LIGHT YOLO FOR HIGH-SPEED GESTURE RECOGNITION

Zihan Ni[1], *Jia Chen*[1], *Nong Sang*[1], *Changxin Gao*[1], *Leyuan Liu*[2]

[1]Key Laboratory of Ministry of Education for Image Processing and Intelligent Control,
School of Automation, Huazhong University of Science and Technology, Wuhan 430074, China.
[2]NERCEL, Central China Normal University, Wuhan 430079, China.

ABSTRACT

This paper proposes an efficient model named Light YOLO for hand gesture recognition on the embedded platforms. Light YOLO improves accuracy, speed, and model size, in three aspects. To deal with the small scale gestures in practical applications, we strengthen the YOLOv2 with a spatial refinement module to obtain fine-grained features. To accelerate the refined network, we propose a selective-dropout channel pruning approach to prune the redundancy convolution kernels in the network. Moreover, we introduce a dataset for hand gesture recognition in complex scenes. The experimental results on this dataset show that the proposed Light YOLO significantly improve the YOLOv2 network, i.e., accuracy from 96.80% to **98.06%**, speed form 40PFS to **125FPS**, and size form 250M to **4MB**.

Index Terms— Object detection, gesture recognition, model compression and acceleration

1. INTRODUCTION

Hand gestures recognition has been attracted in the field of human-computer interaction (HCI), such as VR games, service robots. Since those applications are usually built on the embedded platform, the gesture recognition tasks need to be carried out in real-time with limited computational resources (such as GPU, memory, etc). The main task of gesture recognition is to locate gesture regions and classify them in images, and its difficulties lie in the non-rigid and semantic similarity of the gestures, the complexity of the background, and the variability of the illumination, etc. In this paper, we recognize 10 kinds of gestures which are commonly used in HCI as shown in Fig. 1.

One of the critical problem in gesture recognition is that the hand gestures are often small in practical applications. Therefore, lots of previous works focus on this problem. [1] propose a multi-scale Faster R-CNN (MS-FRCNN) network which improves the Faster R-CNN [2] for detecting gestures. The MS-FRCNN incorporates the feature maps from shallower convolutional layers like conv3, conv4 and conv5 when the Region Proposals Network (RPN) generating proposals and performing ROI Pooling. Since the non-shared calculation in the region-wise subnet for each proposals makes Faster R-CNN interfere slow, Sang and Ni [3] propose to use the Region-based fully convolutional network (R-FCN) to recognize the gestures. Note that, all the aforementioned detection networks are two-stage detectors, which always have high accuracy. However, they are not suite for gesture recognition on the embedded platforms, due to its speed and size. In contrast, single-stage detectors like YOLOv2, which are faster and smaller than two-stage models, may be a good solution. Therefore, we take YOLOv2 as the baseline model for gesture recognize. However, even YOLOv2 cannot be directly used in gesture recognition on the embedded platform. On the one hand, YOLOv2 has worse performance on small objects like hand gestures because there are few information for small objects on the very top feature maps. Motivated by this, we enhance the YOLOv2 with a spatial refinement module to obtain fine-grained features. On the other hand, the interference speed of YOLOv2 is not fast enough, and the model size is lager than 250 MB. Fortunately, YOLOv2, as a generic object detection network, has plenty of parameter redundancy for gesture recognition. As a consequence, we can compress and accelerate the YOLOv2.

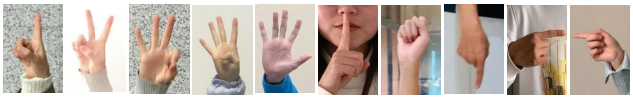

Fig. 1. The 10 kinds of hand gestures in our database.

