Face Detection with YOLOv7: A Comparative Study of YOLO-Based Face Detection Models

Aifian Adi Sufian Chan, M.F.L Abdullah, Saizalmursidi Md Mustam,
Farhana Ahmad Poad, Ariffuddin Joret
Faculty of Electrical & Electronic Engineering
Universiti Tun Hussein Onn Malaysia
Parit Raya, Johor, Malaysia
{ he200022@siswa, faiz, saizal, farhana, ariff }@uthm.edu.my

***Abstract*—Face detection is a subcategory of object detection where the main task is to detect faces in digital media such as images. There is a plethora of face detection models that have been introduced throughout the years that use various deep learning architectures. However, this study focuses on face detection using YOLO architecture. YOLO is a single-stage architecture that is well-known for its accuracy and fast inference speed, making it suitable for various real-time applications. YOLO architecture was introduced back in 2015, and its subsequent generations performed significantly better and became more robust. This study uses the YOLOv7 model as the base model for a face detector and was trained using the Wider Face and FDDB datasets to detect faces. Three variations of face detector modules were created, each with a different number of parameters counts. The lightest model, called YFaces-Tiny, has achieved mAP scores of 94.07%, 92.36%, and 83.15% on the Easy, Medium, and Hard subsets of the Wider Face dataset. YOLOv7face-X, the largest variation, has achieved even better results in all three subsets. Based on the comparative analysis of different face detection YOLO-based models, the YFaces face detector model is capable of competing with other state-of-the-art face detection models.**

***Keywords*—face detection, YOLO, object detection, convolutional neural network**

I. Introduction

Face detection is a computer vision technology that allows devices to identify and locate human faces within digital mediums such as images or videos. This technology forms the foundation for various applications, such as face recognition systems and face expression analysis. Face detection helps to detect and extract faces in a digital medium, where it will be further processed to analyze the facial features for various purposes. Over the years, face detection technology has significantly evolved, and deep learning has played a pivotal role in its advancement.

Traditionally, face detection relies on handcrafted features to detect faces, but it is vulnerable to many different challenges. However, with the adoption of deep learning in this field, face detectors have the capability to automatically learn and recognize complex human facial features within an image. This creates a face detector that is much more robust toward various challenges than the traditional method fails to do. The high performance and robustness of these face detectors have led to the widespread use of face detection in various face-related applications. Through the years, various deep learning architectures have been introduced, each with its own unique approaches and capabilities.

One of the most famous deep learning architectures is the YOLO (You Only Look Once) model, which was designed primarily for object detection tasks. The YOLO model stands out among its competitors due to its ability to process images in real time while maintaining a commendable level of accuracy. This achievement can be attributed to its architectural design, which processes images in a single pass, making it computationally efficient. As a result, many researchers have used the YOLO architecture as their base model for various applications, particularly those that require real-time performance. In this study, YOLOv7 [1] was used as the base model for the face detector, as it not only has real-time capabilities but can also provide robust and accurate face detection. The contribution of this study can be summarized as follows:

- Design a face detector called YFaces using the YOLOv7 object detector model. YFaces has three different variations with different parameter counts.
- Evaluate the three variations of YFaces using the Wider Face [2] dataset. The results show that the variations have achieved state-of-the-art performance and have the capability to compete with other YOLO-based face detectors.

The study was organized as follows: Section II provides a comprehensive literature review of various face detection methods based on YOLO architecture. Section III provides a detailed explanation of the face detector network architecture and its training process. Section IV discusses the evaluation of the developed face detector and provides a comprehensive analysis of its competitor. Finally, Section V provides the conclusion of this study.

II. Literature Review

The aim of object detection is to locate and classify predetermined objects in an image. Face detection is an application of object detection that only focuses on one object in an image, which is the face. Before the rise of deep learning, traditional methods used handcrafted features like HAAR [3], and HOG [4] to detect faces in an image. These methods achieved phenomenal results during their time but encountered many challenges that hindered performance, especially in unconstrained environments.

Currently, the deep learning framework has rise in popularity, dominating many machine learning tasks, including face detection. The main reason is that deep learning frameworks provide better performance than their predecessors. The face detection model based on a deep learning framework is much more accurate and robust to many different challenges than its predecessors. In general, there are two types of deep learning architectures that were used in the face detection model, namely two-stage and one-stage object detectors. Examples of two-stage object detectors are the RCNN family, which includes RCNN [5], Fast-RCNN [6], Faster-RCNN [7], Mask-RCNN [8], and Cascade-RCNN [9].

