Poker Watcher: Playing Card Detection Based on EfficientDet and Sandglass Block

Qianmin Chen, Eric Rigall, Xianglong Wang, Hao Fan, Junyu Dong
Department of Computer Science and Technology
Ocean University of China
Qingdao, China
{cqm, wangxianglong}@stu.ouc.edu.cn, rigall1214397@163.com, {fanhao, dongjunyu}@ouc.edu.cn

***Abstract*—We present a neural network to detect playing cards in real poker scenes through a camera, where the playing card area represents only 0.7% of the shot table area. In the acquired images, the suits of cards are fuzzy and difficult to identify, even to the naked eye. Because of the relatively few pixels corresponding to the cards, traditional image processing and pattern recognition methods struggle to detect them. Therefore, we use deep learning methods to detect, which have shown to be easy-to-use, faster and more accurate in a broad range of computer vision applications over the years. Inspired by the sandglass block, we improved the current state-of-the-art neural network architecture for object detection, EfficientDet, to retain more features. Experiments have been conducted to evaluate the performance of our improved EfficientDet model and showed that it achieved considerable performance improvement compared with the other deep learning models.**

***Index Terms*—Card detection, EfficientDet, Sandglass block, efficient architecture design**

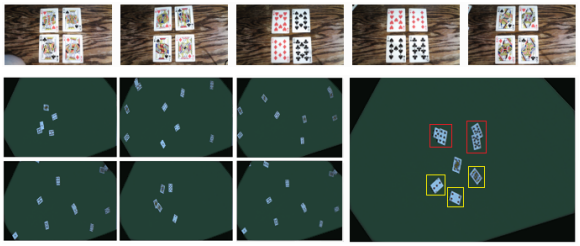

Fig. 1. Image samples of playing cards typically used in traditional digital image processing(top). Compared with the former, the playing cards in real poker scenes have a smaller image proportion, a larger rotation range, and random occlusions(bottom). We simulate a real poker scene in our dataset, with playing cards lying scattered on the poker table, with random placement directions(yellow) and varying degrees of overlap and occlusion(red).

I. Introduction

In a real poker scene, playing cards occupy a small area of the table. When a camera acquires images of the entire poker table, it is difficult to recognize the suits and ranks of these cards. The traditional digital image processing methods perform modeling and matching for each card suit and rank individually, and require additional information of the scene such as the brightness level and orientation angle of the images. Therefore, they are limited in the detection of multiple playing cards that are rotated and tilted or even occluded by each other on the table surface.

In recent years, image classification based on deep learning has achieved tremendous success, which has promoted active research on deep learning techniques for object detection [1]. Object detection can be grouped into two categories: (i) "two-shot detection", such as in Region-based CNN (R-CNN) [2] and (ii) "single-shot detection", such as in YOLO [3]. On the one hand, two-shot detection approach frames the detection task as a "coarse-to-fine" process. Two-shot detectors commonly achieve better detection performance and report state-of-the-art results on public benchmark datasets, but require longer processing time. On the other hand, single-shot detection approach frames it as a "complete in a single step" process. Single-shot detectors are significantly more time-efficient and have greater applicability to real-time object detection.

In this paper, we aim at developing a playing card detector, which is robust enough to have great performance in any rotation of cards, as well as on cards partially occluded. Considering that our playing card detection is performed in a real poker scene, we must pay more attention to speed while ensuring good accuracy, so we focus our study on the latest single-shot network. Apart from other model architectures designed by human experts in the field, research advances in deep learning has developed the concept of neural architecture search (NAS), firstly introduced in [4]. EfficientNet [5] uses neural architecture search, and by using compound scaling on depth, width and resolution, balances the scaling ratio of the three dimensions. The model has attracted wide attention due to its better, faster and more economical advantages. The object detection network EfficientDet [6], also based on neural architecture search, has emerged at its historic moment. It achieves state-of-the-art 51.0 mAP on COCO dataset. Inspired by the novel sandglass building block [7], retaining more high dimensional features, we replace the basic building block in the EfficientDet structure with this block, to improve the network accuracy and speed. The comparative experiments in section IV show that our improved model is effective and reasonable.

