Simplification and Optimization of Visual Tracking Convolutional Neural Networks Parameters

Zhiyong Qin[1,a], Lixin Yu[1,b]

[1]Beijing Microelectronic Technology Institute, Beijing 100076, China.

[a]qinzy2010@163.com, [b]yulx2013@126.com

Keywords: Visual tracking, convolutional neural network, dynamic compression.

Abstract. Convolutional Neural Network (CNN), which has achieved great success in traditional computer vision tasks such as image classification, object detection, scene segmentation, etc. has also gained great success in fields of visual tracking. However, the algorithm based on CNN is both computation and memory intensive, which makes it hard to deploy it on various systems, especially for embedded system. In this paper, we choose the champion of VOT 2015 challenge which is multi-domain convolutional neural network for visual tracking (MDNet) as our algorithm prototype. We eliminate the zero value and even more non-zero intermediate values in the process of algorithm to reduce both computation and memory resources. While the threshold of non-zero values is not easily determined, we experiment some thresholds and give the result of compression ratio and tracking accuracy to find the relationship between thresholds and tracking accuracy. Finally we achieve some relationship such as nearly 50% compression ratio with 2.3% area-under-the-curve (AUC) loss and analyze the reason of the case in which algorithm is failed.

1. Introduction

Visual object tracking refers to the problem that algorithms automatically estimate the location of target [1]. The visual tracking algorithms can be mainly divided into generative models and discriminative models. Discriminative models distinguish between the target and background by training classifier which is also known as tracking-by-detection. At present, most deep learning visual tracking methods are belong to discriminative framework. Since 2013, CNN has been applied to visual tracking and led to great advances in accuracy [2, 3, 4, 5].

While the methods based on CNN get state-of-the-art accuracy, it demands more computation and memory resources. Therefore, we usually have to run the algorithm on CPU, GPU and even server. But there are so much requirement of real-time and high-accuracy visual tracking task, such as unmanned aerial vehicles (UAVs) and robots [6]. So power will be a serious problem for embedded systems.

To solve this problem, many researchers have proposed various parameters compression technology. Han [7] presented a parameters compression technique which reduced massive parameters, but it only considered the static parameters. Cnvlutin (CNV) [8] observes that a large fraction of computation is multiplication with zero, so it eliminates the operation with no accuracy loss. Nevertheless the compression ratio is not high. We need to cut more CNN redundant parameters.

In this paper, we make a deep investigation on relationship of the tracking accuracy and the parameter compression ratio. Specifically, this paper makes the following contributions:

- We analyze the visual tracking CNN, and deal with the parameters after ReLU layer.
- We experiment on OTB100 with MDNet and get an optimized threshold which balance between the accuracy loss and compression ratio.
- We explore the different challenge which make tracking failed and the scene which is appropriate for using our algorithm.

2. Background

CNN has been successfully applied to traditional computer vision problems such as image classification, target detection, image segmentation, and so on. In recent years, CNN has been applied to visual tracking and has achieved outstanding results. Traditional methods require a lot of expert features which are designed artificially. Therefore it has a major drawback that these features are usually used for a certain object or a kind of object. Moreover, when we handle the new visual tracking problem we need to redesign. However, CNN has a strong representation power and provides high generalization ability.

2.1 Convolutional Neural Network Foundation

Convolutional Neural Networks have been tremendously successful in practical applications. CNN usually consists of a series of continuous layers, including the input layer, the output layer and a number of hidden layers. CNN with more than two hidden layers can be considered as deep neural network. The main type of hidden layers are convolution (including the subsequent activation functions), normalization, pooling and fully-connected (and subsequent activation functions). The convolutional layer mainly extracts characteristics of previous layers. Normalization can speed up the training process through accelerating convergence speed and improve accuracy simultaneously. Pooling can significantly reduce the complexity and decrease the number of weights. Fully-Connected layers gather all the features.