These models have very good performance in detecting faces but suffer from long latency and slow speed. Meanwhile, the one-stage detector is the opposite of the two-stage detector, where it is much faster in speed but lacks in terms of performance. However, with the current advancements in deep learning frameworks, the performance gap between these two types has significantly decreased. Examples of one-stage detectors include SSD [10] and the YOLO family [11]. The face detection model is not limited to these two types of architectures, as there are models that use different architectures, such as CenterNet [12], transformer [13], and many more.

As mentioned previously, there are many types of architectures that can be used in a face detection model, but this section focuses mainly on one-stage detectors, specifically the YOLO family. YOLO was introduced in 2015 [11] and has become significantly popular as it provides faster detection than a two-stage detector without sacrificing too much in terms of performance. With the rising popularity of the model, the YOLO has rapidly improved, where its subsequence generation has improved its performance while retaining its fast inference speed. This development has enticed many researchers to modify and improve these YOLO models to better suit face detection.

There are several face detection models that use the YOLO architecture. For example, the authors in this study have designed a deep face detector using the YOLOv3 architecture [14]. The designed YOLOv3-face detector [15] can detect faces greater than 15 pixels and has improved its robustness against occlusion and pose changes. In addition, the YOLOv3-face detector's computational load is significantly reduced by using a single-pass join optimization scheme. Based on the evaluation of the Wider Face [2] dataset, its performance is closely matched to the top performers of the face detection models.

Another example of a face detection model that uses the YOLOv3 architecture [14] is the YOLO-face [16]. YOLO-face uses the YOLOv3 architecture as the base model, with three key modifications to improve its effectiveness in detecting faces of varying sizes. Firstly, it replaces the backbone of the base model from darknet-53 [14] with a new architecture called the deeper darknet. Deeper Darknet is an enhanced version of Darknet-53, whose architecture was improved by increasing the depth of the first two residual blocks. This allows the model to capture small-scale facial features that significantly improve face detection capabilities. Secondly, a novel regression loss function was proposed to address the face detection problem. The novel loss function combines MSE loss and GIoU loss. Lastly, it used k-means clustering to find the best anchor boxes for face detection.

The YOLO-FaceV2 [17] is an improvement of the YOLO-Face in that it is better at detecting multiscale faces and handling face occlusion. The YOLO-FaceV2 [17] is based on the YOLOv5 architecture and has several key elements to address the issues of multiscale face and face occlusion. To tackle the issue of multiscale faces, three design elements were deployed. The design elements are the fusion of P2 layers into the feature pyramid to enhance the resolution of small objects, the development of Scale-Aware RFE module to enhance the receptive field by using diluted convolution, and the Normalized Gaussian Wasserstein Distance (NWD) loss, which is insensitive toward small-scale objects. For face occlusion, two design elements were used, which are the SEAM module and Repulsion Loss. The SEAM modules enhance the learning of facial features by reducing the response loss of occluded faces and strengthening the response of unobstructed faces. The Repulsion Loss helps reduce the false detection rate in intraclass occlusion.

YOLOv5Faces [18] is another example of a face detector model that uses the YOLOv5 architecture. Several modifications were made to improve its face detection capabilities. These include the addition of landmark regression heads using Wing loss function. This extra supervision not only provides versatility to the model but also helps to improve its accuracy. The Focus Layer of YOLOv5 was replaced with a Stem block structure to enhance the network generalization capability while reducing its computational complexity. To make YOLOv5 better suited for face detection, the SPP block was changed, and using a smaller kernel helped improve its detection accuracy. The addition of a P6 output block with a stride of 64 also helps to increase its detection capability, especially concerning large faces, where it is often overlooked by other researchers. In addition, two super light-weight models based on ShuffleNetV2 [19] were also introduced, which have achieved state-of-the-art performance on embedded or mobile devices.

In summary, several face detectors have been developed based on the YOLO architecture. Each model has implemented a wide range of modifications and techniques to enhance its detection capabilities. These adaptations include the alterations to the YOLO architecture, incorporation of novel loss functions, and the introduction of innovative modules to address specific challenges, such as multiscale face detection and face occlusion. This shows the versality of YOLO architecture and its capabilities to produce a real-time face detector model.

