AdaMixer: A Fast-Converging Query-Based Object Detector

Ziteng Gao[1] Limin Wang[1]✉ Bing Han[2] Sheng Guo[2]
[1]State Key Laboratory for Novel Software Technology, Nanjing University, China
[2]MYbank, Ant Group, China

Abstract

Traditional object detectors employ the dense paradigm of scanning over locations and scales in an image. The recent query-based object detectors break this convention by decoding image features with a set of learnable queries. However, this paradigm still suffers from slow convergence, limited performance, and design complexity of extra networks between backbone and decoder. In this paper, we find that the key to these issues is the adaptability of decoders for casting queries to varying objects. Accordingly, we propose a fast-converging query-based detector, named AdaMixer, by improving the adaptability of query-based decoding processes in two aspects. First, each query adaptively samples features over space and scales based on estimated offsets, which allows AdaMixer to efficiently attend to the coherent regions of objects. Then, we dynamically decode these sampled features with an adaptive MLP-Mixer under the guidance of each query. Thanks to these two critical designs, AdaMixer enjoys architectural simplicity without requiring dense attentional encoders or explicit pyramid networks. On the challenging MS COCO benchmark, AdaMixer with ResNet-50 as the backbone, with 12 training epochs, reaches up to 45.0 AP on the validation set along with 27.9 AP_s in detecting small objects. With the longer training scheme, AdaMixer with ResNeXt-101-DCN and Swin-S reaches 49.5 and 51.3 AP. Our work sheds light on a simple, accurate, and fast converging architecture for query-based object detectors. The code is made available at https://github.com/MCG-NJU/AdaMixer.

1. Introduction

Object detection has been a fundamental task in the computer vision area for decades, as it aims to locate varying objects in a single image and categorize them. For a long time, researchers have used spatial dense prior on grids in an image to cover potential objects with great variations.

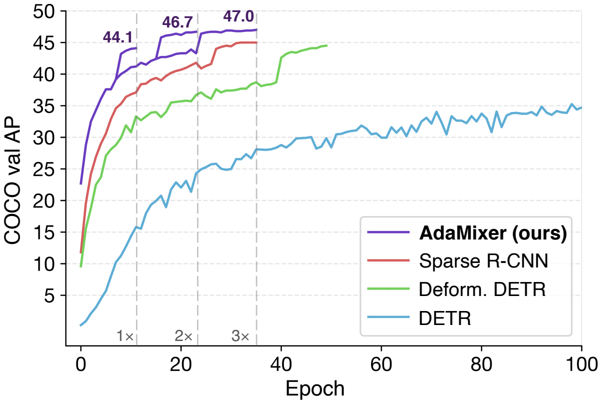

Figure 1. **Convergence curves** of our AdaMixer, DETR, Deformable DETR and Sparse R-CNN with ResNet-50 as the backbone on MS COCO minival set.

This dense paradigm dates back from sliding window methods [11, 36, 41] and still remains prevalent in anchor-based or point-based detectors [15, 23, 26, 32, 34, 38, 50] at the age of convolutional neural networks. Although dense priors has been a dominant role in object detection for its remarkable performance to cover potential objects, they are criticized for several shortcomings in various aspects, including anchor designs [33, 42, 45], training sample selection [13, 14, 48, 49], and post-processing operators on potential redundant detections [2, 18].

Though sorts of remedies on these issues are proposed every year, the underlying detection scheme with dense grid-like prior had remained almost untouched for a long time. Recently, query-based object detectors [4, 37, 53] bring a new perspective on object detection, that is, to use learnable embeddings, also termed queries, to directly represent potential objects by using attention-like operators [40]. This scheme, on the other hand, requires a strong representation power of the network to cast limited queries to potential objects to cope with great variations of objects across images. However, the adaptability of currently employed query decoders to the image content is limited on both how to spatially sample features and how to process

✉: Corresponding author (lmwang@nju.edu.cn).

sampled features. For instance, attention-based decoder in DETR-like detectors [4, 53] are adaptive on which feature to sample but remain static on how to decode it, whereas dynamic interaction head in Sparse R-CNN [37] goes vice versa. The insufficiency in the adaptability to different images leaves the current decoder in a dilemma between limited query representation power and great variations of objects. Also, as compensation for this, query-based object detectors usually bring extra attentional encoders or explicit pyramid necks after the backbone and before the query decoder, in order to involve more semantic or multi-scale modeling, such as TransformerEncoder [40], MultiScaleDeformableTransformerEncoder [53] and FPN [22]. These extra components result in the higher complexity of built detection pipelines in both design and computational aspects. In addition, detectors with them are hungry for more training time and rich data augmentation due to the introduced modules.

In this paper, we present a fast-converging and accurate query-based object detector with a simplified architecture, named AdaMixer, to mitigate issues above. Specifically, to effectively use queries to represent objects, AdaMixer introduces the adaptive 3D feature sampler and the adaptive mixing of channel semantics and spatial structures holistically. First, by regarding multi-scale feature maps from the backbone as a 3D feature space, our proposed decoder can flexibly sample features over space and scales to adaptively handle both of location and scale variations of objects based on queries. Then, the adaptive mixing applies the channel and spatial mixing to the sampled features with dynamic kernels under the guidance of queries. The adaptive location sampling and holistic content decoding notably enhances the adaptability of queries to varying images in detecting varying objects. As a result, AdaMixer is simply made up of a backbone network and our proposed decoder without extra attentional encoders or explicit pyramid networks.

Experimental results show that in a standard 12 epochs training (1× training scheme) with the random flipping as the only augmentation, our AdaMixer with ResNet-50 [17] as the backbone achieves 42.7, 44.1, and 45.0 AP on MS COCO validation set under the settings of 100, 300, and 500 queries, with 24.7, 27.0, and 27.9 AP_s in small object detection. With longer 3× training time and stronger data augmentation aligned to other query-based detectors, our AdaMixer with ResNet-101, ResNeXt-101-DCN [44, 52], and Swin-S [27] achieves 48.0, 49.5, and 51.3 AP with the single scale and single model testing, significantly outperforming the previous state-of-the-art query-based detectors. We hope AdaMixer, as a simply-designed, fast-converging, relatively efficient, and more accurate object detector, will serve as a strong baseline in the future research for the query-based object detection.

