InstanceNet: object instance segmentation using DNN
—— CSI6900 Project

Yuan Gao
ygao133@uottawa.ca

March 31, 2020

Abstract

One-stage object detectors like SSD [1] and YOLO [2] are able to speed up existing two-stage detectors like Faster R-CNN [3] by removing the object proposal stage and making up for the lost performance in other ways. Nonetheless, the same approach is not easily transferable to instance segmentation task. Current one-stage instance segmentation methods can be simply classified into segmentation-based methods which segment first then do clustering, and proposal-based methods which detect first then predict masks for each instance proposal. Proposal-based methods always enjoy a better mAP; by contrast, segmentation-based methods are generally faster when inferencing. In this work, we first propose a one-stage segmentation-based instance segmentation solution, in which a pull loss and a push loss are used for differentiating instances. We then propose two post-processing methods, which provide a trade-off between accuracy and speed.

Introduction

With the continuous decline in hardware costs and the rapid increase in the amount of data collected, the concept of deep learning, which was proposed in the 1980s, has spread all over the globe overnight. And this tipping point happened in the field of computer vision. Alexnet [4] reduced the top-5 error rate by ten percent in the ImageNet image classification competition compared with the previous year's champion. After the surprising success of Alexnet, deep learning has achieved unprecedented results in more and more fields, and it has surprised people time and time again on many tasks that were previously considered impossible by computers. Nowadays, deep learning has penetrated deeply into every aspect of people's lives, such as speech recognition [5], Language translation [6], esports [7].

In the computer vision field, four types of tasks are most commonly used in benchmarks and real-life tasks. They are image classification, object detection, semantic segmentation, and instance segmentation. The task of image classification requires binary labels indicating whether objects are present in an image; see Figure 1(a). Currently, the ImageNet [8] dataset contains over 14 million labeled images and has enabled significant advances in image

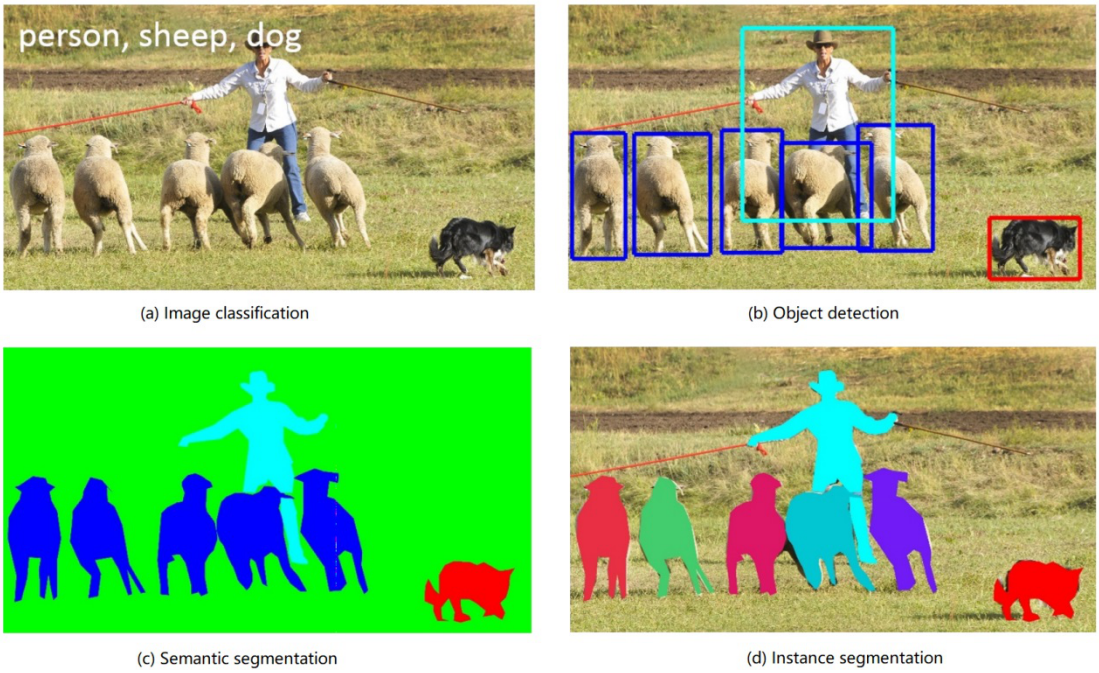


Figure 1: A demo image from MS COCO dataset [9]. 4 types of tasks that are the most commonly used in benchmarks and real life tasks: (a)Image classification, (b)Object detection, (c)Semantic segmentation, (d)Instance segmentation

