Abstract

This paper is about few-shot instance segmentation, where training and test image sets do not share the same object classes. We specify and evaluate a new few-shot anchor-free part-based instance segmenter (FAPIS). Our key novelty is in explicit modeling of latent object parts shared across training object classes, which is expected to facilitate our few-shot learning on new classes in testing. We specify a new anchor-free object detector aimed at scoring and regressing locations of foreground bounding boxes, as well as estimating relative importance of latent parts within each box. Also, we specify a new network for delineating and weighting latent parts for the final instance segmentation within every detected bounding box. Our evaluation on the benchmark COCO-201 dataset demonstrates that we significantly outperform the state of the art.

1. Introduction

This paper addresses the problem of few-shot instance segmentation. In training, we are given many pairs of support and query images showing instances of a target object class, and the goal is to produce a correct instance segmentation of the query given access to the ground-truth instance segmentation masks of the supports. In testing, we are given only one or a very few support images with their ground-truth segmentation masks, and a query image in which we are supposed to segment all instances of the target class. Importantly, the training and test image sets do not share the same object classes. Few-shot instance segmentation is a basic vision problem. It appears in many applications where providing manual segmentations of all object instances is prohibitively expensive. The key challenge is how to conduct a reliable training on small data.

Fig. 1 illustrates a common framework for few-shot instance segmentation that typically leverages Mask-RCNN [12, 43, 29]. First, the support and query images are input to a backbone CNN and feature pyramid network (FPN) [23] for computing the support’s and query’s feature maps. Second, for every feature map and every support’s segmentation mask, the masked average pooling computes the support’s feature vector. Third, the support’s feature vector is used to modulate the query’s feature maps through a channel-wise multiplication, resulting in the conditional query features, which are then input to Mask-RCNN for instance segmentation of the query. Our approach replaces Mask-RCNN with two new modules for anchor-free object detection, and part-based instance segmentation.

This framework has limitations. First, Mask-RCNN is anchor-based, and hence might overfit to particular sizes and aspect ratios of training objects, which do not characterize new classes in testing. Second, the Mask-RCNN learns feature prototypes [12] which are correlated with the feature map from the backbone in order to produce object segmentation. However, the prototypes typically capture global outlines of objects seen in training [12], and hence may not be suitable for segmenting new object classes with entirely...
Unlike the prototypes of not be meaningful (as there is no ground truth for parts).
ject parts, which are smaller object components but may parts shared across all object classes seen in training.
bounding box by localizing a universal set of latent object
trating boxes of the target object class defined by the support
Figure 2. Our F APIS uses the same feature maps as in Fig. 1, and extends prior work with two new modules: anchor-free object detector (AFD) and part-based instance segmentor (PIS). The AFD produces three types of dense predictions for every location \((x, y)\) in the feature map: (a) Figure-ground (FG) classification score; (b) Location of the closest bounding box to \((x, y)\); (c) Relative importance of the latent parts for segmentation of the bounding box closest to \((x, y)\). The PIS consists of the PartNet and Part Assembling Module (PAM). The PartNet predicts activation maps of the latent parts. After NMS selects the top scoring bounding boxes, for every box \(n\), the PAM fuses the part-activation maps according to the predicted part importance for the box \(n\), resulting in the instance segmentation \(m_n\).

different shapes in testing.

To address these limitations, we propose FAPIS – a new few-shot anchor-free part-based instance segmentor, illustrated in Fig. 2. In a given query, FAPIS first detects bounding boxes of the target object class defined by the support image and its segmentation mask, and then segments each bounding box by localizing a universal set of latent object parts shared across all object classes seen in training.

Our key novelty is in explicit modeling of latent object parts, which are smaller object components but may not be meaningful (as there is no ground truth for parts). Unlike the prototypes of [12], our latent parts capture certain smaller components of objects estimated as important for segmentation. As parts may be shared by distinct objects, including new object classes of testing, we expect that accounting for parts will lead to a more reliable few-shot learning than the aforementioned common framework. We are not aware of prior work that learns latent parts for few-shot instance segmentation.

We make two contributions. First, we specify a new anchor-free object detector (AFD) that does not use a set of candidate bounding boxes with pre-defined sizes and aspect ratios, called anchors, and, as shown in [39], in this way mitigates over-fitting to a particular choice of anchors. The AFD (the orange box in Fig. 2) is aimed at three tasks at every location of the query’s feature map: dense scoring and regressing locations of foreground bounding boxes, as well as dense estimation of a relative importance of the latent parts for segmentation. While all of the latent parts are learned to be relevant for object segmentation, differences in sizes and shapes across instances will render some latent parts more important than some others for segmentation of each instance. Thus, the third head in the AFD estimates the part importance which varies across the predicted bounding boxes in the image, as desired. The AFD’s output is passed to the standard non-maximum suppression (NMS) for selecting top scoring bounding boxes.

