# Object Detection System Based on Early-Exit Dynamic Perceiver

Woosung Kang Jiyong Jung Namwoo Kim Jungwan Lee Changyu Lee
KAIST
{raban1234, wjdwldyd1623, skadn4347, milelion, lck3365}@kaist.ac.kr

## Abstract

*In this paper, we address the excessive computational cost associated with traditional object detection models by proposing an innovative object detection model incorporating the early-exit strategy of the Dynamic Perceiver (Dyn-Perceiver). Utilizing RetinaNet as the base framework, we leverage the architecture of Dyn-Perceiver to implement early-exit mechanism, effectively reducing computational requirements without sacrificing detection accuracy. We address the challenge of incomplete feature maps caused by early-exit by exploring two feature map processing methods: zero-filling and downsampling. Additionally, we introduce a novel loss function tailored for early-exit classifiers to enhance the training process. Extensive experiments conducted on MS COCO and ILSVRC datasets demonstrate that our proposed model reduces computational cost while maintaining competitive mAP scores compared to baseline model. This research highlights incorporating early-exit into the object detection task enhances cost efficiency while ensuring its performance. Code is available at* https://github.com/cs470-team10/DynPerceiverBasedDetection.

## 1. Introduction

Since the ImageNet Challenge [12], the field of computer vision has experienced rapid growth [32]. To achieve increasingly better performance, the number of parameters and the size of models have progressively increased. However, these large models require substantial amounts of computation cost for inference. This high computational cost poses a significant limitation for edge computing applications, such as mobile smartphones and autonomous vehicles [4, 28]. To address the high computational cost during inference, early-exit strategies have been proposed. Early-exit reduces computing costs by skipping deep-layer computations for simple images. Early-exit research has been conducted in computer vision [29, 31].

Figure 1 illustrates an example of early-exit for "hard" and "easy" [17, 23], images in object detection tasks. For

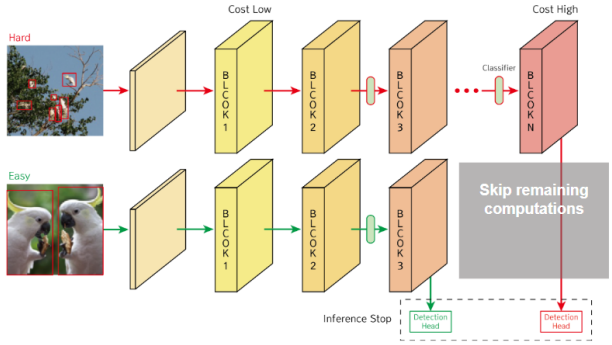


Figure 1. Example of early-exit for hard and easy images.

"hard" sample images, where making a decision is difficult, it is necessary to pass through all deep layers to obtain the feature map. However, for "easy" sample images, where making a decision is easier, it is not necessary to go through all deep layers, and the feature map can be extracted earlier. By using early-exit, the computational cost for "easy" sample images can be reduced, thereby lowering the average computational cost. Among these models, we focused on the Dynamic Perceiver (Dyn-Perceiver) [7]. The Dyn-Perceiver framework for image classification demonstrates superior performance compared to other existing early-exit models for two reasons. First, it explicitly separates the classification branch and the feature branch to prevent performance degradation caused by the early-exit classifier. Second, it enhances the quality of the feature map and the performance of the classifier by introducing latent codes and self-attention mechanisms. We propose a model that implements early-exit in object detection, anticipating that the advantages of the Dyn-Perceiver will also be beneficial in the context of object detection.

In this paper, we propose an object detection model with an early-exit method to improve upon the excessive computational cost of existing object detection models. We use RetinaNet [16] as the object detection framework and implement early-exit using the Dyn-Perceiver [7] structure to leverage its advantages. When processing input images, feature maps need to be extracted across multiple layers;

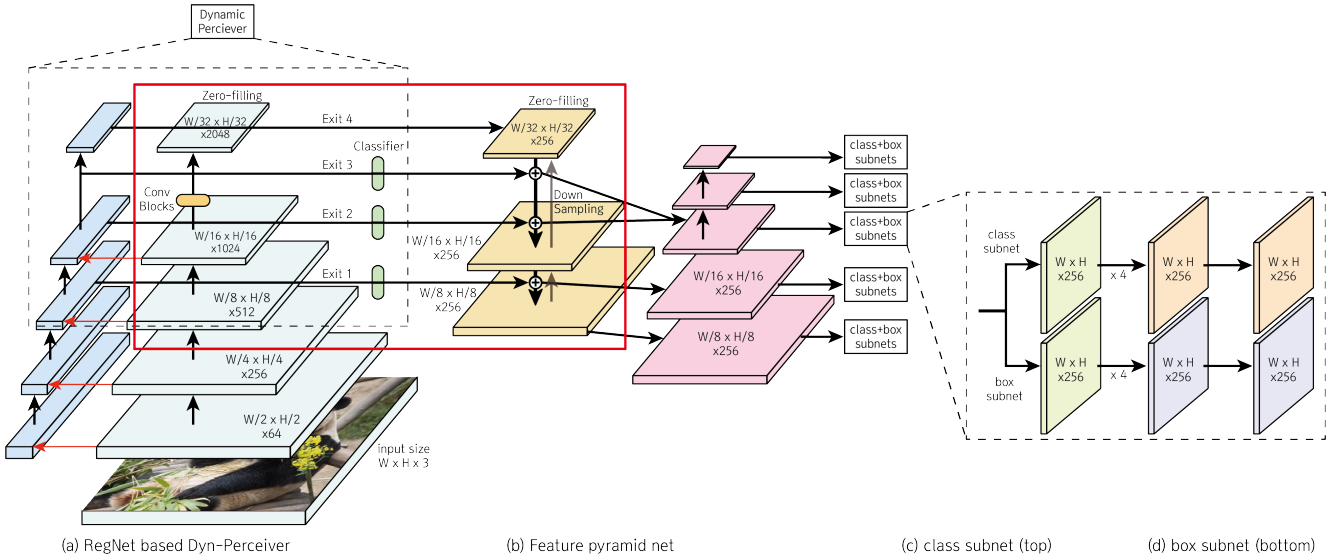


Figure 2. Overall procedure of feature map process between backbone model and Feature Pyramid Network

however, early-exit may cause some layers to be skipped. To compensate for the missing information in these situations, we explore two feature map processing methods: zero-filling and downsampling. Additionally, our proposed approach integrates early-exit classifiers within the architecture, and to manage the impact on model training performance, we define and utilize a new loss function. In our experiments, we compare the performance of the proposed method with baseline models that do not include early-exit, using the MS COCO [15] and ILSVRC [13] datasets. Our results demonstrate that the proposed method can reduce computational cost without degrading object detection performance.

- We have developed an object detection model that supports the early-exit algorithm of the Dyn-Perceiver.
- We proposed novel loss function to train early-exit object detection model and two approaches to compensate for losing high level feature map.
- Our model, with fewer FLOPs, achieved an mAP comparable to models that do not support the Dyn-Perceiver's early-exit. Additionally, when considering a similar level of FLOPs, our model's mAP improved compared to models without the early-exit support.

## 2. Related works

There are many researches for reducing computational cost in image classification tasks and object detection tasks. Recent trends include knowledge distillation [9, 18, 20], pruning [6, 8, 25, 30], EfficientDet [11, 24], YOLO [14, 26, 27], and early-exit [1, 3, 5, 10, 19, 22, 29, 31]. Knowledge distillation [9] transfers knowledge from a large teacher model to a smaller student model, enabling the smaller model to achieve similar performance. Pruning [6] removes unimportant neurons or weights, reducing the model size and computational cost. EfficientDet [24] utilizes EfficientNet as a backbone and optimizes scaling strategies to maintain high performance while reducing computational cost. YOLO [27] employs a single neural network for object detection with a lightweight structure and various optimization techniques to reduce computational cost. Early-exit [7] allows models to bypass deep-layer computations for simpler images, thereby reducing computational costs and improving efficiency.

