

DETR++: Taming Your Multi-Scale Detection Transformer

Chi Zhang^{1,2,*}, Lijuan Liu², Xiaoxue Zang^{2,*},
Frederick Liu², Hao Zhang², Xinying Song², Jindong Chen²

¹ Department of Computer Science, University of California, Los Angeles

² Google Research

chi.zhang@ucla.edu, lijuanliu@google.com, zxx1204007@gmail.com

{frederickliu, haozhangthu, xysong, jdchen}@google.com

Abstract

Convolutional Neural Networks (CNN) have dominated the field of detection ever since the success of AlexNet in ImageNet classification [12]. With the sweeping reform of Transformers [27] in natural language processing, Carion et al. [2] introduce the Transformer-based detection method, i.e., DETR. However, due to the quadratic complexity in the self-attention mechanism in the Transformer, DETR is never able to incorporate multi-scale features as performed in existing CNN-based detectors, leading to inferior results in small object detection. To mitigate this issue and further improve performance of DETR, in this work, we investigate different methods to incorporate multi-scale features and find that a Bi-directional Feature Pyramid (BiFPN) works best with DETR in further raising the detection precision. With this discovery, we propose DETR++, a new architecture that improves detection results by 1.9% AP on MS COCO 2017, 11.5% AP on RICO icon detection, and 9.1% AP on RICO layout extraction over existing baselines.

1. Introduction

All well-performing existing object detection systems leverage the Convolutional Neural Networks (CNN) and address the set prediction problem in matching box proposals and ground truths using predefined heuristics: typical examples include anchors [7, 8, 14, 17, 23], grid [20–22], point centers [26, 34]. However, while good practices have been identified in widely used datasets, like MS COCO [15] and PASCAL [5], the post-processing steps are critical performance factors for relatively under-explored areas, like icon detection and layout extraction in on-device screen understanding tasks [3].

A new Transformer-based [27] detection method, i.e., DETR [2], has been recently proposed to mitigate these issues. Specifically, the DETR model uses a ResNet [9] backbone to extract higher-level visual features. A Transformer encoder further aggregates global features in each token from the flattened visual features. Finally, a Transformer decoder decodes the encoded features into box proposals. While the backbone and the encoder are relatively standard, the design in the decoder does make a difference. Specifically, the decoder module takes as input a zero matrix as the sentence embedding and learnable position embedding referred to as *object queries*. Position embedding for the encoder and the object queries are added to each attention layer’s key and query respectively and decoded parallelly. A final object classifier and a box regressor are attached to the output features of the decoder and generate object proposals represented as (c, cy, cx, h, w) , denoting the class c , box center (cy, cx) , and box height and width (h, w) . The box proposals and the ground truth boxes are matched via the Hungarian algorithm, relieving the detector designer from crafting the matching heuristics, and the classification and box regression losses are jointly minimized.

Despite the simpler design in the DETR architecture, earlier experimental results show that the DETR model is inferior to existing convolutional models and also slower in training. There are two sources contributing to complexity in the model: (1) the self-attention mechanism in the encoder is resource-hungry, especially for visual features that could span over thousands of tokens and (2) the Hungarian matcher is cubic in time. These slow operations make the common strategy of adding multi-scale features in a detector to improve performance a non-trivial work: running the detector head on multi-scale features, e.g., 160×160 in the visual feature hierarchy, or simply increasing the number of proposals is extremely memory- and time-consuming.

Therefore, in this work, we set out to study what is the best way to incorporate multi-level features into the DETR architecture to improve performance, while not incurring

* work done while at Google Research.

the quadratic complexity and the cubic complexity in the self-attention and the matcher. Specifically, we note that: (1) running the head on multi-level features is nearly impossible given the resource and time constraints, (2) the Transformer encoder plays a crucial role in the detector and cannot be removed, (3) the shifted window idea [18] does not work well on multi-level features, (4) specialized DETR heads for different object scales linearly increase complexity but do not fare better than the baseline, (5) a Bi-directional Feature Pyramid architecture improves the performance and only marginally adds to the complexity.

In the remainder of this paper, we review existing works on object detection and multi-level features in Sec. 2, discuss the multi-scale designs we explored and introduce DETR++ in Sec. 3, detail the experimental setup in Sec. 4, and finally conclude the paper in Sec. 5.

