 682 Neural Networks: A Modern Introduction
Amherst, MA

Abstract

Visual object tracking has many practical applications in sports, such as tracking motion trajectory, recording player's speed and helping make strategies. However, modern soccer has become a very fast-paced affair and with players who possess blistering pace and acceleration to volt forward. Considering that the video streams have a higher frame rate, achieving top tracking performance over real-time speed is considered as one of the main challenges. In this paper, we aim to implement a comprehensive soccer player tracking system which can run at frame-rates beyond real-time, and is easy to use in practice.

We equipped the tracker with a state-of-the-art Siamese region proposal network (Siamese-RPN), and trained the network end-to-end off-line with Youtube-BB dataset. In the inference phase, the tracking task can be defined as a local one-shot detection task for a seed up. In order to optimize the feature extractor network in the model, we tested and evaluated different backbones on the public dataset VOT2021. Experiments illustrate that ResNet-50 serving as a backbone in the Siamese-RPN tracking model actually outperforms the modified AlexNet on several aspects, including expected average overlap(EAO), accuracy and robustness. A comprehensive soccer player tracking system also need to manage interaction. As shown in Fig. 1, we used the Tkinter for front-end design, which is one of the most popular graphical user interface(GUI) libraries in Python. Finally, by providing a tracking demo, we can demonstrate that our soccer player tracker runs effectively at least 60 FPS frame rate in a soccer game video stream.

1. Introduction

The problem of visual object tracking comprehensively includes a variety of important techniques, such as image processing, pattern recognition and deep learning. In practice, we can see various practical visual object tracking tasks like autonomous self-driving [3, 5].

However, several per-frame visual attributes can be very challenging in practice, such as target out-of-frame, partial occlusion, aspect change, size change, fast motion, similar objects, out-of-plane rotation, etc. For instance, in a real soccer game, all the teammates will be regarded as similar

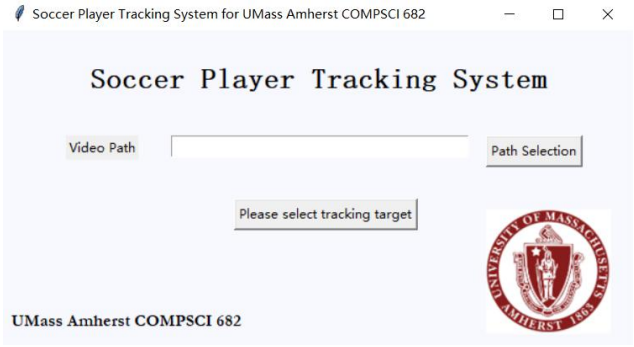

Figure 1: The front-end GUI for the Soccer Player Tracking System was designed by the Tkinter Python library.

objects for each other due to the same soccer jersey. What's more, there always exists partial occlusion of target player, especially when the center back players try to defense. In particular, the high frame rate of video stream the tracker can correctly conduct at is important too.

One of the popular tracking strategies is using correlation filters. A few correlation filter based methods use deep features to track targets through complex motions. But the model update of these technics is not simple [1, 12]. And another tracking strategy uses strong deep features and do not update the model. However, in [2, 7, 11], it was found that they cannot achieve a good result as correlation filter based methods do.

In this paper, we will implement a soccer player tracking system with convolutional neural networks, and which can achieve a top performance while operating at high speed comparing to correlation filters method. In order to design a suitable network architecture for our soccer player tracker, we did research on regular Siamese-FC [7], CFNet [16], MDNet [10], ATOM [15]. Finally, we equipped tracking algorithms with Siamese-RPN network [18]. The main architecture of Siamese-RPN includes two subareas. The first is feature extraction subnetwork, and the second is proposal generation subnetwork.

In the inference phase, we can interpret the tracking task as a local one-shot detection task [4]. One-shot learning aims to learn information about object categories from one, or only a few training images. And also, we adopted two

strategies to make the framework available for a high-speed tracking task. The first proposal selection strategy is that, if one anchor generates a bounding box too far away from the center, we will adopt to discard it. The second strategy is that introducing proper penalty and using cosine window to help determine the best proposal.

Here we analyze and highlight the importance of various state-of-the-art convolutional neural networks serving as a backbone in object tracking models. The backbone of convolutional neural network refers to the feature extractor network. The selection of backbone is obtained by adding different backbones to the first subnetwork Siamese and evaluating them on public dataset.