Convolutional layer gets input from previous layer and convolves with convolutional kernels to obtain the output feature maps. In this paper, 'in' denotes the input of convolutional layer, 'o' denotes the output feature maps, 'i', 'j', 'k' separately denotes the index of three dimensions, 'w' denotes the convolutional kernels, 'x', 'y', 'z' separately denotes the three dimensions of kernels, 'k' denotes the number of feature maps, 's' denotes the stride of kernel and 'b' denotes the offset of feature map, so the calculation of the convolution layer is as follows:

$$o(i,j,k) = \sum_{x=0}^{X-1}\sum_{y=0}^{Y-1}\sum_{z=0}^{Z-1} w^k(x,y,z)\cdot in(x+i\times s, y+j\times s, z) + b^k \quad (1)$$

where X, Y, and Z denote the size of the corresponding dimension of the convolution kernel. Each output point is calculated by the convolution operation of the convolution kernel and the input part. A convolutional kernel moves from left to right, from top to bottom and point by point in a fixed step. All the convolutional kernels take the same operation to get a series of output feature maps. The relationship of output dimension and input dimension are as follows:

$$(O_x, O_y, O_z) = \left(\frac{I_x - X}{s} + 1, \frac{I_y - Y}{s} + 1, k\right) \quad (2)$$

where I_x and I_y respectively denotes the dimension of input feature map, O_x,O_y and O_z denotes the three dimension of output feature map apart. The output feature map needs to be processed by the activation functions. Most people use ReLU as activation functions which can be expressed with:

$$ReLU(x) = max(0, x) \quad (3)$$

From the equation we can know that it is very computationally efficient and converges much faster.

Pooling layer is usually used immediately after convolutional layers. A pooling layer takes each feature map and select the maximum or average value as the output of the subarea. In detail, max-pooling operations are as follows:

$$o(i,j,k) = max\begin{pmatrix} in(x,y,z) & \cdots & in(x,y+n-1,z) \\ \vdots & & \vdots \\ in(x+n-1,y,z) & \ldots & in(x+n-1,y+n-1,z) \end{pmatrix} \quad (4)$$

where x, y and z denote the dimension of input, i, j and k denote the dimension of output, n denotes the region size of subarea.

Fully Connected Layer is a linear mapping from the input to the output. The expression is as follows:

$$O = W \times IN + B \quad (5)$$

Where 'O' denotes the output feature vector, 'IN' denotes the input feature vector, 'W' denotes the weights matrix and 'B' denotes the bias vector.

2.2 OTB 100 dataset

Object Tracking Benchmark [9] has been one of the most important dataset for evaluating tracking algorithms. OTB 100 constructs 100 fully annotated sequences to facilitate the performance evaluation. In these sequences, there are almost all the main tracking challenges, including illumination variation, scale variation, occlusion, deformation, motion blur, fast motion, in-plane and out-of-plane rotation, out-of-view, background clutters and low resolution. In this paper, CNN model are all evaluated by OTB 100 dataset.

2.3 State-of-the-art visual tracking CNN model

In VOT 2015, Hyeonseob Nam etc. won the first place in visual tracking challenge using MDNet [10]. MDNet was a novel visual tracking algorithm based on CNN. The specific network layer of CNN generally includes the convolutional and fully-connected layers. In MDNet, for convolution operation parts, there are convolutional layer, the ReLU activation function layer, the LRU normalization layer and the max-pooling layer; for the fully-connected parts, there are the fully-connected layer, the ReLU activation function layer, Dropout and softmax normalization layer.

3. Motivation

Deep Neural Networks usually have more parameters than the real need for describing a problem properly. These parameters will need a mass of extra footprint and computation resources. As the depth of the convolutional neural network increasing, there are more and more parameters. The computing resource which is required for computing will also increase substantially. It makes that the training and predicting of CNN at different platform become more difficult.

There are a lot of research on the theme. The Deep Compression of Han Song [6] mainly deal with too many static parameters in the network. Firstly, Han prunes redundant branches in the network; Secondly, the parameters are stored in groups and encoded so the parameter of same group only need to store a fine-tune value as the center and the corresponding location only need to store the encoded value. Therefore the amount of storage reduces further. Finally, the usage of Huffman encoding makes a lossless compression. But this method has a main disadvantage that only the static model parameters are processed and it requires a lot of retraining work. Cnvlutin finds that the neural network will produce a large number of zero in the dynamic process so it skip the processing of zero by hardware. It reduces the consumption of storage and computation. We are inspired by the idea, if the target application allow a certain degree of precision loss we can reduce the parameters further, thus saving more storage and computation overhead.

