PATCHDCT: PATCH REFINEMENT FOR HIGH QUALITY INSTANCE SEGMENTATION

Anonymous authors
Paper under double-blind review

ABSTRACT

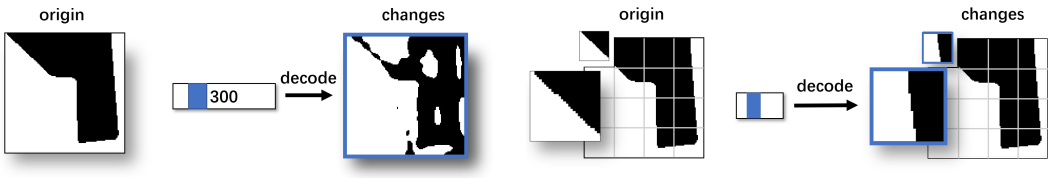
High-quality instance segmentation has shown emerging importance in computer vision. Without any refinement, DCT-Mask directly generates high-resolution masks by compressed vectors. To further refine masks obtained by compressed vectors, we propose for the first time a compressed vector based multi-stage refinement framework, named PatchDCT. However, the vanilla combination does not bring significant gain, because changes in some elements of the DCT vector will affect the prediction of the entire mask. Thus, we separate masks decoded from DCT vector into several patches and refine each patch by the designed classifier and regressor, respectively. Specifically, the classifier is used to distinguish mixed patches from all patches, and to correct previously mispredicted foreground and background patches. In contrast, regressor is used for DCT vector prediction of mixed patches, which further refines the segmentation quality at boundary locations. Experiments on COCO show that our achieves 2.0%, 3.2%, and 4.5% AP and 3.3%, 5.3%, 7.0% Boundary AP improvements over Mask-RCNN on COCO, LVIS, and Cityscapes, respectively. It also surpasses DCT-Mask by 0.7% , 1.4% and 1.3% AP and 0.9%, 1.7% and 4.2% Boundary AP on COCO, LVIS and Cityscapes. Besides, the performance of PatchDCT is also competitive with other state-of-the-art methods, and the code will be made publicly available.

1 INTRODUCTION

Instance segmentation (Li et al., 2017; He et al., 2017) is a fundamental but challenging task in computer vision, which aims to locate objects in images and precisely segment each instance. The mainstream instance segmentation method follows Mask-RCNN (He et al., 2017) paradigm, which often segment instance in a low resolution grid (Kang et al., 2020; Cheng et al., 2020; Chen et al., 2019; Ke et al., 2021). However, limited by the coarse mask representation (i.e. 28×28 in Mask-RCNN), most of these algorithms cannot obtain high-quality segmentation results due to many details are not preserved. DCT-Mask (Shen et al., 2021) achieves considerable performance gain by predicting a informative 300-dimensional Discrete cosine transformation (DCT) (Ahmed et al., 1974) vector which compressed from a 128×128 mask. To further improve the segmentation results of DCT-Mask, we follow the refine mechanism (Ke et al., 2022; Zhang et al., 2021; Kirillov et al., 2020) to correct the mask details in a multi-stage manner.

A straightforward implementation is to refine the 300-dimensional Discrete Cosine Transformation (DCT) vector multiple times. However, experimental results show this naive implementation does not succeed, which improves mask average precision (mAP) by 0.2% from 36.5% to 36.7% on COCO *val set*. The main reason for the limited improvement is that the full 300-dimensional DCT vector is not suitable for refining local regions. As each pixel value in the mask is calculated by all elements of the DCT vector in the inference stage, once the predicted value of some elements in the DCT vector changes, the entire mask will change, and even the correct segmentation area may be affected, refer to Figure 1a. It can be seen that vanilla combinations make it difficult for the model to pay more attention to some important local regions, such as wrong predicted regions and boundary regions in masks.

To overcome above issue, we propose a novel method, called PatchDCT, which divide the mask decoded from DCT vector into several independent patches and refine each patch by a three-class classifier and a regressor, respectively. In detail, each patch is first classified into one of three categories:



(a) Influence of element change in DCT-Mask (b) Influence of element change in PatchDCT

Figure 1: (a) Influence of elements changes in DCT vectors for DCT-Mask. The blue block denotes the changed elements. The box with a blue border represents the part of the mask that is affected by the changes in element values. The change of some elements will affect the whole mask. (b) Influence of elements changes in DCT vectors for PatchDCT. The change of some elements of a vector will only affect the corresponding patch.

foreground, background, and mixed by the classifier, and then previously mispredicted foreground and background patches will be corrected. In contrast, mixed patches are fed into the regressor to predict their corresponding n-dimensional ($n \ll 300$) DCT vector. In the inference stage, we use IDCT to decode the predicted vectors of the mixed patches as their refined masks, and merge them with the masks of other foreground and background patches to obtain a high-quality high-resolution mask, as shown in Figure 1b. It is also worth emphasizing that each patch is independent, so the element change of a DCT vector will only affect the corresponding mixed patch. Figure 1b shows the influence of element changes in a DCT vector on the prediction results of PatchDCT. In general, patching allows the model to focus on the refinement of local regions, thereby continuously improving the quality of segmentation, resulting in significant performance improvements. Our main contributions are:

1) To our best knowledge, PatchDCT is the first compressed vector based multi-stage refinement detector to predict high-quality mask.

2) PatchDCT innovatively adopts the patching technique, which successfully allows the model to focus on the refinement of local important regions, fully exploiting the advantages of multi-stage refinement and high-resolution information compression.