Recently, several attempts have been made to implement Siamese-RPN like SenseTime team [17]. However, the principal aims of our project can be summarized as four folds: 1) Implement a comprehensive soccer player tracking system (powered by the Pytorch) which is easy to use in practice, including the tracker design and GUI design. 2) Perform the whole mathematic process of proposal selection strategies for the special local one-shot detection task. 3) Evaluate and compare the state-of-the-art CNNs serving as a backbone for object tracking task in experiments, and select a suitable backbone for our tracker. 4) Provide a tracking demo for functional testing.

2. Related Works

Considering the main contributions of this paper, the background literature review will include: Siamese architecture, one-shot learning and convolutional neural network backbones.

2.1. Siamese Architecture

Several attempts have been made in object tracking area to improve accuracy and speed of Siamese network based trackers [2, 19, 7, 11, 20].

There are two branches in the Siamese network. As shown in Fig. 2, z is the template branch, and x is the detection branch. Siamese networks apply an identical transformation ϕ to both inputs and then combine their representations using another function g, that is:

$$f(z, x) = g(\phi(z), \phi(x)) \qquad (1)$$

Fully-convolutional Siamese architecture improves the accuracy by introducing the correlation layer as fusion tensor. This architecture is fully convolutional with respect to the search image x. The output is a scalar-valued score map whose dimension depends on the size of the search image. This enables the similarity function to be computed for all translated sub-windows within the search image in one evaluation. For instance, as shown in Fig. 2, the sub-windows in the score map correspond to the similarities of orange and green pixels.

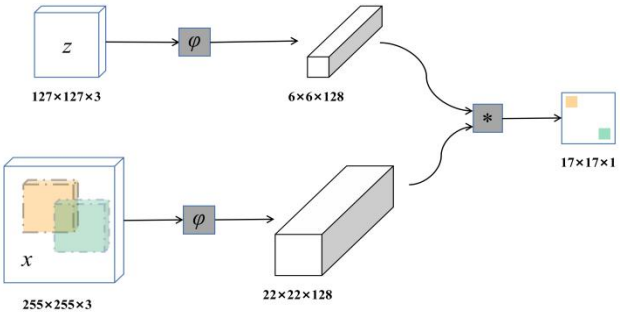

Figure 2: Main architecture of fully-convolutional Siamese. In this example, the sub-windows in the score map correspond to the similarities of orange and green pixels.

However, comparing to correlation filter based tracking methods, trackers based on Siamese-FC [7], CFNet [16], MDNet [10], ATOM [15] can be improved with respect to both accuracy and robustness.

2.2. One-shot Learning

One-shot learning aims to learn information about object categories from one or only a few training images. Learning to learn is a challenging topic, where we should find a technic to embed the category information in the learner. Deep embeddings for online trackers can actually be a topic for future research.

In recent years, several meta-learning based methods [13, 14] have been proposed to solve problem in deep learning. But most of approaches are pertain to classification task rather than tracking task. We will detail how to interpret the tracking task as one-shot detection task in Section 3.4.

2.3. Convolutional Neural Network Backbones

The backbone of convolutional neural network refers to the feature extractor network. Popular convolutional neural network backbones include VGG, DarkNet, AlexNet, ResNet [6].

Firstly, VGG-16 consists of 13 convolutional and 3 fully connected layers with Rectified Linear Units (ReLUs) activation. VGG-16 is generally used in Fast R-CNN [21], Faster R-CNN [9], SSD [22] and HyperNet [30]. Secondly, Darknet-19 consists of 19 convolutional layers and 5 max-pooling layers [31]. It effectively reduces the number of parameters by using 3×3 convolutional filters and several 1×1 filters.

A regular AlexNet composed of 8 layers, where 5 are convolutional and 3 are fully connected. A ResNet [7] are mainly consisting of convolutional and identity blocks. ResNet-50 is composed of 26 million parameters. In this paper, we will evaluate a modified AlexNet and the ResNet-50 in the following sections.