4. Experiment

In order to research the relationship between the compression ratio and the precision of CNN in visual tracking, we choose MDNet to make experiments. The network of MDNet has been introduced in the section of ‘state-of-the-art visual tracking CNN model’. The concrete intermediate parameters are as table 1 showed:

Table 1 the scale of every layer intermediate values

Layers	Conv	ReLU	Normalize	Pooling
parameters	107×107×3×256	51×51×96×256		
	25×25×96×256	11×11×256×256		
Layers	Conv	ReLU	Output	
parameters	5×5×256×256	3×3×512×256	3×3×512×256	

Convolutional neural network will produce many zeros in the course of running. Where are the zeros from? A large number of negative values are generated after the Conv layer, a large number of zeros are generated by the activation function ReLU. So it is effectual to dynamically process the parameters after the ReLU layer. It reduces the computational and storage resources for the normalization, pooling and the next convolution layers. And it can achieve the highest zero detection efficiency compared to process after Conv layer. We use the algorithm 1 to determine

thresholds:

Algorithm1 search threshold

Input: a series of ReLU outputs X
Output: thresholds t
1: sort all the ReLU outputs X from minimum to maximum
2: delete all the zero in X, get X1
3: calculate the total number of X1 is n, then get the index(i) = round(0.1*n*(2*i-1))
4: so t(i)=X1(index(i))

So we get the five thresholds and the five algorithms which are denoted as 10%, 30%, 50%, 70% and 90%. The meaning are not literally the ratio cutting down all the parameters, while they denote the ratio compressing normalization, pooling and the next conv layer parameters except for zero. We give the parameter scale and compression ratio of these algorithms in table 2:

Table 2 parameters and compression ratio of MDNet and five algorithms

	MDNet	10%	30%	50%	70%	90%
Parameters	243.7M	155.3M	139.3M	123.2M	107.1M	91.0M
Compression ratio	1	0.6373	0.5714	0.5051	0.4395	0.3736

We experiment the five algorithms on OTB100. Figure 1 shows some representative sequences:

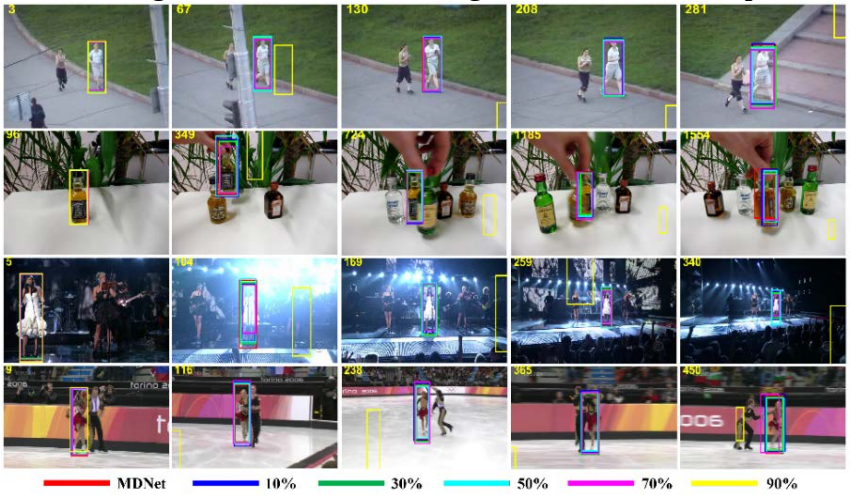

Fig. 1 Qualitative results on some challenging sequences(Jogging-2, liquor, singer1, skating2-1)

Figure 1 mainly demonstrates that decrease in number of intermediate results makes no big difference on tracking results. The results of these sequences give us some intuition of the impact on tracking result of different extent parameter reduction. For completeness, we show the success plots of OPE (one-pass evaluation), and all the challenge factors.