Modern networks accelerating methods can be mainly classified to three categories: optimized implementation [4], quantization [5], and structured simplification [6, 7, 8]. Optimized implementation based methods usually use the algorithm like FFT to accelerate the implementation of the convolution [4]. Quantization based methods speed-up the network through reducing the numbers of bits to represents the weights [5]. The structured simplification approaches mainly involve: tensor factorization [6], sparse connection [7], and channel pruning [8]. [6] uses the SVD algorithm to decompose each separate filters into a sequence of smaller horizontal and vertical filters to speed up interference, based on the low rank approximation for the convolutional kernels

and cross-channel. But this method needs low rank approximation layer by layer, and thus cannot perform global model compression and requires extensive fine-tuning to achieve convergence [9]. [7] first trains the network to learn which connections are important, and then prunes the low-weight connections below the self-designed threshold to turn the dense network into a sparse network. This method can effectively compress the networks, but have no obvious effect on accelerating. The work in [8] proposes an iteratively two-step algorithm to prune the networks layer by layer, which first selects the unimportant channels based on Lasso regression, and then minimizes the reconstruction errors by the linear least squares. This method needs to estimate the sensitivity for each layer in the networks, which adds extra computations. [10] propose a Taylor expansion criterion to iteratively evaluate and prune the least important feature maps in the network, and then use the backpropagation (BP) algorithm to fine tune the pruned network. This method can prune the network globally without sensitivity estimation, but they prune $N = 1$ feature map at each pruning iteration to ensure the performance, and it will consume high costs to pruning the networks.

To prune the networks faster, we propose a selective dropout channel pruning approach. Pruning too much feature maps at each pruning iteration will bring much damage to the performance of the network, which means that these least important feature maps still hold important information which makes the network rely on them. As the random "dropout" can reduce the network's dependence on the hidden units [11], we propose to retrain the network with a selective dropout module on these feature maps before being pruned, which can reduce the network's dependence on them.

Contribution: we propose a Light YOLO network for high-speed gesture recognition in complex scenes. The main contributions of our work are summarized as follows:

- We propose a spatial refinement module which enhance the accuracy of the YOLOv2 network for small hand gestures.
- We propose a Light YOLO model, which for gesture recognition, an iteratively selective-dropout channel pruning approach to prune the Light YOLO globally.
- Experiments on our proposed dataset show that Light YOLO significantly improves the YOLOv2 network, in terms of accuracy, speed, and size.

2. LIGHT YOLO

In this section, we propose a Light YOLO which can recognize gestures in complex scenes. We first introduce a spatial refinement module to enhance the performance of YOLOv2, then present a selective dropout channel pruning approach to accelerate and compress the refined model.

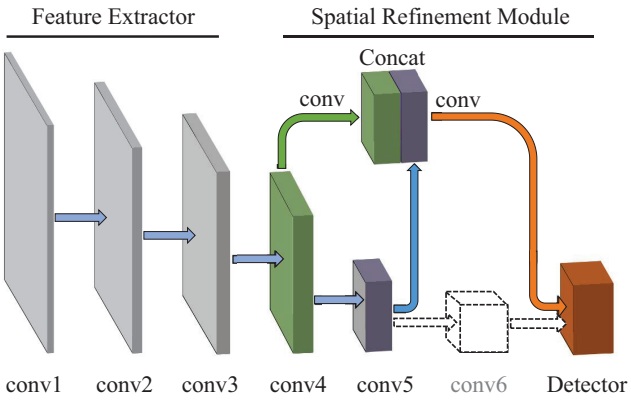

Fig. 2. The spatial refinement module. It first removes the conv6 to make the very top feature maps lager resolution, and use a down-sampling convolutional layer to bring context information from conv4 to the detector.

2.1. Spatial Refinement Module

YOLOv2 is a one-stage detection network which consists of a feature extractor and a detector. The feature extractor extracts deep semantic features from images through six convolutional layer groups and five maxpooling layers. The detector first employs a passthrough layer, which stacks the features from the conv5 on the conv6, to obtain fine-grained feature maps. Then the detector runs on the expanded feature maps, and predicts K anchors at each positions on the top feature maps. YOLOv2 usually has worse performance on small objects such as hand gestures, due to the few information for small gestures on the top feature maps. For example, given a 26×26 pixels hand gesture, there will be less than 1×1 pixels on the top feature maps after five maxpooling layers. What's worse, with the convolutional layers going deeper, the pixels will gather more and more information from outside of the gesture regions which leads to fewer information for the gestures.

