Sample and Computation Redistribution for Efficient Face Detection

Jia Guo [*] [2] Jiankang Deng [*] [1,2] Alexandros Lattas[1] Stefanos Zafeiriou[1]
[1]Imperial College [2]InsightFace
guojia@gmail.com, {j.deng16, a.lattas,s.zafeiriou}@imperial.ac.uk

Abstract

Although tremendous strides have been made in uncontrolled face detection, efficient face detection with a low computation cost as well as high precision remains an open challenge. In this paper, we point out that training data sampling and computation distribution strategies are the keys to efficient and accurate face detection. Motivated by these observations, we introduce two simple but effective methods (1) Sample Redistribution (SR), which augments training samples for the most needed stages, based on the statistics of benchmark datasets; and (2) Computation Redistribution (CR), which reallocates the computation between the backbone, neck and head of the model, based on a meticulously defined search methodology. Extensive experiments conducted on WIDER FACE demonstrate the state-of-the-art efficiency-accuracy trade-off for the proposed `SCRFD` *family across a wide range of compute regimes. In particular,* `SCRFD-34GF` *outperforms the best competitor, TinaFace, by* 3.86% *(AP at hard set) while being more than* $3\times$ faster *on GPUs with VGA-resolution images. We also release our code to facilitate future research.* https://github.com/deepinsight/insightface/tree/master/detection/scrfd

1. Introduction

Face detection is a long-standing problem in computer vision with many applications, such as face alignment [2], face reconstruction [9], face attribute analysis [36, 26], and face recognition [28, 6]. Following the pioneering work of Viola-Jones [32], numerous face detection algorithms have been designed. Among them, the single-shot anchor-based approaches [25, 39, 30, 15, 24, 7, 23, 41] have recently demonstrated the most promising performance. In particular, on the most challenging face detection dataset WIDER FACE [34], the average precision (AP) on its Hard test set has been boosted to 92.4% by TinaFace [41].

[*] Equal contributions. InsightFace is a nonprofit face project.

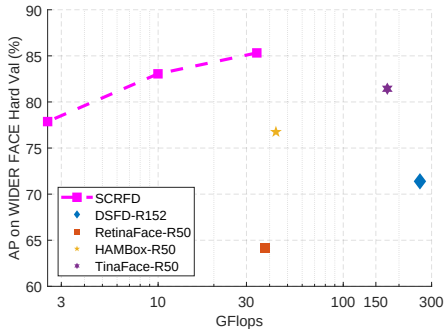


Figure 1. Performance-computation trade-off on the WIDER FACE validation hard set for different face detectors. Flops and APs are reported by using the VGA resolution (640×480) during testing. The proposed `SCRFD` outperforms a range of state-of-the-art open-sourced methods by using much fewer flops.

Even though TinaFace [41] achieves impressive results on unconstrained face detection, it employs large-scale (*e.g.* $1,650$ pixels) testing, which consumes huge amounts of computational cost (as shown in Tab. 1). In addition, TinaFace is designed based on a generic object detector (*i.e.* RetinaNet [19]), directly taking the classification network as the backbone, tiling dense anchors on the multi-scale feature maps (*i.e.* P2 to P7 of neck) and adopting heavy head designs. Without considering the prior of faces, the backbone, neck and head design of TinaFace is thus redundant and sub-optimal.

Since directly taking the backbone of the classification network for object detection is sub-optimal, the recent CR-NAS [17] reallocates the computation across different resolutions. It is based on the observation that the allocation of computation across different resolutions has a great impact on the Effective Receptive Field (ERF) and affects the detection performance. In BFbox [22], a meaningful phenomenon has been observed that the same backbone performs inconsistently between the task of general object detection on COCO [20] and face detection on WIDER FACE [34], due to the huge gap of scale distribution. Based on this observation, a face-appropriate search space is designed in BFbox [22], including a joint optimisation of backbone and neck. In ASFD [35], another interesting phenomenon has

been observed that some previous Feature Enhance Modules (FAE) perform well in generic object detection but fail in face detection. To this end, a differential architecture search is employed in ASFD [35] to discover optimised feature enhance modules for efficient multi-scale feature fusion and context enhancement.

Even though the works [22, 35] have realised the limitation of directly applying general backbone, neck and head settings to face detection, CR-NAS [17] only focuses the optimisation on backbone, BFbox [22] neglects the optimisation of head, and ASFD [35] only explores the best design for neck.

In this paper, we explore efficient face detection under a fixed VGA resolution (*i.e.* 640×480) instead of using large-scale for testing [41] to reduce the computation cost. Under this scale setting, most of the faces (78.93%) in WIDER FACE are smaller than 32×32 pixels, and thus they are predicted by shallow stages. To obtain more training samples for these shallow stages, we first propose a Sample Redistribution (SR) method through a large cropping strategy.

Moreover, as the structure of a face detector determines the distribution of computation and is the key in determining its accuracy and efficiency, we further discover principles of computation distribution under different flop regimes. Inspired by [27], we control the degrees of freedom and reduce the search space. More specifically, we randomly sample model architectures with different configurations on backbone (stem and four stages), neck and head. Based on the statistics of these models, we follow [27] to compute the empirical bootstrap [8] and estimate the likely range in which the best models fall. To further decrease the complexity of the search space, we divide the computation ratio estimation for backbone and the whole detector into two steps.

To sum up, this paper makes the following contributions:

- We explore the efficiency of face detection under VGA resolution, and propose a sample redistribution strategy (SR) that helps to obtain more training samples for shallow stages.
- We design a simplified search space for computation redistribution across different components (backbone, neck and head) of a face detector. The proposed two-step computation redistribution method can easily gain insight on computation allocation.
- Extensive experiments conducted on WIDER FACE demonstrate the significantly improved accuracy and efficiency trade-off of the proposed `SCRFD` across a wide range of compute regimes as shown in Fig. 1.

