2021 IEEE International Conference on Multimedia and Expo (ICME) | 978-1-6654-3864-3/20/$31.00 ©2021 IEEE | DOI: 10.1109/ICME51207.2021.9428258

MSFC: DEEP FEATURE COMPRESSION IN MULTI-TASK NETWORK

Zhicong Zhang[*†], *Mengyang Wang*[*†], *Mengyao Ma*[†], *Jiahui Li*[†], *Xiaopeng Fan*[*]

[*]Harbin Institute of Technology, [†]Huawei Technologies
{zhangzhicong4, wangmengyang4, ma.mengyao, lijiahui666}@huawei.com, fxp@hit.edu.cn

ABSTRACT

With the remarkable success of deep learning, a novel AI-deployment strategy on mobile devices called *collaborative intelligence* (CI) is proposed recently, which can greatly improve the efficiency of neural network by distributing workloads between mobile devices and the cloud. In order to reduce transmission overhead, feature maps obtained from mobile devices need to be compressed before being transmitted to the cloud. In this paper, we propose a multi-scale feature compression (MSFC) framework for applying complex multi-task learning network in CI deployment scenarios, which consists of a multi-scale feature fusion (MSFF) module, a single-stream feature codec (SSFC) and a multi-scale feature reconstruction (MSFR) module. When applied to the popular multi-task network Mask R-CNN, experimental results show that with less than 2% accuracy degradation, the proposed MSFC can compress the 32-bit floating point feature to *0.012* bits on average.

Index Terms— Collaborative intelligence, deep feature compression, multi-task learning

1. INTRODUCTION

Recently, the deep neural network (DNN) has shown its amazing performance in many computer vision tasks, such as image classification [1], object detection [2], visual question answering [3], etc. A common approach to apply AI-based applications on mobile is to use mobile devices as sensors to capture data, and then the data is transmitted to the cloud to perform complex neural network tasks. But this approach suffers from long latency and high communication overhead, and multiple mobile devices will cause congestion which limits the cloud throughput. To this end, a novel AI-deployment strategy on mobile devices called *Collaborative Intelligence* (CI) is proposed in [4], which aims to balance the workload between mobile devices and cloud to optimize the latency and energy efficiency. To be specific, a splitting point is carefully selected to divide the deep model into two parts, where one part extracts the feature maps from visual data on the mobile device, and the other part accomplishes subsequent calculations using the received features on the cloud.

To effectively reduce bandwidth cost and latency while ensuring little performance degradation, deep feature compression becomes quite important in CI scenarios. In [5], an HEVC (High Efficiency Video Coding) intra coding based deep feature compression method was studied on YOLO9000 [6]. Georgiadis [7] proposed a three-stage compression and acceleration pipeline that sparsifies, quantizes and entropy encodes activation maps of convolutional neural network (CNN). Eshratifar et al. [8] proposed a learnable reduction unit called bottleneck unit and a lossy compression-aware training method to realize feature compression on ResNet-50 [9] and VGG-19 [10]. Shao et al. [11] introduced an end-to-end architecture named BottleNet++ for efficient intermediate feature compression on classification networks. All these studies focus on single-task deep models.

With the widespread usage of deep learning, massive complex problems require solving a multitude of tasks concurrently, such as object detection, positioning, segmentation, etc. To solve this problem, multi-task learning (MTL) has attracted wide attention recently. Tang et al. [12] proposed a framework that embeds keypoints, heatmaps and segments from pose estimation into the multi-task learning pipeline for vehicle re-identification (ReID). A multi-task learning approach is proposed in [13] to simultaneously perform classification and segmentation of manipulated facial images. Architectures of multi-task learning models are often complex, and the above deep feature compression methods [5, 6, 7, 8, 11] are not suitable for multi-task models.

In [14], a multi-task network with compressible features for CI and a differentiable loss term are developed to measure the compressibility of bottleneck features. Wang et al. [15] proposed a deep feature compression network for multi-tasking and a quantization method based on importance of channels. Although the above two studies perform deep feature compression on multi-task models, the intermediate feature is a single-stream feature. It is necessary to study the feature compression for multi-stream multi-task networks.