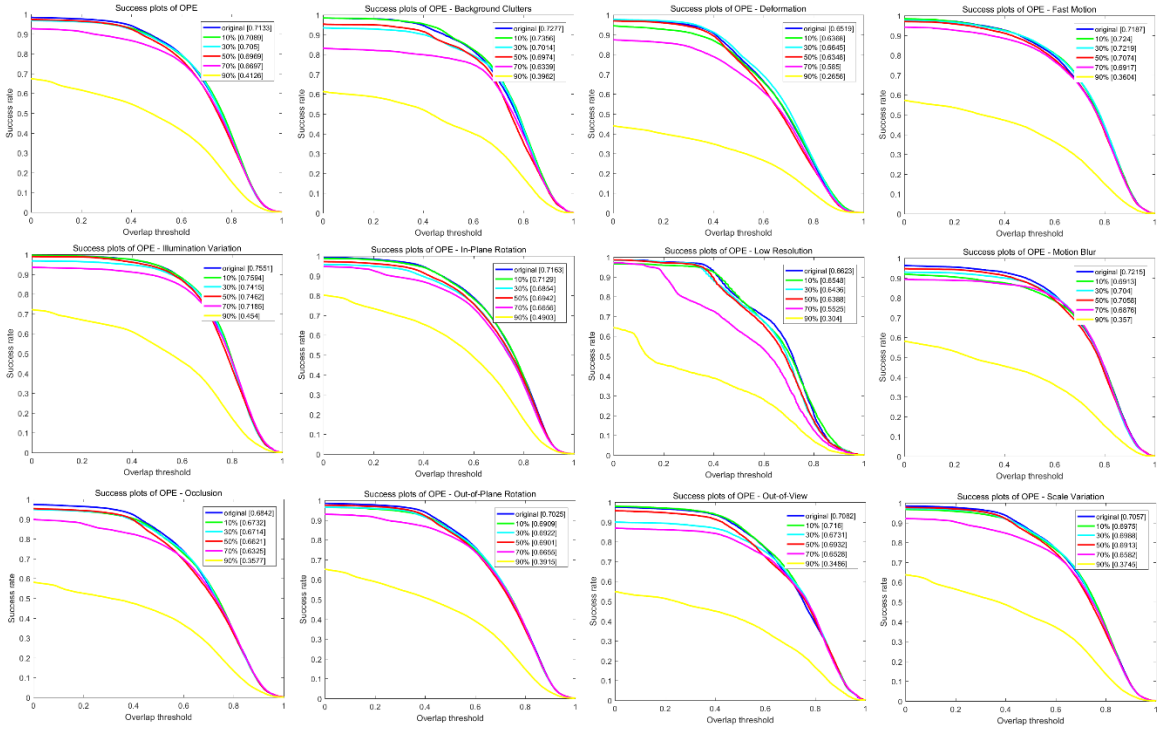

Fig. 2 The success plots for eleven challenge attributes: background clutters, deformation, fast motion, illumination variation, in-plane rotation, low resolution, motion blur, occlusion, out-of-plane rotation, out-of-view, scale variation and all

Figure 2 illustrates success plots based on bounding box overlap ratio. The definition of bounding box overlap ratio is expressed as equation 6:

$$r = \frac{S_p \cap S_g}{S_p \cup S_g} \tag{6}$$

where r denotes the overlap ratio, S_p denotes the extent of predicted bounding box, S_g denotes the area of ground-truth, $\cap$ denotes the intersection operation and $\cup$ denotes the union operation. From figure 3, we know that the 90% algorithm has a large precision loss. And the 70% algorithm will have a certain degree loss compared to MDNet in precision when challenge are BC, LR, OV and etc. For accurate comparison, we judge AUC in table 3:

Table 3 AUC loss compared to MDNet (%)

	IV	SV	OCC	DEF	MB	FM	IPR	OPR	OV	BC	LR	overall
10%	-0.57	1.16	1.61	2.32	4.19	-0.74	0.47	1.65	-1.1	-1.09	1.13	0.62
30%	1.8	0.98	1.87	-1.93	2.43	-0.45	4.31	1.47	4.96	3.61	2.82	1.16
50%	1.18	2.04	3.23	2.62	2.18	1.57	3.09	1.77	2.12	4.16	3.55	2.30
70%	4.85	6.73	7.56	10.26	4.7	3.76	7.08	5.27	7.82	12.89	16.58	6.11
90%	39.88	46.93	47.72	59.26	50.52	49.85	31.55	44.27	50.78	45.55	54.1	42.16

Table 3 illustrates that the 90% algorithm has an accuracy loss exceeding 40%, while the 70% algorithm have about 6% accuracy loss. But the 70% algorithm perform really not well in low resolution, background clutters and deformation. Because the information of target for low resolution is original little, the parameters decrease will make feature information become less and increase the difficulty of tracking. When the feature information reduce to a certain extent, the precision will have an obvious loss. Background clutters make the noise more like signal, so if the storage of signal is not accurate, the tracking result will be bad. Deformation make some useful target information varied, so the details of feature is more important in the scene. 50% algorithm has an accuracy loss of 2.3% which looks great. But we need to notice some scene which is showed in figure 3:

