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 tailored for face detection. While it retains the core principles of RetinaNet, it introduces additional components specifically designed for face detection, such as:

1. Predicting facial landmarks in addition to face bounding boxes.
2. Incorporating a mesh decoder to predict pixel-wise 3D face shapes.

In this study, we will focus more on the face detection capabilities of RetinaFace. We aim to delve deeper into its workings and explore how to fine-tune it.

2.2. RetinaNet

To fully comprehend the intricacies of RetinaFace, it's imperative to first acquaint ourselves with RetinaNet, the foundational architecture upon which RetinaFace is built.

RetinaNet, introduced by [2], is a single-stage object detector that leverages the Focal Loss to tackle the class imbalance challenge inherent in object detection. The nomenclature "RetinaNet" is inspired by its employment of a Feature Pyramid Network (FPN) for multi-scale object detection, reminiscent of the human retina's multi-scale processing.

Historically, object detectors fell into two primary categories: single-stage detectors, such as YOLO and SSD, which prioritize speed over accuracy, and two-stage detectors like Faster R-CNN, which offer higher accuracy at the cost of speed. RetinaNet bridges this gap, designed as a single-stage detector but aspiring for the accuracy of its two-stage counterparts.

Central to RetinaNet's architecture is the FPN, which crafts a multi-scale pyramid of feature maps. This is achieved by merging low-resolution, semantically rich features with high-resolution, semantically weaker features through a top-down approach and lateral connections. Such a representation is pivotal for detecting objects of diverse sizes.

Addressing the class imbalance between foreground and background classes, especially prevalent in single-stage detectors, RetinaNet introduced the Focal Loss. This loss function dynamically scales, down-weighting easy negatives and emphasizing hard negatives and positives, thus facilitating more effective model training on challenging samples.

RetinaNet also utilizes anchor boxes, pre-defined bounding boxes of varied scales and aspect ratios, at each spatial location on its feature maps to predict object positions. For every anchor box, there's a classification head determining the object class and a regression head refining the bounding box coordinates. These heads operate uniformly across all FPN scales, ensuring consistent predictions.

Lastly, RetinaNet typically employs a deep convolutional network, such as ResNet, as its backbone for feature extraction from the input image, which subsequently feeds into the FPN.

2.2.1. Deeper into Focal Loss

The Focal Loss, tailored for addressing class imbalance in object detection, is particularly beneficial for single-stage detectors. In such detectors, the vast disparity between background (no object) and object classes arises due to the dense sampling of potential object locations, resulting in a preponderance of negative samples. The conventional cross-entropy loss, treating all samples uniformly, often allows these abundant easy negatives to dominate, leading to suboptimal models.

Focal Loss, in contrast, is formulated to diminish the influence of easy samples, emphasizing the harder ones. It's expressed as:

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

where p_t is the true class probability, α_t is a balancing factor, and γ is a focusing parameter that modulates the rate of down-weighting easy samples. Notably, with $\gamma = 0$, Focal Loss reverts to the standard cross-entropy loss, but as γ escalates, the modulation's impact grows, reducing the loss for easy samples more aggressively.

The modulating factor, $(1 - p_t)^\gamma$, ensures that easy examples experience a diminished loss. As γ augments, this modulation strengthens, causing the loss for easy samples to plummet, allowing the model to concentrate on challenging, misclassified instances.

The balancing factor, α_t, can be adjusted to counterbalance the significance of positive/negative classes, often set inversely proportional to class frequencies, offering an extra layer of mitigation against extreme class imbalances.

In essence, the Focal Loss prevents models from being swamped by numerous easy negatives, enabling detectors like RetinaNet to reach top-tier performance by surmounting one of their primary challenges. This dynamic loss scaling, emphasizing challenging samples, has been a transformative innovation in object detection, especially beneficial in contexts with pronounced class imbalances.

2.3. Feature Pyramid Network (FPN)

Historically, image pyramids in computer vision processed images at multiple scales to detect objects of varying sizes. Deep convolutional neural networks (CNNs) naturally produce feature maps at diverse resolutions: shallow layers capture high-resolution yet semantically weak features, while deep layers capture semantically rich features at a lower resolution. FPN [3] capitalizes on these multi-scale features for enhanced object detection.

The FPN architecture comprises:

Bottom-up Pathway: The standard forward pass in a CNN, where deeper layers offer increased semantic richness at the cost of spatial resolution.

Top-down Pathway and Lateral Connections: FPN introduces a top-down approach, upsampling spatial resolution. During this, lateral connections from the bottom-up pathway are integrated. These connections merge feature maps from the bottom-up pathway with upsampled features, amalgamating high-level semantic details with fine spatial information.

Multi-scale Predictions: Leveraging the unified features across the pyramid, FPN facilitates predictions at varied scales, adeptly detecting objects of diverse sizes.

Benefits: FPN ameliorates traditional CNN limitations, especially in detecting smaller objects. By fusing low-resolution, semantically potent features with high-resolution, semantically weaker ones, detection across scales is enhanced. Furthermore, FPN constructs a feature pyramid from a single-scale image, optimizing computational efficiency.

Usage in Modern Architectures: Serving as a cornerstone in many cutting-edge object detection architectures, including RetinaNet, FPN's multi-scale feature representations bolster a detector's capability to identify objects across sizes.

In essence, the Feature Pyramid Network (FPN) signifies a monumental stride in object detection, addressing multi-scale object detection challenges. By cohesively integrating multi-scale feature representations, FPN augments both the precision and efficiency of object detectors.

