Learning Category- and Instance-Aware Pixel Embedding for Fast Panoptic Segmentation

Naiyu Gao, Student Member, IEEE, Yanhu Shan, Member, IEEE, Xin Zhao, Member, IEEE, and Kaiqi Huang, Senior Member, IEEE

Abstract—Panoptic segmentation (PS) is a complex scene understanding task that requires providing high-quality segmentation for both thing objects and stuff regions. Previous methods handle these two classes with semantic and instance segmentation modules separately, following with heuristic fusion or additional modules to resolve the conflicts between the two outputs. This work simplifies this pipeline of PS by consistently modeling the two classes with a novel PS framework, which extends a detection model with an extra module to predict category- and instance-aware pixel embedding (CIAE). CIAE is a novel pixel-wise embedding feature that encodes both semantic-classification and instance-distinction information. At the inference process, PS results are simply derived by assigning each pixel to a detected instance or a stuff class according to the learned embedding. Our method not only demonstrates fast inference speed but also the first one-stage method to achieve comparable performance to two-stage methods on the challenging COCO benchmark.

Index Terms—Panoptic Segmentation, Pixel Embedding.

I. INTRODUCTION

SCENE parsing is an important vision task and serves as the basis of many practical applications, including autonomous driving, robotics, and image editing. The recently proposed panoptic segmentation (PS) task [1] requires determining the semantic category of each pixel in an input image while identifying and segmenting each object instance. Moreover, PS segments both amorphous uncountable regions (stuff) and countable objects (thing), which provides more complete scene information when comparing with semantic segmentation or instance segmentation.

A straightforward solution for panoptic segmentation is to deal stuff and thing classes with semantic segmentation and instance segmentation branches, respectively, as in the existing methods [2]–[5]. The semantic segmentation branch outputs pixel-level semantic classification, while the instance segmentation branch detects instances and produces pixel-level instance-distinction. After that, the outputs from two branches are fused to get the final PS results, where heuristic rules or additional modules are developed to handle the conflicts between the two outputs. However, this complex pipeline brings heavy computation complexity and memory footprint, especially for the two-stage methods, where time-consuming instance segmentation modules are employed. Recently, one-stage methods [6]–[11] are developed, where the instance-distinction information is provided with pixel-level instance-sensitive features. Although having advantages in efficiency, the existing one-stage methods fail to achieve comparable performance to two-stage approaches.

Instead of modeling thing and stuff classes separately and handling the conflicts at the later fusing process, this work proposes to consistently model the two classes to simplify the whole pipeline. Specifically, we propose a novel PS framework that extends a one-stage detection model with an additional module to predict category- and instance-aware pixel embedding (CIAE), which encodes cues for both pixel-wise semantic-classification and instance-distinction. Different from the original instance-aware pixel embedding (IAE) [12]–[14], CIAE is specifically designed to map each pixel into a feature space where cues about semantic-classifying and instance-differentiating are modeled simultaneously. Concretely, the feature space is divided into category-specific sub-spaces.
by the learned category-aware query embeddings, and the pixels are mapped not only to distinguish instances but also into corresponding sub-spaces. In this way, the learned pixel embedding captures both category-classifying and instance-distinguishing information.

In the inference procedure, the PS result is simply derived by assigning each pixel to a detected instance or a stuff class according to its embedding feature. Specifically, instance proposals and the CIAE embedding map are firstly predicted with the detection and embedding branches, respectively. After that, query embeddings for each instance and stuff class are obtained. The former is derived from the predicted embedding map based on where the instance is generated, while the latter is a learned memory embedding. Finally, the PS result is obtained by calculating the cosine similarity between pixel embeddings and each query embedding, which is simply implemented as matrix multiplication.

Thanks to the consistency of modeling, our method not only simplifies the PS pipeline but also gains advantages in end-to-end learning and achieves good performance. Experiments on the challenging COCO [15] dataset demonstrate the benefits of our method. As shown in Figure 1, our model surpasses all one-stage methods in both accuracy and inference speed. We are also the first one-stage method to achieve comparable performance to two-stage methods on the COCO dataset.

Our contributions are summarized as the following:

- This work proposes the concept of category- and instance-aware pixel embedding (CIAE), which captures cues about semantic-classification and instance-distinction simultaneously to consistently model countable thing objects and uncountable stuff regions. The pipeline of panoptic segmentation is thus much simplified.

- This work demonstrates the feasibility of jointly modeling pixel-level semantic-classification and instance-distinction, instead of employing semantic segmentation and instance segmentation modules separately.

