AIML 425 PROJECT

Quan Zhao (Student ID: 300471028)

Victoria University of Wellington

1. INTRODUCTION

Face Detection is a very popular research topic and have various applications in industry. Nerual Network plays key role in this work, lots of Face Detection Nerual Networks has been introduced. RetinaFace is one of most populars implementation, which is been introduced in 2019, and be accepted by "Conference on Computer Vision and Pattern Recognition (CVPR)" in 2020. In this study, I will show my understanding of this work based on this paper "RetinaFace: Single-shot Multi-level Face Localisation in the Wild" [1] In real world, lots of edge case can not be covered by this pretrained model. Retrain whole model cost too much. Fine tuning model approach will be introduced in this study.

2. CONTENT ANALYSIS

In paper "RetinaFace: Single-shot Multi-level Face Localisation in the Wild" [1], Author introduced an approach which is a novel pixel-wise face localization method based on a single-stage design call RetinaFace. The method employs a multi-task learning strategy to simultaneously predict various attributes related to a face.

2.1. RetinaFace

RetinaFace, as the name suggests, is a specialized adaptation of the RetinaNet architecture for face detection. While it retains the core principles of RetinaNet, it introduces additional components tailored for face detection, such as:

1. Context Module
2. Cascade Multi-task Loss
3. Predicting facial landmarks in addition to face bounding boxes.
4. Incorporating a mesh decoder for predicting pixel-wise 3D face shapes.

In essence, while RetinaNet was designed for general object detection, RetinaFace adapts and extends the architecture to cater specifically to the challenges and requirements of face detection.

2.2. RetinaNet

RetinaNet [2] is a single-stage object detector that introduced the Focal Loss to address the class imbalance problem inherent in object detection tasks. The name "RetinaNet" is derived from its use of a feature pyramid network (FPN) to detect objects at different scales, similar to the multi-scale processing in the human retina.

The architecture and features of RetinaNet are:

1. Single-stage vs. Two-stage Detectors:
    - Before RetinaNet, there were primarily two categories of object detectors: single-stage detectors (like YOLO and SSD) that are fast but less accurate, and two-stage detectors (like Faster R-CNN) that are more accurate but slower.
    - RetinaNet was designed as a single-stage detector but aimed to achieve the accuracy levels of two-stage detectors.
2. Feature Pyramid Network (FPN):
    - RetinaNet incorporates the Feature Pyramid Network (FPN) to efficiently detect objects at different scales.
    - FPN constructs a multi-scale pyramid of feature maps by combining low-resolution, semantically strong features with high-resolution, semantically weak features using a top-down pathway and lateral connections.
    - This multi-scale representation is crucial for detecting objects of varying sizes in images.
3. Focal Loss:
    - One of the main challenges with single-stage detectors is the class imbalance between foreground (object) and background classes. Since these detectors densely sample possible object locations in an image, there are many more negative samples (background) than positive samples (objects).
    - RetinaNet introduced the Focal Loss to address this imbalance. The Focal Loss is designed to down-weight the contribution of easy negatives

(background samples that are easily classified as background) and focus more on the hard negatives and positives.

- This dynamic scaling of the loss helps in training the model more effectively on challenging samples.

4. Anchor Boxes:

   - Like many object detectors, RetinaNet uses anchor boxes (pre-defined bounding boxes) at each spatial location in its feature maps to predict object locations.
   - Multiple anchor boxes with different scales and aspect ratios are used at each location to cater to objects of different shapes and sizes.

5. Classification and Regression Heads:

   - For each anchor box, RetinaNet has a classification head that determines the object class (or background) and a regression head that refines the bounding box coordinates.
   - These heads are shared across all the scales in the FPN, ensuring consistent predictions across different object sizes.

6. Backbone Network:

   - RetinaNet typically uses a deep convolutional network, like ResNet, as its backbone to extract feature maps from the input image. These feature maps then feed into the FPN.

2.2.1. deeper into Focal Loss

The Focal Loss is a specialized loss function introduced to address the class imbalance problem in object detection, particularly for single-stage detectors.

In object detection, especially with single-stage detectors, there's a significant class imbalance between the background class (no object) and the object classes. This is because detectors densely sample possible object locations in an image, leading to many more negative samples (background) than positive samples (objects).