* Hao Fan is the corresponding author.

II. Related Work

A. Traditional image processing and pattern recognition

Playing card detection is a specific application of object detection: it processes real-time images of a poker table, which contains card recognition areas. The core of the traditional technology is based on image processing and analysis, including image transformation, image enhancement, image compression, image restoration and region segmentation. Pimentel [8] employed a traditional method based on the Hough transform and applied it to the detection of cards as rectangle-shape objects [9]. The SIFT or SURF algorithms can be used to search for card indexes in the images (ranks and suits symbols) [10].

Different from the scenes with a single card picture, our work focuses on scenes where several cards lie on the table. To tackle this problem, algorithms require more preprocessing of raw images, such as enhancing the contrast between the cards and the poker table [11], using Canny or other similar operators for edge detection, setting a standard rectangular shape around the playing cards to filter out the unwanted edges and also rectifying the position of the camera.

Another major research field for playing card detection also refers to pattern recognition methods. Pimentel [8] compared the performance in playing cards detection of template matching, the probabilistic rigid model and the probabilistic deformable model, whose efficiency is greatly influenced by illumination conditions. Zhou *et al.* [12] developed the playing card recognition software based on a BP neural network and correlation method, but the cards ranks and suits must be extracted and segmented separately. Although the pattern recognition technology for card detection is quite advanced, the detectors are only reliable when cards are separated from each other [13]. In addition, since a step-by-step design is required for specific objects, this technology becomes less practicable and relevant under such constraints, compared to more data-driven oriented approaches like deep learning.

B. Detection with deep learning

As the performance of handcrafted features became saturated, in 2012, the emergence of convolutional neural networks (CNN) [14] has replaced them. M. Castillo *et al.* [15] tested various typical machine learning models for playing cards recognition and compared them to deep learning models based on CNN. Their experiments showed that CNNs work much better in terms of variety of zoom, rotation and angle in the images. Deep convolutional networks are able to learn robust and high-level feature representations of images and have led to the rapid development of new model designs for object detection.

YOLO [3] was the first single-shot detector in deep learning. Simple but efficient, YOLO detector processes the full images through a single neural network: the network divides the images into regions and predicts simultaneously the bounding boxes and the relative probability of the detected objects in each region. Although the original YOLO has relatively low localization accuracy, subsequent works [16], [17] have further improved the detection accuracy while maintaining very high detection speed.

Currently, EfficientDet [6], following the design of single-shot detectors, achieves state-of-the-art accuracy. EfficientDet implements EfficientNet [5] backbone network, based on neural architecture search [4], [18], [19], which automatically searches for the most effective network architecture. The design of an efficient backbone network is fundamental for ensuring the overall model accuracy performance.

As a major component of EfficientNet, the inverse residual blocks map the features from high-dimensional to a low-dimensional representation (i.e. bottleneck), making its structure lighter. However, the bottleneck block compression will inevitably cause a loss of information in the feature mapping output. Moreover, it will also weaken the cross-layer propagation of the gradients, and cause gradient confusion due to the reduced feature dimension [20], which affects both the training convergence and performance of the model [21]. Different from the inverted residual block which establishes residual shortcuts between bottlenecks, a new bottleneck design was recently proposed, called sandglass block [7]. This block places residual shortcuts between high-dimensional feature representations, retaining more information between blocks, and therefore propagates more gradients across the network [21]. In this way, its structure explores a wider search space, thereby optimizes better the network training.

III. Method

A. Main building block in EfficientDet

EfficientDet follows the single-shot detector paradigm. It is composed of EfficientNet [5] backbone network, BiFPN [6] feature network, along with a shared class/box prediction network.