Second, we specify a new Part-based instance segmenter (PIS). The PIS (the yellow box in Fig. 2) is aimed at localizing and integrating latent object parts to produce the final instance segmentation within every NMS-selected bounding box. The PIS consists of the PartNet and part assembling module (PAM). The PartNet predicts activation maps of the latent parts, called part maps, where large activations
strongly indicate the presence of the corresponding part in the image. Importantly, high activations of a certain latent part at image location \((x, y)\) may not be important for segmentation of the object instance at \((x, y)\) (e.g., when several latent parts overlap but do not “cover” the full spatial extent of the instance). Therefore, for every NMS-selected bounding box, these part maps are then integrated by the PAM so as to account for the predicted relative importance of the parts for that box. Finally, all instance segmentations form the output query segmentation mask.

Our evaluation on the COCO-20 dataset [29] demonstrates that we significantly outperform the state of the art.

In the following, Sec. 2 reviews prior work; Sec. 3 specifies our deep architecture; Sec. 4 presents our implementation details and experimental results; and Sec. 5 describes our concluding remarks.

2. Related Work

Few-shot semantic segmentation labels pixels in the query with target classes, each defined by \(K\) support examples [36, 32, 33, 6, 47, 38, 28, 14, 37, 46, 45, 44, 31, 41]. Our problem is arguably more challenging than few-shot semantic segmentation, since we need to additionally distinguish object instances of the same class.

Instance segmentation localizes instances of object classes seen in training, whereas we are supposed to segment instances of new classes. There are proposal-based and proposal-free approaches. The former [12, 22, 4, 27, 1] first detects object bounding boxes, and then segments foreground within every box. The latter [16, 17, 2, 8] typically starts from a semantic segmentation, and then leverages certain visual cues (e.g., object center, or Watershed energy) to cluster pixels of the same semantic class into instances.

Our FAPIS and YOLACT [1] are similar in that they both predict activation maps of certain object features, and at each location in the image they fuse these activation maps by weighting their relative importance for segmentation. However, since YOLACT is not aimed at the few-shot setting, there are a number of significant differences. First, our FAPIS models latent object parts, whereas YOLACT models object prototypes representing global outlines of groups of similar object shapes. While the latter has been argued as suitable for instance segmentation, our experiments demonstrate that the prototypes learned on training examples poorly represent new, differently shaped object classes in testing for few-shot instance segmentation. As our latent parts may be components of the new object classes, they are likely to better represent new objects than the global shapes of prototypes. Second, YOLACT is standard anchor-based detector, whereas our AFD object detector is anchor-free producing dense predictions of bounding boxes. Our approach is more suitable for few-shot instance segmentation, as pre-defined anchors in training may not represent well the sizes and aspect ratios of new objects in testing.

Few-shot instance segmentation approaches usually adapt methods for instance segmentation (e.g., Mask-RCNN [12]) to the few-shot setting [29, 43, 7] (see Fig. 1). Our main differences are in replacing the Mask-RCNN with our AFD and PID for modeling latent object parts and segmenting every detected instance by localizing and assembling latent parts relevant for that instance.

Anchor-free Object Detection [39, 48, 19] predicts bounding boxes for all pixels in the feature map. This is opposite to anchor-based approaches [34, 24] where a set of bounding boxes with pre-defined sizes and aspect ratios are classified as presence or absence. Our AFD follows the FCOS approach [39], where at each location \((x, y)\) distances to the top, bottom, left, right sides of the closest bounding box are predicted by regression. We extend FCOS with SimNet to more reliably score bounding boxes, and thus reduce the number of false positives.

Part-based segmentation. While the literature abounds with approaches that use parts for image classification and object detection, there is relatively scant work on part-based segmentation. In [40, 10] the deformable part model [9] is used for semantic segmentation, and in [26] reasoning about “left”, “right”, “bottom” and “top” parts of the foreground object is shown to help instance segmentation. Recent work [21, 11] addresses the problem of human (one class) segmentation, not our multi-class segmentation problem. We are not aware of prior work that learns latent parts for few-shot instance segmentation.

3. Our Approach

3.1. Problem Statement

In training, we are given many pairs of support and query images showing the same target class from the class set \(C_1\), along with their pixel-wise annotations of every instance of the target class. In testing, we are given \(K\) support images, \(\{x_s\}\), and their \(K\) instance segmentation masks \(\{m_s\}\) that define a target class sampled from the class set \(C_2\), where \(C_1 \cap C_2 = \emptyset\). Given a test query, \(x_q\), showing the same target class as the test supports, our goal is to segment all foreground instances in \(x_q\), i.e., estimate the query instance segmentation mask \(m_q\). This problem is called 1-way \(K\)-shot instance segmentation. In this paper, we consider \(K = 1\) and \(K = 5\), i.e., test settings with a very few supports. It is worth noting that we can easily extend the 1-way \(K\)-shot to \(N\)-way \(K\)-shot by running \(N\) support classes on the same query, since every class can be independently detected. However, the \(N\)-way \(K\)-shot problem is beyond the scope of this paper.
3.2. Multi-level Feature Extraction