2. Related Work

Dense object detectors. The dense paradigm of object detectors dates back to sliding window-based approaches [10, 11, 41], which involve exhaustive classification over space and scales due to the assumption of potential objects emerging uniformly and densely w.r.t. spatial locations in an image. This assumption about natural images remains effective in the deep learning era for its power to cover potential objects [36]. Prevalent object detectors in the past few years, *e.g.*, one-stage detectors [26, 32, 33, 38, 47, 50], multiple-stage detectors [3, 5, 16, 30, 34] or point-based methods [9, 20, 50, 51], are also rooted in this dense assumption in either region proposal networks or entire object detector architectures. They apply dense priors, such as anchors or anchor points, upon the feature map to exhaustively find foreground objects or directly classify them.

Query-based object detectors. Recently transformer-based detector, DETR [4], formulates object detection as a direct set prediction task and achieves promising performance. DETR predicts a set of objects by attending queries to the feature map with the transformer decoder. The original architecture of DETR is simply based on the Transformer [40], which contains the multi-layer attentional encoder and decoder. The training for set prediction in DETR is based on the bipartite matching between the predictions and ground-truth objects. While DETR outperforms competitive Faster R-CNN baselines, it still suffers from limited spatial resolution, poor small object detection performance, and slow training convergence. There have been several work to tackle these issues. Deformable DETR [53] considers the shift-equivalence in natural images and introduces the multi-scale deformable family of attention operators in both encoders and decoders of DETR. SMCA [12], Conditional DETR [29], and Anchor DETR [43] explicitly model the positional attention for foreground objects to fasten the convergence. Efficient DETR [46] bridges the dense prior to queries in DETR to improve the performance. Sparse R-CNN [37] brings the query-based paradigm of DETR to Cascade R-CNN [3] and introduces the dynamic instance interaction head and its query-adaptive point-wise convolution, to effectively cast queries to potential objects.

Our AdaMixer generally follows this research line of using queries to attend features for object detection. However, we improve the query-based object detection paradigm from a new perspective: the adaptability of decoding queries across images. Specifically, we focus on how to make decoding scheme on queries more adaptive to the content of images from both semantic and spatial aspects. We present adaptive 3D feature sampling and adaptive content decoding to improve its flexibility to relate queries with each image. This makes AdaMixer a fast-converging query-based object detector without the introduction of extra feature encoders or explicit pyramid networks.

	adaptive to decode locations?	adaptive to decode content?	extra networks before the query decoder[1]?
DETR [4]	*yes*, multi-head attention aggregation	*no*, linear projection	TransformerEncoder
Deformable DETR [53]	*yes*, multi-scale multi-head adaptive sampling	*no*, linear projection[2]	Multi-scale DeformTransEncoder
Sparse R-CNN [37]	*restricted*, RoIAlign [16]	*partially yes*, adaptive point-wise conv.	FPN
AdaMixer (ours)	*yes*, adaptive 3D sampling	*yes*, adaptive channel and spatial mixing	linear projection to form 3D feature space

Table 1. **Comparisons of the adaptability of decoders** across different query-based object detectors. [1]We specify trainable networks introduced after pre-trained backbones before the query decoder. [2]We regard the softmax aggregation in deformable attention as one step in decoding locations as the softmax weights normalize to one.

3. Approach

In this paper, we focus on the query decoder in query-based object detectors since the decoder design is essential to casting learned queries to potential objects in each image. We first revisit decoders in popular query-based object detectors from the perspective of semantic and positional adaptability, and then elaborate on our proposed adaptive query decoder.

3.1. Object Query Decoder Revisited

Plain attention decoders. DETR [4] applies plain multi-head cross attention between queries and features to cast object queries to potential objects. As depicted in Table 1, the cross attention decoder is adaptive to decode sampling locations in the sense that it exploits the relation of object queries and features to aggregate features. However, the linear transformation of features after aggregation fails to adaptively decode them based on the query.

Deformable multi-scale attention decoders. Deformable DETR [53] improves the ability of decoding sampling locations in plain cross attention in terms of shift equivalence and scale invariance by introducing explicit reference points and multi-scale features. But like DETR, the content decoding of sampled features still remains static by the linear transformation. Overall, decoders in DETR and Deformable DETR lack the reasoning of aggregated features conditionally on the query and thus limit the semantic adaptability of queries to features. As a result, both of them require stacks of extra attentional encoders to enrich feature semantics.

RoIAlign and dynamic interactive head as decoders. Sparse R-CNN [37], as the intersection between region-based and query-based detectors, uses the RoIAlign operator and dynamic interactive head as the query decoder. The dynamic interactive head uses point-wise convolutions, whose kernel is adaptive based on the query, to process RoI features. This enables the adaptability of queries to RoI features but only partially, in the sense that the adaptive point-wise convolution can not infer adaptive spatial structures from those features to build queries. Moreover, the sampling locations by RoIAlign operator [16] are restricted inside of the box indicated by a query and a specific level in FPN [22], which limits positional adaptability and requires explicit pyramid networks for multi-scale modeling.

Summary. Given a limited number of queries and varying potential objects across images, an ideal decoder should consider both the semantic and positional adaptability of such queries to the content of images, that is, how to adaptively decode sampling locations and sampled content. This naturally motivates our design of AdaMixer.

3.2. Our Object Query Definition

Starting from the object query definition, we associate two vectors with a query following our semantic and positional view of decoders: one is the content vector $\mathbf{q}$ and the other is the positional vector (x, y, z, r). This is also in line with [37, 43, 53] to disentangle the location or the represented bounding box of a query from its content. The content vector $\mathbf{q}$ is a vector in $\mathbb{R}^{d_q}$ and d_q is the channel dimension. The vector (x, y, z, r) describes scaled geometric properties of the bounding box indicated by a query, that is, the x- and y-axis coordinates of its center point and the logarithm of its scale and aspect ratio. The x, y, z components also directly represent coordinates of a query in the 3D feature space, which will be introduced below.

Decoding the bounding box from a query. We can simply decode the bounding box from the positional vector. The center (x_B, y_B) and the width and height w_B and h_B of the indicated bounding box can be decoded:

$$x_B = s_{\text{base}} \cdot x, \qquad y_B = s_{\text{base}} \cdot y, \tag{1}$$

$$w_B = s_{\text{base}} \cdot 2^{z-r}, h_B = s_{\text{base}} \cdot 2^{z+r}, \tag{2}$$

where s_{base} is the base downsampling stride offset and we set $s_{\text{base}} = 4$ according to the stride of the largest feature map we use in the experiments.