- This work proposes a novel one-stage panoptic segmentation framework by incorporating the proposed CIAE with a one-stage detector. On the COCO benchmark, the proposed model surpasses all one-stage methods in both accuracy and inference speed and is the first one-stage method that achieves comparable performance against time-consuming two-stage methods.

II. RELATED WORK

A. Instance Segmentation

Instance segmentation requires to detect and segment each object instance in an image. Recent methods can be divided into two categories: two-stage and one-stage methods. The first two-stage instance segmentation method Mask R-CNN [16] extends the two-stage detector Faster R-CNN [17] by adding a mask predicting branch. The following works [18]–[21] promote Mask R-CNN with advanced feature aggregation or mask prediction heads. These two-stage methods are showing great performance, but the feature re-pooling and re-sampling operations cause large computational overhead and greatly reduce the efficiency. The one-stage methods are also developed for the instance segmentation task. Some one-stage methods [22]–[25] follow the detect-then-segment manner and extend one-stage detectors [26], [27] to predict masks. Other methods design instance-aware features [7], [12]–[14], [28]–[32] and obtain instance masks through clustering during post-processing. The instance-aware features can be either explicit instance-aware features [28], [29], discriminative pixel embeddings [12]–[14], pixel-pair affinities [7], [30] or instance vectors [31], [32] toward object centroids.

Our method is inspired by the pixel embedding based methods [12]–[14] but differs in many aspects. Firstly, the learned pixel embeddings in previous works are only distinguishable to object instances but agnostic to semantic categories. In our method, the learned pixel embedding encodes both instance-distinction and semantic category information. Secondly, this work proposes to learn pixel embeddings with the triplet loss to query embeddings, which is quite different from previous methods [12]–[14]. Importantly, time-consuming clustering algorithms like mean-shift [33] or K-means [34] are employed in previous methods [12]–[14] to group pixels. However, in our method, instances are proposed with the detection module and pixels are simply assigned by calculating the similarity between pixel embeddings and each query embedding.

B. Panoptic Segmentation

Panoptic segmentation requires to assign a semantic label and an instance id for each pixel in an image. Pixels with the same label and id belong to the same object. The instance id is ignored for stuff labels. Most state-of-the-art methods employ semantic segmentation and instance segmentation modules to tackle this problem, following with fusion steps to merge the two outputs. OANet [3] proposes the Spatial Ranking module to resolve the occlusion between predicted instance masks. Panoptic FPN [2] extends Mask R-CNN [16] with a semantic segmentation branch. AUNet [4] introduces attention modules to promote the performance of stuff classes with the information from the thing branch. A parameter-free panoptic head is proposed in UPSNet [5] to resolves the conflicts in the fusion process by predicting an extra ‘unknown’ class. Porzi et al. [35] propose a segmentation head to integrate the multi-scale features from FPN [36] with contextual information provided by a light-weight, DeepLab-inspired module [37]. AdaptIS [38] generates instance masks from point proposals. Recently, OCFusion [39], BGRNet [40] and BANet [41] propose to improve results with instance occlusion estimation and bidirectional feature aggregation. Li et al. [42] exploit a pairwise instance affinity operation for end-to-end learning. In these methods, the pixel-level semantic classification is provided with semantic segmentation branch as pixel-wise classification scores, and the pixel-level instance-distinction is provided with instance segmentation branch as foreground masks. The instance segmentation branches are mostly two-stage models like Mask R-CNN [16], which shows good performance but is inefficient and memory-consuming. Heuristic rules or additional models are also required to tackle the conflicts in the fusion process, which limits the accuracy and efficiency of the whole system.
Instead of employing two-stage instance segmentation modules like Mask R-CNN [16], some other works attempt to solve panoptic segmentation in a one-stage manner, where the instance-distinction information is provided with pixel-level instance-sensitive features. DeeperLab [6] proposes to solve panoptic segmentation by predicting key-point and multi-range offset heat-maps, following with a grouping process [43]. SSAP [7], [8] proposes to distinguish instances with pixel-pair affinity [30] pyramid and graph-partition method [44]. Panoptic-DeepLab [10] predicts instance centers as well as the offset from each pixel to its corresponding center. PCV [9] emerges instances from pixel consensus on centroid locations and group pixels with back-projection. Hou et al. [11] reuse the predictions from dense object detection with a global self-attention mechanism. Although showing advantages in efficiency, the existing one-stage methods can not achieve comparable accuracy to the state-of-the-art methods.

