ROAM-based visual tracking method

Yue Zhang¹, Huibin Lu² and Xingwang Du³,*

¹School of Yanshan University, Hebei, China
²School of Yanshan University, Hebei, China
³School of Yanshan University, Hebei, China

*Correspondent e-mail: 623983800@qq.com

Abstract. In this paper, we improved the ROAM tracking model as well as replaced the first 12 layers of VGG-16 used in the experiment with the HETCONV convolution layer, where the first layer remains unchanged, as well as the other layers are divided into 4 parts. Each part has only one 3*3 convolution kernel, as well as the other convolution kernels are all 1*1, thus improving the speed of the tracking model while keeping the accuracy unchanged. We have extensively evaluated the improved ROAM as well as the unimproved ROAM on the TrackingNet benchmark, as well as the results prove that this method effectively improves the frame rate.

1. Introduction

Target tracking technology has brought a lot of convenience to our lives, so it has always been a hot topic for many scholars. With the rise of artificial intelligence in recent years, the combination of traditional target tracking methods as well as deep learning has become a new hot spot. However, despite the many novel methods that people have proposed, this technology still has many shortcomings. For example, when encountering complex background occlusion, high-speed target movement, obvious deformation of the target, as well as obvious changes in environmental lighting, it is easy to cause tracking drift as well as cause erroneous tracking results. Therefore, designing a stable as well as robust tracking model has become a top priority.

In the tracking process, it is most likely not precise that using the boundary box to gather the training samples which is utilized to renewal the mould. These minor mistakes is going to gradually flock in each update, eventually leading to tracking failure. Tianyu Yang et al. proposed the ROAM(Recurrently Optimizing Tracking Model) method to solve this problem[1]. Its tracking framework involves generating reply as well as boundary box regress two parts. The head part generates one response graph representing the possibility of overlaying the anchors on the preset gliding window, as well as in the next section The accurate rectangle is obtained by anchors to predict the bounding box. Each location only uses one size anchor, as well as adjusts the size of its corresponding convolution filter through bilinear interpolation to adapt to the shape change, thereby saving model parameters as well as calculation time. Nevertheless, because of the plentiful arguments as well as as well as the huge amount of calculation, the time required is still very long. In order to further improve the computational efficiency, this method has been improved in this paper, using efficient heterogeneous convolution in the convolution process of extracting features Filter. This filter is designed with zero delay, so the delay from input to output can be ignored. Using this improved method can greatly reduce the amount of calculation in the tracking process without sacrificing accuracy, reduce the calculation time, as well as improve the tracking efficiency.
2. Related work

Generally there are two ways to improve the tracking efficiency of the model. One is to design a more efficient filter, as well as the other is to compress the model. In recent years, many experts as well as scholars have designed efficient convolution filters, which are mainly divided into group-wise convolution (GWC) [6], deep-wise convolution (DWC) [4] as well as point-wise convolution (PWC) [5]. GoogleNet [7] uses inception modules as well as irregular stacking structure. Meanwhile, FLOPs can be reduced by the Inception module using GWC as well as PWC. In order to design a more efficient architecture, ResNet [10] uses a bottleneck structure together with remaining connections. Compared with VGG, they can design a deeper architecture. ResNetx [8] uses the ResNet architecture as well as divides each layer with GWC as well as PWC. The results proved that, however, a deeper or wider network is much less effective than increasing the base. The new connection designed by SENet [9], stress the importance on every output of the feature map. Although FLOPs are slightly increased, the performance is improved. MobileNet [11] is one popular of another architecture which is designed for IoT devices including DWC as well as PWC.

Another popular method to improve efficiency is compressional model which could be divided into three different types: 1-connection trimming [12], 2-filter trimming [13] as well as 3-quantization [12][14]. Compared with other methods, the filter trimming method is more effective as well as has a higher compression rate in FLOPs. In addition, the filter trimming method does not require any special support with regard to hardware/software (distributed library). In filter pruning, most part of work is to calculate the weight of the filter according to certain criteria, pruning it according to the weight, as well as then retraining to restore the loss accuracy. However, pruning is done on a trained in advance model, including repetitive training as well as trimming, requiring a lot of work. In addition, if the degree of trigger pruning increases, the accuracy of filter pruning will drop sharply, resulting in tracking failure.

