High Performance and Efficient Real-time Face Detector on CPU Based on Convolutional Neural Network

***Abstract*—Face detection is crucial in the development of face recognition, expression, tracking, and classification. Conventional methods have accuracy constraints on several challenging conditions, including non-frontal faces, occlusions, and complex backgrounds. However, the Convolutional Neural Network (CNN) methods produce high performances despite a large amount of computation. Therefore, CNN requires expensive hardware and is not suitable for low-cost CPUs. This study develops a light architecture for a CNN-based real-time face detector. The proposed architecture consists of two main modules, the backbone to extract distinctive facial features and multi-level detection to perform prediction at multiple scales. Furthermore, it utilizes several approaches to enhance the training result, including balancing loss and tweaks on the training configuration. The proposed detector has one stage and is trained using the input of images from WIDER FACE with challenges, which contains more challenging images than other datasets. As a result, the detector achieves state-of-the-art performance on several benchmark datasets compared with the other CPU-based models. Then, its efficiency is superior to that of competitors, as it runs at 53 frames per second on a CPU for Video Graphics Array (VGA) resolution images.**

***Index Terms*—Convolutional neural network (CNN), central processing unit (CPU), face detector, real-time, light architecture.**

I. Introduction

FACE detection is a vision technology that can detect and localize the presence of human faces in an image. In addition, it is an essential method of face recognition, expression, tracking, and classification. For practical application, such methods must operate in real-time [1], [2]. However, approaches with high accuracy tend to be implemented on expensive devices that can provide the require high computation. Lightweight face detection is the solution to a real-time application without loss of the performance of the detector. Therefore, real-time face detection involves a trade-off between speed and accuracy [3]. The usability of such a method will be increased when it can be implemented on a low-cost device or CPU.

A robust real-time face detection method has been introduced to find local features of faces in a whole image region [4]. The conventional method recognizes facial features using Haar-like features and AdaBoost as a classifier. Small faces, occlusions, non-frontal faces, and complex backgrounds are the weaknesses of this method, though it has superior computation efficiency [5]. These problems can be solved by Convolutional Neural Networks (CNN) methods for object detection. The CNN is a powerful method for extracting features and detecting objects [6], [7]. Several CNN architectures are proposed to solve real-time object detection problems [8], [9], [10]. Various anchor scales are assigned to predict the location and size of the bounding box. These networks are references for object detection because they achieve high performance.

Learning-based methods for detecting specific object features of an image show success when assigned to identify facial features [11], [12]. However, the high performance of CNN is decreases efficiency when implemented on the CPU. This problem requires several CNN methods to function only on GPU [13], [14], [15]. High computing resource does not allow the deep CNN to work on a CPU. Hence, CNN with shallow layers is more efficient but shows drastically reduced performance. FaceBoxes [16] and Densely Connected Face Proposal Network (DCFPN) [17] successfully detect faces in real-time on CPU. FaceBoxes uses an architecture CNN to reduce the dimensions and extractions of feature maps in Rapidly Digested Convolution Layers (RDCL). Multiple Scale Convolution Layers (MSCL) predicts multiple scales of faces and achieves excellent trade-off between speed and accuracy. Optimization of this trade-off is the main goal of this study. The proposed architecture contains shallow layers that involve sub-modules and additional strategies to prevent loss of accuracy.

The proposed detector offers CNN-based real-time face detection with several fully convolutional layers. In addition, this work focuses on improving the performance and efficiency of real-time face detection on a CPU. The contributions are as follows:

1) A light architecture for CNN to detect faces that can operate in real-time. This network consists of two main modules, a backbone to extract distinctive facial features and a multi-level detector to predict the location of faces with scale variations.
2) The detector achieves state-of-the-art performance among the CPU real-time detectors tested on several benchmarks, including Annotated Faces in the Wild (AFW) [18], PASCAL face [19], Face Detection Data Sets and Benchmarks (FDDB) [20], and WIDER FACE [21].
3) The efficiency of a detector is superior to that of other competitors, as it runs at 53 FPS on the CPU.

This paper is organized as follows: Section II explains the proposed architecture. Section III describes the strategy and implementation setup of the architecture in the neural network framework. Section IV contains the experiment and results. Finally, conclusions and future work are presented in Section V.