## 3. Methods

In this section, we propose specific mechanisms that are related to feature maps for supporting early-exit in object detection tasks. We also introduce a new training objective function for training an early-exit object detection model. By merging those methods in subsections to RegNet-based Dyn-Perceiver [7] and FPN (Feature Pyramid Network), our proposed model can be constructed which is in Figure 2.

A feature map is a vector of values that represents the responses of a particular kernel or filter applied to different regions of an input image. In the RetinaNet, feature maps are fed into FPN. To achieve early-exit during object detection, several modifications of the existing model should be followed. We discuss methodologies for extraction of feature map in Section 3.1 and processing steps of feature map through FPN in Section 3.2. In Section 3.3, we propose a new loss function because the original loss function of RetinaNet [16] doesn't consider the classifiers in Dyn-Perceiver.

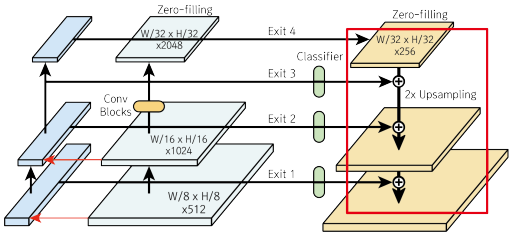


Figure 3. Overall procedure of feature map process between backbone model and Feature Pyramid Network

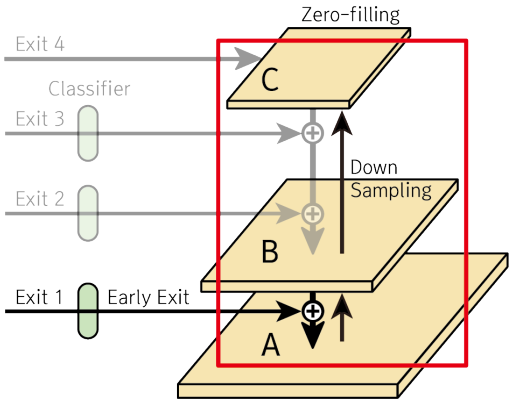


Figure 4. Overall process of downsampling with its direction

### 3.1. Feature map extraction

The Dyn-Perceiver comprises two distinct branches: the feature branch and the classification branch. To support early-exit functionality, intermediate feature maps utilized in object detection must be extracted during the forward pass of the backbone. In Dyn-Perceiver, there are multiple options for selecting intermediate feature maps, as the branches are interconnected through two symmetric cross-attention operations. We use the feature maps generated after the Z2X cross-attention as the intermediate feature maps for object detection. The Z2X cross-attention aggregates semantic information from the classification branch back into the feature branch, resulting in feature maps with richer information compared to those before cross-attention.

### 3.2. High-level feature map generation

To implement early-exit on the RetinaNet based on Dyn-Perceiver, high-level feature maps, which serve as inputs to the FPN are essential. Since the Dyn-Perceiver cannot produce high-level feature maps after early-exit occurs, a new approach is needed to provide these maps. To maintain the computational benefits of early-exit, a method for constructing high-level feature maps with low complexity is required.

**Approach 1. Zero-filling**. We set the high-level feature maps as fixed-size tensors with zero elements. When early-exit occurs, the intermediate feature map already contains sufficient information to classify objects in the input image. Thus, we use zero-filled feature maps for the higher-level feature maps. This straightforward method is solely for enabling early-exit functionality.

**Approach 2. Downsampling**. The FPN processes feature maps from the backbone model to feed into the detection header. As depicted in Figure 3, the backbone model provides feature maps which are then upsampled before being relayed to the bounding box header. However, in the case of an early-exit, the subsequent feature maps are not produced. Typically, this would result in four out of five input data streams to the bounding box header being zeros, thus rendering them useless. While for the Dyn-Perceiver, dealing with a classification task, these subsequent feature maps were unnecessary, our task necessitates more substantial data for object detection, thus requiring inputs for the FPN. To resolve this issue, we adopted a downsampling technique.

The process illustrated in Figure 4, shows how the high-level feature maps are generated by downsampling the feature maps obtained from the forward branches to replace the absent maps. By denoting the feature maps obtainable at each early-exit stage 1, 2, and 3 levels as feature map A, feature map B, and feature map C, if the image exits at early-exit stage 1, we only acquire three instances of feature map A. Feature maps B and C are then derived by downsampling feature map A once and twice, respectively. If the image exits at early-exit stage 2, we obtain feature maps A and B, and feature map C is derived by downsampling feature map B obtained from early-exit stage 2. Using this data allows the standard convolution operations to be utilized, thereby securing more meaningful data compared to before implementing downsampling.

### 3.3. Loss function with early-exit

The original loss used in RetinaNet is as follows:

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

Equation 1 is the loss function $L$ proposed in [16], which calculates the sum of classification loss $L_{cls}$ and bounding box regression loss $L_{bbox}$ of the detection head. And the $\alpha$ creates emphasis between detecting objects (classification) versus precisely localizing them (regression). This method was optimized for the object detection framework, updating weights of the RetinaNet along with weights of the latter layers of the backbone model. Since this training object does not account for classifiers of the backbone model, it could be inappropriate to achieve early-exit in the object detection system. Our early-exit mechanism requires a classification score for each stage of the backbone model and performs early-exit if the classification score exceeds the

| Category | COCO (MS) | COCO-Single |
|---|---|---|
| Total Images | About 330,000 | 14,486 |
| Number of Classes | 80 | |
| Year Started | 2014 | |

Table 1. COCO dataset

| Category | ILSVRC | ILSVRC-Single |
|---|---|---|
| Total Images | About 560,000 | 20,192 |
| Number of Classes | 200 | |
| Year Started | 2014 | |

Table 2. ILSVRC dataset

threshold of each stage which is pre-defined in the framework of Dyn-Perceiver [7].

$$L' = \lambda(L_{cls} + L_{bbox}) + \theta L_{Dyn-Perceiver} \quad (2)$$

To address this problem, we propose a simple training objective function $L'$ to make the classifier of the Dyn-Perceiver suitable for the task of early exit in the object detection system shown in Equation 2 with $\alpha$ value 1. By incorporating a cross-entropy loss of classifier in Dyn-Perceiver $L_{Dyn-Perceiver}$ to Equation 1 which is actively used for fine-tuning object detection system. To regulate the impact of each component of the training objective function, $\lambda$ and $\theta$ factors were introduced.

# 4. Experiments

In this section, we design comparative experiments to evaluate the precision and computational cost of our proposed model. Furthermore, the various experiments are conducted as ablation studies to analyze the contributions of the methods proposed in Section 3.

## 4.1. Experimental setup

### 4.1.1 Dataset preparation

We use MS COCO[15] and ILSVRC[21] dataset for fine-tuning and evaluation of our proposed model. Due to the limitation of GPU and server storage, we were not able to utilize the whole MS COCO and ILSVRC dataset for fine-tuning in a reasonable time. As so, we sub-sampled single object images from MS COCO and ILSVRC by filtering images through each annotation file. We define each sub-dataset as 'COCO-Single' and 'ILSVRC-Single', respectively. The number of images before and after filtering along with the number of classes can be found in Table 1 and Table 2.

### 4.1.2 Evaluation metrics

For all experiments, we report the mAP (mean Average Precision) score and FLOPs (Floating Point Operations Per Second). The mAP is a standard metric used to evaluate the performance of object detection models. It is a crucial metric to demonstrate our proposed model's accuracy. FLOPs refer to the number of floating-point operations required to perform inference with a model. It is a metric used to measure the computational complexity and performance of an object detection model.