As the backbone network, EfficientNet is based on scaling up MobileNet [22], [23] with neural architecture search, which achieves much better accuracy and efficiency than previous ConvNets. Mainly based on the mobile inverted bottleneck MBConv [19], [23], EfficientNet inherits its advantages and disadvantages, which directly affects its performance. On the one hand, the MBConv block is based on depth-wise separable convolution operation, decomposed into a depth-wise convolution and a point-wise convolution. This operation contributes to reduce the amount of calculation in the convolution process and further reduces the overall model complexity. More specifically, to reduce the computational cost, it takes a low-dimensional compressed feature tensor as input and expands it to high-dimensional space through a point-wise convolution. The depth-wise convolution is then applied for spatial context encoding, following another point-wise convolution that produces a compressed low-dimensional feature tensor as the next block input. This unique structural design can improve efficiency without causing much performance degradation. On the other hand, in this mobile inverted bottleneck MBConv, the shortcut connections between inverted residual blocks connect the low-dimensional features,

obtained by compressing the high-dimensional features from the previous residual block. This compression process may lose useful information, causing moderate results.

B. Sandglass block in EfficientDet-S

In light of the limitations of the MBConv block, a new sandglass block [7] structure solves the above problems by flipping the idea of inverse residuals.

Different from the design principle stating that the inverted residual block keeps expanding the features first and then compressing them, the sandglass block rethinks the positions of expansion and reduction layers. It reorders the two point-wise convolutions with the depth-wise convolutions to ensure that the shortcut connects the high-dimensional representations together.

Let $\mathbf{F} \in \mathbb{R}^{H\times W\times M}$ be the input tensor, $\mathbf{G} \in \mathbb{R}^{H\times W\times M}$ be the building block output tensor. For simplicity, we assume that the input and output of this block share the same number of channels M and resolution rate $H \times W$ and we do not consider the depth-wise convolution and activation layers. The formula for the sandglass block can be written as (1) and for the inverted residual block as (2):

$$\mathbf{G} = \Phi_e(\Phi_r(\mathbf{F})) + \mathbf{F} \tag{1}$$

$$\mathbf{G} = \Phi_r(\Phi_e(\mathbf{F})) + \mathbf{F} \tag{2}$$

where Φ_e and Φ_r represent the point-wise convolutions for channel expansion and reduction, respectively. In this way, we can keep the bottleneck in the middle of the residual path to save parameters and computational costs. More importantly, this allows us to use shortcut connections to connect representations of a large number of channels instead of connecting the bottlenecks. Compared to the inverse residual block, the "wider" shortcut provides more information from input $\mathbf{F}$ to output $\mathbf{G}$ and allows more gradients to be passed across multiple layers.

Point-wise convolutions can be used to encode information between channels, but they cannot capture spatial information. The inverse residual block puts a depth-wise convolution between the point-wise convolutions to learn expressive spatial context information. In this way, the bottleneck between blocks have fewer output channels and encode less spatial information. The sandglass block follows the previous mobile network [22] and uses depth-wise separable convolution to encode spatial information.

The bottleneck in the sandglass block is placed in the middle of this block, as shown in Fig. 2. The depth-wise convolutions, added at the end of the block residual path, have high-dimensional representations, which are connected together by the residual shortcuts. Thereby, the sandglass block with bottleneck structure can encode even more expressive spatial information. Experiments [7] show that, compared with MobileNetV2, this structure achieves an accuracy improvement of 1%.

In order to keep intact the high-dimensional output used for classification, activation layers are only added after the

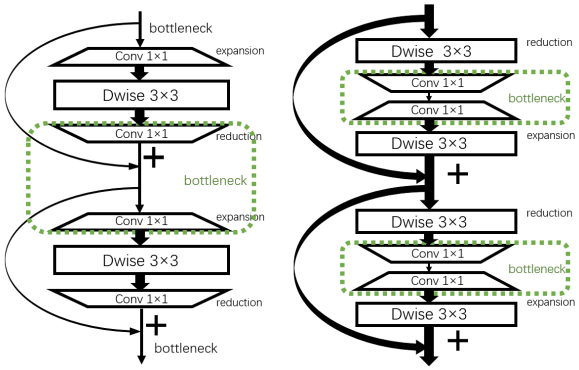

Fig. 2. Different structures of residual bottleneck blocks. The inverted residual block in EfficientDet backbone(left). The novel sandglass block structure(right). The thickness of lines represents the relative number of channels going through them.