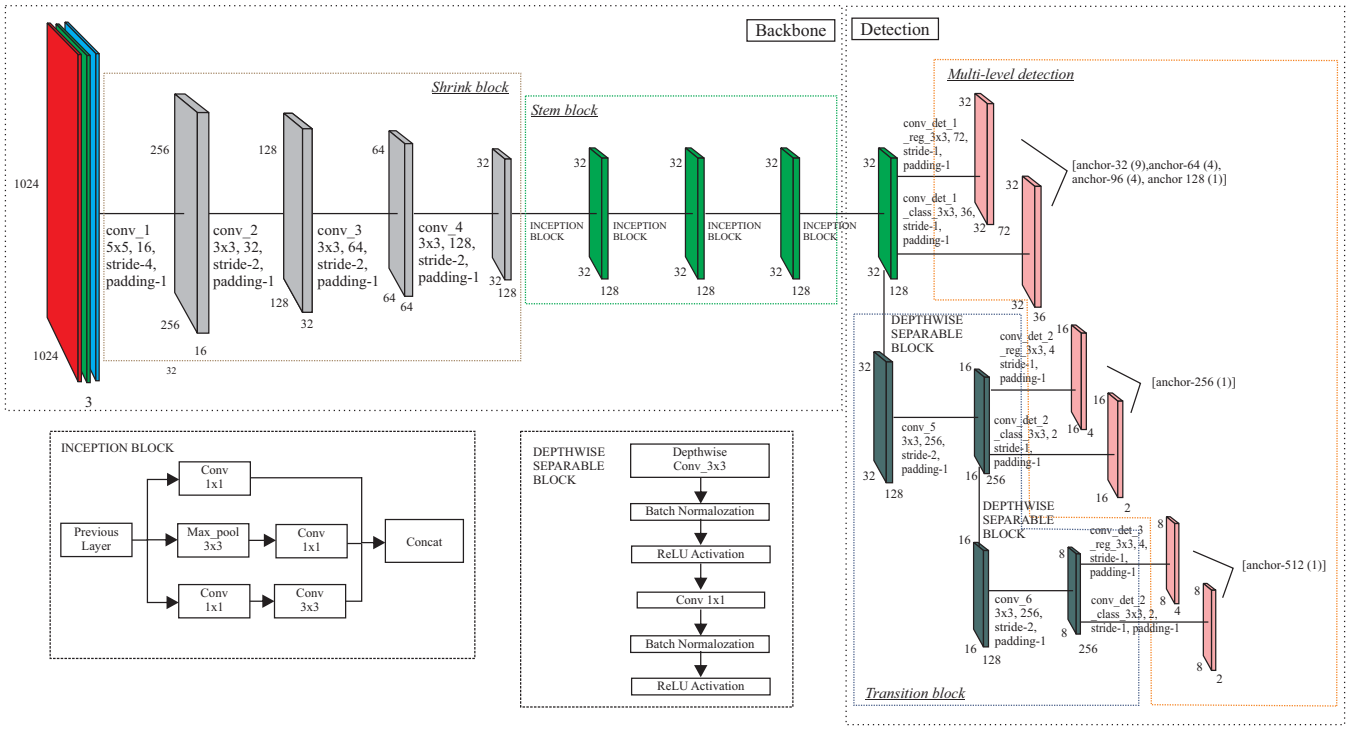


Figure 1. The proposed architecture consists of two main modules of the backbone to extract facial features using CNN and a multi-level detector to predict faces on multiple scales. Inception is applied to the stem block as a crucial module. The architecture consists of 30 layers of CNN and produces one million parameters.

II. Face Detector Architecture

This section discusses the architecture of the real-time face detector. The main modules consist of backbone and predictor, as shown in Fig. 1. The proposed architecture emphasized the improvement of data processing speed and maintained detection accuracy.

A. Backbone Module

The CNN-based feature extraction method for face detection has shown high performance. Rather than a fast data processing speed, deep CNN requires high computation. The detector backbone uses a shallow layer to reduce and extract the spatial input size. The most critical parts of this module are as follows.

1) Shrink block: The shrink block quickly reduces spatial input without losing feature information. The proposed detector shrinks the size and extracts the feature map at the different stages [22]. Hence, this block focuses on reducing the feature map to avoid heavyweight extracted features on large feature maps. All layers in the module are convolution layers aimed at maintaining the quality of feature information before transferring it to the stem block. Four CNN series work to reduce the size of the feature map to 32 times smaller than the input, as shown in Fig. 1. This block uses a 5×5 kernel in the first convolution layer and the 3×3 in the rest. A large filter at the beginning of the process is useful for quickly reducing the feature maps. The stride size of each convolution is 4, 2, 2, and 2, with 128 channels of the last feature map. Batch normalization is used at the end of all convolution operations to accelerate and convergence training. Then, rectified linear units (ReLU) is utilized to activate the output linearly for positive scores.

2) Stem block: The stem block is a crucial module to extract facial features comprehensively. Therefore, this block uses four mini-inception series that are smaller than the original inception. The inception block has shown satisfying results when assigned to extract object features [23]. The receptive field of this block is richer than the standard convolution. However, this module produces light weights to capture different scales of faces. Most works have shown the inception module to be capable of operation in real-time for object detection and classification [24], [25]. Although the detector does not use large kernels or many branches, the stem block can produce a clear feature map. The simple characteristics of facial features support the efficient use of shallow layers. Instead of using one block, a high-performance extraction is achieved. Each block simultaneously processes a set of convolution and pooling operations. Fig. 1 shows that the mini-inception consists of three branches, including a 1×1 convolution, a 3×3 convolution, and a combination series of pooling and 1×1 convolution. Then, a concatenation block plays a role in combining the output of the branches.

B. Detection Module

There are many methods to predict candidates of region proposals with different sizes of the object. The proposed detector implements a pyramidal feature hierarchy using multiple detection layers by grouping predictions according to scale.