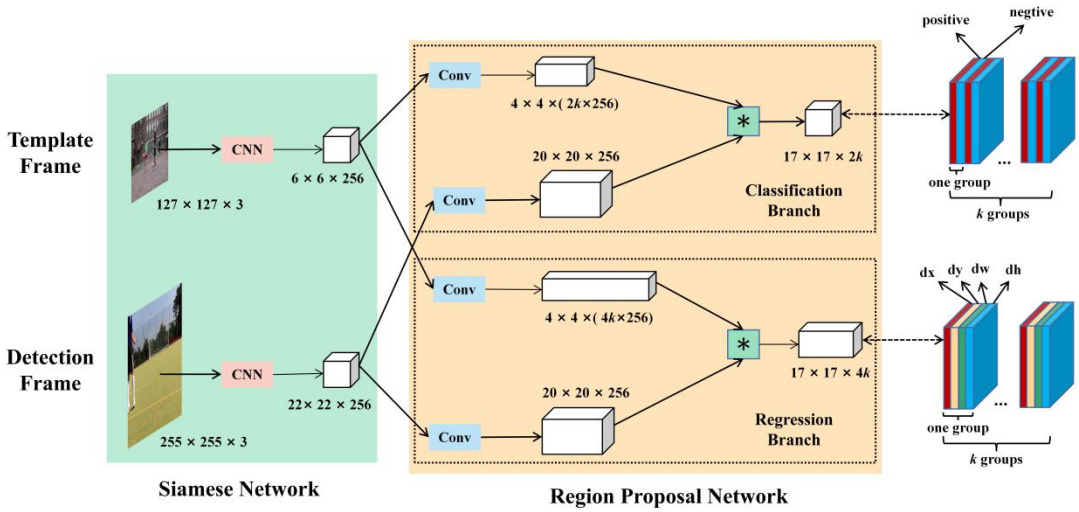

Figure 3: Main architecture of Siamese-RPN which mainly contains two subareas. The first is feature extraction area called Siamese Network, and the other is proposal generation area called Region Proposal Network. Note that $*$ indicates the convolution operation.

3. Approach

This section details the framework of our tracking system. The main disadvantage of the regular Siamese-FC is making much effort to test multiple scale. One idea is adding bounding box regression. As shown in Fig. 3, we are applying an state-of-the-art Siamese-RPN framework to our soccer player tracker [18]. The Siamese-RPN mainly contains two subareas. The first is feature extraction area called Siamese, and the other is proposal generation area called RPN. Hence, the methods mainly include four aspects as follows: Siamese subnetwork, region proposal subnetwork, perform one-shot detection and proposal selection.

3.1. Siamese Subnetwork

The feature extraction module is actually same as regular Siamese-FC, where the translation operator is:

$$(L_\tau x)[u] = x[u - \tau] \qquad (2)$$

Note that this is a fully convolution framework without paddings, and the stride is k:

$$h(L_{k\tau} x) = L_\tau h(x) \qquad (3)$$

Similarly, there are two branches in the Siamese network. The template branch receives target patch in the historical frame as input, and the detection branch receives target patch in the current frame as input. The two branches share parameters in CNN so that the two patches are implicitly encoded by the same transformation which is suitable for the subsequent tasks. Siamese resizes the template patch to $127 \times 127 \times 3$ centering on the historical frame, and also resizes the detection patch to $255 \times 255 \times 3$ in a similar way. Then after going through the CNN block, a $6 \times 6 \times 256$ feature map and a $22 \times 22 \times 256$ feature map will be obtained from the template patch and the detection patch respectively.

In this paper, considering the experiment results of various backbone in section 4.3, we use the ResNet-50 in Siamese subnetwork rather than modified AlexNet.

3.2. Region Proposal Subnetwork

In Faster R-CNN [9], the region proposal module is first introduced. There are two sections in RPN. One is called pair-wise correlation section, and the other is called supervision section. RPN can extracting more precise proposals due to the supervision of both foreground-background classification and bounding box regression.

As shown in Fig. 3, what's different from the regular Siamese-FC is that, the feature maps $\phi(x)$ and $\phi(z)$ obtained from the previous Siamese subnetwork will be split into two branches, and used as the input for a conv layer in each branch. Then if we suppose k is the number of anchors, it will output 2k channels and 4k channels for classification and regression respectively. We can compute the correlation as follows:

$$\begin{aligned} Anc^{cls}_{w\times h\times 2k} &= [\phi(x)]_{cls} * [\phi(z)]_{cls} \\ Anc^{reg}_{w\times h\times 4k} &= [\phi(x)]_{reg} * [\phi(z)]_{reg} \end{aligned} \qquad (4)$$

Note that $*$ indicates the concolution opertation.

