EfficientDet: Scalable and efficient object Detection

Nitin Bisht

21MM61R08, ntnbsht97@gmail.com

November 19, 2021

Abstract

Object detection is one of those core computer vision applications, which has attracted much research attention in recent years. It is due to its applications in video analysis, human behavior analysis, robotics and autonomous driving.

Tan et al. [1] proposed EfficientDet, a new architecture to obtain a faster and an accurate model. The architecture introduces EfficientNet as the backbone and involves Bidirectional Feature Pyramid Network (BiFPN), a multi-layer top-bottom and bottom-top network, for better multi-layer features fusion [1]. Unlike previous detector models where only the backbone part was scaled, EfficientDet involves uniform scaling of everything- backbone, feature network, and prediction network using a compound coefficient obtained heuristically to obtain a more efficient detector.

This paper is an **implementation paper**, where, unlike the typical COCO [2] dataset used in the original EfficientDet paper, Aquarium Dataset [3] is used to train the model. Then a few random ocean videos are taken and segregated into different frames. The trained model is then run on these different video frames, and various sea animals are detected. The video frames are then manually combined and converted into the video, with the labels of detected objects appearing in a rectangular box. In addition, a literature review section is added to highlight the previous works done in the field of object detection. **The code is available at the Github repository.**

1 Introduction

A typical object detection problem consists of different sub-tasks such as skeleton detection, pose detection, face detection and pedestrian detection. However, today, with the advent of Convolutional Neural Networks, there has been a paradigm shift from trivial Computer vision techniques in the Object Detection domain. Today the detector needs to be not only accurate but also fast enough to perform real-time detection. However, the accuracy and the speed are antagonists when it comes to detectors as more often than not, more accurate detectors tend to be computationally more demanding, which is not an ideal scenario for obvious reasons. So, a trade-off needs to done between speed and accuracy to obtain a scalable and efficient real-time object detector. Tan et al. from the Google Brain team came up with EfficientDet- an entirely new family of detectors that are accurate and more efficient, and faster. EfficientDet-D7 achieves state-of-the-art 55.1 AP on COCO test-dev [2] with 77M parameters and 410B FLOPs1, 4x – 9x smaller and 13x – 42x fewer FLOPs than previous object detectors [1] using different backbone and feature networks. Thus, making the model a lot quicker and more efficient compared to its predecessors.

The following section reviews the relevant work done in the field of the object detection problem.

2 Literature Review

P. Viola and M. Jones [4] [5] were the first to tackle the problem of object detection. They created a real-time human face detection technology based on sliding windows. N. Dalal and B. Triggs introduced the Histogram of Oriented Gradients (HOG) [6] descriptor, which significantly improved the scale-invariant feature transform in a picture. P. Felzenszwalb presented the Deformable Past based Model (DPM) [7], a HOG extension that employed the "divide and conquer" detection mechanism.

As computer architecture improved and GPUs were introduced, deep learning became the most common way to handle object classification. There have been two techniques to solving the object detection problem in the deep learning era:

1. Two-stage detection: In this method, the problem is divided into two steps. As a first stage, the model generates region proposals, which are subsequently classified into various groups.

2. One stage-detection: This model regards object detection as a regression or classification problem adopting a unified approach.

Overfeat [8], designed by Sermanet, Pierre, and others, was the first deep neural network for object detection. They created a multi-scale sliding window for object classification, localization, and detection using Convolutional Neural Networks. R. Girschick et al. later proposed Regions with CNN Features (RCNN) [9] for object detection.

K. He et al. suggested utilizing the Spatial Pyramid Pooling Network (SPPNet)[10], which was 20 times faster than R-CNN while maintaining accuracy. It added a new Spatial Pyramid Pooling Layer, which eliminates the need to rescale the input image. R. Girshick invented Fast-RCNN [11] in 2015, which allows a detector and a bounding box regressor to be trained simultaneously in the same network configuration. It was 200 times faster than the RCNN and 10% more accurate. S. Ren et al. introduced Faster RCNN [12], the first end-to-end and near-real-time detector, shortly after the Fast RCNN. The COCO dataset was able to achieve an Average Precision (AP) of 21.9 per cent.