In this paper, we propose an end-to-end trainable multi-scale feature compression (MSFC) framework for multi-stream multi-task networks, which consists of a multi-scale feature fusion (MSFF) module, a single-stream feature codec (SSFC) and a multi-scale feature reconstruction (MSFR) module. The MSFF and part of SSFC are deployed on the mobile and the remaining part of SSFC and MSFR are de-

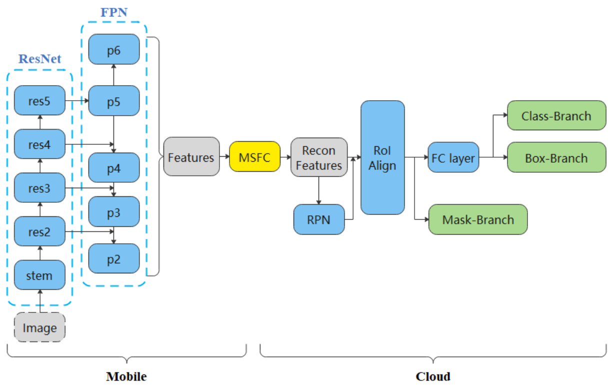


Fig. 1. Overview of the proposed scheme on Mask R-CNN [16]

ployed on the cloud. Considering the existence of much redundancy in multi-scale features, the MSFF combines multi-scale features into a single-feature map on the mobile. The SSFC takes the single-feature map as input for further feature compression. On the cloud-end, the multi-scale feature maps are reconstructed by the MSFR module based on the received features.

This paper is organized as follows: Section 2 presents preliminaries, and Section 3 gives an explanation about the proposed MSFC framework. Section 4 shows the experimental results that illustrate the efficacy of proposed method, followed by conclusion in Section 5.

2. PRELIMINARIES

Mask R-CNN [16], "a simple, flexible and general framework", which simultaneously complete object detection and instance segmentation tasks for each instance. This network extends Faster R-CNN [17] by adding a head for instance segmentation prediction. There are many technical essentials in Mask R-CNN. Firstly, a ResNet-50/101 [9] and a Feature Pyramid Network (FPN) [18] are used as feature extraction network to extract multi-scale semantic information. Secondly, a Region Proposal Network (RPN) gives proposals for an area expected to contain the target object. Thirdly, ROIPooling in Faster R-CNN is replaced by RoI Align to solve the misalignment problem resulting from direct sampling by pooling net. Finally, a special multi-task loss function corresponding to each task is designed as follows which offers a better training performance.

$$L = L_{cls} + L_{bbox} + L_{mask} \tag{1}$$

Experiments are carried out based on Mask R-CNN in this work. In order to apply multi-task learning network in CI deployment scenarios, we propose to split Mask R-CNN after FPN. Earlier or later splitting is not reasonable because an earlier splitting point in the backbone will cause the obtained features irregular, while a later splitting point in the branch will make the feature more specific for a single task and do not perform multi-tasks well. So, in Mask R-CNN, the part before the splitting point is deployed on the mobile and the other part is deployed on the cloud.

The input image can be represented by $(3, h, w)$ which corresponds to $(channels, height, width)$. By splitting Mask R-CNN after FPN, a feature pyramid is obtained, which consists of 5 multi-scale feature maps, labeled with $\{p_2, p_3, p_4, p_5, p_6\}$ respectively. The sizes of different feature maps in the feature pyramid are shown in Table 1. The number of elements in the feature pyramid is much larger than that of the input image, and a lot of redundancy is observed in the feature pyramid. In fact, the semantic information inside the feature pyramid is limited, so an efficient compression method can be used to remove the redundancy and extract a compact representation.

Table 1. Details of feature pyramid

	Size	Bit-depth
Input image	$(3, h, w)$	8
p_2	$256, h/4, w/4$	
p_3	$256, h/8, w/8$	
p_4	$256, h/16, w/16$	32
p_5	$256, h/32, w/32$	
p_6	$256, h/64, w/64$	

To compress the feature pyramid, we propose a multi-scale feature compression (MSFC) framework, which can be integrated into Mask R-CNN after FPN, as shown in Figure 1. More details of MSFC will be given in the next section. It is worth noting that although we take Mask R-CNN as an example to illustrate the proposed framework, MSFC can be easily combined with other multi-task learning networks.