classification. Object detection entails both stating that an object belonging to a specified class is present, and localizing it in the image with a squared bounding box; see Figure 1(b). The task of semantic segmentation requires that each pixel of an image be labeled as belonging to a category. Compared to the aforementioned object detection task, individual instances of objects do not need to be segmented, Figure 1(c). Instance segmentation as the most challenging task can be seen as a combination of object detection and semantic segmentation, where pixels are labeled using per-instance segmentations; see Figure 1(d).

In this work, we focus on instance segmentation task. As the most challenging one among four classic computer vision tasks, instance segmentation has both the characteristics of semantic segmentation, which is pixel-level classification and of object detection, localizing and differentiating different instances, even though they are of the same class. The well known Mask R-CNN [10] is the first outstanding instance segmentation model using deep neural network, which extends classic object detection work — Faster R-CNN by adding an additional branch for predicting object masks in parallel with the existing branch for bounding box regression and class prediction. Nonetheless, Mask R-CNN, as a two-stage detector, suffers from its low inferencing speed. In some real-time applications, however, speed is critical. For instance, Autonomous cars have to make decisions in a limited time to avoid wrong actions.

Our work focus on one-stage instance segmentation. Inspired by previously proposed Associative Embedding [11] mainly for multiperson pose estimation, we introduce a pull loss and a push loss to distinguish instances belonging to the same class. Pull loss works as a drawing-force that draws embeddings together and push loss works as a

repelling-force that pushes embeddings apart from each other. By doing so, pixels belonging to the same instance are separately tagged.

Related works

Researches on one-stage instance segmentation has two main directions: proposal-based and segmentation-based. Proposal-based approaches detect objects first then predict corresponding masks. Segmentation-based approaches, however, perform semantic segmentation of the image first, then clustering them into different object instances with particularly designed instance-aware features.

One-stage proposal-based methods:

Fully Convolutional Instance-aware Semantic Segmentation [12] FCIS is the first fully convolutional end-to-end solution for instance segmentation task. The work uses a region proposal network (RPN) to generate proposals. To introduce translation-variant property, a k x k position-sensitive score map that corresponds to k $\times$ k evenly partitioned cells of objects is introduced to encode relative position information. Each score represents the likelihood of the pixel belongs to some object instance at a relative position in terms of the overall image. The final instance mask is then produced by assembling its k $\times$ k cells from the corresponding score maps. In this way, a pixel can have different scores in different instances as long as the pixel is at different relative positions in the instances.

YOLACT Real-time Instance Segmentation [13] Similar to typical anchor-based object detectors YOLACT has two branches in the prediction heads: one branch to predict c class confidences, and the other to predict 4 bounding box regressors. For mask coefficient prediction, the work simply adds a third branch in parallel that predicts k mask coefficients, one corresponding to each prototype. YOLACT breaks up instance segmentation task into two parallel ones: first generating an array of k global prototype masks over the entire image, second predicting a set of linear combination coefficients per instance. Then producing a full-image instance segmentation from these two components: for each instance, linearly combine the prototypes using the corresponding predicted coefficients and then crop with a predicted bounding box.

One-stage segmentation-based methods:

SSAP: Single-Shot Instance Segmentation With Affinity Pyramid [14] SSAP introduces a pixel-pair affinity pyramid, which computes the probability that two pixels belong to the same instance in a hierarchical manner. The work jointly learns instance-agnostic semantic segmentation labels and instance-aware affinity pyramid in a unified model. For each pixel, a small r $\times$ r window is learned, where each grid in the window corresponds to a pixel in its surrounding and the value represents the likelihood that two pixels belong to the same instance.