YOLO (You Only Look Once)[13], the first one-stage detector in the deep learning era, was proposed by R.Joseph et al. in 2015. It has a high detection speed but at the expense of accuracy when compared to two-stage detectors. W.Liu et al. later proposed the Single Shot Multi-Box Detector (SSD), a second one-stage object detector [14]. Multi-reference and multi-resolution techniques were introduced, significantly improving accuracy. On the COCO dataset, it was able to attain an AP of 26.8

T.-Y. Lin et al. developed Feature Pyramid Networks (FPNs) [15], a top-down architecture with lateral connections, in 2017. It made significant progress in spotting objects of various scales and reached a 36.1 per cent accuracy on the COCO dataset. Later, he invented RetinaNet[16], a one-stage detector that included a new loss function called "focus loss" and achieved enhanced accuracy of 39.1 per cent while keeping one-stage detectors' high detection speed.

In 2019, B. Zoph, E. Cubuk, and colleagues combined an AmoebaNet-D architecture with a Neural Architecture Search-Feature Pyramid Network (NAS-FPN) [17], resulting in a state-of-the-art architecture. The calculation cost was the restriction of these systems with better accuracy. YOLO, RetinaNet, AmoebaNet+ NAS-FPN used 71B, 97B, and 3045B floating-point operations to obtain the needed precision (FLOPs). EfficientDet introduces weighted bi-directional feature pyramid network (BiFPN) and compound scaling approaches to address the problem of getting high accuracy without sacrificing computation cost.

In 2019, B. Zoph, E. Cubuk et al. introduced a Neural Architecture Search-Feature Pyramid Network (NAS-FPN) [17] with an AmoebaNet-D architecture, thus attaining a state-of-the-art architecture. The limitation of these architectures with improved accuracy was the computation cost. YOLO, RetinaNet, AmoebaNet+ NAS-FPN was using 71B, 97B and 3045B floating-point operations (FLOPs) to achieve the desired accuracy. To address this problem of achieving high accuracy without compromising computation cost, EfficientDet introduces weighted bi-directional feature pyramid network (BiFPN) and com-pound scaling methods. These architectural changes resulted in a 55.1 per cent accuracy on the COCO dataset while using just 410B FLOPs (13 times less than AmoebaNet +NAS-FPN).

3 Preliminaries

Tan et al. [1], which first introduced the EfficientDet, had three significant contributions:

1. **BiFPN**: A weighted bidirectional feature network for quicker feature fusion.

2. **Compound Scaling**: Unlike previous detectors, a new technique to scale backbone and feature network, resolution, and the class network.

3. **EfficientDet**: A new family of detectors that use EfficientNet as backbone architecture instead of ResNet.

The main challenges that the paper addressed were:

1. The detectors like RetinaNet, PANet [18], NAS-FPN, which were based on Feature Pyramid Techniques (FPN), considered all the input features contributed equally to the output features, which is not the case. So, multi-scale fusion is used.

2. The earlier models made the backbone architecture bigger for better accuracy. However, this paper scaled up the backbone and also the feature network and prediction network to improve accuracy and efficiency.

3.1 BiFPN

BiFPN is a successor of Feature Pyramid Networks (FPNs) [10] and has Path Aggregation Network (PANet) [18] as the base. There were three subtle variations made in the (PANet) [18] architecture:

1. The nodes with only one input edge will have less contribution to the feature network, and thus, was removed.

2. An extra edge, equivalent to a skip connection, was made from the original input to the output node at the same level to add more features at a lesser cost.

3. PANet had only one top-down and one bottom-up path; BiFPN has multiple such paths for multi-scale feature fusion.

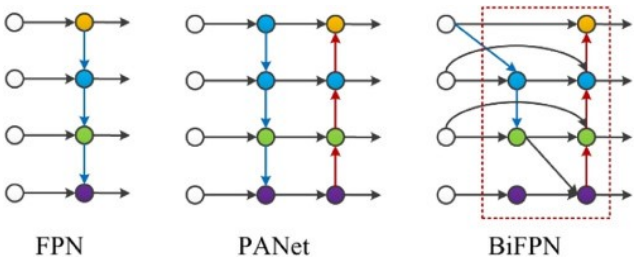


Figure 1: Network Design of FPN, PANet and BiFPN [19]

3.2 Weighted Feature Fusion

Tan et al. [1] introduced the Fast Normalized Fusion strategy to make weight assigning trainable so that the model can learn the optimal weights for different features.

$$O = \sum_i \frac{w_i}{\epsilon + \sum_j w_j} \cdot I_i,$$

This strategy makes the weight values bounded like a SoftMax based fusion but unlike SoftMax which has higher computational cost, this approach is very simple and at the same time efficient

3.3 EfficientDet

Finally, using EfficientNet as the backbone network, along with the BiFPNs, a new type of detector was obtained, namely EfficientDet.