As shown in Fig. 1, FAPIS first extracts multi-level feature maps at different resolutions from $x_1$ and $x_2$ using a backbone CNN with feature pyramid network (FPN), as in [23]. For each level $i = 3, 4, \ldots, 7$, the support feature map $F_{s, i}$ is masked-average pooled within a down-sampled version of the support mask $m_s$ to obtain the support feature vector $f_{s, i}$. Then, for each level $i$, $f_{s, i}$ is used to modulate the corresponding query feature map $F_{q, i}$. Specifically, $f_{s, i}$ and $F_{q, i}$ are multiplied channel-wise. This gives the conditional query feature map $F_{q, i}^\prime$. The channel-wise multiplication increases (or decreases) activations in the query’s feature maps at different resolutions from $F$, when the corresponding support’s activations are high (or low). In this way, the channel features that are relevant for the target class are augmented, and irrelevant features are suppressed to facilitate instance segmentation.

3.3. The Anchor-free Object Detector

For each conditional query feature map $F_{q, i}^\prime$, $i = 3, 4, \ldots, 7$, the AFD scores and regresses locations of foreground bounding boxes. Below, for simplicity, we drop the notation $i$ for the feature level, noting that the same processing is done for each level $i$. The workflow of the AFD is illustrated in the orange box in Fig. 2.

For every location $(x, y)$ in $F_q^\prime$ with height and width $H \times W$, the AFD predicts:

1. Figure-ground (FG) classification scores $C = \{c_{x,y} \} \in [0,1]^{H \times W}$ using the SimNet and Classification Head;
2. Regressions $T = \{t_{x,y}\} \in \mathbb{R}^{H \times W \times 4}$ of top, bottom, left, right distances from $(x, y)$ to the closest box.
3. Relative importance of the $J$ latent parts for instance segmentation of the bounding box predicted at $(x, y)$, $\Pi = \{\pi_{x,y} \} \in \mathbb{R}^{H \times W \times J}$.

**SimNet and Classification Head.** Prior work [43, 29], uses the support feature vector to modulate the query feature map by channel-wise multiplication, and thus detects target objects. However, in our experiments, we have observed that this method results in a high false-positive rate. To address this issue, we specify the SimNet which consists of a block of fully connected layers followed by a single convolutional layer. The first block takes as input $f_s$ and predicts weights of the top convolutional layer. This is suitable for addressing new classes in testing. Then, the top convolutional layer takes as input $F_q^\prime$ and the resulting feature maps are passed to the FG head for predicting the FG classification scores $C$ at every location $(x, y)$. In this way, the SimNet learns how to more effectively modulate $F_q^\prime$, extending the channel-wise multiplication in [43, 29].

For training the FG head, we use the focal loss [24]:

$$L_c = \frac{-1}{H \times W} \sum_{x,y} \alpha_{x,y}(1 - c_{x,y})^\gamma \log(c_{x,y}),$$

where $c_{x,y} = c_{x,y}^* c_{x,y} + (1 - c_{x,y}^*)(1 - c_{x,y})$, $c_{x,y}^* \in \{0, 1\}$ is the ground-truth classification score at $(x, y)$; $\alpha_{x,y} = c_{x,y}^* \alpha + (1 - c_{x,y}^*)(1 - \alpha)$, $\alpha \in [0, 1]$ is a balance factor of classes, and $\gamma \in [0, 5]$ is a focusing parameter to smoothly adjust the rate at which easy examples are down-weighted.

**Bounding Box Regression.** For training the box regression head, we use the general intersection-over-union (GIOU) loss [35]:

$$L_b = \frac{1}{N_{pos}} \sum_{x,y} I(c_{x,y}^* = 1) \cdot \text{GIOU}(t_{x,y}, t_{x,y}^*),$$

where $t_{x,y}^* \in \mathbb{R}^4_+$ is a vector of the ground-truth top, bottom, left, right box distances from $(x, y)$; $I(\cdot)$ is the indicator function having value 1 if $(x, y)$ belongs to the ground-truth box and 0 otherwise, and $N_{pos} = \sum_{x,y} I(c_{x,y}^* = 1)$ is the number of locations in the feature map that belong to the foreground.

**Part-Importance Prediction Head.** For every location $(x, y)$ in $F_q^\prime$, this head predicts the relative importance of the latent parts $\Pi = \{\pi_{x,y} \} \in \mathbb{R}^{H \times W \times J}$. This part-importance prediction seeks to capture the varying dependencies among the latent parts when jointly used to estimate a segmentation mask of the bounding box predicted for every $(x, y)$. Note that we are not in a position to specify an explicit loss for training this head, because the parts are latent, and not annotated in our training set. Importantly, we do get to train this head by backpropagating the instance segmentation loss, as described in Sec. 3.4.