Traditional cross-entropy loss treats every sample equally, which means the vast number of easy negative samples can dominate the loss and lead to degenerate models.

The Focal Loss is designed to down-weight the contribution of easy samples and focus more on the hard samples. It is defined as:

$$\text{FL}(p_t) = -\alpha_t(1 - p_t)^\gamma \log(p_t)$$

where:

- p_t is the probability of the true class.
- α_t is a balancing factor.
- γ is the focusing parameter that adjusts the rate at which easy samples are down-weighted.
- When $\gamma = 0$, Focal Loss is the same as cross-entropy loss. As γ increases, the effect of the modulating factor becomes more pronounced.

1. Effects of the Focusing Parameter γ:

   - The modulating factor $(1 - p_t)^\gamma$ ensures that easy examples (those with a high value of p_t for the true class) have a reduced loss.
   - As γ increases, the effect of this modulation intensifies, meaning that the loss for easy examples decreases rapidly.
   - This allows the model to focus more on hard, misclassified examples.

2. Balancing Factor α_t:

   - α_t can be set to balance the importance of positive/negative classes. For instance, it can be set inversely proportional to the class frequencies.
   - It's an additional factor to handle extreme class imbalance.

3. Benefits:

   - By using the Focal Loss, models are less prone to being overwhelmed by abundant easy negative samples.
   - It allows single-stage detectors, like RetinaNet, to achieve state-of-the-art performance by addressing one of the main challenges they face.

The Focal Loss is a pivotal innovation in the realm of object detection. By dynamically scaling the loss based on the true class probability, it ensures that the training process is not dominated by easy-to-classify samples. This focus on challenging samples leads to improved model performance, especially in scenarios with significant class imbalances.

2.3. feature pyramid network (FPN)

In traditional computer vision, pyramids were used to process images at multiple scales to detect objects of varying sizes. Deep convolutional neural networks (CNNs) inherently produce feature maps at different resolutions, with shallow layers capturing high-resolution but semantically weak features and deep layers capturing low-resolution but semantically strong features. FPN [3] aims to leverage these multi-scale features effectively for object detection.

It consists of the following parts.

1. Bottom-up Pathway:

   This is the standard forward pass in a CNN. As you go deeper into the network, the spatial resolution of the feature maps decreases, but the semantic information increases.

2. Top-down Pathway and Lateral Connections:

   - FPN introduces a top-down pathway where the spatial resolution is upsampled.

   - During this upsampling, FPN also adds lateral connections from the bottom-up pathway. These lateral connections are feature maps from the same level in the bottom-up pathway, which are merged with the upsampled features.

   - This merging process combines high-level semantic information from the top-down pathway with detailed spatial information from the bottom-up pathway.

3. Multi-scale Predictions:

   With the combined features at different levels of the pyramid, FPN can make predictions at multiple scales. This allows for the detection of objects of varying sizes with high accuracy.

4. Benefits:

   - Improved Detection Across Scales: Traditional CNNs often struggled with detecting small objects because the feature maps in deeper layers, which have strong semantic information, are of low resolution. FPN addresses this by combining low-resolution, semantically strong features with high-resolution, semantically weak features.

   - Efficiency: Instead of processing the image at multiple scales separately, FPN efficiently constructs a feature pyramid from a single-scale image, reducing computational overhead.

5. Usage in Modern Architectures:

   FPN has become a foundational component in many state-of-the-art object detection architectures, including RetinaNet. By providing multi-scale feature representations, it enhances the detector's ability to recognize objects of various sizes.

Feature Pyramid Network (FPN) is a pivotal advancement in object detection, addressing the challenge of detecting objects across different scales. By integrating multi-scale feature representations in a unified architecture, FPN enhances both the accuracy and efficiency of object detectors.

2.4. Anchor based Face Deteciton on pyramid network

In RetinaFace, the anchors are indeed densely tiled across the feature maps of each pyramid level, such as P2. However, the anchors don't predict for every single pixel in the sense of moving pixel by pixel. Instead, they are associated with the spatial locations on the feature map, and their density is determined by the stride of that particular pyramid level.

Let's break this down:

1. Stride: Each pyramid level has an associated stride, which indicates the spatial resolution difference between the original image and the feature map at that level. For instance, if P2 has a stride of 4, it means each spatial location in the P2 feature map corresponds to a 4x4 region in the original image.