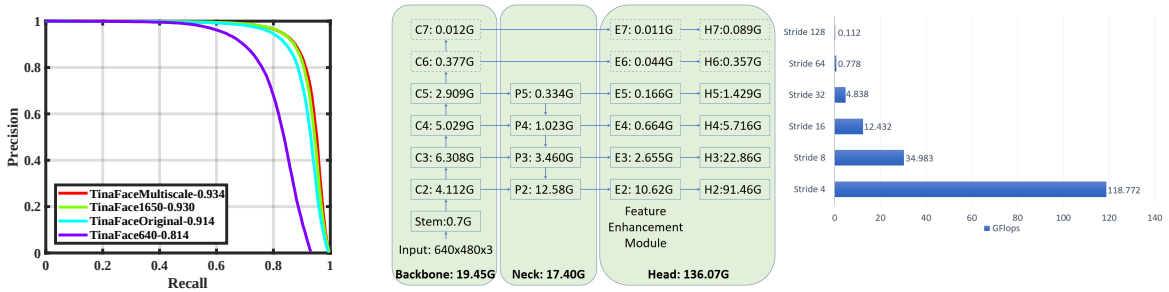
2. Related Work

Face Detection. To deal with extreme variations (*e.g.* scale, pose, illumination and occlusion) in face detection [34], most of the recent single-shot face detectors focus on improving the anchor sampling/matching or feature enhancement. SSH [25] builds detection modules on different feature maps with a rich receptive field. S^3FD [39] introduces an anchor compensation strategy by offsetting anchors for outer faces. PyramidBox [30] formulates a data-anchor-sampling strategy to increase the proportion of small faces in the training data. DSFD [15] introduces small faces supervision signals on the backbone, which implicitly boosts the performance of pyramid features. Group sampling [24] emphasizes the importance of the ratio for matched and unmatched anchors. RetinaFace [7] employs deformable context modules and additional landmark annotations to improve the performance of face detection. HAMBox [23] finds that many unmatched anchors in the training phase also have strong localization ability and proposes an online high-quality anchor mining strategy to assign high-quality anchors for outer faces. BFbox [22] employs a single-path one-shot search method [11] to jointly optimise the backbone and neck for face detector. ASFD [35] explores a differential architecture search to discover optimised feature enhance modules for efficient multi-scale feature fusion and context enhancement. All these methods are either designed by expert experience or partially optimised on backbone, neck and head. By contrast, we search for computation redistribution across different components (backbone, neck and head) of a face detector across a wide range of compute regimes, which is a global optimisation.

Neural Architecture Search. Given a fixed search space of possible networks, Neural Architecture Search (NAS) automatically finds a good model within the search space. DetNAS [4] adopts the evolution algorithm for the backbone search to boost object detection on COCO [20]. By contrast, CR-NAS [17] reallocates the computation across different stages within the backbone to improve object detection. NAS-FPN [10] uses reinforcement learning to search the proper FPN for general object detection. As there is an obvious distribution gap between COCO [20] and WIDER FACE [34], the experience in the above methods is not directly applicable for face detection but gives us an inspiration that the backbone, neck and head can be optimised to enhance the performance of face detection. Inspired by RegNet [27], we optimise the computation distribution on backbone, neck and head based on the statistics from a group of randomly sampled models. We successfully reduce the search space and find the stable computation distribution under a particular complex regime, which significantly improves the model's performance.

3. TinaFace Revisited

Based on RetinaNet [18], TinaFace [41] employs ResNet-50 [12] as backbone, and Feature Pyramid Network (FPN) [18] as neck to construct the feature extractor. For



(a) Different Testing Scales

(b) Computation Distribution of TinaFace@640 × 480

Figure 2. (a) Precision-recall curves of TinaFace-ResNet50 on the WIDER FACE hard validation subset, under different testing scales. (b) Computation distribution of TinaFace on backbone, neck and head with 640 × 480 as the testing scale.

Testing Scale	AP	AP@Precision > 98%	#Flops(G)
Multi-scale	0.934	0.746	42333.64
1650	0.930	0.750	1021.82
Original(1024)	0.914	0.706	508.47
640	0.814	0.539	172.95

Table 1. Performance and computation comparisons of TinaFace under different testing scales. The average scale of original images is around 882 × 1024.

the head design, TinaFace first uses a feature enhancement module on each feature pyramid to learn surrounding context through different receptive fields in the inception block [29]. Then, four consecutive 3×3 convolutional layers are appended on each feature pyramid. Focal loss [19] is used for the classification branch, DIoU loss [40] for the box regression branch and cross-entropy loss for the IoU prediction branch.

To detect tiny faces, TinaFace tiles anchors of three different scales, over each level of the FPN (*i.e.* $\{2^{4/3}, 2^{5/3}, 2^{6/3}\} \times \{4, 8, 16, 32, 64, 128\}$, from level P_2 to P_7). The aspect ratio is set as 1.3. During training, square patches are cropped from the original image and resized to 640×640, using a scaling factor randomly sampled from $[0.3, 0.45, 0.6, 0.8, 1.0]$, multiplied by the length of the original image's short edge. During testing, TinaFace employs single scale testing, when the short and long edge of the image do not surpass $[1100, 1650]$. Otherwise, it employs with short edge scaling at $[500, 800, 1100, 1400, 1700]$, shift with directions $[(0, 0), (0, 1), (1, 0), (1, 1)]$ and horizontal filp.

As shown in Fig. 2(a) and Tab. 1, we compare the performance of TinaFace under different testing scales. For the multi-scale testing, TinaFace achieves an impressive AP of 93.4%, which is the current best performance on the WIDER FACE leader-board. For large single-scale testing (1650), the AP slightly drops at 93.0% but the computation significantly decreases to 1021.82 Gflops. On the original scale (~1024), the performance of TinaFace is still very high, obtaining an AP of 91.4% with 508.47 Gflops. Moreover, when the testing scale decreases to VGA level (640), the AP significantly reduces to 81.4%, with the computation further decreasing at 172.95 Gflops.

