End-to-End Object Detection with Adaptive Clustering Transformer

Minghang Zheng[1] Peng Gao[3] Xiaogang Wang[3] Hongsheng Li[3] Hao Dong[1,2*]
[1]CFCS, CS Dept., Peking University
[2]AIIT, Peking University
[3]The Chinese University of Hong Kong
{minghang, hao.dong}@pku.edu.cn 1155102382@link.cuhk.edu.hk
{xgwang, hsli}@ee.cuhk.edu.hk

Abstract

End-to-end Object Detection with Transformer (DETR) proposes to perform object detection with Transformer and achieve comparable performance with two-stage object detection like Faster-RCNN. However, DETR needs huge computational resources for training and inference due to the high-resolution spatial input. In this paper, a novel variant of transformer named Adaptive Clustering Transformer (ACT) has been proposed to reduce the computation cost for high-resolution input. ACT cluster the query features ***adaptively*** *using Locality Sensitive Hashing (LSH) and approximate the query-key interaction using the prototype-key interaction. ACT can reduce the quadratic $O(N^2)$ complexity inside self-attention into $O(NK)$ where K is the number of prototypes in each layer. ACT can replace the original self-attention module in DETR without influencing the performance of pre-trained DETR model. ACT achieves a good balance between accuracy and computation cost (FLOPs). The code is available at supplementary for the ease of experiment replication and verification.*

1. Introduction

Object detection is the task of predicting a set of bounding boxes and category labels for each predetermined object. Recently popular models [36, 12, 13, 32, 31, 27] solve this task by generating a large number of regional proposals, predicting each proposal, and applying a non-maximum suppression procedure to eliminate those highly overlapping proposals. Two-stage object detection is difficult to deploy and debug due to the complex computation pipeline.

Carion *et al*. [3] proposed a new method, called Detection Transformer or DETR, which uses a encoder-decoder transformer [35] framework to solve this task in an intuitive way utilizing set prediction which has been explored in [23, 34, 33]. DETR uses a deep residual network (ResNet) [15] backbone to extract features and supplements it with a positional encoding before passing it into a transformer encoder. A transformer decoder then takes a small fixed number of learned positional embedding as object queries and additionally attends to the final layer of encoder iteratively. Finally, DETR passes the normalized output of the decoder layer to a feed-forward network (FFN) that predicts either a detection (class and bounding box) or a "no object" class. DETR is trained with Hungarian loss [23] by matching predicted bounding boxes with grounding truth annotations.

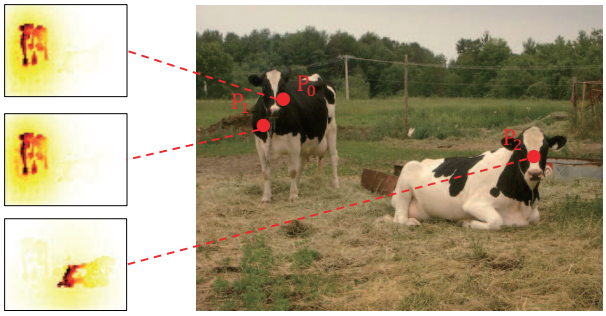

Figure 1. The attention map of some points in the last layer of the transformer encoder. The darker the colour, the greater the weight.

Thanks to the powerful learning ability of Transformer [35], DETR can perform set prediction end-to-end without resorting to human-designed prior like anchor and region proposal thus resulting in a much simpler object detection framework. However, DETR is suffered from high computation complexity in the encoder and slow convergence speed. To achieve good performance, DETR needs a high-resolution image which will increase the computation in the encoder quadratically due to the all-pairs interaction for all positions. Although many improvements of transformer [21, 6, 5, 43, 14, 19, 11, 4, 37] can reduce the com-

*Corresponding author

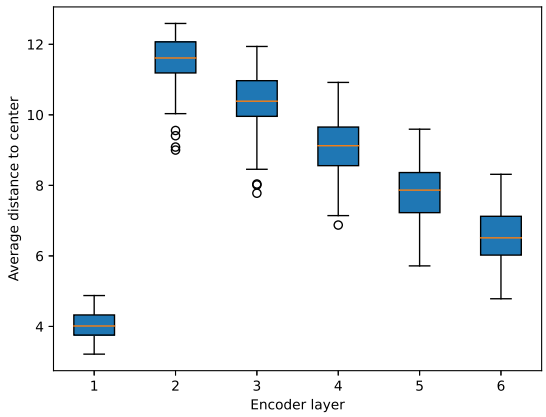

Figure 2. The box plot counts the average distance between the input features and their centre in each encoder layer for the first 100 pictures on the training set.

putation complexity, variants of transformer change the architecture of transformer which require huge trial-and-error cost due to the slow convergence of DETR(1920 GPU hours for single V100). One natural question to ask is whether we can improve the performance and computation trade-off of DETR with acceptable computing resources?

We propose a novel Adaptive Clustering Transformer(ACT) which can replace the self-attention module on the original DETR framework without decrease the performance of pre-trained DETR. ACT is fully compatible with the original transformer thus does not require retraining. The performance gap between ACT and the original transformer can be further closed by equipping with Multi-Task Knowledge Distillation(MTKD). MTKD can also enable seamless switch between models with different FLOPs and Accuracy during inference.

Two observations of DETR motivate our design.