3) Compared to Mask RCNN, PatchDCT improves about 2.0% mAP and 3.3% Boundary AP on COCO, 3.2% mAP and 5.3% Boundary AP on LVIS, 4.5% mAP and 7.0% Boundary AP on Cityscapes. It also achieves 0.7% mAP and 0.9% Boundary AP on COCO, 1.4% mAP and 1.7% boundary on LVIS, 1.3% mAP and 4.2% Boundary AP on Cityscapes over DCT-Mask.

4) Demonstrated by experiments on COCO *test-dev*, the performance of PatchDCT, at 42.2%, is also competitive with other state-of-the-art methods, and the code will be made publicly available.

2 Related Work

Instance segmentation. Instance segmentation assigns a pixel-level mask to each instance of interest. Mask-RCNN (He et al., 2017) generates bounding boxes for each instance with a powerful detector (Ren et al., 2015) and categorizes each pixel in bounding boxes as foreground or background to obtain 28×28 binary grid masks. Several methods built on Mask-RCNN improve the quality of masks. Cascade RCNN (Cai & Vasconcelos, 2018) uses the same mask branch as Mask-RCNN with a multi-stage detector. HTC (Chen et al., 2019) utilizes interleaved execution, mask information flow, and semantic feature fusion to improve Cascade RCNN. BMask RCNN (Cheng et al., 2020) adds a boundary branch on Mask-RCNN to detect the boundaries of masks. Bounding Shape Mask R-CNN (Kang et al., 2020) improves performance on object detection and instance segmentation by its bounding shape mask branch. Mask Scoring RCNN (Huang et al., 2019) regresses mask IoU to learn the quality of predicted instance masks. BCNet (Ke et al., 2021) uses two GCN (Welling & Kipf, 2016) layers to detect overlapping instances.

Towards high-quality instance segmentation. Since many details may be lost by low-resolution masks, recent methods concentrate on generating fine high-resolution masks. A popular technique is

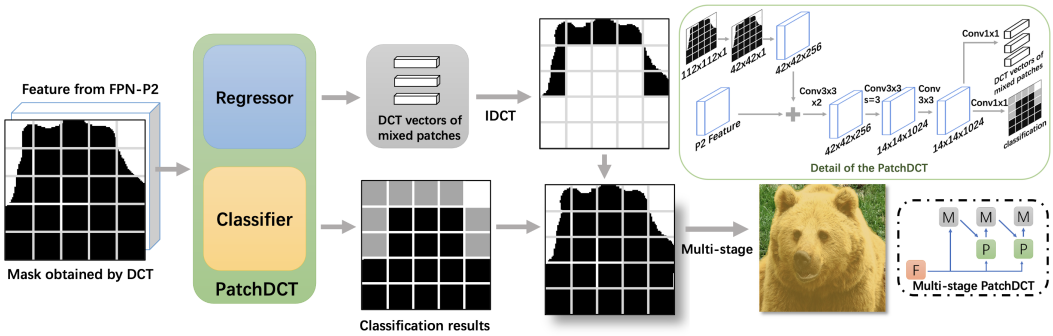


Figure 2: The pipeline of PatchDCT. The classifier differentiates foreground, background and mixed patches. The regressor predicts the DCT vectors of mixed patches. Masks of mixed patches can be obtained by the DCT vectors. PatchDCT combines masks of all patches to obtain a entire mask of instance. The entire mask of instance outputed by PatchDCT can be fed into another PatchDCT module to obtain a finer mask. For the architecture of multi-stage PatchDCT: 'F' is the feature map cropped from FPN-P2. 'M' is the high-resolution mask. 'P' is the PatchDCT module.

refining a coarse mask multiple times to obtain a high-resolution masks. PointRend (Kirillov et al., 2020) adaptively selects several locations to compute labels and applies an iterative subdivision algorithm to generate 224×224 masks from 7×7 coarse masks. RefineMask (Zhang et al., 2021) uses multi-stage procedure and boundary-aware refinement to derive 112×112 masks from 14×14 masks. Mask-Transfiner (Ke et al., 2022) represents image regions as a quadtree and corrects the errors of error-prone tree nodes to generate 112×112 masks. Another common technique is predicting directly compressed vectors of high-resolution masks. MEInst (Zhang et al., 2020) uses PCA to encode binary grid masks into compact vectors and utilizes this method on a single shot instance segmentation framework. DCT-Mask (Shen et al., 2021) regresses DCT vectors that encodes 128×128 masks in training time. SOLQ (Dong et al., 2021) is a query-based method, which also encodes high-resolution masks into DCT vectors and predicts the vectors by queries.

3 Methods

In this section, we show the difficulties in refining DCT vectors and then introduce PatchDCT to overcome these difficulties.

3.1 Difficulties in Refining DCT Vectors

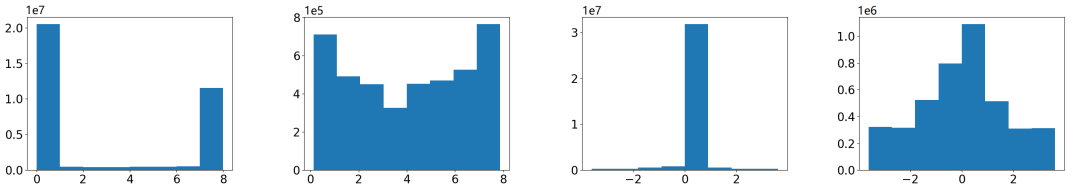
Given a $K \times K$ mask, DCT encodes the mask $\mathbf{M}_{K\times K}$ into the frequency domain $\mathbf{M}^f_{K\times K}$ (Ahmed et al., 1974):

$$M^f_{K\times K}(u,v) = \frac{2}{K}C(u)C(v)\sum_{x=0}^{K-1}\sum_{y=0}^{K-1} M_{K\times K}(x,y) \times \left[\cos\frac{(2x+1)u\pi}{2K}\frac{(2y+1)v\pi}{2K}\right] \tag{1}$$

where $C(w) = 1/\sqrt{2}$ for $w = 0$ and $C(w) = 1$ otherwise. Non-zero values are concentrated in the upper left corner of $\mathbf{M}^f_{K\times K}$, which are low-frequency elements that contain the most information of the mask. The N-dimensional DCT vector is obtained by zigzag scanning(Al-Ani & Awad, 2013) $\mathbf{M}^f_{K\times K}$ and selecting the top-N elements.