3. The proposed method

3.1. HETCONV convolution

This novel filter/convolution (Heterogeneous Kernel-Based Convolutions) [2] uses heterogeneous kernels (for example, a few kernels are 3×3 in size, and other kernels can be 1×1) to decrease FLOPs of remaining patterns, and the precision is equivalent to the initial pattern.

We state the P part, which manages the quantity of different kinds of kernels in the convolution filter. The $\frac{1}{p}$ part of the total kernel is $K\times K$, and the existing part $(1-\frac{1}{p})$ is $1\times 1$. The filters of a specific sheet are placed in a removed means (that is, if the No1. filter begins the $k\times k$ core with the original place, the next filter begins the $k\times k$ kernel from the second position).

In the standard convolution layer, it is assumed that the size of the input (input feature map) is $Di\times Di\times M$. Among them, $Di$ is the width and height of the input square characteristic chart space, and $M$ is the input depth (the number of input channels). The size of the output (output characteristic chart) is $Do\times Do\times N$, where $Do$ is the 3-D width and height of the output square feature map, and $N$ is the output depth (the quantity of output canals). The standard convolution core is $k\times k\times m$, the calculation amount of the standard convolution is $Do\times Do\times M\times N\times K\times K$, The calculation amount of the convolution layer using Hetconv is $\frac{1}{p} + \frac{p}{k^2}$. When $p=1$, this type of heterogeneous convolution is the same as standard convolution. When $p$ is greater than 1, the comparison shows that the improved method reduces the amount of calculation.

3.2. Improved ROAM

The tracking model of ROAM includes two modules, the tracking model as well as the neural optimizer.
The tracking model $\theta$ consists of two embranchments, the response generation (R-generation) branch $\theta_{cf}$ as well as the bounding box regression branch $\theta_{reg}$. The R-generation branch determines the existence of the objective by predicting the confidence score graph, moreover, the bounding box regression branch assesses the accurate box of the objective by returning the coordinate shift to the box anchor set at the slipping shutter position. It can adjust the size adaptive to fit different form alters. The neural optimizer is responsible for updating the pattern to fit transforms in form. We use above-mentioned new convolution method on the basis of ROAM. With the exception of the first convolution layer, the whole excessing convolution layers are substituted with HetConv layers, holding the same number of file managers as before. The improved frame is shown in Figure 1.

$$L(F, \theta, M, B) = \| C(F; \theta_{cf}) - M \|^2 + \| R(F; \theta_{reg}) - B \|^2$$

Among them, is the R-generation network, $R(F; \theta_{reg})$ is the bounding box regression network, $F$ is the feature map input, $\theta$ is the network parameter, as well as $B$ is the ground truth box. The model is updated by formula (2).

$$\theta^{t+1} = \theta^{t-1} - \lambda^{t-1} \nabla_{\theta^{t-1}} l^{t-1}$$

Figure 1. ROAM tracking model framework

The offline training phase uses a recursive neural optimizer to perform a one-step gradient update on the tracking model to minimize losses in future frames. After the offline learning phase is complete, the trained neural optimizer is used to periodically update the tracking model to adapt to changes in appearance. The optimizer trained by the above method can be used for frame tracking, because it can quickly converge the model as well as generalize.
The $\lambda^{(t-l)}$ is the element basis about learning rate which owns the aforesaid dimensions with the tracking the parameters of the model $\theta^{(t-l)}$.

$$Z^{(t-l)} = [\lambda^{(t-l)}, \theta^{(t-l)}, l^{(t-l)}]$$

(3)

$$\lambda^{(t-l)} = \sigma(\theta(Z^{(t-l)}); \omega)$$

(4)