3.3. Adaptive Location Sampling

As discussed in Section 3.1, the decoder should adaptively decide which feature to sample regarding the query. That is, the decoder should decode sampling locations with the consideration of both the positional vector (x, y, z, r) and content vector $\mathbf{q}$. Also, we argue that the decoder must be adaptive not only over (x, y) space but also be flexible in scales of potential objects. Specifically, we can accomplish these goals by regarding multi-scale features as a 3D feature space and adaptively sampling features from it.

Multi-scale features as the 3D feature space. Given a feature map, indexed j, with the downsampling stride s_j^{feat}

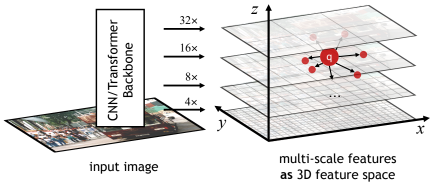

Figure 2. **3D feature sampling process.** A query first obtains sampling points in the 3D feature space and then perform 3D interpolation on these sampling points.

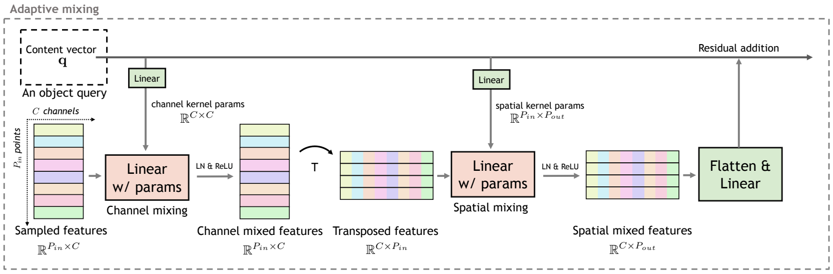
Adaptive mixing
Content vector $\mathbf{q}$
An object query
Residual addition
Linear
channel kernel params $\mathbb{R}^{C\times C}$
Linear
spatial kernel params $\mathbb{R}^{P_{in}\times P_{out}}$
C channels
P_{in} points
Linear w/ params
Channel mixing
LN & ReLU
T
Linear w/ params
Spatial mixing
LN & ReLU
Flatten & Linear
Sampled features $\mathbb{R}^{P_{in}\times C}$
Channel mixed features $\mathbb{R}^{P_{in}\times C}$
Transposed features $\mathbb{R}^{C\times P_{in}}$
Spatial mixed features $\mathbb{R}^{C\times P_{out}}$

Figure 3. **Adaptive mixing procedure** between an object query and sampled features. The object query first generates adaptive mixing weights and then apply these weights to mix sampled features in the channel and spatial dimension. Note that for clarity, we demonstrate adaptive mixing for one sampling group.

from the backbone, we first transform them by a linear layer to the same channel d_{feat} and compute its z-axis coordinate:

$$z_j^{\text{feat}} = \log_2(s_j^{\text{feat}}/s_{\text{base}}). \tag{3}$$

Then we virtually rescale the height and width of feature maps of different strides to the same ones H/s_{base} and W/s_{base}, where H and W is the height and width of the input image, and put them aligned on x- and y-axis in the 3D space as depicted in Figure 2. These feature maps are supporting planes for the 3D feature space, whose interpolation is described below.

Adaptive 3D feature sampling process. A query first generate P_{in} sets of offset vectors to P_{in} points, $\{(\Delta x_i, \Delta y_i, \Delta z_i)\}_{P_{\text{in}}}$, where each offset vector is indexed by i, and depends on its content vector $\mathbf{q}$ by a linear layer:

$$\{(\Delta x_i, \Delta y_i, \Delta z_i)\}_{P_{\text{in}}} = \text{Linear}(\mathbf{q}). \tag{4}$$

Then, these offsets are transformed to sampling locations according to the positional vector of the query for every i:

$$\begin{cases} \tilde{x}_i = x + \Delta x_i \cdot 2^{z-r}, \\ \tilde{y}_i = y + \Delta y_i \cdot 2^{z+r}, \\ \tilde{z}_i = z + \Delta z_i, \end{cases} \tag{5}$$

It is worth noting that the area $\{\Delta x_i, \Delta y_i \in [-0.5, 0.5]\}$ describes the bounding box decoded from the query. Our offsets are not restricted to this range, meaning that a query can sample features "out of the box". Having obtained $\{(\tilde{x}_i, \tilde{y}_i, \tilde{z}_i)\}_{P_{\text{in}}}$, our sampler samples values given these points in the 3D space. In the current implementation, the interpolation over the 3D space is in the compositional manner: it first samples values given points by bilinear interpolation in the (x, y) space and then interpolates over the z-axis by gaussian weighting given a sampling $\tilde{z}$, where the weight for the j-th feature map is:

$$\tilde{w}_j = \frac{\exp(-(\tilde{z} - z_j^{\text{feat}})^2/\tau_z)}{\sum_j \exp(-(\tilde{z} - z_j^{\text{feat}})^2/\tau_z)}, \tag{6}$$

where τ_z is the softing coefficient for interpolating values over the z-axis and we keep $\tau_z = 2$ in this work. With the feature map of the channel d_{feat}, the shape of sampled feature matrix $\mathbf{x}$ is $\mathbb{R}^{P_{\text{in}} \times d_{\text{feat}}}$. The adaptive 3D feature sampling process eases the decoder learning by sampling features with explicit, adaptive and coherent locations and scales regarding a query.

Group sampling. To sample as many points as possible, we introduce the group sampling mechanism, analogous to multiple heads in attentional operators [40] or groups in group convolution [44]. The group sampling first splits the channel d_{feat} of the 3D feature space into g groups, each with the channel d_{feat}/g, and performs 3D sampling individually for each group. With the group sampling mechanism, the decoder can generate $g \cdot P_{\text{in}}$ offset vectors for a query to enrich the diversity of sampling points and exploit richer spatial structure of these points. Sampled feature matrix $\mathbf{x}$ now are of the shape $\mathbb{R}^{g \times P_{\text{in}} \times (d_{\text{feat}}/g)}$. The grouping mechanism is also applied to the adaptive mixing for efficiency as described below, and we term the group sampling and mixing unified as the grouping mechanism.