1) Transition block: The transition block transforms feature map sizes between prediction layers. This module contains

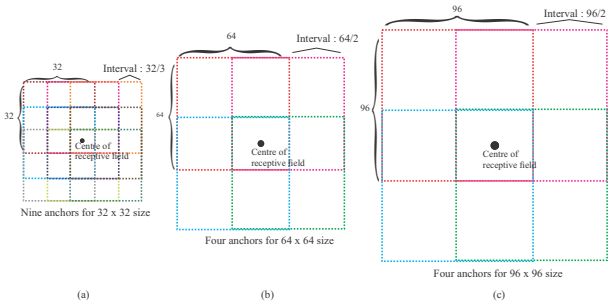


Figure 2. Examples of anchor densification strategies for anchor types are 32 × 32 (a), 64 × 64 (b), and 96 × 96 (c). The anchor unit is in color. The anchor interval shows the distance between neighbor anchors of the same type.

simple blocks to retain information of feature maps. Instead of using pooling to reduce the size of the feature map, convolution with stride two is used for more powerful. The block adopts the depthwise separable convolution [26] as an extractor and a 3 × 3 kernel for decreased feature map size. The depthwise separable convolution utilizes a combination of 3 × 3 depthwise convolution and 1 × 1 pointwise convolution. This combination is simple but more robust than 1 × 1 convolution. The feature map size decreases in inverse proportion to channel size. Finally, the block splits into three layers for prediction at different scales, followed by a single sub-convolution to produce predictive offsets.

2) Multi-level detection: Multi-level detection is used to predicts multiple scales of faces. This module addresses the problem of using a single predictor [8]. A single predictor has a weakness of inconsistency between the fixed receptive field and the objects at different scales. Therefore, this technique ignores information on small face features due to overprocessing in the dimension reduction step. In addition, the Feature Pyramid Network (FPN) requires additional computation [9]. The pyramidal feature hierarchy is used to overcome these problems. The predictor adjusts the size of the feature map and anchor according to face size. Then the module is divided into three levels for the prediction layer. The first layer is responsible for small, the second layer for medium, and the last layer for large faces. The size of the feature map prediction is 32, 16, and 8, respectively. Different anchor types will occupy each prediction layer according to the assignment of the face scale.

C. Anchor Strategy

The proposed detector predicts bounding boxes based on dimensional clusters using anchor boxes. Instead of assigning one type of anchor box, which cannot represent multiple-scale faces, the proposed detector adopts the anchor density strategy to increase the recall of small faces [17]. This strategy applies adjacent anchors of the same type that correspond to intervals based on the center point of the receptive field. In contrast to the original version, anchor scale 96 is employed to reduce the gap of the anchor type on small faces. Various types of anchors with a 1:1 ratio occupy different prediction layers according to scale assignment. Based on this strategy, the detector only implements the first prediction layer to predict small faces by employing small anchors, including the anchor scales of 32, 64, and 96, with anchor numbers of 9, 4, and 4, respectively (see Fig. 2). On the other hand, a single default anchor is assigned to each prediction layer, including 128, 256, and 512, respectively. The strategy is ignored for large-scale anchor types to avoid additional computation.

III. Training Setup and Strategy

In general, the neural network requires comprehensive training to produce high performance. In this section, the loss function, training dataset, augmentation, and parameter settings are introduced to optimize the training process.

A. Balanced Loss Functions

The proposed network generates prediction offsets, including regression (x, y, w, and h) and classes (face and none). The proposed detector adopts the parameterizations of the four coordinates for the regression bounding box [8]. Generally, neural networks use the loss function to quantify the inaccuracy between the predicted value and ground truth. The loss function emphasizes the performance of weight neurons to minimize errors. In object detection, multi-loss is defined as a combination of regression and classification loss. The two objective losses often are imbalanced, so the network tends to work hard on one side. The proposed detector uses a balanced loss to overcome this problem. The loss function associates each error generated by the training network with an anchor box. The loss function of multi-boxes detector is defined as:

$$L(c_i^p, t_i^p) = \frac{\alpha}{N} \cdot \sum_i L_{cls}(c_i^p, c_i^*) + \frac{\beta}{N} \cdot \sum_i L_{loc}(t_i^p, t_i^*), \quad (1)$$

where t_i^p is a set of four coordinate vectors from the predictor location for each i-th anchor and class p, t_i^* is the ground truth box, c_i^p is the prediction class, and c_i^* is the ground truth label. The denominator N is the number of matched default boxes and then can be paired with two parameters, α and β, as the balancer of the two objective losses. $L_{cls}(c_i^p, c_i^*)$ is the softmax-loss as classification loss, defined as [10]:

$$L_{cls}(c_i^p, c_i^*) = -\sum_{i \epsilon Pos} x_i^p Log(c_i^p) - \sum_{i \epsilon Neg} Log(c_i^0), \quad (2)$$

where x_i^p is $[1, 0]$ as an indicator for matching the i-th anchor to ground truth, and c_i^0 is the confidence score for classifiying non-objects. On the other hand, Huber loss defined by [27] is the regression loss detector:

$$L_{loc}(t_i^p, t_i^*) = \sum_{i \epsilon \{x,y,w,h\}} H(t_i^p - t_i^*), \quad (3)$$

in which

$$H(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise} \end{cases} \quad (4)$$

is a robust loss to minimize tuning errors of learning rates during training. This loss function gives a smooth effect when the input value is less than 1. Furthermore, α and β are preset to respectively 2 and 1 to produce optimal training.

