University of Science and Technology of Hanoi

INTRODUCTION TO DEEP LEARNING

Group 44 - Midterm Project Report

EfficientDet for object detection in medical images

Lecturer: Dr. Nghiem Thi Phuong

Authors: Pham Ngoc Minh Chau - 22BI13063
Do Thi Huong Tra - BA12-174
Luu Linh Ly - 22BI13269
Bui Nguyen Ngoc Huyen - 22BI13199
Le Viet Hoang Lam - 22BI13235

Hanoi, VN, October , 2024

Table of contents

1. Introduction

EfficientDet, a scalable object detection architecture, has demonstrated impressive performance in many domains, including medical image analysis. We will study this model, focusing on its ability to efficiently detect objects of different sizes and scales.

A small cluster of cells that grows inside the body, which can be either pedunculated or sessile, is called polyp. Colonoscopy is the most common screening test for early-stage detection of colorectal cancer incidences and removal of polyps and adenomas, potentially reducing mortalities. Since effective polyp detection depends mostly on the gastroenterologist, it is reported that approximately 20% of polyp may be missed.

In this work, we propose an EfficientDet model along with the Pytorch framework for detecting polyps in colonoscopy images which are covered in Kvasir-SEG dataset. This proposed architecture aims to support and improve the detection performance compared to the baseline methods.

Background Information

EfficientDet models use EfficientNet as backbone, weighted bi-directional feature pyramid network (BiFPN) for multiscale feature fusion, and shared class/box prediction networks for bounding box classification and regression. The compound scaling method of the model scales the resolution, depth, and width for backbone, feature network, and box/class prediction networks at the same time. EfficientDet model effectively predicts the object classes and their associated bounding boxes, streamlining the object detection process.

1.1 BiFPN

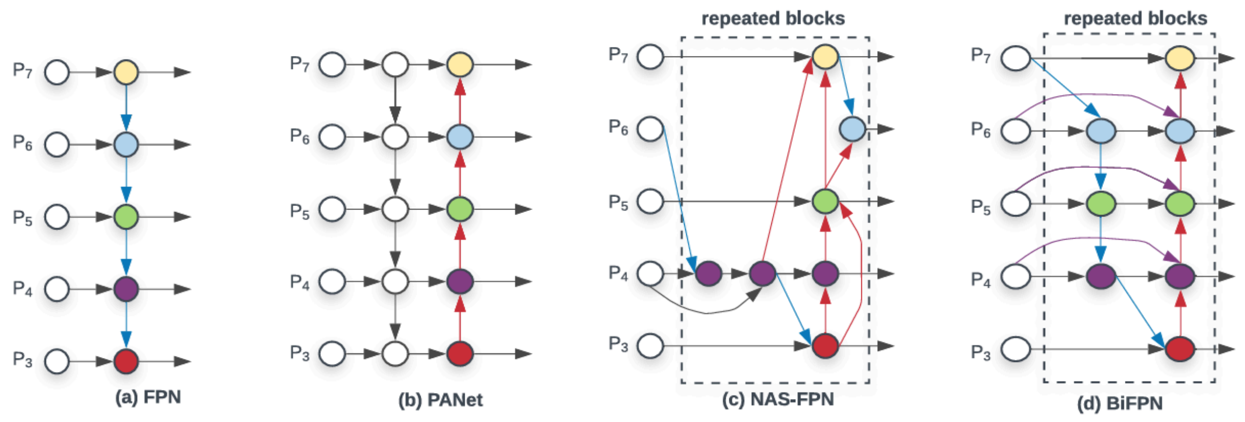


Figure 1 : ***Feature network design***

The BiFPN (Bidirectional Feature Pyramid Network) is a more efficient and flexible feature fusion approach compared to traditional feature pyramids. It ensures that features from different levels are effectively combined for object detection, leading to improved performance. BiFPN introduces top-down connections and cross-scale connections, allowing for more efficient information flow between feature levels [1].

1.2 Compound Scaling

Compound scaling is a technique introduced in the EfficientNet paper that involves scaling the depth, width, and resolution of a neural network in a balanced way to improve its performance. There are three scaling dimensions : depth, width, and resolution.

Depth Scaling: Increase the depth of the network by adding more layers. Deeper networks can capture richer and more complex features.
Width Scaling: Increase the width of the network by adding more channels to the layers. A wider network can learn more detailed features.
Resolution Scaling: Increase the resolution of the input images. Higher resolution images provide more detailed information for the network to learn from.