Instance Segmentation by Jointly Optimizing Spatial Embeddings and Clustering Bandwidth [15] The work learns an offset vector for each pixel, pointing to its object's center. To locate the object's centers, the

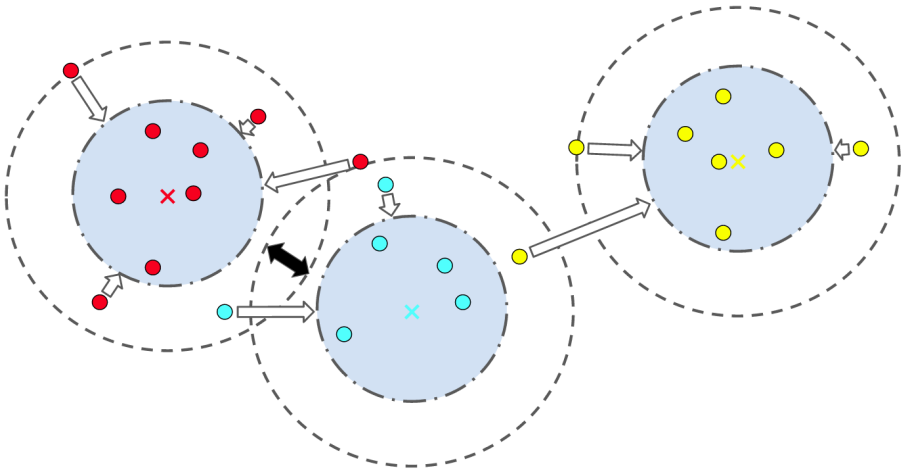


Figure 2: The white arrows indicate the intra-cluster pulles loss which pull embeddings belonging to the same instance towards their embeddings centers. The black double-headed arrow indicate the inter-cluster push losses which pushes clusters that are close to each other apart. The length of these arrows indicate the power of the force. The pull loss and push loss are only activated when the distances reach a margin (hinge loss). These margins are called δ_{pull} for pull loss and δ_{push} for push loss, which are denoted by dotted circles.

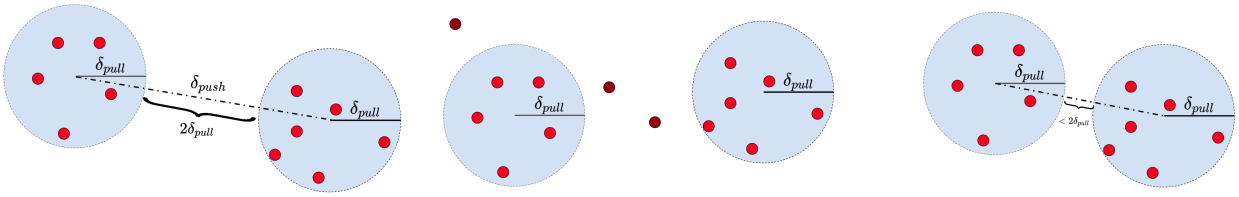
work learns a seed map for each semantic class. Unlike the standard regression approach, the work also learns an optimal clustering region for each object and by doing so the losses are relaxed for pixels far away from the center. At inference time, the procedure is to sample the pixel with the highest value in the seed map and use that location as instance center. At the same location, the sigma value indicates the region of the object is then taken. By using this center and accompanying sigma, the pixels are clustered into one instance. The work next mask out all clustered pixels in the seed map and continue sampling until all pixels in the seed map are masked.

Method

Overview

The naive idea of our proposed work for clustering pixels belonging to the same instance together is to predict an embedding for each pixel in addition to the semantic segmentation score. The pixels with similar embeddings should be grouped together and otherwise be separated into different instances. We focus on not the absolute value of the embeddings, instead, the distances between embeddings are only concerns.

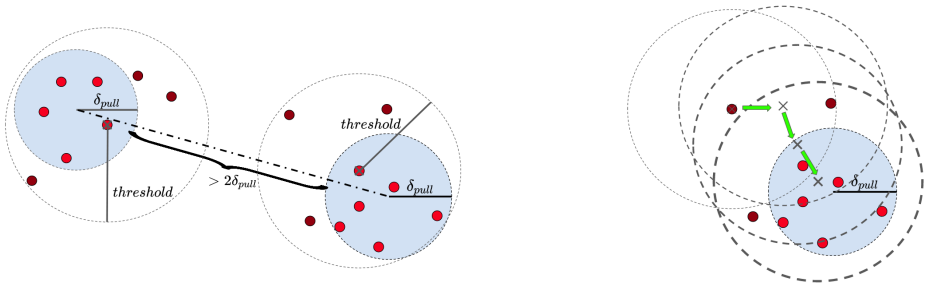
To achieve this, we impose a pull loss and push loss on the predicted embeddings. The pull loss assesses how well the predicted embeddings belonging to one instance are grouped together. Simultaneously, the push loss assesses how well the predicted embeddings not belonging to one instance are clearly separated. Specifically, we retrieve the predicted embeddings for pixels at ground truth locations for a certain class; we then calculate the mean values of all embeddings at locations within each instance. Pull loss for each ground instance is then calculated by the sum of the distances between each embedding within the instance and its mean embedding values. Push loss is achieved