In classification branch, the $\phi(z)$ from template branch is regarded as kernal. In the 17×17 feature map, each point has a 2k channel vector indicating negative and positive activation of each anchor at corresponding foreground or

background on original map. And the loss function we employ is a regular cross-entropy loss:

$$L_{cls} = -\log\left(\frac{e^{f_{y_i}}}{\sum_j e^{f_j}}\right) \tag{5}$$

Similarly, in regression branch, the $\phi(z)$ from template branch is also regarded as kernal. In the 17×17 feature map, each point has a 4k channel vector which indicates dx, dy, dw, dh. Those values are used for computing the final proposal later, that is distance between corresponding groundtruth and anchor. Specifically, the normalized distance between groundtruth boxes and the anchor boxes can be computed as:

$$\delta[0] = \frac{Tru_x - Anc_x}{Anc_w},$$

$$\delta[1] = \frac{Tru_y - Anc_y}{Anc_h}, \tag{6}$$

$$\delta[2] = ln\left(\frac{Tru_w}{Anc_w}\right), \quad \delta[3] = ln\left(\frac{Tru_h}{Anc_h}\right)$$

And the loss function we employ is a smooth L_1 loss:

$$L_{reg} = \sum_{i=0}^{3} smooth_{L_1}(\delta[i], \sigma)$$

$$= \sum_{i=0}^{3} \begin{cases} \frac{1}{2}\sigma^2\delta[i]^2, & |\delta[i]| < \frac{1}{\sigma^2} \\ |\delta[i]| - \frac{1}{2\sigma^2}, & otherwise \end{cases} \tag{7}$$

On balance, we combine the loss for classification and regression, so the final loss function can be:

$$L = L_{cls} + \lambda L_{reg} \tag{8}$$

3.3. Perform One-shot Detection

First of all, we can interpret the tracking task as a local one-shot detection task from [4]. We give the formulation straightforward here:

$$\min_W \frac{1}{n}\sum_{i=1}^{n} L(\, g(\varphi(x_i; W);\ \varphi(z_i; W)), l_i\,) \tag{9}$$

Note that function φ indicates the first subnetwork Siamse for feature extraction. Function g indicates the subnetwork RPN. l_i indicates labels corresponds to detection patch x_i. Now we aim to find the best W which can minimize the average loss L of predictor function.

In particular, the learning to learn process can be interpreted as: on the one hand, the template branch in Siamese area becomes the training parameters, which embeds the category information into the kernel. On the other hand, the detection branch performs detection using the embedded information, that is the outputs from template branches.

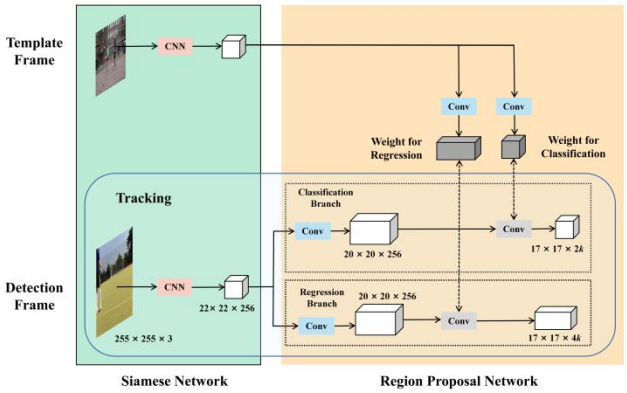

Figure 4: An example of defining the original tracking task as an one-shot detection task. During inference phase, template branch receives the target patch from the initial frame.

In the training process, the pairwise bounding box is the only supervision we consider. During inference phase, template branch receives the target patch from the initial frame. We can get a high speed since kernels have been pre-computed on the first frame and fixed throughout the tracking process. Now the template branch can be pruned and only the detection branch is retained. The detection branch performs online inference. So our tracking task can be interpreted as one-shot detection in other frames as shown in Fig. 4.

We can get the top M proposals from the output of RPN subnetwork by passing forward on the detection branch. Notice that we collect the top K proposals in classification output $Anc^{cls}_{w\times h\times 2k}$ rather than a single top point, since in the next section, we are further introducing the proposal selection strategy to for the specific tracking task.