DCT-Mask (Shen et al., 2021) predicts an N-dimensional vector for each mask and recovers $\mathbf{M}^f_{K\times K}$ by filling the remaining elements to zero. Then each pixel in the mask $\mathbf{M}_{K\times K}$ is calculated as follow:

$$M_{K\times K}(u,v) = \frac{2}{K}C(u)C(v)\sum_{x=0}^{K-1}\sum_{y=0}^{K-1} M^f_{K\times K}(x,y) \times \left[\cos\frac{(2x+1)u\pi}{2K}\frac{(2y+1)v\pi}{2K}\right] \tag{2}$$



(a) DCCs(all) (b) DCCs(mixed) (c) 2-th element(all) (d) 2-th elements(mixed)

Figure 3: The distributions of DCCs and the second elements on COCO *val2017*. Considering the foreground and background patches greatly changes DCT elements' distributions.

Equation 2 shows that each pixel in the mask $\mathbf{M}_{K\times K}$ is calculated by all elements of $\mathbf{M}^f_{K\times K}$. When refining the N-dimensional DCT vector, once an element is incorrectly changed, all pixels in $\mathbf{M}_{K\times K}$ will be affected, even those regions that are correctly segmented, which is also shown in Figure 1. Therefore, when fixing some specific error regions (e.g. borders), it is difficult to get the correct refinement result unless all the elements in DCT vector are correctly refined. In practice, however, it is almost impossible to refine all N elements correctly.

3.2 PATCHDCT

In order to prevent the above issue when refining the global DCT vectors, we proposed PatchDCT which divided the $K \times K$ mask into $m \times m$ patches and refine each patch respectively. The overall architecture of PatchDCT is shown in Figure 2, which mainly consists of a three-class classifier and a DCT regressor. Specifically, the classifier is used to identify mixed patches and refine foreground and background patches. Mixed patches are refined by DCT-regressor, which regress a n-dimensional DCT vector for each patch.

Three-class classifier. We define the patches with only foreground pixels and only background pixels as foreground patches and background patches, respectively, while the others are mixed patches. The task of differentiating patch categories is accomplished by a fully convolutional three-class classifier. Moreover, the mispredicted initial foreground and background patches are corrected by the classifier. The reason we utilize a three-class classifier instead of a regressor of DCT vectors to refine foreground and background patches is the special form of their DCT vectors. For background patches, simply from Equation 1, all elements of DCT vectors are zeros. For foreground patches, except for the first element which named DC component (DCC)

$$DCC = \frac{1}{m}\sum_{x=0}^{m-1}\sum_{y=0}^{m-1} M_{m\times m}(x,y) \tag{3}$$

all the other elements are zero. It can be derived from Equation 3 that DCC is equal to the patch size m in the foreground patch, since $M_{m\times m}(x,y) = 1$. The proof is shown in Appendix.

Therefore, the form of the DCT vectors in foreground and background patches is very special of which elements involve only zero and m. Regressing DCT vectors of foreground and background patches leads to data imbalance and enhance the difficulties in regression of DCT vectors of mixed patches. As demonstrated in Figure 3, the distridution of elements of all the patches is quiet different from the distridution of elements of mixed patches. It may be difficult for the model to predict accurate DCT vectors of mixed patches (namely the boundary), if we regress DCT vectors of all patches. Thus, in order to make the regressor only pay attention to the mixed patches, a three-classifier is essential.

Regressor. A regressor is used to regress the n-dimensional DCT vector for each mixed patch. Since the mixed patches are exactly the boundary of the entire mask, the regressor is a module that focuses on the prediction of the boundary. Besides, the n-dimensional DCT vectors can be very short ($n \ll N$) to reduce the training complexity. Table 1 shows mask AP obtained by different lengths of ground truth DCT vectors using Mask-RCNN framework on COCO *val2017*. Low dimensional DCT vectors are able to provide enough ground truth information

Table 1: Mask AP obtained by different lengths of ground-truth DCT vectors using Mask-RCNN framework on COCO *val2017* dataset. Low dimensional DCT vectors are able to provide enough ground truth information.

Resolution	Patch Size	Dim.	AP
112×112	1×1	1	57.6
112×112	8×8	3	55.8
112×112	8×8	6	57.1
112×112	8×8	9	57.5
112×112	8×8	12	57.5
112×112	112×112	200	55.8
112×112	112×112	300	56.4

Even if an initial mask is provided, prediction of high resolution masks in binary grid representation still has a high training complexity and leads to bad prediction results (Shen et al., 2021). The specific experimental results will be discussed in the Experiments section (Sec. 4). Regressing DCT vectors of mixed patches allows PatchDCT to easily generate fine boundaries with lower training complexity.