In Fig. 2(b), we illustrate the computation distribution of TinaFace on the backbone, neck and head components with a testing scale of 640. From the view of different scales of the feature pyramid, the majority of the computational costs (about 68%) are from stride 4, as the resolution of the feature map is quite large (120×160). From the view of the different components of the face detector, most of the computational costs (about 79%) are from the head, since the backbone structure is directly borrowed from the ImageNet classification task [5], without any modification.

Even though TinaFace achieves state-of-the-art performance on tiny face detection, the heavy computational cost renders it unsuitable for real-time applications. In addition, real-world face detection systems always require high precision (*e.g.* > 98%), in order to avoid frequent false alarms. As shown in Tab. 1, the APs of TinaFace significantly drop, when the threshold of detection scores is increased to meet the requirement of high precision.

4. Methodology

Based on the above analysis of TinaFace, and the following meticulous experimentation, we propose the following efficiency improvements on the design of face detection, conditioned on (1) the testing scale bounded at the VGA resolution (640), and (2) there is no anchor tiled on the feature map of stride 4. In particular, we tile anchors of $\{16, 32\}$ at the feature map of stride 8, anchors of $\{64, 128\}$ at stride 16, and anchors of $\{256, 512\}$ at stride 32. Since our testing scale is smaller, most of the faces will be predicted on stride 8. Therefore, we first investigate the redistribution of positive training samples across different scales of feature maps (Sec. 4.1). Then, we explore the computation redistribution across different scales of feature maps, as well as across different components (*i.e.* backbone, neck and head), given a pre-defined computation budget (Sec. 4.2).

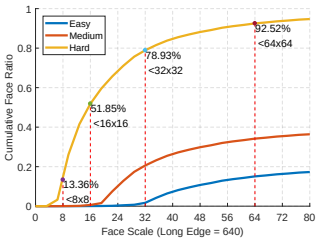
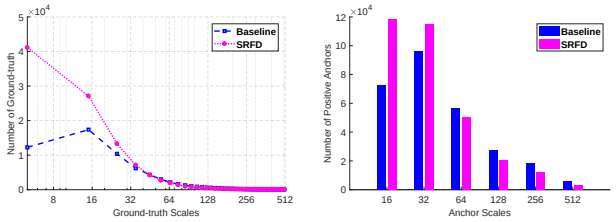


Figure 3. Cumulative face scale distribution on the WIDER FACE validation dataset (Easy $\subset$ Medium $\subset$ Hard). When the long edge is fixed as 640 pixels, most of the easy faces are larger than 32×32, and most of the medium faces are larger than 16×16. For the hard track, 78.93% faces are smaller than 32×32, 51.85% faces are smaller than 16×16, and 13.36% faces are smaller than 8×8.



(a) Ground-truth Distribution (b) Positive Anchor Distribution

Figure 4. Ground-truth and positive anchor distribution within one epoch. The baseline method employs a random size from the set $[0.3, 1.0]$, while our method uses a random size from the set $[0.3, 2.0]$. The number of small faces ($< 32 \times 32$) significantly increases after the large cropping strategy is used.

4.1. Sample Reallocation

We find that the feature map of stride 8 is most important in our setting. This is evident in Fig. 3, where we draw the cumulative face scale distribution on the WIDER FACE validation dataset. When the test scale is fixed as 640 pixels, 78.93% faces are smaller than 32×32.

During training data augmentation, square patches are cropped from the original images with a random size from the set $[0.3, 1.0]$ of the short edge of the original images. To generate more positive samples for stride 8, we enlarge the random size range from $[0.3, 1.0]$ to $[0.3, 2.0]$. When the crop box is beyond the original image, average RGB values fill the missing pixels. As illustrated in Fig. 4(a), there are more faces below the scale of 32 after the proposed large cropping strategy is used. Moreover, even though there will be more extremely tiny faces (*e.g.* $< 4 \times 4$) under the large cropping strategy, these ground-truth faces will be neglected during training due to unsuccessful anchor matching. As shown in Fig. 4(b), positive anchors within one epoch significantly increase from 72.3K to 118.3K at the scale of 16, and increase from 95.9K to 115.1K at the scale of 32. With more training samples redistributed to the small scale, the branch to detect tiny faces can be trained more adequately.

4.2. Computation Redistribution

Directly utilising the backbone of a classification network for scale-specific face detection can be sub-optimal. Therefore, we employ network structure search [27] to reallocate the computation on the backbone, neck and head, under a wide range of flop regimes. We apply our search method on RetinaNet [18], with ResNet [12] as backbone, Path Aggregation Feature Pyramid Network (PAFPN) [21] as the neck and stacked 3×3 convolutional layers for the head. While the general structure is simple, the total number of possible networks in the search space is unwieldy. In the first step, we explore the reallocation of the computation within the backbone parts (*i.e.* stem, C2, C3, C4, and C5), while fixing the neck and head components. Based on the optimised computation distribution on the backbone that we find, we further explore the reallocation of the computation across the backbone, neck and head. By optimising in both manners, we achieve the final optimised network design for face detection.

Computation search space reduction. Our goal is to design better networks for efficient face detection, by redistributing the computation. Given a fixed computation cost, as well as the face scale distribution presented in Fig. 3, we explore the relationship between computation distribution and performance from populations of models.