(a) Ideal case:
where $\delta_{push} = 4\delta_{pull}$ and $distance = 2\delta_{pull}$ Blue background circles indicate ground truth embedding clusters

(b) Actual case 1:
where some embeddings lay outside of δ_{pull} Darker points indicate embeddings lay outside of ground truth clusters

(c) Actual case 2:
where distance between clusters $< 4\delta_{pull}$



(d) Post-processing method 1:
using thresholds satisfying $2\delta_{pull} < threshold < 0.5\delta_{push}$. Black crosses indicate randomly selected embeddings, dotted circles represent the boundary of selected clusters

(e) Post-processing method 2:
a varient of k-means is used. The outlier embedding on the top-left corner is randomly selected at frist and the boundary of the cluster is illustrated by a thin-edged circle. Green arrows indicate the shift of the center of clusters and the thickness of the circle becomes stronger gradually.

Figure 3: First row illustrates different scenarios of the output of the network. Second row illustrates two post-processing method, which provide a trade-off between time efficiency and accuracy. Method 1 is fast but less accurate and method 2 is more accurate but more time consuming.

by a fixed number subtracted by the distance between 2 instances and we apply push loss to all combinations of 2 different instances, see Figure 2.

Network architecture

In this work, we combine embedding prediction with the well-performed semantic segmentation network DeepLabv3+ [16], which employs an Atrous Spatial Pyramid Pooling(ASPP) structure used for expanding conceptional field and a decoder structure for improving segmentation boundary accuracy. We increase the number of final output channels from C to $C + K$, where C denotes the number of classes and K is the length of the embedding. In other words, in parallel with the output of semantic segmentation labels, we add an embedding prediction head. All layers before the final output layer are shared for semantic segmentation and embedding prediction.

Loss function

The intuition of our loss function is that embeddings located within the same instance should end up close to each other, while embeddings belonging to different instances should end up far apart. Alejandro et al.[1] propose

a grouping loss function, which has two terms: one to minimize the sum of squared errors between all values of embeddings within each instance and mean value of these embeddings; and another term to maximize the distance between the embedding means of all possible pairs of any two instances. In our loss function, we follow a similar idea and name the loss for drawing embeddings together as pull loss and the loss for pushing embedding means away as push loss.

Pull loss

Pull loss can be seen as an intra-instance force that draws embeddings towards the embedding mean. Different from the loss used by [11] we slightly relax the restriction of pulling all embeddings towards a single ground-truth value, which increases model stability and makes the weighs easier to be trained. The goal of the pull loss is modified to draw embeddings into a circle with mean embeddings as the center and δ_{pull} as the radius. When the embeddings fall in the circle, it will not contribute to the loss anymore.

Push loss

Push loss can be seen as an inter-cluster force that pushes clusters away from each other, increasing the distance between the cluster centers (embedding means). Like pull loss, the push losses are also hinged, the losses are only activated when the distance between two cluster centers are close to some extend. We set a pre-defined δ_{push} for the push loss.

Regularization loss

Regularization loss is introduced as a small pull-force that draws all cluster centers towards the origin, to keep the model stable.