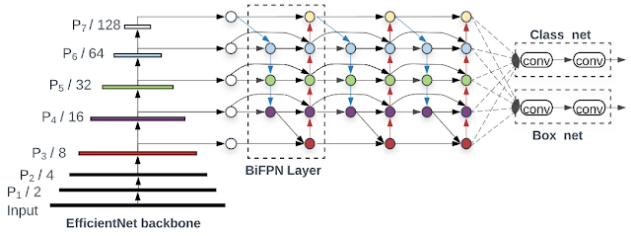


Figure 2: EfficientDet Architecture [1]

4 Architecture

As shown in Figure 2, the backbone networks for an EfficientDet architecture are EfficientNets. The BiFPNs work as feature networks with multiple bottom-top and top-down feature fusion. The features obtained are then fed to a class to produce an object class.

Additionally, a heuristic approach was taken to obtain a compound coefficient ϕ scale up all the dimensions of the backbone network, BiFPN network, resolution and the class/ box network. The following three equations were used for the BiFPN network, Box Prediction and Resolution, respectively.

$$W_{bifpn} = 64 \times (1.35^{\phi}), D_{bifpn} = 2 + \phi \quad (1)$$

$$D_{box} = D_{class} = 3 + \left\lfloor \frac{\phi}{3} \right\rfloor \quad (2)$$

$$R_{input} = 512 + \phi \times 128 \quad (3)$$

Using the three equations and with varying ϕ models from Efficient-D0 to Efficient-D6 were obtained which were computationally viable. The table summarizes all the configurations [1].

	Input size R_{input}	Backbone Network	BiFPN #channels W_{bifpn}	BiFPN #layers D_{bifpn}	Box/class #layers D_{class}
D0 ($\phi = 0$)	512	B0	64	2	3
D1 ($\phi = 1$)	640	B1	88	3	3
D2 ($\phi = 2$)	768	B2	112	4	3
D3 ($\phi = 3$)	896	B3	160	5	4
D4 ($\phi = 4$)	1024	B4	224	6	4
D5 ($\phi = 5$)	1280	B5	288	7	4
D6 ($\phi = 6$)	1408	B6	384	8	5
D7	1536	B6	384	8	5

Table 1: **Scaling configs for EfficientDet D0-D7 –** ϕ is the compound coefficient that controls all other scaling dimensions; *BiFPN, box/class net, and input size* are scaled up using equation 1, 2, 3 respectively. D7 has the same settings as D6 except using larger input size.