Following RegNet [27], we explore the structures of face detectors, assuming fixed standard network blocks (*i.e.*, basic residual or bottleneck blocks with a fixed bottleneck ratio of 4). In our case, the structure of a face detector includes: (1) the *backbone stem*, three 3×3 convolutional layers with w_0 output channels [13], (2) the *backbone body*, four stages operating at progressively reduced resolution, with each stage consisting of a sequence of identical blocks. For each stage i, the degrees of freedom include the number of blocks d_i (*i.e.* network depth) and the block width w_i (*i.e.* number of channels). The structure of the face detector also includes: (3) the *neck*, a multi-scale feature aggregation module by a top-down path and a bottom-up path with n_i channels [21], (4) the *head*, with h_i channels of m blocks to predict face scores and regress face boxes.

As the channel number of the stem is equal to the block width of the first residual block in C2, the degree of freedom of the stem can be merged into w_1. In addition, we employ a shared head design for three-scale of feature maps and fix the channel number for all 3×3 convolutional layers within the heads. Therefore, we reduce the degrees of freedom to three within our neck and head design: (1) output channel number n for neck, (2) output channel number h for head, and (3) the number of 3×3 convolutional layers m. We perform uniform sampling of $n \leq 256$, $h \leq 256$, and $m \leq 6$ (both n and h are divisible by 8).

The backbone search space has 8 degrees of freedom as there are 4 stages and each stage i has 2 parameters: the

number of blocks d_i and block width w_i. Following RegNet [27], we perform uniform sampling of $d_i \leq 24$ and $w_i \leq 512$ (w_i is divisible by 8). As state-of-the-art backbones have increasing widths [27], we also shrink the search space constrained to the principle of $w_{i+1} \geq w_i$.

With the above simplifications, our search space becomes more straightforward. We repeat the random sampling in our search space until we obtain 320 models in our target complexity regime, and train each model on the WIDER FACE training set for 80 epochs. Then, we test the AP of each model on the validation set. Based on these 320 pairs of model statistics (x_i, AP_i), where x_i is the computation ratio of a particular component and AP_i the corresponding performance, we follow [27] to compute the empirical bootstrap [8] to estimate the likely range in which the best models fall.

Finally, to further decrease the complexity of search space, we divide our network structure search into the following two steps:

- $\texttt{SCRFD}_1$: search the computational distribution for the backbone only, while fixing the settings of neck and head to the default configuration.
- $\texttt{SCRFD}_2$: search the computational distribution over the whole face detector (*i.e*. backbone, neck and head), with the computational distribution within the backbone following the optimised $\texttt{SCRFD}_1$.

Here, we use SCRFD constrained to 2.5 Gflops (SCRFD-2.5GF) as an example to illustrate our two-step searching strategy.

Computation redistribution on backbone. Since the backbone performs the bulk of the computation, we first focus on the structure of the backbone, which is central in determining the network's computational cost and accuracy. For $\texttt{SCRFD}_1$-2.5GF, we fix the output channel of the neck at 32 and use two stacked 3×3 convolutions with 96 output channels. As the neck and head configurations do not change in the whole search process of $\texttt{SCRFD}_1$, we can easily find the best computation distribution of the backbone. As described in Fig. 5, we show the distribution of 320 model APs (on the WIDER FACE hard validation set) versus the computation ratio over each component (*i.e*. stem, C2, C3, C4 and C5) of backbone. After applying an empirical bootstrap [8], a clear trend emerges showing that the backbone computation is reallocated to the shallow stages (*i.e*. C2 and C3). In Fig. 6, we present the computation ratio between the shallow (*i.e*. stem, C2, and C3) and deep stages (*i.e*. C4 and C5) of the backbone. Based on the observation from these search results, it is obvious that around 80% of the computation is reallocated to the shallow stages.

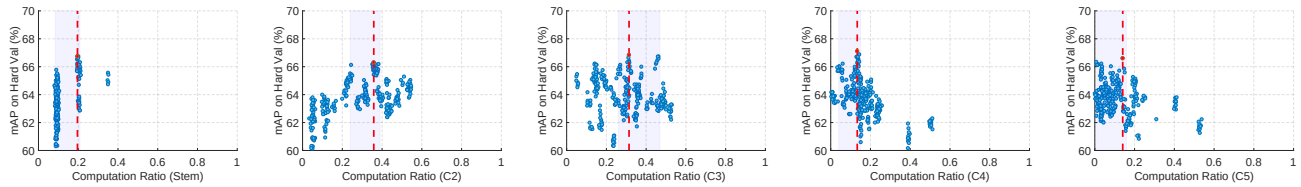
Computation redistribution on backbone, neck and head. After we find the optimised computation distribution within the backbone under a specific computational constraint of 2.5 Gflops, we search for the best computation distribution over the backbone, neck and head. In this step, we only keep the randomly generated network configurations whose backbone settings follow the computation distribution from $\texttt{SCRFD}_1$ as shown in Fig. 5.

Now there are another three degrees of freedom (*i.e*. output channel number n for neck, output channel number h for head, and the number of 3×3 convolutional layers m in head). We repeat the random sampling in our search space, until we obtain 320 qualifying models in our target complexity regime (*i.e*. 2.5 Gflops). As evident in Fig. 9, most of the computation is allocated in the backbone, with the head following and the neck having the lowest computation ratio. Fig. 9(d) depicts the comparison of the model architecture, under the constraint of 2.5 Gflops. The network configuration of the baseline (ResNet-2.5GF) is introduced in Tab. 2.

By employing the proposed two-step computation redistribution method, we find a large amount of capacity is allocated to the shallow stages, resulting in an AP improvement from 74.47% to 77.87% on the WIDER FACE hard validation set.

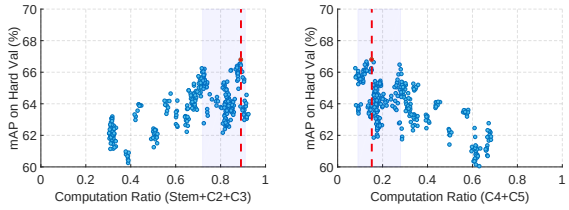
Higher compute regimes and mobile regime. Besides the complexity constraint of 2.5 Gflops, we also utilise the same two-step computation redistribution method to explore the network structure optimisation for higher compute regimes (*e.g*. 10 Gflops and 34 Gflops) and low compute regimes (*e.g*. 0.5 Gflops). In Fig. 7 and Fig. 8, we show the computation redistribution and the optimised network structures under different computation constraints.