2. Related Work

2.1. Object Detectors

Common object detection systems can be categorized into two routes: two-stage object detection and one-stage object detection. In the two-stage detectors, a region proposal network will first propose potential bounding boxes that may contain objects and a final classifier decides what kind of an object the bounding box contains. The most typical two-stage object detector is the Faster R-CNN [23], whose Region Proposal Network and the classifier head are responsible for the two sub-tasks respectively. The one-stage object detectors merge the two sub-routines into one by either predefining the proposal grids [20], the object anchors [14, 17, 26], or the dense object center points [34]. These detection systems predict corresponding bounding boxes with respect to the predefined constructs from the same features used for the object class prediction. However, these one-stage systems fare slightly worse than the two-stage systems potentially due to the region mismatch that is better handled by a region proposal stream. However, neither the two-stage detectors nor the one-stage detectors are relieved from the manual post-processing step of Non-Maximum Suppression (NMS).

A recently proposed Transformer-based detector, *i.e.*, DETR, successfully removes the post-processing steps using set-based matching and prediction. In particular, this one-stage detector predicts bounding boxes and object classes parallelly from an encoder-decoder architecture and matches the proposals with ground truths via Hungarian matching [11]. The matched boxes incur the classification loss and box regression loss, whose gradients are backpropagated through the entire model.

2.2. Multi-Scale Detection

One critical component of existing object detection systems is multi-level feature aggregation. Multi-level feature

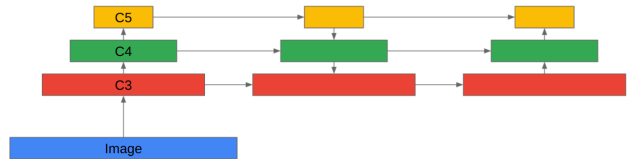


Figure 1. The idea of Bi-direction Feature Pyramid, where, apart from the top-down and lateral pathway in the traditional Feature Pyramid, an additional bottom-up pathway is added.

aggregation allows the model to leverage features of different granularity and would effectively improve the model on small object detection. The most popular multi-scale strategy is using Feature Pyramid Network (FPN) [13]. Specifically, multi-level features from the detection backbone are processed through FPN: the resolution of each level remains the same and the detection head is attached onto feature maps of all resolutions. The detection results are then NMS-filtered to produce the final prediction. Following the idea, the PANet [16] method supplements the top-down direction of FPN with a bottom-up path. Ghiasi *et al.* [6] even consider using Neural Architecture Search to automatically find a better FPN structure. Recently, EfficientDet [25] proposes a simpler, scalable, and effective FPN module that employs the bi-directional idea, residual connection and weighted averaging.

However, due to the computational complexity in DETR, the original model cannot easily incorporate a multi-scale feature aggregation module. The lack of multi-level feature representation negatively impacts small objects’ detection precision in the original DETR model.

3. Multi-Scale Designs

In this section, we discuss a few potential multi-scale strategies before finally presenting how we incorporate an effective design into the original model to make DETR++.

With the time complexity of the self-attention mechanism in the Transformer encoder being $O(N^2)$ and that of the Hungarian matching being $O(N^3)$, simply using large-resolution features for detection and (or) increasing the number of box predictions are not computationally feasible. Therefore, we investigate the following strategies to incorporate multi-scale features.

3.1. Removing the Encoder

Now that one of the computational burdens is from the self-attention mechanism, one intuitive idea is to remove the encoder and only use the decoder as the decoder’s complexity is only of $O(MN)$, where M denotes the number of object queries (100 in DETR) and N denotes the size of the encoder output.

We consider two potential aggregation methods using the decoder only.

Stack In this stacking strategy, we consecutively apply three decoders on the image features from C_3 , C_4 , and C_5 . The decoded output from C_3 is further processed by the decoder for C_4 , followed by the C_5 decoder. We use six-layer transformer decoders for the three scales of input and compute the auxiliary loss on every layer’s output.

Multi-Head We also consider the multi-head method, where we use three six-layer decoders for each resolution similarly to the stacking method. However, unlike the stacking method, each decoder independently produces the box proposals from a single scale. For loss computation, we concatenate all box proposals from all the three decoders.

3.2. Shifted Windows

Inspired by the Swin Transformer [18], we consider applying the Transformer detection head on each shifted window. This will only linearly increase computation rather than quadratically.