Among them, $\sigma(\cdot; \omega)$ is the coordinate-wise of LSTM parameterized by $\omega$ as well as $\omega(\cdot; \omega)$ partakes the parameters in whole dimensions of the input, moreover $\sigma$ is a sigmoid function that limits the expected learning rate. We test the lately updated model $\theta^{(t)}$ on the next three adventitious extracted frames as well as get the loss.

$$L^{(t)} = L(F^{(t+\delta)}, M^{(t+\delta)}, B^{(t+\delta)}, \theta^{(t)})$$

(5)

Among them, $\delta$ is randomly selected within $[0, \tau - l]$. $\tau$ is the frame interval between online tracking model updates. In the offline training phase, we will perform the above steps on a short video as well as get the average meta loss to optimize the neural optimizer. The average loss is expressed by formula (6).

$$\bar{L} = \frac{1}{NT} \sum_{i=1}^{N} \sum_{t=1}^{T} L^{(t)}$$

(6)

The $N$ in the above equation is the size of batch, $T$ is the number of model updates, as well as $V_r \sim p(V)$ is the video montage spot-checked from the training set. Additionally, the earliest tracking model parameters $\theta^{(0)}$ as well as at the same time the earliest learning rate $\lambda^{(0)}$ are trainable variables, that are learned together with the neural optimizer $O$. The flow of the Circulatory Neural Optimizer is shown in Figure 2.

![Figure 2. Cyclic Neural Optimizer flowchart](image)

4. Experiment

The experiment combines as well as uses Python as well as PyTorch machine learning library\[^{18}\], as well as The computer configuration is NVIDIA RTX2080 GPU as well as Intel® Core\textsuperscript{TM} i9 CPU @ 3.6 GHz. We use ADAM\[^{16}\] for optimization. Our framework is trained by 31 video clips. They are divided into 16 batches as well as distributed on 4 GPUs, each of which includes 4 videos.

Our training data set is TrackingNet\[^{3}\]. During the training process, the continuous sequence segments in the video data set are randomly extracted, as well as the same still image is repeated to form the video segment of the image data set. It’s important to note that we scale all frames used in
training. As in earliest regression parameter \( \theta^{(0)} \), earliest learning rate \( \lambda^{(0)} \), we adopt 1e-6 as the learning rate. We use 1e-3 as the earliest learning rate of the recursive neural optimizer. The learning rate of the optimizer as well as tracking framework must be multiplied by 0.5 every 5 cycles.

We replaced the first 12 convolution layers of the pre-trained VGG-16 with Hetconv with \( P=4 \) (in order to improve the spatial resolution of the feature map, we deleted the maximum pooling layer), as well as used it as a feature extractor. As in R-generation network, bounding box regression network are both composed of 2 convolution layers. The first layer is a dimensional reduction layer of 512*64*1*1 (inside the channel, outside the channel, height, as well as width), as well as the second layer They are the correlation layer of 64*1*21*21 or the regression layer of 64*4*21*21. Our neural optimizer uses two LSTM stacked layers, each containing 20 hidden units. The ROI scale factor is 5, the character span of the CNN character selection is 4, the span factor of filter size is 1.5 , 20 is used for R-generation, as well as 281 is used for search size. From \{0.8, 1, 1.2\}, select the scale as well as facet proportion coefficients \( s, r \) for scaling the earliest image block, as well as 9 sets of \( (s, r) \) can be obtained. The ambit factor adopted in RFS is \( K_d = 1.6 \) as well as \( K_{reg} = 1.35 \).

We test on the TrackingNet dataset. TrackingNet provides more than 30,000 videos with about 14 million dense border remarks by filtering short video montages from Youtube-BB.