The AFD’s predictions are forwarded to the standard NMS to select the top scoring $n = 1, \ldots, N$ bounding boxes at all feature-pyramid levels.

3.4. The Part-based Instance Segmenter

The PIS is illustrated in the yellow box in Fig. 2. The PIS consists of the PartNet and the PAM.

**The PartNet** takes as input the largest conditional query feature map – specifically, $F_{q, 3}$ – and predicts activation maps of the $J$ latent parts, or part maps for short, $P \in \mathbb{R}^{H \times W \times J}$. $J$ is a hyper-parameter that we experimentally set to an optimal value. High activations in $P$ for a particular latent part at location $(x, y)$ strongly indicate that part’s presence at $(x, y)$. For every bounding box $n = 1, \ldots, N$ selected by the NMS, we conduct region-of-interest (ROI) aligned pooling of $P$ to derive the $J$ ROI-aligned pooled part maps $P_n \in \mathbb{R}^{H_i \times W_i \times J}$, where $H_i$ and $W_i$ are the reference height and width ensuring that the pooled features of the $N$ bounding boxes have the same reference dimension. The PartNet is learned by backpropagating the instance segmentation loss through the PAM module, since the latent parts are not annotated and we cannot define an explicit loss for training the PartNet.

A strongly indicated presence of a certain latent part at location $(x, y)$ in $P_n$ may not be important for segmentation...
of the object instance in the bounding box $n$ (e.g., when this information is redundant as many other latent parts may also have a high activation at $(x, y)$ in $P_n$). For each location, we obtain a triplet of classification scores, bounding box and part importance $\pi_n$. Therefore, for every bounding box $n = 1, \ldots, N$, the following PAM module takes as input both the pooled part maps $P_n \in \mathbb{R}^{H_r \times W_r \times J}$ and the estimated part importance $\pi_n \in \mathbb{R}^J$ for segmentation of instance $n$.

The PAM computes the segmentation mask of every bounding box $n = 1, \ldots, N$, $m_n \in [0, 1]^{H_r \times W_r}$, as

$$m_n = P_n^+ \otimes \sigma(\pi_n), \quad P_n^+ = \text{MaxNorm}(\text{ReLU}(P_n)), \quad (3)$$

where $\text{MaxNorm}(A) = \max_{x,y} A_{x,y}$ for an activation map $A$, $\otimes$ denotes a inner product, and $\sigma$ is the sigmoid function. Note that $\otimes$ in (3) requires the tensor $P_n^+$ be rectified to a matrix of size $(H_rW_r) \times J$. Thus, the operator $\otimes$ serves to fuse the part maps by their respective importance for instance segmentation.

The MaxNorm-ReLU composite function and the sigmoid in (3) make the part maps and their importance non-negative with values in $[0, 1]$. This design choice is inspired by the well-known non-negative matrix factorization (NMF) [20], so that if the instance segmentation matrix $m_n$ were available, the expression in (3) would amount to the NMF of $m_n$ into the non-negative $P_n^+$ and $\sigma(\pi_n)$. This design choice is conveniently used to regularize learning of the PartNet and the AFD’s head for predicting the part importance. Specifically, in learning, we maximize similarity between the predicted $P_n^+$ and the NMF’s bases computed for the ground-truth instance segmentation masks $\{m_n^* : n = 1, \ldots, D\}$, where $D$ is the total number of instances in the training set. This NMF-based regularization in learning enforces that $P_n^+$ is sparse, justifying our interpretation that $P_n^+$ represents the smaller latent parts of object instances.

**Segmentation loss and NMF-based regularization.**

The prediction $m_n$, given by (3), incurs the following segmentation loss $L_s$ with respect to the ground-truth instance segmentation mask $m_n^*$:

$$L_s = \frac{1}{N} \sum_{n=1}^{N} l_{dice}(m_n, m_n^*), \quad (4)$$

$$l_{dice}(A, B) = 1 - \frac{2 \sum_{x,y} A_{x,y}B_{x,y}}{\sum_{x,y} A_{x,y} + \sum_{x,y} B_{x,y}},$$

where $l_{dice}$ is the dice loss [30] between maps $A$ and $B$.

$L_s$ is backpropagated through the PAM to the PartNet and AFD for improving the predictions $P$ and $\Pi$ such that $L_s$ is minimized. We regularize this learning by using the NMF of the ground-truth instance segmentation masks as follows. All ground-truth segmentations of all object classes in the training dataset $\{m_n^* : n = 1, \ldots, D\}$ are first re-sized and stacked into a large $(H_rW_r) \times D$ matrix $M^*$.