Unlike previous methods model pixel-level semantic-classification and instance-distinction separately, this work proposes category- and instance-aware pixel embedding to encode the two pixel-level features jointly and resolve the conflicts of thing and stuff classes at the modeling stage.

### III. CATEGORY- AND INSTANCE-AWARE PIXEL EMBEDDING

In this section, we show how to jointly model pixel-level semantic-classification and instance-distinction with category- and instance-aware pixel embedding (CIAE) and introduce the CIAE-loss for learning the pixel embeddings.

#### A. Problem Formulation

Drawing inspiration from instance-aware pixel embedding (IAE) [12]–[14], our CIAE is specifically designed. The original IAE is learned to distinguish different instances. Concretely, each pixel is mapped to a feature space. According to the learned pixel embedding, pixels belong to the same instance are close to each other and pixels belong to different instances are separated with a large margin, as shown in Figure 2(c). However, the learned IAE is category-agnostic. Similarly, we refer to pixel embeddings that can determine semantic categories but are ignorant of instances as category-aware pixel embeddings (CAE). As shown in Figure 2(b), CAE can be realized by mapping pixels into a category-specific sub-space. One-hot encoding for classification can be regarded as a special case of CAE, where the sub-spaces are pre-defined. CIAE is the combination of CAE and IAE, which maps each pixel into the category-specific sub-space while pixels from different instances are distinguishable, as shown in Figure 2(d). In this way, both semantic classification and object distinction are achieved with a single embedding map.

Given the embedding map \( P \), \( p_i \) and \( p_j \) represent the embeddings for pixels \( i \) and \( j \), respectively. In practice, the pixel embeddings are distributed on a hypersphere after L2 normalization:

\[
L2_{\text{norm}}(p_i) = \frac{p_i}{\|p_i\|_2}.
\]

The distance between the two embeddings is measured with cosine similarity:

\[
D_{\cos}(p_i, p_j) = 1 - \frac{p_i \cdot p_j}{\|p_i\| \cdot \|p_j\|}.
\]

Assuming the class numbers for thing and stuff classes are \( C^\text{th} \) and \( C^\text{st} \), respectively. The number of thing instances is \( N_{th} \). \( c_i \in \{0, 1, \ldots, C^\text{st} - 1, \ldots, C^\text{st} + C^\text{th} - 1\} \) represents the semantic category of pixel \( i \). Meanwhile, \( s_i \in \{0, 1, \ldots, C^\text{st} - 1, \ldots, C^\text{st} + N_{th} - 1\} \) represents the segment id of pixel \( i \). Pixels belong to the same stuff class or thing instance are with the same segment id. Note that \( s_i \) and \( c_i \) are the same for pixels of stuff classes.

#### B. CIAE-Loss

In this part, the CIAE-loss for learning CIAE is introduced. As mentioned above, learning CIAE is to map each pixel into the category-specific sub-space while pixels from different instances are distinguishable. For this purpose, the proposed CIAE-loss consists of two sub-losses \( L_{\text{CAE}} \) and \( L_{\text{IAE}} \), which respectively improve the discrimination of embeddings between categories and instances. Details are introduced below.

Unlike previous works [12]–[14] learning IAE embeddings with exact sampled pixels, this work proposes to learn CIAE by optimizing the distances from each pixel embedding to the mean embeddings of segments. Concretely, the mean
embeddings are calculated as the mean directions of pixel embeddings in each segment, following with L2 normalization:

\[ m^k_c = L2\text{norm}\left(\sum_{c_i=k} p_i\right), \]

\[ m^k_s = L2\text{norm}\left(\sum_{s_i=k} p_i\right), \]

\[ M_s = \{m^k_s|k = 0, 1, \ldots, C^{st} + N^{th} - 1\} \text{ and } M_c = \{m^k_c|k = 0, 1, \ldots, C^{st} + C^{th} - 1\} \] are mean embeddings for instance and category id maps, respectively. In order to suppress noise, the mean embeddings are calculated with the embedding map before L2 normalization.