Loss function

The total loss can be written as a combination of three losses above. To be specific, the overall per-class loss function is a weighted sum of the pull loss, push loss and regularization loss:

$$L_{pull} = \frac{1}{K}\sum_{k=1}^{K}\frac{1}{N_k}\sum_{i=1}^{N_k}\mathbb{1}_{i\in S_k}[||\mu_k - e_i||_n - \delta_{pull}]_+ \tag{1}$$

$$L_{push} = \frac{1}{K(K-1)}\sum_{\substack{k_1=1\\k_1\neq k_2}}^{K}\sum_{k_2=1}^{K}[2\delta_{push} - ||\mu_{k_1} - \mu_{k_2}||_n]_+ \tag{2}$$

$$L_{regularization} = \frac{1}{K}\sum_{k=1}^{K}||\mu_k||_n \tag{3}$$

$$L_c = \alpha \cdot L_{pull} + \beta \cdot L_{push} + \gamma \cdot L_{regularization} \tag{4}$$

$$L_{total} = \sum_{c=1}^{C} L_c \tag{5}$$

In the functions above: K is the number of ground truth instances belonging to one class, N_k is the number of pixels of instance k, S_k is the set of pixels of instance k, e_i is the embedding of pixel at location i, μ_k is the mean embedding of instance k (the cluster center), $||\cdot||_n$ is L_n norm distance, and $[x]_+ = \max(0, x)$ denotes the hinge. δ_{pull} and δ_{push} are the thresholds for the hinged pull and push loss respectively. C is the number of classes, L_c denotes the loss for class c.

Post processing

First we consider the ideal situation where all embeddings are fallen within a distance of δ_{pull} from their corresponding embedding centers and all embedding centers are at least d apart from each other, where $\delta_{push} = 4\delta_{pull}$. Under this scenario, every embedding lays closer to embeddings belonging to its own cluster than to any other embeddings belonging to other clusters, see Figure 3(a). When inferencing, after semantic segmenting, we can first randomly choose a pixel of a certain class, then use a threshold of $2\delta_{pull}$ and all pixels with embeddings within this distance from the embedding of the chosen pixel belong to the same instance. Then we tag these pixels and run the process again on all un-tagged pixels until all pixels of this class are tagged. Likewise, We repeat this random choice, threshold and tag process for all classes. It should be noted that embeddings are class-agnostic, which means all classes share the same group of embeddings.

In practice, however, things will not be such perfect. Case 1: Embeddings belonging to the same instance will not always ideally lay within the radius of ground truth embedding centers, see Figure 3(b). Case 2: The distances between embedding centers could be less than 4 times of the radius of each embedding cluster, Figure 3(c).

To handle this issue, the first thing we can do is to increase the difference between d and v when training and to use a threshold for tagging larger than $2\delta_{pull}$ but still less than $0.5\delta_{push}$ when inferencing (Figure 3(d)). By using a larger distance between $2\delta_{pull}$ and $0.5\delta_{push}$, clusters are easier to differentiate. The robustness is therefore increased when an embedding close to an embedding center is chosen.

However if we unfortunately selected an outlier to begin thresholding, it still could happen that a real cluster is clustered into two sub-clusters and embeddings belonging to two adjacent clusters are mis-clustered into one. To further alleviate this problem, we introduce a variant of k-means algorithm. As before, we randomly select an un-tagged pixel and do thresholding. Next, we calculate the embedding mean of the selected group of embeddings and use the mean as the new center to do thresholding again. The process keeps until the mean is converged. This process can gradually draw the cluster of selected embeddings from a low-density area (which is near the periphery of a ground truth embedding cluster) to a high-density area, which is closer to the ground truth embedding cluster area. The process is described in Figure 3(e).

semantic segmentation	post-processing	AP
DeepLabv3+	center-fixed	23.4
DeepLabv3+	center-shift	24.9
ground truth mask	center-fixed	30.6
ground truth mask	**center-shift**	**31.7**

Table 1: Evaluation on COCO val2017 split. Different semantic segmentation proposal methods and post-processing methods are used. A big leap can be seen by replacing the semantic segmentation proposal from DeepLabv3+ to the ground truth mask, which is shown on the 1st and the 3rd row.

threshold	AP	threshold	AP
0.6	24.6	1.4	31.6
0.8	27.4	1.6	30.9
1.0	30.8	1.8	29.6
1.2	**31.7**	2.0	29.1

Table 2: Influence of using a different value of threshold for grouping when inferencing. Ground truth mask and center-shift method are also compared. The AP increases first and then decreases as the threshold goes up.