Then, we apply the NMF as $M^* \approx P^*U$, where $P^*$ is the non-negative basis matrix with size $(H_rW_r) \times J$, and $U$ is the non-negative weight matrix with size $J \times D$. The product $P^*U$ is a low-rank approximation of $M^*$. Due to the non-negative constraint, $P^*$ is forced to be sparse. Fig. 3 shows the NMF’s bases in $P^*$ that we computed on the ground-truth masks $\{m_n^*\}$ of the COCO-20$^i$ dataset [29]. Hence, $P^*$ is interpreted as storing parts of all training object classes in $M^*$. Conveniently, the NMF, i.e., $P^*$, can be pre-computed before learning.

We use $P^*$ to regularize our learning of the PartNet such that it produces part maps $P_n^*$ that are similar to $P^*$. Note that the $J$ latent parts in $P_n^*$ may be differently indexed from the NMF’s $J$ bases stored in $P^*$. Therefore, we employ the Hungarian algorithm [18] to identify the one-to-one correspondence between the latent parts $P_n^{+}\|\pi_n$ and the bases $P_n^{+}\|\pi_n$, $j'$ = Hungarian($j$), $j = 1, \ldots, J$, based on their intersection-over-union (IoU) score. With this correspondence, we specify the NMF regularization loss as

$$L_{\text{NMF}} = \frac{1}{N} \sum_{n=1}^{N} \sum_{j=1}^{J} l_{dice}(P_n^{+}\|\pi_n, P_n^{+}\|\pi_{j'}), \quad j' = \text{Hungarian}(j). \quad (5)$$

![Figure 3. Visualization of 16 NMF parts estimated from the size-normalized ground-truth instance masks in COCO-20$^i$ [29].](image)

The total loss for training our FAPIS is specified as

$$L = \lambda_1L_c + \lambda_2L_b + \lambda_3L_s + \lambda_4L_{\text{NMF}} \quad (6)$$

where the $\lambda$s are positive coefficients, specified in Sec. 4.1.

## 4. Experiments

**Dataset & Metrics:** For evaluation, we conduct the standard 4-fold cross-validation on the benchmark COCO-20$^i$ dataset [29]. In each fold $i = 0, \ldots, 3$, we sample 20 test classes from the 80 object classes in MSCOCO [25], and use the remaining 60 classes for training. For each COCO-20$^i$, we randomly sample five separate test sets, and report the average results as well as their standard deviations, as in [29]. We follow the standard testing protocol for 1-way $K$-shot instance segmentation [29]. Our training uses $K = 1$.
support image, while in testing we consider two settings with \( K = 1 \) and \( K = 5 \) support images. For each test query, \( K \) support images are randomly sampled from the remaining test images showing the same test class as the query. We use the evaluation tool provided by the COCO dataset to compute the following metrics: mAP50 of both segmentation and detection is computed at a single intersection-over-union (IoU) threshold of 50\% between our prediction and ground truth; mAR10 is average recall given 10 detections per image. Our results of other metrics for each fold can be found in the supplementary material.

4.1. Implementation Details

Our backbone CNN is ResNet-50 [13] and the FPN of [23], as in prior work. FAPIS is implemented using the mmdetection toolbox [3]. We empirically find that using the P3 to P5 feature map levels gives the best results. The FG scoring and part-importance prediction share the first two convolutional layers with GroupNorm [42] and ReLU activation in between, but use two separate heads for binary class and part-importance predictions. We strictly follow the design of classification head and box regression head of FCOS [39]. SimNet has a block of four fully connected layers with BatchNorm [15] and ReLU in between each layer, followed by a top convolutional layer, where the block predicts the weights of the top layer. The PartNet consists of 5 convolutional layers with GroupNorm and ReLU, and an Upsampling layer to upscale the resolution by two in the middle of layer 3 and 4, following the design of mask head of Mask-RCNN [12]. For learning, we use SGD with momentum [5] with the learning rate of \( 1 \times 10^{-3} \). The number of training epochs is 12, which is similar to the setting of 1x in Mask-RCNN. The mini-batch size is 16. The query images are resized to 800×1333 pixels. The support images and masks are cropped around the ground-truth bounding boxes and re-sized to 128×128 pixels. We set \( H_r = W_r = 32, \alpha = 0.25 \) and \( \gamma = 2 \) in (1), and \( \lambda_1 = \lambda_2 = \lambda_3 = 1, \lambda_4 = 0.1 \) in (6).

4.2. Ablation Study

Ablations and sensitivity to input parameters are evaluated for the setting \( K = 1 \).

**Analysis of the number of parts.** Tab. 1 reports how the number of latent parts \( J \) affects our results. We vary \( J \) while keeping other hyper-parameters unchanged. From Tab. 1, \( J = 16 \) gives the best performance. When \( J \) is small, FAPIS cannot reliably capture object variations, and when \( J \) approaches the number of classes considered in training, the PartNet tends to predict shape prototypes as in YOLACT [1], instead of parts. Therefore, we use \( J = 16 \) for all other evaluations.