Encoder Attention Redundancy Inside the encoder of DETR, features at each position will collect information from other spatial positions adaptively using the attention mechanism. We observe that features that are semantically similar and spatial near each other will generate a similar attention map, vice versa. As shown in figure 1, the attention map for P0 and P1 are similar to each other and contain redundancy while distant points P0 and P3 demonstrate a completely different attention pattern. The redundancy in self-attention motivates ACT to choose representative prototypes and broadcast the feature update of prototypes to its nearest neighbor.

Encoder Feature Diversity As the encoder goes deeper, features will be similar as each feature will collect information from each other. To verify this hypothesis, we calculated the average distance between the features in each layer to their center for the first 100 pictures on the training set. As shown in figure 2, feature similarity will decrease as we go deeper which consolidates our hypothesis. This observation motivates us to adaptively determine the number of prototypes based on the distribution of features among each layer instead of a static number of cluster centers.

To solve the **Encoder Attention Redundancy**, ACT cluster similar query features together and only calculate the key-query attentions for representative prototypes according to the feature average over the cluster. After calculating the feature update for prototypes, the updated feature will broadcast to its nearest neighbor according to the euclidean distance on the query feature space. A naive idea is to cluster query features using K-means with a pre-defined number of centers for all images. As shown in the experiment part, the K-means cluster will significantly deteriorate the performance of pre-trained DETR. **Encoder Feature Diversity** motivates us to design an adaptive clustering algorithm which can cluster features according to the distribution of feature for each image and each layer. Thus we choose a multi-round Exact Euclidean Locality Sensitivity Hashing (E2LSH) which can perform query feature distribution-aware clustering.

Experiments show that we reduce the FLOPS of DETR from 73.4 Gflops to 58.2 Gflops (excluding Backbone Resnet FLOPs) **without any training process**, while the loss in AP is only 0.7%. The loss in AP can be further reduced to 0.2% by a Multi-Task Knowledge Distillation.

Our main contributions are summarised below.

- We develop a novel method called Adaptive Clustering Transformer (ACT) which can reduce the inference cost of DETR. The core idea of ACT is to select representative prototypes from queries using lightweight LSH and then broadcast the feature update of selected prototypes to its nearest query. ACT can reduce the quadratic complexity of the original transformer, at the same time ACT is fully compatible with the original transformer.
- We reduce the FLOPS of DETR from 73.4 Gflops to 58.2 Gflops (excluding Backbone Resnet FLOPs) **without any training process**, while the loss in AP is only 0.7%.
- We have further reduced the loss in AP to 0.2% through a Multi-Task Knowledge Distillation (MTKD) which enables seamless switch between ACT and original transformer.

2. Related Work

2.1. Review of Attention Model on NLP and CV

Attention model [1, 39, 18] has been widely used in CV and NLP fields due to the in-built adaptive information ag-

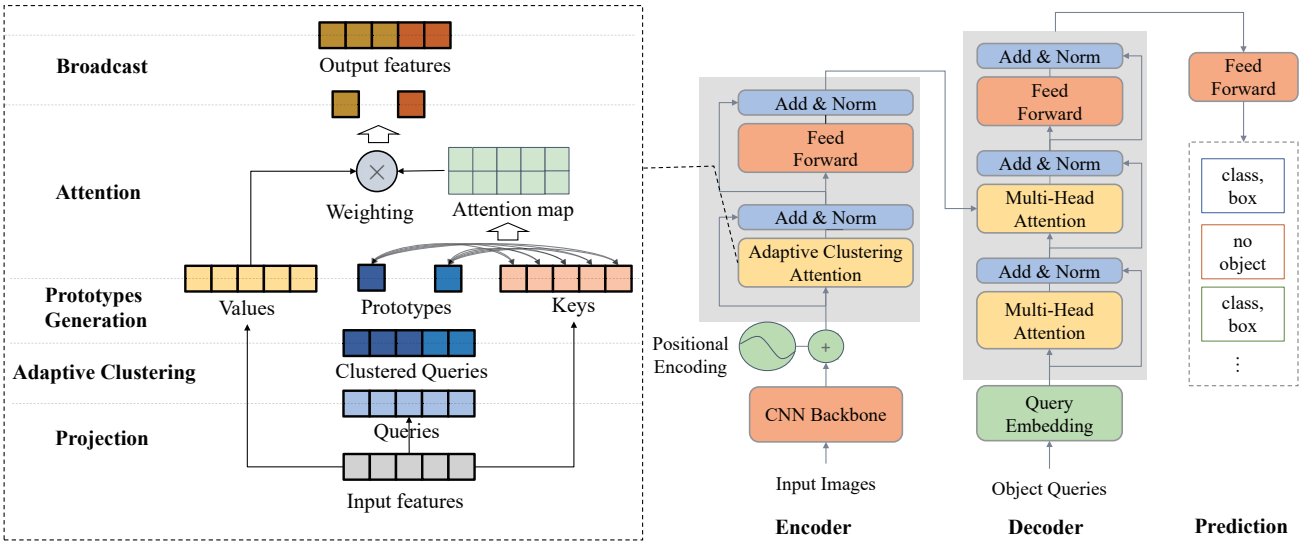

Figure 3. The encoder module of our adaptive clustering transformer. We use a small number of prototypes to represent the queries and only the attention map between prototypes and keys will be calculated. The number of prototypes will be automatically determined based on the distribution of queries. Finally, the attention output will be broadcast to the queries represented by the prototype.