Experiments

Training details

We choose the large-scale object detection benchmark MS COCO dataset [9], and use the train2017 split (115K images) for training and val2017 split(5K images) for reporting results. The input images are resized with the longer side being 513 while the aspect ratio keeps. Only left-right random flip with a probability of 0.5 is used for data augmentation. DeepLabv3+ with 50-layer Resnet [17] pretrained on Pascal VOC dataset [18] is borrowed as off-the-shelf semantic segmentation base network. Embedding prediction branch is randomly initialized with xavier initializer [19]. We first fix the Resnet and train the ASPP and Decoder structure of DeepLabv3+ with learning rate of 0.001 for 80 epochs and train all parts for an extra 60 epochs with learning rate of 0.0001. We train the model with Adam optimizer [20]. Batch normalization is used with weights decay of 0.99 and with batch size set as 4 on single RTX2060s graphic card.

The dimension of embeddings is set to 4 for MS COCO challenge. In terms of loss function, we set $\alpha = 1$, $\beta = 1$ and $\gamma = 0.001$. In our experiment we set the margin for hinge loss as $\delta_{pull} = 0.2$ and $\delta_{push} = 3$. When inferencing, threshold set as 1.2 gives the best result.

Results

Our proposed method can be splited into two sub-tasks: semantic segmentation task and grouping task. The semantic segmentation task solely depends on DeepLabv3+ network and the grouping task depends on embedding prediction task and post-processing task. Consequently, the performance of grouping highly depends on the task of semantic segmentation, as if a pixel is misclassified, there will be no remedial measures for grouping it correctly.

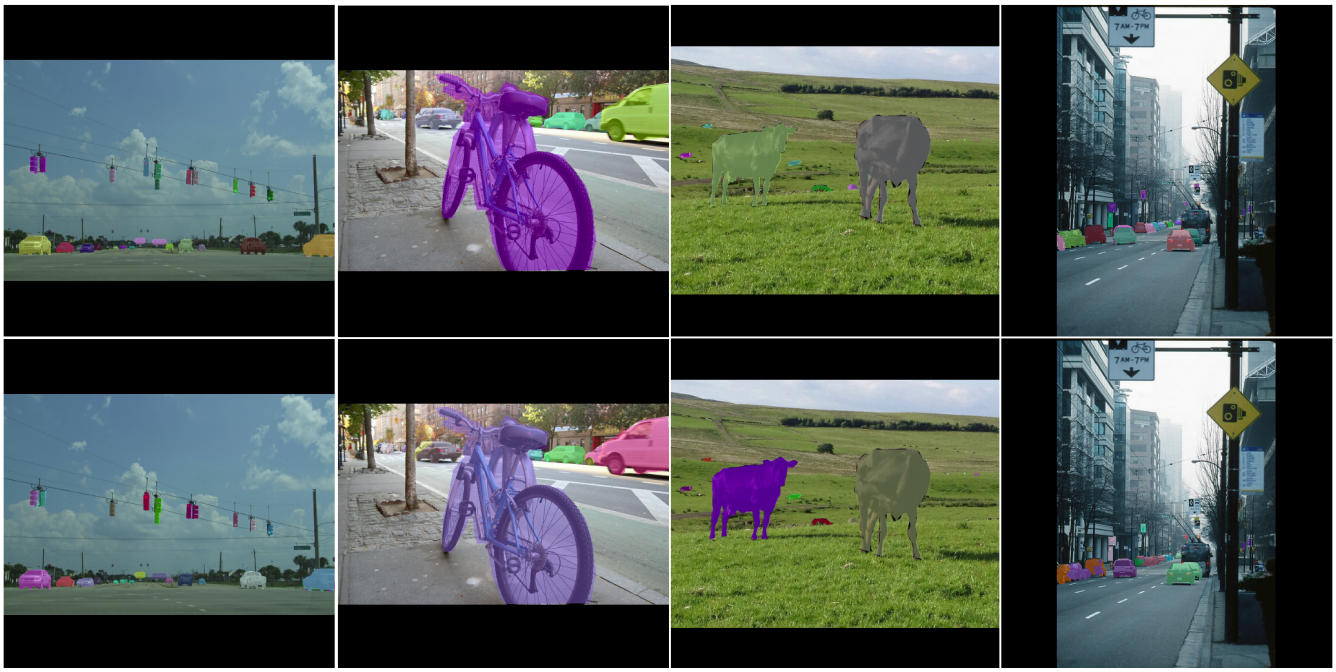


Figure 4: Some examples on MS COCO val2017 split. The first row demonstrates the ground truth and the second row illustrates corresponding instance segmentation results based on ground truth semantic segmentations and center-shift post-processing.