III. METHODOLOGY

This section provides a high-level overview of the YOLOv7 architecture. There are four main aspects that were used in building the YOLOv7 architecture, which are the Extended-Efficient Layer Aggregation Networks (E-ELAN), model scaling for concatenation-based models, planned re-parameterized convolution and the new label assignment method.

A. *Network Architecture*

The YOLOv7 architecture uses an enhanced version of Efficient Layer Aggregation Networks (ELAN) [20], called the E-ELAN, as its backbone. Compared to its original version, E-ELAN is a superior version because it has better learning capabilities. The learning capability was improved by using expand, shuffle, and merge cardinality, which only changed the computational block of the original architecture but still retained the original architecture transition layer. E-ELAN uses group convolution to expand the channel and cardinality of the computational blocks by applying the same group parameter and channel multiplier. The feature maps produced by each computational block will be shuffled into groups according to the set group parameter. Then, the shuffled group feature maps will be concatenated together, and merge cardinality will be performed. This strategy helps to maintain the original ELAN design architecture and its gradient path, but it also improves its learning ability as it can learn more diverse features. Figure 1 shows the architecture of E-ELAN used in YOLOv7.

Model scaling was used to adjust the model's attributes to create models of different scales to meet the criteria of the different inference speeds. YOLOv7 introduces a compound model scaling method for a concatenation-based model where it considers different scaling factors together. Figure 2 shows the proposed model scaling for concatenation-based models. When the depth factor of a computational block is scaled, the changes in the output channel are calculated. Then, the width factor scaling is performed using the same number of changes in the transition layer. This will help to maintain the initial properties of the model and its optimal structure. The next implementation is the planned re-parameterized convolution, where it uses RepConv without an identity connection (RepConvN). The main reason is because it was discovered that the identity connection destroyed the residual in ResNet and the concatenation in DenseNet. This will reduce the diversity of gradients for different feature maps. Hence, RepConvN is used instead.

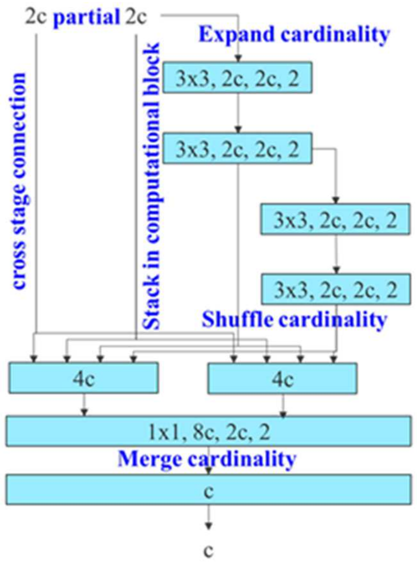

Fig. 1. Demonstration of E-ELAN

A YOLO architecture contains a backbone, a neck, and a head. The head contains the predicted model outputs; here, the model used uses two different heads that are inspired by deep supervision. The first head responsible for final output is referred to as the lead head, while the head used to assist the training process is called the auxiliary head. In addition, YOLOv7 also introduces a mechanism called the "label assigner" that considers both network prediction and ground truth together before assigning the soft labels. This led to a new derivative issue on how to assign soft labels to the auxiliary head and lead head. To solve this issue, YOLOv7 proposed a new label assignment method that uses the lead head prediction to guide both the auxiliary head and the lead head. The lead head prediction is used as guidance to generate coarse-to-fine hierarchical labels that will be used in the auxiliary head and lead head. Figure 3(a) and (b) demonstrate the two proposed deep supervision label assignment strategies. The lead-guided label assigner is calculated based on the prediction of the lead head and the ground truth where it will generate soft labels through the optimization process. The output soft labels will be used as the target training model for both heads.

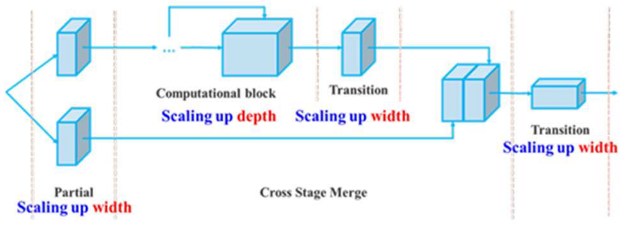

Fig. 2. Visualization of the proposed model scaling for concatenation-based models.