In order to encode CIAE, each pixel embedding is supposed to be close to the correct category and instance mean embedding simultaneously. To this end, two triplet losses are developed to learn the pixel embeddings with \( M_s \) and \( M_c \) as query embeddings, respectively. Given query embeddings, each pixel corresponds to one of them. Using \( M_c \) as example, \( d^p_{pos} \) and \( d^p_{neg} \) are defined as:

\[ d^p_{pos} = D_{\text{cosine}}(p_i, m^c_c), \]

\[ d^p_{neg} = \begin{cases} D_{\text{cosine}}(p_i, m^c_c) & \text{if } k \neq c_i, \\ 0 & \text{if } k = c_i. \end{cases} \]

Given query embeddings \( M \), the embedding loss \( L_{emb} \) is obtained by:

\[ l_{emb}^c = \max\{d^p_{pos} - d^p_{neg} + \text{margin}, 0\}, \]

\[ l_{emb} = -\log \left(1 - \frac{l_{emb}^c}{2 + \text{margin}}\right), \]

\[ L_{emb}(P, M) = \frac{1}{N} \sum_{i} \sum_{k} \mathbb{1}[l_{emb}^k > \tau_k] \cdot l_{emb}^k. \]

\( \mathbb{1}[x] = 1 \) if \( x \) is true and 0 otherwise. The threshold \( \tau_K \) is set in a way that only the pixels with top-K highest losses are selected. We set \( K \) to 5 in experiments. By calculating two losses \( L_{CAE} \) and \( L_{IAE} \), the pixel embeddings learn to be closer to the corresponding category and instance query embeddings simultaneously:

\[ L_{CAE} = L_{emb}(P, M_c), \]

\[ L_{IAE} = L_{emb}(P, M_s). \]

Finally, CIAE-loss is obtained by:

\[ L_{CIAE} = L_{CAE} + \alpha \cdot L_{IAE}. \]

\( \alpha \) is set to 2 to balance the training losses, and we set 5 times loss for pixels with \( \text{thing} \) classes in \( L_{IAE} \).

**C. Category-Aware Memory Embeddings**

Unlike one-hot encoding for classification pre-defining sub-spaces, this work proposes to determine the sub-spaces with learnable category-aware query embeddings.

In order to map pixels into category-specific sub-spaces, during the learning process, the pixel embeddings are supposed to be distinguishable from all other classes instead of only the other classes that included in the mini-batch. To this end, the query embedding of each semantic category is slowly updated during the whole learning process as a memory embedding. Specifically, assuming there are \( T \) iterations in total. \( M_0 \) is randomly initialized for the first iteration. At the \( t \)-th iteration, the mean embeddings \( M_t \) of classes that exist in the current mini-batch are firstly obtained. After that, we calculate an increasing updating momentum:

\[ \lambda_t = 1 - 0.0001 \star (1 - t/T)^3 \]

and update the memory embeddings with:

\[ M_t = L2\text{norm}[\lambda_t \star M_{t-1} + (1 - \lambda_t) \star M_c]. \]

Both \( M_c \) and stuff query embeddings in \( M_s \) are replaced with \( M_{t-1} \) to calculate the CIAE-loss in Eq. 12. In this way, the memory embeddings of semantic categories divide the feature space into category-specific sub-spaces. When learning to be closer to each corresponding memory embedding, the pixel embeddings are better located in target sub-spaces.

**IV. FAST PANOPTIC SEGMENTATION FRAMEWORK**

**A. Network Architecture**

As shown in Figure 3, the proposed model consists of two branches, one for locating instances, the other for predicting pixel-wise embedding features. To locate instances, the feature maps from P3 to P7 FPN levels are fed into a shared proposal head to predict the box classification, center-ness, box regression, and box embeddings. Meanwhile, features from P2 to P5 are used to predict the pixel embeddings. The proposed panoptic segmentation framework is developed upon the state-of-the-art one-stage detector FCOS [27]. Note that it can also be replaced with other detectors [26], [45], [46].

**Proposal head** The proposal head is developed to locate instances. As shown in Figure 3, two branches that consist of four convolutional layers are added after the FPN feature map for classification and regression, respectively. After that, the box classification is predicted from the classification branch while the center-ness, box regression, and embeddings are predicted from the regression branch. The box classification and box regression predict the categories and 4-directional offsets of boxes, respectively. The center-ness predicts the distances to object boundaries to down-weight the low-quality predictions during NMS. We refer readers to FCOS [27] for more details. Unlike FCOS defining all inner regions of bounding boxes as training data for center-ness and box-regression, in our method, location \((x,y)\) is considered positive only if it falls into a ground-truth mask. Different from other outputs, the box embedding is not directly supervised during training, but used to guide the pixel embedding branch, thereby enforcing the objective feature.