gregation mechanism. We mainly focus on one branch of attention called transformer. Transformer performs information exchange between all pairs of entities like words in NLP or regions in CV. Transformer has achieved state-of-the-art performance on Machine Translation [35], Object Detection [3], Multimodality Reasoning [10, 41, 30], Video Classfication [38] and Language Understanding [8]. While transformer has achieved good performance on different scenarios, but it is hard to scale due to the quadratic complexity with respect to the length of the input sequence. Many modifications of transformer have been proposed to tackle the computation bottleneck of transformer. Reformer [21] proposed sharing key and query and use Locality Sensitivity Hashing (LSH) [7] to cluster features near into one cluster, then perform information exchange inside each cluster. Performer [6] approximates the softmax between key and query interaction using Positive Orthogonal Random Features (PORF) with provable approximate error with linear complexity. Linear Attention [19] utilizes association property to Key-Query-Value multiplication from quadratic complexity into linear complexity. Progressive Elimination [14] finds that redundancy exists in transformer and progressively eliminates the input of each layer and achieves comparable performance with the original transformer by reducing computation cost. Asymmetric Attention [43] summarises key features into a few key vectors using multi-scale pooling over key features thus reduce the computation complexity. Global Graph Reasoning [5] transforms the original input into global vectors utilizing weighted pooling and then perform information exchange over the compact global vectors. Previously mentioned methods modified the structure of the original transformer and need huge resources for training and inference. Our proposed Adaptive Clustering Transformer (ACT) shares the same structure as the original transformer. ACT reduces the computation cost of transformer without re-training. Besides, the performance gap between ACT and the original transformer can be further reduced with a few epochs of fine-tuning knowledge distillation [16, 42].

2.2. Object Detection using Deep Learning

The main framework of object detection is dominated by performing classification over a sliding window. Viona-Jones Face detector [36] first introduce the idea of sliding window approach into face detection with adaboost [9]. After the successful application of CNN on object classification [22, 15], deep features have been applied to object detection. Previous research of object detection using deep features can be divided into two-stage and one-stage object detection. RCNN, Fast RCNN and Faster RCNN [13, 12, 32] are two-stage solution while YOLO [31] and SSD [27] are one-stage solution. Previous methods on object detection are suffered by complex post-processing pipeline (NMS) [29, 2], imbalanced loss [25] and head-crafted anchor [27, 31] which increase the difficulty of training and deployment. Unlike sliding-window approaches, object detection has been formulated as a permutation-invariant set prediction problem. Steward et al [34] proposed an end-to-end people detection which encodes image feature using CNN and decodes the bounding box sequen-

tially using LSTM [17]. The predicted bounding box will be matched with ground truth using Hungarian loss [23] and trained end-to-end. Recurrent instance segmentation [33] add an extra segmentation head over end-to-end object detection framework and successfully test the idea on instance segmentation. Recently DETR [3] has successfully made the performance of set-prediction approaches comparable with two-stage Faster RCNN approaches by replacing LSTM [17]with much powerful Transformer [35]. End-to-end set prediction problem significantly simplified the pipeline of object detection and reduce the need for handcrafted prior. However, the convergence of end-to-end set prediction is slow and need huge inference cost caused by the quadratic complexity of self-attention. Our proposed ACT target to reduce the inference computation cost of DETR without the need for retraining.

3. Adaptive Clustering Transformer

In this section, we will first revisit the DETR framework for object detection in Section 3.1. Then, in Section 3.2 we introduce our proposed Adaptive Clustering Transformer (ACT) which can serve as a drop-in module into the pre-trained DETR model. And Section 3.3 provides a simple error analysis. Finally, we will introduce the Multi-Task Knowledge Distillation (MTKD) in Section 3.3. MTKD not only increase the detection accuracy of ACT but also produce seamless switch between models of different performance and computation trade-off.

3.1. Main Structure of DETR

Figure 3 also shows the three stages of DETR. In the encoder, an ImageNet-pre-trained ResNet model is used to extract 2D features from the input image. Positional encoding module uses sine and cosine functions with different frequencies to encode the spatial information. DETR flattens the 2D features and supplements them with the positional encoding and passes them to the 6-layer transformer encoder. Each layer of the encoder has the same structure, including an 8-head self-attention module and an FFN module. The decoder then takes as input a small fixed number of learned positional embeddings, which are called object queries, and additionally attends to the encoder output. The decoder also has 6 layers, and each layer contains an 8-head self-attention module, an 8-head co-attention module, and an FFN module. Finally, DETR passes each output of the decoder to a shared feed-forward network that predicts either a detection (class and bounding box) or a "no object" class.

3.2. Adaptive Clustering Transformer

Determine Prototypes We use Locality Sensitivity Hashing (LSH) to adaptively aggregate those queries with small Euclidean distance. LSH is a powerful tool to solve the Nearest Neighbour Search problem. We call a hashing scheme locality-sensitive if nearby vectors get the same hash with high probability and distant ones do not. By controlling the parameters of the hash function and the number of hashing rounds, we let all vectors with a distance less than ϵ fall into the same hash bucket with a probability greater than p.