3. PROPOSED METHOD

The proposed multi-scale feature compression (MSFC) framework is shown in Figure 2, which consists of a multi-scale feature fusion (MSFF) module, a single-stream feature codec (SSFC) and a multi-scale feature reconstruction (MSFR) module. The MSFF and part of SSFC are deployed on the mobile and the remaining part of SSFC and MSFR are deployed on the cloud.

3.1. MSFF module

Before designing the MSFF module, an experiment is carried out to evaluate the contributions of different feature maps to the task accuracy. To be more specific, Gaussian noise with zero mean and a variance of 100 is added to each feature map in $\{p_2, p_3, p_4, p_5, p_6\}$ respectively. By adding destructive noise to each feature map, the obtained task accuracy can

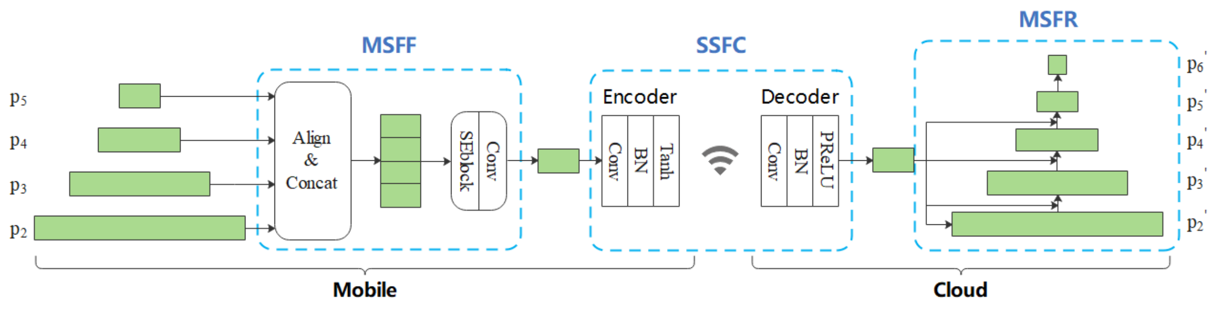


Fig. 2. The proposed multi-scale feature compression (MSFC) framework

reflect the importance of the features. The experiment results are shown in Figure 3. It can be observed that different feature maps in the feature pyramid have unequal importance. Adding noise to p_5 deteriorates the performance most, which indicates that p_5 is the most important among these maps.

In the feature pyramid, p_6 is obtained by pooling p_5. So p_6 has no more information than p_5 in terms of semantic information, therefore there is no need to transmit p_6. The MSFF module takes $\{p_2, p_3, p_4, p_5\}$ as input. Since p_5 is the most important among the feature pyramid, other feature maps are first downsampled to the size of p_5 and then concatenated together with p_5. Based on the observation that different feature maps in the feature pyramid have unequally importance, we use SE block [19] to estimate the importance map and re-weight the concatenated features. After that, a convolutional layer is used to perform channel-wise reduction on the re-weighted features, and the filter number of the layer is the same as the uncompressed features. In this module, the feature pyramid is fused to a single-feature map which has the same size as p_5.

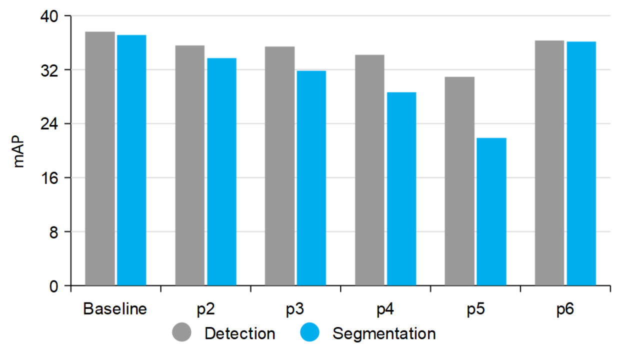


Fig. 3. mAP after adding Gaussian noise to each feature map

3.2. SSFC Module

The SSFC module consists of an encoder and a decoder, which are deployed on the mobile and the cloud respectively. The encoder performs further compression on the fused feature map, which composes a convolutional layer, a batch-normal layer and a Tanh activation function. By controlling the filter number of the convolutional layer, a flexible adjustment of compression ratio can be realized. An n-bit uniform quantization is adopted to further compress the feature map, will be discussed in Section 4. Owing to the Tanh activation function, the output of encoder is constrained within $[-1, 1]$, which is convenient for uniform quantization. A one-layer decoder is utilized to restore the feature map on the cloud.