first depth-wise convolutional layer and the last point-wise convolutional layer in the sandglass block. We noticed that MBConv uses the Swish activation function [24] in the original structure of EfficientNet. The mathematical formula of this activation function is very simple, that is, $f(x) = x \cdot sigmoid(x)$, which has a lower bound but no upper bound like ReLU. The Swish function can be regarded as a smooth function between the linear function and the ReLU function. When the input x $<$ 0, the gradient of the ReLU function is 0 resulting in a complete loss of information, while the non-monotonicity of the Swish function can still maintain a meaningful output. The simplicity of Swish function and its similarity to ReLU allow practitioners to easily replace ReLU with Swish units in any neural network. The experiments from Google Brain [24] prove that the Swish function is adapted to local response normalization, and its performance on multiple models is equal to if not better than ReLU. In addition, [25] indicate that the Gaussian Error Linear Unit (GELU) performs better than Swish in their model. $f(x) \approx x \cdot sigmoid(1.702x)$, is the most similar GELU function form to the Swish function, with an additional fixed coefficient. In this way, this approximation of GELU function behaves like the Swish function, while keeping the specificities of the original GELU function, that is $f(x) = x \cdot \frac{1}{2}[1 + erf(\frac{x}{\sqrt{2}})]$. As shown in Table. III, we have done ablation experiments which show that GELU is faster but Swish is superior. Therefore, we used the Swish activation function in the sandglass block of our network model.

The EfficientDet-S backbone network is formed by stacking sandglass blocks, whose structure is illustrated in Table. II. The authors of [7] did not use the search space strategy, and directly replaced the inverse residual block with the sandglass block on the EfficientNet-b0 architecture, which improved the classification accuracy by 0.4%.

Our improved EfficientDet-S detector employs EfficientNet with sandglass block as the backbone network. We maintain the original structural design. Its feature network is BiFPN, which takes level 3-7 features {P3, P4, P5, P6, P7} from the backbone network and repeatedly applies bidirectional feature

TABLE I

BOTTLENECK BLOCK DESCRIPTION IN EFFICIENTDET AND EFFICIENTDET-S. 'T' IS THE CHANNEL REDUCTION RATIO WHILE 'S' REPRESENTS THE DEPTH-WISE CONVOLUTION STRIDE. THE SWISH ACTIVATION FUNCTION [24] FROM THE EFFICIENTNET BACKBONE REPLACES RELU6 IN [7].

Block in EfficientDet Backbone

Input	Operator	Output
$H \times W \times M$	1×1 conv2d, Swish	$H \times W \times Mt$
$H \times W \times Mt$	3×3 dwise s=s, Swish	$\frac{H}{s} \times \frac{W}{s} \times Mt$
$\frac{H}{s} \times \frac{W}{s} \times Mt$	1×1 conv2d, linear	$\frac{H}{s} \times \frac{W}{s} \times M$

Block in EfficientDet-S Backbone

Input	Operator	Output
$H \times W \times M$	3×3 dwise, Swish	$H \times W \times M$
$H \times W \times M$	1×1 conv2d, linear	$H \times W \times \frac{M}{t}$
$H \times W \times \frac{M}{t}$	1×1 conv2d, Swish	$H \times W \times N$
$H \times W \times N$	3×3 dwise s=s, linear	$\frac{H}{s} \times \frac{W}{s} \times N$

TABLE II

SANDGLASS BLOCKS IN EFFICIENTDET-S BACKBONE NETWORK STRUCTURE. EACH ROW DESCRIBES AN OPERATION WITH ITS RELATIVE NUMBER OF LAYERS.

stage	operator	resolution	channels	layers
0	conv 3×3	512×512	3	1
1	sandglass block	256×256	16	1
2	sandglass block	128×128	24	2
3	sandglass block	64×64	40	2
4	sandglass block	32×32	80	3
5	sandglass block	32×32	112	3
6	sandglass block	16×16	192	4
7	sandglass block	16×16	320	1

fusion. To sum up, the structure of the EfficientDet-S network employed for our experiments is displayed in Fig. 3.