3.4. Adaptive Content Decoding

With features sampled, *how to adaptively decode them* is another key design in our AdaMixer decoder. To capture correlation in spatial and channel dimension of $\mathbf{x}$, we propose to efficiently decode the content in each dimension separately. Specifically, we design a simplified and adaptive variant of MLP-mixer [39], termed as adaptive mixing, with dynamic mixing weights similar to dynamic filters in convolutions [19]. As shown in Figure 3, the procedure contains sequentially the adaptive channel mixing and adaptive spatial mixing to involve both adaptive *channel semantics* and *spatial structures* under the guidance of a query.

Adaptive channel mixing. Given sampled feature matrix $\mathbf{x} \in \mathbb{R}^{P_{\text{in}} \times C}$ for a query in a group, where $C = d_{\text{feat}}/g$, the adaptive channel mixing (ACM) is to use the dynamic weight based on $\mathbf{q}$ to transform features $\mathbf{x}$ on the channel

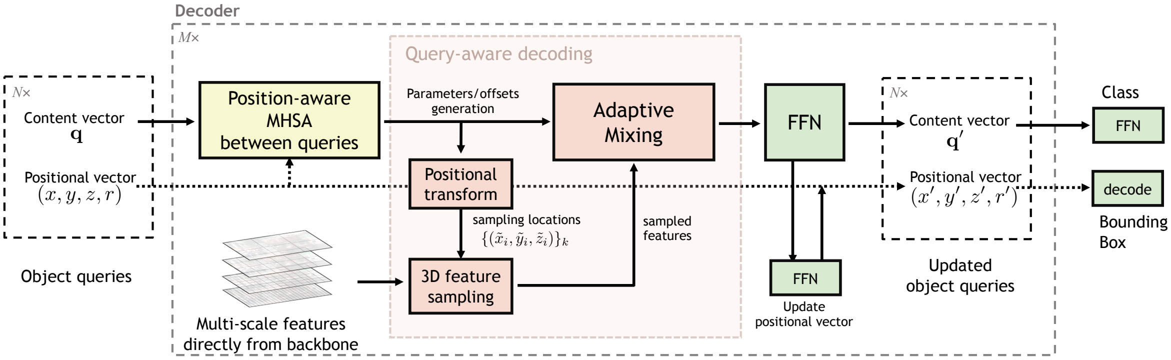

Figure 4. **Our decoder structure** of the AdaMixer. There are two operator streams on a query: one on its content vector $\mathbf{q}$ (the solid horizontal line) and one on its positional vector (x, y, z, r) (the dashed horizontal line). Each operator on the content vector in the decoder is followed by a residual addition and LayerNorm.

dimension to adaptively enhance channel semantics:

$$M_c = \text{Linear}(\mathbf{q}) \in \mathbb{R}^{C\times C} \tag{7}$$

$$\text{ACM}(\mathbf{x}) = \text{ReLU}(\text{LayerNorm}(\mathbf{x}M_c)), \tag{8}$$

where $\text{ACM}(\mathbf{x}) \in \mathbb{R}^{P_{\text{in}}\times C}$ is the channel mixed feature output and the linear layer is individual for each group. The layer normalization [1] is applied to both dimensions of the mixed output. Note that in this step, the dynamic weight is *shared* across *different sampling points* in 3D space, analogous to adaptive 1×1 convolution in [37] on RoI features.

Adaptive spatial mixing. To enable the adaptability of a query to spatial structures of sampled features, we introduce the adaptive spatial mixing (ASM) process. As depicted in Figure 3, ASM can be described as firstly transposing channel mixed feature matrix and applying the dynamic kernel to the spatial dimension of it:

$$M_s = \text{Linear}(\mathbf{q}) \in \mathbb{R}^{P_{\text{in}}\times P_{\text{out}}} \tag{9}$$

$$\text{ASM}(\mathbf{x}) = \text{ReLU}(\text{LayerNorm}(\mathbf{x}^T M_s)), \tag{10}$$

where $\text{ASM}(\mathbf{x}) \in \mathbb{R}^{C\times P_{\text{out}}}$ is the spatial mixed output and P_{out} is the number of spatial mixing out patterns. Note that the dynamic weight is *shared* across *different channels*. As sampling points may be from different feature scales, ASM naturally involves multi-scale interaction modeling, which is necessary for high performance object detection.

The adaptive mixing procedure overall is depicted in Figure 3, where the adaptive spatial mixing follows the adaptive channel mixing, both applied in a sampling group. The final output of the shape $\mathbb{R}^{g\times C\times P_{\text{out}}}$ across group is flattened and transformed to the d_q dimension by a linear layer to add back to the content vector.

3.5. Overall AdaMixer Detector

Like the decoder architecture in [4, 53], we place the self-attention between queries, our proposed adaptive mixing, and feedforward-feed network (FFN) sequentially in a stage of the decoder regarding the query content vector $\mathbf{q}$, as shown in Figure 4. The query positional vector is updated by another FFN at the end of each stage:

$$x' = x + \Delta x \cdot 2^z, y' = y + \Delta y \cdot 2^z, \tag{11}$$

$$z' = z + \Delta z, \qquad r' = r + \Delta r, \tag{12}$$

where $(\Delta x, \Delta y, \Delta z, \Delta r)$ is produced by the lower small FFN block in Figure 4.

Position-aware multi-head self-attentions. Since we disentangle the content and position for a query, the naive multi-head self-attention between the content vectors of queries is not aware of *what geometric relation* between a query and another query is, which is proven beneficial to suppress redundant detections [4]. To achieve this, we embed positional information into the self-attention. Our positional embedding for the content vector in the sinusoidal form and every component of (x, y, z, r) takes up a quarter of channels. We also embed the intersection over foreground (IoF) as a bias to the attention weight between queries to explicitly incorporate the relation of being contained between queries. The attention for each head is

$$\text{Attn}(Q, K, V) = \text{Softmax}\left(QK^T/\sqrt{d_q} + \alpha B\right)V, \tag{13}$$

where $B_{ij} = \log\left(|\text{box}_i \cap \text{box}_j|/|\text{box}_i| + \epsilon\right)$, $\epsilon = 10^{-7}$, $Q, K, V \in \mathbb{R}^{N\times d_q}$, standing for the query, key and value matrix in the self-attention procedure, and α is a learnable scalar for each head. The $B_{ij} = 0$ stands for the box i being totally contained in the box j and $B_{ij} = \log \epsilon \ll 0$ indicates that there is no overlapping between i and j.