B. Training Dataset and Augmentation Data

WIDER FACE is a face detection benchmark dataset used as a training dataset on the proposed detector. This dataset contains 32,203 images, with only 12,800 in the training category for detector knowledge to recognize facial features. Augmentation is used to enrich the variety of training data and prevent overfitting in the training process [17]. This stage consists of sequential processes of color distortion, random cropping, scale transformation, and horizontal flipping. Finally, 1024×1024 is the final size of the training sample after augmentation.

C. Implementation Details

The detector model is implemented in the PyTorch framework, which involves several optimization settings. At the beginning of the training, random weights are initialized at all network layers through end-to-end training. Stochastic Gradient Descent (SGD) is used to optimize weights for each neuron through back-propagation. The optimizer parameters are $5 \cdot 10^{-4}$ weight decay, 0.9 momentum, and learning rate variations at different epochs. The first 200 epochs are applied a 10^{-3} learning rate, the next 50 at a 10^{-4} learning rate, and the last 50 at a 10^{-5} learning rate. The entire dataset is divided into small partitions with a batch size of 32. In the anchor matching process, a set of overlapping anchors is matched with ground truth by the biggest IoU (Intersection over Union). Finally, the best anchor with a threshold of greater than 0.5 is selected.

IV. Experiments

This section discusses the performance and efficiency of the proposed architecture by analyzing the architecture in an ablative study and on the face detection benchmarks, examining the runtime efficiency, and testing it on low-cost devices.

A. Model Analysis

The performance of the proposed detector is comprehensively tested using ablation experiments. The architecture is gradually replaced by module and analyzed for accuracy and speed to determine the strength of each module. This experiment uses the same training strategies except for specified changes to the architecture. Table I shows that almost all proposed modules have a positive impact on improving performance. For example, the multi-level detection module significantly increases accuracy by 5.9% and simultaneously reduces speed by 0.48 ms. This result is more robust than that of single detection of multiple scales of the face. On the other hand, the stem block with a mini-inception significantly increases accuracy by 3.5 ms, but detector performance decreases by 0.6%. In this experiment, mini-inception is replaced by the original inception. This block uses a parallel configuration with various small kernels to increase efficiency. Without compromising a detector speed, the proposed modules improve detector performance.

B. Evaluation on Benchmark

This section presents the evaluation of the proposed detector on benchmarking datasets of AFW, PASCAL face, FDDB, and WIDER FACE. It also compares the performance with that of competitors.

1) AFW dataset: The AFW dataset consists of 203 images with 473 labeled faces from Flickr. This dataset contains a variety of backgrounds and challenging faces based on position, accessories, and expression. The performance result is compared with that of previous works (commercial and research), as illustrated in Fig. 3 (a). The detector performance is superior to others, even 0.28 better than FaceBoxes, the leading competitor. Multiple faces with diverse challenges can be detected, as shown in Fig. 6 (a).

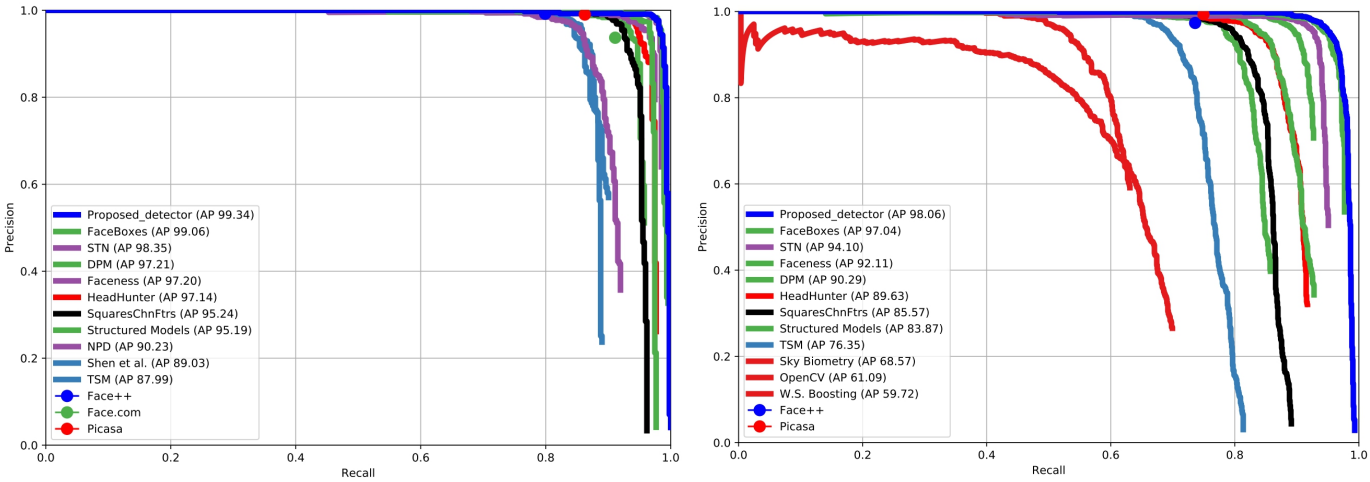
2) PASCAL face dataset: The PASCAL face dataset consists of 851 images with 1,335 labeled faces. This dataset is a subset of the PASCAL VOC dataset containing pose and background variations (indoor and outdoor). Fig. 3 (b) shows that the proposed detector is significantly superior to others. It has 1.02 higher accuracy compared with FaceBoxes. Qualitative results in this dataset are shown in Fig. 6 (b).