2. Anchor Tiling: For each spatial location on a feature map (e.g., P2), multiple anchors of different scales and aspect ratios are tiled. So, if you're at a specific x,y location on P2, there will be multiple anchors centered at that location, each with a different size or shape.

3. Dense Sampling: While the anchors are densely sampled, they aren't associated with every pixel of the original image. Instead, they are associated with the spatial locations of the feature map. Using the P2 example with a stride of 4: for the first spatial location (top-left), the anchors might be centered at pixel (2,2) of the original image (assuming the anchor's center is at the middle of the 4x4 region). The next set of anchors on P2 would be centered at pixel (6,2), and so on.

4. Coverage: Even though the anchors aren't predicting for every single pixel, the combination of multiple scales and aspect ratios ensures that they can potentially cover all objects (faces, in the case of RetinaFace) in the image. The dense tiling, combined with the varying sizes and shapes of the anchors, allows the model to detect faces of different sizes and orientations.

In summary, while RetinaFace does densely tile anchors across the feature maps of each pyramid level, it doesn't predict for every pixel in a pixel-by-pixel manner. Instead, the predictions are made for spatial locations on the feature map, with the density determined by the stride of the pyramid level.

2.5. Context Module in RetinaFace

The "context module" is a significant component of the RetinaFace architecture.

The context module in RetinaFace is designed to capture contextual information from different feature levels. In many face detection architectures, multi-level feature fusion is a common strategy to handle faces of various sizes and in different scenarios. The context module in RetinaFace aims

to enhance the feature representation by aggregating context from different scales.

1. Multi-level Feature Aggregation: The context module takes features from different levels of the network. These features come from various resolutions, capturing both high-level and low-level details.

2. Feature Enhancement: By aggregating these multi-level features, the context module can enhance the representation of each feature level. This is particularly useful for detecting small faces or faces in challenging scenarios where contextual information can provide additional clues.

3. Improved Detection Performance: The inclusion of the context module in RetinaFace helps in achieving better detection performance, especially in scenarios where faces are partially occluded, are of varying sizes, or are in challenging lighting conditions.

In essence, the context module serves as a mechanism to integrate and leverage information from different scales, enhancing the robustness and accuracy of the RetinaFace detector in diverse real-world scenarios.

2.5.1. How Context Module works with FPN

1. Feature Aggregation: The context module takes the multi-scale features provided by the FPN and further refines them by aggregating context. This can be thought of as a two-step enhancement: first, the FPN provides features at multiple scales, and then the context module refines these features by considering broader contextual information.

2. Improved Detection: By combining the strengths of FPN and the context module, RetinaFace achieves a richer feature representation. This enhanced representation is more robust and versatile, allowing the model to detect faces with higher accuracy, especially in challenging scenarios.

In summary, while the FPN in RetinaFace provides a multi-scale feature representation, the context module further refines these features by aggregating contextual information. The combination of these two components results in a powerful and robust face detection mechanism.

2.6. Cascade Regression

Cascade regression refers to a series of regression operations performed in a cascaded manner. In the context of object detection (or face detection, in the case of RetinaFace), this often means refining the bounding box predictions in a step-by-step manner. Instead of predicting the final bounding box coordinates in one go, the model makes an initial prediction and then refines it in subsequent steps.

The idea behind cascade regression is that iterative refinement can lead to more accurate and stable bounding box predictions, especially in challenging scenarios.

2.7. Multi-task Loss

The multi-task loss in the RetinaFace paper is designed to handle multiple objectives simultaneously, including face classification, bounding box regression, facial landmark regression, and dense regression.

$$L = L_{cls}(p_i, p^*i) + \lambda_1 1p^i L * box(t_i, t^*i)$$
$$+\lambda_2 1p^i L * pts(l_i, l^*i) + \lambda_3 1p^i L * pixel$$

Where:

- $1_{p_i^*}$ is an indicator function that is 1 for positive anchors and 0 otherwise.
- $\lambda_1, \lambda_2, \lambda_3$ are loss-balancing parameters.

In the RetinaFace paper, the loss-balancing parameters $\lambda_1, \lambda_2, \lambda_3$ are set to 0.25, 0.1, and 0.01, respectively. This means that the model places a higher significance on better box and landmark locations derived from supervision signals.