To improve the ability of gesture recognition for YOLOv2, we propose a spatial refinement module, as shown in Figure 2. First, we remove the last *maxpooling layer* and the following conv6 to make the network's top feature map higher resolution. The larger size of top feature maps always corresponds to more information and more anchors for gestures, and both of them can improve the performance. We also cut the channels of the convolutional layers in the detector by half, in order to adapt to the channels of the top feature maps.

Second, we replace the passthrough layer with a down-sampling convolutional layer to obtain better fine-grained feature maps. The original passthrough layer directly stack the feature maps from conv5 on the conv6 to form expanded feature maps. But it makes the shallower features which have more parameters domain the high-level features, which makes the model not robust. In this paper, inspired by the downsampling building block in ResNet [12], we downsample the feature maps in conv4 by a 1×1 convolutional layer that have a

stride of 2. And finally we concatenate the fine-grained features with the feature maps in conv5, which provide the access for the detector to the abundant context-information.

2.2. Channel Pruning with Selective Dropout

To effectively prune the networks, we propose a iteratively selective-dropout channel pruning approach, as illustrated in Figure 3. The approach first uses the Taylor expansion criterion to evaluate the importance of feature maps in the refined YOLOv2, then reduces the network's dependence on those least important feature maps by a selective-dropout retraining, and finally prunes the feature maps and fine-tune with the BP algorithm to restore the performance. The details of the method is described as bellow.

First, we use Taylor expansion-based criteria to identify the importance of feature maps in the refined YOLOv2. We regard the channel pruning as an optimization problem, which is to find that prune which feature map can bring the minimal change to the cost function:

$$|\Delta C(h_i)| = |C(D, h_i = 0) - C(D, h_i)| \tag{1}$$

where D denotes the training dataset, $C(\cdot)$ is the cost function of YOLOv2, h_i is the i-th feature maps in the network, and the $C(D, h_i = 0)$ represents the value of the cost function after pruning the feature map h_i.

In order to approximate this value and considering the calculation consumption, we use the first order Taylor expansion formula. And the $C(D, h_i = 0)$ can be regarded as the Taylor expansion at $h_i = 0$:

$$C(D, h_i = 0) = C(D, h_i) - \frac{\delta C}{\delta h_i} h_i + R_1(h_i = 0) \tag{2}$$

To reduce the computational cost, we ignore the first-order remainder:

$$\Theta_{TE}(h_i) = |\Delta C(h_i)| = \left| \frac{\delta C}{\delta h_i} h_i \right| \tag{3}$$

The above formula can be seen as the accumulation of the product of activation and the gradient of the cost function with respect to the activation, which can be easily performed through a forward propagate and a backward propagate. We can get the approximate values of the changes for pruning each feature maps through formula (3).

Second, We choose N feature maps which have smallest Taylor expansion values as the feature maps to be pruned, which we called pruned group. During the retraining stage, we add a selective dropout module after each convolutional layer, and this module will selectively perform dropout on the pruned groups. That is to say, we randomly drop some of the feature maps in the pruned group with a probability of $p = 0.5$. Therefore, the network will reduce the dependence on the pruned groups. The selective-dropout module is shown as Figure 3.

Finally, we prune the convolutional kernels which generate the feature maps in the pruned groups, and fine-tune the pruned networks to recover the performance.