3) FDDB dataset: The FDDB dataset consists of 2,845 images containing 5,171 faces of famous people. The dataset collects the images from Yahoo websites and includes various challenges of position, lighting, and background. The proposed detector can overcome these challenges, even with the occlusions (see Fig. 6 (c)). The dataset is evaluated on discrete criteria by comparing the intersection of the detection results and the annotated faces. The score will be one if the ratio is higher than 0.5 and 0 otherwise. The detector achieves state-of-the-art accuracy compared to competitors working on real-time CPUs (i.e., FaceBoxes, DCFPN), as illustrated in Fig. 4. Although HE-ER [13] and SFD [28] have better performance compared to the proposed detector, these detectors are not feasible for real-time performance on a CPU as the deep architecture of CNN produces a large number of parameters.

4) WIDER FACE dataset: The WIDER FACE dataset contains many challenges that are more difficult than other benchmark datasets, such as variations in scales, poses, expressions, occlusions, and lighting conditions. This dataset is divided into training (40%), validation (10%) and testing (50%) sets. The validation and testing sets contain three levels of difficulty, easy, medium, and hard. Hard level means that there are many tiny faces. The proposed detector has a performance of 0.883 (easy), 0.868 (medium), and 0.730 (hard) on validation sets, while the testing sets are 0.883 (easy), 0.863 (medium) and 0.717 (hard). Fig. 5 shows that the proposed detector is more excellent to FaceBoxes, the main competitor on the easy and medium validation and testing sets. It is inferior for the hard category because the low-layer features are not powerful enough to predict tiny faces. Superficial feature extraction result in weak distinction between small faces and background features. The proposed detector is less accurate than ScaleFace [29] on the medium difficulty testing sets, but the competitor is heavy for use on CPU devices (See Table II). Based on the qualitative results in Fig. 6 (d), the detector achieves strong performance when overcoming the challenging images.

Table I
ABLATIVE RESULTS ON FDDB DATASETS. THE MAXIMUM NUMBER OF FALSE POSITIVES IS 1,000. THE SPEED MODEL IS TESTED ON THE VGA-RESOLUTION IMAGES ON THE CPU.

Modules	**Proposed detector**						
Balancing loss	√						
Anchor strategy	√	√					
Transition block	√	√	√				
Multi-level Detection	√	√	√	√			
Shrink block	√	√	√	√	√		
Stem block	√	√	√	√	√	√	
Accuracy (AP)	0.970	0.967	0.964	0.963	0.904	0.895	0.901
Speed (ms)	18.87	18.86	18.20	18.10	17.62	16.93	20.42



(a) (b)

Figure 3. Evaluation on the AFW dataset (a) and the PASCAL face dataset based on Average Precision (AP) (b).

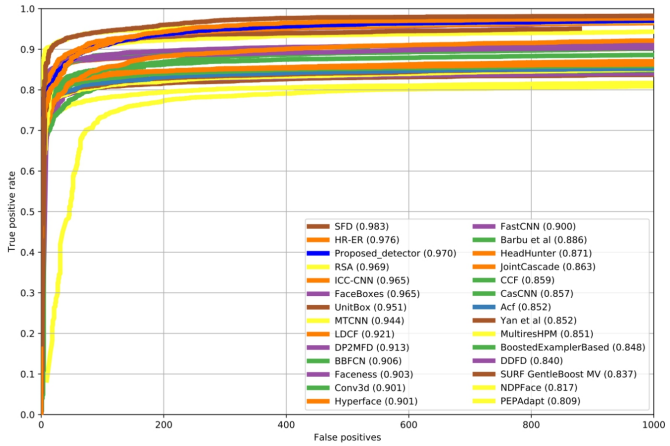


Figure 4. Evaluation on the FDDB dataset based on discrete ROC (Receiver Operating Characteristics) curves.