Our final architectures have almost the same flops as the baseline network. From these redistribution results, we can draw the following conclusions: (1) more computation is allocated in the backbone and the computation on the neck and head is compressed; (2) more capacity is reallocated in shallow stages for the 2.5 Gflops, 10 Gflops and 34 Gflops regimes due to the specific scale distribution on WIDER FACE; (3) for the high compute regime (*e.g*. 34 Gflops), the explored structure utilises the bottleneck residual block and we observe significant depth scaling, instead of width scaling in shallow stages. Scaling the width is subject to over-fitting due to the larger increase in parameters [1]. By contrast, scaling the depth, especially in the earlier layers, introduces fewer parameters compared to scaling the width; (4) for the mobile regime (0.5 Gflops), allocating the limited capacity in the deep stage (*e.g*. C5) for the discriminative features captured in the deep stage, can benefit the shallow small face detection by the top-down neck pathway.



(a) Stem $\sim (10\%, 20\%)$ (b) C2 $\sim (24\%, 39\%)$ (c) C3 $\sim (26\%, 47\%)$ (d) C4 $\sim (4\%, 15\%)$ (e) C5 $\sim (1\%, 16\%)$

Figure 5. Computation redistribution on the backbone (stem, C2, C3, C4 and C5) with fixed neck and head under the constraint of 2.5 Gflops. For each component within the backbone, the range of computation ratio in which the best models may fall is estimated by the empirical bootstrap.



(a) Stem+C2+C3 $\sim (72\%, 91\%)$ (b) C4+C5 $\sim (9\%, 28\%)$

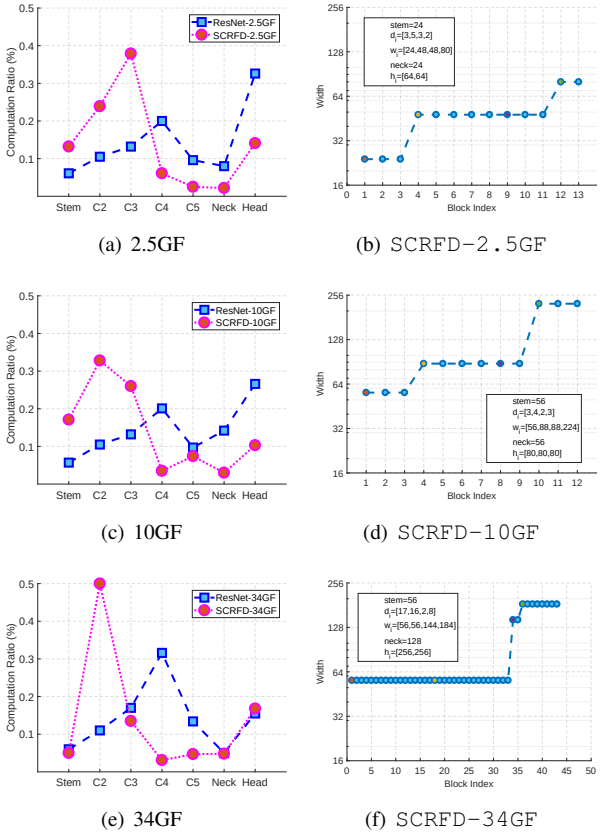
Figure 6. Computation redistribution between the shallow and deep stages of the backbone under the constraint of 2.5 Gflops.

Name	Backbone	Neck	Head
ResNet-2.5GF	ResNet34x0.25	48	[96,96]
ResNet-10GF	ResNet34x0.5	128	[160,160]
ResNet-34GF	ResNet50	128	[256,256]
MobileNet-0.5GF	MobileNetx0.25	32	[80,80]

Table 2. Network configurations for baselines across different compute regimes. $\times 0.25$ and $\times 0.5$ in backbone refer to decreasing the channel number by 0.25 and 0.5 compared to the original designs [12, 14]. Basic residual blocks are used in ResNet-2.5GF and ResNet-10GF, while bottleneck residual blocks are used in ResNet-34GF. For MobileNet-0.5GF, depth-wise convolution is used in both backbone and head.

Method	Easy	Medium	Hard
ResNet-2.5GF	91.87	89.49	67.32
SR + ResNet-2.5GF	93.21	91.11	74.47
CR@backbone	92.32	90.25	69.78
CR@detector	92.61	90.74	70.98
CR@two-step	92.66	90.72	71.37
SR+CR@two-step	93.78	92.16	77.87

Table 3. Ablation experiments of `SCRFD-2.5GF` (*i.e.* SR+CR@two-step) on the WIDER FACE validation subset. The baseline model is ResNet-2.5GF as introduced in Tab. 2. "SR" and "CR" denote the proposed sample and computation redistribution, respectively.



(a) 2.5GF (b) `SCRFD-2.5GF`

(c) 10GF (d) `SCRFD-10GF`

(e) 34GF (f) `SCRFD-34GF`

Figure 7. Computation redistribution (left column) and the searched network structures (right column) under different computation constraints (2.5GF, 10GF, and 34GF). Network diagram legends in the second row contain all information required to implement the `SCRFD` models that we have optimised the computation across stages and components.