### 4.1.3 Threshold

We calculate threshold values for the condition of early-exit. Following the pre-determined proportion of samples terminated at each early-exit stage in Figure 9, we generate 34 threshold distributions. During the evaluation, early-exit is executed based on these thresholds. The threshold value is compared with the maximum value of the softmax probability calculated for each sample by the classifier in Dyn-Perceiver. If this value exceeds the threshold, the forward propagation terminates, avoiding the execution of deeper layers in Dyn-Perceiver.

### 4.1.4 System environment

Our experiments were conducted on a server running Ubuntu 22.04.4 LTS with the GNU/Linux 5.15.0-67-generic x86_64 kernel. The server is equipped with an Intel Xeon Processor (Skylake, IBRS) and an NVIDIA GeForce RTX 3090 GPU. The software environment for our code includes Python 3.10.2, CUDA 11.6, PyTorch 1.12.1, and Torchvision 0.13.1. These specifications ensure reproduction and compatibility for the deep learning models and computations involved in our research.

## 4.2. Model configuration

### 4.2.1 Baseline

We use the RegNet-800MF-based Dyn-Perceiver for the backbone model pre-trained on ImageNet [21], and use RetinaNet for the object detection system with the FPN. ImageNet pre-trained RegNet-800MF was only available and the other RegNet models were not reproducible due to hardware limitations. We select the baseline model by combining the backbone model with RetinaNet and finetune the ImageNet-pretrained checkpoints on COCO-Single and ILSVRC-Single for 12 epochs following the official settings of RetinaNet [16] in the MMDetection code base [2]. We follow the training objective function as Equation 1. Details of RegNet-800MF-based Dyn-Perceiver model configuration are listed in Appendix A.

#### 4.2.2 Proposed model

We train the proposed model to achieve early-exit and compare precision with the baseline model. The backbone of our proposed model is a RegNet-800MF-based DynPerceiver the same as the baseline model. Our proposed model is fine-tuned from the ImageNet-pretrained checkpoints on COCO-Single and ILSVRC-Single for 12 epochs. The main differences between the baseline model and our proposed model are that using the downsampling method for generating high-level feature maps when early-exit occurs and adapting our loss (Equation 2) for training.

### 4.3. Evaluation for the cost efficiency

We conduct an experiment on the proposed model's efficiency for computational costs compared to the baseline model without the early-exit method. Configurations in our model vary in two methods of the feature map generation (i.e., zero-filling and downsampling) and two hyperparameters of the loss function (i.e., $\theta$ and $\lambda$). According to an empirical study with the different configurations, we investigate a specific model with the most significant improvement in cost efficiency.

From the perspective of feature map generation, models using the downsampling method commonly show better results than ones with the zero-filling. In addition, as a pilot experiment on hyperparameters that required training individual models, we attempted to compare evaluation results based on several $\theta$, $\lambda$ pairs such as 1, 1, 0.5, 1, and 0.5, 0.5. Although we had seen that decreasing values of $\lambda$ made the models' performance lower, we need to do more precise experiments regarding different $\theta$ values. Experiments are designed by using configurations consisting of one of $\theta$ candidates from 0.1, 0.05, 0.012, 0.0115, 0.011, 0.01, 0.001 and $\lambda$ as 1. Among all the experimental results of the configurations, we found out the best combination of $\theta$ and $\lambda$ is 0.0115 and 1, respectively. Therefore, we focus on this best configuration to describe the detailed explanation of our experimental results in Section 5.

### 4.4. Ablation studies

#### 4.4.1 Evaluation of early-exit

To verify the effectiveness of our proposed model, we analyze the mAP at each threshold of 3 cases; baseline model, proposed model with random-exit, and proposed model. The baseline model is trained with Equation 1 and the proposed model is trained with Equation 2. Random-exit triggers the early exit of images randomly at each stage with a pre-determined proportion and calculates averages of mAP 10 times. By comparing the mAP versus FLOPs of the baseline model and proposed model, the effectiveness of our training objective function (see Equation 2) which targets the classifier to be trained on the object detection dataset can be evaluated qualitatively. Also, by comparing the mAP versus FLOPs of the proposed model and the proposed model with random-exit, the validity of thresholds at each stage of the backbone model can be evaluated.

| Model | Threshold Index | COCO-Single mAP (%) | COCO-Single FLOPs (G) | ILSVRC-Single mAP (%) | ILSVRC-Single FLOPs (G) |
|---|---|---|---|---|---|
| Baseline | - | 39.3 | 154.54 | 39.9 | 140.87 |
| Ours | 30 | 38.2 | 153.45 | 38.3 | 140.62 |
| | 31 | 39.7 | 153.74 | 38.6 | 140.67 |
| | 32 | 40.2 | 153.87 | 39.1 | 140.74 |
| | 33 | 40.8 | 154.05 | 39.6 | 140.76 |
| | 34 | 43.1 | 154.54 | 40.3 | 140.87 |

Table 3. Evaluation result of our proposed model ($\theta$=0.0115, $\lambda$=1)

#### 4.4.2 Comparing two high-level feature map generation methods

Through this experiment, we investigate two high-level feature map processing methods in the proposed model; zero-filling and downsampling, in terms of computational cost and mAP. As so, we designed two different models that adopt two generation methods of high-level feature maps. We observed computational cost using the FLOPs metric and evaluated the results of the two methods based on which one showed fewer FLOPs for the same mAP.

## 5. Results

This section shows and discusses the results of the experiments described in the Experiment section. And due to space constraints, we provide some details in the Appendix B.

### 5.1. Evaluation for the cost efficiency

The evaluation results of the baseline model and our proposed model after training are presented in Table 3. The mAP and FLOPs resulted on the validation set of COCO-Single and ILSVRC-Single datasets are shown in the table. These results are the output of evaluation when $\theta$=0.0115, $\lambda$=1. We report only the results for the last 5 threshold distributions to demonstrate the effectiveness of our model compared to the baseline model. As mentioned in Section 4.3, we conduct the experiments to find the optimal hyperparameters. The overall results of the optimization is in the Appendix.

The mAP and FLOPs decrease as the threshold index decreases because a lower threshold index means a higher number of samples terminate in the earlier stages of DynPerceiver. On the COCO-Single dataset, our model outperforms the baseline model's mAP value with fewer FLOPs when the threshold index is 31. The mAP metric increased by 3.8%, while the FLOPs were reduced by 0.8 GF. On

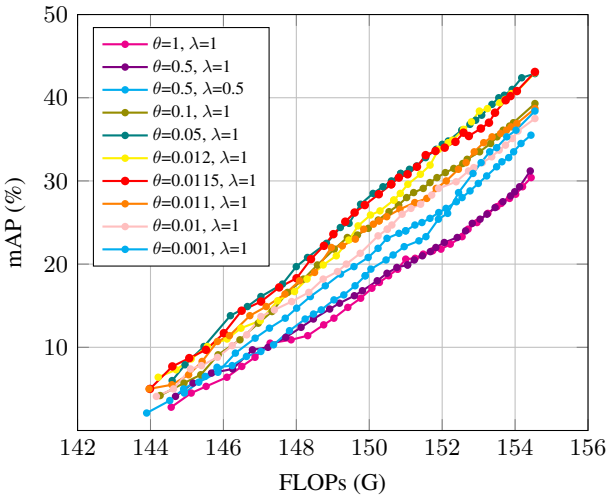


Figure 5. Result for various hyperparameters on COCO-Single

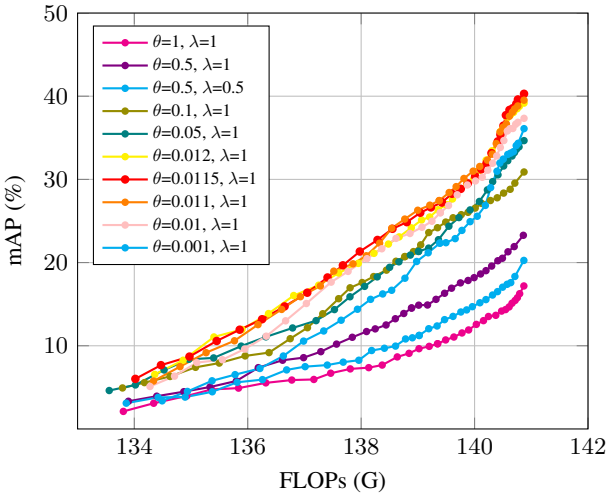


Figure 6. Result of various hyperparameters on ILSVRC-Single

the ILSVRC-Single dataset, our model achieves a higher mAP value than the baseline model while using the same amount of FLOPs. Thus, our model can reduce computational cost through early-exit while achieving a higher mAP value compared to the baseline model.