IV. EXPERIMENT

A. Dataset & Experimental Setup

In our research work, a card can take one of 13 rank values (A,2,...,9,T,J,Q,K), and one of 4 suit values (Clubs, Spades, Hearts, Diamonds). We divided the playing cards into 52 categories (A_C, A_S, A_H, A_D,..., K_C, K_S, K_H, K_D) and discarded the joker cards. We created a Poker dataset having a total of 2981 positive samples with a resolution of 4856 × 3570, including 7677 annotated playing card images. Because the camera keeps a certain angle with the poker table when shooting, the captured images need to be transformed into a front view through perspective projection. This projection results in black triangle areas in the corners of the front view images. An example of the image pre-processing step on our Poker dataset is shown in Fig. 3. Different from the work in [13], our recognition scene considers that the playing cards have the same scale after the projection, thus they are processed in the same way, no matter their distance to the camera. The cards in the picture are face up and placed flat on the table, with variable skewness, whose pixel size ranges from 352 × 200 to 335 × 345, accounting for 0.4% to 0.7% of the entire image size. We use the Poker dataset as the experimental

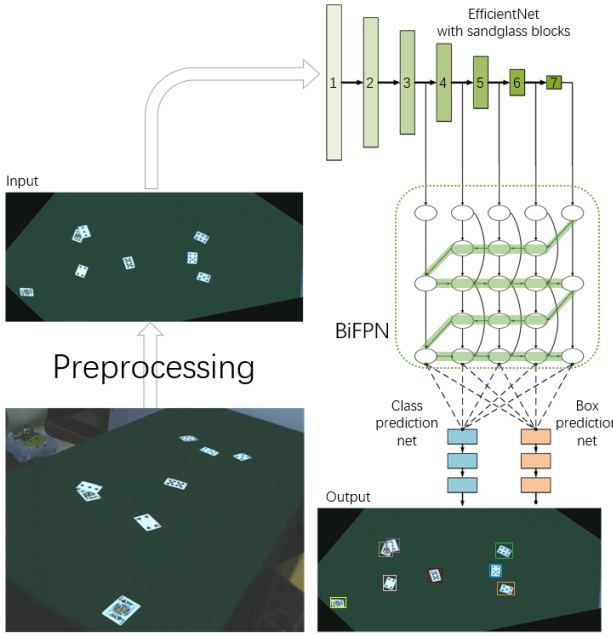

Fig. 3. The whole card detection process. The raw images first go through a perspective projection preprocessing step to get aligned with the front view angle, then are fed into our EfficientDet-S network to predict the cards position in images, as well as their ranks and suits. The EfficientDet-S model is composed of an EfficientNet with sandglass blocks as backbone and a two-layer BiFPN feature network, followed by a three-layer class and bounding box prediction network, based on the configuration of EfficientDet-D0 [6].

data, which is randomly divided into 2477 training samples, 252 verification samples and 252 test samples.

We adopt the PyTorch toolbox 1.4.0 on Ubuntu 16.04 to carry out all our experiments. We use the standard AdamW [26] optimizer with a weight decay factor of 0.95 to train our models. We also employ commonly-used focal loss [27] with α = 0.25 and γ = 1.5, and aspect ratio {1/2, 1, 2}, similarly to EfficientDet. The batch size is set to 8 and two NVIDIA GeForce GTX 2080 GPUs are used for model training. For the training of EfficientDet and EfficientDet-S, the initial learning rate is set to 1e-2.

B. Performance evaluation

Fig. 4 shows a visual performance comparison between EfficientDet on top and EfficientDet-S on bottom, on challenging cases in our Poker dataset. Even for partially missing and occluded card images, EfficientDet-S has still a very satisfactory detection performance. Its prediction confidence is generally higher than that of EfficientDet. Plus, on the first column case, when EfficientDet had a wrong card prediction, EfficientDet-S successfully predicted the right card. We employed two types of evaluation metrics to measure the detection precision and accuracy rate of the compared models. For the detection precision, the average precision (AP) and average recall (AR) on the predicted bounding boxes are the standard evaluation and detection metrics used for COCO dataset.