We choose Exact Euclidean Locality Sensitive Hashing (E2LSH) [7] as our hash function:

$$h(\vec{v}) = \lfloor \frac{\vec{a} \cdot \vec{v} + b}{r} \rfloor \tag{1}$$

where $h : \mathbb{R}^d \rightarrow \mathbb{R}$ is the hash function, r is a hyper-parameter, $\vec{a}, b$ are random variables satisfying $\vec{a} = (a_1, a_2, ..., a_d)$ with $a_i \sim \mathcal{N}(0, 1)$ and $b \sim \mathcal{U}(0, r)$. We will apply L rounds of LSH to increase the credibility of the results. The final hash value will be obtained by equation 2.

$$h(\vec{v}) = \sum_{i=0}^{L-1} C^i h_i(\vec{v}) \tag{2}$$

where each h_i is obtained by equation 1 with independently sampled parameters $\vec{a}$ and b.

Figure 4 shows the principle of our hash function. Each hash function h_i can be regarded as a set of parallel hyperplanes with random normal vector $\vec{a_i}$ and offset b_i. The hyper-parameter r controls the spacing of the hyperplanes. The greater the r, the greater the spacing. Furthermore, L hash functions divide the space into several cells, and the vectors falling into the same cell will obtain the same hash value. Obviously, the closer the Euclidean distance, the greater the probability that the vectors fall into the same cell.

To obtain the prototypes, we calculate the hash value for each query firstly. Then, queries with the same hash value will be grouped into one cluster, and the prototype of this cluster is the center of these queries. More formally, we define $Q \in \mathbb{R}^{N \times D_k}$ as the queries and $P \in \mathbb{R}^{C \times D_k}$ as the prototypes, where C is the number of clusters. Let G_i represents the index of the cluster that Q_i belongs to. The prototype of the $j-th$ cluster can be obtained by equation 3.

$$P_j = \frac{\sum_{i, G_i = j} Q_i}{\sum_{i, G_i = j} 1} \tag{3}$$

Estimate Attention Output After the previous step, each group of queries is represented by a prototype. Thus, only the attention map between prototypes and keys need to be calculated. Then, we get the target vector for each prototype and broadcast it to each original query. Thus, we get an estimation of the attention output. Compared with the exact attention calculation, we reduce the complicity from $O(NMD_k + NMD_v)$ to $O(CMD_k + CMD_v)$, where C

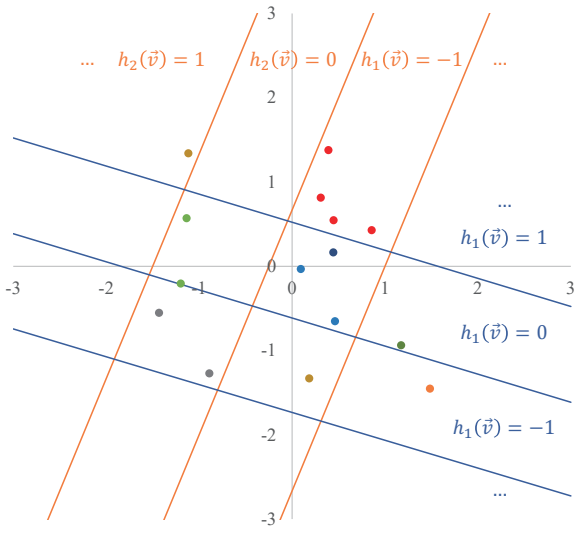

Figure 4. The principle of our hash function. Each hash function h_i can be regarded as a set of parallel hyperplanes with random normal vector $\vec{a_i}$ and offset b_i. The hyper-parameter r controls the spacing of the hyperplanes. L hash functions divides the space into several cells. The vectors falling into the same cell will obtain the same hash value.

is the number of prototypes, which is larger smaller than N and will be adaptively determined.

More formally, we define $K \in \mathbb{R}^{M \times D_k}$ as the keys and $V \in \mathbb{R}^{C \times D_v}$ as the values. We get the estimate of attention output $\tilde{V^o}$ by the following equations:

$$\tilde{A} = softmax(PK^T/\sqrt{D_k}) \tag{4}$$

$$\tilde{W} = AV \tag{5}$$

$$\tilde{V_i^o} = W_j \ , \ if \ G_i = j \tag{6}$$

where the softmax function is applied row-wise and G_i represents the index of the cluster that Q_i belongs to.

3.3. Error Control

What we would like to emphasize is that the error caused by using prototypes instead of the exact queries are controllable. Datar *et al*. have discussed in detail the collision probability of E2LSH [7].

Consider one round of E2LSH shown in Equation 1. For two vectors $\vec{v_1}$ and $\vec{v_2}$, let $c = \|\vec{v_1} - \vec{v_2}\|_2$. Suppose the probability that v_1 and v_2 fall into the same hash bucket is $p(c)$, then we can prove that

$$p(c) = P[h(\vec{v_1}) = h(\vec{v_2})] = \int_0^r \frac{1}{c} f_p(\frac{t}{c})(1 - \frac{t}{r})dt \tag{7}$$

where f_p is the probability density function of a normal distribution with a mean of 0 and a variance of 1.

