Rep-Tracker: a lightweight tiny ball tracking method using structure re-parameterization

Wenjia Wu (wuwenjiamnsd@sina.com)
Minnan Normal University
Lingfei Ren
Minnan Normal University
Wenyuan Yang
Minnan Normal University

Research Article

Keywords: Computer vision, deep learning network, tiny object tracking, prune, re-parameterization

Posted Date: May 15th, 2023

DOI: https://doi.org/10.21203/rs.3.rs-2903142/v1

Additional Declarations: No competing interests reported.

Rep-Tracker: a lightweight tiny ball tracking method using structure re-parameterization

WenjiaWu[1], Lingfei Ren[2] and Wenyuan Yang[2*]

[1]School of Physical Education, Minnan Normal University, Add. 36 Xianqianzhi St., ZhangZhou, 363000, Fujian, China.
[2]Fujian Key Laboratory of Granular Computing and Application,Minnan Normal University, Add. 36 Xianqianzhi St., ZhangZhou, 363000, Fujian, China.

*Corresponding author(s). E-mail(s): yangwycn@gmail.com; wwj0603@mnnu.edu.cn;
Contributing authors: renlf@163.com;

Abstract

Vision-based object tracking techniques are used to analyze sport competition videos. The latest tennis tracking models achieve very high accuracy. However, high precision networks often imply higher computational complexity, which in turn brings higher hardware requirements. Also, part of the networks suffer from the problem of different degrees of adaptation to different courses. In this paper, a lightweight tiny ball tracking method is developed based on deep learning call Rep-Tracker, which incorporates the structure re-parameterization strategy on the basis of the pruned VGG-style model to construct a tennis tracking network. Firstly, the video frames will be sent into a pruned VGG-16 encoder follow by a decoder composed of deconvolution, and heatmap will be output after a softmax layer. Based on the encoder-decoder structure, pruning methods are used to reduce the number of non-essential convolutional layers and channels in the network. Secondly, residual connections are added at each layers to form a multi-branch block at training time. By using a structural re-parameterization strategy, each branch in a multi-branch block will be equivalently transformed into a shape consistent convolution. The trained blocks are convert into a single convolution layer for inference. Finally, based on the heatmap, the position of the target object is obtained by Gaussian distribution function

and Hough gradient function. In the tennis tracking task our method achieves a high accuracy with GFLOPs only 1.29 on RTX 3080 GPU.

Keywords: Computer vision, deep learning network, tiny object tracking, prune, re-parameterization

1 Introduction

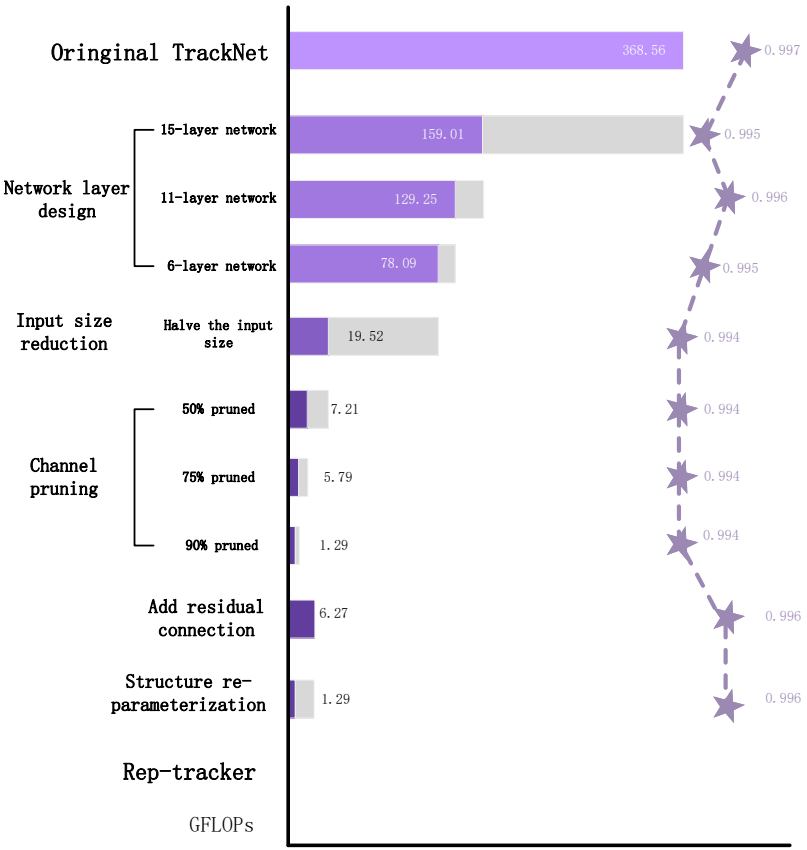

Fig. 1 The original TrackNet is modernized toward the lightweight design. The network layer design, input size reduction and channel pruning method are incorporated in TrackNet and lead to increasingly decline of GFLOPs.

Computer vision and deep learning perform important roles in object detection and tracking over the past decades[1]. Due to the rising interest in deep learning based computer vision task, this field gained great attention and have

wide range of applications such as object tracking[2], instance segmentation[3], image captioning[4], action recognition[5] or scene comprehension[6]. Among them, tiny object detection and tracking are the key and complex problem. Trending research in tiny object tracking field attract many researchers for the applications of driver-less car, medical tumor detection, security-based detection systems etc[1]. Also in the field of sports, tracking fast and tiny objects such as tennis and badminton is one of the important means of sport match analysis.

The application of tiny object tracking technology in sport match analysis has attracted great attention in recent years and has been applied to tennis and other sports. Large amount of data will be extracted from the broadcast videos for analysis using deep learning based method. Different classes of people have different needs for information, which expands the space and scale of information needed[7]. Taking tennis as an example, deep learning object tracking method is capable of getting the position and trajectory of the tennis ball from broadcast video. For athletes and coaches, tennis position and trajectory data is a powerful enabling foundation for analysing the athletes, significant information can be inferred knowing the positions of the balls or players during a match[8, 9]. These data will be helpful for professional tennis players and coaches to optimize training methods or analyse opponents. Consequently, information acquisition from tennis broadcast video using Convolutional Neural Networks(CNNs) become one of the research focuses in tiny object tracking and achieve satisfactory performance[10, 11].