In particular, we set the window size to be the spatial size of the C_5 features and correspondingly crop non-overlapping patches out of C_4 and (or) C_3 features. The patches are sent to the Transformer-based detection head and the proposal boxes are concatenated. In practice, we only use C_4 and C_5 since adding two many patches will correspondingly increase the number of boxes, which slows down training due to the cubic complexity in Hungarian matching.

3.3. Specialized Heads

Another way to better leverage multi-scale information, however, is to disentangle detection for objects of different scales, such that features from larger-scale objects will not negatively impact small-scale objects and vice versa. To implement the idea, we propose to use specialized heads for object detection of different scales.

Specifically, we use three detection heads for small objects, medium objects, and large objects, respectively. All the detection heads operate on the C_5 features and produce box proposals to be concatenated for loss computation.

In practice, we split the object boxes in data processing into the three scales, and during training, we use separate Transformer detection heads for each scale.

3.4. Bi-directional Feature Pyramid

Conventionally, the Feature Pyramid Network [13] is used to produce multiple levels of features, each of which is attached to the detection head for object detection. However, this method simply cannot be used due to the quadratic complexity in the Transformer encoder. Therefore, we propose to aggregate the multiple features using a Bi-directional Feature Pyramid Network (BiFPN) [25]. BiFPN works by supplying the top-down direction of the traditional

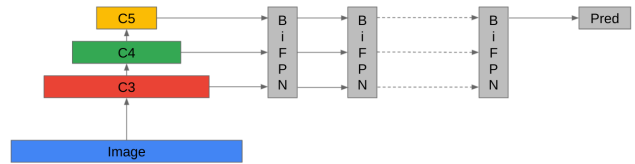


Figure 2. DETR++ model, where multi-scale features are fed into repeated BiFPN layers and only one scale of image features is sent to the encoder and decoder (not shown) for detection.

FPN with an additional bottom-up pathway. See Fig. 1 for a graphical illustration of the idea. We also consider stacking BiFPN layers to produce better representation. Finally, we pick one scale of the final BiFPN layer’s features and attach it to the Transformer detection head for object detection.

3.5. DETR++

After extensive experimentation, we propose the DETR++ architecture that adds the Bi-directional Feature Pyramid into the original DETR model. Specifically, we connect the BiFPN module to the feature output of C_3 , C_4 , and C_5 from the ResNet backbone, and stack it 8 times, before finally feeding the multi-scale feature-aggregated C_5 output to the Transformer architecture. Fig. 2 shows the final architecture of our model.

4. Experiments

We show DETR++ achieves significant performance improvement over baselines in object detection for natural images and on-device screen understanding tasks like icon detection and layout extraction.

Training Details We train DETR++ with the AdamW [19] optimizer. The initial learning rate and the weight decay are both set to 1×10^{-4} . The learning rate is scheduled to decrease 10 times after 200k steps and the training process completes after 500k steps. We use the ImageNet-pretrained ResNet-50 [9] as our backbone. Of note, the backbone has the same learning rate with the rest of the model. The 8-layer BiFPN module is attached to the last three intermediate features and we use six layer of blocks in both the Transformer encoder and the decoder. Both the Transformer encoder and the decoder have a dropout [24] rate of 0.1 and apply post normalization.

We use data augmentation in preprocessing. In particular, we use random horizontal flip and random crop during training such that the short size of the image is at least 480 and at most 800 and the long size of the image does not exceed 1280. The image is then padded to be of shape 1280×1280 .

We follow the original prediction setup in DETR [2], *i.e.*, we overwrite predictions of background with the most-likely non-background objects and their corresponding confidence.

Method	AP	AP@0.5	AP@0.75	AP ^L	AP ^M	AP ^S	Method	AP	AP@0.5	AP@0.75	AP ^L	AP ^M	AP ^S
DETR-NoEnc-Stack	37.3	56.8	39.7	54.4	40.6	16.9	DETR++	48.1	89.8	45.3	52.9	49.6	43.6
DETR-NoEnc-MHead	35.0	54.9	36.3	52.0	37.5	14.6	DETR	47.4	89.4	44.3	52.0	48.8	43.1
DETR-Swin	39.9	59.8	42.2	57.9	43.6	18.4	IconNet	36.6	79.3	26.8	15.1	35.6	36.8
DETR-SHead	36.4	54.0	39.2	54.7	39.5	15.1	Method	AP	AP@0.5	AP@0.75	AP ^L	AP ^M	AP ^S
DETR++	41.8	60.1	44.6	58.6	45.0	22.1	DETR++	25.3	43.6	24.2	28.6	9.7	1.4
DETR	39.9	59.8	42.4	57.2	43.3	18.8	DETR	24.7	42.5	23.4	28.0	8.1	1.1
CenterNet	41.6	59.4	44.2	54.1	43.1	22.5	IconNet	16.2	30.5	15.9	19.8	8.6	5.6

(a) Model performance on MS COCO 2017, where NoEnc denotes the design without the encoder, MHead is short for multi-head, Swin for shifted windows, and SHead for specialized heads.