Overall AdaMixer detector. The detection pipeline of AdaMixer is only composed of a backbone and a AdaMixer decoder. It avoids adding explicit feature pyramid networks or attentional encoders between backbone and decoder. The AdaMixer directly gathers predictions of decoded queries as final object detection results.

4. Experiments

In this section, we first elaborate on the implementation and training details. Then we compare our models with other competitive detectors with limited training epochs. Next, we perform ablation studies on the design of our detector. We also align the training recipe to other query-based detectors and compare our AdaMixer to them fairly.

4.1. Implementation Details

Dataset. We conduct extensive experiments on MS COCO 2017 dataset [24] in mmdetection codebase [6]. Following the common practice, we use `trainval35k` subset consisting up of 118K images to train our models and use `minival` subset of 5K images as the validation set.

query dim d_q	feat. maps used	feat. dim d_{feat}	#stages in decoder	#groups g	C	P_{in}	P_{out}
256	$C_2 \sim C_5$	256	6	4	64	32	128

Table 2. **Default configuration** of our AdaMixer detector.

Configurations. The default hyper-parameters in our AdaMixer detector is elaborated in Table 2. We configure the dimension of the query content vector d_q to 256 following previous query-based work [4, 37, 53]. We use feature maps $C_2 \sim C_5$ from the backbone network. Multi-scale features are processed by the linear transformation to the channel $d_{\text{feat}} = 256$ as supporting planes for the 3D feature space. The number of decoder stages is set to 6 also following the common practice of query-based detectors. The mixer grouping number is set to 4 as default. Accordingly, the channel of sampled features per group is $C = d_{\text{feat}}/g = 64$. Also, the number of sampling points P_{in} and mixing out patterns P_{out} per group is set to 32 and 128. The hidden dimension of FFN on the content vector in the decoder is set to 2048. The FFN dimension for classification and updating positional vectors is set to 256.

Initializations. For training stability in early iterations, we initialize the parameters of linear layers to produce dynamic parameters or sampling offsets as follows: zeroing weights of these layers and initializing biases as expected. This helps stable training by enforcing models to learn the adaptability from zeros. The biases for the linear layer producing sampling offest vectors is initialized in the way that $\Delta x_i, \Delta y_i$ are uniformly drawn in $[-0.5, 0.5]$ and $\Delta z_i = -1$ for all i to align with the RoIAlign [16] level strategy. The bias in the linear layer to produce mixing weights follows the default initialization in the PyTorch. We also initialize all the query positional vectors into decoders such that the their boxes and sampling points cover the whole image in the initial decoder stage like [37]. Backbones are initialized from pre-trained models on the ImageNet 1K dataset [8].

Losses and optimizers. Following [37, 53], the training loss is the matching loss consisting of the focal loss [23] with coefficient $\lambda_{\text{cls}} = 2$, L1 bounding box loss with $\lambda_{L_1} = 5$ and GIoU loss [35] with $\lambda_{\text{giou}} = 2$. We use AdamW [28] as our optimizer with weight decay 0.0001. The initial learning rate is 2.5×10^{-5}.

detector	epochs	AP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l
FCOS [38]	12	38.7	57.4	41.8	22.9	42.5	50.1
Cascade R-CNN [3]	12	40.4	58.9	44.1	22.8	43.7	54.0
GFocalV2 [21]	12	41.1	58.8	44.9	23.5	44.9	53.3
BorderDet [31]	12	41.4	59.4	44.5	23.6	45.1	54.6
Dynamic Head [7]	12	42.6	60.1	**46.4**	**26.1**	**46.8**	56.0
DETR [4]	12	20.0	36.2	19.3	6.0	20.5	32.2
Deformable DETR [53]	12	35.1	53.6	37.7	18.2	38.5	48.7
Sparse R-CNN [37]	12	37.9	56.0	40.5	20.7	40.0	53.5
AdaMixer (N=100)	12	**42.7**	**61.5**	45.9	24.7	45.4	**59.2**
AdaMixer (N=300)	12	**44.1**	**63.4**	**47.4**	**27.0**	**46.9**	**59.5**
AdaMixer (N=500)	12	**45.0**	**64.2**	**48.6**	**27.9**	**47.8**	**61.1**

Table 3. **1× training scheme** performance on COCO `minival` set with different detectors and ResNet-50 as backbone.

Training recipes. We adopt two versions of training recipes for a fair comparison with different detectors. The first one adopts the classic **1× training scheme**, which includes a budget of 12 training epochs with training images of shorter side resized to 800. This recipe includes the random horizontal flipping as only the standard data augmentation and allocates 100 learnable object queries to our AdaMixer detector, to compare with popular and competitive detectors like FCOS [38] and Cascade R-CNN [3] fairly. The second training recipe is to align with other query-based detectors, which leverages more training epochs and performs crop and multi-scale augmentation in [4,37,53]. Our second training recipe adopts the same data augmentation and has a budget of 36 training epochs, namely **3× training scheme**. It uses 300 object queries to compare fairly with [37, 53]. The learning rate is divided by a factor of 10 at epoch 8 and 11 in 1× training scheme or at epoch 24 and 33 in 3× training scheme, scaled proportionally.

The 1× and 3× training scheme for AdaMixer with ResNet-50 typically take about 9 and 29 hours on 8 V100 cards. During the inference stage, we input images of the shorter size resized to 800 without data augmentation. We leave more details about model training and inference and visualizations in the supplementary material.