3.3 Multi-stage Refinement and Loss Function

PatchDCT is a module where the input and output masks have the same resolution, as shown in the upper right corner of Figure 2. Thus, the masks generated by a PatchDCT module can be fed into another PatchDCT module for further refinement.

With multi-stage refinement, the loss function of the mask branch is defined as

$$\mathcal{L}_{mask} = \lambda_0 \mathcal{L}_{dct_N} + \sum_s \lambda_s (\mathcal{L}^s_{cls_{patch}} + \mathcal{L}^s_{dct_n}). \tag{4}$$

λ_0 and λ_s are the loss weights. The first item $\mathcal{L}_{dct_N}$ of Equation 4 is the loss in predicting N-dimensional vectors of the entire masks (Shen et al., 2021).

$$\mathcal{L}_{dct_N} = \frac{1}{N} \sum_i^N R(\hat{V}_i - V_i), \tag{5}$$

where V_i and $\hat{V}_i$ are the i-th element in ground-truth and the prediction vector respectively, R is the loss function. The classification loss $\mathcal{L}^s_{cls_{patch}}$ of s-th stage is the cross-entropy loss over three classes. The regression loss $\mathcal{L}^s_{dct_n}$ of s-th stage is

$$\mathcal{L}^s_{dct_n} = \frac{1}{N_m} \sum_k^{N_{all}} \left[p^k \left(\frac{1}{n} \sum_i^n R(\hat{V}_i - V_i) \right) \right], \tag{6}$$

where N_m, N_{all} is the number of mixed patches and all patches respectively. If the k-th patch is a mixed patch, $p^k = 1$, otherwise $p^k = 0$, which means we only regress DCT vectors of mixed patches.

4 Experiments

4.1 Datasets

We evaluate our method on two standard instance segmentation datasets: COCO (Lin et al., 2014) and Cityscapes (Cordts et al., 2016). We report the standard mask AP metric and the Boundary AP (Cheng et al., 2021) metric(AP_B), which focuses on evaluating the boundary quality.

COCO provides 80 categories with instance-level annotations. Our model is trained on COCO *train2017* and validated on *val2017* and *test-dev2017*. Following (Kirillov et al., 2020), we also report AP^* and AP^*_B, which evaluates COCO *val2017* with high-quality annotations provided by LVIS (Gupta et al., 2019). Note that for AP^* and AP^*_B, models are still trained on COCO *train2017*.

Table 2: Mask AP on COCO with different backbones based on Mask-RCNN framework. AP^* is results obtained from COCO with LVIS annotations. AP_B is Boundary AP. AP^*_B is Boundary AP using LVIS annotations. Models with R101-FPN and RX101-FPN are trained with '3×' schedule.

Backbone	**Model**	**AP**	$\mathbf{AP}_S$	$\mathbf{AP}_M$	$\mathbf{AP}_L$	$\mathbf{AP}_B$	$\mathbf{AP}^*$	$\mathbf{AP}^*_S$	$\mathbf{AP}^*_M$	$\mathbf{AP}^*_L$	$\mathbf{AP}^*_B$	**FPS**
R50-FPN	Mask-RCNN	35.2	17.2	37.7	50.3	21.1	37.6	21.3	43.7	55.1	24.8	**13.9**
	DCT-Mask	36.5	17.7	38.6	51.9	23.6	39.7	**23.5**	46.5	58.5	28.4	13.2
	PatchDCT	**37.2**	**18.3**	**39.5**	**54.2**	**24.5**	**40.8**	**23.0**	**47.7**	**60.7**	**30.1**	**12.3**
R101-FPN	Mask-RCNN	38.6	19.5	41.3	55.3	24.5	41.4	24.5	47.9	61.0	29.0	**13.8**
	DCT-Mask	39.9	20.2	42.6	57.3	26.8	43.7	25.8	50.5	64.6	32.4	13.0
	PatchDCT	**40.5**	**20.8**	**43.3**	**57.7**	**27.6**	**44.4**	**27.0**	**51.5**	**65.3**	**33.8**	11.8
RX101-FPN	Mask-RCNN	39.5	20.7	42.0	56.5	25.3	42.1	25.4	48.0	61.4	29.7	**13.3**
	DCT-Mask	41.2	21.9	44.2	57.7	28.0	45.2	27.4	52.6	64.2	34.0	12.9
	PatchDCT	**41.8**	**22.5**	**44.6**	**58.7**	**28.6**	**46.1**	**27.8**	**53.0**	**66.1**	**35.4**	11.7

Cityscapes is a dataset that focuses on urban street scenes. It contains 8 categories for instance segmentation, providing 2,975, 500 and 1,525 high-resolution images $(1,024 \times 2,048)$ for training, validation, and test respectively.

4.2 Implement Details

We modify the model based on DCT-Mask RCNN (Shen et al., 2021). A 112×112 mask is firstly obtained by decoding from a 300-dimensional DCT vector. This mask is then fed into PatchDCT, together with a 42×42 feature map cropped from FPN-P2 (Lin et al., 2017). PatchDCT refines each patch of the mask and outputs a 112×112 mask. We set the patch size to 8 and each patch is represented by a 6-dimensional DCT vector. Our model is class-specific by default, i.e. one mask per class. $L1$ loss and cross-entropy loss were used for DCT vector regression and patch classification. By default, only one PatchDCT module is used, and both λ_0 and λ_1 are set to 1.

We implemented our algorithm based on Detectron2 (Wu et al., 2019), and all hyperparameters remain the same as MaskRCNN in Detectron2. Unless otherwise stated, we use $1\times$ learning schedule.