3.4. Proposal Selection

Here we introduce the proposal strategy to make the framework available for the high-speed tracking task.

On the one hand, considering the current frame and the next frame, we assume that it is impossible that tracking objects are very far away from each other between the frames. Hence, if one anchor generates a bounding box too far away from the center, we will adopt to discard it.

On the other hand, it is always effective to introduce proper penalty to help determine the best proposal. The penalty will be imposed with respect to both large change in size and large displacement by using a cosine window. We define r as the height- width ratio of proposal, which can be obtained as follows:

$$r = \frac{height}{width} \tag{10}$$

And let h and w be the height and width of the target. p indicates the padding. We define s as the overall scale of the proposal, which can be obtained as follows:

$$(w + p) \times (h + p) = s^2 \tag{11}$$

Then the penalty can be obtained by following equation:

$$|\Delta| = e^{k * max(\frac{r_t}{r_{t-1}}, \frac{r_{t-1}}{r_t}) * max(\frac{s_t}{s_{t-1}}, \frac{s_{t-1}}{s_t})} \tag{12}$$

Note that r_{t-1} and s_{t-1} indicate the results of last frame. Finally, after multiplying scores by the corresponding penalty, the top K proposals in classification output will be re-ordered. We can get the bounding box of the tracking object by using Non-maximum-suppression (NMS). Meanwhile, the size of target should be updated .

4. Experiments

Recently, several attempts have been made to implement Siamese-RPN like SenseTime team [17]. However, the principal aims of our experiments can be summarized as two folds: 1) Evaluate the state-of-the-art CNNs serving as a backbone for object tracking task, and select a suitable backbone for our tracker. 2) Implement a comprehensive soccer player tracking system (powered by the Pytorch), including the tracker design, and GUI design.

4.1. Data

Notice that our tracking system actually runs with image pairs instead of continuous video streams, large-scale sparsely labelled videos are beneficial to the training process. For any input frame, according to the computing process defined in Eq. **11** and the size of target's bounding box, we can resize the template patch to $127 \times 127 \times 3$ centering on the historical frame, and also resize the detection patch to $255 \times 255 \times 3$ in a similar way. On balance, there are two popular public datasets we will use in the following experiments:

1. Youtube-BB: consists of more than 100,000 videos annotated once in every 30 frames [25]. We trained the Siamese-RNP network with different data set size by gradually adding more data from Youtube-BB.
2. VOT2021: consists of 60 sequences with the real-time video stream at least 30FPS. We used VOT for testing [26, 27, 28, 29].

4.2. Evaluation

We will detail two experiment results in the following sections. One is testing results of ResNet-50 and modified AlexNet serving as a backbone in the tracking model. The other is the tracking demo of our soccer player tracker for functional testing. Hence, the evaluation metrics for those two experiments are given respectively as follows:

1. Backbone Selection Experiment: the overall performance of different backbones is evaluated using expected average overlap(EAO), accuracy and robustness.
2. Soccer player Tracking Experiment: the evaluation is based on the frame rate (frames per second or FPS) the tracker can correctly conduct at.

Note that the testing data are from VOT2021. Therefore, several per-frame visual attributes have been considered in the experiments, including target out-of-frame, partial occlusion, aspect change, size change, fast motion, similar objects, out-of-plane rotation, etc.

4.3. Backbone Selection Experiment

The selection of backbone is obtained by testing and evaluating ResNet-50 and modified AlexNet.

A modified AlexNet indicates the groups from conv2 and conv4 are removed [7]. It has been used in Siamese -RPN framework in [18]. We also use ResNet-50 as backbone here for an accuracy improvement. Both of them have been pretrained on ImageNet [24]. The fine-tuning process in each convolution layers can be performed by minimizing the loss function in the Eq. **8** with Stochastic gradient descent (or SGD).

As we discussed in section **3.4** and section **4.1**, the template patch and detection patch will be centering cropped to $127 \times 127 \times 3$ and $255 \times 255 \times 3$ respectively following the Eq. **11**. Note that we define the padding as follows in practice:

$$p = \frac{w + h}{2} \tag{13}$$

In the Table. **1**, we experiment with VOT2021 dataset. And the ResNet-50 outperforms the modified AlexNet in the tracking model.