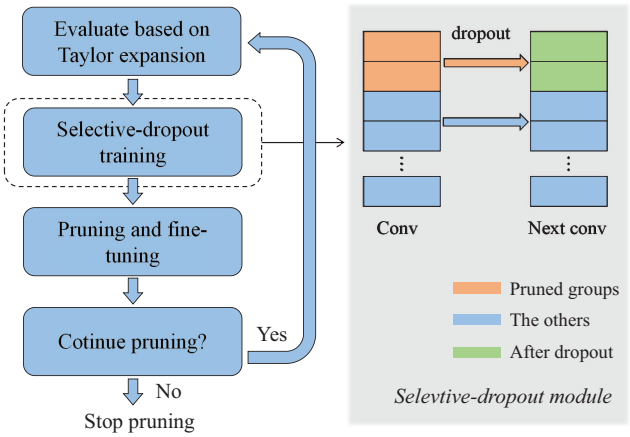

Fig. 3. The selective-dropout channel pruning approach.

Table 1. The gesture recognition results of some popular detection networks on our gesture database. All experiments are evaluated on a GeForce GTX TITAN X (Pascal).

Method	F1(%)	Model(MB)	FPS
Faster-RCNN(VGG)	93.93	513	2.8
R-FCN(Res50)	95.12	97.7	4.2
R-FCN(Res101)	97.63	170	3.2
YOLOv2	96.80	268	40
Light YOLO	**98.06**	**4**	**125**

3. EXPERIMENTS

3.1. Dataset and Measure Method

When collecting the dataset, we consider varieties of complex scenes, such as gesture variations, variable illumination, low resolution, complex background and the gestures across the face. The dataset consists of 5738 samples, including 2868 training samples and 2870 test samples, and the locations and categories has been labeled. In order to take the recall and accuracy all into consideration, we measure the performance of networks with the F measure (F1).

3.2. Setup

First, we fine-tune the YOLOv2 based on the pre-trained ImageNet [13] model. We use a batch size of 8, and start the learning rate at 10^{-4} with a warm-up for the first 100 mini-batch. Then, we decrease it to $5e^{-5}$ at 20 epochs and 10^{-5} at 150 epochs. During training, we perform the data augmentation to balance the samples between classes, and the images are randomly resized to one of the set $(320, 352, ..., 608)$ to make YOLOv2 robust to the size of the input images. We use a single-scale (416×416) testing.

Table 2. Effects of the spatial refinement module.

Method	F1(%)
Remove conv6	97.56
Remove conv6+Downsampling conv	98.04

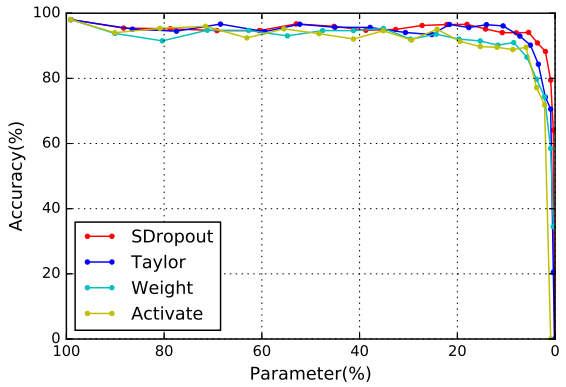

Fig. 4. Pruning the Light YOLO with 10 epochs fine-tuning between pruning iterations. Here, "Weight" represents pruning the kernels which have minimal weights, "Activation" represents pruning the feature maps which have minimal activation, "Taylor" represents Taylor expansion criterion, and "SDropout" is our proposed selective-dropout pruning approach.

3.3. Experiments results

In this section, we first evaluate the performance of our proposed Light YOLO, and then prove the effects of the spatial refinement module and selective-dropout channel pruning approach. We compare our Light YOLO with some popular detection networks, such as Faster RCNN [2], R-FCN [14] and YOLOv2 [15]. Table 1 shows the accuracy, model size and interfere speed of each networks. It can be seen that our Light YOLO achieves the best accuracy among those detection networks, and can process images at **125FPS** and the model size compressed to **4MB**.

In order to prove that the spatial refinement module improves the performance for gesture recognition, we perform the experiments under different settings, and record the results in 2. We can see that removing the conv6 can get 0.7% improvement in accuracy. By replacing the passthrough layer with a downsampling convolutional layer, the detector in Light YOLO can obtain better fine-grained features which increased the accuracy by another 0.5%.