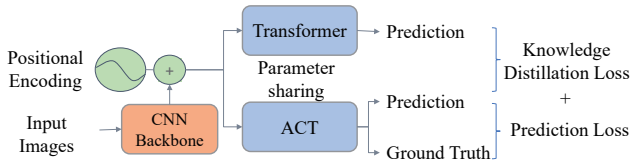

Figure 5. Multi-Task Knowledge Distillation. Image features will be extracted by the pre-trained CNN backbone first. The extracted feature will be passed into ACT and the original transformer parallel. To enable seamless switch between ACT and orignal transformer, MTKD will guide the training.

We apply L rounds of hashing independently, so the collision probability of v_1 and v_2 ($\|\vec{v_1} - \vec{v_2}\|_2 = c$) is $p_L(c) = p(c)^L$. Obviously $P_L(c)$ decreases monotonically with c. Therefore, two queries will be grouped to the same cluster with a probability greater than α if their distance is less than $p_L^{-1}(\alpha)$. Thus, for given confidence α, we can control the distance between queries and their prototype in the same cluster by adjusting the hyper-parameters L and r. The estimation error will decrease with an increase of L and a decrease of r.

3.4. Multi-Task Knowledge Distillation

Although ACT can reduce the computation complexity of DETR without retraining, we show that Multi-Task Knowledge Distillation(MTKD) can further improve ACT with a few-epoch of fine-tuning and produce a better balance between performance and computation. The pipeline of MTKD is illustrated in figure 5. Image features will be extracted by the pre-trained CNN backbone first. The extracted feature will be passed into ACT and the original transformer parallel. To enable a seamless switch between ACT and the original transformer, MTKD will guide the training. The training loss is denoted below:

$$L = L_{pred}(Y, Y_2) + L_{KD}(B_1, B_2.detach()) \tag{8}$$

where Y represents the ground truth, B_1 represent the predicted bounding box of ACT, and B_2, Y_2 represent the predicted bounding box and full prediction of DETR. $L_{pred}(Y, Y_2)$ is the original loss between the ground truth and the prediction of DETR. $L_{KD}(B_1, B_2)$ is the knowledge distillation loss which minimise the L2 distance between predicted bounding box of ACT and DETR.

The training loss aims to train the original transformer jointly with knowledge transfer between full prediction and approximated prediction which enable a seamless switch between ACT and Transformer. The knowledge transformer includes region classification and regression distillation. The regression branch is more sensitive to the approximated error introduced by ACT than the classification branch. Thus, we only transfer the knowledge of the bounding box regression branch. We observe much faster conver-

gence by transferring the box regression branch only.

4. Experiment

4.1. Dataset

We perform experiments on COCO 2017 detection dataset [26], which containing 118k training images and 5k validation images. Each image in the dataset contains up to 63 instances of different sizes. We report AP as bbox AP, the integral metric over multiple thresholds. We also report the average FLOPs for the first 100 images in the COCO 2017 validation set. Only the FLOPs of convolutional layers, fully connected layers, matrix operations in attention, E2LSH, and clustering will be considered.

4.2. Experiment Setup

We choose the pre-trained DETR-DC5 model [3] as our baseline. It uses the deep residual network [15] with 50 layers (ResNet-50) as the backbone and increases the feature resolution by adding a dilated convolution [40] to the last stage of the backbone and removing a stride from the first convolution of this stage. DETR-DC5 contains 6 encoder layers and 6 decoder layers with 8 attention heads.

We replace the attention module in the encoder with our adaptive clustering attention while keeping the other parts unchanged. We randomly sample 1000 images on the training set and calculate the mean square error between our estimated attention map and the true attention map to determine an appropriate hyper-parameter r and control the FLOPs of the model by changing the hyper-parameter L, where L represent the round of E2LSH and r represent the interval of the hashing hyperplanes.

During inference, we resize the input images such that the shortest side is at most 800 pixels while the longest at most 1333. For Multi-Task Knowledge Distillation(MTKD), we adopt a random crop augmentation which has been used in DETR. We use a simple L2 norm for regression distillation and the weight of KD loss is set to be 1. MTKD perform fine-tuning over the pre-trained model for 5 epochs with the learning rate of 10^{-5} and continue running for 2 epochs by reducing the learning rate by 1/10. MTKD is optimised by AdamW [20, 28].

4.3. Ablation Study

The hyper-parameters of the E2LSH have a great influence on the quality of the approximation and the FLOPs of the model. In our ablation analysis, we explore how these hyperparameters affect the results and try to determine appropriate hyper-parameters.

We randomly sample 1000 images on the training set and input the images into DETR and our ACT respectively to obtain the attention map in different encoder layers and different attention heads. We calculate the mean square error between our estimated attention map and the true attention map. We perform two sets of experiments. First, we fix L to 24 and set r to 2,4,6,8,12 respectively. Then, we fix r to 8 and set L to 16, 20, 24, 32 respectively.

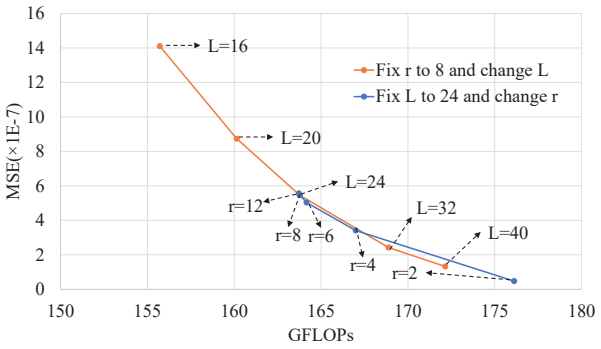

Figure 6. The mean square error between the estimated attention map and the true attention map under an certain computational budget. We fix L to 24 and set r to 2,4,6,8,12 respectively. Then we fix r to 8 and set L to 16, 20, 24, 32 respectively.