Figure 5 presents the evaluation results on COCO-Single across all configurations of our model. Each line represents a specific configuration and each point corresponds to a specific threshold. Similarly, Figure 6 displays the evaluation results on ILSVRC-Single. As the FLOPs decrease, the number of samples that early-exit increases, resulting in a reduction in the mAP value which supports the tendency of Figure 5 and 6. Also, we can observe that the mAP value of our model changes depending on the hyperparameters. The shape of results also differed by dataset; COCO-

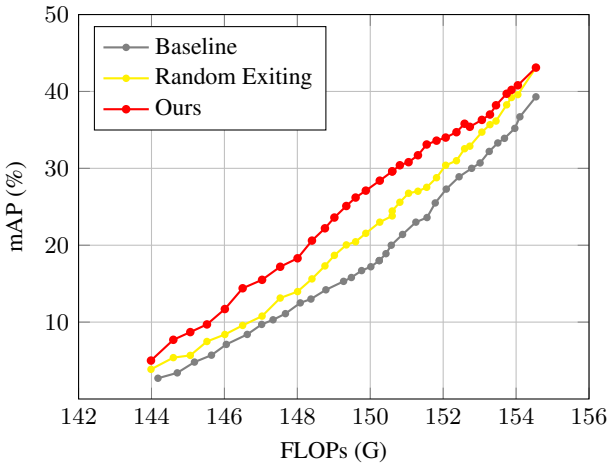


Figure 7. Result for evaluation

| Models | Baseline model | Proposed model |
|---|---|---|
| Accuracy (%) | 1.35 | 70.83 |

Table 4. Classification scores at last stage of backbone of baseline model and proposed model

Single showed linear shape in every range of FLOPS, and ILSVRC-Single showed concave shape at the high FLOPs while rest of part showed linear.

### 5.2. Evaluation of early exit

Figure 7 demonstrates the effectiveness of our model on COCO-Single dataset. Here, our model and baseline model adopt downsampling as high-level feature map generation method when early-exit. We used same threshold values for comparing our model with baseline model, to solely compare effects of our loss function (Equation 2) which includes classification loss in classifier of backbone model. Comparing our model to baseline model, we were able to find out our model showed higher mAP than the baseline model at every value of FLOPs.

Table 4 shows the classification accuracy of classifier at last stage of baseline model and our model. First, classifier trained with our loss function showed significant improvement in classification accuracy. As the classification accuracy of classifier increases, the mAP also increases. These results indicate that the classification scores have a high correlation with the performance of our model. It supports the fact that our model compares threshold with maximum value of softmax probability at each classifier. Thus, the $L_{Dyn-Perceiver}$ (see Equation 2) term in our loss function is essential to ensure high performance in object detection. Also, mAP of our model showed higher values than mAP of the our model with random-exit. Therefore, the process of

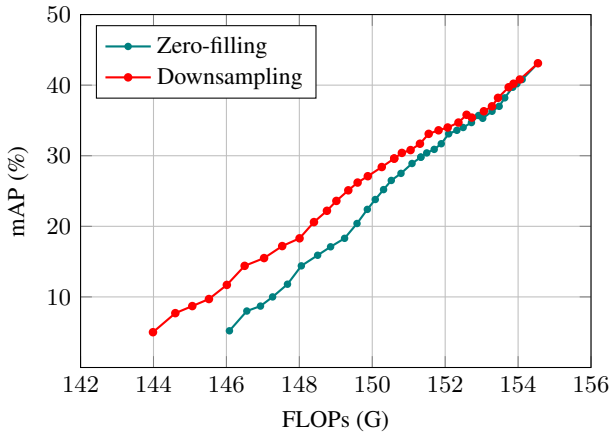


Figure 8. Results of two high-level feature map generating methods

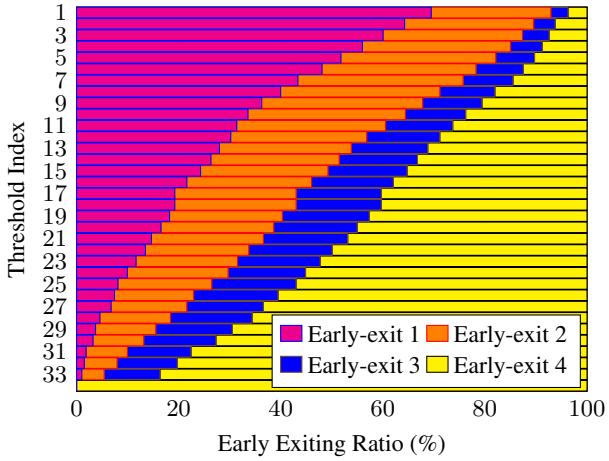


Figure 9. Predefined proportion of various thresholds

generating a threshold based on the classification score and comparing it to each classifier's maximum value of softmax probability is valid.

### 5.3. Comparing two high-level feature map generation methods

In Figure 8, the red line represents the evaluation results of our model with downsampling method at high-level feature map generation. The green line represents the evaluation results of our model with zero-filling at high-level feature map generation. Downsampling applied model exhibits a smaller decrease in mAP relative to the reduction in FLOPs compared to zero-filling applied model. Overall, the graph shows that for the same FLOPs, the model with downsampling achieves a higher mAP.

As the FLOPs decrease, the performance gap between the model with downsampling and the model with zero-filling widens. Lower FLOPs indicate that the stage at which early-exit occurs moves forward, increasing the frequency of feature map passing. The greater the frequency of feature map passing, the larger the mAP difference between the two models, indicating that downsampling is more effective than zero-filling and results in a more significant performance difference. Thus, we adopted downsampling among the two approaches.

### 5.4. Case study

Based on the results of early-exit, we conduct a qualitative analysis of easy and hard images. As shown in Figure 9, the proportion of images exiting at each early-exit classifier is determined by the threshold. Early-exit occurs when the image exceeds the threshold at exit1, exit2, or exit3. Therefore, we suppose that images early-exited can be classified as easy, while those that are not early-exited can be classified as hard. We examine the easy and hard images at a

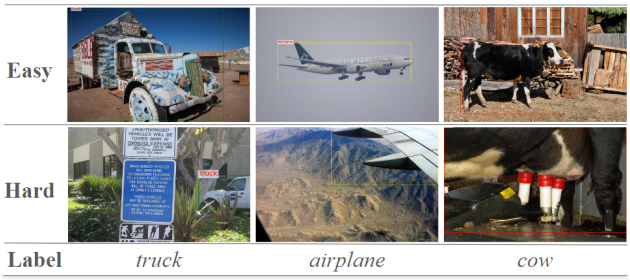


Figure 10. Easy and hard images for object detection

threshold value of 20, where a balanced distribution of easy and hard images is observed. Figure 10 shows examples of easy and hard image samples. Easy images typically feature a large object-to-image ratio or have clear object boundaries. In contrast, hard images often have a small object-to-image ratio, contain only part of the object, or have the object partially obscured by other objects. This observation aligns with the criteria humans use to distinguish objects, indicating some level of consistency in difficulty assessment.