Meanwhile, the coarse-to-fine lead-head guided label assigner has the same process as the lead-guided label assigner, but it generates two sets of soft labels, which are the coarse label and the fine label. The generated fine label is similar to the soft label generated by the lead-head guided label assigner. Meanwhile, the coarse label was generated by allowing more grids to be treated as positive targets by relaxing the constraints of the positive assignment process. However, the additional weight of the coarse label might be too close to that of the fine label, which might produce bad instances at the final prediction. Hence, to reduce the impact of the extra coarse positive grid, restrictions are placed in the decoder so that it cannot produce soft labels perfectly. The final output of the lead head is obtained by filtering out the high precision results from the high recall results. This mechanism allows the importance of fine and coarse labels to be dynamically adjusted during the learning process and makes the optimizable upper bound of the fine label always higher than the coarse label.

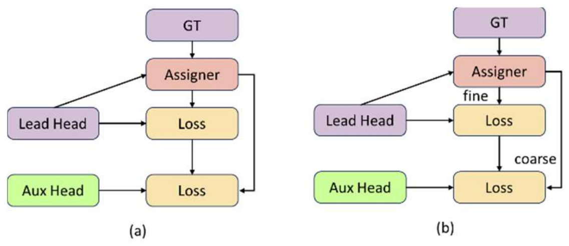

Fig. 3. Visualisation of (a) lead head guided label assigner and(b) coarse-to-fine lead head guided label assigner.

B. Training Phase

1) Experimental Setup

A hardware package equipped with an RTX3060 12GB Nvidia graphic card and 32GB of RAM is used to train and test the model. This hardware provides sufficient computational power that can be used to train and test the model's performance. PyTorch is the main deep learning library that is used to implement the YOLOv7 model.

2) Dataset

Two face datasets were used to train the YFaces detector. The first dataset is the Wider Face dataset [2]. It contains 32,203 images and 393,703 faces that have huge variation that includes scale, pose, occlusion, expression, and illumination. The Wider Face dataset was divided into three different sets, which are the training, validation, and testing sets. The ratio of images in the dataset was set to 50%, 10%, and 40% for the training, validation, and testing sets, respectively. The validation set was defined into three levels of difficulty: Easy, Medium, and Hard. The hard subset is the most challenging and provides the best indicator to measure a face detector model's effectiveness in detecting faces in a challenging environment.

The second face dataset that was used is the Face Detection Data Set and Benchmark (FDDB) dataset [21], which contained 5171 face annotations for 2845 images. The dataset was designed to study the problem of detecting faces in unconstrained environments. The dataset was fully used as a training set to provide the necessary data to train the YFaces face detector model.

3) Hyperparameter tuning

One of the most important aspects of the training phase is hyperparameter tuning. Hyperparameter fine-tuning helps to improve the performance of the model, allows for faster convergence, and lowers the risk of overfitting. Table 1 shows several hyperparameters that were fine-tuned.

TABLE I. LIST OF TUNED HYPERPARAMETERS

Hyperparameters	***Fine-tuned parameter***
Learning Rate	0.01
Weight Decay	0.0005
Optimizer	Stochastic gradient descent

IV. EXPERIMENTATION

This section provides details on the results obtained from testing the three different models' variations using the Wider Face evaluation dataset [2]. The performances of these models were compared with other state-of-the-art YOLO-based face detector models. This will give a clear indication of how well these models perform compared to their predecessors.

A. *Experimental setup*

Three different YOLOv7 models were implemented and trained, each with a different parameter size. The smallest YOLOv7 model is the YFaces-Tiny, which has a total of 6.2 million parameters. The YFaces-M is the medium variation of the YOLOv7 model with 36.9 million parameters, while the largest model is the YFaces-X with 71.3 million parameters.

For the comparative analysis, four different YOLO-based detection models were used. Representing YOLOv3 [14], YOLOFacev1[16] and TinyYOLOv3 [22] were used. YOLOFacev2 [17] and YOLOv5faces [18] were used to represent YOLOv5 models. The models were tested using the three levels of difficulty of the Wider Face validation dataset [2].