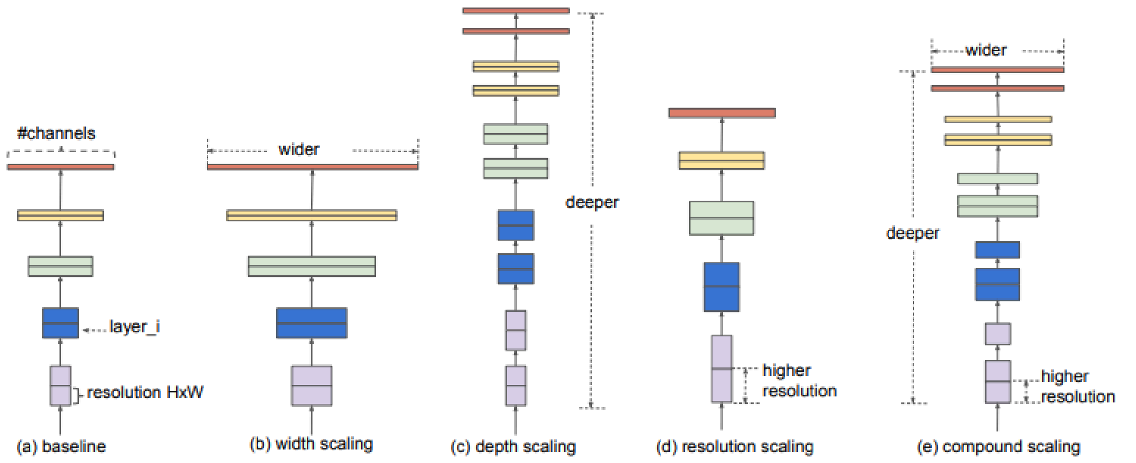


*Figure 2: **Model Scaling.** (a) is a baseline network example; (b)-(d) are conventional scaling that only increases one dimension of network width, depth, or resolution. (e) is our proposed compound scaling method that uniformly scales all three dimensions with a fixed ratio*

$$f = \alpha \,.\, \beta^{\varphi} .\, \gamma^{\varphi} \Rightarrow f = d \,.\, w^{\varphi} .\, r^{\varphi}$$

α = 1.2, β = 1.1, γ = 1.15	
$f = 1,2 \,.\, 1,1^{0} .\, 1,15^{0}$	Efficient-D0
$f = 1,2 \,.\, 1,1^{3} .\, 1,15^{3}$	Efficient-D3
$f = 1,2 \,.\, 1,1^{6} .\, 1,15^{6}$	Efficient-D6

α : depth scaling factor

β : width scaling factor

γ : resolution scaling factor

f : network scaling factor

φ : scaling coefficient

1.3 EfficientNet backbone

EfficientNet is a family of neural networks designed using neural architecture search (NAS), resulting in architectures that are more efficient than previous designs. EfficientDet leverages the EfficientNet architecture as its backbone, which is known for its high accuracy-to-parameter ratio. It reuses the same width/depth scaling coefficients of EfficientNet-B0 to B6 such that we can easily reuse their ImageNet-pretrained checkpoints.