**Embedding branch** The features from P2 to P5 are utilized to predict pixel-wise embeddings, which are with the resolutions of 1/4, 1/8, 1/16, and 1/32, respectively. We employ the feature aggregation strategy similar to the semantic segmentation branch in Panoptic FPN [2], where features from P2 to P5 are progressively up-sampled with several \( 3 \times 3 \) convolution, group normalization [47], ReLU, and 2× bilinear up-sampling (discarded for P2) operators. The up-sampled feature maps are at the same 1/4 scale, which are then
to distinguish distant instances requires a large field of view, pixel embeddings from any two instances. However, learning shown the network can be roughly trained for distinguishing from that of other instances. Previous works [12], [13] have network learns to make each pixel embedding distinguishable when optimized with the training loss described as Eq. 12, the learning category- and instance-aware pixel embeddings.

B. Filtering Strategy

In this work, instance differentiating is achieved through learning category- and instance-aware pixel embeddings. When optimized with the training loss described as Eq. 12, the network learns to make each pixel embedding distinguishable from that of other instances. Previous works [12], [13] have shown the network can be roughly trained for distinguishing pixel embeddings from any two instances. However, learning to distinguish distant instances requires a large field of view, making it difficult to optimize the network. To this end, this work proposes a remote instances filtering strategy, based on the instance bounding boxes. Concretely, the similarity between $p_i$ and $m^k_s$ is set to 0 if the pixel lies outside the bounding box of instance $k$. Ground truth and predicted bounding boxes are utilized for training and inference, respectively. Moreover, in the training phase, the ground truth boxes are expanded to 1.5 times to increase the training data and enhance the robustness of imperfect predictions.

C. Inference Process

At the inference time, the PS result is derived by assigning each pixel to a detected instance or a stuff class according to the learned embedding. As shown in Figure 4, given the learned embedding map $P$ and the detected instance proposals after NMS, the image is segmented by comparing the similarity between pixel embeddings $P$ and query embeddings $V$.

As illustrated in Figure 4(d), query embeddings consist of query embedding for each stuff class and each instance proposal. The learned memory embedding is employed as the query embedding for each stuff class, which is consistent with the training process. For the query embedding of each proposal, we simply derive it from the embedding map, according to where each proposal is generated. Specifically, if the proposed instance is detected from the location $(x, y)$, then we use $P_{x,y}$ as the query embedding. Cosine distance is employed to measure the similarity. Since the norm of both pixel embeddings and query embeddings are 1, the cosine similarity can be simply calculated by matrix multiplication:

$$ P \times V^T = S. $$

(16)

Each pixel is then assigned to the instance proposal (or stuff class) with the highest similarity. After that, the semantic cate-
Fig. 4. Panoptic segmentation from pixel embeddings. Panoptic segmentation result is obtained from the pixel embedding map and detected instance proposals (after NMS) by calculating the cosine similarity between pixel embeddings and the query embeddings. Query embeddings consist of query embedding for each stuff class and instance proposal. The former is the memory embedding for each stuff class and the later is derived from the embedding map, according to where each proposal is generated. c) is the visualization result of the learned embedding map with PCA dimensionality reduction. Best viewed in color and zoom.

gory of instance segments can be provided by the learned pixel embedding or box classification branch. The former can be realized by calculating the similarities between each proposal query embedding and category-aware memory embeddings of thing classes. These two approaches are compared through experiments in Table IV (detailed in Subsection V-F). For simplicity and accuracy, we finally determine the semantic category of each instance segment with the box classification result and merge the thing classes as a single class when learning CIAE. Moreover, stuff regions with a size smaller than 4096 are avoided.

V. EXPERIMENTS

A. Dataset

Our experiments are conducted on the large-scale object detection and segmentation dataset COCO [15], which contains 118K and 5K images for training and validation, respectively. The labels consist of 53 stuff classes and 80 thing classes that contain instance level annotation. We finally submit the result of test-dev split (20K images) to the COCO 2019 panoptic segmentation evaluation server.

B. Metric

The results are evaluated with the standard Panoptic Quality (PQ) metric introduced by Kirillov et al. [1]. PQ can be further divided into the multiplication of Segmentation Quality (SQ) term and Recognition Quality (RQ) term to evaluate segmentation and recognition performances, respectively. The formulation of PQ is:

$$PQ = \frac{\sum_{p,g \in TP} \text{IoU}(p,g)}{|TP|} \times \frac{|TP|}{|TP| + \frac{1}{2}|FP| + \frac{1}{2}|FN|}. \quad (17)$$

where \(\text{IoU}(p,g)\) represents the intersection-over-union between the predicted segment \(p\) and ground truth segment \(g\). \(TP\), \(FN\), and \(FP\) represent matched pairs of segments (\(\text{IoU}(p,g) > 0.5\)), unmatched ground truth segments, and unmatched predicted segments, respectively. PQ, SQ, and RQ of both thing and stuff classes are reported in this work.