## 6. Conclusions

This paper presents a novel object detection model that integrates the early-exit strategy of the Dynamic Perceiver (Dyn-Perceiver) with the RetinaNet architecture to alleviate the high computational costs of traditional models. By incorporating early-exit mechanisms, we reduce the computational cost while preserving detection accuracy. To mitigate the issue of incomplete feature maps due to early exits, we explore and implement zero-filling and downsampling techniques. Furthermore, a new loss function is introduced to optimize the training of early-exit classifiers. Experimental results on MS COCO and ILSVRC datasets validate that

our approach achieves substantial reductions in FLOPs with competitive mAP, underscoring the effectiveness of DynPerceiver's early-exit strategy in object detection.

## References


[1] Tolga Bolukbasi, Joseph Wang, Ofer Dekel, and Venkatesh Saligrama. Adaptive neural networks for efficient inference. In *International Conference on Machine Learning*, pages 527–536. PMLR, 2017. 2

[2] MMDetection Contributors. Mmdetection. https://github.com/open-mmlab/mmdetection, 2018. 4

[3] Michael Figurnov, Maxwell D Collins, Yukun Zhu, Li Zhang, Jonathan Huang, Dmitry Vetrov, and Ruslan Salakhutdinov. Spatially adaptive computation time for residual networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1039–1048, 2017. 2

[4] Abhinav Goel, Caleb Tung, Yung-Hsiang Lu, and George K Thiruvathukal. A survey of methods for low-power deep learning and computer vision. In *2020 IEEE 6th World Forum on Internet of Things (WF-IoT)*, pages 1–6. IEEE, 2020. 1

[5] Alex Graves. Adaptive computation time for recurrent neural networks. *arXiv preprint arXiv:1603.08983*, 2016. 2

[6] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015. 2

[7] Yizeng Han, Dongchen Han, Zeyu Liu, Yulin Wang, Xuran Pan, Yifan Pu, Chao Deng, Junlan Feng, Shiji Song, and Gao Huang. Dynamic perceiver for efficient visual recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5992–6002, 2023. 1, 2, 4

[8] Yang He, Ping Liu, Ziwei Wang, Zhilan Hu, and Yi Yang. Filter pruning via geometric median for deep convolutional neural networks acceleration. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4340–4349, 2019. 2

[9] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. 2

[10] Gao Huang, Danlu Chen, Tianhong Li, Felix Wu, Laurens Van Der Maaten, and Kilian Q Weinberger. Multi-scale dense networks for resource efficient image classification. *arXiv preprint arXiv:1703.09844*, 2017. 2

[11] Jiaqi Jia, Min Fu, Xuefeng Liu, and Bing Zheng. Underwater object detection based on improved efficientdet. *Remote Sensing*, 14(18):4487, 2022. 2

[12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012. 1

[13] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017. 2

[14] Chuyi Li, Lulu Li, Hongliang Jiang, Kaiheng Weng, Yifei Geng, Liang Li, Zaidan Ke, Qingyuan Li, Meng Cheng, Weiqiang Nie, et al. Yolov6: A single-stage object detection framework for industrial applications. *arXiv preprint arXiv:2209.02976*, 2022. 2

[15] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2015. 2, 4

[16] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection, 2018. 1, 2, 3, 4

[17] Zhihao Lin, Yongtao Wang, Jinhe Zhang, and Xiaojie Chu. Dynamicdet: A unified dynamic architecture for object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6282–6291, 2023. 1

[18] Yifan Liu, Changyong Shu, Jingdong Wang, and Chunhua Shen. Structured knowledge distillation for dense prediction. *IEEE transactions on pattern analysis and machine intelligence*, 45(6):7035–7049, 2020. 2

[19] Arne Moos. Efficient single object detection on image patches with early exit enhanced high-precision cnns. *arXiv preprint arXiv:2309.03530*, 2023. 2

[20] Zengyu Qiu, Xinzhu Ma, Kunlin Yang, Chunya Liu, Jun Hou, Shuai Yi, and Wanli Ouyang. Better teacher better student: Dynamic prior knowledge for knowledge distillation. *arXiv preprint arXiv:2206.06067*, 2022. 2

[21] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge, 2015. 4

[22] Amin Sabet, Jonathon Hare, Bashir Al-Hashimi, and Geoff V Merrett. Temporal early exits for efficient video object detection. *arXiv preprint arXiv:2106.11208*, 2021. 2

[23] Petru Soviany, Radu Tudor Ionescu, Paolo Rota, and Nicu Sebe. Curriculum self-paced learning for cross-domain object detection. *Computer Vision and Image Understanding*, 204:103166, 2021. 1

[24] 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

[25] Yehui Tang, Kai Han, Yunhe Wang, Chang Xu, Jianyuan Guo, Chao Xu, and Dacheng Tao. Patch slimming for efficient vision transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12165–12174, 2022. 2

[26] Juan Terven, Diana-Margarita Córdova-Esparza, and Julio-Alejandro Romero-González. A comprehensive review of yolo architectures in computer vision: From yolov1 to yolov8 and yolo-nas. *Machine Learning and Knowledge Extraction*, 5(4):1680–1716, 2023. 2

[27] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7464–7475, 2023. 2

[28] Yulin Wang, Yizeng Han, Chaofei Wang, Shiji Song, Qi Tian, and Gao Huang. Computation-efficient deep learning for computer vision: A survey. *Cybernetics and Intelligence*, 2024. 1

[29] Le Yang, Yizeng Han, Xi Chen, Shiji Song, Jifeng Dai, and Gao Huang. Resolution adaptive networks for efficient inference. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2369–2378, 2020. 1, 2

[30] Le Yang, Haojun Jiang, Ruojin Cai, Yulin Wang, Shiji Song, Gao Huang, and Qi Tian. Condensenet v2: Sparse feature reactivation for deep networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3569–3578, 2021. 2

[31] Le Yang, Ziwei Zheng, Jian Wang, Shiji Song, Gao Huang, and Fan Li. : An adaptive object detection system based on early-exit neural networks. *IEEE Transactions on Cognitive and Developmental Systems*, 2023. 1, 2

[32] Yong Zhou, Lei Xia, Jiaqi Zhao, Rui Yao, and Bing Liu. Efficient convolutional neural networks and network compression methods for object detection: a survey. *Multimedia Tools and Applications*, 83(4):10167–10209, 2024. 1

# Appendix

## A. Model Configuration

The Table 5 shows the detailed configuration of Dyn-Perceiver implemented on RegNet. And SA is the abbreviation of self-attention.

| Configuration | Value |
|---|---|
| RegNet size | 800M |
| token number L | 128 |
| # SA blocks | [3, 3, 9, 6] |
| SA widening factor | 4 |

Table 5. RegNet configurations

## B. Overall experimantal results

This section describes overall experimental results of the our model's hyperparameter configurations. Table 6 shows mAP and FLOPs for each hyperparameter $\theta$ and $\lambda$ for every threshold on COCO-Single dataset. And Table 7 shows the results for same hyperparameters on ILSVRC-Single dataset.