4.2. Fast Convergence with Limited Budgets

We first investigate our proposed AdaMixer with limited training epochs and limited data augmentation, namely 1× training scheme. For a fair comparison, we disable the commonly-used crop and multi-scale data augmentation in query-based detectors and allocate only 100 queries or learnable proposals for these detectors. Experimental results are shown in Table 3. AdaMixer with $N = 100$ queries achieves 42.7 AP, outperforming state-of-the-art traditional and query-based detectors with a limited training budget. Moreover, if we increase the number of queries N to 300 and 500, the performance of the AdaMixer detector reaches 44.1 and 45.0 AP, especially with 27.0 and 27.9 AP_s in de-

adaptive loc.	cont.	AP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l
		35.7	55.2	37.8	20.1	38.1	48.8
✓		37.3	55.8	39.7	20.7	40.1	50.9
	✓	40.4	60.5	43.4	23.0	42.5	56.7
✓	✓	**42.7**	**61.5**	**45.9**	**24.7**	**45.4**	**59.2**

(a) **Adaptability of decoding** sampling locations and sampled content.

mixing		AP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l
ACM	ACM	41.5	60.5	44.3	23.5	44.1	57.4
ASM	ASM	39.8	58.8	42.6	22.8	42.4	56.1
ACM	ASM	**42.7**	**61.5**	**45.9**	**24.7**	**45.4**	**59.2**
ASM	ACM	41.5	60.4	44.5	23.9	44.4	57.1

(b) **Design** in our adaptive mixing procedure.

pyramid	AP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l
FPN [22]	42.1	61.0	45.0	24.1	44.8	58.7
PAFPN [25]	41.7	60.5	44.7	23.5	44.6	58.7
-	**42.7**	**61.5**	**45.9**	**24.7**	**45.4**	**59.2**

(c) **Extra pyramid networks** after the backbone?

P_{in}	AP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l
8	41.2	60.3	44.1	24.0	43.9	57.2
16	41.8	60.9	44.5	24.5	44.6	58.4
32	**42.7**	**61.5**	45.9	24.7	45.4	59.2
64	**42.7**	**61.5**	**46.1**	**24.9**	**45.5**	**59.3**

(d) **Sampling points** P_{in} per group.

P_{out}	AP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l
32	41.1	60.0	44.0	24.5	43.6	57.2
64	42.1	61.2	45.0	24.0	44.8	57.8
128	**42.7**	**61.5**	**45.9**	**24.7**	**45.4**	**59.2**
256	42.4	61.4	45.5	24.4	45.0	58.7

(e) **Spatial mixing out patterns** P_{out} per group.

pos. inf. sinus.	IoF	AP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l
		41.2	59.6	44.2	23.6	43.5	57.9
✓		41.5	59.9	44.3	23.6	44.0	57.8
	✓	42.2	61.2	45.0	**24.8**	45.1	58.8
✓	✓	**42.7**	**61.5**	**45.9**	24.7	**45.4**	**59.2**

(f) **Position information** in self-attention between queries.

Table 4. **AdaMixer ablation experiments** with ResNet-50 on MS COCO `minival` set. Default choice for our model is colored gray

tecting small objects. It is worth noting that these results are achieved with random flipping as the only data augmentation and within 12 training epochs, showing that AdaMixer can be supervised efficiently with training samples.

4.3. Ablation Studies

Due to the limited computational resource, we use ResNet-50 as the backbone network and 1× training scheme to perform ablation studies.

Decoding adaptability. We begin our ablations with the key component, the adaptability design, in our AdaMixer detector. The adaptability in AdaMixer is in two aspects: adaptive sampling for decoding locations and adaptive mixing for decoding content. Table 4a investigates the performance under the condition of whether or not we enable the adaptability in decoding sampling locations and decoding content. The cancellation for adaptability on locations or content stands for enforcing weights of linear layers producing sampling offsets or mixing weights all to zeros during the training and inference. Only biases of these layers can be learned during the training procedure, which are eventually not adaptive based on the query content $\mathbf{q}$. In other words, all sampling offsets or mixing weights are the same across different queries and different images with the cancellation. As shown in Table 4a, the adaptability in both decoding sampling locations and sampled content is essential to a good query-based object detector, which outperforms a non-adaptive counterpart by 7.0 AP.

Adaptive mixing design. Moving forward, we compare different designs of our adaptive mixing in Table 4b. As shown in Figure 3, our default design for adaptive mixing is to mix features first on the channel dimension and then on the spatial dimension. We perform ablations by placing only channel mixing, only spatial mixing, and the reversed order of our design as three variants. The first adaptive channel mixing and then spatial one lead to the best performance. This indicates that channel semantics and spatial structures are both important to the mixing design. For the reversed mixing variant, we suspect that the inferior result is due to the insufficient channel semantics into spatial mixing as features are directly from the backbone.

Extra pyramid networks. AdaMixer enjoys the simplicity for circumventing extra attentional encoders or explicit pyramid networks. Instead, AdaMixer improves the semantic and multi-scale modeling in the decoder. The adaptive 3D sampling and following spatial mixing naturally enable multi-scale feature modeling and enable queries to handle scale variations of objects. In Table 4c, we investigate the performance of the AdaMixer detector with introduction of extra pyramid networks. Models with these extra networks might require a longer training time and more training samples to perform well. These results are in favor of our AdaMixer design as a simplified query-based detector.

Sampling points and spatial mixing out patterns. Table 4d and 4e shows the ablation on the sampling points P_{in} and spatial mixing out patterns P_{out} per group. The performance is generally related to the number of sampling points P_{in} and spatial mixing out patterns P_{out}. A good balance between the complexity and performance is $P_{\text{in}} = 32$ and $P_{\text{out}} = 128$, where the performance saturates for P_{in} and decreases for P_{out} beyond this point.

Positional information in attention between queries. In Section 3.5, we propose to embed the positional information into the self-attention between the content vectors of queries. In addition to the regular sinusoidal positional embedding, we also hardwire the intersection over foreground (IoF) into the attention weight between boxes indicated by queries. We investigate these two ingredients in Table 4f. Results show that combining these two ingredients notably increases the performance. The individual effect of the IoF