We prune the refined network based on some popular pruning criteria, as shown in Figure 4. At each pruning iterations, we remove 256 feature maps with fine-tuning between iterations. For selective dropout approach, we retrain for 10 epochs with a probability of $p = 0.5$, and fine-tune for another 10 epochs after pruning. For the other criteria, we fine-tune 20 epochs after pruning. We can see that our proposed pruning approach achieves the best accuracy for nearly entire range of pruning ratio, which means the selective-drop module can reduce the change of cost function which introduced by pruning. At the end of the pruning, we fine-tune for more 50 epochs to recover the performance, and the accuracy of the pruned networks are shown in the 3. It shows that our selective-dropout pruning approach achieves the best accuracy which is nearly not decreasing compared with the Light YOLO. Figure 5 shows some gesture recognition results based on the pruned Light YOLO, we can see that the pruned model maintains the performance for recognizing the gestures in complex scenes.

Table 3. The accuracy of pruned networks.

Criteria	Weight	Activation	Taylor	SDropout
F1(%)	85.09	88.54	97.67	**98.06**

Fig. 5. **Some gesture recognition results on our database based on the Light YOLO.** Our approach can recognize the gestures from complex scenes such as gestures across the face, illumination variation, complex background.

4. CONCLUSION

In this paper, we present a Light YOLO network for high-speed gesture recognition in complex scenes. The experiments conducted on our dataset show that our spatial refinement module can enhance the performance of the YOLOv2 for recognizing gestures, and the selective dropout channel pruning approach can prune the refined network with no drop in accuracy. Our Light YOLO network achieve the best accuracy compared with other detection networks, with the interfere time speed up to 125FPS and the model size compressed to 4MB.

Acknowledgement. This work was partially supported by the Fundamental Research Funds for the Natural Science Foundation of Hubei Province No.2017CFB504.

5. REFERENCES

[1] T. H. N. Le, C. Zhu, Y. Zheng, K. Luu, and M. Savvides, "Robust hand detection in vehicles," in *International Conference on Pattern Recognition*, 2017, pp. 573–578.

[2] 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*, 2015, pp. 91–99.

[3] N. Sang and Z. Ni, "Gesture recognition based on r-fcn in complex scenes," *Journal of Huazhong University of Science and Technology(Natural Science Edition)*, vol. 10, pp. 54–58, 2017.

[4] N. Vasilache, J. Johnson, M. Mathieu, S. Chintala, S. Piantino, and Y. LeCun, "Fast convolutional nets with fbfft: A gpu performance evaluation," *arXiv preprint arXiv:1412.7580*, 2014.

[5] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," in *European Conference on Computer Vision*. Springer, 2016, pp. 525–542.

[6] M. Jaderberg, A. Vedaldi, and A. Zisserman, "Speeding up convolutional neural networks with low rank expansions," *arXiv preprint arXiv:1405.3866*, 2014.

[7] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," in *Advances in neural information processing systems*, 2015, pp. 1135–1143.

[8] Y. He, X. Zhang, and J. Sun, "Channel pruning for accelerating very deep neural networks," in *International Conference on Computer Vision (ICCV)*, 2017, vol. 2, p. 6.

[9] Y. Cheng, D. Wang, P. Zhou, and T. Zhang, "A survey of model compression and acceleration for deep neural networks," *arXiv preprint arXiv:1710.09282*, 2017.

[10] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz, "Pruning convolutional neural networks for resource efficient transfer learning," *arXiv preprint arXiv:1611.06440*, 2016.

[11] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv preprint arXiv:1207.0580*, 2012.

[12] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[13] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, and Berg A. C., "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[14] J. Dai, Y. Li, K. He, and J. Sun, "R-fcn: Object detection via region-based fully convolutional networks," in *Advances in neural information processing systems*, 2016, pp. 379–387.

[15] J. Redmon and A. Farhadi, "Yolo9000: better, faster, stronger," *arXiv preprint*, vol. 1612, 2016.