VGG[12] as a classic CNNs achieves huge success in computer vision. In pursuit of better performance, there have been many innovations in constructing complex ConvNets base on VGG-style structure. GoogLeNet[13] and Inception models[14–16] adopt a designed multi branch architecture. He *et al.*[17] propose ResNet which designed the residual module to solve the gradient explosion and gradient disappearance problems of deep neural network. Huang *et al.*[18] determine that training should be deeper, more accurate, and more efficient if there is a short connection between the layer near to the input and the layer close to the output.

In recent years, researchers have achieve remarkable progress to solve localization and classification for insufficient information in individual feature layers in tiny object tracking field[19]. Tang *et al.*[20] and Li *et al.*[21] introduce Pyramid-Box and Pyramid-Box++ based on multi-scale representation learning strategy and integrate sufficient low-level features with high-level context semantic features. Hong *et al.*[22] propose SSPNet to generates hierarchical attention heat map by considering the context information, and strengthening the specific scale features of different layers, thus the corresponding features of different layers are shared. Leng *et al.*[23] introduce IENet, a powerful detection system that uses object appearance and context information. Similarly, there are many achievements in ball tracking field. Conventional ball tracking networks are based on VGG-style model mostly. Huang *et al.*[10] porpose TrackNet which use a single branch VGG network as a feature extractor and

get the position information of the ball from the heat map. Voeikov *et al.*[24] propose TTnet and show that VGG-style encoder and have a better performance in almost all metrics while having lower inference time. Both temporal and spatial data is provided and this data association strategy improve tracking performance significantly.

In practical application, network deployment needs to consider not only performance, but also efficiency, which including inference efficiency, training efficiency and data efficiency. For these purpose, there are two main aspects need to be concerned, network structure and data scale. Pruning is an effective way to improve network inference and training efficiency, which saves memory and computation without impairing performance[25, 26]. It is roughly divided into structural pruning and non structural pruning according to the granularity of the removed components[27]. Another way to improve network efficiency is network morphism[28, 29]. Network morphism is able to discard multiple layers into one layer while retaining the original network functions such as model re-parameterization[30–32].

However, the deploy of tennis tracking network still remains many problems. Current CNNs based network usually use neural architecture search. ConvNets with higher performance are produced manually, but it requires a lot of manpower or processing power. Notably, the growth of performance at the expense of speed make it difficult for the deploy of tracking network to mobile and edge devices. Moreover, there are many factors that affect the video like tennis court, camera position, weather etc. These bring difficulties to the generalization of the model.

To address these issues, a lightweight pruned-VGG based network is proposed for tennis ball tracking from broadcast videos. Our method draws on the structure re-parameterization strategy and improves organically on the basis of the encoder-decoder architecture. Network pruning method is introduced to reduce non-essential convolutional layers and channels. On the premise of not increasing the network size, structure re-parameterization is able to give the network the characteristics of a residual connection. Specifically, our Rep-Tracker includes three key steps, and the overall process is shown in Fig. 1. Firstly, tennis match video frames are sent into a pruned VGG-style encoder-decoder based model. Features will be extracted from the frames and then converted to heatmap through a series of up-sampling and deconvolution operations. On the basis of VGG-16 and deconvnet, pruning method is used to reduce redundant layers and channels. Secondly, a multi-branch network and a plain network are designed for training and testing time. At training time, residual connections consisting of 1×1 convolutional layer and identity layer are added at each layers to form a multi-branch block. After training, re-parameterization is introduced to convert the multi-branch block into a convolutional layer. The 1×1 convolutional layer and the identity layer will be equivalent to a convolution layer with the kernel 3×3. And the transformation is completed by convolution addition. Finally, after obtaining the heatmap, Gaussian distribution is used to express the diameter of tennis image. The

position of the tennis will be found on the black and white binary heat map using the Hough gradient method.

To sum up, this paper makes the following contributions:

- A pruned VGG-style based method is used to track the tennis ball from broadcast video. By using the method of structured pruning and unstructured pruning we achieve a lightweight network for tiny ball tracking.
- Based on structure re-parameterization the residual connection is added at each layers at training time for data efficiency. At inference the multi-branch block is converted into a plain convolutional layer equivalently for inference efficiency.
- Our network get a high degree of precision in a series of experiments for tennis ball tracking with GFLOPs only 1.29.

The following is the breakdown of this essay. Section 2 presents some of the work that is connected to this essay. The full implementation of our Rep-tracker is detailed in Section 3. The experimental findings of this tracker are displayed in Section 4. The entire paper's conclusion and future study initiatives are outlined in Section 5.

2 Related works

This section describes evolution of VGG-style model and the structure re-parameterization technology. These are two basic components of our proposed reptracker.

2.1 VGG-style model

As a pioneering work, Alex Krizhevsky *et al.* design AlexNet and open the door for significant use of deep learning in the computer vision sector. Karen *et al.* propose VGG family depending on the numbers of sequential layers to investigate the connection between the convolutional neural network's performance and depth[12, 33]. Compare with previous work, VGG use the 3×3 convolution filters instead of big convolution filters to increase detail feature capture capability and network capacity. In practical application, multiple 3×3 convolutions in series can equivalent to one bigger convolution, which obtains greater learning ability while reducing the number of parameters and maintaining the same receptive field. Subsequently, a residual learning framework is presented by Kaiming He *et al.* to enhance network expression and learning ability while preventing over-fitting and model degradation[17]. ResNet design the Residual block as $H(x) = F(x) + x$, Identity mapping is considered to be a direct component of the network. When x inputs the residual module, the output y is expressed as

$$\mathbf{y} = \mathcal{F}(x, \{W_i\}) + x, \tag{1}$$

where $\mathcal{F}(x, \{W_i\})$ is the target value for learning. Additionally, the residual block has a two-layer weight that activated by Relu.

$$\mathcal{F} = W_2\sigma(W_1x), \tag{2}$$

where σ refers to Relu, W_1 and W_2 refer to two-layer weight. This enables the network to train more efficiently and prevent gradient dispersion.