The results using Swish function, as shown in Table. III, indicate that the AP value of our model prediction

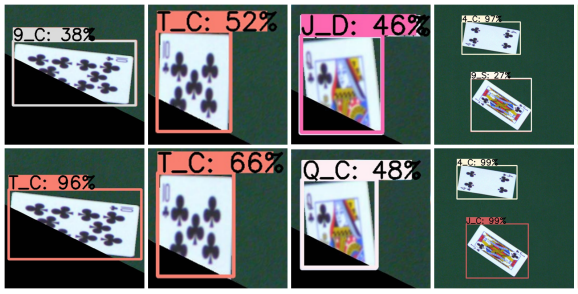

Fig. 4. We enlarged the details in the output images and found that the accuracy of EfficientDet-S(bottom) detection results is generally higher than that of EfficientDet(top).

TABLE III
THE RESULTS OF ABLATION EXPERIMENTS THAT COMPARE DETECTION EVALUATION METRICS BETWEEN EFFICIENTDET-S AND EFFICIENTDET USING DIFFERENT ACTIVATION FUNCTIONS

	EfficientDet-S (Swish)	EfficientDet-S (GELU)	EfficientDet (Swish)	EfficientDet (GELU)
AP (IoU=0.5:0.95)	**91.5%**	89.5%	90.8%	90.9%
AP (IoU=0.75)	**96.4%**	95.1%	95.8%	95.2%
AP (IoU=0.5)	**96.9%**	96.6%	95.9%	95.7%
AR (IoU=0.5:0.95)	**93.3%**	91.6%	93.1%	92.7%
inference time	**0.0715s**	0.0521s	0.0912s	0.0626s

reaches 91.5%, and the AR value reaches 93.3%. Compared with EfficientDet, the detection precision value of the improved EfficientDet-S increased by 0.7% and the recall value increased by 0.2%. We also found that the improved EfficientDet-S processed images faster: the average inference time per image on our GPUs goes from 0.0912s for EfficientDet to 0.0715s for EfficientDet-S. EfficientDet-S can provide expert technical support for real-time card game monitoring. When GELU function replaces Swish, the model runs faster. However, GELU function does not mix well with the sanglass block, and the accuracy is not even as good as with MBConv. Experiments show that the improved EfficientDet-S(Swish) model has a great improvement in terms of precision, recall and inference time on the Poker dataset.

We also focused on the accuracy rate in the detection of all playing cards in the 252 test images, which represents a total of 880 cards. For the particular cases of a few games, such as blackjack, Chinese bridge and baccarat, the card ranks are a more critical factor than the suits in determining the game outcome, therefore we defined additional evaluation metrics to separate the accuracy rate on rank and suit. The evaluation results count 5 indicators, including these absolute accuracy rate, relative accuracy rate, absolute error rate, relative error rate and missed detection rate. We first introduce the indicators used in the model evaluation, and then analyze the experimental results.

Absolutely correct samples refer to the cards detected with the correct suits and ranks.

$$Absolute\ accuracy\ rate = \frac{Absolutely\ correct\ samples}{Total\ number\ of\ samples} \tag{3}$$

Relatively correct samples refer to the cards detected with correct ranks and incorrect suits.

$$Relativeaccuracyrate = \frac{Relatively\ correct\ samples}{Total\ number\ of\ samples} \tag{4}$$

Absolutely wrong samples refer to the cards detected with incorrect suits and ranks.

$$Absolute\ error\ rate = \frac{Absolutely\ wrong\ samples}{Total\ number\ of\ samples} \tag{5}$$

Relatively wrong samples refer to the cards detected with correct suits and incorrect ranks.

$$Relative\ error\ rate = \frac{Relatively\ wrong\ samples}{Total\ number\ of\ samples} \tag{6}$$