The results are shown in Figure 6. Firstly, the estimation error decreases with an increase of L and a decrease of r, which is in line with the analysis in section 3.3. Secondly, when r is greater than 6, continuing to increase r has little effect on estimation errors and FLOPs. Therefore, it is a better choice to obtain models of different FLOPs by the change of L. Finally, we found that when r is less than or equal to 6, continuing to reduce r will cause a larger increase in FLOPs while a smaller decrease in error, which is not cost-effective. Thus, in all subsequent experiments, we fix r to 8.

Another significant discovery is that adaptively clustering keys using our method can also achieve a good result. We perform experiments on clustering queries, clustering keys, and clustering both queries and keys respectively. We adjust the hyper-parameter L to ensure that the three experiments have similar FLOPs, and we compare the AP on the validation set. The results are given in Table 2. As we can see, these three methods have achieved similar AP under the same FLOPs, which also proves the generalization of our model. This means that for some models where the number of keys is significantly more than the number of queries, we can adaptively cluster the keys to obtain higher efficiency.

4.4. Final Performance

Speed Accuracy Trade-off. In this section, we start by comparing the AP of our model with DETR-DC5. We also compare our adaptive clustering method with K-means clustering used by Vyas *et al.* [37]. We refer to the number of clusters in K-means as C. We adjust L and C and calculate the AP under different computational budget. As we can see in Figure 7, under an equalized computa-

Table 1. We compare the AP of our model with DETR-DC5 and Faster RCNN in detail. All the models use dilated ResNet-50 as the backbone. We refer to the bbox AP of large, medium and large size instance as AP_L, AP_M and AP_S respectively.

Model	GFLOPs	AP	AP_L	AP_M	AP_S
Backbone (ResNet-50)	110.7				
DETR-DC5	+73.4	43.3	61.1	47.3	22.5
Faster RCNN-DC5	+209.3	41.1	55.0	45.9	22.9
ACT (L=32)	+58.2	42.6	61.1	46.8	21.4
ACT (L=24)	+53.1	41.3	60.6	45.9	19.2
ACT (L=20)	+49.4	39.7	60.3	44.2	16.9
ACT (L=16)	+45.0	37.1	58.8	41.3	13.9
ACT+MTKD (L=32)	+58.2	43.1	61.4	47.1	22.2
ACT+MTKD (L=24)	+53.1	42.3	61.0	46.4	21.3
ACT+MTKD (L=20)	+49.5	41.8	60.7	45.6	20.6
ACT+MTKD (L=16)	+45.1	40.6	59.7	44.3	18.5

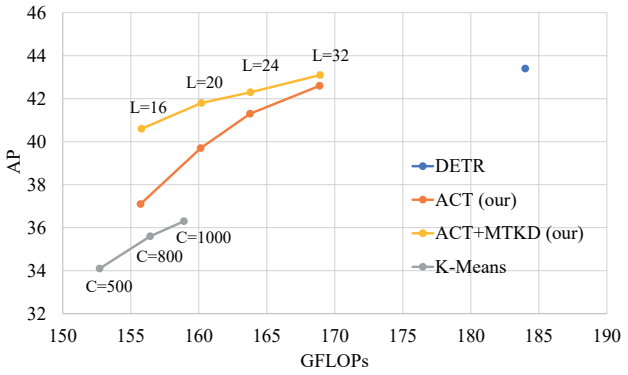

Figure 7. we compare the AP of ACT with DETR-DC5 and the K-mean clustering. We refer to hash rounds in our model as L and refer to the number of clusters in K-means as C.

Table 2. AP and FLOPs under different clustering targets. We perform experiments on clustering queries, clustering keys, and clustering both queries and keys respectively.

Cluster queries	Cluster keys	FLOPs	AP
$L = 24$	$\times$	163.77	0.413
$\times$	$L = 8$	163.48	0.414
$L = 32$	$L = 12$	162.7	0.411

tional budget, the AP of our ACT model is much higher than K-means's. Compared with DETR-DC5, we reduce the FLOPs from 184.1 GFLOPs to 168.9 GFLOPs while the loss in AP is only 0.7%. The yellow line also shows the result of a Multi-Task Knowledge Distillation (MTKD). MTKD can significantly improve AP, especially for those models with smaller flops. Through MTKD, the AP of ACT with $L = 16$ is increased by 4.3%, and ACT with $L = 32$ achieves almost the same performance as DETR-DC5.

We also analyze the advantages of our method compared to the K-means clustering. K-means clustering uses the same number of clusters in all of the encoder layers. But it is hard the determine an appropriate hyper-parameter because the distribution of queries varies greatly with the input image and the index of the encoder layer. An inappropriate hyper-parameter C will lead to bad estimates or a waste of computing resources. Another disadvantage is that K-means may not converge well in some cases, and many clusters will be empty when C is relatively large. Our method adaptively determines the number of prototypes so that these disadvantages can be avoided.