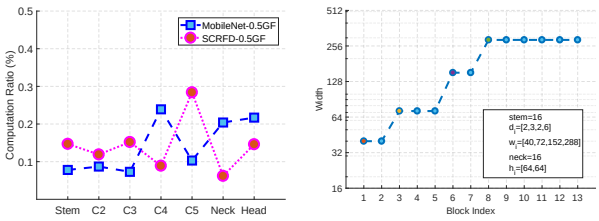
5. Experiments

5.1. Implementation Details

Training. For the baseline methods, square patches are cropped from the original images with a random size from $[0.3, 0.45, 0.6, 0.8, 1.0]$ and then these patches are resized to 640×640 for training. For the proposed Sample Redistribution (SR), we enlarge the random size set by adding $[1.2, 1.4, 1.6, 1.8, 2.0]$ scales. Besides scale augmentation, the training data are also augmented by color distortion and random horizontal flipping, with a probability of 0.5. For the anchor setting, we tile anchors of $\{16, 32\}$, $\{64, 128\}$, and $\{256, 512\}$ on the feature maps of stride 8, 16, and 32,

Method	Backbone	Easy	Medium	Hard	#Params(M)	#Flops(G)	Infer(ms)
DSFD [15] (CVPR19)	ResNet152	94.29	91.47	71.39	120.06	259.55	55.6
RetinaFace [7] (CVPR20)	ResNet50	94.92	91.90	64.17	29.50	37.59	21.7
HAMBox [23] (CVPR20)	ResNet50	95.27	93.76	76.75	30.24	43.28	25.9
TinaFace [41] (Arxiv20)	ResNet50	95.61	94.25	81.43	37.98	172.95	38.9
ResNet-34GF	ResNet50	95.64	94.22	84.02	24.81	34.16	11.8
SCRFD-34GF	Bottleneck Res	96.06	94.92	85.29	9.80	34.13	11.7
ResNet-10GF	ResNet34x0.5	94.69	92.90	80.42	6.85	10.18	6.3
SCRFD-10GF	Basic Res	95.16	93.87	83.05	3.86	9.98	4.9
ResNet-2.5GF	ResNet34x0.25	93.21	91.11	74.47	1.62	2.57	5.4
SCRFD-2.5GF	Basic Res	93.78	92.16	77.87	0.67	2.53	4.2
RetinaFace [7] (CVPR20)	MobileNet0.25	87.78	81.16	47.32	0.44	0.802	7.9
FaceBoxes [38] (IJCB17)	-	76.17	57.17	24.18	1.01	0.275	2.5
MobileNet-0.5GF	MobileNetx0.25	90.38	87.05	66.68	0.37	0.507	3.7
SCRFD-0.5GF	Depth-wise Conv	90.57	88.12	68.51	0.57	0.508	3.6

Table 4. Accuracy and efficiency of different methods on the WIDER FACE validation set. The size of test images is bounded by 640. The proposed sample redistribution is also employed by ResNet-34GF, ResNet-10GF, ResNet-2.5GF and MobileNet-0.5GF for fair comparisons.#Params and #Flops denote the number of parameters and multiply-adds. "Infer" refers to network inference latency with VGA resolution (640×480) images on NVIDIA 2080TI.



(a) 0.5GF (b) SCRFD-0.5GF

Figure 8. Computation redistribution (left) and the searched network structures (right) for the mobile regime (0.5 Gflops).

respectively. The anchor ratio is set as 1.0. In this paper, we employ Adaptive Training Sample Selection (ATSS) [37] for positive anchor matching. In the detection head, weight sharing and Group Normalisation [33] are used [31]. The losses of classification and regression branches are Generalised Focal Loss (GFL) [16] and DIoU loss [40], respectively.

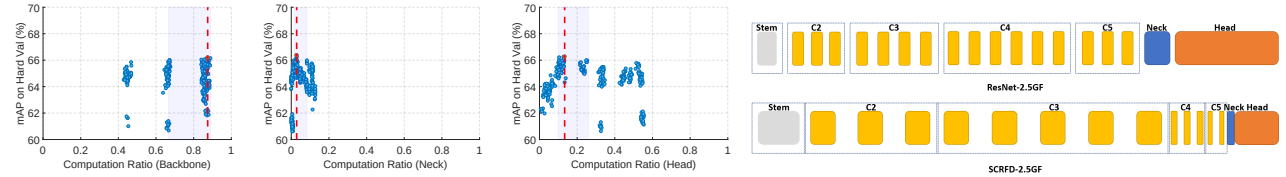
Our experiments are implemented in PyTorch, based on the open-source mmdetection [3]. We adopt the SGD optimizer (momentum 0.9, weight decay 5e-4) with a batch size of 8×4 and train on four Tesla V100. The initial learning rate is set to 0.00001, linearly warming up to 0.01 within the first 3 epochs. During network search, the learning rate is multiplied by 0.1 at the 56-th, and 68-th epochs. The learning process terminates on the 80-th epoch. For re-training of both baselines and the best configurations after search, the learning rate decays by a factor of 10 at the 440-th and 544-th epochs, and the learning process terminates at the 640-th epoch. For the computation constrained baselines in Tab. 2, we also employ the same training schedule. All the models are trained from scratch without any pre-training.

Testing. We only employ single-scale testing, bounded by the VGA resolution (640×480). The results of DSFD [15], RetinaFace [7], TinaFace [41] and Faceboxes [38] are reported by testing the released models, while the HAMBox [23] model is shared from the author.

5.2. Ablation Study

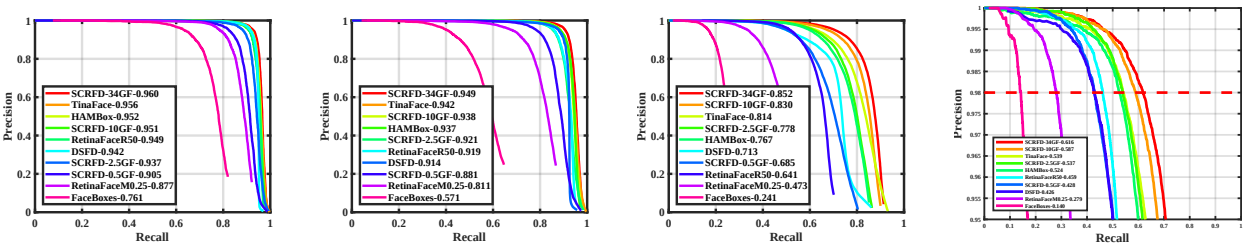
In Tab. 3 we present the AP performance of models by gradually including the proposed sample and computation redistribution methods. Our baseline model (*i.e.* ResNet-2.5GF as introduced in Tab. 2) gets 91.87%, 89.49%, and 67.32% in the three settings on the validation subset. By adding sample redistribution, the hard set AP significantly increases to 74.47%, indicating the benefit from allocating more training samples on the feature map of stride 8.