Now, breakdown this formla.

1. Face Classification Loss (L_{cls}):
   - This loss is responsible for determining whether a given anchor is a face or not.
   - p_i is the predicted probability of anchor i being a face.
   - p_i^* is the ground truth, where it's 1 for a positive anchor (face) and 0 for a negative anchor (non-face).
   - The classification loss L_{cls} uses the softmax loss for binary classes (face/not face).
2. Face Box Regression Loss (L_{box}):
   - This loss refines the bounding box coordinates for detected faces.
   - $t_i = \{t_x, t_y, t_w, t_h\}$ represents the coordinates of the predicted box.
   - $t_i^* = \{t^x, t^y, t_w^*, t_h^*\}$ represents the ground truth box coordinates associated with a positive anchor.
   - The regression targets (i.e., center location, width, and height) are normalized, and the loss L_{box} uses the smooth-L1 loss as defined in the Fast R-CNN paper.
3. Facial Landmark Regression Loss (L_{pts}):

- This loss predicts the locations of five facial landmarks.
- $l_i = \{l_{x1}, l_{y1}, ..., l_{x5}, l_{y5}\}$ represents the predicted five facial landmarks.
- $l_i^* = \{l^{x1}, l^{y1}, ..., l_{x5}^*, l_{y5}^*\}$ represents the ground truth landmarks associated with a positive anchor.
- Similar to the box center regression, the facial landmark regression also employs target normalization based on the anchor center.

4. Dense Regression Loss (L_{pixel}):
   - This loss is related to the dense regression branch of the model, which is designed to provide dense facial landmarks.
   - The specifics of this loss are further detailed in the paper, especially in relation to the mesh decoder.

2.8. Cascade Regression with Multi-task Loss

1. Iterative Bounding Box Refinement: Within the RetinaFace framework, cascade regression is applied to the bounding box regression task. The model first predicts an initial bounding box. This prediction is then refined in subsequent stages, each time getting closer to the ground truth.
2. Joint Optimization: While the bounding box predictions are being refined through cascade regression, the model still jointly optimizes the other tasks (face classification and landmark localization) using the multi-task loss. This ensures that all tasks benefit from the iterative refinement process.
3. Improved Accuracy: The combination of cascade regression and multi-task loss means that the model can achieve more accurate bounding box predictions without compromising the performance of the other tasks. The iterative refinement process allows the model to correct any initial inaccuracies in the bounding box predictions, leading to better overall detection performance.

In essence, by combining cascade regression with a multi-task loss, RetinaFace ensures that the bounding box predictions are iteratively refined, leading to more accurate and stable face detections. This iterative refinement process works in tandem with the other tasks, ensuring that the overall performance of the model is optimized.

3. IMPACT

The "RetinaFace: Single-shot Multi-level Face Localisation in the Wild" paper has significantly influenced the domain of face detection. Its contributions have paved the way for a plethora of subsequent research endeavors. This section delves into the multifaceted impact of RetinaFace on the research community.

3.1. Improving the Architecture

The foundational architecture of RetinaFace has not only been influential in human face detection but has also been adapted for other species. A remarkable adaptation is presented by [4], titled "CattleFaceNet: A cattle face identification approach based on RetinaFace and ArcFace loss." This research innovatively leverages the strengths of RetinaFace for cattle face identification, integrating it with the ArcFace loss to enhance recognition accuracy. Such adaptations underscore the versatility of the RetinaFace model and its potential to serve as a base for diverse identification tasks.

3.2. Adapting to Different Domains

The adaptability of RetinaFace is showcased by its application in various unique scenarios beyond traditional face detection. A significant example is the study by [5], titled "Single Camera Masked Face Identification." This research emphasizes the challenges and nuances of identifying faces when obscured by masks, a scenario that has become increasingly relevant in recent times. It demonstrates the flexibility of the RetinaFace model to cater to domain-specific challenges and its potential to address contemporary issues.

3.3. Optimization for Real-time Applications

The need for real-time face detection, especially in resource-constrained environments, has driven researchers to optimize RetinaFace for speed and efficiency. A notable contribution in this domain is the work by [6], titled "A Faster Real-time Face Detector Support Smart Digital Advertising on Low-cost Computing Device." This research underscores the importance of efficient face detection in the realm of digital advertising, particularly on low-cost computing devices, further emphasizing the adaptability and potential of the RetinaFace model for real-world applications.