Compare AP in Detail. We refer to the bbox AP of the large, medium, and large size instances as AP_L, AP_M, and AP_S respectively. In this section, we will compare these metrics with Faster RCNN and DETR using the same backbone structure (dilated ResNet 50). Table 1 shows the results in detail. Our ACT model with L equal to 32 achieves the same performance as Faster RCNN-FPN with fewer FLOPs. What's more, ACT is much stronger than Faster RCNN-FPN in detecting objects with large or medium size. Our ACT model can approximate the attention map in DETR well especially for large-size objects. For example, when L=32, the AP_L of our model is the same as DETR-DC5. Most of the loss in AP occurs in small and medium-sized objects. Through a few-epoch of MTKD, the APs in different sizes have been significantly improved, especially for those models with fewer FLOPs. And our ACT with L equal to 32 achieves almost the same performance as DETR-DC5. We found that most of the improvement in AP comes from small and medium-sized objects. Another interesting finding is that AP_L of our ACT is 0.3% higher than DETR's. We believe that in the process of knowledge distillation, our clustering attention can be regarded as a dropout operation, which can prevent overfitting.

Composition of FLOPs. We also analyze the composition of FLOPs in DETR-DC5. Table 3 shows the results. As we can see, the backbone accounts for more than 60% of the

Table 3. The composition of FLOPs in DETR-DC5.

Stage	Module Type	GFLOPs	Proportion
Backbone	ResNet-50	110.7	60.1%
Encoder	Attention	43.5	24.6%
Encoder	FFN	22.0	12.0%
Decoder	Attention+FFN	6.05	3.29%
Prediction	FFN	1.85	0.01%
Total		184.1	100%

Table 4. The inference time and memory cost on a Nvidia GeForce GTX TITAN X with batch size of 1.

Model	Inference Time per Image	Memory
DETR-DC5	0.246s	1862MiB
ACT(L=32)	0.218s	1733MiB
ACT(L=24)	0.207s	1584MiB
ACT(L=20)	0.195s	1415MiB
ACT(L=16)	0.183s	1142MiB

FLOPs. Then there is the attention module in the encoder, which accounts for 24.6%. Our method focuses on reducing FLOPs in this part. For example, when L=24, we reduce the FLOPs of the attention module in the encoder from 43.5 GFLOPs to 23.2 GFLOPs. In this way, the FLOPS of attention is balanced with the FLOPS of FFN. The statistics also indicate that if we want to further reduce the FLOPs, we need to do more research on the backbone and FFN.

Inference Time and Memory. The above analyses are based on theoretical computation cost (FLOPs). We also tested the time and memory cost in a real environment. Table 4 shows the inference time and memory cost on an Nvidia GeForce GTX TITAN X with the batch size of 1. As we can see, the acceleration of ACT in a real environment is consistent with the theoretical analysis, and the memory cost is also significantly reduced.

4.5. Visualisation of Adaptive Clustering

To analyze which queries are represented by the same prototype, we visualize some representative clusters in Figure 8. We can easily find that the three clusters displayed are the features of the cow, the sky, and the field. This indicates that our clustering is related to semantics and location. Those queries with similar semantics and similar locations will easily be grouped.

To prove that our method can adaptively determine the number of prototypes based on the distribution of queries, Figure 9 counted the ratio of the number of prototypes to the number of queries in each encoder layer for the first 100 images on the validation set. We can find that as the index of the encoder layer increases, the number of prototypes shows a downward trend, which is consistent with the distribution of queries shown in Figure 2. This shows that our adaptive clustering method is very effective for this situation where the query distribution changes greatly.

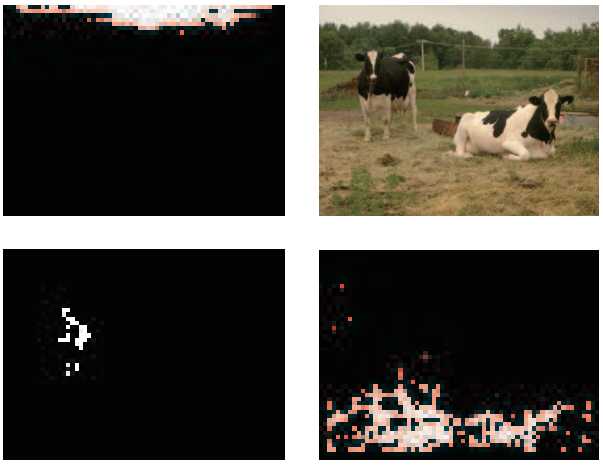

Figure 8. We visualize some representative clusters in the encoder. The queries where the white pixel is located belong to the same cluster. The original image is at the top right. The cluster on the upper left contains features of the sky, the cluster on the bottom left contains features of the cow, and the cluster on the bottom right contains features of the field.

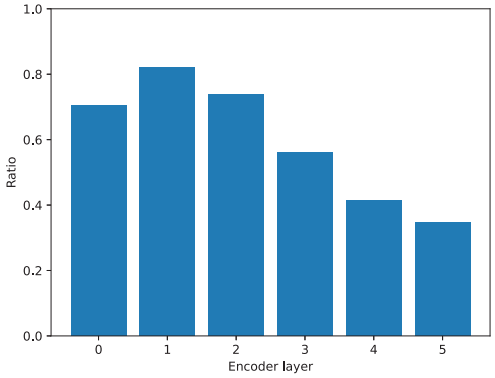

Figure 9. The ratio of the number of prototypes to the number of queries in each encoder layer. We show the average value on the first 100 images in the validation set.