After separately employing the proposed computation redistribution on the backbone and the whole detector, the AP on the hard set improves to 69.78% and 70.98%. This indicates that (1) the network structure directly inherited from the classification task is sub-optimal for the face detection task, and (2) joint computation reallocation on the backbone, neck and head outperforms computation optimisation applied only on the backbone. Furthermore, the proposed two-step computation redistribution strategy achieves AP of 71.37%, surpassing one-step computation reallocation on the whole detector by 0.39%. As we shrink the whole search space by the proposed two-step strategy and our random model sampling number is fixed at 320, the two-step method is possible to find better network configurations from the large search space. Since sample redistribution and computation redistribution are orthogonal, their combination (*i.e.* SCRFD-2.5GF) shows a further improvement,



(a) Backbone $\sim (67\%, 88\%)$ (b) Neck $\sim (1\%, 7\%)$ (c) Head $\sim (10\%, 26\%)$ (d) Architecture Sketches

Figure 9. Computation redistribution and architecture sketches under the constraint of 2.5 Gflops. The computation distribution of `SCRFD-2.5GF` within backbone follows Fig. 5. Please refer to Tab. 2 for the network configuration of ResNet-2.5GF. In (d), the yellow rectangles in C2 to C5 represents the basic residual block. The width of rectangles corresponds to the computation cost. After computation redistribution, more computations are allocated to shallow stages (*i.e.* C2 and C3).



(a) Easy (b) Medium (c) Hard (d) Hard@Precision$>98\%$

(e) Visualisation Results of `SCRFD-2.5GF`

Figure 10. Precision-recall curves and qualitative results on the validation set of the WIDER FACE dataset. Yellow rectangles show the detection results by the proposed `SCRFD-2.5GF` and brightness encodes the detection confidence.

achieving AP of 77.87% on the hard track.

5.3. Accuracy and Efficiency on WIDER FACE

As shown in Tab. 4 and Fig. 10, we compared the proposed method with other state-of-the-art face detection algorithms (*e.g.* DSFD [15], RetinaFace [7], HAMBox [23] and TinaFace [41]). Since we fix the testing scale at 640, the flops and inference time directly correspond to the reported AP. The proposed `SCRFD-34GF` outperforms all these state-of-the-art methods on the three subsets, especially for the hard track, which contains a large number of tiny faces. More specifically, `SCRFD-34GF` surpasses TinaFace by 3.86% while being more than $3\times$ *faster* on GPUs. In addition, the computation cost of `SCRFD-34GF` is only around 20% of TinaFace. As `SCRFD-34GF` scales the depth in the earlier layers, it also introduces fewer parameters, resulting in a much smaller model size ($9.80M$).

Overall, all of the proposed `SCRFD` models (*e.g.* `SCRFD-34GF`, `SCRFD-10GF`, `SCRFD-2.5GF` and `SCRFD-0.5GF`) provide considerable improvements compared to the baseline models (listed in Tab. 2), by optimising the network structure alone (*i.e.* computation redistribution), across a wide range of compute regimes. For the low-compute regimes, `SCRFD-0.5GF` significantly outperforms RetinaFaceM0.25 by 21.19% on the hard AP, while consuming only 63.34% computation and 45.57% inference time.

For real-world face detection system, high precision (*e.g.* $> 98\%$) is required to avoid frequent false alarms. As shown in Fig. 10(d), `SCRFD-2.5GF` obtains comparable AP (53.7%) compared to TinaFace (53.9%) when the threshold score is improved to achieve prevision higher than 98%, while the computation cost is only 1.46% and the inference time is only 10.8%. Fig. 10(e) shows qualitative results generated by `SCRFD-2.5GF`. As can be seen, our face detector works very well in both indoor and outdoor crowded scenes under different conditions (*e.g.* appearance variations from pose, occlusion and illumination). The impressive performance across a wide range of scales indicate that `SCRFD-2.5GF` has a very high recall and can detect faces

accurately even without large scale testing.

6. Conclusions

In this work, we present a sample and computation redistribution paradigm for efficient face detection. Our results show significantly improved accuracy and efficiency trade-off by the proposed SCRFD across a wide range of compute regimes, when compared to the current state-of-the-art.

References

[1] Irwan Bello, William Fedus, Xianzhi Du, Ekin Cubuk, Aravind Srinivas, Tsung-Yi Lin, Jonathon Shlens, and Barret Zoph. Revisiting resnets: Improved training and scaling strategies. *arXiv:2103.07579*, 2021. 5

[2] Adrian Bulat and Georgios Tzimiropoulos. How far are we from solving the 2d & 3d face alignment problem?(and a dataset of 230,000 3d facial landmarks). In *ICCV*, 2017. 1

[3] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, et al. Mmdetection: Open mmlab detection toolbox and benchmark. *arXiv:1906.07155*, 2019. 7

[4] Yukang Chen, Tong Yang, Xiangyu Zhang, Gaofeng Meng, Chunhong Pan, and Jian Sun. Detnas: Neural architecture search on object detection. *arXiv:1903.10979*, 2019. 2

[5] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. 3

[6] Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos Zafeiriou. Arcface: Additive angular margin loss for deep face recognition. In *CVPR*, 2019. 1