Missed samples refer to the cards that were not detected at all.

$$Missed\ detection\ rate = \frac{Missed\ samples}{Total\ number\ of\ samples} \tag{7}$$

In order to verify the feasibility and superiority of the proposed method, a comparative study was conducted on different models to compare its detection accuracy and inference time with Faster-RCNN, YOLOv3 and EfficientDet, whose results are listed in Table. IV. Our improved EfficientDet-S overcomes the other methods in terms of relative accuracy rate, absolute error rate and missed detection rate, while having only one sample error with Faster-RCNN on the absolute correct rate. EfficientDet-S is more accurate in the recognition of card ranks. As for the speed of detection, our improved network is the fastest(FPS). From the qualitative analysis of the output results illustrated in Fig. 5 and Fig. 6, we can see that on occlusion and image border conditions, EfficientDet-S successfully detects the cards, except when cards are more than half occluded, for which none of the four models worked.

CONCLUSION

Detecting small playing cards on an entire large table is difficult and inconvenient using traditional methods. In this paper, we proposed an improved EfficientDet-S lightweight neural network model for card detection. We analyzed the design structure of EfficientDet, and replaced its original residual blocks with sandglass blocks, retaining more high-dimensional features. We also created a dataset of images containing cards on a poker table, named Poker dataset, for our experiments. By conducting comparative experiments with other models, we showed that the proposed network has significant improvement in accuracy, precision and recall, as well as in inference time on our Poker dataset. In brief, we dedicated our work to offer a real-time, efficient and convenient solution for a complex task in a limited domain of applications. We plan to apply the proposed network to a broader range of applications in the future.

TABLE IV
COMPARISON OF DETECTION RESULTS BETWEEN DIFFERENT NETWORKS

	Faster-RCNN	YOLOv3	EfficientDet	EfficientDet-S
Absolute accuracy rate	**851/880=96.70%**	844/880=95.91%	843/880=95.79%	850/880=96.59%
Relative accuracy rate	856/880=97.27%	844/880=95.91%	850/880=96.59%	**859/880=97.61%**
Absolute error rate	8/880=0.91%	23/880=2.61%	11/880=1.25%	**7/880=0.79%**
Relative error rate	9/880=1.02%	**5/880=0.57%**	10/880=1.13%	9/880=1.02%
Missed detection rate	7/880=0.79%	8/880=0.91%	9/880=1.02%	**5/880=0.56%**
FPS	8.009	11.661	10.963	**13.974**

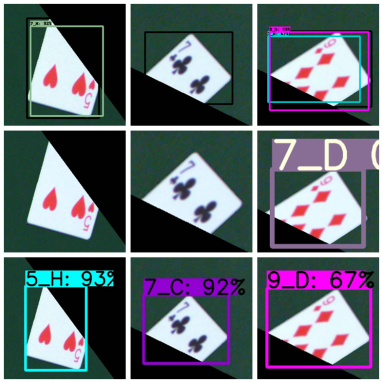

Fig. 5. For the cards on the edges, both Faster-RCNN(top) and YOLOv3(middle) missed them or predicted them incorrectly, while EfficientDet-S(bottom) detected them correctly.

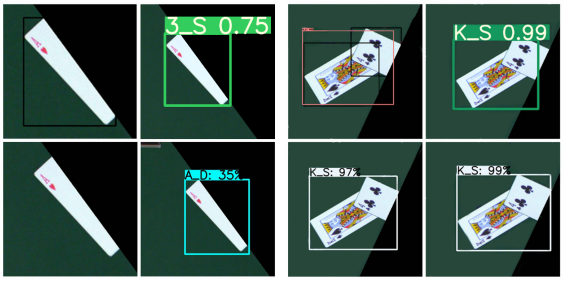

Fig. 6. Detection card results in challenging cases using the four types of networks. The incomplete ace of hearts card on the border(1st column, 2nd column) was completely misidentified by YOLOv3 and missed by both Faster-RCNN and EfficientDet. In contrast, it was detected by EfficientDet-S, though relatively correct: its detected rank is correct but not its suit. Regarding the two stacked cards with a large occlusion area, none of the four methods was successful(3rd column, 4th column).