2.2 Network pruning

Pruning the network entails deleting variables that have no bearing on network accuracy[34–36]. In order to deploy neural networks in restricted environments, such as embedded devices, it seeks to remove unnecessary parameters or neurons that do not significantly affect the accuracy of the output. Depending on how finely the components are removed, modern static pruning technologies are separated into organized and unstructured pruning.[27, 37]. A convolution operation, as depicted below, is the fundamental function of a network.

$$\mathbf{F}_n^{l+1} = \mathbf{A}_n^l = \text{active}\left\{\sum_{m=1}^{M}(\mathbf{W}_{mn}^l * \mathbf{F}_n^l) + \mathbf{b}_n^l)\right\}, \tag{3}$$

where $\mathbf{W}$ represents the weight value of tensor with m input channel and n output channel, $\mathbf{b}$ represents offset vector, $\mathbf{F}^l$ represents input characteristic tensor. One way to prune a network is called brute force pruning, which traverses the whole network in element mode, and removes the weights that do not affect the accuracy. A common metric used to choose which values to trim is provided by the l_p-norm, s.t. $p \in \{N, \infty\}$, where N is natural number.

$$\|x\|_p = \left(\sum_{i=1}^{n}|x_i|^p\right)^{\frac{1}{p}}. \tag{4}$$

These method remove filters that do not affect the accuracy through a more direct approach[38].

2.3 Model re-parameterization

Network transformations are introduced by chen *et al.* in the context of transfer learning[39]. DiracNet tries to remove the fixed layer hopping connection and replace the layer hopping connection with a parameterized method[40]. On these basis, ding *et al.* design RepVGG using the structural re-parameterization technique to achieve the decoupling of the training-time and inference-time architecture[41, 42]. The equivalent transformation of operations is the core ingredients of re-parameterization technique[28]. Concretely for instance, Conv functions by transforming an input feature $I \in \mathbb{R}^{C\times H\times W}$ to an output O *i.e.*,

$$\boldsymbol{O} := o(\boldsymbol{I}) = \boldsymbol{I} \circledast \boldsymbol{F} + \boldsymbol{b} \in \mathbb{R}^{D\times H'\times W'}, \tag{5}$$

where C, H and W refer to channels, height, and width of the input. $F \in \mathbb{R}^{F\times C\times K\times K}$ and $b \in \mathbb{R}^{D}$ are the parameters of the convolution operator $\circledast$. The additivity is guaranteed by the convolution operator's linearity $\circledast$. So if there is two convolutions follow the same configurations $o^{(1)}$ and $o^{(2)}$ with weights $F^{(1)}$ and $F^{(2)}$, they are expressed as

$$\boldsymbol{I} \circledast \boldsymbol{F}^{(1)} + \boldsymbol{I} \circledast \boldsymbol{F}^{(2)} = \boldsymbol{I} \circledast \left(\boldsymbol{F}^{(1)} + \boldsymbol{F}^{(2)}\right). \tag{6}$$

A direct observation is that two compatible Conv operations can therefore be combined into a new Conv operation

$$\boldsymbol{F}^{(3)} = (\boldsymbol{F}^{(1)} + \boldsymbol{F}^{(2)}). \tag{7}$$

This shows that it is possible to combine multi-branch operations, or a set of equivalence operations into a single convolution and have additive properties. By include alternative branches in the training process, these transformations not only improve the neural network's capacity for representation, but also be equivalently turned into simpler operations, lowering the neural network's capacity for inference.

3 Proposed Method

This section introduces the processes and the details of our rep-tracker. rep-tracker is based on a encoder-decoder detector. The related algorithms and expressions are exhibited. Rep-tracker is an efficient method to track tennis ball from broadcast videos.

3.1 Overview of the proposed network

In this paper, structure re-parameterization technology is used to lightweight object tracking network. The main architecture of our network is showed in Fig.2. Different from previous work utilization rate of hardware resources and practicality of application is mainly focused. A pruned VGG-style network is designed as our backbone and use re-parameterizable module as basic module. VGG is an excellent backbone in the field of computer vision and has been widely used until now. In order to minimize the amount of calculation the number of the use of convolutional layers is controlled. In VGG-style network the reduction of convolution layer usually means the decline of model performance. Re-parameterization allow us to construct some extra structures to provide the properties that the model lacks. And at inference time the model is equivalently transformed back to the original one. Branch structures are added in the blocks during to improve the effect of model training. After getting feature map, it will be converted to a heat map. Through Gaussian function and Hough gradient function the position and shape of the ball can be obtained.

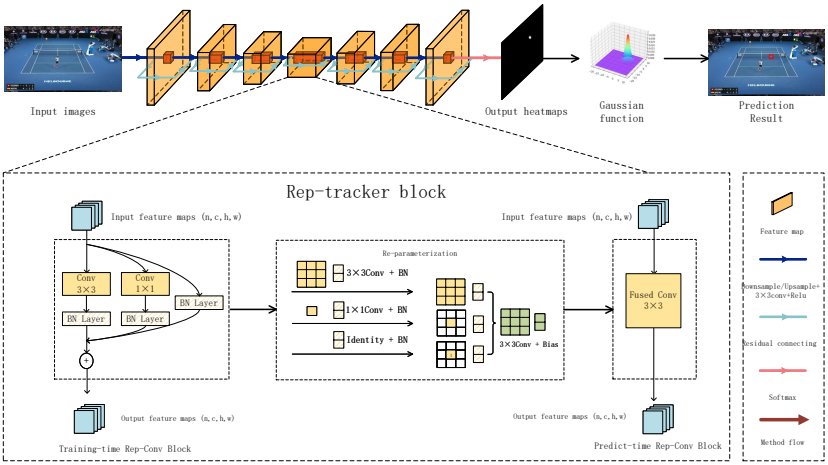

Fig. 2 The figure shows the main network architecture of this paper. Every frames of video are sent into an network using encoder-decoder framework. The encoder of the network is composed of several VGG like convolution layers. The encoder part aims to get the feature map. The decoder part is made up of deconvolution layer. Deconvolution network is regarded as the inverse process of the corresponding layer in the convolution network. Deconvolution inversely maps the feature map back to the pixel space of the input picture, so as to explain which pixels in the picture participate in activating the feature map. After this part, the feature map will be convert into heatmap, and get the position information of the ball through the algorithm. The whole encoder-decoder network has two different forms at training time and inference time. Residual connections are added at each layers at training. At inference time, the complex multi branch block will be converted into a convolution layer by structure re-parameterization strategy.

3.2 Rep-tracker network

Rep-tracker is composed of a pruned VGG-style encoder followed by a deconvolutional neural network decoder. Consecutive frames may be required to produce a heatmap and then use an algorithm to determine the object's position. The deconvolution network creates object segmentation from the features extracted in the convolution network, whereas the convolution network refers to a feature extractor that transforms the input image into a multidimensional feature representation. The network's final output, a probability map the same size as the input image, can show the likelihood that each pixel belongs to the desired feature.