(b) Model performance on RICO icon detection (top) and RICO layout detection (bottom).

Table 1. Experimental evaluation.

Original hyperparameter settings have been inherited. However, we note that the number of boxes and the background class weight are two important hyperparameters to tune for different datasets. One rule of thumb we note is to set the background class weight to be approximately

$$\frac{\text{Ground truth boxes}}{\text{Box proposals} \times \text{Images} - \text{Ground truth boxes}}. \quad (1)$$

Datasets In the experiments, we validate the performance of our model on three datasets, the MS COCO 2017 dataset [15] as a standard benchmark for object detection, and RICO icon detection dataset and RICO layout extraction dataset [3] for smart device screen understanding. The COCO dataset contains natural images with annotated common objects; the RICO icon dataset consists of Android device snapshots with annotated icons; the RICO layout dataset provides bounding boxes and class labels for functional areas in Android device screen snapshots.

Performance on MS COCO Tab. 1a shows the performance of various multi-scale schemes on the MS COCO 2017 dataset. As indicated in the table, the idea to remove the encoder will negatively impact the model performance: AP values significantly drop in both the stack design and the multi-head design. The shifted window idea achieves similar performance with the original DETR model with slightly inferior small object detection precision. The specialized head design neither improves over the baseline DETR model but only linearly increases the computation. Compared to others, our DETR++ model with the BiFPN module significantly improves over the baseline DETR model and is even better than the CenterNet model of similar compute [4] except in small object detection.

Performance on Icon Given the results in MS COCO detection, we evaluate the best multi-scale strategy of BiFPN on the RICO icon detection dataset [3]. Tab. 1b shows the results of the experiments. Notably, the DETR++ model improves the IconNet model [1, 10] by a large margin and also fares better than the DETR baseline. DETR++ is better

than DETR in every aspect, and achieves even better results than IconNet on small and medium object detection.

Performance on Layout The same set of models is run on the RICO layout extraction dataset [3] as well. Consistent with earlier results, the DETR++ model improves the DETR model on every precision metrics. However, in the layout extraction task, the small object detection is more serious than earlier datasets and the DETR++ model becomes inferior to the IconNet in AP^S.

5. Conclusion and Future Work

In this work, we investigate the multi-scale strategies in scaling the DETR model and propose the DETR++ model. Specifically, the DETR++ model incorporates a Bi-direction Feature Pyramid Network to aggregate multi-level image features to improve small object detection in the original DETR model. In experiments, we note that DETR++ improves detection results by 1.9% AP on MS COCO 2017, 11.5% AP on icon detection, and 9.1% AP on layout extraction over existing baselines.

Despite success in using BiFPN for multi-level feature aggregation, there is still room for improvement for DETR++ in small object detection. Specifically, the DETR++ model is still not as performant as CenterNet and IconNet in small object detection in specific datasets. These gaps further motivate us in pursuing this direction to make Transformer-based detectors a first-class citizen.

Convergence speed of the DETR++ model is also slower than existing baselines. This slow-down significantly impacts model iteration. While we have noticed recent attempts to improve convergence speed, few of them will bring the model to the optimal point from the plain method.

Finally, we see a bigger picture of the Transformer-based detector in multi-modal reasoning [28–33, 35]. With the effective large language learning models, we could combine language features and the Transformer detector to enable open-domain object detection. The direction could potentially change the landscape of detection: the detector is no longer fixed on a domain but could quickly use language features to determine where an object is and what that object is.