4. FINE TUNING PRETRAINED MODEL

Fine-tuning a pretrained face detection model involves adapting a model that has been trained on a large dataset to a new, typically smaller, dataset. This can be useful when we have a specific dataset or use-case in mind that might differ from the original training data.

There are three potential approach can be used to fine-tuning the pretrained model, which are fine-tuning hypter-parameters, fine-tuning by freeze early layers and fine-tuning with different anchors.

4.1. Fine-tuning hypter-parameters

RetinaFace is a prominent facial detection model pretrained on a diverse dataset. Even though it exhibits commendable performance right out of the box, for specific applications or datasets, there might be a need to adjust its hyper-parameters to achieve optimal performance. This section discusses the key hyper-parameters and emphasizes those that need particular attention during the fine-tuning phase.

Among the several hyper-parameters that

RetinaFace uses, the following are crucial for performance optimization:

1. **NMS Threshold (nms_threshold)**:

   This parameter represents the Intersection over Union (IoU) threshold. When the IoU of two bounding boxes exceeds this threshold, the bounding box with the lower confidence score is suppressed during the Non-Maximum Suppression (NMS) process. The default value for this threshold is set to 0.4, but it lies within the range [0.0, 1.0]. For some applications, adjusting this threshold can help reduce false positives or improve the recall.

2. **Visualization Threshold (vis_thres)**:

   This is the confidence threshold applied after the NMS process and before the final output. It filters detections based on their confidence scores, allowing only those with scores above this threshold to be part of the final output. Its default value is 0.6, and it can be adjusted between [0.0, 1.0]. Fine-tuning this parameter can help in reducing noise in the final detections.

The other parameters such as *confidence_threshold*, *top_k*, and *keep_top_k* are set to their default values of 0.02, 5000, and 750 respectively.

Their settings generally ensure that the model is sufficiently sensitive (but not overly so) and that a large enough number of top detections are kept both before and after the NMS process.

Fine-tuning these hyper-parameters,

especially the NMS and visualization thresholds, plays a pivotal role in the model's performance.

This is because the balance between precision and recall in object detection tasks is sensitive to these values. A lower NMS threshold might result in more bounding boxes being suppressed, leading to fewer false positives but potentially missing some true positives.

On the other hand, adjusting the visualization threshold can control the quality of detections in the final output. Therefore, by fine-tuning these parameters to suit a specific dataset or application, one can achieve a more desirable balance between precision and recall, ensuring that the model's detections are both accurate and comprehensive.

4.2. Fine-tuning with Frozen Early Layers

Fine-tuning pretrained models on novel datasets is a prevalent strategy in deep learning, offering a balance between leveraging pre-existing knowledge and adapting to new data. A common approach during this process is to freeze the early layers of the model, allowing only the deeper layers to be updated. This section delves into the rationale behind this approach and its implications for model performance.

4.2.1. Rationale for Freezing Early Layers

1. Transfer of Generic Features: Deep neural networks, particularly convolutional architectures, have a hierarchical feature extraction process. The initial layers often capture universal, low-level features such as edges, textures, and colors. These foundational features are generally applicable across diverse datasets, making them valuable for transfer learning.

2. Mitigation of Overfitting: Fine-tuning all layers with a small novel dataset can lead to overfitting, where the model becomes overly specialized to the training data. By freezing the early layers, the number of trainable parameters is reduced, thus decreasing the model's capacity to overfit.

3. Computational Efficiency: Training fewer parameters can expedite the convergence of the model, leading to faster training epochs and potentially requiring fewer epochs to achieve optimal performance on the new dataset.

4.2.2. Why Freezing Early Layers Works?

1. Hierarchical Feature Learning:

   As data progresses through the layers of a neural network, features transition from being generic to specific.

   Early layers often act as filters detecting fundamental patterns, while deeper layers combine these patterns to recognize more complex structures. When transferring to a new task, the basic patterns remain largely consistent, while the higher-level representations require adjustments to cater to the specifics of the new data.