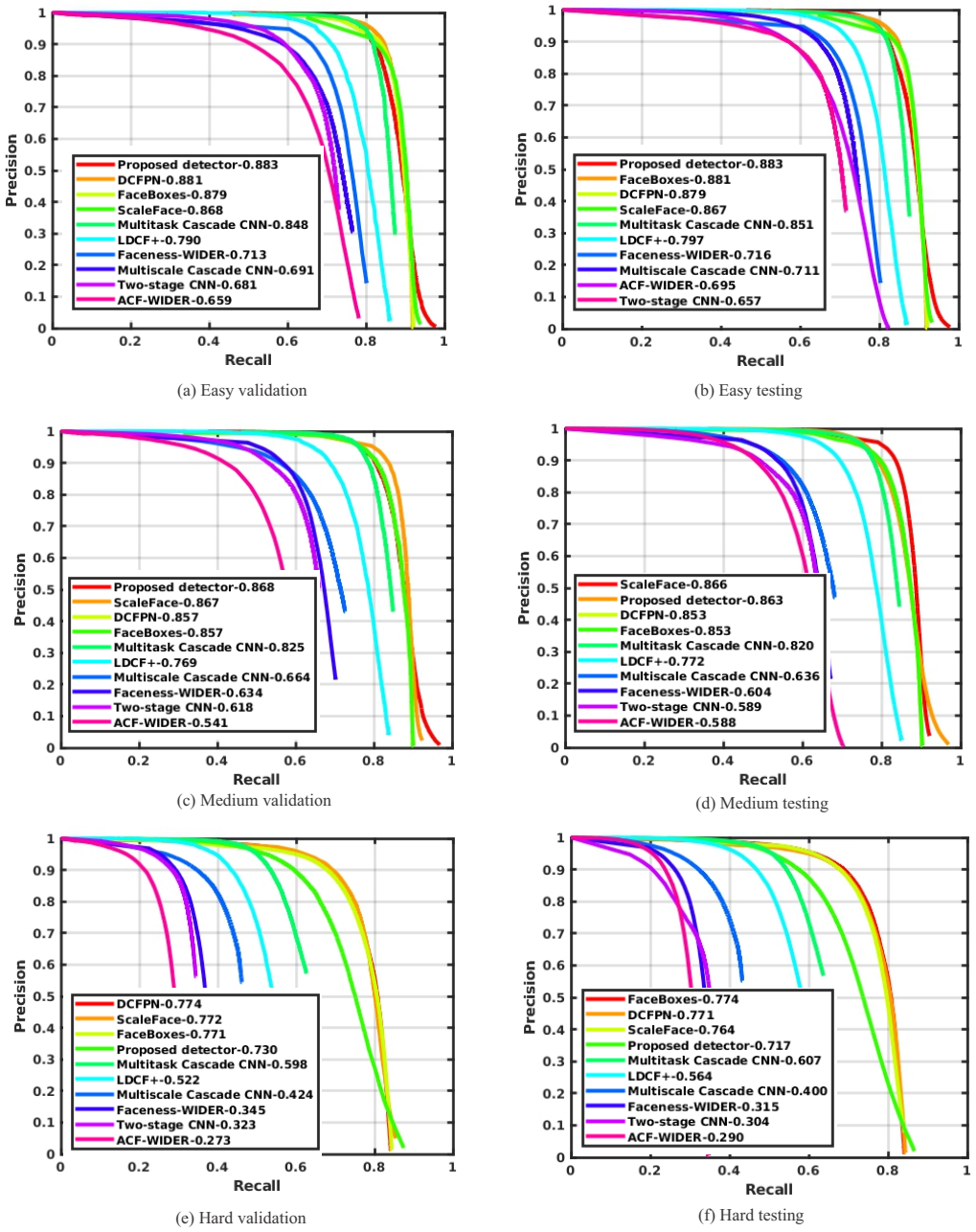
Table II
RUNTIME EFFICIENCY COMPARED TO DIFFERENT ARCHITECTURES. SPEED OF ALL DETECTORS IN VGA-RESOLUTION IMAGES ON CPU.

Architecture	**CPU**	**GHz**	**AP (%)**	**FPS**
ACF	Intel I7-3770	3.40	85.20	20
CasCNN	Intel E5-2620	2.00	85.70	14
FaceCraft	N/A	N/A	90.80	10
STN	Intel I7-4770K	3.50	91.50	10
MTCNN [14]	N/A	2.60	94.40	16
ScaleFace [29]	N/A	N/A	96.00	7
DCFPN [17]	Intel E5-2660v3	2.60	95.40	30
ICC-CNN [30]	N/A	N/A	96.50	12
FaceBoxes [16]	Intel E5-2660v3	2.60	96.50	28
FaceBoxes [16]	Intel I5-6600	3.30	96.50	39
Proposed detector	Intel I5-6600	3.30	97.00	53

C. Runtime Efficiency

In general, CNN-based face detectors can be accelerated using GPU, but the practical applications demand operation on CPUs with lightweight computing. Overall, the proposed architecture generates 989,832 parameters, less than that of the general architecture of CNN for object detection. The detector is tested on a VGA-resolution image (640 × 480 pixels) that is the same as for other competitors. At the 0.05 confidence threshold, a set of positive anchor boxes is selected using a Non-Maximum Suppression (NMS) of 0.3 to produce final bounding boxes. Experiments are conducted on the following hardware: Intel Core I5-6600 CPU @ 3.30 GHz, 8 GB RAM. The detector works in real-time with a processing speed of 53 FPS and achieves state-of-the-art accuracy compared with other CPU real-time detectors. FaceBoxes, as the latest competitor, is slower than the proposed model on the same devices. Table II shows the accuracy of each architecture based on Average Precision (AP) as a metric with 1,000 false positives on the FDDB dataset.