Pruned VGG 16 network is used as encoder part and remove the last classification layer where only one convolution layer is kept at each scale, so our convolution network has only six convolution layers in total. Through the processing of convolution layer, the input image will output the feature layer after passing through the encoder. The operation is expressed as

$$X_{output} = \frac{(X_{input} - X_{input} + 2P)}{S} + 1, \tag{8}$$

where X is images' width or height, S represents convolution kernel size.

To ensure the performance of the model after pruning, residual connections are added in each layers during training. ResNet explicitly constructs a fast branch, modeling the information flow as $y = x + f(x)$. In order to avoid the loss of coupling in channel pruning, an additional 1×1 convolution is added in each layers. So the training-time convolution blocks is $y = x + g(x) + f(x)$. Several such blocks are stacked to build the training-time model.

With several series of disconnected layers, deconvolution layers, and correction layers, the deconvolution network component is a mirror image of the convolution network. The deconvolution network expands the activation using a mix of unpooling and deconvolution operations, in contrast to the convolution network, which reduces the activation size through feedforward.

3.3 Model Compression and Optimization

Our tracking model is based on VGG architecture which is stacked by 3×3 convolution layers. Original VGG style feature extractor achieves good results while consuming a lot of computing resources. When the size of input flame comes to 640×360, the model has the FLOPs of 368.56G and 2226.27MB memory cost. Considering that the more convolutional layers, the more computing resources are consumed, optimal pruned structure for the number of layers is searched.

Table 1 Network structure after pruning

Input size	Operator	In-channels/Out-channels	Activation
640×360	Conv 3×3	9/64	ReLU+BN
640×360		64/64	
320×180		64/128	
160×90		128/256	
320×180		256/128	
640×360		128/64	

As depicted in Tab.1, only one convolution layer is left in each dimension. After a series of experiments, only 7 convolutional layers remain in the final network. This process reduces the GFLOPs from 368.56G to 19.52G with minimal impact on accuracy.

In TrackNet, each layers consists of a simple plain convolution layer. Plain ConvNet has a fatal weakness that it often suffer from poor performance. So shortcut connection is added in each layers and resulting in good performance across different application scenarios.

3.4 Structure re-parameterization

As mentioned before, the structure re-parameterization method is used to covert trained block into inference block. In training time, the model use a three branches block. The main branch is a convolution layer with a kernel size of 3×3 follow by a batch-normalization(BN) layer. Another two shortcuts is a 1×1

convolution layer follow by a BN layer and a branch with only BN connected. This subsection describe the specific algorithm of re-parameterization.

3.4.1 Fusion of convolution layer and BN layer

In Rep-Tracker, a large number of blocks with convolution layer follow by BN layer are used. When the layers are merged, the network performance can be improved. The convolution layer formula with bias is expressed as

$$CONV(x) = W(x) + b, \tag{9}$$

while the BN layer formula is expressed as

$$BN(x) = \gamma * \frac{(x - mean)}{\sqrt{var}} + \beta. \tag{10}$$

Note that BN layer is used in each branch, the convolution results into BN formula are substituted as

$$BN(CONV(x)) = \gamma * \frac{W(x) + b - mean}{\sqrt{var}} + \beta. \tag{11}$$

After further simplification, the formula is expressed as

$$BN(CONV(x)) = \frac{\gamma * W(x)}{\sqrt{var}} + (\frac{\gamma * (b - mean)}{\sqrt{var}} + \beta). \tag{12}$$

Comparing with formulation (2), BN layer fused with convolution actually is regarded as a convolution layer. Since the weight needs to consider the parameters of BN layer, formally $W_{fused} = \gamma * W/\sqrt{var}$ is used to denote the weight after fusion and $B_{fused} = (\gamma * (b - mean))/\sqrt{var} + \beta$ expresses bias. Therefore the final fusion result is expressed as

$$BN(CONV(x)) = W_{fused}(x) + B_{fused}. \tag{13}$$

The fused layer is more efficient and facilitate the subsequent fusion of multiple branches.

3.4.2 Fusion of multi branch structures

For multi branch merge, we further consider the fusion of convolution layers with different kernel size. Currently, the most common convolution layer uses a stack of many spatial convolutions with 3×3 kernel sizes to increase the receptive fields. The process of convolution calculation of the input image is expressed as $O = I \otimes K + B$. Where O denote the output, I, K, B represent input, kernel size and bias. As shown in Fig. 1, a 3×3 kernel can be get by

adding the 1×1 kernels onto the central point of 3×3 kernel. The identification branch is likewise affected by this transition. A 1×1 conv with an identity matrix acting as the kernel is used to conceptualize an identity. So after transformation, three 3×3 kernels are obtained. In general, multi branch fusion is expressed as

$$\begin{aligned} O &= (I \otimes K_1 + B_1) + (I \otimes K_2 + B_2) + (I \otimes K_3 + B_3) \\ &= I \otimes (K_1 + K_2 + K_3) + (B_1 + B_2 + B_3), \end{aligned} \tag{14}$$

Through the above algorithm, a multi branch block is fused into a single branch block.

3.5 ball detection

The network outputs the detection heat map, and the continuous value of each pixel is within the range of $[0, 255]$. After a softmax layer a complete continuous detection heatmap is generated. Let the ball's center be at (x_0, y_0), and the heatmap function is written as

$$G(x, y) = [(\frac{1}{2\pi\sigma^2})e^{-\frac{(x-x_0)^2+(y-y_0)^2}{2\sigma^2}}(2\pi\sigma^2 \cdot 255)], \tag{15}$$

where σ^2 denotes variance. Since a ball's typical radius is around 5 pixels, σ^2 is set to ten. Then, Hough gradient method is used to find circles on the black and white binary detection heat map. The centroid of the circle is returned if there is only one circle found. Other times, it's thought that the heat map failed to find the ball.