	$\mathbf{ResNet-50}$	**AlexNet**
EAO	0.402	0.349
Accuracy	0.595	0.572
Robustness	0.243	0.219

Table 1: Testing results of backbones ResNet-50 and the modified AlexNet on VOT2021 dataset. ResNet-50 actually outperforms the modified AlexNet in our Siamese-RNP tracking model.

Specifically, ResNet-50 serving as a backbone in the Siamese-RPN gains about 15.2% relative increase in EAO, 4.0% relative increase in accuracy, and 11.0% relative increase in robustness.

Figure 5: Overview of soccer player tracking demo. The input soccer game video stream has a 60 FPS frame rates. In particular, we manually draw the target bounding box for the initial frame, as shown in the top left corner. When the soccer player is dribbling forward, the tracking challenges include similar objects, fast motion, shape change, partial occlusion, etc. Nevertheless, our soccer player tracking system still can obtain frame-accurate bounding boxes as we expected in the high frame rate test.

4.4. Soccer Player Tracking Experiment

In this section, we detail the main steps we installed dependencies for our soccer player tracking system as follows: 1) create environments (Conda with Python 3.7) and activate the tracker. 2) install the Pytorch 0.4.1, the Python libarary Numpy, and OpenCV. 3) install other requirements, including PyYAML, YACS, Matplotlib.

Now we investigate that if our soccer player tracking system can get compact bounding boxes in a real soccer game, we provide a tracking demo as follows. Firstly, run the tracking system and we can see the graphical user interface. Secondly, we need to select a valid video path on our computer as input video. Otherwise, an error window will pop up. Here we input a soccer game video stream with 60 FPS frame rate. Finally, we manually draw the target bounding box for the initial frame, as shown in the top left corner of Fig. 5. As we mentioned in previous sections, it is necessary because during the inference phase, template branch will receive the target patch from the initial frame.

In this soccer game video stream, the main challenges include, but not only include: 1) all the teammates will be regarded as similar objects for each other due to the same soccer jersey. 2) there always exists partial occlusion of target player, especially when the center back players try to defense. 3) the target's shape is severely changing. 4) the frame rate of the video stream is 60 FPS.

Nevertheless, we can see the frame-accurate tracking results as shown in Fig. 5. Although there exist several undesirable per-frame visual attributes, such as partial occlusion, size change, fast motion, similar objects, our tracker is still able to predict a promising scale, and provide an accurate ratio of proposal. On balance, we finally obtain many frame-accurate bounding boxes as we expected.

5. Conclusion

In this project, we do research on Siamese framework from regular Siamese-FC to Siamese-RPN. In order to solve the high-speed tracking problem, we learned the whole mathematic process of proposal selection strategies for the local one-shot detection task. In particular, we further explore a suitable backbone in experiment by evaluating ResNet-50 and modified AlexNet. Finally, we understand the implementation of Siamese-RPN more deeply by designing a comprehensive soccer player tracking system, which includes tracker implementation, and front-end GUI design.

From Siamese-FC to Siamese-RPN, we can see one of the improvements lies in the template branch in Siamese area becomes the training parameters, which embeds the category information into the kernel. And the detection branch in RPN performs detection using the embedded information. Hence, we believe future research may include extensions to deep embeddings for online trackers.

References

[1] M. Danelljan, A. Robinson, F. S. Khan, and M. Felsberg. Beyond correlation filters: Learning continuous convolution operators for visual tracking. In European Conference on Computer Vision, pages 472–488, 2016.

[2] D. Held, S. Thrun, and S. Savarese. Learning to track at 100 fps with deep regression networks. In European Conference on Computer Vision, pages 749–765, 2016.

[3] S. Tang, M. Andriluka, B. Andres, and B. Schiele. Multiple people tracking by lifted multicut and person reidentification. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 3539–3548, 2017.

[4] L. Bertinetto, J. F. Henriques, J. Valmadre, P. H. S. Torr, and A. Vedaldi. Learning feed-forward one-shot learners. In Advances in Neural Information Processing Systems, 2016.

[5] K. H. Lee and J. N. Hwang. On-road pedestrian tracking across multiple driving recorders. IEEE Transactions on Multimedia, 17(9):1429–1438, 2015.

[6] Benali Amjoud A., Amrouch M. Convolutional Neural Networks Backbones for Object Detection. In: El Moataz A., Mammass D., Mansouri A., Nouboud F. (eds) Image and Signal Processing. ICISP 2020. Lecture Notes in Computer Science, vol 12119. Springer, Cham, 2020.