The design of SSFC is motivated by [11] but there are several differences:

1) The activation function of the encoder is Tanh in this paper instead of Sigmoid in [11]. The Tanh activation function solves the non-zero-centered problem in Sigmoid and accelerates network convergence.
2) The activation function of the decoder is PReLU rather than ReLU in [11], which is an improved version of ReLU.
3) The encoder in [11] performs both channel-wise and spatial-wise reduction on feature maps, while in this paper the encoder only performs channel-wise reduction. This is based on an observation that spatial-wise reduction on feature map results in a significant performance degradation.

3.3. MSFR Module

A new feature pyramid is reconstructed using the features received by the cloud. Based on the above observation that the p_5 is the most important among the feature pyramid, we propose to use more parameters to predict the p_5. Thus a bottom-up architecture as shown in Figure 2 is adopted in MSFR module.

To be more specific, denoting the input of MSFR module as F', the reconstruction process is shown in the following formula:

$$p_2' = SizeAlign(F', p_2) \tag{2}$$
$$p_3' = Conv(SizeAlign(p_2', p_3)) + SizeAlign(F', p_3) \tag{3}$$
$$p_4' = Conv(SizeAlign(p_3', p_4)) + SizeAlign(F', p_4) \tag{4}$$
$$p_5' = Conv(SizeAlign(p_4', p_5)) + SizeAlign(F', p_5) \tag{5}$$
$$p_6' = Pooling(p_5') \tag{6}$$

Fig. 4. Snapshot of OpenImage dataset [20]

where $Conv$ denotes a 3×3 convolutional layer, the $SizeAlign(A, B)$ denotes aligning the size of A to the size of B through up/down sampling. After MSFR module, a new feature pyramid $\{p_2', p_3', p_4', p_5', p_6'\}$ is reconstructed to continue the rest network computation.

4. EXPERIMENTS

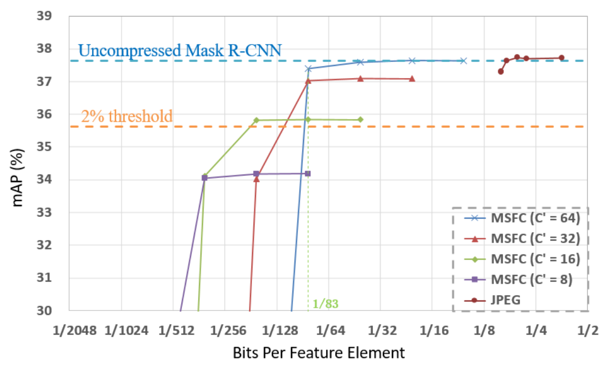
The proposed framework is implemented in Detectron2 [21] which is based on Pytorch [22]. Part of the OpenImage dataset [20] was used for training and testing, and 80 categories were manually selected including clothing, fish, car, guitar etc. There are 176107 images for training and 8746 images for testing in total. For each image, the corresponding bounding boxes and segmentation maps are also included in the dataset. A snapshot of the image dataset is shown in Figure 4. The input images are resized so that the shorter edges are 800 pixels.

As mentioned previously, the proposed MSFC framework can be trained end-to-end together with the Mask R-CNN. A three-step training strategy is utilized to train the total network. More specifically, firstly the layers in Mask R-CNN are optimized. Secondly, the proposed MSFC is inserted into the Mask R-CNN after FPN, and only the layers of MSFC is trained with other parameters frozen. Finally, the entire framework is fine-tuned in an end-to-end manner. Note that the uniform quantization is not applied in the training procedure, but only applied in the inference procedure. The training and inferencing processes are carried out on Tesla V100 GPUs.