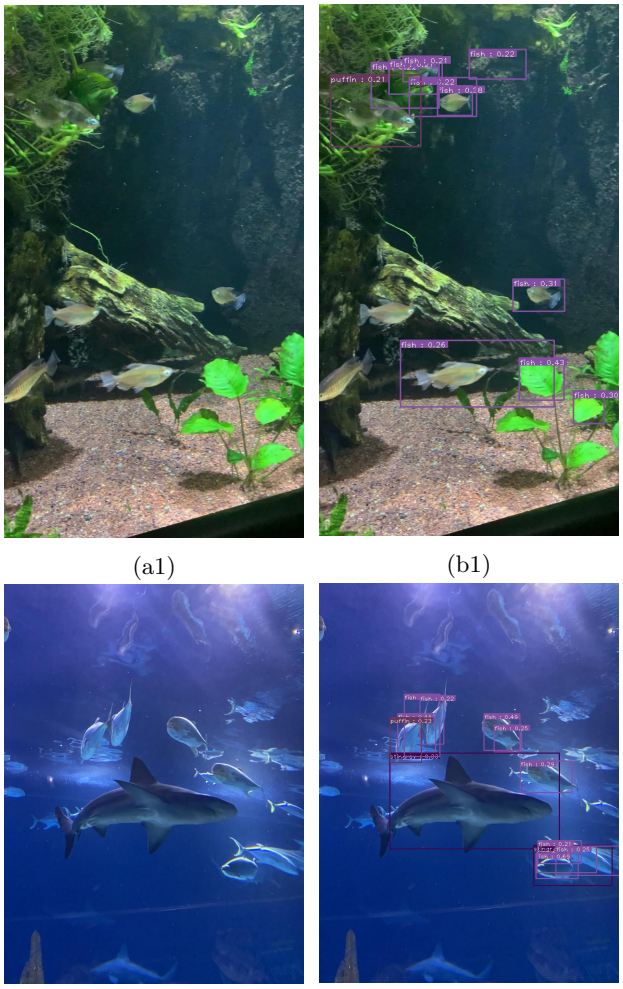
5 Implementation

This section details the implementation of EfficientDet architecture and the results obtained using EfficientDet on the Aquarium Combined dataset. The code for EfficientDet architecture is available as an open-source project on GitHub [20]. The repository is cloned to obtain the architecture. The architecture follows the one-stage detector paradigm. It employs ImageNet-pre-trained EfficientNets as the backbone network [1] and proposes a bi-connected Feature Pyramid Network as the feature network. It takes level 3-7 features from the backbone network and repeatedly applies top-down and bottom-up feature fusion. These fused features are then fed to a class and box network to produce object class and bounding box, respectively.

Since the EfficientDet architecture is a deep-learning-based model and uses a large dataset (here Aquarium Combined Dataset); therefore, the cloud-powered Google Colab is used to train the network and verify the proposed results [1]. Furthermore, the python-based deep-learning-library PyTorch [21] is used to implement the architecture.

The EfficientDet codebase in the GitHub repository mentioned above runs pre-trained PyTorch models on COCO Dataset. However, for this implementation paper, the Aquarium Database is required, which is obtained from the public dataset forum- Roboflow https://public.roboflow.com/object-detection/aquarium/2 [3]. The original EfficientDet research paper mentions nine different EfficientDet architectures (namely EfficientDet D0, D1, D2, D3, D4, D5, D6, D7 and D7x) available which differs in the number of sizes of backbone and feature networks, increasing from D0 to D7x. However, considering the computational constraints, the implementation is done on the most basic EfficientDet architecture.

The dataset, Aquarium Dataset, consists of 4817 annotated high dimension images of sea animals like fish, penguins, sharks, stingrays, starfish etc. collected by Roboflow from The Henry Doorly Zoo in Omaha (October 16, 2020) and the National Aquarium in Baltimore (November 14, 2020). These checkpoints are im-



(a1) (b1)

(a2) (b2)

Figure 3: Object-detection on images from Aquarium dataset.

ported from the codebase in Google Colab and applied on different images. A relatively smaller batch size of 8 is chosen, and the learning rate of 0.0001 is obtained heuristically. A $\text{vis}_{threshold}$ variable is set at 0.2 to limit the box size to predict the creature. Figure 3 shows an instance when the EfficientDet architecture is used to train the dataset. It shows the classifier output on an image and accurately classifies it as the output image is displayed as output.jpg in the .ipynb file.

To further assess the dataset's performance and develop insight from the paper, an interesting application of the model is made. The idea of Object Detection on images is extended to videos that can be used for some real-life problems like coral reef conservation, automated monitoring of the aquarium, the safety of the divers and swimmers and many more. This problem is tried to be solved using the OS library and the OpenCV [22] library for videos. The OpenCV library divides the video into different image frames and then uses the EfficientDet model to detect objects in each of these video frames. These frames are then fed to a temporary directory created by the OS library. The segregated video frames' height and width are set and are then joined together to obtain a modified video with the objects detected. Figure 4 shows the architecture used for object detection in videos.

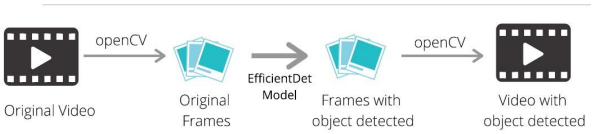


Figure 4: Object detection on video using EfficientDet

The pseudo code to implement the same is as follows:

Algorithm 1 Object detection on Videos

```
function VIDEO_OBJ_DET(model, video)
    frames ← vid_to_frames(video)
    outVideo ← new video()
    for frame in frames do
        outFrame ← model(frame)
        outVideo ← outVideo + outFrame
    end for
    return outVideo
end function
```

The videos were obtained from Pexels [23], a free website consisting of stock videos. The model was implemented on 4-5 videos of high dimension video quality and a satisfactory result was obtained. The inferred videos with detection is stored in the Google drive with its link shared in the GitHub Repository. Furthermore, new videos can easily be uploaded in the .ipynb file and the inferred videos with the animals detected is automatically downloaded into the user's drive.