Therefore, it is reasonable that we evaluate the grouping task separately since this sub-task embodies the core idea of our proposed work.

We first evaluate the performance of the overall network(DeepLabv3+) with post-processing method 1 (center-fixed). Then we replace the post-processing method with method 2 (center-shift). After that, we also report performance for post-processing method 1 and 2 based on ground truth semantic segmentation results. The results of all experiments are recorded in Table 1. It is obvious that the performance of the overall network can be improved by a better semantic segmentation output. Table 2 illustrates the influence of using a different threshold value for grouping when inferencing. Ground truth mask and center-shift method are used in this experiment. The performance improved first while increasing the value of the threshold, as more embeddings belonging to the same instances are correctly clustered. After passing the peak, the performance begins to drop as the value of the threshold keeps increasing. The reason is that too many embeddings are allocated into one cluster, even when they are far from each other. Some final results on COCO val2017 can be seen in Figure 4, where the first row demonstrates the ground truth and second row are corresponding instance segmentation results based on ground truth semantic segmentations.

Conclusion

In this work, we first proposed a method for differentiating instances among semantic segmentation belonging to the same class. The network predicts an embedding for each pixel in the image. Then a pull loss that draws embeddings belonging to the same instances and a push loss that forces embeddings belonging to different instances move apart are applied. When inferencing, two post-processing methods are introduced, which provide a trade-off between time efficiency and accuracy. The center-fixed method uses a fixed threshold and therefore enjoys a fast speed; the center-shift method, however, is more accurate but needs more time. Compared to SOTA counterparts, our work still has much room for improvement. Some possible refinement includes using larger batch size on multiple GPUs, using larger input image size and better semantic segmentation proposal network.

References

[1] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *The European Conference on Computer Vision (ECCV)*, 2016, vol. 9905.

[2] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 779–788.

[3] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems 28*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds. Curran Associates, Inc., 2015, pp. 91–99. [Online]. Available: http://papers.nips.cc/paper/5638-faster-r-cnn-towards-real-time-object-detection-with-region-proposal-networks.pdf

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105. [Online]. Available: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

[5] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, Nov 2012.

[6] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, p. 1735–1780, Nov. 1997. [Online]. Available: https://doi.org/10.1162/neco.1997.9.8.1735

[7] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. P. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of go with deep neural networks and tree search," pp. 484–489, 2016.

[8] J. Deng, W. Dong, R. Socher, L. Li, Kai Li, and Li Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, June 2009, pp. 248–255.

[9] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. Zitnick, "Microsoft coco: Common objects in context," in *The European Conference on Computer Vision (ECCV)*, 2014.

[10] K. He, G. Gkioxari, P. Dollar, and R. Girshick, "Mask r-cnn," in *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[11] A. Newell, Z. Huang, and J. Deng, "Associative embedding: End-to-end learning for joint detection and grouping," in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Curran Associates, Inc., 2017, pp. 2277–2287. [Online]. Available: http://papers.nips.cc/paper/6822-associative-embedding-end-to-end-learning-for-joint-detection-and-grouping.pdf

[12] Y. Li, H. Qi, J. Dai, X. Ji, and Y. Wei, "Fully convolutional instance-aware semantic segmentation," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[13] D. Bolya, C. Zhou, F. Xiao, and Y. J. Lee, "Yolact: Real-time instance segmentation," in *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.

[14] N. Gao, Y. Shan, Y. Wang, X. Zhao, Y. Yu, M. Yang, and K. Huang, "Ssap: Single-shot instance segmentation with affinity pyramid," in *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.

[15] D. Neven, B. D. Brabandere, M. Proesmans, and L. V. Gool, "Instance segmentation by jointly optimizing spatial embeddings and clustering bandwidth," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[16] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *The European Conference on Computer Vision (ECCV)*, September 2018.

[17] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[18] M. Everingham, L. V. Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International Journal of Computer Vision*, vol. 88, pp. 303–308, September 2009, printed version publication date: June 2010. [Online]. Available: https://www.microsoft.com/en-us/research/publication/the-pascal-visual-object-classes-voc-challenge/

[19] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *In Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS'10). Society for Artificial Intelligence and Statistics*, 2010.

[20] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *International Conference on Learning Representations*, 12 2014.