detector	backbone	encoder/pyramid net	#epochs	GFLOPs	AP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l
DETR [4]	ResNet-50-DC5	TransformerEnc	500	187	43.3	63.1	45.9	22.5	47.3	61.1
SMCA [12]	ResNet-50	TransformerEnc	50	152	43.7	63.6	47.2	24.2	47.0	60.4
Deformable DETR [53]	ResNet-50	DeformTransEnc	50	173	43.8	62.6	47.7	26.4	47.1	58.0
Sparse R-CNN [37]	ResNet-50	FPN	**36**	174	45.0	63.4	48.2	26.9	47.2	59.5
Efficient DETR [46]	ResNet-50	DeformTransEnc	**36**	210	45.1	63.1	49.1	28.3	48.4	59.0
Conditional DETR [29]	ResNet-50-DC5	TransformerEnc	108	195	45.1	65.4	48.5	25.3	49.0	**62.2**
Anchor DETR [43]	ResNet-50-DC5	DecoupTransEnc	50	151	44.2	64.7	47.5	24.7	48.2	60.6
AdaMixer (ours)	ResNet-50	-	**12**	132	44.1	63.1	47.8	29.5	47.0	58.8
AdaMixer (ours)	ResNet-50	-	**24**	132	46.7	65.9	50.5	29.7	49.7	61.5
AdaMixer (ours)	ResNet-50	-	**36**	132	**47.0**	**66.0**	**51.1**	**30.1**	**50.2**	61.8
DETR [4]	ResNet-101-DC5	TransformerEnc	500	253	44.9	64.7	47.7	23.7	49.5	62.3
SMCA [12]	ResNet-101	TransformerEnc	50	218	44.4	65.2	48.0	24.3	48.5	61.0
Sparse R-CNN [37]	ResNet-101	FPN	**36**	250	46.4	64.6	49.5	28.3	48.3	61.6
Efficient DETR [46]	ResNet-101	DeformTransEnc	**36**	289	45.7	64.1	49.5	28.2	49.1	60.2
Conditional DETR [29]	ResNet-101-DC5	TransformerEnc	108	262	45.9	66.8	49.5	27.2	50.3	63.3
AdaMixer (ours)	ResNet-101	-	**36**	208	**48.0**	**67.0**	**52.4**	**30.0**	**51.2**	**63.7**
AdaMixer (ours)	ResNeXt-101-DCN	-	**36**	214	**49.5**	**68.9**	**53.9**	**31.3**	**52.3**	**66.3**
AdaMixer (ours)	Swin-S	-	**36**	234	**51.3**	**71.2**	**55.7**	**34.2**	**54.6**	**67.3**

Table 5. **Different query-based detector performance** on COCO `minival` set with **longer training scheme** and single scale testing.

is also compelling. We argue that the IoF between boxes, which describes the geometric relation of being contained directly for corresponding queries, is important for the self-attention to imitate the NMS procedure [4].

g	1	2	4	8
AP	42.5	**42.8**	42.7	41.9
FLOPs	111G	106G	**104G**	106G
Params	191M	148M	**135M**	149M

Table 6. **Grouping** sampling and mixing with g groups.

Mixer group number. The mixer grouping encourages the decoder sampler to sample more diverse points. This also reduces total parameters and computational costs by mixing divided groups of features. We here evaluate the effect of the grouping mechanism with various g in Table 6. The model reaches the least FLOPs and number of parameters with 4 mixer groups with promising performance.

4.4. Comparison with Other Query-based Detectors

We present the final results of our AdaMixer and perform the comparison between our AdaMixer and other state-of-the-art query-based detectors in Table 5. We use the 3× training scheme to train our AdaMixer, which allocates 300 queries and includes the stronger data augmentation to align with the common practice of other query-based methods. Specifically, we train our AdaMixer with ResNet-50, ResNet-101, ResNeXt-101 [44] with deformable convolution layers [52] and Swin-S [27] backbones. We also proportionally stretch the training schedule for AdaMixer with ResNet-50 to investigate the faster convergence speed, as depicted in Figure 1. AdaMixer with assorted backbones significantly outperforms competitive query-based object detectors with less computational cost. With bounding boxes as only supervising signals, AdaMixer with Swin-S reaches 51.3 AP and 34.2 AP_s with the single scale testing. Moreover, among these query-based detectors, only AdaMixer does not require the extra attentional encoders and explicit pyramid networks. These results demonstrate our AdaMixer is a simply-architected, effective, and fast-converging query-based object detector.

5. Conclusion and Limitation

In this paper, we have presented a fast-converging query-based object detection architecture, termed AdaMixer, to efficiently and effectively decode objects from images. Our proposed AdaMixer improves the decoder of query-based detectors with adaptive 3D sampling and adaptive channel and spatial mixing. By improving query decoders, AdaMixer circumvents the requirement of extra network modeling between backbone and decoder. Our AdaMixer achieves superior performance, especially on small object detection, with less computational cost compared to other query-based detectors. Moreover, it enables the fast convergence speed with limited training budgets. We hope that AdaMixer can serve as a strong baseline for future research.

The limitation in our AdaMixer is that though we have applied the grouping mechanism, the total parameter number remains a little bit large. This is mainly due to a large number of parameters in the linear layer to produce dynamic mixing weights. We leave the question of how to further reduce the number of parameters to the future work.

Acknowledgements. This work is supported by National Natural Science Foundation of China (No.62076119, No.61921006), Program for Innovative Talents and Entrepreneur in Jiangsu Province, and Collaborative Innovation Center of Novel Software Technology and Industrialization. Part of the work is done during the internship of Ziteng at MYbank.

References

[1] Lei Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *arXiv*, 2016. 5

[2] Navaneeth Bodla, Bharat Singh, Rama Chellappa, and Larry S. Davis. Soft-nms - improving object detection with one line of code. In *ICCV*, 2017. 1

[3] Zhaowei Cai and Nuno Vasconcelos. Cascade R-CNN: delving into high quality object detection. In *CVPR*, 2018. 2, 6

[4] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *ECCV*, 2020. 1, 2, 3, 5, 6, 8

[5] Kai Chen, Jiangmiao Pang, Jiaqi Wang, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jianping Shi, Wanli Ouyang, Chen Change Loy, and Dahua Lin. Hybrid task cascade for instance segmentation. In *CVPR*, 2019. 2

[6] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, Zheng Zhang, Dazhi Cheng, Chenchen Zhu, Tianheng Cheng, Qijie Zhao, Buyu Li, Xin Lu, Rui Zhu, Yue Wu, Jifeng Dai, Jingdong Wang, Jianping Shi, Wanli Ouyang, Chen Change Loy, and Dahua Lin. Mmdetection: Open mmlab detection toolbox and benchmark. In *arXiv*, 2019. 6

[7] Xiyang Dai, Yinpeng Chen, Bin Xiao, Dongdong Chen, Mengchen Liu, Lu Yuan, and Lei Zhang. Dynamic head: Unifying object detection heads with attentions. In *CVPR*, 2021. 6

[8] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Fei-Fei Li. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. 6

[9] Kaiwen Duan, Song Bai, Lingxi Xie, Honggang Qi, Qingming Huang, and Qi Tian. Centernet: Keypoint triplets for object detection. In *ICCV*, 2019. 2