![Success plots](image)

**Figure 3. Tracking Results.**

**Table 1.** Comparison of unimproved as well as improved ROAM results.

| Method        | AUC | Precision | Frame rate |
|---------------|-----|-----------|------------|
| ROAM          | 0.620 | 0.547     | 13         |
| Improved ROAM | 0.62  | 0.54      | 22         |

Evaluation indicators include the region below the curve (AUC) in Figure 3, accuracy, as well as frame rate. Table 1 lists the specific comparison under the TrackingNet test data set. The improved ROAM method is compared with the traditional ROAM method. The region below the curve as well as accuracy of Figure 3’s graph remain basically unchanged, while the frame rate has undergone a major change, from the original 13 frames per second to 22 frames per second. Experiments prove that our method improves the frame rate by 65.8%, as well as the accuracy remains unchanged.
5. Conclusion

This paper uses a novel structure of convolution that a heterogeneous kernel for tracking as well as compares the accuracy as well as frame rate of this convolution method with that of a stas well asard convolution. The experimental demonstrate that the HetConv convolution is more efficient to use than existing convolutions (less FLOP as well as higher accuracy). Since Hetconv convolution does not add layers (replace one layer with many layers, such as MobileNet) to reduce FLOP, the delay is zero. Using this new type of convolution method can effectively improve tracking efficiency as well as contribute to the future development of visual tracking.

References

[1] T. Yang, P. Xu, R. Hu, H. Chai and A. B. Chan, "ROAM: Recurrently Optimizing Tracking Model," 2020 IEEE/CVF Conference on Computer Vision as well as Pattern Recognition (CVPR), Seattle, WA, USA, 2020, pp. 6717-6726.

[2] P. Singh, V. K. Verma, P. Rai and V. P. Namboodiri, "HetConv: Heterogeneous Kernel-Based Convolutions for Deep CNNs," 2019 IEEE/CVF Conference on Computer Vision as well as Pattern Recognition (CVPR), Long Beach, CA, USA, 2019, pp. 4830-4839.

[3] Matthias Müller, Adel Bibi, Silvio Giancola, Salman Al-Subaihi, and Bernard Ghanem. TrackingNet: A Large-Scale Dataset as well as Benchmark for Object Tracking in the Wild. In ECCV, 2018.

[4] Vincent Vanhoucke. Learning visual representations at scale. ICLR invited talk, 2014.

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In CVPR, pages 770 – 778, 2016.

[6] Matej Kristan, Aleˇs Leonardis, Jiri Matas, as well as Michael Felsberg. The Visual Object Tracking VOT2016 Challenge Results. In ECCV Workshop on VOT, 2016.

[7] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural net-works. In NIPS, pages 1097 – 1105, 2012.

[8] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In Proceedings of the IEEE conference on computer vision as well as pattern recognition, pages 1 – 9, 2015.

[9] Jie Hu, Li Shen, and Gang Sun. Squeeze-as well as-excitation net-works. CVPR, 2018.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, as well as Jian Sun. Identity mappings in deep residual networks. In European conference on computer vision, pages 630 – 645. Springer, 2016.

[11] as well asrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Wey, Marco An-dretto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. ArXiv preprint arXiv:1704.04861, 2017.

[12] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning,trained quantization as well as huffman coding. ICLR, 2016.

[13] Jian-Hao Luo, Jianxin Wu, as well as Weiyao Lin. Thinet: A filter level pruning method for deep neural network compression.In CVPR, pages 5058 – 5066, 2017.

[14] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon,and Ali Farhadi. Xnornet: Imagenet classification using binary convolutional neural networks. In ECCV, pages 525 – 542. Springer, 2016.

[15] Karen Simonyan and rew Zisserman. Very Deep Convolutional Networks for Large-scale Image Recognition. In ICLR, 2015.

[16] Diederik P Kingma as well as Jimmy Ba. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980,2014.
[17] Marcin and asrychowicz, Misha Denil, Sergio Gomez, Matthew W. Hoffman, David Pfau, Tom Schaul, as well as Nas well aso de Freitas. Learning to Learn by Gradient Descent by Gradient Descent. In NeurIPS, 2016.

[18] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014.