1.4 EfficientDet architecture

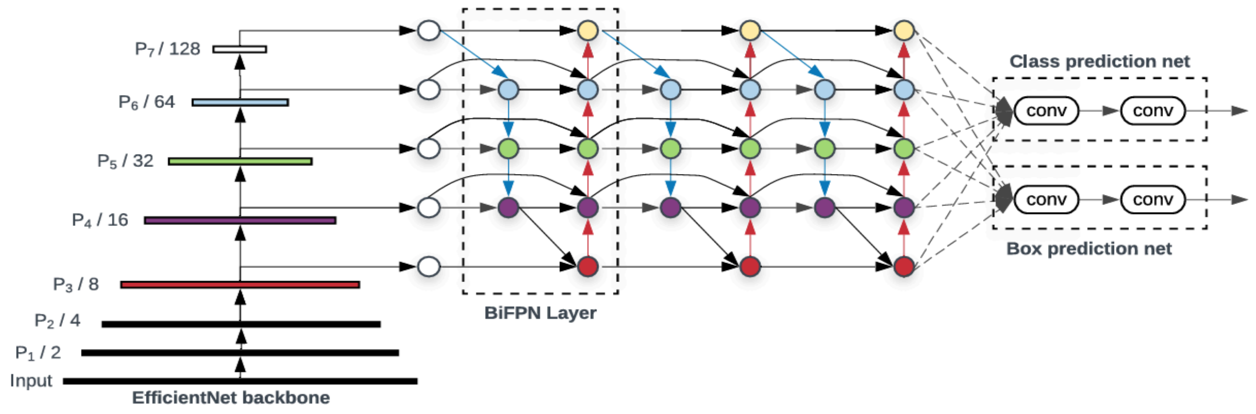


Figure 3: ***EfficientDet architecture***

The initial input image undergoes processing within the backbone network, extracting hierarchical features across multiple scales crucial for object detection. These extracted features are subsequently channeled through the BiFPN to bolster feature fusion and information exchange among different pyramid levels, refining features adept at handling objects of diverse scales. The refined features from the BiFPN then advance to the Class Prediction Network, responsible for forecasting the class identities and bounding box positions of prospective objects within the image.

2. Datasets

We perform experiments on the most popular dataset for polyp segmentation: “Kvasir-SEG”, which contains 1000 polyp images and their corresponding ground truth. The resolution of the images varies between 332 x 487 and 1920 x 1072. This open-access dataset is readily available for research and educational use, offering a valuable resource for various applications.

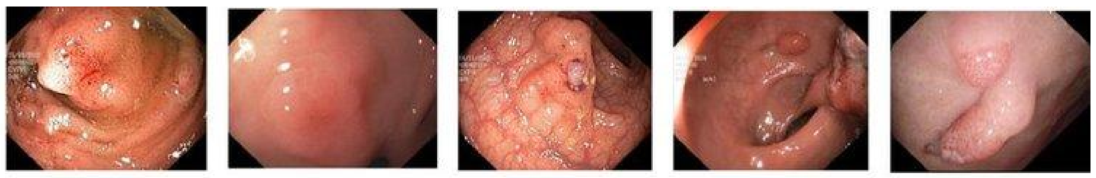
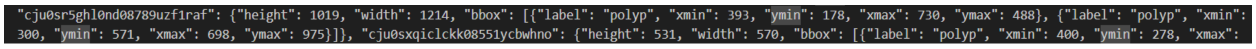


Figure 4: ***Kvasir-SEG dataset***

Most object detection models rely on datasets that provide annotations in a structured format, including bounding boxes and class labels. The Microsoft Common Objects in Context (COCO) format is widely used due to its organized structure, particularly its use of JSON files containing information about images, annotations, and object categories. For example, in this experiment, we have JSON data that includes image dimensions and bounding box coordinates for object detection of polyps. Each image entry provides the height, width, and bounding boxes for the annotated objects. These bounding boxes specify the label, in this case, "polyp," along with the coordinates for each bounding box, defined by the xmin, ymin, xmax, and ymax values. In this experiment, we have two JSON files corresponding to the *training* and *validation* sets.



```
"cju0sr5ghl0nd08789uzf1raf": {"height": 1019, "width": 1214, "bbox": [{"label": "polyp", "xmin": 393, "ymin": 178, "xmax": 730, "ymax": 488}, {"label": "polyp", "xmin":
300, "ymin": 571, "xmax": 698, "ymax": 975}]}, "cju0sxqiclckk08551ycbwhno": {"height": 531, "width": 570, "bbox": [{"label": "polyp", "xmin": 400, "ymin": 278, "xmax":
```

3. Experiments

3.1 Data Preprocessing

We first split the dataset into training, validation, and test sets as 66%, 14%, and 20%, respectively. For the annotation file (JSON file) containing detailed information of categories and images, we also separated into instances_train.json and instances_val.json corresponding to training and validation sets. To ensure the dataset is compatible with our framework and the object detection domain, all annotation files are converted into COCO format.

To enhance variance and improve generalization performance, the following data augmentation techniques have been applied: scale jittering (0.2, 2.0), horizontal flipping, and rotation (0°-360°). Images are resized to the EfficientDet-D0 input size (512x512) while preserving their original aspect ratio. Any necessary padding is added symmetrically to create square inputs, ensuring that the model requirements are met. Additionally, images are normalized using a specified mean and standard deviation to standardize pixel intensities, which aids in model convergence and performance. Metadata about these transformations is stored for accurate post-processing of the predictions.

3.2 Hyperparameter Configuration

The training code reads parameters from a YAML configuration file, which facilitates effective management of hyperparameters and settings.