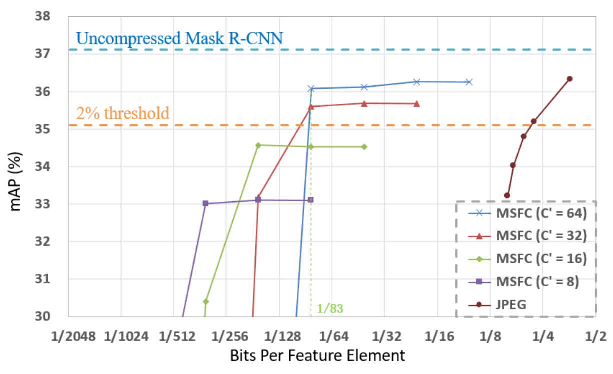
The metric for object detection and instance segmentation is mean Average Precision (mAP). The backbone of Mask R-CNN used in this work is ResNet-50 [9]. After being trained on the dataset, the original Mask R-CNN, i.e. without feature compression, achieves 37.63% mAP in object detection task, and 37.11% mAP in instance segmentation task.

The input image can be represented by $(3, h, w)$ which corresponding to $(channels, height, width)$, while the obtained feature pyramid is shown in Table 1. The feature pyramid is about 7 times as large as the input image in terms of activation elements and the number of bits in the feature pyramid is about 30 times that of the input image. After passing through the MSFF module, the feature maps are compressed to $(256, h/32, w/32)$ with 32 bit-depth, which is about 85 times smaller than the feature pyramid. Then the SSFC module can further reduce the bits by channel-wise feature compression and quantization. The final feature size after compression is $(C', h/32, w/32)$ with n bit-depth. In our experiment, C' is set to $[8, 16, 32, 64]$ respectively to test on different compression ratio, and n is set to $[2, 4, 8, 16, 32]$ respectively to verify the impact of quantization. The uniform quantization procedure is omitted when n is 32. We compare the proposed MSFC scheme with the classical image codec JPEG, which is applied after FPN to compress the feature pyramid. More specifically, the feature maps are quantized to 8-bits , re-arranged into an image, and then compressed by JPEG in different quality levels.

Based on different configurations of C' and n described above, the Bits Per Feature Element (BPFE) [14] is calculated respectively. The mAP performance on the two tasks of Mask R-CNN against BPFE is shown in Figure 5. Each curve of MSFC is determined by five points corresponding to $n = [2, 4, 8, 16, 32]$ respectively, and points lower than 30% mAP is omitted for better visual effect. As seen in Figure 5, the MSFC scheme achieves 10 times less BPFE than JPEG with the same task accuracy. With less than 2% accuracy degradation on both tasks, the proposed MSFC framework can compress the 32-bit floating point feature to 0.012 bits (1/83) on average with $C' = 64$ and $n = 4$, which means the MSFC framework achieves about 2600× bits saving compared with the origin feature pyramid. If more accuracy loss is allowed, the proposed MSFC can further increase the compression ratio, which cannot be achieved by JPEG even with



(a) mAP performance on object detection task



(b) mAP performance on instance segmentation task

Fig. 5. mAP performance vs. Bits Per Feature Element (BPFE) on the two tasks of Mask R-CNN

the maximum quantization level, as shown in Figure 5. The amazing performance mainly comes from the effective fusion of multi-scale feature maps and the exploration of compression and quantization. Although a naïve uniform quantization is used in this paper, the MSFC framework still shows great compression ability. There is no doubt that a more advanced quantization method will further improve the performance.

The above experimental results show that the proposed MSFC framework gains promising compression performance in the popular multi-task network Mask R-CNN. Even when the obtained multi-scale feature pyramid is relatively large in terms of the data volume, the MSFC framework can effectively remove the redundancy inside the feature maps, produce a compact representation, and ensure the accuracy of multiple tasks at the same time. In addition, as mentioned previously that although we take Mask R-CNN as an example to illustrate our proposed framework, MSFC can be easily combined with other multi-task learning networks.