**Analysis of the predicted part importance.** Tab. 2 shows a percentage of the latent parts whose predicted im-

<table>
<thead>
<tr>
<th># parts</th>
<th>1</th>
<th>2</th>
<th>4</th>
<th>8</th>
<th>16</th>
<th>32</th>
<th>64</th>
</tr>
</thead>
<tbody>
<tr>
<td>mAP50</td>
<td>16.3</td>
<td>17.2</td>
<td>17.9</td>
<td>18.4</td>
<td>18.8</td>
<td>18.5</td>
<td>18.0</td>
</tr>
</tbody>
</table>

Table 1. mAP50 for segmentation of FAPIS on COCO-20\( ^0 \) for different numbers of the latent parts \( J \) used.

<table>
<thead>
<tr>
<th>( \theta )</th>
<th>0.1</th>
<th>0.3</th>
<th>0.5</th>
<th>0.7</th>
<th>0.9</th>
</tr>
</thead>
<tbody>
<tr>
<td>( % ) parts with ( \sigma(\pi) &gt; \theta )</td>
<td>0.32</td>
<td>0.29</td>
<td>0.26</td>
<td>0.21</td>
<td>0.09</td>
</tr>
</tbody>
</table>

Table 2. The predicted importance of most latent parts is on average lower than a threshold \( \theta \in (0, 1) \), so instance segmentation is formed from only a few important parts.

<table>
<thead>
<tr>
<th>Ablations</th>
<th>Object detection</th>
<th>Instance segmentation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>mAP50</td>
<td>mAR10</td>
</tr>
<tr>
<td>FIS</td>
<td>18.4</td>
<td>25.0</td>
</tr>
<tr>
<td>FAIS</td>
<td><strong>20.9</strong></td>
<td><strong>28.1</strong></td>
</tr>
<tr>
<td>FAIS-SimNet</td>
<td>20.2</td>
<td>26.9</td>
</tr>
<tr>
<td>FAIS-CWM</td>
<td>16.5</td>
<td>24.3</td>
</tr>
<tr>
<td>FPIS</td>
<td>18.4</td>
<td>25.0</td>
</tr>
<tr>
<td>FAPIS-L( _{NMF} )</td>
<td>20.5</td>
<td>27.5</td>
</tr>
<tr>
<td>FAPIS</td>
<td><strong>20.9</strong></td>
<td><strong>28.1</strong></td>
</tr>
</tbody>
</table>

Table 3. mAP50 and mAR10 of one-shot object detection and instance segmentation on COCO-20\( ^0 \) for different variants of FAPIS. The PIS is used only for instance segmentation, so the variants FIS and FPIS have lower mAP50 for object detection. The AFD is used for both instance detection and segmentation, and thus FAIS and FAPIS have higher mAP50 for both object detection and instance segmentation than FIS and FPIS. The AFD without SimNet decreases its discriminative power so FAIS-SimNet has lower mAP50 for object detection than FAIS. When the channel-wise multiplication (CWM) is replaced with SimNet alone, performance of FAIS significantly decreases, and becomes even lower than that of FIS. This shows that SimNet does not replace but extends CWM. FAPIS-L\( _{NMF} \) has lower mAP50 for instance segmentation than FAPIS.

Portion for segmentation is higher than a threshold \( \theta \in (0, 1) \). As can be seen, for a given object instance, most latent parts are usually estimated as irrelevant. That is, the PAM essentially uses only a few most important latent parts to form the instance segmentation mask.

**Ablations of FAPIS.** Tab. 3 demonstrates how the AFD and PIS affect performance of the following six ablations:

- **FIS**: AFD is replaced with the region proposal network (RPN) [34] for anchor-based detection; and PIS is replaced with the standard mask-head of Mask-RCNN.
- **FAIS**: PIS replaced with mask-head of Mask-RCNN.
- **FAIS-SimNet**: FAIS without SimNet, and FG scores are predicted as in [39] (see Fig. 1).
- **FAIS-CWM**: FAIS without channel-wise multiplication (CWM).
- **FPIS**: AFD is replaced with RPN [34]
• FAPIS-$L_{NMF}$: FAPIS trained without the regularization loss $L_{NMF}$.

• FAPIS: our approach depicted in Fig. 2

The top three variants test our contributions related to anchor-free detection, and the last three test our contributions related to modeling the latent parts.

From Tab. 3, our AFD in FAIS improves performance by 1.6\% on mAP50 over FIS that uses the anchor-based detector. However, removing the SimNet from the AFD in FAIS-SimNet decreases performance, justifying our SimNet as a better way to predict FG scores than in [43, 29]. The PIS in FPIS gives a performance gain of 1\% on mAP50 in instance segmentation over FIS. Our performance decreases when FAPIS is trained without the NMF regularization in FAPIS-$L_{NMF}$. FAPIS gives the best performance in comparison with the strong baselines.