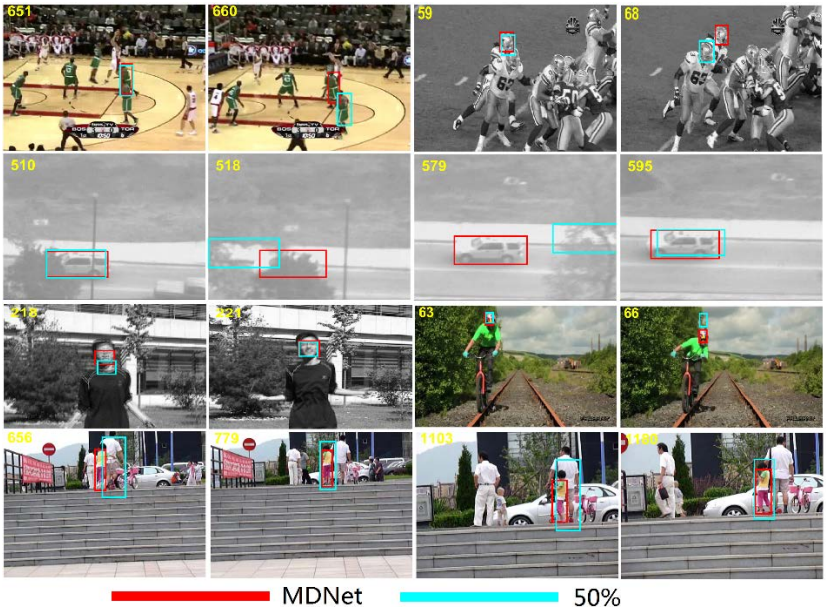

Fig. 3 the tracking results comparison of MDNet and the 50% algorithm to four main challenge (background clutters, occlusion, fast motion, scale variation)

Figure 3 illustrates that the 50% algorithm is not so robust. When the target has a similar object in background or is occluded, the algorithm is easy to track the background object similar to target or the occlusion. When the target move fast which exceeds 20 pixels a frame or varies frequently in scale, the algorithm regulate slower than original algorithm. 30% algorithm performs similar to 50% algorithm. 10% algorithm only has an accuracy loss of 0.62% and performs most like original algorithm.

5. Conclusion

We analyze the representative network of visual tracking algorithm which uses deep convolutional neural network, a multi-domain learning framework referred to as MDNet. In order to decrease the memory footprint and computational resource when CNN runs, we cut all the zeros and even some small values in intermediate results. When we cut the small intermediate results, we design an algorithm to determine the threshold and experiment on OTB100 datasets. We get the area under curve (AUC) of different algorithm, e.g., 10% algorithm has an AUC loss of only 0.62%, while 90% algorithm has an AUC loss of exceeding 40%. Finally, we analyze the reason of AUC loss for most algorithms.

Reference

[1] Ross, David A., et al. "Incremental learning for robust visual tracking." International Journal of Computer Vision 77.1-3 (2008): 125-141.

[2] Wang, Lijun, et al. "Visual tracking with fully convolutional networks." Proceedings of the IEEE International Conference on Computer Vision. (2015).

[3] Hong, Seunghoon, et al. "Online tracking by learning discriminative saliency map with convolutional neural network." arXiv preprint arXiv:1502.06796 (2015).

[4] Li, Hanxi, Yi Li, and Fatih Porikli. "Deeptrack: Learning discriminative feature representations online for robust visual tracking." IEEE Transactions on Image Processing 25.4 (2016): 1834-1848.

[5] Wang, Naiyan, et al. "Transferring rich feature hierarchies for robust visual tracking." arXiv preprint arXiv:1501.04587 (2015).

[6] Han, Song, Huizi Mao, and William J. Dally. "Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding." CoRR, abs/1510.00149 2 (2015).

[7] Han, Song, et al. "EIE: efficient inference engine on compressed deep neural network." arXiv preprint arXiv:1602.01528 (2016).

[8] Albericio, Jorge, et al. "Cnvlutin: ineffectual-neuron-free deep neural network computing." Computer Architecture (ISCA), 2016 ACM/IEEE 43rd Annual International Symposium on. IEEE, (2016).

[9] Wu, Yi, Jongwoo Lim, and Ming-Hsuan Yang. "Object tracking benchmark." IEEE Transactions on Pattern Analysis and Machine Intelligence 37.9 (2015): 1834-1848.

[10] Nam, Hyeonseob, and Bohyung Han. "Learning multi-domain convolutional neural networks for visual tracking." arXiv preprint arXiv:1510.07945 (2015).