4.3 Main Results

COCO We compare Mask-RCNN and DCT-Mask RCNN over different backbones. As shown in Table 2, on COCO *val2017* with R50-FPN, PatchDCT improves 2.0% AP and 3.3% AP_B over Mask-RCNN. Compared with DCT-Mask, PatchDCT also achieved 0.7% AP and 0.9% AP_B improvements. When evaluating with LVIS annotations, PatchDCT yields significant gains of 3.2% AP^* and 5.3% AP^*_B over Mask-RCNN, and 1.4% AP^* and 1.7% AP^*_B over DCT-Mask. Recalling that AP^* and AP^*_B are evaluated with high quality annotations, which better reflect the ability of predicting high-quality mask. Moreover, PatchDCT achieves consistent improvements on R101-FPN and RX101-FPN.

We also compare the performance of PatchDCT with state-of-the-art methods of instance segmentation on COCO *test-dev2017*. All comparison results are listed in Table 4. PatchDCT obtains better performance in comparison with PointRend (Kirillov et al., 2020), which first obtains 7×7 mask and renders it to 224×224. PatchDCT achieves comparable performance with MaskTransfiner (Ke et al., 2022) without using the computationally more expensive Transformer. With RX101 backbone, PatchDCT surpasses RefineMask (Zhang et al., 2021), which utilizes a multi-stage procedure to generates 112×112 binary grid masks. The results demonstrate that PatchDCT is an effective refinement module for high-resolution masks.

Cityscapes We report results on Cityscapes *val* set with R50-FPN in Table 3. In comparison with Mask-RCNN, PatchDCT obtains 4.5% AP and 7.0% AP_B improvements. PatchDCT surpasses DCT-Mask by 1.3% AP and 4.2% AP_B on Cityscapes. PatchDCT also gains improvements on both AP and AP_B compared with other SOTA methods. PatchDCT yields high-quality boundaries. PatchDCT achieves 0.8%, 1.4%, 2.1% AP_B improvements over MaskTransfiner, RefineMask and PointRender respectively. These results indicate that PatchDCT generates better high-resolution masks with finer boundaries.

Table 3: Results on Cityscapes *val* set. AP_B is Boundary AP. All models are based on R50-FPN backbone. Runtime measures on A100. PatchDCT achieves the best performance.

Methods	**Resolution**	**AP**	$\mathbf{AP}_{50}$	$\mathbf{AP}_B$
Mask-RCNN (He et al., 2017)	28×28	33.7	60.9	11.8
PointRender (Kirillov et al., 2020)	224×224	35.9	61.8	16.7
DCT-Mask (Shen et al., 2021)	112×112	36.9	62.9	14.6
RefineMask (Zhang et al., 2021)	112×112	37.6	63.3	17.4
Mask Transfiner (Ke et al., 2022)	112×112	37.9	64.1	18.0
PatchDCT	112×112	**38.2**	**64.5**	**18.8**

Table 4: Comparison of different methods on COCO *test-dev2017*. MS denotes using multi-scale training. '3×' schedules indicates 36 epochs for training.

Method	**Backbone**	**MS**	**Sched.**	**AP**	$\mathbf{AP}_{50}$	$\mathbf{AP}_{75}$	$\mathbf{AP}_S$	$\mathbf{AP}_M$	$\mathbf{AP}_L$
BMask RCNN (Cheng et al., 2020)	R101-FPN		1×	37.7	59.3	40.6	16.8	39.9	54.6
Mask-RCNN (He et al., 2017)	R101-FPN	✓	3×	38.8	60.9	41.9	21.8	41.4	50.5
BCNet (Ke et al., 2021)	R101-FPN	✓	3×	39.8	61.5	43.1	22.7	42.4	51.1
DCT-Mask (Shen et al., 2021)	R101-FPN	✓	3×	40.1	61.2	43.6	22.7	42.7	51.8
MaskTransfiner (Ke et al., 2022)	R101-FPN	✓	3×	40.7	-	-	23.1	42.8	53.8
SOLQ (Dong et al., 2021)	R101-FPN	✓	50e	40.9	-	-	22.5	43.8	54.6
MEInst (Zhang et al., 2020)	RX101-FPN	✓	3×	36.4	60.0	38.3	21.3	38.8	45.7
HTC (Chen et al., 2019)	RX101-FPN		20e	41.2	63.9	44.7	22.8	43.9	54.6
PointRend (Kirillov et al., 2020)	RX101-FPN	✓	3×	41.4	63.3	44.8	24.2	43.9	53.2
RefineMask (Zhang et al., 2021)	RX101-FPN	✓	3×	41.8	-	-	-	-	-
PatchDCT	R101-FPN	✓	3×	40.7	61.8	44.2	22.8	43.2	52.8
PatchDCT	RX101-FPN	✓	3×	**42.2**	**64.0**	**45.8**	**25.0**	**44.5**	**53.9**

4.4 Ablation Experiments

We conduct extensive ablation experiments to further analyze PatchDCT. We adopt R50-FPN as the backbone of Mask-RCNN framework and evaluate the performance on COCO *val2017*.

Simply refine DCT vectors We analyze whether vanilla combination of DCT-Mask and multi-stage refinement could effectively improve the quality of masks. We name the model 'two-stage DCT', in which we regress a new 300-dimensional DCT vector after fusing the masks with a 42×42 feature map from FPN-P2. In the inference time, we decode the second DCT vector and obtain the entire mask directly. From Table 5, this model achieves only little improvements over DCT-Mask, which proves it's very difficult to obtain finer masks by regressing a finer 300-dimensional DCT vector. As mentioned about, this model is unable to focus on previously mispredicted areas, and correctly predicted areas may be influenced by the refinement. As it's important to reduce the interaction of regional refinement, PatchDCT refines each patch respectively.