YAML File: Contains manually set project details, including the names of the training and validation datasets. The hardware configuration specifies the use of 2 GPUs for training. Image normalization parameters, including mean and standard deviation, are carefully tuned to adapt to our specific dataset. Additionally, the file lists the object classes corresponding to the labels present in the dataset, which in this case includes only the class: ['polyp'].

EfficientDet-D0 Weights File: This file, named efficientdet-d0.pth, stores the pre-trained weights for EfficientDet-D0. These weights are downloaded directly from the GitHub repository of Yet-Another-EfficientDet-Pytorch, which is maintained by the community and provides reliable, up-to-date model weight s.

3.3 Training Process

The model is optimized using the Adam optimizer with *learning rate scheduling*, reducing the rate by 0.2 if validation loss did not improve for 10 epochs [2]. Early stopping was applied, halting training after 25 epochs without improvement in validation loss. Moreover, it is defined within a custom class, ModelWithLoss, which efficiently integrates loss calculation directly into the forward pass, streamlining gradient computations.

For evaluation, the COCO API is utilized to compute performance metrics, providing a standardized method for assessing object detection tasks. Additionally, integration with Weights and Biases (WandB) enhances the experiment tracking process, allowing for improved visualization and analysis of results.

To evaluate the model's performance on the validation set, metrics such as precision, recall, F1 score, and mean average precision (mAP) were employed, offering valuable insights into its object detection and classification capabilities. Ultimately, the best-performing model from the validation phase was identified based on the highest F1 score, and its parameters were fine-tuned for optimal performance prior to deployment.

NVIDIA Tesla K80 GPUs are utilized for training and testing this model on the remote Linux server (ICT6).

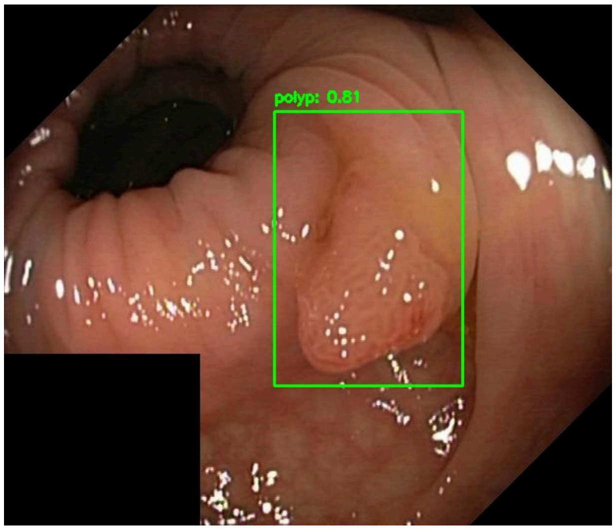
3.4 Training Configuration

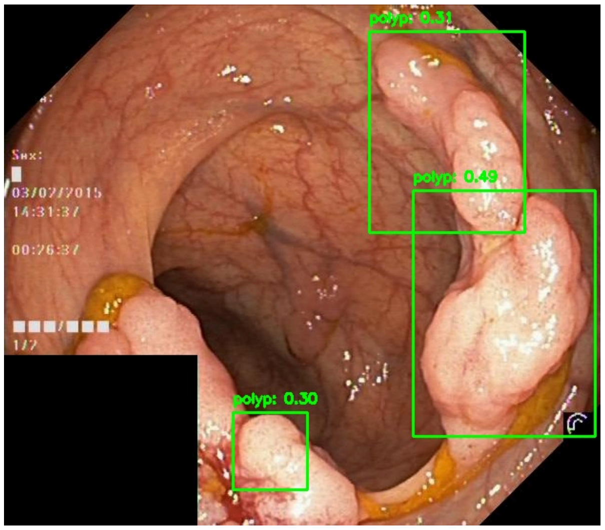
- Epochs: 100 epochs were used for training to ensure the model adequately learns the patterns in the data.
- Batch Size: A batch size of 10 was chosen based on hardware constraints and to ensure stable training.
- Learning Rate: Initial learning rate was set to 0.005, with reduction strategy based on validation performance according to learning rate scheduling as mentioned above.
- Early Stopping Patience: Early stopping was applied with a patience of 25 epochs to prevent overfitting.