Specifically, note that the softmax function is given by

$$P(i, j, k) = \frac{\mathrm{e}^{L(i,j,k)}}{\sum_{l=0}^{255} e^{L(i,j,l)}}, \tag{16}$$

The thermal map value of the pixel is chosen to be the depth k with the highest likelihood. For each pixel, let

$$h(i, j) = \arg\max_k P(i, j, k), \tag{17}$$

indicates the chosen gray-scale value at and the softmax layer output at (i, j). The following two steps can be used to find the ball's coordinates after the entire continuous detection heat map has been generated. The heat map is first intelligently turned into a binary value heat map in a pixel-by-pixel fashion using the threshold t. A pixel is set to 255 if its value is greater than or equal to t. On the other hand, a pixel is set to 0 if its value is smaller than t. The threshold t is set to 128 in accordance with the earlier discussion on the typical radius of tennis balls. The circle on the heat map for black and white binary detection is located in the second stage using the Hough gradient approach.

Algorithm 1 Rep-Tracker tracking method

Input: Tennis match flames dataset $\mathcal{D}$, validation set $\mathcal{D}_v$ and test date $\mathcal{D}_t$
Output: Predict result **P**

Initializing Rep-Tracker network;
Randomly select flames to form training set $\mathcal{D}^{'} \in \mathcal{D}$ and test set $\mathcal{D}_t^{'} \in \mathcal{D}_t$;
Randomly initialize the weight and bias;
Unify the size of the input flames to match the model;
Normalize these flames;
Start the training process;
if At training stage **then**
    for epoch from 1 to n **do**
        Input A for training
        Utilize back propagation to update weights W and bias b;
    end for
end if
Start the inference process;
if At inference stage **then**
    Deploy the trained network to inference time network using structure re-parameterization
    Input video frames to deployed Rep-tracker
    Obtain heat map B
end if
Locating the position of the ball;
return Ball position in frames

4 Experiments and Analysis

This section introduces the process and details of our experiment. Our experiments tested on the device equip with NVIDIA GeForce RTX 3080 graphics card and the core AMD Ryzen 9 5900HX 3.3GHz. During all training in our experiments, the Adam optimizer was used and the learning rate are set 0.001.

4.1 Datasets and analysis

In experiments two tennis datasets is used from TrackNet[10] which are from the broadcast video of tennis single match at several season. Totally the datasets have have 36,962 flames applied to object detection and tracking for tennis which are spitted into 81 complete game clips from serve to score. All the videos have the resolution of 1280×720 and 30fps frame rate. Each frames following five attributes in label file, 'Frame Name', 'Visibility Class', 'Trajectory Pattern', the coordinate of tennis in the pixel coordinate 'X' and 'Y'. To increase the robustness of our model, frames in dataset are randomly selected different frames to form the training set at each training session. Further implementation details will be presented in corresponding section.

Fig. 3 The frames in the dataset.

4.2 Model compression

Table 2 Network structure of model (a)

Layer	Input size	In-channel/ Out-channel
Conv1	640×360	a/b
Conv2	640×360	b/b
Pool1	-	-
Conv3	320×180	b/2b
Conv4	320×180	2b/2b
Pool2	-	-
Conv5	160×90	2b/4b
Conv6	160×90	4b/4b
Pool3	-	-
Conv7	80×45	4b/8b
Conv8	80×45	8b/8b
UpSample1	-	-
Conv9	160×90	8b/4b
Conv10	160×90	4b/4b
UpSample2	-	-
Conv11	320×180	8b/4b
Conv12	320×180	4b/4b
UpSample3	-	-
Conv13	640×360	8b/4b
Conv14	640×360	4b/4b
Conv15	640×360	4b/4b

Table 3 Network structure of model(b) and model(c)

Layer	Input size	In-channel/ Out-channel
Conv1	640×360	a/b
Conv2	640×360	b/b
Pool1	-	-
Conv3	320×180	b/2b
Conv4	320×180	2b/2b
Pool2	-	-
Conv5	160×90	2b/4b
Conv6	160×90	4b/4b
UpSample1	-	-
Conv7	320×180	4b/2b
Conv8	320×180	2b/2b
UpSample2	-	-
Conv9	640×360	2b/b
Conv10	640×360	b/b
Conv11	640×360	b/b

Layer	Input size	In-channel/ Out-channel
Conv1	640×360	a/b
Pool1	-	-
Conv2	320×180	b/2b
Pool2	-	-
Conv3	160×90	2b/4b
UpSample1	-	-
Conv4	320×180	4b/2b
UpSample2	-	-
Conv5	640×360	2b/b
Conv6	640×360	b/b

Several experiments are performed to find the ingredients that inference model performance, robustness and size. For object tracking task in tennis video, real time tracking is the best result to achieve. In Rep-Tracker, stack of convolution layers constitutes the feature extractor. So for model compression, the structured pruning and unstructured pruning philosophy are leveraged. A series of models are designed to test the effect of the number of network layers

on performance in our test. As shown in Tab.2, comparing with TrackNet, three convolution layers are removed in model(a) which in total composed of 15 convolution layers. The FLOPs of the model drop over 50% through such operation. By further streamlining the model by reducing convolution layers, model(b) and model(c) which composed of 11 and 6 convolution layers is proposed as represented in Tab.3. The FLOPs of the model further down to 78.09G.

During each training session 2000 frames are randomly selected to contain different courts from TrackNet datasets to form the training dataset. Also during testing 500 frames will be selected for test set. As shown in Tab.4, properly reducing convolutional layers has no significant effect on the results. In fact when pruning off more than half of the convolution layer the accuracy only drop less then one percent with FLOPs five times down.

Table 4 Network structure after pruning. Comparison of different stage of models.

Network	Operation	accuracy	GFLOPs	memory cost	MAdd	FPS
TrackNet	none	0.997	368.56	2226.27MB	368.56G	7
model(a)		0.995	159.01	2119.92MB	317.5G	7
model(b)	Network layer design	0.996	129.25	2003.91MB	258.0G	7
model(c)		0.995	78.09	1413.28MB	155.82G	9
model(d)	Input size reduction	0.994	19.52	353.32MB	38.96G	18
model(e)		0.994	7.21	137.99MB	14.37G	21
model(f)	Channel pruning	0.994	5.79	97.12MB	5.79G	26
model(g)		0.995	1.29	76.68MB	2.55G	29
model(h)	Add shortcut connection	0.996	5.61	282.18MB	11.12G	21
Rep-Tracker	re-parameterization	0.996	1.29	76.68MB	2.55G	29

The following further studies discuss the effects of reducing the size of the model input image and channel pruning on the model. In model (d) the size of the image is compressed to half of the original size after inputting, and then entering into the model. In model (e) to (g), on the basis of model (e), the the channel pruning operation is carried out. About 50%, 75%, 90% of the channels of each model is pruned respectively and the training implementation details are the same as before. Tab.4 shows the four stages of the model compress operations. Through network layer design operation, the GFLOPs and memory cost of the model is significantly decrease while accuracy is maintained. The FPS in model inference time have no obvious improvement. So the following experiments focused on the impact of the size of the input flames and the channels of the model. After stage two and stage three, the size of the model is further controlled while ensuring accuracy. Moreover, the FPS of the model comes to 29 which meets the requirements of real-time tracking. Considering the stability and universality of the model, the fourth stage is the residual connection of each layer of the model, which brings additional parameters. After the model training is completed, the model can be converted to a single branch structure, that is, the same model structure at the third stage, by re-parameterization.