Binary grid refinement. We analyze whether it's feasible to refine a high-resoluton mask with binary grid representation when the initial mask is provided. We try to generate another 112×112 mask with binary grid representation after fusing the information of the initial mask, which we name 'binary grid refinement'. We fuse the mask with a 56×56 feature map from FPN-P2,with the same operation as PatchDCT, followed by a head the same as Mask-RCNN. However, even if an initial mask is provided, it's still very difficult to predict the binary grid representation of a high-resolution mask because of the high training complexity(Shen et al., 2021). The quality of masks is even worse than the initial masks. Thus, PatchDCT still adopt DCT on mixed patches to generate high-resolution masks.

Effectiveness of three-class classifier In addition to identifying mixed patches, a more important role of three-class classifier is to correct previously mispredicted foreground and background patches. To validate the effectiveness of refining non-mixed patches(i.e. foreground and background patches), we construct a binary-class classifier, which only classify patches as mixed/non-mixed and keeps non-mixed patch same as initial mask. As shown in Table 6, binary-class classifier is inferior to our three-class classifier by 0.3 AP and 0.4 AP^*, since the refinement of previously incorrectly predicted foreground and background patches was ignored.

Refinement of foreground and background patches can also be accomplished with DCT-regressor. However, as discussed in section3.2, the dct vectors of the non-mixed patch are so different from the mixed patch that directly learning the dct vectors of all patches leads to poor results. Table

Table 5: Mask AP obtained by different refinement methods on *val2017*. (G)FLOPs is measured on mask branch. PatchDCT significantly improves the quality of masks.

Method	AP	AP_B	AP^*	AP^*_B	(G)FLOPs
Binary grid	35.7	23.2	39.6	29.1	9.0
Two-stage DCT	36.6	23.9	40.1	29.1	**1.2**
PatchDCT	**37.2**	**24.7**	**40.8**	**30.1**	5.1

Table 6: Mask AP obtained by PatchDCT with no-classifier, two-class classifier and three-class classifier on *val2017*. PatchDCT with three-class classifier achieves the best performance.

Classifier	AP	AP_S	AP_M	AP_L	AP_B	AP^*	AP^*_B
rule-based	36.8	18.1	39.2	53.3	24.1	40.1	29.3
2-class	36.9	18.2	39.3	53.5	24.4	40.4	29.7
3-class	**37.2**	**18.3**	**39.5**	**54.2**	**24.5**	**40.8**	**30.1**

Table 7: Mask AP obtained by PatchDCT with regressor focusing on mixed patches and all patches on *val2017*. The best results are obtained by regressing only the mixed patches.

Regressor	AP	AP_S	AP_M	AP_L	AP_B	AP^*	AP^*_B
all	36.6	17.7	39.5	52.2	23.6	39.6	28.6
mixed	**37.2**	**18.3**	**39.5**	**54.2**	**24.5**	**40.8**	**30.1**

Table 8: Mask AP obtained by PatchDCT with and without the regressor on *val2017*. PatchDCT benefits from the regressor.

Regressor	AP	AP_S	AP_M	AP_L	AP_B	AP^*	AP^*_B
	36.7	18.3	39.0	53.1	23.3	39.6	27.1
✓	**37.2**	**18.3**	**39.5**	**54.2**	**24.5**	**40.8**	**30.1**

Table 9: Mask AP obtained by multi-stage PatchDCT with shared parameters on *val2017*. Two-stage PatchDCT achieves a trade-off between accuracy and computational complexity.

Stage	AP	AP_S	AP_M	AP_L	AP_B	AP^*	(G)FLOPs
1	37.2	**18.3**	39.5	54.1	24.5	40.8	**5.1**
2	**37.4**	17.8	**40.0**	54.0	**24.7**	**41.2**	9.6
3	37.3	17.3	39.7	**54.6**	**24.7**	40.9	14.1

Table 10: Mask AP obtained by models with different patch sizes on *val2017*. PatchDCT with 8×8 patch size obtains the best performance.

Patch Size	AP	AP_S	AP_M	AP_L	AP_B	AP^*	AP^*_B
4×4	37.0	17.5	39.3	53.8	24.4	40.5	29.8
8×8	**37.2**	**18.3**	**39.5**	**54.1**	**24.5**	**40.8**	**30.1**
16×16	37.0	17.6	39.3	53.5	24.4	**40.8**	30.0

Table 11: Mask AP obtained by models with different feature map sizes on *val2017*. The performance saturates with 42×42 feature maps.

Feature Size	AP	AP_S	AP_M	AP_L	AP_B	AP^*	AP^*_B
28×28	37.1	17.8	39.3	53.4	24.5	40.6	30.0
42×42	**37.2**	**18.3**	**39.5**	**54.1**	**24.5**	40.8	**30.1**
56×56	37.0	17.4	39.2	53.0	24.4	**41.0**	30.3

Table 12: Mask AP obtained by models with different dimensions of PatchDCT vectors on COCO *val2017*. Model with 6-dimensional vectors achieves the best performance.