Note that we cannot directly evaluate our part detection, since we do not have ground-truth annotations of parts.

![Figure 4](image)

Figure 4. Visualization of the most important 10 latent parts out of 16 predicted for example instances of the “human” and “car” classes from the COCO-20\(^{0}\) validation dataset. From left to right: (a) input image, (b) GT segmentation, (c) predicted segmentation, (d) 10 most relevant parts. The predicted importance of the parts is color-coded from blue (smallest) to green (largest).

### 4.3. Comparison with prior work

FAPIS is compared with Meta-RCNN [43], Siamese Mask-RCNN [29] and YOLACT [1]. Meta-RCNN and Siamese Mask-RCNN use the same anchor-based framework, but differ in that Meta-RCNN performs the feature correlation after the RPN, whereas Siamese Mask-RCNN does this before the RPN. YOLACT is another anchor-based instance segmentor based on RetinaNet [24], which learns to predict object shapes instead of object parts, and we adapt it to few-shot instance segmentation by changing its feature extraction module as in Fig. 1.

Note that the results of Siamese Mask-RCNN in [29] are reported for evaluation where the support images do not provide ground-truth segmentation masks, but only bounding boxes. Therefore, for fairness, we evaluate Siamese Mask-RCNN (without any changes to their public code) on the same input as for Meta-RCNN, YOLACT, and FAPIS—i.e., when the support images provide ground-truth segmentation masks of object instances. In addition, in [43], the results of Meta-RCNN are reported for a different version of COCO. Therefore, we evaluate Meta-RCNN (without any changes to their public code) on our test sets.

Tab. 4 reports one-shot and five-shot object detection results, and Tab. 5 shows one-shot and five-shot instance segmentation results. From these tables, Siamese Mask-RCNN performs slightly better than Meta-RCNN, since the feature correlation happens before the RPN, enabling the RPN to better adapt its bounding box proposals to the target class. Our FAPIS outperforms YOLACT by 2\% for one-shot and five-shot instance segmentation. This demonstrates advantages of using latent parts and their layouts for assembling object shapes in our approach over the YOLACT’s prototypes in few-shot instance segmentation.

Our gains in performance are significant in the context of previous performance improvements reported by prior work, where gains were only by about 0.6\% by [29] vs [43], by 0.1\% by [29] vs [1], by 0.5 \% [1] vs [43]. We improve by 1.8\% and 2.0\% over [29] on one-shot and five-shot instance segmentation, respectively.

### 4.4. Qualitative Evaluation

Fig. 4 visualizes the most important 10 of 16 latent parts and their relative importance for a few example instances of the “human” and “car” classes from the COCO-20\(^{0}\) validation set. The figure shows that the predicted latent parts are smaller components of the objects, and that some parts may be assigned a meaningful interpretation. Our visualization of the latent parts for other classes can be found in the supplementary material.

A few representative success and failure results on COCO-20\(^{0}\) are illustrated in Fig. 5, and others are included in the supplementary material.

### 5. Conclusion

We have specified a new few-shot anchor-free part-based instance segmenter (FAPIS) that predicts latent object parts for instance segmentation, where part annotations are not available in training. FAPIS uses our new anchor-free object detector (AFD), PartNet for localizing the latent parts, and part assembly module for fusing the part activation maps by their predicted importance into a segmentation of every detected instance. We have regularized learning such that the identified latent parts and similar to the sparse bases of the
<table>
<thead>
<tr>
<th># shots</th>
<th>Method</th>
<th>COCO-20&lt;sup&gt;0&lt;/sup&gt;</th>
<th>COCO-20&lt;sup&gt;1&lt;/sup&gt;</th>
<th>COCO-20&lt;sup&gt;2&lt;/sup&gt;</th>
<th>COCO-20&lt;sup&gt;3&lt;/sup&gt;</th>
<th>mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>K=1</td>
<td>Meta-RCNN [43]</td>
<td>17.7 ± 0.7</td>
<td>19.2 ± 0.6</td>
<td>17.7 ± 0.3</td>
<td>21.1 ± 0.4</td>
<td>18.9</td>
</tr>
<tr>
<td></td>
<td>Siamese M-RCNN [29]</td>
<td>18.3 ± 0.8</td>
<td>19.5 ± 0.7</td>
<td>18.0 ± 0.4</td>
<td>21.5 ± 0.6</td>
<td>19.3</td>
</tr>
<tr>
<td></td>
<td>YOLACT [1]</td>
<td>18.0 ± 0.5</td>
<td>18.8 ± 0.5</td>
<td>17.8 ± 0.6</td>
<td>21.2 ± 0.7</td>
<td>19.0</td>
</tr>
<tr>
<td></td>
<td>FAPIS</td>
<td>20.9 ± 0.4</td>
<td>20.4 ± 0.1</td>
<td>20.0 ± 0.6</td>
<td>23.4 ± 0.5</td>
<td>21.2</td>
</tr>
<tr>
<td>K=5</td>
<td>Meta-RCNN [43]</td>
<td>19.1 ± 0.4</td>
<td>21.2 ± 0.2</td>
<td>19.6 ± 0.5</td>
<td>24.0 ± 0.2</td>
<td>21.0</td>
</tr>
<tr>
<td></td>
<td>Siamese M-RCNN [29]</td>
<td>20.0 ± 0.4</td>
<td>21.6 ± 0.3</td>
<td>20.2 ± 0.4</td>
<td>24.1 ± 0.3</td>
<td>21.5</td>
</tr>
<tr>
<td></td>
<td>YOLACT [1]</td>
<td>20.8 ± 0.4</td>
<td>21.1 ± 0.2</td>
<td>20.2 ± 0.5</td>
<td>24.8 ± 0.2</td>
<td>21.7</td>
</tr>
<tr>
<td></td>
<td>FAPIS</td>
<td>22.6 ± 0.3</td>
<td>22.8 ± 0.0</td>
<td>22.6 ± 0.6</td>
<td>26.4 ± 0.2</td>
<td>23.6</td>
</tr>
</tbody>
</table>