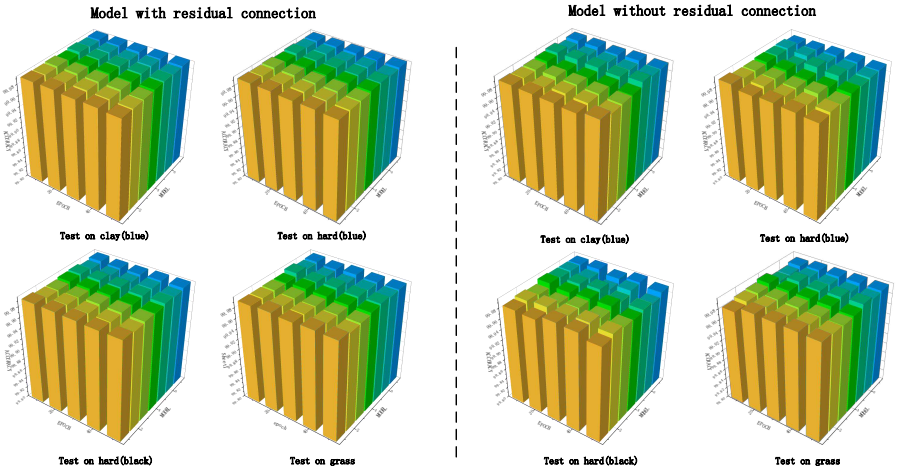

Fig. 4 Results on clay dataset trained 100 epochs.

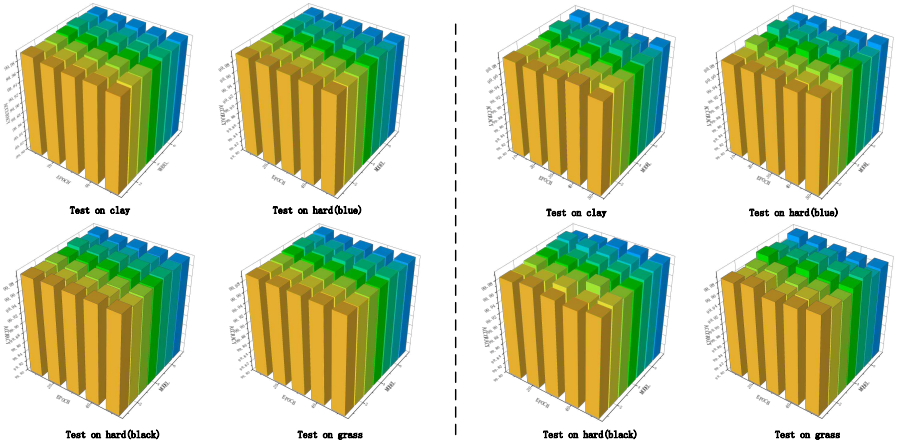

Fig. 5 Results on blue clay dataset trained 100 epochs.

4.3 Anti-interference experiments

CNN-based network often have degradation problem which increase the difficulty of network training and poses robustness problems. To enhance model Anti-interference capability and generalizability without increasing model complexity, inspire by ResNet and Rep-VGG shortcut connections are inserted for each layer.

A series of experiments are conducted to test the robustness of the network in complex environments. The model is trained on a single type of court dataset and test on other type of court. The tennis court can be divided into five specialties, clay, blue clay, blue hard, black hard and grass. In training time 200 frames from corresponding court will be randomly selected to form the training

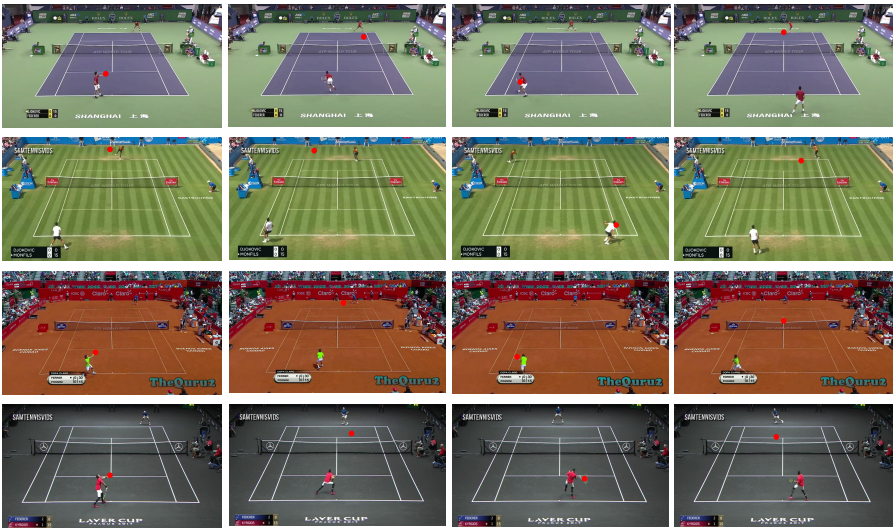

Fig. 6 Tracking result on different court

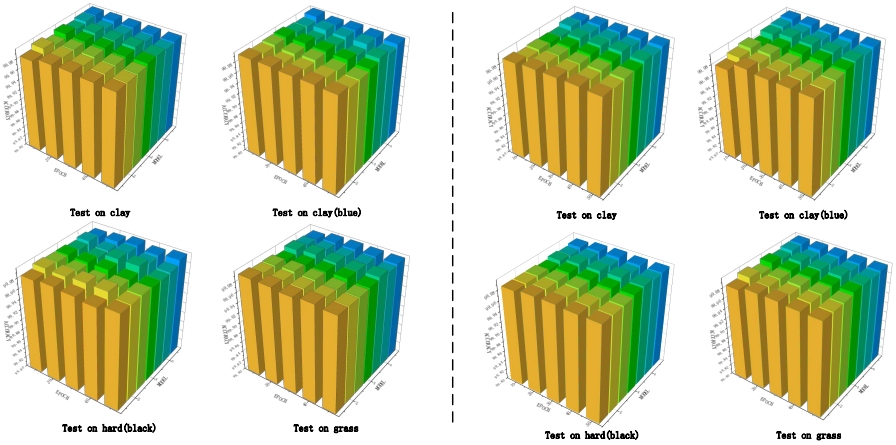

Fig. 7 Results on blue hard dataset trained 100 epochs.