Patch Dim.	AP	AP_S	AP_M	AP_L	AP_B	AP^*	AP^*_B
3	36.8	17.6	39.2	53.5	24.0	40.5	29.5
6	**37.2**	**18.3**	**39.5**	**54.1**	**24.5**	**40.8**	**30.1**
9	36.9	17.1	39.3	53.3	24.3	40.6	30.1

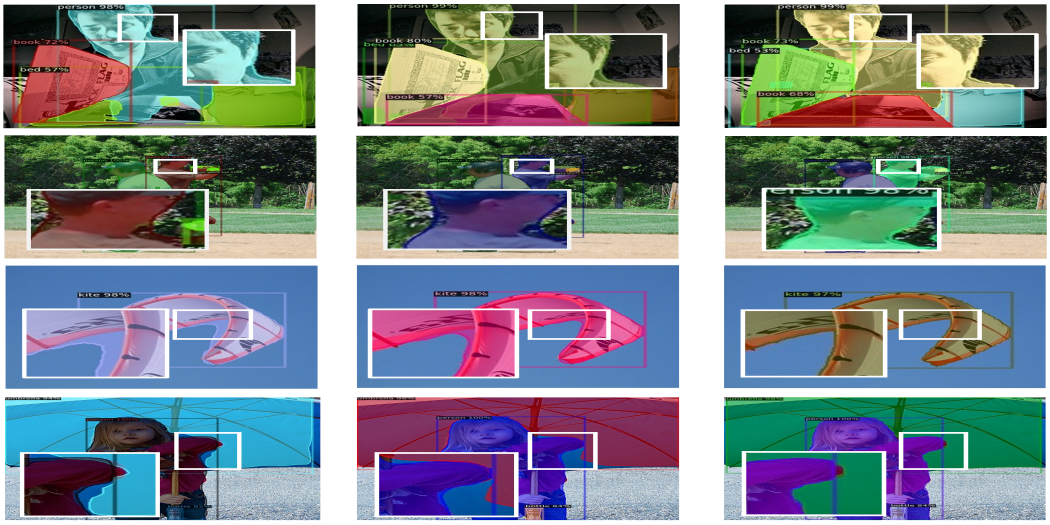
Table 13: Mask AP obtained by PatchDCT with feature maps cropped from all levels and from P2 only on COCO *val2017*. Model with feature maps of P2 obtain higher mAP.

Feature	AP	AP_S	AP_M	AP_L	AP_B	AP^*	AP^*_B
P2	**37.2**	**18.3**	**39.5**	**54.1**	**24.5**	**40.8**	**30.1**
P2-P5	37.1	18.2	39.3	53.3	24.4	40.6	29.8

7 shows that refining non-mixed regions with DCT-regressor will be lower than the method using three-class classifiers by 0.6AP and 1.2 AP^*. Need to note that, AP_B and AP^*_B decreases 0.9 and 1.5 respectively, since learning to regress non-mixed patches affects the prediction of bounded regions.

Use of the regressor. The regressor generates finer boundaries of masks that greatly improve the performance of PatchDCT. We prove the necessity of the regressor by removing the regressor and only refining foregound and background patches by the classifier. The results in Table 8 show that without the regressor both mAP and Boundary AP fall sharply. Although only a short n-dimensional vector needs to be regressed for each mixed patch, a high-resolution boundary is obtained by decoding each DCT vector during inference time. Meanwhile, the resolution of mask is almost equivalent to the resolution of boundary. We obtain high-resolution masks by generating high-resolution boundaries from DCT vectors of mixed patches. Thus the regressor is essential for PatchDCT.

Multi-stage PatchDCT. We compare the performance of multi-stage procedure (Table 9). One-Stage PatchDCT already provides high-quality masks, while two-stage PatchDCT further improves the prediction. However, the computational cost of the mask branch has nearly doubled with small improvements of masks, so we choose to use one-stage PatchDCT in our paper.



(a) Mask-RCNN (b) DCT-Mask (c) PatchDCT

Figure 4: COCO example tuples from Mask-RCNN, DCT-Mask, and PatchDCT. Mask-RCNN, DCT-Mask and PatchDCT are trained based on R50-FPN. PatchDCT provides masks with higher quality and finer boundaries.

Size of the patch. We evaluate the influence of patch size in Table 10. We keep the resolution of masks and the size of input feature map unchanged and compare the model performance with different patch sizes. The performance on 8×8 patches achieves the best performance.

Size of the feature map. We compare the model with different sizes of the feature map used in PatchDCT. The results in Table 11 illustrate that the performance saturates with 42×42 feature maps.

Dimension of PatchDCT vectors We analyze PatchDCT with different dimensions of DCT vectors. Results listed in Table 12 shows that models with 6-dimensional DCT vectors obtain the best performance. As shown by Table 1, 6-dimensional DCT vector already contains most of the ground truth information. As more elements bring only very little incremental information, regressing these elements does not improve the prediction.

Feature map from FPN We evaluate PatchDCT with the feature map cropped from all pyramid levels and from P2. The result in Table 13 shows that PatchDCT benefits from the finer feature map of P2.

4.5 Qualitative Results

In Figure 4 we visualize the outputs of PatchDCT on COCO *val2017*. PatchDCT generates finer boundaries among different instances, which helps PatchDCT to obtain masks of higher quality in comparison with Mask-RCNN and DCT-Mask.

5 Conclusions

In this work, we analyze the difficulties in refining high-resolution masks and propose PatchDCT, a effective refinement module that generates high-quality masks with finer boundaries. PatchDCT contains a three-class classifier and a DCT vector regressor. The three-class refines foregound and background patches, and helps the regressor to focus on boundaries of masks. By predicting DCT vectors of mixed patches, the regressor genenrates high-resolution boundaries. PatchDCT can be further used iteratively to further improve the quality of masks. We hope that our approach will benefit the future study in instance segmentation.