[7] Jiankang Deng, Jia Guo, Yuxiang Zhou, Jinke Yu, Irene Kotsia, and Stefanos Zafeiriou. Retinaface: Single-stage dense face localisation in the wild. In *CVPR*, 2020. 1, 2, 7, 8

[8] Bradley Efron and Robert J Tibshirani. *An introduction to the bootstrap*. CRC press, 1994. 2, 5

[9] Yao Feng, Fan Wu, Xiaohu Shao, Yanfeng Wang, and Xi Zhou. Joint 3d face reconstruction and dense alignment with position map regression network. In *ECCV*, 2018. 1

[10] Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V Le. Nas-fpn: Learning scalable feature pyramid architecture for object detection. In *CVPR*, 2019. 2

[11] Zichao Guo, Xiangyu Zhang, Haoyuan Mu, Wen Heng, Zechun Liu, Yichen Wei, and Jian Sun. Single path one-shot neural architecture search with uniform sampling. *arXiv:1904.00420*, 2019. 2

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 2, 4, 6

[13] Tong He, Zhi Zhang, Hang Zhang, Zhongyue Zhang, Junyuan Xie, and Mu Li. Bag of tricks for image classification with convolutional neural networks. In *CVPR*, 2019. 4

[14] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv:1704.04861*, 2017. 6

[15] Jian Li, Yabiao Wang, Changan Wang, Ying Tai, Jianjun Qian, Jian Yang, Chengjie Wang, Jilin Li, and Feiyue Huang. Dsfd: dual shot face detector. In *CVPR*, 2019. 1, 2, 7, 8

[16] Xiang Li, Wenhai Wang, Lijun Wu, Shuo Chen, Xiaolin Hu, Jun Li, Jinhui Tang, and Jian Yang. Generalized focal loss: Learning qualified and distributed bounding boxes for dense object detection. *arXiv:2006.04388*, 2020. 7

[17] Feng Liang, Chen Lin, Ronghao Guo, Ming Sun, Wei Wu, Junjie Yan, and Wanli Ouyang. Computation reallocation for object detection. In *ICLR*, 2020. 1, 2

[18] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017. 2, 4

[19] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *ICCV*, 2017. 1, 3

[20] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014. 1, 2

[21] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. Path aggregation network for instance segmentation. In *CVPR*, 2018. 4

[22] Yang Liu and Xu Tang. Bfbox: Searching face-appropriate backbone and feature pyramid network for face detector. In *CVPR*, 2020. 1, 2

[23] Yang Liu, Xu Tang, Xiang Wu, Junyu Han, Jingtuo Liu, and Errui Ding. Hambox: Delving into online high-quality anchors mining for detecting outer faces. *CVPR*, 2020. 1, 2, 7, 8

[24] Xiang Ming, Fangyun Wei, Ting Zhang, Dong Chen, and Fang Wen. Group sampling for scale invariant face detection. In *CVPR*, 2019. 1, 2

[25] Mahyar Najibi, Pouya Samangouei, Rama Chellappa, and Larry S Davis. Ssh: Single stage headless face detector. In *ICCV*, 2017. 1, 2

[26] Hongyu Pan, Hu Han, Shiguang Shan, and Xilin Chen. Mean-variance loss for deep age estimation from a face. In *CVPR*, 2018. 1

[27] Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, and Piotr Dollár. Designing network design spaces. In *CVPR*, 2020. 2, 4, 5

[28] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *CVPR*, 2015. 1

[29] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *CVPR*, 2015. 3

[30] Xu Tang, Daniel K Du, Zeqiang He, and Jingtuo Liu. Pyramidbox: A context-assisted single shot face detector. In *ECCV*, 2018. 1, 2

[31] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. Fcos: Fully convolutional one-stage object detection. In *ICCV*, 2019. 7

[32] Paul Viola and Michael J Jones. Robust real-time face detection. *IJCV*, 2004. 1

[33] Yuxin Wu and Kaiming He. Group normalization. In *ECCV*, 2018. 7

[34] Shuo Yang, Ping Luo, Chen-Change Loy, and Xiaoou Tang. Wider face: A face detection benchmark. In *CVPR*, 2016. 1, 2

[35] Bin Zhang, Jian Li, Yabiao Wang, Ying Tai, Chengjie Wang, Jilin Li, Feiyue Huang, Yili Xia, Wenjiang Pei, and Rongrong Ji. Asfd: Automatic and scalable face detector. *arXiv:2003.11228*, 2020. 1, 2

[36] Feifei Zhang, Tianzhu Zhang, Qirong Mao, and Changsheng Xu. Joint pose and expression modeling for facial expression recognition. In *CVPR*, 2018. 1

[37] Shifeng Zhang, Cheng Chi, Yongqiang Yao, Zhen Lei, and Stan Z Li. Bridging the gap between anchor-based and anchor-free detection via adaptive training sample selection. In *CVPR*, 2020. 7

[38] Shifeng Zhang, Xiangyu Zhu, Zhen Lei, Hailin Shi, Xiaobo Wang, and Stan Z Li. Faceboxes: A cpu real-time face detector with high accuracy. In *IJCB*, 2017. 7

[39] Shifeng Zhang, Xiangyu Zhu, Zhen Lei, Hailin Shi, Xiaobo Wang, and Stan Z Li. S3fd: Single shot scale-invariant face detector. In *ICCV*, 2017. 1, 2

[40] Zhaohui Zheng, Ping Wang, Wei Liu, Jinze Li, Rongguang Ye, and Dongwei Ren. Distance-iou loss: Faster and better learning for bounding box regression. In *AAAI*, 2020. 3, 7

[41] Yanjia Zhu, Hongxiang Cai, Shuhan Zhang, Chenhao Wang, and Yichao Xiong. Tinaface: Strong but simple baseline for face detection. *arXiv:2011.13183*, 2020. 1, 2, 7, 8