References

- [1] Chongyang Bai, Xiaoxue Zang, Ying Xu, Srinivas Sunkara, Abhinav Rastogi, Jindong Chen, et al. Uibert: Learning generic multimodal representations for ui understanding. *arXiv preprint arXiv:2107.13731*, 2021. 4
- [2] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *ECCV*, 2020. 1, 3
- [3] Biplab Deka, Zifeng Huang, Chad Franzen, Joshua Hirschman, Daniel Afegan, Yang Li, Jeffrey Nichols, and Ranjitha Kumar. Rico: A mobile app dataset for building data-driven design applications. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*, 2017. 1, 4
- [4] Kaiwen Duan, Song Bai, Lingxi Xie, Honggang Qi, Qingming Huang, and Qi Tian. Centernet: Keypoint triplets for object detection. In *ICCV*, 2019. 4
- [5] Mark Everingham, SM Ali Eslami, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes challenge: A retrospective. *IJCV*, 111(1):98–136, 2015. 1
- [6] Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V Le. Nas-fpn: Learning scalable feature pyramid architecture for object detection. In *CVPR*, 2019. 2
- [7] Ross Girshick. Fast r-cnn. In *ICCV*, 2015. 1
- [8] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014. 1
- [9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 1, 3
- [10] Zecheng He, Srinivas Sunkara, Xiaoxue Zang, Ying Xu, Lijuan Liu, Nevan Wichers, Gabriel Schubiner, Ruby Lee, Jindong Chen, and Blaise Aguera y Arcas. Actionbert: Leveraging user actions for semantic understanding of user interfaces. *arXiv preprint arXiv:2012.12350*, 2020. 4
- [11] Roy Jonker and Anton Volgenant. A shortest augmenting path algorithm for dense and sparse linear assignment problems. *Computing*, 38(4):325–340, 1987. 2
- [12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *NeurIPS*, 2012. 1
- [13] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017. 2, 3
- [14] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *ICCV*, 2017. 1, 2
- [15] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014. 1, 4
- [16] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. Path aggregation network for instance segmentation. In *CVPR*, 2018. 2
- [17] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *ECCV*, 2016. 1, 2
- [18] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. *arXiv preprint arXiv:2103.14030*, 2021. 2, 3
- [19] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 3
- [20] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *CVPR*, 2016. 1, 2
- [21] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *CVPR*, 2017. 1
- [22] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018. 1
- [23] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *NeurIPS*, 2015. 1, 2
- [24] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014. 3
- [25] Mingxing Tan, Ruoming Pang, and Quoc V Le. Efficientdet: Scalable and efficient object detection. In *CVPR*, 2020. 2, 3
- [26] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. Fcos: Fully convolutional one-stage object detection. In *ICCV*, 2019. 1, 2
- [27] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017. 1
- [28] Chi Zhang, Feng Gao, Baoxiong Jia, Yixin Zhu, and Song-Chun Zhu. Raven: A dataset for relational and analogical visual reasoning. In *CVPR*, 2019. 4
- [29] Chi Zhang, Baoxiong Jia, Mark Edmonds, Song-Chun Zhu, and Yixin Zhu. Acre: Abstract causal reasoning beyond co-variation. In *CVPR*, 2021. 4
- [30] Chi Zhang, Baoxiong Jia, Feng Gao, Yixin Zhu, Hongjing Lu, and Song-Chun Zhu. Learning perceptual inference by contrasting. *NeurIPS*, 2019. 4
- [31] Chi Zhang, Baoxiong Jia, Song-Chun Zhu, and Yixin Zhu. Abstract spatial-temporal reasoning via probabilistic abduction and execution. In *CVPR*, 2021. 4
- [32] Chi Zhang, Sirui Xie, Baoxiong Jia, Ying Nian Wu, Song-Chun Zhu, and Yixin Zhu. Learning algebraic representation for systematic generalization in abstract reasoning. *arXiv preprint arXiv:2111.12990*, 2021. 4
- [33] Wenhe Zhang, Chi Zhang, Yixin Zhu, and Song-Chun Zhu. Machine number sense: A dataset of visual arithmetic problems for abstract and relational reasoning. In *AAAI*, 2020. 4
- [34] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as points. *arXiv preprint arXiv:1904.07850*, 2019. 1, 2
- [35] Yixin Zhu, Tao Gao, Lifeng Fan, Siyuan Huang, Mark Edmonds, Hangxin Liu, Feng Gao, Chi Zhang, Siyuan Qi,

Ying Nian Wu, et al. Dark, beyond deep: A paradigm shift to cognitive ai with humanlike common sense. *Engineering*, 6(3):310–345, 2020. [4](#)