[10] Pedro F. Felzenszwalb, Ross B. Girshick, and David A. McAllester. Cascade object detection with deformable part models. In *CVPR*, 2010. 2

[11] Pedro F. Felzenszwalb, Ross B. Girshick, David A. McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32(9):1627–1645, 2010. 1, 2

[12] Peng Gao, Minghang Zheng, Xiaogang Wang, Jifeng Dai, and Hongsheng Li. Fast convergence of DETR with spatially modulated co-attention. In *ICCV*, 2021. 2, 8

[13] Ziteng Gao, Limin Wang, and Gangshan Wu. Mutual supervision for dense object detection. In *ICCV*, 2021. 1

[14] Zheng Ge, Songtao Liu, Zeming Li, Osamu Yoshie, and Jian Sun. OTA: optimal transport assignment for object detection. In *CVPR*, 2021. 1

[15] Ross B. Girshick. Fast R-CNN. In *ICCV*, 2015. 1

[16] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask R-CNN. In *ICCV*, 2017. 2, 3, 6

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 2

[18] Yihui He, Chenchen Zhu, Jianren Wang, Marios Savvides, and Xiangyu Zhang. Bounding box regression with uncertainty for accurate object detection. In *CVPR*, 2019. 1

[19] Xu Jia, Bert De Brabandere, Tinne Tuytelaars, and Luc Van Gool. Dynamic filter networks. In *NIPS*, 2016. 4

[20] Hei Law and Jia Deng. Cornernet: Detecting objects as paired keypoints. In *ECCV*, 2018. 2

[21] Xiang Li, Wenhai Wang, Xiaolin Hu, Jun Li, Jinhui Tang, and Jian Yang. Generalized focal loss V2: learning reliable localization quality estimation for dense object detection. In *CVPR*, 2021. 6

[22] Tsung-Yi Lin, Piotr Dollár, Ross B. Girshick, Kaiming He, Bharath Hariharan, and Serge J. Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017. 2, 3, 7

[23] Tsung-Yi Lin, Priya Goyal, Ross B. Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *ICCV*, 2017. 1, 6

[24] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. In *ECCV*, 2014. 6

[25] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. Path aggregation network for instance segmentation. In *CVPR*, 2018. 7

[26] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott E. Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: single shot multibox detector. In *ECCV*, 2016. 1, 2

[27] 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. In *ICCV*, 2021. 2, 8

[28] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019. 6

[29] Depu Meng, Xiaokang Chen, Zejia Fan, Gang Zeng, Houqiang Li, Yuhui Yuan, Lei Sun, and Jingdong Wang. Conditional DETR for fast training convergence. In *ICCV*, 2021. 2, 8

[30] Jiangmiao Pang, Kai Chen, Jianping Shi, Huajun Feng, Wanli Ouyang, and Dahua Lin. Libra R-CNN: towards balanced learning for object detection. In *CVPR*, 2019. 2

[31] Han Qiu, Yuchen Ma, Zeming Li, Songtao Liu, and Jian Sun. Borderdet: Border feature for dense object detection. In *ECCV*, 2020. 6

[32] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *CVPR*, 2016. 1, 2

[33] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv*, 2018. 1, 2

[34] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. In *NIPS*, 2015. 1, 2

[35] Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian D. Reid, and Silvio Savarese. Generalized intersection over union: A metric and a loss for bounding box regression. In *CVPR*, 2019. 6

[36] Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. In *ICLR*, 2014. 1, 2

[37] Peize Sun, Rufeng Zhang, Yi Jiang, Tao Kong, Chenfeng Xu, Wei Zhan, Masayoshi Tomizuka, Lei Li, Zehuan Yuan, Changhu Wang, and Ping Luo. Sparse R-CNN: end-to-end object detection with learnable proposals. In *CVPR*, 2021. 1, 2, 3, 5, 6, 8

[38] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. FCOS: fully convolutional one-stage object detection. In *ICCV*, 2019. 1, 2, 6

[39] Ilya O. Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, Mario Lucic, and Alexey Dosovitskiy. Mlp-mixer: An all-mlp architecture for vision. In *NeurIPS*, 2021. 4

[40] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, 2017. 1, 2, 4

[41] Paul A. Viola and Michael J. Jones. Rapid object detection using a boosted cascade of simple features. In *CVPR*, 2001. 1, 2

[42] Jiaqi Wang, Kai Chen, Shuo Yang, Chen Change Loy, and Dahua Lin. Region proposal by guided anchoring. In *CVPR*, 2019. 1

[43] Yingming Wang, Xiangyu Zhang, Tong Yang, and Jian Sun. Anchor DETR: query design for transformer-based detector. *arXiv*, 2021. 2, 3, 8

[44] Saining Xie, Ross B. Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *CVPR*, 2017. 2, 4, 8

[45] Tong Yang, Xiangyu Zhang, Zeming Li, Wenqiang Zhang, and Jian Sun. Metaanchor: Learning to detect objects with customized anchors. In *NeurIPS*, 2018. 1

[46] Zhuyu Yao, Jiangbo Ai, Boxun Li, and Chi Zhang. Efficient DETR: improving end-to-end object detector with dense prior. *ArXiv*, 2021. 2, 8

[47] Jiahui Yu, Yuning Jiang, Zhangyang Wang, Zhimin Cao, and Thomas S. Huang. Unitbox: An advanced object detection network. In *ACM Multimedia*, 2016. 2

[48] Shifeng Zhang, Cheng Chi, Yongqiang Yao, Zhen Lei, and Stan Z. Li. Bridging the gap between anchor-based and anchor-free detection via adaptive training sample selection. In *CVPR*, 2020. 1

[49] Xiaosong Zhang, Fang Wan, Chang Liu, Rongrong Ji, and Qixiang Ye. Freeanchor: Learning to match anchors for visual object detection. In *NeurIPS*, 2019. 1

[50] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as points. *ArXiv*, 2019. 1, 2

[51] Xingyi Zhou, Jiacheng Zhuo, and Philipp Krähenbühl. Bottom-up object detection by grouping extreme and center points. In *CVPR*, 2019. 2

[52] Xizhou Zhu, Han Hu, Stephen Lin, and Jifeng Dai. Deformable convnets v2: More deformable, better results. In *CVPR*, 2019. 2, 8

[53] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable DETR: deformable transformers for end-to-end object detection. In *ICLR*, 2020. 1, 2, 3, 5, 6, 8