REFERENCES

Nasir Ahmed, T_ Natarajan, and Kamisetty R Rao. Discrete cosine transform. *IEEE transactions on Computers*, 100(1):90–93, 1974.

Muzhir Shaban Al-Ani and Fouad Hammadi Awad. The jpeg image compression algorithm. *International Journal of Advances in Engineering & Technology*, 6(3):1055, 2013.

Zhaowei Cai and Nuno Vasconcelos. Cascade r-cnn: Delving into high quality object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 6154–6162, 2018.

Kai Chen, Jiangmiao Pang, Jiaqi Wang, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jianping Shi, Wanli Ouyang, et al. Hybrid task cascade for instance segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4974–4983, 2019.

Bowen Cheng, Ross Girshick, Piotr Dollár, Alexander C Berg, and Alexander Kirillov. Boundary iou: Improving object-centric image segmentation evaluation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15334–15342, 2021.

Tianheng Cheng, Xinggang Wang, Lichao Huang, and Wenyu Liu. Boundary-preserving mask r-cnn. In *European conference on computer vision*, pp. 660–676. Springer, 2020.

Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3213–3223, 2016.

Bin Dong, Fangao Zeng, Tiancai Wang, Xiangyu Zhang, and Yichen Wei. Solq: Segmenting objects by learning queries. *Advances in Neural Information Processing Systems*, 34:21898–21909, 2021.

Agrim Gupta, Piotr Dollar, and Ross Girshick. Lvis: A dataset for large vocabulary instance segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 5356–5364, 2019.

Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pp. 2961–2969, 2017.

Zhaojin Huang, Lichao Huang, Yongchao Gong, Chang Huang, and Xinggang Wang. Mask scoring r-cnn. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 6409–6418, 2019.

Ba Rom Kang, Hyunku Lee, Keunju Park, Hyunsurk Ryu, and Ha Young Kim. Bshapenet: Object detection and instance segmentation with bounding shape masks. *Pattern Recognition Letters*, 131:449–455, 2020.

Lei Ke, Yu-Wing Tai, and Chi-Keung Tang. Deep occlusion-aware instance segmentation with overlapping bilayers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 4019–4028, 2021.

Lei Ke, Martin Danelljan, Xia Li, Yu-Wing Tai, Chi-Keung Tang, and Fisher Yu. Mask transfiner for high-quality instance segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4412–4421, 2022.

Alexander Kirillov, Yuxin Wu, Kaiming He, and Ross Girshick. Pointrend: Image segmentation as rendering. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 9799–9808, 2020.

Yi Li, Haozhi Qi, Jifeng Dai, Xiangyang Ji, and Yichen Wei. Fully convolutional instance-aware semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2359–2367, 2017.

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*, pp. 740–755. Springer, 2014.

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*, pp. 2117–2125, 2017.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015.

Xing Shen, Jirui Yang, Chunbo Wei, Bing Deng, Jianqiang Huang, Xian-Sheng Hua, Xiaoliang Cheng, and Kewei Liang. Dct-mask: Discrete cosine transform mask representation for instance segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8720–8729, 2021.

Max Welling and Thomas N Kipf. Semi-supervised classification with graph convolutional networks. In *J. International Conference on Learning Representations (ICLR 2017)*, 2016.

Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. https://github.com/facebookresearch/detectron2, 2019.

Gang Zhang, Xin Lu, Jingru Tan, Jianmin Li, Zhaoxiang Zhang, Quanquan Li, and Xiaolin Hu. Refinemask: Towards high-quality instance segmentation with fine-grained features. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 6861–6869, 2021.

Rufeng Zhang, Zhi Tian, Chunhua Shen, Mingyu You, and Youliang Yan. Mask encoding for single shot instance segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10226–10235, 2020.

A More Technical Details

We prove that all elements except the DCCs for foreground patches are zero.

Note that for a $m \times m$ patch $M^f_{m\times m}(u,v)$ Equation 1 can be written as

$$M^f_{m\times m}(u,v) = \frac{2}{m}C(u)C(v)\left(\sum_{x=0}^{m-1} A(x,u)\right)\left(\sum_{y=0}^{m-1} A(y,v)\right) \tag{7}$$

where $A(a,b) = cos\frac{(2a+1)b\pi}{2m}$.

If u is odd,

$$\begin{aligned} A(m-1-x,u) &= cos\frac{(2(m-1-x)+1)u\pi}{2m} \\ &= cos\left(-\frac{(2x+1)u\pi}{2m} + u\pi\right) \\ &= -A(x,u) \end{aligned} \tag{8}$$

If u is even, since

$$e^{2k\pi} = cos(2k\pi) + isin(2k\pi) = 1, \quad \forall k \in \mathbb{Z}, \tag{9}$$

$$\begin{aligned} \sum_{x=0}^{m-1} A(x,u) &= \sum_{x=0}^{m-1} cos\frac{(2x+1)u\pi}{2m} \\ &= Re\left(\sum_{x=0}^{m-1} e^{\frac{(2x+1)u\pi}{2m}}\right) \\ &= Re\left(e^{\frac{u\pi}{2m}}\frac{1-e^{u\pi}}{1-e^{\frac{u\pi}{m}}}\right) = 0. \end{aligned} \tag{10}$$

We obtain

$$\sum_{x=0}^{m-1} A(x,u) = 0, \quad \forall u \neq 0, \tag{11}$$

and for foreground patches

$$M^f_{m\times m}(i,j) = \begin{cases} m, & i=0, j=0, \\ 0, & otherwise. \end{cases} \tag{12}$$

This illustrates except the DCCs, elements of DCT vectors of foreground patches are all zero.