C. Implementation details

Our experiments are implemented with PyTorch [50] and trained with 8 NVIDIA 2080Ti GPUs. The inference time is measured on 1 NVIDIA 2080Ti GPU with batch size as 1.

Unless specified, ResNet-50 [51] with FPN [36] is utilized as our backbone network. The same hyper-parameters with RetinaNet [46] are utilized. Specifically, our network is trained for 90K iterations with stochastic gradient descent (SGD) and the mini-batch is set to 16. The learning rate is initialized to 0.01 and multiplies a factor of 0.1 at the 60K and 80K iterations, respectively. Weight decay and momentum are set as \(1e^{-4}\) and 0.9, respectively. The backbone network is initialized with the ImageNet [52] pre-trained weights and the newly added layers are initialized as in RetinaNet [46]. Unless specified, we resize the shorter side of the input images to 800 for both training and testing.
D. Main Results

Performance on validation set We summarize the accuracy and speed of our method on the COCO validation set in Table I, where the networks are trained longer (180K iterations) with scale jitter (shorter image side in [640, 800]). We compare the accuracy and efficiency of our method with the state-of-the-art two-stage and one-stage panoptic segmentation algorithms. For inference time, the average end-to-end inference time over the whole validation set is reported. The fusing time for obtaining the final PS result from multiple outputs (in our method: instance proposals and embedding map) is also reported, which is included in the inference time. Our method outperforms all the one-stage methods in both accuracy and inference speed. We are also the first one-stage method that achieves comparable performance against time-consuming two-stage methods.

Performance on test-dev set The comparison with the state-of-the-art methods for panoptic segmentation on the COCO test-dev set are summarized in Table II. The results of our method are predicted with a single model and single inference. No test-time augmentations such as horizontal flipping or multi-scale testing were adopted.

E. Visualization Results

The sampled results from the COCO validation set are visualized in Figure 5. The first row shows the predicted instance proposals and where they are generated. The second row visualizes the learned embedding maps after PCA dimensionality reduction. The third and fourth rows represent the final panoptic segmentation results and the ground truths, respectively. The last three rows show the results of other state-of-art methods [2], [5], [10]. Only single scale inference is used and our model achieves 41.4% PQ while runs at 10 fps.
F. Ablation Experiments

Consistently modeling stuff classes The proposed PS framework models thing and stuff classes consistently with CIAE. To demonstrate the benefits, we conduct an experiment to segment stuff classes with an additional semantic segmentation branch, where an $H/4 \times W/4 \times (N_{st} + 1)$ tensor (thing classes are merged as a single class) is predicted in parallel with $P'$ and supervised with the standard cross-entropy loss. The results are shown in Table III. Similar performances are achieved in mIoU and $PQ_{th}$, however, the performance for $PQ^{th}$ is much lower. This is because the consistent modeling leads to more precise boundaries, which is especially beneficial for small instances.

Proposal categories from CIAE or detector In this part, we test the difference between deriving instance categories through the learned embedding map or the box classification branch. We start by learning specific thing classes with both branches and inferencing with each of them. As shown in the first two rows in Table IV, the box classification branch classifies instance proposals better than the embedding map.
After that, the two branches separately merge *thing* classes as a single class and infer from the other. Experiments show that employing a class-agnostic box classifier reduces PQ by 1.2%. However, since the information to be encoded in CIAE is reduced, merging *thing* classes in learning CIAE promotes PQ<sup>st</sup> with 0.5% and obtains slightly better performance. This setting is used as the default because of its accuracy and simplicity, where the *thing* classes are merged as a single class when learning CIAE and the specific category of each instance segment is determined with the box classification result. Moreover, from the last two rows in Table IV, we find that if the category information for *thing* classes is completely abandoned (realized by setting \( L_{CAE} \) to 0) in CIAE, PQ degrades with 0.8%. This result supports our motivation to encode pixel-wise semantic-classification and instance-distinction simultaneously. Specifically, by adding the CAE loss, the model learns CIAE instead of IAE, and locates the embedding vectors for *thing* instances into a specific subspace, thereby making the network easier for optimization, and finally boosts the performance for both *thing* and *stuff* classes consistently. Importantly, the improvements are achieved only by adding a constraint during training, which does not introduce additional parameters and can boost the performance at no cost.