ACKNOWLEDGMENT

This work was supported by the Natural Science Foundation of Shandong Province of China (ZR2018ZB0852).

REFERENCES

[1] Z. Zou, Z. Shi, Y. Guo, and J. Ye, "Object detection in 20 years: A survey," *arXiv:1905.05055*, 2019.
[2] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *CVPR*, 2014.
[3] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *CVPR*, 2016, pp. 779–788.
[4] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," in *ICLR*, 2017.
[5] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *ICML*, 2019, pp. 6105–6114.
[6] M. Tan, R. Pang, and Q. V. Le, "Efficientdet: Scalable and efficient object detection," in *CVPR*, 2020, pp. 10 778–10 787.
[7] D. Zhou, Q. Hou, Y. Chen, J. Feng, and S. Yan, "Rethinking bottleneck structure for efficient mobile network design," *arXiv:abs/2007.02269*, 2020.
[8] J. P. M. Pimentel, "Machine vision in casino game monitoring," 2011.
[9] D. H. Ballard, "Generalizing the hough transform to detect arbitrary shapes," *Pattern Recognition*, vol. 13, no. 2, pp. 111–122, 1981.
[10] K. Zutis and J. Hoey, "Who's counting? real-time blackjack monitoring for card counting detection," in *ICVS*, 2009.
[11] P. Martins, L. P. Reis, and L. F. Teófilo, "Poker vision: Playing cards and chips identification based on image processing," in *IbPRIA*, 2011, pp. 436–443.
[12] W. Q. Zhou Yimin, Zheng Huaiqiang, "Development & application of playing cards recognition software package," *Computer Simulation*, vol. 20, no. 3, pp. 75–76, 2003.
[13] R. C.Zheng, "Playing card recognition using rotational invariant template matching," *Proceedings of Image and Vision Computing New Zealand 2007*, p. 276–281, 2007.
[14] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, 2012, pp. 1097–1105.
[15] J. W. E. Matias Castillo, Benjamin Goeing, "Computer vision for card games," 2016.
[16] J. Redmon and A. Farhadi, "YOLO9000: better, faster, stronger," in *CVPR*, 2017, pp. 6517–6525.
[17] ——, "Yolov3: An incremental improvement," *arXiv:abs/1804.02767*, 2018.
[18] Z. Guo, X. Zhang, H. Mu, W. Heng, Z. Liu, Y. Wei, and J. Sun, "Single path one-shot neural architecture search with uniform sampling," *arXiv:abs/1904.00420*, 2019.
[19] M. Tan, B. Chen, R. Pang, V. Vasudevan, M. Sandler, A. Howard, and Q. V. Le, "Mnasnet: Platform-aware neural architecture search for mobile," in *CVPR*, 2019, pp. 2820–2828.
[20] K. A. Sankararaman, S. De, Z. Xu, W. R. Huang, and T. Goldstein, "The impact of neural network overparameterization on gradient confusion and stochastic gradient descent," *arXiv:abs/1904.06963*, 2019.
[21] Y. Li and Y. Liang, "Learning overparameterized neural networks via stochastic gradient descent on structured data," 2018.
[22] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv:abs/1704.04861*, 2017.
[23] M. Sandler, A. G. Howard, M. Zhu, A. Zhmoginov, and L. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *CVPR*, 2018, pp. 4510–4520.
[24] P. Ramachandran, B. Zoph, and Q. V. Le, "Swish: a self-gated activation function," *arXiv: Neural and Evolutionary Computing*, 2017.
[25] D. Hendrycks and K. Gimpel, "Gaussian error linear units (gelus)," *arXiv:abs/1606.08415*, 2016.
[26] I. Loshchilov and F. Hutter, "Fixing weight decay regularization in adam," 2017.
[27] T. Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, vol. PP, no. 99, pp. 2999–3007, 2017.