5. Conclusion

In this paper, we propose the Adaptive Clustering Transformer (ACT) to reduce the computation and memory costs for object detection. Most previous efficient transformers need a re-training when applied to DETR. However, training the DETR requires 500 epoch approximately 1920 GPU hours for a single V100 GPU. Our proposed ACT does not need any re-training process due to the compatibility between ACT and Transformer. ACT reduces the redundancy that exists in the pre-trained DETR in a clever adaptive clustering way. In this future, we will look into the ACT on

training from scratch setting and also apply ACT to perform cross-scale information fusion over multi-scale Feature Pyramid Network (FPN) [24].

References

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014. 2

[2] Navaneeth Bodla, Bharat Singh, Rama Chellappa, and Larry S Davis. Soft-nms–improving object detection with one line of code. In *Proceedings of the IEEE international conference on computer vision*, pages 5561–5569, 2017. 3

[3] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. *arXiv preprint arXiv:2005.12872*, 2020. 1, 3, 4, 6

[4] Yunpeng Chen, Yannis Kalantidis, Jianshu Li, Shuicheng Yan, and Jiashi Feng. Aˆ 2-nets: Double attention networks. In *Advances in neural information processing systems*, pages 352–361, 2018. 1

[5] Yunpeng Chen, Marcus Rohrbach, Zhicheng Yan, Yan Shuicheng, Jiashi Feng, and Yannis Kalantidis. Graph-based global reasoning networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 433–442, 2019. 1, 3

[6] Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. Rethinking attention with performers. *arXiv preprint arXiv:2009.14794*, 2020. 1, 3

[7] Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the twentieth annual symposium on Computational geometry*, pages 253–262, 2004. 3, 4, 5

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. 3

[9] Jerome Friedman, Trevor Hastie, Robert Tibshirani, et al. Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). *The annals of statistics*, 28(2):337–407, 2000. 3

[10] Peng Gao, Zhengkai Jiang, Haoxuan You, Pan Lu, Steven CH Hoi, Xiaogang Wang, and Hongsheng Li. Dynamic fusion with intra-and inter-modality attention flow for visual question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6639–6648, 2019. 3

[11] Peng Gao, Haoxuan You, Zhanpeng Zhang, Xiaogang Wang, and Hongsheng Li. Multi-modality latent interaction network for visual question answering. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5825–5835, 2019. 1

[12] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015. 1, 3

[13] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014. 1, 3

[14] Saurabh Goyal, Anamitra Roy Choudhary, Venkatesan Chakaravarthy, Saurabh ManishRaje, Yogish Sabharwal, and Ashish Verma. Power-bert: Accelerating bert inference for classification tasks. *arXiv preprint arXiv:2001.08950*, 2020. 1, 3

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 1, 3, 6

[16] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. 3

[17] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 4

[18] Zhengkai Jiang, Peng Gao, Chaoxu Guo, Qian Zhang, Shiming Xiang, and Chunhong Pan. Video object detection with locally-weighted deformable neighbors. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 8529–8536, 2019. 2

[19] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. *arXiv preprint arXiv:2006.16236*, 2020. 1, 3

[20] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 6

[21] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*, 2020. 1, 3

[22] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017. 3

[23] Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955. 1, 4

[24] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017. 9

[25] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017. 3

[26] 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 *European conference on computer vision*, pages 740–755. Springer, 2014. 6

[27] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C

Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016. 1, 3

[28] Ilya Loshchilov and Frank Hutter. Fixing weight decay regularization in adam. 2018. 6

[29] Alexander Neubeck and Luc Van Gool. Efficient non-maximum suppression. In *18th International Conference on Pattern Recognition (ICPR'06)*, volume 3, pages 850–855. IEEE, 2006. 3

[30] Duy-Kien Nguyen and Takayuki Okatani. Improved fusion of visual and language representations by dense symmetric co-attention for visual question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6087–6096, 2018. 3

[31] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016. 1, 3

[32] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015. 1, 3

[33] Bernardino Romera-Paredes and Philip Hilaire Sean Torr. Recurrent instance segmentation. In *European conference on computer vision*, pages 312–329. Springer, 2016. 1, 4

[34] Russell Stewart, Mykhaylo Andriluka, and Andrew Y Ng. End-to-end people detection in crowded scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2325–2333, 2016. 1, 3

[35] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017. 1, 3, 4

[36] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001*, volume 1, pages I–I. IEEE, 2001. 1, 3

[37] Apoorv Vyas, Angelos Katharopoulos, and François Fleuret. Fast transformers with clustered attention. *Advances in Neural Information Processing Systems*, 33, 2020. 1, 6

[38] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7794–7803, 2018. 3

[39] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057, 2015. 2

[40] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015. 6

[41] Zhou Yu, Jun Yu, Yuhao Cui, Dacheng Tao, and Qi Tian. Deep modular co-attention networks for visual question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6281–6290, 2019. 3

[42] Linfeng Zhang, Jiebo Song, Anni Gao, Jingwei Chen, Chenglong Bao, and Kaisheng Ma. Be your own teacher: Improve the performance of convolutional neural networks via self distillation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3713–3722, 2019. 3

[43] Zhen Zhu, Mengde Xu, Song Bai, Tengteng Huang, and Xiang Bai. Asymmetric non-local neural networks for semantic segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 593–602, 2019. 1, 3