Table 4. mAP50 with standard deviation of one-shot and five-shot object detection on COCO-20<sup>i</sup>. The best results are in bold.

<table>
<thead>
<tr>
<th># shots</th>
<th>Method</th>
<th>COCO-20&lt;sup&gt;0&lt;/sup&gt;</th>
<th>COCO-20&lt;sup&gt;1&lt;/sup&gt;</th>
<th>COCO-20&lt;sup&gt;2&lt;/sup&gt;</th>
<th>COCO-20&lt;sup&gt;3&lt;/sup&gt;</th>
<th>mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>K=1</td>
<td>Meta-RCNN [43]</td>
<td>16.0 ± 0.6</td>
<td>16.1 ± 0.5</td>
<td>15.8 ± 0.3</td>
<td>18.6 ± 0.4</td>
<td>16.6</td>
</tr>
<tr>
<td></td>
<td>Siamese M-RCNN [29]</td>
<td>16.6 ± 0.8</td>
<td>16.6 ± 0.6</td>
<td>16.3 ± 0.7</td>
<td>19.3 ± 0.6</td>
<td>17.2</td>
</tr>
<tr>
<td></td>
<td>YOLACT [1]</td>
<td>16.8 ± 0.6</td>
<td>16.5 ± 0.5</td>
<td>16.1 ± 0.4</td>
<td>19.0 ± 0.6</td>
<td>17.1</td>
</tr>
<tr>
<td></td>
<td>FAPIS</td>
<td>18.8 ± 0.3</td>
<td>17.7 ± 0.1</td>
<td>18.2 ± 0.7</td>
<td>21.4 ± 0.4</td>
<td>19.0</td>
</tr>
<tr>
<td>K=5</td>
<td>Meta-RCNN [43]</td>
<td>17.4 ± 0.3</td>
<td>17.8 ± 0.2</td>
<td>17.7 ± 0.7</td>
<td>21.3 ± 0.2</td>
<td>18.6</td>
</tr>
<tr>
<td></td>
<td>Siamese M-RCNN [29]</td>
<td>17.5 ± 0.4</td>
<td>18.5 ± 0.1</td>
<td>18.2 ± 1.0</td>
<td>22.4 ± 0.2</td>
<td>19.2</td>
</tr>
<tr>
<td></td>
<td>YOLACT [1]</td>
<td>17.6 ± 0.2</td>
<td>18.4 ± 0.2</td>
<td>17.9 ± 0.6</td>
<td>21.8 ± 0.3</td>
<td>18.9</td>
</tr>
<tr>
<td></td>
<td>FAPIS</td>
<td>20.2 ± 0.2</td>
<td>20.0 ± 0.1</td>
<td>20.4 ± 0.7</td>
<td>24.3 ± 0.2</td>
<td>21.2</td>
</tr>
</tbody>
</table>

Table 5. mAP50 with standard deviation of one-shot and five-shot instance segmentation on COCO-20<sup>i</sup>. The best results are in bold.

non-negative matrix factorization of the ground-truth segmentation masks. Our evaluation on the benchmark COCO-20<sup>i</sup> dataset demonstrates that: we significantly outperform the state of the art; our prediction of latent parts gives better performance than using the YOLACT’s prototypes or using the standard mask-head of Mask-RCNN; and our anchor-free AFD improves performance over the common anchor-based bounding-box prediction.

Acknowledgement. This work was supported in part by DARPA XAI Award N66001-17-2-4029 and DARPA MCS Award N66001-19-2-4035.
References