Efficiency comparisons are also performed with different



(a) Easy validation (b) Easy testing

(c) Medium validation (d) Medium testing

(e) Hard validation (f) Hard testing

Figure 5. Evaluation on the WIDER FACE validation and testing sets.

input frame scales based on a standard computer display. The FPS detector graph is proportional to the sizes of the input frame. The small frame size allows higher speed but is detrimental to performance and vice versa. Fig. 7 shows that the proposed detector outperforms FaceBoxes at all input scales. The QQVGA (Quarter Quarter VGA) resolution shows significant differences form the competitor, while the smallest difference is with FULL HD (High Definition) resolution.

D. Real-time Testing on Low-cost Devices

The real challenge of computer vision is the implementation of practical applications. Both performance and speed are essential metrics when a model is working in real-time. However, compared to expensive devices, low-cost devices are widely used for practical applications. Table III shows that the detector is tested on several inexpensive devices with a retail price lower than 300 USD, excluding GPU. The input

Table III
THE REAL-TIME TESTING ON LOW-COST DEVICES

Devices	GHz	Resolution	RAM(GB)	FPS
Personal Computer Intel Core I5-6600	3.3	640 x 480	8	52.86
Notebook AMD A6-1450 224	1.4	640 x 480	4	10.10
Lattepanda Intel Cherry Trail	1.4	640 x 480	4	8.76
Raspberry pi 3B Broadcom BCM2837	1.2	640 x 480	2	2.15

Figure 6. Qualitative results on benchmark datasets of AFW (a), PASCAL face (b), FDDB (c), and WIDER FACE (d).

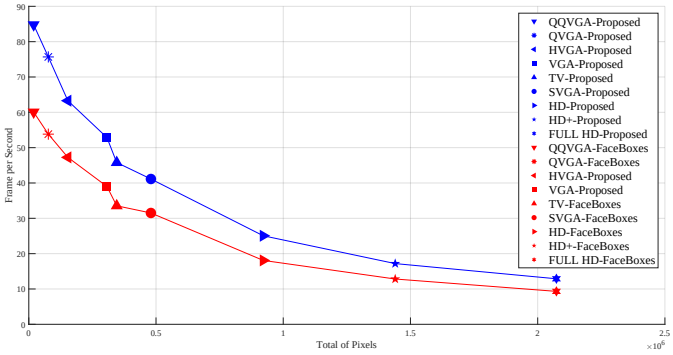


Figure 7. Comparison of detector data processing speed at different video input sizes.

of real-time testing is 1,000 frames from a webcam. The detector can operate in real-time on these devices with different speeds depending on the processor size of the computer. The slow detection speed on a raspberry CPU is due to the low execution speed. The proposed real-time system produces less computation and can work at a decent rate when tested on the CPU. The architecture is designed to lean by emphasizing the number of shallow channel layers for each convolution operation.

V. Conclusion

This paper presents a real-time face detector based on CNN using a light architecture that can work on CPUs with high performance and efficiency. The architecture consists of two main modules, a backbone, and a multi-level detector. The balancing loss and training strategies are used to improve the quality of neuron training. This architecture requires less computation cost compared to the general architecture of CNN-based object detection by producing fewer than one million parameters. Finally, the experiment shows that the proposed detector achieves state-of-the-art performance on the standard face detection benchmarks. In addition, the efficiency of the proposed model is superior to that of competitors by 53 FPS in real-time on a CPU. In future work, the usage of anchors in the object detector improves the accuracy, but it needs more computation power. An anchor-free method can be applied to increase speed and maintain performance.

References

[1] S. Jin, D. Kim, T. T. Nguyen, D. Kim, M. Kim, and J. W. Jeon, "Design and implementation of a pipelined datapath for high-speed face detection using fpga," *IEEE Transactions on Industrial Informatics*, vol. 8, no. 1, pp. 158–167, Feb 2012.

[2] M. D. Putro and K. Jo, "Real-time face tracking for human-robot interaction," in *2018 International Conference on Information and Communication Technology Robotics (ICT-ROBOT)*, Sep. 2018, pp. 1–4.

[3] J. Xiao, D. Xiong, Q. Yu, K. Huang, H. Lu, and Z. Zeng, "A real-time sliding window based visual-inertial odometry for mavs," *IEEE Transactions on Industrial Informatics*, pp. 1–1, 2019.