dataset. And 100 frames from every other court will be selected to form the inference dataset. Consider the black hard court only use in the Laval Cup, it won't be used for training. Each type of the court will be test 100 epochs. After training, structure re-parameterization strategy will be used to convert the weight to the form applicable to the single branch model for inference.

As shown in Fig.4, when training on clay datasets, the testing accuracy of the model with residual connection in other courts is more stable comparing with the model without residual connection, especially in grass. When training model on hard court dataset, the testing accuracy on blue clay is relatively

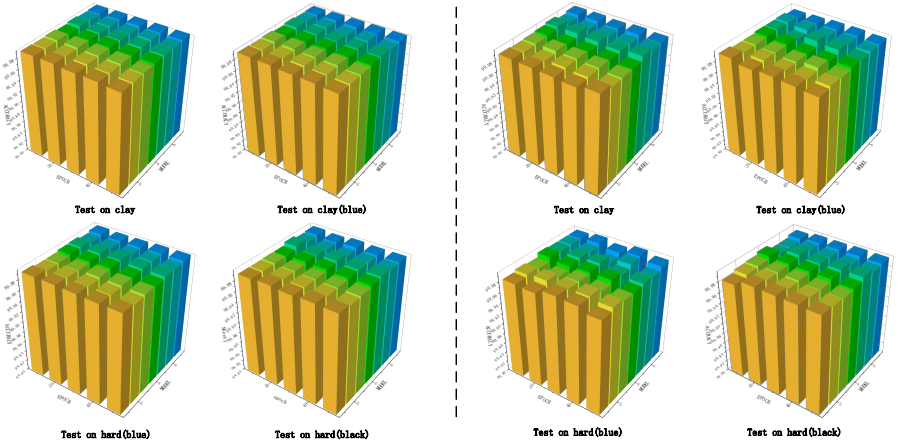

Fig. 8 Results on grass dataset trained 100 epochs.

a. model train on clay

Network	model with residual connection	model with residual connection
blue clay	0.994	0.981
blue hard	0.992	0.921
black hard	0.994	0.963
grass	0.995	0.935

b. model train on blue clay

Network	model with residual connection	model with residual connection
clay	0.992	0.934
blue hard	0.991	0.943
black hard	0.994	0.964
grass	0.994	0.923

c. model train on blue hard

Network	model with residual connection	model with residual connection
clay	0.994	0.924
blue clay	0.996	0.964
black hard	0.992	0.926
grass	0.992	0.922

e. model train on grass

Network	model with residual connection	model with residual connection
clay	0.992	0.936
blue clay	0.993	0.921
blue hard	0.995	0.932
black hard	0.993	0.932

Table 5 Test accuracy after training on different courts

unstable as Fig.7. And when training on grass court dataset, the testing accuracy on blue hard is the most unstable as show in Fig.8. In total, as the results show in Fig.4 to 8 and Tab.5 it is easily be seen that the model with shortcut connection is more stable in accuracy compare with the model without shortcut connection. In tennis match, different courts are not only different in color,

but also different in contrast between courts and balls. In Rep-Tracker, ball detection based on generated heat map. The experimental results show that adding residual connection can improve the effect of generating heat maps. Because of that addition of residual slightly improve the performance of the model in different courts. The Fig.6 shows the tracking result on different court. The Rep-Tracker show acceptable result on different court.

5 Conclusion

In this paper we proposed Rep-Tracker, a lightweight pruned VGG-style based tennis tracking method which use re-parameterization strategy. With pruning and re-parameterization strategy, the method can solve the high hardware requirements of tennis tracking network. Meanwhile, our network achieves satisfactory performance on different courts, shows favorable speed-accuracy trade-off and generalization. In practical applications, the network is able to apply to low computing power devices such as mobile terminals and adapt to different court. Nevertheless, the Rep-Tracker relies on heatmap based localization method, which is prone to false detection in videos with noise. In future work, we will introduce adaptive Consistency Prior to solve this problem.

Competing interests

I declare that the authors have no competing interests as defined by Springer, or other interests that might be perceived to influence the results and/or discussion reported in this paper.

Authors' contributions

Wenjia Wu wrote the main manuscript text and Lingfei Ren prepared figures 1-6 and Wenyuan Yang prepared figures 7-8. All authors reviewed the manuscript.

Funding

This work was supported by the National Natural Science Foundation of China under Grant No. 12101289. This work was supported in part by the Natural Science Foundation of Fujian Province under Grant No. 2020J01821,2022J01891.

Availability of data and materials

All data, models, and code generated and used during the current study are available from the corresponding author on reasonable request.

References

[1] Muzammul, M., Li, X.: A survey on deep domain adaptation and tiny object detection challenges, techniques and datasets. arXiv preprint arXiv:2107.07927 (2021)

[2] Kang, K., Li, H., Yan, J., Zeng, X., Yang, B., Xiao, T., Zhang, C., Wang, Z., Wang, R., Wang, X., *et al.*: T-cnn: Tubelets with convolutional neural networks for object detection from videos. IEEE Transactions on Circuits and Systems for Video Technology **28**(10), 2896–2907 (2017)

[3] He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask r-cnn. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 2961–2969 (2017)

[4] Wu, Q., Shen, C., Wang, P., Dick, A., Van Den Hengel, A.: Image captioning and visual question answering based on attributes and external knowledge. IEEE transactions on pattern analysis and machine intelligence **40**(6), 1367–1381 (2017)

[5] Herath, S., Harandi, M., Porikli, F.: Going deeper into action recognition: A survey. Image and vision computing **60**, 4–21 (2017)