B. *Results and Analysis*

Three variations of the YFaces model were meticulously evaluated using the Wider Face dataset [2], which is categorized into three subsets based on varying difficulty levels. The outcomes of this assessment are summarized in Table II. In the Easy subset, all three variations of the YFaces achieved a mAP of more than 94%. For the Medium subset, YFaces-Tiny achieved mAP of 92.36%, followed by YFaces-M with 94.14% and YFaces-X has mAP of 95.12%. Meanwhile in the Hard subset, 88.14, the highest mAP was achieved by YFaces-X followed by YFaces-M and YFaces-Tiny with 86.20% and 83.06% respectively. Based on these results, it can clearly see that all the variation of YFaces model achieved outstanding results in all three subsets.

Moving on to the comparative analysis, all three variations of YFaces have managed to surpass YOLOfacesV1 [16] in all three subsets of the Wider Face dataset. The TinyYolov3 [22] face detector is another face detector based on the Yolov3 architecture. Almost all the variations of YFaces have surpassed the TinyYolov3 face detector in all three subsets except for YFaces-Tiny in the easy subset. Moving on to the next comparison with YOLOfacesV2 [17], Table II shows that the results of the small and medium variations of YFaces are no match. However, the YFaces-X variation closely matches YOLOfacesV2 in the Easy and Medium subsets with a difference of 2.59% and 2.08%, respectively, but marginally surpasses YOLOfacesV2 in the Hard subset by 0.44%.

The next comparison is with Yolov5Faces [18], where only three variations are used out of the nine variations that the authors have proposed in their work. To ensure fairness, the three variations selected from the YOLOv5Faces share almost identical number parameters with the YFaces variations. In the small size category, Yolov5s slightly outperform the YFaces-Tiny in the three Wider Face subsets. However, in the medium size category, YFaces-M marginally outperformed YOLOv5m in the medium and hard subsets but slightly underperformed in the easy subset, with a difference of 0.06%. Meanwhile, in the large category, YFaces-X managed to outperform YOLOv5l6 in the Medium and Hard subset but slightly underperform in the Easy subset. Figure 4 shows a sample image of detected faces using the YFaces-X model.

TABLE II. COMPARISION PERFORMANCE OF DIFFERENT YOLO-BASED DETECTION MODELS USING WIDER FACE DATASET

Model	***Wider Face Dataset (mAP)***			***Parameters***
	Easy	***Medium***	***Hard***	
YOLOfacesV1 [16]	89.9	87.2	69.3	-
TinyYolov3 [22]	95.26	89.2	77.9	-
YOLOv5s [18]	94.33	92.61	83.15	7.1M
YFaces-Tiny	94.07	92.36	83.06	6.2M
YOLOv5m [18]	95.66	94.10	85.20	35.5M
YFaces-M	95.60	94.14	86.20	36.9M
YOLOv5l6 [18]	96.38	94.90	85.88	76.7M
YOLOfacesV2 [17]	**98.70**	**97.20**	87.70	-
YFaces-X	96.11	95.12	**88.14**	71.3M

w

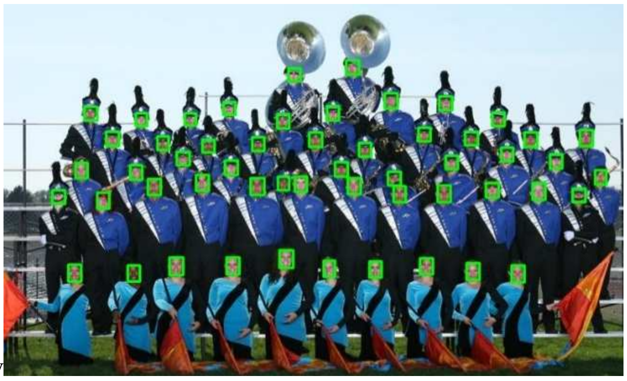

Fig. 4. Sample image of detected faces using YFaces-X.

In summary, these comprehensive findings lead to the unequivocal conclusion that the YOLOv7 face detection model has firmly established itself as the state-of-the-art (SOTA) performer in face detection tasks. It consistently provides outstanding performance across the Easy, Medium, and Hard subsets. In addition, its impressive superiority over TinyYOLOv3, YOLOfacesV1, and YOLOv5Faces shows that YFaces can compete with other state-of-the-art face detectors.