2.4. Anchor-based Face Detection on Pyramid Network

In RetinaFace, anchors are densely arranged across the feature maps of pyramid levels like P2. However, rather than predicting on a pixel-by-pixel basis, anchors correspond to spatial locations on the feature map, with their density governed by the stride of the pyramid level.

Each pyramid level's stride signifies the spatial resolution disparity between the original image and its feature map. For example, a stride of 4 for P2 implies each of its spatial locations corresponds to a 4x4 region in the original image. At each spatial location on a feature map, multiple anchors of varied scales and aspect ratios are positioned. This means that at a specific x,y location on P2, there exist multiple anchors, each differing in size or shape.

While anchors densely populate the feature map, they don't correspond to every pixel of the original image. They relate to the feature map's spatial locations. For instance, with P2's stride of 4, the first set of anchors might center at pixel (2,2) of the original image, with subsequent sets at pixels like (6,2), and so forth.

Despite not predicting for each pixel, the amalgamation of diverse scales and aspect ratios ensures comprehensive coverage of all potential objects. This dense arrangement, coupled with the anchors' varied dimensions, equips the model to detect faces of multiple sizes and orientations.

In essence, RetinaFace's dense anchor tiling across pyramid feature maps doesn't operate on a strict pixel-by-pixel basis. Predictions are tethered to feature map spatial locations, with the density influenced by the pyramid level's stride.

2.5. Context Module in RetinaFace

The context module stands as a pivotal component within the RetinaFace architecture, designed to harness contextual information across various feature levels. Multi-level feature fusion, prevalent in many face detection architectures, is employed to cater to faces of diverse sizes and contexts. By aggregating context from different scales, the context module bolsters the feature representation, proving invaluable for detecting small faces or those in challenging scenarios where context offers crucial insights.

This module's integration results in enhanced detection performance, especially in situations with partial occlusions, varying face sizes, or challenging lighting. Essentially, the context module acts as a conduit to assimilate and capitalize on information from different scales, fortifying the RetinaFace detector's precision and robustness across varied real-world settings.

2.5.1. How Context Module works with FPN

The context module refines the multi-scale features furnished by the FPN by aggregating context, offering a dual enhancement. Initially, the FPN delivers multi-scale features, which the context module subsequently refines by integrating broader contextual data. This synergy between the FPN and the context module culminates in a richer feature representation for RetinaFace, making it more resilient and adept at face detection, particularly in demanding scenarios.

In a nutshell, while RetinaFace's FPN provides multi-scale feature representation, the context module further amplifies these features by incorporating contextual information, resulting in a potent face detection mechanism.

2.6. Cascade Regression

Cascade regression in RetinaFace involves a series of regression operations for iterative bounding box refinement. Instead of a one-shot prediction, the model makes an initial bounding box prediction and refines it in subsequent stages. This iterative approach aims to yield more accurate and stable bounding box predictions, especially in challenging scenarios.

2.7. Multi-task Loss

RetinaFace's multi-task loss is crafted to manage multiple objectives concurrently, including face classification, bounding box regression, facial landmark regression, and dense regression. The loss function is given by:

$$L = L_{cls}(p_i, p^*i) + \lambda_1 1p^i L * box(t_i, t^*i)$$

$$+\lambda_2 1p^i L * pts(l_i, l^*i) + \lambda_3 1p^i L * pixel$$

Where $1_{p_i^*}$ is an indicator function (1 for positive anchors), and $\lambda_1, \lambda_2, \lambda_3$ are loss-balancing parameters. The loss components include face classification (L_{cls}), bounding box regression (L_{box}), facial landmark regression (L_{pts}), and dense regression (L_{pixel}).

2.8. Cascade Regression with Multi-task Loss

Within RetinaFace, cascade regression refines bounding box predictions iteratively. While refining bounding boxes, the model concurrently optimizes other tasks using the multi-task loss, ensuring all tasks benefit from the iterative refinement. This amalgamation of cascade regression and multi-task loss enables the model to achieve precise bounding box predictions without sacrificing the performance of other tasks. In summary, the combination ensures iterative refinement of bounding box predictions, culminating in enhanced and consistent face detections.

3. IMPACT

The publication "RetinaFace: Single-shot Multi-level Face Localisation in the Wild" has left an indelible mark on face detection research, catalyzing numerous subsequent investigations.

3.1. Improving the Architecture

RetinaFace's architecture, while pivotal in human face detection, has seen adaptations for other species. For instance, [4] introduced "CattleFaceNet: A cattle face identification approach based on RetinaFace and ArcFace loss," which melds RetinaFace's strengths with the ArcFace loss for enhanced cattle face recognition. Such innovative applications highlight the model's versatility across varied identification challenges.

3.2. Adapting to Different Domains

RetinaFace's adaptability shines in its applications across diverse scenarios. A pertinent example is [5]'s "Single Camera Masked Face Identification," addressing the complexities of identifying mask-obscured faces, a contemporary challenge. This underscores RetinaFace's ability to navigate domain-specific intricacies.

3.3. Optimization for Real-time Applications

Real-time face detection's importance, especially in constrained environments, has led to RetinaFace's optimization endeavors. [6]'s "A Faster Real-time Face Detector Support Smart Digital Advertising on Low-cost Computing Device" accentuates the model's significance in digital advertising on budget devices, underscoring its real-world applicability.

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.

5. CONCLUSION

6. STATEMENT OF ALL TOOLS USED

Source codes are published in github:

7. 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.