**Oracle performance** We test the upper limits of the proposed method by employing predicted pixel embeddings and ground-truth (GT) bounding boxes for inference. The query embedding of each instance is derived by calculating the mean embedding within the GT mask. The results are shown in Table V. Employing GT bounding boxes nearly doubles the performance of *thing* classes, which shows the great potential of the proposed pixel embedding based framework.

**Box embedding and CoordConv** As described in Subsection IV-A, we propose employing CoordConv and fusing the box embeddings from each FPN level to the embedding branch to enhance the objective information. Both quantitative and qualitative analysis of this structure is carried out. The proposed box embedding structure is compared with the one where the box embedding branch is abandoned. In other words, \( P = f(P^I + P^O) \) is replaced with \( P = f(P^I) \), where \( f \) denotes \( 4 \times \) up-sampling and L2 normalization. As shown in Table VI, fusing box embeddings boosts the performance from 37.1% to 37.8% PQ. Both performances of *thing* and *stuff* classes are improved. The visualization results in Figure 6 indicate that more smooth and complete instance segments are generated after fusing the box embeddings to the embedding branch.
Influence of the filtering strategy. The first row shows the performance of abandoning filtering strategy.

| Box | PQ (%) | PQ$^1h$ (%) | PQ$^1t$ (%) |
|-----|--------|-------------|-------------|
| All | 36.6   | 41.1        | 29.8        |
| 2.0×| 37.4   | 42.1        | 30.4        |
| 1.5×| 37.9   | 42.5        | 30.9        |
| 1.2×| 37.8   | 42.3        | 30.8        |
| 1.0×| 36.6   | 40.8        | 30.1        |

**TABLE IX**

Results of using different triplet loss margins.

| Margin | PQ (%) | PQ$^1h$ (%) | PQ$^1t$ (%) |
|--------|--------|-------------|-------------|
| 0.30   | 36.5   | 39.9        | 30.9        |
| 0.20   | 37.7   | 41.9        | 31.0        |
| 0.15   | 37.9   | 42.5        | 30.9        |
| 0.10   | 37.6   | 42.5        | 30.2        |
| 0.00   | 28.7   | 36.2        | 17.2        |

**TABLE X**

Results of using different embedding dimensions.

| Dim  | PQ (%) | PQ$^1h$ (%) | PQ$^1t$ (%) |
|------|--------|-------------|-------------|
| 48   | 37.9   | 42.5        | 31.0        |
| 32   | 37.9   | 42.5        | 30.9        |
| 24   | 37.8   | 42.5        | 30.6        |
| 16   | 37.3   | 42.2        | 30.0        |

Memory updating momentum As described in Subsection III-C, the category-aware memory embeddings are gradually updated and finally employed as query embeddings at the inference process. We set a large momentum value to stabilize the training process. As shown in Table VII, the network learns well with a large momentum value. When the momentum is set to 0, the training loss oscillates and fails to converge.

Filtering strategy As described in Subsection IV-B, this work proposes a box based filtering strategy to eliminate distant instances from optimizing, and the ground truth boxes are expanded during training. As shown in Table VIII, expanding the ground truth boxes to 1.5 times achieves the best performance. The first row shows the performance drops with 1.3% PQ when the filtering strategy is abandoned, which represents the similarities to all instance mean embeddings are optimized, and the ground truth boxes are simply predicted by assigning each pixel to a detected instance.

Triplet loss margin This work proposes to learn CIAE by optimizing the triplet losses with category and instance query embeddings. Here we test the performances for different margins. Note that the margins in $L_{CIAE}$ and $L_{IAE}$ can be different, but we set them the same for simplifying. As shown in Table IX, the model performs best with $margin = 0.15$. It is difficult to optimize the network with an excessive large margin, and the learned embeddings lack discrimination when the margin is too small.

Embedding dimension Table X shows the results of using different embedding dimensions. We found the embedding dimension only slightly affects the performance for stuff classes, and the model works well even trained with 16 dimension embeddings.

VI. Conclusion

In this paper, we have introduced a novel one-stage panoptic segmentation method that consistently models thing and stuff classes. Our method is based on incorporating a one-stage detector with the proposed category- and instance-aware pixel embedding (CIAE). CIAE is a pixel-wise embedding feature that encodes both semantic-classification and instance-distinction information. The panoptic segmentation result is simply predicted by assigning each pixel to a detected instance or a stuff class according to the learned embedding. Our method has a fast runtime speed and is also the first one-stage method that achieves comparable performance against two-stage methods on the challenging COCO benchmark.