4. Results

4.1 Inference on test set

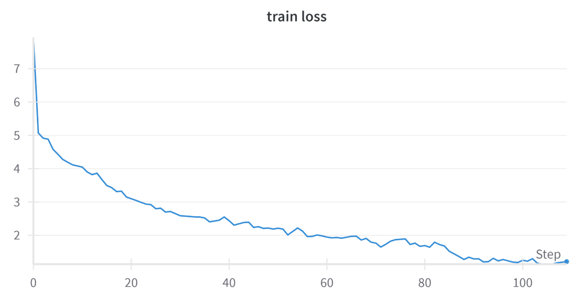
On the test set, we obtained confidence scores for each image, where each detection corresponds to a bounding box. These scores, typically between 0 and 1, indicate the model's confidence that the detected object belongs to the 'polyp' class. This process also generated a JSON file containing prediction information in COCO format.



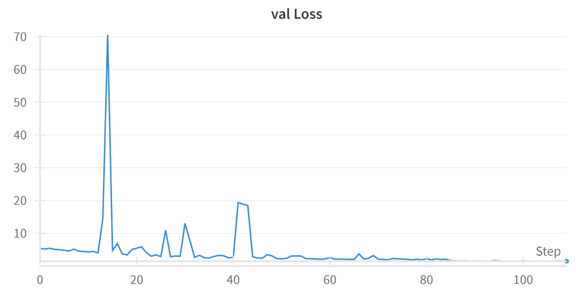


4.2 Evaluation

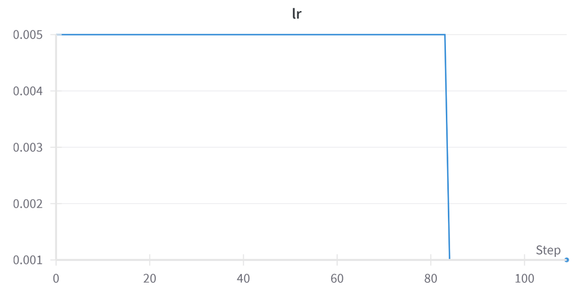
The training loss steadily decreased over 100 epochs, starting above 7 and reaching around 1, indicating effective optimization.



In contrast, the validation loss exhibited fluctuations in the early epochs, particularly around epoch 10, but stabilized and decreased after epoch 30, suggesting improved generalization.



The learning rate remained at 0.005 for the first 80 epochs before sharply dropping due to the *learning rate scheduler* as mentioned above.



See more on:
https://wandb.ai/chaupnm-22bi13063/kvasir2024/runs/4chx466j/workspace?nw=nwuserchaupnm22bi13063

5. Discussion

In order to enhance the accuracy of object detection on the Kvasir SEG dataset, we consider to implement an ensemble approach more with 4 EfficientDet models (D0-D3). The training and evaluation process will be repeated four times with one fold serving as the validation set once while the remaining three folds constitute the training set. This cross-validation approach helps us prevent overfitting and optimize hyperparameters. By training these models with diverse initializations and possibly varying hyperparameters, the ensemble can leverage the collective intelligence of the individual models to improve overall performance [3].

In conclusion, the initial implementation of the EfficientDet D0 model for object detection on the Kvasir SEG dataset marks a foundational step in the development process. By utilizing EfficientDet with PyTorch, we have successfully trained and tested the model on the dataset. The results obtained from running the EfficientDet D0 model provide a baseline for further enhancements and optimizations in pursuit of improved object detection accuracy on the Kvasir SEG dataset. Moving forward, the exploration of advanced techniques such as model ensembling, fine-tuning, and hyperparameter tuning holds promise for elevating the performance and robustness of the object detection system on this specific dataset. The objective is to explore the efficiency and scalability of EfficientDet in a medical setting, where accurate detection of polyps is crucial for early cancer diagnosis.

6. Appendix

6.1. Source Code

https://github.com/hheulwen/EfficientDet-Polyp/tree/main

6.2. References

[1] https://arxiv.org/pdf/1911.09070
[2] Adam: A Method for Stochastic Optimization (arxiv.org)
[3] https://ceur-ws.org/Vol-2886/paper9.pdf
[4]
https://github.com/zylo117/Yet-Another-EfficientDet-Pytorch/tree/master?fbclid=IwY2xjawFnsqBleHRuA2FlbQIxMAABHQCr6nn4TPEK4wMmIfVNgPzA4qnJmzsYGYDnoh1A91mXBMkkOGidri7ghw_aem_aRE7ruAI5QGCMwSQcB5ACg
[5] EfficientDet_d0_weights