6 Application

The video detection of sea animals mentioned here is one of the n applications of automatic object detection. It can vary from surveillance to picture retrieval to improving the workspace efficiency etc. Some other peculiar applications of object detection are as follows:

6.1 Tracking of different objects

This application finds its particular use in surveillance in public transport systems, airports etc. It can also be used in modern games like Football, rugby etc., to track a specific player or the ball itself. A particular example in Football is whether to judge a player off-side or not in high stakes game, thus reducing the chances of error by a linesman.

6.2 Headcount of the People

Object detection provides a quick and efficient manner of finding the people and their count in a crowd. It can be an efficient method to control a mob of people. It can be instrumental in preventing stampede like situations.

6.3 Automated surveillance by CCTV

CCTVs are an efficient measure in maintaining security. However, they require personnel to monitor them continuously and also huge memory size for the recorded video. By using the detectors, we can automate the CCTVs to start recording at a particular time frame where a specific object is detected. It will reduce the personnel count to monitor the CCTV and reduce the need for bulky and expensive storage devices.

6.4 Vehicle Detection

Vehicle detection is another application where the detectors are used to detect the number plate of a speeding vehicle or a vehicle that has met with an accident. In the former case, it will help control the crime rates in the world of ever-increasing cars and in the latter case, it can be beneficial in saving the lives of the people in case of a fatal accident. The detectors can also replace the bulkier systems needed to control traffic and to trace the vehicles violating speed limits.

7 Conclusion

This paper introduces the problem of Object detection and tries to evaluate the state-of-the-art architecture EfficientDet on Aquarium Combined Dataset. After training the model on the mentioned Dataset, the idea is extended to the implementation of EfficientDet on object detection on videos and real-time object detection problem using the OpenCV library. A few videos of the sea animals are obtained from the stock videos present in the internet and the model is evaluated. The future work in this direction can be the implementation of this model for action detection problems. Gupta et al. [24] proposes a method to answer visual based questions by combining Efficient-Det model and Natural Language processing methods. A similar approach can be used for implementing EfficientDet for action detection problems. Semantic segmentation is another domain where the EfficientDet architecture can be successfully applied. The detection head and the loss function of EfficientDet-D4 is replaced by segmentation head and loss while keep the backbone architecture and the scaling factors same. As a result, it's reasonable to conclude that the Google brain team created the most powerful and efficient object identification algorithm to date, as evidenced by the Google Github repository.

References

[1] Mingxing Tan, Ruoming Pang, and Quoc V Le. Efficientdet: Scalable and efficient object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10781–10790, 2020.

[2] Common objects in context.

[3] Roboflow. Aquarium object detection dataset, Apr 2021.

[4] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001*, volume 1, pages I–I. Ieee, 2001.

[5] Paul Viola and Michael J Jones. Robust real-time face detection. *International journal of computer vision*, 57(2):137–154, 2004.

[6] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, volume 1, pages 886–893. Ieee, 2005.

[7] Pedro Felzenszwalb, David McAllester, and Deva Ramanan. A discriminatively trained, multiscale, deformable part model. In *2008 IEEE conference on computer vision and pattern recognition*, pages 1–8. Ieee, 2008.

[8] Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*, 2013.

[9] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Region-based convolutional networks for accurate object detection and segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 38(1):142–158, 2015.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 37(9):1904–1916, 2015.

[11] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[12] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28:91–99, 2015.

[13] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[14] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[15] 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*, pages 2117–2125, 2017.

[16] 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*, pages 2980–2988, 2017.

[17] Barret Zoph, Ekin D Cubuk, Golnaz Ghiasi, Tsung-Yi Lin, Jonathon Shlens, and Quoc V Le. Learning data augmentation strategies for object detection. In *European Conference on Computer Vision*, pages 566–583. Springer, 2020.

[18] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. Path aggregation network for instance segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8759–8768, 2018.

[19] Pengfei Shi, Xiwang Xu, Jianjun Ni, Yuanxue Xin, Weisheng Huang, and Song Han. Underwater biological detection algorithm based on improved faster-rcnn. *Water*, 13(17):2420, 2021.

[20] Roboflow-Ai. Roboflow-ai: A one-stop repository for low-code easily-installable object detection pipelines.

[21] Pytorch.

[22] Opencv.

[23] Pexels.

[24] Rahul Gupta, Parikshit Hooda, Nikhil Kumar Chikkara, et al. Natural language processing based visual question answering efficient: an efficientdet approach. In *2020 4th International Conference on Intelligent Computing and Control Systems (ICICCS)*, pages 900–904. IEEE, 2020.