REFERENCES

[1] A. Kirillov, K. He, R. Girshick, C. Rother, and P. Dollar, “Panoptic segmentation,” in CVPR, 2019. 1, 6
[2] A. Kirillov, R. Girshick, K. He, and P. Dollar, “Panoptic feature pyramid networks,” in CVPR, 2019. 1, 2, 4, 5, 7, 8
[3] H. Liu, C. Peng, C. Yu, J. Wang, X. Liu, G. Yu, and W. Jiang, “An end-to-end network for panoptic segmentation,” in CVPR, 2019. 1, 2, 7
[4] Y. Li, X. Chen, Z. Zhu, L. Xie, G. Huang, D. Du, and X. Wang, “Attention-guided unified network for panoptic segmentation,” in CVPR, 2019. 1, 2, 7
[5] Y. Xiong, R. Liao, H. Zhao, R. Hu, M. Bai, E. Yumer, and R. Urtasun, “UPSN: A unified panoptic segmentation network,” in CVPR, 2019. 1, 2, 7, 8
[6] T.-J. Yang, M. D. Collins, Y. Zhu, J.-J. Huang, T. Liu, X. Zhang, V. Sze, G. Papandreou, and L.-C. Chen, “DeeperLab: Single-shot image parser,” arXiv:1902.05093, 2019. 1, 3, 7
[7] N. Gao, Y. Shan, Y. Wang, X. Zhao, Y. Yu, M. Yang, and K. Huang, “SSAP: Single-shot instance segmentation with affinity pyramid,” in ICCV, 2019. 1, 2, 3, 7
[8] N. Gao, Y. Shan, Y. Wang, X. Zhao, and K. Huang, “SSAP: Single-shot instance segmentation with affinity pyramid,” IEEE TCSVT, vol. 31, no. 2, pp. 661–673, 2021. 1, 3
[9] H. Wang, R. Luo, M. Maire, and G. Shakhnarovich, “Pixel consensus voting for panoptic segmentation,” in CVPR, 2020. 1, 3, 7
[10] B. Cheng, M. D. Collins, Y. Zhu, T. Liu, T. S. Huang, H. Adam, and L.-C. Chen, “Panoptic-DeepLab: A simple, strong, and fast baseline for bottom-up panoptic segmentation,” in CVPR, 2020. 1, 3, 7, 8
[11] R. Hou, J. Li, A. Bhargava, A. Raventos, V. Guizzilini, C. Fang, J. Lynch, and A. Gaidon, “Real-time panoptic segmentation from dense detections,” in CVPR, 2020. 1, 3, 7
[12] B. D. Brabandere, D. Neven, and L. V. Gool, “Semantic instance segmentation with a discriminative loss function,” arXiv:1706.02551, 2017. 1, 2, 3, 5
[13] A. Fathi, Z. Wojna, V. Rathod, P. Wang, H. O. Song, S. Guadarrama, and K. P. Murphy, “Semantic instance segmentation via deep metric learning,” arXiv:1703.10277, 2017. 1, 2, 3, 5
[14] S. Kong and C. C. Fowlkes, “Recurrent pixel embedding for instance grouping,” in CVPR, 2018. 1, 2, 3
[15] T. Lin, M. Maire, S. J. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollar, and C. L. Zitnick, “Microsoft COCO: Common objects in context,” in ECCV, 2014. 2, 6
[16] K. He, G. Gkioxari, P. Dollar, and R. Girshick, “Mask R-CNN,” in ICCV, 2017. 2, 3
[17] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards real-time object detection with region proposal networks,” in NIPS, 2015. 2
[18] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, “Path aggregation network for instance segmentation,” in CVPR, 2018. 2
[19] L. Chen, A. Hermans, G. Papandreou, F. Schroff, P. Wang, and H. Adam, “MaskLab: Instance segmentation by refining object detection with semantic and direction features,” in CVPR, 2018. 2
[20] Z. Huang, L. Huang, Y. Gong, C. Huang, and X. Wang, “Mask scoring R-CNN,” in CVPR, 2019. 2
[21] K. Chen, W. Ouyang, C. C. Loy, D. Lin, J. Pang, J. Wang, Y. Xiong, X. Li, S. Sun, W. Feng et al., “Hybrid task cascade for instance segmentation,” in CVPR, 2019. 2