| Threshold index | $\theta$=1, $\lambda$=1 | | $\theta$=0.5, $\lambda$=1 | | $\theta$=0.5, $\lambda$=0.5 | | $\theta$=0.1, $\lambda$=1 | | $\theta$=0.05, $\lambda$=1 | | $\theta$=0.012, $\lambda$=1 | | $\theta$=0.0115, $\lambda$=1 | | $\theta$=0.011, $\lambda$=1 | | $\theta$=0.01, $\lambda$=1 | | $\theta$=0.001, $\lambda$=1 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | FLOPs (G) | mAP (%) | FLOPs (G) | mAP (%) | FLOPs (G) | mAP (%) | FLOPs (G) | mAP (%) | FLOPs (G) | mAP (%) | FLOPs (G) | mAP (%) | FLOPs (G) | mAP (%) | FLOPs (G) | mAP (%) | FLOPs (G) | mAP (%) | FLOPs (G) | mAP (%) |
| 1 | 144.568 | 2.8 | 144.686 | 4.1 | 144.927 | 4.5 | 144.264 | 4.2 | 144.594 | 6.0 | 144.211 | 6.4 | 143.986 | 5.0 | 143.960 | 5.0 | 144.162 | 4.1 | 143.895 | 2.1 |
| 2 | 145.117 | 4.5 | 145.166 | 5.7 | 145.323 | 5.8 | 144.919 | 5.7 | 144.948 | 7.9 | 144.700 | 7.3 | 144.599 | 7.7 | 144.587 | 5.5 | 144.630 | 5.0 | 144.529 | 3.6 |
| 3 | 145.526 | 5.3 | 145.673 | 6.9 | 145.850 | 7.0 | 145.379 | 6.7 | 145.467 | 10.1 | 145.116 | 8.6 | 145.064 | 8.7 | 145.046 | 6.7 | 145.096 | 7.4 | 144.909 | 5.0 |
| 4 | 146.095 | 6.4 | 146.253 | 7.4 | 146.328 | 9.3 | 145.855 | 9.1 | 146.192 | 13.8 | 145.559 | 10.1 | 145.521 | 9.7 | 145.416 | 8.3 | 145.398 | 7.8 | 145.500 | 6.5 |
| 5 | 146.504 | 7.7 | 146.797 | 9.7 | 146.870 | 11.1 | 146.454 | 10.9 | 146.657 | 14.9 | 146.100 | 11.0 | 146.012 | 11.7 | 145.836 | 10.7 | 145.838 | 8.8 | 145.825 | 7.6 |
| 6 | 146.867 | 8.8 | 147.223 | 10.0 | 147.262 | 12.3 | 146.955 | 12.9 | 147.028 | 16.1 | 146.482 | 12.3 | 146.500 | 14.4 | 146.165 | 11.4 | 146.240 | 10.2 | 146.239 | 7.8 |
| 7 | 147.281 | 10.5 | 147.710 | 11.2 | 147.696 | 13.5 | 147.334 | 14.3 | 147.620 | 17.6 | 147.011 | 13.2 | 147.033 | 15.5 | 146.725 | 13.8 | 146.640 | 11.5 | 146.632 | 8.9 |
| 8 | 147.857 | 10.9 | 148.130 | 12.4 | 148.009 | 14.7 | 147.759 | 16.6 | 147.998 | 19.7 | 147.490 | 15.6 | 147.530 | 17.2 | 147.174 | 14.9 | 147.030 | 13.7 | 147.026 | 9.5 |
| 9 | 148.312 | 11.4 | 148.468 | 13.4 | 148.385 | 16.1 | 148.197 | 18.2 | 148.307 | 20.8 | 147.951 | 16.7 | 148.006 | 18.3 | 147.689 | 16.4 | 147.397 | 14.5 | 147.375 | 10.3 |
| 10 | 148.750 | 12.7 | 148.915 | 14.6 | 148.795 | 17.4 | 148.579 | 19.9 | 148.827 | 22.5 | 148.293 | 18.2 | 148.401 | 20.6 | 148.088 | 18.0 | 147.872 | 15.5 | 147.813 | 12.0 |
| 11 | 149.031 | 13.5 | 149.183 | 15.3 | 149.211 | 18.8 | 149.040 | 21.8 | 149.207 | 24.4 | 148.737 | 19.9 | 148.757 | 22.2 | 148.500 | 19.0 | 148.342 | 16.6 | 148.242 | 13.4 |
| 12 | 149.412 | 14.8 | 149.591 | 16.2 | 149.581 | 19.7 | 149.399 | 23.2 | 149.447 | 24.9 | 149.096 | 21.0 | 149.016 | 23.6 | 148.961 | 22.0 | 148.736 | 18.2 | 148.471 | 14.0 |
| 13 | 149.764 | 15.9 | 149.813 | 16.8 | 149.970 | 20.8 | 149.676 | 23.5 | 149.762 | 27.2 | 149.441 | 22.9 | 149.343 | 25.1 | 149.262 | 22.2 | 149.128 | 19.1 | 148.683 | 14.5 |
| 14 | 150.072 | 17.1 | 150.219 | 18.0 | 150.241 | 21.9 | 149.973 | 24.3 | 150.104 | 28.5 | 149.694 | 24.6 | 149.598 | 26.2 | 149.619 | 23.0 | 149.370 | 20.0 | 149.032 | 15.7 |
| 15 | 150.279 | 17.8 | 150.487 | 18.9 | 150.487 | 23.1 | 150.246 | 25.3 | 150.376 | 29.3 | 150.033 | 25.9 | 149.878 | 27.1 | 149.848 | 24.2 | 149.753 | 21.3 | 149.302 | 16.3 |
| 16 | 150.530 | 18.6 | 150.757 | 19.6 | 150.814 | 23.7 | 150.542 | 26.3 | 150.608 | 30.0 | 150.295 | 26.4 | 150.261 | 28.4 | 150.113 | 24.8 | 150.242 | 23.4 | 149.611 | 17.4 |
| 17 | 150.777 | 19.4 | 151.051 | 19.9 | 150.999 | 24.0 | 150.803 | 27.1 | 150.861 | 30.9 | 150.675 | 27.7 | 150.602 | 29.6 | 150.269 | 25.2 | 150.496 | 24.2 | 149.889 | 18.6 |
| 18 | 150.777 | 19.4 | 151.051 | 19.9 | 150.999 | 24.0 | 150.803 | 27.1 | 150.861 | 30.9 | 150.675 | 27.7 | 150.602 | 29.6 | 150.269 | 25.2 | 150.496 | 24.2 | 149.889 | 18.6 |
| 19 | 151.000 | 20.6 | 151.235 | 20.5 | 151.252 | 24.7 | 151.025 | 28.0 | 151.103 | 31.4 | 150.851 | 28.5 | 150.812 | 30.4 | 150.480 | 25.7 | 150.620 | 24.7 | 150.045 | 19.4 |
| 20 | 151.258 | 20.7 | 151.457 | 21.0 | 151.445 | 25.0 | 151.218 | 28.6 | 151.300 | 31.8 | 151.058 | 29.6 | 151.051 | 30.8 | 150.841 | 26.6 | 150.901 | 26.0 | 150.460 | 20.5 |
| 21 | 151.481 | 21.2 | 151.675 | 21.5 | 151.665 | 25.5 | 151.480 | 29.1 | 151.545 | 32.8 | 151.402 | 30.8 | 151.308 | 31.7 | 151.248 | 27.4 | 151.143 | 26.7 | 150.654 | 21.1 |
| 22 | 151.738 | 21.6 | 151.826 | 22.0 | 151.906 | 26.2 | 151.665 | 29.8 | 151.834 | 33.8 | 151.668 | 31.9 | 151.548 | 33.1 | 151.579 | 27.9 | 151.408 | 27.2 | 150.958 | 22.1 |
| 23 | 151.967 | 21.8 | 152.058 | 22.6 | 152.105 | 26.7 | 151.857 | 30.4 | 152.002 | 34.4 | 151.903 | 33.8 | 151.818 | 33.6 | 151.837 | 29.1 | 151.712 | 28.3 | 151.362 | 22.8 |
| 24 | 152.222 | 22.4 | 152.413 | 23.2 | 152.390 | 27.5 | 152.087 | 31.0 | 152.168 | 34.8 | 152.164 | 34.6 | 152.069 | 34.0 | 152.112 | 30.0 | 151.901 | 29.1 | 151.555 | 23.3 |
| 25 | 152.547 | 23.3 | 152.643 | 24.0 | 152.546 | 28.0 | 152.318 | 31.5 | 152.536 | 36.1 | 152.283 | 34.9 | 152.364 | 34.7 | 152.445 | 31.4 | 152.382 | 29.9 | 151.897 | 25.4 |
| 26 | 152.710 | 24.2 | 152.825 | 24.9 | 152.764 | 28.8 | 152.566 | 32.1 | 152.761 | 36.8 | 152.585 | 36.2 | 152.587 | 35.8 | 152.656 | 32.2 | 152.722 | 31.2 | 152.150 | 26.1 |
| 27 | 152.956 | 25.0 | 153.039 | 25.4 | 152.991 | 29.7 | 152.685 | 32.6 | 152.918 | 37.3 | 152.803 | 37.1 | 152.731 | 35.4 | 152.879 | 33.5 | 152.867 | 31.6 | 152.449 | 28.6 |
| 28 | 153.252 | 26.1 | 153.228 | 26.0 | 153.221 | 30.6 | 153.027 | 33.5 | 153.084 | 37.9 | 153.020 | 38.4 | 153.061 | 36.3 | 153.140 | 34.6 | 153.102 | 32.2 | 152.846 | 30.9 |
| 29 | 153.500 | 26.9 | 153.451 | 26.8 | 153.494 | 31.7 | 153.328 | 34.5 | 153.365 | 39.2 | 153.242 | 38.7 | 153.283 | 37.0 | 153.369 | 35.3 | 153.358 | 32.9 | 153.068 | 32.2 |
| 30 | 153.620 | 27.3 | 153.664 | 27.5 | 153.638 | 32.3 | 153.505 | 35.3 | 153.552 | 40.0 | 153.579 | 39.4 | 153.449 | 38.2 | 153.602 | 35.7 | 153.562 | 33.7 | 153.303 | 33.5 |
| 31 | 153.848 | 27.9 | 153.875 | 28.2 | 153.825 | 32.8 | 153.672 | 36.1 | 153.704 | 40.3 | 153.722 | 39.8 | 153.741 | 39.7 | 153.764 | 36.1 | 153.736 | 34.3 | 153.494 | 34.0 |
| 32 | 154.016 | 28.4 | 153.990 | 28.7 | 153.960 | 33.5 | 153.851 | 36.6 | 153.920 | 41.0 | 153.898 | 40.5 | 153.874 | 40.2 | 153.888 | 36.5 | 153.926 | 35.1 | 153.771 | 35.3 |
| 33 | 154.204 | 29.3 | 154.123 | 29.3 | 154.162 | 34.5 | 153.955 | 37.0 | 154.181 | 42.4 | 154.060 | 40.8 | 154.048 | 40.8 | 154.048 | 36.9 | 154.080 | 36.0 | 154.025 | 36.1 |
| 34 | 154.440 | 30.4 | 154.416 | 31.2 | 154.435 | 35.5 | 154.544 | 39.3 | 154.544 | 42.9 | 154.544 | 43.0 | 154.544 | 43.1 | 154.544 | 38.7 | 154.544 | 37.5 | 154.544 | 38.4 |