V. Conclusion

In this study, the implementation of YOLOv7 was used as the base for the YFaces detector model. A total of three different variations were presented in this study. YFaces-Tiny is the smallest model with 6.2 million parameters, while YFaces-X is the largest model with 71.3 million parameters. The results indicate that these models can compete with other state-of-the-art face detector models that are based on YOLO architecture. It either closely matches or surpasses the model in the three difficulty subsets of the Wider Face dataset. This is a clear indication that the YOLOv7 can achieve state-of-the-art performance in detecting faces that matches other state-of-the-art face detector models.

Acknowledgment

The authors would like to thank University Tun Hussein Onn Malaysia for providing grant RE-GG (H884) funding and Advanced Telecommunication Centre (ATRC) for supporting this research.

References

[1] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "YOLOv7: Trainable Bag-of-Freebies Sets New State-of-the-Art for Real-Time Object Detectors," pp. 7464–7475, 2023.

[2] S. Yang, P. Luo, C. C. Loy, and X. Tang, "WIDER FACE: A face detection benchmark," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2016-December, pp. 5525–5533, Dec. 2016.

[3] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1, 2001.

[4] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," *Proceedings - 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2005*, vol. I, pp. 886–893, 2005.

[5] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 580–587, 2014.

[6] R. Girshick, "Fast R-CNN," *Proceedings of the IEEE International Conference on Computer Vision*, vol. 2015 Inter, pp. 1440–1448, 2015, doi: 10.1109/ICCV.2015.169.

[7] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *IEEE Trans Pattern Anal Mach Intell*, vol. 39, no. 6, pp. 1137–1149, Jun. 2017.

[8] K. He, G. Gkioxari, P. Dollar, and R. Girshick, "Mask R-CNN," *Proceedings of the IEEE International Conference on Computer Vision*, vol. 2017-October, pp. 2980–2988, Dec. 2017.

[9] Z. Cai and N. Vasconcelos, "Cascade R-CNN: Delving into High Quality Object Detection," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 6154–6162, Dec. 2018.

[10] W. Liu *et al.*, "SSD: Single shot multibox detector," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9905 LNCS, pp. 21–37, 2016.

[11] J. S. D. R. G. A. F. Redmon, "(YOLO) You Only Look Once," *Cvpr*, vol. 2016-December, pp. 779–788, Dec. 2016.

[12] K. Duan, S. Bai, L. Xie, H. Qi, Q. Huang, and Q. Tian, "CenterNet: Keypoint triplets for object detection," *Proceedings of the IEEE International Conference on Computer Vision*, vol. 2019-October, pp. 6568–6577, Oct. 2019.

[13] H. Zhang and L. Chi, "End-to-End Spatial Transform Face Detection and Recognition," *Virtual Reality & Intelligent Hardware*, vol. 2, no. 2, pp. 119–131, Apr. 2020.

[14] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," 2018.

[15] F. Gurkan, B. Sagman, and B. Gunsel, "YOLOv3 as a Deep Face Detector," *ELECO 2019 - 11th International Conference on Electrical and Electronics Engineering*, pp. 605–609, Nov. 2019.

[16] W. Chen, H. Huang, S. Peng, C. Zhou, and C. Zhang, "YOLO-face: a real-time face detector," *Visual Computer*, vol. 37, no. 4, pp. 805–813, 2021.

[17] Z. Yu, H. Huang, W. Chen, Y. Su, Y. Liu, and X. Wang, "YOLO-FaceV2: A Scale and Occlusion Aware Face Detector," pp. 1–18, 2022.

[18] D. Qi, W. Tan, Q. Yao, and J. Liu, "YOLO5Face: Why Reinventing a Face Detector," pp. 228–244, May 2021.

[19] N. Ma, X. Zhang, H. T. Zheng, and J. Sun, "Shufflenet V2: Practical guidelines for efficient cnn architecture design," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11218 LNCS, pp. 122–138, 2018.

[20] C. Y. Wang, H. Y. M. Liao, and I. H. Yeh, "Designing Network Design Strategies Through Gradient Path Analysis," *Journal of Information Science and Engineering*, vol. 39, no. 2, pp. 975–995, 2023.

[21] V. Jain and E. Learned-Miller, "FDDB: A Benchmark for Face Detection in Unconstrained Settings".

[22] J. Gao and T. Yang, "Face detection algorithm based on improved TinyYOLOv3 and attention mechanism," *Comput Commun*, vol. 181, pp. 329–337, Jan. 2022.