[4] P. Viola and M. J. Jones, "Robust real-time face detection," *International Journal of Computer Vision*, vol. 57, no. 2, pp. 137–154, May 2004. [Online]. Available: https://doi.org/10.1023/B:VISI.0000013087.49260.fb

[5] Y. Ban, S.-K. Kim, S. Kim, K.-A. Toh, and S. Lee, "Face detection based on skin color likelihood," *Pattern Recognition*, vol. 47, no. 4, pp. 1573 – 1585, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S003132031300455X

[6] X. Han, H. Liu, F. Sun, and X. Zhang, "Active object detection with multistep action prediction using deep q-network," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 6, pp. 3723–3731, June 2019.

[7] H. Gao, B. Cheng, J. Wang, K. Li, J. Zhao, and D. Li, "Object classification using cnn-based fusion of vision and lidar in autonomous vehicle environment," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 9, pp. 4224–4231, Sep. 2018.

[8] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, June 2017.

[9] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," 2018, cite arxiv:1804.02767Comment: Tech Report. [Online]. Available: http://arxiv.org/abs/1804.02767

[10] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *Computer Vision – ECCV 2016*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds. Cham: Springer International Publishing, 2016, pp. 21–37.

[11] S. Zhan, Q.-Q. Tao, and X.-H. Li, "Face detection using representation learning," *Neurocomputing*, vol. 187, pp. 19 – 26, 2016, recent Developments on Deep Big Vision. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0925231215018573

[12] Q. Huang, C. K. Jia, X. Zhang, and Y. Ye, "Learning discriminative subspace models for weakly supervised face detection," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 6, pp. 2956–2964, Dec 2017.

[13] P. Hu and D. Ramanan, "Finding tiny faces," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017, pp. 1522–1530.

[14] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao, "Joint face detection and alignment using multitask cascaded convolutional networks," *IEEE Signal Processing Letters*, vol. 23, no. 10, pp. 1499–1503, Oct 2016.

[15] J. Li, Y. Wang, C. Wang, Y. Tai, J. Qian, J. Yang, C. Wang, J. Li, and F. Huang, "Dsfd: Dual shot face detector," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[16] S. Zhang, X. Wang, Z. Lei, and S. Z. Li, "Faceboxes: A cpu real-time and accurate unconstrained face detector," *Neurocomputing*, vol. 364, pp. 297 – 309, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0925231219310719

[17] S. Zhang, X. Zhu, Z. Lei, X. Wang, H. Shi, and S. Z. Li, "Detecting face with densely connected face proposal network," *Neurocomputing*, vol. 284, pp. 119 – 127, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0925231218300274

[18] X. Zhu and D. Ramanan, "Face detection, pose estimation, and landmark localization in the wild," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, June 2012, pp. 2879–2886. [Online]. Available: https://www.ics.uci.edu/~xzhu/face/

[19] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, Jun. 2010. [Online]. Available: http://host.robots.ox.ac.uk/pascal/VOC/voc2012/

[20] V. Jain and E. Learned-Miller, "Fddb: A benchmark for face detection in unconstrained settings," University of Massachusetts, Amherst, Tech. Rep. UM-CS-2010-009, 2010. [Online]. Available: http://vis-www.cs.umass.edu/fddb/index.html

[21] S. Yang, P. Luo, C. C. Loy, and X. Tang, "Wider face: A face detection benchmark," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. [Online]. Available: http://mmlab.ie.cuhk.edu.hk/projects/WIDERFace/index.html

[22] C. Hong, J. Yu, J. Zhang, X. Jin, and K. Lee, "Multimodal face-pose estimation with multitask manifold deep learning," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 7, pp. 3952–3961, July 2019.

[23] C. Szegedy, Wei Liu, Yangqing Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015, pp. 1–9.

[24] Y. Lee, H. Kim, E. Park, X. Cui, and H. Kim, "Wide-residual-inception networks for real-time object detection," in *2017 IEEE Intelligent Vehicles Symposium (IV)*, June 2017, pp. 758–764.

[25] P. Jiang, Y. Chen, B. Liu, D. He, and C. Liang, "Real-time detection of apple leaf diseases using deep learning approach based on improved convolutional neural networks," *IEEE Access*, vol. 7, pp. 59 069–59 080, 2019.

[26] 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," 2017, cite arxiv:1704.04861. [Online]. Available: http://arxiv.org/abs/1704.04861

[27] R. Girshick, "Fast r-cnn," in *2015 IEEE International Conference on Computer Vision (ICCV)*, Dec 2015, pp. 1440–1448.

[28] S. Zhang, X. Zhu, Z. Lei, H. Shi, X. Wang, and S. Z. Li, "S3fd: Single shot scale-invariant face detector," *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 192–201, 2017.

[29] S. Yang, Y. Xiong, C. C. Loy, and X. Tang, "Face detection through scale-friendly deep convolutional networks," *ArXiv*, vol. abs/1706.02863, 2017.

[30] K. Zhang, Z. Zhang, H. Wang, Z. Li, Y. Qiao, and W. Liu, "Detecting faces using inside cascaded contextual cnn," in *2017 IEEE International Conference on Computer Vision (ICCV)*, Oct 2017, pp. 3190–3198.