Table 6. Evaluation result of various hyper-parameters trained with COCO-Single Dataset

| Threshold index | $\theta$=1, $\lambda$=1 | | $\theta$=0.5, $\lambda$=1 | | $\theta$=0.5, $\lambda$=0.5 | | $\theta$=0.1, $\lambda$=1 | | $\theta$=0.05, $\lambda$=1 | | $\theta$=0.012, $\lambda$=1 | | $\theta$=0.0115, $\lambda$=1 | | $\theta$=0.011, $\lambda$=1 | | $\theta$=0.01, $\lambda$=1 | | $\theta$=0.001, $\lambda$=1 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | FLOPs (G) | mAP (%) | FLOPs (G) | mAP (%) | FLOPs (G) | mAP (%) | FLOPs (G) | mAP (%) | FLOPs (G) | mAP (%) | FLOPs (G) | mAP (%) | FLOPs (G) | mAP (%) | FLOPs (G) | mAP (%) | FLOPs (G) | mAP (%) | FLOPs (G) | mAP (%) |
| 1 | 133.807 | 2.1 | 133.896 | 3.3 | 133.858 | 3.1 | 133.792 | 4.9 | 133.558 | 4.6 | 134.361 | 6.6 | 134.019 | 6.0 | 134.339 | 5.8 | 134.277 | 5.1 | 134.491 | 3.4 |
| 2 | 134.342 | 3.1 | 134.396 | 3.9 | 134.419 | 3.8 | 134.179 | 5.6 | 134.020 | 5.3 | 134.861 | 8.3 | 134.470 | 7.7 | 134.806 | 7.5 | 134.709 | 6.4 | 134.937 | 4.5 |
| 3 | 134.925 | 4.0 | 134.873 | 4.5 | 134.896 | 3.8 | 134.630 | 6.3 | 134.525 | 7.1 | 135.404 | 11.1 | 134.973 | 8.7 | 135.266 | 9.2 | 135.101 | 8.0 | 135.371 | 5.8 |
| 4 | 135.362 | 4.8 | 135.335 | 5.0 | 135.375 | 4.5 | 135.080 | 7.4 | 134.961 | 8.4 | 135.899 | 12.0 | 135.453 | 10.6 | 135.766 | 10.6 | 135.553 | 8.3 | 135.777 | 6.5 |
| 5 | 135.838 | 4.9 | 135.800 | 5.8 | 135.813 | 5.6 | 135.498 | 7.9 | 135.399 | 8.5 | 136.367 | 13.9 | 135.858 | 11.9 | 136.181 | 12.5 | 135.946 | 9.6 | 136.215 | 7.3 |
| 6 | 136.319 | 5.5 | 136.186 | 7.4 | 136.260 | 5.9 | 135.954 | 8.8 | 135.885 | 10.0 | 136.811 | 16.0 | 136.253 | 13.2 | 136.598 | 14.4 | 136.327 | 11.2 | 136.628 | 8.8 |
| 7 | 136.777 | 5.9 | 136.615 | 8.3 | 136.686 | 7.1 | 136.376 | 9.2 | 136.321 | 11.1 | 137.251 | 17.3 | 136.650 | 14.7 | 136.929 | 15.9 | 136.668 | 13.0 | 136.985 | 10.6 |
| 8 | 137.166 | 6.0 | 136.985 | 8.6 | 137.012 | 7.5 | 136.749 | 10.8 | 136.791 | 12.1 | 137.601 | 18.8 | 137.050 | 16.4 | 137.280 | 17.2 | 137.034 | 15.1 | 137.340 | 11.8 |
| 9 | 137.463 | 6.7 | 137.289 | 9.3 | 137.403 | 7.7 | 137.053 | 12.2 | 137.205 | 13.0 | 137.950 | 19.9 | 137.407 | 18.2 | 137.518 | 19.0 | 137.471 | 17.7 | 137.647 | 13.1 |
| 10 | 137.808 | 7.2 | 137.566 | 10.2 | 137.684 | 8.0 | 137.323 | 13.9 | 137.529 | 14.3 | 138.226 | 21.1 | 137.678 | 19.7 | 137.857 | 19.8 | 137.769 | 18.9 | 137.935 | 14.4 |
| 11 | 138.134 | 7.4 | 137.844 | 11.0 | 137.959 | 8.3 | 137.603 | 15.7 | 137.803 | 15.9 | 138.481 | 22.2 | 137.976 | 21.3 | 138.091 | 20.8 | 138.102 | 20.5 | 138.166 | 15.6 |
| 12 | 138.371 | 7.7 | 138.095 | 11.7 | 138.185 | 9.4 | 137.805 | 17.0 | 138.061 | 17.1 | 138.670 | 23.1 | 138.288 | 22.7 | 138.326 | 22.3 | 138.364 | 21.7 | 138.380 | 16.2 |
| 13 | 138.637 | 8.7 | 138.248 | 12.0 | 138.410 | 9.7 | 138.018 | 17.6 | 138.283 | 18.3 | 138.884 | 24.2 | 138.552 | 24.1 | 138.539 | 24.1 | 138.615 | 22.9 | 138.538 | 16.7 |
| 14 | 138.845 | 9.1 | 138.439 | 12.5 | 138.606 | 10.0 | 138.220 | 18.3 | 138.502 | 19.4 | 139.065 | 25.1 | 138.805 | 24.9 | 138.769 | 25.2 | 138.861 | 23.5 | 138.772 | 18.1 |
| 15 | 139.020 | 9.7 | 138.603 | 13.2 | 138.783 | 10.8 | 138.446 | 19.1 | 138.666 | 20.1 | 139.208 | 25.5 | 139.042 | 26.0 | 138.998 | 26.3 | 139.072 | 24.3 | 138.978 | 20.1 |
| 16 | 139.200 | 9.9 | 138.752 | 13.9 | 138.903 | 11.0 | 138.614 | 20.2 | 138.867 | 20.9 | 139.357 | 26.4 | 139.230 | 26.6 | 139.211 | 26.9 | 139.248 | 25.0 | 139.182 | 21.2 |