[6] Zhou, B., Zhao, H., Puig, X., Xiao, T., Fidler, S., Barriuso, A., Torralba, A.: Semantic understanding of scenes through the ade20k dataset. International Journal of Computer Vision **127**(3), 302–321 (2019)

[7] Kamble, P.R., Keskar, A.G., Bhurchandi, K.M.: Ball tracking in sports: a survey. Artificial Intelligence Review **52**(3), 1655–1705 (2019)

[8] Jiang, K., Izadi, M., Liu, Z., Dong, J.S.: Deep learning application in broadcast tennis video annotation. In: 2020 25th International Conference on Engineering of Complex Computer Systems (ICECCS), pp. 53–62 (2020). IEEE

[9] Reno, V., Mosca, N., Marani, R., Nitti, M., D'Orazio, T., Stella, E.: Convolutional neural networks based ball detection in tennis games. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops (2018)

[10] Huang, Y.-C., Liao, I.-N., Chen, C.-H., İk, T.-U., Peng, W.-C.: Tracknet: a deep learning network for tracking high-speed and tiny objects in sports applications. In: 2019 16th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), pp. 1–8 (2019). IEEE

[11] Qin, H., Gong, R., Liu, X., Bai, X., Song, J., Sebe, N.: Binary neural networks: A survey. Pattern Recognition **105**, 107281 (2020)

[12] Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)

[13] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1–9 (2015)

[14] Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: International Conference on Machine Learning, pp. 448–456 (2015). PMLR

[15] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2818–2826 (2016)

[16] Szegedy, C., Ioffe, S., Vanhoucke, V., Alemi, A.A.: Inception-v4, inception-resnet and the impact of residual connections on learning. In: Thirty-first AAAI Conference on Artificial Intelligence (2017)

[17] He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)

[18] Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4700–4708 (2017)

[19] Tong, K., Wu, Y.: Deep learning-based detection from the perspective of small or tiny objects: A survey. Image and Vision Computing, 104471 (2022)

[20] Tang, X., Du, D.K., He, Z., Liu, J.: Pyramidbox: A context-assisted single shot face detector. In: Proceedings of the European Conference on Computer Vision (ECCV), pp. 797–813 (2018)

[21] Li, Z., Tang, X., Han, J., Liu, J., He, R.: Pyramidbox++: high performance detector for finding tiny face. arXiv preprint arXiv:1904.00386 (2019)

[22] Hong, M., Li, S., Yang, Y., Zhu, F., Zhao, Q., Lu, L.: Sspnet: Scale selection pyramid network for tiny person detection from uav images. IEEE Geoscience and Remote Sensing Letters **19**, 1–5 (2021)

[23] Leng, J., Ren, Y., Jiang, W., Sun, X., Wang, Y.: Realize your surroundings: Exploiting context information for small object detection.

Neurocomputing **433**, 287–299 (2021)

[24] Voeikov, R., Falaleev, N., Baikulov, R.: Ttnet: Real-time temporal and spatial video analysis of table tennis. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, pp. 884–885 (2020)

[25] LeCun, Y., Denker, J., Solla, S.: Optimal brain damage. Advances in neural information processing systems **2** (1989)

[26] Han, S., Mao, H., Dally, W.J.: Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. arXiv preprint arXiv:1510.00149 (2015)

[27] Chen, T., Zhang, H., Zhang, Z., Chang, S., Liu, S., Chen, P.-Y., Wang, Z.: Linearity grafting: Relaxed neuron pruning helps certifiable robustness. In: International Conference on Machine Learning, pp. 3760–3772 (2022). PMLR

[28] Huang, T., You, S., Zhang, B., Du, Y., Wang, F., Qian, C., Xu, C.: Dyrep: Bootstrapping training with dynamic re-parameterization. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 588–597 (2022)

[29] Dong, C., Liu, L., Li, Z., Shang, J.: Towards adaptive residual network training: A neural-ode perspective. In: International Conference on Machine Learning, pp. 2616–2626 (2020). PMLR

[30] Guo, S., Alvarez, J.M., Salzmann, M.: Expandnets: Linear over-parameterization to train compact convolutional networks. Advances in Neural Information Processing Systems **33**, 1298–1310 (2020)

[31] Ding, X., Zhang, X., Han, J., Ding, G.: Diverse branch block: Building a convolution as an inception-like unit. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 10886–10895 (2021)

[32] Ding, X., Zhang, X., Han, J., Ding, G.: Scaling up your kernels to 31x31: Revisiting large kernel design in cnns. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 11963–11975 (2022)

[33] Iglovikov, V., Shvets, A.: Ternausnet: U-net with vgg11 encoder pre-trained on imagenet for image segmentation. arXiv preprint arXiv:1801.05746 (2018)

[34] Liang, T., Glossner, J., Wang, L., Shi, S., Zhang, X.: Pruning and quantization for deep neural network acceleration: A survey. Neurocomputing **461**, 370–403 (2021)

[35] Blalock, D., Gonzalez Ortiz, J.J., Frankle, J., Guttag, J.: What is the state of neural network pruning? Proceedings of machine learning and systems **2**, 129–146 (2020)

[36] Xu, S., Huang, A., Chen, L., Zhang, B.: Convolutional neural network pruning: A survey. In: 2020 39th Chinese Control Conference (CCC), pp. 7458–7463 (2020). IEEE

[37] Liu, Z., Li, J., Shen, Z., Huang, G., Yan, S., Zhang, C.: Learning efficient convolutional networks through network slimming. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 2736–2744 (2017)

[38] Li, H., Kadav, A., Durdanovic, I., Samet, H., Graf, H.P.: Pruning filters for efficient convnets. arXiv preprint arXiv:1608.08710 (2016)

[39] Chen, T., Goodfellow, I., Shlens, J., et al.: Accelerating learning via knowledge transfer (2016)

[40] Zagoruyko, S., Komodakis, N.: Diracnets: Training very deep neural networks without skip-connections. arXiv preprint arXiv:1706.00388 (2017)

[41] Ding, X., Zhang, X., Ma, N., Han, J., Ding, G., Sun, J.: Repvgg: Making vgg-style convnets great again. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 13733–13742 (2021)

[42] Ding, X., Hao, T., Tan, J., Liu, J., Han, J., Guo, Y., Ding, G.: Resrep: Lossless cnn pruning via decoupling remembering and forgetting. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 4510–4520 (2021)