2. Regularization Effect:

   By not allowing early layers to change, the model is implicitly regularized. This constraint can be viewed as a form of structural regularization, where certain parts of the model are kept constant, guiding the learning process in the deeper layers and preventing them from fitting to noise in the new dataset.

3. Data-driven Adaptation:

While the early layers remain fixed, the deeper layers, which are more adaptable and data-specific, undergo training. This ensures that the model retains its generalization capability from the pretraining phase while fine-tuning based on the nuances of the new dataset.

Freezing the early layers during fine-tuning strikes a balance between leveraging pre-existing, generic features and adapting to novel data characteristics. This approach, rooted in the hierarchical nature of feature extraction in deep networks, offers a computationally efficient and effective method for transfer learning across diverse tasks. Future research might explore adaptive freezing strategies, where the decision to freeze or train layers is data-driven, further optimizing the fine-tuning process.

4.3. Fine-tuning with different anchors

Face detection models often rely on predefined bounding boxes, known as anchors, to predict the location of faces in an image. The accuracy of these models can be significantly influenced by the choice of these anchors, especially when dealing with diverse face sizes and orientations. In this section, we delve into the methodology and impact of fine-tuning our model with different anchor configurations.

4.3.1. Anchor Analysis and Selection

Before embarking on the fine-tuning process, it's paramount to understand the distribution of face sizes and aspect ratios in the dataset. This preliminary analysis provides insights into potential anchor sizes and ratios that might be most effective.

To determine the optimal sizes for our anchors, we employed k-means clustering on the ground truth bounding boxes. The centroids of these clusters provided a representative set of widths and heights for our anchors. While the number of anchors is often set between 5 and 9 in many applications, our dataset's unique characteristics led us to choose *TODO:X* anchors (where *TODO:X* is the number you decided on).

4.3.2. Anchor Ratios and Scaling

Beyond mere size, the aspect ratio of anchors plays a pivotal role in accurately capturing the variability of faces in images. Common ratios, such as 1:1, 1:2, and 2:1, were considered based on the dataset's distribution.

Due to FPN has been used in RetinaFace models that utilize multiple feature maps to detect objects at varying scales. following the implementation in RetinaFace we use same number anchors for feature maps.

4.3.3. Training and Results

With the newly determined anchors, the model was trained and its performance closely monitored.

Preliminary results indicated a *TODO:Y%* improvement in accuracy (where *TODO:Y* is the improvement percentage you observed). However, it's worth noting that while anchor fine-tuning can enhance performance, it's just one facet of the optimization process. Factors like data augmentation, model architecture, and loss functions also significantly influence the model's efficacy.

4.3.4. Conclusion

Fine-tuning with different anchors offers a promising avenue for enhancing face detection models. By tailoring the anchors to the specific characteristics of the dataset, we can achieve more accurate and robust face detections. Future work might explore the iterative refinement of anchors and delve deeper into the interplay between anchor configurations and other model hyperparameters.

5. RESULTS

5.1. Data Preparation

5.2. Model Fine Tuning and evaluation

6. IMPACT

7. CONCLUSION

8. STATEMENT OF ALL TOOLS USED

Source codes are published in github:

9. REFERENCES

[1] Jiankang Deng, Jia Guo, Evangelos Ververas, Irene Kotsia, and Stefanos Zafeiriou, "Retinaface: Single-shot multi-level face localisation in the wild," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 5203–5212.

[2] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2980–2988.

[3] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie, "Feature pyramid networks for object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2117–2125.

[4] Beibei Xu, Wensheng Wang, Leifeng Guo, Guipeng Chen, Yongfeng Li, Zhen Cao, and Saisai Wu, "Cattlefacenet: A cattle face identification approach based on retinaface and arcface loss," *Computers and Electronics in Agriculture*, vol. 193, pp. 106675, 2022.

[5] Vivek Aswal, Omkar Tupe, Shifa Shaikh, and Nadir N Charniya, "Single camera masked face identification," in *2020 19th IEEE international conference on machine learning and applications (ICMLA)*. IEEE, 2020, pp. 57–60.

[6] Muhamad Dwisnanto Putro, Adri Priadana, Duy-Linh Nguyen, and Kang-Hyun Jo, "A faster real-time face detector support smart digital advertising on low-cost computing device," in *2022 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*. IEEE, 2022, pp. 171–178.