| 17 | 139.348 | 10.2 | 138.889 | 14.5 | 139.023 | 11.2 | 138.771 | 20.7 | 139.012 | 21.3 | 139.503 | 27.0 | 139.423 | 27.2 | 139.376 | 27.5 | 139.396 | 26.0 | 139.388 | 22.3 |
| 18 | 139.348 | 10.2 | 138.889 | 14.5 | 139.023 | 11.2 | 138.771 | 20.7 | 139.012 | 21.3 | 139.503 | 27.0 | 139.423 | 27.2 | 139.376 | 27.5 | 139.396 | 26.0 | 139.388 | 22.3 |
| 19 | 139.490 | 10.7 | 139.015 | 14.9 | 139.192 | 12.1 | 138.922 | 21.3 | 139.190 | 21.7 | 139.611 | 27.6 | 139.602 | 28.3 | 139.552 | 28.4 | 139.540 | 26.9 | 139.495 | 22.4 |
| 20 | 139.635 | 11.0 | 139.170 | 14.9 | 139.339 | 12.4 | 139.050 | 22.1 | 139.363 | 22.7 | 139.756 | 28.9 | 139.757 | 28.9 | 139.675 | 29.1 | 139.694 | 28.2 | 139.658 | 22.9 |
| 21 | 139.778 | 11.5 | 139.320 | 15.6 | 139.471 | 13.1 | 139.192 | 23.6 | 139.547 | 24.4 | 139.861 | 29.5 | 139.880 | 29.5 | 139.812 | 30.1 | 139.872 | 29.3 | 139.777 | 23.9 |
| 22 | 139.895 | 11.9 | 139.471 | 16.3 | 139.620 | 13.5 | 139.346 | 24.2 | 139.724 | 25.4 | 139.983 | 30.0 | 140.009 | 30.4 | 139.975 | 31.0 | 140.016 | 29.8 | 139.921 | 24.9 |
| 23 | 140.031 | 12.6 | 139.594 | 16.9 | 139.758 | 14.1 | 139.491 | 24.9 | 139.921 | 26.3 | 140.101 | 31.1 | 140.118 | 31.2 | 140.091 | 31.6 | 140.135 | 30.2 | 140.047 | 25.6 |
| 24 | 140.157 | 13.0 | 139.751 | 17.6 | 139.860 | 14.3 | 139.621 | 25.4 | 140.085 | 27.4 | 140.204 | 32.3 | 140.212 | 32.0 | 140.226 | 32.3 | 140.233 | 31.1 | 140.190 | 26.9 |
| 25 | 140.245 | 13.5 | 139.886 | 17.8 | 139.961 | 14.7 | 139.765 | 25.8 | 140.226 | 28.7 | 140.308 | 33.4 | 140.297 | 33.2 | 140.311 | 33.0 | 140.318 | 31.9 | 140.283 | 29.0 |
| 26 | 140.374 | 13.7 | 139.992 | 18.2 | 140.094 | 15.2 | 139.885 | 26.0 | 140.319 | 29.7 | 140.398 | 34.5 | 140.409 | 34.6 | 140.382 | 34.3 | 140.391 | 32.8 | 140.395 | 31.0 |
| 27 | 140.476 | 14.2 | 140.106 | 18.6 | 140.200 | 15.6 | 140.011 | 26.5 | 140.413 | 30.5 | 140.453 | 35.7 | 140.455 | 35.4 | 140.450 | 35.7 | 140.454 | 33.8 | 140.446 | 32.0 |
| 28 | 140.547 | 14.4 | 140.201 | 19.0 | 140.317 | 16.1 | 140.125 | 27.1 | 140.513 | 31.6 | 140.519 | 36.4 | 140.509 | 36.5 | 140.502 | 36.3 | 140.516 | 34.7 | 140.506 | 32.4 |
| 29 | 140.609 | 14.7 | 140.301 | 19.6 | 140.405 | 16.5 | 140.280 | 27.5 | 140.585 | 32.2 | 140.576 | 37.4 | 140.552 | 37.7 | 140.561 | 36.7 | 140.578 | 35.8 | 140.585 | 33.0 |
| 30 | 140.660 | 15.1 | 140.410 | 20.2 | 140.494 | 17.0 | 140.399 | 27.8 | 140.661 | 32.8 | 140.636 | 37.8 | 140.615 | 38.3 | 140.614 | 37.6 | 140.623 | 36.1 | 140.635 | 33.2 |
| 31 | 140.715 | 15.4 | 140.491 | 20.5 | 140.574 | 17.4 | 140.512 | 28.4 | 140.703 | 33.2 | 140.688 | 38.1 | 140.672 | 38.6 | 140.668 | 38.1 | 140.675 | 36.1 | 140.684 | 33.3 |
| 32 | 140.755 | 15.8 | 140.589 | 21.3 | 140.638 | 17.6 | 140.628 | 28.8 | 140.738 | 33.6 | 140.751 | 38.5 | 140.735 | 39.1 | 140.706 | 38.5 | 140.724 | 36.6 | 140.734 | 34.1 |
| 33 | 140.795 | 16.3 | 140.699 | 21.9 | 140.711 | 18.3 | 140.710 | 29.6 | 140.784 | 33.9 | 140.790 | 38.8 | 140.764 | 39.6 | 140.762 | 38.8 | 140.764 | 36.9 | 140.768 | 34.3 |
| 34 | 140.869 | 17.2 | 140.860 | 23.3 | 140.869 | 20.3 | 140.872 | 30.9 | 140.872 | 34.7 | 140.872 | 39.2 | 140.872 | 40.3 | 140.872 | 39.5 | 140.872 | 37.3 | 140.872 | 36.1 |

Table 7. Evaluation result of various hyper-parameters trained with ILSVRC-Single Dataset