[7] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. S. Torr. Fully-convolutional siamese networks for object tracking. In European Conference on Computer Vision, pages 850–865, 2016.

[8] Redmon, J., Farhadi, A.: YOLO9000: Better, Faster, Stronger. arXiv:1612.08242 [cs], 2016.

[9] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: towards real-time object detection with region proposal networks. In International Conference on Neural Information Processing Systems, pages 91–99, 2015.

[10] H. Nam and B. Han. Learning multi-domain convolutional neural networks for visual tracking. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2016.

[11] J. Valmadre, L. Bertinetto, J. F. Henriques, A. Vedaldi, and P. H. S. Torr. End-to-end representation learning for correlation filter based tracking. The IEEE Conference on Computer Vision and Pattern Recognition, 2017.

[12] M. Danelljan, G. Bhat, F. Shahbaz Khan, and M. Felsberg. Eco: Efficient convolution operators for tracking. In The IEEE Conference on Computer Vision and Pattern Recognition, 2017.

[13] M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, B. Shillingford, and N. De Freitas. Learning to learn by gradient descent by gradient descent. neural information processing systems, pages 3981–3989, 2016.

[14] F. F. Li, R. Fergus, and P. Perona. One-Shot Learning of Object Categories. IEEE Computer Society, 2006.

[15] Behler, J. (2011). Atom-centered symmetry functions for constructing high-dimensional neural network potentials. The Journal of chemical physics, 134(7), 074106.

[16] J. Valmadre, L. Bertinetto, J. F. Henriques, A. Vedaldi, and P. H. S. Torr. End-to-end representation learning for correlation filter based tracking. The IEEE Conference on Computer Vision and Pattern Recognition, 2017.

[17] F. Zhang, Q. Wang, B. Li, Z. Chen, J. Zhou. PysSOT, Github repository, https://github.com/STVIR/pysot, 2018.

[18] Li, Bo, et al. High performance visual tracking with siamese region proposal network. Proceedings of the IEEE conference on computer vision and pattern recognition, 2018.

[19] D. Gordon, A. Farhadi, and D. Fox. Re3 : Real-time recurrent regression networks for object tracking. arXiv preprint arXiv:1705.06368, 2017.

[20] Q. Wang, J. Gao, J. Xing, M. Zhang, and W. Hu. Dcfnet: Discriminant correlation filters network for visual tracking. arXiv preprint arXiv:1704.04057, 2017.

[21] Girshick, R.: Fast R-CNN. arXiv:1504.08083 [cs], 2015.

[22] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., Berg, A.C.: SSD: Single Shot MultiBox Detector. arXiv:1512.02325 [cs]. 9905, 21–37, 2016.

[23] Kong, T., Yao, A., Chen, Y., Sun, F.: HyperNet: towards accurate region proposal generation and joint object detection. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 845–853. IEEE, Las Vegas, 2016.

[24] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, and M. Bernstein. Imagenet large scale visual recognition challenge. International Journal of Computer Vision, 115(3):211–252, 2014.

[25] E. Real, J. Shlens, S. Mazzocchi, X. Pan, and V. Vanhoucke. Youtube-boundingboxes: A large high-precision humanannotated data set for object detection in video. arXiv preprint arXiv:1702.00824, 2017.

[26] M. Kristan, R. Pflugfelder, A. Leonardis, J. Matas, L. Cehovin, G. Nebehay, T. Vojir, F. G., and et al. The visual object tracking vot2014 challenge results. In ECCV2014 Workshops, Workshop on visual object tracking challenge, 2014.

[27] Nam H , Han B . Learning Multi-Domain Convolutional Neural Networks for Visual Tracking[J]. CVPR，2016.

[28] Kristan, Matej, et al. "The eighth visual object tracking VOT2020 challenge results." European Conference on Computer Vision. Springer, Cham, 2020.

[29] Yan, Bin, et al. "Alpha-refine: Boosting tracking performance by precise bounding box estimation." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2021.

[30] Kong, T., Yao, A., Chen, Y., Sun, F.: HyperNet: towards accurate region proposal generation and joint object detection. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 845 – 853. IEEE, Las Vegas (2016).

[31] Redmon, J., Farhadi, A.: YOLO9000: Better, Faster, Stronger. arXiv:1612.08242 [cs], 2016.