5. CONCLUSION

In this paper, we propose a multi-scale feature compression (MSFC) framework. Firstly, a multi-scale feature fusion (MSFF) module is proposed to fuse multi-scale feature pyramid on the mobile devices, in which the redundancy inside feature maps is reduced based on the observation that different feature maps contribute differently to the accuracy. Then, a two-layer lightweight single-stream feature codec (SSFC) with uniform quantization is further used to produce a compact representation. Finally, a new feature pyramid is reconstructed by multi-scale feature reconstruction (MSFR) module based on the received feature map on the cloud. Experiments show that the proposed MSFC framework can compress the 32-bit floating point feature to 0.012 bits on average within 2% accuracy degradation.

6. REFERENCES

[1] A. Krizhevsky, I. Sutskever, and GE. Hinton, “Imagenet classification with deep convolutional neural networks,” *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[2] D. Erhan, C. Szegedy, A. Toshev, and D. Anguelov, “Scalable object detection using deep neural networks,” in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 2155–2162.

[3] W. Che, X. Fan, R. Xiong, and D. Zhao, “Visual relationship embedding network for image paragraph generation,” *IEEE Transactions on Multimedia*, vol. 22, no. 9, pp. 2307–2320, 2020.

[4] Yiping Kang, Johann Hauswald, Cao Gao, Austin Rovinski, Trevor Mudge, Jason Mars, and Lingjia Tang, “Neurosurgeon: Collaborative intelligence between the cloud and mobile edge,” *ACM SIGARCH Computer Architecture News*, vol. 45, no. 1, pp. 615–629, 2017.

[5] H. Choi and I. V. Bajić, “Deep feature compression for collaborative object detection,” in *2018 25th IEEE International Conference on Image Processing (ICIP)*, 2018, pp. 3743–3747.

[6] J. Redmon and A. Farhadi, “Yolo9000: Better, faster, stronger,” in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 6517–6525.

[7] G. Georgiadis, “Accelerating convolutional neural networks via activation map compression,” in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 7078–7088.

[8] Amir Erfan Eshratifar, Amirhossein Esmaili, and Massoud Pedram, “Bottlenet: A deep learning architecture for intelligent mobile cloud computing services,” in *2019 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*. IEEE, 2019, pp. 1–6.

[9] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[10] Karen Simonyan and Andrew Zisserman, “Very deep

convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[11] J. Shao and J. Zhang, "Bottlenet++: An end-to-end approach for feature compression in device-edge co-inference systems," in *2020 IEEE International Conference on Communications Workshops (ICC Workshops)*, 2020, pp. 1–6.

[12] Z. Tang, M. Naphade, S. Birchfield, J. Tremblay, W. Hodge, R. Kumar, S. Wang, and X. Yang, "Pamtri: Pose-aware multi-task learning for vehicle re-identification using highly randomized synthetic data," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 211–220.

[13] H. H. Nguyen, F. Fang, J. Yamagishi, and I. Echizen, "Multi-task learning for detecting and segmenting manipulated facial images and videos," in *2019 IEEE 10th International Conference on Biometrics Theory, Applications and Systems (BTAS)*, 2019, pp. 1–8.

[14] S. R. Alvar and I. V. Bajić, "Multi-task learning with compressible features for collaborative intelligence," in *2019 IEEE International Conference on Image Processing (ICIP)*, 2019, pp. 1705–1709.

[15] Weiqian Wang, Ping An, Chao Yang, and Xinpeng Huang, "Intermediate deep-feature compression for multitasking," in *Optoelectronic Imaging and Multimedia Technology VI*. International Society for Optics and Photonics, 2019, vol. 11187, p. 111870Z.

[16] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 2980–2988.

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

[18] T. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 936–944.

[19] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7132–7141.

[20] Ivan Krasin, Tom Duerig, Neil Alldrin, Vittorio Ferrari, Sami Abu-El-Haija, Alina Kuznetsova, Hassan Rom, Jasper Uijlings, Stefan Popov, Andreas Veit, et al., "Openimages: A public dataset for large-scale multi-label and multi-class image classification," *Dataset available from https://github.com/openimages*, vol. 2, no. 3, pp. 2–3, 2017.

[21] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick, "Detectron2," https://github.com/facebookresearch/detectron2, 2019.

[22] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al., "Pytorch: An imperative style, high-performance deep learning library," in *Advances in neural information processing systems*, 2019, pp. 8026–8037.